

Department of
**Computer Science
& Engineering**



计算机系统结构实验指导书-LAB03

1. OVERVIEW

1.1 实验名称

简单的类 MIPS 单周期处理器部件实现 – 控制器，ALU

1.2 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

1.3 实验报告与验收办法

需提交电子版实验报告，本实验检查三个仿真结果并验收登记

1.4 注意事项

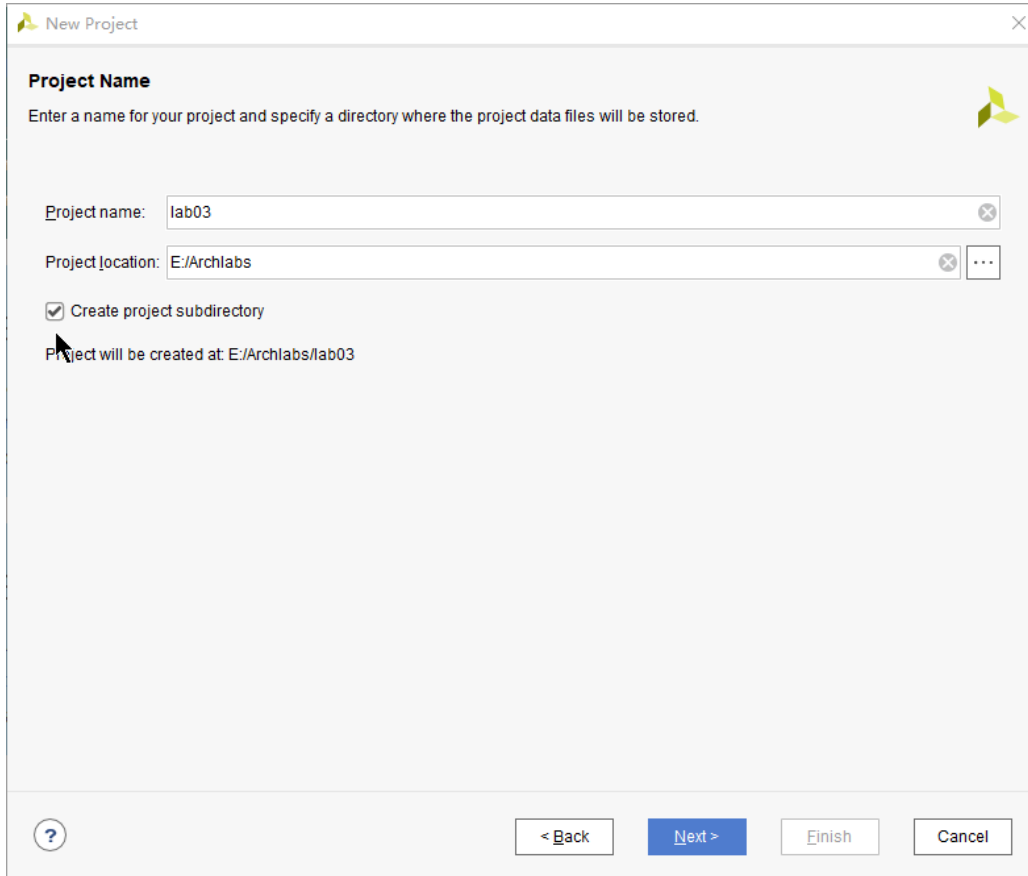
可建立 E:\Archlabs 文件夹，用于放置实验工程等

2. 新建工程

2.1 实验描述

2.1.1 新建工程

1. 启动 Vivado 2018.3
2. 输入工程名称 lab03 。点击 Next



The image shows the 'New Project' dialog box in Vivado. The title bar says 'New Project'. The main area is titled 'Project Name' and contains the instruction 'Enter a name for your project and specify a directory where the project data files will be stored.' There are two input fields: 'Project name:' with the value 'lab03' and 'Project location:' with the value 'E:/Archlabs'. Below these fields is a checkbox labeled 'Create project subdirectory' which is checked. Below the checkbox, it says 'Project will be created at: E:/Archlabs/lab03'. At the bottom of the dialog, there are four buttons: a help button (question mark in a circle), '< Back', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'.

