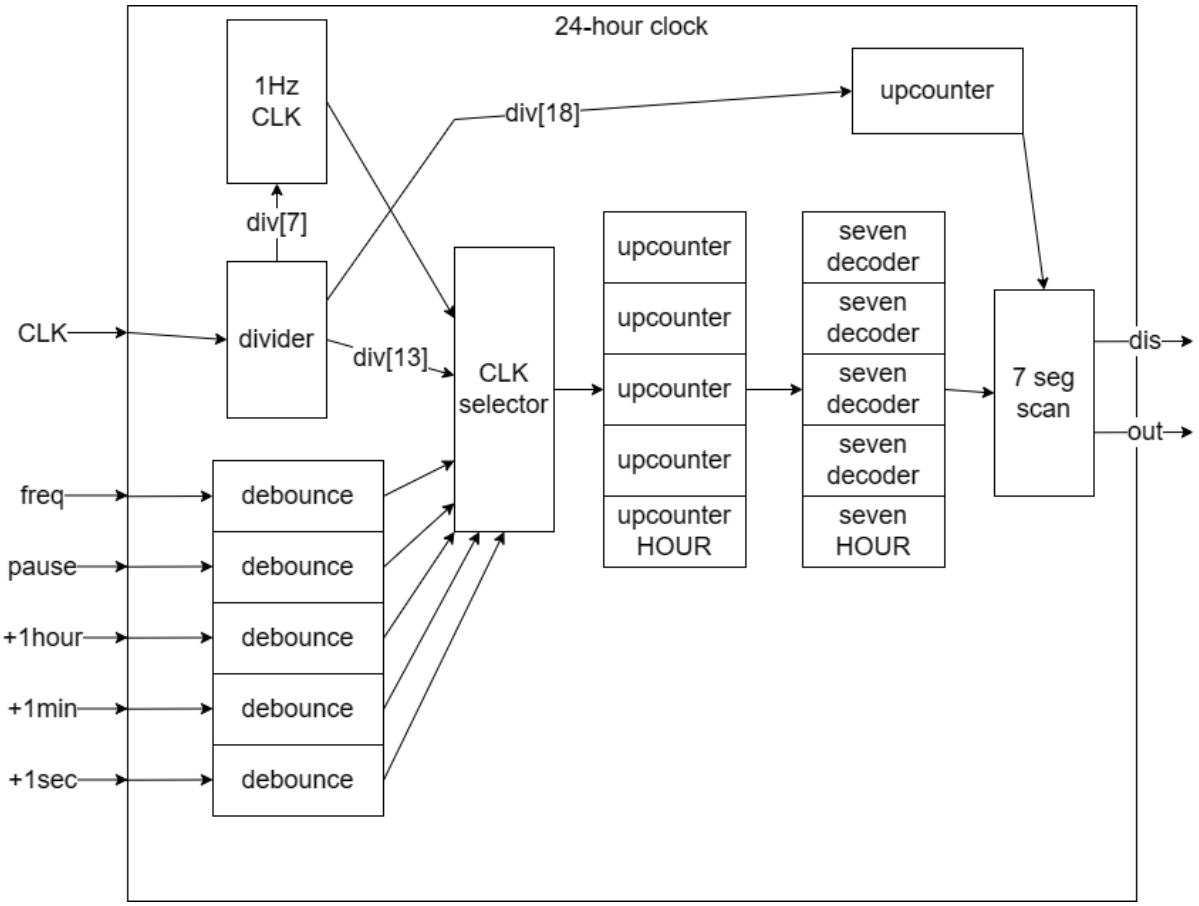
Lab 6 report

Name: Oscar Stark Student ID: F11015127

In this lab I implemented a 24-hour clock displayed in seven segments with a reset button. Only the code for the advanced task is shown on this report, because the advanced is just an extension of the basic task.

