第4章
组合逻辑电路

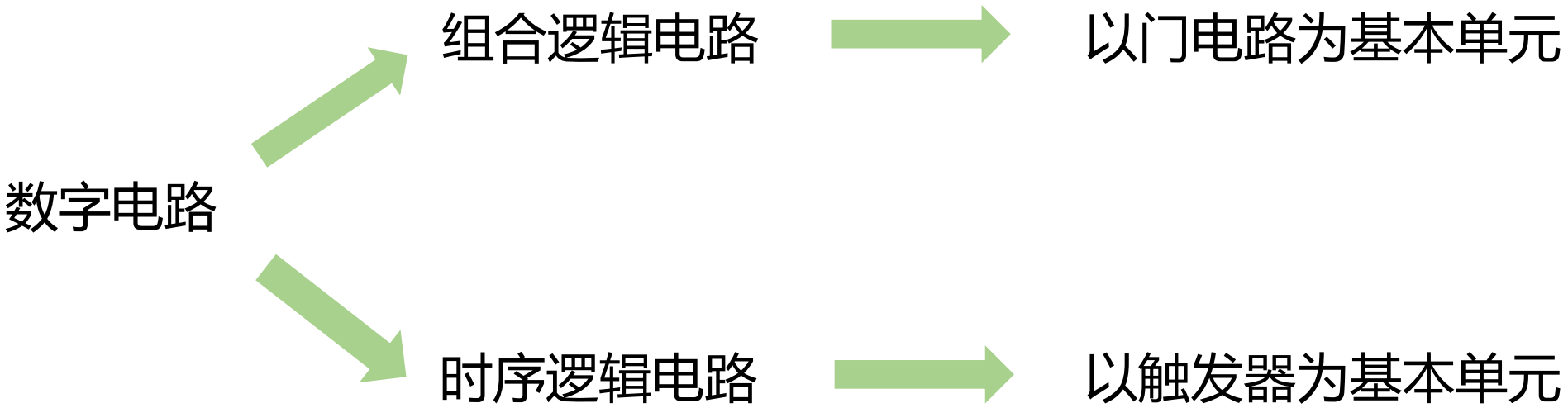
Leng Ximo

组合逻辑电路

- 组合逻辑电路的特点
- 组合逻辑电路的分析与设计方法
- 常用的组合逻辑电路模块（中规模集成器件）
  - 编码器
  - 译码器（数据分配器）
  - 数据选择器
  - 加法器
  - 数值比较器
- 组合逻辑电路中的竞争—冒险

组合逻辑电路

组合逻辑电路概述



组合逻辑电路

组合逻辑电路的特点

功能上，任意时刻的输出仅仅取决于该时刻的输入，与电路原来的状态无关；
结构上，组合逻辑电路不含有存储单元（记忆单元），只有从输入到输出的通路，没有反馈环路；

组合逻辑电路的框图

$$y_1 = f_1(a_1 a_2 \cdots a_n)$$
$$y_2 = f_2(a_1 a_2 \cdots a_n)$$
$$\vdots$$
$$y_m = f_m(a_1 a_2 \cdots a_n)$$

组合逻辑电路的分析

不需要去背！
结合具体题目来感受和理解！

组合逻辑电路的分析方法

给定一个组合逻辑电路 → 分析其电路逻辑功能

根据电路图，从输入到输出逐级分析，得到输出与输入的逻辑函数式；

对得到的逻辑函数式进行化简、形式变换或列出真值表，最后分析其逻辑功能；

组合逻辑电路的设计

不需要去背！
结合具体题目来感受和理解！

组合逻辑电路的设计方法

Step 1：逻辑抽象；

- 分析因果关系，确定输入/输出变量；
- 定义逻辑状态的含意（赋值）；
- 列出真值表；

Step 2：将真值表转换为逻辑函数式；

Step 3：选定器件类型；

（是否有器件类型限制，例如“只能用与非门”这类条件；

是否有指定使用的中规模器件，例如“使用数据选择器”这类条件；）

Step 4：根据所选器件，对逻辑式进行化简或形式变换；（或进行相应的描述（PLD））

Step 5：画出逻辑电路连接图；（或下载到PLD）

Step 6：工艺设计

组合逻辑电路的设计

组合逻辑电路的设计实例

例：设计一个三人表决电路：每人一个按键，如果同意则按下，不同意则不按；
结果用指示灯表示，多数同意时指示灯亮，否则不亮；要求用与非门实现；

Step 1 —— 逻辑抽象：

将每个人的表决情况（按键情况）作为输入变量 A、B、C，同意（按下按键）则为 1， 不同意（不按按键）则为 0；将最终表决结果（指示灯的状态）作为输出变量 Y，表决通过（指示灯亮）则为 1，表决不通过（指示灯灭）则为 0；
则根据题意，列出真值表如下页所示；

组合逻辑电路的设计

组合逻辑电路的设计实例

Step 2 —— 将真值表转化为逻辑函数式：

根据第二章的基础知识，将逻辑函数的真值表转化为对应的最小项之和的形式；

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$Y = A'BC + AB'C + ABC' + ABC$$

组合逻辑电路的设计

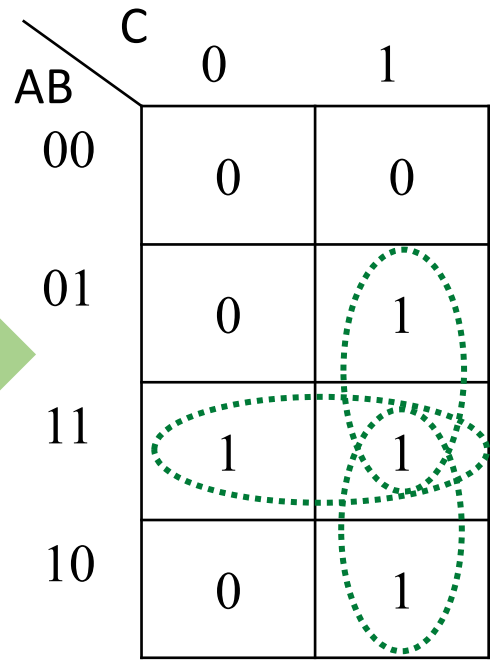
组合逻辑电路的设计实例

Step 3 —— 选定器件类型：根据题意，只能用与非门实现；

Step 4 —— 将逻辑函数式进行化简和特定形式的变换：

由于规定只能用与非门实现，因此需要将逻辑函数式写成最简“与非—与非”式；

$Y = A'BC + AB'C + ABC' + ABC$



$Y = AB + AC + BC$

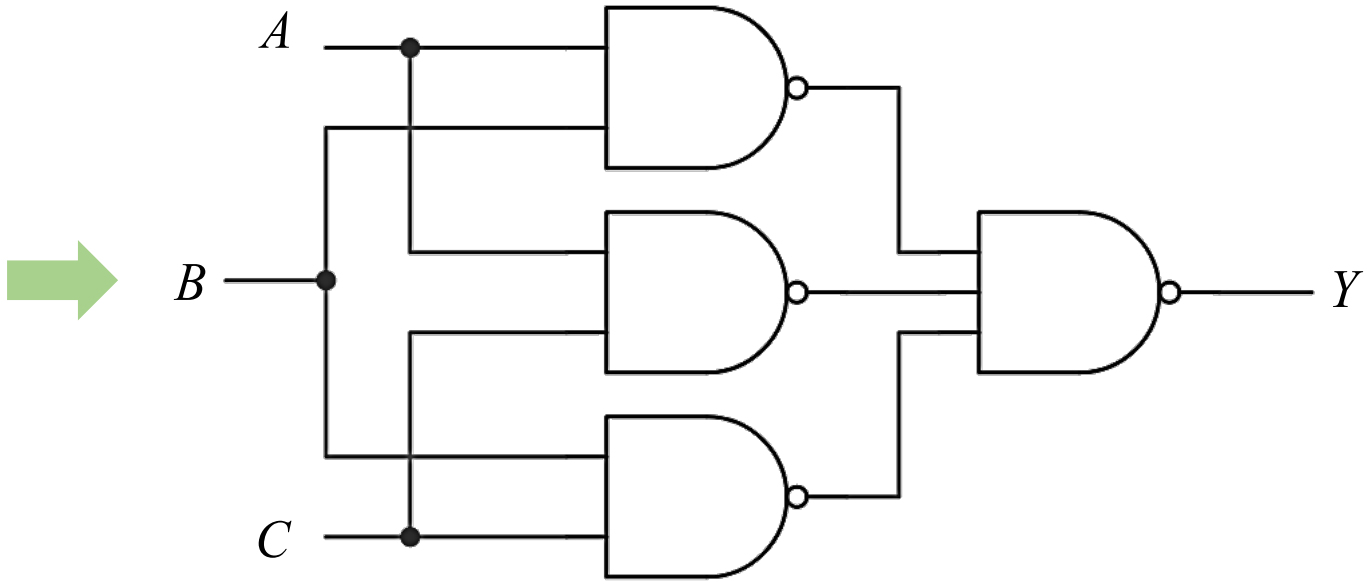
$Y = ((AB)'(AC)'(BC)')'$

组合逻辑电路的设计

组合逻辑电路的设计实例

Step 5 —— 画出逻辑电路连接图；

$$Y = ((AB)'(AC)'(BC)')'$$

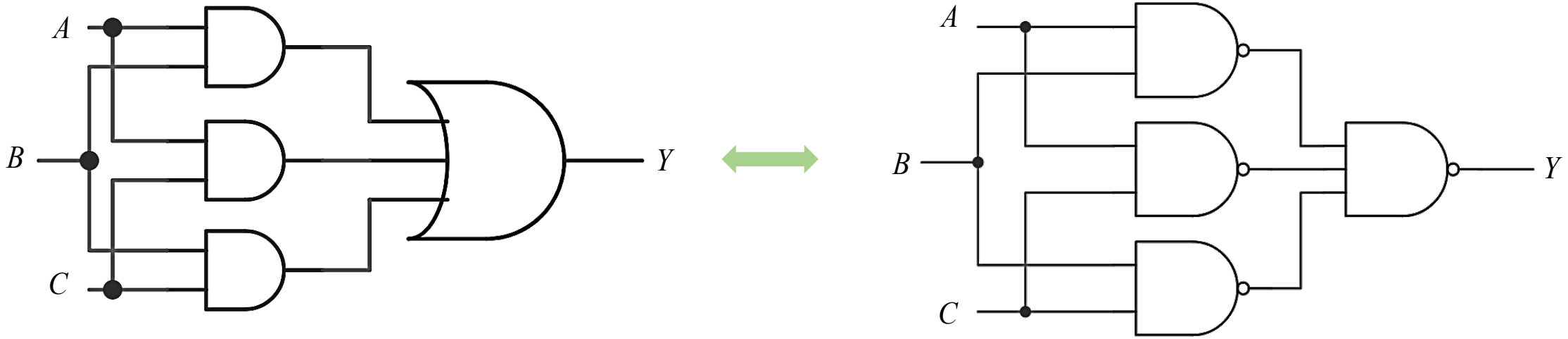


组合逻辑电路的设计

这种方法不具有通用性，
不是必须掌握的！

拓展 —— 一个小技巧

Step 4 和 Step 5 可以在按照常规方法作出最简与或对应的逻辑图后，直接在逻辑函数图上进行变换；只需要把或门的输入端和输出端各加一个取反符号即小圆圈，把或门的符号变成非门，其实就是德·摩根定理在逻辑图上的体现；把与门变成或非门同理；
输入端的反号可以平移至前一级的输出端；



组合逻辑电路的设计

变式 1：

如果每个人可以选择弃权，即每个人的表决意见可以是“同意”、“中立（弃权）”、“不同意”，只有“同意”的数量大于“不同意”的数量表决才通过、指示灯亮，否则表决不通过、指示灯灭；

思考：

在进行第一步逻辑抽象时，输入需要多少个变量？

对于每一个人来说，它的表决意见有三种状态，而想表示三种状态至少需要两位二进制编码，因此每一个人需要两个变量来表示其表决意见的状态，共需要六个输入变量；

例如：规定 11 为 同意，01 为 中立，00 为不同意；
则用 AB 表示第一个人的意见，CD 表示第二个人的意见，EF 表示第三个人的意见；

组合逻辑电路的设计

思考：
这个真值表全吗？
缺少的项的含义是什么？

变式 1：

根据概率论的基础知识，三个独立的人，每个人三种选择，共有 3×3×3 = 27 种结果；

A	B	C	D	E	F	Y	A	B	C	D	E	F	Y	A	B	C	D	E	F	Y
0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0
0	0	0	0	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1	1
0	0	0	1	0	0	0	0	1	0	1	0	0	0	1	1	0	1	0	0	0
0	0	0	1	0	1	0	0	1	0	1	0	1	0	1	1	0	1	0	1	1
0	0	0	1	1	1	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1	1	0	0	1
0	0	1	1	0	1	0	0	1	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1

组合逻辑电路的设计

变式 1：

思考：

含有 AB = 10，CD =10 或 EF = 10 的项属于什么?

根据实际应用的背景，由于两位二进制数最多能表示四种状态而我们只用了三个，因此 AB = 10，CD = 10 ，EF = 10 的输入状态不会出现，即属于我们在第二章中学习的无关项（具体为无关项中的约束项）；

或者说，如果解题时是按照实际情况来列写真值表，例如对于本题来说，三个人，三种表决意见，应该对应 $3^3 = 27$ 种情况，而 6 个输入变量的逻辑函数的真值表应该一共有 $2^6 = 64$ 行，说明按照实际情况写出来的 27 行的真值表并不完整，而缺少的没有写出来的其实就是无关项;

组合逻辑电路的设计

注意：
虽然这个题考察的概率不大，
但是开发思路是很有必要的！

变式 1：

思考：

如果要求根据上面的分析，要求写出逻辑函数式的最简与或式，
（即属于第二章学习的含有无关项的逻辑函数式的化简类问题）
不画卡诺图的话，如何处理？

对于上面这种超过 5 个输入变量的逻辑函数，再用卡诺图去进行化简是繁琐且不现实的，事实上这个时候考察的是对卡诺图和无关项的理解程度；例如，对于本题，联系实际的应用背景，由于 10 是不可能出现的输入情况，因此每个人的第一位为 1 即可以代表其同意，每个人的第二位为0 即可代表其不同意；当有两个人同意时，第三个人的意见无所谓；当只有一个人同意时，第三个人的第二位只要不是 0 而是 1 即可；

组合逻辑电路的设计

思考：
如何证明最终化简的这个结果是正确的？
（根据此结果列出真值表与前面比对）

变式 1：

根据无关项的概念基于实际应用背景化简：

$$Y = \underbrace{A'B'CDEF}_{CDEF} + \underbrace{A'BC'DEF}_{BDEF} + \underbrace{A'BCDE'F}_{BCDF} + \underbrace{A'BCDEF}_{CDEF} + \underbrace{ABC'D'EF}_{ABEF} + \underbrace{ABC'DE'F}_{ABDF} + \underbrace{ABC'DEF}_{ABEF} + \underbrace{ABCDE'F'}_{ABCD} + \underbrace{ABCDE'F}_{ABCD} + \underbrace{ABCDEF}_{ABCD}$$

$$Y = \underbrace{ABCD}_{AC} + \underbrace{ABDF}_{ADF} + \underbrace{ABEF}_{AE} + \underbrace{BCDF}_{BCF} + \underbrace{BDEF}_{BDE} + \underbrace{CDEF}_{CE}$$

$$Y = AC + ADF + AE + BCF + BDE + CE$$ （也可以直接化简第一行）

组合逻辑电路的设计

变式 2：

现题目变为四人表决电路，每个人只能表示“同意”/“不同意”，不能弃权；如果指示灯为 RGB 三色 LED 灯：若“同意”的数量大于“不同意”的数量则表决通过，指示灯显示绿色；若“同意”的数量等于“不同意”的数量表决结果无效，指示灯显示蓝色；若“同意”的数量小于“不同意”的数量则表决不通过，指示灯显示红色；

思考：
Q：在进行第一步逻辑抽象时，输入需要多少个变量？输出需要多少个变量？ A：需要四个输入变量，需要两个输出变量；
Q：增加一个输出变量，真值表的行数会发生变化吗？完整的真值表应该有多少行？ A：不会，只会增加一列；即真值表的行数只取决于输入变量的个数；对于本例即 $2^4 = 16$ 行；
Q：如果规定 $Y_1Y_0 = 11$ 为绿色，$Y_1Y_0 = 01$ 为蓝色，$Y_1Y_0 = 00$ 为红色，需要考虑 $Y_1Y_0 = 10$ 吗？ A：不需要，因为逻辑函数是输入决定输出，在有效的输入范围内不会出现其他输出状态；

组合逻辑电路的设计

本章题目的多样性、多变性大多来自于逻辑抽象
取决于平时积累，多练习！

小结

根据前面的实例及其变式，解决组合逻辑电路这一章的问题最关键最重要的就是第一步逻辑抽象，要能够理解第一章编码的概念，即如何选取输入、输出变量以能够覆盖表示输入输出的所有状态；一旦逻辑抽象完成，那后面的任务实际上就是第二章的基础知识；

在逻辑抽象和后续逻辑函数式的化简时注意：
①对于每个对象（事件）来说，n 位最多能表示其 2^n 种状态；
②根据实际的应用背景列写完真值表后，检查真值表是否完整以及是否保证输入—输出的映射关系即每一组输入只能对应一个输出值；对于 n 个输入变量、m 个输出变量，则完整的真值表应该有 2^n 行，n+m 列且不能出现输入相同而输出不同的行；
③将实际应用问题进行逻辑抽象的同时要考虑无关项的存在，或者说当根据实际的情况列写的真值表不完整时，缺少的项即为无关项；此时这种实际应用问题中的含有无关项的逻辑函数式化简一般可以结合实际应用背景（事实上考察的是概率论）；

常用的组合逻辑电路模块

思考：
这种“封装”、“打包”的思想在哪里学过？
(模电中的集成运放、
编程课中的函数)

“集成化” 的思想

事实上，如果本章只讲到前面一页，组合逻辑电路这一章就可以认为完成了，因为根据第二章逻辑代数的基础知识，任何复杂的逻辑运算关系都可以用最基本的“与、或、非”运算去表示，因此任何一个实际应用问题中的组合逻辑电路都可以用最基本的门电路去设计；

而根据人们不断积累的设计经验，有很多场合下的逻辑问题是具有共性的，为了更方便地设计电路，通常可以将一些具有特定功能、解决某类问题的电路封装打包即集成化，组成一些常用的组合逻辑电路模块（中规模集成器件）；在设计大规模的集成电路时，就可以调用这些典型的电路模块而避免使用复杂且数量巨大的分立的门电路；

