同济大学计算机系

计算机组成原理实验报告

学　　号 **2151140**

姓　　名 王谦

专　　业 信息安全

授课老师 张冬冬

一、实验内容

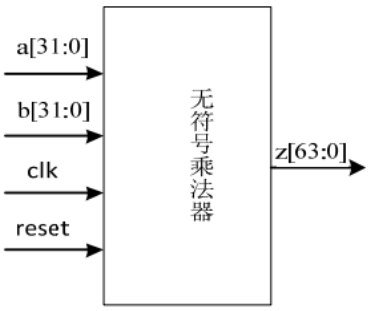
32 位乘法器

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。

实验目标：
（1）了解 32 位带符号、无符号乘法器的实现原理
（2）使用 Verilog 实现 32 位无符号乘法器和带符号乘法器

实验原理：

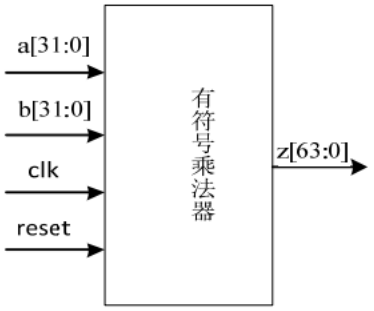
1）无符号乘法器功能为：将两个 32 位无符号数相乘，得到一个 64 位无符号数。



● 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
);
```

3）相关说明

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。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。将低 32 位存放在专用寄存器 LO 中，高 32 位存放在寄存器 HI 中。执行乘法指令过程中不产生异常。本实验不允许使用行为级实现。

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

二、硬件逻辑图

（实验步骤中要求用 logisim 画图的实验，在该部分给出 logisim 原理图，否则该部分在实验报告中不用写）

——本次实验不要求——

三、模块建模

（该部分要求对实验中建模的所有模块进行功能描述，并列出各模块建模的 verilog 代码）

无符号 MULTU：

```
`timescale 1ns / 1ps

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

reg [63:0] temp;
```

```
reg [63:0] store0;
reg [63:0] store1;
reg [63:0] store2;
reg [63:0] store3;
reg [63:0] store4;
reg [63:0] store5;
reg [63:0] store6;
reg [63:0] store7;
reg [63:0] store8;
reg [63:0] store9;
reg [63:0] store10;
reg [63:0] store11;
reg [63:0] store12;
reg [63:0] store13;
reg [63:0] store14;
reg [63:0] store15;
reg [63:0] store16;
reg [63:0] store17;
reg [63:0] store18;
reg [63:0] store19;
reg [63:0] store20;
reg [63:0] store21;
reg [63:0] store22;
reg [63:0] store23;
reg [63:0] store24;
reg [63:0] store25;
reg [63:0] store26;
reg [63:0] store27;
reg [63:0] store28;
reg [63:0] store29;
reg [63:0] store30;
reg [63:0] store31;

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

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

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


reg [63:0] add0_7;
reg [63:0] add8_15;
reg [63:0] add16_23;
reg [63:0] add24_31;

reg [63:0] add0_15;
reg [63:0] add16_31;

//reg [63:0] add0_31;

always @(posedge clk or posedge reset)
begin
    if(reset)
    begin
        temp <= 0;

        store0 <= 0;
        store1 <= 0;
        store2 <= 0;
        store3 <= 0;
        store4 <= 0;
        store5 <= 0;
        store6 <= 0;
        store7 <= 0;
        store8 <= 0;
        store9 <= 0;
        store10 <= 0;
        store11 <= 0;
```

```
        store12 <= 0;
        store13 <= 0;
        store14 <= 0;
        store15 <= 0;
        store16 <= 0;
        store17 <= 0;
        store18 <= 0;
        store19 <= 0;
        store20 <= 0;
        store21 <= 0;
        store22 <= 0;
        store23 <= 0;
        store24 <= 0;
        store25 <= 0;
        store26 <= 0;
        store27 <= 0;
        store28 <= 0;
        store29 <= 0;
        store30 <= 0;
        store31 <= 0;


        add0_1 <= 0;
        add2_3 <= 0;
        add4_5 <= 0;
        add6_7 <= 0;
        add8_9 <= 0;
        add10_11 <= 0;
        add12_13 <= 0;
        add14_15 <= 0;
        add16_17 <= 0;
        add18_19 <= 0;
        add20_21 <= 0;
        add22_23 <= 0;
        add24_25 <= 0;
        add26_27 <= 0;
        add28_29 <= 0;
        add30_31 <= 0;

        add0t1_2t3 <= 0;
        add4t5_6t7 <= 0;
        add8t9_10t11 <= 0;
        add12t13_14t15 <= 0;
        add16t17_18t19 <= 0;
        add20t21_22t23 <= 0;
```

