```
1   //
2   // Digital Logic Lab Part 4
3   // Neil Nie
4   // (c) Yongyang Nie
5   // Oct 31th, 2018
6   //
7
8   module full_adder(
9
10      a,
11      b,
12      out,
13      cin,
14      cout
15  );
16
17  input a;
18  input b;
19  input cin;
20  output out;
21  output cout;
22
23  assign cout = (a & b) | cin & (~a & b | ~b & a);
24  assign out = cin & ~a & ~b | ~cin & ~a & b | cin & a & b | ~cin & a & ~b;
25
26  endmodule
27
28  module four_bit_adder(
29
30      x,
31      y,
32      out,
33      carry
34  );
35
36  input [3:0] x;
37  input [3:0] y;
38  output [4:0] out;
39  output carry;
40
41  wire [3:0] cout;
42
43  full_adder(x[0], y[0], out[0], 0, cout[0]);
44  full_adder(x[1], y[1], out[1], cout[0], cout[1]);
45  full_adder(x[2], y[2], out[2], cout[1], cout[2]);
46  full_adder(x[3], y[3], out[3], cout[2], cout[3]);
47  assign out[4] = cout[3];
48
49  endmodule
50
51  // this display only shows one or nothing
52
53  module half_display(
54
55      in,          // accepts a four bit input
56      out          // returns a six bit output
57  );
58
59  input in;          // A
60  output [6:0] out; // D C B A G
61
62
63  assign out[6] = 1;
64  assign out[5] = 1;
65  assign out[4] = 1;
66  assign out[3] = 1;
67  assign out[2] = ~in;
68  assign out[1] = ~in;
69  assign out[0] = 1;
70
71  endmodule
72
73  // full seven segment display module.
74
75  module seven_segment_display(
76
```

```
77       in,          // accepts a four bit input
78       out          // returns a six bit output
79    );
80
81    input [3:0] in;   // A B C D
82    output [6:0] out; // F E D C B A
83
84    assign out[6] = (~in[3] & ~in[2] & ~in[1]) + (~in[3] & in[2] & in[1] & in[0]);
85    assign out[5] = (in[3] & in[2] & ~in[1]) + (~in[3] & in[1] & in[0]) + (~in[3] & ~in[2] & in[
      0]) + (~in[3] & ~in[2] & in[1]);
86    assign out[4] = (~in[1] & in[2]) | (in[0]);
87    assign out[3] = (in[2] & in[1] & in[0]) + (~in[3] & in[2] & ~in[1] & ~in[0]) + (in[3] & ~in[
      2] & in[1] & ~in[0]) + (~in[2] & ~in[1] & in[0]);
88    assign out[2] = (in[3] & in[2] & ~in[0]) + (~in[3] & ~in[2] & in[1] & ~in[0]) + (in[3] & in[
      2] & in[1]);
89    assign out[1] = (in[2] & in[1] & ~in[0]) + (in[3] & in[1] & in[0]) + (~in[3] & in[2] & ~in[1
      ] & in[0]) + (in[3] & in[2] & ~in[0]);
90    assign out[0] = (in[2] & ~in[1] & ~in[0]) + (in[3] & in[2] & ~in[1]) + (~in[3] & ~in[2] & ~
      in[1] & in[0]) + (in[3] & ~in[2] & in[1] & in[0]);
91
92    endmodule
93
94    module two_one_four_bit_mux (a, b, sel, out);
95
96    input [3:0] a;
97    input [3:0] b;
98    output [3:0] out;
99    input sel;
100
101   assign out[3] = (~sel & a[3]) | (sel & b[3]);
102   assign out[2] = (~sel & a[2]) | (sel & b[2]);
103   assign out[1] = (~sel & a[1]) | (sel & b[1]);
104   assign out[0] = (~sel & a[0]) | (sel & b[0]);
105
106   endmodule
107
108   // returns 1 if in > 9
109   // returns 0 if in < 9
110
111   module comparator(
112
113      in,
114      out
115
116   );
117
118   input [4:0] in;
119   output out;
120
121   assign out = in[4] | (in[3] & (in[2] | in[1]));
122
123   endmodule
124
125   // this converter module convert number > 9
126   // to the appropriate number
127
128   module converter(in, out);
129
130   input [3:0] in;
131   output [3:0] out;
132
133   assign out[3] = (~in[3] & ~in[2] & in[1]);
134   assign out[2] = (~in[3] & ~in[2] & ~in[1]) | (in[3] & in[2] & in[1]);
135   assign out[1] = (~in[3] & ~in[2] & ~in[1]) | (in[3] & in[2] & ~in[1]);
136   assign out[0] = (~in[3] & ~in[2] & in[0]) | (in[3] & in[2] & in[0]) | (in[0] & in[2] & in[3
      ]);
137
138   endmodule
139
140
141   // this is the display unit for part 4.
142   // the module is nearly identitcal to part 3.
143   /* Parameters
144
145      in: four bit binary input
146      hex0: display 'in' if in < 9
```

```
147             display 10 - 'in' if in > 9
148         hex1: display nothing when in < 9
149             display 1 when in > 9
150
151     */
152     module display_unit(in, hex0, hex1, convert_out);
153
154     input  [4:0] in;
155     output [6:0] hex0;
156     output [6:0] hex1;
157
158     // declare internal variables
159     wire compare_out;
160     output [3:0] convert_out;
161     wire [3:0] mux_out;
162
163     comparator(in, compare_out);
164     converter(in[3:0], convert_out); // you only need the last four bits of the input
165
166     two_one_four_bit_mux(in[3:0], convert_out, compare_out, mux_out);
167     seven_segment_display(mux_out, hex0);
168     half_display(compare_out, hex1);
169
170     endmodule
171
172     // ------------------------------------------------------------------
173
174     module Part4(
175        SW,
176        HEX0,
177        HEX1,
178     );
179
180     input [7:0] SW;
181     output [6:0] HEX0;
182     output [6:0] HEX1;
183
184     // assign x & y
185     wire [3:0] x;
186     wire [7:4] y;
187     wire [4:0] sum;
188     wire [6:0] display0;
189     wire [6:0] display1;
190     wire [3:0] mux_out;
191
192     assign x = SW[3:0];
193     assign y = SW[7:4];
194
195     four_bit_adder(x, y, sum);
196
197     display_unit(sum, HEX0, HEX1, mux_out);
198
199     endmodule
200
201
```