第七章 数字逻辑电路

ENIAC - 背景

- Electronic Numerical Integrator And Computer（电子数字积分计算机）
- Eckert 和 Mauchly
- University of Pennsylvania
- 武器弹道表
- 1943年开始
- 1946年完成
- 使用到1955年

ENIAC - 细节

- 十进制
- 18,000 个真空管
- 30 吨
- 15,000 平方英尺
- 140 kW
- 5,000次加法/秒

晶体管

- 取代真空管
- 更小
- 更便宜，由硅制造而成
- 散热更少
- 1947年 Bell 实验室发明
- William Shockley 等人
- 1956年度的诺贝尔物理学奖

集成电路

- 采用一定的工艺，把一个电路中所需的晶体管、二极管、电阻、电容和电感等元件及布线互连一起，制作在一小块或几小块半导体晶片或介质基片上，然后封装在一个管壳内，成为具有所需电路功能的微型结构
- 所有元件在结构上已组成**一个整体**——整个电路的体积大大缩小，且引出线和焊接点的数目也大为减少，从而使电子元件向着**微小型化**、**低功耗**和**高可靠性**方面迈进了一大步

微电子技术

- 建立在以集成电路为核心的各种半导体器件基础上的高新电子技术
- 大规模集成电路
  - 每一单晶硅片上可以集成制作一千个以上的元器件
- 超大规模集成电路
  - 元器件集成度在一万至十万以上

晶体管

- 微处理器
  - Intel Pentium 4 (2000): 48 million
  - IBM PowerPC 750FX (2002): 38 million
  - IBM/Apple PowerPC G5 (2003): 58 million
  - Intel Core(酷睿), 2.91亿
  - iPhone A10, 33亿

- 计算机采用二进制表示数值
- 如何存储这些二进制数值，进行计算？
- 数学基础——二进制逻辑运算

位组合的逻辑运算

位组合的逻辑运算

- 注意：此处的逻辑运算为位运算，与3.2.6节的逻辑运算不同
- “逻辑”，名称来源于使用0和1表示逻辑值“假”和“真”，而位组合的逻辑运算则与这个原始意义没有多大关系

与函数（AND）

- 二元函数，需要两个源操作数
- 只有当两个源操作数都是1时输出才为1，否则为0

真值表

- 有n+1列和2^n行
- 前面n列对应着n个源操作数
  - 既然每个源操作数都是0或1中的一个，那么源操作数组合就有2^n种可能
  - 每组这样的值的组合（或称为输入组合）使用真值表的一行表示出来
- 真值表最后一列表示每种组合的输出

与函数

- 二元函数，需要两个操作数
- 如果两个操作数中有一个为0，那么与函数的输出就为0
- 当且仅当两个操作数都为1时，与函数的输出为1

与函数（AND）

- 2列源操作数，4种输入组合
- 输出为0001

A	B	AND
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

按位运算

- 对两个*m*位的位组合对应位上的数字按位运算
- 位组合的编号规则
  - 自右向左，顺序编号，最右边的一位是[0]，最左边是[*n*-1]
  - 32位组合A

0001 0010 0011 0100 0101 0110 0111 1000

  - [31]位是0，[30]位是0，[29]位是0，[28]位是1
  - 记为A[31：0]

按位与运算

- 对两个*m*位的位组合对应位上的数字按位做与运算
  - *a*，*b*是8位的位组合，*c*是*a*和*b*做与运算的结果

  a:　00111010

  b:　11110000

  c:　00110000
  - *c*的左边4位与*a*的左边4位相同，而*c*的右边4位均为0
- 与0做与运算结果为0，与1做与运算结果保持不变

位屏蔽/掩码

- 假设有一个8位的位组合，称为*A*，最右边的两位有特殊的重要性。根据存储在*A*中的最右边两位数，要求计算机处理4个任务之一。如何把这两位孤立出来？
  - 位屏蔽为00000011
  - 和*A*做与运算，从7位到2位的位置上都为0，而0位和1位中的原来的值还在0和1的位置上
  - 屏蔽了7位到2位上的值，孤立出有重要性的0位和1位
  - 结果将是四种组合之一：00000000，00000001，00000010或00000011

或函数

- 二元函数，需要两个操作数
- 如果两个操作数中有一个为1，那么或函数的输出就为1
- 当且仅当两个操作数都为0时，或函数的输出为0

或函数

- 2列源操作数，4种输入组合
- 输出为0111

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

按位或运算

- 对两个*m*位的位组合**按位**进行或运算，也被称为**包含或**运算（inclusive-OR）
  - *a*，*b*是8位的位组合，*c*是*a*和*b*做或运算的结果

  a:　00111010

  b:　11110000

  c:　11111010
  - *c*的左边4位均为1，右边4位与*a*的右边4位相同
- 与1做或运算结果为1，与0做或运算结果保持不变

非函数

- 一元函数，只作用于一个操作数
- 结果通过对输入进行补运算，即取反操作得到，也被称作补运算
- 当输入为1 时，输出为0；输入为0时，结果为1

A	NOT
1	0
0	1

按位非运算

- 对一个*m*位的位组合**按位**进行非运算
  - *c*是对*a*进行非运算的结果

a:　　00111010

c:　　11000101

异或函数

- 异或（exclusive-OR，XOR），是一个二元函数，需要两个源操作数
- 若两个源操作数不同则异或运算输出为1，否则为0，即“相异为1，相同为0”

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

按位异或运算

- 对*m*位的位组合做异或运算
- 假如*a*和*b*是8位的位组合，*c*是*a*和*b*的异或运算：

  a：00111010

  b：11110000

  c：11001010

  - *c*的左边4位为*a*的左边4位按位取反的结果，而*c*的右边4位则与*a*的右边4位相同
- 与1做异或运算结果表示取反，与0做异或运算结果保持不变

按位取反

- 假如*a*是8位的位组合，*b*是11111111，*c*是*a*和*b*的异或运算：

  a：00111010

  b：11111111

  c：11000101

- c即为a取反的结果

判断两个位组合是否相同

- 既然只有当相应的位上是相同的值时，XOR函数才得到0，那么如果异或函数的输出全为0，两个位组合就是相同的

布尔代数

- 真值表是表示逻辑运算的一个方便的方法
- 还有一种描述逻辑函数的方法是采用逻辑表达式
  - 可以通过布尔代数实现

基本运算符

- 在布尔代数中，与普通代数不同的是，它的数值取值非0即1
- 三种典型的运算符，分别表示或、与、非函数：
  - “与”运算符使用符号“•”表示
    - A•B，表示A AND B，当且仅当二者取值都为1时，结果才为1
  - “或”运算符使用符号“+”表示
    - A+B，表示A OR B，当其中至少有一个变量取值为1时，结果为1
  - “非”运算符使用符号“-”表示
    - $\overline{A}$ 表示NOT A，当A取值为1时，结果为0；当A取值为0时，结果为1

逻辑完备性

- 其他任何逻辑函数都可以写成这三种基本逻辑运算符的逻辑组合
  - 异或函数，可以写成：$(\overline{A} \bullet B) + (A \bullet \overline{B})$
  - 可以利用该逻辑组合的**真值表来证明**

C的按位运算符

- C语言具备一些低级语言的特征，按位运算就是其中之一
- C语言可以对数值进行位运算，这个运算符集合被称作按位运算符
  - “&”：按位“与”运算
  - “|”：按位“或”运算
  - “~”：按位“非”
  - “^”：按位“异或”
  - “<<”：执行左移
  - “>>”：执行右移

示例

- 16位整数
  - 0x1234 | 0x5678　　/*等于0x567C*/
  - 0x1234 & 0x5678　　/*等于0x1230*/
  - 0x1234 ^ 0x5678　　/*等于0x444C*/
  - ~0x1234　　/*等于0xEDCB*/
  - 1234 & 5678　　/*等于1026*/

移位运算符

- “<<”执行左移
- “>>”执行右移
- 第一个操作数是被移位的数值
- 第二个操作数是移动的位数
- 左移，零填充
- 右移，符号扩展

示例

- 16位整数
  - 0x1234 << 3　　/*等于0x91A0*/
  - 0x1234 >> 2　　/*等于0x048D*/
  - 1234 << 3　　/*等于9872*/
  - 1234 >> 2　　/*等于308*/
  - 0x1234 << 5　　/*等于0x4680，结果仍是16位*/
  - 0xFEDC >> 3　　/*等于0xFFDB，需进行符号扩展*/

优先级及结合性

- 优先级
  - “非”运算符有最高的优先级
  - 左移和右移运算符有相同的优先级
  - 然后依次是“与”、“异或”、“或”
- 自左向右结合

示例

- x = a & b;　　/*如果a=3，b=4，x=0*/
- x = a | b;　　/*如果a=3，b=4，x=7*/
- x = a ^ b;　　/*如果a=3，b=4，x=7*/
- x = a << 1;　　/*如果a=3，x=6*/
- x = b << a;　　/*如果a=3，b=4，x=32*/
- x = b >> a;　　/*如果a=3，b=4，x=0*/
- x = ~a | ~b;　　/*如果a=3，b=4，x=-1，因为x是有符号整数*/

习题（一）

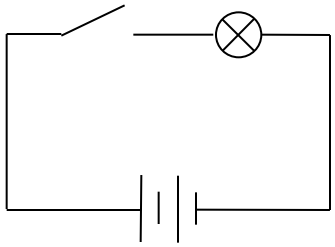
- 书面作业
- 6. 9
- 6. 16

- 上机作业
- 6. 20

晶体管

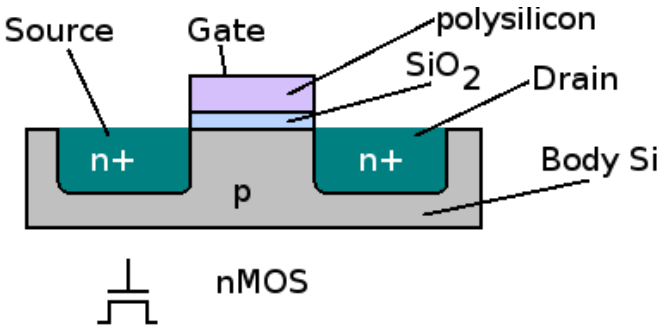
使用开关的简单电路

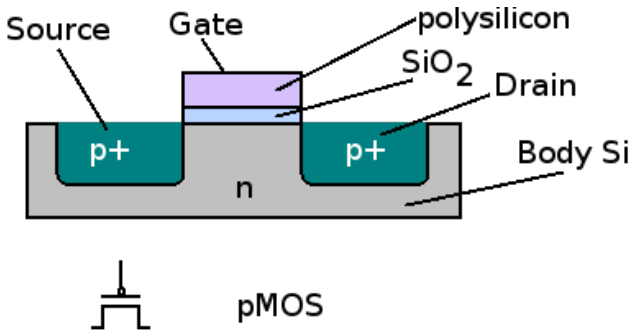
- 操纵开关
- 控制电路的合与开，从而使电灯亮或灭



MOS晶体管

- MOS（Metal-Oxide Semiconductor，金属氧化物半导体）晶体管
- 两种类型：P型和N型

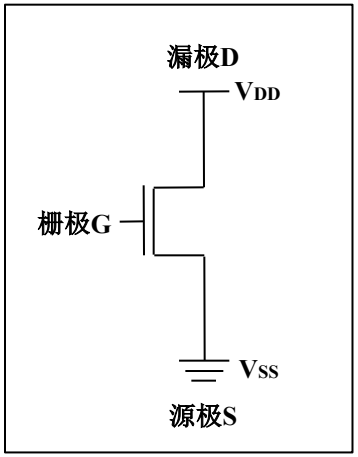




- 逻辑上起到开关的作用

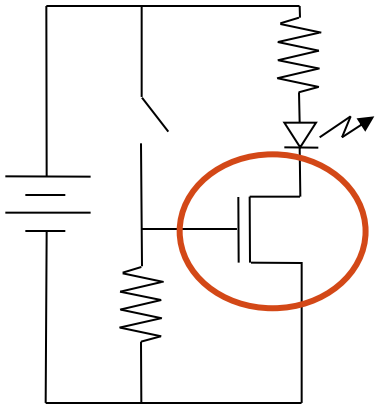
N型MOS晶体管

- 三个终端
- 如果栅极被加以3.3伏电压，从源极到漏极的连接就相当于一段电线，即：在源极和漏极之间存在一个闭合回路，即**导通**
- 如果栅极被加以0伏电压，在源极和漏极之间的连接就被断开，在源极和漏极之间存在一个断路，即**截止**



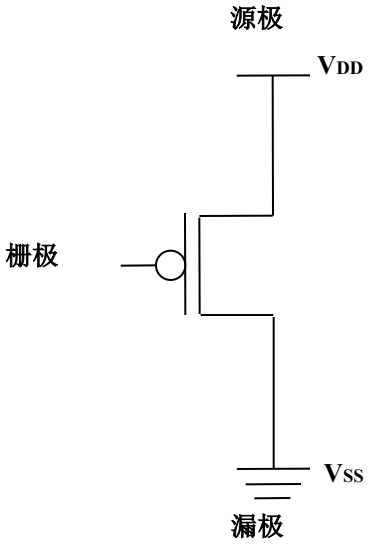
晶体管——电子开关

- 闭合开关，N型晶体管的栅极就被加以3.3伏电压，此时，晶体管相当于一段导线，从而形成回路，使LED发光
- 当打开开关，栅极被加以0伏电压时，晶体管则相当于一个断路，电路被断开，LED不亮



