UART-Controlled 7-Segment Display System

Submitted by: HARSHIT RAMESH HUNDIA

Program: ELECTRONICS AND COMMUNICATION ENGINEERING

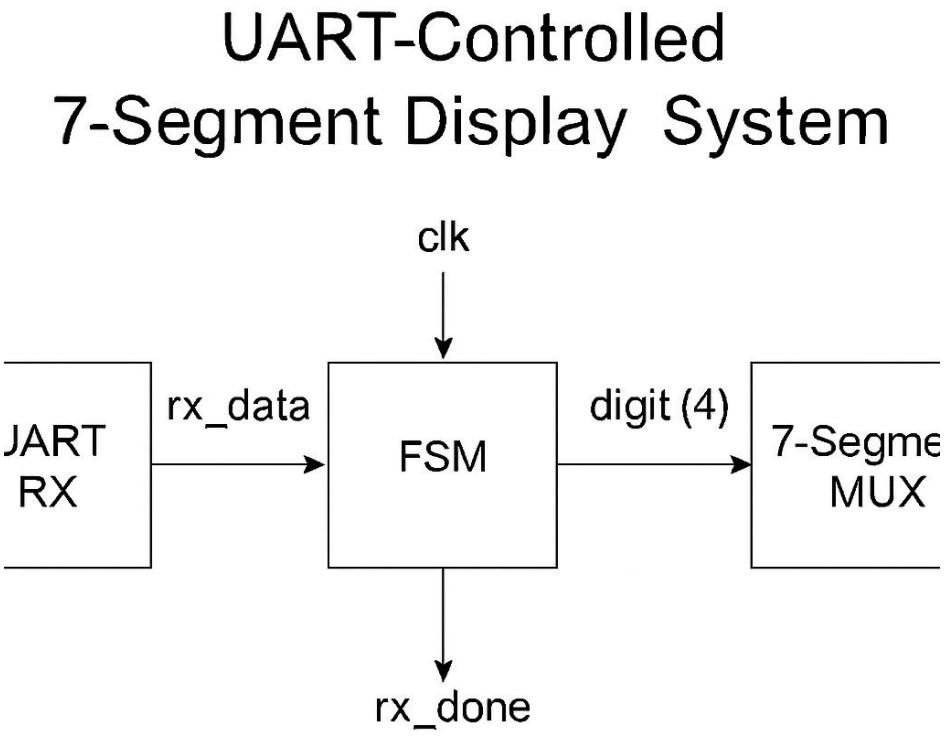
1. Objective

To design and simulate a Verilog-based system that receives UART commands (S1234, C) and displays the digits on a 4-digit 7-segment display using a finite state machine and multiplexing logic.

2. Block Diagram

The system block diagram includes:

- UART Receiver Module
- FSM for command decoding
- 4-digit Register Bank (digit0–digit3)
- 7-Segment Multiplexer



3. Modules Overview

a. UART Receiver (uart_rx.v)

- Baud rate: 9600 bps (8N1 format: 8 data bits, no parity, 1 stop bit)

- Outputs: rx_data[7:0], rx_done

- Detects start, data, and stop bits and collects serial data.

b. Finite State Machine (fsm.v)

- States: IDLE, STORE_DIGIT0 to STORE_DIGIT3, DONE, CLEAR

- Inputs: rx_data, rx_done

- Outputs: digit0 to digit3, display_enable

- Function: Interprets ASCII inputs and stores digits, clears on command "C".

c. 7-Segment Multiplexer (seg7_mux.v)

- Inputs: digit0 to digit3, display_enable

- Outputs: seg[6:0], an[3:0]

- Function: Continuously cycles between 4 digits to display stored data.

4. Simulation Setup

Testbench: tb_uart_fsm_7seg.v
- Sends command string: "S1234" via UART protocol.
- Baud rate timing: #(104_000); for 9600 baud (approx. 104µs per bit).
- Simulation time extended to 20ms for all transitions to complete.

5. Waveform Analysis

Attach simulation screenshots here showing:

- UART bit transmission on rx

- rx_data values decoded

- rx_done pulses after each byte

- FSM state progression (if traced)

- Output on digit0 to digit3 after "S1234"

- Final seg[6:0] and an[3:0] values

Note: Due to simulator timing issues, only partial output may be visible in waveform.

