《计算机组成原理》

课程实验指导书

《计算机组成原理》课程组　　编著

上海海洋大学海洋智能信息实验教学示范中心

目　　录

实验一、Logisim 基础（2 学时）

一、实验目的

熟悉 Logisim 基本功能，常用操作、熟悉 Logisim 基本组件库、掌握 Logisim 自动生成电路的方法。

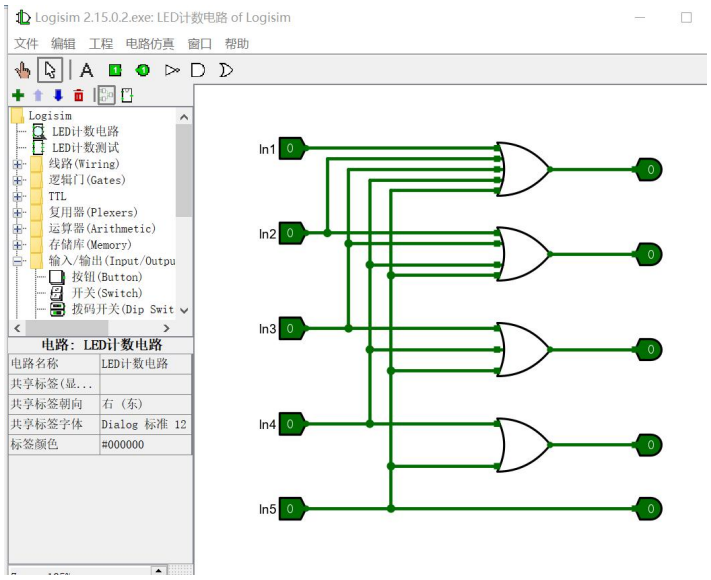
二、实验内容

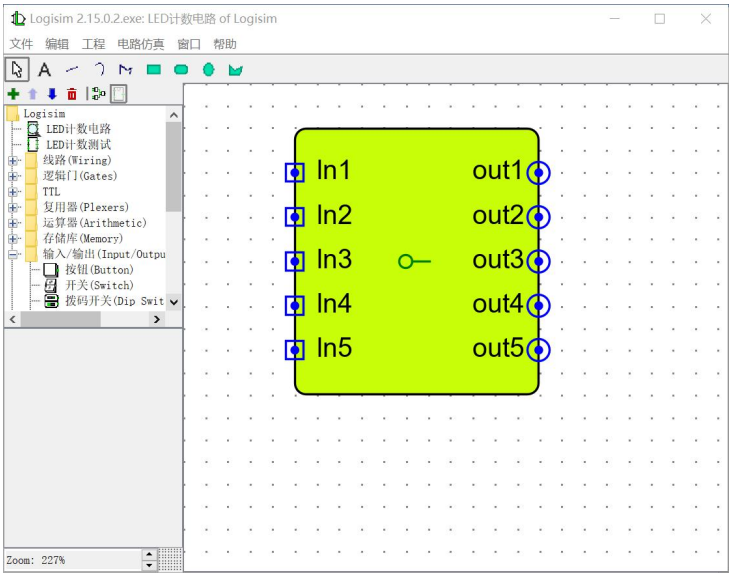
- 设计 LED 计数电路
- 利用真值表建立输入按键编码器(真值表→表达式→自动生成电路)
- 7 段数码管显示驱动。

（具体实验内容见实验视频指导）

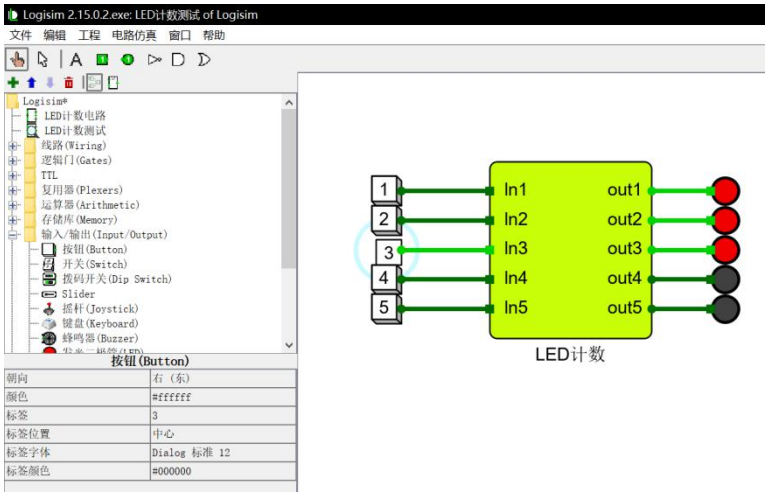
三、实验步骤

(1) LED 计数电路

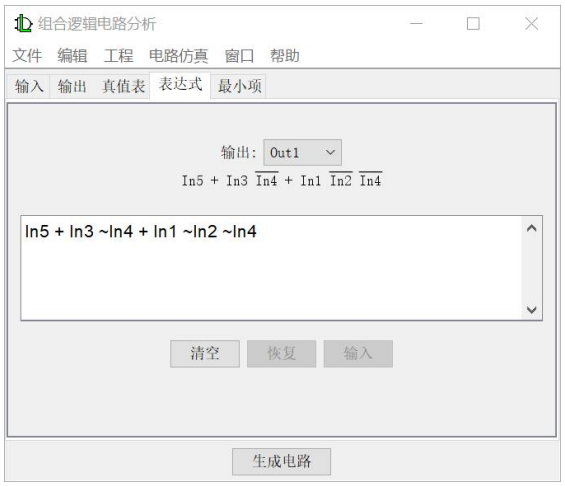


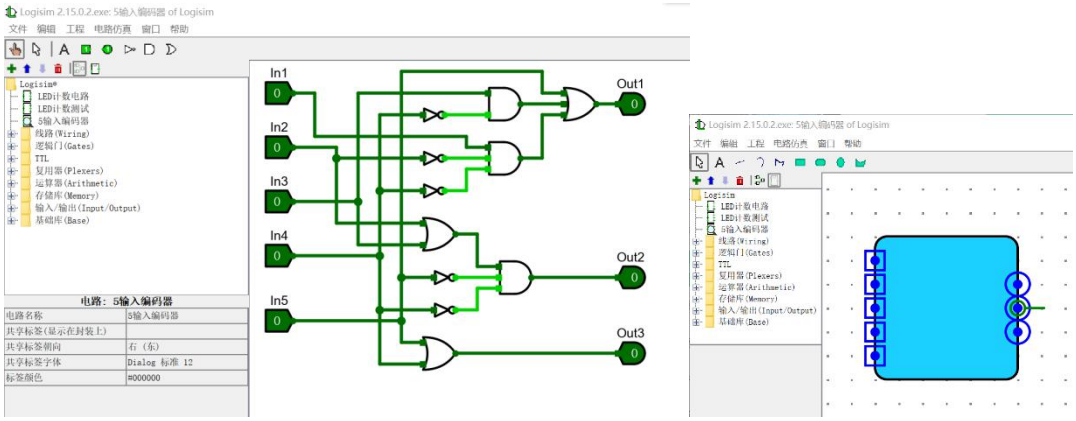


LED 计数测试

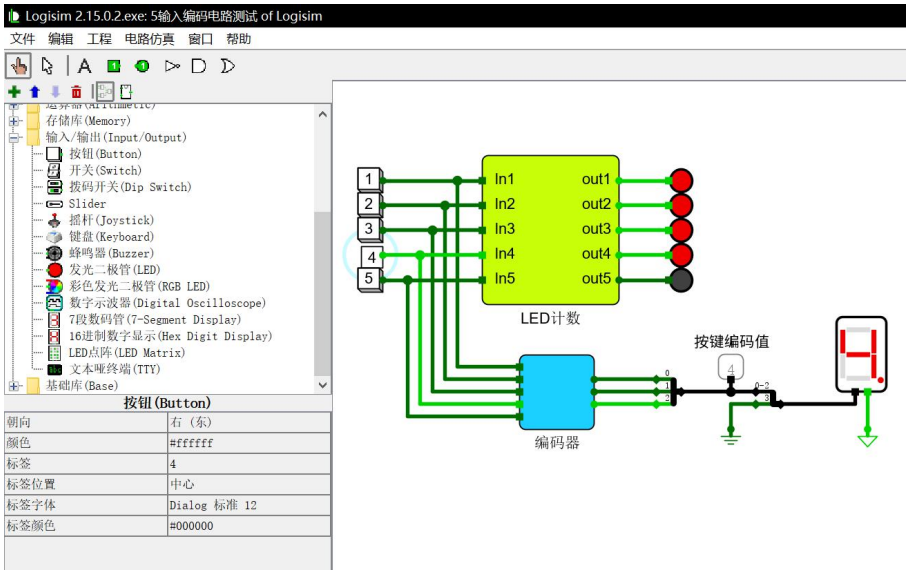


(2) 利用真值表建立五输入按键编码器

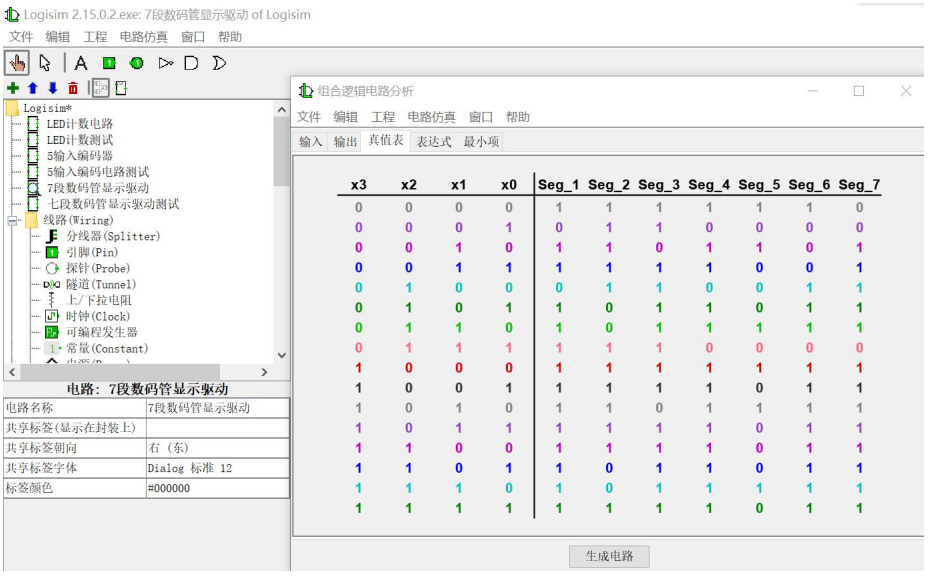


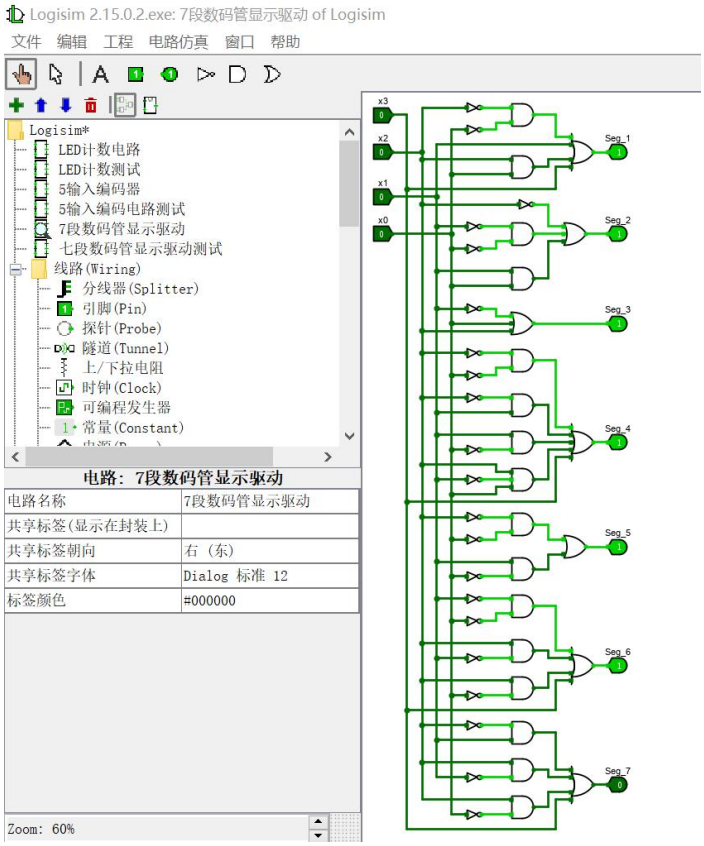


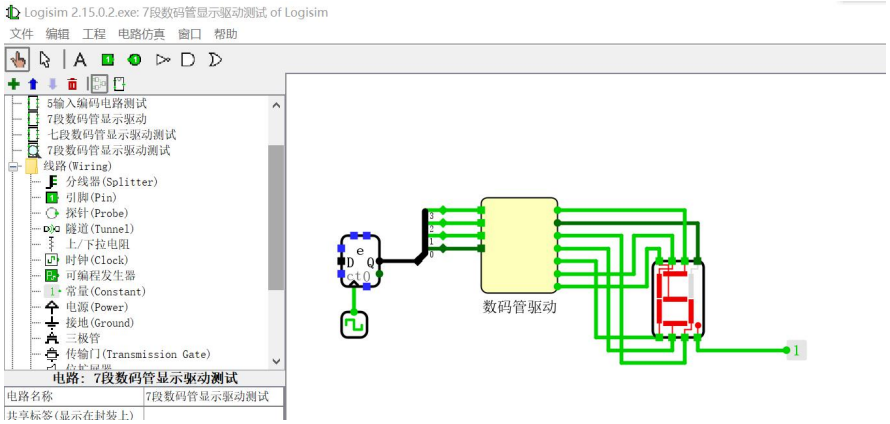
输入编码电路测试

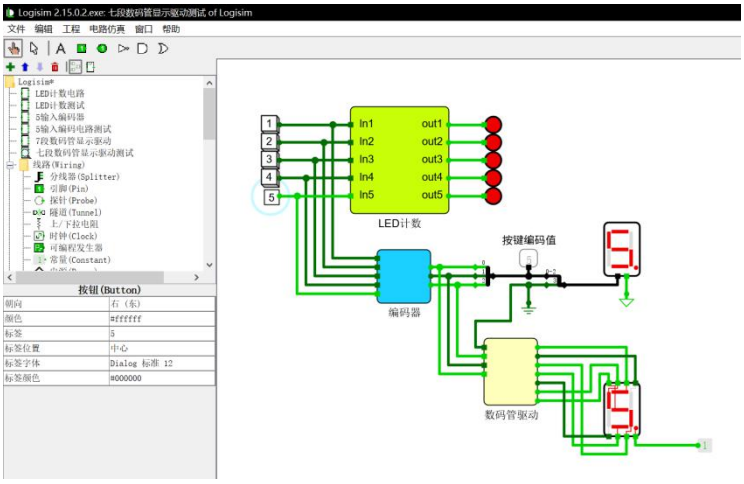


(3) 7 段数码管显示驱动




Logisim 2.15.0.2.exe: 7段数码管显示驱动 of Logisim
文件 编辑 工程 电路仿真 窗口 帮助
Logisim*
LED计数电路
LED计数测试
5输入编码器
5输入编码电路测试
7段数码管显示驱动
七段数码管显示驱动测试
线路(Wiring)
分线器(Splitter)
引脚(Pin)
探针(Probe)
隧道(Tunnel)
上/下拉电阻
时钟(Clock)
可编程发生器
常量(Constant)
电路：7段数码管显示驱动
电路名称 7段数码管显示驱动
共享标签(显示在封装上)
共享标签朝向 右 (东)
共享标签字体 Dialog 标准 12
标签颜色 #000000
Zoom: 60%
x3
x2
x1
x0
Seg_1
Seg_2
Seg_3
Seg_4
Seg_5
Seg_6
Seg_7


Logisim 2.15.0.2.exe: 7段数码管显示驱动测试 of Logisim
文件 编辑 工程 电路仿真 窗口 帮助
5输入编码电路测试
7段数码管显示驱动
七段数码管显示驱动测试
7段数码管显示驱动测试
线路(Wiring)
分线器(Splitter)
引脚(Pin)
探针(Probe)
隧道(Tunnel)
上/下拉电阻
时钟(Clock)
可编程发生器
常量(Constant)
电源(Power)
接地(Ground)
三极管
传输门(Transmission Gate)
电路：7段数码管显示驱动测试
电路名称 7段数码管显示驱动测试
共享标签(显示在封装上)
数码管驱动


Logisim 2.15.0.2.exe: 七段数码管显示驱动测试 of Logisim
文件 编辑 工程 电路仿真 窗口 帮助
按钮(Button)
朝向 右 (东)
颜色 #ffffff
标签 5
标签位置 中心
标签字体 Dialog 标准 12
标签颜色 #000000
In1
In2
In3
In4
In5
out1
out2
out3
out4
out5
LED计数
编码器
按键编码值
数码管驱动

实验二、数码管输出设计（2 学时）

一、实验目的

掌握 Logisim 中通过使用真值表、表达式等方式设计并实现数码管输出。

二、实验内容

对于 4 位二进制数，可以表示出 0~15 这 16 个数，在八段共阳极数码管上则依次显示 0，1，2，……，9，A，B，C，d，E，F（见图 1）。要求使用 Logisim 画出上述对应的电路。（具体实验内容见实验视频指导）

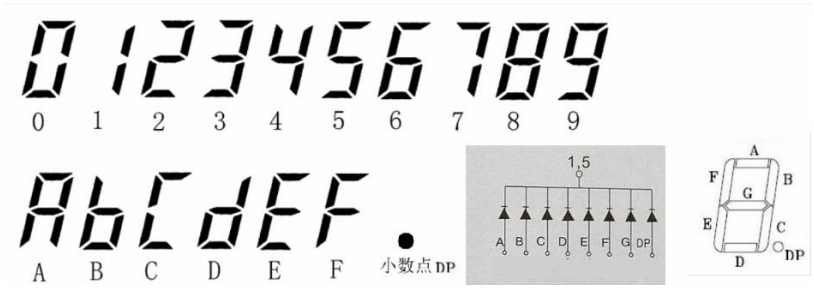


图 1 八段共阳极数码管显示图

三、实验步骤

1. 真值表

十进制	输入				输出							字形
	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	1	1	0	0	1	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0	2
3	0	0	1	1	0	0	0	0	1	1	0	3
4	0	1	0	0	1	0	0	1	1	0	0	4

5	0	1	0	1	0	1	0	0	1	0	0	5
6	0	1	1	0	0	1	0	0	0	0	0	6
7	0	1	1	1	0	0	0	1	1	1	1	7
8	1	0	0	0	0	0	0	0	0	0	0	8
9	1	0	0	1	0	0	0	0	1	0	0	9
10	1	0	1	0	0	0	0	1	0	0	0	A
11	1	0	1	1	1	1	0	0	0	0	0	B
12	1	1	0	0	0	1	1	0	0	0	1	C
13	1	1	0	1	1	0	0	0	0	1	0	d
14	1	1	1	0	0	1	1	0	0	0	0	E
15	1	1	1	1	0	1	1	1	0	0	0	F

2. 表达式（“/”表示“非”）

$a=x_1+x_2+x_3+x_4$

=/D3*/D2*/D1*D0+/D3*D2*/D1*/D0+D3*/D2*D1*D0+D3*D2*/D1*D0

$b=x_1+x_2+x_3+x_4+x_5+x_6$

=/D3*D2*/D1*D0+/D3*D2*D1*/D0+D3*/D2*D1*D0+D3*D2*/D1*/D0+D3*D2*D1*/D0+D3*D2*D1*D0

$c=x_1+x_2+x_3+x_4$

=/D3*/D2*D1*/D0+D3*D2*/D1*/D0+D3*D2*D1*/D0+D3*D2*D1*D0

$d=x_1+x_2+x_3+x_4$

=/D3*/D2*/D1*D0+/D3*D2*/D1*/D0+/D3*D2*D1*D0+D3*/D2*D1*/D0+D3*D2*D1*D0

$e=x_1+x_2+x_3+x_4$

=/D3*/D2*/D1*D0+/D3*/D2*D1*D0+/D3*D2*/D1*/D0+/D3*D2*/D1*D0+/D3*D2*D1*D0+D3*/D2*/D1*D0

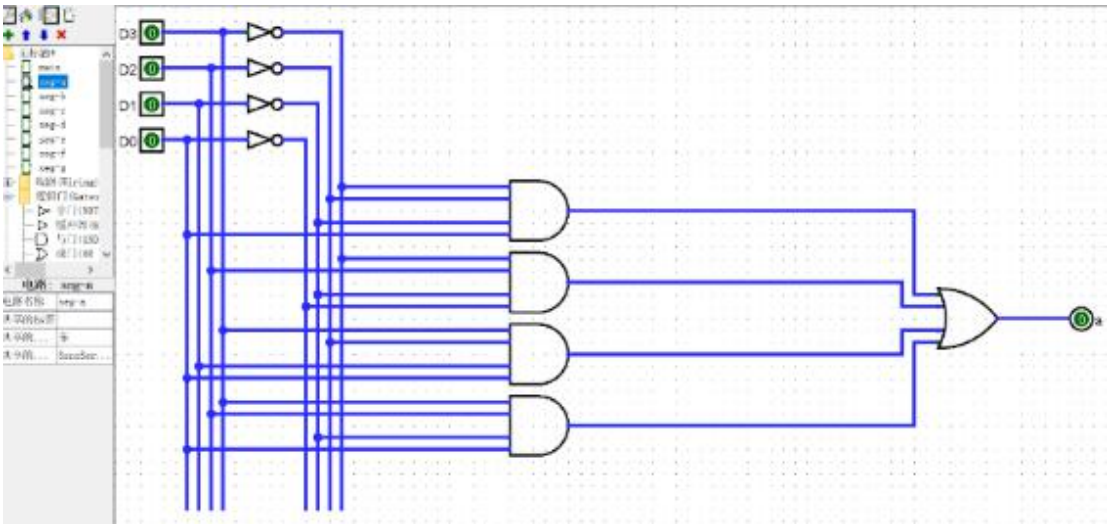
$f=x_1+x_2+x_3+x_4$

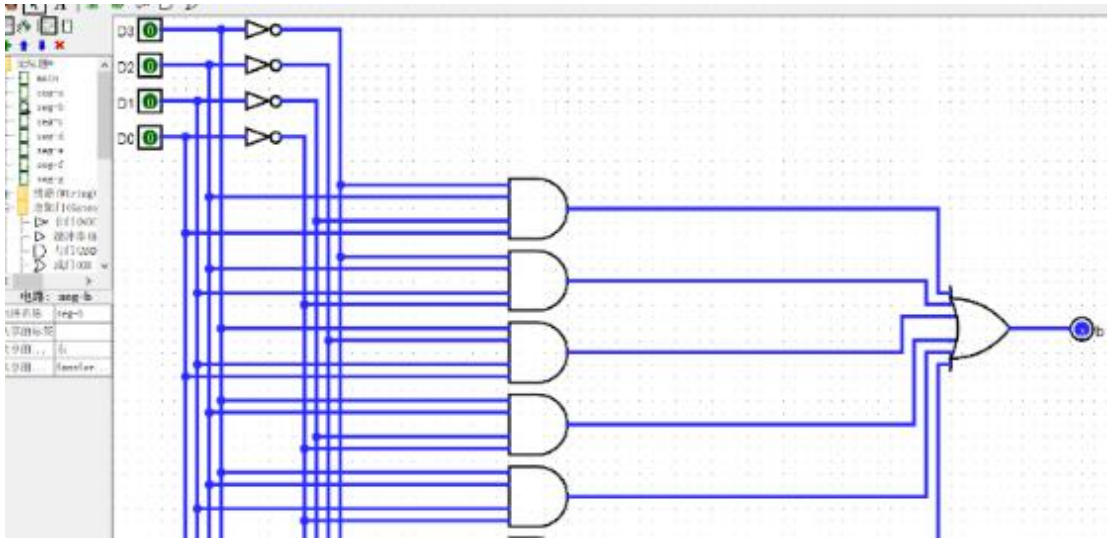
=/D3*/D2*/D1*D0+/D3*/D2*D1*/D0+/D3*/D2*D1*D0+/D3*D2*D1*D0+D3*D2*/D1*D0

