MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Department of Electronics & Communication Engineering

FPGA Lab – IV Semester

MINI PROJECT REPORT

Digital Clock: Design and Display

Submitted By:

Name	Roll No	Registration No	Batch
Anshu Chakraborty	11	230959042	A1
Sidharth Sarangi	17	230959062	A1
Devjeet Nanda	19	230959068	A1

	CONTENTS
1.	Description about the Project
2.	Methodology
3.	Block Diagram
4.	Code
5.	Implemented Design
6.	Simulations and result analysis
7.	References

Introduction

This project involves designing a digital clock using a Field Programmable Gate Array (FPGA) and displaying the time on an external display. The clock will keep track of hours, minutes and will be capable of resetting, adjusting, and updating the time in real-time.

Objective

- To design a **real-time digital clock** using FPGA.
- To implement timekeeping functions using **Verilog**.
- To utilize FPGA's internal **counters and registers** for time calculations.
- To display time using a **seven-segment display or LCD**.

Project Features:

1. Timekeeping: The clock will maintain accurate time by counting seconds, minutes, and hours.
2. Display Interface: The time will be displayed on an external display, which is a 7-segment LED display on the board itself, controlled by the FPGA.
3. Modularity: The design will be modular, separating the timekeeping logic from the display and control logic.

Technical Requirements:

1. FPGA Board: A suitable FPGA development board (e.g., Xilinx) for implementing the digital clock. We are using BASYS-3 FPGA board for this.
2) Programming Language: Verilog for designing the clock logic.
3) External Display: A 7-segment display or LCD for outputting the time.
4) Power Supply: An appropriate power source for the FPGA and display.
5) Input Controls: Buttons for time adjustment and reset.

Challenges

1. Implementing precise timekeeping using the FPGA's internal clock.
2. Efficiently driving the seven segment display on the FPGA.

Methodology

Upon completion, the project will deliver a fully functional digital clock displayed on an external display, controlled and managed by an FPGA. This will demonstrate an understanding of digital design, clock management, and hardware interfacing.

This description outlines the scope, objectives, and technical aspects of the project, providing a clear roadmap for development.

Step 1: Define System Requirements

Identify the basic functionalities of the digital clock: timekeeping (hours, minutes, seconds), time adjustment, and display.

Step 2: Design Clock Logic

Clock Divider: Implement a clock divider to generate a 1 Hz(from a 100MHz internal clock signal) clock signal from the FPGA's internal clock.

Counter Design: Develop counters for seconds, minutes, and hours.

Control Logic: Design logic for handling time adjustments (setting hours, minutes) and resetting the clock.

Step 3: Display Interface

Design the interface between the FPGA and the external display.

Implement the necessary drivers to convert the binary time data into a format suitable for the display (e.g., BCD for a 7-segment display).

Step 4: Input Handling

Implement logic to handle user inputs for resetting the clock.

Step 5: Simulation and Verification

Simulate the design using Vivado to verify the functionality of the clock logic.

Test individual modules (clock divider, counters, display drivers) before integrating them into the complete system.

Step 6: FPGA Implementation

Synthesize the Verilog code and upload it to the FPGA.

Test the digital clock in real hardware to ensure it operates as expected.

Step 7: Testing and Debugging

Conduct thorough testing to check for accuracy in timekeeping and response to user inputs.

Debug any issues related to timing, display, or input handling.

