数字电路学习笔记（七）：经典组合逻辑器件（上）

JoshCena

一、集成电路

在引入“集成电路”后，电路设计实现了从原子到分子的质变。集成电路，最重要的当然是“集成”二字了……总体来说，我们希望设计出的集成电路具有：

- 可复用
- 可扩展
- 高封装度
- 健壮（泛指其可靠性、安全性、一致性等）

等特点。从定义上说，每一个在此之前提到的电路都可以被做成一个集成电路；但这些只能叫做专用集成电路，被用在特定的设备中，不具有可复用性，所以不在本文讨论范围中。同时，由于文章目前进展到组合逻辑电路，因此涉及的几项电路也都是以组合逻辑为基础的，不会牵涉到时序逻辑等内容。

与之前设计的朴素的电路不同，现实中的组合逻辑器件为了便于功能扩展，往往还有一些功能端口，可以控制是否输出，检测是否有输入等，实现与其他逻辑电路的配合。

为探究几种常见的组合逻辑电路，我们从一个实际的例子入手：**计算器**。简单一点，我们的计算器先实现加法和减法。（不保证最终的成果和现实中的设计一致）这也是我在前言中展示的 Minecraft 电路所做的。大概的思路如下：

这个东西只能做一位数的加法，但将它稍微扩展，就可以计算更多位数。我们还要忽略一些小细节，比如涉及存储的部分，以及二进制计算和十进制的转换（详见第二章关于“余三码”的介绍）。

二、编码器

编码器能够将单一的输入信号编码。比如，计算机键盘按下一个键，就会被编码成一个八位的 ASCII 码。

在计算器项目中，首先需要的，是把单一的“按钮输入”事件映射成一个二进制数。按钮板共有十个键，所以有十条输出，每条对应一个按钮。通过编码，可以将其变成易于逻辑理解的输入。由于目标是做计算，映射成二进制码（其实是 BCD 码）当然最为合理。（同样参见第二章：数制与编码）

我们假设每次只输入一个信号，不会两路同时产生输入。那么真值表如下：

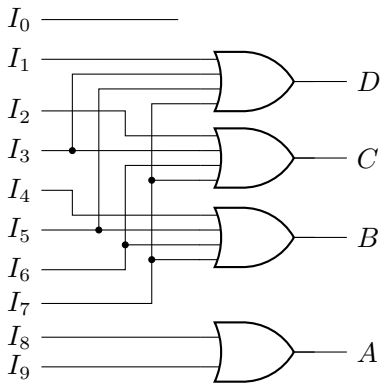
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	A	B	C	D
1										0	0	0	0
	1									0	0	0	1
		1								0	0	1	0
			1							0	0	1	1
				1						0	1	0	0
					1					0	1	0	1
						1				0	1	1	0
							1			0	1	1	1
								1		1	0	0	0
									1	1	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0

未标注的表示为 0

$ABCD$ 表示一个四位二进制数，从高位到低位。任何时候，列出真值表辅助分析都是个好习惯；但在这里，用真值表列式显然不现实。所以，直接分析：例如 B，在 I_4, I_5, I_6, I_7 时为 1，所以 $B = I_4 + I_5 + I_6 + I_7$（实际上，我们把其他项都当作了无关项，这会有一些隐藏的问题，后面分析）。同理：

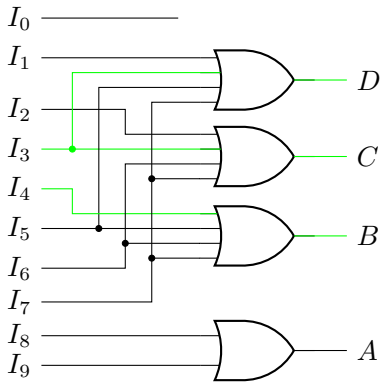
$$
\begin{aligned}
A &= I_8 + I_9 \\
B &= I_4 + I_5 + I_6 + I_7 \\
C &= I_2 + I_3 + I_6 + I_7 \\
D &= I_1 + I_3 + I_5 + I_7
\end{aligned}
$$

