**组成原理实验课程第 4 次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | alu模块的复现与改进 | | | 班级 | 李涛老师 |
| 学生姓名 | 申宗尚 | 学号 | 2213924 | 指导老师 | 董前琨 |
| 实验地点 | 实验A306 | | 实验时间 | 2024.5.9 | |

1. **实验目的**

1. 熟悉并掌握MIPS计算机中寄存器堆的原理和设计方法。

2. 初步了解MIPS指令结构和源操作数/目的操作数的概念。

3. 熟悉并运用verilog语言进行电路设计。

4. 为后续设计cpu的实验打下基础。

1. **实验内容**

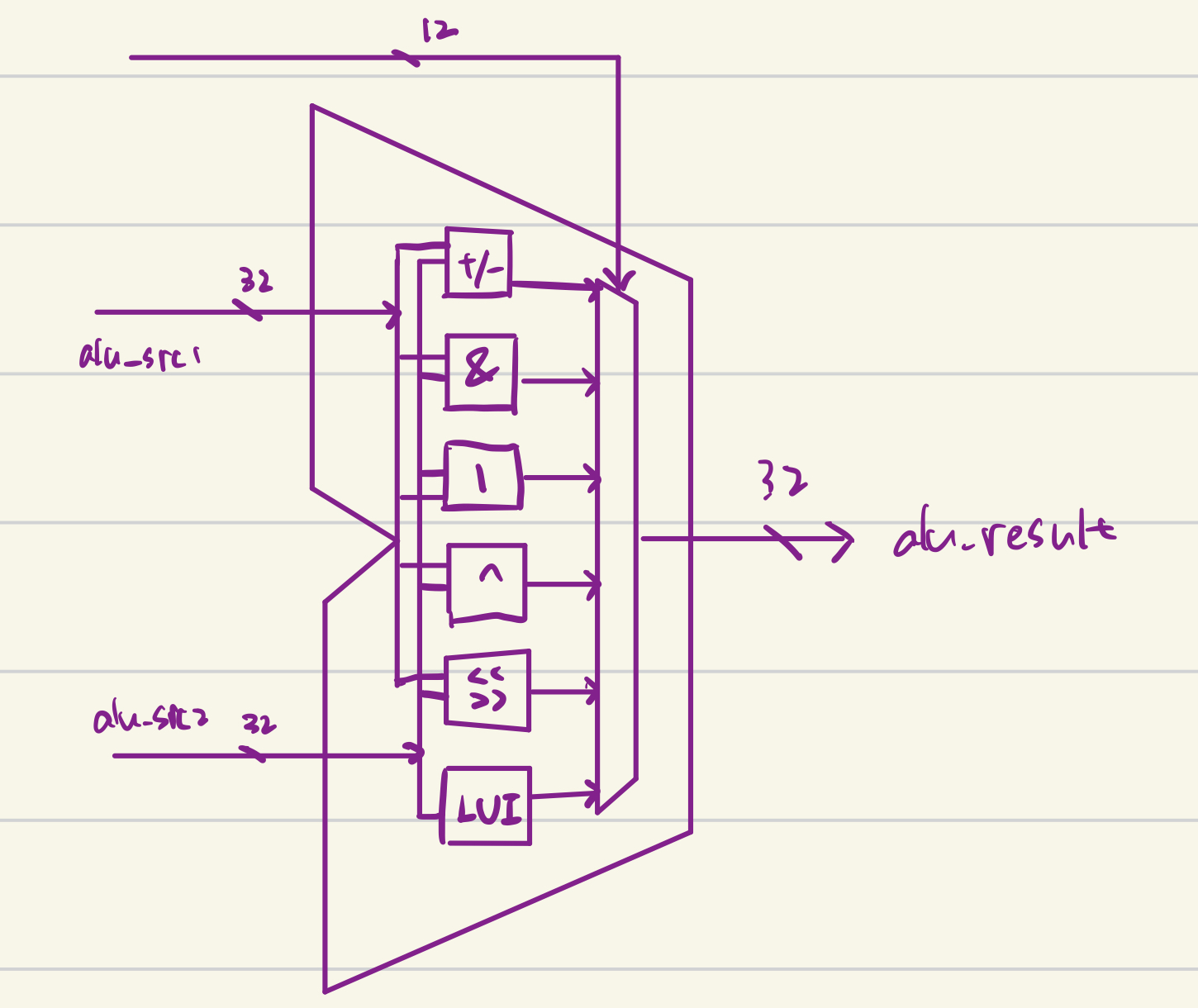
请结合实验指导手册中的实验四(ALU模块实现实验)完成功能改进，实现一个能够完成更多运算的 ALU，注意以下几点:

1、原始代码中只有表5.1中的11种运算，请另外补充至少三种不同类型运算(比较运算、位运算、数据加载运算等)。

2、实验报告中原理图参考5.3，只画出补充的运算部分即可，报告中展示的结果应位补充运算的计算结果。

3、本次实验报告不需要有仿真结果，但需要有实验箱上箱验证的照片，同样，针对照片中的数据需要解释说明。

1. **实验原理图**



1. **实验步骤**

首先，原本的12种操作分别与控制信号alu\_control的0 - 11位绑定

**1、加减法运算**：直接使用实验一中完成的加法器模块。

**2、有符号数小于置位**：利用减法结果比较，其比较的真值表如下：



从而写出其表达式：

slt\_result=(alu\_src1[31]&~alu\_src2[31])|(~(alu\_src1[31]^alu\_src2[31])&adder\_result[31])

**3、无符号数小于置位**：对于32位无符号比较的小于置位，可在其高位前填0组合为33位正数的比较，即 {1’b0,src1}和{1’b0,src2}的比较，最高位符号位为0。对比表5.2可知，对于正数的比 较，只要减法结果的符号位为1，则表示小于。而33位正数相减，其结果的符号位最终 可由32位加法的cout+1’b1得到。故无符号32位比较小于置位运算结果表达式 为:sltu\_result=~adder\_cout。

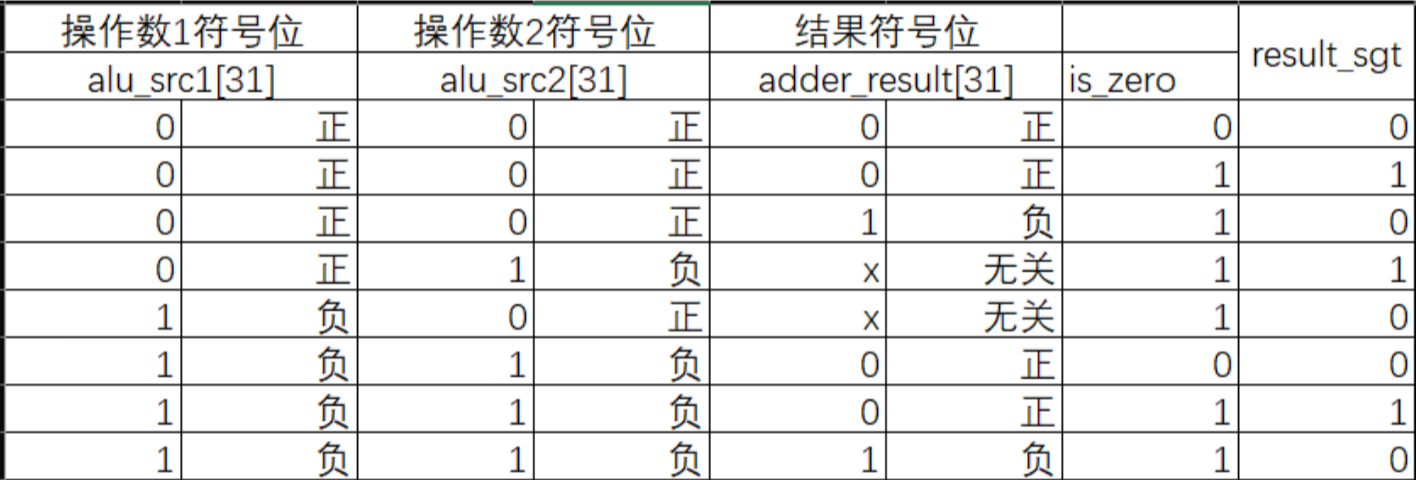
**4、按位与、按位或非、按位或、按位异或**：可直接借助 verilog 语言运算函数&、\和^实现。