Circuit block diagram



Verilog code

Main code

```
module task1(CLK, BplusSec, BplusMin, BplusHour, OUTleft, OUTright, DIS, Spause, Sfreq, reset);
    input CLK, BplusSec, BplusMin, BplusHour, Spause, Sfreq, reset;
    output reg [6:0] OUTleft, OUTright; //for the 4 7SD on the left and the 4 on the right
    output reg [5:0] DIS; //to choose which 7SD lights up

    //wires
    wire [31:0] DIV;
    wire BplusSec_CLK, BplusMin_CLK, BplusHour_CLK, Spause_CLK, Sfreq_CLK;
    wire [3:0] BCD1, BCD2, BCD3, BCD4, HOUR;
    wire [6:0] OUT1, OUT2, OUT3, OUT4, OUT5;
    wire [13:0] OUT56; //this out is different because it's for the hours
    wire [4:0] hour;
    wire C1, C2, C3, C4;
    wire CLK_1Hz;
    //registers
    reg CLK_sec, CLK_min, CLK_hour;

    divider DIVIDER(CLK, DIV, reset);
    debounce Dsec(DIV[16], BplusSec, BplusSec_CLK, reset);
    debounce Dmin(DIV[16], BplusMin, BplusMin_CLK, reset);
    debounce Dhour(DIV[16], BplusHour, BplusHour_CLK, reset);
    debounce Dpause(DIV[16], Spause, Spause_CLK, reset);
    debounce Dfreq(DIV[16], Sfreq, Sfreq_CLK, reset);

    //1Hz Clock. DIV[7]*390625 = 1s
    upcounter1Hz(DIV[7], reset, 390624, CLK_1Hz);
```

```
    //Clock selector
    always@(CLK)
        begin
            if(Spause_CLK)//if the 24h clock is paused,
            begin         //the buttons can be used to adjust the time
                CLK_sec <= BplusSec_CLK;
                CLK_min <= BplusMin_CLK;
                CLK_hour <= BplusHour_CLK;
            end
            else
            begin            //if the 24h clock is not paused,
                if(Sfreq_CLK)//a switch is used to choose the frequency
                    CLK_sec <= DIV[13];
                else
                    CLK_sec <= CLK_1Hz;
                CLK_min <= C2;
                CLK_hour <= C4;
            end
        end

////////////////////////////////////////////////////////////////////////////////
    //2 upcounters for the seconds, 2 upcounters for the minutes,
    //and 1 upcounterHOUR for the hours
    upcounter UPC1(CLK_sec, BCD1, reset, 5, 9, C1, 0);
    upcounter UPC2(C1, BCD2, reset, 4, 5, C2, Spause_CLK);
    upcounter UPC3(CLK_min, BCD3, reset, 9, 9, C3, 0);
    upcounter UPC4(C3, BCD4, reset, 5, 5, C4, Spause_CLK);
    upcounterHOUR UPC56(CLK_hour, HOUR, reset, 23, 23);

////////////////////////////////////////////////////////////////////////////////
    //2 seven decoders for the seconds, 2 seven decoders for the minutes,
    //and 1 sevenHOUR for the hours
    seven SEV1(BCD1, OUT1);
    seven SEV2(BCD2, OUT2);
    seven SEV3(BCD3, OUT3);
    seven SEV4(BCD4, OUT4);
    sevenHOUR SEV56(HOUR, OUT56);
```

```
    //seven seg scan
    wire [3:0] DIScounter;//every DIV[18] the display lit up changes
    upcounter DISupc(DIV[18], DIScounter, reset, 0, 2);
    always@(DIScounter)
        begin
            if(DIScounter==0)
            begin
                OUTright <= OUT1;
                OUTleft <= OUT4;
                DIS <= 6'b001001;
            end
            else
            if(DIScounter==1)
            begin
                OUTright <= OUT2;
                OUTleft <= OUT56[6:0];
                DIS <= 6'b010010;
            end
            else
            if(DIScounter==2)
            begin
                OUTright <= OUT3;
                OUTleft <= OUT56[13:7];
                DIS <= 6'b100100;
            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
```

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
```

Debounce

```
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
```

1Hz clock

```
//upcounter modified to work as a 1Hz clock
module upcounter1Hz(CLK, reset, max, carry);
    input CLK, reset;
    input [18:0] max;
    output reg carry;
    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
```

Up counter

```
//the upcounter has a lot of arguments so it can be used in many
//different ways
module upcounter(CLK, BCD, reset, resetvalue, max, carry, carrycontrol);
    input CLK, reset, carrycontrol;
    input [3:0] resetvalue;
    input [3:0] max;
    output reg [3:0] BCD;
    output reg carry;

    always @(posedge CLK, posedge reset)
    begin
        if(reset)
        begin
            BCD <= resetvalue;
            carry <= 0;
        end
        else
        if(BCD==max)//max is the maximum digit
        begin
            BCD <= 0;
            if(carrycontrol==0)//carrycontrol is used to stop the carry
                carry <= 1;     //from being set to 1 while adjusting the time
        end
        else
        begin
            BCD <= BCD+1;
            carry <= 0;
        end
    end
endmodule
```

Hour up counter