P型MOS晶体管

- 工作原理与N型晶体管恰恰相反
- 当给栅极提供的电压为0伏时，P型晶体管像一段电线，构成闭合回路
- 当所提供的电压为3.3伏时，就出现断路



CMOS电路

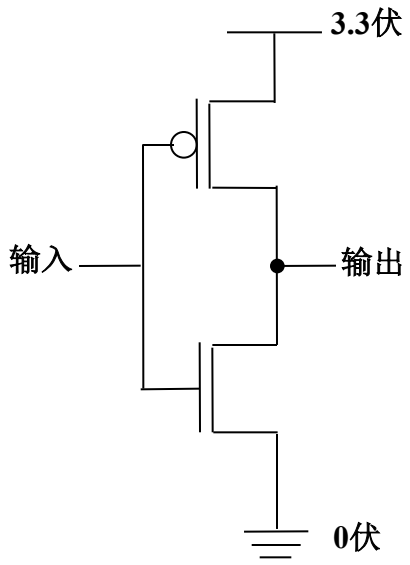
- N型晶体管要求源极不能接电源正极
- P型晶体管则要求源极不能接地
- P型和N型晶体管以互补的方式工作
- CMOS（Complementary MOS）电路
  - 既包含P型晶体管又包含N型晶体管的电路，互补金属氧化物半导体电路

门电路

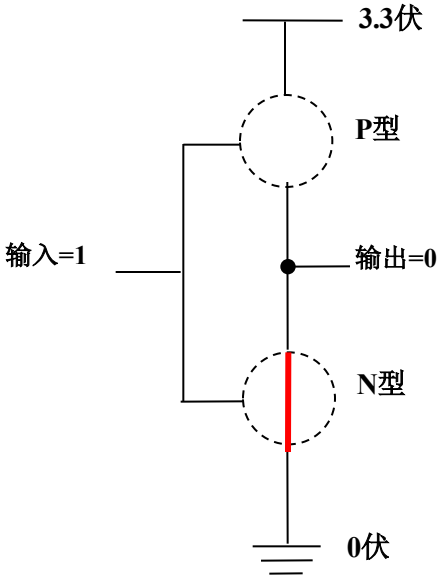
- 只使用MOS晶体管，就可以构建最基本的门电路
- 与门、或门、非门
  - 实现与、或、非逻辑运算的晶体管电路

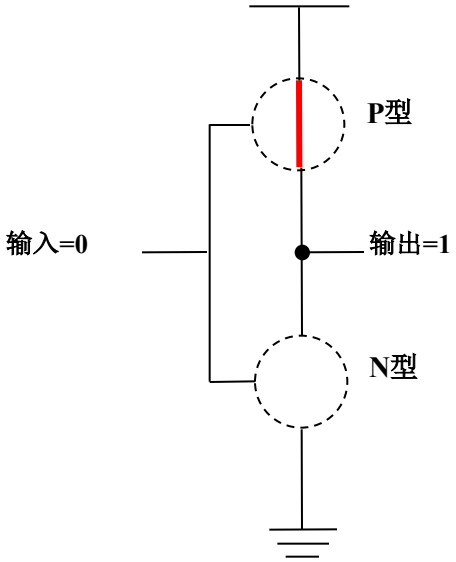
门电路

反相器(非门)



- 栅极连在一起，作为输入端；漏极连在一起，作为输出端；
- 注意
  - PMOS管的源极接电源正极；NMOS管的源极接地





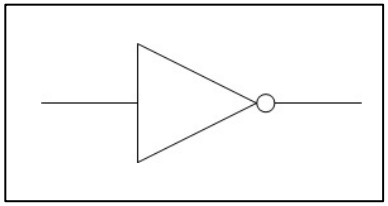
输入	输出
0伏	3.3伏
3.3伏	0伏

真值表

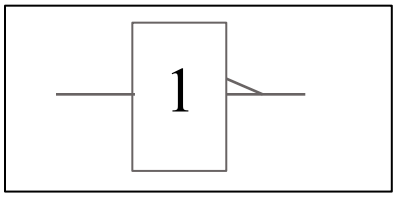
输入	输出
0	1
1	0

非门符号表示

- ANSI/IEEE Std 91-1984
  - IEEE Standard Graphic Symbols for Logic Functions
  - 形状特征型符号

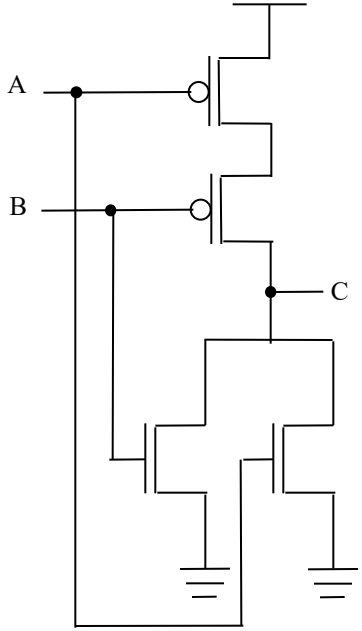


- IEC 60617-12
  - International Electrotechnical Commission, 国际电工委员会, Graphical Symbols for Diagrams-Part 12: Binary Logic Elements
  - 矩形国标符号

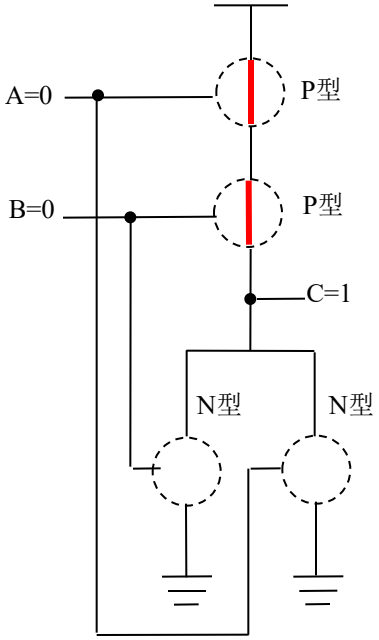


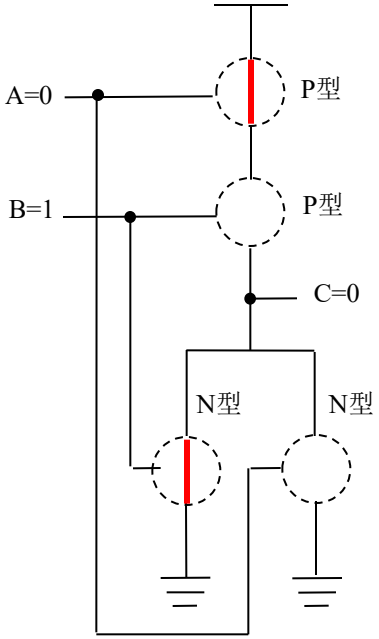
或非门

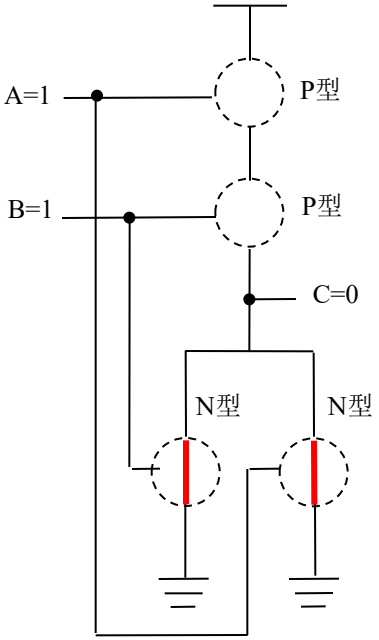
- 顶部串联，底部并联.



或非门



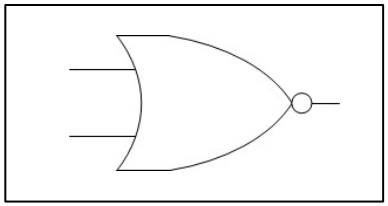




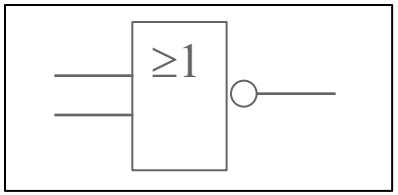
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

或非门符号表示

- ANSI/IEEE Std 91-1984
  - 形状特征型符号

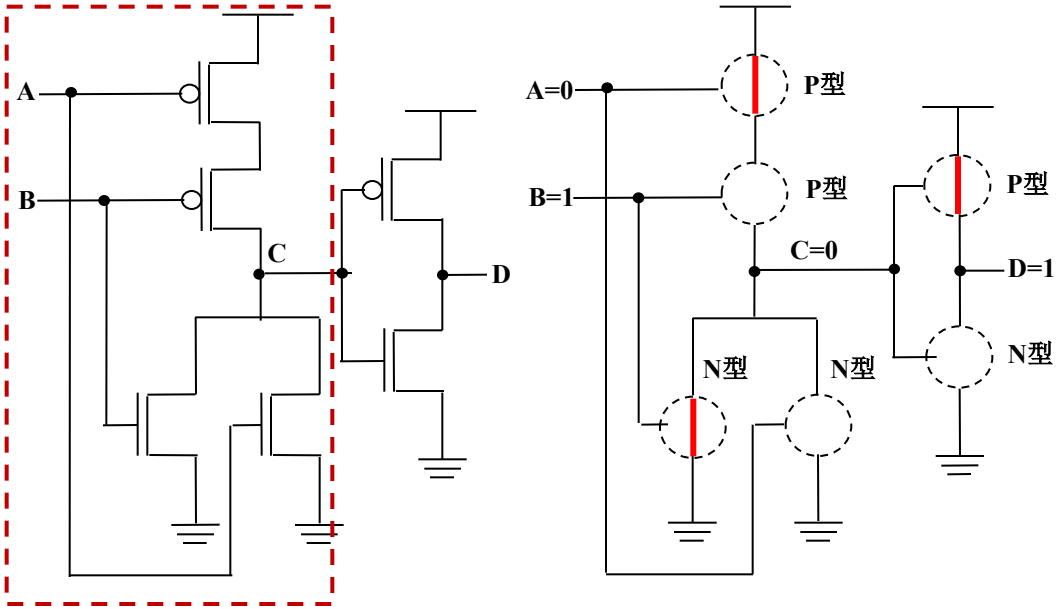


- IEC 60617-12
  - 矩形国标符号



或门

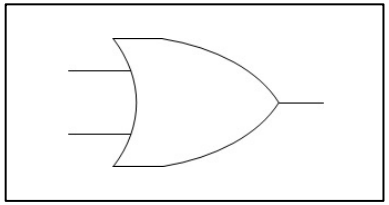
- 在或非门输出端增加一个反相器



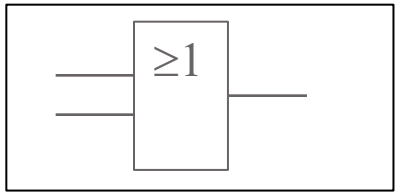
A	B	C	D
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

或门符号表示

- ANSI/IEEE Std 91-1984
  - 形状特征型符号

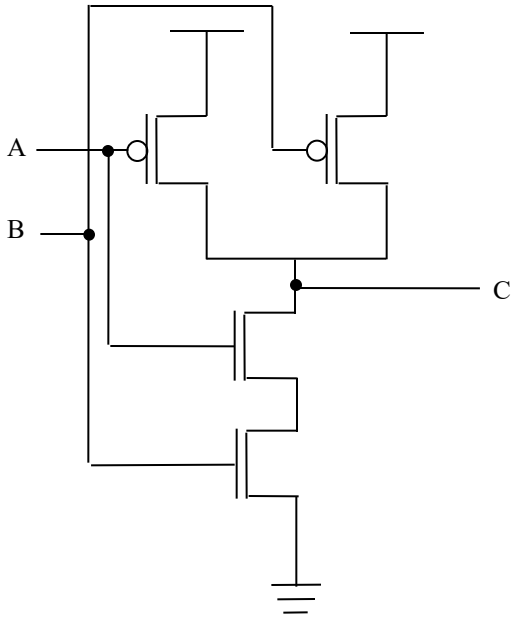


- IEC 60617-12
  - 矩形国标符号



非与门

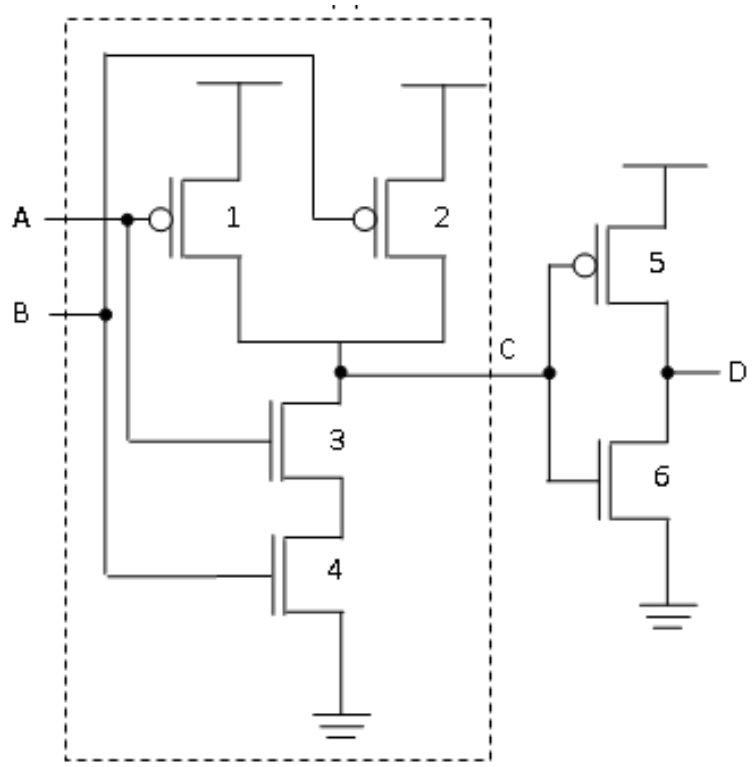
- 顶部并联，底部串联.



