

# 华中科技大学

## 2016

### 计算机组成原理

### · 实验报告 ·

专    业： 计算机科学与技术

班    级： CS1409

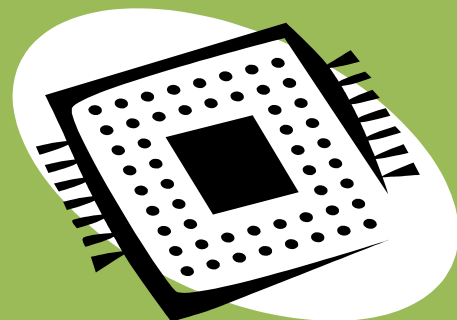
学    号： U201414808

姓    名： 王林

电    话： 13419606224

邮    件： 424435985@qq.com

完成日期： 2017-01-14



计算机科学与技术学院

# 华中科技大学课程实验报告

---

## 目 录

<b>1</b>	<b>运算器实验.....</b>	<b>3</b>
1.1	设计要求 .....	3
1.2	方案设计 .....	5
1.3	实验步骤 .....	9
1.4	故障与调试 .....	13
1.5	测试与分析 .....	13
<b>2</b>	<b>存储器实验.....</b>	<b>15</b>
2.1	设计要求 .....	15
2.2	方案设计 .....	16
2.3	实验步骤 .....	17
2.4	故障与调试 .....	19
2.5	测试与分析 .....	21
<b>3</b>	<b>CPU 实验.....</b>	<b>22</b>
3.1	设计要求 .....	22
3.2	方案设计 .....	23
3.3	实验步骤 .....	27
3.4	故障与调试 .....	29
3.5	测试与分析 .....	30
<b>4</b>	<b>总结与心得.....</b>	<b>32</b>
4.1	实验总结 .....	32
4.2	实验心得 .....	32
	<b>参考文献.....</b>	<b>34</b>

## 1 运算器实验

### 1.1 设计要求

1. 在 logisim 平台上，利用现有的逻辑门电路构造 32 位快速加法器，并进行电子封装。
2. 利用封装好的 32 位加法器以及 logisim 平台中现有运算部件构建一个 32 位运算器，可支持算数加、减、乘、除，逻辑与、或、非、异或运算、逻辑左移、逻辑右移，算术右移运算，支持常用程序状态标志（有符号溢出 OF、无符号溢出 CF，结果相等 Equal），运算器功能以及输入输出引脚见下表，在主电路中详细测试自己封装的运算器。

alu 引脚功能表：

表 1.1 片引脚与功能描述

引脚	输入/输出	位宽	功能描述
X	输入	32	操作数 X
Y	输入	32	操作数 Y
ALU_OP	输入	4	运算器功能码，具体功能见下表
Result	输出	32	ALU 运算结果
Result2	输出	32	ALU 结果第二部分，用于乘法指令结果高位或除法指令的余数位，其他操作为零
OF	输出	1	有符号加减溢出标记，其他操作为零
CF	输出	1	无符号加减溢出标记，其他操作为零
Equal	输出	1	Equal=(x==y)?1:0, 对所有操作有效

运算器功能表：

# 华中科技大学课程实验报告

表 1.2 运算符功能

ALU OP	十进制	运算功能	
0000	0	Result = X << Y	逻辑左移 (Y 取低五位) Result2=0
0001	1	Result = X >>> Y	算术右移 (Y 取低五位) Result2=0
0010	2	Result = X >> Y	逻辑右移 (Y 取低五位) Result2=0
0011	3	Result = (X * Y)[31:0]; Result2 = (X * Y)[63:32] 有符号	
0100	4	Result = X/Y; Result2 = X%Y 无符号	
0101	5	Result = X + Y Result2=0 (Set OF/CF)	
0110	6	Result = X - Y Result2=0 (Set OF/CF)	
0111	7	Result = X & Y Result2=0	
1000	8	Result = X   Y Result2=0	
1001	9	Result = X ⊕ Y Result2=0	
1010	10	Result = ~(X   Y) Result2=0	
1011	11	Result = (X < Y) ? 1 : 0 Signed Result2=0	
1100	12	Result = (X < Y) ? 1 : 0 Unsigned Result2=0	
1101	13	Result = Result2=0	
1110	14	Result = Result2=0	
1111	15	Result = Result2=0	

对 32 位的运算器进行封装

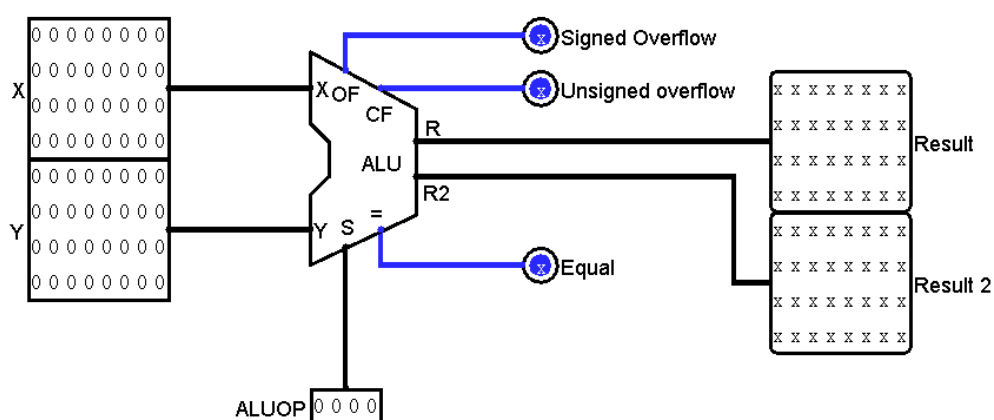


图 1.1 运算器封装图

## 1.2 方案设计

### 1.2.1 1 位全加器设计

输入  $X_i$ 、 $Y_i$ 、低位进位  $C_i$ ，通过组合逻辑得到结果  $S_i$  以及进位  $C_{i+1}$ ，列出关于  $X_i$ 、 $Y_i$ 、 $C_i$ 、 $S_i$ 、 $C_{i+1}$  的真值表，然后写出  $S_i$  和  $C_{i+1}$  关于  $X_i$ 、 $Y_i$ 、 $C_i$  的表达式，画出有基本逻辑门通过组合逻辑实现的门电路。

表 1.3 一位全加器真值表

#	$X_i$	$Y_i$	$C_i$	$S_i$	$C_{i+1}$
1	0	0	0	0	0
2	0	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	1	0	0	1	0
6	1	0	1	0	1
7	1	1	0	0	1
8	1	1	1	1	1

由真值表写出  $S_i$  和  $C_i$  的表达式：

$$S_i = X_i \oplus Y_i \oplus C_i$$

$$C_{i+1} = X_i Y_i + (X_i \oplus Y_i) C_i$$

一位全加器的总体结构图设计如下：

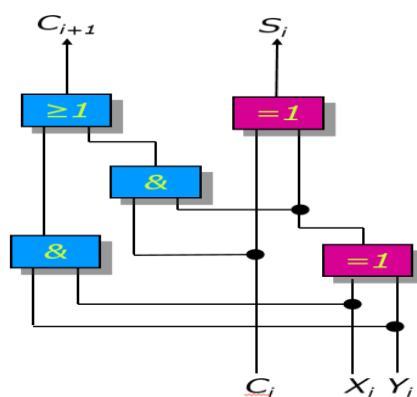


图 1.2 一位全加器的总体结构图

## 1.2.2 4 位先行进位的快速加法器

4 位进行加法运算时,采用串行方式时,高位的运算依赖于低位运算的进位输入,使得电路延迟增加。为了减少电路延时,采用并行结构,提前得到当前位的进位。

$$G_i = X_i Y_i$$

$$P_i = X_i \oplus Y_i$$

$$C_i = G_i + P_i C_{i-1}$$

则计算每一级的进位公式为:

$$C_1 = X_1 Y_1 + (X_1 \oplus Y_1) C_0 = G_1 + P_1 C_0$$

$$C_2 = G_2 + P_2 G_1 + P_2 P_1 C_0$$

$$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$$

$$C_n = G_n + P_n G_{n-1} + P_n P_{n-1} G_{n-2} + P_n P_{n-1} P_{n-2} G_{n-3} \dots + P_n P_{n-1} P_{n-2} \dots P_1 C_0$$

计算结果公示为:

$$S_4 = X_4 \oplus Y_4 \oplus C_3 = P_4 \oplus C_3$$

$$S_3 = X_3 \oplus Y_3 \oplus C_2 = P_3 \oplus C_2$$

$$S_2 = X_2 \oplus Y_2 \oplus C_1 = P_2 \oplus C_1$$

$$S_1 = X_1 \oplus Y_1 \oplus C_0 = P_1 \oplus C_0$$

利用 4 位先行进位的快速加法器实现组间进位传递,从而实现组间并行计算,成组仅为传递信息为:

$$G_4^* = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1$$

$$P_4^* = P_4 P_3 P_2 P_1$$

则成组进位传递函数为:

$$C_4 = G_4^* + P_4^* C_0$$

4 位先行进位的快速加法器电路结构图如下:

