11/14/20

CPE 166 Advanced Logic Design

LAB #4

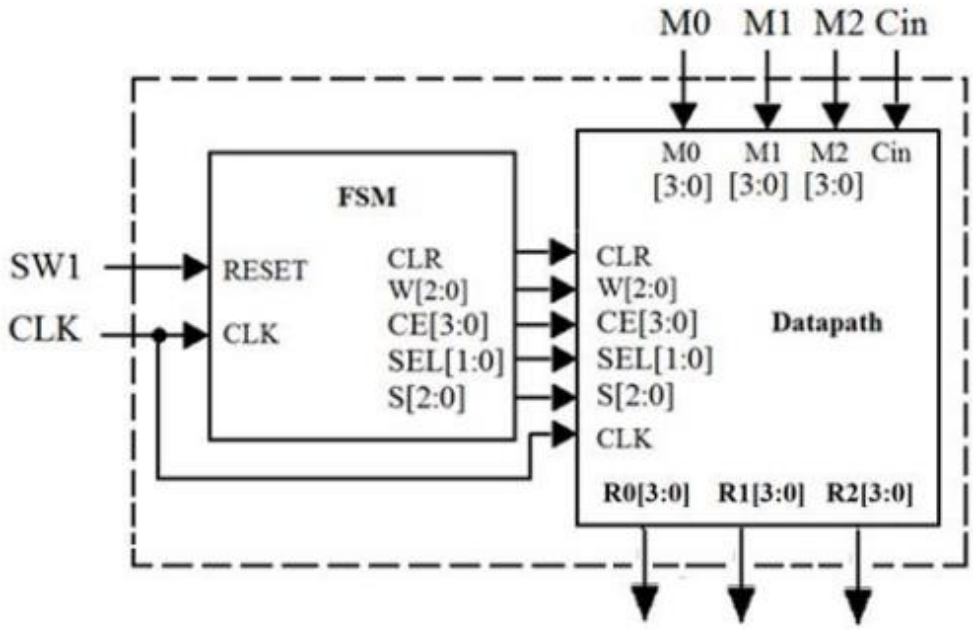
XAVIER HOWELL

Table of contents

Introduction

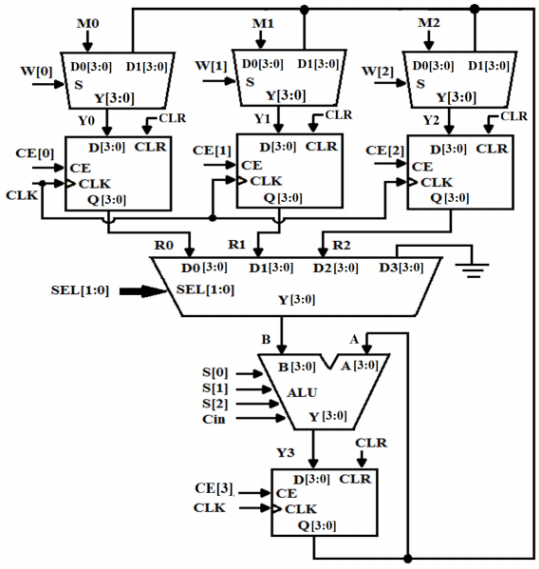
The purpose of parts 2, 3, and part 4 of Lab 4 is to design s simplified microprocessor. The figure below shows a the microprocessor diagram with M0, M1, M2 and Cin inputs, one input SW1 and also clock input. The SW1 serves as asynchronous reset function.

$$R2 = M0 + (\text{not } M1) + Cin$$



Report

4.2 Design:



Part two design is of the microprocessor data path. It includes a mux 2 to 1, a mux 4 to 1, alu, and a dffa. I built the hierarchical of this design without simulation of the sub design because I know the parts work.

