
目录

实验一 从逻辑门到芯片	2
实验目的	2
实验内容	2
逻辑门	2
与门	2
或门	5
非门	7
小结	8
逻辑门组成的双控开关	10
小结	12
从逻辑门电路到芯片	13
设计逻辑门电路	14
设计芯片版图	14
小结	22
实验步骤	23
环境搭建	23
安装虚拟机	23
安装操作系统与工具	23
运行工具	23

实验一 从逻辑门到芯片

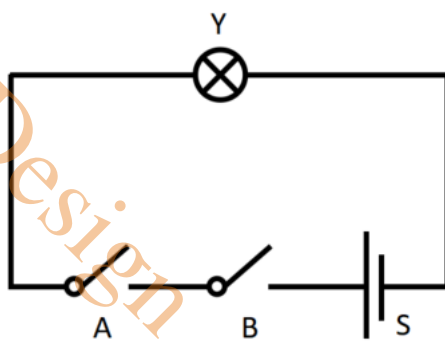
实验目的

- (1) 了解芯片设计及物理设计的基础流程，引入基础概念。
- (2) 掌握实验环境的搭建。
- (3) 介绍 iFlow 工具，带领学生在工具上简单地跑通示例设计。

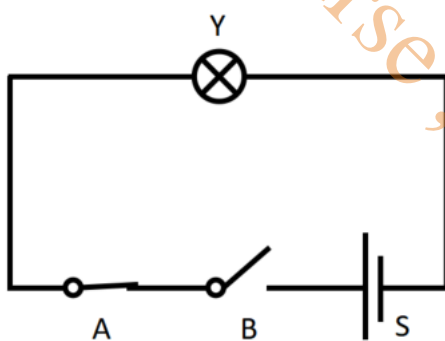
实验内容

逻辑门

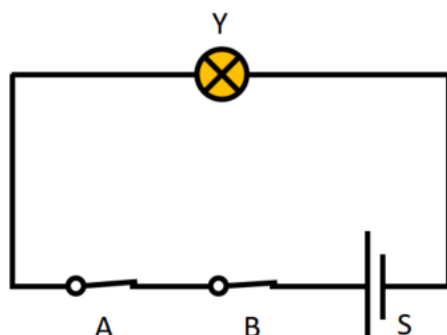
与门



这是一个简单的日光灯电路，它由两个开关（A 和 B），一个电灯泡（Y）和一个电源（S）组成。



当开关 A 处于闭合状态时，什么都不会发生，因为 A 和 B 是串联的。当 A 断开 B 闭合时，同样的什么都不会发生。只有当 A、B 两个开关都闭合时灯泡才会亮起，如下图。



我们对于这个电路的运转情况，可以总结为以下表格。

开关 A	开关 B	灯泡 Y
断开	断开	不亮
断开	闭合	不亮
闭合	断开	不亮
闭合	闭合	亮

一个开关有两个状态，因此可以使用二进制来表示，二进制 0 表示“开关断开”，二进制 1 表示“开关闭合”。

灯泡也有两种状态，同样也可以用二进制数来表示，0 代表“灯泡不亮”，1 表示“灯泡亮”。

通过以上表达，我们将电路运作情况转换表简化为了以下格式。

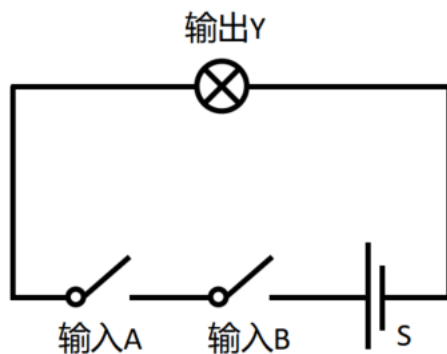
开关 A	开关 B	灯泡 Y
0	0	0
0	1	0
1	0	0
1	1	1

总结一下，只有当两个开关都处于闭合状态时，灯才会亮，否则灯就不亮。从二进制的角度也可以说，只有当两个开关的状态都为 1 时，灯泡的状态才为 1，否则灯泡状态为 0。

我们将这两个开关与灯泡之间的关系称为“与”。“与”的意思为，只有当全部开关都闭合时，灯泡才会亮，否则灯泡不亮。

我们在电路上操控的两个开关可以看成“我们对电路的改变”，这可以看成是我们对电路的输入。

由于开关的变化，进而引起灯泡的亮和不亮可以看成“我们对电路的改变后电路做出的反应”，这可以看成是电路的输出。



使用“输入”来表示开关，“输出”来表示灯泡。进一步将电路运作情况表转换为了以下格式。

输入 A	输入 B	输出 Y
0	0	0
0	1	0
1	0	0
1	1	1

同样的，我们将这两个输入与输出之间的关系称为“与”。只有当输入都为1时（全部开关都闭合时），输出才会为1（灯泡才会亮），否则输出都为0（灯泡不亮）。

为了避免复制的图示，电气工程师们用如下专门的符号来表示“与”，并将其命名为“与门”。



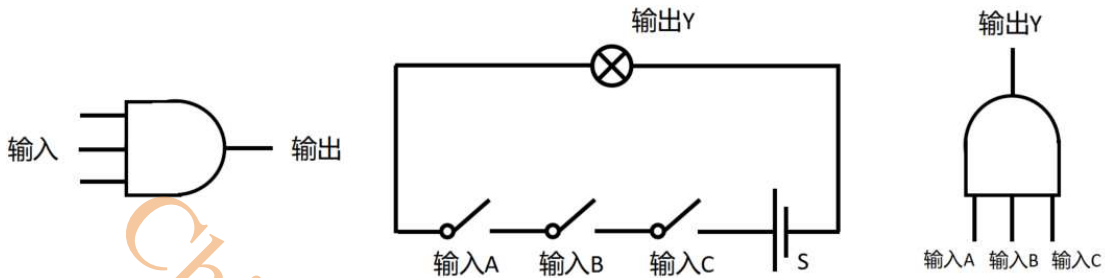
与门是三个基本逻辑门中的一个，与门有两个输入端（上图中的左端）和一个输出端（上图中的右端），与门的功能和日光灯电路是一致的，只有当输入都为1时，输出才会为1。

下面的这个表格是“与门对应的真值表”。

输入 A	输入 B	输出 Y
0	0	0

0	1	0
1	0	0
1	1	1

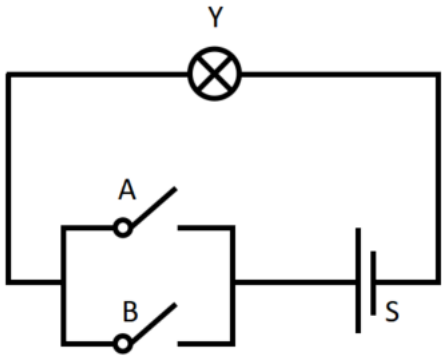
除了有两个输入的与门外，还有三个输入或多个输入的与门，这里以三输入为例子。



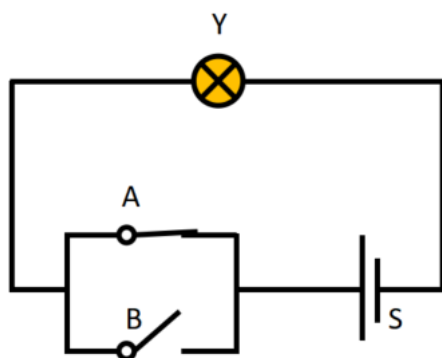
下面的表格是三输入与门的真值表。

输入 A	输入 B	输入 C	输出 Y
0	0	0	0
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

