泰拉瑞亚数字电路文档

混乱沉睡，2025 年 12 月 12 日

目录

绪论

泰拉瑞亚作为一款冒险游戏，拥有独具特色的电路系统。本教程旨在为泰拉瑞亚玩家、其他游戏电路和现实电路爱好者，提供一个系统、全面的了解泰拉瑞亚电路的途径。本教程专注于泰拉瑞亚中的数字电路部分，不涉及过多游戏相关内容。如果你需要了解更多泰拉瑞亚的相关信息，请参考泰拉瑞亚维基。

本教程有配套的地图，并且教程中包含的电路会以地图编辑器简图的形式发布，可以在本教程项目地址查看。

本教程分为六个章节：

- 第一章 基本概念：讲解泰拉瑞亚和数字电路的一些基本概念；
- 第二章 电路基础：讲解电路基本组成和结算逻辑；
- 第三章 电路元件：讲解由单个逻辑门构成的电路元件；
- 第四章 电路部件：讲解由多个逻辑门构成的电路部件；
- 第五章 电路模块：讲解由多个电路部件构成的电路模块；
- 第六章 电路物品：讲解逻辑门以外的部分电源和用电器的用法；

在开始之前，建议安装以下模组，以帮助你更方便地进行电路制作：

- 地图编辑器 TEdit
- 模组启动器 tModLoader
- 电路步进模组 MechScope
- 辅助模组 HERO's Mod

此外，欢迎加入以下泰拉瑞亚电路社区，与其他玩家一起学习和讨论：

- 泰拉瑞亚电路爱好者交流群
- The Grand Designers
- Terraria Wiring Community

本教程参考了以下其他教程，读者可以根据需要进一步阅读以加深理解：

- 泰拉瑞亚电路文档
- The Ultimate Community Wiring Guide

同时欢迎关注我的以下平台账号，我会在这些地方发布电路作品和教程：

- 哔哩哔哩
- Steam 创意工坊
- Github

第一章 基本概念

这部分涉及一些基本概念，可以先跳过，在需要时回来阅读。

1.1 泰拉瑞亚

1.1.1 实体

实体指的是可以发生碰撞的物体，包括NPC、人物（玩家）、射弹、物品（掉落物）、图格等。

> 史莱姆对玩家造成接触伤害，就是史莱姆（NPC）与玩家（人物）的碰撞；玩家用弓射出木箭击中了史莱姆，就是史莱姆（NPC）与木箭（射弹）的碰撞；玩家被血肉墙激光击中，就是玩家（人物）与激光（射弹）的碰撞；玩家、物品、大多数 NPC、大多数射弹不能穿墙，是因为玩家、物品、NPC、射弹会和图格碰撞。

碰撞是通过碰撞箱判定的，泰拉瑞亚中所有实体的碰撞箱均为矩形，有宽度和高度两个属性。

> 例如史莱姆与玩家碰撞，是因为史莱姆的碰撞箱与玩家的碰撞箱有重叠。

1.1.2 上限

泰拉瑞亚中许多实体都有数量上限，而数量上限又分为硬上限和软上限。硬上限是游戏中的静态常量，软上限是游戏中的变量。

> 例如 NPC 硬上限 200 等；活跃刷怪数量（软上限）一般在 5 到 15 等。

从程序角度来说，硬上限是由定长数组的长度决定的，如果尝试突破会导致数组越界。为避免在正常游戏过程中出现崩溃现象，游戏程序中在关键函数中都有越界检查。而软上限是开发者对游戏的平衡控制。

> 例如 NPC 达到上限时，游戏会拒绝生成新 NPC 以防止崩溃；玩家附近的活跃敌怪数量到达 15 则不会进行刷怪。

以下是一些电路中可能用到的硬上限：

对象	上限
傀儡影子	100
NPC	200
人物	255

对象	上限
射弹	1000
冷却机关	1000

1.1.3 帧率

物理帧（tick） 指泰拉瑞亚中时间的最小单位，为 1/60 秒。在物理帧的尺度下，泰拉瑞亚是回合制游戏，每个物理帧中，除电路外所有的游戏机制会依一定顺序结算，电路结算的位置和触发电路的物理电源结算位置有关。本教程中未经特殊说明的情况下将直接用“帧”表示物理帧。

图像帧（frame）指泰拉瑞亚游戏过程中电脑显示屏更新的每帧画面，在游戏中按 F10 可以在游戏窗口左下角显示当前图像帧率。建议将跳帧设置为隐蔽以使图像帧和物理帧相等，跳帧开会在物理帧降低时通过降低图像帧以使物理帧达到最大，跳帧关会令图像帧和屏幕刷新率相等。

1.1.4 坐标

图格的坐标指的是图格在世界中的位置。坐标并非深度计与罗盘所显示的那样。程序中的坐标是以世界左上角第一个图格为 (0, 0)，横坐标向右增加，纵坐标向下增加。世界右下角的坐标为 (世界宽度 - 1, 世界高度 - 1)。

以下是不同大小世界的宽度和高度，其中大世界是泰拉瑞亚中最大的世界：

世界	宽度	高度
小世界	4200	1200
中世界	6400	1800
大世界	8400	2400

世界最外侧有一个区域在游戏中不可见且玩家不可到达的，区域内的图格仍然存在，可以在地图编辑器里面看到它们，世界上侧和左侧不可见区域大小是 41 格，下侧和右侧是 42 格。在不可见区域内的电路仍然可以正常运行，只有在世界最外侧 2 格宽的区域内的电路无法运行。

1.1.5 度量

泰拉瑞亚中的长度单位有：英里、格、英尺、像素。换算关系是 1 英里 = 5280 英尺 = 2640 格，1 格 = 2 英尺 = 16 像素。

> 尽管泰拉瑞亚的图格贴图是按照每个图格 8 × 8 像素绘制的，但是储存和计算时仍然是按照 16 × 16 像素。

> 在游戏中最大能看到 1920 × 1080 像素的范围，即便将游戏分辨率设置的更高，也会被强制缩放到这个范围内。

泰拉瑞亚中的时间单位有：帧、秒、分、天。换算关系是 1 天 = 24 分，1 分 = 60 秒，1 秒 = 60 帧。有的时候帧、秒、分也分别叫做游戏秒、游戏分、游戏时。

速度单位 = 长度单位/时间单位，主要有：英里/小时、格/秒、像素/帧。

程序内部的长度单位是像素，时间单位是帧，速度单位是像素/帧。

1.2 数字电路

1.2.1 模拟电路与数字电路

在现实中存在着许多物理量，其中有一些物理量，比如温度、湿度、压力、速度等，它们在时间上和数值上都具有连续变化的特点，在一定范围内可以取任意实数值，通常称这种连续变化的物理量为模拟量，表示模拟量的电信号称为模拟信号。而处理模拟信号的电路称为模拟电路。

还有一类物理量在时间和数值上都是离散的，它们的大小以及每次的增减变化都是某个最小单位的整数倍。比如每年的利润，若最小单位为元，显然它只能以元为单位增加或减少，并且以年为时间单位记录。这一类物理量称为数字量，表示数字量的电信号称为数字信号。而处理数字信号的电路称为数字电路。

> 泰拉瑞亚的电路中并没有电压和电流这样的模拟量，电源、用电器和电线的状态在时间上和数值上都是离散的，泰拉瑞亚中只存在数字电路。

在数字电路中，最常用的是只由 0 和 1 两种数值组成的数字信号，这种二值数字信号又被称为二进制信号，这类信号中的数值 0 或 1 可以用电平的低或高来表示，也可以用脉冲的有或无来表示。数字电路中最小时间单位被称为周期，电路中所有变化都会以周期为单位发生。

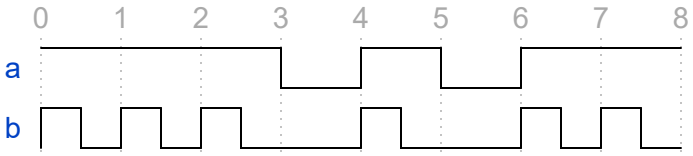
> 泰拉瑞亚逻辑门电路的周期被称为逻辑帧，一个物理帧内可以有任意数量的逻辑帧。一个常见的错误是认为逻辑帧与物理帧相等，实际上在泰拉瑞亚中任意规模的电路都能在瞬间（一个物理帧内）计算完成，详细内容会在后文介绍。

> 后文中的在描述电路时使用的“同时”指的是指某些事情在同一个周期内发生，它们的先后关系对结果没有影响。

数字电路中处理的电信号一般有两种，电平（状态）型数字信号和脉冲（激活）型数字信号。状态信号会在每个周期内维持一个状态（电平）不变，只有在周期间才会发生变化，一般设定高电平为 1、低电平为 0，所以又叫不归零型数字信号，通常关注状态信号在周期内的状态。

脉冲是一种特殊的信号，指电平在短时间内（小于一个周期）突变，随后又迅速返回初始值的变化过程。对于激活信号，一般设定有脉冲为 1、无脉冲为 0，在每个周期中除了有脉冲的瞬间外，会一直维持低电平，所以又叫归零型数字信号，通常关注周期内是否有脉冲存在。

> 泰拉瑞亚的电线中传递的只有激活信号，而在电源和用电器中存在状态信号和激活信号，其中逻辑门电路中只有 0 和 1 两种数值所组成的二进制数字信号。



> 上图为两个数字信号 a 与 b 的波形图，信号内容都是 $(11101011)_2$ ，波形图是反映信号的幅值（电平）随时间的变化图像。
>
> - 信号 a 是以高电平代表 1 ，低电平代表 0 ，称为状态（电平）型数字信号或不归零型数字信号；
> - 信号 b 是以有脉冲代表 1 ，无脉冲代表 0 ，称为激活（脉冲）型数字信号或归零型数字信号；
>
> 脉冲除了是方波以外，还可以是三角波、正弦波、尖顶余弦等等，我们并不关心其波形与持续时间，只关心它是否存在。

1.2.2 进制

按进位规则进行计数称为进位计数制，简称进制。在日常生活中我们最广泛使用的是十进制（Decimal），十进制有 0 ~ 9 十个数字。但在数字电路和计算机中，通常只有类似灯的亮灭、电线的通断、开关的闭合和抬起，这样的两种状态，所以只需要 0 和 1 两个数字来表示，使用的是二进制（Binary）。

> 在泰拉瑞亚逻辑门电路中，只有二进制信号，所以其它类型的信号，比如数字、文字、图像、视频、程序等信息，需要先使用特定的编码转换为二进制信号，才能在电路中传输和处理。处理完成后，由需要将二进制信号转换为其它类型的信号进行表示。

十进制数可以被写做 $(345)_{10}$ ，读做“三百四十五”，这种使用 0 ~ 9 十个数字，每个数字在不同的数位上表示的大小不同的形式，被称为位置记数法。

> 下角标 10 表示括号里的数是十进制，下角标 2 表示括号里的数是二进制，以此类推。

在十进制中，由于只有 0 ~ 9 十个数字，当表示大于 9 的数时，需要进行进位，也就是满 10 进 1 ， 9 + 1 = 10 。当 1 在十位上时，它是个位上的 1 的十倍。对于十进制数 $(345)_{10}$ ，3 在百位上，4 在十位上，5 在个位上；3 代表有三个 100，4 代表有四个 10，5 代表有五个 1。这里的 100、10、1 ，也就

是 10^2、10^1、10^0 是十进制数的位权，十进制数的各个数位的位权是 10 的位数次幂，幂数的底 10 被称为基数，几进制的基数就是几。十进制数第 n 位数的位权是 10^{n-1}。

$$(345)_{10} = 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

> 这种把一个数字的每一数位拆开，写成数码乘以位权的形式，被称为位权展开式。

同理，二进制中满足满 2 进 1 的进位规则，二进制数也可以按照位置记数法，被写做形如 $(10010)_2$ 的形式，表示有 1 个 16，0 个 8，0 个 4，1 个 2，0 个 1。每位的乘数 16、8、4、2、1，也就是 2^4、2^3、2^2、2^1、2^0 是二进制数的位权，而二进制数的基数是 2。二进制数第 n 位数的位权是 2^{n-1}。R 进制数第 n 位数的位权是 R^{n-1}

$$(10010)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

> 任何进制的数字都可以写成位权表达式的形式，计算位权表达式的同时也能将其它进制的数字转换为十进制，而与之对应的，也可以使用短除法将十进制转换为其它进制的数字，详细内容会在后文介绍。

$$(BEEF)_{16} = 11 \times 16^3 + 14 \times 16^2 + 14 \times 16^1 + 15 \times 16^0$$

> 这是一个十六进制数字 $(BEEF)_{16}$ 的位权展开式，在十六进制中，会使用 A ~ F 来表示 10 ~ 15 六个数字。

$$(a_{n-1}a_{n-2}\dots a_1a_0.a_{-1}\dots a_{-m+1}a_{-m})_R = \sum_{i=-m}^{n-1} a_i \times R^i$$

