同济大学
计算机科学与技术系

计算机组成原理实验报告

学　　号　**1652228**

姓　　名　王哲源

专　　业　计算机科学与技术

授课老师　陈永生

日　　期　**2018.3.24**

一、实验目标

通过 verilog 语言实现 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

   MULTU 模块实现对 32 位无符号数 a、b 进行乘法，并将乘积传给 z

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

    reg [63:0] tmp0;
    reg [63:0] tmp1;
    reg [63:0] tmp2;
    reg [63:0] tmp3;
    reg [63:0] tmp4;
    reg [63:0] tmp5;
    reg [63:0] tmp6;
    reg [63:0] tmp7;
    reg [63:0] tmp8;
    reg [63:0] tmp9;
    reg [63:0] tmp10;
    reg [63:0] tmp11;
    reg [63:0] tmp12;
    reg [63:0] tmp13;
    reg [63:0] tmp14;
    reg [63:0] tmp15;
    reg [63:0] tmp16;
    reg [63:0] tmp17;
    reg [63:0] tmp18;
    reg [63:0] tmp19;
    reg [63:0] tmp20;
    reg [63:0] tmp21;
    reg [63:0] tmp22;
    reg [63:0] tmp23;
    reg [63:0] tmp24;
    reg [63:0] tmp25;
    reg [63:0] tmp26;
    reg [63:0] tmp27;
    reg [63:0] tmp28;
    reg [63:0] tmp29;
    reg [63:0] tmp30;
    reg [63:0] tmp31;

    wire [63:0] tans_0_1, tans_2_3, tans_4_5, tans_6_7, tans_8_9, tans_10_11,
tans_12_13, tans_14_15;
    wire [63:0] tans_16_17, tans_18_19, tans_20_21, tans_22_23, tans_24_25,
tans_26_27, tans_28_29, tans_30_31;
    wire [63:0] tans_01_23, tans_45_67, tans_89_1011, tans_1213_1415,
tans_1617_1819, tans_2021_2223, tans_2425_2627, tans_2829_3031;
    wire [63:0] tans_0123_4567, tans_891011_12131415,
tans_16171819_20212223, tans_24252627_28293031;
    wire [63:0] tans_012345678_9101112131415,
tans_1617181920212223_2425262728293031;

    assign tans_0_1 = tmp0 + tmp1;
    assign tans_2_3 = tmp2 + tmp3;
    assign tans_4_5 = tmp4 + tmp5;
    assign tans_6_7 = tmp6 + tmp7;
    assign tans_8_9 = tmp8 + tmp9;
    assign tans_10_11 = tmp10 + tmp11;
    assign tans_12_13 = tmp12 + tmp13;
    assign tans_14_15 = tmp14 + tmp15;
    assign tans_16_17 = tmp16 + tmp17;
    assign tans_18_19 = tmp18 + tmp19;
    assign tans_20_21 = tmp20 + tmp21;
    assign tans_22_23 = tmp22 + tmp23;
    assign tans_24_25 = tmp24 + tmp25;
    assign tans_26_27 = tmp26 + tmp27;
    assign tans_28_29 = tmp28 + tmp29;
    assign tans_30_31 = tmp30 + tmp31;

    assign tans_01_23 = tans_0_1 + tans_2_3;
    assign tans_45_67 = tans_4_5 + tans_6_7;
    assign tans_89_1011 = tans_8_9 + tans_10_11;
    assign tans_1213_1415 = tans_12_13 + tans_14_15;
    assign tans_1617_1819 = tans_16_17 + tans_18_19;
    assign tans_2021_2223 = tans_20_21 + tans_22_23;
    assign tans_2425_2627 = tans_24_25 + tans_26_27;
    assign tans_2829_3031 = tans_28_29 + tans_30_31;
    assign tans_0123_4567 = tans_01_23 + tans_45_67;
    assign tans_891011_12131415 = tans_89_1011 + tans_1213_1415;
    assign tans_16171819_20212223 = tans_1617_1819 + tans_2021_2223;
    assign tans_24252627_28293031 = tans_2425_2627 + tans_2829_3031;