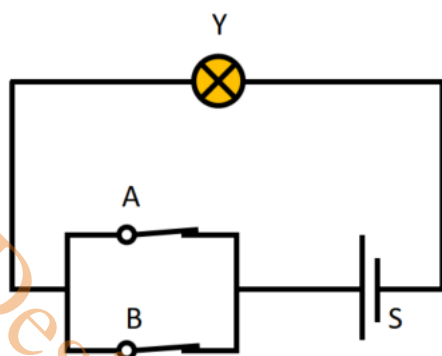
或门



这是另一个简单的日光灯电路，同样的，它由两个开关（A 和 B），一个电灯泡（Y）和一个电源（S）组成。



当开关 A 处于闭合时，电路通路，灯泡点亮。



再把开关 B 闭合，灯泡维持亮的状态。我们对于这个电路的运转情况，可以总结为以下表格。

开关 A	开关 B	灯泡 Y
断开	断开	不亮
断开	闭合	亮
闭合	断开	亮
闭合	闭合	亮

将开关看作为“输入”，灯泡看作为“输出”，用二进制表示状态，那么电路运转情况表被简化为下表。

输入 A	输入 B	输出 Y
0	0	0
0	1	1
1	0	1
1	1	1

我们将这两个开关与灯泡之间的关系称为“或”。“或”的意思为，有一个开关闭合，那么灯泡就会亮，除非全部开关都断开，灯泡才会不亮。电气工程师

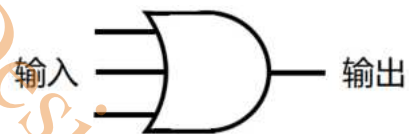
用如下专门的符号来表示“或”，并将其命名为“或门”。



下面的这个表格是“或门对应的真值表”。

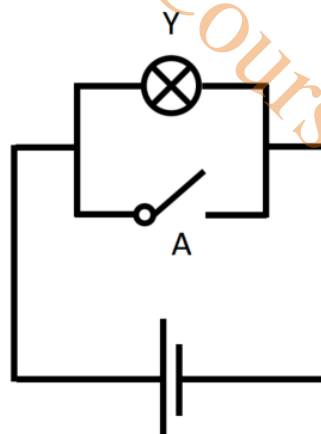
输入 A	输入 B	输出 Y
0	0	0
0	1	1
1	0	1
1	1	1

同样的，或门也可以有多个输入，和与门一样。

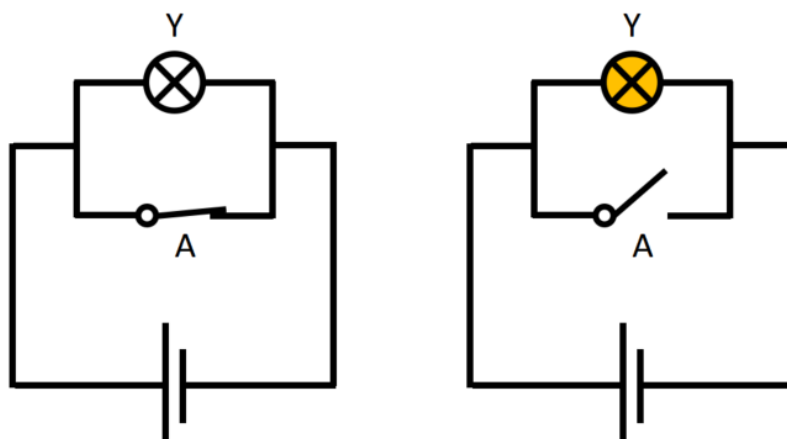


!! 思考：仿照三输入与门的真值表，写出三输入或门的真值表。

非门



这是另一个简单电路。当开关 A 闭合后，灯泡不亮，当开关 A 断开，灯泡点亮。



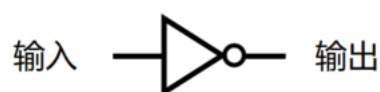
我们对于这个电路的运转情况，可以总结为以下表格。

开关 A	灯泡 Y
闭合	不亮
断开	亮

将开关看作为“输入”，灯泡看作为“输出”，用二进制表示状态，那么电路运转情况表被简化为下表。

输入 A	输出 Y
1	0
0	1

我们将这一个开关与灯泡之间的关系称为“非”。“非”的意思为，当开关闭合，灯泡反而不亮，而开关断开，灯泡则点亮。电气工程师用如下专门的符号来表示“非”，并将其命名为“非门”。



