**中国海洋大学计算机科学与技术系**

**实验报告**

**姓名：岳宇轩 年级：2019** **专业：** 19慧与

**科目：计算机组成原理** **题目：ALU**

**实验时间:** 2021 年5 月6日  **实验教师:** 张巍

**一、实验结果及截图分析：**

（※代码挖空的部分必须截图或复制）

1. **代码补全alu.v**

`timescale 1ns / 1ps

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// > 文件名: alu.v

// > 描述 ：ALU模块，可做12种操作

// > 作者 : LOONGSON

// > 日期 : 2016-04-14

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

module alu(

input [11:0] alu\_control,

input [31:0] alu\_src1,

input [31:0] alu\_src2,

output [31:0] alu\_result

);

reg [31:0] alu\_result;

wire alu\_add; //加法

wire alu\_sub; //减法

wire alu\_slt; //有符号比较，小于置位

wire alu\_sltu; //无符号比较，小于置位

wire alu\_and; //按位与

wire alu\_nor; //按位或非

wire alu\_or; //按位或

wire alu\_xor; //按位异或

wire alu\_sll; //逻辑左移

wire alu\_srl; //逻辑右移

wire alu\_sra; //算数右移

wire alu\_lui; //高位加载

assign alu\_add = alu\_control[11];

assign alu\_sub = alu\_control[10];

assign alu\_slt = alu\_control[ 9];

assign alu\_sltu = alu\_control[ 8];

assign alu\_and = alu\_control[ 7];

assign alu\_nor = alu\_control[ 6];

assign alu\_or = alu\_control[ 5];

assign alu\_xor = alu\_control[ 4];

assign alu\_sll = alu\_control[ 3];

assign alu\_srl = alu\_control[ 2];

assign alu\_sra = alu\_control[ 1];

assign alu\_lui = alu\_control[ 0];

wire [31:0] add\_sub\_result; //加减结果，减法用加法来实现

wire [31:0] slt\_result;

wire [31:0] sltu\_result;

wire [31:0] and\_result;

wire [31:0] nor\_result;

wire [31:0] or\_result;

wire [31:0] xor\_result;

wire [31:0] sll\_result;

wire [31:0] srl\_result;

wire [31:0] sra\_result;

wire [31:0] lui\_result;

wire signed [31:0] temp\_src1; //带符号数的临时变量

assign temp\_src1 = alu\_src1; //方便后面对alu\_src1进行算数右移

assign and\_result = alu\_src1 & alu\_src2; //按位与

assign or\_result = alu\_src1 | alu\_src2; //按位或

assign nor\_result = ~or\_result; //或非

assign xor\_result = alu\_src1 ^ alu\_src2; //异或

assign lui\_result = {alu\_src2[15:0], 16'd0}; //高位加载，第二个操作数的低十六位加载到高十六位上

assign sll\_result = alu\_src1 << alu\_src2; //逻辑左移

assign srl\_result = alu\_src1 >> alu\_src2; //逻辑右移

assign slt\_result = adder\_result[31] ? 1'b1 : 1'b0; // 带符号数小于置位

assign sltu\_result = adder\_cout ? 1'b0 : 1'b1; //无符号数小于置位

assign sra\_result = temp\_src1 >>> alu\_src2; //算数右移

wire [31:0] adder\_operand1;

wire [31:0] adder\_operand2;

wire adder\_cin ;

wire [31:0] adder\_result ;

wire adder\_cout ;

assign adder\_operand1 = alu\_src1;

assign adder\_operand2 = alu\_add ? alu\_src2 : ~alu\_src2; //默认进行减法，为slt和sltu服务

assign adder\_cin = ~alu\_add;

adder adder\_module( //调用实验一中的adder.v加法模块

.operand1(adder\_operand1),

.operand2(adder\_operand2),

.cin (adder\_cin ),

.result (adder\_result ),

.cout (adder\_cout )

);

//代码补全部分

assign add\_sub\_result = adder\_result; //先给加减法结果赋值

always@(\*)

begin

if(alu\_add | alu\_sub) //如果是加法或者减法操作，则把加减法结果add\_sub\_result放入alu最终的运算结果alu\_result中

alu\_result <= add\_sub\_result;

else if(alu\_slt) //如果是有符号比较，则把有符号比较结果slt\_result放入alu最终的运算结果alu\_result中

alu\_result <= slt\_result;

else if(alu\_sltu) //如果是无符号比较，则把无符号比较结果sltu\_result放入alu最终的运算结果alu\_result中

alu\_result <= sltu\_result;

else if(alu\_and) //如果是按位与操作，则把按位与操作结果and\_result放入alu最终的运算结果alu\_result中

alu\_result <= and\_result;

else if(alu\_nor) //如果是按位或非操作，则把按位或非操作结果nor\_result放入alu最终的运算结果alu\_result中

alu\_result <= nor\_result;

else if(alu\_or) //如果是按位或操作，则把按位或操作结果or\_result放入alu最终的运算结果alu\_result中

alu\_result <= or\_result;

else if(alu\_xor) //如果是按位异或操作，则把按位异或操作结果xor\_result放入alu最终的运算结果alu\_result中

alu\_result <= xor\_result;

else if(alu\_sll) //如果是逻辑左移操作，则把逻辑左移的结果alu\_sll放入alu最终的运算结果alu\_result中

alu\_result <= sll\_result;

else if(alu\_srl) //如果是逻辑右移操作，则把逻辑右移的结果alu\_srl放入alu最终的运算结果alu\_result中

alu\_result <= srl\_result;

else if(alu\_sra) //如果是算术右移操作，则把算术右移的结果alu\_sra放入alu最终的运算结果alu\_result中

alu\_result <= sra\_result;

else if(alu\_lui) //如果是高位加载操作，则把高位加载操作的结果lui\_result放入alu最终的运算结果alu\_result中

alu\_result <= lui\_result;

