数字系统逻辑设计
总复习

佟冬
tongdong@pku.edu.cn

微处理器研究开发中心（MPRC）
计算机科学技术系

北京大学

期末考试和大作业Lab检查

- 期末考试：
  - 日期：2020年6月18日
  - 时间： 14:00-16:00
  - 地点：Classin
  - 要求：自备A4纸若干、考完拍照上传教学网。每页都要写上姓名和学号。

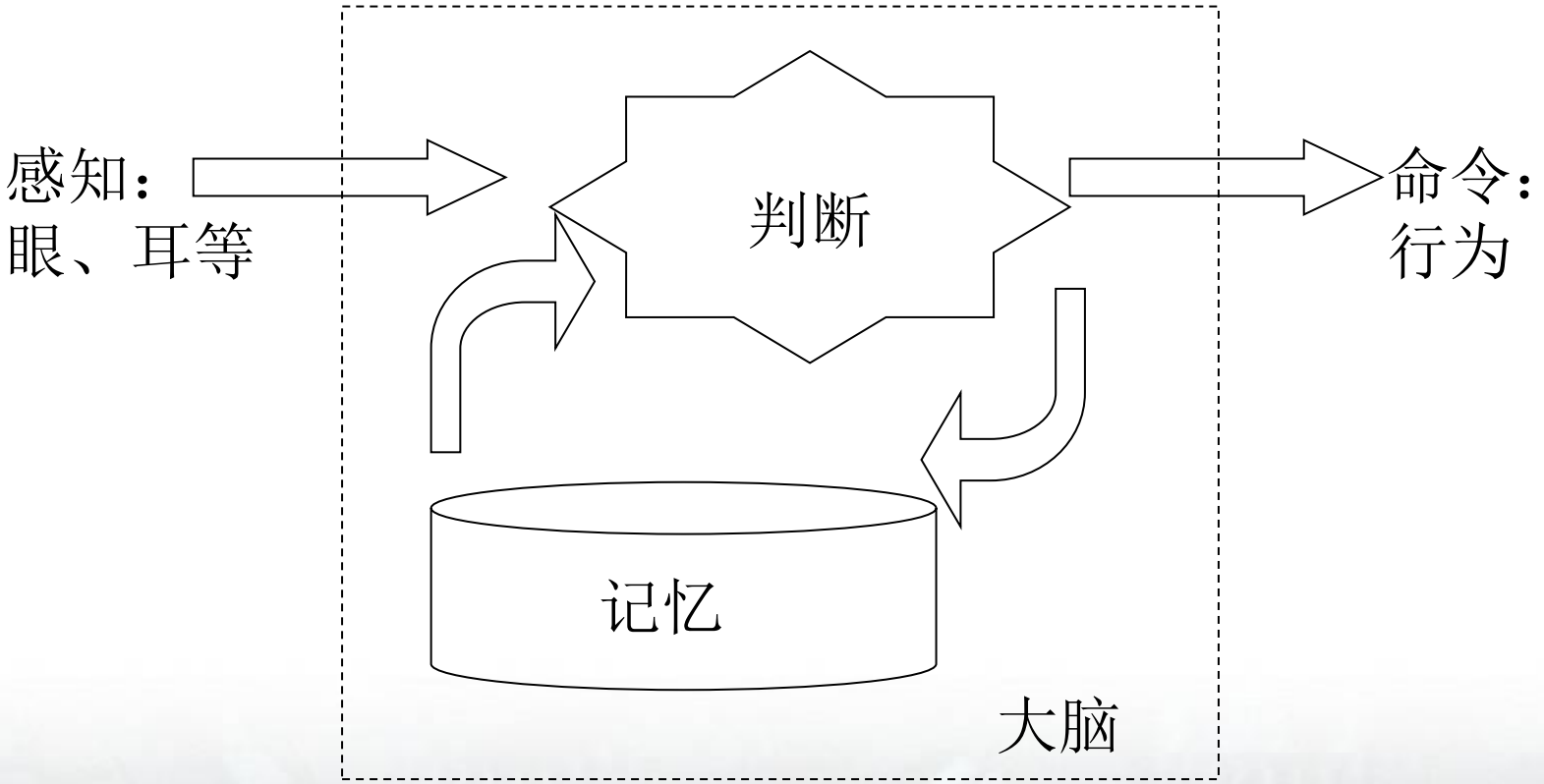
- 大作业检查
  - 网上提交

复习方法

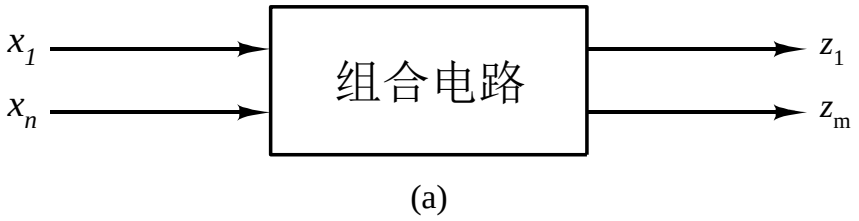
- 搭建整体框架，逐步细化的复习方法
- 数字逻辑课程的主要内容
  - 目的
  - 基础知识和原理
  - 方法和过程
  - 器件
  - 工具

人是如何思考(Thinking)的?

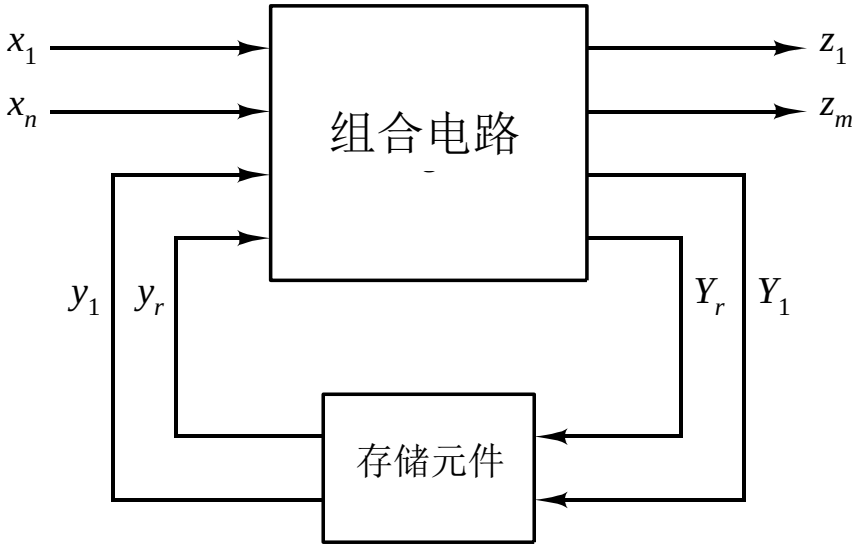
- 简单的模型?



逻辑电路：组合电路和时序电路

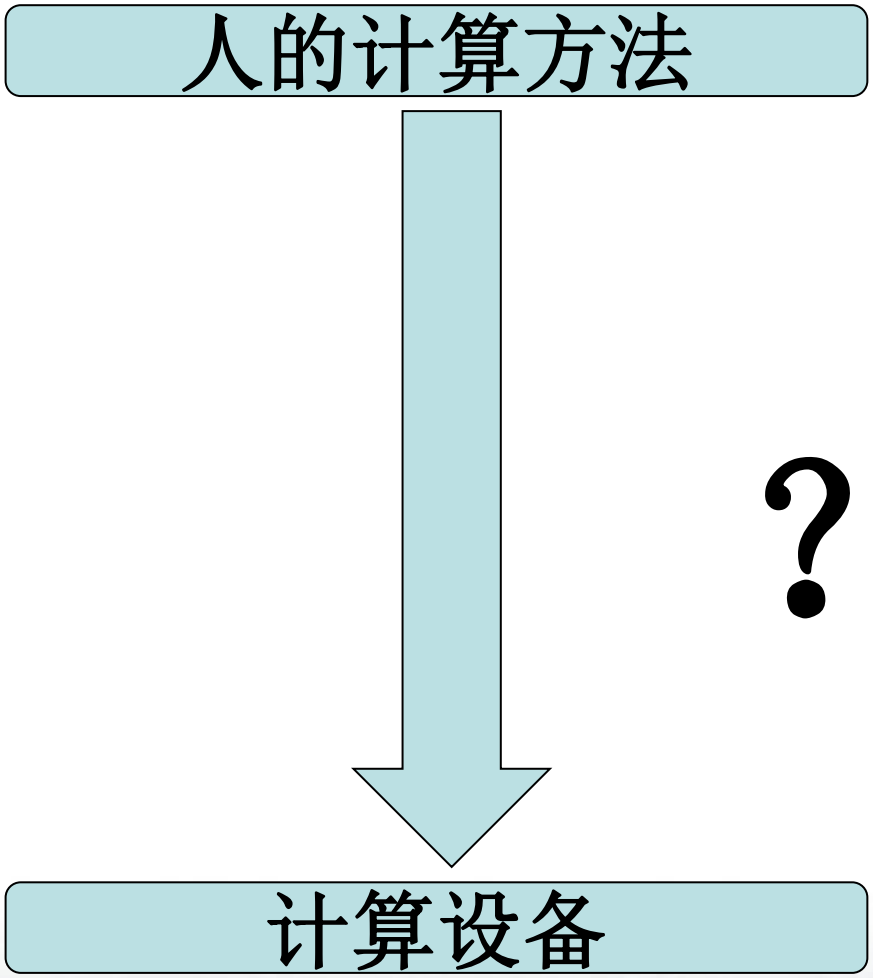


(a)



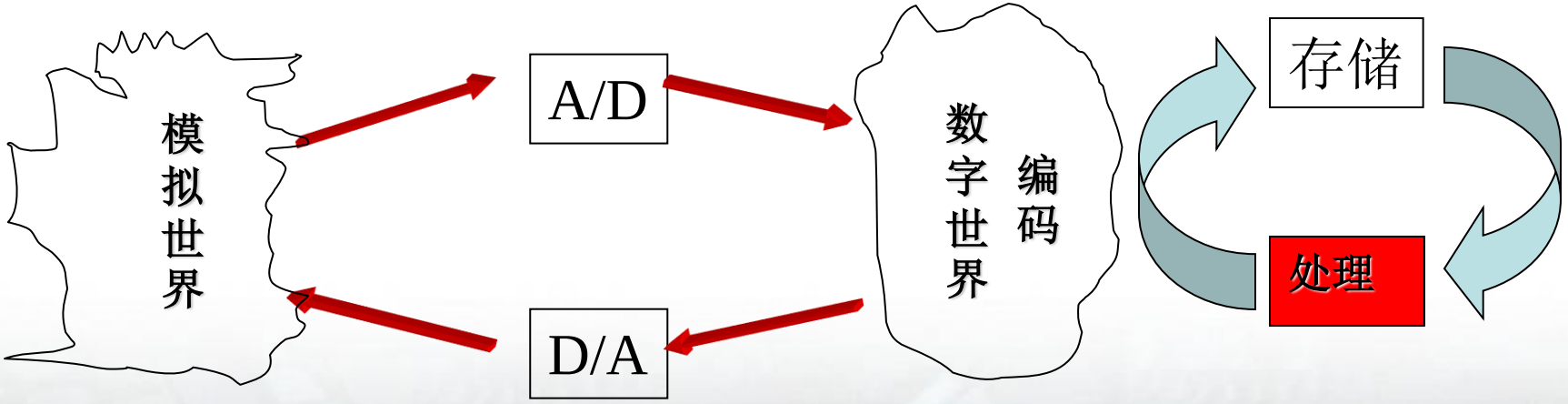
(b)

如何做出一个能计算的设备?



0 数字系统

- 数字系统
  - 对数字处理和存储的系统
- 设计
  - 给定一个问题的表述，从可用部件集合中从优选择，得到一种解决问题的办法。
- 数字系统原理——数制和编码



1 数制和编码

- 数字系统中的数字的基本理论和概念
  - 数制
    - 十进制、二进制、八进制、十六进制
    - 运算
    - 数制之间的转换
  - 有符号数
    - 原码、补码、反码
  - 计算机编码
    - 数字编码：定点和浮点
    - 字符编码和其它编码：ASCII、汉字
    - 检错码和纠错码

2 数字逻辑的理论基础

- 布尔代数
  - 6个公设
  - 定理
  - 数字逻辑课程的理论基础

- 课程要求
  - 熟练掌握
  - 定理的证明

真值表

- 将一个开关函数 *f* 对于其变量每种可能取值的结果用表的形式表示。
- 三个函数：与(AND)、或(OR)、非(NOT)的真值表

a b	$f(a, b) = a+b$
0 0	0
0 1	1
1 0	1
1 1	1

AND

a b	$f(a, b) = ab$
0 0	0
0 1	0
1 0	0
1 1	1

OR

a	$f(a) = \overline{a}$
0	1
1	0

NOT

开关函数

- 积之和SOP
- 和之积POS
- 环和Ring-Sum

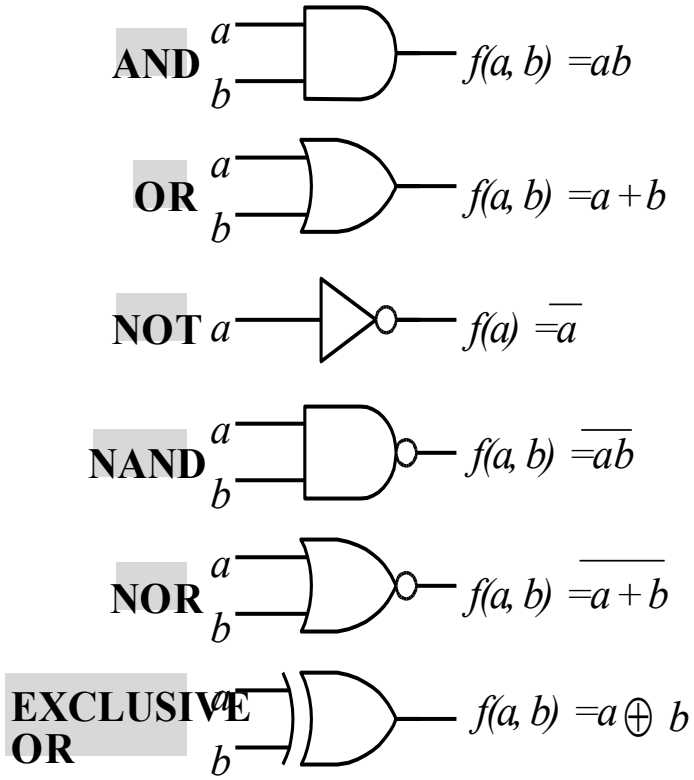
- 最小项范式，最小项列表
- 最大项范式，最大项列表

- 非确定性函数

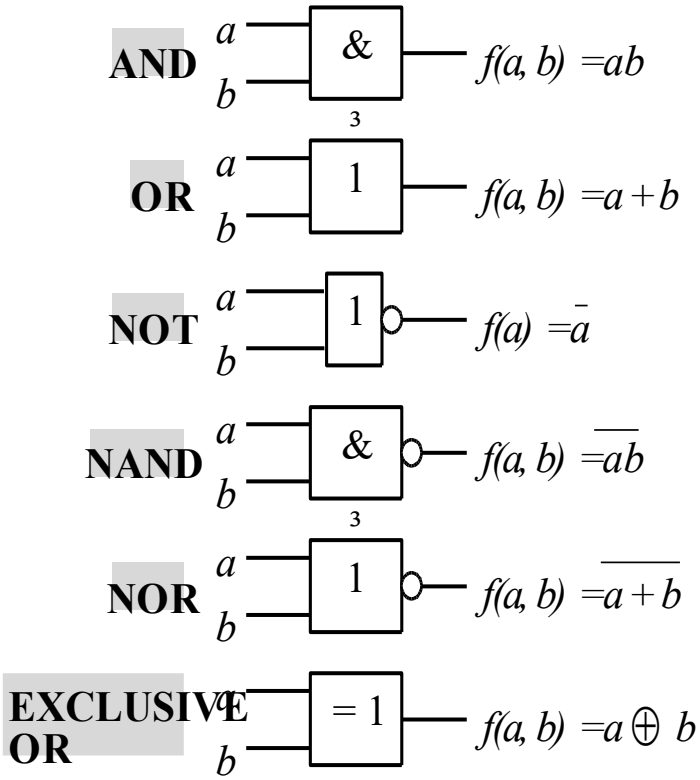
非确定函数

- 开关函数非完全确定的原因
  - 一些确定的输入组合不会产生
  - 一些输出为0或1只对特定的输入组合成立
- 无关(don't care)最小项：忽略一些最小项
- 无关最大项：忽略一些最大项
- 表示
  - 无关最小项：d_i
  - 无关最大项：D_i
- 例子：
  - 无关最小项：$f(A,B,C)=\Sigma m(0,3,7)+d(4,5)$
  - 无关最大项：$f(A,B,C)=\Pi M(1,2,6)\cdot D(4,5)$

