

System Synthesis and Modeling (CSCE 3953)

Final Project (Step 4)

Zack Fravel

10/6/16

zpfravel@uark.edu

Report

Step four of the final project had us design a number of finite state machines that are used to generate the control signals along our datapath. With my designs, I broke up the functionality of the CPU into seven different FSMs. I have one FSM for the register-based operations, one for instruction fetch, and dedicated FSMs for Mov, Movi, Load, Store, and Immediate ALU operations.

With the Immediate FSM's, I have the decoder pass in the value that needs to be send to the ALU and the FSM is responsible for driving the bus and sending these values. Therefore, I have a tri-state output on my Immediate FSM as well as the MovImm FSM. To simulate MFC, I just assumed MFC would be set to high so the whole simulation can be seen clearly at once. All of the FSM's send out a 'finish' signal that will be used to trigger the instruction fetch FSM at the end of each operation. The instruction fetch FSM is also triggered by reset. It would take a huge amount of pictures to include all of the code for these FSMs so I included the top module declarations with the ports and the output logic for each.

Instruction Fetch FSM

```

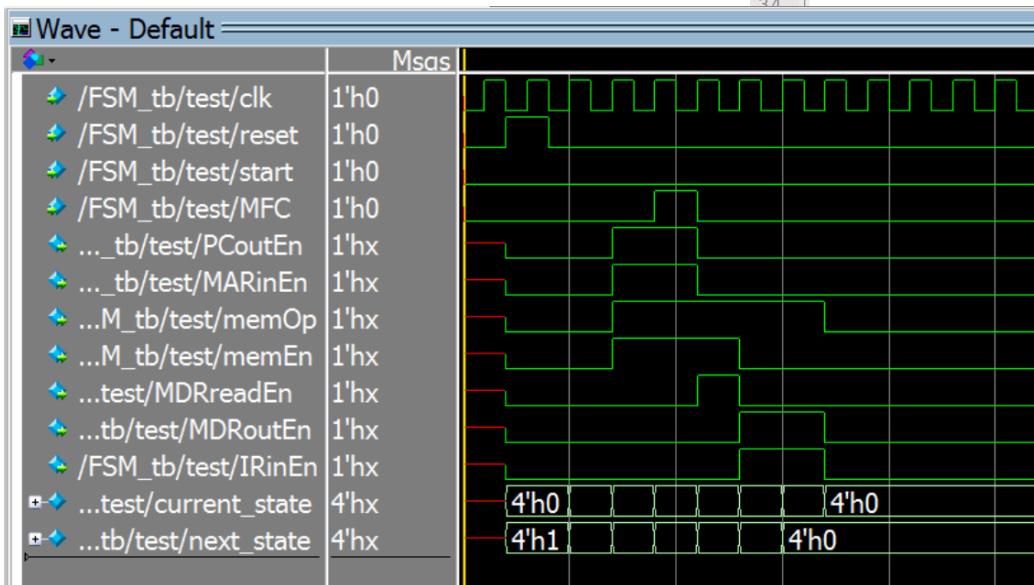
1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 module fetch_FSM(clk, reset, start, PCoutEn, MARinEn, memOp, memEn, MDRreadEn, MDRoutEn, IRinEn, MFC);
6
7 // Port Declaration
8
9 input clk; input reset; input start; input MFC;
10
11 output reg PCoutEn; output reg MARinEn; output reg memOp; output reg memEn;
12 output reg MDRreadEn; output reg MDRoutEn; output reg IRinEn;
13
14 reg[3:0] current_state, next_state;
15
16
17 always @(current_state) begin
18   case (current_state)
19     init: begin
20       PCoutEn = 0; MARinEn = 0; memOp = 0; memEn = 0;
21       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
22     end
23     one: begin
24       PCoutEn = 0; MARinEn = 0; memOp = 0; memEn = 0;
25       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
26     end
27     two: begin
28       PCoutEn = 1; MARinEn = 1; memOp = 1; memEn = 1;
29       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
30     end
31     three: begin
32       PCoutEn = 1; MARinEn = 1; memOp = 1; memEn = 1;
33       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
34     end
35     four: begin
36       PCoutEn = 0; MARinEn = 0; memOp = 1; memEn = 1;
37       MDRreadEn = 1; MDRoutEn = 0; IRinEn = 0;
38     end
39     five: begin
40       PCoutEn = 0; MARinEn = 0; memOp = 1; memEn = 0;
41       MDRreadEn = 0; MDRoutEn = 1; IRinEn = 1;
42     end
43     six: begin
44       PCoutEn = 0; MARinEn = 0; memOp = 1; memEn = 0;
45       MDRreadEn = 0; MDRoutEn = 1; IRinEn = 1;
46     end
47   endcase
48   next_state = current_state;
49   // Output Logic
50   case (current_state)
51     init: begin
52       PCoutEn = 0; MARinEn = 0; memOp = 0; memEn = 0;
53       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
54     end
55     one: begin
56       PCoutEn = 0; MARinEn = 0; memOp = 0; memEn = 0;
57       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
58     end
59     two: begin
60       PCoutEn = 1; MARinEn = 1; memOp = 1; memEn = 1;
61       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
62     end
63     three: begin
64       PCoutEn = 1; MARinEn = 1; memOp = 1; memEn = 1;
65       MDRreadEn = 0; MDRoutEn = 0; IRinEn = 0;
66     end
67     four: begin
68       PCoutEn = 0; MARinEn = 0; memOp = 1; memEn = 1;
69       MDRreadEn = 1; MDRoutEn = 0; IRinEn = 0;
70     end
71     five: begin
72       PCoutEn = 0; MARinEn = 0; memOp = 1; memEn = 0;
73       MDRreadEn = 0; MDRoutEn = 1; IRinEn = 1;
74     end
75     six: begin
76       PCoutEn = 0; MARinEn = 0; memOp = 1; memEn = 0;
77       MDRreadEn = 0; MDRoutEn = 1; IRinEn = 1;
78     end
79   endcase
80 end
81
82
83
84
85
86
87
88
89
90
91
92
93

```

```

1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 `timescale 1ns/1ns
6 module FSM_tb;
7
8 // Inputs
9 reg CLK; reg RESET; reg START; reg MFC;
10
11 // Outputs
12 wire PCoutEn; wire MARinEn; wire memOp; wire memEn;
13 wire MDRreadEn; wire MDRoutEn; wire IRinEn;
14
15 // Instantiate Module
16 fetch_FSM test (CLK, RESET, START, PCoutEn);
17
18 initial begin
19   CLK = 0;
20   RESET = 0;
21   START = 0;
22   MFC = 0;
23   #20;
24   RESET = 1;
25   #20;
26   RESET = 0;
27   #50;
28   MFC = 1;
29   #20;
30   MFC = 0;
31 end
32
33 always #10 CLK = ~CLK;
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

