一、数字逻辑基础

1.数字信号和模拟信号：

数字量(digital variable)——在时间和数量上的变化都离散的物理量。

数字信号(digital signal)——表示数字量的信号。

数字电路(digital circuits)——工作在数字信号下的电路。

如：时钟、自动生产线上送出零件量的检测等。

模拟量(analog variable)——在时间或数值上连续变化的物理量。

模拟信号(analog signal)——表示模拟量的信号。

模拟电路(analog circuits)——工作在模拟信号下的电路。

如：温度、压力变化。

2.BCD 码：四位 BCD 码对应十进制一位数字

（1）8421 码不等于二进制，如：（十进制）925=（二进制）1110011101=（8421BCD 码）1001 0010 0101

（2）8421 码禁用 1010～1111，因为超出了一位十进制数

（3）2421 码不具备单值性，因此禁用 0101～1010

（4）余三码（8421 码+3）也不具备单值性，禁用 0000～0010、1101～1111

3.循环码（格雷码）：相邻数对应编码只有一位不同，减少数字电路中状态变化时出错的可能

4.奇偶校验码：奇校验（添加一位校验位使 1 的个数为奇）和偶校验

5.数学运算符、逻辑运算符、按位运算符、缩位运算符、等式运算符

数学运算符：+（逻辑与）、•（逻辑或）、—（逻辑非）、⊕（逻辑异或）、⊙（逻辑同或）

逻辑运算符：&&（逻辑与）、||（逻辑或）、!（逻辑非）、⊕（逻辑异或）、⊙（逻辑同或）

按位运算符：&（按位与）、|（按位或）、~（按位取反）、^（按位异或）、^~（按位同或）

缩位运算符（对一个多位变量从低位向高位运算）：&（按位与）、|（按位或）、~&（按位与非）、～|（按位或非）、^（按位异或）、^~（按位同或）

等式运算符：==（等于）、===（全等）、!=、!==

6.机器码

机器码	+00001111	−00001111	+0.00001111	−0.00001111
原码	000001111	100001111	0.00001111	1.00001111
补码	000001111	111110001	0.00001111	1.11110001
反码	000001111	111110000	0.00001111	1.11110000
移码	100001111	011110001	小数无移码	小数无移码

由[X]补求[-X]补：

运算过程是：将[X]补连同符号一起将各位取反，末位再加 1

二、逻辑代数基础

1.易错运算律

（1）A+(B•C)=(A+B)•(A+C)　/　A•(B+C)=A•B+A•C

（2）A+A•B=A　/　A•(A+B)=A

（3）A•B+A•~~B~~=A　/　(A+B)(A+~~B~~)=A

（4）A•B+~~A~~•C+B•C=A•B+~~A~~•C　/　(A+B)(~~A~~+C)(B+C)=(A+B)(~~A~~+C)

（注：4.1 证明 B•C 补(A+~~A~~) 4.2 证明对 4.1 取反）

2.对偶式中 0 变 1　+变•　A 不变 ~~A~~!!　F=G 等价于 F'=G'

3.同或逻辑与异或逻辑同时互为相反和对偶

4.下标相同的最大项和最小项互为相反

$$\overline{m_3} = \overline{A\,BC} = A + \overline{B} + \overline{C} = M_3$$

$$\overline{M_3} = \overline{A + \overline{B} + \overline{C}} = \overline{A}\,BC = m_3$$

$$\begin{aligned} &m_1+m_2+m_4+m_5 \\ =\ &\overline{m_1}+\overline{m_2}+\overline{m_4}+\overline{m_5} \\ =\ &\overline{m_1\cdot m_2\cdot m_4\cdot m_5} \\ =\ &M_0\cdot M_3\cdot M_6\cdot M_7 \end{aligned}$$

5.逻辑表达式**化简**（**卡诺图**应用）

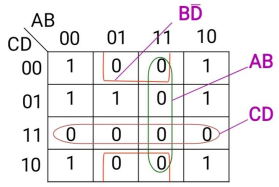
（1）化最简“与-或”式：直接卡诺图

（2）“与-或”式化最简“或-与”式：“两次取反法”（注意一次取反后相当于取**卡诺图的 0 项**）

例　用卡诺图求逻辑函数

$$F(A,B,C,D)=\sum m(0,1,2,5,8,9,10)$$

的最简“或-与”表达式。



解　用卡诺图化简给定函数的过程如右图所示，得到：

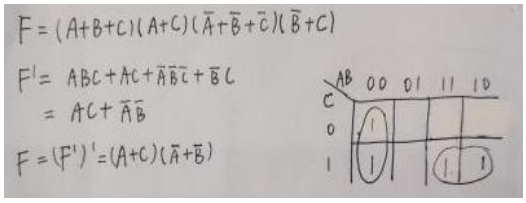
$$\overline{F} = AB + CD + B\overline{D}$$

再对反函数 $\overline{F}$ 的最简“与-或”表达式　$\overline{F} = AB + CD + B\overline{D}$

两边取反，即可求得函数的最简“或-与”表达式

$$F = \overline{\overline{F}} = (\overline{A}+\overline{B})\cdot(\overline{C}+\overline{D})\cdot(\overline{B}+D)$$

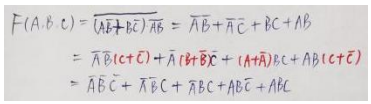
（3）“或-与”式化最简“或-与”式：“两次对偶法”

