中国科学技术大学

数字逻辑电路习题课讲义

学　　院：信息科学技术学院

班　　级：2020 秋数字逻辑电路 06 班

授课教师：胡新伟老师

课程助教：高源

目录

重要问题整理

1.1 无关项的概念及其在化简逻辑函数中的应用

1.1.1 无关项的概念

在处理一些逻辑问题时，会遇到一些特殊情况. 具体表现为，在逻辑式中存在一些最小项，是否将其写入逻辑函数式，从逻辑功能上讲无关紧要，即可以写入也可以删除. 我们称之为无关项.

在化简逻辑函数式的时候，适当使用一些最小项可能可以辅助化简，从而得到更简结果. 所以我们有必要对其进行研究.

无关项按照其对应的逻辑问题类型可以分为两类：

- 约束项
- 任意项

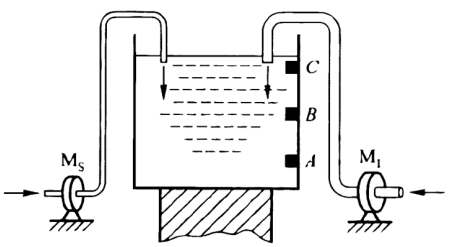
下面我们分别进行介绍.

1.1.2 约束项

约束项是指取值恒等于 **0** 的最小项.

约束项的产生是由于逻辑问题的实际物理意义带来的限制. 在实际问题中，一些最小项对应的状态不会出现，所以其取值恒等于 **0**. 下面以一个例子说明这一问题.

如图所示，水箱由大小两台水泵 M_L 和 M_S 供水. 水箱中设置了 3 个水位检测元件 A、B、C. 水位低于检测元件时，检测元件给出低电平；水位高于检测元件时. 检测元件给出高电平.



考虑两个水泵的控制信号由水位情况决定，即由 A、B、C 的取值决定. 所以在抽象得到的逻辑问题中，用来表示两个水泵的控制信号 Y_L 和 Y_S 是输入变量 A、B、C 的逻辑函数.

从实际问题出发，我们可以认识到，由于逻辑变量 A、B、C 对应的物理意义，其取值关系存在一些约束. 例如，由于水位高于 C 的时候，一定也高于 A、B，所以当逻辑变量 C 取值为 **1** 的时候，逻辑变量 A、B 的取值也一定为 **1**，即不会出现 A、B、C 取值 001、011、101 的情况，相应的最小项取值一定为 **0**.

当逻辑函数式中加入一个恒为 **0** 的项作或运算，对于逻辑函数取值没有任何影响. 所以在化简的时候可以直接使用这些项.

1.1.3 任意项

任意项是指取值为 **0** 或为 **1** 皆可的最小项.

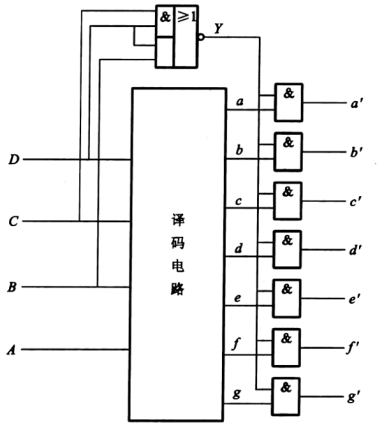
约束项的产生是由于逻辑设计时候一些逻辑函数的屏蔽作用，此时无论这些最小项怎样取值，不会影响最终结果，也就不会影响电路功能. 但是要注意的一点是，其虽然不影响最终结果，但是对于逻辑函数式本身的取值是可能有影响的，只不过当前的逻辑函数式在作为后面电路的输入的时候被屏蔽了所以不产生影响. 这一点是约束项和任意项的重要区别，即约束项取值恒为 **0**，加入约束项不影响逻辑函数式的取值，而任意项若取值为 **1** 时，加入任意项会影响逻辑函数式的取值.

下面以一个例子说明这一问题.

考虑设计一个拒绝伪码的七段显示译码器. 所谓拒绝伪码，指在输入为 $1010 \sim 1111$ 时输出无任何字形显示，即 $a \sim g$ 输出全部为 **0**.

可以列出真值表 (见下页).