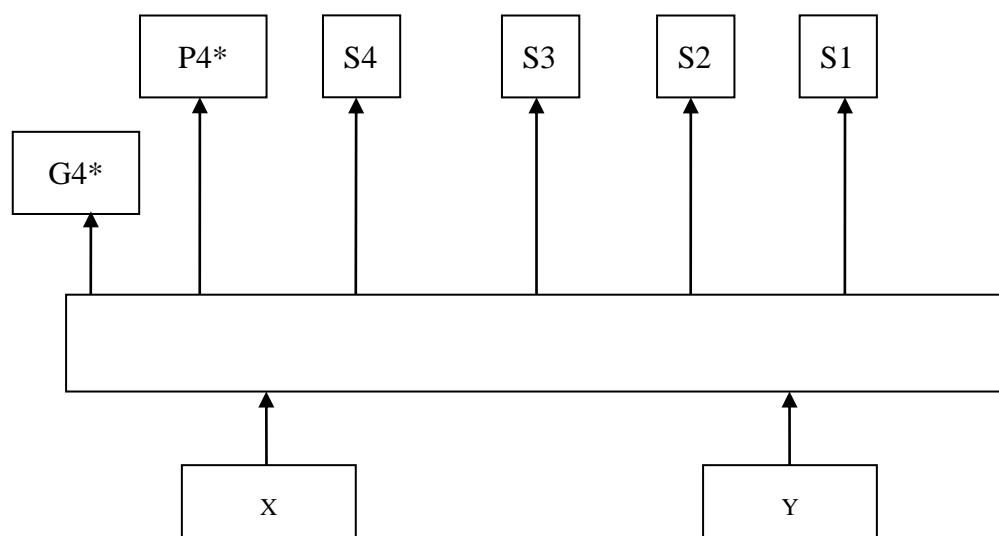


图 1.3 4 位先行进位加法器结构图

## 1.2.3 CLA74182 电路

实现组间先行进位，接收每组的进位传递信息，得到下一组的进位，同时生成高位的组间进位传递信息。

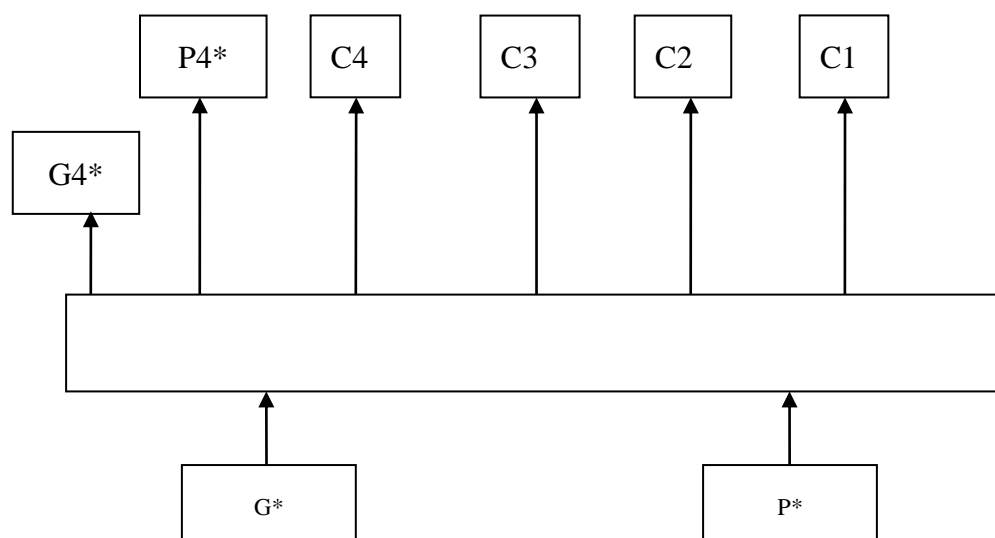


图 1.4 CLA74182 先行进位电路结构图

## 1.2.4 32 位先行进位加法电路

首先采用 4 位先行进位的快速加法器构造 16 位组内先行进位、组间先行进位的



# 华中科技大学课程实验报告

加法器电路，4 位一组，构成 4 位先行进位，然后 4 组作为一组，构成 16 位先行进位的加法器，然后两个 16 位的先行进位加法器串行构成 32 位的加法器。需要 2 片 CLA74182 电路，8 片 ALU74181 电路。32 位加法器电路结构如下：

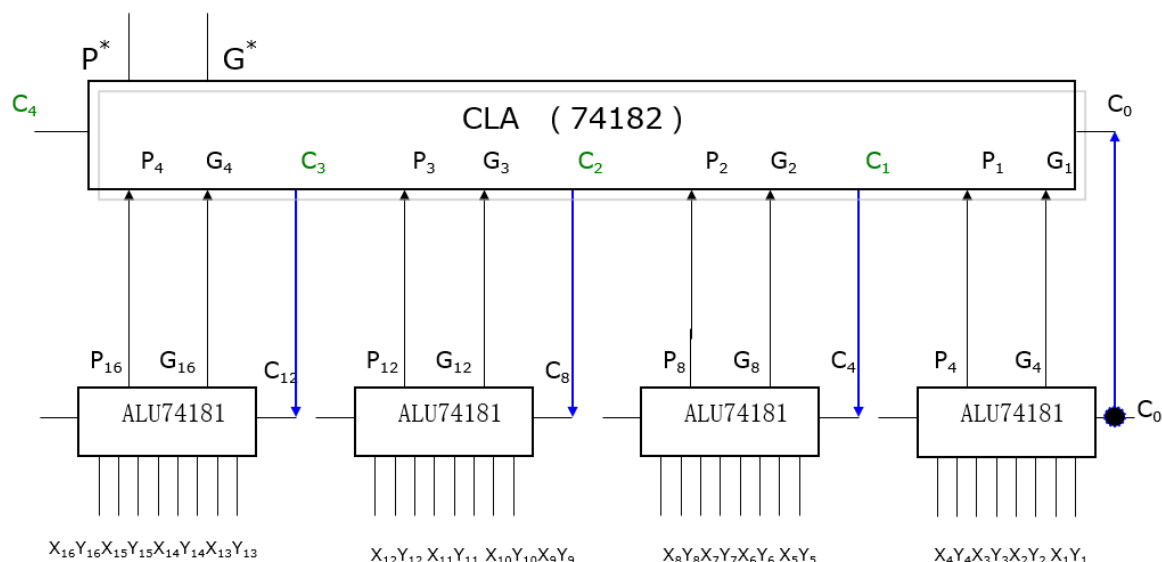


图 1.5 16 位先行进位电路结构图

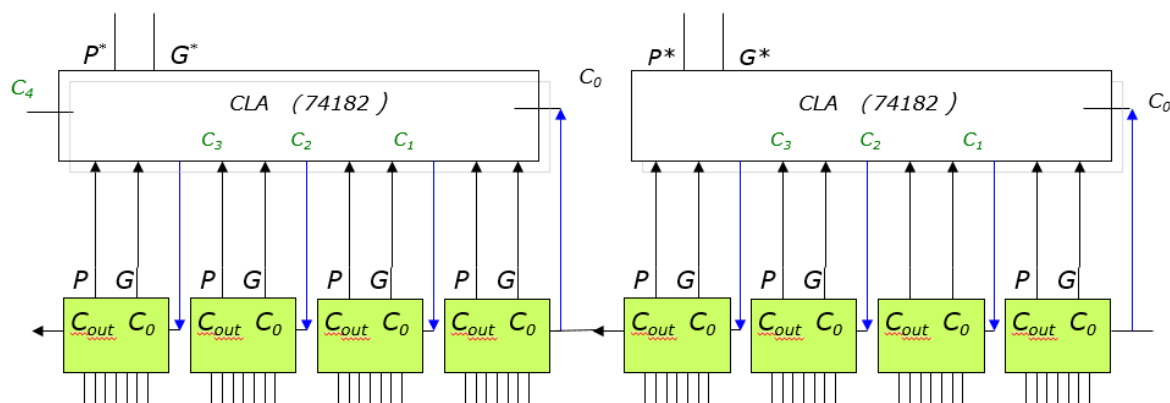


图 1.5 32 位先行进位电路结构图

## 1.2.5 32 位 ALU 运算器设计

ALU 有三个输入，分别是操作数 X、Y 以及 ALU 功能码 ALU\_OP，5 个输出，ALU 运算结果 result1、result2（乘法高位，出发余数），有符号加减溢出 OF、无符号加减一处 UOF、相等标志 Equal。利用前面实现的 32 位加法器实现加减运算，利用 logisim 现有的器件完成其他移位操作、乘除法、与操作、或操作、异或操作、或

非操作、比较操作。通过组合逻辑每次对于两个输入的操作数，一次计算出所有操作的结果，同时设置各标志，然后根据输入的 ALU 功能码，通过多路选择器选择对应的结果输出。ALU 结构如下：