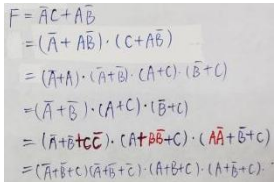


6.**最简**表达式化标准“与-或”式/“或-与”式

（1）标准“与-或”式：添项 A=A•(B+~~B~~)（分配律在+•下不太看得出来可以**看成与和或**）

（2）标准“或-与”式：添项 A=A+B•~~B~~





三、组合逻辑电路

1.设计方法

（1）真值表取 1 -> 卡诺图化简 -> 逻辑表达式 -> 电路图

例、三人裁判举重比赛，一个主裁判，两个副裁判。认为杠铃举上时，各裁判按自己前面的电键（为1），否则不按（为0）；裁判结果用红绿灯表示，红绿灯均亮（为1）表示"完全举上"，只红灯亮表示"需研究录像决定"，其余为未举上。

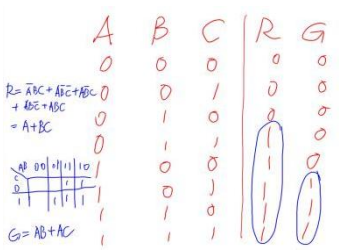
（1）三个裁判均按下自己的电键，红绿灯全亮；

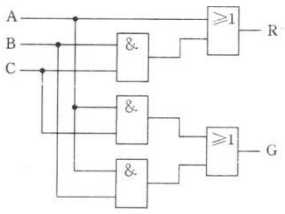
（2）两个裁判（其中一个为主裁判）按下自己的电键，红绿灯全亮；

（3）两个副裁判或一个主裁判按下自己的电键，只红灯亮；

（4）其余情况红绿灯全灭。

试用两级与或电路实现满足上述四种要求的逻辑控制电路。





举重裁判电路逻辑图

（2）逻辑转化为表达式 -> 电路图

设计两个3位二进制数是否相等的数值比数器。

解 设两3位二进制数分别为$A=A_3A_2A_1$，$B=B_3B_2B_1$，电路的输出为F。当A=B时，F为1；否则F为0。

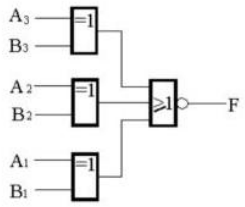
根据常识可知，要使A=B，必须使$A_3=B_3$、$A_2=B_2$、$A_1=B_1$，即要使F为1，必须使A_3和B_3同时为0或同时为1、A_2和B_2同时为0或同时为1、A_1和B_1同时为0或同时为1。即

$$F=(\overline{A_3}\,\overline{B_3}+A_3B_3)(\overline{A_2}\,\overline{B_2}+A_2B_2)(\overline{A_1}\,\overline{B_1}+A_2B_2)$$

$$=\overline{A_3\oplus B_3}\bullet\overline{A_2\oplus B_2}\bullet\overline{A_1\oplus B_1}$$

$$=\overline{(A_3\oplus B_3)+(A_2\oplus B_2)+(A_1\oplus B_1)}$$

画出用异或门实现的电路图：



2.注意事项：

（1）题意转化逻辑表达式时需要将状态抽象，用几个变量组合代表情况，如下示例中用 2 个变量代表 4 种输入

· 设计判断献血者与受血者的血型是否相容的电路。血型相容规则如下表所示，表中"√"表示血型相容。

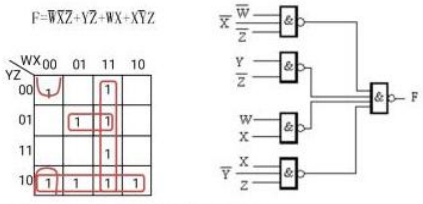
献血	受血			
	A	B	AB	O
A	√		√	
B		√	√	
AB			√	
O	√	√	√	√

血型	献	受
	WX	YZ
A	00	00
B	01	01
AB	10	10
O	11	11

解 四种血型可用两个变量的四种编码表示。设变量W、X表示献血者的血型，Y、Z表示受血者的血型，采用上表所示的编码。电路的输出用F表示，当血型相容时F为1，否则F为0。

根据血型相容规则，可直接得出函数的卡诺图如下图所示，由卡诺图可得函数的最简"与或"式为：

$$F=\overline{W}\overline{X}Z+\overline{Y}Z+W\overline{X}+X\overline{Y}\overline{Z}$$



用"与非"门实现的电路图如上图所示。

（2）化"与-非"门：**两次取反（1 项太多的话找 0 项然后取反）**（只有"与-非"门，没有单独的非门，想用非门可以直接单输入的"与-非"门）

· 组合逻辑电路设计中应考虑的问题

1 逻辑函数形式的变换

（1）逻辑函数的"与非"门实现

有两种方法：一种是对F两次求反，一次展开；另一种是对F三次求反，一次展开。

例 用"与非"门实现逻辑函数 $F=A\overline{B}+B\overline{C}+C\overline{D}+D\overline{A}$

解 方法一：对F两次求反，一次展开可得：

$$F=A\overline{B}+B\overline{C}+C\overline{D}+D\overline{A}=\overline{(\overline{AB})\bullet(\overline{BC})\bullet(\overline{CD})\bullet(\overline{DA})}$$

方法二：对F三次求反，一次展开可得：

$$\overline{F}=\overline{A\overline{B}+B\overline{C}+C\overline{D}+D\overline{A}}=\overline{A}BC\overline{D}+\overline{A}\overline{B}C\overline{D}+\overline{A}\overline{B}\overline{C}D$$

$$F=\overline{\overline{A}BC\overline{D}+\overline{A}\overline{B}C\overline{D}+\overline{A}\overline{B}\overline{C}D}=\overline{(\overline{A}BC\overline{D})\bullet(\overline{A}\overline{B}C\overline{D})\bullet(\overline{A}\overline{B}\overline{C}D)}$$

（3）化"或-非"门：取对偶 -> 化成最简"与-或"式 -> **两次取反** -> 取对偶

(2) 逻辑函数的"或非"门实现

可以采用对F两次求对偶的方法。即，先求F的对偶式F_d，并将其化为最简"与非-与非"式，然后再求F_d的对偶式$(F_d)_d$，则$(F_d)_d$即是F的最简"或非-或非"式。

例：用"或非"门实现函数 $F=A\bar{B}+B\bar{C}+C\bar{A}$

解 先求F的对偶式F_d并将其化成最简"与-或"式：

$$F_d=(A+\bar{B})(B+\bar{C})(C+\bar{A})=AB\bar{C}+A\bar{B}C$$

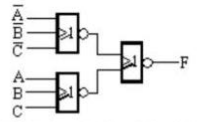
再将F_d的最简"与-或"式两次求反，一次展开变为"与非-与非"式：

$$F_d=\overline{(\overline{ABC})\cdot(\overline{A\bar{B}C})}$$

再求对偶，则得：

$$F=(F_d)_d=\overline{(\overline{A}+\overline{B}+\overline{C})+(\overline{A}+B+\overline{C})}$$

画出下图所示的逻辑电路图：



也可先求出F的最简"或-与"式，并将其两次取反，一次展开，即可得到最简的"或非-或非"式。该方法请自己练习。

（4）化"与-或-非"门：两次取反

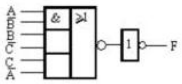
·(3) 逻辑函数的"与或非"门实现

例：用"与或非"门实现函数 $F=A\bar{B}+B\bar{C}+C\bar{A}$

两种方法，一种是对F两次求反，另一种是对$\bar{F}$一次求反。

解 方法一：对F两次求反，可得：

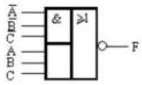
$$F=\overline{\overline{A\bar{B}}\cdot\overline{B\bar{C}}\cdot\overline{C\bar{A}}}$$



方法二：先求F，再对F一次求反可得：

$$F=\overline{\overline{A\bar{B}}\cdot\overline{B\bar{C}}\cdot\overline{C\bar{A}}}=A\bar{B}C+AB\bar{C}$$

$$F=\overline{(F)}=\overline{A\bar{B}C+AB\bar{C}}$$



两种方法对应的电路图比较：

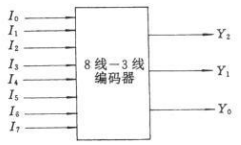
3.常见器件

（1）普通编码器

一、普通编码器（Common Encoder）

特点：任何时刻只允许输入一个编码信号，否则将发生混乱。

3位二进制普通编码器示例：



3位二进制普通编码器框图

其余2**8-8种输入均为约束项，无效输入

3位二进制普通编码器真值表

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

由于普通编码器在任何时刻 $I_0 \sim I_7$ 当中仅有一个取值为1，即只有真值表中所列的8种状态，而且它的（2^8-8）种状态均为约束项。因此，由真值表可得到逻辑式：

$$Y_2=\bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3I_4\bar{I}_5\bar{I}_6\bar{I}_7+\bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4I_5\bar{I}_6\bar{I}_7+\bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5I_6\bar{I}_7+\bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5\bar{I}_6I_7$$

因为只有一个有效输入，因此小项中省略其他项

$$\begin{cases}Y_2=I_4+I_5+I_6+I_7\\ Y_1=I_2+I_3+I_6+I_7\\ Y_0=I_1+I_3+I_5+I_7\end{cases}$$

或门实现

$$\begin{cases}Y_2=\overline{\overline{I_4}\cdot\overline{I_5}\cdot\overline{I_6}\cdot\overline{I_7}}\\ Y_1=\overline{\overline{I_2}\cdot\overline{I_3}\cdot\overline{I_6}\cdot\overline{I_7}}\\ Y_0=\overline{\overline{I_1}\cdot\overline{I_3}\cdot\overline{I_5}\cdot\overline{I_7}}\end{cases}$$

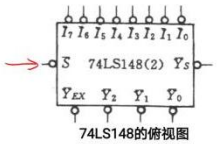
与非门实现

（2）优先编码器（74LS148）

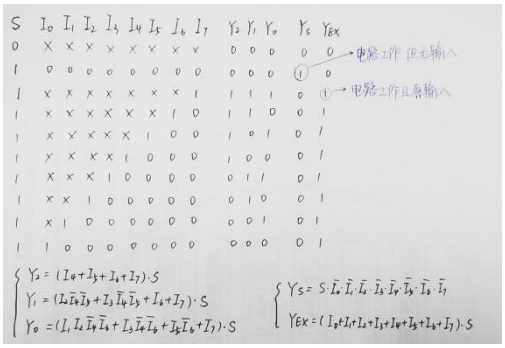
二、优先编码器（Priority Encoder）

特点：允许同时输入两个以上编码信号。不过在设计优先编码器时已经将所有的输入信号按优先顺序排了队，当几个输入信号同时出现时，只对其中优先权最高的一个进行编码。

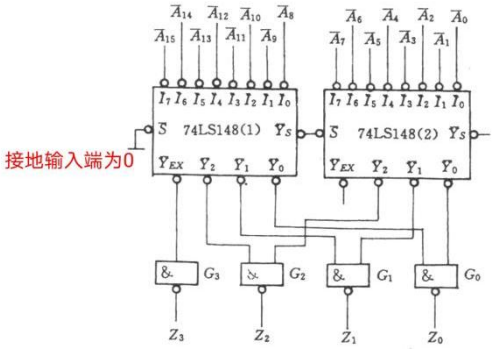
下面以8线-3线优先编码器74LS148为例分析优先编码器的工作原理。74LS148框图（俯视图）如下：



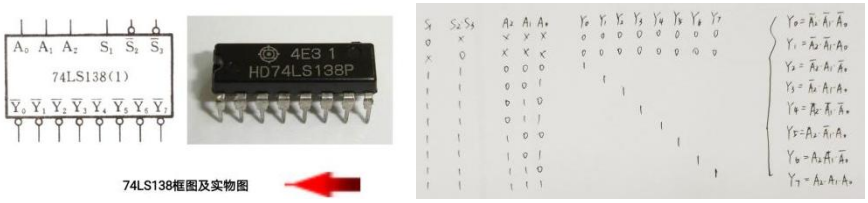
74LS148的俯视图



应用：两片 74LS148 扩成 4 位编码器

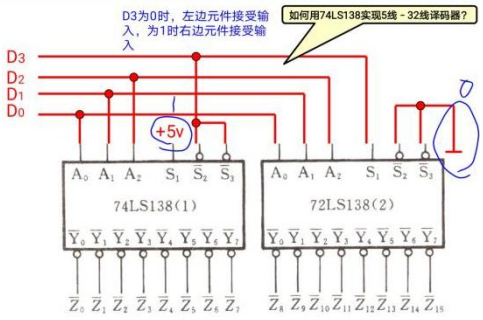


（3）最小项译码器（**74LS138**）



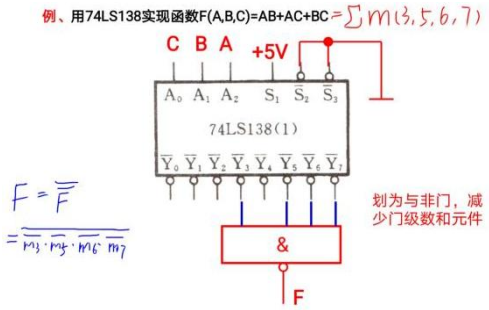
74LS138框图及实物图

应用 1：两片 74LS138 扩成 4 位译码器



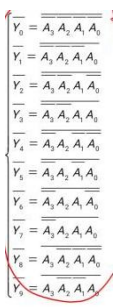
用两片74LS138接成的4线－16线译码器

应用 2：实现逻辑函数（划为**最小项**的**与非形式**）



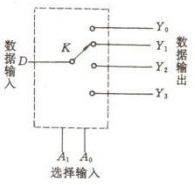
（4）二-十进制译码器

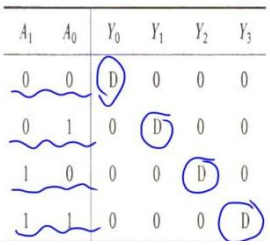
74LS42功能表



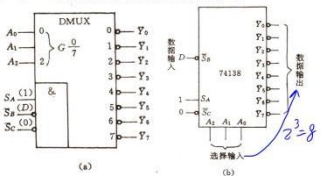
（5）数据分配器 DEMUX

DEMUX的功能如同多位开关一样，将输入D送到选择输入指定的通道上（如图所示）。



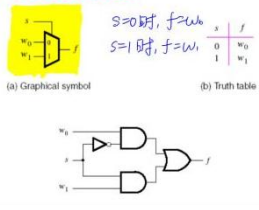


74138不仅可以作3线－8线译码器，而且还可用作1路－8路数据分配器（如图所示）。

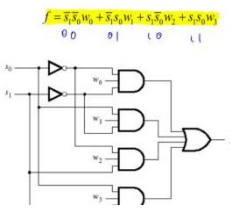


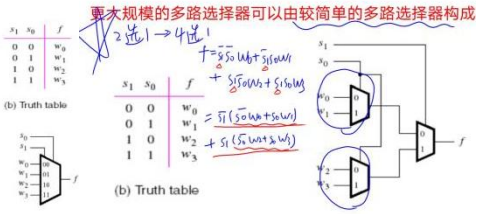
（6）数据选择器 MUX

2 选 1 多路选择器



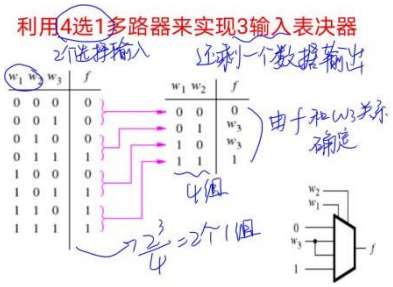
4 选 1 多路选择器





更大规模的多路选择器可以由较简单的多路选择器构成

应用 1：列出真值表后分组求 f 表达式



应用 2：结合**香农展开定理**

假设我们想要用2选1的多路器和其他必须的门电路来实现以下函数：

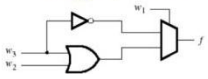
$$f = \overline{w}_1\overline{w}_3 + w_1w_2 + w_1w_3$$

若以w 1 为基础来进行香农展开，则函数可以表示为：

$$f = \overline{w}_1 f_{\overline{w}_1} + w_1 f_{w_1}$$
$$= \overline{w}_1(\overline{w}_3) + w_1(w_2 + w_3)$$

w_1为选择信号 w_2 w_3为输入信号

w_1=0代入f w_1=1代入f



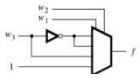
香农展开也能以多个变量进行分解，以w1 w2例，所得结果如下：

$$f(w_1,w_2,\ldots,w_n) = \overline{w}_1\overline{w}_2 f(0,0,w_3,\ldots,w_n) + \overline{w}_1 w_2 f(0,1,w_3,\ldots,w_n) + w_1\overline{w}_2 f(1,0,w_3,\ldots,w_n) + w_1 w_2 f(1,1,w_3,\ldots,w_n)$$

上述展开式可用 4 选 1 多路器实现

$$f = \overline{w}_1\overline{w}_3 + w_1w_2 + w_1w_3$$

$$f = \overline{w_1}\,\overline{w_2} f_{\overline{w_1}\,\overline{w_2}} + \overline{w_1} w_2 f_{\overline{w_1} w_2} + w_1\overline{w_2} f_{w_1\overline{w_2}} + w_1 w_2 f_{w_1 w_2}$$
$$= \overline{w_1}\,\overline{w_2}(\overline{w_3}) + \overline{w_1} w_2(\overline{w_3}) + w_1\overline{w_2}(w_3) + w_1 w_2(1)$$

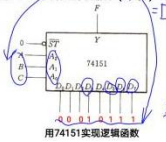


应用 3：实现逻辑函数（**最小项**）

原理：若逻辑函数变量的数目与数据选择器选择输入端的数目相等，则函数最小项的数目就与数据选择器输入端的数目相同，这样可以直接用数据选择器实现逻辑函数。

例、试用74151实现逻辑函数F(A,B,C)=AB+AC+BC

$=\sum m(3,5,6,7)$



数据输入直接0/1

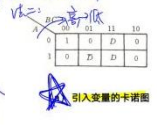
例、试用一片74151实现逻辑函数

$F(A,B,C,D) = AB\overline{C}\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\overline{C}D + ABCD + ABC(1)$

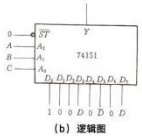
变量数目比选择输入端记多，因此数据输入端要输入变量

法一：卡诺图降维

法二：



引入变量的卡诺图

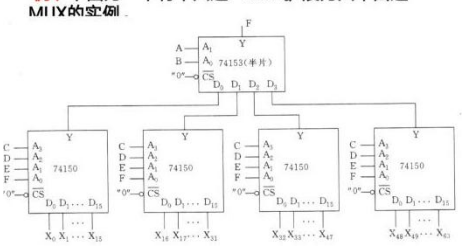


思考：如何用一片74151实现逻辑函数

$$F(A,B,C,D,E) = A\overline{B}C(\overline{D}\,\overline{E}) + AB\overline{C}(\overline{D}\,\overline{E}) + A\overline{B}\overline{C}(D\,\overline{E}) + A\overline{B}C(D\,E) + \overline{A}\overline{B}\overline{C}(\overline{D}\,E)$$

应用 4：扩展

例、下图为一个将十六选一MUX扩展为六十四选一MUX的实例



（7）数值比较器

一位：

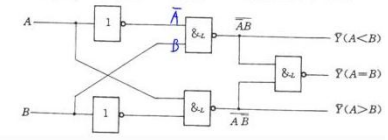
两个1位二进制数A,B相比的情况有以下几种：

①A>B（即A=1,B=0），则 $A\overline{B}=1$，所以可用 $A\overline{B}$ 作为A>B的输出信号 $Y_{(A>B)}$ 。

②同理可用 $\overline{A}B$ 作为A<B的输出信号 $Y_{(A<B)}$ 。

③同理可用A⊙B 作为A=B的输出信号 $Y_{(A=B)}$ 。

于是，1位数值比较器的电路图可如下设计：



多位：从高向低比较

（8）加法器

半加器：

半加器的真值表

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

半加器的逻辑表达式

$$\begin{cases} S=A\overline{B}+\overline{A}B=A\oplus B \\ CO=AB \end{cases}$$

全加器：

将两个多位二进制数相加时，除了最低位以外，每一位都应考虑来自低位的进位，即将两个对应的加数和来自低位的进位3个数相加。这种运算称为全加，所用电路称为全加器。

1位全加器的真值表、逻辑表达式、电路图和惯用符号如下所示：

全加器的真值表

输入			输出	
CI	A	B	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

全加器的逻辑表达式

$$\begin{cases} S=\overline{\overline{A\overline{B}\cdot\overline{CI}}+\overline{A\overline{B}\cdot CI}+\overline{\overline{A}B\cdot CI}+\overline{AB\cdot\overline{CI}}} \\ CO=\overline{\overline{A\overline{B}}+\overline{B\cdot CI}+\overline{A\cdot CI}} \end{cases}$$

或

$$\begin{cases} S=A\oplus B\oplus CI \\ CO=AB+CI\ (A+B) \end{cases}$$

4.竞争冒险现象

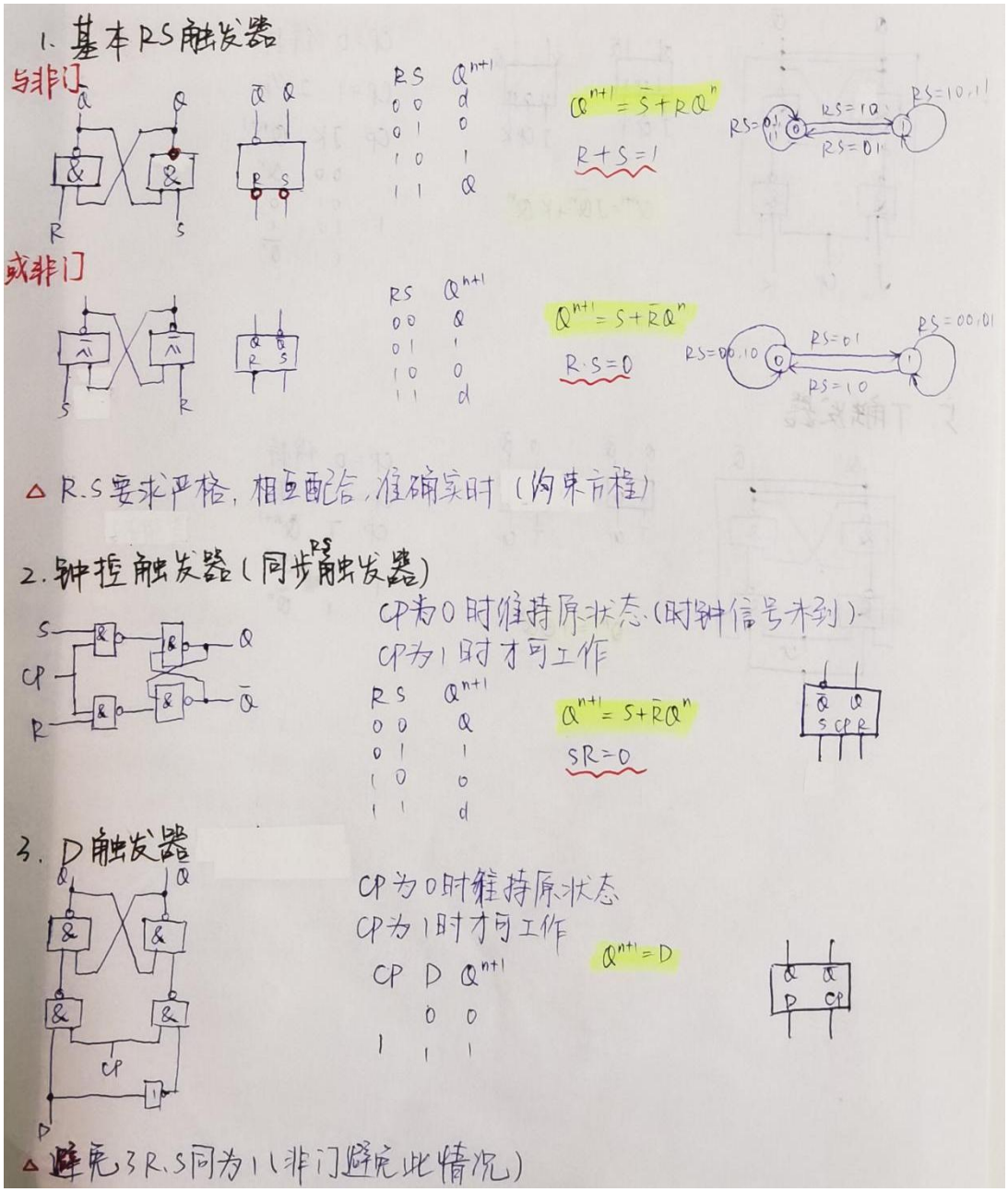
竞争：两个输入端向相反方向变化且有时差

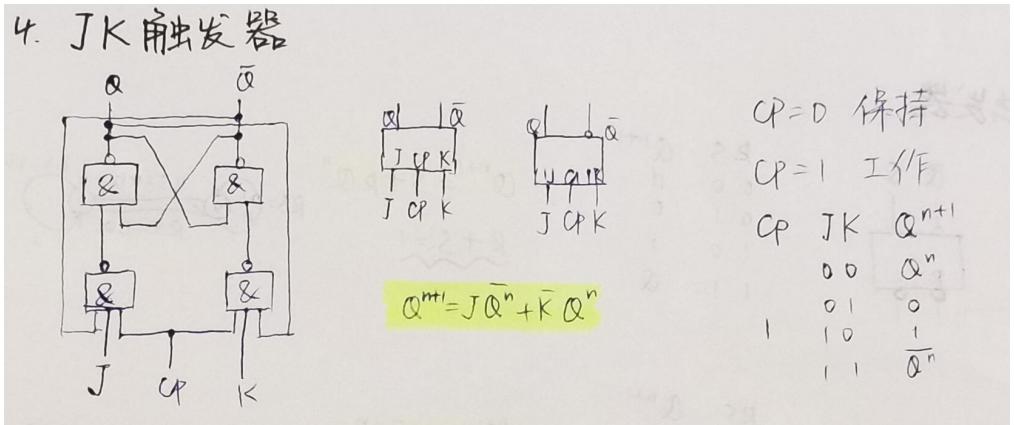
冒险：竞争并且产生了尖峰脉冲

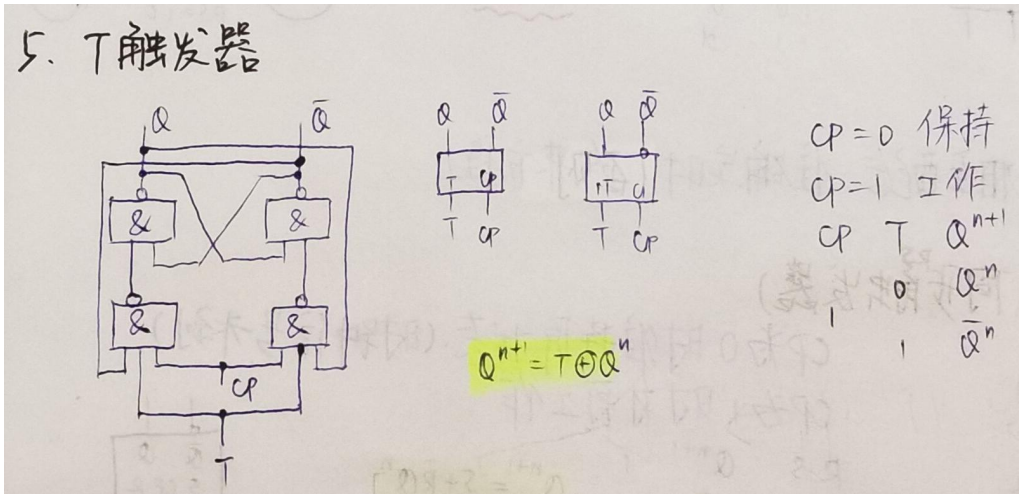
四、时序逻辑电路

1.时序逻辑电路不一定要有时钟信号，如：锁存器（电平触发，无时钟信号）

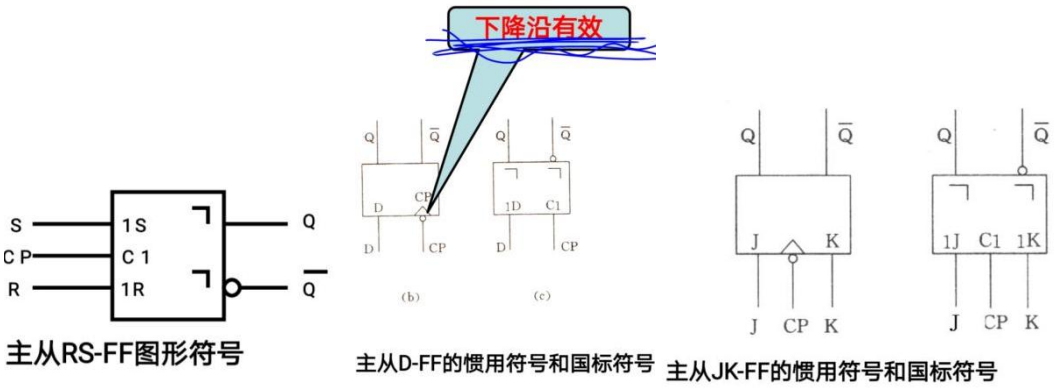
2.触发器







附：触发器图形符号



主从RS-FF图形符号　　主从D-FF的惯用符号和国标符号　　主从JK-FF的惯用符号和国标符号

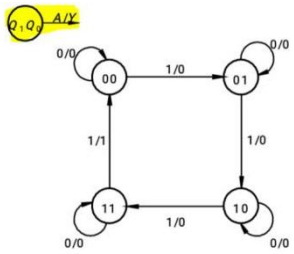
3.触发器与锁存器

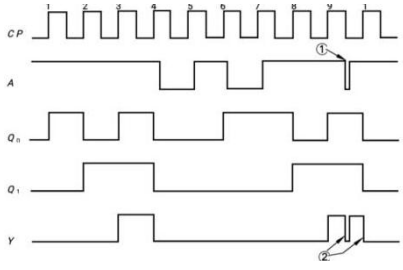
触发器是由边沿信号触发，锁存器是由电平信号触发

4.时序电路：Mealy 型（输入和存储共同决定输出）和 Moore 型（输入不决定输出）

5.图表描述：状态转换表、状态转换图、时序图

$Q_1^nQ_0^n$	$Q_1^{n+1}Q_0^{n+1}/Y$	
	A=0	A=1
00	00/0	01/0
01	01/0	10/0
10	10/0	11/0
11	11/0	00/1





6.同步时序电路分析过程

同步时序电路分析的一般步骤：

1、从给定的逻辑图中写出每个触发器的驱动方程；

2、把得到的这些驱动方程代入相应触发器的特性方程，得出每个触发器的状态方程，从而得到由这些状态方程组成的整个时序电路的状态方程组；

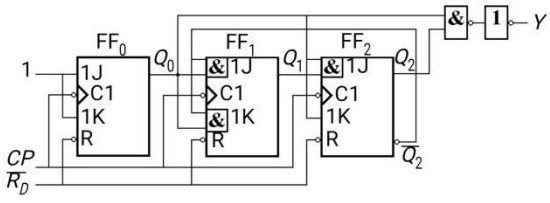
3、根据逻辑图写出电路的输出方程；

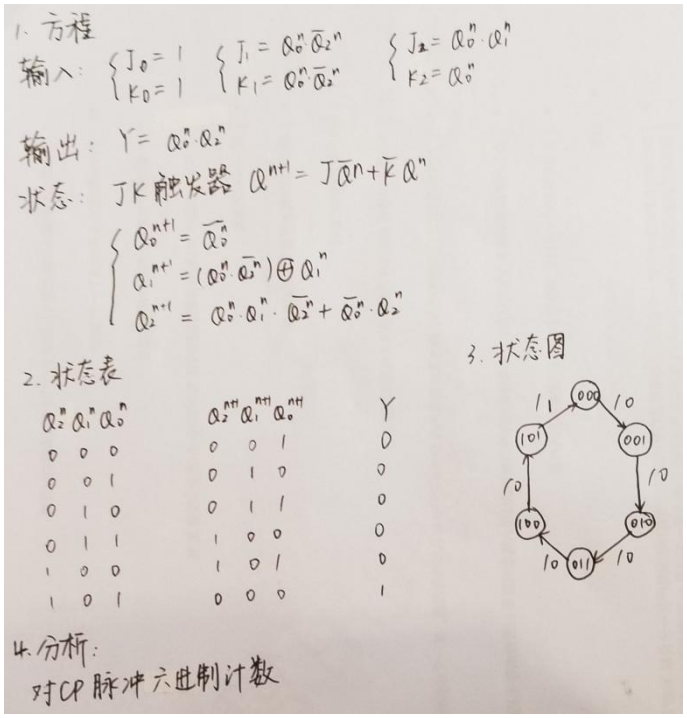
4、列出该电路的状态转换表；

5、根据状态表画出状态转换图（或时序图）；

6、根据图表描述电路的逻辑功能，并进行自启动验证。

Example：

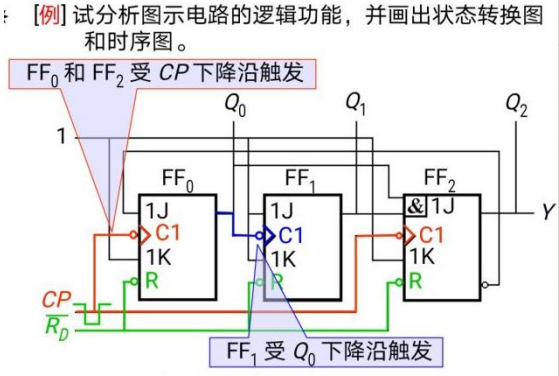


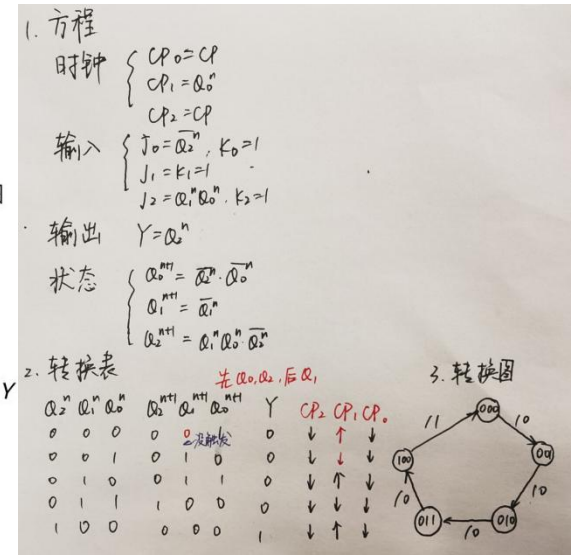


7.异步时序电路分析

（1）**先分析系统时钟触发的状态**，再分析中间结果触发的状态

（2）加一个时钟方程





8.同步时序电路设计

（1）**状态化简**

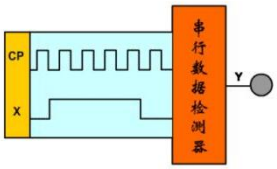
（2）状态编码时采用“**相邻**”选择，0 和 1 表示触发器的现态值

（3）画卡诺图时**卡诺圈尽量画大**，以减少表达式中的逻辑变量

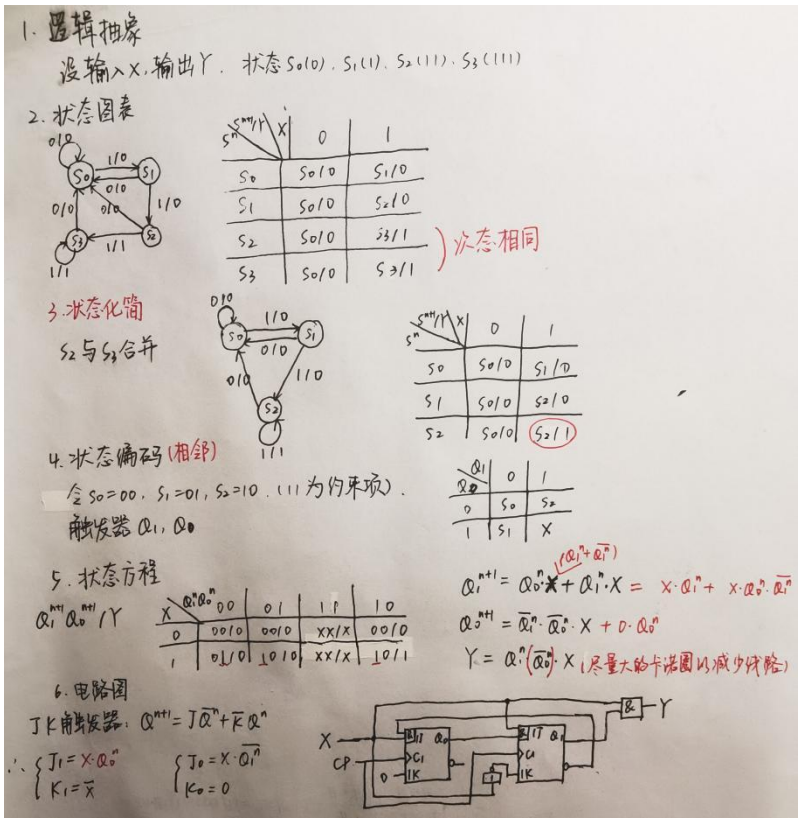
（4）化状态方程时**•(A+A)**或者**+0•A**

普通情况（有输入、输出）：

例:试设计一个串行数据检测器。对它的要求是：连续输入三个或三个以上的'1'时输出为'1'，其它情况输出为'0'(试用上边沿JK-FF完成设计)。



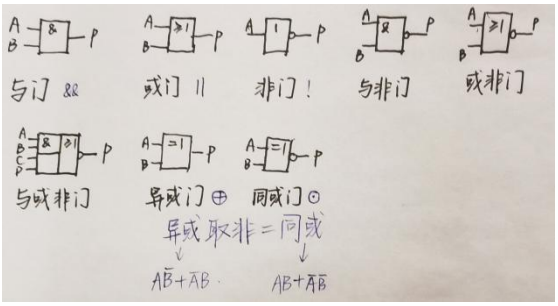
总体设计效果示意图



无输入情况：

五、画图

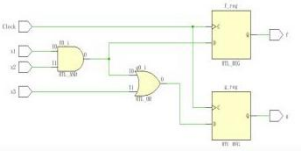
1.基本门元件



2.阻塞与非阻塞赋值

阻塞赋值

```
87module example7_5 (x1, x2, x3, Clock, f, g);
88    input x1, x2, x3, Clock;
89    output reg f, g;
90    always @(posedge Clock)
91    begin
92        f = x1 & x2;
93        g = f | x3;
94    end
95endmodule
```



非阻塞赋值

```
87module example7_5 (x1, x2, x3, Clock, f, g);
88    input x1, x2, x3, Clock;
89    output reg f, g;
90    always @(posedge Clock)
91    begin
92        f <= x1 & x2;
93        g <= f | x3;
94    end
95endmodule
```