华中科技大学

2023

硬件综合训练　　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：	CS2108
学　　号：	I202120037
姓　　名：	郑澍频
电　　话：	15623315168
邮　　件：	i202120037@hust.edu.cn

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	JALR	转移到指定寄存器	If $v0==10 halt(停机指令)
24	ECALL	系统调用	else 数码管显示$a0 值
25	CSRRSI	访问 CP0	中断相关，可简化，选做
26	CSRRCI	访问 CP0	中断相关，可简化，选做
27	URET	中断返回	异常返回，选做
28	SRA	右算术位移	指令格式参考 RISC-V32 指令集，最终功能以 RARS 模拟器为准。
29	XOR	异或	
30	SB	字节储存	
31	BGE	大于等于	

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微程序控制，且主、控存分开的方案，即采用微程序控制方式，实现主存储器（MM）和微程序控制存储器（CM）不共用一个存储器的方式完成方案的设计。同时在实施的过程中，采用部分电路用 FPGA 方式下载、部分电路用硬件搭建的方式完成。

总体结构图如图 2.1 所示。

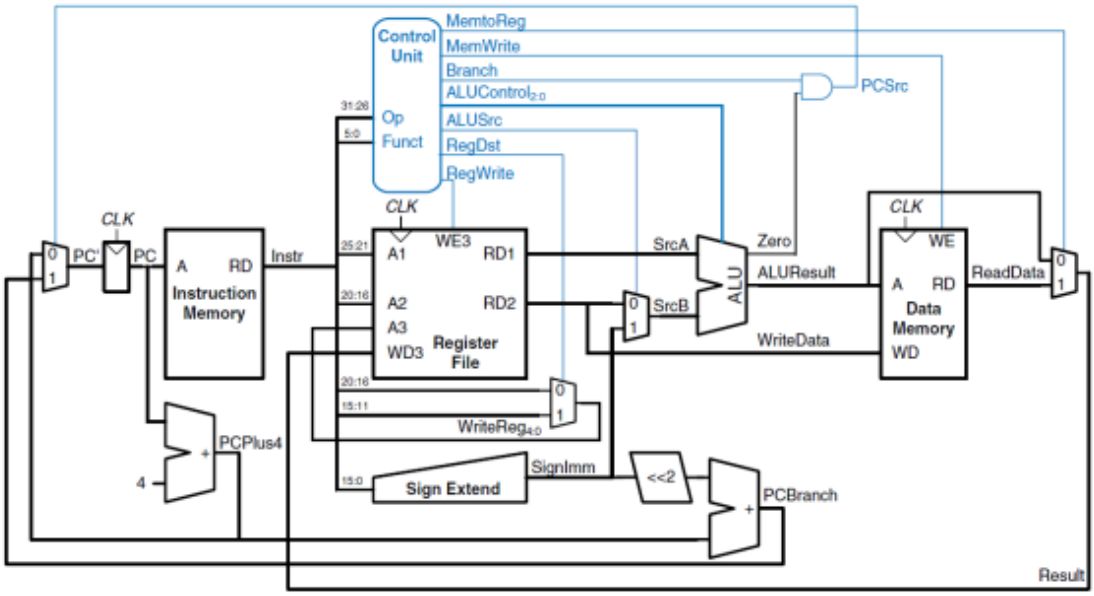

图 2.1 总体结构图

2.1.1 主要功能部件

单周期 CPU 的主要功能部件为程序计数器 PC、指令储存器 IM、寄存器堆 RF、运算器 ALU 和数据存储器 DM。

1. 程序计数器 PC

程序计数器由一个 32 位寄存器实现，输入由是否是跳转类指令决定是分支目标地址还是 PC+4。

2. 指令存储器 IM

指令寄存器 IM 是由只读存储器实现的寄存器，一个地址对应四个字节，即一条指令的数据长度。用于存储需要执行的指令，由于指令寄存器 IM 是个只读存储器实现的寄存器，因此再 Logisim 仿真平台中采用只读存储器 ROM 来实现。使用过程中需要加载不同的数据镜像。

3. 运算器 ALU

运算器用于进行各种运算，以从 Regfile 中读取的寄存器内容或者指令中的立即数扩展得到的值作为运算数，输出运算结果或者比较信号供其他电路使用。ALU 引脚功能描述如表 2.1 所示，支持的运算功能及运算码如表 2.2 所示。

表 2.1 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
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

表 2.2 运算码及其运算功能描述

ALU_OP	运算功能
0000	Result = X << Y 逻辑左移 （Y 取低五位） Result2=0
0001	Result = X >>Y 算术右移 （Y 取低五位） Result2=0
0010	Result = X >> Y 逻辑右移 （Y 取低五位） Result2=0

ALU_OP	运算功能
0011	Result = (X * Y)[31:0]; Result2 = (X * Y)[63:32] 无符号乘法
0100	Result = X/Y; Result2 = X%Y 无符号除法
0101	Result = X + Y (Set OF/UOF)
0110	Result = X - Y (Set OF/UOF)
0111	Result = X & Y 按位与
1000	Result = X | Y 按位或
1001	Result = X⊕Y 按位异或
1010	Result = ~(X |Y) 按位或非
1011	Result = (X < Y) ? 1 : 0 符号比较
1100	Result = (X < Y) ? 1 : 0 无符号比较

4. 寄存器堆 Regfile

寄存器堆 Regfile 是由 CS3410 提供的 Register File，它提供了 32 个 32 位寄存器。具体寄存器说明如下表显示。

表 2.3 寄存器说明

寄存器	别名	说明
X0	Zero	恒零寄存器
X1	Ra	返回地址
X2	Sp	栈指针
X3	Gp	全局指针
X4	Tp	线程指针
X5-X7	T0-T2	临时变量寄存器 0 到 2
X8	S0/fp	保存寄存器 0/栈帧指针
X9	S1	保存寄存器 1
X10-X17	A0-A7	函数参数 0 到 7
X18-X27	S2-S11	保存寄存器 2 到 11
X28-X31	T3-T6	临时变量寄存器 3 到 6

