

计算机系统结构实验

## 实验 3 报告

简单的类 MIPS 单周期处理器  
部件实现-控制器，ALU

李子龙

518070910095

2021 年 5 月 19 日

### 目录

<b>1</b>	<b>实验目的</b>	<b>2</b>
<b>2</b>	<b>原理分析</b>	<b>2</b>
2.1	主控制器单元模块 Ctr . . . . .	2
2.2	算术逻辑单元控制器模块 ALUCtr . . . . .	3
2.3	ALU 模块 . . . . .	3
<b>3</b>	<b>代码实现</b>	<b>4</b>
3.1	主控制模块 Ctr . . . . .	4
3.2	算术逻辑单元控制器模块 ALUCtr . . . . .	5
3.3	ALU 模块 . . . . .	6
<b>4</b>	<b>仿真结果</b>	<b>7</b>
4.1	主控制模块 Ctr . . . . .	7
4.2	算术逻辑单元控制器模块 ALUCtr . . . . .	8
4.3	ALU 模块 . . . . .	8
<b>5</b>	<b>实验心得</b>	<b>9</b>

1 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

2 原理分析

2.1 主控制器单元模块 Ctr

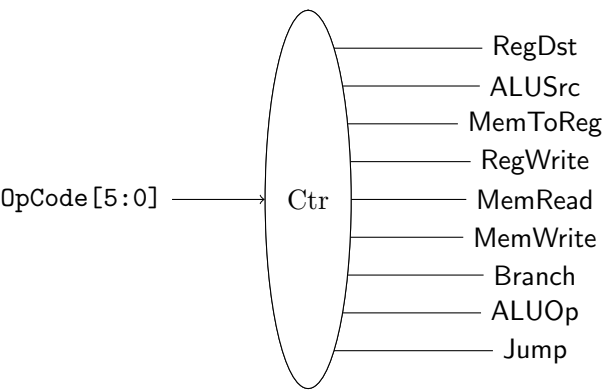


图 1: 主控制器单元模块

主控制器单元对输入指令 `Instruction[31:26]`（即 `opCode` 部分）译码。指令与对应的 `opCode` 对应列表于表 1。

表 1: 指令操作码表

指令	opCode
R: <code>add,sub,and,or,slt</code>	000000
I: <code>lw</code>	100011
I: <code>sw</code>	101011
I: <code>beq</code>	000100
J: <code>j</code>	000010

表 2: 主控制模块真值表

信号名	R	lw	sw	beq	j
RegDst	1	0	X	X	X
ALUSrc	0	1	1	0	X
MemToReg	0	1	X	X	X
RegWrite	1	1	0	0	0
MemRead	0	1	0	0	0
MemWrite	0	0	1	0	0
Branch	0	0	0	1	0
ALUOp	10	00	00	01	XX
Jump	0	0	0	0	1

译码完成后，需要针对对应的针脚输出对应的值，输出参数表如表 2 所示。

## 2.2 算术逻辑单元控制器模块 ALUCtr

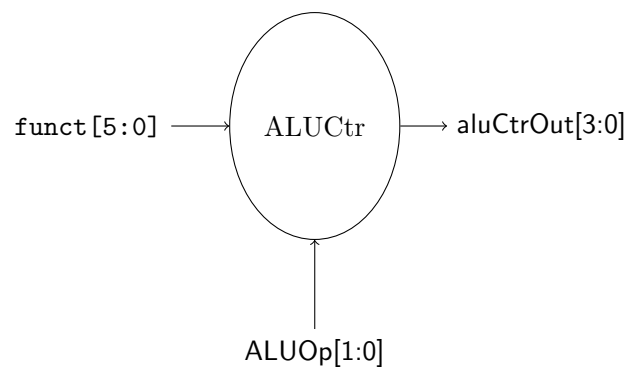


图 2: ALUCtr 模块

算术逻辑单元控制器模块（ALUCtr）根据主控制器（Ctr）传来的 `ALUOp` 来判断指令类型，并根据 `Instruction[5:0]`（即 `funct`）来判断指令类型。并通过 `aluCtrOut[3:0]` 输出对应信号。对应列表列于表 3。

