**浙大城市学院实验报告**

课程名称 计算机综合课程设计（一）

实验项目名称 数字逻辑电路实验

学生姓名 曹鑫 专业班级 计算机2401 学号 32408078

指导老师 实验日期

## 一、实验目的

1. 熟悉基本的门电路；

2. 掌握Logisim分析设计简单的组合电路；

3. 掌握Logisim分析设计简单的时序电路；

## 二、实验内容

1. Logisim的下载安装

2.Logsim基本门电路实验

3.Logisim组合电路设计

4.Logisim时序电路设计

## 三、实验步骤

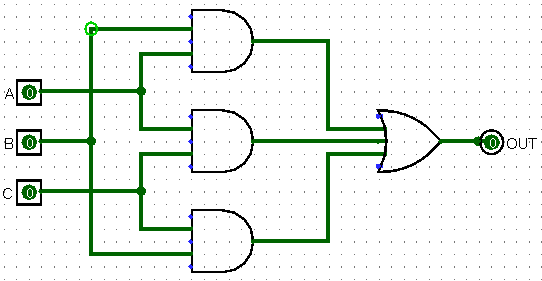
**请完成以下任务，把设计的逻辑电路图截图放在文档中，并把文档和\*.circ文件打包一起上交。**

### 1. 基本门电路实验

任务1：在Logisim中根据给定的布尔代数表达式（F=AB+BC+CA）绘制逻辑电路。

**案例场景举例：**举重比赛裁判电路。在举重比赛中，通常有三位裁判（A、B、C）对运动员的试举是否成功进行裁决，当有两位或三位裁判判定试举成功则该运动员试举成功（F）并获得此次试举的重量记录。

**逻辑电路图截图：**

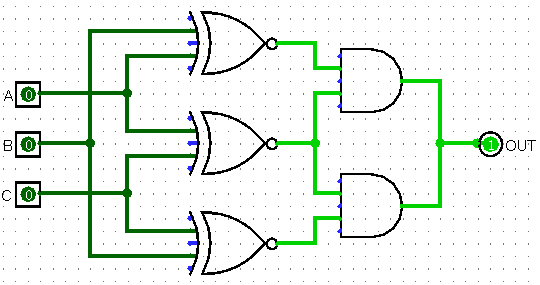
****

任务2：根据如下给定的真值表绘制逻辑电路，其中A、B、C是输入变量，F是输出变量。

| ***A*** | ***B*** | ***C*** | ***F*** |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**案例场景举例：**一致性电路。当多人（以3人A、B、C为例）进行表决，当大家意见完全一致时通过决议，此时F=1。

**逻辑电路图截图：**

****

### 2.Logisim组合电路设计

**2.1 任务1：加法器设计**

**半加器设计：**利用在Logisim中的“组合逻辑分析”工具自动生成半加器电路。

半加器电路是指对两个输入的二进制数据位A、B相加（没有进位输入），输出和Sum与进位Cout​，是实现两个一位二进制数的加法运算电路。真值表如下：

| ***A*** | ***B*** | ***Cout*​** | ***Sum*** |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

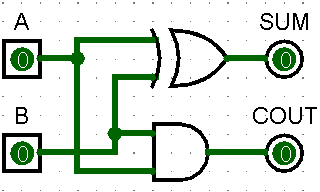
**逻辑表达式如下：** Sum=A⊕B Cout​=AB

**说明：**

在Logisim中打开实验电路框架，在工程中的“半加器自动生成”子电路中，启动“组合逻辑分析”工具，操作如下图鼠标点击填写真值表，如下图完成真值表后，点击“生成电路”按钮即可！

|  |  |
| --- | --- |
|  |  |

**逻辑电路图截图：**

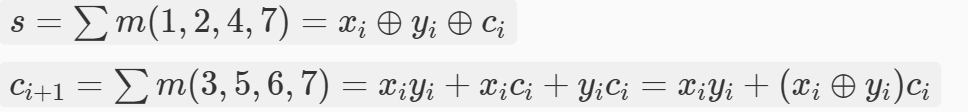
****

**全加器设计：根据全加器原理图在Logisim中手工绘制全加器电路**

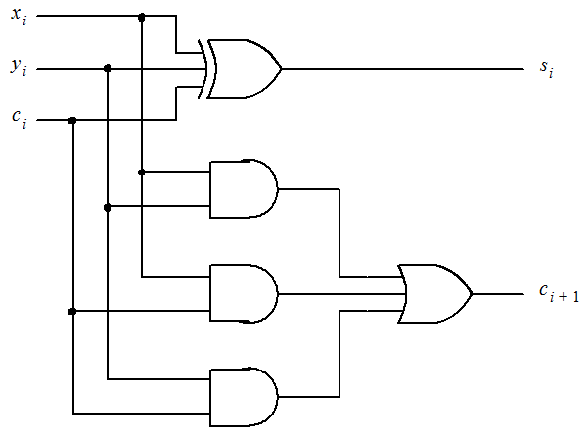
全加器FA（Full Adder）是实现两个1位二进制数（xi​、yi​）和来自低位进位（ci​）相加，产生和（si​）与进位输出（ci+1​）的组合逻辑电路。真值表如下：

| ***xi*​** | ***yi*​** | ***ci*​** | ***ci*+1​** | ***si*​** |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

