实验三实验报告

一、实验名称

简单的类 MIPS 单周期处理器实现-控制器，ALU

二、实验目的

理解 CPU 控制器，ALU 的原理

三、实验范围

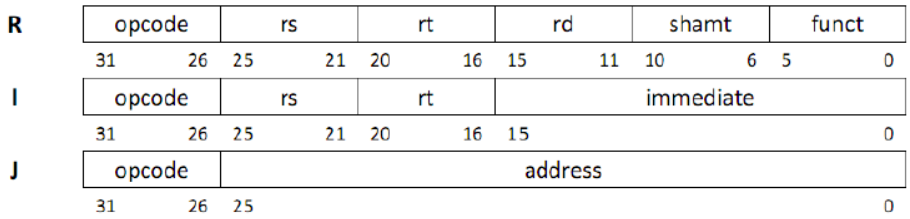
本次实验将覆盖以下范围

1. ISE 的使用
2. Spartan-3E 实验板的使用
3. 使用 Verilog HDL 进行逻辑设计
4. CPU 控制器的实现
5. ALU 的实现

四、主要的设计思想和测试仿真

1.主控制单元 Ctr

主控制单元输入为指令的 opCode ，即操作码。操作码经过主控制单元的译码，给 ALUCtr，Data Memory，Registers，Muxs 等部件输出正确的控制信号。



Mips 基本指令格式

由各个指令 opCode 和控制编码输出的关系表

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

主控制模块真值表（OpCode 与控制输出的编码关系）

注：Jump 指令编码是 000010，Jump 信号输出 1，其余输出 0

使用 case 语句，将每一种 opCode 对应的译码输出写出即可，这里以 R-type 指令为例。

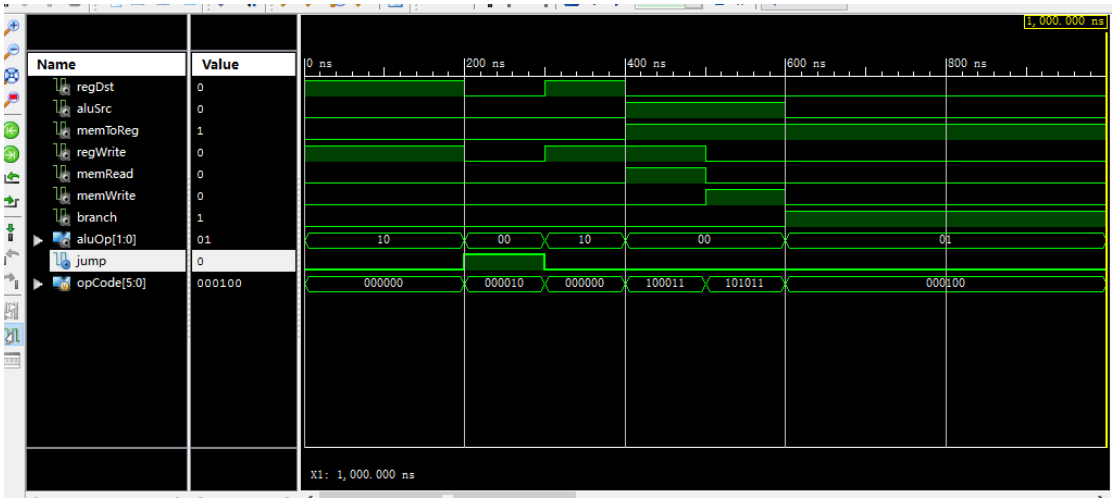
```
6'b000000: //R
begin
    regDst=1;
    aluSrc=0;
    memToReg=0;
    regWrite=1;
```

```
        memRead=0;
        memWrite=0;
        branch=0;
        aluOp=2'b10;
        jump=0;
    end
```

主控制单元 Ctr 的仿真即将所有指令的 opCode 输入，检查输出即可。

```
    #100;
    #100 opCode = 6'b000010;
    #100 opCode=6'b000000;
    #100 opCode=6'b100011;
    #100 opCode=6'b101011;
    #100 opCode=6'b000100;
```

