華中科技大學

2022

数字电路与逻辑设计
实验报告

专　　业：计算机科学与技术

班　　级：计卓 2101

学　　号：U202112071

姓　　名：王彬

电　　话：15005276201

邮　　件：wangbin2002@hust.edu.cn

完成日期：2022.12.19

实验报告及电路设计评分细则

评 分 项 目	满分	得分	备注	
文档格式（段落、行间距、缩进、图表、编号等）	15			实验报告总分
设计方案与实验过程	60			
遇到的问题及处理	10			
设计方案存在的不足	5			
心得（含思政）	5			
意见和建议	5			
电路（头歌）	100			
教师签名			日　　期	

备注：实验过程将从电路的复杂度、是否考虑竞争和险象、电路的美观等方面进行评分。

实验课程总分=电路（头歌）*0.4+实验报告*0.6

目　录

1　实验概述

1.1 实验名称

运动码表系统设计。

1.2 实验目的

实验将提供一个完整的数字逻辑实验包，从真值表方式构建 7 段数码管驱动电路，到逻辑表达式方式构建四位比较器，多路选择器，利用同步时序逻辑构建 BCD 计数器，从简单的组合逻辑电路到复杂时序逻辑电路，最终集成实现为运动码表系统。

实验由简到难，层次递进，从器件到部件，从部件到系统，通过本实验的设计、仿真、验证 3 个训练过程使同学们掌握小型数字电路系统的设计、仿真、调试方法以及电路模块封装的方法。

1.3 实验环境

软件：Logisim2.15.0.2 软件一套。

平台：https://www.educoder.net

1.4 实验内容

设计一个运动码表系统，具体内容及要求如下：

输入：4 个按钮，分别为 Start、Stop、Store 和 Reset。

输出：4 个 7 段数码管显示数字，分别显示秒和百分秒。

具体功能：

（1）当按下 Start 时，计时器清零，重新开始计时；

（2）当按下 Stop 时，计时器停止计时，显示计时数据；

（3）当按下 Store 时，若当前计时数据小于系统记录，则更新系统记录，并显示当前计时数据；否则不更新系统记录，但显示系统记录。

（4）当按下 Reset 时，复位，计时=0.00, 系统记录=99.99。

1.5 实验要求

（1） 根据给定的实验包，将运动码表系统切分为一个个实验单元；

（2） 对每一个实验单元，按要求设计电路并使用 Logisim 软件进行虚拟仿真；

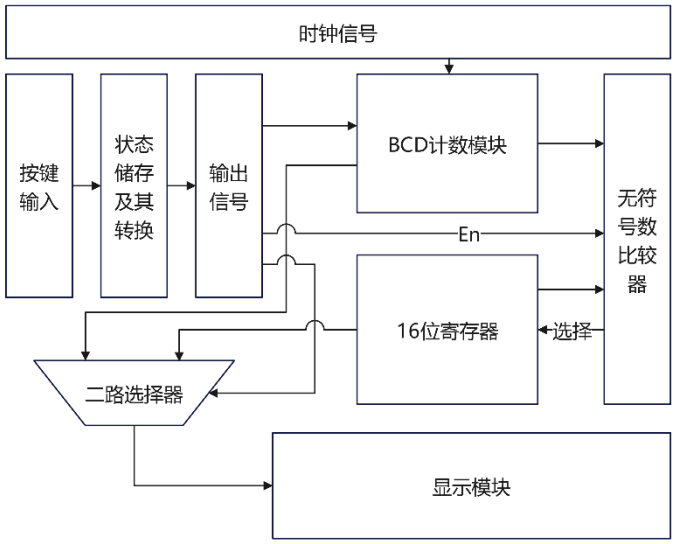
（3） 设计好的电路在 educoder 平台上提交并进行评测，直到通过全部关卡。

2　设计方案与实验过程

2.1 方案设计

我们对电路的总体结构进行设计。该运动码表包括 LED 计数电路、数码管驱动电路、16 位 2 路选择器、16 位无符号整数比较器、16 位并行加载寄存器、BCD 计数器、码表计数器、码表显示驱动和码表控制器。我们的运动码表是由这一系列组件所构成的。

通过按键输入和码表所处的状态得到运动码表的次状态，于是对该状态输出为信号以统摄全局。输出信号操控计数模块是否开始计时，如果我们按下“停止”键即可停止计数，且将本次的成绩同寄存器比较，如果较小则填充入寄存器。显示为时间或寄存器的存储成绩。因此，我们对运动码表的大致信号传递图设计如下。



2.1.1　7 段数码管驱动电路设计

我们对 7 段数码管驱动电路进行设计，通过填写真值表的方式使得其电路得以自动生成。通过数码管我们可以得到七个输出与四个输入对应的关系，填写好对应的真值，就可以得到表达式。

其真值表如下所示。

X3	X2	X1	X0	Seg_1	Seg_2	Seg_3	Seg_4	Seg_5	Seg_6	Seg_7
0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1	1	1	0
0	0	1	1	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	0	0	1
0	1	0	1	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	1	1	1
0	1	1	1	0	0	1	1	0	0	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

根据该真值表使用电路的自动生成功能实现对电路的自动生成。

2.1.2 2 路选择器（1 位）电路设计

1 位二路选择器具有三个输入段 X0， X1， Sel 和一个输出段 Out。当当 Sel 端为 0 时，输出 X0；当 Sel 端为 1 时，输出 X1。因此其电路的逻辑功能为 Out = (Sel == 0)?X0:X1；获得了其基本方法后，我们采用真值表的方式实现对其电路的生成。

其逻辑表达式为：

基于该逻辑函数实现对其电路的自动生成。

2.1.3 2 路选择器（16 位）电路设计

16 位二路选择器可以视作为将 16 个 1 位二路选择器并行地连接在一起，将 Sel 端视为总线，并依次对每一位进行二路选择。因而，当 Sel 端为 0 时，输出 X；当 Sel 端

为 1 时，输出 Y。

2.1.4 1 位无符号比较器电路设计

1 位无符号比较器具有两个一位输入 X 和 Y，具有三个逻辑输出 Greater，Equal 和 Less，分别以 x，y，z 作为别称。其功能为判断 X 和 Y 之间的大小关系。

根据其逻辑关系我们写出它的真值表：

Xi	Yi	x	y	z
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

通过该真值表建立电路即可。1 位无符号比较器是构建 4 位和 16 位无符号比较器的基础。

2.1.5 4 位无符号比较器电路设计

我们利用 1 位无符号比较器，构建 4 位无符号比较器。该比较器具有两个四位输入 X 和 Y。我们的判断思路为，依次比较每一位 X 和 Y 的大小，若最高位 X 比 Y 大，则 X 大于 Y；反之，则 Y 比 X 大。若第一位 X 和 Y 相等，则再比较下一位。如此反复比较四次，可以得到 X 和 Y 的大小关系。如果这四位 X 和 Y 均相等，则可以认为 X 等于 Y。

我们根据这种思路，可以列出 Greater 的逻辑表达式为：

```
Greater  =   X3>Y3
           + X3=Y3 & X2>Y2
           + X3=Y3 & X2=Y2 & X1>Y1
           + X3=Y3 & X2=Y2 & X1=Y1 & X0>Y0。
```

判断相等和小于的输出和如上逻辑表达式类似。

我们根据这种逻辑表达式，对该器件的线路进行手动连接，即可得到其电路。

2.1.6 16 位无符号比较器电路设计

16 位无符号比较器和 4 位的无符号比较器同理。只需将 4 位无符号比较器中的 1 位无符号比较器更换为 4 位的即可。这样就可以实现 16 位数的逻辑比较。

2.1.7 4 位并行加载寄存器电路设计

4 位并行加载寄存器具有三个输入 Din、使能端 En 和时钟沿 CLK。它同时具有一个

输出 Q 作为对 Din 的寄存。其功能为：当 En 为 1 时，输入数据；当 En 为 0 时，保持上次输入的数据。

为了实现该 4 位并行加载寄存器，我们需要四个 D 触发器，因为 D 触发器的触发是由时钟信号决定的，所以我们可以由逻辑关系可知：只需将 D 触发器的各类接口对 Din 的每一位依次并行地连接即可。

2.1.8 16 位并行加载寄存器电路设计

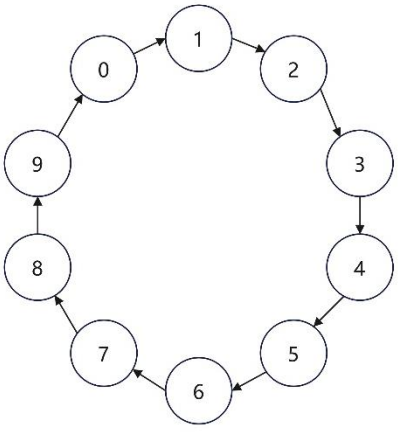
16 位并行加载寄存器和 4 位的同理。只需将 4 位并行加载寄存器中的 D 触发器更换为 4 位并行加载寄存器的即可。这样就可以实现 16 位数的并行寄存。同时，需要使用分线器将每位数据进行分段处理。

2.1.9 4 位 BCD 计数器电路设计

4 位 BCD 计数器可以存储 1 个十位数，并进行依次的累加。该计数器主要组成部分有状态转换，触发器模块和输出函数。触发器模块用以存储该计数器的状态（共 10 个状态，0..9），输出函数为该计数器的进位。我们计数器使用了 Moore 电路。Moore 电路的输出仅与现态有关，而 Mealey 的输出与现态，输入都有关。因此可以有效降低噪声，提高其稳定性。

触发器模块由四个 D 触发器构成，以存储该计数器所处的状态，并作为输出 Q 以输出。每次受到时钟信号，都会通过状态转换电路更改到下一个状态。而四个 D 触发器的状态也会由输出函数电路的运作以得到输出的解。该输出函数为 Cout = (Q==9)?1:0;

它的十个状态之间的转换关系为：



由同步时序电路状态转换表可以得到表达式。其转换表为：

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	start	stop	store	reset	NewRecord	In6	In7	In8	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
0	0	1	1	3									4	0	1	0	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									9	1	0	0	1
1	0	0	1	9									0	0	0	0	0

因此得到其逻辑表达式为：

输出: N3

$\overline{S3}\ S2\ S1\ S0 + S3\ \overline{S2}\ \overline{S1}\ \overline{S0}$

~S3 S2 S1 S0 + S3 ~S2 ~S1 ~S0

输出: N2

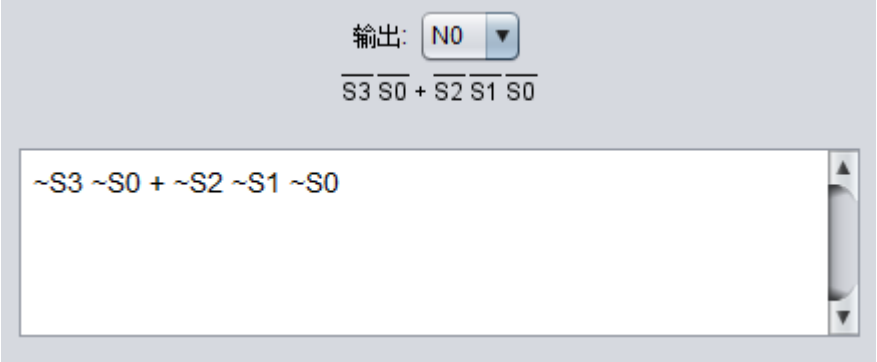
$\overline{S3}\ \overline{S2}\ S1\ S0 + \overline{S3}\ S2\ \overline{S1} + \overline{S3}\ S2\ \overline{S0}$

~S3 ~S2 S1 S0 + ~S3 S2 ~S1 + ~S3 S2 ~S0

输出: N1

$\overline{S3}\ \overline{S1}\ S0 + \overline{S3}\ S1\ \overline{S0}$

~S3 ~S1 S0 + ~S3 S1 ~S0



根据这些逻辑函数，实现电路的自动生成即可。

对于输出函数，即 BCD 计数器的进位输出，直接判断 Q 是否为 9 即可，因此可以直接写出其逻辑表达式：

```
Cout = S3 ~S2 ~S1 S0
```

通过该表达式直接对输出函数的电路进行自动生成。

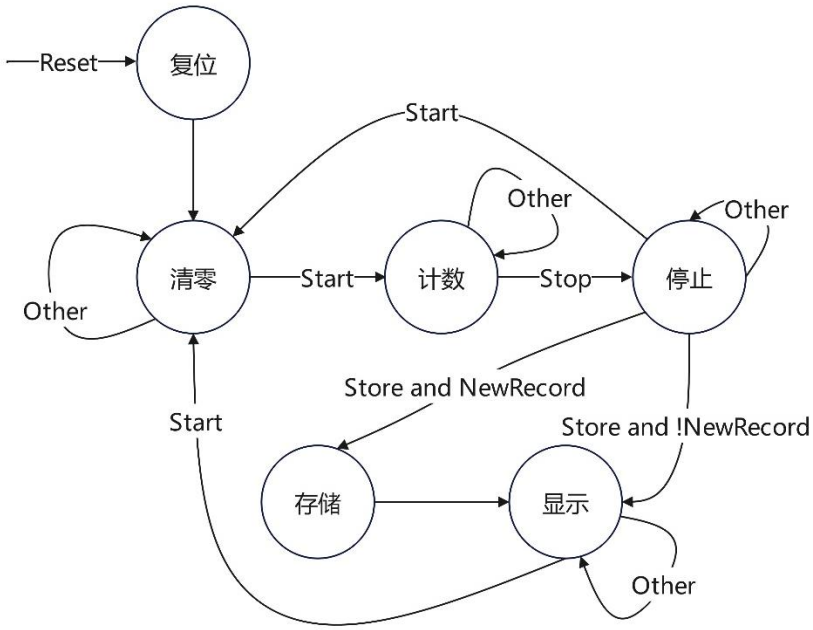
2.1.10 码表计数器及其显示电路设计

码表计数器只需将 4 个四位 BCD 计数器进行依次连接即可。而码表显示驱动只需将 4 个数码管驱动电路进行依次的连接即可。

2.1.11 码表控制器电路设计

为了实现对整个码表的控制，我们将码表控制器分为三个部分：状态转换、状态存储和输出函数。由于该运动码表有五个按键：start, stop, store, reset, NewRecord，和一个时钟端 CLK，用户的输入和触发器当前所处的状态经过状态转换函数后，转换为次态，并通过输出函数电路确定该状态下的输出信号。输出信号有 5 个，分别是 SDsel, SDen, DP-Sel, TMen, TMreset。

