华中科技大学

2021

计算机组成原理　　课程设计报告

题　　目：	5 段流水 CPU 设计
专　　业：	计算机科学与技术
班　　级：	CS1804
学　　号：	U201814604
姓　　名：	黄俊淇
电　　话：	13977133562
邮　　件：	2474268727@qq.com

目 录

1 课程设计概述

1.1 课设目的

计算机组成原理是计算机专业的核心基础课。该课程力图以“培养学生现代计算机系统设计能力”为目标，贯彻“强调软/硬件关联与协同、以 CPU 设计为核心/层次化系统设计的组织思路，有效地增强对学生的计算机系统设计与实现能力的培养”。课程设计是完成该课程并进行了多个单元实验后，综合利用所学的理论知识，并结合在单元实验中所积累的计算机部件设计和调试方法，设计出一台具有一定规模的指令系统的简单计算机系统。所设计的系统能在 LOGISIM 仿真平台和 FPGA 实验平台上正确运行，通过检查程序结果的正确性来判断所设计计算机系统正确性。

课程设计属于设计型实验，不仅锻炼学生简单计算机系统的设计能力，而且通过进行中央处理器底层电路的实现、故障分析与定位、系统调试等环节的综合锻炼，进一步提高学生分析和解决问题的能力。

1.2 设计任务

本课程设计的总体目标是利用 FPGA 以及相关外围器件，设计五段流水 CPU，要求所设计的流水 CPU 系统能支持自动和单步运行方式，能正确地执行存放在主存中的程序的功能，对主要的数据流和控制流通过 LED、数码管等适时的进行显示，方便监控和调试。尽可能利用 EDA 软件或仿真软件对模型机系统中各部件进行仿真分析和功能验证。在学有余力的前提下，可进一步扩展相关功能。

1.3 设计要求

（1） 根据课程设计指导书的要求，制定出设计方案；

（2） 分析指令系统格式，指令系统功能。

（3） 根据指令系统构建基本功能部件，主要数据通路。

（4） 根据功能部件及数据通路连接，分析所需要的控制信号以及这些控制信号的有效形式；

（5） 设计出实现指令功能的硬布线控制器；

（6） 调试、数据分析、验收检查；

（7） 课程设计报告和总结。

1.4 技术指标

（8） 支持表 1.1 前 27 条基本 32 位 MIPS 指令；

（9） 支持教师指定的 4 条扩展指令；

（10）支持多级嵌套中断，利用中断触发扩展指令集测试程序；

（11）支持 5 段流水机制，可处理数据冒险，结构冒险，分支冒险；

（12）能运行由自己所设计的指令系统构成的一段测试程序，测试程序应能涵盖所有指令，程序执行功能正确。

（13）能运行教师提供的标准测试程序，并自动统计执行周期数

（14）能自动统计各类分支指令数目，如不同种类指令的条数、冒险冲突次数、插入气泡数目、load-use 冲突次数、动态分支预测流水线能自动统计预测成功与失败次数。

表 1.1 指令集

#	指令助记符	简单功能描述	备注
1	ADD	加法	指令格式参考 MIPS32 指令集，最终功能以 MARS 模拟器为准。
2	ADDI	立即数加	
3	ADDIU	无符号立即数加	
4	ADDU	无符号数加	
5	AND	与	
6	ANDI	立即数与	
7	SLL	逻辑左移	
8	SRA	算数右移	
9	SRL	逻辑右移	
10	SUb	减	
11	OR	或	
12	ORI	立即数或	

#	指令助记符	简单功能描述	备注
13	NOR	或非	
14	LW	加载字	
15	SW	存字	
16	BEQ	相等跳转	
17	BNE	不相等跳转	
18	SLT	小于置数	
19	STI	小于立即数置数	
20	SLTU	小于无符号数置数	
21	J	无条件转移	
22	JAL	转移并链接	
23	JR	转移到指定寄存器	If $v0==10 halt(停机指令)
24	SYSCALL	系统调用	else 数码管显示$a0 值
25	MFC0	访问 CP0	中断相关，可简化，选做
26	MTC0	访问 CP0	中断相关，可简化，选做
27	ERET	中断返回	异常返回，选做
28	SLLV	逻辑可变左移	
29	SRLV	逻辑可变右移	
30	LB	加载字节	
31	BGTZ	大于 0 转移	

2 总体方案设计

2.1 单周期 CPU 设计

本次我们采用的方案是硬布线控制，且采用了哈佛结构，即将指令存储器和数据存储器分开，用两个存储器表示。这个 CPU 支持表 1.1 所示 1-24 条基本指令以及 28-31 条差异化指令。设计过程中采用 logisim 来完成逻辑电路设计。

总体结构图如图 2.1 所示。（注意设计阶段只需要简单的原理示意图）

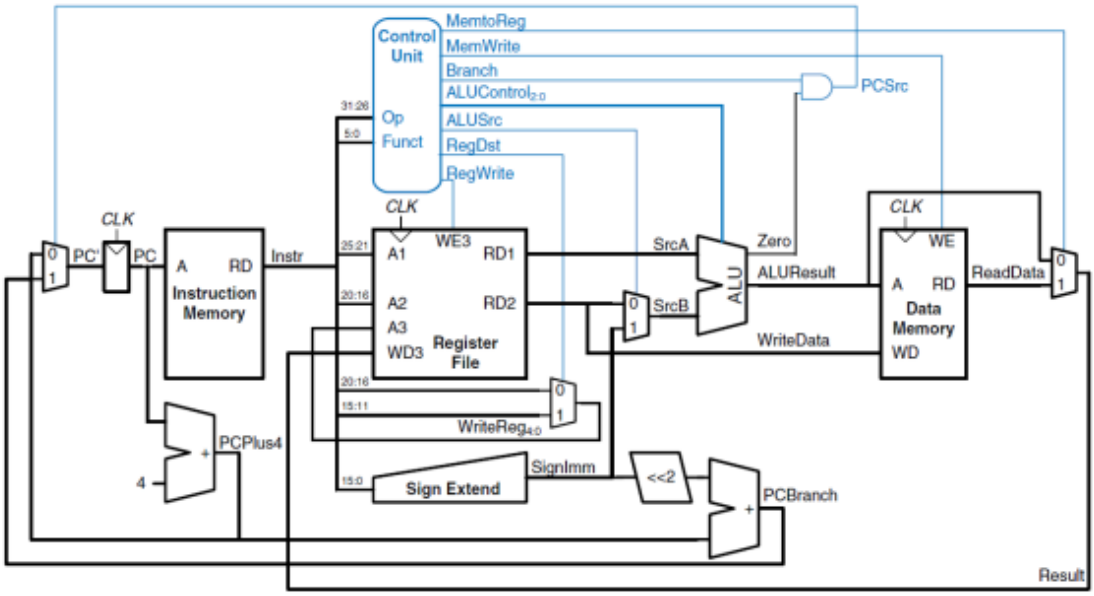


图 2.1 总体结构图

2.1.1 主要功能部件

该单周期 MIPS CPU 主要包括：程序计数器 PC，指令存储器 IM，运算器 ALU，寄存器堆 RF，数据存储器 DM。其中单个部件设计如下所述：

1. 程序计数器 PC

程序计数器 PC 是一个 32 位寄存器，存储下一条指令的地址。每个周期中，CPU 会从 PC 寄存器中取出访问指令的地址，在指令存储器取出相应的指令执行。

2. 指令存储器 IM

指令存储器 IM 用于存储程序的所有指令。每个周期中，根据 PC 取出的地址得到相应的指令交给硬布线控制器等元件执行。

3. 运算器

表 2.1 运算器规格

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 = $(X * Y)_{[63:32]}$ 无符号乘法
0100	4	Result = X/Y; Result2 = X%Y 无符号除法
0101	5	Result = X + Y (Set OF/UOF)
0110	6	Result = X - Y (Set OF/UOF)
0111	7	Result = X & Y 按位与
1000	8	Result = X | Y 按位或
1001	9	Result = X⊕Y 按位异或
1010	10	Result = ~(X |Y) 按位或非
1011	11	Result = (X < Y) ? 1 : 0 符号比较
1100	12	Result = (X < Y) ? 1 : 0 无符号比较

表 2.2 算术逻辑运算单元引脚与功能描述

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

引脚	输入/输出	位宽	功能描述
OF	输出	1	有符号加减溢出标记，其他操作为零
UOF	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

