同济大学计算机系

计算机组成原理实验报告

学　　号	2351753
姓　　名	黄保翔
专　　业	计算机科学与技术
授课老师	张冬冬

一、实验内容

实现 32 位无符号乘法器和 32 位带符号乘法器

二、模块建模

1、32 位无符号乘法器

两个二进制数 a 和 b 相乘，可以认为是 a 和 b 的每一位相乘后移位后的结果相加。 关于 a 与 b 的每一位相乘产生的中间结果，如果 b 那位是 0，那么中间结果就是 0；如果是 1，那么中间结果就是在 a 前后补上相应位数的零通过字符拼接的方式表示。然后将这些中间乘积相加就是最后的结果

代码:

```
module MULTU(
    input clk, //乘法器时钟信号
    input reset, //复位信号，高电平有效
    input [31:0] a, //输入数 a(被乘数)
    input [31:0] b, //输入数 b（乘数）
    output [63:0] z //乘积输出 z
);

wire [63:0] temp;
// 申请寄存器
reg [63:0] stored0;
reg [63:0] stored1;
reg [63:0] stored2;
reg [63:0] stored3;
reg [63:0] stored4;
reg [63:0] stored5;
reg [63:0] stored6;
reg [63:0] stored7;
reg [63:0] stored8;
reg [63:0] stored9;
reg [63:0] stored10;
reg [63:0] stored11;
reg [63:0] stored12;
reg [63:0] stored13;
reg [63:0] stored14;
reg [63:0] stored15;
reg [63:0] stored16;
reg [63:0] stored17;
reg [63:0] stored18;
reg [63:0] stored19;
reg [63:0] stored20;
reg [63:0] stored21;
reg [63:0] stored22;
reg [63:0] stored23;
reg [63:0] stored24;
reg [63:0] stored25;
reg [63:0] stored26;
reg [63:0] stored27;
reg [63:0] stored28;
reg [63:0] stored29;
reg [63:0] stored30;
reg [63:0] stored31;

wire [63:0] add0_1;
wire [63:0] add2_3;
wire [63:0] add4_5;
wire [63:0] add6_7;
wire [63:0] add8_9;
wire [63:0] add10_11;
wire [63:0] add12_13;
wire [63:0] add14_15;
wire [63:0] add16_17;
wire [63:0] add18_19;
wire [63:0] add20_21;
wire [63:0] add22_23;
wire [63:0] add24_25;
wire [63:0] add26_27;
wire [63:0] add28_29;
wire [63:0] add30_31;

wire [63:0] add0t1_2t3;
wire [63:0] add4t5_6t7;
wire [63:0] add8t9_10t11;
wire [63:0] add12t13_14t15;
wire [63:0] add16t17_18t19;
wire [63:0] add20t21_22t23;
wire [63:0] add24t25_26t27;
wire [63:0] add28t29_30t31;

wire [63:0] add0t3_4t7;
wire [63:0] add8t11_12t15;
wire [63:0] add16t19_20t23;
wire [63:0] add24t27_28t31;
```

```verilog
wire [63:0] add0t7_8t15;
wire [63:0] add16t23_24t31;

assign z = temp;
assign add0_1 = stored0 + stored1;
assign add2_3 = stored2 + stored3;
assign add4_5 = stored4 + stored5;
assign add6_7 = stored6 + stored7;
assign add8_9 = stored8 + stored9;
assign add10_11 = stored10 + stored11;
assign add12_13 = stored12 + stored13;
assign add14_15 = stored14 + stored15;
assign add16_17 = stored16 + stored17;
assign add18_19 = stored18 + stored19;
assign add20_21 = stored20 + stored21;
assign add22_23 = stored22 + stored23;
assign add24_25 = stored24 + stored25;
assign add26_27 = stored26 + stored27;
assign add28_29 = stored28 + stored29;
assign add30_31 = stored30 + stored31;

assign add0t1_2t3 = add0_1 + add2_3;
assign add4t5_6t7 = add4_5 + add6_7;
assign add8t9_10t11 = add8_9 + add10_11;
assign add12t13_14t15 = add12_13 + add14_15;
assign add16t17_18t19 = add16_17 + add18_19;
assign add20t21_22t23 = add20_21 + add22_23;
assign add24t25_26t27 = add24_25 + add26_27;
assign add28t29_30t31 = add28_29 + add30_31;

assign add0t3_4t7 = add0t1_2t3 + add4t5_6t7;
