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

```
1  `default_nettype none
2
3  // Magnitude comparator
4  module MagComp
5   #(parameter WIDTH = 16)
6    (input  logic [WIDTH-1:0] A, B,
7     output logic AltB, AeqB, AgtB);
8
9    assign AltB = A < B;
10   assign AeqB = A == B;
11   assign AgtB = A > B;
12
13 endmodule : MagComp
14
15 // Adder
16 module Adder
17  #(parameter WIDTH = 16)
18   (input  logic cin,
19    input  logic [WIDTH-1:0] A, B,
20    output logic [WIDTH-1:0] S,
21    output logic cout);
22
23   assign {cout, S} = A + B + cin;
24
25 endmodule : Adder
26
27 // Mux
28 module Multiplexer
29  #(parameter WIDTH = 16)
30   (input  logic [WIDTH-1:0] I,
31    input  logic [$clog2(WIDTH)-1:0] S,
32    output logic Y);
33
34   assign Y = I[S];
35
36 endmodule : Multiplexer
37
38 // Mux2to1
39 module Mux2to1
40  #(parameter WIDTH = 16)
41   (input  logic [WIDTH-1:0] I0, I1,
42    input  logic S,
43    output logic [WIDTH-1:0] Y);
44
45    assign Y = S ? I1 : I0;
46
47 endmodule : Mux2to1
48
49 // Decoder
50 module Decoder
51  #(parameter WIDTH = 16)
52   (input  logic [$clog2(WIDTH)-1:0] I,
53    input  logic en,
54    output logic [WIDTH-1:0] D);
55
56   assign D = en ? 1'b1 << I : 1'b0;
57
58 endmodule : Decoder
59
60 // DFlipFlop
61 module DFlipFlop
62   (input  logic D,
63    input  logic clock, preset_L, reset_L,
64    output logic Q);
65
66   always_ff @(posedge clock, negedge reset_L, negedge preset_L)
67     // when reset_L and preset_L asserted at the same time, output x
68     if (~reset_L && ~preset_L)
69       Q <= 1'bx;
```

```
70      else if (~reset_L)
71        Q <= 1'b0;
72      else if (~preset_L)
73        Q <= 1'b1;
74      else
75        Q <= D;
76
77  endmodule : DFlipFlop
78
79  // Register
80  module Register
81   #(parameter WIDTH = 16)
82    (input  logic en, clear, clock,
83     input  logic [WIDTH-1:0] D,
84     output logic [WIDTH-1:0] Q);
85
86    always_ff @(posedge clock)
87      if (en)
88        Q <= D;
89      else if (clear)
90        Q <= 1'b0;
91
92  endmodule : Register
93
94  // Counter
95  module Counter
96   #(parameter WIDTH = 16)
97    (input  logic en, clear, load, up, clock,
98     input  logic [WIDTH-1:0] D,
99     output logic [WIDTH-1:0] Q);
100
101   always_ff @(posedge clock)
102     if (clear)
103       Q <= 1'b0;
104     else if (load)
105       Q <= D;
106     else if (en) begin
107       if (up)
108         Q <= Q + 1;
109       else
110         Q <= Q - 1;
111     end
112
113 endmodule : Counter
114
115 // Sync
116 module Synchronizer
117   (input  logic async, clock,
118    output logic sync);
119
120   always_ff @(posedge clock)
121     sync <= async;
122
123 endmodule : Synchronizer
124
125 // ShiftRegister_SIPO
126 module ShiftRegister_SIPO
127  #(parameter WIDTH = 16)
128   (input  logic serial, en, left, clock,
129    output logic [WIDTH-1:0] Q);
130
131   always_ff @(posedge clock)
132     if (en) begin
133       if (left)
134         Q <= {Q[WIDTH-2:0], serial};
135       else
136         Q <= {serial, Q[WIDTH-1:1]};
137     end
138
139 endmodule : ShiftRegister_SIPO
140
```

```
141 // ShiftRegister_PIPO
142 module ShiftRegister_PIPO