> 这是一个具有 n 位整数和 m 位小数的 R 进制实数的位权展开式。

对于 3 位十进制数，可以表示 $(000)_{10}$ ~ $(999)_{10}$ ，共计 1000 （10^3）个数，也就是第四位的位权。同理对于五位二进制数，可以表示 $(00000)_2$ ~ $(11111)_2$ ，共计 32 （2^5）个数。由此可见，对于 n 位二进制数，可以表示 2^n 个数；对于 n 位 R 进制数，可以表示 R^n 个数。

1.2.3 常用进制

10 并不是 2 的整数次幂，在十进制和二进制间转换是比较困难的。虽然在电路中使用的都是二进制，但是二进制的数码比较少，数字会比较长，难以阅读。而八进制和十六进制的基数是二的整数次幂，数码的数量相对较多，是电路和计算机中另外两个比较常用的进制。

进制	规则	数码	基数	位权
二进制	逢二进一	0 ~ 1	2	2^{n-1}
八进制	逢八进一	0 ~ 7	8	8^{n-1}

进制	规则	数码	基数	位权
十进制	逢十进一	$0 \sim 9$	10	10^{n-1}
十六进制	逢十六进一	$0 \sim 9, A \sim F$	16	16^{n-1}

八进制和十六进制的基数分别是 8 和 16，所以每一位八进制和十六进制可以直接转换为 3 位和 4 位二进制，同理每 3 位和 4 位二进制也可以直接转换为一位八进制和十六进制。

> 例如 $(1110\_0000\_0000\_0010)_2 = (E002)_{16}$，$(101\_110\_111)_2 = (567)_8$。

具体的对应关系可以在下方的速查表中找到，使用速查表可以快速的完成二进制与八进制与十六进制数字间的转换。

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

对于十进制，可以将每一位十进制直接转换为 BCD（Binary-Coded Decimal）码，这种编码将十进制的 0 ~ 9 使用四位二进制来表示。

> 例如十进制数 1857 可以使用 8421 BCD 码表示为 $0001\_1000\_0101\_0111$ 。

具体的对应关系可以在下方速查表中找到，使用速查表可以快速的完成十进制数字与 BCD 码间的转换。

十进制	8421 码	5421 码	余 3 码	格雷码
0	0000	0000	0011	0000
1	0001	0001	0100	0001
2	0010	0010	0101	0011
3	0011	0011	0110	0010
4	0100	0100	0111	0110
5	0101	1000	1000	0111
6	0110	1001	1001	0101
7	0111	1010	1010	0100
8	1000	1011	1011	1100
9	1001	1100	1100	1101

> 还有一些 BCD 码未列出（如 2421 码和余 3 循环码），读者可自行了解。最常用的 BCD 码是 8421 BCD 码，后文中提到的 BCD 码默认是 8421 BCD 码。

> 对于二进制、八进制、十六进制数字，除了使用下标 2、8、16，还可以使用前缀 0b、0o、0x，或后缀 B、O、H 来表示。十进制无前缀，后缀是 D。
>
> 下文中十进制数字不会使用特殊表示，其它进制数字会使用前缀表示。

进制	英文	前缀	后缀
二进制	Binary	0b	B
八进制	Octal	0o	O
十进制	Decimal	无	D

进制	英文	前缀	后缀
十六进制	Hexadecimal	0x	H

1.2.4 逻辑函数

数字电路的基本单元是开关器件，开关器件只有“接通”和“断开”两种状态，可以使用逻辑变量（二值变量）来描述，逻辑变量只有 0 和 1 两种可能的取值，也就是一位二进制。

函数是一种将一个集合中的每个元素（自变量）与另一个集合中的一个元素（因变量）关联起来的关系，逻辑函数是研究逻辑变量之间的因果关系的函数。对于有 n 个逻辑变量 $(A_1, A_2, \cdots, A_n)$ 的逻辑函数，当 n 个逻辑变量取任意一组确定值后，逻辑函数 $Y = F(A_1, A_2, \cdots, A_n)$ 的值也就被唯一地确定了，显然 Y 也只有 0 或 1 两种可能的取值。

> 例如，存在一个两变量的逻辑函数 $Y = F(A, B)$ ，对于二位二进制，存在 $2^2 = 4$ 个不同的取值，所以对于变量 A，B 有 4 种可能的取值：00，01，10，11。令 $F(0, 0) = 0$， $F(0, 1) = 0$ ， $F(1, 0) = 0$， $F(1, 1) = 1$ ，可得 A，B 与 Y 的完整映射关系。
>
> 输入为 00 时，可知输出为 0；
> 输入为 01 时，可知输出为 0；
> 输入为 10 时，可知输出为 0；
> 输入为 11 时，可知输出为 1。

对于 n 变量的逻辑函数，存在 2^n 个可能的取值，当我们建立起全部可能的自变量取值与对应的因变量的完整映射关系，即可唯一确定逻辑函数。

1.2.5 真值表

对于一个逻辑函数，我们可以使用真值表来描述它。真值表是一种列出一个逻辑函数的所有输入与其对应输出的表格。例如上一节所述函数，我们可以使用下面的真值表来描述：

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

> 输入为 00 时，查表可知输出为 0；
> 输入为 01 时，查表可知输出为 0；

> 输入为 10 时，查表可知输出为 0；
> 输入为 11 时，查表可知输出为 1。

对于 n 个输入的逻辑函数，会有 2^n 个可能的输入取值，和与之对应的输出，真值表会有 $n+1$ 列，2^n 行。

有时候部分输入情况是不存在或不需要考虑的，此时我们可以省略这些情况对应的行，或者将对应输出用 X 表示。

> 例如，独热码是一种全部位中只有一位为 1 的二进制编码，所以对于一个输入为独热码的逻辑函数来说，非独热码的输入情况是不需要考虑的。
>
> | A | B | C | Y |
> | --- | --- | --- | --- |
> | 0 | 0 | 0 | X |
> | 0 | 0 | 1 | 0 |
> | 0 | 1 | 0 | 0 |
> | 0 | 1 | 1 | X |
> | 1 | 0 | 0 | 1 |
> | 1 | 0 | 1 | X |
> | 1 | 1 | 0 | X |
> | 1 | 1 | 1 | X |
>
> 也可以表示为：
>
> | A | B | C | Y |
> | --- | --- | --- | --- |
> | 0 | 0 | 1 | 0 |
> | 0 | 1 | 0 | 0 |
> | 1 | 0 | 0 | 1 |

1.2.6 基本逻辑运算

类似加、减、乘、除四则运算，在逻辑函数中，存在最基本的逻辑运算：与、或、非。任何逻辑运算和任何逻辑函数都可以使用这三种运算来表示。

与（AND）

与运算用符号“ · ”表示，称为与。

与运算是二元运算，在所有输入都为 1 时，输出才为 1，有任一输入为 0 时输出为 0。

两个变量与运算的逻辑表达式为：

$$A \cdot B$$

> 读作 A 与 B。

两个变量与运算的真值表为：

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

n 个变量的与运算的逻辑表达式为：

$$A_1 \cdot A_2 \cdot \cdots \cdot A_n$$

> 多个变量相与与多个变量两两相与等价。

$$A_1 \cdot A_2 \cdot \cdots \cdot A_n = (\cdots((A_1 \cdot A_2) \cdot A_3) \cdot \cdots \cdot A_n)$$

或（OR）

或运算用符号“ $+$ ”表示，称为或。

或运算是二元运算，在有任一输入为 1 时，输出就为 1，全部输入都为 0 时输出为 0。

两个变量或运算的逻辑表达式为：

$$A + B$$

> 读作 A 或 B

两个变量或运算的真值表来为：

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

n 个变量的或运算的逻辑表达式为：

$$A_1 + A_2 + \cdots + A_n$$

> 多个变量相或与多个变量两两相或等价。

$$A_1 + A_2 + \cdots + A_n = (\cdots((A_1 + A_2) + A_3) + \cdots + A_n)$$

非（NOT）

非运算用符号“ ′ ”表示，称为非，非运算是一元运算。在输入为 0 时，输出 1；在输入为 1 时，输出 0。

> 非运算也叫反相运算，或者取反运算。

非运算的逻辑表达式为：

$$A'$$

> 读作 A 非，非运算只能有一个逻辑变量输入。

非运算的真值表为：

输入 A	输出 Y
0	1
1	0

1.2.7 逻辑表达式

上一节介绍了与、或、非三种基本逻辑运算，以及对应的运算符号“ · ”，“ ＋ ”，“ ′ ”，它们统称为逻辑运算符。由逻辑变量和逻辑运算符组成的表达式被称为逻辑函数表达式或逻辑表达式，简称为逻辑式。它是描述逻辑函数与逻辑变量之间关系的表达式，逻辑式和真值表都是描述逻辑函数的重要工具。

有一些较为复杂的逻辑式也是由基本运算符组合而成的。在逻辑式中，相同运算符的运算次序是 **先左后右**，而不同运算符的运算次序是 **非 > 与 > 或**，若有括号，则先进行括号内运算。

> 例如逻辑式 $A' \cdot B + A \cdot B'$ ，读作 A 非与 B 或 A 与 B 非。计算时先计算两个非运算，然后再计算两个与运算，最后再计算或运算。

设有两个逻辑函数：

$$F_1(A_1, A_2, \cdots, A_n)$$

$$F_2(A_1, A_2, \cdots, A_n)$$

如果对于 A_1 ~ A_n 的任何一组取值使 F_1 和 F_2 具有相同的值，则称这两个逻辑函数相等或等价，既 $F_1 = F_2$。相等的逻辑函数一定有相同的真值表，反之亦然。

> 对于上述逻辑表达式 $A' \cdot B + A \cdot B'$
>
> 输入 00 时 $0' \cdot 0 + 0 \cdot 0' = 1 \cdot 0 + 0 \cdot 1 = 0 + 0 = 0$；
> 输入 01 时 $0' \cdot 1 + 0 \cdot 1' = 1 \cdot 1 + 0 \cdot 0 = 1 + 0 = 1$；
> 输入 10 时 $1' \cdot 0 + 1 \cdot 0' = 0 \cdot 0 + 1 \cdot 1 = 0 + 1 = 1$；
> 输入 11 时 $1' \cdot 1 + 1 \cdot 1' = 0 \cdot 1 + 1 \cdot 0 = 0 + 0 = 0$。
>
> 进行与、或、非运算时可以通过查上一节的真值表得到结果。

1.2.8 其它常用逻辑运算

在电路中，除了与、或、非三种基本逻辑运算外，也有一些常用的逻辑运算。

与非（NAND）、或非（NOR）

与非运算是对与运算的结果进行取反，而或非运算是对或运算的结果进行取反。

两个变量与非运算的逻辑表达式是：

$$(A \cdot B)'$$

两个变量或非运算的逻辑表达式是：

$$(A + B)'$$

> 与非运算和或非运算不使用新的运算符。

两个变量与非运算的真值表是：

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

两个变量或非运算的真值表是：

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

n 个变量的与非运算的逻辑表达式为：

$$(A_1 \cdot A_2 \cdot \cdots \cdot A_n)'$$

n 个变量的或非运算的逻辑表达式为：

$$(A_1 + A_2 + \cdots + A_n)'$$

异或（XOR）

异或是一种非常常用的运算，在非常多的地方会用到。异或运算用符号“^”表示，称为异或。

异或运算有两种类型，一种被称为“两两异或”，而另一种被称为“多输入异或”。在输入只有两个时，两两异或与多输入异或等价。

> 在泰拉瑞亚中，两两异或不需要逻辑门就能完成，所以也被称为“线异或”，是最常用的运算；而泰拉瑞亚的异或逻辑门逻辑是多输入异或，与线异或不同，详细内容会在后文介绍。