4.2 VHDL

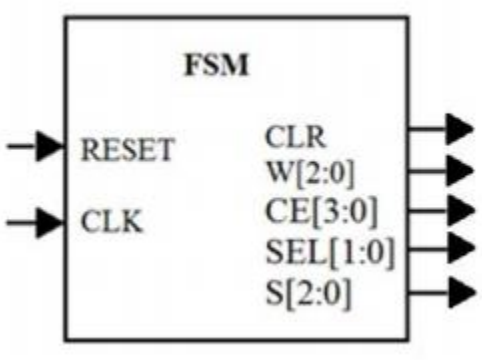
```
module mux2_1(d0, d1, s, y);
input          [3:0] d0, d1;
input          s;
output         [3:0] y;

reg            [3:0] y;

always@( s or d1 or d0 )
begin
   if (s)
          y = d1;
   else
          y = d0;
end

endmodule
```

```
module dffa (d, clr, ce, clk, q);
input          [3:0] d;
input          clr, clk, ce;
output         [3:0] q;

reg            [3:0] q;

always@(posedge clr or  posedge clk)
begin
    if(clr) q <= 0;
    else if (ce)
       q <= d;
end

endmodule
```

```
module mux4_1(d0, d1, d2, d3, sel, y);
    input          [3:0] d3, d2, d1, d0;
    input          [1:0] sel;
    output         [3:0] y;

    reg            [3:0] y;

    always@( sel or d3 or d2 or d1 or d0 )
    begin
        case (sel)
            0 : y = d0;
            1 : y = d1;
            2 : y = d2;
            3 : y = d3;
            //default : y = d0;
        endcase
    end
endmodule
```

```
module alu(b, a, s, cin, y);
    input          [3:0] b, a;
    input          cin;
    input          [2:0] s;
    output         [3:0] y;

    reg            [3:0] y;

    always@( s or a or b or cin )
    begin
        case (s)
            0 : y = a + b + cin;
            1 : y = a + ~b + cin;
            2 : y = b;
            3 : y = a;
            4 : y = (a & b);
            5 : y = (a | b);
            6 : y = ~a;
            7 : y = (a ^ b);
        endcase
    end
endmodule
```

4.3 Design: Part three is the finite state machine diagram for the “FSM” unit. The FSM is used to implement the control path for the microprocessor. The control path block diagram is shown below.



4.3 VHDL

```
module fsm(clk, rst, clr, sel, w, s, ce);

input clk, rst;
output clr;
output [1:0] sel;
output [2:0] w, s;
output [3:0] ce;

reg clr;
reg [1:0] sel;
reg [2:0] w, s;
reg [3:0] ce;

parameter s0 = 0, s1 = 1, s2 = 2, s3 = 3, s4 = 4, s5 = 5;

reg [2:0] cs, ns;

always@(posedge clk or posedge rst) begin
   if( rst ) begin
      cs <= s0;
   end else begin
      cs <= ns;
   end
end
   always@(cs) begin
   case(cs)
      s0 : begin
         clr = 1'b1;
         w = 3'b100;
         s = 3'b010;
         sel = 2'b00;
         ce = 4'b0000;
```

```
            ns <= s1;
        end
        s1 : begin
            clr = 1'b0;
            w = 3'b100;
            s = 3'b010;
            sel = 2'b00;
            ce = 4'b1111;
            ns <= s2;
        end
        s2 : begin
            clr = 1'b0;
            w = 3'b100;
            s = 3'b010;
            sel = 2'b00;
            ce = 4'b1111;
            ns <= s3;
        end
        s3 : begin
            clr = 1'b0;
            w = 3'b100;
            s = 3'b001;
            sel = 2'b01;
            ce = 4'b1111;
            ns <= s4;
        end
        s4 : begin
            clr = 1'b0;
            w = 3'b100;
            s = 3'b001;
            sel = 2'b01;
            ce = 4'b0111;
            ns <= s5;
        end
        s5 : begin
            clr = 1'b0;
            w = 3'b100;
            s = 3'b001;
            sel = 2'b01;
            ce = 4'b0111;
            ns <= s5;
        end
        //default : ns = s0;
    endcase
    end
endmodule
```

4.4 Design: This part is to complete the design of the top module for the simplified 4-bit microprocessor.

4.4 VHDL

```
module top(rst, clk, m0, m1, m2, cin, r0, r1, r2);

input clk, cin, rst;
input [3:0] m0, m1, m2;
output [3:0] r0, r1, r2;

wire clr;
wire [1:0] sel;
wire [2:0] s, w;
wire [3:0] ce;

fsm fsm1(.clk(clk), .rst(rst), .clr(clr), .sel(sel), .w(w), .s(s), .ce(ce));

datapath datapath1(.clr(clr), .w(w), .ce(ce), .sel(sel), .s(s), .clk(clk), .m0(m0), .m1(m1), .m2(m2), .cin(cin), .r0(r0), .r1(r1), .r2(r2));

endmodule
```

4.4 Testbench

```
module top_tb();

reg clk, rst, cin;
reg [3:0] m0, m1, m2;
wire [3:0] r0, r1, r2;

top uut(.rst(rst), .clk(clk), .m0(m0), .m1(m1), .m2(m2), .cin(cin), .r2(r2), .r0(r0), .r1(r1));

always
begin
#5 clk = ~clk;
end

initial
begin
clk = 0; rst = 1; cin = 1; m0=0; m1=0; m2=0;

 #10 m0 = 3; rst = 0;
    m1 = 13;
#150

$stop;
end
endmodule
```

Conclusion

In conclusion the purpose of this lab was to design a simple microprocessor that implemented the logic equation R2 = M0 +not (M1) + Cin. We used a hierarchal design method which made the lab easier because we have already previously mode the modules used in this design earlier.