    assign tans_012345678_9101112131415 = tans_0123_4567 +
tans_891011_12131415;
    assign tans_1617181920212223_2425262728293031 =
tans_16171819_20212223 + tans_24252627_28293031;

    assign z = tans_012345678_9101112131415 +
tans_1617181920212223_2425262728293031;

    always @(posedge clk or negedge reset)
    begin
        if(reset == 0)
        begin
            tmp0 <= 'b0; tmp1 <= 'b0; tmp2 <= 'b0; tmp3 <= 'b0;
            tmp4 <= 'b0; tmp5 <= 'b0; tmp6 <= 'b0; tmp7 <= 'b0;
            tmp8 <= 'b0; tmp9 <= 'b0; tmp10 <= 'b0; tmp11 <= 'b0;
            tmp12 <= 'b0; tmp13 <= 'b0; tmp14 <= 'b0; tmp15 <= 'b0;
            tmp16 <= 'b0; tmp17 <= 'b0; tmp18 <= 'b0; tmp19 <= 'b0;
            tmp20 <= 'b0; tmp21 <= 'b0; tmp22 <= 'b0; tmp23 <= 'b0;
            tmp24 <= 'b0; tmp25 <= 'b0; tmp26 <= 'b0; tmp27 <= 'b0;
            tmp28 <= 'b0; tmp29 <= 'b0; tmp30 <= 'b0; tmp31 <= 'b0;
        end
        else
        begin
            tmp0 <= b[0] ? {32'b0, a} : 'b0;
            tmp1 <= b[1] ? {31'b0, a, 1'b0} : 'b0;
            tmp2 <= b[2] ? {30'b0, a, 2'b0} : 'b0;
            tmp3 <= b[3] ? {29'b0, a, 3'b0} : 'b0;
            tmp4 <= b[4] ? {28'b0, a, 4'b0} : 'b0;
            tmp5 <= b[5] ? {27'b0, a, 5'b0} : 'b0;
            tmp6 <= b[6] ? {26'b0, a, 6'b0} : 'b0;
            tmp7 <= b[7] ? {25'b0, a, 7'b0} : 'b0;
            tmp8 <= b[8] ? {24'b0, a, 8'b0} : 'b0;
            tmp9 <= b[9] ? {23'b0, a, 9'b0} : 'b0;
            tmp10 <= b[10] ? {22'b0, a, 10'b0} : 'b0;
            tmp11 <= b[11] ? {21'b0, a, 11'b0} : 'b0;
            tmp12 <= b[12] ? {20'b0, a, 12'b0} : 'b0;
            tmp13 <= b[13] ? {19'b0, a, 13'b0} : 'b0;
            tmp14 <= b[14] ? {18'b0, a, 14'b0} : 'b0;
            tmp15 <= b[15] ? {17'b0, a, 15'b0} : 'b0;
            tmp16 <= b[16] ? {16'b0, a, 16'b0} : 'b0;
            tmp17 <= b[17] ? {15'b0, a, 17'b0} : 'b0;
            tmp18 <= b[18] ? {14'b0, a, 18'b0} : 'b0;
            tmp19 <= b[19] ? {13'b0, a, 19'b0} : 'b0;
            tmp20 <= b[20] ? {12'b0, a, 20'b0} : 'b0;
            tmp21 <= b[21] ? {11'b0, a, 21'b0} : 'b0;
            tmp22 <= b[22] ? {10'b0, a, 22'b0} : 'b0;
            tmp23 <= b[23] ? {9'b0, a, 23'b0} : 'b0;
            tmp24 <= b[24] ? {8'b0, a, 24'b0} : 'b0;
            tmp25 <= b[25] ? {7'b0, a, 25'b0} : 'b0;
            tmp26 <= b[26] ? {6'b0, a, 26'b0} : 'b0;
            tmp27 <= b[27] ? {5'b0, a, 27'b0} : 'b0;
            tmp28 <= b[28] ? {4'b0, a, 28'b0} : 'b0;
            tmp29 <= b[29] ? {3'b0, a, 29'b0} : 'b0;
            tmp30 <= b[30] ? {2'b0, a, 30'b0} : 'b0;
            tmp31 <= b[31] ? {1'b0, a, 31'b0} : 'b0;
        end
    end
endmodule
```

2. 32 位带符号乘法器

MULT 模块实现对 32 位带符号数 a、b 进行乘法，并将乘积传给 z