我们设计了六个状态，分别是复位、清零、计数、停止、显示和存储。这六个状态的功能如下：复位为清除寄存器中的记录，并将记录设为最大；清零即将计时器清零；计数即开始计时；停止为停止计时；存储为将目前的数字存储至寄存器，如果目前记录较大则不存储；显示为将寄存器中的记录显示至显像管。我们设计了该码表的状态和状态转移关系如下图：



为了使得该设计落实，我们对设计的六个状态进行编码，实现对状态的储存。由于我们共设计了六个状态，只需要三个 D 触发器即可存储我们的所有状态 ，因此我们采用三位二进制数来编码状态。我们对状态的编码如下表：

状态名	*编码*
复位	000
清零	001
计数	010
停止	011
存储	100
显示	101

根据上述设计，我们填写了状态转移关系表。该表格详细地表现了各个状态之间的转换关系，其内容如下：

				输入信号								下一状态（次态）			
S2	S1	S0	现态10进制	start	stop	store	reset	NewRecord	In6	In7	In8	次态10进制	N2	N1	N0
0	0	0	0	d	d	d	1	d				0	0	0	0
0	0	1	1	d	d	d	1	d				0	0	0	0
0	1	0	2	d	d	d	1	d				0	0	0	0
0	1	1	3	d	d	d	1	d				0	0	0	0
1	0	0	4	d	d	d	1	d				0	0	0	0
1	0	1	5	d	d	d	1	d				0	0	0	0
0	0	0	0				0					1	0	0	1
0	0	1	1	1	0		0					2	0	1	0
0	1	0	2	0	1		0					3	0	1	1
0	1	1	3	1	0		0					1	0	0	1
0	1	1	3			1	0	1				4	1	0	0
1	0	0	4				0					5	1	0	1
0	1	1	3			1	0	0				5	1	0	1
1	0	1	5	1			0					1	0	0	1
0	1	1	3	0	d	0	0	0				3	0	1	1
0	1	0	2	d	0	d	0	d				2	0	1	0
1	0	1	5	0	d	d	0	d				5	1	0	1
0	0	1	1	0	d	d	0	d				1	0	0	1
0	1	1	3	0	d	0	0	d				3	0	1	1

根据该表格所生成的码表控制器状态转换关系逻辑函数如下。

输出: N2

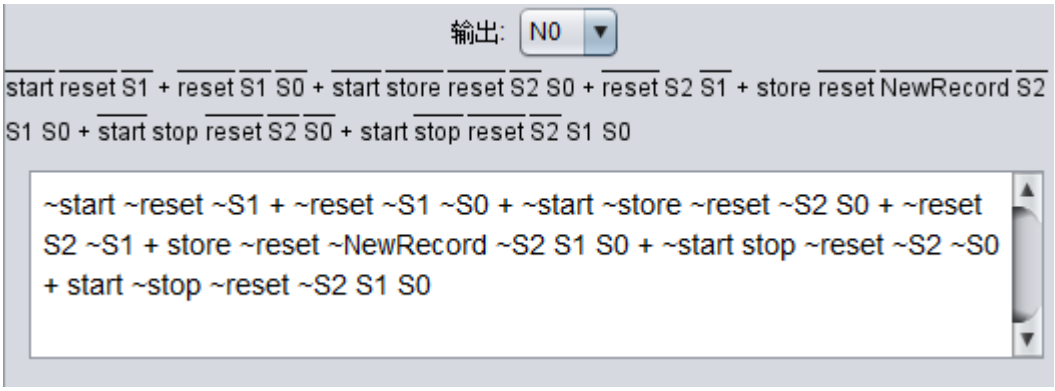
$\overline{start}\ \overline{reset}\ S2\ \overline{S1} + \overline{reset}\ S2\ \overline{S1}\ \overline{S0} + store\ \overline{reset}\ \overline{S2}\ S1\ S0$

~start ~reset S2 ~S1 + ~reset S2 ~S1 ~S0 + store ~reset ~S2 S1 S0

输出: N1

$\overline{start}\ \overline{store}\ \overline{reset}\ \overline{S2}\ S1 + \overline{start}\ \overline{reset}\ \overline{S2}\ S1\ \overline{S0} + \overline{stop}\ \overline{reset}\ \overline{S2}\ S1\ \overline{S0} + start\ \overline{stop}\ \overline{reset}\ \overline{S2}\ \overline{S1}\ S0$

~start ~store ~reset ~S2 S1 + ~start ~reset ~S2 S1 ~S0 + ~stop ~reset ~S2 S1 ~S0 + start ~stop ~reset ~S2 ~S1 S0

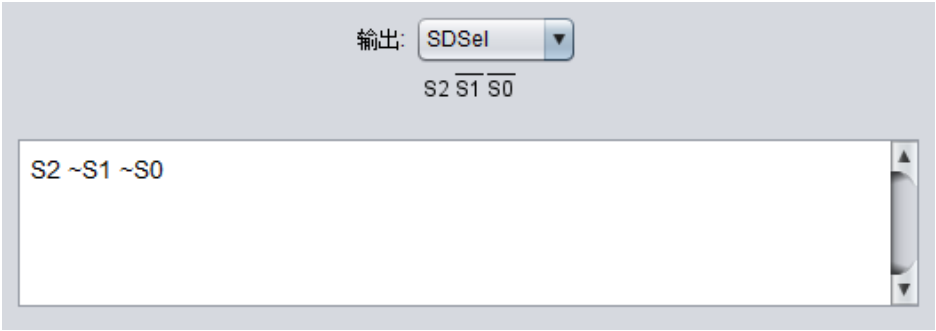


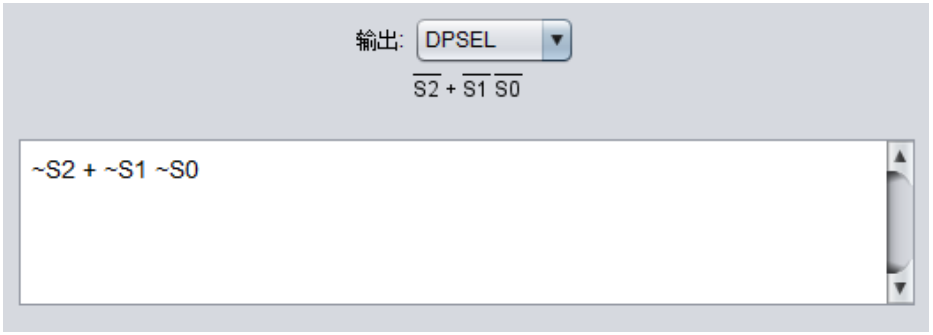
根据这几个逻辑表达式，实现状态转移函数的电路自动生成即可。

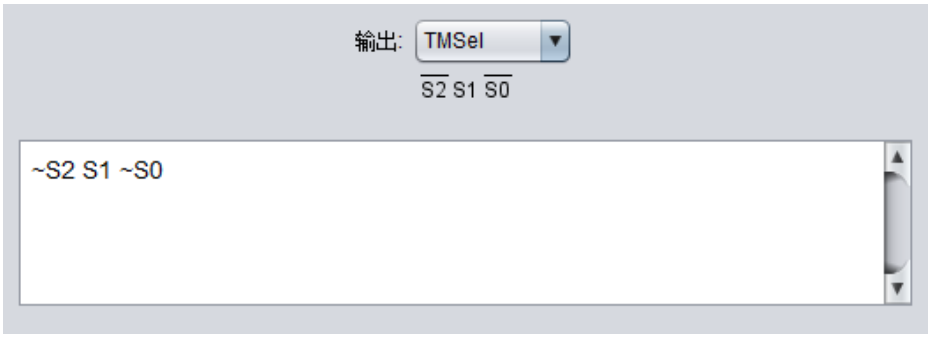
对于码表控制器的输出函数，我们写出每个状态对应的输出信号如下表。

当前状态(现态)					输入信号											
S3	S2	S1	S0	现态10进制	start	stop	store	reset	NewRecord	…	…	SDsel	SDen	DPsel	TMen	TMreset
0	0	0	0	0									1	1		1
0	0	0	1	1										1		1
0	0	1	0	2										1	1	
0	0	1	1	3										1		
0	1	0	0	4								1	1	1		
0	1	0	1	5												

我们根据该输出函数的函数表，生成了现态的输出如下。







根据这些状态至信号函数的映射，我们对码表控制器输出函数电路进行自动生成即可。

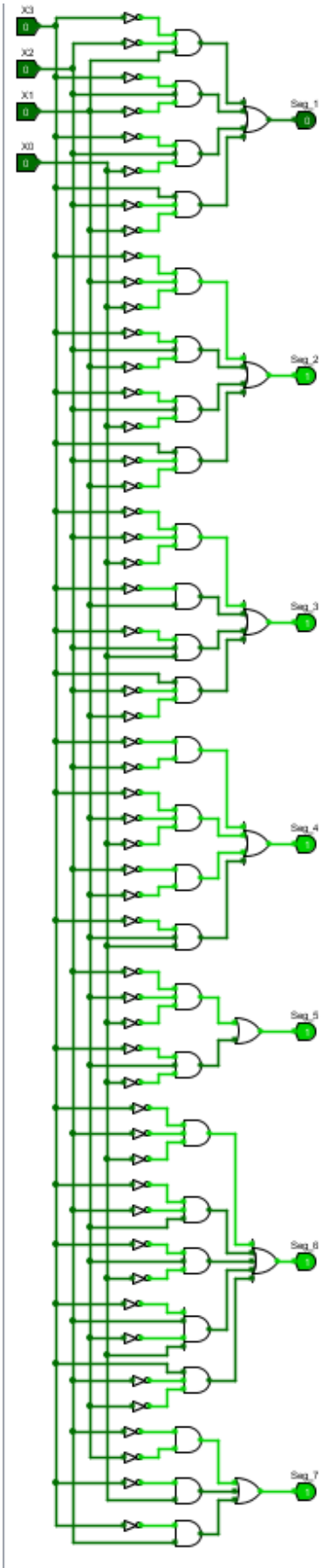
2.2 实验过程

按照设计方案，按照模块给出电路图以及测试图，对于测试图要求描述清楚模块的功能并对测试用例进行说明，最后进行测试分析。

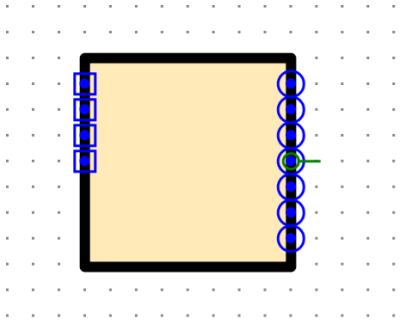
2.2.1 7 段数码管驱动电路

（1）电路图

该模块的内部结构电路图如下。

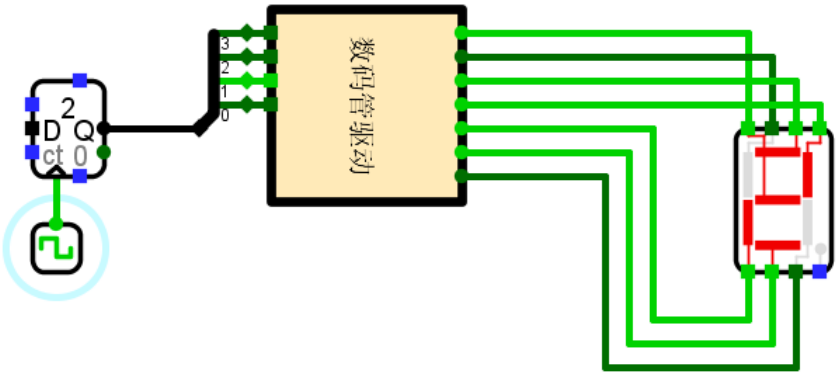

X3
X2
X1
X0
Seg_1
Seg_2
Seg_3
Seg_4
Seg_5
Seg_6
Seg_7

该模块的封装电路图如下。



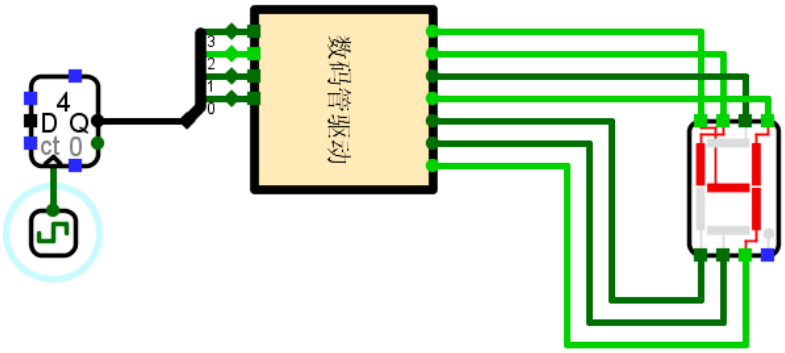
（2）测试图

当状态为 2 且时钟端为上升沿时，其数码管驱动使得输出显像为 2.



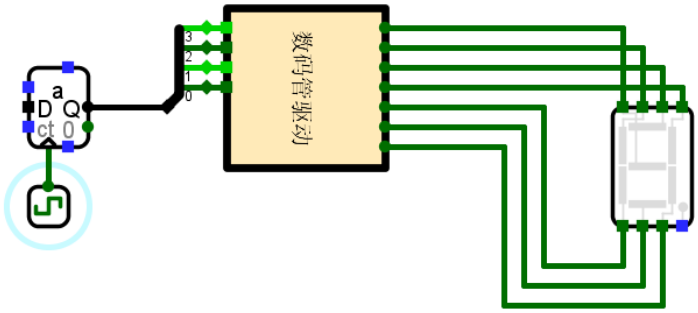
Ctrl+T驱动时钟单步运行测试

当状态为 4 且时钟端为下降沿时，其数码管驱动不改变，使得输出显像为 4.



