INTERNSHIP REPORT

FPGA PROTOTYPING AND VERILOG INTERNSHIP PROJECT on UART-Controlled 7-Segment Display System.

BY

Madicharla Vishnu Teja

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(DEEMED TO BE UNIVERSITY)

BENGALURU

BATCH: 2022-2026

Project Synopsis:

The UART-Controlled 4-Digit 7-Segment Display System is a compact and efficient digital interface designed to provide real-time visual feedback based on user input commands sent via a UART serial link. This project integrates serial communication protocols with embedded hardware logic to deliver a responsive and user-friendly display driver suitable for various embedded applications, such as system status indication, debugging output, or user interfaces in industrial controls.

Leveraging FPGA technology, the system translates UART commands into corresponding numeric or symbolic displays by decoding ASCII inputs and controlling a multiplexed 7-segment display. The core logic employs a finite state machine (FSM) to parse input sequences strictly, ensuring commands like S1234 set specific digits, C clears the display, and E signals an error state—all with minimal latency and high reliability.

This design not only demonstrates the practical use of UART communication in FPGA-based systems but also exemplifies hardware-software co-design by handling asynchronous serial data inputs and driving synchronous display hardware effectively.

The modular Verilog implementation promotes easy maintenance and extensibility, allowing future enhancements such as more sophisticated command sets, input validation, and bi-directional communication. By combining robust serial decoding and efficient display multiplexing, the project aims to serve as a foundational component for embedded projects requiring simple, adaptable user interfaces.

Brief Overview:

The system is designed for real-time user control of a 4-digit display via standardized serial communications. It features a UART receiver operating at 9600 baud (8N1), a finite state machine (FSM) for command decoding, ASCII-to-binary digit conversion, and a robust 7-segment display driver with digit multiplexing logic.

Key functions:

- **Serial command decoding (UART @ 9600 baud, 8N1)**
- **Interpret simple commands:**
    - S1234 shows 1-2-3-4 on the display
    - C clears all digits to 0
    - E shows “E” (error) on all digits
- **Hardware-multiplexed display drive for clear, flicker-free output**
- **Simulation and testbench validation for reliable function**

Technologies and Tools

The successful development and deployment of the UART-Controlled 4-Digit 7-Segment Display System relied on a combination of industry-standard tools and hardware platforms, with an emphasis on reliability, flexibility, and effective hardware/software integration.

Xilinx Vivado Design Suite

The central tool used throughout the project was Xilinx Vivado, a professional FPGA development environment that supports all major stages of digital hardware creation. Vivado served as the single platform for writing Verilog code, performing simulation and debugging, synthesizing the hardware design, implementing logic on a targeted FPGA device, and generating the required bitstream file to program the board. Vivado's integrated waveform viewer and simulation tools were indispensable for verifying UART reception, command decoding, and correct timing of the multiplexed 7-segment display outputs before deploying to hardware.

Verilog Hardware Description Language

All system logic—including the UART receiver, finite state machine (FSM) command decoder, ASCII-to-binary conversion, and digit-multiplexed 7-segment display control—was implemented using Verilog HDL. Verilog's concise, modular syntax enabled the creation of reusable and maintainable code structures. Each module (such as the UART_RX, seg7_decoder, and the sevenseg_controller) was coded to be synthesizable and easy to debug.