end

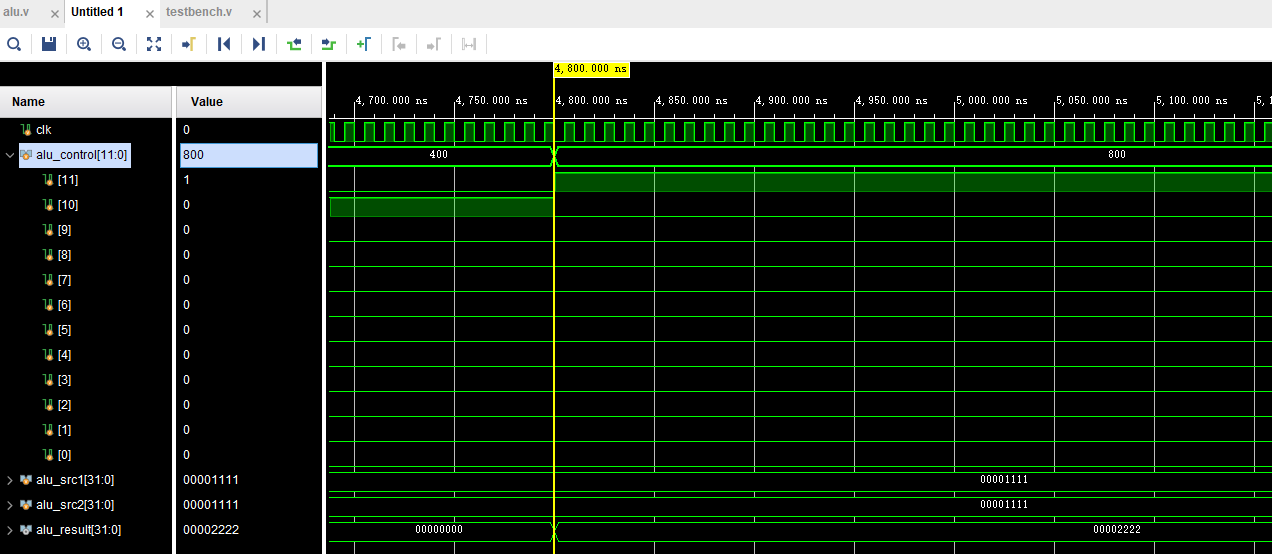
endmodule

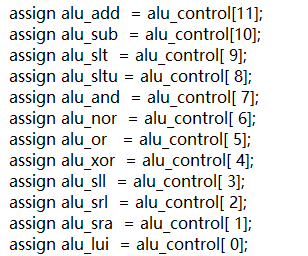
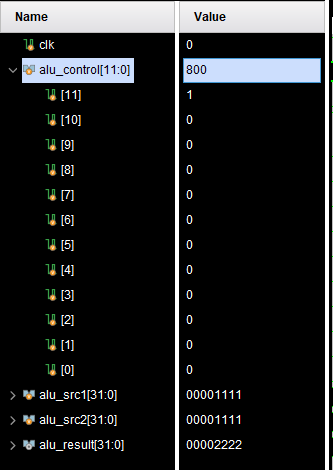
**2.添加adder.v adder\_display.v alu\_display.v testbench.v**

之前实验用到过的文件，这里不再分析解释了

1. **完成仿真，生成仿真波形图像**

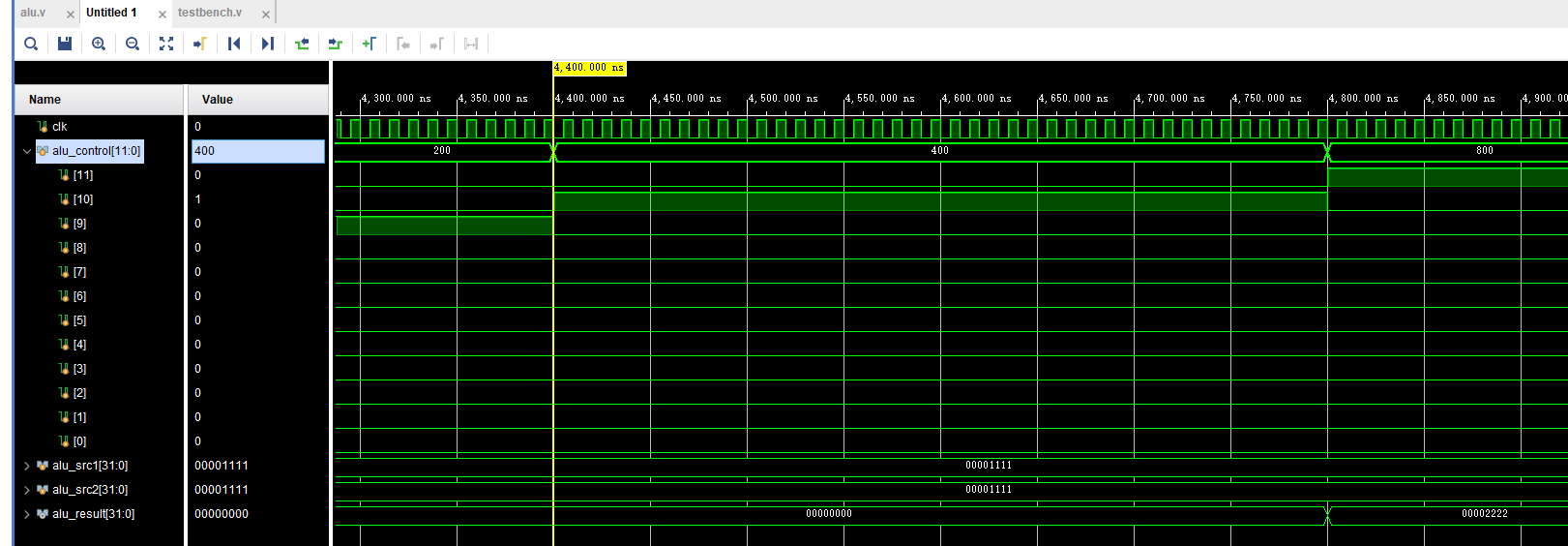
**3.1加法**

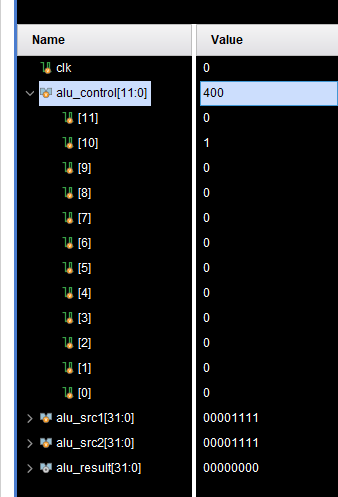




可以看到，alu控制信号为1000 0000 0000B，对照上方右侧定义，alu操作为加法操作，两个操作数的值分别为1111H和1111H，加法运算后的结果为2222H。