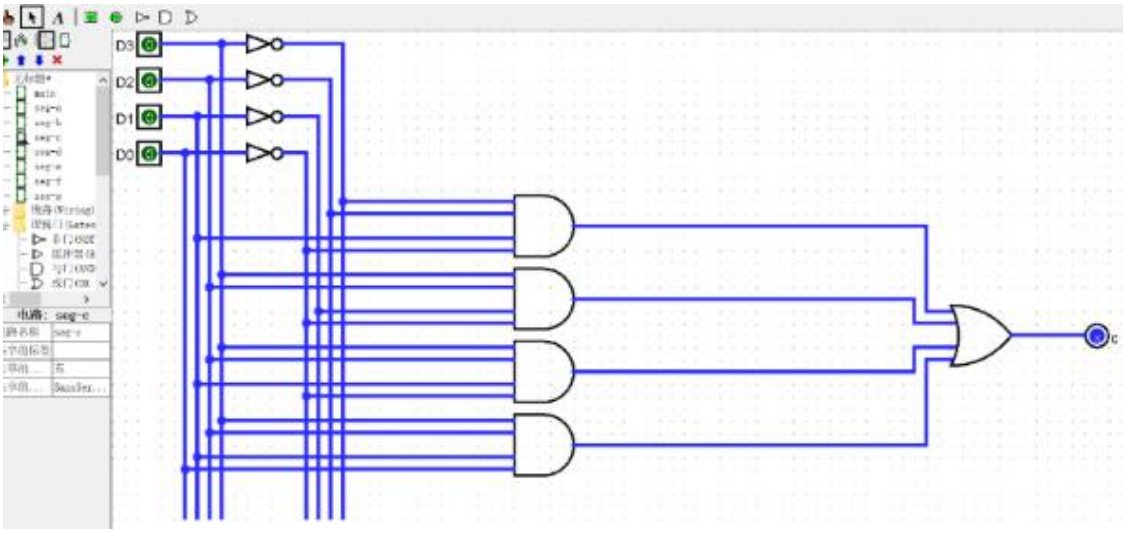
$g=x_1+x_2+x_3+x_4$

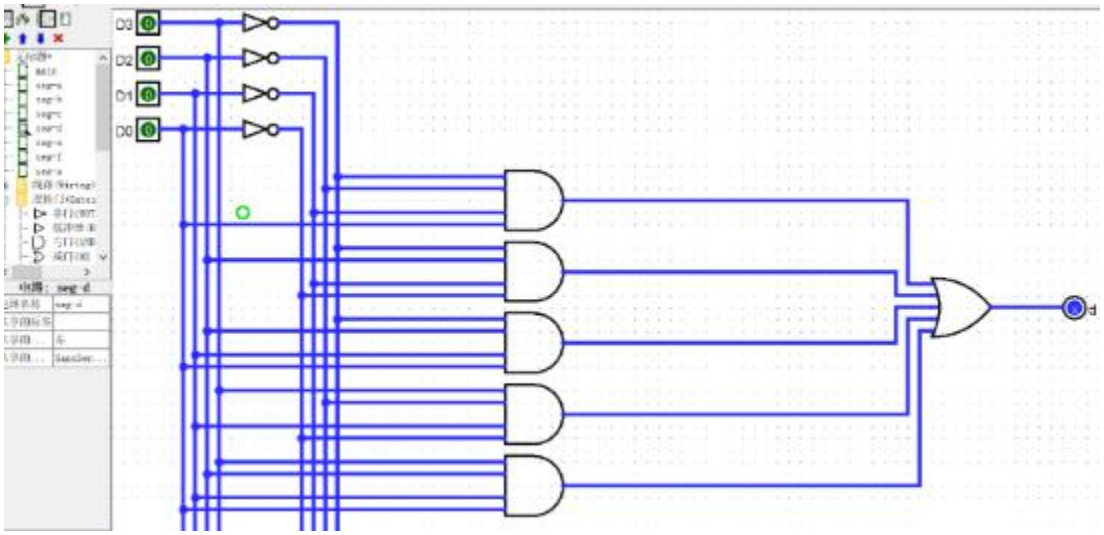
=/D3*/D2*/D1*/D0+/D3*/D2*/D1*D0+/D3*D2*D1*D0+D3*D2*/D1*/D0

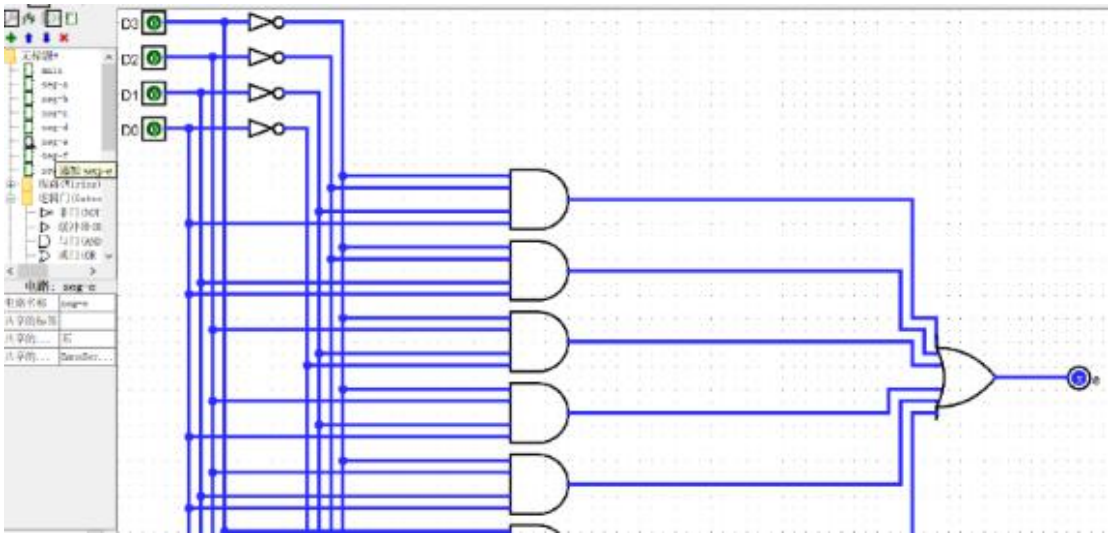
3. 使用 Logism 画出电路图

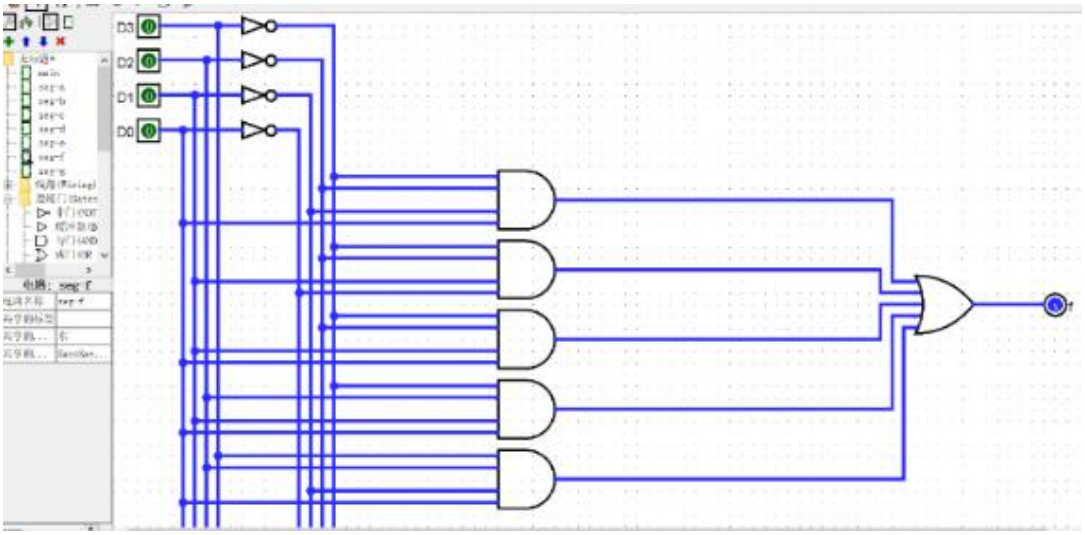


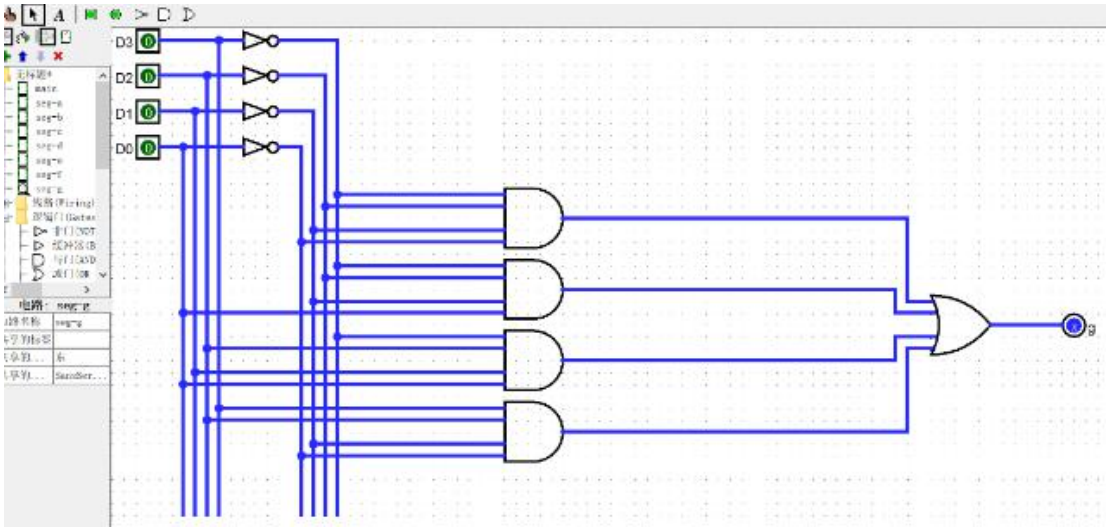




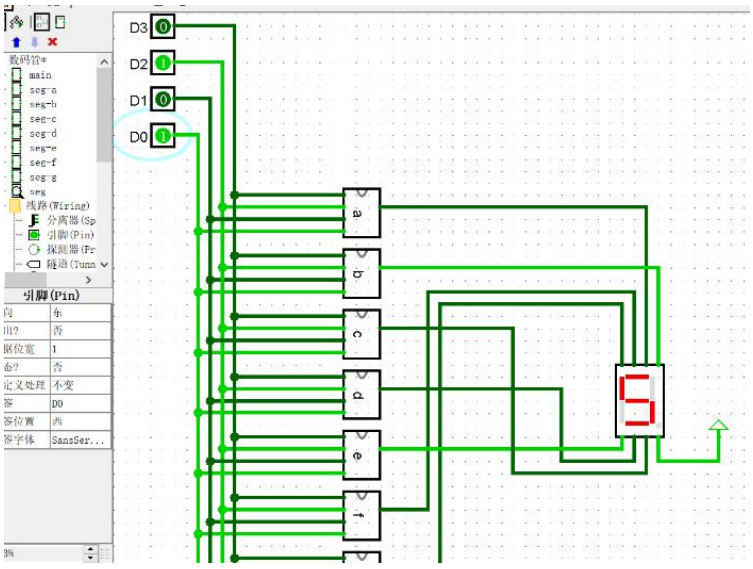

D3
D2
D1
D0

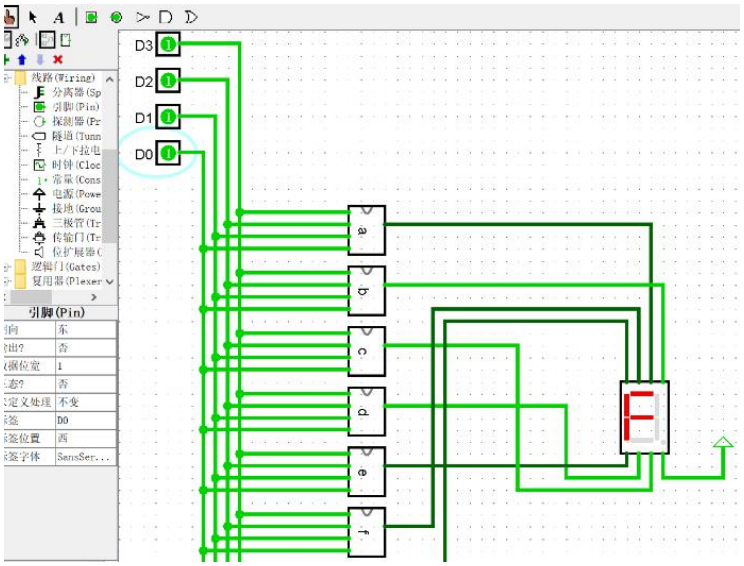

D3
D2
D1
D0

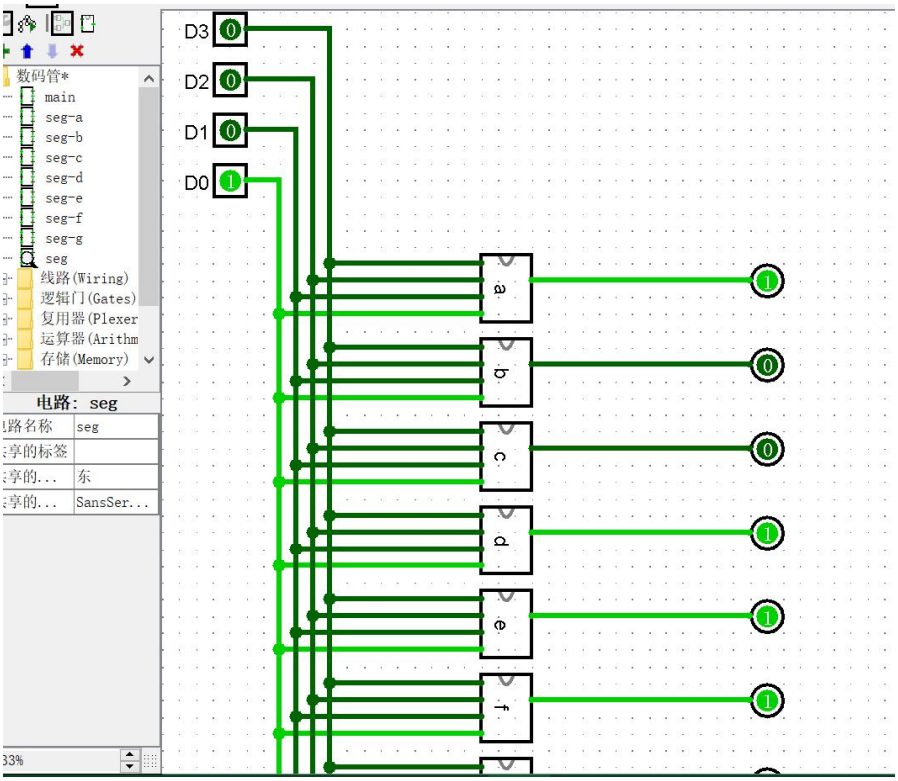

D3
D2
D1
D0



四、实验结果







实验通过 7 段数码管的真值表，得到 seg-a 到 seg-g 的表达式，根据表达式连出电路。之后分别将 7 段数码管封装，与 4 位输入和 7 段输出相连，做成一个封装。连线时要注意 7 段输出要和数码管的触点顺序相对应，要将数码管模块设置成低电平有效即可。

实验三、多端口输入设计（2 学时）

一、实验目的

在 Logisim 中设计并实现含两个端口的输入，并将其与数码管输出整合。

二、实验内容

（1）在 Logisim 中模拟键盘的数字输入，输入一个 0~15 的数字（通过按钮模拟实现，见图 1（a)），通过电路设计输出一个 4 位的二进制数（见图 1（b)）。

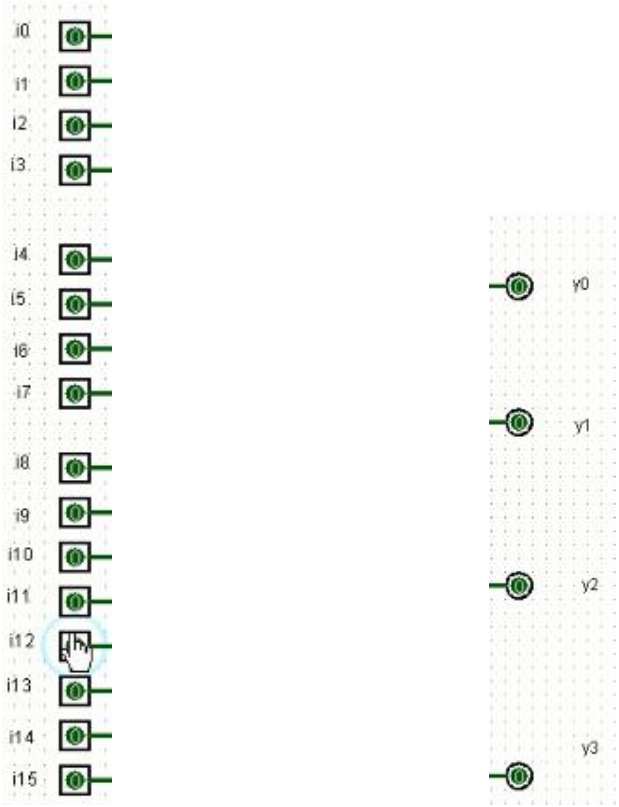


图 1（a）按钮模拟实现 0~15 的输入　（b）4 位的二进制数输出

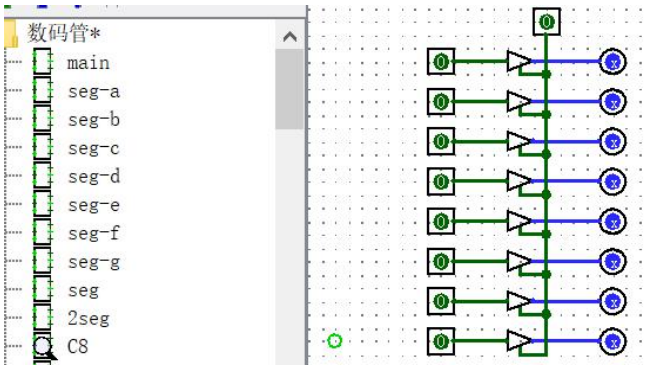
（2）在内容（1）的基础上再加一个 0~15 的数字输入，并添加添加三态门，设计输出一个 8 位的二进制数。