本章的另一个重要内容就是如何根据要求使用的中规模集成器件来设计组合逻辑电路；

编码器

编码器

根据第一章的概念，编码就是将每一个事物（事件）用一个代码表示，即从现实的物理世界抽象到逻辑世界；数字电路中的编码器是在已经实现了非电信号到电信号的转换、模拟信号到数字信号的转换之后对信息进行编码，即此时输入的信号是一组高、低电平 （1 和 0 ）；编码器的功能就是根据输入的每一个高、低电平来输出其对应的二进制代码；

根据二进制代码的概念，n 位二进制代码最多能表示 2^n 种物理状态；因此对于编码器来说，输入 n 个电平信号，至少需要 $\log_2 n$ 个输出端口即 $\log_2 n$ 位二进制编码（向上取整）；

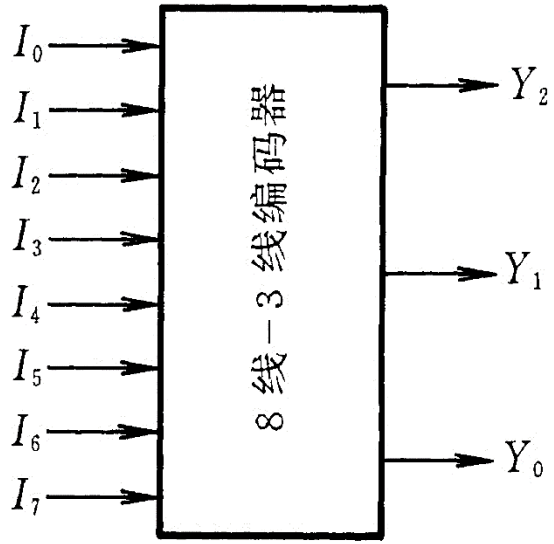
现实生活中的编码器实例例如键盘有 101 个键，按下每个键对应相当于改变输入电平状态，通过编码器将 101 个键盘的状态转换为 7 条数据线上的代码值，而计算机通过编码就可以确定用户按下的键；（计算机翻译这个代码的含义的过程就是下一节学习的译码）

编码器

普通编码器

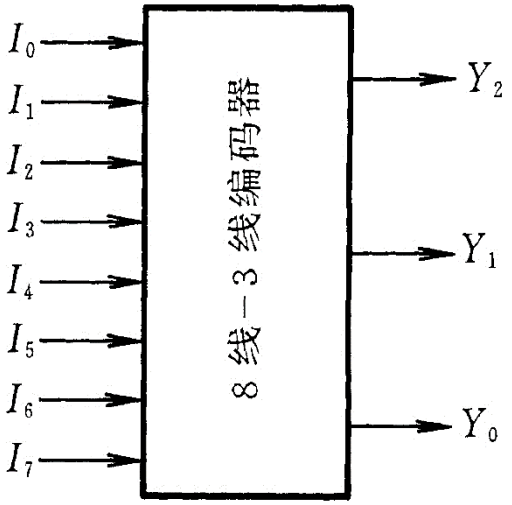
特点：任何时刻有且只有一个输入信号被编码，即只允许一个有效电平输入；

如图所示，为三位二进制普通编码器，根据编码的概念，其有 8 个输入端，3 个输出端；规定高电平有效输入，即输入高电平（为 1）表示对此输入进行编码；输出的 $Y_2 Y_1 Y_0$ 即为 0 ~ 7 对应的三位二进制代码 000~111；



编码器

普通编码器设计示例 —— 8/3 普通编码器



根据其逻辑功能，三位二进制普通编码器的真值表如下：

输入								输出		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

思考：
这个真值表完整吗？缺少什么？

（表格不完整，完整的表格应该有 2^8 行，
因为普通编码器规定了任意时刻有且只有一个输入信号，
属于给定的约束条件，
因此没有给出的行对应的最小项都是无关项）

编码器

普通编码器设计示例 —— 8/3 普通编码器

根据真值表列写其逻辑函数式，并进行化简：

注意这是含有无关项的化简，此时并不需要通过卡诺图来进行化简，直接结合实际应用背景无关项的概念即可，因为任一时刻只能有一个输入为 1，因此只需要将输入为 1 的变量保留，其余的变量无关（相当于在卡诺图中 x 的方块填入 1 合并消去无关变量）；

输入								输出		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

最小项之和的形式：

Y_2 = ……;
Y_1 = ……;
Y_0 = ……;

此处省略；

化简为最简与或式：

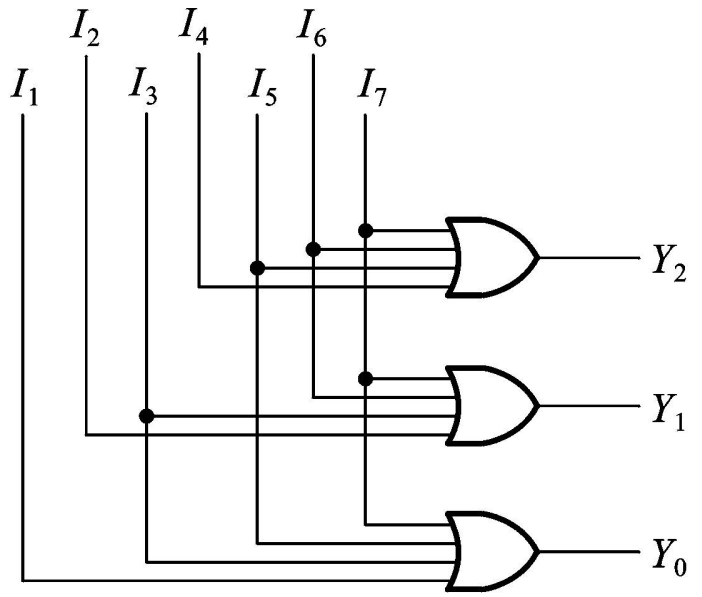
$$Y_2 = I_4 + I_5 + I_6 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_0 = I_1 + I_3 + I_5 + I_7$$

编码器

普通编码器设计示例 —— 8/3 普通编码器

根据其逻辑函数式绘制其对应的逻辑电路图：

$$Y_2 = I_4 + I_5 + I_6 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_0 = I_1 + I_3 + I_5 + I_7$$



此设计存在的问题：不能有多个信号同时输入，也不能没有信号输入（因为根据我们的约束条件，00000000 也属于无关项）；因此我们以上的设计只是一个思路的训练，最终的设计结果并不能直接应用到实际当中；

编码器

优先编码器

特点：允许多个信号同时输入，编码时存在优先顺序；

以设计一个 8/3 优先编码器为例：

规定高电平输入有效，即输入 1 表示对此输入信号编码，
输出的 $Y_2Y_1Y_0$ 即为 0 ~ 7 对应的三位二进制代码 000~111；
规定二进制数大的优先级高，即 I_7 优先级最高，I_0 最低；

则真值表如右图所示：

养成习惯，思考：此真值表是否完整?
不完整！当前只有 $2^8 - 1$ 行，缺少了一行 00000000 ！

输入								输出		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0
X	X	X	X	X	X	X	1	1	1	1
X	X	X	X	X	X	1	0	1	1	0
X	X	X	X	X	1	0	0	1	0	1
X	X	X	X	1	0	0	0	1	0	0
X	X	X	1	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0
X	1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0

编码器

实际应用的中规模集成芯片的附加端

继续前一页的分析设计，当前列出的真值表并不完整，缺少了 00000000 这一行，也就是没有信号输入被编码的情况；如果类似前面我们设计的 8/3 普通编码器一样简单粗暴地规定不允许输入 00000000 即作为约束条件，那么在很多实际应用场合中是不现实的；

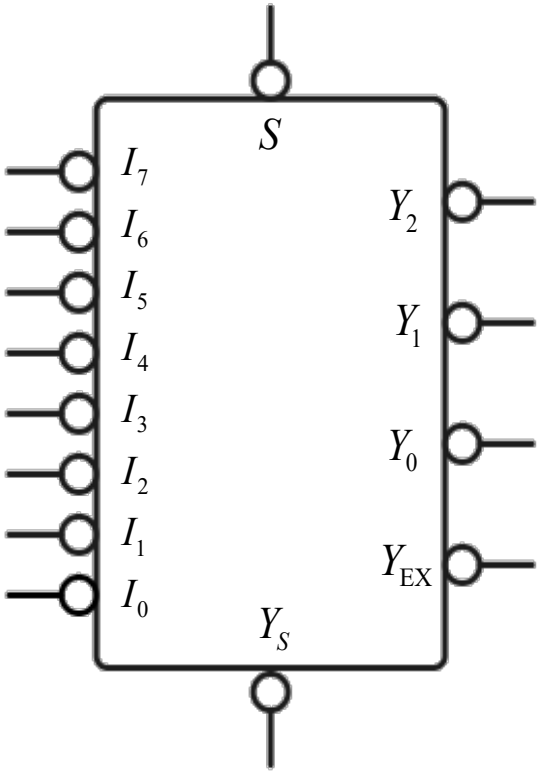
如果将输入全为 0 即没有输入作为无关项，那么设计出来的逻辑电路会出现的一个问题就是输入全为 0 时输出为 000，而 000 同时又是 I_0 输入对应的编码；此时存在的问题就是当前的三个输出变量即三位二进制数只能表示 8 种状态，而我们还需要表示另外一种“没有信号输入”的状态，因此解决的办法就是增加输出端即增加输出代码的位数；

在实际应用的 8/3 优先编码器中，还附加了其他的输入输出端口，以便于片选和拓展；具体如下页 74HC148 所示；

编码器

优先编码器典型芯片 —— 74HC148 （ 8/3 优先编码器）

一个典型的 8 线 — 3 线 优先编码器芯片为 74HC148：



I_7 ~ I_0：被编码信号的输入端，带小圆圈表示低电平输入有效，从 I_7 到 I_0 优先级依次降低；

Y_2 ~ Y_0 ： 编码后信号的输出端，带小圆圈表示低电平输出有效；

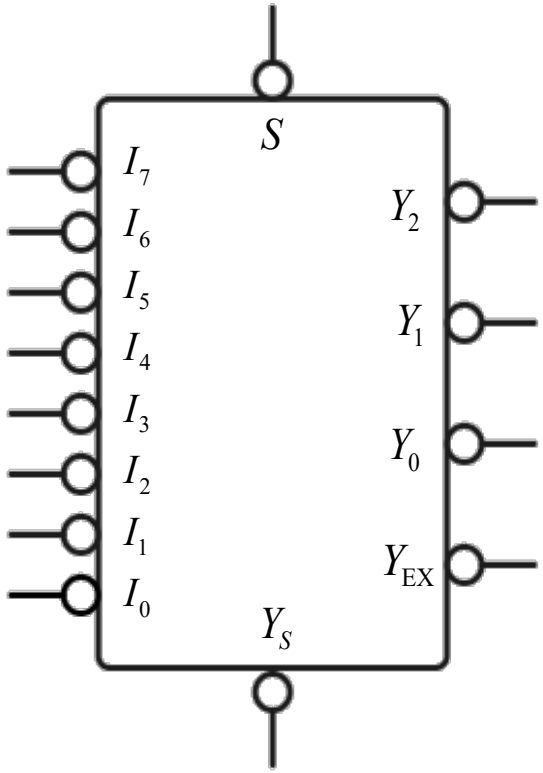
S： 选通输入端，Select，带小圆圈表示低电平输入有效；

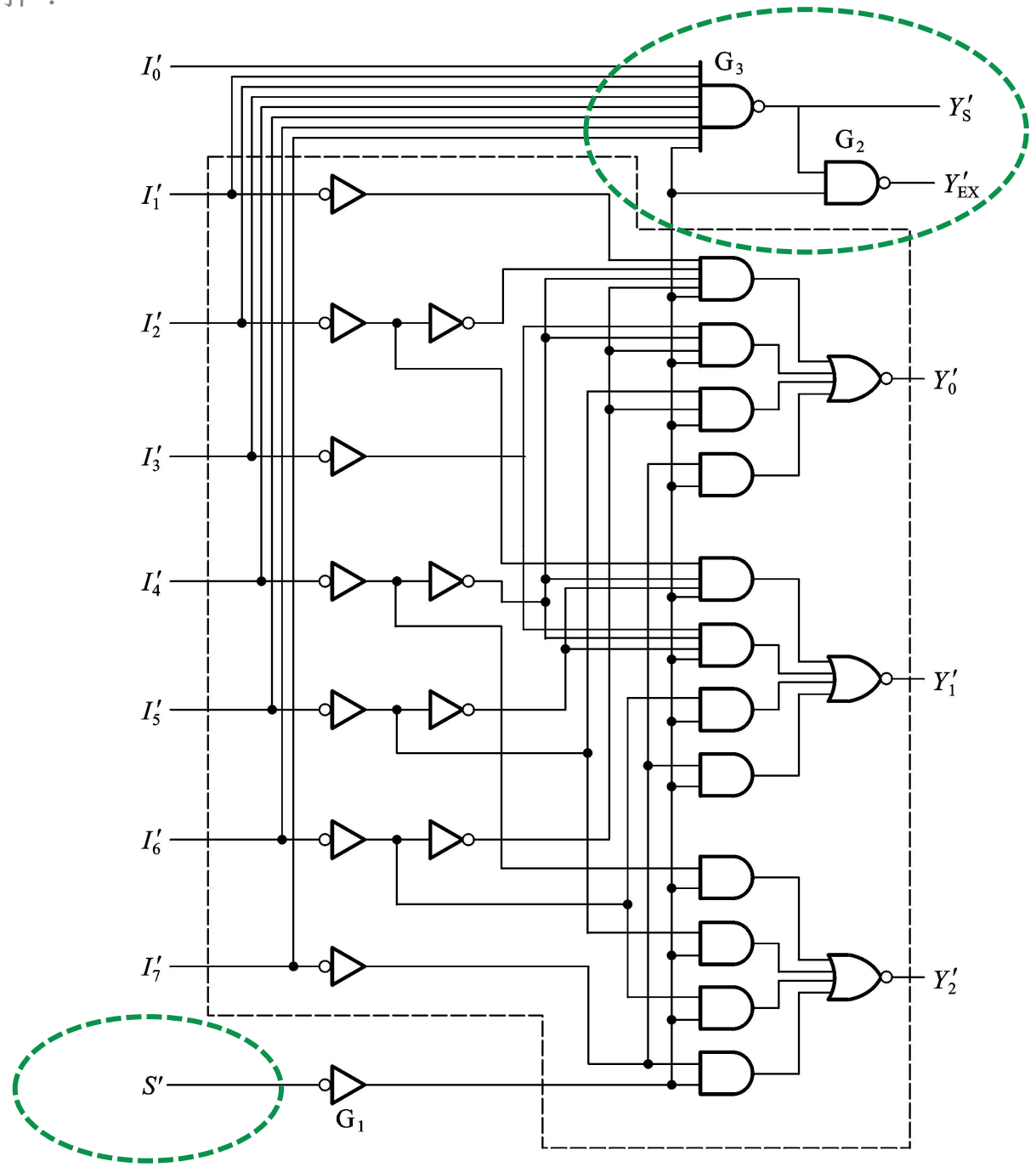
Y_S： 选通输出端，带小圆圈表示低电平输出有效；

Y_{EX}： 拓展输出端，Expand，带小圆圈表示低电平输入有效；

编码器

74HC148 内部结构（了解）





编码器

选通输入端

$$Y_2' = (I_7 + I_6 + I_5 + I_4)'$$
$$Y_1' = (I_7 + I_6 + I_5' I_4' I_3 + I_5' I_4' I_2)'$$
$$Y_0' = (I_7 + I_6' I_5 + I_6' I_4' I_3 + I_6' I_4' I_2' I_1)'$$

$$Y_2' = ((I_7 + I_6 + I_5 + I_4) \cdot S)'$$
$$Y_1' = ((I_7 + I_6 + I_5' I_4' I_3 + I_5' I_4' I_2) \cdot S)'$$
$$Y_0' = ((I_7 + I_6' I_5 + I_6' I_4' I_3 + I_6' I_4' I_2' I_1) \cdot S)'$$

（这两组逻辑函数式并不需要掌握，
只是为了帮助理解如何实现的选通）

注意：
如果给定的其他芯片的选通输入端
不带小圆圈，用原变量表示，
则高电平输入有效，
即 输入 1 选通芯片；

可见 S 信号的作用是在原本的优先编码器的基础上进行与每一个输入信号相与，因此：

S = 0，即 S′ = 1 即无效的选通信号端输入时：
此芯片不工作，全部输出端被封锁在高电平；

S = 1，即 S′ = 0 即有效的选通信号端输入时：
芯片可以正常工作；

（选通输入端的特点才是需要掌握的！）

编码器

选通输出端和拓展端

$$Y_S' = (I_7'I_6'I_5'I_4'I_3'I_2'I_1'I_0'S)'$$

$$Y_{EX}' = [(I_7'I_6'I_5'I_4'I_3'I_2'I_1'I_0'S)' \quad S]'$$

$$[(I_7 + I_6 + I_5 + I_4 + I_3 + I_2 + I_1 + I_0)S]'$$

（这两个逻辑函数式并不需要掌握，
只是为了帮助理解两者的特点）

- 当 S = 0 即 S′ = 1 即芯片不工作时：
  全部输出端包括 Y_S' 和 Y_{EX}' 都为高电平；

- 当 S = 1 即 S′ = 0 即芯片被选通时：

  若没有信号输入参与编码，即 I_7' ~ I_0' 全部为 1：
  Ys′ = 0，Y_{EX}' = 1；
  若有信号输入参与编码，即 I_7' ~ I_0' 至少有一个为 0：
  Ys′ = 1，Y_{EX}' = 0；

（选通输出端与拓展输出端的特点需要掌握！）

编码器

74HC148 的功能表

输入									输出				
S'	I_0'	I_1'	I_2'	I_3'	I_4'	I_5'	I_6'	I_7'	Y_2'	Y_1'	Y_0'	Y_S'	Y_{EX}'
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
0	X	X	X	X	X	X	X	0	0	0	0	1	0
0	X	X	X	X	X	X	0	1	0	0	1	1	0
0	X	X	X	X	X	0	1	1	0	1	0	1	0
0	X	X	X	X	0	1	1	1	0	1	1	1	0
0	X	X	X	0	1	1	1	1	1	0	0	1	0
0	X	X	0	1	1	1	1	1	1	0	1	1	0
0	X	0	1	1	1	1	1	1	1	1	0	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0

S'	Y_S'	Y_{EX}'	状态
1	1	1	不工作
0	0	1	工作，但无信号输入
0	1	0	工作，且有信号输入
×	0	0	不可能出现

正常的编码工作区域
即 S′ = 0 选通，
I_7' ~ I_0' 至少有一个为 0
（有效输入）
注意是低电平有效输出！

编码器

附加端的作用

根据前面的分析，通过引入这些附加输入端和输出端就可以对相同的输出信号进行区分：例如 $Y_2Y_1Y_0=000$ 即 $Y_2'Y_1'Y_0'=111$ 时，对应的是三种不同的状态：

