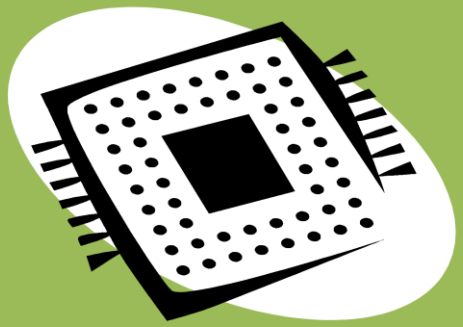
华中科技大学

2017

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	CS1503
学　　号：	U201514559
姓　　名：	周铭昊
电　　话：	15802740273
邮　　件：	630212894@qq.com
完成日期：	2018-1-12



计算机科学与技术学院

目　录

1 数据表示实验

1.1 设计要求

1.1.1 汉字编码

1) 设计国标码转区位码电路

输入：GB2312 16 位国标码；输出：区号，行号（区号行号均从 1 开始计数），下图为电路引脚定义，请在电路中复制隧道连接信号，注意不要增改引脚，不要修改子电路封装，以免影响该子电路在其他电路模块中的调用。

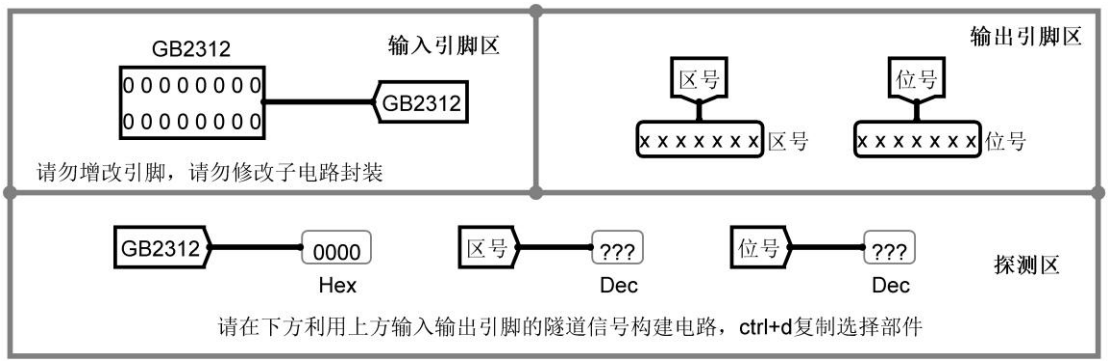


图 1.1 汉字国标码转区位码电路引脚

2) 汉字 GB2312 编码实验

完成国标码到区位码的转换电路后，可以在汉字显示电路中进行测试，尝试在下面电路中的 ROM 存储器中存入 100 个成句的汉字（要求与原始数据不同），ROM 存储器使用方法见 Logisim 参考手册。

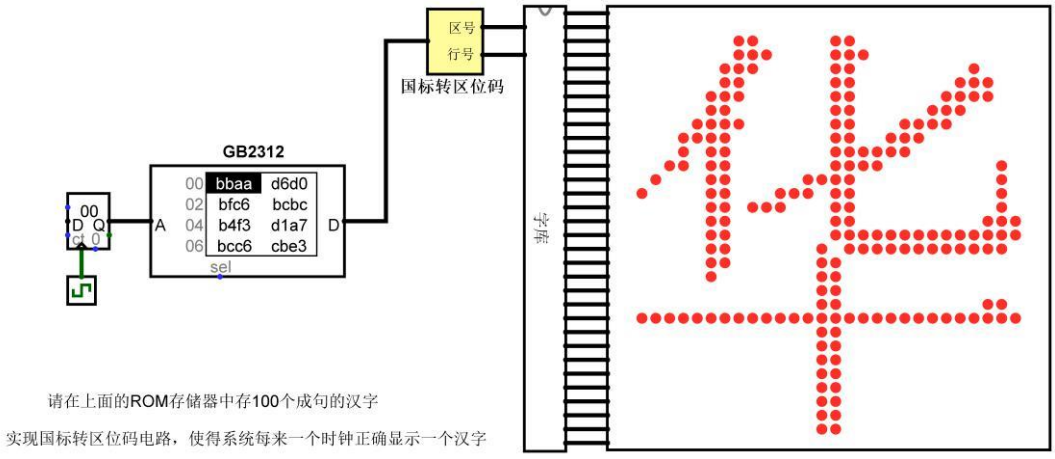


图 1.2 汉字字模码显示电路

1.1.2 偶校验

输入：16 位原始数据；输出：17 位校验码（16 位数据位+1 位校验位），输入输出引脚定义如下：

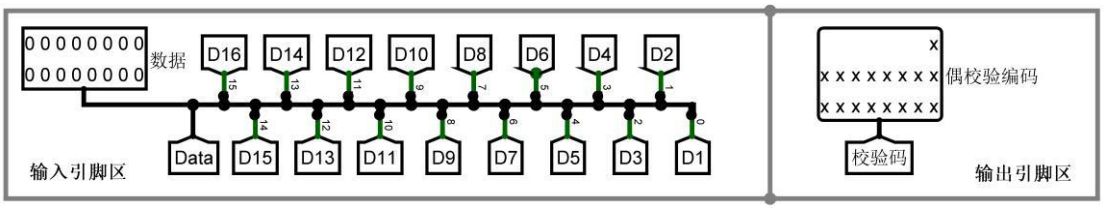


图 1.3 海明编码电路输入输出引脚定义

1） 设计 17 位偶校验编码的检错电路

输入：17 位校验码；输出：16 位原始数据，1 位检错位；输入输出引脚定义如下：

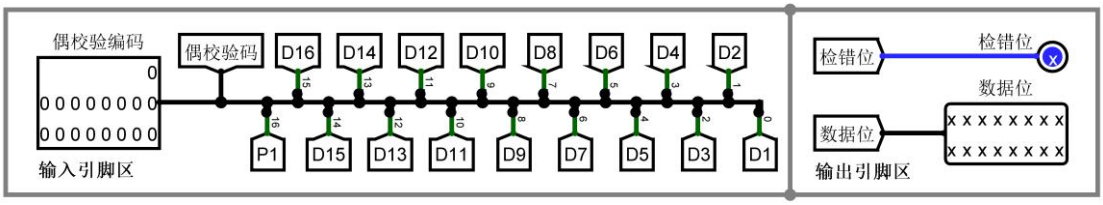


图 1.4 海明解码电路引脚定义

3） 偶校验传输测试

在偶校验传输测试 1 电路中测试偶校验编解码电路的正确性，并观察数据传输过程何时会出现误报情况，分析奇偶校验传输的性能，如果已经实现汉字显示模块，可直接使用偶校验传输测试 2 电路。

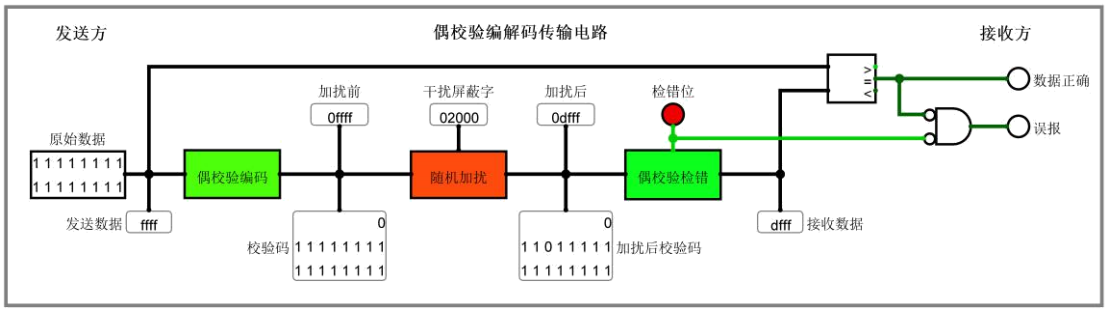


图 1.5 偶校验传输测试电路 1

在偶校验传输测试 2 电路中测试偶校验编解码电路实现是否正确，测该电路引入了汉字显示模块，可以直接显示接收端和发送端的编码的汉字，通过汉字显示可以很直观观察传输是否发生错误，从而观察采用偶校验进行数据传输时传输的可靠性，用户可以使用 ctrl+t 快捷键开启时钟自动仿真测试，具体测试电路如下图所示：

