组成原理课程第 四 次实报告

实验名称：ALU 模块实现

学号： 2213041 姓名： 李雅帆 班次： 李涛老师

一、实验目的

1. 熟悉 MIPS 指令集中的运算指令，学会对这些指令进行归纳分类。
2. 了解 MIPS 指令结构。
3. 熟悉并掌握 ALU 的原理、功能和设计。
4. 进一步加强运用 verilog 语言进行电路设计的能力。
5. 为后续设计 cpu 的实验打下基础。

二、实验内容说明

1. 学习 MIPS 指令集，熟知指令类型，了解指令功能和编码，归纳基础的 ALU 运算指令。
2. 归纳确定自己本次实验中准备实现的 ALU 运算，要求不实现定点乘除指令和浮点运算指令，要求至少实现 5 种 ALU 运算，其中要包含加减运算，其中减法在内部要转换为加法，与加法运算共同调用实验一里自己完成的加法模块去做。
3. 自行设计本次实验的方案，画出结构框图，大致结构框图如图 5.1。图 5.1 中的操作码位数和类型请自行设计，可以设计为独热码（一位有效编码）或二进制编码。比如，设计方案中预定实现 7 种 ALU 运算，则操作码采用独热码，则需 7bit 数据，每位单独指示一种运算；若采用二进制编码，则只用 3bit 数据位即可，但在需 ALU 内部先进行解码，才能确定 ALU 作何种运算。
4. 根据设计的实验方案，使用 verilog 编写相应代码。
5. 对编写的代码进行仿真，得到正确的波形图。
6. 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块。外围模块中需调用封装好的 LCD 触摸屏模块，显示 ALU 的两个源操作数、操作码和运算结果，并且需要利用触摸功能输入源操作数。操作码可以考虑用 LCD 触摸屏输入，也可以用拨码开关输入。
7.将编写的代码进行综合布局布线，并下载到试验箱中的 FPGA 板子上进行演示。
8.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。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）。

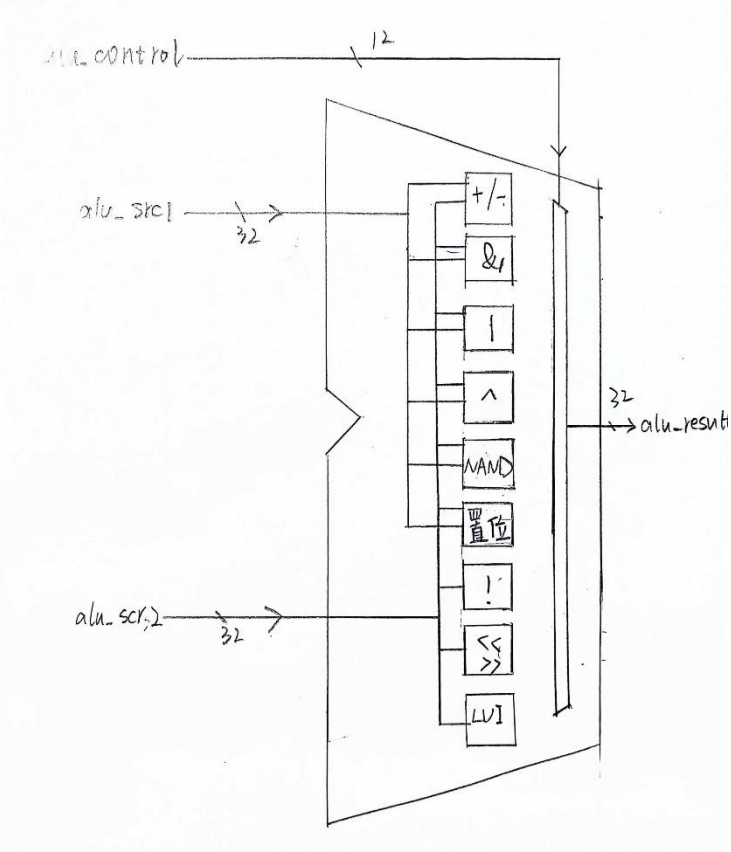
三、实验原理图

在原来的基础上将三个运算加入到 ALU 中：

①按位取反

②与非

③大于置位



四、实验步骤

1.分析：

（1）压缩 ALU 运算器的符号控制独热码至 4 位

（2）将三个运算加入到 ALU 中：

①按位取反

②与非

③大于置位

2.修改后的计算表格：

001	高位加载
002	算数右
003	逻辑右
004	逻辑左
005	异或
006	或
007	或非
008	与
009	有符号置位
00A	无符号置位
00B	减
00C	加