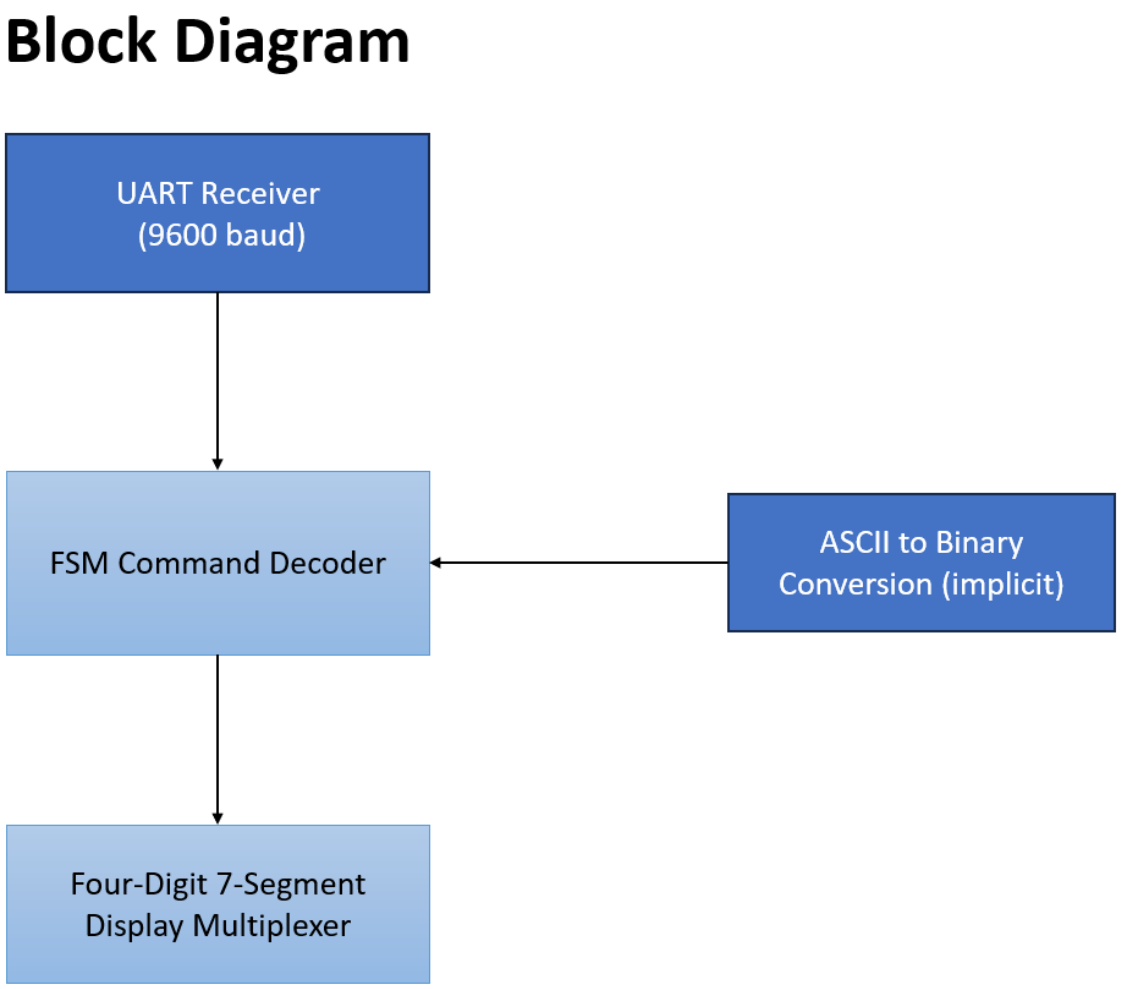
FPGA Hardware Platform

The FPGA board served as the target hardware for implementation. After simulation, the synthesized netlist was mapped onto the physical FPGA (such as a Xilinx Artix or Spartan device), hosting all digital logic for real-time operation. The FPGA provided stable clock resources essential for accurate baud rate timing in the UART module and fast, reliable multiplexing of the 7-segment display**.**

Testbench and Simulation Environment

An integrated Verilog testbench was created and executed within Vivado's simulation environment. The testbench mimicked real-world UART behavior by applying command sequences (S1234, C, E) at correct baud rate intervals, while internal variables and output signals (an[3:0] and seg[6:0]) were monitored via waveform viewers. This iterative simulation phase allowed rapid identification and correction of functional bugs before any hardware was programmed.