开关电路—逻辑门



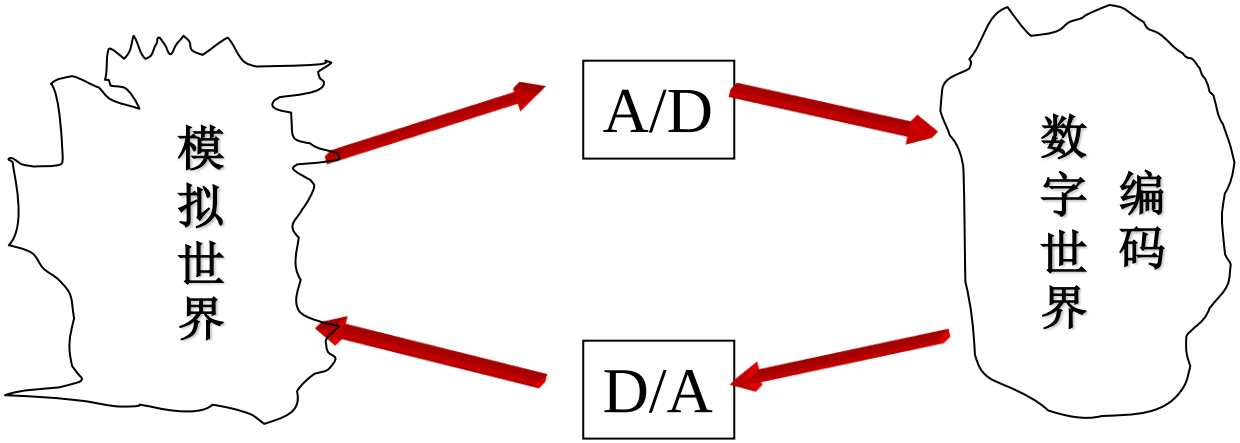
Symbol set 1

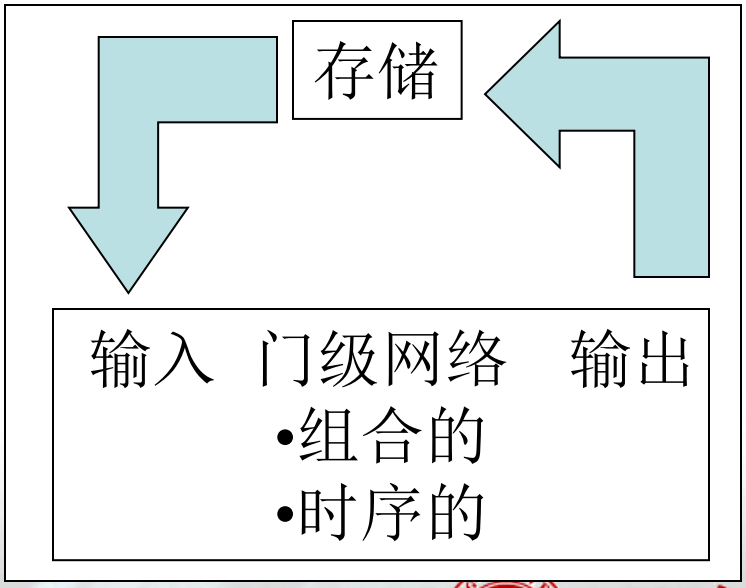


Symbol set 2
(ANSI/IEEE Standard 91-1984)

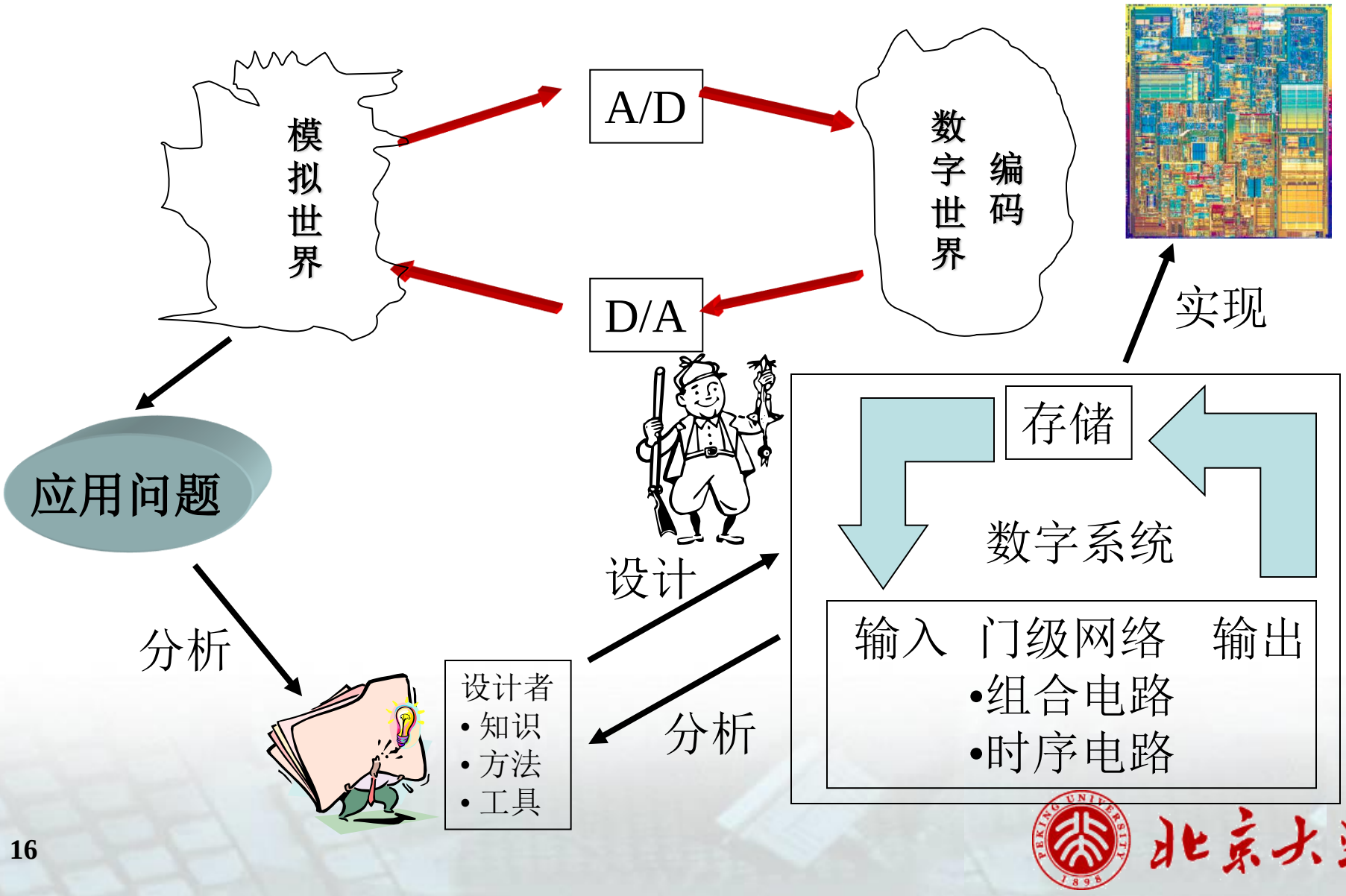
门级网络

- 门级网络（Gate Network）是相互连接的门、网络输入端口和网络输出端口的有限集合。门级网络应该满足以下的约束：
  - 一个门的输出端口或一个网络的输入端口不能连接另一个门的输出端口或网络另一个输入端口。
  - 每一个门的输入端口或者网络的输出端口连接一个常值或者一个门的输出端口或者网络的输入端口。
  - 门的输出和输出不能相连！





3 组合逻辑分析与设计



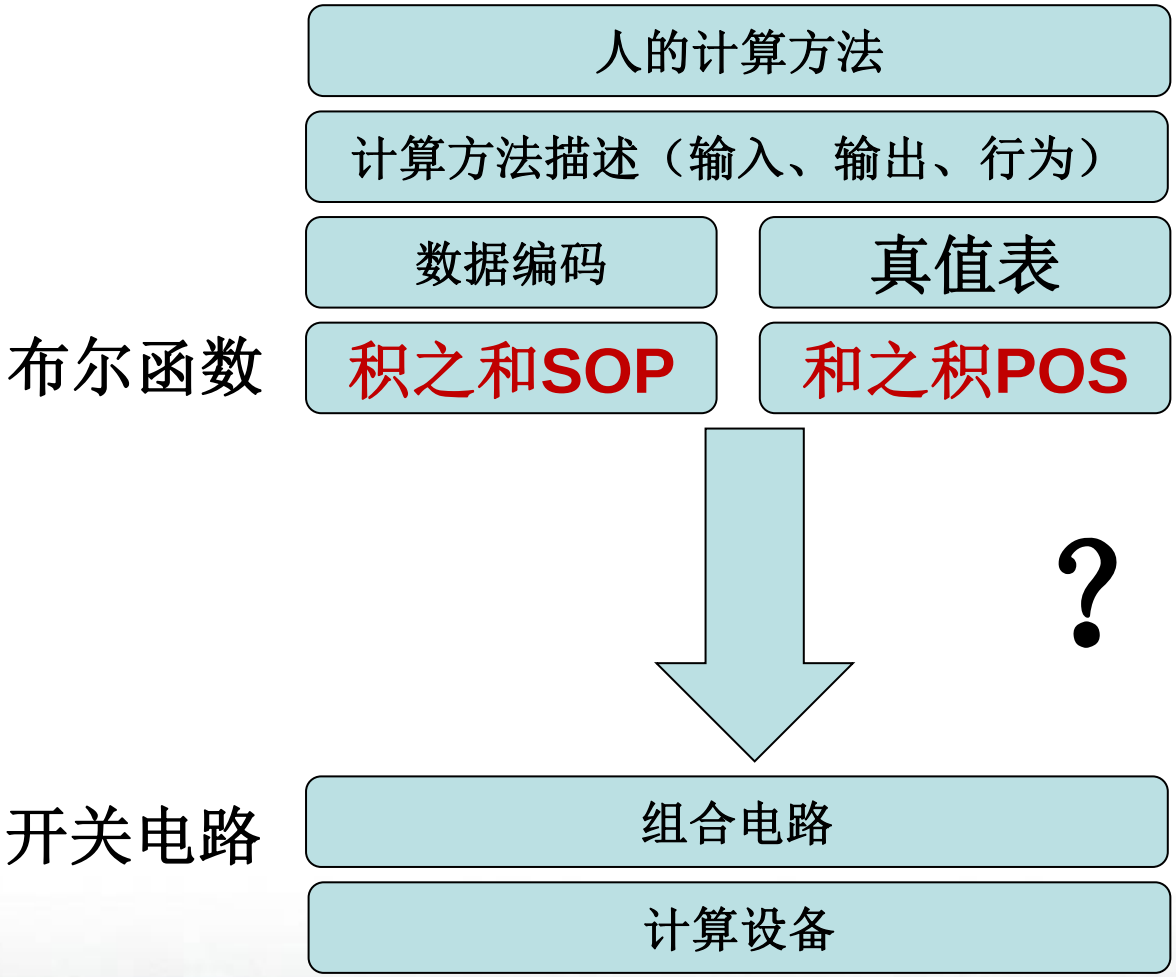
电路分析与设计

- 电路设计：从一个功能的文字描述到一组开关函数进而到门级、PLD或其它逻辑单元实现的转化过程。
- 电路分析：从一个数字电路的实现出发，得到电路的某种形式的描述
  - 开关表达式(Switch expression)
  - 真值表(Truth table)
  - 时序图(Timing diagram)
  - 其它行为描述(behavioral description)
- 设计与分析是相反的过程

组合逻辑电路的综合

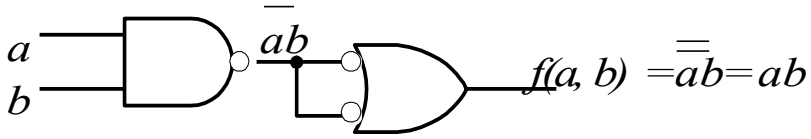
- 实现NAND(NOR)逻辑的流程
  - 第一步：将函数表示为最小项列表（最大项列表）的形式；
  - 第二步：写出最小项（最大项）；
  - 第三步：利用开关代数简化函数为积之和（和之积）的形式；
  - 第四步：利用定理8a(8b)和定理3将表达式转化为NAND(NOR)的形式。
  - 第五步：画出NAND(NOR)逻辑图
- 与或非门AND-OR-INV

如何做一个能计算的设备?

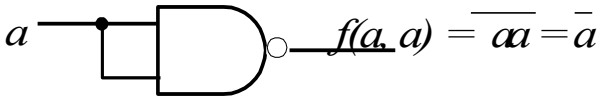


与非门的性质

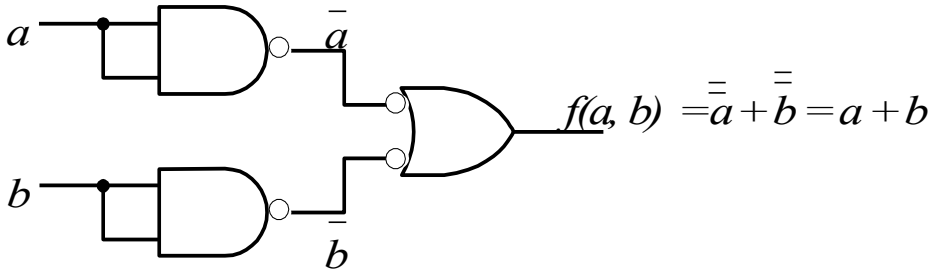
- AND, OR, NOT门可以用NAND门来构造



AND gate



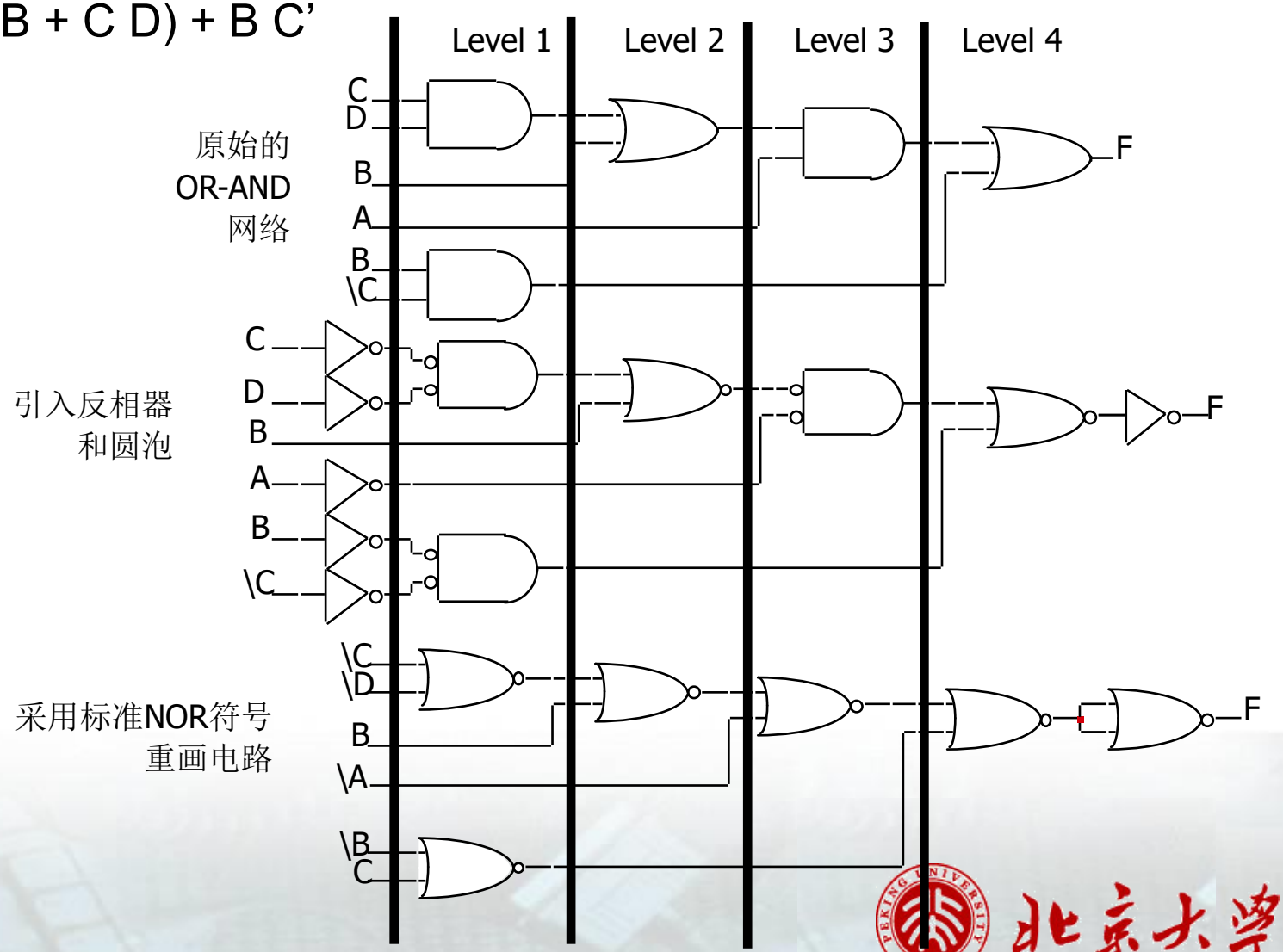
NOT gate



OR gate

多级逻辑的NOR变换

- F = A (B + C D) + B C’