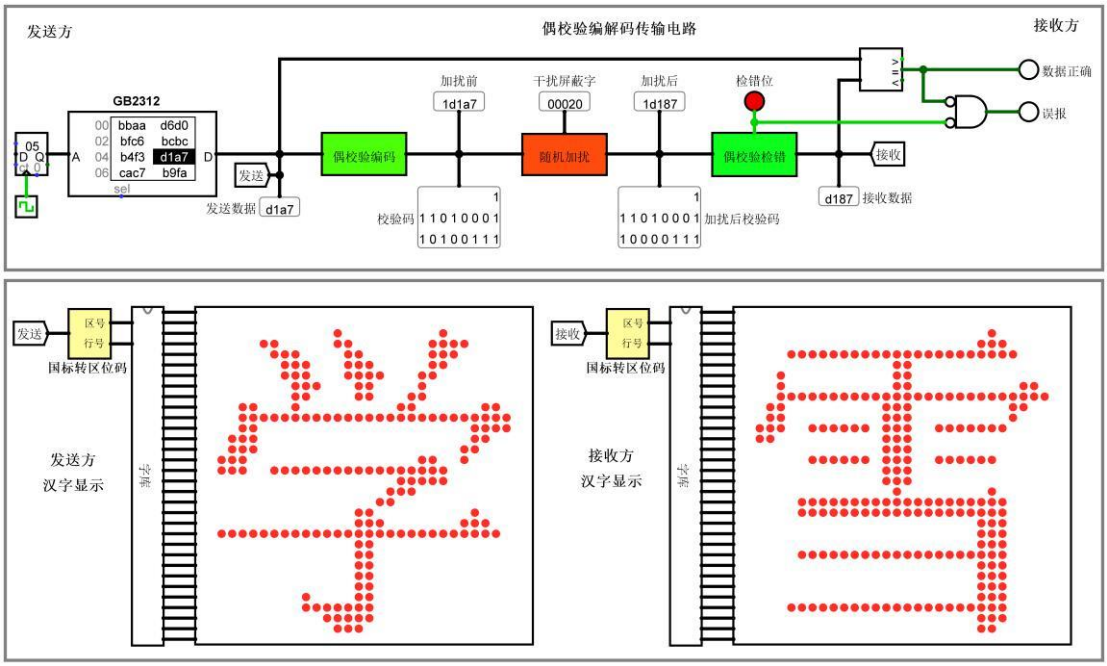


图 1.6 偶校验传输测试电路 2

1.1.3 海明校验

1） 设计 16 位数据编码的海明校验编码电路

输入：16 位原始数据；输出：22 位校验码（16 位数据位+5 位校验位），输入输出引脚定义如下：

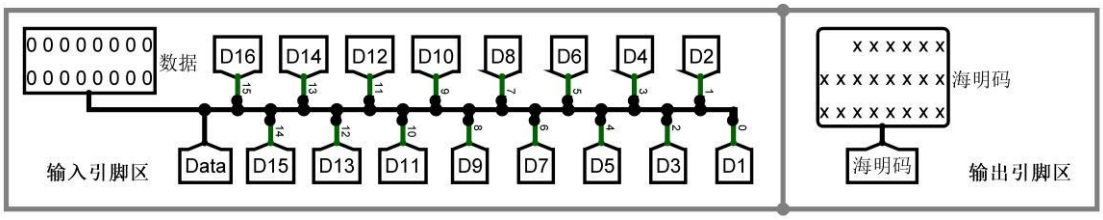


图 1.7 海明编码电路引脚定义

2） 设计 22 位海明校验码的解码电路

输入：22 位校验码；输出：16 位原始数据，1 位检错位；2 位检错位；无错误状态位；输入输出引脚定义如下：

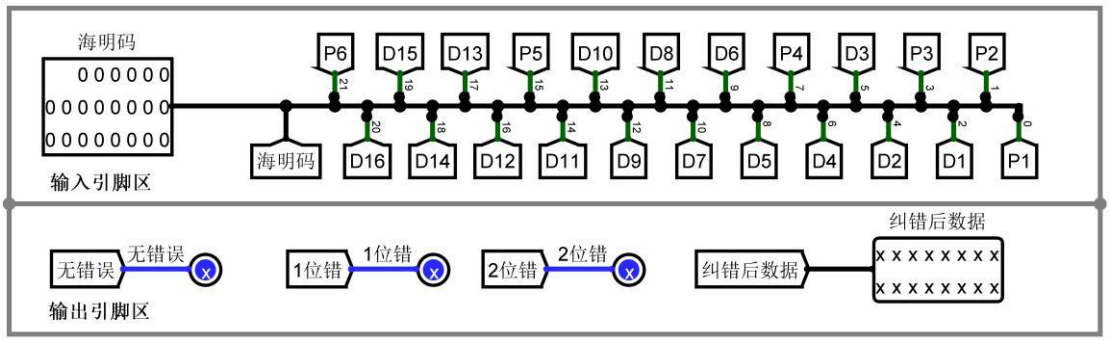


图 1.8 海明解码电路引脚定义

3） 海明校验传输测试

在海明校验传输测试 1 子电路中测试海明校验编解码电路的正确性，注意图中随机干扰电路只能产生最多 2 位错误，具体测试电路如下图所示：

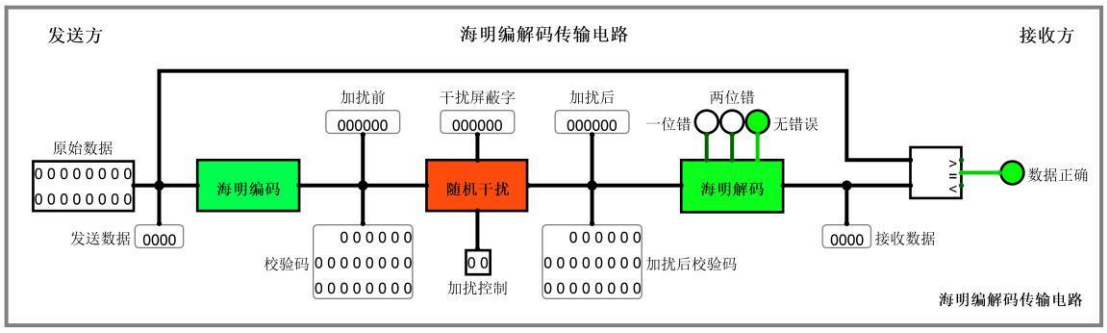


图 1.9 海明编码传输电路 1

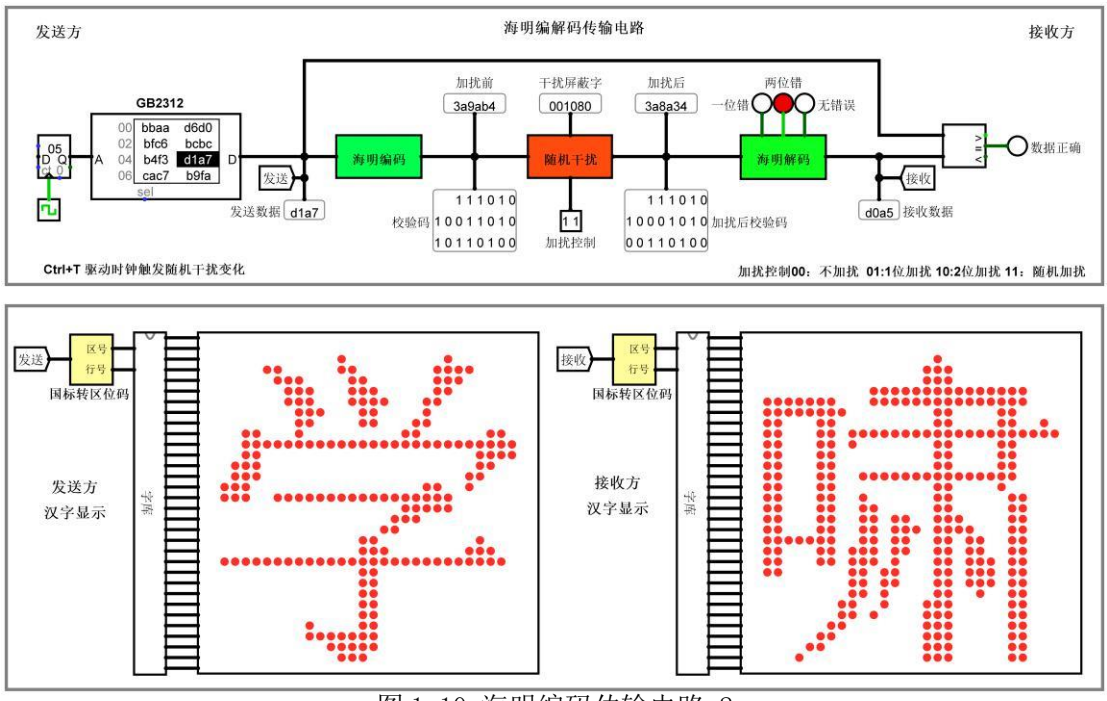


图 1.10 海明编码传输电路 2

在海明校验传输测试 2 子电路中测试海明校验编解码电路的正确性，该电路引入了汉字显示模块，可以直接显示接收端和发送端的编码的汉字，通过汉字显示可以很直观的看出海明纠错的效果，用户可以使用 ctrl+t 快捷键开启时钟自动仿真测试，具体测试电路如下图所示：

4) 海明编码流水传输测试

下图中将海明编码传输过程分成了 5 个阶段（取数，编码，传输，解码，显示）类似 CPU 指令流水线的处理过程，中间蓝色长条为流水接口部件（内部实际是若干寄存器，用于传输数据和控制信号），流水接口部件提供同步清零控制信号，试启用时钟自动仿真运行该电路，观察接收方接受到的信息，当发生两位错时，将会发生错误，尝试简单修改该电路，使得解码阶段出现两位错时，系统能自动重传对应编码（类似指令流水线中的分支跳转），从而使得该电路能正确传输所有数据。

