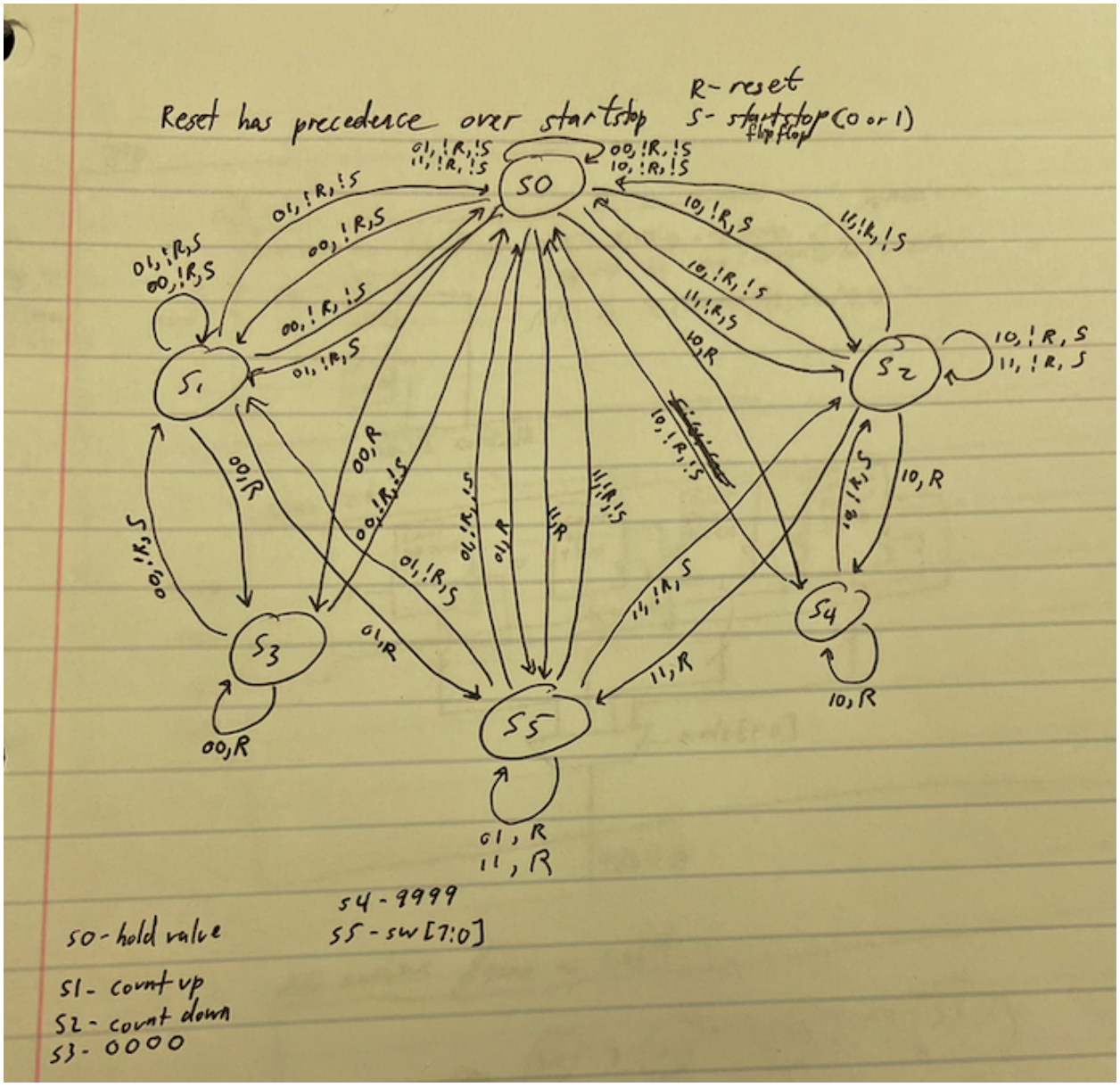
Lab 6 Report

Name: Connie Wang, Harshika Jha

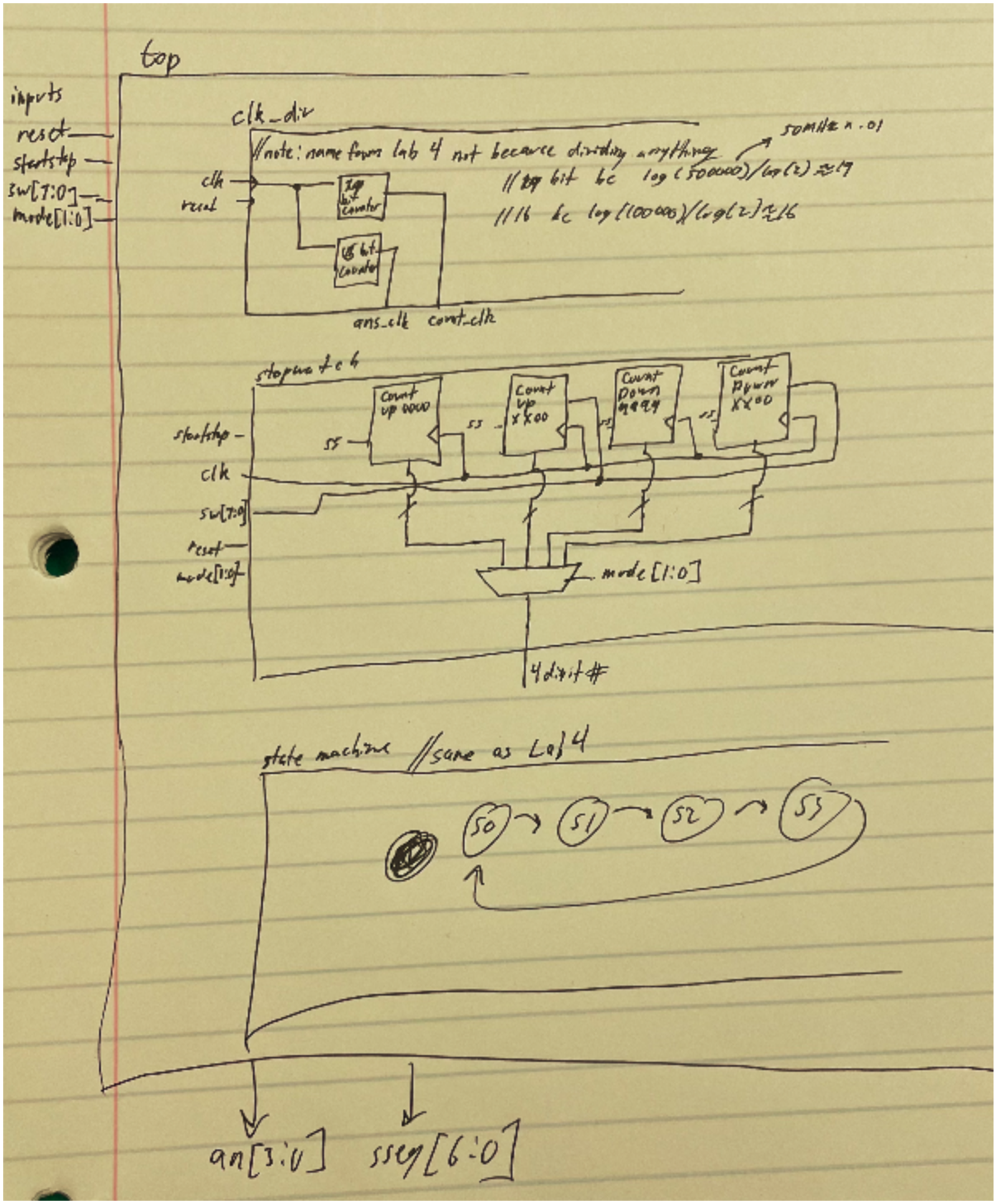
UT EID: cw39276, hj6963

Section: 17745, 17745

Report
a. HLSM describing the system.



b. Processor architecture with the datapath and controller FSM.