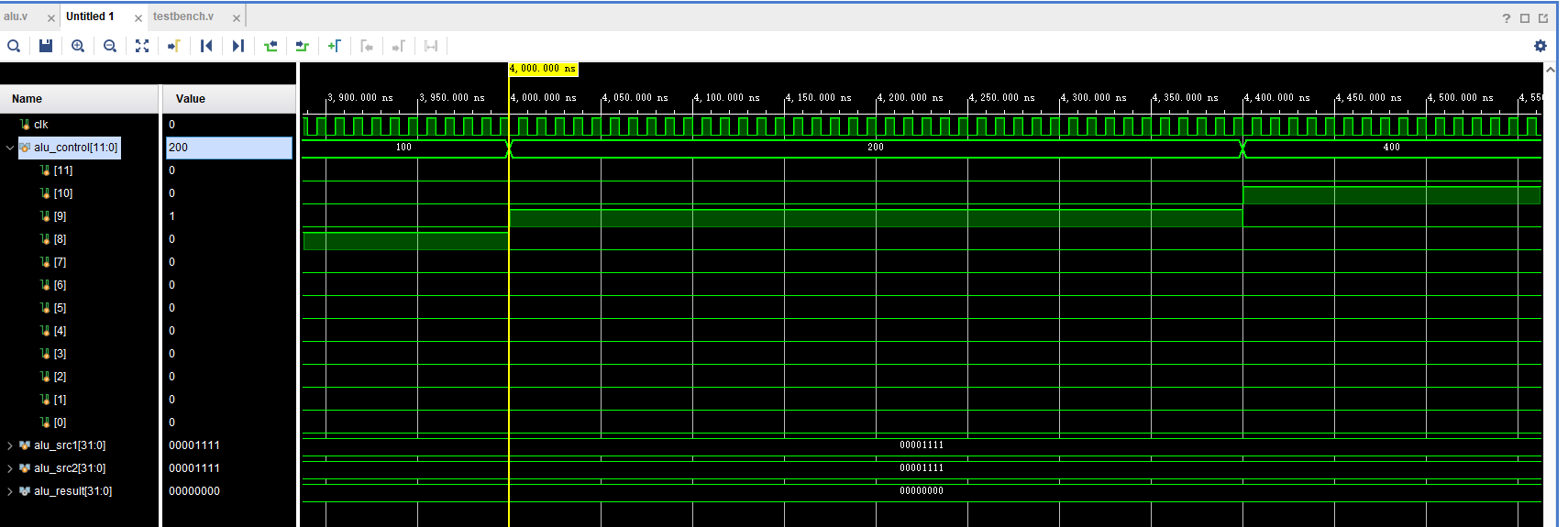
**3.2减法操作**

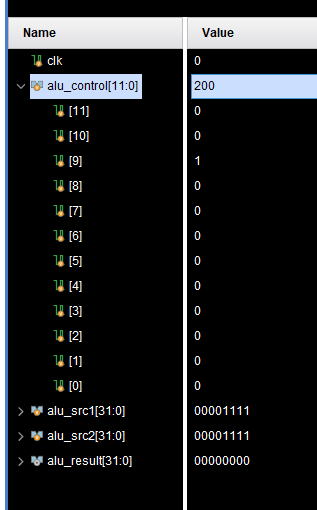




可以看到，alu控制器值为0100 0000 0000B，选择了减法操作，1111H-1111H = 0

**3.3有符号比较操作**

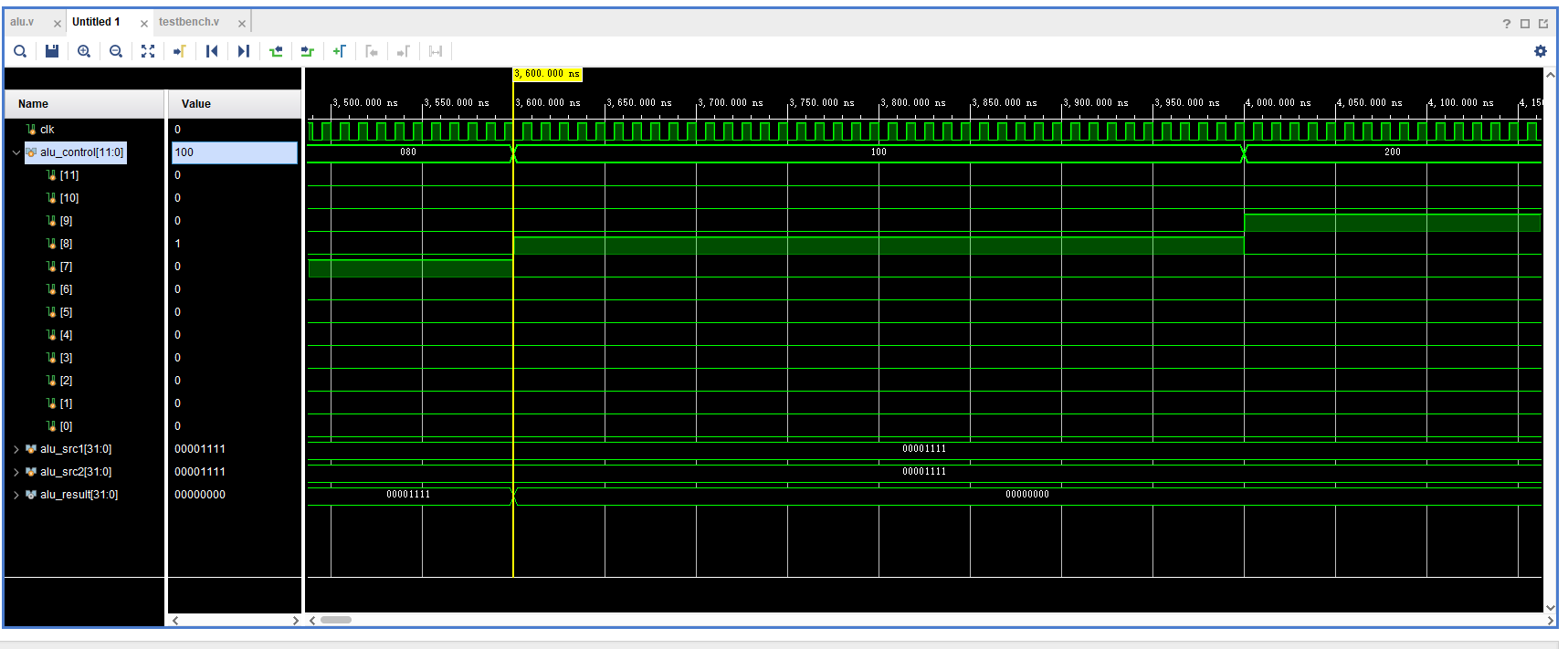


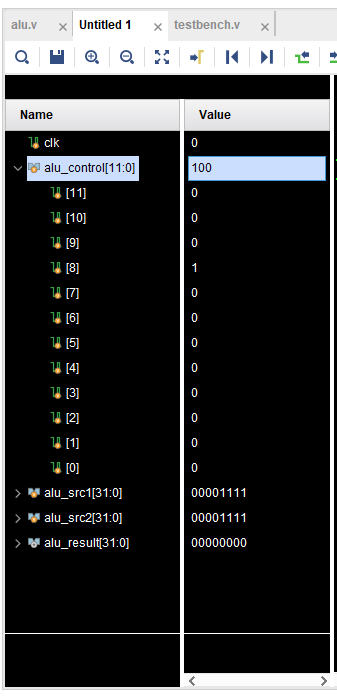


alu控制器值为0010 0000 0000B，选择了有符号数比较操作。

