EE 316 – Lab 6 Report: *Custom Processor Design: Programmable Stopwatch/Timer*

Ayan Basu

a) HLSM Design

a) HLSM Design

Inputs: start stop, reset, sw [7:0], mode [1:0]

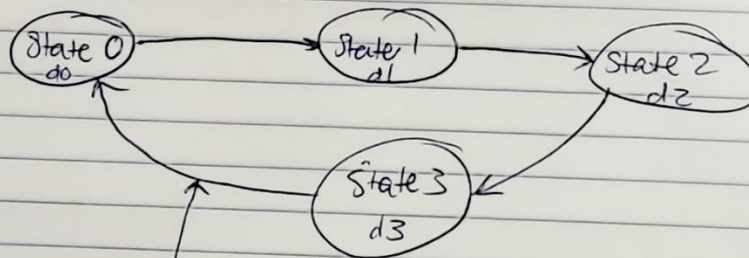**Output:** an[3:0], sseg [6:0]

Clk-divider

Input: clk, reset

output: refresh-clk, time clk

Stopwatch

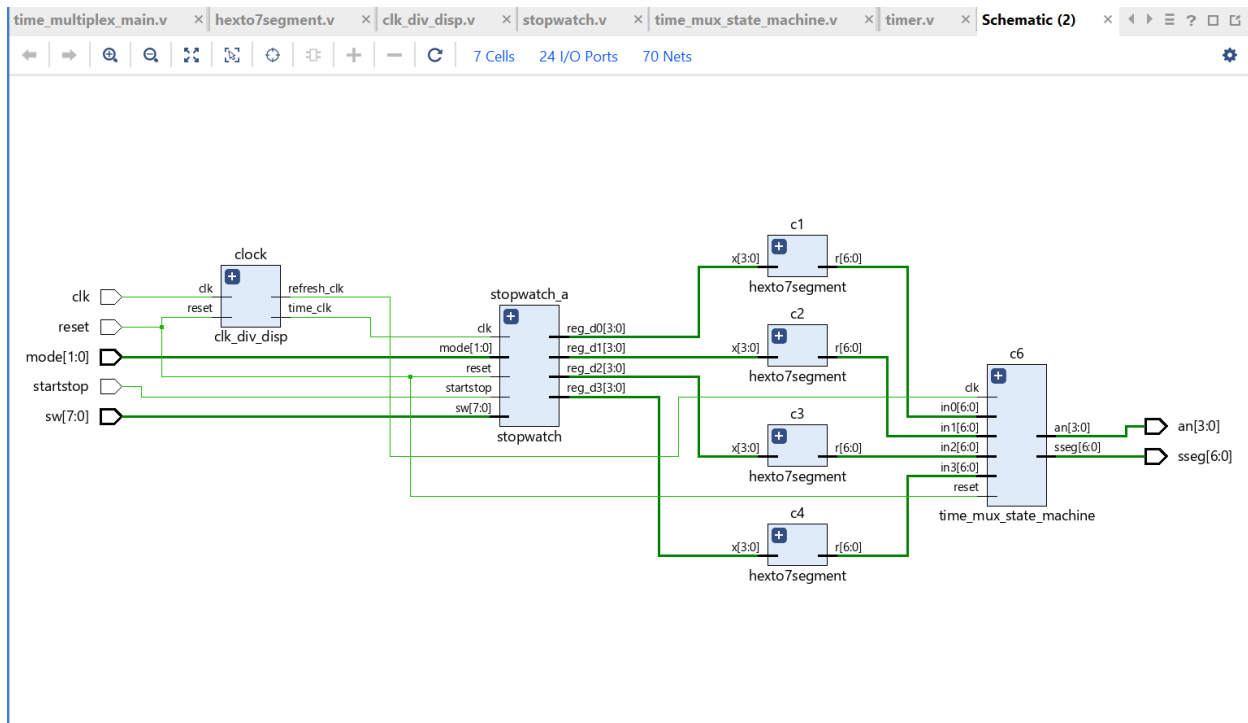Input: clk, StartStop, reset, mode [1:0], switch [7:0]