两两异或运算是二元运算，每次只考虑全部输入中的两个输入，在两个输入中只存在一个 1 时，输出才为 1；两个输入都为 1 或都为 0 时，输出为 0。

> 两输入异或运算也叫奇校验，只有在输入有奇数个 1 时输出才为 1，否则为 0。

多输入异或是多元运算，需要对全部输入进行考虑，在全部输入中只存在一个 1 时，输出才为 1，其余全部情况的输入，输出都为 0。

> 多输入异或运算也叫独热码检测，只有在输入为独热码时输出才为 1，否则为 0。

两个变量的两两异或的逻辑表达式是：

$$A \hat{} B$$

两个变量的多输入异或的逻辑表达式是：

$$\hat{}(A, B)$$

两个变量异或的逻辑表达式也可以用与、或、非来表示：

$$A' \cdot B + A \cdot B'$$

> 两者等价的证明已在上一节完成。

两个变量的异或的真值表是：

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

n 个变量的两两异或运算的逻辑表达式为：

$$A_1 \hat{} A_2 \hat{} \cdots \hat{} A_n$$

> 多个变量两两异或和多个变量分别两两异或等价。

$$A_1 \hat{} A_2 \hat{} \cdots \hat{} A_n = (\cdots((A_1 \hat{} A_2) \hat{} A_3) \hat{} \cdots \hat{} A_n)$$

n 个变量的多输入异或运算的逻辑表达式为：

$$\hat{}(A_1, A_2, \cdots, A_n)$$

> 多个变量多输入异或时需要同时考虑所有输入。

> 例如，三个变量的两两异或的真值表是：

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

> 三个变量的多输入异或的真值表是：

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

> 可以发现在三个输入全为 1 时两两异或和多输入异或结果不同。

同或（XNOR）

而同或和异或的关系类似与非和与的关系，同或的结果是异或的结果取反。

两个变量的两两同或的逻辑表达式是：

$$(A \hat{} B)'$$

> 两输入同或运算也叫偶校验，只有在输入有偶数个 1 时输出才为 1，否则为 0。

两个变量的多输入同或的逻辑表达式是：

$$\curlywedge(A,B)'$$

两个变量同或的逻辑表达式也可以用与、或、非来表示：

$$A\cdot B+A'\cdot B'$$

两个变量的同或的真值表是：

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

n 个变量的两两同或运算的逻辑表达式为：

$$(\cdots((A_1\curlywedge A_2)'\curlywedge A_3)'\curlywedge\cdots\curlywedge A_n)'$$

n 个变量的多输入同或运算的逻辑表达式为：

$$\curlywedge(A_1,A_2,\cdots,A_n)'$$

第二章 电路基础

2.1 电源、电线、用电器

2.1.1 电路激活过程

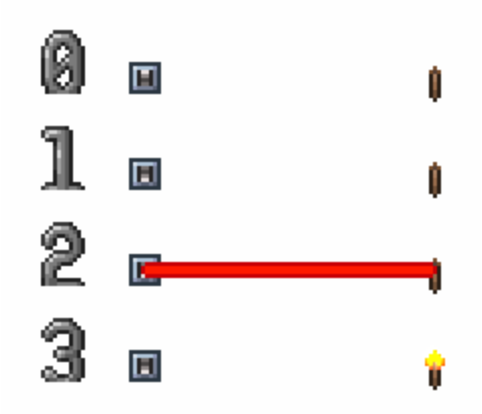
在泰拉瑞亚中，电路由电源、电线、用电器组成，电源在满足条件时会产生一个脉冲，经电线传导到用电器，用电器响应该信号。

> 电路由电源（开关）、电线（红色电线）、用电器（火把）组成。

电源或用电器处于图格图层，而电线在单独的电线图层，与电源或用电器处于不同的图层，它们可以重叠。如果某电线与某电源或用电器重叠，我们说该电源或用电器在该电线下，该电线在该电源或用电器上。

> 电源（开关）和用电器（火把）位于图格图层，电线（红色电线）位于电线图层。

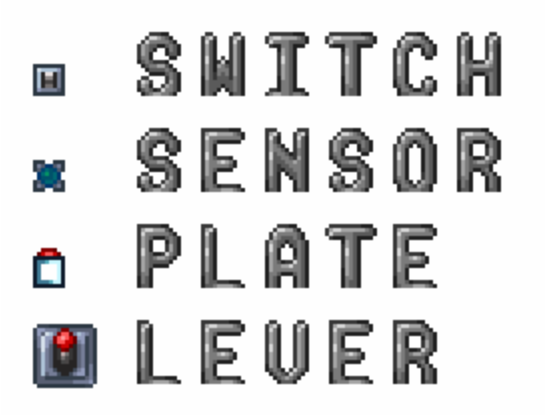
接下来的讨论中，将会用激活来表示电路的活跃状态：**电源激活** → **电源上的电线激活** → **电线下的用电器激活**。电源的激活指电源通过其上电线发出信号，电线的激活指其传递其下电源发出的信号到其下用电器，用电器的激活指其上电线被激活。



> ① 电源（开关）激活 → ② 电源上的电线（红色电线）激活 → ③ 电线下的用电器（火把）激活

2.1.2 电源和用电器

电源指可以激活其上电线的图格，每个电源有特定的激活条件。逻辑门被称为逻辑电源，除了逻辑门以外的电源被称为物理电源。物理电源可以将非电信号转换为电信号，是逻辑电路的输入端口。只有物理电源可以新建逻辑结算，逻辑电源不能新建逻辑结算。

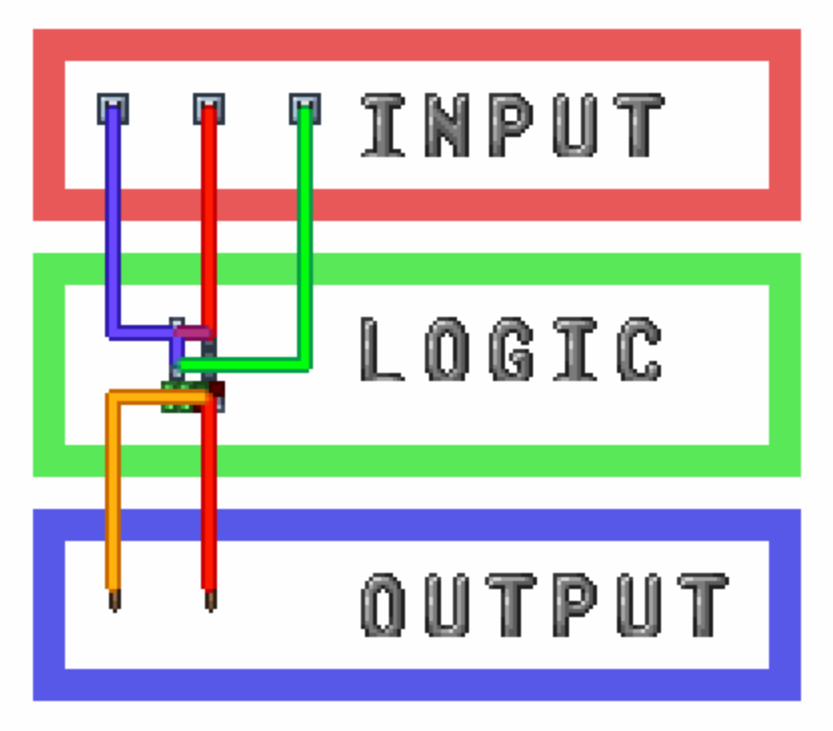


> 如开关、拉杆、压力板、逻辑感应器等。

用电器指可被其上电线激活的图格，每个用电器有特定的响应方式。逻辑门灯被称为逻辑用电器，除了逻辑门灯以外的用电器被称为物理用电器。物理用电器可以将电信号转换为非电信号，是逻辑电路的输出端口。

> 如火把、传送机、传送枪站、机关等。

而只由电线、逻辑电源（逻辑门）、逻辑用电器（逻辑门灯）组成的电路被称为逻辑电路。电路是由 **输入端口、逻辑电路、输出端口** 组成的。



> 输入端口（开关）、逻辑电路（逻辑门、逻辑门灯、电线）、输出端口（火把）。

> 存在即是电源又是用电器的图格，如计时器。

> 物理电源和物理用电器的激活条件和响应方式可查阅附录。

多图格电源和用电器

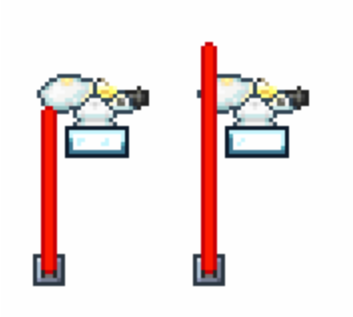
对于由多个图格组成的电源，并不会因为一根电线连接该电源的多个图格而增加电源激活时该电线的激活次数，电线只连接多图格电源的一个图格和多个图格是一样的。在电源激活时，其上的每根电线都只会被激活一次。

> 左右两种连接方式完全一样，尽管第二种连接了更多图格。

> 但不同颜色的电线或同颜色但中间没有连接的电线不是一根电线，在电源激活时，其上的每根电线都会被激活一次。

> 激活拉杆时，左边的火把会被红线激活一次，点亮；中间的火把会被蓝线激活一次，点亮；右边的火把会被蓝线激活一次，红线激活一次，先点亮后熄灭（过程不可见）。

对于由多个图格组成的用电器，并不会因为一根电线连接该用电器的多个图格而增加该电线激活时该用电器的响应次数，电线只连接多图格用电器的一个图格和多个图格是一样的。在一根电线激活时，其下的用电器只会响应一次。



> 左右两种连接方式完全一样，尽管第二种连接了更多图格。

> 但不同颜色的电线或同颜色但中间没有连接的电线不是一根电线，在激活用电器时，用电器会分别响应每一根电线上的每一次信号。

> 激活开关时，左边的开关只连接了一种颜色的电线，炮台只旋转一次；右边的开关连接了四种颜色的电线，炮台会旋转四次（过程不可见）。

有些用电器的响应效果会根据激活图格的位置不同而改变。

> 传送枪站、大炮、兔兔炮等多图格用电器会因为激活的图格位置不同而响应不同。

有冷却时间的用电器

有些用电器的响应会受到冷却时间限制，在冷却时间结束前被激活不会响应。

> 如炮台发射、机关、雕像、马桶等。

有冷却时间的用电器被激活并响应后，用电器所在的所有图格坐标与冷却时间（帧）会被加入到所有有冷却时间的电路物品共用的冷却机关列表（多图格用电器的每个格子都会被加入）。

每帧冷却机关列表内的所有图格的冷却时间会减一，在图格冷却时间为零后，冷却时间结束，该坐标会从列表中移除。

> 有冷却时间的用电器的名字和冷却时间可以在附录查询。

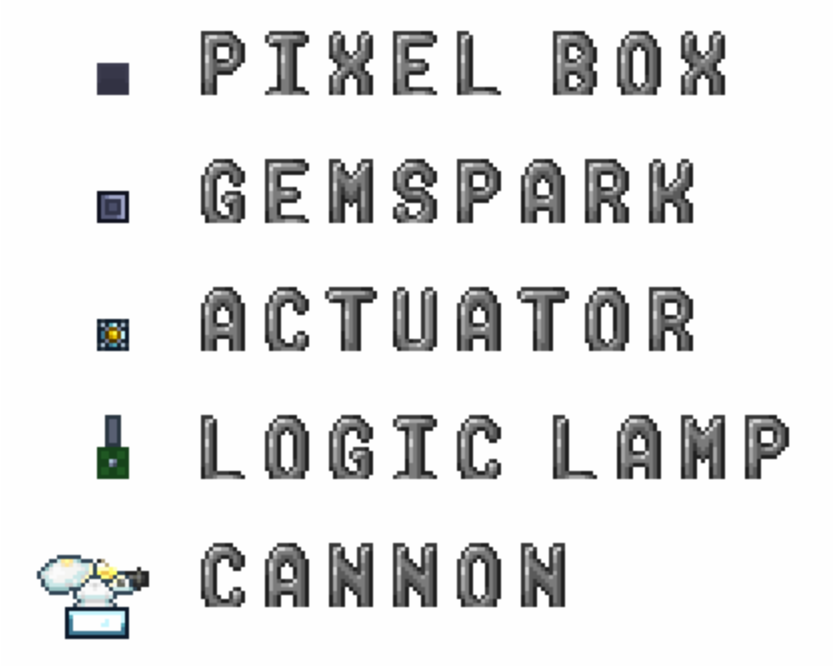
当列表已满或用电器的所有图格都在冷却列表中时（多图格用电器需要所有图格都在列表里），用电器被激活后不会响应。

> 计时器也使用冷却时间列表记录下一次激活的时间，但开启和关闭并不受冷却时间影响。列表满后开启的计时器不会激活。

将正在冷却的用电器挖掉并不会从列表中移除该用电器的图格，但挖掉计时器会将计时器从列表中移除，计时器不会再激活。

> 对于冷却时间相同的项，冷却机关列表是后进先出的，计时器电路中会使用这个性质。

没有冷却时间的用电器可以在一帧内被激活任意次数。



> 如逻辑门灯、炮台转向、像素盒、晶莹宝石块、促动器等。

状态表示

有些电源看起来有两个或多个状态，并且每次状态切换会激活其上电线，这种电源被称为状态表示电源，简称状态电源。状态电源在不满足条件到满足条件和满足条件到不满足条件两种情况时都会激活。

> 如加重压力板有抬起和按下两种状态，在按下到抬起和抬起到按下时都会激活；普通逻辑门有亮和灭两种状态，在灭切换到亮和亮切换到灭都会激活。

> 加重压力板按下时会激活一次火把，加重压力板抬起时也会激活一次火把。

有些用电器看起来有两个或多个状态，其上电线激活时会在几个状态间切换，这种用电器被称为状态表示用电器，简称状态用电器。状态用电器是有记忆的，它会记住之前的状态，下一次状态变化会在当前状态的基础上变化。

