Lab Code [10 points]
Filename: ChipInterface.sv

```
 1 `default_nettype none
 2
 3 module ChipInterface
 4  (output logic [6:0] HEX7, HEX6, HEX5, HEX4,
 5   output logic [17:0] LEDR,
 6  input logic [17:0] SW);
 7
 8  logic [2:0] coin1, coin2;
 9  logic [3:0] leftover, hex5val, hex4val;
10  logic always_on, hex5_on;
11
12  assign always_on = 1;
13
14  make_change asdsad (.Cost(SW[17:14]), .Paid(SW[3:0]), .Pentagons(SW[13:12]),
15  .Triangles(SW[11:10]), .Circles(SW[9:8]),
16   .FirstCoin(coin1), .SecondCoin(coin2),
17  .ExactAmount(LEDR[17]), .NotEnoughChange(LEDR[16]),
18  .CoughUpMore(LEDR[15]), .Remaining(leftover));
19
20  SevenSegmentDigit seg1 (.bcd(coin1), .blank(always_on), .segment(HEX7));
21  SevenSegmentDigit seg2 (.bcd(coin2), .blank(always_on), .segment(HEX6));
22
23  always_comb begin
24  if (leftover < 10)
25  begin
26  hex5_on = 0;
27  hex4val = leftover;
28  hex5val = 0;
29  end
30  else
31  begin
32  hex5_on = 1;
33  hex4val = leftover - 10;
34  hex5val = 1;
35  end
36  end
37
38  SevenSegmentDigit seg3 (.bcd(hex5val), .blank(hex5_on), .segment(HEX5));
39  SevenSegmentDigit seg4 (.bcd(hex4val), .blank(always_on), .segment(HEX4));
40
41 endmodule: ChipInterface
42
43
44
45
46
47
```

Lab Code [10 points]
Filename: lab3_test.sv

```
`default_nettype none

module test_change();
  logic [3:0] cost, paid;
  logic [1:0] pentagons, triangles, circles;
  logic [2:0] first_coin, second_coin;
  logic exact_amount, not_enough_change, cough_up_more;
  logic [3:0] remaining;

  make_change DUT(.Cost(cost), .Paid(paid), .Pentagons(pentagons),
                  .Triangles(triangles), .Circles(circles),
                  .FirstCoin(first_coin), .SecondCoin(second_coin),
                  .ExactAmount(exact_amount),
                  .NotEnoughChange(not_enough_change),
                  .CoughUpMore(cough_up_more), .Remaining(remaining));

  initial begin
  $monitor("cost = %b, paid = %b, pentagons = %b, triangles = %b, \
            circles = %b   --    first_coin = %b, second_coin = %b, \
            exact_amount = %b, not_enough_change = %b, \
            cough_up_more = %b, remaining = %b\n",
            cost, paid, pentagons, triangles, circles,
            first_coin, second_coin, exact_amount,
            not_enough_change, cough_up_more, remaining);

  cost = 4'b0000;
  paid = 4'b0111;
  pentagons = 2'b10;
  triangles = 2'b11;
  circles = 2'b11;

  #10 cost = 4'b0011;
    paid = 4'b1010;
    pentagons = 2'b00;
    triangles = 2'b00;
    circles = 2'b01;
  #10 cost = 4'b0101;
    paid = 4'b1010;
    pentagons = 2'b00;
    triangles = 2'b10;
    circles = 2'b00;
  #10 cost = 4'b0101;
    paid = 4'b1010;
    pentagons = 2'b01;
    triangles = 2'b10;
    circles = 2'b11;
  #10 cost = 4'b0101;
    paid = 4'b1010;
    pentagons = 2'b00;
    triangles = 2'b01;
    circles = 2'b11;
  #10 cost = 4'b1111;
    paid = 4'b0111;
    pentagons = 2'b00;
    triangles = 2'b00;
    circles = 2'b00;
  #10 cost = 4'b0010;
    paid = 4'b0100;
    pentagons = 2'b11;
    triangles = 2'b11;