	Y_S'	Y_{EX}'	含义
$Y_2'Y_1'Y_0'=111$	1	1	芯片不工作
	0	1	工作，但没有信号输入被编码
	1	0	工作，且有信号输入被编码

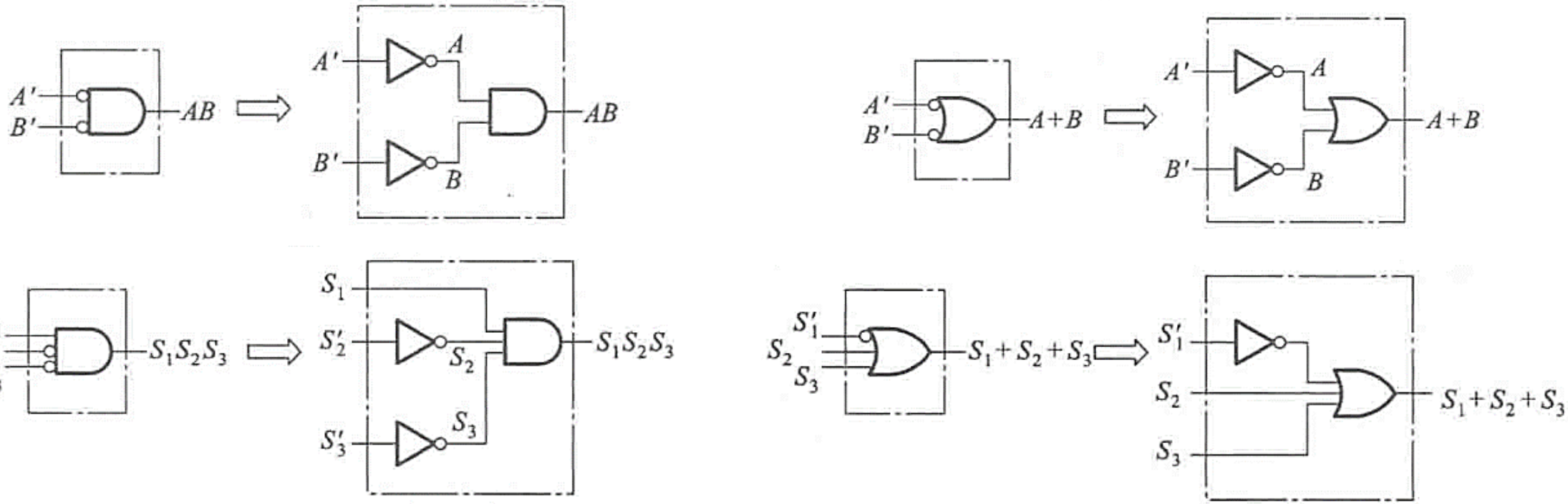
除此之外，这些附加端被设计的另一重要用途是可以实现功能的拓展；例如当需要编码的位数增加时，16 / 4 编码器、32 / 5 编码器都可以通过 8 / 3 编码器拓展组装搭建，满足电路需求的灵活性和多样性；

编码器

对于低电平有效的信号，
器件引脚的名称仍用原变量（矩形里边），
（即内部实际的输入端口）
外部输入信号用反变量表示，且加一个小圆圈，
就可以保持分析时是一致的；

关于“低电平有效”的理解

如何理解一些中规模集成器件输入端口和输出端口处的“小圆圈”？



即看作输入信号通过一个反相器才加载到后面的输入端中；

编码器

用附加端实现拓展实例
—— 将 8/3 优先编码器拓展为 16/4 优先编码器

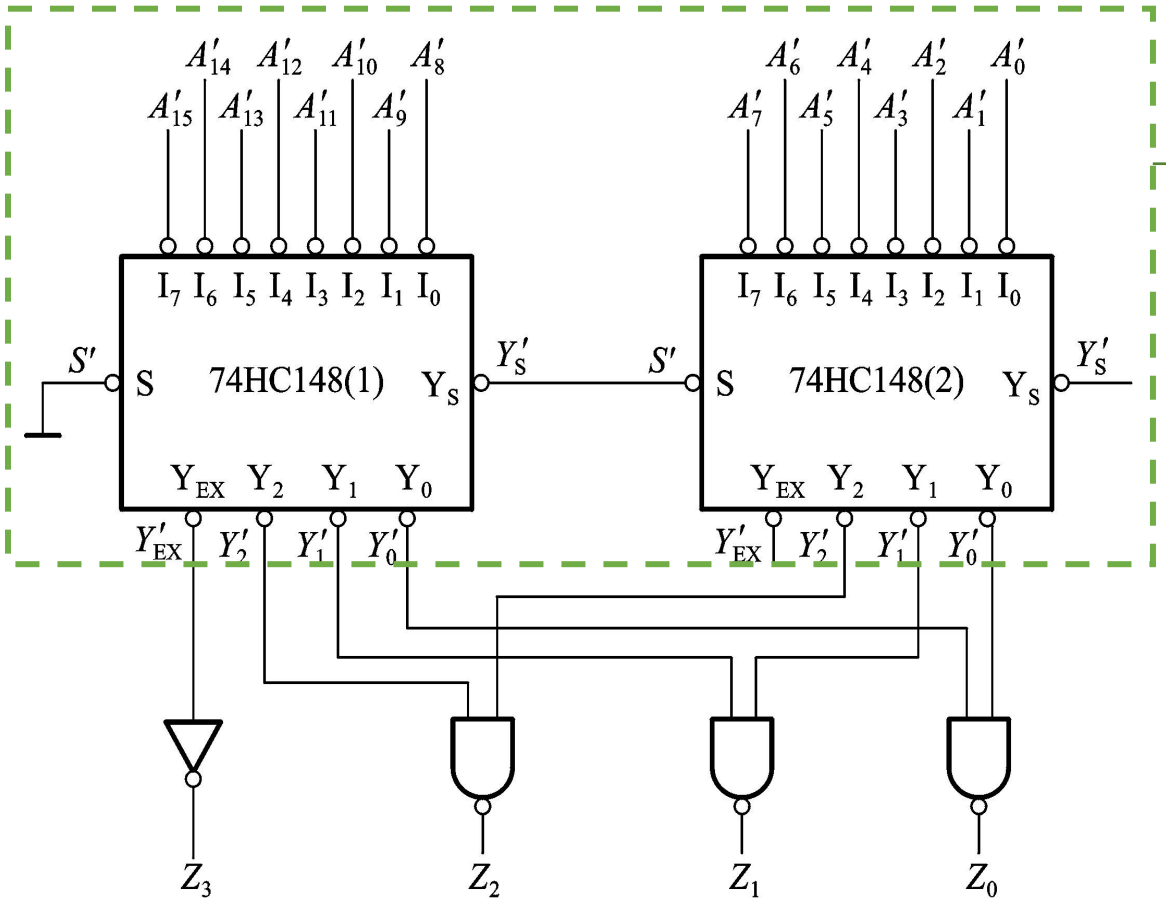
要求用两片 74HC148 接成 16 线 — 4 线优先编码器，将 A_{15}' ~ A_0' 16 个低电平输入信号编码为 1111 ~ 0000 16 个四位二进制代码，其中 A_{15}' 的优先权最高，A_0' 的优先权最低；

思路：

将 A_{15}' ~ A_0' 分为两组输入到两片 74HC148 的信号输入端；
由于优先级的存在，要求 A_{15}' ~ A_8' 有信号输入参与编码时，另一片不能工作；
而表示是否有信号输入参与编码即 Y_S' 端口，因此用第一片的选通输出端去控制第二片的选通输入端即可；
两组四位二进制编码的区别就是最高位为 1 / 0，低三位都是相同的；因此将 Y_{EX}' 作为最高位即可；

编码器

用附加端实现拓展实例
—— 将 8/3 优先编码器拓展为 16/4 优先编码器



分为两组输入；

第一片的输入信号优先级最高，其选通输入端始终为低电平（接地）即可；

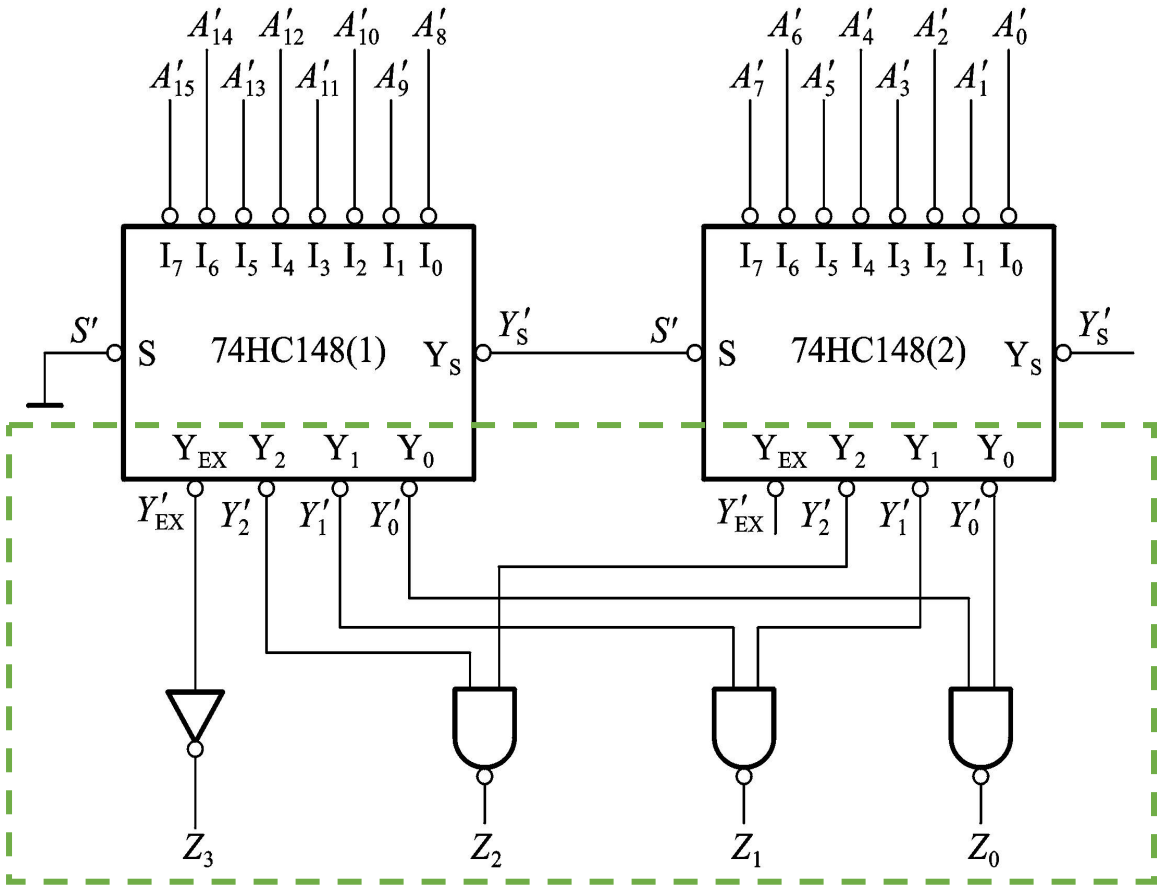
当第一片有任何一个有效信号输入即需要编码，则其 Y_S' = 1，对应第二片 S' = 1，第二片不允许工作，全部输出封锁在高电平；

当第一片没有有效输入信号即不编码，其 Y_S' = 0，则第二片允许工作；

编码器

关于低电平有效输出，
每个人的理解方式不相同，
找到一种有助于自己理解和记忆的方式，
不把自己绕晕即可！

用附加端实现拓展实例
—— 将 8/3 优先编码器拓展为 16/4 优先编码器



由于题目要求的是 A_{15}' 对应的编码是 1111，A_0' 的编码是 0000，而 74HC148 是低电平输出有效，因此在用输出端信号表示最终编码结果时要注意：

最高位使用第一片的 Y_{EX}' 表示，由于第一片编码时 Y_{EX}' 为 0，而 8~15 的最高位为 1，因此需要反相器；

由于低三位不是取第一片的输出就是取第二片的输出，因此逻辑上是“或”的关系；但由于低电平输出有效，因此实际电路为与非门；

编码器

用附加端实现拓展实例

变式 1 —— 在前面的基础上，如果想将设计的电路封装成一个类似于 74HC148 的 16/4 优先编码器集成化电路，请补充连线并标注端口；（提示：缺少拓展输出端）

思路：

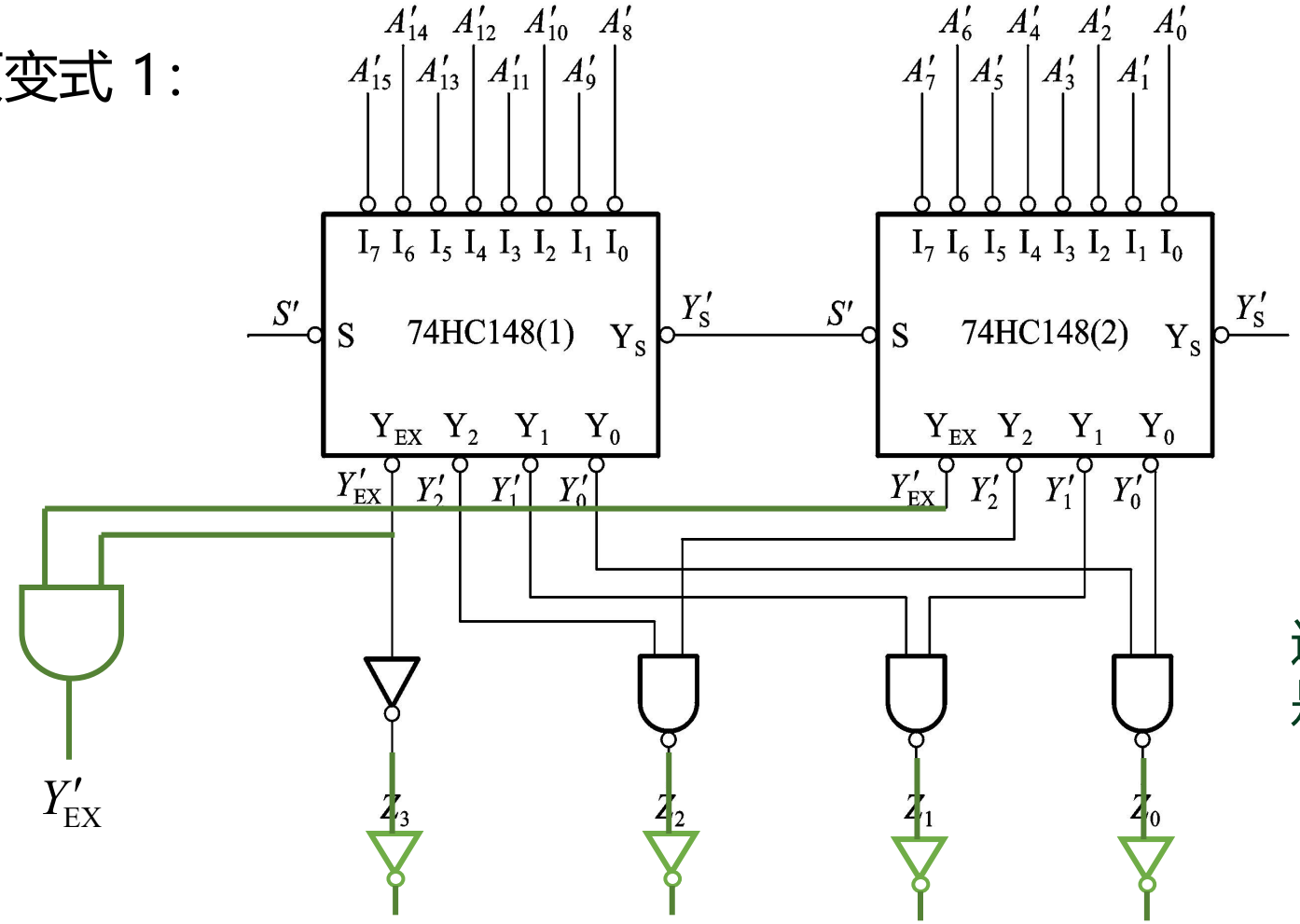
根据前面的设计分析可知，第一片的选通输入端可以作为整体的选通输入端，第二片的选通输出端可以作为整体的选通输出端，而如果想封装继续拓展的话，还需要引出一个拓展输出端；

拓展输出端应满足任何一片有编码信号输入时此端口有效输出低电平，而这两片的任何一个 Y_{EX}' 为 0 都能代表系统在进行编码工作，逻辑上是“或”的关系，而低电平输出有效，因此使用与门相连则是负逻辑低电平有效输出的拓展输出端；

编码器

用附加端实现拓展实例

接上一页变式 1：



即可打包封装为
16 / 4 优先编码器

注意：
这里每个输出端再加一个反相器
是为了保持封装后的优先编码器
仍然为低电平输出有效

编码器

用附加端实现拓展实例

变式 2 —— 试用 4 片 74HC148 接成 32 线 — 5 线优先编码器，将 A_{31}' ~ A_0' 32 个低电平输入信号编码为 11111 ~ 00000 32 个五位二进制代码，其中 A_{31}' 的优先权最高，A_0' 的优先权最低；

思路：

变式 2 可以选择在变式 1 的基础上解决；

也可以类似前面的思路直接进行设计，因为其实需要认真考虑一下的也就是输出端的接法；

编码器

用附加端实现拓展实例

如果在 【变式 1】 的基础上设计实现 【变式 2】；

根据变式 1，每个 16/4 优先编码器的 Y_3' 是内部优先级较高的 8/3 的 Y_{EX}' ；
Y_2' 、Y_1' 、Y_0' 是内部两片 8/3 的 Y_2' 、Y_1' 、Y_0' 的与门运算结果；
Y_{EX}' 是内部两片 8/3 的 Y_{EX}' 与门运算结果；

因此，对于 32/5 优先编码器：（这里为了防止混淆最终输出的代码用 $D_4D_3D_2D_1D_0$ 表示，注意最终输出的代码要求是正逻辑即 32 对应 11111，0 对应 00000）

最低三位是两片 16/4 的 Y_2' 、Y_1' 、Y_0' 的与非，也就是内部四片 8/3 的 Y_2' 、Y_1' 、Y_0' 的与非；
次高位 D_3 是两片的 16/4 的 Y_3' 的与非，因此就是两个 16/4 各自内部优先级较高的 8/3 的 Y_{EX}' 的与非；
最高位 D_4 是优先级较高的 16/4 的 Y_{EX}' 的取反，也就是优先级较高的 16/4 内部两片 8/3 的 Y_{EX}' 的与非；

编码器

“遇事不决”列真值表！

用附加端实现拓展实例

（注意，此页这里给出的真值表不完整但是其余的是无关项，因为任一时刻只可能有一片的拓展输出端为有效输出低电平，不影响后面的分析！）

如果直接设计实现 【变式 2】；

现在最关键的问题就是每个芯片的输出端如何连接、通过什么逻辑运算得到最终编码的五位，在直接思考逻辑关系有些困难的情况下，我们可以列出真值表：（相当于把这些芯片的输出看作是下一级的输入变量，设计一个由门电路搭建的组合逻辑电路，按照前面基本的分析思路即可）

