第七章 数字逻辑电路

本章重点

- 二进制逻辑运算
- 集成电路
- 晶体管
- 门电路
- 组合逻辑电路
- 基本存储元件
- 时序逻辑电路

二进制逻辑运算

- 逻辑运算概念
- 基本运算符
- 按位运算
- C的逻辑运算符

位组合

- 二进制是计算机编码存储和操作信息的核心
- 计算机要解决一个真正的问题，必须能唯一的识别出许多不同的数值，而不仅仅是0和1
- 唯一的识别多个数值，必须对多个位进行组合
- 如果用8位（对应8根线路上的电压）
  - 0100 1110，1110 0111……
  - 最多能区分出256（即2^8）个不同的值
- 如果有*k*位，最多能区分出2^k个不同的值
  - 每一种组合都是一个编码，对应着某个特定的值

位组合运算

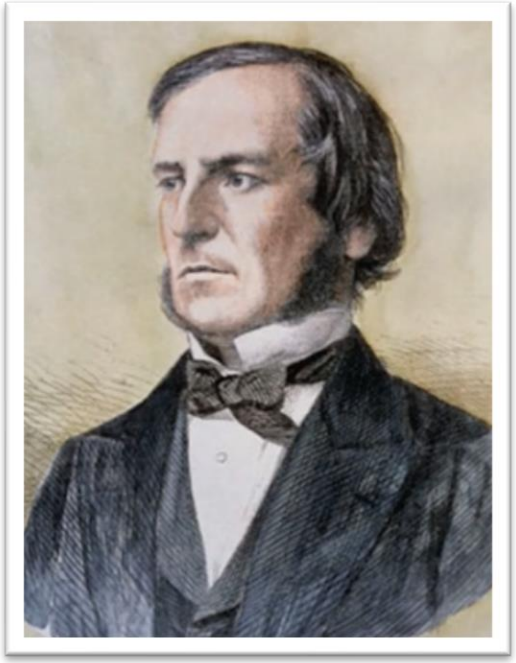
- 除了需要表示出不同的数值之外，还需要对这种表示出来的信息进行运算
  - 1679年，德国数学家莱布尼茨发表了一篇关于二进制表示及**算术运算**的论文
  - 英国数学家乔治·布尔在1854年给出了二进制的**逻辑运算**，布尔代数即由此得名
  - 这些工作奠定了现代计算机工作的**数学基础**

逻辑运算

- 又称布尔运算，用数学方法研究逻辑问题，等式表示判断，推理表示等式的变换
  - 布尔代数：变换的有效性依赖于符号的组合规律
  - 20世纪30年代，逻辑运算在电路系统上获得应用
  - 复杂电子计算机系统遵从布尔代数的变换规律
- 逻辑表达式：用逻辑运算符将关系表达式或逻辑量连接起来的有意义的式子
  - 运算结果是一个逻辑值，即“true”或“false”
  - C语言编译系统逻辑运算结果对应二进制的1和0

布尔代数

- 英国数学家George Boole
- 逻辑值True（真）和False(假）
- 编码为二进制位的1和0
- 逻辑推理的基本原则
  - 实现逻辑表达式
  - 描述逻辑函数
  - 表示逻辑运算



布尔代数运算符

- 数值取值非0即1
- 三种典型的运算符，分别表示与、或、非函数：
  - “与(AND)”运算符使用符号“•”表示
    - A•B，即A AND B，当且仅当二者取值都为1时，结果才为1
  - “或(OR)”运算符使用符号“+”表示
    - A+B，即A OR B，当其中至少有一个变量取值为1时，结果为1
  - “非(NOT)”运算符使用符号“-”表示
    - $\overline{A}$，即NOT A，当A取值为1时，结果为0；当A取值为0时，结果为1
- 其他两种常用的符，分别表示异或、同或函数：
  - “异或(XOR)”运算符“⊕”表示
    - A⊕B，相异为1，相同为0
  - “同或(XNOR)”运算符“⊙”表示
    - A⊙B，相同为1，相异为0

真值表

- 表示逻辑运算的简便方法
- *n*+1列
  - 前*n*列对应*n*个源操作数
  - 最后一列表示每种组合的输出
- 2^n行
  - 每个源操作数0或1，源操作数组合有2^n种可能
  - 每组值组合（输入组合）：真值表中的一行

A	B	AND/OR/NOT
0	0	
0	1	
1	0	
1	1	

与函数（AND）

- A AND B(A•B)，二元函数，需要两个源操作数
- 两个操作数中有一个为0，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
- 简单理解：A、B两个开关的串联电路

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**

或函数（OR）

- A OR B(A+B)，二元函数，需要两个操作数
- 两个操作数中有一个为1，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
- 简单理解:A、B两个开关的并联电路

或函数（OR）

- 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

非函数（NOT）

- NOT A（$\overline{A}$），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

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**

异或函数（ exclusive-OR，XOR）

- A XOR B （A⊕B），二元函数，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

判断两个位组合是否相同

简单理解：加法运算

同或函数(XNOR)

- A XNOR B (A⊙B)，二元函数，两个源操作数
- 相同为1，相异为0

A	B	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

布尔代数的基本公式、定律

类型	OR	AND	NOT
恒等定律	A+0=A	A · 1=A	
0/1定律	A+1=1	A · 0=0	
重叠/还原律	A+A=A	A · A=A	$\overline{\overline{A}} = A$
互补律	$A + \overline{A}$=1	$A \cdot \overline{A}$=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)	
吸收律	A+A · B=A	A · (A+B)=A	
反演律/德摩根定律	$\overline{A \cdot B \cdot C \cdots}=\overline{A}+\overline{B}+\overline{C}+\cdots$	$\overline{A + B + C \cdots}=\overline{A} \cdot \overline{B} \cdot \overline{C}\cdots$	
其他恒等式	$A + \overline{A}$ · B=A+B	$A \cdot (\overline{A} + B) = A \cdot B$	

逻辑函数化简

- ABC+$\overline{A}$BC
  - =(A+$\overline{A}$)BC=BC（与·可省去）
  - 利用互补律A+$\overline{A}$=1
- AB+ABC
  - =AB（1+C）=AB
  - 利用0/1定律1+A=1或吸收律A+AB=A
- AB+$\overline{A}$ C+$\overline{\mathrm{B}}$C
  - =AB+($\overline{A}$ +$\overline{\mathrm{B}}$)C=AB+$\overline{\mathrm{AB}}$C=AB+C
  - 利用德摩根$\overline{A}$ +$\overline{\mathrm{B}}$= $\overline{\mathrm{AB}}$，恒等式$A+\overline{A}\cdot B=A+B$
- AB+BC+A$\overline{\mathrm{C}}$
  - =AB(C+$\overline{C}$)+BC+A$\overline{\mathrm{C}}$=ABC+AB$\overline{C}$+BC+A$\overline{\mathrm{C}}$

    =（A+1）BC+(B+1)A$\overline{\mathrm{C}}$=BC+A$\overline{\mathrm{C}}$
  - 利用的0/1定律1+A=1、互补律$A+\overline{A}$=1

减少所含乘积项的个数
&
减少每个乘积项中变量的个数

按位(逻辑)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[*m*-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

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

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

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

C的按位运算符

- C语言的低级语言特征之一
- 对数值进行位（布尔）运算的运算符集合
  - “&”：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

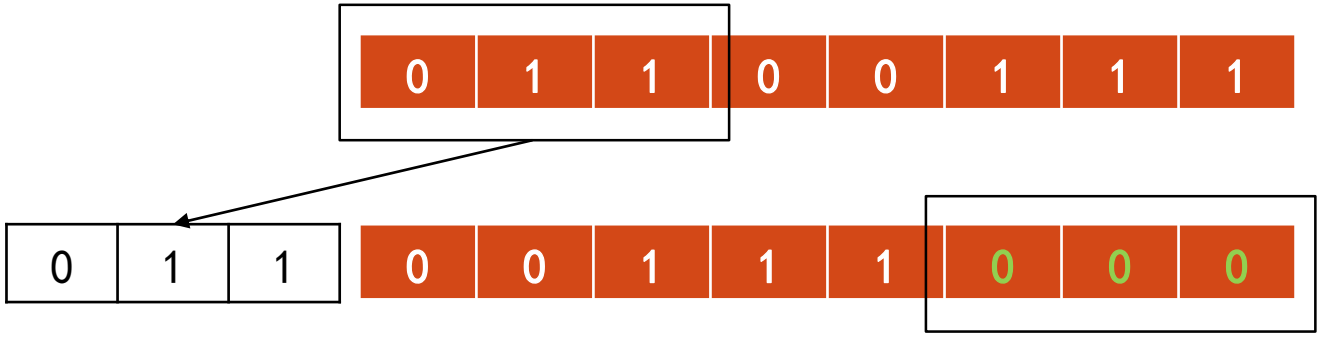
位运算表达式	二进制	位运算结果	十六进制
~0x41	~[0100 0001]	[1011 1110]	0xBE
~0x00	~[0000 0000]	[1111 1111]	0xFF
0x69 & 0x55	[0110 1001]&[0101 0101]	[0100 0001]	0x41
0x69 | 0x55	[0110 1001]|[0101 0101]	[0111 1101]	0x7D

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
- 有符号数算术移位，无符号数逻辑移位
- 算术/逻辑左移，零填充，不考虑符号
- 逻辑右移，零填充，不考虑符号
- 算术右移，符号扩展

