

计算机硬件基础课程设计-----P1 课下指导书

1.1 走进 Logisim

:)首先恭喜各位同学专业分流中成功进入软件学院，开启全新的大二生活。

学习软件开发，你可能第一时间想到的是：**C 语言、Java、Python、数据结构、前端后端、算法等等**，没错，这些技术都是软件开发的关键技术，这些将来大家都会学习。然而，想成为一名优秀的软件工程师，仅仅掌握软件方面的知识，是完全不够的。

如果我们把一台计算机必做一个人，那么硬件就是它的肉体，软件就是它的灵魂；更进一步讲，CPU（中央处理器）是硬件中最重要的组成部分，是计算机的心脏（或者说大脑）；操作系统是软件中最重要最基本的组成部分，是计算机的“人格”。软件和硬件谁也离不开谁，如果想要学好软件开发，首先就需要对计算机的硬件，尤其是 CPU 有一个深刻的了解，我们这门课程开设以及改革的目的是如此，希望大家能真正了解 CPU 并且自己动手实现一个 CPU。听起来很吓人是吧？实则不然，只要认真学习，做一个 CPU 是非常简单的。

上面对 CPU 的介绍还是有点抽象，CPU 究竟是怎么运作的？这里简单介绍一下：举个最简单的例子，我写了一段 C 代码，现在要把它编译成程序并运行，这个过程是什么样的？计算机肯定是不理解你写的这些东西：

```
#include<stdio.h>
//老天爷保佑我的代码顺利 AC!
int main(void)
{
    balabala...//此处省略若干行
    return 0;
}
```

但是编译器理解啊，它会把这些高级语言代码翻译成汇编语言代码，大概长这个样子：

`lui $t0,$t0,0xf00f`//把 t0 寄存器的值或上 0xf00f 后赋给 t0 寄存器

`add $t1,$s1,$s2`//把 s1 寄存器的值和 s2 寄存器的值相加然后赋给 t1 寄存器

.....

可以看出，每一条汇编语言代码都是一个比较基本的，面向硬件的操作指令，针对我们即将学习的 MIPS-lite2 指令集，每一条汇编语言代码都对应一段 32 位等长机器码（说白了就是一个 32 位的 0、1 串），把这个机器码传给 CPU，这时候它就理解自己需要做什么了，就可以执行一条指令。所有的指令执行完毕后，你的程序也就执行完毕了。

简单来讲，CPU 就是一个执行汇编语言指令的东西。

自己动手写一个小 CPU 肯定是需要工具的，初学者推荐使用 Logisim，你可以用它画出电路图，并且模拟运行。

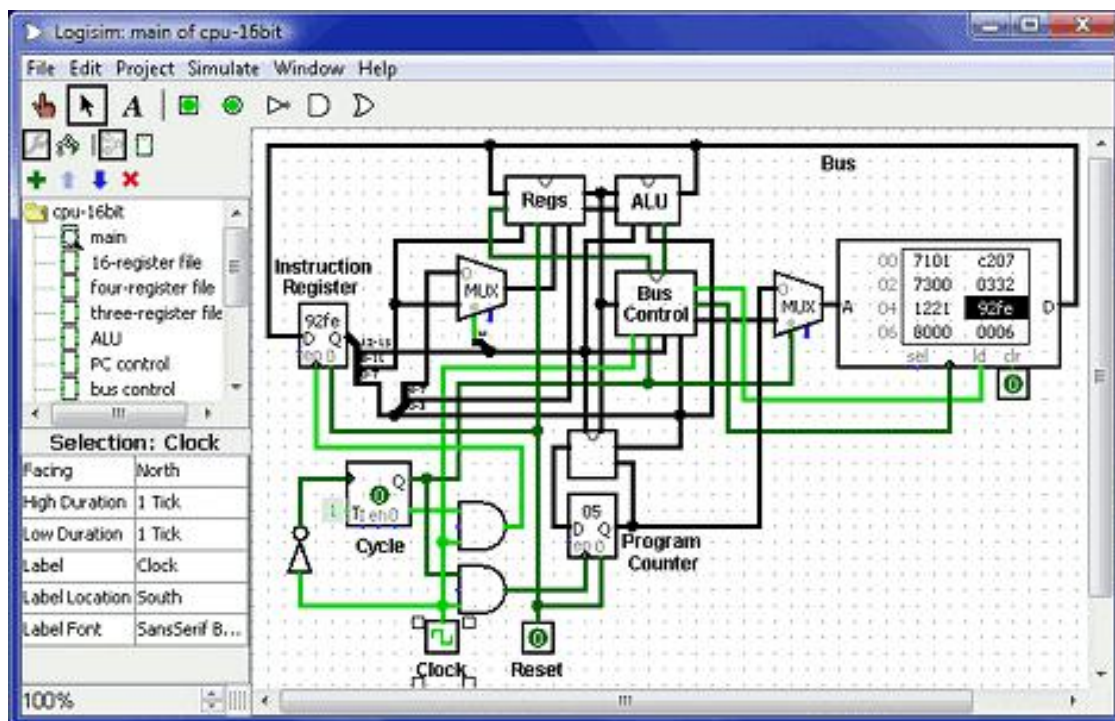


图 1-1 Logisim 界面

当然工业生产中肯定不能用 Logisim 这种工具来设计 CPU，现代实际生产中一般采用 VerilogHDL（这个貌似最受欢迎）和 VHDL 两种硬件描述语言来进行开发设计。硬件描述语言不是编程语言，你写出的代码，不是一段程序，而是一段电路逻辑，或者说，你正在用写代码的方式画电路图。相比 Logisim，硬件描述语言可能没那么直观，但是简洁、方便，便于设计复杂电路。**需要指出的是：本门课程《计算机硬件基础》不要求大家掌握 VerilogHDL 和 VHDL，只要求掌握 Logisim。**感兴趣的同学可以自行查阅资料了解，或者参考计算机学院开设的《计算机组成原理》课程指导书了解学习，但是我们的实验课程里完全不涉及 VerilogHDL 和 VHDL。

oh，忘了介绍一下实验要求了。

本章的题目是：《简单部件与状态机设计实验》，本章不要求大家掌握 CPU 设计，只要求大家学会 Logisim 的基本使用，简单电路设计以及有限状态机的设计。你需要在本周完成以下任务：

- (1) 阅读指导书学习相关知识
- (2) 独立自主完成指导书中的课下测试题目（一共五道，**课下测试 1~5** 为必做，不完成不能参加课上测试）
- (3) 实验课现场独立完成课上测试题目

请大家一定要独立自主完成，不要抄袭！有问题可以询问老师、同学、助教。只有自己认真学习，才能顺利通过课上测试。抄袭的话，骗得了自己，骗不了别人，课上测试就会不知所措导致实验失败。另外，评测系统会对大家的课下、课上作业进行查重！

（请大家注意，修改电路外观，挪动元件位置这种无聊的办法是不能骗过查重系统的！就好比 c 语言改个变量名，查重系统是在汇编语言代码层面查重的，这种无聊的小伎俩什么用都没有；同样，我们的 Logisim 查重程序也不会简单地通过电路元件位置来判断是否涉嫌抄袭，我们有更深层次的方法，一言以蔽之：

抄袭必被抓！之前的 pre 测试已经有两名同学用上述方法企图欺骗评测机被约谈。
大家一定要记住，自己做，才能学到真本事，抄来的，永远不是你自己的！)

1.2 获取 Logisim

Logisim 需要 java 环境的支持，大家需要安装 JDK15.0.2 的版本。

(如果你不想使用下面的链接下载需要的文件，没关系，下面所需的所有文件我已经帮你打包好了。你可以前往课程中心下载，或者使用北航云盘链接：

<https://bhpan.buaa.edu.cn:443/link/A96230834ED88C18B0D179A3ECC71250>

有效期限：2022-09-16 22:00)