在一种设计方案中，引入了输出缓冲器，其控制信号为 $Y = (DC + DB)'$. 此时，当 $DCBA = 1010 \sim 1111$ 时, 无论 $a \sim g$ 是 **1** 还是 **0**，$\tilde{a} \sim \tilde{g}$ 恒为 **0**，即符合设计要求. 所以此时，$DCBA = 1010 \sim 1111$ 对应的最小项是否出现并不影响最终结果，即为任意项. 当然我们也可以看到，是否加入这些最小项对于逻辑函数式 $a \sim g$ 是有影响的，只不过并不影响最终电路输出 $\tilde{a} \sim \tilde{g}$.



真值表

数字	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	0	0	0	0	0	0	0
11	1	0	1	1	0	0	0	0	0	0	0
12	1	1	0	0	0	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0	0	0
14	1	1	1	0	0	0	0	0	0	0	0
15	1	1	1	1	0	0	0	0	0	0	0

1.1.4 无关项在化简中的应用

在我们的课程学习中，逻辑函数式化简的常用方法包括

- 公式法
- 卡诺图法

二者各有适用条件. 一般对于变量数不超过 4 的复杂 (如果可以直接使用公式的简单问题可以选择公式法) 逻辑函数式化简，往往首选卡诺图法. 对于这类含有无关项的逻辑函数式化简，我们也是首选卡诺图法.

卡诺图法化简的原理是，利用几何相邻性反映逻辑相邻性，通过合并最小项实现化简. 当存在无关项的时候，我们可以适当选择无关项加入逻辑函数式，使得可以更好地利用逻辑相邻性实现化简.

在卡诺图中，我们把无关项记为 $\times$，表示其既可以作为 **0**(不加入逻辑函数式)，也可以作为 **1**(加入逻辑函数式). 是否加入，则需要根据实际情况，看怎样能够实现最简.

特别要注意的就是，无关项并不是一定要作为 **1**(加入逻辑函数式). 有时候把所有无关项都加入逻辑函数式得到的不一定是最简结果.

1.2 逻辑函数不同表达形式之间的转换

1.2.1 真值表转换为逻辑函数式

在组合逻辑电路设计中，往往是首先根据自然语言描述的设计问题，转化为和自然语言最接近的表达形式——真值表，然后将真值表转换为逻辑函数式. 具体操作过程为

- 首先从真值表中找出所有使函数值等于 **1** 的那些输入变量取值组合
- 每一组使输出为 **1** 的输入变量取值下，必然有一个最小项的值等于 **1**. 取值为 **1** 的变量在这个最小项中写为原变量，取值为 **1** 的变量在这个最小项中写为反变量
- 将所有的这些最小项相加，就得到了所求的逻辑函数式

1.2.2 逻辑函数式转换为真值表

在组合逻辑电路分析中，往往是首先根据逻辑电路图得到逻辑函数式. 一般情况下，仅仅从逻辑函数式出发并不能很直观地观察出其要实现的功能，所以我们常常要把逻辑函数式转换为真值表，进而可以观察到，在不同输入的情况下输出的取值情况，能够帮助我们分析其实现的功能. 具体操作过程为

- 按照一定顺序列出输入的所有可能取值 (习惯上按照顺序来列表，一种方法是，将变量依次取值构成的二进制串看成二进制数，按照数值从最小到最大的顺序来列表)
- 将所有的输入变量取值组合逐一代入逻辑式，算出输出的函数值，填入相应的位置即可

1.2.3 逻辑函数式转换为卡诺图

在逻辑函数化简时，往往会需要转换为卡诺图. 具体操作过程为

- 将逻辑函数式展开为最小项之和的形式
- 出最小项的卡诺图, 在函数式中包含的最小项对应的位置上填入 **1**，其余位置上填入 **0**
- 如果函数式中包含无关项, 则在相应位置上填入“×”表示填入 **0** 或 **1** 均可

1.2.4 波形图转换为真值表

在实际测试电路时，往往得到的是电压波形图. 而为了更方便分析电路，往往需要转换为真值表. 具体操作过程为

- 在周期性重复的波形图中, 将每个时间段内输入变量和输出变量的取值对应列表, 即可得到函数的真值表
- 若波形图中有些输入变量状态组合始终没有出现, 则这些输入变量组合下等于 **1** 的最小项为函数的约束项

1.3 逻辑函数式常见形式变换

在讨论组合逻辑电路设计的时候，我们了解到，在得到了逻辑函数式后，要根据实际实现电路的器件使用情况来对逻辑函数作适当的变换，包括化简和逻辑函数形式变换.