00D	按位取反
00E	与非
00F	大于则置位（无符号）

3.ALU 操作原理

（1）加减法

利用之前实验的 adder。

（2）置位操作

使用 adder 计算两个操作数的差，若为有符号数，根据差的符号位和两个操作数的符号位，由此化简真值表得到结果；对于 32 无符号位比较，在其最高位前填 0 作为 33 位正数比较。

(3)逻辑操作

按位与、或、非、异或等。

(4)左、右移

根据移动位数来按位左移和右移几位，注意算数左移和逻辑右移实际是一样的。

(5)高低位加载

读取指定操作数的高、低 16 位的值，使用 veilog 的切片操作即可。

4.代码修改

(1)压缩控制信号至 4 位

```
module alu(
    input  [3:0] alu_control,  // ALU控制信号，改为4位
    input  [31:0] alu_src1,    // ALU操作数1，为补码
    input  [31:0] alu_src2,    // ALU操作数2，为补码
    output [31:0] alu_result   // ALU结果
    );
```

(2)根据 4 位控制信号进行独热编码，且增加三个独热码

①增加三个独热码

```
// ALU控制信号，独热码
wire alu_add;   //加法操作
wire alu_sub;   //减法操作
wire alu_slt;   //有符号比较，小于置位，复用加法器做减法
wire alu_sltu;  //无符号比较，小于置位，复用加法器做减法
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;   //算术右移
wire alu_lui;   //高位加载
wire alu_not;   //按位取反
wire alu_nand;  //按位与非
wire alu_sgt;   //有符号比较，大于置位，复用加法器做减法
```

②4 位控制信号转为独热码

```
assign alu_add  = alu_control==4'b0001;
assign alu_sub  = alu_control==4'b0010;
assign alu_slt  = alu_control==4'b0011;
assign alu_sltu = alu_control==4'b0100;
assign alu_and  = alu_control==4'b0101;
assign alu_nor  = alu_control==4'b0110;
assign alu_or   = alu_control==4'b0111;
assign alu_xor  = alu_control==4'b1000;
assign alu_sll  = alu_control==4'b1001;
assign alu_srl  = alu_control==4'b1010;
assign alu_sra  = alu_control==4'b1011;
assign alu_lui  = alu_control==4'b1100;

assign alu_not  = alu_control==4'b1101;
assign alu_nand = alu_control==4'b1110;
assign alu_sgt  = alu_control==4'b1111;
```

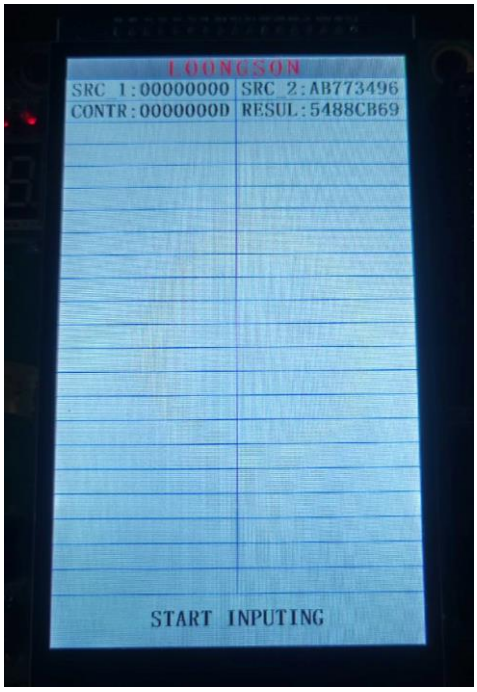
(3)增加三个操作代码

```
assign not_result = ~ alu_src2;                 //按位取反操作对源操作数2进行按位取反
assign nand_result = ~and_result;               //与非结果为与结果取反
```

```
//sgt结果
    //adder_src1[31] adder_src2[31] adder_result[31]
    //      0            1          X(0或1)       "正-负"，显然大于成立
    //      0            0            1           相减为负，说明不大于（实际上小于）
    //      0            0            0           相减为正（或0），说明大于或等于
    //      1            1            1           相减为负，说明不大于（实际上小于）
    //      1            1            0           相减为正（或0），说明大于或等于
    //      1            0          X(0或1)       "负-正"，显然大于不成立
    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]))
                            & (|(alu_src1-alu_src2)); //对slt的判断按位取反后再去掉两个操作数相等的情况
```