5. 数据存储器 DM

由随机存储器实现的数据存储器，一个地址对应四个字节的数据。数据存储器用于存储程序运行中产生的数据，需支持读写操作所以选择使用 RAM 来实现。分成字访问，半字访问，字节访问的不同模式。不同模式需要不同数量的 RAM 实现。

2.1.2 数据通路的设计

数据通路的设计由下表显示。

表 2.4 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
add	PC+4	PC	RS1	RS2	RD	ALU	RS1	RS2	5	-	-
sub	PC+4	PC	RS1	RS2	RD	ALU	RS1	RS2	6	-	-
and	PC+4	PC	RS1	RS2	RD	ALU	RS1	RS2	7	-	-
or	PC+4	PC	RS1	RS2	RD	ALU	RS1	RS2	8	-	-
slt	PC+4	PC	RS1	RS2	RD	ALU	RS1	RS2	11	-	-
sltu	PC+4	PC	RS1	RS2	RD	ALU	RS1	RS2	12	-	-
addi	PC+4	PC	RS1	-	RD	ALU	RS1	Imm	5	-	-
andi	PC+4	PC	RS1	-	RD	ALU	RS1	Imm	7	-	-
ori	PC+4	PC	RS1	--	RD	ALU	RS1	Imm	8	-	-
xori	PC+4	PC	RS1	-	RD	ALU	RS1	Imm	9	-	-
slti	PC+4	PC	RS1	-	RD	ALU	RS1	Imm	11	-	-
slli	PC+4	PC	RS1	-	RD	ALU	RS1	Imm	0	-	-
srli	PC+4	PC	RS1	-	RD	ALU	RS1	Imm	2	-	-
srai	PC+4	PC	RS1	-	RD	ALU	RS1	Imm	1	-	-
lw	PC+4	PC	RS1	-	RD	DM	RS1	Imm	5	ALU	-
sw	PC+4	PC	RS1	RS2	-	-	RS1	Imm	5	ALU	R2
ecall	PC+4	PC	$a7	$a0	-	-	RS1	RS2	-	-	-

指令	PC	IM	RF				ALU			DM	
			R1#	R2#	W#	Din	A	B	OP	Addr	Din
beq	PC+OFFSET/PC+4	PC	RS1	RS2	-	-	RS1	RS2	-	-	-
bne	PC+OFFSET/PC+4	PC	RS1	RS2	-	-	RS1	RS2	-	-	-
jal	PC+OFFSET	PC	-	-	RD	PC+4	-	-	-	-	-
jalr	ALU	PC	RS1	-	RD	PC+4	RS1	Imm	5	-	-
CSRRSI	PC+4	PC	-	-	-	-	-	-	-	-	-
CSRRCI	PC+4	PC	-	-	-	-	-	-	-	-	-
URET	PC+4	PC	-	-	-	-	-	-	-	-	-
SRA	PC+4	PC	RS1	RS2	RD	ALU	RS1	Imm	1	-	-
XOR	PC+4	PC	RS1	RS2	RD	ALU	RS1	Imm	9	-	-
SB	PC+4	PC	RS1	RS2	-	-	RS1	Imm	5	-	-
BGE	PC+OFFSET/PC+4	PC	RS1	RS2	-	-	RS1	RS2	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 2.5。

表 2.5 主控制器控制信号的作用说明

控制信号	取值	说明
ALI_OP	0-12	选择 ALU 要进行的运算类型，具体运算见表 2.2
MemToReg	0	选择 ALU 运算结果为寄存器写入值
	1	选择数据存储器读出值为寄存器写入值，如 LW 指令
MemWrite	0	
	1	加载值到数据存储器，如 SW 指令
ALU_SrcB	0	ALU 的输入端送入 R2
	1	ALU 的输入端 B 送入立即数
RegWrite	0	
	1	将 RDin 写入寄存器 Rd

控制信号	取值	说明
Ecall	0/1	ecall 指令译码信号
S_Type	0/1	S 型指令译码信号
BEQ	0/1	BEQ 指令译码信号
BNE	0/1	BNE 指令译码信号
JAL	0/1	JAL 指令译码信号
JALR	0/1	JALR 指令译码信号
BGE	0/1	BGE 指令译码信号
SB	0/1	SB 指令译码信号
R1Used	0	未使用 R1#源寄存器
	1	使用 R1#源寄存器
R2Used	0	未使用 R2#源寄存器
	1	使用 R2#源寄存器

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 2.66 所示。

表 2.6 主控制器控制信号框架

指令	ALU_OP	MemToReg	MemWrite	ALU_SrcB	RegWrite	Ecall	S_type	BEQ	BNE	JAL	JALR	BGE	SB
add	5				1								
sub	6				1								
and	7				1								
or	8				1								
slt	11				1								
sltu	12				1								
addi	5			1	1								
andi	7			1	1								
ori	8			1	1								

指令	ALU_OP	MemToReg	MemWrite	ALU_SrcB	RegWrite	Ecall	S_type	BEQ	BNE	JAL	JALR	BGE	SB
xori	9			1	1								
slti	11			1	1								
slli	0			1	1								
srli	2			1	1								
srai	1			1	1								
lw	5	1		1	1								
sw	5		1	1			1						
ecall						1							
beq	5							1					
bne	5								1				
jal	5				1					1			
jalr	5			1	1						1		
csrrsi													
csrrci													
uret													
sra	1				1								
xor	9				1								
sb	5		1	1			1						1
bge	11											1	

2.2 中断机制设计

2.2.1 总体设计

中断机制可以分为外部中断和内部中断，本次实验内容中使用的是外部中断。本次实验将中断机制分成了单级中断和多级中断两种形式，需要软硬件两者的共同支持。

单级中断的原理是在转到中断服务程序前，需要在中断断点寄存器 EPC 中维护断点，也就是 PC 的值。这样是为了确保执行中断服务完毕后，程序能够返回进入终

端的位置。

多级中断的设计原理是，由于涉及到不同的中断源的优先级，需要在硬件中判断此中断的优先级来决定是否打断正在进行的中断，并使用堆栈存储处理多级中断进行时的不同断点的值。

