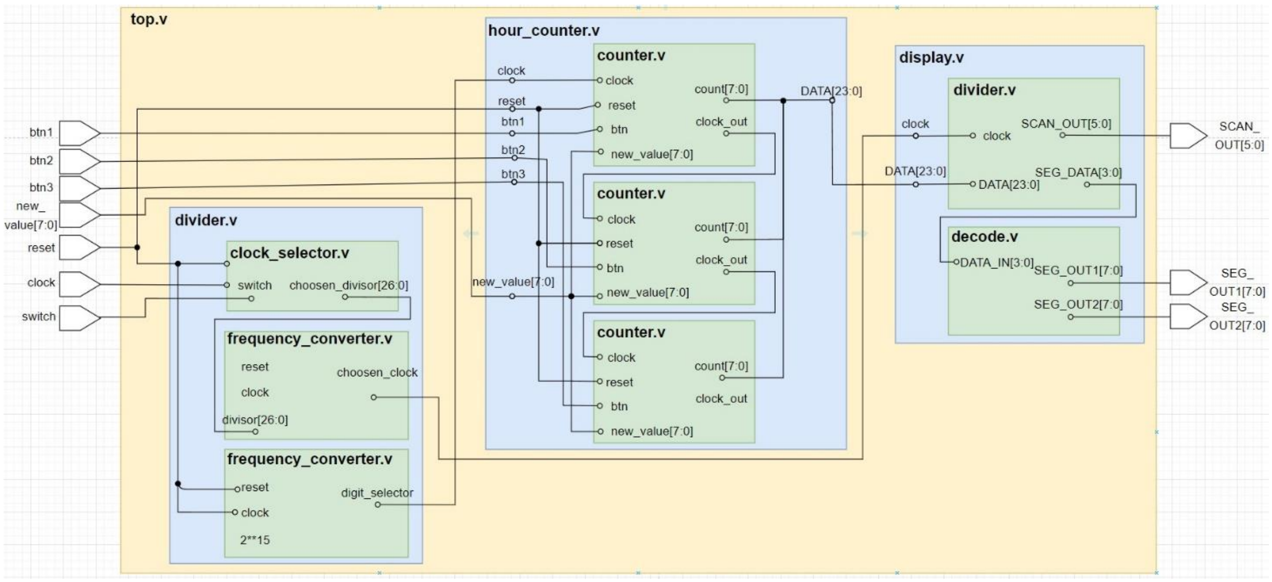
Lab 6

Student ID: F10915108 Name: Gerardo Sigfredo Fisch Paredes

1. Topic: 24 Hour Clock Display

- Task: 24h clock (adjustable time)

2. Circuit Block Diagram:



3. Code:

The program had the following structure

- top.v
  - divider.v
    - clock_selector.v
    - frequency_converter.v
  - hour_counter.v
    - counter.v
  - display.v
    - seg_scan.v
    - decode.v

a) Code:

- top.v

```
module top(reset, clock, switch, btn1, btn2, btn3, new_value, SCAN_OUT, SEG_OUT1, SEG_OUT2);
    input reset, clock, switch, btn1, btn2, btn3;
    input [6:0] new_value;
    output [5:0] SCAN_OUT;            // used to determine which position to display
    output [7:0] SEG_OUT1, SEG_OUT2;    // used to determine which digit to display
    wire choosen_clock;
    wire digit_selector;
    wire [23:0] DATA;

    divider div(reset, clock, switch, choosen_clock, digit_selector);
    hour_counter hr_counter(reset, choosen_clock, btn1, btn2, btn3, new_value, DATA);
    display disp(digit_selector, DATA, SCAN_OUT, SEG_OUT1, SEG_OUT2);
endmodule
```

