```verilog
//
// Digital Logic Lab Part 4
// Neil Nie
// (c) Yongyang Nie
// Oct 31th, 2018
//

module full_adder(

    a,
    b,
    out,
    cin,
    cout
);

input a;
input b;
input cin;
output out;
output cout;

assign cout = (a & b) | cin & (~a & b | ~b & a);
assign out = cin & ~a & ~b | ~cin & ~a & b | cin & a & b | ~cin & a & ~b;

endmodule

module four_bit_adder(

    x,
    y,
    out,
    carry
);

input [3:0] x;
input [3:0] y;
output [4:0] out;
output carry;

wire [3:0] cout;

full_adder(x[0], y[0], out[0], 0, cout[0]);
full_adder(x[1], y[1], out[1], cout[0], cout[1]);
full_adder(x[2], y[2], out[2], cout[1], cout[2]);
full_adder(x[3], y[3], out[3], cout[2], cout[3]);
assign out[4] = cout[3];

endmodule

// this display only shows one or nothing

module half_display(

    in,          // accepts a four bit input
    out          // returns a six bit output
);

input in;           // A
output [6:0] out; // D C B A G


assign out[6] = 1;
assign out[5] = 1;
assign out[4] = 1;
assign out[3] = 1;
assign out[2] = ~in;
assign out[1] = ~in;
assign out[0] = 1;

endmodule

// a full full seven segment display module.

module full_display(
    in,          // accepts a four bit input
    out         // returns a six bit output
);

input [3:0] in;    // A B C D
output [6:0] out; // F E D C B A

assign out[6] = (~in[3] & ~in[2] & ~in[1]) + (~in[3] & in[2] & in[1] & in[0]);
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]);
assign out[4] = (~in[1] & in[2]) | (in[0]);
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]);
assign out[2] = (in[3] & in[2] & ~in[0]) + (~in[3] & ~in[2] & in[1] & ~in[0]) + (in[3] & in[
2] & in[1]);
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]);
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]);

endmodule

module two_one_four_bit_mux(a, b, sel, out);

input [3:0] a;
input [3:0] b;
output [3:0] out;
input sel;

assign out[3] = (~sel & a[3]) | (sel & b[3]);
assign out[2] = (~sel & a[2]) | (sel & b[2]);
assign out[1] = (~sel & a[1]) | (sel & b[1]);
assign out[0] = (~sel & a[0]) | (sel & b[0]);

endmodule

// returns 1 if in > 9
// returns 0 if in < 9

module comparator(

    in,
    out

);

input [4:0] in;
output out;

assign out = in[4] | in[3] & (in[2] | in[1]);

endmodule

// this converter module convert number > 9
// to the appropriate number

module converter(

    in,
    out
);

input [3:0] in;
output [3:0] out;

assign out[3] = (in[3] & ~in[2] & ~in[1]);
assign out[2] = (in[2] & in[1]) | (~in[3] & ~in[2] & ~in[1] & ~in[0]);
assign out[1] = (in[3] & in[2] & ~in[1]) | (~in[3] & in[2] & in[1]) | (~in[3] & ~in[2] & ~in
[1] & ~in[0]);
assign out[0] = (in[3] & in[0]) | (in[2] & in[1] & in[0]);

endmodule


// this is the display unit for part 4.
// the module is nearly identitcal to part 3.
/* Parameters
```

```

	in: four bit binary input
	hex0: display 'in' if in < 9
	      display 10 - 'in' if in > 9
	hex1: display nothing when in < 9
	      display 1 when in > 9

*/
module display_unit(

	in,
	hex0,
	hex1
);

input  [4:0] in;
output [6:0] hex0;
output [6:0] hex1;

// declare internal variables
wire        compare_out;
wire [3:0]  convert_out;
wire [3:0]  mux_out;

comparator(in, compare_out);
converter(in[3:0], convert_out); // you only need the last four bits of the input

two_one_four_bit_mux(in[3:0], convert_out, compare_out, mux_out);
full_display(mux_out, hex0);
half_display(compare_out, hex1);

endmodule

// --------------------------------------------------------------

module Part4(

	SW,
	HEX0,
	HEX1
);

input [7:0] SW;
output [6:0] HEX0;
output [6:0] HEX1;

// assign x & y
wire [3:0] x;
wire [7:4] y;
wire [4:0] sum;
wire [6:0] display0;
wire [6:0] display1;

assign x = SW[3:0];
assign y = SW[7:4];

four_bit_adder(x, y, sum);

display_unit(sum, display0, display1);

assign HEX0 = display0;
assign HEX1 = display1;

endmodule


```