> 如火把和普通逻辑灯会在激活时由亮切换到灭，或由灭切换到亮，具体是哪种取决于激活前的状态。

> 火把被激活第一次由灭变亮，激活第二次由亮变灭。

对于有两个状态的电源或用电器，我们可以人为设定一个状态为 0（一般指灭或初始态），另一个状态为 1（一般指亮或另一态）。在不激活时，状态电源和用电器会维持之前的状态不变。

> 也有一些状态电源和用电器不止两个状态，比如烟囱有三个状态、彩线灯泡有十六个状态、传送枪台有十八个状态。

我们称只使用状态电源、状态用电器和电线的电路为状态表示电路，简称状态电路。状态表示电路的状态可以从电路中直接读出，所有电源和用电器的状态是同步变化的。

> 如果认为开关向上/火把灭为 0 ，开关向下/火把亮为 1，则开关和火把的状态同步变化，同时为 0 或同时为 1。

激活表示

有些电源看起来只有一个状态，或者看起来有多个状态但是只在一个状态切换到另一个状态时会激活，反过来不会，这种电源被称为激活表示电源，简称激活电源。激活电源只在不满足条件到满足条件时激活，不会在满足条件到不满足条件时激活。

> 如普通压力板尽管看起来有抬起和按下两种状态，但是只会在抬起到按下时会激活，按下到抬起不会激活，所以是激活电源；而故障逻辑门看起来只有一种状态，是激活电源。

> 压力板按下一次时会激活一次火把，压力板按下两次时会激活两次火把。

有些用电器看起来只有一个状态，这种用电器被称为激活表示用电器，简称激活用电器。激活用电器是没有记忆的，激活时会直接响应而不考虑之前的状态。

> 如广播盒会在被激活时发出内部文本，传送器会在被激活时传送上方人物与 NPC。

> 广播盒被激活发送文本。

对于激活电源和用电器的每个周期，每个电源和用电器都有激活和不激活两种状态，我们可以令激活态为 1，不激活态为 0。在不激活时，激活电源和用电器会恢复并维持稳态（不激活态）。

> 这里的周期指的是电路结算的最小单位，对于逻辑电路来说是逻辑帧。

我们称只使用激活电源、激活用电器和电线的电路为激活表示电路，简称激活电路。激活与否并不会改变激活电源和激活用电器的状态，所以电路的状态不能从电路中直接读出。

> 我们看不到激活电源和用电器的状态，只能通过其它方式间接判断。

> 如果我们忽略状态电路的状态，那么也可以将状态电路看做激活电路。

> 我们可以忽略开关的状态，把状态电源开关当做激活电源使用。

最简形式

上一节提到电路是由物理输入端口、逻辑电路、物理输出端口组成的，而逻辑电路是由逻辑电源（逻辑输入端口）、逻辑用电器（逻辑输出端口）、电线组成的。所以我们可以得到电路的最简形式：一根电线与其下所有的电源和用电器。这样的电路被称为最简形式电路，简称最简电路，任何电路都是由这样的一个或多个最简电路组成。

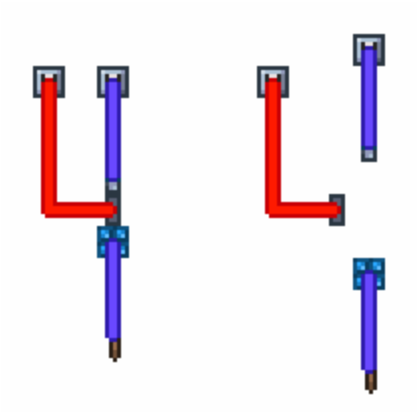
> 最简电路，由一根红色电线（电线）、三个开关（电源）、三个火把（用电器）组成。

由于大部分电路同时包括状态表示和激活表示，所以对于电路状态表示和激活表示的讨论通常对最简电路进行。当最简电路中只有状态电源、状态用电器、电线时，被称为最简状态电路；当最简电路中只有激活电源、激活用电器、电线时，被称为最简激活电路。

> 最简状态电路和最简激活电路。

当最简电路的电源中存在激活电源，用电器中存在状态用电器，则该最简电路存在激活转状态；当最简电路的电源中存在状态电源，用电器中存在激活用电器，则该最简电路存在状态转激活。

> 激活转状态和状态转激活。

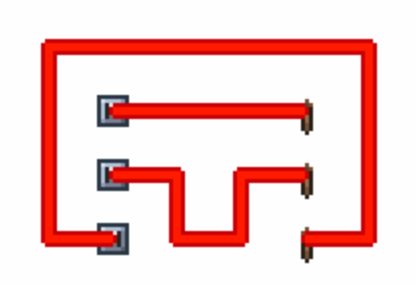


> 我们可以将复杂的电路拆分成最简形式，然后逐条电线的分析。

2.1.3 电线

现实中的数字电路电线会有低电平、高电平、高阻态、未知态等状态，但是泰拉瑞亚中的电线只能传递脉冲信号（激活）一种信号，本身并没有状态或电平。电线只是标记电源和用电器的连接关系（拓扑关系），与颜色、长度和形状无关。电线没有长度和连接电源或用电器数量的限制，连接或不连接电线不会改变电源和用电器的状态。

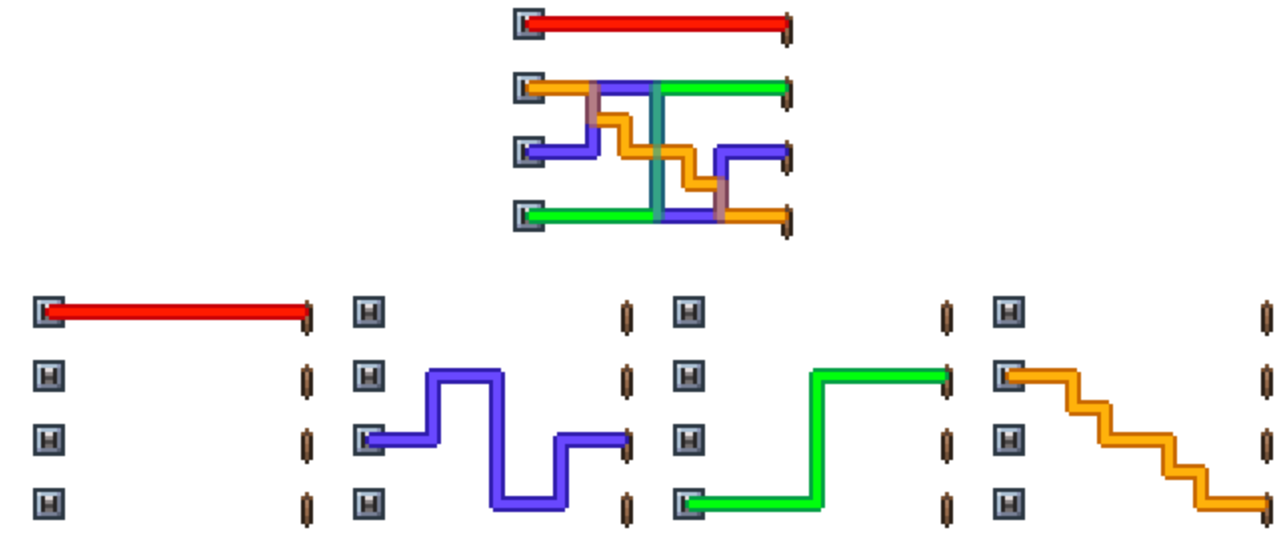
> 电源和用电器有状态与激活两种，但是电线上传递的信号只有激活信号一种。



> 电线只标记电源和用电器的连接关系，与颜色、长度和形状无关。

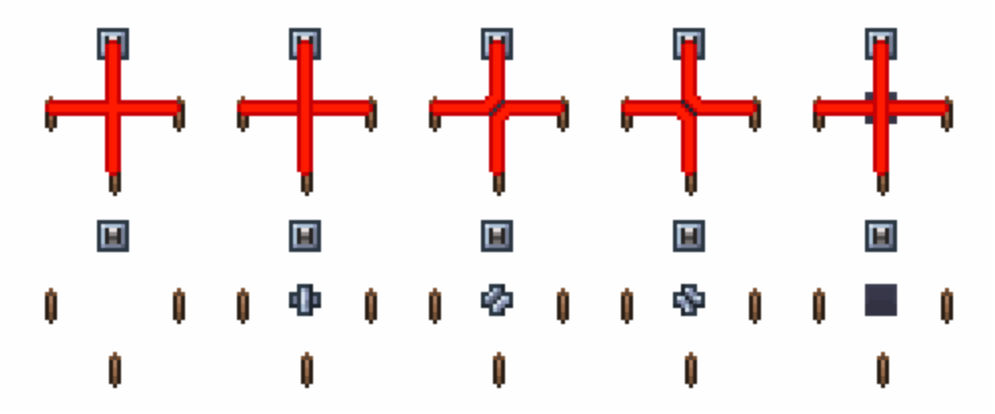
共有四种颜色的电线，相同颜色的电线在相邻图格会被连接到一起，而不同颜色的电线是在不同的图层上，不会被连接到一起。

> 四色电线



> 不同颜色电线在不同图层，当同一格有多色电线时，它们的显示颜色会混合，但彼此间互不影响。

使用分线盒或像素盒可以把一个图格内的相同颜色的电线分开。分线盒同时作用于一个图格中所有颜色的电线。一共有三种分线盒，分别把来自上下和左右、上左和下右、上右和下左的同色电线分开。像素盒只能将来自上下和左右的同色电线分开。



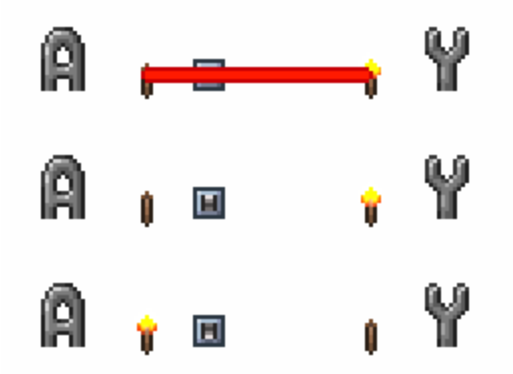
> 开关分别能控制：全部火把、下方火把、左侧火把、右侧火把、下方火把。

参考状态

尽管电线是没有状态的，但是有时为了便于讨论，我们会假设电线有状态，这种状态被称为参考状态。最简状态电路中的电线被称为状态传递电线，简称状态电线；最简激活电路中的电线被称为激活传递电线，简称激活电线。当最简电路中同时存在激活和状态表示时，我们根据更加关注的是状态还是激活来选择电线的类型，通常由用电器的类型决定，忽略电源的类型。后文会用将状态电线的参考状态简称为电线的状态。一般设定电线的初态/稳态为 0；另一态/暂态为 1。

线非

对于最简状态电路，电源和用电器的状态是同步变化的。但是由于电源和用电器的状态是可以人为设定的，如果我们设定一对电源和用电器的状态相反，将电源状态看做输入，将用电器看做输出，我们就会得到一个输入与输出始终相反的电路，也就是非逻辑。



> 设定左侧火把与开关状态相同，则可以用火把代替开关状态。使用电线连接输入（状态电源开关）与输出（状态用电器火把），输入和输出状态始终相反，存在非逻辑。

$$Y = A'$$

像这样不使用逻辑门，只使用一个任意状态电源、一个任意状态用电器和一根电线实现的非逻辑被称为线非。由于线非的存在，我们可以在任意状态用电器出实现取反，而不需要使用逻辑门。

线异或

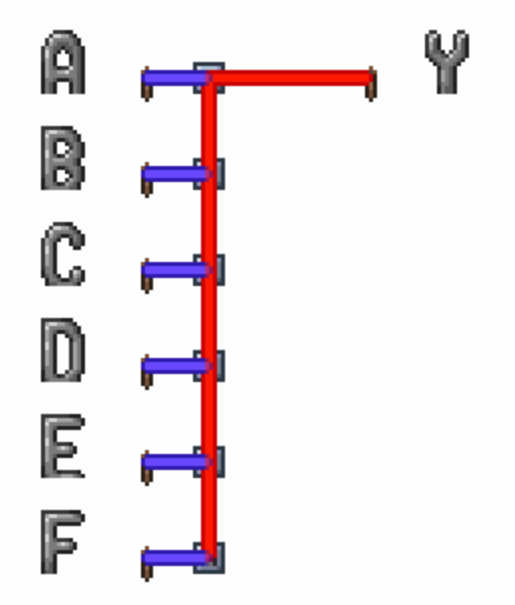
两两异或逻辑又被叫做可控非逻辑，因为当一个输入为 0 时，输出与另一个输入相等；当一个输入为 1 时，输出为另一个输入的反相。考虑最简状态电路中的两个电源和一个用电器，将电源状态看做输入，将用电器看做输出，当一个输入为 0 时，输出与另一个输入相等；当一个输入为 1 时，输出为另一个输入的反相；满足两两异或逻辑。

