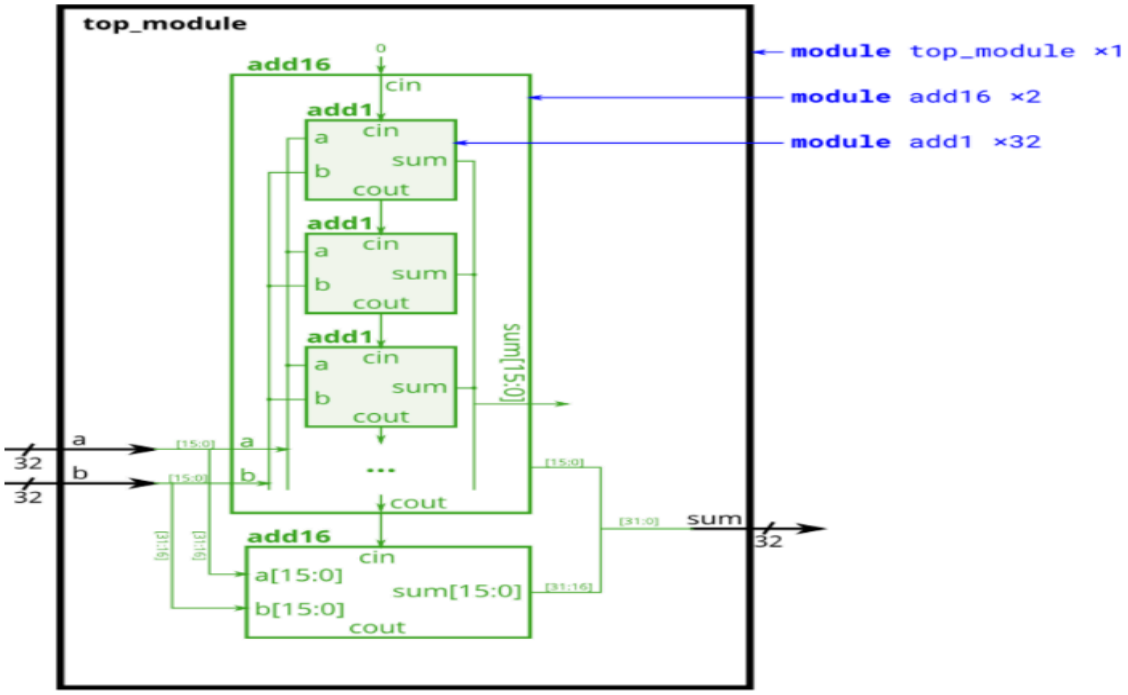
KARTHICK RAJA B
126004121

Given:

In this exercise, you will create a circuit with two levels of hierarchy.
Your top_module will instantiate two copies of add16, each of which will instantiate 16 copies of add1. Like module_add, you can design 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).



Verilog Code:

```
// add1
module add1 (a,b,cin,sum,cout);
input a,b,cin;
output sum,cout;
assign{cout,sum}= a + b + cin;
endmodule
```

```
// add16
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

// add32
module RTL_Day_8(a,b,sum);
input [31:0] a,b;
output [31:0]sum;
wire [16:1]w;

// instantiating for lower 16 bits results
add1 m1   (a[0],b[0],0,sum[0],w[1]);                 //1
add1 m2   (a[1],b[1],w[1],sum[1],w[2]);              //2
add1 m3   (a[2],b[2],w[2],sum[2],w[3]);              //3
add1 m4   (a[3],b[3],w[3],sum[3],w[4]);              //4
add1 m5   (a[4],b[4],w[4],sum[4],w[5]);              //5
add1 m6   (a[5],b[5],w[5],sum[5],w[6]);              //6
add1 m7   (a[6],b[6],w[6],sum[6],w[7]);              //7
add1 m8   (a[7],b[7],w[7],sum[7],w[8]);              //8
add1 m9   (a[8],b[8],w[8],sum[8],w[9]);              //9
add1 m10 (a[9],b[9],w[9],sum[9],w[10]);             //10
add1 m11 (a[10],b[10],w[10],sum[10],w[11]);  //11
add1 m12 (a[11],b[11],w[11],sum[11],w[12]);   //12
add1 m13 (a[12],b[12],w[12],sum[12],w[13]);  //13
add1 m14 (a[13],b[13],w[13],sum[13],w[14]);  //14
add1 m15 (a[14],b[14],w[14],sum[14],w[15]);  //15
add1 m16 (a[15],b[15],w[15],sum[15],w[16]);  //16

// instantiating for upper 16 bits results
add16 a2(a[31:16],b[31:16],w[16],sum[31:16],cout);
```