4. 寄存器堆 RF

寄存器堆 RF 是上跳沿触发的寄存器堆。

5. 数据存储器 DM

数据存储器 DM 用于存储程序的所有数据，可以存储运算结果等数据。

2.1.2 数据通路的设计

表 2.3 指令系统数据通路框架

指令	PC	IM	RF				ALU			DM	
			R1#	R2#	W#	Din	A	B	OP	Addr	Din
SLL	PC+4	PC	rs	rt	rd	ALU		R2	0		
SRA	PC+4	PC	rs	rt	rd	ALU		R2	1		
SRL	PC+4	PC	rs	rt	rd	ALU		R2	2		
ADD	PC+4	PC	rs	rt	rd	ALU	R1	R2	5		
ADDU	PC+4	PC	rs	rt	rd	ALU	R1	R2	5		
SUB	PC+4	PC	rs	rt	rd	ALU	R1	R2	6		
AND	PC+4	PC	rs	rt	rd	ALU	R1	R2	7		
OR	PC+4	PC	rs	rt	rd	ALU	R1	R2	8		
NOR	PC+4	PC	rs	rt	rd	ALU	R1	R2	10		
SLT	PC+4	PC	rs	rt	rd	ALU	R1	R2	11		
SLTU	PC+4	PC	rs	rt	rd	ALU	R1	R2	12		
JR	R1	PC	rs								
SYSCALL	PC+4	PC	2	4							

指令	PC	IM	RF				ALU			DM	
			R1#	R2#	W#	Din	A	B	OP	Addr	Din
J	(PC+4)[31:28]||IMM26||00	PC									
JAL	(PC+4)[31:28]||IMM26||0	PC			31	PC+4					
BEQ	PC+4/PC+4+IMM<<2	PC	rs	rt			R1	R2			
BNE	PC+4/PC+4+IMM<<2	PC	rs	rt			R1	R2			
ADDI	PC+4	PC	rs		rt	ALU	R1	IMM	5		
ANDI	PC+4	PC	rs		rt	ALU	R1	IMM	7		
ADDIU	PC+4	PC	rs		rt	ALU	R1	IMM	5		
SLTI	PC+4	PC	rs		rt	ALU	R1	IMM	11		
ORI	PC+4	PC	rs		rt	ALU	R1	IMM	8		
LW	PC+4	PC	rs		rt	Dout	R1	IMM	5	ALU	
SW	PC+4	PC	rs	rt			R1	IMM	5	ALU	R2
SLLV	PC+4	PC	rs	rt	rd	ALU	R1	R2	0		
SRLV	PC+4	PC	rs	rt	rd	ALU	R1	R2	2		
LB	PC+4	PC	rs		rt	Dout	R1	IMM	5	ALU	
BGTZ	PC+4/PC+4+IMM<<2	PC	rs						11		

2.1.3 控制器的设计

首先对于控制信号进行统计，包括各个主要部件所需要输入的控制信号，以及数据通路合并表中所示的具有多输入的主要部件需要进行输入选择的控制信号，并且对各个统计信号的各种取值情况进行定义，统计得到的控制信号以及说明如表 2.4。

表 2.4 主控制器控制信号的作用说明

控制信号	取值	说明
AluOP	运算器操作控制符（4 位）	R 型指令根据 Func 选择
MemToReg	寄存器写入数据来自存储器	lw 指令
MemWrite	写内存控制信号	sw 指令　未单独设置 MemRead 信号
AluSrcB	运算器 B 输入选择	lw 指令，sw 指令，立即数运算类指令
RegWrite	寄存器写使能	寄存器写回信号
Syscall	Syscall 指令译码信号	根据$V0 寄存器的值，决定是停机还是输出
SignedExt	立即数符号扩展	ADDI、ADDIU、SLTI 指令
RegDst	写入寄存器编号 rt/rd 选择	R 型指令
Beq	Beq 指令译码信号	Beq 指令，用于有条件分支控制
Bne	Bne 指令译码信号	Bne 指令，用于有条件分支控制
JR	寄存器跳转指令译码信号	JR 指令
JMP	无条件分支控制信号	J、JAL、JR 指令，选择无条件分支地址
JAL	JAL 指令译码信号	JAL 指令 ，选择寄存器写回编号，写回值
SLLV	SLLV 指令译码信号	SLLV 指令，用于选择是否通过寄存器得到偏移位数
SRLV	SRLV 指令译码信号	SRLV 指令，用于选择是否通过寄存器得到偏移位数
LB	LB 指令译码信号	LB 指令
BGTZ	BGEZ 指令译码信号	BGTZ 指令，用于有条件分支控制
R1	R1 寄存器使用情况	用于生成源寄存器使用情况
R2	R2 寄存器使用情况	用于生成源寄存器使用情况

对照所有控制信号，依次分析各条指令，分析该指令执行过程中需要哪些控制信号，对于与本条指令无关的控制信号，控制信号的取值一律为 0，以简化控制器电路的设计。该控制信号表的框架如表 2.5 所示。

表 2.5 主控制器控制信号框架

A	B	C	D	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
#	指令	OpCode (十进制)	FUNCT (十进制)	ALU_OP	MemtoReg	MemWrite	ALU_SRC	RegWrite	SYSCALL	SignedExt	RegDst	BEQ	BNE	JR	JMP	JAL	SLLV	SRLV	LB	BGTZ	R1	R2
1	SLL	0	0	0				1			1											1
2	SRA	0	3	1				1			1											1
3	SRL	0	2	2				1			1											1
4	ADD	0	32	5				1			1										1	1
5	ADDU	0	33	5				1			1										1	1
6	SUB	0	34	6				1			1										1	1
7	AND	0	36	7				1			1										1	1
8	OR	0	37	8				1			1										1	1
9	NOR	0	39	10				1			1										1	1
10	SLT	0	42	11				1			1										1	1
11	SLTU	0	43	12				1			1										1	1
12	JR	0	8	x										1	1						1	
13	SYSCALL	0	12	x					1												1	1
14	J	2	X	x											1							
15	JAL	3	X	x				1							1	1						
16	BEQ	4	X	x						1		1									1	1
17	BNE	5	X	x						1			1								1	1
18	ADDI	8	X	5			1	1		1											1	
19	ANDI	12	X	7			1	1													1	
20	ADDIU	9	X	5			1	1		1											1	
21	SLTI	10	X	11			1	1		1											1	
22	ORI	13	X	8			1	1													1	
23	LW	35	X	5	1		1	1		1											1	
24	SW	43	X	5		1	1			1											1	1
25	SLLV	0	4	0				1			1						1				1	1
26	SRLV	0	6	2				1			1							1			1	1
27	LB	32	x	5	1		1	1		1									1			1
28	BGTZ	7	x	11						1										1	1	

2.2 中断机制设计

2.2.1 总体设计

中断后保存现场，然后通过中断号得到中断入口地址，切换到中断入口地址执行，恢复现场。

2.2.2 硬件设计

在硬布线中增加 ERET、MFC0、MTC0 信号，ERET 用来表示中断，通过 MFC0 和 MTC0 选择中断号，通过多路选择器选择相应的中断入口地址，通过寄存器堆保存现场，跳到中断入口地址执行指令，执行完指令，逐级恢复现场。

2.2.3 软件设计

使用资料包中对应的 hex 文件。

2.3 流水 CPU 设计

2.3.1 总体设计

将 MIPS 指令分为 5 个阶段，分别为 IF、ID、EX、MEM、WB，通过流水接口部件锁存每个阶段的数据，使得在一个周期中，可以并发执行 5 个阶段各自的任务。

2.3.2 流水接口部件设计

流水接口部件主要由 IF/ID，ID/EX，EX/MEM，MEM/WB 四个寄存器组构成。用来暂存上一个周期的数据，和给下一个周期提供数据。对于每个流水接口部件，包含时钟端，清零端，使能端，数据输入以及数据输出端。

2.3.3 理想流水线设计

根据流水线定义，将 5 个阶段以 4 个主要流水接口部件分开，数据通路与单周期 MIPS CPU 类似。

2.4 气泡式流水线设计

理想流水线不能处理有冲突的情况。因此当流水线有数据冲突时，不断插入气泡直到数据冲突解决。ID 段和 WB 段之间存在的险象可以将寄存器文件改为下跳沿触发即可。检测数据冲突的部件就是检测 R1，R2，MEM.RegWrite，EX.RegWrite 之间的相关逻辑是否会造成数据冲突。

