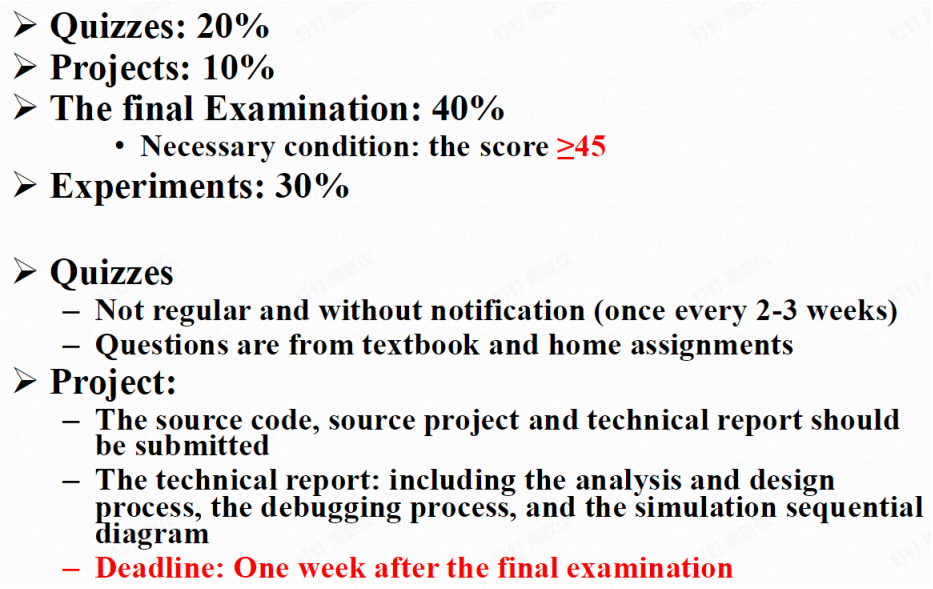
Logic and Computer Design Fundamentals 数字与逻辑设计

> Tip



> Note

卡诺图、状态图、decoder、encoder

SR，JK filp-flop 是啥

verilog

1. Digital Systems and Information

数字系统与信息

1.1 Digital System

Digital System: Takes a set of discrete information inputs and discrete internal information (system state) and generates a set of discrete information outputs.

Synchronous Sequential System 同步时序电路：状态在离散的时间点更新

Asynchronous Sequential System 异步时序电路：状态在任何时间点更新

> Note

现代计算机基本都选择同步！异步的错误率很高

> 1/主频 = 周期 是规定的更新时间间隔

A Digital Counter 计数器是异步电路

AD 转换：模拟-> 数字

Two level, or binary values are the most prevalent values in digital systems

1.2 Number System

数字系统表示

The string of digits represents the power series:

$$(\text{Number})_r = \left(\sum_{i=0}^{i=n-1} A_i \cdot r^i\right) + \left(\sum_{j=-m}^{j=-1} A_j \cdot r^j\right)$$

$$(\text{Integer Portion}) + (\text{Fraction Portion})$$

二进制补码 = 反码+1，则二进制减法可以用 `被减数+减数的补码` 来算（最后把溢出的高位舍去）

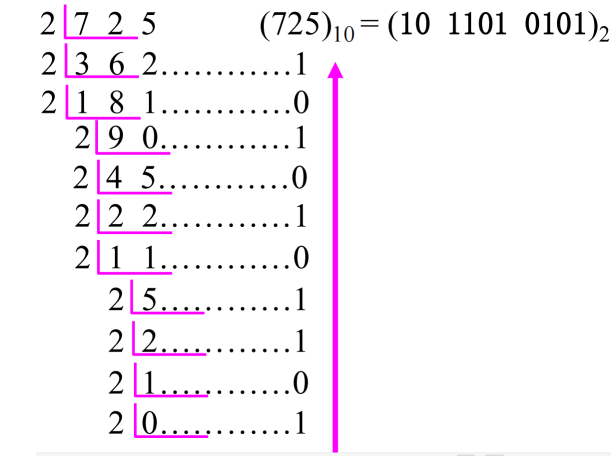
1.2.1 进制转化

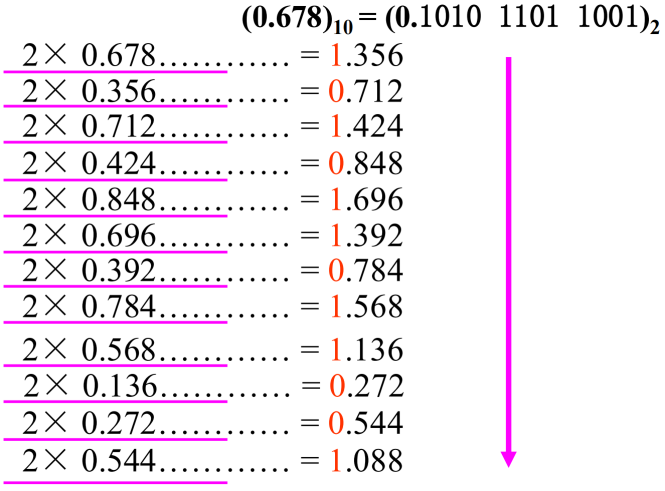
To convert from one base to another:

1. Convert the Integer Part
2. Convert the Fraction Part
3. Join the two results with a radix point

10-> 2（10进制转2进制）

- 整数部分 除二取余，从下往上得到结果
- 小数部分 乘二取整，从上往下得到结果





2-> 10：$\sum$ (digit × respective power of 2)

8 <-> 16：通过 2 进制转换

1.2.2 Binary Coded Decimal (BCD)

二进制编码的十进制数：The BCD code is the 8,4,2,1 code

$13_{10} = 1101_2$ (This is conversion)

13 $\Leftrightarrow$ 0001|0011 (This is coding)

BCD Arithmetic

8	**1000**	**Eight**
+5	**+0101**	**Plus 5**
13	**1101**	**is 13 (> 9)**
	+0110	**so add 6** +6和-10的效果一样
carry = 1	**0011**	**leaving 3 + cy**
0001	**0011**	**Final answer (two digits)**

1.3 Coding

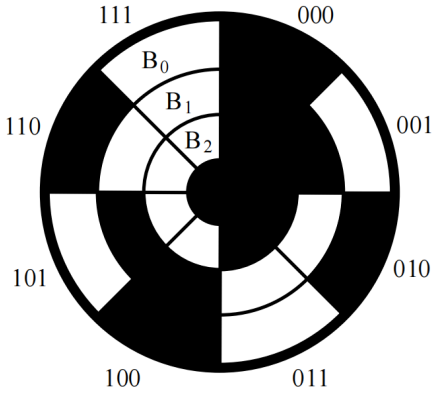
1.3.1 PARITY BIT Error-Detection Codes

一种简单的冗余形式是奇偶校验 parity，即在代码字上附加一个额外的位，以使 1 的数量为奇数或偶数。奇偶校验可以检测所有单比特错误和一些多比特错误。

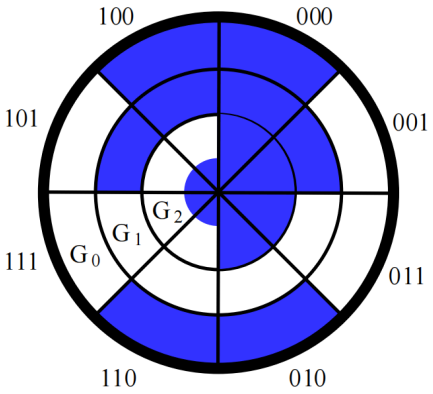
1.3.2 GRAY CODE

格雷码：相邻数之间只有一个 bit 不一样

- **An Example: Optical Shaft Encoder**



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

计算方法：错开一位异或可得

2. Combinational Logic Circuits

组合逻辑电路

2.1 Gate Circuits and Boolean Equations

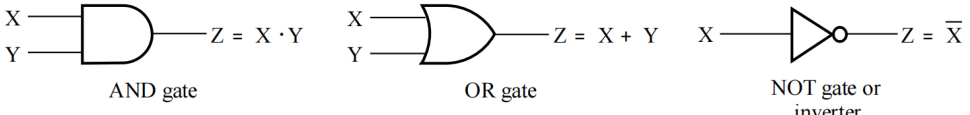
2.1.1 Binary Logic and Gates

二值逻辑和逻辑门

	英语	符号表示
与	AND	× 或者 ∧
或	OR	+ 或者 ∨
非	NOT	$\overline{\quad}$

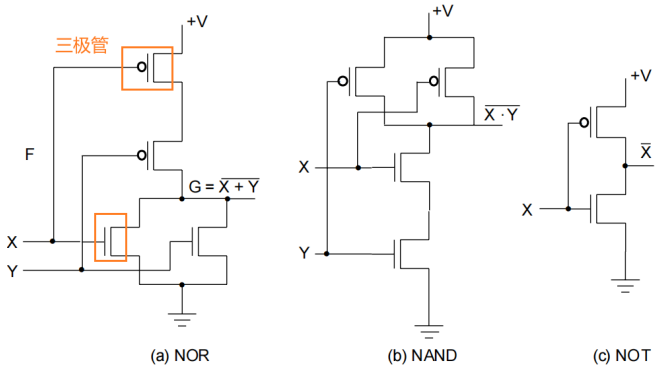
真值表 Truth table

- **Logic gates have special symbols:**



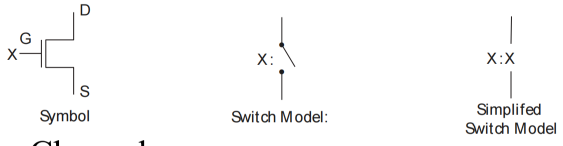
(a) Graphic symbols

MOS Transistor

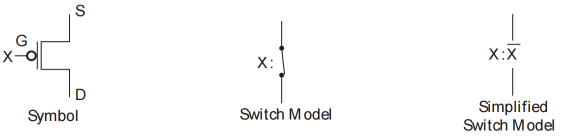


Switch models

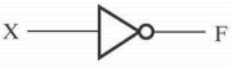
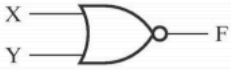
- n-Channel – Normally Open (NO) Switch Contact



- p-Channel – Normally Closed (NC) Switch Contact



- NO = 0, NC = 1

Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
AND		$F = XY$	X Y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = X + Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT (inverter)		$F = \overline{X}$	X | F 0 | 1 1 | 0
NAND		$F = \overline{X \cdot Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y$ $= X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\overline{Y}$ $= X \oplus Y$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

通用门(Universal Gate)：能够表示其他所有门的逻辑门，NAND 和 NOR 都是通用门

2.1.2 Boolean Algebra

The **dual** 对偶 of an algebraic expression is obtained by interchanging + and · and interchanging 0's and 1's.

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative
12.	$(X + Y) + Z = X + (Y + Z)$	13.	$(XY)Z = X(YZ)$	Associative
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

Unless it happens to be self-dual, the dual of an expression does not equal the expression itself.