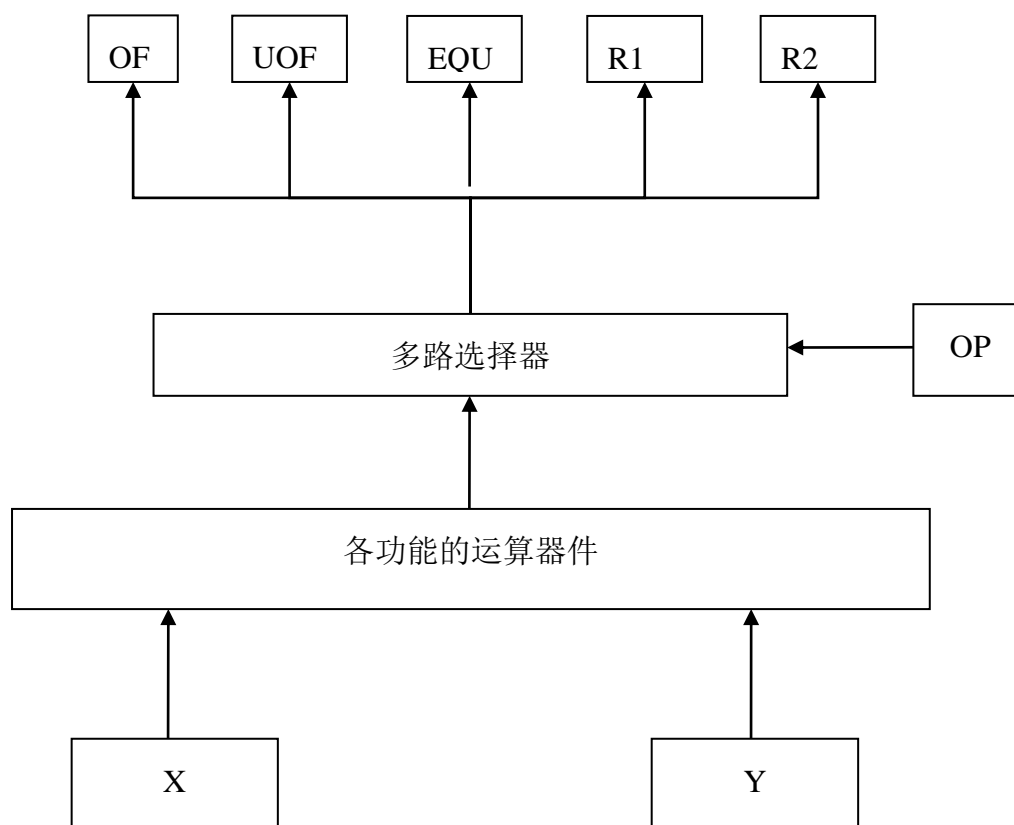


图 1.6 32 位 ALU 电路结构图

## 1.3 实验步骤

### (1) 构造 4 位具有先行进位特征的并行加法器

依照前面的设计，先完成一位的全加器，然后完成 4 为先行进位的加法器。电路图如下：

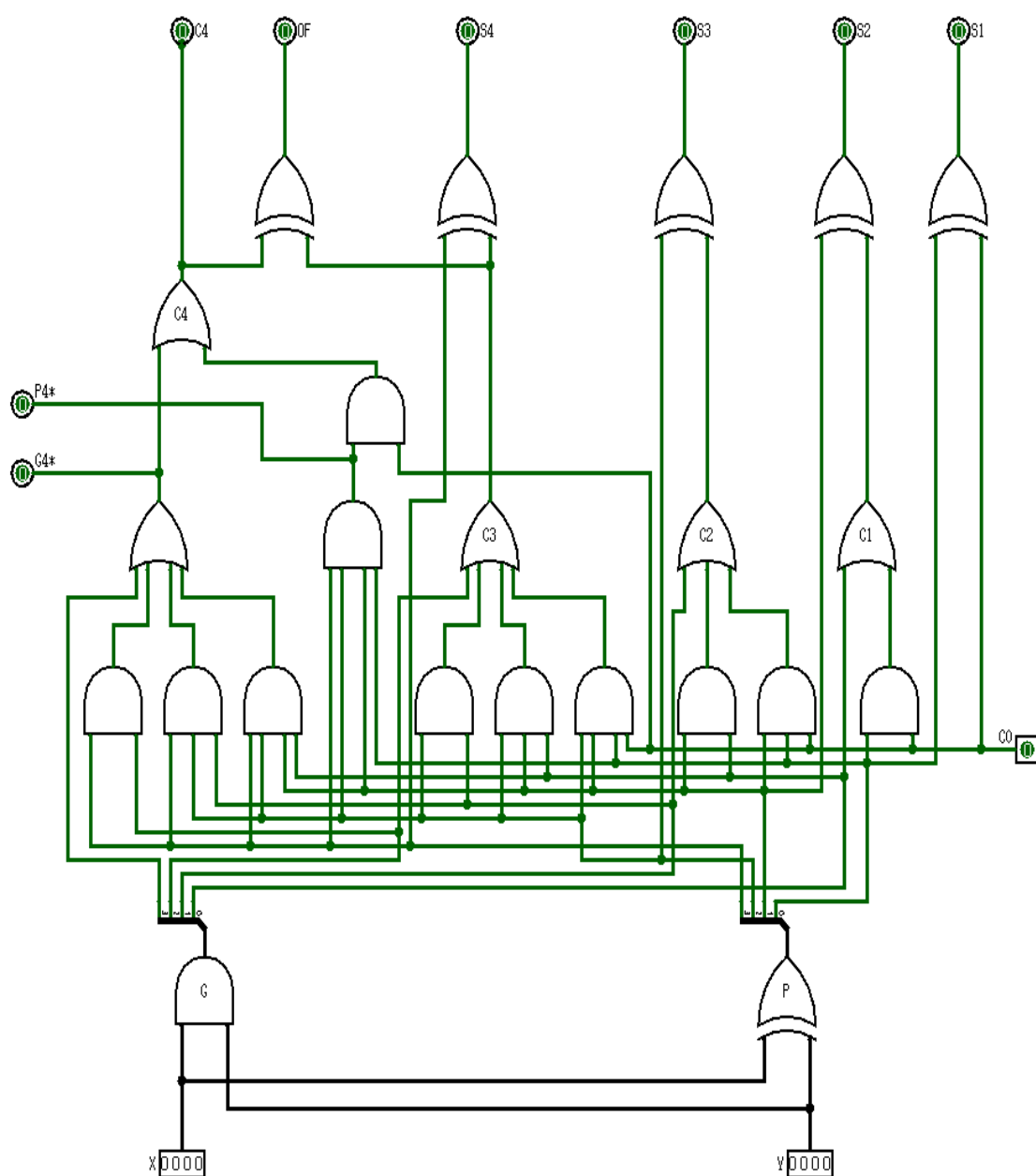
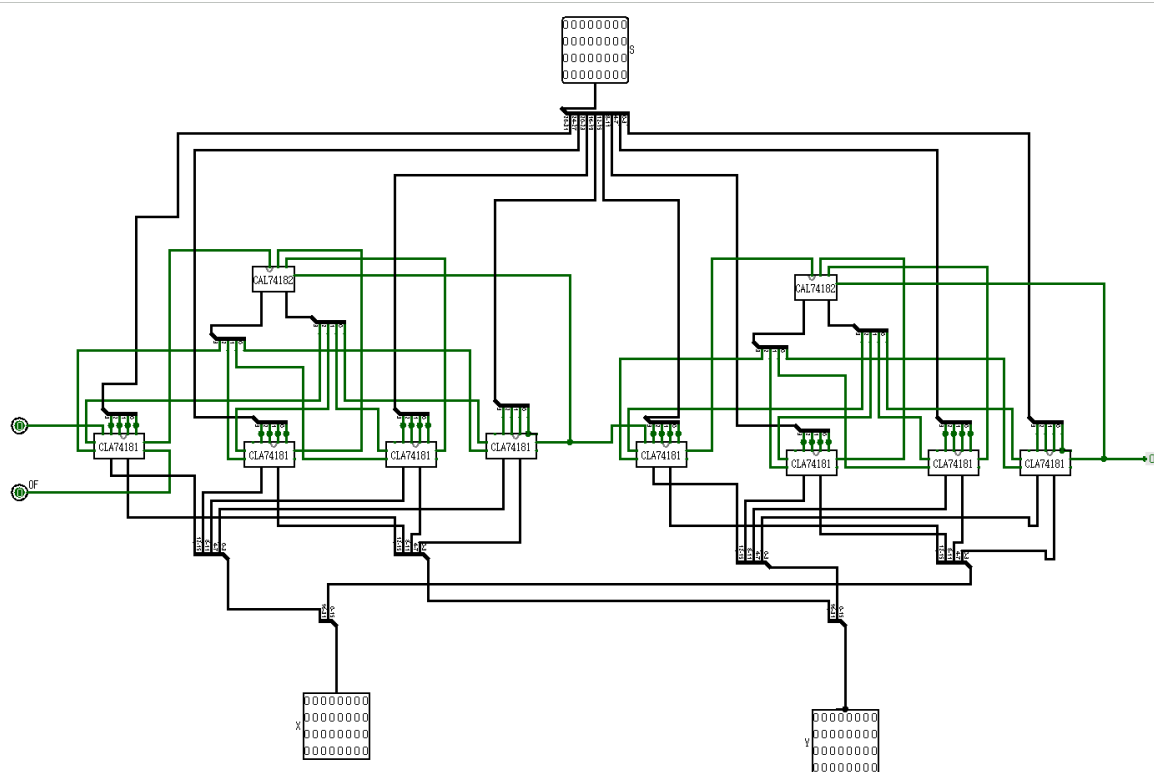


图 1.7 4 位先行进位加法器电路图

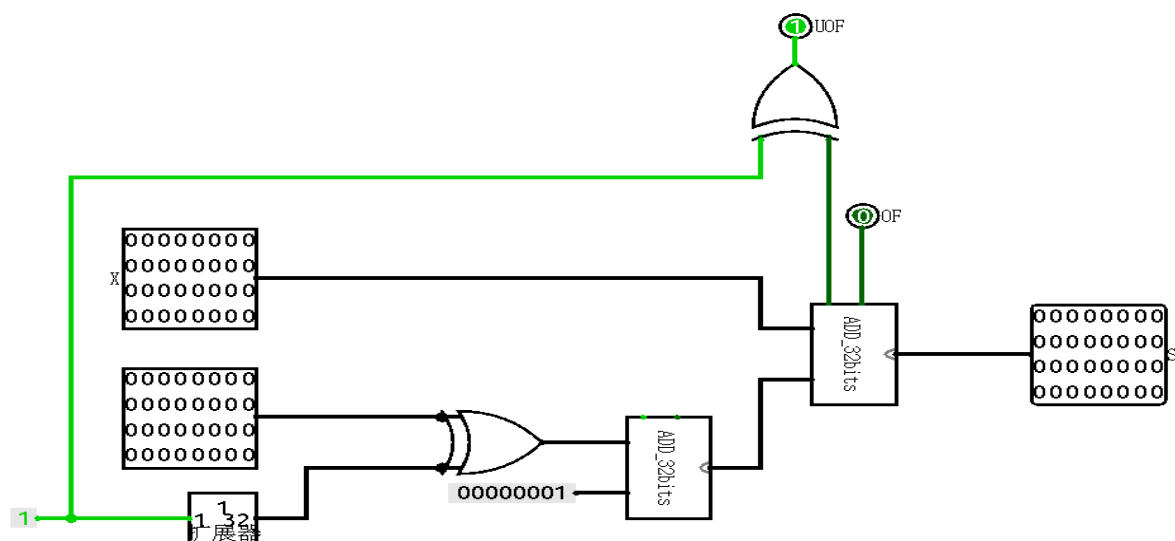
## (2) 构造 32 位先行进位的加法器、减法器

依照前面的设计,先用 4 位先行进位的加法器实现 16 位先行进位的加法器,在用 16 位先行进位的加法器构成 32 为先行进位的加法器,电路图如下:



**图 1.8 32 位先行进位加法器电路图**

通过 32 位加法器来实现减法器，实现减法时，第二个操作数取反加一操作后作为加法器的第二个操作数，得到减法的结果，同时设置溢出标志位，电路图如下：



**图 1.9 32 位减法器电路图**

## (3) 构造 ALU 电路

依照前面的设计，通过 logisim 现有的器件以及前面的 32 位加法器、32 位减法器，实现 alu 的各种功能和操作，电路图如下：

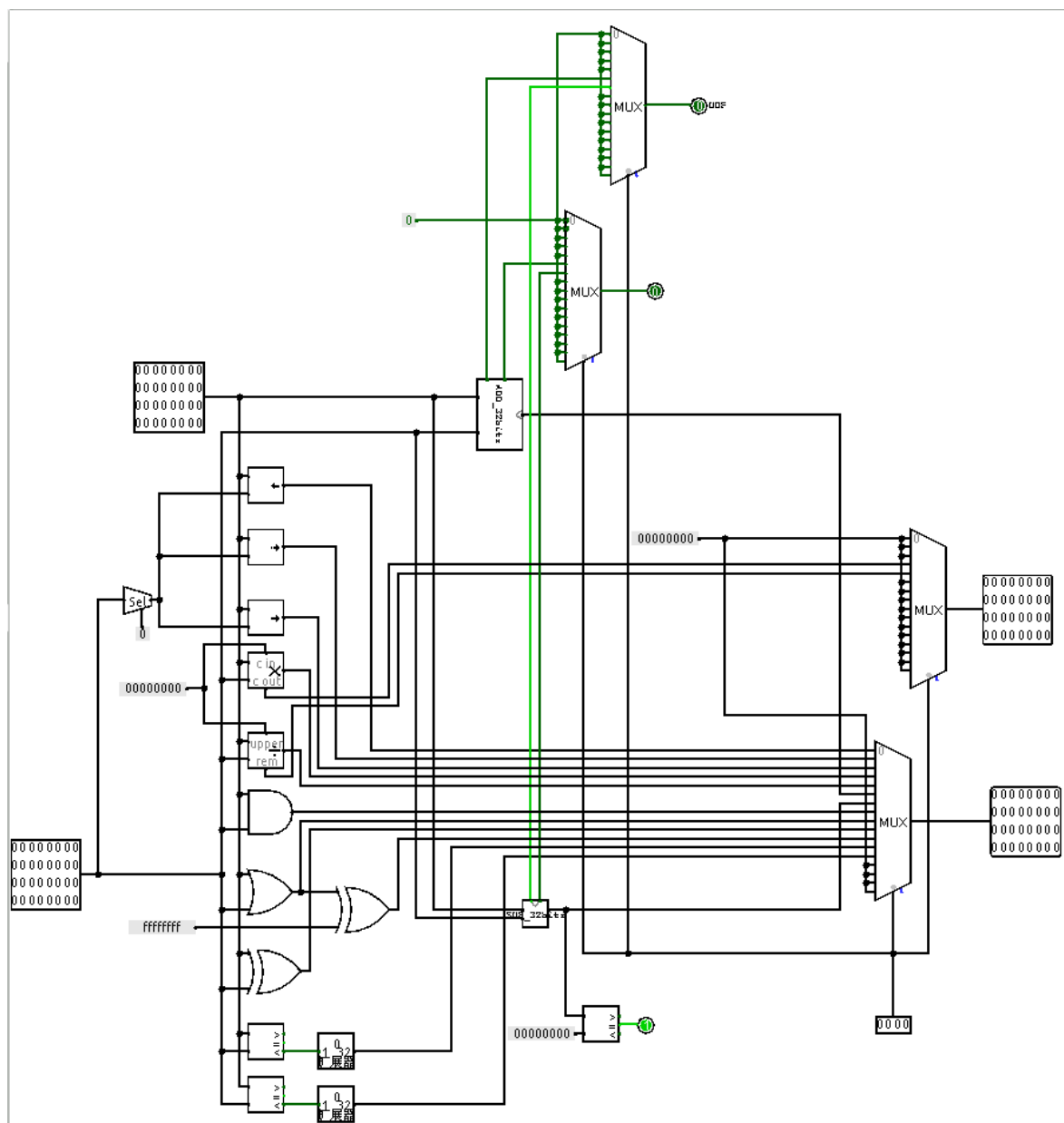


图 1.10 32 位 ALU 电路图

## (4) 测试

编制数据进行测试

## 1.4 故障与调试

### 1.4.1 接口处数据传输问题

**故障现象：**CIA74181 电路不能正常工作。

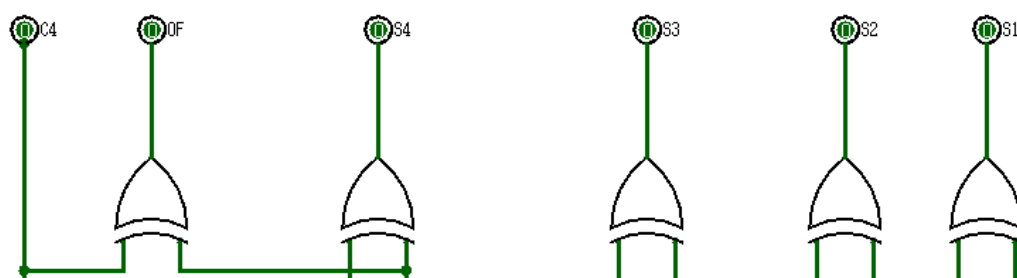


图 1.11 CLA74182 部分电路图

**原因分析：**如图 1.11，为正确的电路，此处为计算每一位的加法结果，根据前面的公式，此处应该用疑惑操作，但是之前实验时，错误的选择了或门，导致  $x=1$ ,  $y=0$ ,  $c=1$  的情况计算错误。

**解决方案：**将错误的或门改为正确的异或门，修改电路为正确电路。

### 1.4.2 输入某些 alu\_op 时，电路出现错误问题

**故障现象：**部分 alu\_op 的功能能够正常实现，比如移位、加减比较等操作，但是对于加减法等操作，当输入对应的 alu\_op 时，电路中出现错误信号。

**原因分析：**加减法的操作通过前面构造的 32 位加法器和 32 位减法器实现，32 位加法器中用到 2 片 CLA74182 电路和 8 片 CLA74181 电路，检查这部分电路，发现输入为某些值时，电路出现错误，检查对应错误电路，发现封装后的 CLA741812 电路在构成 32 位加法器时，有两个引脚短路，所以当输入某些值时，电路出现错误。

**解决方案：**修改短路的引脚，重新测试。

### 1.4.3 故障 2

## 1.5 测试与分析

溢出测试用例见表 1.4。

# 华中科技大学课程实验报告

表 1.4 溢出信号测试用例

#	A	B	F	运算	OF	UOF
1	0x1	0x1	0x2	加	○	○
2	0x7fffffe	0x1	0x7fffff	加	○	○
3	0x7ffffff	0x1	0x80000000	加	●	○
4	0xffffffff	0x800000000	0x7fffff	加	●	●
5	0xffffffff	0x800000000	0x7fffff	减	●	○
6	0xffffffff	0x000000000	0xffffffff	减	○	●

测试结果入上表，对有符号加减法，无符号加减法进行测试，比较实验结果、实验 OF 标志位、UOF 标志位和预期结果，一致，则测试通过。

移位操作测试用例见 1.4

表 1.4 移位操作测试用例

#	A	B	F	运算
2	0xffffffff	0x1	0xffffffe	逻辑左移
3	0xffffffff	0x1	0xeffffff	逻辑右移
4	0xffffffff	0x1	0xfffffff	算数右移
6	0xeffffff	0x000000001	0x3fffffff	算数右移

测试结果如上表，对逻辑左移、算数左移、算数右移做了测试，发现实验结果和预期结果一致。测试通过。

## 2 存储器实验

### 2.1 设计要求

1. 利用 logisim 平台构建一个 MIPS 寄存器组, 内部包含 32 个 32 位寄存器, 其具体功能如下, 具体封装文件为 regfile.circ。

表 2.1 芯片引脚与功能描述

引脚	输入/输出	位宽	功能描述
R1#	输入	5	读寄存器 1 编号
R2#	输入	5	读寄存器 2 编号
W#	输入	5	写入寄存器编号
Din	输入	32	写入数据
WE	输入	1	写入使能信号, 为 1 时, CLK 上跳沿将 Din 数据写入 W#寄存器
CLK	输入	1	时钟信号, 上跳沿有效
R1	输出	32	R1#寄存器的值
R2	输出	32	R2#寄存器的值
\$s0	输出	32	编号为 16 的寄存器的值
\$s1	输出	32	编号为 17 的寄存器的值
\$s2	输出	32	编号为 18 的寄存器的值
\$ra	输出	32	编号为 31 的寄存器的值

2. 利用实验一封装好的运算器, 以及 RAM 模块, 封装好的 MIPS 寄存器文件, 计数器等 logisim 模块构建一个自动运算电路, 该电路由时钟驱动, 可自动完成 RAM 模块 (32\*32 位) 0-31 号单元的累加, 并将累加的中间结果回存到同一 RAM 模块 32-63 号单元。同时运算器结果直接用 16 进制数码管显示。



## 2.2 方案设计

### 2.2.1 寄存器组设计

寄存器组有 32 个寄存器，编号 0~31，零号寄存器值应该恒零，所以 0 号寄存器只能读，不能写，其他寄存器可读可写，查看寄存器器件的描述，一个使能端，为 0 时忽略时钟输入，一个写入端，始终触发时更新寄存器的值，则可以通过对寄存器使能端的 0/1 状态来控制是否将数据写入到这个寄存器，一共有 32 个寄存器，是否写入用 0/1 控制，0 不写入，1 写入，则写入寄存器通过解复用器来实现，一个输入为选择的寄存器 0~31，一个输入是 0/1, 0 表示写入，1 表示不写入，同时 0 号寄存器不连接，则始终无法写入 0 号寄存器，且 0 号寄存器始终置 0；寄存器还有一个输出端口，输出当前寄存器的值，通过数据选择器来实现，数据选择器两个输入，一个是数据来源，有 32 个，来自 32 个寄存器，选择信号有 5 位，可以选择 32 个数据。则寄存器组输入包括写信号，要写的寄存器编号，要读的寄存器编号，以及对应读的寄存器的值。

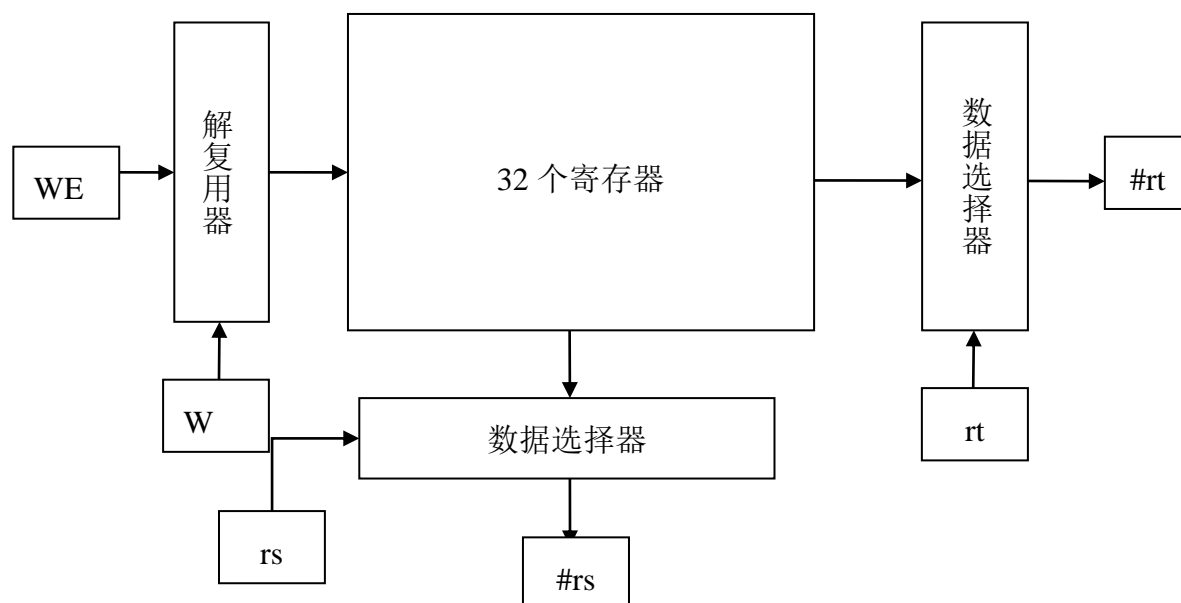


图 2.1 寄存器组设计结构图

## 2.2.2 自动累加器设计

内存有 64 个单元,前面 32 个单元储存有初始数据,后 32 个单元初始数据为空,用来存储计算的中间结果,寄存器组有 31 个可写入位,而计算的中间结果刚好 31 个,则设计为 1~31 号寄存器储存 31 次计算的中间结果,然后再将 32 个寄存器的值(包括 0 号寄存器的 0)写入到存储器的后 32 个单元。

