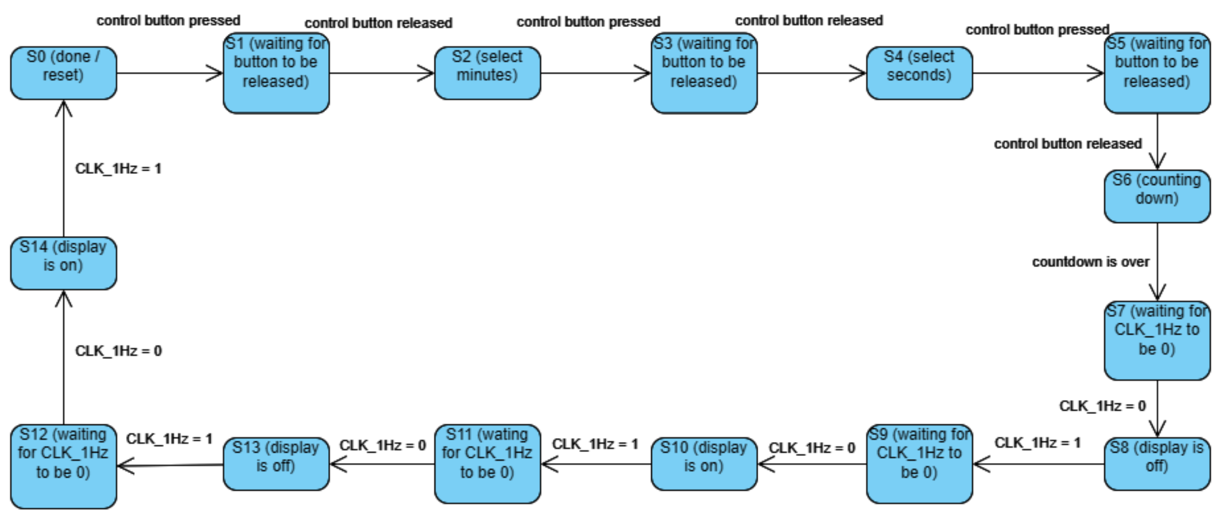
Lab 7 report

Name: Oscar Stark   Student ID: F11015127

In this lab I implemented a countdown timer. The button on pin R17 is the reset, the one on pin R11 is used for control, and the one on pin U4 is used to add minutes or seconds during the "select minutes" and "select seconds" states. When the countdown is over, the seven-segment display flashes twice.