（3）实现输入接口第二个端口设计、状态端口的控制电路。并对多输入接口进行验证。

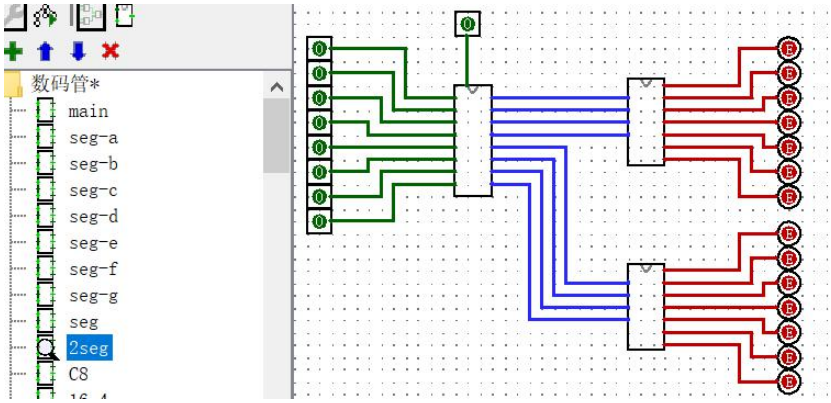
（详细实验内容见实验视频指导）

三、实验步骤

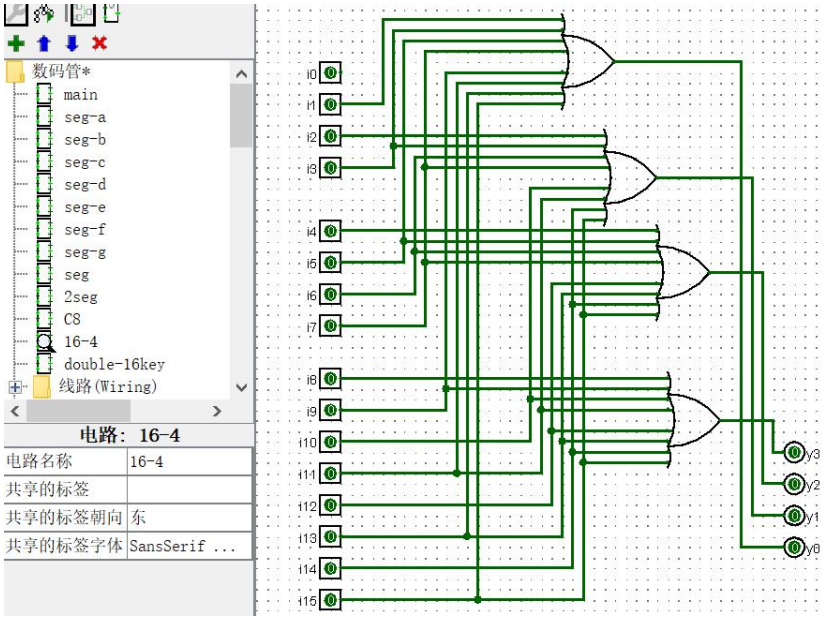
建立新电路：C8。



建立新电路：两路数码管 2seg。将 C8 放入 2seg 中。

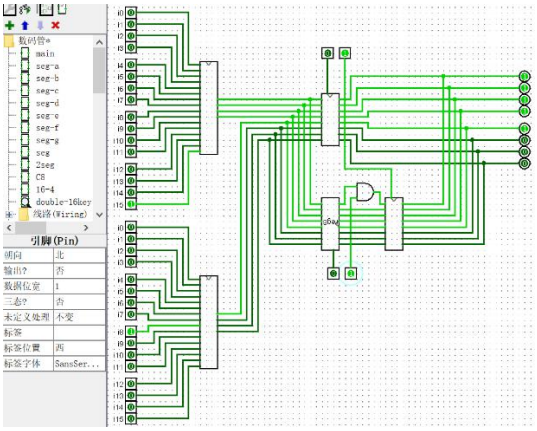


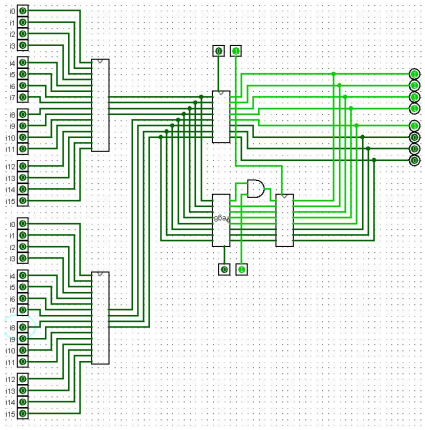
建立新电路：根据真值表，建立 16-4。



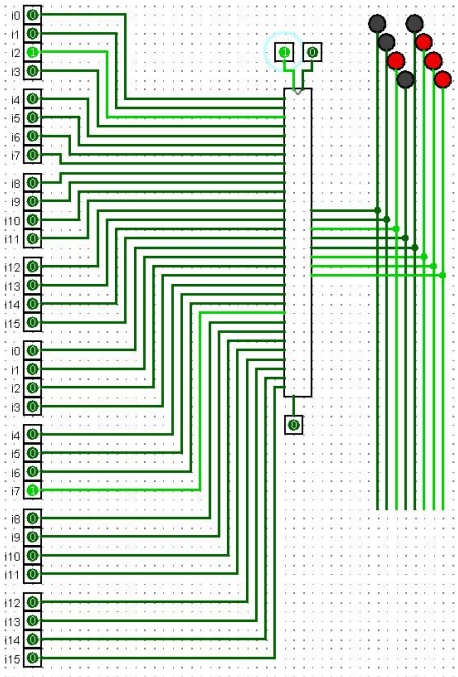
在 main 电路中，建立输出端。

建立新电路：double-16key：做了一个键盘接口，接口中包含两个端口，一个端口是真正的数据，一个端口是状态端口。

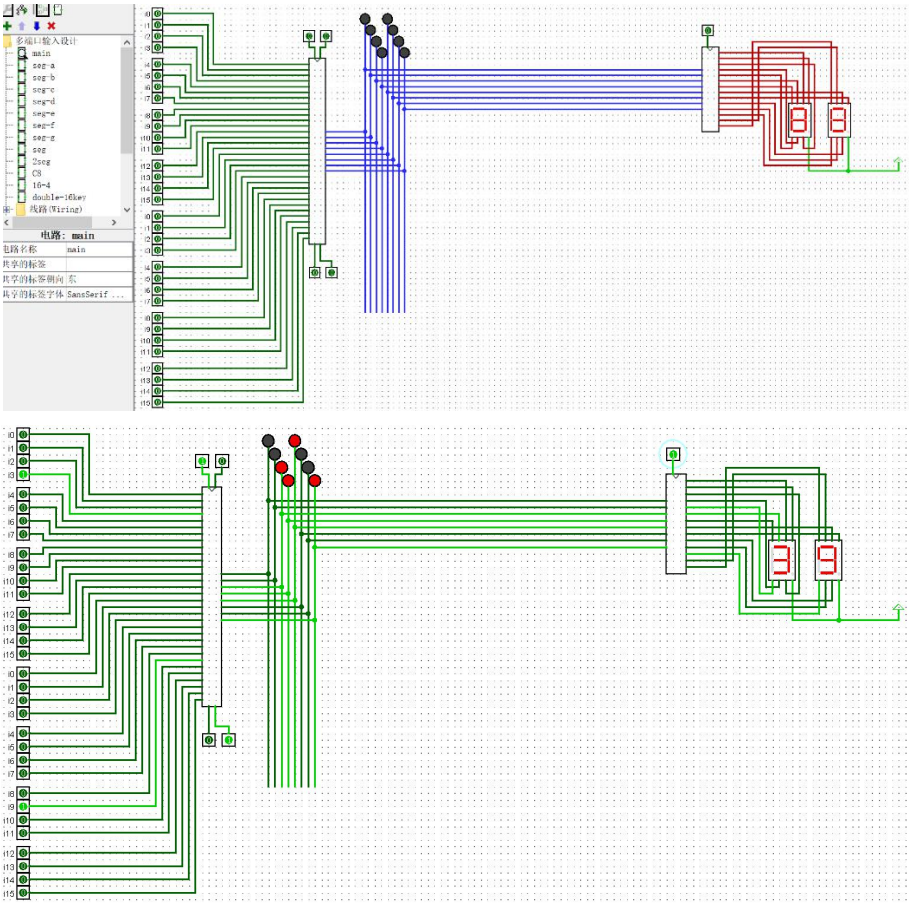




在 main 电路中，建立输入端，并验证。

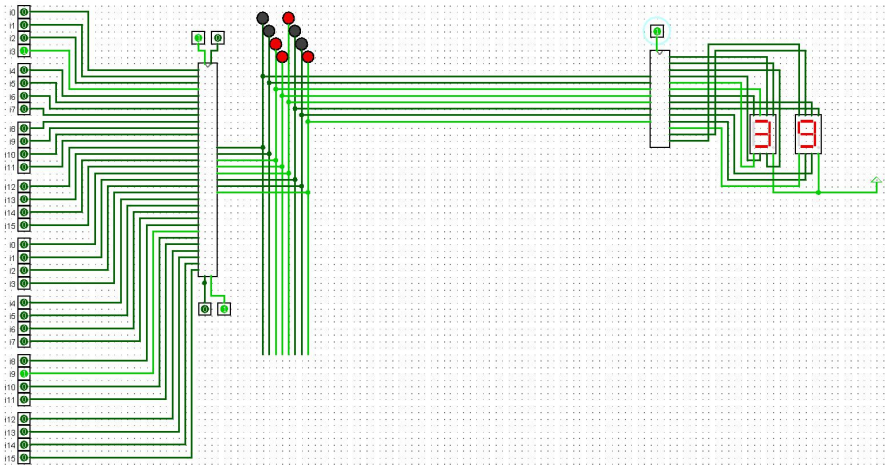


在 main 电路中将输入端和输出端连起来，并验证。



四、实验结果

在键盘中输入数据，改变 register 的电平。数据准备好时，在控制端输入“1”，控制输入端口 2 的位置输入“0”，控制输入端口 1 的位置输入“1”，显示数据。



实验四、运算器设计（4 学时）

一、实验目的

掌握 1 位半、全加器的实现逻辑，能在 Logisim 中实现多位串行进位加法器电路。掌握寄存器在 ALU 中的作用，实现运算器中的双向总线设计。

二、实验内容

（1）运算器（加法器）的设计原理。

（2）1 位半加器和全加器设计。

（3）串行进位加法器实现。

（4）寄存器中 ALU 中的使用

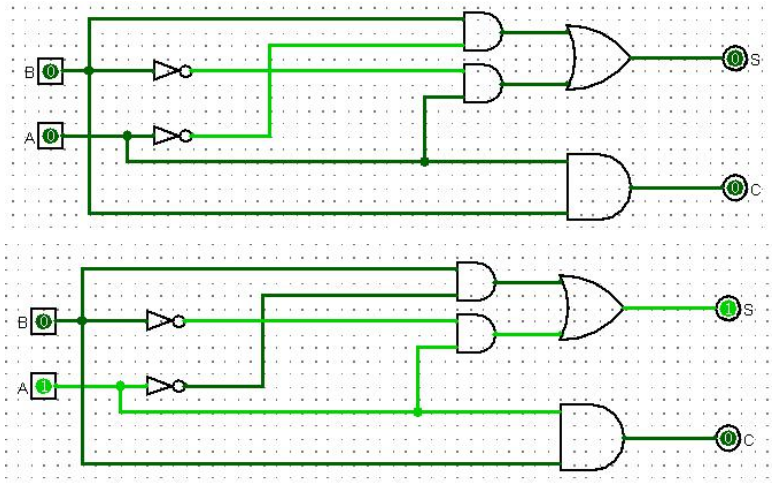
（5）运算器中的双向总线

（6）实现手摇式计算机实现运算器功能

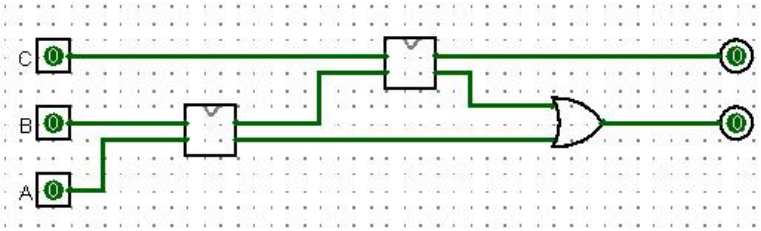
（具体实验内容见实验视频指导）

三、实验步骤

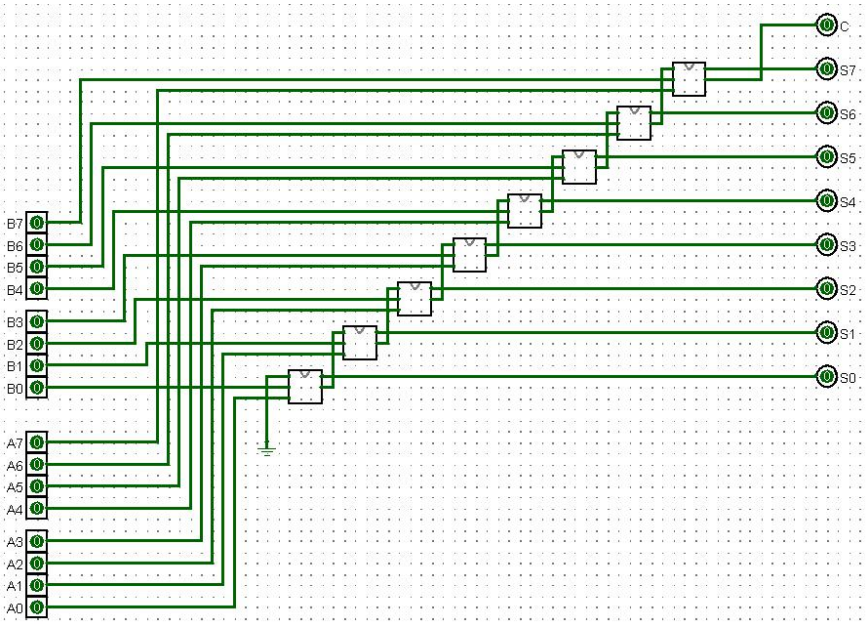
制作一个半加器 ha，根据真值表做出逻辑表达式，根据真值表绘制电路：两个输入 A、B，两个输出 S、C，加入与门和或门。并验证。

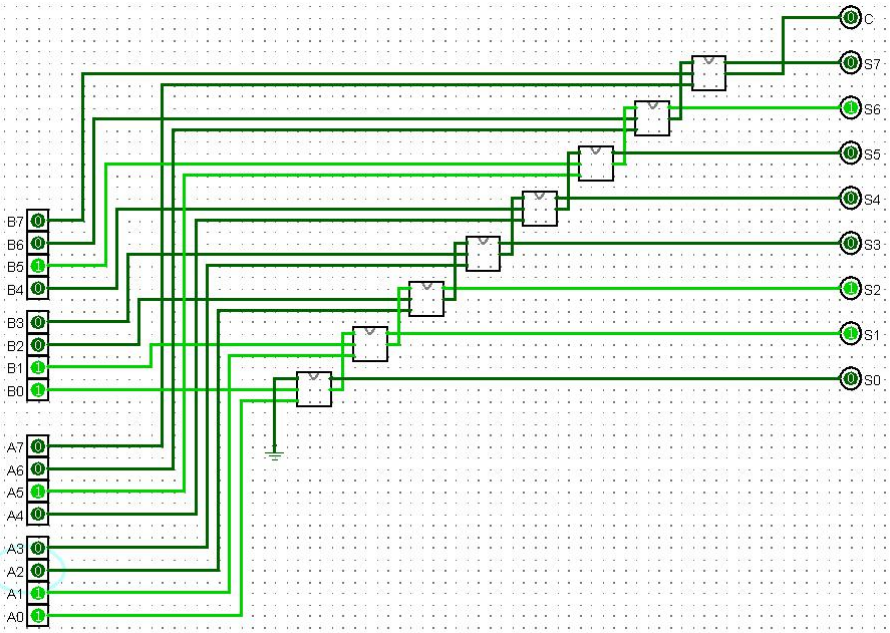


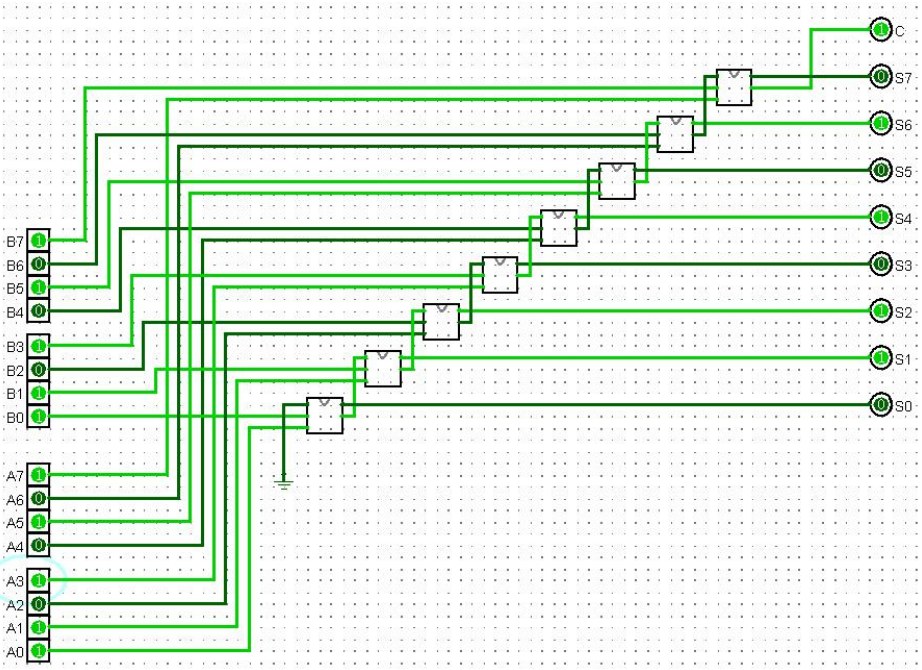
制作一个全加器 FA，将做好的半加器放入全加器中，两个半加器形成一个全加器：三个输入 A、B、C，两个输出。



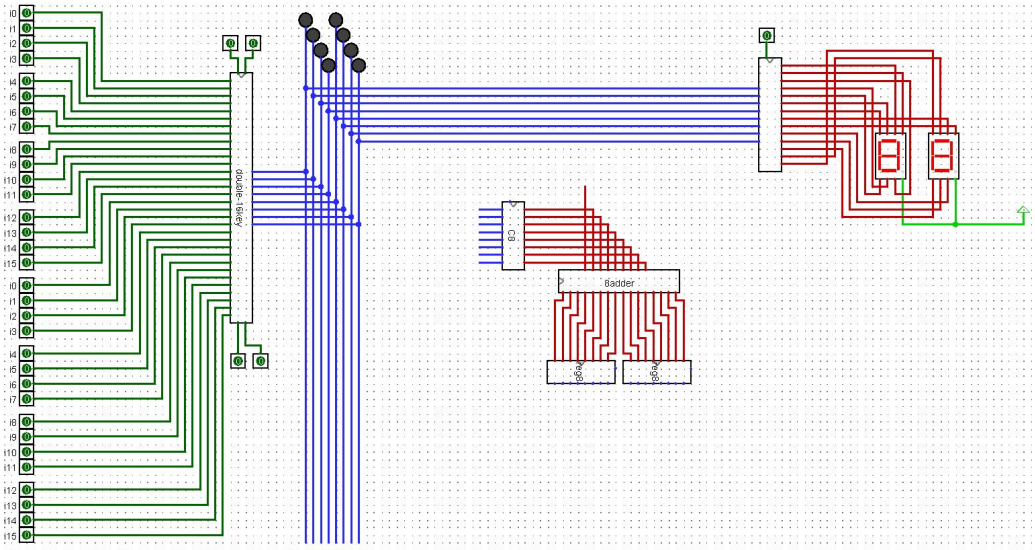
制作一个 8 位的串行加法器 adder，并验证。



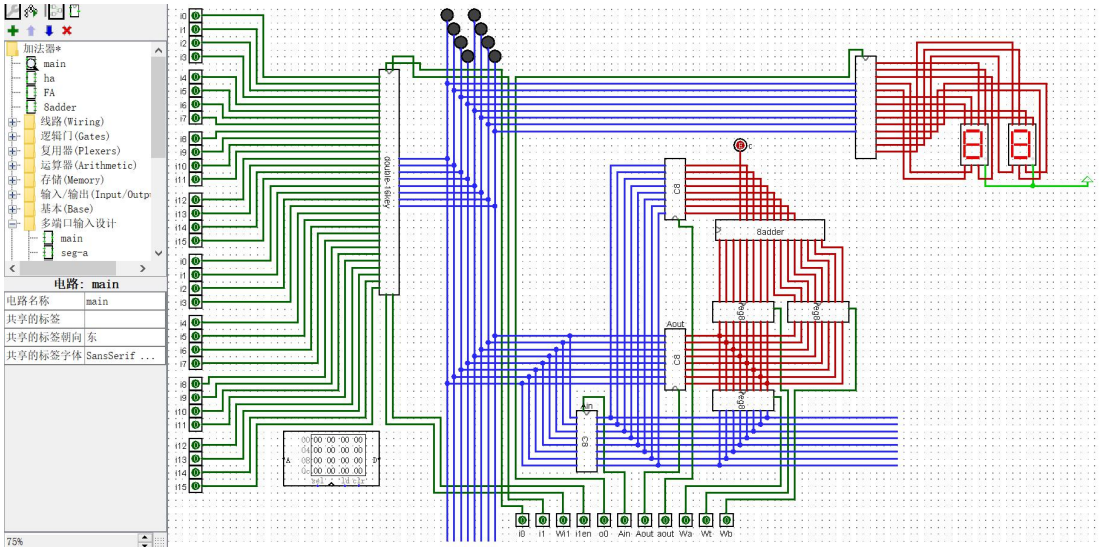


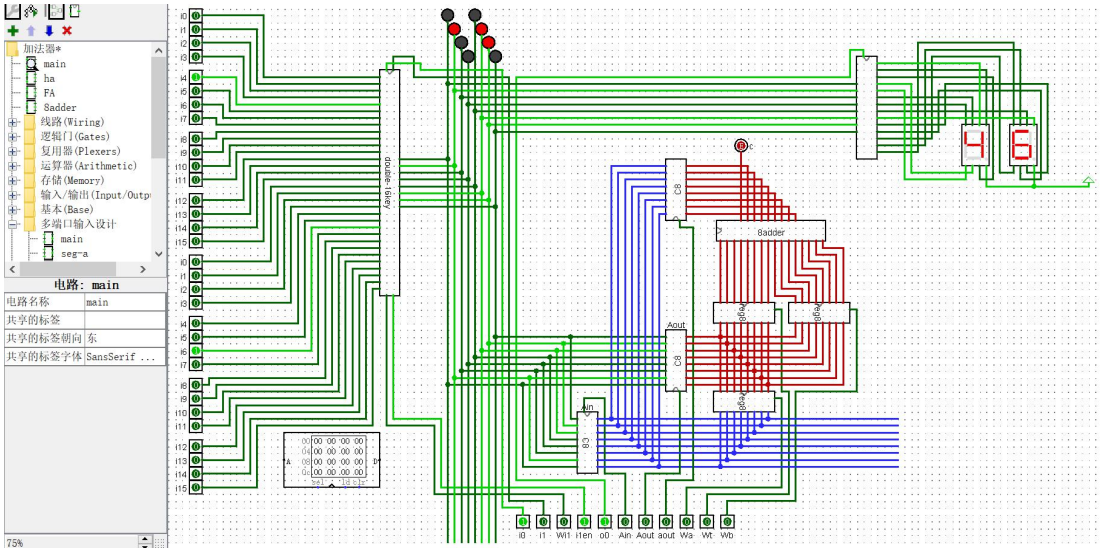


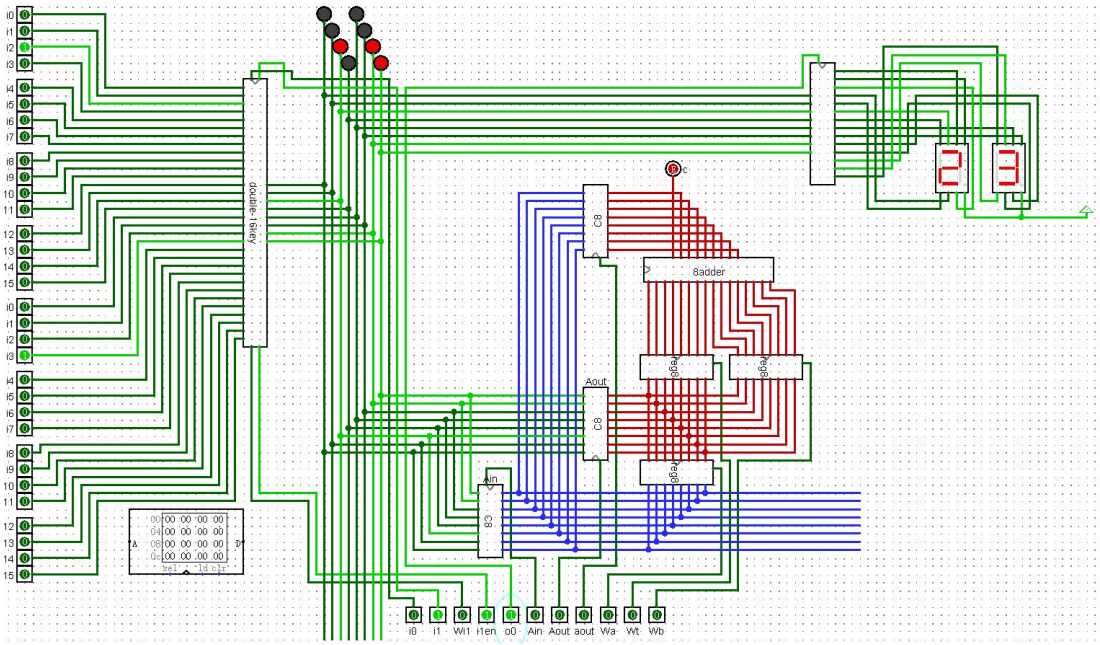
在主电路中加入做好的 8 位串行加法器。在输入端上加入两个寄存器 reg8，在输出端加入一个控制器 C8。