前 32 个时钟要进行读内存操作和计算,并将结果写入到寄存器组,后 32 个时钟将寄存器的值写入到内存,通过计数器和比较器来实现读写信号的控制。

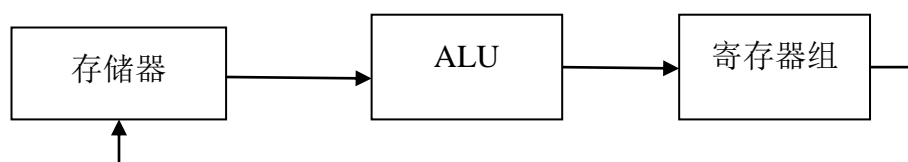


图 2.2 自动累加器电路结构图

因为第一次计算要把第一个数据和第二个数据都准备好,所以增加一个寄存器的使用,第一个时钟来时,将第一个数取出来放在寄存器,当第二个时钟来时,取出第二个数,然后将两个数送 alu 的两个操作数,完成第一次的加法,中间通过比较器和计数器来完成取数据和送数据的操作。

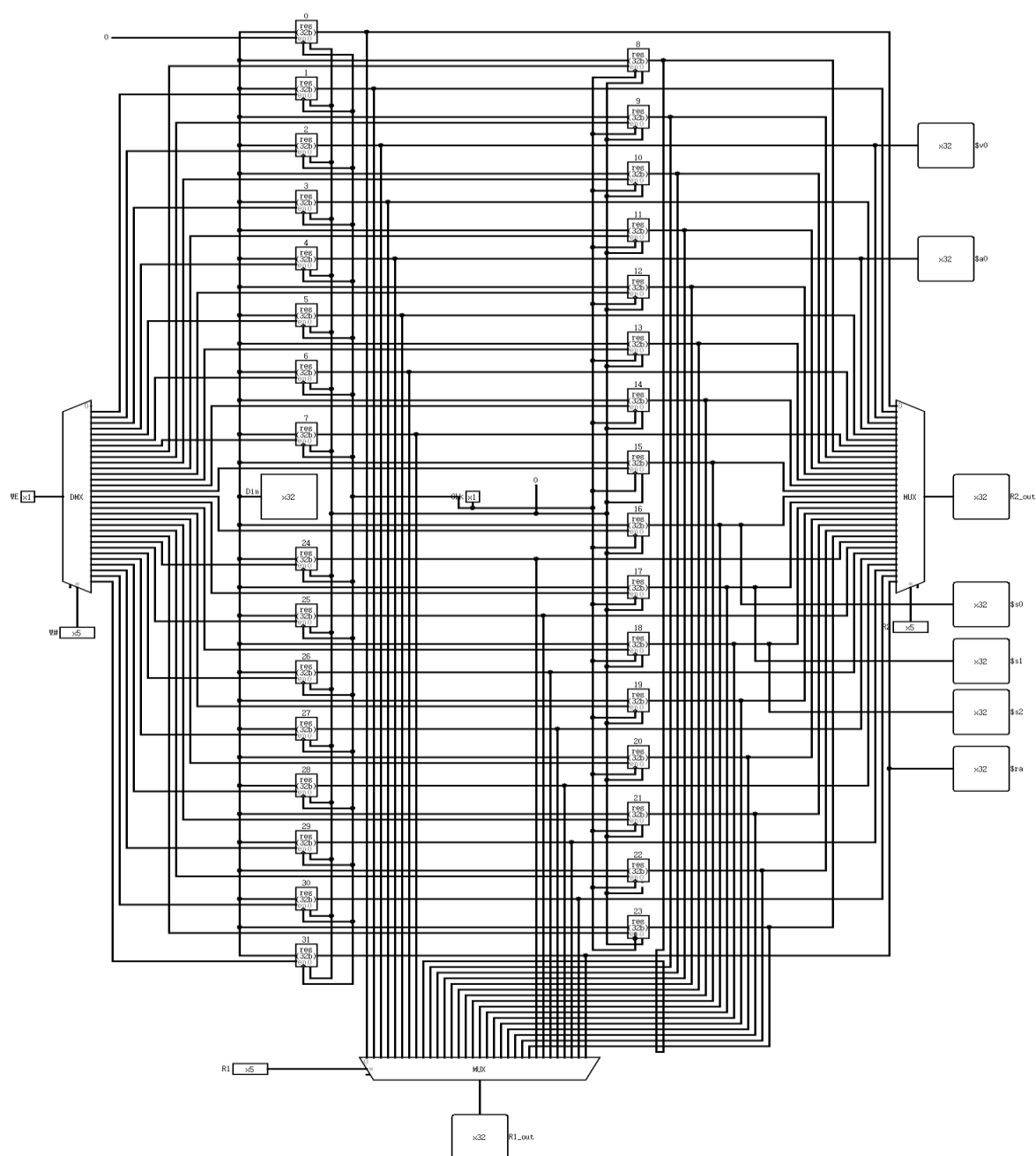
因为 0 号寄存器不可写,所以写寄存器的计数器从 1 开始,读寄存器的计数器从 0 开始。

为了完成循环计算,在经过 64 个时钟后,读寄存器的计数器第一个改为写寄存器的计数器,因为此时对寄存器的计数器还是上一个状态,并不是 0,下一个时钟来后,读寄存器的计数器重新更新为正确值。从而实现循环计数。

## 2.3 实验步骤

### (1) MIPS 寄存器组设计

按照前面的设计,用 32 个寄存器,一个解复用器,2 个数据选择器完成寄存器组电路的链接,寄存器组电路如下:



**图 2.2 mips 寄存器组电路图**

## (2) 构建自动累加器的电路图

利用实验一的 alu 和前面的 mips 寄存器组，以及 logisim 现有的 RAM 器件、比较器、计数器等器件，依照前面的设计，完成电路图如下：

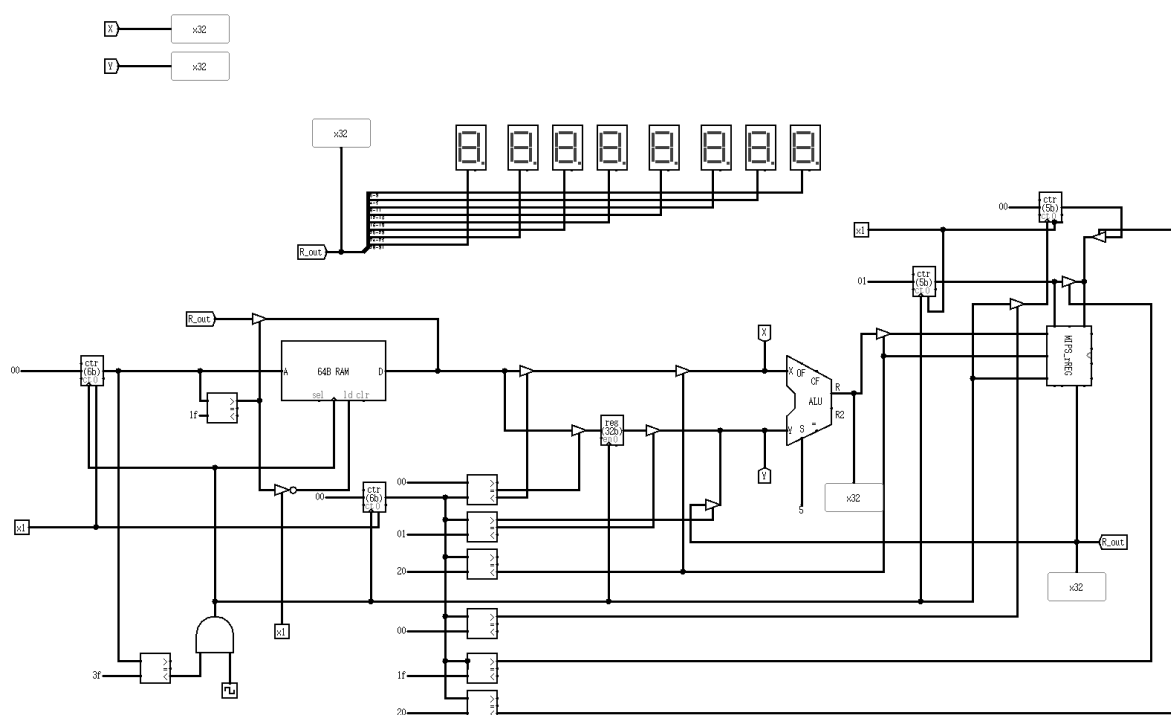


图 2.3 自动累加器电路图

## 2.4 故障与调试

### 2.4.1 寄存器值更新问题

**故障现象：**寄存器的值没有正确的更新，在完成写入寄存器操作时，预期时钟来时，寄存器没有及时更新，总是慢半个时钟。

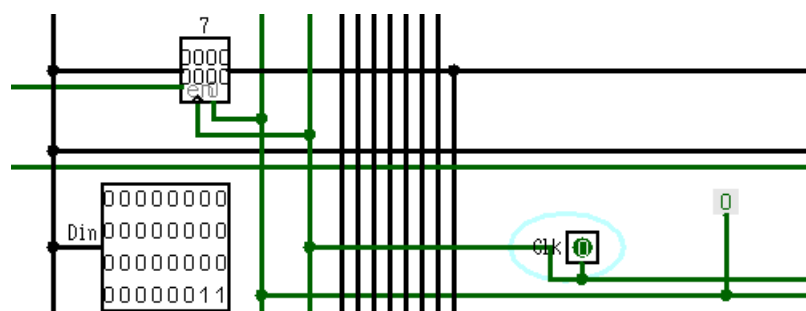


图 2.1 寄存器连接图

**原因分析：**如图 2.1，寄存器的链接如图，经过检查，寄存器的连接电路没有问题，通过检查寄存器，发现寄存器的触发方式有 4 种，当前使用的下降沿出触发，则导致每次寄存器刷新的时间是在时钟的下降沿，造成一定延迟。

**解决方案：**修改寄存器的触发条件，修改为上升沿触发，达到预期的效果。

## 2.4.2 故障 2

**故障现象：**第一次计数完成后，进行第二次计数时，计算结果不正确。

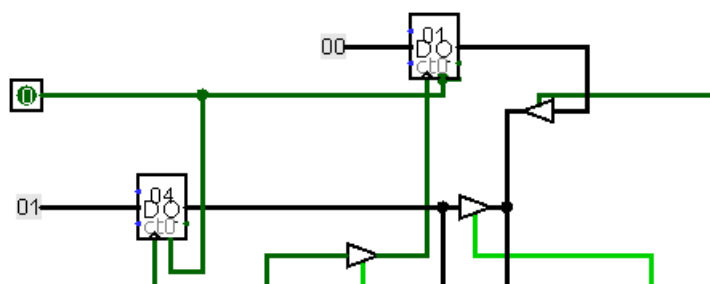


图 2.3 寄存器读写控制连接图

**原因分析：**如图 2.3，是对寄存器读写控制的连接图，开始都是从 0 开始，并且当重新计数时，没有对读写寄存器进行控制，则重新计数时，读寄存器的计数器还是 0x1f，则导致独处的结果不正确，从而使计算结果不正确。

**解决方案：**将电路改为如图所示，当计数器 0x3f 时，读寄存器的计数器改为写寄存器的计数器值，这样才能保证读到正确的寄存器值。

## 2.4.3 无法停止问题

**故障现象：**为了方便实验检查，需要停在计算结果的时刻，但是试验中发现无法停止。

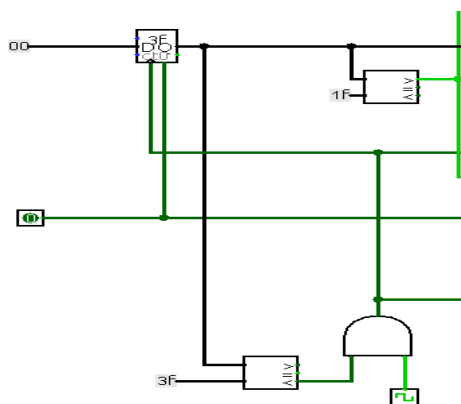


图 2.4 终止计算控制电路连接图

**原因分析：**如图 2.4，是修改后的计算终止控制电路图，没修改之前，通过控

# 华中科技大学课程实验报告

制计数器超出计数保留当前值得方法来控制结束，但是发现计数会多一次，因为时钟是上升沿触发，所以计数多了一次。

**解决方案：**如图 2.4 所示，通过一个比较器，计数达到 64 次时，就屏蔽时钟信号，从而使计数次数正确。

## 2.5 测试与分析

mips 寄存器组测试结果表 2.1:

表 2.1 寄存器组测试用例

WE	#WE	RS	读结果	写结果
1	0	-	-	0 号寄存器不变
1	1	-	-	1 号寄存器改变
0	-	2	读出 2 号寄存器的值	

如表 2.1，测试了读寄存器和写寄存器的操作，实验结果和预期一致，测试通过。

自动累加器测试，首先在 ram 中写入镜像文件，内容是 32 值，本次测试中值为 1~32，后 32 个单元为 0。开始时钟后，电路中开始运算，同时数码管上一次显示运算的中间结果，通过对比，中间结果正确，最后的停止时的运算结果为 0x210，与预期结果一致，则测试通过。

## 3 CPU 实验

### 3.1 设计要求

利用实验 1、2 中构建的运算器，寄存器文件等部件以及 logsim 中其他功能部件构建一个 32 位 MIPS CPU 处理器，该处理器应支持下表中所述指令，具体指令功能参见附件中的 MIPS 标准文档。最终设计完成的 CPU 能运行教师提供的标准测试程序，程序存储在 logsim ROM 模块中（指令存储器、数据存储器分开）

表 3.1 指令格式

#	指令	格式	备注
1	Add	add \$rd, \$rs, \$rt	
2	Add Immediate	addi \$rt, \$rs, immediate	
3	Add Immediate Unsigned	addiu \$rt, \$rs, immediate	
4	Add Unsigned	addu \$rd, \$rs, \$rt	
5	And	and \$rd, \$rs, \$rt	
6	And Immediate	andi \$rt, \$rs, immediate	
7	Shift Left Logical	sll \$rd, \$rt, shamt	
8	Shift Right Arithmetic	sra \$rd, \$rt, shamt	
9	Shift Right Logical	srl \$rd, \$rt, shamt	
10	Sub	sub \$rd, \$rs, \$rt	
11	Or	or \$rd, \$rs, \$rt	
12	Or Immediate	ori \$rt, \$rs, immediate	
13	Nor	nor \$rd, \$rs, \$rt	
14	Load Word	lw \$rt, offset(\$rs)	
15	Store Word	sw \$rt, offset(\$rs)	
16	Branch on Equal	beq \$rs, \$rt, label	
17	Branch on Not Equal	bne \$rs, \$rt, label	

指令功能及指令格式  
参考 MIPS32 指令集

下一页续表：

# 华中科技大学课程实验报告

续上一页表:

#	指令	格式	备注
18	Set Less Than	slt \$rd, \$rs, \$rt	
19	Set Less Than Immediate	slti \$rt, \$rs, immediate	
20	Set Less Than Unsigned	sltu \$rd, \$rs, \$rt	
21	Jump	j label	
22	Jump and Link	jal label	
23	Jump Register	jr \$rs	
24	syscall (display or exit)	syscall	If \$v0==10 halt(停机指令) else 数码管显示\$a0 值

## 3.2 方案设计

### 3.2.1 CPU 总体结构设计

MIPS 指令集包含 R 型指令、I 型指令、J 型指令三种指令，首先需要一个 PC 计数器，用来控制指令的读取，需要指令存储器，存储指令，需要前面实验的寄存器组，作为指令中的操作数，用于保存数据，状态，地址等信息，需要数据存储器，存储数据，需要运算器 ALU 实现数据的处理，需要实现数据读写，还需要一个控制器，用于产生各类控制信号。则总体结构如图：

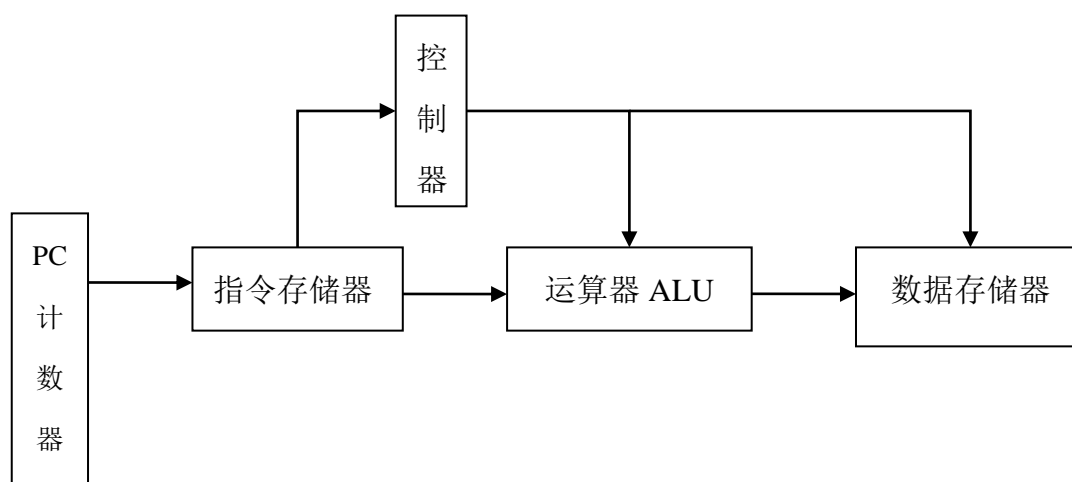


图 3.1 cpu 总体设计结构图



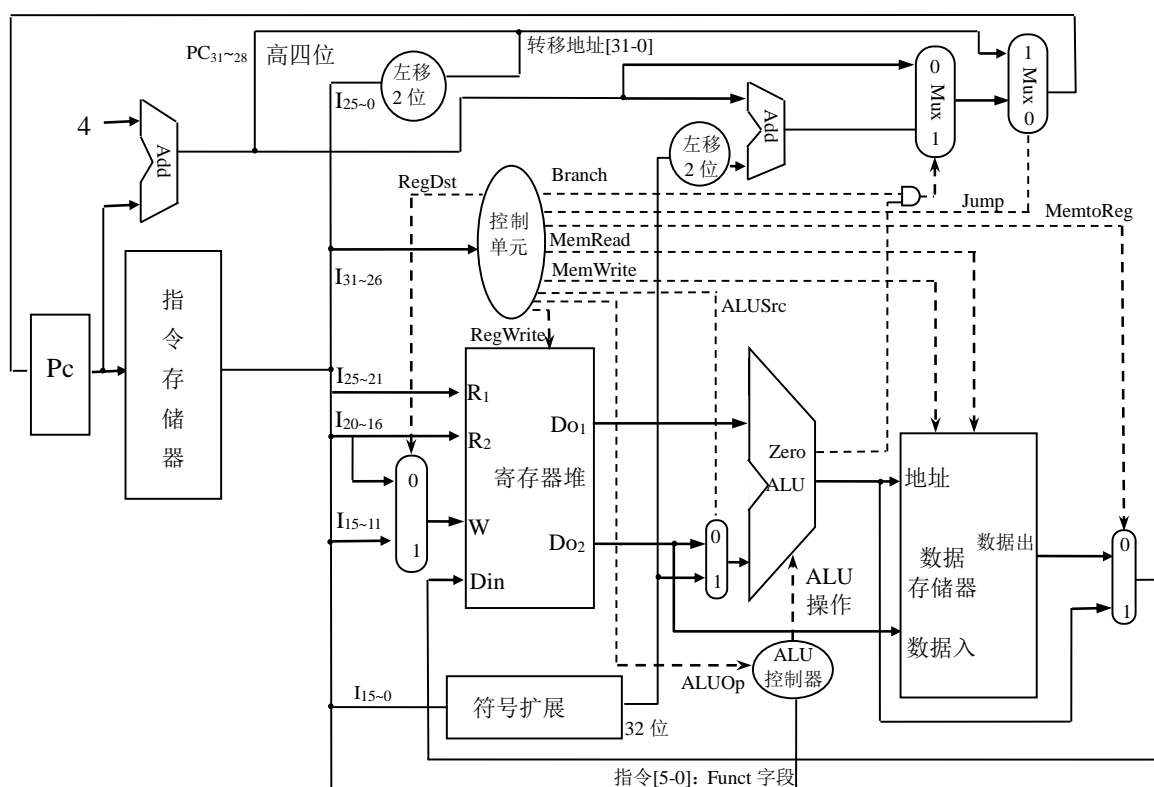


图 3.2 cpu 数据通路高层视图

## 3.2.2 控制器设计

### 1. alu 功能码信号产生设计

alu 功能码由 4 位表示, 则对于 mips 指令集中的所有指令, 根据指令的 op 字段, funct 字段, shamt 字段出 op4 位数值的真值表, 根据真值表写出 op 对应位的表达式, 利用 logisim 平台的电路产生功能产生 alu\_op 的电路。

### 2. 跳转指令 jr, jal, jl 信号

查 mips 指令集表, 找到三条跳转指令, 根据指令字段生成 jr, jal, jl 的信号, jl 和 jal 是先 PC+4, 然后加上指令的跳转字段, 所以先对 26 位数据进行扩展到 32 位, 然后和 PC+4 的结果相加, 作为新的 PC 值, jr 是直接读出对应寄存器的值, 作为新的 PC。