Example: $F = (A + \overline{C}) \cdot B + 0$

dual $F = (A \cdot \overline{C} + B) \cdot 1 = A \cdot \overline{C} + B$

Example: $G = X \cdot Y + (\overline{W + Z})$

dual G =

Example: $H = A \cdot B + A \cdot C + B \cdot C$ self-dual

dual H = (A+B)(A+C)(B+C)=(A+AC+AB+BC)=(A+BC)(B+C)=AB+AC+BC

> **Note**
>
> 公式（两边取 dual 也成立）
>
> - $x \cdot y + \overline{x} \cdot y = y$ $(x + y)(\overline{x} + y) = y$ **Minimization**
> - $x + x \cdot y = x$ $x \cdot (x + y) = x$ **Absorption**
> - $x + \overline{x} \cdot y = x + y$ $x \cdot (\overline{x} + y) = x \cdot y$ **Simplification**
> - $x \cdot y + \overline{x} \cdot z + y \cdot z = x \cdot y + \overline{x} \cdot z$ **Consensus**
>
> $(x + y) \cdot (\overline{x} + z) \cdot (y + z) = (x + y) \cdot (\overline{x} + z)$
>
> - $\overline{x + y} = \overline{x} \cdot \overline{y}$ $\overline{x \cdot y} = \overline{x} + \overline{y}$ **DeMorgan's Laws**

Example

$$L = AB + A\overline{C} + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B + ADE(F+G)$$

$$L = A\overline{\overline{B}C} + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B + ADE(F+G) \quad \text{DeMorgan Laws}$$

$$= A + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B + ADE(F+G) \qquad A + \overline{A}B = A + B$$

$$= A + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B \qquad A+AB=A$$

$$= A + \overline{B}C(D + \overline{D}) + \overline{C}B + \overline{B}D + \overline{D}B(C + \overline{C}) \qquad A + \overline{A} = 1$$

$$= A + \overline{B}CD + \overline{B}C\overline{D} + \overline{B}\overline{C} + \overline{B}D + \overline{D}BC + \overline{D}B\overline{C} \quad \text{Distributive Laws}$$

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

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

$$= A + C\overline{D} + B\overline{C} + \overline{B}D \qquad A + \overline{A} = 1$$

互补函数(Complementing Functions): Use DeMorgan's Theorem to complement a function:

1. Interchange AND and OR operators
2. Complement each constant value and literal

Example: Complement F = $\overline{x}\ y\ \overline{z} + x\ \overline{y}\ \overline{z}$; $\overline{F} = (x + \overline{y} + z)(\overline{x} + y + z)$

一致性定理(Consensus Theorem)：
$XY + \overline{X}Z + YZ = XY + \overline{X}Z + (X + \overline{X})YZ = XY + \overline{X}Z$

2.1.3 Standard Forms

Canonical Forms 标准型

Minterms are AND terms with every variable present in either true or complemented form.

最小项：逻辑值为 1 的乘积项。所有变量都以源变量或者反变量的形式出现，且仅出现一次的乘积项。其特征是在真值表中仅仅表示二进制变量的一个组合，而且对于那种组合其值为 1，对于其他组和其值为 0

- 对于 n 个变量，一共有 2^n个不同的最小项
- 每个项都要包含所有变量

最大项：**Maxterms** are OR terms with every variable in true or complemented form.

二变量 X、Y 的 4 个最小项和 4 个最大项

- **Examples: Two variable minterms and maxterms.**

Index	Minterm	Maxterm
0	$\overline{x}\,\overline{y}$	$x + y$
1	$\overline{x}\,y$	$x + \overline{y}$
2	$x\,\overline{y}$	$\overline{x} + y$
3	$x\,y$	$\overline{x} + \overline{y}$

- **The index above is important for describing which variables in the terms are true and which are complemented.**

- 最大项和最小项之间是互补(complemented)的关系，即 $\overline{m_j} = M_j$

最小项之和(SOM, sum of minterm)

- If $F = m_0+m_1+m_7 = \sum m_i$, then $F = M_2M_3M_4M_5M_6$
- 最小项序号 Index：$x\ \overline{y}\ z = 101 = 5$
- 挑出真值表中所有结果是 `1` 的最小项

最大项之积(POM)

- 挑出所有结果为 `0` 的最大项

Standard Sum-of-Products (SOP)form 积之和 ：equations are written as **an OR of AND terms**

Standard Product-of-Sums (POS) form 和之积 ：equations are written as **an AND of OR terms**

Examples:

- **SOP:** $A\,B\,C + \overline{A}\,\overline{B}\,C + B$
- **POS:** $(A+B)\cdot(A+\overline{B}+\overline{C})\cdot C$

These "mixed" forms are <u>neither SOP nor POS</u>

- $(A\,B + C)\,(A + C)$
- $A\,B\,\overline{C} + A\,C\,(A + B)$

- POS 最外面是乘法，括号里面+连起来的单项不能是两项

> **Note**
>
> SOP：化简得
>
> POS：计算反函数（求其非），再用德摩根

2.2 Circuit Optimization

2.2.1 Two-Level Optimization

2.2.1.1 cost criteria

Literal cost (L) 文字成本： the number of literal appearances in a Boolean expression corresponding to the logic circuit diagram

Gate input cost (G)门输入成本：the number of inputs to the gates in thcose implementation corresponding exactly to the given equation or equations.

> Tip
>
> 门输入成本 = 全部文字数 + 除单个文字之外的全部项数 + 不同取反值的单个文字总数

Gate input cost with NOTs (GN)：把取反也算作操作

Example 2:

$F = A\,B\,C + \bar{A}\,\bar{B}\,\bar{C}$

L = 6 G = 8 GN = 11

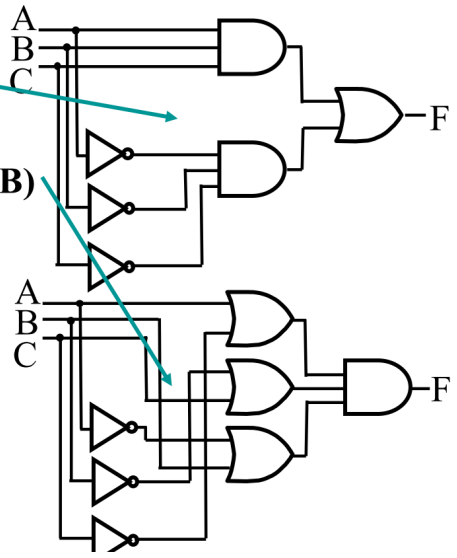
$F = (A + \bar{C})(\bar{B} + C)(\bar{A} + B)$

L = 6 G = 9 GN = 12

Same function and same literal cost

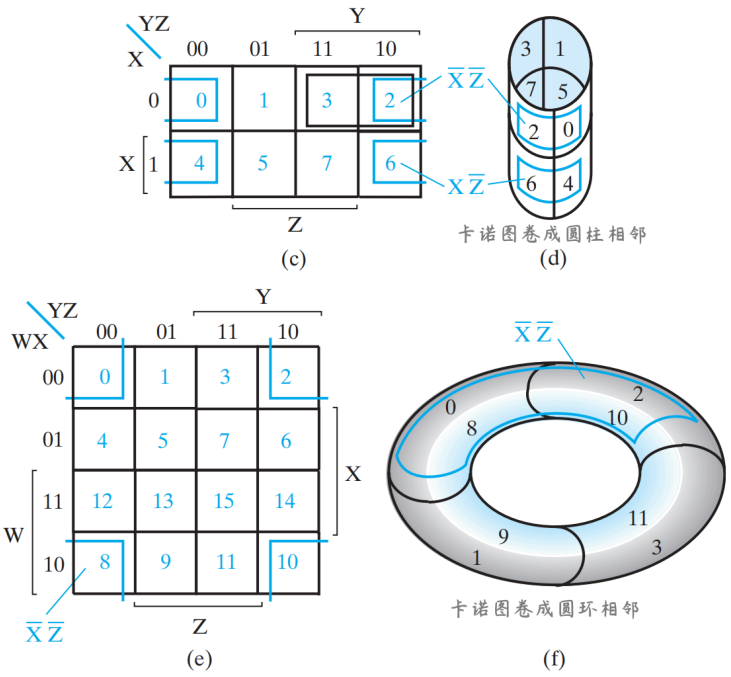
But first circuit has better gate input count and better gate input count with NOTs

Select it!



2.2.2 Karnaugh Maps (K-map)

卡诺图

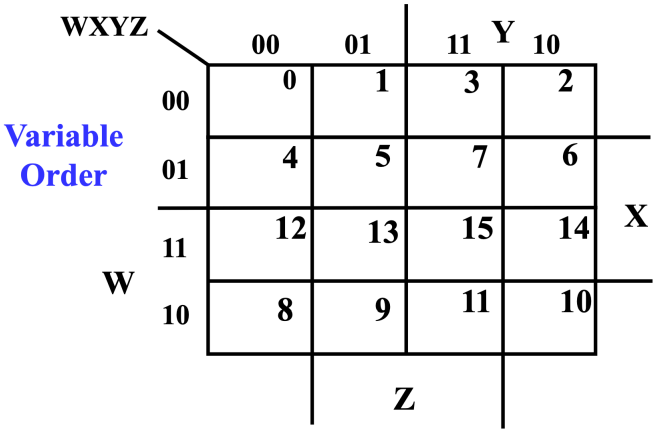


- 当表示函数最小项的两个方格有相同边缘，这些方格就可以组合到一起形成一个少了一个变量的乘积项
- 对于三变量、四变量的卡诺图，要几何上理解共享边缘
  - eg, 4✕4 的 4 个边角可以合并成一个矩形

2.2.2.1 二变量卡诺图

2.2.2.2 三变量卡诺图

2.2.2.3 四变量卡诺图



Four variable maps can have rectangles corresponding to:
• A single 1 = 4 variables, (i.e. Minterm)
• Two 1s = 3 variables,
• Four 1s = 2 variables
• Eight 1s = 1 variable,
• Sixteen 1s = zero variables (i.e.Constant "1")

目标:

1. Find all prime implicants 标 1
2. Include all essential prime implicants in the solution 找到最大的矩形覆盖尽可能多的 1
3. Select a minimum cost set of non-essential prime implicants to cover all minterms not yet covered
4. Minimize the overlap among prime implicants as much as possible. 减少重叠
5. 得到的矩形数量就是简化后布尔函数的项数（积之和）

> **Note**
>
> 合并 1 = 合并 0
>
> 最后划分的结果可能不唯一

主蕴涵项（prime）：卡诺图中的 **极大** 蕴含项

质主蕴含项（又称必要蕴涵项 essential）：质主蕴含项是包含只被它（基本主蕴含项）覆盖的 `1` 的主蕴含项

> 对于任意函数，主蕴含项一定存在，但是质主蕴含项不一定存在

无关最小项(don't care condition)：函数中没有指定的最小项，在卡诺图中用“×”表示

- 包含无关最小项的矩阵划分结果可能不唯一

2.2.3 Multiple-Level Optimization

不好用卡诺图

提取公因式降门输入成本

Algebraic Factoring

$F = \overline{A}\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C} + ABC + AC\,\overline{D}$ G = 16

- **Factoring:**

$F = \overline{A}\,(\overline{C}\,\overline{D} + B\overline{C}) + A\,(BC + C\,\overline{D})$ G = 18

- **Factoring again:**

$F = \overline{A}\,\overline{C}(B + \overline{D}) + AC\,(B + \overline{D})$ G = 12

- **Factoring again:**

$F = (\overline{A}\,\overline{C} + AC)\,(B + \overline{D})$ G = 10

2.3 Additional Gates and Circuits

2.3.1 Other Gate Types

primitive gate: 一次操作 AND, OR + an inversion

complex gate: 不止一次操作

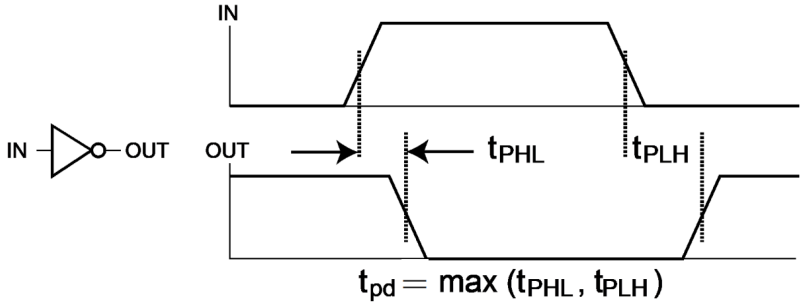
High-Impedance Outputs

2.3.2 Propagation Delay

High-to-low (HL) and low-to-high (LH) transitions are defined with respect to the output, not the input. 输出从高-> 低 / 低-> 高