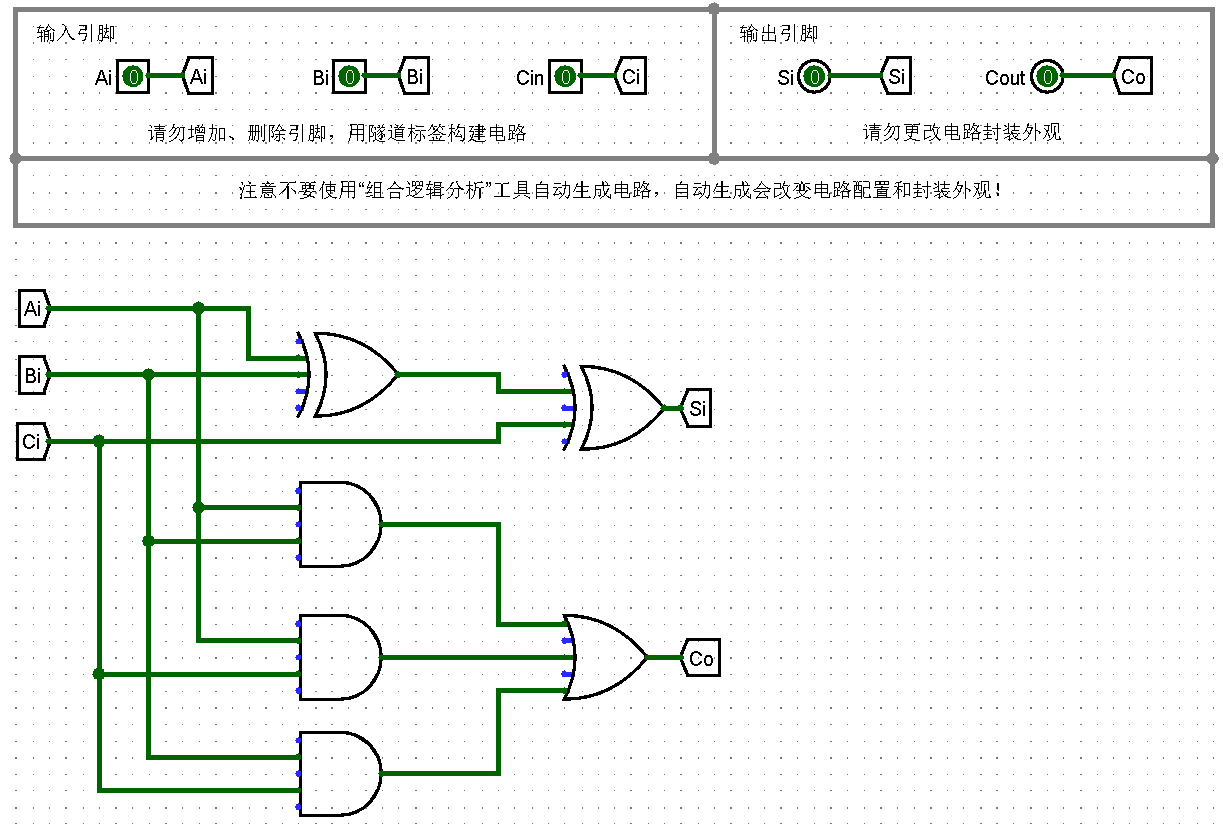
全加器的逻辑表达式如下：



路原理图如下：

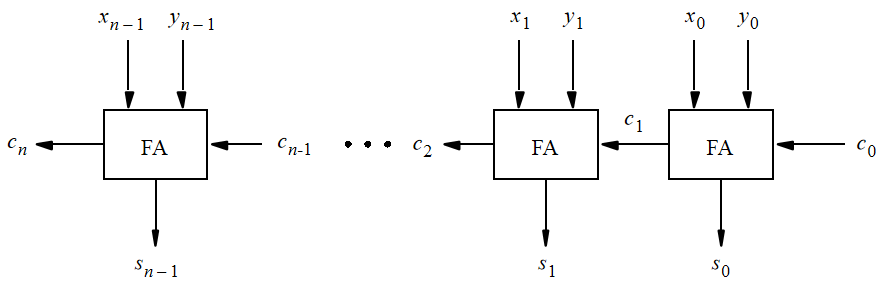


**逻辑电路图截图：**

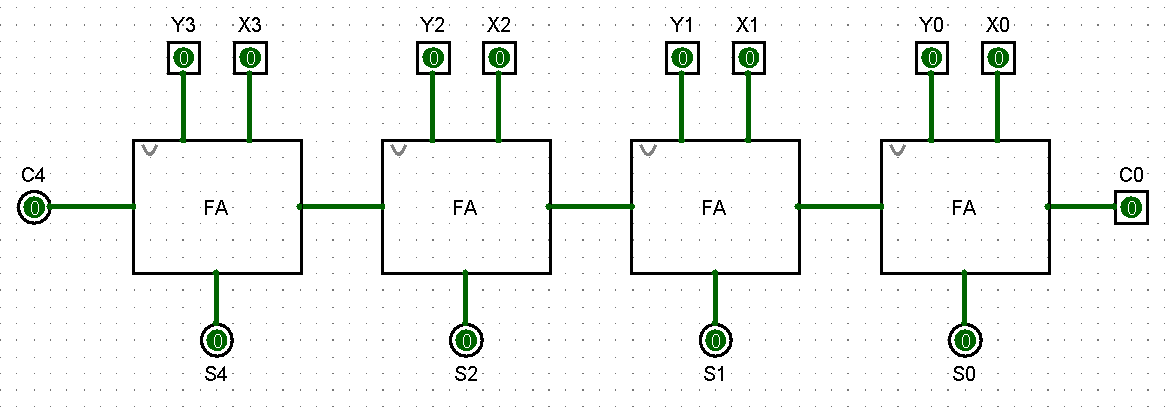
****

**行波进位加法器设计：在Logisim中，利用上一关设计的全加器FA级联设计一个4位的行波进位加法器。**

行波进位是指进位信号从低位逐位向高位传递，特点是结构简单，但速度比较慢。原理示意图如下：



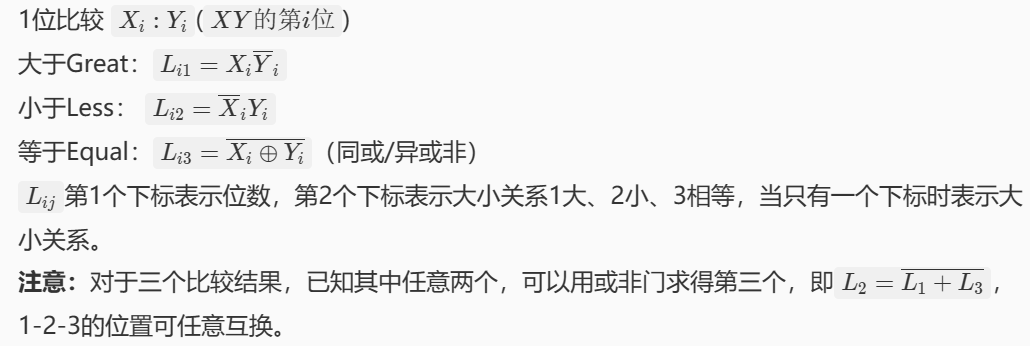
**逻辑电路图截图：**

****

**2.2 任务2：比较器设计**

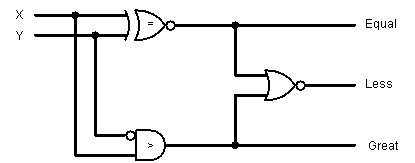
**一位比较器设计：在Logisim中完成1位比较器电路的绘制并完成测试。**

在数字电路中，经常需要对两个位数相同的二进制数进行比较，以判断它们的相对大小或者是否相等，用来实现这一功能的逻辑电路就称为数值比较器，简称比较器。

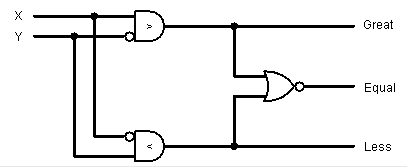
**1位比较器原理**：

**1位比较器原理图：**

原理图之一：



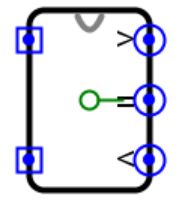
原理图之二：



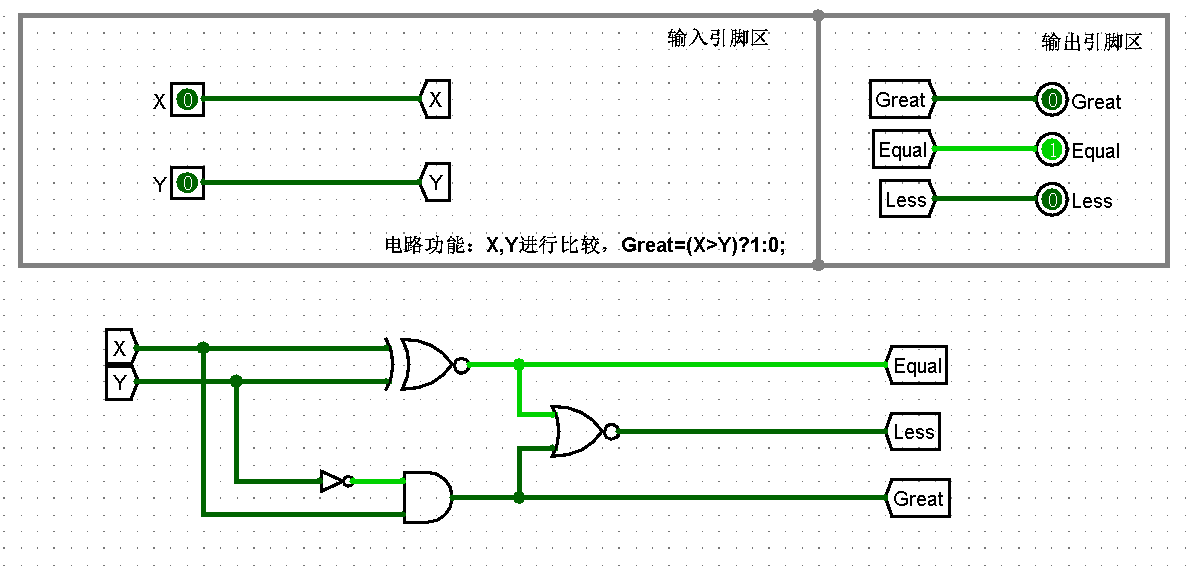
**电路引脚1**



子电路外观