五、实验结果分析

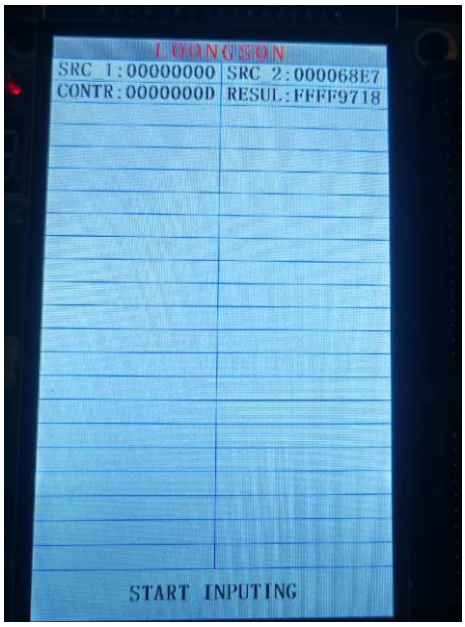
1.按位取反运算



计算器运算结果：

NOT(AB773496)

FFFF FFFF 5488 CB69

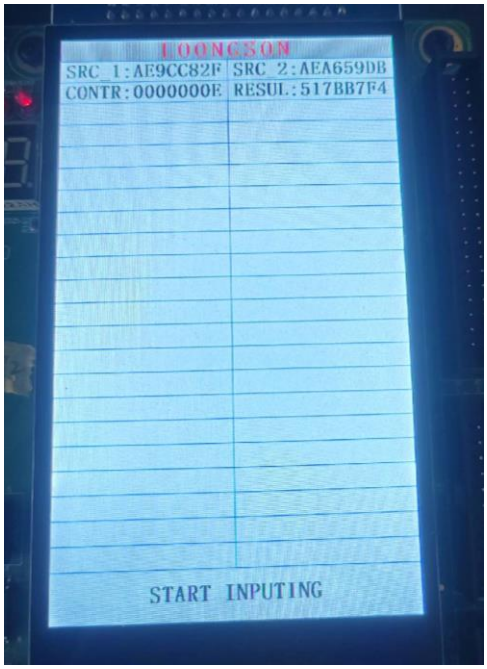


计算器运算结果：

NOT(68E7)