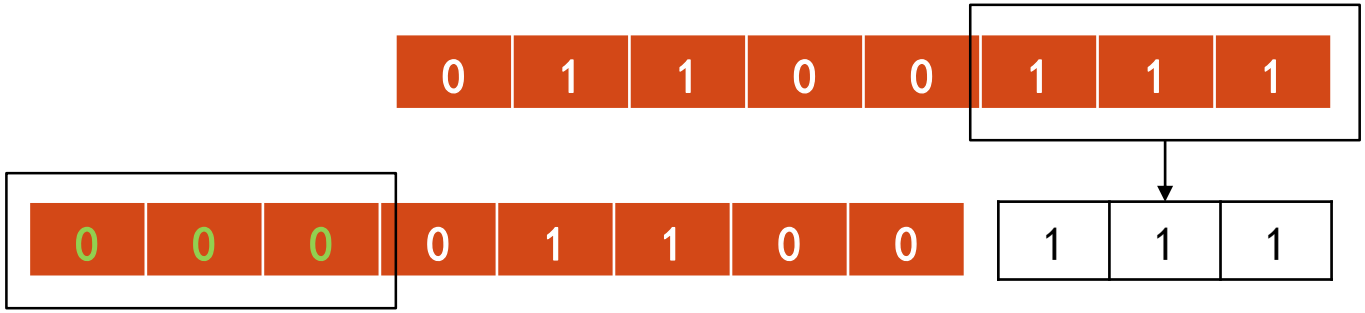
示例

- 8位整数：01100111 << 3（算数/逻辑左移）



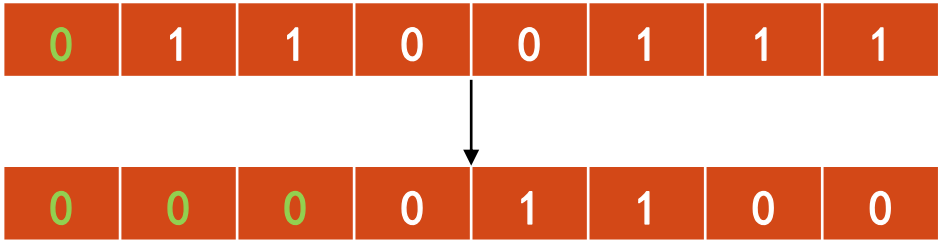
示例

- 8位整数：01100111 >> 3（逻辑右移）

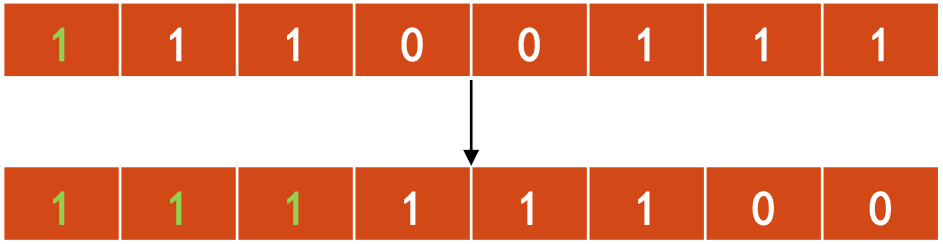


示例

- 8位整数：01100111 >> 3（算数右移）

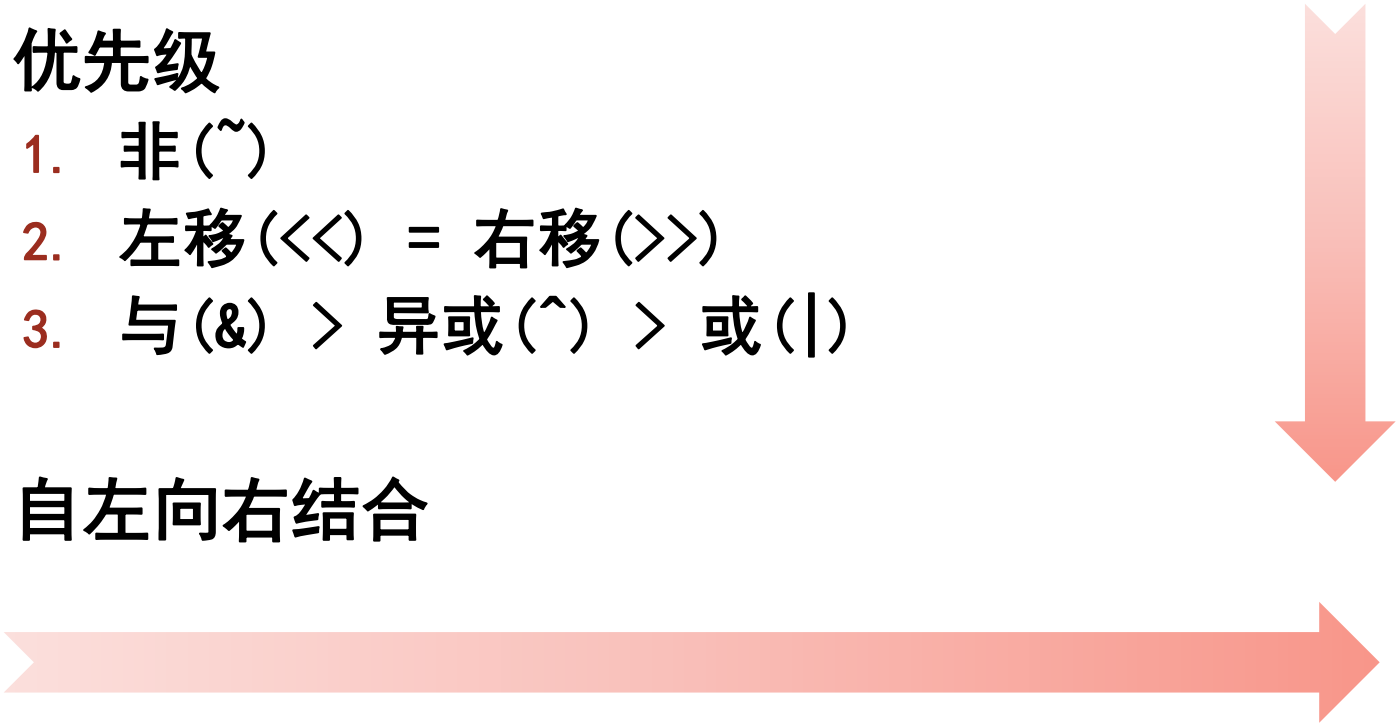


- 8位整数：11100111 >> 3（算数右移）



优先级及结合性

- 优先级
  1. 非(~)
  2. 左移(<<) = 右移(>>)
  3. 与(&) > 异或(^) > 或(|)
- 自左向右结合



示例

- x = ~a | ~b;　　/*如果a=3，b=4*/
  - x = ~(0011) | ~(0100)
  - x = ~(0011) | ~(0100) = (1100) | ~(0100)
  - x = (1100) | ~(0100) = (1100) | (1011)
  - x = (1100) | (1011) = (1111)

- C语言的按位运算符，操作数都不能是浮点数，语句中的x、a、b都是整数。
- x = (1111) = -1，这里x是有符号整数

C的逻辑运算符

- 与位运算不同
- 所有非零的参数都为True，只有参数0为False
  - 位运算只有在特殊的数值条件下才得到0或1
- 逻辑运算的运算符集合
  - “&&”：“与”（AND）
  - “||”：“或”(OR)
  - “！”：“非”(NOT)

示例

逻辑运算表达式	结果
!0x41	0x00
!0x00	0x01
!!0x41	0x01
0x69 && 0x55	0x01
0x69 || 0x55	0x01
if(a && 5/a)	若a=0，则结果为0x00

习题（1）

- 书面作业
- 6.9
- 6.16

- 上机作业
- 6.20

集成电路

大型计算机时代

第一代计算机

1946-1958年

主要元件：电子管

特征：体积大、耗电量大、速度慢、存储容量小、可靠性差、维护困难、价格贵

性能：数千~万次/秒

应用领域：军事和科学计算

第二代计算机

1958-1964年

主要元件：晶体管

特征：尺寸小、重量轻、寿命长、效率高、发热少、功耗低

性能：数十万次/秒

应用领域：科学计算、数据处理、过程和工业控制

第三代计算机

1964-1971年

主要元件：中小规模集成电路

特征：体积更小、速度更快、价格更便宜

性能：数十万次~百万次/秒

应用领域：文字处理、图形图像处理

第四代计算机

1971-至今

主要元件：大规模和超大规模集成电路

特征：体积缩小、存储容量扩大、外部设备种类多、用户使用方便

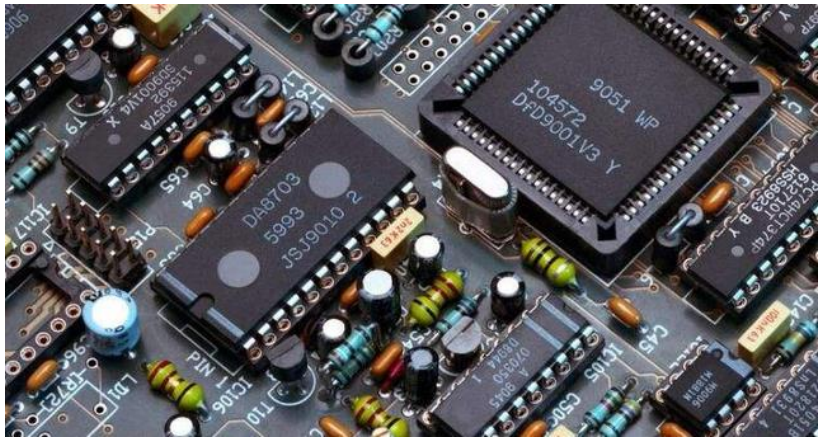
性能：数百万次~数千万次/秒

应用领域：渗透到各个行业

集成电路（Integrated Circuit）

- 采用一定工艺，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/几小块**半导体（硅）**晶片/介质基片上
- 封装在一个管壳内，使其具有所需电路功能

- ✓ **小体积**
- ✓ **低功耗**
- ✓ **高可靠**



按元器件集成度分类

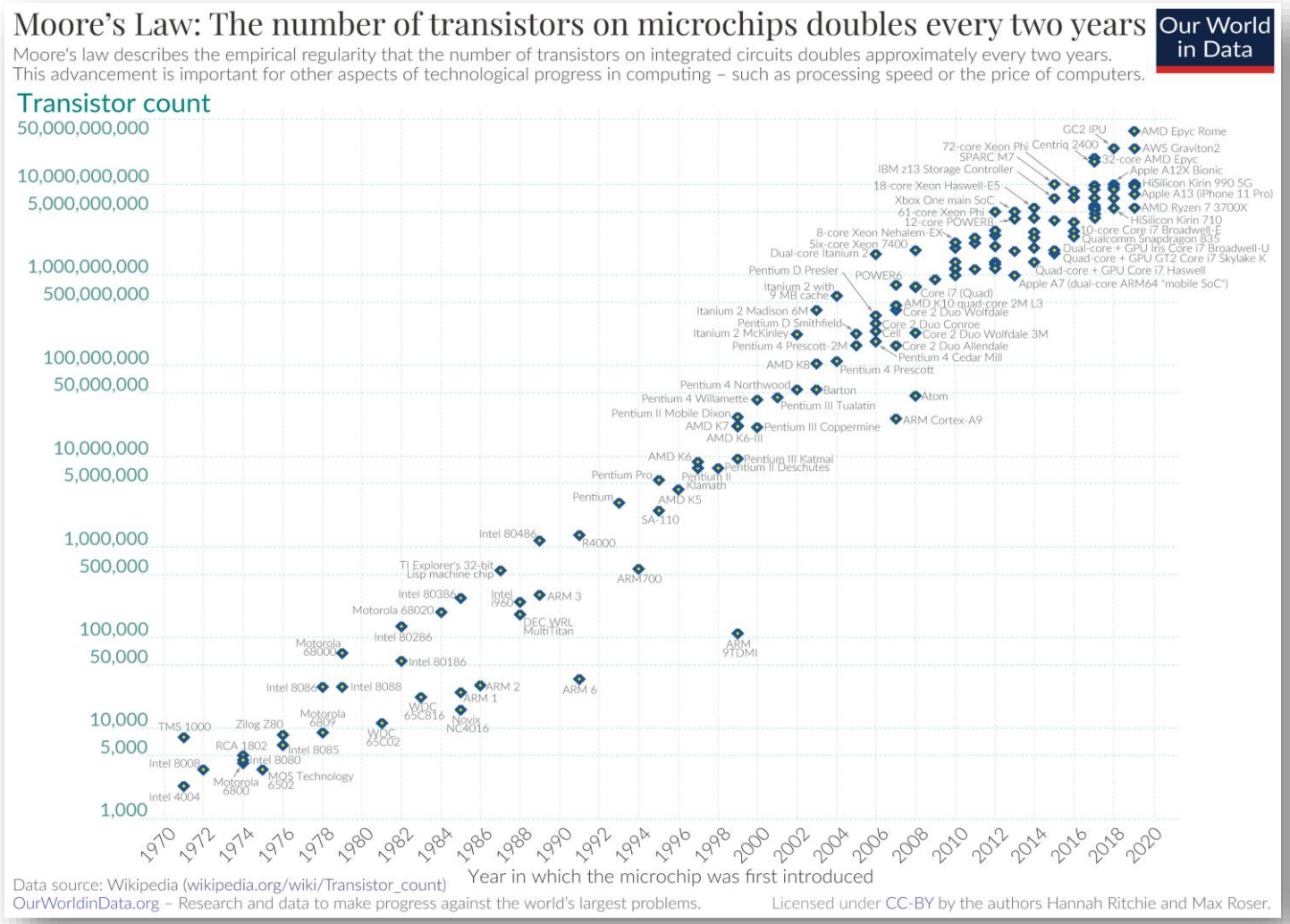
集成电路	
小规模集成电路	SSI（Small Scale Integration）
	元器件集成度小于100
	各种逻辑门电路、集成触发器等
中规模集成电路	MSI（Medium Scale Integration）
	元器件集成度大于100小于1000
	译码器、数据选择器、寄存器等
大规模集成电路	LSI（Large Scale Integration）
	元器件集成度大于1000（数万）
	存储器、计算机外设等
超大规模集成电路	VLSI（Very large scale integration）
	元器件集成度大于十万
	存储器、微型计算机等
特大规模集成电路	ULSI（Ultra Large-Scale Integration）
	元器件集成度大于一千万（器件尺寸进入深亚微米级领域）
	16M FLASH（闪存）和256M DRAM（动态随机存取存储器）等
巨大规模集成电路	GSI（Giga Scale Integration）
	元器件集成度大于一亿
	1G DRAM等

摩尔定律

- 集成电路上的元器件集成度每隔18-24个月便会增加一倍，成本保持不变
- 揭示了信息技术进步的速度

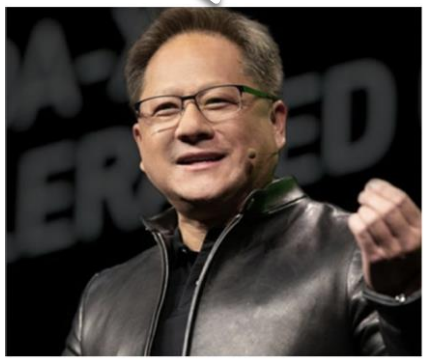
英特尔（Intel）创始人之一
Gordon Moore（戈登·摩尔）

摩尔定律


Moore's Law: The number of transistors on microchips doubles every two years
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.
Our World in Data
Transistor count
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
Year in which the microchip was first introduced
GC2 IPU
AMD Epyc Rome
72-core Xeon Phi
Centriq 2400
AWS Graviton2
SPARC M7
32-core AMD Epyc
IBM z13 Storage Controller
Apple A12X Bionic
18-core Xeon Haswell-E5
HiSilicon Kirin 990 5G
Apple A13 (iPhone 11 Pro)
Xbox One main SoC
AMD Ryzen 7 3700X
61-core Xeon Phi
12-core POWER8
HiSilicon Kirin 710
8-core Xeon Nehalem-EX
10-core Core i7 Broadwell-E
Six-core Xeon 7400
Qualcomm Snapdragon 835
Dual-core Itanium 2
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Pentium D Presler
POWER6
Quad-core + GPU Core i7 Haswell
Apple A7 (dual-core ARM64 "mobile SoC")
Itanium 2 with 9 MB cache
Core i7 (Quad)
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Atom
Pentium 4 Willamette
Pentium III Tualatin
Pentium II Mobile Dixon
ARM Cortex-A9
AMD K7
Pentium III Coppermine
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II
Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
ARM 6
ARM 1
WDC 65C816
Motorola 6809
Novix NC4016
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8008
Intel 8080
MOS Technology 6502
Motorola 6800
Intel 4004
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.
Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