传输延迟（propagation delay）：输出响应输入的变化

惯性延迟（inertial delay）：类似传输延迟，但如果输入变化使输出在一个小于拒绝时间（rejection time）的间隔内发生两次变化，那么两次变化中的第一次将不会发生



传输延迟 $t_{pd} = \max(t_{PHL}, t_{PLH})$

2.3.3 Exclusive-OR Operator and Gates

XOR 异或：复杂门，相同出 0、不同出 1

- $X \oplus Y = X\overline{Y} + \overline{X}Y$

XNOR 同或（异或非）：复杂门，相同出 1、不同出 0

- $\overline{X \oplus Y} = XY + \overline{X}\overline{Y}$

The XOR identities:

$$X \oplus 0 = X \qquad X \oplus 1 = \overline{X}$$
$$X \oplus X = 0 \qquad X \oplus \overline{X} = 1$$
$$X \oplus Y = Y \oplus X$$
$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$
$$X \oplus \overline{Y} = \overline{X \oplus Y}$$

$$X \oplus Y \oplus Z = \overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ$$

多变量异或运算又被称为奇函数（odd function）

奇函数的反函数称为偶函数（even function）

3. Combinational Logic Design

3.1 Implementation Technology and Logic Design

3.1.1 Design Procedure

Hierarchical Design 分层设计

> **Note**
>
> **真值表推导出逻辑表达式的步骤**：
>
> 1. **确定输出为 1 的行**：对于每个输出信号（如 S0, S1, ..., S5），找出所有 **输出为 1 的行**
> 2. **写出对应的乘积项**：对于每个输出为 1 的行，根据 **输入变量**（A, B, C）的值写出一个乘积项。如果变量值为 1，则直接写变量；如果为 0，则写变量的非
> 3. **将所有乘积项相加**：将这些乘积项用逻辑或（+）连接起来，得到该输出的逻辑表达式
> 4. **可能的化简**：使用卡诺图

3.1.2 Combinational Logic

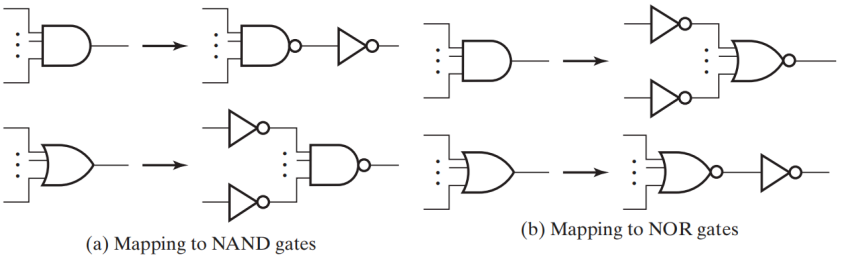
宽线

3.1.3 Enable 使能

使能 EN：允许/阻止输入 X 到达输出

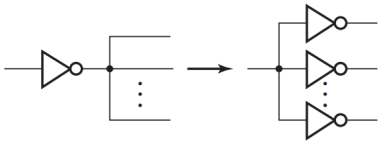
3.2 Technology mapping

工艺映射

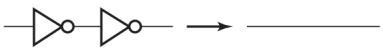


(a) Mapping to NAND gates

(b) Mapping to NOR gates



(c) Pushing an inverter through a “dot”



(d) Canceling inverter pairs

3.3 Decoder 译码

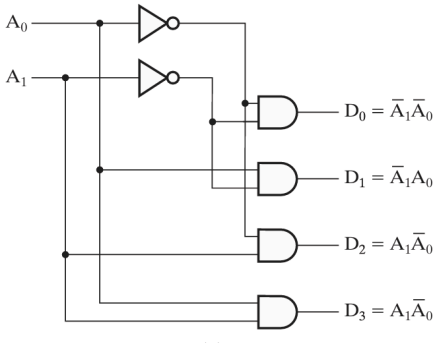
the conversion of an n-bit input code to an m-bit output code with $n \le m \le 2^n$ such that each valid code word produces a unique output code

译码就是将一个 n 位的输入码转化成一个 m 位的输出码，decoder 出来是最小项

2-4 译码器：

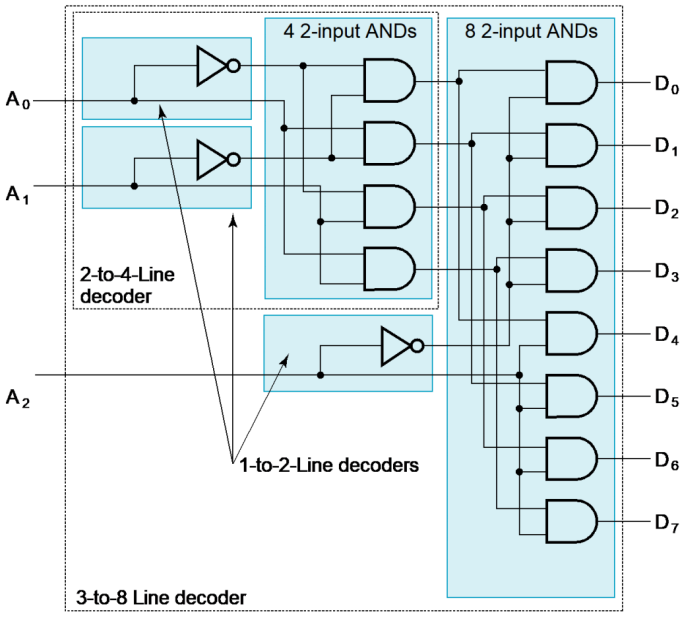
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)



(b)

3.3.1 n-m 译码器



3-8 译码器 = 1 个 2-4 译码器+1 个 1-2 译码器

- 门输入成本 = 3 + 2×4 + 2×8 = 27

1 个 2-4 译码器 = 2 个 1-2 译码器

- 门输入成本 = 2 + 2×4 = 10

如何构造译码器?

1. 使 k = n。
2. 如果 k 是偶数，则将 k = k/2，并使用 2^k 个与门，这些门被两个译码器驱动，每个译码器有 $2^k/2$ 个输出。如果 k 是奇数，计算出(k+1)/2 和(k-1)/2，并使用 2 个与门，这些与门被两个译码器驱动，其中一个译码器有 $2^{(k+1)/2}$ 个输出，另一个译码器有 $2^{(k-1)/2}$ 个输出。
3. 对于由步骤 2)得到的每一个译码器，使用由步骤 2)得到的 k，重复步骤 2)，直到 k = 1。如果 k = 1，则使用一个 1-2 译码器。

Eg,6-64 译码器：两个 3-8 译码器

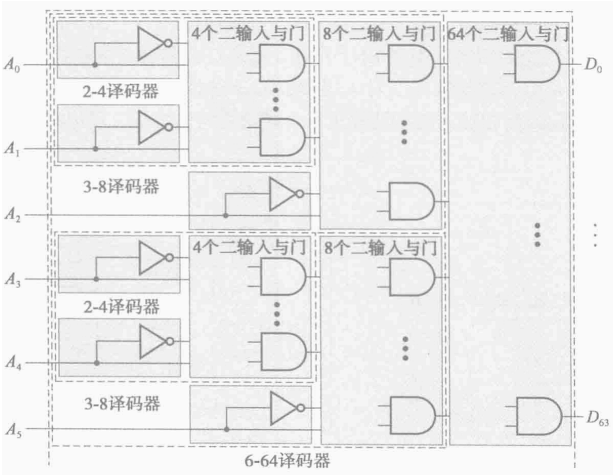


图 3-15　6-64 译码器

门输入成本 = 6+2(2×4)+2(2×8)+2×64 = 182

4-16 译码器？。 = 4+ 2(2×4) + 2×16 =

3.3.2　Decoder with Enable

n-m decoder 输出端连接 m 个使能电路（使能信号 EN），能够控制连接

demultiplexer 多路分配器

3.3.3　基于译码器的组合电路

combinational logic implementation - Decoder and OR gates

任何 n 输入 m 输出的组合电路都可以用 1 个 n-2^n译码器和 m 个或门实现

全加器：3 位输入，输出 sum 和进位 carry（3-8 译码器）

3.4　Encoder 编码

2^n输入，n 输出

优先编码器 priority encoder

- 可以处理多输入，按照优先级依次处理

3.5　Selection

多路复用器：组合电路从多条输入中选择一个输入，并将信息直接传输到输出

- 逻辑表达式就是最小项：Y = Σm(0,1,2,….) [其中每个最小项对应Ii]

2-to-1-line multiplexer：

- 1 个 1-2 decoder + 两个使能电路 EN + 1 个两输入或门
- $Y = \overline{S}I_0 + SI_1$（S 是选择输入，I 是输出）
-

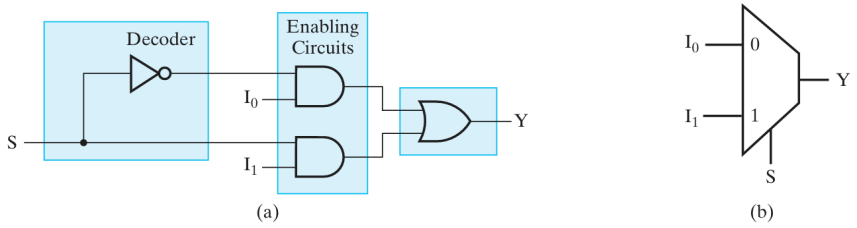


FIGURE 3-24
(a) Single-Bit 2–to–1-Line Multiplexer; (b) common Symbol for a Multiplexer

4-to-1-line multiplexer：

- 1 个 2-4 decoder + 4 个 与门 + 1 个四输入或门（1 个 4×2 与或门） --> cost = 2 + 2(4×2)+ 4 = 22
- $Y = \overline{S_1}\overline{S_0}I_0 + \overline{S_1}S_0I_1 + S_1\overline{S_0}I_2 + S_1S_0I_3$
-

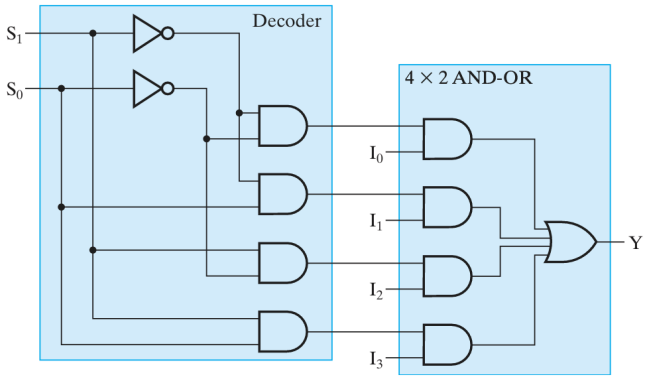
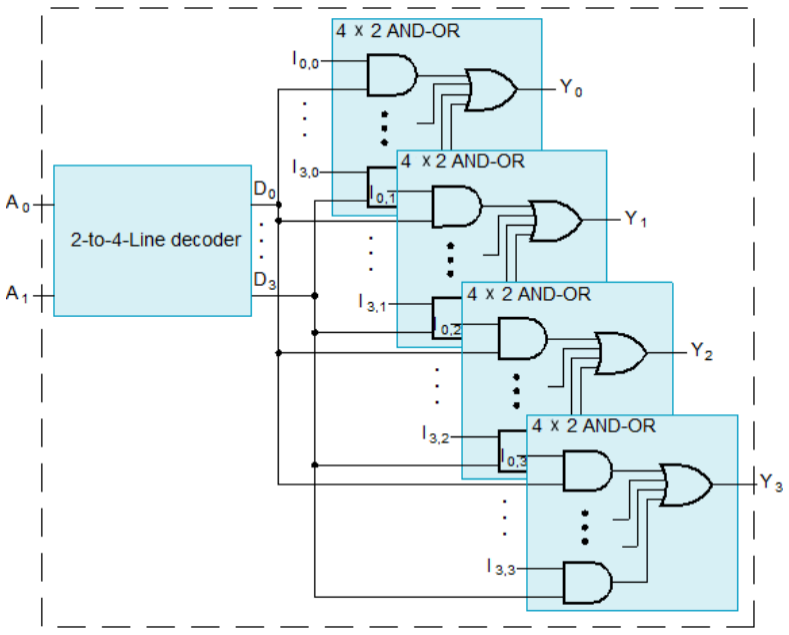


FIGURE 3-25
A Single-Bit 4–to–1-Line Multiplexer

64-to-1-line multiplexer：1 个 6-64 译码器+1 个 64×2 的与或门 --> cost = 182 + 128 + 64 = 374

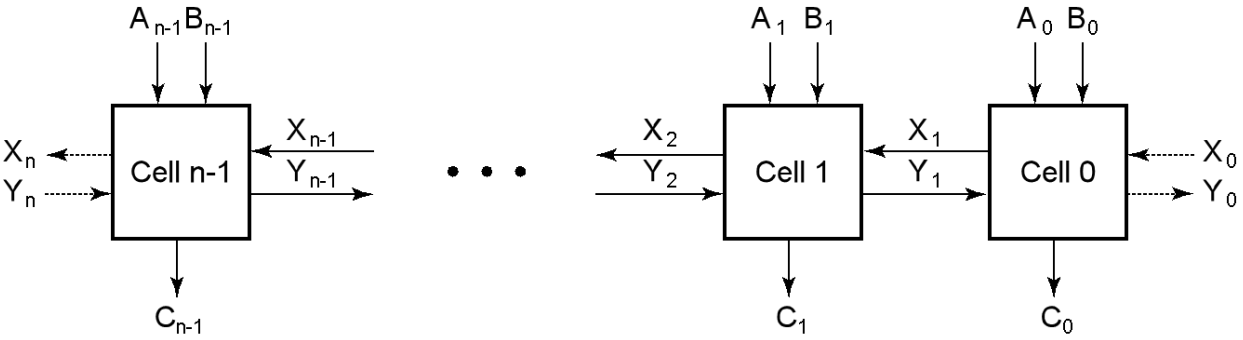


若有 n 个变量，找到 Truth tables 的最小项，

BCD 码转化为 7 段码的译码器，用 8-1 多路复用器

3.6 Arithmetic Functions

3.6.1 Iterative Combinational circuits 迭代组合电路



数字 $A_nA_{n-1}\dots A_1A_0 + B_nB_{n-1}\dots B_1B_0$

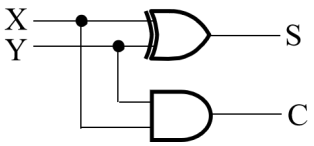
3.6.2 Adder 加法器

3.6.2.1 半加器（两个输入、两个输出）

$S = X \oplus Y$

$C = XY$

半加器由一个异或门，一个与门组成，两位二进制加法



3.6.2.2 全加器（三个输入、两个输出）

$S = X \oplus Y \oplus Z = \overline{XY}Z + \overline{X}Y\overline{Z} + X\overline{YZ} + XYZ$

$C = XY + (X \oplus Y)Z = XY + XZ + YZ$

$X \cdot Y$ is carry generate(G), $X \oplus Y$ is carry propagate(P)

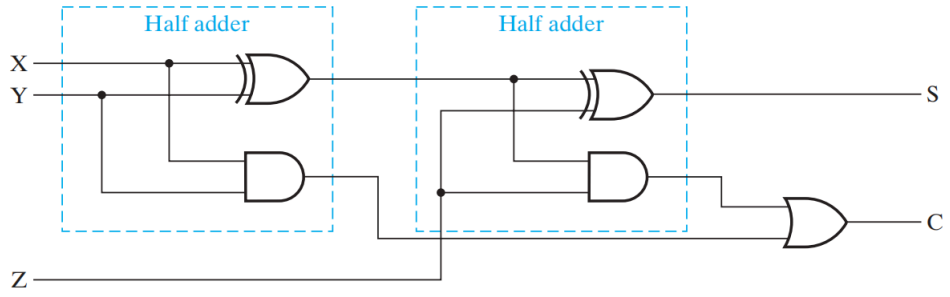


FIGURE 3-42
Logic Diagram of Full Adder

全加器由两个半加器 + 一个或门组成，三位二进制加法

3.6.2.3 Carry Lookahead Adder

C4 = G3 + P3G2 + P3P2G1+ P3P2P1G0 + P3P2P1P0 C0 = G0~3+ P0~3C0

行波进位加法器 ripple carry adder

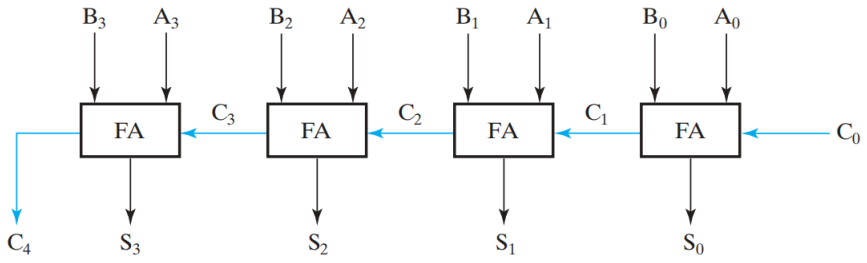
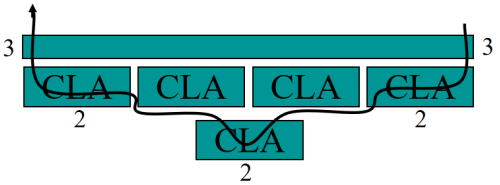


FIGURE 3-43
4-Bit Ripple Carry Adder

延迟

- **Specifications:**
  - **16-bit CLA**
  - **Delays:**
    - **NOT = 1**
    - **XOR = Isolated AND = 3**
    - **AND-OR = 2**



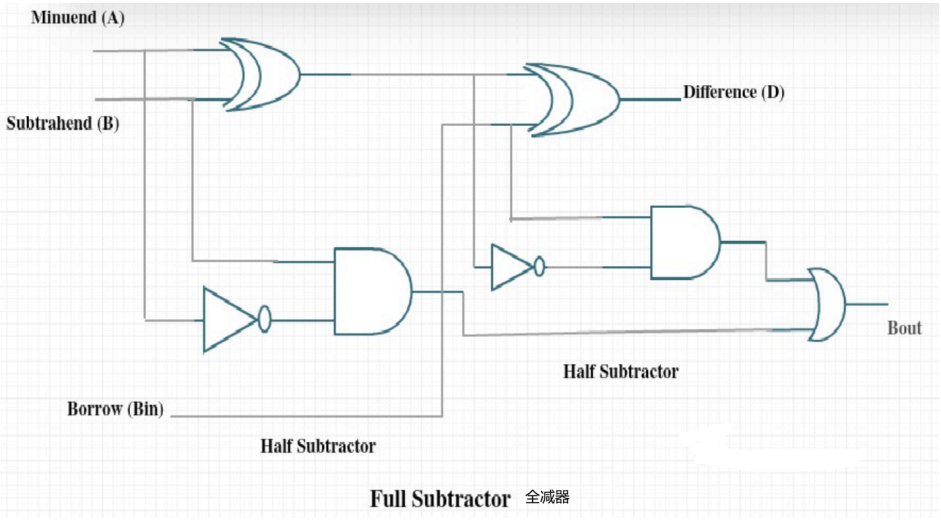
- **Longest Delays:**
  - **Ripple carry adder* = 3(xor) + 15 × 2(an-or) + 3(xor) = 36**
  - **CLA = 3 + 3 × 2 + 3 = 12**

3.6.3 Subtracter 减法

对于 n < 0, 2's complement 补码 = 反码（对除了符号位取反）+1

对于 n > 0, 补码 = 反码 = 源码

补码再求一次补码就是源码



3.6.4 Overflow

overflow 检查符号位，和第一个操作数一致则没有溢出

用 V 表示溢出

Extension

3.6.5 ALU 全加器

$S_i = A_i \oplus B_i \oplus C_i$

4. Sequential Circuits

时序电路

组合电路是时序电路的一部分

4.1 Storage Elements and Sequential Circuit Analysis

4.1.1 Introduction

存储组合电路的状态 state

next state = f(input, state)

outputs(Mealy 米利模型) = g(inputs, state) 显示表达 inputs，即使 state 不变， inputs 变化就会变化

outputs(Moore 摩尔模型)= h(state) 隐含 inputs，只有状态变化才输出变化

- 计算机大多采用 Moore 形式

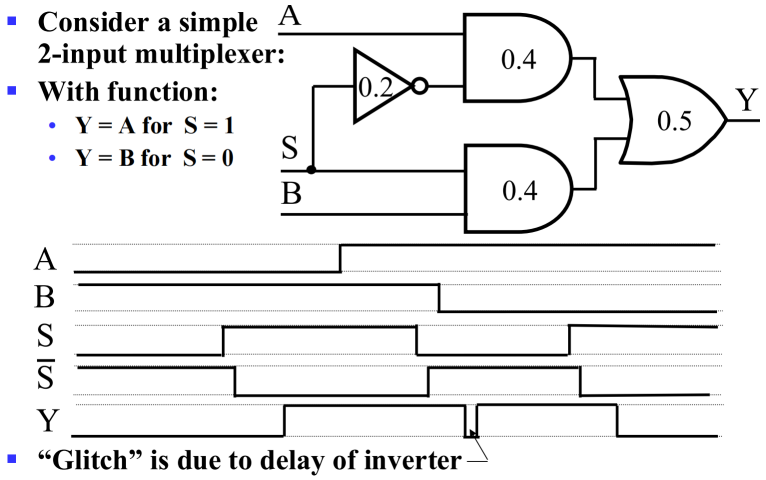
同步时序电路 synchronous：离散时间点，时钟锁定

异步时序电路 asynchronous：即时变化，如果时钟也作为输入的一部分，就是异步

4.1.2 types

gate delay

- transition time：从打算输出到最终输出的时间
- propagation delay：输入到输出端到端的延迟



Glitch 的形成原因：非门的延迟 propagation delay of the inverter (NOT gate)

> Here's a breakdown of why:
>
> **Multiplexer Function:** The circuit is designed so that when the select signal S is 1, Y = A, and when S is 0, Y = B.
>
> **Signal Transition:** The glitch happens when the select signal S transitions from 1 to 0. At this specific moment in the timing diagram, both inputs A and B are high (1).
>
> **Ideal vs. Real Behavior:** Ideally, when S changes from 1 to 0, the output Y should switch from following A to following B. Since both A and B are high, Y should remain high.
>
> **Inverter Delay:** However, the inverter which generates S^- (the inverted version of S) has a delay (indicated as 0.2 time units in the diagram).
>
> Race Condition:
>
> When S goes from 1 to 0:
>
> - The signal S = 0 arrives quickly at the input of the top AND gate, disabling it.
> - The signal S^- remains 0 for a short time (0.2 units) due to the inverter's delay before it becomes 1.
> - During this short delay period, *both* S and S^- are effectively 0 at the inputs of their respective AND gates.
>
> **Output Drops:** With both AND gates receiving a 0 on their select inputs (S = 0 for the top, S^- = 0 temporarily for the bottom), both AND gates output 0.
>
> **Glitch Formation:** Since both inputs to the final OR gate are 0, its output Y drops to 0. This creates the brief low pulse, known as a glitch.
>
> **Recovery:** Once the inverter delay passes, S^- becomes 1, enabling the bottom AND gate. Since B is 1, the bottom AND gate outputs 1, and the OR gate output Y goes back to 1.
>
> In summary, the delay in the inverter causes a brief period where neither input A nor B is selected, leading to the temporary incorrect low output (the glitch).

震荡电路 oscillator，不稳定 unstable

4.2 Storage elements

4.2.1 Latches 锁存器

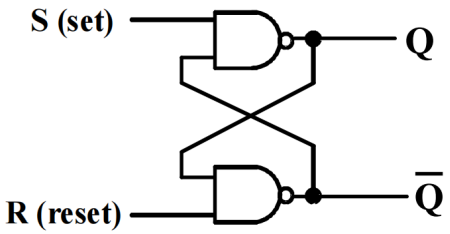
锁存器是构造触发器的基本元件；只要输入信号不变化，保持输出不变，即锁存器是透明的

Triggers 触发器可以当锁存器用，反之不行

SR 和 $\overline{SR}$ 锁存器

4.2.1.1 SR 低有效(NAND)

两个交叉耦合的 *与非门 NAND*



Time

R	S	Q	Q̄	Comment
1	1	?	?	Stored state unknown
1	0	1	0	"Set" Q to 1
1	1	1	0	Now Q "remembers" 1
0	1	0	1	"Reset" Q to 0
1	1	0	1	Now Q "remembers" 0
0	0	1	1	Both go high
1	1	?	?	Unstable!

所以S=R=0 is forbidden

输入S/R如果是0就能决定NADN门输出是1

输出 Q = 1 且 $\overline{Q}$ = 0，置位状态 set state

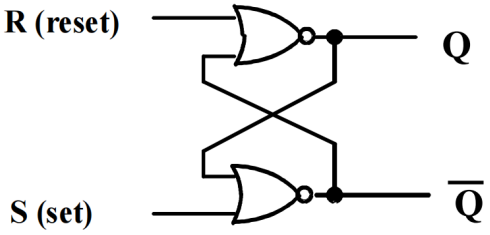
输出 Q = 0 且 $\overline{Q}$ = 1，复位状态 reset state

两个输入 S = R = 0，输出 = 1，禁止

两个输入 S = R = 1，输出 = 0，未定义状态

4.2.1.2 SR 高有效(NOR)

两个交叉耦合的 *或非门 NOR*



输入 S = 1/0，R = 0，输出 Q = 1 且 $\overline{Q}$ = 0，置位状态 set state

输入 S = 0，R = 0/1，输出 Q = 0 且 $\overline{Q}$ = 1，复位状态 reset state

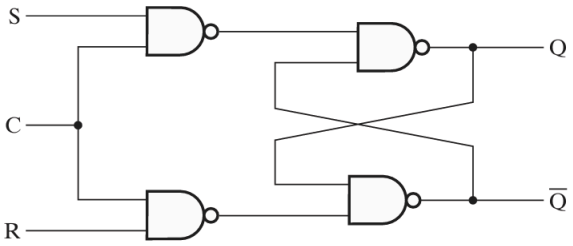
两个输入 S = R = 0，输出 = 1，未定义状态

Time ↓

R	S	Q	$\overline{Q}$	Comment
0	0	?	?	Stored state unknown
0	1	1	0	“Set” Q to 1
0	0	1	0	Now Q “remembers” 1
1	0	0	1	“Reset” Q to 0
0	0	0	1	Now Q “remembers” 0
1	1	0	0	Both go low
0	0	?	?	Unstable!

4.2.1.3 门控制 Clocked S - R Latch

增加时钟控制锁存器何时对输入敏感



(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

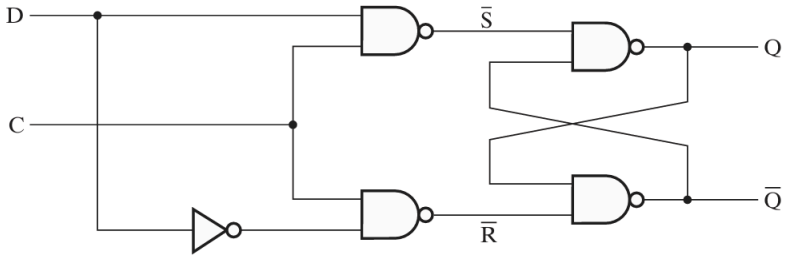
FIGURE 4-7
SR Latch with Control Input

- 不能称之为触发器，不满足触发器要求

4.2.1.4 D Latch

D 锁存器：确保输入输入信号永远不会同时取 1 以消除未定义状态，只有两个输入 D（数据信号）和 C（控制信号）

- 从 SRC 锁存器转化而来，强制要求 $S = \overline{R}$
- 只有当 C 为 `1` 时，D 锁存器才能写入数据；而当 C 为 `0` 时，D 锁存器的数据就不会变化
- 锁存器是透明的（transparent），因为当控制输入端位 1 时，从输出端可以看到数据输入端的值



(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

FIGURE 4-8
D Latch

4.2.2 触发器 Flip-flop

触发器是一个能够储存 1 位信息的二进制储存元件，在最简单的时钟控制的时序电路中使用

- 触发现象（trigger）：触发器中，输入信号值的改变可以控制内部锁存器的状态
- 触发器消除了透明性，在输出信号改变之前，输入信号和输出信号之间的通路被断开；触发器的状态只取决于前面一个瞬间的状态，不会发生多次状态改变的现象

4.2.2.1 S-R Master-Slave 主从触发器

4.2.2.2 edge-triggered 边沿触发器

边沿触发器（D 触发器）只在时钟信号跳变时触发（忽略保持阶段的时钟脉冲）

正边沿（0-> 1），负边沿（1-> 0）

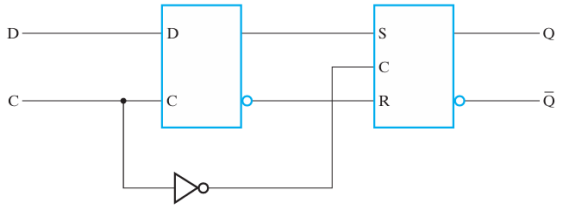


FIGURE 4-9
Negative-Edge-Triggered D Flip-Flop

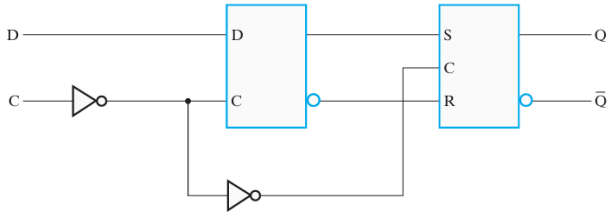
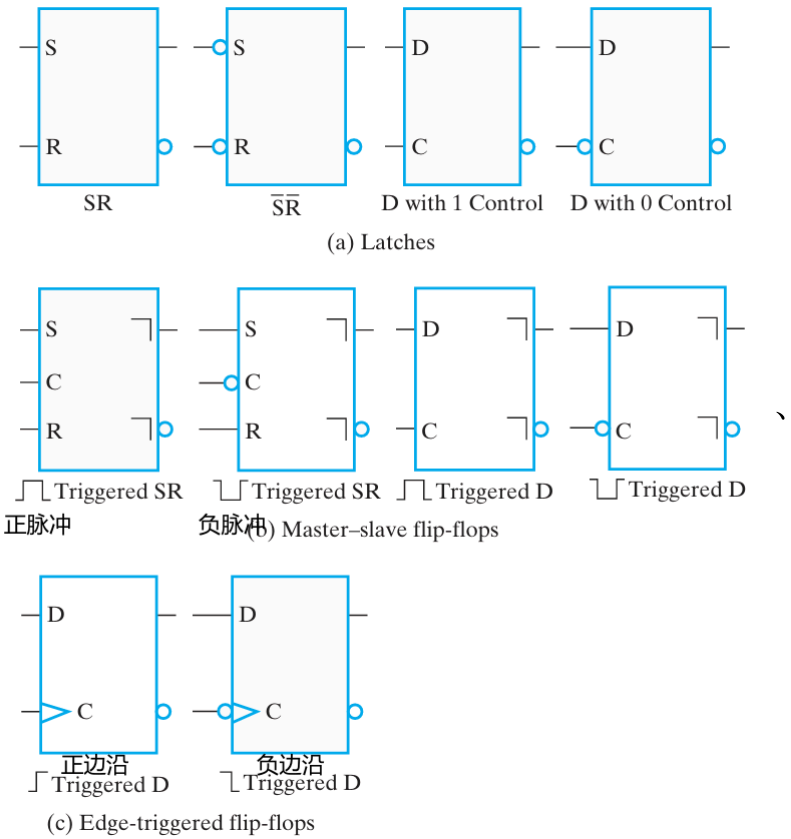


FIGURE 4-10
Positive-Edge-Triggered D Flip-Flop

- 正边沿多一个反相器

标准图形符号



直接输入：在加载时钟前，将数字系统中的触发器设置成初始状态

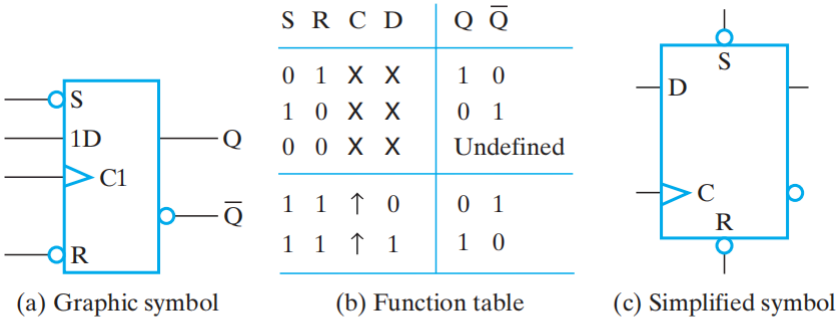


FIGURE 4-12 直接置位和复位的D触发器
D Flip-Flop with Direct Set and Reset

4.2.3 Difference between Latches and Flip-flops

- Difference between Latches and Flip-Flops

Latch	Flip-Flop
Latches are level sensitive devices	Flip-flops are edge sensitive devices
Latches are sensitive to glitches	Flip-flops are immune to glitches
Latches take less gates and power	Flip-flops take more gates and power
Latches are faster	Flip-flops are slower

- Latches are transparent.
- Master-slave flip-flops use alternating clocks to break the path from input to output.
- S-R/J-K master-slave flip-flops have "1s catching" behavior.
- Edge-triggered flip-flops respond to the input at a well-defined moment (at the clock-transition).

4.3 Sequential circuit analysis

4.3.1 State Diagrams 状态图

状态等价：Two states are **equivalent** if their response for each possible input sequence is an identical output sequence.

Alternatively, two states are **equivalent** if their outputs produced for each input symbol is identical and their next states for each input symbol are the same or equivalent.

弧线上是输入/输出，如果输出在里面（moore 模型），说明输入不会改变输出的值

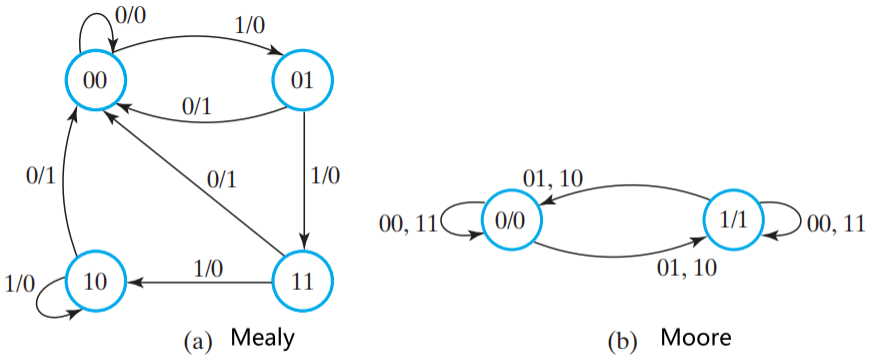


FIGURE 4-15
State Diagrams

4.3.2 Moore and Mealy Models

Moore Model 摩尔模型

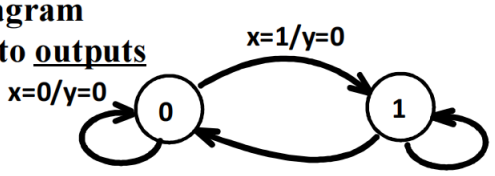
- Named after E.F. Moore
- Outputs are a function ONLY of states

- Usually specified on the states 输出仅依赖于状态

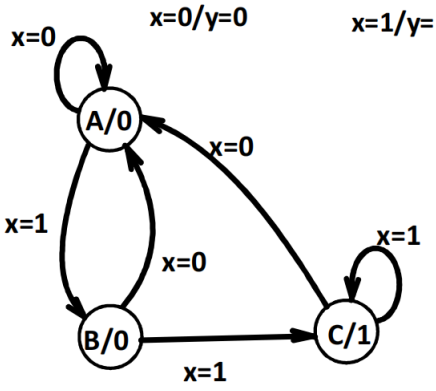
Mealy Model 米利模型

- Named after G. Mealy
- Outputs are a function of inputs AND states
- Usually specified on the state transition arcs 依赖于状态和输出条件（转换条件）

- **Mealy Model State Diagram**
  maps inputs and state to outputs



- **Moore Model State Diagram**
  maps states to outputs



4.3.3 State tables 状态表

当前状态、输入、下一状态、输出

一个包含 m 个触发器和 n 个输入的时序电路的状态表有 2^{m+n} 行

4.4 Sequential Circuit Design

1. Finding a State Diagram
2. Convert it to a state table

4.4.1 状态简化

状态简化：获得一个最小化的状态表。这个表不仅能正确地反映设计的全部要求，而且状态的数目最少

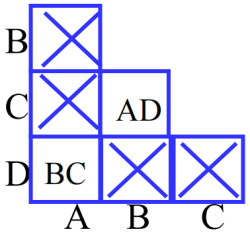
状态等效（equivalent）：状态 S1 和 S2 是完全确定状态表中的两个状态，对于所有可能的输入序列，输出响应序列完全相同

- 或次态相同
  或次态交错
  或次态循环

完全确定状态表： 状态表中的次态和输出都有确定的状态和确定的输出值

4.4.1.1 隐含表化简

- **(AD)、(BC)输出相同，次态循环，故状态A、D等效，状态B、C等效。**

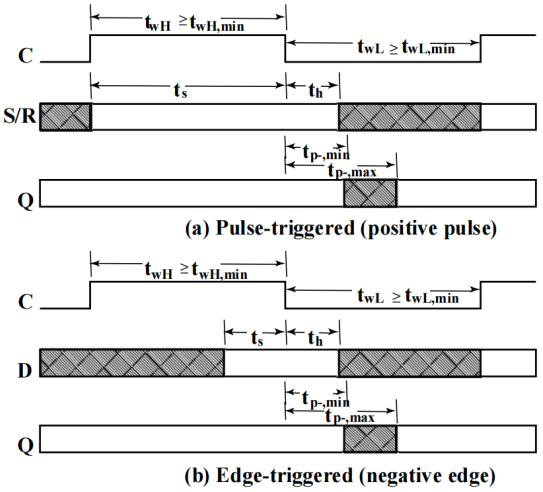


Q \ X	0	1
A	A/0	B/0
B	A/1	C/0
C	D/1	C/0
D	A/0	C/0

Q \ X	0	1
A	A/0	B/0
B	A/1	B/0

4.5 Sequential Circuit Timing

- t_s - setup time
- t_h - hold time
- t_w - clock pulse width
- t_{px} - propagation delay
  - t_{PHL} - High-to-Low
  - t_{PLH} - Low-to-High
  - t_{pd} - max (t_{PHL}, t_{PLH})



a) 主从触发器：传播延迟 $> t_{hold}$，$t_{setup} = t_w$

b) 边缘触发器（负）：$t_{setup} < t_w$

t_s - **setup time** 触发器建立时间，输入保持一段时间不变 - minimum time for which the *S* and *R* or *D* inputs must be maintained at a constant value prior to the occurrence of the clock transition that causes the output to change

- Master-slave - Equal to the width of the triggering pulse
- Edge-triggered - Equal to a time interval that is generally much less than the width of the the triggering pulse

t_h - **hold time** 在输出改变后保持一段时间- Often equal to zero
t_{px} - **propagation delay** 传播延迟，时钟触发沿与输出稳定为一个值之间的时间间隔

- Same parameters as for gates except
- Measured from clock edge that triggers the output change to the output change

$t_{pd,FF}$ - 触发器延迟

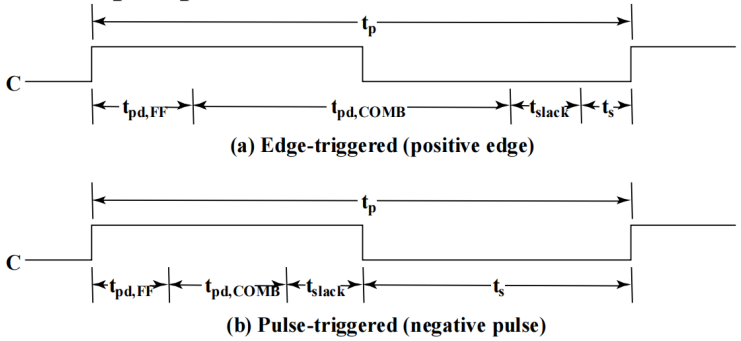
$t_{pd,COMB}$ - 组合逻辑电路总延迟时间（触发器输入-输出）

t_{slack} - 松弛时间 - extra time in the clock period in addition to the sum of the delays and setup time on a path ≥0

t_p(clock period = 1/clock frequency)是所有这些时间之和 ≥ max($t_{pd,FF}$ + $t_{pd,COMB}$ + t_s)

If the **clock period is too short**, some data changes will not propagate through the circuit to flip-flop inputs before the setup time interval begins

- **Timing components along a path from flip-flop to flip-flop**



计算可允许的 $t_{pd,COMB}$：主从触发器可允许的 gates 要少于边缘触发器

5. Digital Hardware Implementation

5.1 The Design Space

5.2 Programmable Implementation Technologies

可编程实现技术

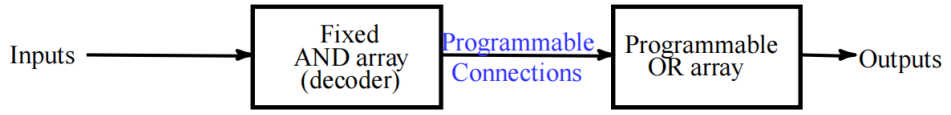
Read Only Memories (ROM): Fixed AND array and programmbale ORs

- PROM have: N input lines, M output lines, and 2^N decoded minterms

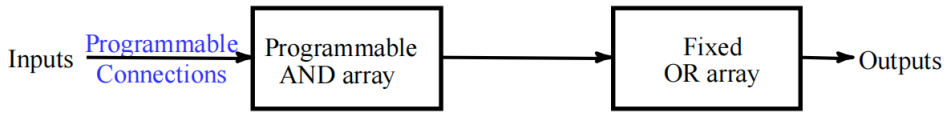
Programmable Array Logic (PAL) 与项可编程: The PAL is the opposite of the ROM, having a programmable set of ANDs combined with fixed ORs

Programmable Logic Array (PLA): Compared to a ROM and a PAL, a PLA is the most flexible having a programmable set of ANDs combined with a programmable set of ORs.

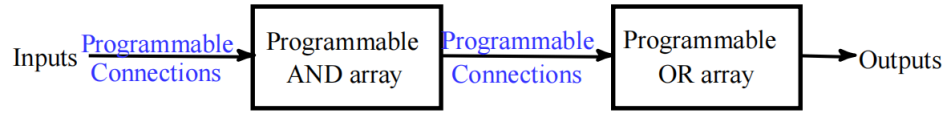
AND	OR	DEVICE
Fixed	Fixed	not programmable
Fixed	Programmable	PROM
Programmable	Fixed	PAL
Programmable	Programmable	PLA



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL) device



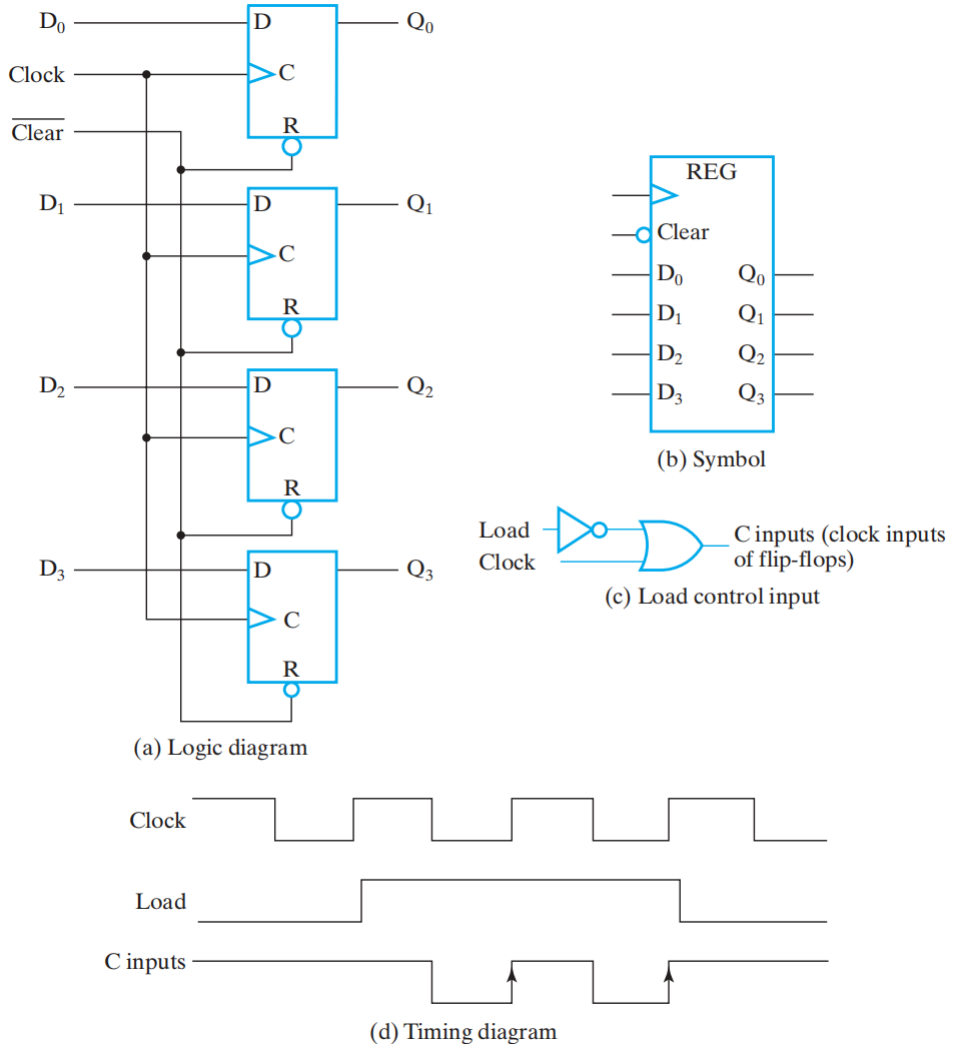
(c) Programmable logic array (PLA) device

6. Register and Register Transfers

6.1 Register, Microoperations and Implementations

6.1.1 Register

6.1.1.1 寄存器



- (a) 由四个 D 触发器构成的寄存器，输入 $\overline{clear}$ 必须是 0 才能触发异步复位
- (b) 寄存器符号表示，clear 外的圆圈表示：输入该信号加载逻辑 0，触发清零操作
- Load-Controlled Feedback 寄存器载入/保持（loading）
  - load = 0, hold current values 保持
  - load = 1, load input values 传输

6.1.1.2 并行加载寄存器

clock gating 门控时钟

- $C_{input} = \overline{load} + clock$
- load = 1，$C_{input} = clock$，新数据再时钟上升沿传输至寄存器
- load = 0，$C_{input} = 1$，寄存器内容不变
- load =0，C=X，load D

6.1.2 寄存器传输

微操作(microoperation)：对寄存器存储数据执行的基本操作

寄存器传输语言(Register Transfer Language, RTL)

寄存器类型：地址寄存器（AR）、程序计数器（PC）、指令寄存器（IR）、R2（寄存器2）

$if(K1=1)then(R2 \leftarrow R1)$ 可以写为 $K1: R2 \leftarrow R1$

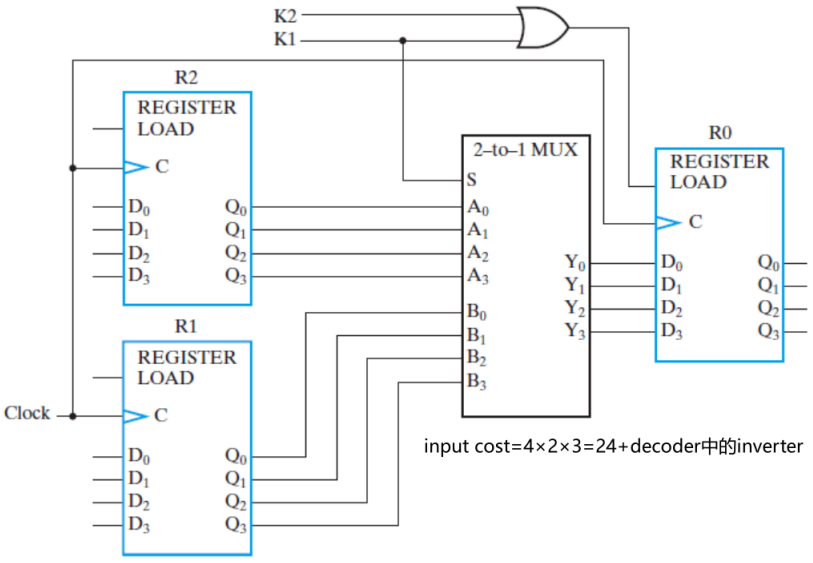
微操作类型：算术微操作、传输微操作、逻辑微操作、移位微操作

Symbolic Designation	Description
R0 ← R1 + R2	Addition
R0 ← $\overline{R1}$	One's Complement
R0 ← $\overline{R1}$ + 1	Two's Complement
R0 ← R2 + $\overline{R1}$ + 1	R2 minus R1 (2's Comp.)
R1 ← R1 + 1	Increment (count up)
R1 ← R1 – 1	Decrement (count down)

6.2 Counter, Register cells, Buses & Serial operations

6.2.1 对单个寄存器的微操作

基于多路复用器的传输



3-1 多路复用器实际上用的是 4-1（1 个位弃用）

有总线的多路复用器多寄存器传输相比只用多路复用器更加便宜（gate input cost 低）

three-state bus 三态总线

6.2.2 移位寄存器

移位寄存器 shift register

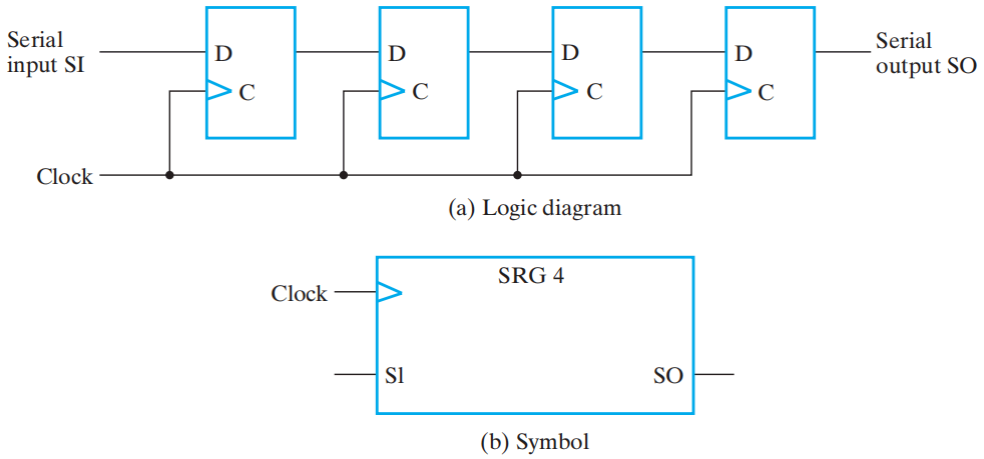


FIGURE 6-9
4-Bit Shift Register

带有 **并行加载** 功能的移位寄存器

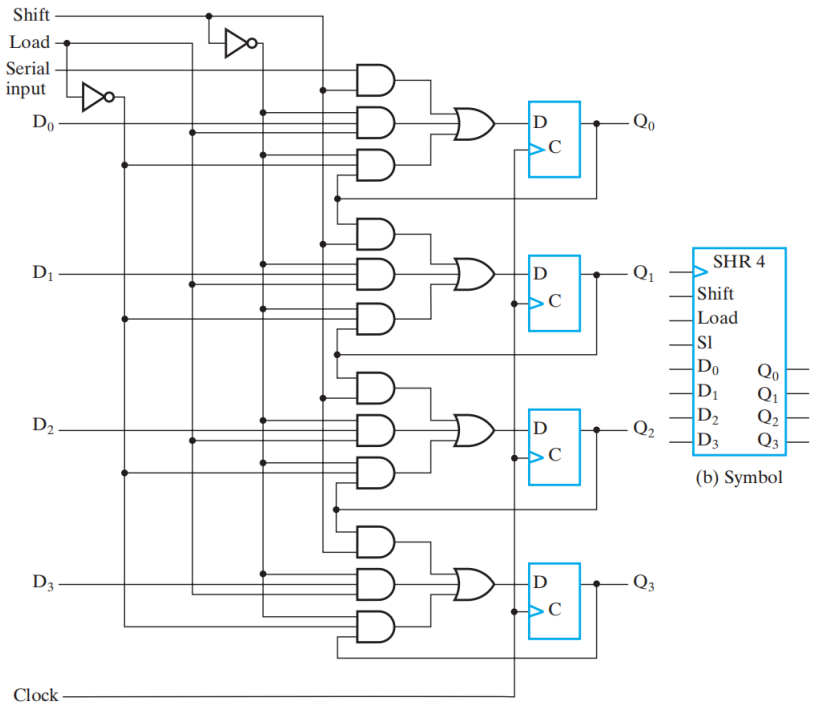


FIGURE 6-10
Shift Register with Parallel Load

- $Shift$: $Q \leftarrow \text{sl}\ Q$，移位
- $\overline{Shift} \cdot Load$: $Q \leftarrow D$，并行载入
- $\overline{Shift} \cdot \overline{Load}$: $Q \leftarrow Q$，保持

双向移位寄存器(bidirectional shift register)

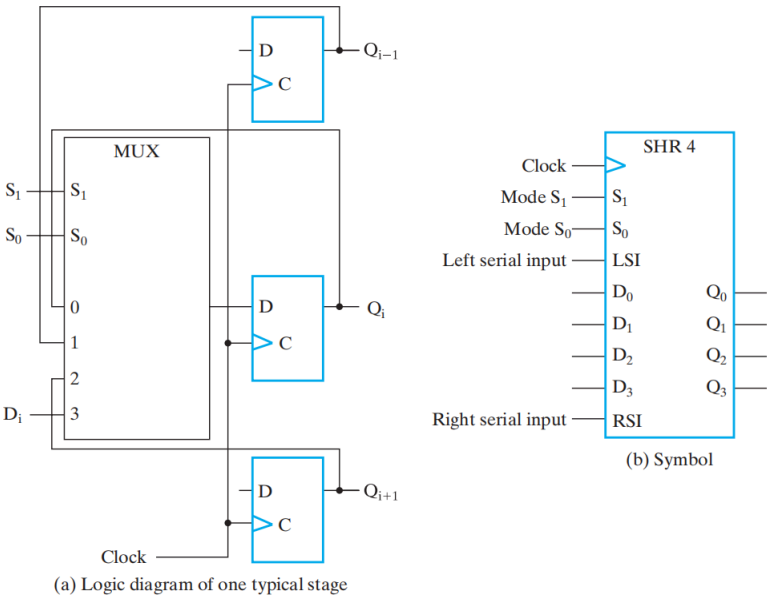


FIGURE 6-11
Bidirectional Shift Register with Parallel Load

$$\overline{S}_1 \cdot S_0\colon\ Q \leftarrow \mathrm{sl}\, Q$$
$$S_1 \cdot \overline{S}_0\colon\ Q \leftarrow \mathrm{sr}\, Q$$
$$S_1 \cdot S_0\colon\ Q \leftarrow D$$

- 每一级由一个 D 触发器和一个 4-1 多路复用器构成
-

S0	S1	寄存器操作
0	0	保持不变
0	1	左移
1	0	右移
1	1	并行加载到 D 触发器

6.3 Counter

计数器：能够在输入脉冲序列的激励下便利指定状态序列的寄存器

一个 n 位的二进制计数器由 n 个触发器构成，计数范围为 0~2^n-1

6.3.1 Ripple Counter 行波计数器

行波计数器：加载到某些触发器 C 输入端的值不是公用的时钟脉冲，而是其他触发器的输出信号

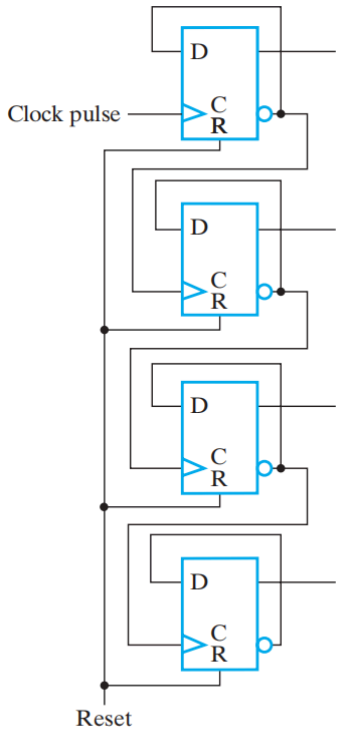


FIGURE 6-12
4-Bit Ripple Counter

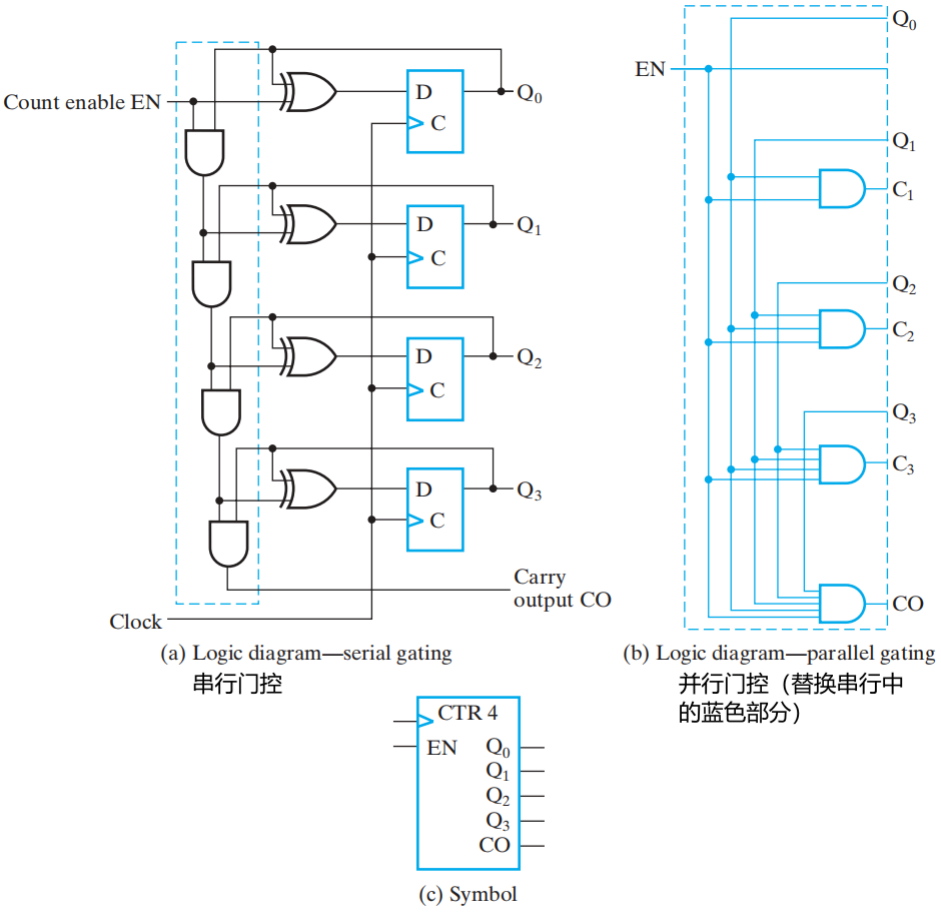
- 每次 Q_n 发生从 **1 到 0 的跳变** 将使 Q_{n+1} 发生翻转
- 行波计数器硬件简单，但是是异步时序电路，时延大时很不稳定

6.3.2 Synchornous Counter 同步计数器

同步计数器：所有触发口 C 输入端口都加载公用的时钟信号

4 位二进制同步计数器

并行计数器从状态 1111 到状态 0000 的转变只需要一个与门的时延，二串行计数器需要 4 个



(a) Logic diagram—serial gating
串行门控

(b) Logic diagram—parallel gating
并行门控（替换串行中的蓝色部分）

(c) Symbol

FIGURE 6-13
4-Bit Synchronous Binary Counter

- EN 为计数器使能输入信号
  - 当 EN = 1，计数器可以正常向上或向下计数
  - 当 EN = 0，计数器不执行计数操作

6.3.3 其他计数器

并行加载功能的 4 位二进制计数器

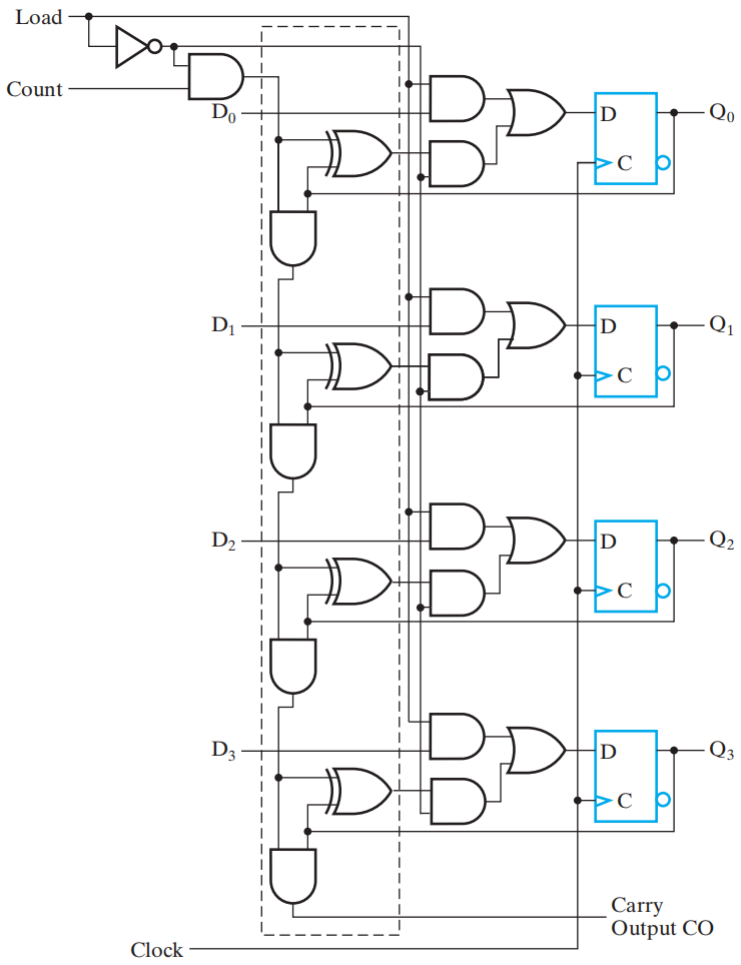


FIGURE 6-14
4-Bit Binary Counter with Parallel Load

同步 BCD 计数器（无加载功能）

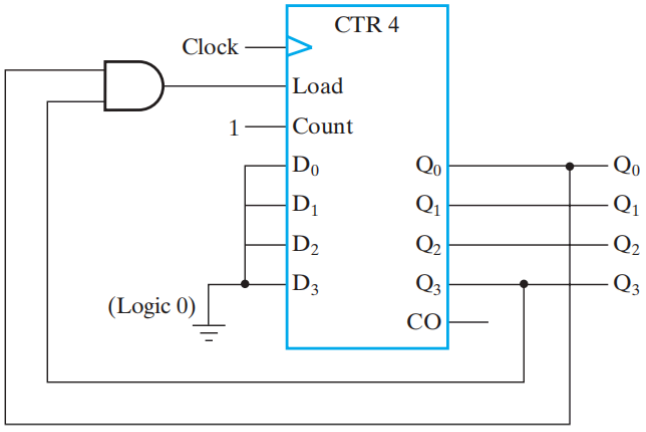


FIGURE 6-15
BCD Counter

任意计数序列的计数器?

1. 状态表（当前-下一状态）
2. 得到简化的方程：D_A =?
3. 画逻辑图

6.3.4 寄存器单元设计

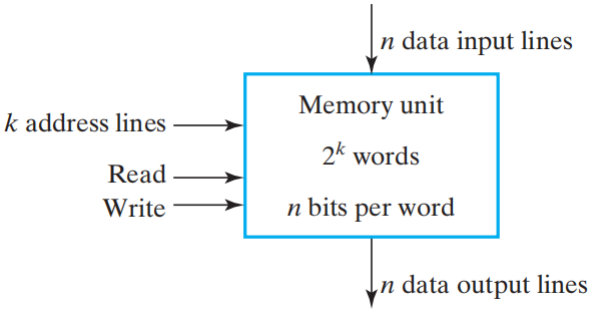
Serial Adder 串行加法器

6.4 Control of Register Transfers

7. Memory Basics

7.1 Definition

Block Diagram of Memory



- k address lines are decoded to address 2^k words of memory.
- Each word is n bits.
- Read and Write are single control lines defining the simplest of memory operations.

总共 k（$k > log_2$储存器字数） 个地址线，n 个输入输出线

7.2 RAM

Random Access Memory (RAM) 随机访问内存

- 读操作：将目标二进制地址加载到地址线；存入的数据信息加载到数据输入线；激活输入信号
- 写操作：将目标二进制地址加载到地址线；激活输入信号

内存大小 12bit to 4096 12bit = 12×4096/8 = 6k

静态 RAM(SRAM)由储存二进制信息的内部锁存器构成，信息会一直被存储直到断电

动态 RAM(DRAM)以电荷电容的形式存储信息，易于使用，读写周期短，且不需要刷新

7.2.1 SRAM

SRAM cell

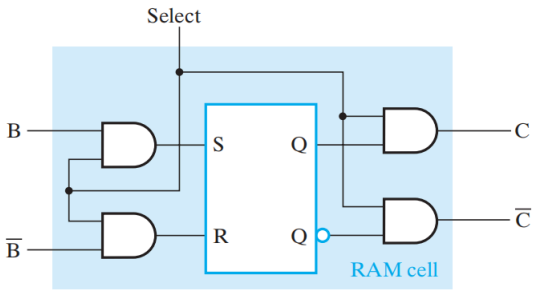
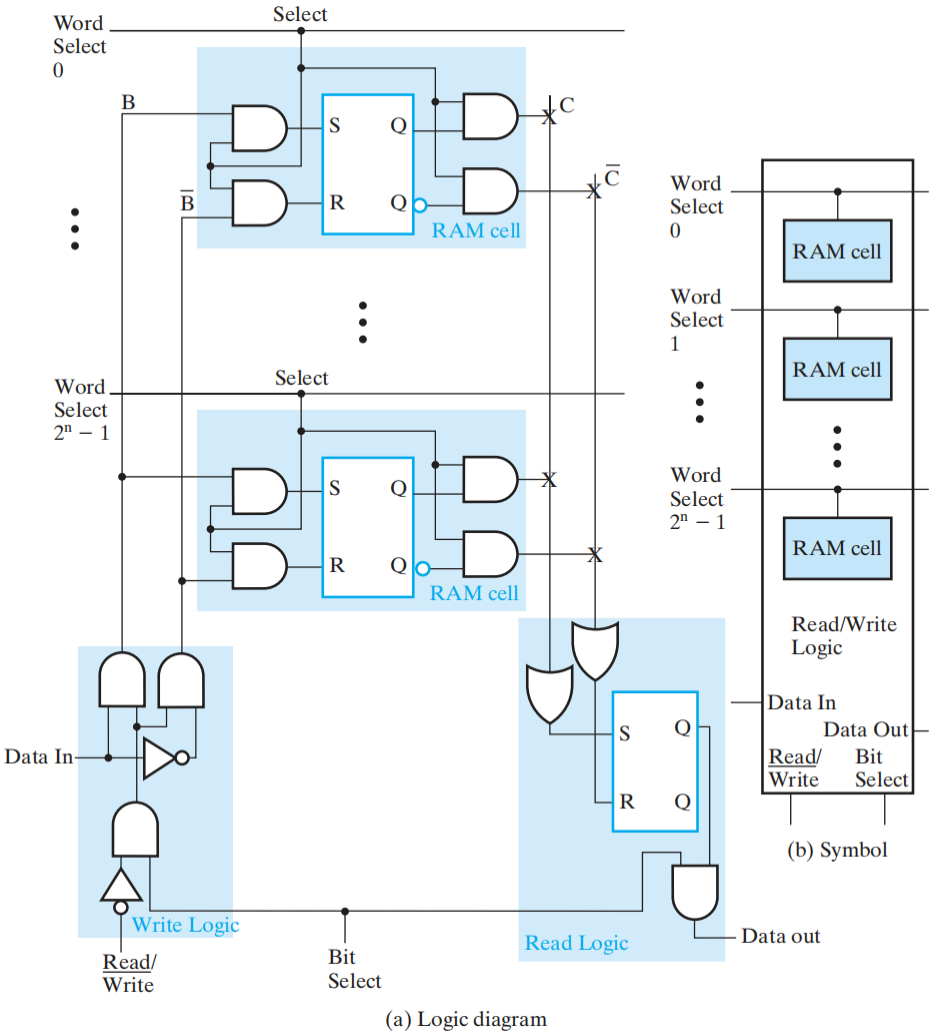


FIGURE 7-4
Static RAM Cell

- SR 锁存器
- Select 为输入使能控制
  - Select = 0，内容保持不变；C 和 $\overline{C}$ 都为 0
  - Select = 1，内容由 B 和 $\overline{B}$ 的值决定；C 为储存值



为了修改已储存的值，$Read/\overline{Write}$ must be 0 and Bit Select must be 1

Decoder — decodes the n address lines to 2^n word select lines

Coincident Selection 重合选择：用两个 decoder，分别负责横向和纵向的寻址

- Word select becomes Row select
- Bit select becomes Column select

RAM 芯片中的译码器：具有 k 个输入和 2^k个输出，需要 2^k个具有 k 个输入的与门

7.2.2 DRAM

扩展内存

- **字扩展**（扩展 word 的数量）：将多个 RAM “并联”，并相应地扩展地址的位宽；
- **位扩展**（扩展 word 的位宽）：将多个 RAM “串联”，并相应地扩展输入输出的位宽；

Address multiplexing 地址复用

Coincident selection 重合选择