CRC Coding

By: Sourav Kumar Mandal(20JE0970)

& Gyanam Modi(20JE0388)

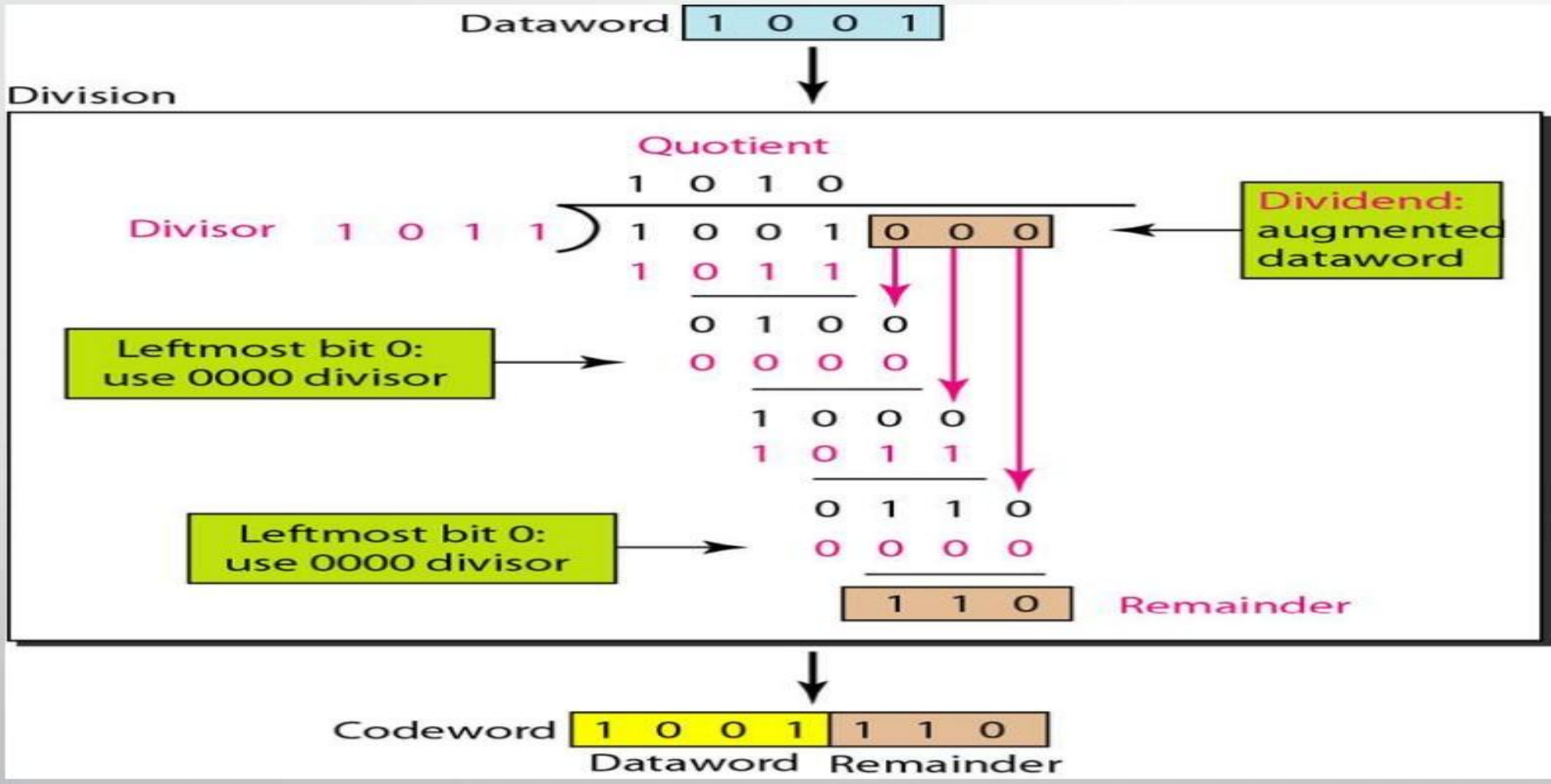
Introduction

- 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.

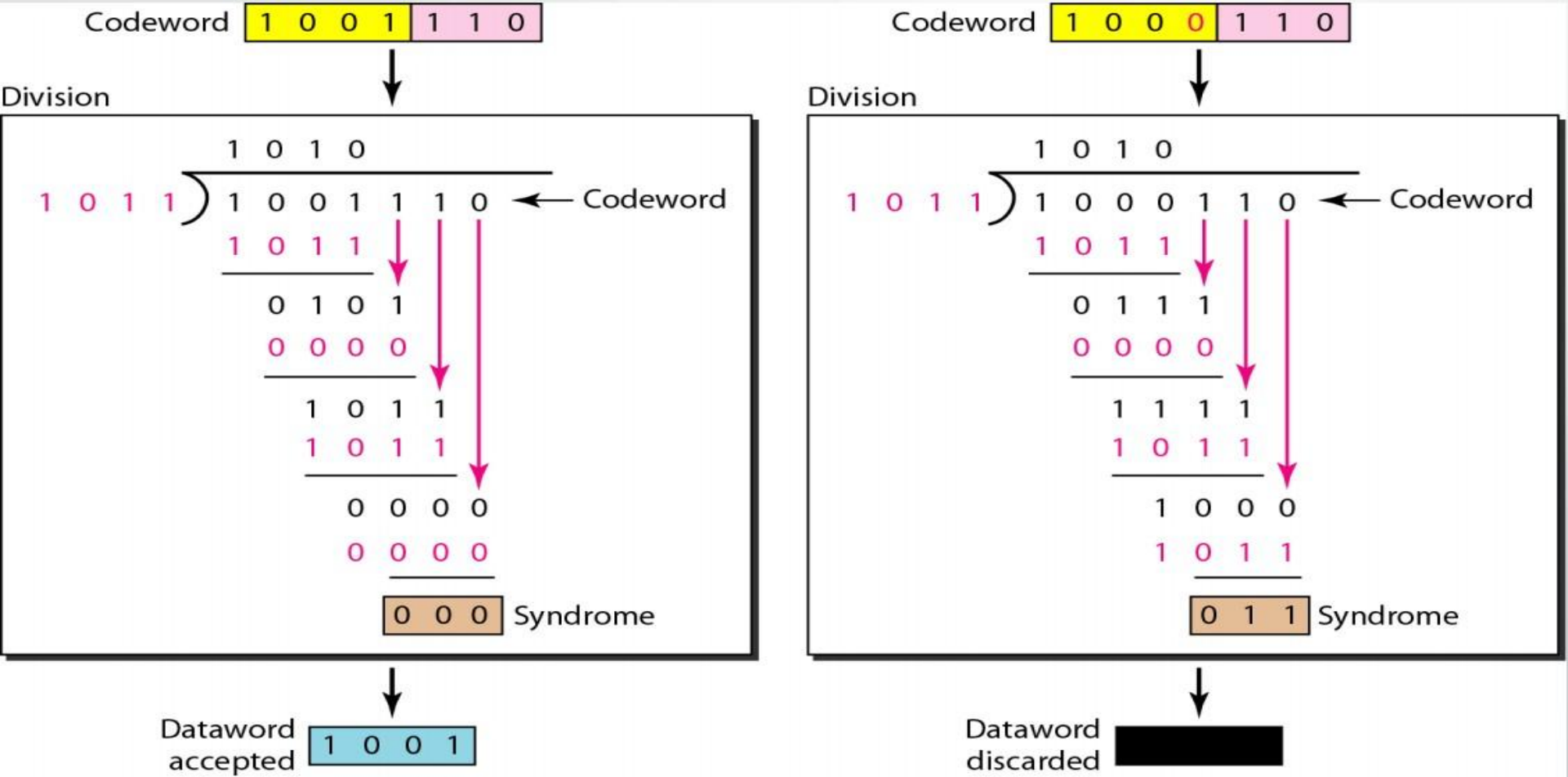
What is CRC?