A	B	C
0	0	*1*
0	1	*1*
1	0	*1*
1	1	*0*

与门

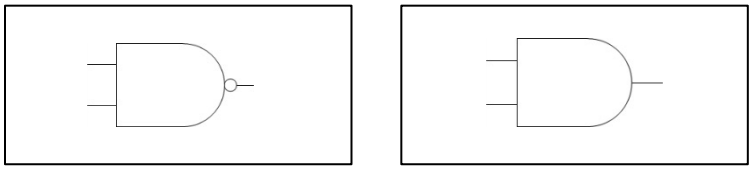
- 在非与门后增加反相器.



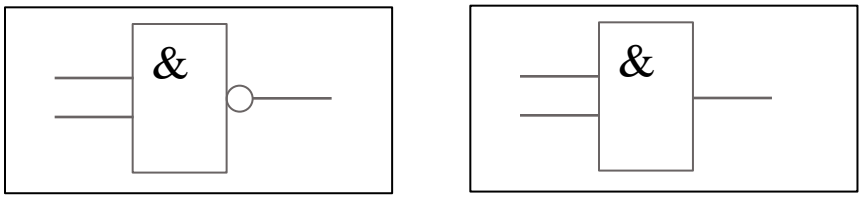
A	B	C	D
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

与非门/与门符号表示

- ANSI/IEEE Std 91-1984
  - 形状特征型符号

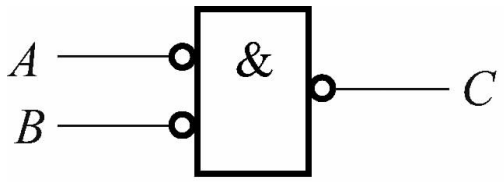


- IEC 60617-12
  - 矩形国标符号

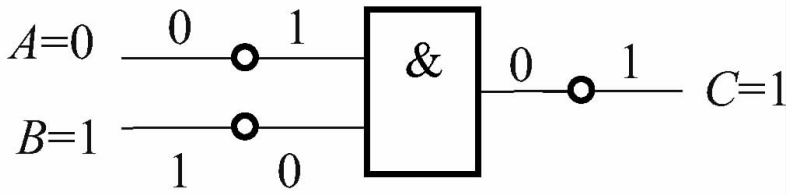


德摩根定律

- A OR B



(a)



(b)

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$	$\overline{\overline{A} \cdot \overline{B}}$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

德摩根定律

- NOT (A OR B) = (NOT A) AND (NOT B)

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

- NOT (A AND B) = (NOT A) OR (NOT B)

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

两个以上输入的门

- 有N个输入的与门
  - 仅当所有的输入变量都为1时，输出才为1；只要有一个输入为0结果就为0
- 有N个输入的或门
  - 只要任意一个输入变量为1输出就为1；也就是说，仅当所有的输入变量都为0时输出才为0

3个输入的与门——符号表示

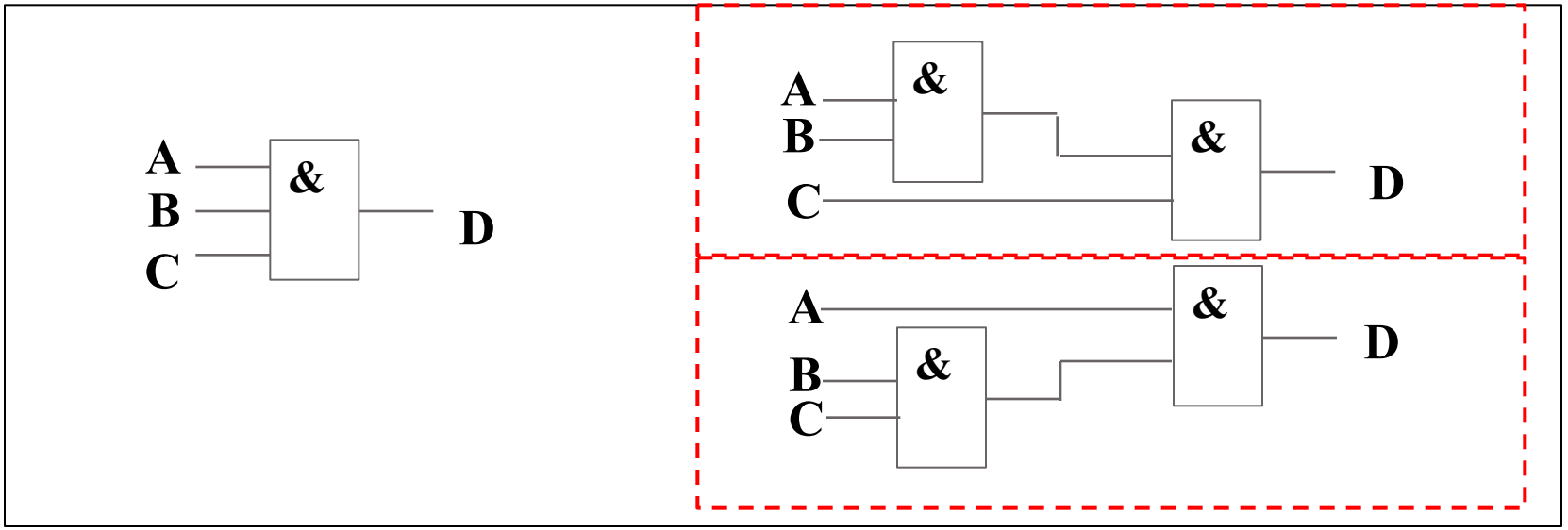
3个输入的与门——真值表

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

思考

- 1、一个有三个输入的与门的晶体管级电路如何构建？一个有四个输入的或门的晶体管级电路如何构建？
- 2、如果现有若干与门（有2个输入），如何构建一个有三个输入的与门？

3个输入的与门



习题（二）

- 书面作业
- 7. 1
  - 1）
  - 2）
    - II
- 7. 2
- 7. 3

组合逻辑电路

逻辑结构

- 连接门电路，构建逻辑结构
  - 实现信息的运算和存储
- 两种基本类型
  - 能够存储信息
  - 不能存储信息

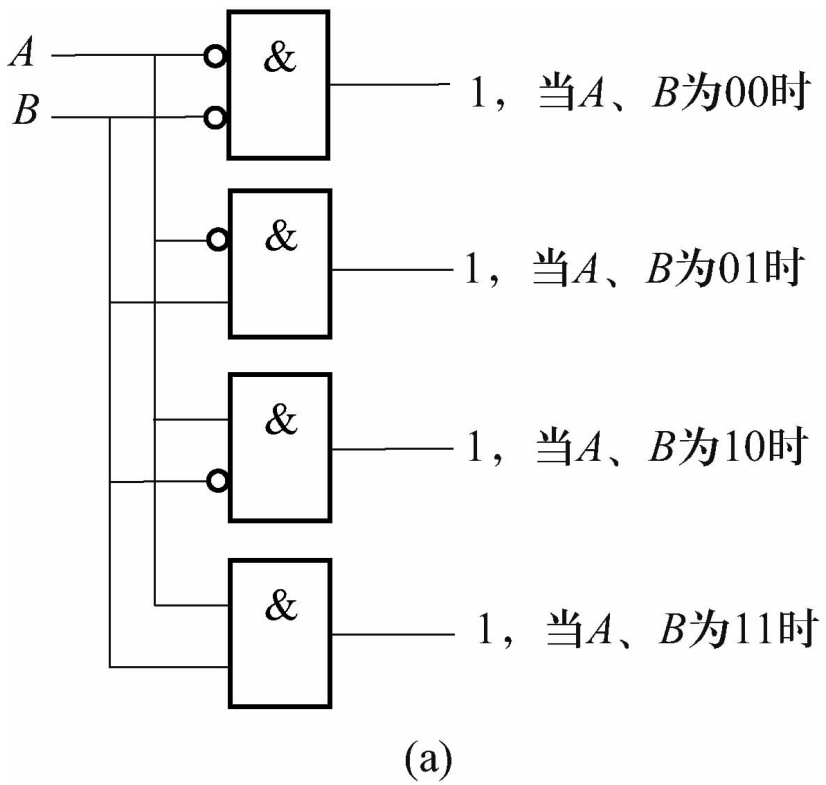
组合逻辑电路

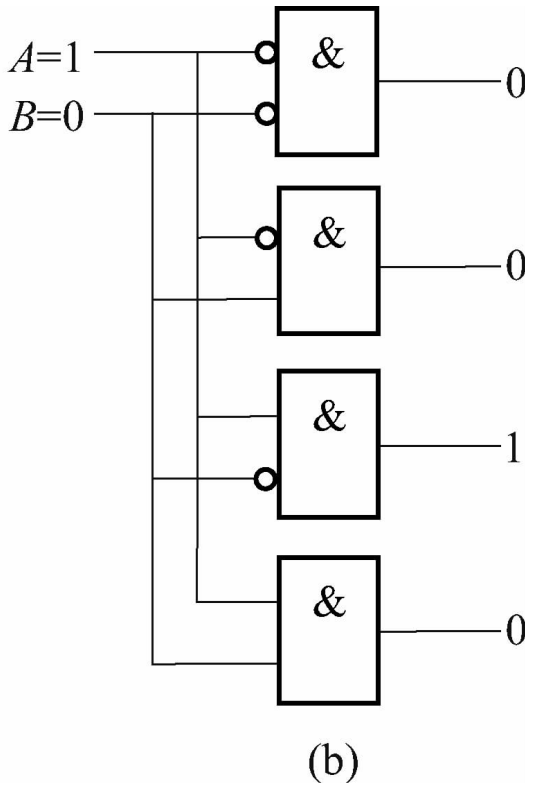
- 不能存储信息的逻辑结构，“判定元件”，组合逻辑结构
  - 它们的输出仅由当前输入值的组合决定，不由任何过去存储在其中的信息所决定
  - 信息不能存储在组合逻辑电路中
- 组合逻辑结构主要用于处理信息
  - 译码器，多路选择器，全加法器等

译码器

- 只有一个输出为1，其他全为0
- 输出为逻辑1的是对应于要被检测的输入组合
- 通常，译码器有n个输入，2^n个输出
- 被检测的输入组合的输出为1，所有其他的输出则为0

译码器n=2

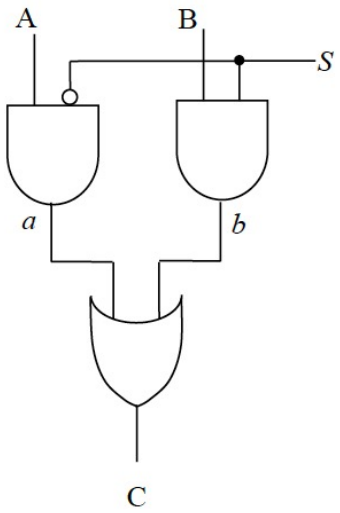




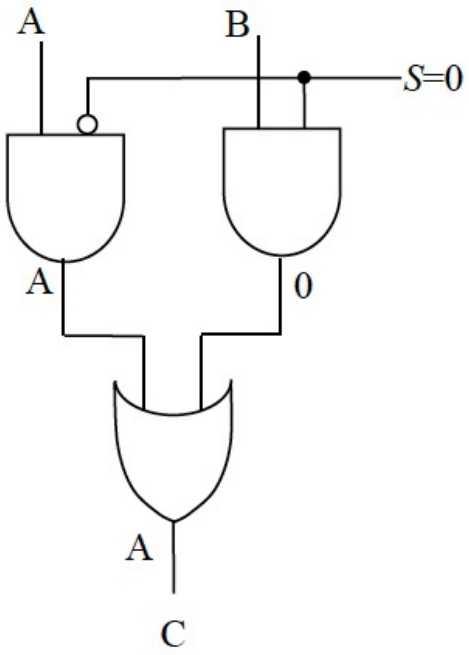
- 2个输出的译码器门级电路
- 在输入A和B的四种可能的组合中，在任意时刻，只有一个输出为1