在逻辑函数形式的变换过程中，最重要的是要保证等价性. 根据我们在逻辑代数基础部分所学，可以想到的等价变换方法包括

- 两次求反 (还原律)
- 反演定理 (对于有取反运算的可以直接应用反演定理，如果没有直接的取反运算可以两次求反)

1.3.1 与或式 $\Rightarrow$ 与非与非式

具体方法是

- 对原始与或形式的逻辑函数式作两次求反的等价变换
- 保留最外层取反，内层取反运算使用反演定理

1.3.2 与或式 $\Rightarrow$ 与或非式

有两种方法

- 与或式两次求反并对内层取反使用反演定理 (内层的运算比转换为与非与非式更彻底，直接得到与或形式)
- 对于逻辑函数式中不包含的最小项求和再取反 (对应在卡诺图上圈 **0** 取反)

1.3.3 与或式 ⇒ 或与式

有两种方法

- 与或式转换为与或非式，然后对最外层的取反运算使用反演定理进行运算
- 反复使用公式 $A + BC = (A + B)(A + C)$

1.3.4 与或式 ⇒ 或非或非式

具体方法是

- 与或式转换为或与式
- 两次求反，使用反演定理

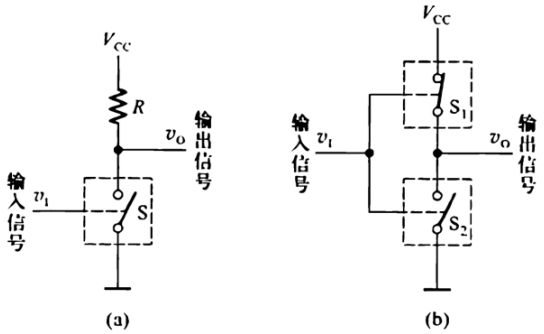
1.4 怎样学习第三章

第三章主要研究门电路的结构及其工作原理，并基于此来讨论一些功能和性能方面的问题. 第三章的学习难度较大，在学习第三章的时候，建议按照理解基本电路模型及其演进过程来学习.

1.4.1 基本电路——互补开关电路

要思考的问题包括

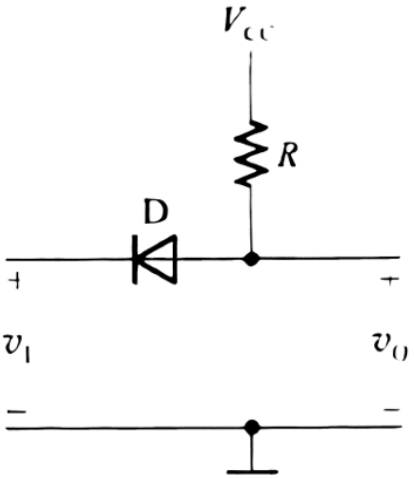
- 模型电路工作原理
- 怎样实现
- 存在哪些问题，怎样改进



1.4.2 一种实现方法——二极管开关电路

要思考的问题包括

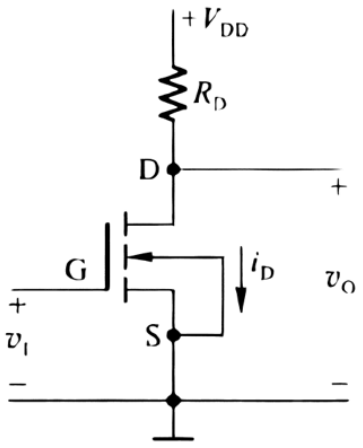
- 工作原理
- 存在哪些问题，怎样改进



1.4.3 一种实现方法——MOS 管基本开关电路

要思考的问题包括

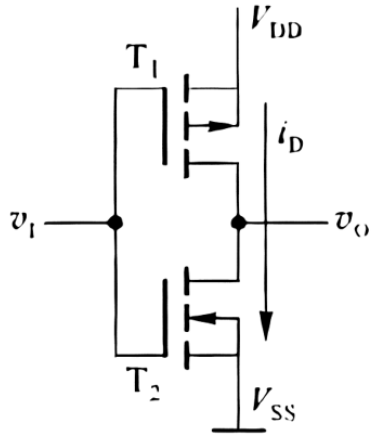
- 工作原理
- 相对于二极管开关电路，在哪些方面做出了改进
- 还存在哪些问题，怎样继续改进 (现在的改进思路是否可以继续应用，如果不可以，是否有其他改进思路)



