

Problem 1: [12 points] Drill problem
Filename: library.sv

```
1 `default_nettype none
2
3 module MagComp
4     # (parameter WIDTH = 1)
5     (input logic [WIDTH-1:0] A, B,
6      output logic AltB, AeqB, AgtB);
7
8     always_comb begin
9         if (A < B)
10             begin
11                 AeqB = 1'b0;
12                 AltB = 1'b1;
13                 AgtB = 1'b0;
14             end
15         else if (A == B)
16             begin
17                 AeqB = 1'b1;
18                 AltB = 1'b0;
19                 AgtB = 1'b0;
20             end
21         else
22             begin
23                 AgtB = 1'b1;
24                 AltB = 1'b0;
25                 AeqB = 1'b0;
26             end
27         end
28     end
29 endmodule: MagComp
30
31
32 module Multiplexer
33     # (parameter WIDTH = 2)
34     // parameter S_WIDTH = $clog2(WIDTH);
35     (input logic [WIDTH-1:0] I,
36      input logic [$clog2(WIDTH)-1:0] S,
37      output logic Y);
38
39     always_comb begin
40         if (S < WIDTH)
41             Y = I[S];
42         end
43     end
44 endmodule: Multiplexer
45
46 module Mux2to1
47     // input width
48     # (parameter WIDTH = 1)
49     (input logic [WIDTH-1:0] I0, I1,
50      input logic S,
51      output logic [WIDTH-1:0] Y);
52
53     assign Y = S ? I1 : I0;
54
55 endmodule: Mux2to1
56
57 module Decoder
58     # (parameter WIDTH = 1)
59     // parameter I_WIDTH = $clog2(WIDTH);
60     (input logic [$clog2(WIDTH)-1:0] I,
61      input logic en,
62      output logic [WIDTH-1:0] D);
63
64     always_comb begin
65         D = 0;
66         if (en)
67             begin
68                 if (I < WIDTH)
69                     D[I] = 1'b1;
70             end
71     end
72 endmodule: Decoder
```

```
71     else
72         D = I;
73     end
74
75 endmodule: Decoder
76
77 module Adder
78     # (parameter WIDTH = 1)
79     (input logic [WIDTH-1:0] A, B,
80      input logic Cin,
81      output logic [WIDTH-1:0] S,
82      output logic Cout);
83
84     assign {Cout, S} = A + B + Cin;
85
86 endmodule: Adder
87
88 module Register
89     # (parameter WIDTH = 1)
90     (input logic [WIDTH-1:0] D,
91      input logic en, clear,
92      input logic clock,
93      output logic [WIDTH-1:0] Q);
94
95     always_ff @(posedge clock)
96         if (en)
97             Q <= clear ? 0 : D;
98
99 endmodule: Register
100
101
102
103
104
105
106
107
108
109
110
111
```

Problem 1: [12 points] Drill problem
Filename: library_tests.sv

```
1 module MagComp_test ();
2   logic [4:0] A1, B1;
3   logic AltB, AgtB, AeqB;
4
5   MagComp #(5) five (.A(A1), .B(B1), .AltB, .AgtB, .AeqB);
6
7   initial begin
8     $monitor($time,, "A1=%d, B1=%d, AltB=%b, AgtB=%b, AeqB=%b",
9              A1, B1, AltB, AgtB, AeqB);
10    #5 A1 = 5'b10000;
11    B1 = 5'b00010;
12    #5 A1 = 5'b00001;
13    B1 = 5'b00001;
14    #5 B1 = 5'b01111;
15    #1 $finish;
16  end
17 endmodule: MagComp_test
18
19 module Multiplexer_test ();
20   logic [6:0] I;
21   logic Y;
22   logic [2:0] S;
23
24   Multiplexer #(7) m1 (.*);
25
26   initial begin
27     $monitor($time,, "I=%b, Y=%b, S=%d",
28              I, Y, S);
29    #5 I = 7'b0000000;
30    S = 3'b000;
31    #5 S = 3'b101;
32    #5 I = 7'b0100000;