```

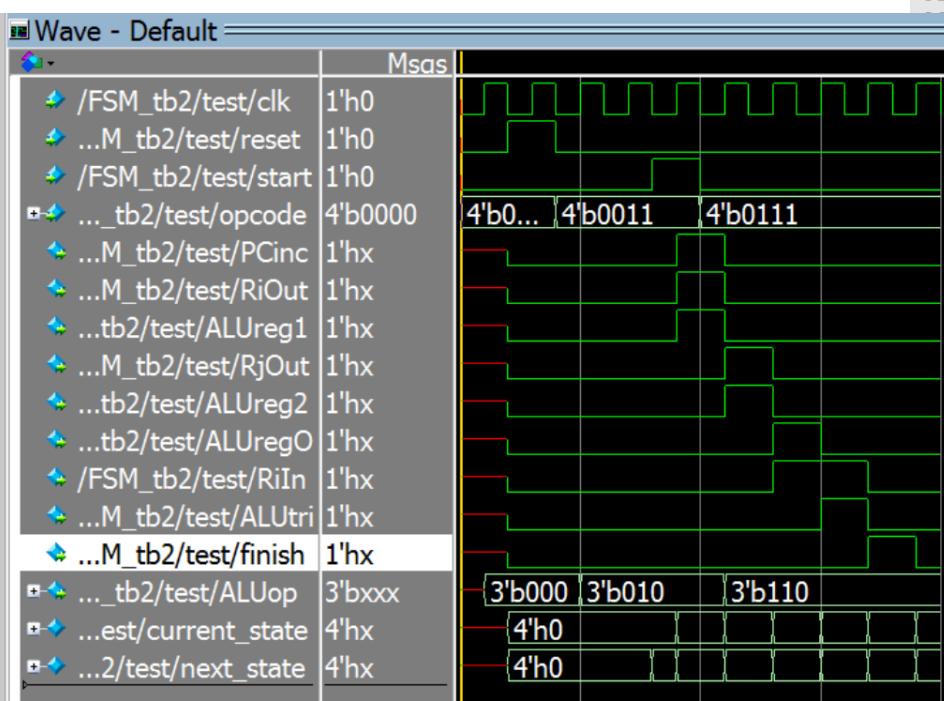


Register FSM

```

1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 module Register_FSM(clk, reset, start, PCinc, RiOut, ALUreg1, RjOut, ALUreg2, ALUreg0, RiIn, ALUtri, finish, opcode, ALUop);
6
7 // Port Declaration
8
9 input clk; input reset; input start; input[3:0] opcode;
10
11 output reg PCinc; output reg RiOut; output reg ALUreg1; output reg RjOut;
12 output reg ALUreg2; output reg ALUreg0; output reg RiIn; output reg ALUtri; output reg finish;
13 output reg[2:0] ALUop;
14
15 reg[3:0] current_state, next_state;
16
.
.
.
58 always @(current_state) begin
59   case (current_state)
60     init: begin
61       PCinc = 0; RiOut = 0; RjOut = 0; RiIn = 0;
62       ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0; ALUtri = 0; finish = 0;
63     end
64     one: begin
65       PCinc = 1; RiOut = 1; RjOut = 0; RiIn = 0;
66       ALUreg1 = 1; ALUreg2 = 0; ALUreg0 = 0; ALUtri = 0; finish = 0;
67     end
68     two: begin
69       PCinc = 0; RiOut = 0; RjOut = 1; RiIn = 0;
70       ALUreg1 = 0; ALUreg2 = 1; ALUreg0 = 0; ALUtri = 0; finish = 0;
71     end
72     three: begin
73       PCinc = 0; RiOut = 0; RjOut = 0; RiIn = 1;
74       ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 1; ALUtri = 0; finish = 0;
75     end
76     four: begin
77       PCinc = 0; RiOut = 0; RjOut = 0; RiIn = 1;
78       ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0; ALUtri = 1; finish = 0;
79     end
80     five: begin
81       PCinc = 0; RiOut = 0; RjOut = 0; RiIn = 0;
82       ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0; ALUtri = 0; finish = 1;
83     end
84     six: begin
85       PCinc = 0; RiOut = 0; RjOut = 0; RiIn = 0;
86       ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0; ALUtri = 0; finish = 0;
87     end
88   end
89 end
90
91
.
.
.
5 `timescale 1ns/1ns
6 module FSM_tb2;
7
8 // Inputs
9 reg CLK; reg RESET; reg START; reg[3:0] opcode;
10
11 // Outputs
12 wire PCinc; wire ALUreg1; wire ALUreg2; wire ALUreg0; wire RiIn; wire RiOut; wire RjOut; wire ALUtri; wire finish; wire [2:0] ALUop;
13
14 // Instantiate Module
15 Register_FSM test (CLK, RESET, START, opcode, ALUop);
16
17 initial begin
18   CLK = 0;
19   RESET = 0;
20   START = 0;
21   OPCODE = 4'b0000;
22   #20;
23   RESET = 1;
24   #20;
25   RESET = 0;
26   OPCODE = 4'b0011;
27   #40;
28   START = 1;
29   #20;
30   START = 0;
31   OPCODE = 4'b0111;
32
33 always #10 CLK = ~CLK;
34
35 endmodule