State Transition Graph

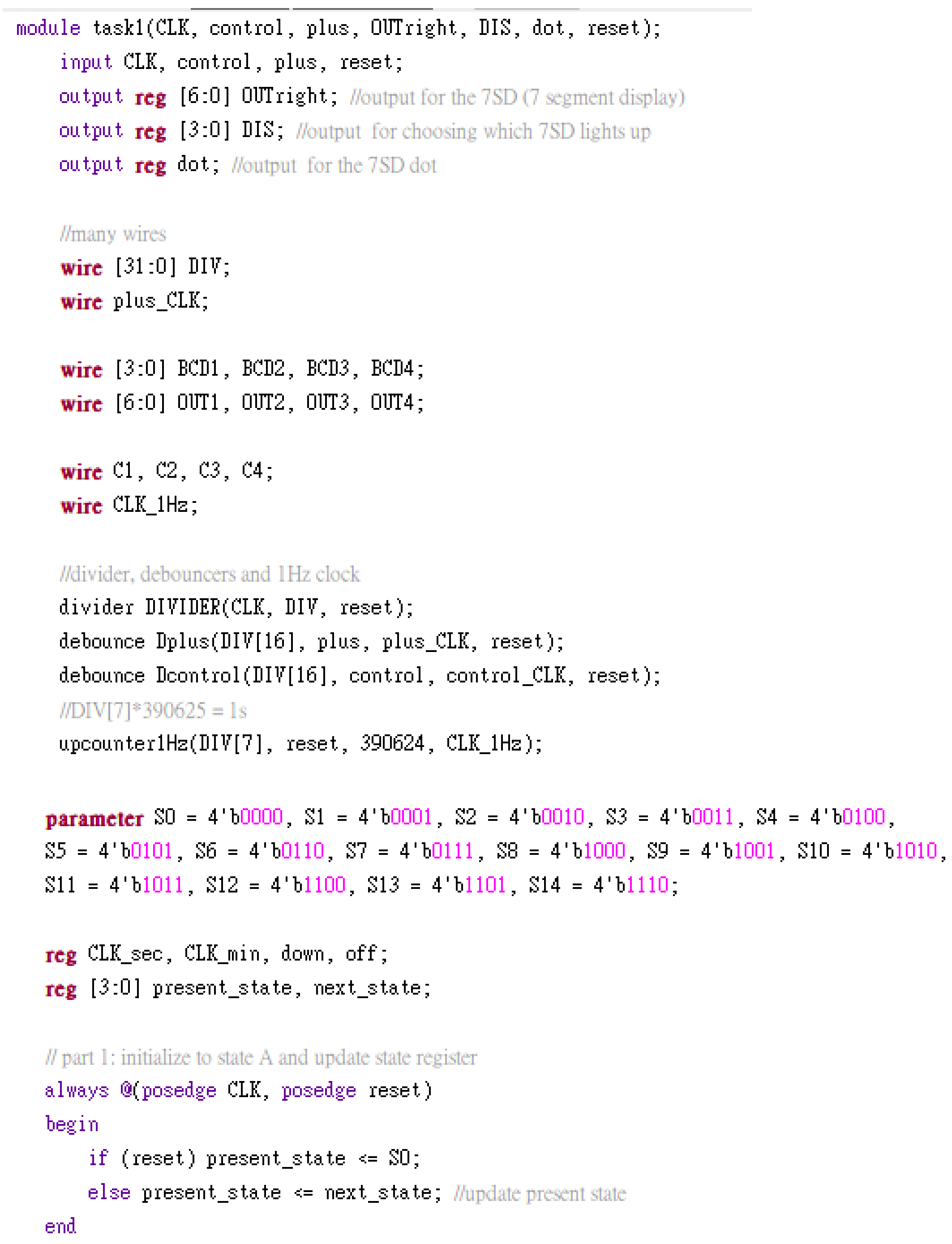


State Transition Table

Present State (PS)	Next State (NS)	
S0	if (control_CLK == 1) next_state = S1	if (control_CLK == 0) next_state = S0
S1	if (control_CLK == 0) next_state = S2	if (control_CLK == 1) next_state = S1
S2	if (control_CLK == 1) next_state = S3	if (control_CLK == 0) next_state = S2
S3	if (control_CLK == 0) next_state = S4	if (control_CLK == 1) next_state = S3
S4	if (control_CLK == 1) next_state = S5	if (control_CLK == 0) next_state = S4
S5	if (control_CLK == 0) next_state = S6	if (control_CLK == 1) next_state = S5
S6	if (the countdown is over) next_state = S7	if (the countdown is not over) next_state = S6
S7	if (CLK_1Hz == 0) next_state = S8	if (CLK_1Hz == 1) next_state = S7
S8	if (CLK_1Hz == 1) next_state = S9	if (CLK_1Hz == 0) next_state = S8
S9	if (CLK_1Hz == 0) next_state = S10	if (CLK_1Hz == 1) next_state = S9
S10	if (CLK_1Hz == 1) next_state = S11	if (CLK_1Hz == 0) next_state = S10
S11	if (CLK_1Hz == 0) next_state = S12	if (CLK_1Hz == 1) next_state = S11
S12	if (CLK_1Hz == 1) next_state = S13	if (CLK_1Hz == 0) next_state = S12
S13	if (CLK_1Hz == 0) next_state = S14	if (CLK_1Hz == 1) next_state = S13
S14	if (CLK_1Hz == 1) next_state = S0	if (CLK_1Hz == 0) next_state = S14

