LAB 6 Report

Name: Dhruv Sandesara

UT EID: djs3967

Section: 15295 (Lab Wednesday 12-1)

Checklist:

1. Report

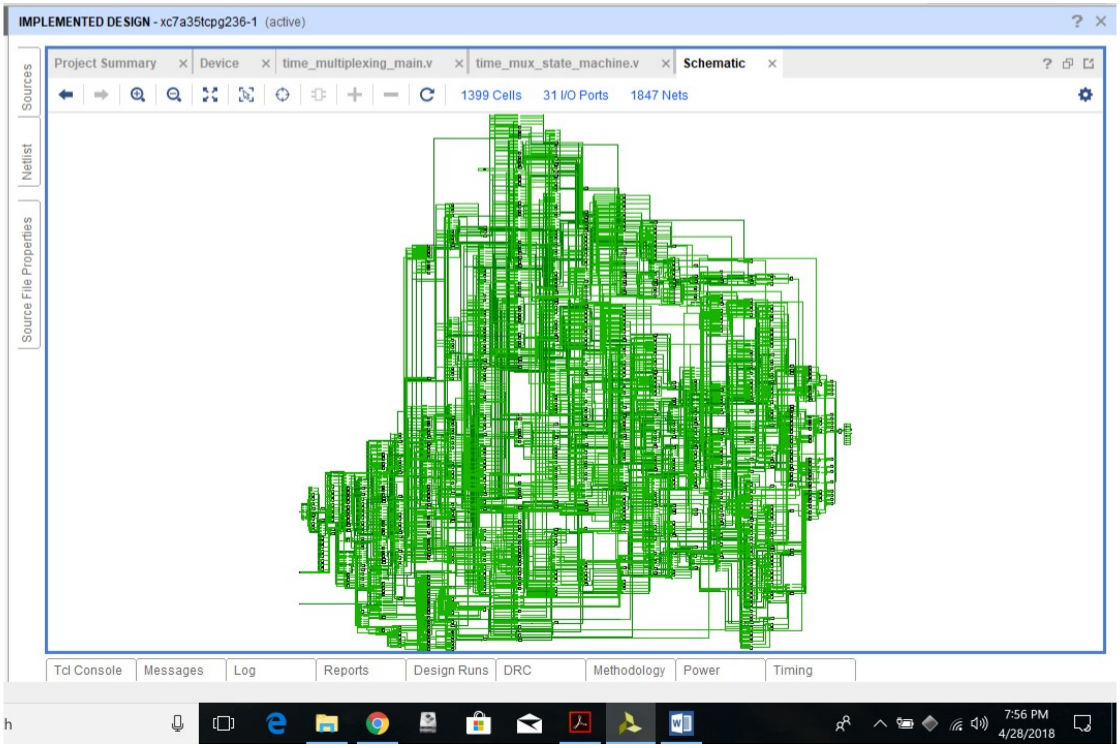
a. Documentation of the design process (in about 400-500 words) – thought process behind the design, how did you make it work, and what were the major problems faced during the design (things which you tried but did not work, if any):

The way I approached this problem was breaking it down into problems and modules. The things that I identified I needed to do were: Display the digits, Get a slower clock to increment and decrement and change the display at a moderate pace, and initializing the count and determining whether to count up or down. For displaying the digits I need a lookup table which displayed the AN outputs depending on the digits that were inputted. Next, I needed to cycle through the different digits which I took care of with implementing an FSM which cycles through the different digits and displays the inputted digit. Most of the code that I got for this from Lab 4 part 3. The only change that I made was to make the clock faster so that I don't see the delays between two digit displays and I see all the digits at the same time instead of them cycling through. Next, I made a case to initialize the counter to the