2.2.2 硬件设计

对于单周期单级中断：参考中断按键参考电路，为每个中断号提供一个中断采集电路，当多个中断请求信号同时存在时，根据优先级处理中断信号，将中断入口地址送入 PC，并让中断使能寄存器 IE 关闭中断。中断进行时无法被其他中断请求打断，中断进行后 CPU 会根据 URET 指令执行中断返回，将 EPC 中的值送回 PC，跳转到断点出继续执行主程序，同时中断使能寄存器恢复位开中断的状态。

对于单周期多级中断：在单周期单级中断里增加一个中断优先级判断电路。为了简约明了，将无中断设置为 0 级中断服务程序。使用中断优先级判断电路判断中断优先级后，将高级中断直接中断使能寄存器 IE 开中断并保存断点，将中断入口地址送入 PC；将低级中断通过中断使能寄存器暂存。嵌套的中断产生的断点也会通过堆栈的方式保存，堆栈通过硬件堆栈实现。

2.2.3 软件设计

支持 CSRRSI、CSRRCI 和 URET 指令，配合硬件完成开关中断，中断返回等操作。同时需要从程序中获取中断服务程序入口的地址，以此作为常量连接相关硬件逻辑电路来模拟中断向量表。

2.3 流水 CPU 设计

2.3.1 总体设计

CPU 流水线设计共有 5 个阶段，分别是取指令阶段 IF、译码阶段 ID、执行阶段 EX、存储阶段 MEM 以及写回阶段 WB。在每个阶段之间插入流水寄存器锁存当前阶段的数据，这样就可以在每个不同阶段处理不同的指令，提高执行效率。理想流水线不考虑分支冲突、数据冲突等冒险问题。这些是气泡流水线和重定向流水线的基础，因此需要在理想流水线中完成流水线寄存器设计和基本结构。

2.3.2 流水接口部件设计

流水接口部件的设计参考了设计运算器时使用过的乘法流水线接口部件。每个流水线接口包括时钟端，同步清零端，使能端和数据输入输出端口。因为每个不同阶段需要的数据不同，所以每个不同阶段的流水接口部件的数据段也有所不同，需要根据实际情况修改流水寄存器所需要存的数据。

2.3.3 理想流水线设计

X 刘辉接口部件设计完成后，只需要讲单周期 CPU 设计中的每个不同阶段需要用到的部件分离，并在每个阶段间插入流水接口部件。在这期间需要特别主义的是细微的逻辑变化会导致错误的逻辑，所以需要更小心的处理。

2.4 气泡式流水线设计

对于分支冲突的处理方式是在 EX 阶段进行分支跳转时，清除 IF/ID 流水寄存器和 ID/EX 流水寄存器的内容，来删除误取得指令。

对于数据冲突的处理方式是在 ID 阶段判断是否与 EX 阶段或 MEM 阶段发生数据相关。如有，暂停 ID/EX 段流水寄存器得指令，插入空气泡，等待数据变成不相关再继续执行。

对于数据相关的处理是通过源寄存器得使用情况，即：EX 或 MEM 是否写入寄存器、ID 寄存器 R1#R2#与 EX 段得 WriteReg#或 MEM 段得 WriteReg 是否相等组成逻辑来确认是否发生数据相关。

从数据相关信号、分支跳转信号、halt 信号输出 PC 得使能端信号以及各个流水线继勋奇的使能端和清零端信号，送入对应部件中。

2.5 重定向流水线设计

对于分支冲突的处理方法与气泡流水线相同，相关内容请查看 2.4 气泡流水线设计。

对于数据冲突的处理方法是在发生数据相关的情况下不插入气泡，而在下一个时钟周期将正确且未写回到寄存器中的数据重定向到 EX 段。

对于 LoadUse，如果像个相邻的指令存在数据相关，且前一条指令是访存指令时，

这种数据相关不适用重定向处理。如若使用重定向方式处理，则需要将数据存储器的输出重定向到 EX 段，使关键路径包含访存阶段来加长关键路径影响 CPU 的工作效率。

对于选择信号模块，使用气泡流水线中数据相关模块修改，输出 Fwd1 和 Fwd2 信号，由由 Fwd1 和 Fwd2 信号作为多路选择器的选择端以选择 EX 段需要重定向的正确数据。