3. 选择 开发板的 FPGA 参数:

Product Category: **ALL**

Family: **Kintex-7**

Package: **ffg676**

Speed: **-2**

Temperature: **ALL Remaining**

3. 主控制单元模块

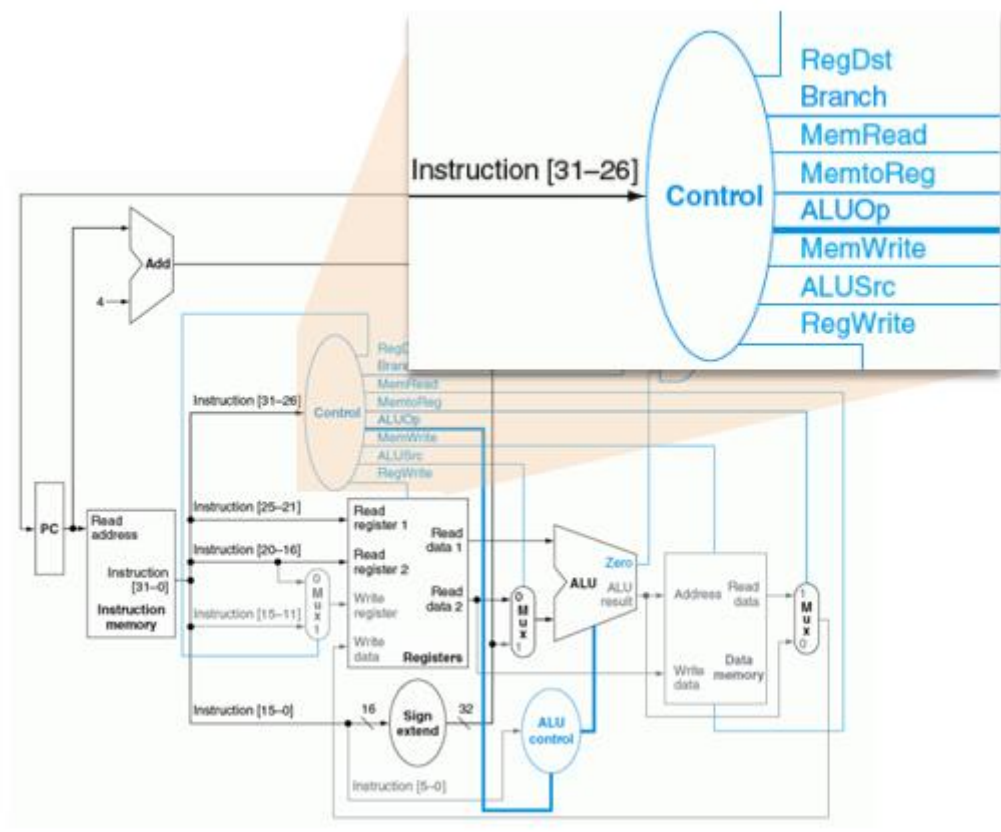
3.1 实验描述

3.1.1 模块描述

主控制单元（Ctr）的输入为指令的 opCode 字段，操作码经过 Ctr 的译码，给 ALUCtr, Data Memory, Registers, Muxs 等部件输出正确的控制信号。

R	opcode		rs		rt		rd		shamt		funct	
	31	26	25	21	20	16	15	11	10	6	5	0
I	opcode		rs		rt		immediate					
	31	26	25	21	20	16	15	0				
J	opcode		address									
	31	26	25	0								

