《数字逻辑》

（第8次实验：在Logisim和FPGA开发板上实现综合应用举例电路）

厦门大学信息学院软件工程系　曾文华

2024年12月16日

目录

- 第一部分：在Logisim上实现综合应用举例电路
- 第二部分：在FPGA开发板上实现综合应用举例电路

第一部分：在Logisim上实现综合应用举例电路

请打开设计文件“第9章综合应用举例电路的实现.circ”

1、在Logisim上实现简单运算器电路

- 简单运算器电路如图9.4所示：

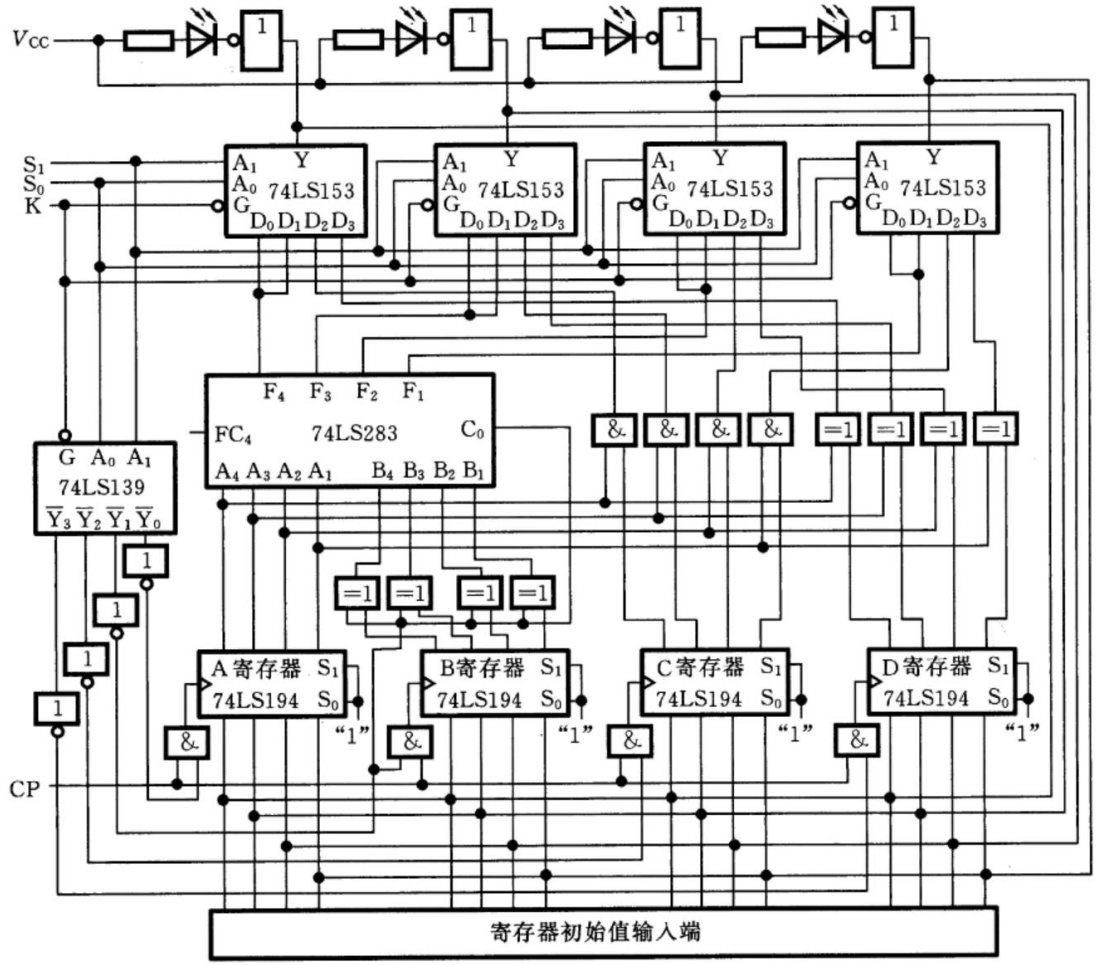
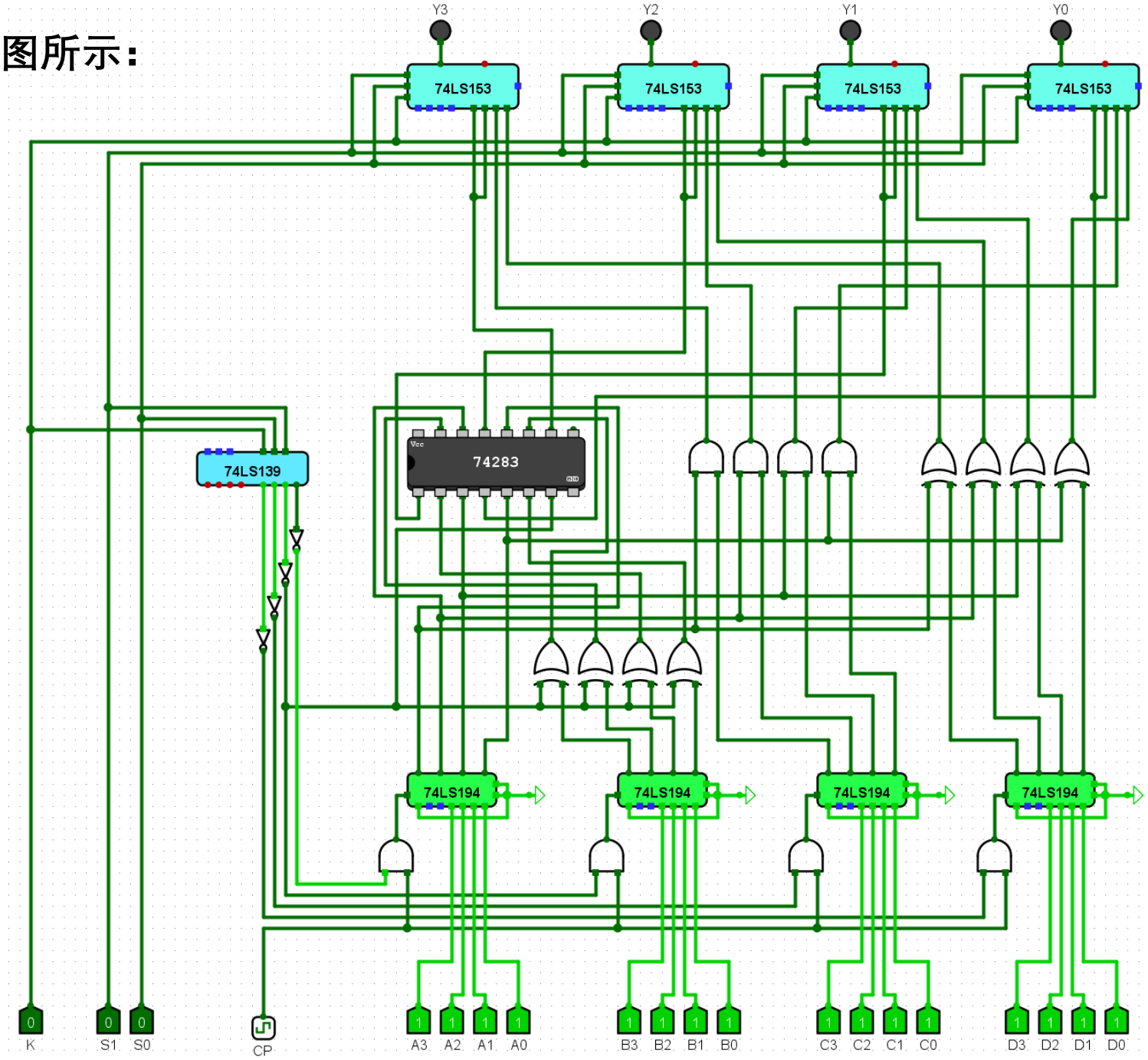
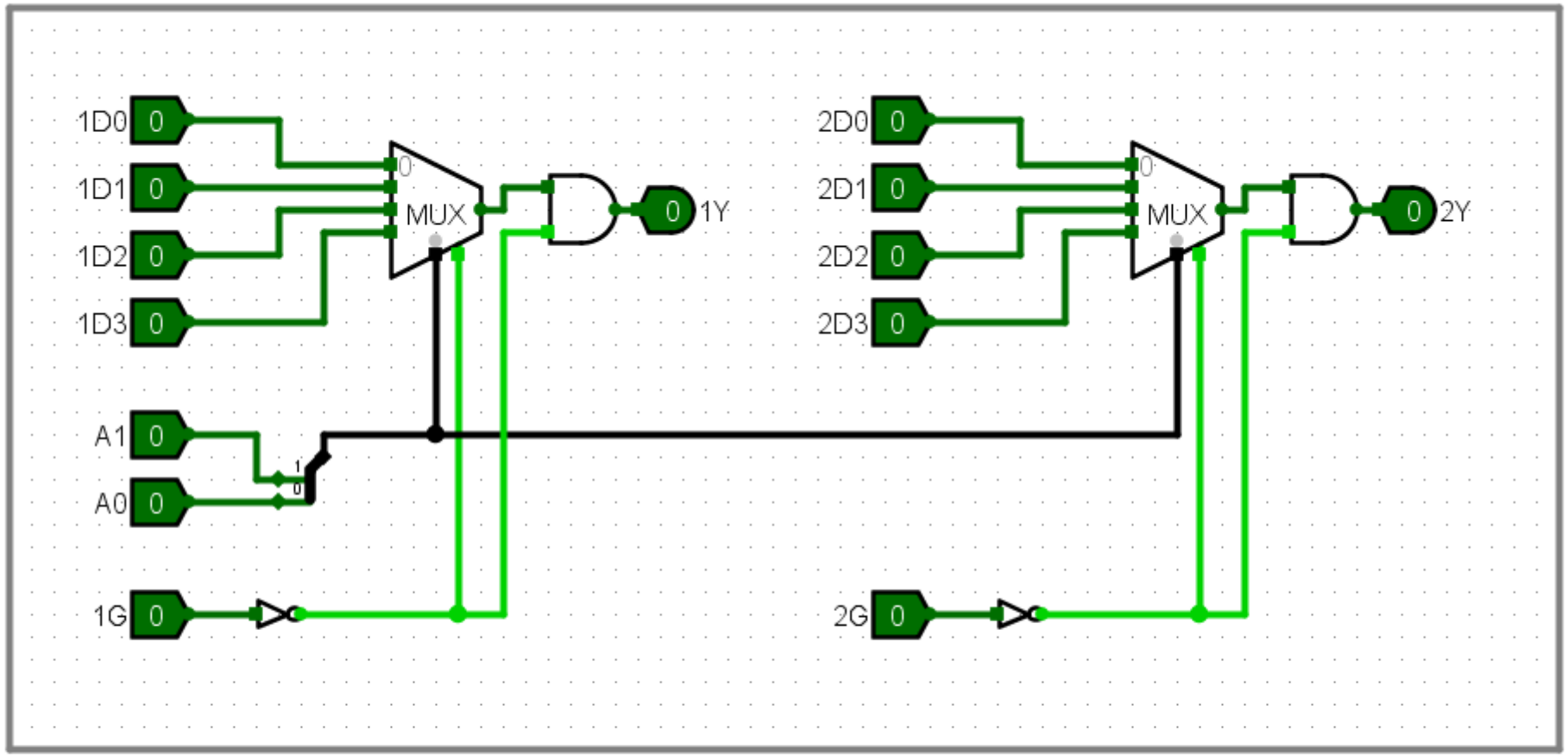