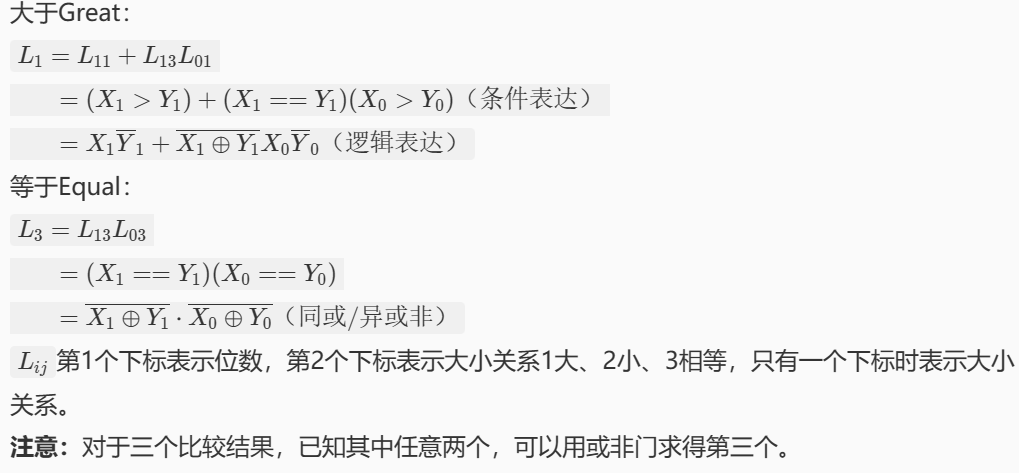
**逻辑电路图截图：**

****

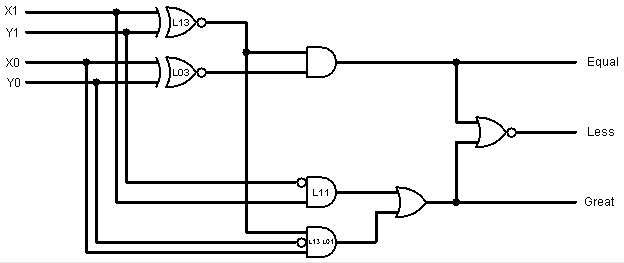
**二位比较器设计：在Logisim中完成2位比较器电路的绘制并完成测试。**

**2位比较器原理：**

2位比较 *X*1​*X*0​:*Y*1​*Y*0​，比较顺序从高位到低位，当高位大、小关系确定时则无需看低位，当高位相等时再看相邻低位的关系。



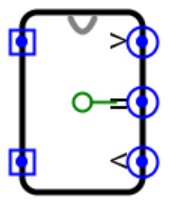
**原理图**



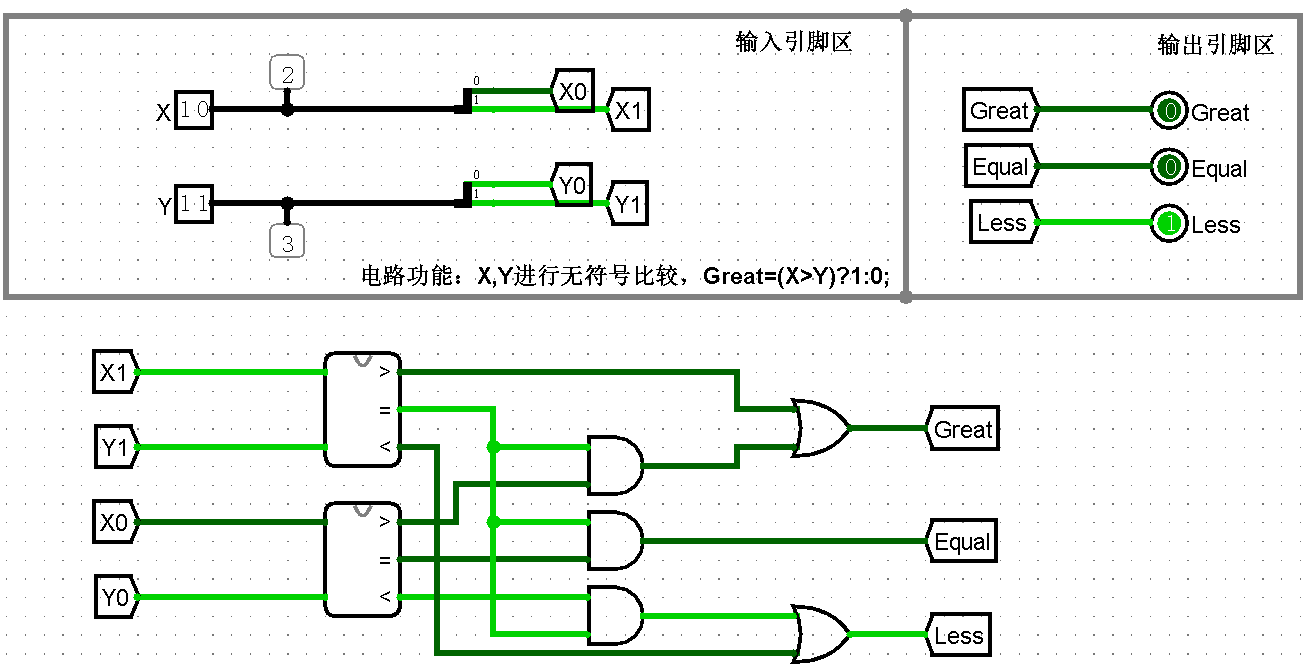
**电路引脚**



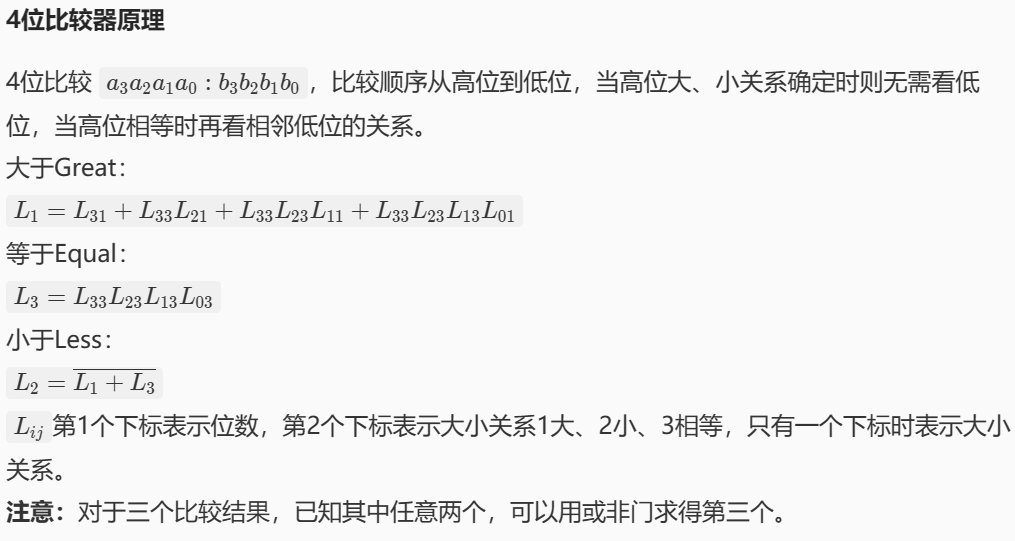
子电路外观



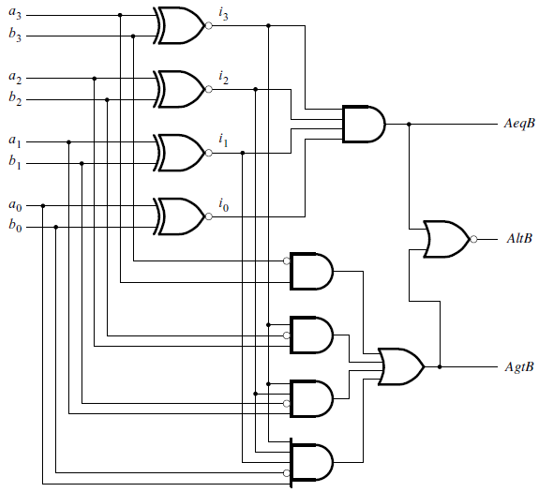
**逻辑电路图截图：**

****

**四位比较器设计：在Logisim中完成4位比较器电路的绘制并完成测试。**



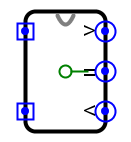
**原理图**



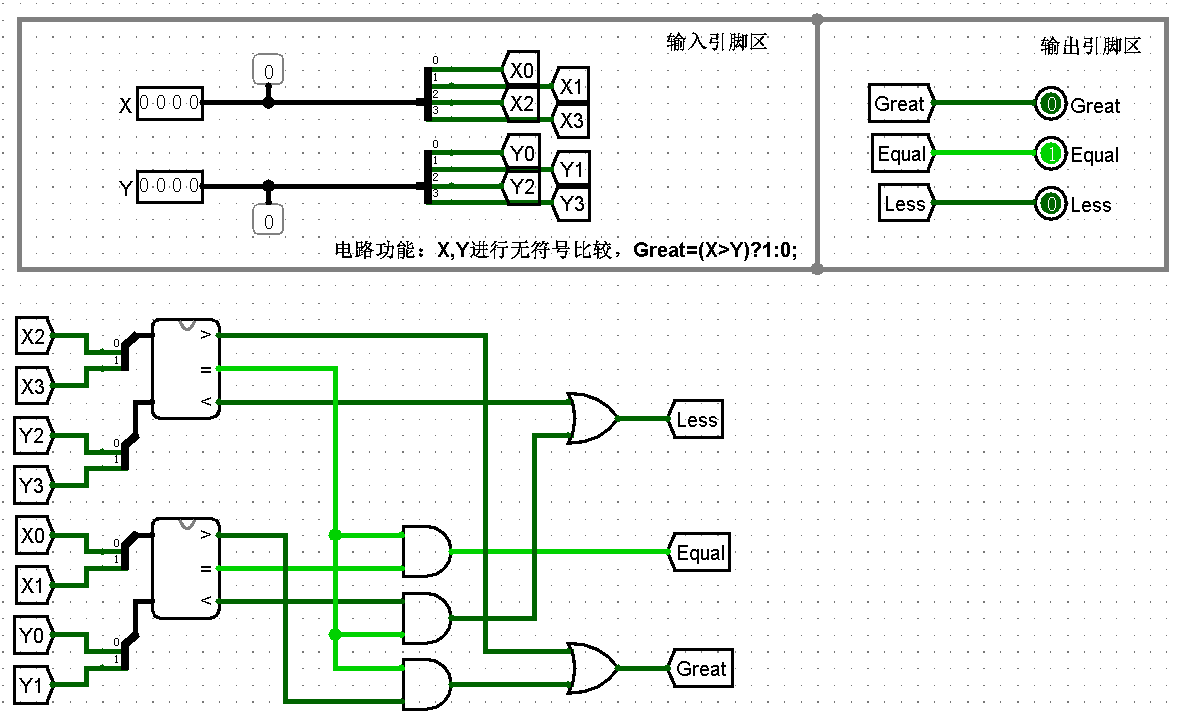
**电路引脚**



子电路外观



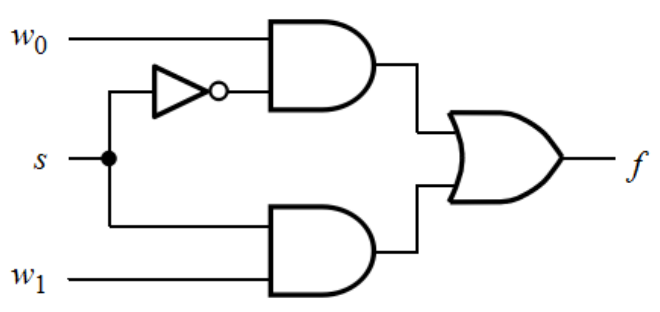
