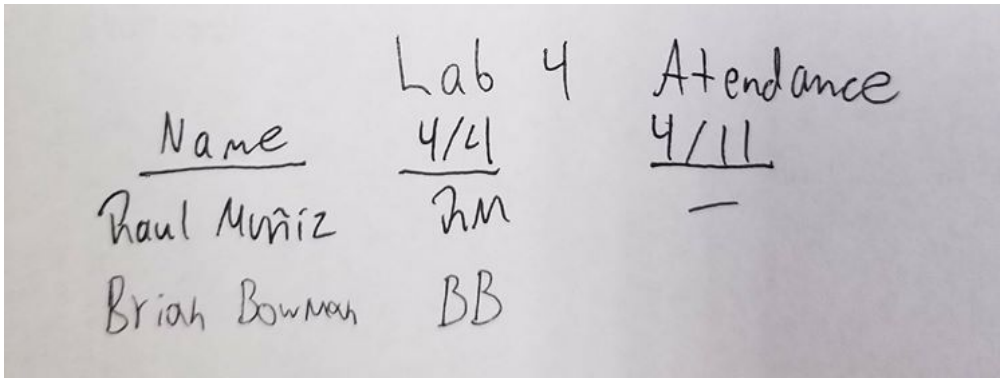
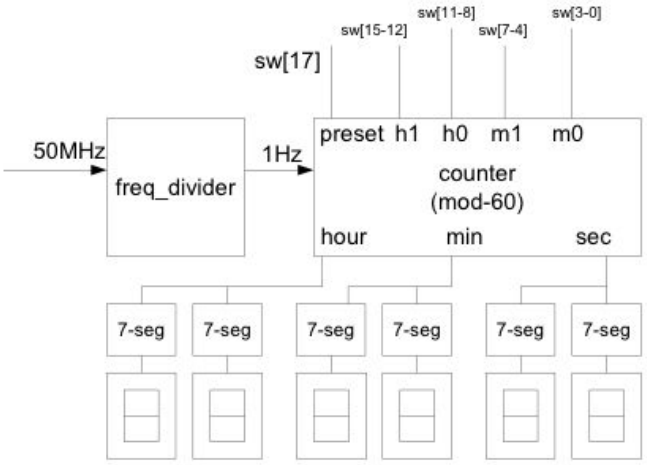


Part 1

I. **Problem:** Implement a digital clock on the DE2 board.

II. **Conceptual Design:**



III. **Verilog Design:**

```
//digital clock with preset by switches
module lab4part1 (CLOCK_50, SW, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5,
HEX6, HEX7);
	// When SW[17] = 1 we are in time setting mode. When it's 0,
clock running mode.
	// When SW[17] = 1:
	//   * SW[15:12] for hour1, SW[11:8] for hour0. (2-digit hour)
	//   * SW[7:4] for minute1, SW[3:0] for minute0. (2-digit minute)
	//   * second1, second0 are 00. (2-digit seconds)
	input CLOCK_50; // 50MHz clock
	input [17:0] SW;

	// HEX7 HEX6 are turned off
	// HEX5 HEX4 for hour
	// HEX3 HEX2 for minute
```