表 3: ALUCtr 输入输出关系

指令	详细指令	ALUOp	funct	ALU 操作	altCtrOut
lw	load word	00	XXXXXX	add	0010
sw	store word	00	XXXXXX	add	0010
beq	branch equal	01	XXXXXX	subtract	0110
R	add	10	100000	add	0010
R	sub	10	100010	subtract	0110
R	and	10	100100	and	0000
R	or	10	100101	or	0001
R	slt	10	101010	set less than	0111

2.3 ALU 模块

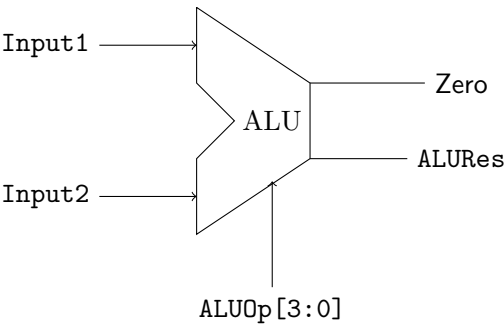


图 3: ALU 模块

ALU 模块对输入 `Input1` 和 `Input2` 进行数值运算。`ALUOp[3:0]` 将会指定操作模式，并输出为 `ALURes`，`Zero` 将指示结果是否为 0。操作模式与对应功能列于表 4。

表 4: ALU 操作

ALUOp	功能
0000	按位和
0001	按位或
0010	和
0110	减
0111	小于则设定
1100	按位或非

### 3 代码实现

#### 3.1 主控制模块 Ctr

使用了 case 语句判断对应的 opCode 以输出。例如对于 R 型语句：

Listing 1: Ctr.v

```

1  always @(opCode) begin
2      case (opCode)
3          6'b000000:    // R type
4              begin
5                  RegDst    = 1;
6                  ALUSrc    = 0;
7                  MemToReg  = 0;
8                  RegWrite  = 1;
9                  MemRead   = 0;
10                 MemWrite  = 0;
11                 Branch    = 0;
12                 ALUOp     = 2'b10;
13                 Jump      = 0;
14             end
15             //...
16         endcase
17     end

```

其余的部分对照表 2 编写。

### 3.2 算术逻辑单元控制器模块 ALUCtr

结合 aluOp 和 funct 编码，使用 case 语句判断，并根据表 3 输出。

Listing 2: ALUCtr.v

```

1 module ALUCtr(
2     input [5:0] funct,
3     input [1:0] aluOp,
4     output [3:0] aluCtrOut
5 );
6
7     reg [3:0] ALUCtrOut;
8
9     always @(aluOp or funct) begin
10         casex ({aluOp, funct})
11             8'b00xxxxxx: ALUCtrOut = 4'b0010;
12             8'bx1xxxxxx: ALUCtrOut = 4'b0110;
13             8'b1xxx0000: ALUCtrOut = 4'b0010;
14             8'b1xxx0010: ALUCtrOut = 4'b0110;
15             8'b1xxx0100: ALUCtrOut = 4'b0000;
16             8'b1xxx0101: ALUCtrOut = 4'b0001;
17             8'b1xxx1010: ALUCtrOut = 4'b0111;
18             default:     ALUCtrOut = 4'b1111;
19         endcase
20     end
21
22     assign aluCtrOut = ALUCtrOut;
23 endmodule

```

### 3.3 ALU 模块

按照表 4 构建 case 语句。注意 slt 采用符号数比较（事实上 sltu 是无符号数比较）。

Listing 3: ALU.v

```

1 always @(input1 or input2 or aluCtr) begin
2     case (aluCtr)
3         4'b0010: ALURes = input1 + input2;

