组成原理实验课程第 四 次实验报告

实验名称	ALU 模块实现			班级	李涛老师
学生姓名	辛浩然	学号	2112514	指导老师	董前琨
实验地点	实验楼 A 区 304			实验时间	2023/5/9

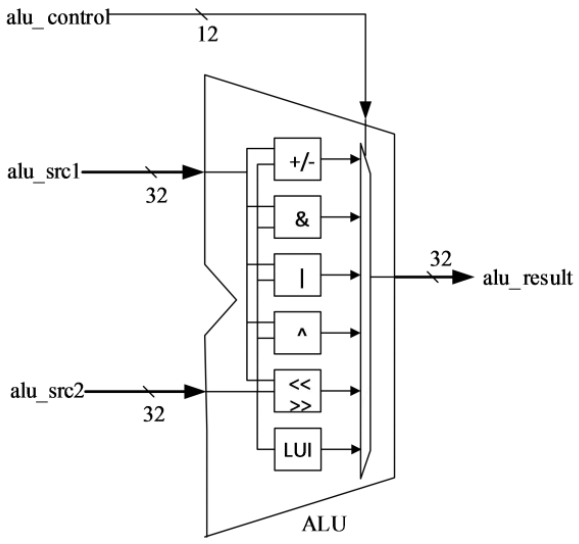
1 实验目的

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。

2 实验内容说明

1. 将原有的操作码进行位压缩，调整操作码控制信号位宽为 4 位。
2. 操作码调整成 4 位之后，在原有 11 种运算的基础之上，自行补充 3 种不同类型的运算，操作码和运算自行选择，需要上实验箱验证计算结果。

3 实验原理图



4 实验步骤

4.1 代码实现

增加三种运算。分别为 seq：等于则置位；xnor：按位同或；lli：低位加载立即数。

```
module alu(
    input  [3:0] alu_control,   // ALU控制信号
    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
    );

    // ALU控制信号，独热码
    // 增加三个运算
    wire alu_seq;               // 等于则置位
    wire alu_xnor;              // 按位同或
    wire alu_lli;               // 低位加载

    // 从0000开始，4位不同值控制不同运算
    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_seq  = (alu_control == 4'b1101);
    assign alu_xnor = (alu_control == 4'b1110);
    assign alu_lli  = (alu_control == 4'b1111);

    // 存放增加的运算的运算结果
    wire [31:0] xnor_result;
    wire [31:0] seq_result;
    wire [31:0] lli_result;

    // 立即数装载结果为立即数移位至低半字节
    assign lli_result = {16'd0,alu_src2[15:0]};
```

```
    //同或结果为异或结果按位取反
    assign xnor_result = ~xor_result;

    // 等于则置位，如果用加法器做减法结果是32位全0，那么置为1；否则为0
    assign seq_result[31:1] = 31'd0;
    assign seq_result[0]    = (adder_result == 32'h00000000) ? 1 : 0;

    // 选择相应结果输出
    assign alu_result   =   (alu_add|alu_sub) ? add_sub_result[31:0] :
                            alu_slt           ? slt_result :
                            alu_sltu          ? sltu_result :
                            alu_and           ? and_result :
                            alu_nor           ? nor_result :
                            alu_or            ? or_result  :
                            alu_xor           ? xor_result :
                            alu_sll           ? sll_result :
                            alu_srl           ? srl_result :
                            alu_sra           ? sra_result :
                            alu_lui           ? lui_result :
                            alu_xnor          ? xnor_result:
                            alu_lli           ? lli_result :
                            alu_seq           ? seq_result:
                            32'd0;
```

alu_control 修改为 4 位后，display 文件中的对应部分也要修改：

```
//-----{调用ALU模块}begin
    reg   [3:0] alu_control;   // ALU控制信号
    reg   [31:0] alu_src1;     // ALU操作数1
    reg   [31:0] alu_src2;     // ALU操作数2
    wire  [31:0] alu_result;   // ALU结果
    alu alu_module(
        .alu_control(alu_control),
        .alu_src1   (alu_src1   ),
        .alu_src2   (alu_src2   ),
        .alu_result (alu_result )
    );
//-----{调用ALU模块}end
```

```
//-----{从触摸屏获取输入}begin
    //当input_sel为00时，表示输入数控制信号，即alu_control
    always @(posedge clk)
    begin
        if (!resetn)
```

```
        begin
            alu_control <= 4'd0;
        end
        else if (input_valid && input_sel==2'b00)
        begin
            alu_control <= input_value[3:0];
        end
    end
```