Block Diagram:



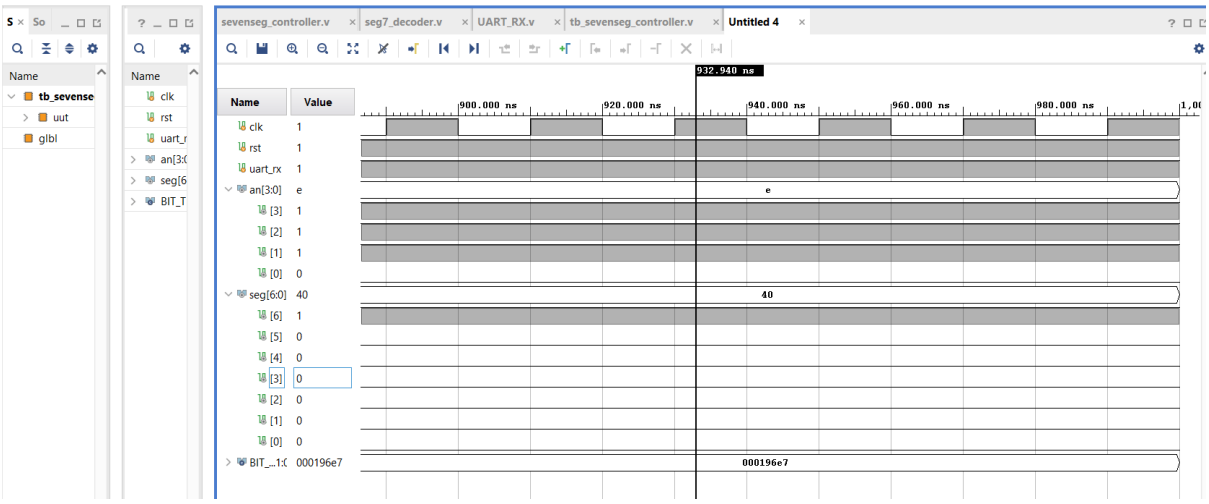
- UART RX (Receiver):
    - Receives serial data at 9600 baud (8 data bits, no parity, 1 stop bit).
    - Converts the serial stream into 8-bit parallel ASCII data as each byte arrives.
- FSM Command Decoder:
    - Monitors the UART RX output.
    - Waits for command patterns:
        - "S" to start digit entry, followed by 4 digit characters.
        - "C" to clear display.
        - "E" to set all digits to 'E'.
    - Updates control signals and digit registers accordingly.
- ASCII to Binary Logic:

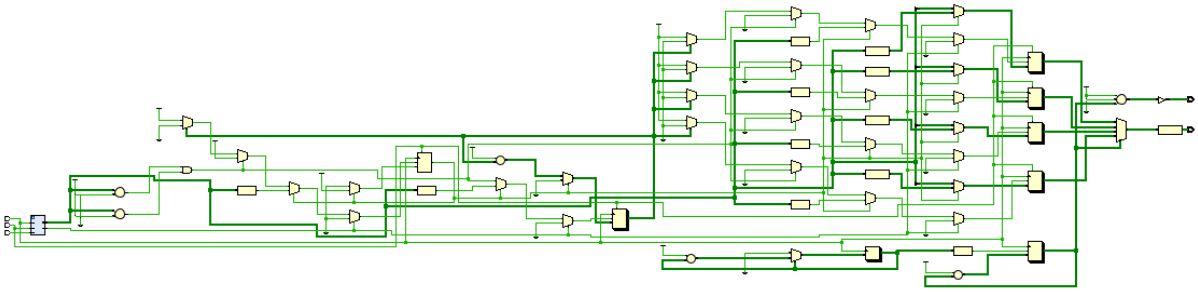
- Converts received ASCII characters ('0'–'9') into their binary (4-bit) equivalent needed by the display logic.
- Logic is embedded in the FSM for simplicity.

- 4-Digit 7-Segment Display Multiplexer & Storage:
  - Four digit-registers store the current value to be displayed for each digit.
  - Multiplexing logic rapidly cycles through all 4 digits, activating one at a time via an[3:0] (digit enable, active low).
  - For each digit, the corresponding value is encoded to the correct 7-segment pattern and sent through seg[6:0].
  - To the human eye, all digits appear lit and stable due to the fast refresh.