**逻辑电路图截图：**



**2.3 任务3：多路选择器设计**

**2选1多路选择器：在Logisim中绘制2选1多路选择器。**

多路选择器又称数据选择器、多路开关或复用器（MUX，Multiplexer），在多路数据传送过程中，能够根据需要将其中任意一路数据选中送到输出端的电路。 2选1多路选择器电路原理图如下：



2路选择器简化真值表如下：

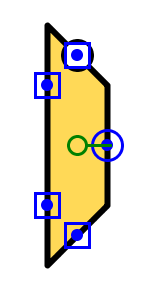
| ***s*** | ***f*** |
| --- | --- |
| 0 | *w*0​ |
| 1 | *w*1​ |

表达式：f=sw0+sw1

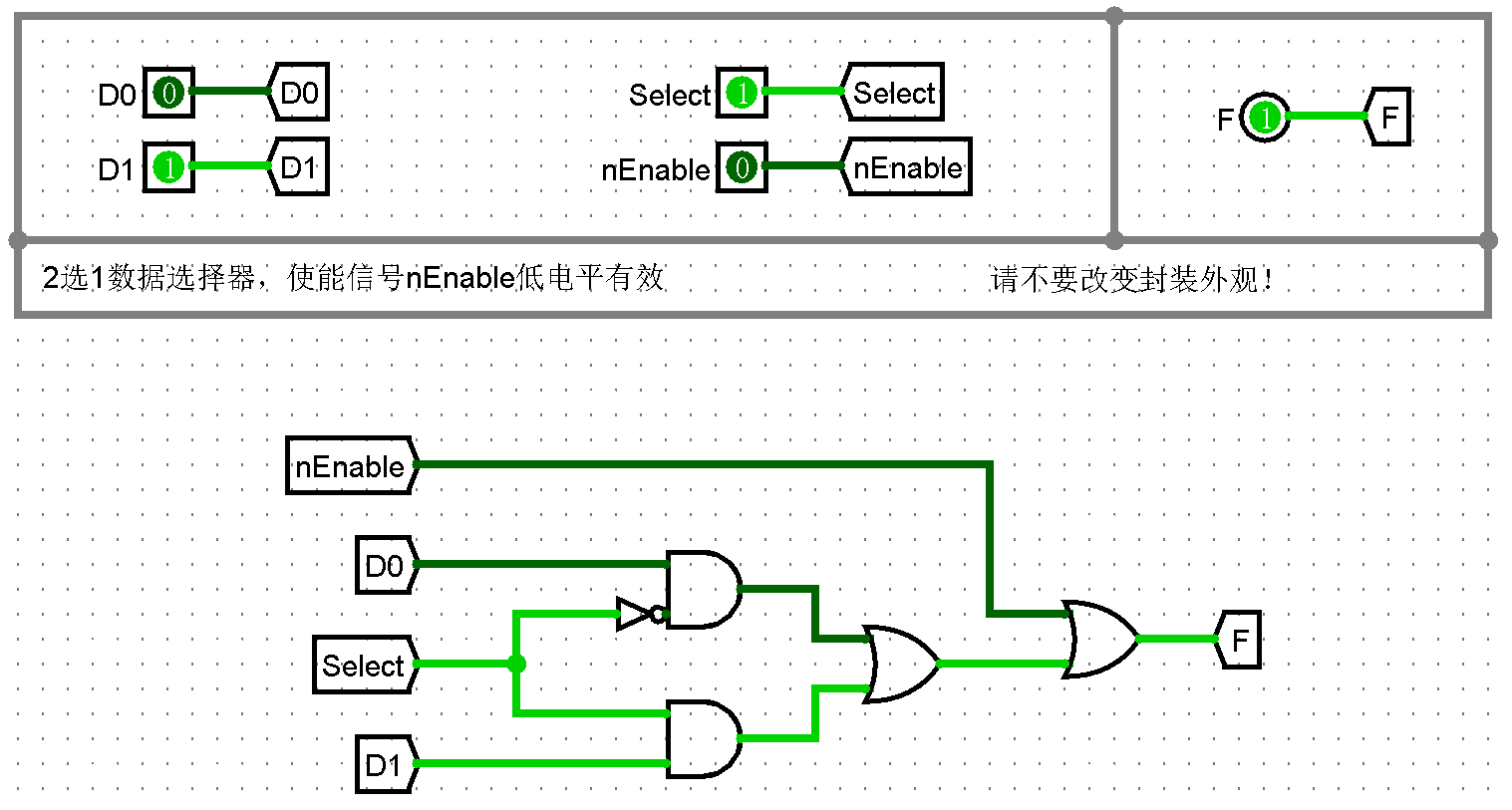
引脚说明

|  |  |  |  |
| --- | --- | --- | --- |
| **信号** | **I/O** | **位宽** | **说明** |
| nEnable | 输入 | 1 位 | 使能端，低电平有效。nEnable=0时选择；nEnable=1禁止选择，此时输出恒为1。 |
| Select | 输入 | 1 位 | 地址选择（数据端口选择） |
| D1,D0 | 输入 | 1 位 | 2路数据输入 |
| F | 输出 | 1 位 | 数据输出端 |

子电路外观

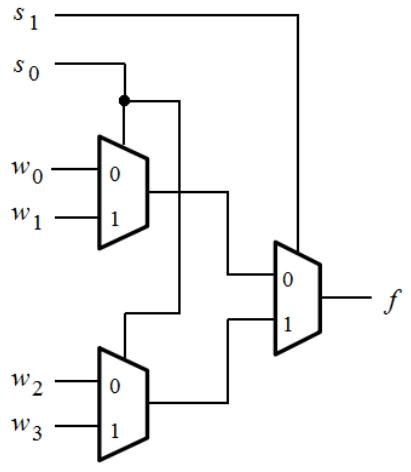


**逻辑电路图截图：**

****

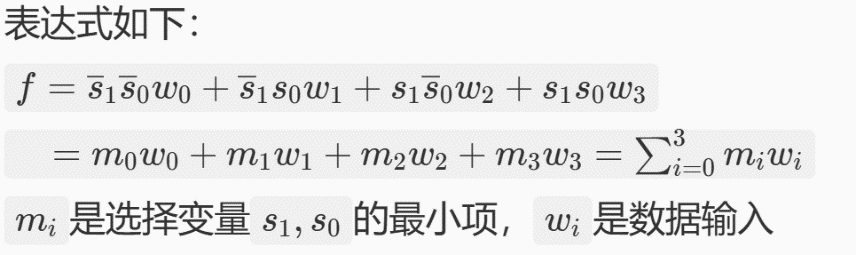
**4选1多路选择器：在Logisim中利用上一关的2选1选择器级联设计一个4选1多路选择器。**

