Final System

Submitted by: Sandy Samir

Contents

Introduction 2
Testbench Overview 2
Test Cases 2
1. Test Case 1: Write to Register File 2
2. Test Case 2: Write to Register File with Parity Error 3
3. Test Case 3: Write to Register File 4
4. Test Case 4: Read from Register File 4
5. Test Case 5: Read from Register File 5
6. Test Case 6: ALU Addition Operation with Operands 6
7. Test Case 7: ALU Subtraction Operation without Operands 6
8. Test Case 8: ALU Multiplication Operation with Operands 7
9. Test Case 9: ALU Division Operation without Operands 8
10. Test Case 10: ALU XNOR Operation with Operands 8
11. Test Case 11: ALU Compare Operation without Operands 9
12. Test Case 12: ALU Shift Right Operation 10
Conclusion 10

Introduction

This project involves the verification of the 'System_TOP' module using a comprehensive testbench written in Verilog. The testbench aims to validate the functionality of the 'System_TOP' module, which includes various UART commands, register file operations, and ALU operations. This report will discuss the setup of the testbench, the test cases executed, and the results obtained from the waveform analysis.

Testbench Overview

The testbench is designed to simulate the 'System_TOP' module by generating appropriate clock signals, applying reset conditions, and sending UART frames with different commands and data. The testbench includes tasks and functions to streamline the process of sending UART frames and calculating parity bits. The key components of the testbench are:

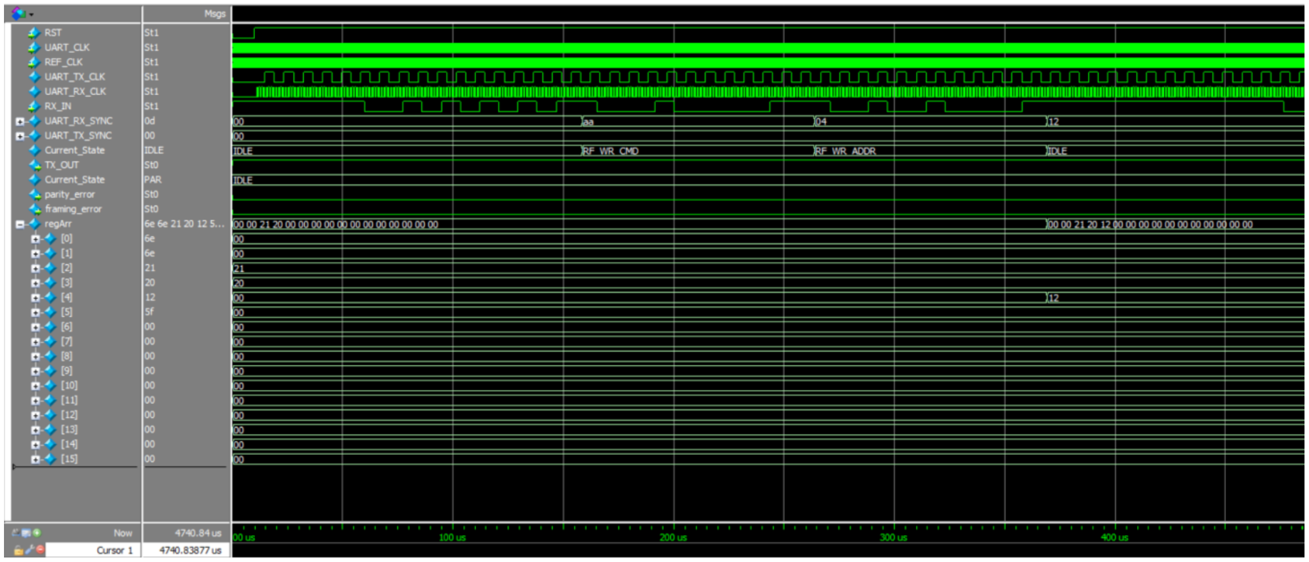
- **Clock Generation:** Two clocks are generated, 'UART_CLK' and 'REF_CLK', with frequencies of 3.6864 MHz and 50 MHz, respectively.
- **Initialization and Reset:** A task to initialize the signals and another task to apply the reset condition.
- **UART Frame Transmission:** A task to send UART frames, including start, data, parity, and stop bits.
- **Even Parity Calculation:** A function to calculate the even parity bit for a given data byte.