```
        add24t25_26t27 <= 0;
        add28t29_30t31 <= 0;

        add0_7 <= 0;
        add8_15 <= 0;
        add16_23 <= 0;
        add24_31 <= 0;

        add0_15 <= 0;
        add16_31 <= 0;

        //add0_31 <= 0;
    end

    else begin
        store0 <= b[0]? {32'b0, a} : 64'b0;
        store1 <= b[1]? {31'b0, a, 1'b0} : 64'b0;
        store2 <= b[2]? {30'b0, a, 2'b0} : 64'b0;
        store3 <= b[3]? {29'b0, a, 3'b0} : 64'b0;
        store4 <= b[4]? {28'b0, a, 4'b0} : 64'b0;
        store5 <= b[5]? {27'b0, a, 5'b0} : 64'b0;
        store6 <= b[6]? {26'b0, a, 6'b0} : 64'b0;
        store7 <= b[7]? {25'b0, a, 7'b0} : 64'b0;
        store8 <= b[8]? {24'b0, a, 8'b0} : 64'b0;
        store9 <= b[9]? {23'b0, a, 9'b0} : 64'b0;
        store10 <= b[10]? {22'b0, a, 10'b0} : 64'b0;
        store11 <= b[11]? {21'b0, a, 11'b0} : 64'b0;
        store12 <= b[12]? {20'b0, a, 12'b0} : 64'b0;
        store13 <= b[13]? {19'b0, a, 13'b0} : 64'b0;
        store14 <= b[14]? {18'b0, a, 14'b0} : 64'b0;
        store15 <= b[15]? {17'b0, a, 15'b0} : 64'b0;
        store16 <= b[16]? {16'b0, a, 16'b0} : 64'b0;
        store17 <= b[17]? {15'b0, a, 17'b0} : 64'b0;
        store18 <= b[18]? {14'b0, a, 18'b0} : 64'b0;
        store19 <= b[19]? {13'b0, a, 19'b0} : 64'b0;
        store20 <= b[20]? {12'b0, a, 20'b0} : 64'b0;
        store21 <= b[21]? {11'b0, a, 21'b0} : 64'b0;
        store22 <= b[22]? {10'b0, a, 22'b0} : 64'b0;
        store23 <= b[23]? {9'b0, a, 23'b0} : 64'b0;
        store24 <= b[24]? {8'b0, a, 24'b0} : 64'b0;
        store25 <= b[25]? {7'b0, a, 25'b0} : 64'b0;
        store26 <= b[26]? {6'b0, a, 26'b0} : 64'b0;
        store27 <= b[27]? {5'b0, a, 27'b0} : 64'b0;
        store28 <= b[28]? {4'b0, a, 28'b0} : 64'b0;
```

```
        store29 <= b[29]? {3'b0, a, 29'b0} : 64'b0;
        store30 <= b[30]? {2'b0, a, 30'b0} : 64'b0;
        store31 <= b[31]? {1'b0, a, 31'b0} : 64'b0;

        add0_1 <= store0 + store1;
        add2_3 <= store2 + store3;
        add4_5 <= store4 + store5;
        add6_7 <= store6 + store7;
        add8_9 <= store8 + store9;
        add10_11 <= store10 + store11;
        add12_13 <= store12 + store13;
        add14_15 <= store14 + store15;
        add16_17 <= store16 + store17;
        add18_19 <= store18 + store19;
        add20_21 <= store20 + store21;
        add22_23 <= store22 + store23;
        add24_25 <= store24 + store25;
        add26_27 <= store26 + store27;
        add28_29 <= store28 + store29;
        add30_31 <= store30 + store31;