assign add8t11_12t15 = add8t9_10t11 + add12t13_14t15;
assign add16t19_20t23 = add16t17_18t19 + add20t21_22t23;
assign add24t27_28t31 = add24t25_26t27 + add28t29_30t31;

assign add0t7_8t15 = add0t3_4t7 + add8t11_12t15;
assign add16t23_24t31 = add16t19_20t23 + add24t27_28t31;

assign temp = add0t7_8t15 + add16t23_24t31;

always @(posedge clk or posedge reset)
begin
    // reset 置零
    if(reset == 1)
    begin
        stored0 <= 0;
        stored1 <= 0;
        stored2 <= 0;
        stored3 <= 0;
        stored4 <= 0;
        stored5 <= 0;
        stored6 <= 0;
        stored7 <= 0;
        stored8 <= 0;
        stored9 <= 0;
        stored10 <= 0;
        stored11 <= 0;
        stored12 <= 0;
        stored13 <= 0;
        stored14 <= 0;
        stored15 <= 0;
        stored16 <= 0;
        stored17 <= 0;
        stored18 <= 0;
        stored19 <= 0;
        stored20 <= 0;
        stored21 <= 0;
        stored22 <= 0;
        stored23 <= 0;
        stored24 <= 0;
        stored25 <= 0;
        stored26 <= 0;
        stored27 <= 0;
        stored28 <= 0;
        stored29 <= 0;
        stored30 <= 0;
        stored31 <= 0;
    end
    else
    begin
        //通过字符拼接方式表示出中间相乘值，并相加
        stored0 <= b[0]? {32'b0, a} :64'b0;
        stored1 <= b[1]? {31'b0, a, 1'b0} :64'b0;
        stored2 <= b[2]? {30'b0, a, 2'b0} :64'b0;
        stored3 <= b[3]? {29'b0, a, 3'b0} :64'b0;
        stored4 <= b[4]? {28'b0, a, 4'b0} :64'b0;
        stored5 <= b[5]? {27'b0, a, 5'b0} :64'b0;
        stored6 <= b[6]? {26'b0, a, 6'b0} :64'b0;
        stored7 <= b[7]? {25'b0, a, 7'b0} :64'b0;
        stored8 <= b[8]? {24'b0, a, 8'b0} :64'b0;
        stored9 <= b[9]? {23'b0, a, 9'b0} :64'b0;
        stored10 <= b[10]? {22'b0, a, 10'b0} :64'b0;
        stored11 <= b[11]? {21'b0, a, 11'b0} :64'b0;
        stored12 <= b[12]? {20'b0, a, 12'b0} :64'b0;
        stored13 <= b[13]? {19'b0, a, 13'b0} :64'b0;
        stored14 <= b[14]? {18'b0, a, 14'b0} :64'b0;
        stored15 <= b[15]? {17'b0, a, 15'b0} :64'b0;
        stored16 <= b[16]? {16'b0, a, 16'b0} :64'b0;
        stored17 <= b[17]? {15'b0, a, 17'b0} :64'b0;
        stored18 <= b[18]? {14'b0, a, 18'b0} :64'b0;
        stored19 <= b[19]? {13'b0, a, 19'b0} :64'b0;
        stored20 <= b[20]? {12'b0, a, 20'b0} :64'b0;
        stored21 <= b[21]? {11'b0, a, 21'b0} :64'b0;
        stored22 <= b[22]? {10'b0, a, 22'b0} :64'b0;
        stored23 <= b[23]? {9'b0, a, 23'b0} :64'b0;
        stored24 <= b[24]? {8'b0, a, 24'b0} :64'b0;
        stored25 <= b[25]? {7'b0, a, 25'b0} :64'b0;
        stored26 <= b[26]? {6'b0, a, 26'b0} :64'b0;
        stored27 <= b[27]? {5'b0, a, 27'b0} :64'b0;
        stored28 <= b[28]? {4'b0, a, 28'b0} :64'b0;
        stored29 <= b[29]? {3'b0, a, 29'b0} :64'b0;
        stored30 <= b[30]? {2'b0, a, 30'b0} :64'b0;
        stored31 <= b[31]? {1'b0, a, 31'b0} :64'b0;
    end
end
endmodule
```