```



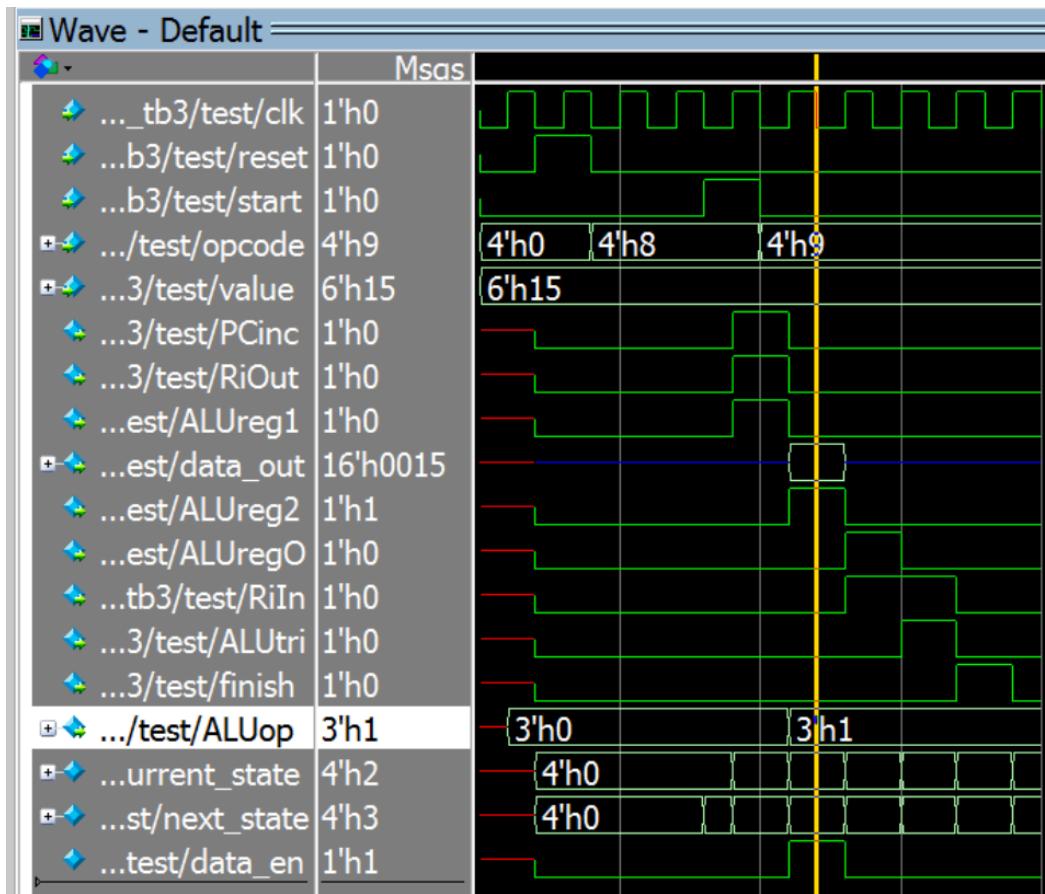
ADD/SUB Imm FSM

```
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 module Immediate_FSM(clk, reset, start, value, PCinc, RiOut, ALUreg1, data_out, ALUreg2, ALUreg0, RiIn, ALUTri, finish, opcode, ALUop);
6
7 // Port Declaration
8
9 input clk; input reset; input start; input[3:0] opcode; input[5:0] value;
10
11 output reg PCinc; output reg RiOut; output reg ALUreg1; output[15:0] data_out;
12 output reg ALUreg2; output reg ALUreg0; output reg RiIn; output reg ALUTri; output reg finish;
13 output reg[2:0] ALUop;
14
15
16 always @(current_state) // Output Logic
17 begin
18
19     case (current_state)
20
21         init: begin
22             PCinc = 0; RiOut = 0; ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0;
23             RiIn = 0; ALUTri = 0; finish = 0; data_en = 0;
24         end
25         one: begin
26             PCinc = 1; RiOut = 1; ALUreg1 = 1; ALUreg2 = 0; ALUreg0 = 0;
27             RiIn = 0; ALUTri = 0; finish = 0; data_en = 0;
28         end
29         two: begin
30             PCinc = 0; RiOut = 0; ALUreg1 = 0; ALUreg2 = 1; ALUreg0 = 0;
31             RiIn = 0; ALUTri = 0; finish = 0; data_en = 1;
32         end
33         three: begin
34             PCinc = 0; RiOut = 0; ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 1;
35             RiIn = 1; ALUTri = 0; finish = 0; data_en = 0;
36         end
37         four: begin
38             PCinc = 0; RiOut = 0; ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0;
39             RiIn = 1; ALUTri = 1; finish = 0; data_en = 0;
40         end
41         five: begin
42             PCinc = 0; RiOut = 0; ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0;
43             RiIn = 0; ALUTri = 0; finish = 1; data_en = 0;
44         end
45         six: begin
46             PCinc = 0; RiOut = 0; ALUreg1 = 0; ALUreg2 = 0; ALUreg0 = 0;
47             RiIn = 0; ALUTri = 0; finish = 0; data_en = 0;
48         end
49
50     endcase
51
52 end
53
54
55 always@ (posedge clk) // Convert Opcode for ALU
56 begin
57
58     case (opcode)
59         4'b1000: begin ALUop = 3'b000; end      //Addi
60         4'b1001: begin ALUop = 3'b001; end      //Subi
61         default: begin ALUop = 3'b000; end
62     endcase
63
64 end
65
66 assign data_out = (data_en) ? value:16'hzzzz;
67
68
69 endmodule
```

```

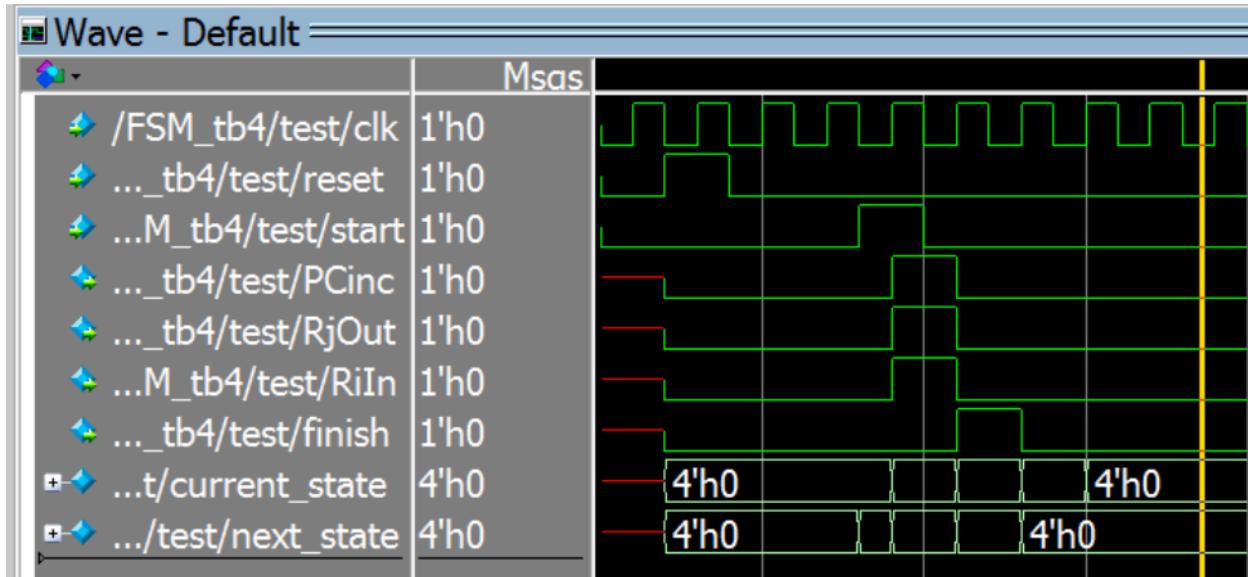
1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 `timescale 1ns/1ns
6 module FSM_tb3;
7
8 // Inputs
9     reg CLK; reg RESET; reg START; reg[3:0] OPCODE; reg[5:0] value;
10 // Outputs
11 wire PCinc; wire ALUreg1; wire ALUreg2; wire ALUregO;
12 wire RiIn; wire RiOut; wire RjOut; wire finish; wire ALUtri; wire[2:0] ALUop; wire[15:0] data_out;
13
14 // Instantiate Module
15 Immediate_FSM test (CLK, RESET, START, value, PCinc, RiOut, ALUreg1, data_out, ALUreg2, ALUregO, RiIn, ALUtri, fin
16
17 initial
18 begin
19     CLK = 0;
20     RESET = 0;
21     START = 0;
22     OPCODE = 4'b0000;
23     value = 6'b010101;
24     #20;
25     RESET = 1;
26     #20;
27     RESET = 0;
28     OPCODE = 4'b1000;
29     #40;
30     START = 1;
31     #20;
32     START = 0;
33     OPCODE = 4'b1001;
34 end
35
36 always #10 CLK = ~CLK;
37
38 endmodule
39