2.5 重定向流水线设计

重定向流水线与气泡流水线相同，都是用来解决数据冲突。不同在于，重定向流水线是将所需要读但没有写入寄存器的文件直接定向到需要改数据的位置。这样解决了写后读的问题。相比于气泡流水线插入气泡导致流水线有一段时间会不工作，重定向效率更高。

3 详细设计与实现

3.1 单周期 CPU 实现

3.1.1 主要功能部件实现

1) 程序计数器（PC）

① Logism 实现：

使用一个 32 位寄存器实现程序计数器 PC，触发方式为下降沿触发，输入为下一条将要执行的指令的地址，输出为当前执行指令的地址。Halt 为停机信号，将此控制信号通过非门取反之后和时钟相与，当需要进行停机时，Halt 控制信号为 1，经过非门之后为 0，与时钟信号相与，屏蔽时钟信号，使整个电路停机。如图 3.1 所示。

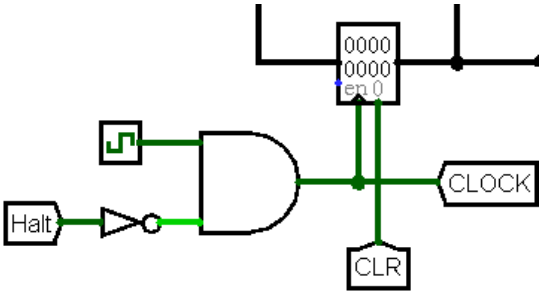


图 3.1 程序计数器（PC）

② FPGA 实现：

程序计数器 PC 的 Verilog 代码如下：

```
always@(negedge clk,posedge clear)
begin
    if(clear)
        pc_out<=0;
    else if(!halt)
        pc_out<=pc_in;
end
```

2) 指令存储器（IM）

① Logism 实现：

使用一个只读存储器 ROM 实现指令存储器（IM）。设置该只读存储器的地址位

宽为 10 位，数据位宽为 32 位。因为 PC 中存储的指令地址有 32 位，而 ROM 地址线宽度有限，仅为 10 位，故将 32 位指令地址高位部分和字节偏移部分直接屏蔽，使用分线器只取 32 位指令地址的 2-11 位作为指令存储器的输入地址。如图 3.2 所示。

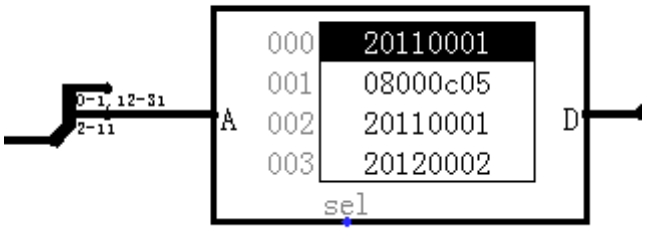


图 3.2 指令存储器（IM）

② FPGA 实现：

直接使用 Vivado 中自带的 ROM 作为指令存储器，其设置如**错误!未找到引用源。**所示。选择 ROM 的数据位宽为 32 位，因为该 ROM 的地址位宽为 10 位，所以选择 ROM 的大小选择为 1024。

指令存储器 IM 的 Verilog 代码如下：

```
pc pcmeml(im_in[11:2],im_out);
```

直接调用之前设置的 ROM 作为指令存储器，输入为指令地址的 2-11 位，输出为该指令。

3） 运算器

Logisim 实现：

使用资料包中的 ALU 即可，如图 3.3 所示。

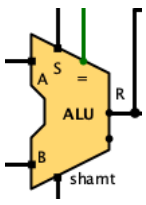


图 3.3 运算器（ALU）

4） 寄存器堆

Logisim 实现：

使用资料包中的 RegFile 即可，如图 3.4 所示。

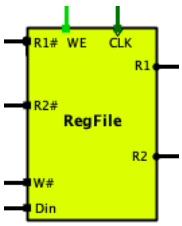


图 3.4 寄存器堆（RF）

5） 数据存储器

Logisim 实现：

直接使用库文件的 RAM 即可，如图 3.5 所示

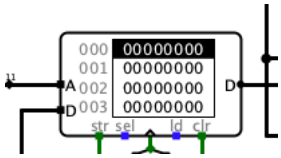


图 3.5 数据存储器（DM）

3.1.2 数据通路的实现

本次课程设计采用的工程化的设计模式，一次性构建所有的数据通路。主要实现方法为，对于每一条指令，将其改写成 RTL（Register Transfer Level），忽略控制类信号，仅保留数据类信号，根据 RTL 功能填写对应指令的数据通路表，描述五大部件之间的连接关系，记录各部件输入端数据来源。

根据总体方案设计中数据通路设计那一小节的详细内容，具体分析每一条指令在执行过程中各个主要部件的输入和输出端口的连接，完成指令系统数据通路表的填写，如表 3.1 所示。

表 3.1 指令系统数据通路表

指令	PC	IM	RF				ALU			DM		Tube
			R1#	R2#	W#	Din	A	B	OP	Addr	Din	
ADD	PC+4	PC	rs	rt	rd	alu	r1	r2	5			
ADDI	PC+4	PC	rs		rt	alu	r1	立即数	5			
ADDIU	PC+4	PC	rs		rt	alu	r1	立即数	5			
ADDU	PC+4	PC	rs	rt	rd	alu	r1	r2	5			
AND	PC+4	PC	rs	rt	rd	alu	r1	r2	7			

指令	PC	IM	RF				ALU			DM		Tube
			R1#	R2#	W#	Din	A	B	OP	Addr	Din	
ANDI	PC+4	PC	rs		rt	alu	r1	立即数	7			
SLL	PC+4	PC		rt	rd	alu	r2	立即数	0			
SRA	PC+4	PC		rt	rd	alu	r2	立即数	1			
SRL	PC+4	PC		rt	rd	alu	r2	立即数	2			
SUB	PC+4	PC	rs	rt	rd	alu	r1	r2	6			
OR	PC+4	PC	rs	rt	rd	alu	r1	r2	8			
ORI	PC+4	PC	rs		rt	alu	r1	立即数	8			
NOR	PC+4	PC	rs	rt	rd	alu	r1	r2	10			

在完成指令系统数据通路表的填写之后，根据列出的数据通路表，进行多指令数据通路的合并输入数，表，将各个主要功能部件进行连接，根据数据通路合并表的最终结果，对于所有的多输入部件使用多路选择器进行输入选择。最终便可以完成数据通路的搭建。