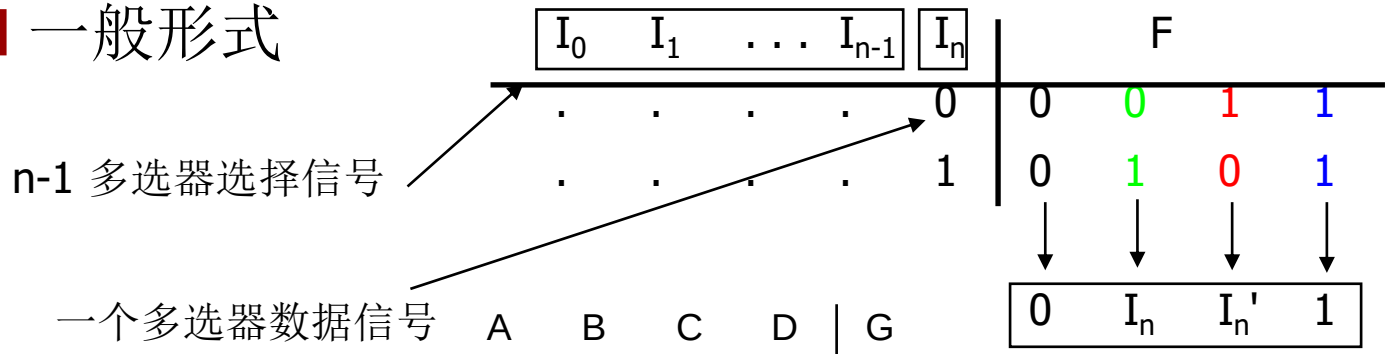


5 标准模块化组合电路

- 自顶向下和自底向上设计
- 模块化设计
- 标准组合电路
  - 译码/编码器（各种应用）
  - 多路选择器（各种应用）
  - 加法器/减法器
  - 比较器
  - ALU设计

多选器实现逻辑电路

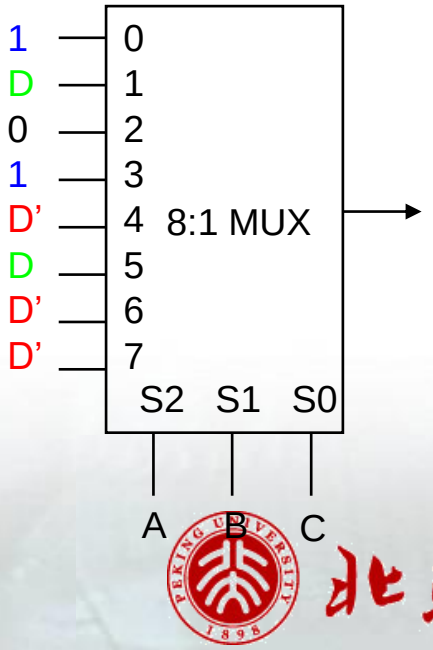
- 一般形式



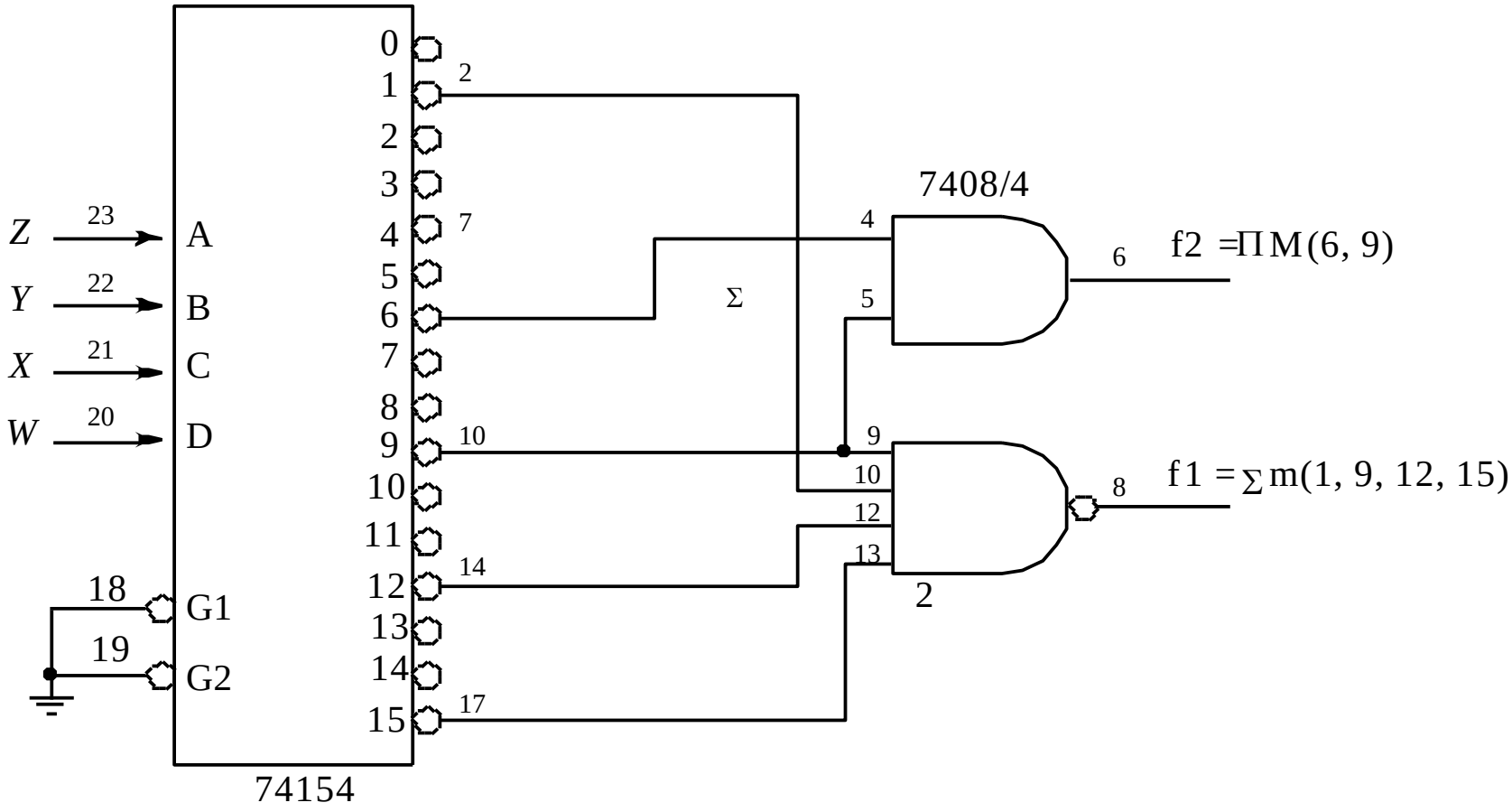
- 例子

A	B	C	D	G	
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	D
0	0	1	1	1	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	1	D'
1	0	0	1	0	
1	0	1	0	0	D
1	0	1	1	1	
1	1	0	0	1	D'
1	1	0	1	0	
1	1	1	0	1	D'
1	1	1	1	0	

选择A,B,C作为选择信号



最小项范式的实现



4 组合逻辑化简

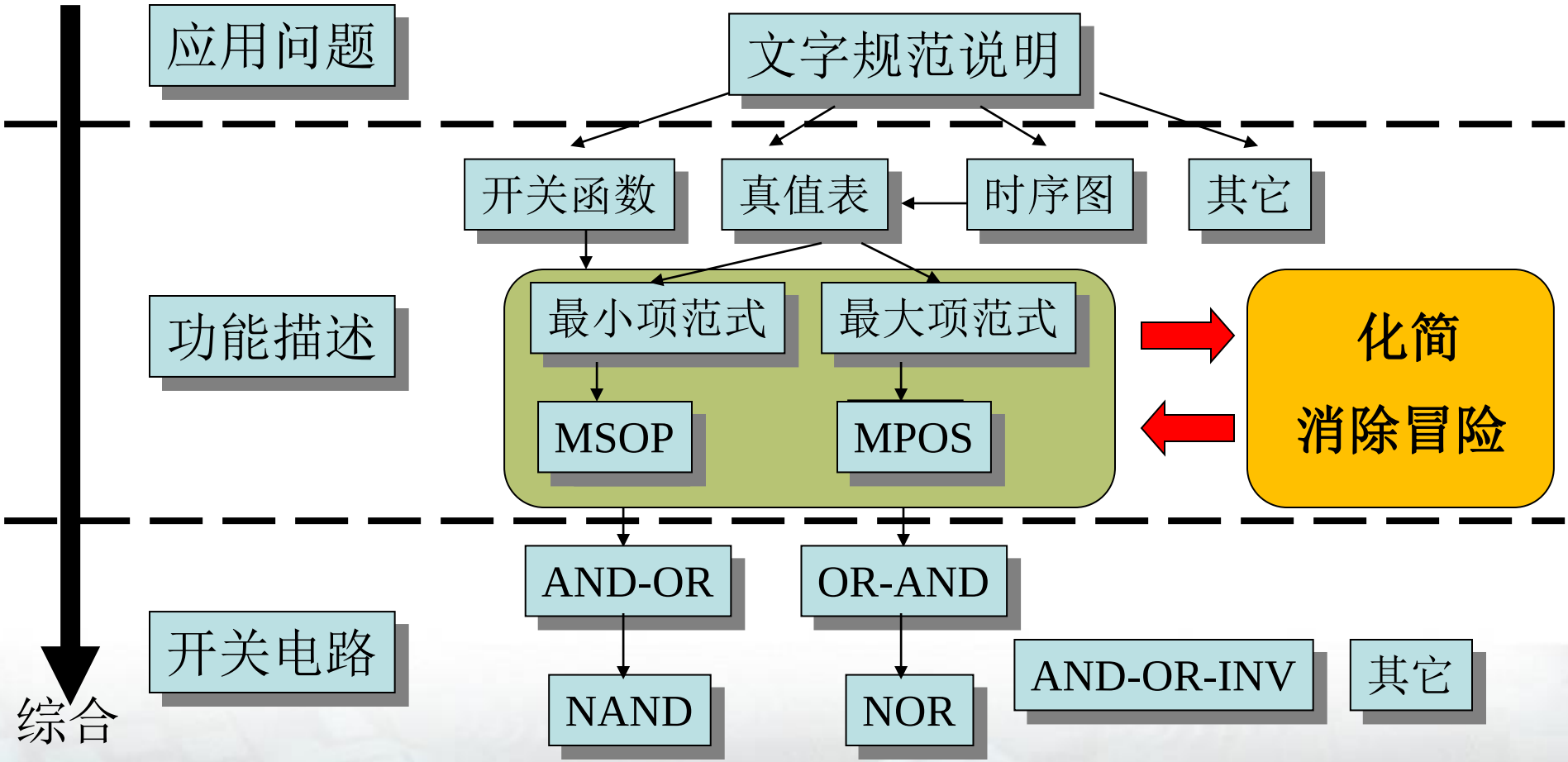
- 目的：减少实现指定开关函数的成本(cost)
- 成本的度量和其它考虑
  - 门的数量
  - 电路级的数量(时延)
  - 门的扇入和扇出
  - 互连结构的复杂性
  - 避免冒险(hazards)
- 两级实现(Two-level realizations)
  - 门数最少（开关函数中项的个数）
  - 扇入最小（开关函数的项中字母个数）

化简方法

- 常用技术
  - 布尔代数的公设和定理
  - 卡诺图(Karnaugh maps, K-map)
  - Quine-McCluskey方法
  - Petrick方法
  - 通用算法
- 特性
  - 启发式Heuristics (次优化)
  - 算法Algorithms (优化)

函数最小化和电路化简

- 组合电路的分析与综合



卡诺图(Karnaugh Map)

- 卡诺图(K-maps)是表示开关函数的最常用的表示形式（平面卡诺图最大可表示6个变量的开关函数）。
- 卡诺图是寻找MSOP和MPOS的最基本的启发式方法形式。
- n个变量的卡诺图有2^n个单元，每个单元对应真值表中的一行。
- 卡诺图的每一个单元标以相应真值表的行号。
- 卡诺图的单元采用以下的排列方式：相邻单元的真值表输入只有一位不同（距离为1，逻辑相邻）
- 开关函数映射：将每一个单元对应的函数值(0, 1, d)映射到该单元内。
- 卡诺图是真值表的二维形式。

卡诺图化简指导原则

- n变量卡诺图中的每个单元都有n个逻辑相邻的单元。
- 单元可以被合并为大小为 $2,4,8,\ldots,2^k$ 的组。
- 每个被合并组中包含的所有单元都对一些变量有相同的值。
- 合并尽可能多的单元，这将导致组所对应的项内字母的个数最少（扇入）。
- 尽可能用最少的组覆盖所有的最小项，这将导致结果中包含最少的积项（门数和最后一级或门的扇入）。
- 应该从最“孤立”单元开始。

4.5 质蕴涵项和覆盖(SOP)

- 蕴涵项(*implicant*)是一个积项，它能覆盖一个函数的一些最小项。
- 质蕴涵项 (*prime implicant*) 是一个不被该函数的任何其它蕴涵项覆盖的积项。
- 实质蕴涵项(*essential prime implicant*)是一个至少包含一个最小项的质蕴涵项，它不能被任何其它任何质蕴涵项覆盖。
- 一组蕴涵项被称为函数的一个覆盖(Cover)，如果函数的每个最小项至少被该组内的蕴涵项覆盖。
- 最小覆盖是一个包含最少的质蕴涵项和最少的符号的覆盖。

Quine-McCluskey最小化方法

- Q-M方法相对于卡诺图的优点
  - 可计算的
  - 可以处理多于六个变量的函数
- 算法概况
  - 给出函数的所有最小项
  - 找出所有的质蕴涵项（第一步和第二步）
    - 第一步：将最小项按重量（1的个数）分组。
    - 第二步：穷尽地找出所有的质蕴涵项。
  - 找出最小的质蕴涵项覆盖（第二步和第四步）
    - 构造质蕴涵项图
    - 选择最小数目的质蕴涵项覆盖

生成质蕴涵项

列表1			列表2			列表3		
最小项	A B C D		最小项	A B C D		最小项	A B C D	
2	0 0 1 0	√	2, 6	0 – 1 0	PI2	8, 9, 12, 13	1 – 0 –	PI1
4	0 1 0 0	√	2, 10	– 0 1 0	PI3			
8	1 0 0 0	√	4, 6	0 1 – 0	PI4			
6	0 1 1 0	√	4, 12	– 1 0 0	PI5			
9	1 0 0 1	√	8, 9	1 0 0 –	√			
10	1 0 1 0	√	8, 10	1 0 – 0	PI6			
12	1 1 0 0	√	8, 12	1 – 0 0	√			
13	1 1 0 1	√	9, 13	1 – 0 1	√			
15	1 1 1 1	√	12, 13	1 1 0 –	√			
			13, 15	1 1 – 1	PI7			

质蕴涵项图表

	2	4	6	? 8	? 9	10	? 12	? 13	? 15
* * PI_1				?	?		?	?	
PI_2	?		?						
PI_3	?					?			
PI_4		?	?						
PI_5		?					?		
PI_6				?		?			
* * PI_7								?	?

简化的图表

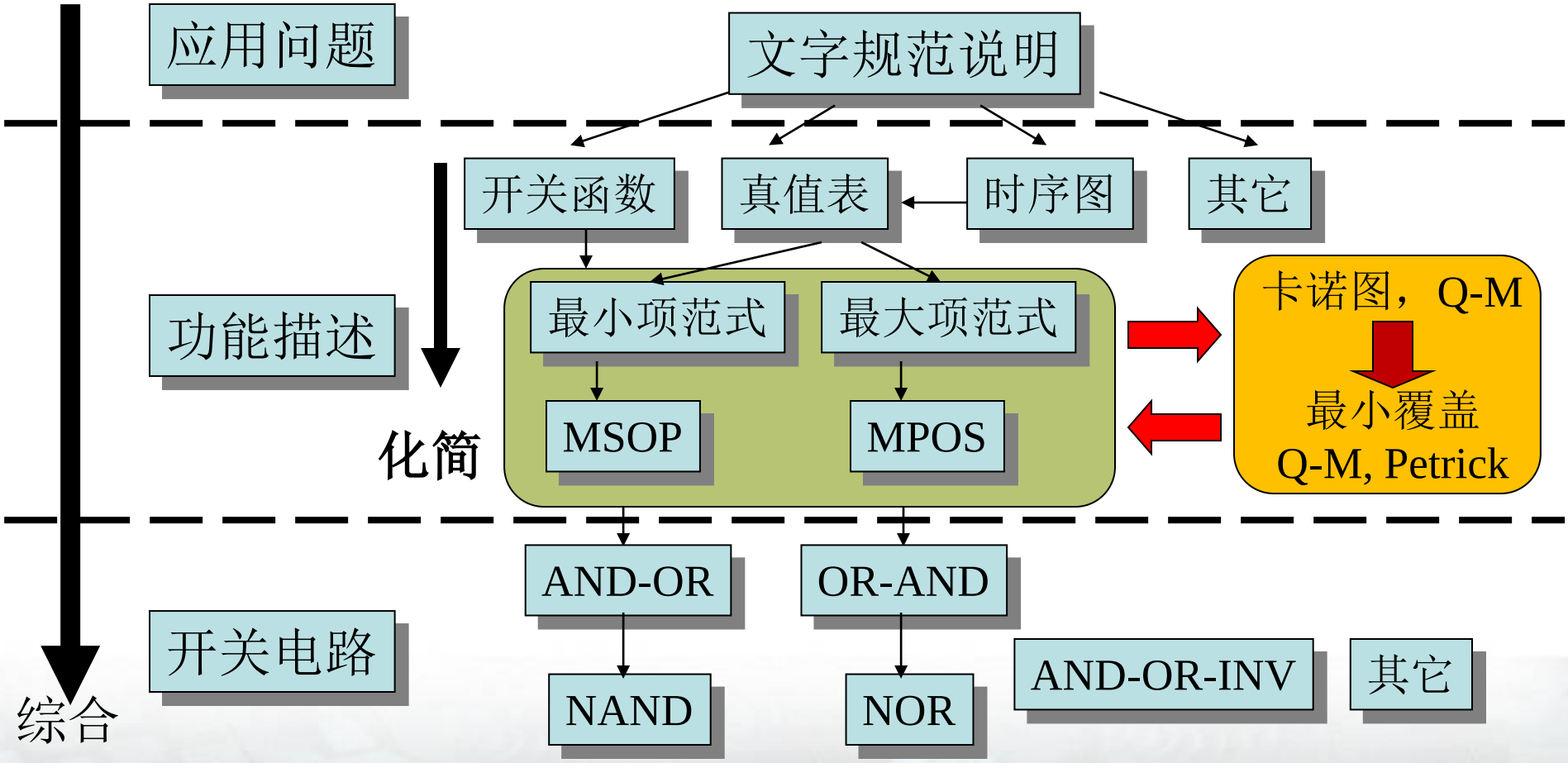
	5	10	11	13
PI_2	?			?
PI_3	?			?
PI_4		?	?	
PI_5			?	?
PI_6		?	?	