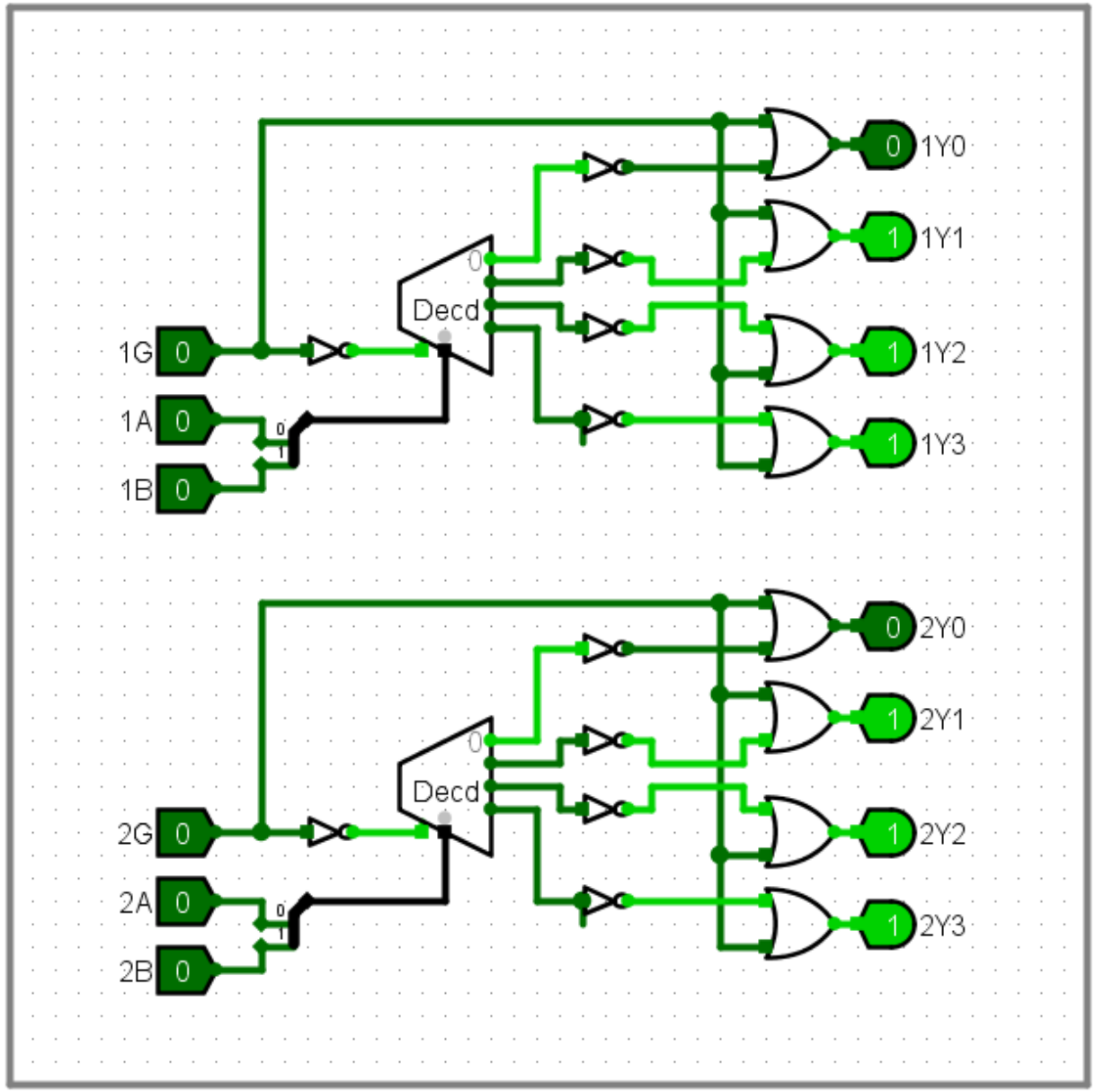
图 9.4　运算电路的完整逻辑电路

- Logisim上实现的简单运算器电路如下图所示：

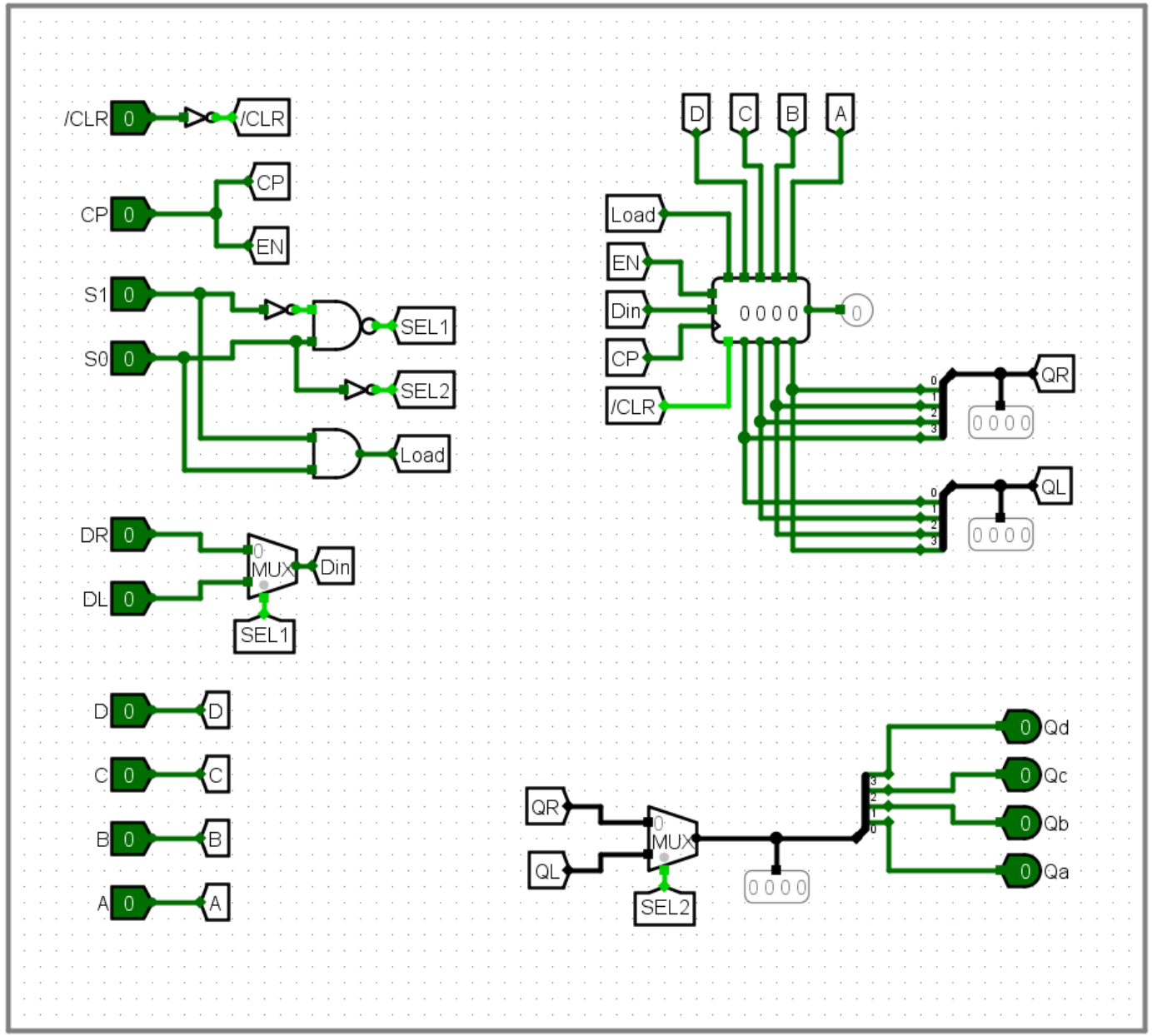




74LS153芯片（双4路多路选择器）



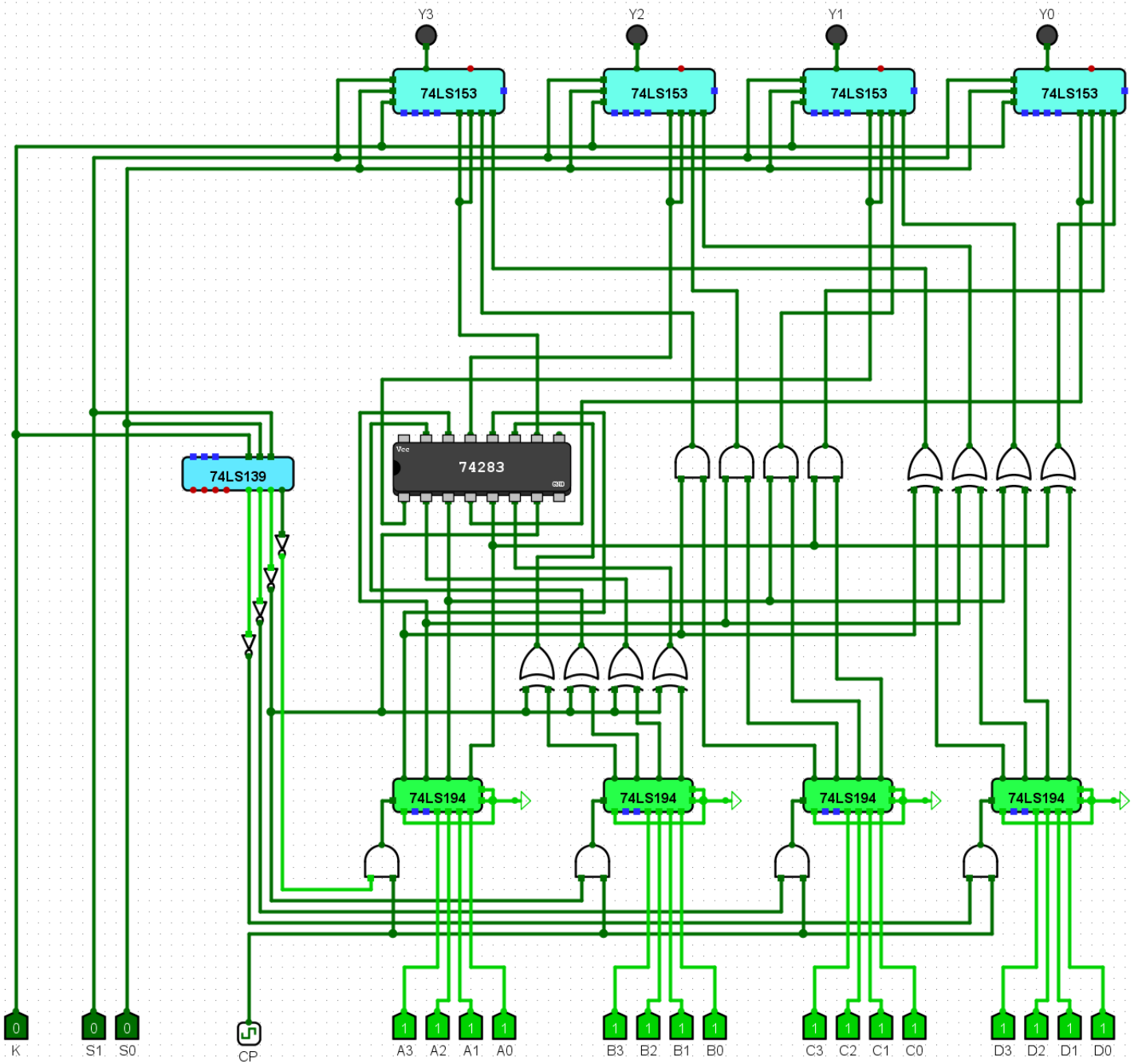
74LS139芯片（双2-4译码器）



74LS194芯片（4位双向移位寄存器）

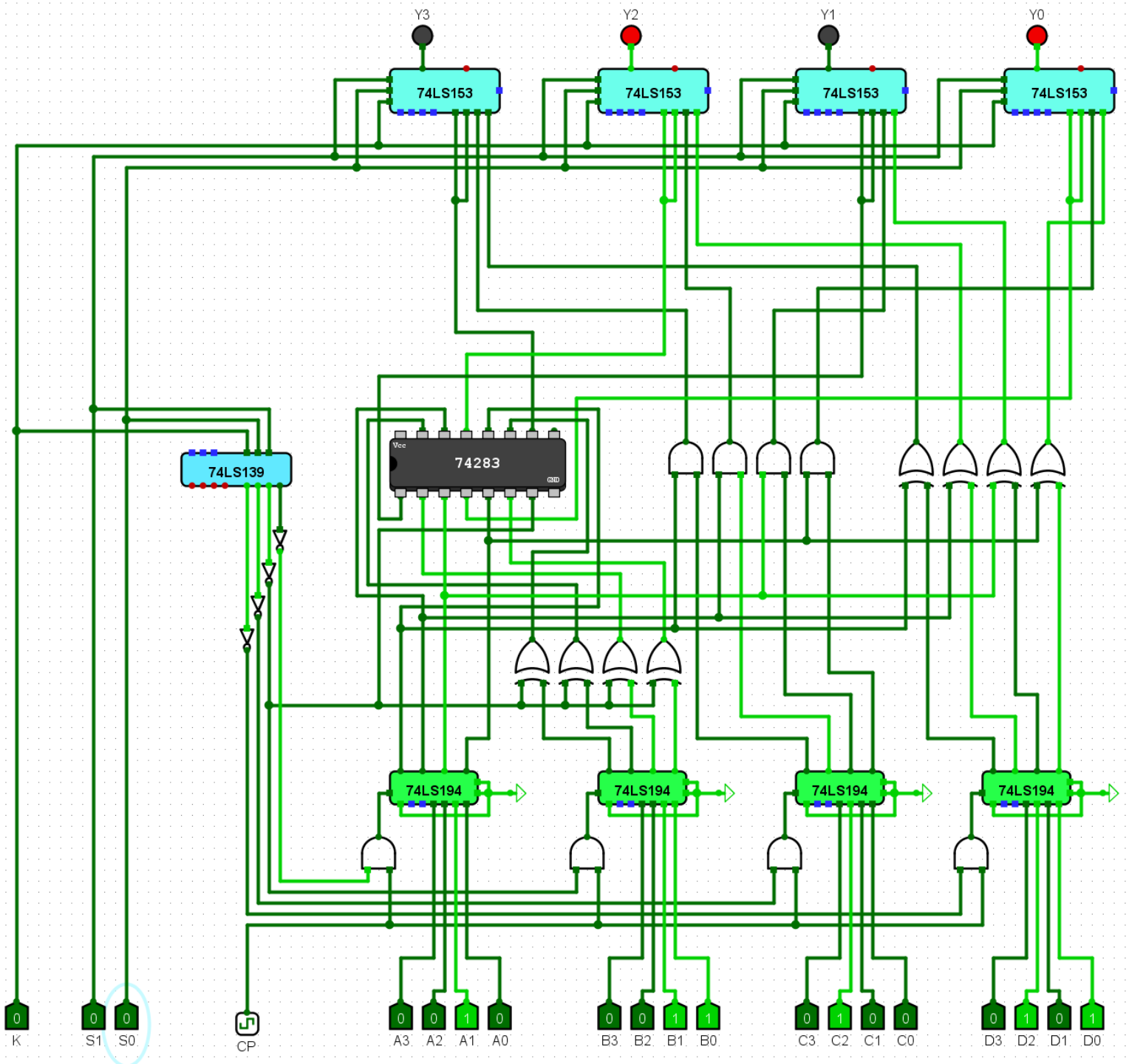
- 第1步：按Ctrl+R
- 第2步：置S1S0=00、A3A2A1A0=0010，按CP
- 第3步：置S1S0=01、B3B2B1B0=0011，按CP
- 第4步：置S1S0=10、C3C2C1C0=0100，按CP
- 第5步：置S1S0=11、D3D2D1D0=0101，按CP

置数



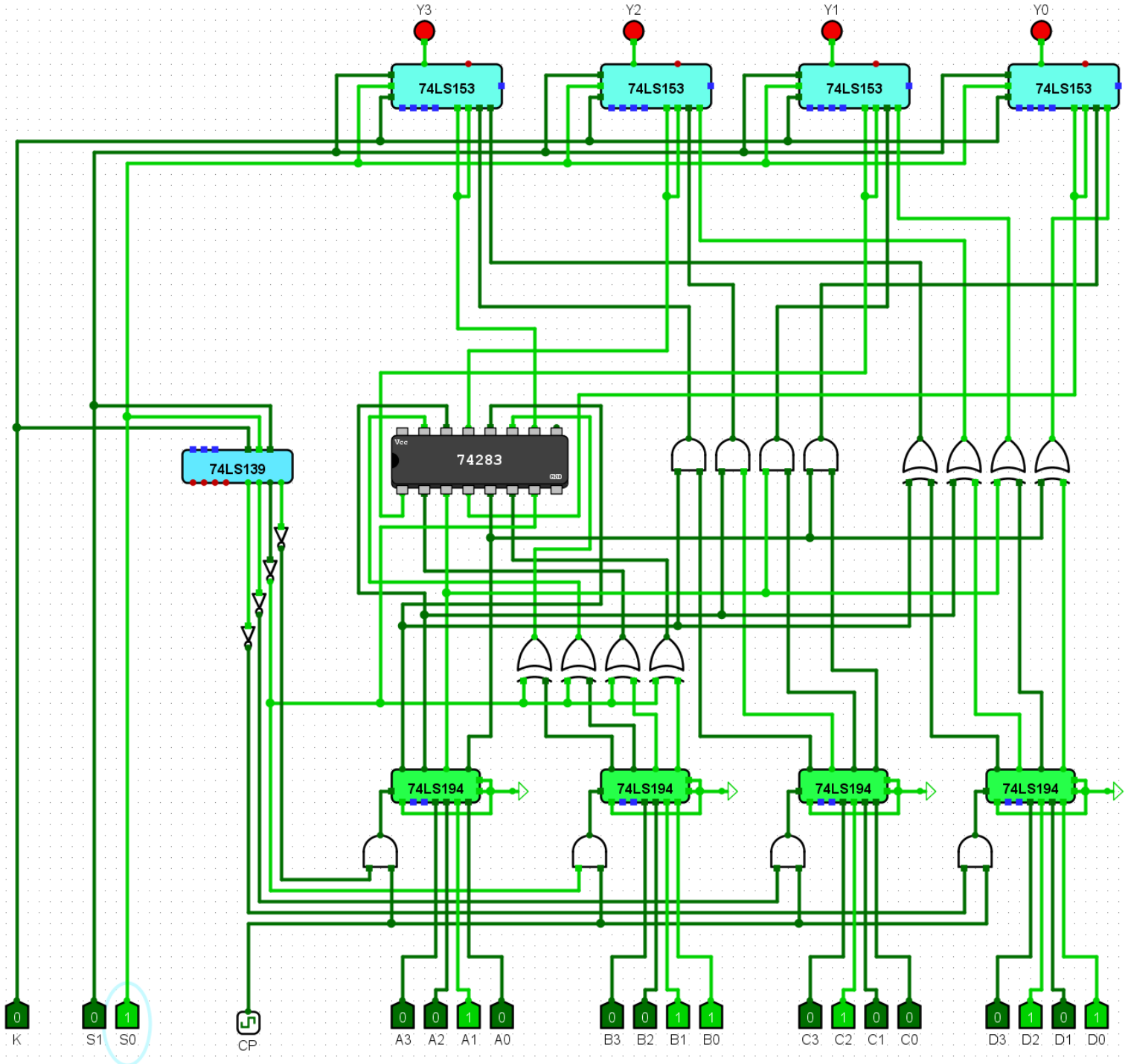
- 第6步：置S1S0=00，观看运算结果是不是为：
  A+B=2+3=5=0101=Y3Y2Y1Y0

加法运算：Y=A+B



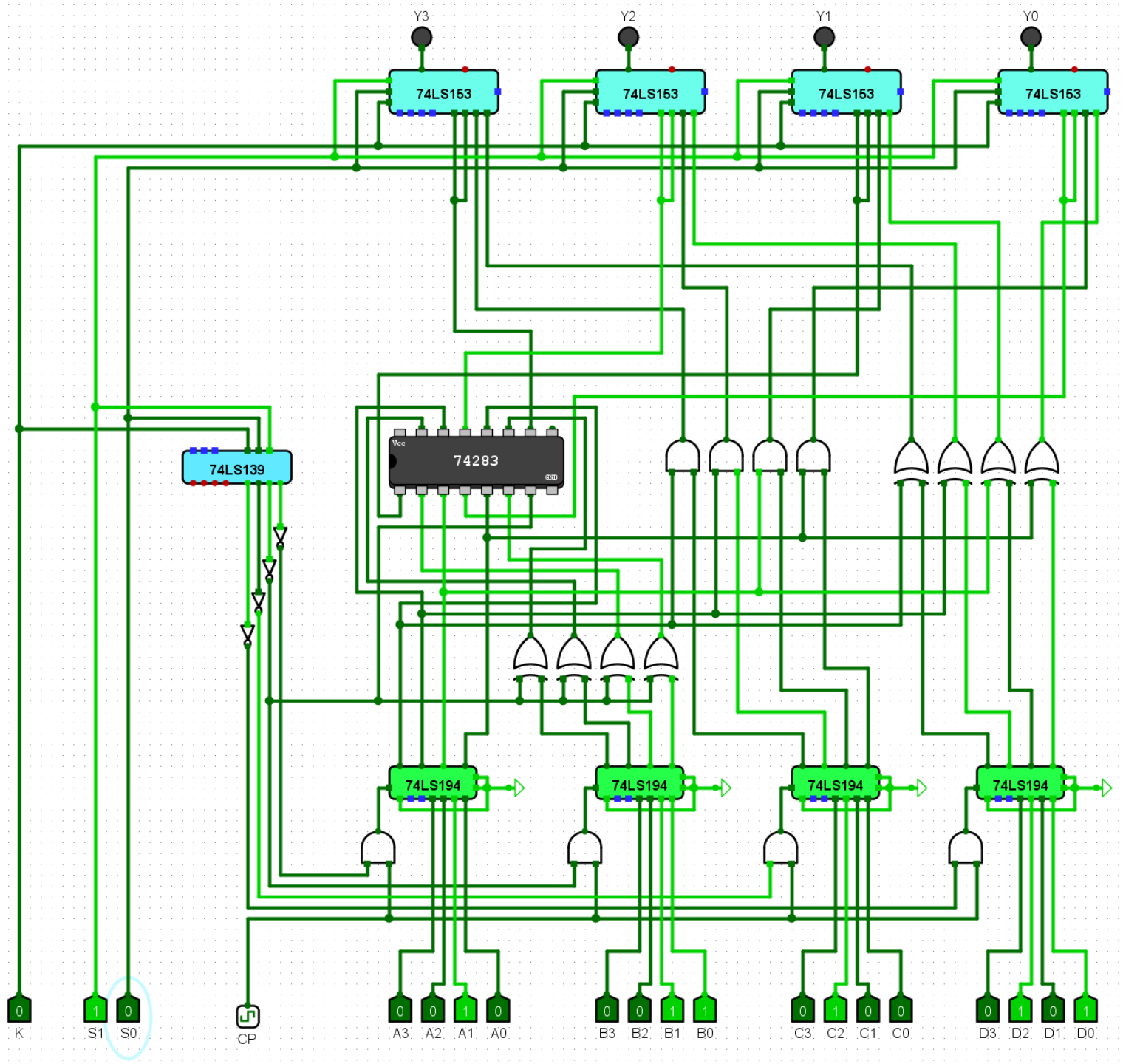
- 第7步：置S1S0=01，观看运算结果是不是为：A-B=2-3=-1=1111=Y3Y2Y1Y0

减法运算：Y=A-B



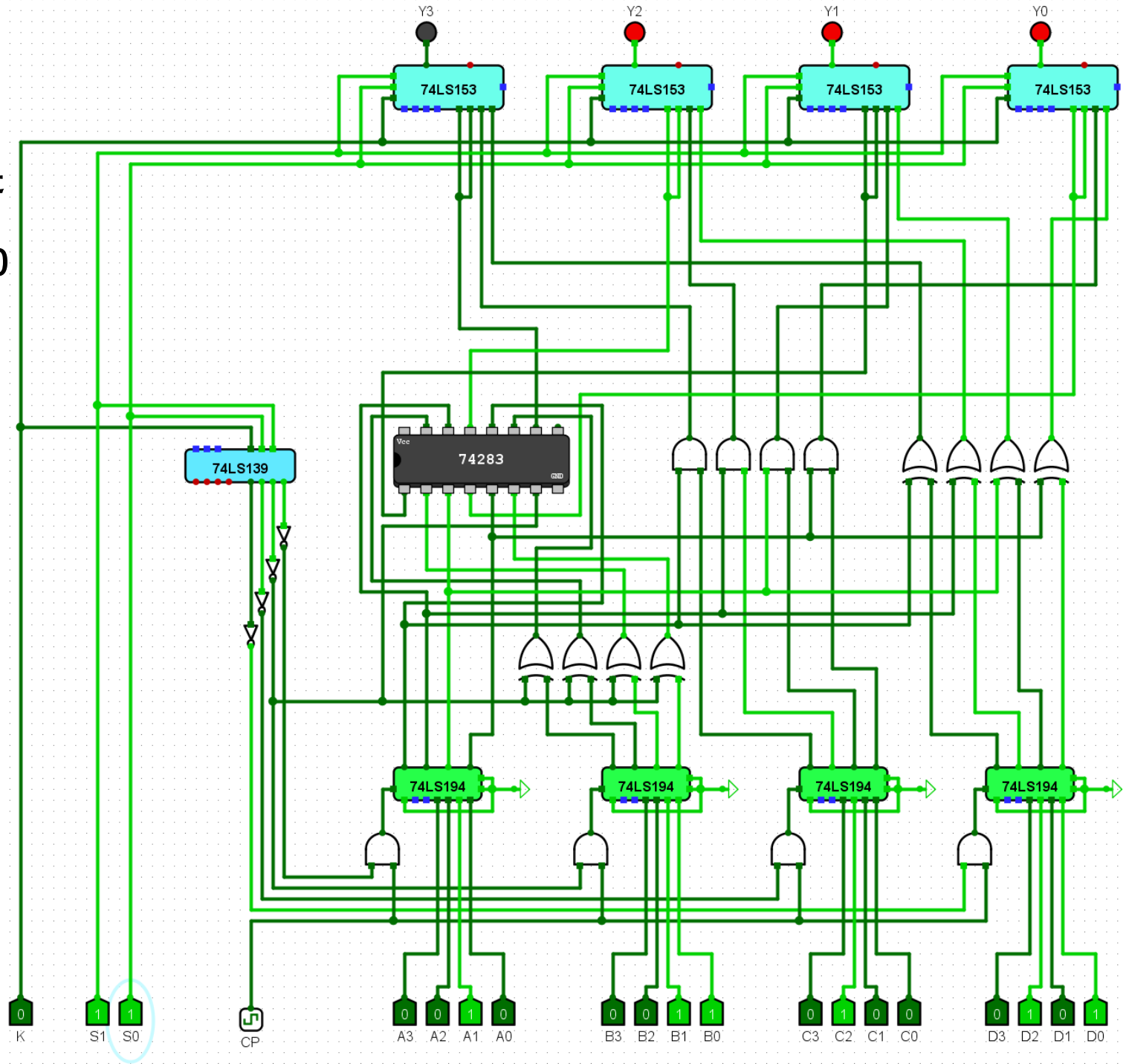
- 第8步：置S1S0=10，观看运算结果是不是为：
  A·C=0010·0100=0000=Y3Y2Y1Y0