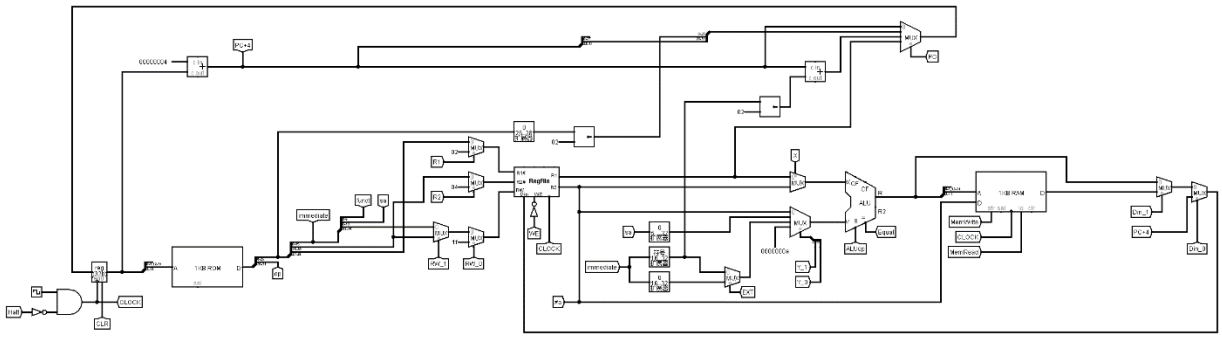


图 3.6 单周期 CPU 数据通路（Logism）

在 Vivado 中使用 Verilog 语言搭建的数据通路的原理图如图 3.7 所示。

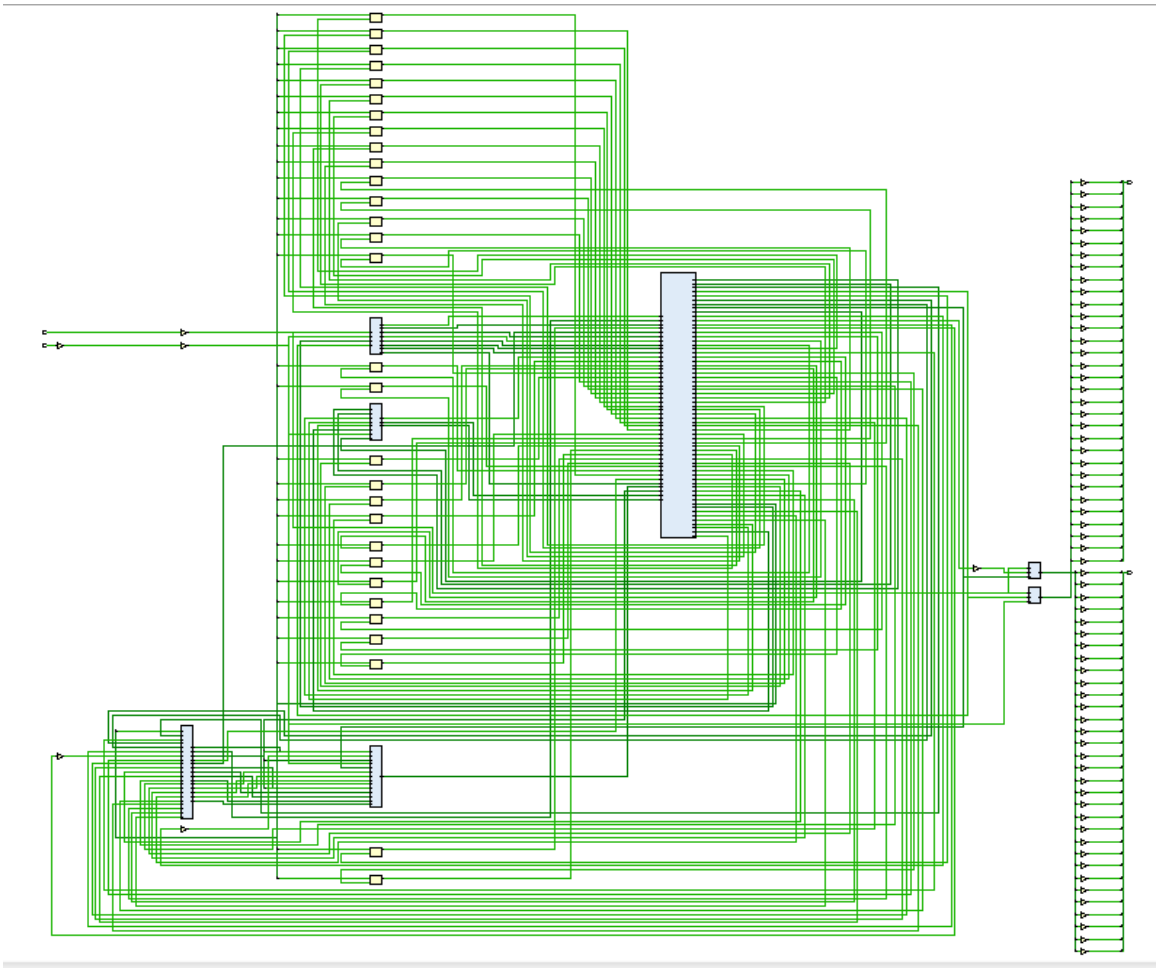


图 3.7 单周期 CPU 数据通路（FPGA）

3.1.3 控制器的实现

根据总体方案设计中控制器的设计那一小节的相关内容，分别在 Logism 和 Vivado 上进行主控制器、Branch 控制器、SYSCALL 控制器的具体实现。

主控制器

对照表 3.2 所示。

表 3.2 主控制器控制信号

指令	R	RW	WE	X	EXT	Y	ALUop	MemWrite	MemRead	Din	Branch	SYSCALL
ADD	00	00	1	0	0	00	0101	0	0	00	00	0
ADDI	00	10	1	0	0	10	0101	0	0	00	00	0
ADDIU	00	10	1	0	0	10	0101	0	0	00	00	0
ADDU	00	00	1	0	0	00	0101	0	0	00	00	0
AND	00	00	1	0	0	00	0111	0	0	00	00	0
ANDI	00	10	1	0	1	10	0111	0	0	00	00	0
SLL	00	00	1	1	0	01	0000	0	0	00	00	0
SRA	00	00	1	1	0	01	0001	0	0	00	00	0
SRL	00	00	1	1	0	01	0010	0	0	00	00	0
SUB	00	00	1	0	0	00	0110	0	0	00	00	0
OR	00	00	1	0	0	00	1000	0	0	00	00	0
ORI	00	10	1	0	1	10	1000	0	0	00	00	0
NOR	00	00	1	0	0	00	1010	0	0	00	00	0
LW	00	10	1	0	0	10	0000	0	1	10	00	0
SW	00	00	0	0	0	10	0000	1	0	00	00	0
BEQ	00	00	0	0	0	00	0000	0	0	00	01	0
BNE	00	00	0	0	0	00	0000	0	0	00	01	0
SLT	00	00	1	0	0	00	1011	0	0	00	00	0
SLTI	00	10	1	0	0	10	1011	0	0	00	00	0
SLTU	00	00	1	0	0	00	1100	0	0	00	00	0
J	00	00	0	0	0	00	0000	0	0	00	10	0
JL	00	01	1	0	0	00	0000	0	0	01	10	0
JR	00	00	0	0	0	00	0000	0	0	00	11	0
SYSCALL	11	00	0	0	0	11	0000	0	0	00	00	1

部生成。

① FPGA 实现

根据在 Logism 实现中得到的各个一位控制信号的表达式，直接使用数据流建模，使用 assign 分的 Verilog 代码过于冗长，故只取对于控制信号 X 的生成代码举例如下：

```
assign
    X=(~OP[5]&~OP[4]&~OP[3]&~OP[2]&~OP[1]&~OP[0]&~F[5]&~F[4]&~F[3]&~F[
    2]&~F[1]&~F[0])|(~OP[5]&~OP[4]&~OP[3]&~OP[2]&~OP[1]&~OP[0]&~F[5]&~F[
    4]&~F[3]&~F[2]&F[1]&F[0])|(~OP[5]&~OP[4]&~OP[3]&~OP[2]&~OP[1]&~OP[0]
    &~F[5]&~F[4]&~F[3]&~F[2]&F[1]&~F[0]);
```