Test Cases

1. Test Case 1: Write to Register File

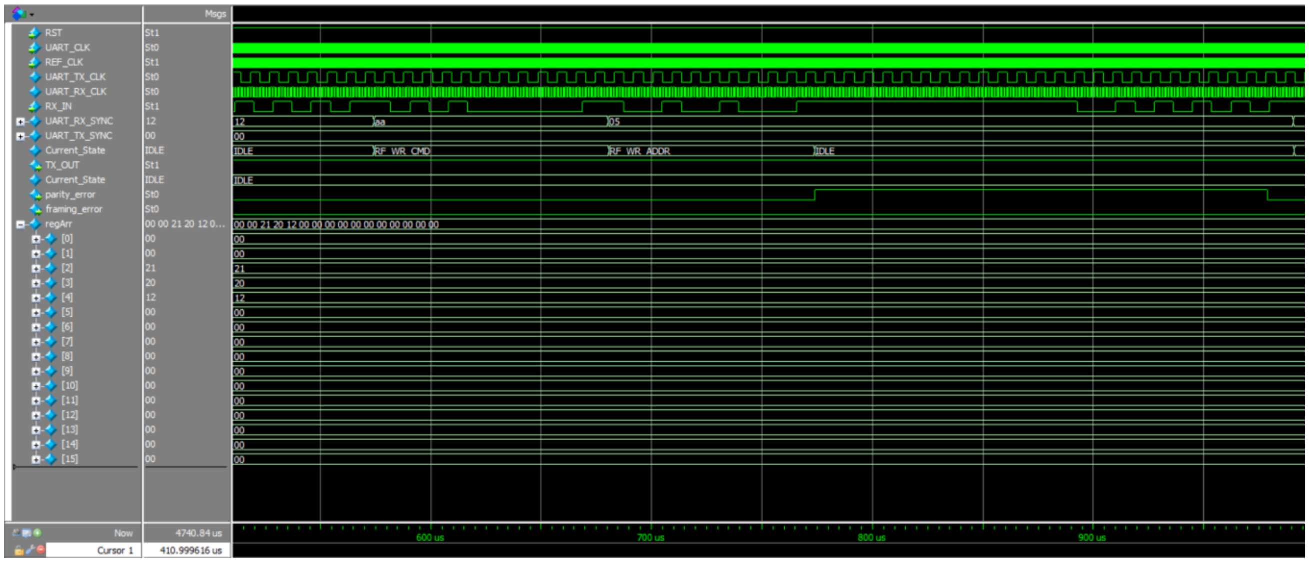
- **Description**: Write the value (0x12) to the register at address (0x04).
- **Steps**:
  1. Send RF_WR_CMD (0xAA).
  2. Send RF_WR_ADDR (0x04).
  3. Send RF_WR_DATA (0x12).

- **Waveform**:



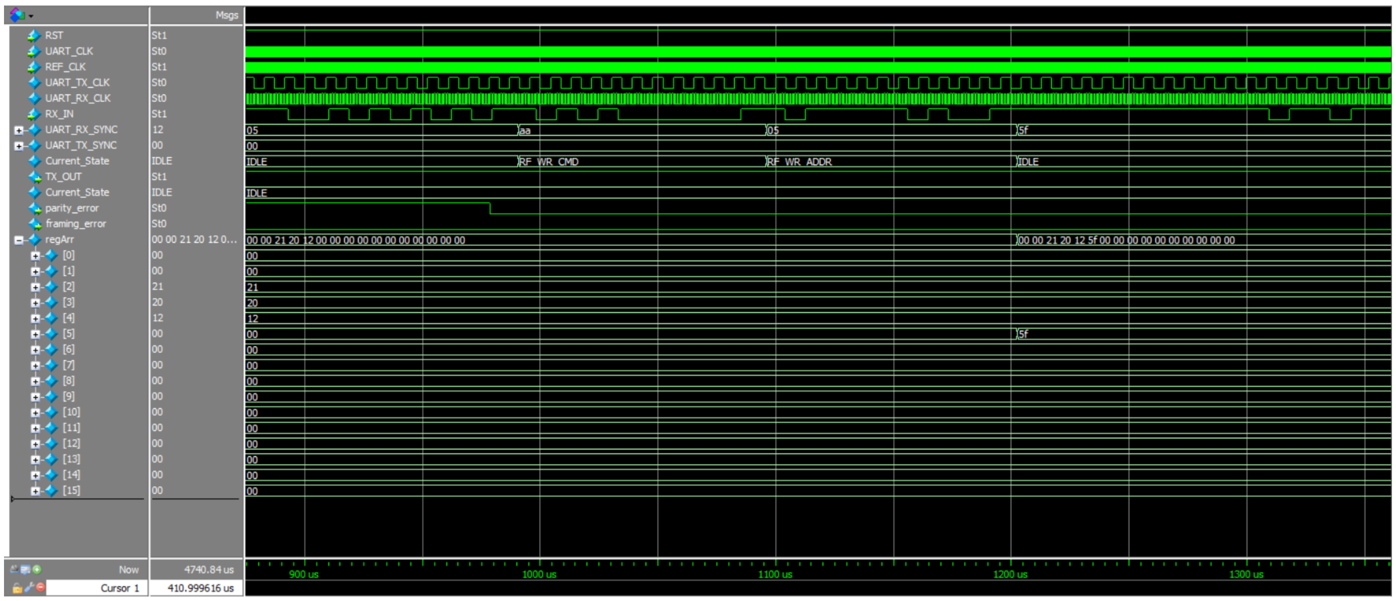
2. Test Case 2: Write to Register File with Parity Error

