CSE 4224 : Digital Systems Design Laboratory

Title: **8-bit CRC Generator and Checker**
Submission Date : 25/01/2025

Submitted By,

Name : Sumiya Islam Barsha
Roll : 1907076

Name : Nushrat Tarmin Meem
Roll : 1907083

Submitted To,

Md. Sakhawat Hossain
Lecturer
Dept. of CSE , KUET

Md. Badiuzzama Shuvo
Lecturer
Dept. of CSE , KUET

8-bit CRC Generator and Checker

Objectives:

- To design and implement an 8-bit CRC (Cyclic Redundancy Check) generator and checker
- To verify the functionality of the CRC system using a state machine and logical operations
- To ensure error detection in digital data transmission through the CRC method
- To explore hardware implementation using sequence registers, decoders, and control logic

Introduction:

Cyclic Redundancy Check (CRC) is a widely used and highly effective error-detection mechanism in digital systems, designed to ensure data integrity during transmission. Errors often occur during data communication due to noise, interference, or transmission faults, and detecting these errors is critical for maintaining the reliability of digital systems. CRC uses polynomial division to generate a unique remainder, or "checksum," for a given block of data, which is then transmitted along with the data. At the receiving end, the same process is applied, and the results are compared to detect errors. This process is both efficient and computationally simple, making CRC an ideal choice for many real-time applications. This report focuses on the design and implementation of an 8-bit CRC generator and checker, providing an in-depth analysis of its operational workflow. It includes key elements such as the state diagram, which illustrates the sequence of operations, the flowchart for understanding the logical flow, and the hardware components essential for implementation. The system employs mathematical operations like concatenation, XOR division, and polynomial-based calculations to generate and verify the checksum, ensuring accurate and error-free data transmission. By utilizing a modular approach to the design, the report emphasizes the scalability and adaptability of the CRC method for different communication systems. The project also highlights the practical applications of CRC in modern digital systems. From network communication protocols to storage systems and embedded devices, CRC plays a pivotal role in safeguarding data integrity. Its simplicity in implementation and high effectiveness make it a cornerstone in error detection mechanisms. This report serves as a detailed guide, offering insights into the design principles and practical implications of CRC, reaffirming its importance in secure and reliable data transmission.

Process:

❑ Design of the CRC Generator:

- The CRC generator begins with the creation of message data (M) and a binary polynomial (X).
- At the sender side, the message is concatenated with (N-1) zeros, where N is the degree of the binary polynomial, creating a modified data stream.
- XOR division is performed between the modified data stream and the polynomial, producing a remainder (R), which is appended to the message before transmission.

❑ Error Detection at the Receiver:

- The receiver takes the transmitted data, including the appended remainder, and applies the same XOR division using the same polynomial (X).
- A new remainder (R′) is generated. If R′ equals zero, the data is error-free; otherwise, an error is detected.

❑ Sequence Register and Decoder Usage:

- A sequence register and decoder method was utilized to efficiently manage the XOR division and concatenation process at both sender and receiver ends.
- These components simplified the handling of data bits and ensured synchronization during the operations.

❑ Hardware Components:

- The hardware design included sequence registers for data storage, control logic for managing state transitions, and a 4x16 decoder to handle system operations.
- Boolean functions were derived for D-flipflops, enabling state transitions and generation of control signals like flags and resets.

❑ Block Diagram Integration:

- The system's block diagram incorporated essential modules such as input/output registers, an Arithmetic Logic Unit (ALU) for concatenation and XOR division, and a comparator for detecting errors.
- These modules worked cohesively, ensuring a seamless flow of operations and accurate error detection in the CRC system.