```



Move FSM

```
1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 module move_FSM(clk, reset, start, PCinc, RjOut, RiIn, finish);
6
7 // Port Declaration
8
9 input clk; input reset; input start;
10
11 output reg PCinc; output reg RjOut; output reg RiIn; output reg finish;
12
13 reg[3:0] current_state, next_state;
14
15
52 always @(current_state) // Output Logic
53 begin
54
55 case (current_state)
56
57 init: begin
58   PCinc = 0; RjOut = 0; RiIn = 0; finish = 0;
59 end
60
61 one: begin
62   PCinc = 1; RjOut = 1; RiIn = 1; finish = 0;
63 end
64
65 two: begin
66   PCinc = 0; RjOut = 0; RiIn = 0; finish = 1;
67 end
68
69 three: begin
70   PCinc = 0; RjOut = 0; RiIn = 0; finish = 0;
71 end
72
73 endcase
74
75
76 // Zack Fravel
77 // System Synthesis and Modeling
78 // Final Project (Step 4)
79
80 `timescale 1ns/1ns
81 module FSM_tb4;
82
83 // Inputs
84   reg CLK; reg RESET; reg START; reg[5:0] value;
85 // Outputs
86   wire PCinc;
87   wire RiIn; wire[15:0] data_out; wire finish;
88
89 // Instantiate Module
90   moveImm_FSM test (CLK, RESET, START, value, PCinc, data_out, RiIn, finish);
91
92 initial
93 begin
94   CLK = 0;
95   RESET = 0;
96   START = 0;
97   value = 6'b001101;
98   #20;
99   RESET = 1;
100  #20;
101  RESET = 0;
102  #40;
103  START = 1;
104  #20;
105  START = 0;
106 end
107
108 always #10 CLK = ~CLK;
109
110 endmodule
111
```

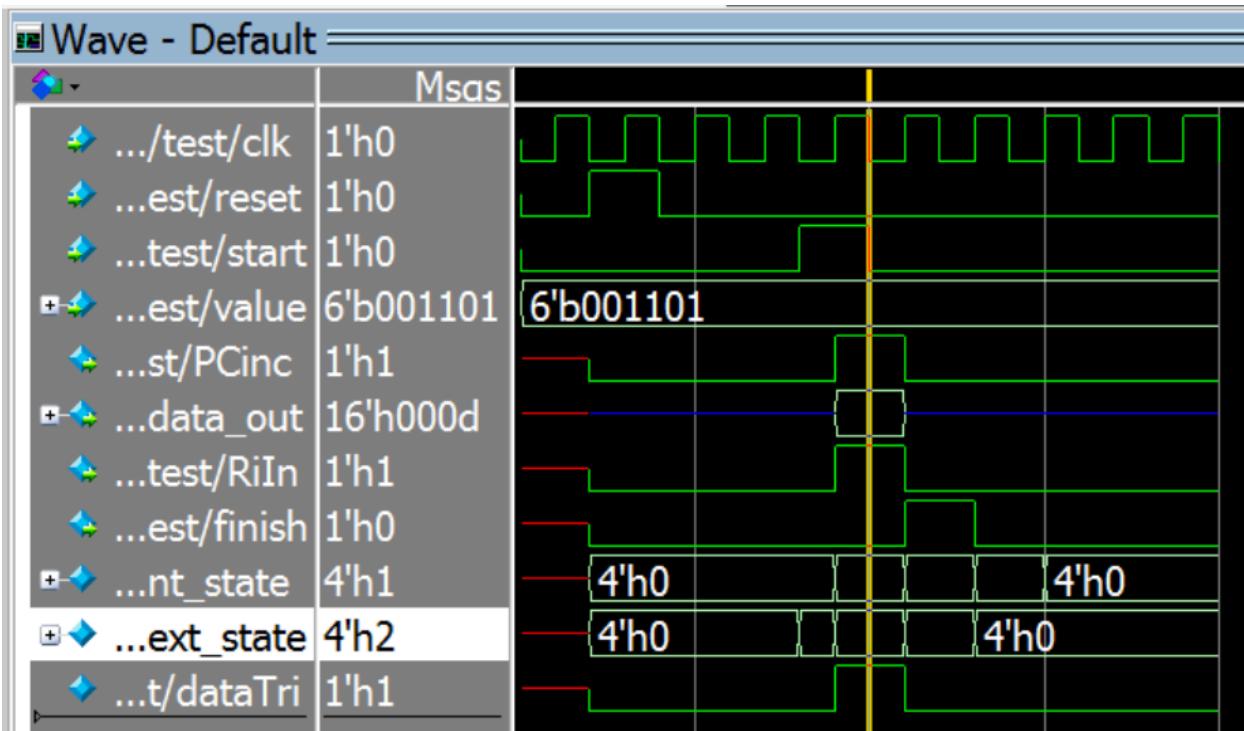


Move Imm FSM (similar tb)

```

Ln# 1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 module moveImm_FSM(clk, reset, start, value, PCinc, data_out, RiIn, finish);
6
7 // Port Declaration
8
9 input clk; input reset; input start; input[5:0] value;
10
11 output reg PCinc; output[15:0] data_out; output reg RiIn; output reg finish;
12
13 reg[3:0] current_state, next_state;
14 reg dataTri;
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54 always @(current_state)                                // Output Logic
55 begin
56
57 case (current_state)
58
59 init: begin
60     PCinc = 0; dataTri = 0; RiIn = 0; finish = 0;
61 end
62 one: begin
63     PCinc = 1; dataTri = 1; RiIn = 1; finish = 0;
64 end
65 two: begin
66     PCinc = 0; dataTri = 0; RiIn = 0; finish = 1;
67 end
68 three: begin
69     PCinc = 0; dataTri = 0; RiIn = 0; finish = 0;
70 end
71 endcase
72
73 end
74
75 assign data_out = (dataTri) ? value:16'hzzzz;
76
77
78 endmodule