与运算：Y=A·C



- 第9步：置S1S0=11，观看运算结果是不是为：
  A⊕D=0010⊕0101=0111=Y3Y2Y1Y0

异或运算：Y=A⊕D



2、在Logisim上实现时序信号发生器电路

- 简单运算器电路如图9.11所示：

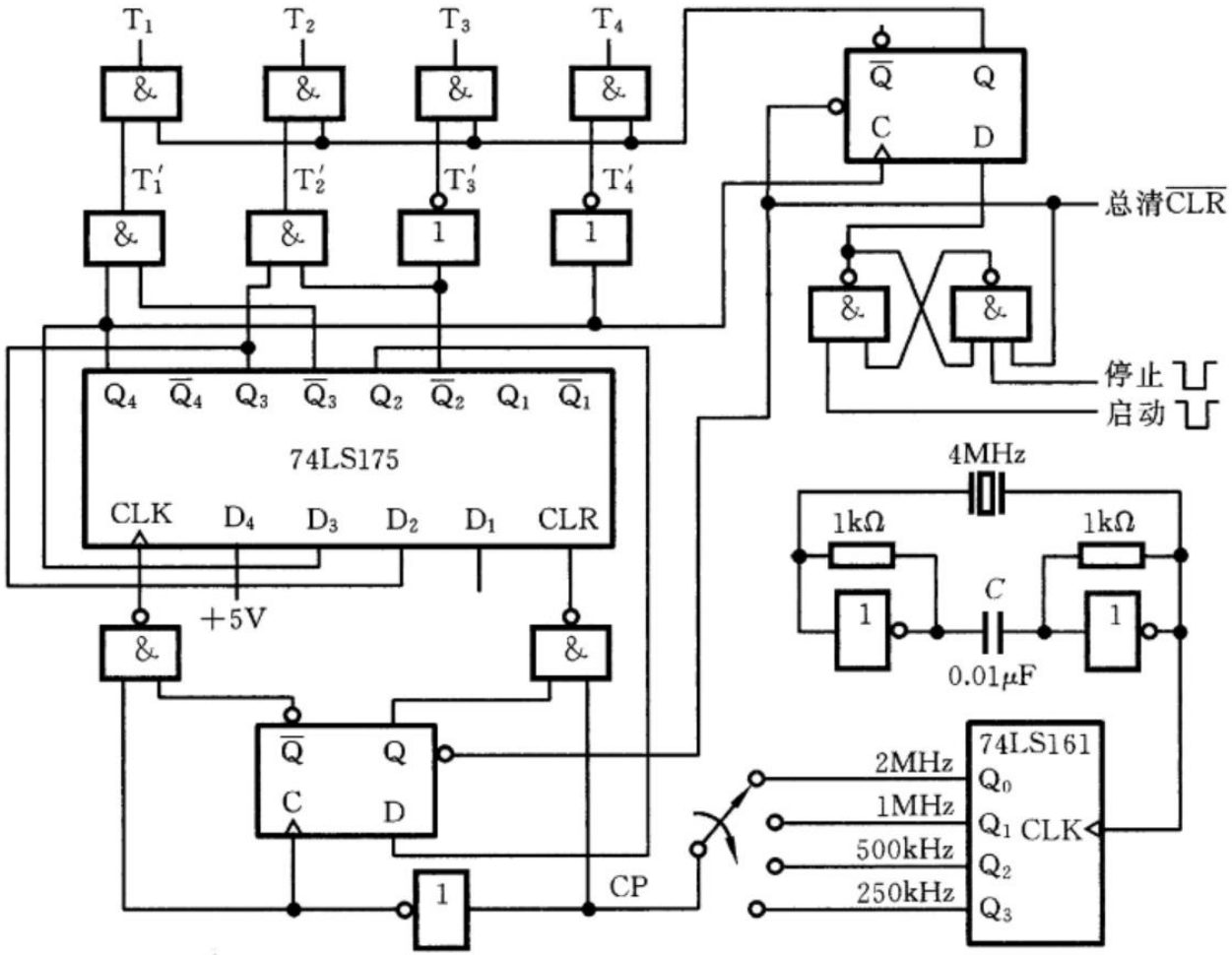
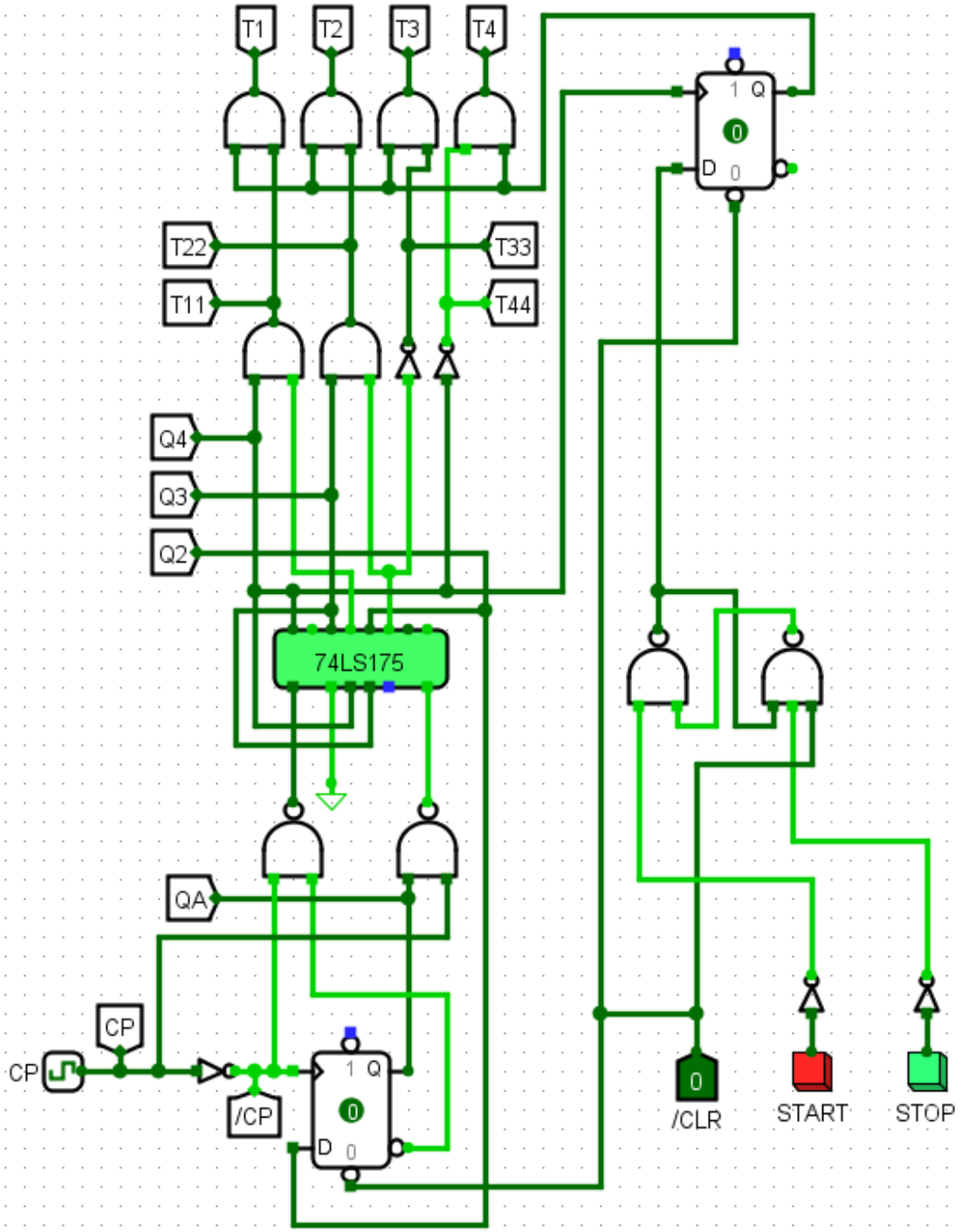
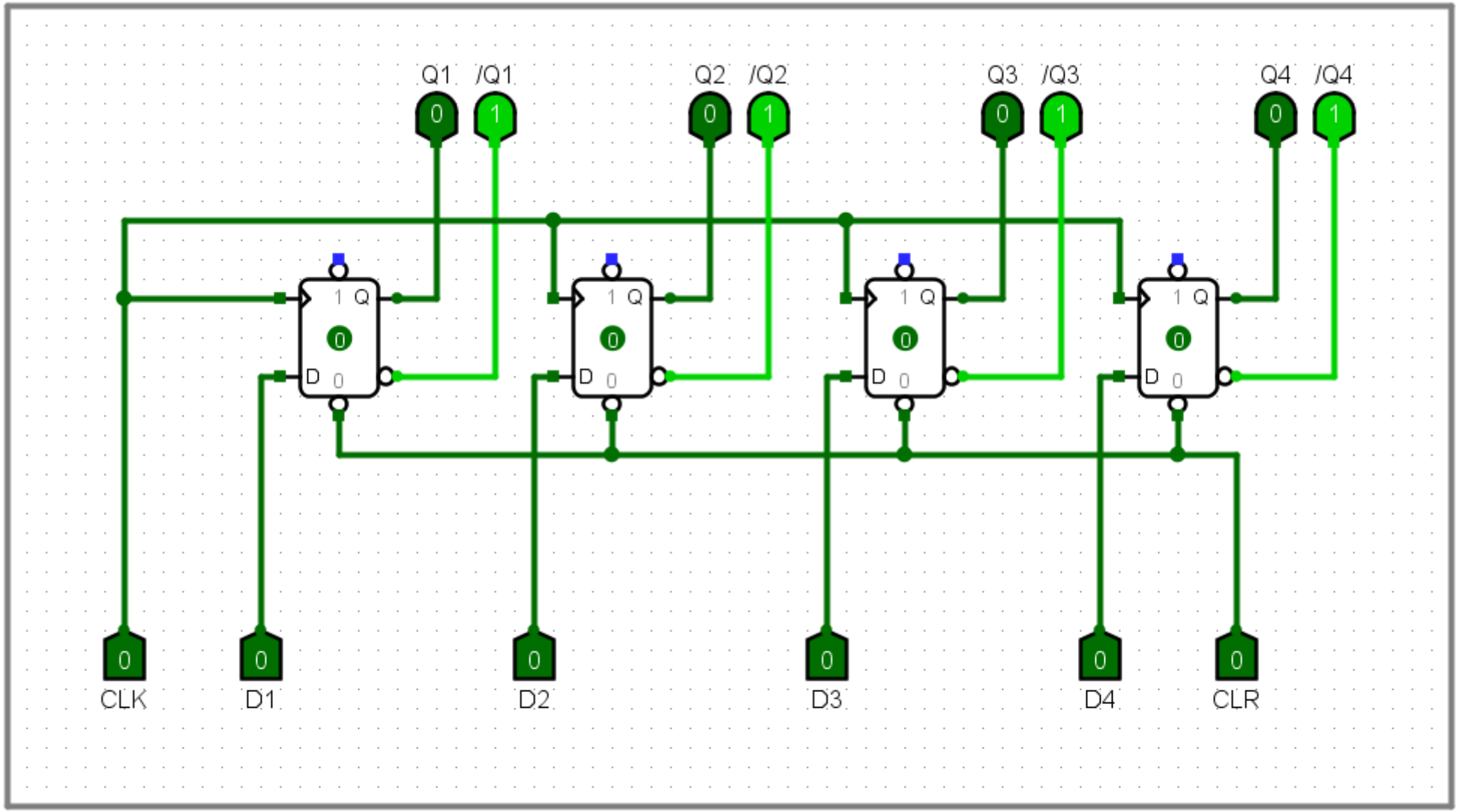


图 9.11　时序信号发生器的完整电路

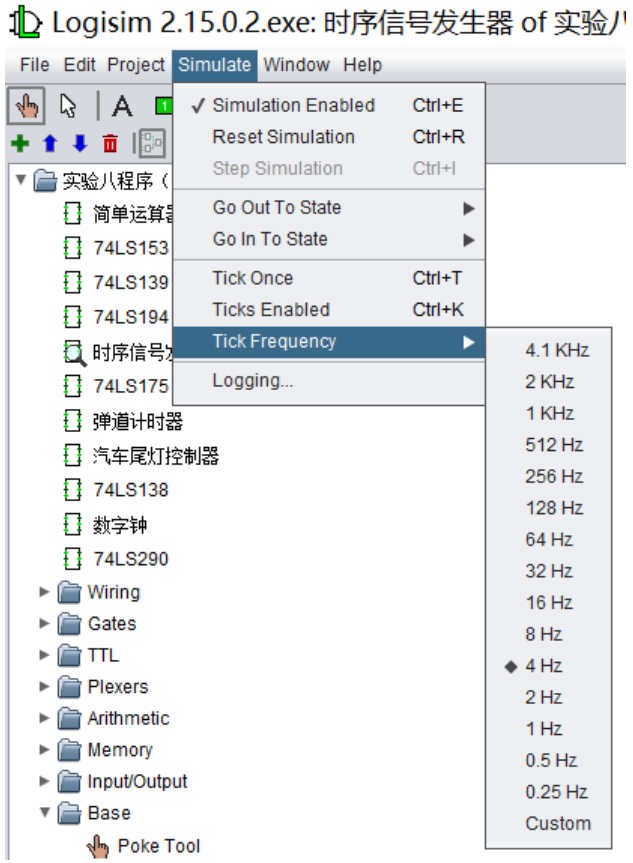
- Logisim上实现的时序信号发生器电路如下图所示：

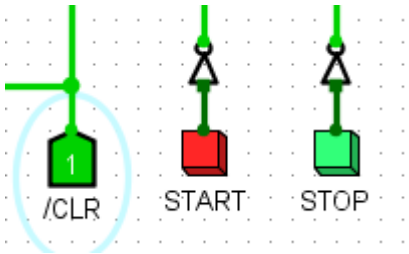




74LS175芯片（4D触发器）

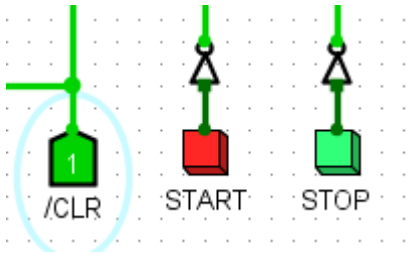
- 第1步：按Ctrl+R
- 第2步：置时钟频率=4Hz
- 第3步：置/CLR=1
- 第4步：按Ctrl+K，波形开始走动