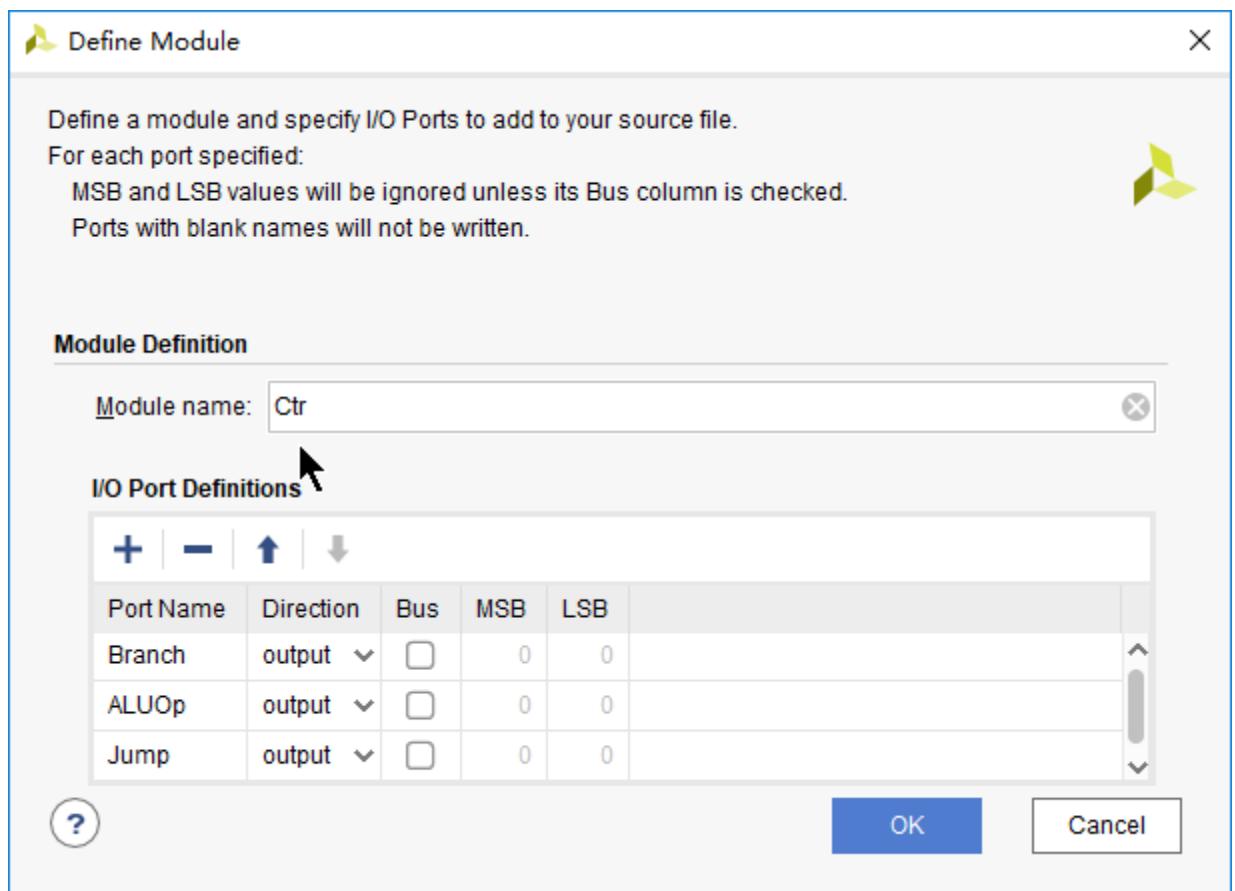
Mips 基本指令格式



主控制模块的 IO 定义

3.1.2 新建模块 Verilog 文件

1. 定义其 I/O 端口



```
20 //////////////////////////////////////////////////
21 module Ctr(
22     input [5:0] opCode,
23     output regDst,
24     output aluSrc,
25     output memToReg,
26     output regWrite,
27     output memRead,
28     output memWrite,
29     output branch,
30     output [1:0] aluOp,
31     output jump
32 );
33
34
35 endmodule
```

3.1.3 编写译码功能

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

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

指令	opCode
R 型: add, sub, and, or, slt	000000
I 型: lw	100011
I 型: sw	101011
I 型: beq	000100
J 型: J	000010