Ctrl+T驱动时钟单步运行测试

当状态为 a 且时钟端为下降沿时，其数码管驱动不改变。且由于状态 a 不合法，不是合规的输入，使得输出不显像。



Ctrl+T驱动时钟单步运行测试

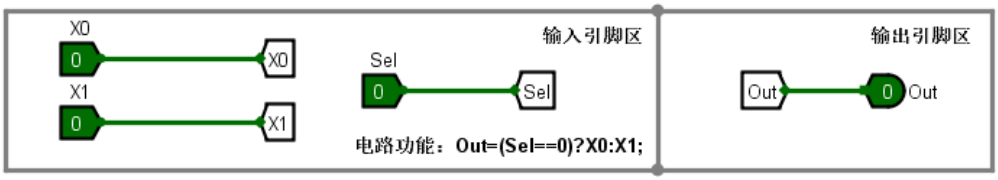
（3）测试分析

由于输出的恰为 7 段数码管驱动的信号，使得输出正确。

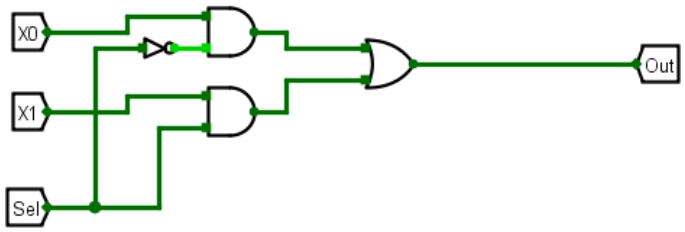
2.2.2 2 路选择器（16 位）电路

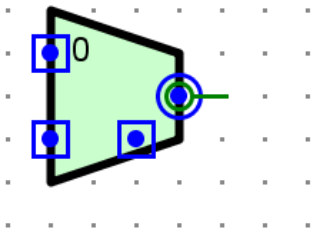
（1）电路图

2 路选择器（1 位）的电路图和封装图如下。

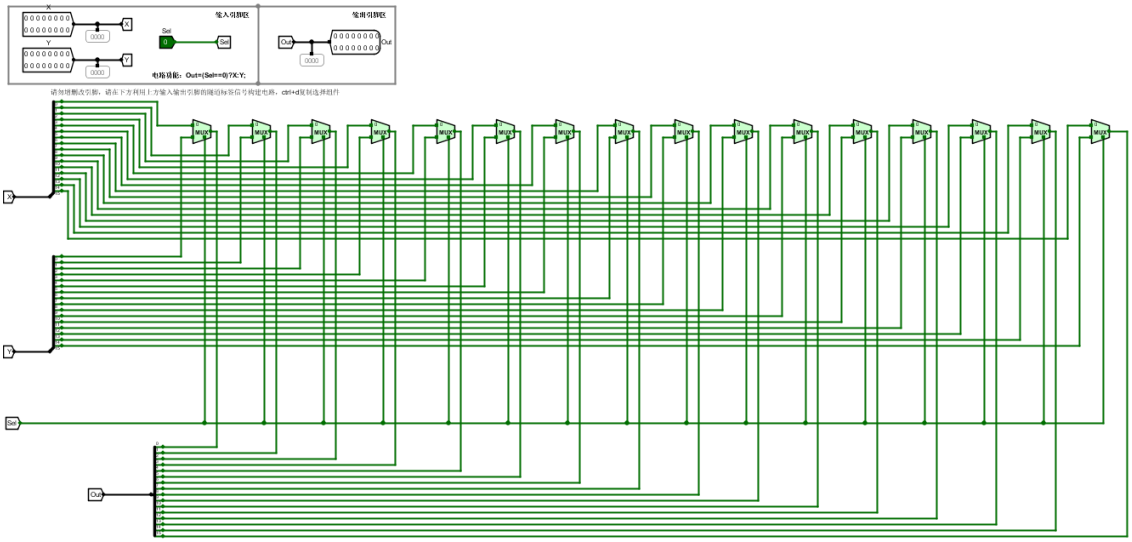


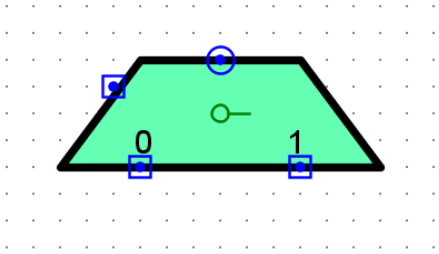
请勿增删改引脚，请在下方利用上方输入输出引脚的隧道标签信号构建电路，ctrl+d复制选择组件





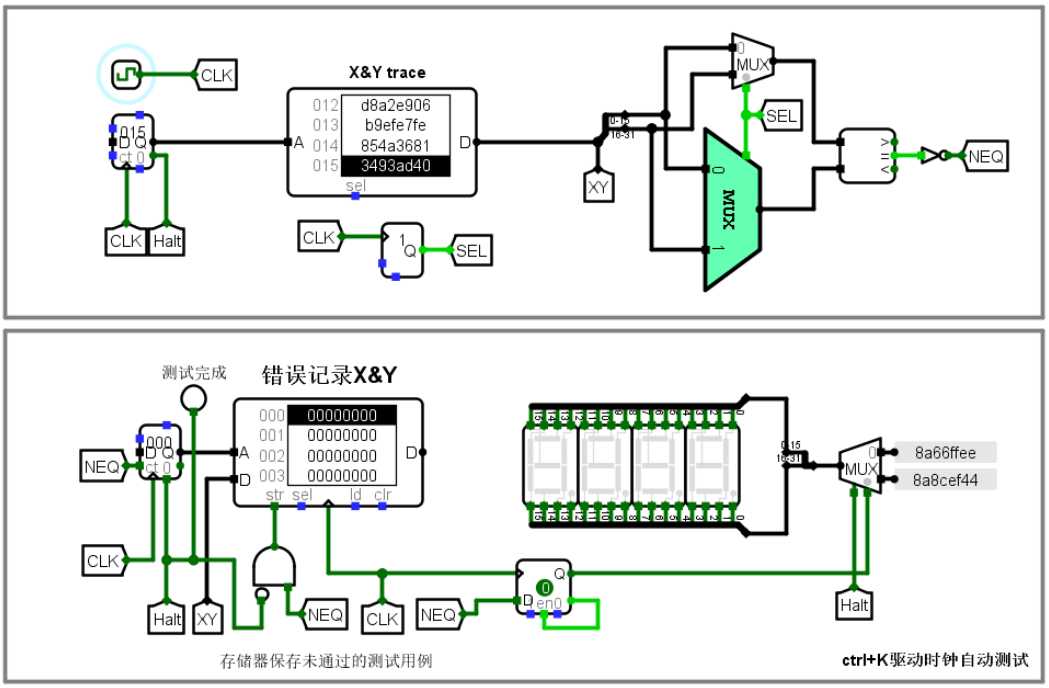
我们将 16 个 1 位 2 路选择器并行连接在一起，以构成 16 位二路选择器。2 路选择器（16 位）的电路图和封装图如下。