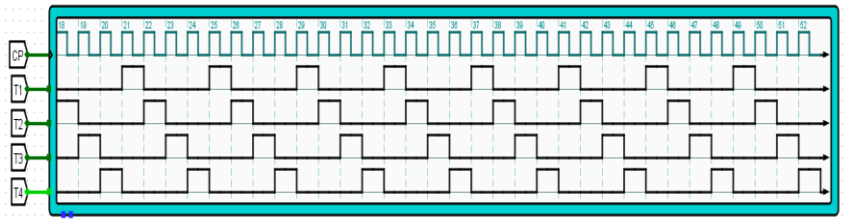


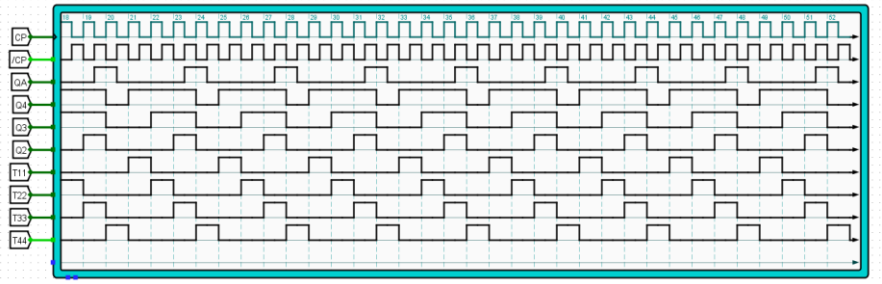


- 第5步：按STRAT

- 第6步：等所有波形都出来后，按Ctrl+K，波形停转走到，观看仿真得到的波形与教材上的波形是否一致？







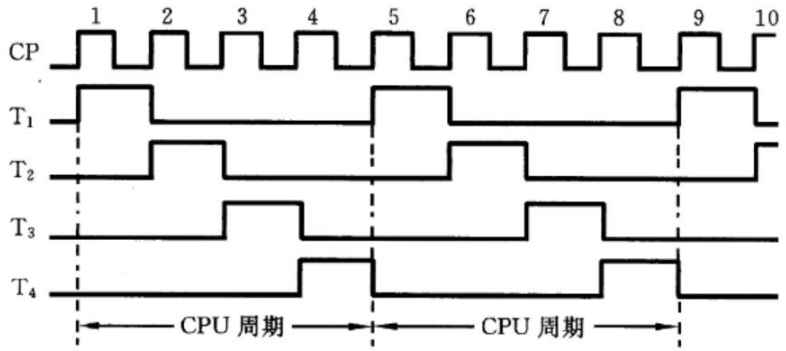


图 9.6　时序信号的波形

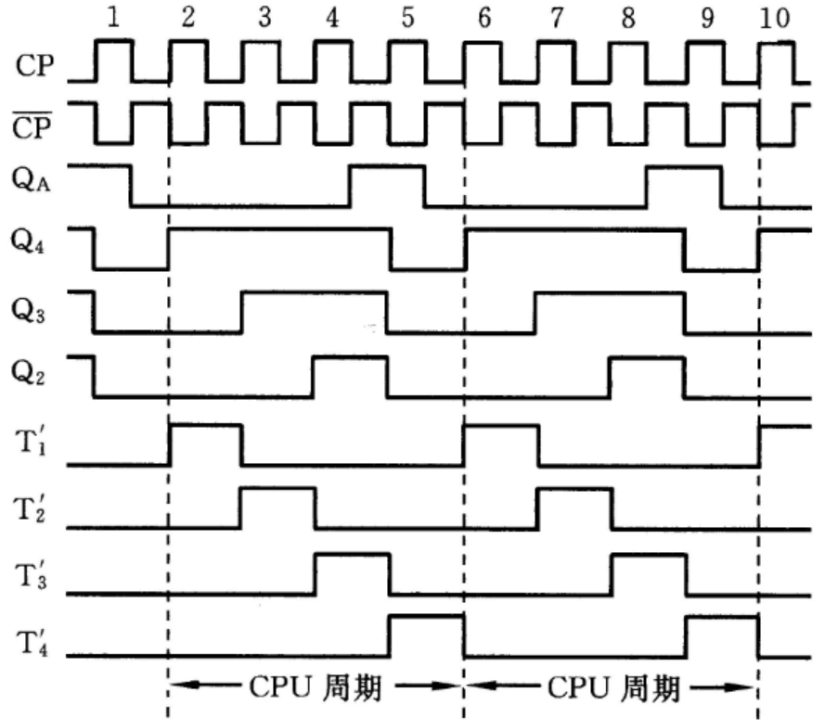
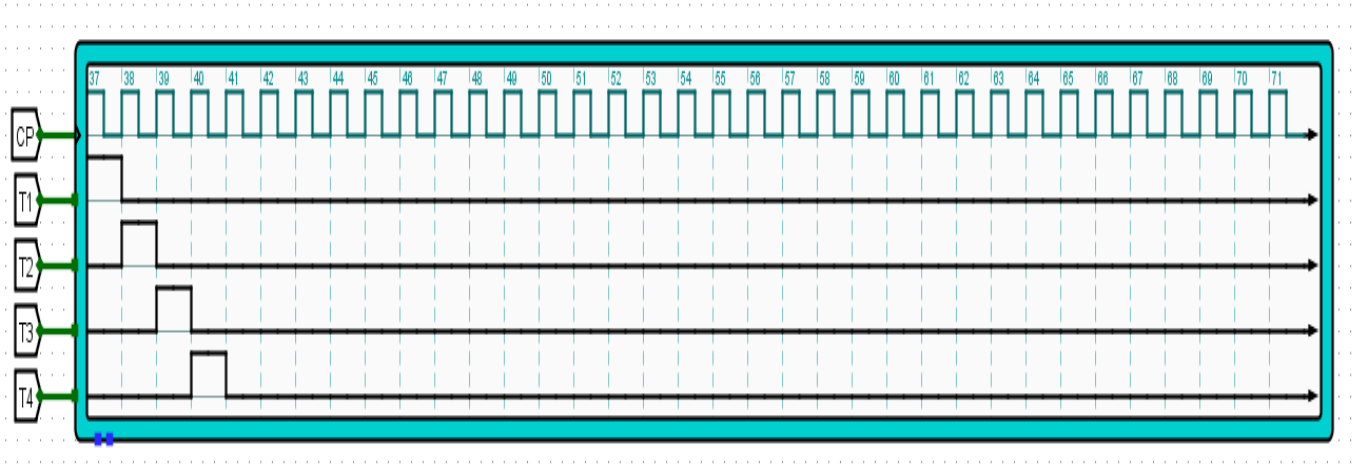


图 9.9　时序信号产生电路的波形

- 第7步：再按Ctrl+K，波形又开始走动

- 第8步：按STOP，观看是不是上面的数字示波器中波形变为直线?



3、在Logisim上实现弹道计时器电路

- **弹道计时器电路如图9.17所示：**

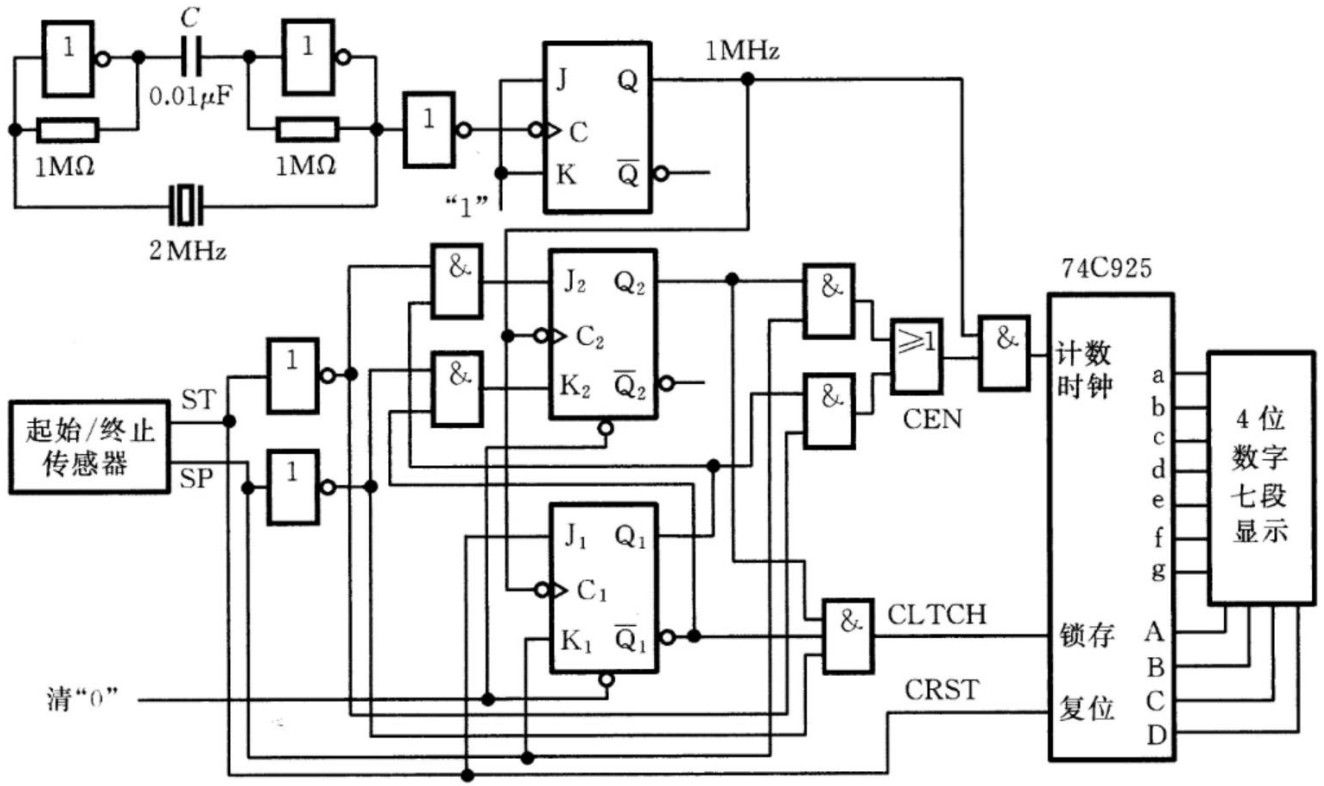
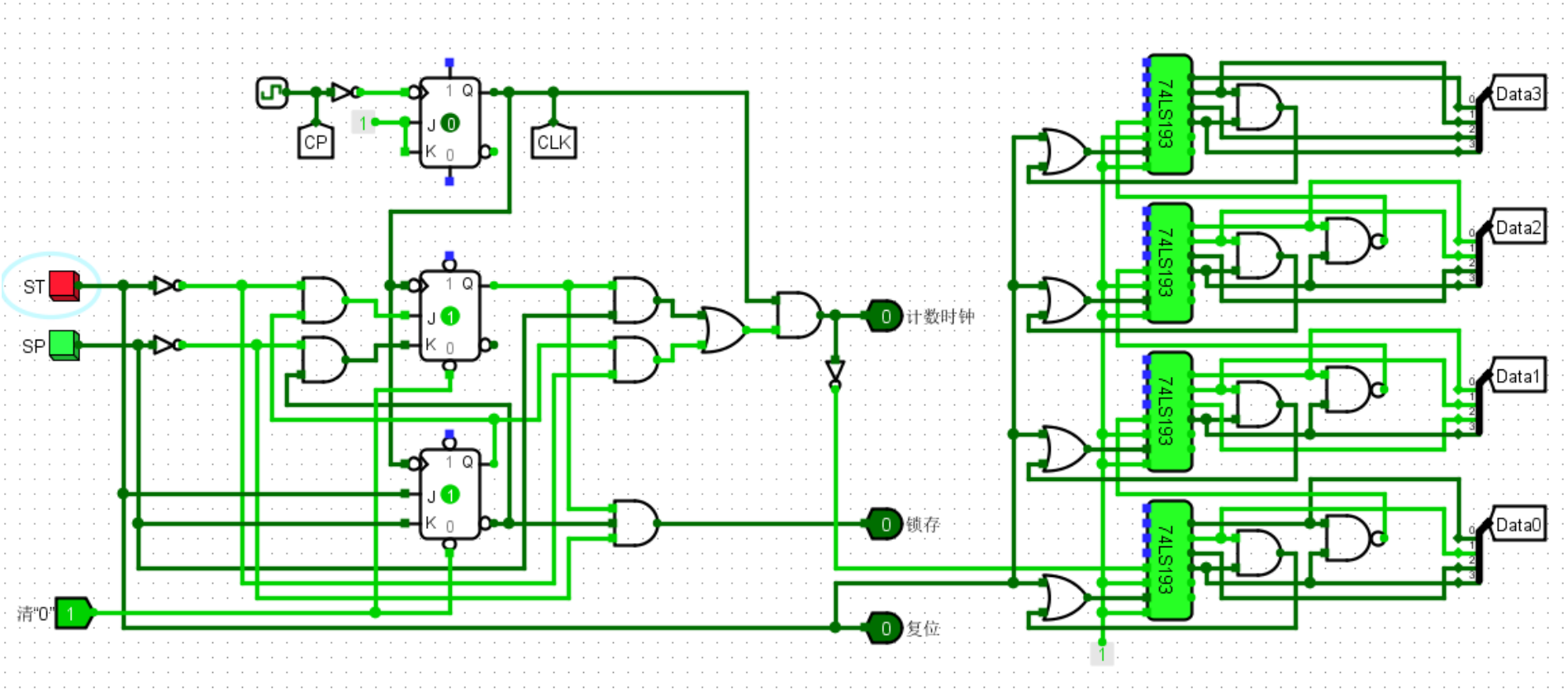
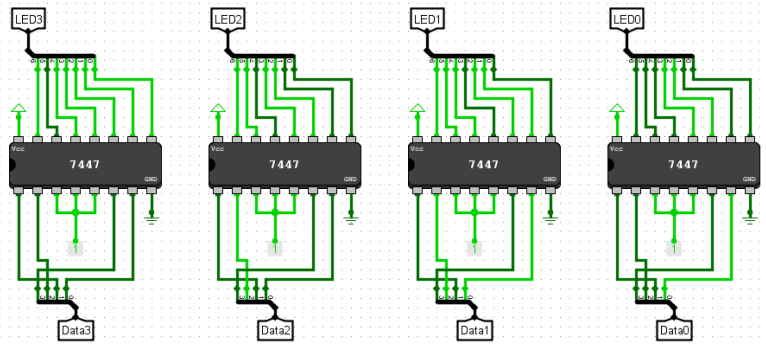
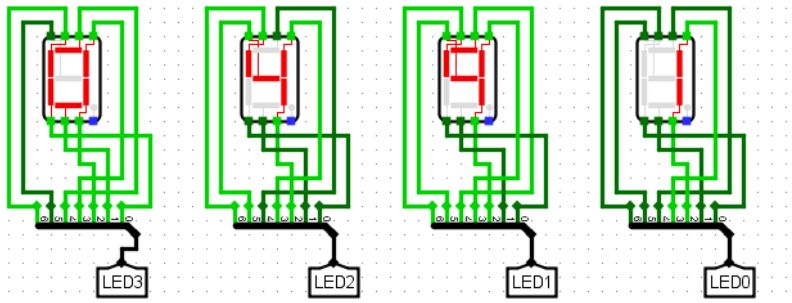