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

        add0_7 <= add0t1_2t3 + add4t5_6t7;
        add8_15 <= add8t9_10t11 + add12t13_14t15;
        add16_23 <= add16t17_18t19 + add20t21_22t23;
        add24_31 <= add24t25_26t27 + add28t29_30t31;

        add0_15 <= add0_7 + add8_15;
        add16_31 <= add16_23 + add24_31;

        temp <= add0_15 + add16_31;
    end
end

assign z = temp;
```

```
endmodule
```

有符号 MULT：

```
`timescale 1ns / 1ps

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

reg [32:0] a_1;
reg [32:0] b_1;
reg [65:0] temp;

always @(posedge clk or posedge reset)
begin
    if(reset)
    begin
        a_1 <= 33'b0;
        b_1 <= 33'b0;
        temp <= 66'b0;
    end

    else begin
        a_1 = {a[31], a[31:0]};
        b_1 = {b[31:0], 1'b0};
        temp = {33'b0, b_1[32:0]};
        repeat(32)
        begin
            if(temp[1] == temp[0])
            begin
                temp = temp >>> 1;
                temp[65] = temp[64];
            end
            else if(temp[1:0] == 2'b01)
            begin
                temp = temp + {a_1, 33'b0};
                temp = temp >>> 1;
                temp[65] = temp[64];
            end
            else if(temp[1:0] == 2'b10)
            begin
                temp = temp - {a_1, 33'b0};
```

```
                temp = temp >>> 1;
                temp[65] = temp[64];
            end
        end
    end
end

assign z = temp[64:1];

endmodule
```

四、测试模块建模

（要求列写各建模模块的 test bench 模块代码）

两者采用相近的 tb 数据，但是结果会有所不同：

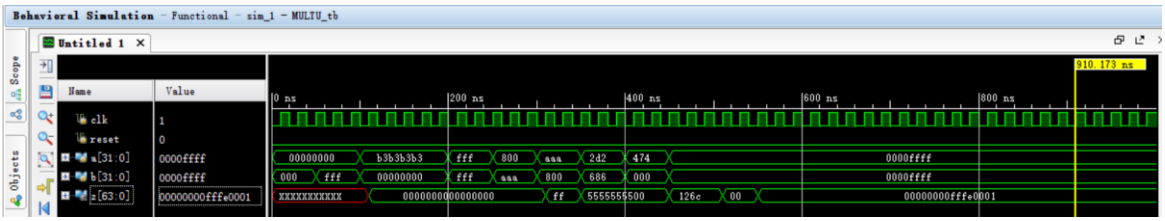
```
`timescale 1ns / 1ps

module MULTU_tb();

reg clk;
reg reset;
reg [31:0] a;
reg [31:0] b;
wire [63:0] z;

initial
reset = 0;

initial
clk = 0;
always #10 clk = ~clk;

initial
begin
    clk = 0;
    a = 0;
    b = 0;
    #50
    a = 0;
    b = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
    #50
```

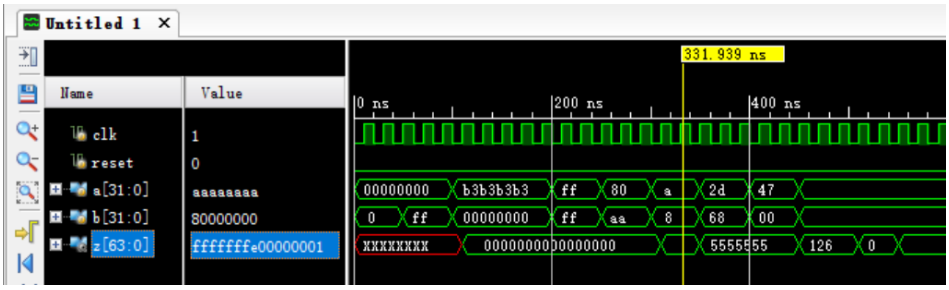
```
    a = 32'b1011_0011_1011_0011_1011_0011_1011_0011;
    b = 0;
    #50
    #50
    a = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
    b = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
    #50
    a = 32'b1000_0000_0000_0000_0000_0000_0000_0000;
    b = 32'b1010_1010_1010_1010_1010_1010_1010_1010;
    #50
    a = 32'b1010_1010_1010_1010_1010_1010_1010_1010;
    b = 32'b1000_0000_0000_0000_0000_0000_0000_0000;
    #50
    a = 32'b1011_0100_1011_0100_1011_0100_1011_01;
    b = 32'b1101_0000_1101_0000_1101_0000_1101_000;
    #50
    a = 32'b1000_1110_1000_1110_1000_1110_1000_111;
    b = 32'b0000_0000_0000_0000_1110_1110_1110_1110;
    #50
    a = 32'b1111_1111_1111_1111;
    b = 32'b1111_1111_1111_1111;
end