> 使用电线连接输入（两个状态电源开关）与输出（状态用电器火把）满足两两异或逻辑。

$$Y = A \hat{} B$$

由于两两异或具有交换率，交换两个输入对输出没有影响，所以输出与两个输入的顺序无关，具有普适性。同理，两两异或具有结合律，所以即便有多个输入，输出仍然满足所有输入间两两异或，可以任意

扩展。



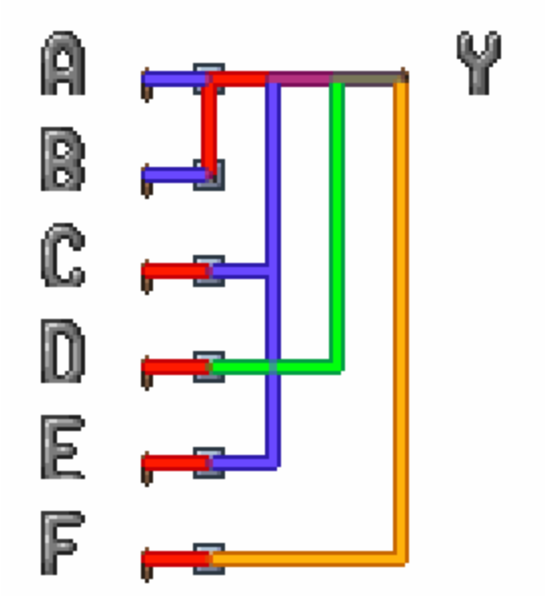
$$Y = A \hat{} B \hat{} C \hat{} D \hat{} E \hat{} F$$

像这样不使用逻辑门，只使用多个状态电源和一个状态用电器实现两两异或逻辑的方式被称为线异或。由于线异或的存在，我们可以在任意状态用电器处实现两两异或，而不需要使用逻辑门。

> 注意区分线异或的两两异或和异或门的多输入异或逻辑在输入数量大于等于 3 时是不同的。

> 线非是线异或只有一个输入的特殊情况。

> 实际上状态用电器的最终状态是自身状态和其上所有电线上所有输入电源两两异或的结果，在用电器处完成，不要求所有输入电源都在同一根电线上。



$$Y = A \hat{} B \hat{} C \hat{} D \hat{} E \hat{} F$$

> 线异或是泰拉瑞亚电路中最常用的运算。

线或

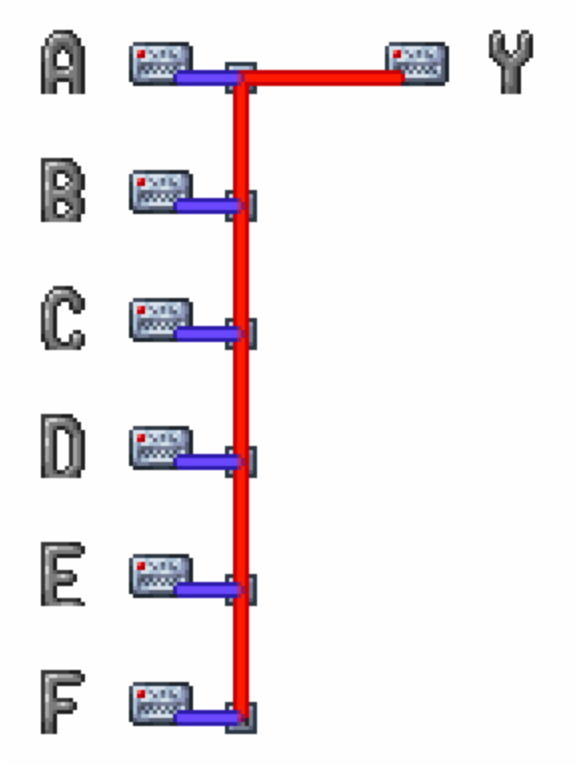
或逻辑中，当任意输入为 1 时，输出为 1；只有全部输入都为 0 时，输出为 0。考虑最简激活电路中的多个电源和一个用电器，任意电源激活时，用电器会被激活；没有电源激活时，用电器不会被激活；满足或逻辑。

> 使用电线连接输入（两个激活电源开关）与输出（激活用电器广播盒）满足或逻辑。

$$Y = A \cdot B$$

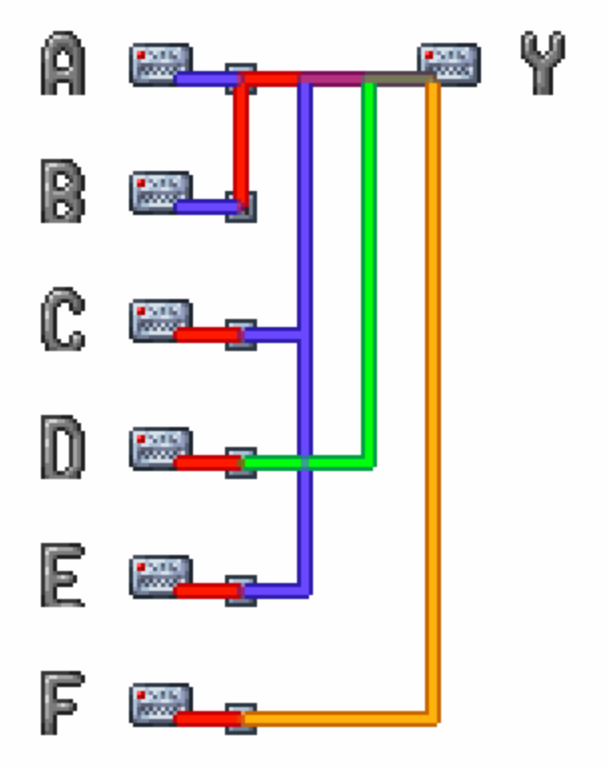
像这样不使用逻辑门，只使用多个激活电源和一个激活用电器实现或逻辑的方式被称为线或。由于线或的存在，我们可以在任意激活用电器处实现或，而不需要使用逻辑门。

> 多输入或与两两或等价，所以线或与多输入或门等价。



$$Y = A \cdot B \cdot C \cdot D \cdot E \cdot F$$

> 实际上激活用电器是否激活，取决于其上所有电线上所有输入电源中是否有激活的电源，在用电器处完成，不要求所有输入电源都在同一根电线上。



$$Y = A \cdot B \cdot C \cdot D \cdot E \cdot F$$

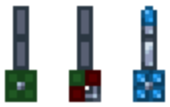
> 然而对于没有冷却时间的物理激活用电器（如广播盒），被激活一次就会响应一次，即便是同时激活多次，每一次都会单独响应，并不是严格的线或逻辑。

> 严格的线或逻辑只存在于逻辑电路（故障灯），通常用于检测多位信号中是否存在或全为 0 / 1，用来代替较大的多输入或门，较线异或应用较少。

2.2 逻辑门

2.2.1 逻辑门灯和逻辑门

泰拉瑞亚中的逻辑门由逻辑门和逻辑门灯组成，逻辑门灯可以被堆叠于逻辑门上，逻辑门的状态取决于其上中间不相隔任何非逻辑灯图格的全部逻辑门灯的状态。



> 逻辑灯堆叠于逻辑门上。

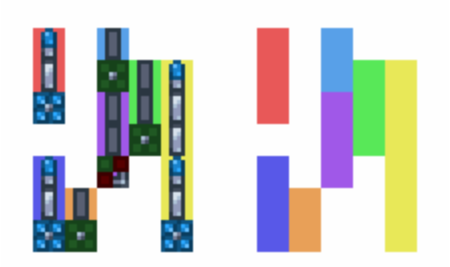
逻辑门灯

逻辑门灯有三种，分别是 **逻辑门灯（亮）、逻辑门灯（灭）、逻辑门灯（故障）** 。其中逻辑门灯（亮）和逻辑门灯（灭）被称为普通逻辑门灯，简称普通灯；逻辑门灯（故障）被称为故障逻辑门灯，简称故障灯。

> 逻辑门灯（亮）、逻辑门灯（灭）、逻辑门灯（故障）。

逻辑灯是用电器，输入是电线，输出是逻辑门。普通灯是状态用电器，有记忆，输入满足线非和线异或逻辑；故障灯是激活用电器，无记忆，输入满足线或逻辑。

逻辑灯必须堆叠在逻辑门上，我们说这些逻辑灯在该逻辑门上，该逻辑门在这些逻辑灯下。



> 这些紧密排列的逻辑门中，相同颜色的区域是同一个逻辑门。

逻辑门

当逻辑门上没有故障灯时，逻辑门被称为普通逻辑门，简称普通门；当逻辑门上有故障灯时，逻辑门变成蓝色，被称为故障逻辑门，简称故障门。

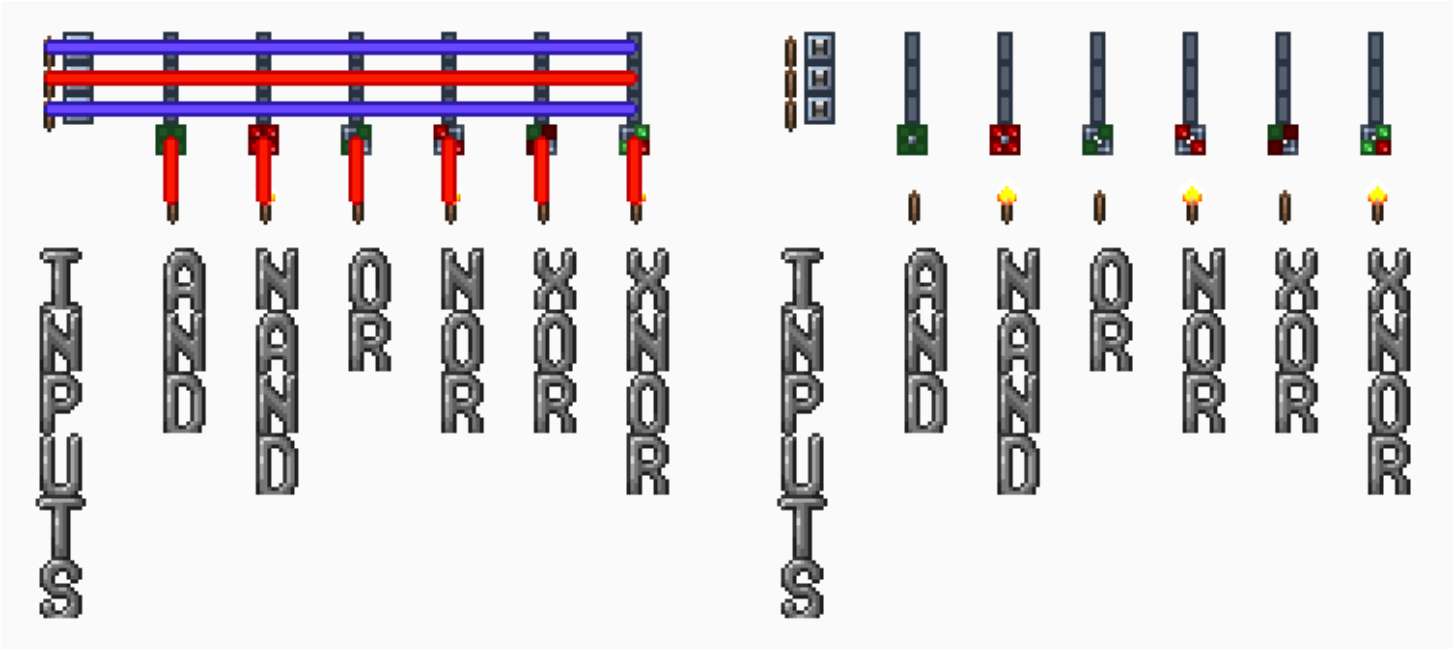
> 普通逻辑门（亮）、普通逻辑门（灭）、普通逻辑门（故障）。

逻辑门是电源，输入是逻辑灯，输出是电线，普通门是状态电源，状态取决于其上普通灯状态；故障门是激活电源，激活取决于其上故障灯激活。一个逻辑灯只对应一个逻辑门，而一个逻辑门可以对应任意多个逻辑灯。