- 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.

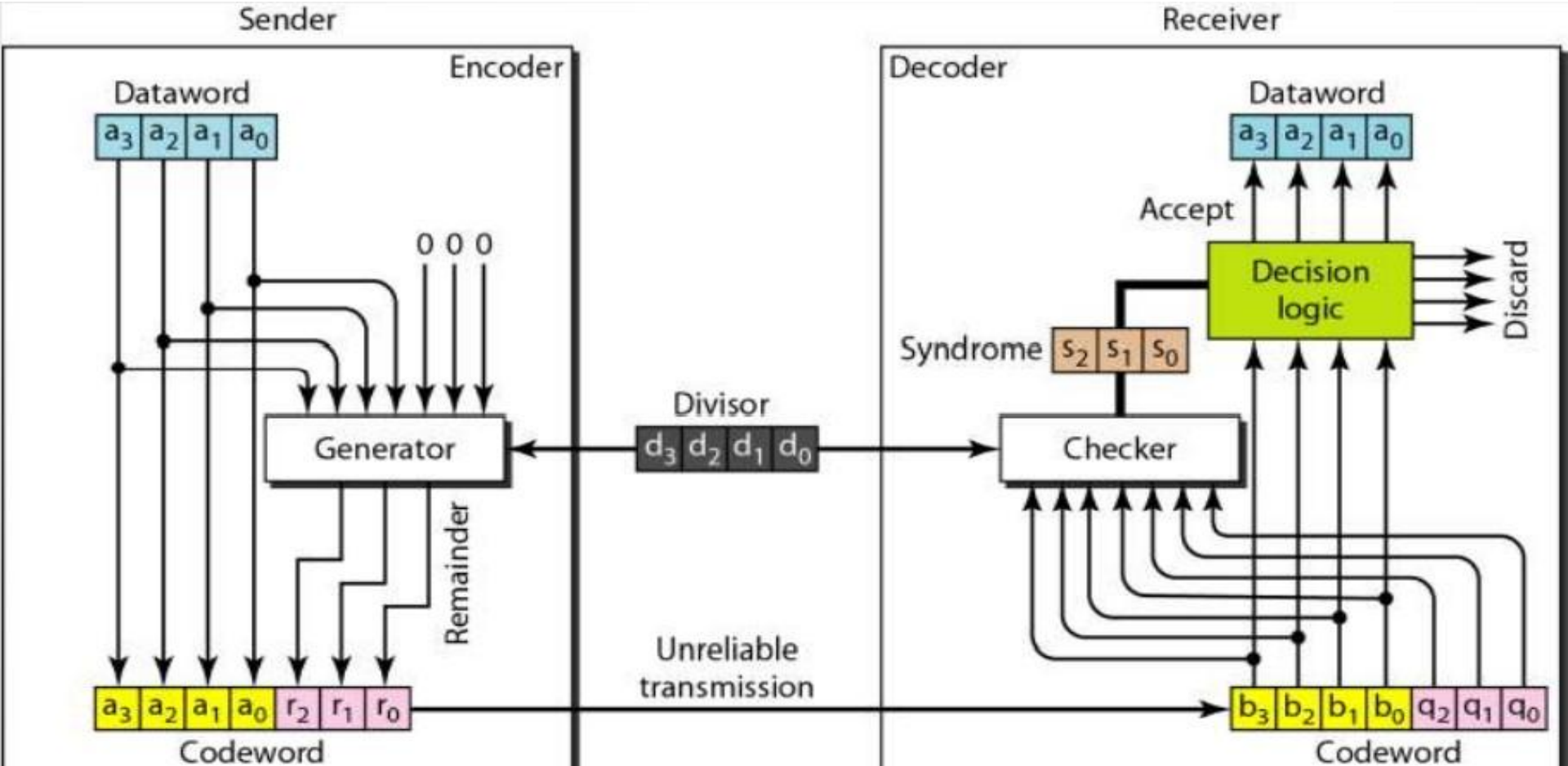
Division in CRC Encoder



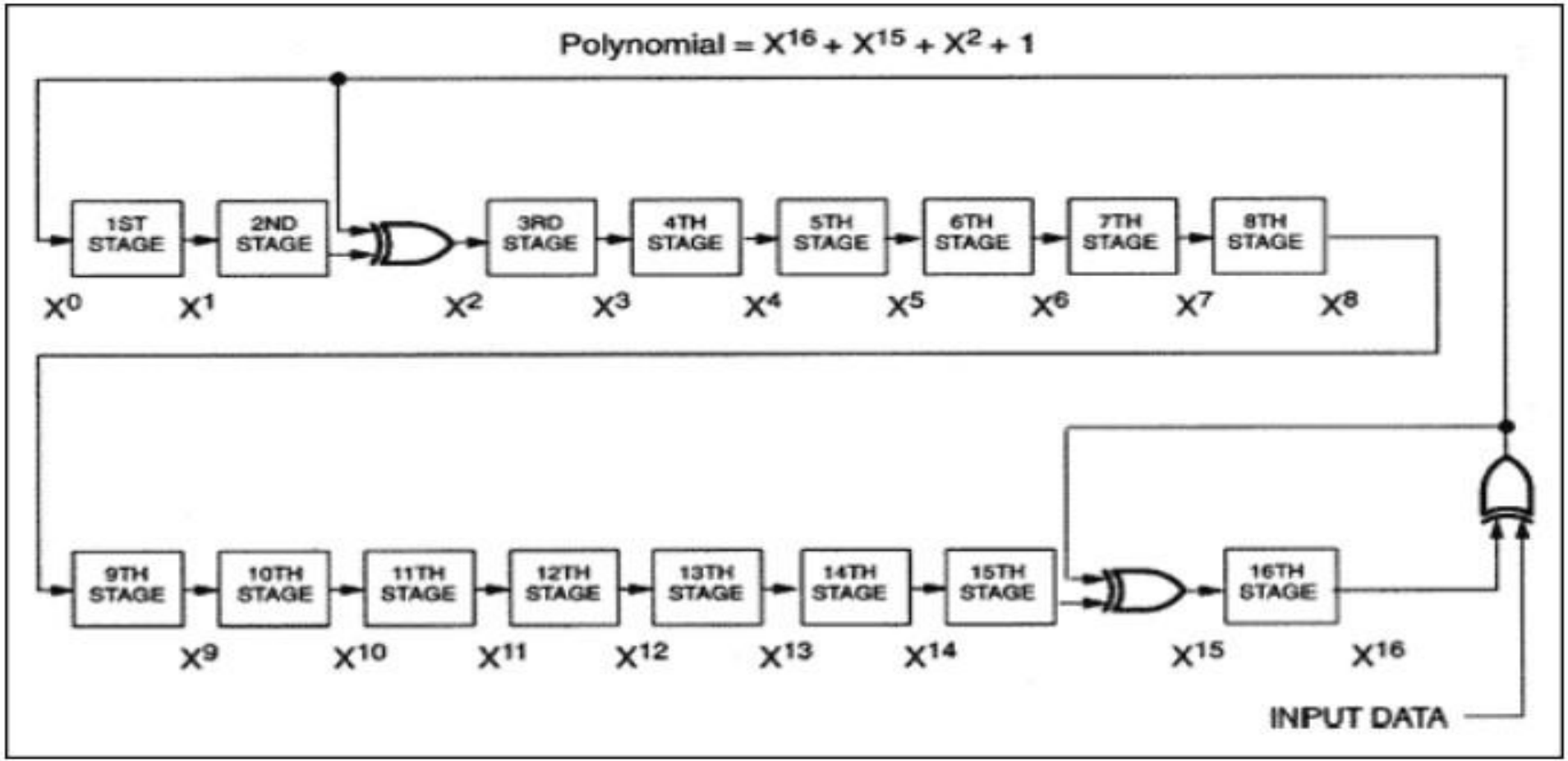
Division in Decoder



