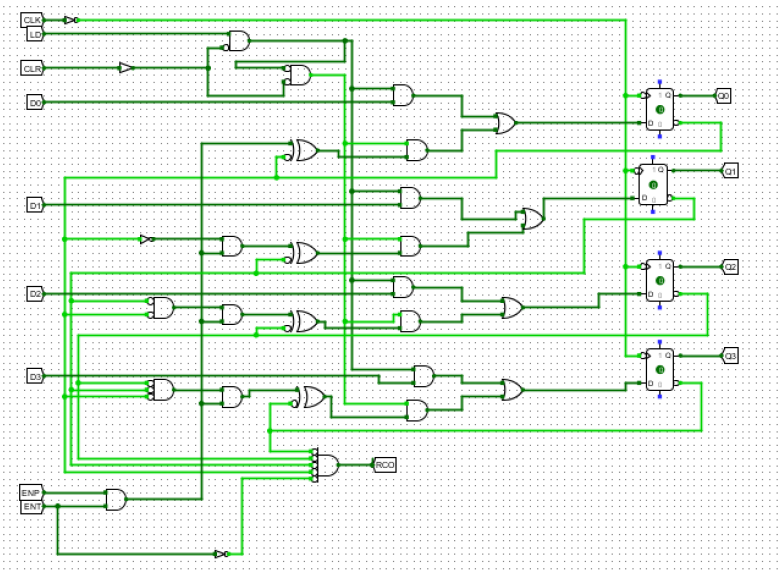
目录

1． 计数器实验

1.1 分析需求

根据表 3.1 给出的功能表和图 3.1 所示电路原理图构建 4 位二进制同步计数器电路

1.2 Logisim，启动!



1.3 测试

通过仿真测试，结合功能表，观察输出，发现均满足要求

表 3.1 4 位同步二进制计数器功能表

Inputs				Current State				Next State			
CLR	LD	ENT	ENP	Q3	Q2	Q1	Q0	Q3*	Q2*	Q1*	Q0*
1	x	x	x	x	x	x	x	0	0	0	0
0	1	x	x	x	x	x	x	D3	D2	D1	D0
0	0	0	x	x	x	x	x	Q3	Q2	Q1	Q0
0	0	x	0	x	x	x	x	Q3	Q2	Q1	Q0
0	0	1	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	1	0	0	1	0
						...					
0	0	1	1	1	1	0	1	1	1	1	0
0	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	1	1	1	1	0	0	0	0