    circles = 2'b11;
  #10 cost = 4'b0100;
    paid = 4'b0100;
    pentagons = 2'b10;
    triangles = 2'b10;
    circles = 2'b00;
  #10 cost = 4'b0100;
    paid = 4'b0001;
    pentagons = 2'b10;
    triangles = 2'b11;
```

```
 71     circles = 2'b11;
 72   #10 cost = 4'b0000;
 73     paid = 4'b0000;
 74     pentagons = 2'b00;
 75     triangles = 2'b01;
 76     circles = 2'b00;
 77   #10 cost = 4'b0010;
 78     paid = 4'b1000;
 79     pentagons = 2'b11;
 80     triangles = 2'b11;
 81     circles = 2'b11;
 82   #10 cost = 4'b0001;
 83     paid = 4'b0111;
 84     pentagons = 2'b00;
 85     triangles = 2'b10;
 86     circles = 2'b00;
 87   #10 cost = 4'b0011;
 88     paid = 4'b1011;
 89     pentagons = 2'b01;
 90     triangles = 2'b00;
 91     circles = 2'b11;
 92   #10 cost = 4'b0001;
 93     paid = 4'b1111;
 94     pentagons = 2'b11;
 95     triangles = 2'b11;
 96     circles = 2'b11;
 97   #10 cost = 4'b0011;
 98     paid = 4'b1010;
 99     pentagons = 2'b01;
100     triangles = 2'b11;
101     circles = 2'b00;
102   #10 cost = 4'b1111;
103     paid = 4'b1111;
104     pentagons = 2'b00;
105     triangles = 2'b00;
106     circles = 2'b00;
107   #10 cost = 4'b0010;
108     paid = 4'b1000;
109     pentagons = 2'b11;
110     triangles = 2'b11;
111     circles = 2'b00;
112
113   #10 $finish;
114   end
115
116 endmodule: test_change
117
118
119
```

Lab Code [10 points]
Filename: lab3_testCoin.sv

```
 1 `default_nettype none
 2
 3 module lab3_testCoin ();
 4    logic [3:0] FirstChange;
 5    logic [1:0] Pentagons, Triangles, Circles;
 6    logic CoughUpMore;
 7    logic [2:0] Coin;
 8    logic [1:0] PentLeft, TriLeft, CircLeft;
 9
10    CalcCoin c1 (.FirstChange, .Pentagons, .Triangles, .Circles,
11                 .CoughUpMore, .FirstCoin(Coin),
12                 .PentLeft, .TriLeft, .CircLeft);
13
14    initial begin
15    $monitor(,, "FirstChange = %d, Pentagon = %d, Triangles = %d, Circles = %d,\
16                 CoughUpMore = %b, Coin = %d, PentLeft = %d, TriLeft = %d, /
17                 CircLeft = %d",
18                 FirstChange, Pentagons, Triangles, Circles,
19                 CoughUpMore, Coin, PentLeft,
20                 TriLeft, CircLeft);
21    #1 CoughUpMore = 0;
22    #5 FirstChange = 7;
23       Pentagons = 2;
24       Triangles = 0;
25       Circles = 1;
26    #5 FirstChange = 9;
27       Pentagons = 0;
28       Triangles = 3;
29       Circles = 3;
30
31    end
32 endmodule: lab3_testCoin
33
```

Lab Code [10 points]
Filename: make_change.sv

```
 1 `default_nettype none
 2
 3 module GetRemaining
 4  (input logic [3:0] Remaining,
 5   input logic CoughUpMore,
 6   output logic NotEnoughChange);
 7
 8   always_comb begin
 9   NotEnoughChange = 1'b0;
10   if (~CoughUpMore)
11     begin
12     if (Remaining > 1'b0)
13       NotEnoughChange = 1'b1;
14     end
15   end
16 endmodule: GetRemaining
17
18 module CalcCoin
19  (input logic [3:0] FirstChange,
20   input logic [1:0] Pentagons, Triangles, Circles,
21   input logic CoughUpMore,