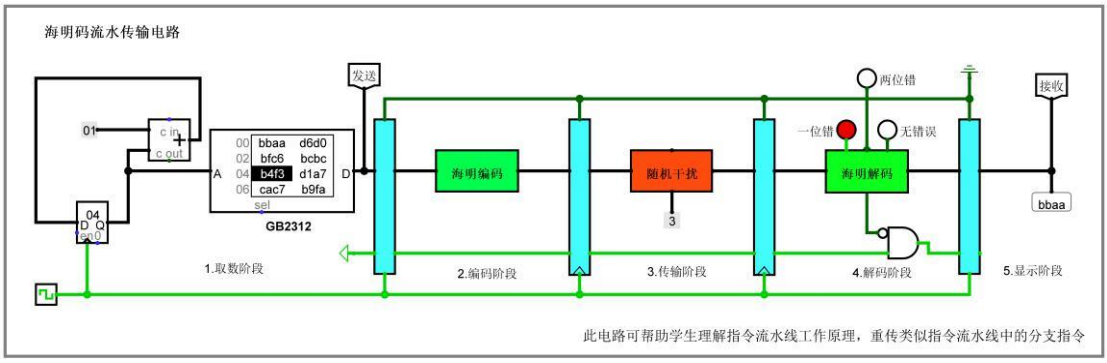


图 1.11 海明编码流水传输电路

1.2 方案设计

1.2.1 国际转区位码

（区位码的十六进制表示）＋2020H＝国标码，因此采用一个减法器，让国际码减去 2020H，得到的结果低六位为位号，高 5 位为区号。具体电路如图 1.12 国际转区位码电路所示

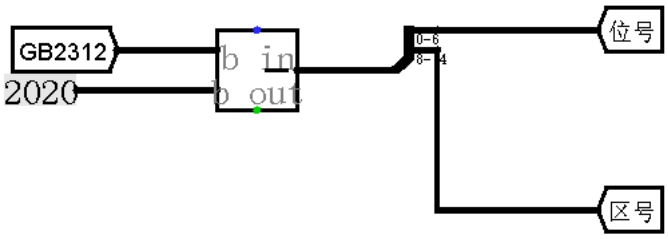


图 1.12 国际转区位码电路

1.2.2 偶校验编码

要实现 16 为数据位的偶校验编码，将这 16 位数据连接在一个 logisim 自带的偶校验器上，输出端作为偶校验编码的第 17 位，偶校验编码前 16 为即为输入数据。具体电路如图 1.13 所示

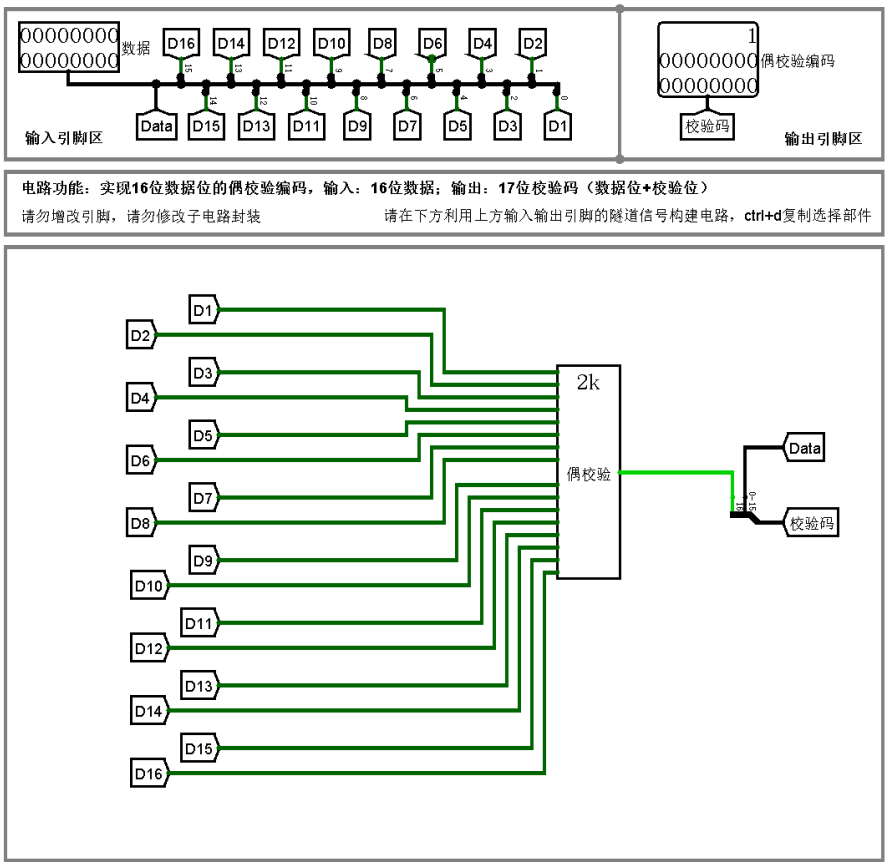


图 1.13 偶校验编码电路

1.2.3 偶校验检错

要实现 17 位偶校验编码的检错，将前 16 位数据位再进行一次偶校验，若结果与第 17 位相同，则没有检查出错误，若不同则检错成功。具体电路如图 1.14 所示。

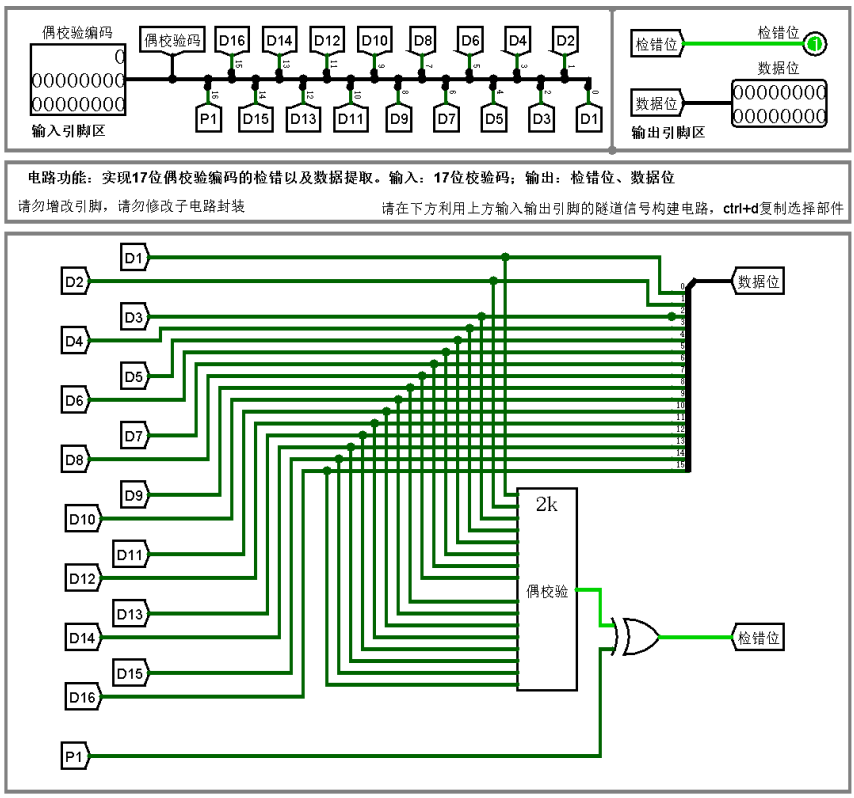


图 1.14 偶校验检错电路

1.2.4 海明编码

要实现输入 16 位数据，输出 22 位(数据位+校验位)的海明编码，首先确认校验码的位置，要实现 2 位检错和 1 位纠错，具体设计如表 1-1 所示

表 1-1

位置	1	2	3	4	5	6	7	8
内容	P1	P2	D1	P3	D2	D3	D4	P4
位置	9	10	11	12	13	14	15	16
内容	D5	D6	D7	D8	D9	D10	D11	P5
位置	17	18	19	20	21	22		
内容	D12	D13	D14	D15	D16	P6		

采用偶校验方式进行校验，校验位 P1=D1⊕ D2 ⊕ D4 ⊕ D5 ⊕ D7 ⊕ D9 ⊕ D11 ⊕ D12 ⊕ D14 ⊕ D16，校验位 P6 为 P1-P5 所有校验位加上所有数据位进行的偶校验结果，具体电路如图 1.15 海明编码电路所示。

