计算机系统结构实验报告　实验 3

颜培深 518030910094

2020 年 5 月 17 日

摘要

本实验实现了简易 MIPS 处理器中的几个重要部件:主控制单元 (Ctr)、ALU 控制单元 (ALUCtr) 以及算术逻辑单元 ALU。他们分别用于产生处理器所需要的控制信号以及根据控制信号的不同进行相应算术逻辑运算操作输出结果。实验通过软件仿真的方式进行结果验证。

目录

1 实验目的

1. 理解 CPU 控制器，ALU 的原理
2. 主控制器 (Ctr) 的实现
3. 运算单元控制器 (ALUCtr) 的实现
4. ALU 的实现
5. 使用功能仿真验证实现的正确性

2 原理分析

2.1 主控制器 (Ctr) 的设计

主控制器的基本功能是对当前执行指令最高六位进行解析（这里对应输入的 OpCode 域），来初步判断该指令的类型，将指令区分为 R 型（R 型内部具体指令不做区分），I 型：包括 load 指令 (lw)、store 指令 (sw)、branch 指令 (beq)，J 型指令：jump 指令 (j)。由于能力有限，这里我们所要考虑的指令仅包含实验指导书中给出的部分。根据解析的结果产生并输出对应的控制信号。本实验中用到的控制信号如表1所示。

信号名称	具体含义
ALUSrc	ALU 的第二个输入操作数来源（0：使用 rt 读出值；1：使用立即数）
RegWrite	寄存器写使能信号，高电平有效
RegDst	目标寄存器选择信号（0：写入 rt；1：写入 rd）
MemRead	内存读使能信号，高电平有效
MemWrite	内存写使能信号，高电平有效
MemToReg	选择是否将内存读取内容写入寄存器（0：不写回；1：写回）
Branch	条件跳转使能信号，高电平有效
Jump	无条件跳转使能信号，高电平有效
ALUOp	ALU 需要执行的操作类型

表 1: 主控制器发出的控制信号

这里 ALUOp 为一个两位的控制信号，在主控制器中的输出值仅包含 00、10、01 三种，取值与含义的对应关系如表2所示。

ALUOp 取值	含义
10	当前指令为 R 型，具体算术逻辑运算方式取决于指令第六位（即 Funct 域）
00	ALU 执行加法运算
01	ALU 执行减法运算

表 2: ALUOp 信号含义

由于非算数运算指令仍包含一些需要 ALU 进行运算的部分，所以这里 ALUOp 信号的目的就是为了区分这些指令需要做加法或是减法，同时将 R 型指令与非 R 型指令区分开来，所以，这里的 ALUOp 信号并不是最终决定 ALU 运算的信号，仍需配合运算单元控制器 ALUCtr 对指令 funct 域的解析处理才能决定具体的运算种类送入 ALU。

而主控制器产生的控制信号值可以完全由指令的 OpCode 域决定，对应关系如表3所示。

OpCode	$6'b000000$	$6'b100011$	$6'b101011$	$6'b000100$	$6'b000010$
指令类型	R 型指令	lw 指令	sw 指令	beq 指令	无条件转移指令
ALUSrc	0	1	1	0	0
RegWrite	1	1	0	0	0
RegDst	1	0	0	0	0
MemRead	0	1	0	0	0
MemWrite	0	0	1	0	0
MemToReg	0	1	0	0	0
Branch	0	0	0	1	0
Jump	0	0	0	0	1
ALUOp	10	00	00	01	00

表 3: 主控制器发出的控制信号

根据表3中内容即可实现主控制器的控制信号功能，在出现其他指令情况下，利用默认赋值将所有控制信号置零，保证数据不受影响，从而实现简单的异常处理功能。

2.2 运算单元控制器 (ALUCtr) 的设计

运算单元控制器的主要功能是解析指令字的 funct 域，配合主控制器发出的 ALUOp 信号对 ALU 将要实现的算术运算功能进行决定，这里输出信号为四位功能信号 ALUCtrOut，该信号取值与 ALU 功能对应关系如表4所示。