大家可以在官网上下下载 java，你也可以选择其它网站：

<https://www.oracle.com/java/technologies/javase/jdk15-archive-downloads.html>

下载完毕后，请参照教程完成 java 环境的安装，这里推荐这个教程，你也可以参照其他教程：<https://www.runoob.com/java/java-environment-setup.html#win-install>

如果配置成功，打开命令提示符窗口，输入 `java -version` 并按下回车，将会看到如图 2-1 所示的情景。

```
C:\Users\lcf20>java -version
java version "15.0.2" 2021-01-19
Java(TM) SE Runtime Environment (build 15.0.2+7-27)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

图 2-1 Java 环境配置成功

Logisim 下载地址：<https://sourceforge.net/projects/circuit/>

Logisim 官网：<http://www.cburch.com/logisim/>

Logisim 官方指导文档（英文）：

<http://www.cburch.com/logisim/docs/2.7/en/html/guide/tutorial/index.html>

<http://www.cburch.com/logisim/docs/2.7/en/html/guide/index.html>

<http://www.cburch.com/logisim/docs/2.7/en/html/libs/index.html>

Logisim 中文指导手册（强烈建议在开始实验之前阅读一下，掌握 Logisim 基本元件的使用）：[LogiSim 用户手册（中文） - 道客巴巴 \(doc88.com\)](#)

参考书目：《数字设计和计算机体系结构》机械工业出版社 《Digital Design and

Computer Architecture》David Money Harris, Sara L. Harris 第 1, 2, 3,

5 章

《DIGITAL DESIGN Principles and Practices》高等教育出版社 《数字

设计——原理与实践》John F. Wakerly 第 1, 2, 3, 4, 6, 7, 8 章

1.3 Logisim 门电路

门电路是最基本的数字电路元件，以下是几个常见的 Logisim 门电路元件的简介

(1) 或门

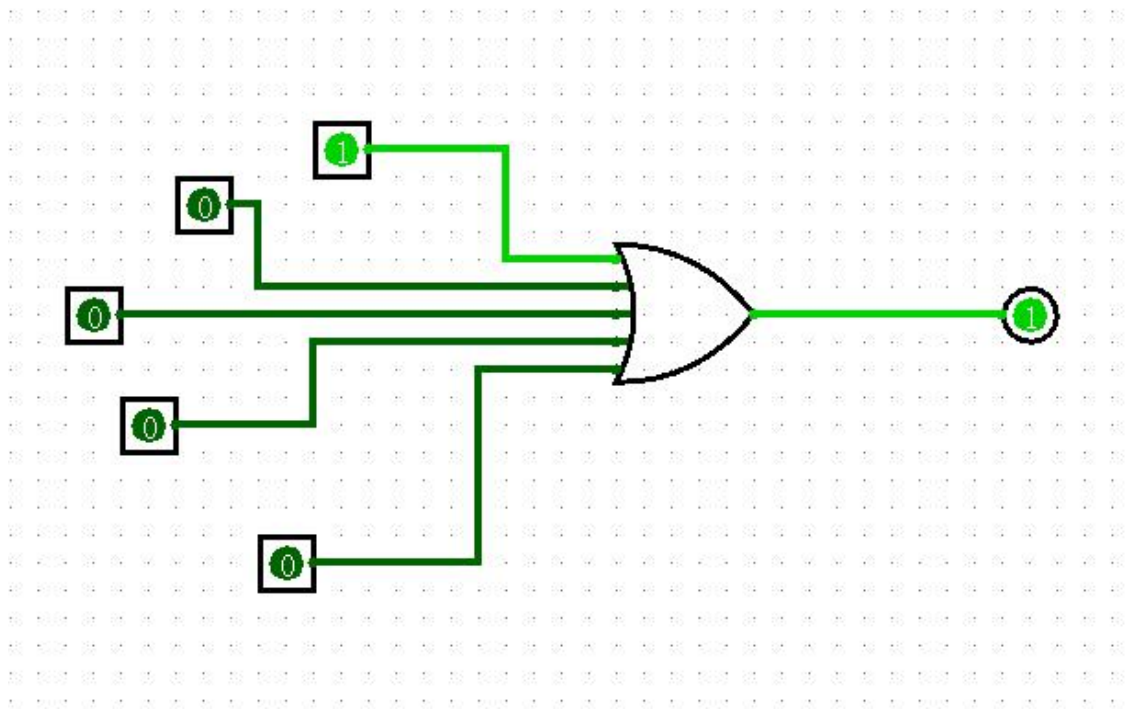


图 3-1 Logisim 中的或门

如图 3-1 所示为一个 1 位 5 输入的或门，满足逻辑或的运算规则。

我们还可以自定义或门元器件的输入端数量、位宽等信息。

Selection: OR Gate	
Facing	East
Data Bits	1
Gate Size	Medium
Number Of Inputs	5
Output Value	0/1
Label	
Label Font	SansSerif Plain 12
Negate 1 (Top)	No
Negate 2	No
Negate 3	No
Negate 4	No
Negate 5 (Bottom)	No

图 3-2 或门设置面板