下面的这个表格是“或门对应的真值表”。

输入 A	输出 Y
1	0
0	1

非门只能有一个输入和一个输出。

小结

本结学习了三个基本的逻辑门，“与门”的特点是，只有当输入都为 1 时，其输出才为 1。“或门”的特点是，只有当输入都为 0 时，其输出才为 0。“非

门”的特点是输入和输出的状态是相反的。下一章将介绍如何使用这三个逻辑门组成一个简单的电路。

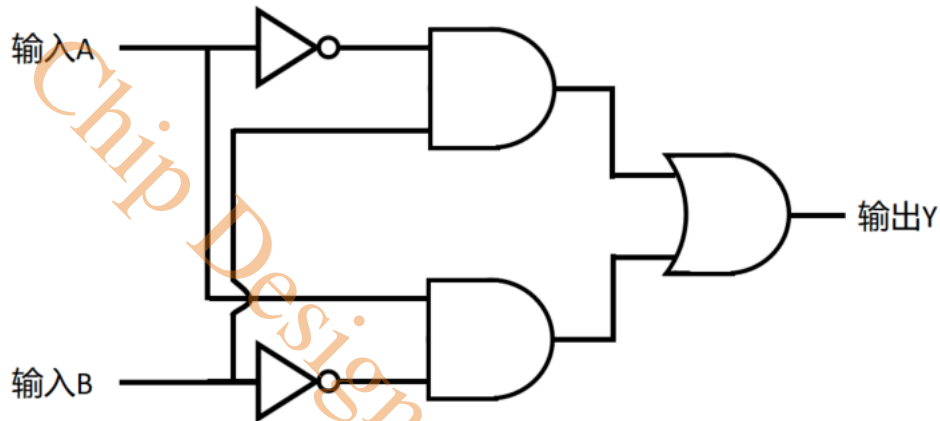
Chip Design Course, SZU

逻辑门组成的双控开关

三个基本的逻辑门可以互相组合，实现具有多种多样功能的门电路。比如下面这个门电路可以实现双控开关的功能。

什么是双控开关电路？

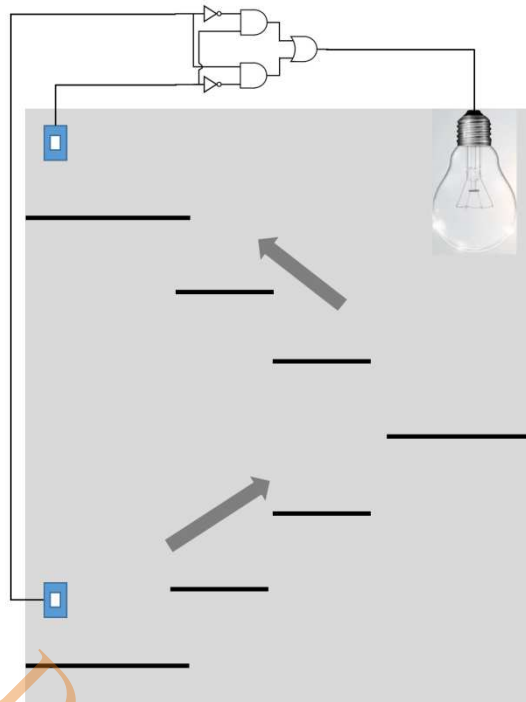
通常由两个开关控制一个灯或其它电器，比如，在楼下打开开关，到楼上关闭开关。如果是采取传统的开关的话，想要把灯关上，就要跑下楼去关，采用双控开关，在楼下楼上各安一个，就可以避免这个麻烦。



这个电路有两个输入，两个非门，两个与门，一个或门和一个输出。这个电路的真值表如下。

输入 A	输入 B	输出 Y
0	0	0
0	1	1
1	0	1
1	1	0

我们把输入 A 作为楼下开关，输入 B 作为楼上开关，输出 Y 为楼道内的灯泡。开关的初始状态为，楼下和楼上的开关都断开（输入 A：0，输入 B：0），每按下一次开关，都算作开关的一次改变。



准备上楼，按楼下开关，输入 A 从 0 变为 1（输入 A: 1，输入 B: 0），此时输出 Y 为 1，灯泡亮起。

到楼上后，按楼上开关，输入 B 从 0 变为 1（输入 A: 1，输入 B: 1），此时输出 Y 为 0，灯泡熄灭。

准备下楼，按楼上开关，输入 B 从 1 变为 0（输入 A: 1，输入 B: 0），此时输出 Y 为 1，灯泡亮起。

到楼下后，按楼下开关，输入 A 从 1 变为 0（输入 A: 0，输入 B: 0），此时输出 Y 为 0，灯泡熄灭。

将这个过程的表格总结如下。

序号	楼下开关	楼上开关	灯泡
①	0	0	0
②	1	0	1
③	1	1	0
④	1	0	1
⑤	0	0	0

从表格中可以看出，从①到②，只变化了楼下开关，从②到③，只变化了楼上开关，剩下的每相邻两步都是如此。实际上，可以发现，双控开关可以不关心

开关状态到底是闭合还是断开，实际上只要其中一个开关发生了变化，灯泡的状态就会发生变化。

小结

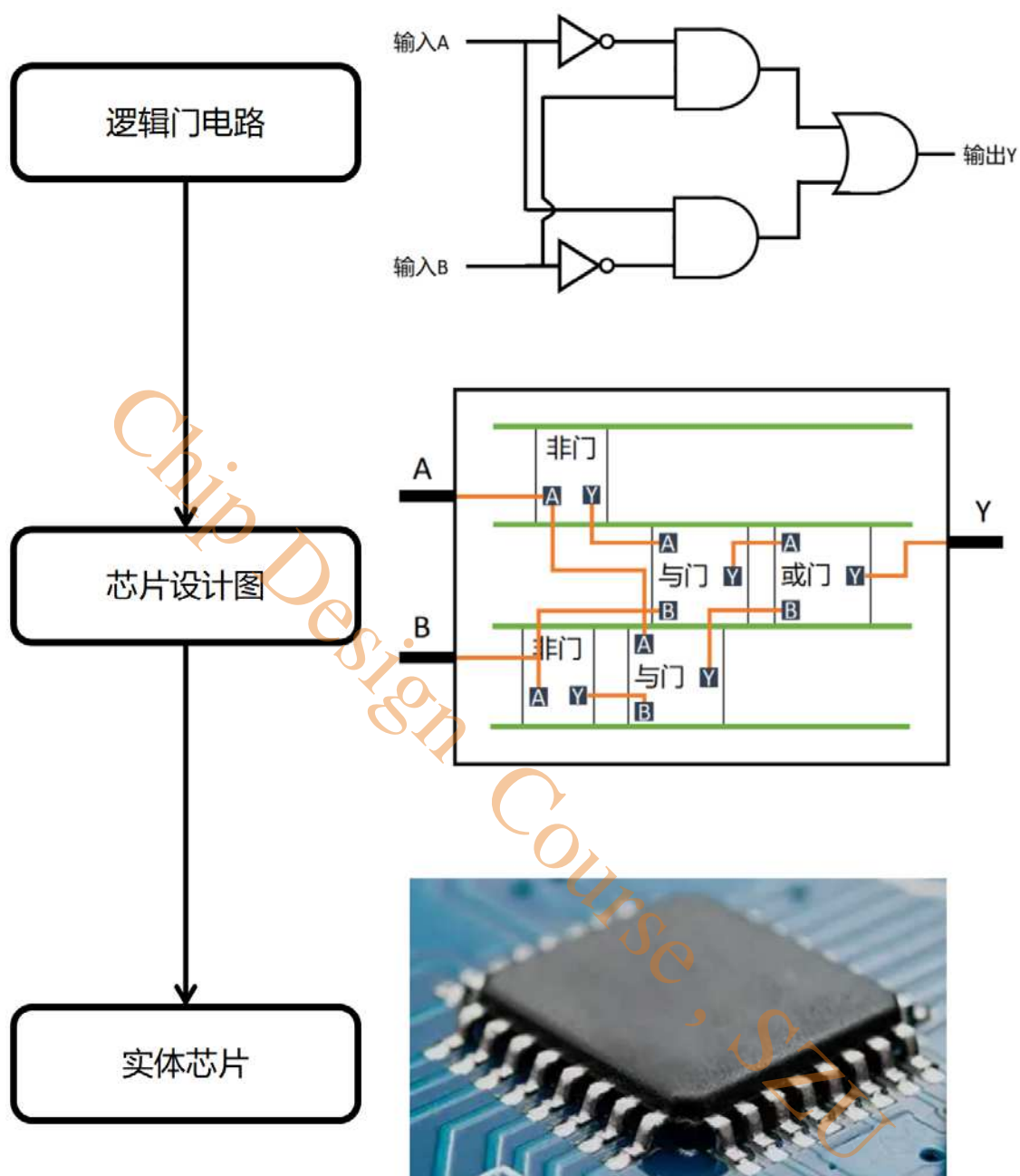
本章介绍了如何使用几个逻辑门实现双控开关电路，这个电路实现了多个开关控制一个电器的功能。这是一个非常有用的功能，也许可以把这个电路做成一个芯片，让想用这功能的人直接使用芯片就好，而不再去关注逻辑门该如何连接。下一章将介绍从逻辑门电路到实体芯片的流程。

