计组实验三报告

孙启翔 241220098

April 2025

1 4 位同步计数器设计

1.1 实验整体方案设计

本题想要设计一个同步计数器，其中引脚 CLK 代表时钟信号，LD 代表初始化计数器（控制计数器从几开始），CLR 代表将信号清零，$D_0 - D_3$ 代表计数器用于初始化的 4 位数据，ENP 和 ENT 只有同时为 1 时计数器才能正常工作。

当时钟信号开始后，输出信号 $Q_0 - Q_3$ 依次从 0000 逐步变为 1111，并且每个时钟周期变化一次，需要注意题目中要求的 D 触发器是沿下降沿触发。

1.2 实验原理图和电路图

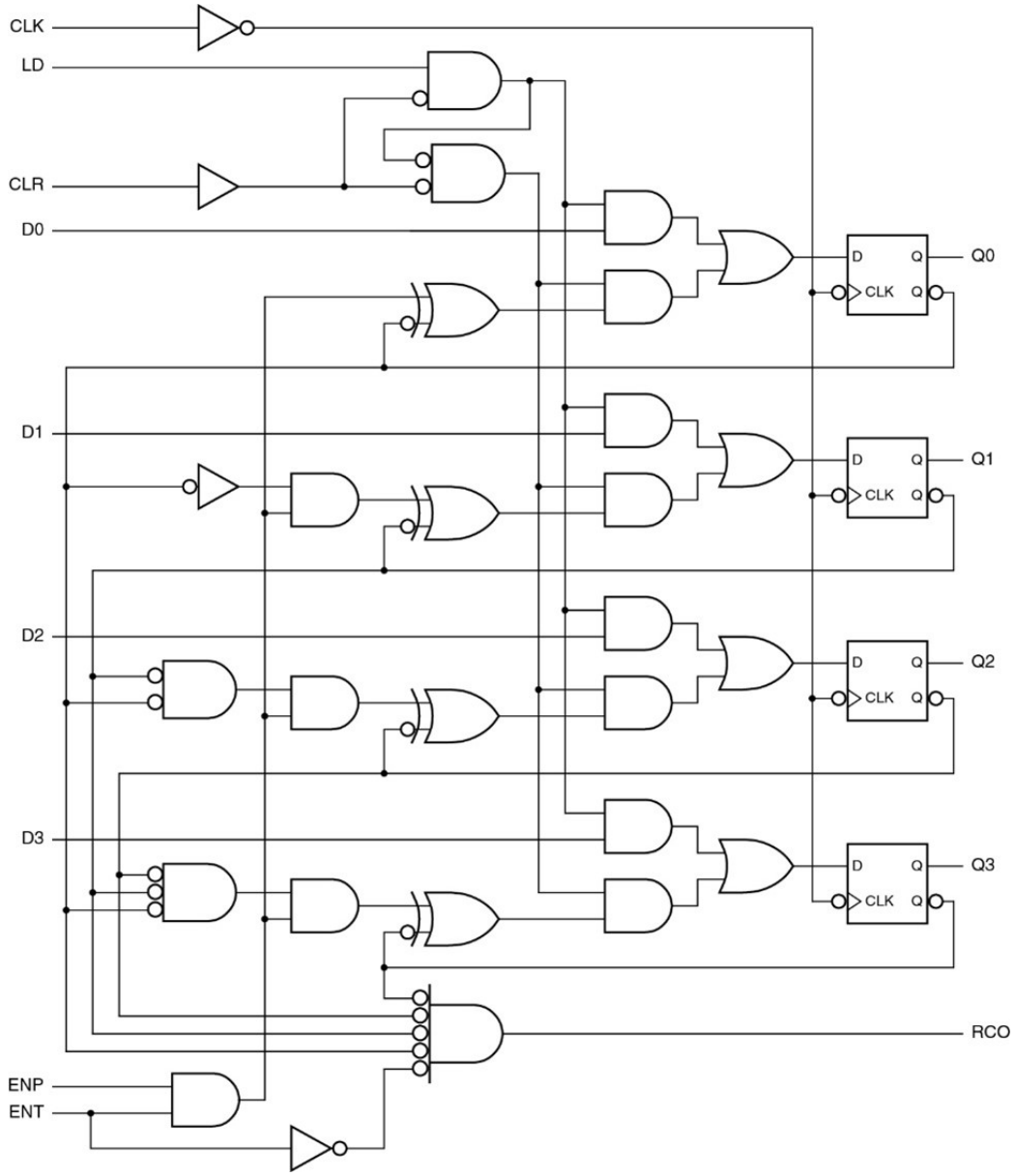

图 1: 原理图

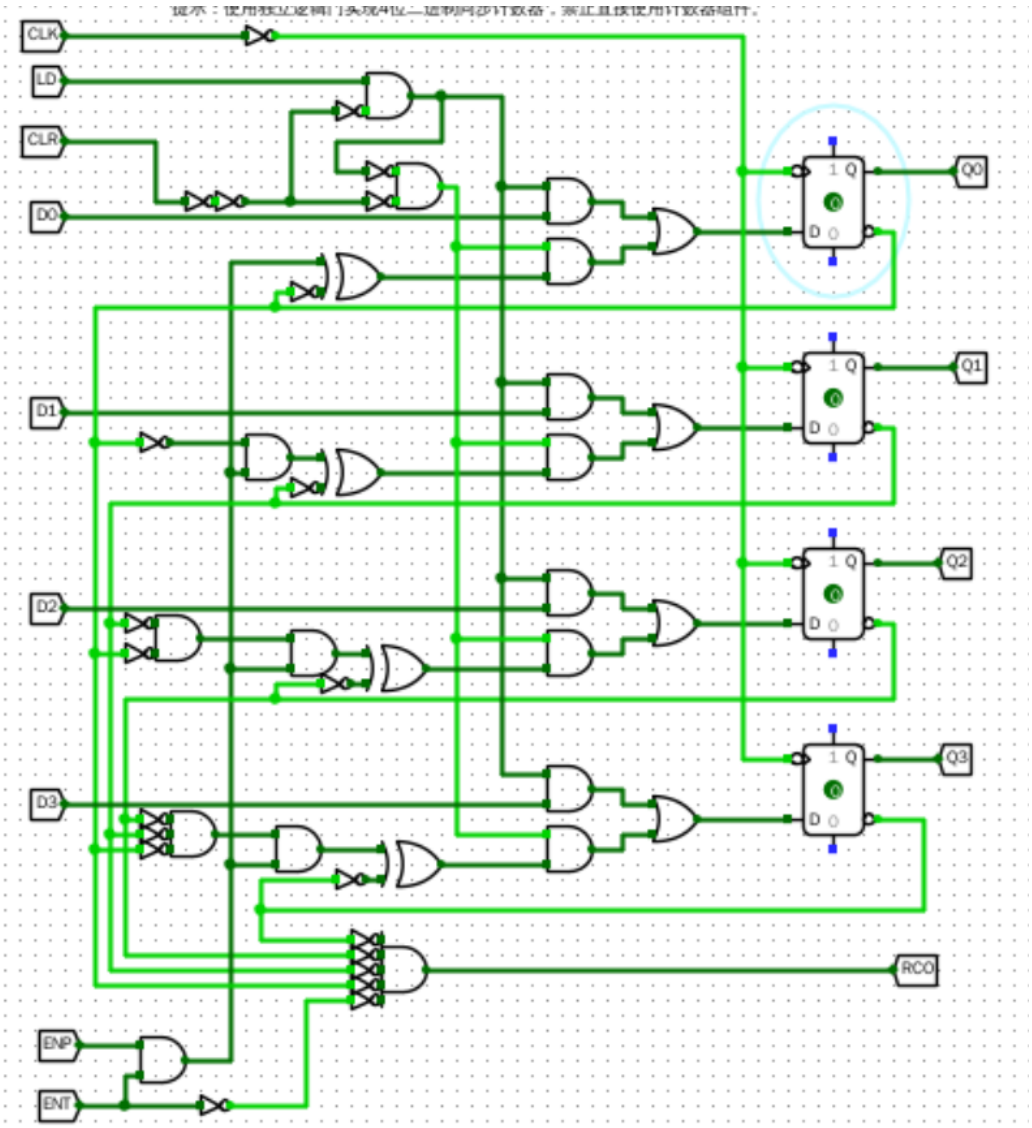

图 2: 电路图

1.3 实验数据仿真测试图

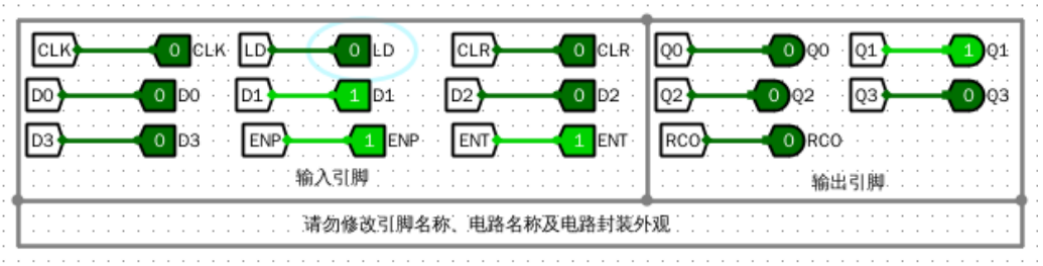

图 3: 仿真测试 1

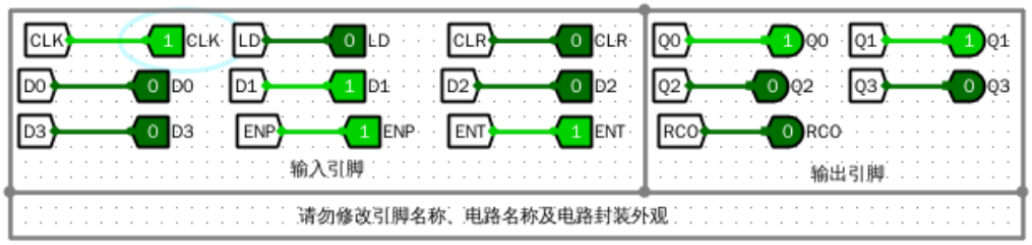

图 4: 仿真测试 2

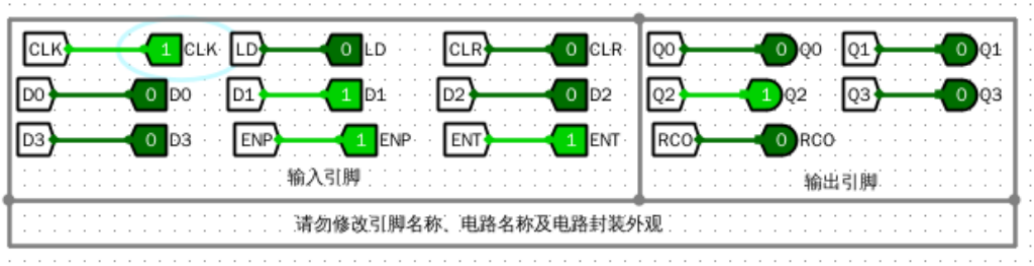

