同济大学计算机系
计算机组成原理实验报告
32 位乘法器实验

1. 实验介绍

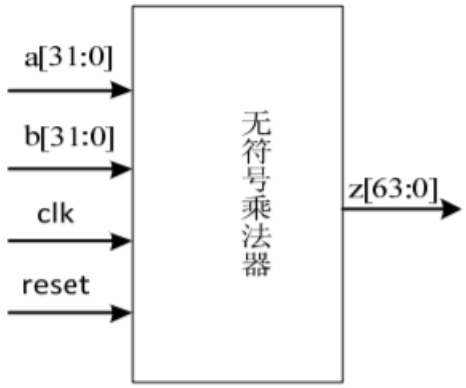
通过本次试验，了解乘法器的实现原理，并学习如何实现一个乘法器，本实验将实现32 位无符号乘法器和 32 位带符号乘法器。

2. 实验目标

- 了解 32 位带符号、无符号乘法器的实现原理
- 使用 Verilog 实现 32 位无符号乘法器和带符号乘法器

3. 实验原理

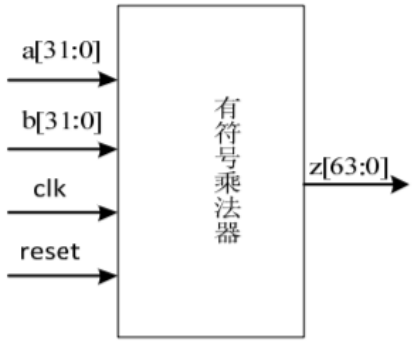
1）无符号乘法器功能为：将两个 32 位无符号数相乘，得到一个 64 位无符号数。



接口定义

```
module MULTU(
    input clk,          //乘法器时钟信号
    input reset,        //复位信号，低电平有效
    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
);
```

2) 带符号乘法器功能为：将两个 32 带无符号数相乘，得到一个 64 位带符号数



接口定义

```
module MULT(
    input clk,          //乘法器时钟信号
    input reset,        //复位信号，低电平有效
    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
);
```

4.实验步骤

1. 新建 Vivado 工程
2. 编写各个模块
3. 用 ModelSim 仿真测试各模块

5.模块建模

1）MULTU

功能描述：

作为 32 位无符号乘法器使用，输入 clk、reset、[31:0]a、[31:0],输出[63:0]z。其中 clk 为时钟信号，reset 为复位信号，[31:0]a 为被乘数，[31:0]b 为乘数，[63:0]z 为乘积输出。

Verilog 代码：

```
module MULTU(
    input clk,
    input reset,
    input [31:0]a,
    input [31:0]b,
    output [63:0]z
    );
    // 申请寄存器
```

```
reg [63:0] temp;
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;
always @(posedge clk or negedge reset)
begin
    // reset 置零
    if(reset)
    begin
        temp <= 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
    //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} : 63'b0;
    stored1 <= b[1]? {31'b0, a, 1'b0} :63'b0;
    stored2 <= b[2]? {30'b0, a, 2'b0} :63'b0;
```

```
            stored3 <= b[3]? {29'b0, a, 3'b0} :63'b0;
            stored4 <= b[4]? {28'b0, a, 4'b0} :63'b0;
            stored5 <= b[5]? {27'b0, a, 5'b0} :63'b0;
            stored6 <= b[6]? {26'b0, a, 6'b0} :63'b0;
            stored7 <= b[7]? {25'b0, a, 7'b0} :63'b0;
            stored8 <= b[8]? {24'b0, a, 8'b0} :63'b0;
            stored9 <= b[9]? {23'b0, a, 9'b0} :63'b0;
            stored10 <= b[10]? {22'b0, a, 10'b0} :63'b0;
            stored11 <= b[11]? {21'b0, a, 11'b0} :63'b0;
            stored12 <= b[12]? {20'b0, a, 12'b0} :63'b0;
            stored13 <= b[13]? {19'b0, a, 13'b0} :63'b0;
            stored14 <= b[14]? {18'b0, a, 14'b0} :63'b0;
            stored15 <= b[15]? {17'b0, a, 15'b0} :63'b0;
            stored16 <= b[16]? {16'b0, a, 16'b0} :63'b0;
            stored17 <= b[17]? {15'b0, a, 17'b0} :63'b0;
            stored18 <= b[18]? {14'b0, a, 18'b0} :63'b0;
            stored19 <= b[19]? {13'b0, a, 19'b0} :63'b0;
            stored20 <= b[20]? {12'b0, a, 20'b0} :63'b0;
            stored21 <= b[21]? {11'b0, a, 21'b0} :63'b0;
            stored22 <= b[22]? {10'b0, a, 22'b0} :63'b0;
            stored23 <= b[23]? {9'b0, a, 23'b0} :63'b0;
            stored24 <= b[24]? {8'b0, a, 24'b0} :63'b0;
            stored25 <= b[25]? {7'b0, a, 25'b0} :63'b0;
            stored26 <= b[26]? {6'b0, a, 26'b0} :63'b0;
            stored27 <= b[27]? {5'b0, a, 27'b0} :63'b0;
            stored28 <= b[28]? {4'b0, a, 28'b0} :63'b0;
            stored29 <= b[29]? {3'b0, a, 29'b0} :63'b0;
            stored30 <= b[30]? {2'b0, a, 30'b0} :63'b0;
            stored31 <= b[31]? {1'b0, a, 31'b0} :63'b0;

temp=stored0+stored1+stored2+stored3+stored4+stored5+stored6+stored7+stored8+st
ored9+

stored10+stored11+stored12+stored13+stored14+stored15+stored16+stored17+stored1
8+stored19+

stored20+stored21+stored22+stored23+stored24+stored25+stored26+stored27+stored2
8+stored29+
            stored30+stored31;
```

```
        end
    end
    assign z = temp;
endmodule
```