2、32 位有符号乘法器

同 32 为无符号加法器计算思路，在计算之前先判断 a,b 符号，将其取绝对值存储计算，将计算后得到的绝对值根据 a,b 符号异或结果重新添加符号，若 a,b 异号，则进行取补码运算，否则直接输出

```
module MULT(
    input clk,
    input reset,
    input [31:0] a,
    input [31:0] b,
    output [63:0] z
);

wire [63:0] temp;
// 申请寄存器
reg [31:0] temp_a;
reg [31:0] temp_b;
reg [63:0] stored0;
reg [63:0] stored1;
reg [63:0] stored2;
reg [63:0] stored3;
reg [63:0] stored4;
reg [63:0] stored5;
reg [63:0] stored6;
reg [63:0] stored7;
reg [63:0] stored8;
reg [63:0] stored9;
reg [63:0] stored10;
reg [63:0] stored11;
reg [63:0] stored12;
reg [63:0] stored13;
reg [63:0] stored14;
reg [63:0] stored15;
reg [63:0] stored16;
reg [63:0] stored17;
reg [63:0] stored18;
reg [63:0] stored19;
reg [63:0] stored20;
reg [63:0] stored21;
reg [63:0] stored22;
reg [63:0] stored23;
reg [63:0] stored24;
reg [63:0] stored25;
reg [63:0] stored26;
reg [63:0] stored27;
reg [63:0] stored28;
reg [63:0] stored29;
reg [63:0] stored30;
reg [63:0] stored31;

wire [63:0] add0_1;
wire [63:0] add2_3;
wire [63:0] add4_5;
wire [63:0] add6_7;
wire [63:0] add8_9;
wire [63:0] add10_11;
wire [63:0] add12_13;
wire [63:0] add14_15;
wire [63:0] add16_17;
wire [63:0] add18_19;
```

```verilog
wire [63:0] add20_21;
wire [63:0] add22_23;
wire [63:0] add24_25;
wire [63:0] add26_27;
wire [63:0] add28_29;
wire [63:0] add30_31;

wire [63:0] add0t1_2t3;
wire [63:0] add4t5_6t7;
wire [63:0] add8t9_10t11;
wire [63:0] add12t13_14t15;
wire [63:0] add16t17_18t19;
wire [63:0] add20t21_22t23;
wire [63:0] add24t25_26t27;
wire [63:0] add28t29_30t31;

wire [63:0] add0t3_4t7;
wire [63:0] add8t11_12t15;
wire [63:0] add16t19_20t23;
wire [63:0] add24t27_28t31;

wire [63:0] add0t7_8t15;
wire [63:0] add16t23_24t31;

assign z = a[31]^b[31] ? ~temp + 1'b1 : temp; // 乘法结果的符号位由 a 和
b 的符号位异或决定
assign add0_1 = stored0 + stored1;
assign add2_3 = stored2 + stored3;
assign add4_5 = stored4 + stored5;
assign add6_7 = stored6 + stored7;
assign add8_9 = stored8 + stored9;
assign add10_11 = stored10 + stored11;
assign add12_13 = stored12 + stored13;
assign add14_15 = stored14 + stored15;
assign add16_17 = stored16 + stored17;
assign add18_19 = stored18 + stored19;
assign add20_21 = stored20 + stored21;
assign add22_23 = stored22 + stored23;
assign add24_25 = stored24 + stored25;
assign add26_27 = stored26 + stored27;
assign add28_29 = stored28 + stored29;
assign add30_31 = stored30 + stored31;

assign add0t1_2t3 = add0_1 + add2_3;
assign add4t5_6t7 = add4_5 + add6_7;
assign add8t9_10t11 = add8_9 + add10_11;
assign add12t13_14t15 = add12_13 + add14_15;
assign add16t17_18t19 = add16_17 + add18_19;
assign add20t21_22t23 = add20_21 + add22_23;
assign add24t25_26t27 = add24_25 + add26_27;
assign add28t29_30t31 = add28_29 + add30_31;

assign add0t3_4t7 = add0t1_2t3 + add4t5_6t7;
assign add8t11_12t15 = add8t9_10t11 + add12t13_14t15;
assign add16t19_20t23 = add16t17_18t19 + add20t21_22t23;
assign add24t27_28t31 = add24t25_26t27 + add28t29_30t31;

assign add0t7_8t15 = add0t3_4t7 + add8t11_12t15;
assign add16t23_24t31 = add16t19_20t23 + add24t27_28t31;

assign temp = add0t7_8t15 + add16t23_24t31;

always @(posedge clk or posedge reset)
begin
    // reset 置零
    if(reset == 1)
    begin
        temp_a <= 0;
        temp_b <= 0;
        stored0 <= 0;
        stored1 <= 0;
        stored2 <= 0;