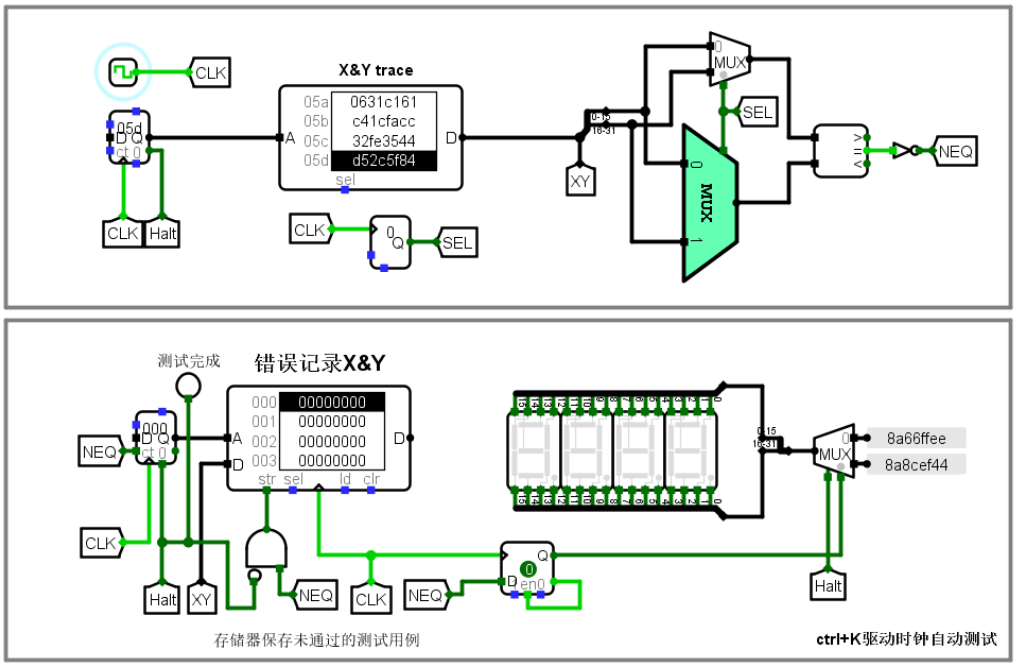


（2）测试图

由于该实验电路较为简易，我们使用自动测试器测试了一段时间，未见异常抛出。



多路选择器自动测试



多路选择器自动测试

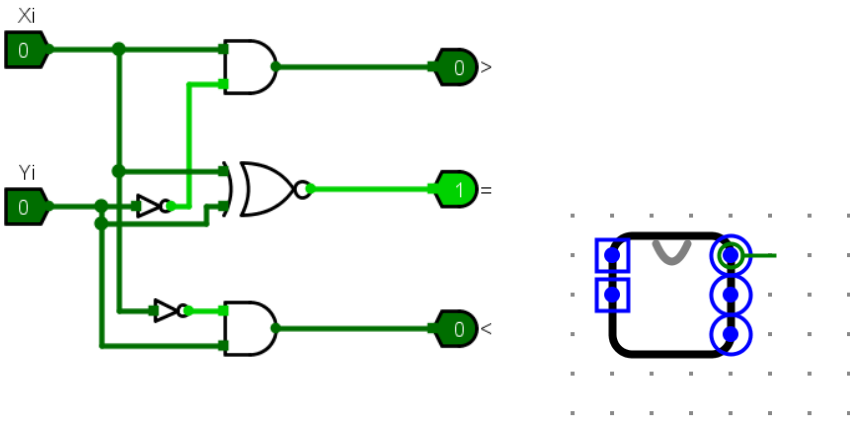
（3）测试分析

如上图，未见异常。

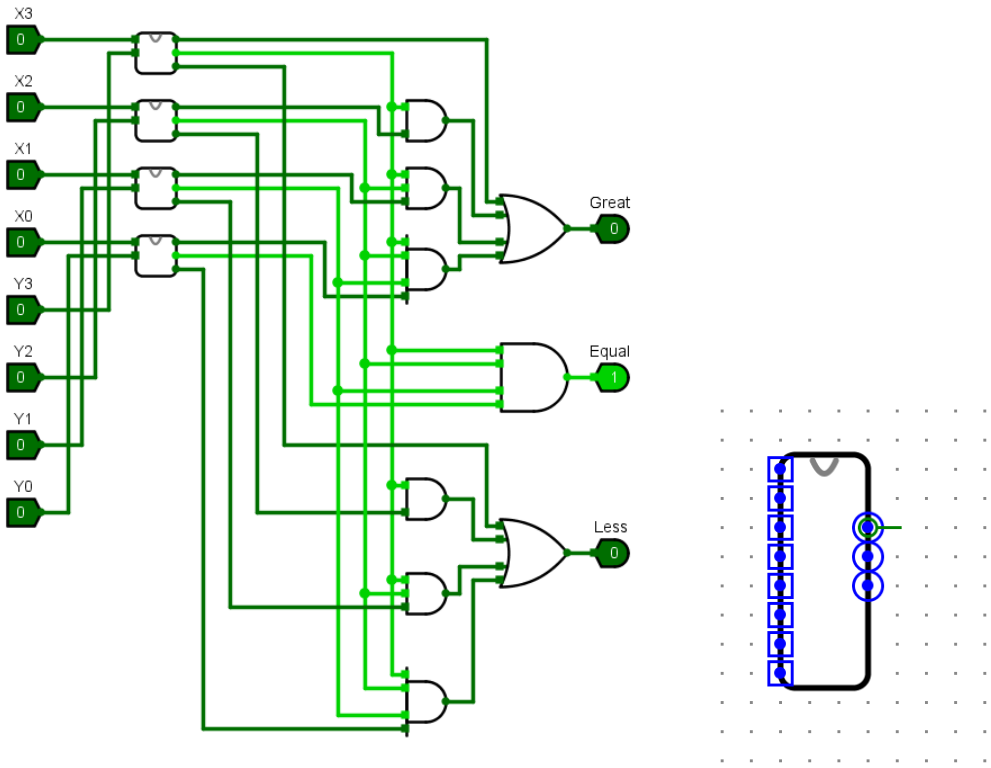
2.2.3 16 位无符号比较器电路

（1）电路图

1 位无符号比较器的电路图和封装图如下。

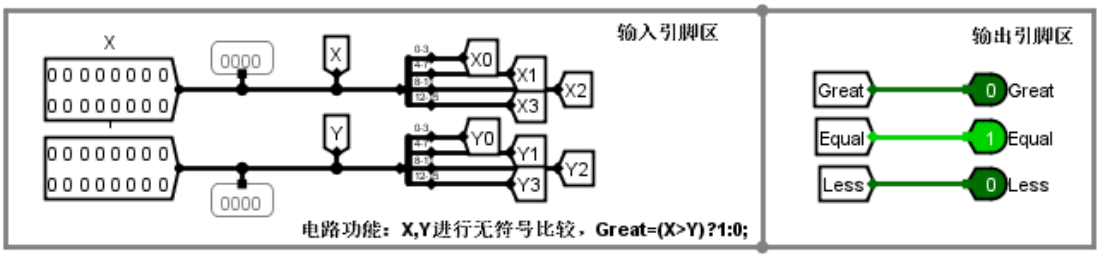


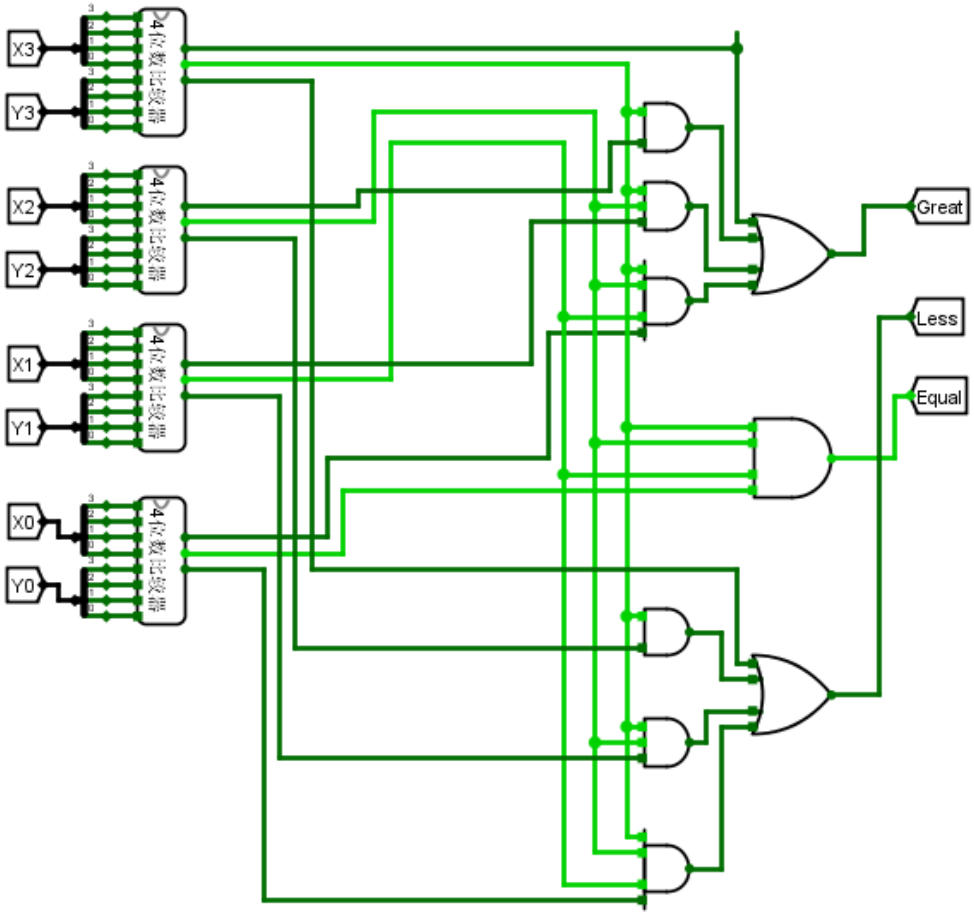
我们将 4 个 1 位无符号比较器并行连接在一起，加之以一些逻辑结构，以构成 4 位无符号比较器器。4 位无符号比较器的电路图和封装图如下。

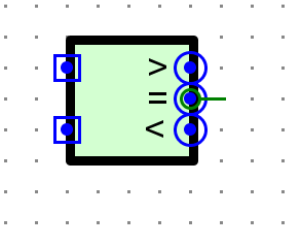


我们将 4 个 4 位无符号比较器并行连接在一起，加之以一些逻辑结构，以构成 16

位无符号比较器器。16 位无符号比较器的电路图和封装图如下。

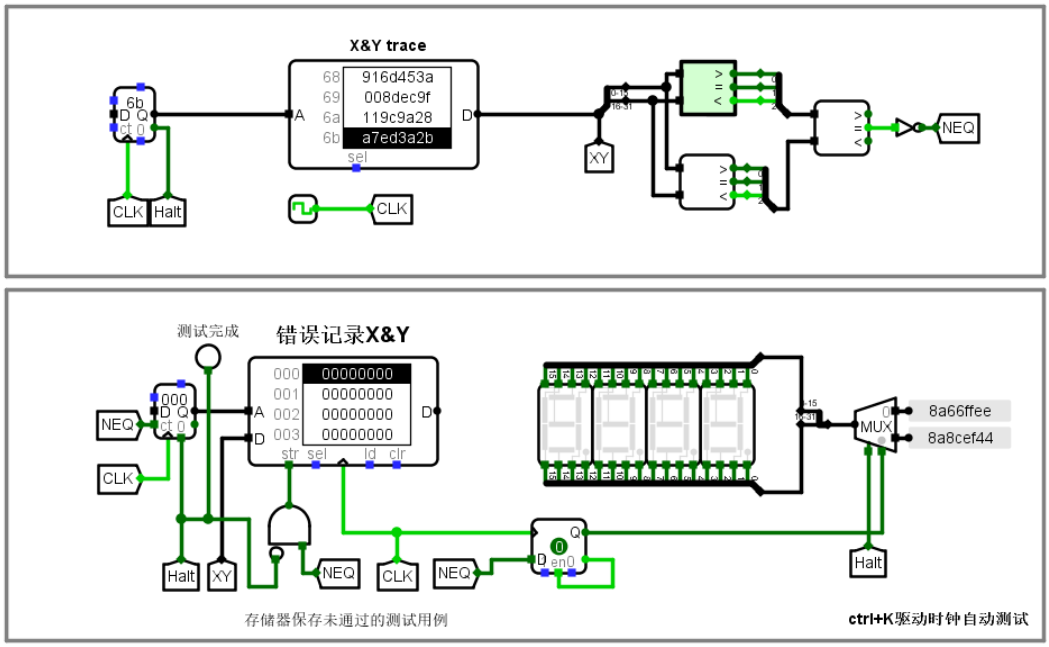




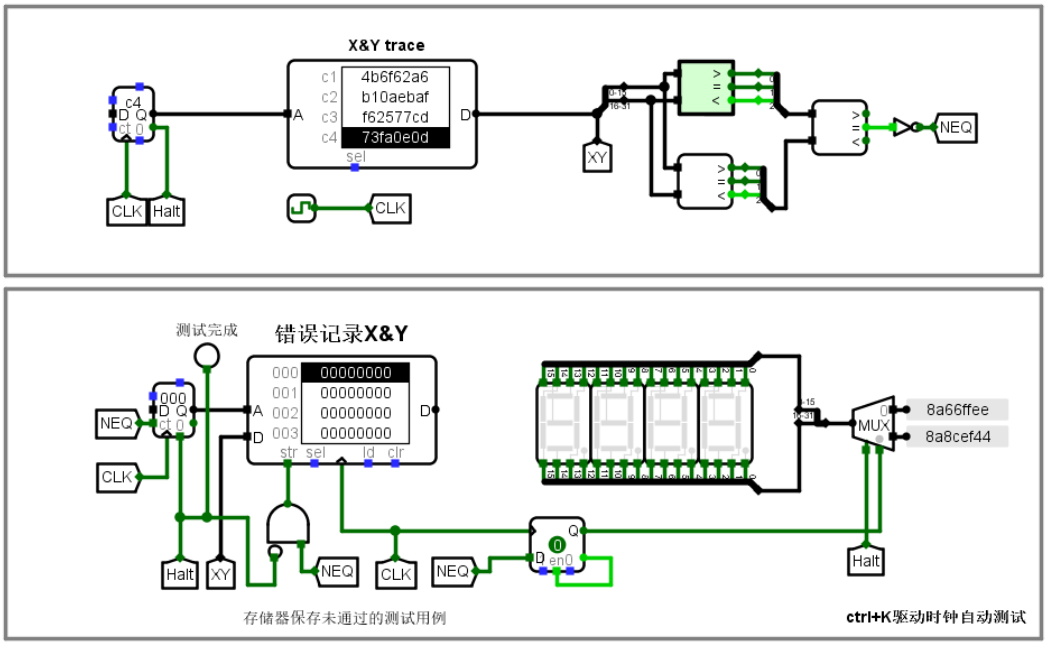


（2）测试图

由于该电路实现较简单，该测试中我们仍然使用 16 位无符号数比较器自动测试，并运行了一段时间，未见异常。



16位数无符号比较器自动测试



16位数无符号比较器自动测试

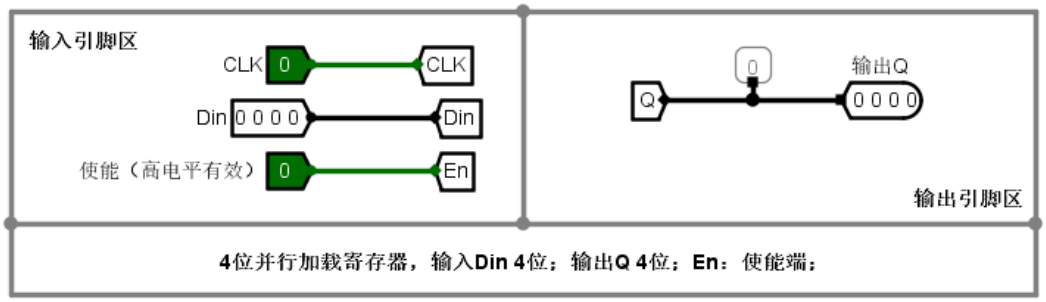
（3）测试分析

如上图，未见异常抛出，故验证了电路的正确性。

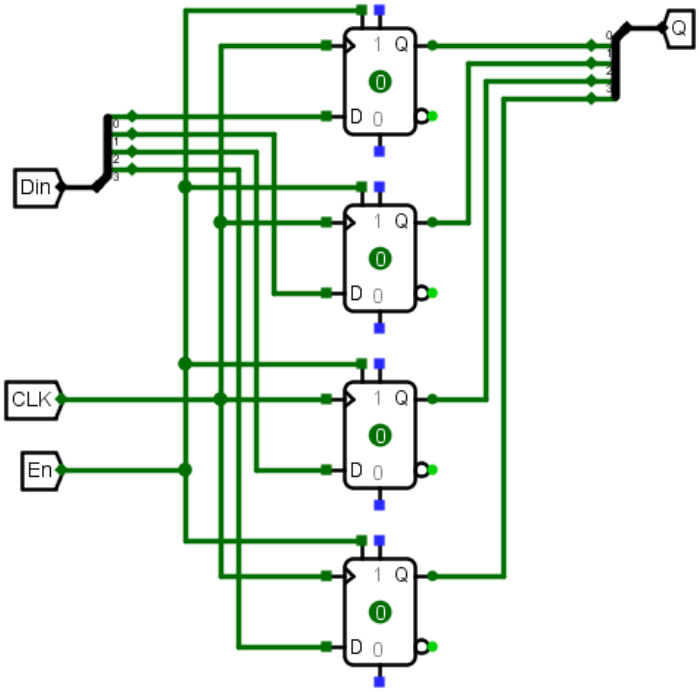
2.2.4 16 位并行加载寄存器电路

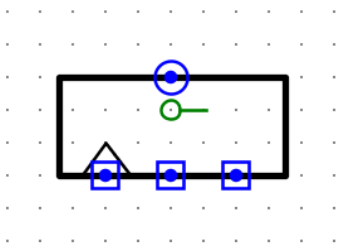
（1）电路图

我们首先构建 4 位并行加载寄存器，它的主体部分是 4 个 D 触发器。4 位并行加载寄存器的电路图和封装图如下。

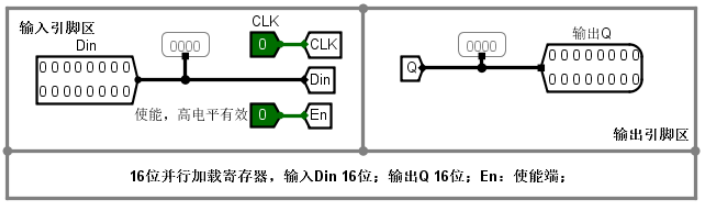


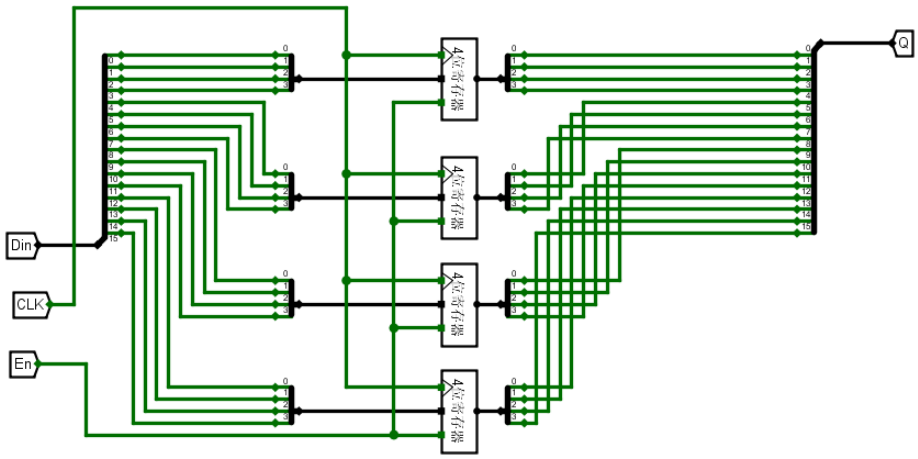
请勿增删改引脚，请在下方利用上方输入输出引脚的隧道标签信号构建电路，ctrl+d复制选择组件