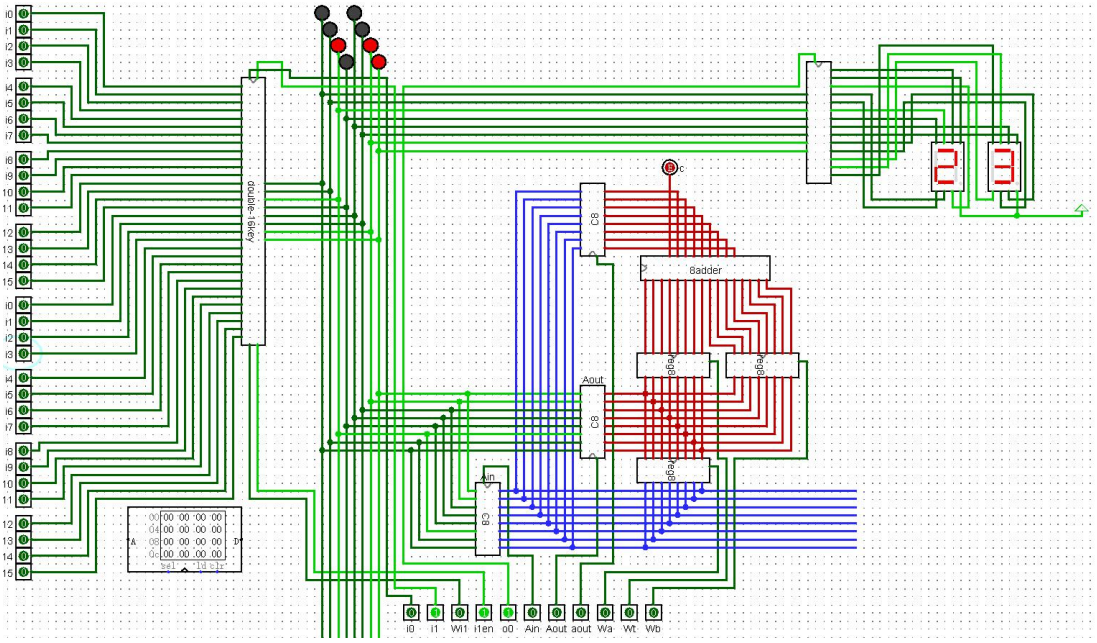


加入一对方向相反的控制器 C8，一个累加器 reg8，把两个暂存器和累加器分别与内部总线相连。此时运算器有三个寄存器，三个三态门。加入输入端。并验证。

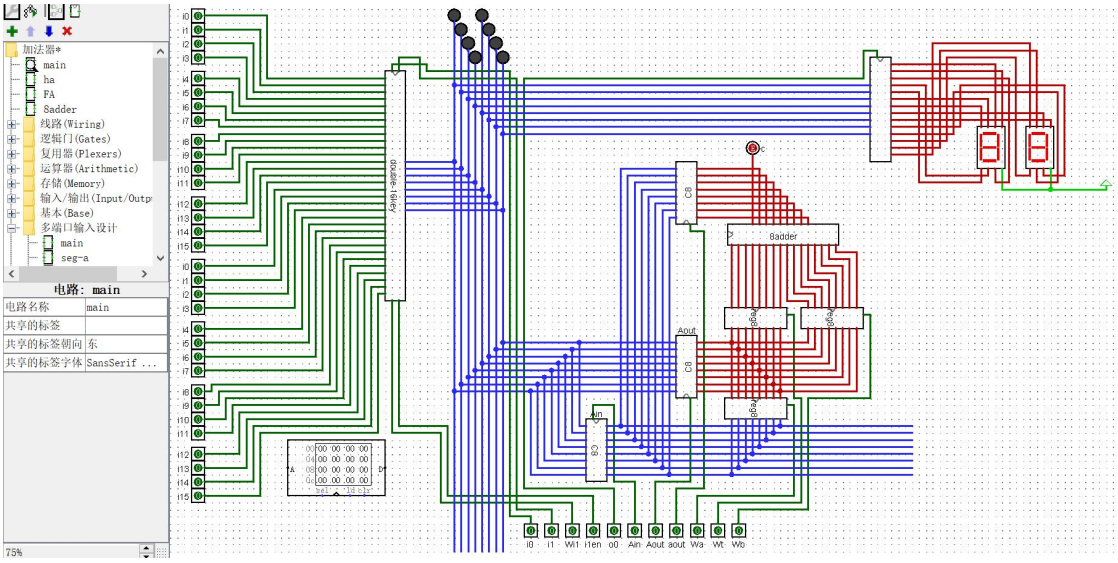

加法器*
main
ha
FA
8adder
线路(Wiring)
逻辑门(Gates)
复用器(Plexers)
运算器(Arithmetic)
存储(Memory)
输入/输出(Input/Outp
基本(Base)
多端口输入设计
main
seg-a
电路: main
电路名称 main
共享的标签
共享的标签朝向 东
共享的标签字体 SansSerif ...
75%
double-16key
C8
8adder
Aout
Ain
C8
seg8
seg8
seg8
8
8
i0 i1 Wi1 i1en o0 Ain Aout aout Wa Wt Wb


加法器*
main
ha
FA
8adder
线路(Wiring)
逻辑门(Gates)
复用器(Plexers)
运算器(Arithmetic)
存储(Memory)
输入/输出(Input/Outp
基本(Base)
多端口输入设计
main
seg-a
电路: main
电路名称 main
共享的标签
共享的标签朝向 东
共享的标签字体 SansSerif ...
75%
double-16key
C8
8adder
Aout
Ain
C8
seg8
seg8
seg8
4
6
i0 i1 Wi1 i1en o0 Ain Aout aout Wa Wt Wb


double-16key
C8
8adder
Aout
Ain
C8
seg8
seg8
seg8
2
3
i0 i1 Wi1 i1en o0 Ain Aout aout Wa Wt Wb

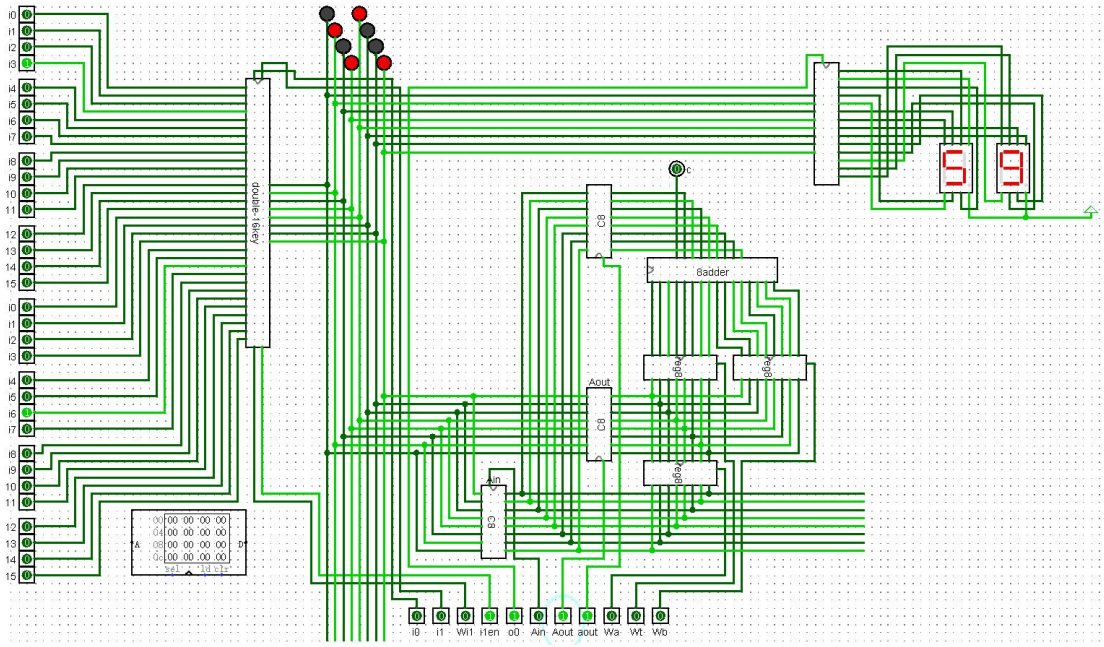


四、实验结果



把两个数分别放在 i0 和 i1 中，打开 Ain 将第一个数据写入寄存器 reg8 中，打开关闭 Wa（高电平转低电平），将第一个数据输入到 A 中，打开关闭 Wb（高电平转低电平），把 A 里的数送到 B 中，数据到达加法器中。关闭 i0，打开 i1 以及 Ain，将第二个数据写入寄存器 reg8，打开关闭 Wt（高电平转低电平），将第二个数据送入 A 中，数据到达加法器中。先关闭 Ain，再打开 aout，运算结果

在内总线上。打开关闭 Wa，将结果送到 A 中，先关闭 i1，再打开 Aout，得到结果。



实验五、自动执行逻辑设计（2 学时）

一、实验目的

掌握能够自动运行加法运算的程序存储计算机（冯诺依曼体系结构)。

二、实验内容

（1）根据手摇式计算机的手摇原理，改为使用 RS 触发器进行手动计算。

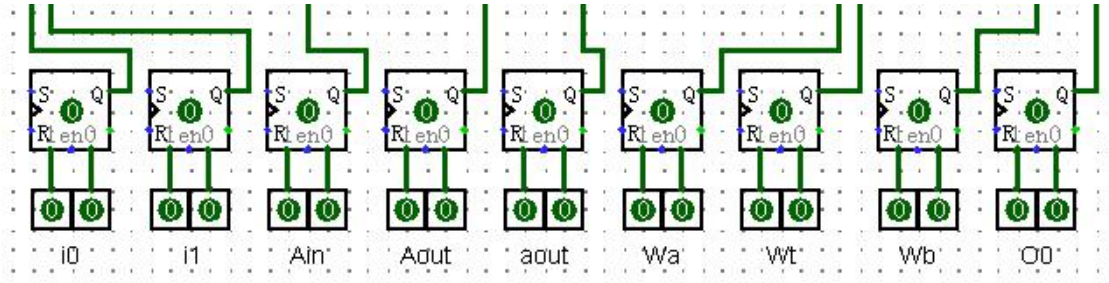
（2）实现硬布线计算机。

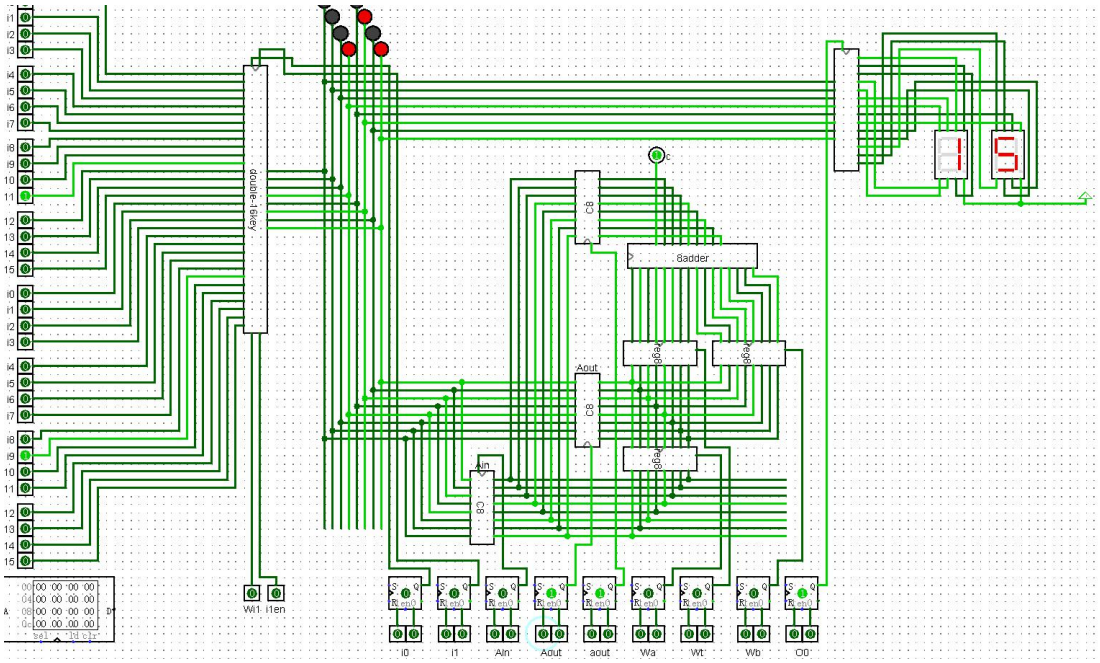
（3）掌握存储器的使用，实现一个简单的冯诺依曼体系结构的程序存储计算机。

（具体实验内容见实验视频指导）

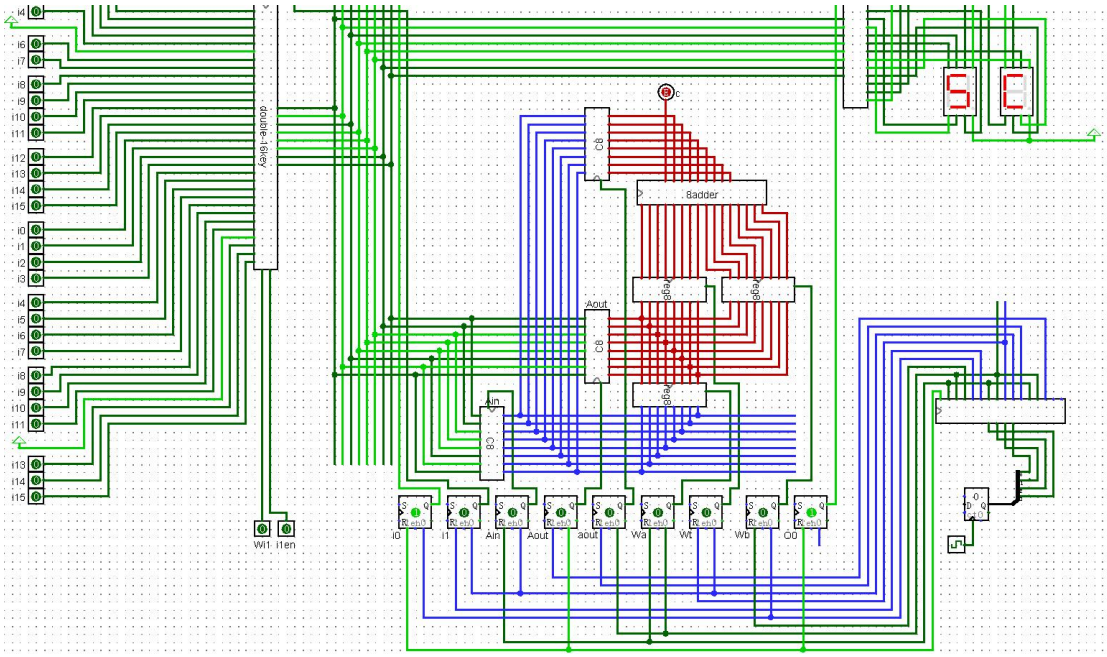
三、实验步骤

在主电路中加入 RS 锁存器，在锁存器的输入端加入两个输入，一个负责置“1”一个负责置“0”，整个部件代替原来的输入端，目的是消除按键的二义性，这种操作更规整。并验证。