1.4 总结与心得

本实验的线路较多，要注意小心错误连接和眼花，特别是有无反向圈这种小细节。

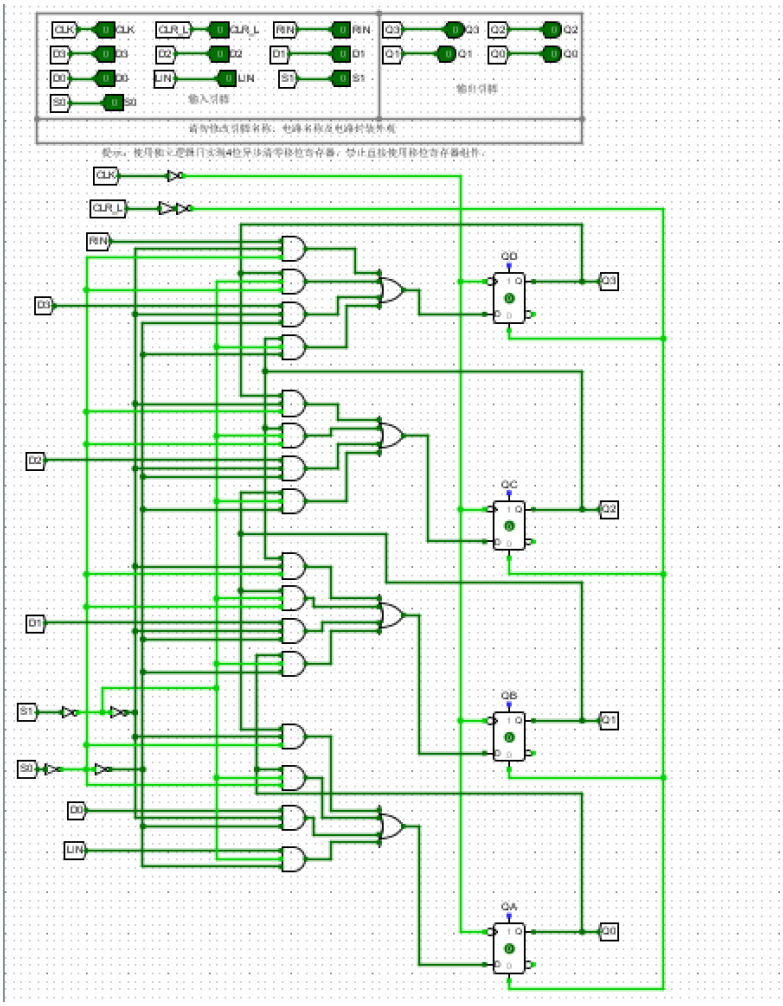
2． 移位寄存器实验

2.1 分析需求

根据表 3.2 给出的功能描述和图 3.5 给出的电路原理图，构建 4 位通用移位寄存器电路 SHRG4U，该移位寄存器带有异步复位（清 0）信号 CLR，它是低电平有效信号，当它为低电平时，所有 D 触发器的状态输出为 0。

2.2 Logisim，启动!

按要求连接好电路



2.3 仿真测试

测试结果符合功能表

表 3.2 4 位移位寄存器功能表

功能	输入			下一个状态			
	CLR	S1	S0	Q3*	Q2*	Q1*	Q0*
清零	0	x	x	0	0	0	0
保持	1	0	0	Q3	Q2	Q1	Q0
右移	1	1	0	RIN	Q3	Q2	Q1
左移	1	0	1	Q2	Q1	Q0	LIN
装载	1	1	1	D3	D2	D1	D0

2.4 总结与心得

本实验的线路较多，要注意小心错误连接和眼花

3． 4 位无符号数乘法器

3.1 分析需求

实验将实现两个四位二进制无符号数相乘的功能，并通过数码管将其转换成十六进制显示出来

3.2 逻辑结构图

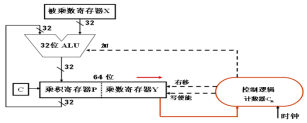
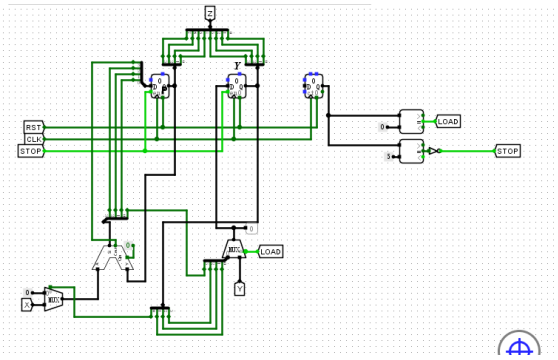