2.2.2 普通逻辑门

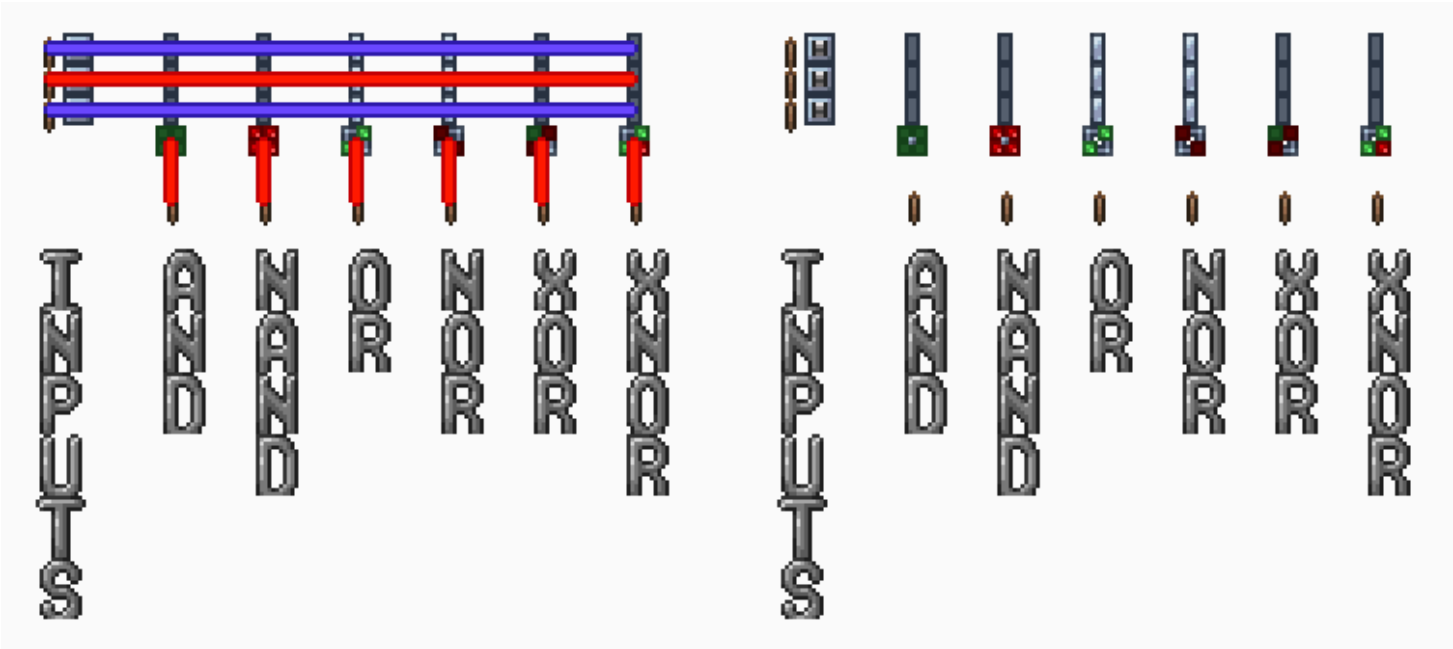
普通门有 **与、与非、或、或非、异或、同或** 六种。普通门有亮和灭两种状态，由其上普通灯状态决定。如果普通灯是输入，普通门是输出，亮灯/门是 1，灭灯/门是 0，输入与输出的关系与上一章所述同名逻辑运算相同。

> 与、与非、或、或非、异或、同或六种普通门。



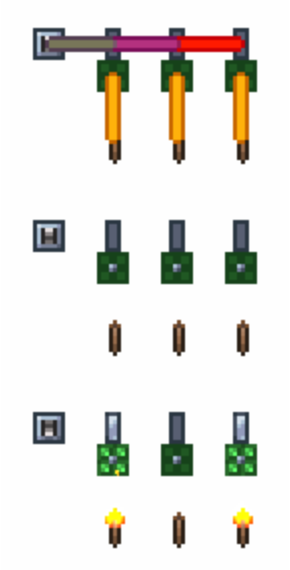
> 六种三输入普通逻辑门。

利用线非与反演律可以将六种普通门等价为与门和异或门两种普通门，与门在其上所有普通灯点亮时点亮，异或门在其上只有一个普通灯点亮时点亮。在任何电路中，都可以只使用这两种普通门实现全部逻辑，而不需要其它普通门。



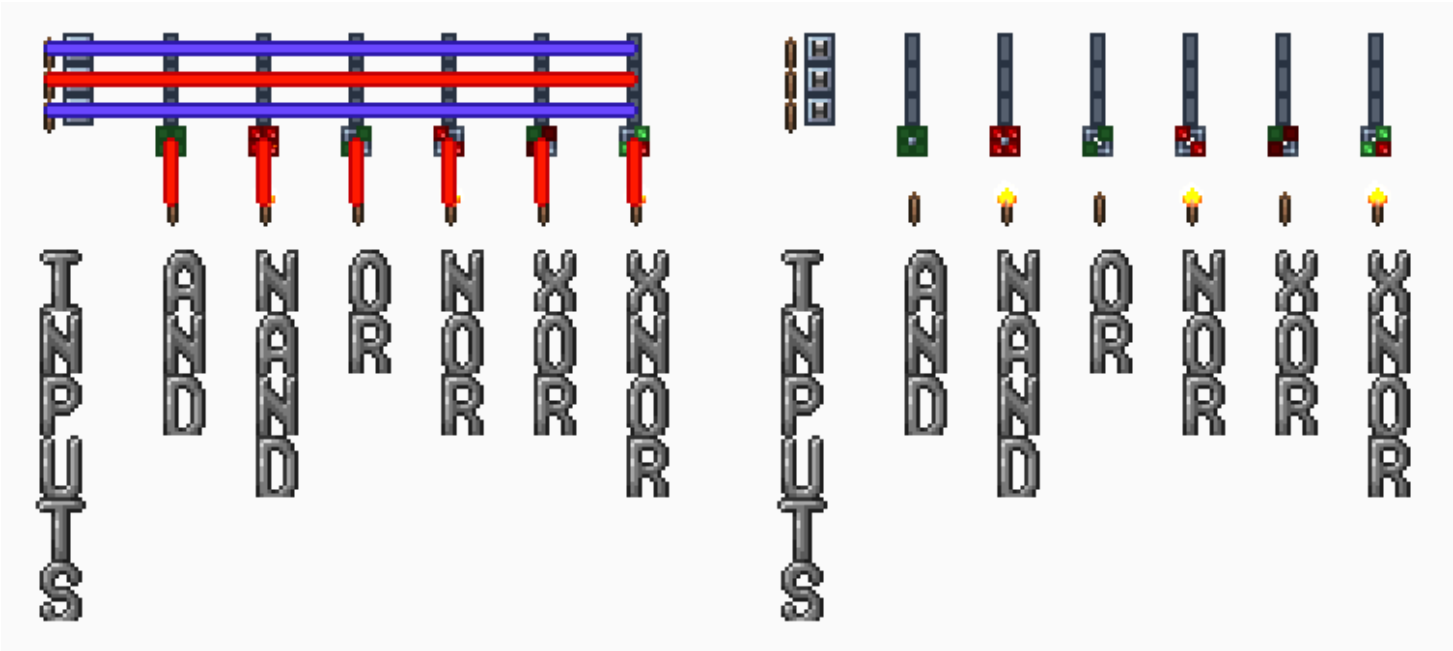
> 对于所有可能的输入，左侧四种逻辑门、右侧两种逻辑门的输出一致，证明其等价。

普通灯是状态用电器，普通灯切换状态时，下方普通门会根据上方所有普通灯状态决定自己状态；而普通灯同时切换多次状态，下方普通门只会考虑普通灯的最后状态（即线异或），而不考虑中间状态。普通门是状态电源，在由亮到灭和由灭到亮切换状态时都会激活其上电线。



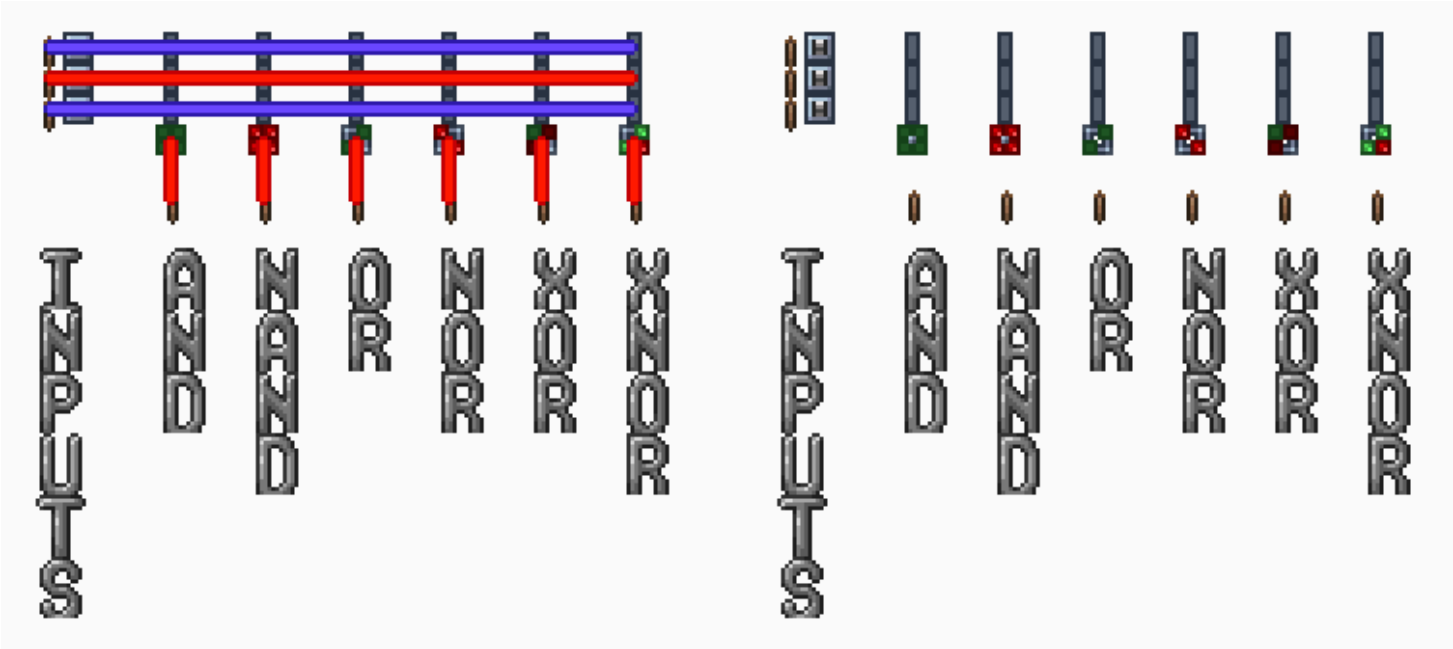
> 左右逻辑灯分别被激活一次和三次，线异或结果是 1，输出 1；中间逻辑灯被激活两次，线异或结果是 0，输出 0。

逻辑门等价的证明



> 等价前的普通门。

由于线非的存在，我们可以直接令输出状态用电器的状态与输入电源普通门的状态相反，令与门和与非门、或门和或非门、异或门和同或门等价。



> 利用线非令与门和与非门、或门和或非门、异或门和同或门等价。

类似乘法分配律这样的运算律，逻辑运算中也存在一些运算率，被称为逻辑运算律。反演律（摩根律）是一个常用的运算律，表述如下：

$$(A \cdot B)' = A' + B'$$

$$(A + B)' = A' \cdot B'$$

> 关于这个运算律的证明可以通过画图或者列真值表完成，对于三项及三项以上仍然是成立的。

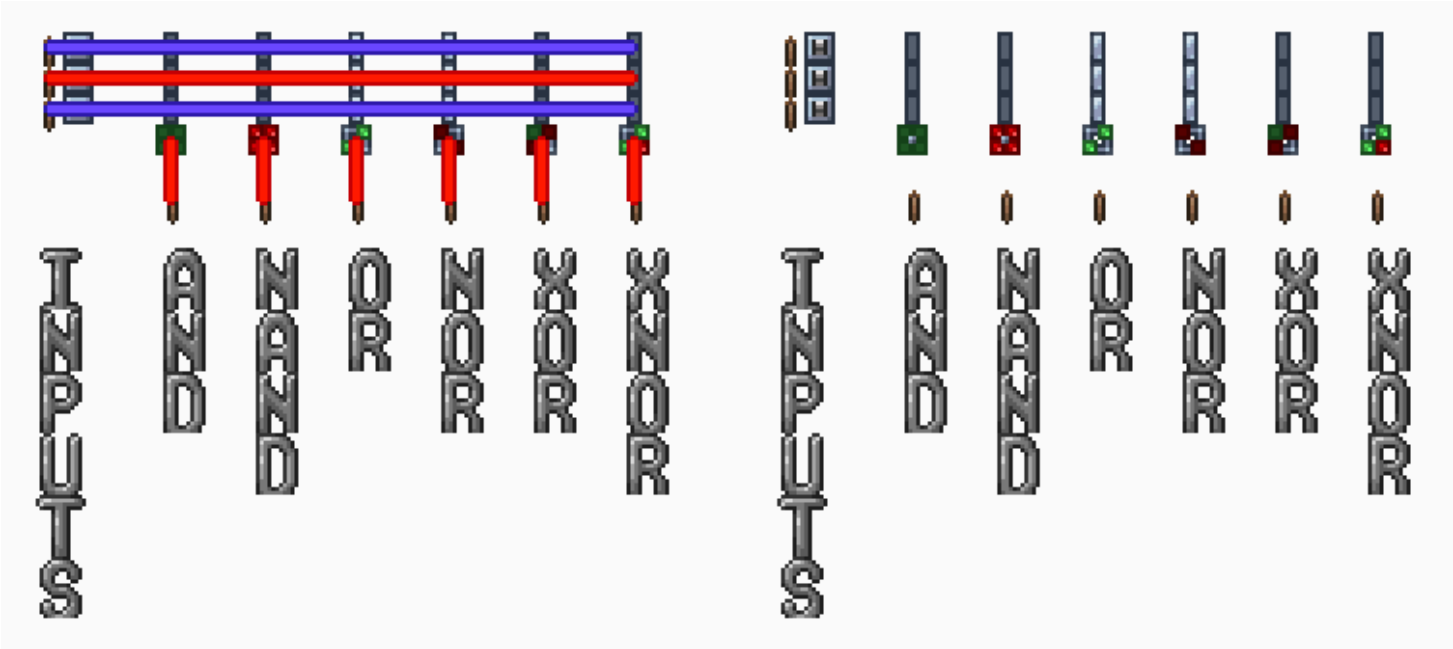
将或门的逻辑表达式进行两次取反（两次取反结果不变），然后使用反演律可将或逻辑转换为与逻辑：