22   output logic [2:0] FirstCoin,
23   output logic [1:0] PentLeft, TriLeft, CircLeft);
24
25   always_comb begin
26   PentLeft = Pentagons;
27   TriLeft = Triangles;
28   CircLeft = Circles;
29   FirstCoin = 3'd000;
30   if (~CoughUpMore)
31     begin
32     if (FirstChange >= 3'd5)
33 begin
34 if (Pentagons > 0)
35  begin
36 FirstCoin = 3'b101;
37 PentLeft = Pentagons - 1;
38  end
39 else if (Triangles > 0)
40  begin
41 FirstCoin = 3'b011;
42 TriLeft = Triangles - 1;
43  end
44 else if (Circles > 0)
45  begin
46 FirstCoin = 3'b001;
47 CircLeft = Circles - 1;
48  end
49 else
50  FirstCoin = 3'b000;
51 end
52
53     else if (FirstChange >= 3'd3)
54       begin
55         if (Triangles > 0)
56           begin
57             FirstCoin = 3'b011;
58             TriLeft = Triangles - 1;
59           end
60         else if (Circles > 0)
61           begin
62           // for else's begin
63             FirstCoin = 3'b001;
64             CircLeft = Circles - 1;
65           end
66         else
67           FirstCoin = 3'b000;
68       end
69
70     else if (FirstChange >= 1)
```

```
 71       begin
 72         if (Circles > 0)
 73           begin
 74             FirstCoin = 3'b001;
 75             CircLeft = Circles - 1;
 76           end
 77         else
 78           FirstCoin = 3'b000;
 79   end
 80
 81     else
 82       FirstCoin = 3'b000;
 83   end //always_combs
 84   end
 85 endmodule: CalcCoin
 86
 87
 88 module make_change
 89   (input logic [3:0] Cost, Paid,
 90    input logic [1:0] Pentagons, Triangles, Circles,
 91    output logic [2:0] FirstCoin, SecondCoin,
 92    output logic ExactAmount, NotEnoughChange, CoughUpMore,
 93    output logic [3:0] Remaining);
 94
 95    logic [3:0] FirstChange, SecondChange;
 96    logic [1:0] PentLeft, TriLeft, CircLeft, Pent2Left, Tri2Left, Circ2Left;
 97
 98    always_comb begin
 99      if (Paid < Cost)
100  begin
101        CoughUpMore = 1'b1;
102 FirstChange = 3'd0;
103 SecondChange = 3'd0;
104 Remaining = 4'd0;
105  end
106      else
107        begin
108        CoughUpMore = 1'b0;
109        // get amount for first coin
110        FirstChange = Paid - Cost;
111        SecondChange = Paid - Cost - {1'b0, FirstCoin};
112        Remaining = SecondChange - {1'b0, SecondCoin};
113 end
114    end // for always_begin
115
116 assign ExactAmount = (Paid == Cost) ? 1'b1 : 1'b0;
117
118    CalcCoin c1 (.FirstChange, .Pentagons, .Triangles,
119                 .Circles, .FirstCoin, .PentLeft,
120                 .TriLeft, .CircLeft, .CoughUpMore);
121
122    CalcCoin c2 (.FirstChange(SecondChange), .Pentagons(PentLeft),
123                 .Triangles(TriLeft), .Circles(CircLeft),
124                 .FirstCoin(SecondCoin), .PentLeft(Pent2Left),
125                 .TriLeft(Tri2Left), .CircLeft(Circ2Left),
126                 .CoughUpMore);
127        // calculate remaining
128    GetRemaining r1 (.Remaining, .NotEnoughChange, .CoughUpMore);
129
130
131 endmodule: make_change
132
133
134
135
136
137
138
139
140
141
```

```
142
143
144
145
146
147
148
```