- **Description:** Attempt to write the value (0x12) to the register at address (0x05) with an incorrect parity bit.
- **Steps**:
  1. Send RF_WR_CMD (0xAA).
  2. Send RF_WR_ADDR (0x05).
  3. Send RF_WR_DATA (0x12) with incorrect parity.
- **Waveform:**



3. Test Case 3: Write to Register File

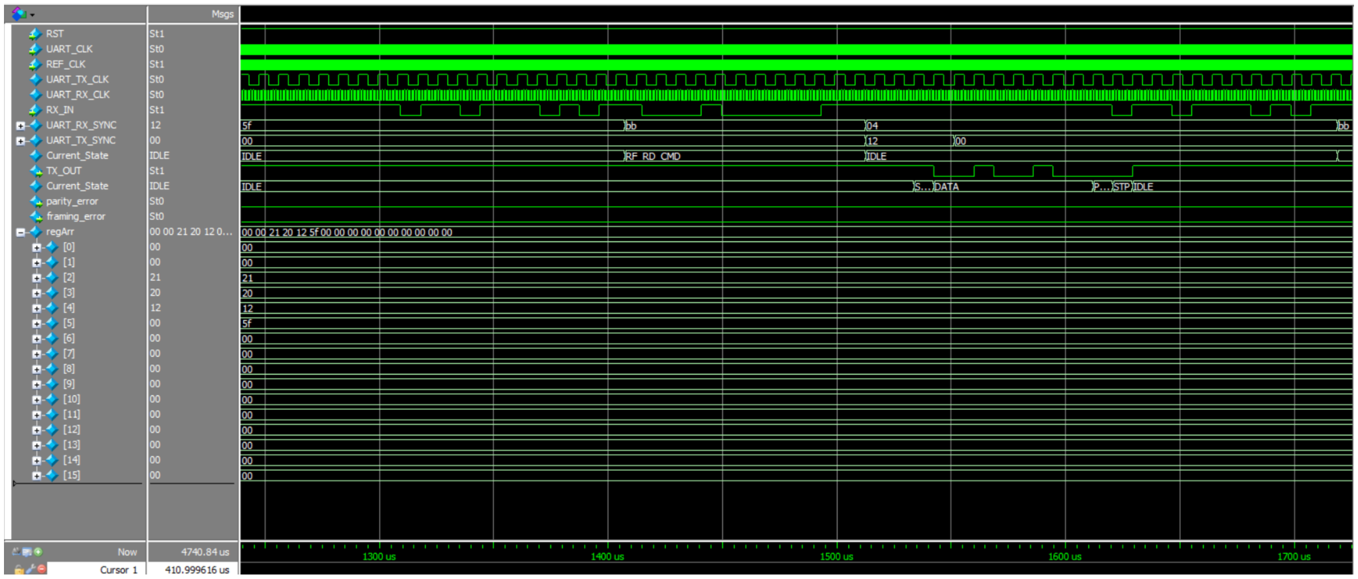
- **Description:** Write the value (0x5F) to the register at address (0x05).
- **Steps**:
  1. Send RF_WR_CMD (0xAA).
  2. Send RF_WR_ADDR (0x05).
  3. Send RF_WR_DATA (0x5F).
- **Waveform:**