（片的编号按优先级顺序，A_{31}' ~ A_{24}' 为第 1 片，A_7' ~ A_0' 为第 4 片；）

Y_{EX}'(第 1 片)	Y_{EX}'(第 2 片)	Y_{EX}'(第 3 片)	Y_{EX}'(第 4 片)	D_4（最高位）	D_3（次高位）
0	1	1	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	0	0	0

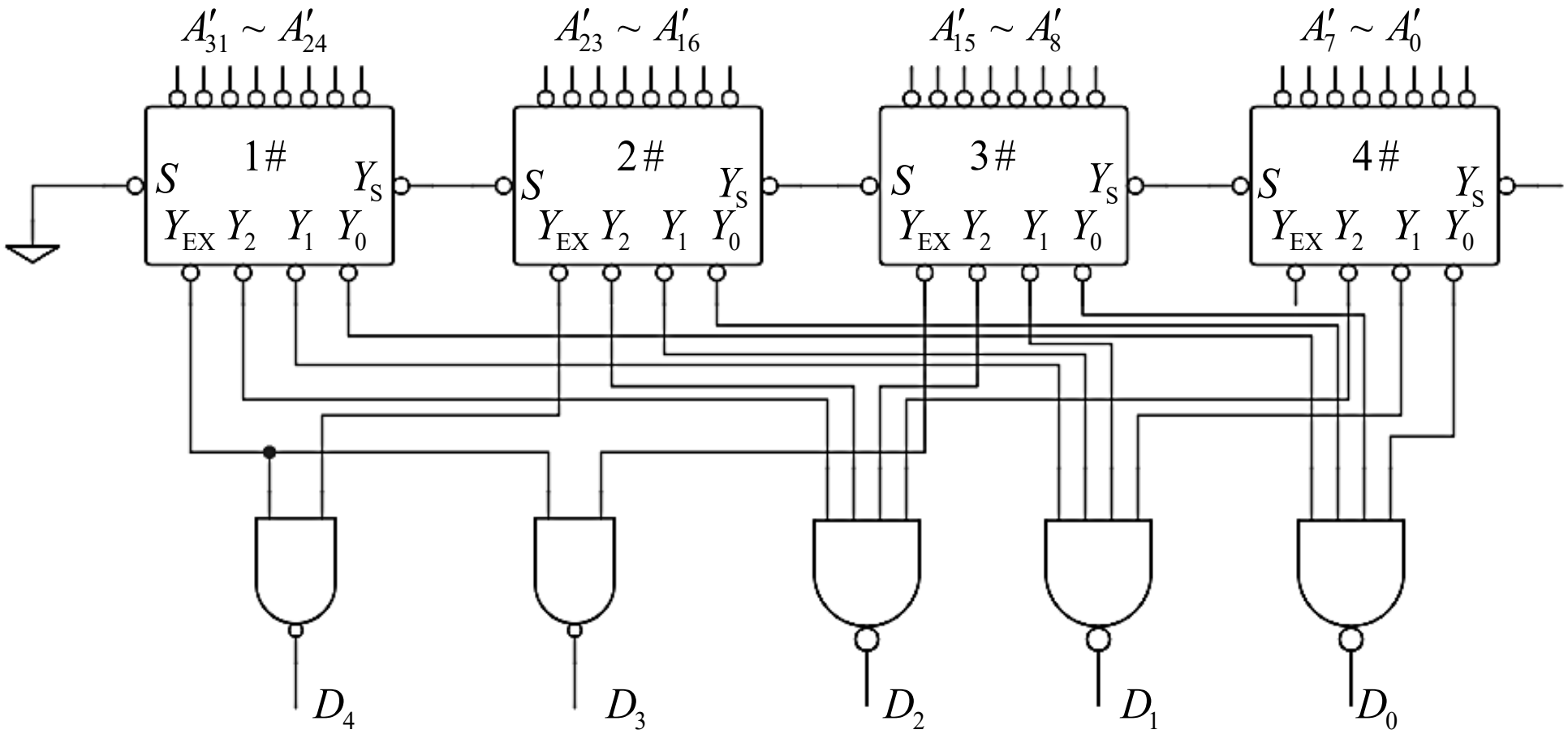
（如果不适应这个真值表（因为输入侧现在是低电平有效负逻辑），可以把 Y_{EX} 作为输入侧变量）

（低三位仍然是四个芯片的三位输出的与非，这里无需再列出）

编码器

用附加端实现拓展实例

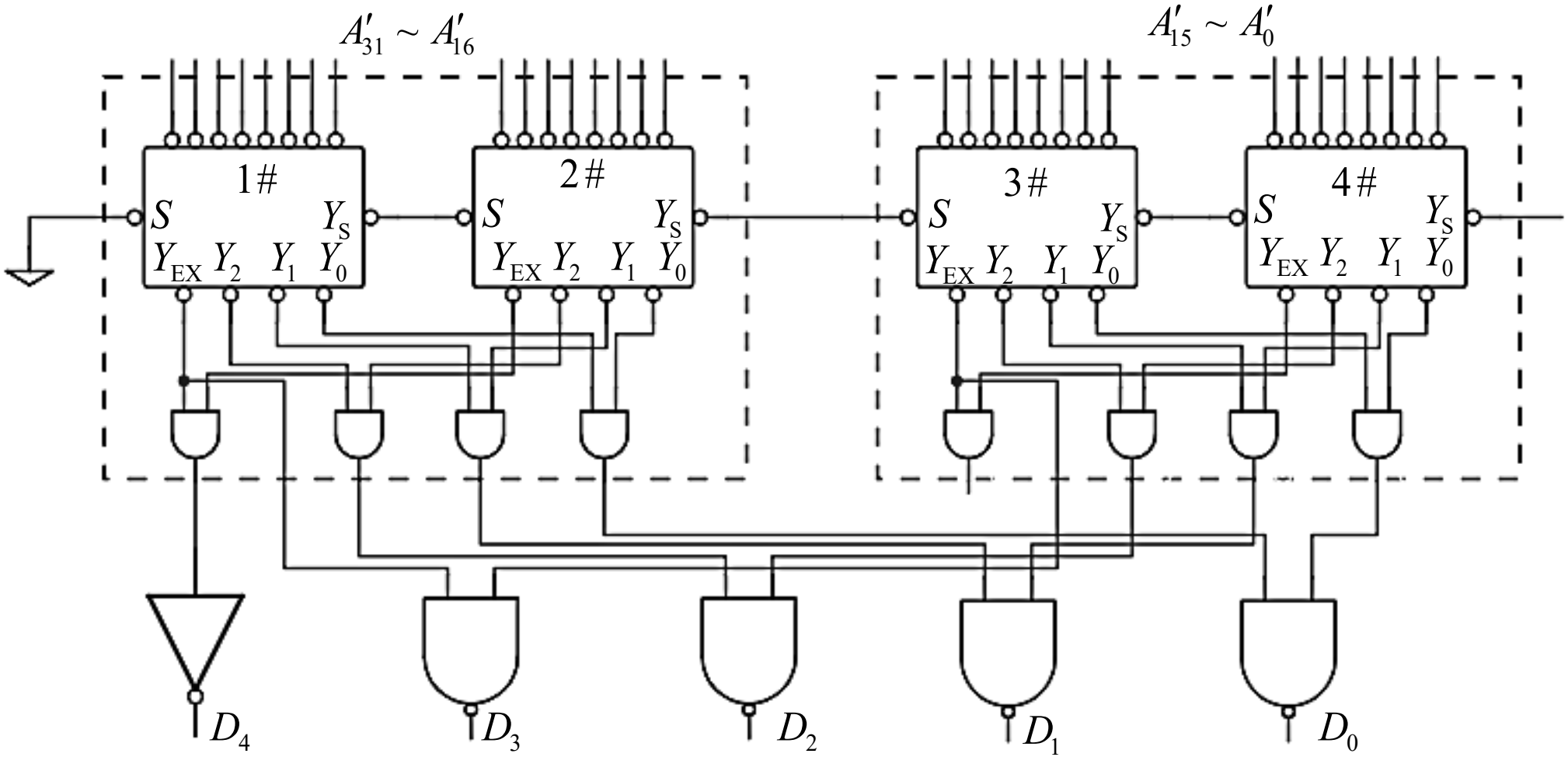
变式 2 接线图如下：



编码器

用附加端实现拓展实例

变式 2 接线图如下（用封装的思想）



编码器

用附加端实现拓展实例

变式 3 —— 如果给定一 8/3 优先编码器芯片其功能表如下所示，请用两片此器件设计一个 16/4 优先编码器，将 A_{15} ~ A_0 16 个高电平输入信号编码为 1111 ~ 0000 16 个四位二进制代码，其中 A_{15} 的优先权最高，A_0 的优先权最低；

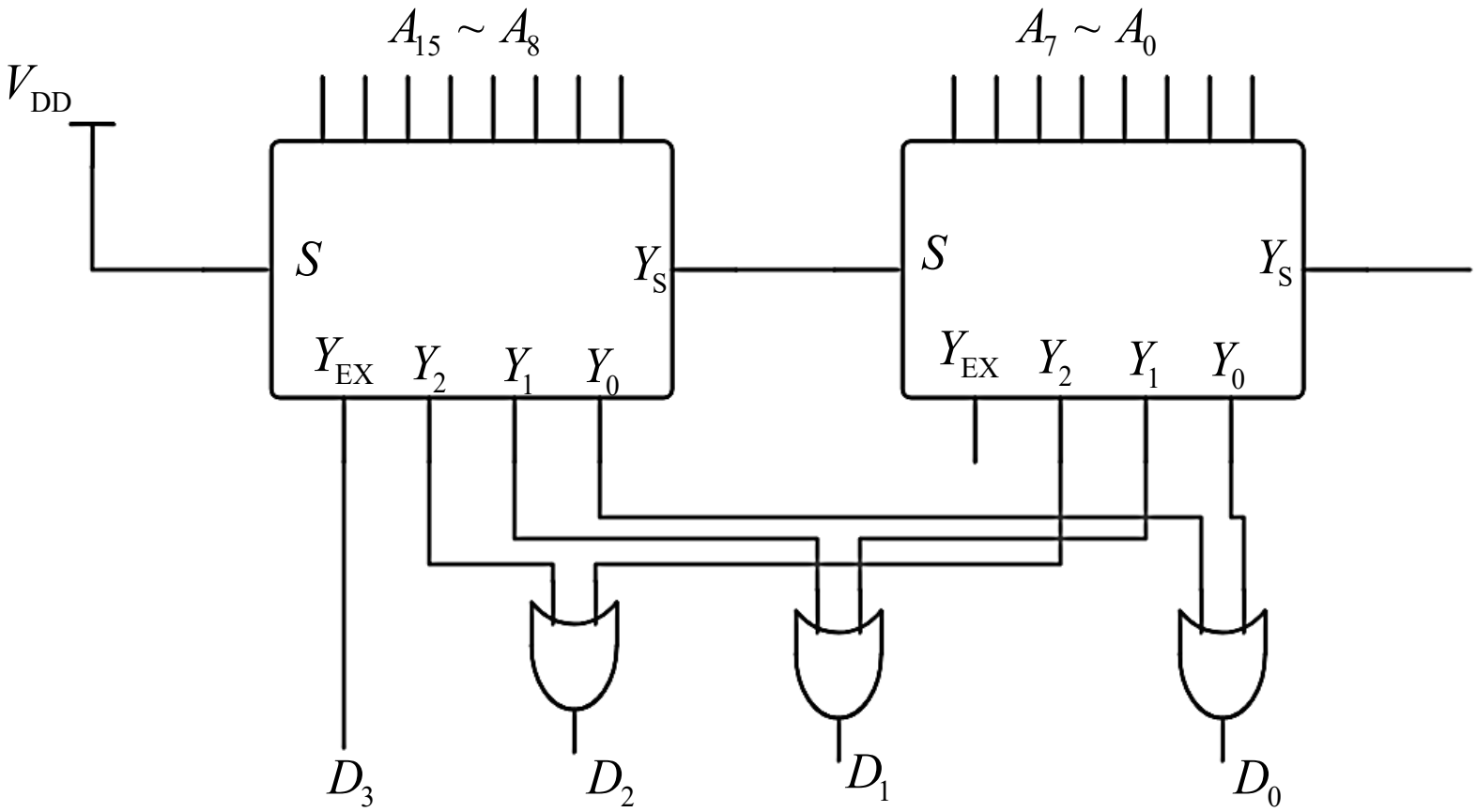
输入									输出				
S	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0	Y_S	Y_{EX}
0	X	X	X	X	X	X	X	X	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	X	X	X	X	X	X	X	1	1	1	1	0	1
1	X	X	X	X	X	X	1	0	1	1	0	0	1
1	X	X	X	X	X	1	0	0	1	0	1	0	1
1	X	X	X	X	1	0	0	0	1	0	0	0	1
1	X	X	X	1	0	0	0	0	0	1	1	0	1
1	X	X	1	0	0	0	0	0	0	1	0	0	1
1	X	1	0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	0	0	0	0	1

思路：
事实上这就是把我们教材上学习的优先编码器变为高电平有效输入输出，基本的思路是一致的；

编码器

用附加端实现拓展实例

变式 3 接线图如下：



其实运用我们前面讲的
把或门变成与非门的小技巧，
就可以体会到
高电平有效输出和
低电平有效输出的联系！

编码器

这也是我们学习后面其他几种电路模块时的关注重点！

编码器小结

- 编码器的作用是将输入的一组高低电平信号（ 2^n 个）编码为 n 位二进制代码输出；即将现实物理世界中的高低电平抽象为逻辑世界中的代码；可联系“键盘”实例记忆理解其特点和功能；普通编码器只允许一个有效信号输入，优先编码器允许多个信号同时输入，存在优先级；典型的 8/3 优先编码器 —— 74HC148（或 74LS148）；

- 在应用中规模器件时，我们不需要再关注其内部的结构，而是只关注其外部输入输出端口的逻辑功能特点，特别需要注意的是一些附加端口，以及每个输入和输出端口是高电平有效还是低电平有效；

- 在应用中规模器件完成其他逻辑功能实现或者是进行拓展时，一般需要通过一些其他的门电路辅助实现，这个时候事实上只需要关注这些中规模器件的外部特性，即相当于把这些芯片的输出看作是下一级的输入信号，再按照前面组合逻辑电路设计的基本思路去完成最终的设计目标即可；需要注意的细节就是高/低电平有效；

编码器

思考：
4 个输出位能表示 16 种状态
需不需要考虑另外的 4 个代码?
（不需要！逻辑函数是输入决定输出！）

二—十进制优先编码器（了解）

二—十进制编码器的功能是将输入信号编码为数字 0~9 的 BCD码（8421码）；

典型的二—十进制优先编码器如 74LS147； 有十个输入端口（其中一个为 NC 即 No Connection ），四个输出端口；（另外两个引脚为电源和地）

（这里教材上有些表述问题，0 没有单独的输入端口，全部输入为高电平时才是代表对 0 编码，输出为 1111 即 0000 的反码）

具体的功能表参考教材即可；

译码器

译码器

功能特点：将输入的二进制代码译成对应的输出高、低电平信号或者另外一个代码；即将输入的代码表示的含义翻译出来；

分类：

二进制译码器 —— 输入为 n 位二进制代码，输出 2^n 个高低电平信号；

二—十进制译码器 —— 输入为 0~9 对应的 BCD 码，输出为 10 个高低电平信号；

显示译码器 ——将数字、字母、文字、符号等对应的二进制代码翻译并显示出来；

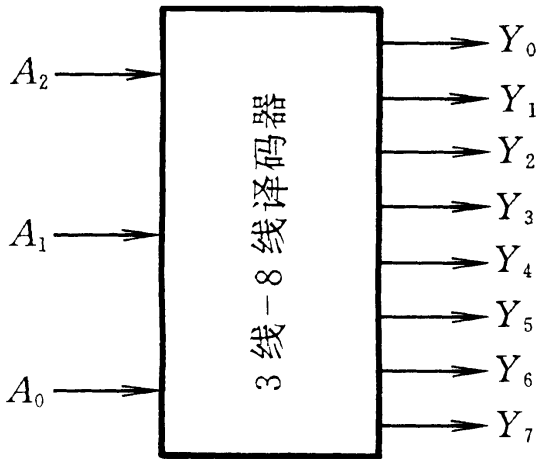
译码器

二进制译码器

功能：输入为 n 位二进制代码，输出 2^n 个高低电平信号；

思考：
真值表是否完整？
（完整！不仅完整，最大的一个优点是包含了全部的最小项，后面会继续讨论）

以三位二进制译码器为例，根据译码的概念，其有 3 个输入端，8 个输出端；3 个输入端构成的 3 位二进制代码能够表示 8 种状态，对应每一个输出端口的高、低电平信号；



输　入			输　出							
A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

译码器

思考：
如果是低电平有效输出的译码器呢?
(输出的是各个最小项取反，
要使用与非门)

二进制译码器的逻辑功能特点

特点：根据二进制译码器的功能以及列出的真值表，2^n 个输出信号恰好对应的是 n 个输入变量的全部最小项；而根据第二章逻辑代数的基础知识，任何一个逻辑函数都可以唯一表示成最小项之和的形式，因此使用 $n/2^n$ 译码器配合或门就可以实现任何形式的 n 变量组合逻辑函数；