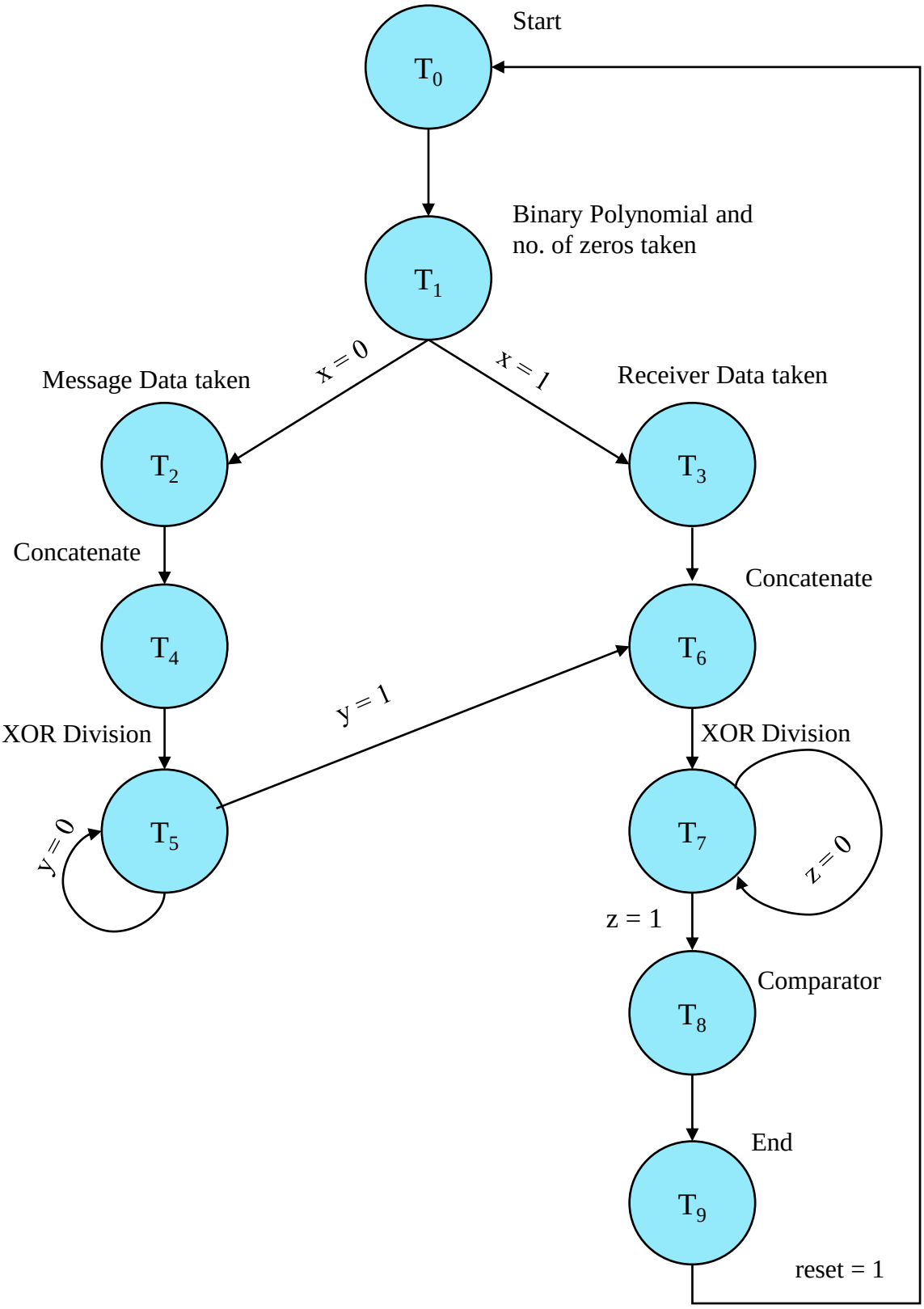


Figure 1: State Diagram of CRC Generator and Checker

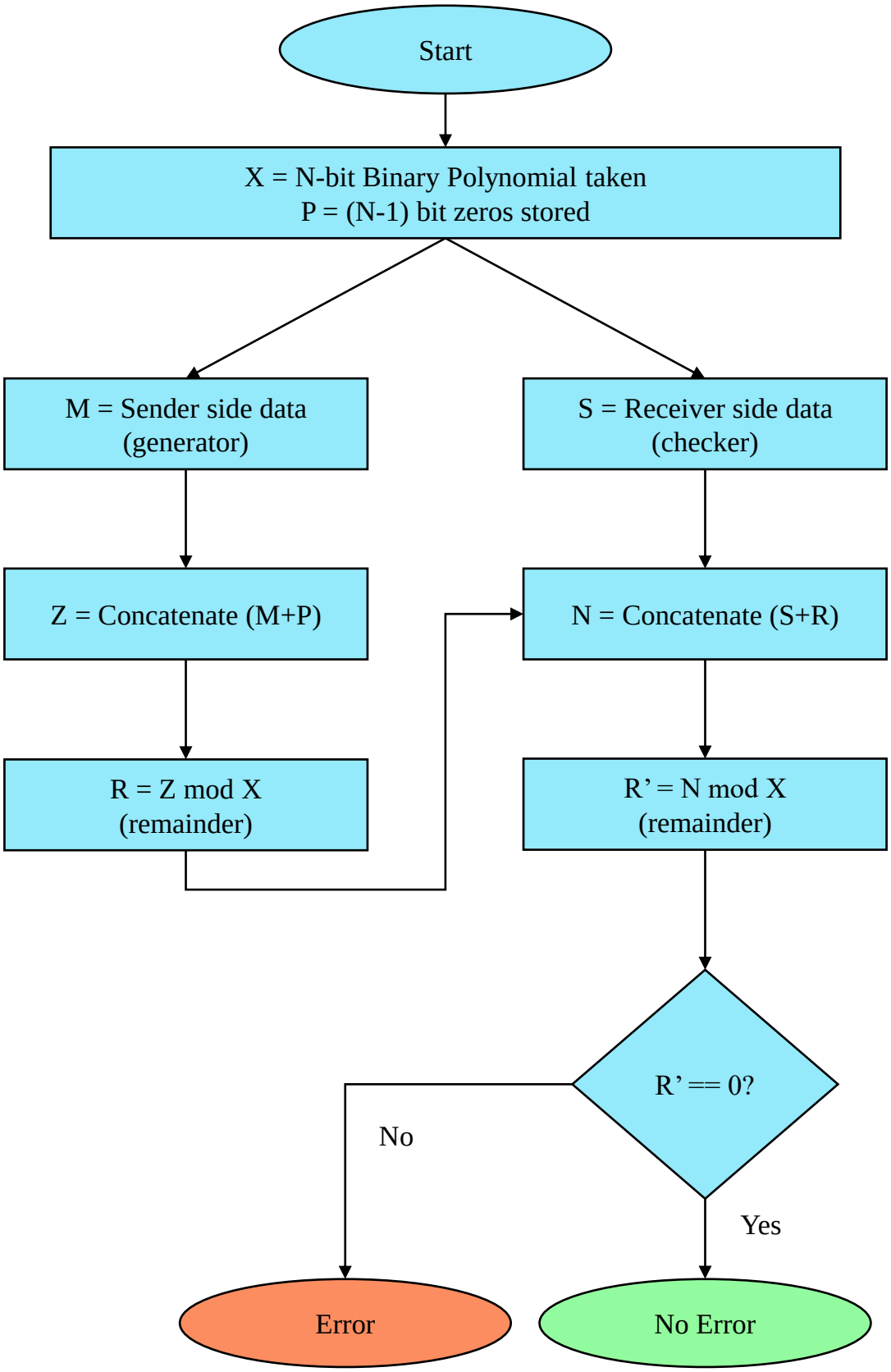


Figure 2: Flowchart for CRC

Sequence Register and Decoder Method

	Q4	Q3	Q2	Q1	x	y	z	Q4’	Q3’	Q2’	Q1’	D4	D3	D2	D1
T0	0	0	0	0	X	X	X	0	0	0	1	0	0	0	1
T1	0	0	0	1	0	X	X	0	0	1	0	0	0	1	0
					1	X	X	0	0	1	1	0	0	1	1
T2	0	0	1	0	X	X	X	0	1	0	0	0	1	0	0
T3	0	0	1	1	X	X	X	0	1	1	0	0	1	1	0
T4	0	1	0	0	X	X	X	0	1	0	1	0	1	0	1
T5	0	1	0	1	X	0	X	0	1	0	1	0	1	0	1
					X	1	X	0	1	1	0	0	1	1	0
T6	0	1	1	0	X	X	X	0	1	1	1	0	1	1	1
T7	0	1	1	1	X	X	0	0	1	1	1	0	1	1	1
					X	X	1	1	0	0	0	1	0	0	0
T8	1	0	0	0	X	X	X	1	0	0	1	1	0	0	1
T9	1	0	0	1	X	X	X	0	0	0	0	0	0	0	0

Equations for D-flipflops by using these we’ll get q4, q3, q2 and q1:

D4 = (T7 & z) | T8

D3 = T2 | T3 | T4 | T5 | T6 | (T7 & ~z)

D2 = T1 | T3 | (T5 & y) | T6 | (T7 & ~z)

