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（一）：控制器与 ALU

姚宣骋 520021910431

2022 年 3 月 22 日

摘要

本实验实现了简单的类 MIPS 处理器的几个功能部件：主控制器（Ctr）、运算单元控制器（ALUCtr）、算数逻辑运算单元（ALU）。Ctr 的功能是根据指令最高 6 位（OpCode）产生各种控制信号；ALUCtr 的功能是根据 Ctr 的控制信号（ALUOp）与指令最低 6 位（Funct）产生控制 ALU 的信号（ALUCtrOut）；ALU 的功能是根据 ALUCtr 的信号和两个输入执行相应的算数逻辑运算并输出结果。本实验用软件仿真的形式进行实验结果的验证。

目录

1 实验目的

本次实验有 4 个目的:

- 理解主控制部件或单元、ALU 控制器单元、ALU 单元的原理;
- 熟悉所需的 Mips 指令集;

- 使用 Verilog HD 设计和实现主控制器(Ctr)、运算单元控制器(ALUCtr)和算数逻辑运算单元（ALU)；
- 使用功能仿真验证功能实现的正确性。

2 原理分析

2.1 主控制器（Ctr）原理分析

主控制器会对指令的最高 6 位的 OpCode 进行解析，初步判定指令的类型并产生相应的处理器信号。在 Ctr 中，OpCode 可以大致分为 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)。控制信号则如表 1 所示。

表 1: Ctr 控制信号

信号	具体说明
ALUSrc	ALU 的第二个操作数来源（0：使用 rt；1：使用立即数）
ALUOp（*）	发送给 ALUCtr 用来进一步解析运算类型的控制信号
Branch	条件跳转信号，高电平说明当前指令是条件跳转指令 (branch)
Jump	无条件跳转信号，高电平说明当前指令是无条件跳转指令 (jump)
memRead	内存读使能信号，高电平说明当前指令需要进行内存读取 (load)
memToReg	写寄存器的数据来源 (0: 使用 ALU 运算结果；1: 使用内存读取数据)
memWrite	内存写使能信号，高电平说明当前指令需要进行内存写入 (store)
regDst	目标寄存器的选择信号 (0: 写入 rt 代表的寄存器；1: 写入 rd 代表的寄存器)
regWrite	寄存器写使能信号，高电平说明当前指令需要进行寄存器写入

其中 ALUOp 信号为两个二进制为的信号，其含义比一般信号复杂，如表 2 所示。此信号还需要经过 ALUCtr 处理后送入 ALU，才能对 ALU 进

表 2: ALUOp 信号具体含义与解析方式

ALUOp 的信号内容	指令	具体说明
00	lw, sw, j	ALU 执行加法运算
01	beq	ALU 执行减法运算
1x	R 型指令	ALU 具体执行内容需要根据指令最后 6 位的 Funct 域决定

行控制。

OpCode 指令与控制信号的对应关系如表 3 所示一一对应。若出现其

表 3: OpCode 指令与控制信号的对应关系

OpCode 指令	000000 R 型指令	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

余未设定指令，我们先暂且将所有控制信号置 0，即为空指令（nop）。

2.2 运算单元控制器（ALUCtr）原理分析

运算单元控制器（ALUCtr）对 ALUOp 信号和指令后 6 位 Funct 进行解析，给出控制 ALU 的控制信号 ALUCtrOut。输入与输出的对应关系如表 4 所示。

表 4: 运算单元控制器（ALUCtr）的解析方式

指令	ALUOp	Funct	ALUCtrOut	具体说明
add	1x	100000	0010	ALU 执行加法运算
sub	1x	100010	0110	ALU 执行减法运算
and	1x	100100	0000	ALU 执行逻辑与运算
or	1x	100101	0001	ALU 执行逻辑或运算
slt	1x	101010	0111	ALU 执行小于时置位运算
lw	00	xxxxxx	0010	ALU 执行加法运算
sw	00	xxxxxx	0010	ALU 执行加法运算
beq	01	xxxxxx	0110	ALU 执行减法运算
j	00	xxxxxx	0010	ALU 执行加法运算