输　入			输　出							
A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$$Y_0 = A_2' A_1' A_0'$$

$$\vdots$$

$$Y_7 = A_2 A_1 A_0$$

$$Y_i = m_i$$

译码器

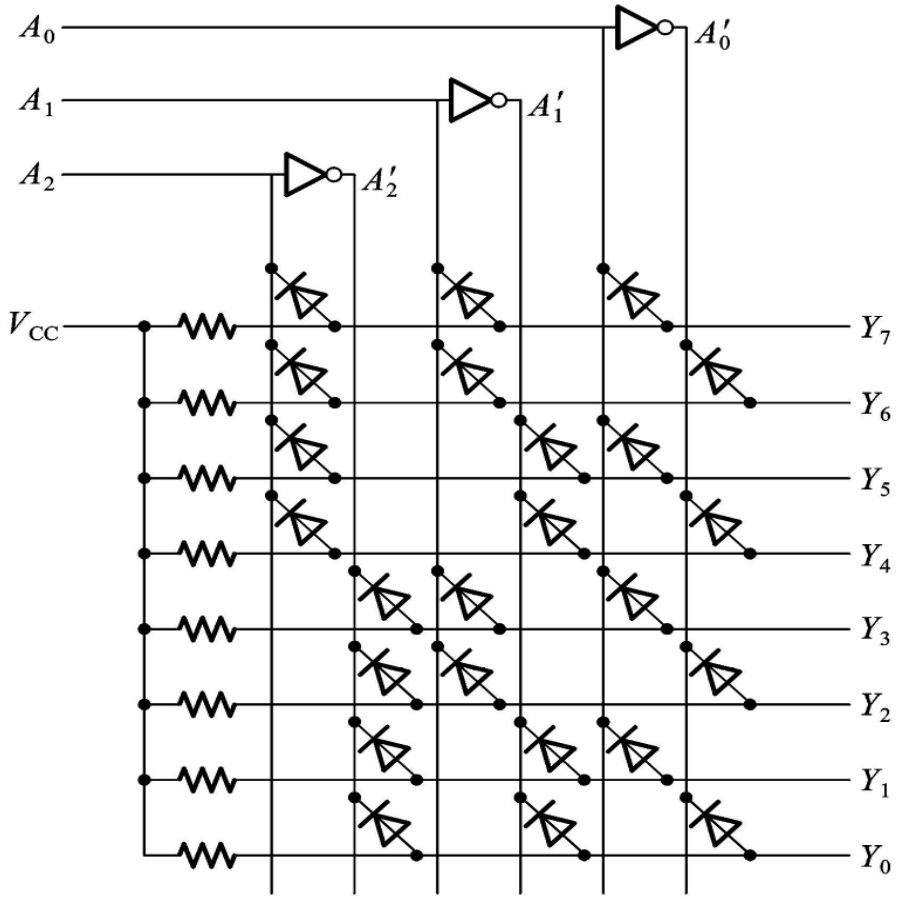
二极管与门阵列实现的 3/8 译码器

$$Y_0 = A_2' A_1' A_0'$$

$$\vdots$$

$$Y_7 = A_2 A_1 A_0$$

$$Y_i = m_i$$



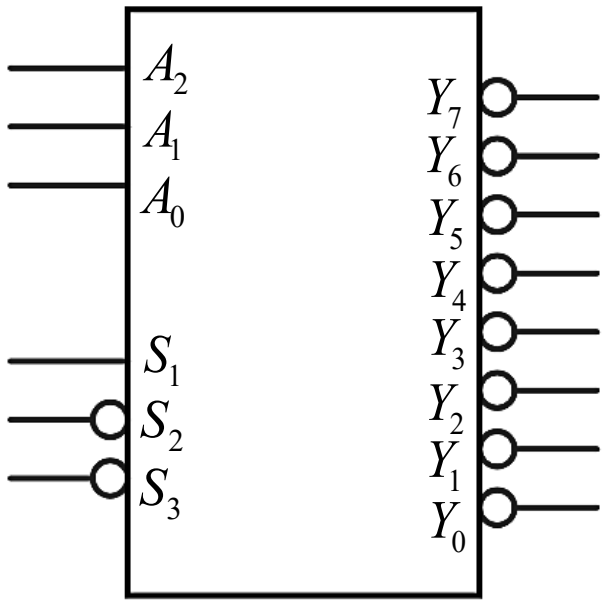
根据第三章的二极管门电路基础知识，共阳二极管与上拉电阻构成二极管与门，因此根据逻辑函数式，设计出如图所示的二极管与门阵列，完成 3/8 译码；

此阵列形式的译码器会在半导体存储器一章中继续讨论，不是本章的重点；

译码器

二进制译码器典型芯片 —— 74HC138 （3/8 译码器）

一个典型的 3 线 — 8 线 译码器芯片为 74HC138：



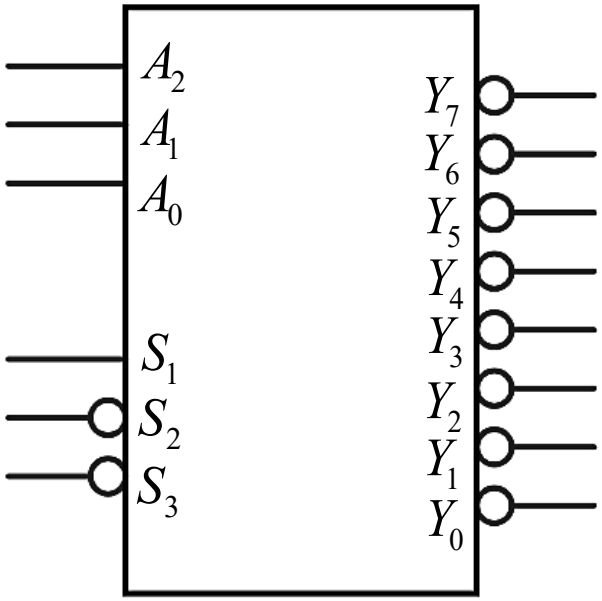
A_2 ~ A_0：二进制编码的输入端，高电平输入有效；

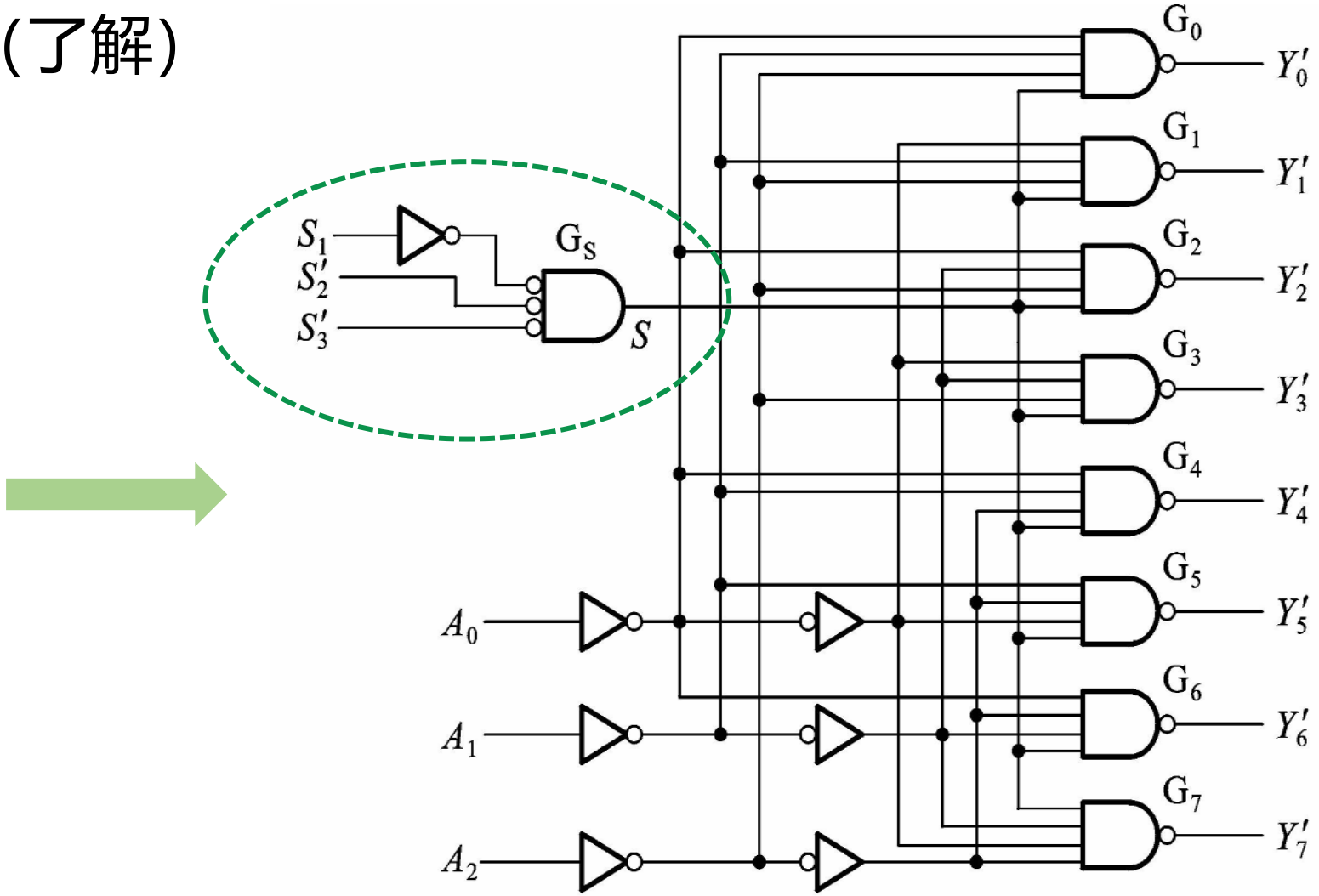
Y_7 ~ Y_0：输出的高低电平信号，带小圆圈表示低电平输出有效；在正常译码时有且仅有一个输出为低电平；

S_1、S_2、S_3：选通输入端，Select，S_1 为高电平输入有效；S_2、S_3 带小圆圈表示低电平输入有效；

译码器

74HC138 内部结构（了解）





译码器

74HC138 的功能表

输入					输出							
S_1	$S_2'+S_3'$	A_2	A_1	A_0	Y_7'	Y_6'	Y_5'	Y_4'	Y_3'	Y_2'	Y_1'	Y_0'
0	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	1	1	1	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	1	1	1	1	1
1	0	1	1	0	1	0	1	1	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1	1	1

只有全部的选通输入端都输入有效信号，即 $S_1 = 1$，$S_2' = 0$，$S_3' = 0$ 时芯片才能正常工作，否则输出全部封锁在高电平；

正常的译码工作区域
芯片选通后，
有且仅有一个输出为低电平
(注意是低电平有效输出！)
即每个输出对应的是
各个最小项的取反；

$$Y_i' = m_i'$$

译码器

用附加端实现拓展实例
—— 将 3/8 译码器拓展为 4/16 译码器

要求用两片 74HC138 接成 4 线 — 16 线译码器，将 D_3 ~ D_0 四位二进制代码输入信号译为 16 个独立的低电平信号 Z_{15}' ~ Z_0' ；

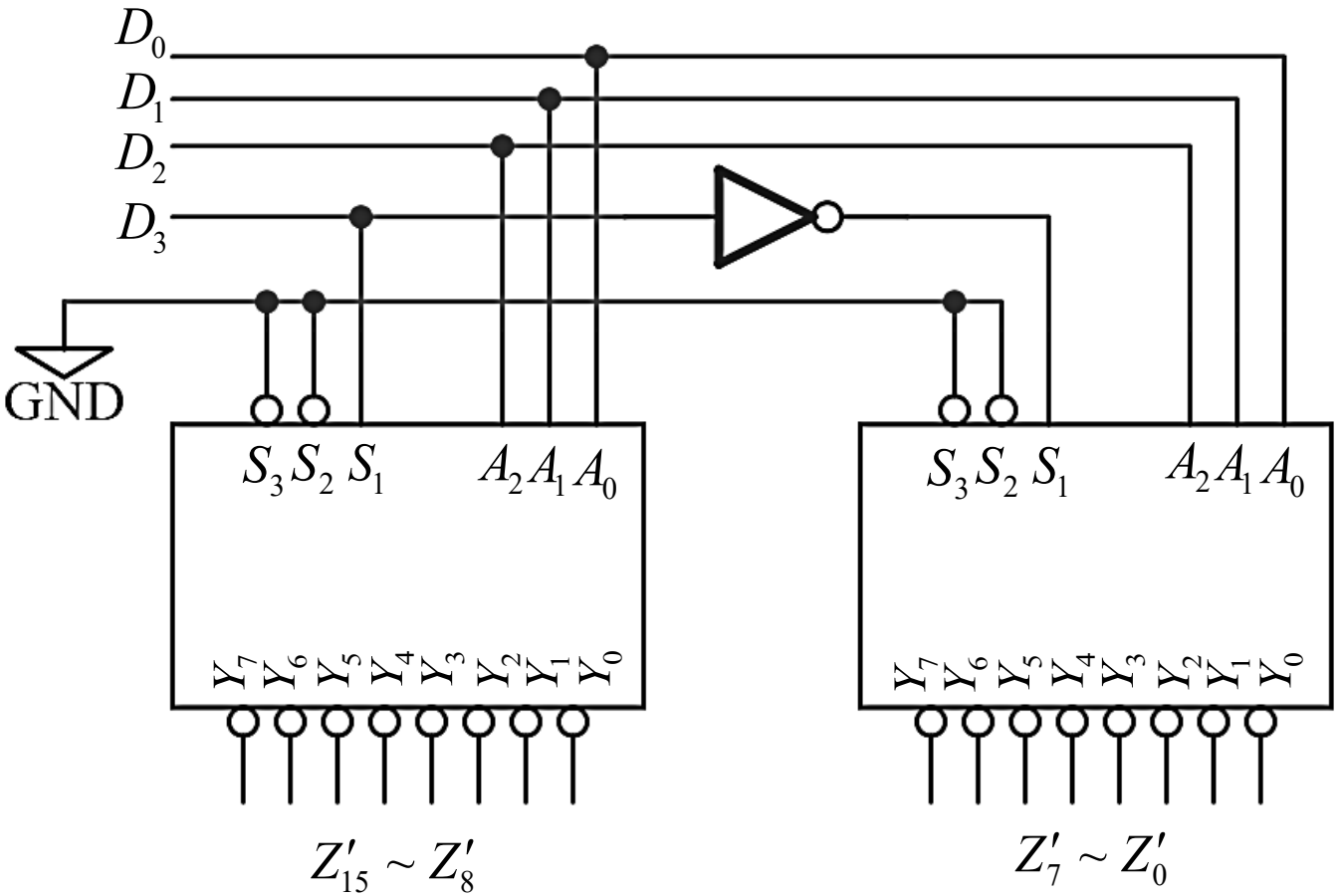
思路：

将 D_3 位作为两片的选通输入端控制信号，
当 D_3 = 1 时，则第 1 片工作，第 2 片不工作，此时即相当于对 1000~1111 8 个代码进行译码；
当 D_3 = 0 时，则第 1 片不工作，第 2 片工作，此时即相当于对 0000~0111 8 个代码进行译码；
由于 74HC138 不工作时输出全部被封锁在高电平，且低电平输出有效，因此对应 16 个低电平；

译码器

用附加端实现拓展实例
—— 将 3/8 译码器拓展为 4/16 译码器

思考：
如果规定不允许使用其他门电路器件呢？
（把 D_3 接到第二片的 S_2 和 S_3，
因为这两个端口低电平输入有效，
本身就是和 S_1 互补的，
第二片的 S_1 接电源电压即可）



变式：
如果想将该 4/16 译码器封装，
要求具有两个附加控制端，
如何引出接线？
（自行完成，
不再单独设页讲解）

译码器

基于译码器设计组合逻辑电路实例

例：利用 74HC138 设计一个多输出的组合逻辑电路，输出逻辑函数式为：

$$Z_1 = AC' + A'BC + AB'C$$
$$Z_2 = BC + A'B'C$$
$$Z_3 = A'B + AB'C$$
$$Z_4 = A'BC' + B'C' + ABC$$

思路：n/2^n 译码器的输出包含全部的最小项，把输出写成最小项和的形式即可；
需要注意的细节就是给定的译码器芯片是低电平输出有效还是高电平输出有效；
对于常见的 74HC138（以及 74LS138 ）、74LS154 （ 4/16 译码器）输出是低电平有效，即输出为各个最小项取反，因此应该用与非门去实现最后的输出目标；

译码器

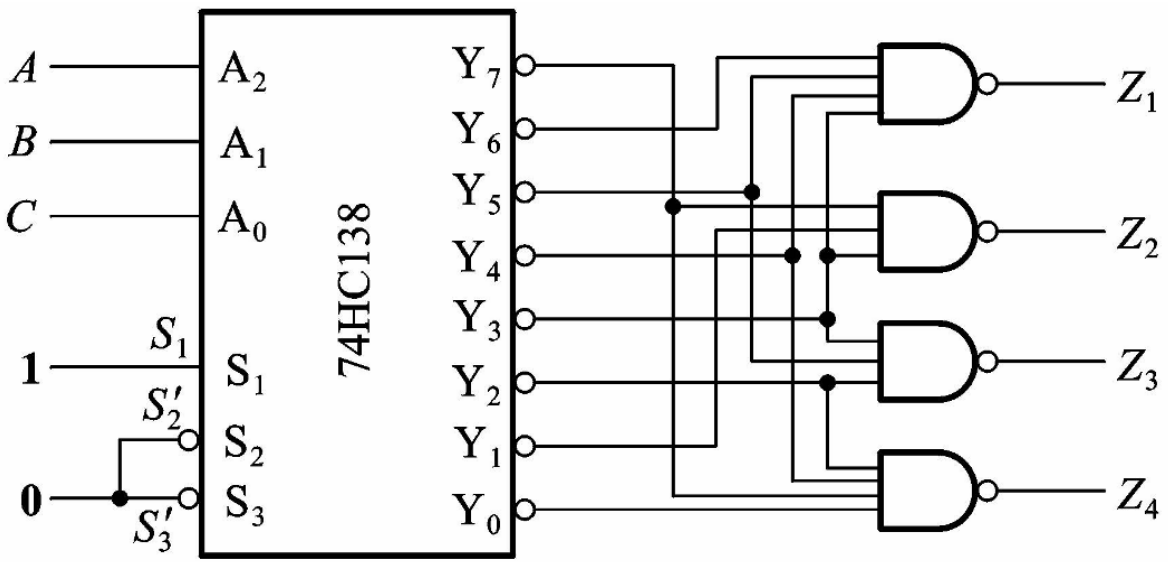
思考：
如果此题给定的译码器是规定高电平输出有效呢？
（用或门即可，具体题目具体分析）

基于译码器设计组合逻辑电路实例

接上页：

$$Z_1 = AC' + A'BC + AB'C$$
$$Z_2 = BC + A'B'C$$
$$Z_3 = A'B + AB'C$$
$$Z_4 = A'BC' + B'C' + ABC$$

$$Z_1 = AC' + A'BC + AB'C = \sum m(3,4,5,6)$$
$$Z_2 = BC + A'B'C = \sum m(1,3,7)$$
$$Z_3 = A'B + AB'C = \sum m(2,3,5)$$
$$Z_4 = A'BC' + B'C' + ABC = \sum m(0,2,4,7)$$



译码器

思考：
使用二—十进制译码器设计有什么限制?
(输出只含有十个最小项)