**5、逻辑左移、逻辑右移、算数右移**：移位分三步进行，第一步根据移位量低2位即[1:0]位做第一次移位，第二步在第一次移位基础上根据移位量[3:2]位做第二次移位，第三步在第二次移位基础上根据移位量[4]位做第三次移位。

**6、高位加载**：将高16位数放入寄存器的高16位，低16位用0填充。

**7、新增的操作sgt，nxor，hui**：分别与控制信号alu\_control的C、D、E位绑定

**Sgt**：有符号数大于置位运算与有符号数的小于置位有所不同，需要考虑两数相等的情况，在 模块实际设计过程中，增加 is\_zero 变量判断运算结果是否为0。真值表如下如所示:



从而得到表达式：

wire is\_zero;

assign is\_zero=|adder\_result;

assign sgt\_result[31:1] = 31'd0;

assign sgt\_result[0] = ~((alu\_src1[31] & ~alu\_src2[31]) | (~(alu\_src1[31]^alu\_src2[31]) & adder\_result[31])) & is\_zero;

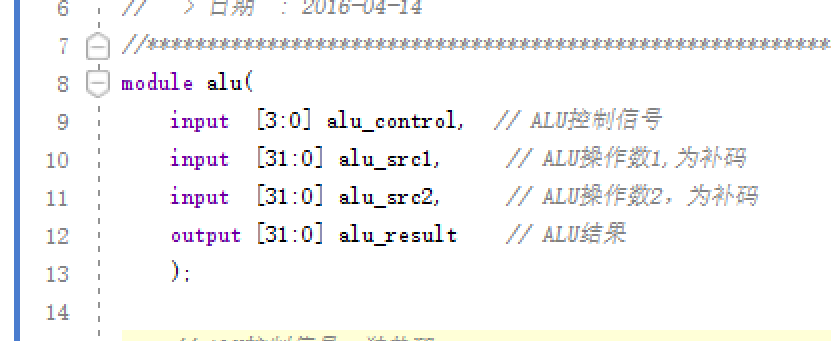
**Nxor：**对xor操作取反即可

**Hui：**低位加载将低16位操作数放入寄存器低16位，高16位用0补充。

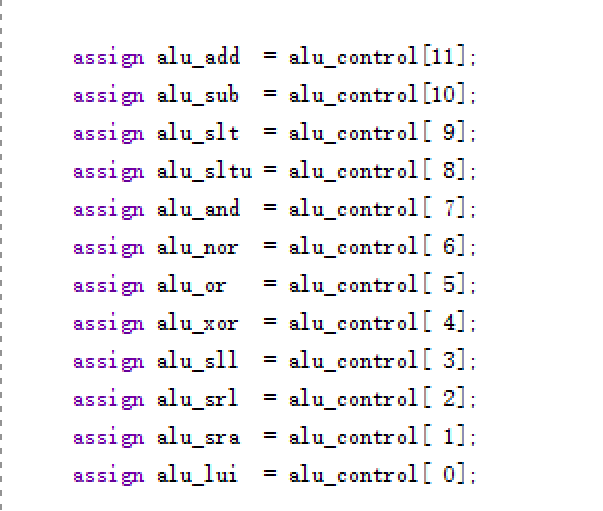
本次实验增加的三种运算：按位同或、大于置位、低位加载，代码如下：

首先，在alu\_module.v中进行修改：

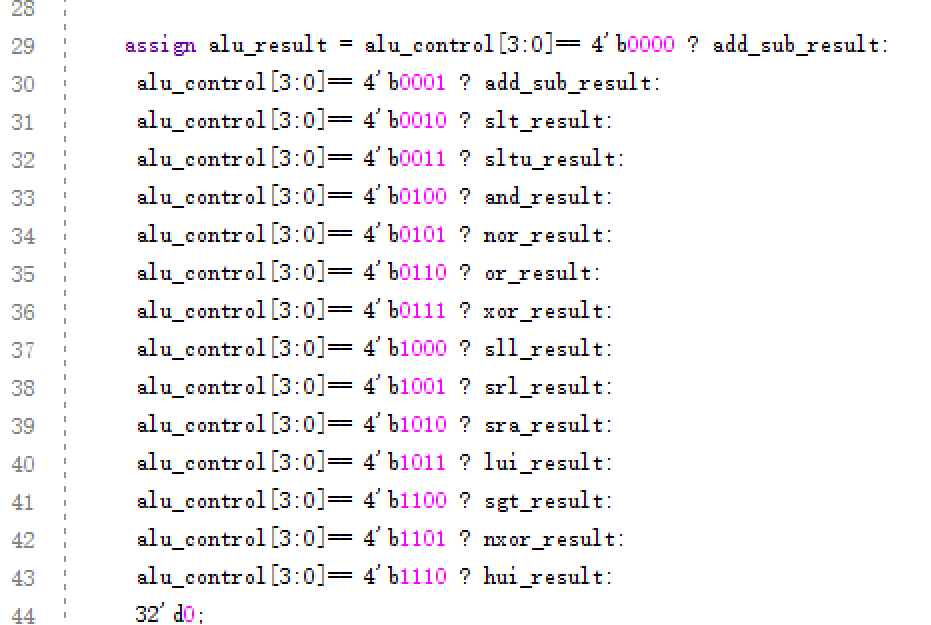
1. 我们更改了运算的数量，所以需要将alu\_control控制信号的带宽进行调整修改，此处改为4位编码：



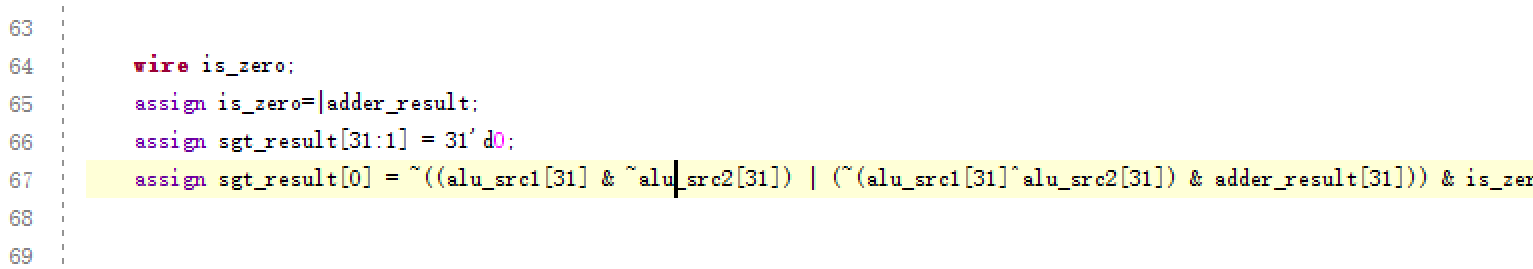
1. 由于我们使用了四位编码的使能信号，我们将原来的如下部分，每个操作单独一个使能信号的部分删除：