指令操作码

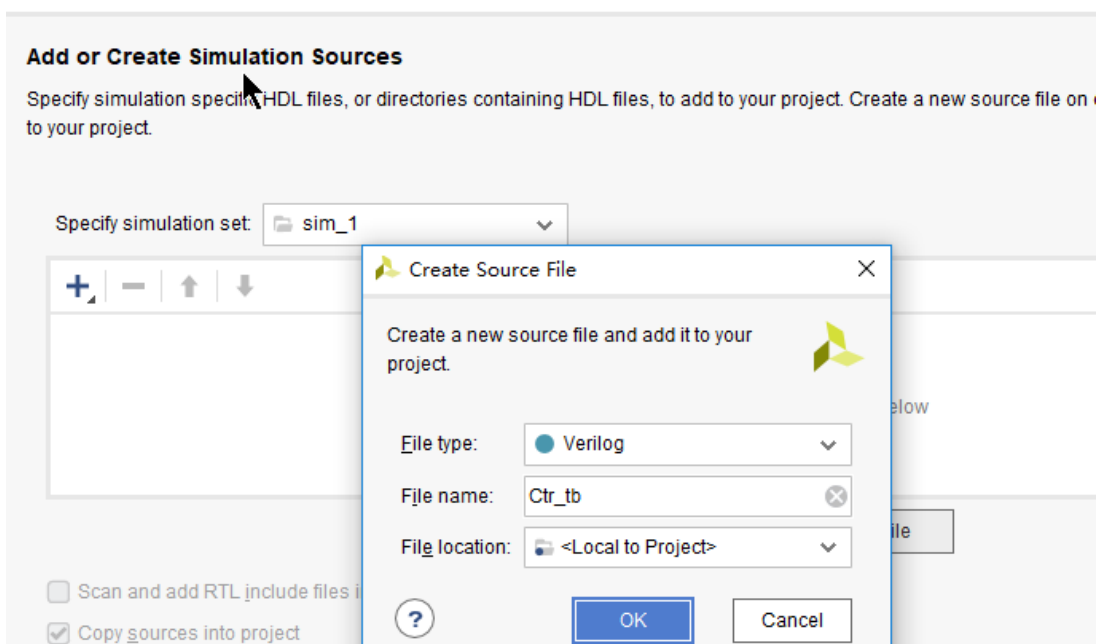
用 verilog 代码写出上述真值表，实现方式多种多样，这里给出一种使用 case 语句的参考样例，如下图：

```
14     reg RegDst;
15     reg ALUSrc;
16     reg MemToReg;
17     reg RegWrite;
18     reg MemRead;
19     reg MemWrite;
20     reg Branch;
21     reg [1:0] ALUOp;
22     reg Jump;
23
24     always @(OpCode)
25     begin
26         case (OpCode)
27             6'b000000: //R type
28             begin
29                 RegDst = 1;
30                 ALUSrc = 0;
31                 MemToReg = 0;
32                 RegWrite = 1;
33                 MemRead = 0;
34                 MemWrite = 0;
35                 Branch = 0;
36                 ALUOp = 2'b10;
37                 Jump = 0;
38             end
91
92         //add beq
93         //6'bxxxxxx;
94         begin
95             ...
96         end
97
98         //add lw
99         //add sw
100        //add Jump
101
102        default:
103        begin
104            RegDst = 0;
105            ALUSrc = 0;
106            MemToReg = 0;
107            RegWrite = 0;
108            MemRead = 0;
109            MemWrite = 0;
110            Branch = 0;
111            ALUOp = 2'b00;
112            Jump = 0;
113        end
114    endcase
```

3.1.4 功能仿真

1. 新建激励文件 Ctr_tb

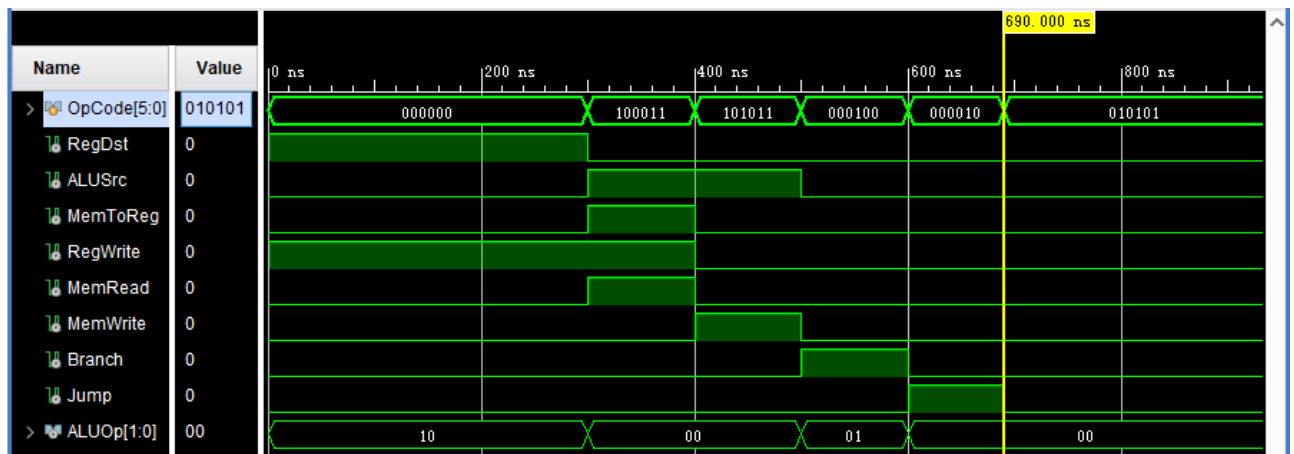
 Add Sources



2. 添加如下激励；设定不同的输入，覆盖所有的情况，以保证逻辑的正确。观察波形是否满足逻辑。若由误，修改代码，重新仿真

```
51
52 initial begin
53     // Initialize Inputs
54     OpCode = 0;
55
56     // Wait 100 ns for global reset to finish
57     #100;
58
59     #100 OpCode = 6'b000000; //R-type
60     //Add orther stimulus here
61
```