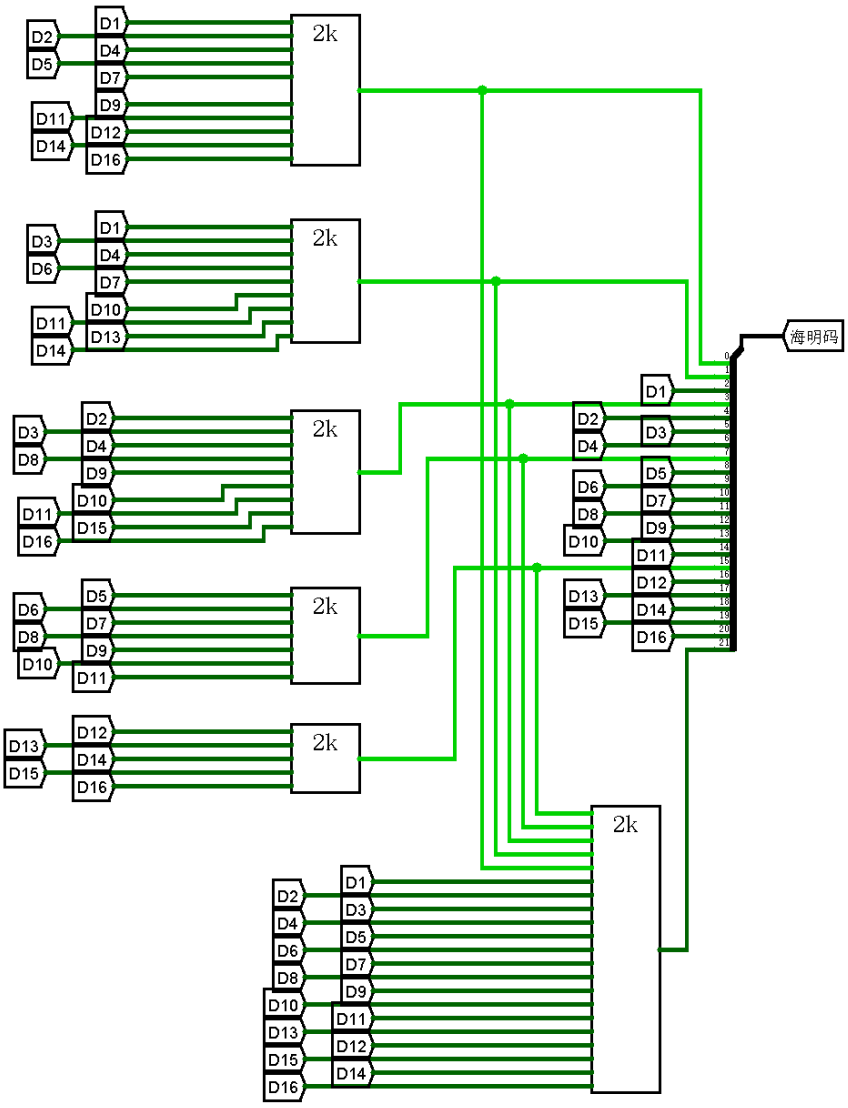


图 1.15 海明编码电路

1.2.5 海明解码

指错字由 G1G2G3G4G5G6 组成，其中 G1= P1 ⊕ D1⊕ D2 ⊕ D4 ⊕ D5 ⊕ D7 ⊕ D9 ⊕ D11 ⊕ D12 ⊕ D14 ⊕ D16，要实现自动纠错只需取指错字对应的二进制数，再和具体位本身进行异或，1⊕1=0，1⊕0=1，0⊕1=1，0⊕0=0，因此指错字对应的二进制数对应的数据位将会取反，这样就实现了自动纠错的功能。具体电路如图 1. 所示。

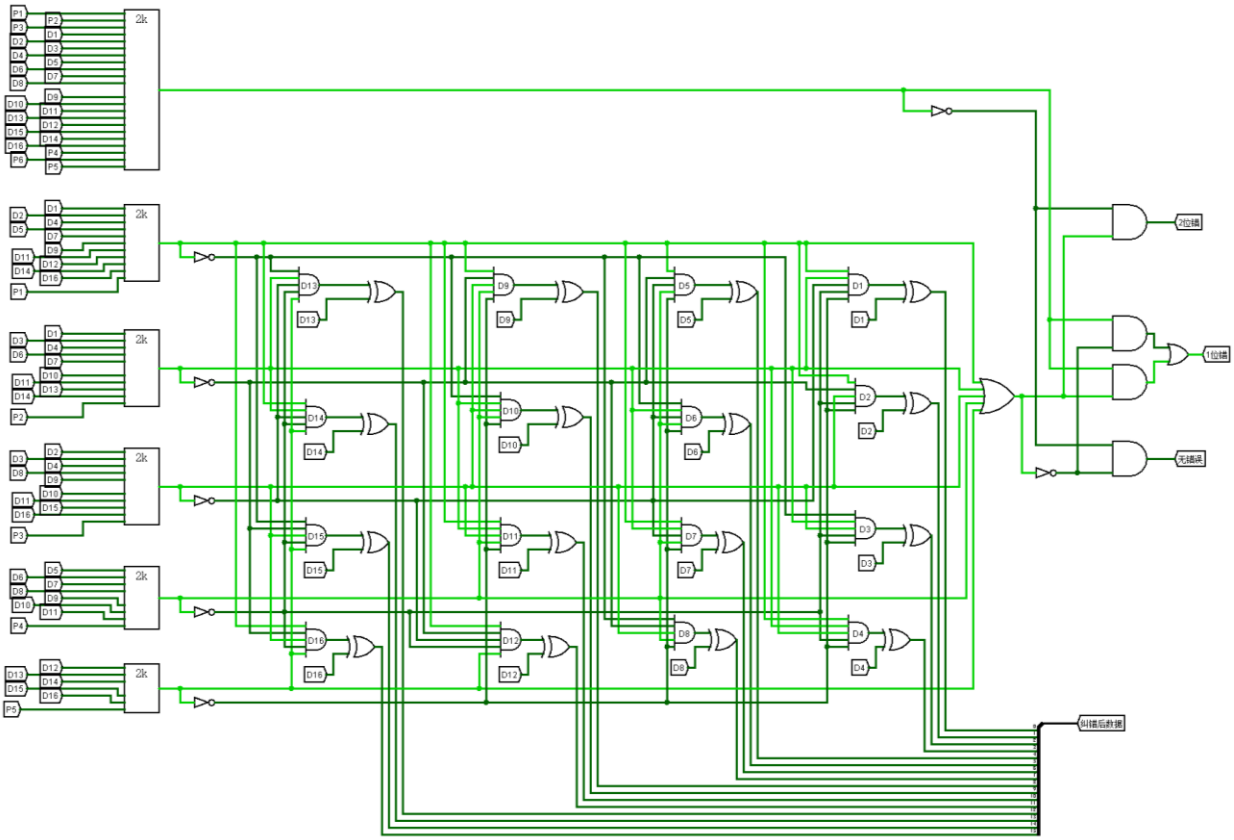


图 1. 16 海明解码电路

1.2.6 海明流水传输

要实现流水线工作，只有当海明解码无错误时才显示，出现错误后应当把之前存储的数据都清空，重新进行取数->编码->传输，具体电路如图 1. 所示

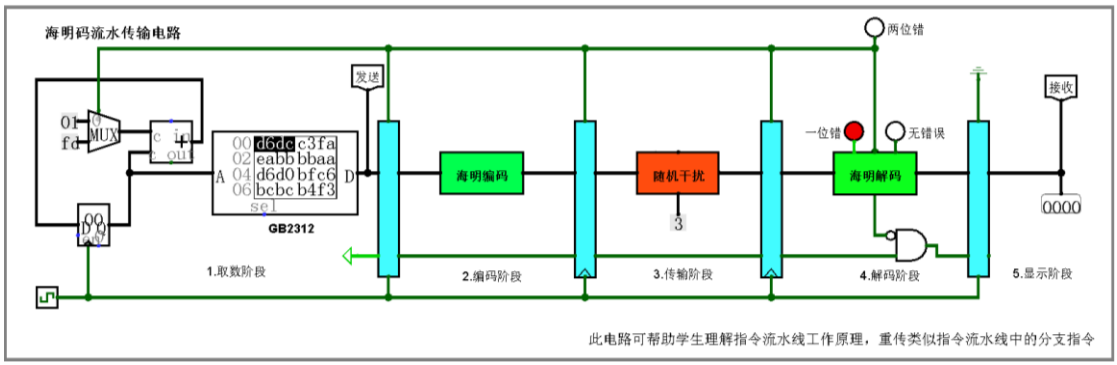


图 1. 17 海明流水传输电路

1.3 实验步骤

（1） 复习有关数据表示和校验码相关的内容。
（2） 熟悉电路中各部分的关系及信号间的逻辑关系。
（3） 设计实验电路，画出各模块的图，注意各引脚的标注，节省实验的时间。

