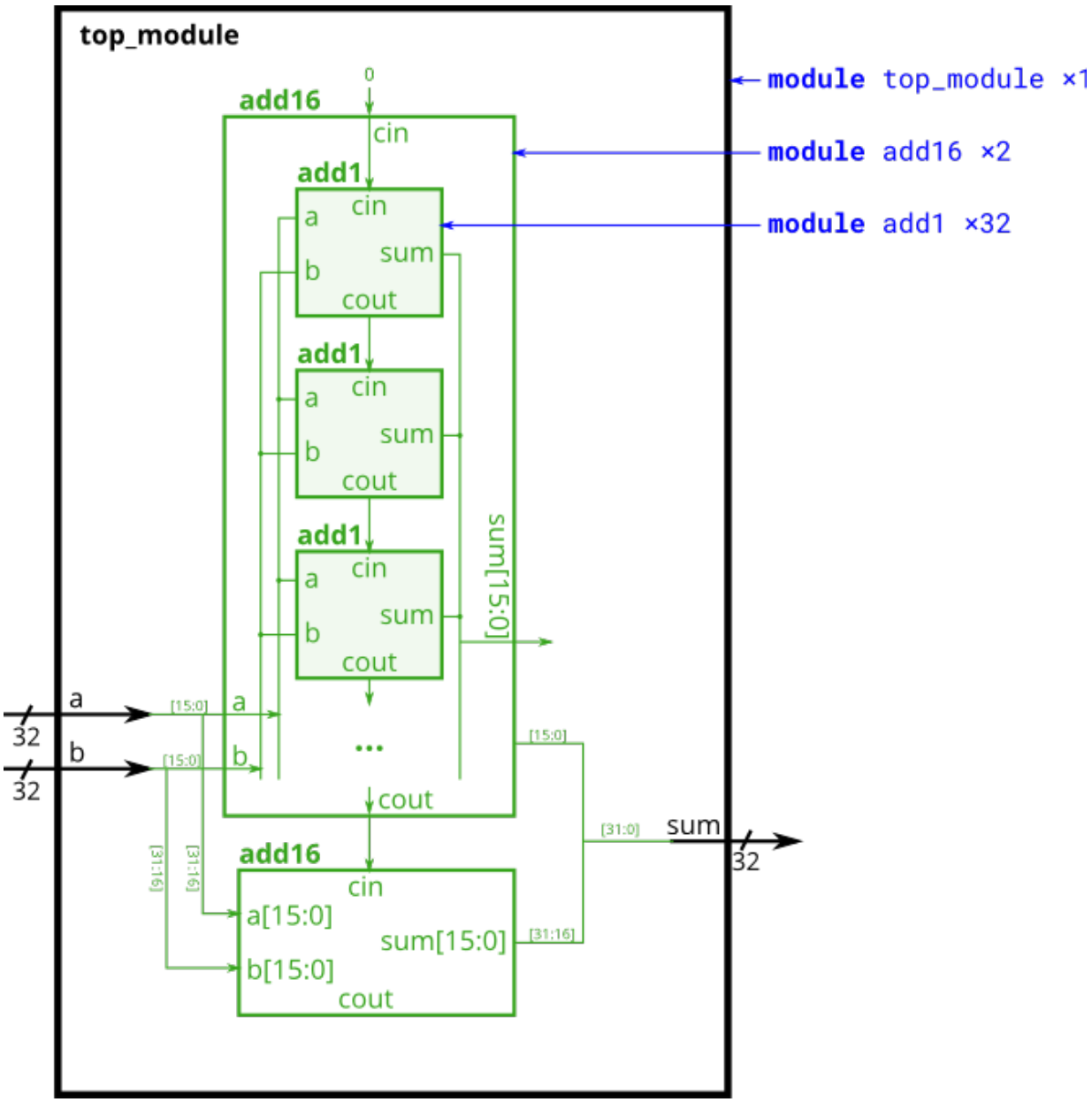
```
module top_module (
    input [31:0]    a,
    input [31:0]    b,
    output  [31:0]  sum
);
    wire [15:0] suma161, suma162;
    wire suma1,suma2,suma3,suma4,suma5,suma6,suma7,suma8,suma9,suma10,suma11,suma12,suma13,suma14,suma15
,suma16;
    wire couta1,couta2,couta3,couta4,couta5,couta6,couta7,couta8,couta9,couta10,couta11,couta12,couta13,
couta14,couta15,couta16;
    wire [15:0] cout;
    add1 a1 (a[0], b[0], 1'b0, suma1, couta1);
    add1 a2 (a[1], b[1], couta1, suma2, couta2);
    add1 a3 (a[2], b[2], couta2, suma3, couta3);
    add1 a4 (a[3], b[3], couta3, suma4, couta4);
    add1 a5 (a[4], b[4], couta4, suma5, couta5);
    add1 a6 (a[5], b[5], couta5, suma6, couta6);
    add1 a7 (a[6], b[6], couta6, suma7, couta7);
    add1 a8 (a[7], b[7], couta7, suma8, couta8);
    add1 a9 (a[8], b[8], couta8, suma9, couta9);
    add1 a10 (a[9], b[9], couta9, suma10, couta10);
    add1 a11 (a[10], b[10], couta10, suma11, couta11);
    add1 a12 (a[11], b[11], couta11, suma12, couta12);
    add1 a13 (a[12], b[12], couta12, suma13, couta13);
    add1 a14 (a[13], b[13], couta13, suma14, couta14);
    add1 a15 (a[14], b[14], couta14, suma15, couta15);
    add1 a16 (a[15], b[15], couta15, suma16, couta16);
    add16 a162 (a[31:16], b[31:16], couta16,suma162, cout);
    assign suma161 = {suma16,suma15, suma14,suma13,suma12,suma11,suma10,suma9,suma8,suma7,suma6,suma5,su
ma4,suma3,suma2,suma1};
    assign sum = {suma162,suma161};
endmodule
// module add16 (input [15:0] a, input [15:0] b, input cin, output[15:0] sum, output cout); (provided) =
> no need to write
//module add1 (input a, input b, input cin, output sum, output cout);
module add1 (input a, input b, input cin, output sum, output cout);
// Full adder here
assign sum = cin ^ a ^ b;
assign cout = a&cin|b&cin|a&b;
endmodule
```

```
module top_module (
    input [31:0]    a,
    input [31:0]    b,
    output  [31:0]  sum
);
    wire [15:0] suma161, suma162;
    wire suma1,suma2,suma3,suma4,suma5,suma6,suma7,suma8,suma9,suma10,suma11,suma12,suma13,suma14,suma15,
suma16;
    wire couta1,couta2,couta3,couta4,couta5,couta6,couta7,couta8,couta9,couta10,couta11,couta12,couta13,c
outa14,couta15,couta16;
    wire [15:0] cout;
    add1 a1 (add16.a[0], add16.b[0], 1'b0, suma1, couta1);
    add1 a2 (add16.a[1], add16.b[1], couta1, suma2, couta2);
    add1 a3 (add16.a[2], add16.b[2], couta2, suma3, couta3);
    add1 a4 (add16.a[3], add16.b[3], couta3, suma4, couta4);