二—十进制译码器

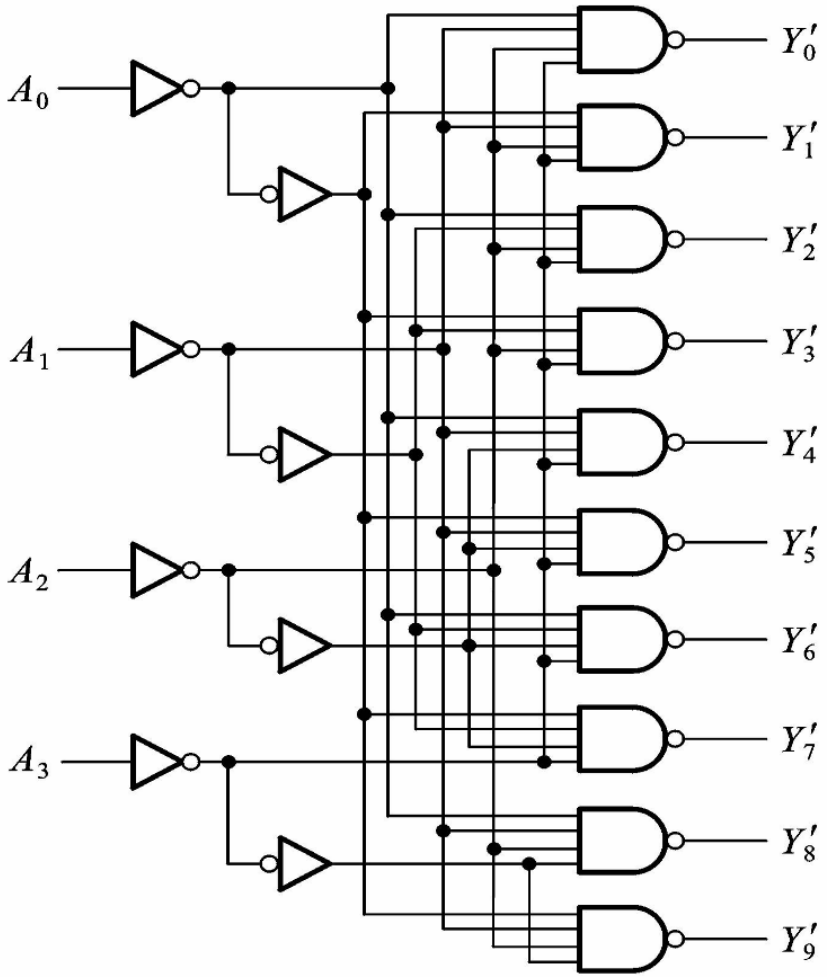
功能：输入为 4 位 BCD 码（8421码），输出 10 个高低电平信号；
(与二—十进制编码器相反)

二—十进制译码器设计的基本思路和功能表与普通的二进制译码器类似，需要注意的一个问题就是，由于 4 位代码共有 16 种状态，而 BCD 译码器只能接受 0 ~ 9 对应的 10 个 BCD 码，因此电路要能够拒收伪码，即拒绝翻译 1010 ~ 1111（A~F）这 6 个代码，此时没有有效输出电平；如果规定低电平输出有效，即输出全部被封锁在高电平；这就相当于在设计逻辑电路化简逻辑函数式时，将 6 个伪码对应的最小项视为无关项处理；

译码器

对于 二—十进制译码器
以及二—十进制优先编码器，
掌握它们的基本概念和输入输出的特点即可！

二—十进制译码器典型芯片 —— 74HC42



$$Y'_i = m'_i (i = 0,1,...9)$$

功能特点：

高电平有效输入，低电平有效输出；

能够拒收 1010 ~ 1111 ，输入伪码时输出全部封锁在高电平；

具体功能表参考教材即可；

译码器

显示译码器（了解）

显示译码器就是将译码器输出的高低电平用来驱动对应的发光二极管，将输入的数字、字母、汉字、字符等代码翻译并显示出来；常见的如教材上给出的 BCD—七段显示译码器就是将 BCD 代码译成 0 ~ 9 并显示出来；（此时的输出不再是某一个输出端口的高/低电平，而是与其发光二极管排列对应的另一套代码）

显示译码器具体使用时参考数据手册按照不同的输入代码对应的输出以及对应的图形接线即可，一般的显示译码器会带有测试端、灭零输入、灭灯输入/灭零输出控制端，具体功能特点应用时参考教材、查阅相关资料即可，这里不再展开；

译码器

译码器小结

- 译码器的作用是将输入的代码转换为一组高、低电平或者是其他代码输出，可以看作是编码的反过程；译码器主要分为二进制译码器、二—十进制译码器以及显示译码器；
- 译码器掌握的重点是利用二进制译码器进行组合逻辑电路设计，因为二进制译码器的特点是其 n 个输入变量的 2^n 个输出刚好对应了全部最小项（或最小项取反）；因此，将译码器的输出端通过门电路进一步拓展即可实现任意逻辑函数目标；即在解决问题时如果题目规定使用译码器来进行设计，此时逻辑函数式并不是化至最简，而是变换为最小项之和的形式（即最小项取反的与非）；另外，注意附加端的使用；注意区别低电平有效和高电平有效；
- 了解二—十进制译码器以及显示译码器的概念、原理和功能特点即可；
(可以把二—十进制译码器和二—十进制优先编码器联系记忆)

数据选择器

思考：

对于来自 2^n 个通道上的数据（即数据输入端有 2^n 个），其地址码需要多少位（即地址输入端需要多少个）？

（2^n 个数据输入对应 n 位地址代码 ）

数据选择器和数据分配器

- 数据选择器（Mux）：

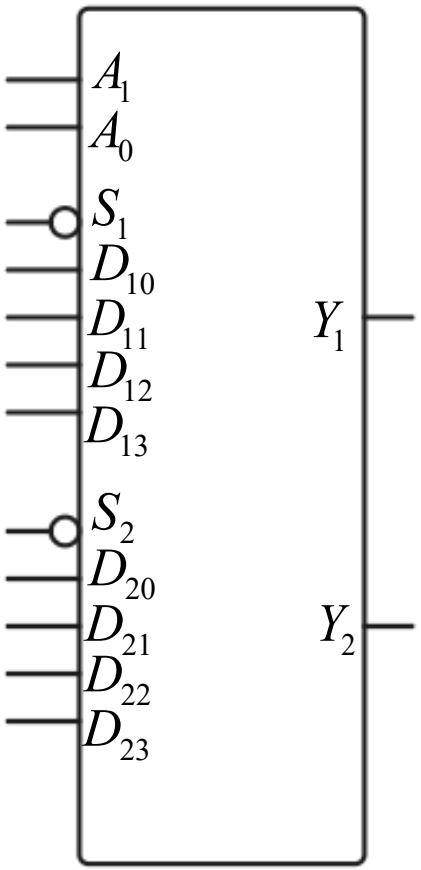
将来自多个通道的数据传送到唯一的公共数据通道上去，相当于一个单刀多掷开关，控制量就是地址代码，选择将给定的地址代码对应的通道的数据从输入端传输到输出端；

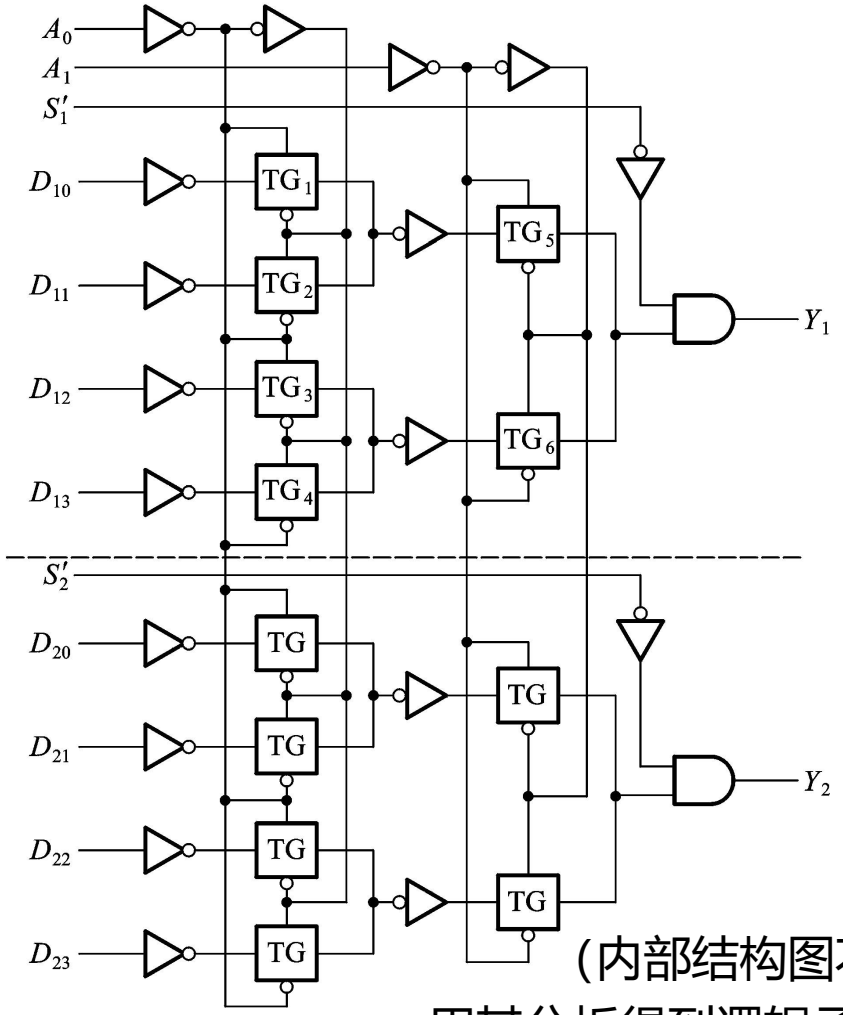
- 数据分配器（Demux）：

和数据选择器相对，将一个公共数据通道上的数据根据其地址代码，分配至该地址代码对应的通道；事实上，我们前面学习的译码器就可以实现数据分配的功能，例如对于 74HC138，如果把代码输入端 $A_2A_1A_0$ 作为地址码，S_1 作为输入端一个要被分配的数据（只不过此时的数据特指一位 0 / 1 ），在 $S_2' = 0$ 和 $S_3' = 0$ 时，就可以将 S_1 的数据根据地址码传输到地址码对应的输出端（以低电平的形式）；因此本章将不再单独介绍数据分配器，重点关注数据选择器及其应用；

数据选择器

数据选择器典型芯片 —— 74HC153（双 4 选 1 数据选择器）





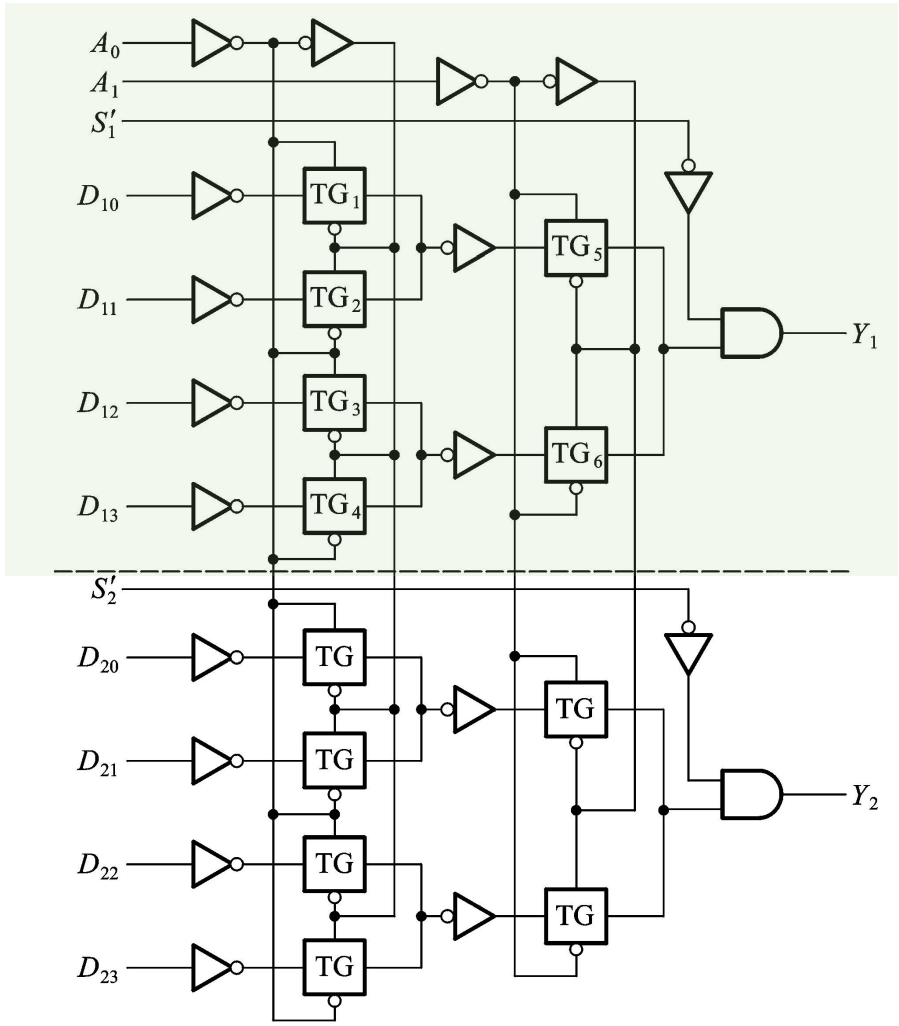
所谓的双 4 选 1 就是在一个芯片上有两个相同的 4 选 1 数据选择器，两个选择器的数据输入端是独立的，（ D 即 Data ），而两个选择器的地址输入端是公共的；根据原理，4 个数据输入需 2 位地址代码；

除此之外，两个选择器都带有一个选通信号端，为低电平有效输入；可以用来芯片功能拓展；

（内部结构图不用掌握，
用其分析得到逻辑函数式结论即可）

数据选择器

数据选择器的逻辑功能特点



分析其中的一个 4 选 1

S'	A_1	A_0	Y_1
1	X	X	0
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3

$$Y = [D_0(A_1'A_0') + D_1(A_1'A_0) + D_2(A_1A_0') + D_3(A_1A_0)] \cdot S$$

当 S′ = 1 时数据选择器不工作，输出封锁在低电平；

当 S′ = 0 即选通信号有效时：

$$Y = \sum D_i m_i$$

（m_i 是地址变量编号为 i 的最小项）

数据选择器

数据选择器的逻辑功能特点

根据前面的分析，具有 2 个地址位输入（即对应 4 个数据位输入）的 4 选 1 数据选择器，其输出与输入的逻辑函数式为（在选通信号有效输入的前提下）：

$$Y = D_0(A_1'A_0') + D_1(A_1'A_0) + D_2(A_1A_0') + D_3(A_1A_0) = \sum D_i m_i$$

因此，对于一个三变量的逻辑函数式，将其写成与或式之后，选择其中两个变量作为地址输入信号，令各个数据输入信号为第三个变量的适当形式（原变量、反变量、0 或 1），就可以在 4 选 1 的数据选择器的输出端产生三变量的组合逻辑函数；

同理，对于 2^n 选 1的数据选择器，即地址有 n 位，可以产生任何形式输入变量不超过 n + 1 的组合逻辑函数；例如 8 选 1 的数据选择器输出逻辑函数式为：

$$\begin{aligned} Y = {} & D_0(A_2'A_1'A_0') + D_1(A_2'A_1'A_0) + D_2(A_2'A_1A_0') + D_3(A_2'A_1A_0) \\ & + D_4(A_2A_1'A_0') + D_5(A_2A_1'A_0) + D_6(A_2A_1A_0') + D_7(A_2A_1A_0) = \sum D_i m_i \end{aligned}$$

数据选择器

用附加端实现拓展实例 —— 将 4 选 1 拓展为 8 选 1

要求将双 4 选 1 芯片 74HC153 接成一个 8 选 1 的数据选择器；

思路：

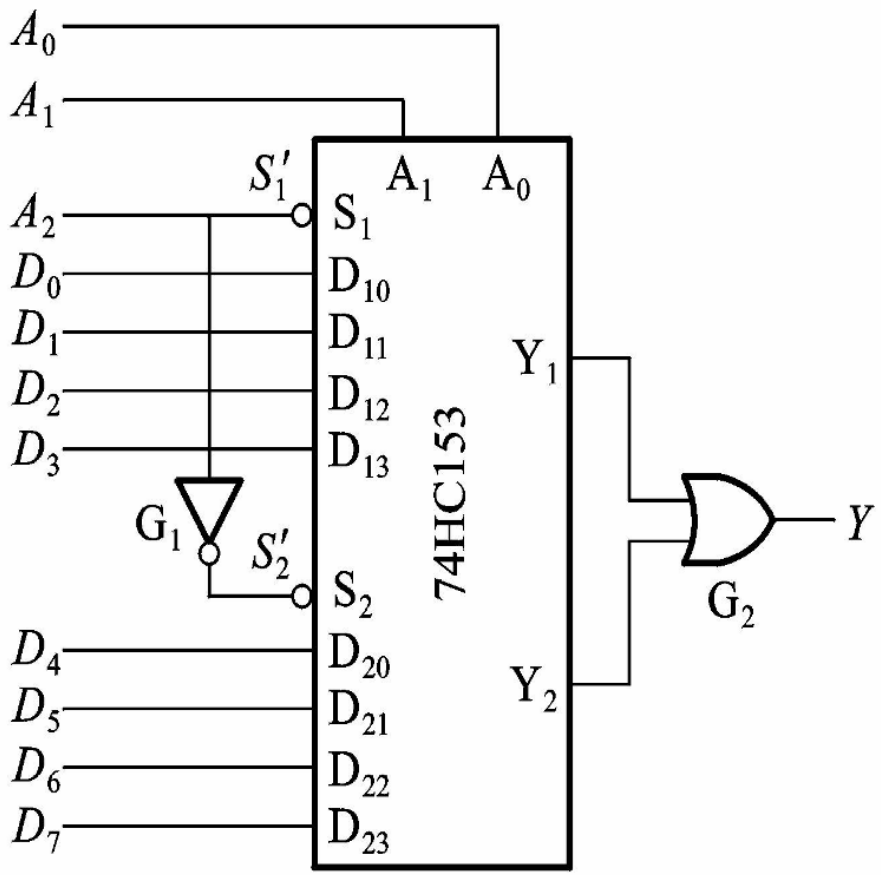
由于 4 选 1 只有两个地址输入端，因此必须使用芯片的附加端即选通输入端作为第三个地址位；此时该芯片两个选通输入端应该保持互补关系，即第一片选通时，对应地址代码为 000~011，此时输出 D_0 ~ D_3；第二片选通时，对应地址代码为 100~111，此时输出 D_4 ~ D_7；

由于最终的输出不是取自第一片就是取自第二片（每一片不工作时输出被封锁为低电平），因此为逻辑“或”的关系，即两片的输出通过一个或门即为最终的 8 选 1 的输出；

数据选择器

用附加端实现拓展实例 —— 将 4 选 1 拓展为 8 选 1

接上页，拓展接线图如下：



变式：

基于此拓展结果，

如果想将此 8 选 1 数据选择器封装，

再增加一个控制端（选通输入端），

如何接线?