$$Y = A + B = (A + B)'' = (A' \cdot B')'$$

将与门的逻辑表达式进行两次取反（两次取反结果不变），然后使用反演律可将与逻辑转换为或逻辑：

$$Y = A \cdot B = (A \cdot B)'' = (A' + B')'$$

由此我们可以直接令输入状态电源的状态与输出用电器普通灯的状态相反，令与门和或非门、或门和与非门等价，然后再利用前述结论，可令与门和或门等价。



> 利用反演律令与门和或非门、或门和与非门等价，再利用线非令与门和或门、与非门和或非门等价。

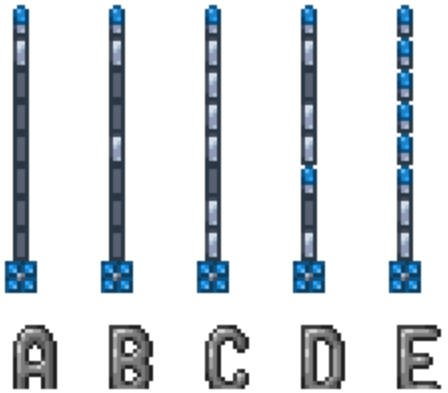
那么我们是否能将与门和异或门等价呢？不可能，原因会在后文介绍。在泰拉电路中，我们只使用与门和异或门就足够了，其他普通门都可以用这两种门代替。

2.2.3 故障逻辑门

当在普通门上堆叠故障灯时，普通门会变成蓝色的故障门，所有故障门逻辑都是一样的，尽管它们看起来不同。

> 故障逻辑门，尽管贴图不同，但是逻辑相同

故障门与其上最靠下的故障灯之间的普通灯称为有效逻辑灯，简称有效灯。当故障逻辑门上有任意一个故障逻辑灯被激活时，故障门会按照有效灯中亮灯的数量比有效灯总数为概率激活。



> 当激活图中五个故障门的故障灯时，故障门激活的概率分别是：
>
> A: $\frac{1}{7}$; B: $\frac{2}{7}$; C: $\frac{6}{7}$; D: $\frac{1}{2}$; E: 1;

- 其他故障灯（激活效果与最下方故障灯相同）
- 其他普通灯（不影响概率）
-
-
-
- 最下方故障灯（下方为有效灯）
- 有效灯（激活故障灯时故障门激活概率为有效亮灯比全部有效灯）
-
- 故障门

> 由于故障门是概率激活的，所以又被叫做随机门。

故障门上的故障灯为激活用电器，而普通灯为状态用电器。改变普通灯的状态不会使故障门激活，而激活故障灯有可能会使故障门激活。一个故障逻辑门上的所有故障灯都是相同的，而除有效灯以外的普通灯的状态不会影响故障逻辑门的概率，除有效灯以外的普通灯可以看做不存在。故障灯每次激活时，故障门都有概率激活；而故障灯同时被激活多次，下方故障门只会尝试判断激活一次（即严格线或）。故障门是激活电源，在成功激活时会激活其上电线。

2.3 逻辑结算

2.3.1 逻辑结算

帧结算

泰拉瑞亚中每一帧中，除电路外所有的游戏机制会依一定顺序结算，电路结算的位置和触发电路的物理电源结算位置有关。也就是说，当实体触发电路时，会在实体所在结算，触发电路的位置，插入一个电路结算，插入的电路结算被称为逻辑结算。

> 比如由某人物触发的压力板，会在该人物结算的压力板结算位置处，触发一个逻辑结算；由某射弹触发的垫板，会在该射弹结算的垫板结算位置处，触发一个逻辑结算。

> 有些电路会受物理电源的类型影响，例如计时器作为物理电源时，不会直接激活自己，但可以用逻辑门绕开；逻辑感应器（玩家）作为物理电源时无法触发传送机，需要更换电源类型；加重压力板作为物理电源时，触发传送机会导致传送否定等等，详细内容会在后文介绍。

逻辑结算

当一个物理电源激活一次时，会新建一个逻辑结算，逻辑电源无法新建逻辑结算。

每帧内逻辑结算的数量没有限制，在逻辑结算结算完成前，后续的代码不会运行。无论逻辑结算的规模有多大，有多少逻辑门和多少电线，都会在一帧内完成。可以说泰拉瑞亚电路是没有频率上限且没有延迟的。

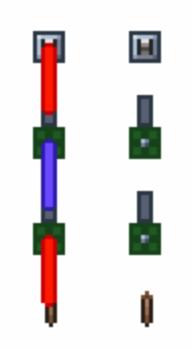
> 实际上频率会受限于能制作出来的驱动/时钟频率；延迟会受限于电脑性能，当计算量较大时，帧结算需要的时间会增加，物理帧会降低。

在泰拉瑞亚中，逻辑结算被叫做时钟周期，简称周期；而接近现实电路周期概念的电路结算最小单位被叫做逻辑帧，后文将使用上述叫法。能在一个周期内完成计算的电路被称为单周期电路，需要多个周期才能完成计算的电路被称为多周期电路。

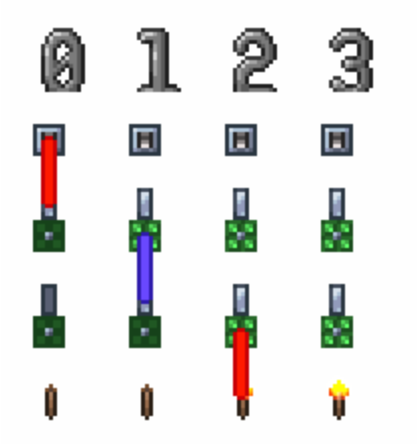
逻辑帧结算

对于逻辑门电路，逻辑结算被分步进行，每一步中都进行 **电源（逻辑门）激活判断** → **电源（逻辑门）激活** → **电线激活** → **用电器（逻辑灯）激活** 四小步，我们将这样的四步统称为逻辑帧。一个逻辑电路的结算过程中，反复运行每个逻辑帧中的四小步，直到没有逻辑门需要激活。

> 考虑这样的电路：



> 激活开关后：



> 第 0 逻辑帧：开关激活 → 电线激活 → 逻辑灯激活；
> 第 1 逻辑帧：逻辑门判断 → 逻辑门激活 → 电线激活 → 逻辑灯激活；
> 第 2 逻辑帧： 逻辑门判断 → 逻辑门激活 → 电线激活 → 火把激活；

在逻辑灯状态改变后，并不会马上进行逻辑灯下逻辑门是否激活的判断，而是在下一逻辑帧再进行激活判断。所以某一逻辑帧逻辑门是否激活只需要考虑逻辑门上逻辑灯在上一逻辑帧的终态，不需要考虑中间态（普通灯考虑亮灭，故障灯考虑被激活）。

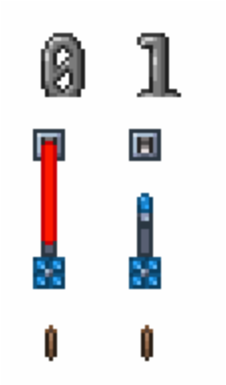
> 考虑这样的电路：

> 激活开关后：

> 第 0 逻辑帧：开关激活 → 电线激活 → 逻辑灯激活：普通灯由灭到亮，故障灯由稳态到激活态；
> 第 1 逻辑帧：逻辑门判断：故障灯激活态，普通灯亮，故障门应激活 → 逻辑门激活 → 电线激活 → 火把激活；

> 考虑这样的电路：

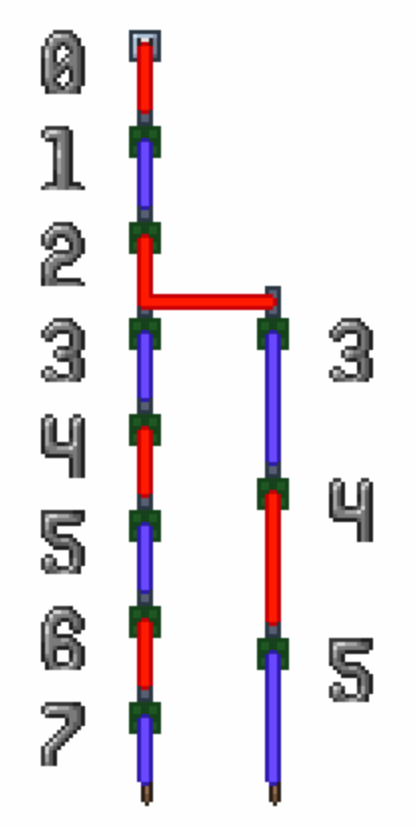
> 激活开关后：



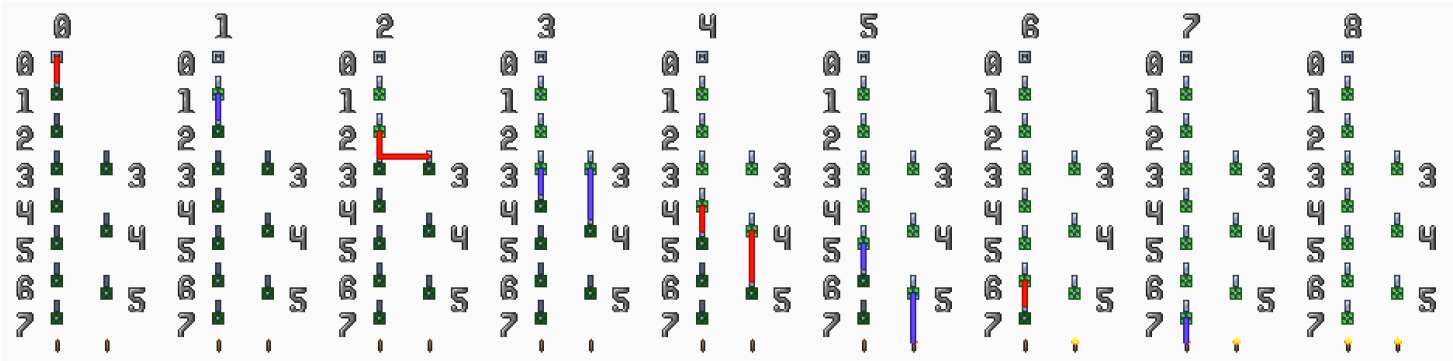
> 第 0 逻辑帧：开关激活 → 电线激活 → 逻辑灯激活：普通灯由亮到灭，故障灯由稳态到激活态；
> 第 1 逻辑帧：逻辑门判断：故障灯激活态，普通灯灭，故障门不应激活；

逻辑帧是逻辑门电路的最小时间单位，逻辑帧内的事情是同时发生的，所以我们会以逻辑帧为单位分析逻辑门电路。两个相邻逻辑帧间相隔一个逻辑门，令物理电源所在逻辑帧为第 0 逻辑帧，每经过一个逻辑门逻辑帧 +1 ，可以通过数出一个信号从物理电源传递到某个逻辑门时经过的逻辑门数量，来计算该逻辑门在此逻辑结算所在逻辑帧位置。当需要分析一个信号到达两个逻辑门时相差多少逻辑帧时，我们可以上溯到它们最近的公共逻辑帧。在逻辑电路中的延迟指的是逻辑帧延迟，表示信号的先后逻辑帧差，同逻辑结算中没有时间差。

