

华中科技大学

2021

计算机组成原理

• 实验报告 •

专 业： 计算机科学与技术

班 级： CS1904

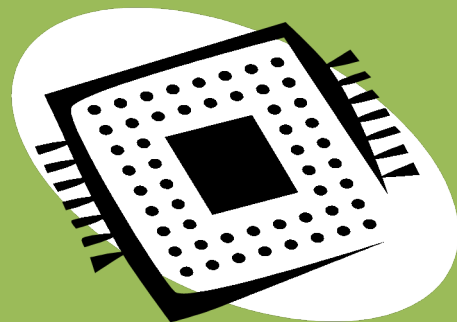
学 号： U201915034

姓 名： 龚宇蒙

电 话： 18672153370

邮 件： 1355130043@qq.com

完成日期： 2021-12-15



计算机科学与技术学院

华中科技大学课程实验报告

目 录

1 CPU 设计实验	2
1.1 设计要求	2
1.2 方案设计	3
1.3 实验步骤	13
1.4 故障与调试	13
1.5 测试与分析	14
2 总结与心得	16
2.1 实验总结	16
2.2 实验心得	16
参考文献	18

1.1 设计要求

需要自己完成设计三级时序的硬布线控制器，其中包括以下内容：

- ### (5) 三级时序硬布线控制器.

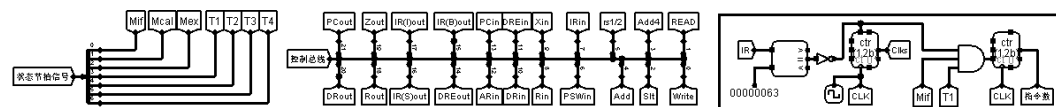


图 1.1 单总线三级时序变长指令总体设计结构图

1.1.2 单总线现代时序中断机制实现

需要我们完成微程序控制器设计，其中包括以下内容:(控制器支持中断操作)

- (1) 指令译码器.
- (2) 支持中断的微程序入口查找逻辑.
- (3) 支持中断的微程序条件判别逻辑.
- (4) 微程序地址转移逻辑.
- (5) 微程序指令存储器.

此外，我们还需要完成硬布线控制器的设计，其中包括以下内容：

- (1) 时序产生器(有限状态机设计).
- (2) 硬布线控制器.

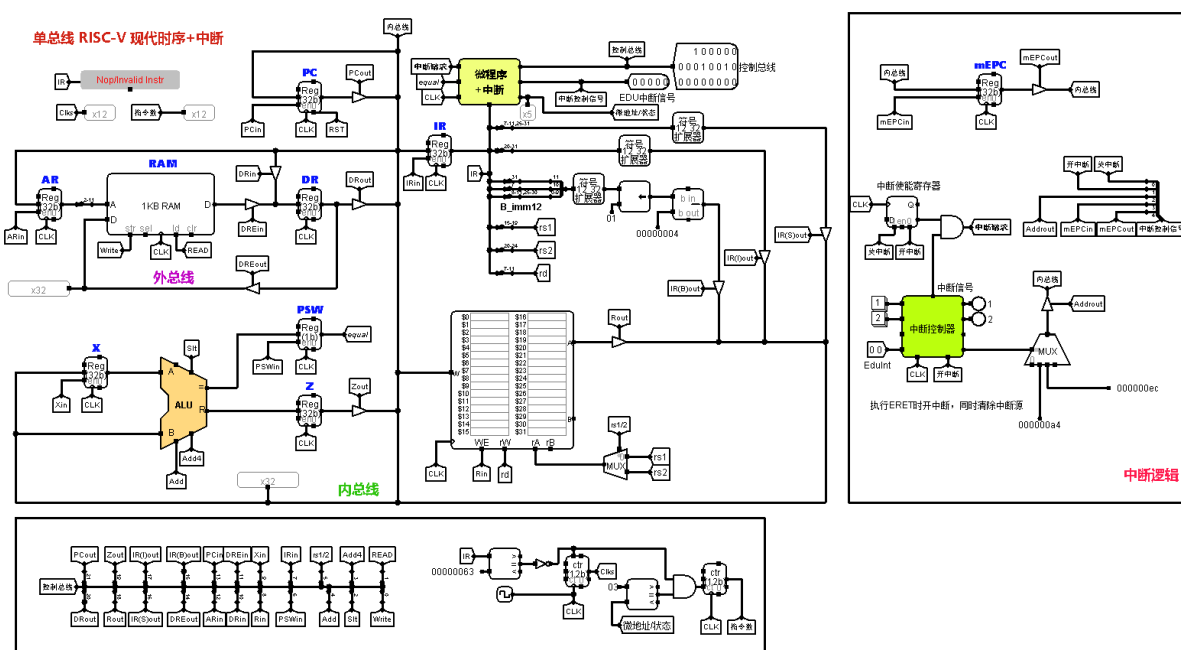


图 1.2 单总线结构现代时序设计结构图

1.2 方案设计

1.2.1 单总线三级时序变长指令 CPU

※指令译码器

输入：32 位指令寄存器 IR 内容.