两操作数的机器码相同，按照有符号数比较结果是相同的。

**3.4无符号比较操作**

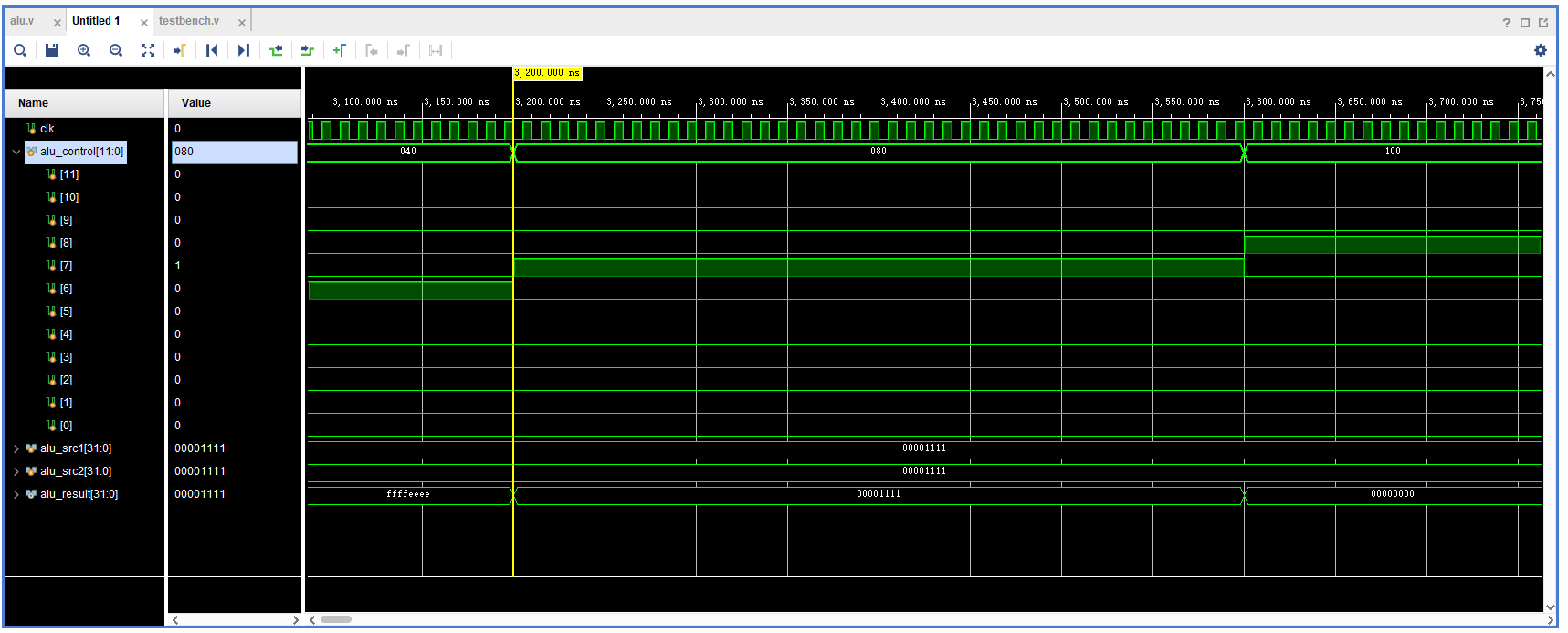


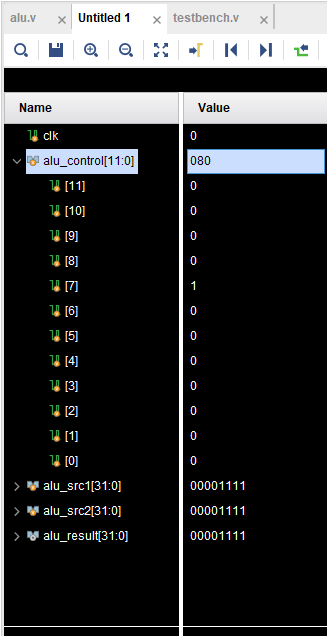


alu控制器值为0001 0000 0000B，选择了无符号数比较操作。

两操作数的机器码相同，按照无符号数比较结果是相同的。

**3.5按位与操作**

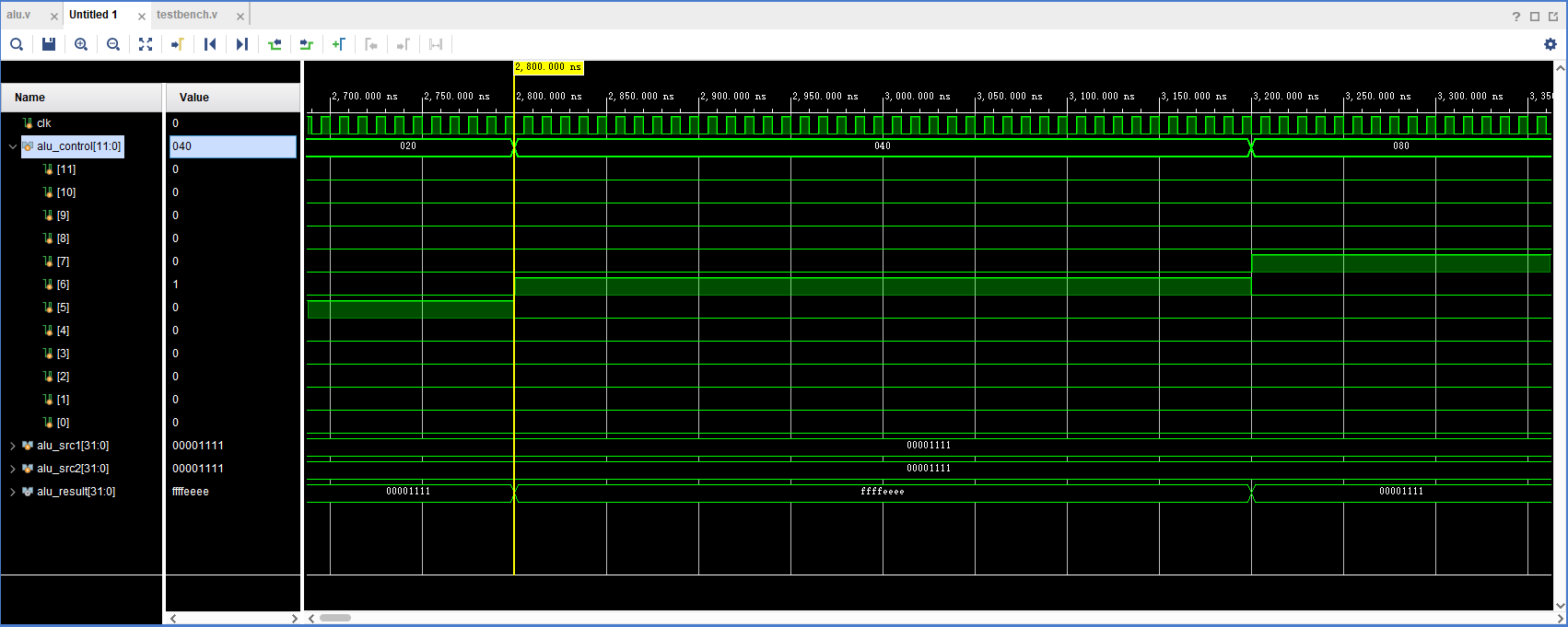


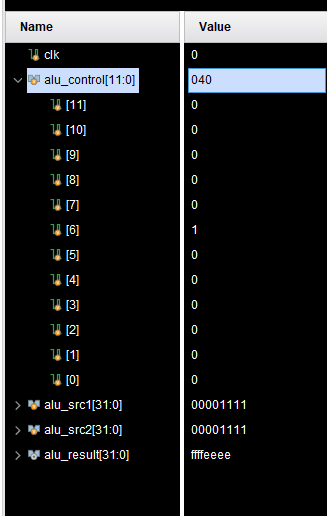