143  #(parameter WIDTH = 16)
144   (input  logic en, left, load, clock,
145    input  logic [WIDTH-1:0] D,
146    output logic [WIDTH-1:0] Q);
147
148   always_ff @(posedge clock)
149     if (load)
150       Q <= D;
151     else if (en) begin
152       if (left)
153         Q <= Q << 1;
154       else
155         Q <= Q >> 1;
156     end
157
158 endmodule : ShiftRegister_PIPO
159
160 // BarrelShiftRegister
161 module BarrelShiftRegister
162  #(parameter WIDTH = 16)
163   (input  logic en, load, clock,
164    input  logic [1:0] by,
165    input  logic [WIDTH-1:0] D,
166    output logic [WIDTH-1:0] Q);
167
168   always_ff @(posedge clock)
169     if (load)
170       Q <= D;
171     else if (en) begin
172       Q <= Q << by;
173     end
174
175 endmodule : BarrelShiftRegister
```

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

```
1  `default_nettype none
2
3  module MagComp_test();
4    logic [3:0] A, B;
5    logic AltB, AeqB, AgtB;
6
7    MagComp #(4) DUT (.*);
8
9    initial begin
10     $monitor($time,, "A: %b | B: %b | AltB: %b | AeqB: %b | AgtB: %b",
11              A, B, AltB, AeqB, AgtB);
12     #0 A = 4'd15;
13        B = 4'd0;
14     #5 A = 4'd0;
15     #5 B = 4'd15;
16     #1 $finish;
17   end
18
19 endmodule : MagComp_test
20
21 module Adder_test();
22   logic cin, cout;
23   logic [3:0] A, B, S;
24
25   Adder #(4) DUT (.*);
26
27   initial begin
28     $monitor($time,, "A: %b | B: %b | cin: %b | S: %b | cout: %b",
29              A, B, cin, S, cout);
30     #0 A = 4'd10;
31        B = 4'd2;
32        cin = 1'd0;
33     #5 B = 4'd5;
34     #5 cin = 1'd1;
35     #5 B = 4'd10;
36     #1 $finish;
37   end
38
39 endmodule : Adder_test
40
41 module Multiplexer_test();
42   logic [3:0] I;
43   logic [1:0] S;
44   logic Y;
45
46   Multiplexer #(4) DUT (.*);
47
48   initial begin
49     $monitor($time,, "I: %b | S: %b | Y: %b",
50              I, S, Y);
51     #0 I = 4'd10;
52        S = 2'd0;
53     #5 S = 2'd1;
54     #5 S = 2'd2;
55     #5 S = 2'd3;
56     #1 $finish;
57   end
58
59 endmodule : Multiplexer_test
60
61 module Mux2to1_test();
62   logic[3:0] I0, I1, Y;
63   logic S;
64
65   Mux2to1 #(4) DUT (.*);
66
67   initial begin
68     $monitor($time,, "I0: %b | I1: %b | S: %b | Y: %b",
69              I0, I1, S, Y);
```

```
70      #0 I0 = 4'd15;
71         I1 = 4'd0;
72         S = 1'b0;
73      #5 S = 1'b1;
74      #1 $finish;
75    end
76
77  endmodule : Mux2to1_test
78
79  module Decoder_test();
80    logic [1:0] I;
81    logic en;
82    logic [3:0] D;
83
84    Decoder #(4) DUT (.*);
85
86    initial begin
87      $monitor($time,, "I: %b | en: %b | D: %b",
88               I, en, D);
89      #0 I = 2'd0;
90         en = 1'b1;
91      #5 I = 2'd3;
92      #5 en = 1'b0;
93      #1 $finish;
94    end
95
96  endmodule : Decoder_test
97
98  module DFlipFlop_test();
99    logic D, Q, clock, preset_L, reset_L;
100
101   DFlipFlop DUT (.*);
102
103   initial begin
104     clock = 1'b0;
105     forever #10 clock = ~clock;
106   end
107
108   initial begin
109     $monitor($time,, "D: %b | preset_L: %b | reset_L: %b | Q: %b",
110              D, preset_L, reset_L, Q);
111     D <= 1'b1;
112     preset_L <= 1'b1;