以此类推，最终便可以实现整个主控制器中所有控制信号的生成。在 Vivado 中使用 Verilog 语言构成的主控制器原理图如图 3.8 所示。

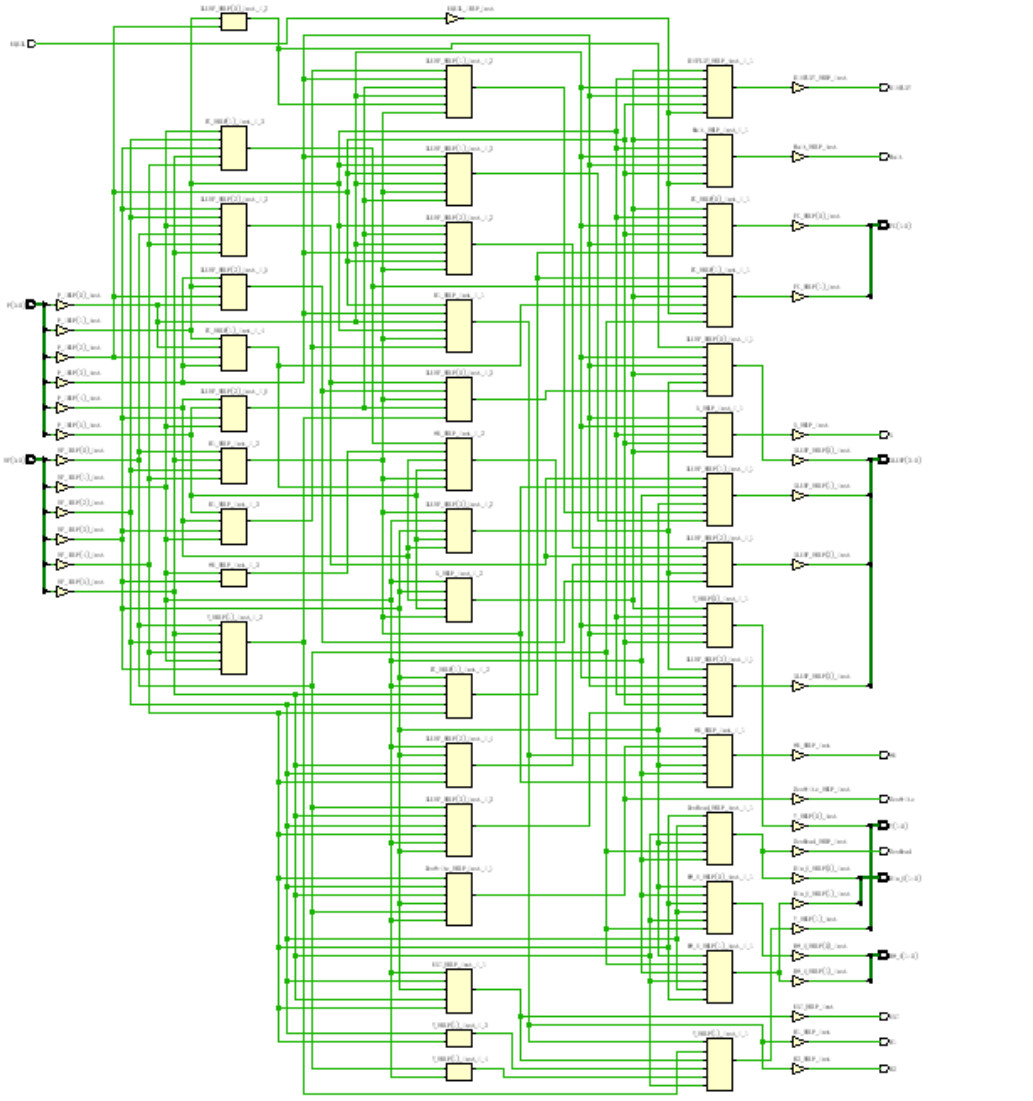


图 3.8 主控制器原理图

3.2 中断机制实现

3.2.1 ir1、ir2、ir3 生成逻辑

通过 mooc 对应章节的 ppt 可以画出相应电路图。

Logisim 实现如图 3.9 所示

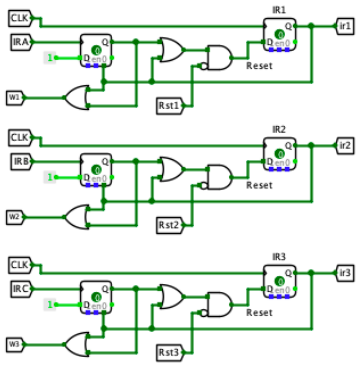


图 3.9ir1、2、3logisim 实现

3.2.2 中断入口逻辑实现

通过优先编码器对 ir1，ir2，ir3 进行编码得到当前最优中断号，通过组选择和多路选择器选择相应的中断入口地址。最优中断号通过译码器和 ERET 信号得到 Rst1，Rst2，Rst3。

Logisim 实现如图 3.10 所示

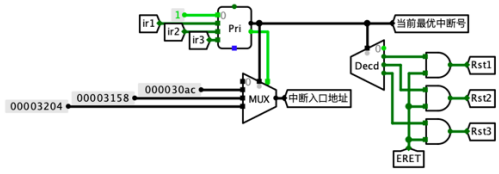


图 3.10 中断入口地址 logisim 实现

3.2.3 寄存器堆逻辑实现

通过多个寄存器暂存每一级中断前的指令地址，用来恢复现场。

Logisim 实现如图 3.11 所示：

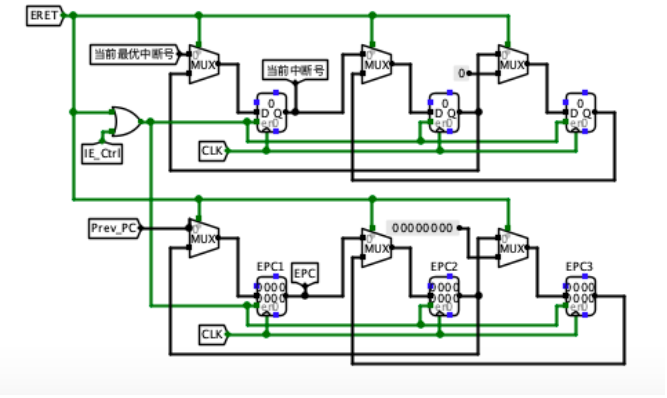


图 3.11 寄存器堆 logisim 实现

3.2.4 中断入口选择信号生成

通过 ERET、MFC0、MTC0 信号以及当前中断号，当前最优中断号得到。Logisim 实现如图 3.12 所示：

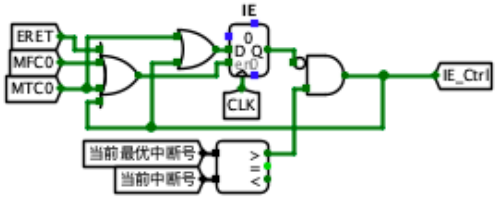


图 3.12 中断入口选择信号 logisim 实现

3.3 流水 CPU 实现

3.3.1 流水接口部件实现

流水接口部件包含时钟端输入，清零端，使能端，数据输入和输出端，以 ID/EX 流水接口部件为例，把上一阶段的数据暂存在寄存器，当时钟到来输出，其余 3 个部件修改寄存器个数或者位宽即可，如图 3.13 所示：