从 LoadUse 相关信号、分支跳转信号、halt 信号输出 PC 得使能端信号以及各个流水线继勋奇的使能端和清零端信号，送入对应部件中。

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）

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。

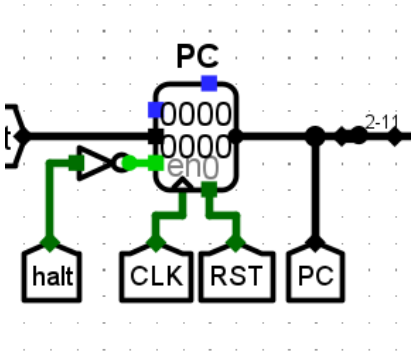

图 3.1 程序计数器（PC）

2) 指令存储器（IM）

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。

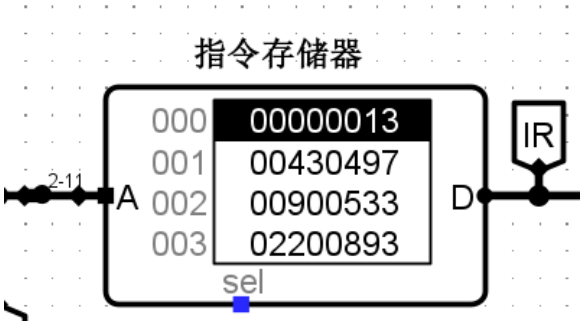

图 3.2 指令存储器（IM）

3) 运算器（ALU）

使用 cs3410.jar 中的运算部件。电路如图 3.3 所示

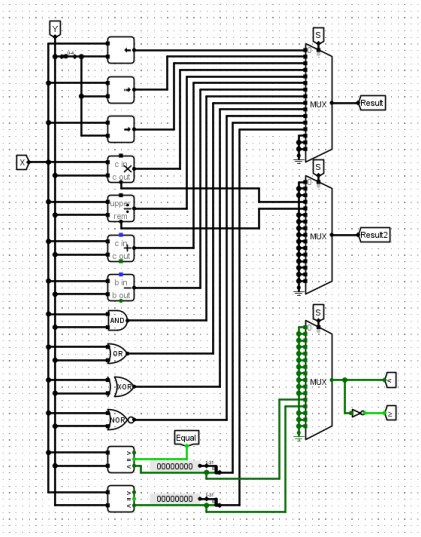

图 3.3 运算器（ALU）

4) 寄存器堆（RegFile）

使用 cs3410.jar 中的寄存器堆部件。电路如图 3.4 所示

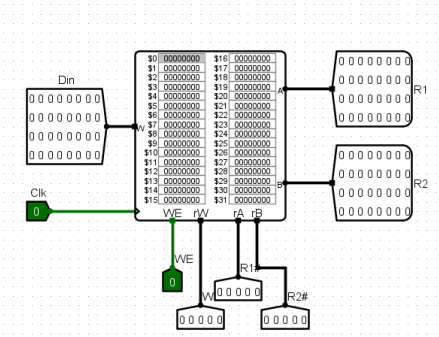

图 3.4 寄存器堆（RegFile）

5) 数据存储器（DM）

如图 3.5 所示

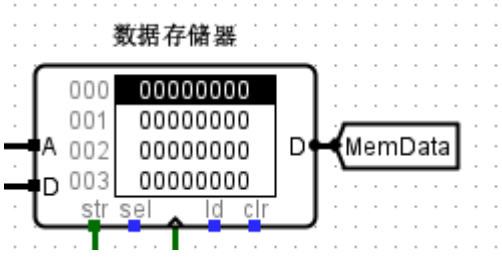

图 3.5 数据存储器（DM）

3.1.2 数据通路的实现

数据通路采用简单迭代法实现，先完成一条基础指令（如 R 型指令）的数据通路，然后再在其之上不断增加新的数据通路，按照 2.1.2 节中表 2.3 与附加条件设计，完成全部通路设计。如图 3.6 所示。

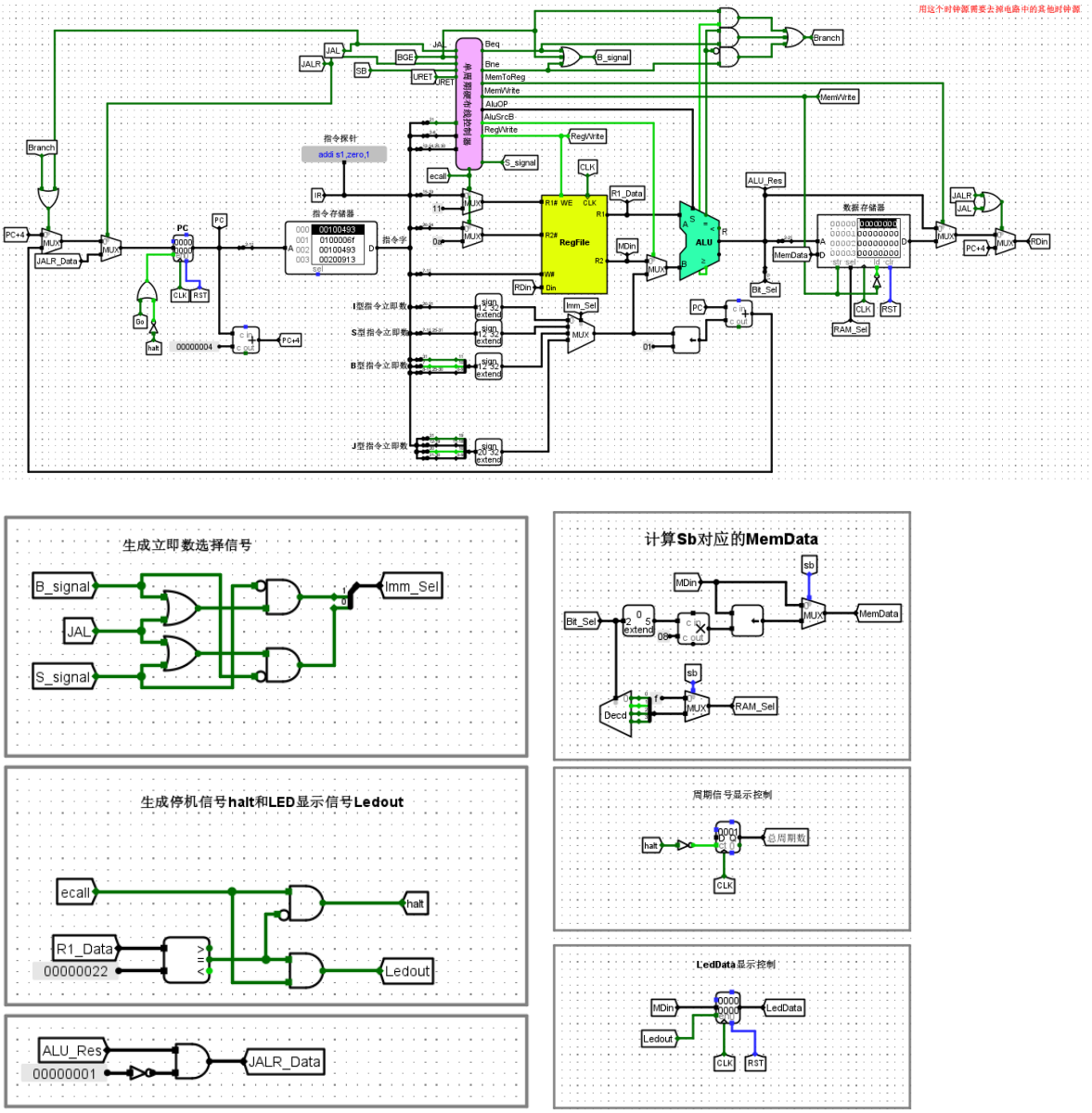

图 3.6 数据通路的实现

3.1.3 控制器的实现

按照表 2.5 在硬布线控制器表达式自动生成 excel 表中填表，获取控制信号的表达式，借助 Logisim 的组合逻辑电路分析功能，生成硬布线控制器。为方便自动生成电路，为 ALU_OP 单独用一个电路输出，再将所有控制信号整合输出，如图 3.7 所示。