- divider.v

```
module divider(reset, clock, switch, choosen_clock, digit_selector);
    input reset, clock, switch;
    output choosen_clock;
    output digit_selector;
    wire [26:0] choosen_divisor;

    clock_selector cs(switch, choosen_divisor);
    frequency_converter fc(reset, clock, choosen_divisor, choosen_clock);
    frequency_converter fc2(reset, clock, 2 ** 15, digit_selector);
endmodule
```

- clock_selector.v

```
module clock_selector(switch, choosen_divisor);
    input switch;
    output reg [26:0]choosen_divisor;

    always @(switch) begin
        // If switch = 1, then 0,1 seconds clock
        // else 1 second clock
        choosen_divisor <= (switch) ? 10000000 : 100000000;
    end
endmodule
```

- frequency_converter.v

```
module frequency_converter(reset, clock, divisor, out_clock);
    input reset, clock;
    input [26:0] divisor;
    output reg out_clock;
    reg [26:0] counter = 0;

    initial out_clock = 0;
    always @(posedge clock or posedge reset)
    begin
        counter <= counter + 1;
        if (reset) begin
            out_clock <= 26'b0;
            counter <= 0;
        end
        else begin
            if (counter >= divisor - 1) begin
                counter <= 0;
                out_clock <= ~out_clock;
            end
        end
    end
endmodule
```

- hour_counter.v

```
module hour_counter(reset, clock, btn1, btn2, btn3, new_value, DATA);
    input reset, clock;
    input btn1, btn2, btn3;
    input [7:0] new_value;
    output [23:0] DATA;
    wire timer_seconds, timer_minutes, timer_hours;

    // 8'b01100000 = 60 (BCD), 8'b01000101 = 45 (BCD)
    counter seconds(reset, clock, btn1, new_value, 8'b01100000, 8'b01000101, DATA[7:0], timer_seconds);
    // 8'b01100000 = 60 (BCD), 8'b01011001 = 59 (BCD)
    counter minutes(reset, timer_seconds, btn2, new_value, 8'b01100000, 8'b01011001, DATA[15:8], timer_minutes);
    // 8'b00100100 = 24 (BCD), 8'b00100011 = 23 (BCD)
    counter hours(reset, timer_minutes, btn3, new_value, 8'b00100100, 8'b00100011, DATA[23:16], timer_hours);

endmodule
```