Verilog Code

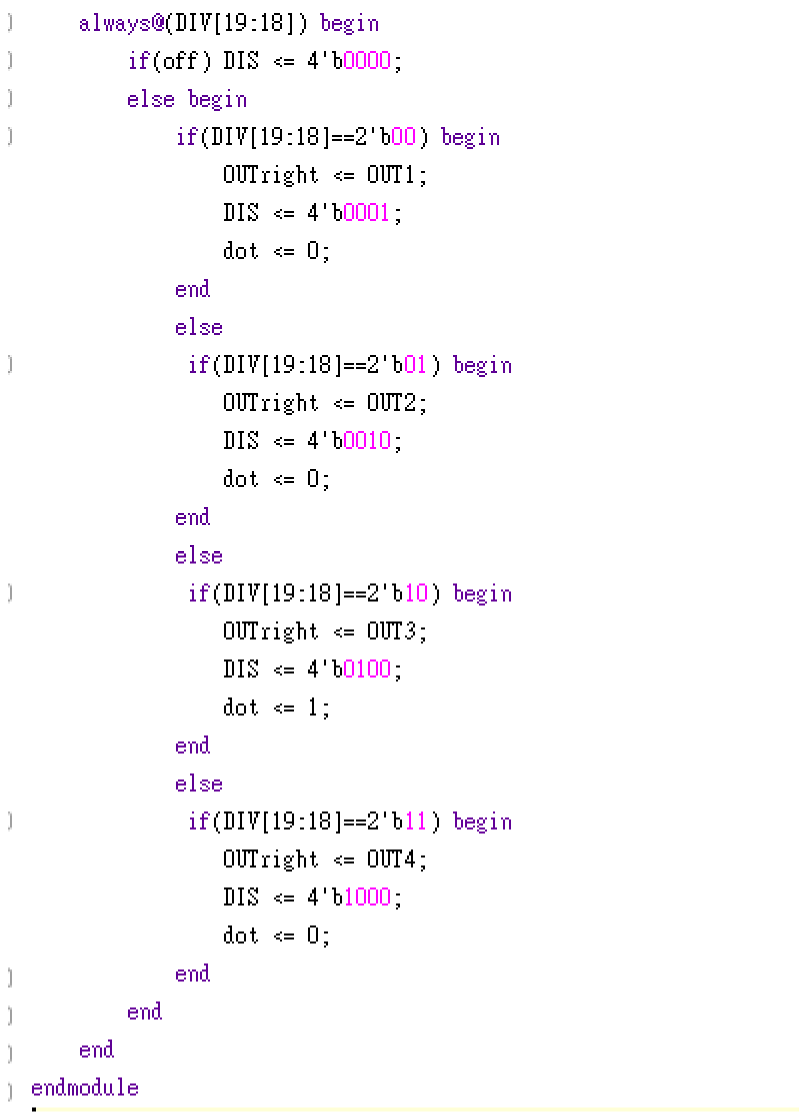
Main code



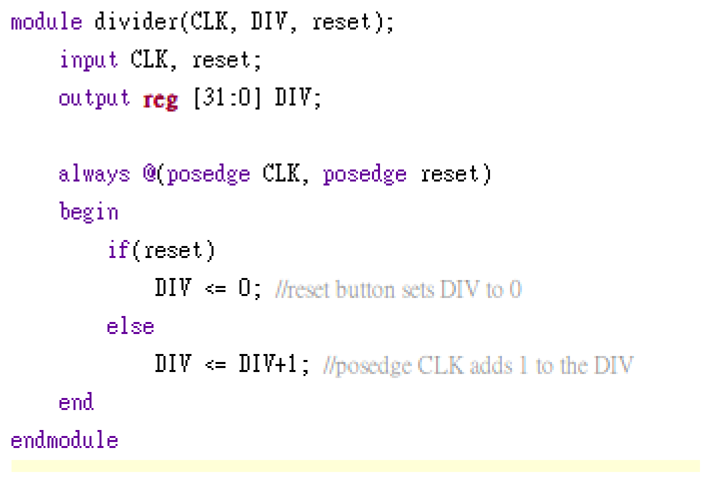
```
module task1(CLK, control, plus, OUTright, DIS, dot, reset);
    input CLK, control, plus, reset;
    output reg [6:0] OUTright; //output for the 7SD (7 segment display)
    output reg [3:0] DIS; //output  for choosing which 7SD lights up
    output reg dot; //output  for the 7SD dot

    //many wires
    wire [31:0] DIV;
    wire plus_CLK;

    wire [3:0] BCD1, BCD2, BCD3, BCD4;
    wire [6:0] OUT1, OUT2, OUT3, OUT4;

    wire C1, C2, C3, C4;
    wire CLK_1Hz;

    //divider, debouncers and 1Hz clock
    divider DIVIDER(CLK, DIV, reset);
    debounce Dplus(DIV[16], plus, plus_CLK, reset);
    debounce Dcontrol(DIV[16], control, control_CLK, reset);
    //DIV[7]*390625 = 1s
    upcounter1Hz(DIV[7], reset, 390624, CLK_1Hz);

   parameter S0 = 4'b0000, S1 = 4'b0001, S2 = 4'b0010, S3 = 4'b0011, S4 = 4'b0100,
   S5 = 4'b0101, S6 = 4'b0110, S7 = 4'b0111, S8 = 4'b1000, S9 = 4'b1001, S10 = 4'b1010,
   S11 = 4'b1011, S12 = 4'b1100, S13 = 4'b1101, S14 = 4'b1110;

   reg CLK_sec, CLK_min, down, off;
   reg [3:0] present_state, next_state;

   // part 1: initialize to state A and update state register
   always @(posedge CLK, posedge reset)
   begin
       if (reset) present_state <= S0;
       else present_state <= next_state; //update present state
   end
```

```
// part 2: determine next state
always @(present_state, control_CLK, CLK_1Hz)
begin
    case (present_state)
        S0: begin if(control_CLK) next_state=S1; else next_state=S0; end //done / reset
        S1: begin if(~control_CLK) next_state=S2; else next_state=S1; end //waiting for button to be released
        S2: begin if(control_CLK) next_state=S3; else next_state=S2; end //select minutes
        S3: begin if(~control_CLK) next_state=S4; else next_state=S3; end //waiting for button to be released
        S4: begin if(control_CLK) next_state=S5; else next_state=S4; end //select seconds
        S5: begin if(~control_CLK) next_state=S6; else next_state=S5; end //waiting for button to be released
        S6: begin if(BCD1==0 && BCD2==0 && BCD3==0 && BCD4==0) next_state=S7; else next_state=S6; end //counting down
        S7: begin if(~CLK_1Hz) next_state=S8; else next_state=S7; end //waiting for CLK_1Hz to be 0
        S8: begin if(CLK_1Hz) next_state=S9; else next_state=S8; end //off
        S9: begin if(~CLK_1Hz) next_state=S10; else next_state=S9; end //waiting for CLK_1Hz to be 0
        S10: begin if(CLK_1Hz) next_state=S11; else next_state=S10; end //on
        S11: begin if(~CLK_1Hz) next_state=S12; else next_state=S11; end //waiting for CLK_1Hz to be 0
        S12: begin if(CLK_1Hz) next_state=S13; else next_state=S12; end //off
        S13: begin if(~CLK_1Hz) next_state=S14; else next_state=S13; end //waiting for CLK_1Hz to be 0
        S14: begin if(CLK_1Hz) next_state=S0; else next_state=S14; end //on
    endcase
end

// part 3: evaluate output function z
always @(present_state, plus_CLK, CLK_1Hz)
begin
    case (present_state)
        S0: begin CLK_sec = 0;  CLK_min = 0; off = 0; end
        S1: off = 0;
        S2:
        begin
            off = 0;
            down = 0;
            if (plus_CLK) //set minutes
            begin
                CLK_min = 1;
            end
            else
            begin
                CLK_min = 0;
            end
        end
        S3:off = 0;
        S4:
        begin
            off = 0;
            down = 0;
            if (plus_CLK) //set seconds
            begin
                CLK_sec = 1;
            end
            else
            begin
                CLK_sec = 0;
            end
        end
        S5:off = 0;
        S6:
        begin
            off = 0;
            CLK_sec = CLK_1Hz;
            CLK_min = C2;
            down = 1;
        end
        S7: begin off = 0; down = 0; CLK_sec = 0;  CLK_min = 0; end
        S8:off = 1;
        S9:off = 1;
        S10:off = 0;
        S11:off = 0;
        S12:off = 1;
        S13:off = 1;
        S14:off = 0;
    endcase
end
```