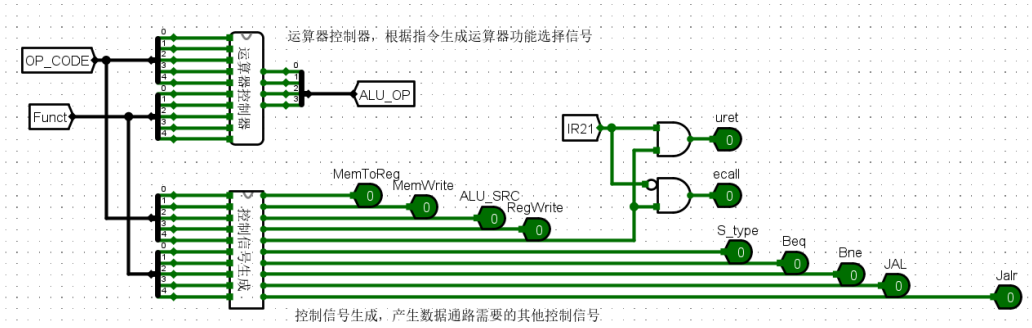

图 3.7 控制器的实现

3.2 中断机制实现

3.2.1 单周期单级中断机制

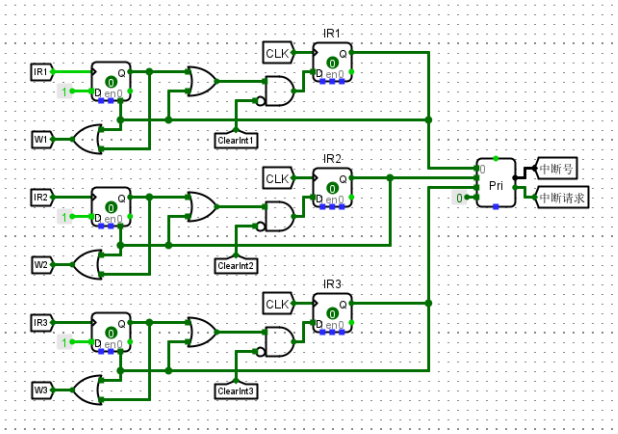

图 3.8 中断信号采样电路

单周期单级中断 CPU 建立在单周期 CPU 的基础上再进行稍微一点点的修改。

修改 1：增加中断信号采样电路收集中断信号，如图 3.8 所示。被收集的中断信号经过优先级排序后，进入中断入口选择。同时，优先编码器解析的存在中断请求信号传递到中断使能寄存器中关中断，将 IE 寄存器的信号取非，再与中断请求信号进行与操作，将结果信号作为是否中断服务程序的信号。

3.2.2 单周期多级中断机制

单周期多级中断 CPU 建立在单级中断的基础上再进行一点点修改。

中断使能部分考虑到高级中断需要中断低级中断，因此当多个中断请求同时出现时，应进行中断等级比较，中断使能寄存器应相应关中断指令 CSRRCI、开中断指令 CSRRSI。对于断电保存部分，需要保存多个断点，理由计数器设计硬件堆栈实现，同时要对低级中断请求进行暂存。

