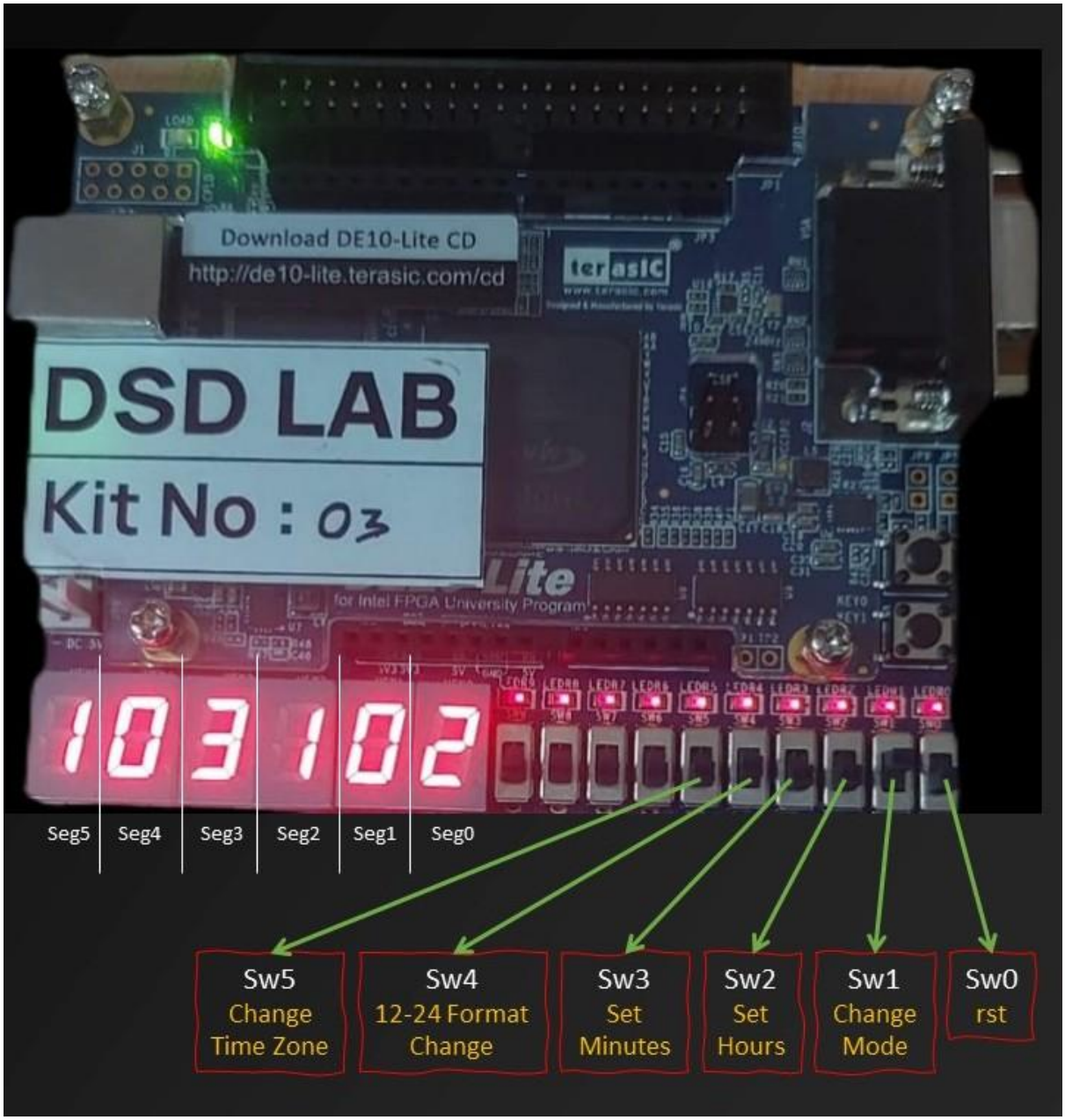
DIGITAL CLOCK USING SEVEN SEGMENT DISPLAY

DESCRIPTION OF THE PROJECT:



PIN	FUNCTION	
SW0	Reset	Sets clock to 00:00:00
SW1	Mode Change	Allows setting time
SW2	Set Hours	
SW3	Set Minutes	
SW4	Toggle 12-24 hour mode	High-12hr ,Low-24 hr
SW5	Switch Time Zone	High-IST,Low-GMT+3:30

This project involves implementing a digital clock using Verilog on an FPGA-DE10 board. The clock employs a 50 MHz input clock to manage real-time hours, minutes, and seconds, with the output displayed on 7-segment displays. Key features include a time-setting mode, options for 12/24-hour formats, and a +2 hour time zone adjustment with wrap-around support.

