

Ch 1

Number Systems

进制转换

10进制转2进制

整数部分**除2取余，逆序排列**，小数部分**乘2取整，顺序排列**（这个过程直到为1时结束，但也有小数不能用有限位二进制数表示，保留至需要的精度即可）。

如 $725.678 = 10\ 1101\ 0101.1010\ 1101\ 1001$

2进制转16进制

每四位转化即可，**整数右对齐，小数左对齐**。

如 $725.678 = (2D5.AD9)_{16}$

2进制转8进制

每三位转化，**整数右对齐，小数左对齐**。

如 $725.678 = (1325.6331)_8$

数字编码

10进制	BCD码 (8421码)	Excess-3码	84-2-1码	Gray码
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

BCD码

加法大于9时，加6校正，产生的进位为新的一位，如 $1000 + 0101 = 1101(13 > 9)$, $1101 + 0110 = 0011, carry = 1$, 最终结果为 $0001\ 0011$ 。

Excess-3码和84-2-1码

十进制下和为9的两数，在Excess-3和84-2-1下和为1111。

Gray码

相邻两数之间至多有一位的变化。

校验位 | parity bit

	even parity	odd parity
1010100	11010100	01010100
1000001	01000001	11000001

偶检验：有偶数个1时最高位填0，反之填1。

奇校验：有偶数个1时最高位填1，反之填0。

Non-Numeric

字符编码

ASCII字符编码

7位二进制编码，可表示128个字符，高三位构成下表的列，低四位构成下表的行。

B ₄ B ₃ B ₂ B ₁	B ₇ B ₆ B ₅							
	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL
控制字符								

Unicode



2bytes，几乎所有字符。

Ch 2

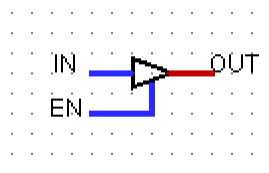
二值逻辑和逻辑门

容易忘的逻辑门

XOR&XNOR

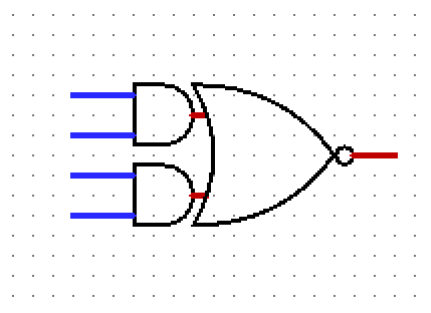
异或 (XOR)		$F=X\bar{Y}+\bar{X}Y$ $=X\oplus Y$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	X	Y	F	0	0	0	0	1	1	1	0	1	1	1	0
X	Y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
异或非 (XNOR)		$F=XY+\bar{X}\bar{Y}$ $=\overline{X\oplus Y}$	<table><tr><th>X</th><th>Y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X	Y	F	0	0	1	0	1	0	1	0	0	1	1	1
X	Y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

三态门 | 3-State Buffer



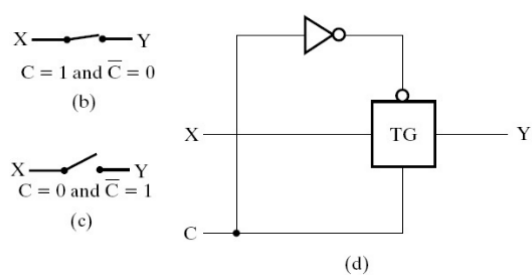
EN	IN	OUT
0	x	Hi-Z (高阻)
1	0	0
1	1	1

AND-OR-INVERT | AOI



不加反相器是AND-OR (AO)，类似地有OAI和OA。

传输门 | Transmission Gate



高电平通路，低电平断路。

布尔代数

对偶式(duality), 反函数

def: 变量为**字符(literal)**。

对偶: and->or, or->and, 保持literal不变。对偶的对偶是函数本身。

反函数: 记函数 \overline{F} 为 F 反函数, 等价于 F 对偶后, 再将所有literal取反。

相关恒等式

德摩根律	$\overline{X_1 + X_2 + \dots + X_n} = \overline{X_1} \overline{X_2} \dots \overline{X_n}$	$\overline{X_1 X_2 \dots X_n} = \overline{X_1} + \overline{X_2} + \dots + \overline{X_n}$
分配律	$X + YZ = (X + Y)(X + Z)$	
Minimization	$XY + \overline{X}Y = Y$	$(X + Y)(\overline{X} + Y) = Y$
Absorption	$X + XY = X$	$X(X + Y) = X$
Simplification	$X + \overline{X}Y = X + Y$	$X(\overline{X} + Y) = XY$
Consensus	$XY + \overline{X}Z + YZ = XY + \overline{X}Z$	$(X + Y)(\overline{X} + Z)(Y + Z) = (X + Y)(\overline{X} + Z)$

Shannon Formula

- $xf(x, \overline{x}, y, \dots, z) = xf(1, 0, y, \dots, z)$
- $\overline{x}f(x, \overline{x}, y, \dots, z) = \overline{x}f(0, 1, y, \dots, z)$
- $x + f(x, \overline{x}, y, \dots, z) = x + f(0, 1, y, \dots, z)$
- $\overline{x} + f(x, \overline{x}, y, \dots, z) = \overline{x} + f(1, 0, y, \dots, z)$

Shannon Expansion

$$f(x, \overline{x}, y, \dots, z) = xf(x, \overline{x}, y, \dots, z) + \overline{x}f(x, \overline{x}, y, \dots, z) = xf(1, 0, y, \dots, z) + \overline{x}f(0, 1, y, \dots, z)$$

PS: 直接化简较复杂时, 考虑先求对偶, 化简完后再求对偶。

标准形式

最大项(Maxterms)与最小项(Minterms)

- 最大项是以OR连接的, 如 $X + Y + Z, X + \overline{Y} + Z$, 分别记作 M_0, M_2 。
- 最小项是以AND连接的, 如 $XYZ, X\overline{Y}Z$, 分别记作 m_7, m_5 。
- 两者之间有 $M_i = \overline{m_i}$ 。

Sum of Minterms(SOM)&Product of Maxterms(POM)

SOM的找法是: 根据真值表, 找出其中所有函数结果为**1**的行, 每行即为一个最小项, 将他们之间用OR连接。

POM的找法是: 根据真值表, 找出其中所有函数结果为**0**的行, 每行即为一个最大项, 将他们之间用AND连接。

Sum of Products(SOP)&Products of Sum(POS)

即SOM和POM的标准形式。

反函数

注意下标就行了。

举个例子, $f(x, y, z) = \sum_m(1, 3, 5, 7)$

$\bar{f}(x, y, z) = \sum_m(0, 2, 4, 6)$

也可写为: $\bar{f}(x, y, z) = \prod_M(1, 3, 5, 7)$

电路成本标准

Literal Cost

看literal的多少。如 $F = BD + ABC + ACD$ 的cost是8。

Gate Input Cost

- literal的数目。
- 除单个literal外的项数。
- 不同取反值的单个literal。

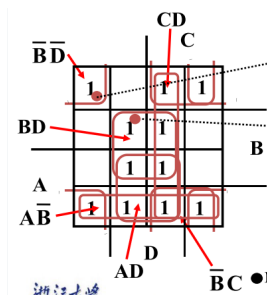
如 $F = BD + \bar{A}\bar{B}C + A\bar{C}\bar{D}$, $G=11$, $GN=14$ 。

卡诺图 | K-map

几个概念

具体的讲解书上很清楚, 这里再重申几个概念。

- 蕴含项 | implicant: 卡诺图中为1的方格。
- 主蕴含项 | prime implicant: 由 2^m ($m = 0, 1, \dots, n$) 个1方格构成的矩形的集合, 每个矩形包含了尽可能多的1方格。
- 质主蕴含项 | essential prime implicant: 至少包含了一个不被其他主蕴含项包含的1方格。



如上图, 质主蕴含项是 BD 和 $\bar{B}\bar{D}$ 。

! 两种函数形式优化

若给出的是SOP形式函数:

- 找出标1的项, 化简得SOP形式;
- 找到标0的项, 求出 \bar{F} 的SOP形式, 再取反得到 F 的POS形式。

若给出的是POS形式函数:

- 利用 $M_i = \bar{m}_i$, 对POS形式的函数先求反;
- 对求反后的函数, 将最小项对应的位置标0, 其他地方标1;
- 合并1方格可得SOP表达式, 合并0方格再求反得POS表达式。

无关项

合理使用无关项可以进一步化简函数。

门的传播延迟

传播延迟 | propagation delay

- t_{PHL} : 输出从高变到低时, 从输入到输出的时间差;
- t_{PLH} : 输出从低变到高时, 从输入到输出的时间差;
- $t_{pd}: \max(t_{PHL}, t_{PLH})$ 。

传输延迟 | transport delay

输入响应输出的变化, 在指定的传播延迟之后发生改变。

惯性延迟 | inertial delay

类似传输延迟, 但两次输入变化发生于小于拒绝时间 (rejection time), 那么两次变化中的第一次将不会发生。

PS: 拒绝时间小于等于传播延迟。

Ch 3

分层设计 | Hierarchical Design

注意模块的重复利用。

工艺映射 | Technology Mapping

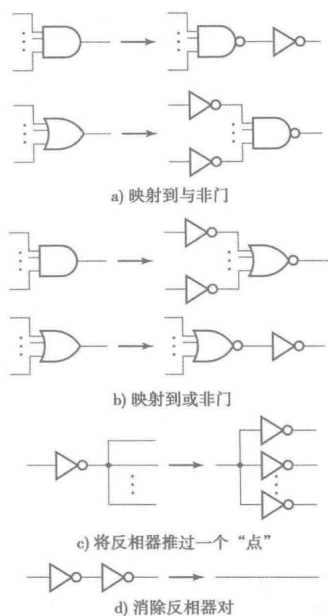
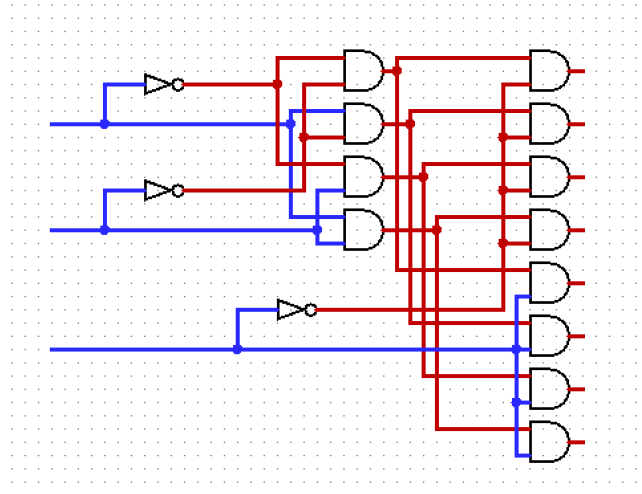


图 3-3 将与门、或门和反相器映射为与非门、或非门和反相器

译码

3-8 Decoder



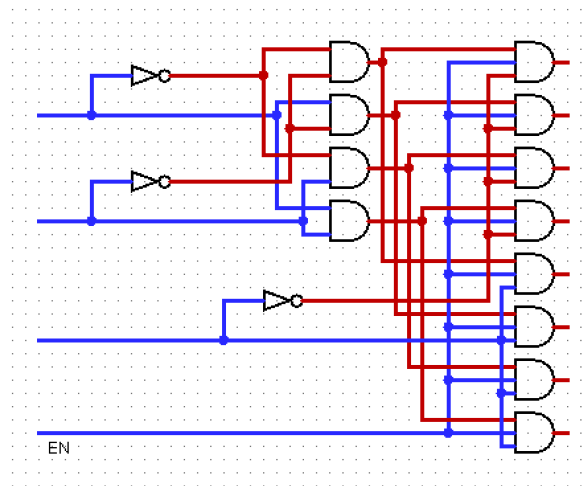
一般性原则 ($n - to - 2^n$)

可以拆成如下规模的decoder:

$$\begin{cases} 2^{\frac{n}{2}} \text{ and } 2^{\frac{n}{2}}, n \text{ is even} \\ 2^{\frac{n-1}{2}} \text{ and } 2^{\frac{n+1}{2}}, n \text{ is odd} \end{cases}$$

重复上面步骤直到 $n = 1$, 对于 $n = 1$, 用 1-to-2 decoder。

带使能的译码器



编码 | Encoding

Priority Encoder

如果有多于一个的输入1, 选取优先级最高的。

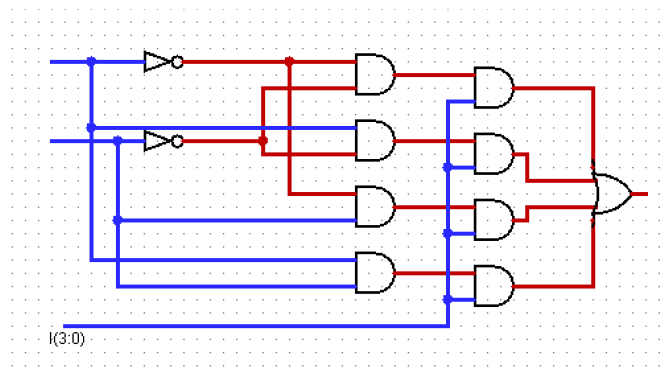
表 3-6 优先编码器真值表

输 入				输 出		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

这样可以减少无关信号，在卡诺图中可以使用无关项化简。

多路复用器 | Mux

4-1 多路复用器

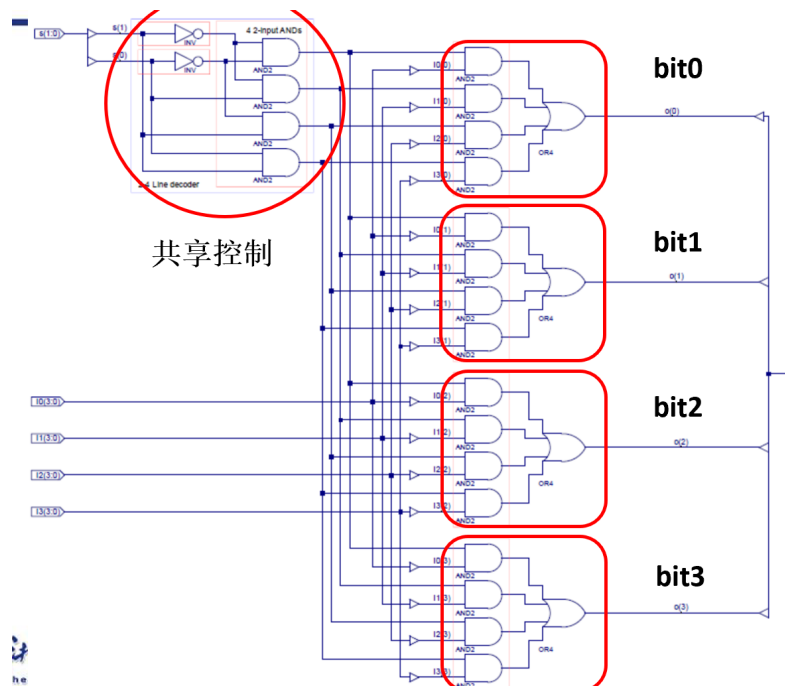


四重4-1多路复用器

$2^n - 1$ 的Mux输出都是一位的，考虑将输出向量化，变为多位。

四重4-1Mux可以选择四个**四位**的向量，通过共享译码模块结合四个选择模块，用16个与门。

重数即为可选择向量数。



七段数码管



图 3-38 7 段显示

涉及到BCD码转换到七段码真值表，卡诺图化简。

迭代组合电路

半加器和全加器

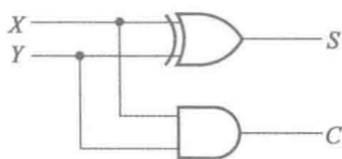


图 3-40 半加器的逻辑电路图

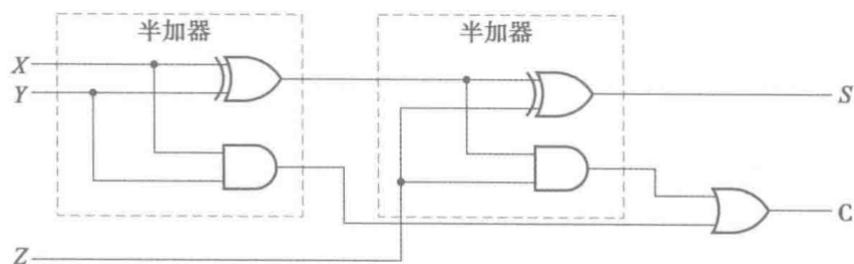


图 3-42 全加器的逻辑图

行波进位加法器

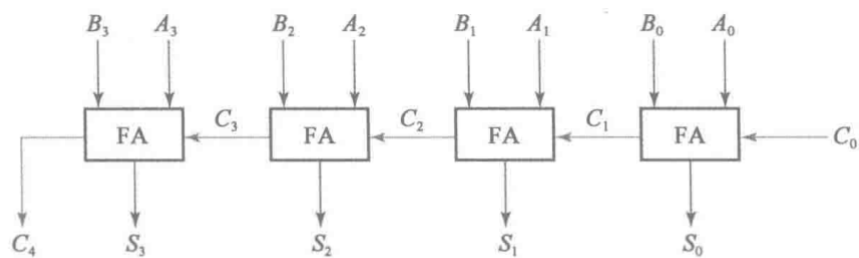


图 3-43 4 位行波进位加法器

补码

r 进制系统均有**基数补码**和**基数反码**。

2进制：

反码 (1's complement)：定义为 $(2^n - 1) - N$ ，逐位求反得。

补码 (2's complement)：定义为 $2^n - N$ ，逐位求反再加1。

采用补码的无符号数减法

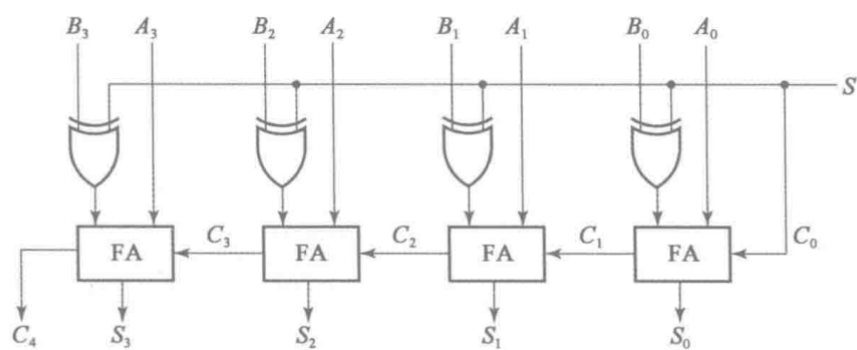


图 3-45 加减法器电路

- 当执行减法 ($S = 1$)， $C_4 = 0$ ，即 $A - B > 0$ 时，得到的结果就为实际结果；
- $C_4 = 1$ ， $A - B \leq 0$ 时，得到的结果为实际结果补码的相反数。

有符号数加减法

有符号数的加减法对上面做了校正使得减法结果统一起来，均为补码形式。正数补码为自身，负数补码取反加一。

溢出 | Overflow

无符号数：最高位相加产生进位时则发生溢出。

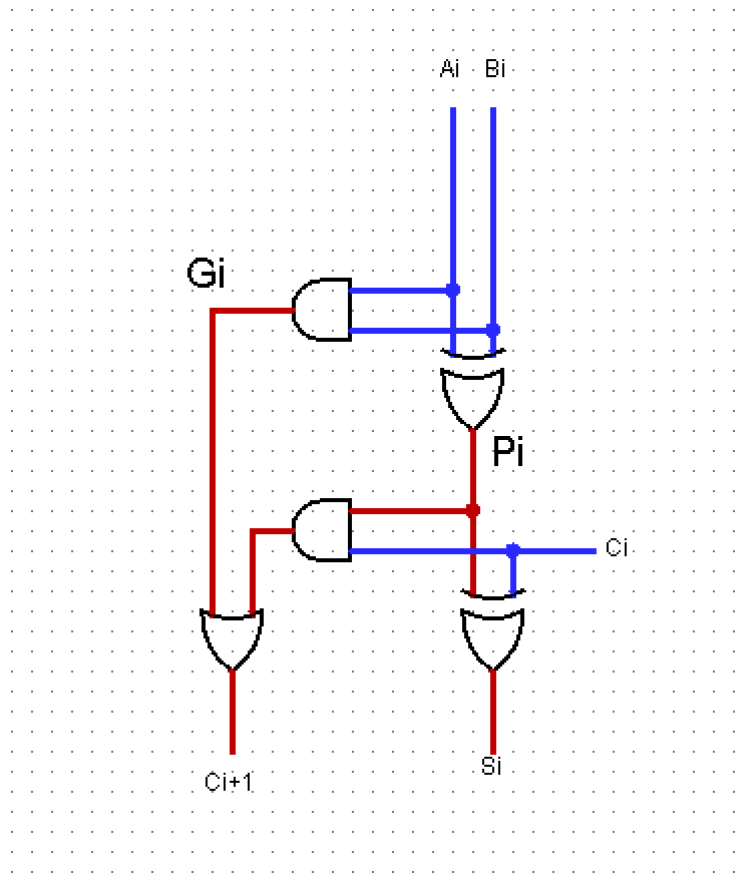
有符号数：溢出检测可以通过检测符号位的进位输入和进位输出是否相等，若不等，则发生溢出。

✓PS：正+正，负+负才可能产生溢出。

超前进位加法器 | Carry Lookahead

行波进位加法器是逐级传递进位的，位数较多时，会很慢。

某一级全加器



定义 $G_i = A_i B_i$, $P_i = A_i \oplus B_i$ 。

于是有 $S_i = P_i \oplus C_i$, $C_{i+1} = G_i + P_i C_i$, 于是有如下代换：

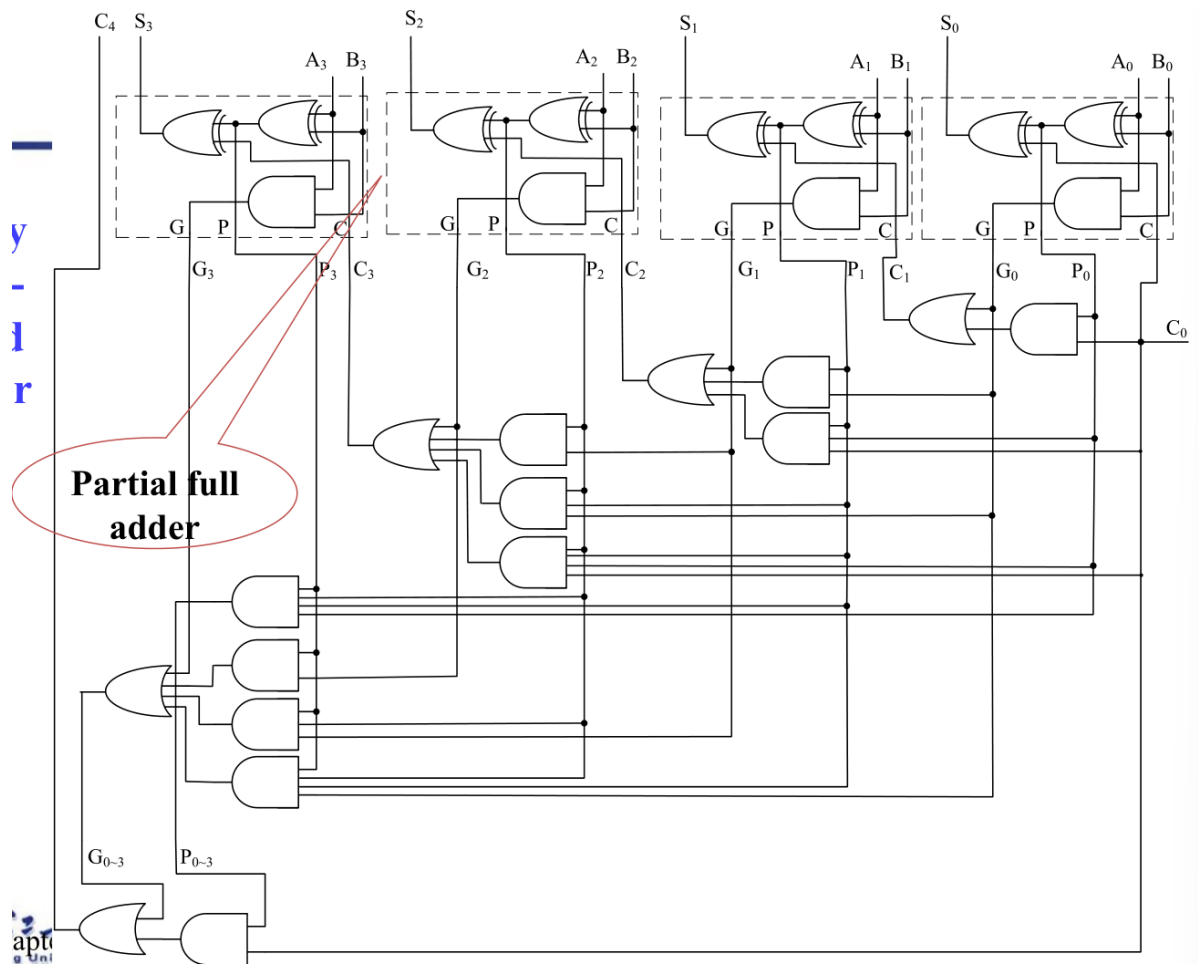
$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

可以注意到每一级的进位不再依赖上一级的结果，而是在一开始就可以算好，这样就提高了计算速率。



Group Carry Lookahead

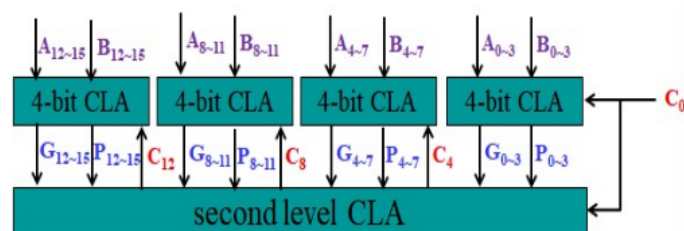
随着位数的增加，门的扇入越来越大，这是不可接受的，因此考虑用迭代的方式来加速更多位的计算。

定义 $G_{0\sim3} = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$, $P_{0\sim3} = P_3P_2P_1P_0$ 。

每四位一划分，有

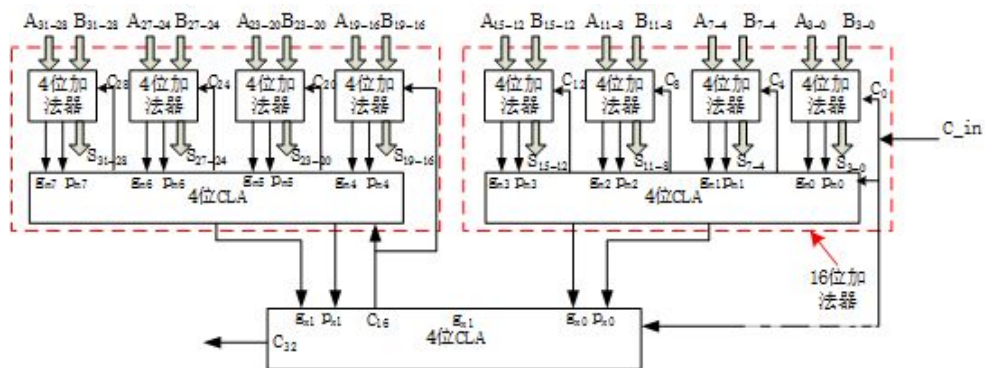
$$\begin{aligned} C_4 &= G_{0\sim3} + P_{0\sim3}C_0 \\ C_8 &= G_{4\sim7} + P_{4\sim7}C_4 \\ C_{12} &= G_{8\sim11} + P_{8\sim11}C_8 \\ C_{16} &= G_{12\sim15} + P_{12\sim15}C_{12} \end{aligned}$$

通过逐次代换 C_4, C_8, C_{12} 可以使式子全部只含 C_0 ，16位CLA如下图：



加速后的时间复杂度从 $O(n)$ 变为 $O(\log_2 n)$ (但是我不知道怎么证明的hhh)

? Question: 要设计32位CLA，一种方案是把两个16CLA采用串联方式，即级内超前进位，级间行波串联。但这不符合设计超前进位的初衷，因此采用多级CLA的方式。



注：这里第三级CLA实际上只用到两位。

其他算数单元

- 自增 | Incrementing
- 自减 | Decrementing
- 乘常数/除常数（这里进一步特殊化为常数2，即为左移、右移）
- 零扩展/符号扩展

Ch4

时序电路

基础概念

定义

时序电路的输出由输入和当前状态决定。

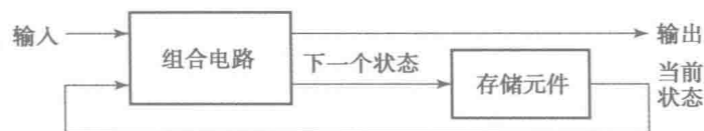


图 4-1 时序电路的框图

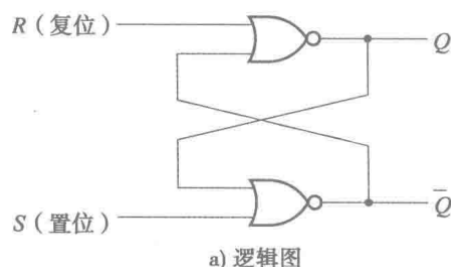
类型

同步 | Synchronous: 状态的改变总发生由时钟控制，改变发生在时钟变化时，是离散变化的。现在计算机几乎都是同步的。

异步 | Asynchronous: 状态改变可以发生在任何时候。

锁存器

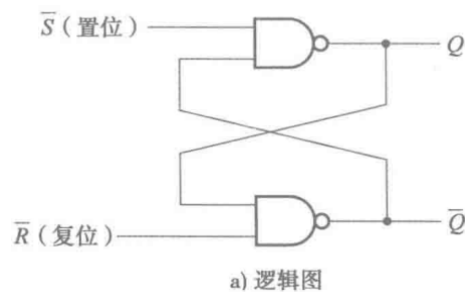
S-R Latch (NOR)



R	S	Q	\overline{Q}	状态描述
0	0	x	x	储存未知状态
0	1	1	0	Q置1
0	0	1	0	Q储存为1
1	0	0	1	Q置0
0	0	0	1	Q储存为0
1	1	0	0	都为0, 导致状态未定义
0	0	x	x	不稳定状态!

$S = 1, R = 1$ 是非法的, 会导致锁存器处于未定义状态。

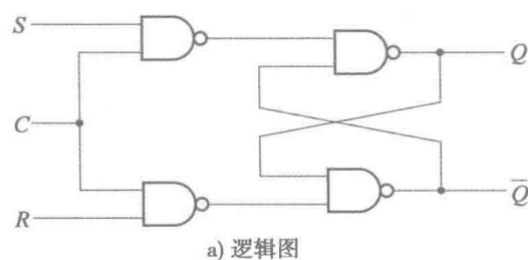
$\overline{S} - \overline{R}$ Latch (NAND)



\overline{R}	\overline{S}	Q	\overline{Q}	状态描述
1	1	x	x	储存未知状态
1	0	1	0	Q置1
1	1	1	0	Q储存为1
0	1	0	1	Q置0
1	1	0	1	Q储存为0
0	0	1	1	都为1, 导致状态未定义
1	1	x	x	不稳定状态!

$\overline{S} = 0, \overline{R} = 0$ 是非法的, 会导致锁存器处于未定义状态。

SR触发器 | Clocked S-R Latch

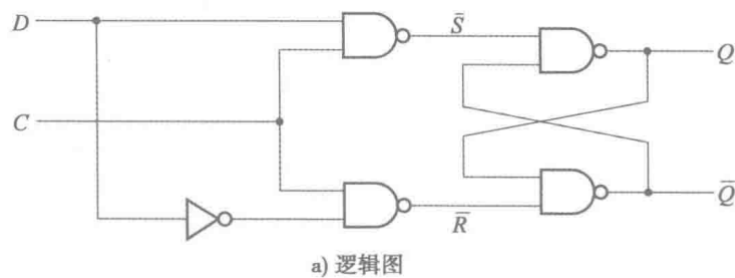


实际上用了NAND门, 但又取反了, 输入和S-R Latch相同, 所以还是叫SR触发器了

C	S	R	Next Q	Comment
0	x	x	无变化	无变化
1	0	0	无变化	无变化
1	0	1	0	复位
1	1	0	1	置位
1	1	1	x	不允许

D锁存器 | D Latch

上面还是没有解决未定状态的问题，因此引入D锁存器。

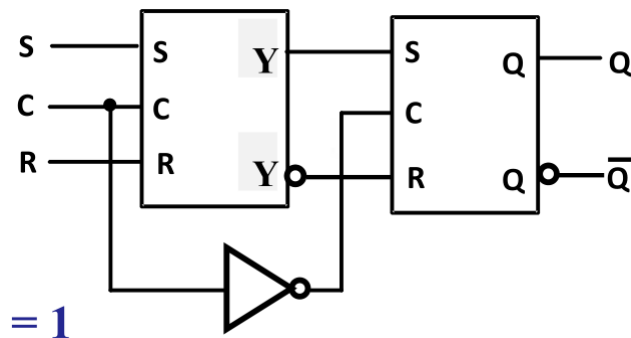


触发器 | Flip-Flops

问题引入

为了解决一个时钟周期内输出多次改变的空翻现象。在一个时钟周期内，我们希望触发器只翻转一次。

主从触发器 | Master Slave Flip-Flop



时钟在高电平时，master有效；时钟低电平时，salve有效。

1's catching: 造成问题的原因是S端口有0->1->0的Giltch。假设R=0, Q=0, S=0, Gitch为从0到1再到0，这样锁存器捕捉到1的信号，然后将Y置1，在时钟低电平时，将Q置1。但是实际上S端的有效信号是0，Q端不应该有变化。R端口有Giltch也会造成这个问题。

边沿触发式触发器 | Edge-Triggered D Flip-Flop

这种触发器不会引发1's catching的问题。

负边沿触发

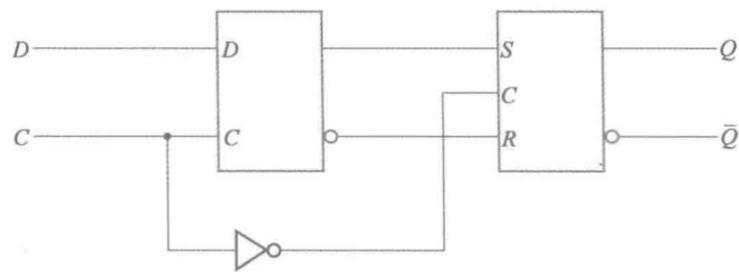


图 4-9 负边沿触发的 D 触发器

负边沿触发即C从1变到0时才触发（对于slave而言）。

正边沿触发

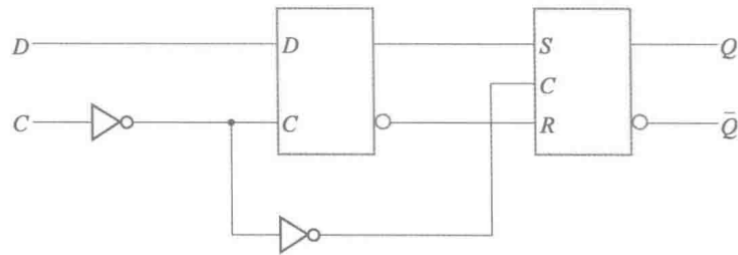
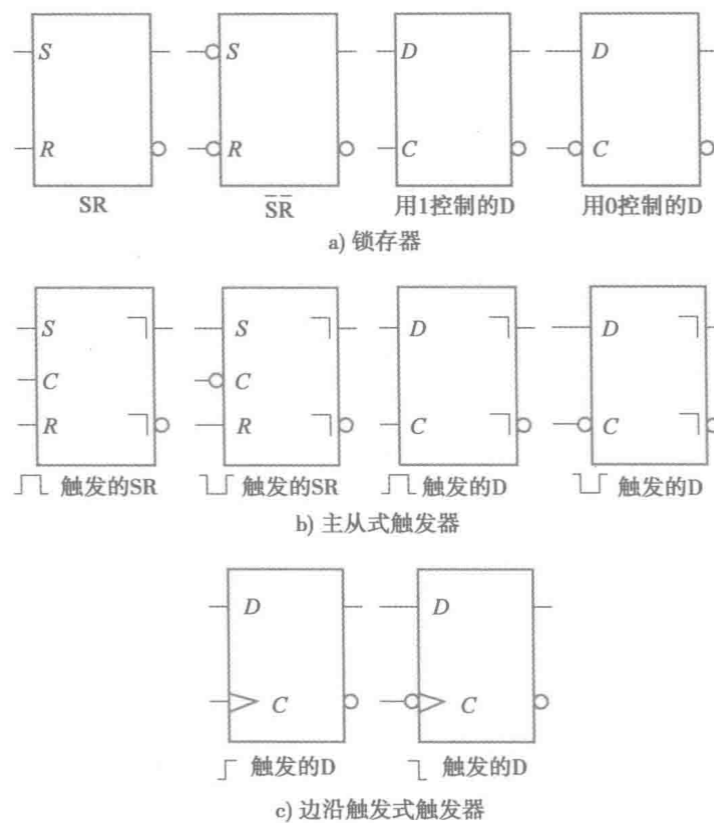


图 4-10 正边沿触发的 D 触发器

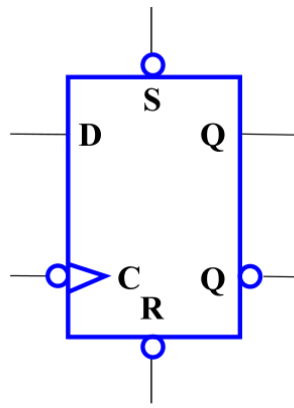
正边沿触发即C从0变到1时才触发。

标准图形符号



直接输入

即初始化输入，这部分输出不受时序电路的控制。



\overline{R} 置0, 触发器置0; \overline{S} 置0, 触发器置1。

其他触发器

JK触发器

类似于SR触发器, J对应S, K对应R, **但J=K=1是允许的**, 这时输出状态变为和之前的相反的状态。同样的, 也有1's catching的问题。但可以用D触发器修正这个问题。

T触发器

T=0的时候, 保持; T=1时, 变为相反状态。其实就是把JK短接。

时序电路分析

状态表、图

时序电路中真值表变为状态表。

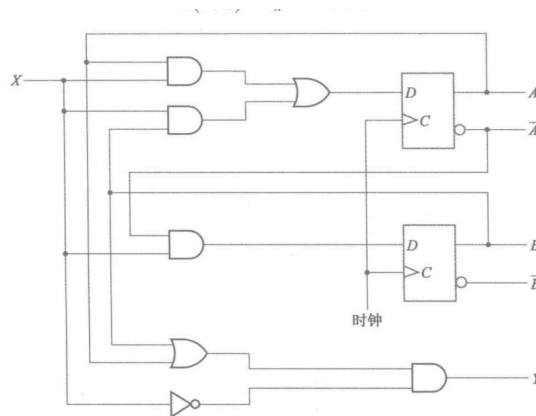
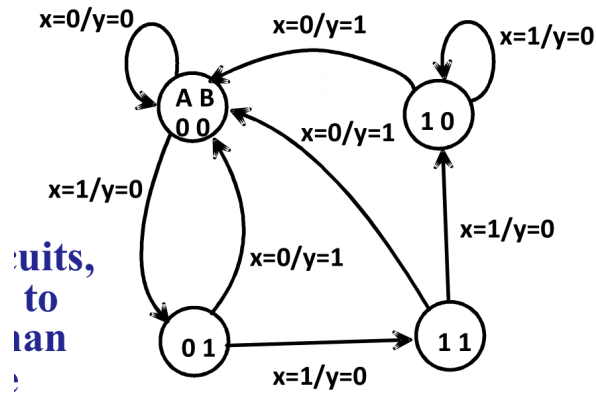


图 4-13 时序电路的例子

一般来说, 包含m个触发器 (m个状态变量) 和n个输入的时序电路状态表有 2^{m+n} 行。

- 若电路输出和当前状态和输入都有关, 则为Mealy型。
- 若只和当前状态有关, 则为Moore型。
- 输出在**箭头**上: Mealy
- 输出在**圆圈**里: Moore

如下图为Mealy



等价状态

对于两个状态，对于每一个可能的输入，他们的输出和状态转移的结果都一样。据此可以实现状态的合并化简。

时序电路设计

步骤

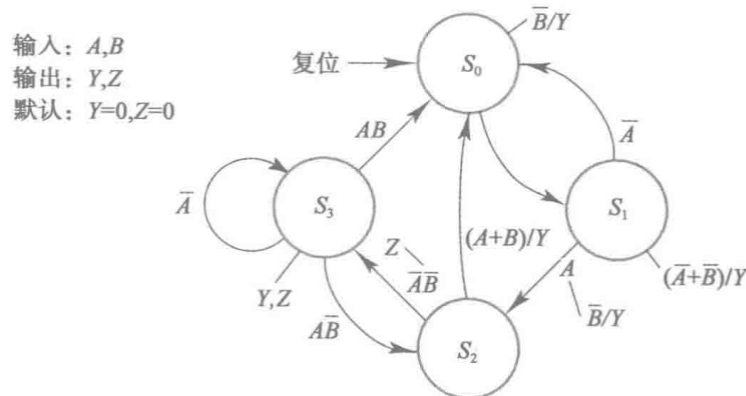
1. 形式化 | Formulation: 画状态图和状态表
2. 状态赋值
 - counting order
 - 格雷码顺序
 - one hot码
3. 确定触发器输入方程和输出方程
4. K-map优化

无效状态的处理

- Way1: 保证无效状态的输出不会危害正常状态
- Way2: 采取额外输出或者未使用的代码说明电路进入无效状态
- Way3: 定义无效状态的下一个状态，使得下一状态不论外界输入如何，电路在几个周期后可以回到正常状态。

状态机图 | State Machine Diagram

状态转移条件不是直接的0/1的值，而是具体的表达式了，如：



为了保证无效状态不会出现，对于每个状态的转换条件有：

$$T_{ij}T_{ik} = 0 \quad (j \neq k)$$

$$\sum T_{ij} = 1$$

输出也要满足：

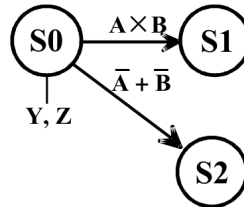
$$O_{ij}O_{ik} = 0 \ (j \neq k)$$

$$\sum O_{ij} = 1$$

无条件转移隐含1，无条件输出隐含1。

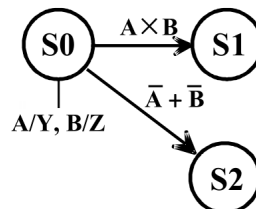
几种输出模型

- Moore Output



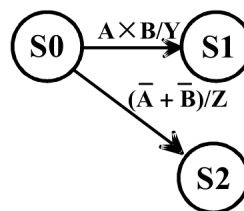
Ex. 1: Moore Outputs

- TCI(Transition condition independent) Mealy Output



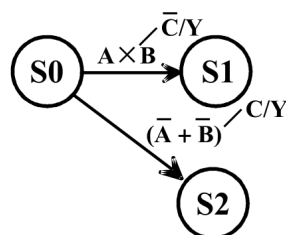
Ex. 2: TCI Outputs

- TCD Mealy Output



Ex. 3: TCD Outputs

- TCOD Mealy Output



Ex. 4: TCOD Outputs

Timing

触发器Timing

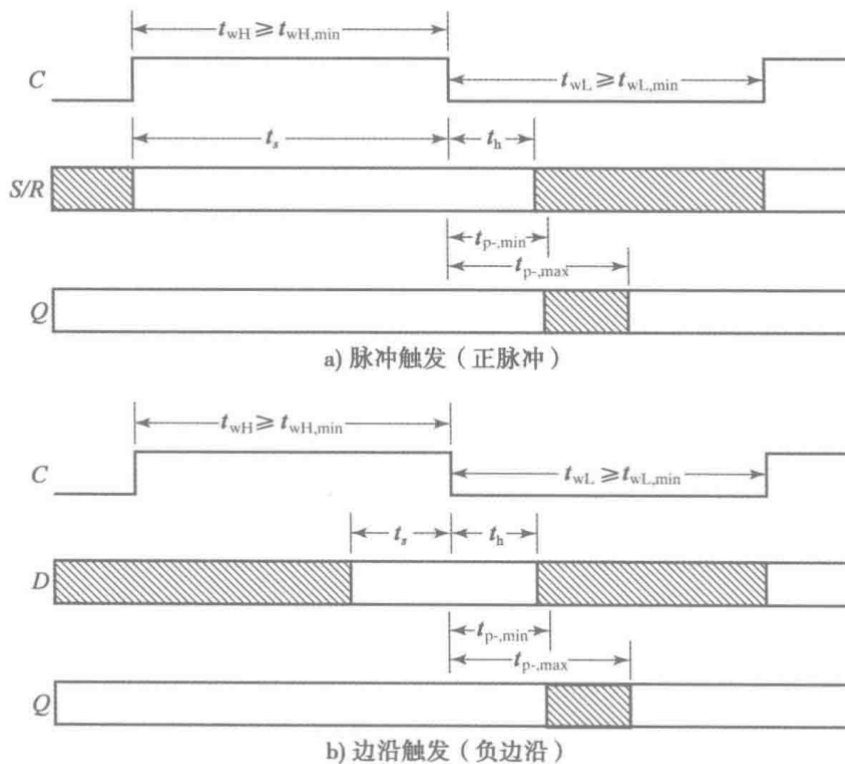


图 4-36 触发器的定时参数

- t_s 为 setup time, 建立时间, 即需要维持输入一段时间不变以杜绝错误变化。
- t_h 为 hold time, 保持时间, 即需要保持一段时间不变以保证锁存器的值不会响应输入而变化。
- t_w 为 clock pulse width, 时钟周期的最小值, 不得小于这个值。

时序电路Timing

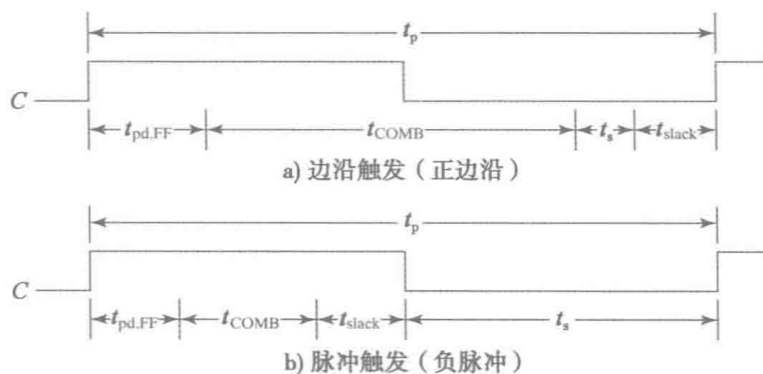


图 4-37 时序电路的定时参数

t_{COMB} 为组合逻辑的延时, t_{slack} 为传播路径上需要的额外时间, $t_{pd,FF}$ 为触发器延时。

因此有：

$$t_p \geq \max(t_{pd,FF} + t_{COMB} + t_s)$$

为最短时钟周期。

异步输入与亚稳态

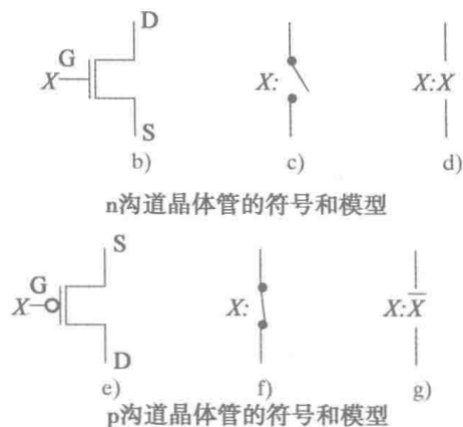
书上说的清楚。

Ch5

CMOS电路

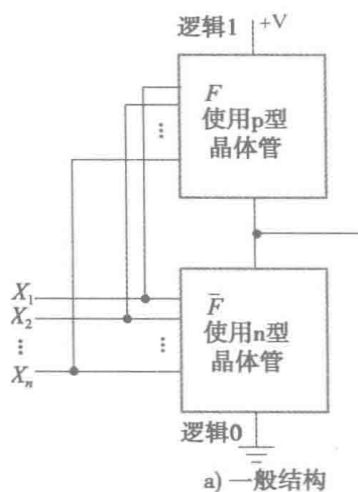
CMOS晶体管

N型和P型



完全互补的CMOS电路

计概中曾讲过，p型晶体管不能直接接地，否则会导致输出电平不是二值，这就排除了直接p接地来构造逻辑门的可能。



给出逻辑表达式 $f(X_1, X_2, \dots, X_n)$ 构造晶体管级电路方法：

- 直接用n型实现 $f(X_1, X_2, \dots, X_n)$ ，再用p型实现 \bar{f} ，后接NOT即可。

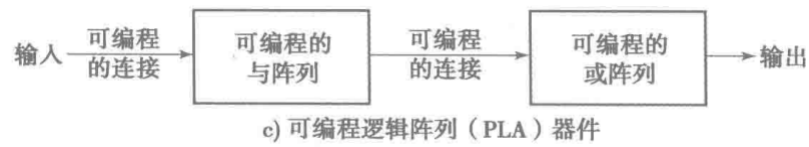
可编程实现技术 | Programmable Implementation Technology

可编程逻辑部件

- PLD
 - PROM | Programmable Read Only Memory



- PLA | Programmable Logic Array

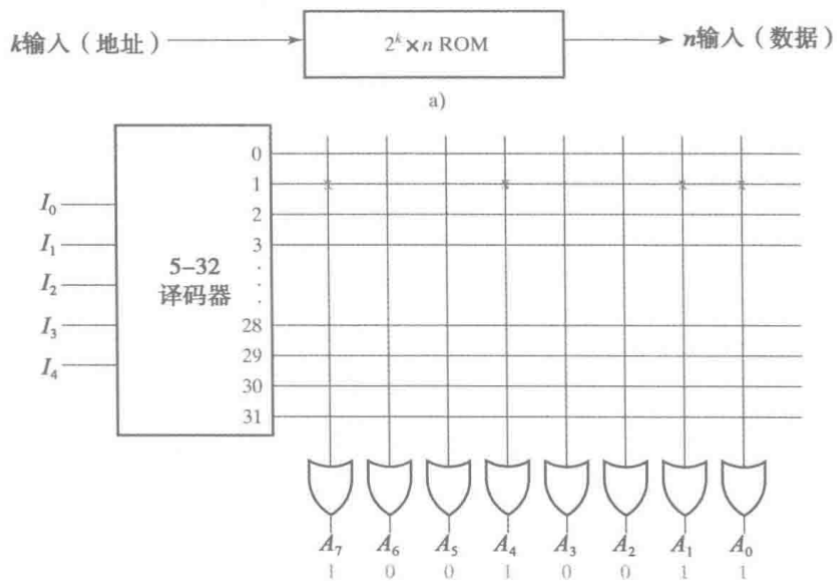


◦ PAL | Programmable Array Logic



ROM

$m \times n$ ROM是指 m 个地址, n 位宽的数据的一块ROM。



Ch6

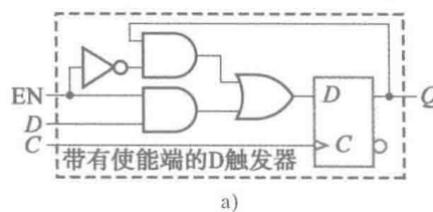
寄存器

Registers with Clock Gating



产生的问题：有门控的时钟和没有门控的时钟之间可能产生时钟偏移。

Registers with Load-Controlled Feedback



前面的部分实际上是2-1 Mux

寄存器传输

微操作

算术微操作

符号名称	描 述
$R0 \leftarrow R1 + R2$	寄存器 $R1$ 的值与寄存器 $R2$ 的值相加，结果传输给 $R0$
$R2 \leftarrow \overline{R2}$	对寄存器 $R2$ 的值求反码
$R2 \leftarrow \overline{R2} + 1$	对寄存器 $R2$ 的值求补码
$R0 \leftarrow R1 + \overline{R2} + 1$	寄存器 $R1$ 的值加上寄存器 $R2$ 值的补码，结果传输给 $R0$
$R1 \leftarrow R1 + 1$	寄存器 $R1$ 的值加 1
$R1 \leftarrow R1 - 1$	寄存器 $R1$ 的值减 1

逻辑微操作

符号名称	描 述
$R0 \leftarrow \overline{R1}$	逻辑按位取反
$R0 \leftarrow R1 \wedge R2$	逻辑按位与
$R0 \leftarrow R1 \vee R2$	逻辑按位或
$R0 \leftarrow R1 \oplus R2$	逻辑按位异或

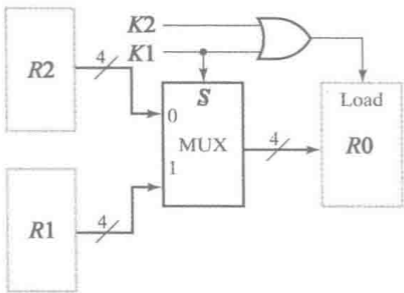
移位微操作

类型	符号名称	8 位数据示例	
		源寄存器 $R2$	移位后：目的寄存器
左移	$R1 \leftarrow sl\ R2$	10011110	00111100
右移	$R1 \leftarrow sr\ R2$	11100101	01110010

Mux-Based Transfer

以接受两个寄存器传输为例

$K1 : R0 \leftarrow R1, \overline{K1} K2 : R0 \leftarrow R2$



类似地，接受n个寄存器传输，

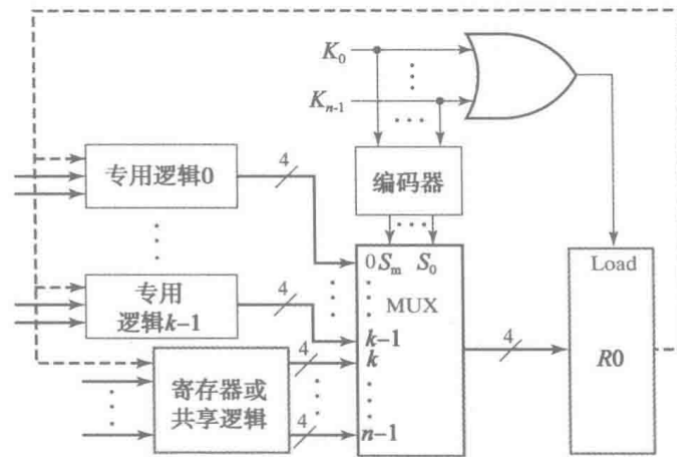
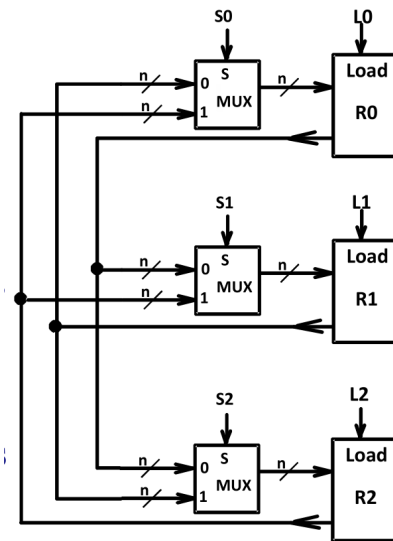
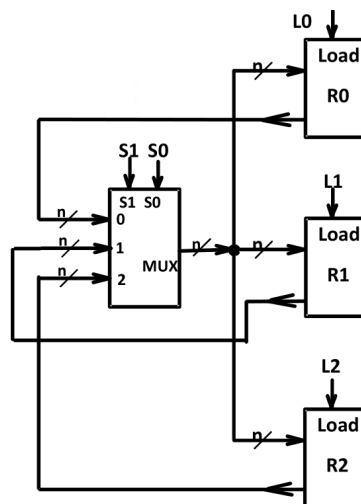


图 6-8 具有 n 个源的多路复用器的概括

Dedicated (专用) Mux-Based

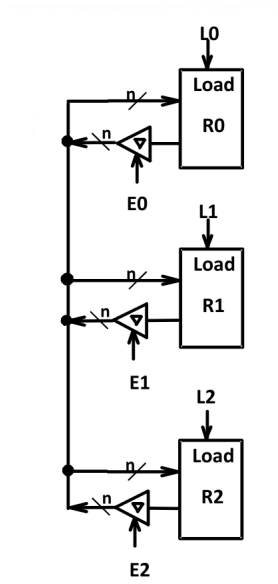


Mux Bus

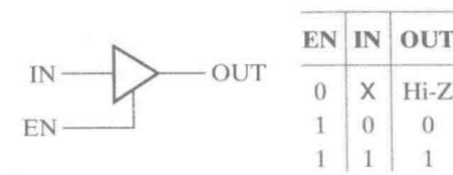


基于总线的Mux可以替换为三态门

Three State Bus



三态门的特性如下图



在使用三态门进行传输时，要保证 $E0$, $E1$, $E2$ 不能同时为1，否则传输会出现问题。解决方案是Enable使用译码器来保证每次只有一个被选中。

移位寄存器 & 计数器

书或ppt