c. Verilog, constraint and bitstream files

```
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 ans_clk;
        wire count_clk;
        wire [3:0] d0; //count for right most digit
        wire [3:0] d1; //count for second right most digit
        wire [3:0] d2; //count for second left most digit
        wire [3:0] d3; //count for left most digit

        // Module instantiation of hexto7segment decoder
        hexto7segment c1 (.x(d0), .r(in0));
        hexto7segment c2 (.x(d1), .r(in1));
        hexto7segment c3 (.x(d2), .r(in2));
        hexto7segment c4 (.x(d3), .r(in3));

        // Module instantiation of clock divider
        // same functionality as the clk_div before, but may have a different
width requirement
        clk_div clock (.clk(clk), .reset(reset), .ans_clk(ans_clk),
.count_clk(count_clk));


        stopwatch c5 (
        .clk(count_clk),
        .startstop(startstop),
        .reset(reset),
        .mode(mode[1:0]),
        .sw(sw[7:0]),
        .d0(d0),
```

```
        .d1(d1),
        .d2(d2),
        .d3(d3));


        // Module instantiation of the multiplexer
        //replace slow_clk with clk for simulation, and vice versa
        time_mux_state_machine c6 (
        .clk(ans_clk),
        .reset(reset),
        .in0(in0),
        .in1(in1),
        .in2(in2),
        .in3(in3),
        .an(an),
        .sseg(sseg));



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'b0000100;
        4'b1010: r = 7'b0001000;
```

```
        4'b1011: r = 7'b1100000;
        4'b1100: r = 7'b0110001;
        4'b1101: r = 7'b1000010;
        4'b1110: r = 7'b0110000;
        4'b1111: r = 7'b0111000;
        endcase
endmodule


module clk_div(
        input clk,
        input reset,
        output ans_clk,
        output count_clk
        );

        reg [16:0] ans;
        reg [19:0] count;
        reg a;
        reg c;
        assign ans_clk = a;
        assign count_clk = c;


        always @(posedge clk) begin
        if (count < 500000) begin
        count <= count + 1;
        end
        else begin
        c <= ~c;
        count <= 0;
        end
        end
        always @(posedge clk) begin
        if (ans < 100000) begin
        ans <= ans + 1;
```

```
        end
        else begin
        a <= ~a;
        ans <= 0;
        end
        end
Endmodule


module stopwatch(
        input clk,
        input startstop,
        input reset,
        input [1:0] mode,
        input [7:0] sw,
        output reg [3:0] d0,
        output reg [3:0] d1,
        output reg [3:0] d2,
        output reg [3:0] d3
        );

        reg hold;
        reg ss = 1;

always @ (posedge clk) begin
        hold <= startstop;
        if( hold && !startstop)
        ss <= ~ss;
end

always @ (posedge clk) begin

//Mode 1
        if (mode == 2'b00) begin
        if (reset == 1)
                begin
```

```
        d0 <= 0;
        d1 <= 0;
        d2 <= 0;
        d3 <= 0;
        end

else if (ss == 1 && reset != 0)
        begin
        //store
        d0 <= d0;
        d1 <= d1;
        d2 <= d2;
        d3 <= d3;
        end

else if (ss != 1)
        begin
        if(d0 == 9) begin
        d0 <= 0;
        if (d1 == 9) begin
                d1 <= 0;
                if (d2 == 9) begin
                        d2 <= 0;
                        if(d3 == 9)begin
                        d3 <= 0;
                        d2 <= 0;
                        d1 <= 0;
                        d0 <= 0;
                        end else
                        d3 <= d3 + 1;
                end else
                        d2 <= d2 + 1;
                end else
                d1 <= d1 + 1;
        end else
        d0 <= d0 + 1;
```

```
            end
        end

// Mode 2
        if (mode == 2'b01) begin
        if (reset ==1)
            begin
            d0 <= 0;
            d1 <= 0;
            d2 <= sw[3:0];
            d3 <= sw[7:4];
            end

        else if (ss == 1 && reset != 0)
            begin
            d0 <= d0;
            d1 <= d1;
            d2 <= d2;
            d3 <= d3;
            end

            else if (ss != 1)
            begin
            if(d0 == 9) begin
            d0 <= 0;
            if (d1 == 9) begin
                    d1 <= 0;
                    if (d2 == 9) begin
                            d2 <= 0;
                            if(d3 == 9)begin
                            d3 <= 0;
                            d2 <= 0;
                            d1 <= 0;
                            d0 <= 0;
                            end else
                            d3 <= d3 + 1;
```

```
                end else
                        d2 <= d2 + 1;
                end else
                d1 <= d1 + 1;
        end else
        d0 <= d0 + 1;
        end

    end


// Mode 3
    if (mode == 2'b10) begin
    if (reset ==1)
    begin
    d0 <= 9;
    d1 <= 9;
    d2 <= 9;
    d3 <= 9;
    end

    else if (ss == 1 && reset != 0)
    begin
    d0 <= d0;
    d1 <= d1;
    d2 <= d2;
    d3 <= d3;
    end

    else if (ss != 1) begin

    if(d0 == 0) begin
        d0 <= 9;
        if (d1 == 0) begin
        d1 <= 9;
        if (d2 == 0) begin
```

```
                d2 <= 9;
                if(d3 == 0) begin
                        d0 <= 9;
                        d1 <= 9;
                        d2 <= 9;
                        d3 <= 9;
                end else
                        d3 <= d3 - 1;
                end else
                d2 <= d2 - 1;
        end else
        d1 <= d1 - 1;
end else
d0 <= d0 - 1;
end
end

//Mode 4
        if (mode == 2'b11) begin
        if (reset == 1 ) begin
        d0 <= 0;
        d1 <= 0;
        d2 <= sw[3:0];
        d3 <= sw[7:4];
        end
        else if (ss == 1) begin
        d0 <= d0;
        d1 <= d1;
        d2 <= d2;
        d3 <= d3;
        end
        else if (ss != 1) begin
        if(d0 == 0) begin
                d0 <= 9;
                if (d1 == 0) begin
                d1 <= 9;
```

```
                if (d2 == 0) begin
                        d2 <= 9;
                        if(d3 == 0) begin
                                d0 <= 9;
                                d1 <= 9;
                                d2 <= 9;
                                d3 <= 9;
                        end else
                                d3 <= d3 - 1;
                        end else
                        d2 <= d2 - 1;
                end else
                d1 <= d1 - 1;
        end else
        d0 <= d0 - 1;
        end
 end


end

endmodule


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_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
end
always@ (*) begin
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_state;
end
```

```
## Clock signal - Uncomment if needed (will be used in future labs)
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]

        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 W16 [get_ports {sw[0]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
        set_property PACKAGE_PIN W17 [get_ports {sw[1]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
        set_property PACKAGE_PIN W15 [get_ports {sw[2]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
        set_property PACKAGE_PIN V15 [get_ports {sw[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
        set_property PACKAGE_PIN W14 [get_ports {sw[4]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
        set_property PACKAGE_PIN W13 [get_ports {sw[5]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
        set_property PACKAGE_PIN V2 [get_ports {sw[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
        set_property PACKAGE_PIN T3 [get_ports {sw[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
##7 segment display
        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]}]
```