113     reset_L <= 1'b1;
114     @(posedge clock);
115     reset_L <= 1'b0;
116     @(posedge clock);
117     reset_L <= 1'b1;
118     D <= 1'b0;
119     preset_L <= 1'b0;
120     @(posedge clock);
121     #1 $finish;
122   end
123
124 endmodule : DFlipFlop_test
125
126
127 module Register_test();
128   logic [3:0] D, Q;
129   logic en, clear, clock;
130
131   Register #(4) DUT (.*);
132
133   initial begin
134     clock = 1'b0;
135     forever #10 clock = ~clock;
136   end
137
138   initial begin
139     $monitor($time,, "en: %b | clear: %b | D: %b | Q: %b",
140              en, clear, D, Q);
```

```
141     D <= 4'd1;
142     en <= 1'b1;
143     clear <= 1'b0;
144     @(posedge clock);
145     D <= 4'd15;
146     en <= 1'b0;
147     @(posedge clock);
148     en <= 1'b1;
149     @(posedge clock);
150     D <= 4'd10;
151     en <= 1'b1;
152     clear <= 1'b1;
153     @(posedge clock);
154     en <= 1'b0;
155     @(posedge clock);
156     #1 $finish;
157   end
158
159 endmodule : Register_test
160
161 module Counter_test();
162   logic en, clear, load, up, clock;
163   logic [3:0] D, Q;
164
165   Counter #(4) DUT (.*);
166
167   initial begin
168     clock = 1'b0;
169     forever #10 clock = ~clock;
170   end
171
172   initial begin
173     $monitor($time,, "D: %b | en: %b | clr: %b | load: %b | up: %b | Q: %b",
174              D, en, clear, load, up, Q);
175     D <= 4'd10;
176     en <= 1'b1;
177     clear <= 1'b0;
178     load <= 1'b1;
179     up <= 1'b1;
180     @(posedge clock);
181     load <= 1'b0;
182     @(posedge clock);
183     up <= 1'b0;
184     @(posedge clock);
185     en <= 1'b0;
186     @(posedge clock);
187     en <= 1'b1;
188     load <= 1'b1;
189     D <= 4'd1;
190     clear <= 1'd1;
191     @(posedge clock);
192     #1 $finish;
193   end
194
195 endmodule : Counter_test;
196
197 module Synchronizer_test();
198   logic async, clock, sync;
199
200   Synchronizer DUT (.*);
201
202   initial begin
203     clock = 1'b0;
204     forever #10 clock = ~clock;
205   end
206
207   initial begin
208     $monitor($time,, "async: %b | sync: %b",
209              async, sync);
210     #0 async = 1'b1;
211     #1 async = 1'b0;
```

```
212     #1 async = 1'b1;
213     @(posedge clock);
214     #1 async = 1'b0;
215     @(posedge clock);
216     #1 $finish;
217   end
218 endmodule : Synchronizer_test
219
220 module ShiftRegister_SIPO_test();
221   logic serial, en, left, clock;
222   logic [3:0] Q;
223
224   ShiftRegister_SIPO #(4) DUT (.*);
225
226   initial begin
227     clock = 1'b0;
228     forever #10 clock = ~clock;
229   end
230
231   initial begin
232     $monitor($time,, "serial: %b | en: %b | left: %b | Q: %b",
233              serial, en, left, Q);
234     serial <= 1'b1;
235     en <= 1'b0;
236     left <= 1'b1;
237     @(posedge clock);
238     en <= 1'b1;
239     @(posedge clock);
240     @(posedge clock);
241     @(posedge clock);
242     left <= 1'b0;
243     @(posedge clock);
244     @(posedge clock);
245     en <= 1'b0;
246     @(posedge clock);
247     @(posedge clock);
248     #1 $finish;
249   end
250
251 endmodule : ShiftRegister_SIPO_test
252
253 module ShiftRegister_PIPO_test();
254   logic en, left, load, clock;