左侧的控制面板就可以对这些信息进行修改，例如 Data Bits 指的是元件的位宽，Gate Size 指与门图形大小，Label 是标签，用于标记（给你的元件起名），等等。

(2) 与门

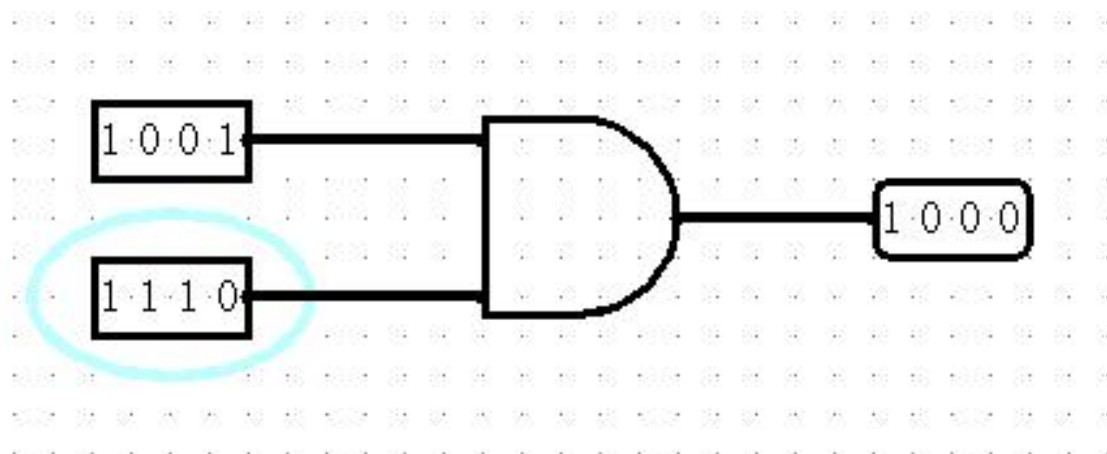


图 3-3 Logisim 中的与门

如图 3-3 所示，这是一个 4 位 2 输入的与门，进行按位与运算。

(3) 非门

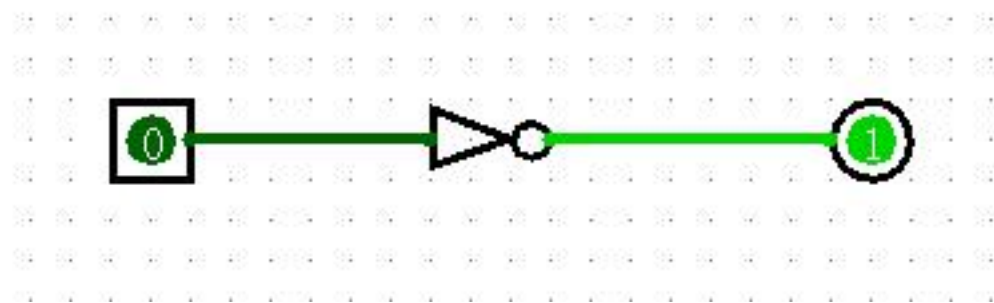


图 3-4 Logisim 中的非门

此外还有或非门、与非门等等门电路元件可以使用。请参照中文的指导手册学习。

基本的门电路元件就介绍到这里，下面我们来练习一下：

课下测试 1

知识点：Logisim 门电路的使用

难度：非常简单

$$Y = \overline{A}B + C + AD$$

一个简单的逻辑表达式，请你使用 Logisim 电路将其实现出来。

提交要求

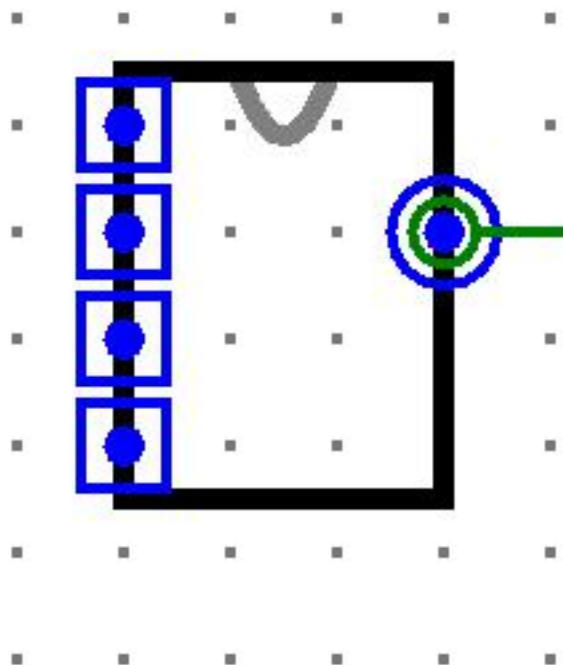
- (1) 输入：A、B、C、D
- (2) 输出：Y
- (3) 输入输出均为 1bit 数据。
- (4) 请将文件命名为 test1.circ，文件内主模块命名为 test1
- (5) 如果你评测不通过，请首先检查你的电路的主模块的命名是否符合要求。

注意:请保证模块的 appearance 与下图完全一致，否则有可能造成评测错误(查



看模块 appearance 方法:在 Logisim 中打开相应模块后点击左上角

按钮)



- 输入端从上至下依次是 A、B、C、D

1.4 组合逻辑

基本的门电路知识掌握以后，接下来我们介绍一下数字电路的基本知识：组合逻辑和时序逻辑。

数字电路根据逻辑功能的不同特点，可以分成两大类，一类叫组合逻辑电路（简称组合电路），另一类叫做时序逻辑电路（简称时序电路）。他们有什么区别呢，其实很简单：大家都知道，我们做的每个小电路，都可以抽象成一块封装好的，具有一定功能的电路板，它有输入端、电路板主体（就是内部结构）和输出端。电路内部结构和输入信号共同决定输出信号。**组合逻辑电路的输出仅仅取决于当前输入；时序逻辑电路的输出取决于当前的输入和电路原来的状态，或者说：时序逻辑电路的输出，不仅与当前输入有关，也与过去的输入有关。**时序逻辑是有记忆的电路板，组合逻辑是没有记忆的电路板。

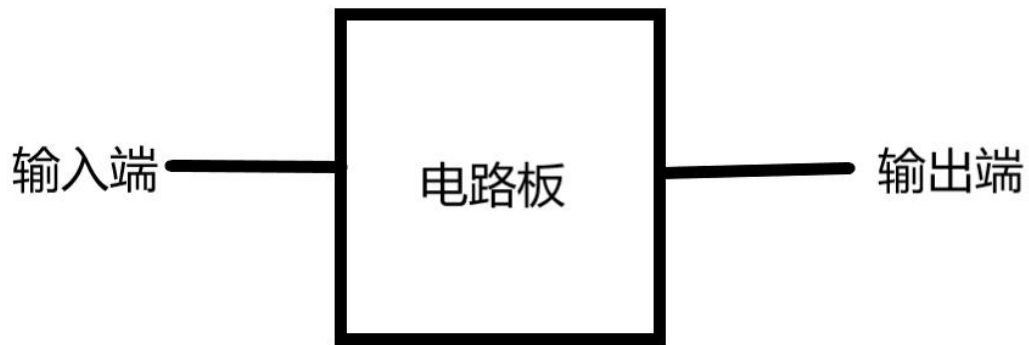


图 4-1 电路板

我们首先来学习组合逻辑。

组合逻辑很简单，理论上，只要明确了我们需要实现的电路的**逻辑表达式**（这玩意大家离散数学课都学过吧？），就可以按照逻辑表达式轻而易举的做出电路图。所以组合逻辑这块的重点就是写好逻辑表达式：大家需要分析题目的需求，抽象出逻辑表达式，然后画出对应的电路图，组合逻辑的问题就是这样。

我们来看一道例题：现在需要你实现一个电路，它有三个输入端：S、i1、i2，和两个输出端 o1、o2。要实现的功能很简单：当 S=0 时，你要保证 o1=i1 且 o2=i2；当 S=1 时，你要保证 o1=i2，o2=i1。

题目的需求了解了，这就是一个一位 swap 电路。首先我们要按照题目需求写出真值表。

S	i1	i2	o1	o2
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0

S	i1	i2	o1	o2
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

接下来按照真值表写出对应的逻辑表达式并化简。

说明：写出并化简逻辑表达式的方法可以参考《数字设计原理与实践 第四版》P213。