实现硬布线计算机：加入一个时序电路，可以在启动时钟模拟时，让机器动起来。



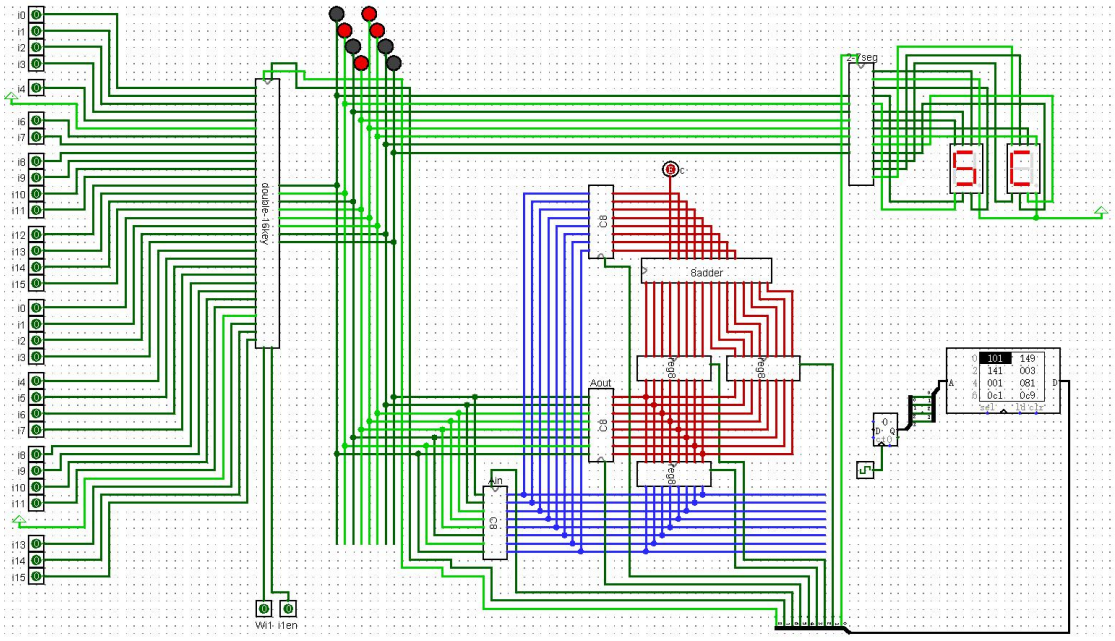
将硬布线计算机变成一个程序存储的计算机。

加入一个 4 位的 RAM，用分线器将其与时钟相连。对加法的步骤进行总结得到表格，并将这个码(mypro1)填入 RAM 中。

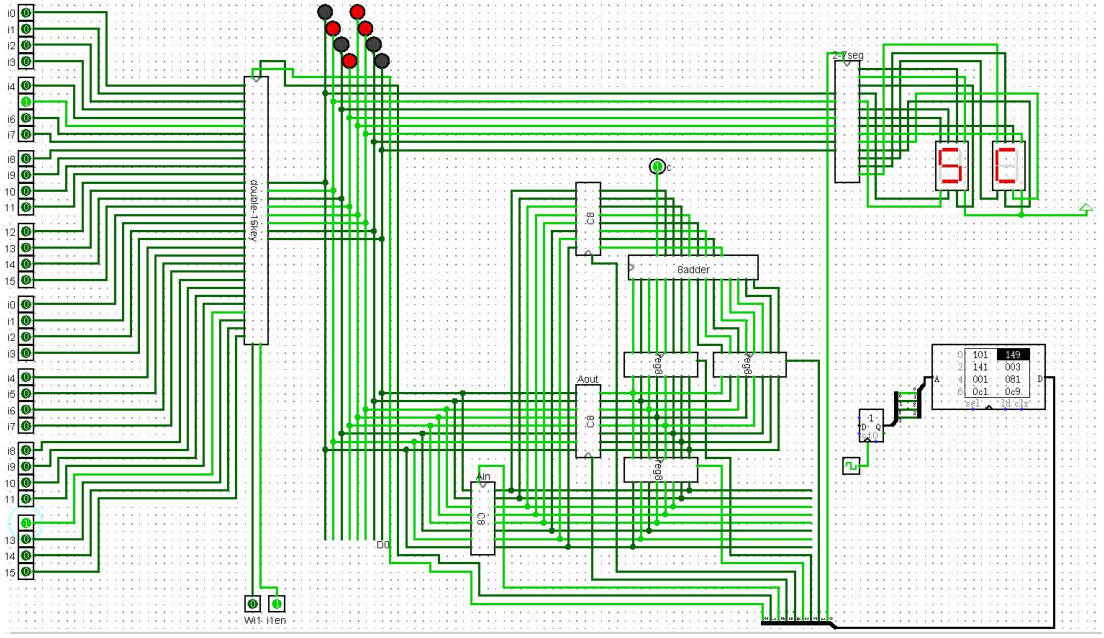
101	149	141	003	001	081	0c1

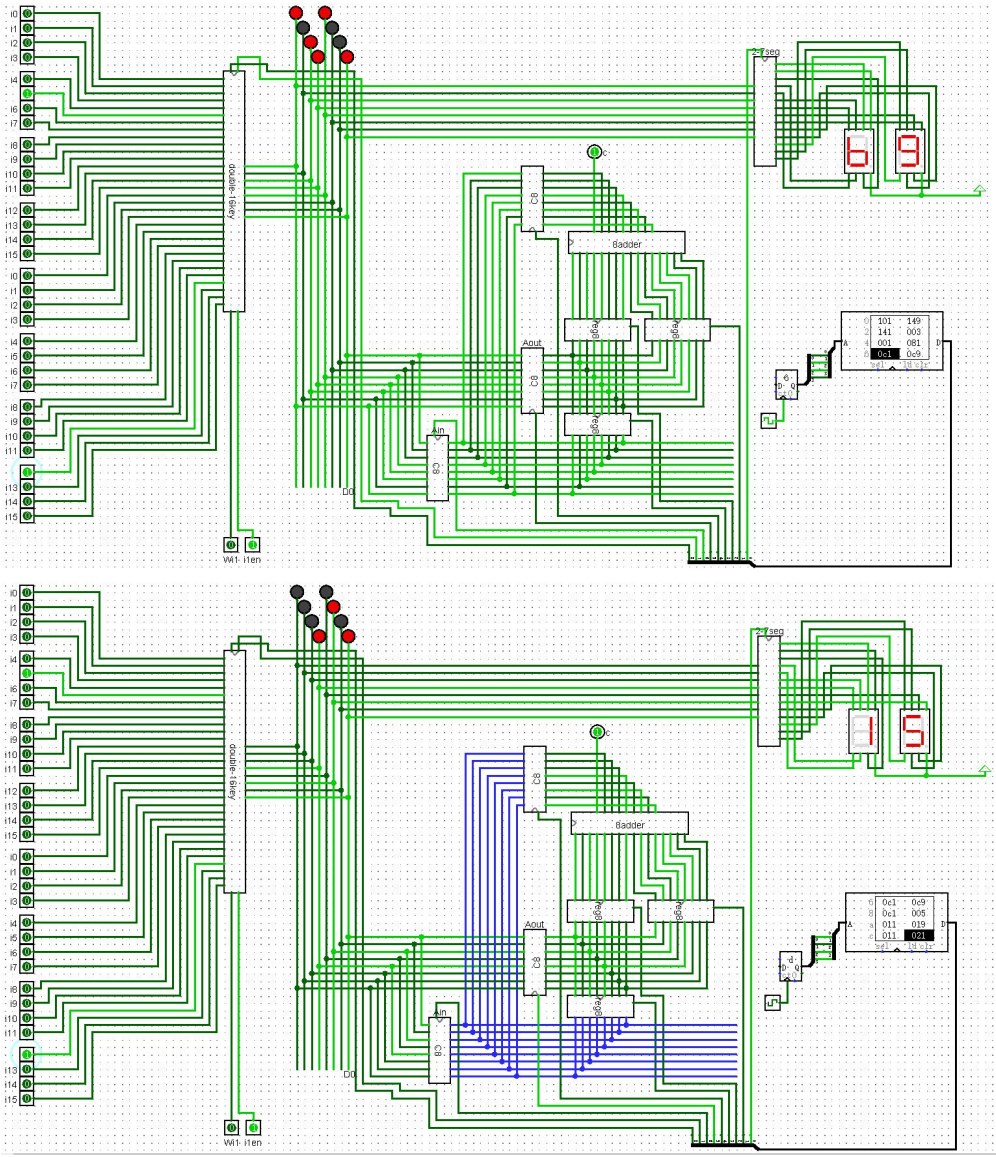
0c9	0c1	005	011	019	011	021

将 RAM 与端口相连。



四、实验结果





启用时钟模拟，得到 5c+b9=15，结果正确。

实验六、控制器设计（2 学时）

一、实验目的

了解微程序控制器结构，掌握微程序的设计与实现。

二、实验内容

（1）将指令转换为微程序。

（2）实现带指令系统的简单计算机。

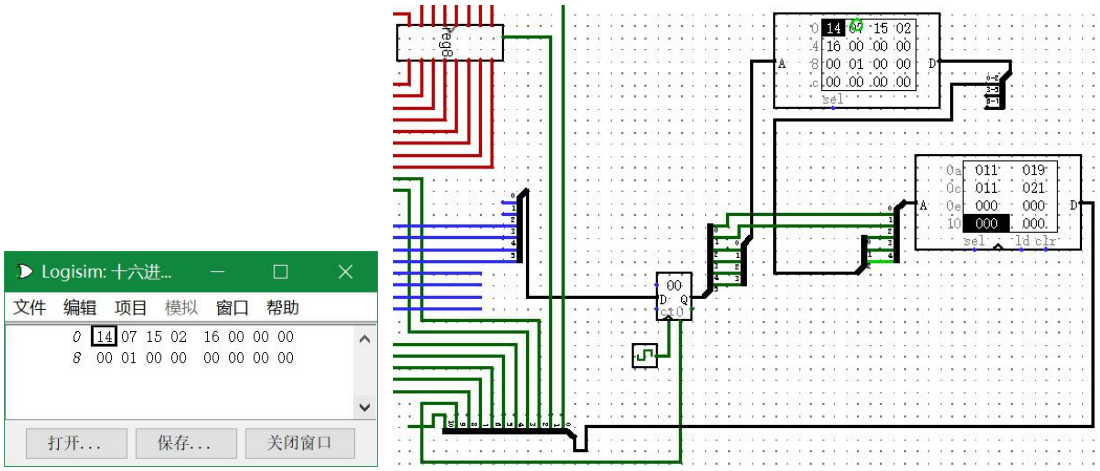
（具体实验内容见实验视频指导）

三、实验步骤

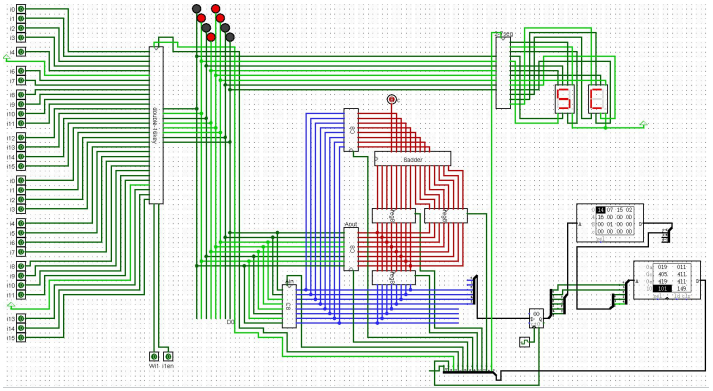
加入一个 ROM，用来放程序。根据指令表填入 ROM 中。

14	07	15	02	16	00	00	00
00	01	00	00	00	00	00	00

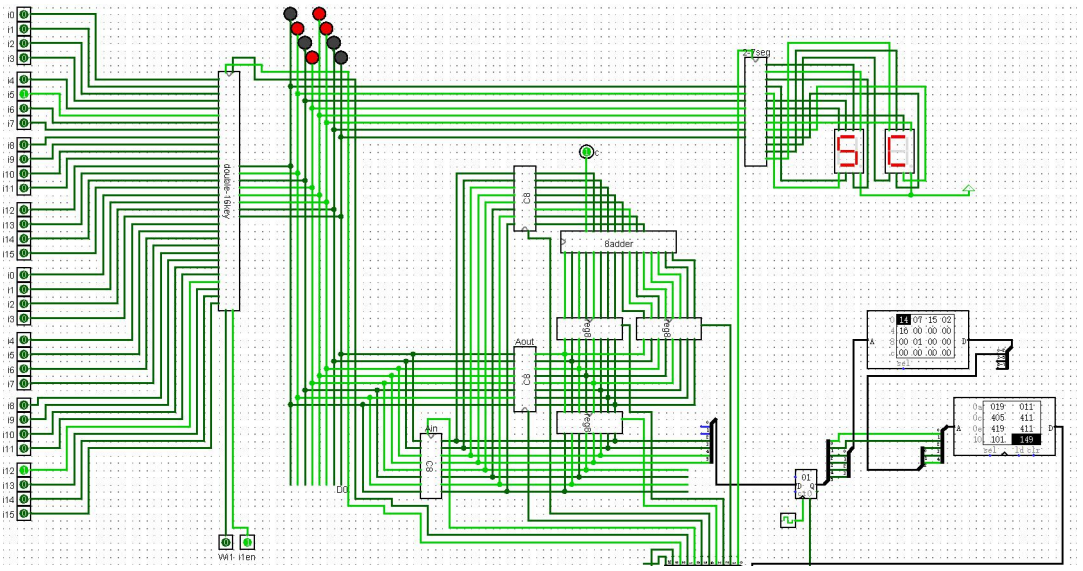
将计数器从 4 位改成 6 位，其中 4 位接新加入的 ROM。将原有的 ROM 的地址位宽改为 5，加入分线器将地址印出来，分线器的低两位与旁边的 6 位计数器相接，剩下三位与新加入的 ROM 相连。由于 Reset 清零需要加一位，控制加减法还需要加一位，故需要增加两位。由于计数器的数据位宽为 6 位，但其中只有四位有用，故一端接入总线（将数据总线的低四位与计数器相连）。

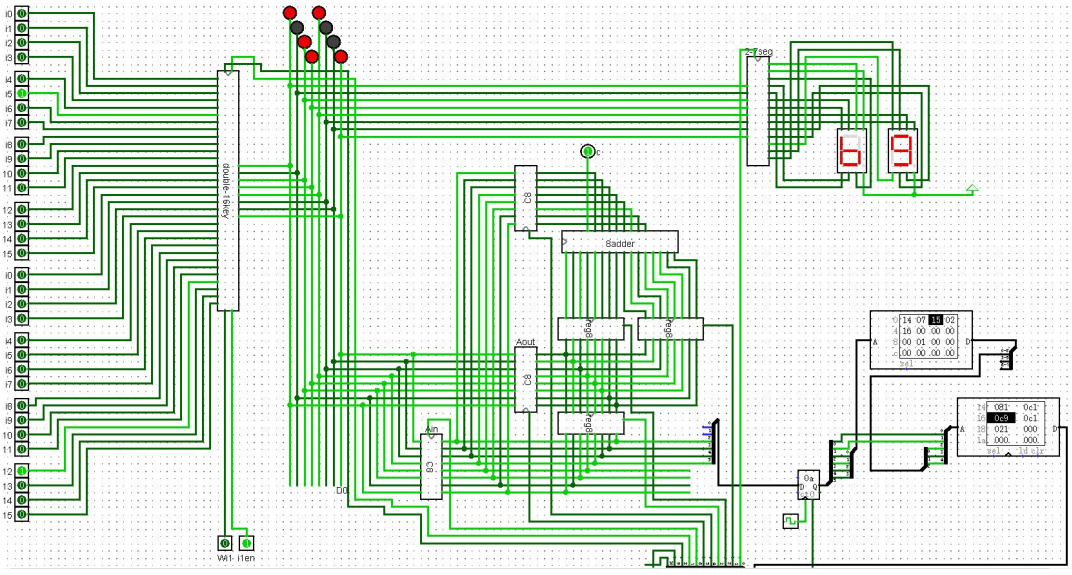


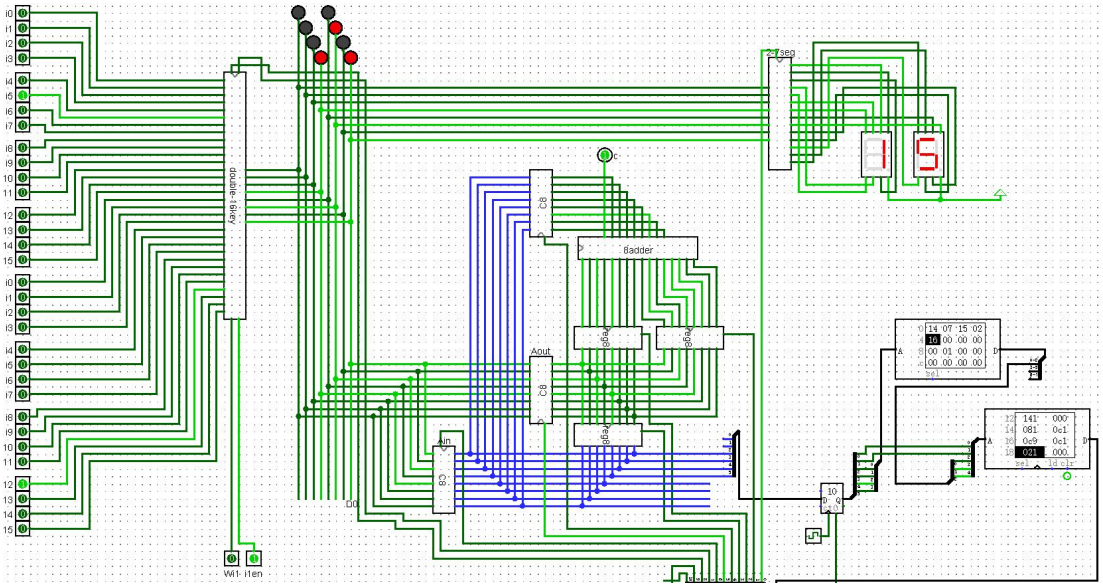
将指令转为微程序，实现带指令系统的简单计算机。



四、实验结果







启用时钟模拟，得到 5c+b9=15，结果正确。

实验七、综合实验（2 学时）

具有加法和乘法功能的、带 8 位指令系统的计算机

一、介绍

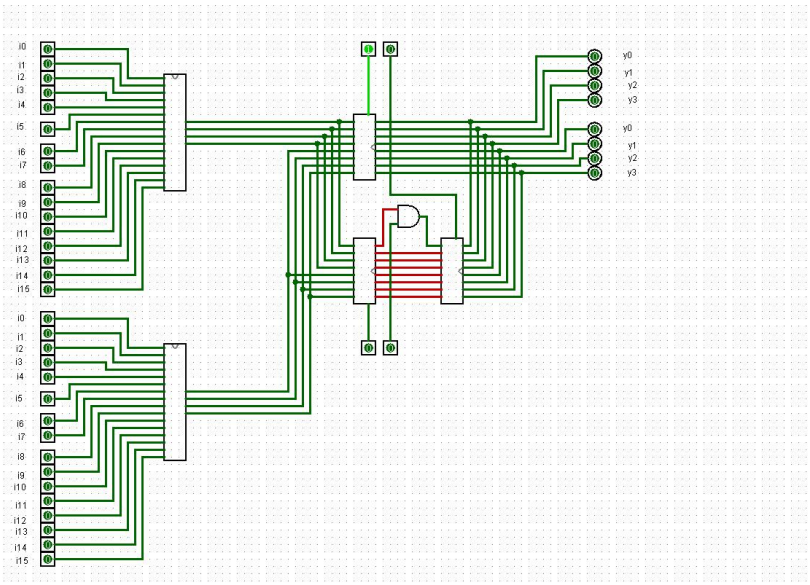
本次实验完成了一个带八位指令系统的计算机，在此计算机中，有输入功能，一次可以输入两个值在 0-15 之间的十进制数，经过电路处理，将这两个数转换为一个有八位的二进制数。本计算机可以实现 8 位加法器运算和 4*4 位的乘法器运算。通过连续的脉冲信号，将实现写好的微指令自动运行。最后通过输出系统，将结果显现在两个 7 段数码管上。

本实现将介绍并演示加法器的实现即运转，乘法器的实现及运转，还有微指令的编写过程以及自动运行的原理和结果。

二、整体设计

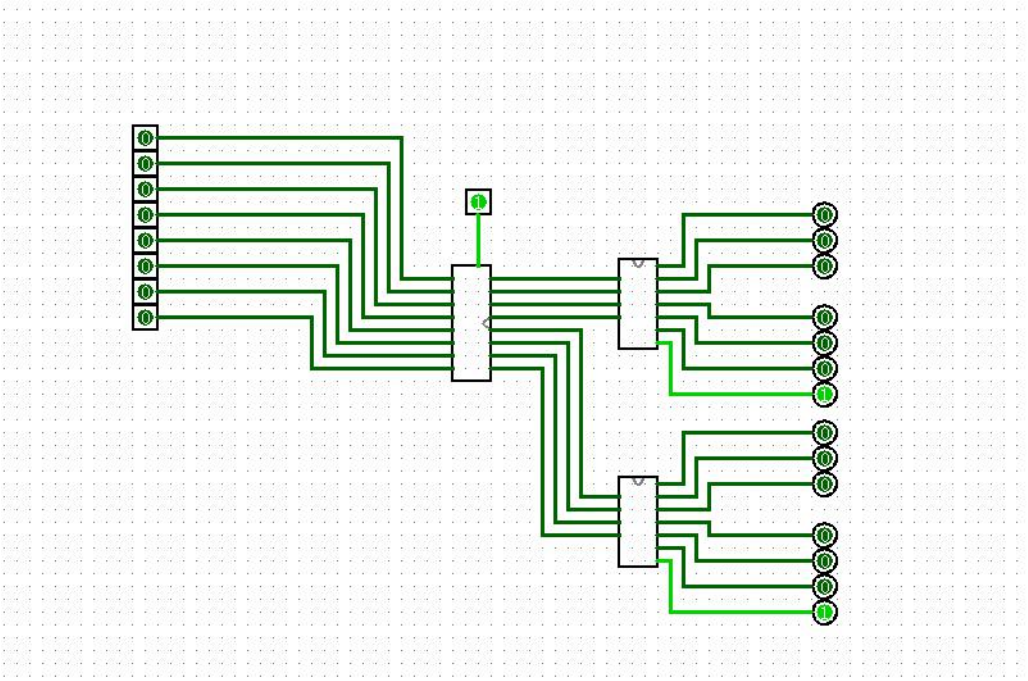
本计算机主要分为四个大部分：输入系统，输出系统，运算器系统以及微指令系统。下面通过电路图进行简单介绍。

- **输入系统的设计**



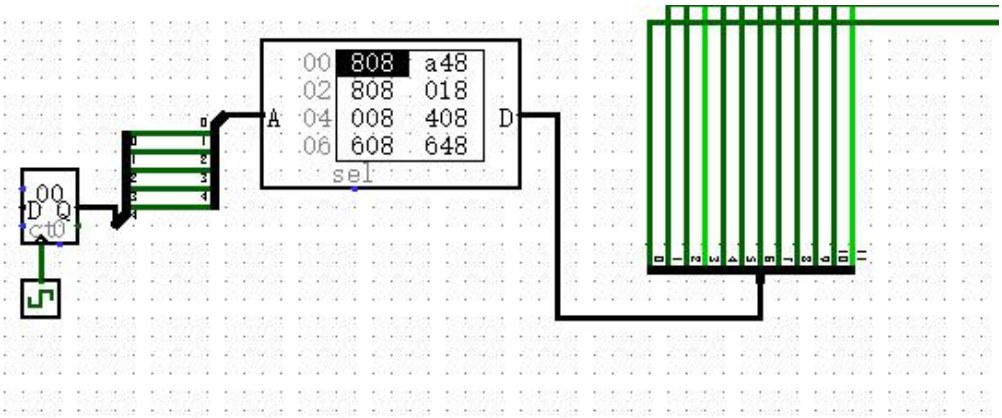
主要电路是 16-key 电路，此电路将两个值为 0-15 之间的十进制数转换为一个八位的二进制数，是本八位计算机的数据来源。