D1 = T0 | (T1 & x) | T4 | (T5 & ~y) | T6 | (T7 & ~z) | T8

Control Outputs Table

	S2	S1	S0	load	reset	flag
T0	0	0	0	0	0	0
T1	0	0	1	0	0	0
T2	0	0	1	0	0	0
T3	0	0	1	0	0	0
T4	0	1	0	1	0	0
T5	0	1	0	1	0	0
T6	0	1	0	1	0	0
T7	0	1	0	1	0	0
T8	1	0	0	0	0	1
T9	0	0	0	0	1	0

Boolean functions for output control:

S2 = T8

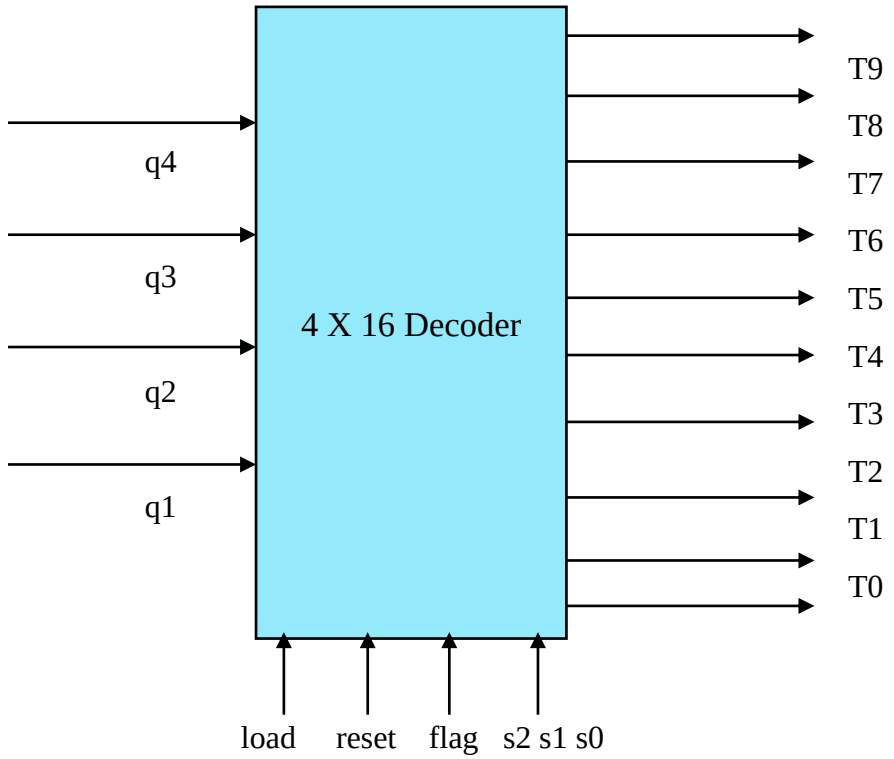
S1 = T4 | T5 | T6 | T7

S0 = T1 | T2 | T3 | T5 | T7

flag = T8

load = T4 | T5 | T6 | T7

reset = T9



Input				Output															
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Block Diagram