4.2 实验箱结果

4.2.1 原代码验证

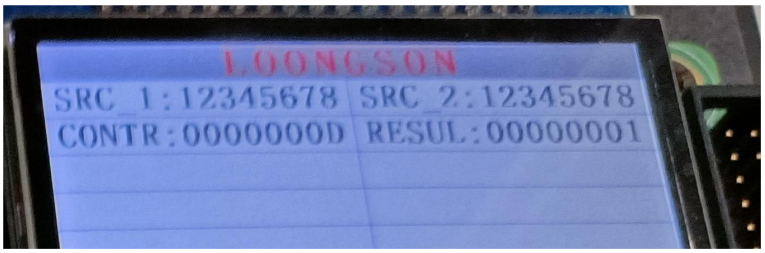
原代码验证结果如下表所示

控制信号	ALU 操作	SRC_1	SRC_2	RESULT
800	加法	33333333	99999999	CCCCCCCC
400	减法	23456789	12345678	11111111
200	有符号数比较，小于则置位	80000000	FFFFFFFF	1
		12345678	80000000	0
		11111111	22222222	1
100	无符号数比较，小于则置位	81111111	11111111	0
		FFFFF239	FFFFFFF7	1
80	按位与	84218421	99999999	80018001
40	按位或非	84218421	12481248	69966996
20	按位或	84218421	12481248	96699669
10	按位异或	12345678	FFFFFFFF	EDCBA987
8	逻辑左移	4	84211248	42112480
4	逻辑右移	4	84211248	08421124
2	算术右移	4	84211248	F8421124
1	低位加载	0	1235677E	0000677E

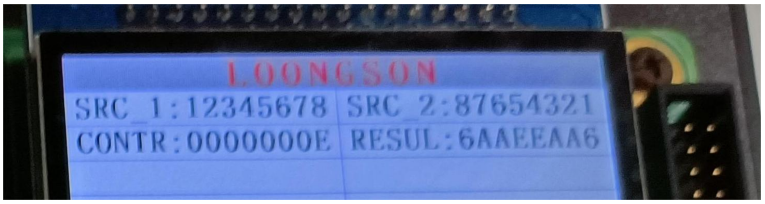
4.2.2 修改后代码验证

控制信号	ALU 操作	SRC_1	SRC_2	RESULT
1	加法	33333333	AAAAAAAA	DDDDDDDD
2	减法	23456789	12345678	11111111
3	有符号数比较，小于则置位	80000000	FFFFFFFF	1
		12345678	80000000	0
		11111111	22222222	1
4	无符号数比较，小于则置位	81111111	11111111	0
		FFFFF239	FFFFFFF7	1
5	按位与	84218421	99999999	80018001
6	按位或非	84218421	12481248	69966996
7	按位或	84218421	12481248	96699669
8	按位异或	12345678	FFFFFFFF	EDCBA987
9	逻辑左移	8	84211248	21124800
A	逻辑右移	8	84211248	00842112
B	算术右移	8	84211248	FF842112
C	高位加载	0	84211248	12480000
D	等于则置位	12345678	12345678	1
		12345678	11111111	0
E	按位同或	12345678	87654321	6AAEEAA6
F	低位加载	0	1235677E	0000677E

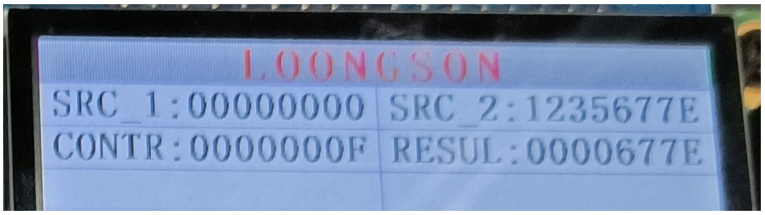
新增三种运算上箱图示：



ALU 控制信号为 D 时，进行等于则置位运算。输入操作数 1 为 12345678，输入操作数 2 为 12345678，相等置位，结果为 1，结果正确。



ALU 控制信号为 E 时，进行同或操作。输入操作数 1 为 12345678，输入操作数 2 为 87654321，同或后结果为 6AAEEAA6，结果正确。



ALU 控制信号为 F 时，进行低位加载操作。输入操作数 1 为 0，输入操作数 2 为 1235677E，结果为 0000677E，加载了低位，结果正确。

5 总结感想

通过本次实验：

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