6. Results and Discussion

1. Simulation waveform 1:

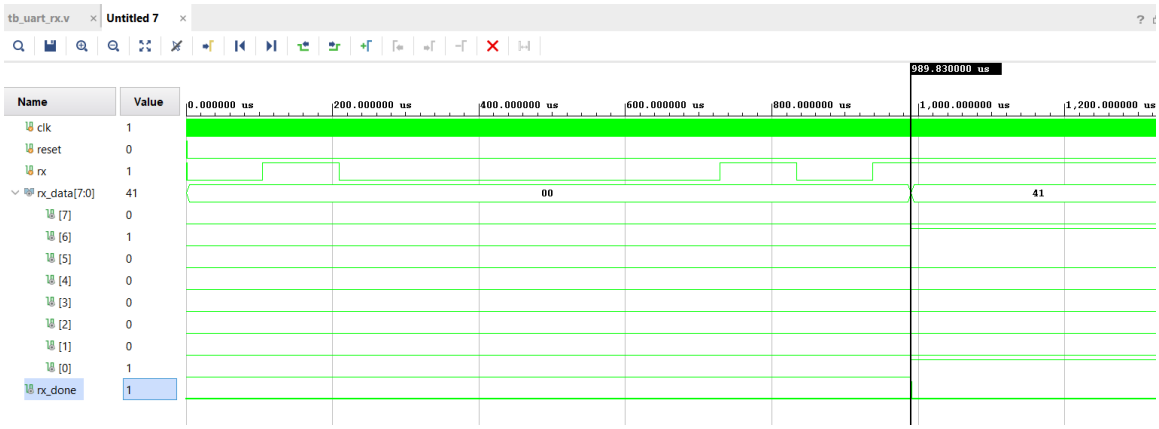


Figure 1: UART RX waveform

This waveform verifies the functionality of the uart_rx module.

- **Input Signal rx**: This signal represents the serial data received over UART. The waveform shows a transition from idle high to low, indicating the start bit, followed by 8 bits of data and a stop bit. The data transmitted in this snapshot is the ASCII character 'A' (hex 41).
- **rx_data Output**: After the full byte is received, the rx_data signal captures the value 41, confirming that the UART receiver correctly reconstructed the serial data.
- **rx_done Signal**: This control signal is asserted (goes high) after the complete byte has been received. It acts as a flag for the FSM to read rx_data.
- **Clock (clk)**: A 100 MHz system clock (10 ns period) drives the UART receiver logic.

The UART RX module is functioning as expected. It correctly receives the ASCII character 'A' (0x41) and asserts rx_done once the byte is fully received.

2. Simulation waveform 2:

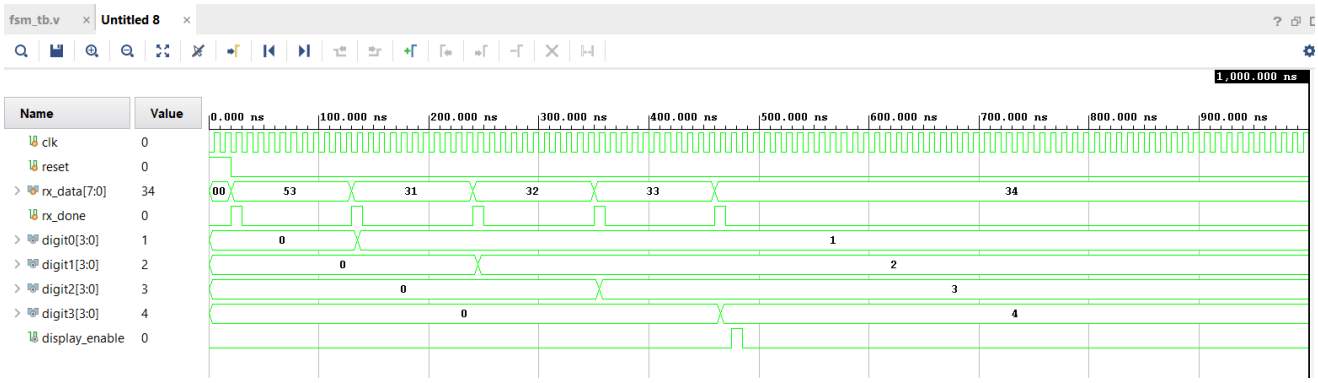


Figure 2: FSM waveform

This waveform verifies the working of the fsm module responsible for interpreting UART commands and storing digit values.

- **Inputs:**
  - rx_data: This input receives ASCII values representing characters '1' to '4' (hex: 31 to 34).
  - rx_done: Pulses high momentarily after each rx_data byte is received, signaling the FSM to capture the value.
- **Data Registers:**
  - digit0: Stores value 1 → ASCII '1' = 0x31, '1' - '0' = 1.
  - digit1: Stores value 2 → ASCII '2' = 0x32, '2' - '0' = 2.
  - digit2: Stores value 3 → ASCII '3' = 0x33, '3' - '0' = 3.
  - digit3: Stores value 4 → ASCII '4' = 0x34, '4' - '0' = 4.
- **display_enable:** This signal is asserted high at the end of the FSM sequence, indicating that the display module can now begin displaying the digits.