### 3. BEQ 和 BNQ 指令信号

类似跳转指令的处理。

### 4. 访存指令 sw, lw 信号

查 mips 指令集表，找到所有需要访存的指令，生成对应的电路，输出为访存信号，信号作为存储器读和写的输入，根据时钟更新内存。

## 5.写寄存器 we 信号

查找需要写寄存器的指令，生成对应的写寄存器信号，作为寄存器组写使能端的输入，寄存器根据时钟来更新寄存器的值，实现写操作。

## 6.SYSCALL 信号

本次实验中主要需要通过 syscall 信号来显示 4 号寄存器的值，以及 syscall\_b 信号来终止 cpu 的运行，通过查询 SYSCALL 指令来写出 syscall 信号的表达式，生成电路，其中 syscall 信号作为输出 4 号寄存器的使能端输入，syscall\_b 信号作为屏蔽时钟输入的的信号，实现终止 cpu。

### 3.2.3 取指令数据通路设计

其中 PC 由时钟驱动，每个时钟自动完成取值以及  $PC=PC+4$  的功能，此处指令存储器选用 10 位地址线，32 位数据线的 ROM 部件，MIPS32s 位地址为字节地址，ROM 地址线宽度有限，将 CPU 32 位地址高位部分和字节偏移部分直接屏蔽，数据存储器也参照类似方法处理。

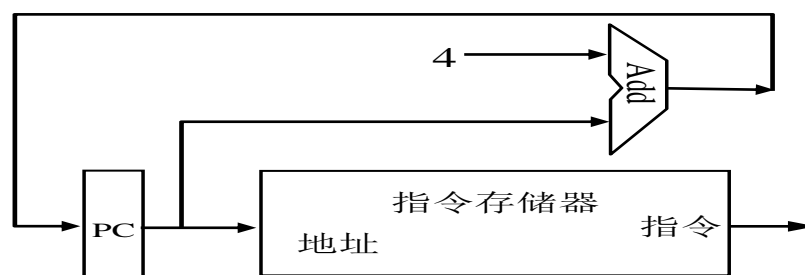


图 3.3 取指令数据通路结构图

### 3.2.4 R 型指令设计

根据 PC 得到指令地址，然后取出指令，对指令译码，生成 ALU 的功能码，操作数寄存器的编号，并和寄存器组相连，寄存器组的输出作为 alu 的输入，实现两

个操作数的运算，然后将运算结果写回寄存器。寄存器通过时钟来更新寄存器的值。

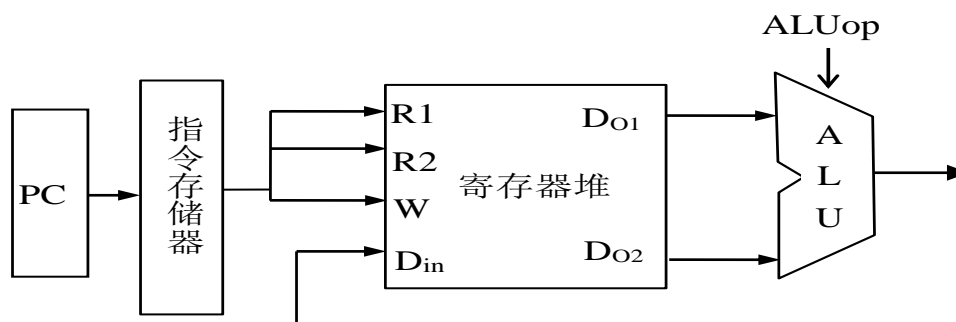


图 3.4 R 型指令数据通路结构图

## 3.2.5 I 型指令设计

I 型指令有访存操作，所以需要将控制器产生的访存信号接入存储器，然后因为 alu 的操作数来源发生变化，通过一个数据选择器实现对操作数来源的选择。对于写入寄存器的数据来源有多个时，通过多路选择器来选择对应的数据来源。数据通路结构图如图：

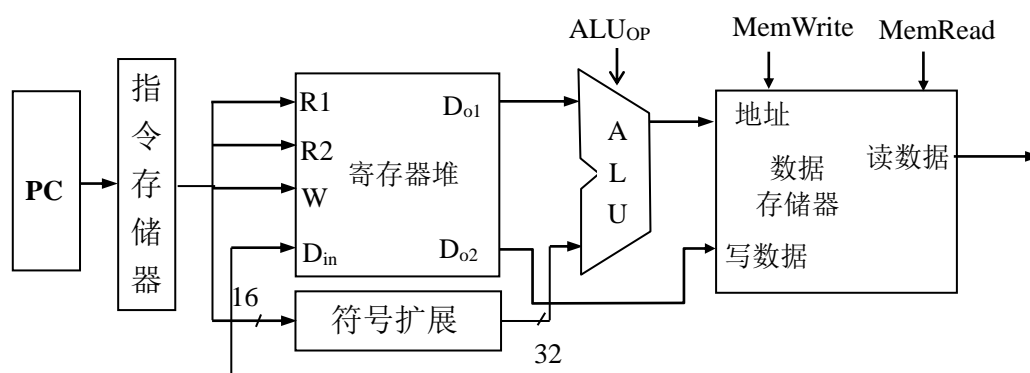


图 3.5 I 型指令数据通路结构图

## 3.2.6 J 型指令设计

j 型指令实现跳转操作，有无条件跳转，条件转移，寄存器跳转，条件转移和无条件转移时，先实现  $PC+4$ ，然后跳转至指令中的跳转字段 0 扩展至 32 位，与 PC 相加，作为新的 PC 值，寄存器跳转直接取出对应寄存器的值作为新的 PC。对于 PC 值的来源有多个时，通过多路选择器来选择对应的数据来源。

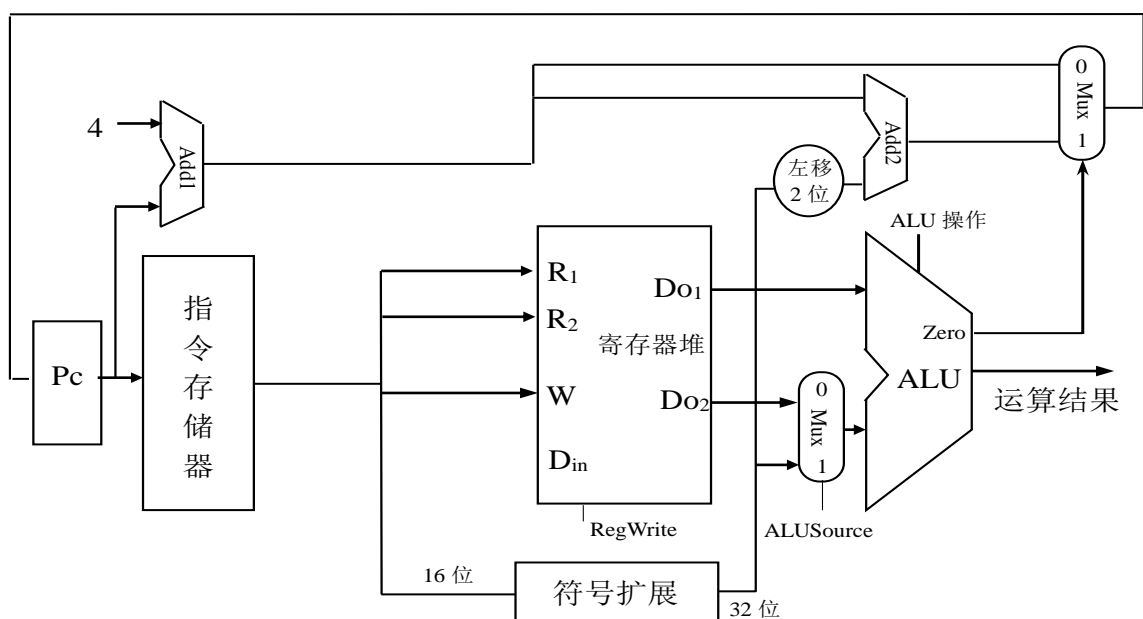


图 3.6 J 型指令条件转移数据通路结构图

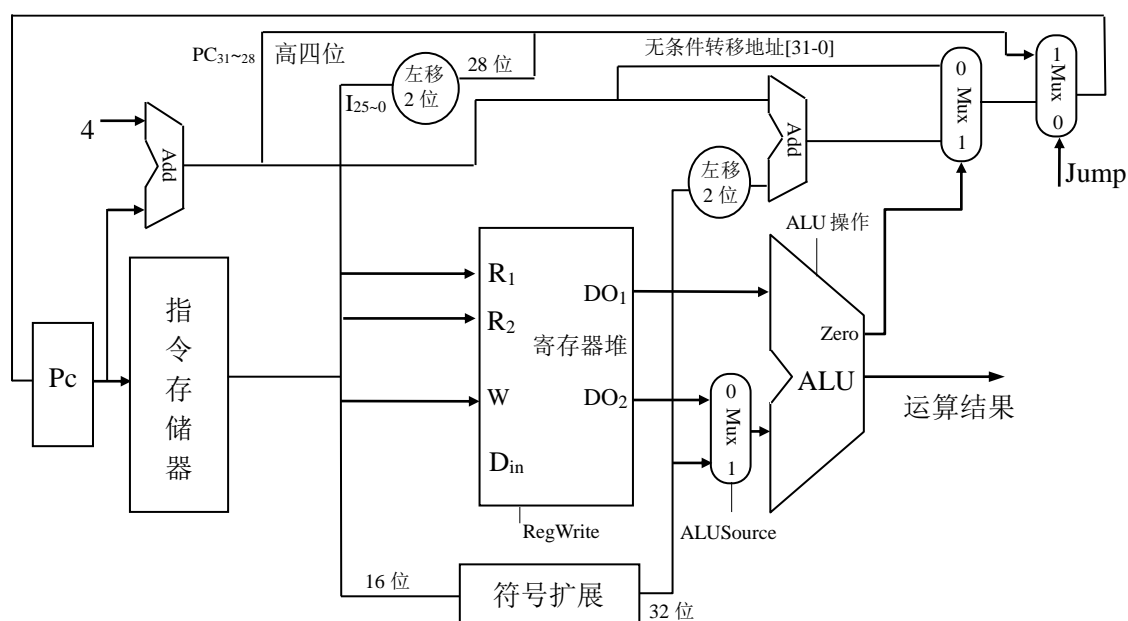
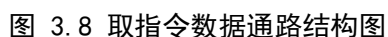


图 3.7 J 型指令无条件转移数据通路结构图

## 3.3 实验步骤

### (1) 取指令数据通路实现

依据前面的设计，利用 logisim 平台现有的器件，实现曲智林的数据通路，电路结构如下：

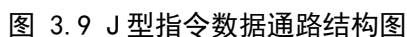


依据前面的设计,构造真值表,写出对应控制信号的表达式,利用 logisim 平台的生成电路功能,生成对应控制信号的表达式。

依据前面的设计，利用 logisim 现有的逻辑器件，构造 R 型指令的数据通路。

依据前面的设计，利用 logisim 现有的逻辑器件，构造 I 型指令的数据通路。

依据前面的设计，利用 logisim 现有的逻辑器件，构造 J 型指令的数据通路。



总的电路图如下所示:

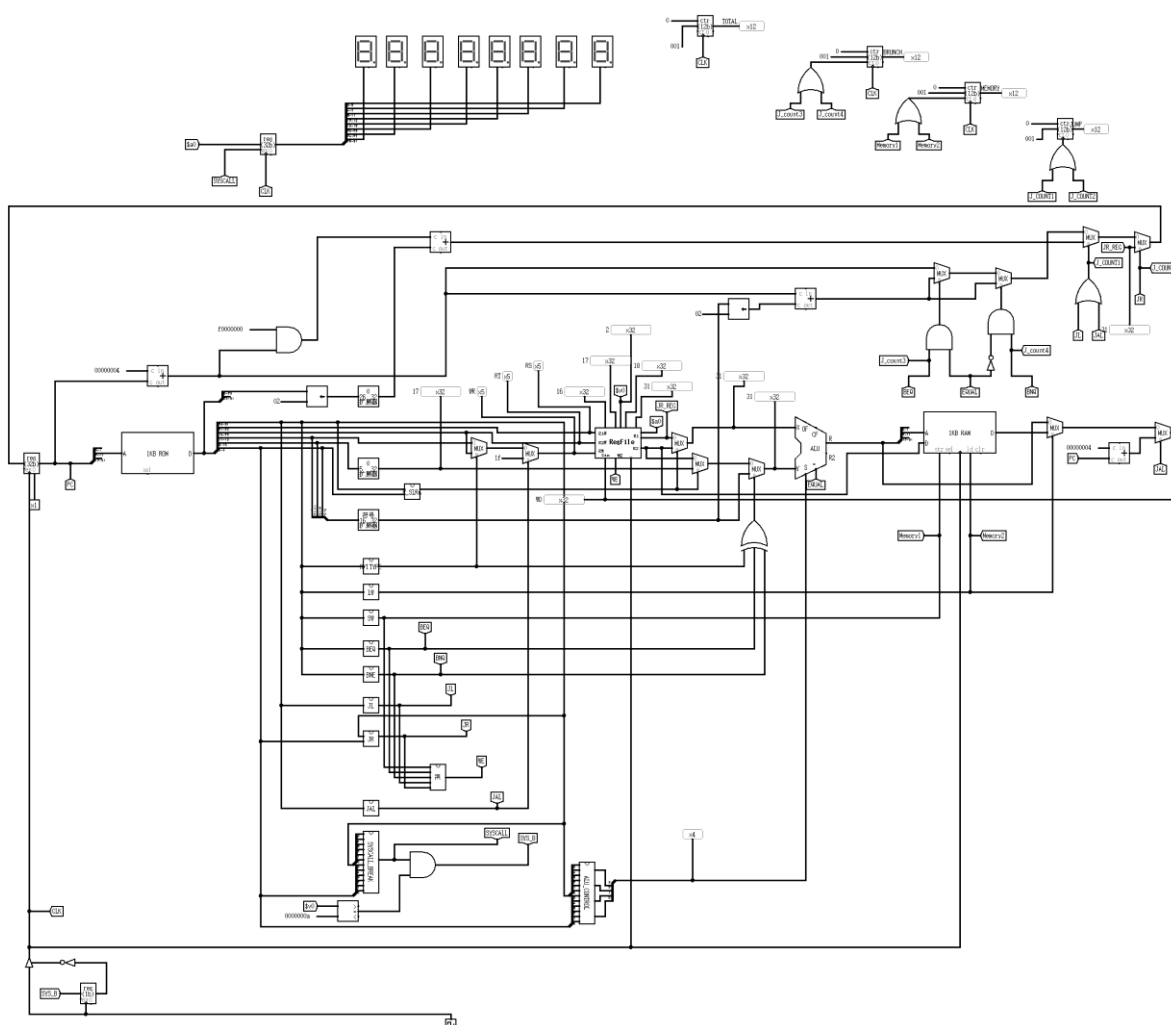


图 3.10 cpu 总的电路结构图

## 3.4 故障与调试

### 3.4.1 指令执行顺序问题

**故障现象：**执行指令的顺序不对，没有按照正确的指令执行顺序执行，每次有跳转指令时，都无法跳转到正确的地址。

**原因分析：**检查电路，发现没有实现 jr 指令，即不能正确的返回函数调用之前的地址继续执行，使得指令执行顺序出错。

**解决方案：**添加 jr 指令的控制信号，然后添加 jr 指令的数据通路。

### 3.4.2 或指令不能正确执行问题

# 华中科技大学课程实验报告

**故障现象：**在调试过程中，发现或指令没有正常进行，即对于 0xffff0000 和 0x0000000f 没有得到正确的或擦偶偶结果。

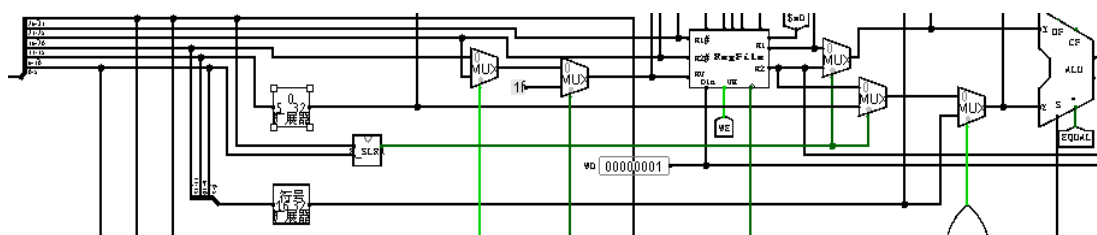


图 3.10 扩展问题截图

**原因分析：**检查电路，对于 ori 指令，第二个操作数来自于立即数的符号扩展，这里采用符号扩展是因为执行加法和减法时，需要符号扩展，但是 ori 指令执行时，应该是 0 扩展，智力出现错误的扩展方式。

**解决方案：**更改扩展方式，增加一个扩展器件，做 0 扩展。

## 3.5 测试与分析

### 1. 跳转指令测试

编写汇编程序段，利用 mars 生成指令，加载到指令寄存器，观察指令执行结果。

跳转指令测试样例表：

指令	结果
j jmp_next1	正确
jr \$31	正确
jal jmp_count	正确

跳转指令正确执行，测试通过。

### 2 数据相关操作测试

测试指令中有关数据操作的指令，包括加法，减法，乘除法，移位，与，或，非等操作，编写汇编程序段，利用 mars 生成指令，加载到指令寄存器，观察指令执行结果。

数据相关操作测试样例表：

# 华中科技大学课程实验报告

指令	结果
addi \$s1, \$0, 4	正确
addi \$s0, \$0, 0	正确
srl \$s1, \$s1, 2	正确
sll \$s1, \$s1, 2	正确
sra \$s1, \$s1, 3	正确
andi \$s0, \$s0, 15	正确
or \$s3, \$s3, \$s0	正确

经过测试，与数据有关的操作都可以正常执行，且结果正确，测试通过。

## 3.整体功能测试

测试用例：排序测试.asm,benchmark.asm，利用 mars 生成对应的指令，加载到指令寄存器，加上时钟信号，统计个指令的执行次数以及周期数，与预期结果比较，结果一致。

项目	预期结果	实验结果
周期数	1546	1546
分支指令	338	338
访存指令	272	272
跳转指令	38	38

经过测试，指令执行周期和次数与预期结果一致，则证明指令执行正确，测试通过。



## 4 总结与心得

### 4.1 实验总结

本次实验主要完成了如下几点工作：

#### 1) 完成方案总结

实验 1：设计 4 位先行进位加法器，设计 32 位先行进位加法器，设计 32 位减法器，设计了运算器 alu，实现了 alu 操作的 12 中操作。

实验 2：设计了具有 32 个寄存器的寄存器组，设计了据用循环自动累加的计算器，实现了累加器。

实验 3：设计了单周期 mipsCPU 取指令数据通路，设计了单周期 mipsCPU 型指令数据通路，设计了单周期 mipsCPU 型指令数据通路，设计了单周期 mipsCPU 型指令数据通路，实现了能执行 24 条指令的单周期 mipsCPU。

#### 2) 功能总结

实验 1：完成 32 位先行进位加法器，可以对两个 32 位数据进行加法计算，完成 alu，可以对应 12 个功能码进行操作，包括移位操作，加、减、乘、除操作，与、或、异或、或非操作，比较操作...

实验 2：完成具有 32 个寄存器的寄存器组，可以对其中的寄存器读写。完成具有循环自动累加功能的累加器，可以自动将 32 个数据求和并保存。

实验 3：完成对应 24 条指令的 mipsCPU，可以执行 24 条指令，对于机器代码，能够正确执行并得到正确结果。

### 4.2 实验心得

- 1) 实验 1 中，对于先行进位有了更加充分的理解，1 位全加器逻辑门的延时是  $3T$ ，则  $n$  个串联就是  $3nT$ ，采用先行进位方式，可以降低逻辑门的延时，降低开销，提高效率。
- 2) 实验 2 中，通过构造自动累加器，对数据通路有了简单的了解。对寄存器组有了更好的了解。

# 华中科技大学课程实验报告

---

- 3) 实验 3 中, 通过对 CPU 的设计, 更好的了解了指令执行的数据通路, 对 CPU 的内部结构以及寄存器的使用和功能有了更好的了解, 对指令的执行过程有了更深刻的理解。
- 4) 建议: 希望课内实验实验可以长一点, 做实验时可以有充分的设计和思考时间, 可以更好的理解。本学期实验较多, 课外实验占用了大量时间 (所有的实验)。

## 参考文献

- [1] DAVID A. PATTERSON(美). 计算机组成与设计硬件/软件接口(原书第4版). 北京: 机械工业出版社.
- [2] David Money Harris(美). 数字设计和计算机体系结构(第二版). 机械工业出版社
- [3] 秦磊华, 吴非, 莫正坤. 计算机组成原理. 北京: 清华大学出版社, 2011 年.
- [4] 袁春风编著. 计算机组成与系统结构. 北京: 清华大学出版社, 2011 年.
- [5] 张晨曦, 王志英. 计算机系统结构. 高等教育出版社, 2008 年.

• 指导教师评定意见 •

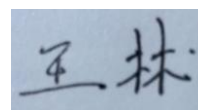
---

### 一、原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字：



### 二、对课程实验的学术评语（教师填写）

### 三、对课程设计的评分（教师填写）

评分项目 (分值)	报告撰写 (30 分)	课设过程 (70 分)	最终评定 (100 分)
得分	1	23	24

指导教师签字：\_\_\_\_\_ 2017-01-14