多路选择器

- 多路选择器的功能就是选择一个输入连接到输出
- 由选择信号决定由哪个输入连接到输出
- 一般说来，一个多路选择器由*n*条选择线和2^n个输入组成

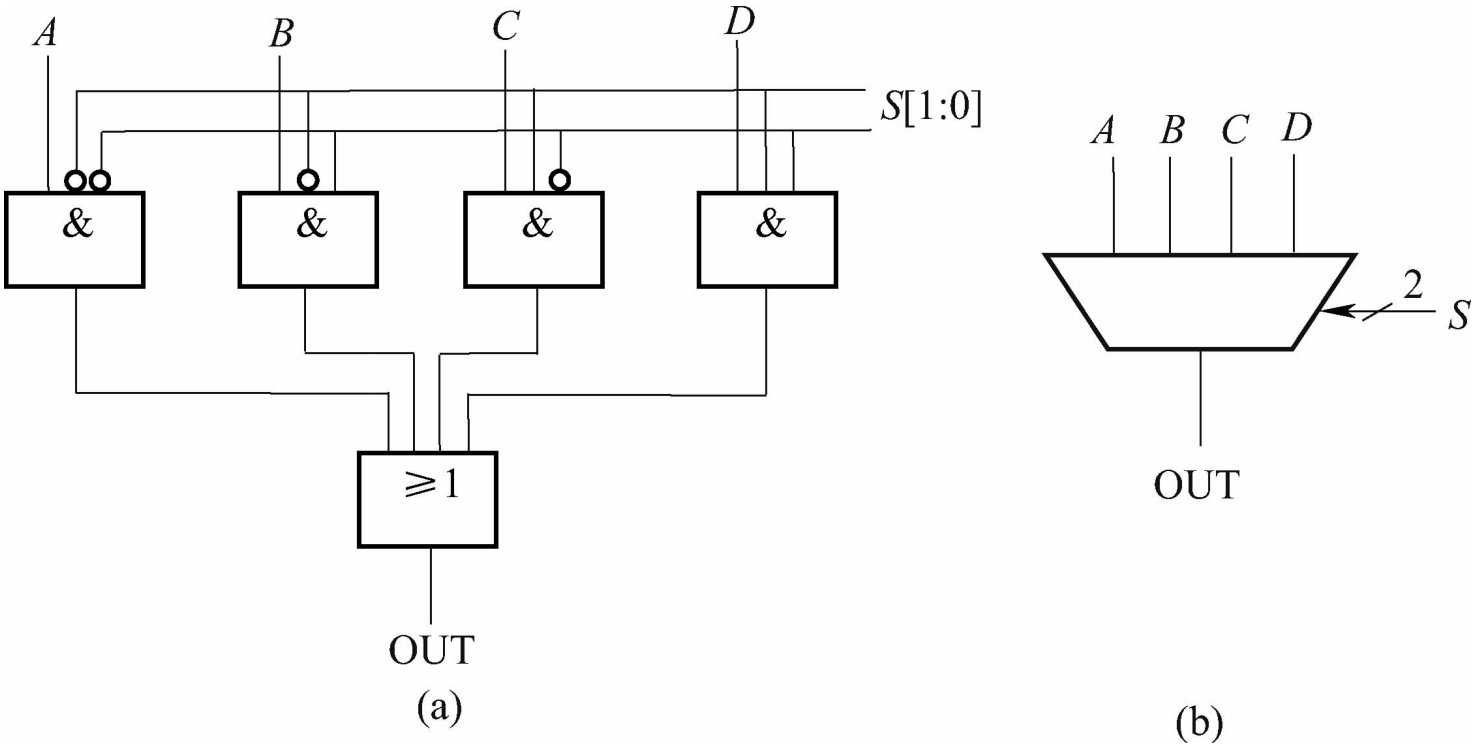


S=0



多路选择器n=2

- 4-1 选择器:
- 取决于S的值（00, 01, 10, 11），输出为A, B, C或D的值

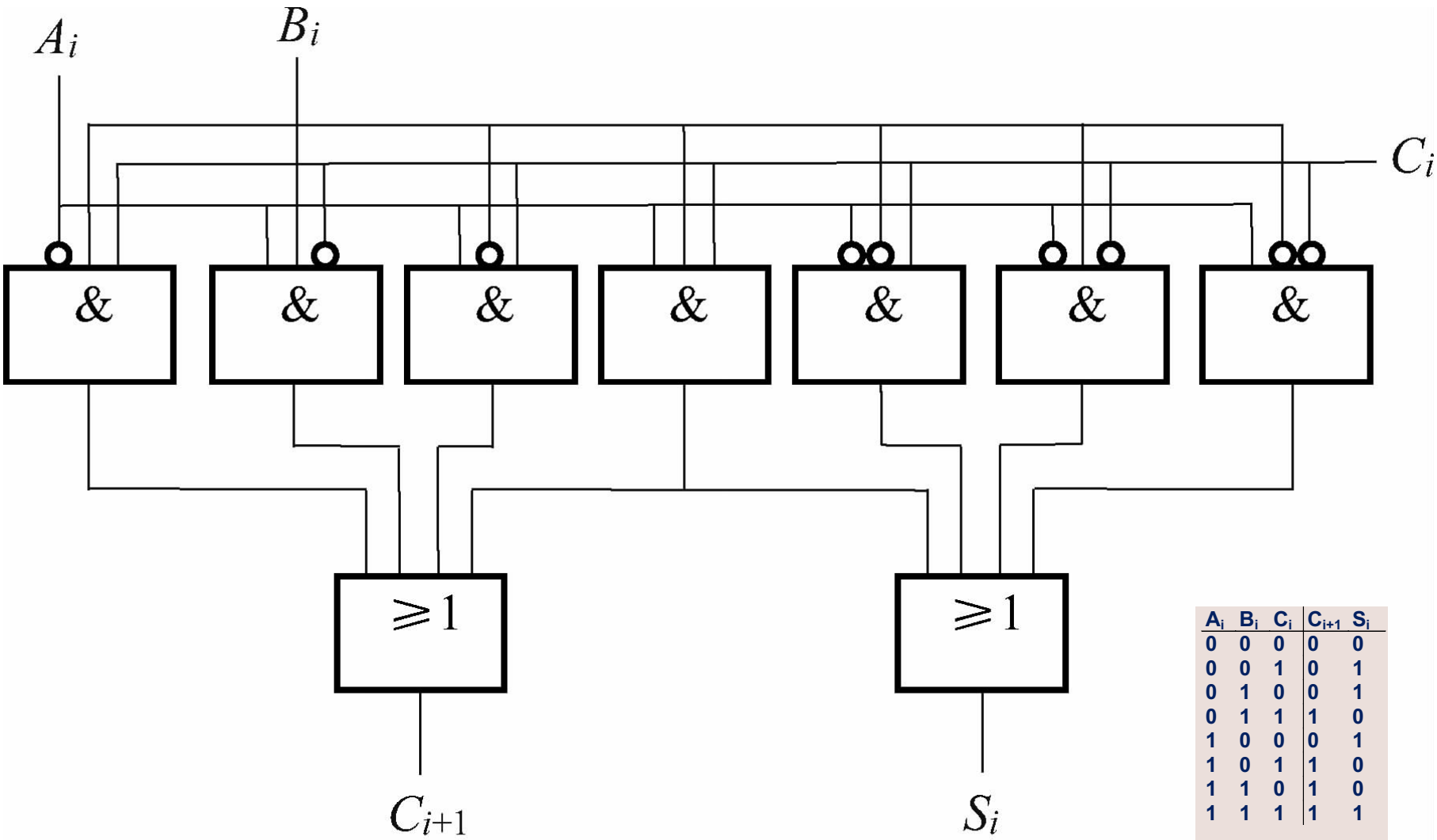


二进制加法

- 真值表
- 两个*n*位操作数的某一列进行二进制加法

A_i	B_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

全加法电路

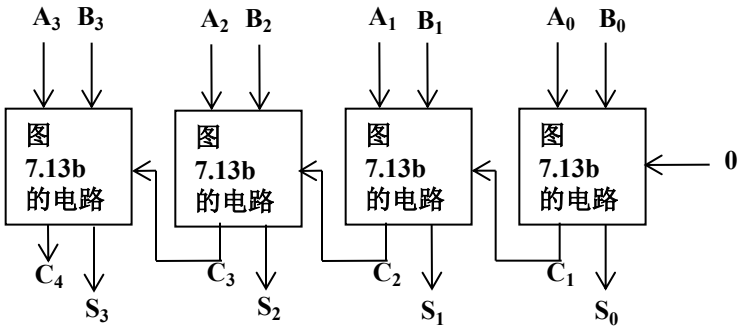


A_i	B_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

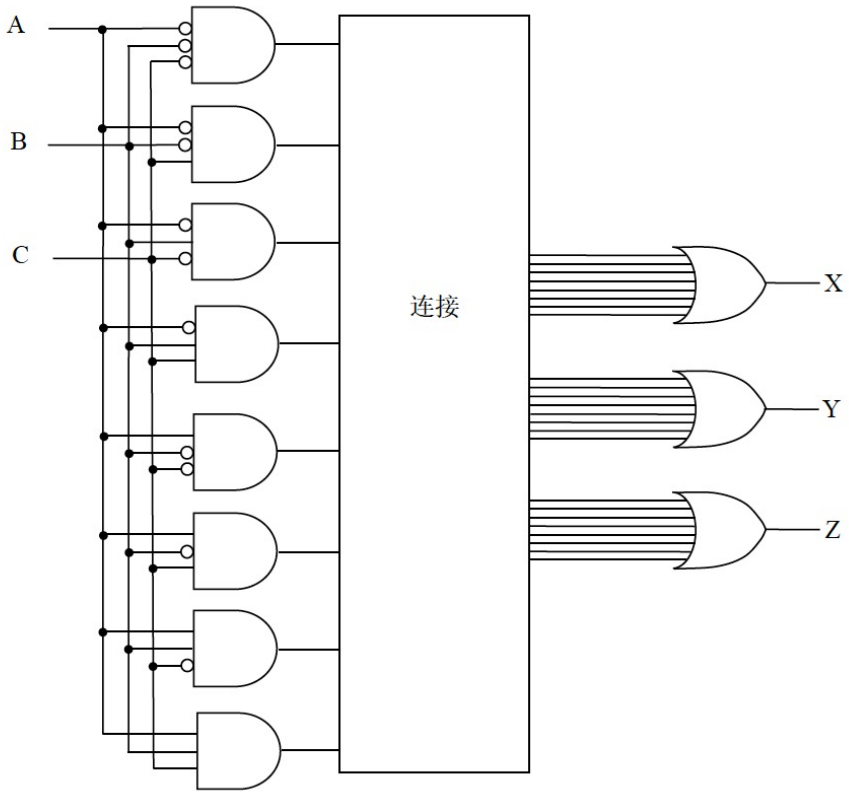
真值表——门级电路

- 对于真值表中A_i，B_i和C_i的7种输入组合之一（除000组合之外7种组合），都有一个与门产生输出1
- 当输入是使真值表中C_{i+1}为1的相应组合时，或门的输出C_{i+1}必为1
- 产生输出S_i的或门的输入，就是真值表中S_i为1的输入组合，经过与门所产生的输出
- 输入组合000对于S_i或C_{i+1}都不会产生值为1的输出，它对应的与门就不是或门的输入

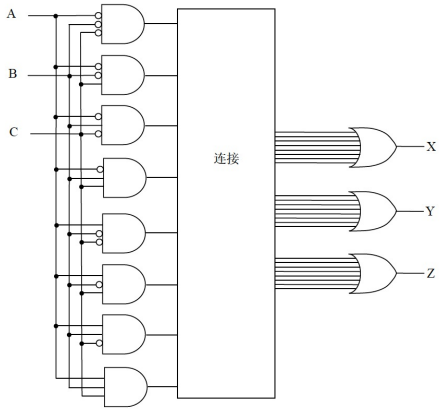
两个4位二进制数的加法电路



可编程逻辑阵列



可编程逻辑阵列



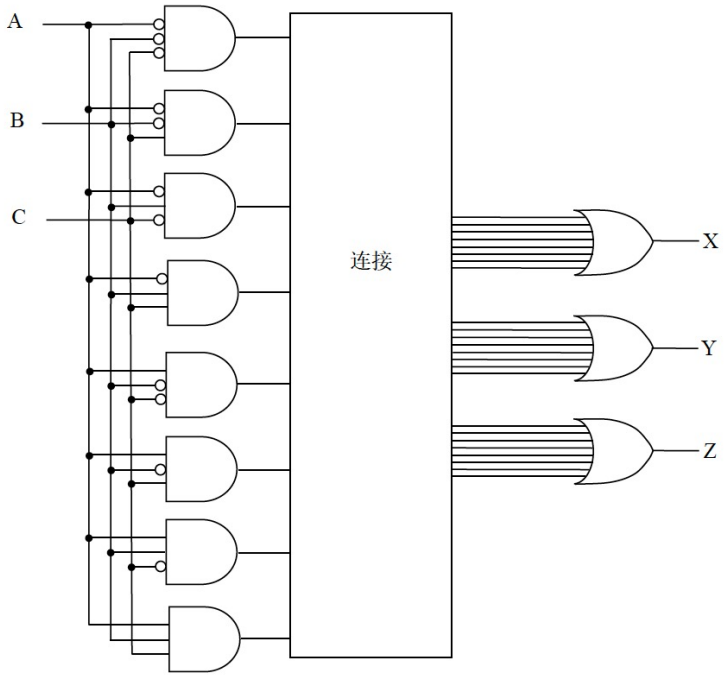
- 可编程逻辑阵列（Programmable Logic Array，PLA）
- 可以实现任意逻辑函数的通用组件
- 由一组与门（被称为与阵列），以及其后的一组或门（被称为或阵列）组成
- 与门的数目对应于真值表中输入组合（行）的数目，对于有n个输入的逻辑函数，PLA将包括2^n个与门，每个与门有n个输入
- 或门的数目对应于真值表中输出的列数

可编程

- 可编程：对于真值表的输出列中产生输出为1的对应的行，只需将PLA中的该与门的输出与或门的输入相连，就可以实现该真值表
- 通过对与门的输出与或门的输入连接进行编程，来实现希望实现的逻辑函数

