**组成原理实验课程第4次实报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | ALU模块实现 | | | 班级 | 李涛 |
| 学生姓名 | 张丛 | 学号 | 2113662 | 指导老师 | 董前琨 |
| 实验地点 | A区308 | | 实验时间 | 2023.05.09 | |

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。

1. **实验内容说明**

1. 做好预习：

1） 熟知指令类型，了解指令功能和编码；

2） 归纳基础的 ALU 运算指令，确定自己准备实现的 ALU 运算；

3） 设计本次实验的方案，列出准备实现的 ALU 运算和操作码的编码；

4） 在课前画好实验方案的设计框图，即补充完善图 5.1；

5） 如果对 FPGA 板了解的话，可确定设计中与 FPGA 板上交互的接口，画出

包含外围模块的整体设计框图，即补充完善图 5.2。

2. 实验实施：

1） 确认 ALU 模块的设计框图的正确性；

2） 编写 verilog 代码；

3） 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材

料；

4） 完成调用 ALU 模块的外围模块的设计，并编写代码；

5） 对代码进行综合布局布线下载到试验箱里 FPGA 板上，进行上板验证。

3. 实验检查：

1） 完成上板验证后，让指导老师或助教进行检查，进行现场演示。先说明

自己实现的 ALU 运算类型，按照检查人员的要求，对特定源操作数进行特定运

算操作，检查运算结果的正确性，可对演示结果进行拍照作为实验报告结果一

项的材料。Page

62

4. 实验报告的撰写：

1） 实验结束后，需按照规定的格式完成实验报告的撰写。

1. **实验原理图**



原理为alu\_src1以及alu\_src2代表两个源操作数的输入，alu\_control信号控制ALU最后输出何种运算的结果。在输入两个源操作数之后，会将这两个源操作数赋值给所有运算单元，所有运算单元都会并行运算。只不过最后的输出运算结果由alu\_control控制。

1. **实验步骤**

1.alu.v修改：

①对输入的控制信号操作码进行位压缩，即调整位宽为4位：

module alu(

input [3:0] alu\_control,

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添加3种运算：低位加载；无符号比较，大于置位；按位同或。

// ALU控制信号，独热码

wire alu\_luidiwei; //低位加载

wire alu\_sltudayu; //无符号比较，大于置位

wire alu\_sor; //按位同或

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

③ 对应增加并修改assign语句为4位位宽形式：

assign alu\_luidiwei = (alu\_control==4'b1110)?1:0;

assign alu\_sltudayu = (alu\_control==4'b1101)?1:0;

assign alu\_sor = (alu\_control==4'b1100)?1:0;

assign alu\_add = (alu\_control==4'b1011)?1:0; //原来的11

assign alu\_sub = (alu\_control==4'b1010)?1:0;

assign alu\_slt = (alu\_control==4'b1001)?1:0;

assign alu\_sltu = (alu\_control==4'b1000)?1:0;

assign alu\_and = (alu\_control==4'b0111)?1:0;

assign alu\_nor = (alu\_control==4'b0110)?1:0;

assign alu\_or = (alu\_control==4'b0101)?1:0;

assign alu\_xor = (alu\_control==4'b0100)?1:0;

assign alu\_sll = (alu\_control==4'b0011)?1:0;

assign alu\_srl = (alu\_control==4'b0010)?1:0;

assign alu\_sra = (alu\_control==4'b0001)?1:0;

assign alu\_lui = (alu\_control==4'b0000)?1:0;//原来的0

④补齐表示运算结果的wire语句：

wire [31:0] add\_sub\_result;

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 [31:0] luidiwei\_result;

wire [31:0] sltudayu\_result;

wire [31:0] sor\_result;

⑤添加新运算的实现：

assign and\_result = alu\_src1 & alu\_src2; // 与结果为两数按位与

assign or\_result = alu\_src1 | alu\_src2; // 或结果为两数按位或

assign nor\_result = ~or\_result; // 或非结果为或结果按位取反

assign xor\_result = alu\_src1 ^ alu\_src2; // 异或结果为两数按位异或

assign lui\_result = {alu\_src2[15:0], 16'd0}; // 立即数装载结果为立即数移位至高半字节

assign luidiwei\_result = {16'd0,alu\_src2[15:0]}; // 立即数装载结果为立即数移位至低半字节，即将操作数2的低16位装载到结果的低16位，然后将结果的高16位补0。