华中科技大学课程实验报告

输出：当前指令为 LW 指令时 LW 信号输出为 1，当前指令为 SW 指令时 SW 信号输出为 1，当前指令为 BEQ 指令时 BEQ 信号输出为 1，当前指令为 SLT 时 SLT 信号输出为 1，当前指令为 ADDI 时 ADDI 信号输出为 1。

实现逻辑：根据 RISC-V 指令表，查出这五条指令的不同。可以发现利用 32 位 IR 输入当中的 2-6(opcode)和 12-14(func3)位的不同来比对查找对应指令。这里采用比较器比较的方式，若和上述五条指令的结果都不同则属于其他指令。

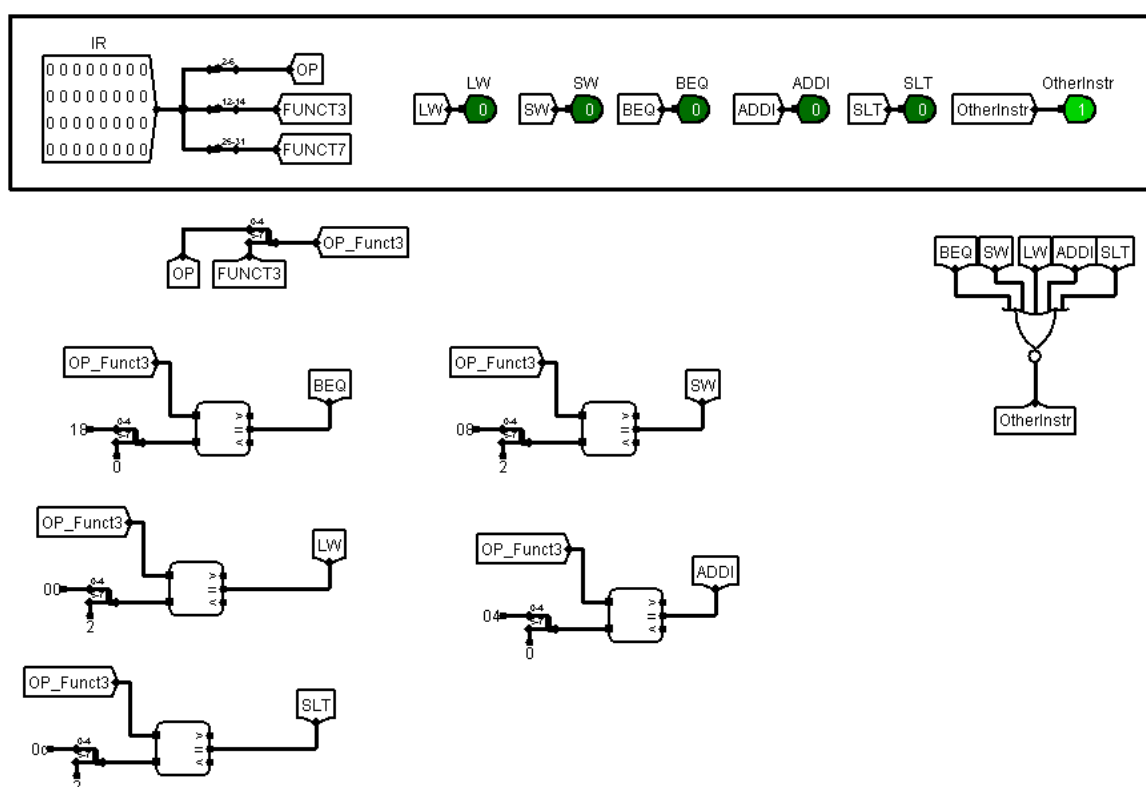


图 1.3 单总线结构三级时序指令译码器设计结构图

※硬布线时序状态机

输入：当前所处状态.

输出：次态.

实现逻辑：可以很明显的发现，我们只有在取址周期结束，且输入指令信号为 LW 或 SW 或 BEQ 时才会进入计算周期。而很明显指令 ADDI 以及 SLT 没有计算过程，因此输入为这两个信号时直接由取址周期变成执行周期。当然所有的执行周期结束时，都需要返回到取址周期。因此状态 8 结束后自动回到状态 0。并且在同一个机器周期内状态自动切换。取址周期包含四个节拍，计算周期包含两个节拍，执行周期包含三个节

华中科技大学课程实验报告

拍。因此我们只需要将相应的状态变化填入表中，自动生成电路逻辑表达式，将表达式加入电路中自动生成电路即可。

当前状态(现态)					输入信号							下一状态 (次态)				
S3	S2	S1	S0	现态 10进制	LW	SW	BEQ	SLT	ADDI	ERET	IntR	次态 10进制	N3	N2	N1	N0
0	0	0	0	0								1	0	0	0	1
0	0	0	1	1								2	0	0	1	0
0	0	1	0	2								3	0	0	1	1
				3								4	0	1	0	0
0	0	1	1	3	1							4	0	1	0	0
0	0	1	1	3		1						4	0	1	0	0
0	0	1	1	3			1					4	0	1	0	0
0	0	1	1	3				1				6	0	1	1	0
0	0	1	1	3					1			6	0	1	1	0
0	1	0	0	4								5	0	1	0	1
0	1	0	1	5								6	0	1	1	0
0	1	1	0	6								7	0	1	1	1
0	1	1	1	7								8	1	0	0	0
1	0	0	0	8								0	0	0	0	0

图 1.4 单总线结构三级时序硬布线时序发生器状态变化表

※硬布线时序发生输出函数生成器

输入：当前状态.