2）MULT

功能描述：

作为 32 位带符号乘法器使用，输入 clk、reset、[31:0]a、[31:0],输出[63:0]z。其中 clk 为时钟信号，reset 为复位信号，[31:0]a 为被乘数，[31:0]b 为乘数，[63:0]z 为乘积输出。

Verilog 代码：

```
module MULT(
    input clk,
    input reset,
    input [31:0]a,
    input [31:0]b,
    output [63:0]z
    );
    // 申请寄存器
    reg [63:0] temp;
    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 [31:0] a_inv;
assign a_inv = ~a + 1;
always @(posedge clk or negedge reset)
begin
    // reset 置零
    if(reset)
    begin
        temp <= 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
    //通过字符拼接方式表示出中间相乘值，并相加
    stored0 <= b[0]? {{32{a[31]}}, a} : 63'b0;
    stored1 <= b[1]? {{31{a[31]}}, a, 1'b0} :63'b0;
    stored2 <= b[2]? {{30{a[31]}}, a, 2'b0} :63'b0;
    stored3 <= b[3]? {{29{a[31]}}, a, 3'b0} :63'b0;
    stored4 <= b[4]? {{28{a[31]}}, a, 4'b0} :63'b0;
    stored5 <= b[5]? {{27{a[31]}}, a, 5'b0} :63'b0;
    stored6 <= b[6]? {{26{a[31]}}, a, 6'b0} :63'b0;
    stored7 <= b[7]? {{25{a[31]}}, a, 7'b0} :63'b0;
    stored8 <= b[8]? {{24{a[31]}}, a, 8'b0} :63'b0;
    stored9 <= b[9]? {{23{a[31]}}, a, 9'b0} :63'b0;
    stored10 <= b[10]? {{22{a[31]}}, a, 10'b0} :63'b0;
    stored11 <= b[11]? {{21{a[31]}}, a, 11'b0} :63'b0;
    stored12 <= b[12]? {{20{a[31]}}, a, 12'b0} :63'b0;
    stored13 <= b[13]? {{19{a[31]}}, a, 13'b0} :63'b0;
    stored14 <= b[14]? {{18{a[31]}}, a, 14'b0} :63'b0;
    stored15 <= b[15]? {{17{a[31]}}, a, 15'b0} :63'b0;
    stored16 <= b[16]? {{16{a[31]}}, a, 16'b0} :63'b0;
    stored17 <= b[17]? {{15{a[31]}}, a, 17'b0} :63'b0;
    stored18 <= b[18]? {{14{a[31]}}, a, 18'b0} :63'b0;
    stored19 <= b[19]? {{13{a[31]}}, a, 19'b0} :63'b0;
    stored20 <= b[20]? {{12{a[31]}}, a, 20'b0} :63'b0;
```

```
            stored21 <= b[21]? {{11{a[31]}}, a, 21'b0} :63'b0;
            stored22 <= b[22]? {{10{a[31]}}, a, 22'b0} :63'b0;
            stored23 <= b[23]? {{9{a[31]}}, a, 23'b0} :63'b0;
            stored24 <= b[24]? {{8{a[31]}}, a, 24'b0} :63'b0;
            stored25 <= b[25]? {{7{a[31]}}, a, 25'b0} :63'b0;
            stored26 <= b[26]? {{6{a[31]}}, a, 26'b0} :63'b0;
            stored27 <= b[27]? {{5{a[31]}}, a, 27'b0} :63'b0;
            stored28 <= b[28]? {{4{a[31]}}, a, 28'b0} :63'b0;
            stored29 <= b[29]? {{3{a[31]}}, a, 29'b0} :63'b0;
            stored30 <= b[30]? {{2{a[31]}}, a, 30'b0} :63'b0;
            stored31 <= b[31]? {{1{a_inv[31]}}, a_inv, 31'b0} :63'b0;

temp=stored0+stored1+stored2+stored3+stored4+stored5+stored6+stored7+stored8+st
ored9+

stored10+stored11+stored12+stored13+stored14+stored15+stored16+stored17+stored1
8+stored19+

stored20+stored21+stored22+stored23+stored24+stored25+stored26+stored27+stored2
8+stored29+
            stored30+stored31;
        end
    end
    assign z = temp;
endmodule
```