- **输出系统的设计**



主要电路是 2seg 电路，此电路把八位二进制数转换为 16 位二进制数，显现在两个 7 段数码管上。

- **微指令系统的设计**

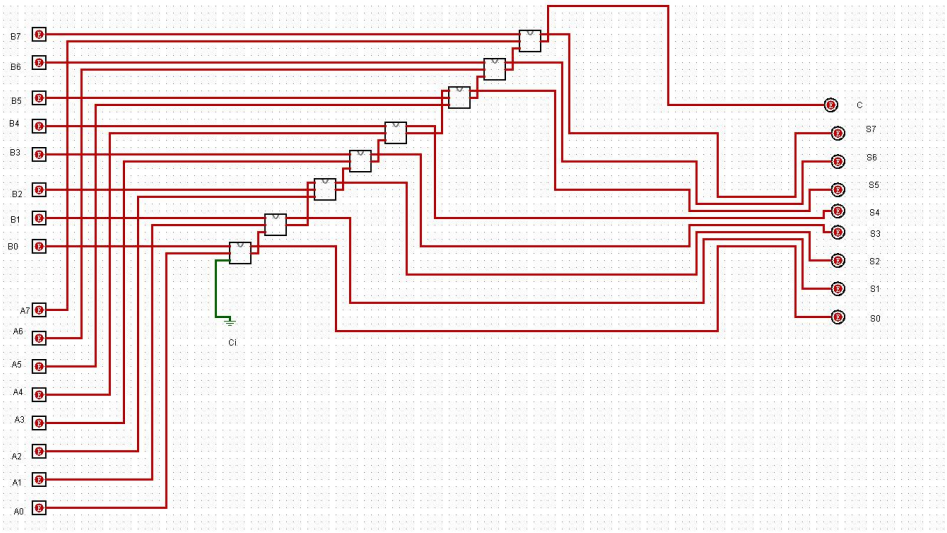


此电路有一个计数器，一个时钟，一个 ROM 和若干分线器组成。ROM 是存储微指令的，而计数器是时钟每变化一次，计数器加一，作为微指令的地址，输

入到 ROM 去，从而是 ROM 的微指令在时钟每变化一次，就自动跳转到下一条指令去。

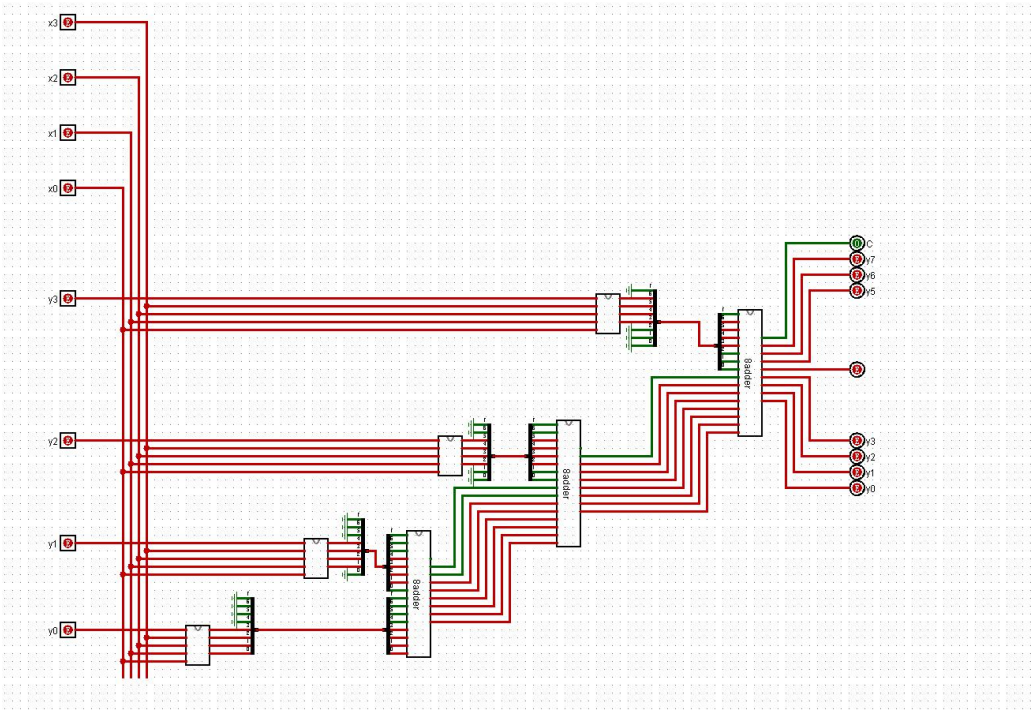
- **运算器系统的设计**

  ✧ 加法器



此电路为 8 位加法器电路，它有 8 个全加器组成，计算两个八位二进制数的和。

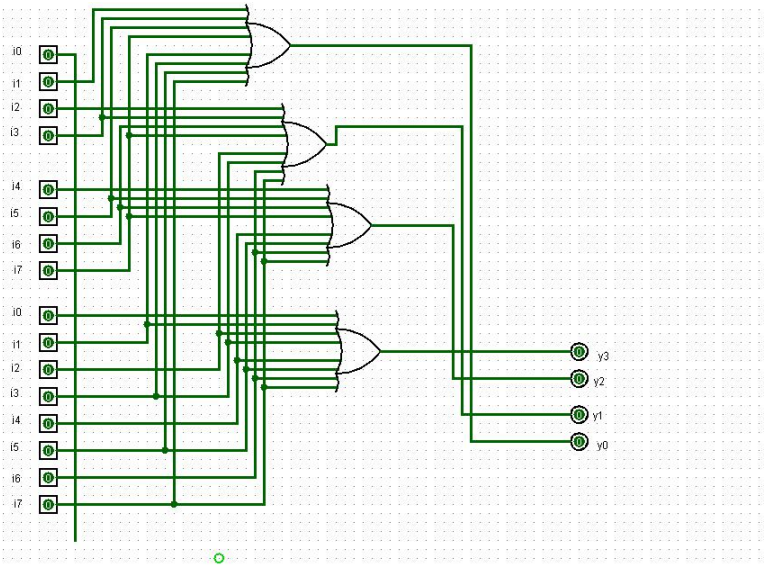
  ✧ 乘法器（扩展）



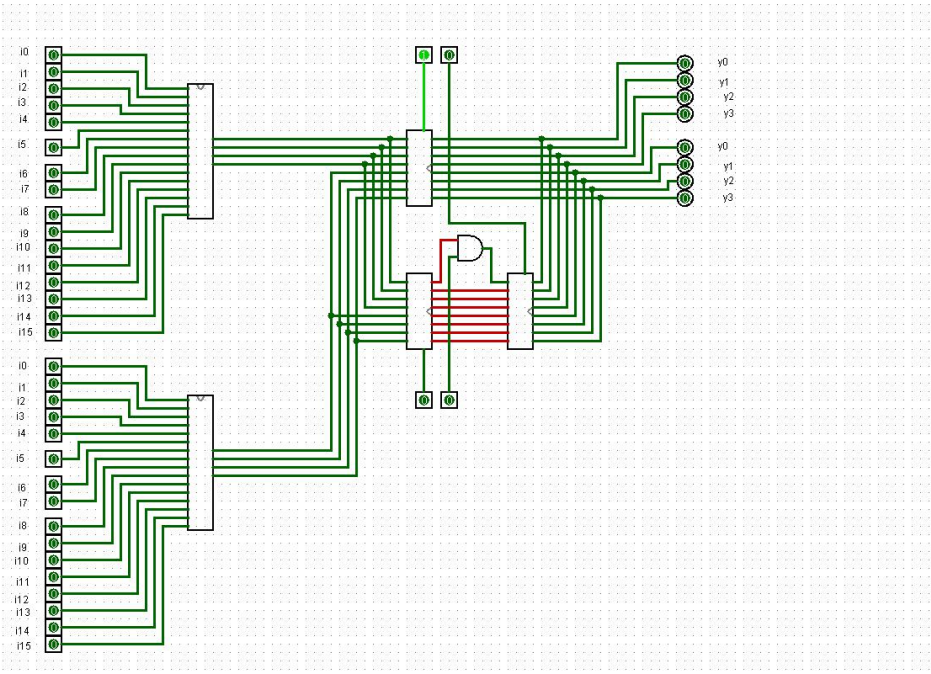
此电路为 4 乘 4 位乘法器的电路，由四个 4 乘 1 位的乘法器，3 个加法器和若干个分线器组成。根据乘法步骤，两个 4 位二进制数相乘，需要 4 次相乘和三次相加即可。图中分线器是将运算结果向左移一位。

三、具体实现步骤

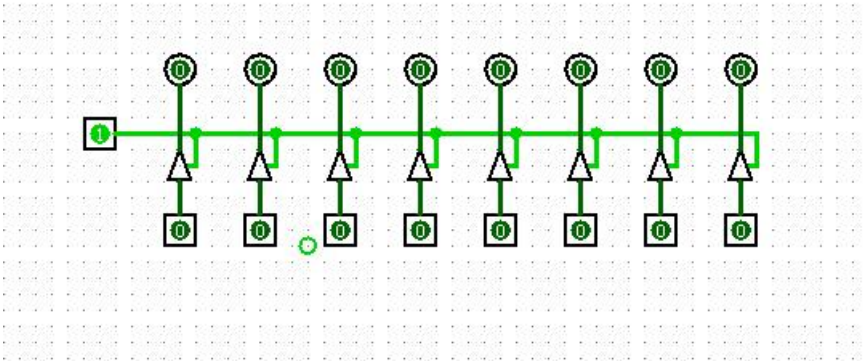
- **输入系统的具体实现**



此 16-4 电路是输入系统的基础电路，它将一位十进制数转化为 4 位二进制数。



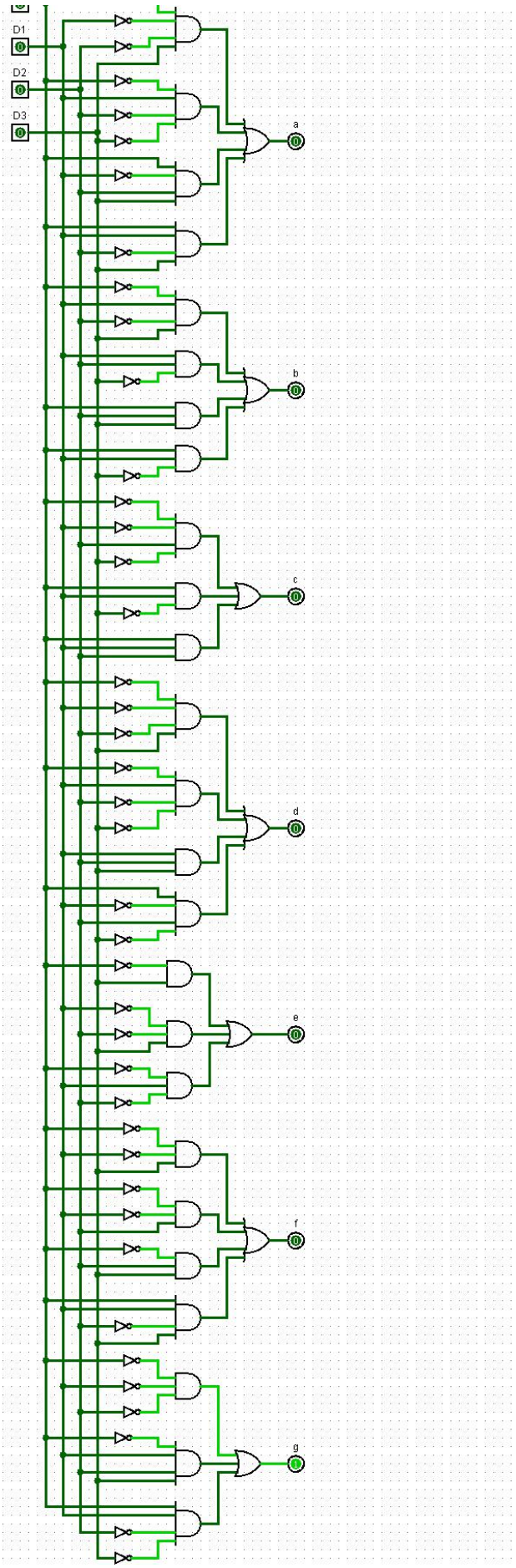
如上图所示，将转换好的二进制数输入到电路 c8 中，如下图



将一个输入作为控制端，如果置 1，那么控制缓冲区将二进制数进行输出，反之则不输出。

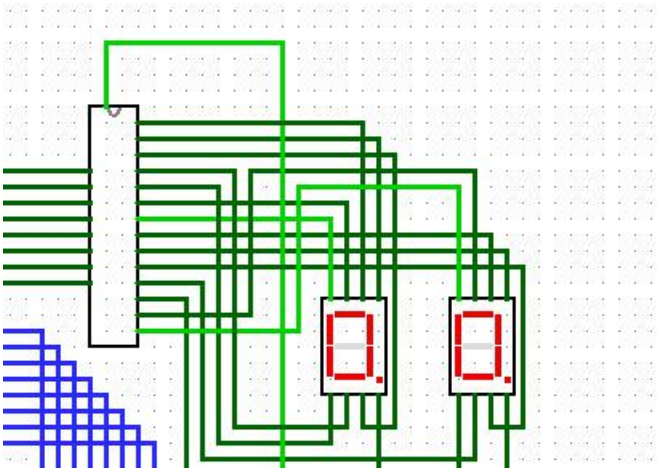
在 16-key 电路中可以看到，还增设了一个 8 位寄存器，此寄存器用来存储转换过的二进制数（在加法器以及乘法器中都有用到），只需给予一次正负脉冲即可进行存储。

- **输出系统的具体实现**


D1
D2
D3
a
b
c
d
e
f
g

此电路 4 位二级制数转换位是 7 段数码管显现对应数值的 7 位二进制数。可以通过 logism 自带的真值表实现电路。真值表如下图:

D0	D1	D2	D3	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0	0	1	0
1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0



同样用 c8 电路作为输出控制电路，置时是使数码管显示数字。

- **指令系统的具体实现**

指令的设计

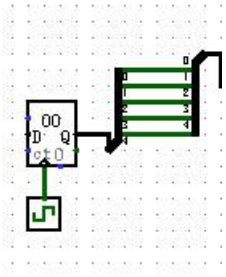
本计算机有多个数据控制端口，所以我们只需要在合适的时机将这些控制端口置 1 即可，不用的控制端口置 0。本计算机有 12 个控制端口，所以 ROM 的数据宽度应该是 12 位，用 12 输出的分线器连接，再将每个控制端口连接在分线器上。本实验要求两个八位二进制数相加之和，以及一次输入的两个四位二进制数

的乘积。指令表如下表:

操作	I0	I1	Ain	Aout	aout	Wa	Wt	Wb	O0	mul	mulcon	addcon	指令
Input0	1								1				808
	1		1			1			1				a48
	1		1			0			1				808
Move a,b								1	1				018
								0	1				008
Input 1		1							1				408
		1	1						1				608
		1	1			1			1				648
		1	1			0			1				608
Add a,b							1		1				028
					1		0		1			1	089
					1	1			1			1	0c9
					1	0			1			1	089
Output add				1					1				109
Input 1		1							1				408
		1	1						1				608
		1	1						1	1			60c
		1	1						1	0			608
mul					1	1			1		1		0ca
					1	0			1		1		08a
Output mul				1					1		1		10a
Input 0	1								1				808
	1		1						1				A08
	1		1						1	1			A0c
	1		1						1	0			A08
mul					1	1			1		1		0ca
					1	0			1		1		08a
Output mul				1					1		1		10a

具体指令相关含义会在运算器系统中讲解。

此指令集有 28 条操作指令, 所以 ROM 的地址长度为 5 为才能装下所有指令集。

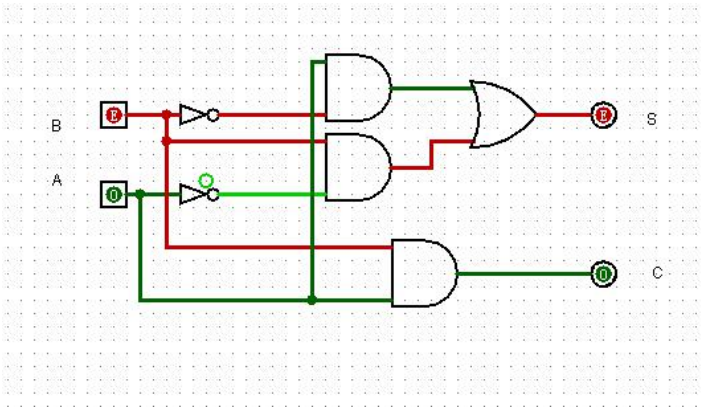


如上图，时钟每变化一次，计数器就会加一，通过分线器转化为二进制数作为指令地址输入到 ROM 中去。所以计数器数据长度也为 5 位才能使指令地址范围大于 28。

- **运算器系统的具体实现**

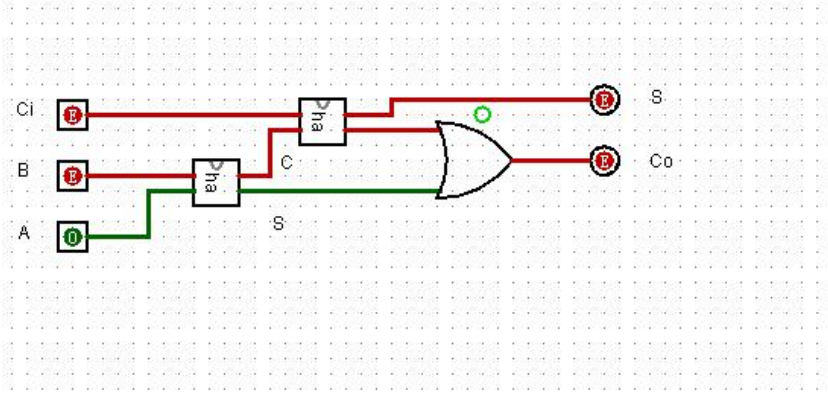
  加法器的实现

  1）半加器



根据 S = A 异或 B　　C = AB 可得出以上电路。

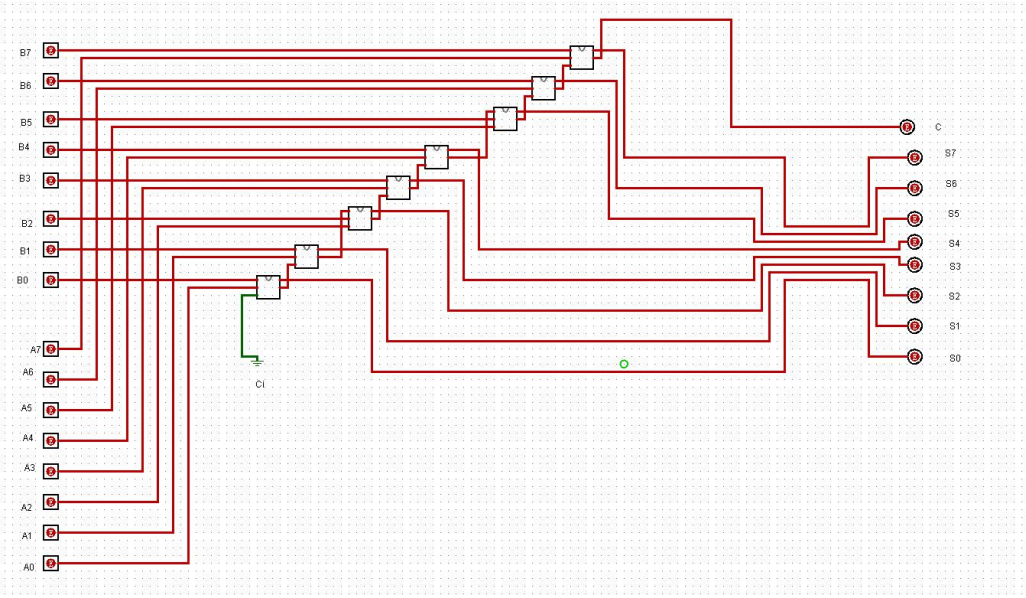
2）全加器



将 A 和 B 产生的进位与上一次运算的进位相加得到的进位作为新的和，

将 A 和 B 产生的和与 A 和 B 产生的进位与上一次运算的进位相加得到的和做与操作作为新的进位。

3）八位加法器



将 8 个全加器串位相连，第一个全加器进位为 0，全加器的输出作为进位输入到下一个全加器中。

4）运算数据的输入和运算结果的输出

1.寄存器将第一个输入的数存起来，作为加法运算的加数，加法需要两个数，如果不进行存储，则读第二个数有可能造成数据的流失。只需要给 W1 一次正负脉冲变化即可将数据进行存储。

2.然后打开 I0 端口，输入第二个加数。

3.接着打开 Ain 控制端口，将数据送入运算器系统中，接着给 Wa 一个正负脉冲变化，将值存入到寄存器中。

4.给 Wb 一个正负脉冲变化，将值存入到寄存器中。

5.关闭 I0 端口，打开 I1 端口。

6. 接着打开 Ain 控制端口，将数据送入运算器系统中，接着给 Wa 一个正负

脉冲变化，将值存入到寄存器中。

7.给 Wt 一个正负脉冲变化，将值存入到寄存器中。

8.然后将两个寄存器的值输入到加法器中，进行加法运算。

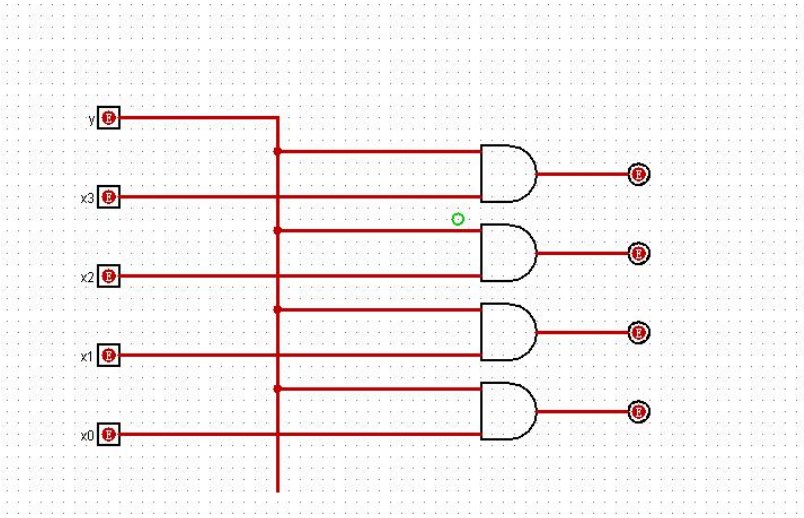
9.打开 addcon 端口和 aout 端口，给 Wa 一个正负脉冲变化，将值存入到寄存器中。

10.关闭 aout 端口，打开 Aout 端口，将运算结果输入到输出系统中去，最后通过 7 段数码管显示结果。

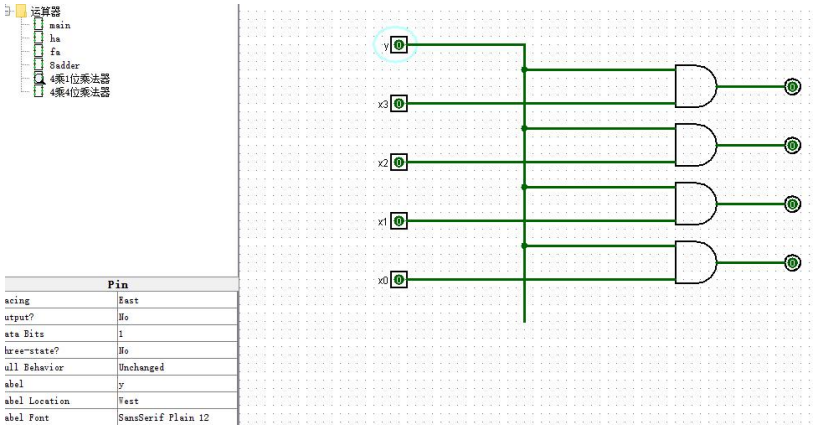
- **乘法器的实现**

计算机中的乘法就是将乘数中的每一位与被乘数相乘，再左移一位，与上一步的乘法结果相加，不断重复此步骤。

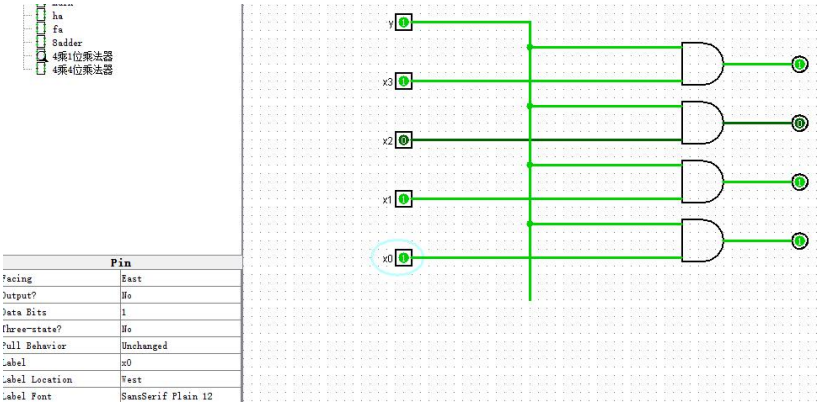
1）4 乘 1 位乘法器



如上图所示，有四个输入为被乘数，而 y 为乘数中的其中一位。将 y 与被乘数的每一位做与运算，得出的四个输出结果几位相乘结果。其实输出结果就只有两种，如果 y 为 1，那么输出结果为被乘数本身，如果 y 为 0，则输出结果全为 0.