多路选择器又称数据选择器、多路开关或复用器（MUX，Multiplexer），在多路数据传送过程中，能够根据需要将其中任意一路数据选中送到输出端的电路。 4选1选择器级联扩展原理如下：



4路选择器简化真值表如下：

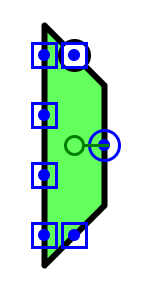
|  |  |  |
| --- | --- | --- |
| ***s*1​** | ***s*0​** | ***f*** |
| 0 | 0 | w0 |
| 0 | 1 | w1 |
| 1 | 0 | w2 |
| 1 | 1 | w3 |



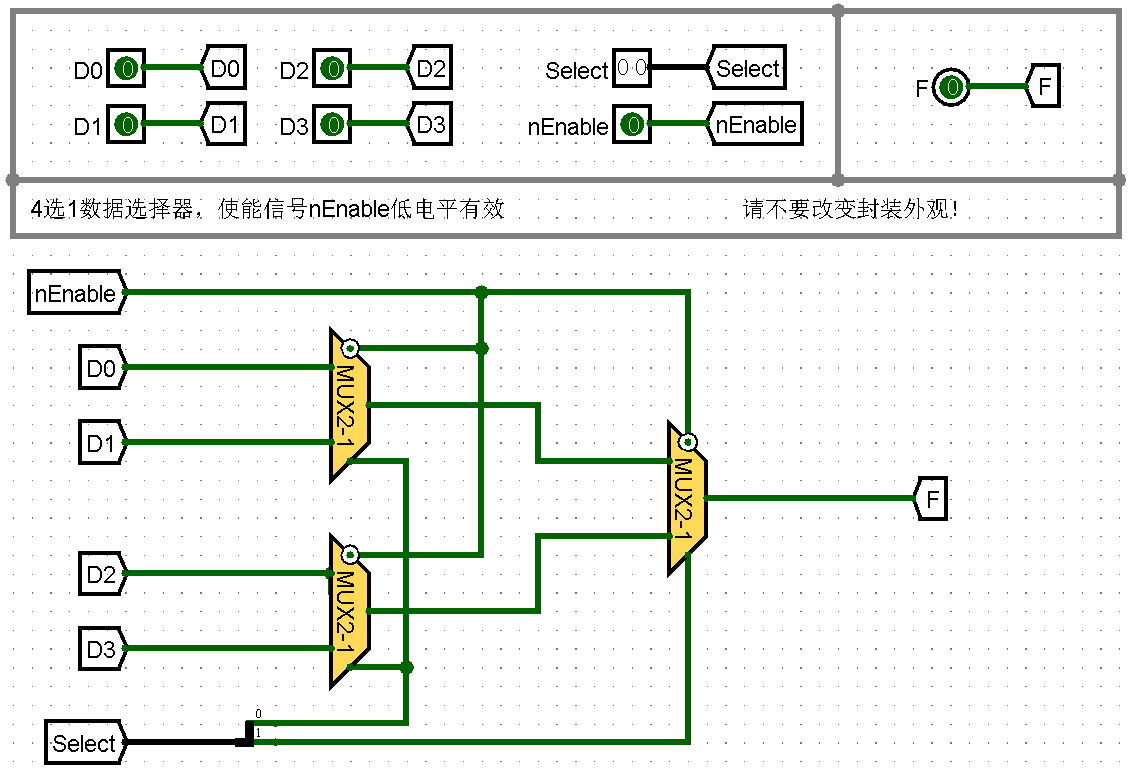
引脚说明

|  |  |  |  |
| --- | --- | --- | --- |
| **信号** | **I/O** | **位宽** | **说明** |
| nEnable | 输入 | 1 位 | 使能端：nEnable=0时选择；nEnable=1禁止选择，此时输出恒为1。 |
| Select | 输入 | 2位 | 地址选择（数据端口选择） |
| D3~D0 | 输入 | 1位 | 4路数据输入 |
| F | 输出 | 1 位 | 数据输出端 |

子电路外观



**逻辑电路图截图：**

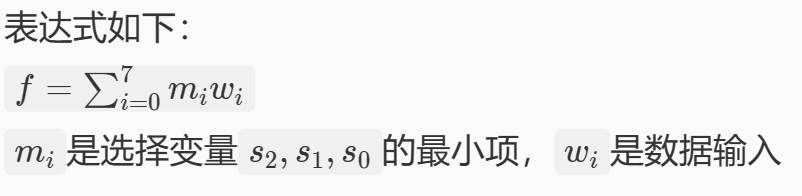
****

**8选1多路选择器：利用前面两关的选择器级联设计一个8选1多路选择器。**

多路选择器又称数据选择器、多路开关或复用器（MUX，Multiplexer），在多路数据传送过程中，能够根据需要将其中任意一路数据选中送到输出端的电路。

8路选择器简化真值表如下：

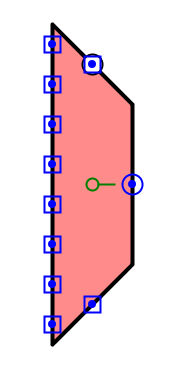
|  |  |  |  |
| --- | --- | --- | --- |
| ***s*2​** | ***s*1​** | ***s*0​** | ***f*** |
| 0 | 0 | 0 | w0 |
| 0 | 0 | 1 | w1 |
| 0 | 1 | 0 | w2 |
| 0 | 1 | 1 | w3 |
| 1 | 0 | 0 | w4 |
| 1 | 0 | 1 | w5 |
| 1 | 1 | 0 | w6 |
| 1 | 1 | 1 | w7 |



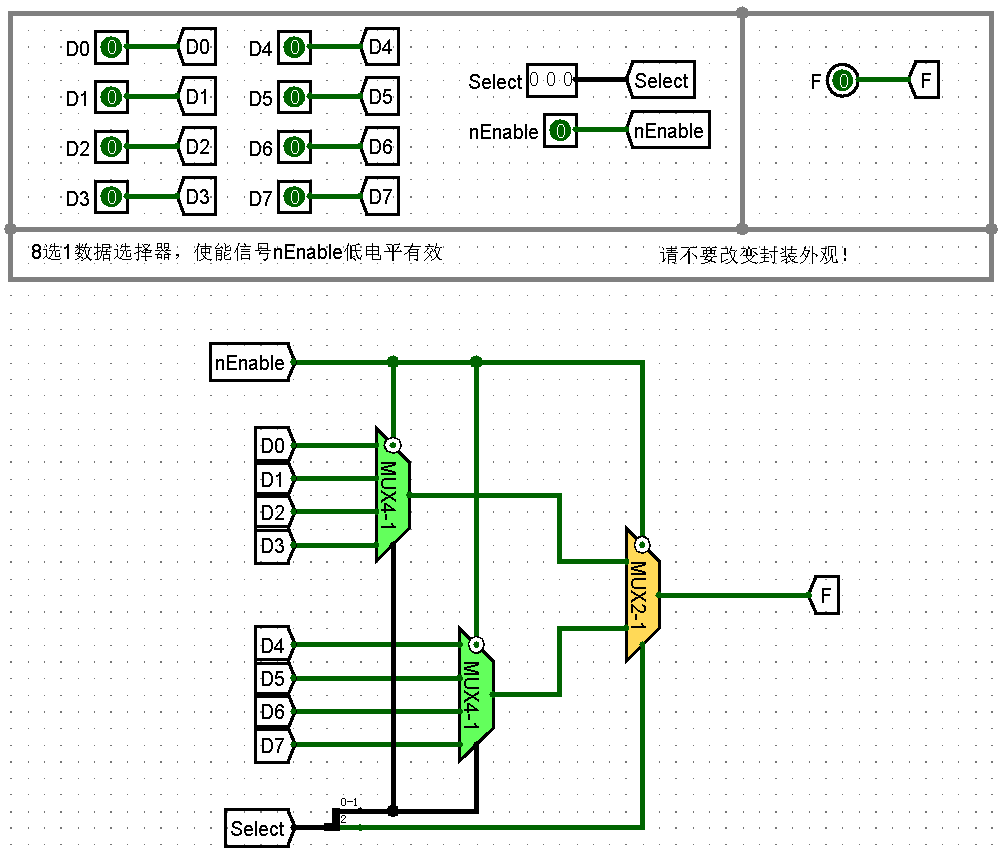
引脚说明

|  |  |  |  |
| --- | --- | --- | --- |
| **信号** | **I/O** | **位宽** | **说明** |
| nEnable | 输入 | 1 位 | 使能端：nEnable=0时选择；nEnable=1禁止选择，此时输出恒为1。 |
| Select | 输入 | 3位 | 地址选择（数据端口选择） |
| D7~D0 | 输入 | 8 位 | 8路数据输入 |
| F | 输出 | 1 位 | 数据输出端 |