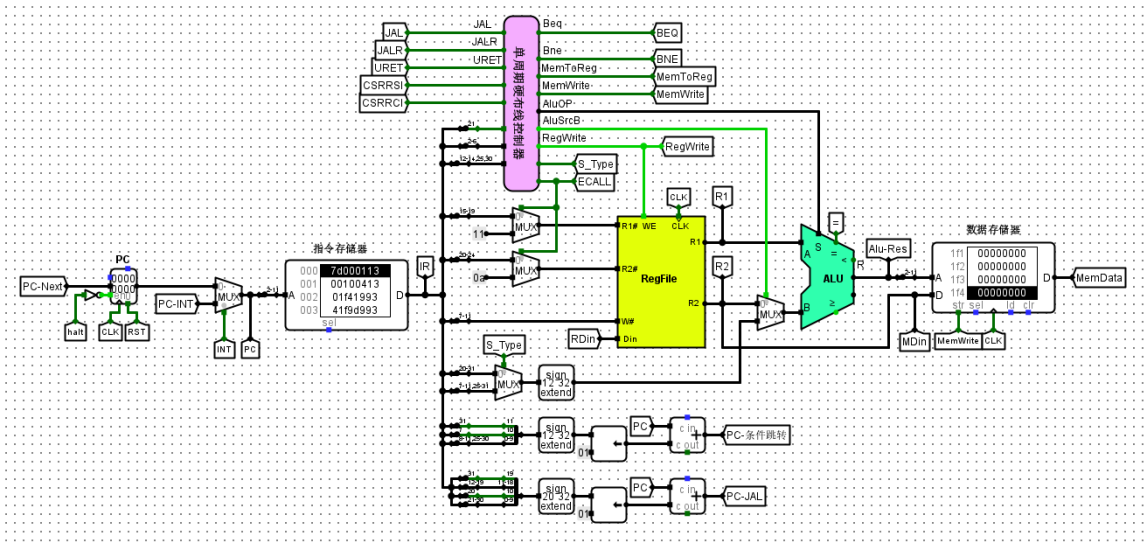

图 3.9 单周期多级中断机制

3.3 流水 CPU 实现

3.3.1 流水接口部件实现

流水接口部件内部采用寄存器进行锁存，寄存器均采用上升沿触发，同时需要实现同步清零端、使能端。以 ID/EX 段流水接口部件为例，如图 3.10 所示。

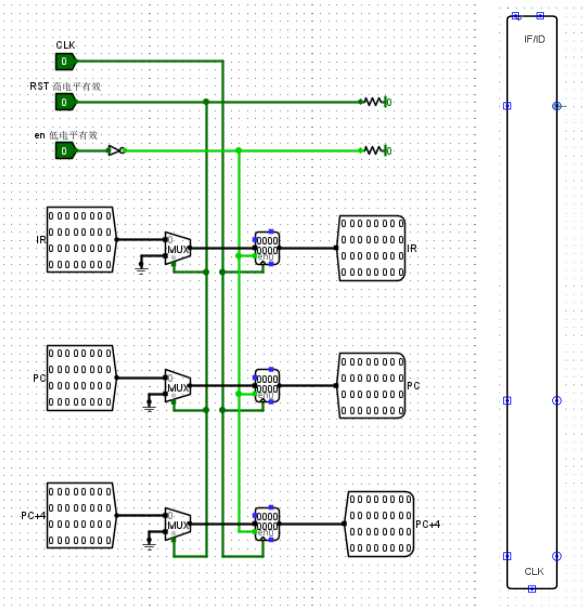

图 3.10 流水接口部件实现

3.3.2 理想流水线实现

理想流水线没有分支冲突和数据冲突的处理，因此只要完成各个控制信号的传递。在布线时需要不同段数据的正确性，如寄存器写入时的相关信号和数据。

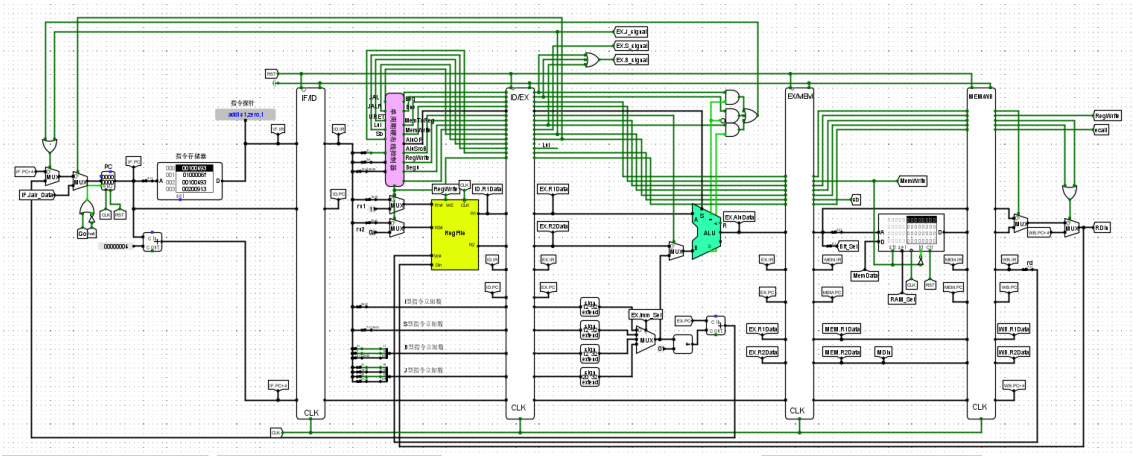

图 3.11 理想流水线实现

3.4 气泡式流水线实现

根据 2.4.2 节 DataHazzard 公式实现数据相关检测逻辑，如图 3.12 所示。其中 r1used 和 r2used 信号的获取借助硬布线控制器表达式自动生成表，然后使用 Logisim 组合逻辑电路分析自动生成电路。在 ID 段插入数据相关检测逻辑。根据 2.4.2 节添加数据相关冲突处理，通过对接口部件和 PC 寄存器输入清零信号和使能信号来实现

气泡的插入。

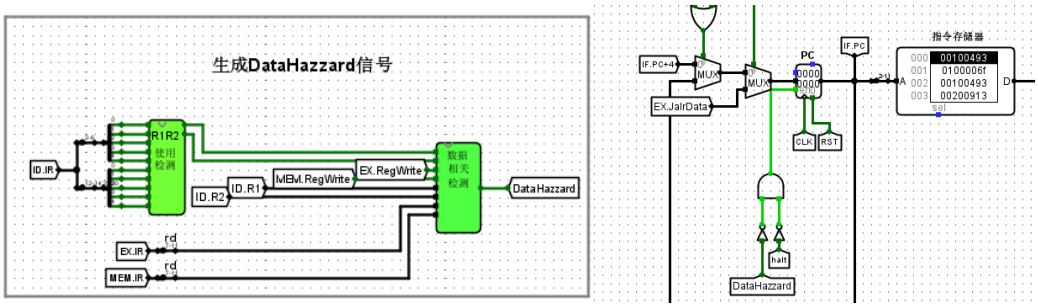

图 3.12 生成 DataHazzard 信号

3.5 重定向流水线实现

在实现重定向逻辑时，只需要在气泡逻辑的基础上添加 LoadUse 相关的检测、Fwd1 和 Fwd2 信号的生成，并更改相关信号的逻辑。具体实现如图 3.13 所示。