摩尔定律

摩尔定律已死，芯片成本会随着时间推移而下降的想法已经是过去时了……加速计算才是真正的前进道路。



英伟达CEO**黄仁勋**

摩尔定律不会死，至少在未来的十年里依然有效。我们希望到2030年在一个芯片封装上可以有1万亿个晶体管。

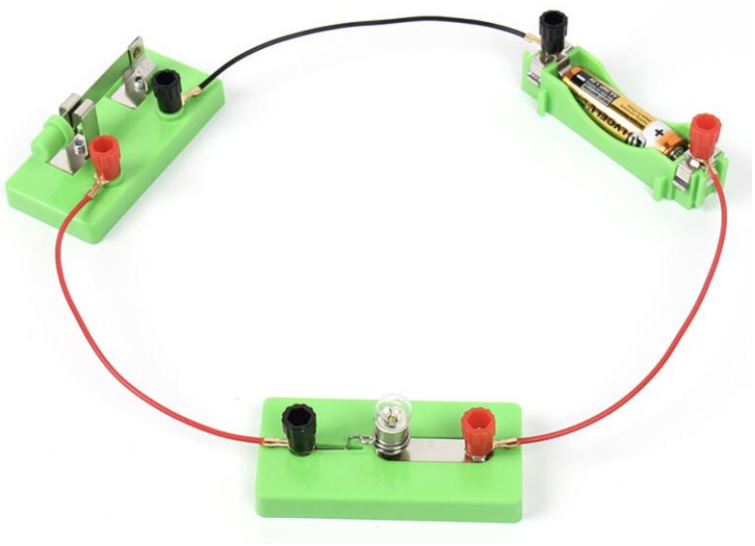
英特尔CEO**帕特·基辛格**

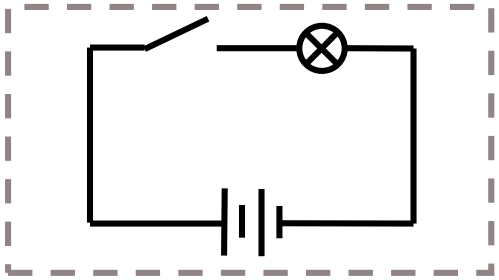
晶体管

- MOS晶体管
- NMOS晶体管
- PMOS晶体管
- CMOS电路

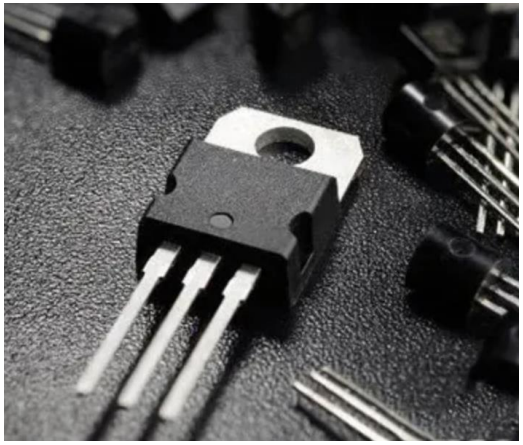
简单电路

- 电灯发光：闭合电路、电子的流动
- 操纵开关：控制电路的合与开、电灯的亮与灭
- 开关逻辑上可用**晶体管**来代替





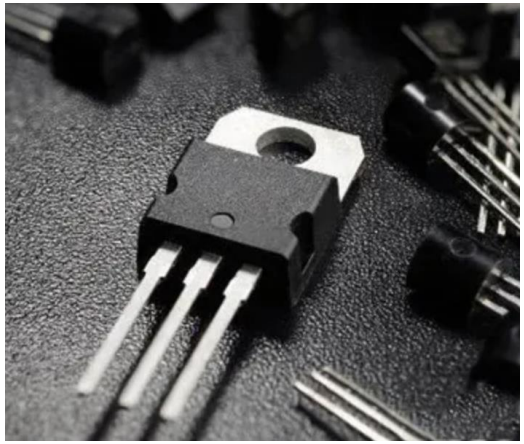
MOS晶体管

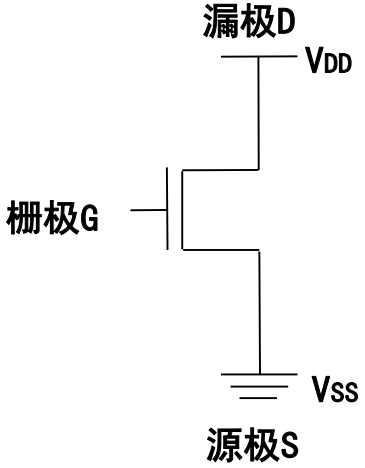


- 1960年发明
- MOSFET（Metal-Oxide-Semiconductor Field Effect Transister，金属氧化物半导体场效应晶体管）
- 占据半导体市场的极大份额
- 广泛应用于各种电子器件
- 数以百亿计的晶体管集成度（摩尔定律）
- 推动现代电子产业的高密度高集成化发展

MOS晶体管

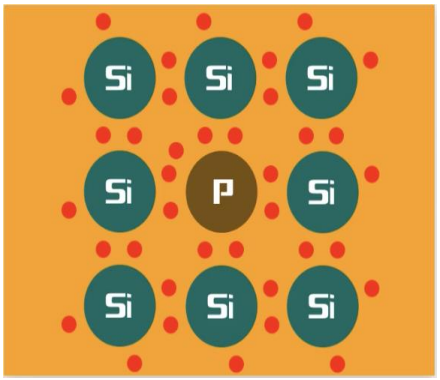
- 原理：依靠输入回路的电场效应来控制输出回路电流的半导体元件
- 与机械开关不同：用电压控制电路开关
- 三个端口：**源极（S）、栅极（G）、漏极（D）**

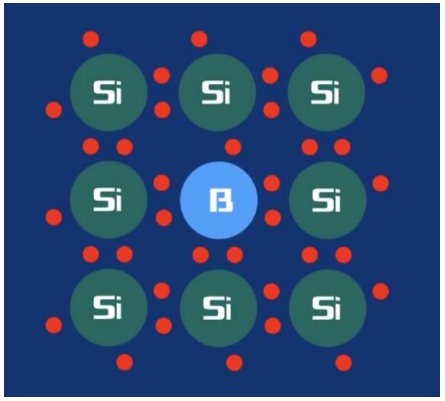




MOS晶体管

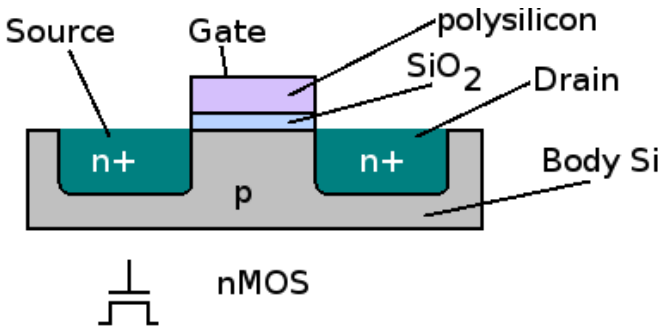
- N型(沟道)和P型(沟道) MOS晶体管
- 都会用到P型和N型半导体
- N型半导体
  - 纯硅中掺入5价磷元素
  - 原本四价稳定的硅元素多一个电子
  - 自由电子带负电，N型半导体
- P型半导体
  - 纯硅中掺入3价硼元素
  - 缺少电子，产生空穴
  - 吸引电子，对外显正电，P型半导体





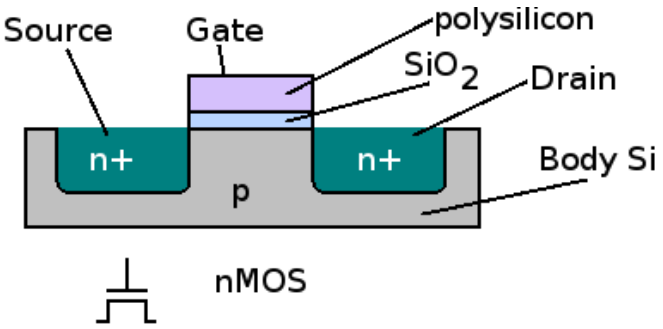
N型(沟道)MOS晶体管（NMOS）

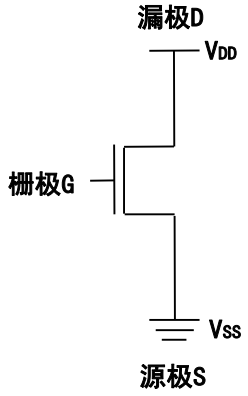
- P型半导体打底
- 两块N型半导体（源极S和漏极D）
- 中间二氧化硅+金属板，引出栅极G
- 栅极施加正电压，金属板电场吸引P型半导体中的电子到绝缘层附近，形成全部是电子的N沟道
- N型半导体联通，电路闭合



NMOS

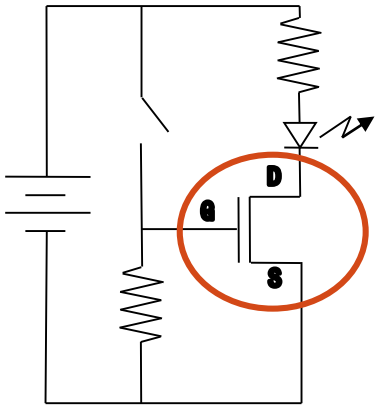
- 如果栅极施加正电压，源极到漏极间相当于一段电线，即存在闭合回路，**导通**
  - **V_{DD}表示漏极接电源正极，V_{SS}表示源极接电源负极**
- 如果栅极被施加负电压，源极和漏极之间的连接就被断开，在源极和漏极之间存在断路，即**截止**





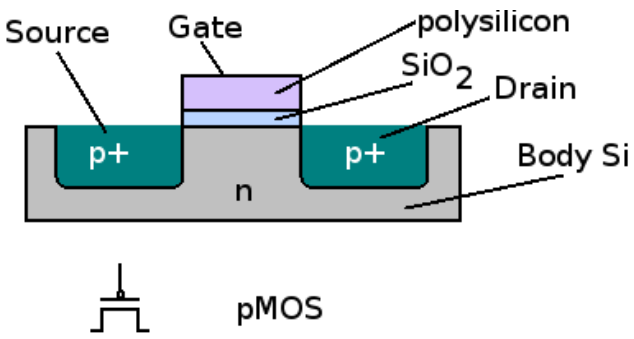
NMOS晶体管—电子开关

- 闭合开关，NMOS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



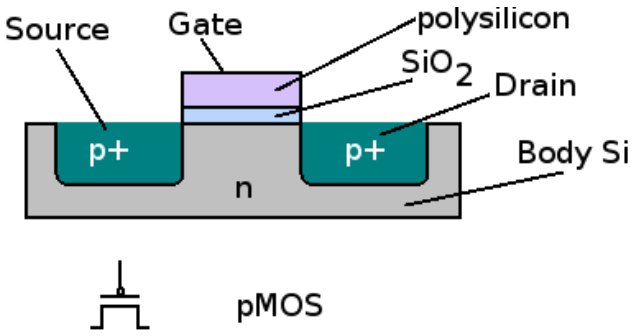
P型(沟道)MOS晶体管（PMOS）

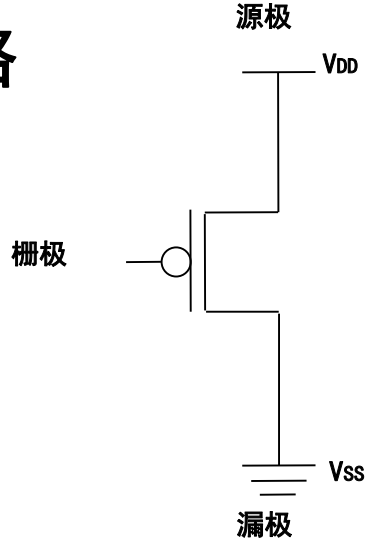
- N型半导体打底
- 两块P型半导体（源极S和漏极D）
- 中间二氧化硅+金属板，引出栅极G
- 栅极施加负电压，N型半导体的负电子被排斥，远离绝缘层，空穴被吸引，形成P沟道
- P型半导体联通，电路闭合



P型(沟道)MOS晶体管（PMOS）

- 工作原理与NMOS晶体管相反
- 当给栅极施加负电压时，P型晶体管像一段电线，构成闭合回路
  - V_{DD}表示源极接电源正极，V_{SS}表示漏极接电源负极
- 当施加正电压时，就出现断路





CMOS电路

- NMOS晶体管用于源极接地、栅极施加正电压的低端驱动，要求漏极接电源正极
- PMOS晶体管用于源极接电源正极、栅极施加负电压的高端驱动，要求漏极接地
- P型和N型晶体管以互补的方式工作
- CMOS（Complementary MOS）电路
  - 既包含PMOS晶体管又包含NMOS晶体管的电路
  - 互补金属氧化物半导体电路

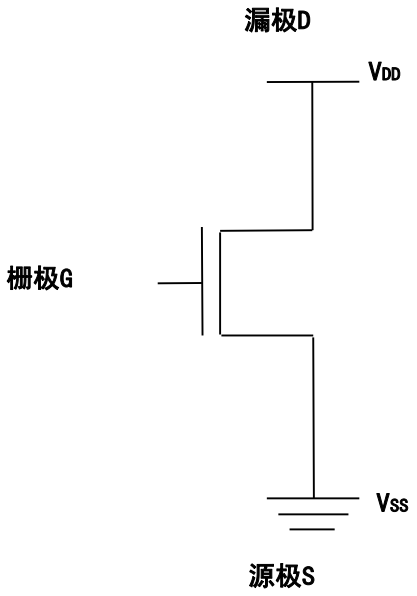
门电路

- 非门
- 或门、或非门
- 与门、与非门

门电路

- 多复杂的芯片都是由不同的逻辑门电路组成
- 三种基本类型：与门、或门、非门
  - 实现与、或、非**逻辑运算**的晶体管电路
  - 在此基础上的与非门、或非门
- 构建门电路最基本的元器件是**MOS晶体管**