endmodule

Testbench Code:

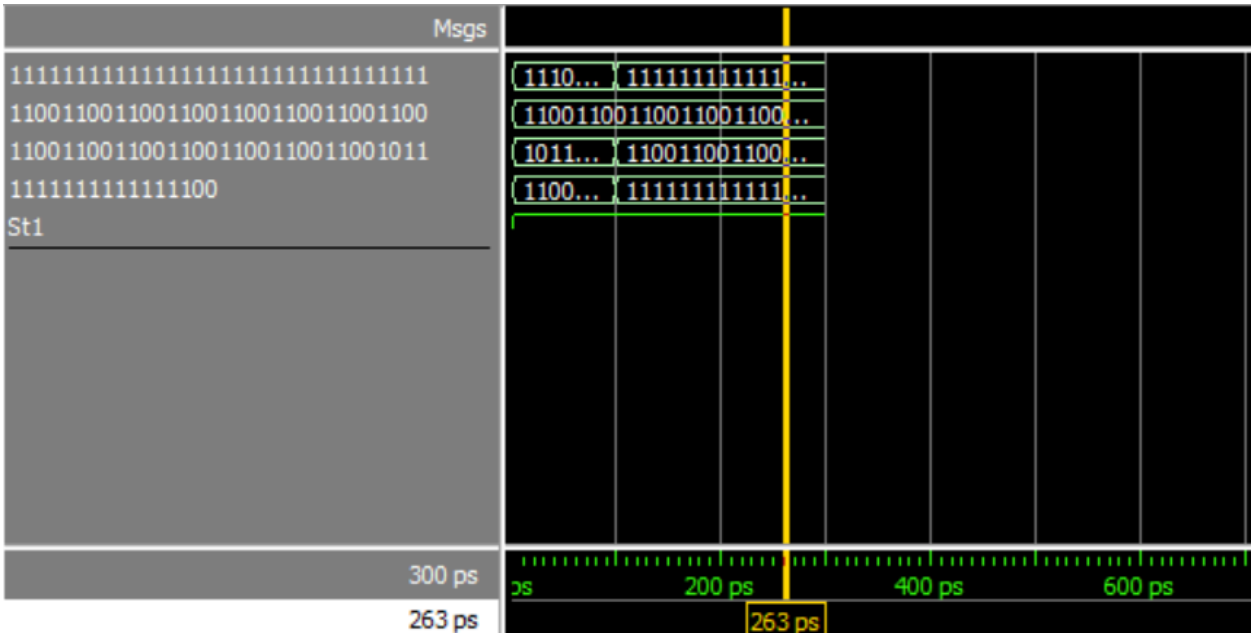
```
module test_bench_tb8 ();
reg [31:0] a,b;
wire [31:0] sum;

RTL_Day_8 s1 (a,b,sum);
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:



Alternative Verilog Code:

```
module add_16bit(input [15:0]a,b,input cin,output [15:0]sum,output cout);
wire [15:0] c;
genvar i;
generate
for (i=0;i<16;i=i+1) begin
  if (i == 0) begin
    add_1bit m1(a[i],b[i],cin,sum[i],c[i]);
  end
  else
   add_1bit m2(a[i],b[i],c[i-1],sum[i],c[i]);
  end
endgenerate
assign cout = c[15];
endmodule
```