assign sltudayu\_result = (adder\_result==32'b0)?32'd0:{31'd0, adder\_cout}; //无符号大于置位，使用三目运算符，如果低32位加法器运算结果全为0，就将结果置全0；若不为0，则可以直接由最后32位加法器的最高位进位来判断是置0还是置1。

assign sor\_result = ~(alu\_src1^alu\_src2); //同或结果，将异或结果取反即可

2.alu\_display.v修改：

修改前部分：

reg [11:0] alu\_control; // ALU控制信号

6'd3 :

begin

display\_valid <= 1'b1;

display\_name <= "CONTR";

display\_value <={20'd0, alu\_control};

end

修改后：

reg [3:0] alu\_control; // ALU控制信号

6'd3 :

begin

display\_valid <= 1'b1;

display\_name <= "CONTR";

display\_value <={28'd0, alu\_control};

end

即修改ALU控制信号的声明并补位28个0

3.alu.xdc修改（便于实验箱的调整验证）：

修改前部分：

#拨码开关连接，用于输入，依次为sw0,sw1

set\_property PACKAGE\_PIN Y6 [get\_ports input\_sel[0]]

set\_property PACKAGE\_PIN AD24 [get\_ports input\_sel[1]]

修改后：

set\_property PACKAGE\_PIN AD24 [get\_ports input\_sel[0]]

set\_property PACKAGE\_PIN AC21 [get\_ports input\_sel[1]]

1. **实验结果分析**

**表一：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | alu\_src1 | alu\_src2 | alu\_result | alu\_control |
| 加法 | 12345678 | ABCDEF01 | BE024579 | 11 |
| 减法 | FEDCBA98 | 76543210 | 88888898 | 10 |
| 有符号小于置位 | FFFFFFFF | 00000001 | 1 | 9 |
| 无符号小于置位 | FFFFFFFF | 00000001 | 0 | 8 |
| 按位与 | 55555555 | F0F0F0F0 | 50505050 | 7 |
| 按位或非 | AAAAAAAA | F0F0F0F0 | 05050505 | 6 |
| 按位或 | 12345678 | ABCDEF01 | BBFDFF79 | 5 |
| 按位异或 | 12345678 | ABCDEF01 | B99EB979 | 4 |
| 逻辑左移 | 9816E9A7 | 37B5516B | DAA8B580 | 3 |
| 逻辑右移 | 9816E9A7 | 37B5516B | 006F6AA2 | 2 |
| 算术右移 | 94177AA8 | A94127B9 | FFA94127 | 1 |
| 高位加载 | 94177AA8 | A94127B9 | 27B90000 | 0 |

**表二：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | alu\_src1 | alu\_src2 | alu\_result | alu\_control |
| 加法 | 12345678 | ABCDEF01 | BE024579 | 11 |
| 减法 | FEDCBA98 | 76543210 | 88888898 | 10 |
| 有符号小于置位 | FFFFFFFF | 00000001 | 1 | 9 |
| 无符号小于置位 | FFFFFFFF | 00000001 | 0 | 8 |
| 按位与 | 55555555 | F0F0F0F0 | 50505050 | 7 |
| 按位或非 | AAAAAAAA | F0F0F0F0 | 05050505 | 6 |
| 按位或 | 12345678 | ABCDEF01 | BBFDFF79 | 5 |
| 按位异或 | 12345678 | ABCDEF01 | B99EB979 | 4 |
| 逻辑左移 | 9816E9A7 | 37B5516B | DAA8B580 | 3 |
| 逻辑右移 | 9816E9A7 | 37B5516B | 006F6AA2 | 2 |
| 算术右移 | 94177AA8 | A94127B9 | FFA94127 | 1 |
| 高位加载 | 94177AA8 | A94127B9 | 27B90000 | 0 |
| 低位加载 | 0x211 | A85EEAA5 | 0xEAA5 | 14 |
| 无符号大于置位 | 54167EC7 | A856ECA2 | 0 | 13 |
| 按位同或 | 12345678 | 56781234 | BBB3BBB3 | 12 |

通过计算得出实验初始的11种运算都是正确的。

补充运算的实验箱图：

装载低位：



按位同或：



无符号的大于置位：



1. **总结感想**

亲自实验体会了ALU的运行原理。

熟悉了控制逻辑单元、并行计算、逻辑计算。

进一步熟悉verilog语言。