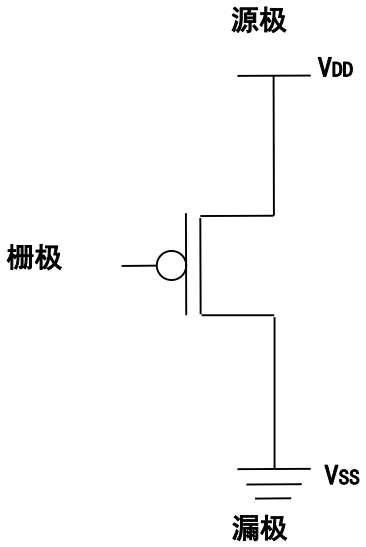
NMOS管



- 栅极施加正电压导通，负电压断开
  - 正电压用1表示，负电压用0表示
- 漏极接电源正极，源极接地

输入	结果
正电压（1）	电路导通
负电压（0）	电路断开

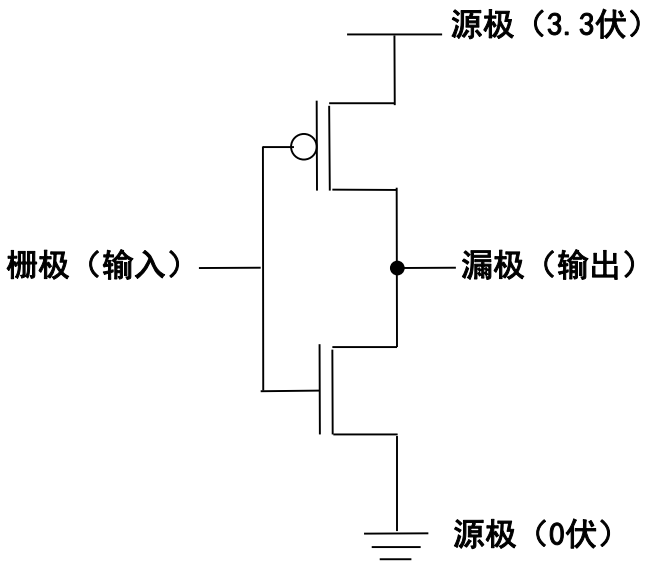
PMOS管



- 栅极施加负电压导通，正电压断开
  - 正电压用1表示，负电压用0表示
- 源极接电源正极，漏极接地

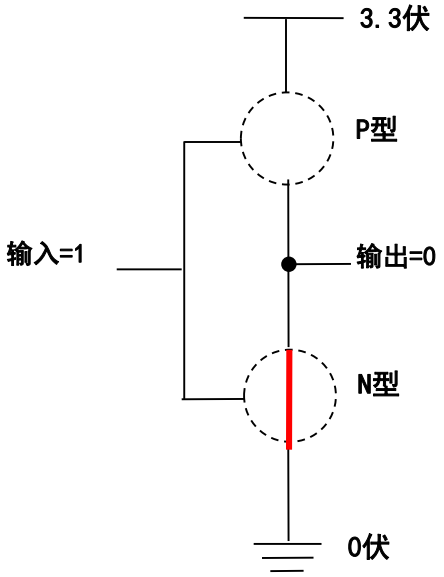
输入	结果
正电压（1）	电路断开
负电压（0）	电路导通

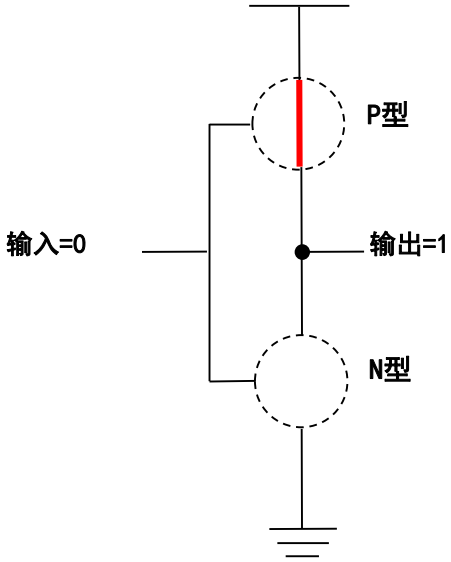
非门（反相器）



- PMOS（上）和NMOS（下）接在一起
- 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（上）接电源正极；
  - NMOS管的源极（下）接地

非门（反相器）





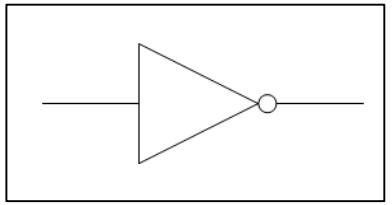
输入	输出
正电压（1）	0伏（NMOS导通）
负电压（0）	3.3伏（PMOS导通）

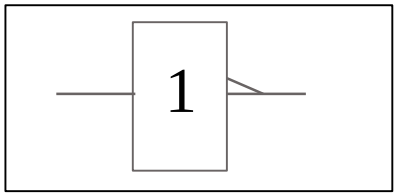
输入	输出
1	0
0	1

二进制逻辑运算非函数

非门（反相器）

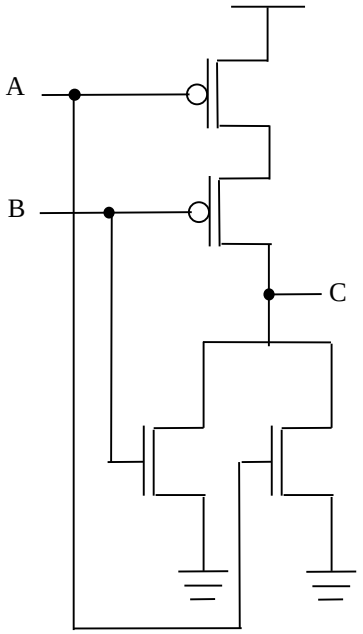
- ANSI/IEEE Std 91-1984
  - IEEE Standard Graphic Symbols for Log Functions
  - 形状特征型符号
- IEC 60617-12
  - International Electrotechnical Commission，国际电工委员会，Graphical Symbols for Diagrams-Part 12: Binary Logic Elements
  - 矩形国标符号



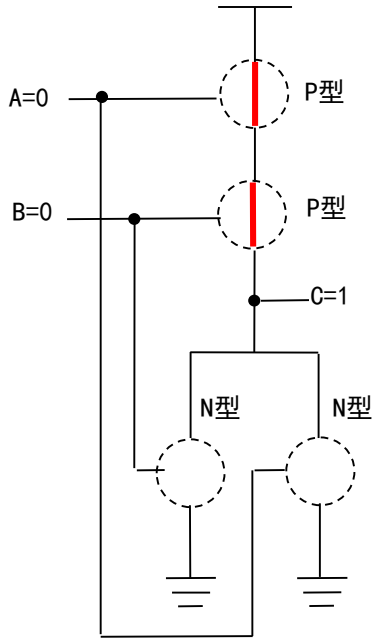


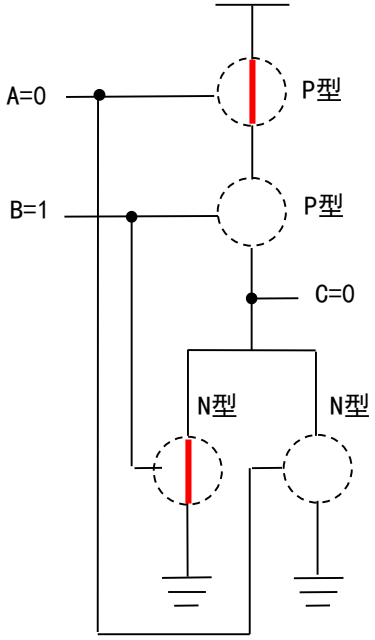
或非门

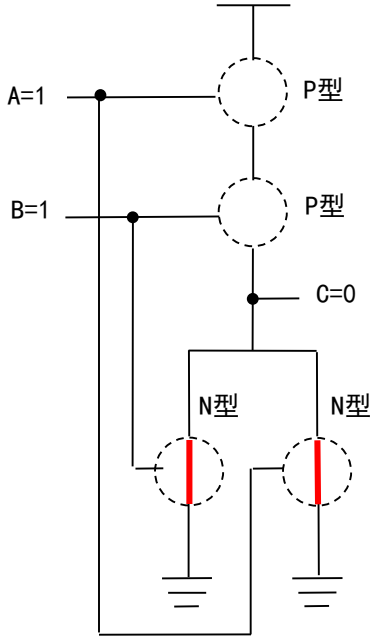
- 顶部两个PMOS管，串联
- 底部两个NMOS管，并联
- 编号从1-4
- 输入为A、B
  - A连接1和4的栅极
  - B连接2和3的栅极
- 输出为C，连接漏极
- PMOS管1的源极接电源正极
- NMOS管3、4的源极接地