MULTU MULTU_init(
    .clk(clk),
    .reset(reset),
    .a(a),
    .b(b),
    .z(z)
);

endmodule
```

五、实验结果

（该部分可截图说明，要求 logisim 逻辑验证图、modelsim 仿真波形图、以及下板后的实验结果贴图（实验步骤中没有下板要求的实验，不需要下板贴图））

无符号 MULTU：

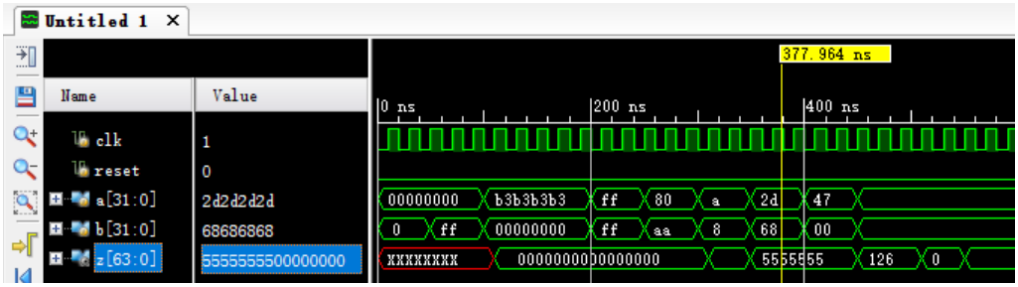


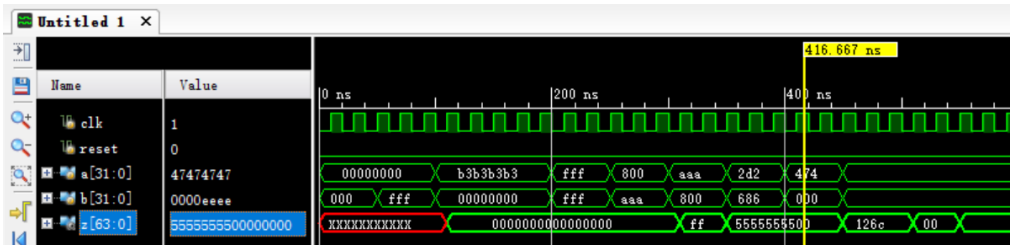


程序员

1111111111111111111111111111111 × 1111111111111
1111 1111 1111 1111 1111 1111 1111 1110 0000 0000 0000 0000 0000 0000 0(

HEX	FFFF FFFE 0000 0001
DEC	-8,589,934,591
OCT	1 777 777 777 700 000 000 001
BIN	1111 1111 1111 1111 1111 1111 1111 1110 0000 0000 0000 0000 0000 0000 0000 0001

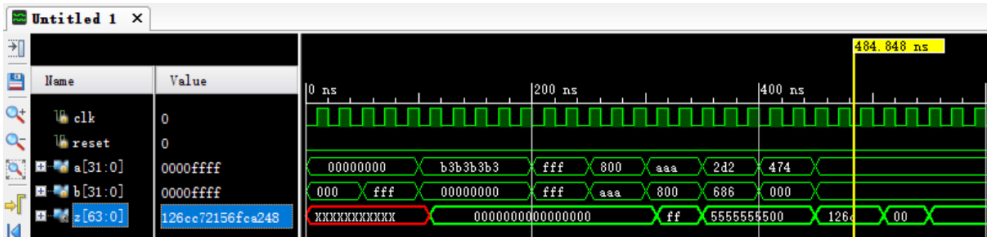




程序员

)00000000000 × 10101010101010101010101010101010 =
0101 0101 0101 0101 0101 0101 0101 0101 0000 0000 0000 0000 0000 00

HEX	5555 5555 0000 0000
DEC	6,148,914,689,804,861,440
OCT	525 252 525 240 000 000 000