Key Features:

1. **Real-Time Clock Functionality**:
    - Utilizes the 50 MHz system clock to generate a 1-second pulse.
    - Tracks seconds, minutes, and hours, handling time wrap-around
2. **Time Settings**:
    - **Time Setting Mode**: A switch allows the user to enter time-setting mode, where physical buttons can increment hours and minutes manually.
    - **12/24 Hour Mode**: : A toggle switch allows switching between 12-hour and 24-hour formats, correctly managing midnight (0:00) and noon (12:00).
3. **Time Zone Adjustment**:
    - An additional switch facilitates a +2 hour time zone adjustment. For instance, if the current time is set to Iran's time zone (GMT +3:30), activating the +2 hour adjustment will shift the time to India's time zone (GMT +5:30 (IST)).
    - This change includes automatic wrap-around (e.g., 23:00 in Iran's time would become 01:00 in India's time).
4. **Display System**:
    - Utilizes six 7-segment displays for hours, minutes, and seconds
    - A decoder function translates binary-coded time digits into 7-segment signals, ensuring readability
5. **Efficient Timekeeping**:
    - A clock divider converts the 50 MHz input clock to a 1 Hz pulse, ensuring accurate time increments.
    - Wrap-around logic is implemented for seconds, minutes, and hours to ensure the clock's continuous and reliable operation

VERILOG CODE:

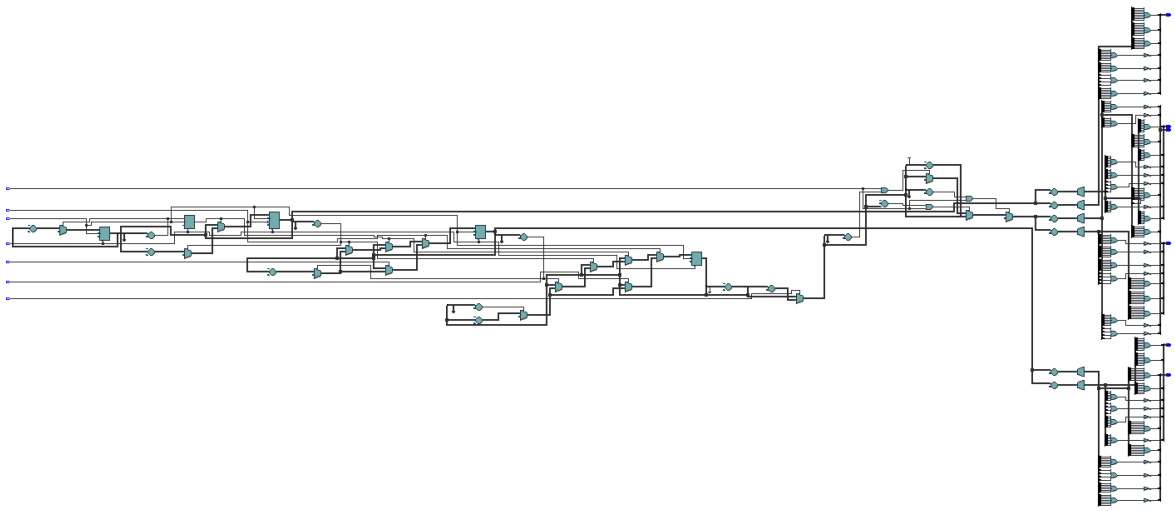
```
module digital_clock(
    input wire clk,
    input wire reset,
    input wire mode_switch,
    input wire button_hours,
    input wire button_minutes,
    input wire hour_mode_switch,
    input wire time_zone_switch,
    output reg [6:0] seg0,
    output reg [6:0] seg1,
    output reg [6:0] seg2,
    output reg [6:0] seg3,
    output reg [6:0] seg4,
    output reg [6:0] seg5
);

    parameter DIVISOR = 50000000;

    reg [25:0] clk_divider = 0;
    reg one_sec_pulse = 0;
    reg [5:0] seconds = 0;
    reg [5:0] minutes = 0;
    reg [4:0] hours = 0;
```