3. 下面给出一个仿真波形样例：



Ctrl 的仿真波形

4. 观察波形，查看仿真结果，是否满足当初的设计。如果有错，检查代码，重新仿真

3.2 实验报告

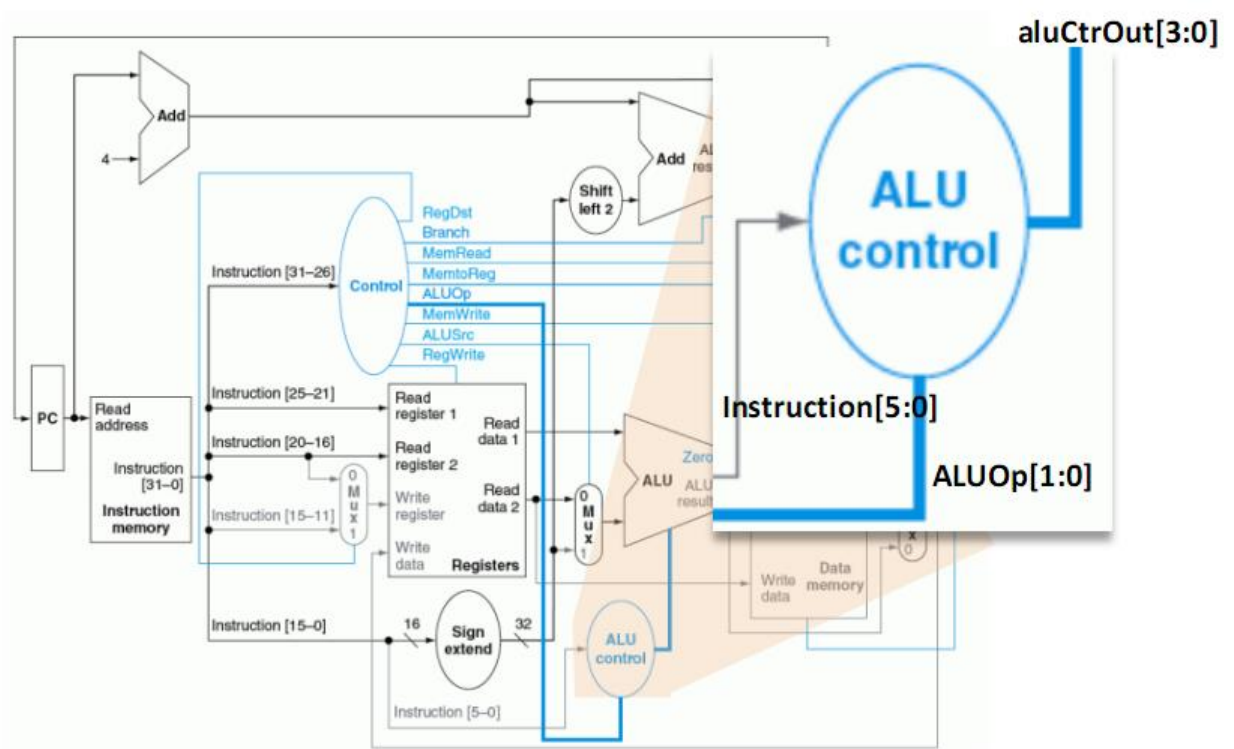
4. ALU 控制单元模块

4.1 实验描述

4.1.1 模块描述

ALU 的控制单元模块（ALUCtr）是根据主控制器的 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。

R	opcode		rs		rt		rd		shamt		funct	
	31	26	25	21	20	16	15	11	10	6	5	0
I	opcode		rs		rt		immediate					
	31	26	25	21	20	16	15	0				
J	opcode		address									
	31	26	25	0								



4.1.2 新建模块源文件

略

4.1.3 编写译码功能

ALU Control （即输出 aluCtrOut[3:0]）的值与 ALU 操作的对应关系如下：

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

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	and	0000
R-type	10	OR	100101	or	0001
R-type	10	set on less than	101010	set on less than	0111

Funct, ALUOp 与 ALU Control 编码关系

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
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