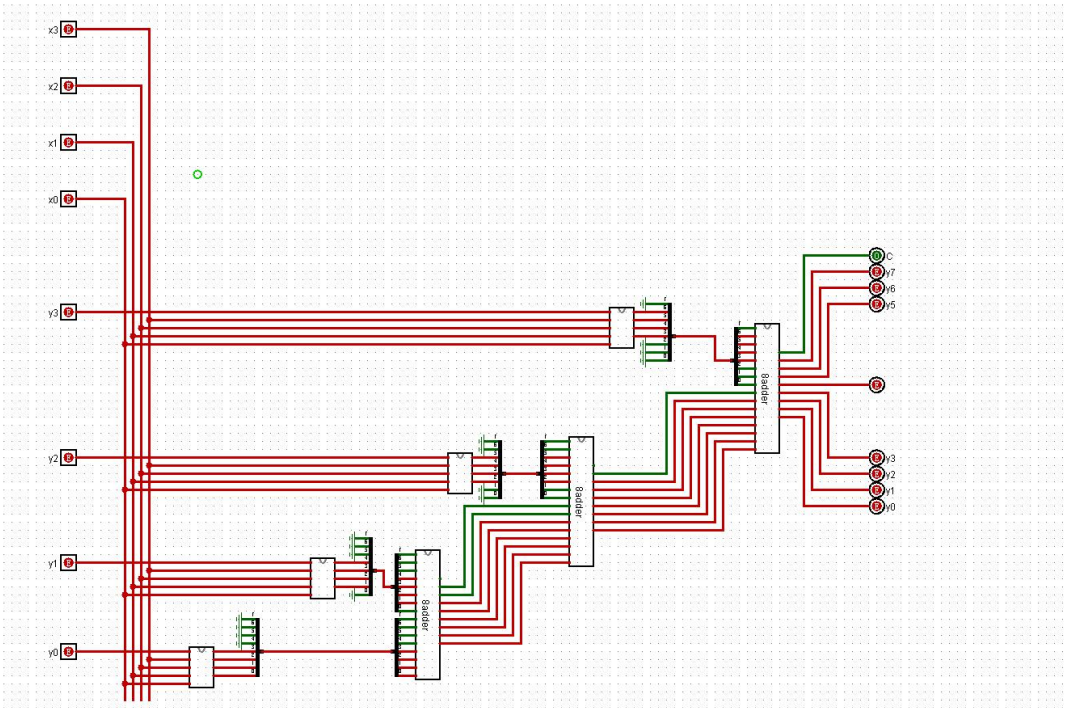


如上图，如果 1011 和 0 相乘，输出全为 0。



如果 1011 和 1 相乘，结果为 1011 本身。

2）4 乘 4 位乘法器

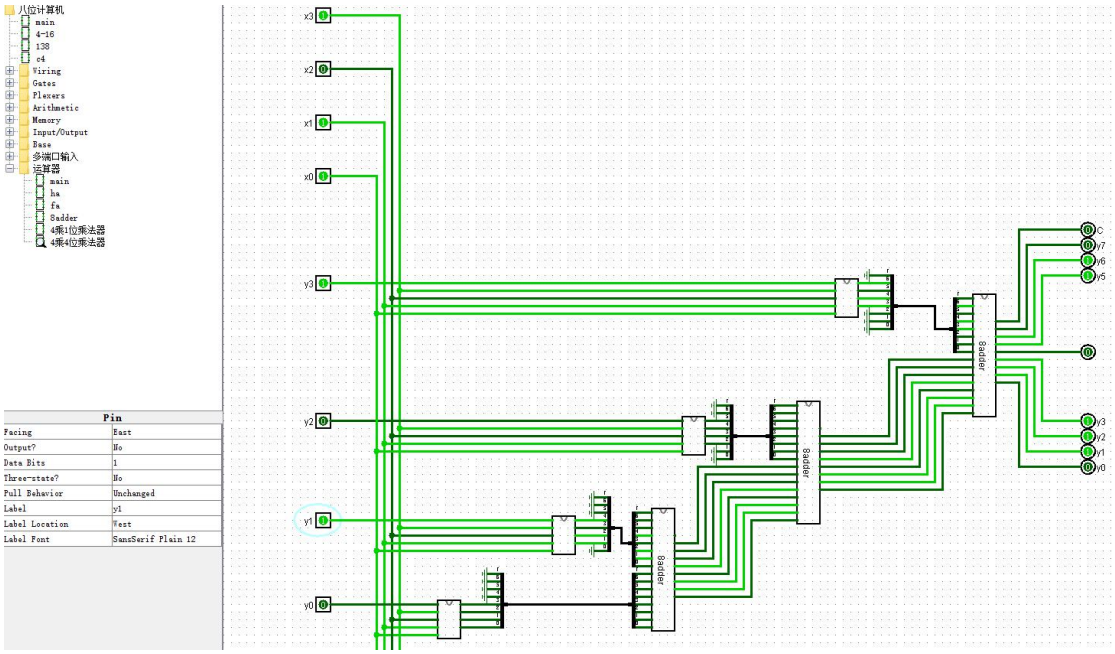


如图所示：

1. 先将乘数的第一位与被乘数相乘，将输出的四位接到分线器的第四位上，分线器的高四位全接地。组成了 8 位二进制数做下一步运算。

2. 将乘数的第二位与被乘数相乘，得出的四位二进制结果需要左移一位，此操作由分线器完成，即将分线器的 1，2，3，4 端连接乘法结果，第 0 位和第 5，6，7 位接地，这样就相当于变相向左移一位。再将上一步的运算结果和这一步的运算结果通过八位加法器进行相加。后续步骤就是不断重复前两步。

3. 最终得到了一个八位的乘法结果。4 位二进制数相乘需要进行 4 次相乘，3 次移位和 3 次相加。



如上图，1011 和 1010 相乘，计算出结果为 01101110.

3）运算数据的输入和运算结果的输出

1.寄存器将第一个输入的数存起来，作为加法运算的加数，加法需要两个数，如果不进行存储，则读第二个数有可能造成数据的流失。只需要给 W1 一次正负脉冲变化即可将数据进行存储。

2.然后打开 I0 端口，输入第二个加数。

3.接着打开 Ain 控制端口，将数据送入运算器系统中，接着给 mul 端口一个正负脉冲变化，将值存入到寄存器中。

4. 将寄存器的值输入到乘法器中，进行乘法运算。

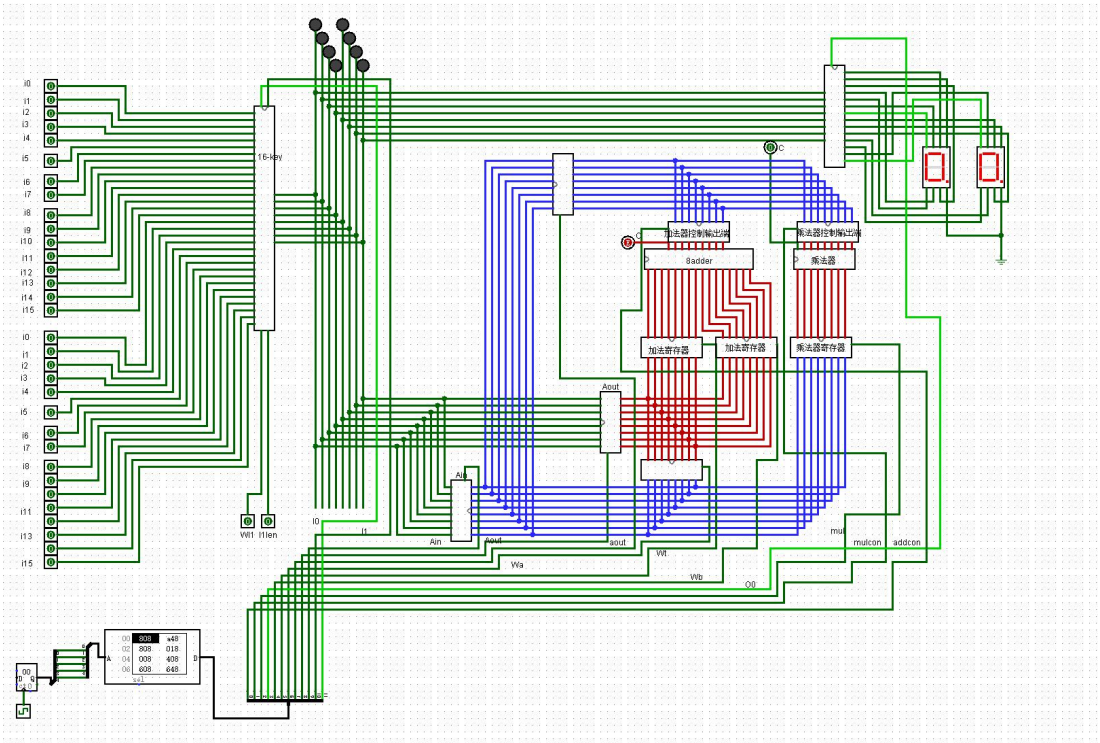
5. 打开 mulcon 端口和 aout 端口，给 Wa 一个正负脉冲变化，将值存入到寄存器中。