```

```

4      4'b0110: ALURes = input1 - input2;
5      4'b0000: ALURes = input1 & input2;
6      4'b0001: ALURes = input1 | input2;
7      4'b0111: begin          // slt
8          if($signed(input1) < $signed(input2))
9              ALURes = 1;
10         else ALURes = 0;
11     end
12     4'b1100: ALURes = ~(input1 | input2);
13     default: ALURes = 0;
14 endcase
15 if(ALURes == 0)
16     Zero = 1;
17 else
18     Zero = 0;
19 end

```

## 4 仿真结果

### 4.1 主控制模块 Ctr

主控制模块的激励信号如下所示。

Listing 4: Ctr\_tb.v

```

1  initial begin
2      OpCode = 0;
3      #100;
4      #100 OpCode = 6'b000000;
5      #100 OpCode = 6'b100011;
6      #100 OpCode = 6'b101011;
7      #100 OpCode = 6'b000100;
8      #100 OpCode = 6'b000010;
9      #100 OpCode = 6'b010101;
10 end

```

仿真结果如图 4 所示。对照表 2 可得信号一致。

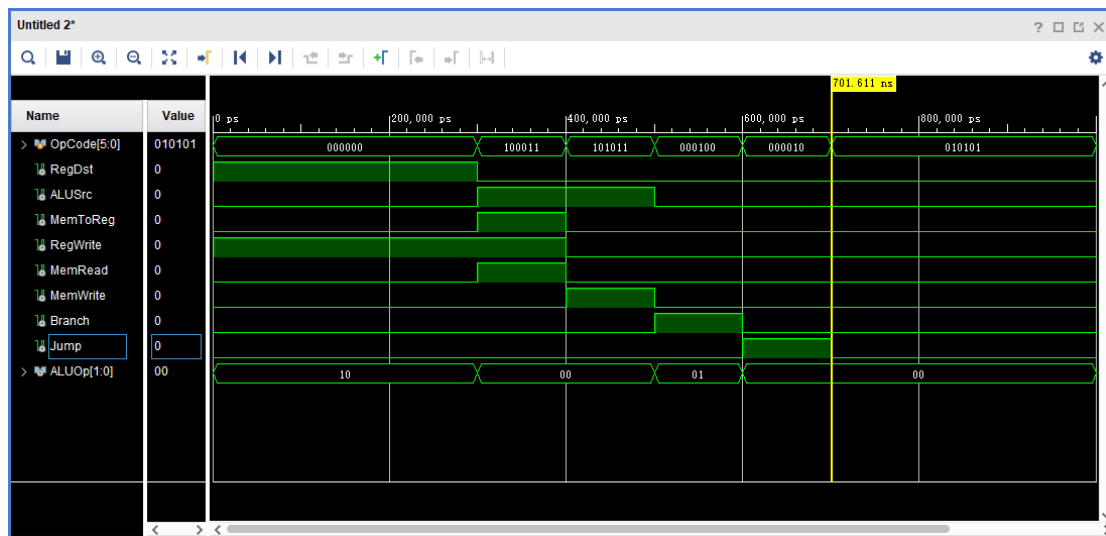


图 4: Ctr 仿真波形

## 4.2 算术逻辑单元控制器模块 ALUCtr

算术逻辑单元控制器模块的激励信号如下所示。

Listing 5: ALUCtr\_tb.v

```

1  initial begin
2      Funct = 0;
3      ALUOp = 0;
4      #100;
5
6      #100;
7      Funct = 6'b0000000;
8      ALUOp = 2'b00;
9
10     #100;
11     Funct = 6'b0000000;
12     ALUOp = 2'b01;
13
14     #100;
15     Funct = 6'b0000000;
16     ALUOp = 2'b10;
17
18     #100;
19     Funct = 6'b000010;
20     ALUOp = 2'b10;
21
22     #100;
23     Funct = 6'b000100;
24     ALUOp = 2'b10;
25
26     #100;
27     Funct = 6'b000101;
28     ALUOp = 2'b10;
29
30     #100;
31     Funct = 6'b001010;
32     ALUOp = 2'b10;
33
34 end

