计算机系统结构实验报告 Lab03

简单的类 MIPS 单周期处理器功能部件的设计与实现

徐阳　521021910363

2023 年 5 月 3 日

目录

1 简介

在本实验中，我学习并在 Verilog 中实现了 Mips 处理器中的几个核心部件的模块：主控制器单元 Ctr 模块、运算单元控制器单元 ALUCtr 模块、算数逻辑运算单元 ALU 模块，他们分别实现了根据指令解析处理器控制信号、产生运算单元控制信号和根据控制信号进行运算的功能。最后通过行为仿真验证了程序的正确性。

2 实验目的

1. 理解主控制器单元 Ctr、运算单元控制器单元 ALUCtr、算数逻辑运算单元 ALU 的原理
2. 熟悉所需的 Mips 指令集
3. 使用 Verilog 设计与实现主控制器单元 Ctr、运算单元控制器单元 ALUCtr、算数逻辑运算单元 ALU
4. 掌握激励文件的编写，使用行为仿真验证程序正确性

3 实验原理与功能实现

3.1 主控制器单元 Ctr

3.1.1 原理分析

主控制器单元 Ctr 的输入为指令的高六位 OpCode，输出为发送给 ALU 和 ALUCtr 的两位 ALUOp 控制信号和处理器各部分的控制信号：目标寄存器的选择信号 regDst，ALU 第二个操作数的来源 ALUSrc，写寄存器的数据来源 memToReg，内存读使能信号 memRead，内存写使能信号 memWrite，条件跳转信号 branch，无条件跳转信号 Jump。

主控制器单元 Ctr 根据指令高六位的 OpCode 初步判断指令是 R 型，I 型还是 J 型指令，并产生相应的处理器各部分控制信号。在本次实验中，我们实现了最基本的几条指令的控制信号解析，更复杂的指令集解析能力将在之后的任务中实现。其中，对 R 型指令，我们不对其具体类型加以分析，而是把任务交给运算单元控制器单元 ALUCtr，对于 I 型指令，其有 lw，sw，beq 三种指令，对于 R 型指令，当前只支持 Jump 指令。具体地信号对应如表 1所示，其中 x 表示 0 和 1 都可以。

3.1.2 代码实现

具体的代码实现上，我们使用 case 语句进行 OpCode 的类型判断，需要注意的是，在 Verilog 的 case 语句中末尾一定要写上 default 给没有列出的情况赋值，否则出现没

表 1: Ctr 产生的控制信号与指令的对应

OpCode	000000(R)	000010(j)	000100(beq)	100011(lw)	101011(sw)
ALUSrc	0	0	0	1	1
ALUOp	1x	00	01	00	00
Branch	0	0	1	0	0
Jump	0	1	0	0	0
memRead	0	0	0	1	0
memToReg	0	0	0	1	0
memWrite	0	0	0	0	1
regDst	1	0	0	0	0
regWrite	1	0	0	1	0

有列出的情况时，输出会默认为高阻态 Z。在如下的代码示例中，展示了 R 型指令的解析，全部指令类型完整的解析见源代码。

另一个需要注意的细节是，Verilog 中 output 默认为 wire 类型，无法在 always 模块中直接赋值，需要在初始化时指定为 output reg 类型，或者我们可以设置 reg 类型中间变量，在 always 模块中赋值给中间变量，再在最后使用 assign 语句赋值给 output。

——Ctr.v——

```
...
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
...
}
```

3.2 运算单元控制器单元 ALUCtr

3.2.1 原理分析

运算单元控制器单元 ALUCtr 输入为两位 ALUOp，指令低六位的 funct，输出为给 ALU 的运算控制信号 ALUCtrOut，该控制信号作用于 ALU，指定其运算类型，具体的解析及型号对应如表 2所示。

表 2: ALUCtr 中的信号解析

指令	ALUOp	funct	ALUCtrOut	运算类型
add	1x	100000	0010	add
sub	1x	100010	0110	sub
and	1x	100100	0000	and
or	1x	100101	0001	or
slt	1x	101010	0111	slt
lw	00	xxxxxx	0010	add
sw	00	xxxxxx	0010	add
beq	01	xxxxxx	0110	sub
j	00	xxxxxx	0010	add