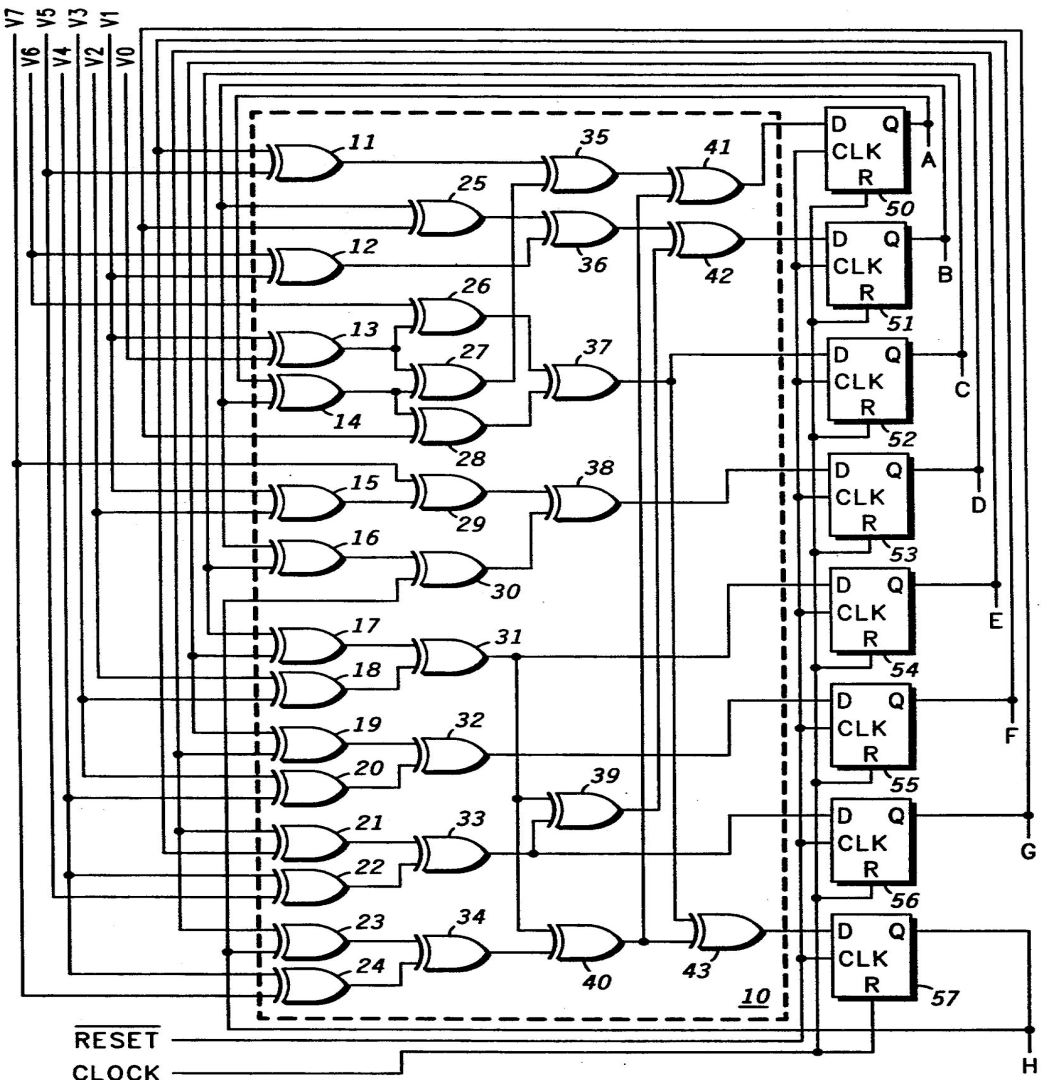
CRC Encoder and Decoder



CRC 16 bit serial example



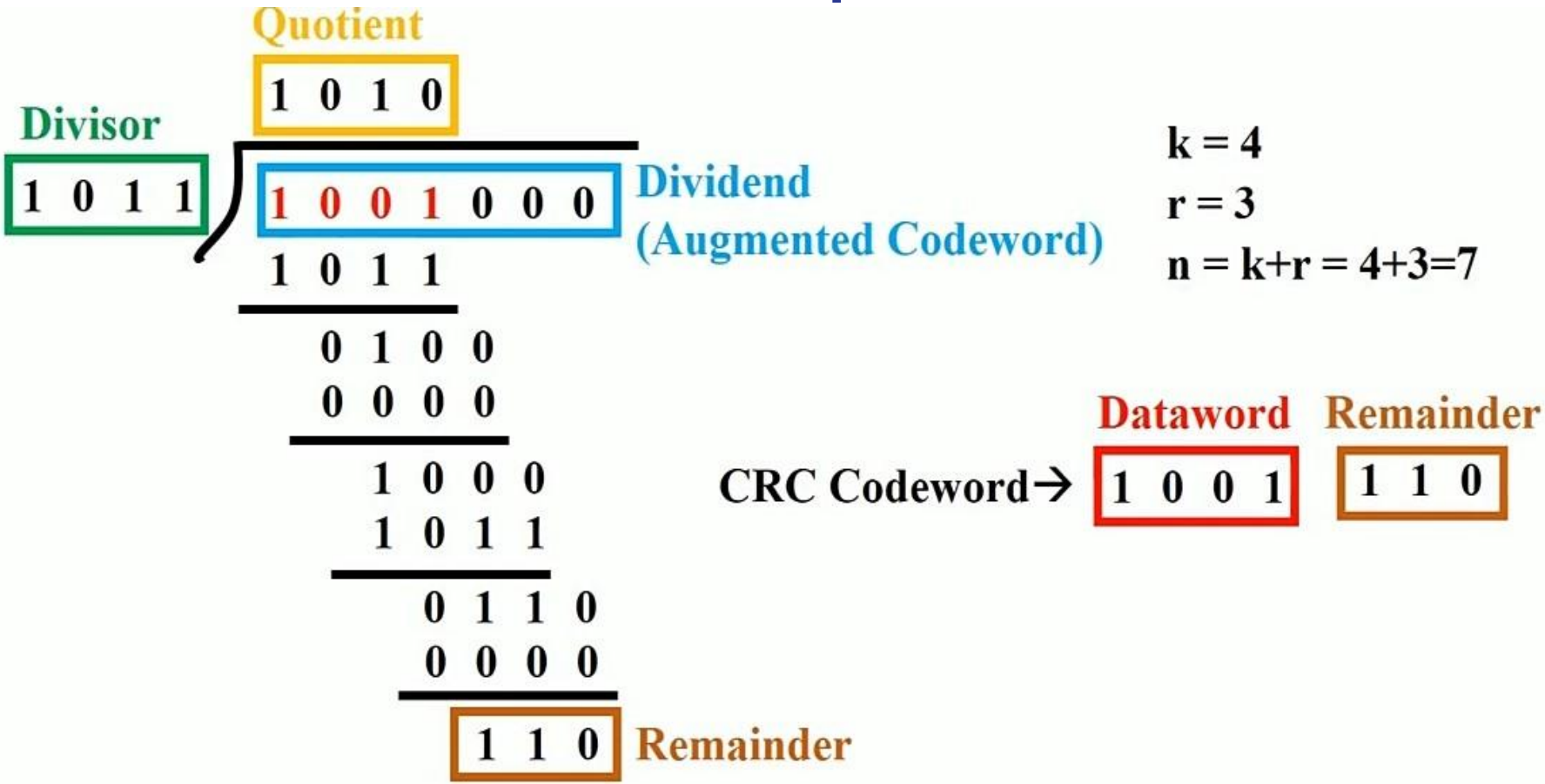
CRC 8 bit Parallel Example



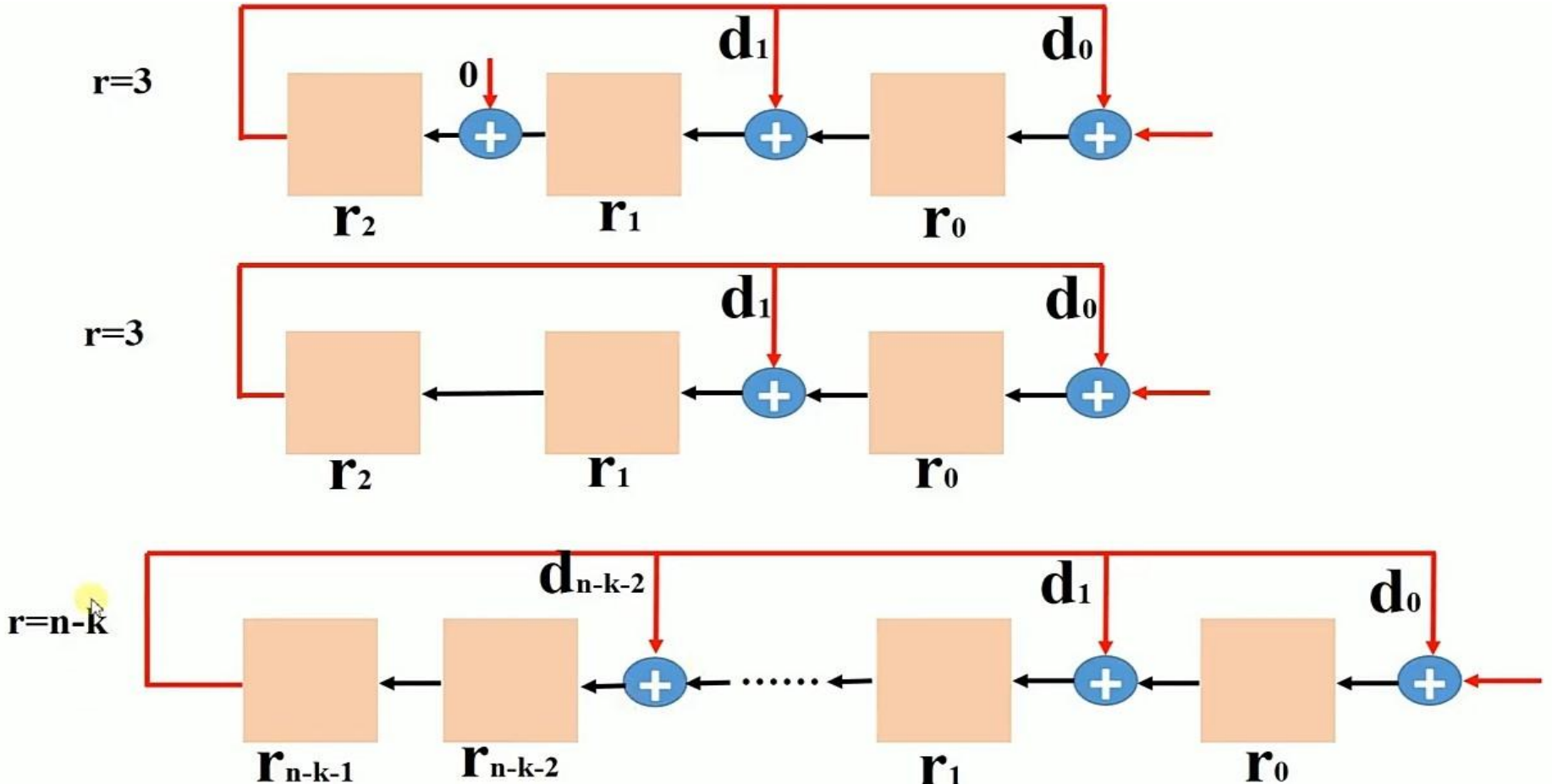