BIN	0101 0101 0101 0101 0101 0101 0101 0101 0000 0000 0000 0000 0000 0000 0000 0000

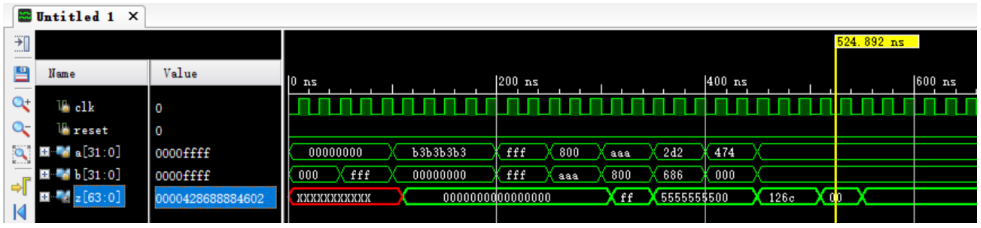


程序员

0110100101101 × 11010000110100001101000001101000 =

0001 0010 0110 1100 1100 0111 0010 0001 0101 0110 1111 1100 1010 0010

HEX	126C C721 56FC A248
DEC	1,327,654,936,174,699,080
OCT	111 546 162 052 677 121 110
BIN	0001 0010 0110 1100 1100 0111 0010 0001 0101 0110 1111 1100 1010 0010 0100 1000

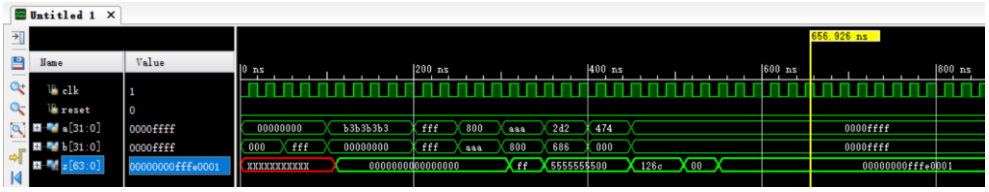


程序员

0111010001110100011101000111 × 1110111011101110 =

0100 0010 1000 0110 1000 1000 1000 1000 0100 0110 0000 0010

HEX	4286 8888 4602
DEC	73,145,583,683,074
OCT	2 050 321 042 043 002
BIN	0100 0010 1000 0110 1000 1000 1000 1000 0100 0110 0000 0010



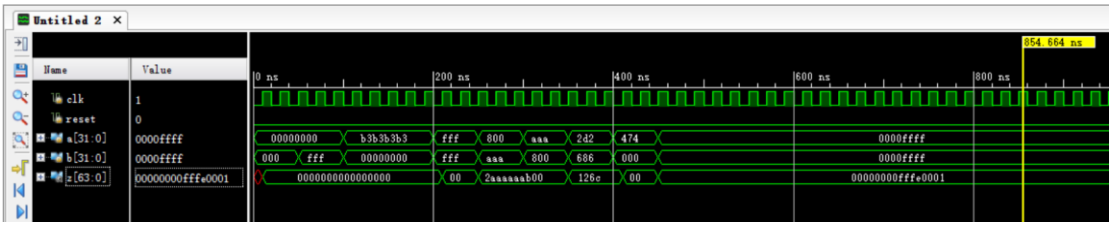
程序员

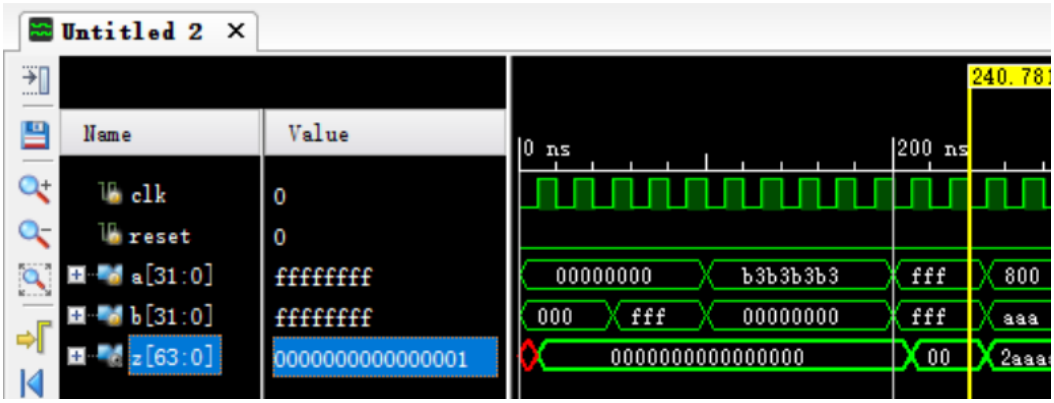
111111111111111 × 111111111111111 =

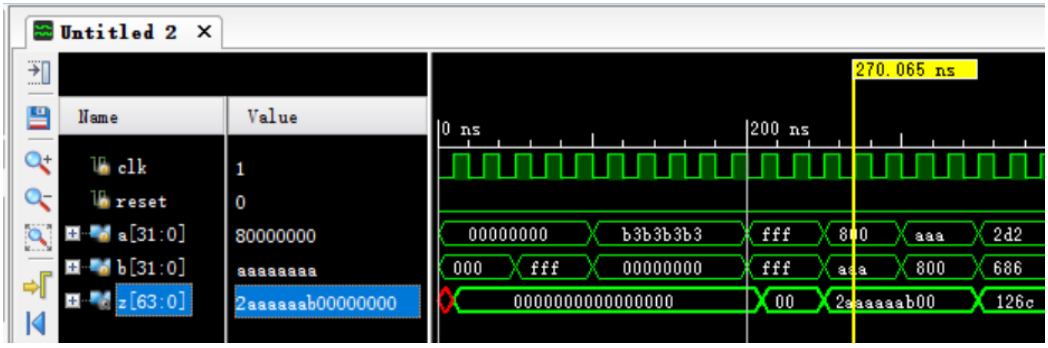
1111 1111 1111 1110 0000 0000 0000 0001