（在 G_2 后面加一级与门，与 S 相与后输出，

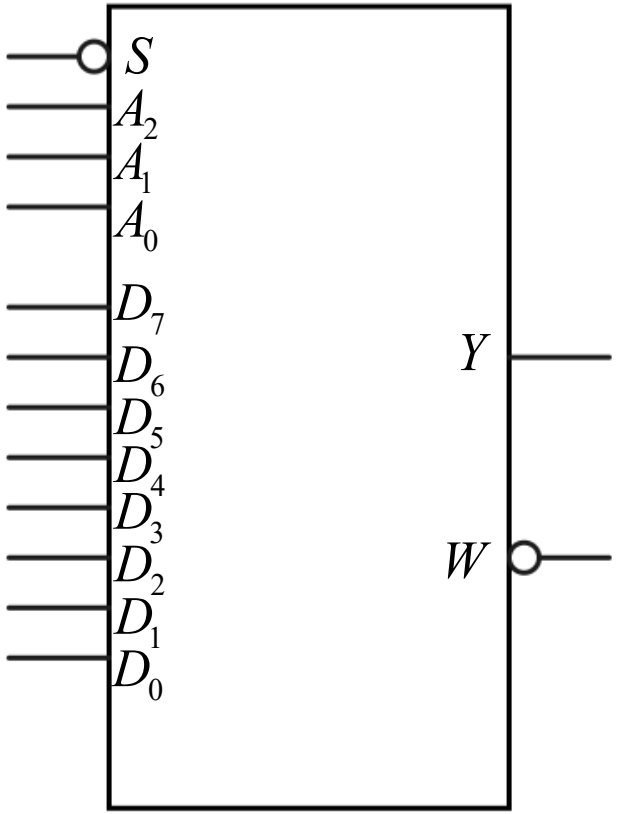
S = 1 则正常输出 Y，

S = 0 则输出为 0；

如果是低电平有效的控制端则给 S 再加一个反相器即可）

数据选择器

数据选择器典型芯片 —— 74HC151（8 选 1 数据选择器）



S：使能端，带小圆圈说明低电平输入有效；

A_2 ~ A_0：地址输入端；

D_7 ~ D_0：数据输入端；

Y：输出端，高电平输出有效；

W：事实上 W 就是 Y’ ；

数据选择器

基于数据选择器设计组合逻辑电路实例

试用 4 选 1 数据选择器 和 8 选 1 数据选择器分别产生三变量逻辑函数：

$$Z = A'B'C' + AC + A'BC$$

4 选 1 变换为：

$$Z = A'(B'C') + A(B'C) + ABC + A'BC = A'(B'C') + A(B'C) + 0\cdot(BC') + 1\cdot(BC)$$

或

$$Z = (A'B')\cdot C' + (A'B)\cdot C + (AB')\cdot C + (AB)\cdot C$$

8 选 1 变换为：

$$Z = 1\cdot(A'B'C') + 0\cdot(A'B'C) + 0\cdot(A'BC') + 1\cdot(A'BC) + 0\cdot(AB'C') + 1\cdot(AB'C) + 0\cdot(ABC') + 1\cdot(ABC)$$

接线图略，请自行完成；（注意每个位的排列顺序）

数据选择器

数据选择器小结

- 数据选择器的作用是将一组数据（ 2^n 个数据输入）根据其 n 位地址代码选择传输到输出端输出；

- 数据选择器掌握的重点是利用数据选择器进行组合逻辑电路设计，因为数据选择器的逻辑功能特点是输出变量与地址位输入、数据位输入有着特定形式的逻辑关系，因此，对于 2^n 选 1的数据选择器，即地址有 n 位，可以产生任何形式输入变量不超过 n + 1 的组合逻辑函数；当解决问题要求使用数据选择器设计组合逻辑电路时，此时逻辑函数式不是化成最简，而是要改写成数据选择器输出—输入的逻辑关系形式与之对应；注意附加端的使用；注意区别低电平有效和高电平有效；

加法器

加法器

根据第一章的基础知识，任何二进制数的加减乘除运算最终都可以转换为加法和移位操作；因此加法器也是数字电路中常用的典型电路模块；

加法器的逻辑功能特点其实并不需要特别强调，因为其真值表（逻辑关系）就是对应第一章学习的二进制数的加法规则；需要注意的是，在多位二进制数进行加法时，要考虑来自低位的进位以及向高位的进位，因此实际的加法器设计时要有进位输入端(CI)和进位输出端(CO)；

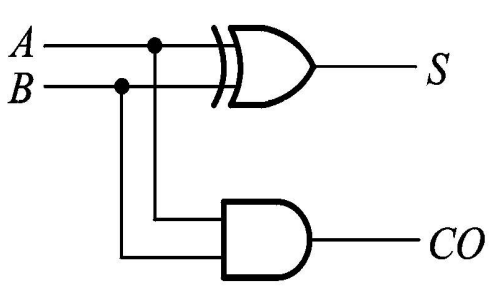
加法器

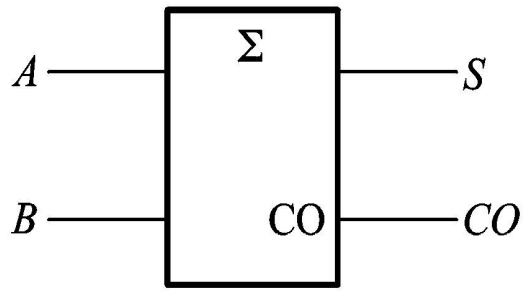
半加器

半加器即不考虑来自低位的进位，将两个 1 位二进制数相加的 1 位加法器；

输	入	输	出
A	B	S	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = A \oplus B$$
$$CO = AB$$





逻辑运算中的异或对应算术运算中的加法！

加法器

全加器

1 位全加器即考虑来自低位的进位，将两个 1 位二进制数相加的 1 位加法器；

输入			输出	
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = (A'B'CI' + AB'CI + A'BCI + ABCI')'$$
$$CO = (A'B' + B'CI' + A'CI')'$$

$$S = A \oplus B \oplus CI$$
$$CO = AB + (A + B)CI$$

逻辑运算中的异或对应算术运算中的加法！
（奇数个 1 相加，S 才为 1）

产生进位有两种情况：
两个加数都为 1，AB = 1，无论进位；
至少有一个加数为 1 且进位为 1，(A+B)CI = 1

加法器

思考：
串行进位加法器的设计原理
对应的实质上是我们第二章学习的哪个基本定理？
（代入定理）
（在电路上应用代入定理即级联，牺牲“时间”换取“空间”）

串行进位加法器

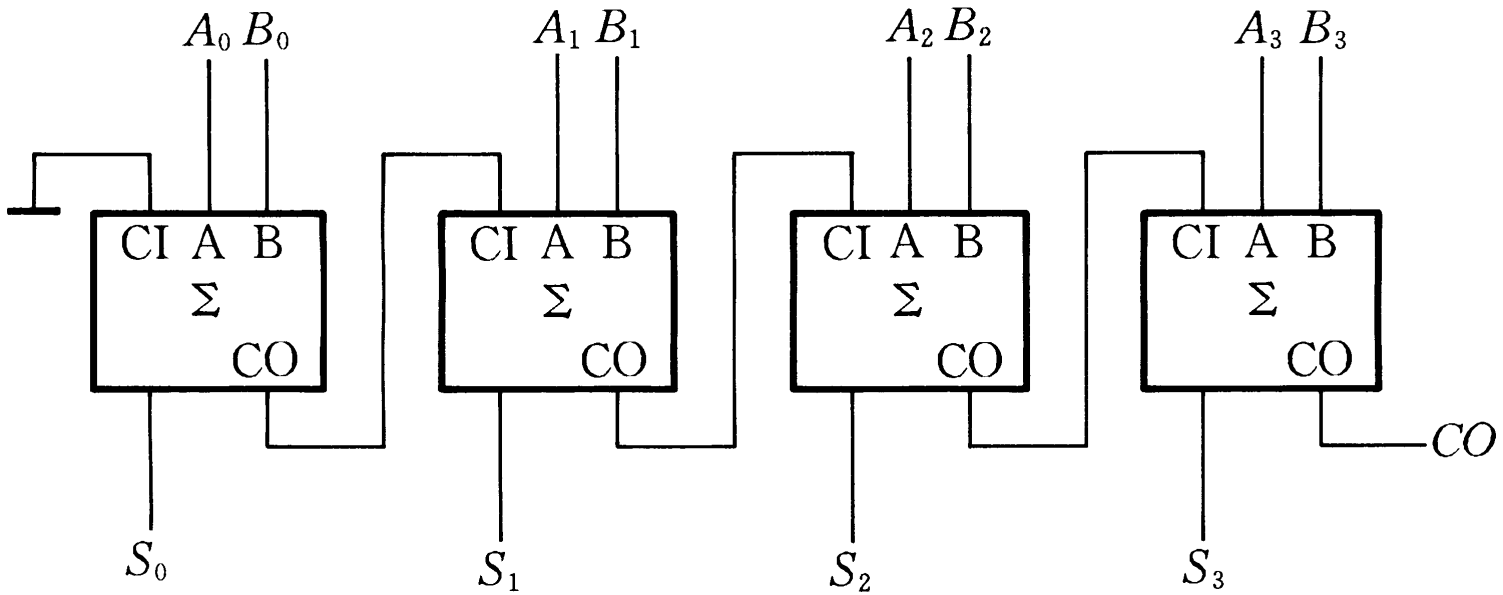
在进行多位数运算时，必须使用全加器；最简单的思路就是将每一位全加器的进位输出端连接至下一位全加器的进位输入端，即可构成一个多位加法器；按照此原理构成的加法器称为串行进位加法器，或称为行波进位加法器；所谓“串行”就是各位的运算是逐个进行的，每一位的相加结果必须等到低一位的进位产生后才能建立；

因此，串行进位加法器虽然结构简单，但是代价是牺牲了运算速度；例如对于用 n 个 1 位全加器搭建而成的 n 位串行进位加法器，其传输延迟时间为 n 个全加器的传输延迟时间的和，而且运算结果的每一位并不是同时产生；

加法器

串行进位加法器示例

4 位串行进位加法器（用 1 位全加器实现）：



$$(CI)_i = (CO)_{i-1}$$

$$S_i = A_i \oplus B_i \oplus (CI)_i$$

$$(CO)_i = A_i B_i + (A_i + B_i)(CI)_i$$

加法器

超前进位加法器

如果直接根据逻辑运算关系确定每一位的进位输入信号和进位输出信号，就能直接计算每一位的运算结果，而不用等待低位运算完毕再传递，这种加法器称为超前进位加法器；

超前进位加法器的优点是运算快速（从输入到输出的关键路径门的个数少），但是对应来讲，电路的结构也变得更复杂；当运算位数继续增加时，将很难进行设计；（因此通常更多位数的加法器可以通过较少位数的超前进位加法器按照串行进位的方式连接实现）

超前进位加法器的实现原理其实就是基于前面 1 位全加器给出的逻辑式，不断地迭代即可：

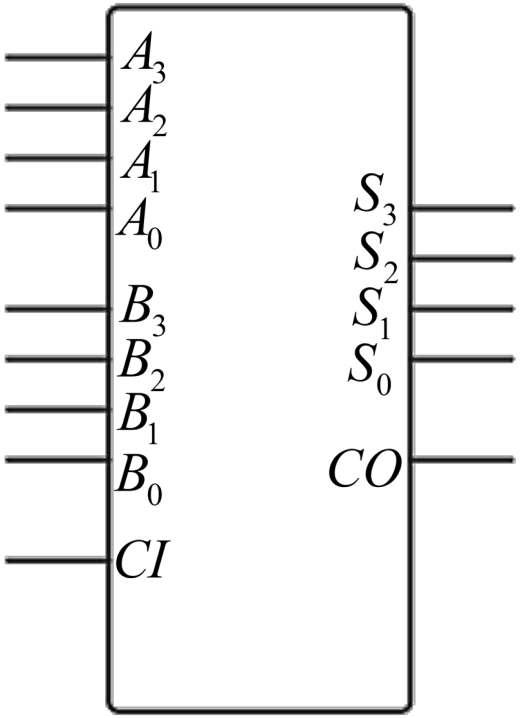
$$S = A \oplus B \oplus CI$$

$$CO = AB + (A + B)CI$$

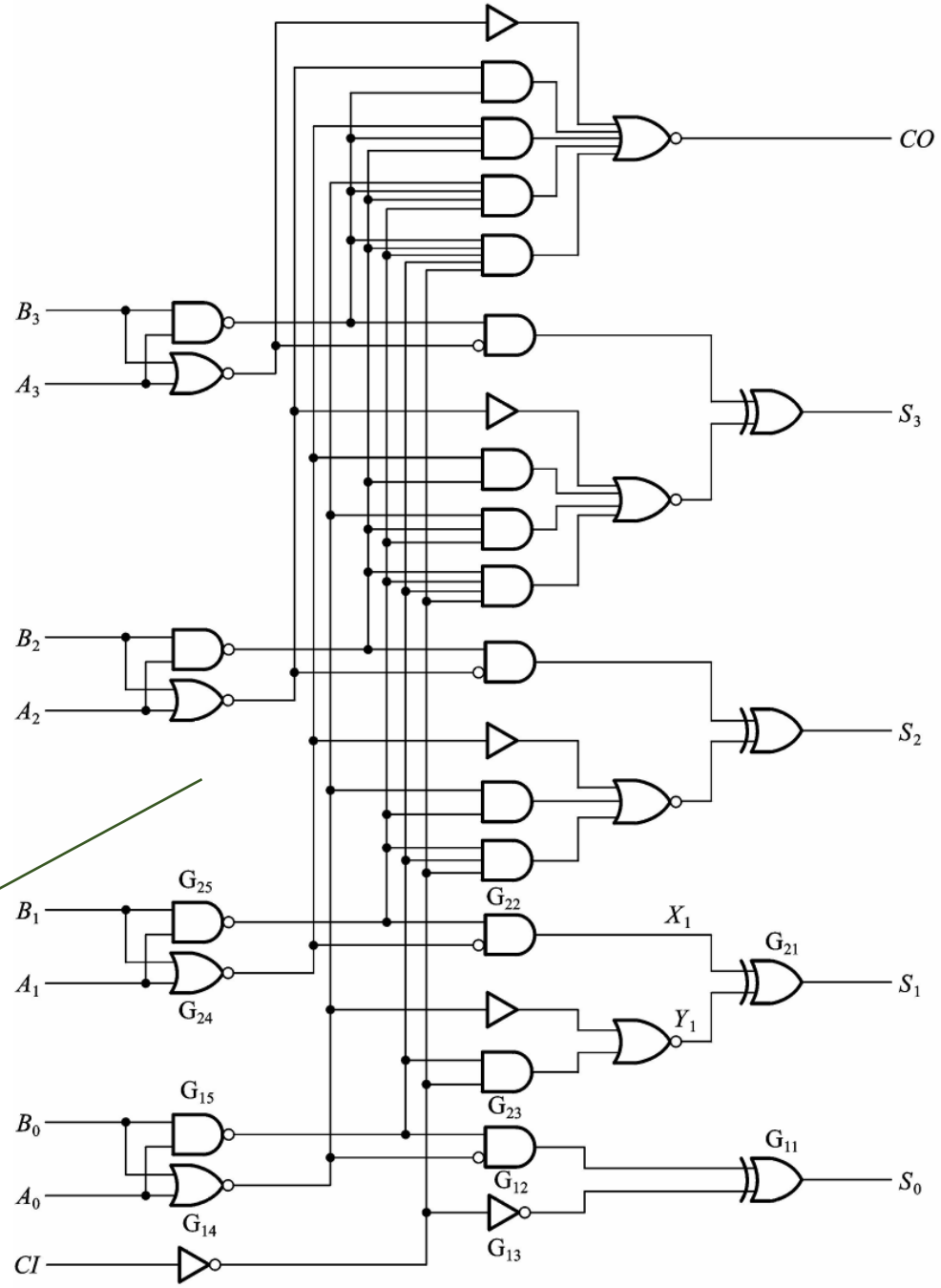
加法器

超前进位加法器示例

4 位超前进位加法器芯片 74LS283：
（芯片内部电路结构图不用掌握，
理解其实现的原理即可）



可以看到，关键路径
（最长的路径）
即从输入端到每个位输出端
经过 3 级门电路；



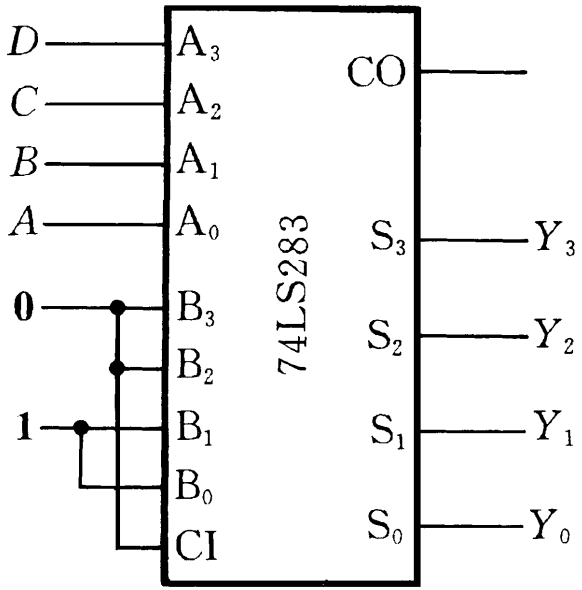
加法器

基于加法器设计组合逻辑电路实例

试用 4 位超前进位加法器 74LS283 实现 BCD 码到余 3 码的转换；

基本思路：BCD 码 → 余 3 码，加 3 即加 0011 即可；

输入				输出			
D	C	B	A	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0



（这个例题属于最简单的，
其他请参考习题课
以及
自行补充练习！）

加法器

加法器小结

- 掌握 1 位全加的原理，算术运算与逻辑运算的对应关系；

- 理解串行进位和超前进位的概念，两者实现原理和性能的比较；了解数字电路中“时间”与“空间”两个内在矛盾的性能指标；

- 因为加法器并不具有如译码器、数据选择器一样通用的输入—输出逻辑函数形式，因此使用加法器来设计组合逻辑电路一般是对于一些特定的问题，但仍然需要掌握；加法器的逻辑功能特点无需总结成逻辑函数式的形式，直接对应第一章的二进制加法运算规则即可；（例如像 4 位加法器九个输入变量，五个输出变量，列出真值表并不现实）比较常见的如代码转换电路（ BCD码—余 3 码）、减法运算、乘法运算、加/减运算、其他进制算术运算（例如二—十进制加法器）等；题目难度波动范围较大，因此通过多练习总结经验；

数值比较器

思考：
如果是我们自己设计数值比较器，
三种输出状态最少可以设置几个输出端？
（2 个即可，2 位二进制代码最多可以表示 4 种状态）
（这里用 3 个输出端即无需译码，输出的是独立高低电平）