```
//upcounter modified to represent the hours
module upcounterHOUR(CLK, HOUR, reset, resetvalue, max);
    input CLK, reset;
    input [4:0] resetvalue;
    input [4:0] max;
    output reg [4:0] HOUR;

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

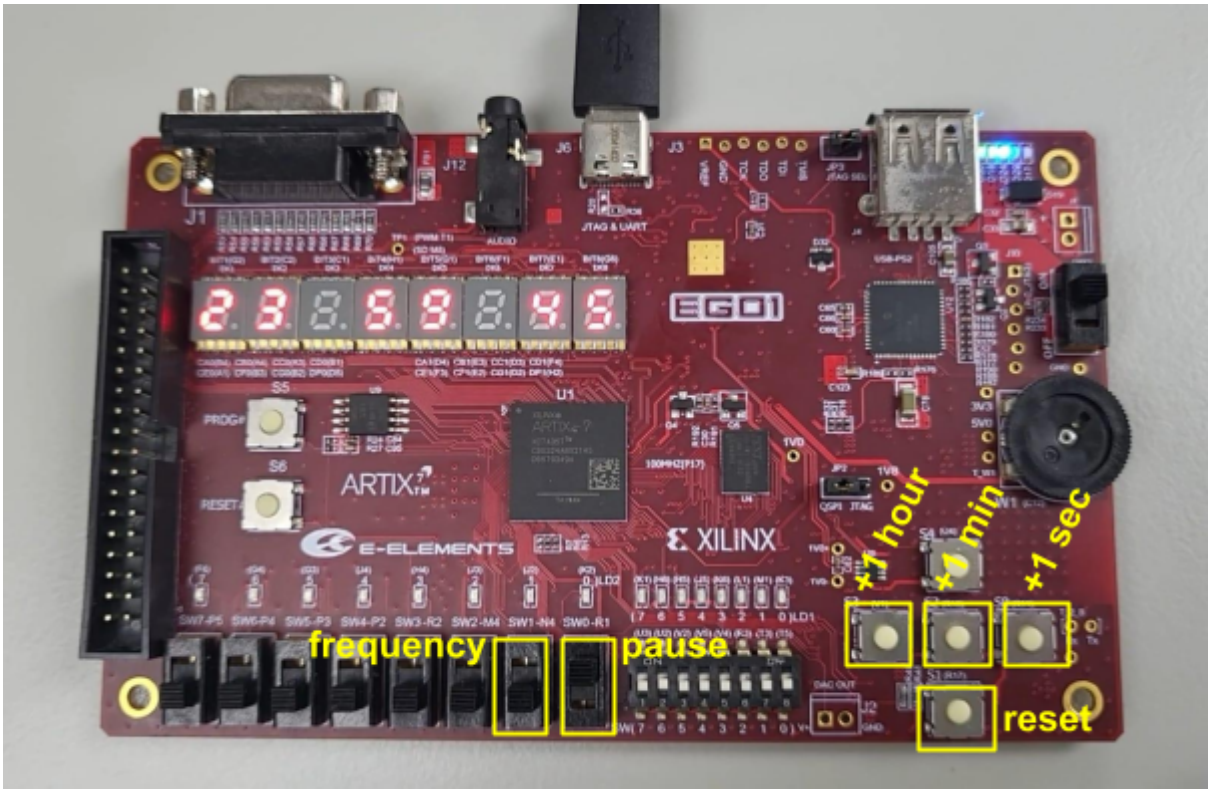
Seven segment decoder

```
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
```