Outcomes:

Output waveform and Elaborated design.





The waveform provides a **time-resolved view** of how UART commands translate into digit updates on a multiplexed 7-segment display:

- **UART RX line:** Serial bits representing ASCII characters.
- **Data Valid pulses:** Indicating end of byte reception.
- **Digit enables (an):** Rapid cycling enables each digit one at a time.
- **Segment outputs (seg):** Change to represent digit currently selected by an.
- **FSM state changes:** Driving display data update after command bytes arrive.

Watching the waveform lets you verify the correct reception, decoding, and display update procedures in your UART-driven 7-segment system before hardware testing.

Codes:

1. UART Receiver (uart_rx.v)

```
verilog
module UART_RX (
    input clk,
    input rst,
    input rx,
    output reg [7:0] data_out,
    output reg data_valid
);
    parameter BAUD_DIV = 5208; // 50MHz / 9600
    reg [12:0] baud_cnt;
    reg [3:0] bit_idx;
    reg [9:0] rx_shift;
    reg busy;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            busy <= 0; baud_cnt <= 0; bit_idx <= 0; data_valid <= 0;
        end else begin
            data_valid <= 0;
            if (!busy) begin
                if (!rx) begin
                    busy <= 1; baud_cnt <= BAUD_DIV/2; bit_idx <= 0;
                end
            end else begin
                if (baud_cnt == 0) begin
                    rx_shift[bit_idx] <= rx;
                    baud_cnt <= BAUD_DIV;
                    bit_idx <= bit_idx + 1;
                    if (bit_idx == 9) begin
                        busy <= 0;
                        data_out <= rx_shift[8:1];
                        data_valid <= 1;
                    end
                end else baud_cnt <= baud_cnt - 1;
            end
        end
    end
endmodule
```