1.4 故障与调试

1.4.1 多输入异或非门实验偶校验问题

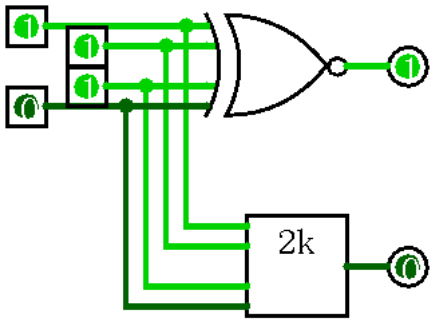


图 1.18 多输入异或门现象

故障现象：采用多输入异或非门时，无法正常实现偶校验。

原因分析：如图 1.所示，采用多输入异或非门时，结果与采用偶校验器的结果不同。这里应该采用分开的多个双输入异或门或者直接用 logisim 提供的偶校验器。

解决方案：将多输入异或非门更换为偶校验器。

1.4.2 海明解码一位错标识

故障现象：海明解码一位错标识与测试中应当出现的结果不符。

原因分析：原本一位错的情况只考虑了整体出错并且某一位出错的情况，其实整体出错但没有某一位出错也是属于一位错的情况

解决方案：在一位错标识前添加一个或门，加入新的判断条件。

1.5 测试与分析

1.5.1 汉字显示测试

指错字由 G1G2G3G4G5G6 组成

在汉字显示电路中测试国际码转区位码的电路是否正确，在 rom 中存放国际码，经过国际码转区位码电路后在点阵中输出具体字符，测试结果如图 1.所示，结果正确。

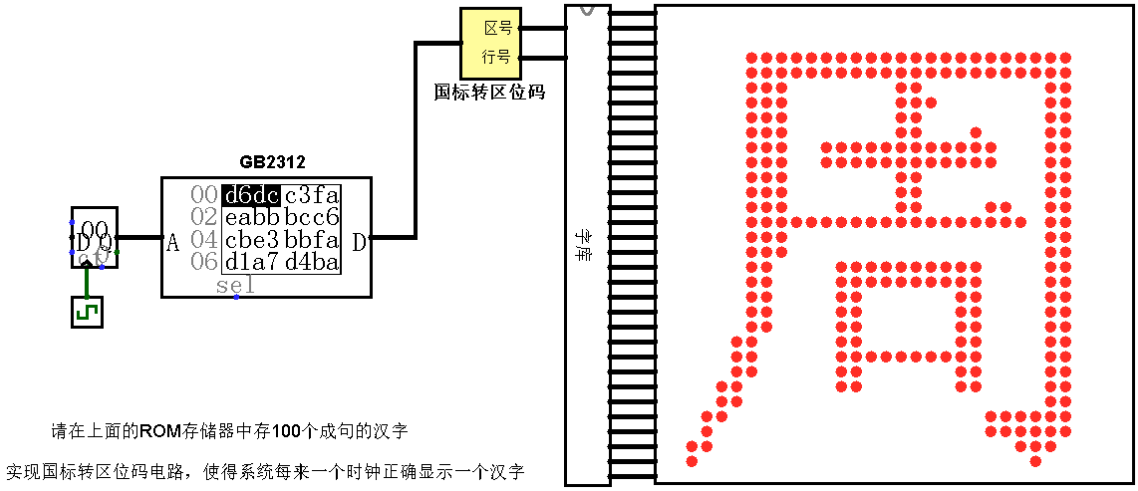


图 1.19 汉字显示测试

1.5.2 偶校验传输测试

在偶校验传输测试 2 电路中进行测试，发送方在 ROM 中预存类一段中文文字，通过计数器驱动电路每个时钟传输一个汉字到接受方，可以在中间窗口观察发送方和接收方汉字显示是否一致以观察检错性能，结果如图 1. 所示，偶校验传输测试成功。

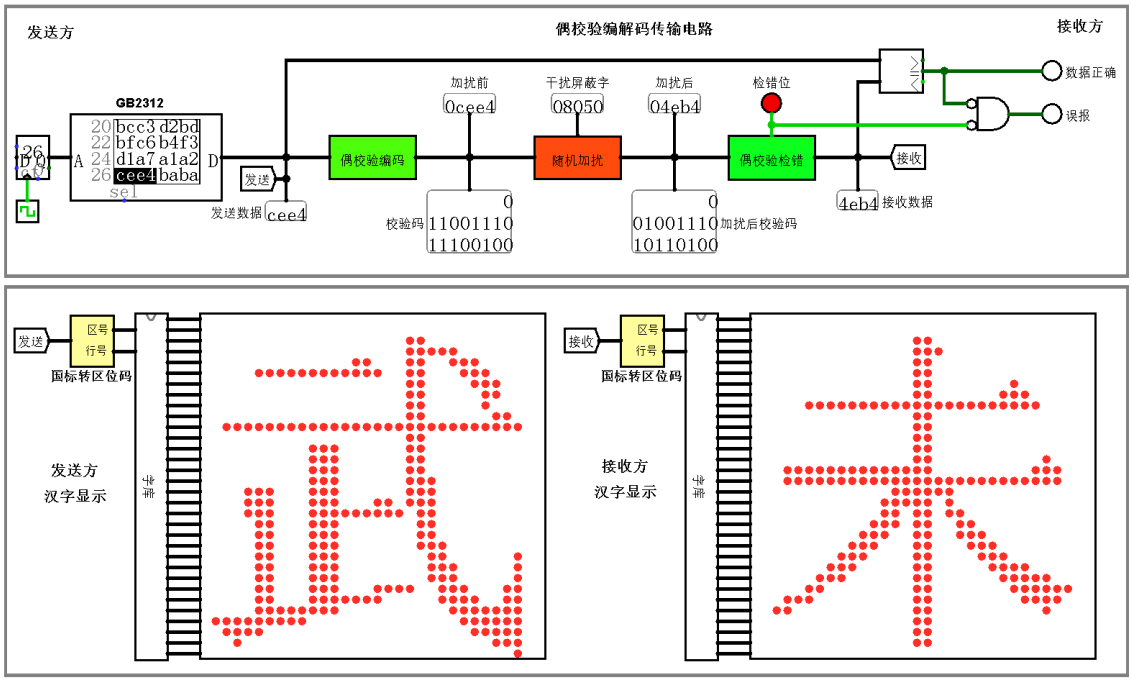


图 1.20 偶校验传输测试结果

1.5.3 海明传输测试

在海明传输测试 2 电路中进行测试，发送方在 ROM 中预存类一段中文文字，通过计数器驱动电路每个时钟传输一个汉字到接受方，可以在中间窗口观察发送方和接收方汉字显示是否一致以观察检错性能，当电路中出现一位错时，应当自动纠错并在标识位中显示，出现两位错时，无法自动纠错，但应该在标识为中显示，结果如图 1. 和图 1. 所示，海明传输测试成功。

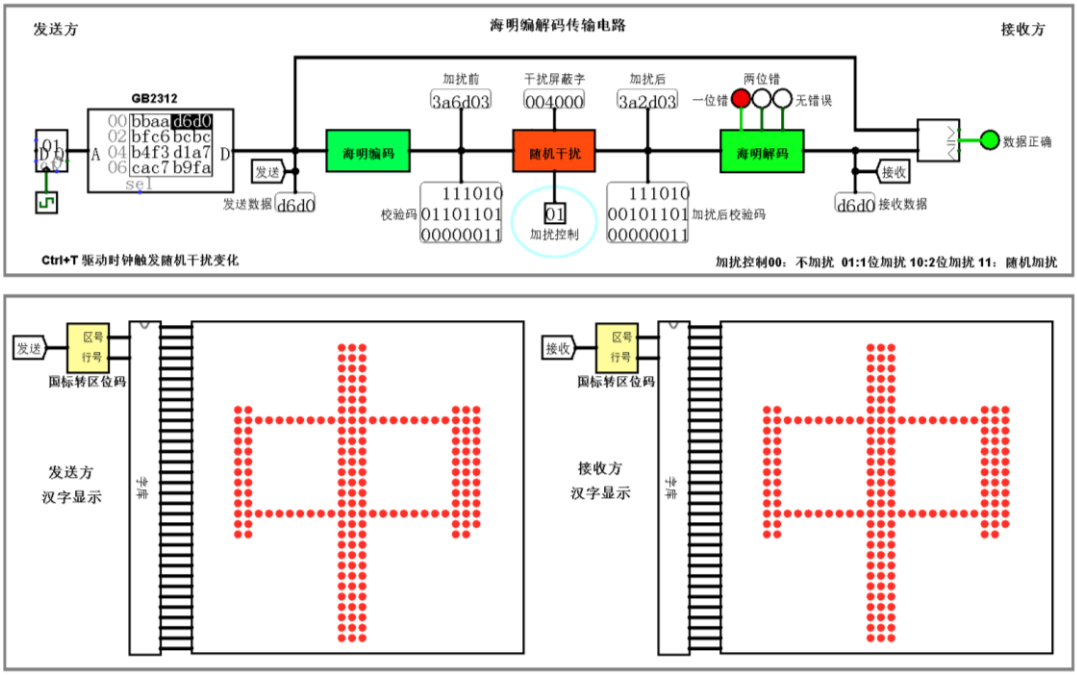


图 1.21 海明传输测试 1 位错

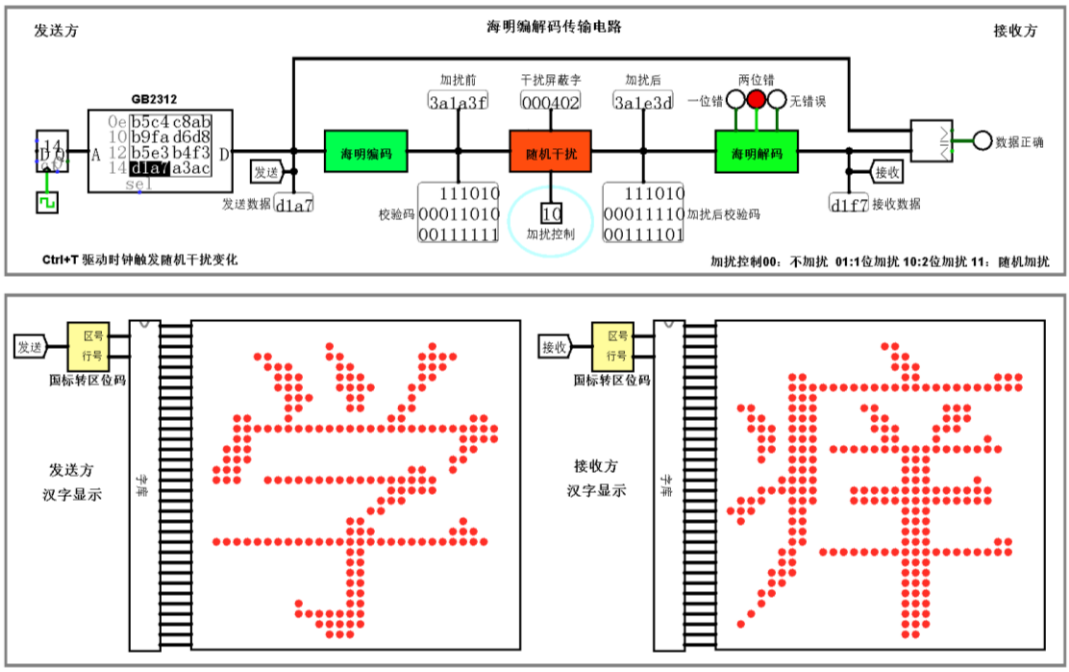


图 1.22 海明传输测试 2 位错

1.5.4 海明流水传输测试

在海明流水传输电路中进行测试，发送方在 ROM 中预存类一段中文文字，通过计数器驱动电路每个时钟传输一个汉字到接受方，可以在中间窗口观察发送方和接收方汉字显示是否一致以观察检错性能，右边的窗口显示的应当为一段连续的话，测试结果如图 1.2 所示，海明流水传输测试成功。