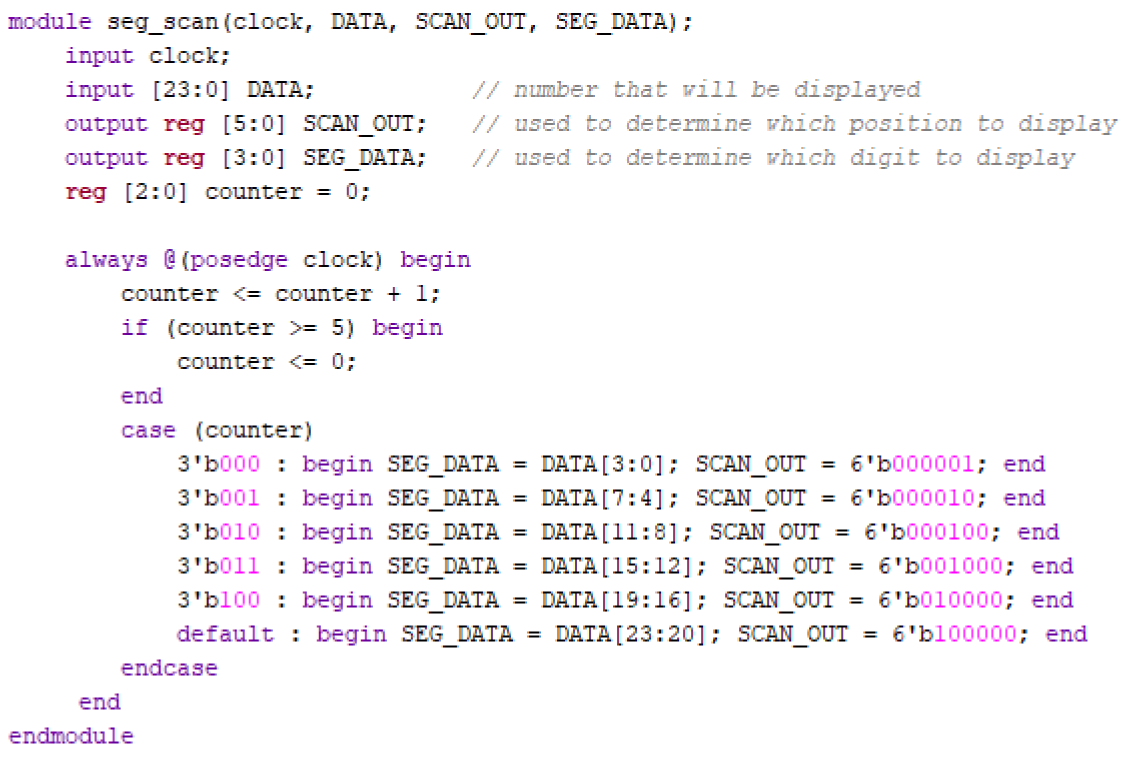
- counter.v

```
module counter(reset, clock, btn, new_value, max_value, reset_value, count, clock_out);
    input reset, clock, btn;
    input [7:0] new_value, max_value, reset_value;
    output reg [7:0] count;
    output reg clock_out;           // used as clock signal for the next count (Ex.: clock_out (of sec) -> clock (of min))
    reg [7:0] max_value_minus_one;

    initial count = reset_value;

    always @(posedge clock or posedge reset or posedge btn) begin
        if (reset) begin                    // if the reset button is pressed, count = reset_value
            count[7:4] <= reset_value[7:4];
            count[3:0] <= reset_value[3:0];
            clock_out <= 0;
        end
        else if (btn) begin                 // if the btn is pressed, count = new_value
            count[7:4] <= new_value[7:4];
            count[3:0] <= new_value[3:0];
        end
        else if (clock) begin               // if clock is 1, then count as usual
            if (max_value[3:0] == 0) begin
                max_value_minus_one[7:4] = max_value[7:4] - 4'b0001;
                max_value_minus_one[3:0] = 4'b1001;
            end
            else begin
                max_value_minus_one = max_value - 8'b00000001;
            end
            if (count[7:4] >= max_value_minus_one[7:4]) begin
                if (count[3:0] >= max_value_minus_one[3:0]) begin
                    count <= 0;
                    clock_out <= 1;
                end
                else begin
                    count <= count + 1;
                    clock_out <= 0;
                end
            end
            else begin
                if (count[3:0] >= 9) begin
                    count[7:4] <= count[7:4] + 1;
                    count[3:0] <= 0;
                    clock_out <= 0;
                end
                else begin
                    count <= count + 1;
                    clock_out <= 0;
                end
            end
        end
    end
endmodule
```