	?	?
	5	10
* PI_2	?	
*PI_4		?

覆盖过程(Covering Procedure)

- 第一步：标示出所有只被某个PI覆盖的最小项，将这些PI加入覆盖中。
- 第二步：在质蕴涵项图中去掉所有在第一步中表示出的PI覆盖的行，同时去掉被这些PI覆盖的最小项的列。
  - 规则1：如果一行被其它行覆盖，去掉该行
  - 规则2：如果一列覆盖其它列，去掉该列
- 第三步：如果第二步的结果是一个循环图表，进入第五步。否则再次执行第一部和第二步。
- 第四步：如果第三步的结果是一个循环图表，进入第五步。否则进入第一步。
- 第五步：重复执行第五步，任意选择一个质蕴涵项。直到产生一个空图表或者非循环表。如果表中还有符号，进入第一步。

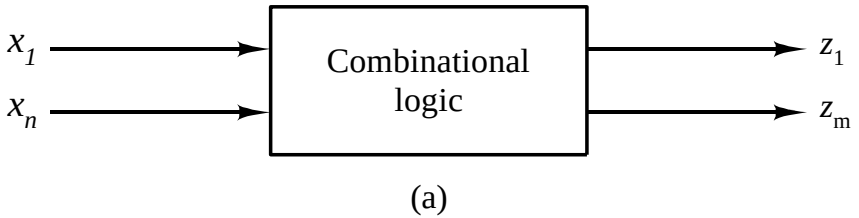
组合电路的化简总结



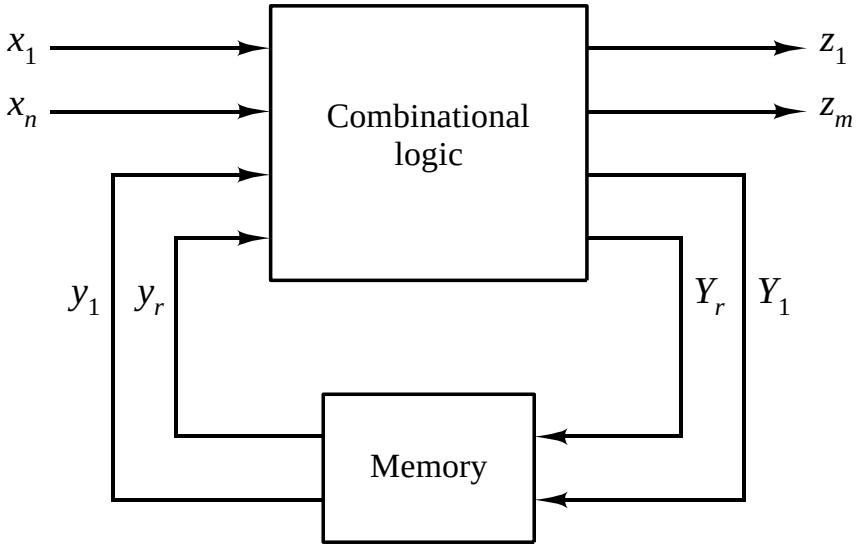
6 同步时序电路

- 时序电路基本概念
- 状态
  - 现态
  - 次态
- 输入信号
- 输出信号
  - Moore机
  - Mealy机
- 状态变换

时序电路的表示—图表示



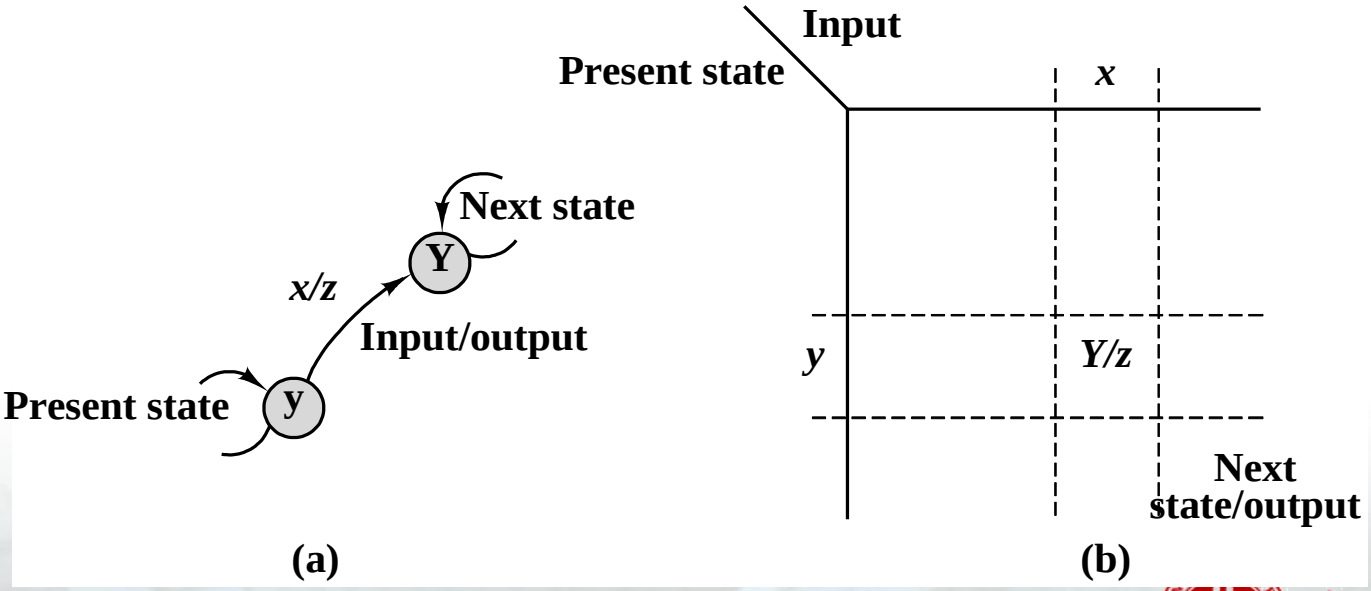
(a)



(b)

状态表和状态图

- 状态图
  - 圆：状态
  - 线：状态变换
  - 线上标注：产生状态变换的输入和相应输出



存储元件

- 存储元件(Memory device)
  - 双稳态(bistable)电子线路
  - 状态 0
  - 状态 1
- 二进制的存储
  - 状态0，表示存储逻辑“0”。
  - 状态1，表示存储逻辑“1”。
  - 输出Q，指示存储元件的现态

存储元件

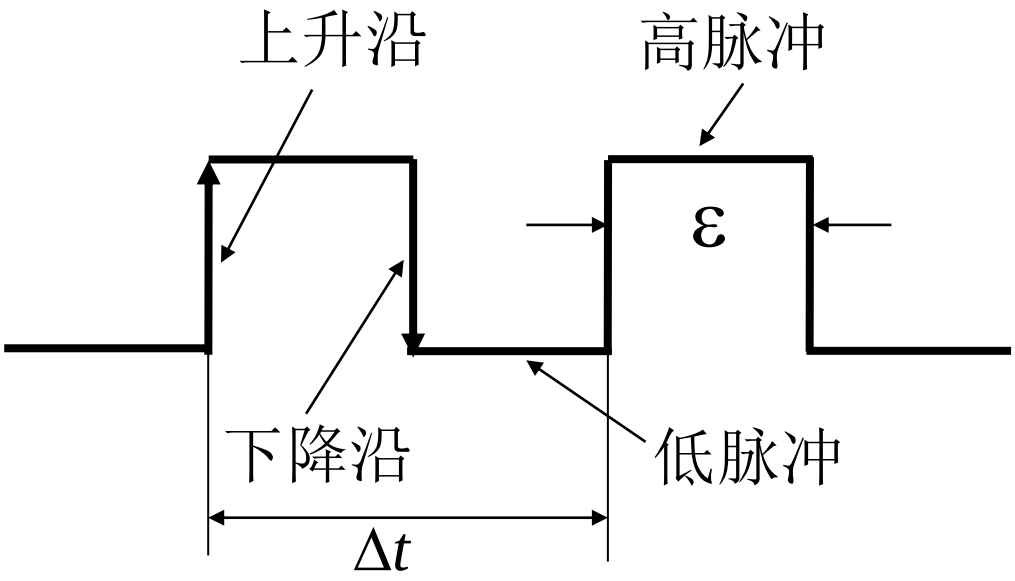
- 激励输入(excitation inputs)
  - 每个存储元件有多个输入，能激励或者驱动存储元件进入确定的状态的输入，被称为激励输入。
  - 一般的存储元件的命名是根据它与其它存储元件不同的激励输入。
- 存储元件的类型
  - 锁存器(*latch*)
  - 触发器(*flip-flop*)

存储元件(3)

- 锁存器
  - 锁存器的激励输入控制元件的状态。
  - 置位锁存器(set latch)，激励输入强制元件的输出为1。
  - 复位锁存器(reset latch)，激励输入强制元件的输出为0。
  - 置位复位锁存器(set-reset latch)，同时具有置位和复位激励信号的元件。

时钟

- 时钟(clock)



- 边沿触发(edge-triggered)触发器
- 脉冲触发(pulse-triggered)触发器

存储元件(4)

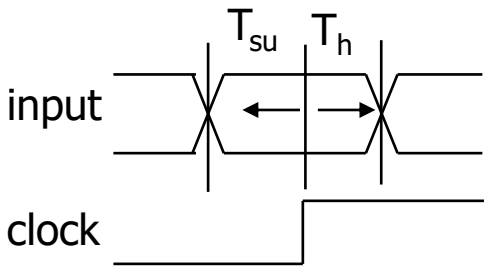
- 触发器
  - 时钟控制信号(*clock*)
  - 时钟信号向触发器发命令，触发器根据激励信号改变状态。
  - 在多触发器的电路中，时钟信号可以使所有的触发器同步(synchronized)的改变状态。
  - 时钟树的概念

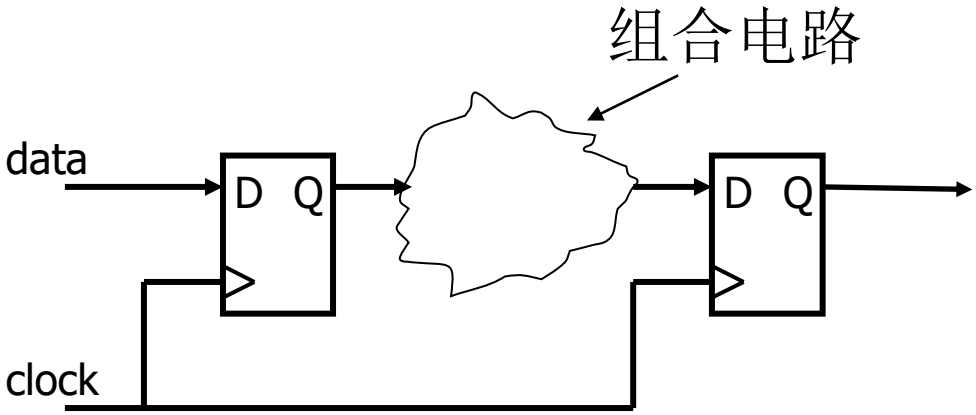
锁存器和触发器的总结

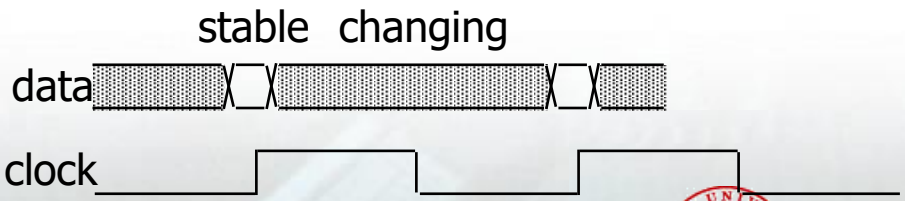
电路设备	特征方程
SR锁存器	$Q^* = S + \overline{R}Q$
门控SR锁存器	$Q^* = SC + \overline{Q}R + \overline{C}Q$
D锁存器	$Q^* = DC + \overline{C}Q$
SR触发器	$Q^* = S + \overline{R}Q$
D触发器	$Q^* = D$
JK触发器	$Q^* = \overline{K}Q + J\overline{Q}$
T触发器(边沿触发)	$Q^* = \overline{Q}$
T触发器(钟控)	$Q^* = T\overline{Q} + \overline{T}Q$

时序设计方法

- 定义
  - 时钟clock:
  - 建立时间setup time: (Tsu)
  - 保持时间hold time: (Th)