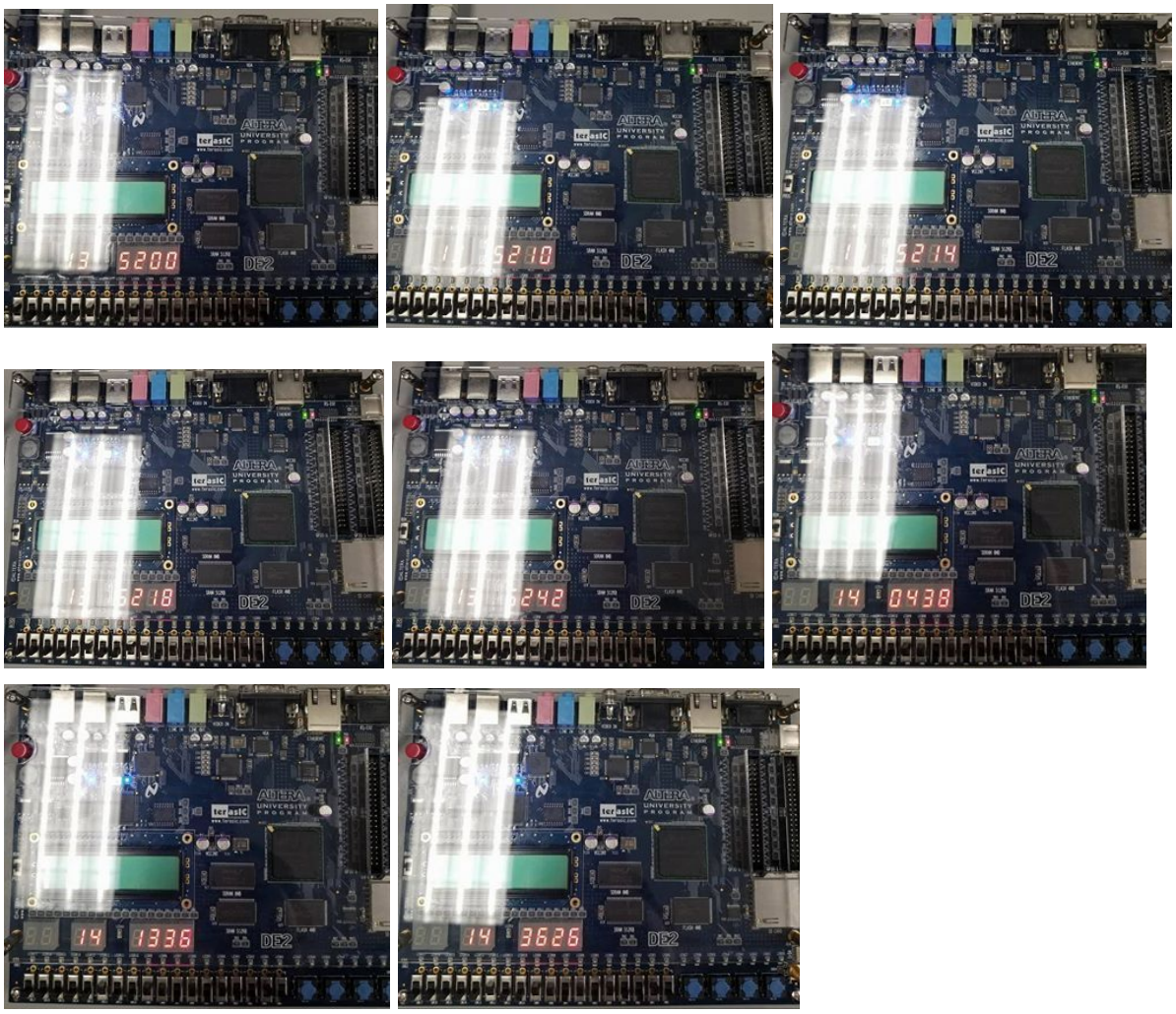
```
	// HEX1 HEX0 for seconds
	output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;

	wire clock_1Hz;

	reg [3:0] h1, h0, m1, m0, s1, s0;

	clock_div M0 (CLOCK_50, 25000000, clock_1Hz); // generate 1Hz
clock

	hex2seg7 hour1 (h1, HEX5);
	hex2seg7 hour0 (h0, HEX4);
	hex2seg7 minute1 (m1, HEX3);
	hex2seg7 minute0 (m0, HEX2);
	hex2seg7 second1 (s1, HEX1);
	hex2seg7 second0 (s0, HEX0);
	assign HEX6 = 7'b1111111;
	assign HEX7 = 7'b1111111;

	always @ (posedge clock_1Hz) begin
	if (SW[17]) begin
		h1 = SW[15:12];
		h0 = SW[11:8];
		m1 = SW[7:4];
		m0 = SW[3:0];
		s1 = 0;
		s0 = 0;
	end else begin
		s0 = s0 + 1;
		if (s0 == 10) begin
			s1 = s1 + 1;
			s0 = 0;
		end
		if (s1 == 6) begin
			m0 = m0 + 1;
			s1 = 0;
		end
		if (m0 == 10) begin
			m1 = m1 + 1;
			m0 = 0;
		end
		if (m1 == 6) begin
			h0 = h0 + 1;
```

```
                m1 = 0;
            end
            if (h0 == 10) begin
                h1 = h1 + 1;
                h0 = 0;
            end
            if (h1 == 2 && h0 == 4) begin
                h1 = 0;
                s0 = 0;
            end
      end
      end
endmodule

// output frequency = clk_in frequency / (2 * clkscale)
module clock_div (clk_in, clkscale, clk_out);