        stored3 <= 0;
        stored4 <= 0;
        stored5 <= 0;
        stored6 <= 0;
        stored7 <= 0;
        stored8 <= 0;
        stored9 <= 0;
        stored10 <= 0;
        stored11 <= 0;
        stored12 <= 0;
        stored13 <= 0;
        stored14 <= 0;
        stored15 <= 0;
        stored16 <= 0;
        stored17 <= 0;
        stored18 <= 0;
        stored19 <= 0;
        stored20 <= 0;
        stored21 <= 0;
        stored22 <= 0;
        stored23 <= 0;
        stored24 <= 0;
        stored25 <= 0;
        stored26 <= 0;
        stored27 <= 0;
        stored28 <= 0;
        stored29 <= 0;
        stored30 <= 0;
        stored31 <= 0;
    end
    else
    begin
        case({a[31], b[31]})
            2'b00 : begin
                    temp_a <= a;
                    temp_b <= b; // 正数不变
            end
            2'b01 : begin
                    temp_a <= a; // 正数不变
                    temp_b <= ~(b - 1'b1); // 负数补码
            end
            2'b10 : begin
                    temp_a <= ~(a - 1'b1); // 负数补码
                    temp_b <= b; // 正数不变
            end
            2'b11 : begin
                    temp_a <= ~(a - 1'b1); // 负数补码
                    temp_b <= ~(b - 1'b1); // 负数补码
            end
        endcase
        //通过字符拼接方式表示出中间相乘值，并相加

        stored0 <= temp_b[0]? {32'b0, temp_a} :64'b0;
        stored1 <= temp_b[1]? {31'b0, temp_a, 1'b0} :64'b0;
        stored2 <= temp_b[2]? {30'b0, temp_a, 2'b0} :64'b0;
        stored3 <= temp_b[3]? {29'b0, temp_a, 3'b0} :64'b0;
        stored4 <= temp_b[4]? {28'b0, temp_a, 4'b0} :64'b0;
        stored5 <= temp_b[5]? {27'b0, temp_a, 5'b0} :64'b0;
        stored6 <= temp_b[6]? {26'b0, temp_a, 6'b0} :64'b0;
        stored7 <= temp_b[7]? {25'b0, temp_a, 7'b0} :64'b0;
        stored8 <= temp_b[8]? {24'b0, temp_a, 8'b0} :64'b0;
        stored9 <= temp_b[9]? {23'b0, temp_a, 9'b0} :64'b0;
        stored10 <= temp_b[10]? {22'b0, temp_a, 10'b0} :64'b0;
        stored11 <= temp_b[11]? {21'b0, temp_a, 11'b0} :64'b0;
        stored12 <= temp_b[12]? {20'b0, temp_a, 12'b0} :64'b0;
        stored13 <= temp_b[13]? {19'b0, temp_a, 13'b0} :64'b0;
        stored14 <= temp_b[14]? {18'b0, temp_a, 14'b0} :64'b0;
        stored15 <= temp_b[15]? {17'b0, temp_a, 15'b0} :64'b0;
        stored16 <= temp_b[16]? {16'b0, temp_a, 16'b0} :64'b0;
        stored17 <= temp_b[17]? {15'b0, temp_a, 17'b0} :64'b0;
        stored18 <= temp_b[18]? {14'b0, temp_a, 18'b0} :64'b0;
        stored19 <= temp_b[19]? {13'b0, temp_a, 19'b0} :64'b0;
        stored20 <= temp_b[20]? {12'b0, temp_a, 20'b0} :64'b0;
        stored21 <= temp_b[21]? {11'b0, temp_a, 21'b0} :64'b0;
        stored22 <= temp_b[22]? {10'b0, temp_a, 22'b0} :64'b0;
        stored23 <= temp_b[23]? {9'b0, temp_a, 23'b0} :64'b0;
        stored24 <= temp_b[24]? {8'b0, temp_a, 24'b0} :64'b0;
        stored25 <= temp_b[25]? {7'b0, temp_a, 25'b0} :64'b0;
        stored26 <= temp_b[26]? {6'b0, temp_a, 26'b0} :64'b0;
        stored27 <= temp_b[27]? {5'b0, temp_a, 27'b0} :64'b0;
        stored28 <= temp_b[28]? {4'b0, temp_a, 28'b0} :64'b0;
        stored29 <= temp_b[29]? {3'b0, temp_a, 29'b0} :64'b0;
        stored30 <= temp_b[30]? {2'b0, temp_a, 30'b0} :64'b0;
        stored31 <= temp_b[31]? {1'b0, temp_a, 31'b0} :64'b0;
    end
end

endmodule
```