```



Load FSM

```

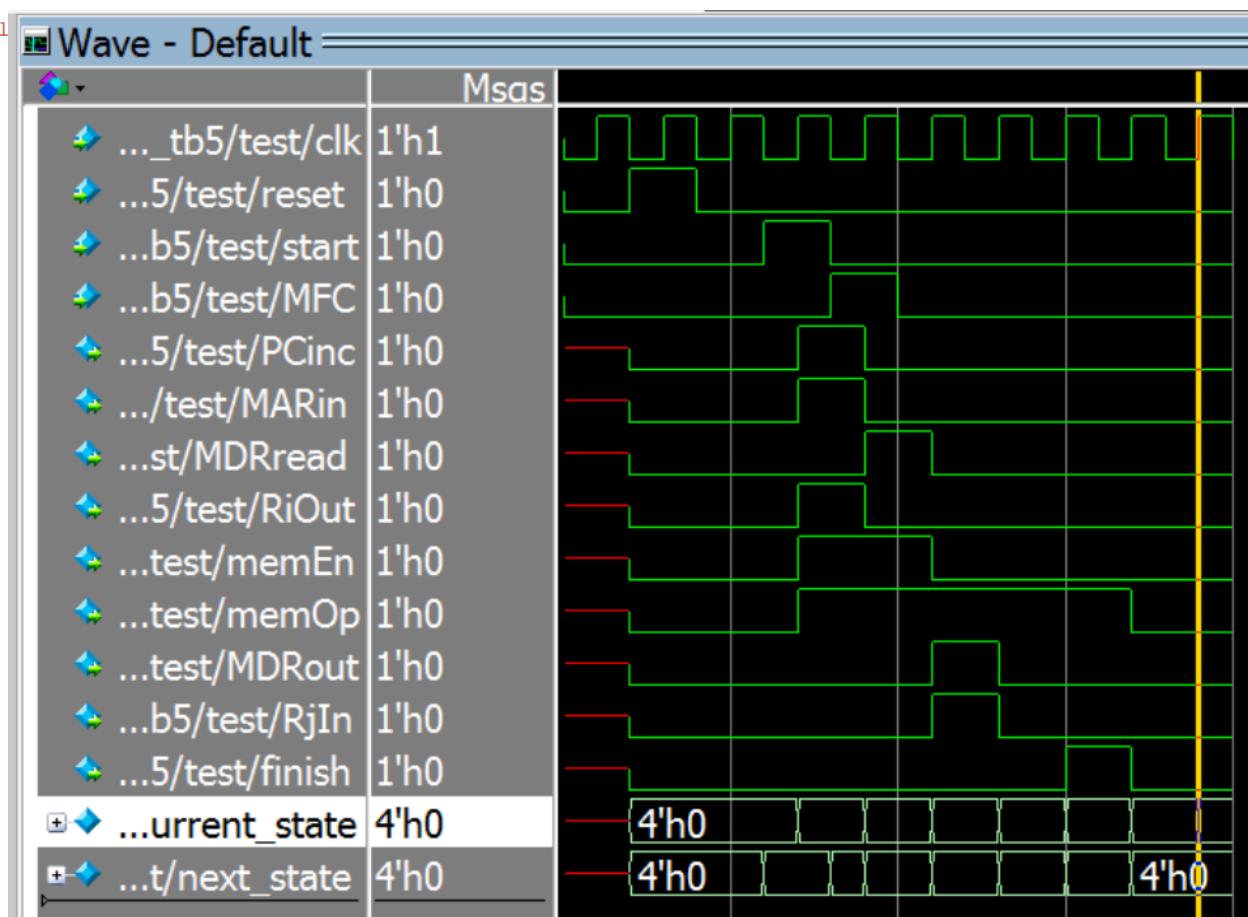
1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 module Load_FSM(clk, reset, start, MFC, PCinc, RiOut, MARin, MDRread, memEn, memOp, MDRout, RjIn, finish);
6
7 // Port Declaration
8
9 input clk; input reset; input start; input MFC;
10
11 output reg PCinc; output reg MARin; output reg MDRread; output reg RiOut;
12 output reg memEn; output reg memOp; output reg MDRout; output reg RjIn; output reg finish;
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97

```

```

1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 `timescale 1ns/1ns
6 module FSM_tb5;
7
8 // Inputs
9     reg CLK; reg RESET; reg START; reg MFC;
10 // Outputs
11 wire PCinc; wire MARin; wire MDRwrite; wire RiOut;
12 wire memEn; wire memOp; wire MDRout; wire RjOut; wire finish;
13
14 // Instantiate Module
15     Store_FSM test (CLK, RESET, START, MFC, PCinc, RiOut, MARin, MDRwrite, memEn, memOp, MDRout, RjOut, finish);
16
17 initial
18 begin
19     CLK = 0;
20     RESET = 0;
21     START = 0;
22     MFC = 0;
23     #20;
24     RESET = 1;
25     #20;
26     RESET = 0;
27     #20;
28     START = 1;
29     #20;
30     START = 0;
31     MFC = 1;
32     #20;
33     MFC = 0;
34 end
35
36 always #10 CLK = ~CLK;
37
38 endmodule