alu控制器值为0000 1000 0000B，选择按位与操作

1111H 与1111H进行按位与操作，结果为1111H

**3.6按位或非操作**



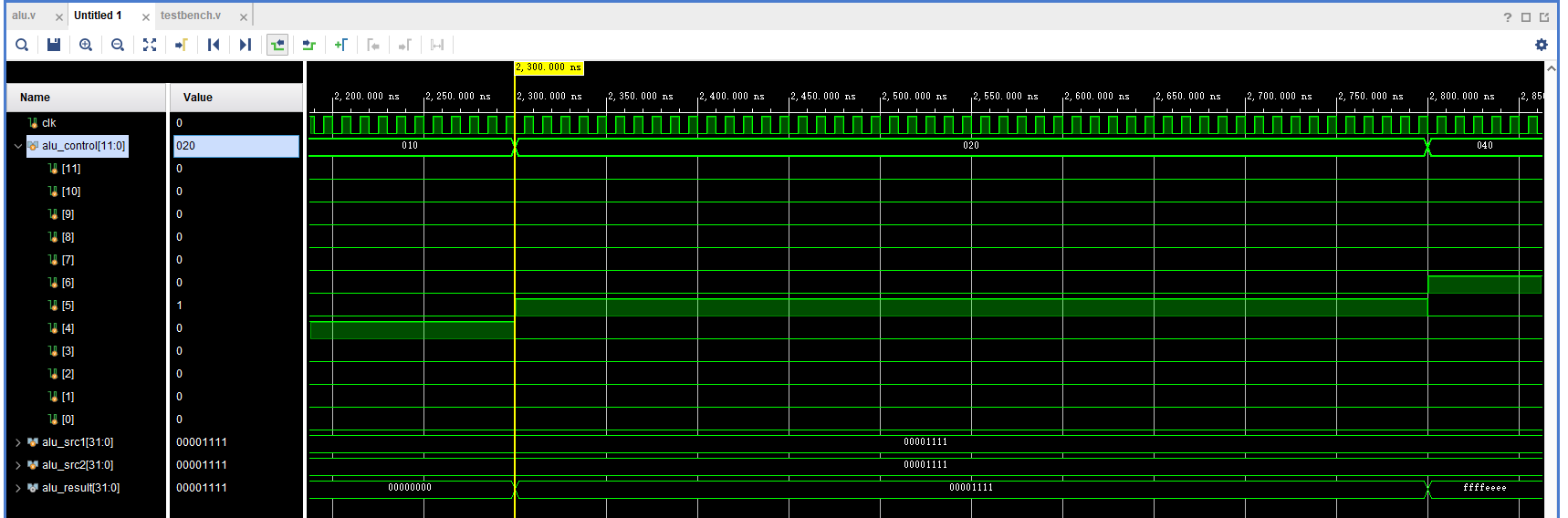


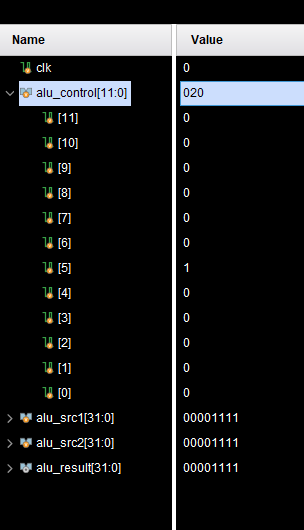
alu控制器值为0000 0100 0000B，选择操作为按位或非

0000 0000 0000 0000 0001 0001 0001 0001B与0000 0000 0000 0000 0001 0001 0001 0001B按位或非结果为