大材小用？

当然，肯定没有人会将双控开关电路做成一个芯片，往往芯片中装的都是能够实现多种功能的复杂电路，那些电路有上百万个逻辑门。但这并不妨碍我们拿双控开关电路来用于教学，毕竟如果不在乎金钱成本，这真的可以制造出来。

Chip Design Course, SZU

从逻辑门电路到芯片



仅需要三步就可以做出属于自己的芯片，步骤见上图。

第一步得到逻辑门电路。得到逻辑门电路有很多种方式，其中最原始的就是画出对应的电路图。

当电路结构复杂时，让工程师画出逻辑门的方式就捉襟见肘了。随着时代发展，现在的工程师都通过硬件描述语言代替逻辑门来描述一个电路。硬件描述语言有很多种，如果你学习过 LC3 前端实验，那么 `chisel` 就是其中一个。图中的逻

辑门电路以“双控开关电路”为例子。

第二步制作芯片设计图。一颗芯片该如何制造，往往需要将想法以设计图的方式展现给工厂，工厂再拿着这个设计图进行生产。如何一步一步做出芯片的设计图是本次课程的主要内容，接下来会展开介绍。

第三步生产实体芯片。将芯片设计图交给工厂进行生产，生产出来之后就可以使用自己的芯片了。

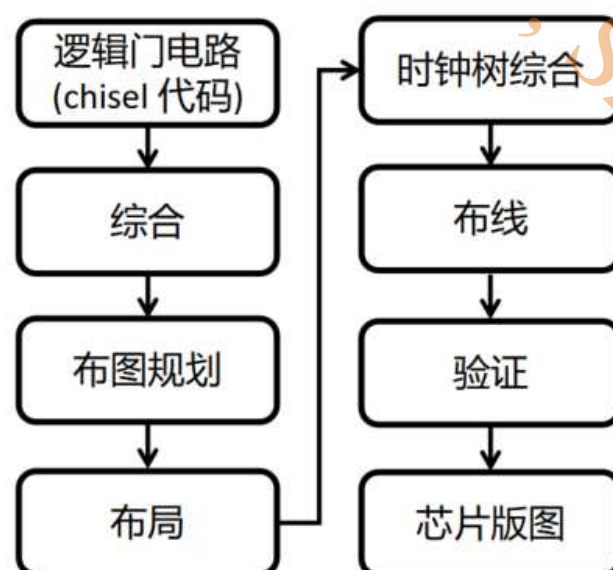
芯片设计图是为了方便理解概念而这么说的，实际上，行业内的专业名字叫做“版图（layout）”，接下来的所有实验里，都用“芯片版图”或“版图”的叫法来代替芯片设计图，在理解的过程中，也可以认为芯片版图指的就是芯片的设计图。

设计逻辑门电路

使用第 2 章设计的双控开关逻辑门电路。

设计芯片版图




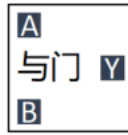


从逻辑门到芯片版图，需要经过 6 大步骤，这些步骤分别是综合、布图规划、布局、时钟树综合、布线和验证。下面将对每一步进行大致介绍



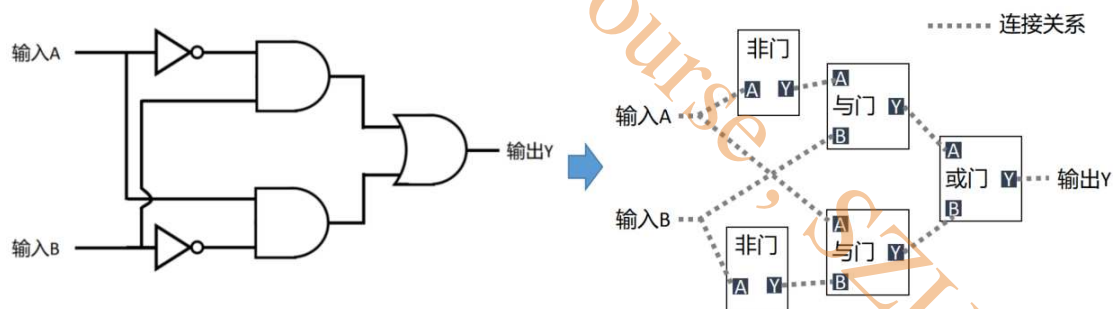
(1) 综合

逻辑门只是一种表示符号，每个逻辑门都有对应实现相同功能的器件，我们称这些器件为“标准单元（standard cell）”。我们暂且不去追究标准单元内部是如何实现的，可以把它看成一个内部已经实现逻辑门相同功能的黑盒子，并且在盒子上有对应的输入输出接口。

如下表，左边是逻辑门，右边是与逻辑门对应有同等功能的标准单元。

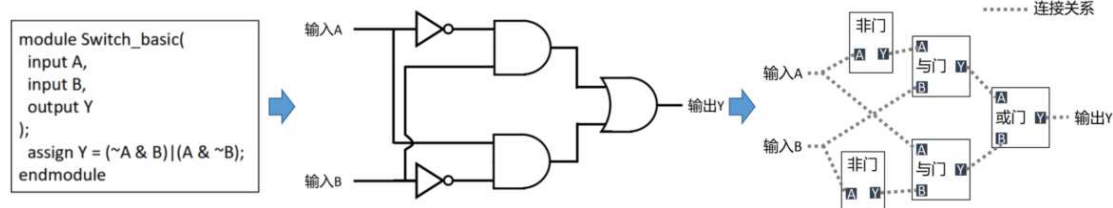
逻辑门	标准单元
	
	
	

综合的主要过程是，将逻辑门用对应同等功能的标准单元进行替换，同时还要保留逻辑门之间的连接关系。



上图中的左边是双控开关的逻辑门电路，通过标准单元将逻辑门替换，生成了右边标准单元之间具有连接关系的电路图，这样具有连接关系的电路图看上去很像一张单元之间的关系网，其专业名称被叫为“网表（netlist）”。

在大规模电路下，无法通过绘图的方式来描述一个逻辑门电路，所以工程师们用硬件描述语言（如 chisel）来描述电路图。所以，在逻辑门转换为标准单元之前还需要将硬件描述语言转换为逻辑门。