4. Test Case 4: Read from Register File

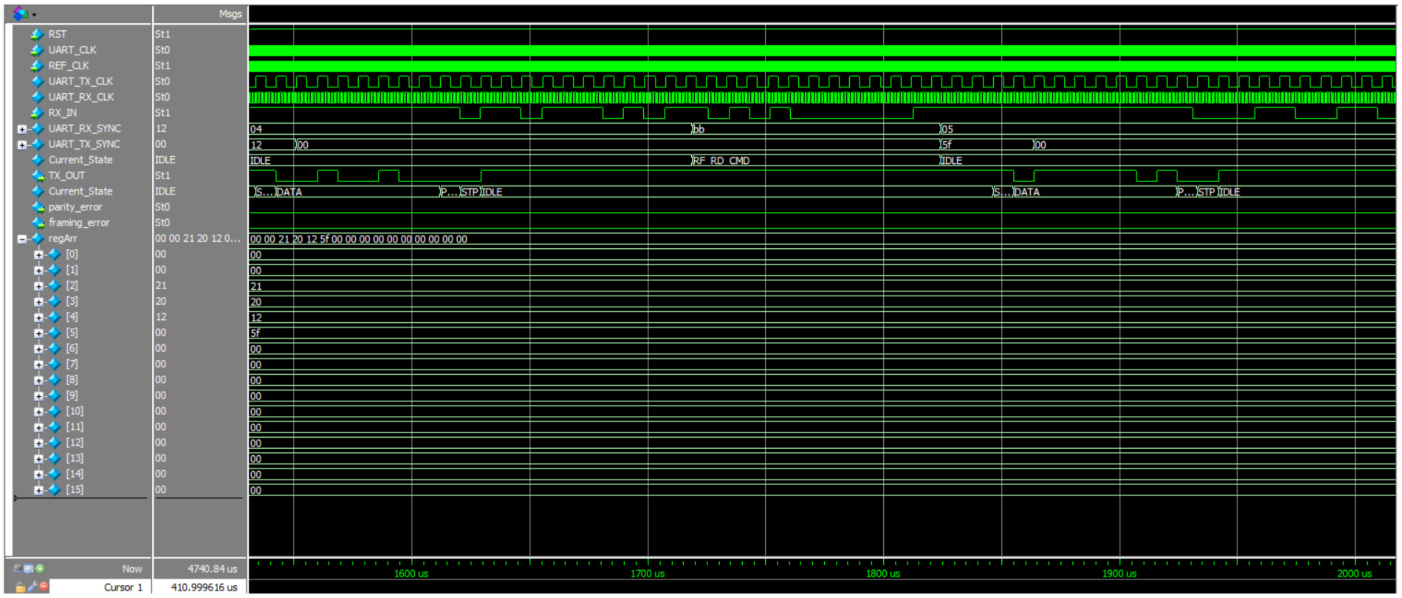
- **Description:** Read the value from the register at address (0x04).
- **Steps**:
  1. Send RF_RD_CMD (0xBB).
  2. Send RF_RD_ADDR (0x04).

- **Waveform:**



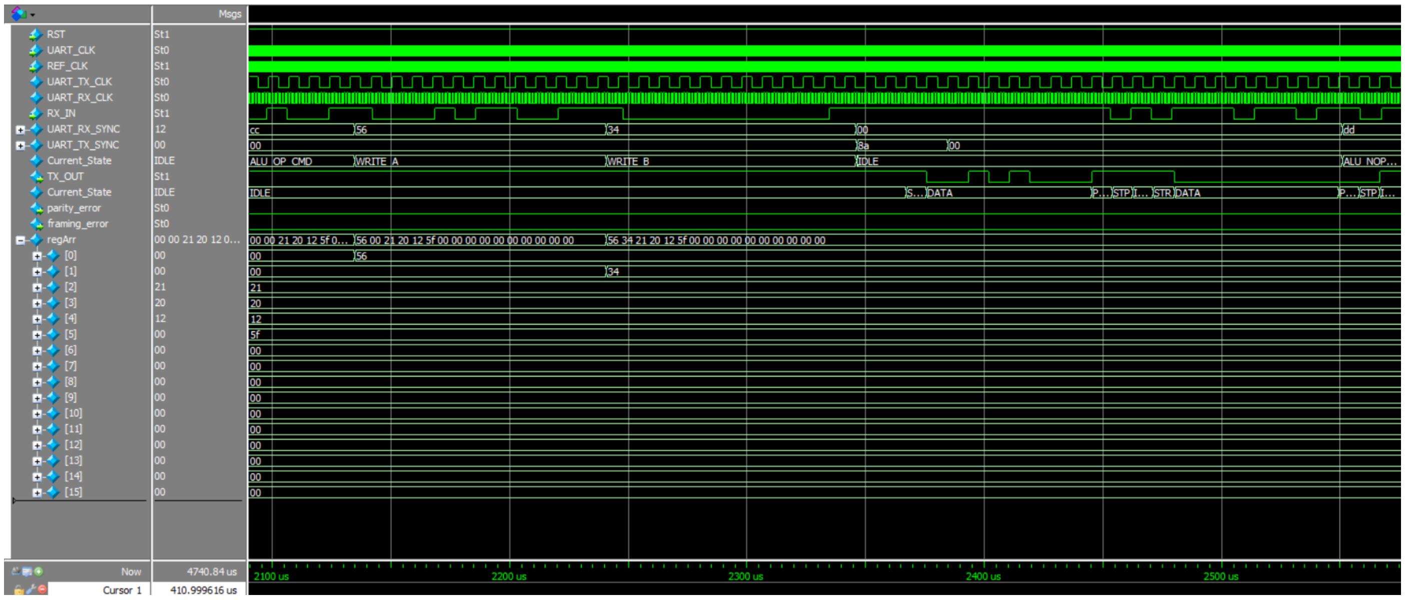
5. Test Case 5: Read from Register File

- **Description:** Read the value from the register at address (0x05).
- **Steps**:
  1. Send RF_RD_CMD (0xBB).
  2. Send RF_RD_ADDR (0x05).
- **Waveform:**