1111 1111 1111 1111 1110 1110 1110 1110B，十六进制表示为ffff eeeeH

**3.7按位或操作**

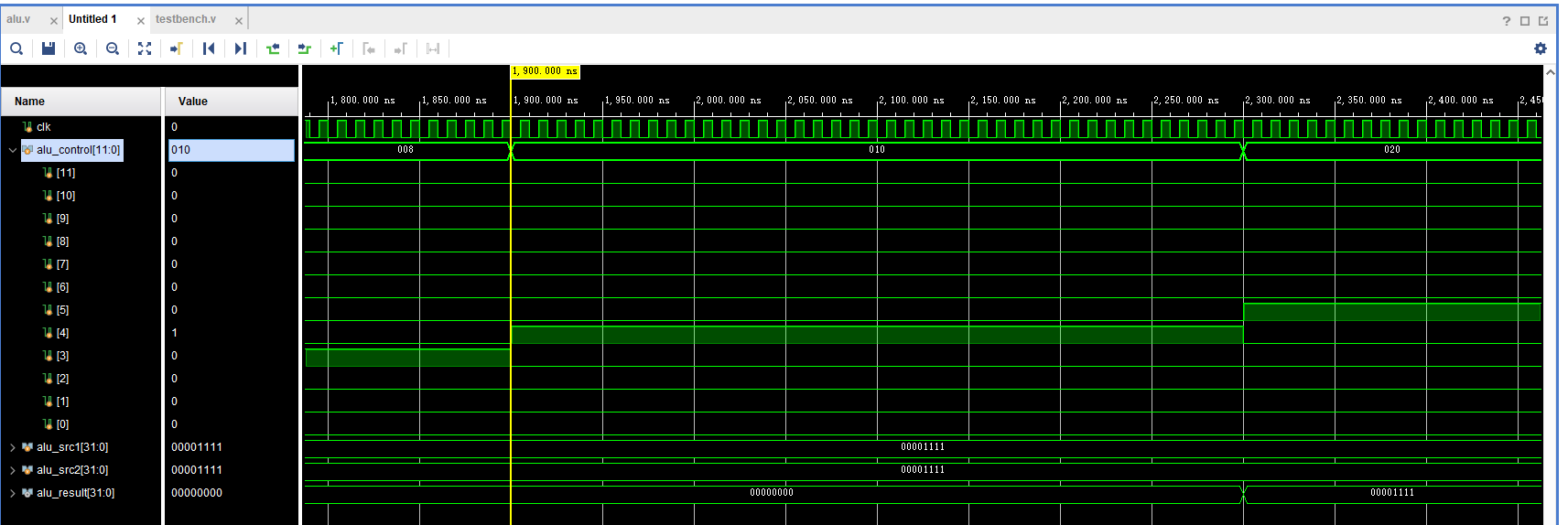


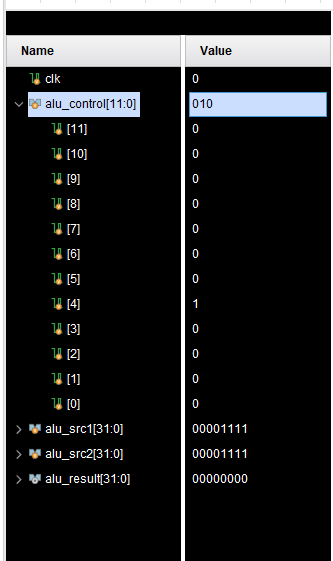


alu控制器值为0000 0010 0000B选择了按位或操作

1111H与1111H按位或，结果为0001 0001 0001 0001B，十六进制表示为1111H

**3.8按位异或操作**

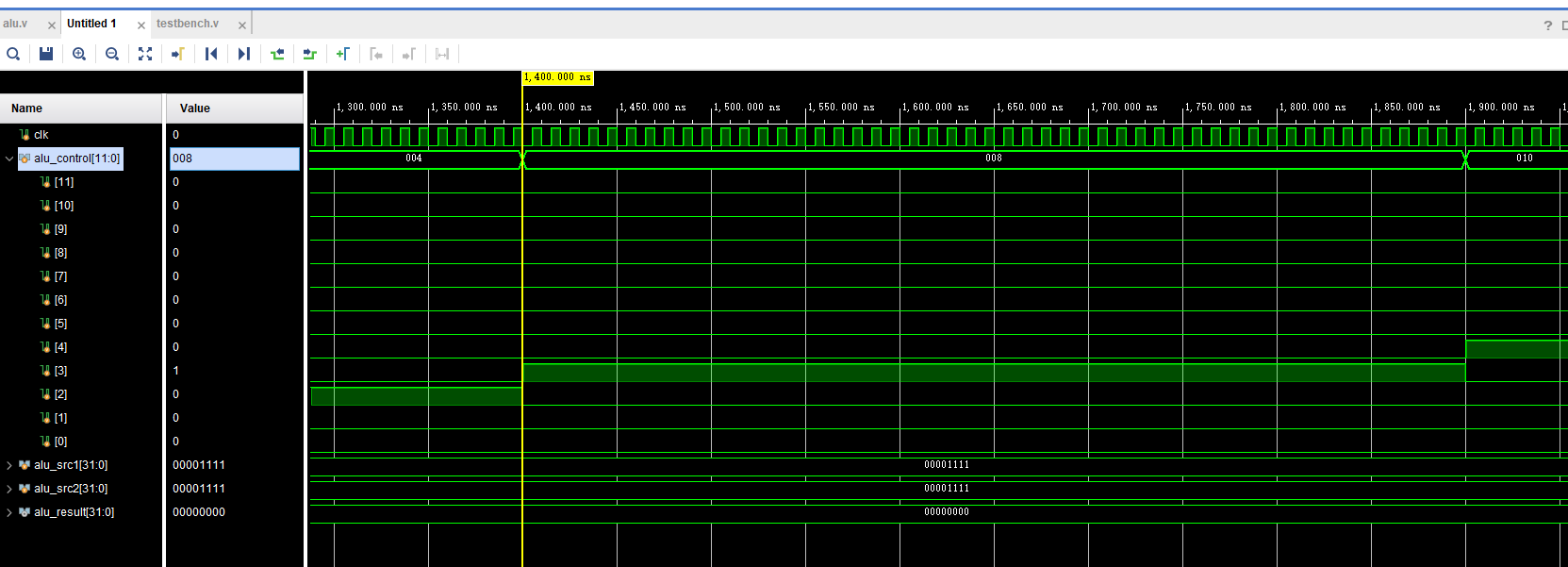


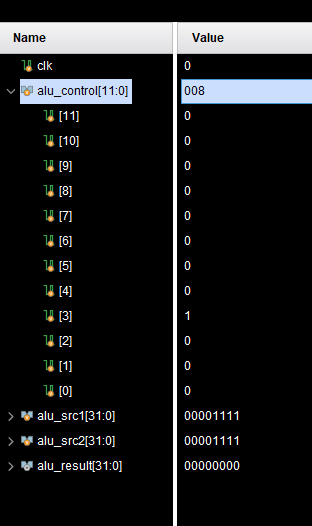


alu控制器值为0000 0001 0000B，选择操作为按位异或，0000 0000 0000 0000 0001 0001 0001 0001B与0000 0000 0000 0000 0001 0001 0001 0001B按位异或结果为