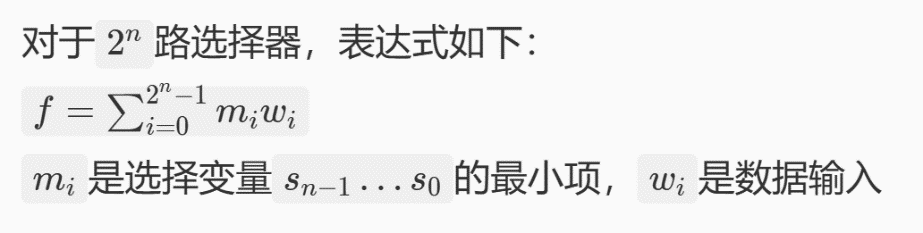
子电路外观



**逻辑电路图截图：**

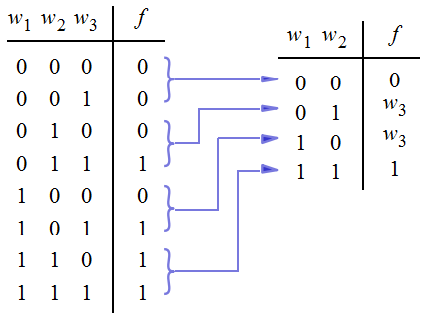
****

**多路选择器应用：在Logisim中选择在设计的多路选择器进行组合逻辑设计，设计全加器FA。**

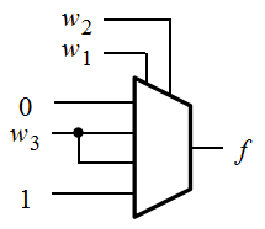


**真值表变换**

原理：将待实现函数的真值表变换成对应多路选择器的简化真值表形态即可！ 例如：用4选1MUX实现3输入择多函数f(w1​,w2​,w3​)=Σm(3,5,6,7)

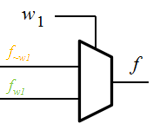


用4选1MUX实现如下：



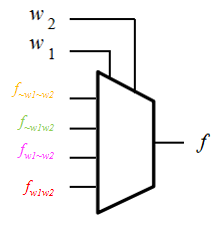
*w*1​,*w*2​作为选择变量*s*1​,*s*0​，将0、*w*3​、*w*3​和1分别接入0号、1号、2号和3号数据端口。注意函数变量与选择变量的对应顺序！！ 若用8选1MUX实现3输入函数，则真值表无需做任何变换！ 此时，*w*1​,*w*2​,⋯,*w*8​作为选择变量*s*7​,*s*6​,⋯,*s*0​，将8个函数值分别送入对应数据端口。

**香农展开（1变量展开）**

任何*n*变量函数*f*(*w*1​,*w*2​,⋯,*wn*​)可以对*w*1​展开表示为： *f*=*w*1​⋅*f*(0,*w*2​,⋯,*wn*​)+*w*1​⋅*f*(1,*w*2​,⋯,*wn*​) =*w*1​⋅*fw*1​​+*w*1​⋅*fw*1​​ 对任意*wi*​展开都有效！ 此时可以用2选1MUX实现该函数，如图所示： *w*1​作为选择变量*s*，将*fw*1​​和*fw*1​​的逻辑实现分别接入0号和1号数据端口。

**香农展开（2变量展开）**

任何*n*变量函数*f*(*w*1​,*w*2​,*w*3​,⋯,*wn*​)可以对*w*1​,*w*2​展开表示为： *f*=*w*1​*w*2​⋅*f*(0,0,*w*3​,⋯,*wn*​)+*w*1​*w*2​⋅*f*(0,1,*w*3​,⋯,*wn*​) +*w*1​*w*2​⋅*f*(1,0,*w*3​,⋯,*wn*​)+*w*1​*w*2​⋅*f*(1,1,*w*3​,⋯,*wn*​) =*w*1​*w*2​⋅*fw*1​*w*2​​+*w*1​*w*2​⋅*fw*1​*w*2​​+*w*1​*w*2​⋅*fw*1​*w*2​​+*w*1​*w*2​⋅*fw*1​*w*2​​ 对任意*wi*​,*wj*​(*i*​=*j*)展开都有效！ 此时可以用4选1MUX实现该函数，如图所示：

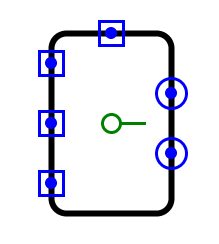


*w*1​,*w*2​作为选择变量*s*1​,*s*0​，将*fw*1​*w*2​​、*fw*1​*w*2​​、*fw*1​*w*2​​和*fw*1​*w*2​​的逻辑实现分别接入0号、1号、2号和3号数据端口。 依此类推，可以对任意的*k*个变量展开(*k*≤*n*)，然后用2*k*选1MUX实现该函数！当*k*=*n*时，展开式为函数的正则SOP式（标准与或式）。

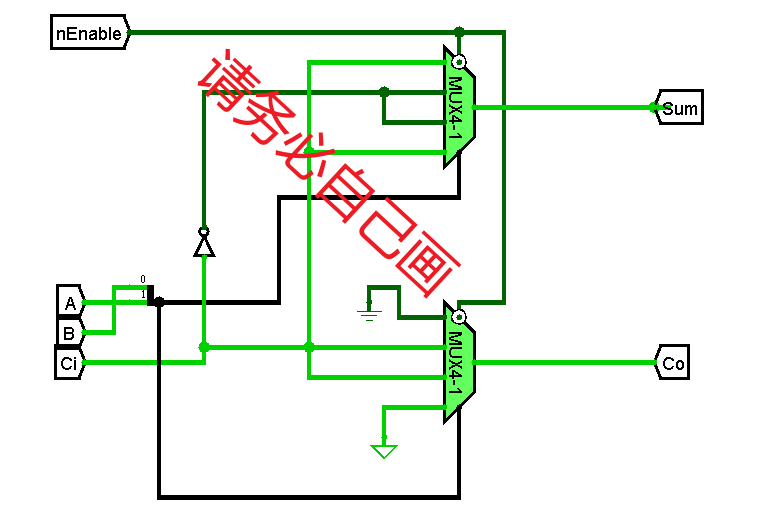
**引脚说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **信号** | **I/O** | **位宽** | **说明** |
| nEnable | 输入 | 1位 | 使能端：nEnable=0时做加法；nEnable=1禁止相加，此时输出恒为1。 |
| A,B,Ci | 输入 | 1位 | 三个相加数 |
| Co,Sum | 输出 | 1位 | 进位输出、和输出 |