```

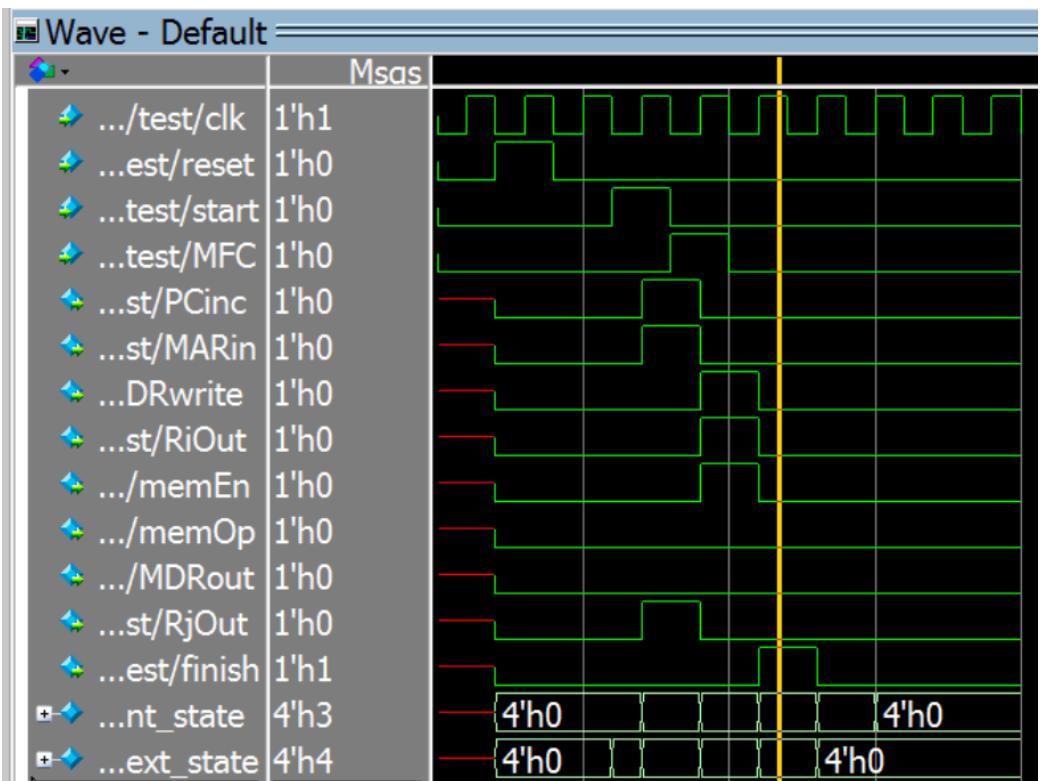


Store FSM (similar tb)