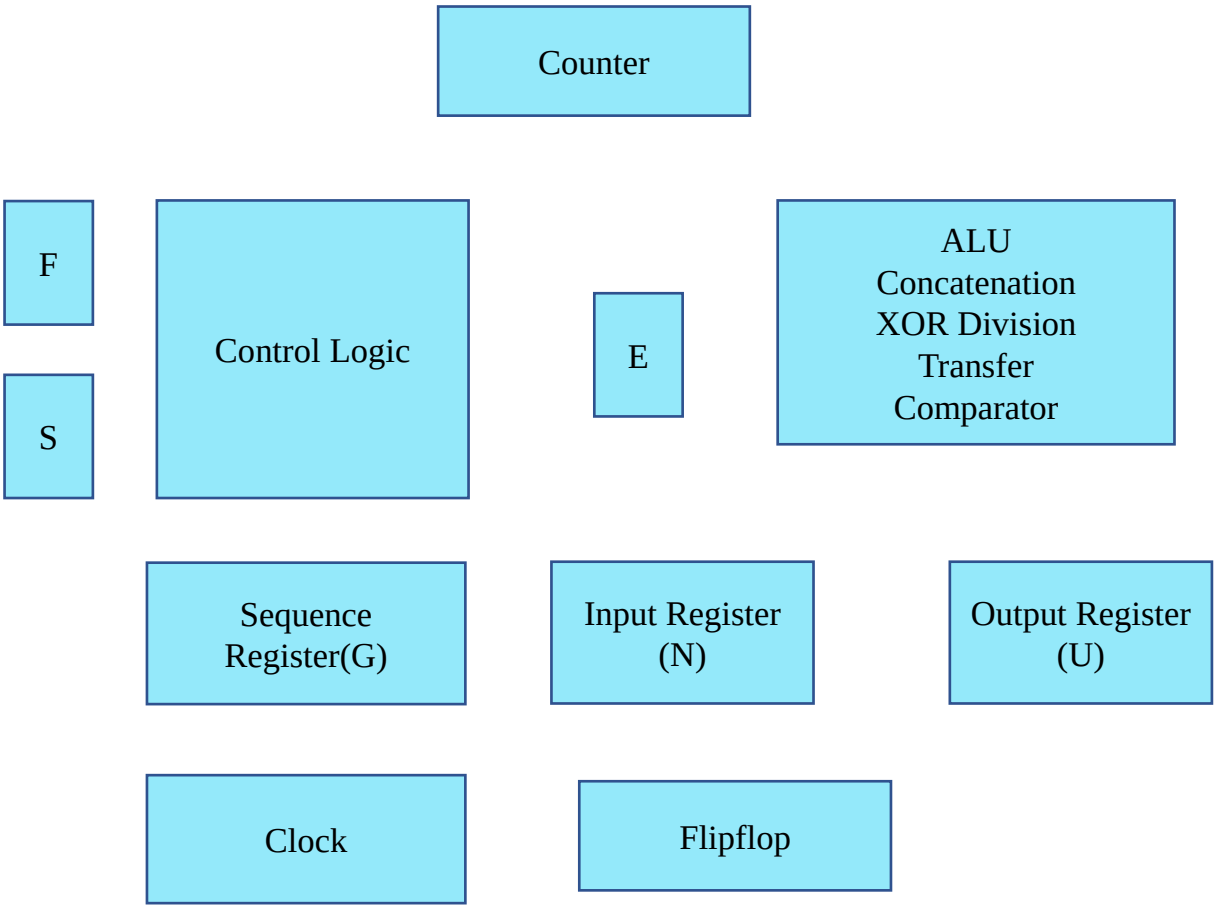


Figure 3: Block Diagram for CRC

Result Analysis:

```
Console
# KERNEL: CRC Computation Testbench
# KERNEL: Polynomial: x^5 + x^3 + x^2 + 1
# KERNEL: Sender: S_Data = 11011101, S_CRC = 01111
# KERNEL: Receiver: R_Data = 11010101, R_CRC = 11111
# KERNEL: Test 1 Failed: Error detected
```

```
Console
# KERNEL: CRC Computation Testbench
# KERNEL: Polynomial: x^5 + x^2 + 1
# KERNEL: Sender: S_Data = 10011101, S_CRC = 01000
# KERNEL: Receiver: R_Data = 10011101, R_CRC = 00000
# KERNEL: Test 2 Passed: No error detected
```

Signal name	Value
data_in	DD
crc_en	1
rst	0
clk	0
crc_out	00
receiver_data_in	D5
remainder_last	1F
appending_bits	0F
polynomial	2D
flag_bit	x
enable	1
load	1
x	0
y	x
z	x
state_outputs	0010
counter_output	0D