图 5: 仿真测试 3

1.4 错误现象及分析

在本题中，其实并不太容易出现错误。因为在实验手册中已经将完整的电路实现提供了出来。只是需要注意 CLR 引脚连接的是一个缓冲器而不是反相器。

2 4 位移位寄存器设计

2.1 实验整体方案设计

本题需要设计一个 4 位移位寄存器，其中输入与输出引脚的意义分别为：CLK 依然为时钟脉冲信号，在本题中 D 触发器由时钟信号的下降沿触发。Rin 和 Lin 用于控制写入的左移或右移的信号。s_1, s_0 用于控制寄存器的功能，包括保持、右移、左移、装载，具体对应可见功能表。CLR 为清零信号，低电平有效。$D_0 - D_3$ 为初始输入信号。

当时钟信号开始后，D 触发器写入信号，并根据 s_0, s_1 的值控制其将要进行的操作。

2.2 实验原理图和电路图

	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

图 6: 功能表

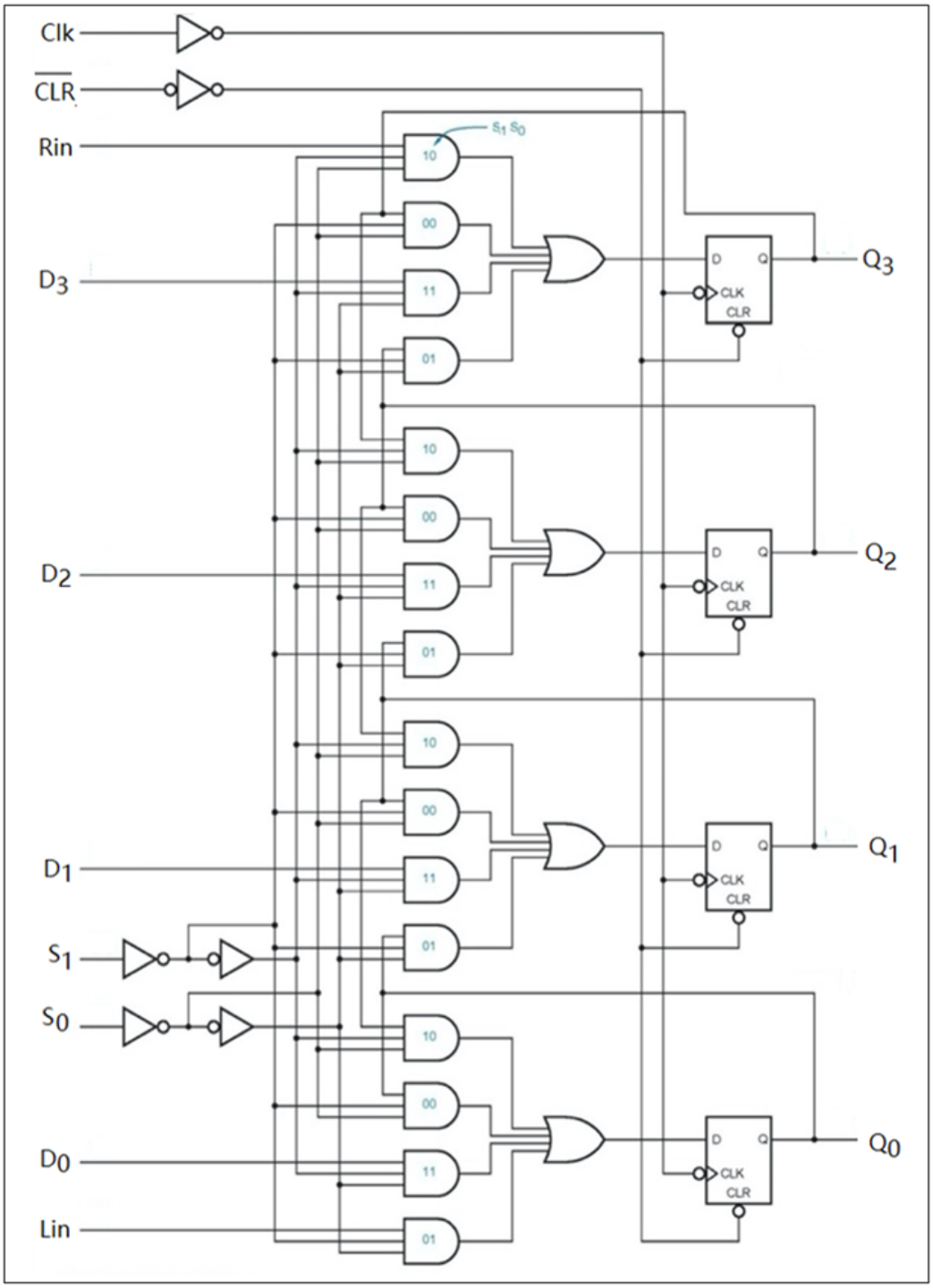
Clk
CLR
Rin
D3
D2
D1
S1
S0
D0
Lin
S1 S0
10
00
11
01
D
Q
CLK
CLR
Q3
Q2
Q1
Q0