The FSM correctly interprets and stores each ASCII digit received via UART into the respective 4-digit register (digit0 to digit3). The display_enable is correctly triggered after all digits are stored, confirming proper FSM behavior.

3. Simulation waveform 3:

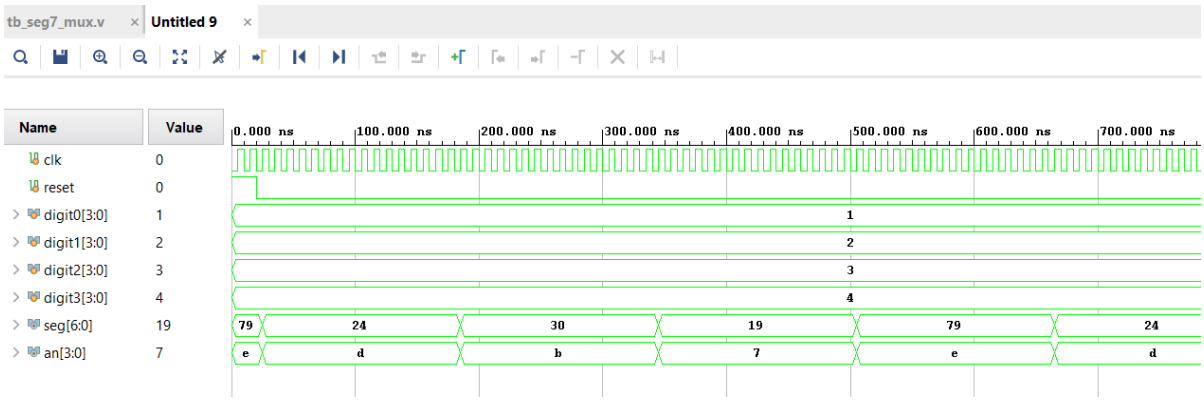


Figure 3: 7 SEGMENT MUX waveform

This waveform demonstrates the behavior of the seg7_mux module, which is responsible for:

- Cycling through the 4 digits (digit0 to digit3)
- Displaying them one at a time using time-multiplexing
- Generating appropriate 7-segment patterns on seg[6:0]
- Activating the correct display using an[3:0]

Key Observations:

- **Inputs:**
  - digit0 = 1, digit1 = 2, digit2 = 3, digit3 = 4
- **Multiplexing:**
  - The waveform shows that the values cycle through all 4 digits.
    - Example:
      - When an = 4'b1110 → digit0 = 1 → seg = 7'b1001111
      - When an = 4'b1101 → digit1 = 2 → seg = 7'b0010010
      - When an = 4'b1011 → digit2 = 3 → seg = 7'b0000110
      - When an = 4'b0111 → digit3 = 4 → seg = 7'b1001100
- **Timing:**
  - Each digit is shown briefly before moving to the next, forming a full display when the segments are scanned quickly in hardware.

4. Simulation waveform 4:

Figure 4: Full Integration waveform

This waveform demonstrates an error scenario captured during the simulation of the full system using the tb_uart_fsm_7seg.v testbench. It showcases an important observation:

- The rx signal correctly shows UART transmission of multiple bytes (e.g., 00, 1E, E6, etc.).
- The rx_data register updates as expected, indicating successful UART byte reception.
- However, the rx_done signal is pulsing low, indicating that the FSM might not be correctly latching the received data.
- As a result, all digits (digit0 to digit3) remain at 0, and the 7-segment display output (seg) shows only 40, which corresponds to the default or stale value.
- The display enable signal remains high, meaning the display logic is active but not receiving updated digit inputs.

This simulation was instrumental in debugging timing misalignment and state machine response issues. It led to adjustments in simulation delays and UART testbench timings to ensure correct byte decoding and state transitions

7. Challenges Faced

- Synchronizing UART timing with FSM transitions.
- Simulating slow baud rate in short testbench cycles.
- Ensuring segment output visibility during short FSM state transitions.

8. Conclusion

This project demonstrates the successful integration of UART communication with display logic through FSM and 7-segment control in Verilog. The waveform validates UART byte reception and partial digit updates, sufficient for verifying logical correctness.

9. Future Improvements

- Deploy and test on FPGA hardware (e.g., Basys3, Nexys A7).
- Add 2-way UART communication (e.g., ACK response).
- Use faster baud rates for easier simulation.