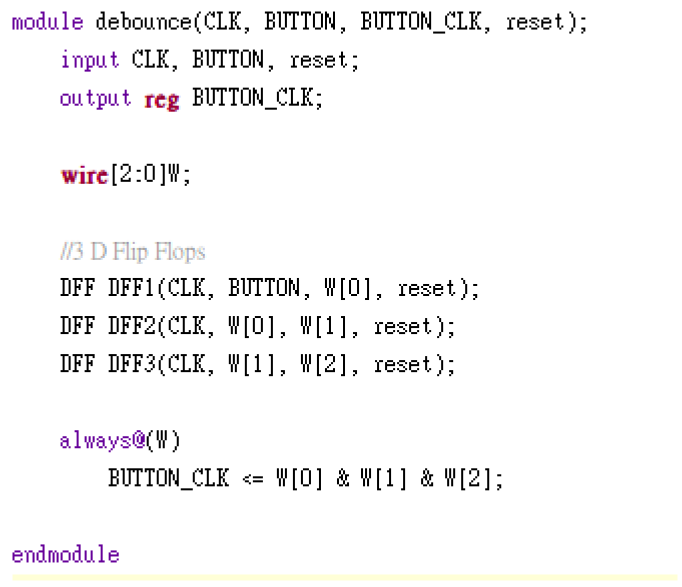
```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

  upcounter UPC1(CLK_sec, BCD1, reset, 0, 9, C1, 0, down);
  upcounter UPC2(C1, BCD2, reset, 0, 5, C2, 0, down);
  upcounter UPC3(CLK_min, BCD3, reset, 0, 9, C3, 0, down);
  upcounter UPC4(C3, BCD4, reset, 0, 9, C4, 0, down);

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

  seven SEV1(BCD1, OUT1);
  seven SEV2(BCD2, OUT2);
  seven SEV3(BCD3, OUT3);
  seven SEV4(BCD4, OUT4);
```


```
    always@(DIV[19:18]) begin
        if(off) DIS <= 4'b0000;
        else begin
            if(DIV[19:18]==2'b00) begin
                OUTright <= OUT1;
                DIS <= 4'b0001;
                dot <= 0;
            end
            else
             if(DIV[19:18]==2'b01) begin
                OUTright <= OUT2;
                DIS <= 4'b0010;
                dot <= 0;
            end
            else
             if(DIV[19:18]==2'b10) begin
                OUTright <= OUT3;
                DIS <= 4'b0100;
                dot <= 1;
            end
            else
             if(DIV[19:18]==2'b11) begin
                OUTright <= OUT4;
                DIS <= 4'b1000;
                dot <= 0;
            end
        end
    end
endmodule
```

Divider

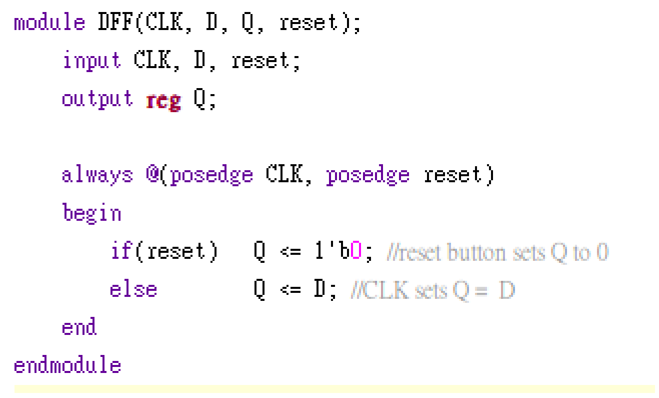

```
module divider(CLK, DIV, reset);
    input CLK, reset;
    output reg [31:0] DIV;

    always @(posedge CLK, posedge reset)
    begin
        if(reset)
            DIV <= 0; //reset button sets DIV to 0
        else
            DIV <= DIV+1; //posedge CLK adds 1 to the DIV
    end
endmodule
```