然后按照化简完的表达式画出电路图就可以了：

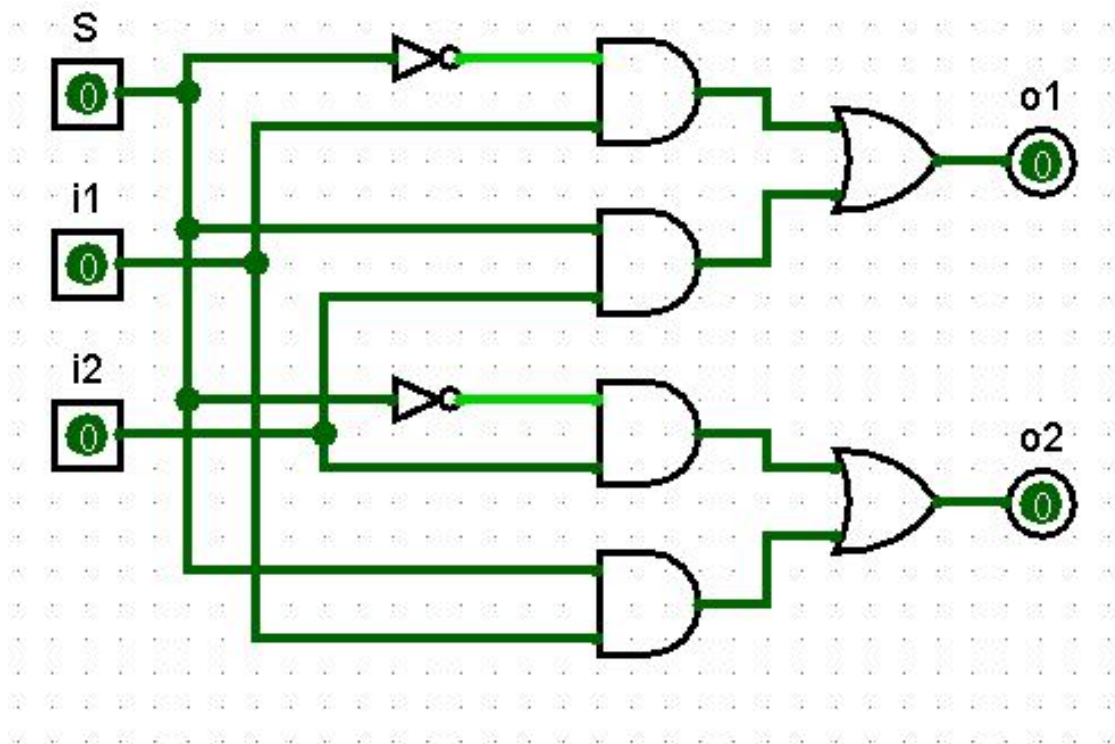


图 4-2 swap 电路

就是这么简单！

也许你会觉得这个过程很繁琐，的确，尤其是输入输出信号变多的时候，手动进行“写真值表->写逻辑表达式->化简逻辑表达式->制作电路”这个过程将变得相当复杂。

没关系，Logisim 提供了一种更加简便的方法：**Analyze Circuit**

点击左上角 **Project->Analyze Circuit**，定义你的输入输出信号名称，再点击窗口中的 **Table**，手动输入真值表，就可以自动生成化简好的逻辑表达式以及电路了，非常的方便！

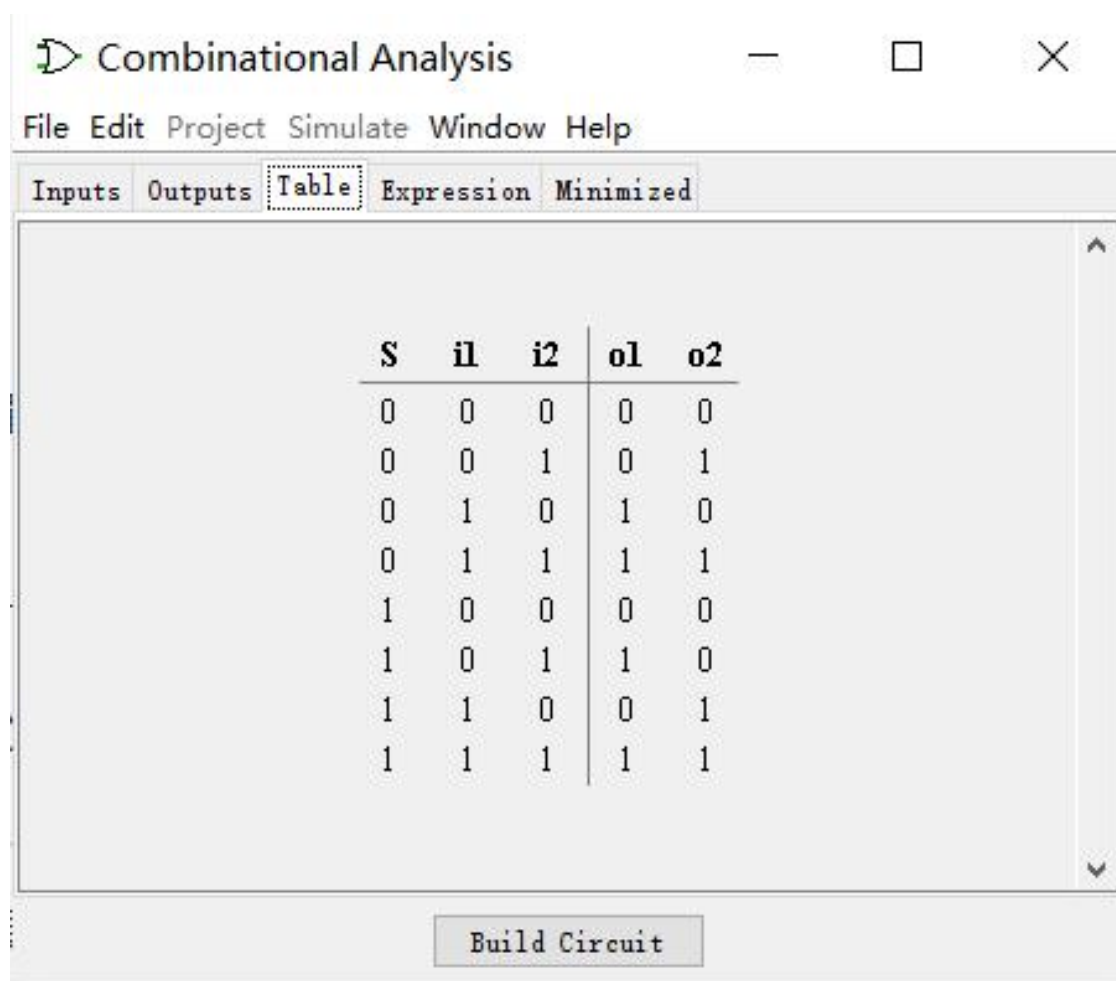


图 4-3 Analyze Circuit

绝大多数情况，课设需要你实现的组合逻辑问题，都可以用 Analyze Circuit 完成，大家记得写完真值表后检查一下，看看有没有疏漏，以防出现马虎产生的 bug。

课下测试 2

知识点：组合逻辑、Logisim 中 Plexers 元件的使用

难度：简单

请在开始写这个作业之前，参照手册或视频，自行学习 Plexers 等 Logisim 元件的使用。

说明：Plexers 等元件的使用可以参考 Logisim 中文指导手册

实验任务

算术逻辑单元(arithmetic and logic unit) 是实现多组算术运算和逻辑运算的组合逻辑电路，简称 ALU，是中央处理器（CPU）的重要组成部分。简单来讲，就是 CPU 中负责进行计算（算数和逻辑运算）的电路元件。

这道题要求你自己动手搭建一个小型 ALU。

实验具体要求

要做 ALU，首先我们要了解 ALU 的输入输出端口。ALU 本身是一个纯组合逻辑电路，它需要我们传入要进行计算的数据，并且“告诉”它对于这两个数据，要进行何种运算，之后它就会输出结果。

那么如何通知 ALU 要进行何种运算呢？我们把它抽象成一个数字输入信号，例如在本题中，我们采用一个四位二进制数输入信号作为 ALU 的控制信号

（ALUOp），具体对应关系如下：

ALUOp 对应的运算

0000 $i1 \ll i2$

ALUOp 对应的运算

0001 $i1 \ggg i2$

0010 $i1 \gg i2$

0011 $i1 * i2$ (无符号乘法, 仅保留低 32 位结果)

0100 $i1 / i2$ (无符号除法, 仅保留 32 位商)

0101 $i1 + i2$

0110 $i1 - i2$

0111 $i1 \& i2$

1000 $i1 | i2$

1001 $i1 \oplus i2$

1010 $i1 \text{ NOR } i2$

1011 $R = (X < Y) ? 1 : 0$ (有符号比较)

1100 $R = (X < Y) ? 1 : 0$ (无符号比较)

注意:

(1) 不要自己手搓加法器和减法器! **logisim** 库里有现成的, 直接使用

logisim 自带的加法器减法器就可以。手搓可能会导致评测 TLE!

(2) 进行移位运算时, 32 位二进制数 $i2$ 表示移多少位, 其高 27 位应该被舍弃, 仅低 5 位有效。

输入输出要求

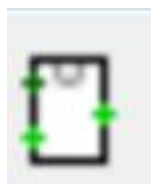
(1) 输入: $i1$ (第一个运算数, 32bit), $i2$ (第二个运算数, 32bit),
ALUOp (ALU 控制信号, 4bit)

(2) 输出: R (运算结果, 32bit)