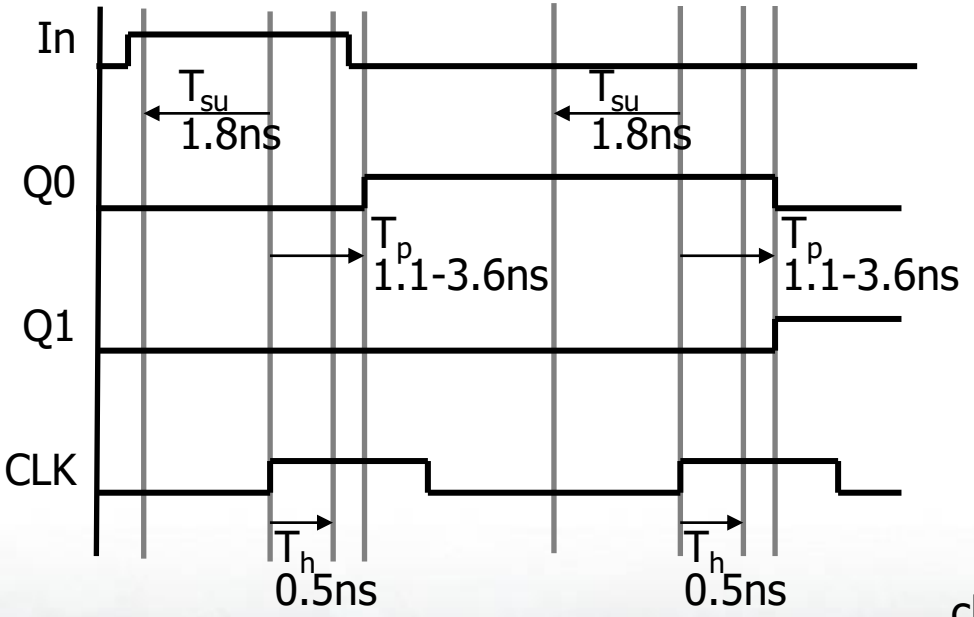


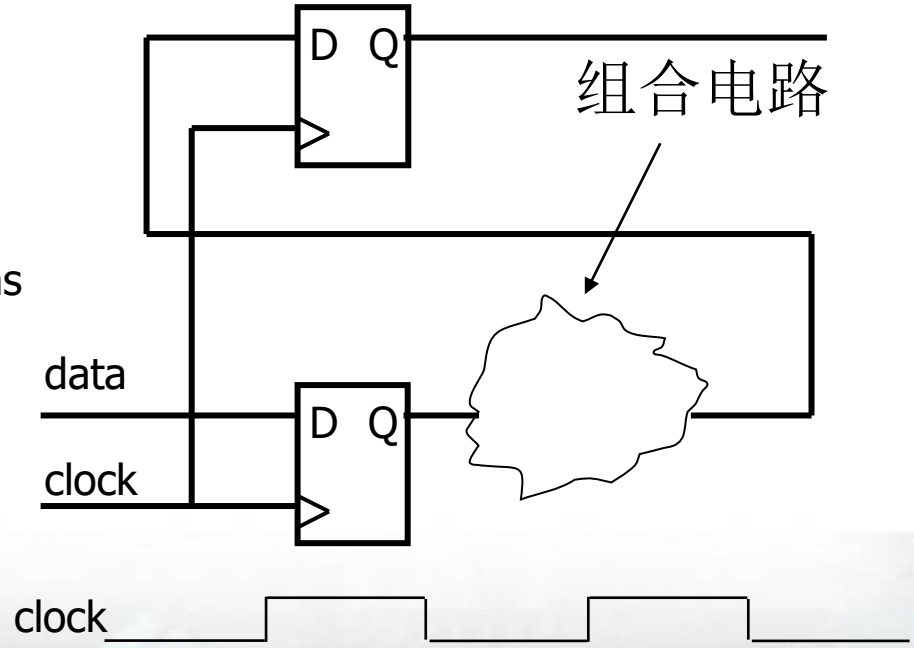




级联边沿触发器

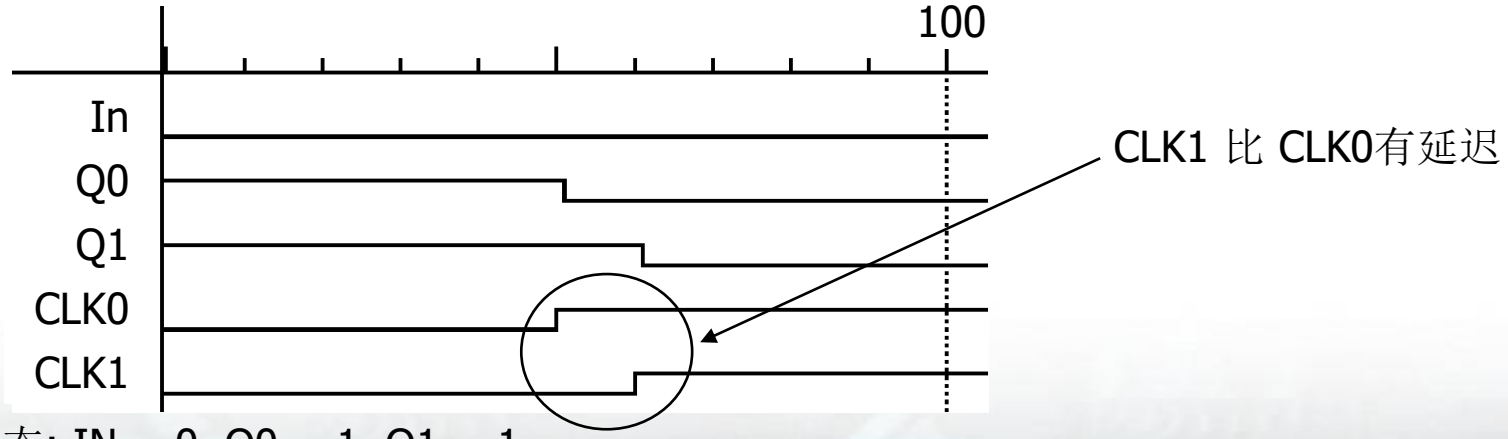
- 为什么可以工作
  - 延迟时间大于保持时间 $T_p > T_h$
  - 时钟周期大于于建立时间和延迟时间 $T_{clock} > T_p + T_{su}$
  - 可以保证后面的触发器锁存正确的值





时钟扭斜(clock skew)

- 问题
  - 正确的电路行为假设所有存储部件的下一个状态在同一个时刻由所有的存储部件测定
  - 在高性能系统中是很难的达到的，因为到达所有触发器的时钟延迟和逻辑延迟是可比的
  - 级联触发器的扭斜影响:
  - $T_{pd} + T_p > T_h + T_{skew}$;($T_{skew}$>0)
  - $T_{clock} + T_{shew} > T_{pd} + T_p + T_{su}$;($T_{skew}$<0)



原始状态: IN = 0, Q0 = 1, Q1 = 1
因为扭斜，下一个状态变为Q0 = 0, Q1 = 0而不是 Q0 = 0, Q1 = 1

7 标准模块化的时序逻辑

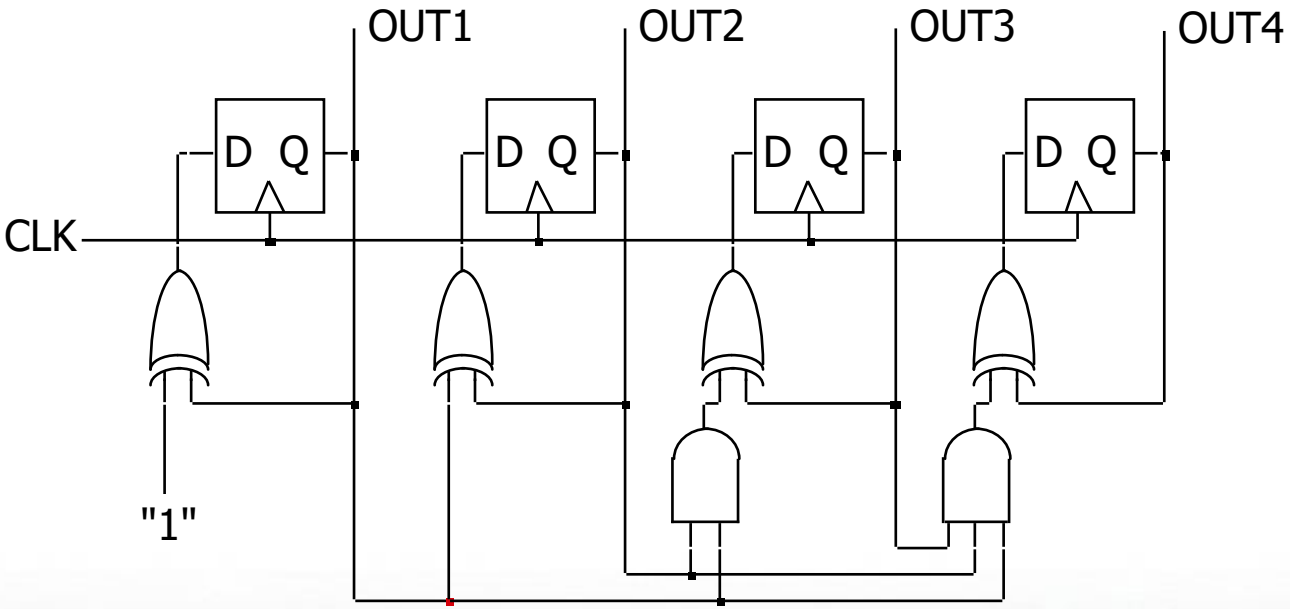
- 移位寄存器
  - 通用移位寄存器模型
- 计数器
  - 二进制和非二进制
  - 同步/异步
  - Up/Down
- 模*N*计数器
  - BCD计数器
- 其它

7.4 模*N*计数器

- 实际的数字系统设计中需要从0数到*N*-1
- 实现模*N*计数器的要求：
  - 计数器
  - *N*-1到0的翻转
- 计数器位数*n*
  - $2^{n-1} < N \leq 2^n$
  - 二进制计数器：模2^n计数器
- 分类：同步、异步

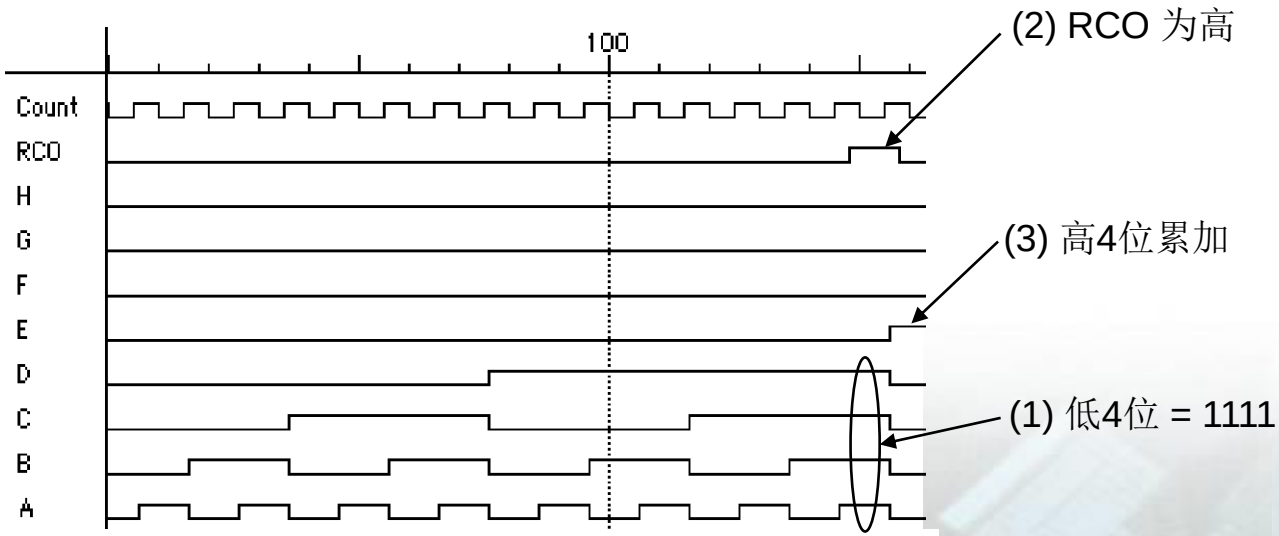
简单二进制计数器

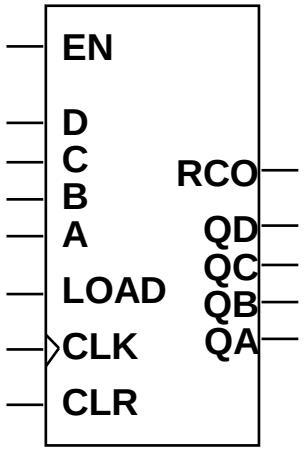
- 寄存器的逻辑（不采用多选器）
  - XOR门决定那一位什么时候反转
  - 低位全部为‘1’时



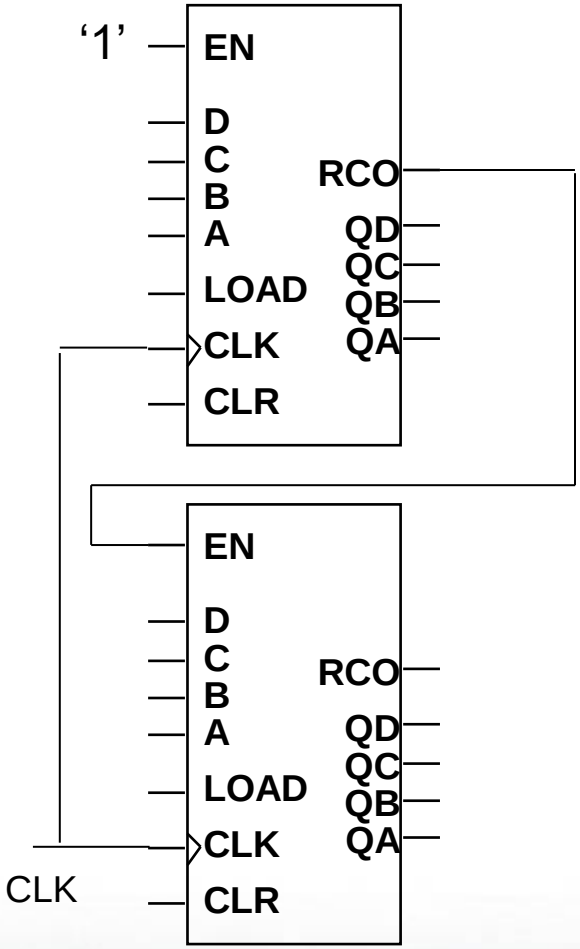
4位同步计数器

- 许多应用采用的标准模块
  - 上升沿触发器同步装载load和清除clear输入
  - 并行装载数据 D, C, B, A
  - 使能输入enable
  - RCO：形波进位输出可用于级联计数器
    - 高high：当计数器为1111
    - 与门实现