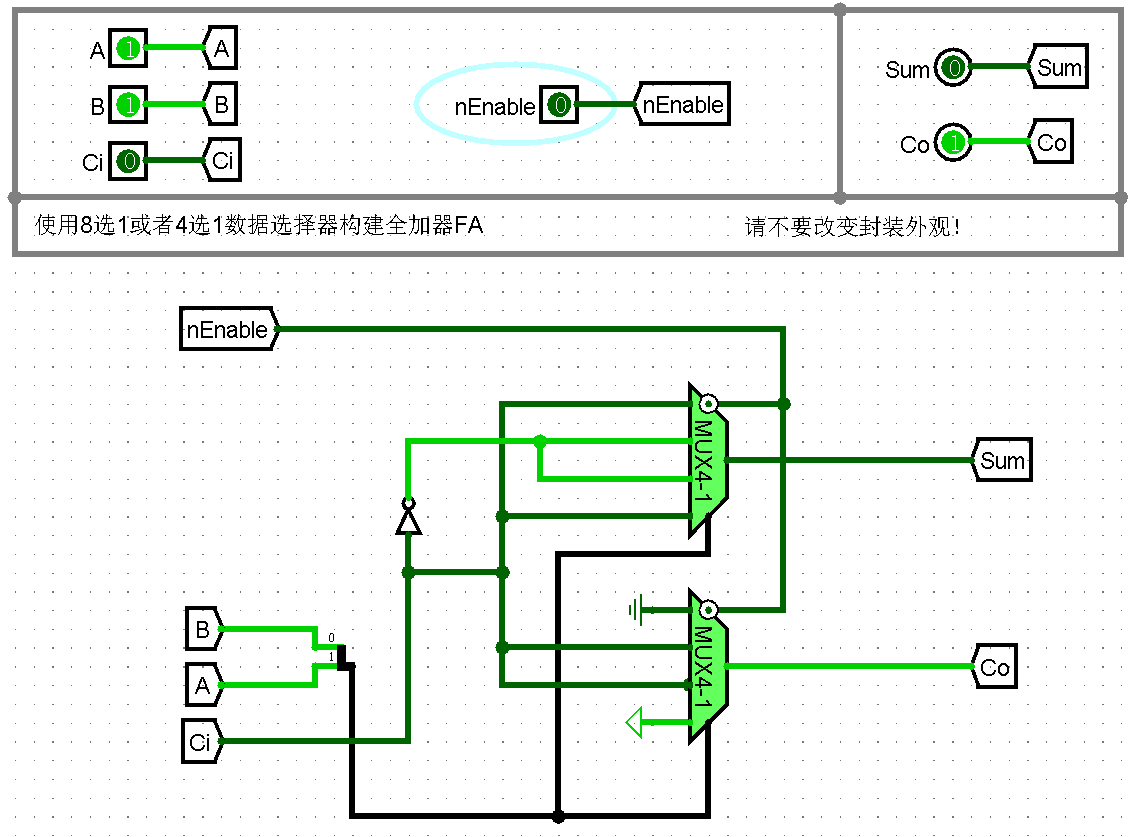
子电路外观



**参考实现：**



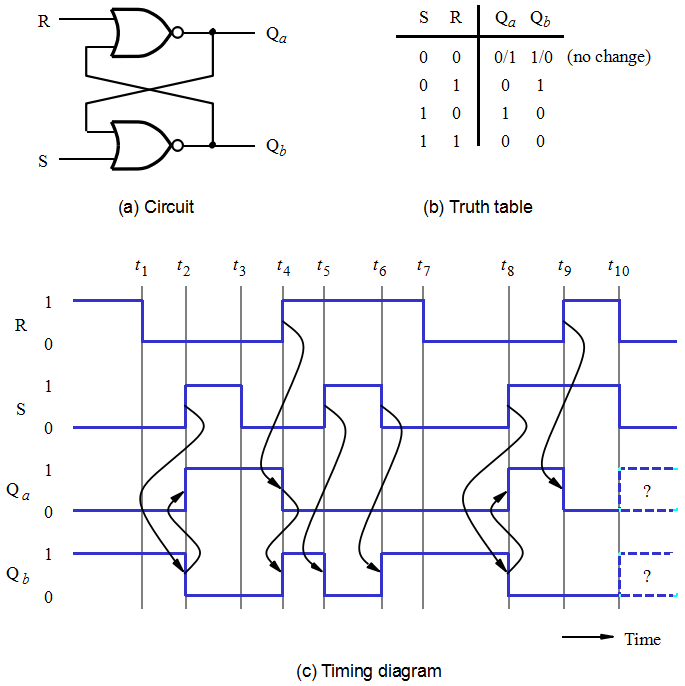
**逻辑电路图截图：**

****

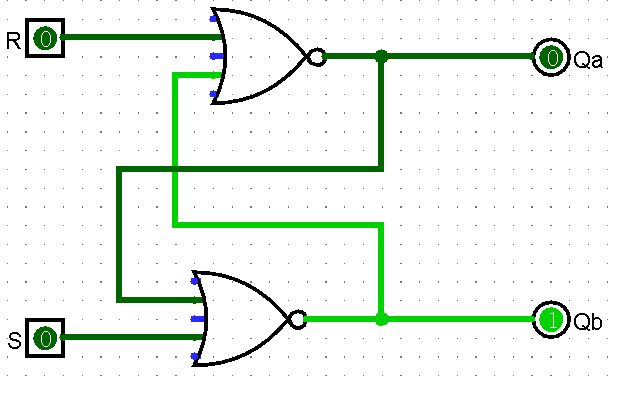
### 3.Logisim时序电路设计

**基本锁存器工作原理和设计：在Logisim中，构建由两个或非门构成的基本SR锁存器。**

锁存器(Latch)是一种对脉冲电平敏感的存储单元电路，它们可以在特定输入脉冲电平作用下改变状态。锁存，就是把信号暂存以维持某种电平状态。锁存器的最主要作用是缓存。锁存器的结构特点是交叉耦合（反馈），这是形成记忆能力的关键。 由或非门构成的基本SR锁存器原理如下：

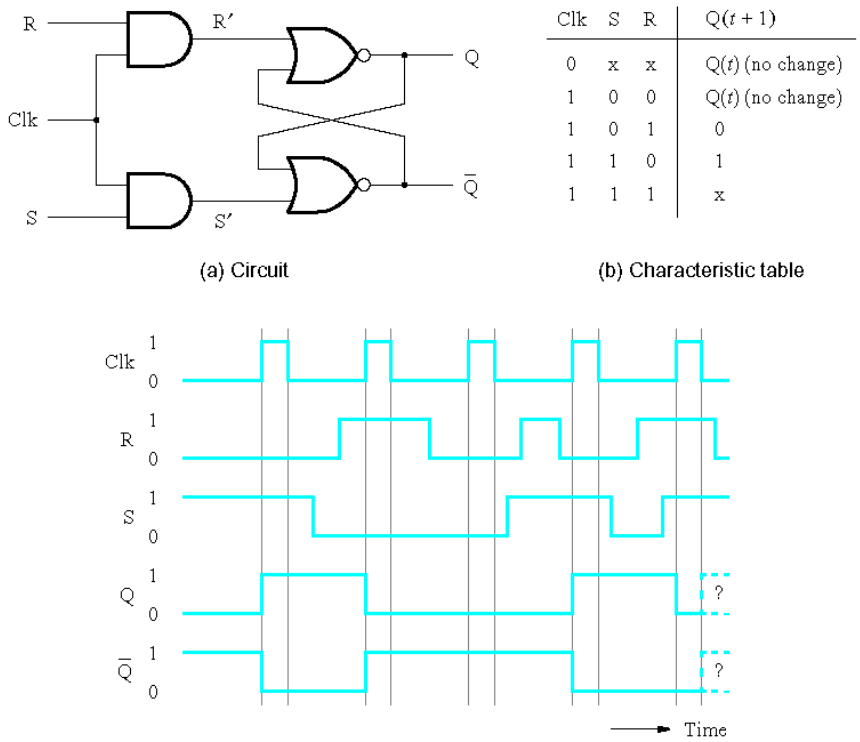


**逻辑电路图截图：**

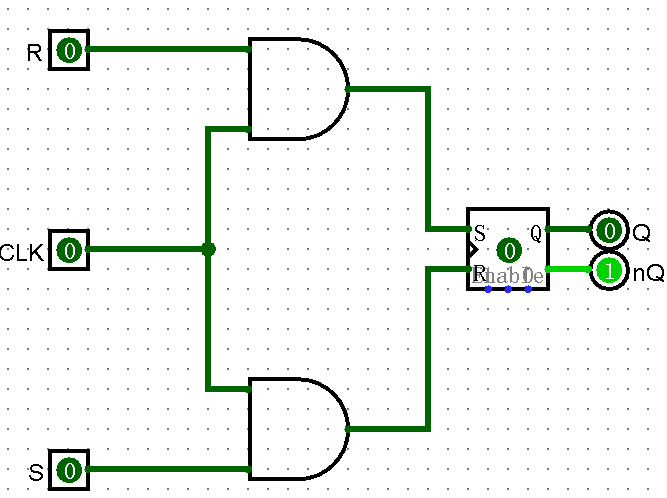
****

**门控SR锁存器：在Logisim中，在基本锁存器的基础上构建门控SR锁存器。**

锁存器(Latch)是一种对脉冲电平敏感的存储单元电路，它们可以在特定输入脉冲电平作用下改变状态。锁存，就是把信号暂存以维持某种电平状态。锁存器的最主要作用是缓存。锁存器的结构特点是交叉耦合（反馈），这是形成记忆能力的关键。 门控SR锁存器原理如下，Clk被称为使能或者时钟信号。