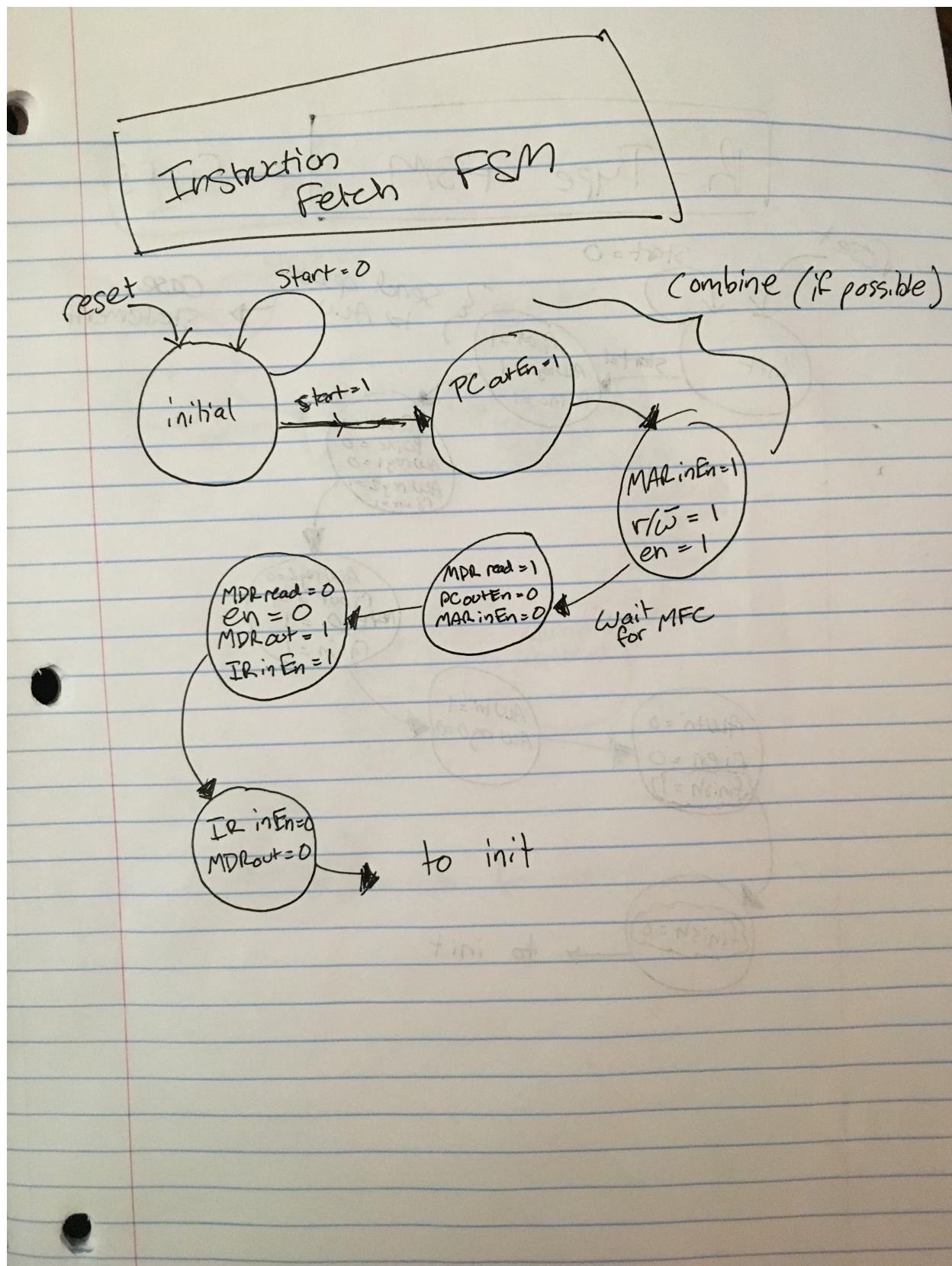
```

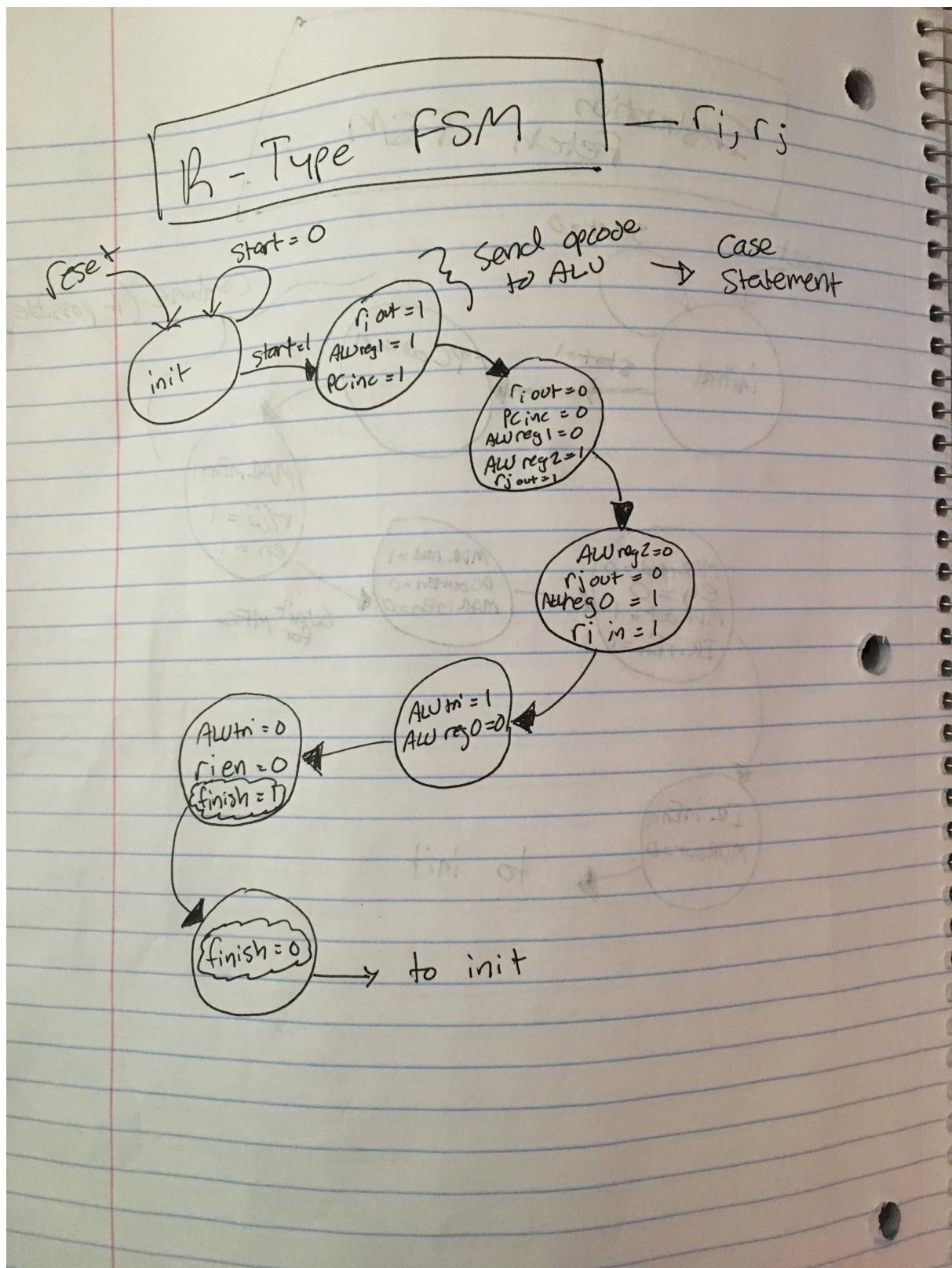
1 // Zack Fravel
2 // System Synthesis and Modeling
3 // Final Project (Step 4)
4
5 module Store_FSM(clk, reset, start, MFC, PCinc, RiOut, MARin, MDRwrite, memEn, memOp, MDRout, RjOut, finish);
6
7 // Port Declaration
8
9 input clk; input reset; input start; input MFC;
10
11 output reg PCinc; output reg MARin; output reg MDRwrite; output reg RiOut;
12 output reg memEn; output reg memOp; output reg MDRout; output reg RjOut; output reg finish;
13
14 reg[3:0] current_state, next_state;
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55 always @(current_state)                                // Output Logic
56 begin
57
58     case (current_state)
59
60         init: begin
61             PCinc = 0; MARin = 0; RiOut = 0; MDRwrite = 0;
62             memEn = 0; memOp = 0; MDRout = 0; RjOut = 0; finish = 0;
63         end
64         one: begin
65             PCinc = 1; MARin = 1; RiOut = 0; MDRwrite = 0;
66             memEn = 0; memOp = 0; MDRout = 0; RjOut = 1; finish = 0;
67         end
68         two: begin
69             PCinc = 0; MARin = 0; RiOut = 1; MDRwrite = 1;
70             memEn = 1; memOp = 0; MDRout = 0; RjOut = 0; finish = 0;
71         end
72         three: begin
73             PCinc = 0; MARin = 0; RiOut = 0; MDRwrite = 0;
74             memEn = 0; memOp = 0; MDRout = 0; RjOut = 0; finish = 1;
75         end
76         four: begin
77             PCinc = 0; MARin = 0; RiOut = 0; MDRwrite = 0;
78             memEn = 0; memOp = 0; MDRout = 0; RjOut = 0; finish = 0;
79         end
80     endcase
81
82 end
83
84 endmodule
85
86

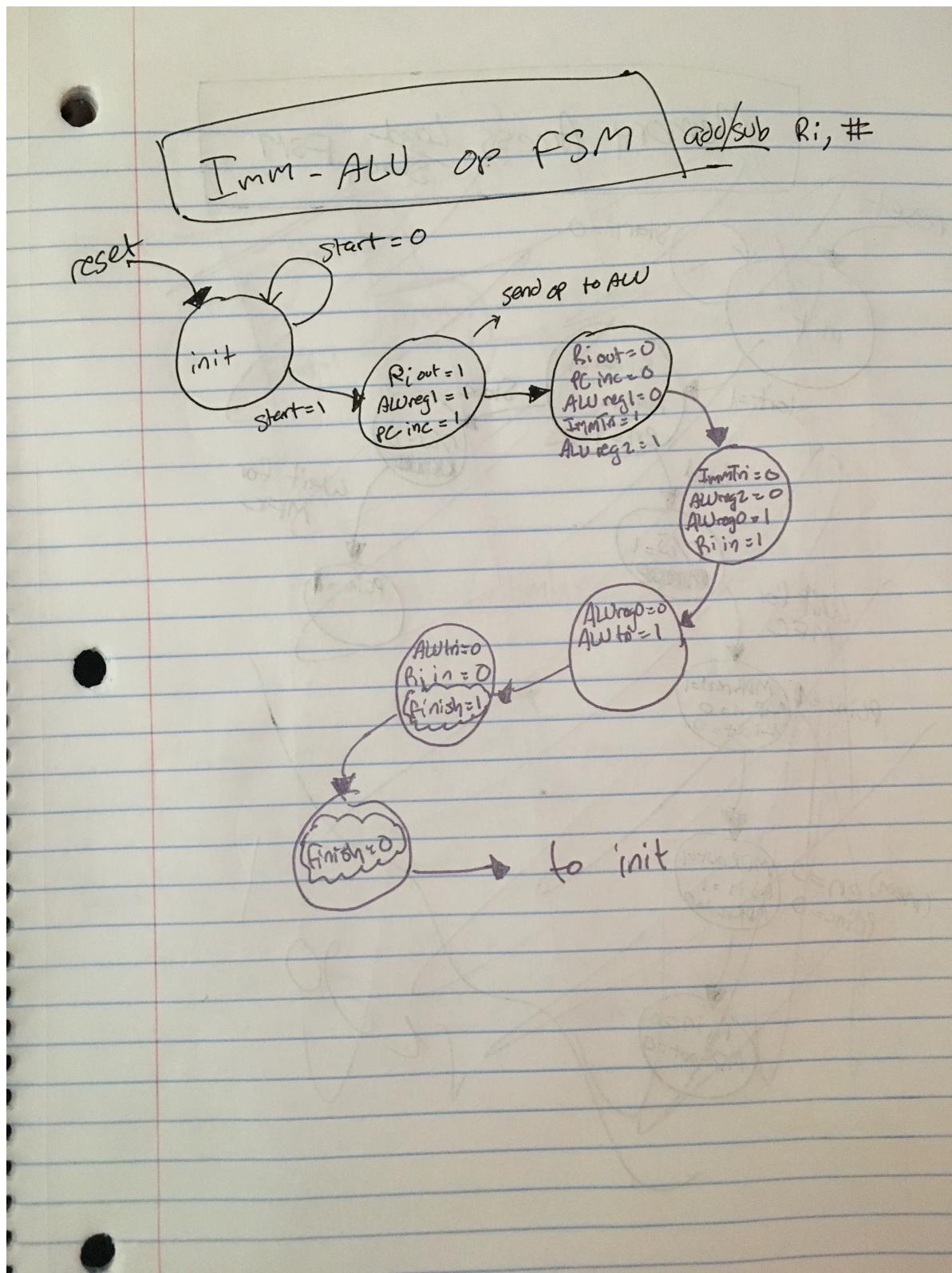
```