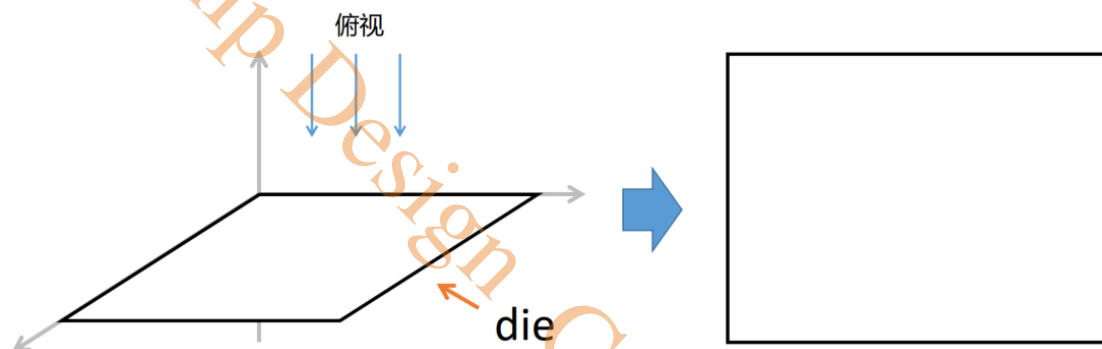


总的来说，综合的主要过程如上图所示，先将硬件描述语言转换为逻辑门电路，再将逻辑门电路转换为标准单元之间具有连接关系的电路图（网表，netlist）。

（2）版图规划

综合阶段生成了网表，现在要将标准单元放置在芯片内。

首先需要有一块可以让标准单元放置的“板子”。这块“板子”是芯片的基底，被称之为裸片（die）。

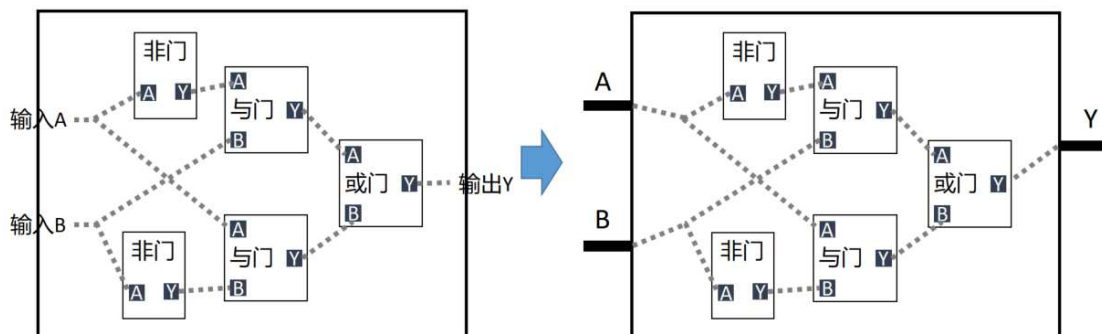


die 是一个矩形形状的平面板（在三维上），为方便接下来的描述，之后的步骤将从俯视图来进行描述。

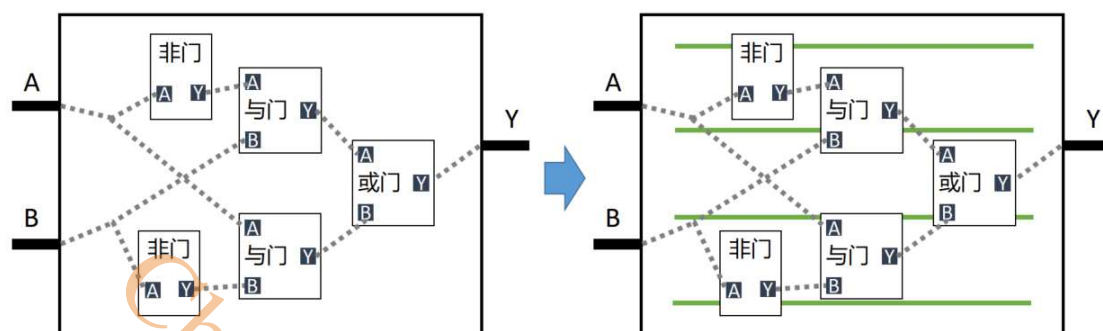
为什么是矩形形状？

在生活中，我们看到的芯片都是矩形的，之所以这么做，主要是为了提高芯片的利用率，就正如大部分房子一样，方方正正的形状可以很高效地使用空间。

如下图所示，将网表“扔”到 die 上，把网表的输入输出移到矩形边缘外，原“输入 A”、“输入 B”和“输出 Y”转变为“A”、“B”和“Y”，“ABY”是整个芯片与外部的全部交互接口，这个接口被称之为 IOPin (Input-Output pin)。



有 IOPin 后,准备给芯片布置电源网络,电源网络主要用于给标准单元供电。标准单元要能够完成正确的逻辑功能需要有电源,可以将标准单元看作电器元件,有电之后才能正常工作。下图中绿色的线条就是供电网络。这里的标准单元还不能正常工作,因为没有“插”到电源网络上,这一动作将在布局阶段完成。



总的来说,布图规划首先生成了一个芯片的基底,这个基底用于放置标准单元。随后在芯片的边缘处生成了对应于逻辑门电路的输入输出接口。最后加上电源网络。

(3) 布局

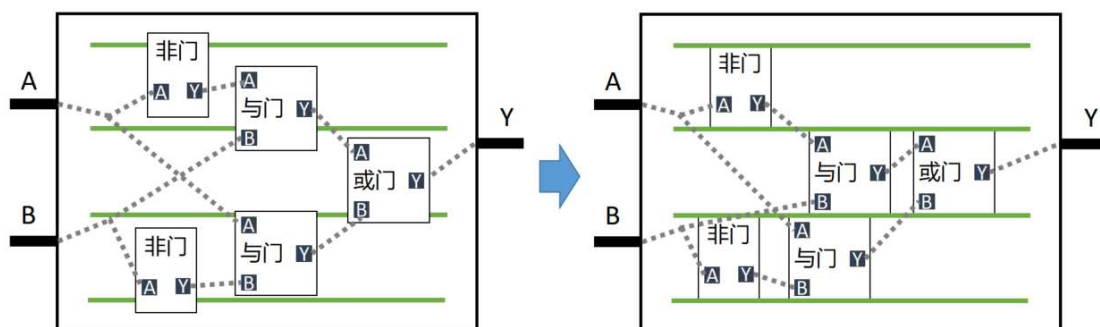
布图规划的结果并不能使得标准单元正常工作,而是要将标准单元摆放到电源网络上,看起来就像将标准单元嵌入到两根绿色条线中间。如下图所示,左图的非门是无法正常工作的,因为没有正确接入到电源网络上,而右图非门正常工作。



将所有标准单元正确地接入到电源网络上 is 布局的基本目标。如下图所示,移动标准单元并将其“嵌入”到电源网络上,并且保证单元与单元之间不能重叠。

工程师就可以完成?