用 verilog 代码写出上述真值表内容。
这里给出一种使用 casex 语句的参考样例，如下图：

```

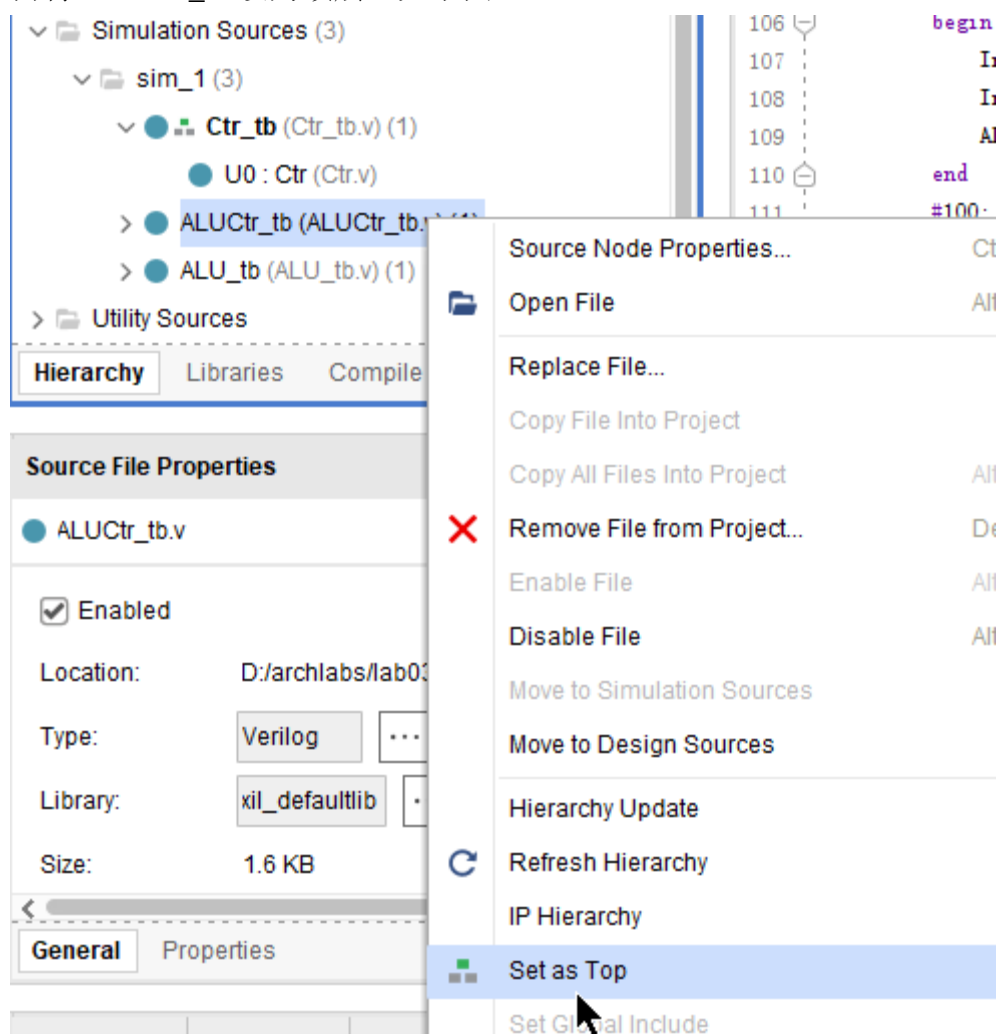
29     always @ (ALUOp or Funct )
30     begin
31         casex ({ALUOp, Funct})
32             8'b00xxxxxx : ALUCtrOut = 4'b0010;
33             //add orther few situations here
34             ...
35         endcase
36     end
37
38