图 7: 原理图

2.3 实验数据仿真测试图

由于本人在做题时将代码仓库重置而没有保存，所以已经通关的电路图和仿真测试图没有办法提供（已经告知助教 gg）。

2.4 错误现象及分析

这道题目的连线相对繁琐，所以处理的时候需要小心一点。对于没有通过的测试样例，可以逐一比对自己电路中信号的 0/1 是否符合预期，很有可能出现某一小块多连了一根线导致整道题出问题的情况。

3 4 位无符号数乘法器设计

3.1 实验整体方案设计

这道题有 4 个子电路，分别为 4 位二进制同步计数器、4 位二进制数加减法器、全加器和 8 位桶形移位器。由于这 4 个子电路在之前的实验中都有涉及并且并没有太大的改动（基本没变），所以就不多赘述。这里主要阐述一下主电路的思路。

类似于手算乘法，判断 X 的本位是否为 0。若为 0，则部分积右移一位得到新的部分积。若为 1，则加 X 后再右移一位。最后的结果由 P 和 Y 组成 8 位数字，其中 P 为高位，Y 为低位。同步计数器用于控制循环操作共进行 5 次，包含 1 次初始化操作。

3.2 实验原理图和电路图

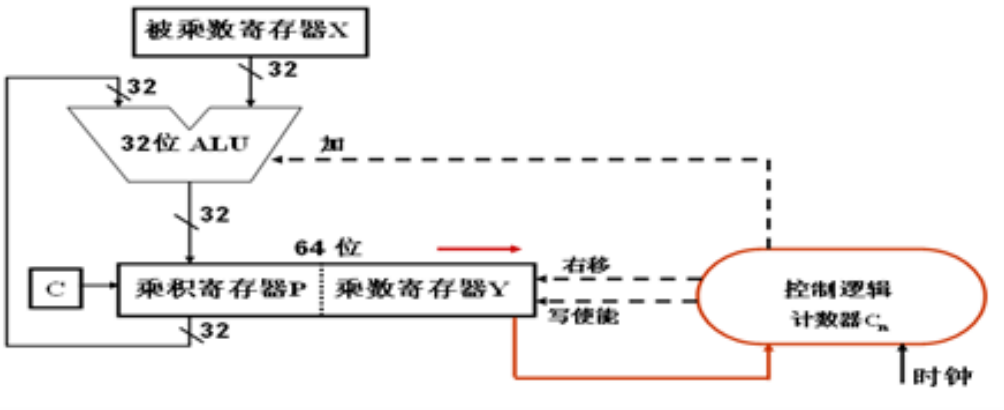

图 8: 原理图

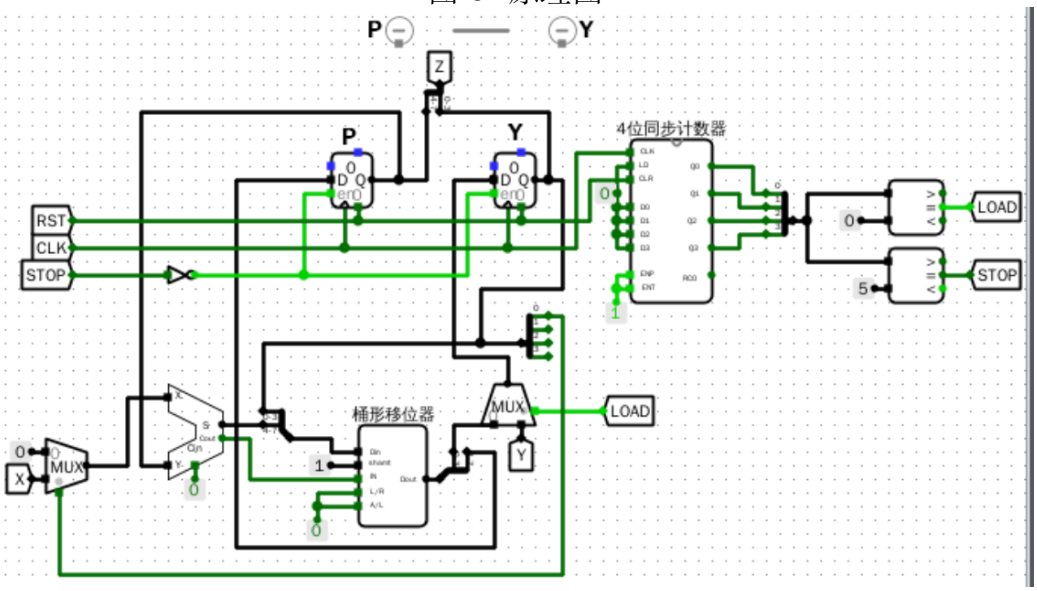

图 9: main

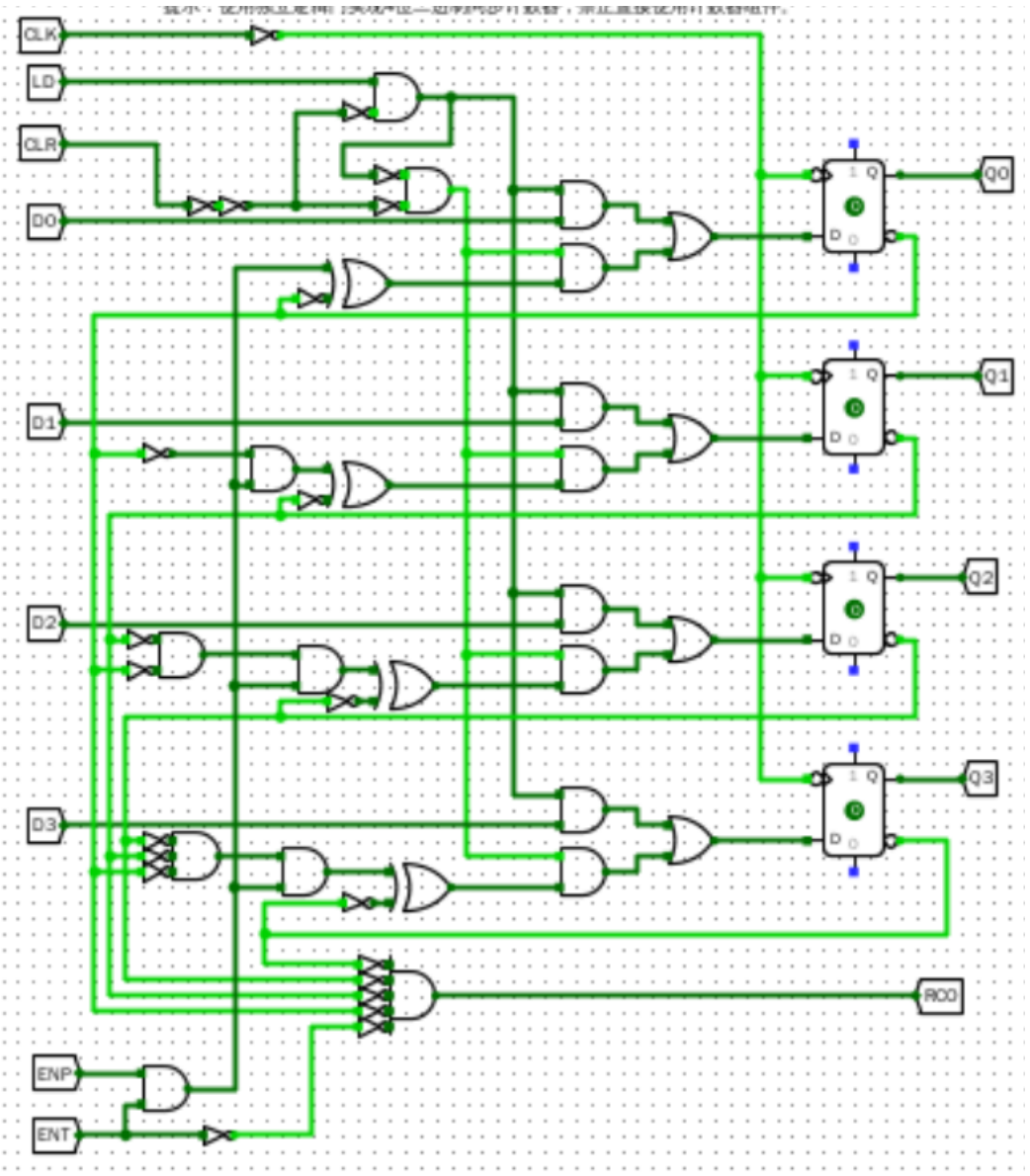

图 10: CNTR4U

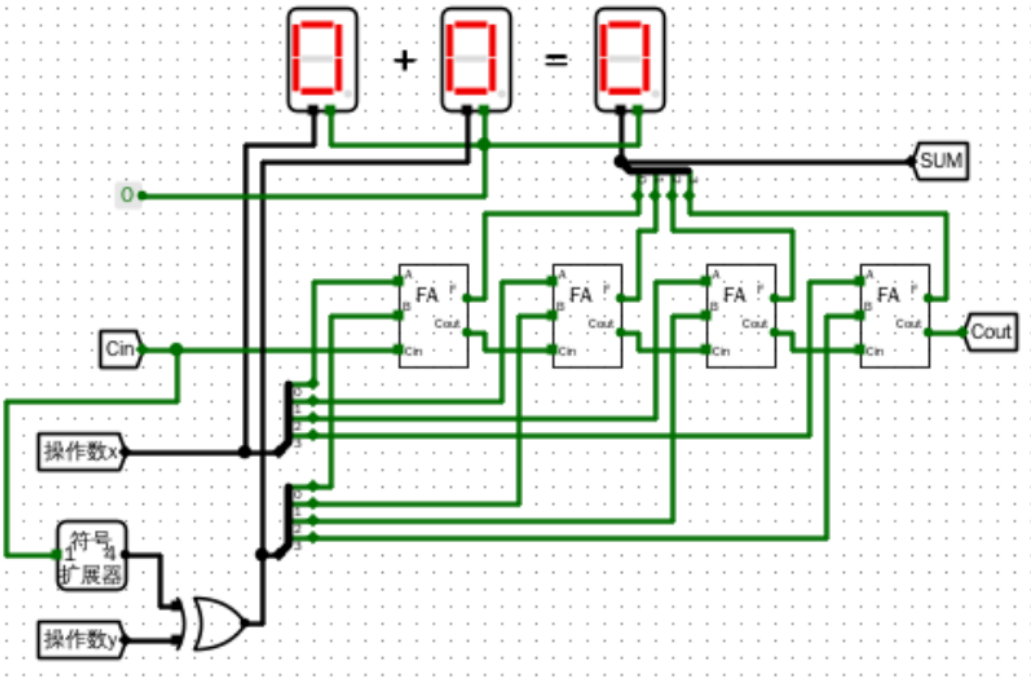

图 11: ADDER4b

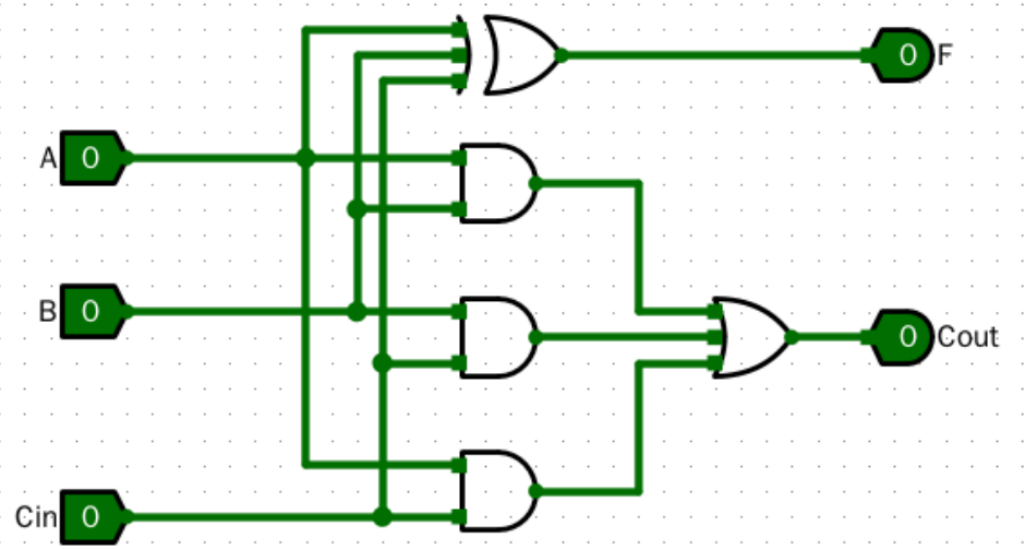

图 12: FA

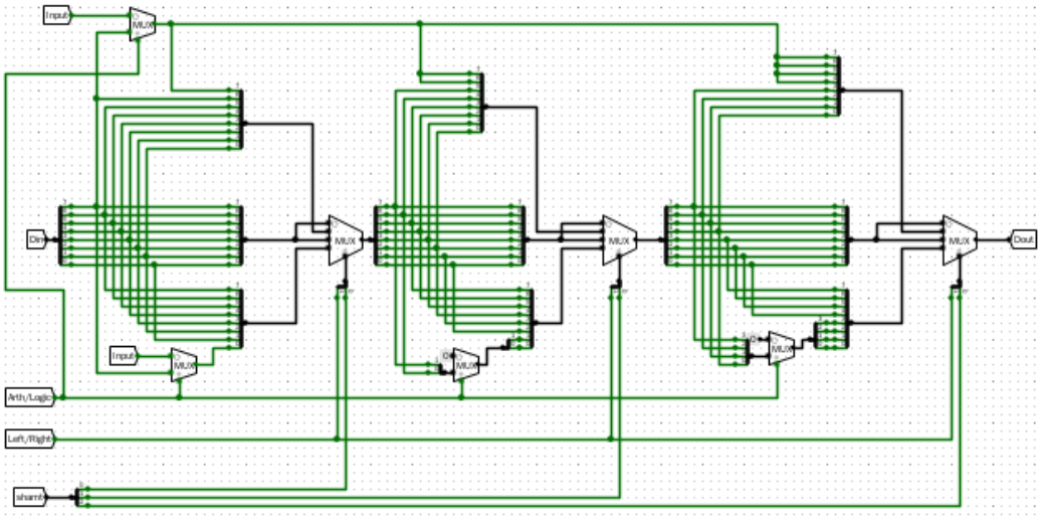

图 13: BarrelSFT8b

3.3 实验数据仿真测试图

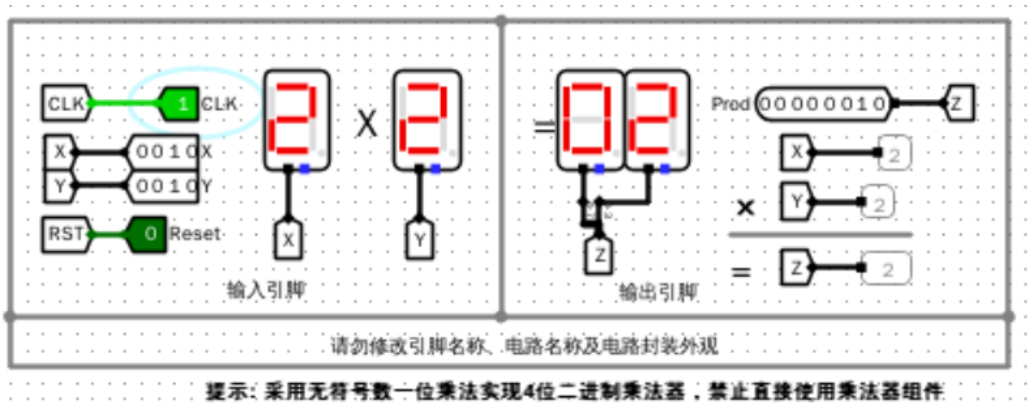

图 14: 仿真测试 1

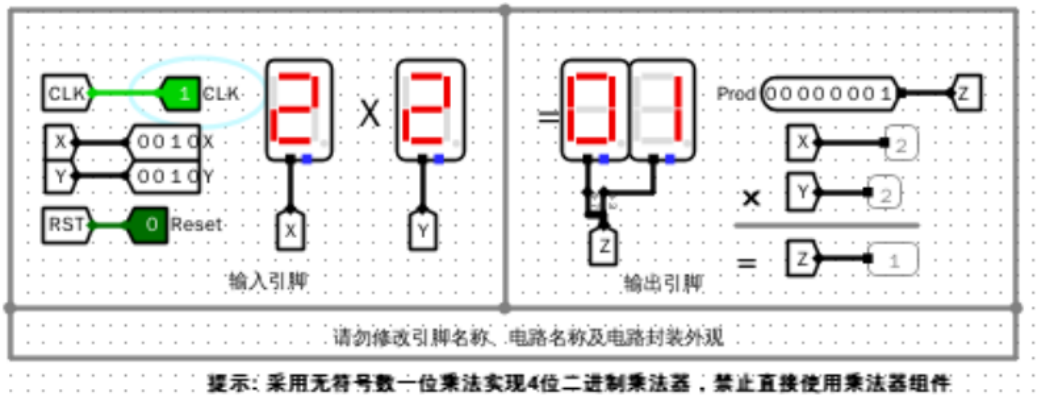

图 15: 仿真测试 2

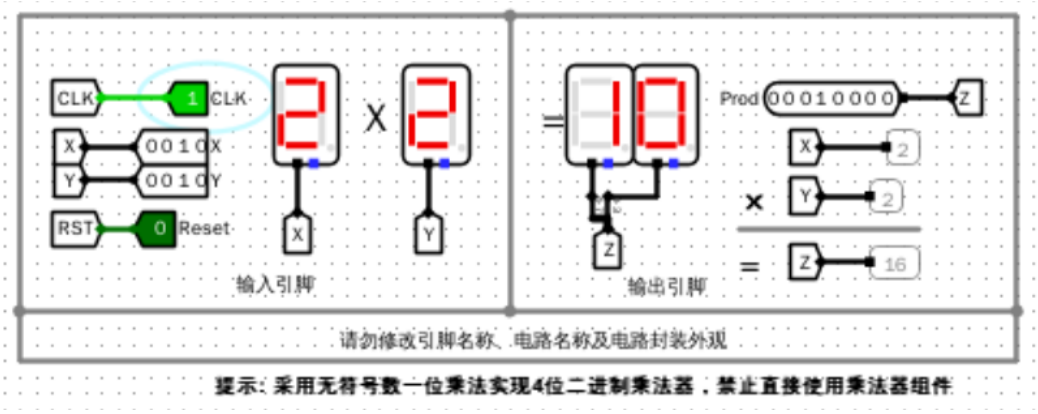