6. Test Case 6: ALU Addition Operation with Operands

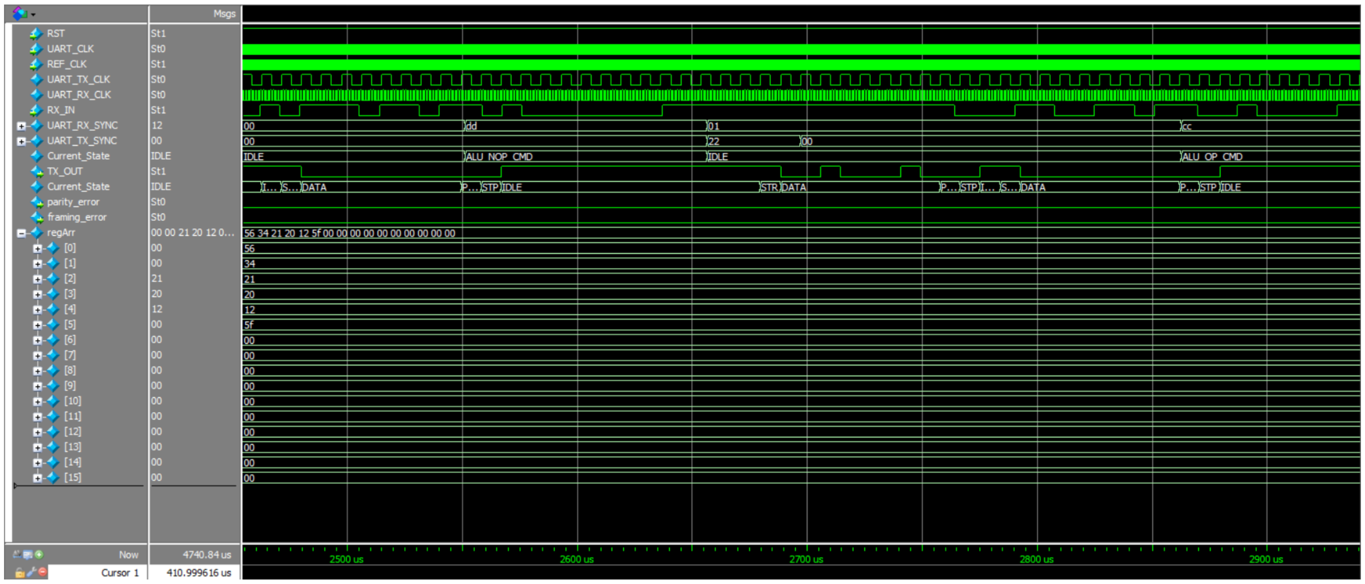
- **Description:** Perform an addition operation with operands (0x56) and (0x34).
- **Steps**:
  1. Send ALU_OP_CMD (0xCC).
  2. Send OPERAND_A (0x56).
  3. Send OPERAND_B (0x34).
  4. Send operation code for addition (0x00)
- **Waveform:**



7. Test Case 7: ALU Subtraction Operation without Operands

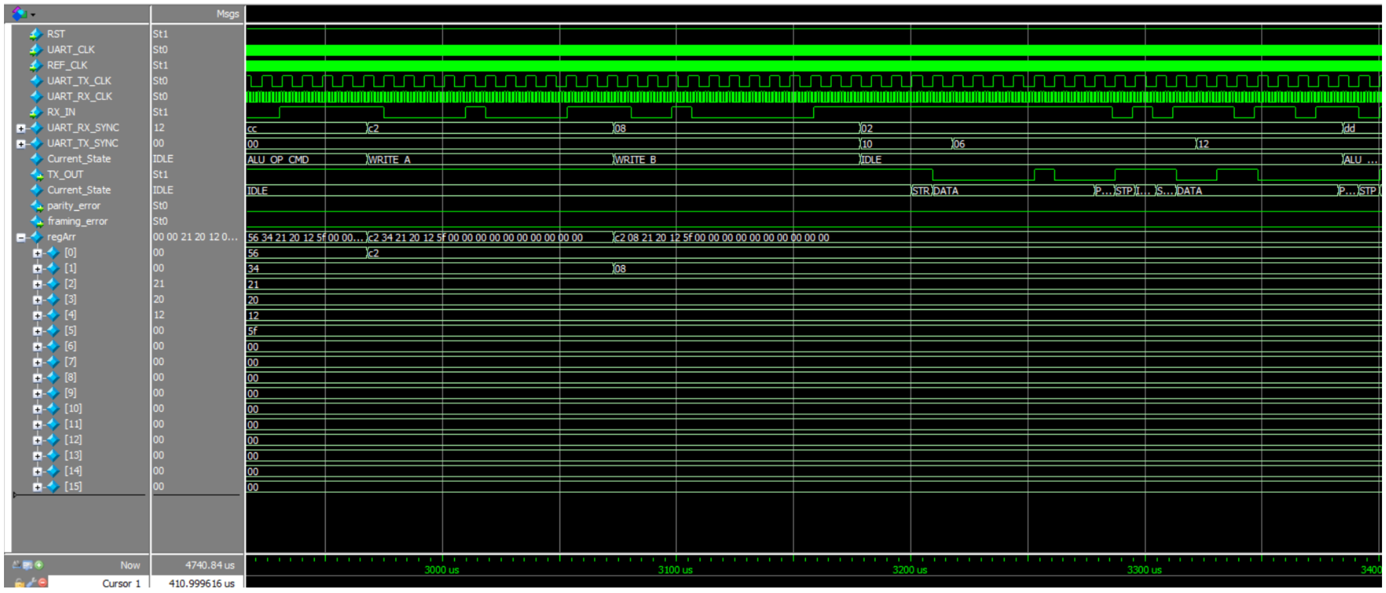
- **Description:** Perform a subtraction operation.
- **Steps**:
  1. Send ALU_OP_CMD (0xDD).
  2. Send operation code for subtraction (0x01).