或非门





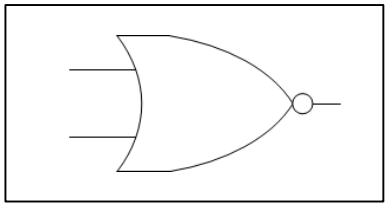


A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

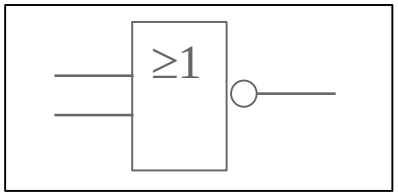
或函数的相反结果

或非门符号表示

- ANSI/IEEE Std 91-1984
  - 形状特征型符号



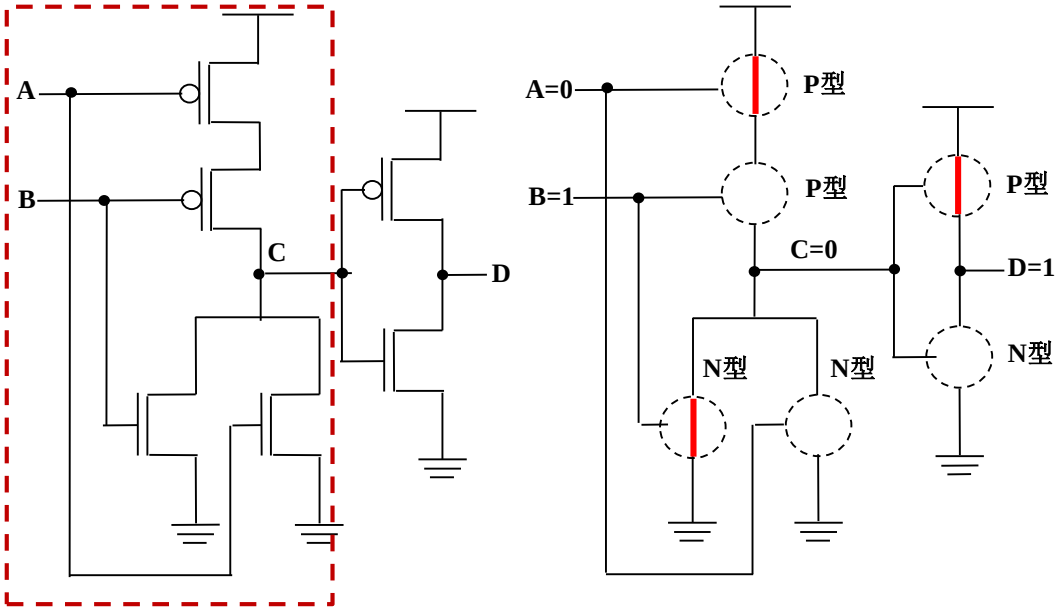
- IEC 60617-12
  - 矩形国标符号



或门

=

- 在或非门输出端增加一个非门（反相器）

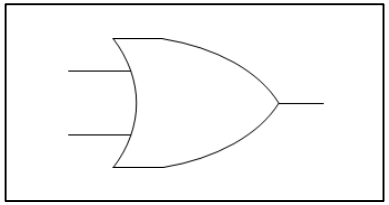


A	B	C	D
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

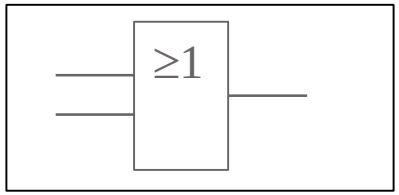
二进制逻辑运算或函数

或门符号表示

- ANSI/IEEE Std 91-1984
  - 形状特征型符号

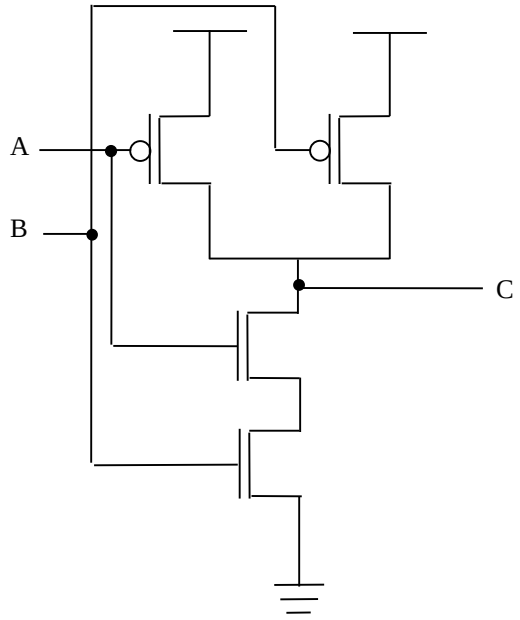


- IEC 60617-12
  - 矩形国标符号



与非门

- 顶部两个PMOS管并联，底部两个NMOS管串联

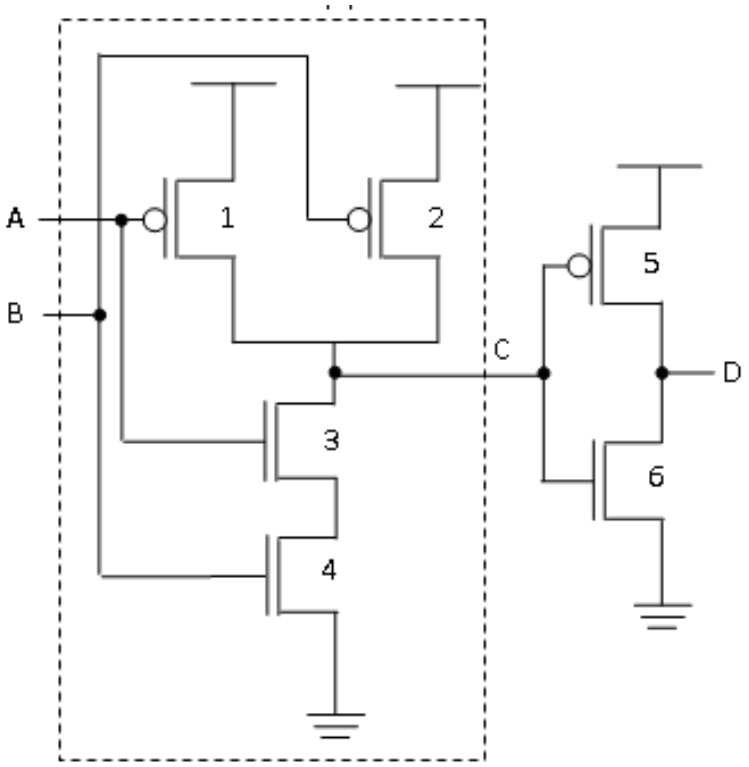


A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

与函数的相反结果

与门

- 在与非门后增加非门（反相器）

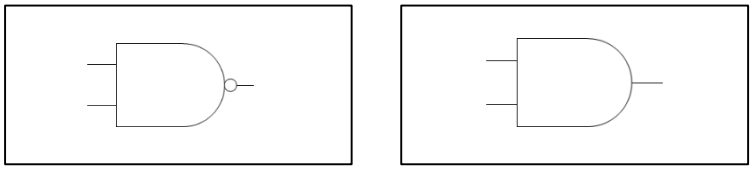


A	B	C	D
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

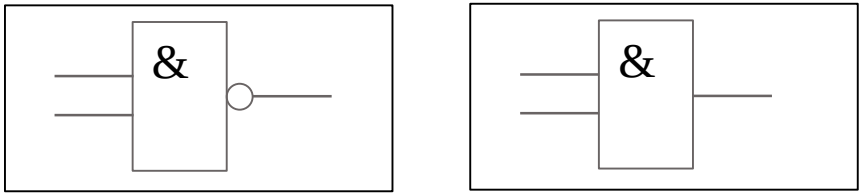
二进制逻辑运算与函数

与非门/与门符号表示

- ANSI/IEEE Std 91-1984
  - 形状特征型符号

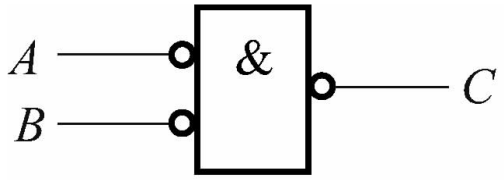


- IEC 60617-12
  - 矩形国标符号

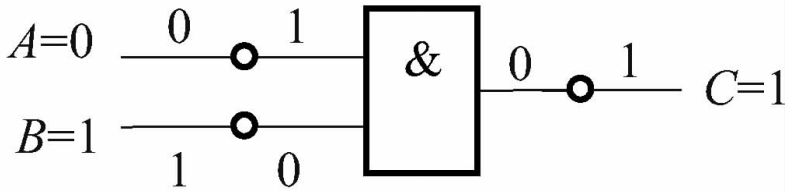


德摩根定律

- NOT ((NOT A) AND (NOT 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

<==> A + B

德摩根定律

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

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

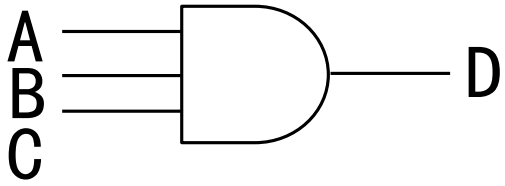
- NOT (A AND B) = (NOT A) OR (NOT B)

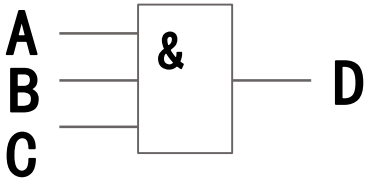
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

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
  - 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





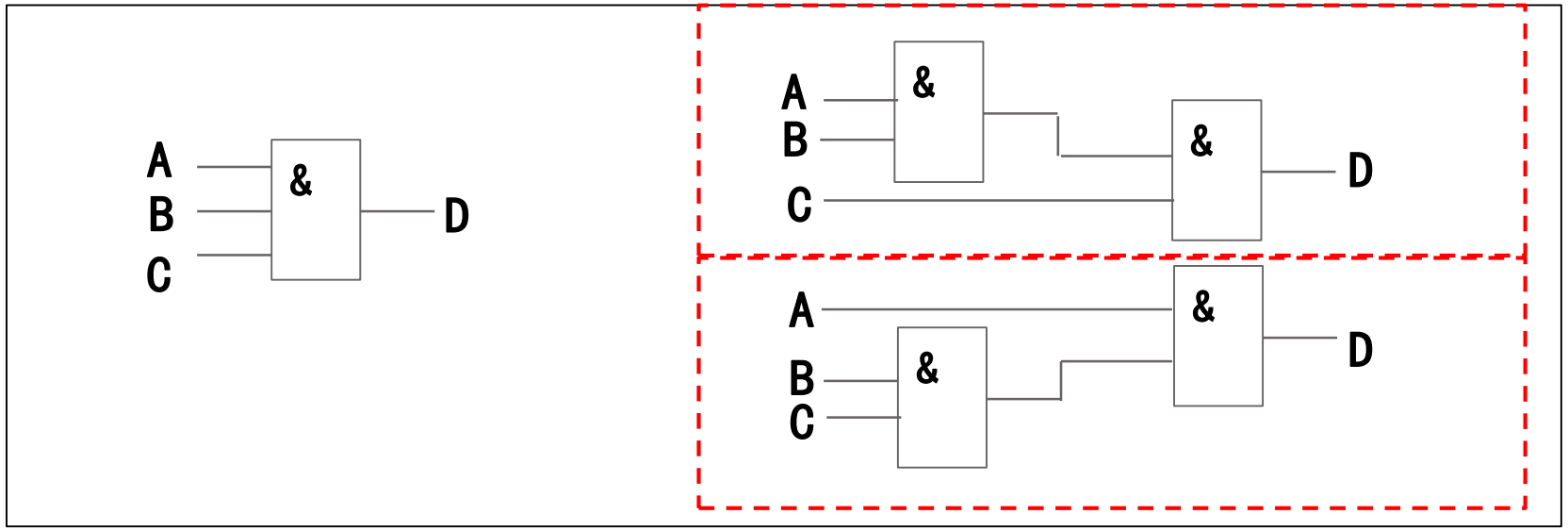
3个输入的与门——真值表

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？
- 2、一个有四个输入的或门的晶体管级电路如何构建？
- 3、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（2）

- 书面作业
- 7.1
  - 1）
  - 2）
    - II
- 7.2
- 7.3

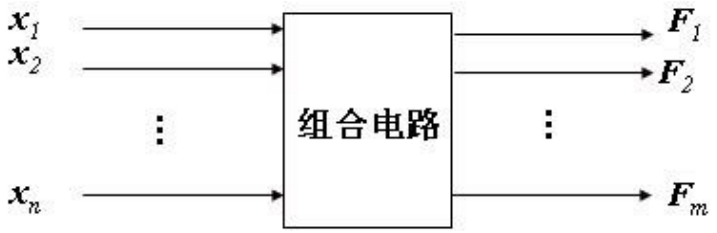
组合逻辑电路

- 基本介绍
- 译码器
- 多路选择器
- 全加法器

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



$$F_i = f(x_1, x_2, \cdots, x_n) \qquad (i = 1,2,\cdots,m)$$

- 不能存储信息的逻辑结构
- “判定元件”、组合逻辑结构/电路
  - 输出在任何时刻只取决于同一时刻的输入状态
  - 与电路过去的状态无关
  - 输出和输入之间可用逻辑函数来描述
- 特点：
  - 信息不能存储在组合逻辑电路中（无记忆功能）
  - 输出、输入之间没有反馈延迟通路
- 应用：主要用于处理信息
  - 译码器，多路选择器，全加法器等

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（尽量采用与非门设计）

通用性
简单性
低功耗
高集成度
快速响应
容易级联

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

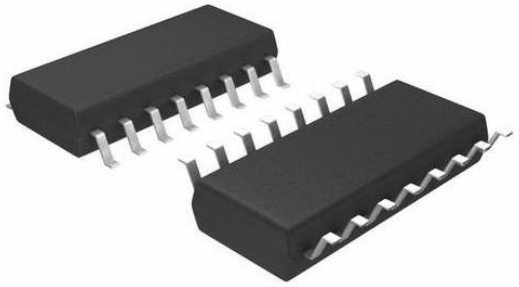
$$C = \bar{A}B + A\bar{B} = A \oplus B$$

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

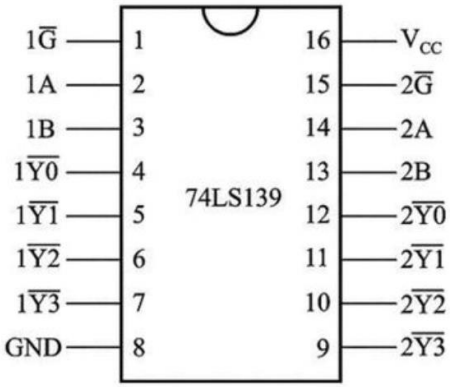
一、译码器

- 译码：将特定含义的二进制码转换成输出信号
  - 输入少，输出多
  - 编码的逆过程
  - 编码：将信号转换成二进制代码，多对少
- 译码器：具有译码功能的逻辑电路
  - 唯一地址译码器：一系列代码译成对应的有效信号
    - 二进制译码器：存储器寻址（地址分配、单元选择）
    - 二-十进制译码器：二进制转换为BCD格式(数字显示、计数器)
    - 显示译码器：数码显示器（驱动信号可视化显示）
  - 代码变换器：一种代码转换成另一种代码

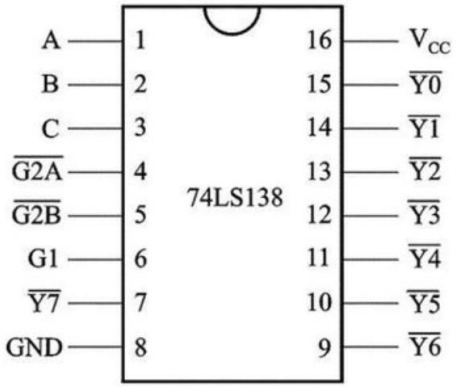
二进制译码器



- 有使能端、n个输入，2^n个输出
- （双）2-4线译码器、3-8线译码器、4-16线译码器
- 只有一个输出为有效
- 有效输出对应于要被检测的输入组合



双2-4线译码器



3-8线译码器

2-4线译码器

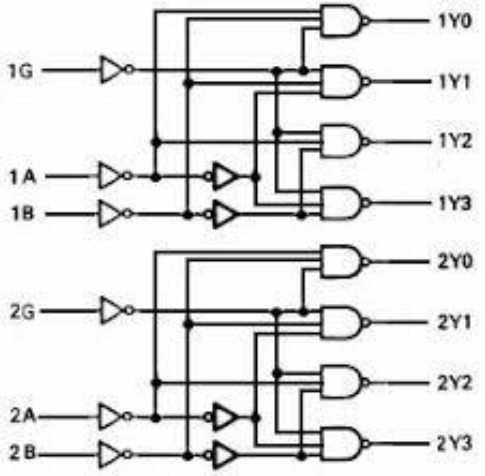
- n=2，1个使能端G，2个输入(A、B)，4个输出(Y_0-Y_3)
- 使用非门产生A和B的反变量$\overline{A}$ 和$\overline{B}$
- 信号送入4个不同的与非门
- 四种可能的输入组合中，仅有一个输出有效（低/高电平）
- 双2-4线译码器：2个独立的译码器

$\overline{G}$	B	A	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

*这里$\overline{G}$、$\overline{Y}*$代表低电平0有效，不是取非，等价于G、Y_{0-3}

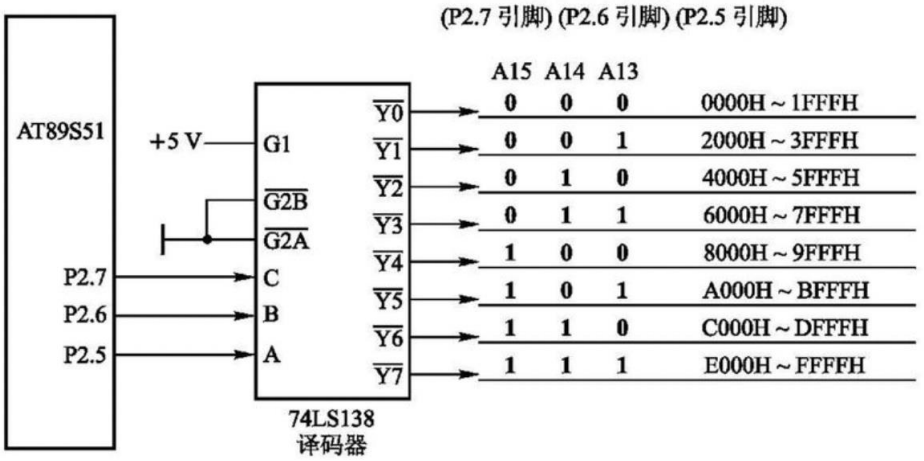
$$\overline{Y_0} = \overline{\overline{G} \cdot \overline{B} \cdot \overline{A}}$$
$$\overline{Y_1} = \overline{\overline{G} \cdot \overline{B} \cdot A}$$
$$\overline{Y_2} = \overline{\overline{G} \cdot B \cdot \overline{A}}$$
$$\overline{Y_3} = \overline{\overline{G} \cdot B \cdot A}$$