图 16: 仿真测试 3

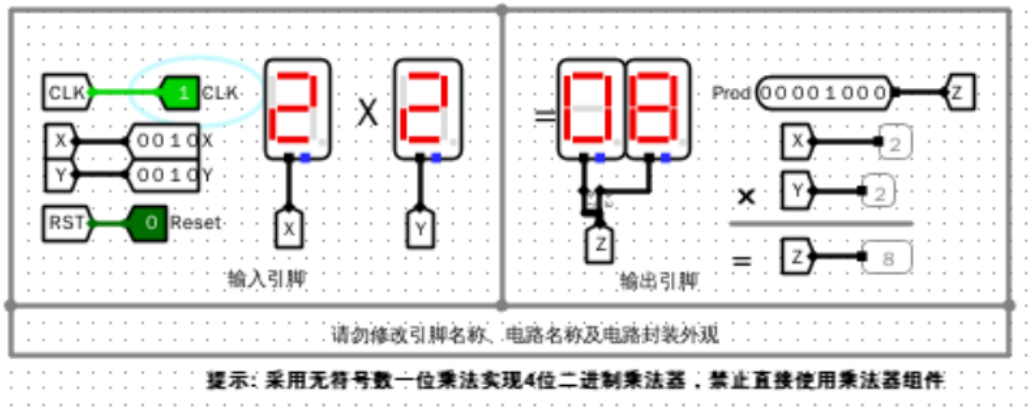

图 17: 仿真测试 4

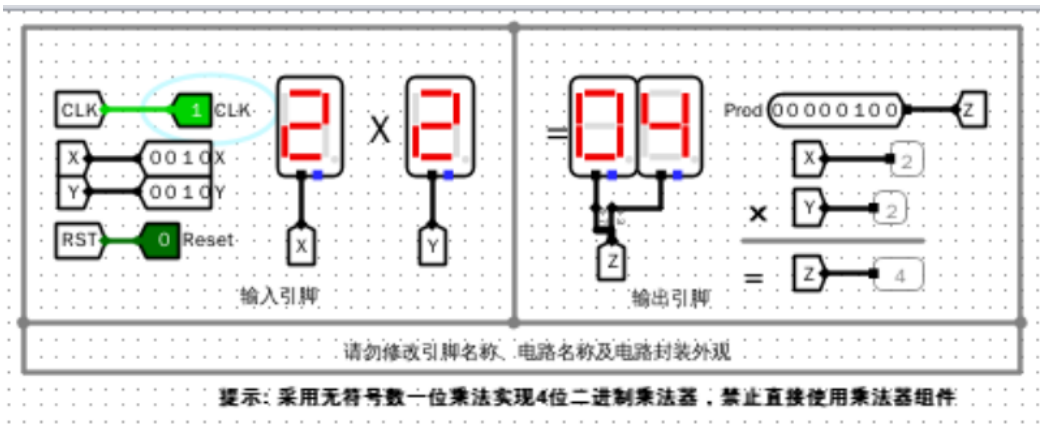

图 18: 仿真测试 5

3.4 错误现象及分析

本题的难点是合理地利用给出的各个子电路使其能按照预期实现相应的功能。但本题 debug 的难度会比较大，因为毕竟有 5 步:(

4 寄存器堆实验

4.1 实验整体方案设计

本题可以说是目前为止象征计算机“绣花”原理的集大成者，相当简单的结构，特别折磨的连线，我是不太理解为什么不能出成 4 位或者 8 位的，复杂度并没有什么变化。

下面简单说一下这道题的思路：写使能输入端 WE 控制是否在下个时钟触发边沿到来时，开始将 busW 线上的数据写入寄存器堆中。RA 和 RB 分别是读口 1 和读口 2 的寄存器编号，RW 是写口的寄存器编号。在写使能信号（WE）有效的情况下，下个时钟触发边沿到来时开始将 busW 上的信息写入 RW 指定的寄存器中。