3.2.2 代码实现

我们采用支持通配符 x 的 casex 语句实现 ALUCtr，以下是代码的关键部分，同样主要到这里也要写 default 语句。

——ALUCtr.v——

```
...
    always @ (aluOp or funct)
    begin
        casex({aluOp,funct})
            8'b00xxxxxx:
                ALUCtrOut = 4'b0010;
            8'b01xxxxxx:
                ALUCtrOut = 4'b0110;
            8'b10xx0000:
                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 算数逻辑运算单元 ALU

3.3.1 原理分析

算数逻辑运算单元 ALU 的输入为两个 32 位操作数 input1 和 input2，以及运算控制信号 ALUCtr，输出为 zero 标志位 32 位结果 output。其根据 ALUCtr 的类型对两个操作数进行相应的算数逻辑运算，并将结果输出到 output，若运算结果为 0，则将 zero 标志位置一，否则其为零，该标志位将在之后的实验中用于条件转移指令的判断。

在 ALU 中，ALUCtr 信号与 ALU 执行的运算类型的对应关系如表 3所示。

表 3: ALUCtr 中的信号解析

ALUCtr	运算类型
0000	and
0001	or
0010	add
0110	sub
0111	slt
1100	nor

3.3.2 代码实现

仿照前两个模块，同样采用 case 语句实现 ALU 单元，关键的代码部分如下，完整代码见源文件。

在实现小于时置位 (slt) 时，因为 Verilog 中的 input 类型默认为无符号类型，需要对其使用 $signed() 进行强制转化为有符号类型。另外创建并传递数据给 int 类型专门用于比较也是一种可行的办法，在这里我们选择了前者。

——ALU.v——

```
...
    always @ (input1 or input2 or aluCtr)
    begin
        case(aluCtr)
        4'b0000:
            ALURes = input1 & input2;
        4'b0001:
            ALURes = input1 | input2;
        4'b0010:
            ALURes = input1 + input2;
        4'b0110:
            ALURes = input1 - input2;
        4'b0111:
            ALURes = ($signed(input1) < $signed(input2));
        4'b1100:
            ALURes = ~(input1 | input2);
        endcase
        if(ALURes == 0)
            Zero = 1;
        else
            Zero = 0;
```

```
    end
...
}
```

4 仿真验证

4.1 主控制器单元 Ctr 仿真验证

编写激励文件代码如下，进行仿真验证。在仿真中检验了多条指令的解析，仿真结果如图 1所示，可见仿真结果正确，我们实现了主控制器单元 Ctr。

——Ctr_tb.v——