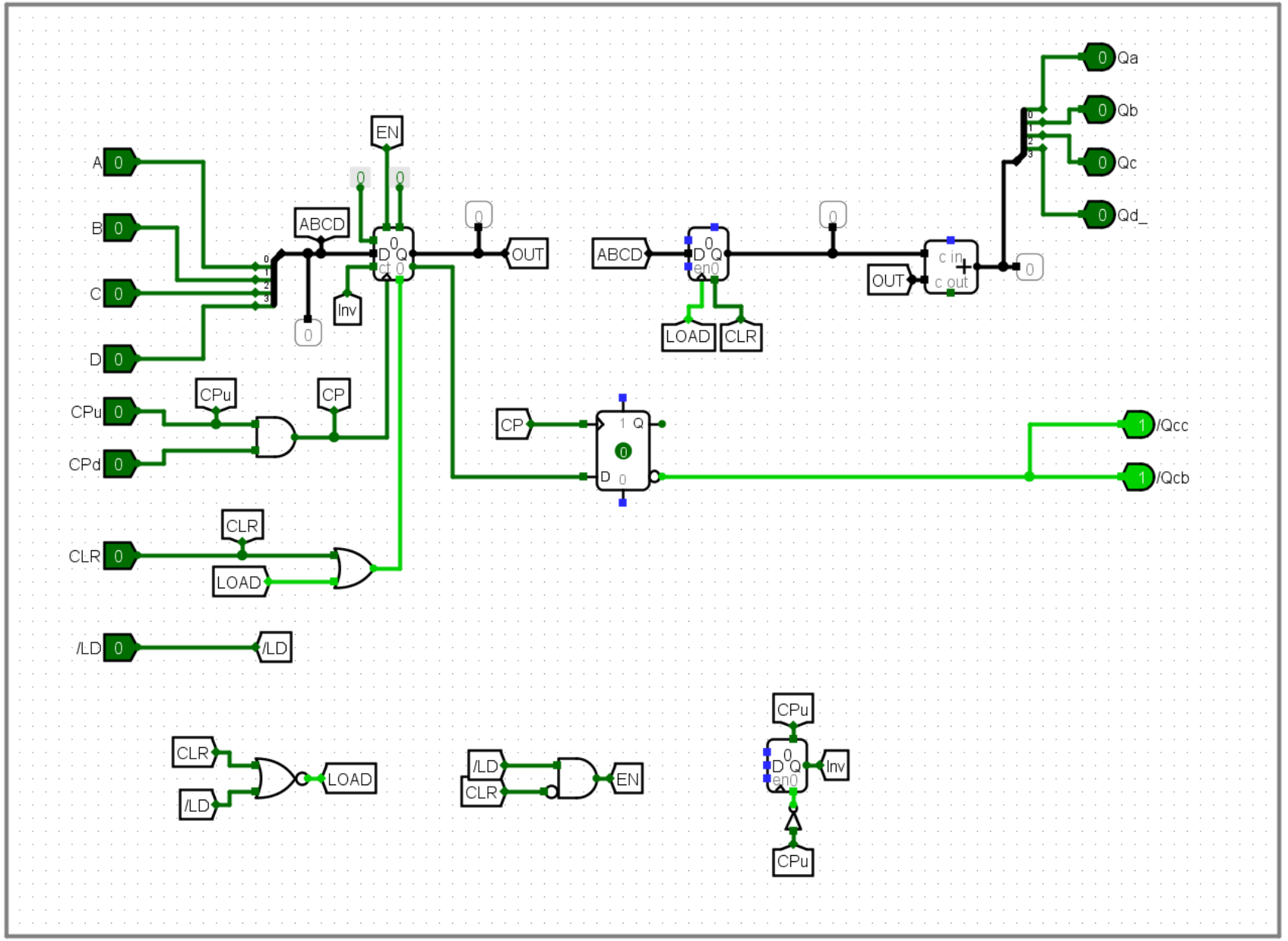
图 9.17　弹道计时器的完整电路

- Logisim上实现的弹道计时器电路如下图所示：



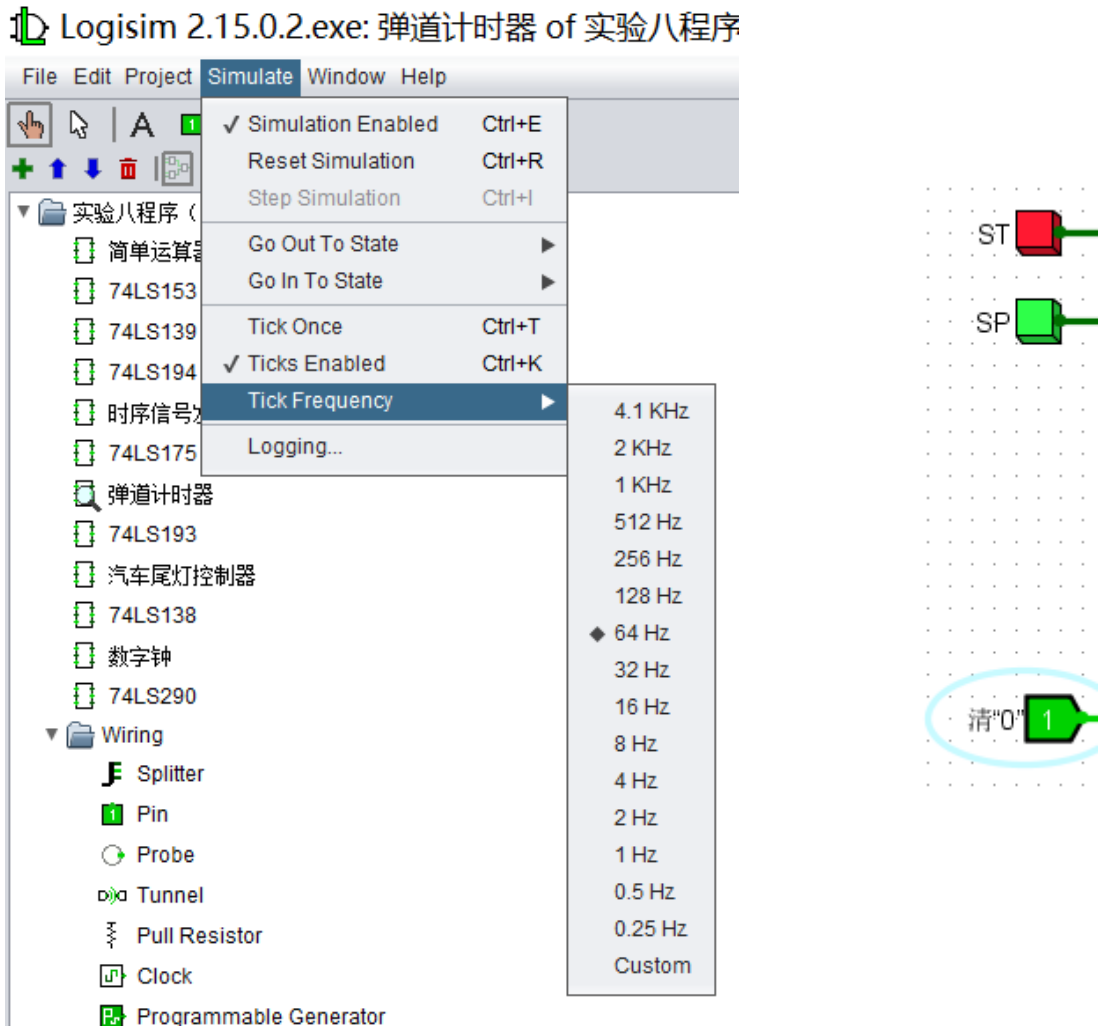






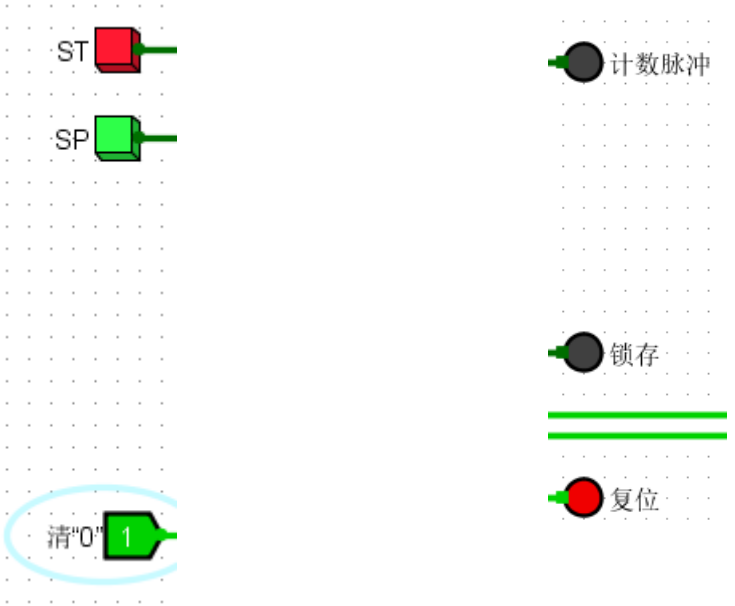
74193芯片（双时钟4位二进制同步可逆计数器）

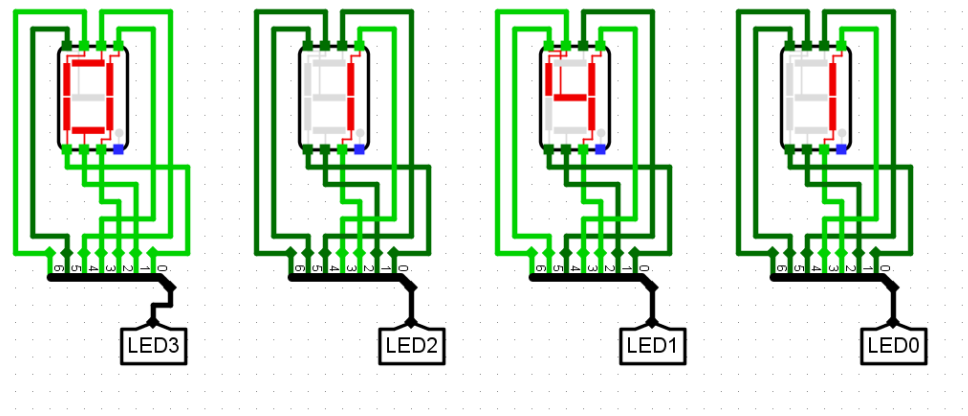
- 第1步：按Ctrl+R
- 第2步：置时钟频率=64Hz
- 第3步：置清“0”=1
- 第4步：按Ctrl+K



- 第5步：按ST按钮，此时4个数码管的数字开始变化，复位指示灯亮一下，计数脉冲指示灯会一闪一闪

- 第6步：按SP按钮，此时4个数码管的数字停止变化，锁存指示灯亮一下

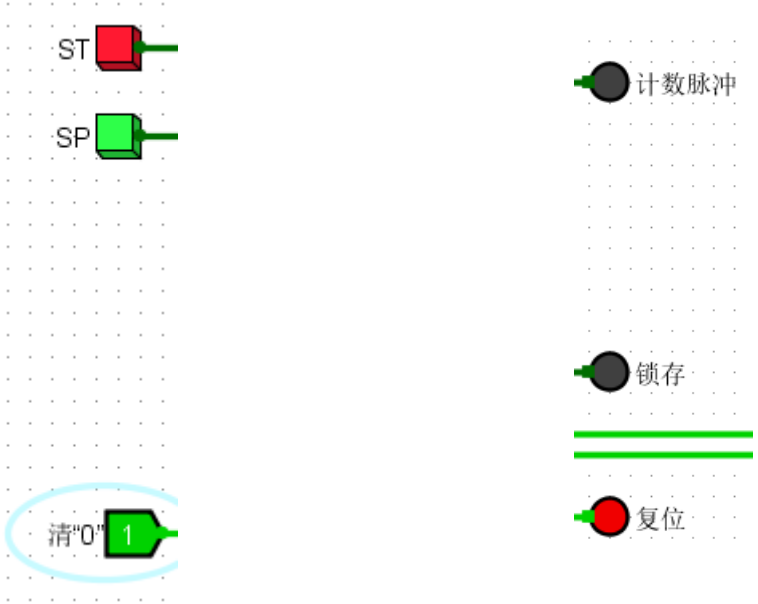


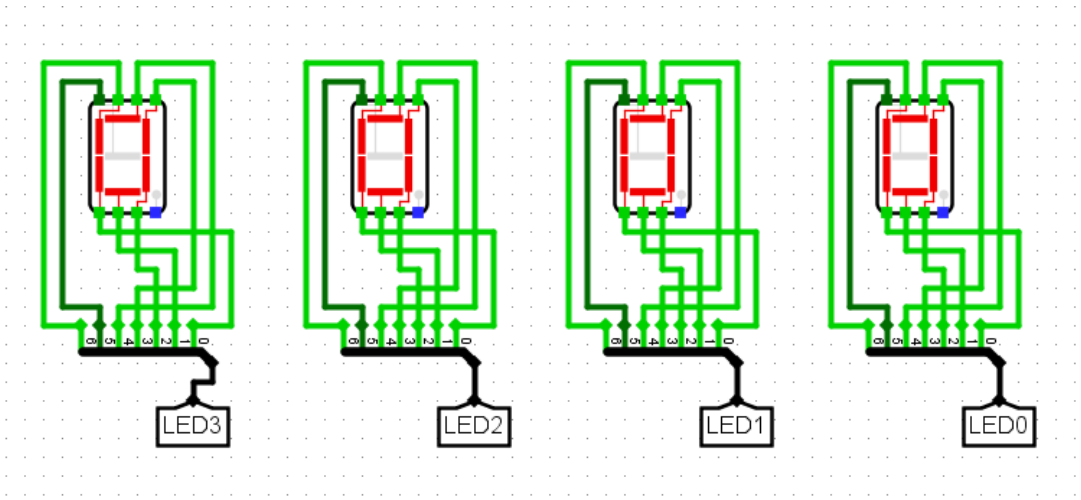


- 第7步：再按ST按钮，按住不放，此时，4个数码管的数字全部为0，复位指示灯亮一下

- 第8步：放开ST按钮，数码管的数字开始变化，计数脉冲指示灯会一闪一闪

- 第9步：再按SP按钮，数码管的数字不再变化，锁存指示灯亮一下





4、在Logisim上实现汽车尾灯控制器电路

- 简单运算器电路如图9.22所示：

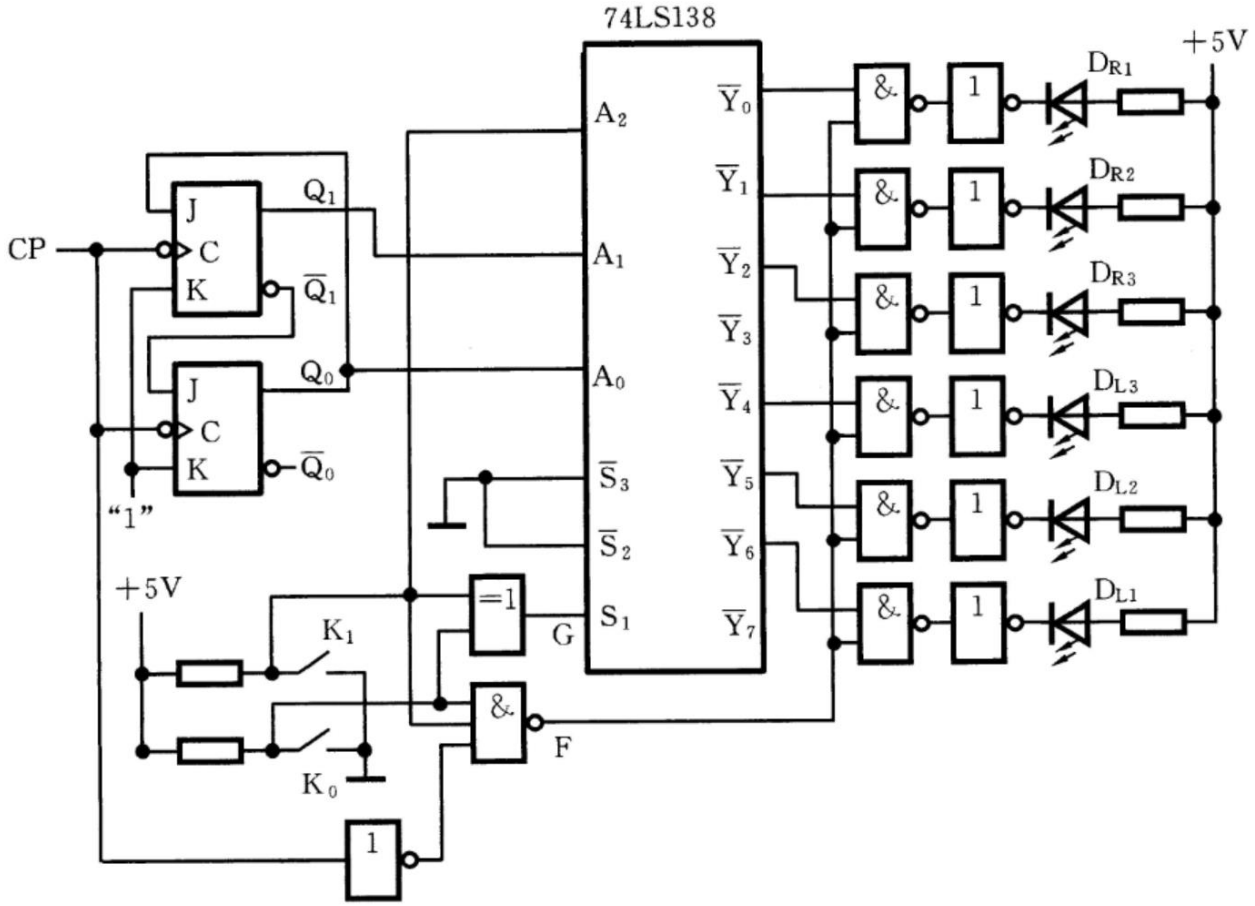
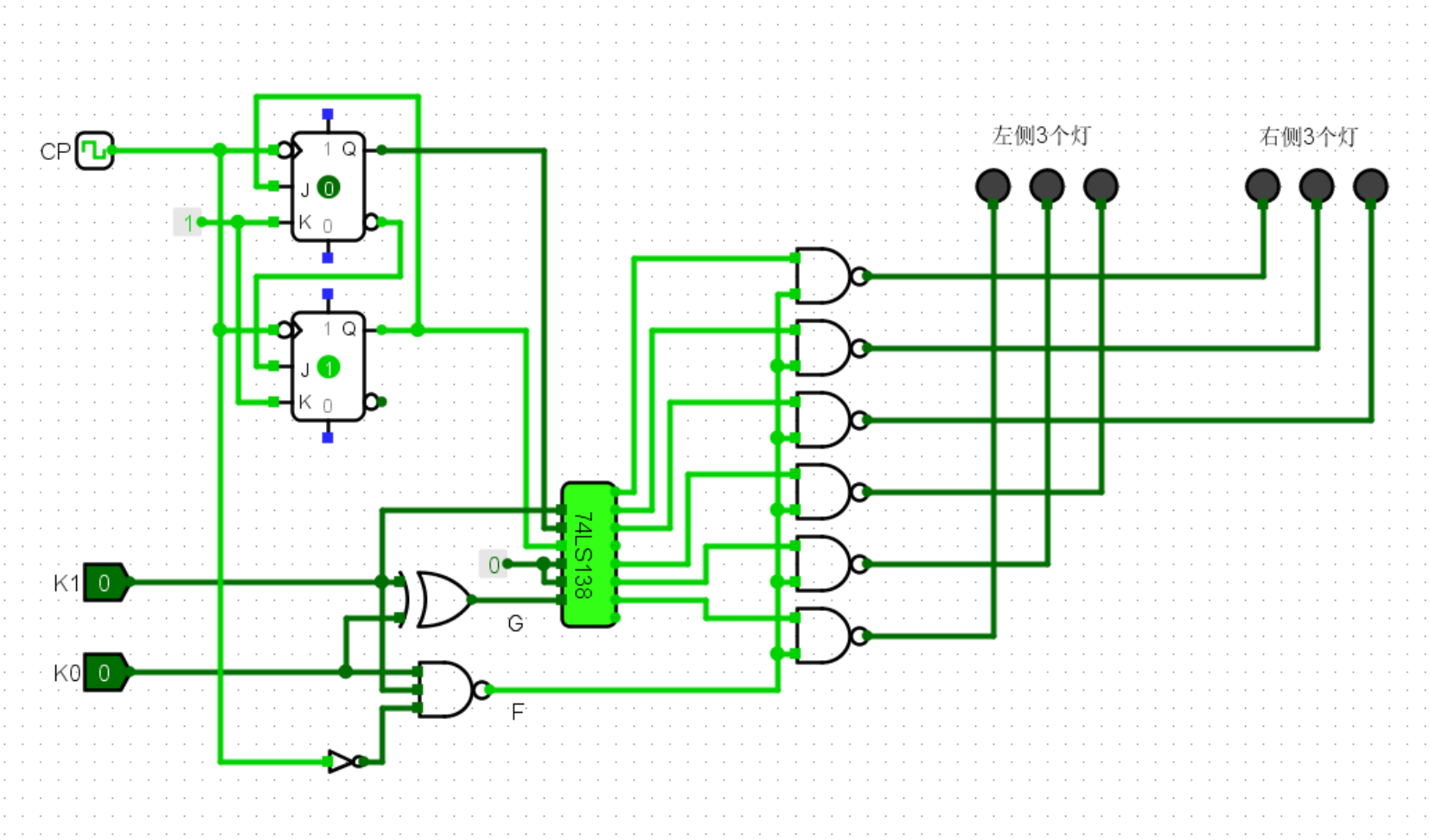
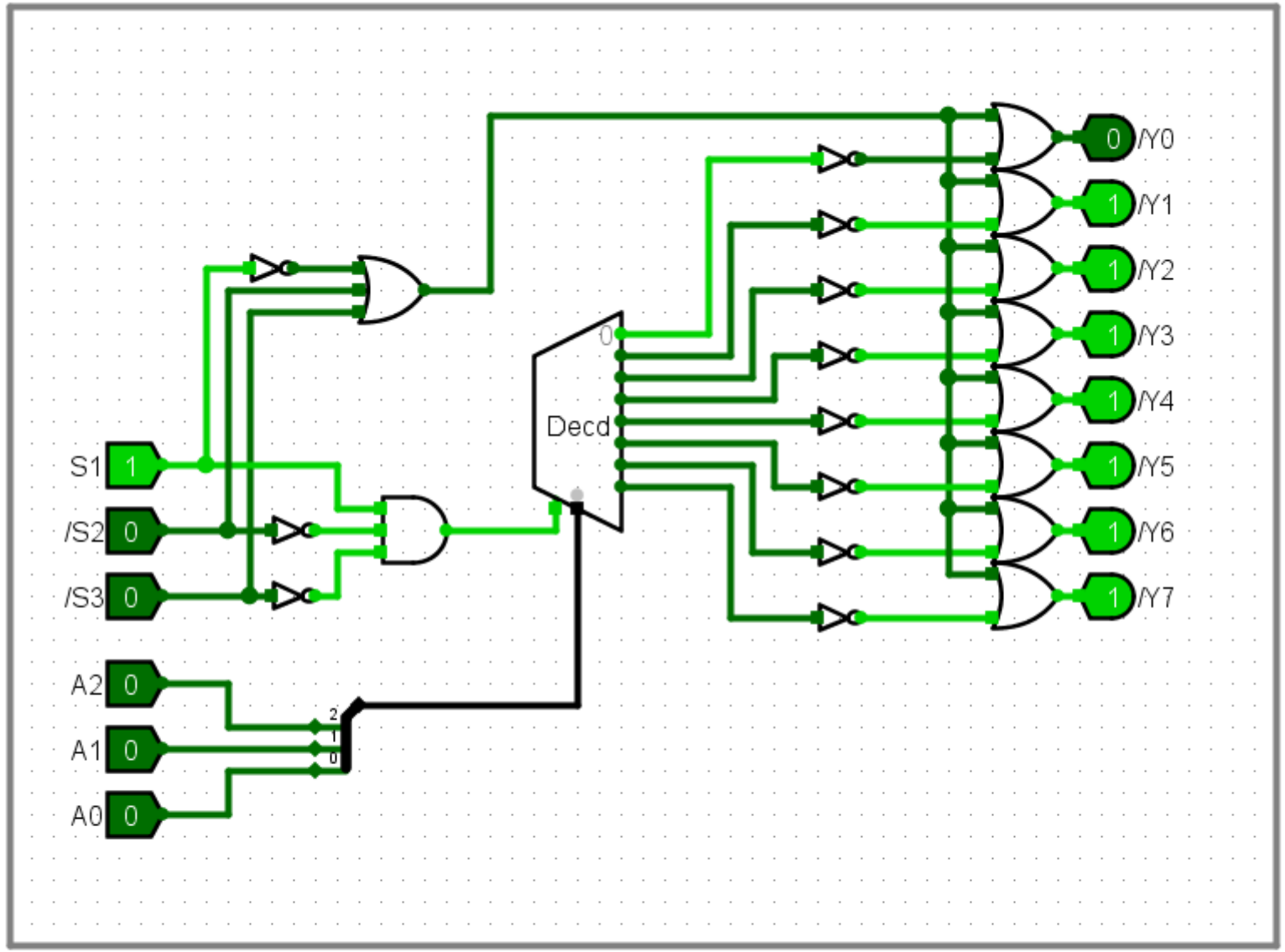