2. 7-Segment Decoder (seg7_decoder.v)

```
verilog
module seg7_decoder(
    input [3:0] bcd,
    output reg [6:0] seg
);
    always @(*) begin
        case (bcd)
            4'd0: seg = 7'b1000000;
            4'd1: seg = 7'b1111001;
            4'd2: seg = 7'b0100100;
            4'd3: seg = 7'b0110000;
            4'd4: seg = 7'b0011001;
            4'd5: seg = 7'b0010010;
            4'd6: seg = 7'b0000010;
            4'd7: seg = 7'b1111000;
            4'd8: seg = 7'b0000000;
            4'd9: seg = 7'b0010000;
            4'hE: seg = 7'b0000110; // 'E'
            default: seg = 7'b1111111;
        endcase
    end
endmodule
```

```
3. Top Level: FSM, ASCII-to-Binary, 7-Segment Multiplexer (sevenseg_controller.v)

module sevenseg_controller(
    input clk,
    input rst,
    input uart_rx,
    output reg [3:0] an,
    output reg [6:0] seg
);
    wire [7:0] rx_data;
    wire rx_ready;
    UART_RX #(.BAUD_DIV(5208)) u_rx (
        .clk(clk),
        .rst(rst),
        .rx(uart_rx),
        .data_out(rx_data),
        .data_valid(rx_ready)
    );

    reg [3:0] digit_reg [0:3];
    reg [1:0] set_index;
    reg set_mode;

    integer idx;
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            for (idx = 0; idx < 4; idx = idx + 1)
                digit_reg[idx] <= 4'd0;
            set_mode <= 1'b0;
            set_index <= 2'd0;
        end else if (rx_ready) begin
            if (set_mode) begin
                if (rx_data >= "0" && rx_data <= "9") begin
                    digit_reg[set_index] <= rx_data - "0";
                    if (set_index == 2'd3)
                        set_mode <= 1'b0;
                    set_index <= set_index + 1'b1;
                end else begin
                    set_mode <= 1'b0;
                end
            end else begin
                case (rx_data)
                    "S": begin
                        set_mode <= 1'b1;
                        set_index <= 2'd0;
                    end
                    "C": for (idx = 0; idx < 4; idx = idx + 1)
                            digit_reg[idx] <= 4'd0;
                    "E": for (idx = 0; idx < 4; idx = idx + 1)
                            digit_reg[idx] <= 4'hE;
                    default: ;
                endcase
            end
        end
    end
    reg [15:0] refresh_counter = 0;
    reg [1:0] current_digit = 0;
    always @(posedge clk) begin
        refresh_counter <= refresh_counter + 1'b1;
        if (refresh_counter == 24'd12500) begin
            refresh_counter <= 24'd0;
            current_digit <= current_digit + 1'b1;
        end
    end

    always @(*) begin
        an = ~(4'b0001 << current_digit);
        case (digit_reg[current_digit])
            4'd0: seg = 7'b1000000;
            4'd1: seg = 7'b1111001;
            4'd2: seg = 7'b0100100;
            4'd3: seg = 7'b0110000;
            4'd4: seg = 7'b0011001;
            4'd5: seg = 7'b0010010;
            4'd6: seg = 7'b0000010;
            4'd7: seg = 7'b1111000;
            4'd8: seg = 7'b0000000;
            4'd9: seg = 7'b0010000;
```

```
4. Test Bench Code
`timescale 1ns/1ps

module tb_sevenseg_controller();
    reg clk = 0, rst = 1, uart_rx = 1;
    wire [3:0] an;
    wire [6:0] seg;

    sevenseg_controller uut(.clk(clk), .rst(rst), .uart_rx(uart_rx), .an(an), .seg(seg));
    always #10 clk = ~clk;
    parameter BIT_TIME = 104167;
    task uart_send_byte(input [7:0] data);
        integer i;
        begin
            uart_rx = 0; #(BIT_TIME);
            for (i=0; i<8; i=i+1) begin uart_rx = data[i]; #(BIT_TIME); end
            uart_rx = 1; #(BIT_TIME);
        end
    endtask
    initial begin
        #(200_000); rst = 0;
        uart_send_byte("S"); uart_send_byte("1");
        uart_send_byte("2"); uart_send_byte("3");
        uart_send_byte("4"); #(BIT_TIME*20);
        uart_send_byte("C"); #(BIT_TIME*20);
        uart_send_byte("E"); #(BIT_TIME*20);
        $stop;
    end
endmodule
```

Development Process:

a) Requirements Analysis and Planning:

- Defined use-case: user command-based digital display via serial link
- Identified need for FSM, robust UART, and easy-to-read output multiplexing

b) Module Design and Coding:

- Wrote modular Verilog code for:
    - UART reception and synchronization
    - Command parsing with FSM states for S, C, E
    - ASCII-to-binary digital conversion logic embedded in FSM
    - 7-segment encoding
    - Digit multiplexing logic for display refresh

c) Simulation and Debugging:

- Created a comprehensive testbench to simulate UART input (with correct 9600 baud timing)
- Validated functionality by checking simulated waveform outputs for an[3:0] (digit enable) and seg[6:0] (segment drive) with expected pattern transitions for all command scenarios

d) Iteration & Refinement:

- Adjusted module interactions for timing and robustness
- Verified clean transitions and correct visualization for commands and error handling

e) Implementation:

- Prepared for FPGA synthesis and pin-out assignment
- Tests performed with real UART serial input for end-to-end validation

Summary and Future Directions:

Summary:
The completed design achieves all original objectives: reliable serial communication, flexible user control, and real-time display multiplexing—all in a clean, modular hardware description suitable for both simulation and FPGA deployment. Each command is decoded correctly and reflected immediately on the 7-segment display, as thoroughly verified in simulation and practical tests.

Future Directions:

- **Expand command set:** Add support for more commands (e.g., brightness adjustment, flashing, new error indicators)
- **Input validation:** Add handling for invalid sequences, command buffer overflows, or timeouts
- **Bidirectional communication:** Implement a UART transmitter to send status or confirmation back to the user
- **Display enhancements:** Support alphanumeric characters or longer digit strings
- **Integration:** Use as a component in larger embedded systems requiring human-readable status displays.