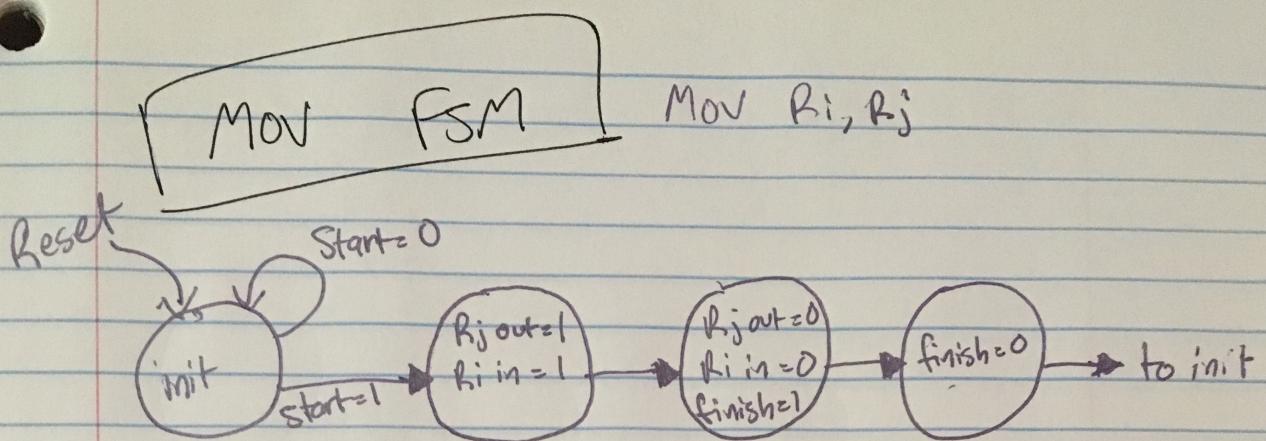


STG's

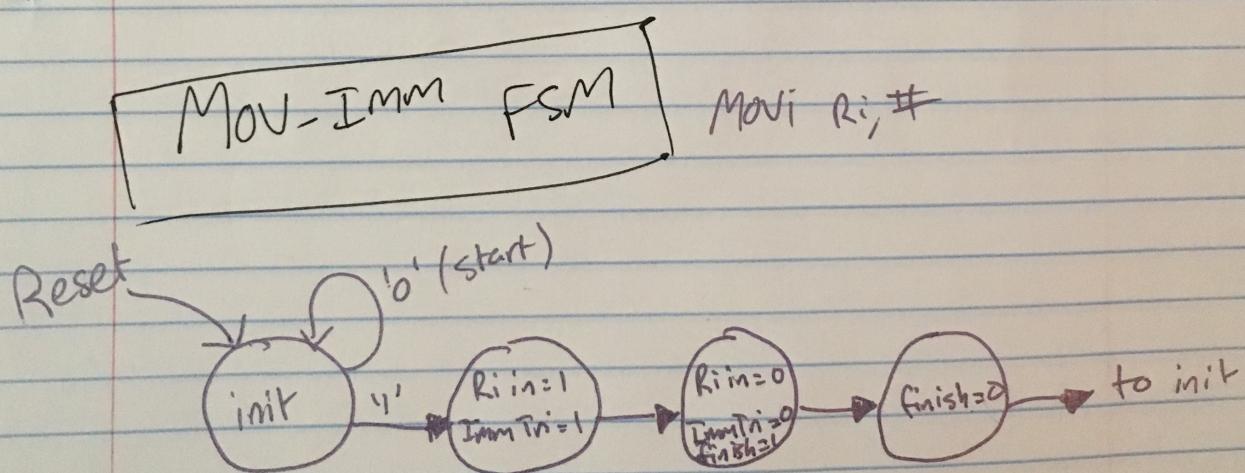








(potentially combine these?)



(ended up splitting these up)