- display.v

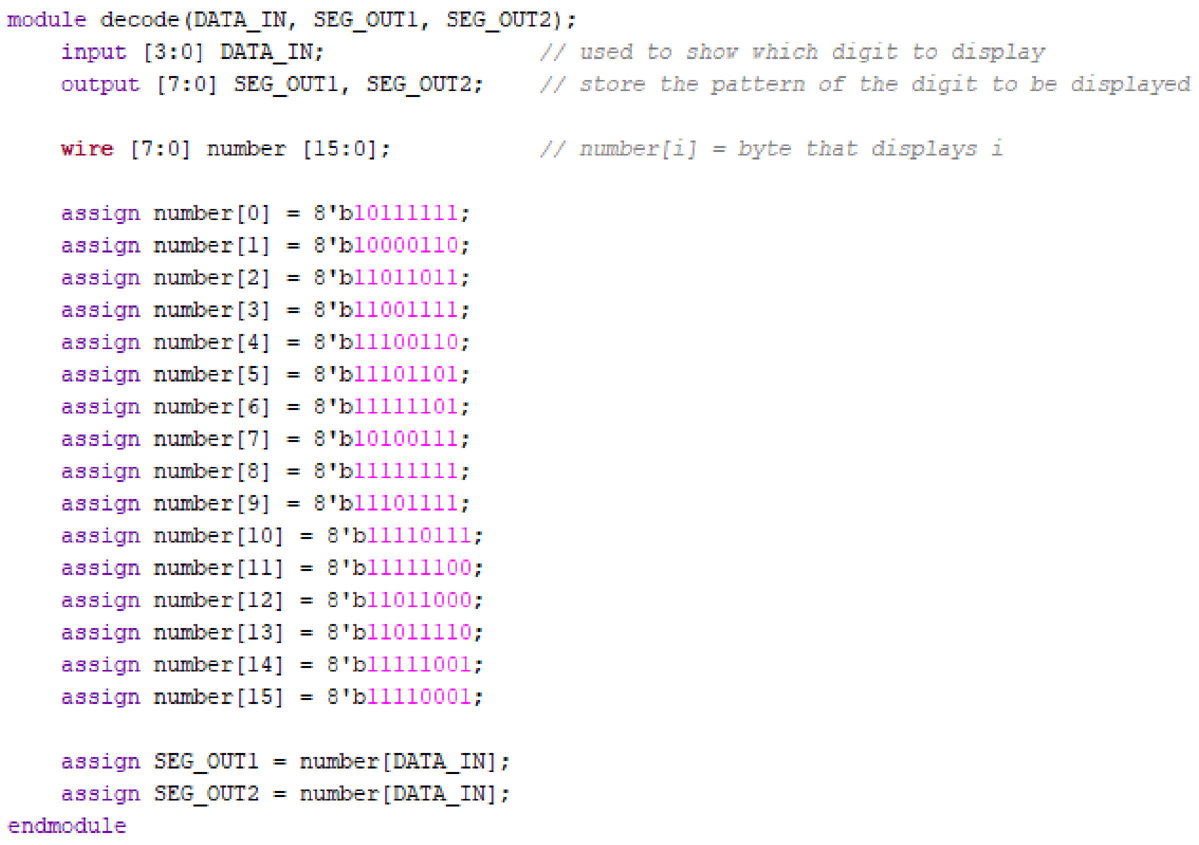
```
module display(clock, DATA, SCAN_OUT, SEG_OUT1, SEG_OUT2);
    input clock;
    input [23:0] DATA;                  // number that will be displayed
    output [5:0] SCAN_OUT;              // used to determine which position to display
    output [7:0] SEG_OUT1, SEG_OUT2;    // used to determine which digit to display
    wire[3:0] temp;

    seg_scan SS(clock, DATA, SCAN_OUT, temp);
    decode DEC(temp, SEG_OUT1, SEG_OUT2);
endmodule
```

- seg_scan.v



```verilog
module seg_scan(clock, DATA, SCAN_OUT, SEG_DATA);
    input clock;
    input [23:0] DATA;           // number that will be displayed
    output reg [5:0] SCAN_OUT;   // used to determine which position to display
    output reg [3:0] SEG_DATA;   // used to determine which digit to display
    reg [2:0] counter = 0;

    always @(posedge clock) begin
        counter <= counter + 1;
        if (counter >= 5) begin
            counter <= 0;
        end
        case (counter)
            3'b000 : begin SEG_DATA = DATA[3:0]; SCAN_OUT = 6'b000001; end
            3'b001 : begin SEG_DATA = DATA[7:4]; SCAN_OUT = 6'b000010; end
            3'b010 : begin SEG_DATA = DATA[11:8]; SCAN_OUT = 6'b000100; end
            3'b011 : begin SEG_DATA = DATA[15:12]; SCAN_OUT = 6'b001000; end
            3'b100 : begin SEG_DATA = DATA[19:16]; SCAN_OUT = 6'b010000; end
            default : begin SEG_DATA = DATA[23:20]; SCAN_OUT = 6'b100000; end
        endcase
     end
endmodule
```