2.3 算术逻辑运算单元（ALU）原理分析

ALU 根据 ALUCtr 产生的控制信号 ALUCtrOut 对两个输入数进行对应的算数逻辑运算，并且输出运算结果以及部分控制信号（zero）。该信号用于与 branch 指令结合判断是否满足转移条件。ALU 执行的算术逻辑运算类型与运算单元控制信号 ALUCtrOut 的对应方式如表 5 所示。

表 5: ALU 的算术逻辑运算类型与 ALUCtrOut 的对应方式

ALUCtrOut	ALU 执行算数逻辑运算类型
0000	逻辑与 (and)
0001	逻辑或 (or)
0010	加法 (add)
0110	减法 (sub)
0111	小于时置位 (slt)
1100 (*)	逻辑或非 (nor)

3 功能实现

3.1 主控制器（Ctr）功能实现

根据 2.1 节中的控制信号对应表，使用 Verilog 中的 case 语句分别对各种情况进行实现，部分代码如下，完整代码详见附录 A.1。

```
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
            6'b100011://␣lw
␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣RegDst=0;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUSrc=1;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣MemToReg=1;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣.....

```

```
24 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUOp=2'b00;
25                         Jump=0;
26                 end
27             endcase
28         end
```

3.2 运算单元控制器（ALUCtr）功能实现

根据 2.2 节中的控制信号对应表，使用 Verilog 中的 casex 语句分别对各种情况进行实现，部分代码如下，完整代码详见附录 A.2。

```
1  always @(aluOp or funct)
2      begin
3          casex ({aluOp, funct})
4              8'b00xxxxx:
5  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUCtrOut=4'b0010;
6              8'b01xxxxx:
7  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUCtrOut=4'b0110;
8              8'b1xxx0000:
9  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUCtrOut=4'b0010;
10             8'b1xxx0010:
11 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUCtrOut=4'b0110;
12             8'b1xxx0100:
13 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUCtrOut=4'b0000;
14             8'b1xxx0101:
15 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUCtrOut=4'b0001;
16             8'b1xxx1010:
17 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ALUCtrOut=4'b0111;
18         endcase
```

```
19      end
```

3.3 算术逻辑运算单元（ALU）功能实现

根据 2.3 节中的控制信号对应表，使用 Verilog 中的 case 语句分别对各种情况进行实现，并用 if 语句对 zero 信号单独设置。部分代码如下，完整代码详见附录 A.3。

```
1   always @ (input1 or input2 or aluCtr)
2       begin
3           case(aluCtr)
4           4'b0000:
5   ␣␣␣␣␣␣␣␣ALURes=input1␣&␣input2;
6   ␣␣␣␣␣␣␣␣4'b0001:
7           ALURes=input1 | input2;
8           4'b0010:
9   ␣␣␣␣␣␣␣␣ALURes=input1+input2;
10  ␣␣␣␣␣␣␣␣4'b0110:
11          ALURes=input1-input2;
12          4'b0111:
13  ␣␣␣␣␣␣␣␣ALURes=($signed(input1)<$signed(input2));
14  ␣␣␣␣␣␣␣␣4'b1100:
15          ALURes=~(input1 | input2);
16          default:
17          ALURes=0;
18          endcase
19          if(ALURes==0)
20              Zero=1;
21          else
```

```
22                  Zero=0;
23          end
```

4 结果验证

4.1 主控制器（Ctr）结果验证

我们使用 Verilog 编写激励文件，采用软件仿真的形式对于主控制器 Ctr 进行测试（代码实现参见附录 B.1）。分别对 R 型指令、load 指令 (lw)、store 指令 (sw)、branch 指令 (beq)、jump 指令 (j) 以及其他不支持的指令进行测试，测试结果如图 1 所示。从图 1 中可以看出，主控制器正确产生