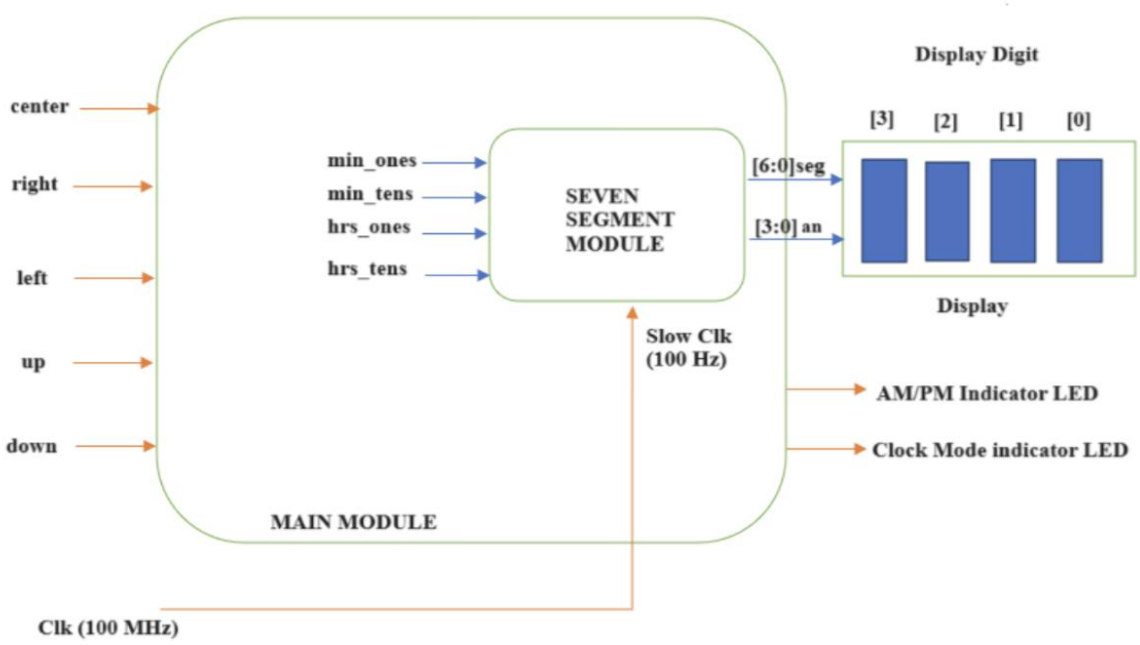
Make any necessary adjustments to the design.

Step 8: Documentation and Finalization

Document the design process, including the design rationale, code, and testing results.

Finalize the project with a working demonstration of the digital clock.

Block Diagram



Code

```
module digital_clock(
    input clk_100MHz,              // 100MHz system clock
    input reset_pb,                // Reset push-button to reset
the clock to 12:00
    output [6:0] seg,              // Seven segment display
output (for digits)
    output [3:0] an                // Seven segment display
anode control
);


    // Clock and time variables
    reg [31:0] counter = 0;
```

```
    parameter max_count = 10_00_00_000;  // 100MHz / 1Hz = 100M / 2 (for low and high pulses) = 50M
    reg [5:0] hrs, min = 0;     // hours and minutes (0-59 for minutes, 0-23 for hours)
    reg [3:0] min_ones, min_tens, hrs_ones, hrs_tens = 0; // Seven-segment digits

    // Seven-segment display module instantiation
    seven_seg ssd (
        .clk_100MHz(clk_100MHz),   // Pass the clock to the seven-segment display module
        .min_ones(min_ones),
        .min_tens(min_tens),
        .hrs_ones(hrs_ones),
        .hrs_tens(hrs_tens),
        .seg(seg),
        .an(an)
    );

    // Main clock logic with reset and counter
    always @(posedge clk_100MHz or posedge reset_pb) begin
        if (reset_pb) begin
            // Reset time to 12:00 when reset button is pressed
            hrs <= 12;
            min <= 0;
            counter <= 0;
        end else begin
            // Increment the counter to keep track of time
            if (counter < max_count)
                counter <= counter + 1;
            else begin
                counter <= 0;
                min <= min + 1;  // Increment minutes
```

```
                if (min >= 60) begin
                    min <= 0;
                    hrs <= hrs + 1;  // Increment hours when
minutes reach 60
                    if (hrs >= 24) begin
                        hrs <= 0;  // Reset hours after 24
                    end
                end
            end
        end

        // Update the digits for the seven-segment display
        min_ones <= min % 10;  // Get ones digit of minutes
        min_tens <= min / 10;  // Get tens digit of minutes

        if (hrs < 10) begin
            hrs_ones <= hrs;  // For hours 0-9
            hrs_tens <= 0;    // Tens digit for hours 0-9 will
be 0
        end else begin
            hrs_ones <= hrs % 10; // Get ones digit of hours
            hrs_tens <= hrs / 10; // Get tens digit of hours
        end
    end
endmodule
```