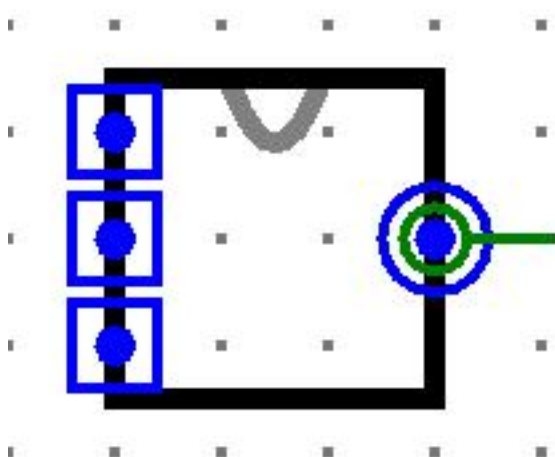
(3) 请将文件命名为 test2.circ, 文件内主模块命名为 **ALU**

(4) 注意：请保证模块的 **appearance** 与下图完全一致，否则可能造成评测错误！(查看模块 **appearance** 方法:在 Logisim 中打开相应模块后点击

左上角



按钮)



(输入端从上到下依次是 i1,i2,ALUOp)

思考

课下测试 2 做出的 ALU 其实是有缺陷的。试想一下：两个 32 位数相乘，得到的应该是一个 64 位数；同样，两个 32 位数相除，应该得到一个 32 位的商和一个 32 位的余数。

真正我们在设计 CPU 的时候是绝对不允许这样的缺陷存在的，否则我们的 CPU 将无法正确执行乘除指令。

我们使用的 MIPS-Lite2 指令集规定：对于乘法指令，得到的结果，高 32 位存储在 hi 寄存器中，低 32 位存储在 lo 寄存器中；对于除法指令，32 位的商存储在

lo 寄存器中，32 位的余数存储在 lo 寄存器中。而且，真实情况下乘除法运算部件的延迟是比其他算数、逻辑运算的延迟高得多的。因此在真正搭建 CPU 时，乘除法运算通常是放在单独的乘除法运算模块进行，而不是在 ALU 中。

感兴趣的同学可以自行查阅资料了解。

1.5 时序逻辑

理论课大家已经学习了时序电路的基本知识，下面我们结合 Logisim 再帮助大家理解一下时序电路。

说明：时序逻辑理论知识可以参考《数字设计原理与实践 第四版》P437 Sequential logic 章节。

和组合逻辑电路不同，时序电路需要实现“记忆”的功能，那么电路里必然存在用于记忆过去状态的部件，如何存储过去的信息呢？首先我们一起看一下这个“小物件”：**SR 锁存器**

它长这个样子：

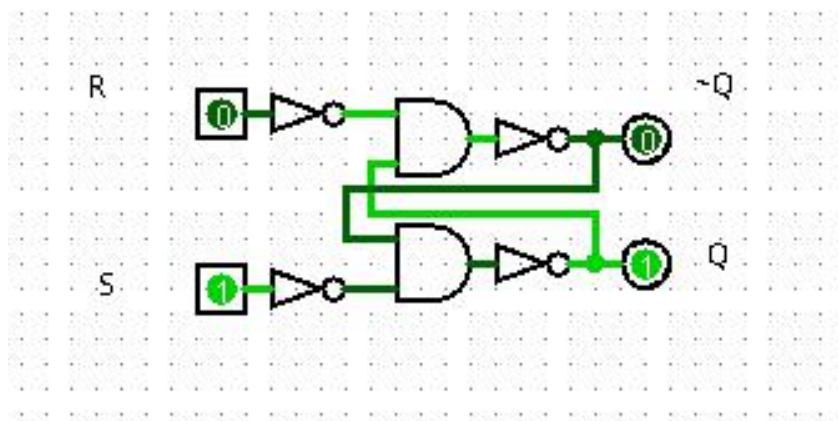


图 5-1 SR 锁存器

我们来看一下它的真值表：

$\overline{S_D}$	$\overline{R_D}$	Q^n	Q^{n+1}	功能
0	1	0	1	置1：次态为1
0	1	1	1	
1	0	0	0	置0：次态为0
1	0	1	0	
1	1	0	0	保持：次态不变
1	1	1	1	
0	0	0	约束条件， $\overline{S_D}$ 和 $\overline{R_D}$ 不能同时为0，即必有： $\overline{S_D} + \overline{R_D} = 1$	
0	0	1		

图 5-2 SR 锁存器的特性真值表

看起来可能不太好理解，我们一起分析一下：SR 锁存器使得电路具有了记忆上一个状态的功能。其中 $\overline{S_D}$ 和 $\overline{R_D}$ 为是电路输入的两个端口， Q^n 表示电路当前的输出， Q^{n+1} 表示电路下一个状态的输出。由真值表可以看出，这个电路的输出不仅和当前输入有关，也和上一次的输出有关。显然这个电路就是一种能记住自己之前状态的电路。其实，这就是一个最简单的时序电路，我们称它为 SR 锁存器。我们使 $\overline{S_D} = \overline{R_D} = 0$ ，电路就可以锁住上一个状态的 Q （ Q 不仅仅取决于 $\overline{S_D}$ 和 $\overline{R_D}$ ，还包括上一次的 Q ，所以 SR 锁存器不是组合逻辑而是时序逻辑），这就是锁存器名字的由来。