三、测试模块建模

1、无符号乘法器

```
module MULTU_tb;
    reg [31:0] a, b;
    wire [63:0] product;
    reg clk;
    reg reset;
    // 实例化 MULTU 模块
    MULTU uut (
        .clk(clk), // 时钟信号
        .reset(reset), // 复位信号
        .a(a),
        .b(b),
        .z(product)
    );

    initial begin
        clk = 1'b0;
        forever #5 clk = ~clk;
    end

    initial begin
        // 复位信号置位
        reset = 1'b1;
        #10;
        reset = 1'b0;
        #10;
        a = 32'b11111111111111111111111111111111; b =
32'b11111111111111111111111111111111;
        #10;
        // 测试用例
        a = 32'b00000000000000000000000000000000; b =
32'b00000000000000000000000000000000;
        #10;
        a = 32'b00000000000000000000000010110011; b =
32'b00000000000000000000000000000000;
        #10;
        a = 32'b00000000000000000000000011111111; b =
32'b00000000000000000000000011111111;
        #10;
        a = 32'b00000000000000000000000010000000; b =
32'b00000000000000000000000010101010;
        #10;
        a = 32'b00000000000000000000000010101010; b =
32'b00000000000000000000000010000000;
        #10;
        a = 32'b00000000000000000000000000101101; b =
32'b00000000000000000000000011010000;
        #10;
        a = 32'b0000000000000000000000001000111; b =
32'b00000000000000000000000000001110;
        #10;

    end
```