4.2 实验原理图及电路图

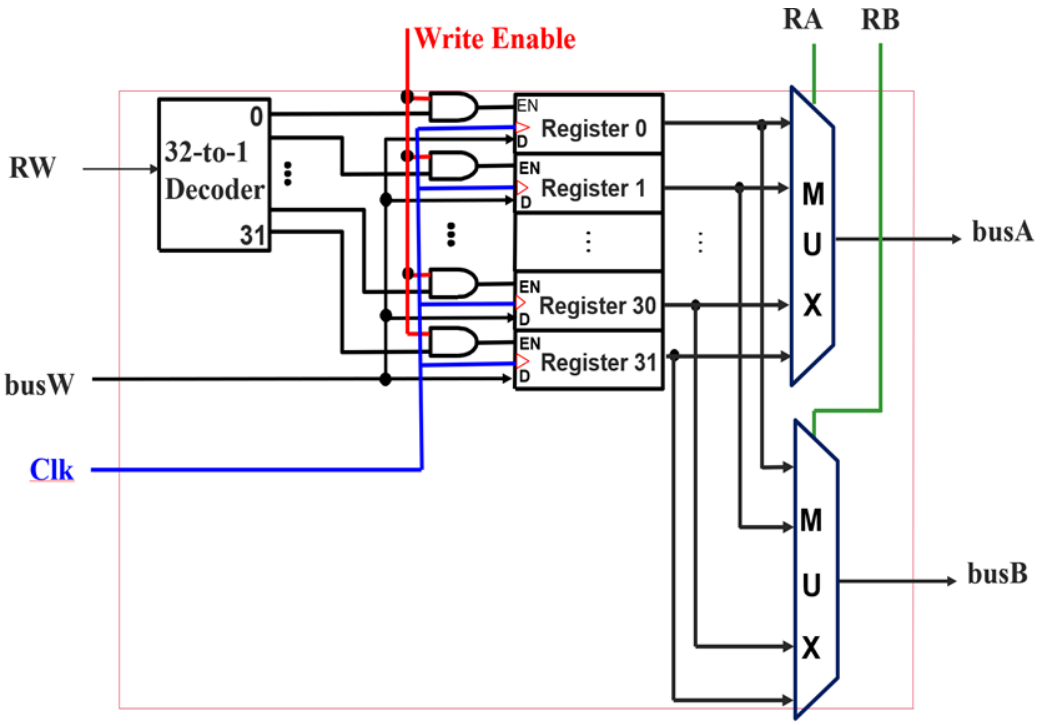

图 19: 原理图

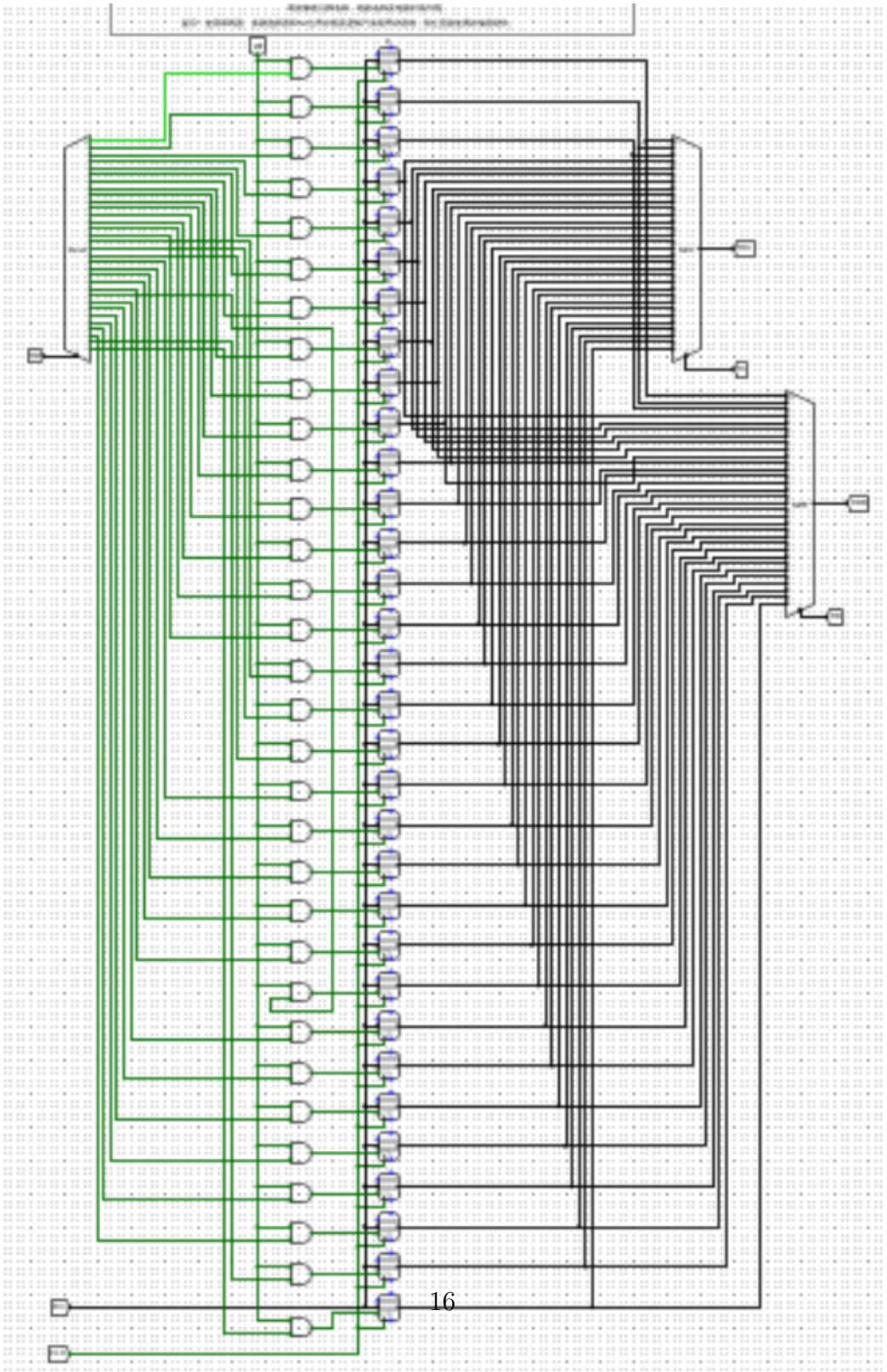

图 20: 电路图

4.3 实验数据仿真测试图

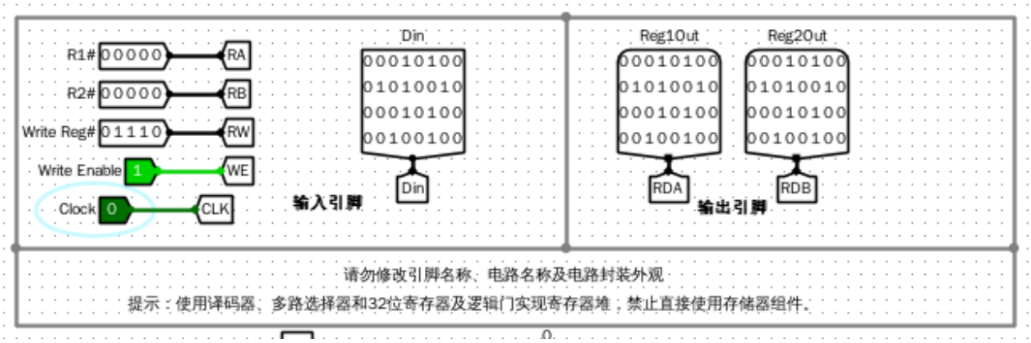

图 21: 仿真测试 1

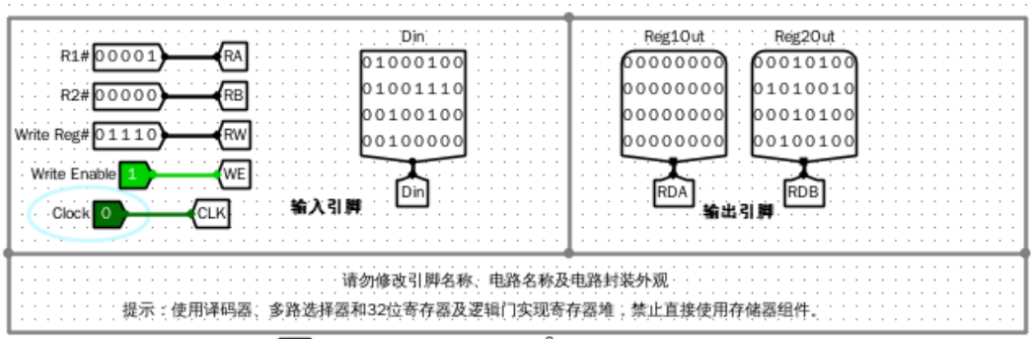

图 22: 仿真测试 2

4.4 错误现象及分析

绣花总有扎到手的时候，小心点就好了

5 数字时钟设计

5.1 实验整体方案设计

本题的目标是设计一个数字时钟，其中每经过整点 LED 会依次亮起并持续 10 个时钟周期。LED 的亮起顺序依据格雷码。主电路还需要判断输入的时间是否正确，若错误则 LD 失效。

对于 InErr，其可以通过 Logisim 内置的比较器和一些与或门实现，需要注意的是只有当 LD 信号为 1 时 InErr 信号才有可能为 1.

对于 LED，本题采用了 CNTR4U 这个 4 位同步计数器来实现。对于 LED 的输出则简单采用异或门进行处理。时钟周期从 1 到 a 共计数 10 次。

对于主电路，需要注意每个 CNTR4U 的 ENP 和 ENT 信号只有在需要进位的情况下才会生效，只有最后一个秒位才始终为 1.