观察它的功能一栏，可以看到，通过改变 SD 和 RD 为合适的值,我们可以改变电路的输出，而当 SD 和 RD 均为 0 时，电路会一直保持原来的输出不变，这看上去很像 U 盘之类的设备（通电时能够修改存储的内容，断电时保持内容不变）。事实上，通过配合合适的外部电路，我们就可以使用这个电路来存储整个电路的状态，从而搭建起更复杂的时序电路。

掌握了时序电路的基本原理之后，请同学们根据教材自学 D 锁存器和 D 触发器的原理。

说明：D 锁存器和 D 触发器相关知识可以参考《数字设计原理与实践 第四版》P442。

如果你懒得自学研究 D 锁存器和 D 触发器的原理（还是建议大家了解一下），那么至少要知道 D 触发器的特性：

$$Q_{n+1}=D$$

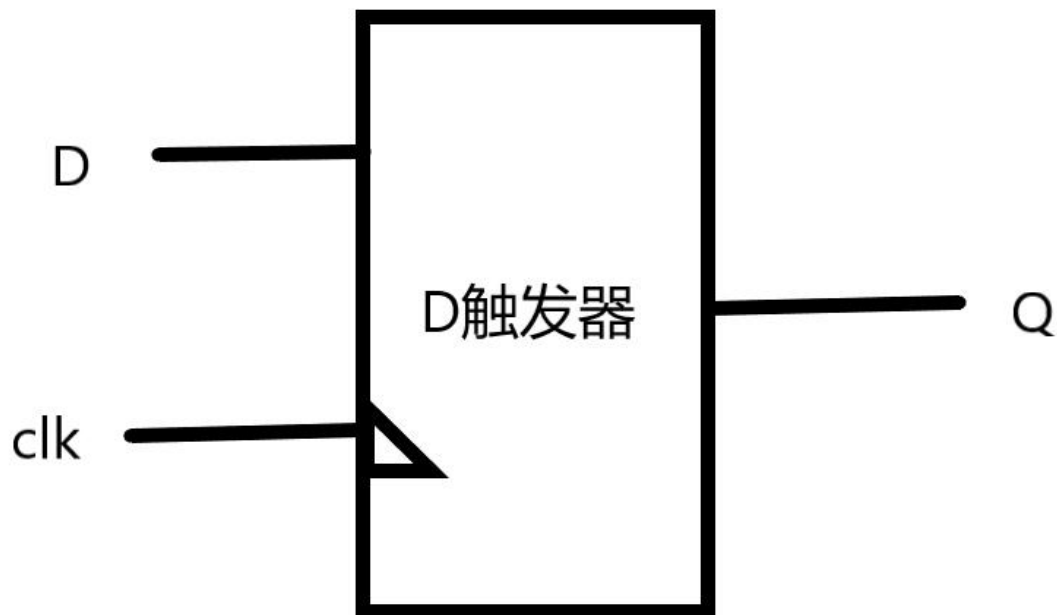


图 5-3 D 触发器

这个逻辑表达式是什么意思呢？我们来看一下 D 触发器的示意图：D 和 clk 都是一位输入信号，clk 还有个特殊的名字叫“时钟信号”，因为我们人为规定时钟信号是规则的方波信号。无论 Q 原来是多少，当 clk 信号由 0 跳变到 1 时（这就是我们说的**时钟上升沿**），Q 就会变成此时的 D，并且保持不变直到下一个时钟上升沿到来。在两个时钟上升沿之间，无论 D 怎么变化，Q 都保持不变，以实现记忆功能。

D 触发器是最常用的时序逻辑基本电路，多个 D 触发器并联在一起就构成了时序逻辑电路的最重要元件：**寄存器**。

例如 32 个共享 clk 信号的 D 触发器放在一起就是一个 32 位的寄存器。

接下来的有限状态机和 CPU 的设计，就要使用到寄存器。

1.6 有限状态机

注意：本文所有内容假设阅读者具有基本的有限状态机相关理论知识！如果你不具备相应的理论知识，请在课堂上学习，或者查阅《数字设计和计算机体系结构》有限状态机章节进行学习。

说明：有限状态机理论知识可以参考《数字设计原理与实践 第四版》P452 clocked synchronous state-machine analysis 部分。

简而言之，**Moore** 型有限状态机输出仅仅和当前状态有关；**Mealy** 型状态机输出和当前状态以及输入信号都有关。

但是二者的下一状态都是仅和当前状态以及输入信号有关，它们之间的关系是组合逻辑关系。

另外扯一句：CPU 本质上可以理解为大型有限状态机的封装。

接下来我们学习一下如何用 Logisim 实现有限状态机。

在搭建有限状态机前，首先要对状态机进行设计，这是整个电路搭建工作的重点和难点，设计好了，之后制作电路的过程就是非常程式化的了。最基本的，你首先要根据问题设计出状态转移图；下一步就是对状态进行二进制编码。我们来看下面的这个状态转移图。

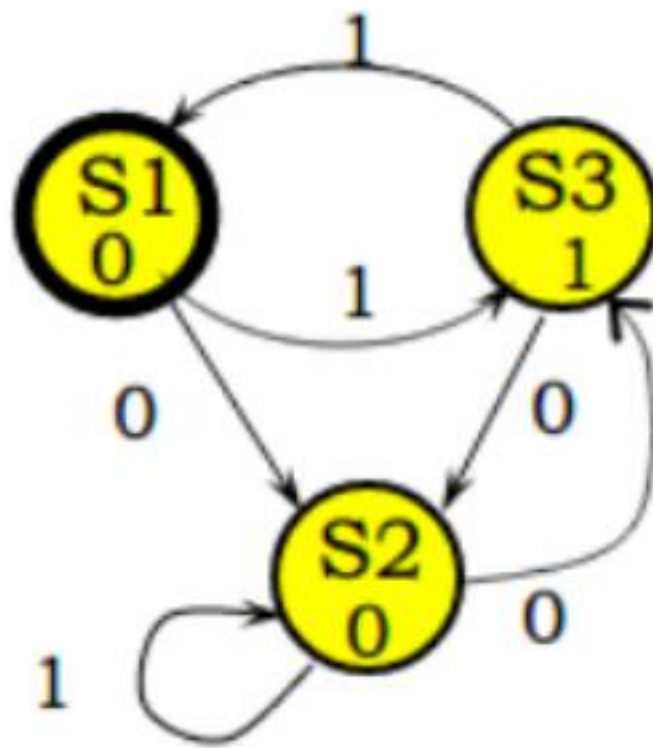


图 6-1 状态转移图

这个有限状态机有三个状态，我们可以采用最简单的方式对其进行编码：

三个状态至少需要 2 位二进制数才能不重复地表示出来

状态 编码

状态 编码

S1 00

S2 01

S3 10

也可以使用独热编码：

独热编码就是：每一个状态的编码中仅有一位是 1，其余全是 0 的编码。

状态 编码

S1 001

S2 010

S3 100

还可以采用其他任意方式编码，总之你能用 m 位二进制数表示出每一个状态即可（实际生产中独热编码使用较多）。

有限状态机的输入 n 位（本题 $n=1$ ），输出 k 位（本题 $k=1$ ）。

接下来就是实现状态转移逻辑：

首先写出状态转移表：

当前状态 输入 下一状态

S1 0 S2

S1 1 S3

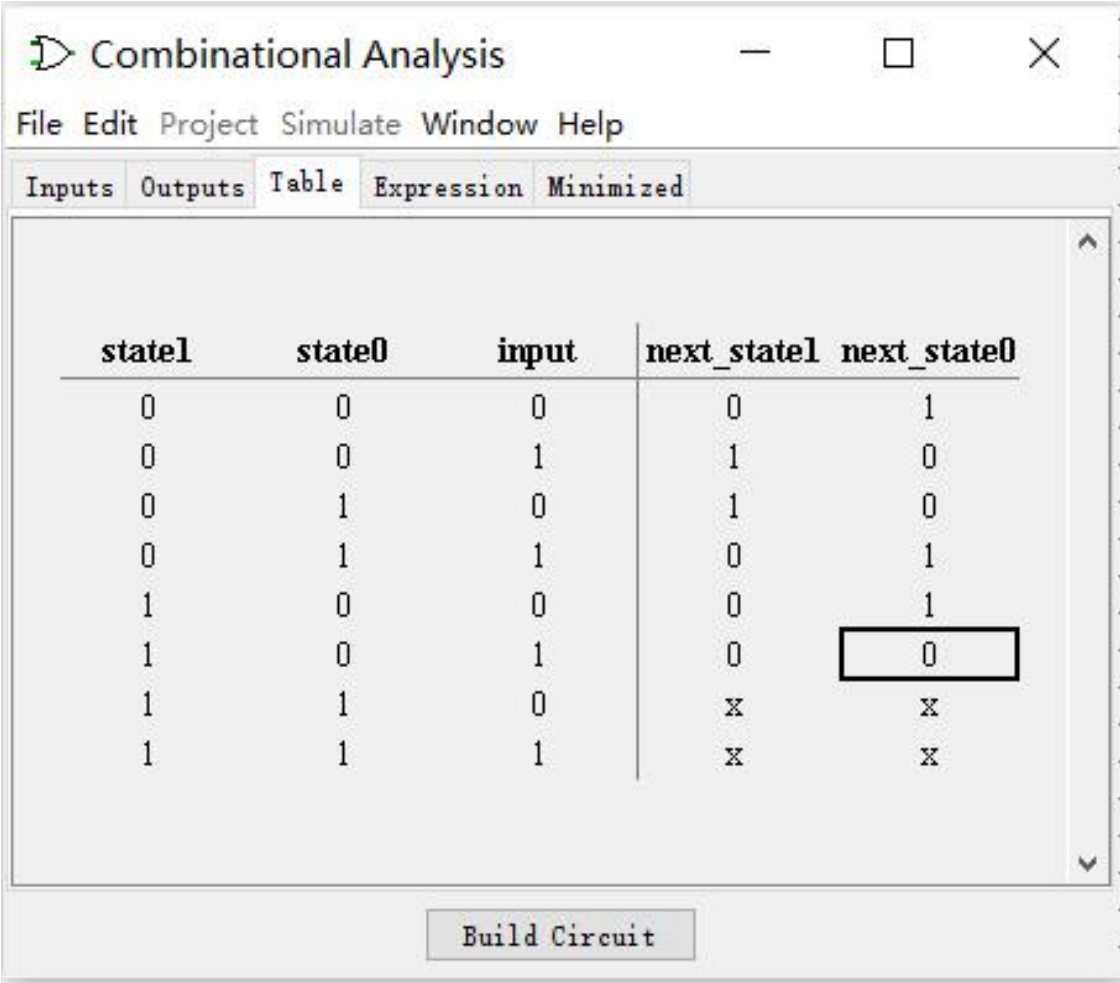
S2 0 S3

S2 1 S2