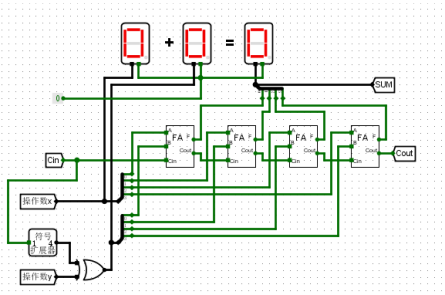
图 3.7 实现 32 位无符号数乘法运算的逻辑结构图

3.3 Logisim，启动!

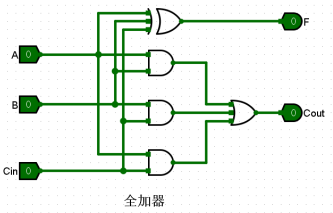
乘法器的实现



4 位全加器的实现



全加器的实现



全加器

4． 寄存器堆实验

4.1 寄存器原理图

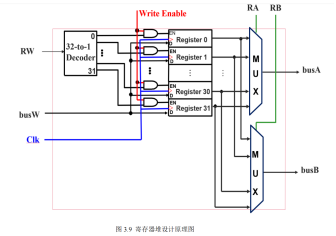
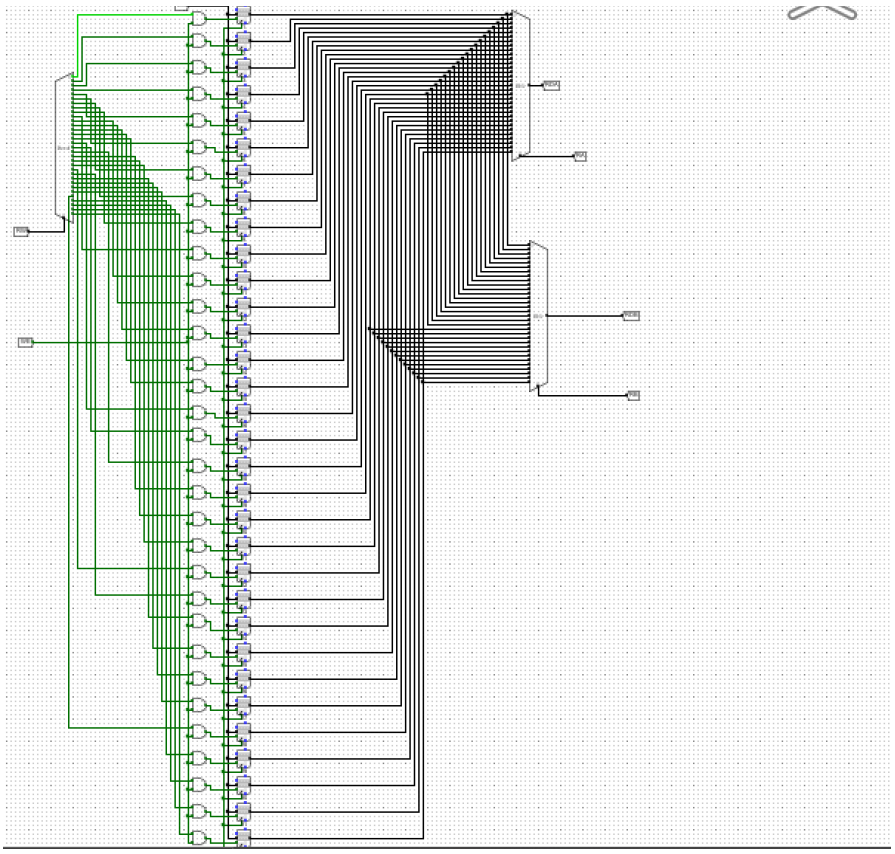


