Department of Computer Science & Engineering



# 计算机系统结构实验指导书-LAB3

# **1. OVERVIEW**

### 1.1 实验名称

简单的类 MIPS 单周期处理器功能部件的设计与实现(一)

# 1.2 实验目的

- 1. 理解主控制部件或单元、ALU 控制器单元、ALU 单元的原理
- 2. 熟悉所需的 Mips 指令集
- 3. 使用 Verilog HD 设计与实现主控制器部件(Ctr)
- 4. 使用 Verilog 设计与实现 ALU 控制器部件(ALUCtr)
- 5. ALU 功能部件的实现
- 6. 使用 Vivado 进行功能模块的行为仿真

# 1.3 实验报告与验收办法

按指定时间提交电子版实验报告和工程文件。本实验线上自查分 析三个仿真结果并截图

# 1.4 注意事项

本实验开始请务必按照实验指导书给定的单元部件、控制线路、数据通路 等部件图的规范来进行实验(可参见 Computer\_Organization\_and\_Design  $3^{rd}$  or  $4^{th}$  edition)

### 1.5 实验预计时间

150 分钟

# 2. 新建工程

2.1 实验描述

# 2.1.1 新建一个工程

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

🝌 New Project					×
Project Name Enter a name for yo	our project and specify a directory where th	ne project data files will b	e stored.		4
<u>P</u> roject name:	lab03				
Project location:	E:/Archlabs				⊗ …
🕑 Create projec	t subdirectory				
Project will be cr	eated at: E:/Archlabs/lab03				
?		< <u>B</u> ack	<u>N</u> ext≻	Einish	Cancel

3. 选择 开发板的 FPGA 参数: Product Category: ALL Family: Kintex-7 Package: ffg676 Speed: -2 Tempeature: 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 等功能单元输出正确的控制信号。

R	орсо	de		rs			rt			rd	sham	t		funct	
	31	26	25		21	20		16	15	11	10	6	5		0
I .	орсо	de		rs			rt				immedi	ate			
	31	26	25		21	20		16	15						0
J	орсо	de							ac	ddress					
	31	26	25												0

图 1. Mips 基本指令格式



图 2. 简单的 Mips 处理器主控制器单元模块的 IO 定义

# 3.1.2 新建模块 Verilog 文件

1. 定义其 I/O 端口

🝌 D	efine Module						×
Def For M P	ine a module a each port spec SB and LSB va orts with blank	nd specify I/( ified: lues will be names will r	D Ports ignored not be v	to add t d unless written.	o your s its Bus	ource file. column is checked.	4
Мо	dule Definition						
	Module name:	Ctr					8
	I/O Port Definit	ions					
	+ -	<b>t</b>   +					
	Port Name	Direction	Bus	MSB	LSB		
	Branch	output 🗸		0	0		<u>^</u>
	ALUOp	output 🐱		0	0		
	Jump	output 🐱		0	0		~
?	)					ОК	Cancel

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

PS:模块创建时自动生成的任何一句头信息都不要删除、也不更改默认值

# 3.1.3 编写译码功能

Input or output	Signal name	<b>R-format</b>	٦w	SW	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	0p3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	Х	Х
	ALUSrc	0	1	1	0
	MemtoReg	0	1	Х	Х
	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

图 3. 主控制模块的真值表

注意: 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: Iw	100011				
Ⅰ型: sw	101011				
I型: beq	000100				
J型: J	000010				

图 4. 指令操作码

用 verilog HDL 描述上述真值表,实现方式多种多样,这里给出一种使用 case 语句的参考样例(你需要写出所有真值表情况),如下图:

		91	
14	reg RegDst;	92	//add beg
15	reg ALUSrc;	93	//6'bxxxxx;
16	reg MemToReg;	94	begin
17	reg RegWrite;	95	
18	reg MemRead;	96	end
19	reg MemWrite;	97	
20	reg Branch;	98	//add lw
21	reg [1:0] ALUOp;	99	//add sw
22	reg Jump;	100	//add Jump
23		101	//add bump
24	always @(OpCode)	101	dofault
25	begin	102	derault:
26	case (Opcode)	103	begin DesDet - 0.
27	6.B000000: //R type	104	RegDst = 0;
20	PegIn PegDet = 1.	105	ALUSTC = 0;
30	RegDSC = 1, MUSrc = 0:	106	MemToReg = 0;
31	MemToPeg = 0:	107	RegWrite = 0;
32	RegWrite = 1:	108	MemRead = 0;
33	MemBead = 0:	109	MemWrite = 0;
34	MemWrite = 0:	110	Branch = 0;
35	Branch = 0:	111	ALUOp = 2'b00;
36	ALUOp = 2'b10;	112	Jump = 0;
37	Jump = 0;	113	end
38	end	114	endcase

# 3.1.4 功能仿真

1. 新建激励文件 Ctr\_tb

🝌 Add Sources

#### Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on a to your project.