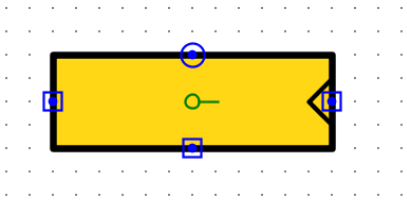




基于如上电路，我们将 4 个 4 位并行加载寄存器连接在一起，加之以一些逻辑结构，以构成 16 位并行加载寄存器。16 位并行加载寄存器的电路图和封装图如下。

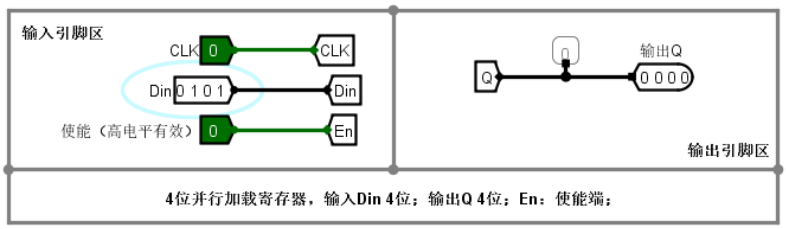




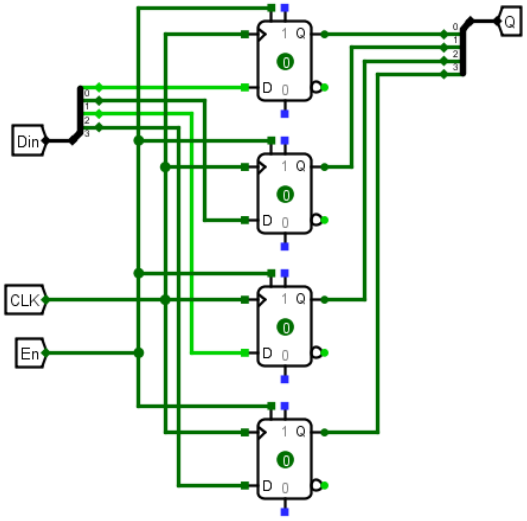


（2）测试图

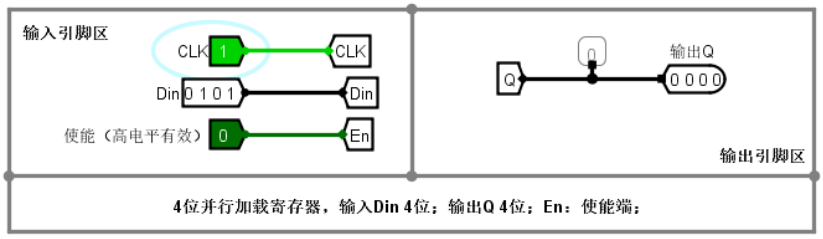
我们首先将 Din 设置为“0101”，未传来时钟信号 CLK 时，输出保持为“0000”。



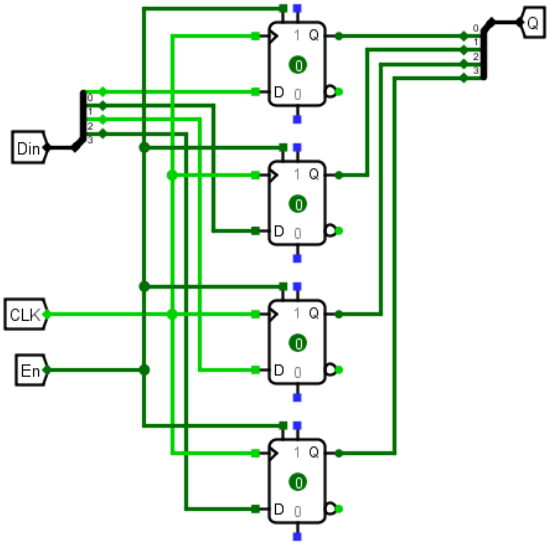
请勿增删改引脚，请在下方利用上方输入输出引脚的隧道标签信号构建电路，ctrl+d复制选择组件



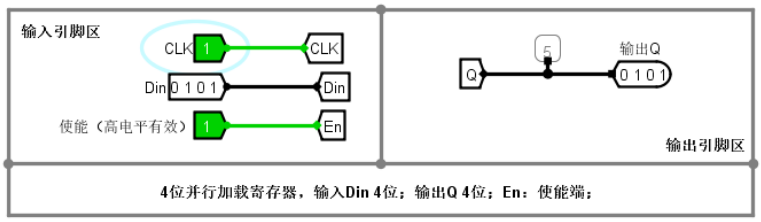
传来了时钟上升沿信号，但使能端 En=0，输出仍然未发生变化。



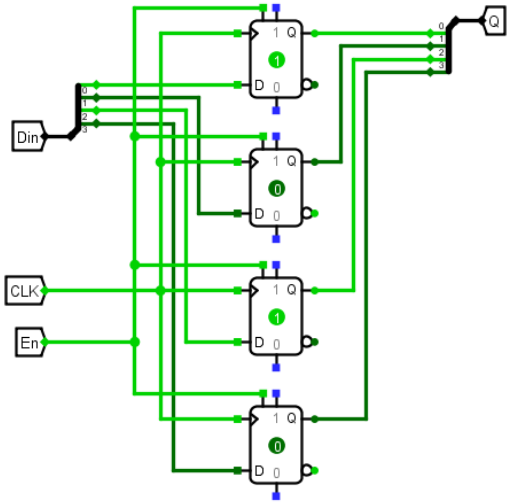
请勿增删改引脚，请在下方利用上方输入输出引脚的隧道标签信号构建电路，ctrl+d复制选择组件



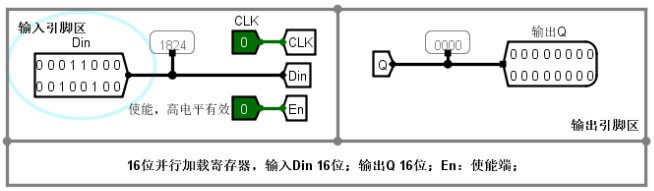
使能端为高电平，且具有时钟信号时，寄存器的值改为“0101”。



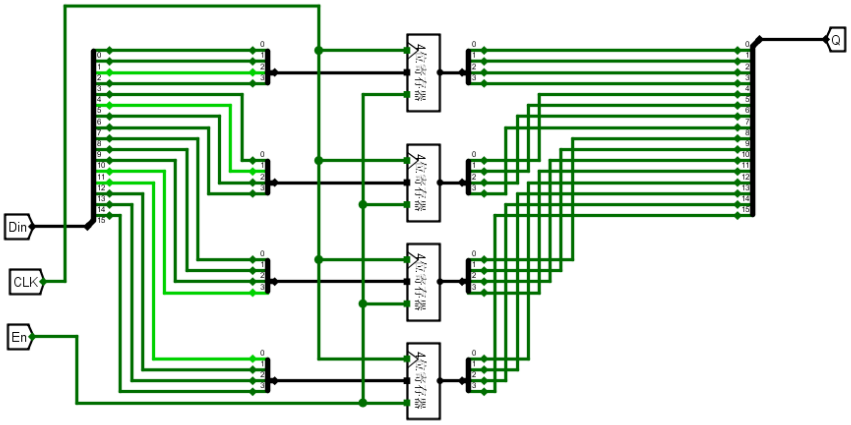
请勿增删改引脚，请在下方利用上方输入输出引脚的隧道标签信号构建电路，ctrl+d复制选择组件

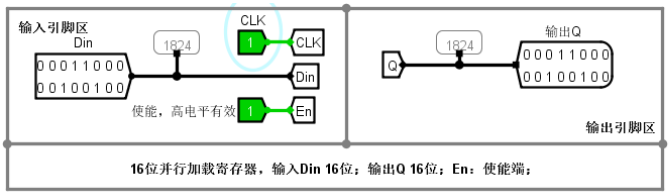


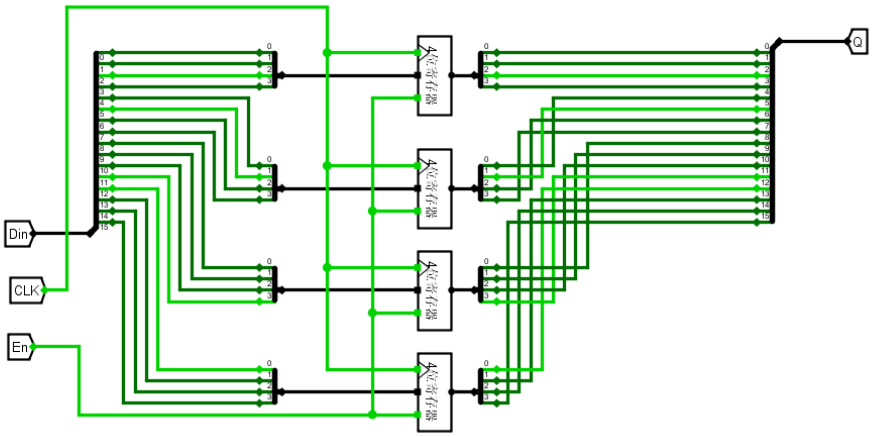
以上为对 4 位并行加载寄存器的测试。下面我们开始对 16 位并行加载寄存器测试。由于实验包中提供了对应的自动测试，我们运用实验包运行了一段时间，未见异常抛出。



请勿增删改引脚，请在下方利用上方输入输出引脚的隧道标签信号构建电路，ctrl+d复制选择组件







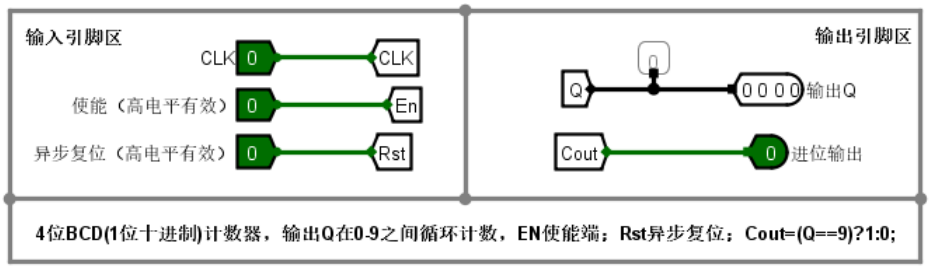
（3）测试分析

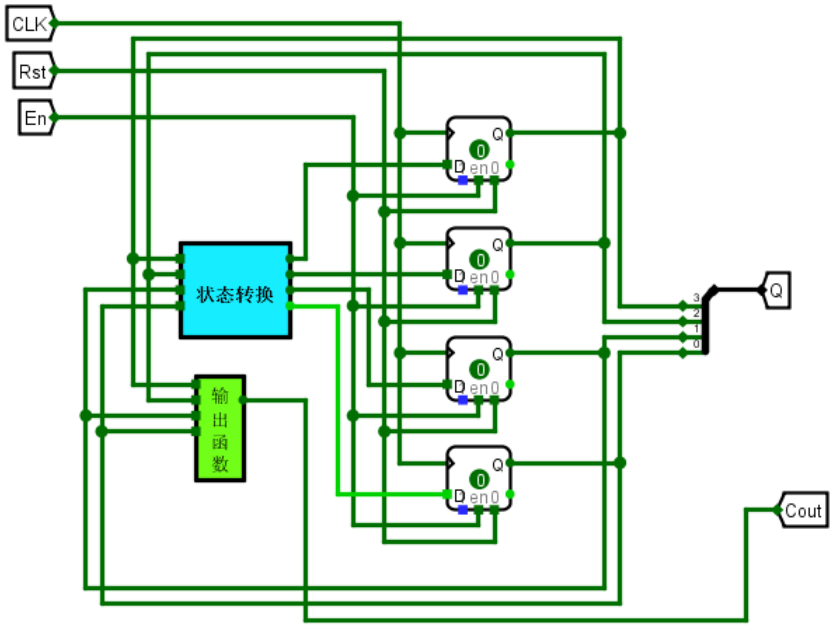
对于 4 位并行加载寄存器，当使能端为高电平，且时钟端为上升沿时，寄存器存储了相应的值，这表明了电路的正确性。而且自动测试未抛出异常，进一步确认了我们设计的电路是正确的。