255   logic [3:0] D, Q;
256
257   ShiftRegister_PIPO #(4) DUT (.*);
258
259   initial begin
260     clock = 1'b0;
261     forever #10 clock = ~clock;
262   end
263
264   initial begin
265     $monitor($time,, "D: %b | en: %b | left: %b | load: %b | Q: %b",
266              D, en, left, load, Q);
267     D <= 4'd10;
268     load <= 1'b0;
269     left <= 1'b1;
270     en <= 1'b0;
271     @(posedge clock);
272     load <= 1'b1;
273     @(posedge clock);
274     en <= 1'b1;
275     @(posedge clock);
276     load <= 1'b0;
277     @(posedge clock);
278     @(posedge clock);
279     left <= 1'b0;
280     @(posedge clock);
281     @(posedge clock);
282     #1 $finish;
```

```
283   end
284
285 endmodule : ShiftRegister_PIPO_test
286
287 module BarrelShiftRegister_test();
288   logic en, load, clock;
289   logic [1:0] by;
290   logic [3:0] D, Q;
291
292   BarrelShiftRegister #(4) DUT (.*);
293
294   initial begin
295     clock = 1'b0;
296     forever #10 clock = ~clock;
297   end
298
299   initial begin
300     D <= 1'd1;
301     en <= 1'b0;
302     load <= 1'b0;
303     by <= 2'd1;
304     @(posedge clock);
305     load <= 1'b1;
306     @(posedge clock);
307     en <= 1'b1;
308     @(posedge clock);
309     load <= 1'b0;
310     @(posedge clock);
311     @(posedge clock);
312     #1 $finish;
313   end
314
315 endmodule : BarrelShiftRegister_test
```

Problem 3: [10 points] Drill problem
Filename: hw6prob3.sv
AndrewID: xinyew

```
`default_nettype none

module hw6prob3
 (input  logic                d_in_ready, clock, reset,
  input  logic [29:0]         d_in,
  output logic                d_out_ready,
  output logic [4:0] d_out);


  logic [3:0] count1, count2, count3, count4,
              r1_Q, r2_Q, r3_Q, r4_Q;

  Count8Bits counter1 (.bits(d_in[7:0]),
                       .count(count1));
  Count8Bits counter2 (.bits(d_in[15:8]),
                       .count(count2));
  Count8Bits counter3 (.bits(d_in[23:16]),
                       .count(count3));
  Count8Bits counter4 (.bits({2'b00, d_in[29:24]}),
                       .count(count4));

  logic en_L1, clear_L1;
  Register #(4) r1 (.D(count1),
                    .Q(r1_Q),
                    .clock(clock),
                    .en(en_L1),
                    .clear(clear_L1));
  Register #(4) r2 (.D(count2),
                    .Q(r2_Q),
                    .clock(clock),
                    .en(en_L1),
                    .clear(clear_L1));
  Register #(4) r3 (.D(count3),
                    .Q(r3_Q),
                    .clock(clock),
                    .en(en_L1),
                    .clear(clear_L1));
  Register #(4) r4 (.D(count4),
                    .Q(r4_Q),
                    .clock(clock),
                    .en(en_L1),
                    .clear(clear_L1));

  logic [3:0] S_L1_part1, S_L1_part2;
  logic cout_L1_part1, cout_L1_part2;
  Adder #(4) a1 (.A(r1_Q),
                 .B(r2_Q),
                 .cin(1'b0),
                 .cout(cout_L1_part1),
                 .S(S_L1_part1));

  Adder #(4) a2 (.A(r3_Q),
                 .B(r4_Q),
                 .cin(1'b0),
                 .cout(cout_L1_part2),
                 .S(S_L1_part2));

