计算机系统结构实验 Lab03

朱凡悦 522031910547

May 2024

1 摘要

在 Lab03 中实现了简单的类 MIPS 处理器的几个重要部件：主控制器 (Ctr)、运算单元控制器 (ALUCtr) 以及算术逻辑运算单元 (ALU)，其作用分别是产生处理器所需要的各种控制信号、产生算术逻辑运算单元 (ALU) 所需要的控制信号以及根据运算单元控制器 (ALUCtr) 发出的信号执行相应的算术逻辑运算输出结果，并通过软件仿真的形式进行实验结果的验证。

目录

2 实验目的

1. 理解主控制部件或单元、ALU 控制器单元、ALU 单元的原理
2. 熟悉所需的 Mips 指令集
3. 使用 Verilog HD 设计与实现主控制器部件（Ctr）
4. 使用 Verilog 设计与实现 ALU 控制器部件（ALUCtr）
5. ALU 功能部件的实现
6. 使用 Vivado 进行功能模块的行为仿真

3 原理分析

3.1 主控制器 Ctr

主控制器需要对指令的最高 6 位的 OpCode 域进行解析，初步判断指令的类型并产生对应的处理器控制信号。指令分为 3 类：R 型指令；I 型指令中的 load 指令 (lw)，store 指令 (sw) 与 branch 指令 (beq)；J 型指令中的 jump 指令 (j)。本实验需要的控制信号如下：

1. RegDst: 目标寄存器的选择信号
2. AluSrc: 算术逻辑运算单元 (ALU) 的第二个操作数的来源
3. MemToReg: 写寄存器的数据来源
4. RegWrite: 写寄存器使能信号
5. MemRead: 内存读取使能信号
6. MemWrite: 内存写入使能信号
7. Branch: 条件跳转信号
8. AluOp: 发送给 ALUCtr 用于进一步解析运算类型的控制信号
9. Jump: 无条件跳转信号

Opcode	000000 R 型	000010 j	000100 beq	100011 lw	101011 sw
RegDst	1	0	x	0	X
ALUSrc	0	0	0	1	1
MemToReg	0	0	x	1	x
RegWrite	1	0	0	1	0
MemRead	0	0	0	1	0
MemWrite	0	0	0	0	1
Branch	0	0	1	0	0
AluOp	10	00	01	00	00
Jump	0	1	0	0	0

表 1: Ctr 产生的各种控制信号与指令的对应方式

对应真值表如下：

若出现其他不支持的指令，则将所有的控制信号均置为 0，使得该指令对数据没有影响，保证数据的正确性。

上述真值表的可以用 case 语句实现，根据 Opcode 不同输出不同信号。如下所示。

```
    always @(OpCode)
    begin
        case(OpCode)
        6'b000000:
        begin
            RegDst=1;
            AluSrc=0;
            MemToReg=0;
            RegWrite=1;
            MemRead=0;
            MemWrite=0;
            Branch=0;
            AluOp=2'b10;
            Jump=0;
         end
         6'b100011:
         begin
            RegDst=0;
            AluSrc=1;
            MemToReg=1;
```

```
            RegWrite=1;
            MemRead=1;
            MemWrite=0;
            Branch=0;
            AluOp=2'b00;
            Jump=0;
         end
         6'b101011:
         begin
            RegDst=0;
            AluSrc=1;
            MemToReg=0;
            RegWrite=0;
            MemRead=0;
            MemWrite=1;
            Branch=0;
            AluOp=2'b00;
            Jump=0;
         end
         6'b000100:
         begin
            RegDst=0;
            AluSrc=0;
            MemToReg=0;
            RegWrite=0;
            MemRead=0;
            MemWrite=0;
            Branch=1;
            AluOp=2'b01;
            Jump=0;
         end
         6'b000010:
         begin
            RegDst=0;
            AluSrc=0;
            MemToReg=0;
            RegWrite=0;
            MemRead=0;
            MemWrite=0;
            Branch=0;
            AluOp=2'b00;
            Jump=1;
         end
         default:
         begin
            RegDst=0;
```

```
            AluSrc=0;
            MemToReg=0;
            RegWrite=0;
            MemRead=0;
            MemWrite=0;
            Branch=0;
            AluOp=2'b00;
            Jump=0;
          end
       endcase
   end
```

3.2 运算单元控制器 ALUCtr

运算单元控制器 (ALUCtr) 结合指令的最后 6 位的 Funct 域和主控制器产生的 ALUOp 信号进行解析，给出最终的运算单元控制信号 ALUCtrOut。该信号传递给 ALU，实现对于 ALU 具体计算功能的选择。解析如下表所示。

Instruction	ALUOp	Funct Field	ALU Action	ALU Control Output
lw	00	xxxxxx	add	0010
sw	00	xxxxxx	add	0010
add	10	100000	add	0010
subtract	10	100010	substract	0110
AND	10	100100	and	0000
OR	10	100101	or	0001
slt	10	101010	set on less than	0111
beq	01	xxxxxx	subtract	0110