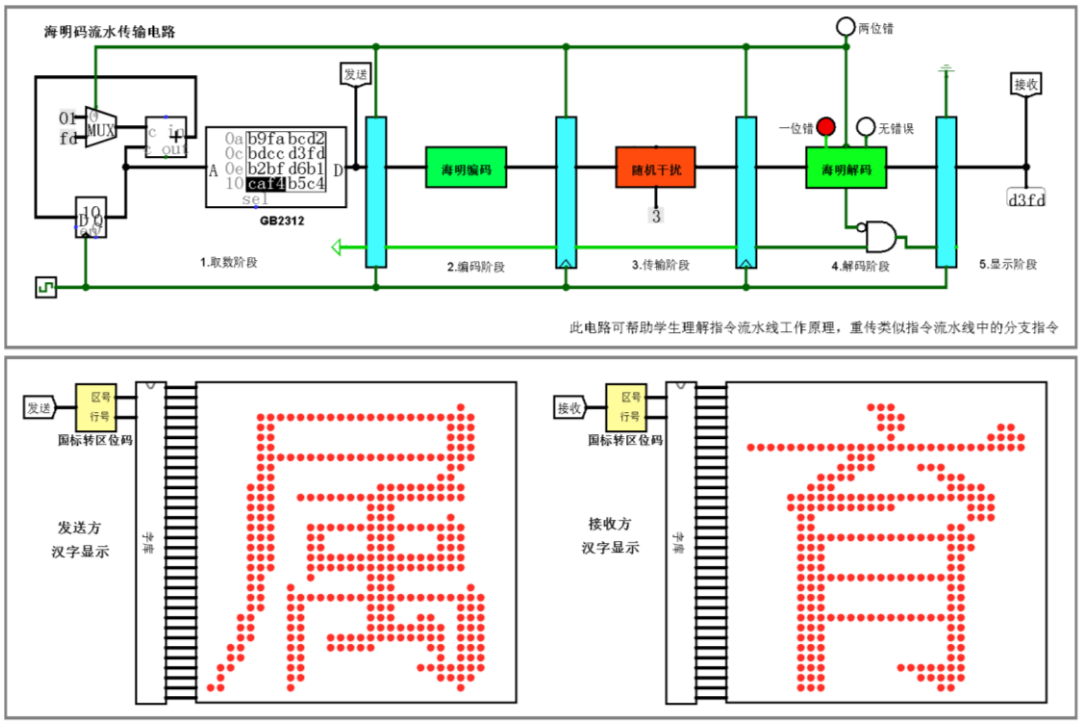


图 1.23 海明流水传输测试

2 CPU 设计实验

2.1 设计要求

利用运算器实验，存储系统实验中构建的运算器、寄存器文件、存储系统等部件以及 Logisim 中其他功能部件构建一个 32 位 MIPS CPU 单周期处理器，该处理器应支持基础指令集中列出的所有指令，见表 2-1 基础指令集，另外还必须支持扩展指令集中的 2 条 C 类运算指令，1 条 M 类存储指令，1 条 B 类分支指令（详见表 2-2），具体任务每位同学不一样，任务要求见任务分配清单，每位同学得到一个 4 位数据表示的任务编号，分别对应 CCMB 指令的选择。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 应能运行教师提供的标准测试程序，程序存储在 Logisim ROM 模块中（指令存储器、数据存储器分开）。

表 2-1 基础指令集

#	指令	格式	备注
1	Add	add $rd, $rs, $rt	指令功能及指令格式参考 MIPS32 指令集
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	
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 注意数码管输入数据应该用寄存器锁存

表 2-2 扩展指令集

指令分类	编号	指令助记符	简单功能描述	备注
运算 (C)	1	SLLV	逻辑可变左移	指令格式参考 MIPS32 指令集，最终功能以 MARS 模拟器为准。
	2	SRLV	逻辑可变右移	
	3	SRAV	算术可变右移	
	4	SUBU	无符号减	
	5	XOR	异或	
	6	XORI	异或立即数	
	7	LUI	立即数加载至高位	
	8	SLTIU	小于立即数置 1(无符号)	
	9	MULTU	乘无符号	
	A	DIVU	无符号除	
	B	MFLO	读 LO 寄存器	
	1	LB	加载字节	

存储访问（M）	2	LBU	加载字节(无符号)	
	3	LH	加载半字	
	4	LHU	加载半字(无符号)	
	5	SB	存储字节	
	6	SH	存储半字	
跳转（B）	1	BLEZ	小于等于 0 转移	
	2	BGTZ	大于 0 转移	
	3	BLTZ	小于 0 转移	
	4	BGEZ	大于等于 0 转移	

2.2 方案设计

2.2.1 主要功能部件

PC 寄存器：采用一个 32 位的寄存器，地址转移逻辑 NPC 未单独作为一个部件，而是嵌入在整个电路中，正常的执行完一条指令后，PC 寄存器中的值+1，指向下一条指令，遇到条件分支或者无条件跳转指令时，PC 寄存器的值相应改变。

指令存储器 IM：采用地址线 10 位的 32 位 ROM，其中保存着指令对应的机器码，每一条指令为 32bit，PC 寄存器的低 10 位用于取对应地址的指令。

数据存储器 DM：采用地址线 16 位的 32 位 RAM，因为需要修改其中内容，所以不采用只读内存 ROM。

立即数扩展器 EXT：根据要求的指令，有三种扩展方式，一种是把指令的 0-15 位进行有符号扩展，一种是把指令的 0-15 位进行无符号扩展（ori），还有一种是把指令的 6-10 位进行无符号扩展（sll、sra、srl）。立即数扩展是为了使立即数能够参与 ALU 运算，根据 2 位操作码的不同执行不同功能。

运算器 ALU：往期实验内容，已实现。

寄存器堆 RF：采用 CS3401 库中的 32 位寄存器组。

2.2.2 数据通路

首先画出主要功能输入来源表格，如表 2-3 所示

表 2-3 输入来源表

主要功能输入来源											
指令	PC寄存器	指令存储器IM	寄存器堆RF				运算器ALU		数据存储器DM		立即数符号位扩展EXT
			R1#	R2#	W#	Din	A	B	Addr	Din	
add		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
addi		PC	IM[25:21]		IM[20:16]	ALU	RF.R1	EXT.s			IM[15:0]
addiu		PC	IM[25:21]		IM[20:16]	ALU	RF.R1	EXT.s			IM[15:0]
addu		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
and		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
andi		PC	IM[25:21]		IM[20:16]	ALU	RF.R1	EXT.s			IM[15:0]
sll		PC	IM[20:16]		IM[15:11]	ALU	RF.R1	EXT.10			IM[10:6]
sra		PC	IM[20:16]		IM[15:11]	ALU	RF.R1	EXT.10			IM[10:6]
srl		PC	IM[20:16]		IM[15:11]	ALU	RF.R1	EXT.10			IM[10:6]
sub		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
or		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
ori		PC	IM[25:21]		IM[20:16]	ALU	RF.R1	EXT.01			IM[15:0]
nor		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
lw		PC	IM[25:21]		IM[20:16]	DM	RF.R1	EXT.s	ALU		IM[15:0]
sw		PC	IM[25:21]	IM[20:16]			RF.R1	EXT.s	ALU	RF.R2	IM[15:0]
beq	EXT.s	PC	IM[25:21]	IM[20:16]							IM[15:0]
bne	EXT.s	PC	IM[25:21]	IM[20:16]							IM[15:0]
slt		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
slti		PC	IM[25:21]		IM[20:16]	ALU	RF.R1	EXT.s			IM[15:0]
sltu		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
j	index	PC									
jal	index	PC			0x1F	PC+4					
jr	RF.R1	PC	IM[25:21]								
syscall		PC	0x2	0x4							
XOR		PC	IM[25:21]	IM[20:16]	IM[15:11]	ALU	RF.R1	RF.R2			
SLTIU		PC	IM[25:21]		IM[20:16]	ALU	RF.R1	EXT.s			IM[15:0]
SH		PC	IM[25:21]	IM[20:16]			RF.R1	EXT.s	ALU	RF.R2	IM[15:0]
BLEZ	EXT.s	PC	IM[25:21]								IM[15:0]

得到输入来源后，连接电路中的数据通路，若一个端口有两个以上的数据输入，则添加若干个多路选择器，如此便可以实现特定指令传输特定数据的功能，多路选择器的控制端连接的信号即为控制器所需要的控制信号。

2.2.3 控制器

首先根据数据通路中出现的控制信号，画出控制信号表格，如表 2-4 所示，空着的表格即为默认 0，表中蓝色背景为 ALU 的功能，并不是控制信号，之后会根据这些 ALU 的功能来确定 ALUOP，ExtOP 共有三种形式 00 为默认，01，10 分别为 16 位无符号扩展和 5 位无符号扩展。