Specify simulation set: 📄 sim_1		~		
+,   -   +   +	🍌 Create Sour	rce File	×	
	Create a new s project.	4	elow	
	<u>F</u> ile type:	Verilog	~	
	F <u>i</u> le name:	Ctr_tb	$\otimes$	
	Fil <u>e</u> location:	😜 <local project="" to=""></local>	~	ile
<ul> <li>Scan and add RTL include files</li> <li>Copy sources into project</li> </ul>	?	ОКСа	incel	

2. 添加激励即输入信号的控制。设定不同的输入,覆盖所有的情况,以保证 逻辑正确:

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

...... PS:编写激励代码时要像 Lab1 或 Lab2 一样,需对要仿真的模块进行实例化

3. 进行行为仿真,观察波形,分析仿真结果是否满足当初的设计。若由误,修 改代码,重新仿真。下面给出一个仿真波形样例:

									690.000 ns	· · · · · · · · · · · · · · · · · · ·
Name	Value	1 <sup>0 ns</sup>		200 ns		400 ns		600 ns		1800 ns
> 😽 OpCode[5:0]	010101		000000		100011	101011	000100	000010	0	10101
🕌 RegDst	0									
14 ALUSrc	0									
谒 MemToReg	0									
🐻 RegWrite	0									
🐻 MemRead	0									
🐻 MemWrite	0									
🔓 Branch	0									
谒 Jump	0									
> 😼 ALUOp[1:0]	00		10		0	0	01	<u>k</u>	00	<u> </u>

图 5. Ctr 的仿真波形

 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。(属本次实验自查点之一、截图存档)

**3.2** 工程设计、下载验证 (本次不做)

# 4. ALU 控制单元模块

#### 4.1 实验描述

### 4.1.1 模块描述

算数逻辑单元 ALU 的控制单元(ALUCtr)是根据主控制器的 ALUOp 控制 信号来判断指令类型,并依据指令的后 6 位区分 R 型指令。综合这两种输入,以控制 ALU 做正确操作。

R	орсо	de		rs			rt			rd		shamt			funct	
	31	26	25		21	20		16	15	11	1	10	6	5		0
1	орсо	de		rs			rt					immedia	te			
	31	26	25		21	20		16	15							0
J	орсо	de							ac	ldress						
	31	26	25													0

图 1. R、I、J型指令格式(参见 Mips 指令集)



图 2. ALU 控制器模块的 IO 定义

### 4.1.2 新建模块源文件

步骤略(可参照之前的方法,比如 3.1.2的步骤)

**PS**:现在起所有的实验主要基于同学所学的理论知识、前期的实验经验,凭借同学们的聪明才智和实干探索精神独自完成 (若实际遇到不解之处可搜索问题资源的解决经验、代码编写技巧、设计实现思路等)

# 4.1.3 编写译码功能

ALU Control (即上图的输出 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

图 3. 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

图 4. Funct, ALUOp 与 ALU Control 编码关系

AL	UOp			Funct				
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	FO	Operation
0	0	Х	Х	Х	Х	Х	Х	0010
Х	1	Х	Х	Х	Х	Х	Х	0110
1	x	Х	Х	0	0	0	0	0010
1	x	Х	Х	0	0	1	0	0110
1	X	Х	Х	0	1	0	0	0000
1	Х	Х	Х	0	1	0	1	0001
1	Х	Х	Х	1	0	1	0	0111

图 5. ALU 控制单元输入输出真值表

用 verilog 代码写出上述真值表内容。

实现方式多种多样,这里给出一种使用 casex 语句的参考样例(余下情况需自己补全),如下图:

always @ (ALUOp or Funct )
begin
casex ({ALUOp,Funct})
8'b00xxxxxx : ALUCtrOut = 4'b0010;
<pre>//add orther few situations here</pre>
endcase