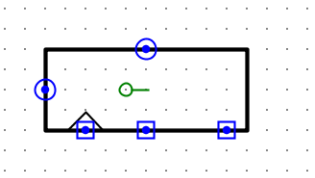
2.2.5 BCD 计数器及其码表计数器电路

（1）电路图

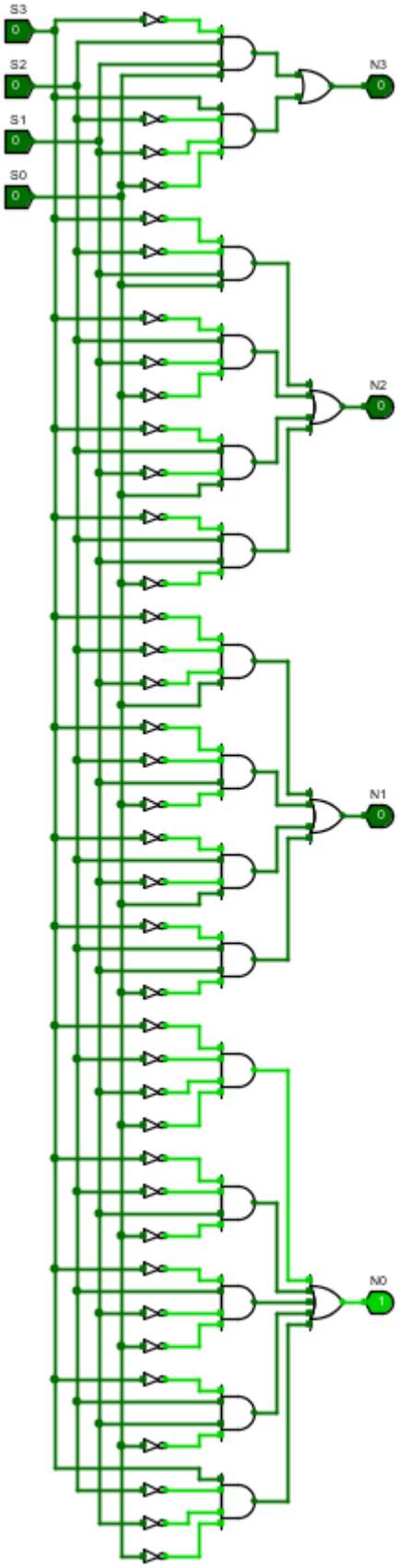
我们首先构建 4 位 BCD 计数器。其电路图和封装图如下。

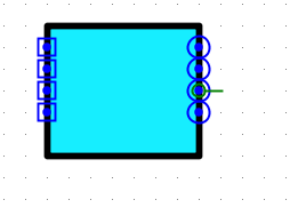




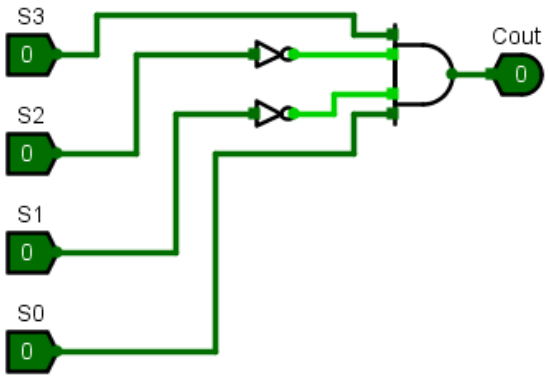


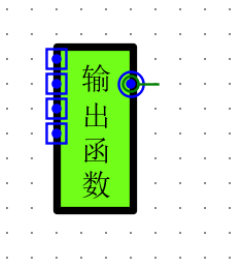
它的状态转换电路如下图。


S3
0
S2
0
S1
0
S0
0
N3
0
N2
0
N1
0
N0
1

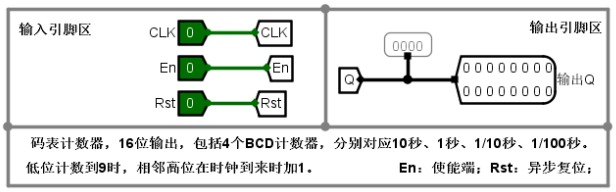


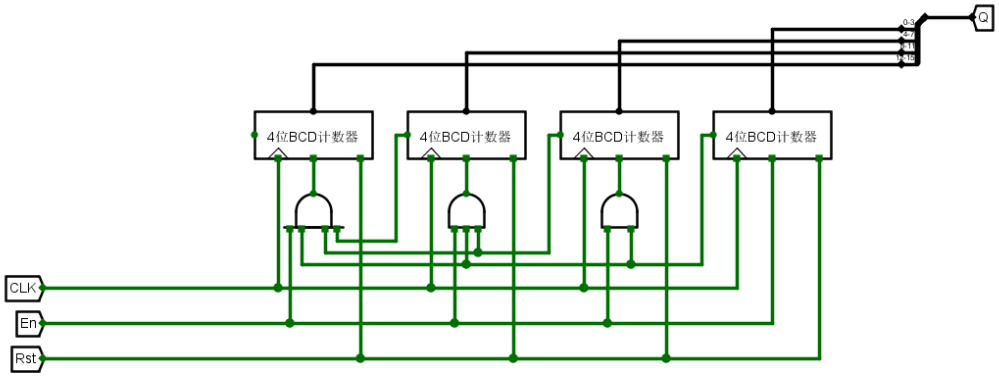
其输出函数电路如下图。

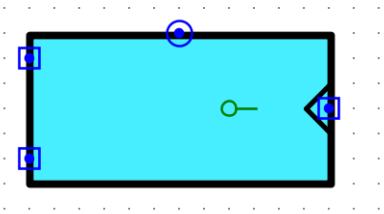




其次是码表计数器，即 16 位输出的 4 个 BCD 计数器。

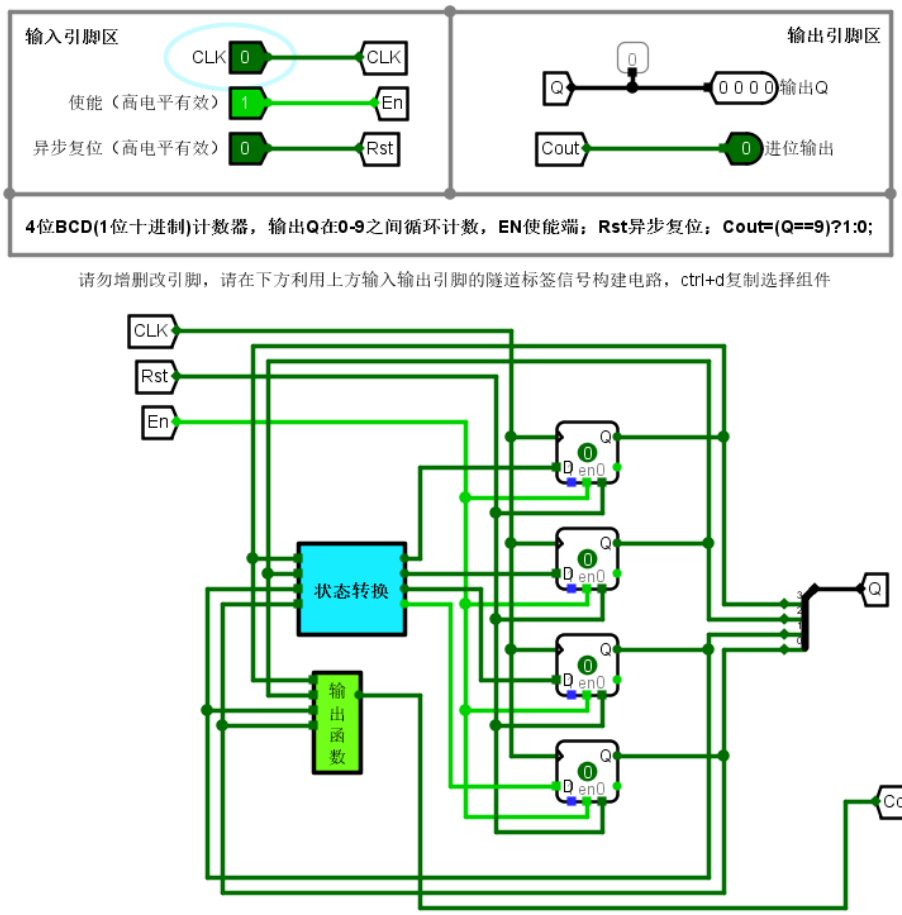




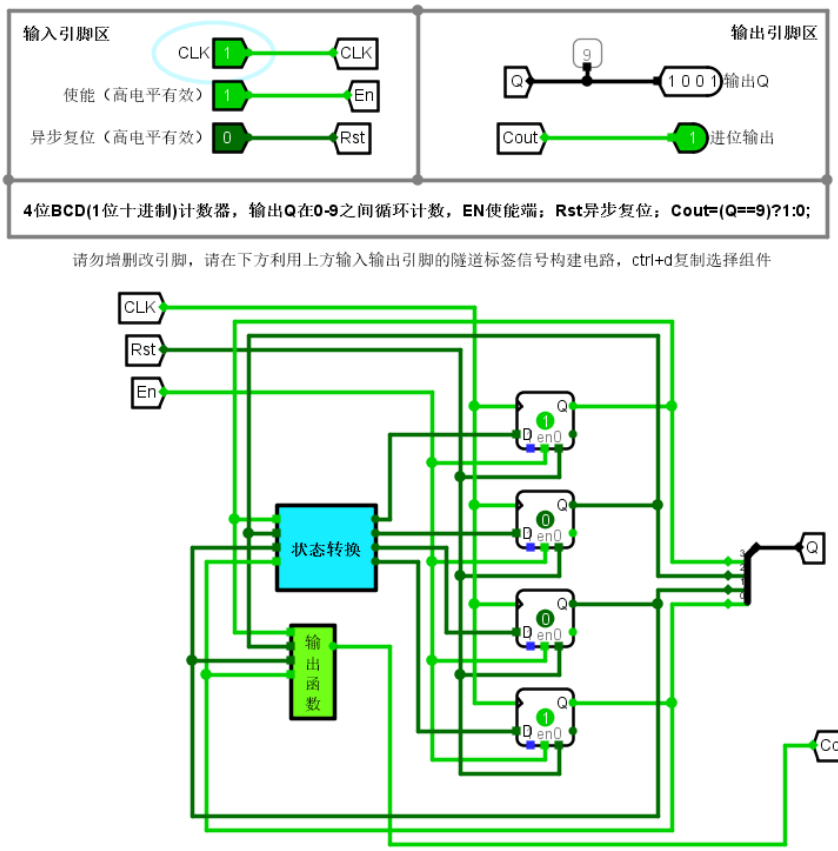


（2）测试图

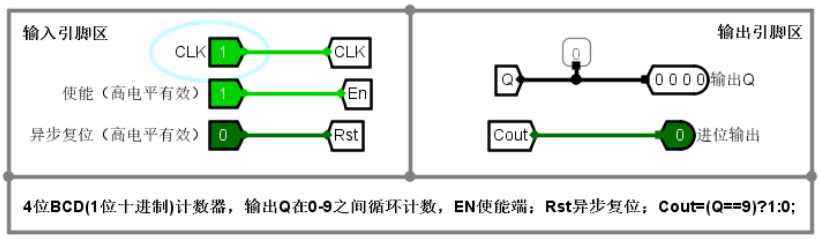
初始状态为“0”状态，我们持续给 BCD 计数器施加 CLK 信号。



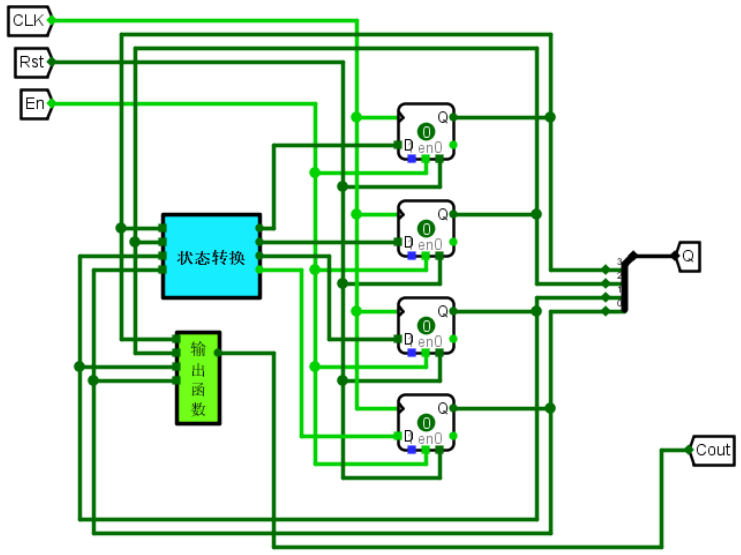
当次态为 9 时，我们输出 Q 为“1001”，且进位输出为 1。



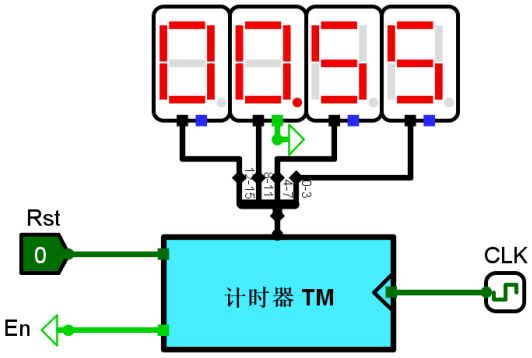
状态为 9 之后，再一次施加时钟信号，使状态更改为 0 状态，输出更改为 0。



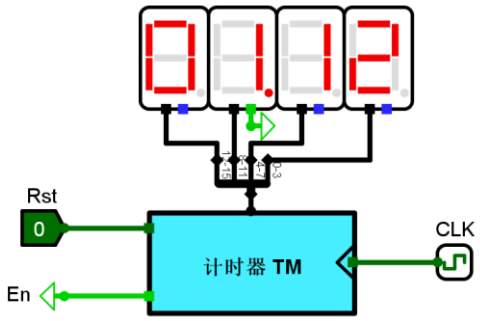
请勿增删改引脚，请在下方利用上方输入输出引脚的隧道标签信号构建电路，ctrl+d复制选择组件