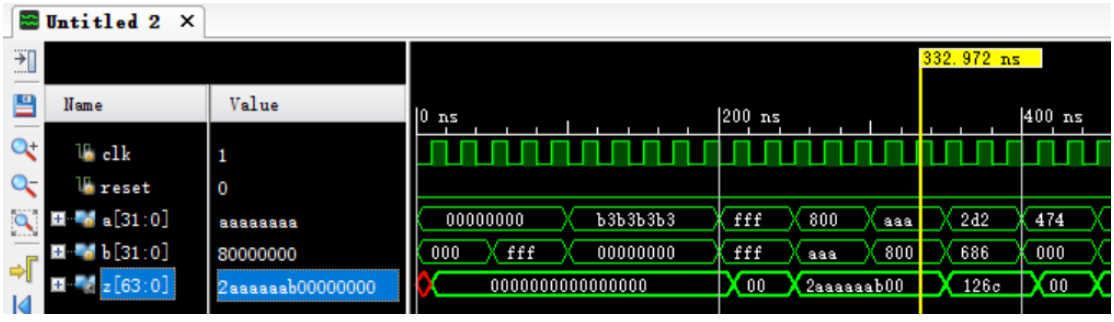
HEX	FFFE 0001
DEC	4,294,836,225
OCT	37 777 400 001
BIN	1111 1111 1111 1110 0000 0000 0000 0001

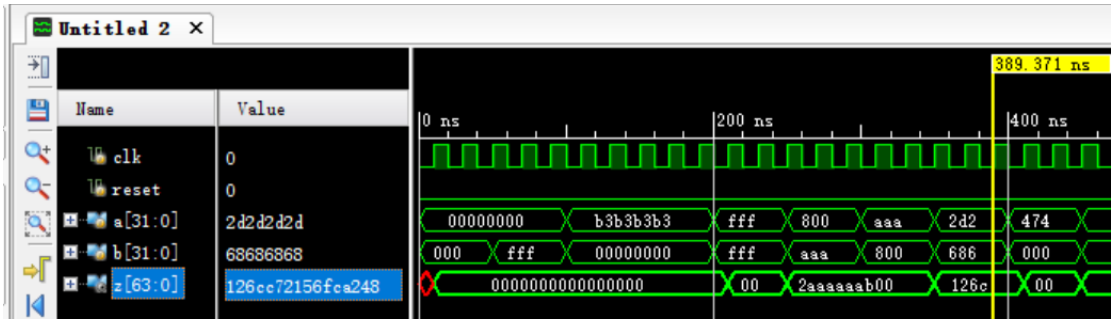
有符号 MULT：

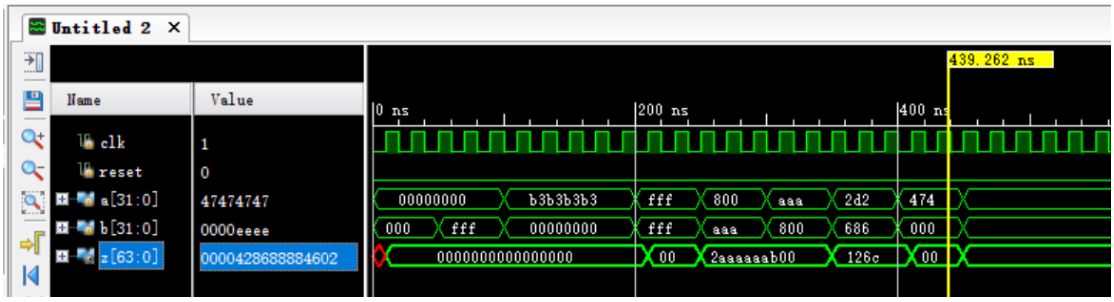

Untitled 2
Name
Value
clk
reset
a[31:0]
b[31:0]
z[63:0]
0 ns
200 ns
400 ns
600 ns
800 ns
854.664 ns

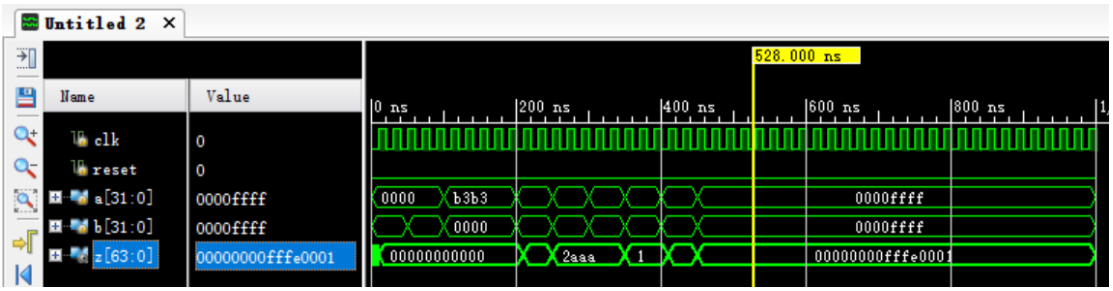

Untitled 2
Name
Value
clk
0
reset
0
a[31:0]
ffffffff
b[31:0]
ffffffff
z[63:0]
0000000000000001
0 ns
200 ns
240.78
00000000
b3b3b3b3
fff
800
000
fff
00000000
fff
aaa
0000000000000000
00
2aaa


Untitled 2
Name
Value
clk
1
reset
0
a[31:0]
80000000
b[31:0]
aaaaaaaa
z[63:0]
2aaaaaab00000000
0 ns
200 ns
270.065 ns
00000000
b3b3b3b3
fff
800
aaa
2d2
000
fff
00000000
fff
aaa
800
686
0000000000000000
00
2aaaaaab00
126c


Untitled 2
Name
Value
clk
1
reset
0
a[31:0]
aaaaaaaa
b[31:0]
80000000