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

    reg [63:0] tmp0;
    reg [63:0] tmp1;
    reg [63:0] tmp2;
    reg [63:0] tmp3;
    reg [63:0] tmp4;
    reg [63:0] tmp5;
    reg [63:0] tmp6;
    reg [63:0] tmp7;
    reg [63:0] tmp8;
    reg [63:0] tmp9;
    reg [63:0] tmp10;
    reg [63:0] tmp11;
    reg [63:0] tmp12;
    reg [63:0] tmp13;
    reg [63:0] tmp14;
    reg [63:0] tmp15;
    reg [63:0] tmp16;
    reg [63:0] tmp17;
    reg [63:0] tmp18;
    reg [63:0] tmp19;
    reg [63:0] tmp20;
    reg [63:0] tmp21;
    reg [63:0] tmp22;
    reg [63:0] tmp23;
    reg [63:0] tmp24;
    reg [63:0] tmp25;
    reg [63:0] tmp26;
    reg [63:0] tmp27;
    reg [63:0] tmp28;
    reg [63:0] tmp29;
    reg [63:0] tmp30;
    reg [63:0] tmp31;

    wire [63:0] tans_0_1, tans_2_3, tans_4_5, tans_6_7, tans_8_9, tans_10_11,
tans_12_13, tans_14_15;
    wire [63:0] tans_16_17, tans_18_19, tans_20_21, tans_22_23, tans_24_25,
tans_26_27, tans_28_29, tans_30_31;
    wire [63:0] tans_01_23, tans_45_67, tans_89_1011, tans_1213_1415,
tans_1617_1819, tans_2021_2223, tans_2425_2627, tans_2829_3031;
    wire [63:0] tans_0123_4567, tans_891011_12131415,
tans_16171819_20212223, tans_24252627_28293031;
    wire [63:0] tans_012345678_9101112131415,
tans_1617181920212223_2425262728293031;

    assign tans_0_1 = tmp0 + tmp1;
    assign tans_2_3 = tmp2 + tmp3;
    assign tans_4_5 = tmp4 + tmp5;
    assign tans_6_7 = tmp6 + tmp7;
    assign tans_8_9 = tmp8 + tmp9;
    assign tans_10_11 = tmp10 + tmp11;
    assign tans_12_13 = tmp12 + tmp13;
    assign tans_14_15 = tmp14 + tmp15;
    assign tans_16_17 = tmp16 + tmp17;
    assign tans_18_19 = tmp18 + tmp19;
    assign tans_20_21 = tmp20 + tmp21;
    assign tans_22_23 = tmp22 + tmp23;
    assign tans_24_25 = tmp24 + tmp25;
    assign tans_26_27 = tmp26 + tmp27;
    assign tans_28_29 = tmp28 + tmp29;
    assign tans_30_31 = tmp30 - tmp31;

    assign tans_01_23 = tans_0_1 + tans_2_3;
    assign tans_45_67 = tans_4_5 + tans_6_7;
    assign tans_89_1011 = tans_8_9 + tans_10_11;
    assign tans_1213_1415 = tans_12_13 + tans_14_15;
    assign tans_1617_1819 = tans_16_17 + tans_18_19;
    assign tans_2021_2223 = tans_20_21 + tans_22_23;
    assign tans_2425_2627 = tans_24_25 + tans_26_27;
    assign tans_2829_3031 = tans_28_29 + tans_30_31;

    assign tans_0123_4567 = tans_01_23 + tans_45_67;
    assign tans_891011_12131415 = tans_89_1011 + tans_1213_1415;
    assign tans_16171819_20212223 = tans_1617_1819 + tans_2021_2223;
    assign tans_24252627_28293031 = tans_2425_2627 + tans_2829_3031;

    assign tans_012345678_9101112131415 = tans_0123_4567 +
tans_891011_12131415;
    assign tans_1617181920212223_2425262728293031 =