S3 0 S2

S3 1 S1

根据状态转移表，使用 Logisim 的 Analyze Circuit 功能生成状态转移电路：我这里对状态采用第一种编码方式。



state1	state0	input	next_state1	next_state0
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	0
1	1	0	x	x
1	1	1	x	x

Build Circuit

图 6-2 状态转移真值表

我们的状态编码是两位的，state0 表示编码第 0 位，state1 表示编码第 1 位。

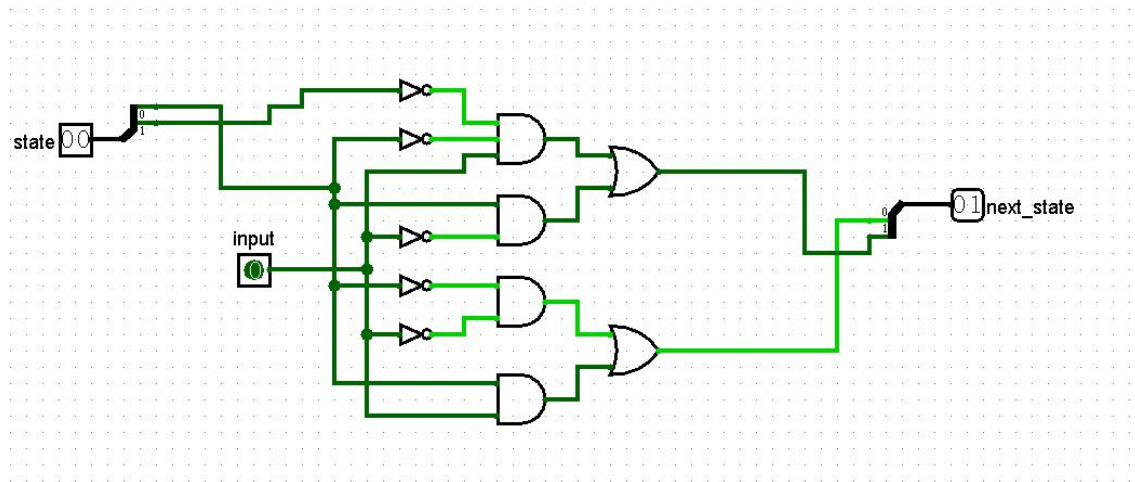


图 6-3 状态转移电路

如上图，我们的状态转移电路就完成了。

接下来我们需要完成状态输出电路。根据状态转移图可知，这是一个 Moore 型有限状态机，输出仅跟当前状态相关。

这道题的输出逻辑比较简单，就是当且仅当当前状态为 S3 时，输出 1，否则输出 0。

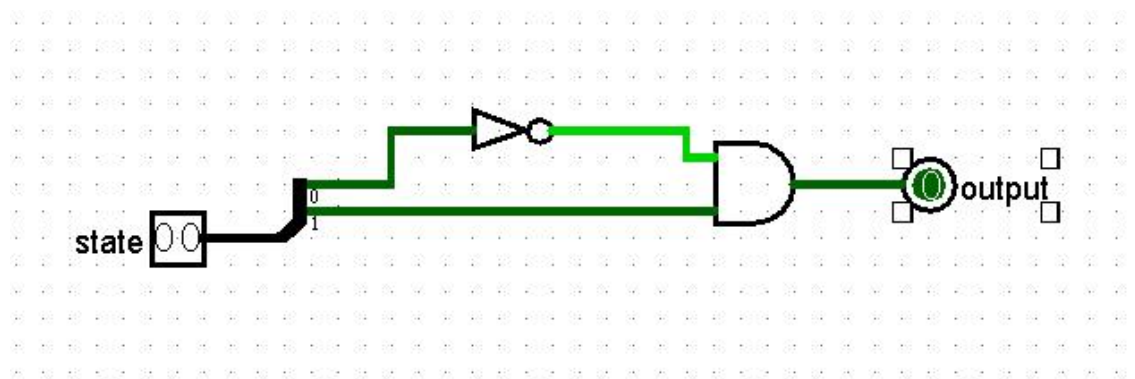


图 6-4 状态输出电路

所谓状态存储，就是一个寄存器，例如你的状态编码是 3 位的，那么就准备一个 3 位寄存器即可，本题中我采用 2 位编码，因此选一个 2 位寄存器。

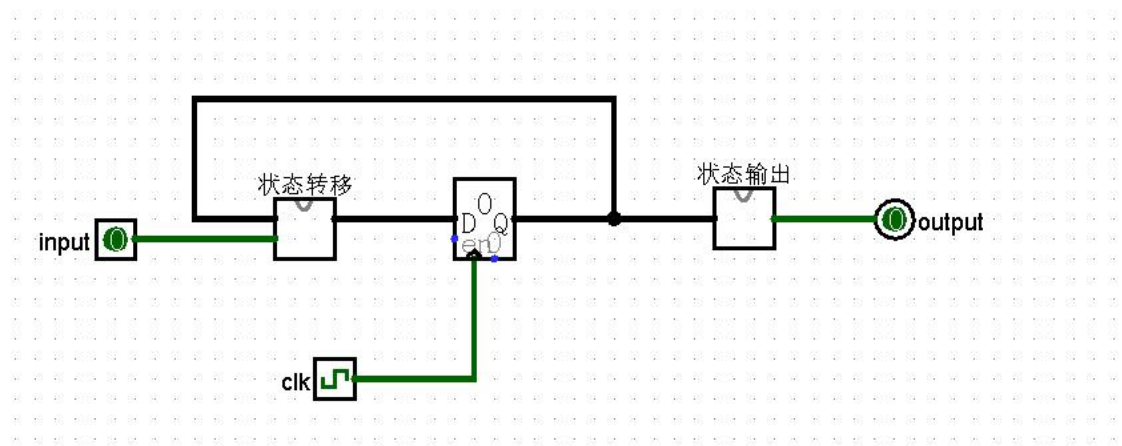


图 6-5 Moore 型状态机

如上图所示，一个 Moore 型有限状态机就搭建好了。

Mealy 型状态机的搭建方法和 Moore 型相同，唯一区别就是状态输出部分需要把输入信号也考虑进去，整体上看起来就是比 Moore 型多一根输入信号到状态输出电路的连线，下面是一个 Mealy 型状态机的示意图。

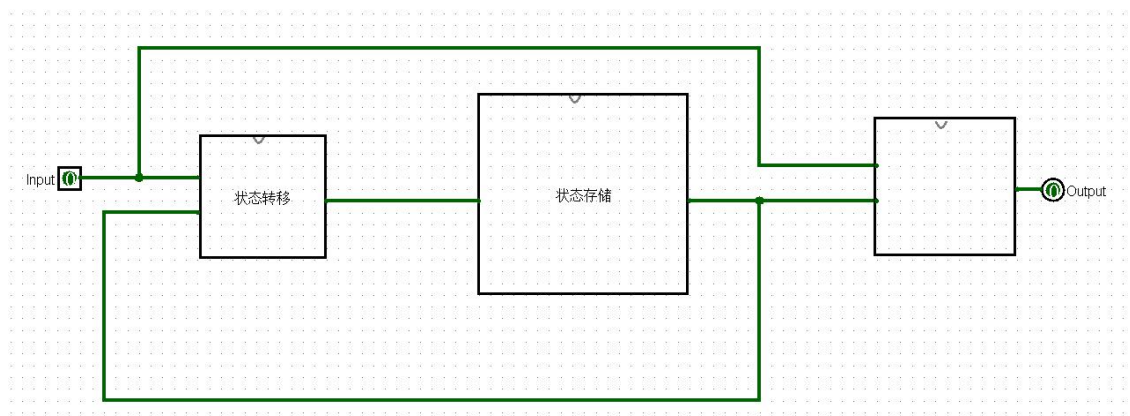


图 6-6 Mealy 型状态机

有限状态机是一个常用的时序电路模型，它并不复杂，因为除了寄存器部分外，状态转移和状态输出电路都是都是纯组合逻辑，只要画好状态转移图，接下来的电路设计就易如反掌。下面我们来练习一下。

课下测试 3

知识点：时序逻辑、有限状态机的搭建

难度：中等

实验任务

很多数学问题都可以使用有限状态机解决。

这道题要求你搭建一个 **Mealy 型状态机** 解决 $2^n \bmod 5$ 的数学问题。

实验具体要求

使用 Logisim 搭建电路，该电路串行输入一个二进制无符号数 B(先从高位输入,每输入一个数字就相当于之前输入的数左移一位再加上当前输入的数字)，输出“2 的 B 次幂”模 5 的余数的电路并提交。

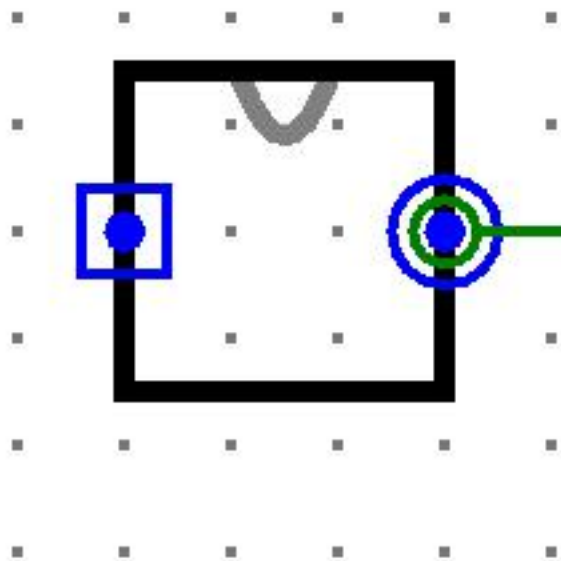
输入输出要求

- (1) 输入：**In (1bit 串行输入)**（每个时钟上升沿读入一个 1bit 数）
- (2) 输出：**RESULT (运算结果，3bit 二进制数)**
- (3) 请将文件命名为：test3.circ，**文件内主模块命名为 2nmod5**
- (4) 注意：请保证模块的 **apperance** 与下图完全一致，否则可能造成评测错误！(查看模块 appearance 方法:在 Logisim 中打开相应模块后点击

左上角



按钮)



提示：

这道题看起来可能没有头绪，实际上是一个找规律的问题：2 的 n 次方求余 5，无论 n 怎样变化，结果就只有那么几个，而且如果你把 n 用二进制表示出来，你就会发现求余结果仅和 n 的固定几位有关系。

在纸上写一写，找规律：