ALUCtrOut 取值	ALU 功能
0000	与运算
0001	或运算
0010	加法运算算
0110	减法运算
0111	小于时置位

表 4: ALUOp 信号含义

指令	ALUOp	funct 域	ALUCtrOut
add	10	100000	0010
sub	10	100010	0110
and	10	100100	0000
or	10	100101	0001
slt	10	101010	0111
lw	00	xxxxxx	0010
sw	00	xxxxxx	0010
beq	01	xxxxxx	0110

表 5: 指令对应 ALUCtrOut 输出

正如我们在上一部分所说，指令为 R 型时，仅凭借 ALUOp 无法分辨 ALU 所要进行的算术逻辑运算操作，所以应当根据 funct 域解析得到具体的 ALU 操作类型。而当指令为 I 型或 J 型时，ALUOp

为 00 和 01，可以直接输出加法/减法运算控制信号。不同 ALUOp 与 funct 域值对应 ALU 控制信号如表5所示。

2.3 算术逻辑运算单元 ALU 的设计

ALU 的主要功能是根据算术逻辑运算控制单元输出的 ALUCtrOut 信号，对两个输入操作数进行相应的算术逻辑运算，并输出运算结果与部分控制信号（如 zero 信号：运算结果为 0 时信号为高电平）。ALU 的算术逻辑运算类型如表4所示。

3 功能实现

3.1 主控制器 (Ctr) 的实现

主控制器利用 OpCode 域解析出控制信号输出，可以通过 Verilog 的 case 语句实现，部分代码如下，这里只展示 R 型指令以及 default 赋值对应部分，其余形式类似，具体见工程文件。

```
always@(opCode) begin
    case(opCode)
    6'b000000: //R type
     begin
         RegDst = 1;
         ALUSrc = 0;
         MemToReg = 0;
         RegWrite = 1;
         MemRead = 0;
         MemWrite = 0;
         Branch = 0;
         ALUOp = 2'b10;
        Jump = 0;
    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
```

这里的 default 赋值部分在前文中已经提到过，是为了防止无效指令输入对系统的影响，所有输出信号置零的代码部分。

3.2 运算单元控制器 (ALUCtr)

当用于判断 case 情况的编码中存在无关紧要值的情况下，通常我们可以直接利用 x 来代替，所以在运算单元控制器的代码实现部分可以使用 Verilog 中 casex 语句块进行实现，具体实现代码如下：

```
always @ (ALUOp or Funct) begin
    casex ({ALUOp,Funct})
        8'b00xxxxxx : ALUCtrOut = 4'b0010;
        8'b01xxxxxx : 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;
    endcase
end
```

3.3 算术逻辑单元 ALU 的实现

算术逻辑单元 ALU 根据 ALUCtrOut 的控制信号输入选择对两个输入操作数的算术逻辑运算类型的选择，输出结果以及 zero 信号量，在这里可以选用 if-else 语句完成代码，具体实现部分代码如下：

```
always @ (input1 or input2 or aluCtr) begin
    if (aluCtr == 4'b0010) //add
        aluRes = input1 + input2;
    else if(...)
    ...
    ...
    ...
    if(aluRes == 0)
        zero = 1;
    else
        zero = 0;
end
```

4 结果验证

对于本实验中实现的简易 MIPS 处理器中控制信号以及算术逻辑单元相关的三个模块采用软件仿真的方法进行测试，用 Verilog 语言编写测试激励文件，进行仿真，观察各信号波形变化，比对是否符

合预期。

4.1 主控制器 (Ctr) 的测试

每隔 100ns 送入一次不同 OpCode 数值，分别代表 R 型指令、lw 指令、sw 指令、beq 指令、jump 指令以及无效指令，用于分别测试几种情况下主控制器的仿真性能，仿真结果如图1所示。

4.2 运算控制单元 (ALUCtr) 的测试

每隔 100ns 对 ALUOp 以及 funct 数值进行一次赋值，分别代表 lw 指令、sw 指令、beq 指令、add 指令、sub 指令、and 指令、or 指令以及 slt 指令，用于分别测试几种情况下主控制器的仿真性能，仿真结果如图2所示。

