module MULTU(

input clk, //乘法器时钟信号

input reset, //复位信号

input [15:0] a, //输入a(被乘数)

input [15:0] b, //输入b(乘数)

output [31:0] z //乘积输出z

);

reg [31:0] result; //申请寄存器

reg[31:0] stored[15:0];

reg[31:0] add1[7:0];

reg[31:0] add2[3:0];

reg[31:0] add3[1:0];

integer i;

always @( posedge clk or reset) begin

if(reset) begin

stored[0] <= 32'b0;

stored[1] <= 32'b0;

stored[2] <= 32'b0;

stored[3] <= 32'b0;

stored[4] <= 32'b0;

stored[5] <= 32'b0;

stored[6] <= 32'b0;

stored[7] <= 32'b0;

stored[8] <= 32'b0;

stored[9] <= 32'b0;

stored[10] <= 32'b0;

stored[11] <= 32'b0;

stored[12] <= 32'b0;

stored[13] <= 32'b0;

stored[14] <= 32'b0;

stored[15] <= 32'b0;

add1[0] <= 32'b0;

add1[1] <= 32'b0;

add1[2] <= 32'b0;

add1[3] <= 32'b0;

add1[4] <= 32'b0;

add1[5] <= 32'b0;

add1[6] <= 32'b0;

add1[7] <= 32'b0;

add2[0] <= 32'b0;

add2[1] <= 32'b0;

add2[2] <= 32'b0;

add2[3] <= 32'b0;

add3[0] <= 32'b0;

add3[1] <= 32'b0;

result <= 32'b0;

end

else begin

stored[0] <= b[0] ? { 16'b0, a } : 32'b0;

stored[1] <= b[1] ? { 15'b0, a, 1'b0 } : 32'b0;

stored[2] <= b[2] ? { 14'b0, a, 2'b0 } : 32'b0;

stored[3] <= b[3] ? { 13'b0, a, 3'b0 } : 32'b0;

stored[4] <= b[4] ? { 12'b0, a, 4'b0 } : 32'b0;

stored[5] <= b[5] ? { 11'b0, a, 5'b0 } : 32'b0;

stored[6] <= b[6] ? { 10'b0, a, 6'b0 } : 32'b0;

stored[7] <= b[7] ? { 9'b0, a, 7'b0 } : 32'b0;

stored[8] <= b[8] ? { 8'b0, a, 8'b0 } : 32'b0;

stored[9] <= b[9] ? { 7'b0, a, 9'b0 } : 32'b0;

stored[10] <= b[10] ? { 6'b0, a, 10'b0 } : 32'b0;

stored[11] <= b[11] ? { 5'b0, a, 11'b0 } : 32'b0;

stored[12] <= b[12] ? { 4'b0, a, 12'b0 } : 32'b0;

stored[13] <= b[13] ? { 3'b0, a, 13'b0 } : 32'b0;

stored[14] <= b[14] ? { 2'b0, a, 14'b0 } : 32'b0;

stored[15] <= b[15] ? { 1'b0, a, 15'b0 } : 32'b0;

add1[0] <= stored[0] + stored[1];

add1[1] <= stored[2] + stored[3];

add1[2] <= stored[4] + stored[5];

add1[3] <= stored[6] + stored[7];

add1[4] <= stored[8] + stored[9];

add1[5] <= stored[10] + stored[11];

add1[6] <= stored[12] + stored[13];

add1[7] <= stored[14] + stored[15];

add2[0] <= add1[0] + add1[1];

add2[1] <= add1[2] + add1[3];

add2[2] <= add1[4] + add1[5];

add2[3] <= add1[6] + add1[7];

add3[0] <= add2[0] + add2[1];

add3[1] <= add2[2] + add2[3];

result <= add3[0] + add3[1];

end

end

assign z = result;

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 2023/05/22 21:58:18

// Design Name:

// Module Name: MULTUtest

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module MULTUtest;

reg clk, reset;

reg[15:0] a, b;

wire[31:0] z;

MULTU MULTU(

.clk(clk),

.reset(reset),

.a(a),

.b(b),

.z(z)

);

initial begin

clk = 0;

reset = 0;

a = 16'h7;

b = 16'h8;

#100

a = 16'h9;

b = 16'h9;

#100

a = 16'h3;

b = 16'h2;

#100

a = 16'h1;

b = 16'h1;

end

always #5 clk = ~clk;

endmodule