仿真波形如下：



2. ALU 单元模块 ALUCtr

ALUCtr 根据主控制器提供的 ALUOp 来判断指令类型。根据指令的后 6 位区分 R 型指令。综合这两种输入，控制 ALU 做正确的操作。

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

aluCtrOut 和 alu 操作的对应关系

由 ALUOp 和 function field 得出的 ALU operation 有下标的对应的关系

ALUOp		Funct field						
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

输入输出真值表

所以，同样适用 case 语句，将各个不同的输入的对应的输出写出来即可，相关代码如下：

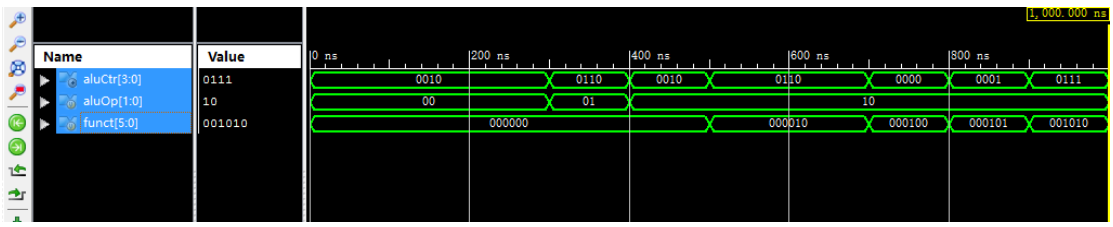
```
casex({aluOp, funct})
        8'b00xxxxxx: aluCtr = 4'b0010;
        8'bx1xxxxxx: aluCtr = 4'b0110;
        8'b1xxx0000: aluCtr = 4'b0010;
        8'b1xxx0010: aluCtr = 4'b0110;
```

```
        8'b1xxx0100: aluCtr = 4'b0000;
        8'b1xxx0101: aluCtr = 4'b0001;
        8'b1xxx1010: aluCtr = 4'b0111;
    endcase
```

ALUCtr 的仿真测试代码：

```
        #100;
        #100 aluOp = 2'b00;
        #100 aluOp = 2'b01;
        #100 aluOp = 2'b10; funct = 6'b000000;
        #100 aluOp = 2'b10; funct = 6'b000010;
        #100 aluOp = 2'b10; funct = 6'b000010;
        #100 aluOp = 2'b10; funct = 6'b000100;
        #100 aluOp = 2'b10; funct = 6'b000101;
        #100 aluOp = 2'b10; funct = 6'b001010;
```

仿真波形如下：



3. ALU

根据 ALUCtr，对两个输入做对应的操作。ALURes 输出结果。如果是

减法操作，若结果为 0，zero 输出置为 1.

输入：input1 (32bit)，input2 (32bit)，aluCtr (4bit)

输出：zero(1bit)，aluRes (32bit)

部分关键代码如下：

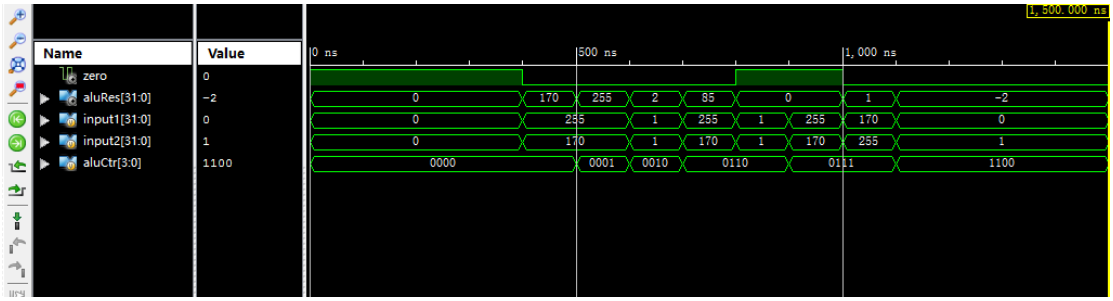
```
always @ (input1 or input2 or aluCtr)
   begin
      if(aluCtr == 4'b0010) //+
         aluRes = input1 + input2;
      else if(aluCtr == 4'b0110) //-
      begin
         aluRes = input1 - input2;
         if(aluRes == 0)
            zero = 1;
         else
            zero = 0;
      end
      else if(aluCtr == 4'b0000) //&
      begin
         aluRes = input1 & input2;
         if(aluRes == 0)
            zero = 1;
```

```
    else
        zero = 0;
end
```

ALU 的仿真测试代码：

```
#100;
#100 input1 = 0; input2 = 0; aluCtr = 4'b0000;
#100 input1 = 0; input2 = 0; aluCtr = 4'b0000;
#100 input1 = 255; input2 = 170; aluCtr = 4'b0000;
#100 input1 = 255; input2 = 170; aluCtr = 4'b0001;
#100 input1 = 1; input2 = 1; aluCtr = 4'b0010;
#100 input1 = 255; input2 = 170; aluCtr = 4'b0110;
#100 input1 = 1; input2 = 1; aluCtr = 4'b0110;
#100 input1 = 255; input2 = 170; aluCtr = 4'b0111;
#100 input1 = 170; input2 = 255; aluCtr = 4'b0111;
#100 input1 = 0; input2 = 1; aluCtr = 4'b1100;
```

仿真波形如下：



五、心得体会

这次实验算是真正接触了 Verilog 这个语言，收获不小。对于 always， case, begin end 等相关语法知识有了一定的了解。实验本身逻辑并不复杂，主要就是一些判断语句，将所有情况的输入输出的关系搞清楚即可。同时还使用仿真波形来验证，更加直观。在后来的 lab 中会发现，仿真波形能让我们更容易发现程序中的错误。