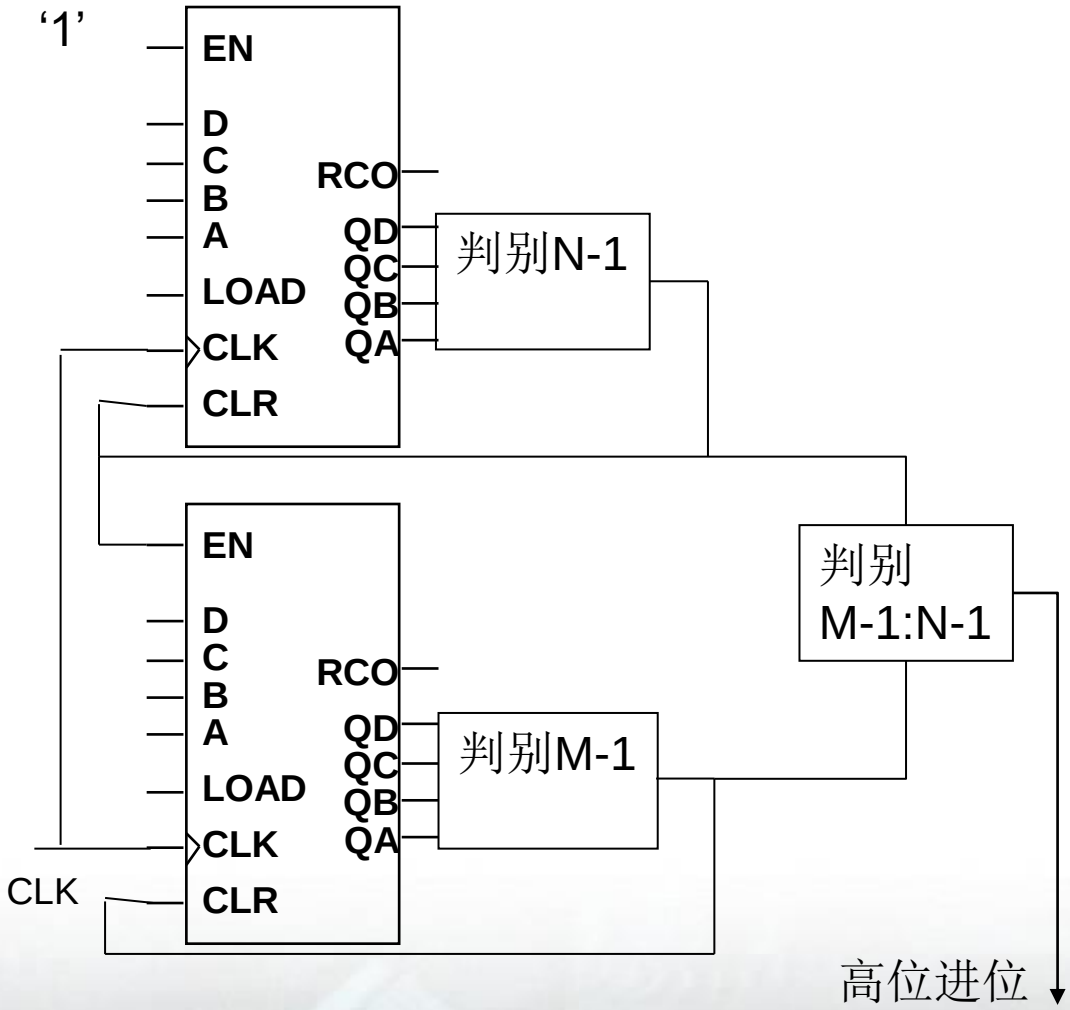




计数器的级联



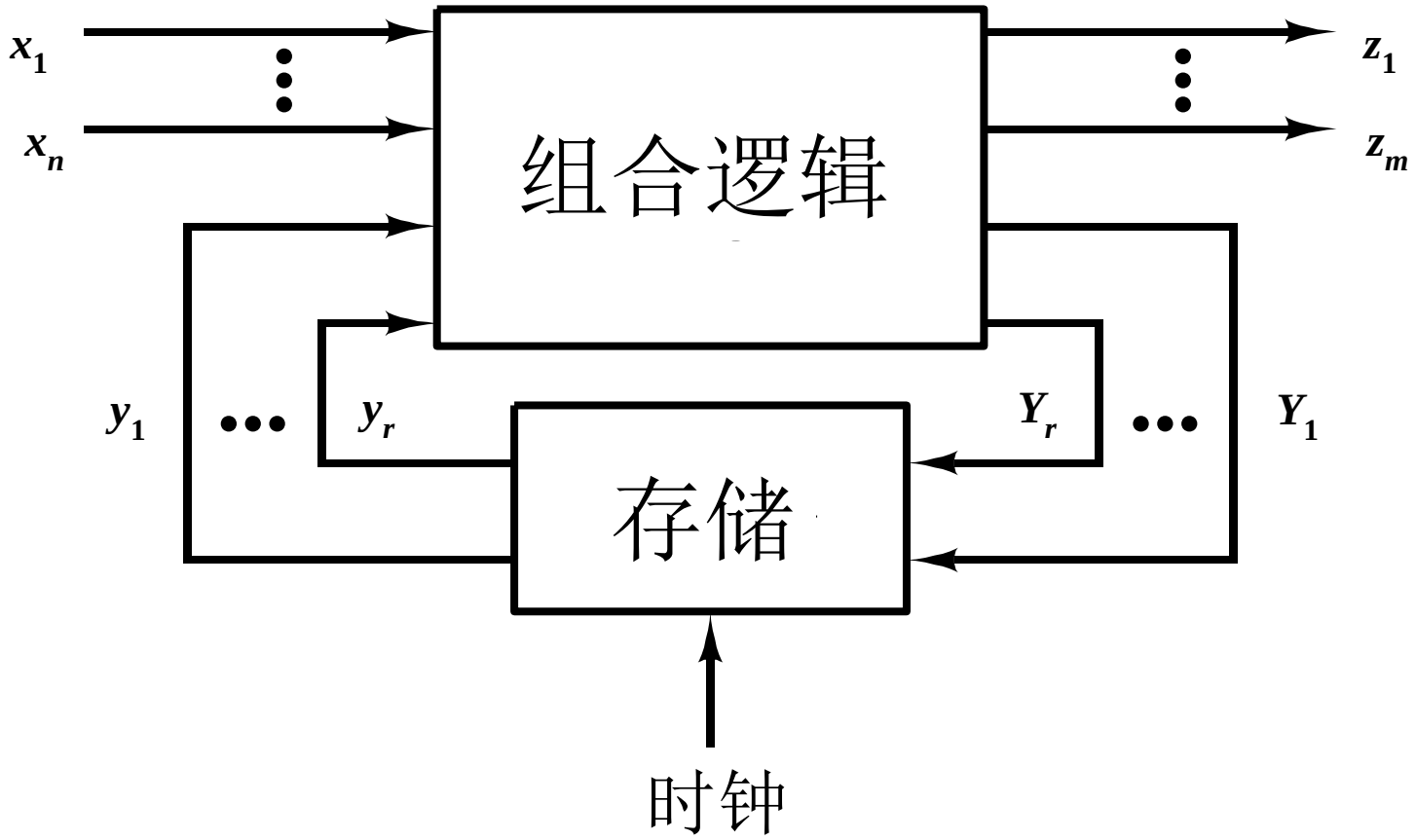
8位二进制计数器



8同步时序电路

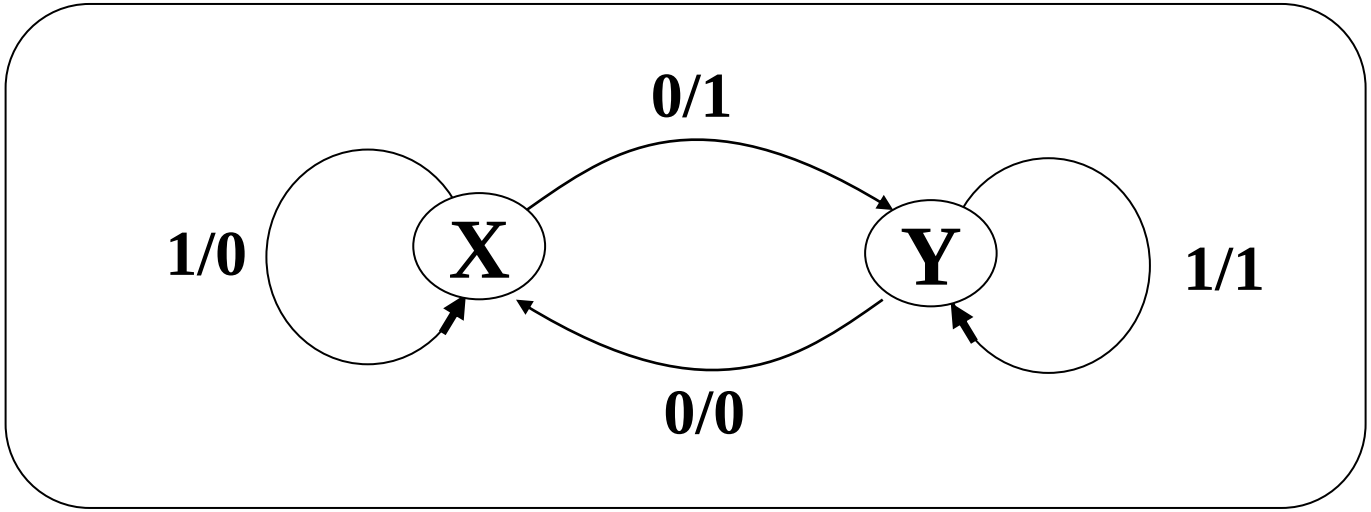
- 同步(Synchronous)时序电路
  - 时序电路中状态的改变由系统统一时钟控制
- 异步(Asynchronous)时序电路
  - 时序电路中状态的改变不受统一时钟的控制

8.1 同步时序电路模型



Mealy机

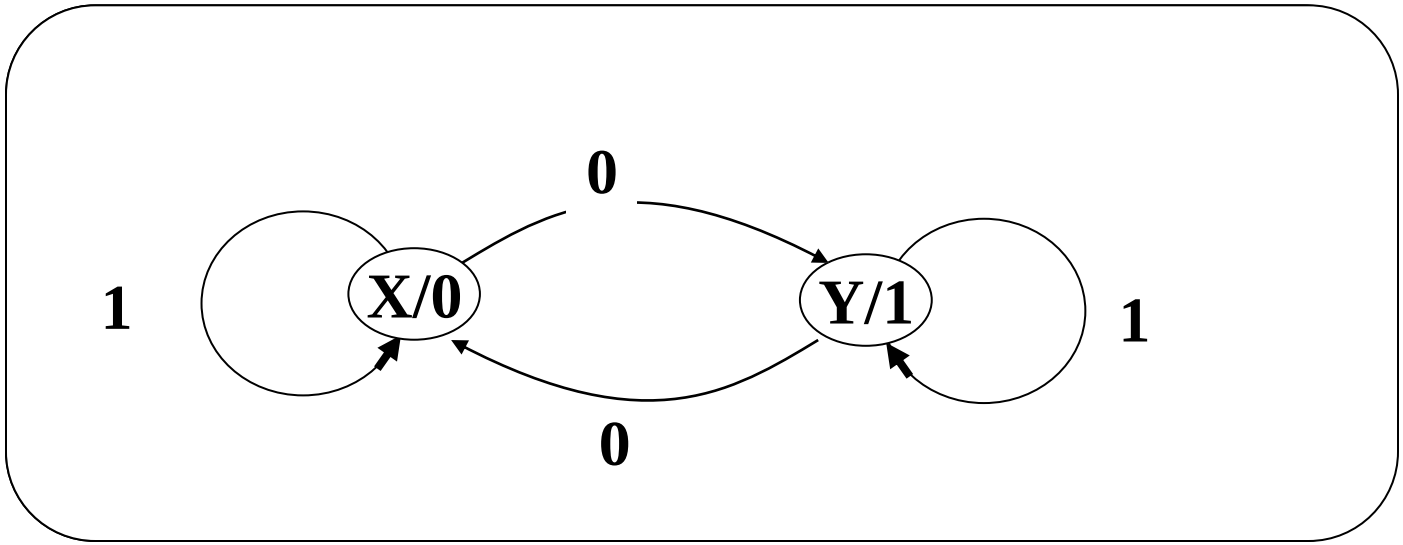
- 电路的输出由电路输入和现态决定



现态 \ 输入	0	1
X	Y/1	X/0
Y	X/0	Y/1

Moore机

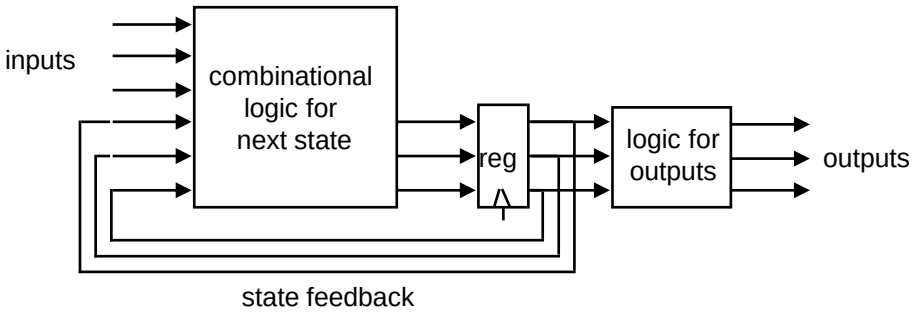
- 电路的输出只由电路的现态决定



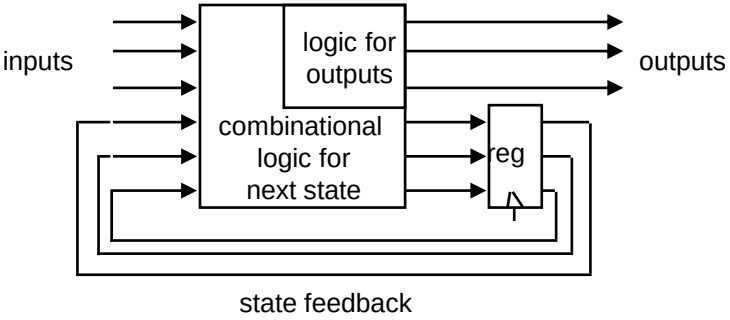
输入 / 现态	0	1	输出
x	Y	X	0
y	X	Y	1

Mealy机和 Moore机的比较

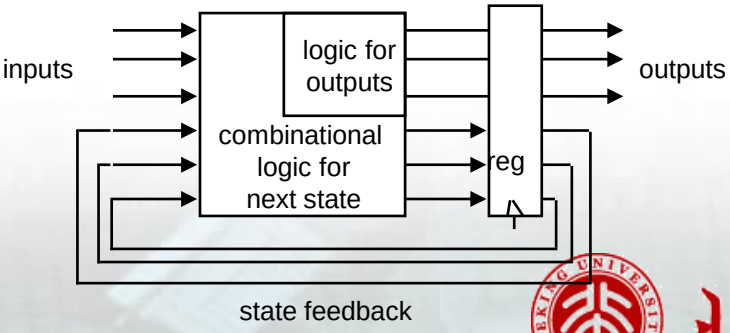
- Moore机



- Mealy机



- 同步Mealy机



状态转换表

状态分配

A -> 00
B -> 01
C -> 11
D -> 10

y1y2 \ x	0	1
00	00/0	01/0
01	00/0	11/0
11	01/0	10/0
10	11/1	10/0

Y_1Y_2/Z

同步时序电路综合

- 同步时序电路综合
  - 给定电路的功能
  - 或者给定状态图、状态表和所用触发器类型
  - 画出电路图
- 方法
  - 利用触发器的激励输入表
  - 利用触发器的特征方程

同步时序电路的综合过程

第一步:从文字描述得出状态表

第二步:化简状态表，得到最小状态的等效电路

第三步:进行状态分配，产生状态和输出的转换表

第四步:确定所用的记忆单元,导出激励卡诺图

第五步:从激励图得出开关逻辑方程，及输出方程

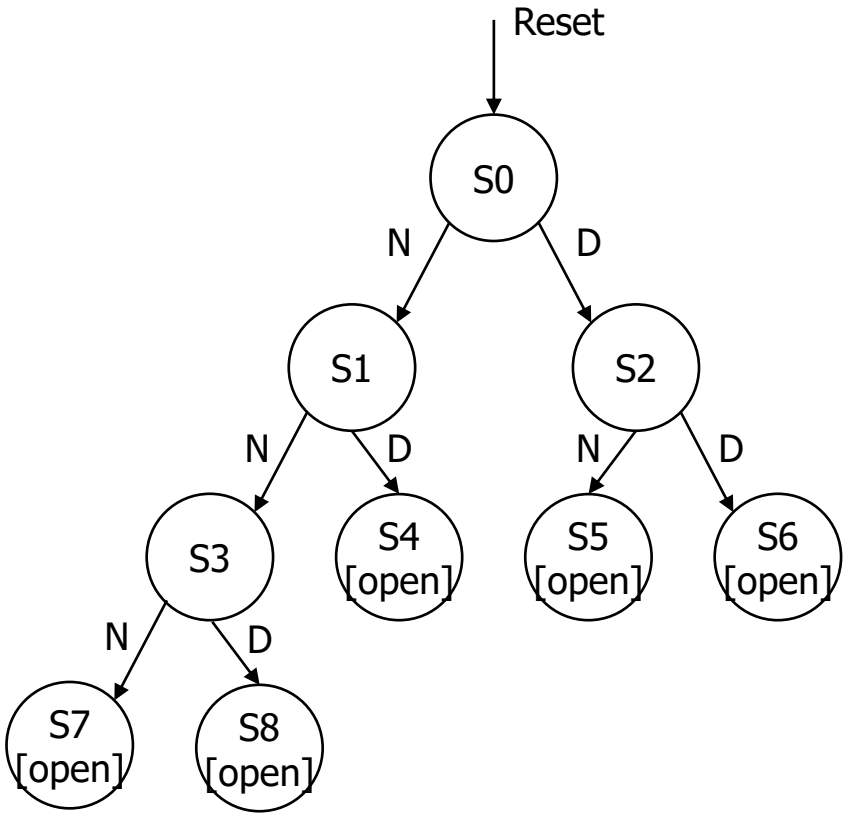
第六步:由逻辑方程和记忆单元画出电路图

同步时序电路状态机设计

- 二进制序列识别
- 串行加法器，计数器
- 有限状态自动机设计FSM
- ASM状态机表示

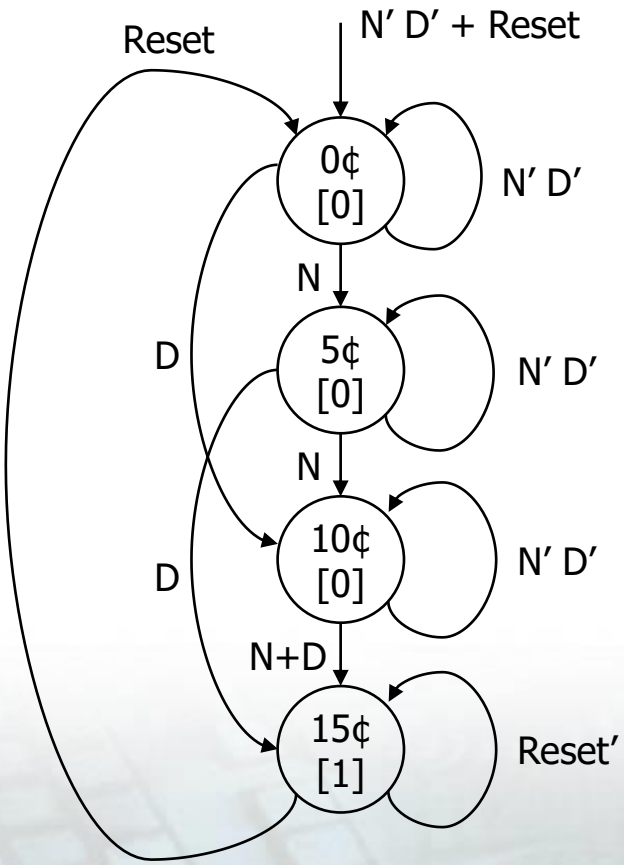
自动售货机（续）

- 适当的抽象表示
  - 列出典型的硬币输入序列：
    - 3个五分
    - 一个五分接着一个十分
    - 一个十分接着一个五分
    - 两个十分
  - 画出状态图：
    - 输入: N, D, reset
    - 输出: open chute
  - 假设：
    - 每个时钟周期已能投入一个5分或者一个10分
    - 如果N = D = 0 （没有硬币）状态不变