```

注：{a,b}是 verilog 位拼接运算符

4.1.4 仿真测试

1. 新建 ALUCtr_tb
2. 在测试文件中设定不同的输入，覆盖全部情形
3. 需将 ALUCtr_tb 设为顶层，如下图：



4. 下面给出仿真样例：

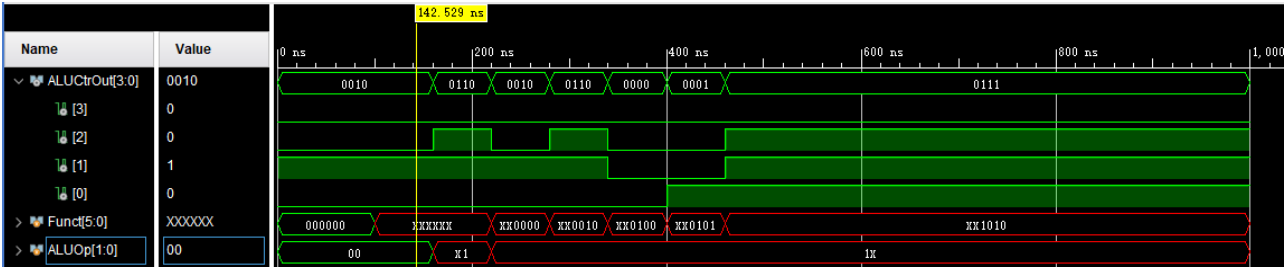


图 A 仿真波形

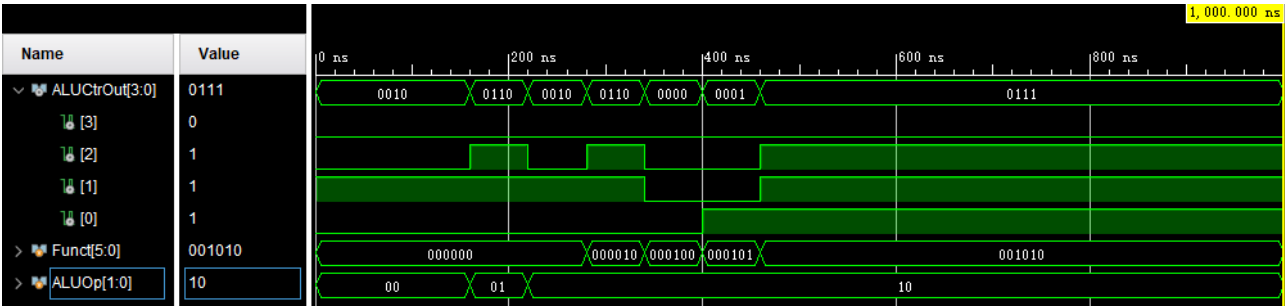


图 B 仿真波形

4. 注意图 A 和图 B 的代码区别

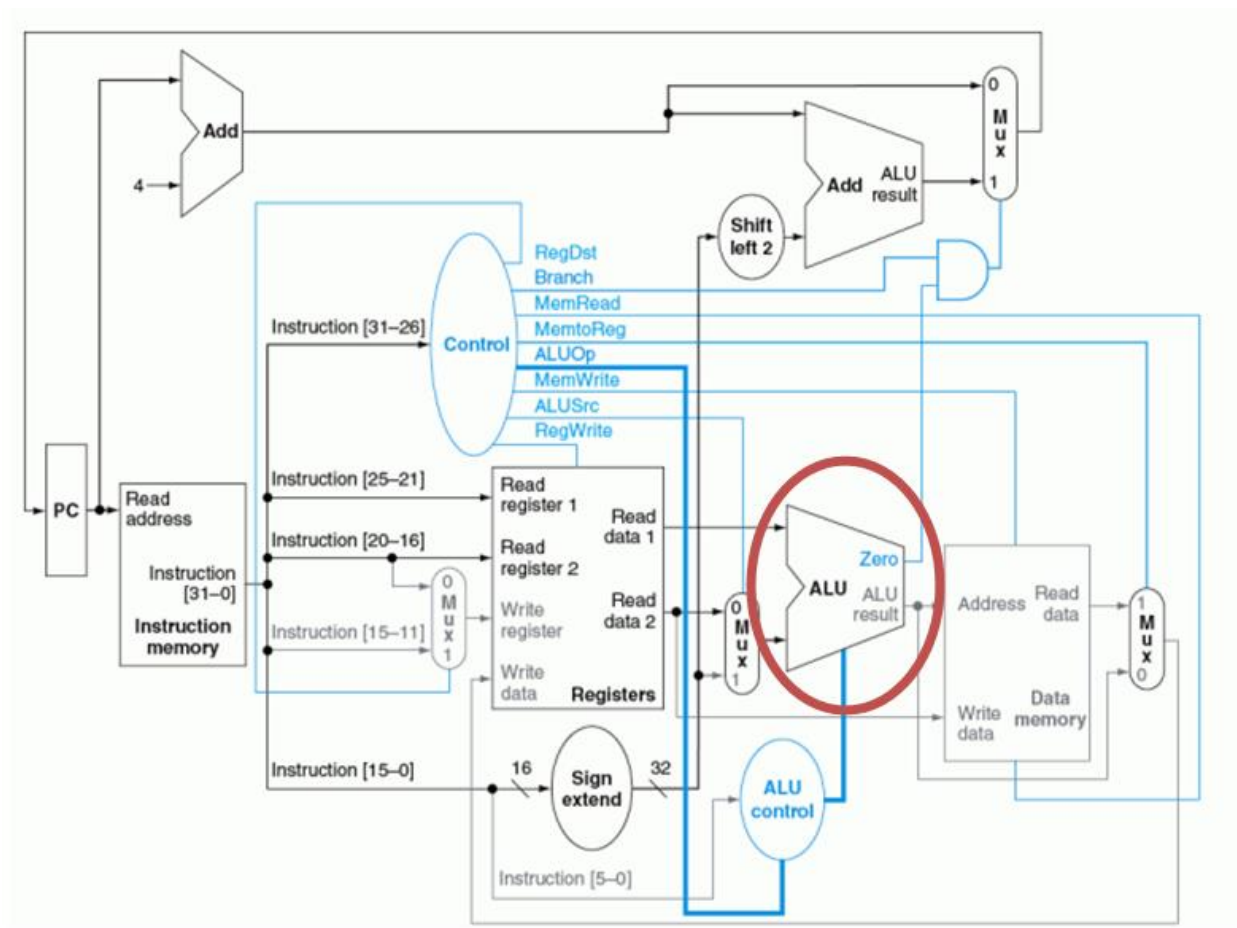
4.2 实验报告

5. ALU

5.1 实验描述

5.1.1 模块描述

根据 ALUCtr, ALU 对两个 输入执行对应的操作。ALURes 为输出结果。若减法操作 ALURes 的结果为 0 时, 则 Zero 输出置为 1。



5.1.2 新建模块源文件

略

5.1.3 实现功能

用 verilog 代码实现 ALU 功能。

实现方式可以是 case 语句；这里给出另一种参考样例，如下图：

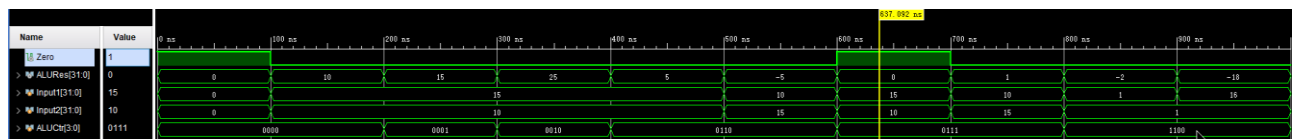
```
21 module Alu(input1, input2, aluCtr, zero, aluRes);
22     input [31:0] input1;
23     input [31:0] input2;
24     input [3:0] aluCtr;