这样，我们的第一个编码器就做成了：



这种简单的设计在绝大多数情况下就够用了，但有两个问题：

我们假设每个时刻只有一路输入，但这是不靠谱的。比如，如果同时按下 3 和 4，那么实际输出的是 0111；我们不知道当输出是 0000 时，究竟表示的是按下了 0，还是什么输入都没有。无论如何，让“0”输入直接吊着而不接进电路显然不好。



输入 3，4 的样子；要注意这不是 3+4，而只是无意义的输出，比如 3 和 5 的输出也是 0111

问题 1 的解决方法是优先编码：如果同时按下两个键，则只编码比较大的数，而忽视较小的数，相当于把真值表变成：

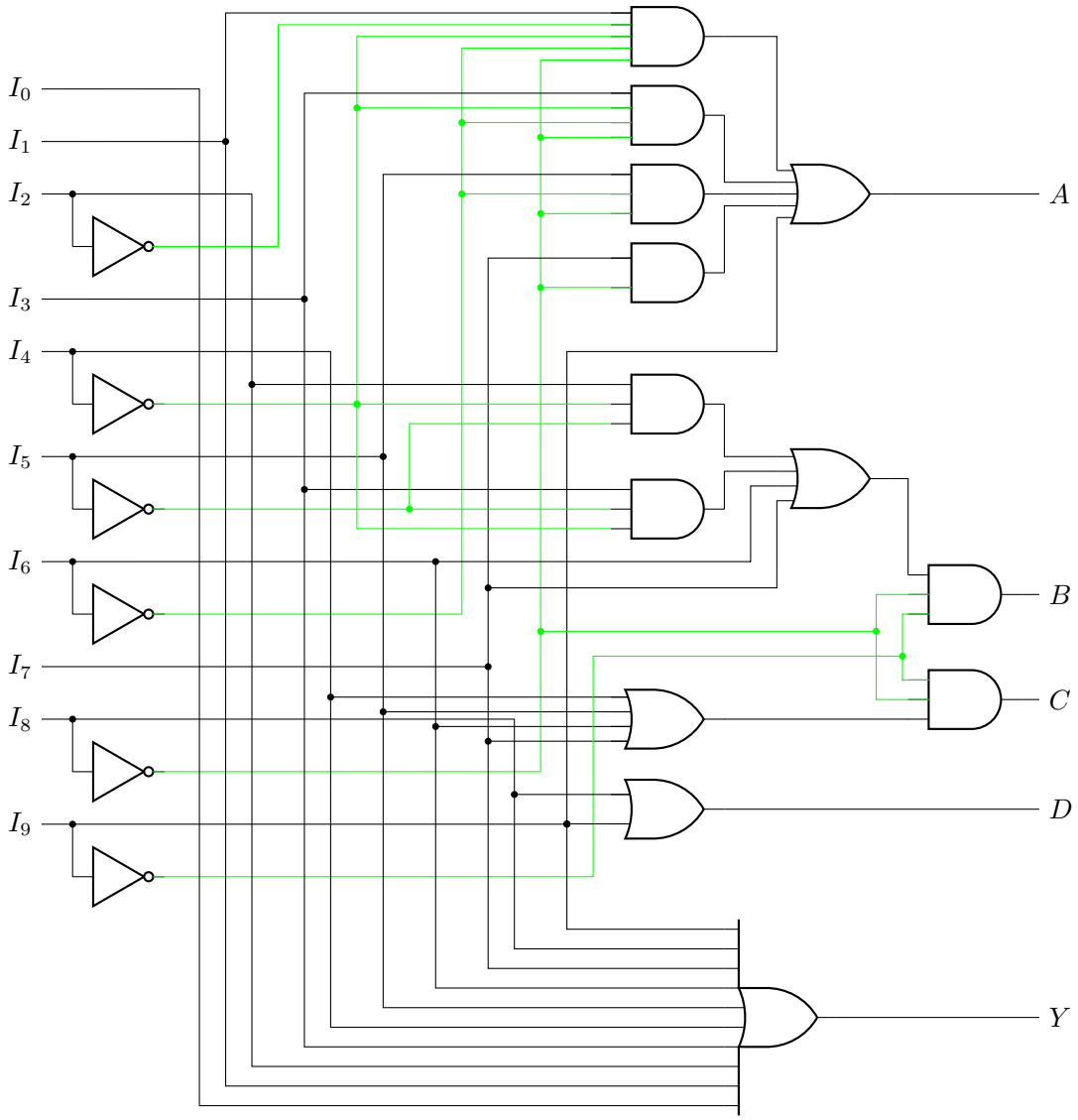
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	A	B	C	D
1										0	0	0	0
×	1									0	0	0	1
×	×	1								0	0	1	0
×	×	×	1							0	0	1	1
×	×	×	×	1						0	1	0	0
×	×	×	×	×	1					0	1	0	1
×	×	×	×	×	×	1				0	1	1	0
×	×	×	×	×	×	×	1			0	1	1	1
×	×	×	×	×	×	×	×	1		1	0	0	0
×	×	×	×	×	×	×	×	×	1	1	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0