Output: reg=d0-d3, components for each digit

State Machine

State 0 d0 → State 1 d1 → State 2 d2 → State 3 d3 → (back to State 0)

transitions on each pos edge of clk

b)   Processor Architecture

c)   Verilog Code

constraint.xdc

Clock signal

set_property PACKAGE_PIN W5 [get_ports clk]

        set_property IOSTANDARD LVCMOS33 [get_ports clk]

        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

Switches

set_property PACKAGE_PIN V2 [get_ports {sw[0]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]

set_property PACKAGE_PIN T3 [get_ports {sw[1]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]

set_property PACKAGE_PIN T2 [get_ports {sw[2]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]

set_property PACKAGE_PIN R3 [get_ports {sw[3]}]

```
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
set_property PACKAGE_PIN W2 [get_ports {sw[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
set_property PACKAGE_PIN U1 [get_ports {sw[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
set_property PACKAGE_PIN T1 [get_ports {sw[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
set_property PACKAGE_PIN R2 [get_ports {sw[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]



set_property PACKAGE_PIN V17 [get_ports {mode[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {mode[0]}]
set_property PACKAGE_PIN V16 [get_ports {mode[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {mode[1]}]



set_property PACKAGE_PIN W7 [get_ports {sseg[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sseg[6]}]
set_property PACKAGE_PIN W6 [get_ports {sseg[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sseg[5]}]
set_property PACKAGE_PIN U8 [get_ports {sseg[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sseg[4]}]
set_property PACKAGE_PIN V8 [get_ports {sseg[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sseg[3]}]
set_property PACKAGE_PIN U5 [get_ports {sseg[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sseg[2]}]
set_property PACKAGE_PIN V5 [get_ports {sseg[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sseg[1]}]
```

set_property PACKAGE_PIN U7 [get_ports {sseg[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[0]}]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]

set_property PACKAGE_PIN U4 [get_ports {an[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]

set_property PACKAGE_PIN V4 [get_ports {an[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]

set_property PACKAGE_PIN W4 [get_ports {an[3]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]

set_property PACKAGE_PIN T18 [get_ports startstop]

set_property IOSTANDARD LVCMOS33 [get_ports startstop]

set_property PACKAGE_PIN U18 [get_ports reset]

set_property IOSTANDARD LVCMOS33 [get_ports reset]

clk_div_disp.v

```verilog
module clk_div_disp(
    input clk,
    input reset,
    output refresh_clk,
    output time_clk
    );

    reg [16:0] refresh;
    reg [19:0] time_count;
    reg ref;
```

```verilog
    reg time_c;

    assign refresh_clk = ref;

    assign time_clk = time_c;


    always @(posedge clk) begin

        if (time_count < 500000) begin

            time_count <= time_count + 1;

        end

        else begin

            time_c <= ~time_c;

            time_count <= 0;

        end

    end


    always @(posedge clk) begin

        if (refresh < 100000) begin

            refresh <= refresh + 1;

        end

        else begin

            ref <= ~ref;

            refresh <= 0;

        end

    end

endmodule
```

hexto7segment.v

```verilog
module hexto7segment(

    input [3:0] x,
```

```verilog
    output reg [6:0] r

);


always @(*)
  case(x)
    4'b0000: r = 7'b0000001;

    4'b0001: r = 7'b1001111;

    4'b0010: r = 7'b0010010;

    4'b0011: r = 7'b0000110;

    4'b0100: r = 7'b1001100;

    4'b0101: r = 7'b0100100;

    4'b0110: r = 7'b0100000;

    4'b0111: r = 7'b0001111;

    4'b1000: r = 7'b0000000;

    4'b1001: r = 7'b0000100;

  endcase


endmodule
```