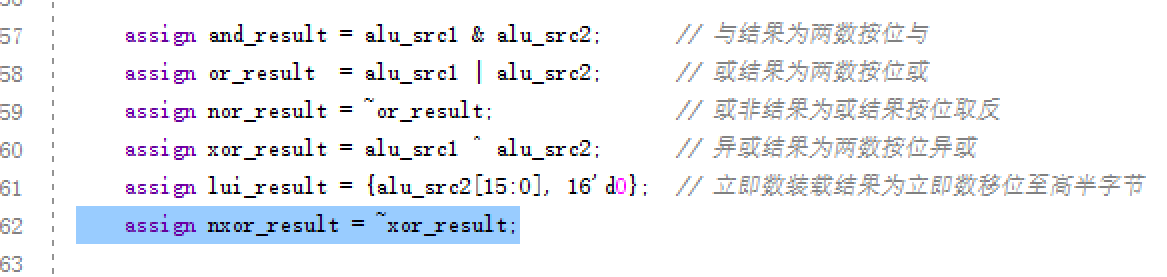
并调整修改如下：直接通过 alu\_control 的值来对最终模块输出的结果进行赋值。



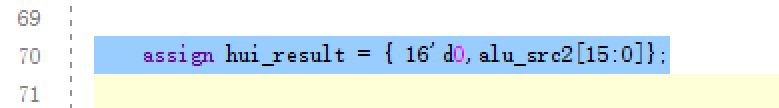
1. Sgt模块运算（原理见前）：



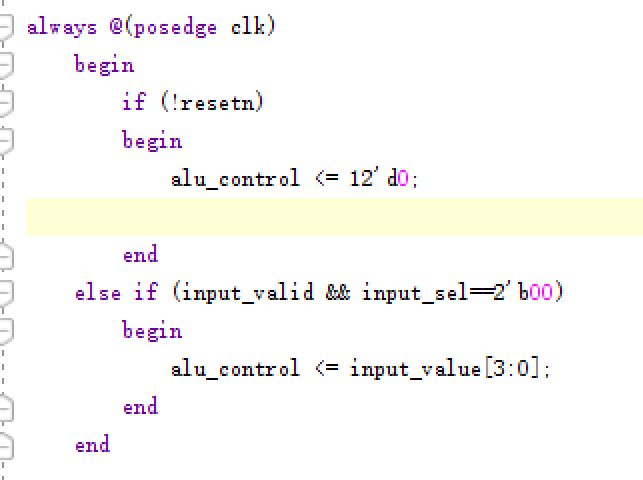
1. Nxor模块运算：



1. Hui模块运算（原理见前）：



然后，进行顶层模块的修改，由于与顶层模块有关联的只有alu\_control信号的位宽，因此我们修改其位宽：



1. **实验结果分析**

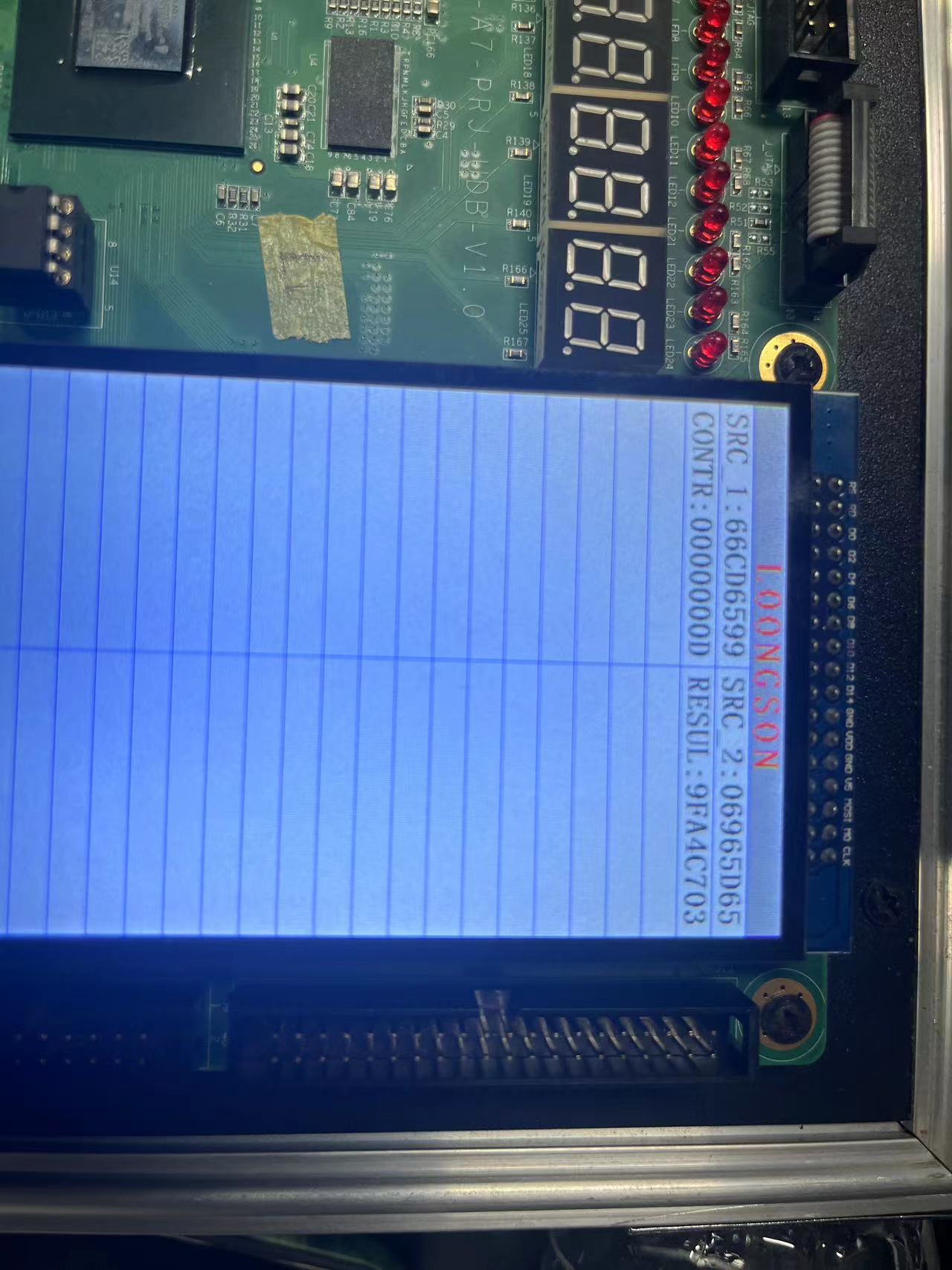