可编程逻辑阵列n=3

- 如果用A、B、C分别表示a_i，b_i和$carry_i$，用X表示S_i，用Y表示C_{i+1}，经过适当的连接，可以得到全加法器电路



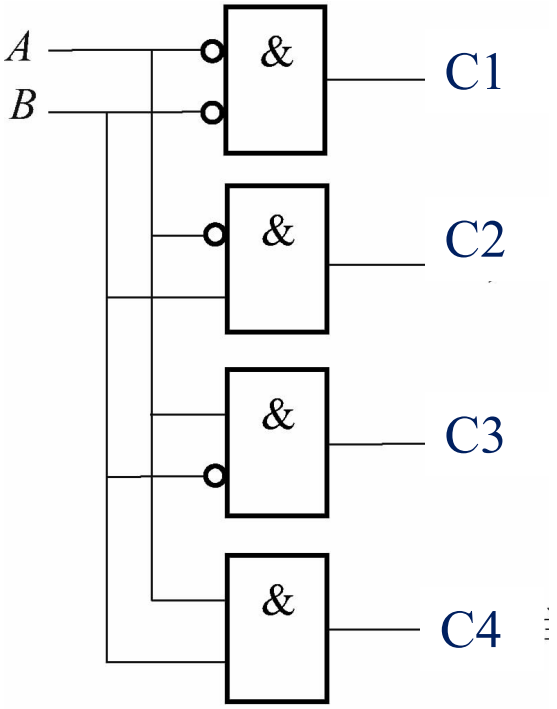
逻辑完备性

- 逻辑完备性（logical completeness）
- 任意逻辑函数都可以通过一个PLA来实现，而PLA只由与门、或门和非门组成
- 对于任意逻辑函数，只要提供足够多的与门、或门、非门，就可以实现
- 门集合{与、或、非}在逻辑上是完备的
- 不需要使用任何其他种类的门就可以实现任何一个真值表的电路

译码器

- n=2

A	B	C1	C2	C3	C4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



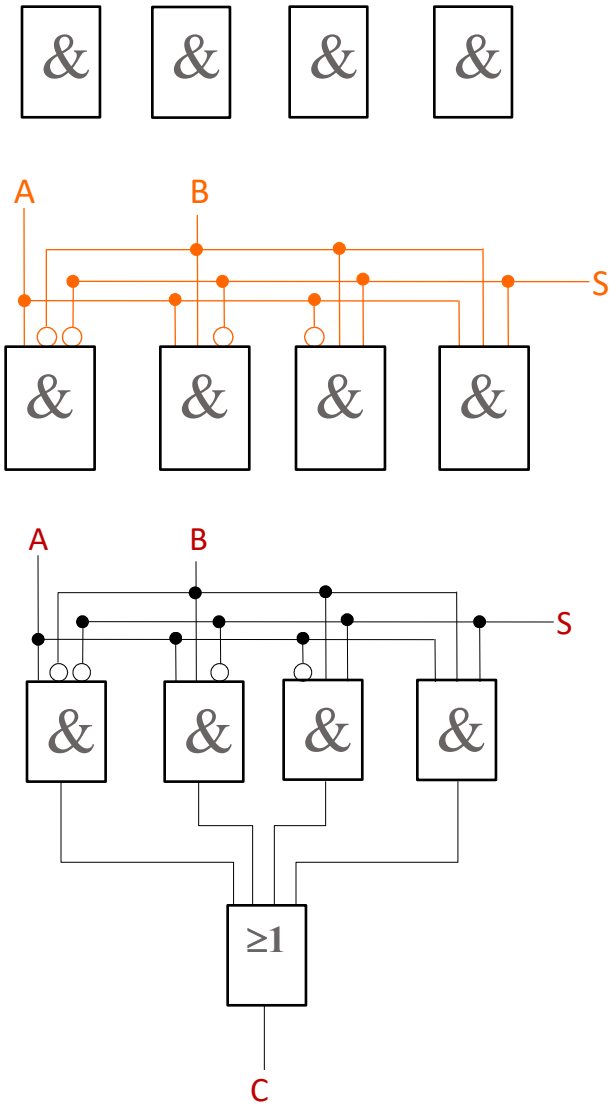
实践

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

多路选择器

- n=1

A	B	S	C
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1



布尔代数与逻辑电路

- 使用PLA实现逻辑函数，不需要任何其他种类的门就可以实现，但是门的数目可能很大
- 借助一些布尔代数定律和德摩根定律进行逻辑等式变换，有时可减少门的数目

布尔代数定律

- 与普通代数类似，布尔代数也有交换律、分配律、结合律等定律
  - 恒等定律：A+0=A，A•1=A
  - 0/1定律：A+1=1，A•0=0
  - 互补律：$A+\overline{A}=1, A\bullet\overline{A}=0$
  - 交换律：A+B=B+A，A•B= B•A
  - 结合律：A+(B+C)=(A+B)+C，A•(B•C)= (A•B)•C
  - 分配律：A• (B+C)= (A•B)+(A•C)，
    A+(B•C)=(A+B) •(A+C)

多路选择器n=1

A	B	S	C
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

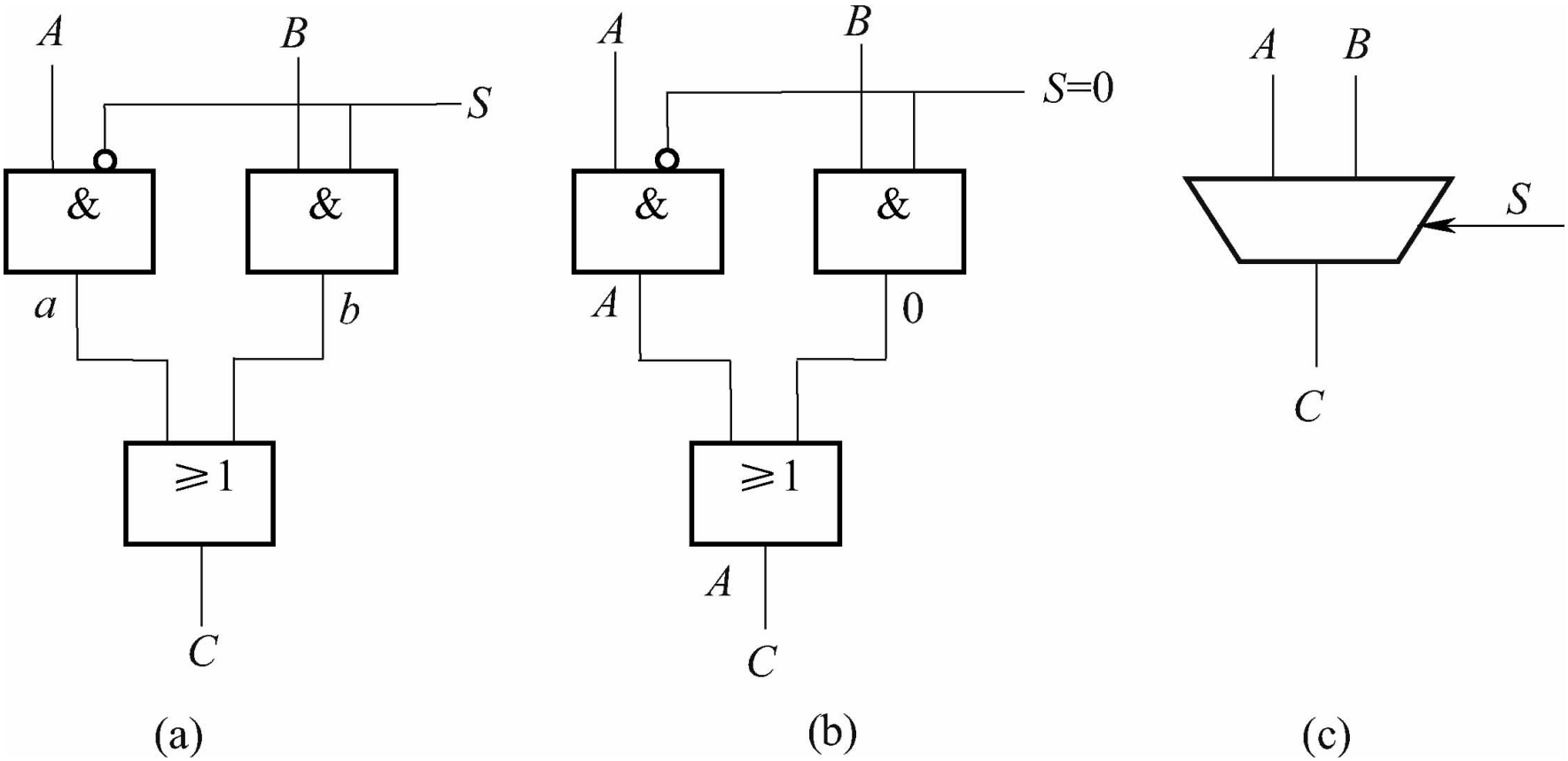
- PLA实现对应的布尔代数描述为：

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

$$\begin{aligned}C&=(\overline{A}\cdot B\cdot S)+(A\cdot\overline{B}\cdot\overline{S})+(A\cdot B\cdot\overline{S})+(A\cdot B\cdot S)\\&=(\overline{A}\cdot B\cdot S+A\cdot B\cdot S)+(A\cdot\overline{B}\cdot\overline{S}+A\cdot B\cdot\overline{S})\\&=((\overline{A}+A)\cdot B\cdot S)+((\overline{B}+B)\cdot A\cdot\overline{S})\\&=(1\cdot B\cdot S)+(1\cdot A\cdot\overline{S})\\&=(B\cdot S)+(A\cdot\overline{S})\end{aligned}$$

只需要2个与门和一个或门！

多路选择器n=1



逻辑函数表示

- 逻辑函数可以被表示为
  - 真值表
  - 逻辑表达式
  - 逻辑电路

习题（三）

- 书面作业
- 7. 4
- 7. 5
- 7. 7
- 7. 8
- 7. 9
- 7. 10
- 7. 11
- 7. 12

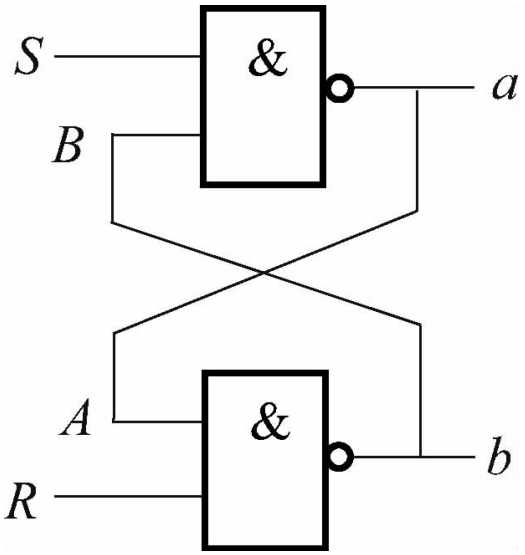
基本存储元件

基本存储元件

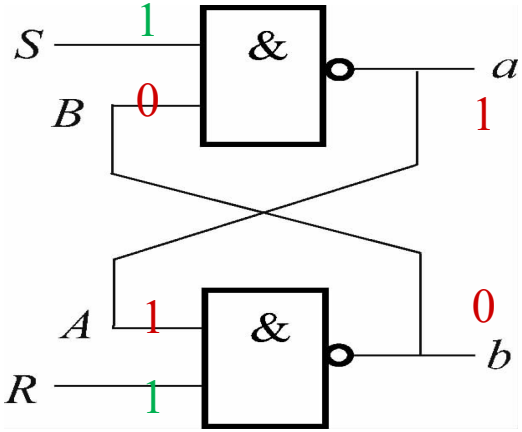
- 译码器、多路选择器和全加法器：不能存储信息的逻辑结构
- 存储1位信息的基本存储元件
  - 锁存器