```
...
    initial begin
        //Initialize inputs
        OpCode = 0;
        //Wait 100 ns for global reset to finish
        #100;
        #100 OpCode = 6'b000000; //R type
        #100 OpCode = 6'b100011; //lw
        #100 OpCode = 6'b101011; //sw
        #100 OpCode = 6'b000100; //beq
        #100 OpCode = 6'b000010; //jump
        #100 OpCode = 6'b010101;
    end
...
}
```

4.2 运算单元控制器单元 ALUCtr 仿真验证

编写激励文件进行仿真验证。在仿真中检验了各个类型的信号的解析，仿真结果如图 2和图 3所示。其中，图 2的情况是输入信号带通配符 x 的，而图 3是实际情况中 x 为具体的 0 或 1 的情况。可见 casex 正确的发挥了作用，仿真结果正确，我们实现了运算单元控制器单元 ALUCtr。

4.3 算数逻辑运算单元 ALU 仿真验证 2

编写激励文件，进行仿真验证。在仿真中分别验证了表 3中的各个运算类型，仿真结果如图 4所示，在这里我们特别截取了 and，or 与 xor 的运算的二进制型号，我们都

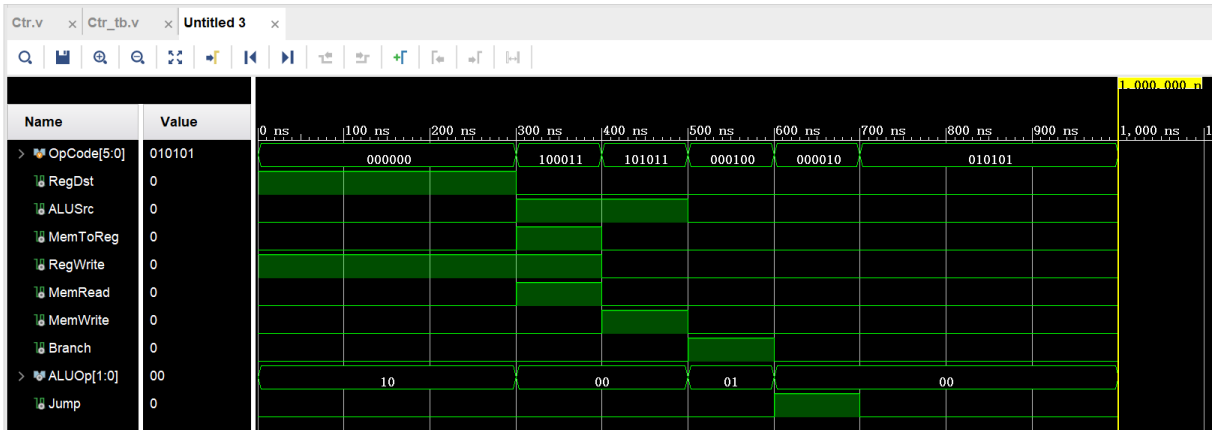

图 1: 主控制器单元 Ctr 仿真验证

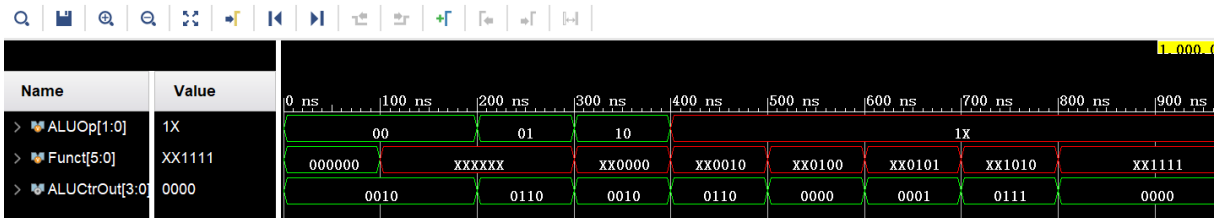

图 2: 运算单元控制器单元 ALUCtr 仿真验证 1

得到了正确的 output 结果和 zero 标志位，可见仿真结果正确，我们实现了算数逻辑运算单元 ALU。

5 总结与思考

本实验的目标是理解并实现主控制器单元 Ctr、运算单元控制器单元 ALUCtr、算数逻辑运算单元 ALU。在计算机系统结构的课堂上，我们初步认识了 mips 处理器的各个部件的作用，但是我对其中各部分的作用仍有一些疑惑。虽然本次实验只实现了最基本的几种指令，但“千里之行，始于足下”，通过这次简单易懂实验，我具体实现了各个控制单元与运算部件的信号处理，让我对处理器这部分的知识理解的更加深刻了。

通过这次实验，我也对 Verilog 和 Vivado 的模块化思想和其中的 case、casex 语句有了更深刻的了解，让我的 Verilog 编程能力有了不小的提高。

6 致谢

感谢老师课堂上对于相关内容的教学。感谢助教课堂上对于我的疑问的解答。感谢老师与助教提供的实验手册对于实验的指导。

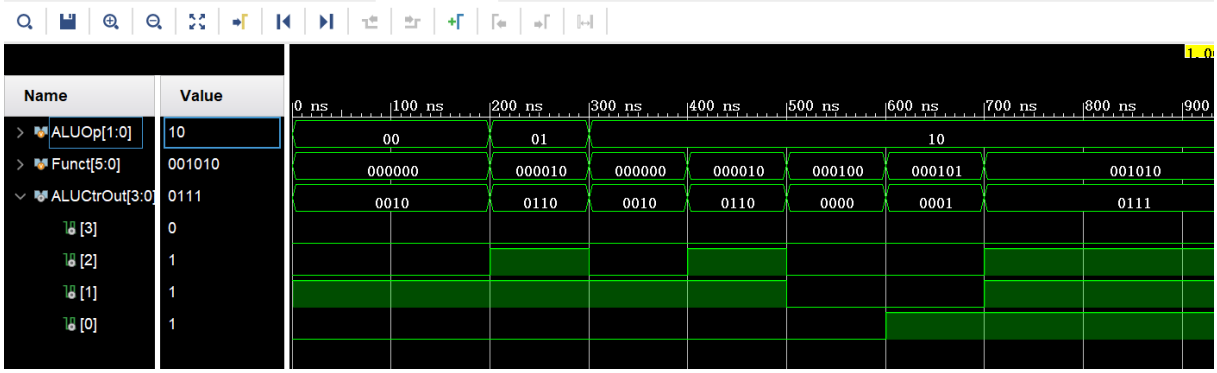

图 3: 运算单元控制器单元 ALUCtr 仿真验证

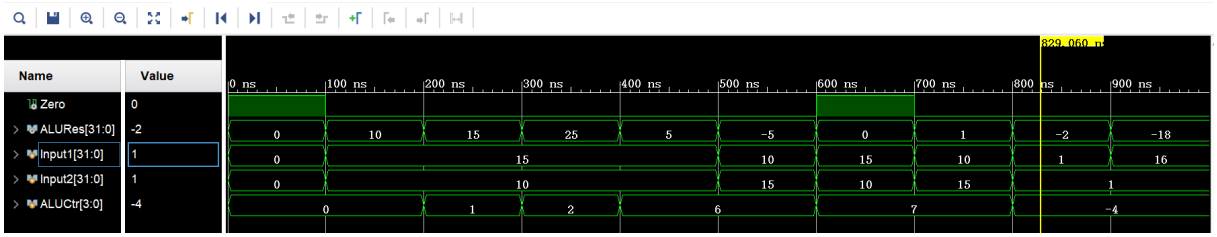

图 4: 运算单元控制器单元 ALUCtr 仿真验证

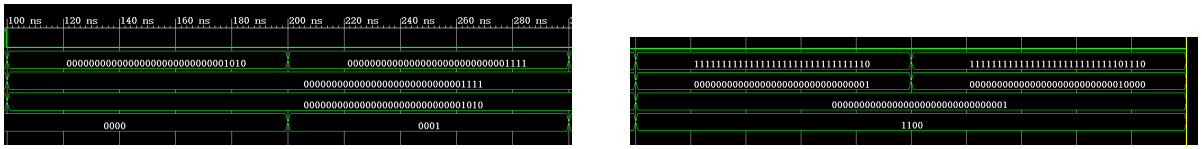

图 5: and，or 与 xor 的运算结果