```
1
2   // 32-bit adder with 2 adder_16b instances. This module represents the top-level entity
    used in the ALU for the ADD and SUB operations.
3
4   module adder_32b (
5      input cin, // carry-in (used for subtraction)
6      input [31:0] x, y, // Summands
7      output cout, // carry-out (not used in ALU due to fixed register size in reg_file)
8      output [31:0] s // Result (wire)
9      );
10
11     // carry-in signal for each 16-bit sub-adder. We use 'h' to denote a 'hierarchical' carry.
12     wire [2:0] hc;
13     assign hc[0] = cin;
14
15     // 'hierachical' Generate and Propagate signals.
16     wire [1:0] hP, hG;
17
18     // 2 adder_16b instances, each for a given 16-bit subset of x and y.
19     genvar i;
20     generate
21     for (i=0; i<2; i = i+1) begin : subadders
22        adder_16b subadder (hc[i], x[16*i+15: 16*i], y[16*i+15: 16*i], hP[i], hG[i], s[16*i+15
    : 16*i]);
23     end
24     endgenerate
25
26     // Hierarchical carries according to the lookahead framework.
27     assign hc[1] = hG[0] | hP[0] & cin;
28     assign hc[2] = hG[1] | hP[1] & hG[0] | hP[1] & hP[0] & cin;
29     assign cout = hc[2];
30
31  endmodule
32
33  //---------------Testbench---------------//
34
35  module adder_32b_testbench();
36
37     reg cin;
38     reg [31:0] x, y;
39     wire cout;
40     wire [31:0] s;
41
42     // Design under test.
43     adder_32b dut (cin, x, y, cout, s);
44
45     // 6 edge-cases are validated for testbench concision.
46     initial begin
47        // Test 1: Small positive values
48        cin = 0; x = 32'b00000000000000000000000000000101; y =
    32'b00000000000000000000000000001011; #10;
49        $display("Test 1: cin = %b, x = %b, y = %b, s = %b, cout = %b", cin, x, y, s, cout);
50
51        // Test 2: Mixed range values
52        cin = 0; x = 32'b00000000111111110000000011111111; y =
    32'b00000011111110000111111110000111; #10;
53        $display("Test 2: cin = %b, x = %b, y = %b, s = %b, cout = %b", cin, x, y, s, cout);
54
55        // Test 3: Maximum 32-bit values (check carry-out well handled)
56        cin = 0; x = 32'b11111111111111111111111111111111; y =
    32'b11111111111111111111111111111111; #10;
57        $display("Test 3: cin = %b, x = %b, y = %b, s = %b, cout = %b", cin, x, y, s, cout);
58
59        // Test 4: Carry-in enabled (check that all only two bits are high)
60        cin = 1; x = 32'b00000000000000001111111111111111; y =
    32'b00000000000000000000000000000001; #10;
61        $display("Test 4: cin = %b, x = %b, y = %b, s = %b, cout = %b", cin, x, y, s, cout);
62
63        // Test 5: Large values, no carry-out
64        cin = 0; x = 32'b11110000111100001111000011110000; y =
    32'b00001111000011110000111100001111; #10;
65        $display("Test 5: cin = %b, x = %b, y = %b, s = %b, cout = %b", cin, x, y, s, cout);
66
67        // Test 6: zero inputs
68        cin = 0; x = 32'b0; y = 32'b0; #10;
```

```
69          $display("Test 6: cin = %b, x = %b, y = %b, s = %b, cout = %b", cin, x, y, s, cout);
70
71          $stop;
72      end
73  endmodule
74
```