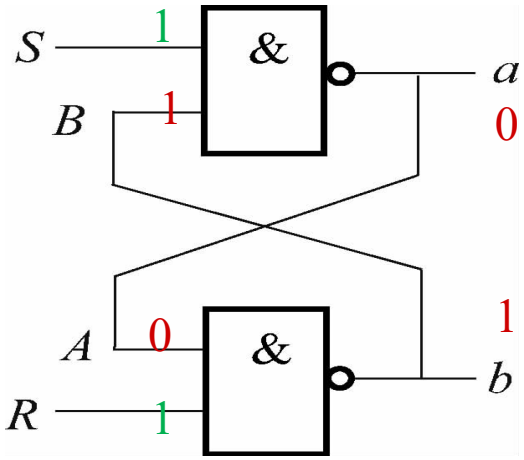
R-S锁存器

- 工作原理：“保持/静止状态”，S=R=1
  - R：“reset”，复位
  - S：“set”，置位
- 非与门也可以用非或门来代替



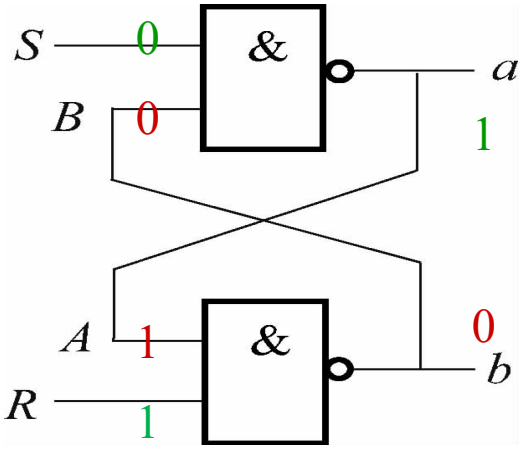
“保持”状态





“置位”状态

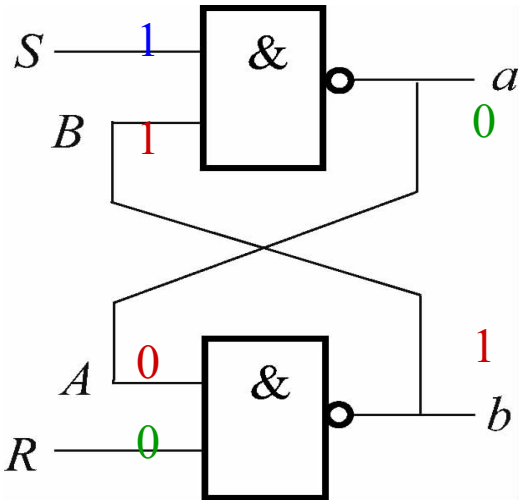
- S=0， R=1



- S=1， R=1，a的值不变——存储1

“复位”状态

- S=1， R=0



- S=1， R=1，a的值不变——存储0

R-S锁存器

- R = S = 1
  - “保持”状态
- S = 0, R=1
  - “置位”状态
- R = 0, S = 1
  - “复位”状态

R-S锁存器

- R = S = 0 ×
  - 锁的最后状态取决于组成门的晶体管的电子特性而不是取决于被操作的逻辑值
  - 是逻辑电路“延迟”造成的

真值表

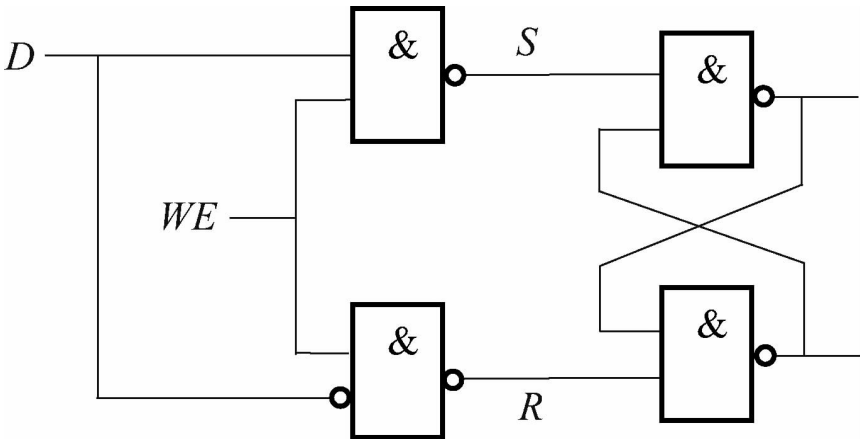
S	R	a	b
0	0	不定	
0	1	1	0
1	0	0	1
1	1	保持	

门控D锁存器

- 控制
  - 何时置位、何时复位
  - S和R不同时为0

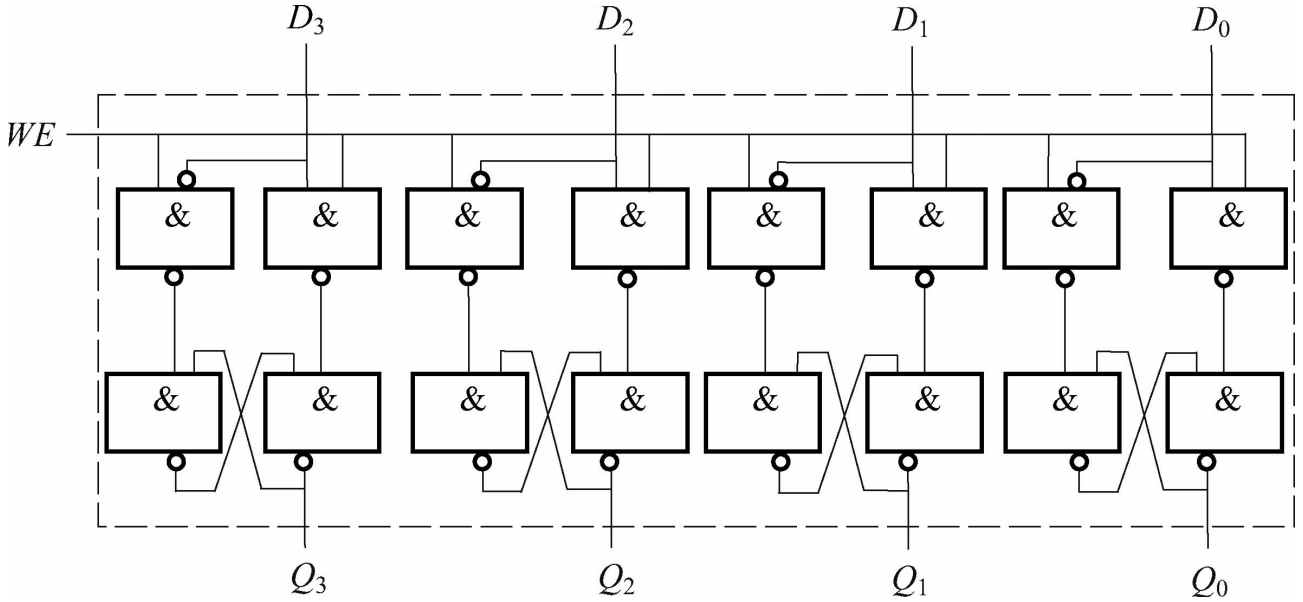
门控D锁存器

- 两个输入：D（data）和 WE（write enable）
  - WE = 1，输出 = D
    - S = NOT(D)，R = D
  - WE = 0，存储D的值
    - S = R = 1



寄存器

- 将多位数据存储于一个独立单元的结构
- 使用一组门控D锁存器，**WE共享**
  - WE=1，n位D的值被写入寄存器



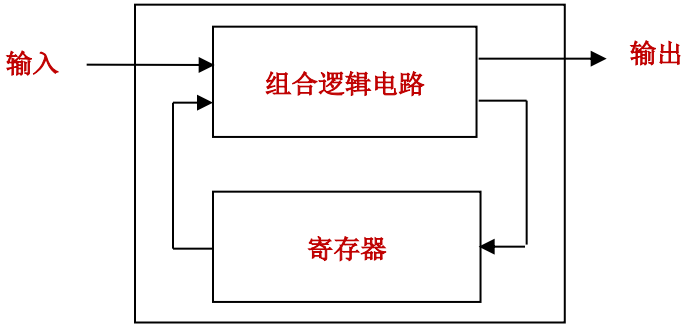
习题（四）

- 书面作业
- 7. 13

时序逻辑电路

时序逻辑电路

- 既能处理信息，也能存储信息
- 不只根据现在的输入，而且基于之前发生的事做判定

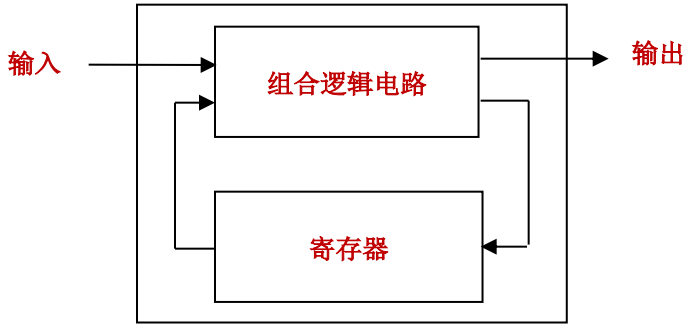


时序逻辑电路

- 可用来实现一种非常重要的被称为有限状态机的机制
  - 有限状态机可被用作电子系统、机械系统、航空系统等的控制器

时序逻辑电路简图

- 输出既取决于当前的输入，也取决于存储在存储元件中的值，而存储在寄存器中的值则反映了之前发生的历史情况



电话应答机

- 可以根据响铃的次数（如3次），决定是否开启录音机录音
- 电话应答机的输出（是否开启录音机）不仅仅取决于当前的输入（是否响铃），还取决于这次输入（响铃）之前的一系列输入（已经响过2次铃）——时序逻辑结构

状态的概念

- 一个系统的状态，是在某一特定时刻，系统内所有相关部分的一个瞬态图
- 电话应答机的4个状态：
  - A．不开启录音机，还未响铃；
  - B．不开启录音机，但已响铃1次；
  - C．不开启录音机，但已响铃2次；
  - D．开启录音机。
- 这4种情况分别被标记为A、B、C和D，每一种情况都被称为应答机的一种状态

计算“int”字符串出现次数

- 该问题可以使用状态描述如下：
  - 0、计数器不变，还未遇到“i”；
  - 1、计数器不变，但已遇到“i”；
  - 2、计数器不变，但已遇到“in”；
  - 3、计数器加1。
- 共有四种可能的状态

有限状态机

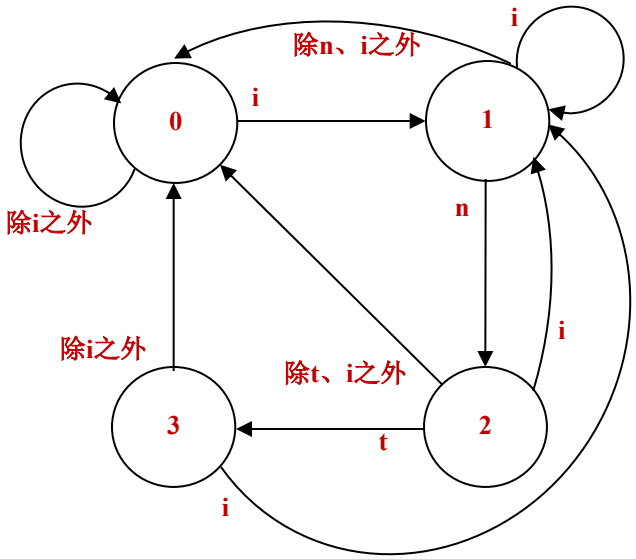
- 寄存器容量是有限的，所以状态的数目必须是有限的
- 通常，使用有限状态机来描述系统的行为
- 有限状态机由5个元素组成：
  - 有限数目的状态；
  - 有限数目的外部输入；
  - 有限数目的外部输出；
  - 明确定义的状态转换函数；
  - 明确定义的外部输出函数。

状态图

- 有限状态机可以通过被方便的表示出来
- 一组圆（每一个圆对应于一个状态），和一些状态之间的一组连接弧线（每条连接弧线被画为一个箭头）
  - 每一条弧线确定一个状态的转换
  - 每条弧线的箭头说明系统从哪一个状态来，要到哪一个状态去
  - 把来的状态称为当前状态，要去的状态称为下一个状态

计算“int”字符串出现次数

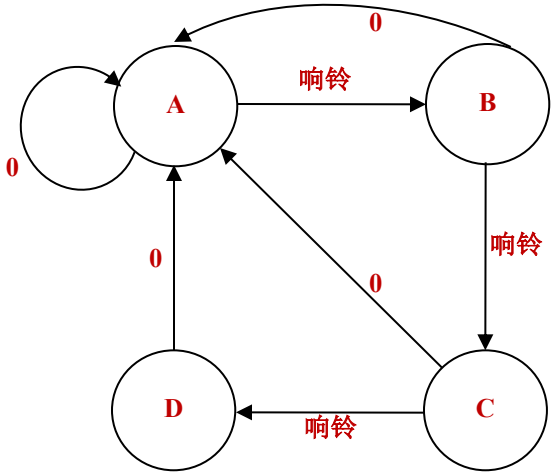
- 由4个状态组成，有10个状态转换
  - 外部输入是读到的字符
  - 下一个状态是由当前状态和当前的外部输入的组合决定的
  - 系统的输出为计数器的值：当系统的状态是0、1和2时，计数器不变；当系统的状态是3时，计数器加1



- 通常，对于一个当前状态，存在有多个到下一个状态的转换，发生哪一个状态转换取决于外部输入的值。简而言之，下一个状态是由当前状态和当前的外部输入的组合决定的。
- 系统的输出值仅由系统的当前状态决定，或者由当前状态和当前的外部输入的组合决定。

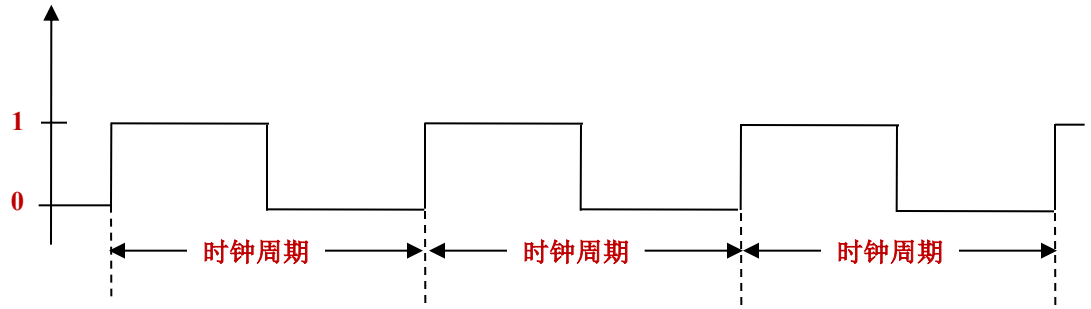
电话应答机

- 外部的输入是响铃，0表示在规定的时间内不再响铃
- 从每一个状态出去的弧线可能有多条，分别表示不同的输入到达的状态
- 输出与每个状态相关，应答机的输出为是否录音，在状态A、B和C，不录音，在状态D，录音



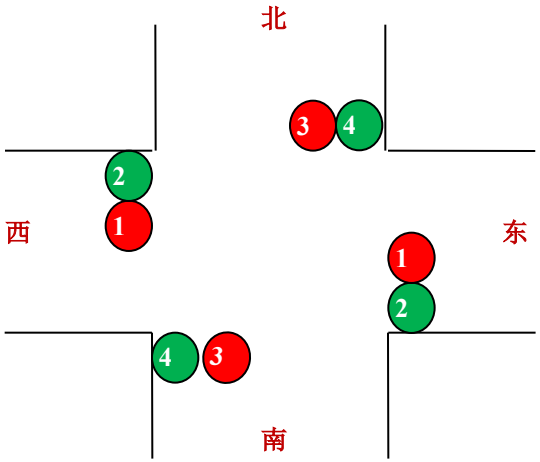
时钟

- 触发状态向下一个状态转换的机制
- 应答机，响铃触发了状态的转换
- 通常，触发状态从一个向下一个转换的机制是时钟电路，时钟
  - 时钟发出的信号值在0伏和某个特殊的固定的电压之间交替
  - 时钟周期是指重复的时间间隔序列中的一个时间间隔