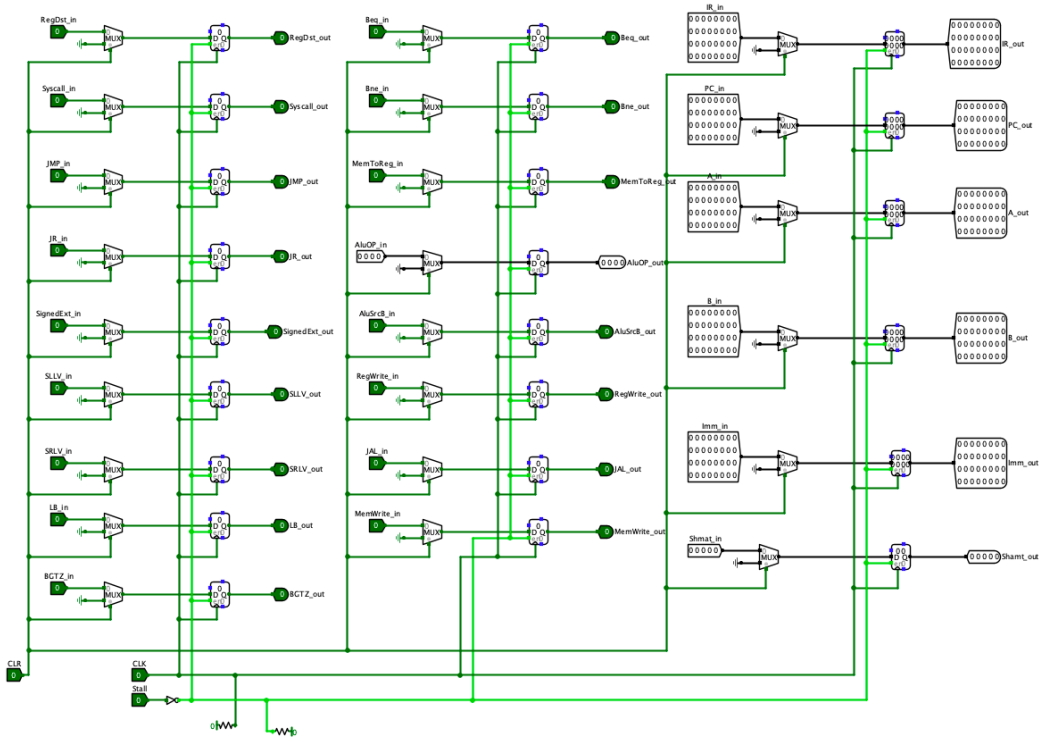


图 3.13 ID/EX 流水接口部件 logisim 实现

3.3.2 理想流水线实现

无需修改数据通路，将对应阶段的数据以流水接口部件分开，如图 3.14 所示。

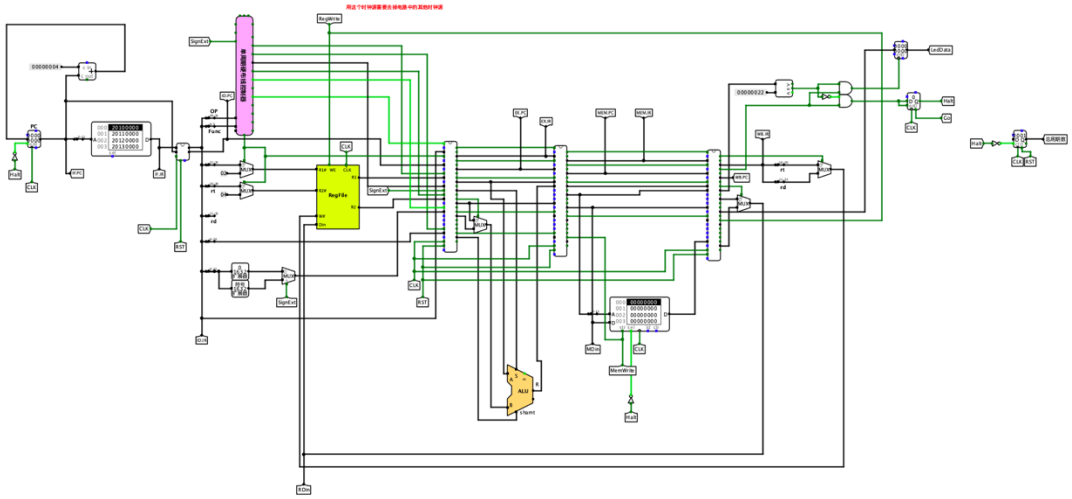


图 3.14 理想流水线 logisim 实现

3.4 气泡式流水线实现

3.4.1 源寄存器使用情况实现

通过表 2.5 所示 R1，R2 的使用情况自动生成逻辑表达式和电路即可，如图 3.15 所示：

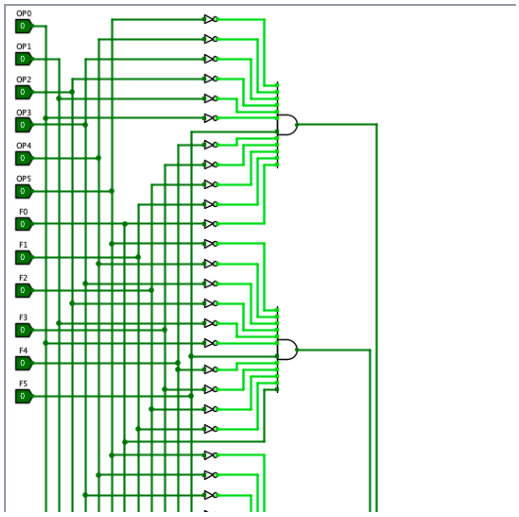


图 3.15 源寄存器使用情况 logisim 实现

3.4.2 数据相关检测实现

通过 MEM.RegWrite,EX.RegWrite,以及源寄存器使用情况即可画出相应电路，如图 3.16 所示

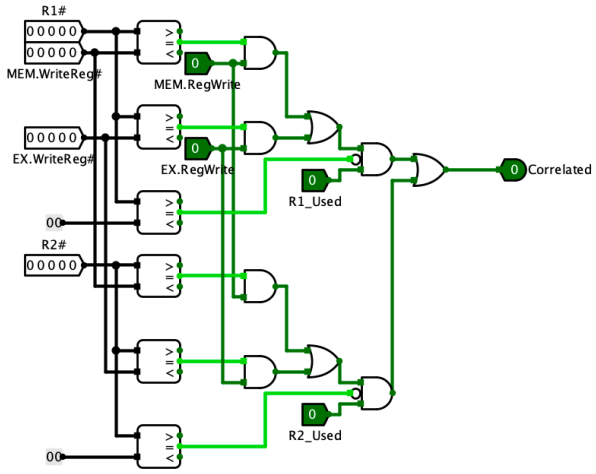


图 3.16 数据相关检测 logisim 实现

3.4.3 整体电路实现

有了源寄存器使用情况子电路和数据相关检测子电路，就可以进行插入气泡的相关操作，如图 3.17 所示：

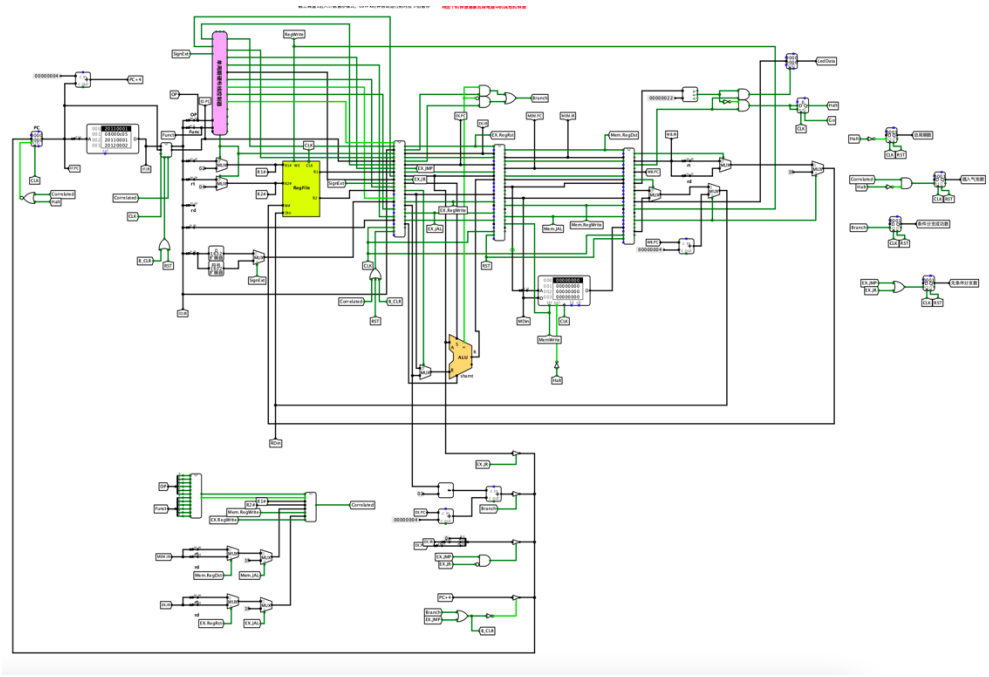


图 3.17 气泡流水线 logisim 实现

3.5 重定向流水线实现

3.5.1 Forward 信号实现

重定向与气泡流水线的不同就在于生成 Forward 信号，通过 Forward 信号进行重定向，如图 3.18 所示：

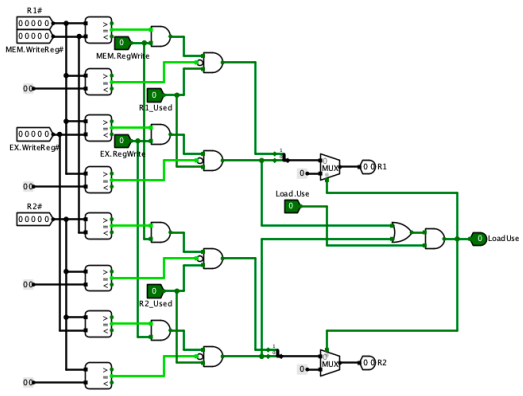


图 3.18 Forward 信号生成 logisim 实现

3.5.2 整体电路实现

重定向也需要使用源寄存器使用情况，与气泡流水线一样，再通过 Forward 信号进行重定向即可，如图 3.19 所示：

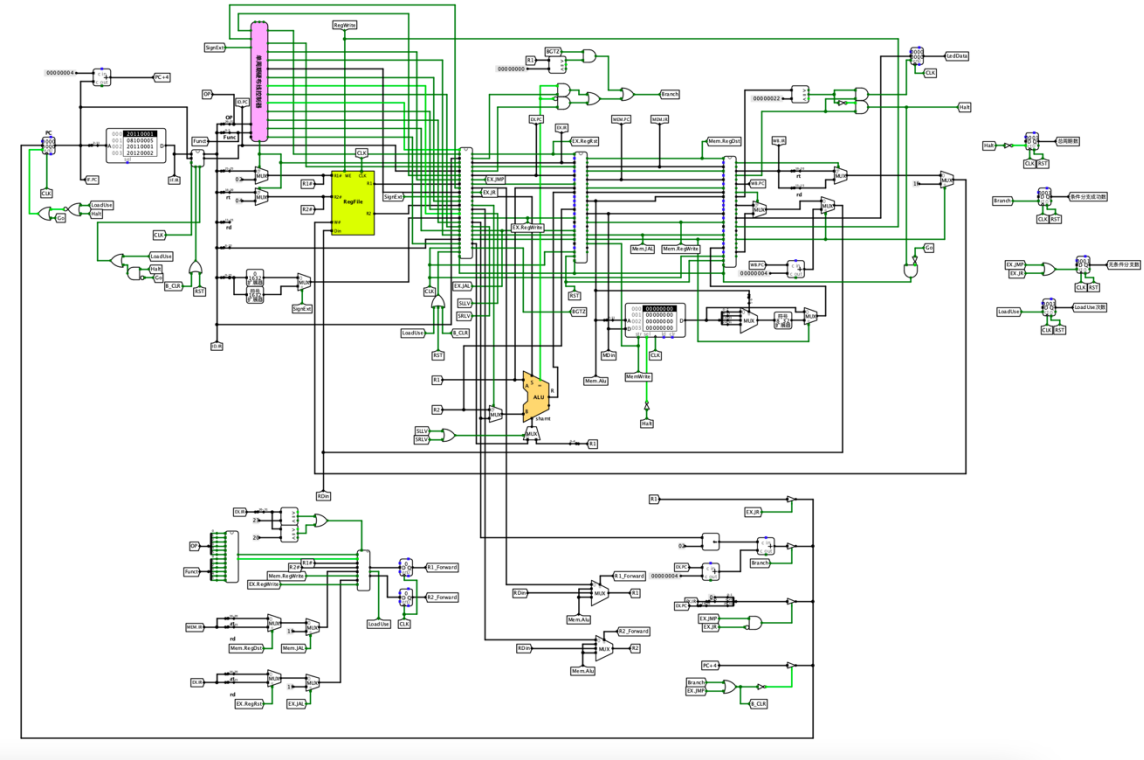


图 3.19 重定向流水线 logisim 实现

4 实验过程与调试

4.1 测试用例和功能测试

测试用例采用 benchmark.hex 文件，由于差异化指令以及中断测试比较难以展示，此处仅展示差异化指令之前的测试结果，并展示 educoder 对应关卡的得分。

4.1.1 单周期 MIPS CPU 测试结果

LED 以及总周期数展示，如图 4.1 所示：

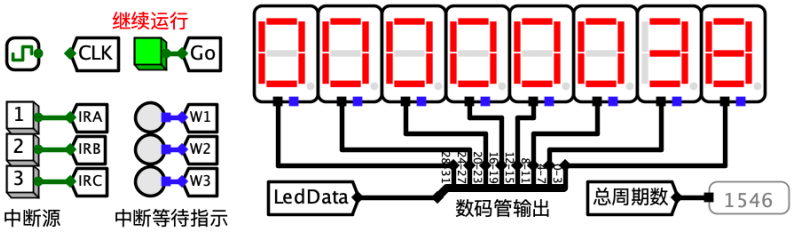


图 4.1 单周期 MIPS LED 测试结果

数据存储器内容如图 4.2 所示

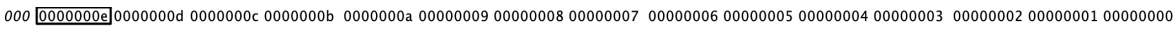


图 4.2 数据存储器内容