* $\overline{G}$、$\overline{Y}*$等价于G、Y_{0-3}



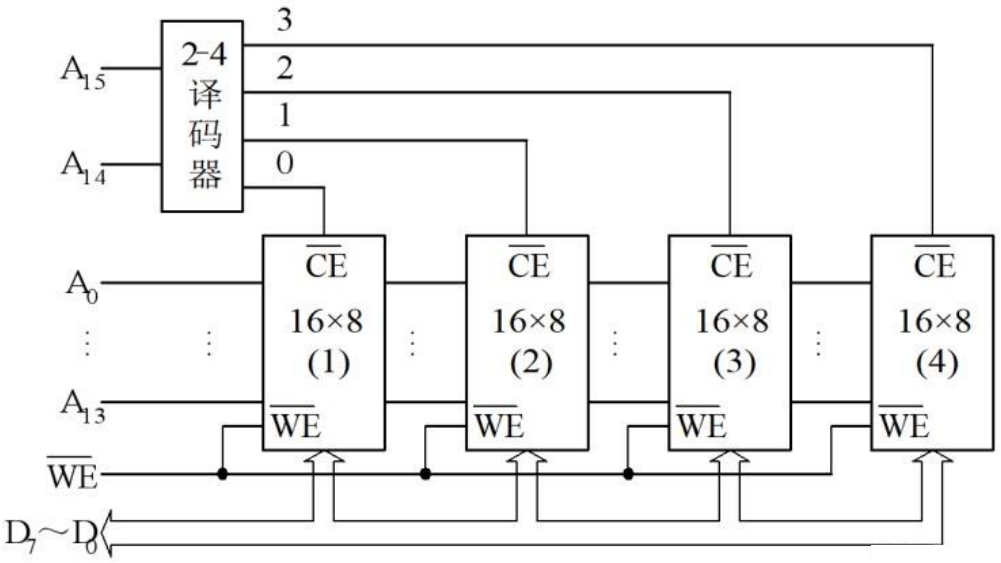
译码器应用

- 以72LS138为例，进行**地址分配**和**单元选择**
- 64KB地址值空间划分成8个8KB空间
- AT89S51发出16位信号
  - 高3位连接72LS138的3个输入端
  - 译码器8个输出连接8片芯片，有效输出实现8选1
  - 低13位完成选中芯片的存储单元选择



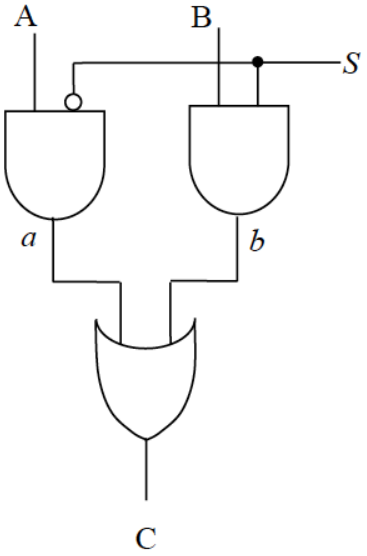
译码器应用

- 用4个16K×8位的芯片扩展为64k×8位的存储器

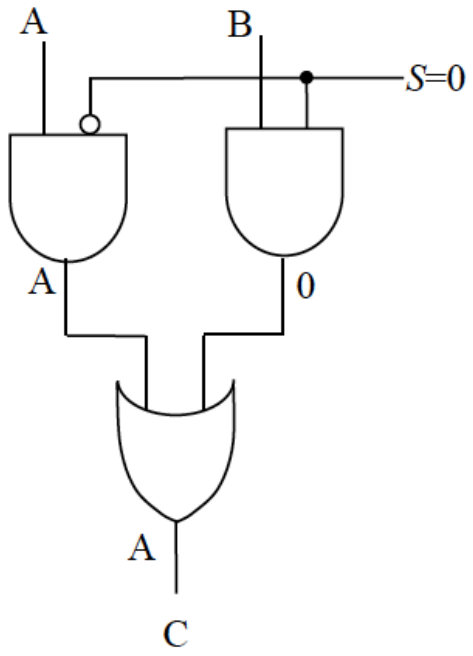


二、多路选择器

- 又叫**数据选择器、多路开关**
- 功能：选择多路输入数据中的**一路数据**连接到输出线路，实现数据选择功能
  - 类比：多个输入的单刀多掷开关
  - （总线）数据选择切换、多个输入源切换…
- 由选择信号S决定哪个输入连接到输出
- 一般由***n*条选择线**和**2^n个输入**组成
  - 右图n=1，2个输入
  - 2个与门，1个或门



多路选择器(n=1)

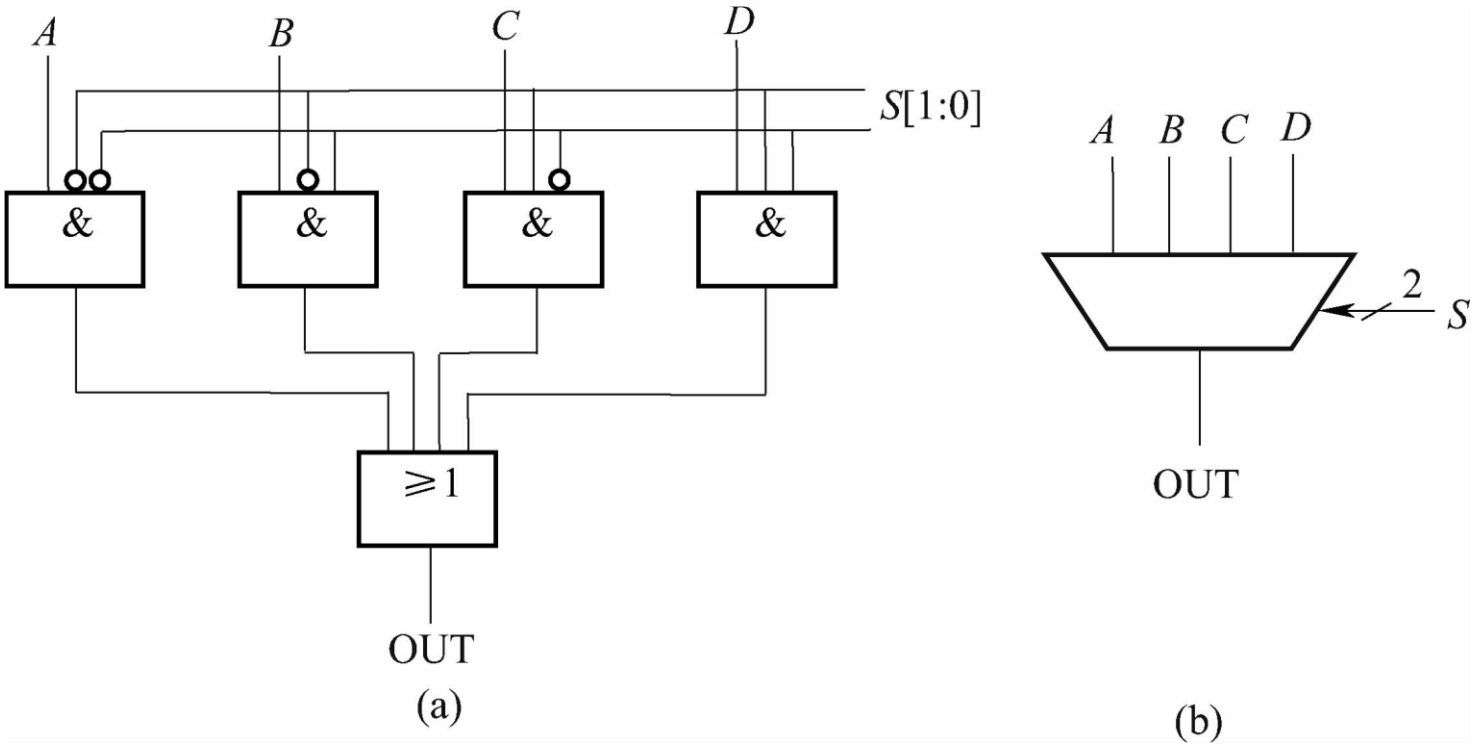


2选1选择器

选择信号S	输出
0	A
1	B

多路选择器(n=2)

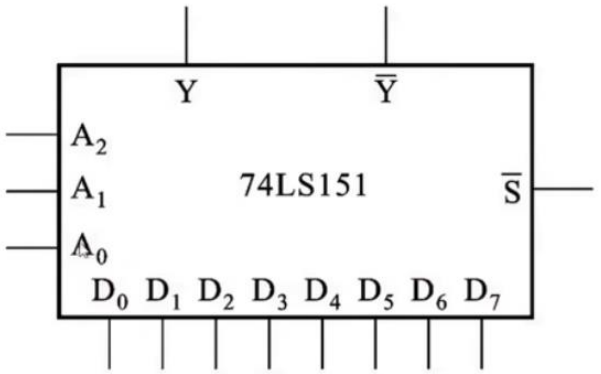
- 4选1选择器
- S（00，01，10，11），输出A,B，C或D的值



多路选择器(n=3)

- 8选1选择器74LS151
- 此处选择信号为A_2-A_0，选择对应的输出D_0-D_7
- $\overline{S}$为使能信号（低电平有效）

输入					输出	
D	A_2	A_1	A_0	$\overline{S}$	Y	$\overline{Y}$
×	×	×	×	1	0	1
D_0	0	0	0	0	D_0	$\overline{D}_0$
D_1	0	0	1	0	D_1	$\overline{D}_1$
D_2	0	1	0	0	D_2	$\overline{D}_2$
D_3	0	1	1	0	D_3	$\overline{D}_3$
D_4	1	0	0	0	D_4	$\overline{D}_4$
D_5	1	0	1	0	D_5	$\overline{D}_5$
D_6	1	1	0	0	D_6	$\overline{D}_6$
D_7	1	1	1	0	D_7	$\overline{D}_7$

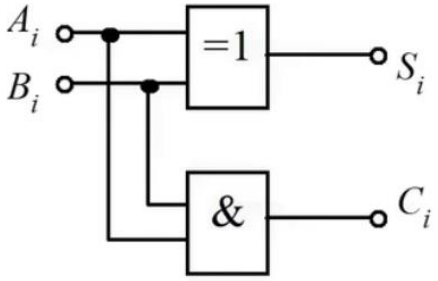


三、加法器

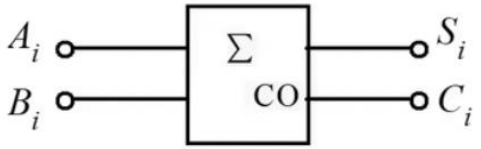
- **半加器**：实现本位两个二进制数相加的逻辑电路
  - 两个输入A_i和B_i
  - 两个输出S_i和C_i，分别表示“和”与“进位”
  - $S_i=A_i \oplus B_i=\overline{A_i}\cdot B_i+A_i\cdot\overline{B_i}$　　$C_i=A_i\cdot B_i$

A_i	B_i	S_i	C_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

半加器真值表



半加器电路图



半加器符号

加法器（全加器）

- 全加器：实现两个本位二进制数与来自低位来的进位相加的逻辑电路
- 三个输入A_i、B_i、C_{i-1}
- 两个输出S_i与C_i表示“和”与“进位”
- $S_i=A_i \oplus B_i \oplus C_{i-1}$

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

- $C_i=\overline{A_i} \cdot B_i \cdot C_{i-1}+A_i \cdot \overline{B_i} \cdot C_{i-1}+A_iB_i \cdot \overline{C_{i-1}}+ A_i \cdot B_i \cdot C_{i-1}$

  $=(\overline{A_i} \cdot B_i+A_i \cdot \overline{B_i}) \cdot C_{i-1}+A_iB_i(\overline{C_{i-1}}+C_{i-1})$

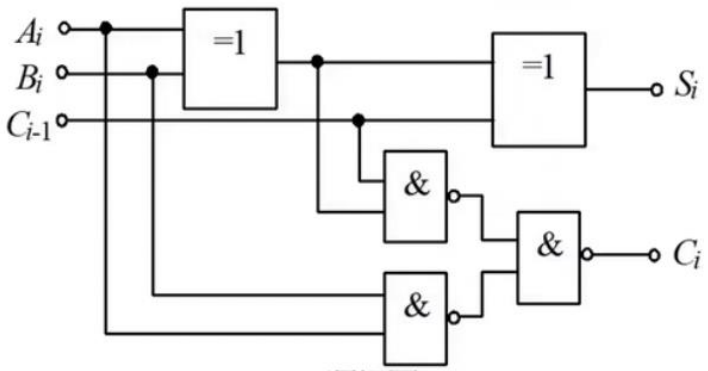
  $=(A_i \oplus B_i) \cdot C_{i-1}+A_iB_i$

加法器（全加器）

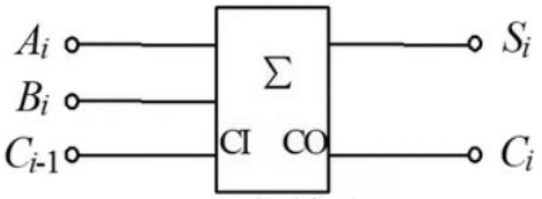
- $S_i=A_i \oplus B_i \oplus C_{i-1}$
- $C_i=\overline{A_i} \cdot B_i \cdot C_{i-1}+A_i \cdot \overline{B_i} \cdot C_{i-1}+A_i B_i \cdot \overline{C_{i-1}}+ A_i \cdot B_i \cdot C_{i-1}$

  $=(\overline{A_i} \cdot B_i+A_i \cdot \overline{B_i}) \cdot C_{i-1}+A_i B_i(\overline{C_{i-1}}+\cdot C_{i-1})$