tans_16171819_20212223 + tans_24252627_28293031;

    assign z = tans_012345678_9101112131415 +
tans_1617181920212223_2425262728293031;

    always @(posedge clk or negedge reset)
    begin
        if(reset == 0)
        begin
            tmp0 <= 'b0; tmp1 <= 'b0; tmp2 <= 'b0; tmp3 <= 'b0;
            tmp4 <= 'b0; tmp5 <= 'b0; tmp6 <= 'b0; tmp7 <= 'b0;
            tmp8 <= 'b0; tmp9 <= 'b0; tmp10 <= 'b0; tmp11 <= 'b0;
            tmp12 <= 'b0; tmp13 <= 'b0; tmp14 <= 'b0; tmp15 <= 'b0;
            tmp16 <= 'b0; tmp17 <= 'b0; tmp18 <= 'b0; tmp19 <= 'b0;
            tmp20 <= 'b0; tmp21 <= 'b0; tmp22 <= 'b0; tmp23 <= 'b0;
            tmp24 <= 'b0; tmp25 <= 'b0; tmp26 <= 'b0; tmp27 <= 'b0;
            tmp28 <= 'b0; tmp29 <= 'b0; tmp30 <= 'b0; tmp31 <= 'b0;
        end
        else
        begin
            tmp0 <= b[0] ? {{32{a[31]}}, a} : 'b0;
            tmp1 <= b[1] ? {{31{a[31]}}, a, 1'b0} : 'b0;
            tmp2 <= b[2] ? {{30{a[31]}}, a, 2'b0} : 'b0;
            tmp3 <= b[3] ? {{29{a[31]}}, a, 3'b0} : 'b0;
            tmp4 <= b[4] ? {{28{a[31]}}, a, 4'b0} : 'b0;
            tmp5 <= b[5] ? {{27{a[31]}}, a, 5'b0} : 'b0;
            tmp6 <= b[6] ? {{26{a[31]}}, a, 6'b0} : 'b0;
            tmp7 <= b[7] ? {{25{a[31]}}, a, 7'b0} : 'b0;
            tmp8 <= b[8] ? {{24{a[31]}}, a, 8'b0} : 'b0;
            tmp9 <= b[9] ? {{23{a[31]}}, a, 9'b0} : 'b0;
            tmp10 <= b[10] ? {{22{a[31]}}, a, 10'b0} : 'b0;
            tmp11 <= b[11] ? {{21{a[31]}}, a, 11'b0} : 'b0;
            tmp12 <= b[12] ? {{20{a[31]}}, a, 12'b0} : 'b0;
            tmp13 <= b[13] ? {{19{a[31]}}, a, 13'b0} : 'b0;
            tmp14 <= b[14] ? {{18{a[31]}}, a, 14'b0} : 'b0;
            tmp15 <= b[15] ? {{17{a[31]}}, a, 15'b0} : 'b0;
            tmp16 <= b[16] ? {{16{a[31]}}, a, 16'b0} : 'b0;
            tmp17 <= b[17] ? {{15{a[31]}}, a, 17'b0} : 'b0;
            tmp18 <= b[18] ? {{14{a[31]}}, a, 18'b0} : 'b0;
            tmp19 <= b[19] ? {{13{a[31]}}, a, 19'b0} : 'b0;
            tmp20 <= b[20] ? {{12{a[31]}}, a, 20'b0} : 'b0;
            tmp21 <= b[21] ? {{11{a[31]}}, a, 21'b0} : 'b0;
            tmp22 <= b[22] ? {{10{a[31]}}, a, 22'b0} : 'b0;
            tmp23 <= b[23] ? {{9{a[31]}}, a, 23'b0} : 'b0;
            tmp24 <= b[24] ? {{8{a[31]}}, a, 24'b0} : 'b0;
            tmp25 <= b[25] ? {{7{a[31]}}, a, 25'b0} : 'b0;
            tmp26 <= b[26] ? {{6{a[31]}}, a, 26'b0} : 'b0;
            tmp27 <= b[27] ? {{5{a[31]}}, a, 27'b0} : 'b0;
            tmp28 <= b[28] ? {{4{a[31]}}, a, 28'b0} : 'b0;
            tmp29 <= b[29] ? {{3{a[31]}}, a, 29'b0} : 'b0;
            tmp30 <= b[30] ? {{2{a[31]}}, a, 30'b0} : 'b0;
            tmp31 <= b[31] ? {{1{a[31]}}, a, 31'b0} : 'b0;
        end
    end
endmodule
```