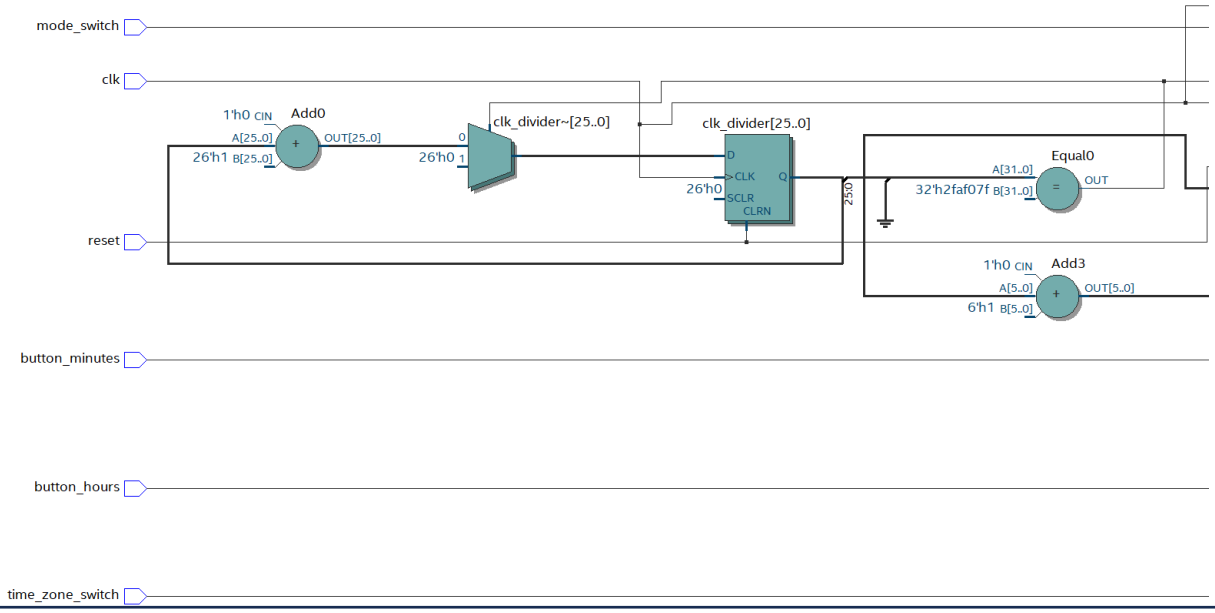
```
   // Time setting
   always @(posedge clk or posedge reset) begin
       if (reset) begin
           hours <= 0;
           minutes <= 0;
           seconds <= 0;
       end else if (mode_switch) begin
           // Time Setting Mode
           if (button_hours) begin
               if (hours == 23)
                   hours <= 0;
               else
                   hours <= hours + 1;
           end

           if (button_minutes) begin
               if (minutes == 59)
                   minutes <= 0;
               else
                   minutes <= minutes + 1;
           end
       end
       else if (one_sec_pulse) begin
           // Normal clock operation
           if (seconds == 59) begin
               seconds <= 0;
               if (minutes == 59) begin
                   minutes <= 0;
                   if (hours == 23)
                       hours <= 0;
                   else
                       hours <= hours + 1;
               end else begin
                   minutes <= minutes + 1;
               end
           end else begin
               seconds <= seconds + 1;
           end
       end
   end
```

```
    // Time Zone Adjustment
    wire [4:0] adjusted_hours;
    assign adjusted_hours = (time_zone_switch) ? ((hours + 2) % 24) : hours;
    // 12/24 Hour Mode Switch
    wire [4:0] display_hours;
    assign display_hours = (hour_mode_switch && (adjusted_hours == 0)) ? 5'd12 :
                           (hour_mode_switch && (adjusted_hours > 12)) ? (adjusted_hours - 12) :
                           (hour_mode_switch && (adjusted_hours == 12)) ? 5'd12 :
                           adjusted_hours;

    // 7-Segment Display Decoder
    function [6:0] seven_segment_decoder;
        input [3:0] digit;
        case (digit)
            4'd0: seven_segment_decoder = 7'b1000000;
            4'd1: seven_segment_decoder = 7'b1111001;
            4'd2: seven_segment_decoder = 7'b0100100;
            4'd3: seven_segment_decoder = 7'b0110000;
            4'd4: seven_segment_decoder = 7'b0011001;
            4'd5: seven_segment_decoder = 7'b0010010;
            4'd6: seven_segment_decoder = 7'b0000010;
            4'd7: seven_segment_decoder = 7'b1111000;
            4'd8: seven_segment_decoder = 7'b0000000;
            4'd9: seven_segment_decoder = 7'b0010000;
            default: seven_segment_decoder = 7'b1111111;
        endcase
    endfunction

    always @(*) begin
        seg0 = seven_segment_decoder(seconds % 10);
        seg1 = seven_segment_decoder(seconds / 10);
        seg2 = seven_segment_decoder(minutes % 10);
        seg3 = seven_segment_decoder(minutes / 10);
        seg4 = seven_segment_decoder(display_hours % 10);
        seg5 = seven_segment_decoder(display_hours / 10);
    end

endmodule
```

RTL VIEW:

(zoomed out)





(zoomed in – input side)

VIDEO LINKS:

1. First Iteration
2. Mode Change Time Setting Buttons
3. Checking if Time is Incrementing Properly After Setting 11:59
4. Time Change Can Only Be Made When Mode Change Switch is High
5. Added 12-24 Hour Functionality
6. Another 24-12 Hour Video
7. Time Zone and Overall Video
8. Verilog code drive link - Click here to view the file