Kassandra Marquez
Compe470L
9 November 2021

Lab 6 - Digital Clock - COPME470L

I. Introduction

Within this lab, we will design a digital clock that features a ‘set feature’ and ‘alarm feature’. Three modes are required: clock counting mode with alarm off, alarm setting mode, and clock counting mode with alarm on. The digital clock will be implemented on the Basys3 where the 7 segment display is needed, one switch button to set alarm mode, and a few LEDs to represent that the alarm went off.

II. Source Code

The code is divided into 2 modules. One module, ‘clock’, will have our 1 second timer, clock counting logic that will cause 60 seconds to turn into a minute etc., our finite state machine for the different modes, and the logic to output which specific number at the specified place of the 7 segment display. To represent these placements, we have 4 variables named ‘hourten, ‘hour’, ‘minten’, and ‘min’. For example, hourten is the MSB of the hour, so it will take /10 (integer) while min is the LSB of a minute hence it will take %(modulo) 10.

Within the FSM, we will have our count regular clock logic by increasing the seconds and setting alarm clock logic which is controlled by btnL, btnR, btnU, btnD along with its alarm mode. The middle button will trigger the set alarm mode.
Pressing left or right will increase or decrease the hours. Pressing up and down will increase or decrease the minutes. When the switch0 is on, set alarm clock mode exits and alarm mode is on.

The second module, ‘sev_seg’, controls the display of the alarm clock. Since the 7 segment display cannot display more than one number at a time, we will implement delay logic to trick the human eye.

```
//clock----------------------------------------------------------------------------------------------
`timescale 1ns / 1ps

module clock(

input clk,

//REGULAR CLOCK push button
input rst, //regular clock mode -> PUSH BUTTON

///push buttons to set alarm clocks
input btnC, btnU, btnD, btnR, btnL,
```

Kassandra Marquez
Compe470L
9 November 2021

```
input [3:0] sw,

output [6:0] seg,
output [3:0] an,
output reg [3:0] led //1b
    );

//**************MEMORY REGS*******************

//1second counter, Basys3 - 100MHz
reg[31:0] count = 0;
parameter max_count = 1000000; //1562500;

//count clock logic
reg [5:0] mins, secs = 0;
reg [5:0] hours = 12;
//count alarm clock logic
reg [5:0] a_mins, a_secs, a_hours = 0;
//og clock vals before alarm setting mode
reg [5:0] og_hours, og_secs, og_mins = 0;

//CLK regs - set to 7 seg displays
reg [3:0] c_hourten, c_hour, c_minten, c_min = 0;

reg [0:0] current_bit = 0; //when setting clock, this chooses between setting hr or min



//inst seven seg display mod
sev_seg display(.clk(clk), .s_hourten(c_hourten), .s_hour(c_hour), .s_minten(c_minten),
.s_min(c_min) ,.an(an), .seg(seg)) ;

//Clock Modes
parameter normal_mode = 2'b00;
parameter set_alarm_clock = 2'b01;
parameter alarm_mode = 2'b10;
reg [1:0] current_mode = normal_mode; //default
```

Kassandra Marquez
Compe470L
9 November 2021

```
always @(posedge clk) begin
   //**mode states
   led = 4'b0000;

   case(current_mode)
      normal_mode:
         begin
         if(a_hours == hours && a_mins == mins)
                        begin
                           led = 4'b1111;
                        end
         else
                        begin
                           led = 4'b0000;
                        end
            if(btnC) //middle button
               begin
                  count <= 0;
                  secs <= 0;
                  current_bit <= 0;
                  current_mode <= set_alarm_clock;
               end
            if (count < max_count)
               begin
                  count <= count + 1;
               end
            else
               begin
                  count <= 0;
                  secs <= secs + 1;

                  //save og time
                  og_secs = secs;
                  og_mins = mins;
                  og_hours = hours;
               end
         end//normal mode

      set_alarm_clock:
         begin
```

Kassandra Marquez
Compe470L
9 November 2021

```
count <= 0;
secs <= 0;
current_bit <= 0;
a_mins <= mins;
a_hours <= hours;

if(btnC) //if middle button pressed again, go back to normal mode
   begin
      current_mode = normal_mode;
   end


if(count < 25000000)
   begin
      count <= count + 1;
   end

else
   begin
      if(btnU) //up button -> inc minutes
         begin
            mins <= mins + 1;
         end
     if(btnD) //down button -> dec minutes
        begin
            if(mins > 0)
               mins <= mins - 1;

       end

    if(btnL)
        begin
            hours <= hours + 1;
        end
    if(btnR)
        begin
              if(hours > 1)
                        begin
                           hours <= hours - 1;
```