图 9.22　汽车尾灯控制器的逻辑电路

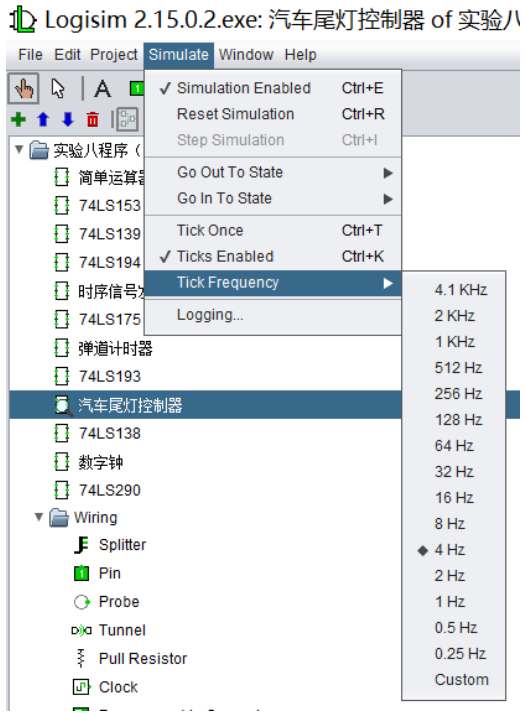
- Logisim上实现的汽车尾灯控制器电路如下图所示：

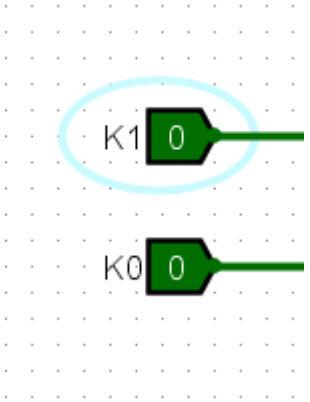


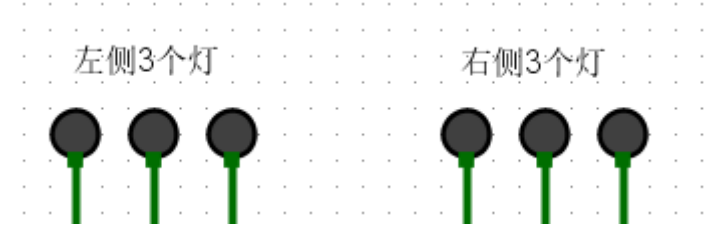


74LS138（3-8译码器）

- 第1步：按Ctrl+R
- 第2步：置时钟频率=4Hz
- 第3步：按Ctrl+K，观看K1K0是不是=00？6个LED灯是不是全灭？



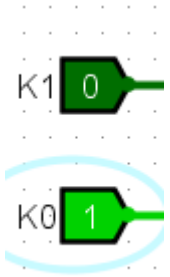


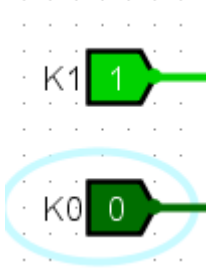


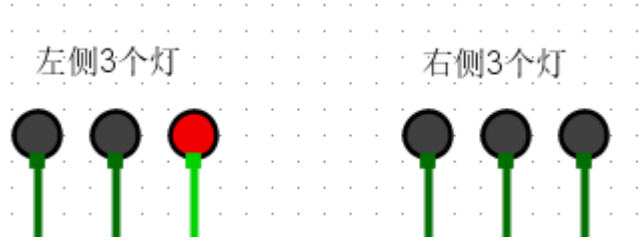
- 第4步：置K1K0=01，观察右侧3个灯是不是从左到右轮流亮？

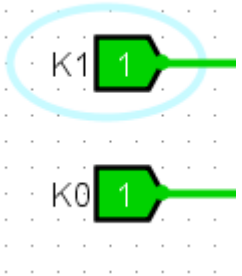
- 第5步：置K1K0=10，观察左侧3个灯是不是从右到左轮流亮？

- 第6步：置K1K0=11，观察6个灯是不是一闪一闪？









5、在Logisim上实现数字钟电路

- **数字钟电路如图9.28所示：**

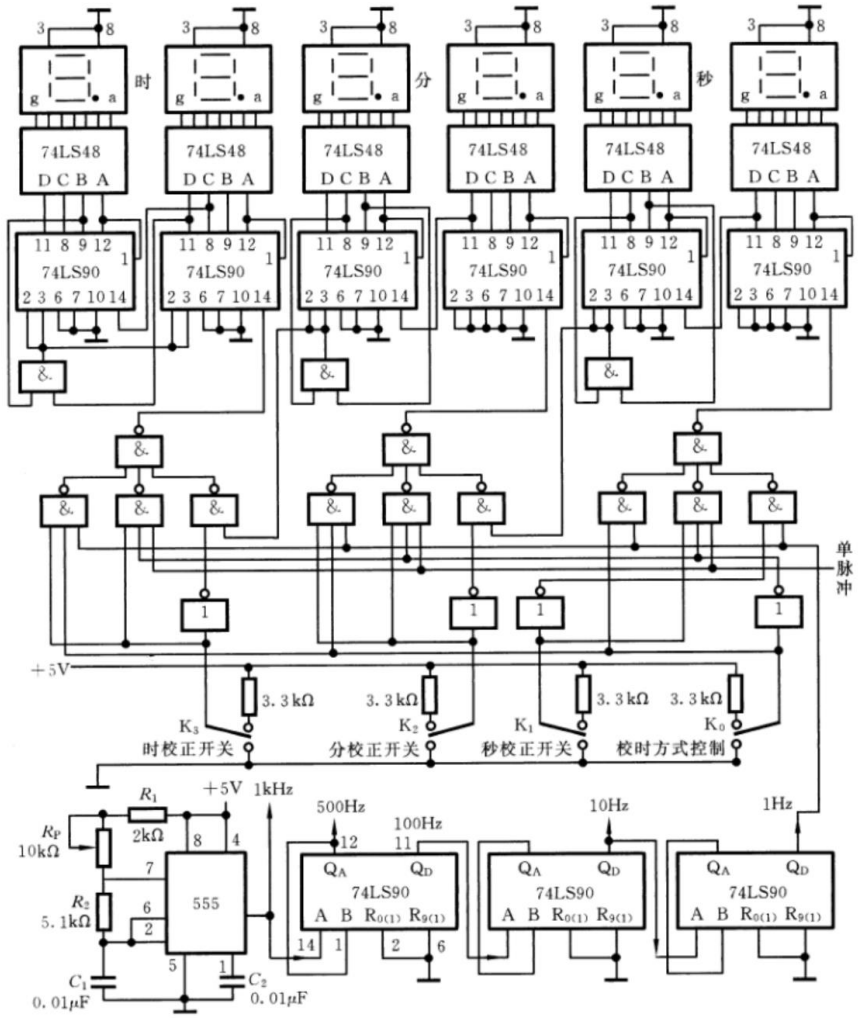
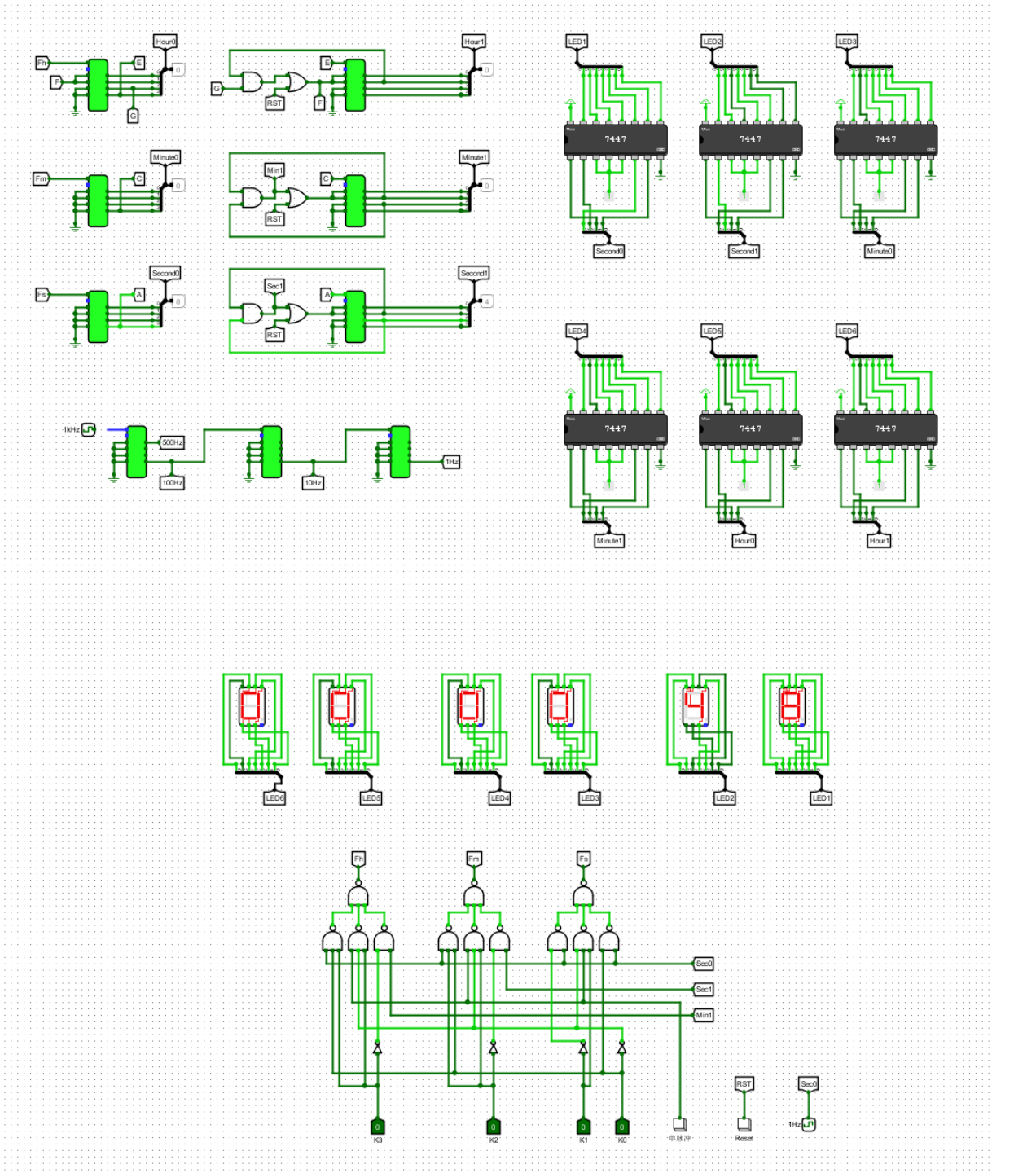
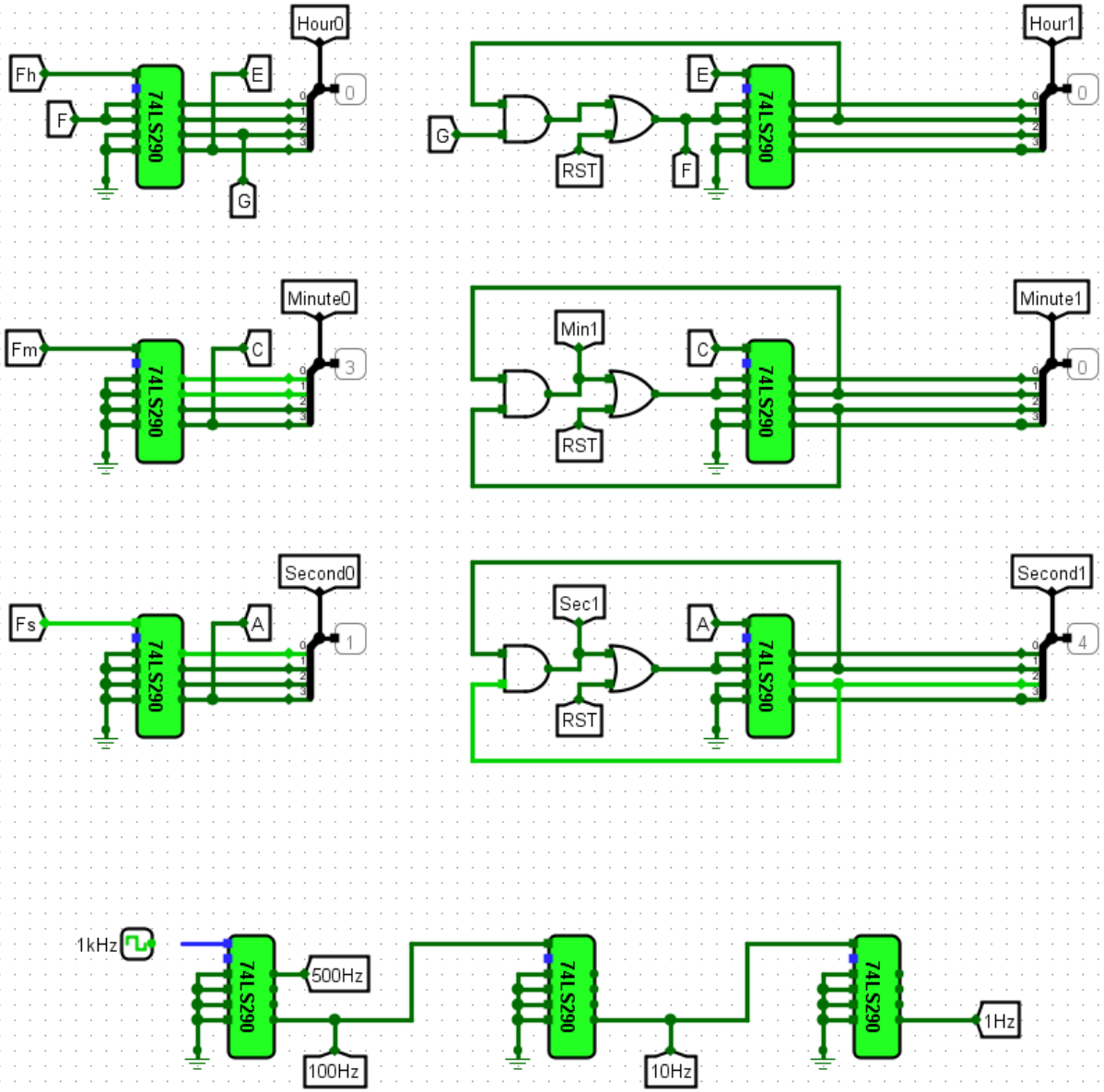
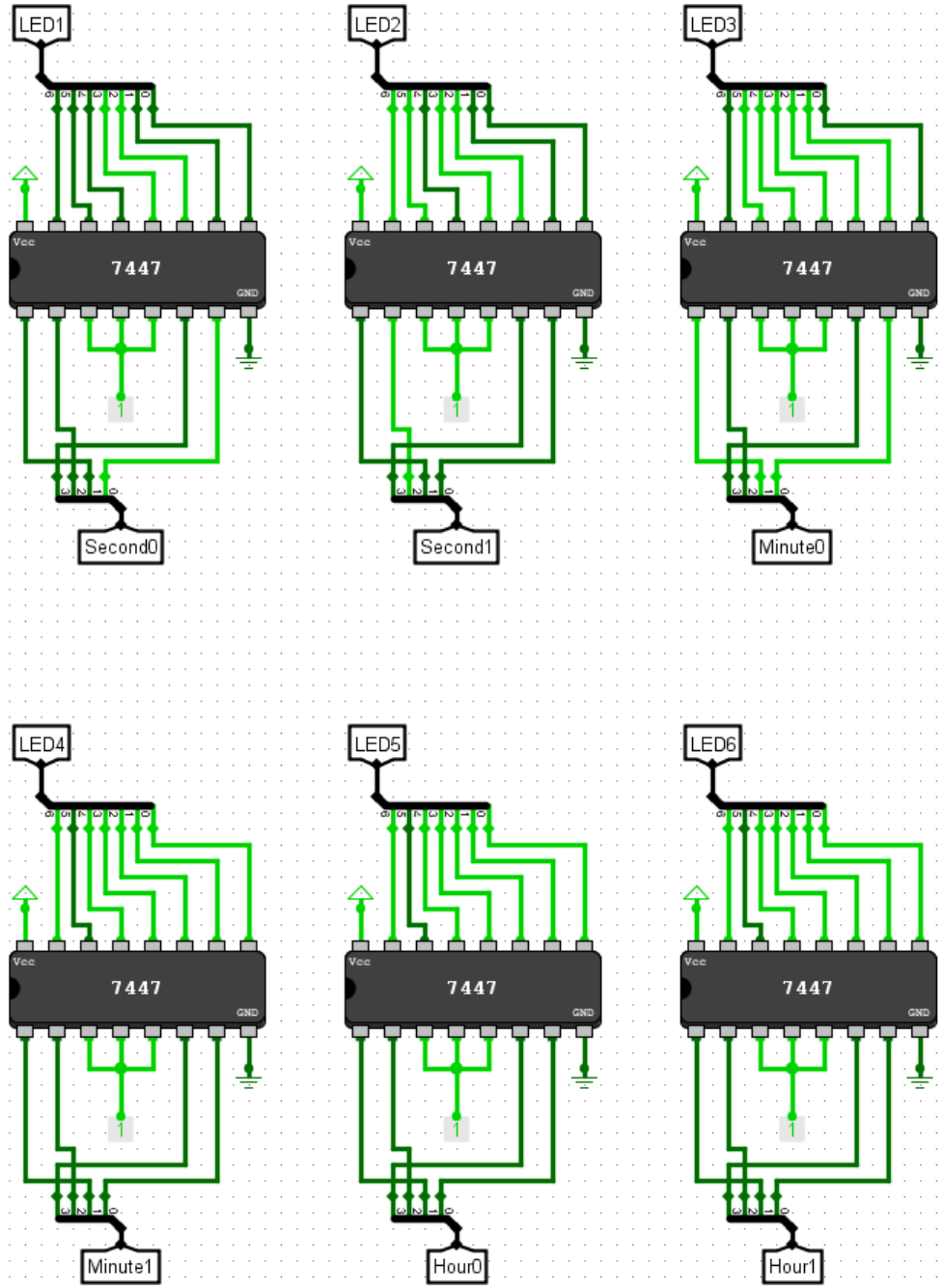