33    #5 S = 3'b111;
34    #5 I = 7'b1111000;
35    S = 3'b001;
36    #5 S = 3'b100;
37    #1 $finish;
38  end
39
40 endmodule: Multiplexer_test
41
42 module Mux2to1_test ();
43   logic [6:0] I0, I1, Y;
44   logic S;
45
46   Mux2to1 #(7) m1 (.*);
47
48   initial begin
49     $monitor($time,, "I0=%b, I1=%b, Y=%b, S=%b",
50              I0, I1, Y, S);
51    #5 I0 = 7'b0000000;
52    I1 = 7'b1111111;
53    S = 1'b1;
54    #5 S = 1'b0;
55    #1 $finish;
56  end
57 endmodule: Mux2to1_test
58
59 module Decoder_test ();
60   // `define WIDTH1 3
61   // `define I_WIDTH1 2
62
63   logic [2:0] D;
64   logic [1:0] I;
65   logic en;
66
67   Decoder #(3) dec (.*);
68
69   initial begin
70     $monitor($time,, "I=%d, en=%b, D=%b", I, en, D);
```

```
71     #5 I = 2'b11;
72     en = 0;
73     #5 en = 1;
74     #5 I = 2'b01;
75     #1 $finish;
76 end
77 endmodule: Decoder_test
78
79 module Adder_test ();
80     // `define 8 8
81     logic Cin, Cout;
82     logic [7:0] A, B, S;
83
84     Adder #(8) add (.*);
85
86     initial begin
87         $monitor($time,, "A=%d, B=%d, Cin=%b, Cout=%b, S=%d", A, B, Cin, Cout, S);
88         #5 A = 27;
89         B = 0;
90         Cin = 1'b0;
91         #5 A = 38;
92         B = 200;
93         #5 Cin = 1'b1;
94         #5 A = 129;
95         B = 129;
96         #1 $finish;
97     end
98 endmodule: Adder_test
99
100
101 module Register_test ();
102     // `define 3 3
103     logic [2:0] D, Q;
104     logic en, clear;
105     logic clock, reset_L;
106
107     Register #(3) regis (.*);
108
109     initial begin
110         $monitor($time,, "Q=%d, D=%d, en=%d, clear=%b",
111             Q, D, en, clear);
112         clock = 0;
113         forever #5 clock = ~clock;
114     end
115
116     initial begin
117         D = 3'b000;
118         en = 0;
119         clear = 0;
120         reset_L <= 0;
121         @(posedge clock);
122         reset_L <= 1;
123         @(posedge clock);
124         D = 3'b100;
125         // reset is synchronous, so must wait for a clock edge
126         @(posedge clock);
127         en = 1;
128         @(posedge clock);
129         D = 3'b010;
130         @(posedge clock);
131         D = 3'b011;
132         clear = 1;
133         @(posedge clock);
134         en = 0;
135         @(posedge clock);
136         clear = 0;
137         @(posedge clock);
138         @(posedge clock);
139         D = 3'b111;
140         en = 1;
141         @(posedge clock);
```

```
142     D = 3'b101;
143     en = 0;
144     @(posedge clock);
145     #1 $finish;
146 end
147
148 endmodule: Register_test
149
150
151
152
153
154
155
156
157
```

Problem 5: [20 points]
Filename: hw7prob5.sv

```
1 `default_nettype none
2
3 module fsm
4   (input logic clock, reset_L, start, inputB, inputC,
5     output logic en, done, clear);
6
7   enum logic [3:0] {init=4'b1000, checkC=4'b0100,
8                     checkB=4'b0010, stop=4'b0000} Q, D;
9
10  always_comb
11    unique case (Q)
12      init: D <= start ? checkC : init;
13      checkC: D <= inputC ? checkB : stop;
14      checkB: D <= checkC;
15      stop: D <= stop;
16    endcase
17
18  always_ff @(posedge clock, negedge reset_L)
19    if (~reset_L) Q <= init;
20    else Q <= D;
21