- decode.v



```
module decode(DATA_IN, SEG_OUT1, SEG_OUT2);
    input [3:0] DATA_IN;                   // used to show which digit to display
    output [7:0] SEG_OUT1, SEG_OUT2;       // store the pattern of the digit to be displayed

    wire [7:0] number [15:0];              // number[i] = byte that displays i

    assign number[0] = 8'b10111111;
    assign number[1] = 8'b10000110;
    assign number[2] = 8'b11011011;
    assign number[3] = 8'b11001111;
    assign number[4] = 8'b11100110;
    assign number[5] = 8'b11101101;
    assign number[6] = 8'b11111101;
    assign number[7] = 8'b10100111;
    assign number[8] = 8'b11111111;
    assign number[9] = 8'b11101111;
    assign number[10] = 8'b11110111;
    assign number[11] = 8'b11111100;
    assign number[12] = 8'b11011000;
    assign number[13] = 8'b11011110;
    assign number[14] = 8'b11111001;
    assign number[15] = 8'b11110001;

    assign SEG_OUT1 = number[DATA_IN];
    assign SEG_OUT2 = number[DATA_IN];
endmodule
```

b) Test Bench:

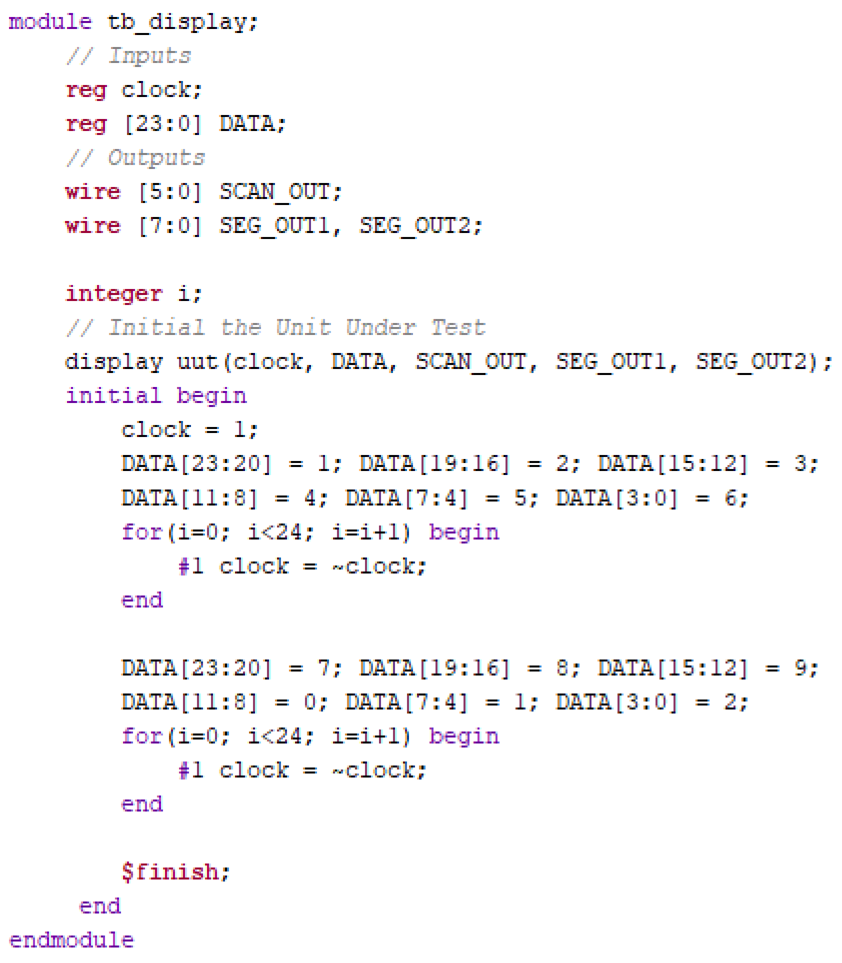
- tb_top.v

```
module tb_top;
    // Inputs
    reg reset, clock, switch, btn1, btn2, btn3;
    reg [6:0] new_value;
    // Output
    wire [5:0] SCAN_OUT;
    wire [7:0] SEG_OUT1, SEG_OUT2;

    integer i;

    // Initial the Unit Under Test
    top uut(reset, clock, switch, btn1, btn2, btn3, new_value, SCAN_OUT, SEG_OUT1, SEG_OUT2);
    initial begin
        new_value = 0;
        btn1 = 0; btn2 = 0; btn3 = 0;
        switch = 1;
        clock = 0;
        reset = 0;

        for (i=0; i<200; i=i+1) begin
            #1 clock = ~clock;
        end
        $finish;
    end
endmodule
```