1.4.4 一种实现方法——CMOS 反相器

要思考的问题包括

- 相对于 MOS 管基本开关电路，在哪些方面做出了改进
- 在演进过程中，哪些基本的东西没有改变，怎样基于互补开关模型电路、二极管开关电路、MOS 管基本开关电路，来理解 CMOS 反相器的工作原理



1.5 关于高阻态

1.5.1 概念

指的是电路的一种输出状态，既不是高电平也不是低电平，如果高阻态再输入下一级电路的话，对下级电路无任何影响. 其电位可能是高电平也可能是低电平，由后一级电路决定.

1.5.2 表示方法

- 在逻辑函数式中表示方法是 Z
- 在卡诺图中表示方法是 $\times$

1.5.3 注意事项

常见的易错点有：在分析含三态门的电路的时候，写出输出逻辑函数式，忘记高阻态

1.6 可用来实现逻辑函数的常用组合逻辑电路模块

1.6.1 译码器

由于 n 位输入的译码器，可以产生所有 n 变量的最小项. 所以，n 位译码器可以用来生成多输出的变量数不超过 n 的任意逻辑函数.

其中重点注意

- 可以生成的逻辑函数变量数不超过译码器输入
- 能够产生多输出逻辑函数

1.6.2 数据选择器

由数据选择器工作原理，n 位地址输入端产生地址选择信号，本质上是 n 变量的最小项. 所以，n 位地址输入的数据选择器可以产生变量数不超过 $(n+1)$ 的任意逻辑函数.

其中重点注意

- 可以生成的逻辑函数变量数不超过 $(n+1)$
- 只能产生单输出逻辑函数

1.6.3 加法器

和以上两种组合逻辑电路模块不同，加法器用来实现逻辑函数时比较有限，一般只能用来产生特定形式的逻辑函数. 具体地讲，就是能够表示为若干逻辑变量或常量相加的形式的逻辑函数.

其中重点注意，这里所说的能够表示为包括能够转化为. 常见的可转化的包括

- 减法——利用补码运算
- 乘法——表示为移位相加

1.7 组合逻辑电路分析

1.7.1 本质

组合逻辑电路分析的本质是逻辑图到自然语言描述的转换. 实际上，可以进一步描述为，是逻辑函数不同表达形式之间的转换.

1.7.2 方法

组合逻辑电路分析的方法实际上就是逻辑函数不同表达形式之间的转换的方法. 具体来讲，其转换过程为

- 逻辑图
- 逻辑函数式
- 真值表

1.7.3 想法分析

组合逻辑电路分析，实际上就是根据组合逻辑电路图，用自然语言描述其功能，也就是前述从逻辑图到自然语言描述的转换. 由于逻辑图是逻辑函数的一种表达形式，我们也可以将组合逻辑电路分析看作，逻辑函数到自然语言描述的转换.

为了实现这一目标，基于我们对逻辑函数及其表达形式的理解，我们想到，应该选择其中最接近自然语言的表达形式来完成这一转换任务. 我们知道，逻辑函数表达形式中，真值表是最直观的. 所以，我们的想法就是，先将逻辑图转换为真值表，进而可以转换为自然语言描述. 其中，借助了逻辑函数式来实现.

1.8 组合逻辑电路设计

1.8.1 本质

组合逻辑电路分析的设计，本质上是从自然语言描述到逻辑图之间的转换. 实际上，也可也看作是逻辑函数不同表达形式之间的转换.