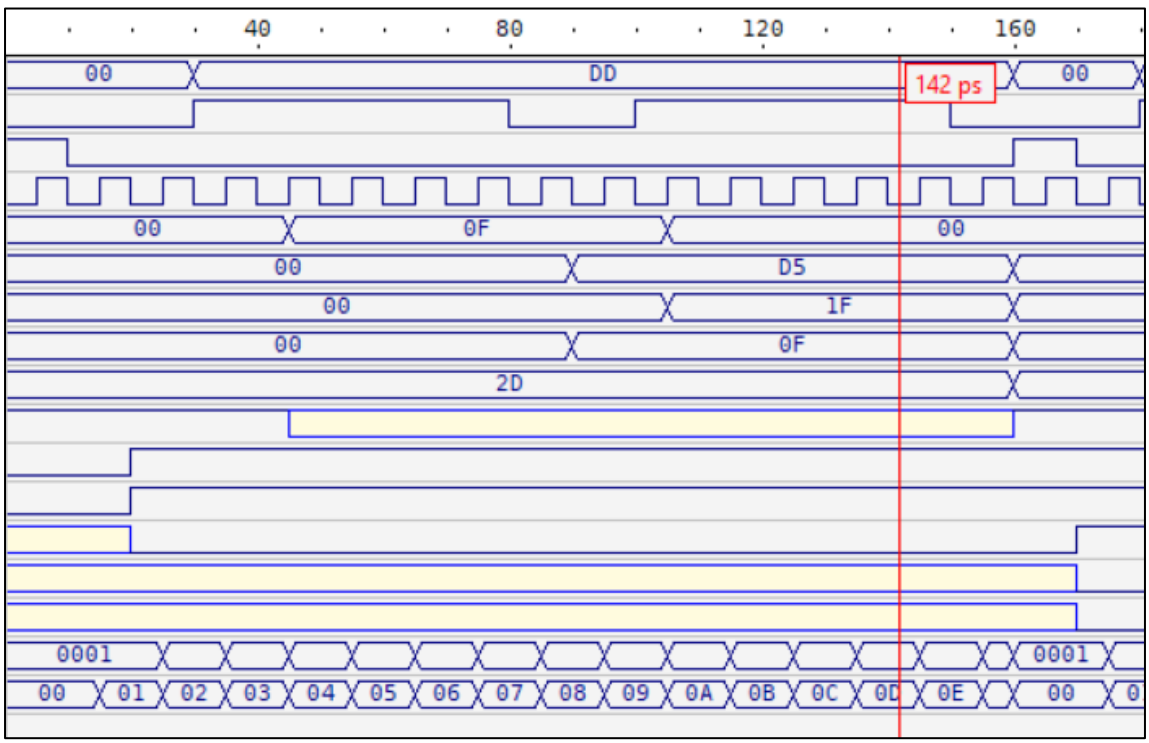


Figure 4: Waveform Outputs

The testbench results validate the accuracy of the CRC computation, demonstrating its effectiveness in error detection. In Test 1, the received data contained errors, resulting in a nonzero remainder, leading to a failed test. Conversely, Test 2 successfully passed as the received remainder was zero, confirming error-free transmission. The polynomial (x^5 + x^3 + x^2 + 1) was correctly applied, generating expected CRC values at both sender and receiver ends. The simulation completed execution verifying the correctness of the implemented CRC design.
On the other hand, the waveform (figure 4) simulation output of the 8-bit CRC generator and checker showcases the sequential operation of the system, validating its correctness. The data_in signal, holding a hexadecimal value of DD, is fed into the system while crc_en is enabled (1), allowing the CRC computation process to proceed. The clk signal exhibits a regular clock pulse, driving the sequential logic components. As the data propagates through the system, the crc_out signal, initially zero, undergoes changes as XOR division is performed.The receiver_data_in signal (D5) indicates the received data, which, when processed with the appended bits (0F), generates a remainder (1F). This aligns with the expected CRC process where the remainder is used for error detection. The polynomial value (2D) confirms that the predefined polynomial is correctly applied in the division process. Additionally, control signals such as enable (1) and load (1) confirm the system's active state. The state_outputs (0010) and counter_output (0D) reflect the system's internal state transition and iterative process in computing the CRC. The transitions of variables x, y, and z contribute to the stepwise execution of the CRC algorithm. The red cursor at 142 ps marks a critical event in the simulation, showing a transition point where computation completes or a key state is reached. This waveform analysis validates the successful operation of the CRC generator and checker, ensuring reliable error detection in transmitted data.

Conclusion:

The successful implementation of the 8-bit CRC generator and checker validates CRC as a reliable and efficient error-detection technique in digital communication. The waveform simulation confirms accurate polynomial division, remainder calculation, and state transitions, ensuring proper data integrity verification. The system processes input data through XOR-based division, appends the computed remainder, and checks for errors at the receiver end by performing the same division. Essential hardware components like sequence registers, control logic, and decoders ensure efficient real-time error detection and smooth system operation. The modular architecture enhances adaptability, allowing scalability for higher-bit CRC variations such as CRC-16 and CRC-32, which are widely used in networking, embedded systems, and data storage. This project highlights the critical role of CRC in maintaining data reliability and security, minimizing transmission errors, and improving overall communication efficiency, providing a strong foundation for future advancements in error detection techniques.