- tb_hour_counter.v

```
module tb_hour_counter;
    // Inputs
    reg reset, clock, btn1, btn2, btn3;
    reg [7:0] new_value;
    // Output
    wire [23:0] DATA;

    integer i;
    // Initial the Unit Under Test
    hour_counter uut(reset, clock, btn1, btn2, btn3, new_value, DATA);

    initial begin
        clock = 0;
        reset = 0;
        btn1 = 0;
        btn2 = 0;
        btn3 = 0;
        new_value = 0;
        for (i=0; i<6; i=i+1) begin
            #1 clock = ~clock;
        end
        new_value[7:4] = 1;
        new_value[3:0] = 2;
        #1 btn1 = 1;
        #1 btn1 = 0;
        for (i=0; i<6; i=i+1) begin
            #1 clock = ~clock;
        end
        #1 btn2 = 1;
        #1 btn2 = 0;
        for (i=0; i<6; i=i+1) begin
            #1 clock = ~clock;
        end
        #1 btn3 = 1;
        #1 btn3 = 0;
        for (i=0; i<6; i=i+1) begin
            #1 clock = ~clock;
        end
        $finish;
    end
endmodule
```

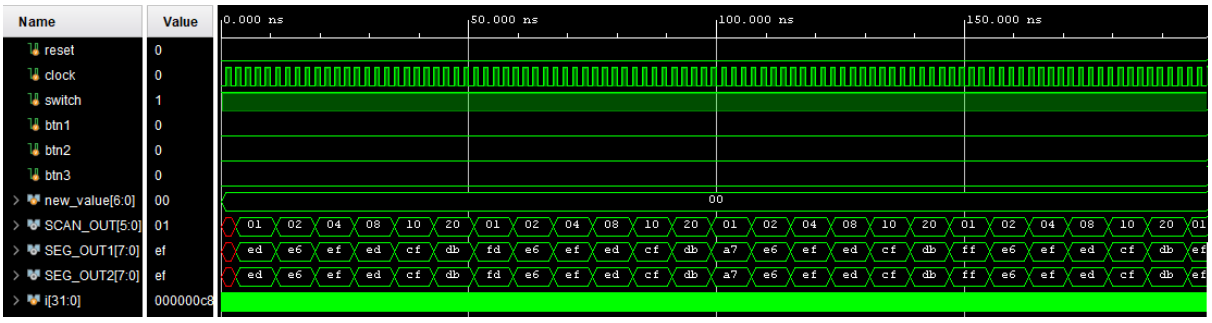
- tb_display.v



```
module tb_display;
    // Inputs
    reg clock;
    reg [23:0] DATA;
    // Outputs
    wire [5:0] SCAN_OUT;
    wire [7:0] SEG_OUT1, SEG_OUT2;

    integer i;
    // Initial the Unit Under Test
    display uut(clock, DATA, SCAN_OUT, SEG_OUT1, SEG_OUT2);
    initial begin
        clock = 1;
        DATA[23:20] = 1; DATA[19:16] = 2; DATA[15:12] = 3;
        DATA[11:8] = 4; DATA[7:4] = 5; DATA[3:0] = 6;
        for(i=0; i<24; i=i+1) begin
            #1 clock = ~clock;
        end

        DATA[23:20] = 7; DATA[19:16] = 8; DATA[15:12] = 9;
        DATA[11:8] = 0; DATA[7:4] = 1; DATA[3:0] = 2;
        for(i=0; i<24; i=i+1) begin
            #1 clock = ~clock;
        end

        $finish;
     end
endmodule
```