1.8.2 方法

从自然语言到逻辑函数的转换，首先要确定如何描述逻辑函数. 我们首先要做的是引入逻辑变量并赋予其意义. 然后根据意义得到逻辑函数式，再将其转换为合适的形式. 具体操作为

- 首先确定逻辑变量个数和意义
- 选取逻辑函数表达形式其中和自然语言最接近的，也就是“最直观”的——真值表
- 将自然语言表达的问题转换为真值表
- 进一步将真值表转换为逻辑函数式、逻辑电路图

1.8.3 注意

在组合逻辑电路设计过程中，完成从自然语言到真值表、从真值表到逻辑函数式、从逻辑函数式到逻辑图的转换过程. 其中，从逻辑函数式到逻辑图的转换中，要注意，实际操作中要根据器件供应情况，可能需要对逻辑函数形式做一些适当的变换 (包括化简、逻辑函数式常见形式之间的转换等).

1.9 常用组合逻辑电路模块的功能扩展

1.9.1 本质

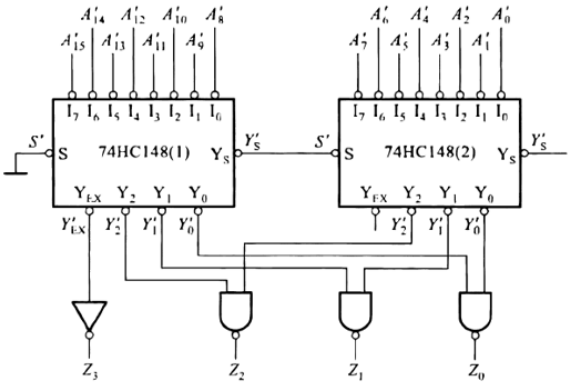
常用组合逻辑电路模块的功能扩展本质上，是基于常用组合逻辑电路模块的组合逻辑电路设计. 即其本质仍然是组合逻辑电路设计，只不过在器件选取时，选择使用常用组合逻辑电路模块，而不是选择基本门电路.

1.9.2 方法

理解了其本质，我们就可以认识到，实际上功能扩展的方法，仍然可以使用组合逻辑电路设计的方法. 只不过从实际操作上，可以认为二者有些区别. 具体表现为，在功能扩展时，主要操作过程是

- 首先确定输入输出变量个数，确定基本模块的个数
- 分析输入变量之间的关系，确定各个模块之间的连接关系
- 分析输出和模块输出之间的关系，对于不能直接由模块输出得到的输出分析如何由输入得到
- 完成各个模块之间的连接
- 将输入和各个模块输入之间对应起来
- 将输出和模块输出以及可能需要的通过输入变换得到的输出对应起来

1.9.3 例子



经典例题总结

2.1 数制和码制

2.1.1 十进制码和格雷码的转换

$(876)_{10}$ 对应的格雷码为 $(\quad\quad)_{格雷}$.

答案：1011011010.

解析：要将十进制数转换为对应的格雷码，要借助二进制码. 进一步利用二进制转换为格雷码的算法，即可得到答案.

按照十进制码转换为二进制码的方法进行转换可以得到

$$(876)_{10} = (1101101100)_2$$

二进制码转换为格雷码的方法是：二进制码的最高位即为格雷码的最高位；从二进制码的最高位起，两两位做异或运算，就得到了对应的格雷码.

2.1.2 二进制码和 BCD 码的转换

$(101101011.101)_2$ 对应的 BCD 码是 $(\quad\quad)_{8421\text{BCD}}$.

答案：$(0011011000011.011000100101)_{8421\text{BCD}}$.

解析：要将二进制码转换为对应的 BCD 码，需要借助十进制码. 进一步利用十进制码转换为 BCD 码的算法，即可得到答案.

按照二进制码转换为十进制码的方法进行转换可以得到

$$(101101011.101)_2 = (363.625)_{10}$$

十进制码转换为 BCD 码的方法是：对于十进制数的每一位，按照十进制码和 BCD 码的对应关系进行转换，进而可得对应的 BCD 码.

2.2 逻辑代数基础

2.2.1 异或运算和同或运算

99 个 1 异或运算的结果记作 F_1,99 个 0 同或运算的结果记作 F_2. 则 $F_1 \oplus F_2$=(
).

答案：1.

解析：对于异或运算，奇数个 1 异或的结果是 1，偶数个 1 异或结果为 0；对于同或运算，奇数个 0 同或结果为 0，偶数个 0 同或结果为 1.

2.2.2 利用公共项实现最简逻辑函数式

试使用最少数目的与非门实现逻辑函数 $Y = A \oplus B$(只用原变量).