输出：当前 FSM 状态所处的机器周期以及节拍电位.

实现逻辑：根据前一个 FSM 状态机变化我们可知，取址、计算、执行三个机器周期分别有 4、2、3 个节拍电位，每个节拍电位都含有一个时钟周期。因此我们只需要将当前状态对应的机器周期和节拍电位填入表中即可，利用表中组合逻辑自动生成电路。

当前状态(现态)					输出							
S3	S2	S1	S0	现态 10进制	Mif	Mcal	Mex	Mint	T1	T2	T3	T4
0	0	0	0	0	1				1			
0	0	0	1	1	1					1		
0	0	1	0	2	1						1	
0	0	1	1	3	1							1
0	1	0	0	4		1			1			
0	1	0	1	5		1				1		
0	1	1	0	6			1		1			
0	1	1	1	7			1			1		
1	0	0	0	8			1				1	

图 1.5 单总线结构三级时序硬布线时序发生器输出函数逻辑表

华中科技大学课程实验报告

※硬布线组合逻辑输出组件

输入：当前状态的所在的机器周期以及节拍电位，以及反馈信号和指令译码信号。

输出：对应各种控制信号的序列。

实现逻辑：根据五条 RSIC-V 指令的具体执行过程，将产生的控制信号填入对应的表格之中，之后在 logism 组合生成器中生成对应的电路。

五条指令具体执行中对应 RTL 如下图：

#	指令	汇编代码	指令类型	RTL 功能说明
1	lw	lw rd,imm(rs1)	I 型	$R[rd] \leftarrow M[R[rs1] + \text{SignExt}(imm)]$
2	sw	sw rs2,imm(rs1)	S 型	$M[R[rs1] + \text{SignExt}(imm)] \leftarrow R[rs2]$
3	beq	beq rs1,rs2,imm	B 型	$\text{if}(R[rs1] == R[rs2]) \quad PC \leftarrow PC + \text{SignExt}(imm) \ll 1$
4	addi	addi rd,rs1,imm	I 型	$R[rd] \leftarrow R[rs1] + \text{SignExt}(imm)$
5	add	add rd,rs1,rs2	R 型	$R[rd] \leftarrow R[rs1] + R[rs2]$
6	slt	slt rd,rs1,rs2	R 型	$\text{If}(rs1 < rs2) \quad R[rd] \leftarrow 1 \text{ else } R[rd] \leftarrow 0$
7	j	j imm26	J 型	$PC \leftarrow \{(PC+4)_{31:28}, \text{imm26} \ll 2\}$

图 1.6 五条指令的 RTL 功能说明

根据指令的 RTL,我们推出了每一步的控制信号的序列。将其填入给出的 Excel 表。

输入 (填1或0, 不填为无关x)														输出 (只填写为1的情况)																						
Mif	Mcal	Mex	Mint	T1	T2	T3	T4	LW	SW	BEQ	SLT	ADDI	ADD	PCout	DRout	Zout	Rout	IRout	IRSeout	IRBseout	DRSeout	PCin	ARin	DRin	DRin	Xin	Rin	IRin	PSWin	rs1/rs2	Add	Add4	Slr	READ	WRITE	
1				1										1									1				1									
1					1																													1		
1						1											1						1		1											1
1							1								1															1						
	1			1				1										1									1									
	1				1				1										1																1	
	1					1				1											1															1
	1			1							1												1													
	1				1							1																								
	1					1							1											1												
	1						1							1																						
	1							1							1																					
	1								1							1																				
	1									1							1																			
	1										1							1																		
	1											1							1																	
	1												1							1																
	1													1							1															
	1														1							1														
	1															1																				
	1																1																			
	1																	1																		
	1																		1																	
	1																			1																
	1																				1															
	1																					1														
	1																						1													
	1																							1												
	1																								1											
	1																									1										
	1																										1									
	1																											1								
	1																												1							
	1																													1						
	1																														1					
	1																															1				
	1																																1			
	1																																	1		
	1																																		1	
	1																																			1

图 1.7 单总线结构三级时序控制器控制信号逻辑自动生成表

※硬布线控制器

输入：指令寄存器内容 IR，时钟信号以及反馈信号 EQUAL。

输出：控制总线信号和状态节拍信号。

华中科技大学课程实验报告

实现逻辑：根据课堂所学可知，寄存器 IR 中指令字首先通过指令译码器产生指令译码信号，状态寄存器输入为 FSM 次态信号，输出为现态信号，现态信号经过状态机转换器变成次态信号输入状态寄存器。现态信号作为输出函数的输入与输出函数直接相连，之后输出函数将产生时钟周期以及节拍电位等状态节拍信号。硬布线控制组合逻辑单元利用这些信号生成控制总线信号。