表 2-4 控制信号表

	控制信号	add	addi	addiu	addu	and	andi	sll	sra	srl	sub	or	ori	nor
0	SLL							1						
1	SRA								1					
10	SRL									1				
101	ADD	1	1	1	1									
110	SUB										1			
111	AND					1	1							
1000	OR											1	1	
1001	XOR													
1010	NOR													1
1011	SLT													
1100	SLTU													
	ALUOP													
	RegDst	1			1	1		1	1	1	1	1		1
	ALUsrc		1	1			1	1	1	1			1	
	Regwrite	1	1	1	1	1	1	1	1	1	1	1	1	1
	MemToReg													
	Memwrite													
	Extop							10	10	10			1	
	blez													
	beq													
	bne													
	j													
	jal													
	jr													
	Syscall													
	sh													

（续上表）

lw	sw	beq	bne	slt	slti	sltu	j	jal	jr	syscall	XOR	SLTIU	SH	BLEZ
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	

随后查询资料，根据 mips 指令对应的机器码中 op 和 func 来确定具体操作，并在控制器中实现这个操作对应的控制信号，例如 add 指令，op=000000，func=100000，根据表 2-4 的内容，add 操作对应的控制信号为 RegDst 和 RegWrite，同时还有 ALUOP（这个后面考虑），那么 add 操作的部分电路如图 2.1 所示。

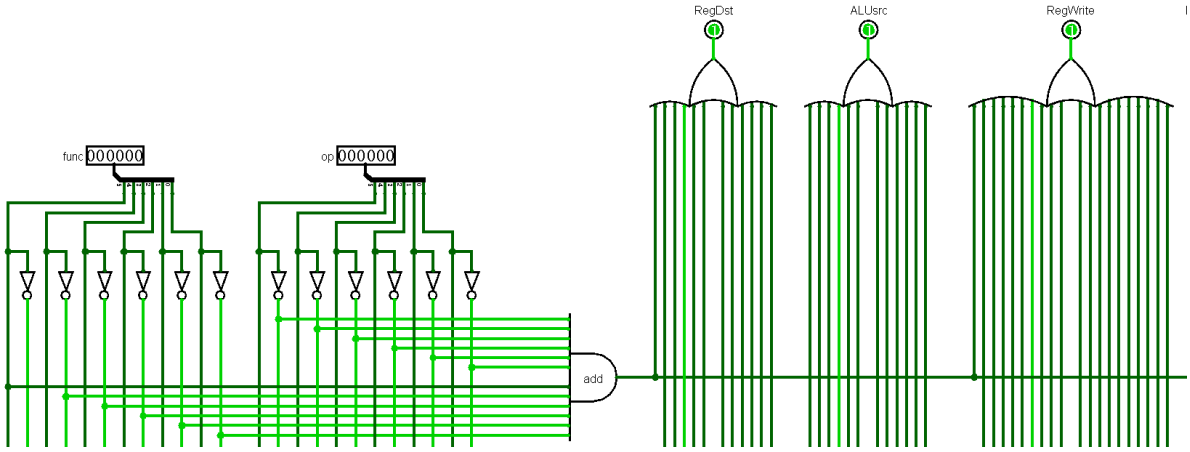


图 2.1 add 操作部分控制电路

其他指令与 add 类似，在此不赘述。接下来要确定 ALUOP 的值，这是要选择 ALU 的具体功能，列出 ALU 的真值表如表 2-5 ALU 真值表所示。

表 2-5 ALU 真值表

SLL	SRA	SRL	ADD	SUB	AND	OR	XOR	NOR	SLT	SLTU	ALUOP[3]	ALUOP[2]	ALUOP[1]	ALUOP[0]
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	0	1	0	1	0
0	0	0	0	0	0	0	0	0	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	ALUOP[3]	OR+XOR+NOR+SLT+SLTU
	ALUOP[2]	ADD+SUB+AND+SLTU
	ALUOP[1]	SRL+SUB+AND+NOR+SLT
	ALUOP[0]	SRA+ADD+AND+XOR+SLT

根据 ALU 真值表，画出 ALUOP 的电路如图 2.2 所示。至此控制器设计完成。

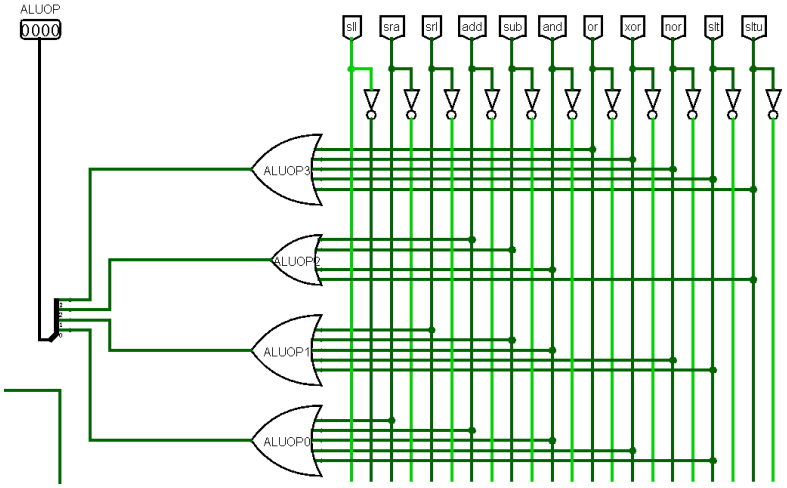


图 2.2 ALUOP 生成电路

2.2.4 各类输出信息

首先输出 SyscallOut 到数码管上，SysCall 的功能如下：

if $v0==10

halt(停机指令)

else

数码管显示$a0 值

把寄存器 v0 与 10 的比较结果与 syscall 信号相与作为多路选择器的控制信号，同时添加一个寄存器缓存$a0 的值，避免险象发生，因此 syscallout 显示电路如图 2.3 所示。左下角输入来自于寄存器 RF.B 输出。

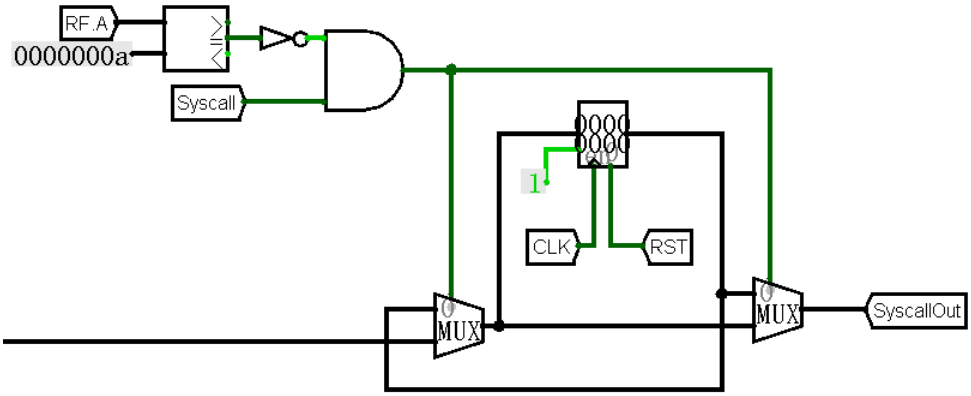


图 2.3 SysCallOut 电路

总周期数在停机之后不再计数，但仍然要计算停机指令这条指令，因此添加一个寄存器作为缓冲，当停机时，寄存器中保存的还是未停机状态，计数器继续计数，到下一个周期后，寄存器中的状态更新为停机，计数器停止计数。电路如图 2.4 所示。