    add1 a5 (add16.a[4], add16.b[4], couta4, suma5, couta5);
    add1 a6 (add16.a[5], add16.b[5], couta5, suma6, couta6);
    add1 a7 (add16.a[6], add16.b[6], couta6, suma7, couta7);
    add1 a8 (add16.a[7], add16.b[7], couta7, suma8, couta8);
    add1 a9 (add16.a[8], add16.b[8], couta8, suma9, couta9);
    add1 a10 (add16.a[9], add16.b[9], couta9, suma10, couta10);
    add1 a11 (add16.a[10], add16.b[10], couta10, suma11, couta11);
    add1 a12 (add16.a[11], add16.b[11], couta11, suma12, couta12);
    add1 a13 (add16.a[12], add16.b[12], couta12, suma13, couta13);
    add1 a14 (add16.a[13], add16.b[13], couta13, suma14, couta14);
    add1 a15 (add16.a[14], add16.b[14], couta14, suma15, couta15);
    add1 a16 (add16.a[15], add16.b[15], couta15, suma16, couta16);
    add16 a162 (a[31:16], b[31:16], couta16,suma162, cout);
    assign suma161 = {suma16,suma15, suma14,suma13,suma12,suma11,suma10,suma9,suma8,suma7,suma6,suma5,sum
a4,suma3,suma2,suma1};
    assign sum = {suma162,suma161};
endmodule
// module add16 (input [15:0] a, input [15:0] b, input cin, output[15:0] sum, output cout); (provided) =>
 no need to write
//module add1 (input a, input b, input cin, output sum, output cout);
module add1 (input a, input b, input cin, output sum, output cout);
// Full adder here
assign sum = cin ^ a ^ b;
assign cout = a&cin|b&cin|a&b;
endmodule
```



In this exercise, you will create a circuit with two levels of hierarchy. Your top_module will instantiate two copies of add16 (provided), each of which will instantiate 16 copies of add1 (which you must write). Thus, you must write two modules: top_module and add1.
Like module_add, you are given a module add16 that performs a 16-bit addition. You must instantiate two of them to create a 32-bit adder. One add16 module computes the lower 16 bits of the addition result, while the second add16 module computes the upper 16 bits of the result. Your 32-bit adder does not need to handle carry-in (assume 0) or carry-out (ignored).

Connect the add16 modules together as shown in the diagram below. The provided module add16 has the following declaration:

module add16 (input[15:0] a, input[15:0] b, input cin, output[15:0] sum, output cout);

Within each add16, 16 full adders (module add1, not provided) are instantiated to actually perform the addition. You must write the full adder module that has the following declaration:

module add1 (input a, input b, input cin, output sum, output cout);