FFFF FFFF FFFF 9718

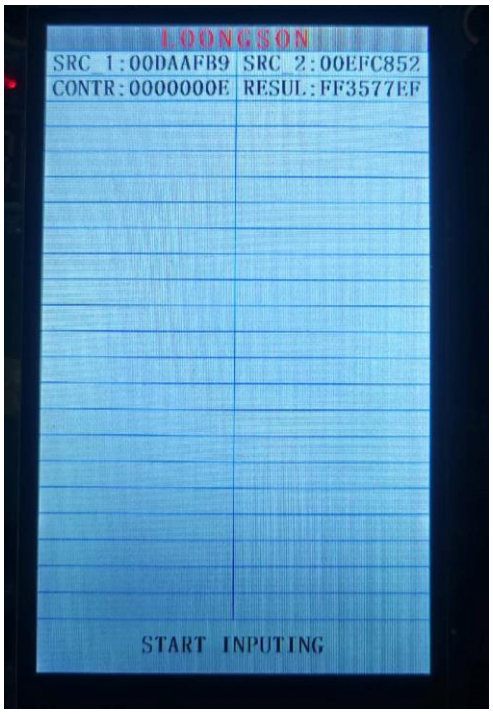
2.与非运算



计算器运算结果：

AE9CC82F NAND AEA659DB =

FFFF FFFF 517B B7F4



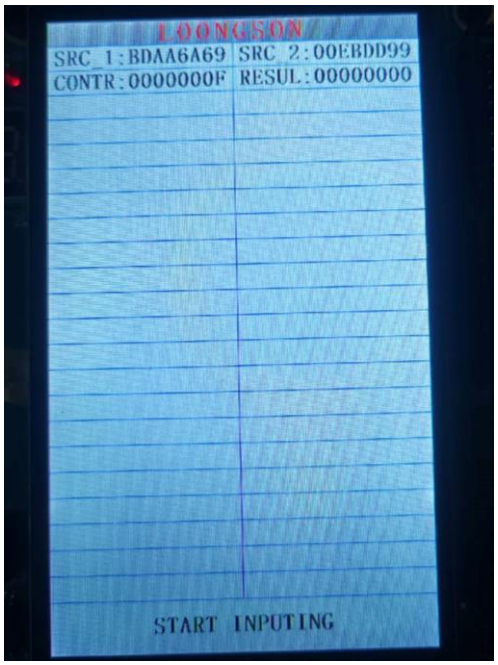
计算器运算结果：

AAFB9 NAND EFC852 =

FFFF FFFF FFF5 77EF

3. 大于则置位

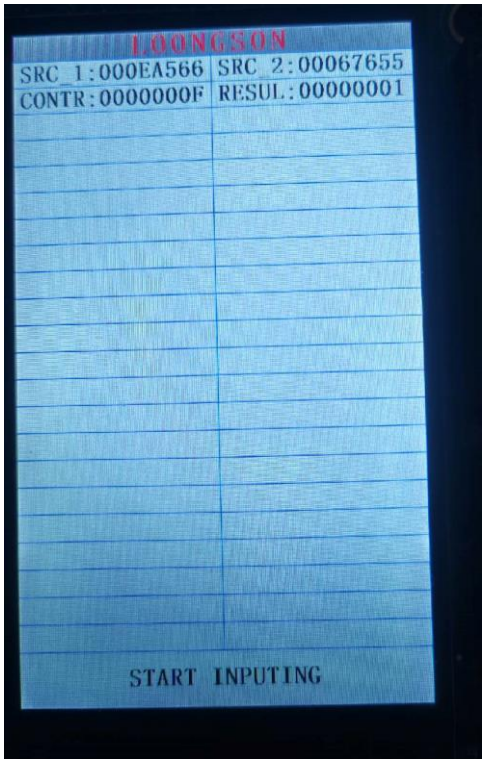
无符号数组 SCR_1 大于 SCR_2，则置位 1



alu_src1 = BDAA6A69 (负数)

alu_src2 = 00EBDD99 (正数)

alu_src1 小于 alu_src2,因此，根据代码中的逻辑：sgt_result[0] 为 0，表示 alu_src1 小于或等于 alu_src2。



alu_src1 和 alu_src2 都是正数，alu_src1 大于 alu_src2，sgt_result[0]的值为 1。

六、总结感想

通过学习 MIPS 指令集，我深入了解了指令的种类、功能和编码方式。

其次，通过设计和实现 ALU，我加深了对 ALU 原理和功能的理解，尤其是在设计 ALU 时考虑了与非、按位取反以及其他基本运算的实现方式。

另外，在设计外围模块时，我也学会了如何与其他模块进行交互，并且利用 LCD 触摸屏模块和拨码开关实现了用户输入和输出的功效。