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:

🗸 📄 Simulati	on Sources (3)		106 🤤	begin
v 🚍 sim_	<b>1</b> (3)		107	Iı T.
~ • :	Al			
	U0 : Ctr (Ctr.v)		110 🖨	end
> • /	ALUCtr_tb (ALUCtr_tb.	1.741	111	#100·
	ALU_tb (ALU_tb.v) (1)		Source Node Properties	Ct
> 🚍 Utility So	urces		Open File	Alt
Hierarchy	Libraries Compile		Replace File	
			Copy File Into Project	
Source File Pro	operties		Copy All Files Into Project	Alt
ALUCtr_tb.v			Remove File from Project	De
			Enable File	Alt
Enabled			Disable File	Alt
Location:	D:/archlabs/lab0(		Move to Simulation Sources	
Туре:	Verilog ···		Move to Design Sources	
Library:	xil_defaultlib		Hierarchy Update	
Size:	1.6 KB	С	Refresh Hierarchy	
<			IP Hierarchy	
General Pr	operties	<b>.</b>	Set as Top	
7.10			Set Global Include	

### 4. 下面给出仿真波形样例:

			142, 529 a	as							
Name	Value	10 ns		200 ns			400 ns		600 ns	1800 ns	<sup>1, 000</sup>
✓ ₩ ALUCtrOut[3:0]	0010	0010	0110	0010	0110	0000	0001	<	0111		
14 [3]	0										
[2]	0										
14 [1]	1										
14 [0]	0										
> 😻 Funct[5:0]	XXXXXXXX	000000 3	*****	XX0000	XX0010	XX0100	XX0101		xx 10 10		
> 😻 ALUOp[1:0]	00	00	Х1						111		5



							1, 000. 000 ns
Name	Value	10 ns	200 ns		400 ns	1600 ns	800 ns
✓ ₩ ALUCtrOut[3:0]	0111	0010	0110 0010	0110 0000	0001	0111	_')
16 [3]	0						
12 [2]	1						
l <b>a</b> [1]	1						
14 [0]	1						
> 😻 Funct[5:0]	001010	000000		000010	000101	001010	
> 😻 ALUOp[1:0]	10	X	01			10	)

图 B. 仿真波形

以上两种仿真波形都正确,图 A 和图 B 的代码有何区别(属本次实验自查点之一、 截图存档)

# **4.2** 工程设计、下载验证 (本次不做)

# 5. ALU 模块

# 5.1 实验描述

# 5.1.1 模块描述

算术逻辑单元 ALU 根据 ALUCtr 的控制信号将两个输入执行与之对应的操作。 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。



图 6. ALU 模块的 IO 定义

#### 5.1.2 新建模块源文件

# 5.1.3 设计功能

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

aluCtrOut[3:0]的值与 ALU 操作的对应关系如下:

注意: beq 时 实际是个减法操作

#### 用 verilog 代码编写 ALU 功能 实现方式可以是 case 语句;但这里给出另一种参考方案(仅部分代码):

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

注意 变量名的大小写前后一致

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(属本次实验自查点之一、截图存档):

									637.092 ns			
Name	Value	10 ns	100 ns	200 ns	300 ns	400 ns	1500 ns	600 ns		1 <sup>700</sup> ns	800 ns	1900 ns
18 Zero	1											
> 🛿 ALURes[31:0]	0		10	15	25	5	-5		0	1	-2	-18
> 🚺 Input1[31:0]	15			1	5		10		15	10	1	16
> 😽 input2[31:0]	10			1	0		15		10	15		·;
> 😽 ALUCtr[3:0]	0111	0	000	0001	0010	01	10		0	11	11	
												4.1

ALU 仿真波形(含 Zero 为1时)

800 ns	850 ns	900 ns	950 ns
(111111111111111111	1111111111111111	111111111111111111	1111111111101110
000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	0000000000010000
<	000000000000000000000000000000000000000	000000000000000000000000000000000000000	
	11	.00	

NOR 运算的二进制显示

# **5.2** 工程设计、下载验证 (本次不做)

# 5.3 实验报告

Ctr、ALUctr 和 ALU 这三部分的实验报告也可以合并起来总写