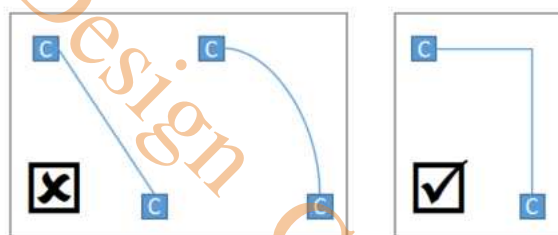
在芯片发展初期,电路规模还很小,那时候通过工程师手摆单元是可以的,但随着科技的发展,一个芯片内的逻辑门数量达到了上百万个,人力已经无法完成,这时候就需要在计算机辅助下完成。



总的来说，布局阶段的基本目标是将所有标准单元正确地接入到电源网络上，并且保证单元之间不重叠。

（4）时钟树综合

由于时钟树需要进行布线，在介绍时钟树之前，先介绍一下基本的布线原则。由于制造方面的原因，所有的线只能通过横平竖直的方式连接，否则芯片制造厂无法生产，我们把这些连线统称为“互连线（wire）”。



除了布线原则之外，时钟树中还需要用到一种新的标准单元，下面介绍这个单元的作用。

时钟信号从时钟源在互连线中传递，但是过长的互连线会使得接收端收到的信号强度非常弱，低于了有效信号强度，为了保证一定的信号强度，会在互连线中间加上类似于“泵”的标准单元，被称为“缓冲器（buffer）”。

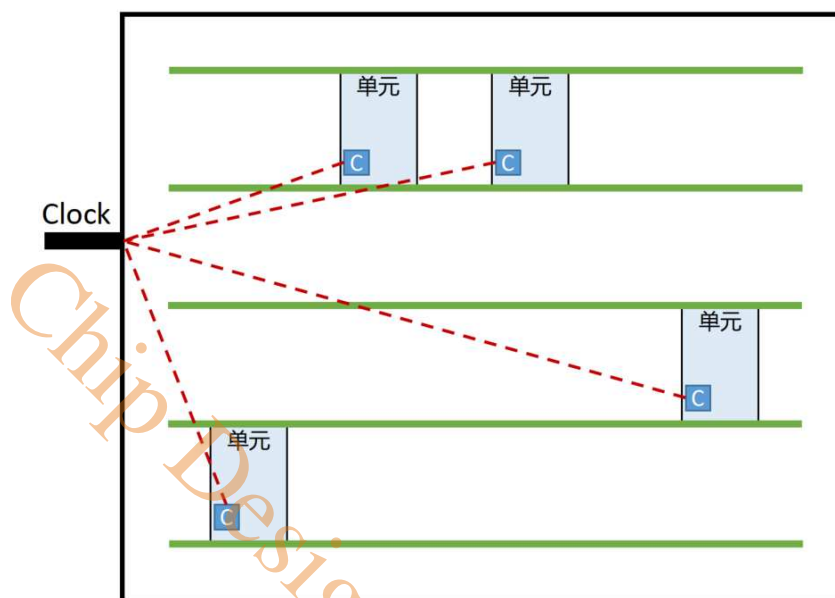


如上图所示，在接近时钟源（Clock）的信号强度很强（呈现红色），但随着信号的传递，到达单元的接收端（C）时，信号强度变弱（呈现蓝色）。信号强度弱导致单元无法正常工作。

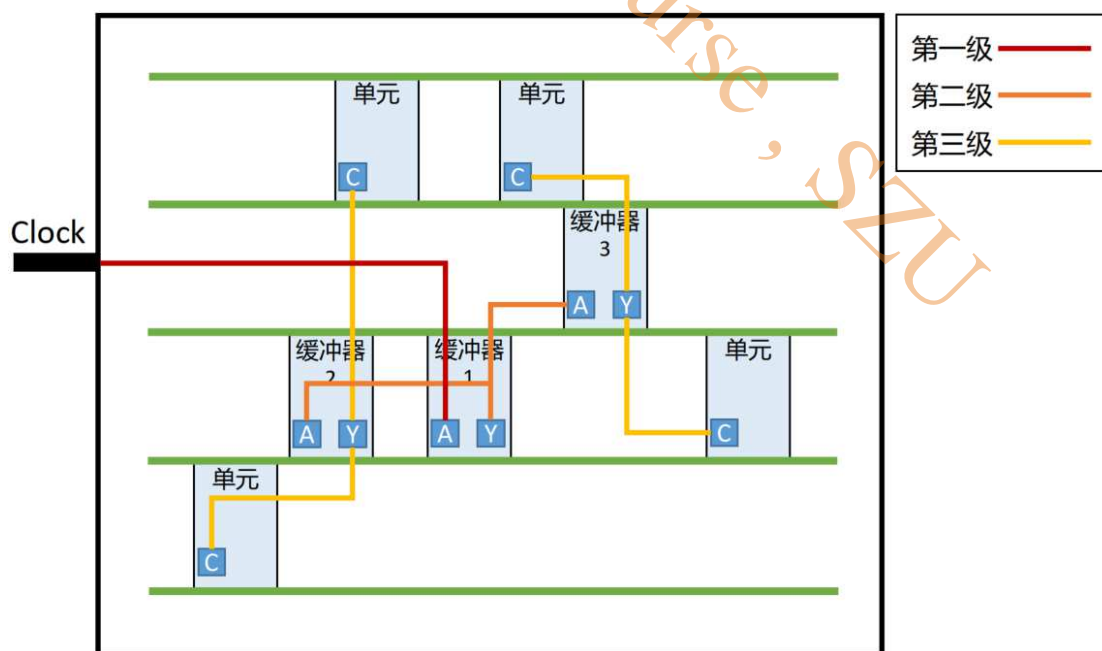


通过在互连线中添加缓冲器来加强信号强度，保证了单元的正常工

作。电路根据逻辑功能的不同被划分为两种，分别是组合逻辑电路和时序逻辑电路，由于双控开关电路没有时序，所以也就没有时钟树的概念。这里用一个新的电路例子来介绍时钟树。

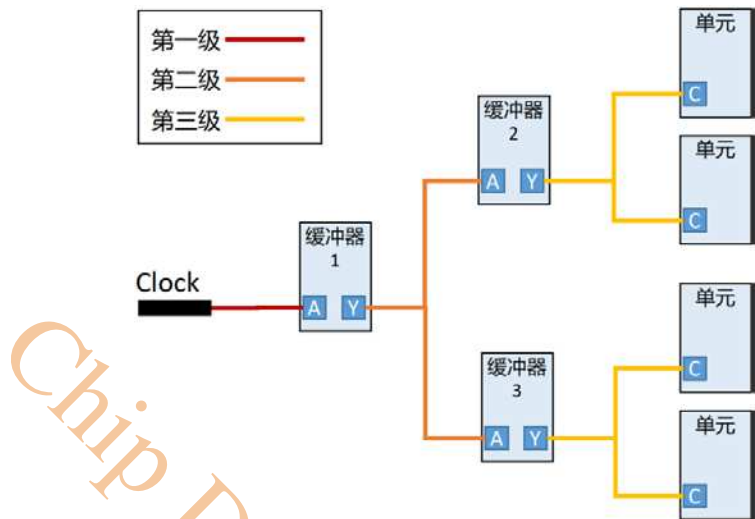


如上图所示，其中芯片时钟源（Clock）与各个单元时钟端（C）的“连接关系”用红色虚线表示。一个好的时钟电路应该使得从芯片时钟端（Clock）到各个单元时钟端（C）的长度尽量一致，同时还要保证有效时钟信号强度。