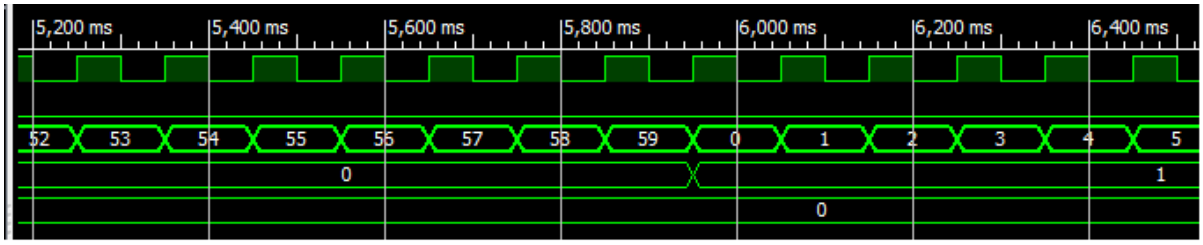
Code for the 7-segment display:

```
module seven_seg_disp_module(
    input clk_100MHz,
    input [3:0] min_ones, min_tens, hrs_ones, hrs_tens,
    output reg [6:0] seg,
    output reg [3:0] an
);
```

```
// Declaring registers and wires
reg [1:0] digit_display_ssd = 0;
reg [6:0] display_ssd [3:0];
reg [18:0] counter = 0;
parameter max_count = 500000; // 100MHz/100Hz = 1M/2 (for low
and high pulses) = 500,000;
wire [3:0] four_bit_ssd [3:0];

// Assigning values that need to be displayed on the SSD
assign four_bit_ssd[0] = min_ones;
assign four_bit_ssd[1] = min_tens;
assign four_bit_ssd[2] = hrs_ones;
assign four_bit_ssd[3] = hrs_tens;

// Generating 100Hz slow clock from the inbuilt 100MHz clock
always @(posedge clk_100MHz) begin
    if(counter < max_count)
        counter <= counter + 1;
    else begin
        digit_display_ssd <= digit_display_ssd + 1;
        counter <= 0;
    end

    // BCD to seven segment display
    case(four_bit_ssd[digit_display_ssd])
   4'b0000: display_ssd[digit_display_ssd] <= 7'b0000001; // 0
   4'b0001: display_ssd[digit_display_ssd] <= 7'b1001111; // 1
   4'b0010: display_ssd[digit_display_ssd] <= 7'b0010010; // 2
   4'b0011: display_ssd[digit_display_ssd] <= 7'b0000110; // 3
   4'b0100: display_ssd[digit_display_ssd] <= 7'b1001100; // 4
```

```
  4'b0101: display_ssd[digit_display_ssd] <= 7'b0100100; // 5
  4'b0110: display_ssd[digit_display_ssd] <= 7'b0100000; // 6
  4'b0111: display_ssd[digit_display_ssd] <= 7'b0001111; // 7
  4'b1000: display_ssd[digit_display_ssd] <= 7'b0000000; // 8
  4'b1001: display_ssd[digit_display_ssd] <= 7'b0001000; // 9
    default: display_ssd[digit_display_ssd] <= 7'b1111111;
    endcase

    // Enabling each segment and displaying the digit
    case(digit_display_ssd)
        2'b00: begin
            an <= 4'b1110; // Enable first display (min_ones)
            seg <= display_ssd[0];
        end
        2'b01: begin
            an <= 4'b1101; // Enable second display (min_tens)
            seg <= display_ssd[1];
        end
        2'b10: begin
            an <= 4'b1011; // Enable third display (hrs_ones)
            seg <= display_ssd[2];
        end
        2'b11: begin
            an <= 4'b0111; // Enable fourth display (hrs_tens)
            seg <= display_ssd[3];
        end
    endcase
end
endmodule
```

Simulation



Implemented Design

Simulation and Result Analysis:

Upon completion, the project delivers a fully functional digital clock displayed on a seven segment displayed, controlled and managed by the FPGA. This will demonstrate an understanding of digital design, clock management, and hardware interfacing.