图 3.9 寄存器堆设计原理图

4.2 Logisim，启动!

寄存器电路图



4.3 总结与心得

本题难度不大，但是连线的工作量较大，要更加小心。

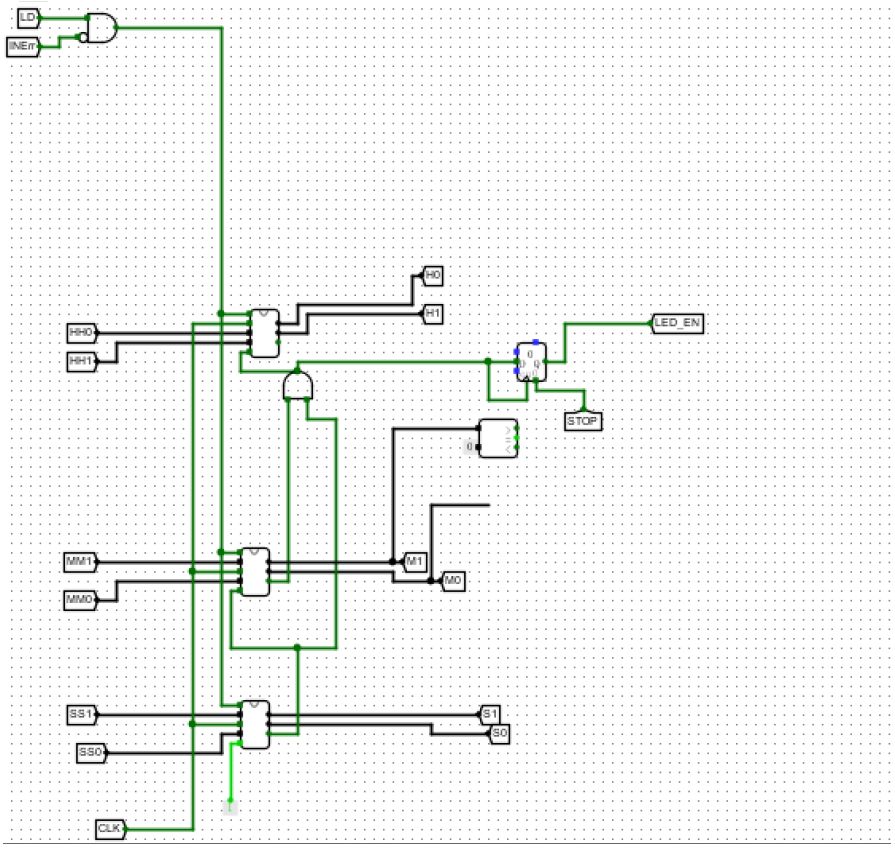
5． 数字时钟实验

5.1 分析需求

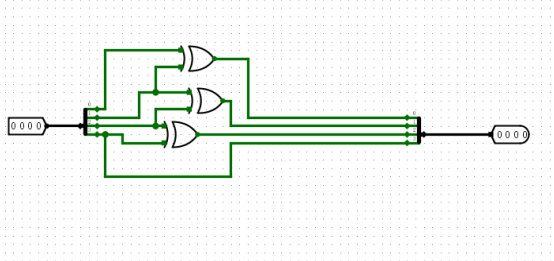
在 6 个 7 段数码管上显示数字时钟时分秒，当计时到 23:59:59 后进入 00:00:00，时分秒。之间用小数点分隔；到整点时轮流点亮三色 LED 灯组件；使用 8421BCD 码设置初始时间，在载入时如果初始时间数值超出实际范围，则报错，且不能被载入。