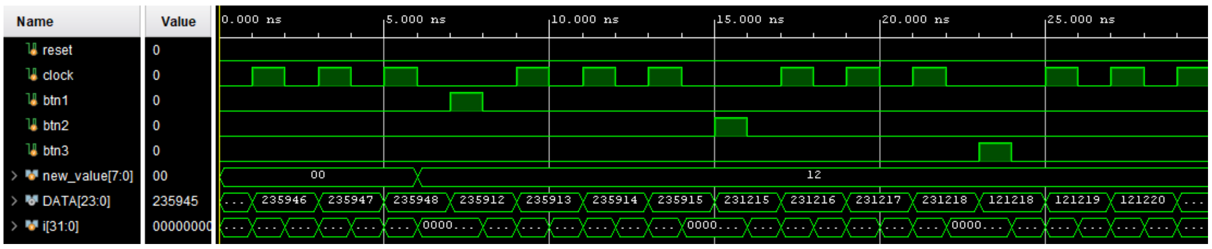
4. Simulation Waveforms:

- tb_top.v

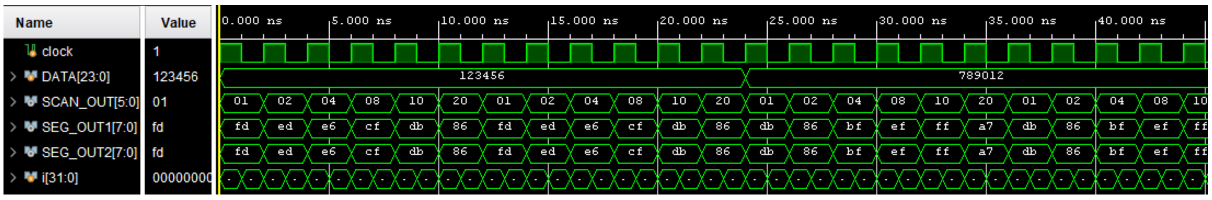


Obs.: To be able to see the results the frequency of the clocks was reduced.