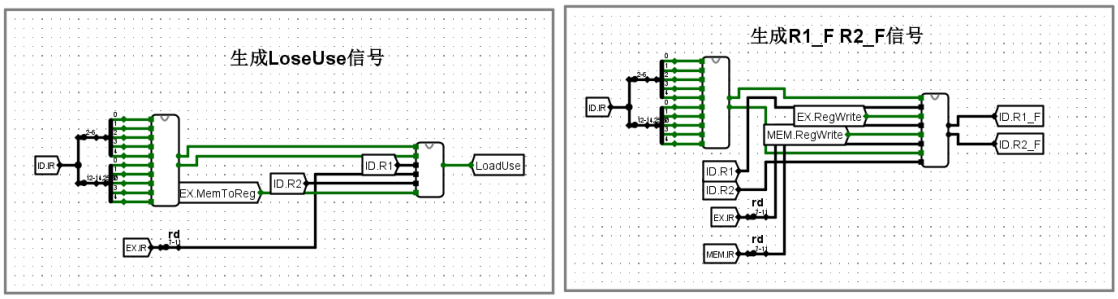

图 3.13 重定向逻辑实现

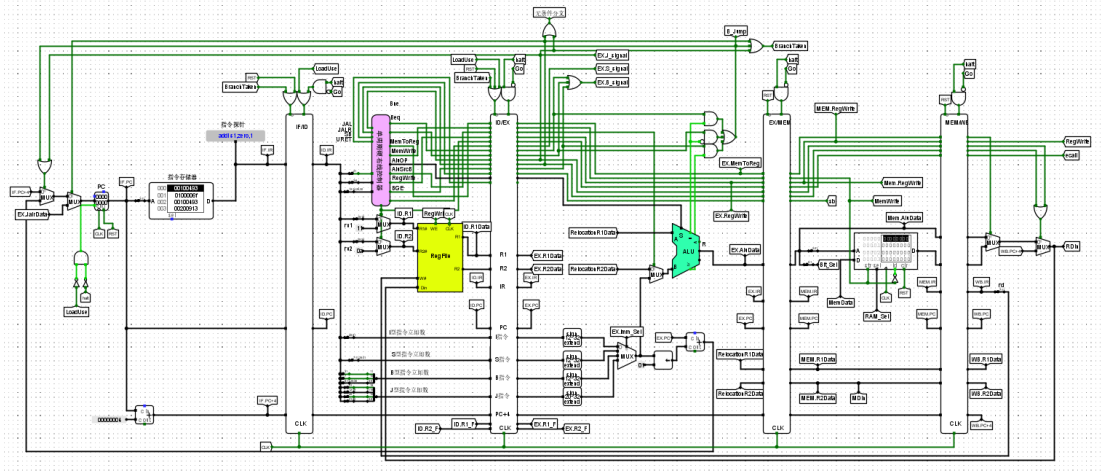

图 3.14 重定向流水线实现

4 实验过程与调试

4.1 测试用例和功能测试

4.1.1 CCAB 指令测试

1）SRA 指令测试

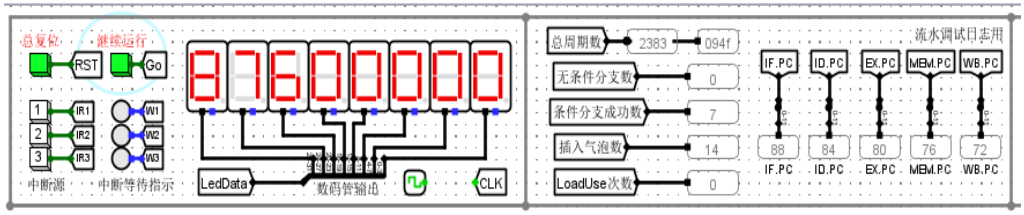

图 4.1 SRA 指令测试结果

2）XOR 指令测试

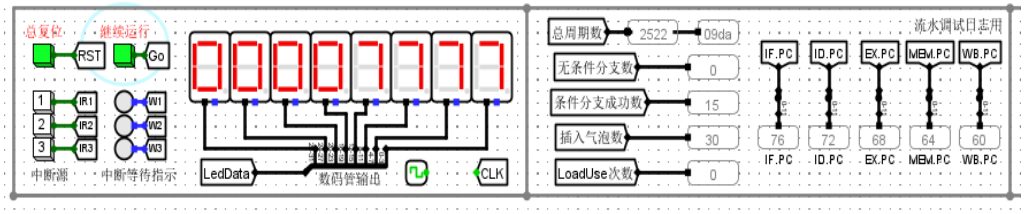

图 4.2 XOR 指令测试结果

3）SB 指令测试

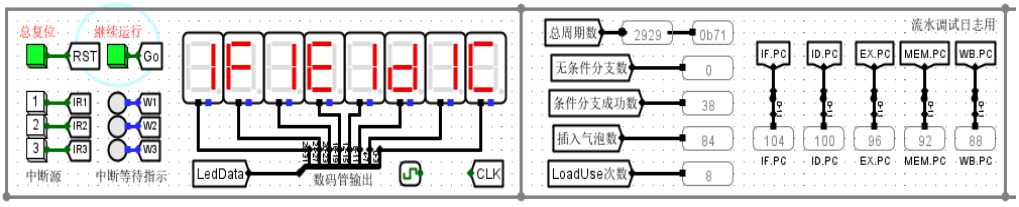

图 4.3 SB 指令测试结果

4）BGEU 指令测试

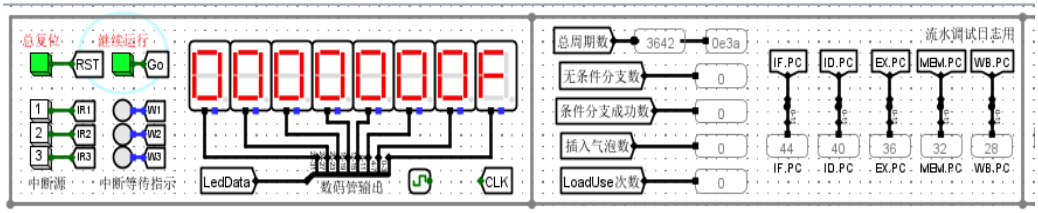

图 4.4 BGEU 指令测试结果

4.1.2 流水线测试

1）气泡流水线测试

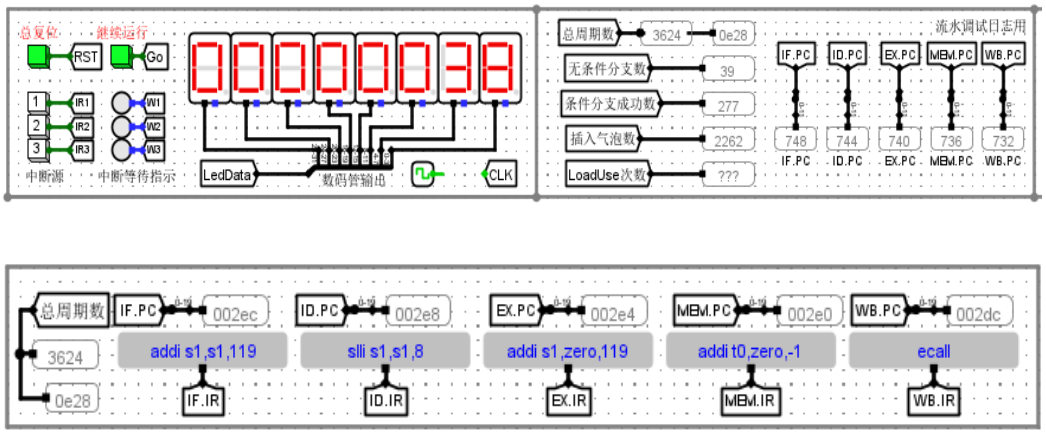

图 4.5 气泡流水线测试结果

2）重定向流水线测试

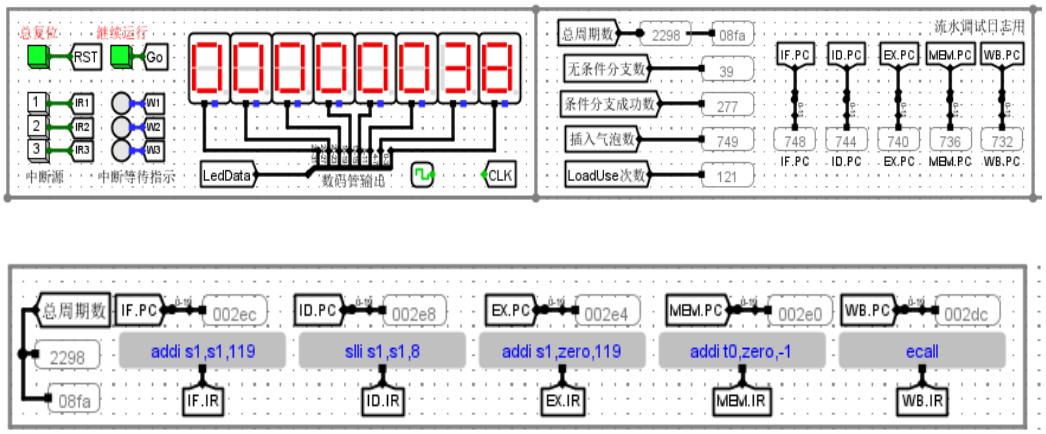

图 4.6 重定向流水线测试结果

4.2 性能分析

单周期 CPU：1546 周

气泡流水线：3624 周

重定向流水线：2298 周

单周期 CPU 硬布线的时钟周期数最短，但是每条指令的平均执行时间最长，效率最差。气泡流水线的时钟周期数最多，这是因为处理数据冲突和分支冲突都使用插入旗袍的方式导致大量的气泡被插入。重定向流水线的时钟周期数比单周期 CPU 多，

但比气泡流水线的少。因为重定向流水线采用的数据冲突的处理方式减少了插入的旗袍数量，减少了周期。

4.3 主要故障与调试

4.3.1 接口部件同步清零故障

气泡流水线： 接口插入气泡问题。

故障现象：在流水接口部件通过同步清零的方式插入气泡时，数据寄存器不能正确同步清零。

原因分析：在接口内的数据寄存器的清零端接入时钟信号和清零信号的与信号。但这样无法真正的达到时钟同步，还会导致新一个指令也被清除。

解决方案：选择多路选择器，在收到清零信号的时候对数据寄存器输入 0，作为伪清零方式。

4.4 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。
第三天	学习流水线相关知识。
第四天	正在实现流水线接口部件。