5.2 Logisim，启动！

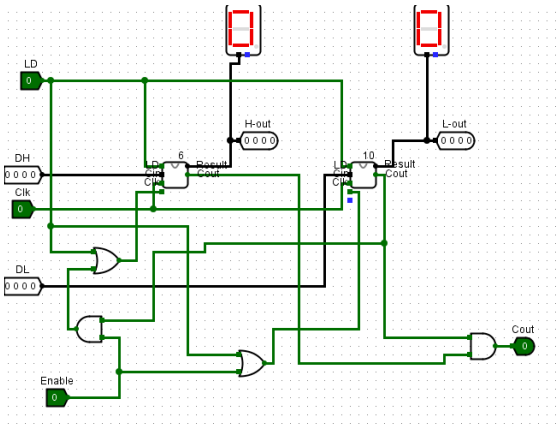
电路的完整实现



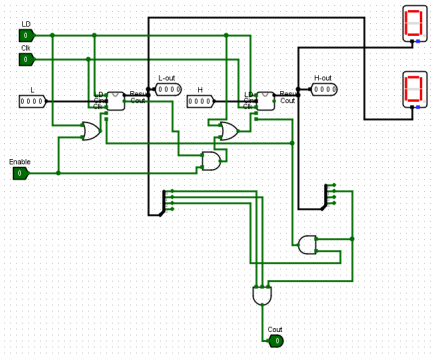
二进制转换为格雷码的实现



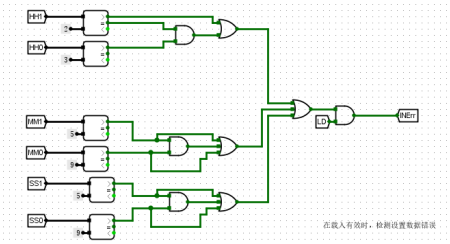
60 进制计数器的实现



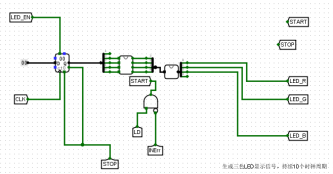
24 进制计数器的实现



InErr 的实现



RGB 灯的实现



5.3 总结与心得

怎么说呢，这个题目我觉得最会遇到问题的其实是在对于 RGB 亮灯的文字解读上 () 所以很容易出现面向 OJ 的编程这样子 () 不过也是没有办法的事情，毕竟自然语言的歧义还是很大的。

6． 思考题

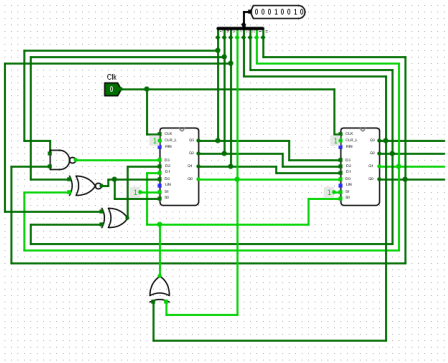
6.1 8 位二进制伪随机数生成电路

要求级联 2 个 4 位移位寄存器子电路实现生成 8 位二进制伪随机数生成电路。

查阅资料

可以使用反馈线性移位寄存器（LFSR）的方法，使用逻辑门和寄存器来生成伪随机序列, 连接两个寄存器。将第一个寄存器 R1 的输出与第二个寄存器 R2 的输入连接起来，在两个寄存器中选择适当的反馈线路，以确保伪随机序列的生成。

电路实现



6.2 查找资料学习如何利用加法器实现 8 位无符号数的快速乘法器

查阅资料

阵列乘法器采用类似人工计算的方法进行乘法运算。人工计算方法是用乘数每一位去乘被乘数，然后将每一位权值对应相加得出每一位的最终结果。用乘数的每一位直接去乘被乘数得到部分积并按位列为一行，每一行部分积末位与对应的乘数数位对齐，体现对应数位的权值。将各次部分积求和，即将各次分积的对应数位求和即得到最终乘积的对应数位的权值。为了进一步提高乘法的运算速度，可采用大规模的阵列乘法器来实现，阵列乘法器的乘数与被乘数都是二进制数。可以通过乘数从最后一位起一个一个和被乘数相与，自第二位起要依次向左移一位，形成一个阵列的形式。这就可将其看成一个全加的过程，将乘数某位与被乘数某位与完的结果加上乘数某位的下一位与被乘数某位的下一位与完的结果再加上前一列的进位进而得出每一位的结果。[https://zhuanlan.zhihu.com/p/644232007]

参考原理图

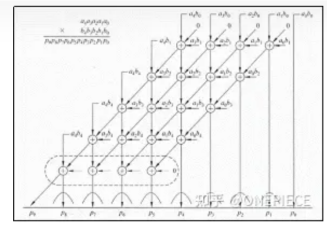


图2 斜向进位阵列乘法器原理

八位无符号数可通过类似的原理实现。

6.3 修改寄存器堆

实验要求

修改寄存器堆的设计电路，将输入信号分别连接寄存器的使能端和时钟端，验证寄存器堆的读写功能，分析使能信号和写入地址信号的先后时序关系变化是否影响到写入结果。

(由于电路图片太大贴图略)

分析

如果使能信号在时钟信号之前变为有效（即在上升沿之前），那么即使时钟信号上升沿到来，寄存器也不会写入数据，因为使能信号未激活。

如果使能信号在时钟信号上升沿之后变为有效，那么在下一个时钟上升沿时，数据将被写入寄存器。

6.4 闹钟

分析

我们已经实现了数字时钟，那么要做的就是将时钟的结果与我们想要的“设定时间”判断是否相等。

电路图及分析

其中，HH、HL、MH、ML、SH、SL 用于输入设定的时间，并且与时钟输出的时间比较，全都相等时闹钟生效，蜂鸣器响。