如上图所示，从 Clock 出发，到缓冲器 1 的 A 端为时钟树第一级，缓冲器 1 的 Y 端到缓冲器 2 和缓冲器 3 的 A 端为时钟树第二级，从缓冲器 2 和缓冲器 3

的Y端出发，到各个单元的C端为第三级。同时保证了Clock到各个单元C端的互连线长度基本一致。之所以被称之为时钟树，实际上通过以下方式可以更清楚认识到。

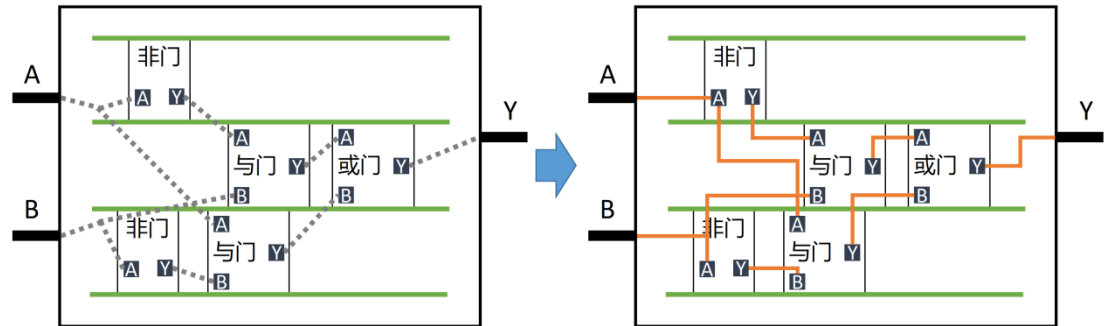


从Clock端到所有单元的长度都是一致的，实际上，时钟树的目标是从Clock端到各个单元的时延是一致的，时延的产生较为复杂，将会在之后的实验五讲到。通常在电路中，互连线长度与其时延呈正比，所以我们以长度作为目标也是没有问题的。

总的来说，时钟树综合阶段的基本目标是构建一棵从时钟源到各个单元之间路径长度一致的树，并且通过缓冲器来保证每个单元接收到的时钟信号强度是有效的。

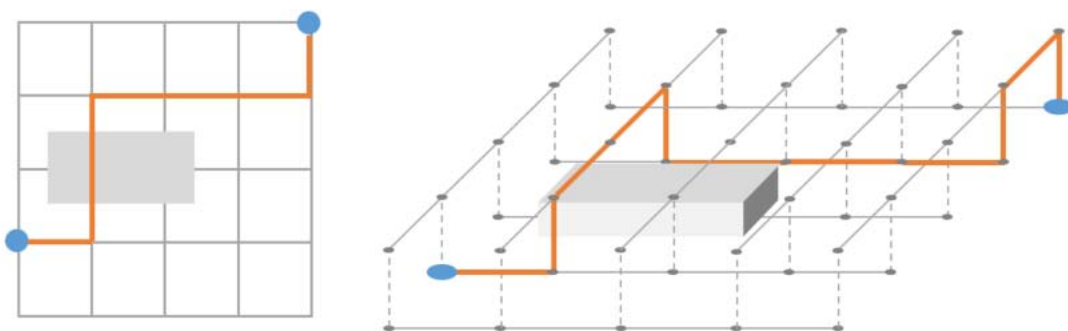
(5) 布线

布线是把已经布局好的标准单元，按照单元上的接口关系连接起来，这是布线的基本目标。由于制造方面的原因，所有的线能且只能通过横平竖直的方式连接。在连接的过程中还要保证线之间不能有重叠。



值得注意的是，芯片内部是三维立体结构，并不是一个平面，所以在俯视图

看来重叠的线段在三维上并不会。



如上图所示，虽然在俯视图上线与障碍有交叠，但在三维上是躲避了障碍的结果。和布局一样，如果不关心布线规模，这完全可以手工完成。在大规模电路下，布线所有过程中耗时最长的。

总的来说，布线的基本目标是将所有连接关系以直角线的形式连接起来，并且保证没有线与线之间的重叠。

(6) 验证

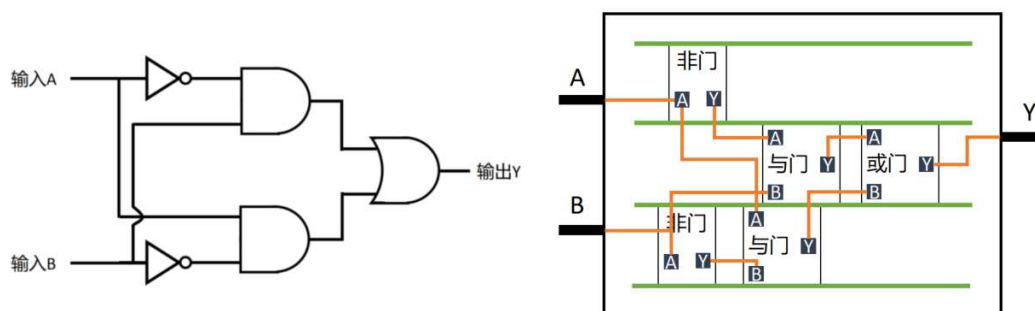
当布线完成后，设计芯片版图的基本流程就结束了。如果一切正常，就已经可以拿这个版图去生产芯片了。但是在设计过程中有时也会出现问题，这可能导致无法生产，这时候就需要进行验证。基本的验证有设计规则检查（DRC）和逻辑检查（LVS）。

设计规则检查（DRC）主要是针对一些无法制造的情况。比如单元之间有重叠，互连线之间的间距太窄。



逻辑检查（LVS）主要是针对在设计的过程中有没有将逻辑功能改变。

逻辑门电路 vs 芯片版图



(7) 生产芯片

最后将芯片版图交给芯片制造厂生产，芯片生产厂有台积电（TSMC），中芯国际（SIMC）等。生产芯片的过程也被称为“流片”。

小结

这一章了解了一颗芯片的生产需要经过设计逻辑门电路、设计芯片版图和生产芯片三个步骤。其中设计芯片版图还有六个子步骤。

综合是将逻辑门电路里的逻辑门用标准单元置换。

布图规划是规划出芯片 die 的大小，输入输出接口和电源网络。

布局是将标准单元接入到电源网络上。

时钟树综合是构建一棵树，让芯片的时钟源到所有单元的路径一致。

布线是将具有连接关系的单元用直角互连线连起来。

验证是检查在芯片设计的过程中有没有产生错误，如设计规则检查和逻辑检查

实验步骤

环境搭建

在前三章的学习中，了解了从逻辑门电路到设计芯片版图最后生产芯片的基础步骤。其中设计芯片版图是本课程的主要内容，这一设计过程可以通过一个工具完成其全流程。下面我们将学习如何安装并简单使用这个工具，简单快速地感受一下设计流程，并查看实际的芯片版图结果。

我们用到的工具名称为 iFlow，这个工具需要运行在 linux 操作系统上，由于现在的家用电脑都是 windows 操作系统，如果要使用 linux 操作系统的话，需要先在 windows 上安装虚拟机，再在虚拟机里安装 linux。下面我们先来安装虚拟机。