    input clk_in;
    input [31:0] clkscale;
    output reg clk_out;

    reg [31:0] clkq = 0;

    always @ (posedge clk_in) begin
       clkq = clkq + 1;
       if (clkq == clkscale) begin
            clk_out = ~clk_out;
            clkq = 0;
       end
    end
endmodule

module hex2seg7 (bin, seg7);
    input [3:0] bin;
    output reg [6:0] seg7;

    always @ (*) begin
       case (bin)
            4'b0000 : seg7 = 7'b1000000; // 0
            4'b0001 : seg7 = 7'b1111001; // 1
            4'b0010 : seg7 = 7'b0100100; // 2
            4'b0011 : seg7 = 7'b0110000; // 3
            4'b0100 : seg7 = 7'b0011001; // 4
            4'b0101 : seg7 = 7'b0010010; // 5
```

```
            4'b0110 : seg7 = 7'b0000010; // 6
            4'b0111 : seg7 = 7'b1111000; // 7
            4'b1000 : seg7 = 7'b0000000; // 8
            4'b1001 : seg7 = 7'b0010000; // 9
            4'b1010 : seg7 = 7'b0001000; // A
            4'b1011 : seg7 = 7'b0000011; // B
            4'b1100 : seg7 = 7'b1000110; // C
            4'b1101 : seg7 = 7'b0100001; // D
            4'b1110 : seg7 = 7'b0000110; // E
            4'b1111 : seg7 = 7'b0001110; // F
            default : seg7 = 7'b1111111; // blank
        endcase
    end
endmodule
```