5.2 实验原理图及电路图

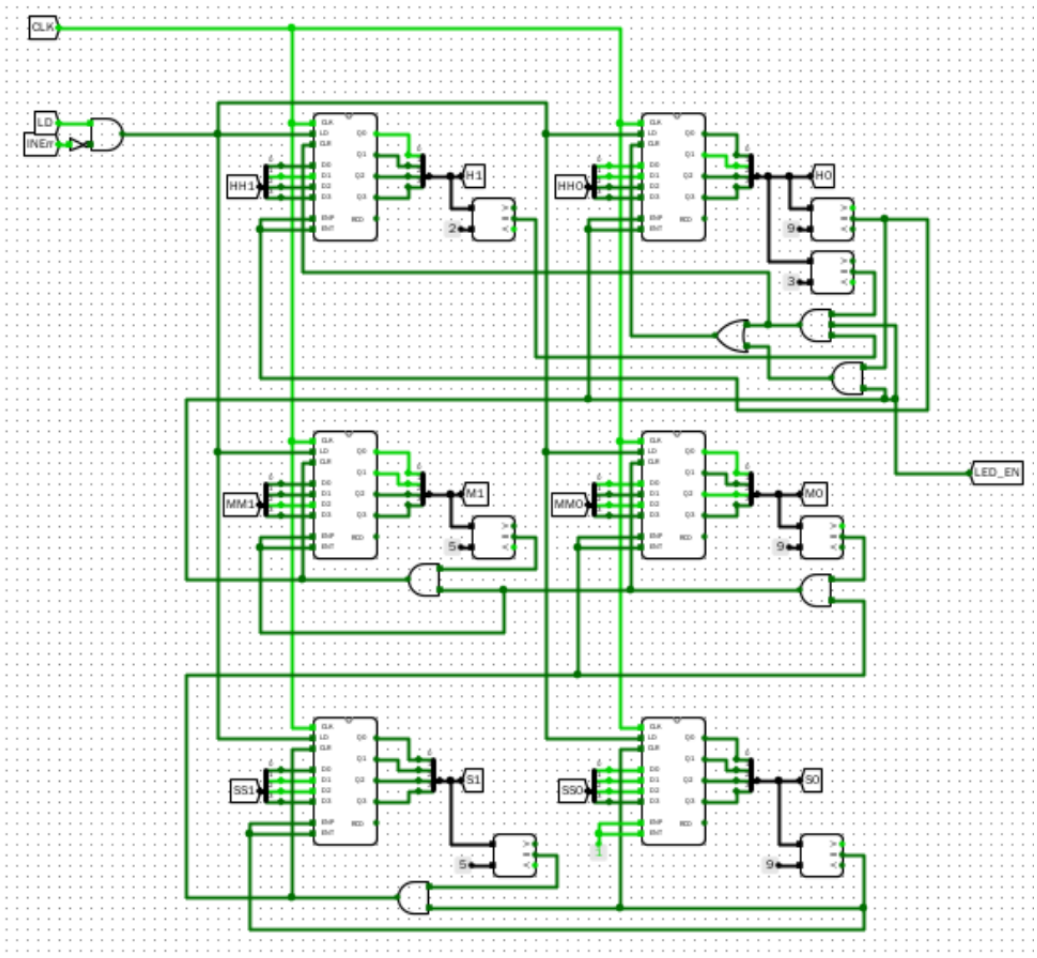

图 23: main

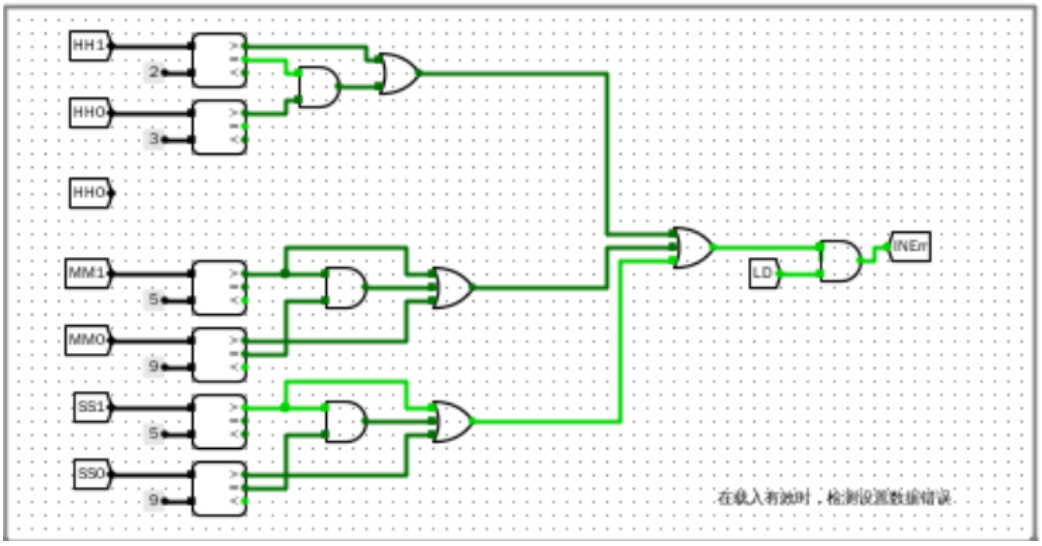

图 24: InErr

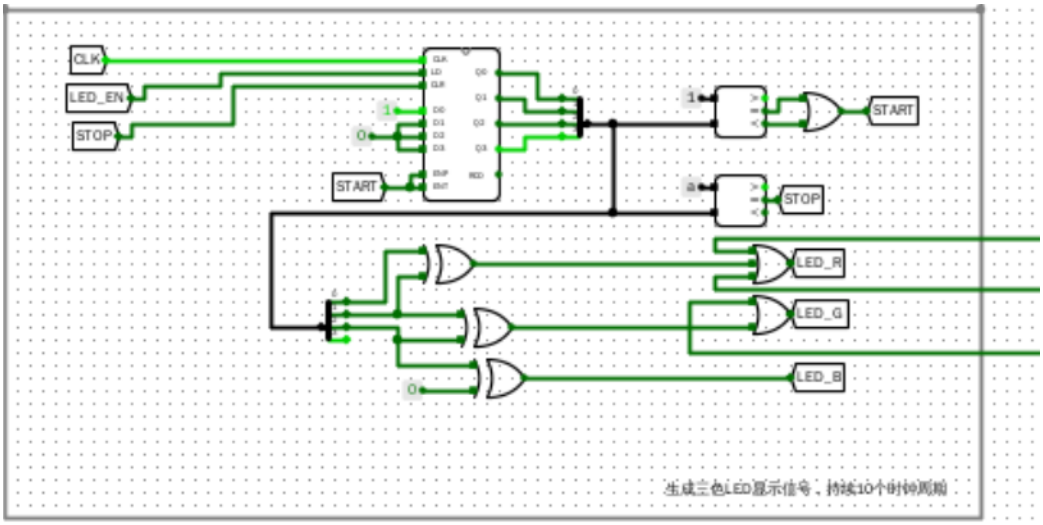

图 25: LED

5.3 实验数据仿真测试图

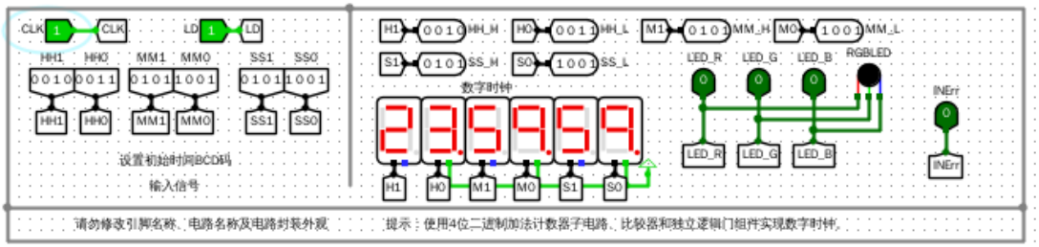

图 26: 仿真测试 1

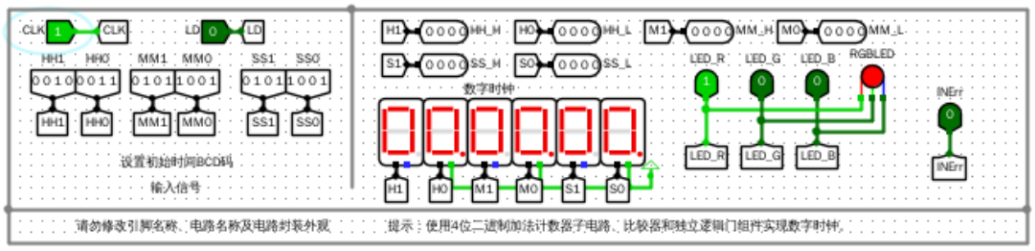

图 27: 仿真测试 2

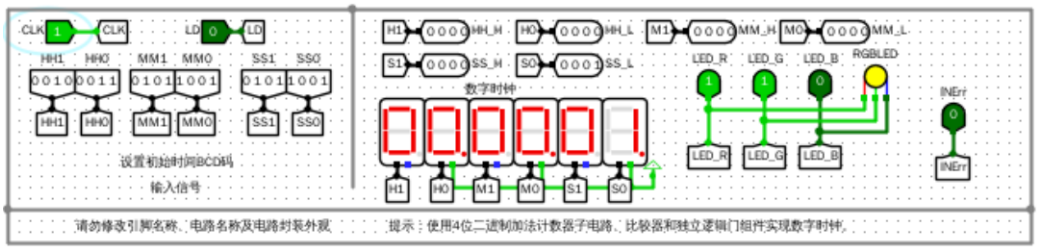

图 28: 仿真测试 3

5.4 错误现象及分析

本题有一点面向 OJ 编程的意思，有一些小的点题目描述的并不是十分明确，需要针对 OJ 给出的结果来修正自己的理解。

6 思考题

6.1 修改同步计数器电路设计，实现异步清零和异步置数的 4 位计数器，并说明该计数器的应用特性。

若想要将当前的电路变为异步清零和异步置数，则需要去除当前 CLR 和 LD 信号对 CLK 信号的等待，即若 CLR 为 0，直接将 $D_0 - D_3$ 置为 0，若 LD 为 1，直接将 $D_0 - D_3$ 信号载入 D 触发器的输入端。

可以通过更换一个支持异步清零和异步置数的 D 触发器来实现相应的功能。