0000 0000 0000 0000 0000 0000 0000 0000B，十六进制表示为0000 0000H

**3.9逻辑左移操作**

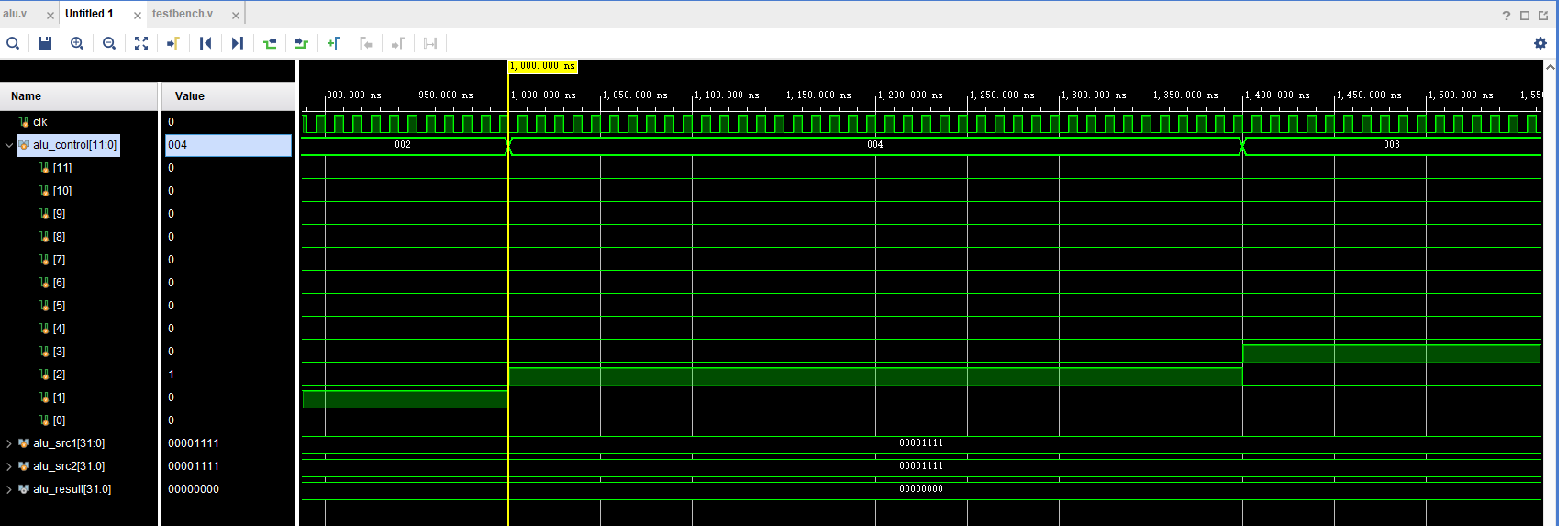


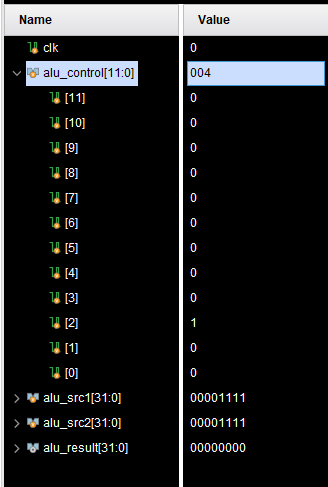


alu控制器值为0000 0000 1000B，选择操作为逻辑左移，高位移出，高位补零，0000 0000 0000 0000 0001 0001 0001 0001B逻辑左移0000 0000 0000 0000 0001 0001 0001 0001B位结果为