答案：$Y = \Big(\big(A \cdot (AB)'\big) \cdot \big((AB)' \cdot B\big)'\Big)'$. 逻辑图略.

解析：按照我们掌握的与或式转换与非与非式的方法，二次取反利用逻辑代数公式即可实现. 而题目要求使用最少的逻辑门，就是要求得到在相应逻辑函数形式下的最简逻辑函数. 我们熟悉与或形式的化简方法，公式法、卡诺图法等. 其他逻辑函数形式的化简并不作为重点要求，但如果遇到相应的题目，也要有一定的解题思路. 其中，利用公共项是一种重要的思路.(这道例题仅作为一种思路的启发，课程要求重点掌握与或式的化简即可)

2.2.3 无关项在化简中的应用

将逻辑函数 $B'CD' + AB'C + C'D' + BC'$ 化简为 $B'D' + B \oplus C$ 时使用了无关项 (
).

答案：$A'B'CD$.

解析：无关项在逻辑函数化简中有着重要的应用. 在使用无关项进行逻辑函数化简时，一般会使用卡诺图法. 对于这一问题，逆向考察. 画出化简前逻辑函数对应的卡诺图，根据化简结果确定“画圈”的方式，即可确定使用的无关项.

2.2.4 多变量逻辑函数的化简

化简 $Y = (AB'C'D + AC'DE + B'DE' + AC'D'E)'$

解：首先观察，题目没有指定方法. 然后发现，逻辑函数式中有 5 个变量，而我们往往在变量数不大于 4 的情况下使用卡诺图法进行化简，理由是当变量数超过 4 时，逻辑相邻性在卡诺图上的表示变得复杂，不易于求解，而且容易忽略一些逻辑相邻关系从而没有得到最简的结果. 这道题目我们以公式法 (结合卡诺图法) 和卡诺图法分别进行求解，希望读者可以比较二者的使用，并产生思考.

法一：使用公式法化简往往是难度较大的，因为不易于直接找到合适的公式进行化简，对于这种多个变量的情形难度往往更大. 这时我们有一个思路，就是希望通过把多个变量的逻辑函数式转化为变量个数较少的逻辑函数式，进而可以方便求解，而且当变量数不超过 4 的时候可以接着使用卡诺图求解. 按照这个思路，我们分析这道题目. 首先记

$$Z = AB'C'D + AC'DE + B'DE' + AC'D'E$$

进行化简可得

$$Z = AC'(B'D + E) + B'DE'$$

此时可以发现，逻辑函数式中 $AC', B'D$ 都是以整体形式出现，则可以令 $F = AC', G = B'D$，进而可得

$$Z = EF + E'G + FG$$

所以得

$$Y = (EF + E'G + FG)' = EF' + E'G' + F'G'$$

此时只有 3 个变量，可以使用卡诺图法化简得到

$$Y = EF' + EG$$

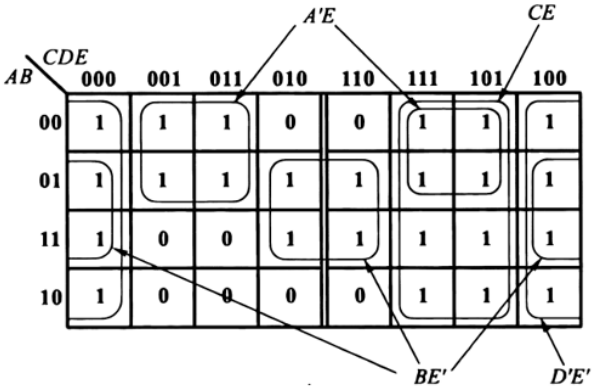
所以得到化简结果

$$Y = A'E + BE' + CE + D'E'$$

法二：使用卡诺图法进行化简. 对于多个变量的情形，我们如果仍然使用卡诺图法进行化简，要注意其逻辑相邻性在卡诺图上的表达形式. 由题意知

$$\begin{aligned} Y(A,B,C,D,E) &= (AB'C'D + AC'DE + B'DE' + AC'D'E)' \\ &= (m_2 + m_6 + m_{17} + m_{18} + m_{19} + m_{22} + m_{25} + m_{27})' \end{aligned}$$

进而可画出卡诺图



注意其中沿中轴线对称的两侧仍为逻辑相邻，化简得到结果

$$Y = A'E + CE + BE' + D'E'$$

建议读者阅读法一中的思路，并比较两种方法，或者考虑其他思路. 可以选择自己觉得合适的方法以应对可能出现的多个变量的逻辑函数式化简问题.