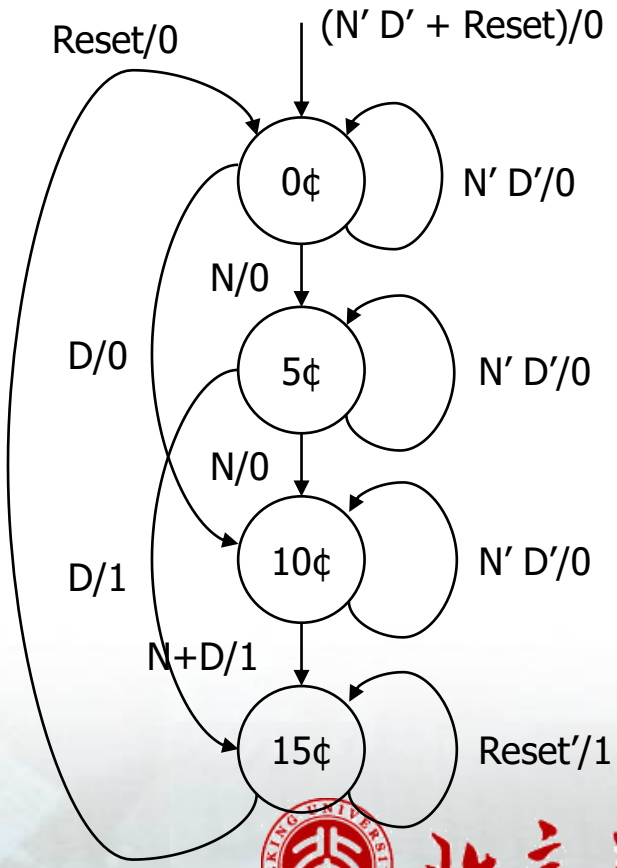


等价的Mealy机和Moore机

- Moore机



- Mealy机



例3：有限二进制串模式识别

- 有限二进制串模式识别
  - 一个输入(X) 和一个输出 (Z)
  - 输出有效的条件：在输入序列中发现…010…，当序列…100…出现后停止识别

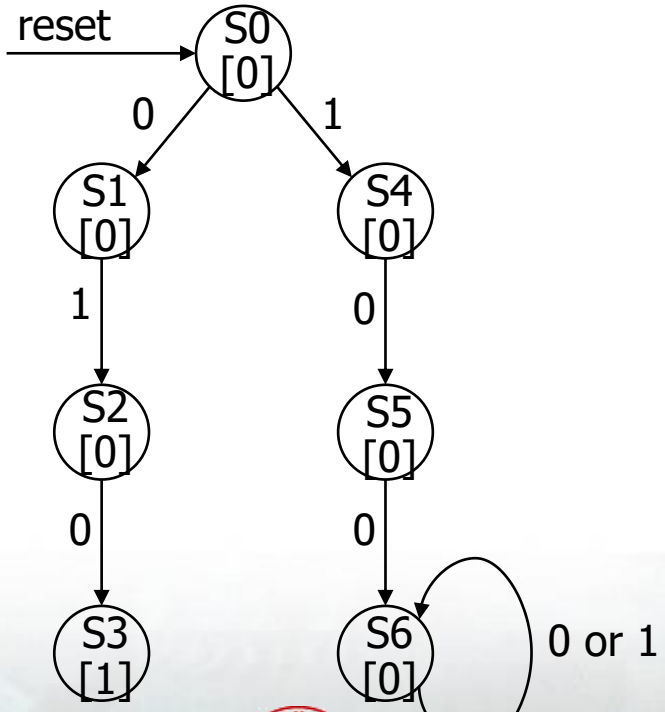
- 第1步：理解问题
  - 简单的输入/输出行为：

X: 0 0 1 0 1 0 1 0 0 1 0 …
Z: 0 0 0 1 0 1 0 1 0 0 0 …

X: 1 1 0 1 1 0 1 0 0 1 0 …
Z: 0 0 0 0 0 0 0 1 0 0 0 …

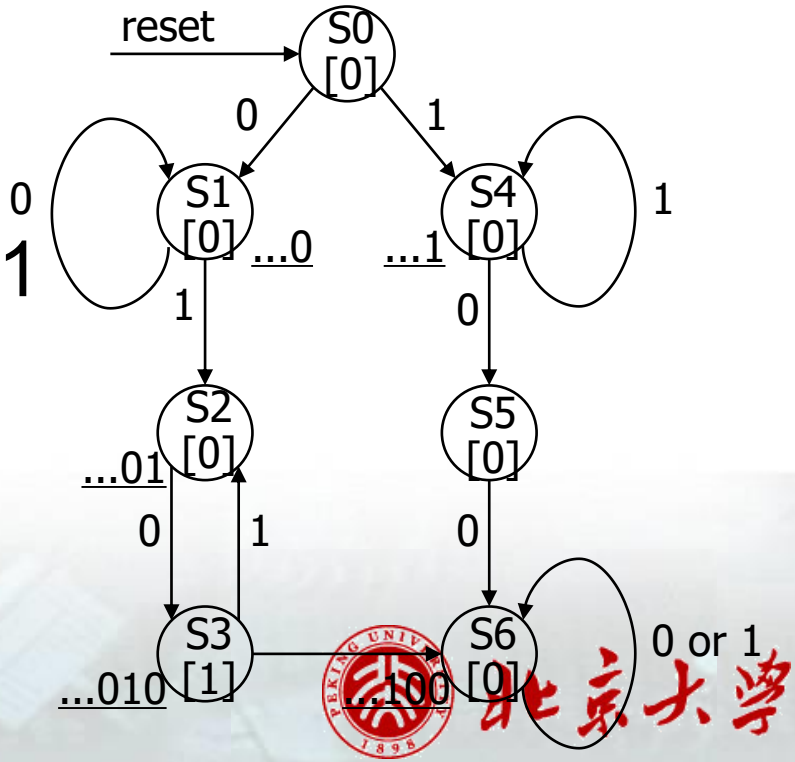
有限二进制串模式识别 (第二步)

- 画状态图
  - 必须识别的二进制串010 and 100
  - Moore机实现



有限二进制串模式识别 (第二步)

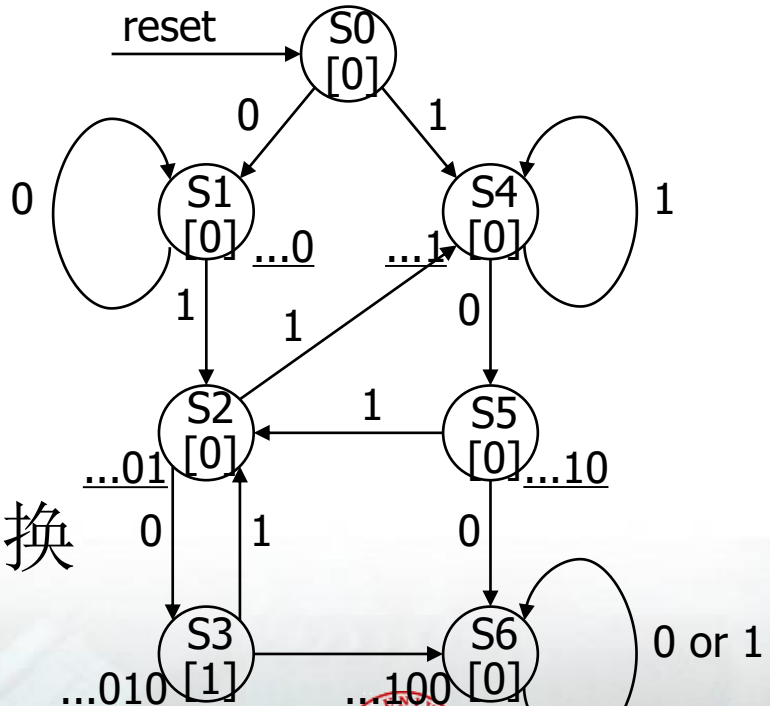
- S3的退出条件：识别…010
  - 如果下一个输入时0： …0100 = ...100 （S6）
  - 如果下一个输入时1： …0101 = …01 （S1）
- S1的退出条件：识别多0串…0
  - 没有1出现
  - 如果输入 0，循环跳回S1
- S4的退出条件：识别多1串…1
  - 没有0出现
  - 如果输入1，循环跳回S4



有限二进制串模式识别 (第二步)

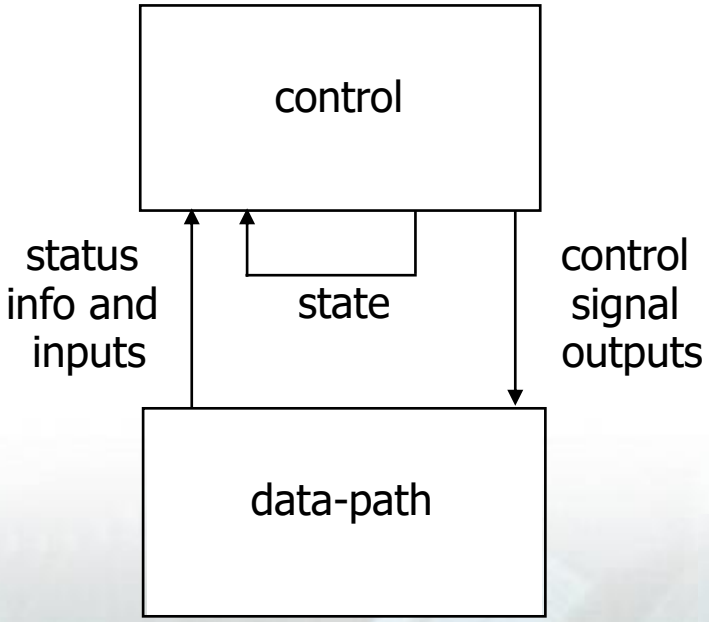
- S2 和 S5 还没有完全的跳转
  - S2 = …01；如果下一个输入1
    输入串 (01)1(00)
    S4符合这种情况
  - S5 = …10；如果下一个输入1
    输入串 (10)1(0)
    S2符合这种情况
- 尽可能复用状态
  - 寻找具有相同含义的状态
  - 减少状态数
- 所有的状态具有完全的状态转换

则完成状态机设计

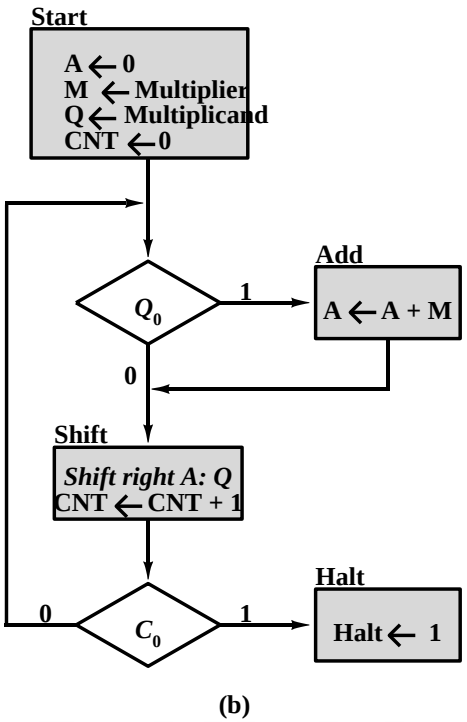


数据通路和控制

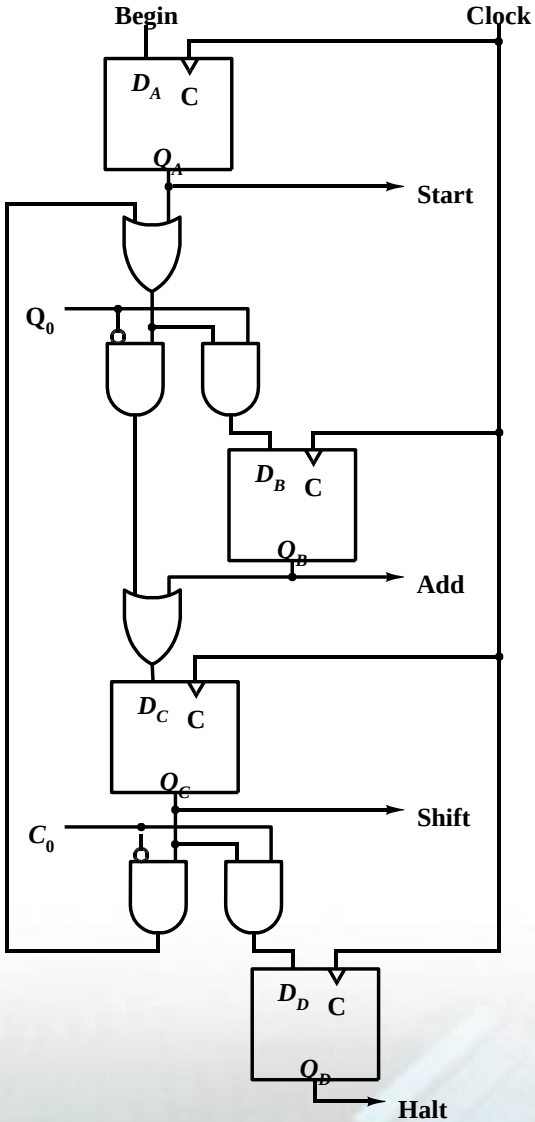
- 数字硬件系统 = 数据通路 + 控制
  - 数据通路：寄存器，计数器，组合逻辑（ALU），通讯（总线）
  - 控制：FSM产生控制信号的序列，控制数据通路的行为



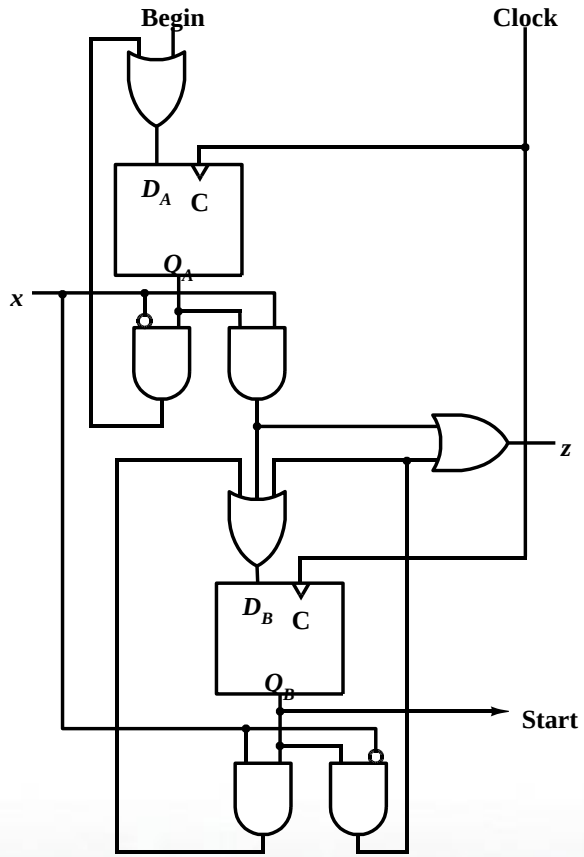
ASM与One-Hot状态编码



(b)



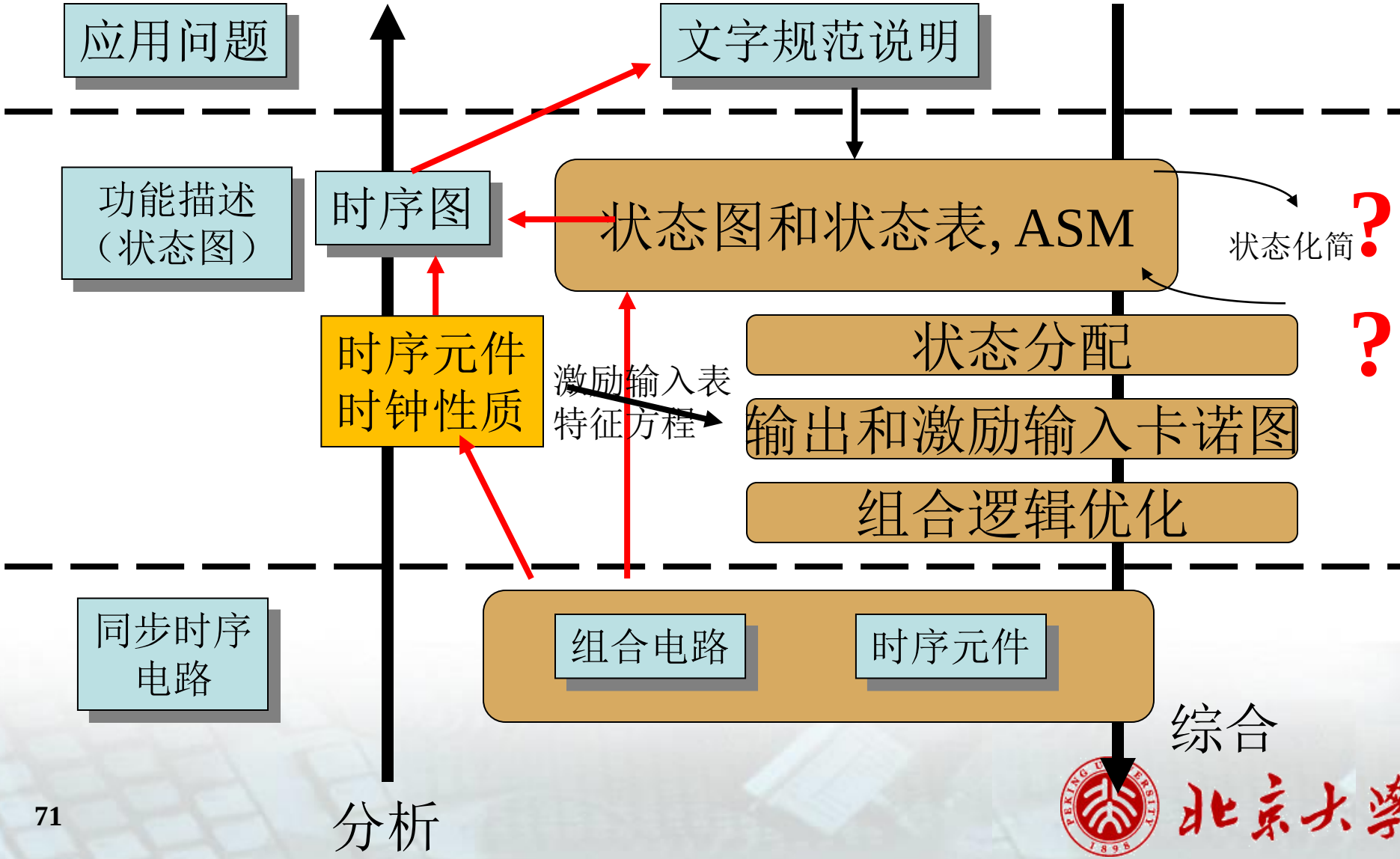
(a)



(b)

北京大学

时序电路分析与综合总结



9 同步时序电路化简

- 消除冗余状态的重要性
  - 成本：存储元件的数目和状态的数目直接相关。
  - 复杂性：电路中的状态越多，设计实现越复杂。
  - 辅助故障分析：诊断程序一般都默认为电路中没有冗余状态。