下面我们对集成的码表计数器进行自动测试。如下两图，验证了码表的正常运作。



Ctrl+K驱动时钟自动测试



Ctrl+K驱动时钟自动测试

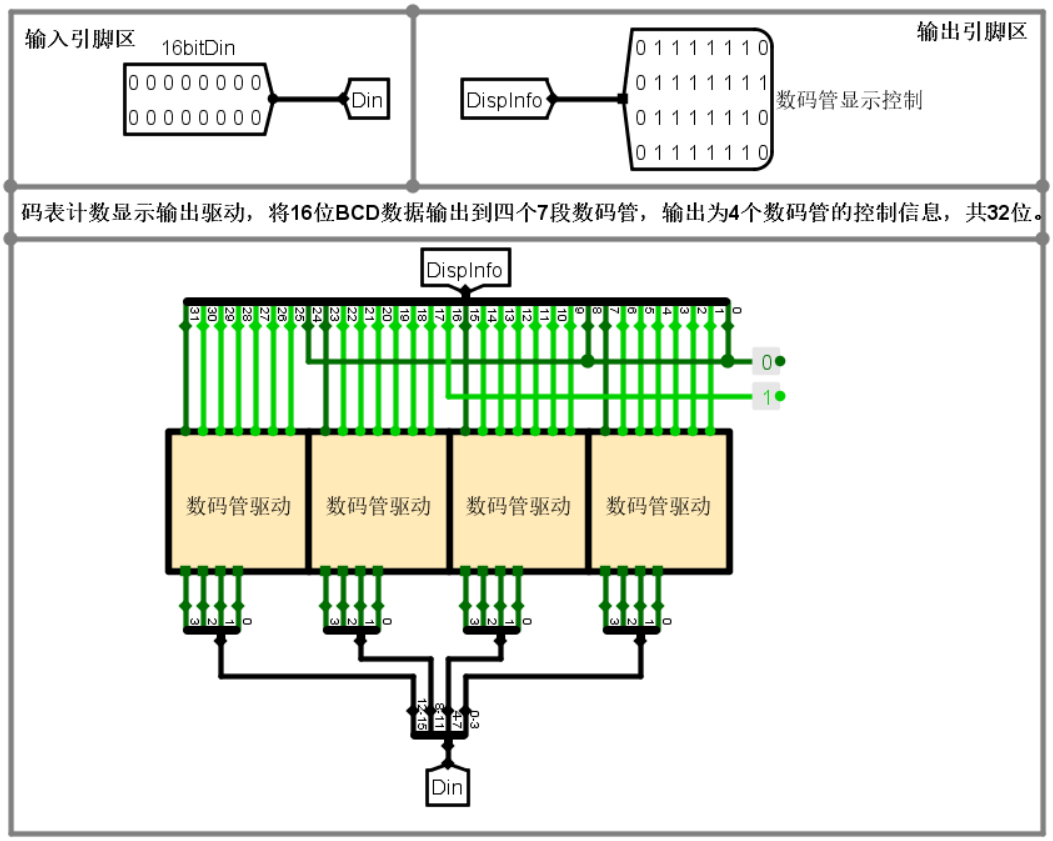
（3）测试分析

BCD 计数器的状态转换正常，可以从状态 9 跳入状态 0，且进位输出亦正常。而自动测试也佐证了我们电路设计的正确性。

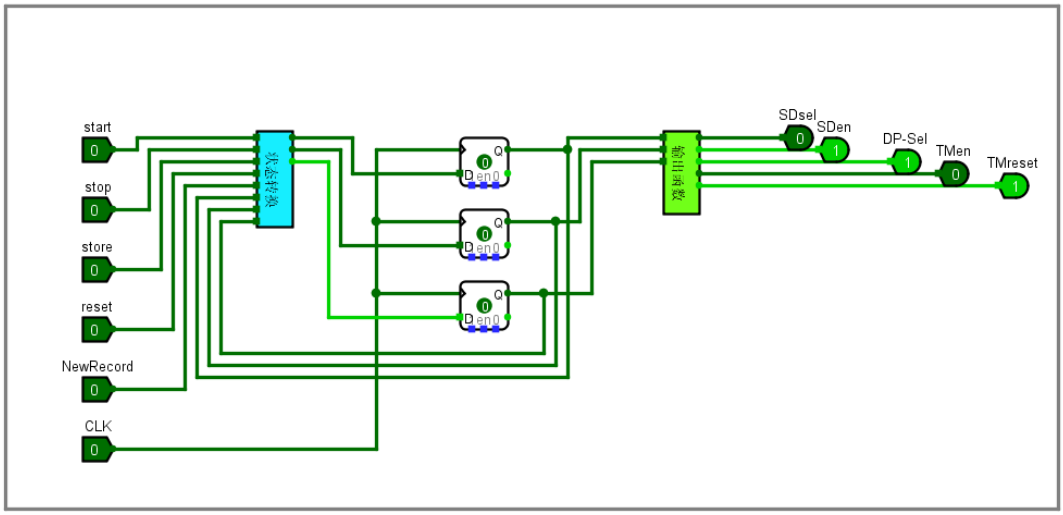
2.2.6 码表控制器电路

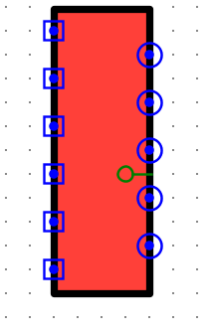
（1）电路图

首先我们给出码表计数显示驱动电路。

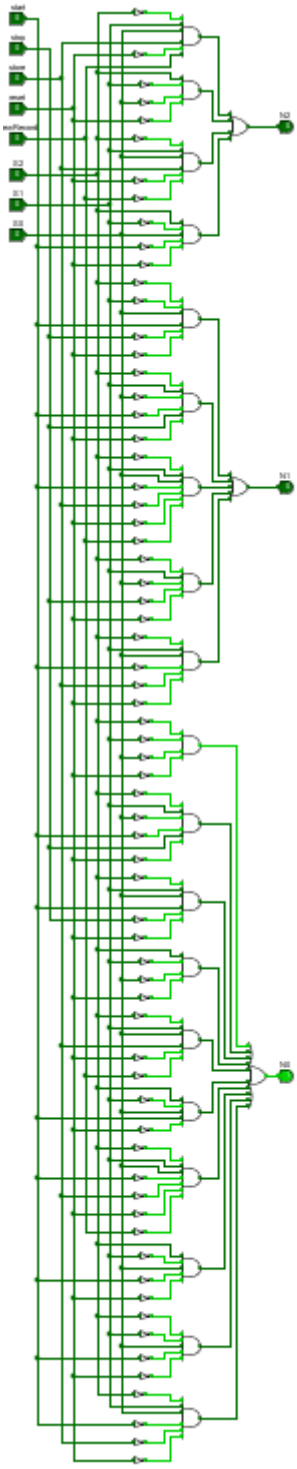


其次是码表控制器的总电路及其封装。

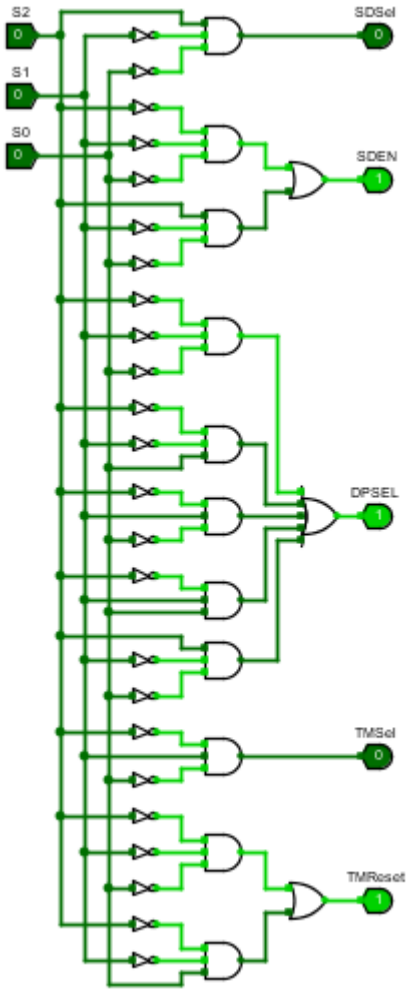




如下是码表控制器的状态转移电路。

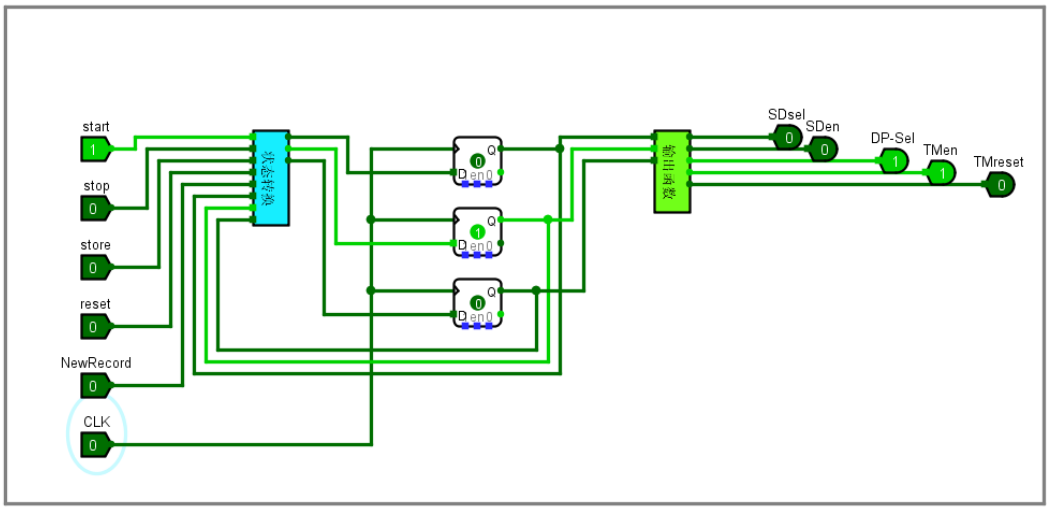


如下是码表控制器的输出函数电路。



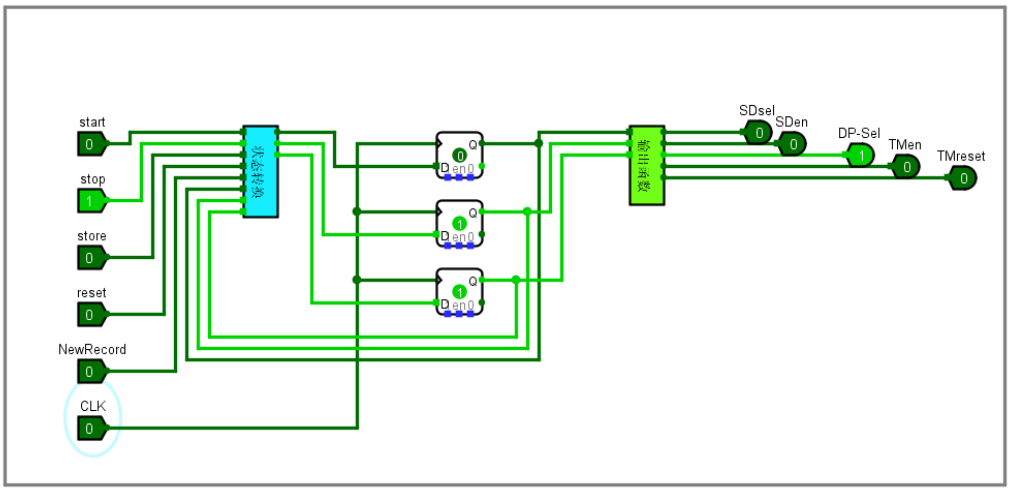
（2）测试图

首先我们传入 start 信号。可见状态转换为“010”。

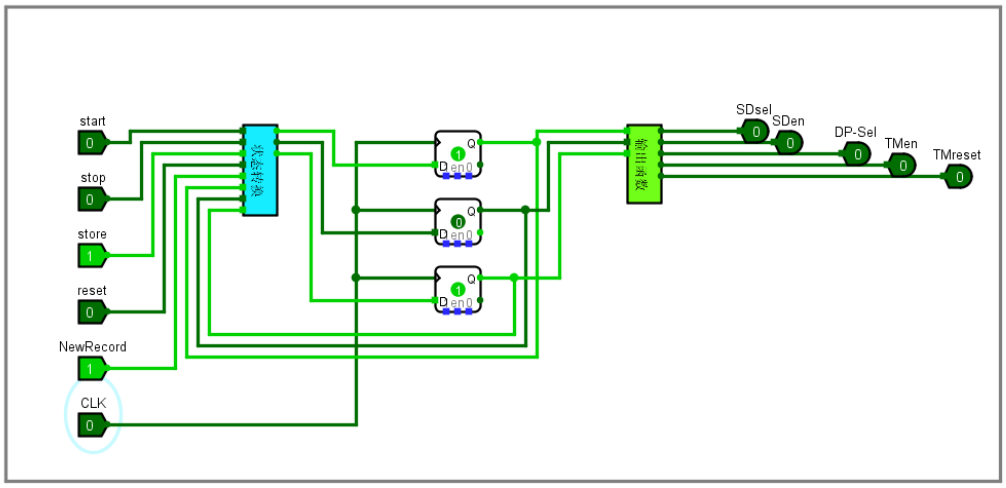


其次我们让码表停止。输入“stop”信号，码表状态更改为“011”，且输出函数为

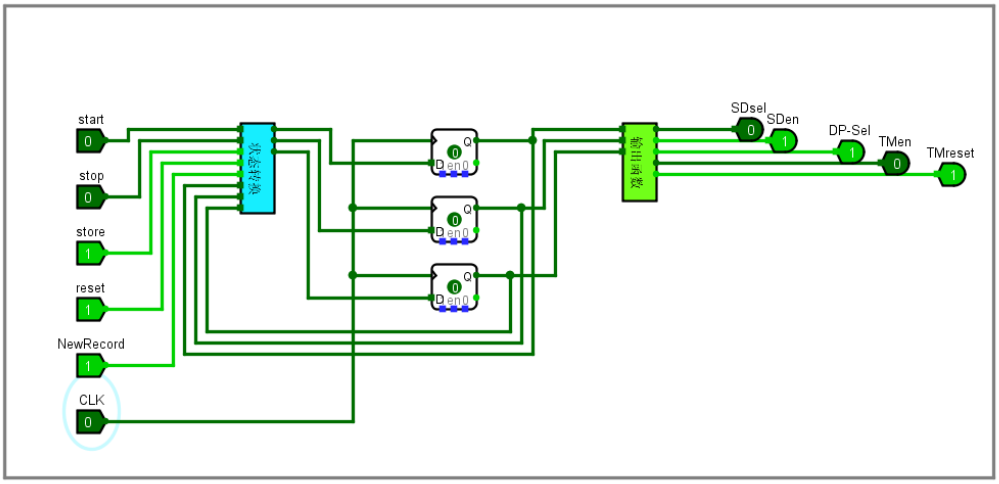
DP-Sel。



再次我们按下 store 键，即具有 store 和 NewRecord 输入。此时，对于的状态更改为“101”，由于我们这是连续传入了两次 CLK 信号，致使输出函数均为零。