  $=(A_i \oplus B_i) \cdot C_{i-1}+A_i B_i=\overline{\overline{(Ai \oplus B_i) \cdot C_{i-1}} \cdot \overline{A_i B_i}}$



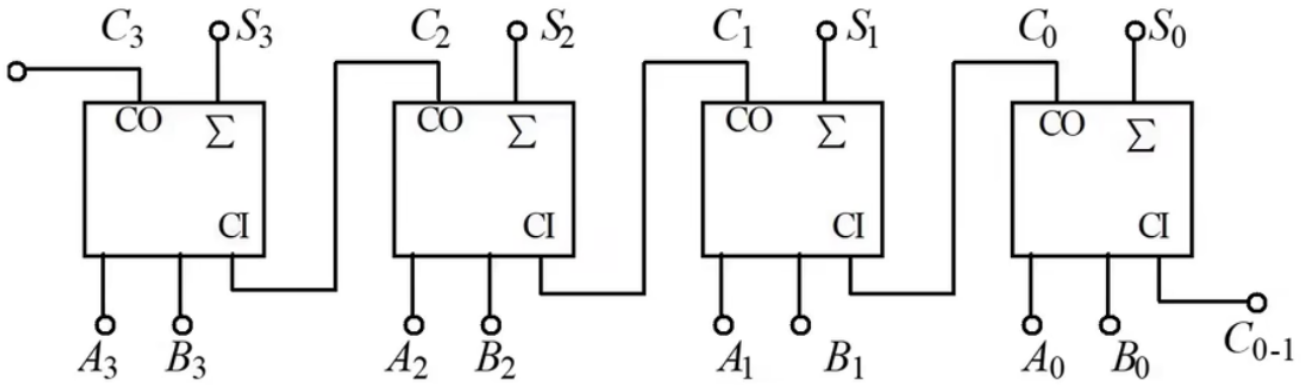
全加器电路图（异或门和与非门）



全加器符号

两个4位二进制数的加法电路

- 以1011+0001=1100为例

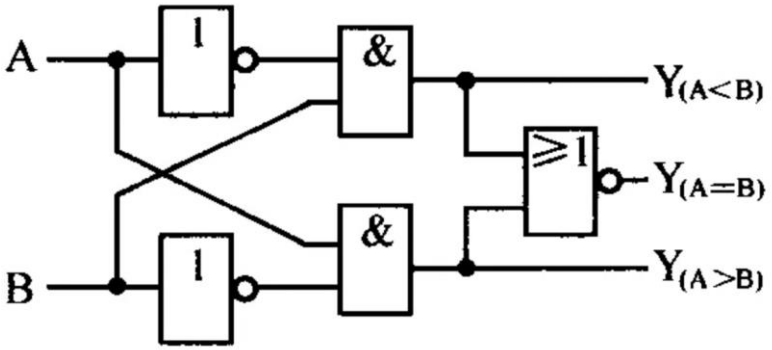


	A_i	B_i	C_{i-1}	S_i	C_i
i=0	1	1	0	0	1
i=1	1	0	1	0	1
i=2	0	0	1	1	0
i=3	1	0	0	1	0

数值比较器

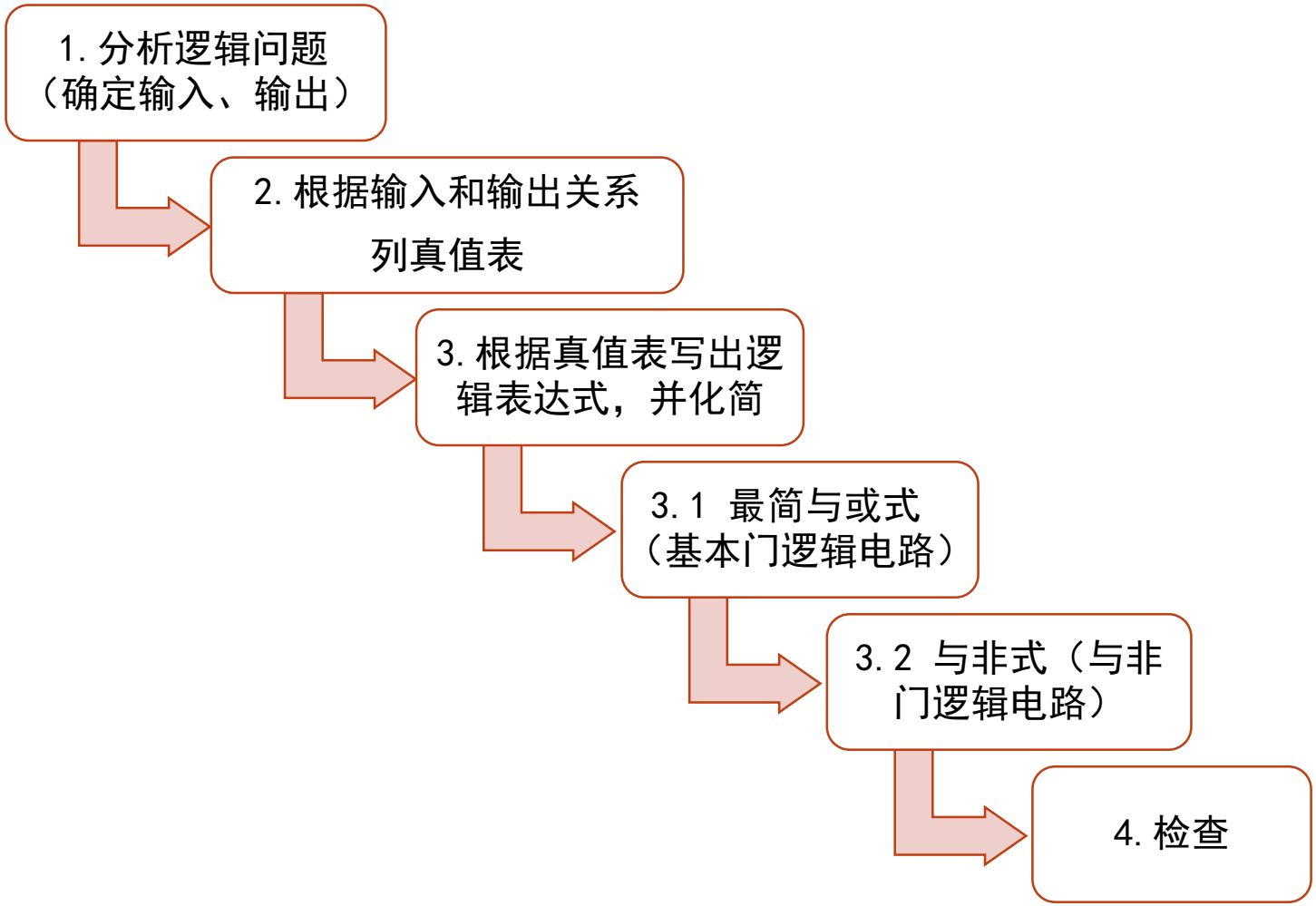
- 对两个1位二进制数进行比较的逻辑电路
  - A>B（A=1 B=0），$A\overline{B}$=1（输出信号$Y_{A>B}$）
  - A<B（A=0 B=1），$\overline{A}B$=1（输出信号$Y_{A<B}$）
  - A=B，$A \odot B$=1（输出信号$Y_{A=B}$）

A	B	$Y_{A>B}$	$Y_{A<B}$	$Y_{A=B}$
1	1	0	0	1
1	0	1	0	0
0	1	0	1	0
0	0	0	0	1

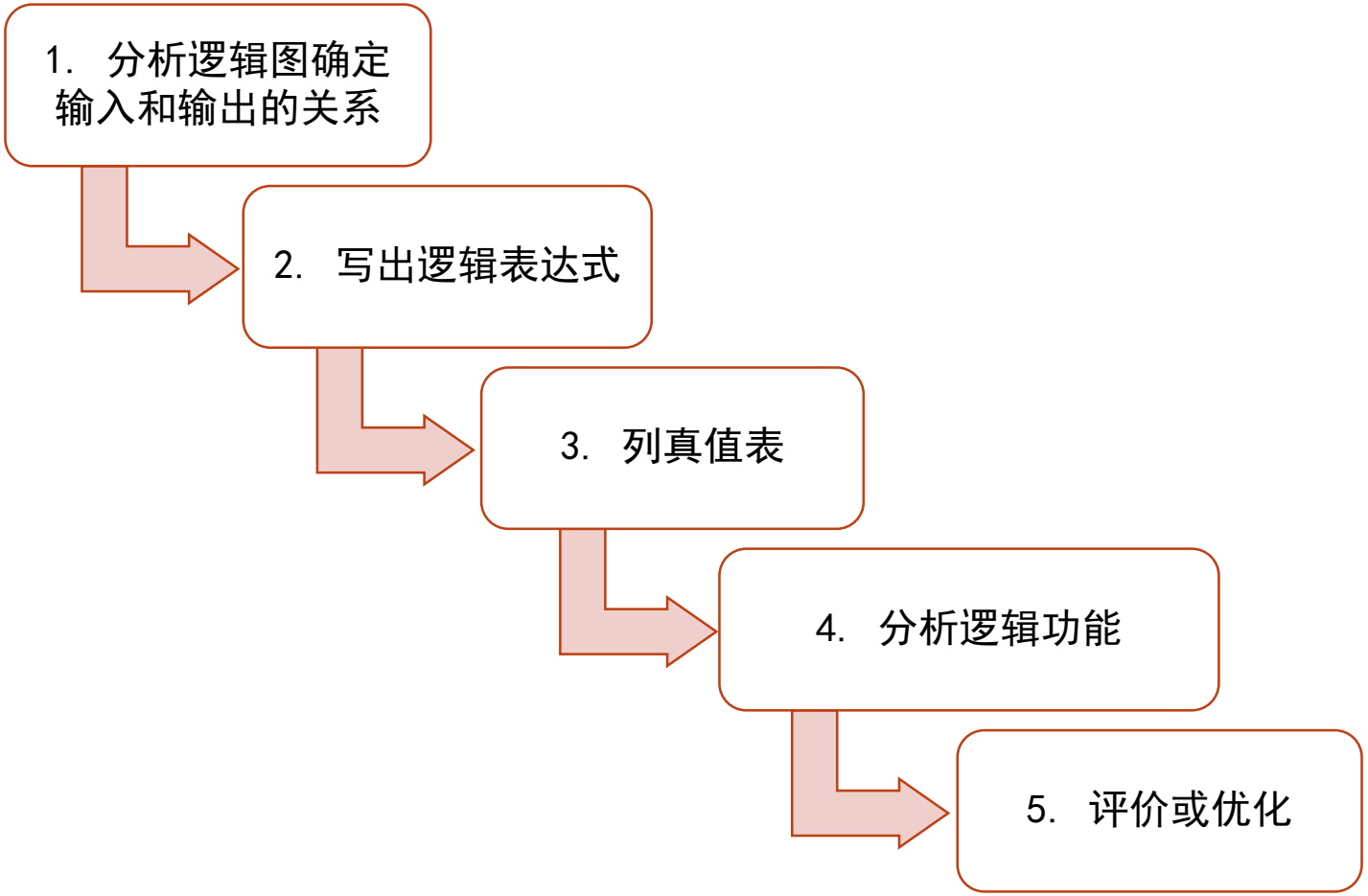


- 那么，多位二进制数比较如何设计呢？

组合逻辑电路设计


1. 分析逻辑问题
（确定输入、输出）
2. 根据输入和输出关系
列真值表
3. 根据真值表写出逻辑表达式，并化简
3. 1 最简与或式
（基本门逻辑电路）
3. 2 与非式（与非门逻辑电路）
4. 检查

组合逻辑电路分析



习题（3）

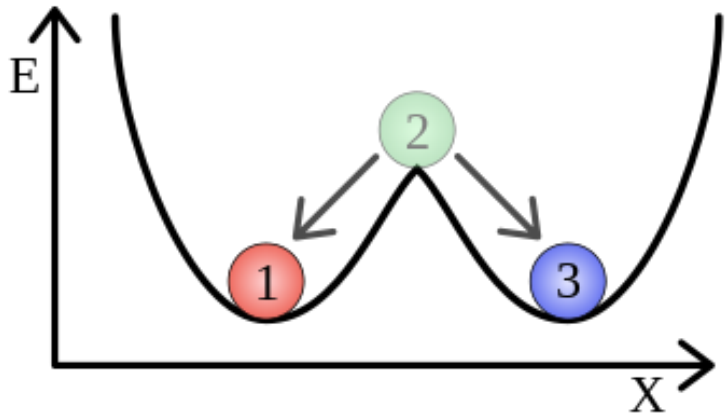
- 书面作业
- 7.4
- 7.5
- 7.7
- 7.8
- 7.9
- 7.10
- 7.11
- 7.12

基本存储元件

- 锁存器
  - R-S锁存器
  - 门控D锁存器
- 触发器
  - 主从D触发器

双稳态电路

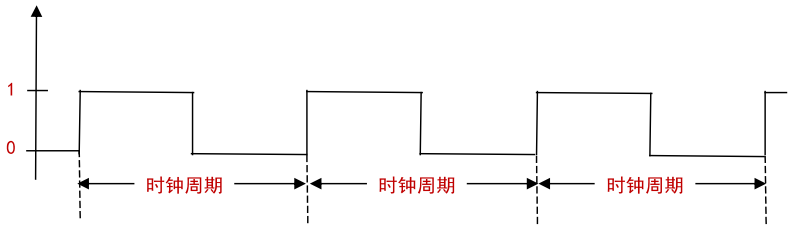
- 双稳态电路：有两个稳定状态，稳定状态的翻转依赖于外加触发信号的作用



基本存储元件

- 存储元件：能存储信息的逻辑结构
- 基本存储元件：存储1位二进制信息的逻辑结构
- 晶体管-门电路-锁存器&触发器-时序逻辑电路-存储器/其他数字处理器
- 锁存器
  - R-S锁存器
  - 门控D锁存器
- 触发器
  - 主从D触发器