stopwatch.v
```verilog
module stopwatch(
    input clk,

    input startstop,

    input reset,

    input [1:0] mode,

    input [7:0] sw,

    output reg [3:0] reg_d0, //count for right most digit

    output reg [3:0] reg_d1, //count for 2nd right most digit

    output reg [3:0] reg_d2, //count for 2nd left most digit
```

```verilog
   output reg [3:0] reg_d3  //count for left most digit

   );


   reg startstop_ff;

   reg ss = 1;

   reg finish;

   reg idle = 1;


always @ (posedge clk) begin

   startstop_ff <= startstop;

   if( startstop_ff && !startstop)

      ss <= ~ss;

end


always @ (posedge clk) begin


/*--------------- Mode 1 (Counting Up from 00.00) ---------------*/

   if (mode == 2'b00) begin

      if (ss == 1 && reset == 1) //if both stop & reset asserted

         begin

            reg_d0 <= 0; //counter0 is 0

            reg_d1 <= 0; //counter1 is 0

            reg_d2 <= 0; //counter2 is 0

            reg_d3 <= 0; //counter3 is 0

            finish = 0;

         end


      // if only stop signal is asserted, store the previous count

      // when stop button is pressed again, resume the old count
```

```verilog
    else if (ss == 1 && reset != 0)

        begin

            //store the old count

            reg_d0 <= reg_d0;

            reg_d1 <= reg_d1;

            reg_d2 <= reg_d2;

            reg_d3 <= reg_d3;

        end


    //Start stopwatch
    else if (ss != 1 && finish != 1)

        begin

            if(reg_d0 == 9) begin

                reg_d0 <= 0;

                if (reg_d1 == 9) begin

                    reg_d1 <= 0;

                    if (reg_d2 == 9) begin

                        reg_d2 <= 0;

                        if(reg_d3 == 9)begin

                            reg_d2 <= 9;

                            reg_d1 <= 9;

                            reg_d0 <= 9;

                            finish = 1;

                        end else

                            reg_d3 <= reg_d3 + 1;

                    end else

                        reg_d2 <= reg_d2 + 1;

                end else

                    reg_d1 <= reg_d1 + 1;
```

```verilog
         end else
             reg_d0 <= reg_d0 + 1;
         end
     end


/*--------------- Mode 2 (Counting Up from XX.00) ---------------*/
   if (mode == 2'b01) begin
      if(ss == 1 && reset == 0 && idle == 1) begin
          reg_d0 <= 0; //counter0 is 0
          reg_d1 <= 0; //counter1 is 0
          reg_d2 <= sw[3:0]; //counter2 is 0
          reg_d3 <= sw[7:4]; //counter3 is 0
          finish = 0;
          end
      if (ss == 1 && reset ==1) //if both stop & reset asserted
            begin
               reg_d0 <= 0; //counter0 is 0
               reg_d1 <= 0; //counter1 is 0
               reg_d2 <= sw[3:0]; //counter2 is 0
               reg_d3 <= sw[7:4]; //counter3 is 0
               finish = 0;
               idle = 1;
            end


        // if only stop signal is asserted, store the previous count
        // when stop button is pressed again, resume the old count
        else if (ss == 1 && reset != 0)
           begin
              //store the old count
```

```verilog
        reg_d0 <= reg_d0;

        reg_d1 <= reg_d1;

        reg_d2 <= reg_d2;

        reg_d3 <= reg_d3;

        idle = 1;

    end


//Start stopwatch
else if (ss != 1 && finish != 1)
  begin
   idle = 0;
    if(reg_d0 == 9) begin
      reg_d0 <= 0;
      if (reg_d1 == 9) begin
          reg_d1 <= 0;
          if (reg_d2 == 9) begin
              reg_d2 <= 0;
              if(reg_d3 == 9)begin
                reg_d2 <= 9;
                reg_d1 <= 9;
                reg_d0 <= 9;
                finish = 1;
                end else
                reg_d3 <= reg_d3 + 1;
            end else
              reg_d2 <= reg_d2 + 1;
        end else
          reg_d1 <= reg_d1 + 1;
    end else
```

```verilog
            reg_d0 <= reg_d0 + 1;
        end
    end



/*--------------- Mode 3 (Counting Down from 99.99) ---------------*/
  if (mode == 2'b10) begin
    if (ss == 1 && reset ==1)
    begin
        reg_d0 <= 9;
        reg_d1 <= 9;
        reg_d2 <= 9;
        reg_d3 <= 9;
    end

    else if (ss == 1 && reset != 0)
    begin
        reg_d0 <= reg_d0; //store the old count
        reg_d1 <= reg_d1; //store the old count
        reg_d2 <= reg_d2; //store the old count
        reg_d3 <= reg_d3; //store the old count
    end

    else if (ss != 1 && finish == 0) begin

      if(reg_d0 == 0) begin
        reg_d0 <= 9;
        if (reg_d1 == 0) begin
            reg_d1 <= 9;
```

```verilog
            if (reg_d2 == 0) begin

                reg_d2 <= 9;

                if(reg_d3 == 0) begin

                    reg_d0 <= 0;

                    reg_d1 <= 0;

                    reg_d2 <= 0;

                    reg_d3 <= 0;

                    finish = 1;

                end else

                    reg_d3 <= reg_d3 - 1;

            end else

                reg_d2 <= reg_d2 - 1;

        end else

            reg_d1 <= reg_d1 - 1;

    end else

        reg_d0 <= reg_d0 - 1;

    end

end


/*-------------- Mode 4 (Counting Down from XX.00) --------------*/

if (mode == 2'b11) begin

  if (ss == 1 && reset == 0  && idle == 1) begin

    reg_d0 <= 0;

    reg_d1 <= 0;

    reg_d2 <= sw[3:0];

    reg_d3 <= sw[7:4];

  end

  else if (ss == 1 && reset != 0) begin

    reg_d0 <= reg_d0; //store the old count
```

```verilog
        reg_d1 <= reg_d1; //store the old count
        reg_d2 <= reg_d2; //store the old count
        reg_d3 <= reg_d3; //store the old count
        idle = 1;
    end
    else if (ss != 1 && finish == 0) begin //if no stop
      idle = 0;
    if(reg_d0 == 0) begin
        reg_d0 <= 9;
        if (reg_d1 == 0) begin
            reg_d1 <= 9;
            if (reg_d2 == 0) begin
                reg_d2 <= 9;
                if(reg_d3 == 0) begin
                    reg_d0 <= 0;
                    reg_d1 <= 0;
                    reg_d2 <= 0;
                    reg_d3 <= 0;
                    finish = 1;
                end else
                    reg_d3 <= reg_d3 - 1;
            end else
                reg_d2 <= reg_d2 - 1;
        end else
            reg_d1 <= reg_d1 - 1;
    end else
      reg_d0 <= reg_d0 - 1;
    end
end
```