IV. Lab Procedure: To quickly verify the proper operation of the clock, we increased the speed of the internal clock to 100x. This allowed us to verify that the seconds, minutes, and hours would wrap around properly.

Pictures

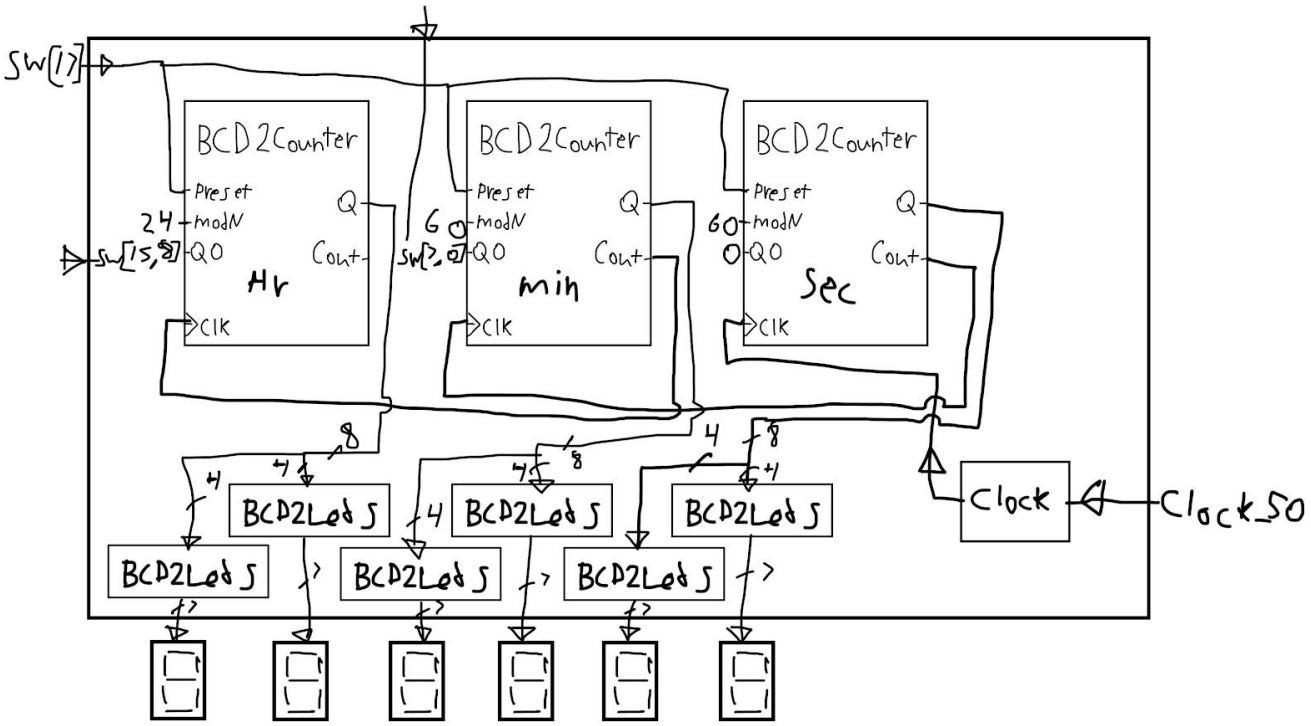


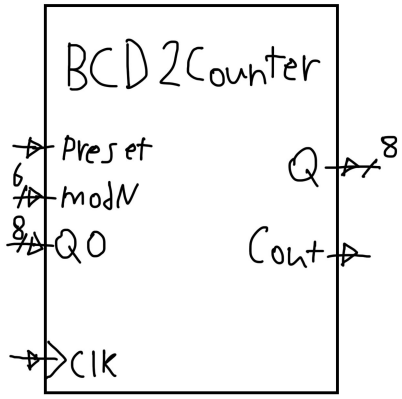
Troubleshooting: When we first programmed the board, all the numbers appeared garbled. We found out the problem was in the declaration of the 7-segment display outputs. We had "`[0:6]`" instead of "`[6:0]`". Once we fixed this, the board functioned correctly.

Conclusion: We learned how to implement counting of the type used in digital clocks.

Part 2

I. **Problem:** Implement a digital clock on the DE2 as in part 1, but using a hierarchical design.
II. **Conceptual Design:**





III. Verilog Design:

```
//digital clock with preset by switches
module lab4part2 (CLOCK_50, SW, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5,
HEX6, HEX7);
	// When SW[17] = 1 we are in time setting mode. When it's 0,
clock running mode.
	// When SW[17] = 1:
	//   * SW[15:12] for hour1, SW[11:8] for hour0. (2-digit hour)
	//   * SW[7:4] for minute1, SW[3:0] for minute0. (2-digit minute)
	//   * second1, second0 are 00. (2-digit seconds)
```

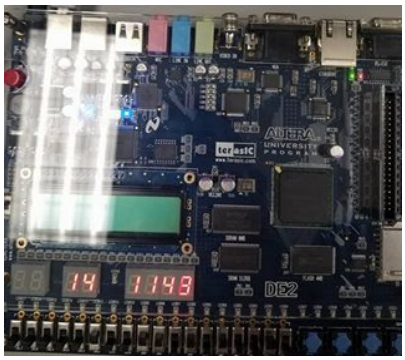
```
      input CLOCK_50; // 50MHz clock
      input [17:0] SW;

      // HEX7 HEX6 are turned off
      // HEX5 HEX4 for hour
      // HEX3 HEX2 for minute
      // HEX1 HEX0 for seconds
      output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;

      wire clock_1Hz, clock_min, clock_hr, clock_day;

      wire [23:0] Q;

      clock_div M0 (CLOCK_50, 250000, clock_1Hz); // generate 1Hz clock

      BCD2Counter U0 (clock_1Hz, 60, SW[17], 0, Q[7:0], clock_min);
      BCD2Counter U1 (clock_min, 60, SW[17], SW[7:0], Q[15:8],
clock_hr);
      BCD2Counter U2 (clock_hr, 24, SW[17], SW[15:8], Q[23:16],
clock_day);

      hex2seg7 hour1 (Q[23:20], HEX5);
      hex2seg7 hour0 (Q[19:16], HEX4);
      hex2seg7 minute1 (Q[15:12], HEX3);
      hex2seg7 minute0 (Q[11:8], HEX2);
      hex2seg7 second1 (Q[7:4], HEX1);
      hex2seg7 second0 (Q[3:0], HEX0);
      assign HEX6 = 7'b1111111;
      assign HEX7 = 7'b1111111;
endmodule

//4 bit 2-digit BCD counter with modN, and preset control and Q0
starting point
module BCD2Counter (clk, modN, preset, Q0, Q, Cout);
    input clk;
      input [5:0] modN; // 6bit mod number :mod-60 counter for min,
sec, and mod-24 counter for hr
      input preset; // high active preset
      input [7:0] Q0; // to set initial value

      output [7:0] Q; // 2 bcds
      output reg Cout; // carry out
```