未标注的表示为 0

这样，如果按下 3 和 4，则输出 0100，3 则被当成了无关项，不影响输出。虽然这使得电路变复杂了，但提升了健壮性，符合设计方针，是值得的。

问题 2 的解决方法自然是把所有位加在一起，单独作为一路输出——只要这路有输出，就代表有输入。

略过计算过程（可以作为练习），我们最终得到的，就是一个类似这样的电路：



它是两个集成电路的奇怪混搭。首先是 **74HC147**，它是 10 线-4 线编码器，将 10 条线路输入编码成 4 位的二进制数，类似本电路实现的功能；然后是 **74HC148**，是 8 线-3 线编码器，所以只能编码 0...7 的数字，但有不少扩展端口，所以比起 74HC147，通用性更高，可以几个连在一起实现更多位数的编码。它有检测是否有输入的功能。

这样，我们完成了计算器的第一部分：把按钮输入对应成一个能够计算的二进制数。

三、译码器

与编码器相对，译码器把一个二进制码译回单一的输出。在本项目的最后，也需要将计算出的数字译回 0...9，方便由屏幕的驱动器再做计算。真值表如下：

A	B	C	D	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9
0	0	0	0	1									
0	0	0	1		1								
0	0	1	0			1							
0	0	1	1				1						
0	1	0	0					1					
0	1	0	1						1				
0	1	1	0							1			
0	1	1	1								1		
1	0	0	0									1	
1	0	0	1										1

$O_{0..9}$ 的推导没什么意义，不列出了。几个例子：

$$\begin{cases} O_0 = A'B'C'D' \\ O_1 = A'B'C'D \\ O_2 = A'B'CD' \end{cases}$$

所以，如果输入的是 0111，输出就是 O_7。

常用的译码集成电路是 **74HC138**，翻译的是 0...7 之间的数字。它有一个很有意思的性质，就是八个输出正好对应八个最小项（回顾笔记（五）：逻辑设计基础），可以用来拼接逻辑函数，比如 $A'BC' + ABC$，就直接将 O_2 与 O_7 用或门接在一起即可。但在我们的体系中没有什么用处，因为我们只强调设计电路而不强调利用已有的电路。

四、数据分配器

注意到我们的输入不仅是一个数字，还包括“现在正在输入第 _____ 个数字”这个开关。（这是我觉得我这个设计中不合理的地方，在后期有“加减乘除”多个运算后，这个开关会用按下对应的运算按钮代替。）这个开关，决定了究竟是由第一路还是第二路编码/存储模块处理输入的信号。这里，就运用了数据分配器。

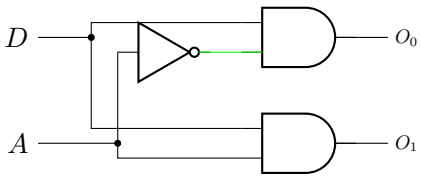
数据分配器有两组输入：第一组只有一路，是待分配的数据；第二组是一个数（“地址码”），表示将这位数据分配到哪一路上。在我们的设计中，只需要 1 分 2，输入一位地址码即可，0 表示“上路”，1 表示“下路”。

如果输入数据 D 和地址码 A，输出 O_0 或 O_1，那么真值表是：

D	A	O_0	O_1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

所以

$$\begin{cases} O_0 = DA' \\ O_1 = DA \end{cases}$$



Simple as that.