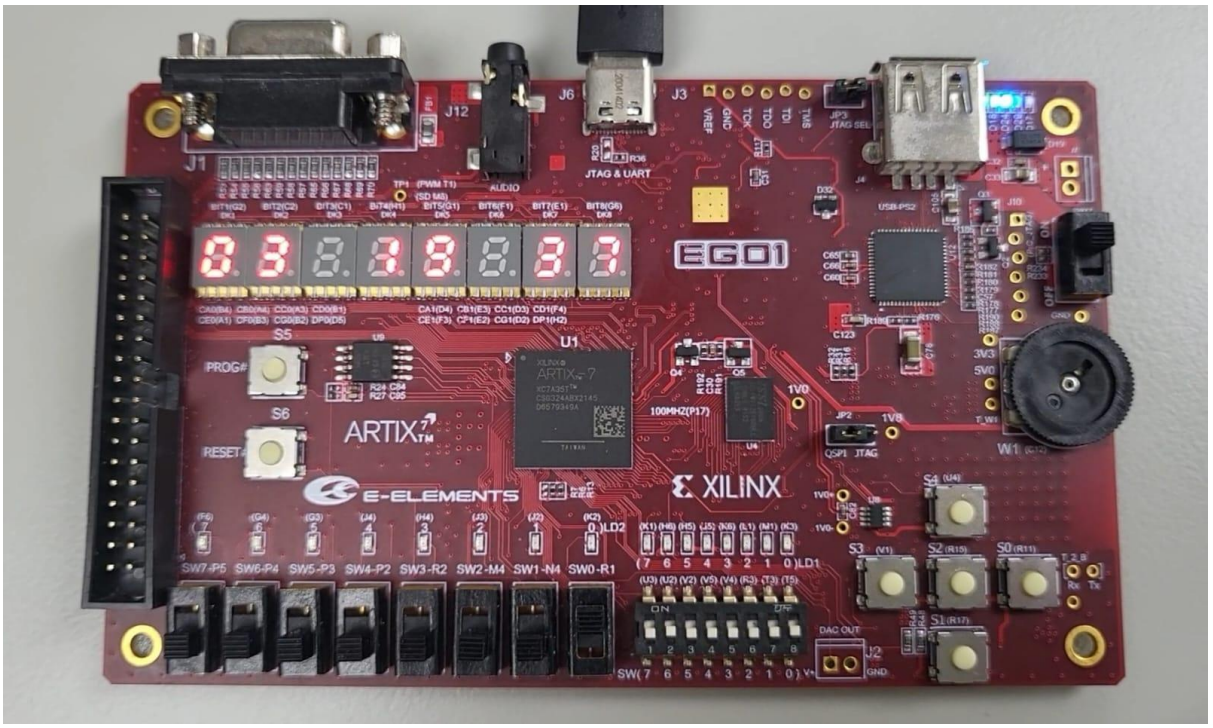
Hour seven segment decoder

```
//seven decoder modified to represent the hours
module sevenHOUR(HOUR, OUT);
    input [4:0] HOUR;
    output reg [13:0] OUT;

    always@(HOUR) // this controls how the 7SD lights up
        case({HOUR})
            5'b00000: {OUT} = 14'b11111101111110; // 0
            5'b00001: {OUT} = 14'b11111100110000; // 1
            5'b00010: {OUT} = 14'b11111101101101; // 2
            5'b00011: {OUT} = 14'b11111101111001; // 3
            5'b00100: {OUT} = 14'b11111100110011; // 4
            5'b00101: {OUT} = 14'b11111101011011; // 5
            5'b00110: {OUT} = 14'b11111101011111; // 6
            5'b00111: {OUT} = 14'b11111101110000; // 7
            5'b01000: {OUT} = 14'b11111101111111; // 8
            5'b01001: {OUT} = 14'b11111101111011; // 9
            5'b01010: {OUT} = 14'b01100001111110; // 10
            5'b01011: {OUT} = 14'b01100000110000; // 11
            5'b01100: {OUT} = 14'b01100001101101; // 12
            5'b01101: {OUT} = 14'b01100001111001; // 13
            5'b01110: {OUT} = 14'b01100000110011; // 14
            5'b01111: {OUT} = 14'b01100001011011; // 15
            5'b10000: {OUT} = 14'b01100001011111; // 16
            5'b10001: {OUT} = 14'b01100001110000; // 17
            5'b10010: {OUT} = 14'b01100001111111; // 18
            5'b10011: {OUT} = 14'b01100001111011; // 19
            5'b10100: {OUT} = 14'b11011011111110; // 20
            5'b10101: {OUT} = 14'b11011010110000; // 21
            5'b10110: {OUT} = 14'b11011011101101; // 22
            5'b10111: {OUT} = 14'b11011011111001; // 23
        endcase
endmodule
```

FPGA board results



The time is 23:59:45. The clock resets to this time.



The time is 3:19:37.