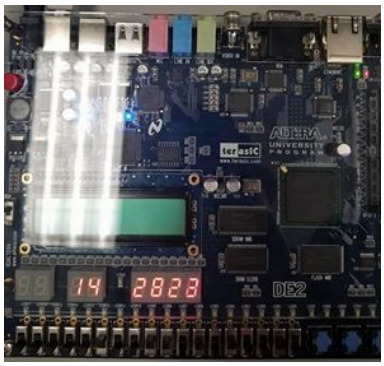
```
	reg [3:0] bcd1; // high digit
	reg [3:0] bcd0; // high digit

	assign Q = { bcd1,bcd0 }; // assemble to 2 bcd outputs

	always @(posedge clk, posedge preset) begin
	if (preset) begin // setting mode
		bcd1 <= Q0[7:4]; // presetting mode
		bcd0 <= Q0[3:0];
		Cout <= 0;
	end else if ((10 * bcd1 + bcd0) < (modN - 1)) begin // clock mode
		if (bcd0 < 9) begin
			bcd0 <= bcd0 + 1; // add 1 to bcd0 only
		end else begin // bcd0 reset to 0, try adding 1 to bcd1
			bcd0 <= 0;
			bcd1 <= bcd1 + 1;
		end

		Cout <= 0;
	end else begin
		bcd0 <= 0;
		bcd1 <= 0;
		Cout <= 1;
	end
	end
endmodule
```

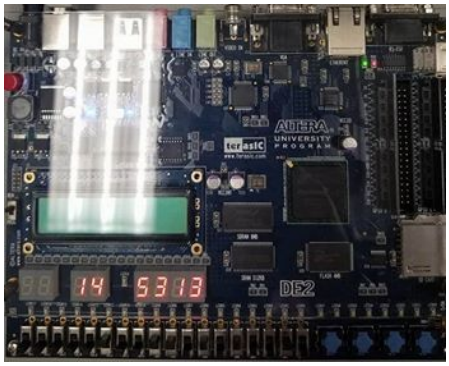
IV. Lab Procedure: We tested the operation of our code on the board in the same manner as part 1.

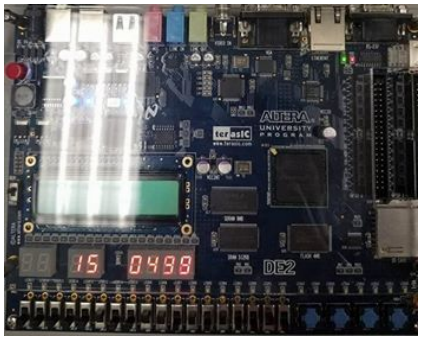
Pictures

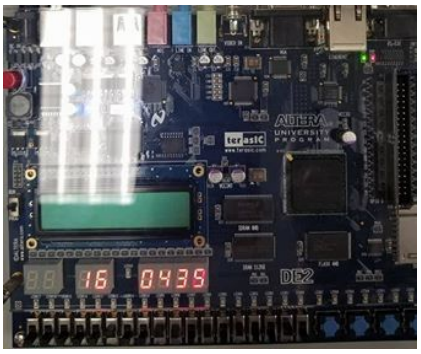
Lab4
Brian Bowman
Raul Muniz

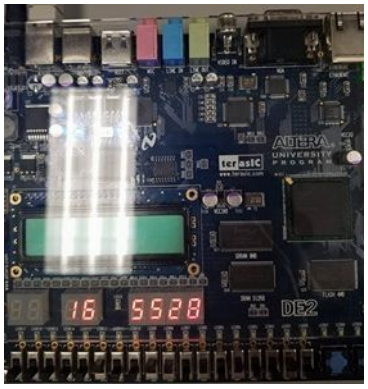


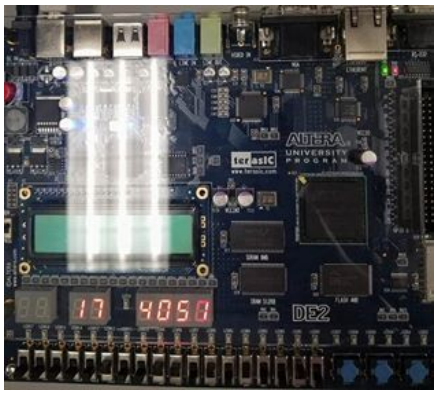












Troubleshooting: n/a

Conclusion: The hierarchical design also managed to prove an efficient method to display a digital clock in the lab. Both successfully performed the task as wanted. Although the verilog code for part 2 was more complex to design.