Kassandra Marquez
Compe470L
9 November 2021

```
                                    end
                        end




            end//end else

          //trigger FSM
          if(sw == 4'b0001)
           begin
            //alarm_mode is on
            current_mode = alarm_mode;
           end

          else
           begin
            //stay in set_alarm_clk
            current_mode = set_alarm_clock;
           end

       end//set alarm clock


    alarm_mode: //switch is ON! need LED
       begin
         if(sw == 4'b0000)
            begin
               current_mode = normal_mode;
            end

         if(sw == 4'b0001)
            begin
                     secs = og_secs;
                     mins = og_mins;
                     hours = og_hours;


            end  //end else
       end//alarm mode
```

Kassandra Marquez
Compe470L
9 November 2021

```
    endcase

    //**REGULAR clock count logic
    if(secs >= 60)
        begin
            secs <= 0;
            mins <= mins + 1;
        end
    if(mins >= 60)
        begin
            mins <= 0;
            hours <= hours + 1;
        end

    if(hours == 13)
        begin
            hours <= 1; //rst hours
        end

    //**set outputs to display**
    c_min <= mins % 10; //take remainder / 10 since LSB of min
    c_minten <= mins / 10; //take int math /10 since MSB of min
    if(hours < 10) //0-9
        begin
            c_hourten <= 0;
            c_hour <= hours % 10;
        end
    else //10-12
        begin
            c_hourten <= hours / 10;
            c_hour <= hours % 10;
        end



end//end posedge clk
endmodule//clk mod

//**********************************************************************
```

Kassandra Marquez
Compe470L
9 November 2021

```
//7 SEG DISPLAY----------------------------------------------------------------------------------------
`timescale 1ns / 1ps

module sev_seg(

input clk,
input [3:0] s_hourten, s_hour, s_minten, s_min,
output reg[3:0] an, //4 places display , or 7seg selector! 'an' from const.
output reg[6:0] seg //7 segment display # 'seg' from const.

    );

//**************MEMORY REGS*******************
reg[1:0] current_place = 0; //are we in hourten,hr,mint, min? (0-3)
reg[6:0] seg_out [3:0]; //we will use 'curr_place' for 3-0

reg [18:0] count = 0; //timer for illusion
parameter max_count = 500000; //500,000/100Mhz -> 5ms

//***CONNECT 7seg to CLK module*************
wire [3:0] four_b_data [3:0]; //size 4b will array 0-4 ea rep hourten,hr,mtn,min..
assign four_b_data[0] = s_min;
assign four_b_data[1] = s_minten;
assign four_b_data[2] = s_hour;
assign four_b_data[3] = s_hourten;

always @(posedge clk) begin
    if(count <= max_count)
        begin
            count <= count + 1;
        end
    else
        begin
            current_place <= current_place + 1;
            count <= 0;

        end

    case(four_b_data[current_place])
```

Kassandra Marquez
Compe470L
9 November 2021

```
        4'b0000: seg_out[current_place] <= 7'b1000000;
        4'b0001: seg_out[current_place] <= 7'b1111001;
        4'b0010: seg_out[current_place] <= 7'b0100100;
        4'b0011: seg_out[current_place] <= 7'b0110000;
        4'b0100: seg_out[current_place] <= 7'b0011001;
        4'b0101: seg_out[current_place] <= 7'b0010010;
        4'b0110: seg_out[current_place] <= 7'b0000010;
        4'b0111: seg_out[current_place] <= 7'b1111000;
        4'b1000: seg_out[current_place] <= 7'b0000000;
        4'b1001: seg_out[current_place] <= 7'b0011000;
        default:seg_out[current_place] <= 7'b0000000;
    endcase

case(current_place)
    0:
        begin
            an<= 4'b1110;
            seg <= seg_out[0];
        end

    1:
        begin
            an<= 4'b1101;
            seg <= seg_out[1];
        end
    2:
        begin
            an<= 4'b1011;
            seg <= seg_out[2];
        end
    3:
        begin
            an<= 4'b0111;
            seg <= seg_out[3];
        end
endcase
end //end always
endmodule

//**********************************************************************
```

Kassandra Marquez
Compe470L
9 November 2021

III. Demo

https://youtu.be/jCI8hgL1nb0

IV. Conclusion

As a result of this lab, I learned how to control a 7segment display through my clock logic and also trick the human eye when trying to display more than one number at a time so that our digital clock can be displayed. I also understood the difference between / and % as it comes in handy when choosing which part of the minute or hour is to be displayed.