- **Waveform:**



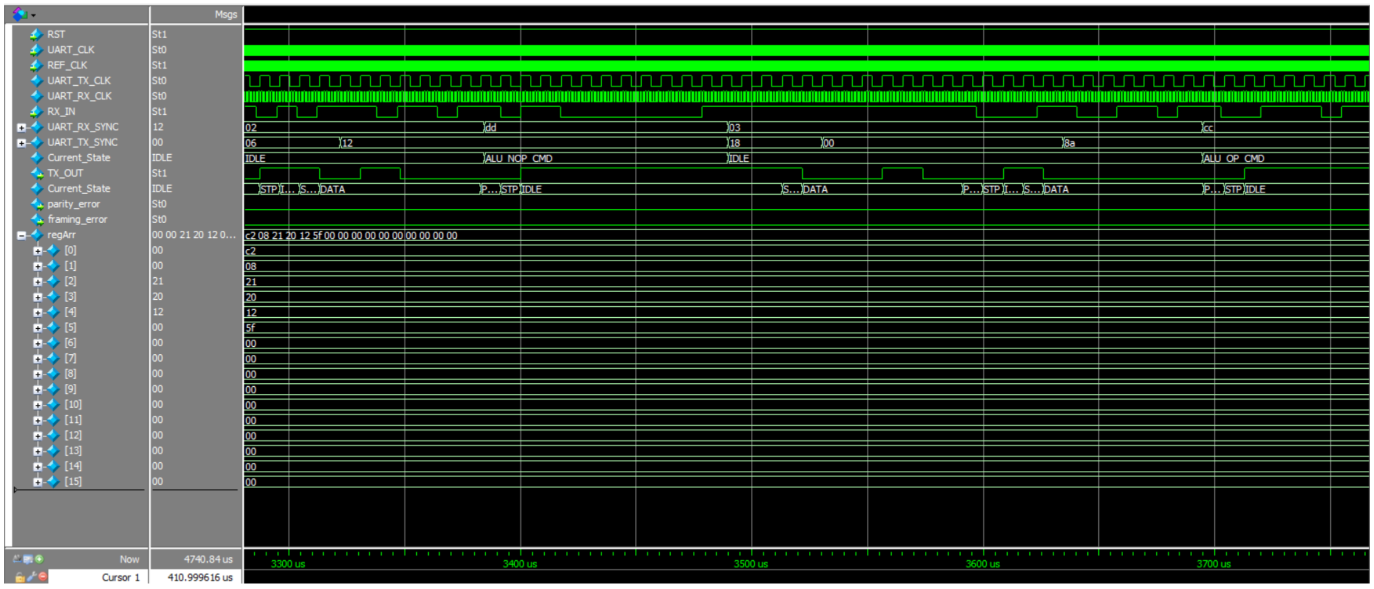
8. Test Case 8: ALU Multiplication Operation with Operands

- **Description:** Perform a multiplication operation with operands (0xC2) and (0x08).
- **Steps**:
  1. Send ALU_OP_CMD (0xCC).
  2. Send OPERAND_A (0xC2).
  3. Send OPERAND_B (0x08).
  4. Send operation code for multiplication (0x02)
- **Waveform:**