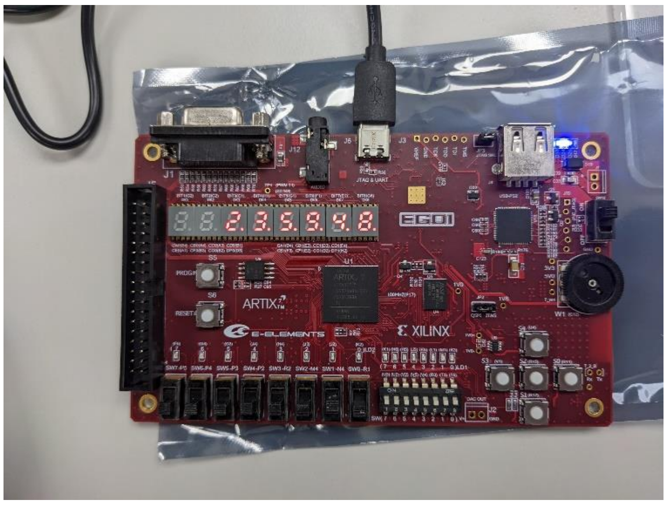
- tb_hour_counter.v

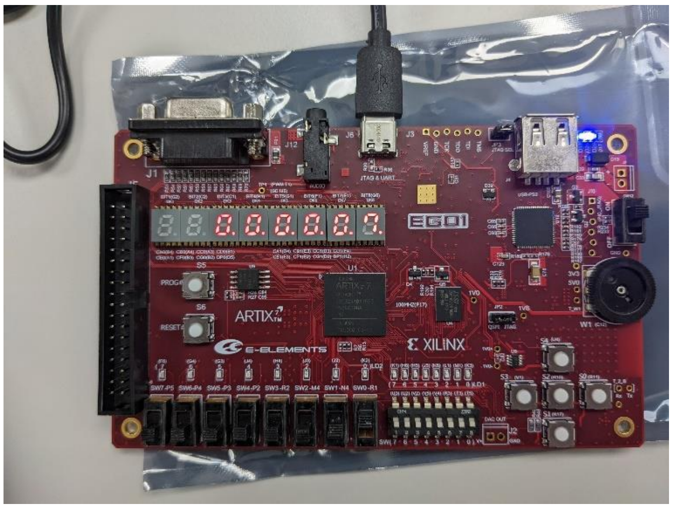


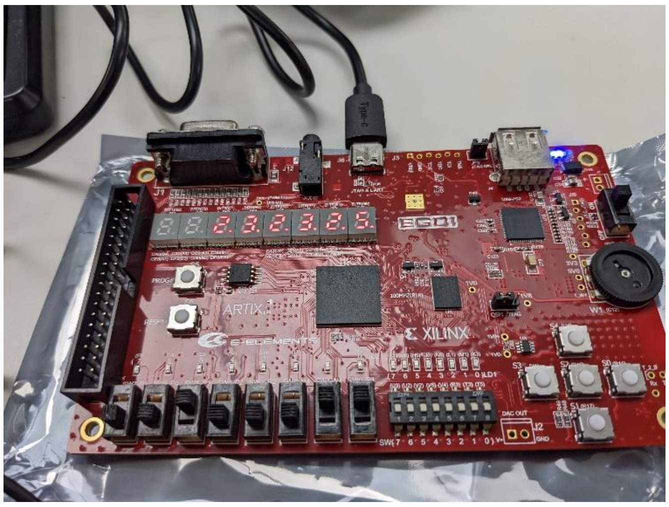
- tb_display.v

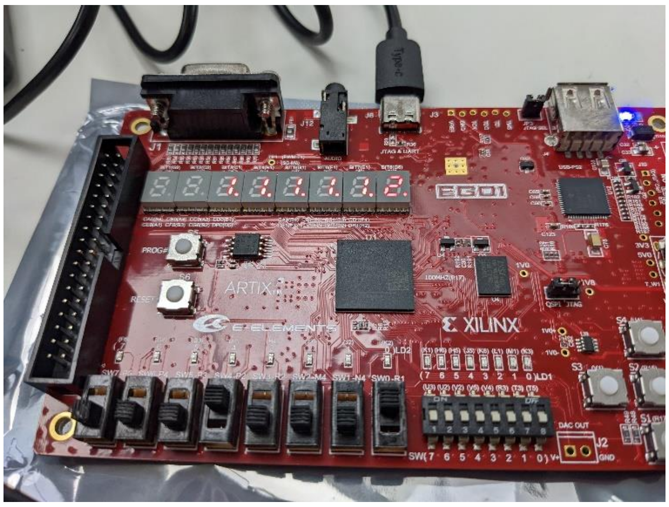


5. FPGA Results:


23:59:48


00:00:07


22:22:25
time shown 3 seconds after changing the hour, minutes and seconds to 22


11:11:12
time shown 1 second after changing the hour, minutes and seconds to 11

6. Reflection:

This lab was the most complex of the semester, it can be seen from the circuit block diagram. However, doing this lab step by step helped to finish more easily. The first thing was to create a clock that had a frequency of 1 second. The second one was to create the hour counter. After that, it was basically modifying code used in the previous lab. And from the results we can see that the lab went as expected.