0000 0000 0000 0000 0000 0000 0000 0000B，十六进制表示为0000 0000H

**3.10逻辑左移操作**

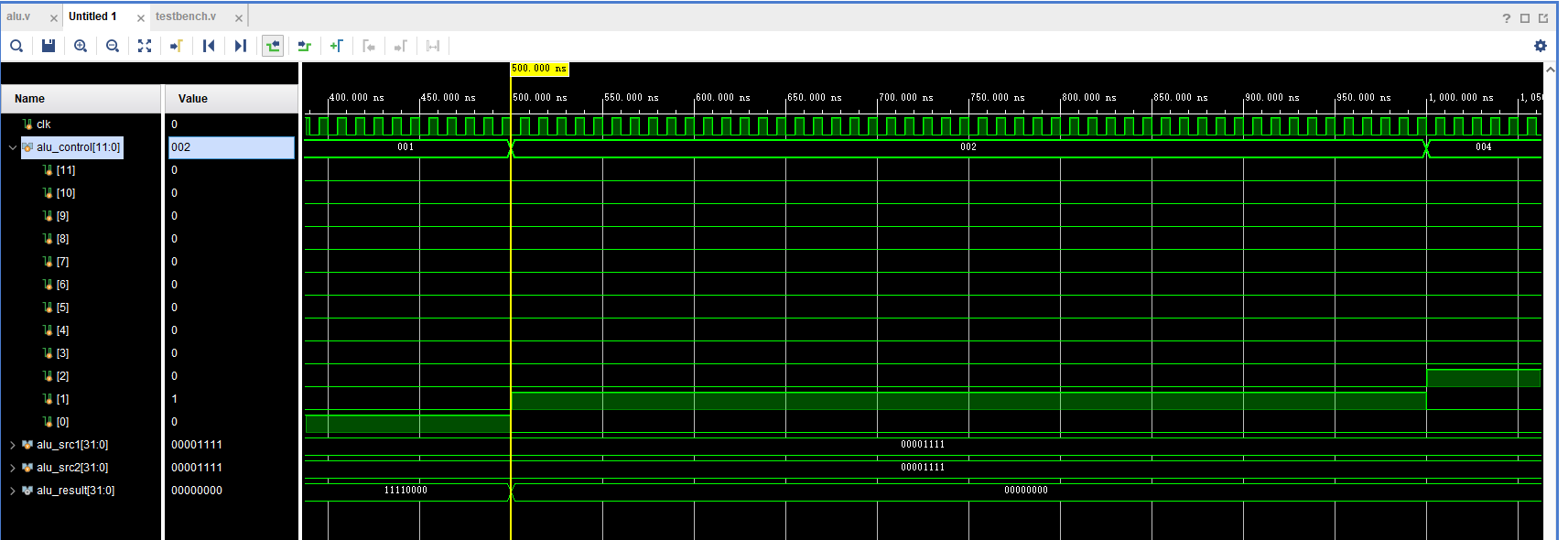


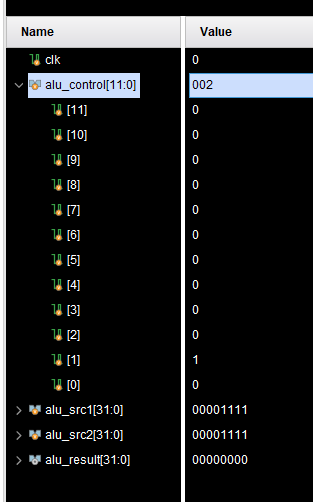


alu控制器值为0000 0000 0100B，选择操作为逻辑右移，低位移出，高位补零，0000 0000 0000 0000 0001 0001 0001 0001B逻辑右移0000 0000 0000 0000 0001 0001 0001 0001B位结果为

0000 0000 0000 0000 0000 0000 0000 0000B，十六进制表示为0000 0000H

**3.11算术左移操作**

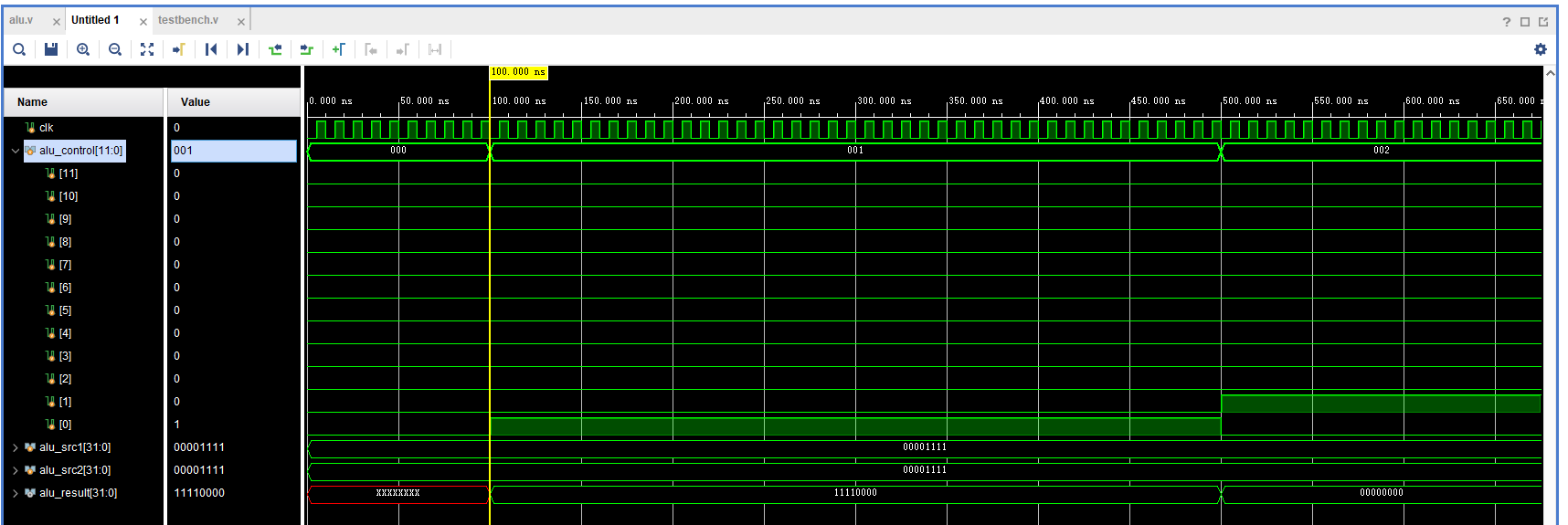


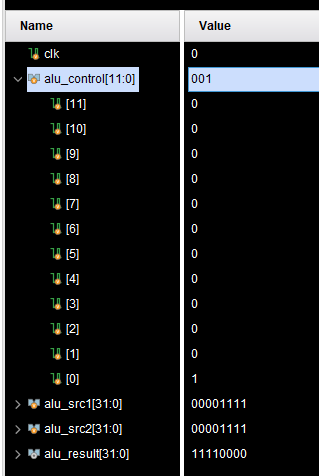


alu控制器值为0000 0000 0010B，选择操作为算术左移，高位移出，低位补零，0000 0000 0000 0000 0001 0001 0001 0001B算术左移0000 0000 0000 0000 0001 0001 0001 0001B位结果为

0000 0000 0000 0000 0000 0000 0000 0000B，十六进制表示为0000 0000H

**3.12高位加载操作**





alu控制器值为0000 0000 0001B，选择操作为高位加载，第二个操作数的低十六位加载到高十六位上，低十六位填零。 0000 0000 0000 0000 0001 0001 0001 0001B高位加载的结果为0001 0001 0001 0001 0000 0000 0000 0000B，十六进制表示为1111 0000H

1. **实验总结**

通过这次实验，我对于alu的原理有了更加清楚的认知。

首先，alu具有多种功能，如加法，减法等，因此它需要一个控制信号，来告诉他需要进行那种操作。alu有两个源操作数，一个目的操作数。在代码补全的过程中，实现了对alu控制信号的判断，凭此来决定将那种运算的结果作为alu运算的结果进行输出。以及，在实现每种操作的过程中，我对于各种运算操作的原理和实现过程有了更加清楚的认知。

本次实验最大的收获是对于仿真波形图像的理解。我学习了对于含有选择控制信号电路的仿真波形图像的观察和分析，控制信号不同的值代表选择了不同的输出通路。在本次实验中，两个源操作数一直不变，而控制信号是不断变化的，alu结果也随着控制信号的改变对应输出相应运算后的结果。通过观察每个时钟周期前后各个值的变化，能更好的理解alu使用过程中的原理。