组成原理实验课程第_四_次实报告

实验名称	ALU 模块实现			班级	张金老师班
学生姓名	蒋薇	学号	2110957	指导老师	董前琨
实验地点	实验楼 A306		实验时间	2023.4.24	

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、本次实验不用仿真波形，直接上实验箱验证即可。注意改进实验上实验箱验证时，操作码应该已经压缩到 4 位位宽。

4、实验报告中的原理图就用图 5.3 即可，不再是顶层模块图。实验报告中应该有两个表，第一个表为验证实验初始的 11 种运算，表中列出操作码、操作数和运算结果；第二个表是改进实验后的 11+3 种运算的验证，表中列出操作码、操作数和运算结果。注意自行添加的三种运算还需要附上实验箱验证照片。

3、 实验原理图

ALU 模块的原理图如下：

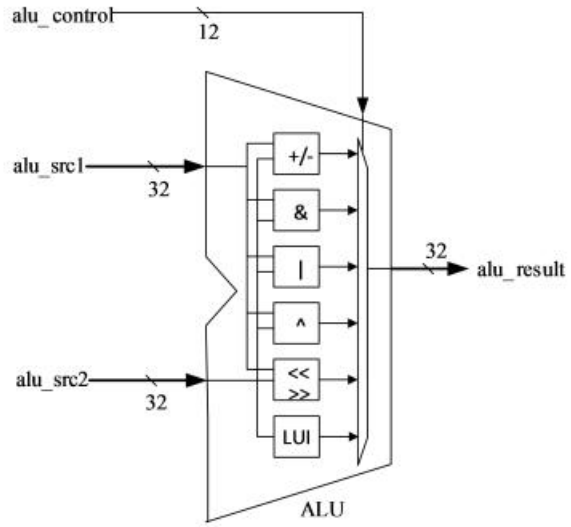


图 5.3 ALU 的原理图

4、 实验步骤

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。

```
//input  [11:0] alu_control,  //
input [3:0] alu_control1,
input  [31:0] alu_src1,      // AL
input  [31:0] alu_src2,      // AL
output [31:0] alu_result     // AL
```

增加三个运算：

```
//算术左移
wire alu_sla;
//按位与或
wire alu_xand;
//高位加载8位
wire alu_load;
```

```
assign alu_add1 = alu_control1[0]&~alu_control1[1]&~alu_control1[2]&~alu_control1[3];
assign alu_sub1  = ~alu_control1[0]&~alu_control1[1]& alu_control1[2]&~alu_control1[3];
assign alu_slt1 = ~alu_control1[0]&~alu_control1[1]&alu_control1[2]&alu_control1[3];
assign alu_sltu1 = ~alu_control1[0]&alu_control1[1]&~alu_control1[2]&~alu_control1[3];
assign alu_and1  = ~alu_control1[0]&alu_control1[1]&~alu_control1[2]&alu_control1[3];
assign alu_nor1  = ~alu_control1[0]&alu_control1[1]&alu_control1[2]&~alu_control1[3];
assign alu_or1   = ~alu_control1[0]&alu_control1[1]&alu_control1[2]&alu_control1[3];
assign alu_xor1  = alu_control1[0]&~alu_control1[1]&~alu_control1[2]&~alu_control1[3];
assign alu_sll1  = alu_control1[0]&~alu_control1[1]&~alu_control1[2]&alu_control1[3];
assign alu_srl1  = alu_control1[0]&~alu_control1[1]&alu_control1[2]&~alu_control1[3];
assign alu_sra1  = alu_control1[0]&~alu_control1[1]&alu_control1[2]&alu_control1[3];
assign alu_lui1  = alu_control1[0]&alu_control1[1]&~alu_control1[2]&~alu_control1[3];
assign alu_sla = alu_control1[0]&alu_control1[1]&~alu_control1[2]&alu_control1[3];
assign alu_xand = alu_control1[0]&alu_control1[1]&alu_control1[2]&~alu_control1[3];
assign alu_load = alu_control1[0]&alu_control1[1]&alu_control1[2]&alu_control1[3];
```

```
wire [31:0] sla_result;
wire [31:0]xand_result;
wire[31:0] load_result;
```

实现指令功能的代码：

```
//与非
assign xand_result = ~and_result;
//低八位加载
assign load_result = {alu_src2[7:0],24'd0};
//算术左移与逻辑左移相相似
```

```
//算术左移
wire [31:0] sll_step1;
wire [31:0] sll_step2;
assign sll_step1 = {32{shf_1_0 == 2'b00}} & alu_src2
                 | {32{shf_1_0 == 2'b01}} & {alu_src2[30:0], 1'd0}
                 | {32{shf_1_0 == 2'b10}} & {alu_src2[29:0], 2'd0}
                 | {32{shf_1_0 == 2'b11}} & {alu_src2[28:0], 3'd0};
assign sll_step2 = {32{shf_3_2 == 2'b00}} & sll_step1
                 | {32{shf_3_2 == 2'b01}} & {sll_step1[27:0], 4'd0}
                 | {32{shf_3_2 == 2'b10}} & {sll_step1[23:0], 8'd0}
                 | {32{shf_3_2 == 2'b11}} & {sll_step1[19:0], 12'd0};
assign sll_result = shf[4] ? {sll_step2[15:0], 16'd0} : sll_step2;
```

```
alu_lui     ? lui_result :
alu_xand    ? xand_result:
alu_sla     ? sla_result:
alu_load    ? load_result:
32'd0;
```