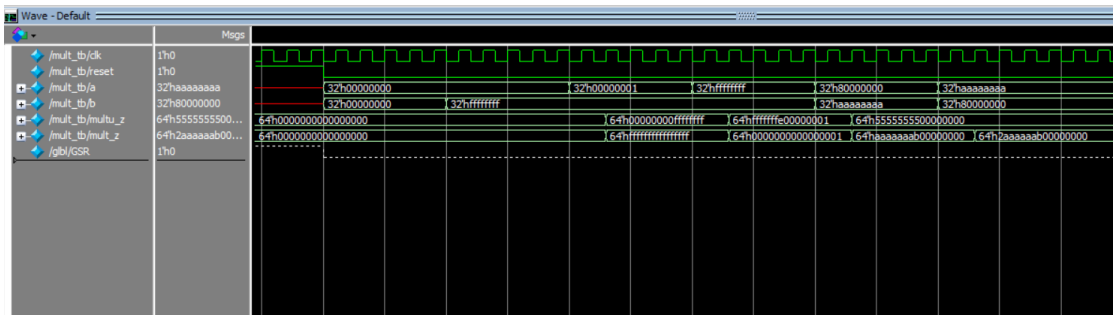
6.测试模块建模

```
`timescale 1ns / 1ps
module mult_tb;
    reg clk;
    reg reset;
    reg [31:0]a;
    reg [31:0]b;
    wire [63:0]multu_z;
    wire [63:0]mult_z;
    MULTU U1(.clk(clk),.reset(reset),.a(a),.b(b),.z(multu_z));
    MULT U2(.clk(clk),.reset(reset),.a(a),.b(b),.z(mult_z));
    initial
```

```
    begin
        reset = 1;
        clk = 0;
        # 100
        reset = 0;
        a = 0;
        b = 0;
        # 100
        a = 0;
        b = 32'b11111111_11111111_11111111_11111111;
        # 100
        a = 32'b1;
        b = 32'b11111111_11111111_11111111_11111111;
        # 100
        a = 32'b11111111_11111111_11111111_11111111;
        b = 32'b11111111_11111111_11111111_11111111;
        # 100
        a = 32'b10000000_00000000_00000000_00000000;
        b = 32'b10101010_10101010_10101010_10101010;
        # 100
        a = 32'b10101010_10101010_10101010_10101010;
        b = 32'b10000000_00000000_00000000_00000000;
    end
    always
    #10 clk<=~clk;
endmodule
```

7. 实验结果

modelsim 仿真波形图：



可以看出：

1) a = 0

b = 0

multu_z = 0

mult_z = 0

2) a = 0

b = 32'b11111111_11111111_11111111_11111111

multu_z = 0

mult_z = 0

3) a = 1

b = 32'b11111111_11111111_11111111_11111111

multu_z = 64'h00000000_ffffffff

mult_z = 64'hffffffff_ffffffff

4) a = 32'b11111111_11111111_11111111_11111111

b = 32'b11111111_11111111_11111111_11111111

multu_z = 64'hfffffffe_00000001

mult_z = 64'h00000000_00000001

5) a = 32'b10000000_00000000_00000000_00000000;

b = 32'b10101010_10101010_10101010_10101010;

multu_z = 64'h55555555_00000000

mult_z = 64'haaaaaaab_00000000

6) a = 32'b10101010_10101010_10101010_10101010

b = 32'b10000000_00000000_00000000_00000000

multu_z = 64'h55555555_00000000

mult_z = 64'h2aaaaaab_00000000

上述测试结果无误，因此 32 位乘法器设计完成。