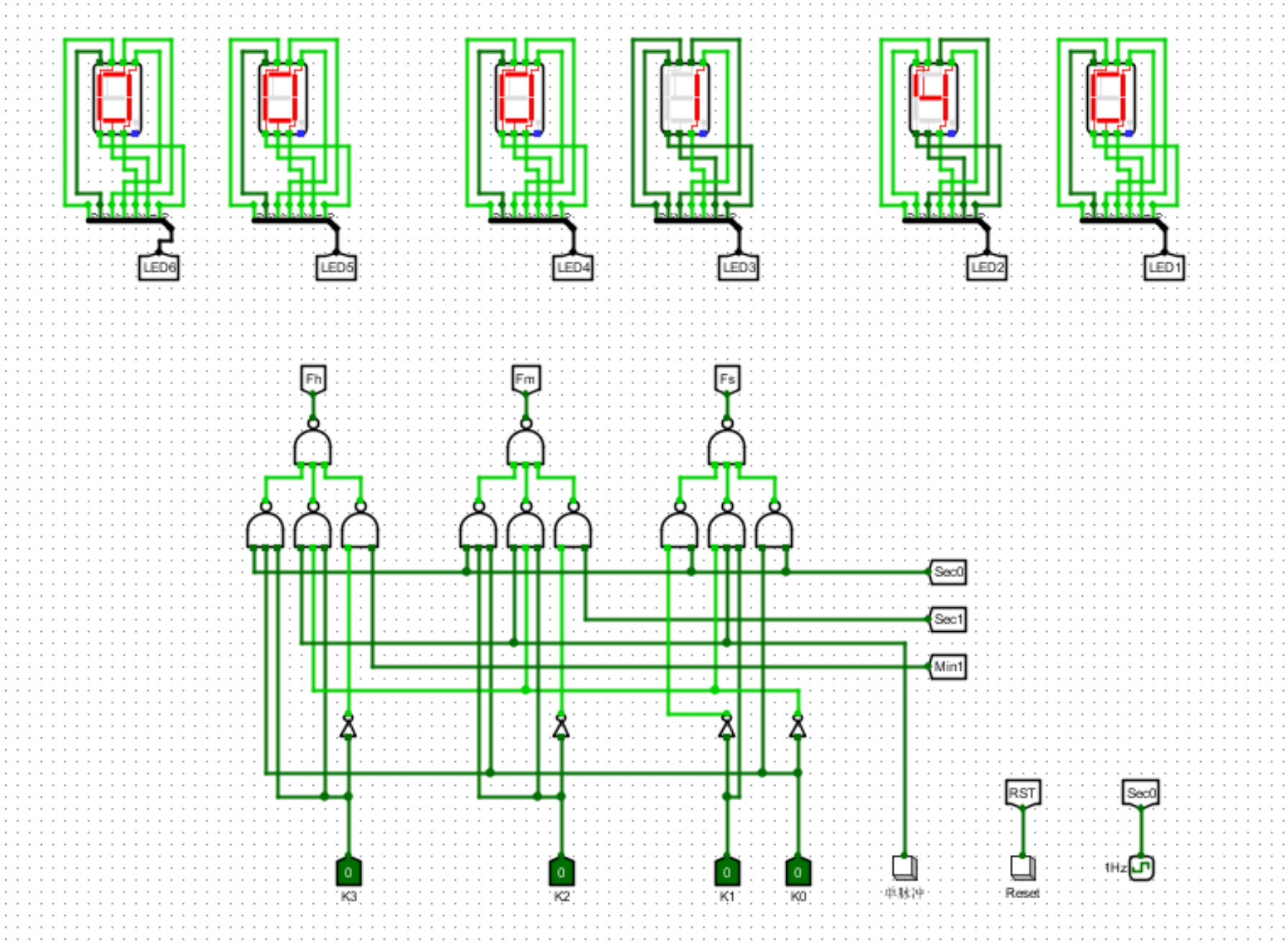
图 9.28　数字钟电路

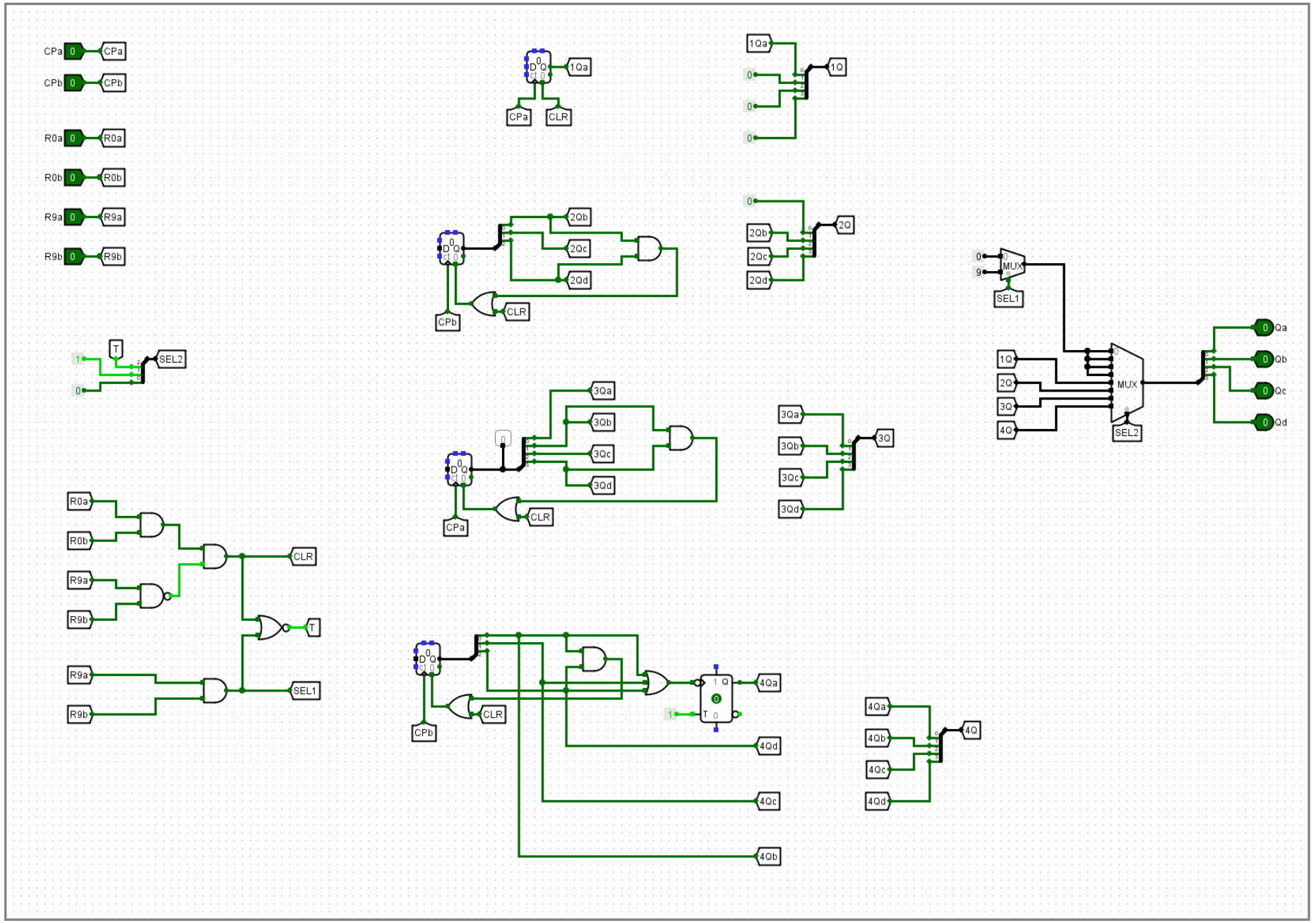
- Logisim上实现的数字钟电路如下图所示：




Hour0
Fh
F
E
G
74LS290
Hour1
RST
Minute0
Fm
C
3
Minute1
Min1
Second0
Fs
A
1
Second1
Sec1
4
1kHz
500Hz
100Hz
10Hz
1Hz

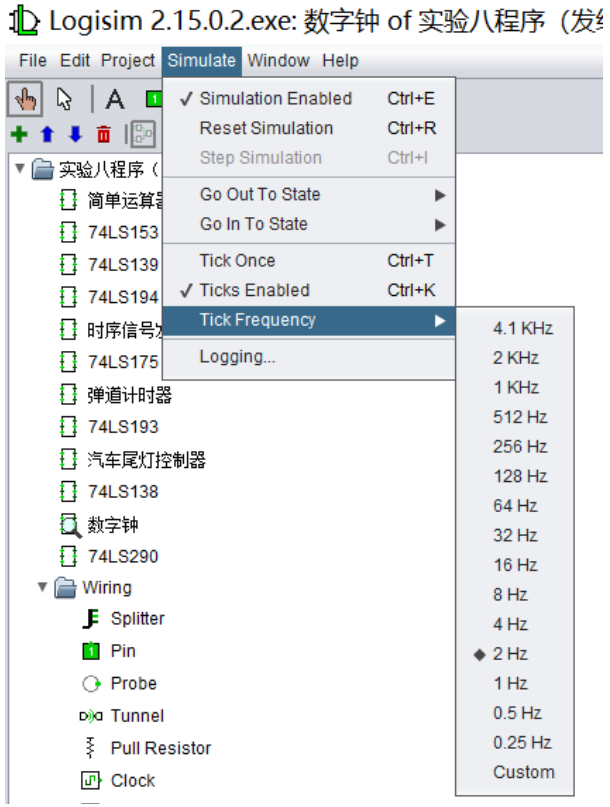

LED1
LED2
LED3
7447
Vcc
GND
Second0
Second1
Minute0
LED4
LED5
LED6
Minute1
Hour0
Hour1

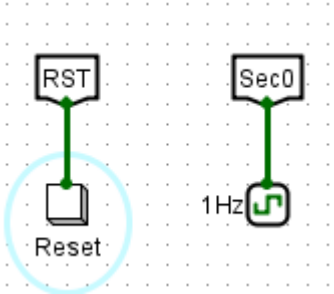

LED6
LED5
LED4
LED3
LED2
LED1
Fh
Fm
Fs
Sec0
Sec1
Min1
0
K3
0
K2
0
K1
0
K0
中断冲
RST
Reset
Sec0
1Hz

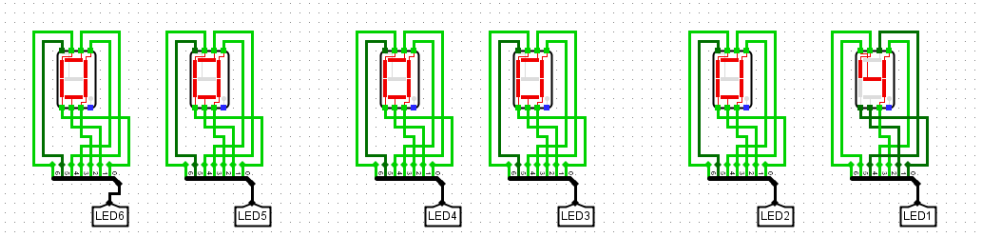


74290芯片（二-五-十进制加法计数器）

- 第1步：按Ctrl+R
- 第2步：置时钟频率=2Hz
- 第3步：按Ctrl+K，此时数码管上的数字钟开始走动
- 如果电路有异常，请按Reset按钮

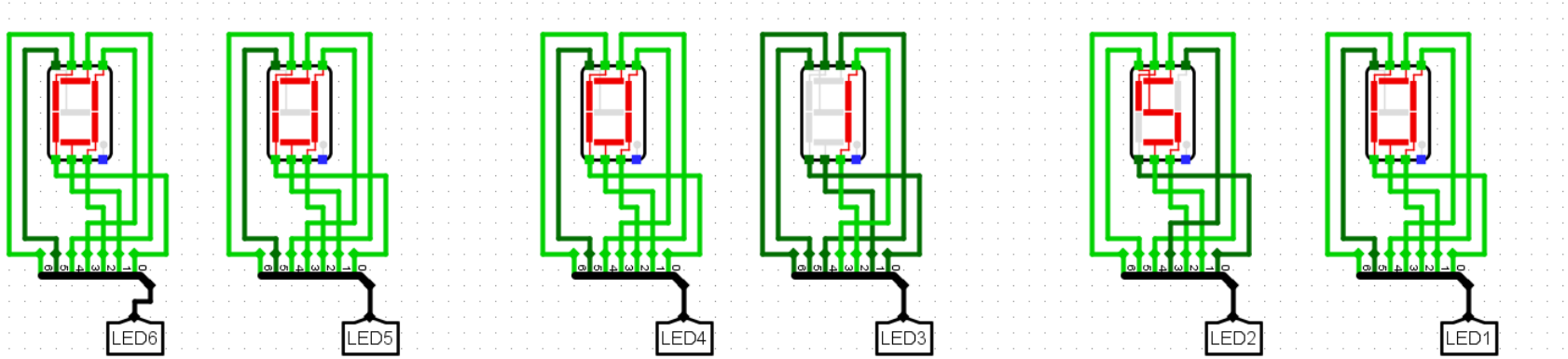




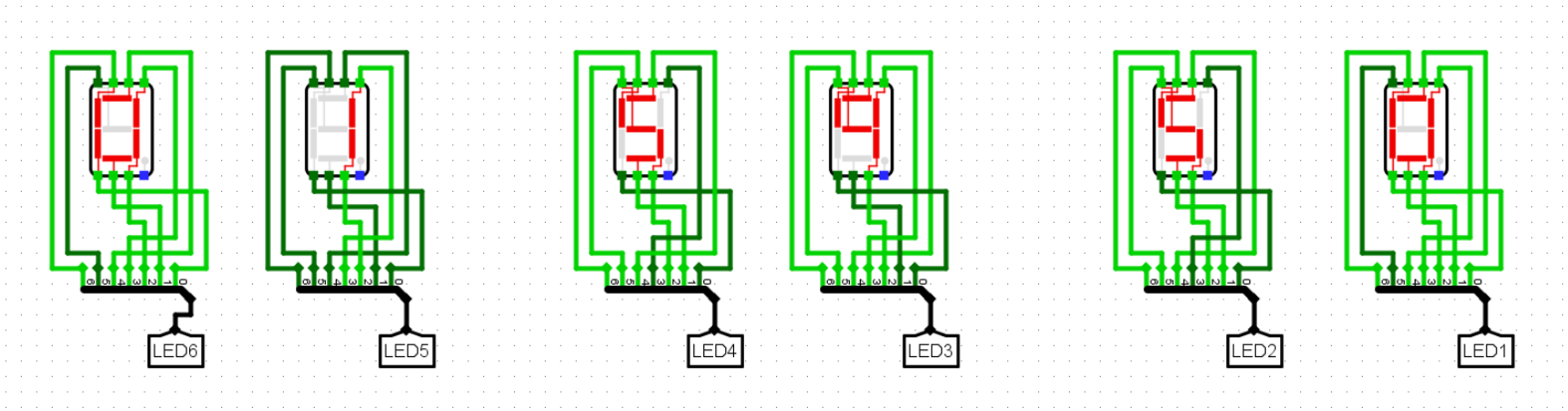


- 第4步：按Ctrl+K，数字钟停转走动

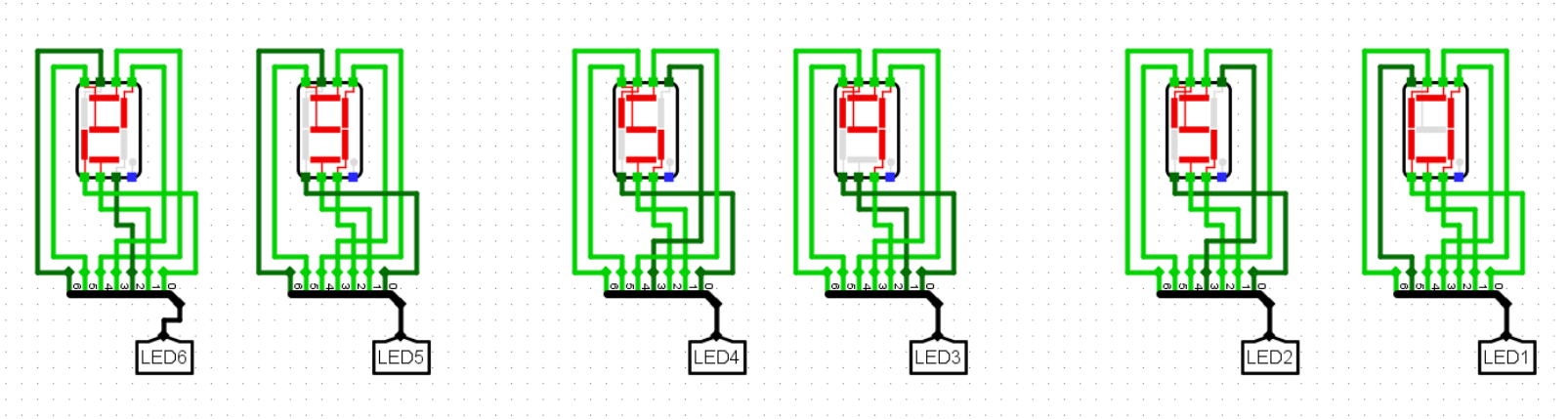
- 第5步：置K1=1，此时数字钟停转走动，按“单脉冲”按钮，会改变“秒”的数字



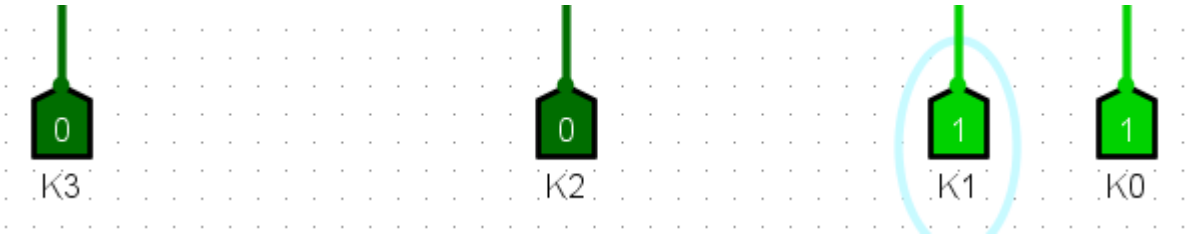
- 第6步：置K2=1，此时数字钟停转走动，按“单脉冲”按钮，会改变“分”的数字

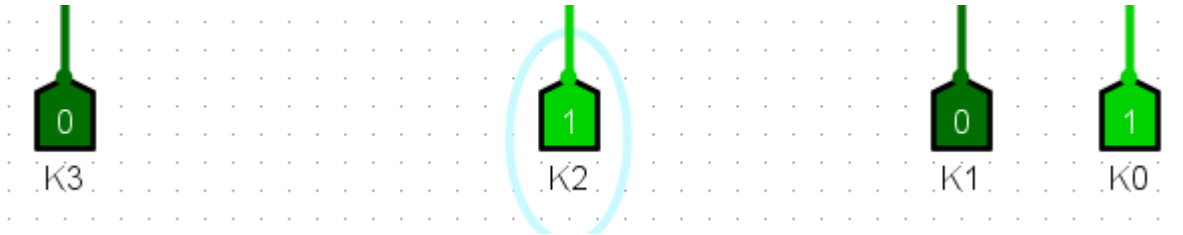


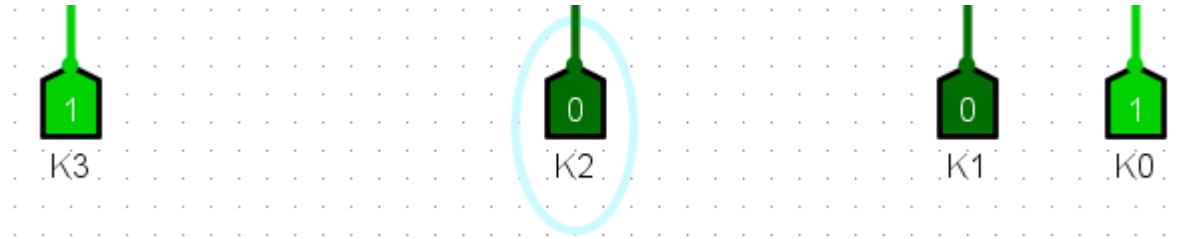
- 第7步：置K3=1，此时数字钟停转走动，按“单脉冲”按钮，会改变“时”的数字