Hardware Implementation Series

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

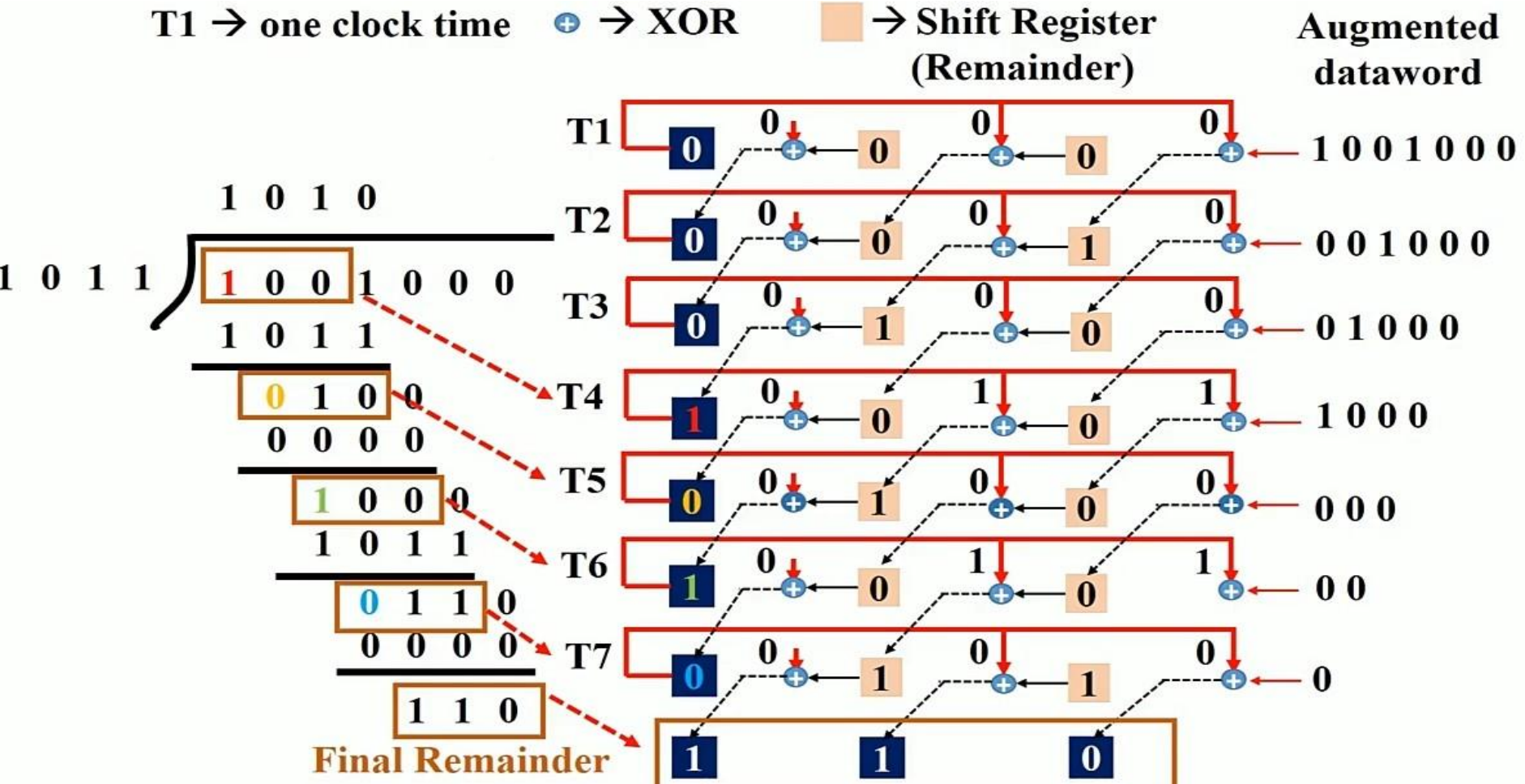
Concept

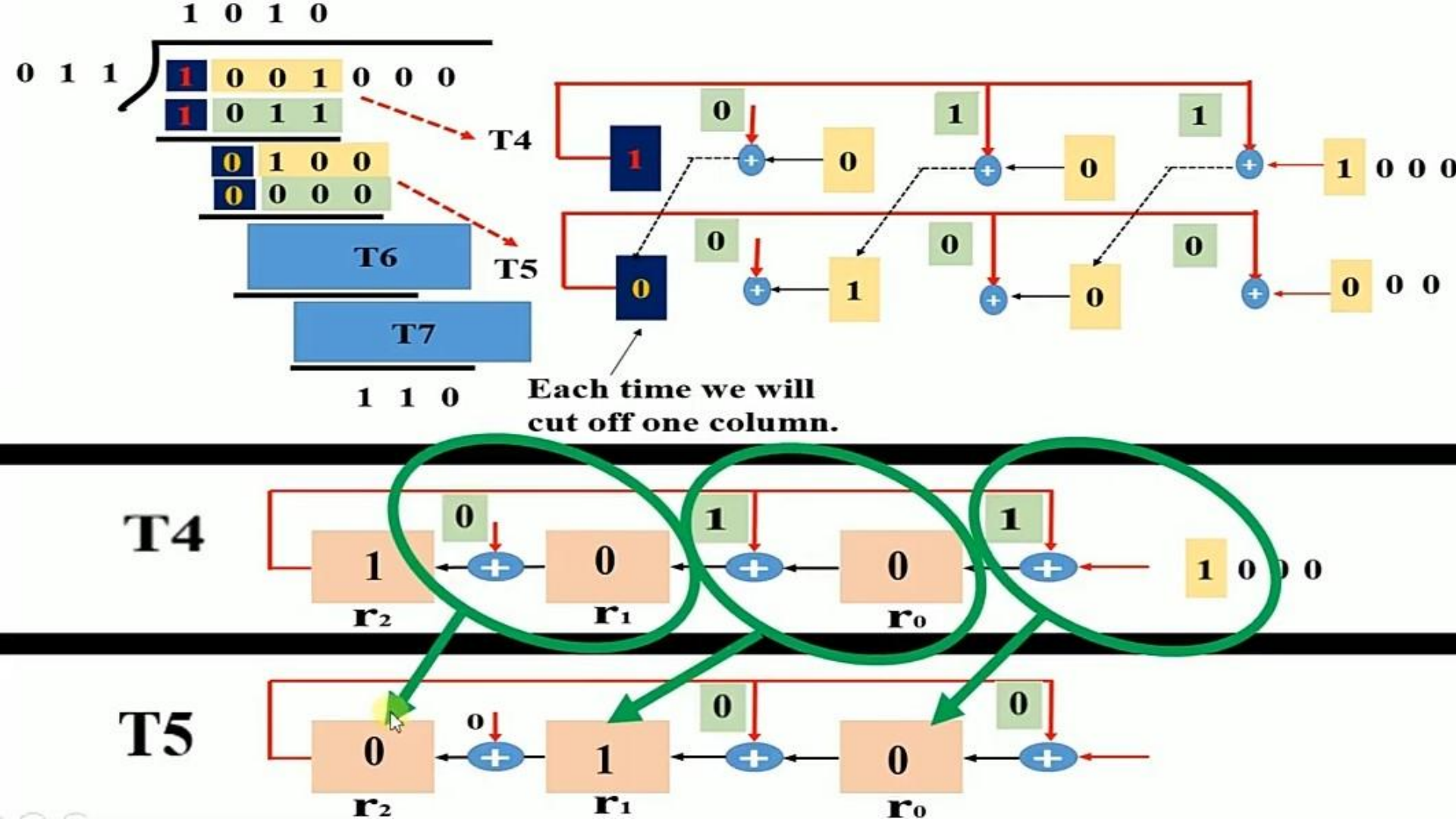

Quotient
1 0 1 0
Divisor
1 0 1 1
1 0 0 1 0 0 0
Dividend
(Augmented Codeword)
1 0 1 1
0 1 0 0
0 0 0 0
1 0 0 0
1 0 1 1
0 1 1 0
0 0 0 0
1 1 0
Remainder
k = 4
r = 3
n = k+r = 4+3=7
Dataword
Remainder
CRC Codeword→
1 0 0 1
1 1 0