会发现，没有专门做“数据分配器”的集成电路，这是因为所有的数据分配器都可以用带控制端口的译码器实现——比如前文提到的 **74HC138**。它的控制端口如果有输入信号，则输出被锁定在高电平。

思考，如果将 D 连在控制端口，而三位地址码 $A_2A_1A_0$ 连在输入端口，那么，当 $A_2A_1A_0 = 011$ 时，D 取 0 或 1，在 O_5 端口分别有什么输出？O_0 呢？

五、数据选择器

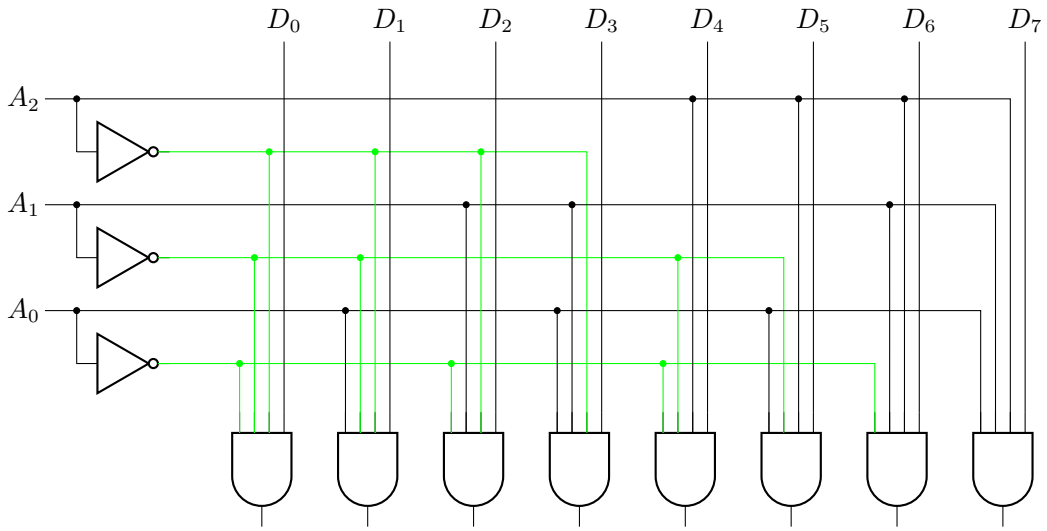
虽然在本项目中没有用到数据选择器，但既然都有了分配器，它就没有道理不出场。选择器和分配器相对，负责按照输入的地址码，从几路输入中选择一路作为输出。

以 8 选 1 选择器为例。它需要八位数据输入和三位地址码，输出一位。顺便介绍一种简化真值表的方法：

A_2	A_1	A_0	O
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

略作分析：O 与哪一个 D 有关，取决于 $A_2A_1A_0$ 的值；比如，$A_2A_1A_0 = 000$ 时，O 只和 D_0 有关。因此，

$$O = D_0A_2'A_1'A_0' + D_1A_2'A_1'A_0 + D_2A_2'A_1A_0' + D_3A_2'A_1A_0 + \cdots + D_7A_2A_1A_0$$



这便是 **74HC151** 的结构。另一个常用的选择器是双 4 选 1 选择器 **74HC153**。

总结：本文介绍了四类逻辑器件：

- 编码器
- 译码器
- 数据分配器
- 数据选择器

同时介绍了五个常用集成电路：

- 74HC147（10 线-4 线编码器）
- 74HC148（8 线-3 线编码器）
- 74HC138（3 线-8 线译码器）
- 74HC151（8 选 1 选择器）
- 74HC153（双 4 选 1 选择器）

在这些器件的结构推导中，有意忽略了一些额外的控制端，以不影响行文逻辑的连贯性。这些端口可以很轻松地集成进来，而不影响主体的工作。

与计算相关的器件，留至下期。