由于 reset 的优先级最高，只要按下“reset”键，并且正处于时钟上升沿，码表即处于复位状态。



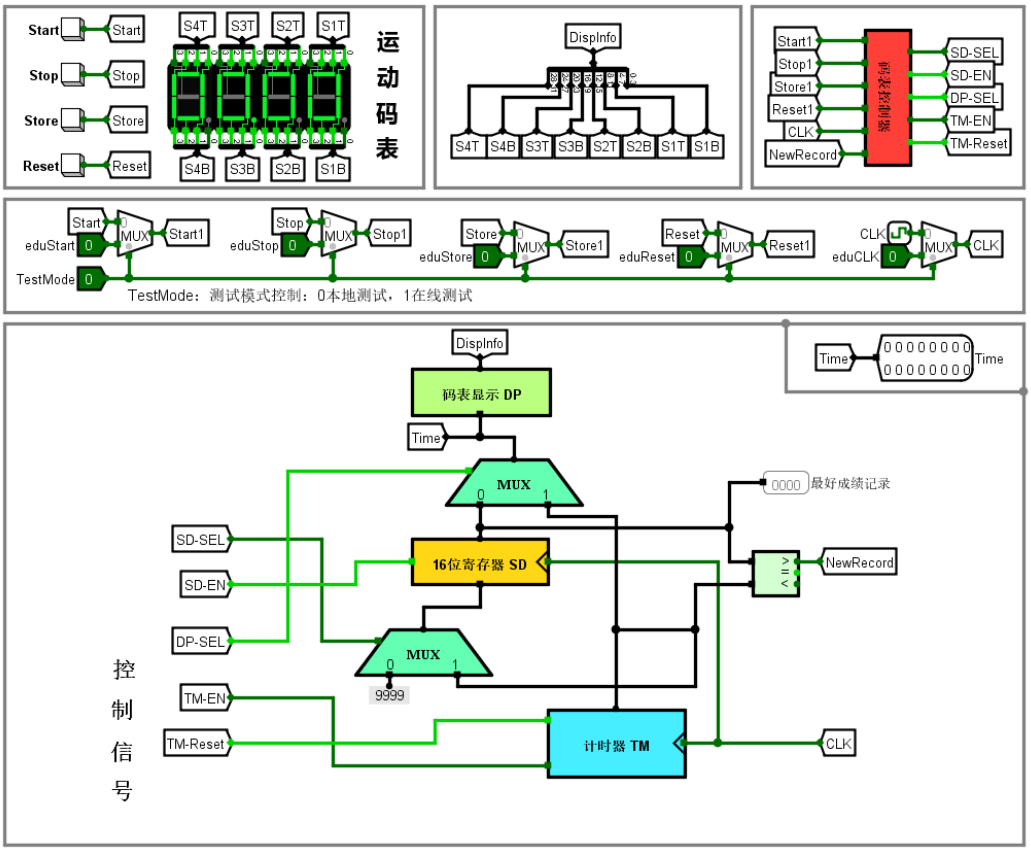
（3）测试分析

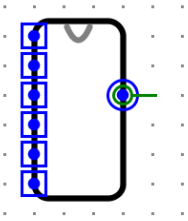
根据我们对码表状态的编码，和我们的测试，都证明了码表状态转移和输出的正确性。

2.2.10 运动码表

（1）电路图

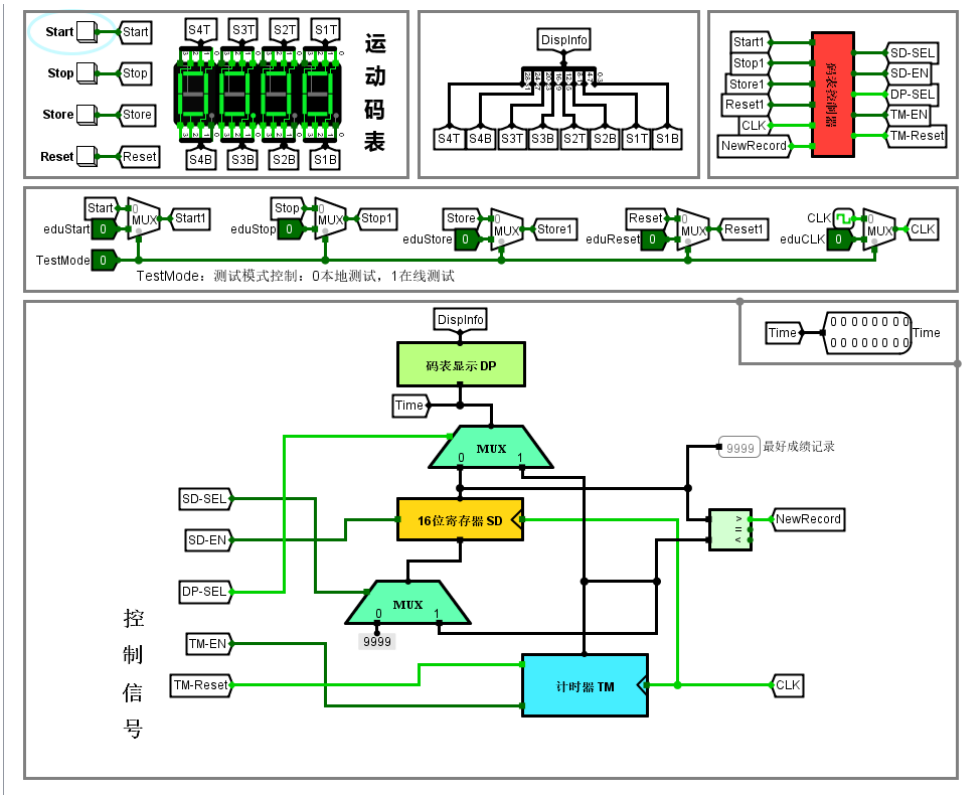
下为运动码表的总电路及其封装。



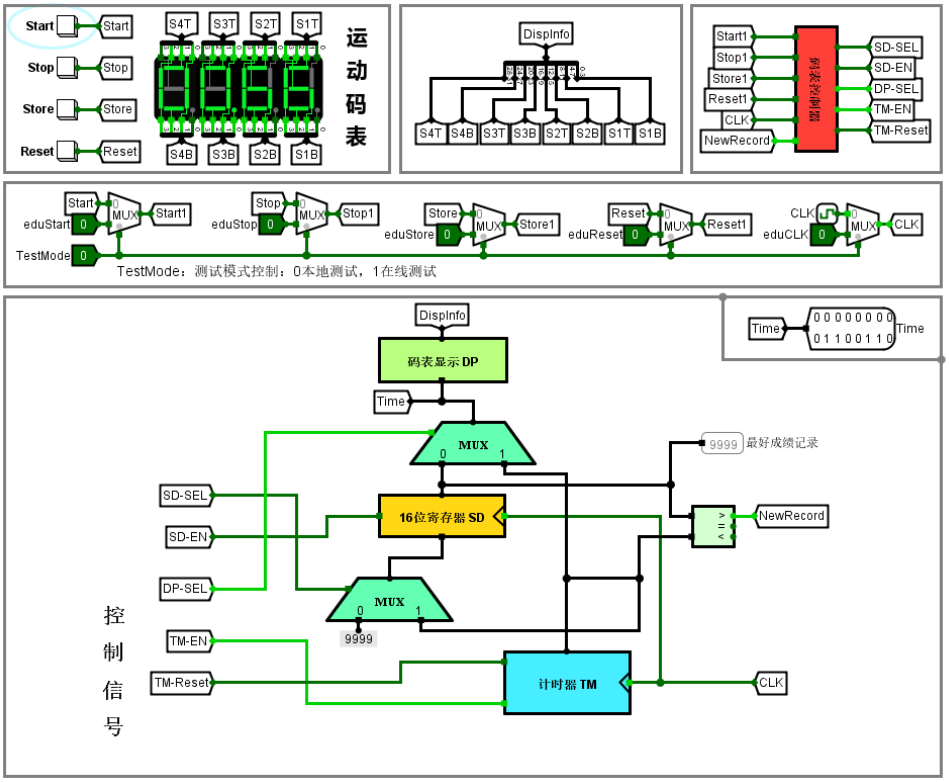


（2）测试图

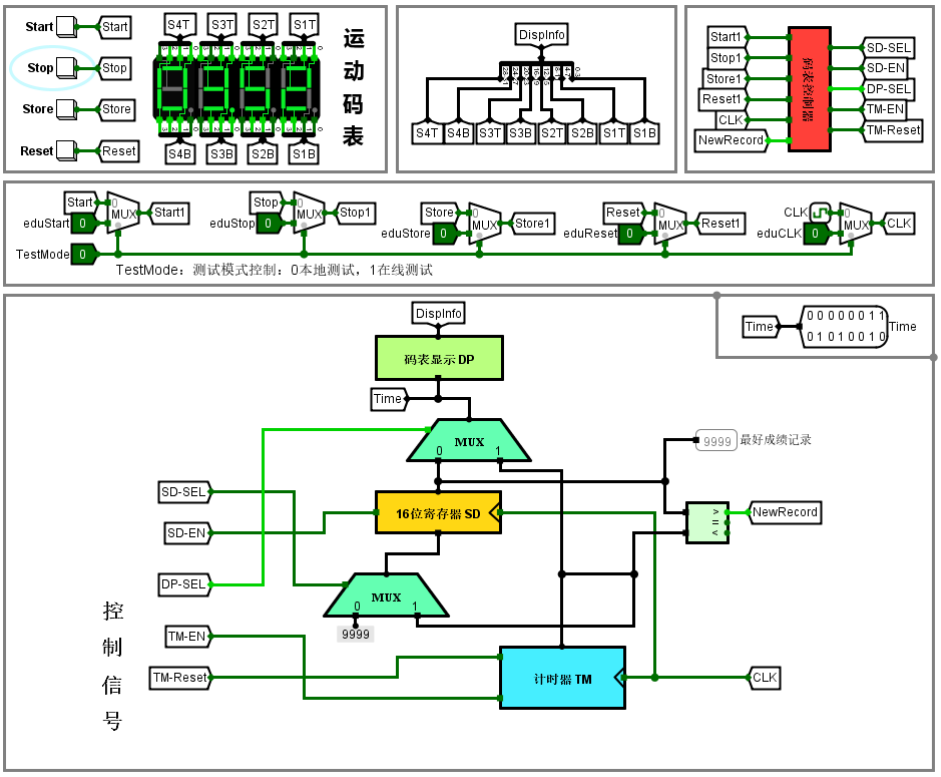
下图为运动码表的初始状态。



我们对码表进行计时操作。如下图，计时了 0.66 秒，寄存器存储为最大数 99.99，因为还未有新的数据储存入寄存器。

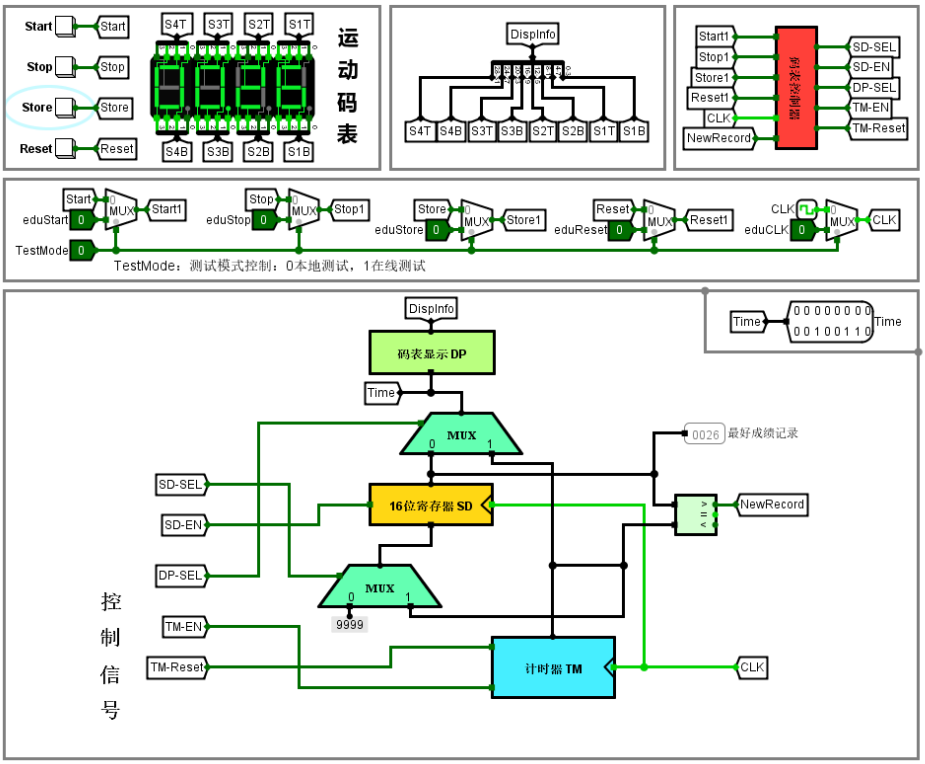


如下图，我们在计数器为 3.52 秒时让计时器停止，可见码表正常停止。



我们再一次对让码表进行计时。这一次当码表为 0.26 秒时，我们让码表停止，且

按下 store 键。可见屏幕上显示了 0.26 秒，且寄存器可存储了相应的值。



（3）测试分析

根据前文所述，码表工作正常。

3　设计总结与心得

3.1 实验总结

我们对实验过程进行总结。

3.1.1 遇到的问题及处理

在实验具体设计过程中，我也遇到了一些困难。我印象较深刻的问题是码表控制器的状态转换电路设计中，我由于考虑不周全，数次使得电路输出错误的答案。后来我反复调试了 reset 相关的状态转移真值表，使得该模块终于输出了正确的答案。

另一个遇到的问题在于码表计数器中，我以为使能端可以是常数高电平，未将使能端与四个 BCD 计数器依次连接，导致了该电路的设计不能得到正确答案。后来我经过反复的观察发现了这一个漏洞，并解决了问题。

3.1.2 设计方案存在的不足

我们所设计的码表功能不够齐全。在真实的码表应用场景中，需要记录多组数据以供教练使用，且我们未设计“暂停”键，只设计了“停止”键。实际码表经常需要暂停计时，而后恢复计时，而我们并未对该功能进行支持。

3.2 实验心得

数字电路是一门需要动手，需要“make hands dirty”的学科。在实验具体设计过程中，我对寄存器、BCD 计数器等一系列基础元件有了更加深刻了理解，同时也对 Logisim 这一精简的软件深感其强大。

通过这门实验课，我基本学习了 Logisim 的软件使用方法，并且学习了数字电路的设计原则和基本方法。我们将根据数字电路的学习，为将来计算机组成原理的学习而继续加油！

3.3 意见与建议

这是一门有趣的实验。为了使实验更有意思，我建议可以加大实验的自由度，可以不止是按部就班地制作运动码表，而可以为该码表加入一些自定义元素，比如“暂停”键选项。

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。

已阅读并同意以下内容。

判定为不合格的一些情形：

（1） 请人代做或冒名顶替者；

（2） 替人做且不听劝告者；

（3） 实验报告内容抄袭或雷同者；

（4） 实验报告内容与实际实验内容不一致者；

（5） 实验电路抄袭者。

作者签名：王彬

最终提交的文件

（1）实验电路[电子版]；

（2）实验报告[电子版]；

（3）实验报告[纸质版]。

提交的电子版文件无需压缩，每个学生放在一个文件夹，文件夹及文件命名方式：班级-学号-姓名。如：计算机 2101-U21010101-张三-运动码表实验报告

全班收齐后统一打包压缩交给指导教师。