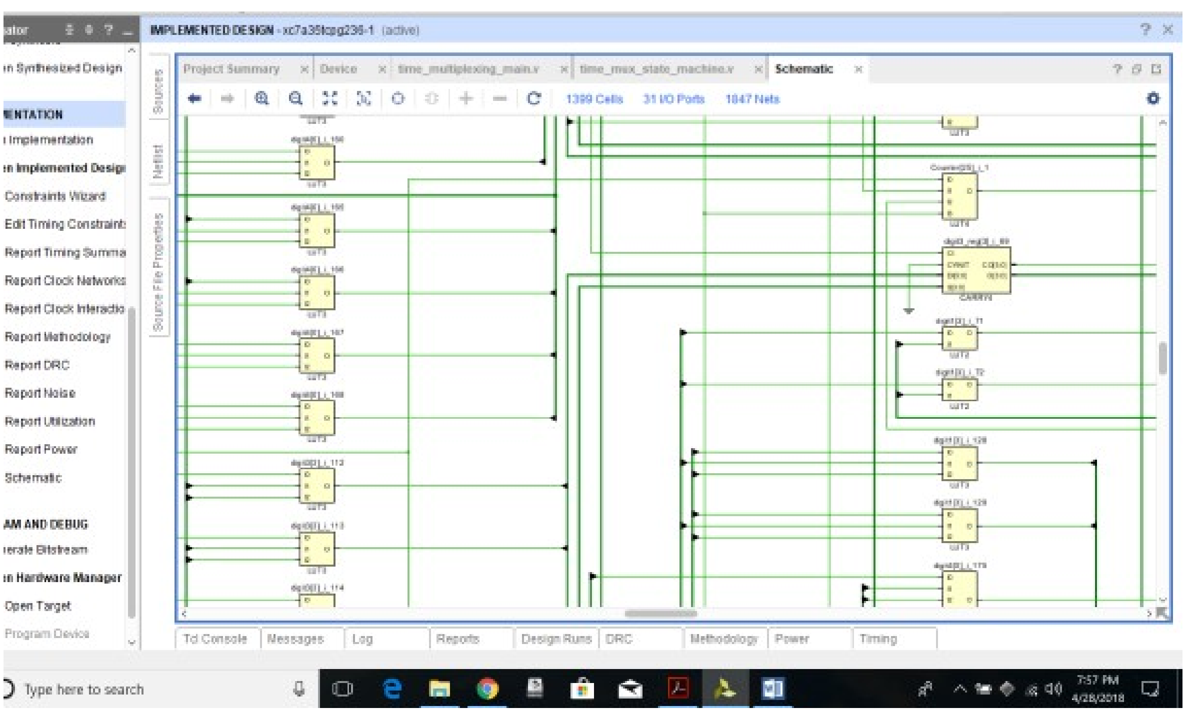
inputs on the sw[7:0]. I do this by first initializing the counter to the 4 most significant bits. Then I multiply it by ten and add the least significant bits. Finally, I multiply it by 100 to get the 4 digits to be displayed on the stopwatch. Now I need to figure out on a case by case basis of the 4 cases of how to handle it. I see for the first 2 case inputs, we need to increment the counter. Also if the input is 00, I override the counter to 0. Similarly, for the next 2 input we count down the counter and for 11, we override with 9900. Also then I implement the reset mode and play mode. I initialize the values only in reset mode so that the changing does not take place midway. Also, I have a running reg which keeps track of the play and pause by signifying 1 or a 0. So implementing all this gave my 90% of the problem solved. The only thing that remained was that I needed to check whether the code was supposed to stop when it reaches the limit. The problems I faced were that when I tried to do a modular approach, Verilog would not allow me to change a variable in different parts of the code which is why I had to implement all of the logic in one giant loop and only call other functions to display the different digits and to get the clock for it.

Word count:421 words.