状态等价性

- 定义1：完全确定的时序电路中状态$S_1, S_2, \ldots, S_j$被称为等价的，当且仅当对于任意的输入序列，将$S_1, S_2, \ldots, S_j$中的任意状态作为初始状态，电路的输出序列都是相同的。
- 定义2：设S_i和S_j是完全确定时序电路的两个状态，S_k和S_l是在输入I_p时S_i和S_j的下一个状态，S_i和S_j是等价的当且仅当对于每一可能的I_p满足下列的条件：
  - 1. S_i和S_j的输出相同；
  - 2. 下一个状态S_k和S_l是等价的。

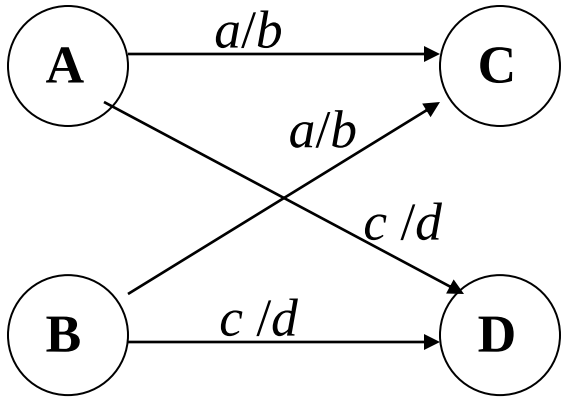
完全确定电路的状态化简

- 三种方法
  - 1 观察法
  - 2 划分法
  - 3 蕴含表法

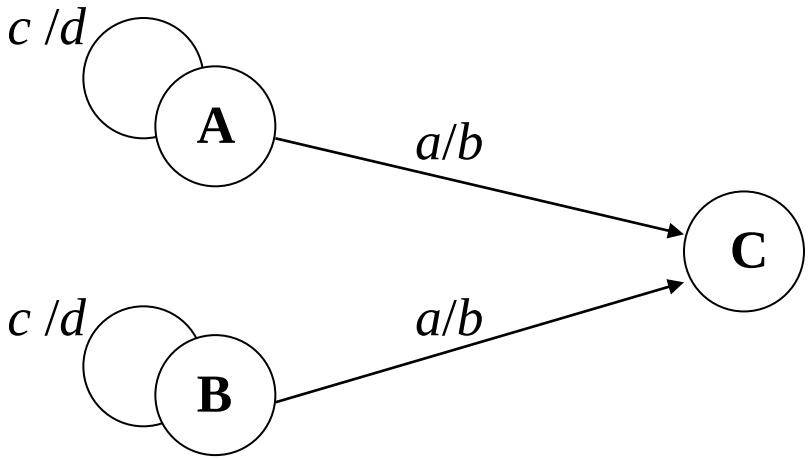
观察法的原则

- 两个状态等价
  - 在相同输入和输出的前提下
    - 下一个状态是相同的
    - 下一个状态是这两个状态之一

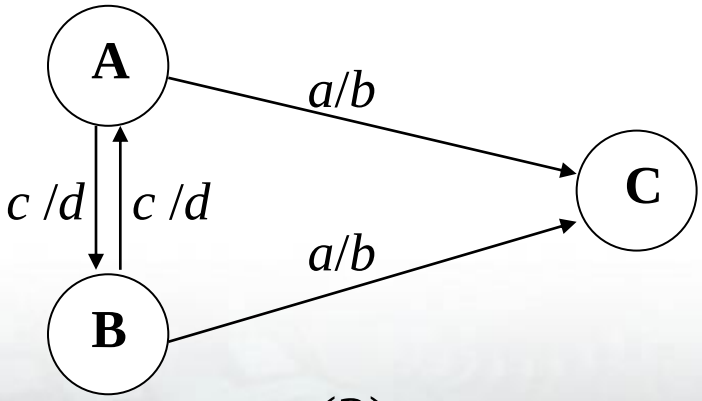
观察法的4种模式（A和B等价）



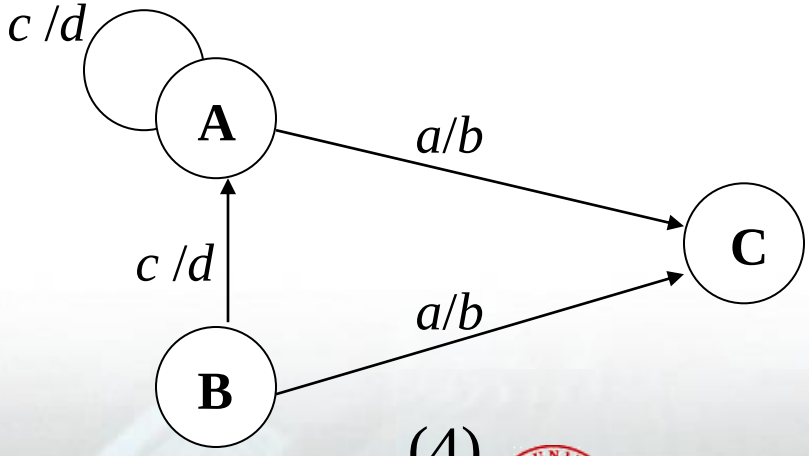
(1)



(2)

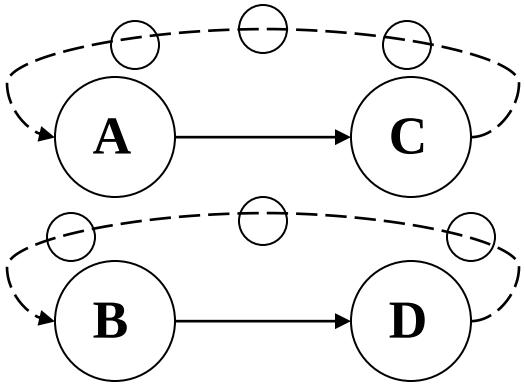


(3)

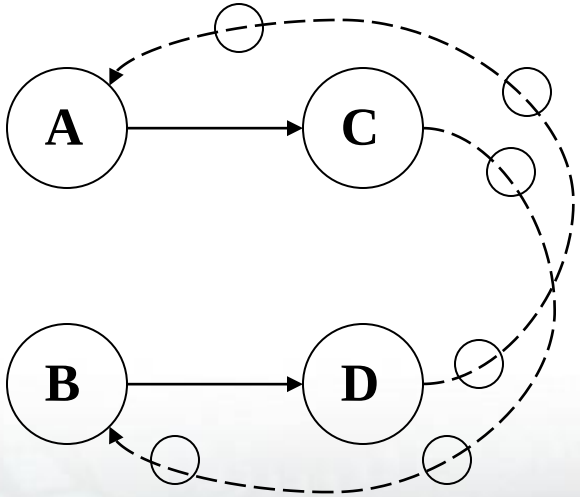


(4)

另两种模式（A和B等价）



(5)



(6)

- 对于这两种模式，采用观察法很难发现

划分法(Partitioning)——行匹配发

- 划分法是一组连续的过程。
- 每步形成的划分Pk，由一些块(block)组成，每个块中的状态是K-等价的。
- 划分步骤
  - 第一步：将输出相同的状态分在一个块内，形成1-等价划分。
  - 第二步：用下面的方法连续得到Pk, k=2,3,4,5...; 直到Pk-1=Pk。
    - 若对于每个输入,Si与Sj的次态都在同一个Pk-1的块内,则Si与Sj放在Pk的同一块内

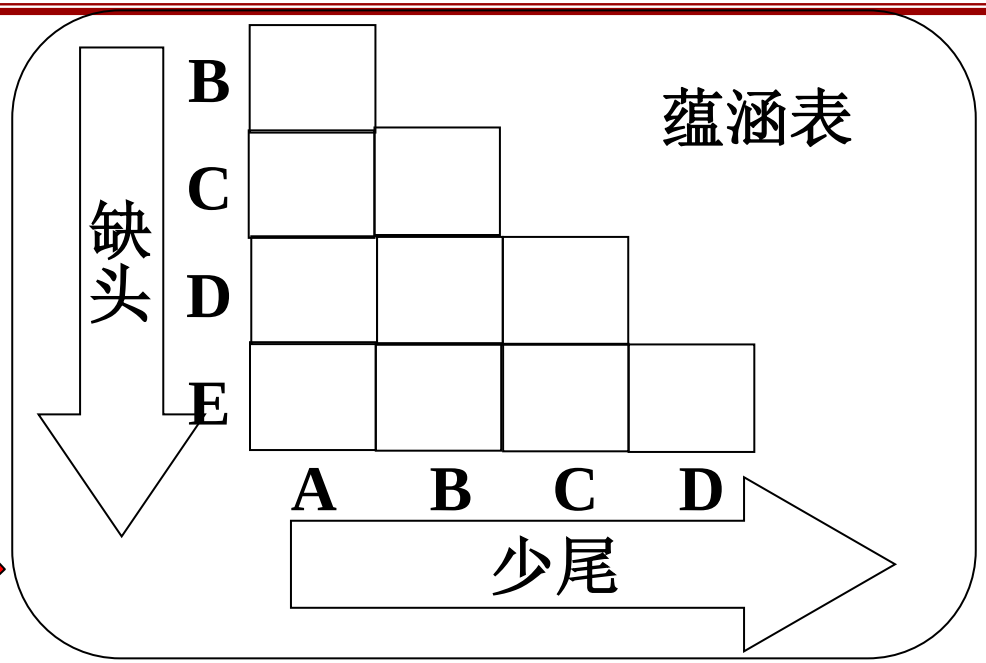
例1

	Partition blocks				Action
Partition P_0	$(ABCDE)$				
Output for x = 0	11100				Separate (ABC) and (DE)
Output for x = 1	00011				Separate (ABC) and (DE)
Partition P_1	(ABC)		(DE)		
Next state for x = 0	CCB		DE		
Next state for x = 1	BEE		BA		Separate (A) and (BC)
Partition P_2	(A)	(BC)	(DE)		
Next state for x = 0	C	CB	DE		
Next state for x = 1	B	EE	BA		Separate (D) and (E)
Partition P_3	(A)	(BC)	(D)	(E)	
Next state for x = 0	C	CB	D	E	
Next state for x = 1	B	EE	B	A	
Partition $P_4 = P_3$	(A)	(BC)	(D)	(E)	

States B and C are equivalent

蕴涵表法

	X=0	X=1
A	C/1	B/0
B	C/1	E/0
C	B/1	E/0
D	D/0	B/1
E	E/0	A/1



	A	B	C	D
B	BE			
C	BC, BE	V		
D	X	X	X	
E	X	X	X	BA

	A	B	C	D
B	~~BE~~			
C	~~BC, BE~~	V		
D	X	X	X	
E	X	X	X	~~BA~~

蕴含表法

(1)画出空的蕴涵表，蕴含表的方格表示所有可能的状态对。

(2)检查蕴涵表的每个方块，当对应的两状态输出不相同时，在该方块内打“X”。

(3)添完蕴涵表:对于每一种输入可能，将输出相同的次态对添入对应的蕴涵单元内。当蕴涵表单元的蕴涵项等于相对应的两状态或为同一个状态时：打对号“V”；当单元包含的所有蕴涵对都变为“V”时，其对应两状态位等价，对应的单元可变为“V”。

(4)处理蕴涵表，确定每个状态对的等价性。当蕴涵单元包含的蕴涵项有一个为“X”时,则其对应的状态对不等价，该单元画“X”.

(5)从蕴涵表得到等价状态对，导出等价划分：没有画“X”的单元对应的状态对为等价对.

三种方法的比较

- 观察法：直观，功能不强。
- 划分法：功能强，可编程性不太强。
- 蕴含表法：功能强，可编程性强，步骤繁琐，所用时间多。

状态分配指导原则

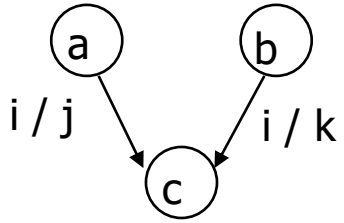
- 问题：如何指导选择状态分配方案来降低次态生成逻辑的复杂性。
- 总体思路：选择一种状态编码方式，使在状态转换表（卡诺图）中的“1”组成尽可能大的组（质蕴含项）。
- 工具：状态分配表，蕴含图

启发式状态分配

- 规则1
  -

I	Q	Q^+	O
i	a	c	j
i	b	c	k

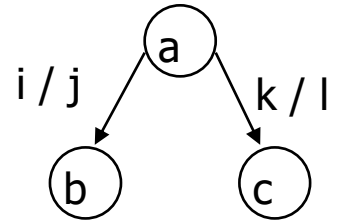
c = i * a + i * b



- 规则2
  -

I	Q	Q^+	O
i	a	b	j
k	a	c	l

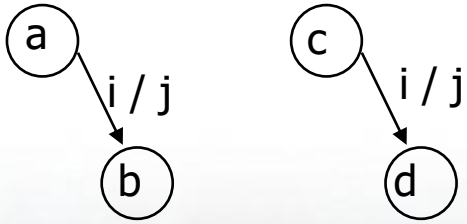
b = i * a
c = k * a



- 输出规则

I	Q	Q^+	O
i	a	b	j
i	c	d	j

j = i * a + i * c
b = i * a
d = i * c

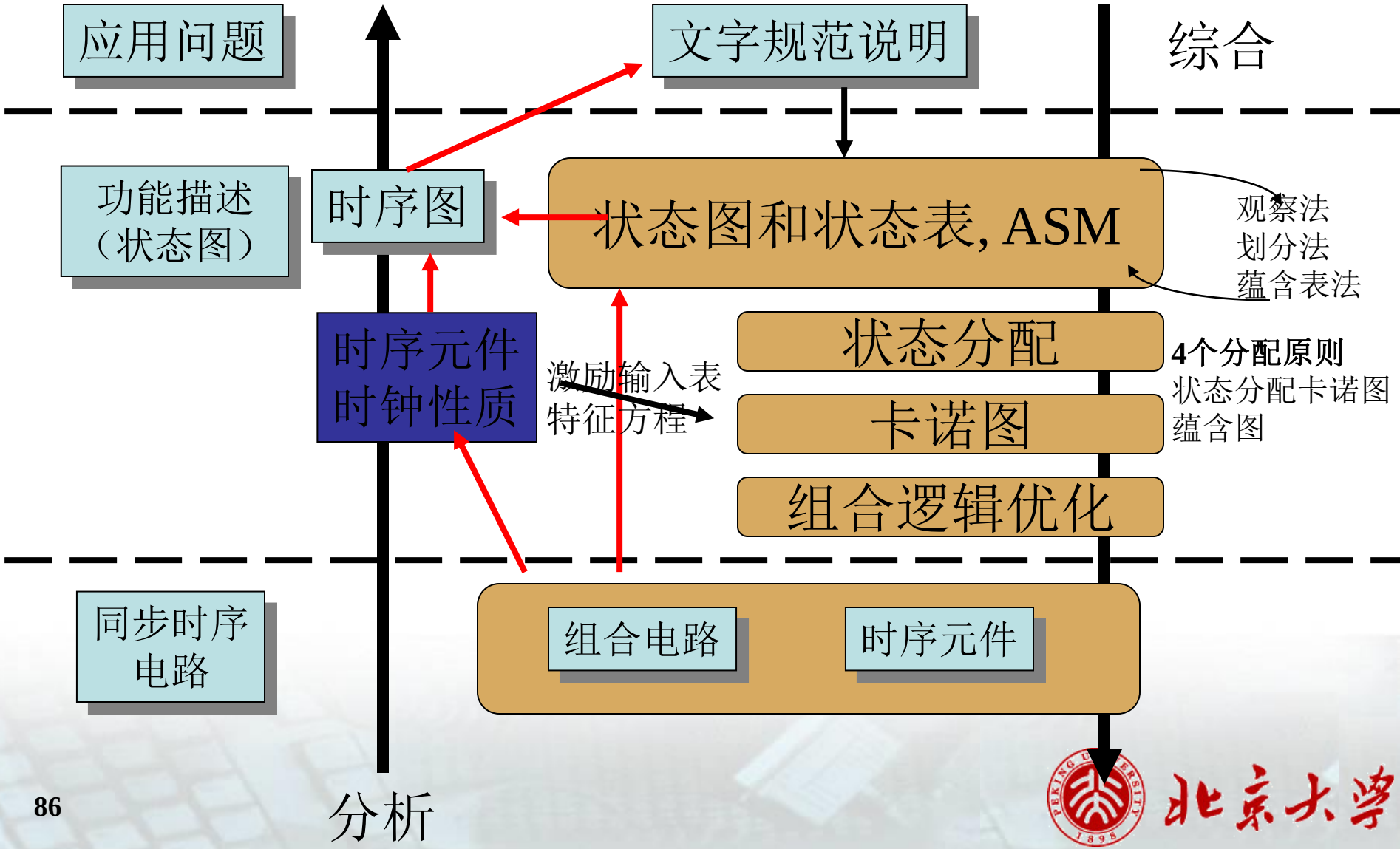


状态分配规则

- 规则4：已经利用规则1、2、3确定分配相邻编码的状态对，其次态对应该分配相邻的编码。

- 规则5：找到在化简的状态表中作为次态次数最多的状态，分配“全0”的编码，然后利用规则1、2、3、4进行状态分配。尽可能的满足所有的规则。

时序电路化简的总结



学习数字逻辑设计的收获

- 硬件工程师
- 软件工程师

- 设计基础和理论
- 足够的细心和耐心
- EDA工具

北京大学

《数字系统逻辑设计》课程结束
The End
“Digital System Logic Design”