交通灯控制器

- 东西向大街和南北向大街相交的十字路口
- 在东西向和南北向各有一组交通灯
  - 每组灯只包括红灯（1、3）和绿灯（2、4）
- 一组通行按钮，供行人按下，控制东西向和南北向的交通灯

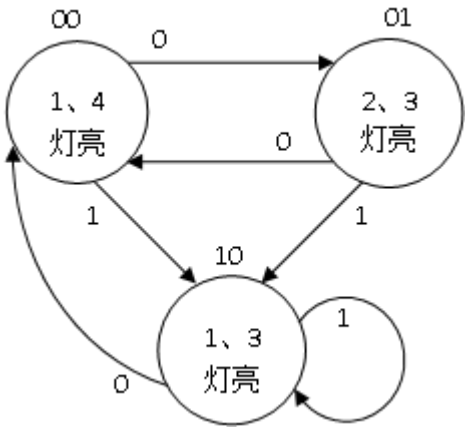


控制器

- 当没有行人时，在第一个时钟周期，1号灯和4号灯亮；下一周期，2号灯和3号灯亮；然后，重复这个顺序。
- 当有行人按下通行按钮时，在当前的时钟周期结束时，1号灯和3号灯亮（东西向和南北向红灯都亮），并保持一个时钟周期，然后，回到1号灯和4号灯亮，继续交替变化。

状态图

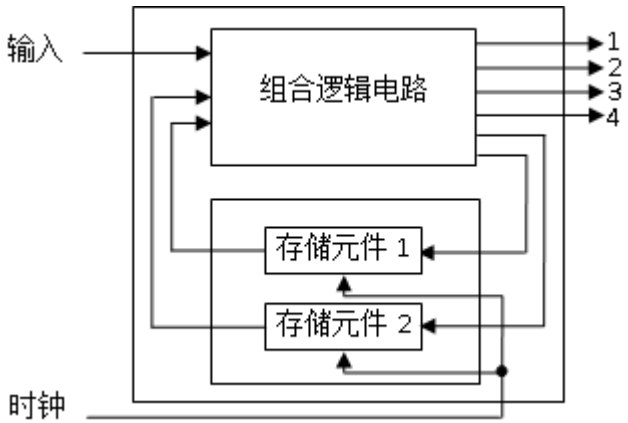
- 共有3个状态：1、4号灯亮，2、3号灯亮以及1、3号灯亮。



- 如果无行人按通行按钮（用0表示），交通灯就按照时钟周期从状态00转换到状态01，再转换到状态00，重复进行；
- 如果有行人按下按钮（用1表示），在当前时钟周期结束，状态总是转换到10，如果无行人按钮，在下一时钟周期，再转换到状态00，如果有行人按钮，则保持在状态10。

时序逻辑电路

- 简化：黄灯，时间，……
- 1个外部输入：行人的按钮行为
- 4个外部输出：分别用于控制1、2、3和4号灯何时亮
- 寄存器：2个存储元件，记录控制器处于哪一个状态（00，01，10），由交通信号的过去的行为决定的（内部输出）
- 时钟：使状态转换每隔0.5分钟即可发生



组合逻辑电路——输出和下一状态

- 系统输出
  - 用于控制灯的一组外部输出：有4个输出（标记为1、2、3和4）用来控制4盏灯的亮与灭
  - 由当前状态决定
- 下一状态
  - 用于寄存器输入的一组内部输出：有2个（标记为下一状态[1]和[0]），也就是下一状态
  - 由当前状态和当前的外部输入的组合决定

外部输出

- 输出表
  - 控制灯的行为
  - 由当前状态决定

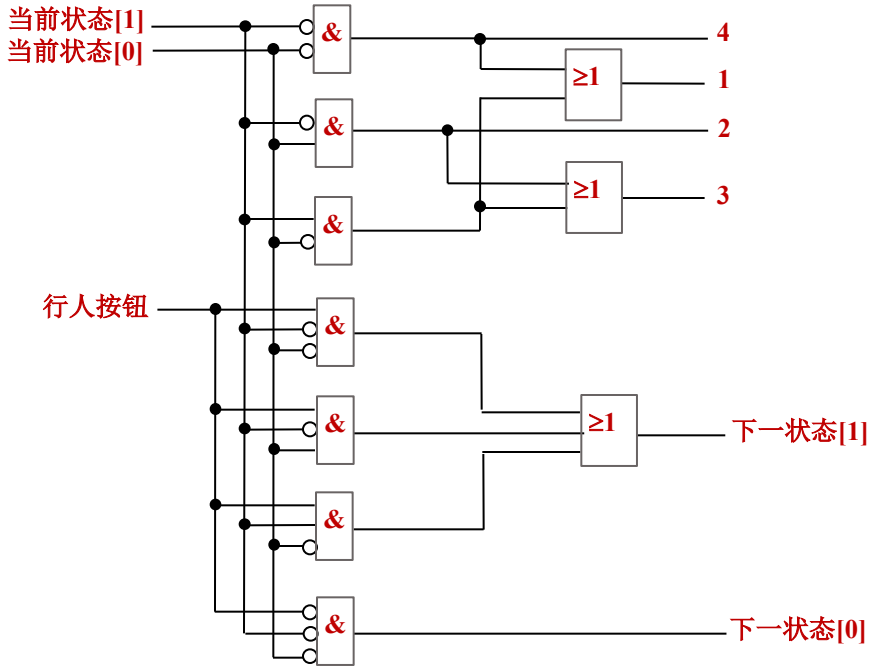
当前状态[1]	当前状态[0]	1灯	2灯	3灯	4灯
0	0	1	0	0	1
0	1	0	1	1	0
1	0	1	0	1	0

内部输出

- 2个：标记为下一状态[1]和[0]
- 下一状态/次态表
  - 由当前状态和当前的外部输入的组合决定

行人按钮	当前状态[1]	当前状态[0]	下一状态[1]	下一状态[0]
0	0	0	0	1
0	0	1	0	0
0	1	0	0	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0

组合逻辑电路

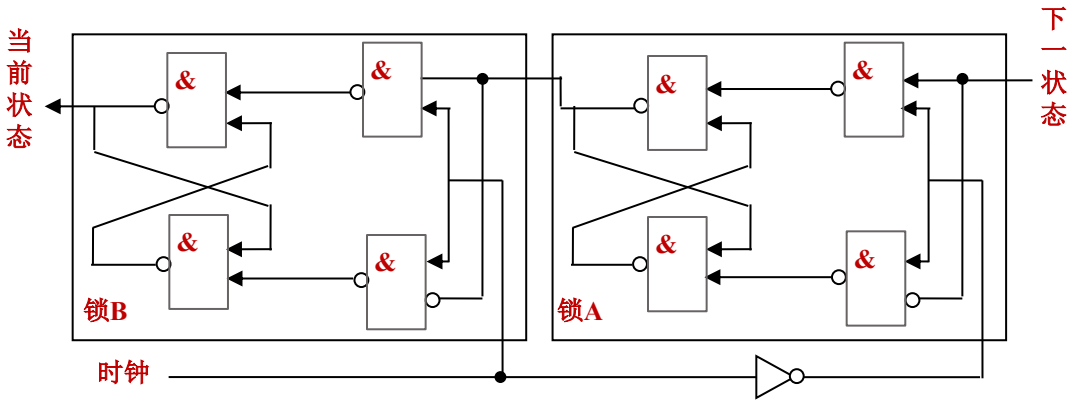


寄存器

- 注意：此处不能使用由门控D锁存器组成
- 原因：
  - 如果使用门控D锁存器，输入会立即发生作用，改写了寄存器中的值，而不是等到下一个周期开始再改写

寄存器

- **主从触发器**，由2个门控D锁存器构成
  - 时钟周期前半段（1），不改变存储在A锁（主锁）中的值，无论A锁中是什么值，都会传给B锁（从锁）
  - 时钟周期后半段（0），不改变存储在B锁中的值，在A锁中的值将被改变



习题（五）

- 上机作业
- 7. 17

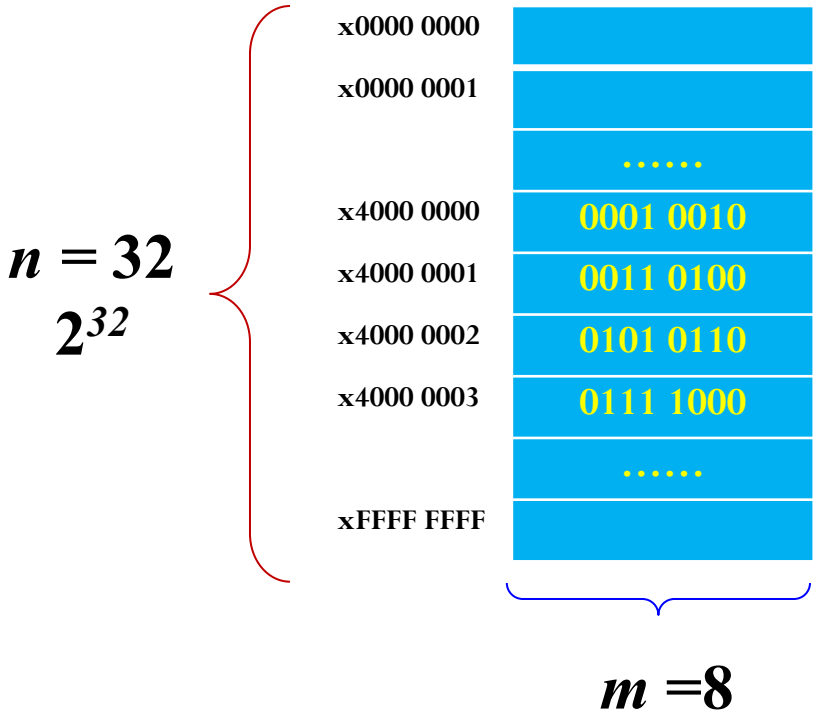
存储器

存储器

- 基本存储元件
  - 锁存器、触发器
  - 存储一位信息
- 寄存器
  - 存储多位信息
- 存储器（Memory），内存
  - 二维阵列，2^n行，每一行存储m位
  - 行：存储单元（Memory Location）

存储器

- 地址
  - 和每一个单元联系在一起的唯一的标识符
  - 二进制地址
- 寻址能力
  - 存储在每一个单元中的信息的位数



地址空间

- 唯一可识别的单元总数
  - 2^{10}，1024，1K
  - 2^{20}，1M
  - 2^{30}，1G
  - 2^{32}，4G，4294967296

寻址能力

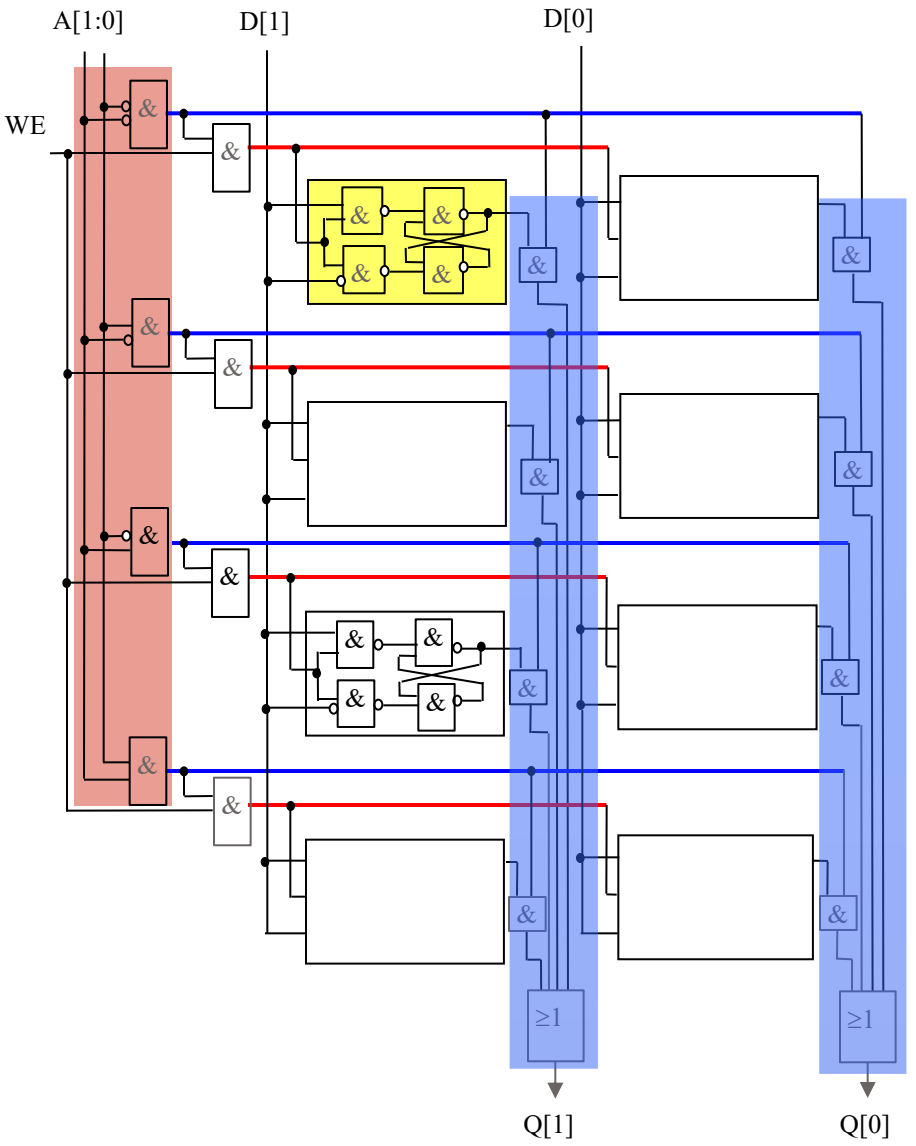
- 存储在每个单元中的位数
- 大多数的存储器，字节可寻址
  - B（byte，字节），表示8位的信息量
  - 原始操作数据，键盘上键入的某个字符
- 64位可寻址
  - 用于科学计算的计算机