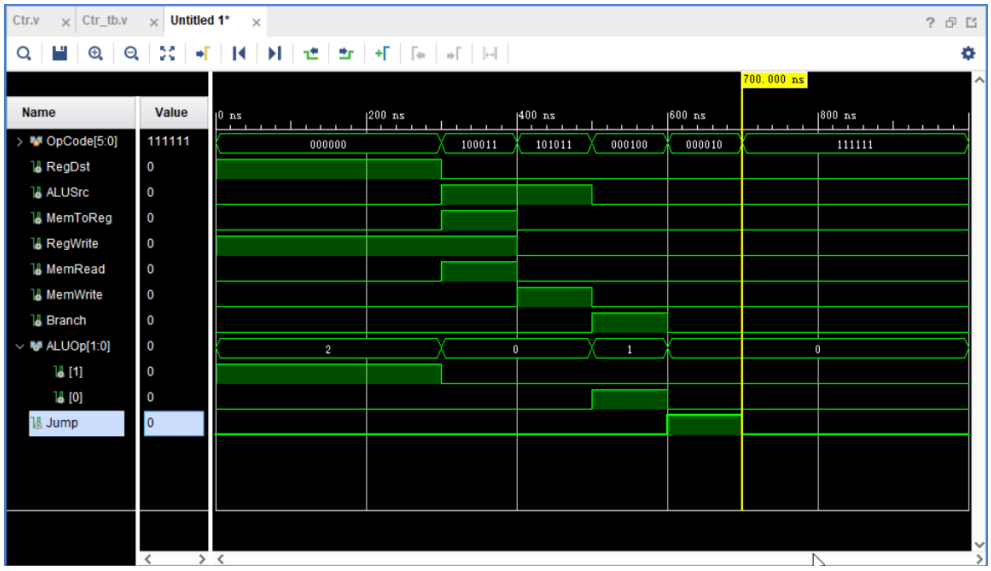

图 1: 对于主控制器（Ctr）的测试结果

了第 2.1 节中表 1 中的所有控制信号；同时对于暂时不支持的其他无效指令，进行了置 0 操作。

4.2 运算单元控制器（ALUCtr）结果验证

我们使用 Verilog 编写激励文件，采用软件仿真的形式对运算单元控制器 (ALUCtr) 模块进行测试（代码实现参见附录 B.2）。分别对加法指令 (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 ALUOp 为 00 或 01 的指令（如 load 指令 (lw)、branch 指令 (beq) 等）进行测试，测试结果如图 2 所示。从图 2 中看出，ALUCtr 正

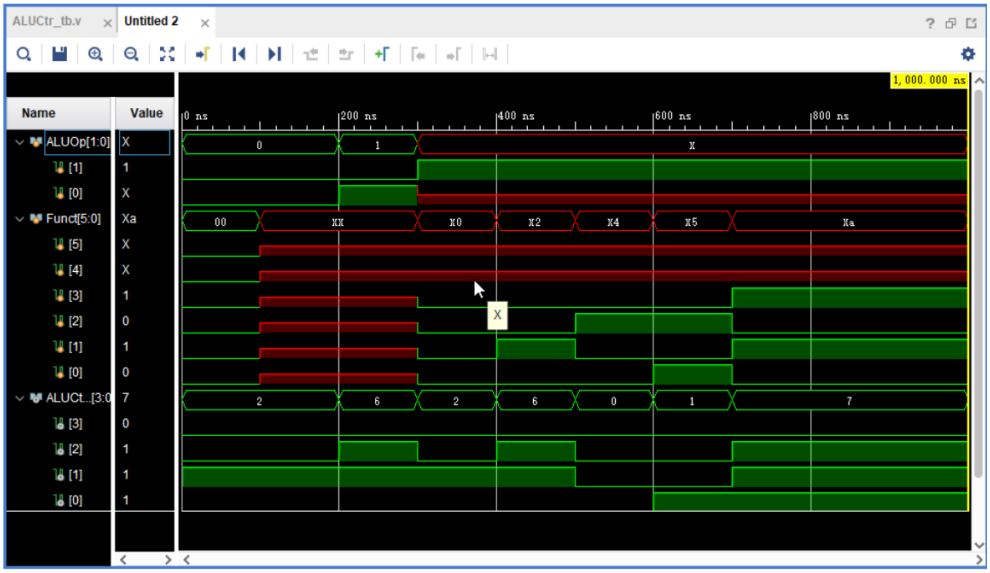

图 2: 对于运算单元控制器（ALUCtr）的测试结果

确产生了控制信号 ALUCtrOut。

4.3 算术逻辑运算单元（ALU）结果验证

我们使用 Verilog 编写激励文件，采用软件仿真的形式对算术逻辑运算单元（ALU）模块进行测试（代码实现参见附录 B.3）。分别对加法、减法、与运算、或运算、小于时置位、或非运算进行测试，结果如图 3 所示。从图 3 中看出，ALU 根据指令执行了正确的运算与输出。