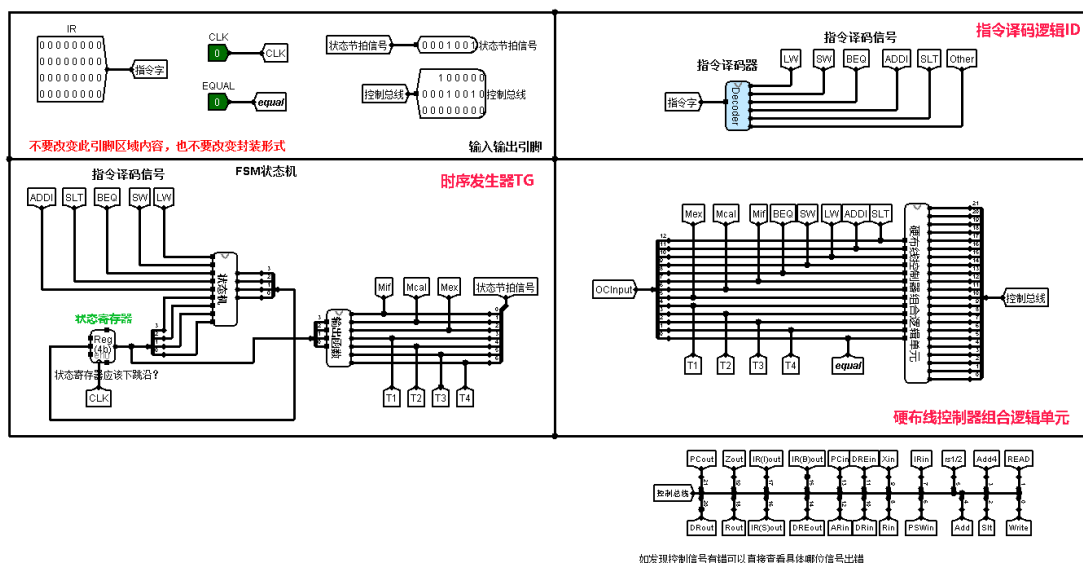


图 1.7 单总线结构三级时序硬布线控制器结构图

1.2.2 单总线现代时序中断机制实现

※指令译码器

实现逻辑与 1.2.1 指令译码器实现逻辑相同，此处省略。

※支持中断的微程序入口查找逻辑

输入：6 个指令译码选择信号，分别代表不同的译码指令。

输出：微程序地址入口地址. $[S_4S_3S_2S_1S_0]$

实现逻辑：可以很明显的发现，输入指令译码信号为 LW 时，会跳转到状态 4,而输入指令译码信号为 SW 时，会跳转到状态 9，输入指令译码信号为 BEQ 时，会跳转到状态 14，输入指令译码信号为 SLT 时，会跳转到状态 19，输入为 ADDI 时，会跳转到状态 22，输入为 ERET 时，会跳转到状态 25。根据这些状态的变迁将对应的值填入给出的 Excel 表中来构造组合逻辑,再利用组合逻辑生成器在 logism 中自动生成电路。

机器指令译码信号						微程序入口地址					
LW	SW	BEQ	SLT	ADDI	ERET	入口地址 10进制	S4	S3	S2	S1	S0
1						4	0	0	1	0	0
	1					9	0	1	0	0	1
		1				14	0	1	1	1	0
			1			19	1	0	0	1	1
				1		22	1	0	1	1	0
					1	25	1	1	0	0	1

图 1.8 单总线现代时序(带中断)微程序入口查找组合逻辑生成表项图

利用如上的表生成的电路如下：

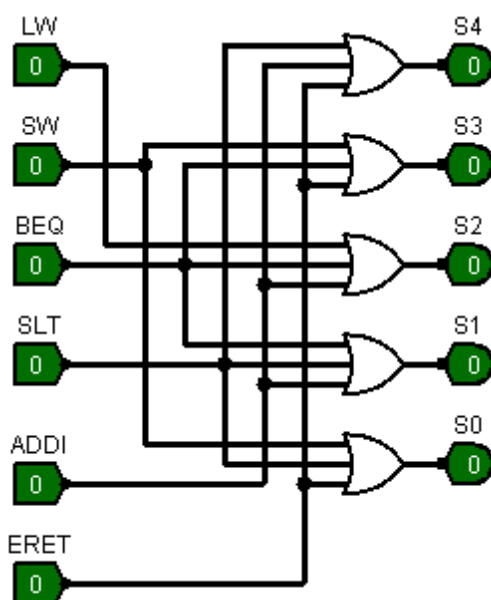


图 1.8 单总线现代时序(带中断)微程序入口查找组合逻辑总体结构图

※支持中断的微程序条件判别逻辑

输入：微程序流程控制信号 $[P_2P_1P_0]$ 和条件状态位 equal 以及中断判断位 InterR。

输出：微程序地址转移控制信号 $[S = S_2S_1S_0]$ 。结果为 0 说明下一条地址(计数器默认+1)，结果为 1 说明取入口地址，即取地址转移逻辑的输出，结果为 2 说明取 BEQ 指令且当前运算是相等的跳转地址。结果为 3 说明是中断处理程序的第一条微程序地址。结果为 4 说明回到取址微程序的第一条微程序地址。