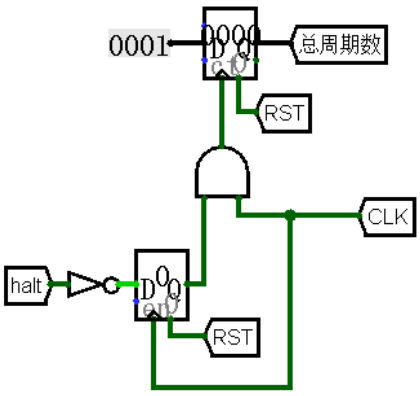


图 2.4 总周期数电路

无条件分支指令数、有条件分支指令数、有条件分支成功跳转即为简单计数器，

不必考虑停机问题，因为停机以后不会再取新的指令，他们的数量也就不再变化，具体电路如图 2.5 所示。

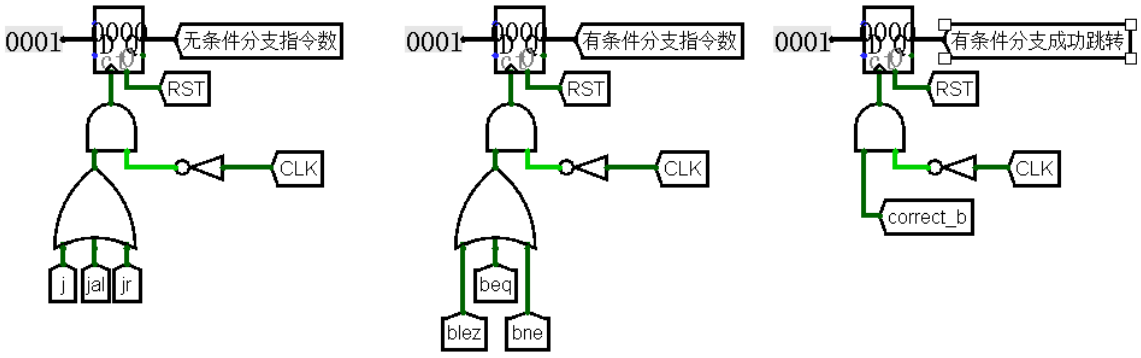


图 2.5 输出电路

2.2.5 CPU 汇总设计

经过以上多个步骤模块，设计出的 CPU 电路如图 2.6 所示

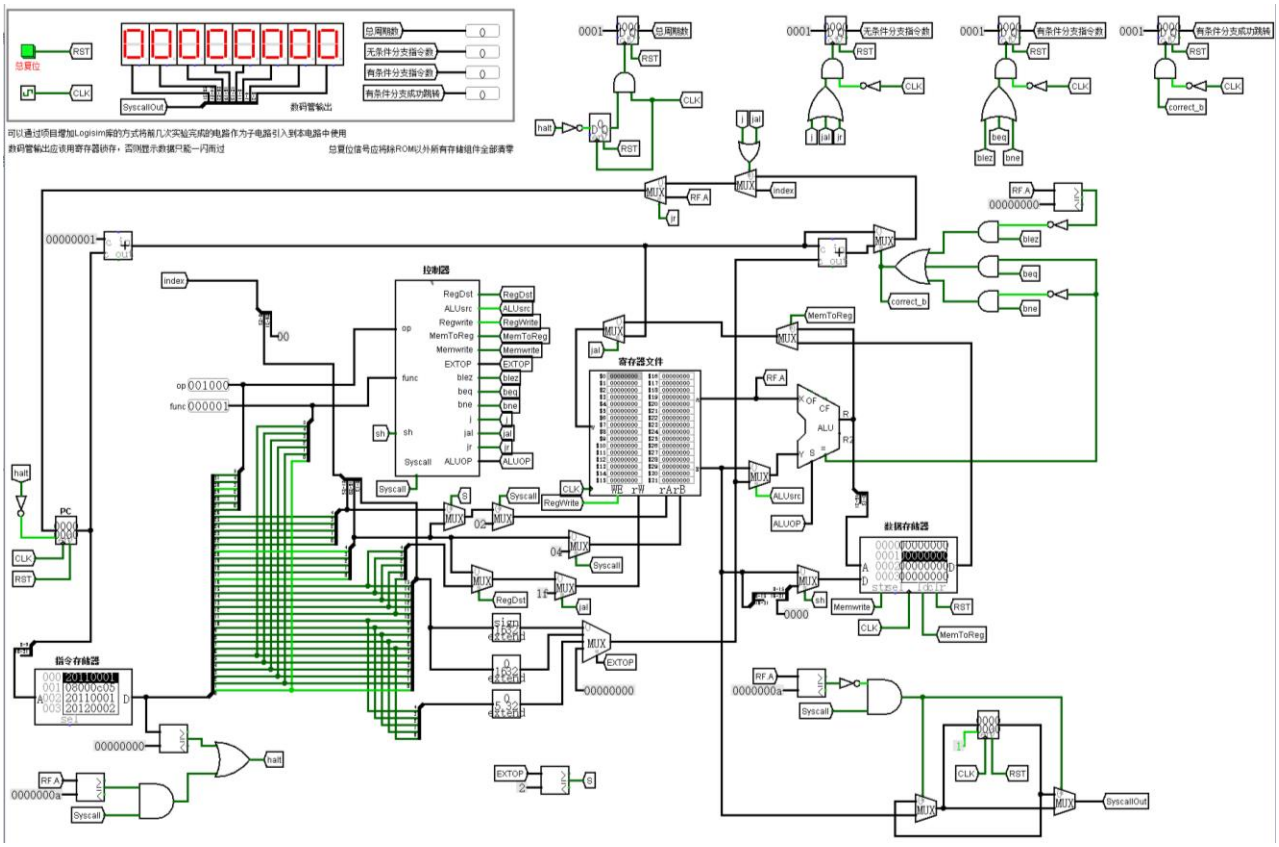


图 2.6 CPU 电路

2.3 实验步骤

（1）构建主要功能部件。

（2）一次性构建所有数据通路。

（3）输入源合并。

（4）控制信号综合。

（5）系统调试。

2.4 故障与调试

2.4.1 数码管显示错误问题

故障现象：执行 syscall 指令时，数码管显示的是上一个周期的值。

原因分析：原本把 syscall 作为控制信号，多路选择器输出端直接接到数码管上出现了险象。

解决方案：添加一个寄存器保存 RF.B 的值，如果控制信号为 1，则直接把 RF.B 作为数码管的输入端，如果控制信号为 0，则用寄存器锁存的值作为数码管的输入

2.4.2 总周期少 1 问题

故障现象：在执行 benchmark 测试时，跑完一遍发现总周期数少 1。

原因分析：原本没有把停机指令计算进去，一停机计数器就停止计数了。

解决方案：添加一个寄存器作为缓冲，当停机时，寄存器中保存的还是未停机状态，计数器继续计数，到下一个周期后，寄存器中的状态更新为停机，计数器停止计数。

2.5 测试与分析

2.5.1 Benchmark 测试

将老师提供的 benchmark.asm 经过 Mars 汇编后生成 benchmark.hex，导入 CPU 的指令存储器中进行测试，测试结果如图 2.7 所示，cpu 正确运行。

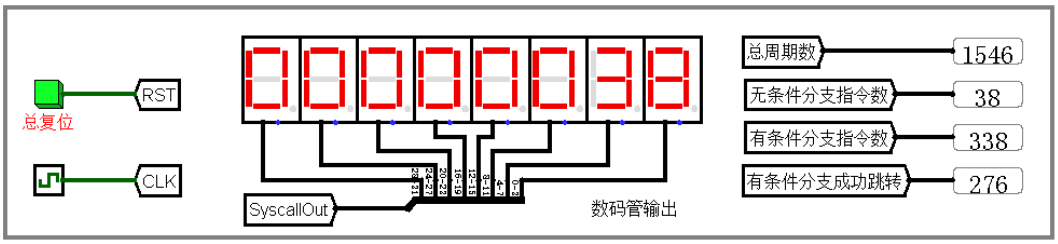


图 2.7 benchmark 测试

2.5.2 CCMB 测试

根据要求自己编写汇编代码 CCMB.asm，代码如下：

```
# U201514559 周铭昊 CS1503 班
# 5 XOR 异或
# 8 SLTIU 小于立即数置 1(无符号)
# 6 SH 存储半字
# 1 BLEZ 小于等于 0 转移
.text
    addi $s0,$zero,1    #s0 寄存器置 1
    addi $s1,$zero,1    #s1 寄存器置 1
    addi $s2,$zero,0xffffffff   #s2 寄存器置 0xffffffff
    xor $a0, $s0, $s1 #s0 与 s1 异或：1 xor 1 = 0 赋值给 a0
    sltiu $a1, $a0, 1 #a0 < 1 ,a1 置 1
    xor $a1, $a1, $s2 #s2 与 a1 异或，结果为 ffffffff xor 00000001 = fffffffe
    sh $a1, 0($zero) #把 a1 半字保存在数据存储器中，即 fffe 保存在 0 号位置
    blez $a0, Next1     #a0 <= 0，转移到 Next1
Next1:
    addi $v0,$zero,1    #v0 寄存器置 1
    addi $a0,$zero,8    #a0 寄存器置 8
    syscall             #display 8
```

cpu 运行结果如图 2.8 所示，四种指令均正确执行。

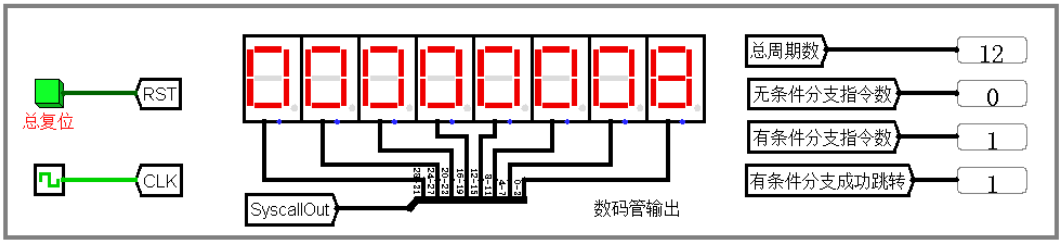

RST
总复位
CLK
SyscallOut
数码管输出
总周期数
12
无条件分支指令数
0
有条件分支指令数
1
有条件分支成功跳转
1

图 2.8 CCMB 测试

3 总结与心得

3.1 实验总结

本次实验主要完成了如下几点工作：

1） 设计了偶校验编码以及海明编码解码电路，设计了流水线传输的电路，设计了 CPU 控制器电路，设计了一个 32 位 CPU 的数据通路，设计了 ALU 真值表，设计了立即数扩展器 EXT，设计了 CCMB 汇编程序。

2） 实现了国标码到区位码的转化，实现了偶校验检错，实现了海明码 1 位自动纠错 2 位检错，实现了海明编码流水线传输正确顺序的文字，实现了能跑 33 种指令的 CPU。

3） 完成了一个流水线传输电路，完成了一个 32 位 MIPS CPU，完成了 CCMB 测试。

3.2 实验心得

1） 经过本次实验，我对编码和解码的方式以及 CPU 的内部结构和运行方式更加熟悉，尤其是跑自己编写的汇编程序时，软硬件的知识相结合，有一种对计算机运行方式豁然开朗的感受。

2） 在老师成熟的实验设计下，我们只需要完成最核心的内容，这是十分高效的实验方式，只是一些测试程序比如 Mars 汇编需要自学，希望能在实验课上获取一些额外工具的知识。

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: 周铭昊

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）	报告撰写 （30 分）	课设过程 （70 分）	最终评定 （100 分）
得分			

指导教师签字: _____ **2018-01-12**