```





图 5: ALUCtr 仿真结果

激励结果如图 5 所示。结果与表 3 一致。

### 4.3 ALU 模块

ALU 模块的激励信号如下所示。

Listing 6: ALU_tb.v	
1	<code>initial begin</code>
2	<code>Input1 = 0;</code>
3	<code>Input2 = 0;</code>
4	<code>ALUCtr = 0;</code>
5	
6	<code>#100;</code>
7	<code>Input1 = 15;</code>
8	<code>Input2 = 10;</code>
9	
10	<code>#100;</code>
11	<code>ALUCtr = 4'b0001;</code>
12	
13	<code>#100;</code>
14	<code>ALUCtr = 4'b0010;</code>
15	
16	<code>#100;</code>
17	<code>ALUCtr = 4'b0110;</code>
18	
19	<code>#100;</code>
20	<code>Input1 = 10;</code>
21	<code>Input2 = 15;</code>
22	
23	<code>#100;</code>
24	<code>Input1 = 15;</code>
25	<code>Input2 = 10;</code>
26	<code>ALUCtr = 4'b0111;</code>
27	
28	<code>#100;</code>
29	<code>Input1 = 10;</code>
30	<code>Input2 = 15;</code>
31	
32	<code>#100;</code>
33	<code>Input1 = 1;</code>

34	Input2 = 1;	38	Input1 = 16;
35	ALUCtr = 4'b1100;	39	end
36			
37	#100;		

仿真结果如图 6 所示。完成了对应的功能。

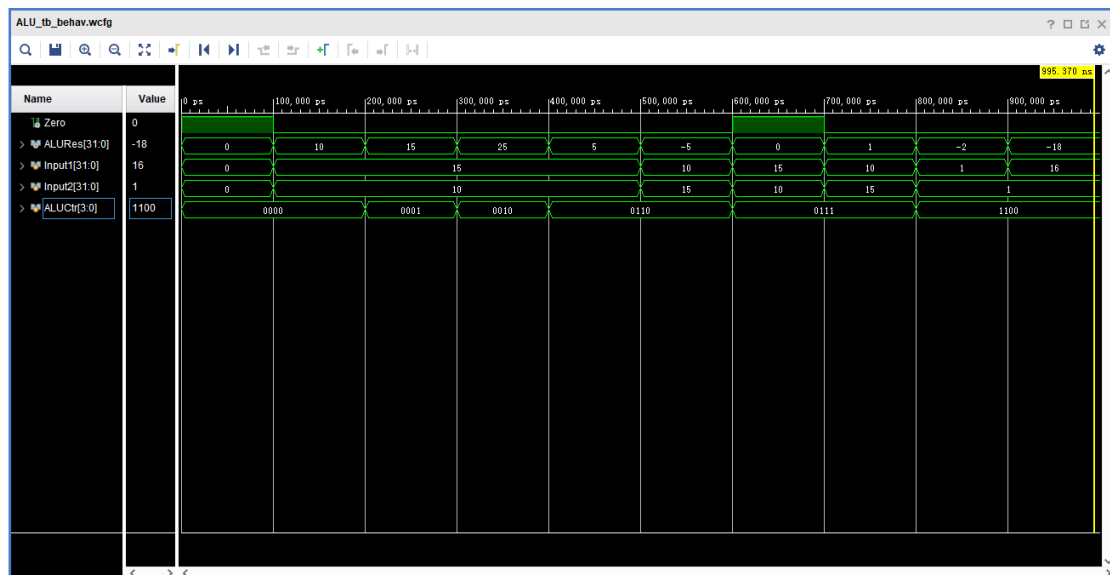


图 6: ALU 仿真波形

## 5 实验心得

本实验实现了 Ctr, ALUCtr, ALU 等模块，为之后的完整实现奠定了基础。通过本次实验，我熟悉了 Verilog 的相关语法（特别是 `case` 和连接 `{0p1,0p2}`），意识到在某些方面其与 C 语言的相似性（比如按位运算符）以及其 Visual Basic 类似的环境语法（`begin, ..., end`），熟悉了模块编写 – 激励编写 – 仿真波形 – 调试流程。熟悉了 MIPS 的译码和运算流程，对理解处理器结构很有帮助。