9. Test Case 9: ALU Division Operation without Operands

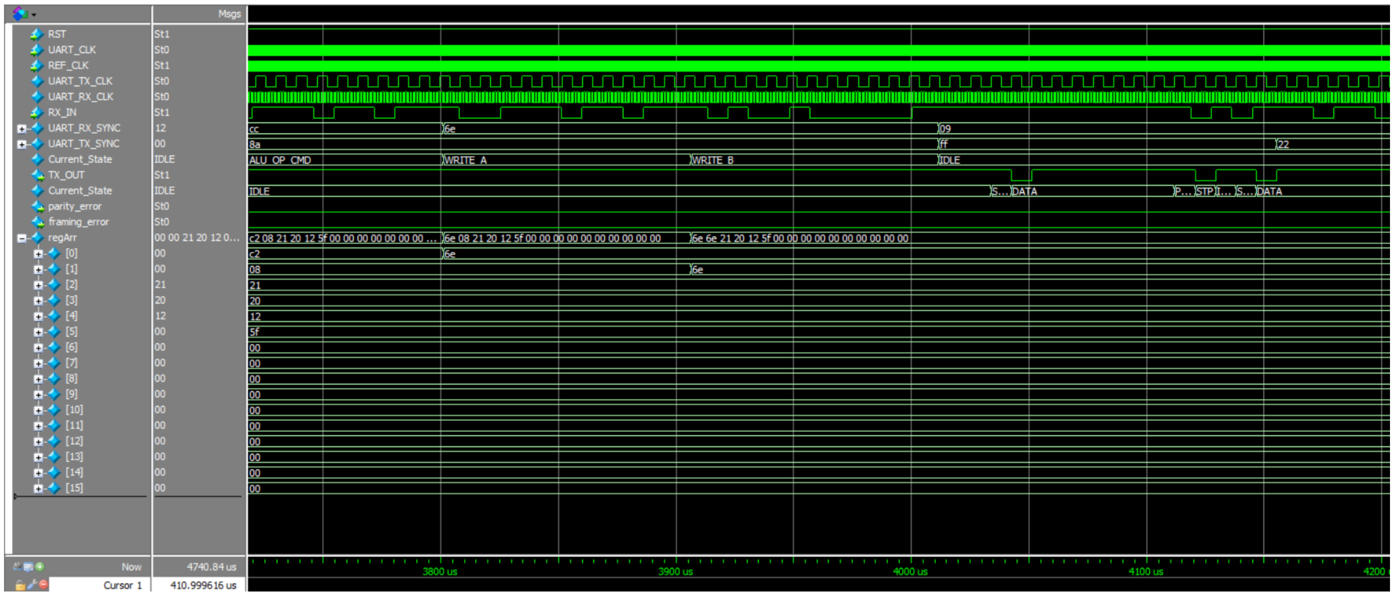
- **Description:** Perform a division operation.
- **Steps**:
  1. Send ALU_OP_CMD (0xDD).
  2. 2. Send operation code for division (0x03).
- **Waveform:**



10. Test Case 10: ALU XNOR Operation with Operands

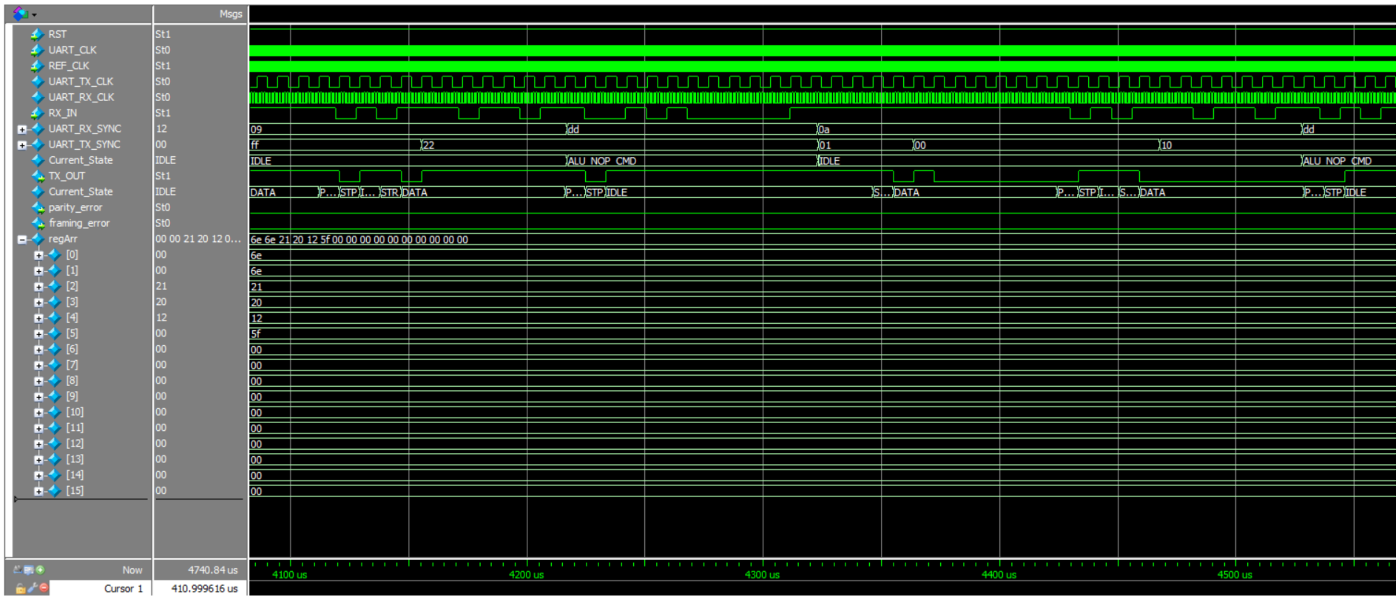
- **Description:** Perform an XNOR operation with operands (0x6E) and (0x6E).
- **Steps**:
  1. Send ALU_OP_CMD (0xCC).
  2. Send OPERAND_A (0x6E).
  3. Send OPERAND_B (0x6E).
  4. Send operation code for XNOR (0x09)