Debouncer



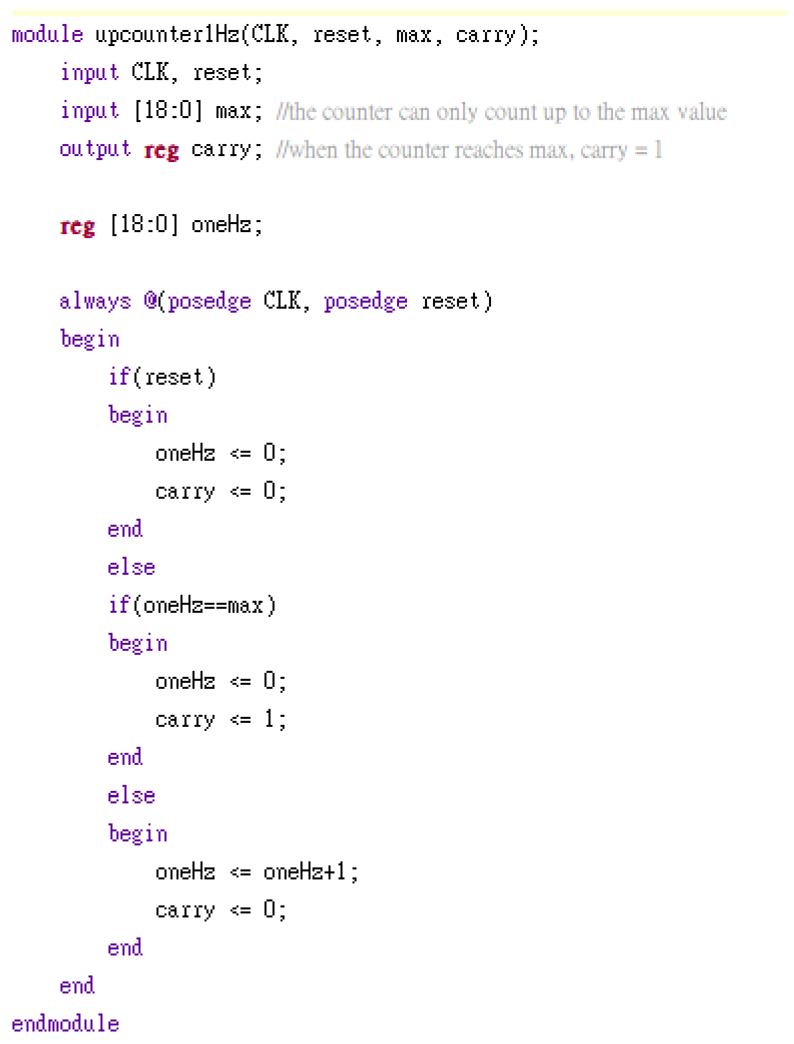
```
module debounce(CLK, BUTTON, BUTTON_CLK, reset);
    input CLK, BUTTON, reset;
    output reg BUTTON_CLK;

    wire[2:0]W;

    //3 D Flip Flops
    DFF DFF1(CLK, BUTTON, W[0], reset);
    DFF DFF2(CLK, W[0], W[1], reset);
    DFF DFF3(CLK, W[1], W[2], reset);

    always@(W)
        BUTTON_CLK <= W[0] & W[1] & W[2];

endmodule
```

D Flip Flop



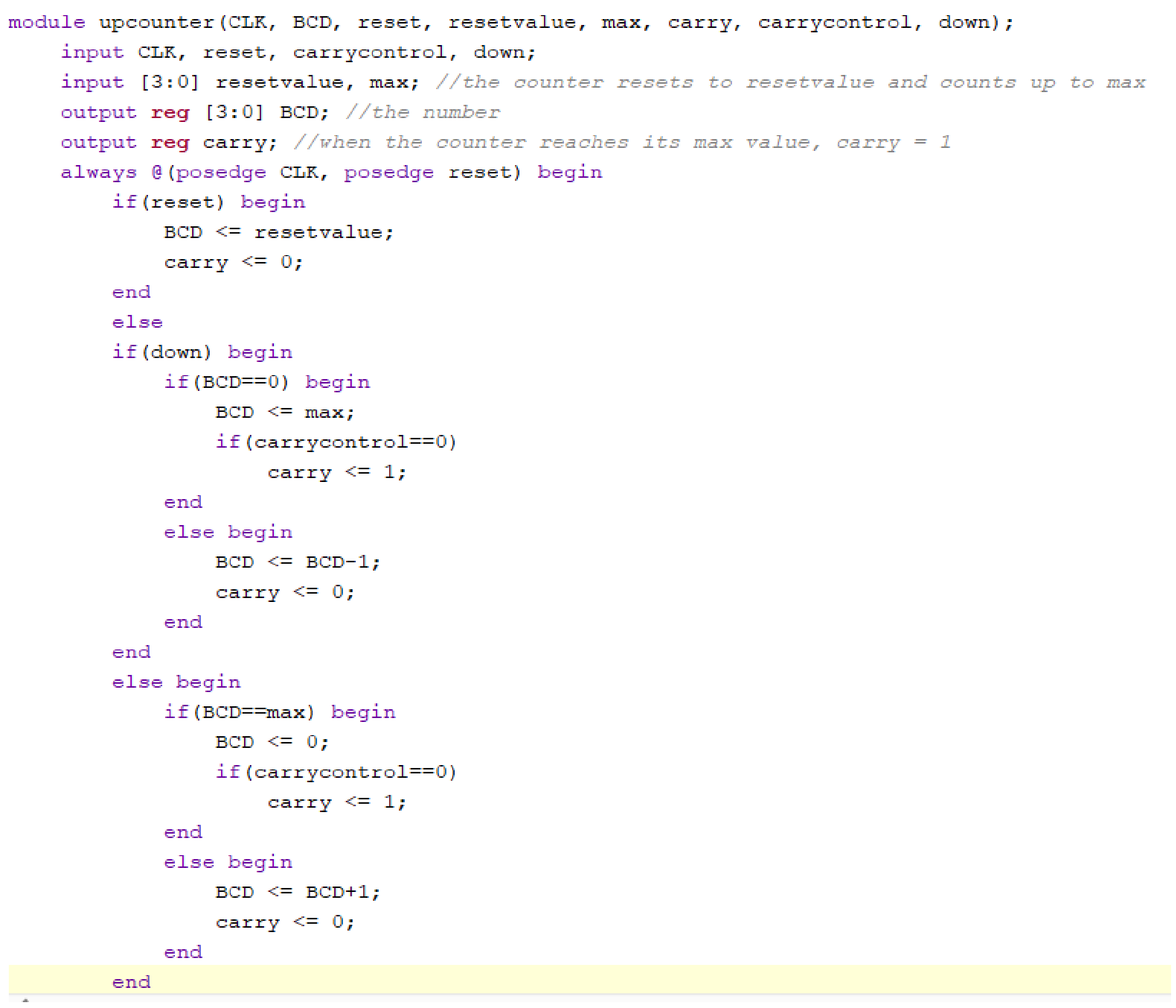
```
module DFF(CLK, D, Q, reset);
    input CLK, D, reset;
    output reg Q;

    always @(posedge CLK, posedge reset)
    begin
        if(reset)   Q <= 1'b0; //reset button sets Q to 0
        else        Q <= D; //CLK sets Q = D
    end
endmodule
```

