Main code

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 08:12:28 03/26/2018

// Design Name:

// Module Name: booth

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module p(

input [15:0] a, //input of 16 bits(multiplicand)

input [15:0] b, // input of 16 bits(multiplier)

output reg[63:0] c, //output of 64 bits

input clk,

input rst //reset

);

reg[31:0] y; //sub partial products without radix4

reg[2:0] x; //3 bits conactenated from the multiplier

reg[4:0] i=0; // numbers of concatenated 3 bits(x)

reg[32:0] b1; // multiplier conatenated with 0 at the lsb bit

reg[63:0] y1; //partial product with radix 4

always@(posedge clk)

begin

if(rst==1) // when reset is 1 we initialize the outputs and the regs

begin

c=0;

i=0;

y=0;

end

if(rst==0 && i<16)// when reset is 0 we begin the operation

begin

if(rst==0)

begin

b1={b,1'b0}; // concatenated multiplier with 0 at LSB bit

if(i<16)

begin

x={b1[2\*i+2],b1[2\*i+1],b1[2\*i]}; // 3 bits taken from the multiplier(b1) from LSB

case(x) //compared with the booth radix 4 table and multiplied respectively with the multiplicand(a)

3'b000:y=0\*a;

3'b001:y=1\*a;

3'b010:y=1\*a;

3'b011:y=2\*a;

3'b100:y=(-2)\*a;

3'b101:y=(-1)\*a;

3'b110:y=(-1)\*a;

3'b111:y=0\*a;

default:y=31'dX;

endcase

y1={{32{y[31]}},y}; // sign extension of the msb bit of y to get the 64 bits partial product

c=c+(y1\*(4\*\*i)); //partial products summed up and stored in output(c)

i=i+1; //increment of the number of x

end

end

end

end

endmodule

testbench code

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 09:06:51 03/27/2018

// Design Name:

// Module Name: p\_tb

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module p\_tb();

reg[15:0] a,b; //inputs of 16 bits

reg clk;

reg rst; //reset

wire[31:0]c; //output of multiplication

p dut(a,b,c,clk,rst );//calling main program

initial begin //setting clock

clk = 0;

forever #1 clk=~clk;

end

initial begin

rst=1; //when reset is 1 output is assigned 0 according to main program

#10

rst=0;a=16'b0000000011111111; b=16'b0000000011111111; // setting different cases which is input(a=255,b=255)

#1000

rst=1;

#10

rst=0;a=16'b0000000011111111;b=16'b1111111100000001; //a=255,b=-255

#1000

rst=1;

#10

rst=0;a=16'b1111111100000001;b=16'b1111111100000001;//a=-255,b=-255

#1000

rst=1;

#10

rst=0;a=16'b1111111111111111;b=16'b1111111111111111;#1000;//a=ffff,b=ffff

$finish; //finishing the simulation

end

endmodule