This description outlines the scope, objectives, and technical aspects of the project, providing a clear roadmap for development.

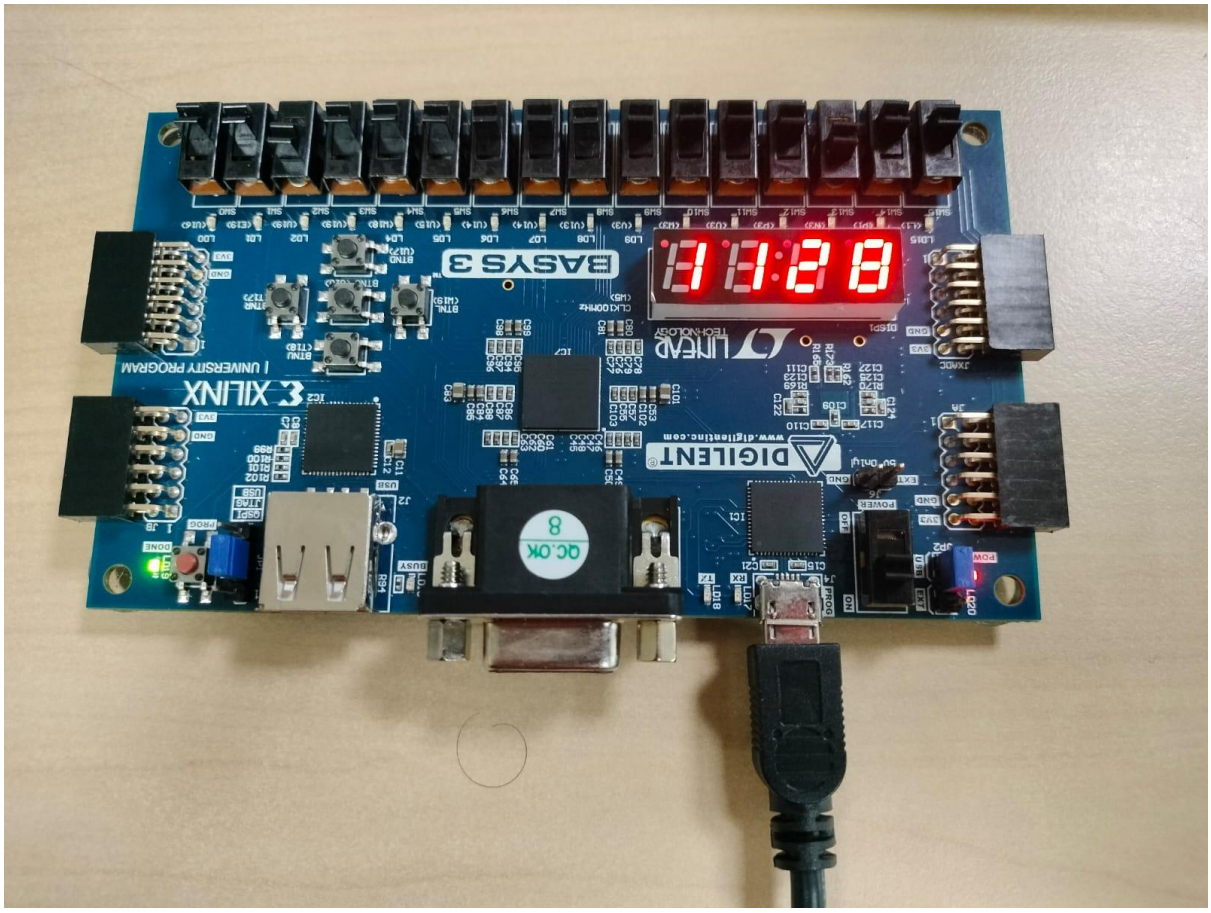


FIG: FPGA showing time

References:

1))www.idaratech.net

2)https://www.instructables.com/

3) www.wikepedia.com

4)www.instructables.com

5) digilent.com/reference/programmable-logic/nexys-4-ddr/reference-manual