4.1.2 理想流水线测试结果

因为理想流水线不能处理数据冲突，因此只能跑理想流水线对应的测试 hex，如图 4.3 所示：

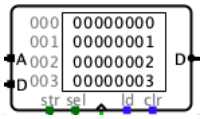


图 4.3 理想流水线测试结果

4.1.3 气泡流水线测试结果

使用 benchmark 测试，LED 显示，总周期数以及气泡数如图 4.4 所示，与标准答案相同：

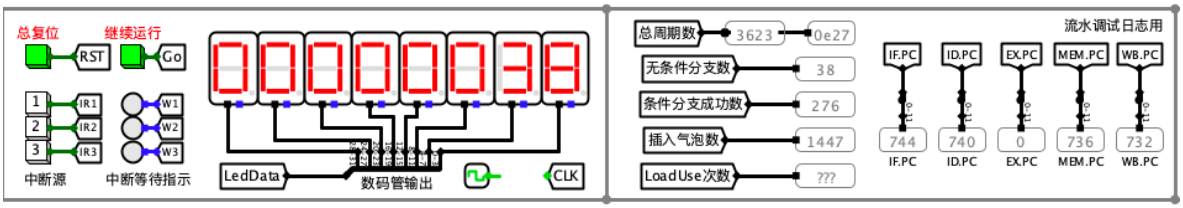


图 4.4 气泡流水线测试结果

4.1.4 重定向流水线测试结果

使用 benchmark 测试，LED 显示，总周期数以及 Load Use 次数如图 4.5 所示，与标准答案相同：

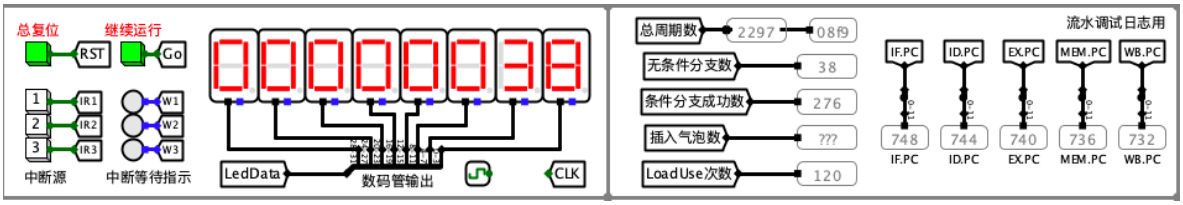


图 4.5 重定向流水线测试结果

4.2 educoder 检测

educoder 平台提交结果如图 4.6 所示：

关卡	任务名称	开启时间	评测次数	完成时间	实战耗时	是否查看答案	经验值	关卡得分	调分
1	单周期CPU(24条指令)	2021-02-21 11:11	17	2021-03-02 08:07	41分钟 39秒	否	1000/1000	11.11/11.11	11.11
2	理想流水线设计	2021-03-02 10:59	9	2021-03-03 11:06	52分钟 33秒	否	300/300	11.11/11.11	11.11
3	气泡流水线设计(EX段分支3624版本)	2021-03-03 11:09	2	2021-03-05 12:01	10分钟 9秒	否	800/800	11.11/11.11	11.11
4	重定向流水线(EX段分支2298版本)	2021-03-05 12:01	6	2021-03-05 21:46	49分钟 32秒	否	1000/1000	11.11/11.11	11.11
5	重定向流水线设计(ID段分支2103版本)	2021-03-03 11:09	0	--	--	否	0/1000	0.00/11.11	0.00
6	气泡流水线设计(ID段分支3309版本)	2021-03-03 11:31	0	--	--	否	0/700	0.00/11.11	0.00
7	单周期MIPS+单级中断	2021-03-05 21:57	6	2021-03-05 23:04	38分钟 48秒	否	700/700	11.11/11.11	11.11
8	多级嵌套中断(EPC硬件堆栈保存)	2021-03-10 08:54	2	2021-03-10 08:56	2分钟 7秒	否	1000/1000	11.11/11.11	11.11
9	多级嵌套中断(EPC内存堆栈保存)	2021-03-11 21:00	0	--	--	否	0/1000	0.00/11.12	0.00