数值比较器

数值比较器的功能是实现两个数值大小的比较；

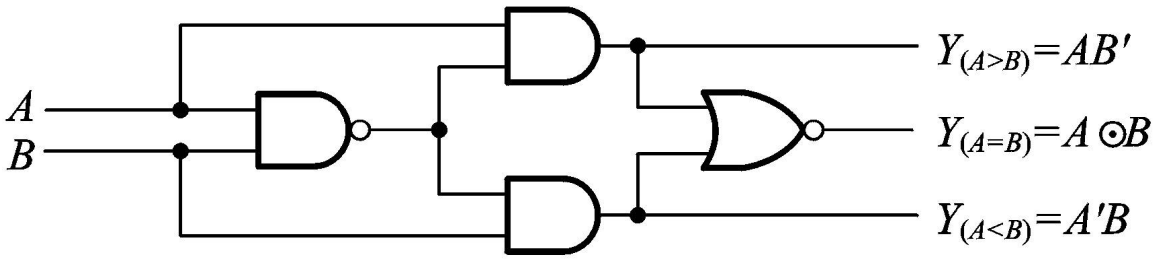
学习数值比较器和加法器类似，不用总结成逻辑函数式或者真值表的形式，按照两个数值比较的规则即可；

两个数值进行比较，结果有三种可能，即 A > B，A = B，A < B；因此数值比较器的输出可以选择三个输出端口，为独立的高低电平；

数值比较器

1 位数值比较器

首先讨论最简单最基本的 1 位二进制数的比较：



① A > B，即 A = 1 ， B = 0 ， 此时 $Y_{(A>B)}$ = 1；

② A < B，即 A = 0 ， B = 1 ， 此时 $Y_{(A<B)}$ = 1；

③ A = B，即 A = B =0 或 A = B = 1 ， 此时 $Y_{(A=B)}$ = 1；

$Y_{(A>B)}$、 $Y_{(A<B)}$、 $Y_{(A=B)}$ 为独立且互斥的输出变量，任一时刻只可能有一个为 1 （高电平）；

数值比较器

多位数值比较器

多位数大小比较即从高到低逐位比较即可，只有在高位相等时才需要比较低位；
因此可以很自然地得到下面的逻辑函数式（以 4 位二进制数比较为例）：
（逻辑函数式不用掌握，理解即可）

比较$A_3A_2A_1A_0$和$B_3B_2B_1B_0$

$$Y_{(A<B)} = A_3'B_3 + (A_3 \oplus B_3)'A_2'B_2 + (A_3 \oplus B_3)'(A_2 \oplus B_2)'A_1'B_1$$
$$+ (A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)'A_0'B_0$$

$$Y_{(A=B)} = (A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)'(A_0 \oplus B_0)'$$

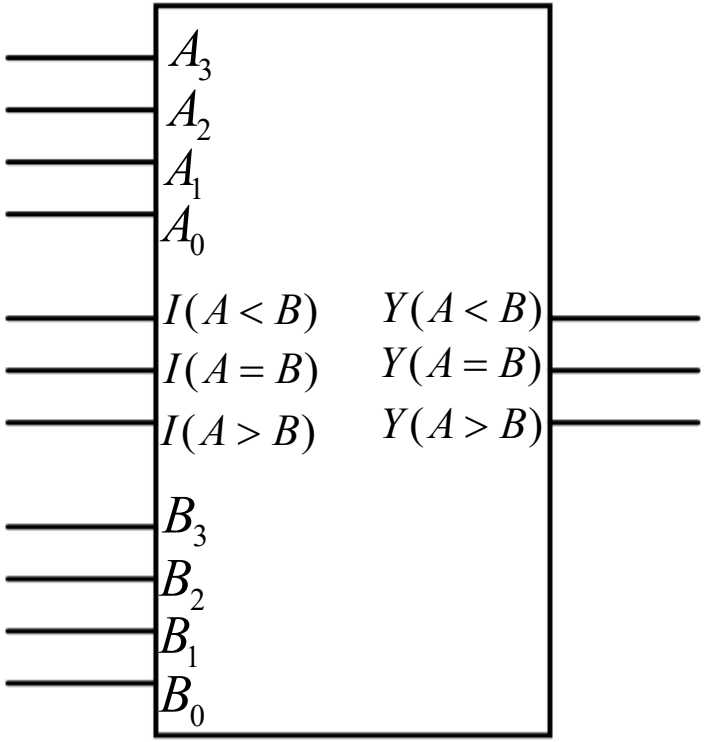
$$Y_{(A>B)} = (Y_{(A<B)} + Y_{(A=B)})'$$

$Y_{(A>B)}$、$Y_{(A<B)}$、$Y_{(A=B)}$ 为独立且互斥的输出变量，任一时刻只可能有一个为高电平；三者的和为 1；

数值比较器

多位数值比较器典型芯片 —— 74LS85

74LS85 为一个典型的 4 位数值比较器：



A_3 ~ A_0 、 B_3 ~ B_0 ： 参与比较的两个 4 位二进制数输入端；

$I_{(A>B)}$ 、 $I_{(A=B)}$ 、 $I_{(A<B)}$ 为来自低位的比较结果，即当 $A_3A_2A_1A_0 = B_3B_2B_1B_0$ 时，需要根据 $I_{(A>B)}$ 、 $I_{(A=B)}$ 、 $I_{(A<B)}$ 的状态来决定最终的输出；（这三个附加输入变量互斥，即只能有一个输入为 1，其实对应的就是低位的数值比较结果输出）

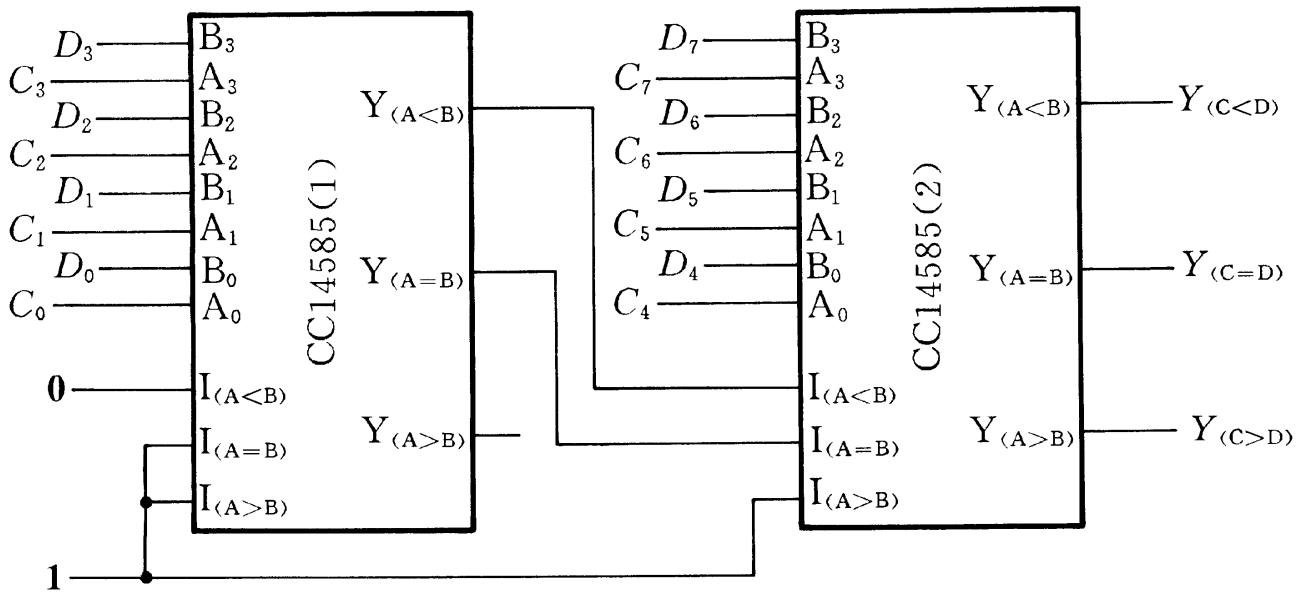
$Y_{(A>B)}$ 、 $Y_{(A=B)}$ 、 $Y_{(A<B)}$ 为结果位，任一时刻只有一个为 1；

数值比较器

用附加端实现拓展实例 —— 将 4 位数值比较器拓展为 8 位数值比较器

要求用两片 CC14585（功能同 74LS85） 组成一个 8 位数值比较器；

思路：将低 4 位的比较结果的输出端连接至高 4 位的附加输入端即可；
低 4 位的附加输入端将 $I_{(A=B)}$ = 1，另两个为 0 即可（即没有低位）；



数值比较器

数值比较器小结

- 数值比较器并不具有如译码器、数据选择器一样通用的输入—输出逻辑函数形式，因此使用数值比较器来设计组合逻辑电路一般是对于一些特定的问题，但仍然需要掌握；

- 数值比较器的逻辑功能特点无需总结成逻辑函数式的形式，直接根据两数比较的原理就能分析其逻辑功能；（可以联想结合编程课中的 if – else if 多级嵌套关系记忆）；注意附加端的使用；

常用的组合逻辑电路模块 —— 小结

编码器 (重点掌握优先编码器)	将 2^n 个独立的高低电平输入量编为 n 位代码输出；	74HC148 (8/3)	对于优先编码器， 输入侧存在优先级
译码器 (重点掌握二进制译码器)	将输入的 n 位代码译为 2^n 个 独立的高低电平输出；	74HC138 (3/8)	输出覆盖全部最小项 通过或门（与非门） 可以实现任意逻辑函数
数据选择器	根据 n 位地址代码，选择 2^n 个数据 的其中一个传输至输出端输出；	74HC153 (4选1) 74HC151 (8选1)	2^n 选 1，即 n 位地址代码 最多可实现 n+1 个变量的 逻辑函数
加法器	实现多位二进制数的加法，考虑来自 低位的进位输入和向高位的进位输出；	74LS283 (4位)	输入输出遵循二进制算术 运算的规则
数值比较器	实现多位二进制数的比较，考虑来自 低位的比较结果；	74LS85 (4位)	输入输出遵循数值比较的 规则

常用的组合逻辑电路模块 —— 小结

注意：
对于这些附加控制端，
不能只按照教材上给出的表格去记，
也不用去专门记每个芯片的端口带不带小圆圈，
重要的是理解这些附加控制端的概念和特点，
实际解题结合给出的具体功能表，
注意区分高电平有效还是低电平有效！！！

中规模集成器件中常见的附加端

- 选通输入端 —— 片选器件，只有当选通输入端输入有效信号（$S = 1$ 或 $S' = 0$），器件才允许工作；否则器件的输出被全部封锁在高电平/低电平（具体封锁在高电平还是低电平结合实际情况，例如 74HC138 输出低电平有效，则不工作时输出全部被封锁在高电平；而 74HC153 输出高电平有效，则不工作时输出全部为低电平；）

- 优先编码器的两个拓展输出端 Y_S 和 Y_{EX} 的特点；

- 多位加法器的进位输入端 CI 和进位输出端 CO；

- 数值比较器的低位比较结果输入端 $I_{(A>B)}$、$I_{(A=B)}$、$I_{(A<B)}$；

常用的组合逻辑电路模块 —— 小结

不需要去背！
结合具体题目来感受和理解！

组合逻辑电路的设计方法（要求使用中规模器件实现）

Step 1：逻辑抽象 —— 分析因果关系，确定输入/输出变量、定义逻辑状态的含意（赋值）、列出真值表；

Step 2：将真值表转换为逻辑函数式；

Step 3：选定器件类型；（根据题目要求使用的中规模器件）

Step 4：根据所选器件，将逻辑式变换为该器件逻辑功能对应的形式；
如果选用的中规模器件位数不够或产生的逻辑函数式只是目标输出的一部分，则需要拓展（几片级联使用、附加其他的门电路等）；

Step 5：画出逻辑电路连接图；

组合逻辑电路中的竞争—冒险

竞争—冒险的概念

我们前面对组合逻辑电路的分析和设计都是基于静态（稳态）即输入、输出为稳定的逻辑高低电平，而根据门电路一章的基础知识，数字电路中的开关器件内部存在寄生电容，因此门电路以及由门电路组成的集成器件会存在传输上的延迟，因此有必要去研究当输入信号的逻辑电平发生跃变时的工作情况；

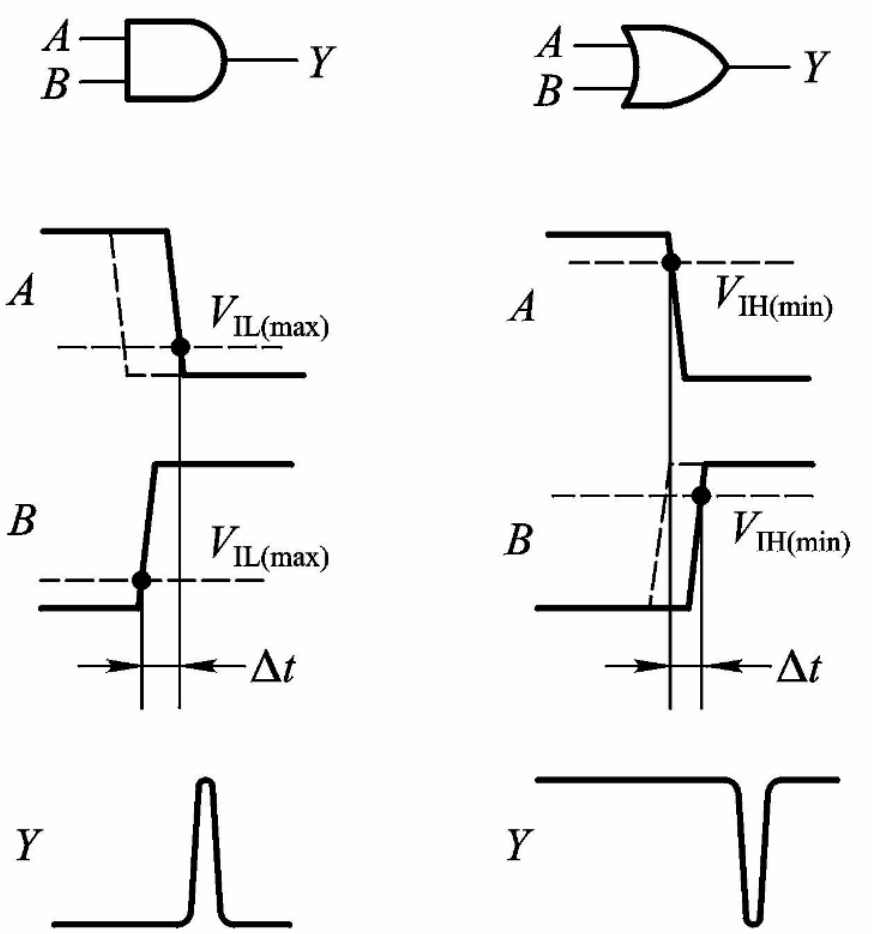
在组合逻辑电路中，考虑门电路的两个输入信号同时向相反的逻辑电平变化，即一个从 1 变为 0，另一个从 0 变为 1，将这种情况称为两个输入信号之间存在“竞争”；所谓“竞争”即两个输入信号到达稳态的次序先后不确定，而因为这种不确定性，就可能会导致输出端产生不期望的尖峰脉冲（噪声），因此称为“竞争—冒险”；

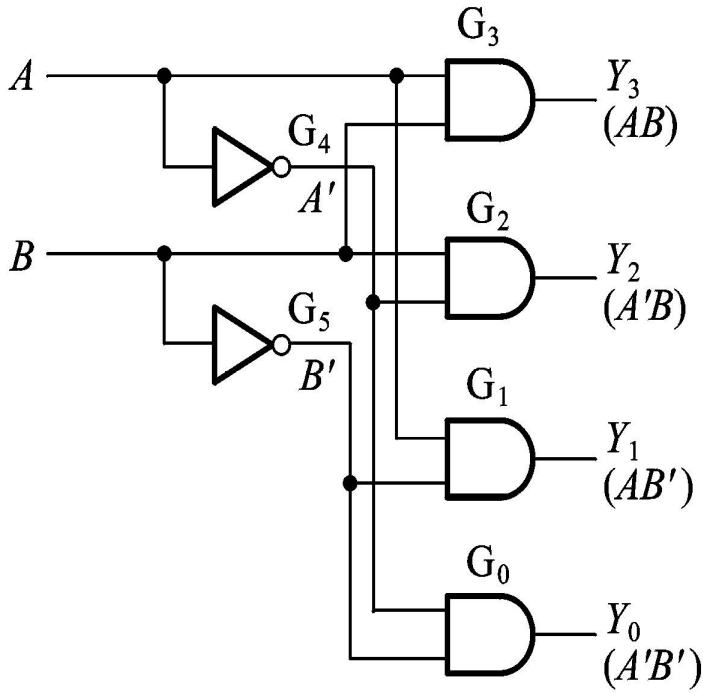
这种尖峰脉冲可能会导致后级负载的误工作（例如下一章学习的触发器）；

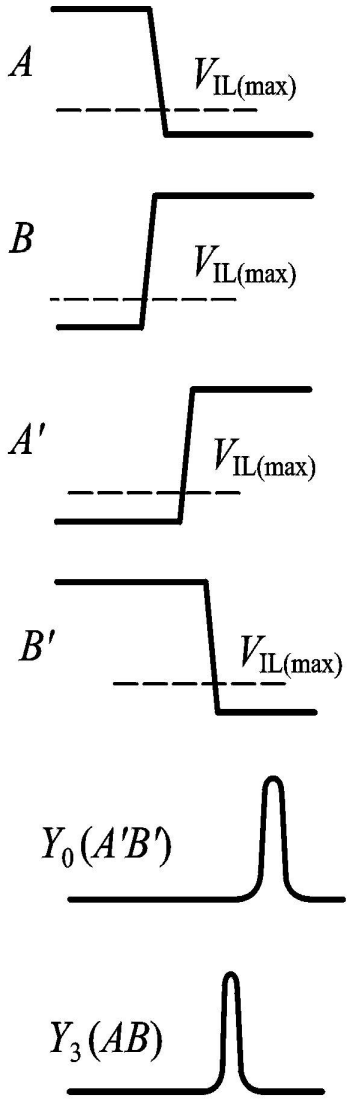
组合逻辑电路中的竞争—冒险

能够结合例子理解
“竞争—冒险”的概念即可！

竞争—冒险的示例







组合逻辑电路中的竞争—冒险

检查竞争—冒险的方法（了解）

- 根据逻辑函数式；
- 用计算机辅助分析；
- 实验；

消除竞争—冒险的方法（了解）

- 接入滤波电容；（利用电容电压不能突变，削弱尖峰幅度，但同时也破坏了原有波形）
- 引入选通脉冲；
- 修改逻辑设计；

组合逻辑电路——小结

- 组合逻辑电路的功能特点和结构特点；
- 组合逻辑电路的分析方法和设计方法；
- 常用的几种组合逻辑电路模块（中规模集成器件）；
- 编码、译码、数据选择、数据分配等概念的理解；
- 利用中规模集成器件设计组合逻辑电路；
- 竞争 — 冒险的概念；