安装虚拟机

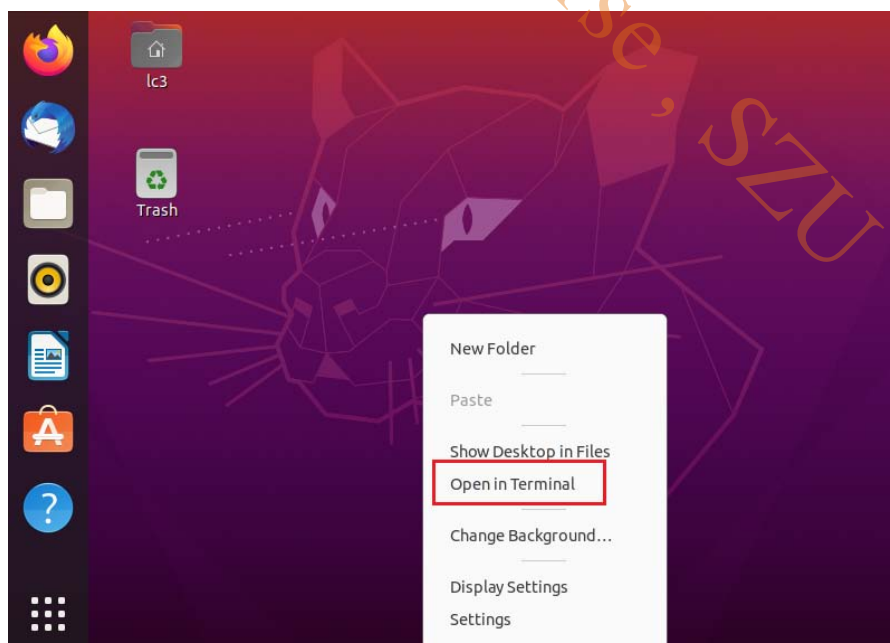
操作过程参考《附录 1》。

安装操作系统与工具

操作过程参考《附录 2》，推荐使用“导入快照”的方式安装。

运行工具

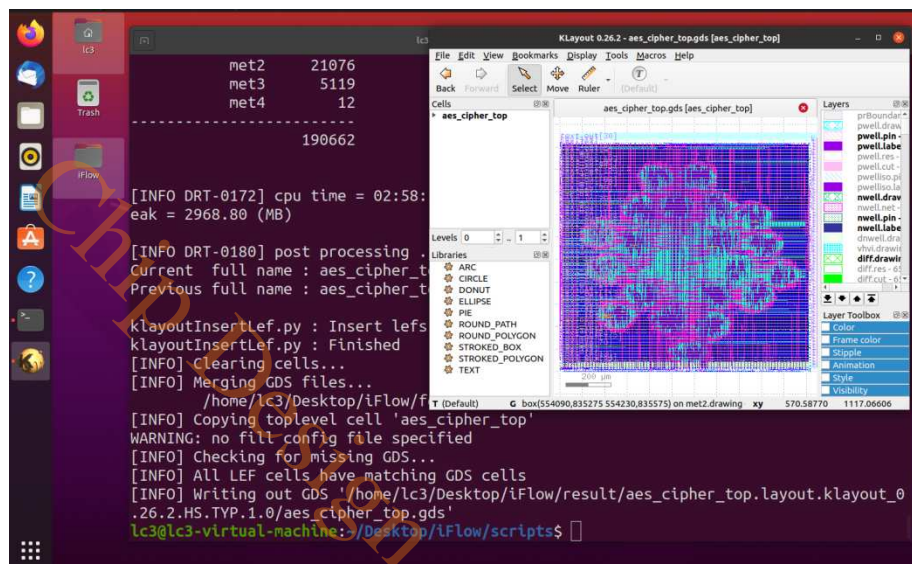
在屏幕中右键，点击 Open in Terminal



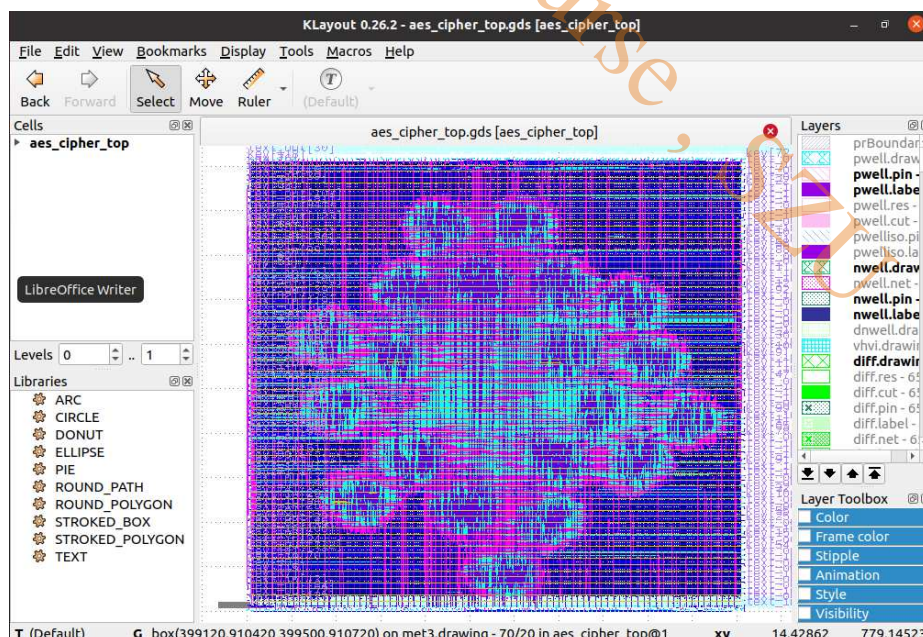
按照以下步骤，步骤 1 进入 iFlow 的运行目录。步骤 2 运行示例文件。


```
lc3@lc3-virtual-machine: ~/Desktop/iFlow/scripts
lc3@lc3-virtual-machine:~/Desktop$ cd iFlow/scripts/ 1
lc3@lc3-virtual-machine:~/Desktop/iFlow/scripts$ pwd
/home/lc3/Desktop/iFlow/scripts
lc3@lc3-virtual-machine:~/Desktop/iFlow/scripts$ ./run_flow.py -d aes_cipher_top
-s synth,floorplan,tapcell,pdn,gplace,resize,dplace,cts,filler,groute,droute,la
yout -f sky130 -t HS -c TYP -v 1.0 -l 1.0
Current full name : aes_cipher_top.synth.yosys_0.9.HS.TYP.1.0
Previous full name : aes_cipher_top...HS.TYP.1.0 2
```

成功后图示：



示例使用的是一个加密芯片的设计（设计名字为 aes_cipher_top），下图为工具运行完成后，生成的 aes_cipher_top 芯片版图（简单浏览，不用细看）。



看到这个版图后，说明环境已经搭建成功了。后续的实验将展开描述设计芯片版图每一步的动作和效果。