22
23  assign clear = (Q == init) ? 1 : 0;
24  assign en = (((Q == checkB) && inputB) || (Q == init)) ? 1 : 0;
25  assign done = (Q == stop) ? 1 : 0;
26 endmodule: fsm
27
28 module prob5FSMD
29   // #(parameter W = 8)
30   (input logic start, inputC, inputB,
31     input logic clock, reset_L,
32     input logic [7:0] inputA,
33     output logic done,
34     output logic [7:0] value);
35
36   logic en, dontUse, clear;
37   logic [7:0] addOut;
38
39   fsm dut (.*);
40   Adder #(8) a1 (.A(inputA), .B(value), .Cin(1'b0),
41                 .S(addOut), .Cout(dontUse));
42   Register #(8) r1 (.D(addOut), .en, .clear,
43                    .clock, .Q(value));
44 endmodule: prob5FSMD
45
46
47
48
49
50
51
```

Problem 6: [12 points]
Filename: hw7prob6.sv

```
1 `default_nettype none
2 module fsm
3   (input logic clock, reset_L,
4     output logic firstInput, clear);
5
6   enum logic [1:0] {init=2'b00, round1=2'b01, next=2'b10} D, Q;
7
8   always_comb
9     unique case(Q)
10       init: D <= round1;
11       round1: D <= next;
12       next: D <= next;
13       default: D <= init;
14     endcase
15
16
17   always_ff @(posedge clock, negedge reset_L)
18     if (~reset_L) Q <= init;
19     else Q <= D;
20
21   always_comb begin
22     unique case (Q)
23       next: begin
24         firstInput = 0;
25         clear = 0;
26       end
27       init: begin
28         firstInput = 1;
29         clear = 1;
30       end
31       round1: begin
32         firstInput = 1;
33         clear = 0;
34       end
35     endcase
36   end
37 endmodule: fsm
38
39
40 module serialTwosComp
41   (input logic A, clock, reset_L,
42     output logic B);
43
44   logic Cin, Cout, en, clear, firstInput, flippedA;
45   // logic [1:0] nextState;
46   assign flippedA = ~A;
47   Adder #(1) a1 (.A(flippedA), .B(firstInput), .Cin, .Cout, .S(B));
48   Register #(1) coutReg (.D(Cout), .en(1'b1), .clear,
49     .clock, .Q(Cin));
50   fsm dut (.*);
51
52 endmodule: serialTwosComp
53
54 module serialTwosComp_test ();
55   // number is 10111011
56   logic A, B, Cin, Cout;
57   logic reset_L, clock;
58
59   serialTwosComp stc (.*);
60   initial begin
61     $monitor($time,, "A=%b, B=%b, Cin=%b, Cout=%b, \
62       firstInput=%b, reset=%b, s=%s",
63       A, B, stc.Cin, stc.Cout,
64       stc.firstInput, reset_L, stc.dut.Q.name);
65     clock = 0;
66     forever #5 clock = ~clock;
67   end
68
69   initial begin
70     reset_L = 0;
```

```
71     A = 1;
72     Cin = 0;
73     Cout = 0;
74     @(posedge clock);
75     reset_L = 1;
76     @(posedge clock);
77     A = 1;
78     @(posedge clock);
79     @(posedge clock);
80     A = 0;
81     @(posedge clock);
82     A = 1;
83     @(posedge clock);
84     @(posedge clock);
85     @(posedge clock);
86     A = 0;
87     @(posedge clock);
88     A = 1;
89     @(posedge clock);
90     reset_L = 0;
91     @(posedge clock);
92     A = 0;
93     reset_L = 1;
94     @(posedge clock);
95     @(posedge clock);
96     @(posedge clock);
97     @(posedge clock);
98     @(posedge clock);
99     @(posedge clock);
100    @(posedge clock);
101    @(posedge clock);
102    #1 $finish;
103    end
104
105 endmodule: serialTwosComp_test
106
107
108
109
110
111
112
113
114
115
116
```