z[63:0]
2aaaaaab00000000
0 ns
200 ns
400 ns
332.972 ns
00000000
b3b3b3b3
fff
800
aaa
2d2
474
000
fff
00000000
fff
aaa
800
686
000
0000000000000000
00
2aaaaaab00
126c
00


Untitled 2
Name
Value
clk
0
reset
0
a[31:0]
2d2d2d2d
b[31:0]
68686868
z[63:0]
126cc72156fca248
0 ns
200 ns
400 ns
389.371 ns
00000000
b3b3b3b3
fff
800
aaa
2d2
474
000
fff
00000000
fff
aaa
800
686
000
0000000000000000
00
2aaaaaab00
126c
00


Untitled 2
Name
Value
clk
1
reset
0
a[31:0]
47474747
b[31:0]
0000eeee
z[63:0]
0000428688884602
439.262 ns
0 ns
200 ns
400 ns
00000000
b3b3b3b3
fff
800
aaa
2d2
474
000
fff
00000000
fff
aaa
800
686
000
0000000000000000
00
2aaaaaab00
126c
00


Untitled 2
Name
Value
clk
0
reset
0
a[31:0]
0000ffff
b[31:0]
0000ffff
z[63:0]
00000000fffe0001
528.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
0000
b3b3
0000ffff
0000
0000ffff
00000000000
2aaa
1
00000000fffe0001