华中科技大学课程实验报告

实现逻辑：根据输出的实现逻辑，将对应的输入和输出结果填入 Excel 表中构造组合逻辑，之后利用 logism 组合逻辑生成对应电路。

输入 (填1或0, 不填为无关项x)							
P0	P1	P2	equal	IntR	S2	S1	S0
0	0	0	0	0	0	0	0
1					0	0	1
0	1	1	1		0	1	0
0	1	1	0	1	0	1	1
0	1	1	0	0	1	0	0
0	0	1		1	0	1	1
0	0	1		0	1	0	0
0	1	0	0		1	0	0
0	1		1		0	1	0

图 1.9 单总线现代时序(带中断)微程序条件判别逻辑组合逻辑设计表

※控制存储器

输入：微程序地址。

输出：30 位微程序微指令。

实现逻辑：带中断的现代时序微程序位指令设计和不带中断的差不多。这里唯一的不同点在于不再使用下址地址而是使用计数器方式，此外加上了中断微程序。微程序的地址对应硬布线的时序，我们只需要对着硬布线中每个节拍电位的控制信号输出来填写现代时序的控制信号输出即可。之后根据状态机变迁的关系设计控制信号 P。

设计控制信号需要遵循下面几条规则：

- 1.如果当前微程序是取址指令的最后一条微程序时，需要标记 P_0 转入入口查询逻辑寻找入口地址。
- 2.如果当前微指令时判断 equal 的分支时，这个时候标记 P_1 可能需要跳转到处理 equal 分支的微程序。
- 3.如果当前微程序时此条指令最后一条微程序时，标记 P_2 代表此条指令结束，转入到微指令地址入口出进行取址微程序。

华中科技大学课程实验报告

微指令功能	PCout	DRout	Zout	Rout	IRInout	PCIn	ARin	DRIn	Xin	Rin	IRin	PSWin	rs1/2	Add	Add4	St	READ	WRITE	EPCout	EPCIn	Address	STI	CLI	P0	P1	P2	微指令	微指令十六进制
取指令	0	1					1			1																10000000100100000000000000000000	20120000	
取指令	1														1											00000000000000000000000000000000	800	
取指令	2			1			1	1									1									00100000101000000000001000000000	8280200	
取指令	3	1									1												1			01000000000000001000000000000100	10008004	
LW	4			1						1																00010000000010000000000000000000	4020000	
LW	5				1										1											00001000000000000010000000000000	2001000	
LW	6			1				1																		00100000010000000000000000000000	8100000	
LW	7							1									1									00000000010000000001000000000000	80200	
LW	8	1									1												1			01000000000010000000000000000000	10010001	
SW	9			1						1																00010000000010000000000000000000	4020000	
SW	10				1												1									00000100000000000100000000000000	1001000	
SW	11			1				1																		00100000010000000000000000000000	8100000	
SW	12			1						1					1											00010000000100001000000000000000	4042000	
SW	13							1									1						1			00000010000000000001000000000000	400101	
BEQ	14			1						1																00010000000010000000000000000000	4020000	
BEQ	15			1									1	1										1	1	001000000000001100000000000011	4006003	
BEQ	16	1								1																10000000000100000000000000000000	20020000	
BEQ	17					1											1									00000010000000000100000000000000	801000	
BEQ	18			1			1																		1	00100000010000000000000000000000	8200001	
SLT	19				1					1																00010000000010000000000000000000	4020000	
SLT	20			1											1			1								00010000000000001001000000000000	4002400	
SLT	21			1							1														1	00100000000001000000000000000000	8010001	
ADDI	22				1					1																00010000000010000000000000000000	4020000	
ADDI	23				1										1											00001000000000000010000000000000	2001000	
ADDI	24			1							1														1	00100000000001000000000000000000	8010001	
ERET	25						1											1			1				1	00000001000000000000010010001	200091	
中断响应	26	1																		1			1			10000000000000000000000001001000	20000048	
中断响应	27						1														1				1	0000000100000000000000000100001	200021	

图 1.10 单总线现代时序(带中断)微程序微指令自动生成表项图

得到了各条微指令的 16 进制值后，将最右边一列微指令复制输入到微程序控制器(带中断)的微指令寄存器中。

※微程序控制器

输入：指令字 IR, 时钟信号 CLK, 中断信号 InterR, 条件判断信号 Equal.

输出：控制总线输出和中断控制信号.