end

endmodule

time_multiplex_main.v

```verilog
module time_multiplex_main(
    input clk,
    input startstop,
    input reset,
    input [7:0] sw,
    input [1:0] mode,
    output [3:0] an,
    output [6:0] sseg
    );


    wire [6:0] in0, in1, in2, in3;
    wire refresh_clk;
    wire time_clk;
    wire [3:0] reg_d0; //count for right most digit
    wire [3:0] reg_d1; //count for second right most digit
    wire [3:0] reg_d2; //count for second left most digit
    wire [3:0] reg_d3; //count for left most digit



    // Module instantiation of hexto7segment decoder
    hexto7segment c1 (.x(reg_d0), .r(in0));
    hexto7segment c2 (.x(reg_d1), .r(in1));
```

```verilog
hexto7segment c3 (.x(reg_d2), .r(in2));

hexto7segment c4 (.x(reg_d3), .r(in3));


// Module instantiation of clock divider

// same functionality as the clk_div before, but may have a different width requirement

clk_div_disp clock (.clk(clk), .reset(reset), .refresh_clk(refresh_clk), .time_clk(time_clk));



stopwatch stopwatch_a (

    .clk(time_clk),

    .startstop(startstop),

    .reset(reset),

    .mode(mode[1:0]),

    .sw(sw[7:0]),

    .reg_d0(reg_d0),

    .reg_d1(reg_d1),

    .reg_d2(reg_d2),

    .reg_d3(reg_d3));



// Module instantiation of the multiplexer

//replace slow_clk with clk for simulation, and vice versa

time_mux_state_machine c6 (

    .clk(refresh_clk),

    .reset(reset),

    .in0(in0),

    .in1(in1),

    .in2(in2),

    .in3(in3),
```

```verilog
        .an(an),

        .sseg(sseg));



Endmodule
```