1Hz Clock



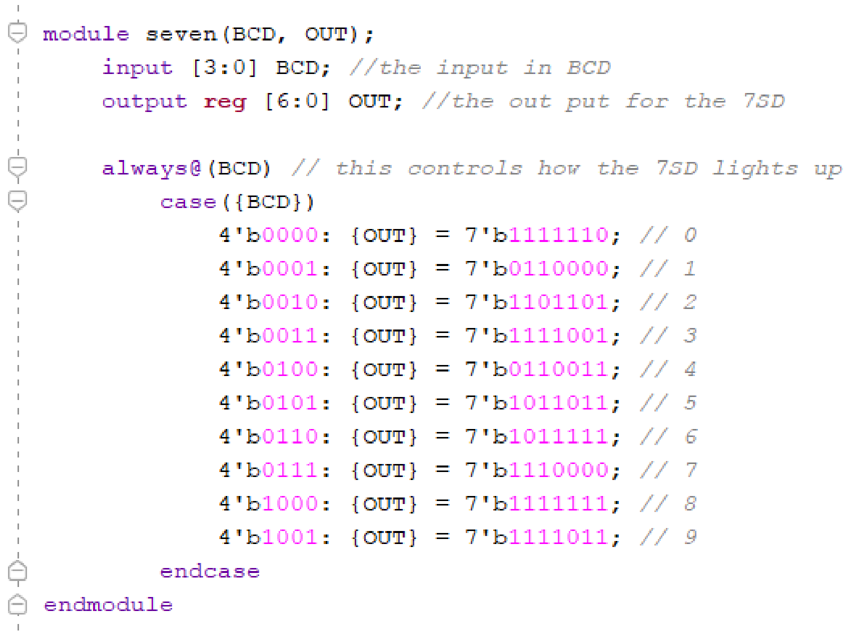
```
module upcounter1Hz(CLK, reset, max, carry);
    input CLK, reset;
    input [18:0] max; //the counter can only count up to the max value
    output reg carry; //when the counter reaches max, carry = 1

    reg [18:0] oneHz;

    always @(posedge CLK, posedge reset)
    begin
        if(reset)
        begin
            oneHz <= 0;
            carry <= 0;
        end
        else
        if(oneHz==max)
        begin
            oneHz <= 0;
            carry <= 1;
        end
        else
        begin
            oneHz <= oneHz+1;
            carry <= 0;
        end
    end
endmodule
```

Upcounter (can also count down)