三、测试模块设计

1. 32 位无符号乘法器 testbench

```
`timescale 1ns / 1ps
module MULTU_tb;
    reg clk = 0;
    reg reset = 0;
    reg [31:0] a,b;
    wire [63:0] c;

    MULTU uut(.clk(clk), .reset(reset),
                    .a(a), .b(b), .z(c));

    always #5 clk = ~clk;
    initial
    begin
            #13 reset <= 'b1; a <= 'd2; b <= 'd3;
            #10 a <= 'd2200000000; b <= 'd2200000000;
            #30 reset <= 'b0;
    end
endmodule
```

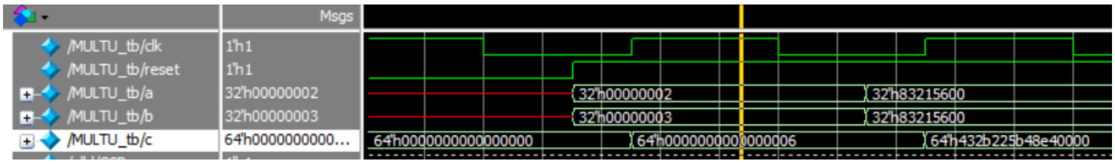
2. 32 位带符号乘法器 testbench

```
`timescale 1ns / 1ps
module MULT_tb;
        reg clk = 0;
        reg reset = 0;
        reg signed [31:0] a,b;
        wire signed [63:0] c;

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

        always #5 clk = ~clk;
        initial
        begin
                #13 reset <= 'b1; a <= 'd3; b <= 'd2;
                #20 a <= 'd5; b <= 'h80000000;
                #20 a <= -5;
                #20 a <= 'd0;
                #30 reset <= 'b0;
        end
endmodule
```

四、实验结果

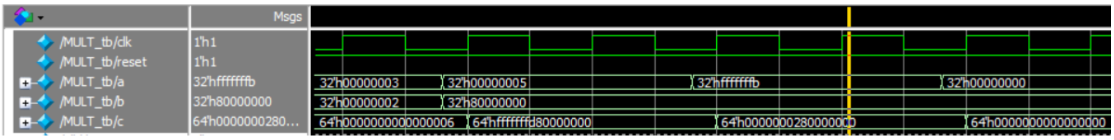
1. 32 位无符号乘法器

   1.1 modelsim 仿真波形图



2. 32 位带符号乘法器

   2.1 modelsim 仿真波形图



五、总结

1. 32 位无符号乘法器可以直接使用阵列乘法实现，而 32 位带符号乘法使用阵列乘法时应当注意被乘数与乘数每一位获得的结果最高位应当用被乘数的符号补齐，而不是用 0，否则结果会出错
2. 被乘数与乘数每一位相乘获得的结果最后进行累加时使用两两相加的方法效率会比 32 个寄存器同时相加所产生的响应时间更低
3. 善用 C++编写打表程序可以提高编程效率