存储器容量

- “4GB”
  - 4G，约40亿个存储单元
  - 每个单元包含一个字节的信息

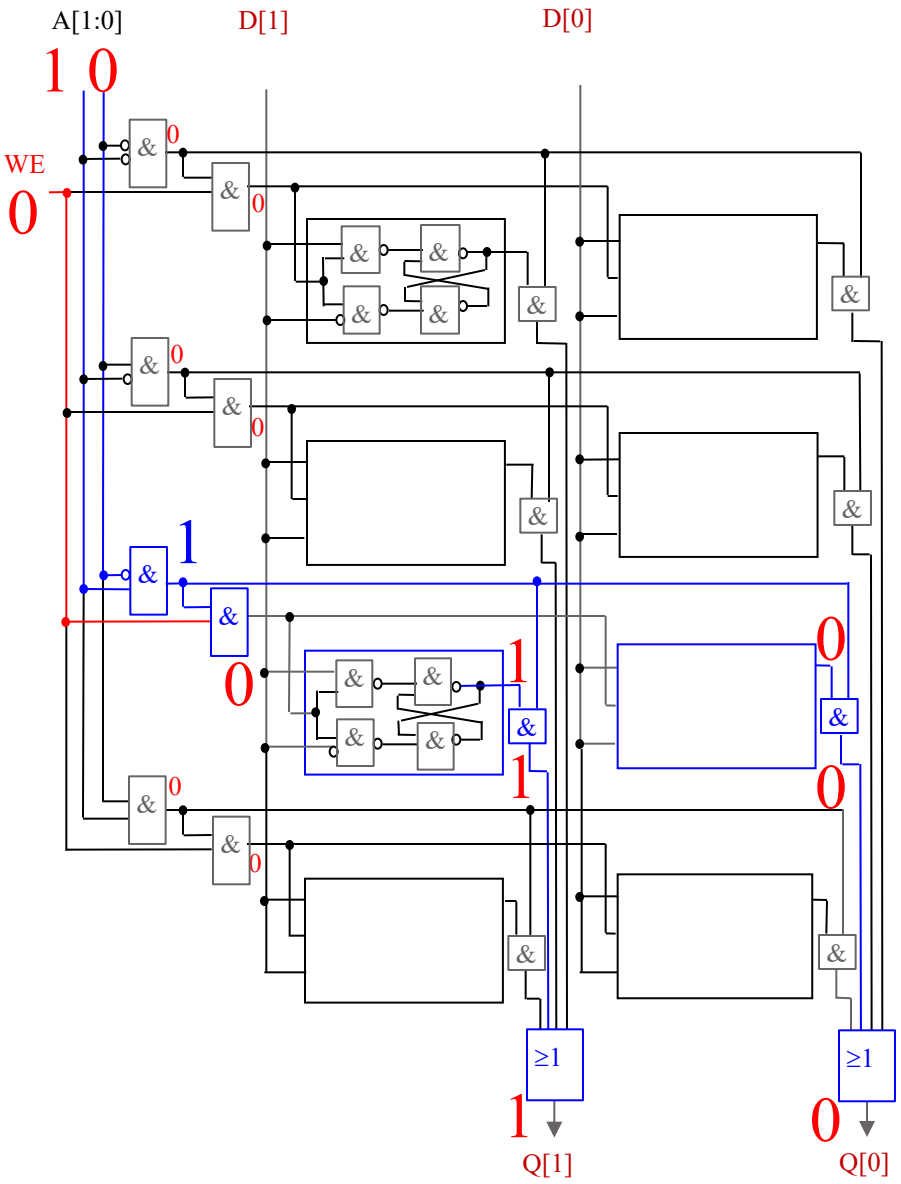
一个4×2的存储器

- 地址：A[1:0]
- 地址译码器（红色阴影）
- 字线（蓝线）
- 字（2位）
  - 每一位由一个存储元（Memory cell）组成
  - 存储元：可以是一个门控D锁存器（黄色阴影）
- 多路选择器（蓝色阴影）
- 输出：Q[1:0]
- 输入：D[1:0]
- WE，字WE（红线）



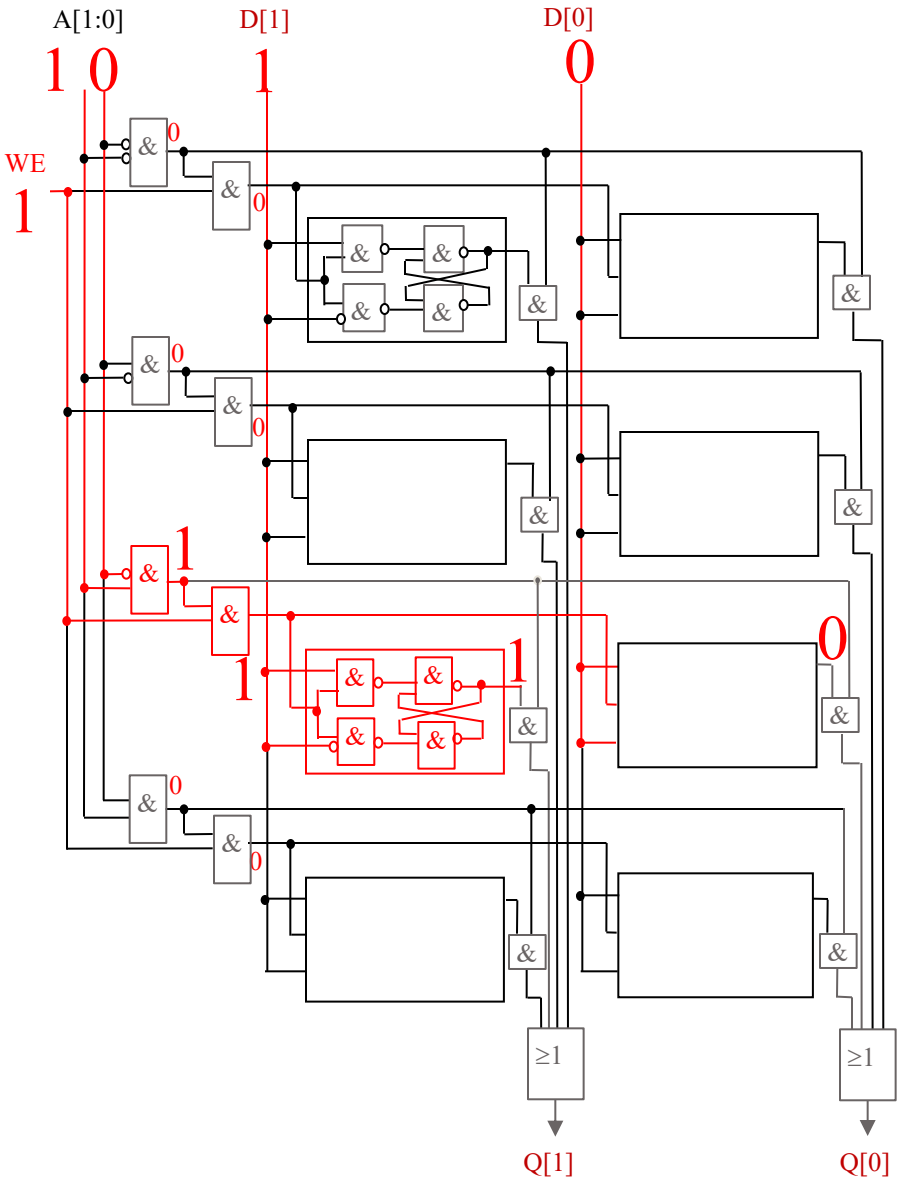
读单元2

- 地址
  - A[1:0]
- WE=0
- 输出
  - Q[1:0]



写/存储

- 地址
  - A[1:0]
- 输入
  - D[1:0]
- WE=1
  - 字WE=1



SRAM

- SRAM（Static Random Access Memory，静态随机访问存储器）
- 结构相对简单
- “静态”：只要给它供电，其内部数据就不会丢失，可以一直保存
- “随机访问”：可以以任意顺序访问，而不必关心前一次访问的是哪一个单元
- 存储元：可以用更少的晶体管实现

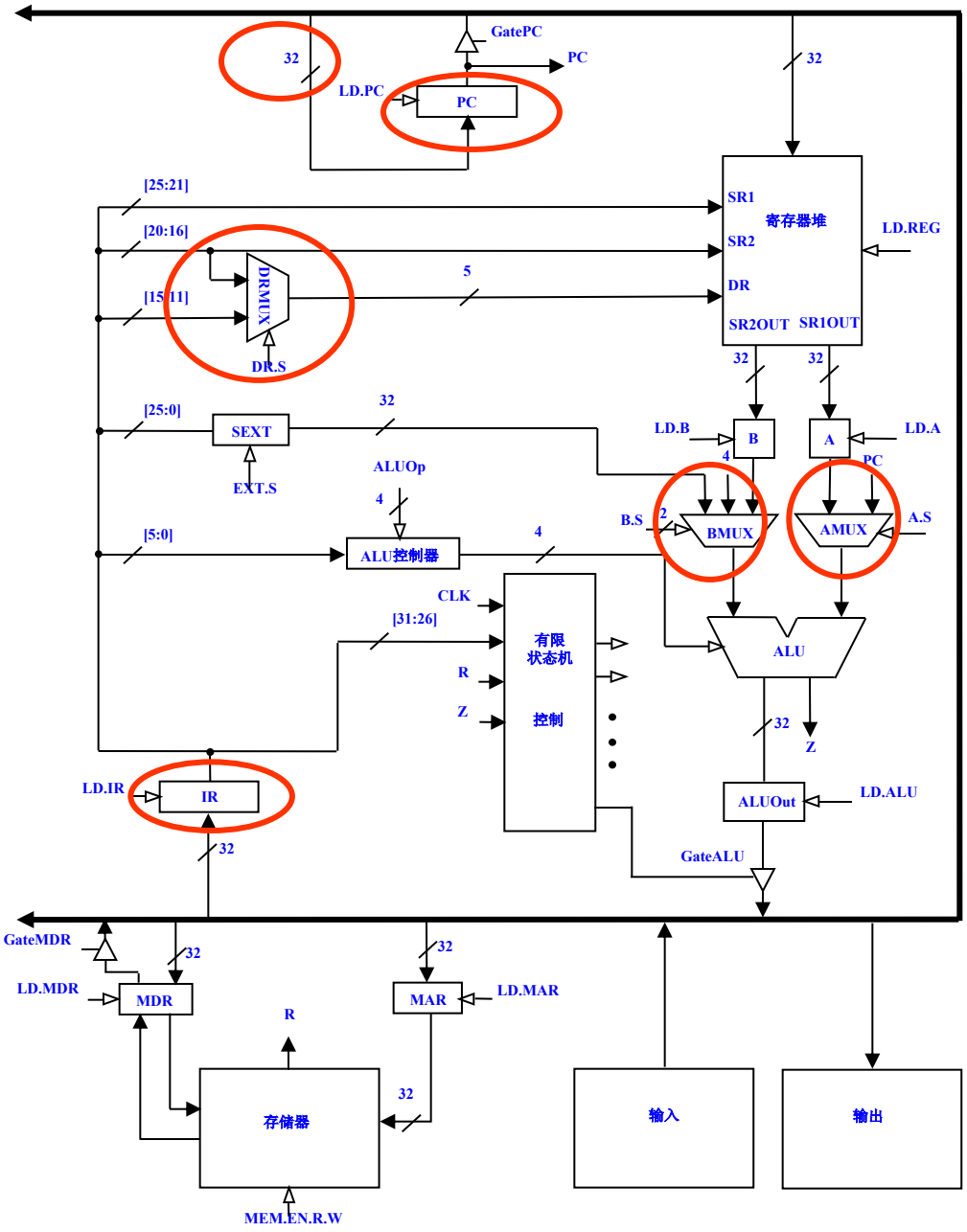
习题（六）

- 书面作业
- 7. 14
- 7. 15
- 7. 16

DLX子集的数据通路

DLX子集的数据通路

- 数据通路
  - 在计算机内部用于处理信息的所有元件的总和
- 寄存器
  - 寄存器堆/文件，程序计数器PC、指令寄存器IR
  - 32位
- 多路选择器
  - DRMUX提供一个5位的地址给寄存器堆
  - AMUX和BMUX分别提供一个32位的数值给ALU
- 每根用交叉斜线标记32的线表示该线内共有32条线，每条用来传送1位的信息


GatePC
PC
32
LD.PC
PC
32
[25:21]
[20:16]
[15:11]
SR1
寄存器堆
SR2
DR
SR2OUT
SR1OUT
LD.REG
DRMUX
DR.S
5
32
32
[25:0]
SEXT
32
LD.B
B
A
LD.A
EXT.S
4
PC
ALUOp
4
B.S
2
BMUX
AMUX
A.S
[5:0]
ALU控制器
4
CLK
[31:26]
有限
状态机
R
Z
控制
ALU
32
Z
LD.IR
IR
ALUOut
LD.ALU
32
GateALU
GateMDR
32
32
LD.MDR
MDR
MAR
LD.MAR
R
存储器
32
输入
输出
MEM.EN.R.W