- 第8步：也可以自动校时，此时，置K0=1，按Ctrl+K，数字钟开始走动，然后分别置K1=1、K2=1、K3=1，就可以实现自动校时



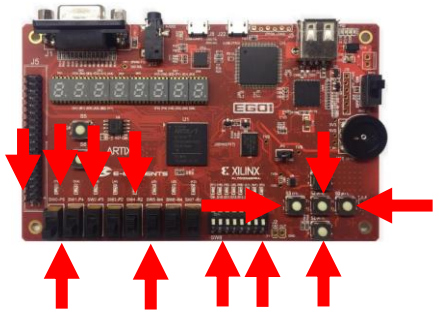




第二部分：在FPGA开发板上实现综合应用举例电路

1、在FPGA开发板上实现简单运算器电路

- 采用行为描述方式实现9.1小节的简单运算器电路。该电路的输入为4个4位二进制数A、B、C、D，4个运算控制按钮，输出为4位二进制数F。



EGO1开发板

- 输入A为开发板上左边的4个拨动开关，输入B为开发板上右边的4个拨动开关，输入C为开发板上左边的4个DIP开关，输入D为开发板上右边的4个DIP开关，加法运算控制按钮为开发板上的S4按键，减法运算控制按钮为开发板上的S1按键，与运算控制按钮为开发板上的S3按键，异或运算控制按钮为开发板上的S0按键，输出F为开发板上左边的4个LED灯。

- 验证步骤：
  - 运行程序后，通过8个拨动开关和8个DIP开关设置A、B、C、D值，然后分别按S4、S1、S3、S0按键，观看4个LED灯，验证是否实现加、减、与、异或运算？

example_9_1_EGO1.v - 记事本

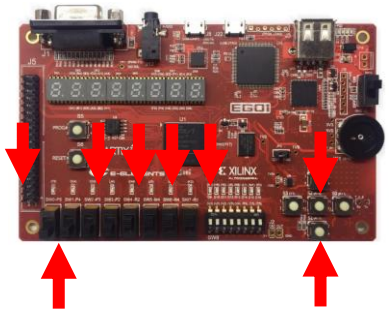
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现9.1小节的简单运算器电路，输入为开发板上8个拨动开关、8个DIP开关、4个按键，输出为开发板上最左边的4个LED灯。
//最左边的4个拨动开关为输入A，最右边的4个拨动开关为输入B，最左边的4个DIP开关为输入C，最右边的4个DIP开关为输入D，按键S4为加法运算、按键S1为减法运算、按键S3为与运算、按键S0为异或运算。

`timescale 1ns / 1ps

module example_9_1(
    input sw_pin[7:0],                                  //8个拨动开关
    input dip_pin[7:0],                                 //8个DIP开关
    input btn_0, btn_1, btn_3, btn_4,                   //S0、S1、S3、S4按键
    output reg [15:0] led_pin                           //16个led灯
);

    always @(*)                                         //行为描述方式
    begin
            if(btn_4 == 1)
                    begin
                            {led_pin[0], led_pin[1], led_pin[2], led_pin[3]} <= {sw_pin[0], sw_pin[1], sw_pin[2], sw_pin[3]} + {sw_pin[4], sw_pin[5], sw_pin[6], sw_pin[7]};
                    end
            if(btn_1 == 1)
                    begin
                            {led_pin[0], led_pin[1], led_pin[2], led_pin[3]} <= {sw_pin[0], sw_pin[1], sw_pin[2], sw_pin[3]} - {sw_pin[4], sw_pin[5], sw_pin[6], sw_pin[7]};
                    end
            if(btn_3 == 1)
                    begin
                            led_pin[0] <= sw_pin[0] & dip_pin[0];
                            led_pin[1] <= sw_pin[1] & dip_pin[1];
                            led_pin[2] <= sw_pin[2] & dip_pin[2];
                            led_pin[3] <= sw_pin[3] & dip_pin[3];
                    end
            if(btn_0 == 1)
                    begin
                            led_pin[0] <= sw_pin[0] ^ dip_pin[4];
                            led_pin[1] <= sw_pin[1] ^ dip_pin[5];
                            led_pin[2] <= sw_pin[2] ^ dip_pin[6];
                            led_pin[3] <= sw_pin[3] ^ dip_pin[7];
                    end
    end

endmodule
```

2、在FPGA开发板上实现时序信号发生器电路

- 采用行为描述方式实现9.2小节的时序信号发生器电路。该电路的输入/CLR、START、STOP；输出为CP、T1、T2、T3、T4。

- 输入/CLR为开发板上最左边的拨动开关，输入START为开发板上的S4按键，输入STOP为开发板上的S1按键，输出CP为开发板上最左边的LED灯，输出T1、T2、T3、T4为开发板上最右边的4个LED灯。



EGO1开发板

- 验证步骤：
  - 运行程序后，首先置/CLR=0，此时，CP一直在闪烁，T1、T2、T3、T4不亮。
  - 置/CLR=1，按S4按键，此时，T1、T2、T3、T4灯轮流闪烁。
  - 按S1按键，此时，T1、T2、T3、T4灯不亮。
  - 再S4按键，此时，T1、T2、T3、T4灯轮流闪烁。

```
//采用行为描述方式实现9.2小节的时序信号发生器电路
//输入为开发板上最左边的拨动开关（/CLR）、按键S4（START）、按键S1（STOP）
//输出为开发板上最左边的LED灯（CP）、最右边的4个LED灯（T1、T2、T3、T4）

`timescale 1ns / 1ps

module example_9_2(
    input sys_clk_in,
    input sys_rst_n,
    input sw_pin[7:0],                                  //8个拨动开关
    input btn_1, btn_4,                                 //S1、S4按键
    output reg [15:0] led_pin                           //16个led灯
);

    reg [24 : 0] cnt_reg;
    reg [3 : 0] eight_reg;
    reg flag_start;
    reg btn_4_curr, btn_4_prev;
    reg btn_1_curr, btn_1_prev;
    reg [19:0] debounce_counter1 = 0;                   //用于btn_4的去抖动计数器
    reg [19:0] debounce_counter2 = 0;                   //用于btn_1的去抖动计数器
    parameter DEBOUNCE_PERIOD = 10000;                  //去抖动时间长度，根据时钟频率调整
```

```
always @(posedge sys_clk_in or negedge sys_rst_n)                    // 消除按键的抖动（去抖动）
begin
        if (!sys_rst_n)
                begin
                        btn_4_curr <= 1;
                        btn_4_prev <= 1;
                        debounce_counter1 <= 0;
                        btn_1_curr <= 1;
                        btn_1_prev <= 1;
                        debounce_counter2 <= 0;
                end
        else
                begin

                        btn_4_prev <= btn_4_curr;
                        if(btn_4==btn_4_curr)
                                begin
                                        btn_4_curr <= btn_4;
                                end
                        else if (btn_4 != btn_4_prev)
                                begin
                                        if(debounce_counter1 != DEBOUNCE_PERIOD - 1)
                                                begin
                                                        debounce_counter1 <= debounce_counter1 + 1;
                                                end
                                        if( debounce_counter1 == DEBOUNCE_PERIOD - 1)
                                                begin
                                                        debounce_counter1 <= 0;
                                                btn_4_curr <= btn_4;
                                                end
                                end
                         btn_1_prev <= btn_1_curr;
                         if(btn_1 == btn_1_curr)
                                begin
                                        btn_1_curr <= btn_1;
                                end
                        else if (btn_1 != btn_1_prev)
                        begin
                                 if(debounce_counter2 != DEBOUNCE_PERIOD - 1)
                                        begin
                                                debounce_counter2 <= debounce_counter2 + 1;
                                        end
                                        if( debounce_counter2 == DEBOUNCE_PERIOD - 1)
                                                begin
                                                        debounce_counter2 <= 0;
                                                        btn_1_curr <= btn_1;
```

消除2个按键的抖动

```
always@(posedge sys_clk_in)                    //根据sw_pin[0]拨动开关、S4按键、S1按键，确定LED灯是否闪烁
begin
   if (!sys_rst_n)
      flag_start <= 0;
   else
      begin
         if(sw_pin[0] == 1)
           begin
            if(!btn_4_curr && btn_4_prev)
                     flag_start <= 1;
            if(!btn_1_curr && btn_1_prev)
                     flag_start <= 0;
          end
         else
                     flag_start <= 0;
      end
end

always @ (posedge sys_clk_in)                  //延时计数
begin
   if (!sys_rst_n)
      cnt_reg <= 0;
   else
      cnt_reg <= cnt_reg + 1;
end

always @ (posedge sys_clk_in)        //eight_reg计数：0-7        最左边的LED不停地闪烁
begin
   if (!sys_rst_n)
      eight_reg <= 4'b0000;
   else if (cnt_reg == 25'h1ffffff)      //调整 25'h1ffffff 这个值，可以改变LED灯的闪烁时间，值小则闪烁快，值大则闪烁慢
   begin
      if (eight_reg == 4'b1000)
         eight_reg <= 4'b0000;
      else
         eight_reg <= eight_reg + 1;
      led_pin[0] <= ~led_pin[0];
   end
end
```

根据sw_pin[0]拨动开关、S4按键、S1按键，确定LED灯是否闪烁

延时计数

计数：0-7

最左边的LED灯不停闪烁

```
always @ (posedge sys_clk_in)        //4个LED轮流亮
begin
 if(flag_start == 1)
  begin
    if (eight_reg == 0)
       begin
          led_pin[4] <= 1;
          led_pin[5] <= 0;
          led_pin[6] <= 0;
          led_pin[7] <= 0;
       end
    if (eight_reg == 2)
       begin
          led_pin[4] <= 0;
          led_pin[5] <= 1;
          led_pin[6] <= 0;
          led_pin[7] <= 0;
       end
    if (eight_reg == 4)
       begin
          led_pin[4] <= 0;
          led_pin[5] <= 0;
          led_pin[6] <= 1;
          led_pin[7] <= 0;
       end
    if (eight_reg == 6)
       begin
          led_pin[4] <= 0;
          led_pin[5] <= 0;
          led_pin[6] <= 0;
          led_pin[7] <= 1;
       end
  end
 else
  begin
          led_pin[4] <= 0;
          led_pin[5] <= 0;
          led_pin[6] <= 0;
          led_pin[7] <= 0;
  end
 end

endmodule
```

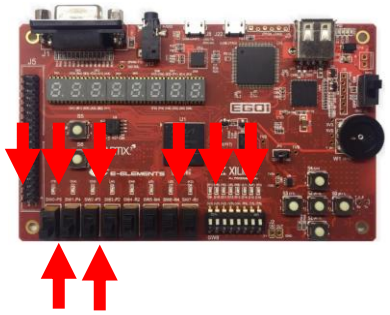
4个LED灯轮流闪烁

3、在FPGA开发板上实现汽车尾灯控制器电路

- 采用行为描述方式实现9.4小节的汽车尾灯控制器电路。该电路的输入K1、K0；输出为L1、L2、L3、R1、R2、R3。

- 输入K1、K0为开发板上最左边的2个拨动开关，输出L1、L2、L3为开发板上最左边的3个LED灯，输出R1、R2、R3为开发板上最右边的3个LED灯。

- 验证步骤：
  - 运行程序后，首先置K1=0、K0=0，此时，6个灯都不亮。
  - 置K1=0、K0=1，此时，左边3个灯不亮，右边3个灯从左到右轮流闪烁。
  - 置K1=1、K0=0，此时，右边3个灯不亮，左边3个灯从右到左轮流闪烁。
  - 置K1=1、K0=1，此时，6个灯一起闪烁。



EGO1开发板

example_9_4_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现9.4小节的汽车尾灯控制器电路
//输入为开发板上最左边的2个拨动开关（K1、K0）
//输出为开发板上最左边的3个LED灯（L1、L2、L3）、最右边的3个LED灯（R1、R2、R3）

`timescale 1ns / 1ps

module example_9_4(
    input sys_clk_in,
    input sys_rst_n,
    input sw_pin[7:0],                                  //8个拨动开关
    output reg [15:0] led_pin                           //16个led灯
);

    reg [24 : 0] cnt_reg;
    reg [2 : 0] six_reg;

    always @ (posedge sys_clk_in)                       //延时计数
    begin
        if (!sys_rst_n)
            cnt_reg <= 0;
        else
            cnt_reg <= cnt_reg + 1;
    end

    always @ (posedge sys_clk_in)          //six_reg计数：0-5
    begin
        if (!sys_rst_n)
            six_reg <= 3'b000;
        else if (cnt_reg == 25'h1ffffff)       //调整 25'h1ffffff 这个值，可以改变LED灯的闪烁时间，值小则闪烁快，值大则闪烁慢
        begin
            if (six_reg == 3'b110)
                six_reg <= 3'b000;
            else
                six_reg <= six_reg + 1;
        end
    end
```

延时计数

计数：0-5

```
always @ (posedge sys_clk_in)          //6个LED轮流亮
begin
   if(sw_pin[0] == 0 && sw_pin[1] ==0)
      begin
         led_pin[0] <= 0;
         led_pin[1] <= 0;
         led_pin[2] <= 0;
         led_pin[5] <= 0;
         led_pin[6] <= 0;
         led_pin[7] <= 0;
      end
   if (sw_pin[0] == 0 && sw_pin[1] ==1)
      begin
       if(six_reg == 0)
       begin
         led_pin[0] <= 0;
         led_pin[1] <= 0;
         led_pin[2] <= 0;
         led_pin[5] <= 1;
         led_pin[6] <= 0;
         led_pin[7] <= 0;
       end
       if(six_reg == 2)
       begin
         led_pin[0] <= 0;
         led_pin[1] <= 0;
         led_pin[2] <= 0;
         led_pin[5] <= 0;
         led_pin[6] <= 1;
         led_pin[7] <= 0;
       end
       if(six_reg == 4)
       begin
         led_pin[0] <= 0;
         led_pin[1] <= 0;
         led_pin[2] <= 0;
         led_pin[5] <= 0;
         led_pin[6] <= 0;
         led_pin[7] <= 1;
       end
      end
```

6个灯全部不亮

左边3个灯不亮

右边3个灯轮流亮

```
   if (sw_pin[0] == 1 && sw_pin[1] ==0)
      begin
       if(six_reg == 0)
       begin
         led_pin[0] <= 0;
         led_pin[1] <= 0;
         led_pin[2] <= 1;
         led_pin[5] <= 0;
         led_pin[6] <= 0;
         led_pin[7] <= 0;
       end
       if(six_reg == 2)
       begin
         led_pin[0] <= 0;
         led_pin[1] <= 1;
         led_pin[2] <= 0;
         led_pin[5] <= 0;
         led_pin[6] <= 0;
         led_pin[7] <= 0;
       end
       if(six_reg == 4)
       begin
         led_pin[0] <= 1;
         led_pin[1] <= 0;
         led_pin[2] <= 0;
         led_pin[5] <= 0;
         led_pin[6] <= 0;
         led_pin[7] <= 0;
       end
      end
```

左边3个灯轮流亮

右边3个灯不亮

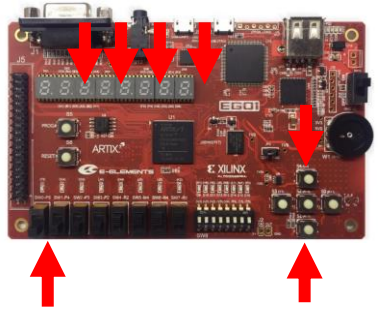
```
if(sw_pin[0] == 1 && sw_pin[1] ==1)
begin
 if(six_reg == 0)
 begin
 led_pin[0] = 0;
 led_pin[1] = 0;
 led_pin[2] = 0;
 led_pin[5] = 0;
 led_pin[6] = 0;
 led_pin[7] = 0;
 end
 if(six_reg == 1)
 begin
 led_pin[0] = 1;
 led_pin[1] = 1;
 led_pin[2] = 1;
 led_pin[5] = 1;
 led_pin[6] = 1;
 led_pin[7] = 1;
 end
 if(six_reg == 2)
 begin
 led_pin[0] = 0;
 led_pin[1] = 0;
 led_pin[2] = 0;
 led_pin[5] = 0;
 led_pin[6] = 0;
 led_pin[7] = 0;
 end
 if(six_reg == 3)
 begin
 led_pin[0] = 1;
 led_pin[1] = 1;
 led_pin[2] = 1;
 led_pin[5] = 1;
 led_pin[6] = 1;
 led_pin[7] = 1;
 end
```

6个灯同时闪烁

```
 if(six_reg == 3)
 begin
 led_pin[0] = 1;
 led_pin[1] = 1;
 led_pin[2] = 1;
 led_pin[5] = 1;
 led_pin[6] = 1;
 led_pin[7] = 1;
 end
 if(six_reg == 4)
 begin
 led_pin[0] = 0;
 led_pin[1] = 0;
 led_pin[2] = 0;
 led_pin[5] = 0;
 led_pin[6] = 0;
 led_pin[7] = 0;
 end
 if(six_reg == 5)
 begin
 led_pin[0] = 1;
 led_pin[1] = 1;
 led_pin[2] = 1;
 led_pin[5] = 1;
 led_pin[6] = 1;
 led_pin[7] = 1;
 end
 end
 end

endmodule
```

4、挑战实验：在FPGA开发板上实现弹道计时器电路

- 采用行为描述方式实现9.3小节的弹道计时器电路。该电路的输入为“清0”、ST、SP，输出为4个数码管显示值。



EGO1开发板

- 输入“清0”为开发板上最左边的1个拨动开关，输入ST为开发板上的S4按键，输入SP为开发板上的S1按键，输出为开发板上最右边的4个数码管。

- 实验工程文件夹命名：example_9_3_EGO1
- 设计文件命名：example_9_3_EGO1.v

- 程序执行后，置“清0”=1（最左边的拨动开关），按ST（S4按键）后，数码管开始计数，按SP（S1按键）数码管停止计数。置“清0”=1，按ST后，数码管清0，且不计数。

实验要求

- 1、在Logisim上完成验证实验，将实验过程中的主要结果通过截屏（或拍照）的方式，拷贝到实验报告中，并给予适当的文字说明。
- 2、在FPGA开发板上完成验证实验，将实验过程中的主要结果通过截屏（或拍照）的方式，拷贝到实验报告中，并给予适当的文字说明。
- 3、实验报告命名为：学号+姓名+第7次实验报告.docx。
- 4、完成挑战实验（在FPGA开发板上实现弹道计时器电路）的同学，本次实验程序加10分。

Thanks