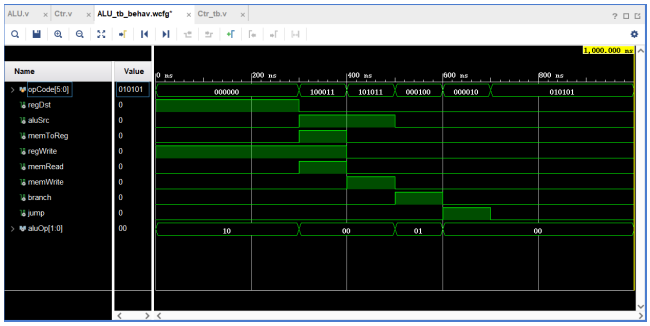

图 1: 主控制器仿真波形

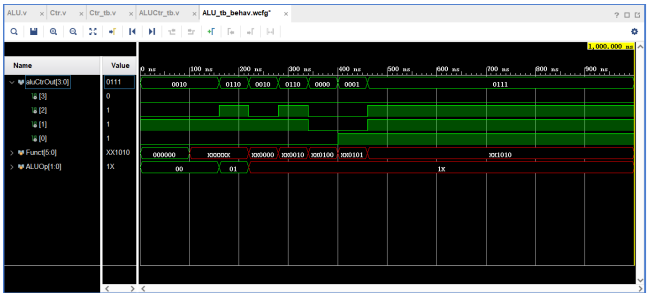

图 2: 运算控制单元仿真波形

4.3 算术逻辑运算单元 ALU 的测试

每隔 100ns 对 ALUCtrOut 的输入或源操作数进行一次赋值，输入的控制信号以及对应的输入数据如表6所示，仿真得到的波形如图3及图4所示。

运算类型	源操作数 1	源操作数 2
and	0	0
and	15	10
or	15	10
add	15	10
sub	15	10
sub	10	15
slt	15	10
slt	10	15
nor	1	1
nor	16	1

表 6: ALU 测试内容

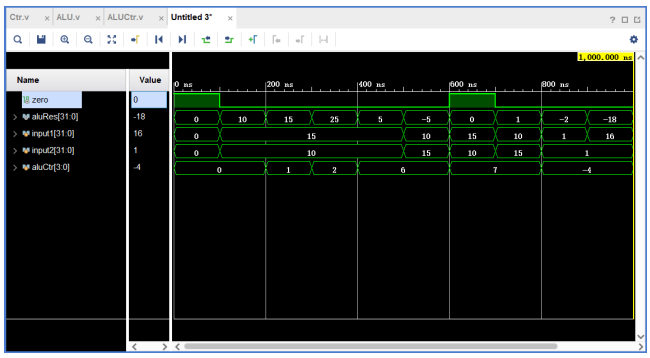

图 3: ALU 仿真波形

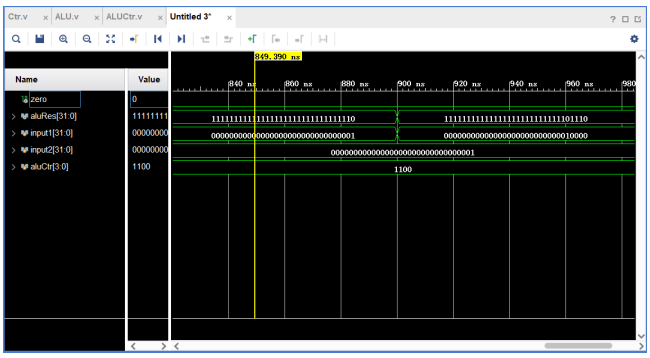

图 4: 其中 NOR 运算展示

可以看到 ALU 正确执行了各项运算，并且根据结果正确地设置了 zero 信号的输出值，证明本次实验验证成功。

5 总结与反思

本实验实现了简易 MIPS 处理器中主控制器、运算控制单元以及算数运算单元的简单仿真，为接下来的单周期 MIPS 处理器以及多周期流水线 MIPS 处理器的实现奠定了基础，但是这里实现的这三个部件都只是最基础的功能，ALU 中逻辑位移操作并没有进行实现，而且主控制器能够解析的指令非常少，一些基础的 jr 指令等在后面的实验中仍需进行完善，同时，这里的算术逻辑运算我们直接使用了 Verilog 代码中的算术逻辑运算，而并不是像实验 2 中的四位全加器一样仅通过最基础的与、或、异或运算来实现，有机会可以尝试完成。