这类计数器允许外部事件实时控制寄存器的状态，反应速度更快。

6.2 如何仅利用 4 位同步加法计数器在不增加其它逻辑门的情况下，实现模 10 计数器。

可以将第五题数字时钟设计中 LED 的设计，即对输出与 1010 进行比较，若输出大于或等于 10 就强制触发 LD 信号，将 $D_0 - D_3$ 分别置为 1000 并传入同步加法计数器。

这里 ChatGPT 给出的说法是计数器自带终端检测信号 TC，若达到 10 则自动将 LD 置为 1，给出的例子是 74 系列芯片。我认为其内部实现也需要与 10 进行比较。

6.3 利用 4 位移位寄存器设计 8 位二进制伪随机序列电路，写出输出序列值。

可以通过令输出值为 $Q_2 \oplus Q_3$，并将上一次的输出值置为 4 位数右移的输入。将此过程进行 8 次得到一个 8 位二进制数，可以作为得到的二进制伪随机序列。

6.4 如何实现寄存器堆的写后读功能。

顺序执行：在时序上，确保写操作先于读操作执行。这意味着寄存器堆的写入操作应优先于所有依赖此寄存器的读操作。可以使用管线调度或资源锁来控制。

数据前推机制：如果在某些并行执行的场景中，写操作和读操作可能会发生在不同的时钟周期，系统需要能够动态地处理写后读问题。这通常通过更新寄存器堆的状态信息或通过重排指令来实现，以确保寄存器的写入不会被误读。

以上来自 ChatGPT，个人认为实际实现中应该要保证随时写随时读，即需要动态处理，可以设计成每当读取或写入后将寄存器恢复至初始状态。