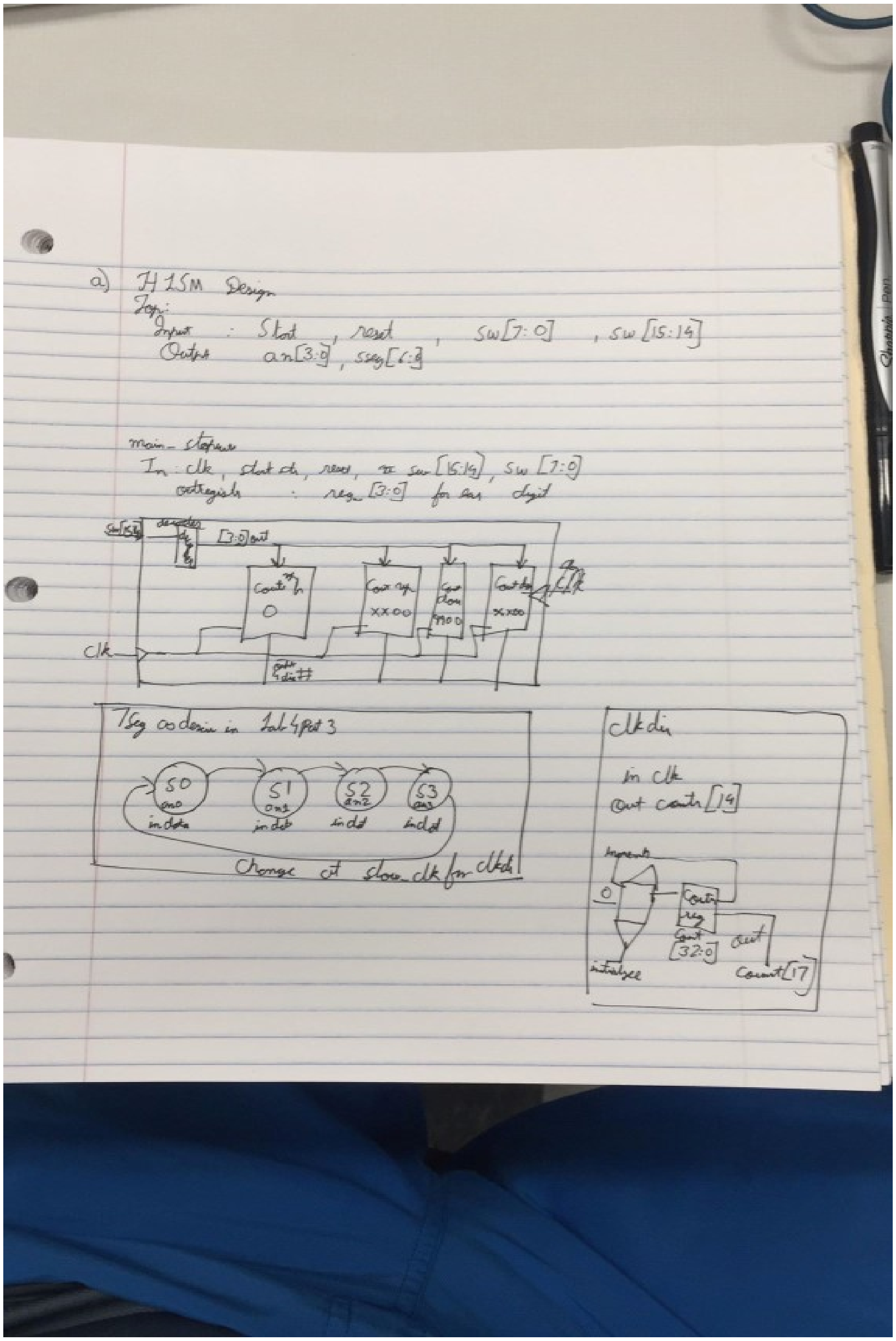
b. Processor architecture with the datapath and controller FSM



zoomed in:



Can explain in detail at checkout as my actual datapath and controller FSM is exactly the same.