2.2.5 一种形式的逻辑函数式的化简

请用公式法化简

$Y = (A+B+C+D)(A'+B+C+D)(A+B'+C+D)(A+B+C'+D)(A+B+C+D')$

解：这类题目求解时，首先要注意是否指定了方法，如果有则必须按照指定的方法进行化简. 如果没有，可以根据变量的个数进行选择. 如果变量个数较少，则可以考虑使用卡诺图法；如果变量个数较多，考虑是否可以转化为变量个数较少情形，如果可以则尝试转化，如果不可以，则往往只能使用公式法化简.

这道题目指定了使用公式法进行化简. 使用公式法化简时，首先要根据逻辑函数式的形式确定方法，以及公式的选择. 根据这道题目的形式，可以考虑两种思路. 一种是利用反演公式，写出 Y' 的表达形式，并化简，然后再使用反演公式得到 Y 的最简与或式. 另一种思路是表达成最大项之积的形式，进而得到最小项之和的形式，合并相邻项即可得.

2.3 组合逻辑电路

2.3.1 利用加法器实现逻辑函数

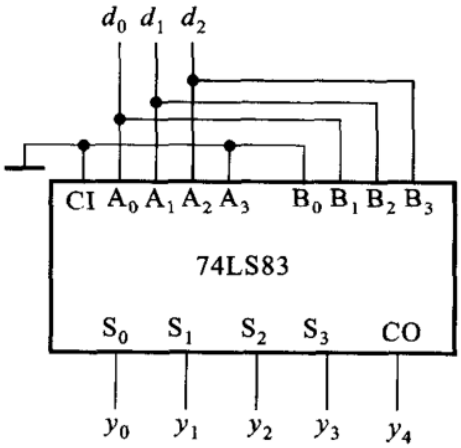
请用 4 位加法器 74LS83 实现一个三位二进制数的 3 倍乘法运算.

解：首先分析题目要求，利用中规模器件实现组合逻辑电路设计. 这类题目在处理的时候要注意，对于译码器和数据选择器这类可以实现任意 (前提是变量数符合要求) 逻辑函数的中规模器件，往往题目可以有多种可能. 而对于其他中规模逻辑器件，往往是实现功能扩展，或者要实现的功能和其本身能够实现的功能有着紧密的联系. 所以在解题的时候要做的就是分析提供的逻辑器件所能够实现的功能，以及要求实现的功能，找到二者之间的联系.

考虑提供的中规模器件——加法器，我们知道可以利用加法器实现组合逻辑电路设计，往往是实现逻辑变量相加或者变量和常量相加运算，而题目要求实现乘法运算，所以需要考虑将其转化为加法运算. 我们要思考的就是如何将乘法转化为加法.

在第一章讨论数制的时候我们学习了数制的基本概念，并且可以发现，对于二进制数来说，左移移位低位补 0 相当于乘 2 运算. 而题目要求的乘 3 运算恰好可以拆分成乘 2 运算结果和原数相加.

所以，我们得到了设计思路，即将原数作为一个输入，将其左移一位作为另一个输入，空余位输入低电平. 即可实现乘 3 运算.



作业题目

3.1 批改作业反馈

3.1.1 第一次作业

- 保留有效数字位数——注意题目要求 (无法得到精确解且题目没有明确说明的话一般保留三位有效数字)
- 计算错误

3.1.2 第二次作业

- 逻辑函数不同表达形式之间转换出错 (常见的如，真值表缺少某些项)
- 逻辑函数化简没有得到最简结果

3.1.3 第三次作业

- 画卡诺图时存在问题——无关项的使用、几何相邻与逻辑相邻
- 逻辑函数式的常见表达形式之间转换

3.1.4 第四次作业

- 第三章利用公式计算的题目，公式掌握不够熟练
- 高阻态的理解
- 画波形图的时候要明确画出输出和输入之间的时序关系