表 2: ALUCtr 解析方式

上述真值表可以用 casex 实现。根据 ALUOp 和 Funct 结合的不同结果，实现如下所示：

```
    always @(ALUOp or Funct)
    begin
        casex ({ALUOp,Funct})
        8'b00xxxxxx:ALUCtrOut=4'b0010;
        8'bx1xxxxxx:ALUCtrOut=4'b0110;
        8'b1xxx0000:ALUCtrOut=4'b0010;
```

```
        8'b1xxx0010:ALUCtrOut=4'b0110;
        8'b1xxx0100:ALUCtrOut=4'b0000;
        8'b1xxx0101:ALUCtrOut=4'b0001;
        8'b1xxx1010:ALUCtrOut=4'b0111;
        default:ALUCtrOut=4'b0000;
        endcase
    end
```

3.3 算术逻辑实现单元 ALU

ALU 主要根据由运算单元控制器产生的运算单元控制信号 ALUCtrOut 对输入的两个数执行对应的算术逻辑运算，输出运算的结果以及部分控制信号。其输出的一个重要信号为 zero 信号，用于与 branch 指令结合判断是否满足转移条件。当运算结果为 0 时置 zero 为 1，否则置 0。ALU 执行的算术逻辑运算类型与运算单元控制信号 ALUCtrOut 的对应方式如下所示：上述对应关系可以用 case 或 if else 语句实现，下面给出一种 if else

ALUCtrOut	算术逻辑运算类型
0000	and
0001	or
0010	add
0110	sub
0111	slt

语句实现的形式：

```
   always @(input1 or input2 or aluCtr)
  begin
    if (aluCtr==4'b0010) aluRes=input1+input2;
    else if (aluCtr==4'b0110)
    begin
       aluRes=input1-input2;
       if (aluRes==0) zero=1; else zero=0;
    end
    else if (aluCtr==4'b0000) aluRes=input1 & input2;
    else if (aluCtr==4'b0001) aluRes=input1 | input2;
    else if (aluCtr==4'b0111)
    begin
       aluRes=(input1<input2)?1:0;
       if (aluRes==0) zero=1; else zero=0;
```

```
    end
    else if (aluCtr==4'b1100) aluRes=~(input1 | input2);
    if (aluRes==0) zero=1; else zero=0;
  end
```

4 功能实现

基于上述原理，易于实现 ALU.v, ALUCtr.v, Ctr.v。实现后，生成 ALU_tb.v, ALUCtr_tb.v, Ctr_tb.v 激励文件用以仿真测试。

5 结果验证

5.1 主控制器 Ctr

在激励文件中对于 R 型指令、load 指令 (lw)、store 指令 (sw)、branch 指令 (beq)、jump 指令 (j) 进行了测试，测试结果如图 1 所示。

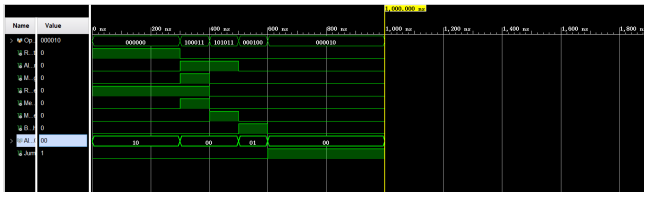

图 1: Ctr 模块仿真结果

5.2 运算单元控制器 ALUCtr

在激励文件中对于加法指令 (add)、减法指令 (sub)、逻辑与指令 (and)、逻辑或指令 (or)、小于时置位指令 (slt) 以及 load 指令 (lw)、store 指令 (sw) 进行了测试，测试结果如图 2 所示。

5.3 算术逻辑实现单元 ALU

在激励文件中对于加法、减法、逻辑与、逻辑或、小于时置位、逻辑或非等算数逻辑运算都进行了测试，测试结果如图 3 所示。或非运算的测试部分具体展示如下：

图 2: ALUCtr 模块仿真结果

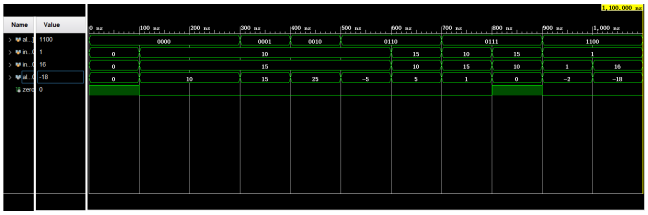

图 3: ALU 模块仿真结果

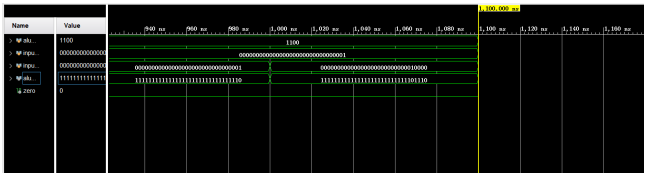

图 4: ALUCtr 模块或非运算仿真结果

6 总结与反思

本实验设计并实现了类 MIPS 处理器的三个重要组成部件：主控制器 (Ctr)、运算单元控制器 (ALUCtr) 以及算术逻辑运算单元 (ALU)，并且通过软件仿真模拟的方法验证了它们的正确性，为后面的单周期类 MIPS 处理器以及流水线处理器的实现奠定基础。