Principle And Implementation

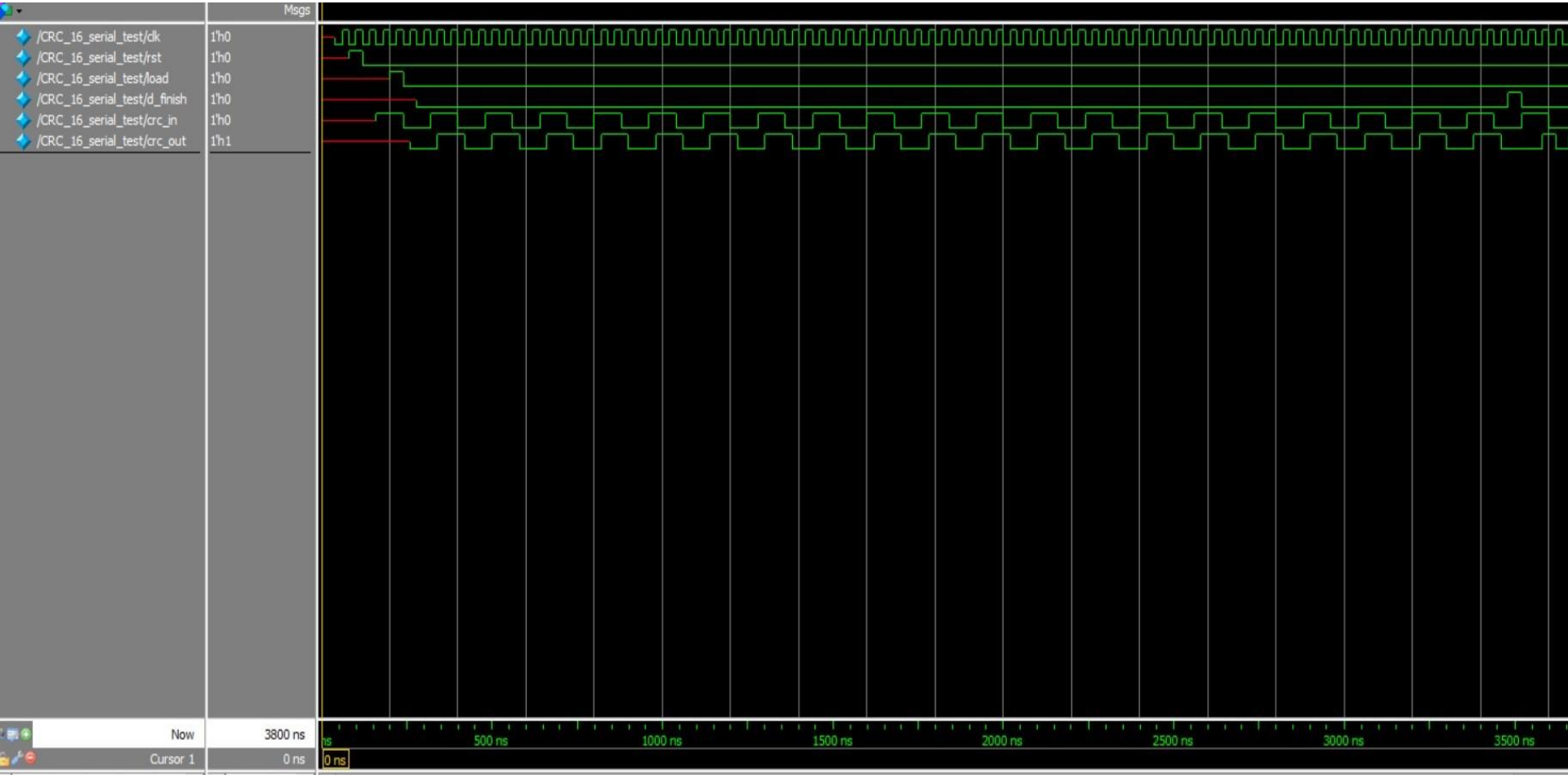


4 bit Series implementation

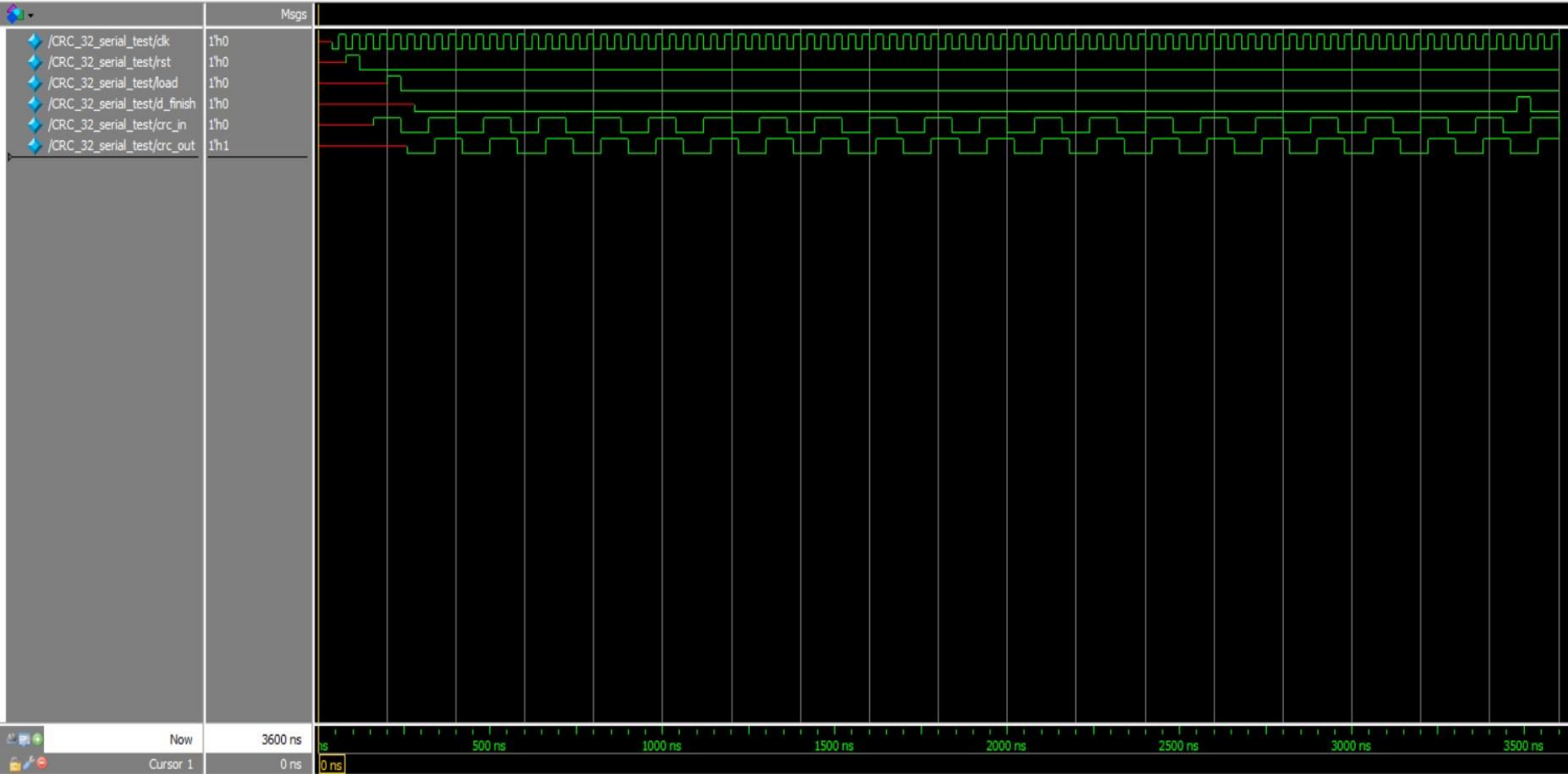



1 0 1 0
0 1 1
1 0 0 1 0 0 0
1 0 1 1
T4
0 1 0 0
0 0 0 0
T6
T5
T7
1 1 0
Each time we will cut off one column.
T4
T5
r2
r1
r0