6. 10.关闭 aout 端口，打开 Aout 端口，将运算结果输入到输出系统中去，最后通过 7 段数码管显示结果。

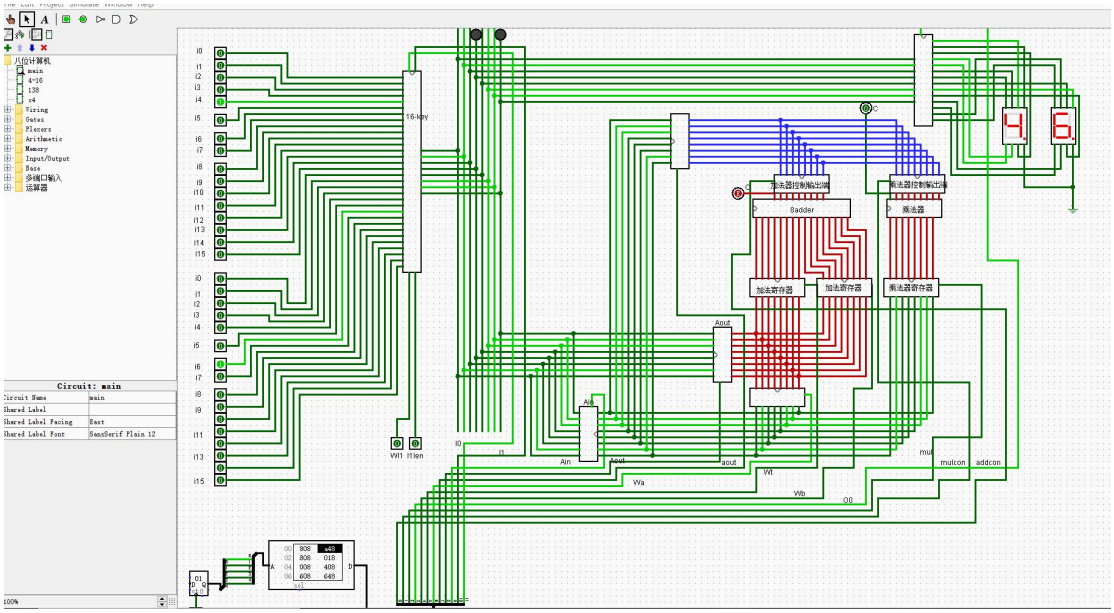
7.另外两个操作数运算步骤与前面步骤相同。

四.实现效果

- **总电路图**

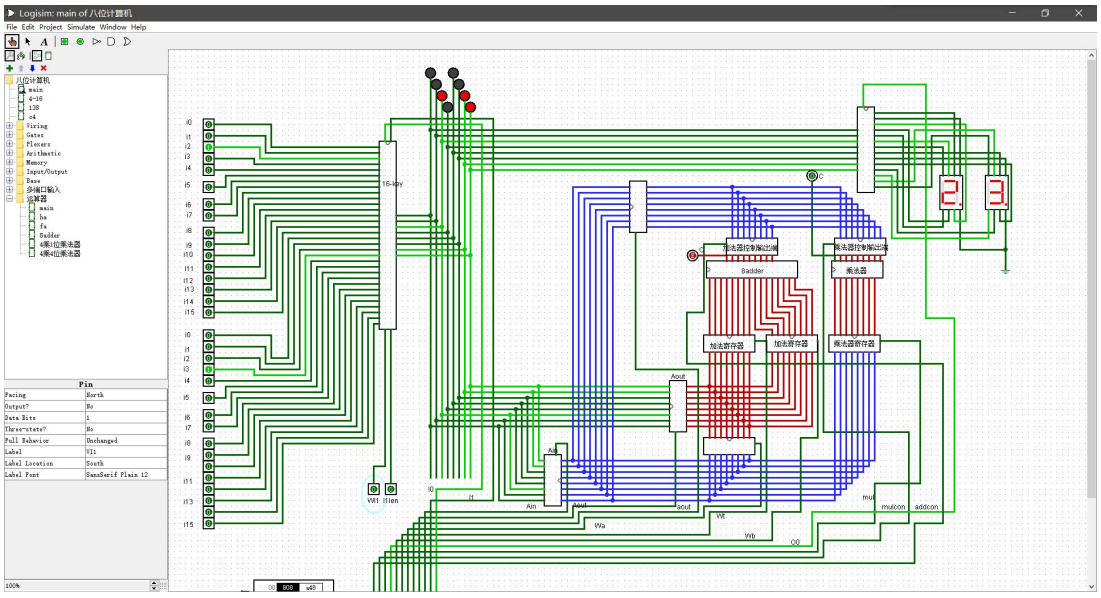


- **总体效果**

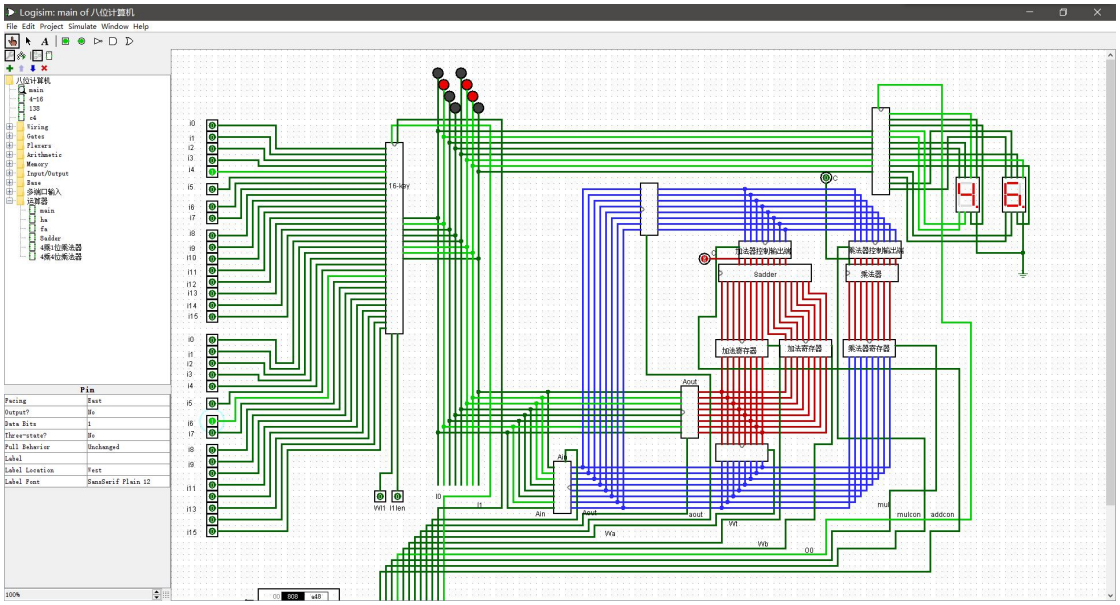


- **具体步骤讲解**

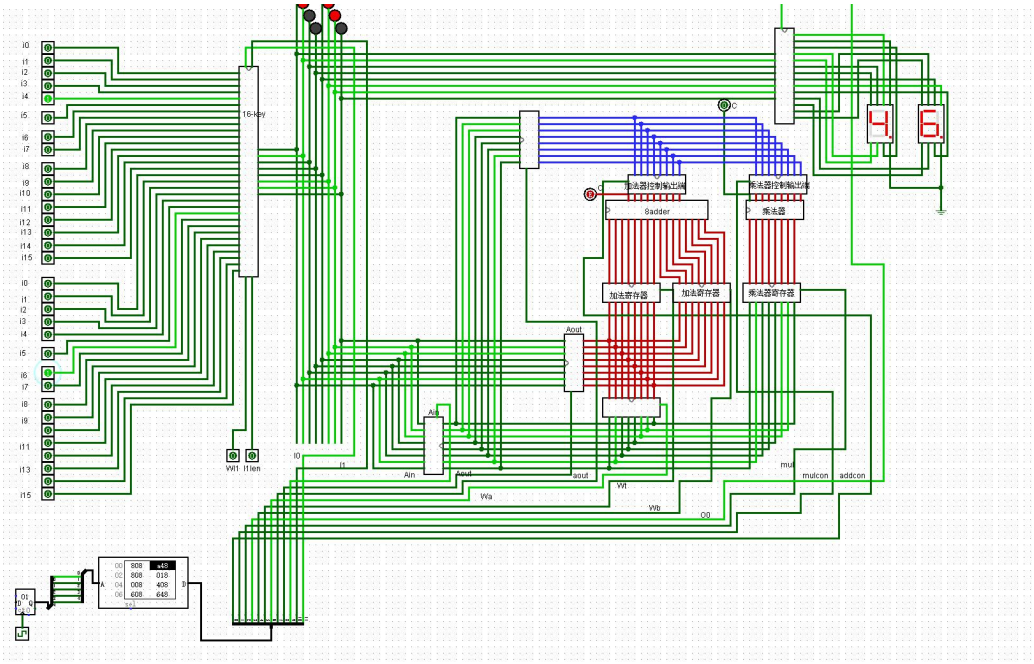
本计算机先实现两个加数的相加结果，在实现同一时间输入的两个数的相乘结果。



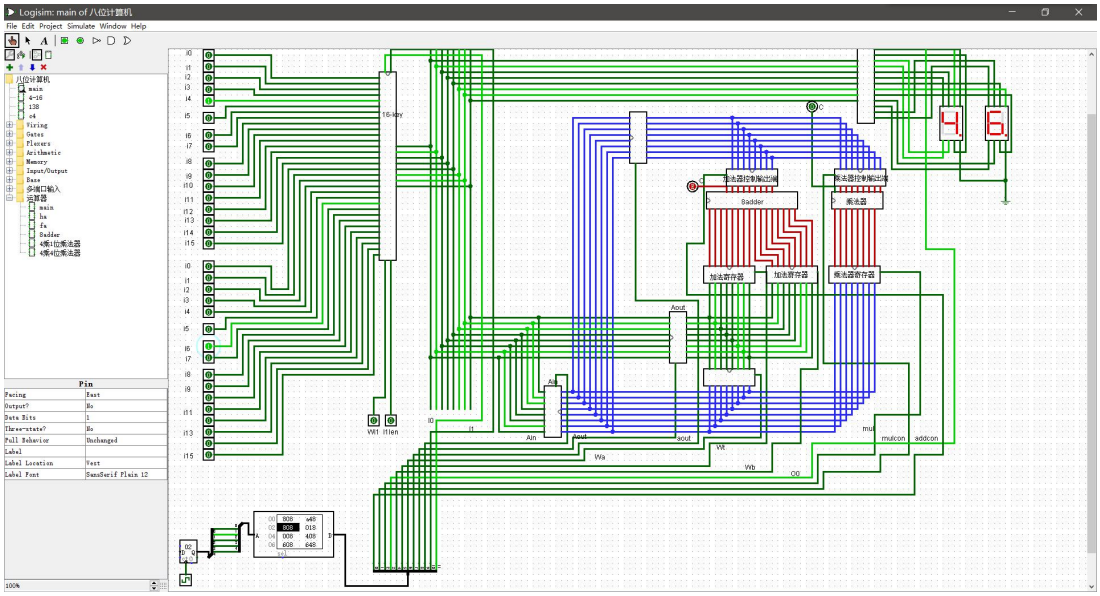
先输入 2，3 通过 W1 存储在寄存器中。



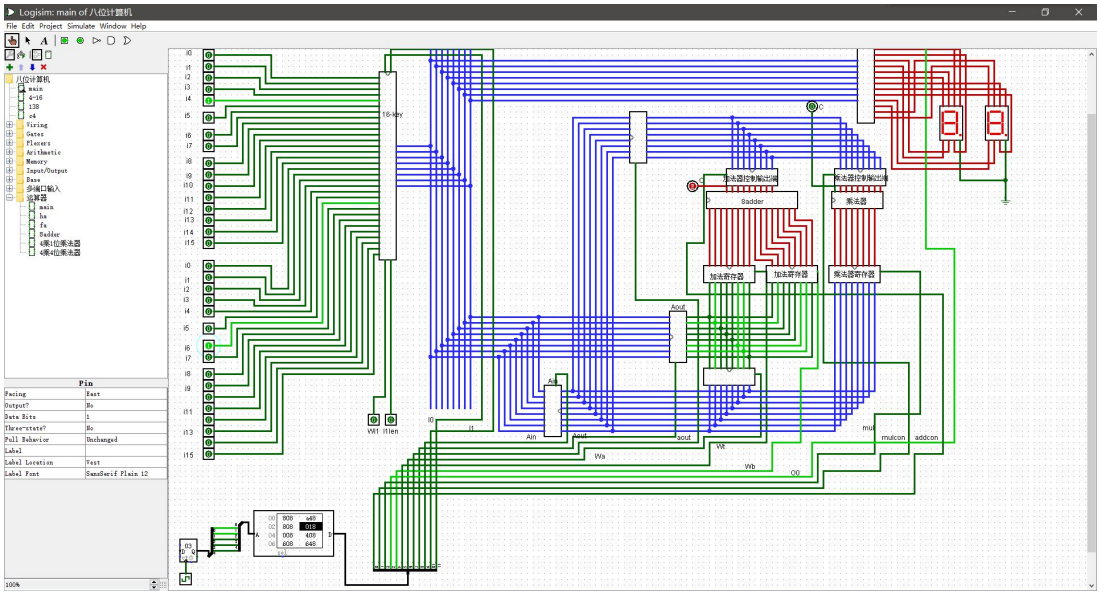
接着输入 4、6，然后开始给计算机一个连续的时钟信号。



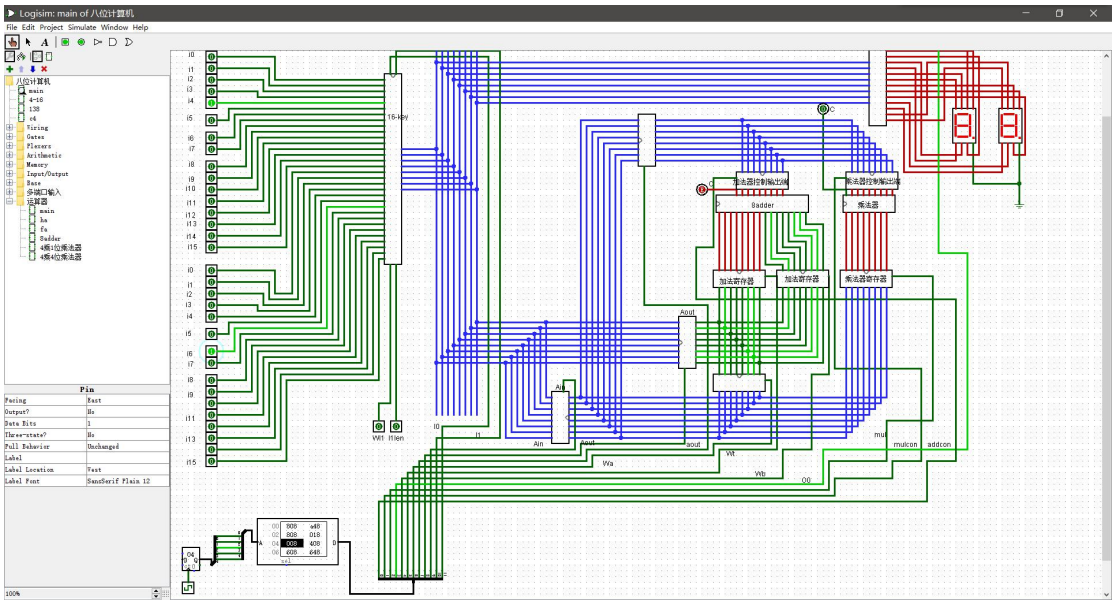
计算机打开 Ain,I0，O0，Wa 端口，将数据输入到运算器系统中。



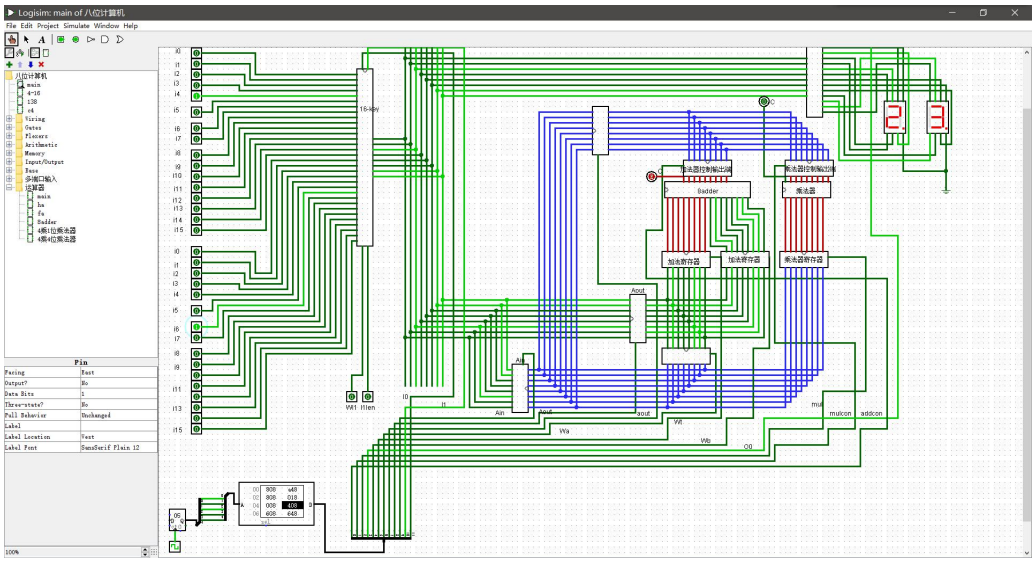
计算机将 Wa 置 0，此时 Wa 经历了脉冲变化，将加数存储在寄存器中。



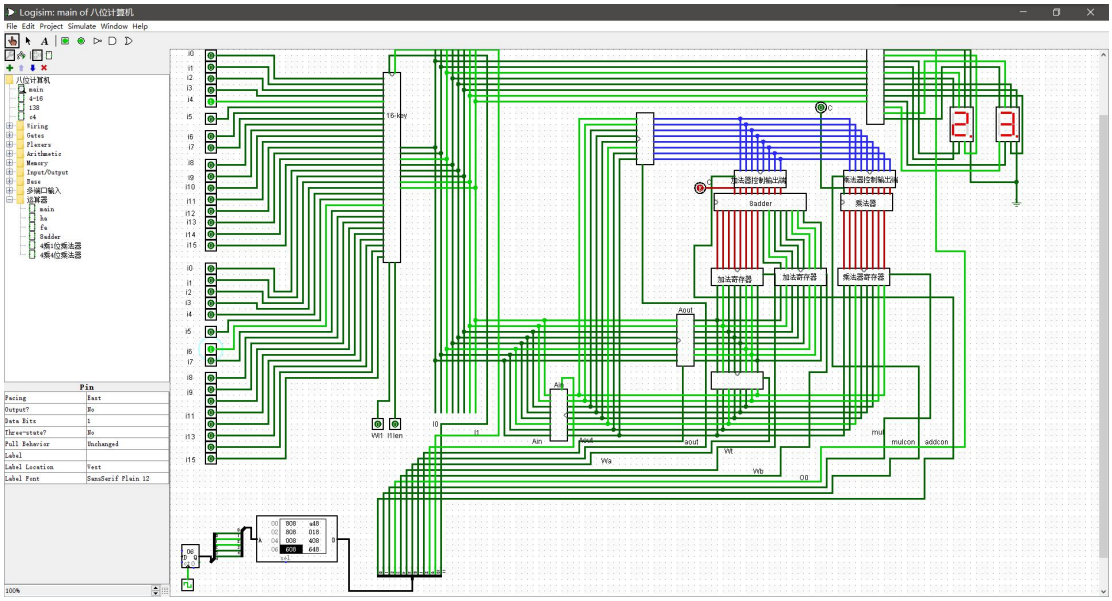
将 Wb 端口置 1



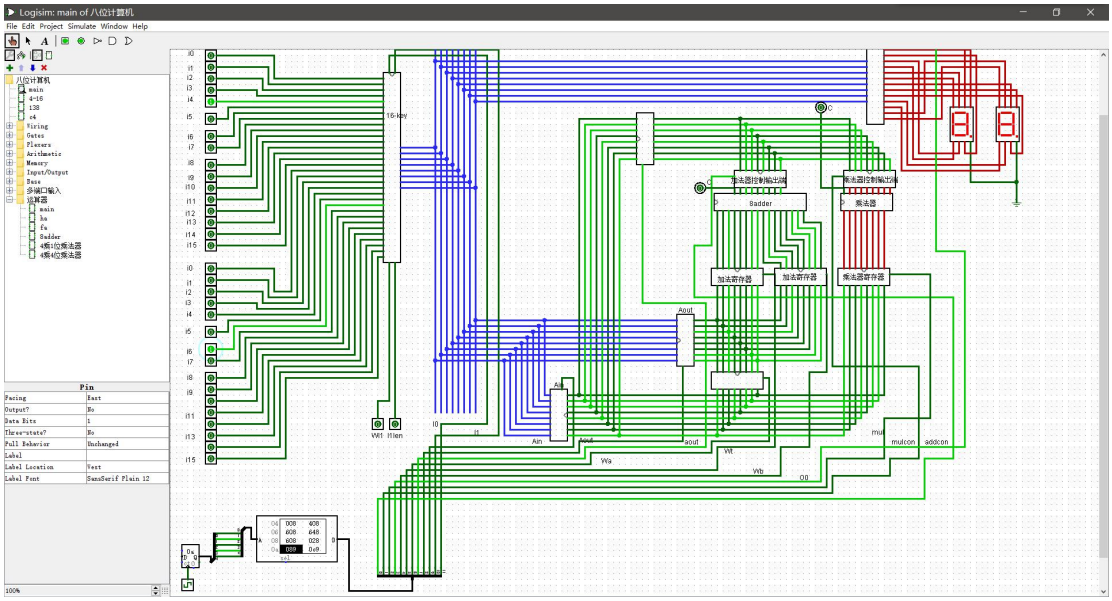
将 Wb 端口置 0，此时 Wb 经历了脉冲变化，将加数存储在寄存器中。



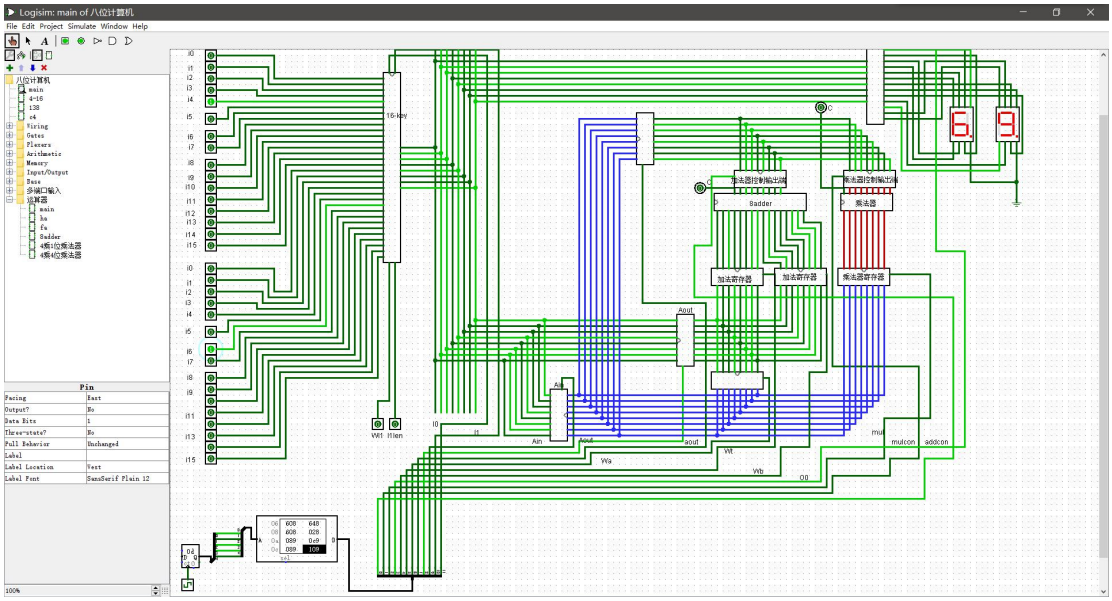
关闭 I0 端口，打开 I0 端口，将之前存储在寄存器中的数取出。



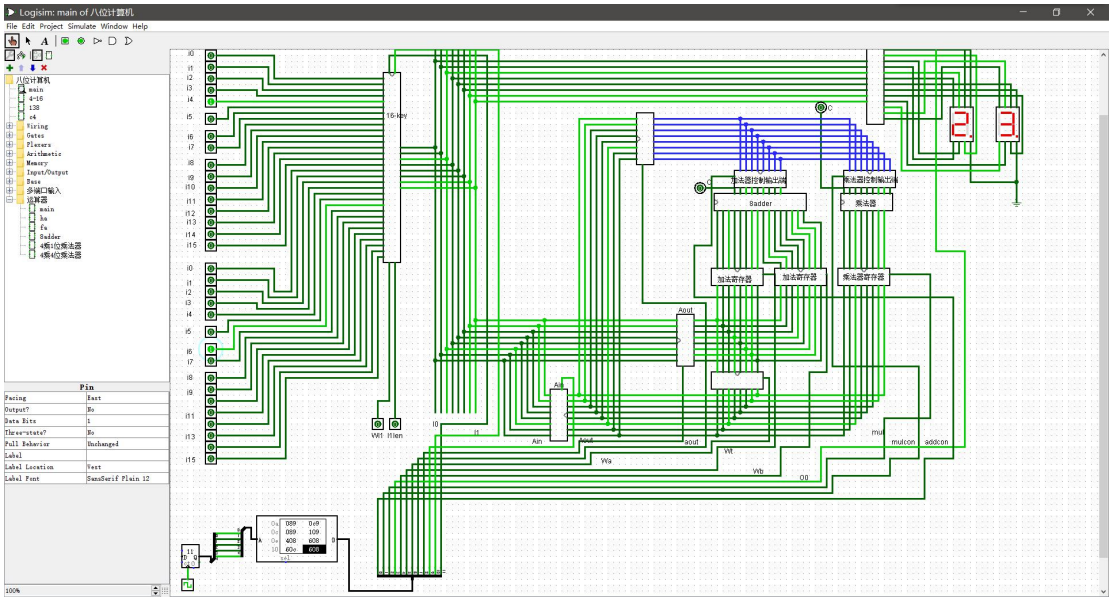
计算机打开 Ain,I0，O0 端口，将数据输入到运算器系统中。



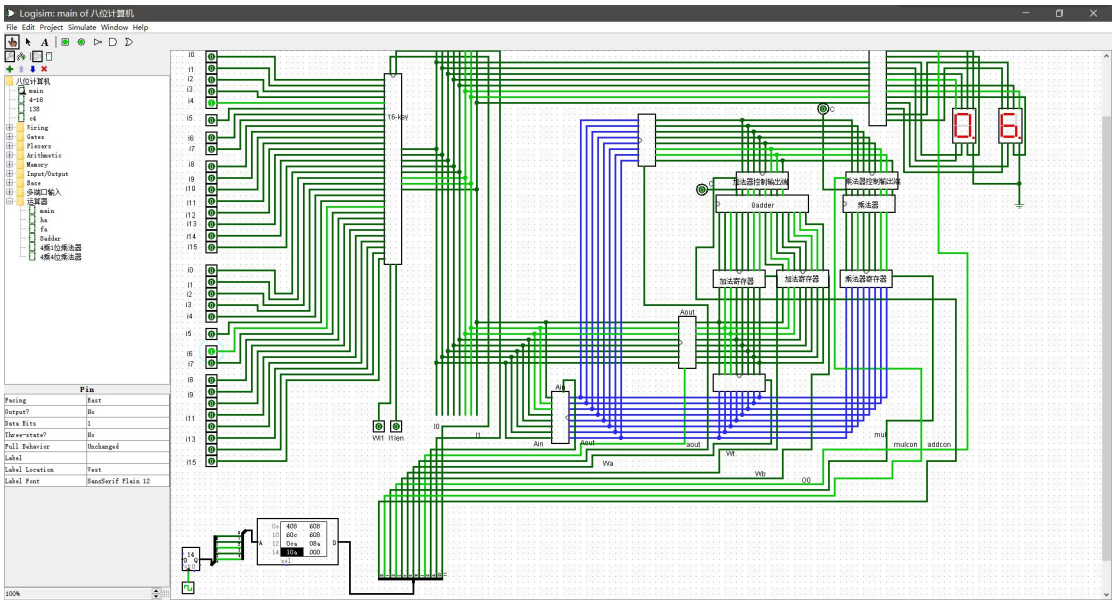
给 Wt 经历一次正负脉冲变化，将两个数送入加法器中进行运算。



打开 aout，addcon 端口，给 Wa 一次正负脉冲变化，将结果存储在寄存器中。再关闭 aout 端口，打开 Aout 端口，将运算结果输入到输出系统中去，运算结果如 7 段数码管所示。

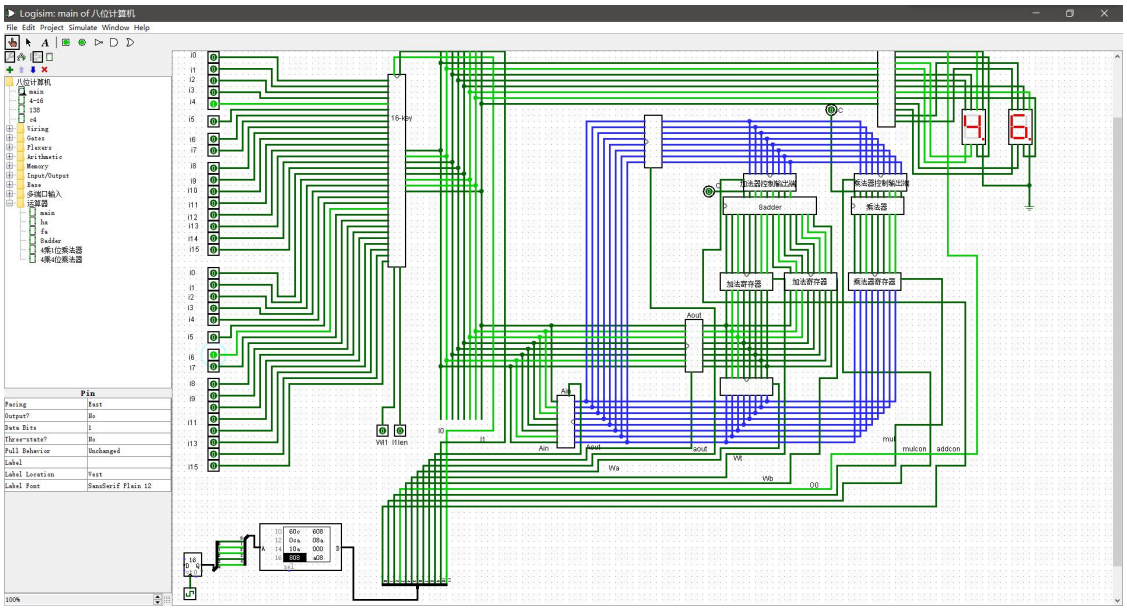


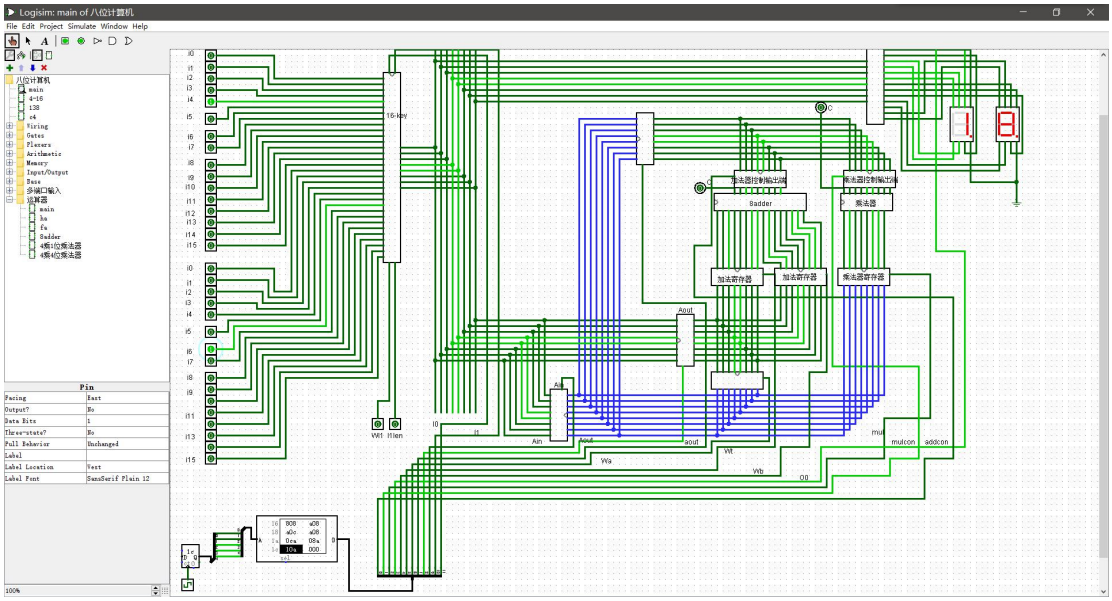
之后进行乘法运算，是将 2 和 3 相乘。导入数据与加法运算相同，接着给乘法寄存器一个正负脉冲变化，将乘数和被乘数存储在计算机中。



接着进行乘法运算，打开 aout，mulcon 端口，给 Wa 一次正负脉冲变化，将结果存储在寄存器中。再关闭 aout 端口，打开 Aout 端口，将运算结果输入到输出系统中去，运算结果如 7 段数码管所示。

第二次输入的两个数与前面步骤相同，在这里展示结果。(7 段数码管中的数为 16 进制)




Logisim: main of 八位计算机
File Edit Project Simulate Window Help
八位计算机
main
4-16
138
74
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
多端口输入
运算器
main
ha
fa
Subber
4数4位乘法器
4数4位乘法器
Pin
Facing East
Output? No
Data Bits 1
Three-state? No
Pull Behavior Unchanged
Label
Label Location West
Label Font SansSerif Plain 12
100%
16-key
Badder
加法寄存器
加法寄存器
乘法器寄存器
乘法器
Aout
Ain
Aout
aout
Wa
Wb
Wt
mul
mulcon
addcon
C