$2^0\%5=?$, $2^1\%5=?$, $2^2\%5=?$, $2^3\%5=?$, $2^4\%5=?$, $2^5\%5=?$, $2^6\%5=?$,
 $2^7\%5=?$

1.7 补充习题

注意：补充习题也必须完成才有资格参加课上测试。

课下测试 4

知识点：组合逻辑、Logisim 自带库元件的使用

难度：简单

实验任务

奇偶校验码是非常常用的一种校验码，它可以用来检查数据传输（或储存）过程中是否发生了突变错误。我们现在有一种最简单的奇偶校验码：发送方发送的数据是 9 位，其中，高 8 位是原始数据（也就是真正有意义需要传输的信息），最后一位是机器生成的校验码（用于检验信息正确性），例如 **110111010**，其中 **11011101** 是原始数据，最末尾的 **0** 是校验码。如果原始的 8 位数据中 **1** 的个数为偶数，那么机器生成的校验码就是 **0**；反之，若原始数据中 **1** 的个数为奇数，那么机器生成的校验码就应该是 **1**，拼装好的这个 9 位数就会被传送出去，接收方接收到这个 9 位数，需要校验数据在传输过程中是否发生偏差，就是根据这个校验码检验。

例如，如果接收到这样一个数 **001100111**，对应的 8 位原始数据是 **00110011**，**1** 的个数是偶数，然而尾随的校验码 **1** 却显示 8 位原始数据中 **1** 的个数应为奇数，这就说明数据在传输的过程中一定发生了错误。

本题要求你写一个接收机检验数据是否正确的电路。

实验具体要求

我们的电路输入是接收到的 9 位数，对其进行检验，如果数据是正确的，就输出 8 位原始数据；如果是错误的，就输出 **11111111**。

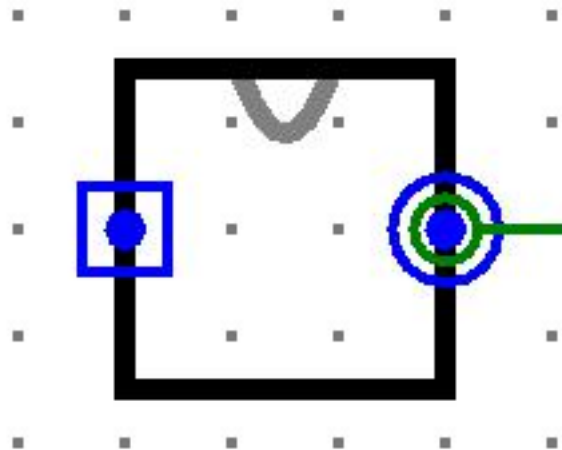
输入输出要求

- (1) 输入：**input** (9bits)
- (2) 输出：**output** (8bits)
- (3) 请将文件命名为 test4.circ，文件内主模块命名为 **check**
- (4) 注意：请保证模块的 **appearance** 与下图完全一致，否则可能造成评测错误！（查看模块 **appearance** 方法：在 Logisim 中打开相应模块后点击

左上角



按钮)



课下测试 5

知识点：组合逻辑、swap 电路、子电路的灵活运用。

难度：困难

实验任务

这道题要求你搭建一个具有排序功能的电路。

实验具体要求

给定 4 个输入：i1, i2, i3, i4，均为 4 位二进制无符号数。

你需要对它们进行排序并输出排序后的结果，即保证四个输出的关系为 $o1 \geq o2 \geq o3 \geq o4$ 。

例如：i1 = 1, i2 = 2, i3 = 4, i4 = 5，则：o1 = 5, o2 = 4, o3 = 2, o4 = 1。

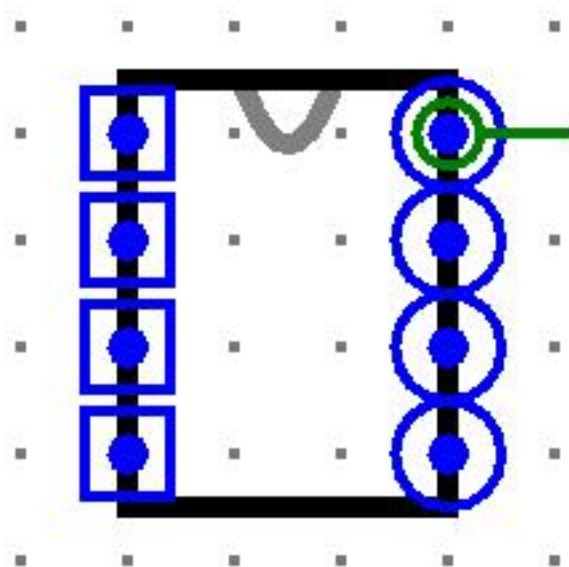
输入输出要求

- (1) 输入: **i1, i2, i3, i4** (均为 4bit)
- (2) 输出: **o1, o2, o3, o4** (均为 4bit)
- (3) 请将文件命名为 test5.circ, **文件内主模块命名为 sort**
- (4) 注意: 请保证模块的 **appearance** 与下图完全一致, 否则可能造成评测错误! (查看模块 appearance 方法: 在 Logisim 中打开相应模块后点击

左上角



按钮)



(输入端从上到下依次是 i1,i2,i3,i4。输出端从上到下依次是 o1,o2,o3,o4。)