- **Waveform:**



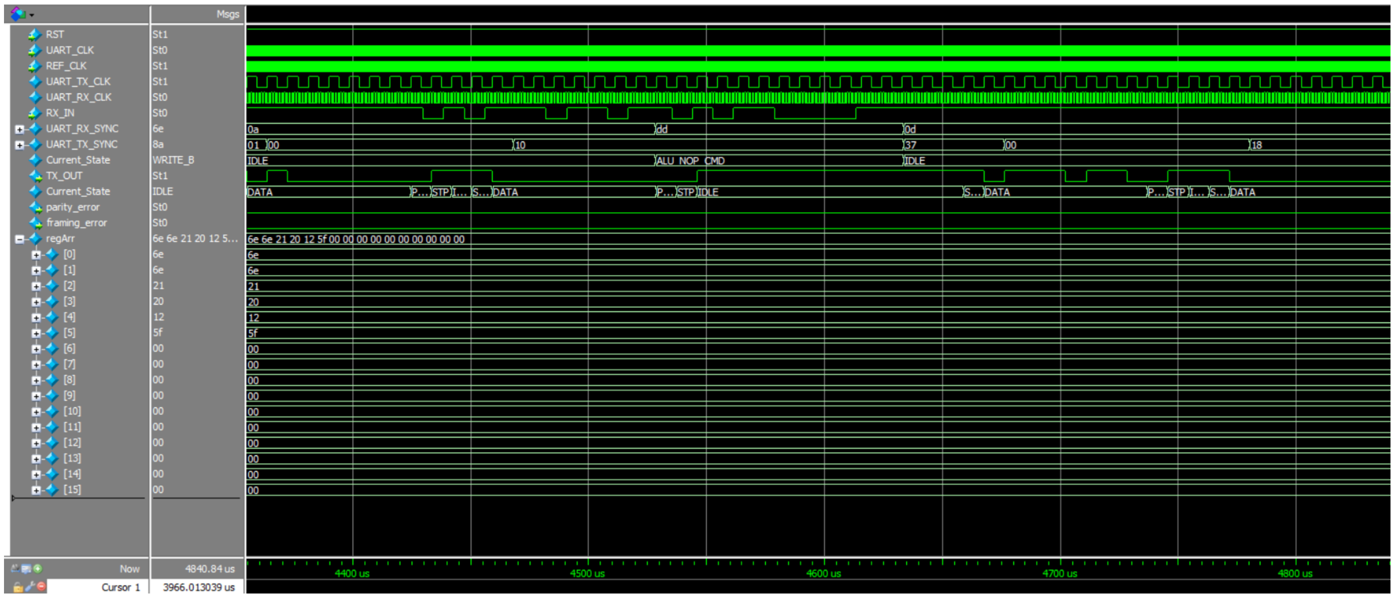
11. Test Case 11: ALU Compare Operation without Operands

- **Description:** Perform a compare operation.
- **Steps**:
  1. Send ALU_OP_CMD (0xDD).
  2. Send operation code for compare (0x0A).
- **Waveform:**



12. Test Case 12: ALU Shift Right Operation

- **Description:** Perform a shift right operation on operand A stored in the register file.
- **Steps**:
  1. Send ALU_OP_CMD (0xDD).
  2. Send operation code for shift right (0x0D).
- **Waveform:**



Conclusion

The testbench successfully verified the functionality of the 'System_TOP' module by simulating various UART commands, register file operations, and ALU operations. The results from the waveform analysis confirm that the module behaves as expected for all the test cases.