第五天	完成流水线接口部件的实现。
第六天	完成理想流水线，进行测试与调试。
第七天	学习气泡流水线，并实现。
第八天	完成并测试与调试气泡流水线电路。
第九天	完成并对重定向流水线进行测试与调试。
第十天	完成并对中断机制电路进行测试与调试。

5 设计总结与心得

5.1 课设总结

本次课程设计主要实现了如下：

1） 完成方案总结：设计了单周期 CPU、理想流水线 CPU、气泡流水线 CPU、重定向流水线 CPU、支持单级及多级中断的单周期 CPU。

2） 功能总结：单周期 CPU、气泡流水线 CPU、重定向流水线 CPU、支持动态分支预测机制的重定向流水线 CPU 都实现了包括 CCAB 指令的指令集；气泡流水线通过插入气泡的方式实现了处理数据冲突和分支冲突；重定向流水线通过重定向数据的方式减少了插入气泡数。

5.2 课设心得

硬件综合训练真的是一个非常具有挑战性的课程，难度真的很大。对于喜欢挑战的同学可能是一个快乐的冒险，但是对于我这种基础不好且能力不好的同学来说真的是在地狱里排队受刑了。但我知道这门课存在的重要性，通过这门课也对上个学期计组的课程内容有了更深入的了解。

虽然这门课我的完成度没有那么高，但我能保证是我学习到最多的一门课。两周内要完成一个 CPU 的设计真的不容易，语言问题也成了我的一大障碍。在此真的非常感谢在自己也很忙的情况下依旧抽出时间帮我的同学！

第一个实验，也就是单周期 CPU 的设计是相对来说最简单的一个实验，所以完成的速度也比较快。之后的实验难度也越来越大，单说用来理解实验基础知识和题目的时间都是单周期 CPU 设计的几倍。因此我建议老师们可以考虑在合理的情况下稍微延长实验的时间，让一些基础不好的同学可以有更多的时间去完整的学习相关知识，以便能够更好的完成任务。

最后，非常感谢那些在自己也很忙的情况下依旧耐心帮我解释的同学们。虽然没有能力完成实验的所有要求，实验的过程也很是痛苦，但是回头看看自己还是跌跌撞撞完成了实验的基础部分还是非常的有成就感。如果时间更加充裕的情况下，真的会努力挑战更多的内容的！

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] 谭志虎，周军龙，肖亮. 计算机组成原理实验指导与习题解析.北京：人民邮电出版社，2022.

[5] 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.

[6] 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：郑澍频