25     output zero;
26     output [31:0] aluRes;
27     reg zero;
28     reg [31:0] aluRes;
29
30     always @ (input1 or input2 or aluCtr)
31     begin
32         if (aluCtr == 4'b0010) // add
33             aluRes = input1 + input2;
34         else if(aluCtr == 4'b0110) // sub
35             begin
36                 aluRes = input1 - input2;
37                 if (aluRes == 0)
38                     zero = 1;
39                 else
40                     zero = 0;
41             end
42         // add and, or, slt here
43     end
44
45 endmodule
```

注意 变量名的大小写

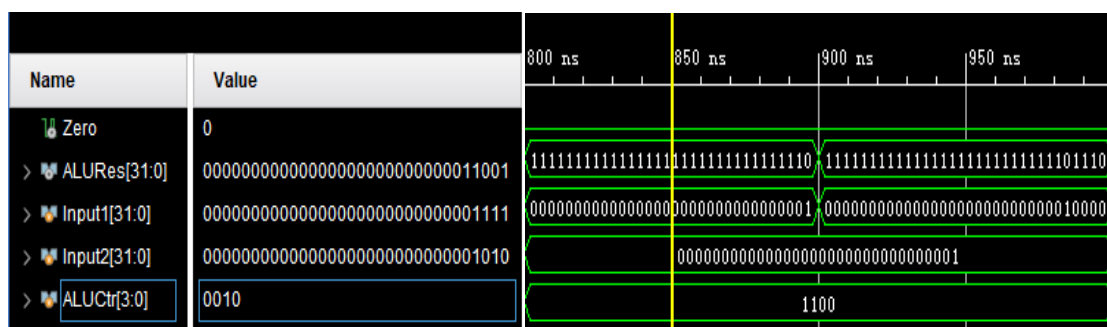
5.1.4 行为仿真

1. 新建测试文件 ALU_tb

2. 下面给出仿真样例：



ALU 仿真波形



NOR 运算的仿真波形

5.2 实验报告