上箱验证：

对有符号数大于置位操作进行正确性验证，将控制信号 control 设置为`C，输入操作数：



由于0x66CD6599>0x06965D65，因此置位为1，RESULT=1，结果正确。

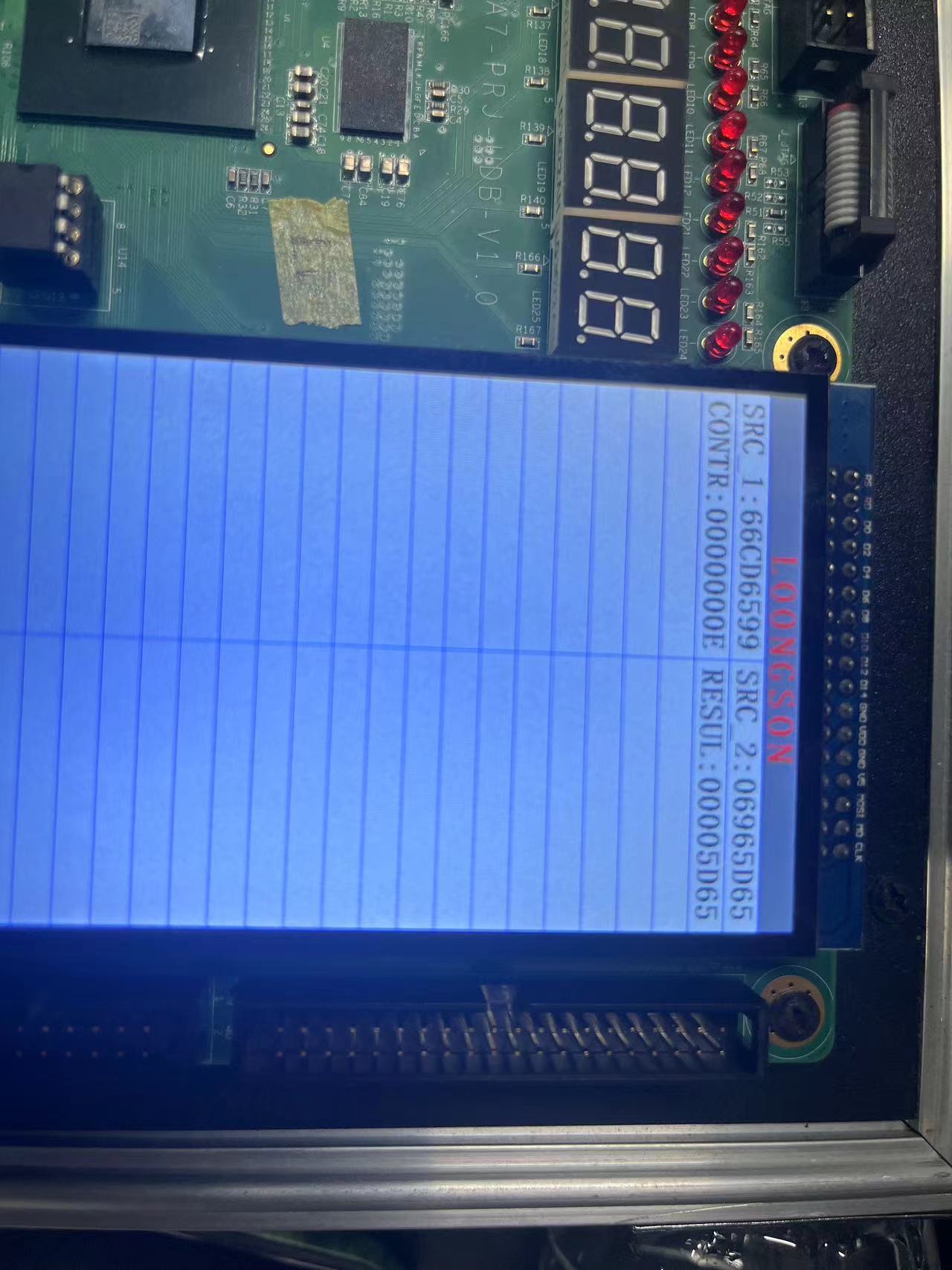
对nxor操作进行正确性验证，将控制信号 control 设置为`D，输入操作数:





由计算机验证，0x66CD6599 nxor 0x06965D65 = 0x9FA4C703，结果正确

对低位加载16位操作进行正确性验证，将控制信号 control 设置为 E ，输入操作数:



结果正确。

综上，结果均正确，实验成功。

1. **总结感想**

在本次实验中，我对ALU模块进行了复现与改进，通过增加新的运算功能，进一步深入理解了MIPS计算机中寄存器堆的原理和设计方法，并熟悉了MIPS指令结构及其源操作数和目的操作数的概念。

首先，复现了原始代码中的基本运算，并通过补充了三种新的运算类型：有符号数大于置位（sgt）、按位同或（nxor）、低位加载（hui）。这些运算的实现加深了我对Verilog语言的理解，还让我更好地掌握了数字电路设计中的位运算、比较运算及数据加载等操作的具体实现方法。

通过此次实验，我对ALU的整体设计与实现流程有了更加全面的认识。从初步设计，到具体代码实现，再到最终的上箱验证，每一个步骤都让我受益匪浅。尤其是在设计sgt运算时，通过判断两数的符号位及计算结果的符号位来确定最终结果，使我更深刻地理解了有符号数比较的复杂性及其实现方式。