  logic [4:0] r5_Q, r6_Q;
  logic en_L2, clear_L2;
  Register #(5) r5 (.D({cout_L1_part1, S_L1_part1}),
                    .Q(r5_Q),
                    .clock(clock),
                    .en(en_L2),
                    .clear(clear_L2));
  Register #(5) r6 (.D({cout_L1_part2, S_L1_part2}),
                    .Q(r6_Q),
                    .clock(clock),
                    .en(en_L2),
                    .clear(clear_L2));
```

```
 70
 71   logic [4:0] S_L2;
 72   Adder #(5) a3 (.A(r5_Q),
 73                  .B(r6_Q),
 74                  .cin(1'b0),
 75                  .S(S_L2));
 76
 77   logic en_L3, clear_L3;
 78   Register #(5) r7 (.D(S_L2),
 79                     .Q(d_out),
 80                     .clock(clock),
 81                     .en(en_L3),
 82                     .clear(clear_L3));
 83
 84   fsm control (clock, reset, d_in_ready, d_out_ready,
 85                en_L1, en_L2, en_L3, clear_L1, clear_L2, clear_L3);
 86
 87 endmodule: hw6prob3
 88
 89 module Count8Bits
 90   (input logic [7:0] bits,
 91    output logic [3:0] count);
 92
 93   assign count = bits[7] + bits[6] + bits[5] + bits[4] +
 94                  bits[3] + bits[2] + bits[1] + bits[0];
 95
 96 endmodule : Count8Bits
 97
 98 module fsm
 99   (input  logic clock, reset, d_in_ready,
100    output logic d_out_ready,
101    output logic en_L1, en_L2, en_L3, clear_L1, clear_L2, clear_L3);
102
103   enum logic [2:0] {A = 3'd0, B = 3'd1, C = 3'd2,
104                D = 3'd3, E = 3'd4} cur_state, n_state;
105
106   always_comb begin
107     case (cur_state)
108       A: begin  //State A -> waiting for d_in_ready
109          n_state = d_in_ready ? B : A;
110          clear_L1  = d_in_ready ? 1 : 0;
111          clear_L2  = d_in_ready ? 1 : 0;
112          clear_L3  = d_in_ready ? 1 : 0;
113          en_L1   = 0;
114          en_L2   = 0;
115          en_L3   = 0;
116          d_out_ready = 0;
117          end
118       B: begin  //State B -> all Regs cleared
119          n_state = C;
120          clear_L1 = 0;
121          clear_L2 = 0;
122          clear_L3 = 0;
123          en_L1 = 1;
124          end
125       C: begin // State C -> L1 Regs filled from counters
126          n_state = D;
127          en_L1 = 0;
128          en_L2 = 1;
129       end
130       D: begin // State D -> L2 Regs filled from Adders
131          n_state = E;
132          en_L2 = 0;
133          en_L3 = 1;
134       end
135       E: begin // State E -> L3 Regs filled, Done
136          n_state = A;
137          en_L3 = 0;
138          d_out_ready = 1;
139       end
140     endcase
```

```
141   end
142
143   always_ff @(posedge clock, posedge reset)
144     if (reset)
145       cur_state <= A;
146     else
147       cur_state <= n_state;
148
149 endmodule: fsm
```

Problem 4: [8 points]
AndrewID: xinyew

Compilation Errors:

```
hw6prob4.sv: file does not exist
```

```
Problem 4: [8 points]
Filename: hw6prob4.sv
AndrewID: xinyew

File hw6prob4.sv was not found
```

Problem 6: [8 points]
AndrewID: xinyew

Compilation Errors:

```
hw6prob6.sv: file does not exist
```

```
Problem 6: [8 points]
Filename: hw6prob6.sv
AndrewID: xinyew

File hw6prob6.sv was not found
```