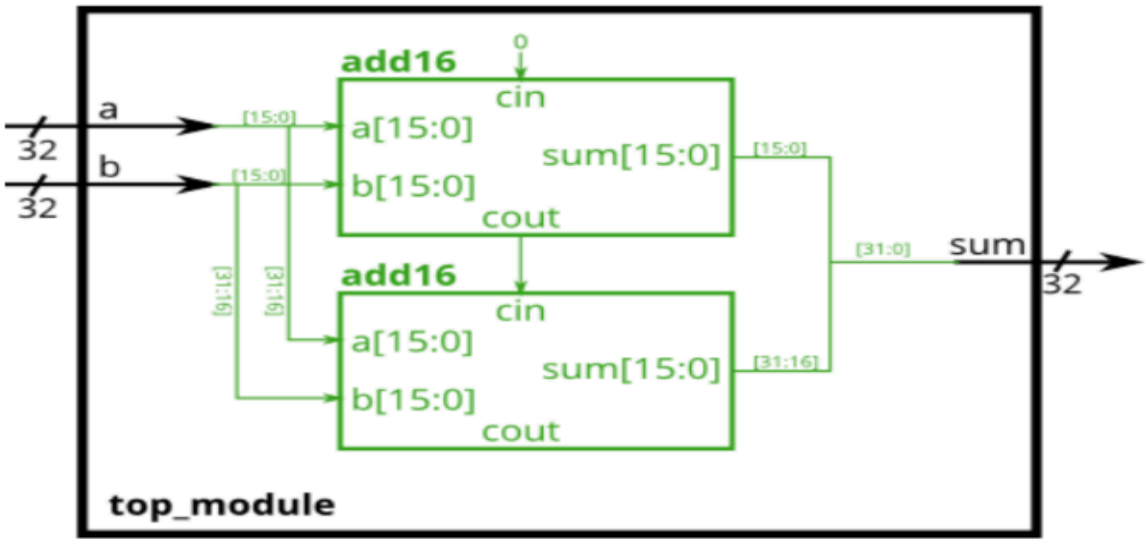
KARTHICK RAJA B
126004121

Given:

You have to design a module add16 that performs a 16-bit addition. 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, after receiving the carry-out from the first adder. Your 32-bit adder does not need to handle carry-in (assume 0) or carry-out (ignored), but the internal modules need to in order to function correctly. (In other words, the add16 module performs 16-bit a + b + cin, while your module performs 32-bit a + b).



Verilog Code:

```
module add16 (a,b,cin,sum,cout);
input [15:0] a,b;
input cin;
output [15:0] sum;
output cout;
assign {cout,sum} = a + b + cin;
endmodule

module add32 (a,b,sum);
input [31:0] a,b;
```

```
output [31:0] sum;
wire w1;
add16 a1(a[15:0],b[15:0],cin,sum[15:0],w1);
add16 a2(a[31:16],b[31:16],w1,sum[31:16],cout);
endmodule
```

Testbench Code:

```
module test_bench_tb7 ();
reg [31:0] a,b;
wire [31:0] out;

add32 s1 (a,b,out);
initial
begin

a={32{1'b1}}; b={32{1'b0}};   #100;
a={32{1'b1}}; b={32{1'b1}};   #100;
a={32{1'b1}}; b={16{2'b10}};  #100;
a={16{2'b01}}; b={32{1'b0}};

end
endmodule
```

Functional Simulation: