



---

# 数字逻辑设计

-- 新书介绍 --

薛宏熙

清华大学计算机系

[xuehx@tsinghua.edu.cn](mailto:xuehx@tsinghua.edu.cn)

# 提 要



- 第一部分            概貌
- 第二部分            VHDL行为描述与结构描述的对比
- 第三部分            各章主要内容

## 数字逻辑设计

薛宏熙 胡秀珠 编著  
清华大学出版社 2008.10



# 第一部分 概貌

# 内容的取舍



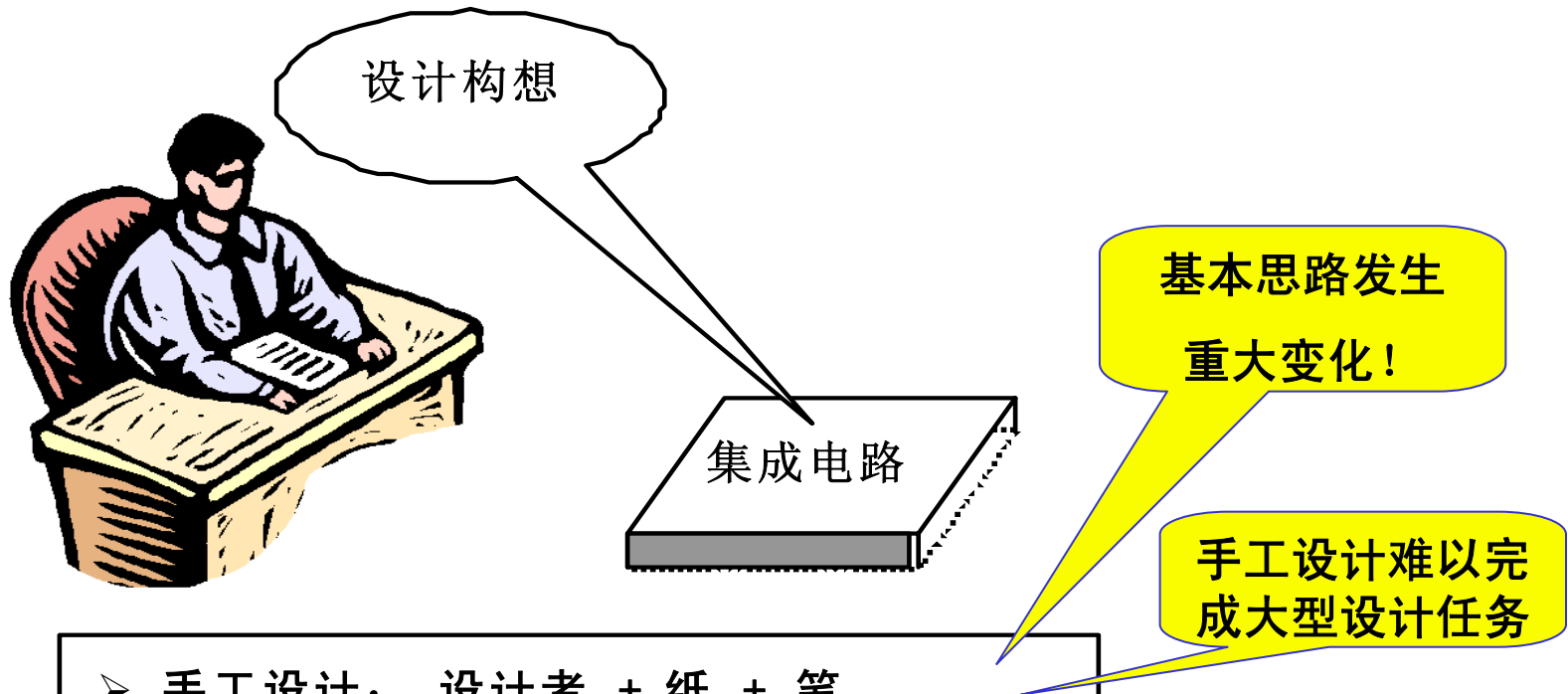
- ◆ 教材和教学方法随科技进步而**更新**，在学好**基本理论**的同时，学会用**EDA工具**进行**设计**：
  - 硬件描述语言：**VHDL**；
  - EDA 工具：**Quartus II**；
  - 以**FPGA**为基础的实验板。
- ◆ **适当扩展**：
  - M O S 电路及集成电路物理实现；
  - EDA 工具实现的算法 → 大体了解EDA工具怎样工作
- ◆ **适当减少**：
  - TTL电路的分量；
  - 略去有关中规模集成电路的介绍（74系列电路）。
    - 现在的重点是在芯片上集成（**SOC**）而不是在板上集成。

# 写作风格（作者的目标）



- ◆ 在不失**严谨**的前提下力求**简明易懂**；
- ◆ 多用**图、表**；
- ◆ 每章包含一节**解题示例**；
- ◆ 和相关课程衔接而不搭接过多，非基本要求加注**星号**；
- ◆ **VHDL代码**：
  - 保留字用**黑体字**，以求醒目；
  - 加注**释**；
- ◆ 逻辑门采用**EDA工具**所用的**国际通用符号**；
- ◆ **光盘**：
  - **Quartus II 8.0**，感谢**Altera**公司的支持！
  - 本书中所有**图片**和**表格**；
  - 本书中所有设计实例的**VHDL代码**。
- ◆ 供参考的**课件**：
  - **EDA工具Quartus II**使用简介；
  - 硬件描述语言**VHDL**简介；
  - **习题解答** 可在清华大学出版社网站下载：
    - 登录：[www.tup.com.cn](http://www.tup.com.cn)
    - 搜索：**数字逻辑设计 薛宏熙**
    - 点击：**课件**

# 设计方法的比较



- 手工设计：设计者 + 纸 + 笔
- 自动设计：设计者 + EDA 工具
  - 硬件描述语言 VHDL / Verilog
  - EDA 工具 Quartus II
  - 以 FPGA 为基础的实验板

# 使用EDA工具设计集成电路

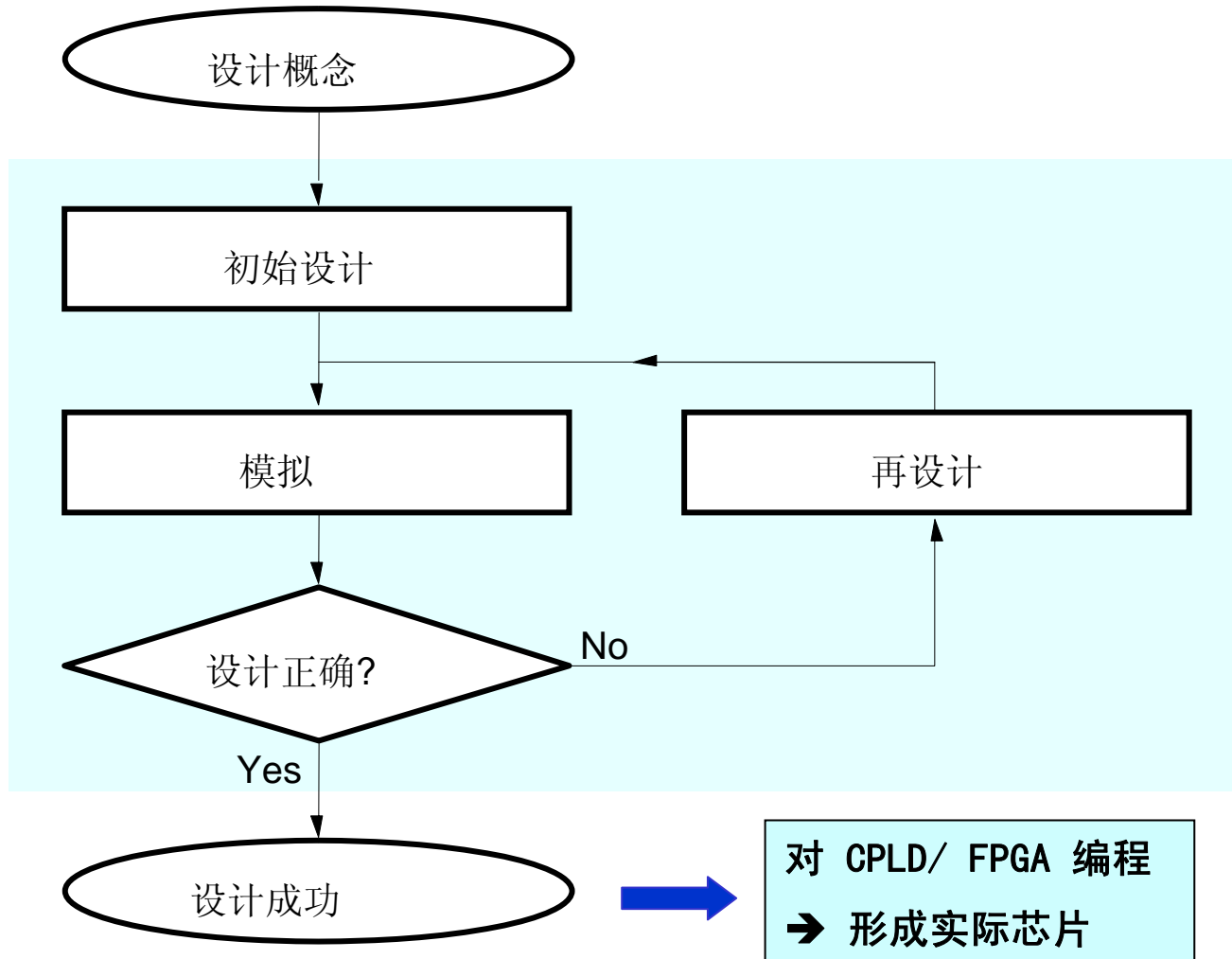


设计描述  
(图形或硬件描述语言)  
→ EDA 工具

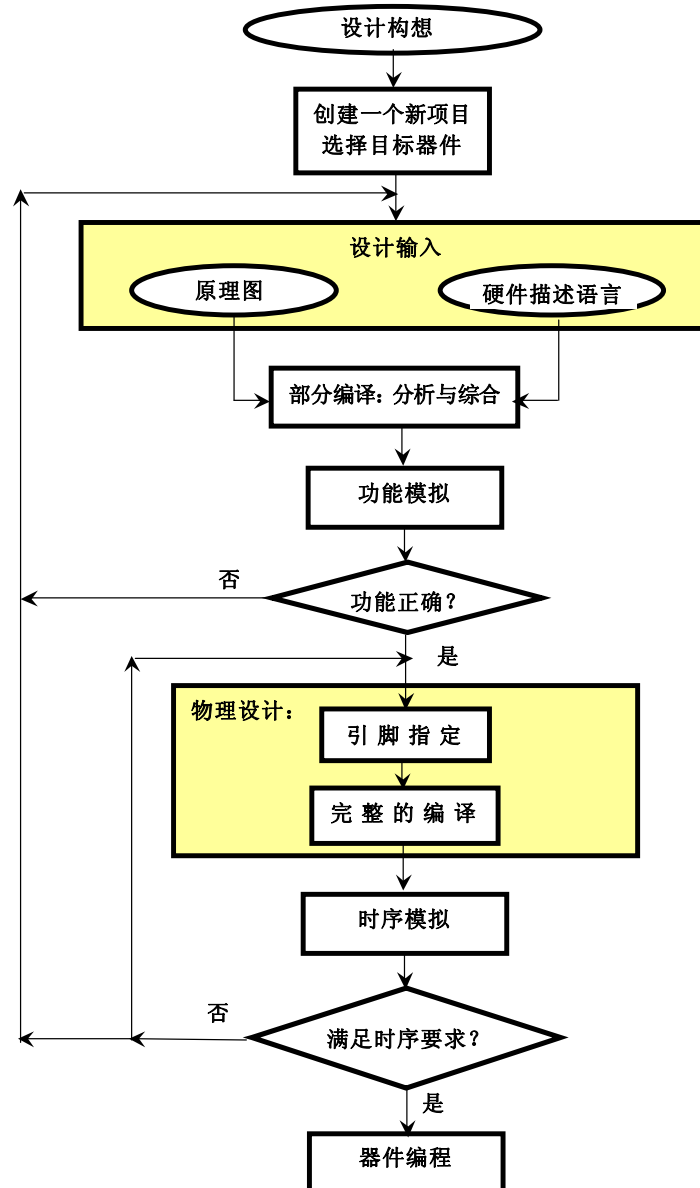


- 设计者的注意力集中于设计描述，细节交给EDA工具

# 基本设计流程



# Quartus II设计流程（详细）



# 教材结构和教学方法的选择



- ◆ VHDL简介和Quartus II都放在附录里；
- ◆ 不是讲完了VHDL和Quartus II才讲数字逻辑，而是：
  - 简略介绍VHDL和Quartus II；
  - 由浅入深、由简到繁讲解数字逻辑；
  - 随着设计实例的逐步引入，VHDL的各种特性也逐渐呈现。
- ◆ VHDL可以表现
  - 电路的结构：结构描述
  - 电路的功能：行为描述

# 提供了一个很好的实验平台

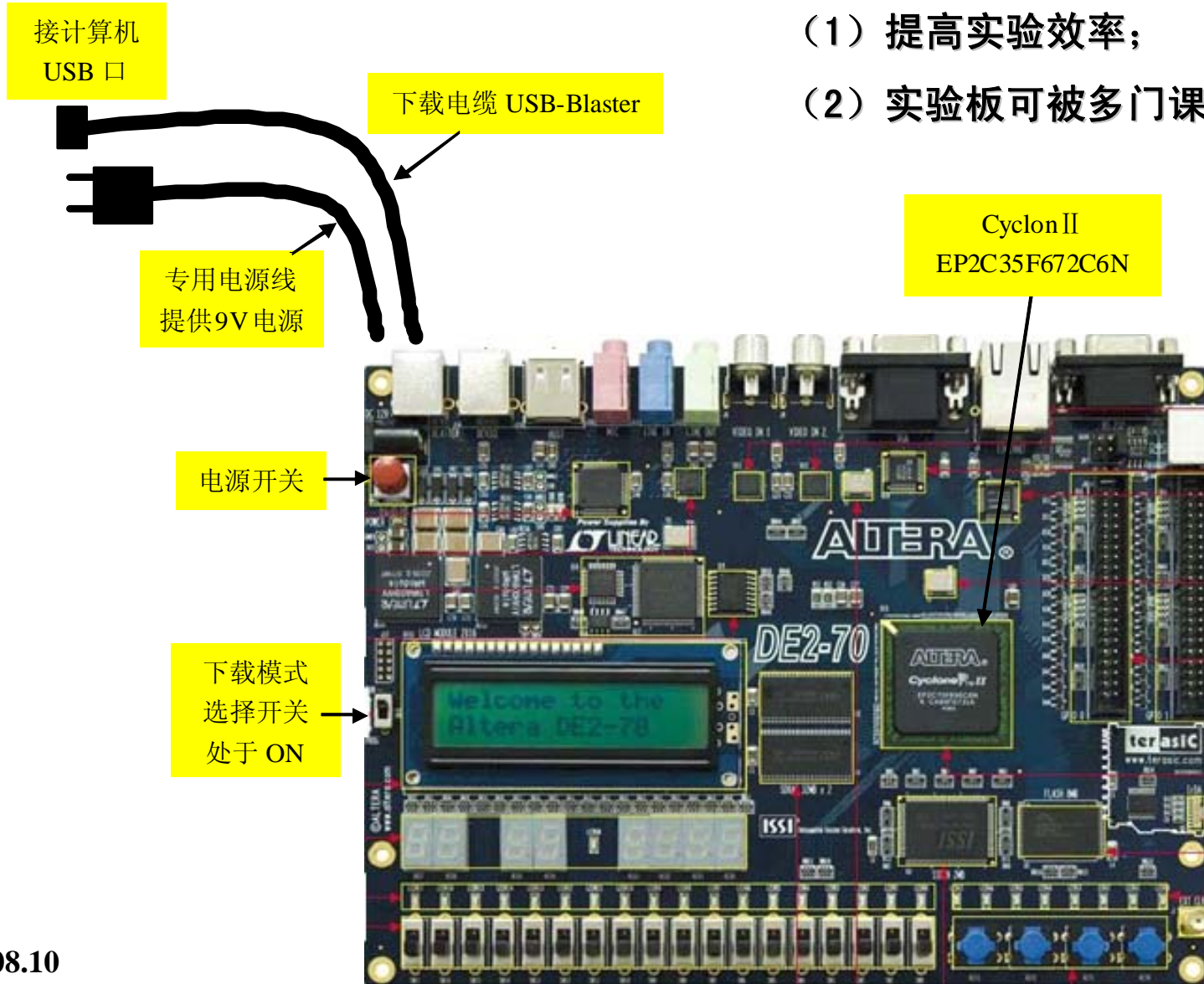


- ◆ **Quartus II :**
  - ALTERA免费提供;
  - 运行于PC, 资源丰富。
- ◆ **设计的正确性验证:**
  - 分析模拟波形: 功能模拟和时序模拟;
  - 硬件实验: 基于FPGA的实验板是一个通用的实验装置。做实验时连线工作很少、效率高。
- ◆ **实验资源的易获得性 → 学生有机会**
  - 自我检验、自主学习;
  - 积小胜为大胜, 增加创新意识。
- ◆ **学习基本理论的同时掌握一门实用技术**

# 实验板举例：DE2



- (1) 提高实验效率；
- (2) 实验板可被多门课程共享。





## 第二部分 VHDL行为描述与原理图描述的对比

Quartus II 可以接受:

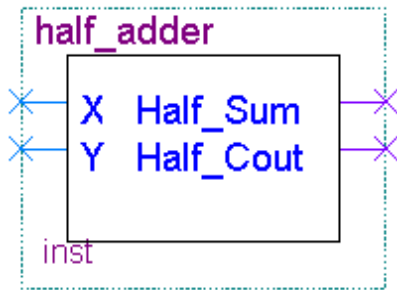
- ◆ 原理图描述（结构描述）；
- ◆ 硬件描述语言描述（**VHDL** / Verilog / AHDL）。
  - 结构描述
  - **行为描述**

# 原理图描述



## 半加器的原理图实现:

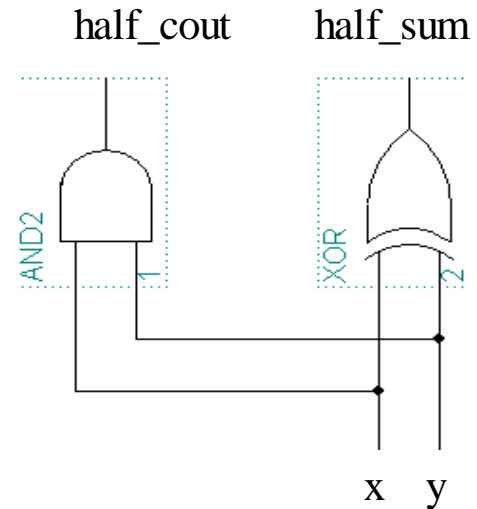
结构描述



x	y	half_cout	half_sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\text{half\_sum} = x \oplus y$$

$$\text{half\_cout} = x \cdot y$$



(a) 符号图

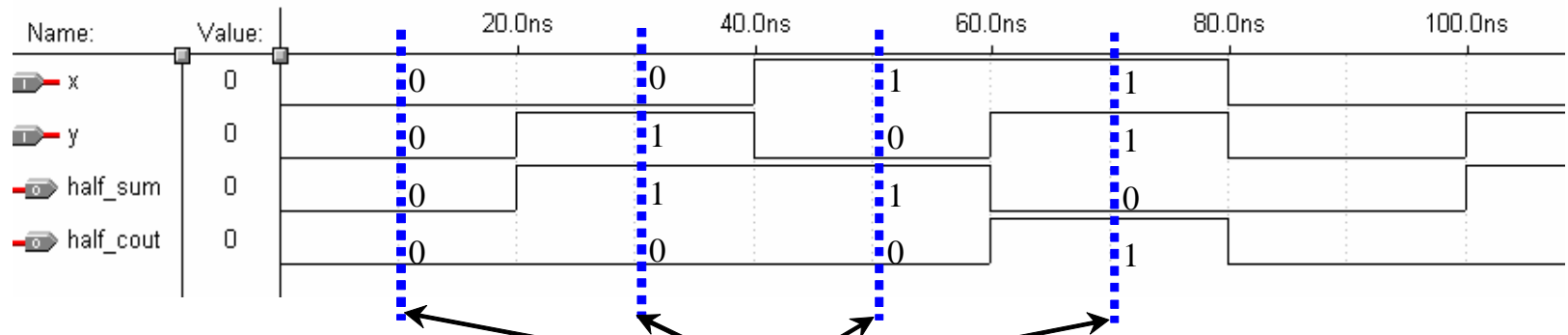
(b) 真值表和布尔表达式

(c) 原理图

# 设计正确性验证：分析模拟波形

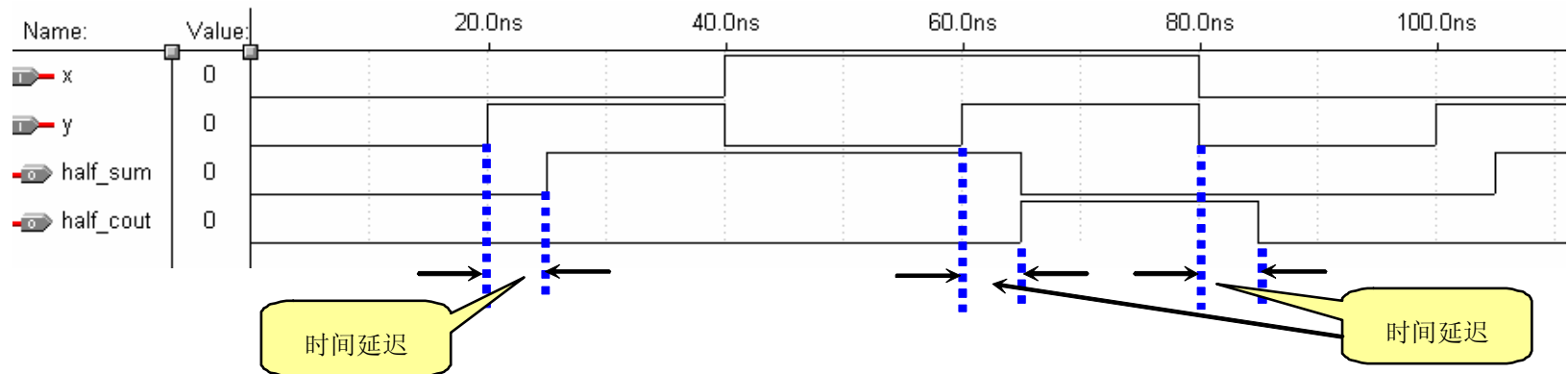


## ◆ 半加器功能模拟结果



观察输入 / 输出信号取值的对应关系，判断此半加器的功能是否与预期相符

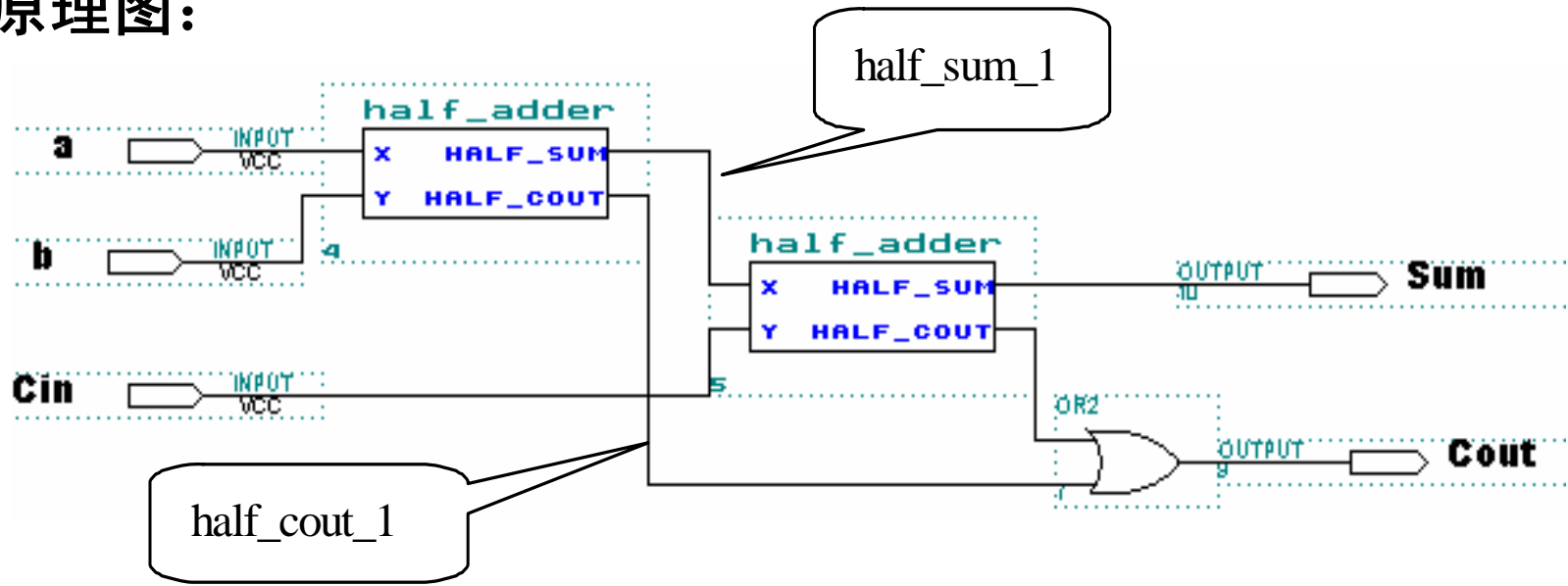
## ◆ 半加器时序模拟结果



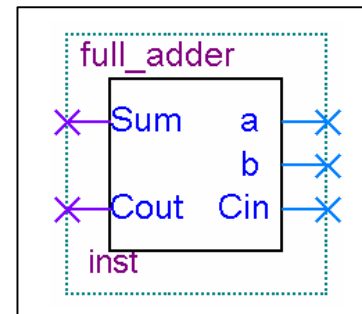
# 用半加器组成 1 位全加器



## ◆ 原理图:



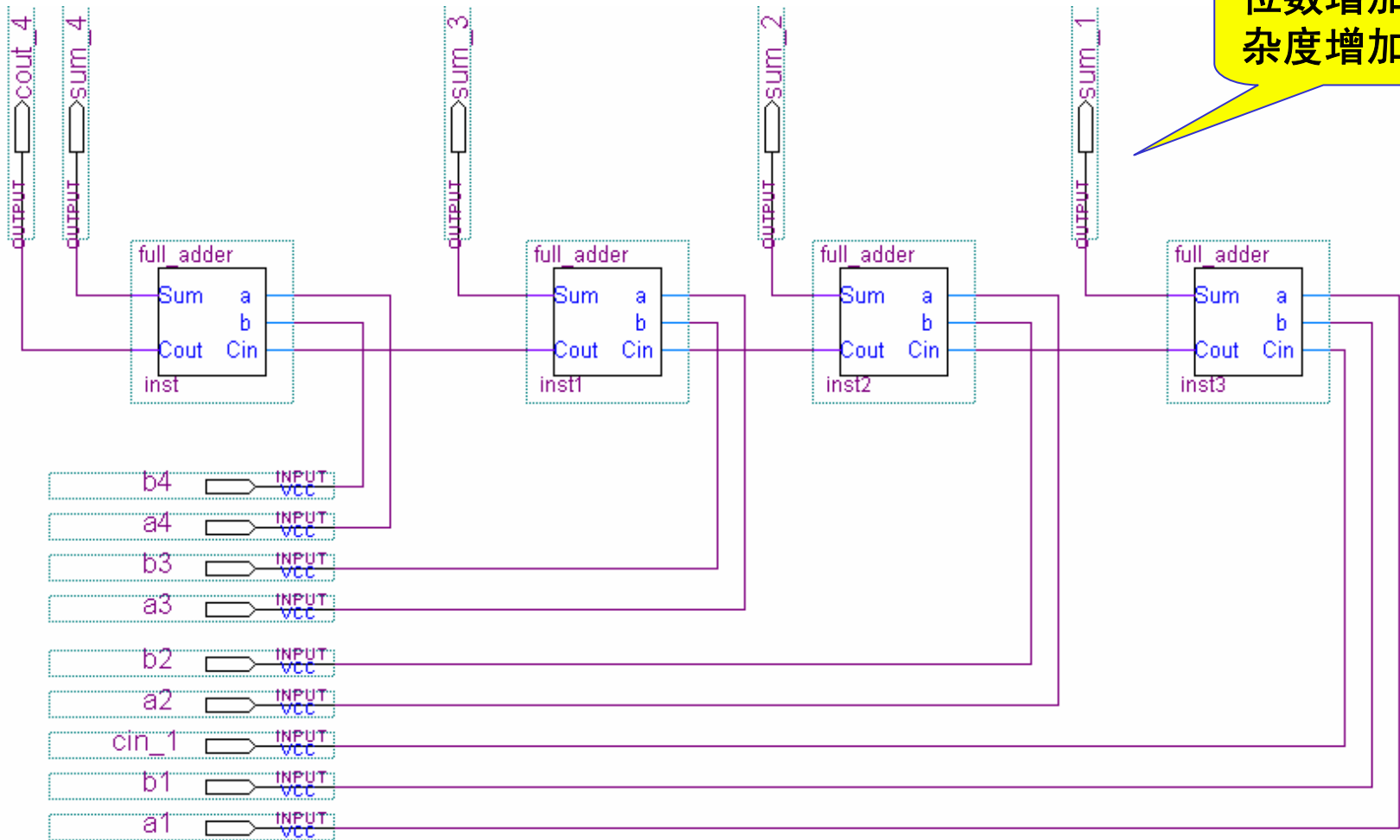
## ◆ 符号图:



# 用 1 位全加器组成 4 位加法器



## ◆ 原理图：



位数增加复  
杂度增加！

# 4 位加法器的VHDL的行为描述



```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;      -- operator '+' is overwritten in the package

ENTITY Adder4 IS                       -- 定义 Adder4 的外部视图
    GENERIC ( width : integer := 4 );  -- 定义一个类属参数 width, 其默认值为 4
    PORT ( a, b: IN std_logic_vector ( width - 1 DOWNTO 0 );
          cin: IN std_logic;
          cout: OUT std_logic;
          Sum: OUT std_logic_vector ( width - 1 DOWNTO 0 )
    );
END Adder4;
```

如果想定义16位加法器，只需将 width 修改为16即可！位数增加VHDL代码复杂度不增加！



## 第3部分 各章主要内容



## ◆ 描述数字电路功能的3种表示方法：

- 真值表
- 卡诺图
- 逻辑表达式

## ◆ 简化逻辑函数的3种方法：

- 公式法
- 卡诺图法
- \*表格法：

- 不受变量个数的限制，而且有严格的操作规则，但当函数规模较大时，手工操作不胜其烦。
- 是学习逻辑综合算法的一个入门向导。

# \*表格法提要

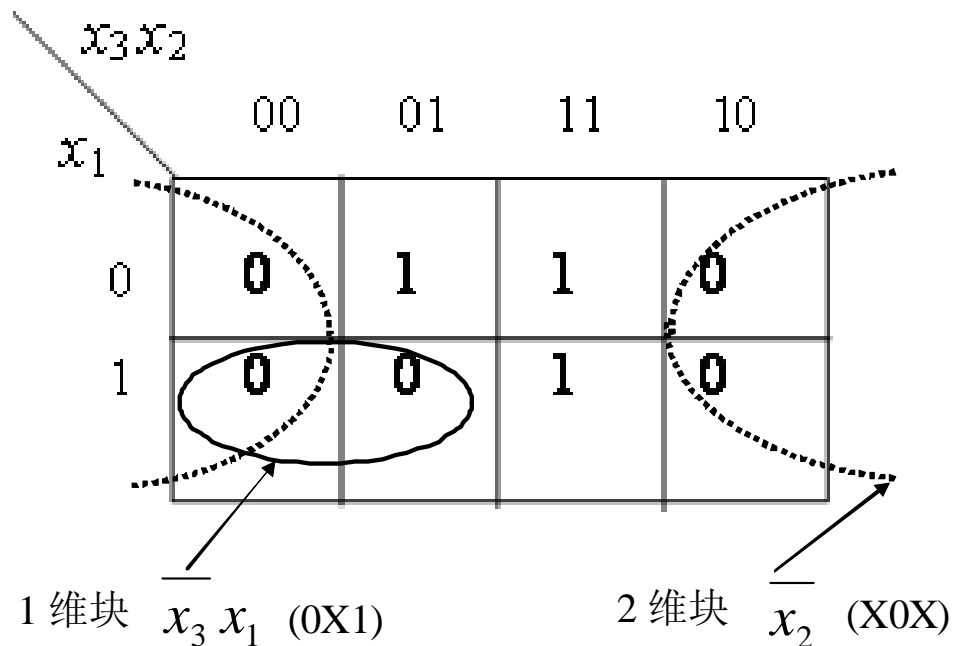


- ◆ 从函数的真值表描述求出其全部质蕴含项的集合，记作 $Z$ 。
- ◆ 从质蕴含项集合 $Z$ 中挑选出一个子集 $M$ ，要求 $M$ 是最小覆盖。即
  - $M$ 覆盖全部最小项。至于无关项，属于可覆盖、可不覆盖之列。
  - $M$ 的成本最低。

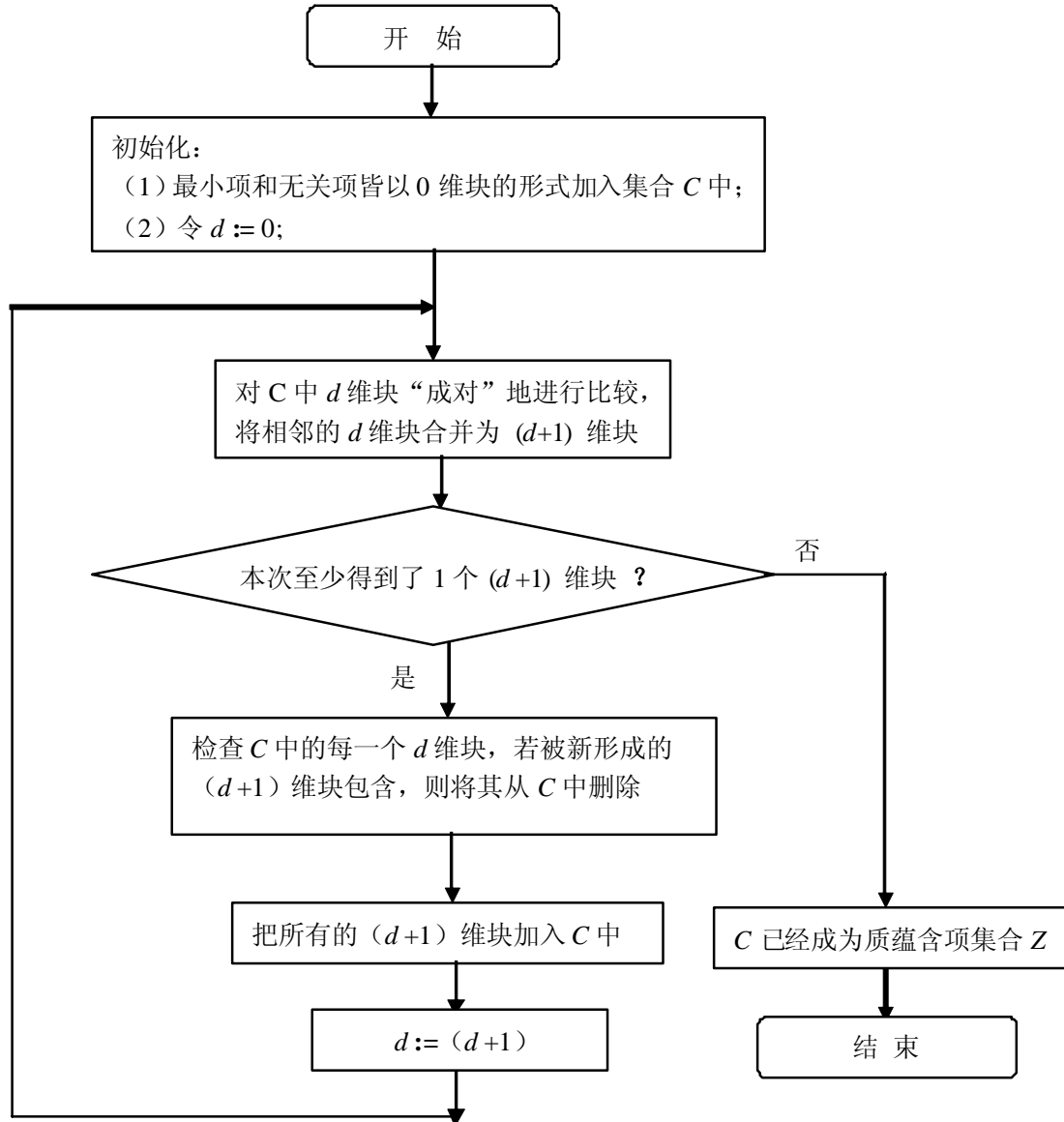
# \*表格法求质蕴含项集合Z



- ◆ 2个相邻的0维块可以合并为1个1维块
- ◆ 2个相邻的1维块可以合并为1个2维块
- ◆ .....
- ◆ 2个相邻的 $d$ 维块可以合并为1个 $(d+1)$ 维块, ...直到不能进一步合并时为止。



# \*表格法求质蕴含项集合Z



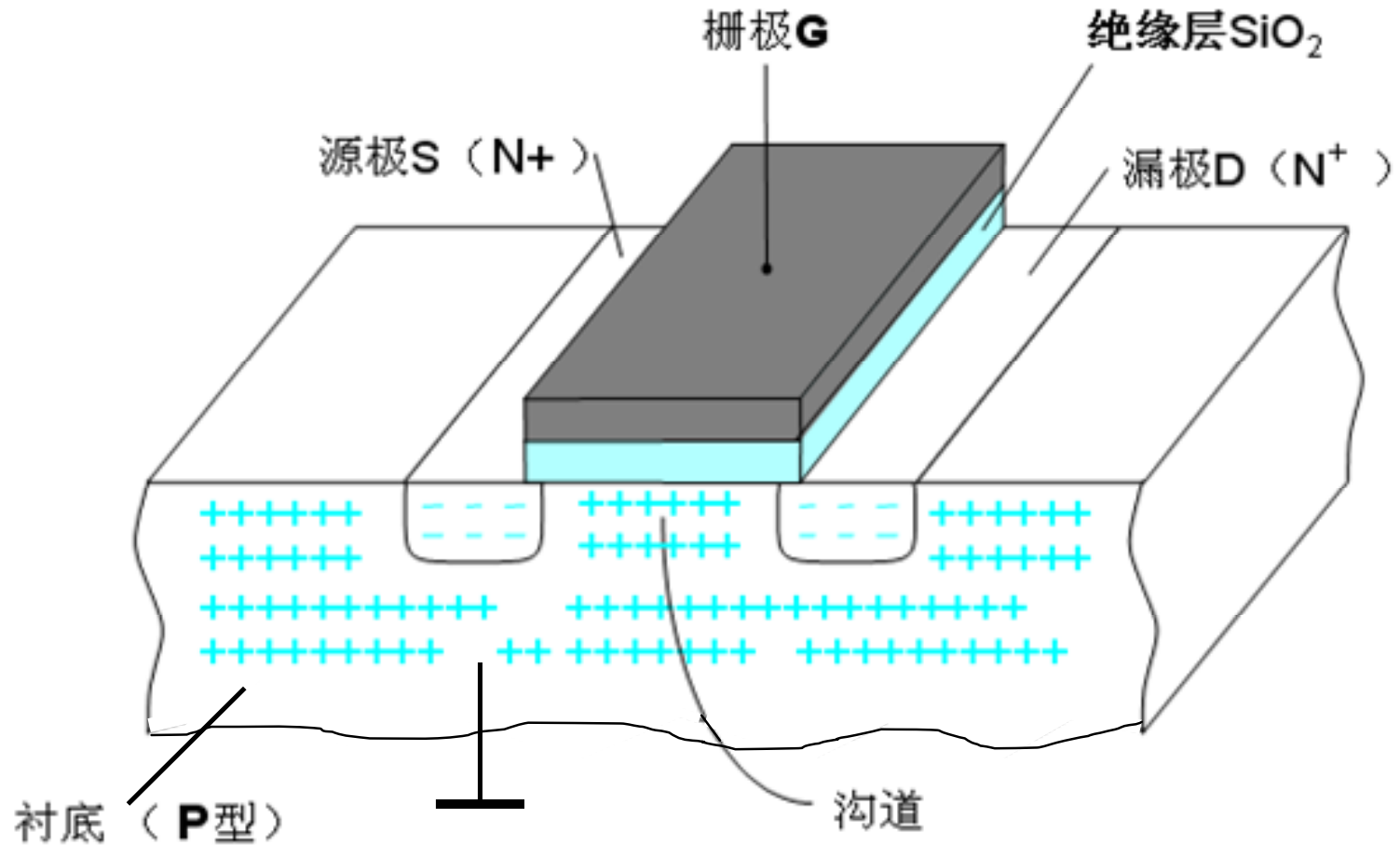
## 第2章 数字集成电路的基本元件 — 门电路



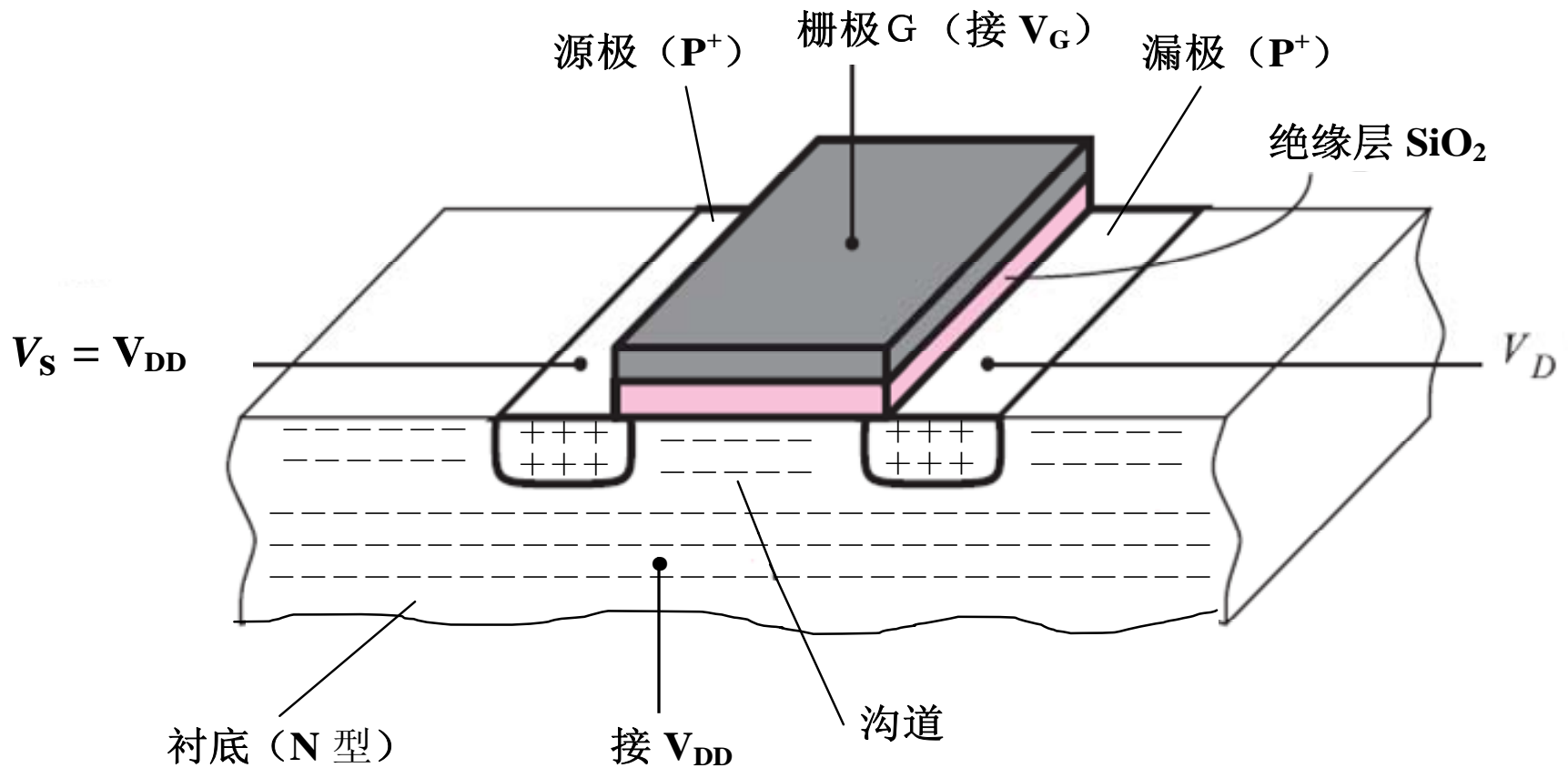
目前的主流

- ◆ TTL集成门电路
- ◆ MOS场效应晶体管
- ◆ MOS门电路
  - NMOS门电路
  - **CMOS门电路**
- ◆ 可编程逻辑器件
  - CPLD
  - FPGA

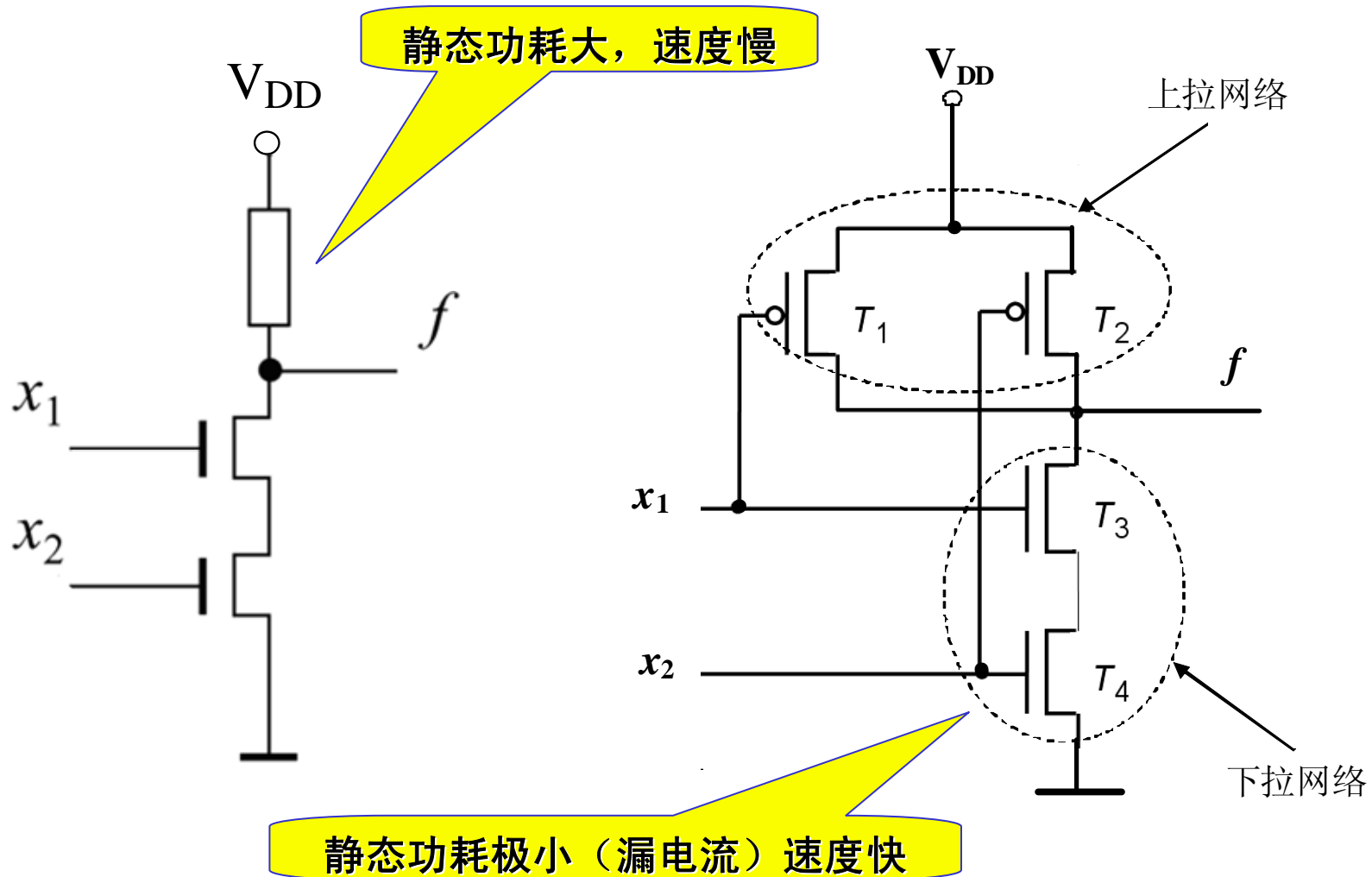
# NMOS晶体管的物理结构



# PMOS晶体管的物理结构



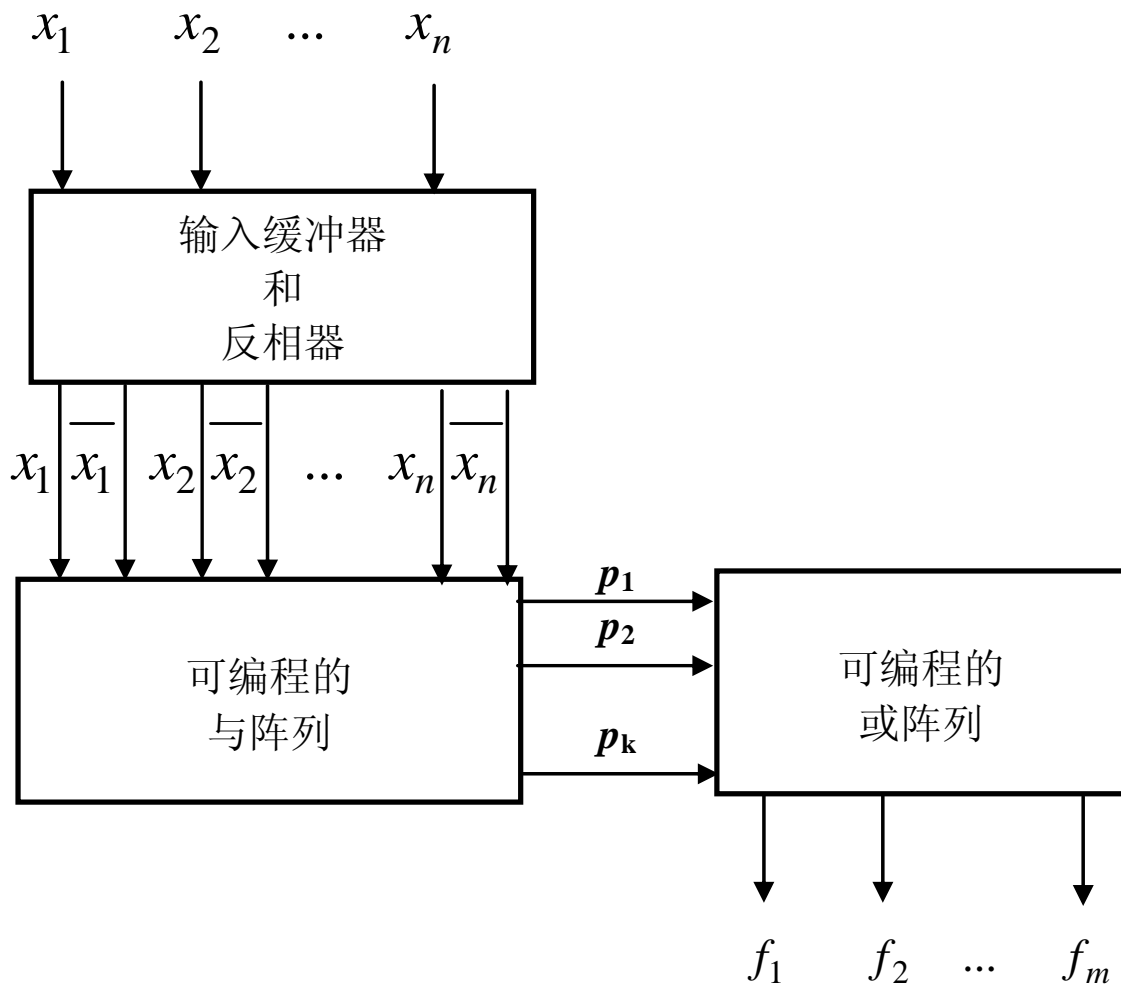
# NMOS与CMOS与非门对比



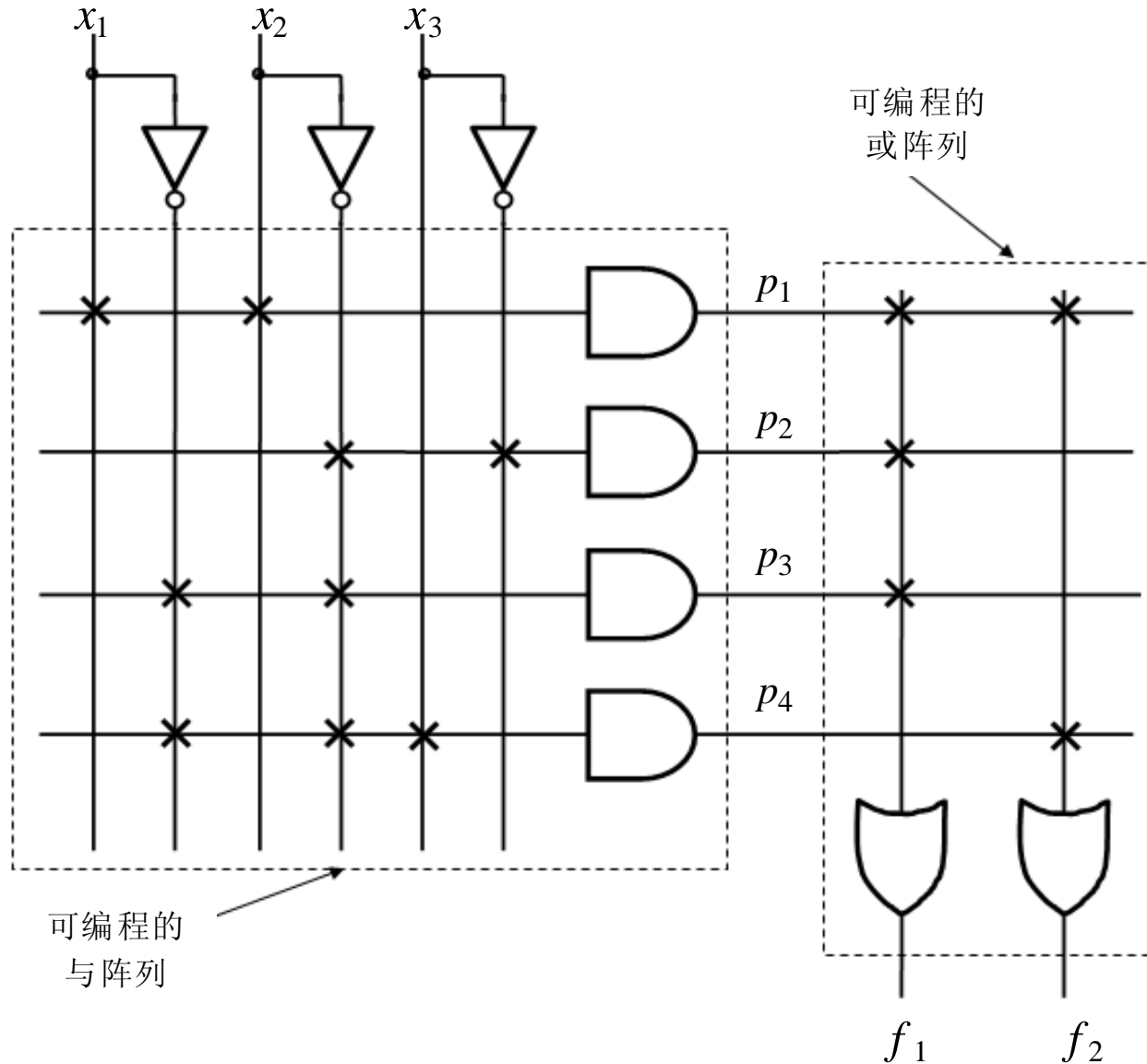
# 可编程逻辑器件的演变



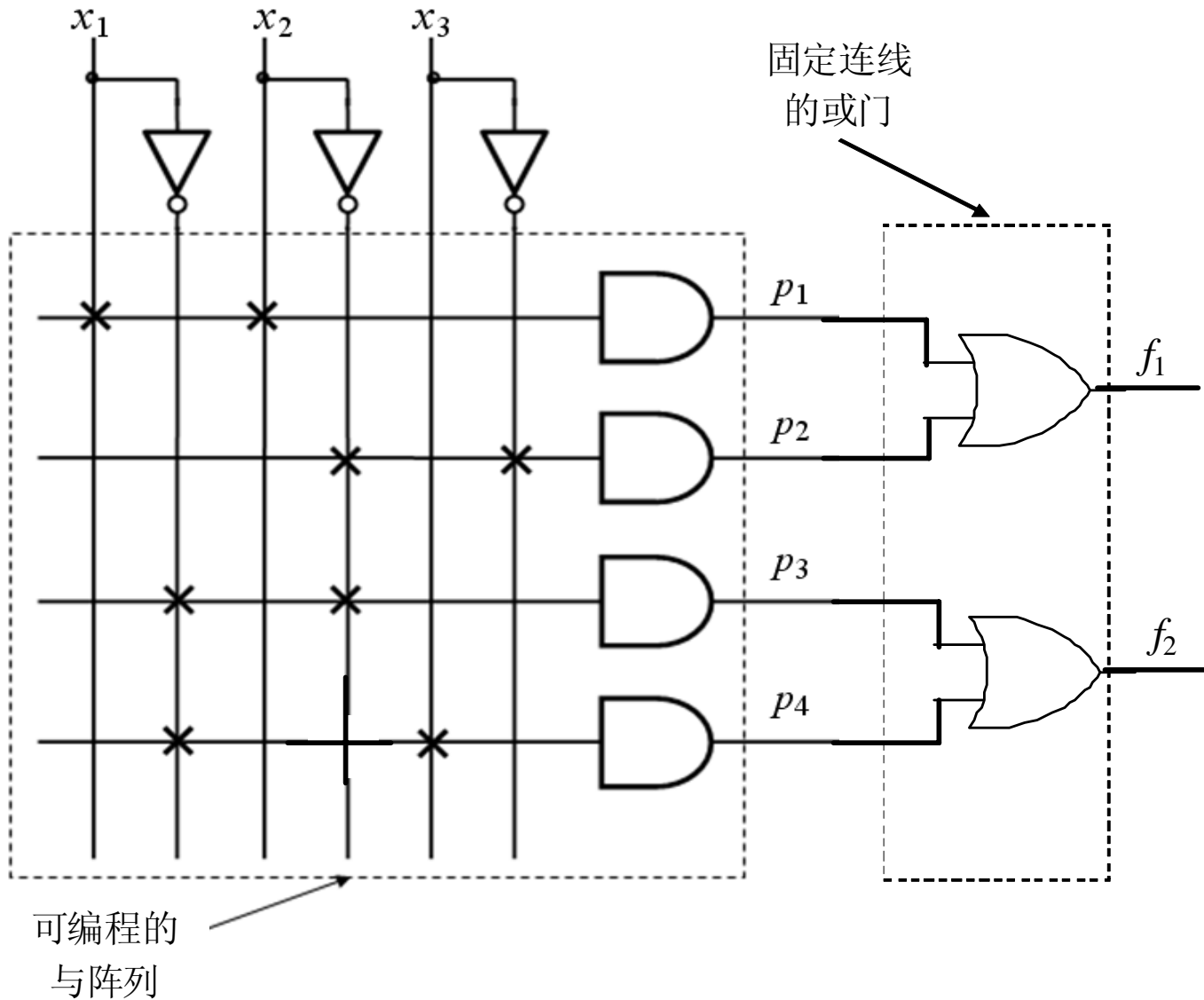
## ◆ 简单可编程逻辑器件PLA



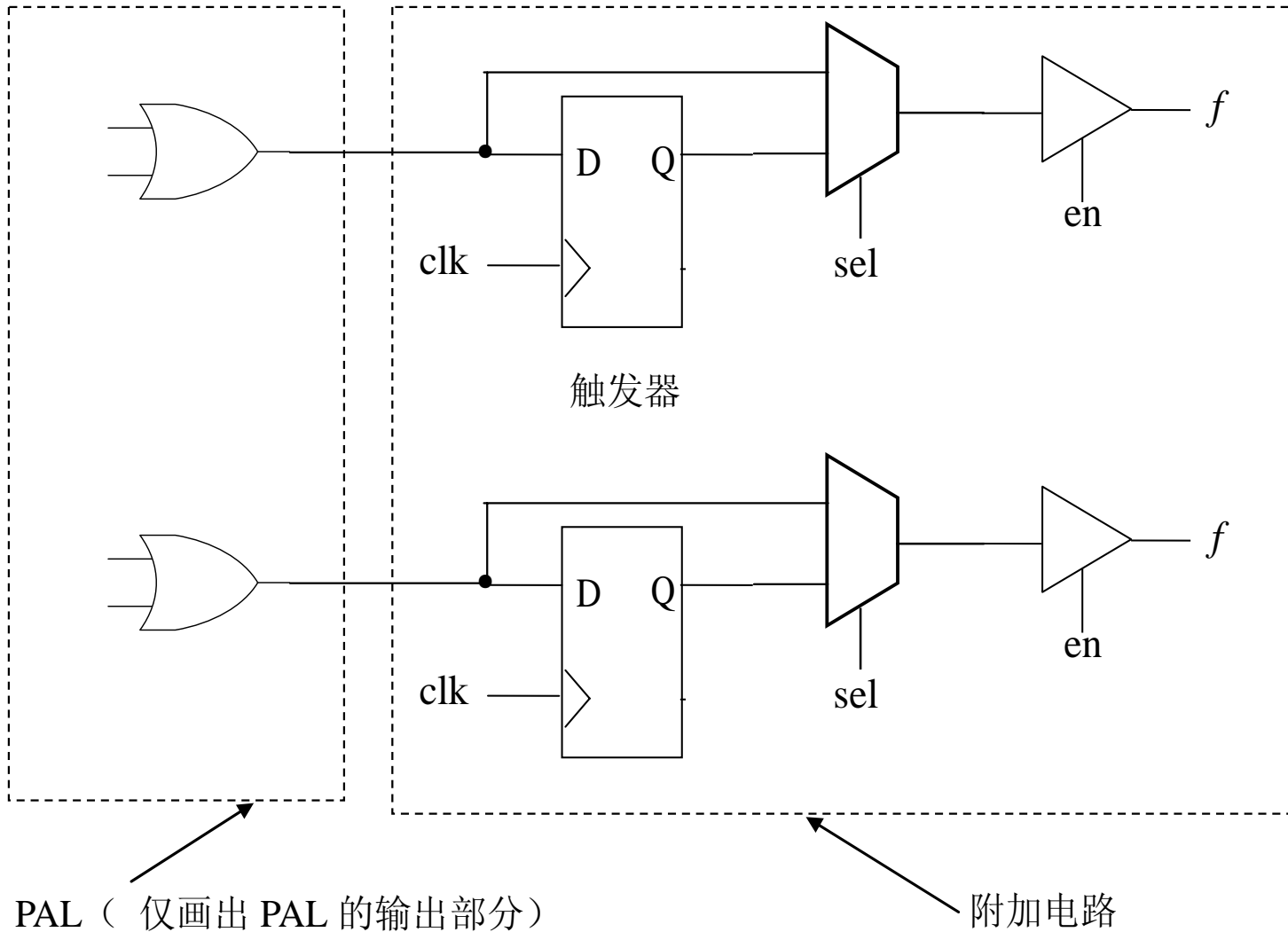
# PLA的一个具体示例



# PAL的一个具体示例



# GAL的电路结构

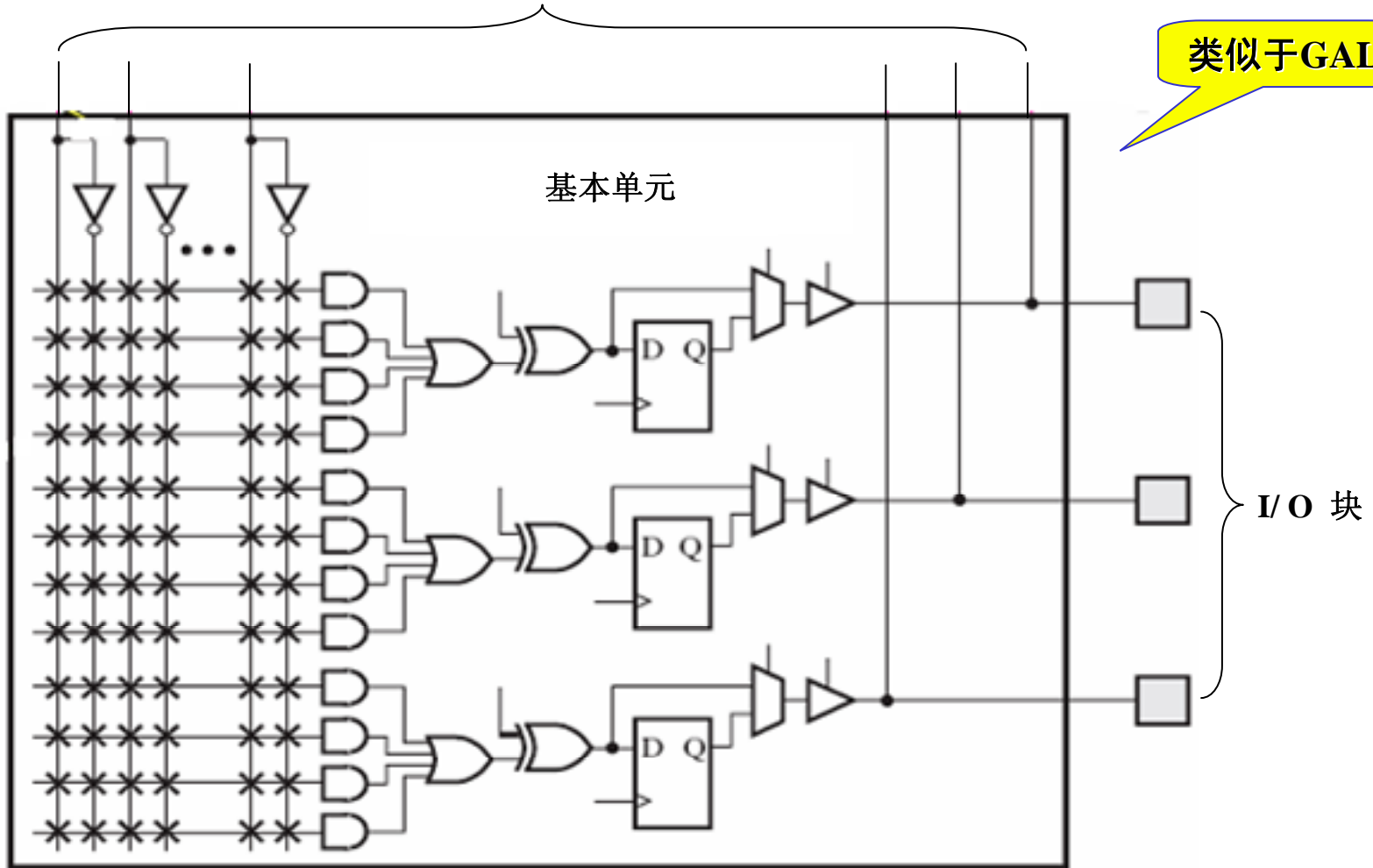


# CPLD的基本单元

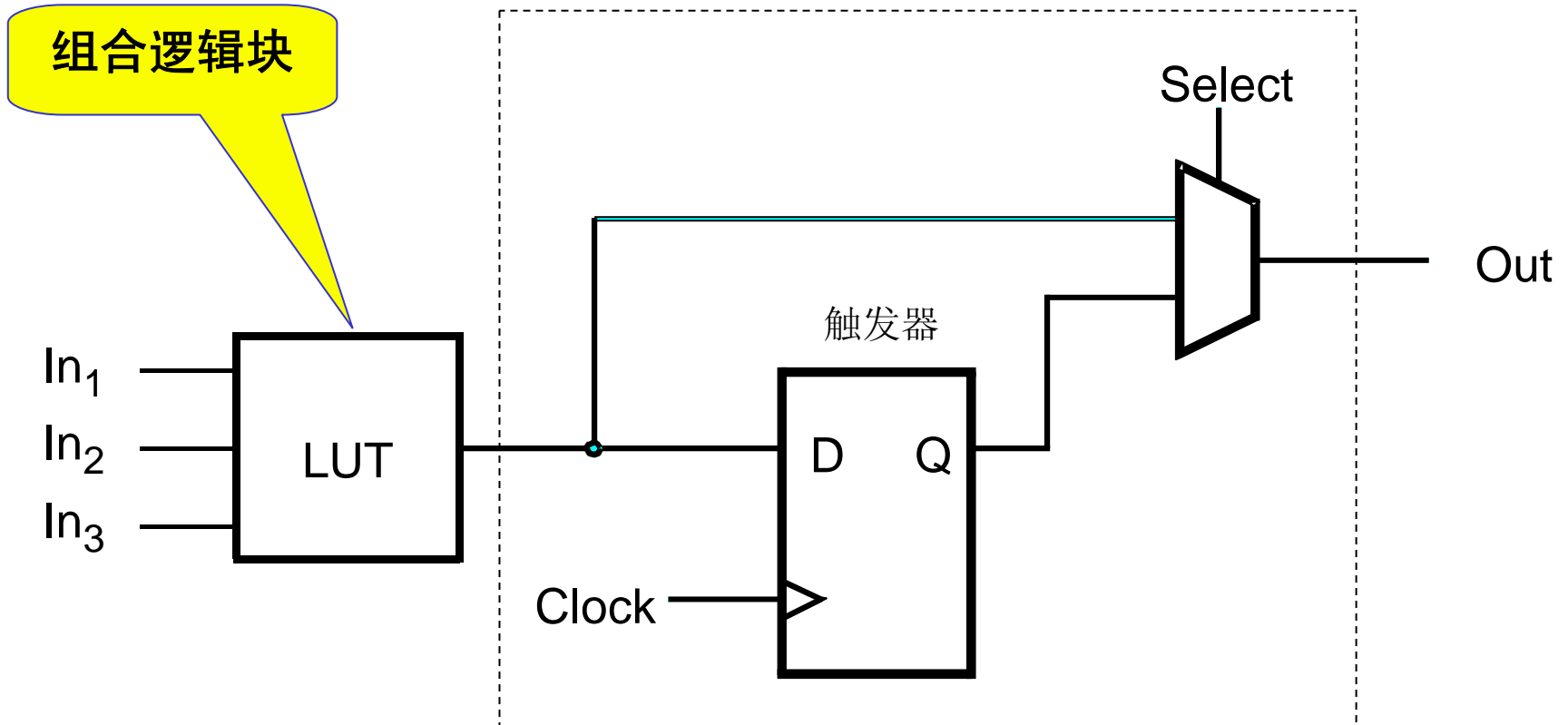


与其它基本单元相连

类似于GAL



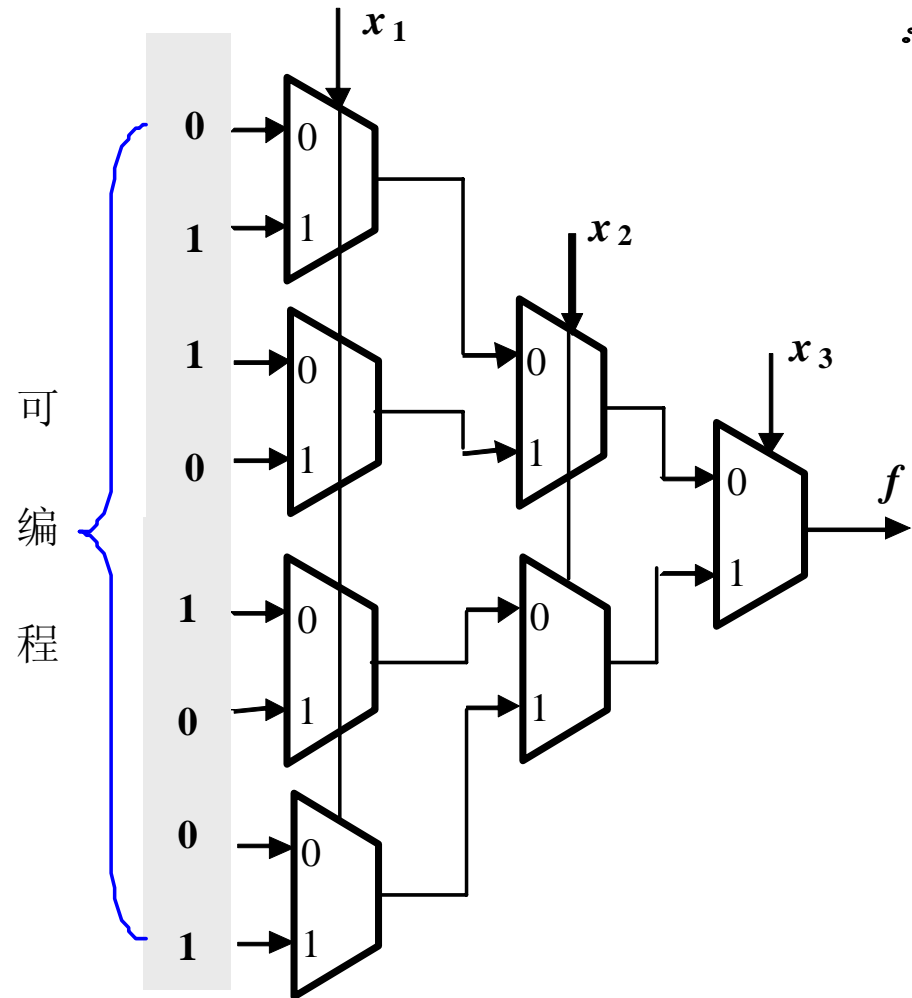
# FPGA的基本单元



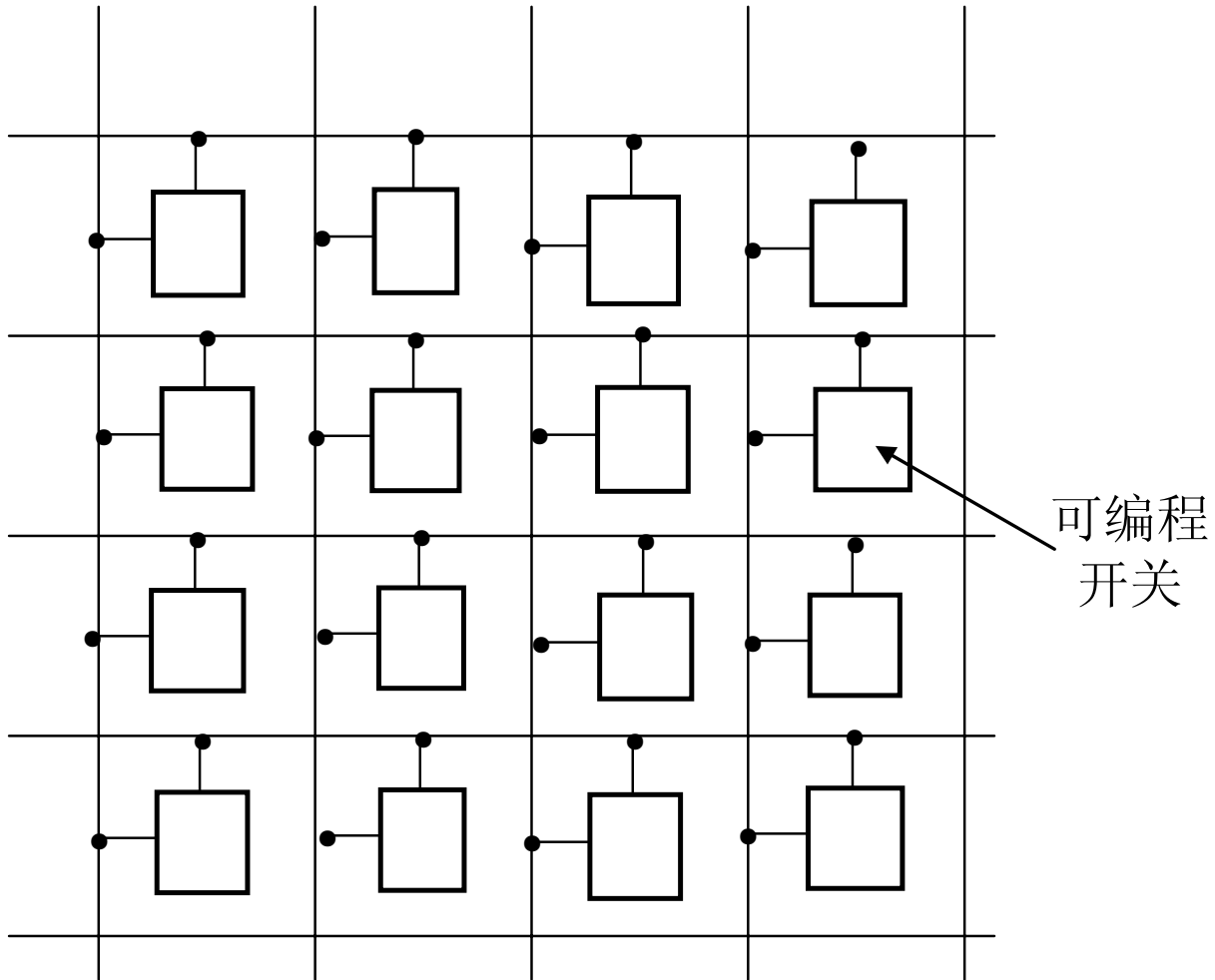
# 3输入LUT实现方案示例



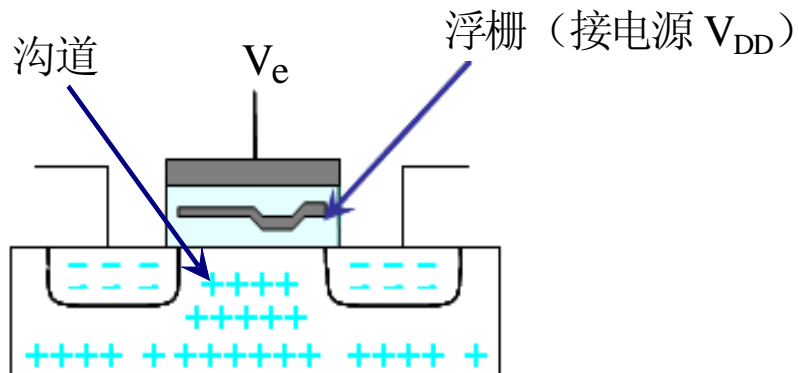
$x_3$	$x_2$	$x_1$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



# 可编程开关示意图



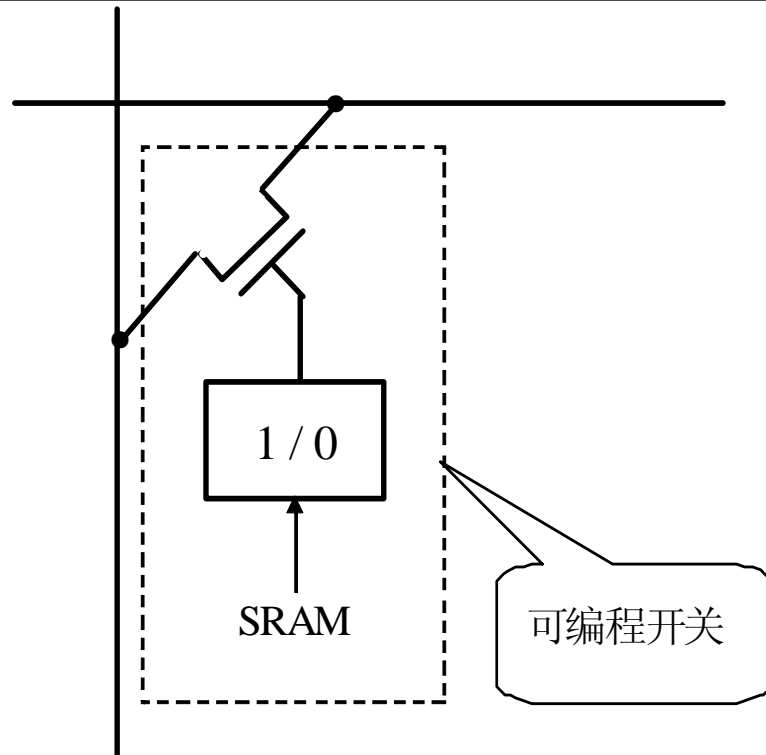
# 可编程开关的物理实现



EEPROM 晶体管

## CPLD的可编程开关

未编程前，EEPROM晶体管导通  
编程时 $V_e$ 加较高电压，晶体管截止，  
并保持此状态不变



## FPGA的可编程开关

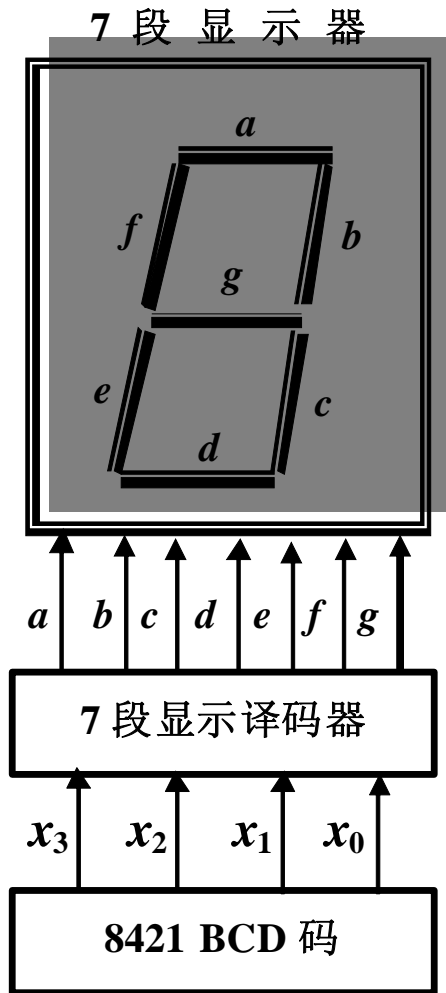
断电后SRAM中的信息消失  
下次加电后必须重新编程

# 第3章 组合逻辑电路的优化实现



- ◆ 组合逻辑电路的特点与**优化实现**
  - 卡诺图
  - VHDL
- ◆ **单输出函数和多输出函数**
- ◆ **多变量函数和多级电路**
- ◆ **组合电路积木块**
  - 多路选择器
  - 用LUT构建更大规模的组合逻辑电路
  - 编码器
  - 译码器
- ◆ **组合逻辑电路中的竞争和险象**
  - 险象的分析与消除

# 多输出函数举例



真 值 表

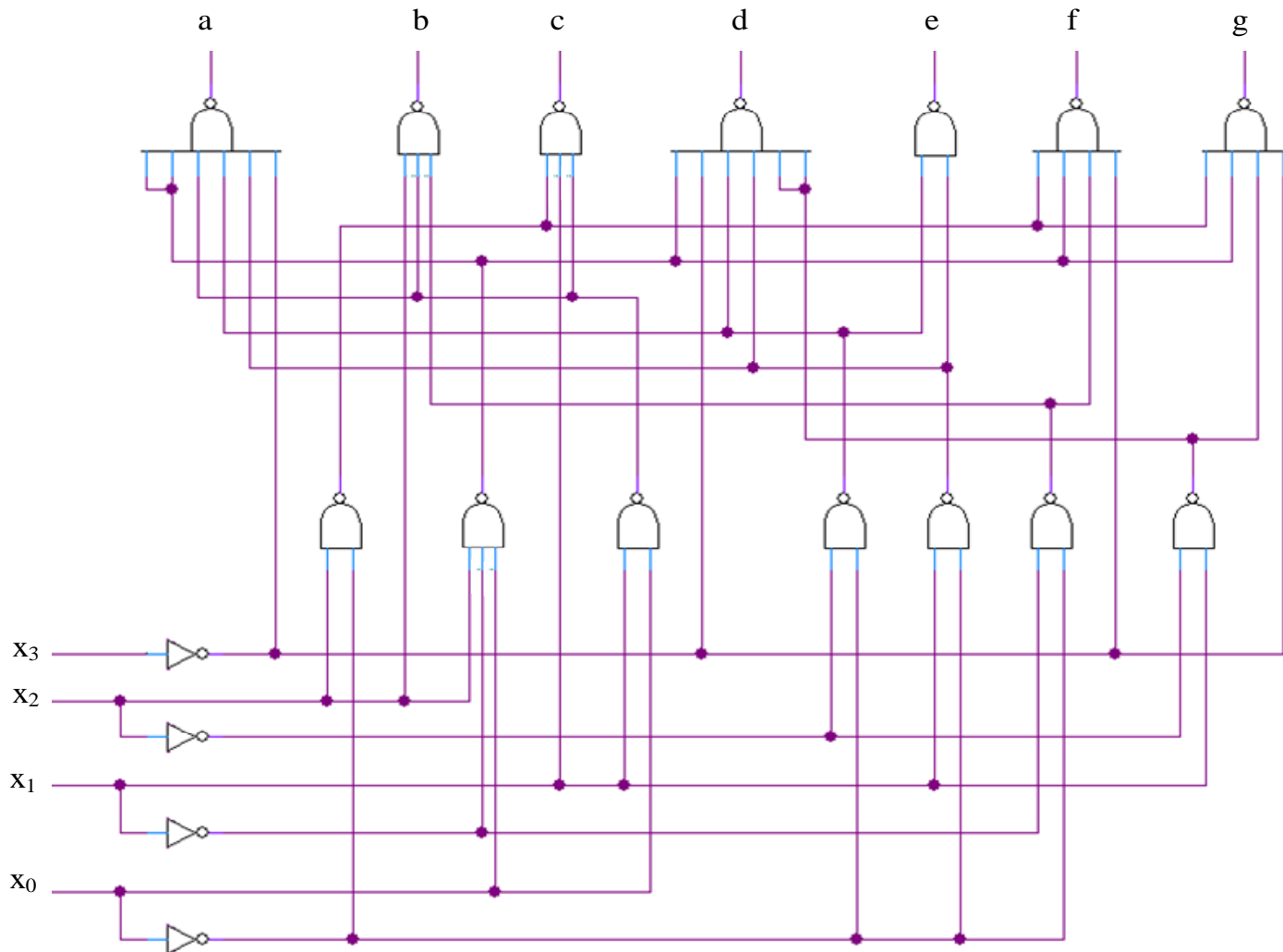
$x_3x_2x_1x_0$	$a b c d e f g$	说 明
0 0 0 0	1 1 1 1 1 1 0	十进制数 0
0 0 0 1	0 1 1 0 0 0 0	十进制数 1
0 0 1 0	1 1 0 1 1 0 1	十进制数 2
0 0 1 1	1 1 1 1 0 0 1	十进制数 3
0 1 0 0	0 1 1 0 0 1 1	十进制数 4
0 1 0 1	1 0 1 1 0 1 1	十进制数 5
0 1 1 0	1 0 1 1 1 1 1	十进制数 6
0 1 1 1	1 1 1 0 0 0 0	十进制数 7
1 0 0 0	1 1 1 1 1 1 1	十进制数 8
1 0 0 1	1 1 1 1 0 1 1	十进制数 9
1 0 1 0	d d d d d d d	不可能出现的输入组合, d 代表无关项
1 0 1 1	d d d d d d d	
1 1 0 0	d d d d d d d	
1 1 0 1	d d d d d d d	
1 1 1 0	d d d d d d d	
1 1 1 1	d d d d d d d	



# 7段译码器的手工设计



◆ 真值表 → 逻辑表达式 → 化简 → 原理图



# 7段译码器的VHDL行为描述



```
LIBRARY ieee; -- 1 打开 ieee 库
USE ieee.std_logic_1164.ALL; -- 2 打开此程序包，使类型 std_logic 和 std_logic_vector 可用
ENTITY decoder7 IS -- 3 实体声明，用于定义端口
    PORT( x3, x2, x1, x0 : IN std_logic; -- 4
          a, b, c, d, e, f, g: OUT std_logic ); -- 5
END decoder7; -- 6

ARCHITECTURE comb1 OF decoder7 IS -- 7 结构体用于定义电路的行为
    SIGNAL xx : std_logic_vector( 3 DOWNTO 0 ); -- 8 声明 1 个内部信号 xx
    SIGNAL yy : std_logic_vector( 6 DOWNTO 0 ); -- 9 声明 1 个内部信号 yy
BEGIN -- 10
```

# 7段译码器的VHDL行为描述（续）

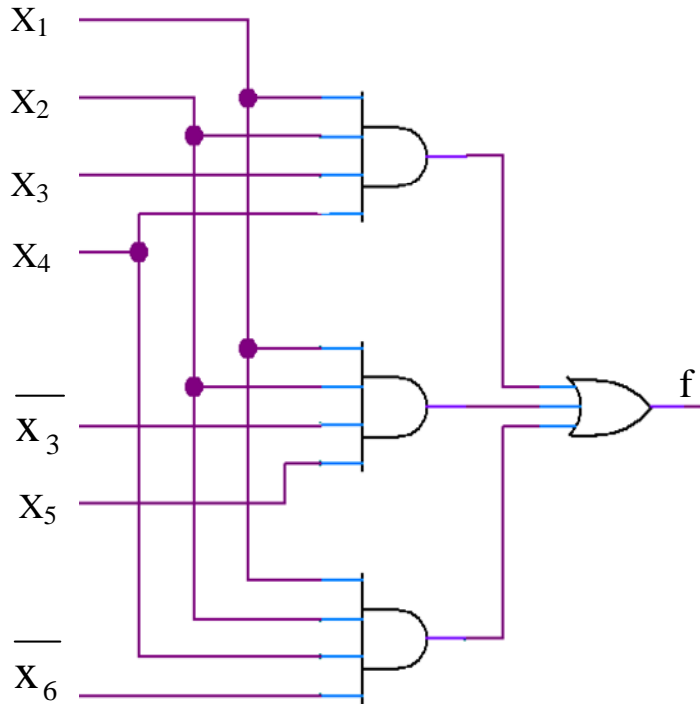


```
xx <= x3 & x2 & x1 & x0;
WITH xx SELECT
  yy <= "1111110" WHEN "0000",
      "0110000" WHEN "0001",
      "1101101" WHEN "0010",
      "1111001" WHEN "0011",
      "0110011" WHEN "0100",
      "1011011" WHEN "0101",
      "1011111" WHEN "0110",
      "1110000" WHEN "0111",
      "1111111" WHEN "1000",
      "1111011" WHEN "1001",
      "-----" WHEN OTHERS;
a <= yy(6);
b <= yy(5);
c <= yy(4);
d <= yy(3);
e <= yy(2);
f <= yy(1);
g <= yy(0);
END comb1;
```

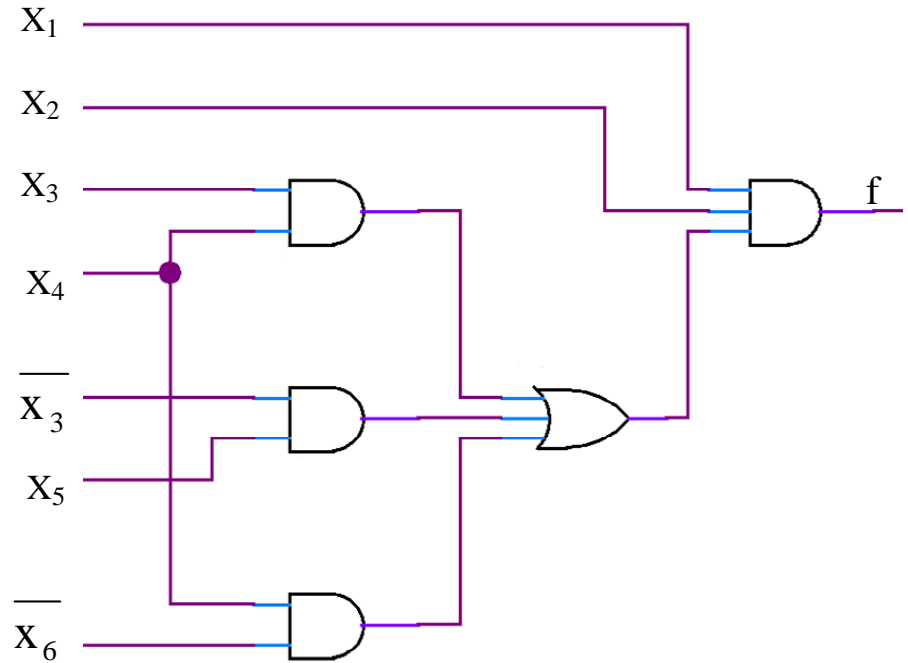
-- 11 &是连接符, 用于将信号 x3x2x1x0 连接成1个位串  
-- 12 并行条件赋值语句, 描述译码器的功能  
-- 13 输入信号取值为"0000"时的输出  
-- 14 输入信号取值为"0001"时的输出  
-- 15 输入信号取值为"0010"时的输出  
-- 16 输入信号取值为"0011"时的输出  
-- 17 输入信号取值为"0100"时的输出  
-- 18 输入信号取值为"0101"时的输出  
-- 19 输入信号取值为"0110"时的输出  
-- 20 输入信号取值为"0111"时的输出  
-- 21 输入信号取值为"1000"时的输出  
-- 22 输入信号取值为"1001"时的输出  
-- 23 输入信号取值为其他值时的输出为 don't care  
-- 24 把输出信号值赋给输出端口 a  
-- 25 把输出信号值赋给输出端口 b  
-- 26 把输出信号值赋给输出端口 c  
-- 27 把输出信号值赋给输出端口 d  
-- 28 把输出信号值赋给输出端口 e  
-- 29 把输出信号值赋给输出端口 f  
-- 30 把输出信号值赋给输出端口 g  
-- 31



# 多级电路的优化实现



(a) 2级逻辑电路



(b) 多级逻辑电路

# 怎样用4输入LUT实现多变量函数？



## ◆ 香农展开定理：

$$\begin{aligned} & f(x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \\ &= \overline{x_i} f_0(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) + x_i f_1(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \end{aligned}$$

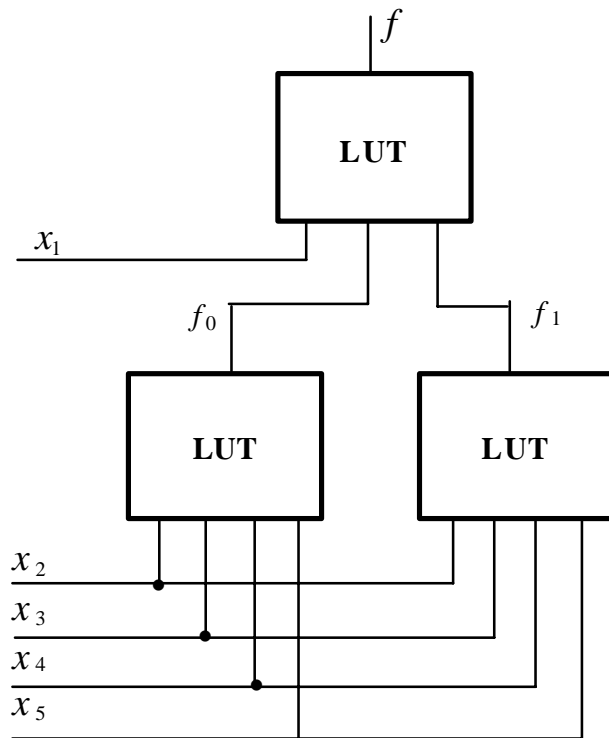
- 子函数  $f_0$  和  $f_1$  所包含的变量都比函数  $f$  减少了1个；
- 可以持续地应用香农展开定理，直到满足要求时为止。
- 当变量个数很多时，先对哪一个变量实施展开，有技术问题（从略）。

# 实例：用4输入LUT实现5变量函数



- ◆ 对5变量函数实施香农展开：

$$f(x_1, x_2, x_3, x_4, x_5) = \overline{x_1} f_0(0, x_2, x_3, x_4, x_5) + x_1 f_1(1, x_2, x_3, x_4, x_5)$$



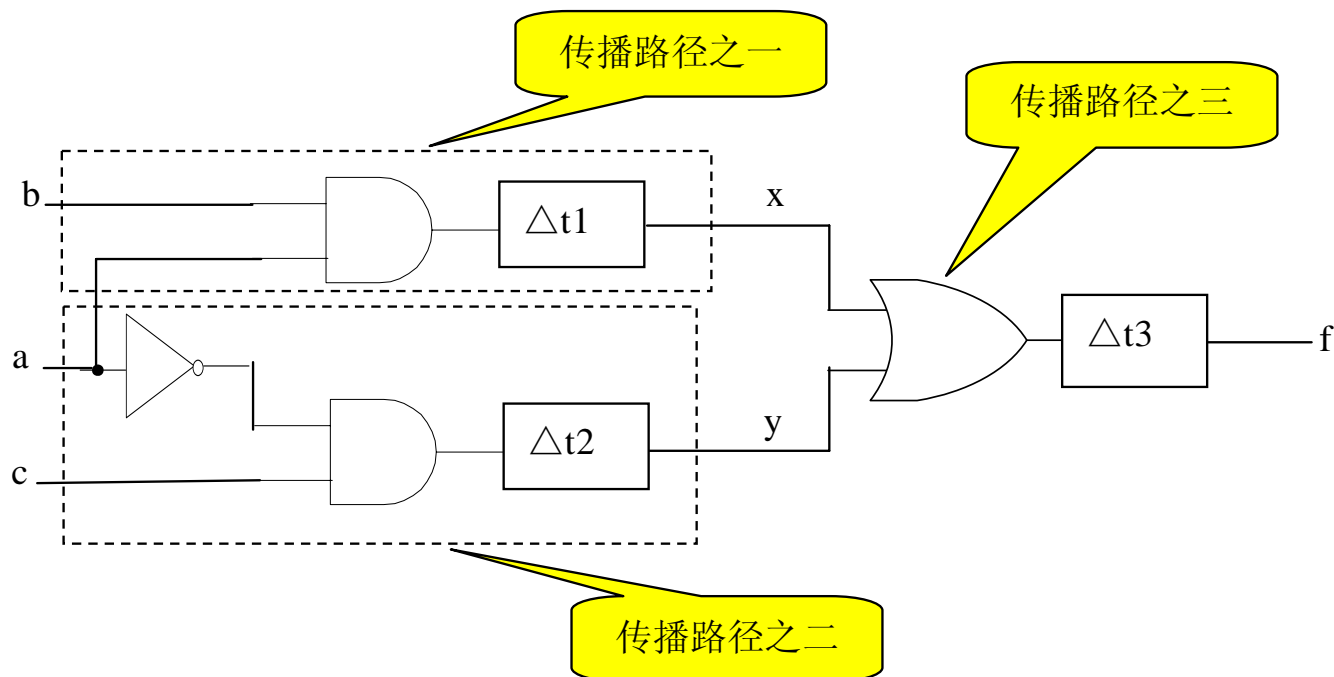
5变量函数展开为  
2个4变量函数！

- ◆ 当变量个数很多时，先对哪一个变量实施展开？有关技术从略（BDD）。

# 组合逻辑电路中的竞争和险象



- ◆ 信号传播路径上延时不同导致有先有后地到达同一门的输入端，此为竞争，导致险象（毛刺）。



$$f = ab + \bar{a}c \quad \text{当 } b=c=1 \text{ 时} \quad f = a + \bar{a} = 1$$

- ◆ 毛刺是否会产生永久性后果取决于其负载。

# 险象的消除

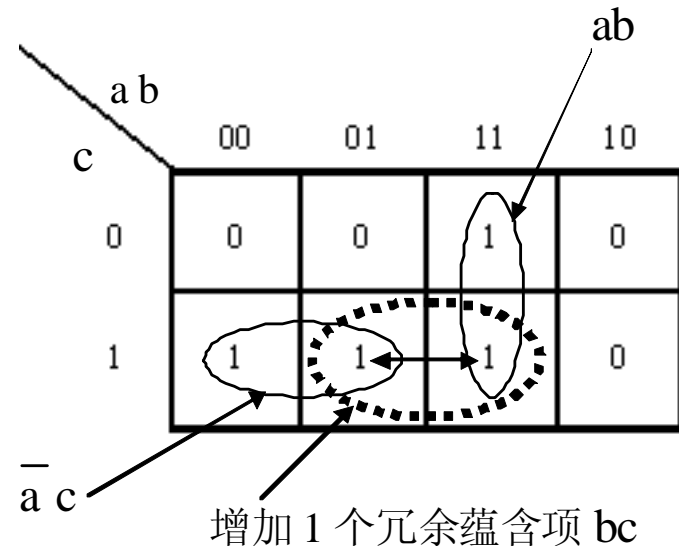
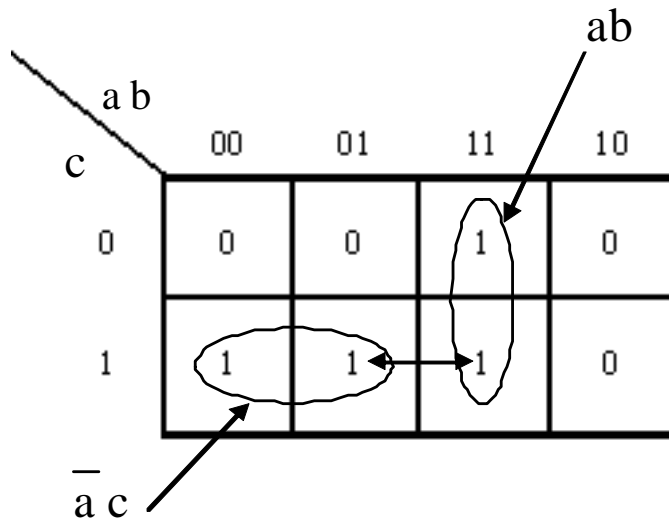


- ◆ **代数法**：设目标电路的逻辑表达式为  $f$ ，若在任何输入组合情况下都不会演变为

$$f = a + \bar{a} = 1 \qquad f = a \cdot \bar{a} = 0$$

则不会产生险象（毛刺）。

- ◆ **卡诺图法**：卡诺图中维块之间的关系可分为**相交**、**分离**和**相邻**3种，通过增添冗余维块可**消除相邻维块**，从而消除险象发生的根源。



# 第4章 数的表示方法和算术运算电路



## ◆ 数制和编码

- 十进制数、二进制数、有符号数、无符号数、格雷码、奇偶校验码 .....

## ◆ 无符号数的加法运算

- 二进制数 / 十进制数

## ◆ 有符号数的表示方法和算术运算

- 原码 / 补码 / 反码 的加减运算
  - 给出运算规则和实例，但略去证明

## ◆ 用EDA工具设计算术运算电路示例

(1) 基本原理

(2) 设计工作：（VHDL + Quartus II）

# 第5章 锁存器、触发器和寄存器



## ◆ 锁存器

- 基本R-S锁存器
- 选通D锁存器

## ◆ D触发器

- 从总体的角度观察D触发器
- D触发器和D锁存器的比较
- 带使能控制的D触发器

## ◆ 主从D触发器

## ◆ 其它类型的触发器

- T触发器
- JK触发器

## ◆ 寄存器

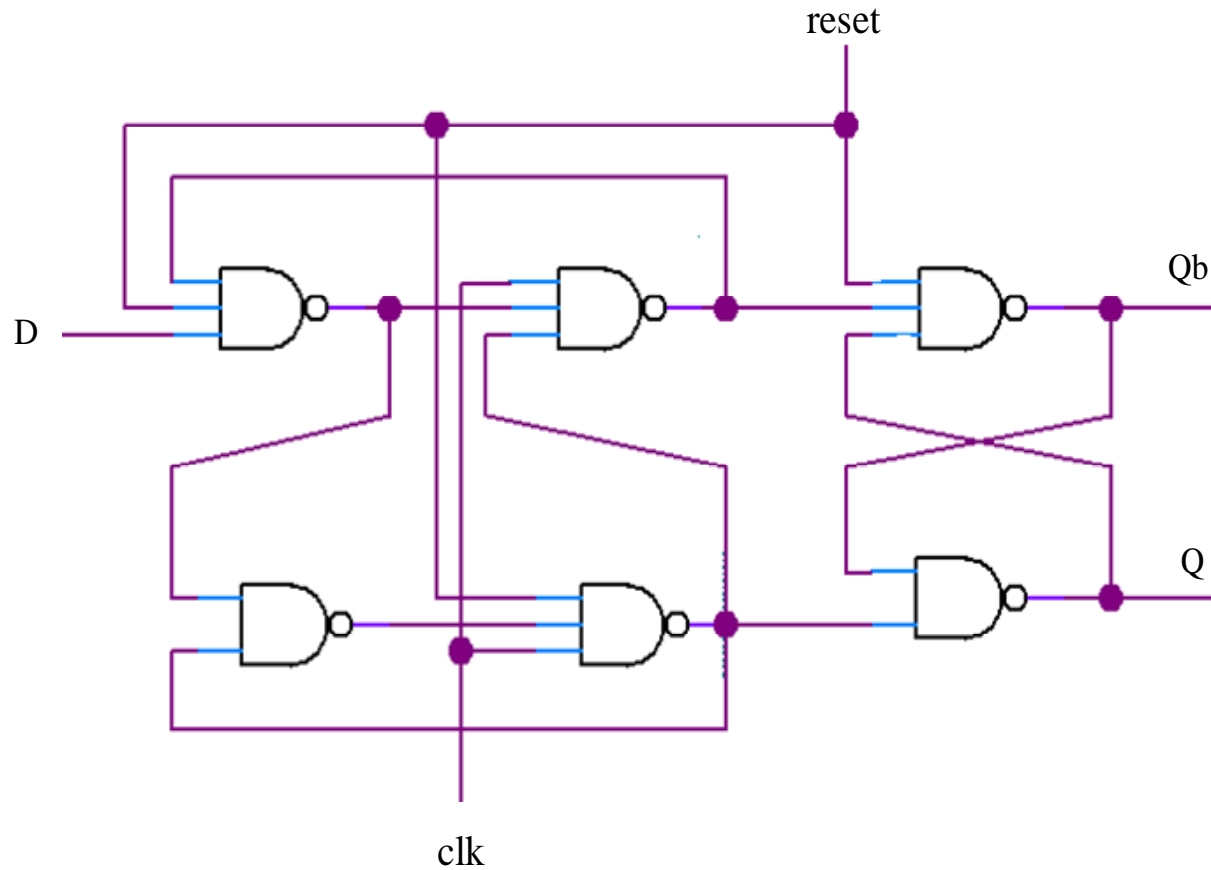
## ◆ 设计示例

不去讲解维持-阻塞，  
只讲外特性  
原理图的导出留给第7章

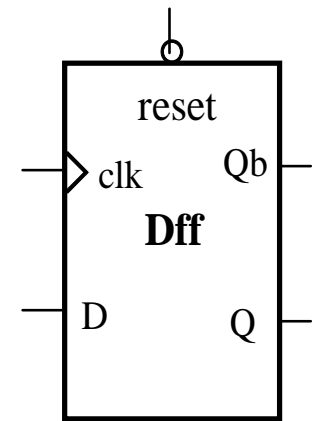
用波形图比较二者的外特性，  
从对比中加深认识

对D触发器稍加演变即成

# D触发器原理图



(a) 电路图



(b) 符号

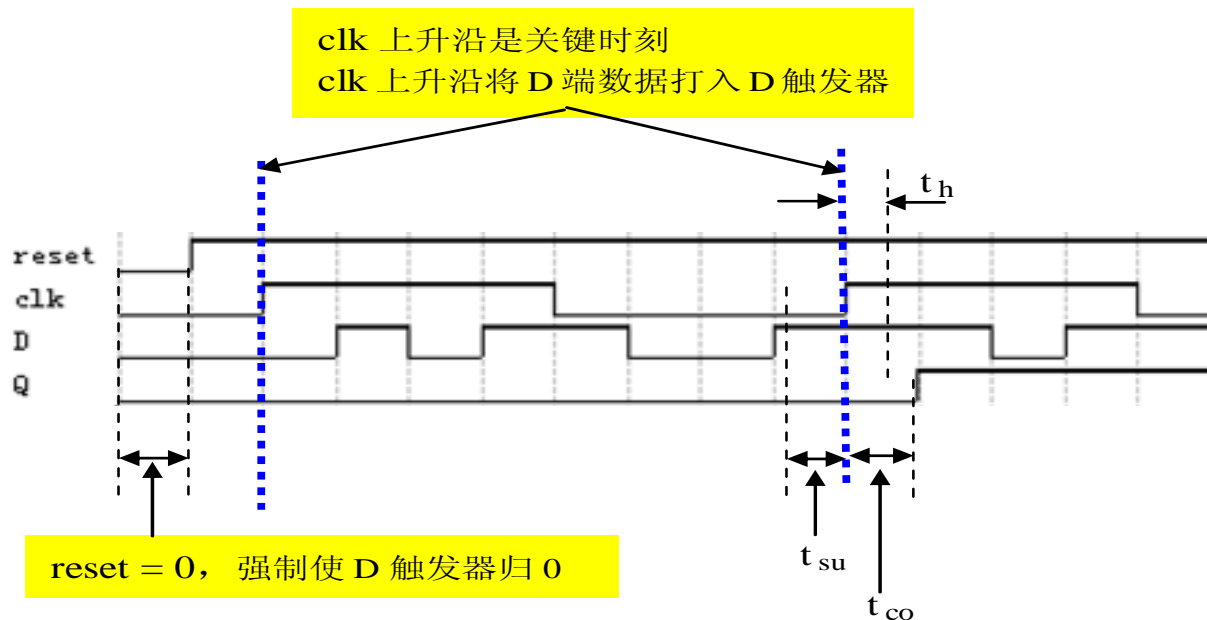
# 用功能表 / 波形图说明D触发器特性



功能表

输入			次态 $Q^{n+1}$	功能解释
reset	clk	$D^n$		
0	X	X	0	reset 低电平有效，实现异步清零
1	nr	X	$Q^n$	clk 未发生正跳变时，D 触发器保持原状态不变
	↑	0	0	clk 上升沿将 D 端数据打入 D 触发器
		1	1	

说明： clk = ↑ 表示 clk 正跳变； clk = nr 表示 clk 未发生正跳变； X 表示取值任意；  
 $D^n$  代表触发器 D 端的当前值。



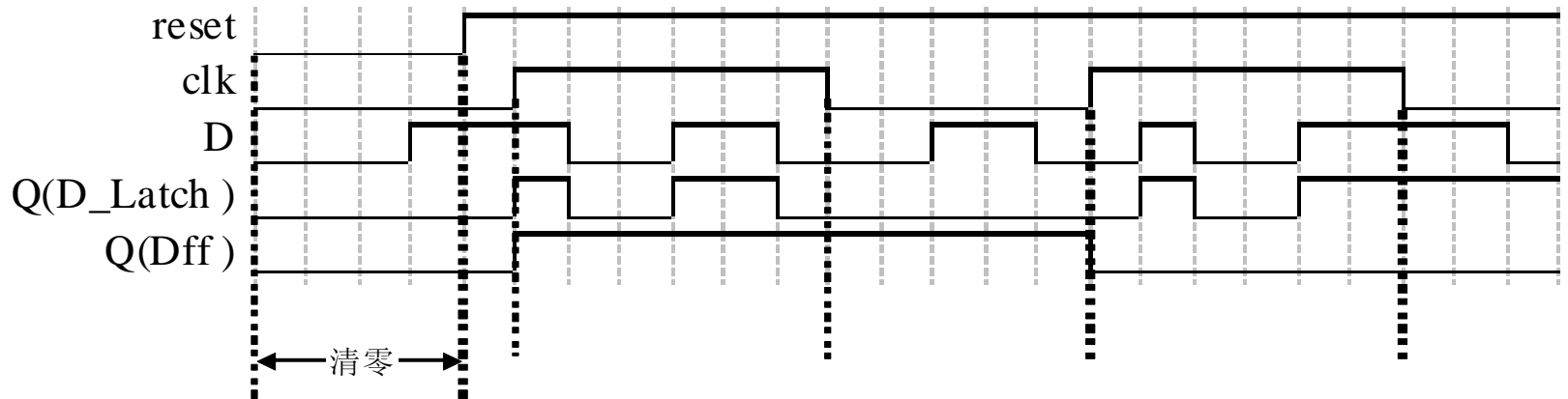
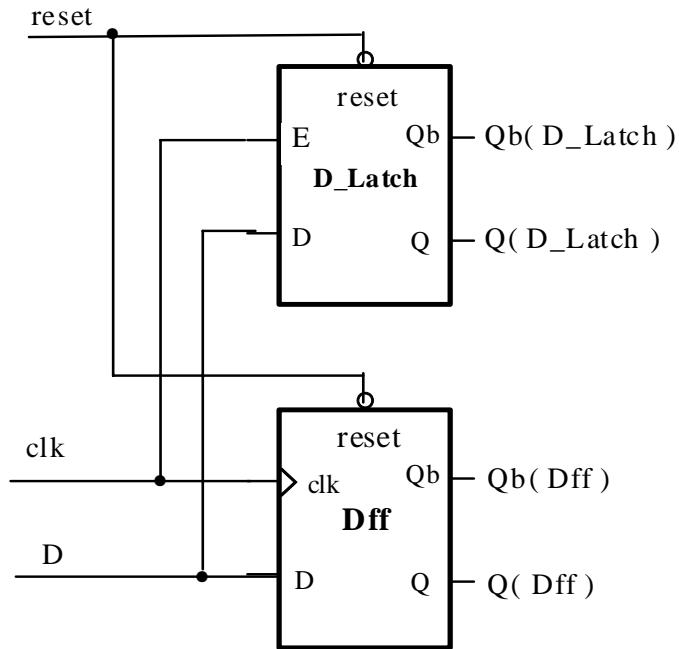
# D触发器的VHDL行为描述



```
ENTITY d_flip_flop IS -- 实体声明， 用于定义D 触发器的端口
    PORT( reset , d , clk : IN Bit; -- 输入端口
          q , qb : OUT Bit ); -- 输出端口
END d_flip_flop;

ARCHITECTURE behav OF d_flip_flop IS -- 结构体用于定义D 触发器的行为
BEGIN
    PROCESS( reset , clk ) -- 进程本身是并行语句，进程内部使用顺序语句
    BEGIN -- reset 和 clk 是其敏感信号
        IF ( reset = '0' ) THEN -- reset 为异步清零信号，低电平有效
            Q <= '0';
            Qb <= '1';
        ELSIF ( clk 'event AND clk = '1' ) -- 时钟正跳变将 D 端数据打入触发器
            q <= d;
            qb <= NOT d;
        END IF;
    END PROCESS;
END behav;
```

# D触发器和D锁存器的比较



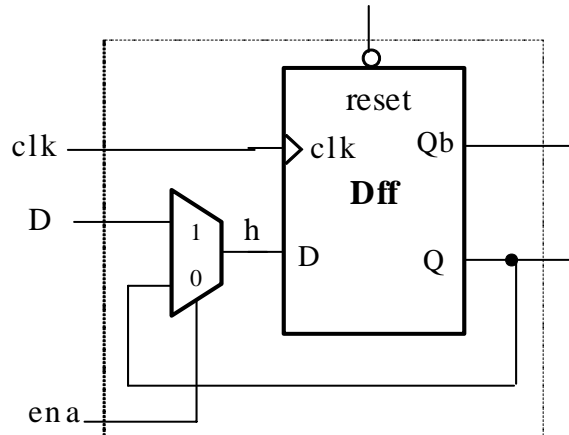


---

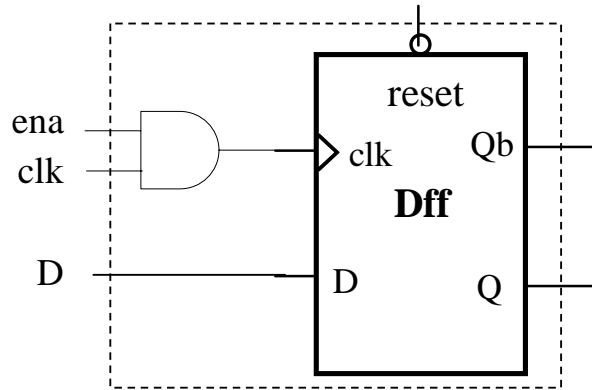
# 其它类型触发器与D触发器

-- 从**对比**中学习，不必从原理图学起 --

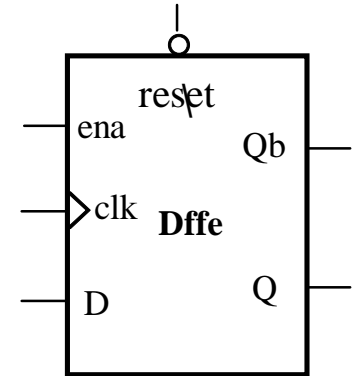
# 带使能控制的D触发器



原理图之一



原理图之二

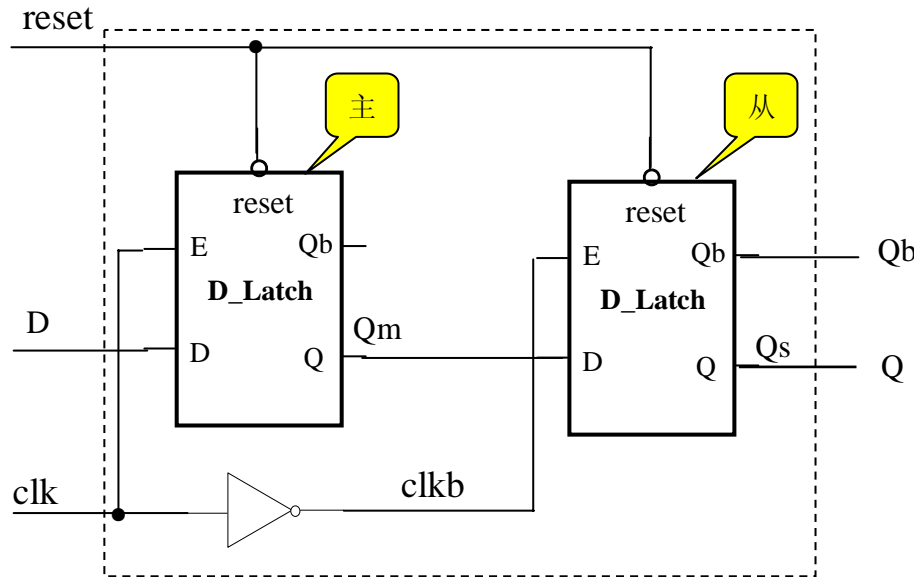


符号图

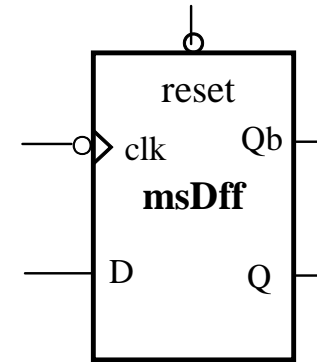
输入				次态 $Q^{n+1}$	功能解释
reset	ena	clk	$D^n$		
0	X	X	X	0	reset 低电平有效，实现异步清零
1	0	X	X	$Q^n$	ena = 0，保持原状态不变
	1	↑	0	0	clk 上升沿将 D 端数据打入 D 触发器
			1	1	

说明： clk = ↑ 表示 clk 正跳变； X 表示取值任意；  $Q^n$  代表当前状态；  $D^n$  代表触发器 D 端的当前值。

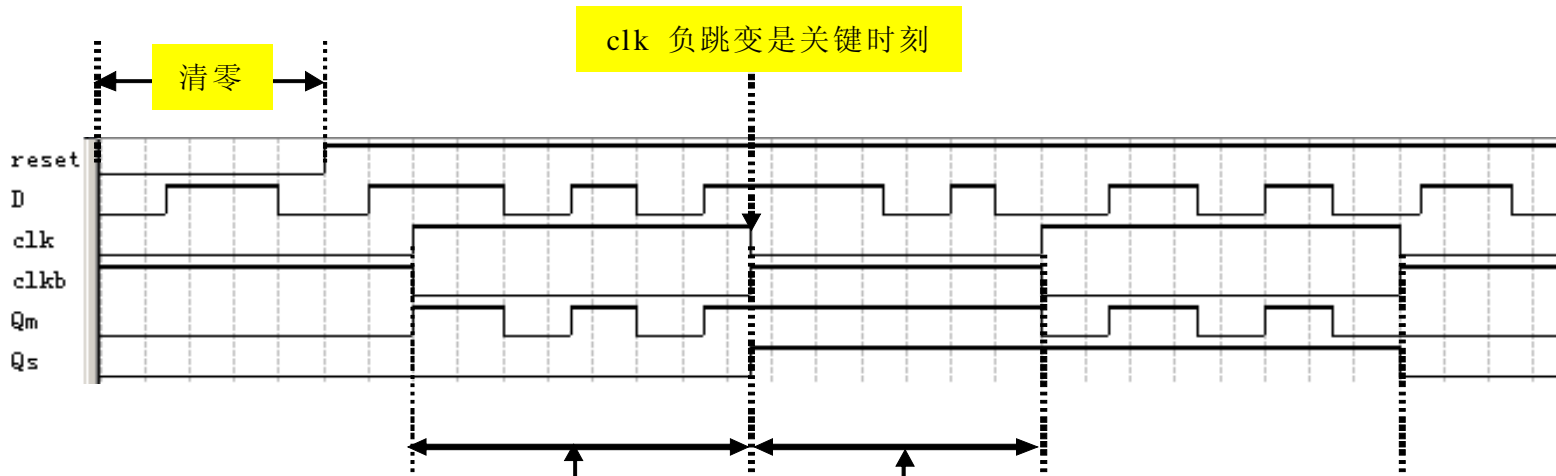
# 主从D触发器



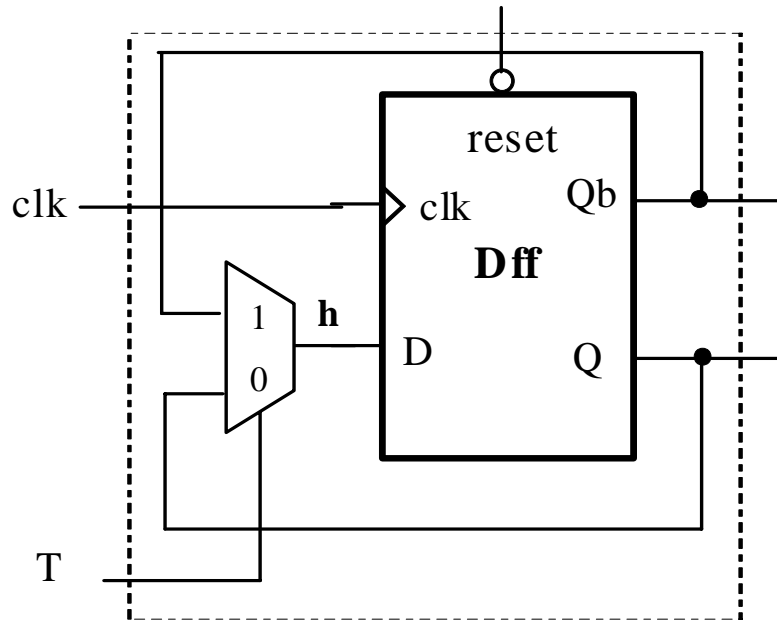
(a) 电路图



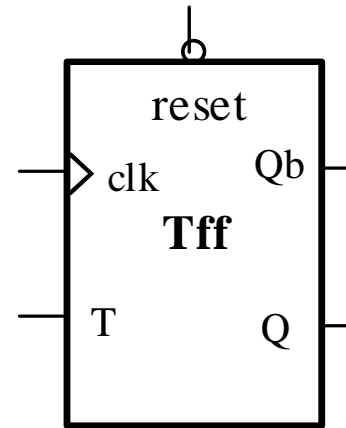
(b) 符号



# T触发器



(a) 电路图

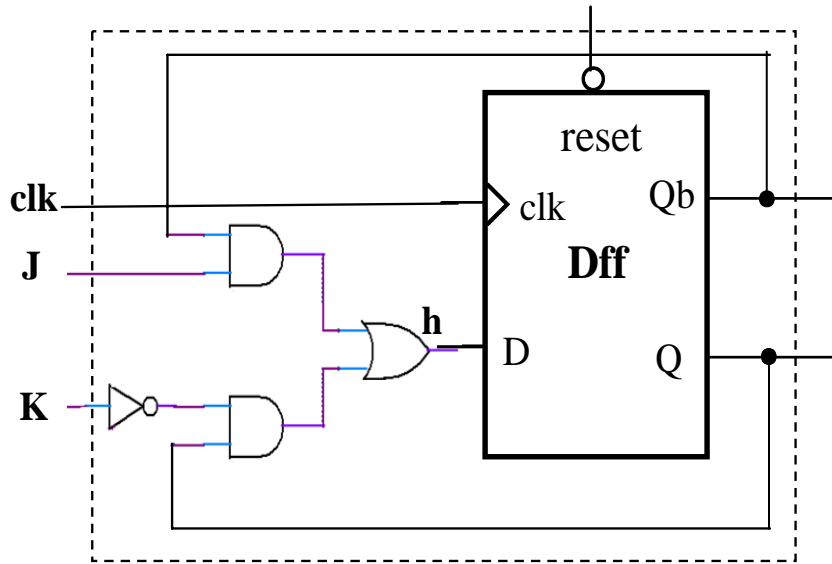


(b) 符号

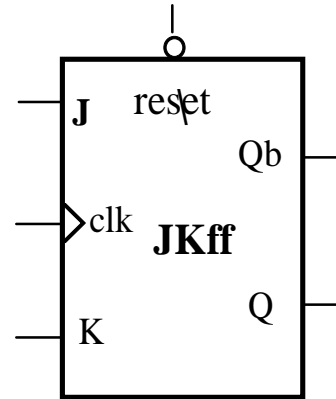
输入 clk	$T^n$	$Q^{n+1}$
↑	0	$Q^n$
↑	1	$\overline{Q^n}$
$T^n$ 代表端口 T 的当前值		

(c) 功能表

# JK触发器



(a) 电路图



(b) 符号

输入 clk $J^n$ $K^n$	$Q^{n+1}$
↑ 0 0	$Q^n$
↑ 0 1	0
↑ 1 0	1
↑ 1 1	$\overline{Q^n}$

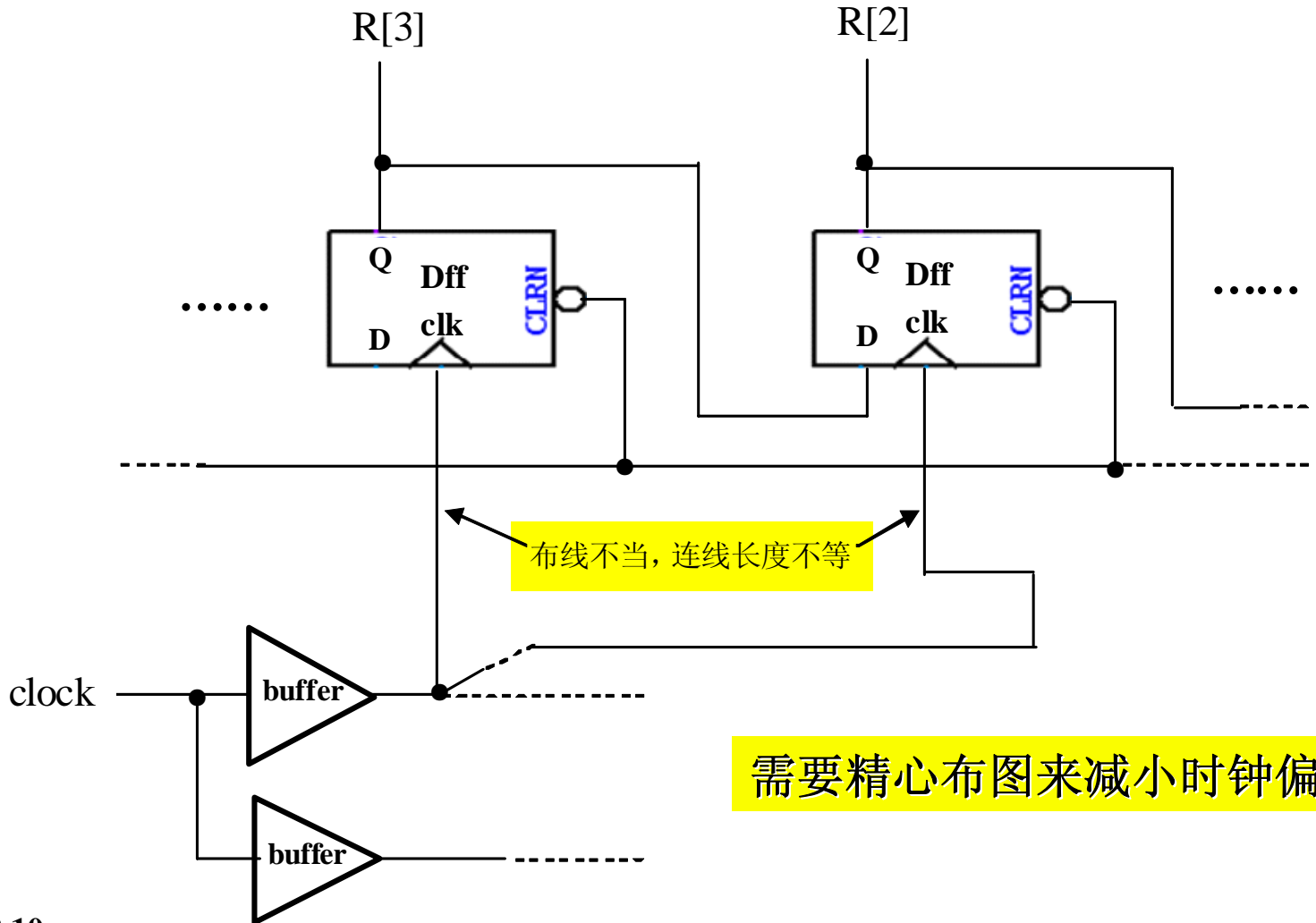
$J^n$  代表端口 J 当前值  
 $K^n$  代表端口 K 当前值

(c) 功能表

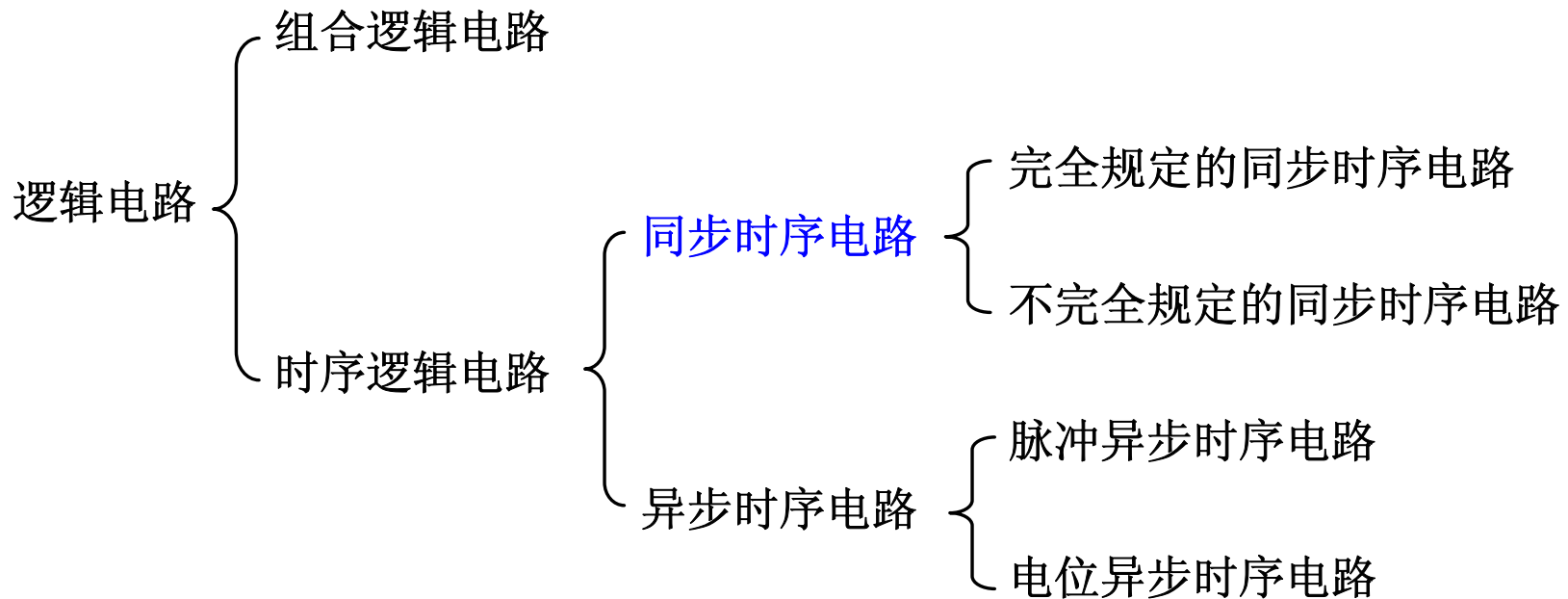
# 时钟偏移 (clock skew) 问题



- ◆ 逻辑相邻的2个触发器的时钟不一定能同时到达:



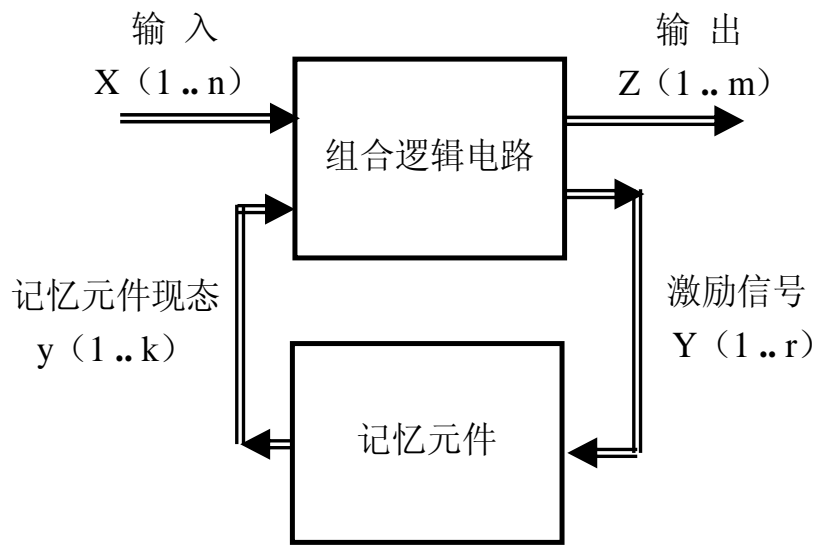
# 第6章 同步时序电路



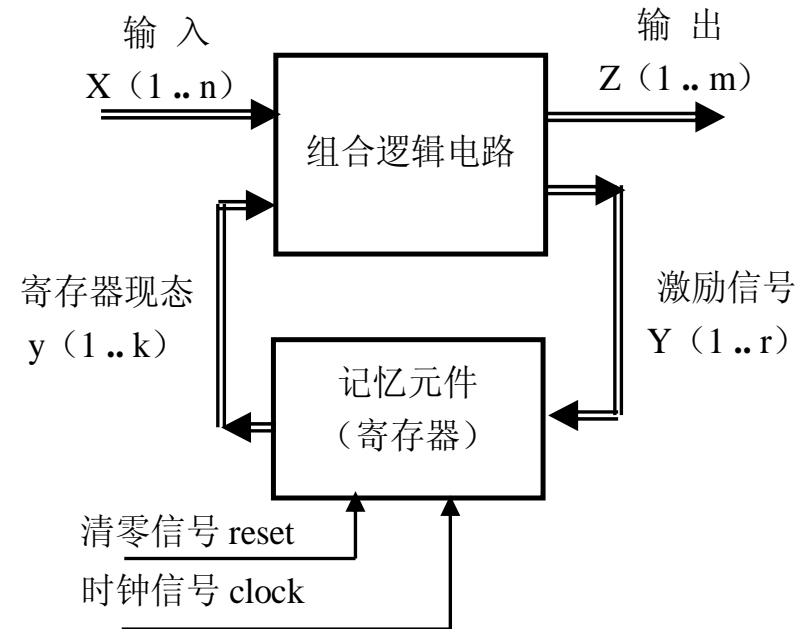
# 时序电路的电路模型



## ◆ 时序电路的数学模型是有限状态机 (finite state machine, FSM)

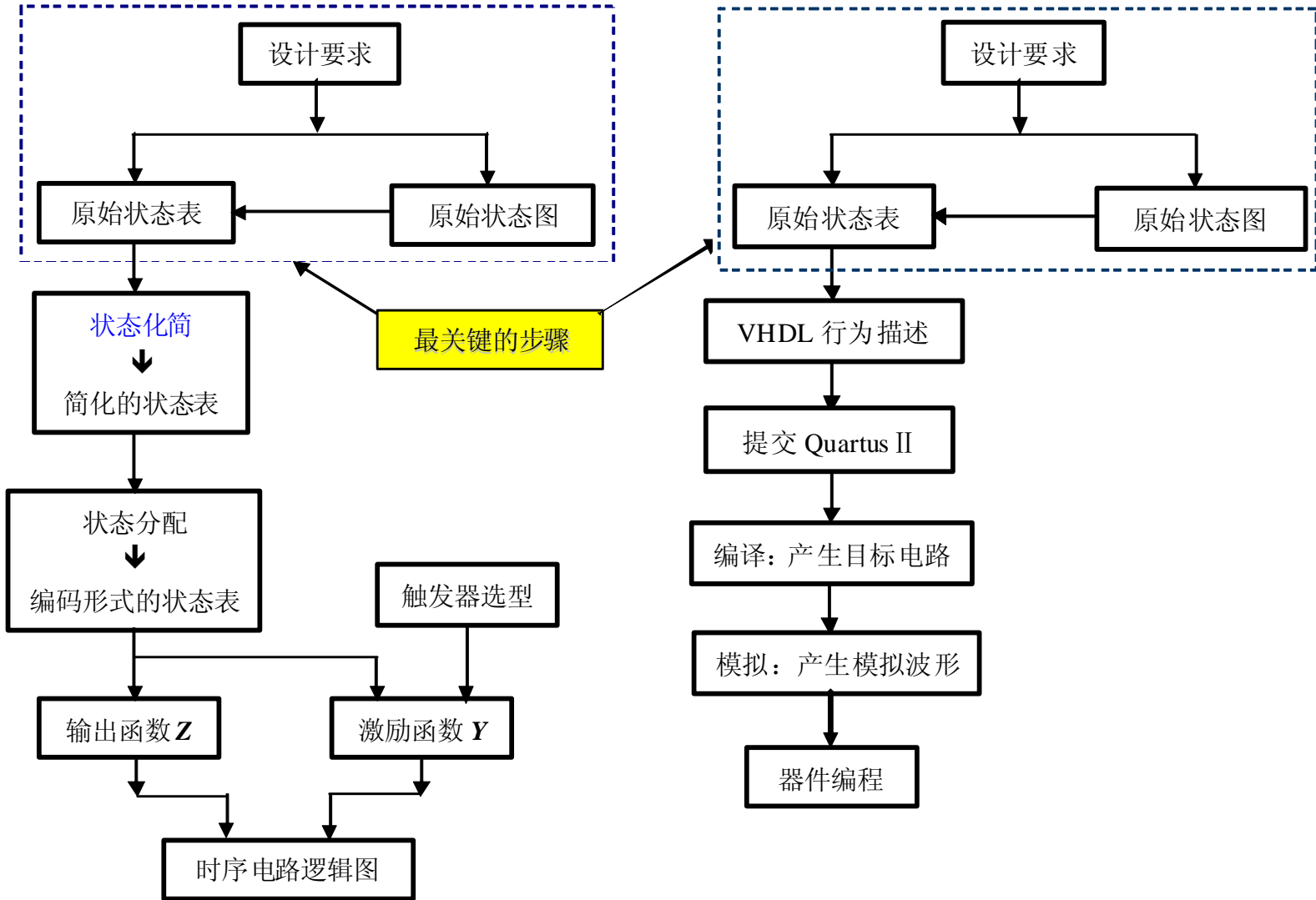


(a) 时序电路通用模型

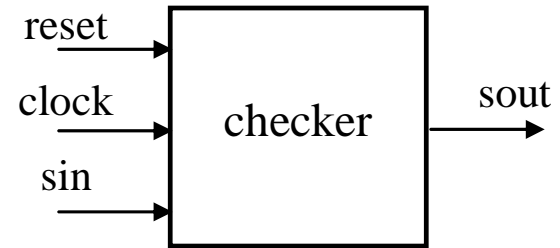


(b) 同步时序电路模型

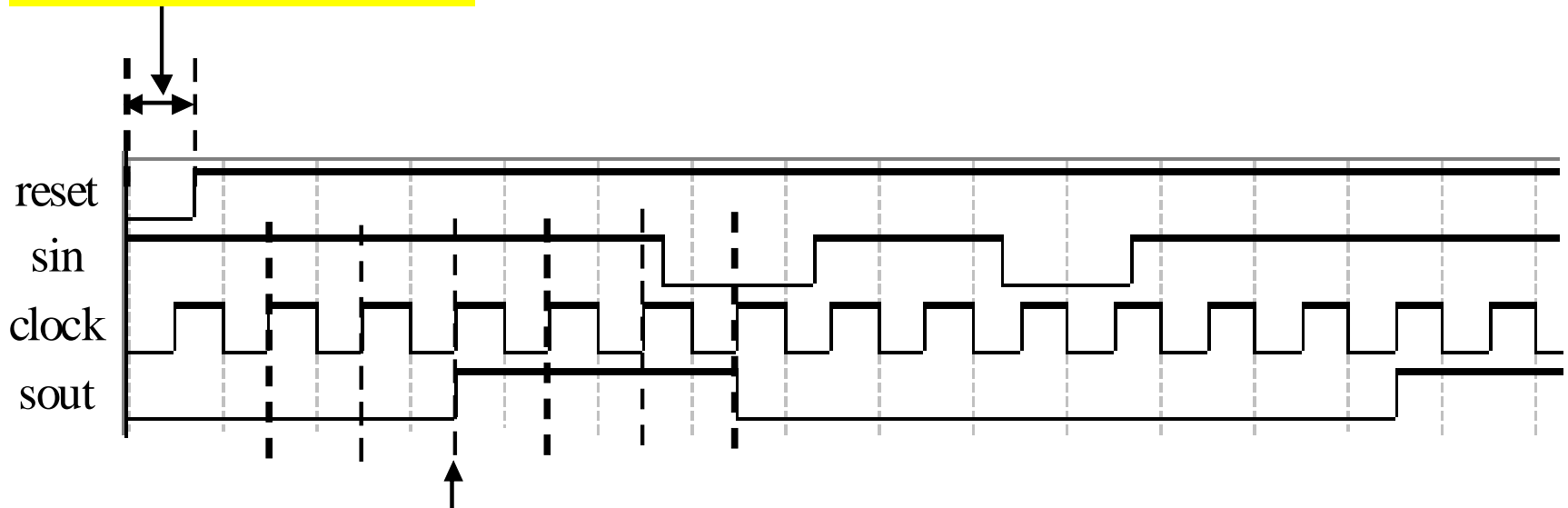
# 时序电路的设计步骤



# 简单实例：输入信号电平检测器



Reset = 0, 强制进入初始状态

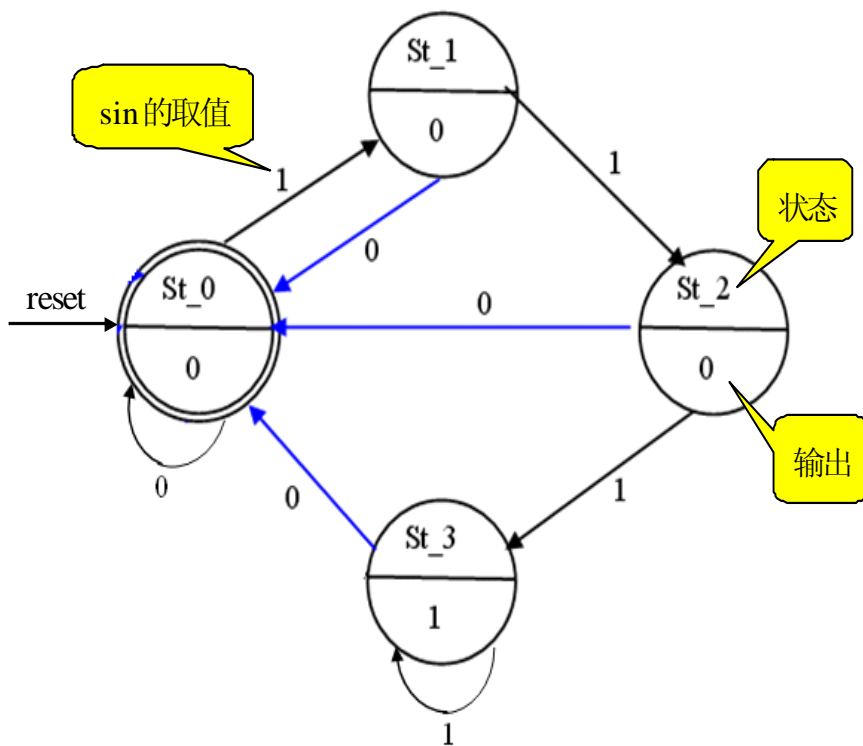


clock 上升沿检测 sin 的电平，连续 3 次或 3 次以上检测到高电平，输出 sout 为 1

# 检测器的Moore型状态图



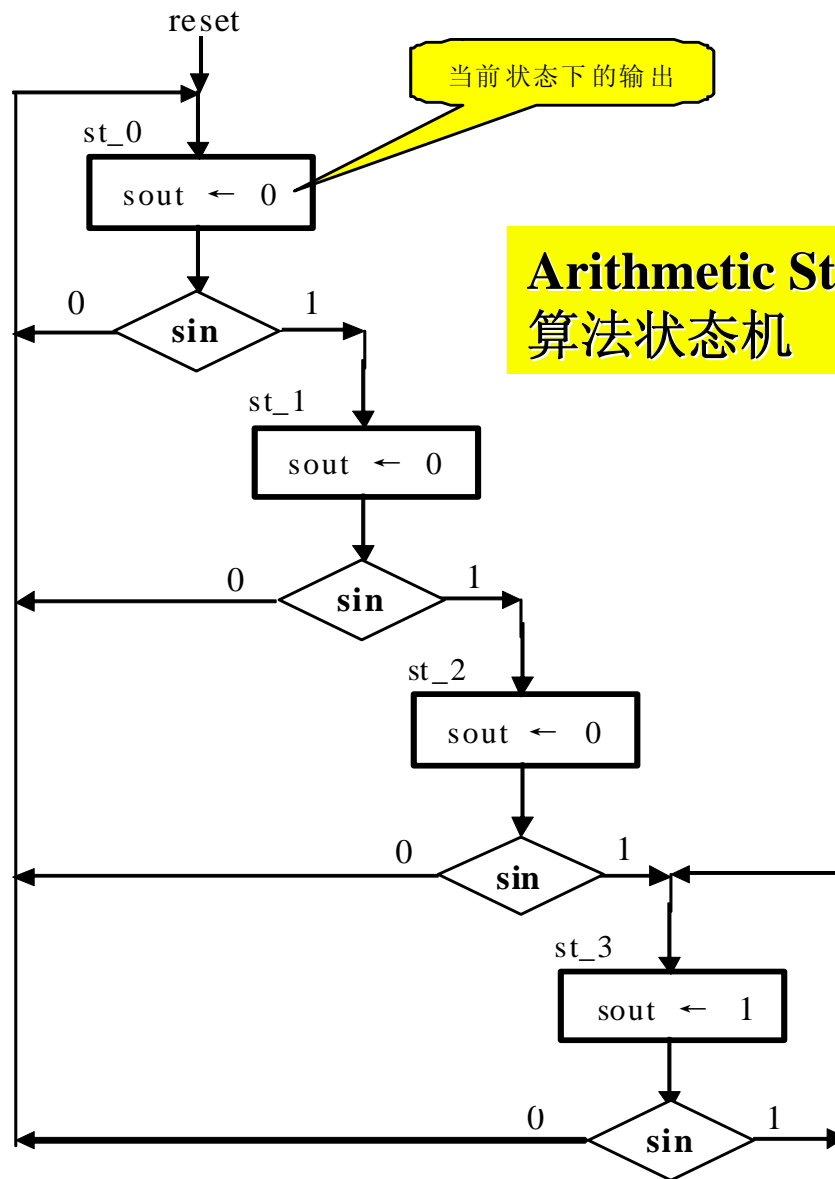
## ◆ 输出取值仅和当前状态有关



状态表

输入 sin	现态	输出 sout	次态
0	St_0	0	St_0
1			St_1
0	St_1	0	St_0
1			St_2
0	St_2	0	St_0
1			St_3
0	St_3	1	St_0
1			St_3

# 与Moore型状态图等价的ASM图

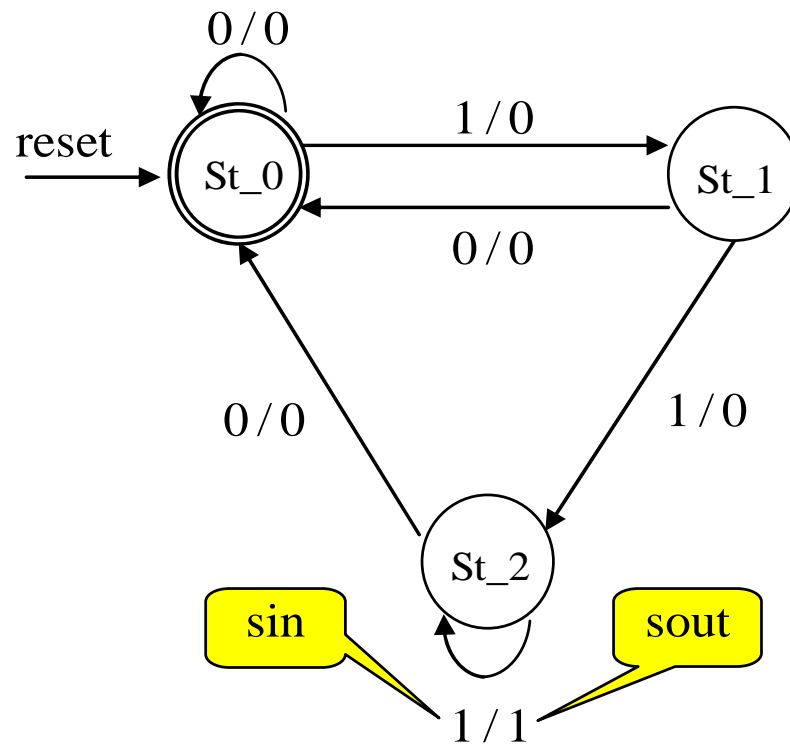


**Arithmetic State Machine, ASM**  
算法状态机

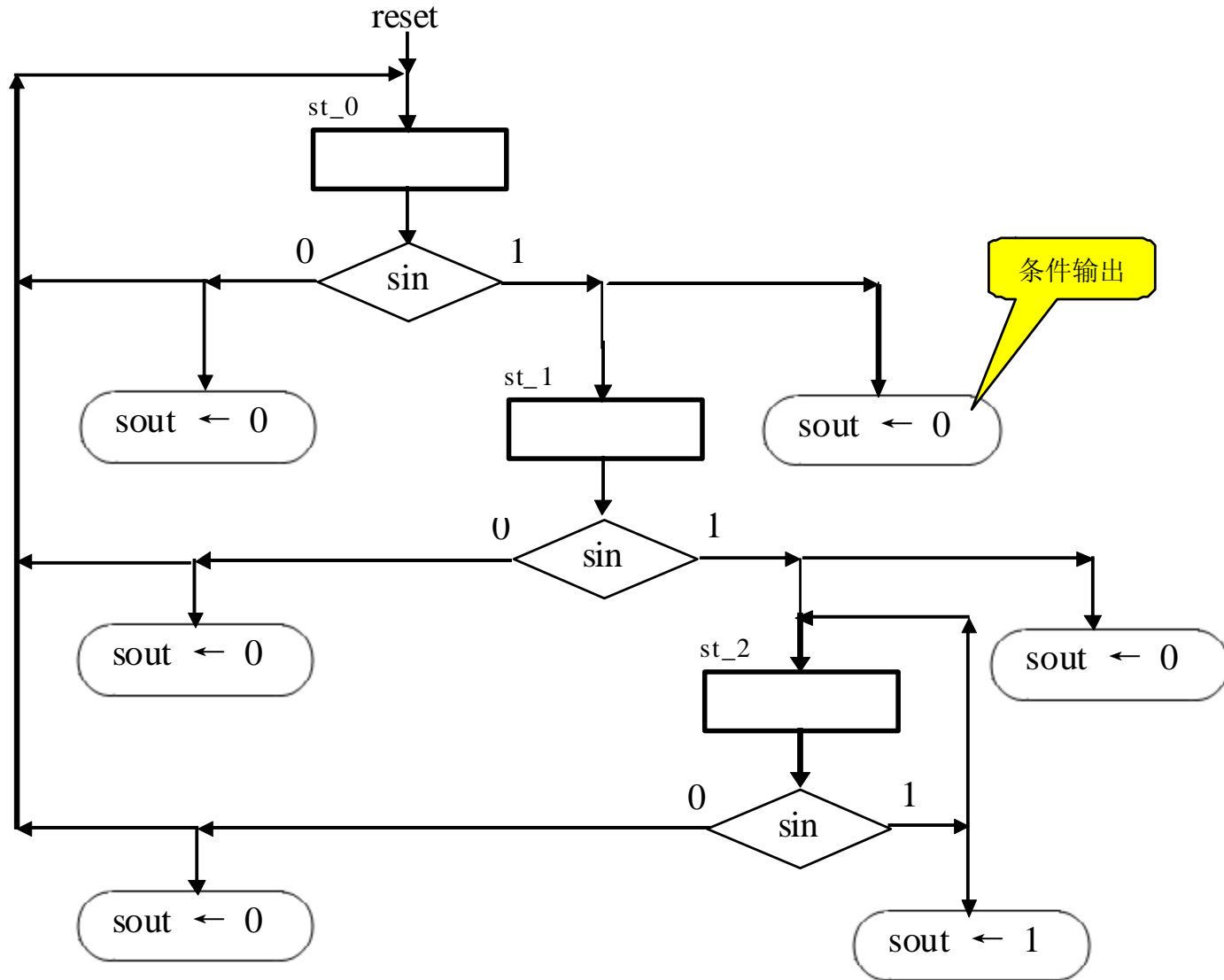
# 检测器的Mealy型状态图



- ◆ 输出取值不仅和当前状态有关，还和当前输入有关。



# 与Mealy型状态图等价的ASM图



# 小结



- ◆ 状态图（或ASM图） → 状态表 → （状态化简）  
→ （状态分配） → VHDL代码 → Quartus2
- ◆ 状态化简：
  - 完全规定的有限状态机
  - 不完全规定的有限状态机
- ◆ 同步时序电路中的竞争和险象
- ◆ 当电路规模较大时，手工设计工作量大。

# 第7章 异步时序电路



- ◆ 异步时序电路的特点
  - ◆ \*脉冲异步时序电路
  - ◆ \*电位异步时序电路
  - ◆ \*电位异步时序电路综合中防范险象的措施
- 
- ◆ 有理论意义，难度较大，**选修**。
  - ◆ EDA工具目前还不支持**异步时序电路的综合**。

# 异步时序电路的特点



## ◆ 记忆元件：

- **脉冲异步时序电路**：记忆元件是触发器，但不要求所有触发器共享同一时钟；
- **电位异步时序电路**：记忆元件通常是带反馈的门电路。

## ◆ 约束条件：

各输入信号不允许同时发生变化，并且输入信号第 $i$ 次变化引起电路的变化达到稳定后，才允许输入信号发生第 $(i+1)$ 次变化。

## ◆ 基本的记忆元件（包括**D锁存器**、**D触发器**、**JK触发器**等）都应当属于最简单的电位异步时序电路。

- 都可以用**电位异步时序电路的理论**设计出来。

# 设计实例：D触发器



## ◆ 例7.4 上升沿触发的D触发器

- 功能表是设计要求（原始出发点）：

D 触发器功能表

输 入			次 态 $Q^{n+1}$	功 能 解 释
reset	clk	$D^n$		
0	X	X	0	reset 低电平有效，实现异步清零
1	nr	X	$Q^n$	clk 未发生正跳变时，D 触发器保持原状态不变
	↑	0	0	clk 上升沿将 D 端数据打入 D 触发器
		1	1	

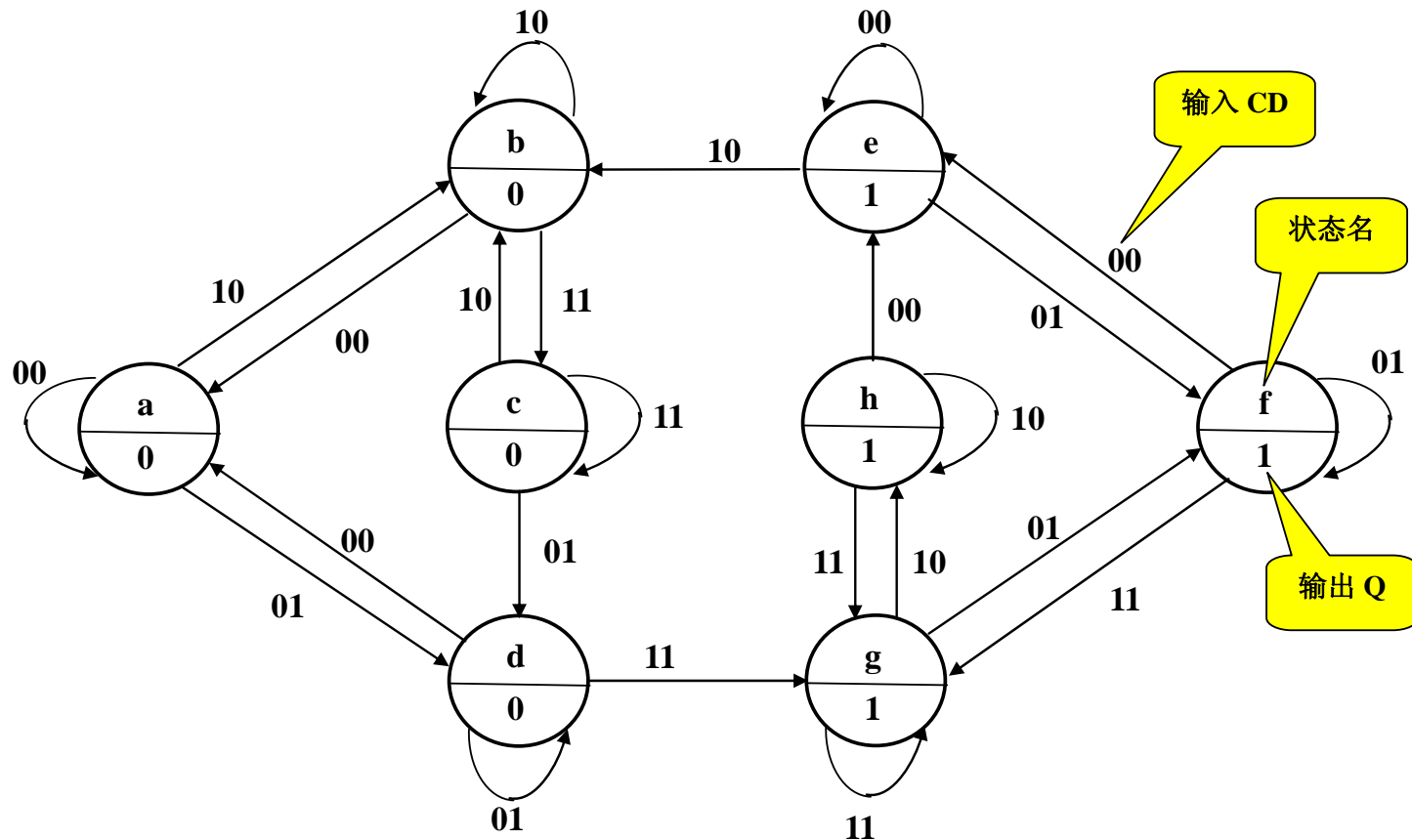
说明： clk = ↑ 表示 clk 正跳变； clk = nr 表示 clk 未发生正跳变； X 表示取值任意；  
 $D^n$  代表触发器 D 端的当前值。

# D触发器设计 (续)



## ◆ 功能表 → 原始状态图

- 正确、无遗漏;
- 状态化简可以留给后续步骤。



# D触发器设计（续）



## ◆ 原始状态图 → 原始流程表:

次态 Y 现态 y	输入 CD				输出 Q
	00	01	11	10	
a	a	d	∅	b	0
b	a	∅	c	b	0
c	∅	d	c	b	0
d	a	d	g	∅	0
e	e	f	∅	b	1
f	e	f	g	∅	1
g	∅	f	g	h	1
h	e	∅	g	h	1

说明：现态与次态相同时，该状态为稳定状态，外加圆圈表示；  
∅表示次态未指定（即可由设计者任意指定）。

# D触发器设计（续）



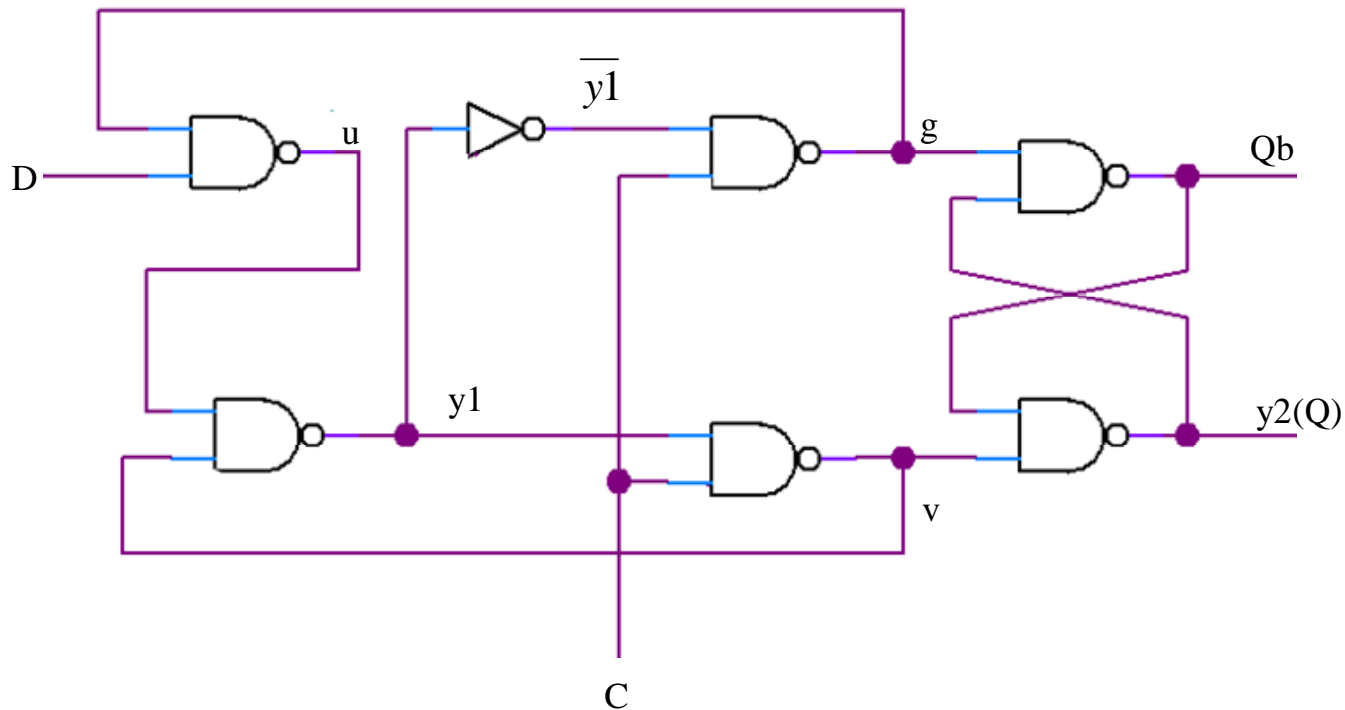
◆ 原始流程表 → 状态化简 → 化简后的流程表：

次态 Y 现态 y	输入 CD				输出 Q
	00	01	11	10	
A (a, b, c)	(A)	D	(A)	(A)	0
D (d)	A	(D)	F	∅	0
F (f, g, h)	E	(F)	(F)	(F)	1
E (e)	(E)	F	∅	A	1

# D触发器设计（续）



- ◆ 状态化简后的流程表 → 状态编码 → 次态函数的逻辑表达式 → 化简 → 原理图：

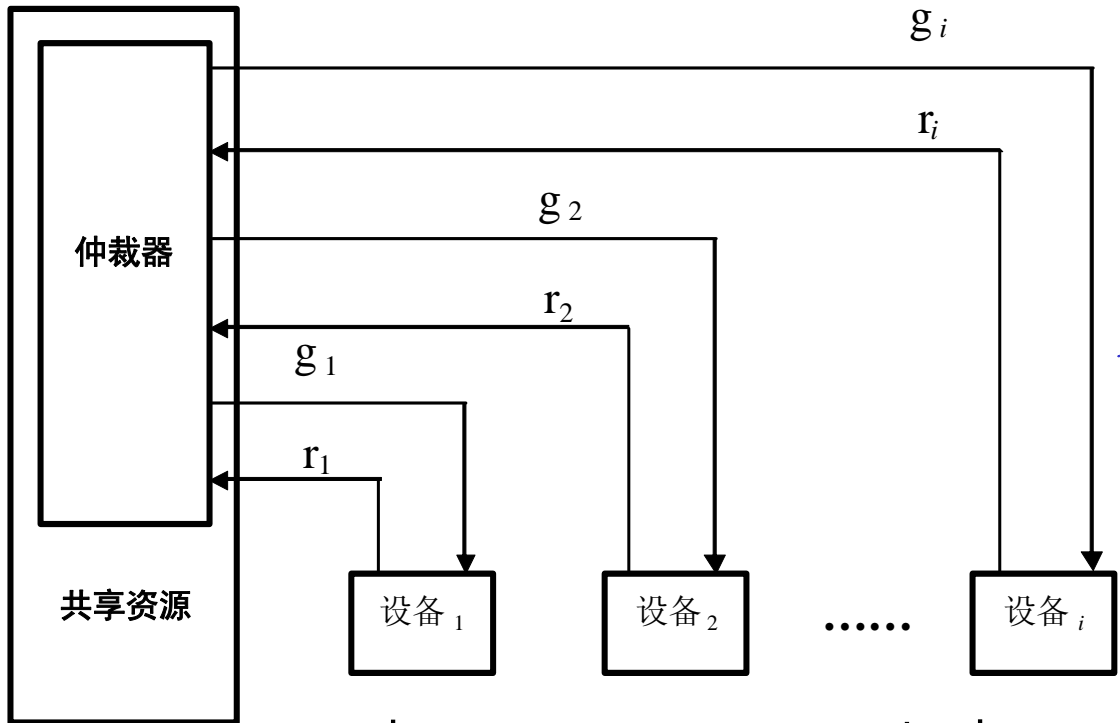


# 电位异步时序电路的其它实例

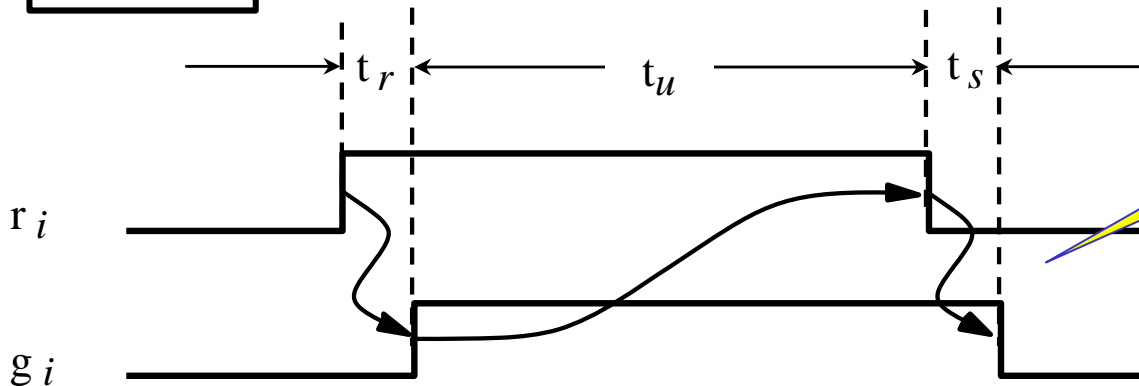


- ◆ D锁存器的分析;
- ◆ 仲裁器的设计:
  - 用异步时序电路的方法设计;
  - 用同步时序电路的方法设计;
- ◆ 3分频器的设计:
  - 用异步时序电路的方法设计;
  - 用同步时序电路的方法设计;

# 设计实例：异步特性的同步电路实现

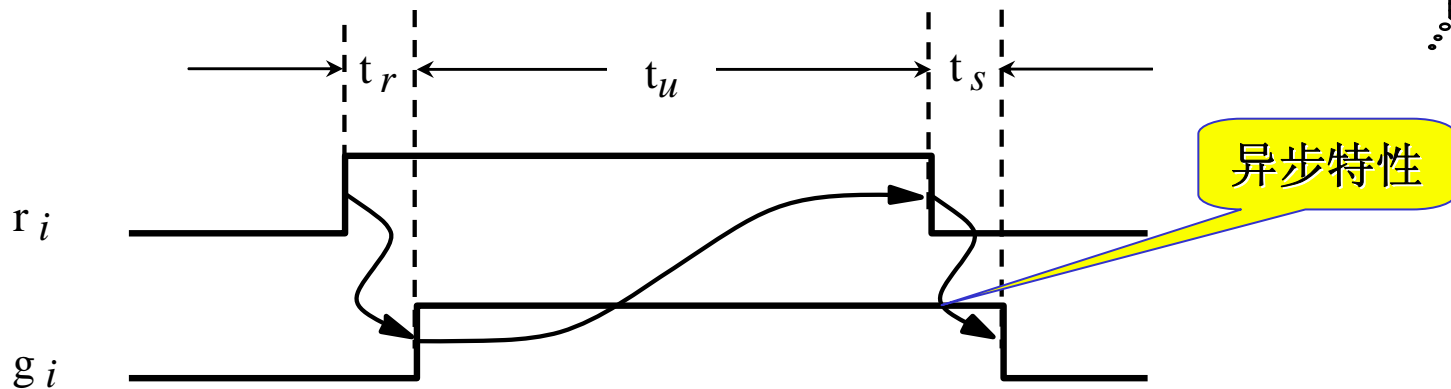


仲裁器  
示意图



握手信号的  
异步特性

# 异步特性的同步电路实现



- ◆ 仲裁器用高频的时钟信号检测输入信号 $r_i$ 的电平
  - 可能有少许时间延迟
  - 时钟周期非常短的情况下，这种延迟是否可以容许？
- ◆ 后果：
  - 优点：设计工作量减少，EDA工具支持；
  - 缺点：响应速度低，成本略高。

# 第8章 数字系统设计



## ◆ 数字系统的特点和设计方法：

- 通常指规模较大、能完成一个完整任务的时序电路。
- 一般用同步时序电路实现。

## ◆ 设计实例：

- 交通灯控制器设计；
- 求最大值电路。

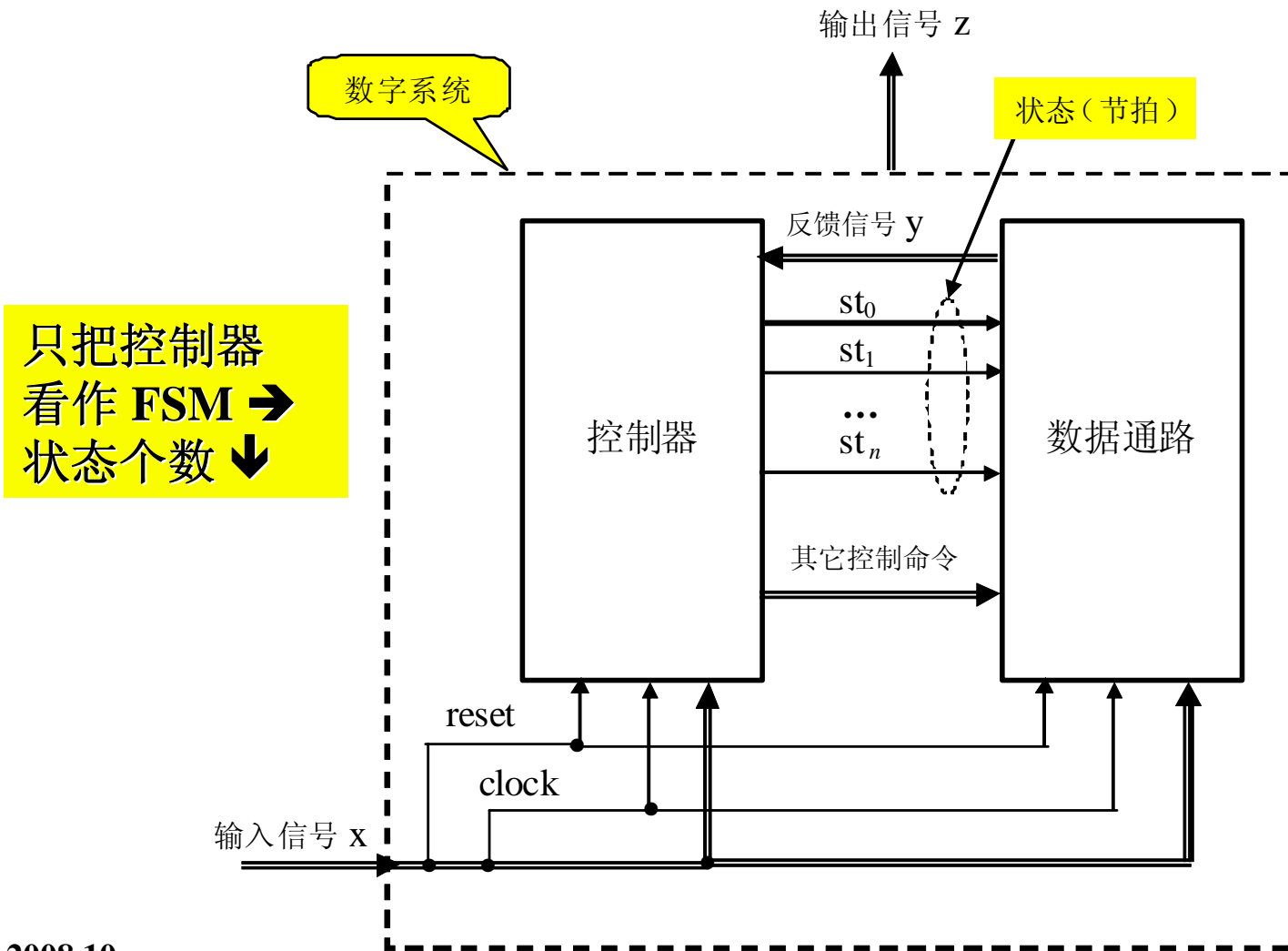
## ◆ 数字系统中某些技术细节：

- 减少时钟偏移的布线网络；
- 消除机械开关抖动的电路。

# 数字系统的分解



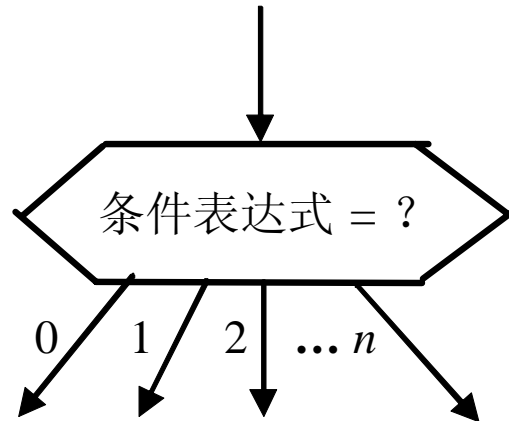
## ◆ 分解的目的：降低FSM中的状态个数



# ASM图的变更和扩充

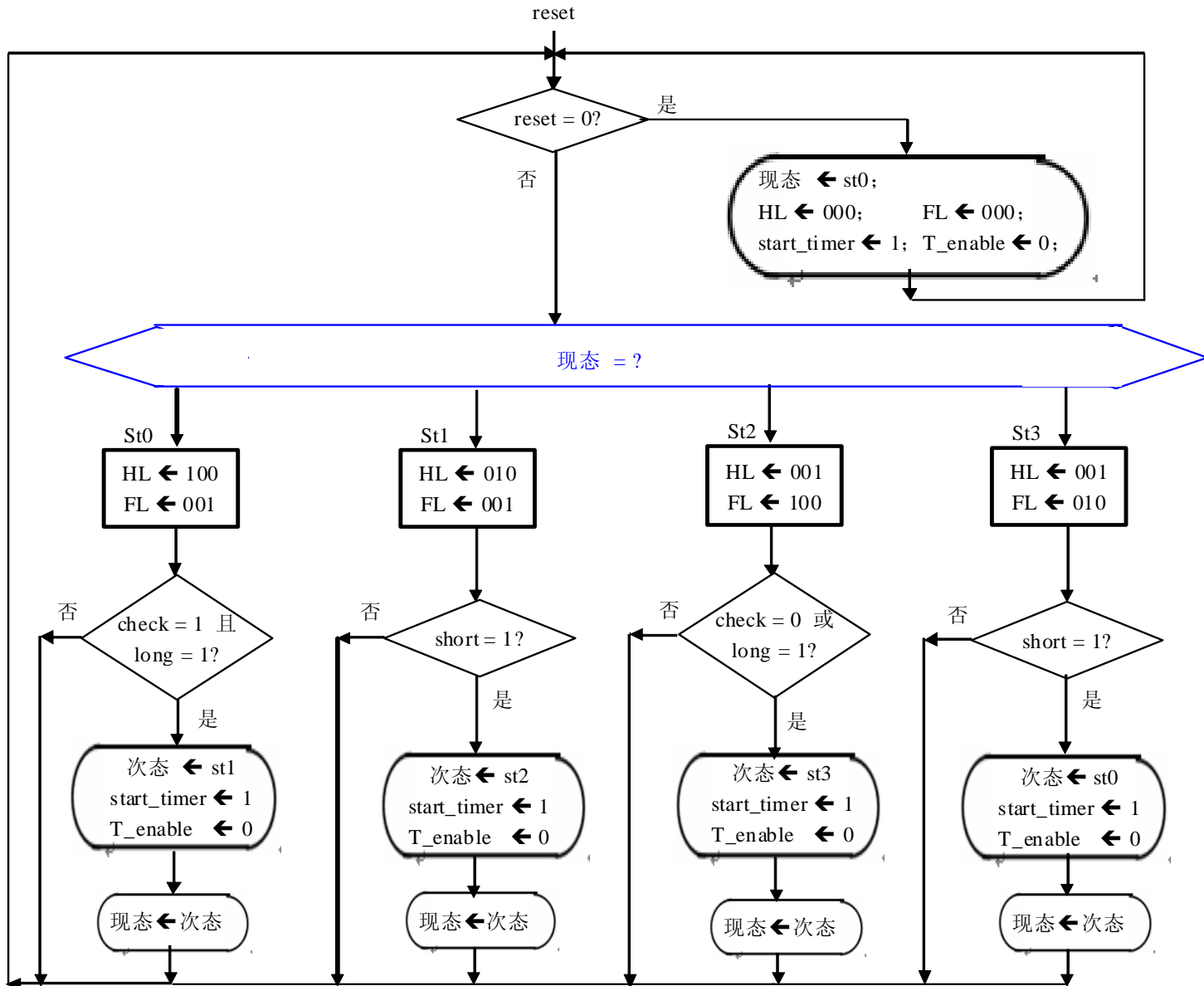


- ◆ 判别框增加一个多分枝框：



- ◆ 把操作和给输出信号赋值不加区分，一律看作操作。集中精力关注（在当前状态和当前输入条件下）：
  - 执行什么操作；
  - 即将转入哪一个次态？
- ◆ 对应的状态表中允许使用条件语句。

# ASM图举例



# 状态表举例

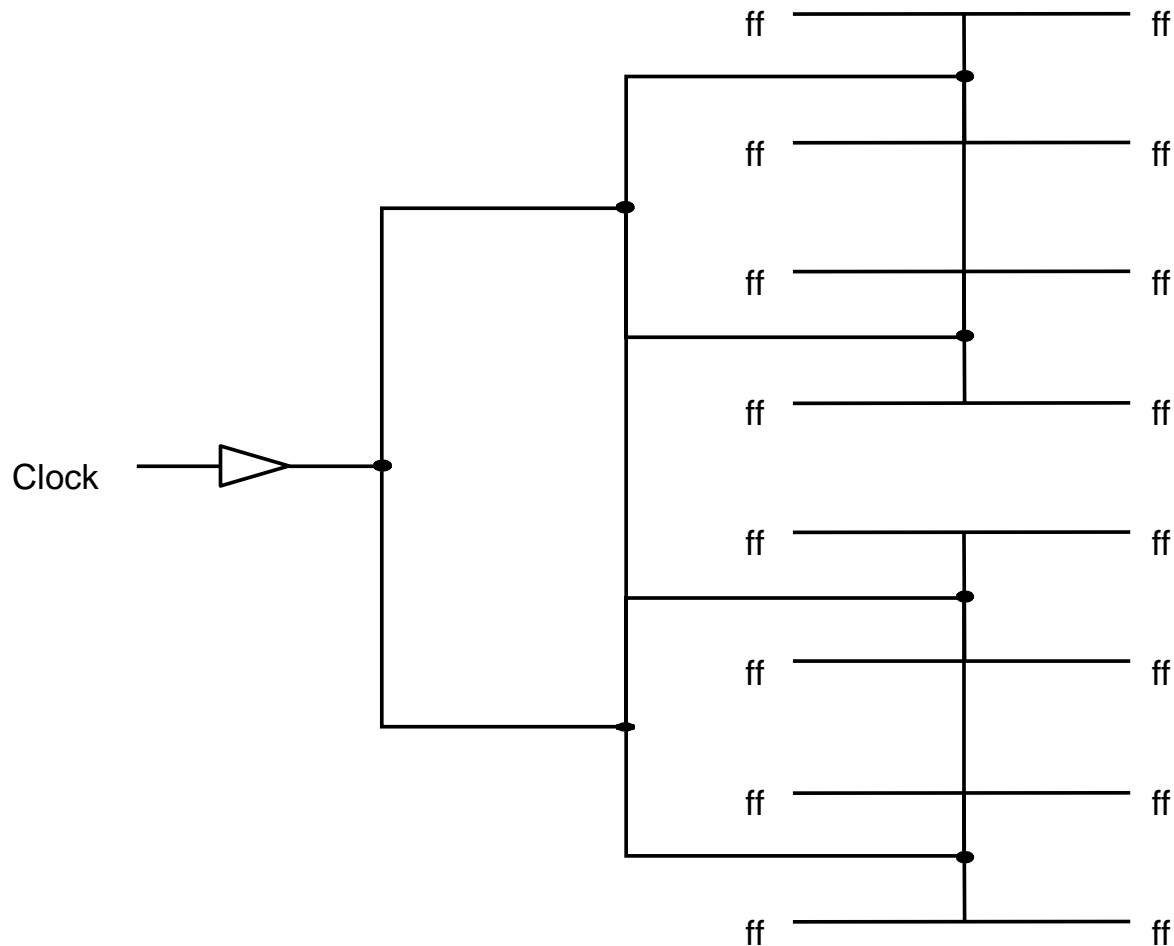


reset	当前状态	执行操作	状态变迁
0	X	HL ← 000; FL ← 000; start_timer ← '1'; T_enable ← '0';	current_state ← st0;
1	st0	HL ← "100"; FL ← "001"; IF (check = '1') AND (long = '1') THEN start_timer ← '1'; T_enable ← '0'; END IF;	IF (check = '1') AND (long = '1') THEN next_state ← st1; END IF;
	st1	HL ← "010"; FL ← "001"; IF short = '1' THEN start_timer ← '1'; T_enable ← '0'; END IF;	IF short = '1' THEN next_state ← st2; END IF;

# 时钟偏移 (clock skew)



## ◆ 减少时钟偏移的措施: global clock





---

# 局部时钟和全局时钟的比较

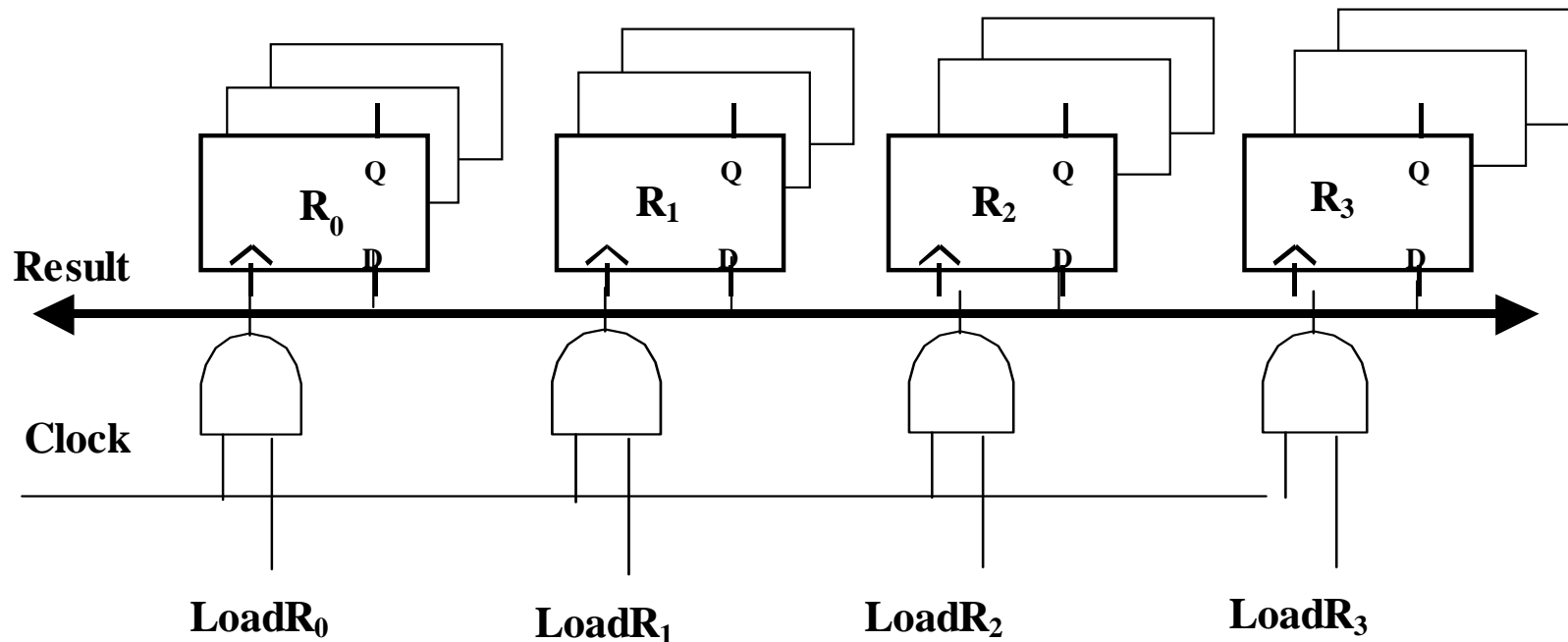
全局时钟：减少时钟畸变的影响

# 如何将运算结果加载于指定的 $R_i$



方案之一：控制加载的时钟脉冲（局部时钟）。

触发器为普通的D\_FlipFlop。 优缺点？



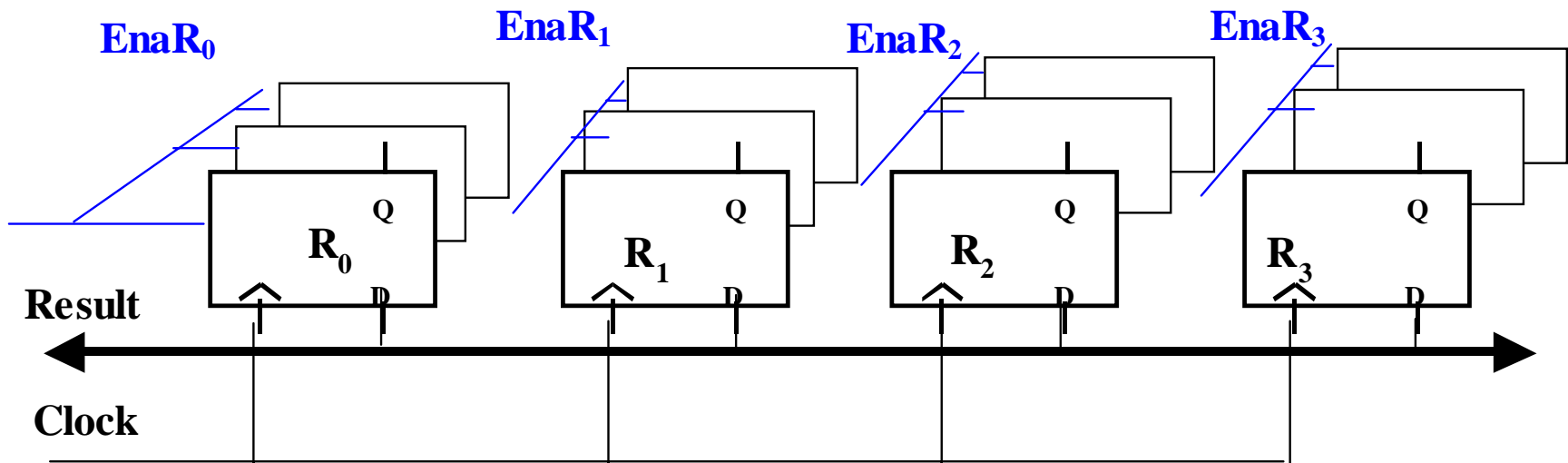
# 如何将运算结果加载于指定的 $R_i$



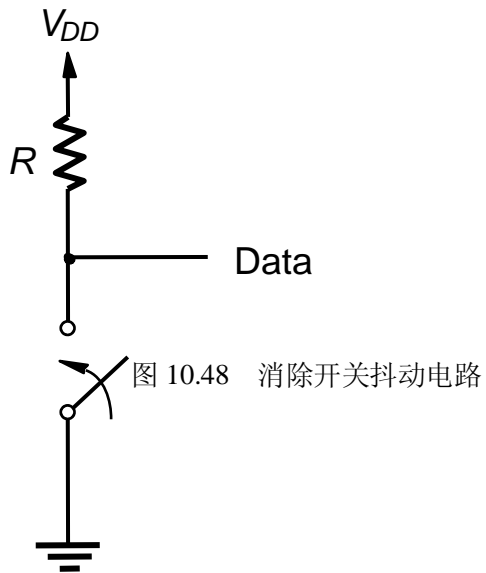
方案之二：使用全局脉冲，触发器带使能控制Enable。



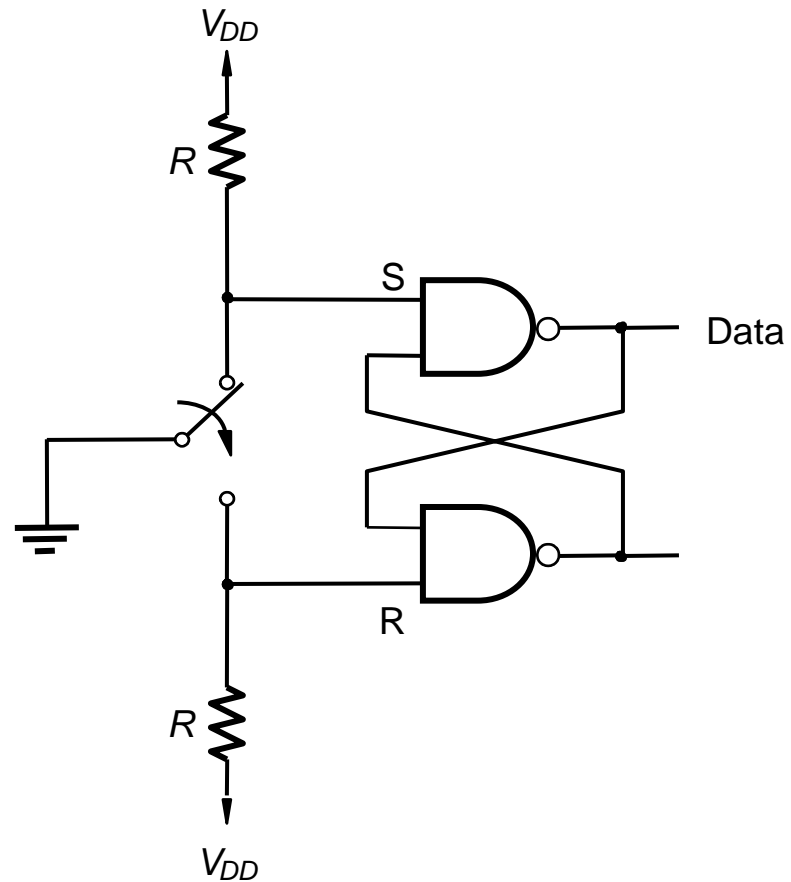
优缺点？（Altera的PLD器件提供全局时钟引脚。）



# 消除开关抖动的电路



(a) 单极单掷开关



(b) 带基本SR锁存器的单极双掷开关

# 附录A EDA工具Quartus II 简介

---



- ◆ 是简介，不是全面的使用说明书；
- ◆ 实例引导方式：
  - 跟着实例一步、一步走，可学会基本内容。
- ◆ 共查阅用；
- ◆ 在清华大学出版社网站有.ppt文件；

# 附录B VHDL简介

---



- ◆ 提供VHDL的基本内容；
- ◆ 供查阅用；
- ◆ 在清华出版社网站有.ppt文件；

# 主要目标



- ◆ 掌握数字逻辑的基本理论;
- ◆ 独立设计一个简单的数字系统（例如 第8章中的习题）。
  - 理解题意;
  - 状态图 / ASM图;
  - 状态表;
  - VHDL代码;
  - 提交Quartus II ;
    - 模拟验证;
    - 硬件实验;



---

欢迎交流  
批评指正  
谢谢！