实现逻辑：现代时序微程序(带中断控制)的所有组件已在上面实现，现在只需要将组件进行连接即可。将六条指令和入口查找的对应输入相连，其输出为入口地址，和多路选择器的 1 号位置相连。多路选择器输出与计数寄存器相连，计数寄存器存储当前微指令在控制存储器中的地址。正常情况计数器的输出和加法器相连，实现自增加 1，之后送回多路选择器 0 号位置作为顺序地址的下一次取址地址。当然为了跳转到 BEQ 的分支，需要将 BEQ 分支微指令的第一条地址值和多路选择器 2 号位置相连。同时为了实现中断指令，需要将中断入口微指令的地址和多路选择器的 3 号位置相连。同时，我们要求一条指令结束后能够回到取址阶段，因此需要将控制存储器 0 号地址和多路选择器的 4 号位置相连。当然如何跳转到多路选择器的对应位置，需要判别测试逻辑的输出作为多路选择器的选择端控制。这样，微程序的控制的连接便已经完成。现代时序带中断的 CPU 设计大部分已经完成。现在只需要在 CPU 主界面完成中断控制选项即可。

华中科技大学课程实验报告

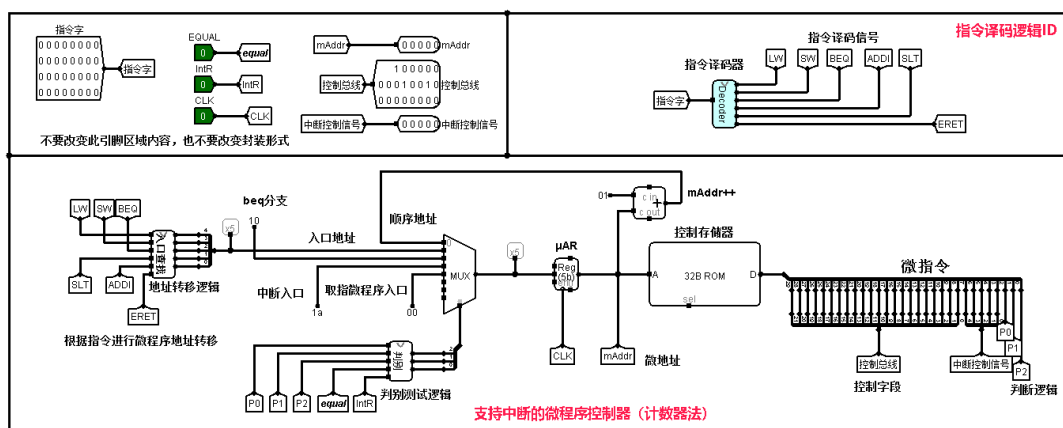


图 1.11 单总线现代时序(带中断)微程序控制器总体结构

※中断逻辑

mEPC 寄存器：和数据通路的其他锁存器是同一种构造模式，在写入信号为 0 时，通过使能端控制寄存器忽略时钟信号，不把输入端内容锁存，锁存器输出接一个三态门，三态门只有当 EPC 输出信号有效时才能进行输出，不能进行输出的时候三态门阻挡。

对于带中断的程序，从主程序在第 42 行结束，从第 43 行开始是中断程序，接着找到中断程序的第一条指令在第 43 行，我们接着之后在第 61 行找到此条指令，说明这时第二个中断程序的第一条微指令。所以中断入口指令分别是第 41 条指令和第 59 条指令。可知一条指令占用 4 个字节，因此我们可以知道中断指令的地址。分析出来是 0x30a4 和 0x30ec。

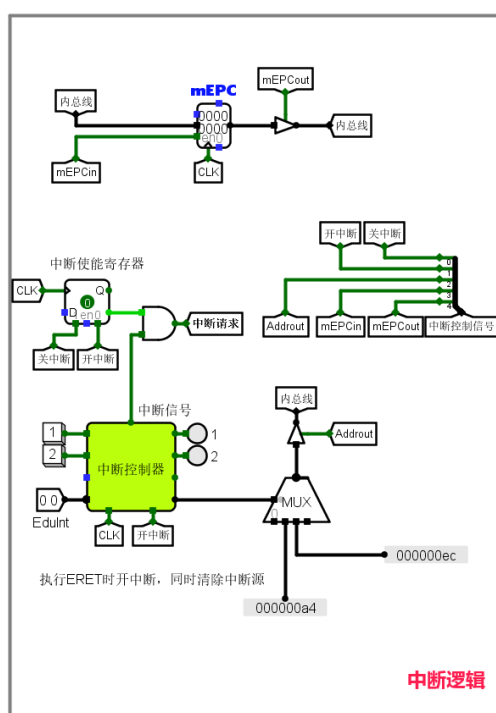


图 1.12 单总线现代时序中断逻辑总体结构图

※硬布线控制器

输入：指令字 IR, 时钟信号 CLK, 中断信号 InterR, 条件判断信号 Equal.