图 4.6 educoder 提交结果

4.3 性能分析

对于单周期 MIPS 而言，它的总周期数虽然是最少的，但流水线技术是为了增加并发度，缩短每个硬件的时钟周期，可能由于测试程序的关系才会有这样的结果，实

际中应该流水线技术会优于单周期 MIPS CPU。

而两个流水线对路比较而言，显然重定向的流水线更优于气泡流水线，因为不再产生气泡使 CPU 某个阶段没有指令执行，因此获得了巨大的性能提升。

4.4 主要故障与调试

4.4.1 RegFile 上升沿与流水线

故障现象：在 0x1a 拍报错

原因分析：在 WB 段将数据写回时会有 0.5 拍的延迟，对于气泡流水线而言，它增加的气泡只能避免 1 拍的延迟

解决方案：因此要解决 0.5 拍就需要将 Reg File 的上升沿，改为下降沿，这样就解决了 0.5 拍。如图 4.7 所示，这是第三天记录的相关问题：

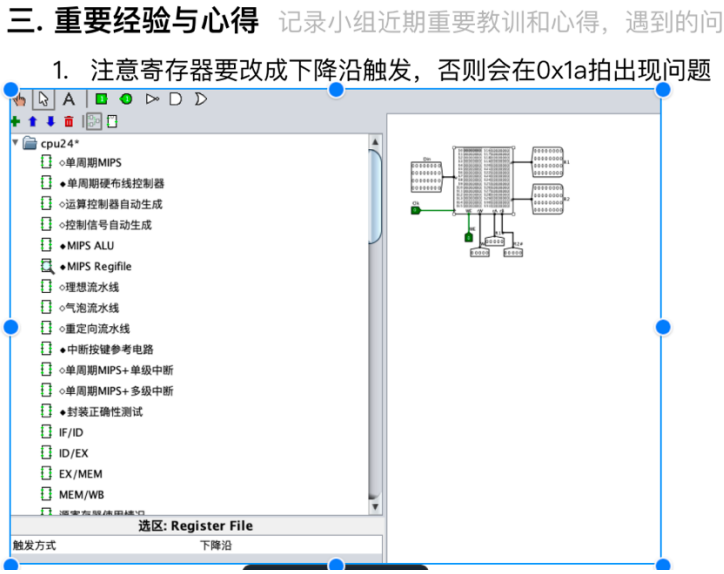


图 4.7 下降沿

4.4.2 加入差异化指令后无法暂停

故障现象：benchmark 中加入差异化相关测试代码后，流水线无法在执行差异化指令之前暂停。

原因分析：使能端逻辑没有写好，用了一个寄存器暂存 Halt，导致有一拍的延迟。

解决方案：重写使能端逻辑，并跟 Go 信号的非进行与即可，如图 4.8 所示

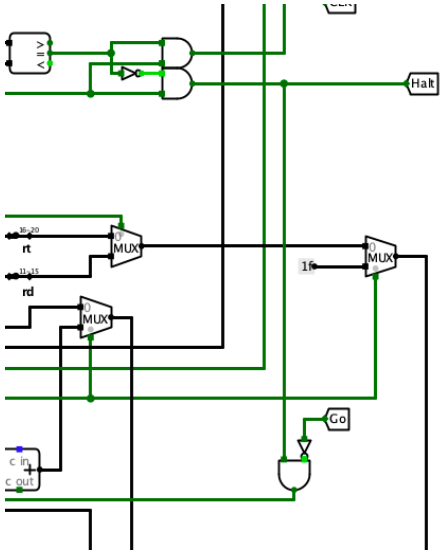


图 4.8 重写使能端逻辑

4.5 实验进度

表 4.1 课程设计进度表

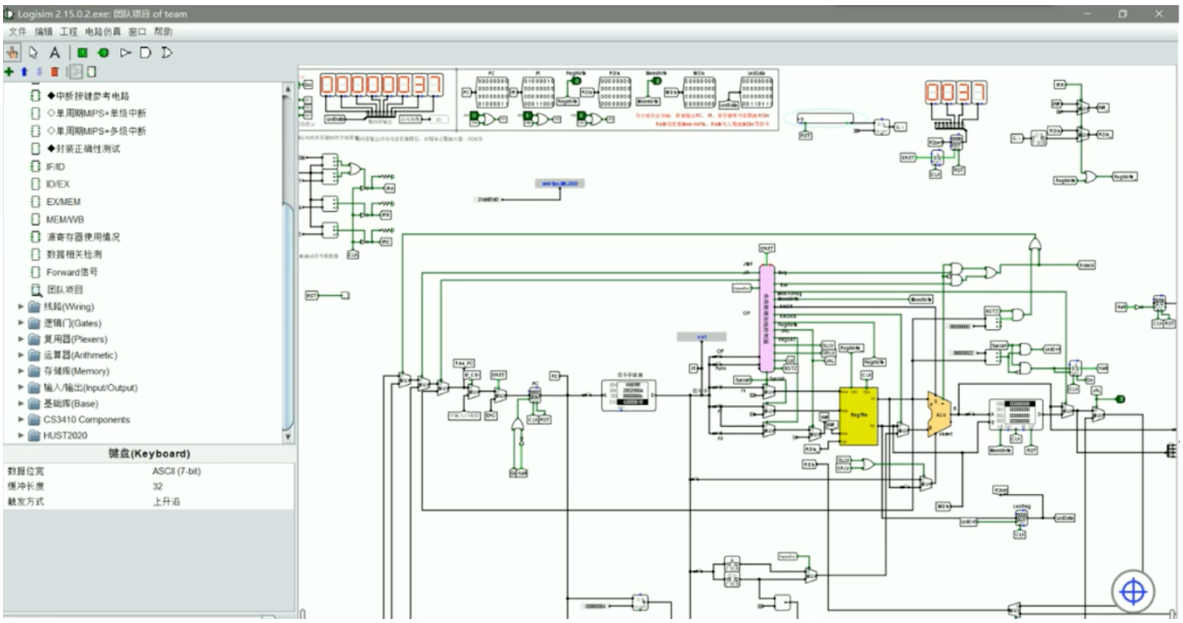
时间	进度
第一天	复习组成原理 CPU 相关理论知识，阅读课设任务书，阅读 MIPS 指令手册，并列出 CPU 各部件的数据通路表，并完成数据通路的基本构建。
第二天	完成单周期 CPU 的控制信号表，使用 Logisim 搭建控制器，实现了单周期 CPU 并且通过了测试。完成部分 Logism 单周期 CPU 故障报告。
第三天	完成理想流水线的测试。
第四天	完成气泡流水线的测试。
第五天	完成重定向流水线的测试。
第六天	完成单级中断的电路，但提交没通过。
第七天	发现是封装有问题，单级中断测试通过。
第八天	完成多级中断的测试。
第九天	加入差异化指令，但无法进行暂停。
第十天	重写使能端逻辑，完成差异化指令测试。

5 团队任务

5.1 选题与设计

本实验输出斐波那契数列第 n 项数字，其中 n 由输入决定。输入由中断信号控制，产生中断时，将键盘字符读入到寄存器中，每次产生的中间结果也会输出，最后停留在最终结果上。

5.2 效果展示



5.3 报告链接

https://www.bilibili.com/video/BV1wh411S78v/

6 设计总结与心得

6.1 课设总结

基于对象的存储是为了克服当前基于块的存储存在的诸多难题，在存储接口和结构层次的重要发展。可以根据应用负载选择优化的存储策略。作了如下几点工作：

1）设计了 24+3+4 条指令相关的数据通路以及对应的生成信号表，完成了单周期 MIPS CPU 的电路绘制。

2）设计并实现了理想流水线的逻辑电路图。

3）设计并实现了使用插入气泡法解决数据冲突的气泡流水线。

4）设计并实现了使用重定向解决数据冲突的重定向流水线。

5）设计并实现了可以多级嵌套中断的逻辑电路图。

6.2 课设心得

本次课程设计可以说是大学三年中最痛苦，难度最大的一次课程设计。不过，也是学到的内容最多的一次课程设计。刚开始写单周期MIPS和理想流水线没什么体会，比较简单，一两天就完成了。可是后面的气泡流水线，重定向流水线可以说是最难的一部分。从设计电路到绘制电路，没有一步是轻松的，后面加入了差异化指令之后，原本以为没问题的电路又不能用了，又画了两个晚上。到后面的中断，逻辑非常简单，保存现场，执行，恢复现场，但画起来难度也是很大，特别是多级嵌套，要考虑最优的中断号，也是想了很久，看了很多资料。

不过，有一点相比于其他课设是比较好的，可以使用 educoder 平台查看错误信息，这真的是节省了我很多时间。得到相应错误的周期数再去找错误会非常简单。

对于团队任务，即使我们想做一个非常简单的输入输出还是非常的困难，值得庆幸的是，键盘输入有现成的元件，让我们开心了好一会，但是手写 MIPS 这一步是真的想放弃。逻辑很简单，MARS 上面就是报操作数类型不正确，我们也没有系统学过 MIPS，只能对着现有的测试代码慢慢看，也是比较痛苦的。

然而对于本次课程设计，我还有一些小小的建议和改进。首先就是 MIPS 代码，真的希望能有一个直接将高级语言转换成 MIPS 代码的工具。其次就是一些知识，mooc 以及 educoder 都没有相应的内容，比如多级中断，mooc 上就只有一个表格，以

及 mooc 上说有个转时空图的工具也没有。

最后在这里也感谢三位老师对于我在本次课程设计中无数问题的耐心解答，也感谢本组所有成员在课程设计中对于我的帮助和建议。我相信组成原理课程设计必将成为我整个大学生涯中一段无比难忘的回忆。

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] 谭志虎，秦磊华，吴非，肖亮.计算机组成原理. 北京：人民邮电出版社，2021 年.

[4] 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.

[5] 谭志虎，秦磊华，胡迪青.计算机组成原理实践教程.北京：清华大学出版社，2018.

[6] 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.

[7] 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008 年.

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：黄俊淇