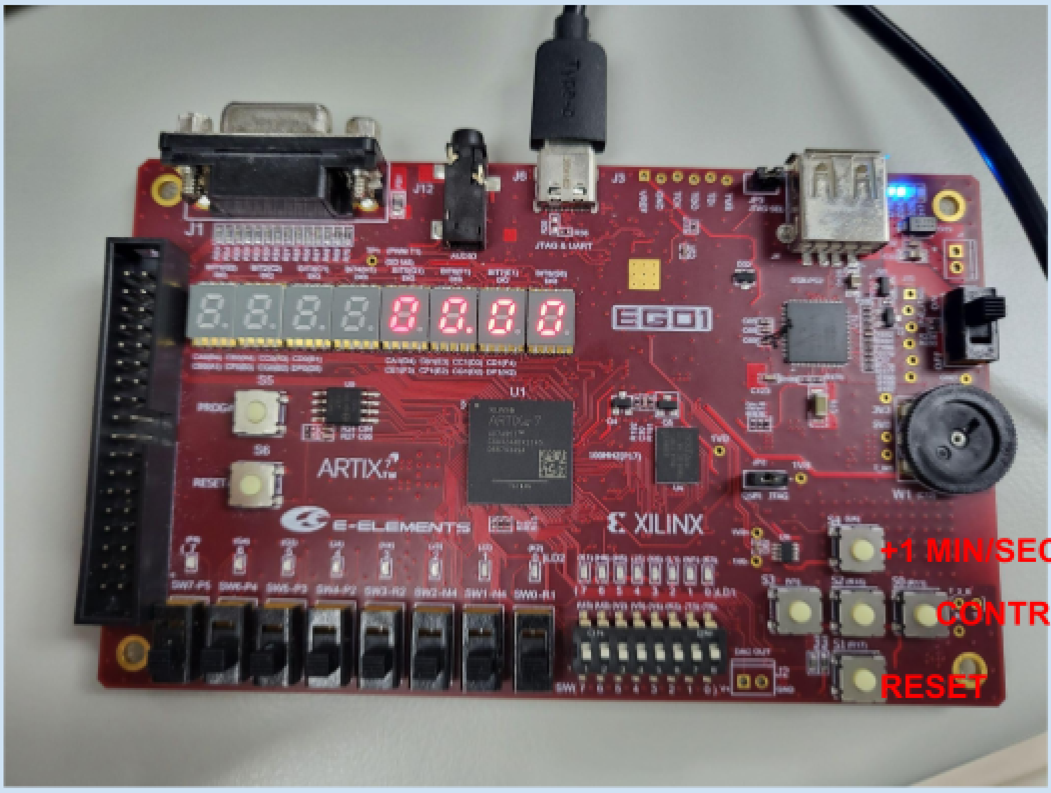
```
module upcounter(CLK, BCD, reset, resetvalue, max, carry, carrycontrol, down);
    input CLK, reset, carrycontrol, down;
    input [3:0] resetvalue, max; //the counter resets to resetvalue and counts up to max
    output reg [3:0] BCD; //the number
    output reg carry; //when the counter reaches its max value, carry = 1
    always @(posedge CLK, posedge reset) begin
        if(reset) begin
            BCD <= resetvalue;
            carry <= 0;
        end
        else
        if(down) begin
            if(BCD==0) begin
                BCD <= max;
                if(carrycontrol==0)
                    carry <= 1;
            end
            else begin
                BCD <= BCD-1;
                carry <= 0;
            end
        end
        else begin
            if(BCD==max) begin
                BCD <= 0;
                if(carrycontrol==0)
                    carry <= 1;
            end
            else begin
                BCD <= BCD+1;
                carry <= 0;
            end
        end
```

7 Segment Decoder

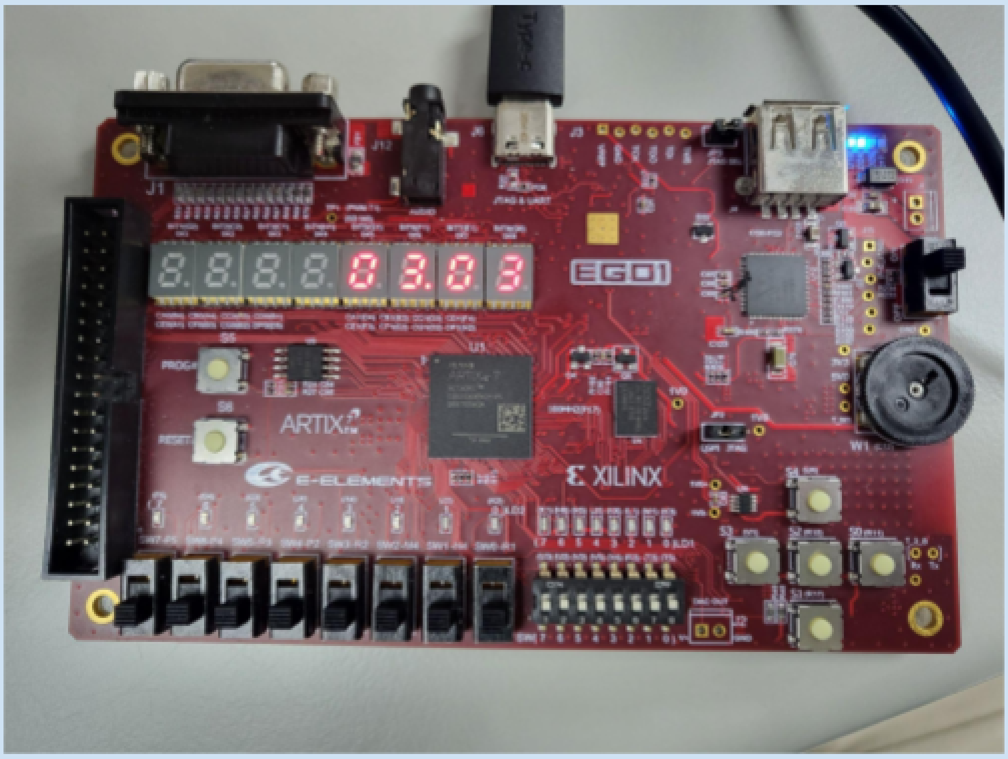


```
module seven(BCD, OUT);
    input [3:0] BCD; //the input in BCD
    output reg [6:0] OUT; //the out put for the 7SD

    always@(BCD) // this controls how the 7SD lights up
        case({BCD})
            4'b0000: {OUT} = 7'b1111110; // 0
            4'b0001: {OUT} = 7'b0110000; // 1
            4'b0010: {OUT} = 7'b1101101; // 2
            4'b0011: {OUT} = 7'b1111001; // 3
            4'b0100: {OUT} = 7'b0110011; // 4
            4'b0101: {OUT} = 7'b1011011; // 5
            4'b0110: {OUT} = 7'b1011111; // 6
            4'b0111: {OUT} = 7'b1110000; // 7
            4'b1000: {OUT} = 7'b1111111; // 8
            4'b1001: {OUT} = 7'b1111011; // 9
        endcase
endmodule
```