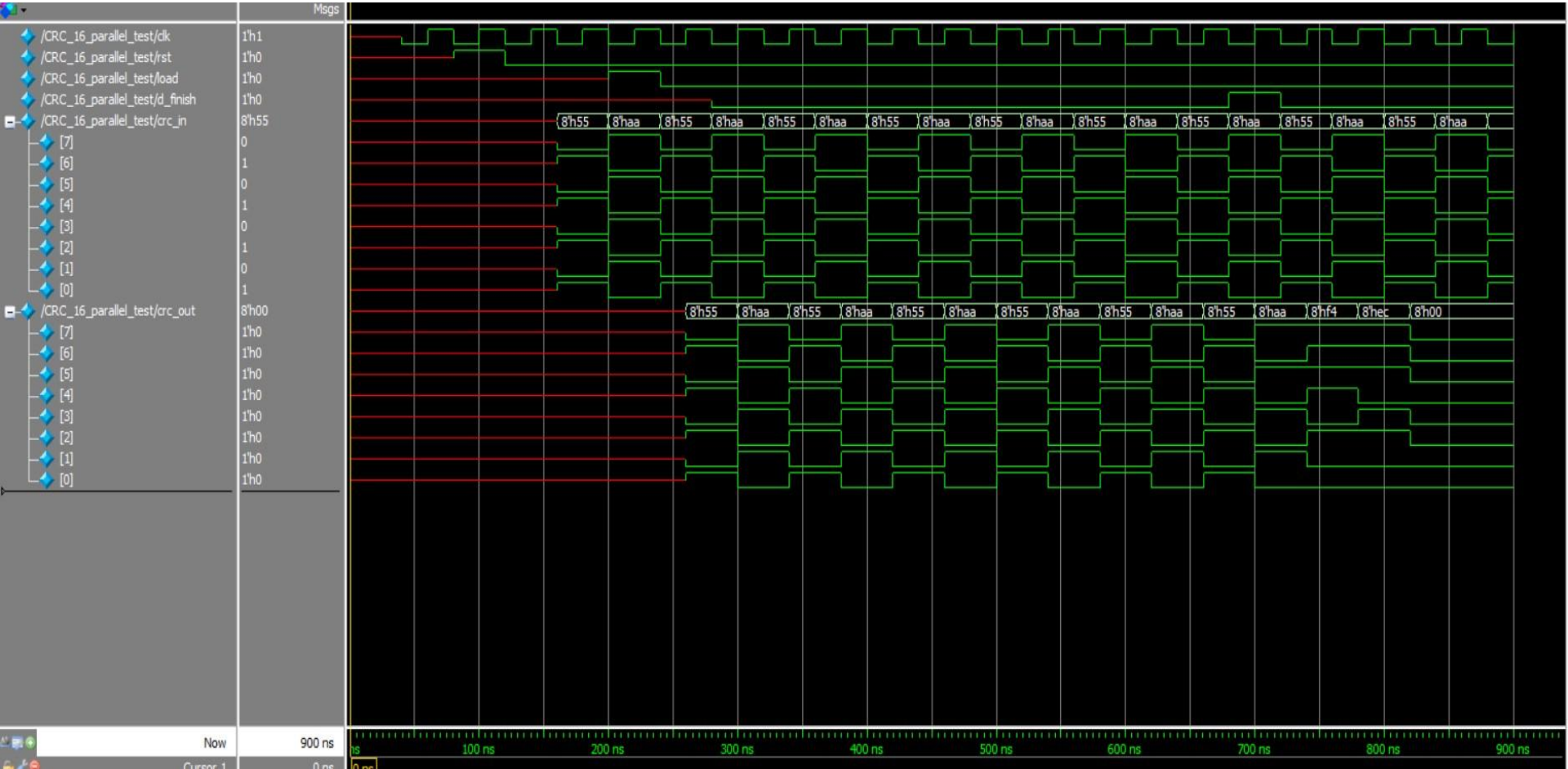
Results


Msgs
/CRC_16_serial_test/clk
/CRC_16_serial_test/rst
/CRC_16_serial_test/load
/CRC_16_serial_test/d_finish
/CRC_16_serial_test/crc_in
/CRC_16_serial_test/crc_out
1'h0
1'h0
1'h0
1'h0
1'h0
1'h1
Now
3800 ns
Cursor 1
0 ns
ns
500 ns
1000 ns
1500 ns
2000 ns
2500 ns
3000 ns
3500 ns
0 ns

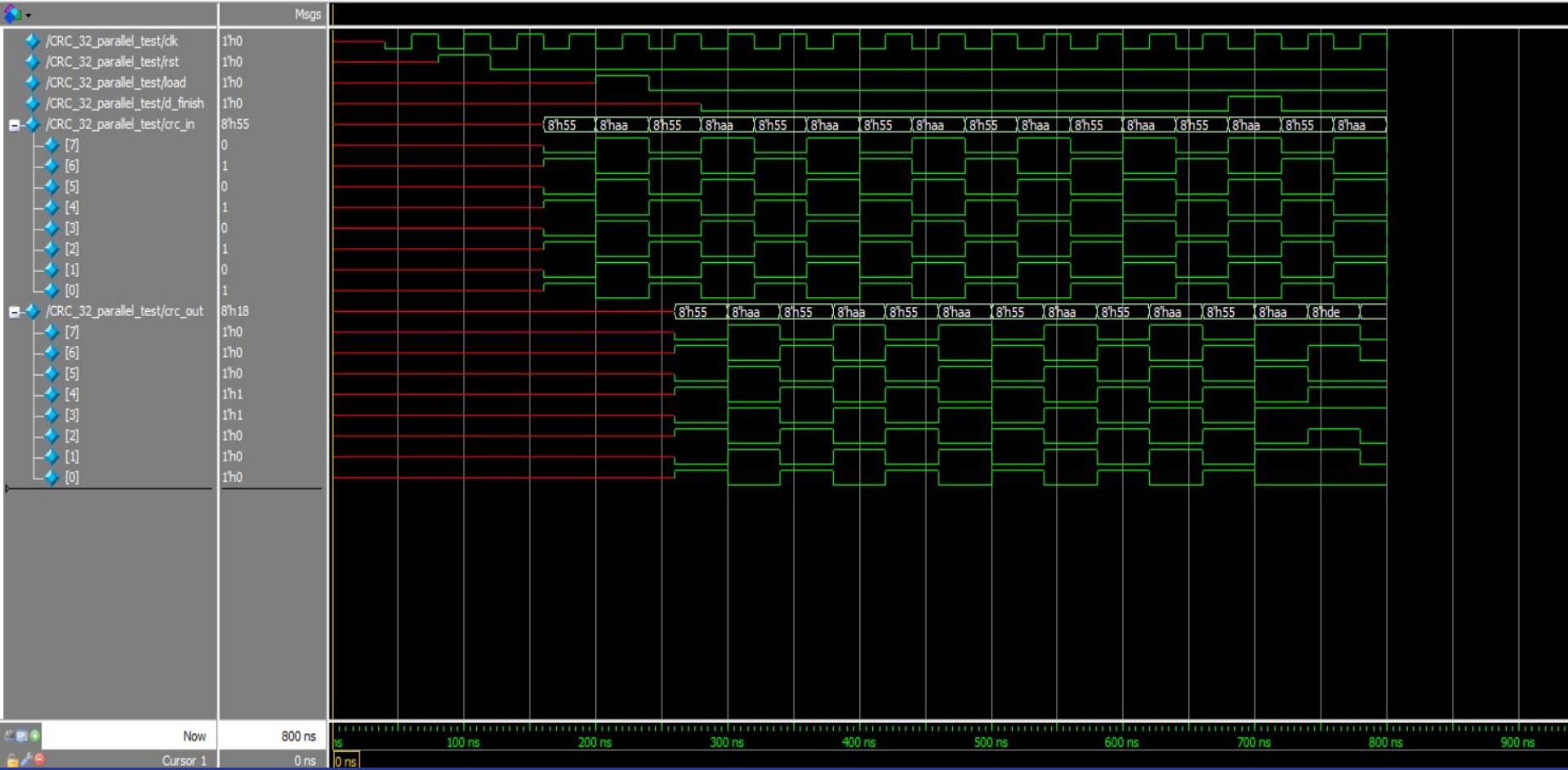
Results


Msgs
/CRC_32_serial_test/clk
/CRC_32_serial_test/rst
/CRC_32_serial_test/load
/CRC_32_serial_test/d_finish
/CRC_32_serial_test/crc_in
/CRC_32_serial_test/crc_out
1'h0
1'h0
1'h0
1'h0
1'h0
1'h1
Now
3600 ns
Cursor 1
0 ns
500 ns
1000 ns
1500 ns
2000 ns
2500 ns
3000 ns
3500 ns
0 ns

Results


Msgs
/CRC_16_parallel_test/clk
/CRC_16_parallel_test/rst
/CRC_16_parallel_test/load
/CRC_16_parallel_test/d_finish
/CRC_16_parallel_test/crc_in
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
/CRC_16_parallel_test/crc_out
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
1'h1
1'h0
1'h0
1'h0
8'h55
0
1
0
1
0
1
0
1
8'h00
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa
8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'hf4 8'hec 8'h00
Now
900 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns

Results


Msgs
/CRC_32_parallel_test/clk
/CRC_32_parallel_test/rst
/CRC_32_parallel_test/load
/CRC_32_parallel_test/d_finish
/CRC_32_parallel_test/crc_in
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
/CRC_32_parallel_test/crc_out
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
1'h0
1'h0
1'h0
1'h0
8'h55
0
1
0
1
0
1
0
1
8'h18
1'h0
1'h0
1'h0
1'h1
1'h1
1'h0
1'h0
1'h0
8'h55
8'haa
8'hde
Now
800 ns
Cursor 1
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns

Thank You