输出：控制总线输出和中断控制信号。

实现逻辑：在硬布线状态机逻辑中我们生成了相应的状态机变化逻辑。这里只需要将各条指令输入和带中断硬布线状态机逻辑相连，将其输出送入计数寄存器，同时将计数寄存器的输出送入带中断硬布线状态机逻辑即可。

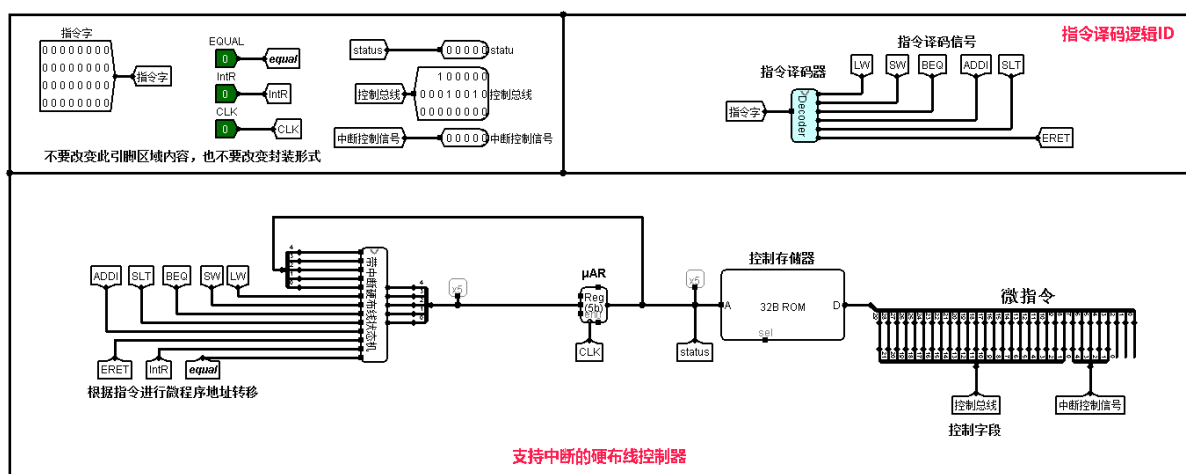


图 1.13 单总线现代时序硬布线控制器总体结构

1.3 实验步骤

- (1) 在 logism 中设计各个部件的实现逻辑结构.
- (2) 在 logism 上运行一遍电路, 测试电路是否正常
- (3) 将得到的电路文件输入到 educoder 上测试.

1.4 故障与调试

1.4.1 控制器输出结果与实际不符

故障现象: 三级时序控制器各组件按正常顺序连接后, educoder 测试结果出现大片的不相符的情况。

原因分析: 计数器的时钟调变为上升沿, 导致状态跳变过程和各寄存器锁存处在同一个时钟上升沿。没法维持稳定的 Setup 时间和 Hold 时间导致实际输出不符。

解决方案: 将计数器的上升沿触发改成下降沿触发即可, 需要满足锁存时间和各寄存器的锁存维持互斥, 才能保证各寄存器能正确锁存。

1.4.2 在现代时序的控制器测试当中, 始终会出现测试结果有两项没法通过 educoder

故障现象: 现代时序各组件按正常顺序连接后, educoder 测试结果出现大片的不相符的情况。

原因分析: 查看了微指令表项之后, 发现 BEQ 的跳转分支指令处判别位 P 忘记填了, 导致 BEQ 后续没有跳转到正常分支而是直接接着取了取址微程序第一条微指令地址。

解决方案: 将微指令表项的 BEQ 跳转分支微指令的 p_1 填 1, 之后把更新之后的各微指令新值存入控制寄存器当中。

1.4.3 带中断的 CPU 无法触发中断过程

故障现象: 中断 CPU 正常完成各项连接之后, 却无法触发中断。

原因分析: 查看了中断逻辑, 发现中断控制器的输出接线位置错误, 本来要接在 Intno 处却接在了 IntR 处。

解决方案：将连线改正即可。

1.5 测试与分析

1.5.1 Logism CPU 测试结果（三级时序测试）

在三级时序 CPU 中 RAM 加载 sort-5-riscv.hex 程序，ctrl+k 自动运行，程序应该运行至 0x81d 节拍停下，指令计数为 251，注意最后一条指令是一条 beq 分支指令，会跳回当前指令继续执行，是死循环。