time_mux_state_machine.v

```verilog
module time_mux_state_machine(
    input clk,
    input reset,
    input [6:0] in0,
    input [6:0] in1,
    input [6:0] in2,
    input [6:0] in3,
    output reg [3:0] an,
    output reg [6:0] sseg
    );


    reg [1:0] state;
    reg [1:0] next;


    always @ (*) begin
        case(state)
            2'b00: next = 2'b01;

            2'b01: next = 2'b10;

            2'b10: next = 2'b11;

            2'b11: next = 2'b00;
        endcase
    end
```

```verilog
    always @ (*) begin

      case(state)

        2'b00: sseg = in0;

        2'b01: sseg = in1;

        2'b10: sseg = in2;

        2'b11: sseg = in3;

      endcase


      case(state)

        2'b00: an = 4'b1110;

        2'b01: an = 4'b1101;

        2'b10: an = 4'b1011;

        2'b11: an = 4'b0111;

      endcase

    end


    always @ (posedge clk or posedge reset) begin

      if (reset)

        state <= 2'b00;

      else

        state <= next;

    end



endmodule
```

timer.v

```verilog
module timer(
```

```verilog
    input clk,

    input reset,

    output reg [3:0] reg_d0, //count for right most digit

    output reg [3:0] reg_d1, //count for 2nd right most digit

    output reg [3:0] reg_d2, //count for 2nd left most digit

    output reg [3:0] reg_d3  //count for left most digit

    );


    reg [8:0] counter = 1'd9999;

    reg startstop; //start, stop or resume signal


    always @ (*) begin

    if (startstop == 1 && reset ==1) //if both stop & reset asserted

        begin

            reg_d0 <= 1'd9; //counter0 is 0

            reg_d1 <= 1'd9; //counter1 is 0

            reg_d2 <= 1'd9; //counter2 is 0

            reg_d3 <= 1'd9; //counter3 is 0

        // if only stop signal is asserted, store the previous count

        // when stop button is pressed again, resume the old count

        end

        else if (startstop == 1)

        begin

            reg_d0 <= reg_d0; //store the old count

            reg_d1 <= reg_d1; //store the old count

            reg_d2 <= reg_d2; //store the old count

            reg_d3 <= reg_d3; //store the old count

        end

        else if (startstop != 1) //if no stop
```

```verilog
    begin
      if(reg_d0 == 9) // if count is xxx9
       begin
         reg_d0 <= 0; //assign count0 to 0
         if (reg_d1 == 9) //if count is xx99
           begin
             reg_d1 <= 0; //assign count1 to 0
             if (reg_d2 == 9) // if count is x999
               begin
                 reg_d2 <= 0; // assign count2 to 0
                 if(reg_d3 == 9) //if count is 9999
                   reg_d3 <= 0; //assign count3 to 0
                 else
                   reg_d3 <= reg_d3 - 1; //else case for count 9999
               end else //else case for count x999
                 reg_d2 <= reg_d2 - 1;
           end else //else case for count xx99
             reg_d1 <= reg_d1 - 1;
       end else // else case for count xxx9
         reg_d0 <= reg_d0 - 1;
     end
   end

endmodule
```