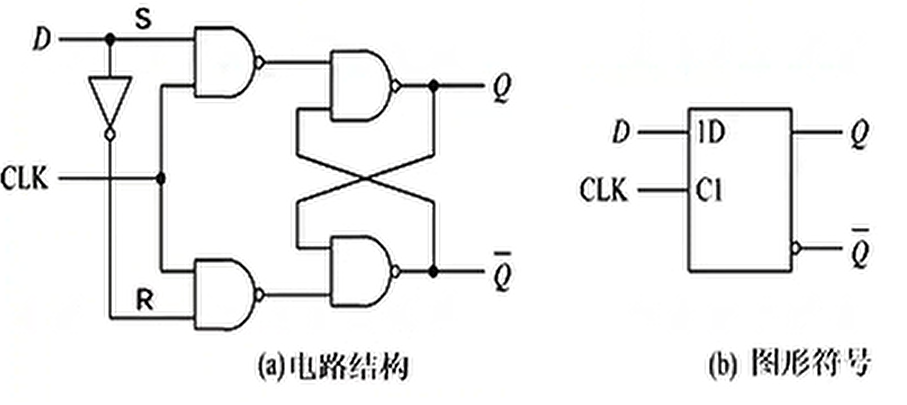


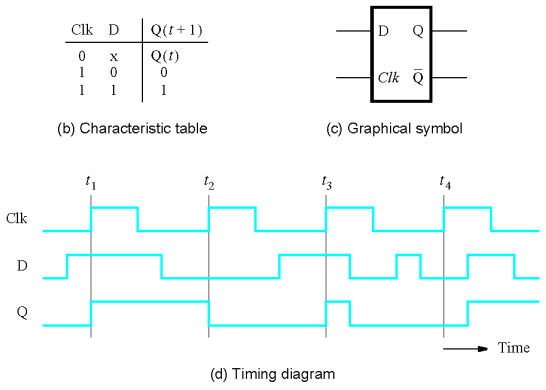
**逻辑电路图截图：**



**门控D锁存器：在Logisim中，用四个与非门构建门控D锁存器。**

锁存器(Latch)是一种对脉冲电平敏感的存储单元电路，它们可以在特定输入脉冲电平作用下改变状态。锁存，就是把信号暂存以维持某种电平状态。锁存器的最主要作用是缓存。锁存器的结构特点是交叉耦合（反馈），这是形成记忆能力的关键。 与非门构成的门控SR锁存器原理如下，在此基础上只需添加一根连线即可构建门控D锁存器。

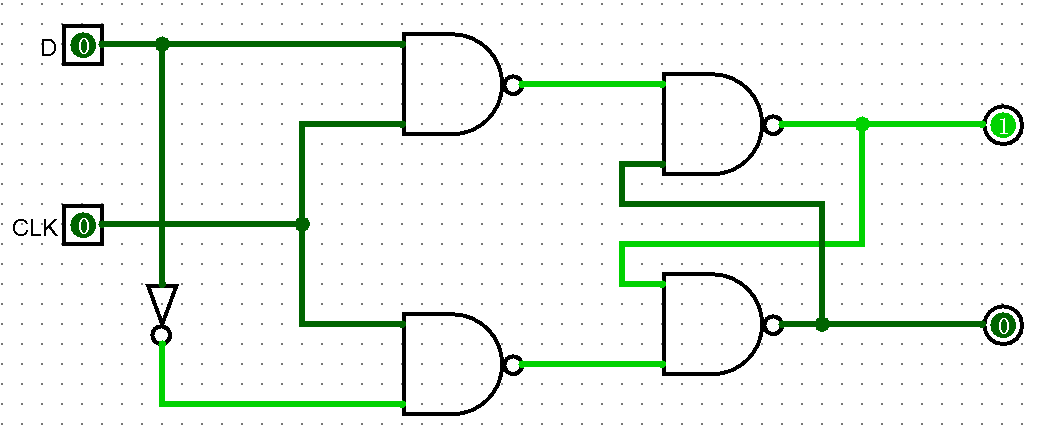




D锁存器的特征方程如下：

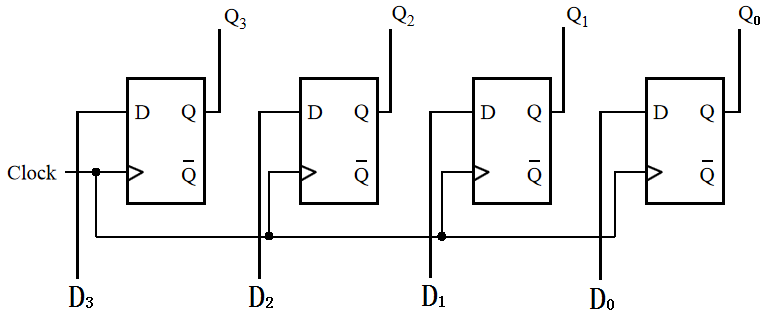
*Q*(*t*+1)=*D* Clk = 1有效

**逻辑电路图截图：**

****

**基本寄存器：在Logisim中，构建由4个D触发器构成的4位基本寄存器，要求上升沿触发，具有异步复位功能。**

寄存器（Register）的功能是存储二进制代码，它是由具有存储功能的触发器组合起来构成的。一个触发器可以存储1位二进制代码，故存放n位二进制代码的寄存器，需用n个触发器来构成。 基本寄存器（Basic Register）的结构简单，只能并行送入数据，也只能并行输出，如下图所示。

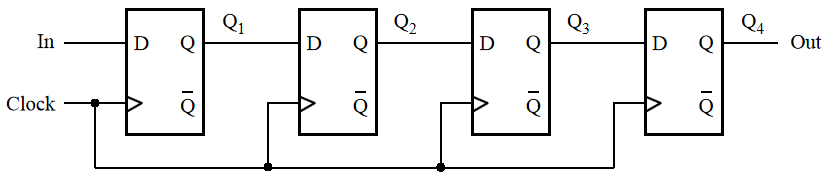


**引脚说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **信号** | **I/O** | **位宽** | **说明** |
| Reset | 输入 | 1 位 | 异步复位端：Reset=1时复位，Q=0 |
| Clock | 输入 | 1 位 | 时钟脉冲，上升沿有效 |
| Din | 输入 | 4 位 | 并行数据输入端 |
| Q | 输出 | 4 位 | 并行数据输出端：Reset=0、Clock↑时Q=Din |

**移位寄存器：在Logisim中，构建由4个D触发器构成的4位右移移位寄存器，要求上升沿触发，具有异步复位功能。**

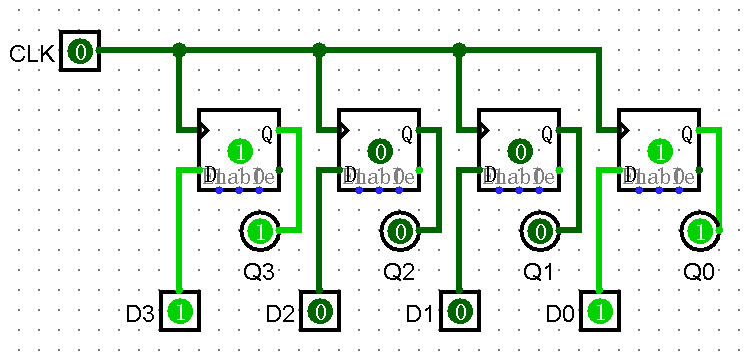
寄存器（Register）的功能是存储二进制代码，它是由具有存储功能的触发器组合起来构成的。一个触发器可以存储1位二进制代码，故存放n位二进制代码的寄存器，需用n个触发器来构成。 移位寄存器（Shift Register）的结构也比较简单，其中的数据可以在移位脉冲作用下依次逐位右移或左移，如下图所示。

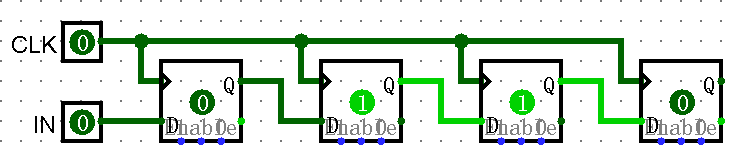


**引脚说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **信号** | **I/O** | **位宽** | **说明** |
| Reset | 输入 | 1 位 | 异步复位端：Reset=1时复位，Out=0 |
| Clock | 输入 | 1 位 | 时钟脉冲，上升沿有效 |
| In | 输入 | 1 位 | 右移数据输入端 |
| Out | 输出 | 1 位 | 右移数据输出端 |

**逻辑电路图截图：**

****



## 四、收获感想

**记录实验感受、操作过程中遇到的困难及解决办法、遗留的问题、意见和建议等。**