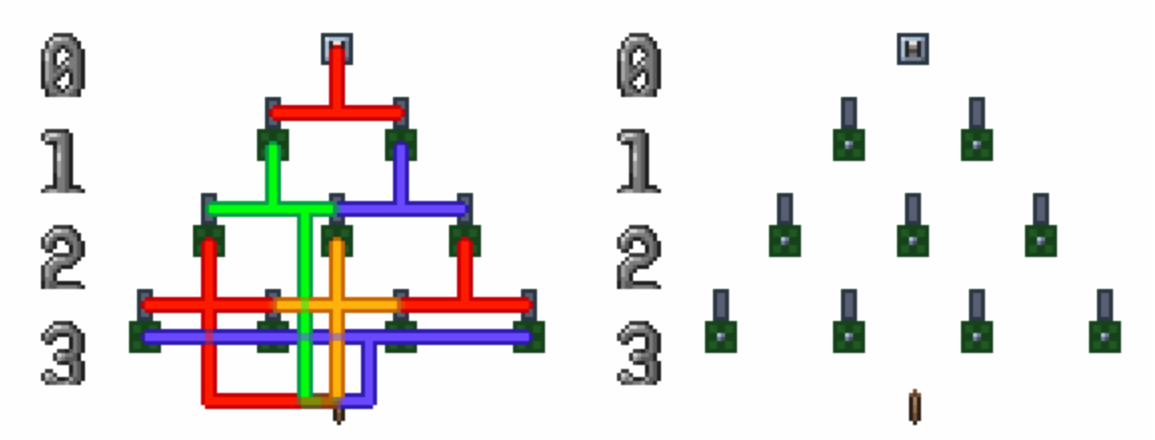
> 考虑这样的电路：



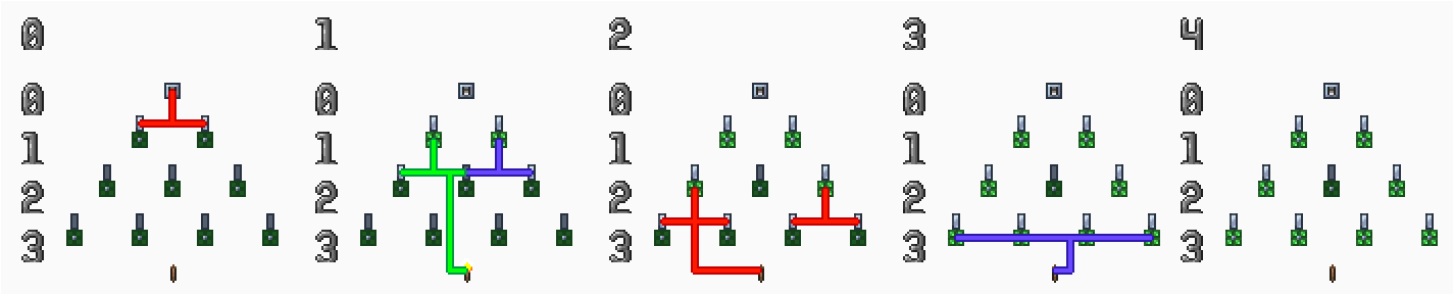
激活开关后：



考虑这样的电路：



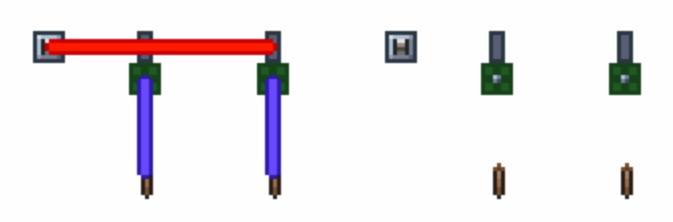
激活开关后：



电源结算

每个逻辑帧中，电线会先后激活一些逻辑灯，而这些逻辑灯下的逻辑门可能会在下一逻辑帧被激活。逻辑门激活的顺序取决于其上逻辑灯在上一逻辑帧被激活的顺序，先激活的逻辑灯下的逻辑门先激活，后激活的逻辑灯下的逻辑门后激活。

> 考虑这样的电路：



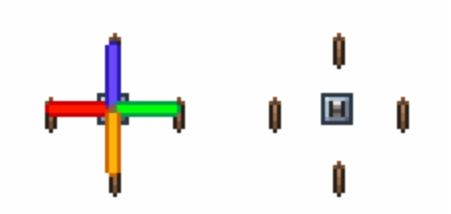
> 激活开关后：

> 第 0 逻辑帧：左边的逻辑灯先被激活，右边的逻辑灯后被激活；
> 第 1 逻辑帧：左边的逻辑门先激活，右边的逻辑门后激活。

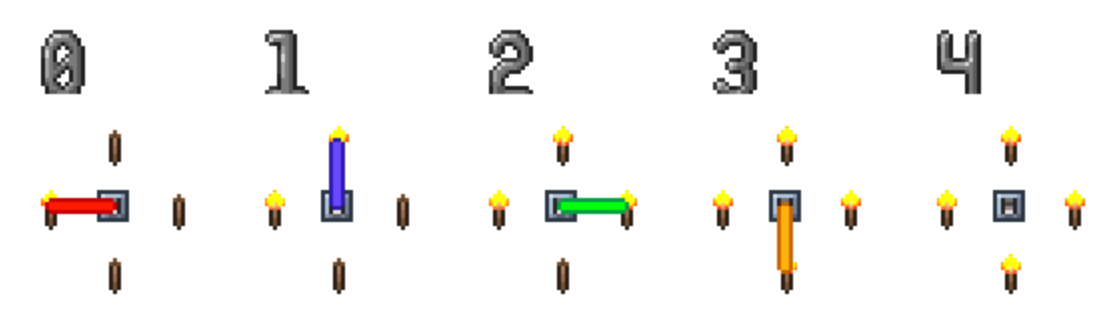
电线结算

每个电源上的每个格子，会按照 **先从左到右，再从上到下** 激活每个格子里的电线。每个格子的四色电线中，会按照 **红** → **蓝** → **绿** → **黄** 的颜色顺序依次进行结算。

> 考虑这样的电路：

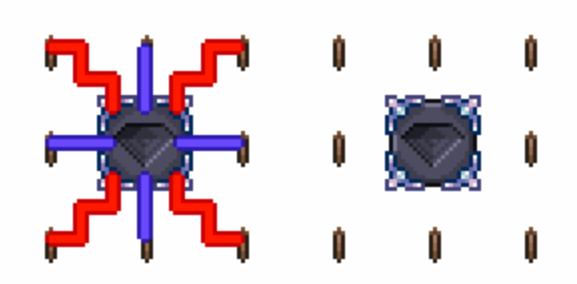


> 激活开关后：



> 电源上每一格的电线会按照 红 → 蓝 → 绿 → 黄 的顺序结算。

> 考虑这样的电路：



> 激活开关后：

> 电源的每一格会按照先从左到右，再从上到下的顺序结算其上电线。

格子结算

每根电线进行结算时，会从电源开始，使用四连通广度优先搜索（BFS）来遍历该电线上连接的所有图格，添加到队列的顺序是 **下** → **上** → **右** → **左** 。

一般情况下，我们只需要知道，电线上各个格子按照沿电线上距离电源的距离由近到远的顺序激活。

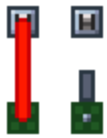
> 考虑这样的电路：

> 激活开关后：

> 电线的每一格会按照从电源出发，沿着电线逐格结算。

2.3.2 爆门

> 泰拉电路没有延迟，逻辑结算不完成程序不会继续执行，那么激活这样的电路会发生什么？



> 开关激活 → 电线激活 → 逻辑灯激活 → 逻辑门激活 → 电线激活 → 逻辑灯激活 → 逻辑门激活 → ……

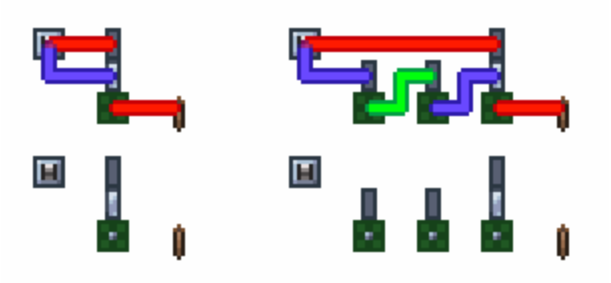
> 这会陷入死循环，卡死游戏吗？然而实际尝试后我们会发现逻辑门冒烟了，但是游戏并没有卡死，这是怎么回事呢？

泰拉为了防止无延迟的电路出现死循环卡死游戏，为逻辑门添加了一个设定：同一个逻辑门在一次逻辑结算中只能激活一次。如果尝试激活第二次，逻辑门的状态会正常切换，但是不会激活其上电线，并且会有白烟从逻辑门中冒出，这种现象被称为 **爆门**。

> 上方的示例电路中红线会激活两次，第一次电源是开关，第二次电源是逻辑门，而逻辑门只会激活一次。

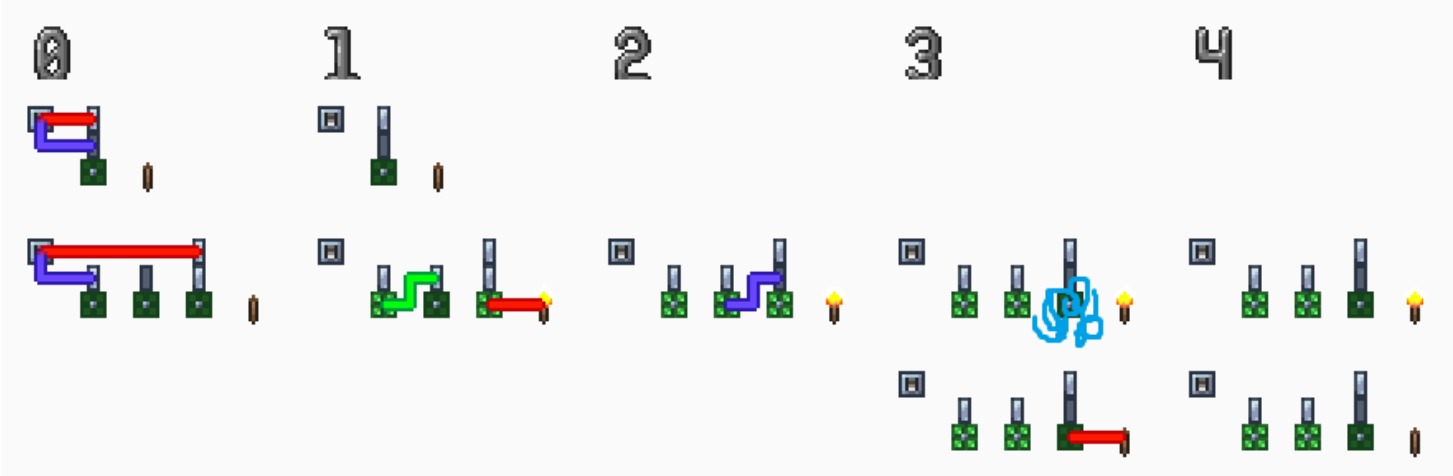
实际上，使用爆门来防止死循环，也会产生一些副作用，有些不会死循环的电路也会发生爆门。在实际应用中，爆门经常导致电路出问题，这是因为我们往往希望逻辑门在不同逻辑帧多次状态切换时可以多次激活，但实际上不会。学会了预测爆门，就可以想办法避免爆门，甚至利用爆门。

> 考虑这样的电路：



> 左右逻辑一样，区别是左侧两个信号没有延迟同时到达；右侧的下方信号会延迟两个逻辑帧到达。

> 激活开关后：



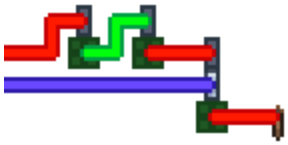
> 第一行是左侧电路激活后情况，火把不会点亮；
> 第二行是右侧电路激活后情况，上方信号先激活逻辑门点亮火把，下方信号第二次激活逻辑门失败，触发爆门，逻辑门没有发出信号；逻辑门状态改变但火把状态未改变，逻辑门和火把状态不一致；
> 第三行是假设没有爆门时右侧电路激活后情况，上方信号先激活逻辑门点亮火把，下方信号第二次激活逻辑门熄灭火把，与左侧电路结果一致。

竞争和冒险

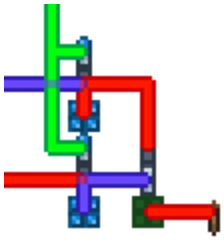
输入信号通过两条或以上路径到达同一个逻辑门时，由于两条路径上的延迟（逻辑帧）不同，出现了到达先后时间差，被称为竞争；由于竞争的存在，先到达的信号激活逻辑门后，后到达的信号再想激活逻辑门会导致逻辑门爆门，出现错误，被称为冒险。

我们通常使用两个方法来避免爆门：

- 给先到达的信号添加逻辑延迟，让两个信号同时到达同一个逻辑门；
- 使用寄存器暂存信号，需要时再将它们同时发送给同一个逻辑门；



> 给先到达的信号（上方信号）添加逻辑延迟，让两个信号同时到达逻辑门。



> 使用寄存器暂存信号，需要时再将它们同时发送给逻辑门。