移位器，选择相应的结果输出：

（分布介绍依次完成了哪些代码修改，从而实现了什么样的功能）

5、 实验结果分析

（仿真结果截图或者实验箱运行结果拍照，注意需要对实验结果进行分析，输入是什么，输出是什么，结果是什么，是否验证了正确性）

实验初始的 11 种运算，表中列出操作码、操作数和运算结果

下表控制信号为 12 位二进制数，

从最下一行向上十进制表示分别为：

1 2 4 8 16 32 64 128 256 512 1024 2048 0

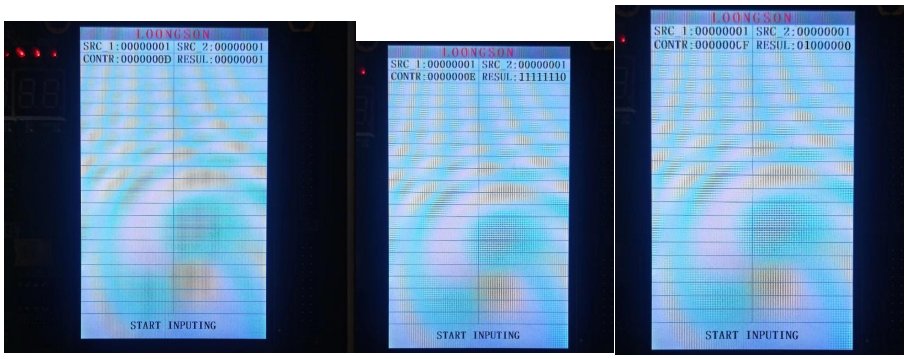
上试验箱验证时，控制信号将二进制转换为八位十六进制输入

1 2 4 8 10 20 40 80 100 200 400 800 0

	控制信号	ALU 操作	源操作数 1	源操作数 2	控制信号输入	结果
	000000000000	无	00000000	00000000	00000000	00000000
0	100000000000	加法	00000009	00000001	00000800	0000000A
1	010000000000	减法	00000009	00000001	00000400	00000008
2	001000000000	有符号比较，小于置位	00000009	00000001	00000200	00000000
3	000100000000	无符号比较，小于置位	00000009	00000001	00000100	00000000
4	000010000000	按位与	00000001	00000001	00000080	00000001
5	000001000000	按位或非	00000001	00000001	00000040	FFFFFFFE
6	000000100000	按位或	00000001	00000001	00000020	00000001
7	000000010000	按位异或	00000001	00000001	00000010	00000000
8	000000001000	逻辑左移	00000001	00000001	00000008	00000010
9	000000000100	逻辑右移	00000001	00000001	00000004	00000000
10	000000000010	算术右移	00000001	00000001	00000002	00000000
11	000000000001	高位加载	00000001	00000001	00000001	00010000

改进实验后的 11+3 种运算的验证，表中列出操作码、操作数和运算结果。

操作码控制信号位宽为 4 位。



二进制转换为十六进制

依次为：

1 2 3 4 5 6 7 8 9 A B C D E F

	控制信号	ALU 操作	源操作数 1	源操作数 2	控制信号输入	结果
	0000	无	00000000	00000000	00000000	00000000
0	0001	加法	00000009	00000001	00000001	0000000A
1	0010	减法	00000009	00000001	00000002	00000008
2	0011	有符号比较，小于置位	00000009	00000001	00000003	00000000
3	0100	无符号比较，小于置位	00000009	00000001	00000004	00000000
4	0101	按位与	00000001	00000001	00000005	00000001
5	0110	按位或非	00000001	00000001	00000006	FFFFFFFE
6	0111	按位或	00000001	00000001	00000007	00000001
7	1000	按位异或	00000001	00000001	00000008	00000000
8	1001	逻辑左移	00000001	00000001	00000009	00000010
9	1010	逻辑右移	00000001	00000001	0000000A	00000000
10	1011	算术右移	00000001	00000001	0000000B	00000000
11	1100	高位加载	00000001	00000001	0000000C	00010000
12	1101	算术左移	00000001	00000001	0000000D	00000010
13	1110	按位与非	00000001	00000001	0000000E	11111110
14	1111	低位加载	00000001	00000001	0000000F	01000000

6、 总结感想

（说说本次实验的总结感想）

MIPS 常见的指令：

1、Load/Store 指令：lw、sw、lb、sb 等

2、算数指令：add、sub、addi、subi、mult、div 等

3、逻辑指令：and、or、xor、nor、andi、ori、xori 等

4、移位指令：sll、srl、sra、sllv、srlv、srav 等

5、分支指令：beq、bne、blt、ble、bgt、bge 等

6、跳转指令：j、jal、jr 等

7、协处理器指令：mfc0、mtc0 等

8、系统调用指令：syscall、break 等。