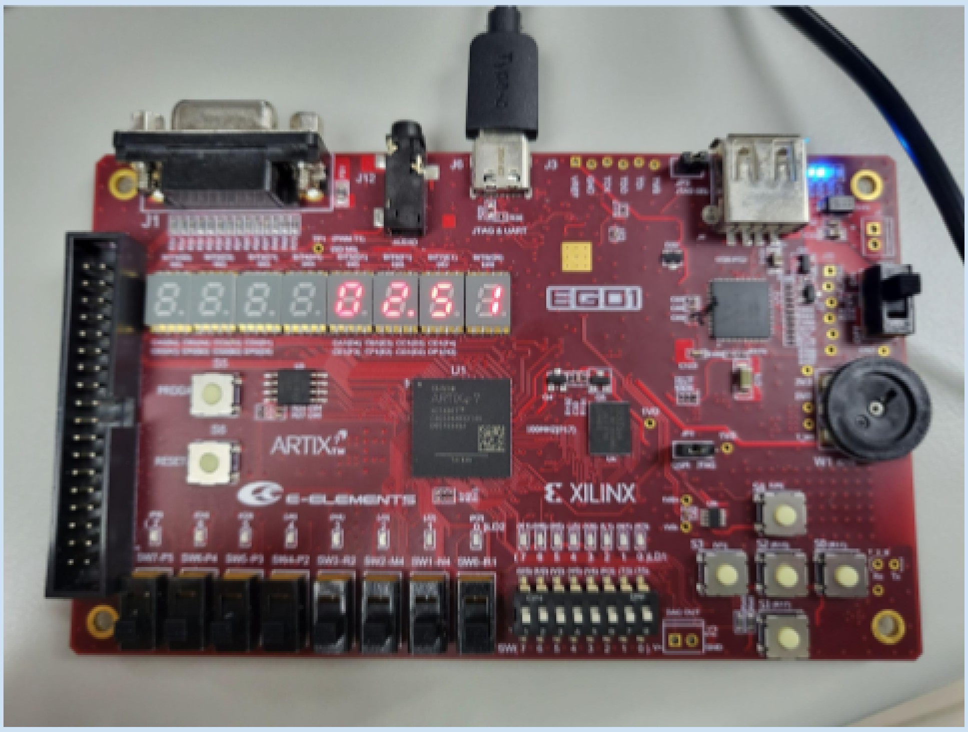
FPGA Results



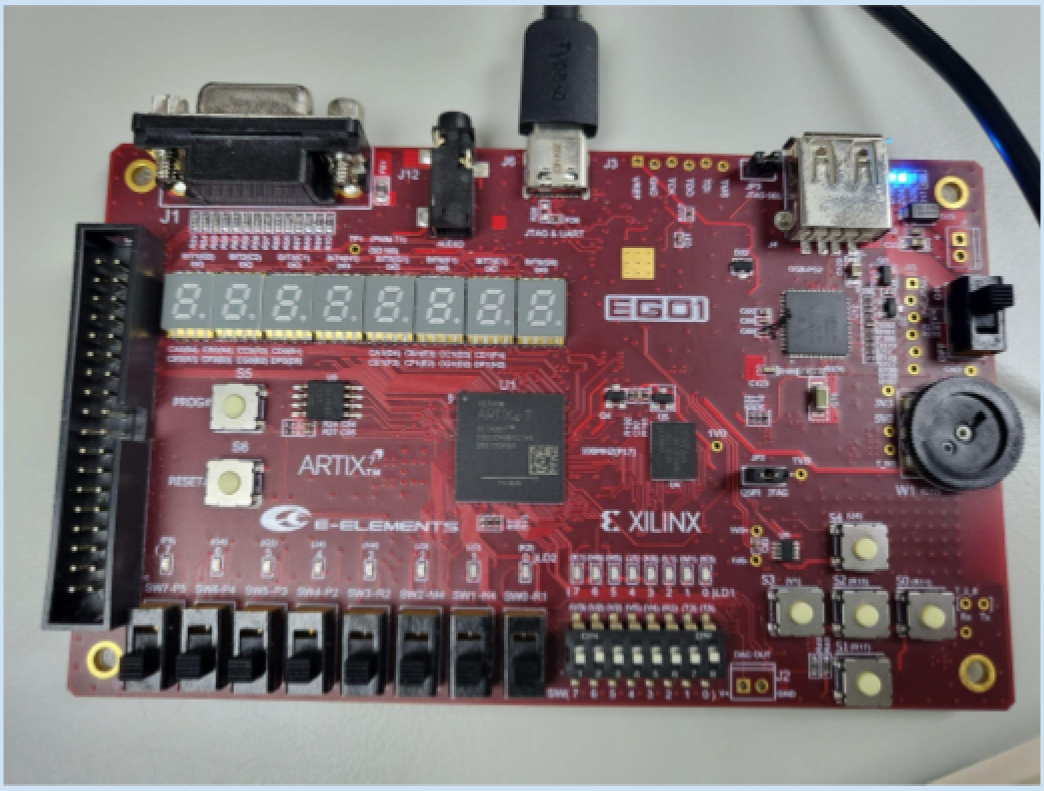
Initially all the digits are 0.



The timer is set to 3 minutes and 3 seconds.



The timer is counting down.



The countdown is over and the display is blinking (it is off).