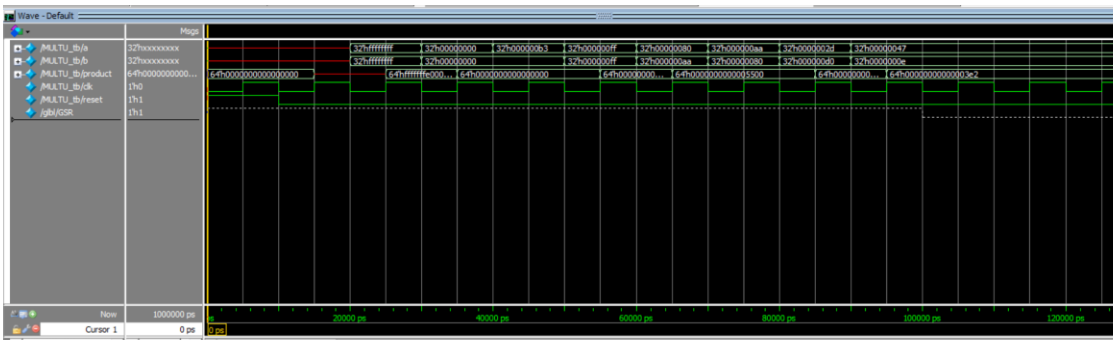
```
endmodule
```

2、有符号乘法器

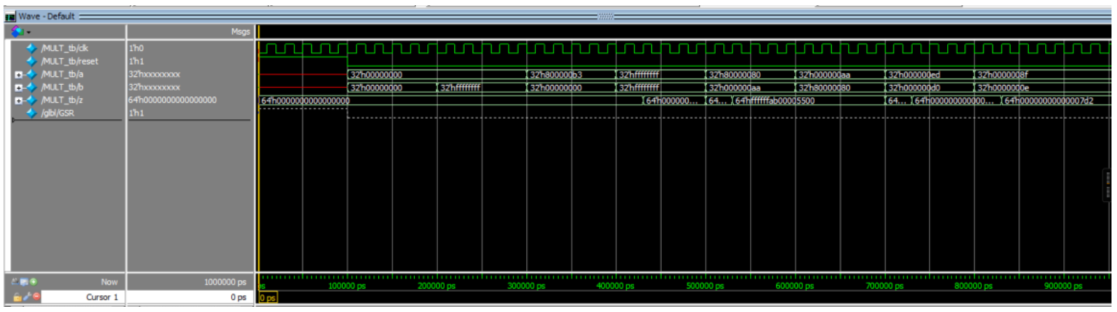
```
`timescale 1ns / 1ps
module MULT_tb;
    reg clk; // 乘法器时钟信号
    reg reset; // 复位信号，高电平有效
    reg [31:0] a; // 输入 a (被乘数)
    reg [31:0] b; // 输入 b (乘数)
    wire [63:0] z; // 乘积输出 z

    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end

    initial begin
        reset = 1;
        #100;
        reset = 0;
        a = 0;
        b = 0;

        #100;
        a = 0;
        b = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
        #100;

        a = 32'b1000_0000_0000_0000_0000_0000_1011_001
        b = 0;
        #100;

        a = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
        b = 32'b1111_1111_1111_1111_1111_1111_1111_1111;  //等于 1
        #100;

        a = 32'b1000_0000_0000_0000_0000_0000_1000_0000;
        b = 32'b0000_0000_0000_0000_0000_0000_1010_1010;
        #100;

        a = 32'b0000_0000_0000_0000_0000_0000_1010_1010;
        #100;
        a = 32'b0000_0000_0000_0000_0000_0000_1110_1101;
        b = 32'b0000_0000_0000_0000_0000_0000_1101_0000;
        #100;

        a = 32'b0000_0000_0000_0000_0000_0000_1000_1111;
        b = 32'b0000_0000_0000_0000_0000_0000_0000_1110;
    end

    MULT uut (
        .clk(clk),
        .reset(reset),
        .a(a),
        .b(b),
        .z(z)
    );
endmodule
```

四、实验结果

1、 无符号乘法器



2、 有符号乘法器


Wave - Default
Msgs
/MULT_tb/clk 1'h0
/MULT_tb/reset 1'h1
/MULT_tb/a 32'hxxxxxxxx
/MULT_tb/b 32'hxxxxxxxx
/MULT_tb/z 64'h0000000000000000
/glbl/GSR 1'h1
32'h00000000
32'h800000b3
32'hffffffff
32'h80000080
32'h000000aa
32'h000000ed
32'h0000008f
32'h000000d0
32'h0000000e
64'h0000000000000000
64'hffffffab00005500
64'h00000000000007d2
Now 1000000 ps
Cursor 1 0 ps
100000 ps
200000 ps
300000 ps
400000 ps
500000 ps
600000 ps
700000 ps
800000 ps
900000 ps