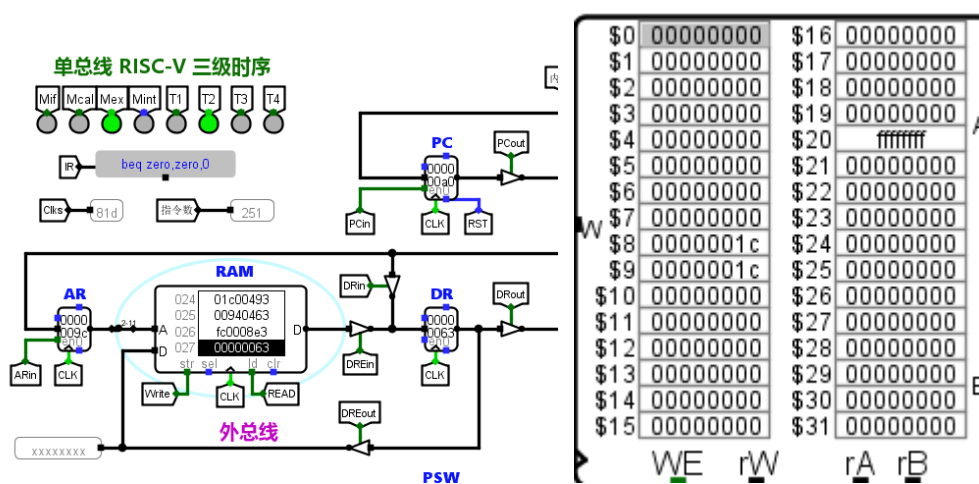


图 1.14 三级时序 CPU 运行排序程序结果图

可以发现和预期结果相同，指令数为 251，最后一条指令为 beq 分支指令，说明程序运行正确。

1.5.2 Logism CPU 测试结果（中断测试）

```
080 00000006 00000005 00000001 00000002 00000003 00000004 00000000 ffffffff C
090 00000002 00000002 00000002 00000002 00000002 00000002 00000002 00000002 C
```

图 1.15 运行 1 号中断的结果 (2 次一号中断)

```
080 00000006 00000005 00000001 00000002 00000003 00000004 00000000 ffffffff
090 00000002 00000002 00000002 00000002 00000002 00000002 00000002 00000002
0a0 fffffffe fffffffe fffffffe fffffffe fffffffe fffffffe fffffffe fffffffe
0b0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

图 1.15 运行 2 号中断的结果 (2 次二号中断)

如按下按键 1，程序将在 90 开始的 8 个字单元全部加 1，执行两次结果正确。如按

华中科技大学课程实验报告

下按键 2，程序将在 a0 开始的 8 个字单元全部减 1，执行两次可以发现结果正确。说明中断程序也是可以正确执行的。

1.5.3 Educoder 平台评测

RISC-V单总线CPU设计(现代时序)(HUST)  **已截止**

[详情](#) 

谭志虎 已开始做题 398 人 未开始做题 29 人 已完成做题 383 人 评阅剩余时间: 0 天 4 小时 1 分

单总线RISC-V CPU设计(变长指令周期3级时序)(HUST)  **已截止**

[详情](#) 

谭志虎 已开始做题 396 人 未开始做题 31 人 已完成做题 382 人 评阅剩余时间: 0 天 4 小时 1 分

RISC-V现代时序中断机制实现(HUST)  **已截止**

[详情](#) 

谭志虎 已开始做题 383 人 未开始做题 44 人 已完成做题 357 人 评阅剩余时间: 0 天 4 小时 1 分

三次实验均已完成全部正常评测，测试全部通过。

2 总结与心得

2.1 实验总结

本次实验主要完成了如下几点工作：

- 1) 设计了对于海明码检验和校正的电路、16 位和 32 位快速加法器的、补码一位乘法器，也设计了 MIPS 寄存器以及 4 路相联 cache，还设计了现代时序以及三级时序的控制器和入口查找逻辑以及硬布线状态机等 CPU 相关组件的电路。此外，还设计了现代时序 CPU 的中断执行电路。
- 2) 实现了对于海明码的检验和校正的功能，也实现了对于 16 位、32 位加法的快速执行以及补码一位的乘法计算。同时也实现了 MIPS 的寄存器功能以及 4 路相联 cache 的 LRU 快速查找以及写入功能。最后也实现了三级时序的 RISC-V 指令的 CPU 功能，以及现代时序的带中断执行的 RISC-V 指令的 CPU 功能。
- 3) 完成了对于数据表示的实践和了解，也完成了对于计算机中 ALU 的构建和存储器的快速读取的实现以及 RISC-V 指令集下 CPU 的执行过程的学习和深入理解。

2.2 实验心得

- 1) 与之前其他课程的实验相比，我认为组原实验应该是我学习至今以来做过最完备、最细致、最适合加深课程理解的实现。他每一次实验都紧紧深入课堂内容，在做实验的时候一方面加深了对于课堂学习的理解，另一方面许多上课没有懂的过程通过实验中一步一步通过对于具体的电路元件的组合以及电路组件的过程中也让我逐渐明白。特别是 CPU 实验，真正的让我明白了 CPU 执行指令时的过程，构建了清晰的印象，也让我在期末复习以及考试中轻松很多。可以说，组员实验是做做过的最好的配套课程实验。
- 2) 当然，今年第一次改做 RISC-V 实验，还有一些实验内容不够完善。比如中断实验和现代时序中对于入口状态逻辑的表格，两个表格输入项目有一些差异，但是却只给了一个现代时序的表格。我当时在做中断实验时就非常苦恼，最

华中科技大学课程实验报告

后用的去年 MIPS 实验的中断表格填写完成的生成组合逻辑。希望在改变实验内容时配套设备也能更加完善，带给同学们更好的学习组原的体验。

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 5 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, 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 年.

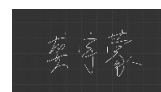
• 指导教师评定意见 •

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：龚宇蒙



二、对课程实验的学术评语（教师填写）

三、对课程实验的评分（教师填写）

评分项目 (分值)	课程目标 1 工具应用 (10 分)	课程目标 2 设计实现 (70 分)	课程目标 3 验收与报告 (20 分)	最终评定 (100 分)
得分				

指导教师签字：_____