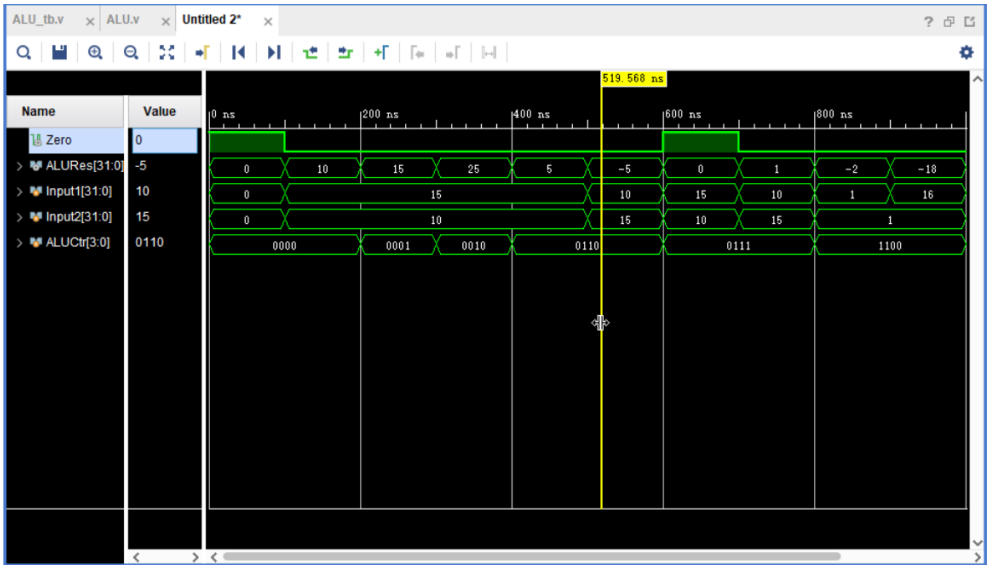

图 3: 对于算术逻辑运算单元（ALU）的测试结果

5 总结反思

本次实验是我们第一次自己设计代码实现功能。并且是类 MIPS 处理器最重要的三个基础组件：主控制器 (Ctr)、运算单元控制器 (ALUCtr) 以及算术逻辑运算单元 (ALU)，为以后更复杂的功能器件设计提供基础。同时通过软件仿真模拟的方式验证其正确性。

同时，我还学习并熟悉了 case 以及 casex 等 Verilog 语言的使用方式与技巧，使得我对于该门语言使用更加熟练。

6 致谢

感谢本次实验中指导老师在课程微信群里为同学们答疑解惑；
感谢上海交通大学网络信息中心提供的远程桌面资源；
感谢计算机科学与工程系相关老师对于课程指导书的编写以及对于课程的设计，让我们可以更快更好地学习相关知识，掌握相关技能；

感谢电子信息与电气工程学院提供的优秀的课程资源。

A 设计文件完整代码实现

A.1 主控制器（Ctr）的代码实现

参见代码文件 Ctr.v。

A.2 运算单元控制器（ALUCtr）的代码实现

参见代码文件 ALUCtr.v。

A.3 算术逻辑运算单元（ALU）的代码实现

参见代码文件 ALU.v。

B 激励文件完整代码实现

B.1 主控制器（Ctr）的激励代码实现

参见代码文件 Ctr_tb.v。

B.2 运算单元控制器（ALUCtr）的激励代码实现

参见代码文件 ALUCtr_tb.v。

B.3 算术逻辑运算单元（ALU）的激励代码实现

参见代码文件 ALU_tb.v。。