c. Verilog codes and constraints

```
`timescale 1ns / 1ps


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,
    output reg dp
    );


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


    always @(*) begin
        case(state)
```

```
        2'b00: next_state = 2'b01;
        2'b01: next_state = 2'b10;
        2'b10: next_state = 2'b11;
        2'b11: next_state = 2'b00;
    endcase
end

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:
        begin
```

```
an = 4'b1110;
dp=1'b1;
end
2'b01:
begin
 an = 4'b1101;
 dp=1'b1;
 end
2'b10:
begin

an = 4'b1011;
dp=1'b0;
end
2'b11:
begin
dp=1'b1;
an = 4'b0111;
 end
endcase
```

```
  end


  always @(posedge clk or posedge reset) begin
    if(reset)
      state <= 2'b00;
    else
      state <= next_state;
  end

endmodule
```

```
`timescale 1ns / 1ps
```

```
module hexto7segment(
   input [3:0] x,
   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'b0001100;
         4'b1010 :   r = 7'b1111111;
         4'b1011 :   r = 7'b1111111;
         4'b1100 :   r = 7'b1111111;
```

```
          4'b1101 :   r = 7'b1111111;
          4'b1110 :   r = 7'b1111111;
          4'b1111 :   r = 7'b1111111;
       endcase
endmodule
```

```
`timescale 1ns / 1ps

module clkdiv(
   input clk,
   input reset,
   output slow_clk
   );
```

```
    reg [25:0] COUNT;



    assign slow_clk = COUNT[14];

    always @(posedge clk) begin



        if(reset)

            COUNT = 0;


        else

            COUNT = COUNT + 1;

        end

endmodule



`timescale 1ns / 1ps
```

```
module time_multiplexing_main(
    input clk,
    input reset,
    input play,
    input [15:0] sw,
    output [3:0] an,
    output [6:0] sseg,
    output dp
    );

    wire [6:0] in0, in1, in2, in3 ;

    wire slow_clk;



reg [25:0] FastCounter;
reg [25:0]Counter;
integer digit1, digit2, digit3, digit4;
```

```
reg running;
//initial running=0;

integer state;
integer increment;

always @(posedge clk) begin

    if(play)begin

    if(running)
       running =0;
    else
       running=1;

       state = sw[15:14];
 end
```

```
if(reset) begin

running=0;

Counter=0;


 case(sw[7:4])

 4'b0000 :   Counter=0;

 4'b0001 :   Counter=1;

 4'b0010 :   Counter=2;

 4'b0011 :   Counter=3;

 4'b0100 :   Counter=4;

 4'b0101 :   Counter=5;

 4'b0110 :   Counter=6;

 4'b0111 :   Counter=7;

 4'b1000 :   Counter=8;

 4'b1001 :   Counter=9;

 4'b1010 :   Counter=9;
```

```
4'b1011 :   Counter=9;

4'b1100 :   Counter=9;

4'b1101 :   Counter=9;

4'b1110 :   Counter=9;

4'b1111 :   Counter=9;

endcase

 case(sw[3:0])

4'b0000 :   Counter=10*Counter+0;

4'b0001 :   Counter=10*Counter+1;

4'b0010 :   Counter=10*Counter+2;

4'b0011 :   Counter=10*Counter+3;

4'b0100 :   Counter=10*Counter+4;

4'b0101 :   Counter=10*Counter+5;

4'b0110 :   Counter=10*Counter+6;

4'b0111 :   Counter=10*Counter+7;

4'b1000 :   Counter=10*Counter+8;

4'b1001 :   Counter=10*Counter+9;

4'b1010 :   Counter=10*Counter+9;

4'b1011 :   Counter=10*Counter+9;

4'b1100 :   Counter=10*Counter+9;
```

```
 4'b1101 :   Counter=10*Counter+9;
 4'b1110 :   Counter=10*Counter+9;
 4'b1111 :   Counter=10*Counter+9;
 endcase




case(sw[15:14])
2'b00 :begin
    increment=1;
   Counter=0;
   end
2'b01 :begin
   increment=1;
   Counter=Counter*100;
   end
2'b10 :begin
   increment=0;
   Counter=9900;
```

```
      end
    2'b11 :begin
      increment=0;
      Counter=Counter*100;
      end
    endcase

     digit1= Counter/1000;
     digit2= (Counter%1000)/100;
     digit3= (Counter%100)/10;
     digit4= (Counter%10)/1;

end
```

```
if(running)begin
   FastCounter= FastCounter+1;
   if(FastCounter>1000000)begin
      if(increment)
        Counter=Counter+1;
      else
        Counter=Counter-1;
     digit1= Counter/1000;
      digit2= (Counter%1000)/100;
      digit3= (Counter%100)/10;
      digit4= (Counter%10)/1;
      FastCounter=0;
```

```
if(((Counter==9999)&&(increment==1))||((Counter==0)&&(increment
==0)))

        running=0;

        end

        end




end



   hexto7segment c1 (.x(digit4), .r(in0));

   hexto7segment c2 (.x(digit3), .r(in1));

   hexto7segment c3 (.x(digit2), .r(in2));

   hexto7segment c4 (.x(digit1), .r(in3));



  clkdiv c5 (.clk(clk), .reset(reset), .slow_clk(slow_clk));
```

```
    time_mux_state_machine c6(
        .clk (slow_clk),
        .reset (reset),
        .in0 (in0),
        .in1 (in1),
        .in2 (in2),
        .in3 (in3),
        .an (an),
        .sseg (sseg),
        .dp(dp));

Endmodule




`timescale 1s / 1ps


module tb_time_multiplexing_main;
reg clk;
```

```
reg reset;

reg play;

reg [7:0] sw;

wire [3:0] an;

wire [6:0] sseg;

wire dp;


time_multiplexing_main uut (

    .clk(clk),

    .reset(reset),

    .play(play),

    .sw(sw),

    .an(an),

    .sseg(sseg),

    .dp(dp)

);


initial begin


clk = 0;
```

```
reset = 0;
sw = 8'b00000000;
play = 0;
reset=0;

#30;

reset = 1;
#30;
reset = 0;

#30;
play = 1;

#100;
play = 1;

end

always
```

```
#5 clk = ~clk;

Endmodule
```

d. Simulation Waveforms of the testbench