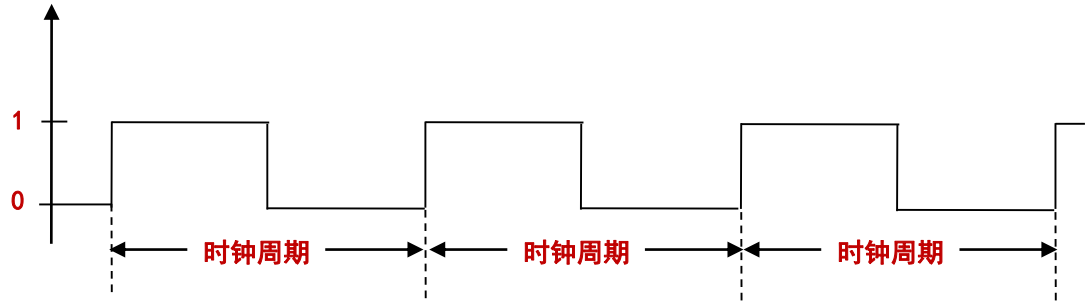
锁存器和触发器



- 共同点：都具有存储功能（1位信息），都是时序逻辑电路的基本逻辑单元
- 锁存器：对脉冲电平（0/1）敏感的存储电路
  - 状态改变取决于特定的脉冲电平值
  - 但当输入信号不稳定，输出会出现毛刺
  - 消耗的门资源相对较少
- 触发器：对脉冲边沿（上升/下降沿）敏感的存储电路
  - 状态只在时钟上升沿或下降沿到来的瞬间改变
  - 不易产生毛刺
  - 消耗的门资源相对于锁存器多

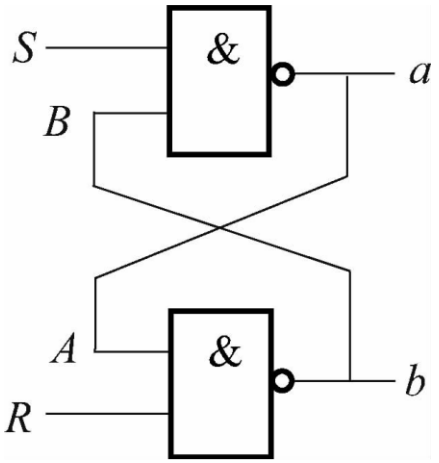
时钟脉冲

- 按一定电压幅度，以一定时间间隔连续发出的脉冲信号
  - 时钟发生器（石英晶体振荡器）
  - 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：重复的时间间隔序列中的一个时间间隔



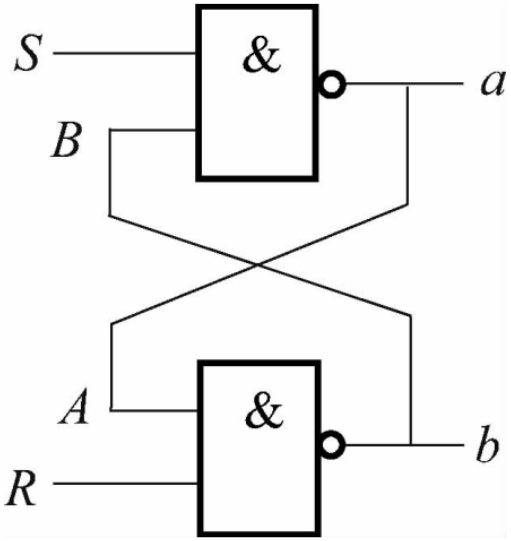
R-S锁存器（或SR锁存器）

- 静态存储单元中最基本的、电路结构最简单
- 两个与非门（或者或非门）组成
- 工作原理：设置R和S来控制电路状态
  - R：“reset”，复位引脚
  - S：“set”，置位引脚
  - 两输出a和b（重点看a）
- 应用：
  - 缓存
  - 数字系统中某些特定标志的设置
  - 消除机械开关触点抖动引起的脉冲输出



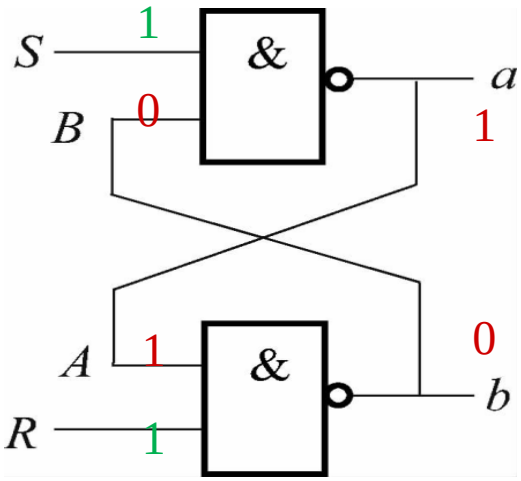
R-S锁存器

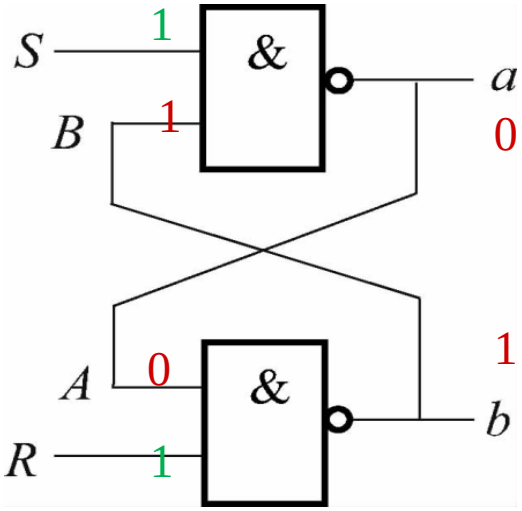
- R = S = 1
  - “保持”状态
- S = 0, R = 1
  - “置位/1”状态
- S = 1, R = 0
  - “复位/置0”状态
- R = S = 0 ×



R-S锁存器

- S=1， R=1（保持状态）

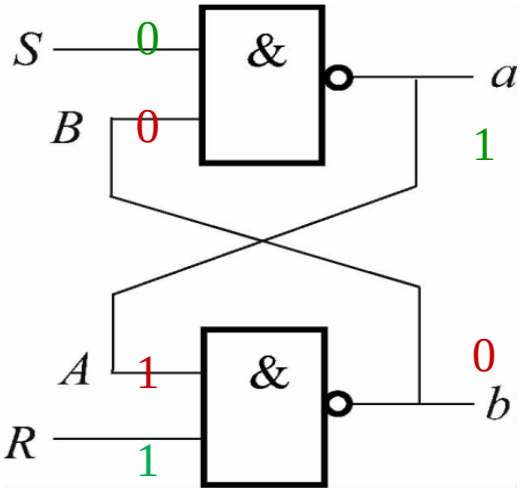




S	R	a	a*
1	1	1	1
1	1	0	0

R-S锁存器

- S=0， R=1（置位/1状态）

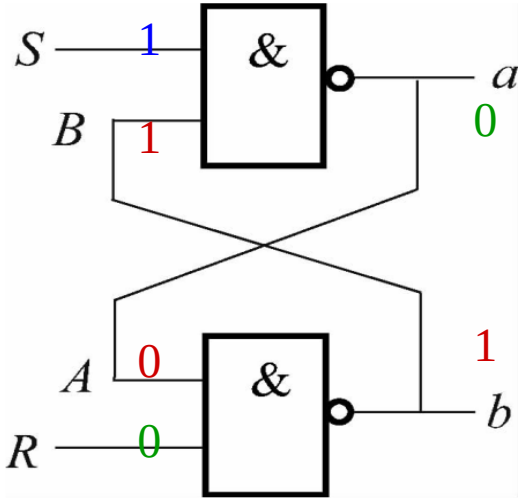


S	R	a	a*
0	1	0	1
0	1	1	1
1	1	1	1

- a的值原本为0，置为1；原本为1，不变

R-S锁存器

- S=1， R=0（复位/置0状态）



S	R	a	a^*
1	0	0	0
1	0	1	0
1	1	0	0

- a的值原本为1，置为0；原本为0，不变

R-S锁存器

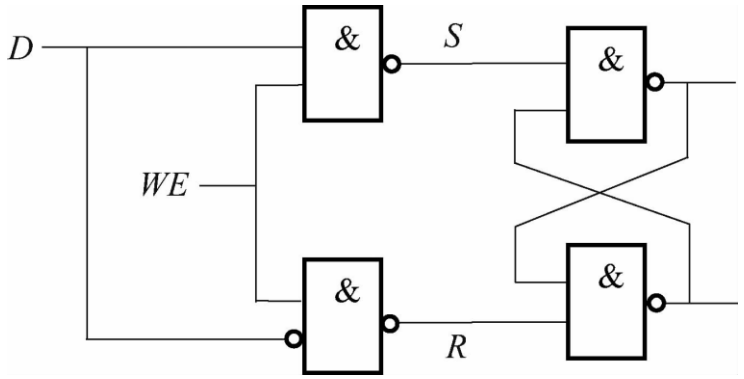
- R = S = 0 ×
  - a、b输出都为0
  - 锁的最后状态取决于组成门的晶体管的电子特性，而不是取决于被操作的逻辑值
  - 是逻辑电路“延迟”造成的

R-S锁存器

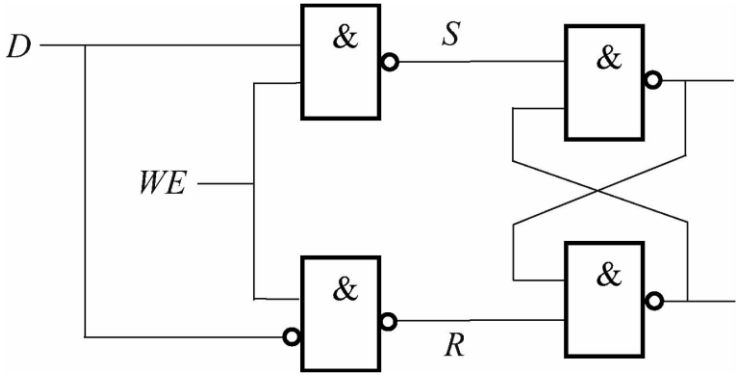
S	R	a	a*
1	1	1	1
1	1	0	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
0	0	0	不定
0	0	1	不定

门控D锁存器

- 输入控制门电路（两个与非门）+ R-S锁存器
- 两个输入端：D(Data) 和 WE(Write Enable)
  - 控制何时置位、何时复位、何时保持
  - 工作中不存在非定义状态，即保证S、R不同时为0
  - WE = 1, 输出 = D
    - S = NOT(D), R = D
    - D=0, S=1, R=0, 置0
    - D=1, S=0, R=1, 置1
  - WE = 0, 保持
    - S = R = 1，输出状态保持不变



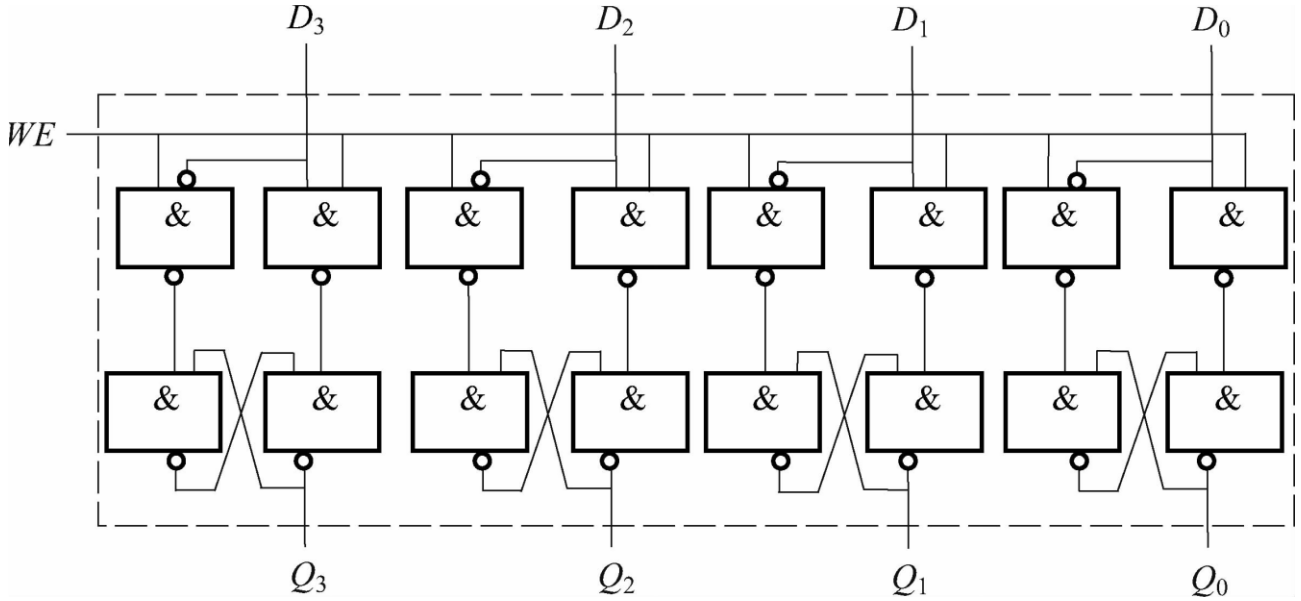
门控D锁存器



WE	D	S	R	a	a*
0	X	1	1	0	0
0	X	1	1	1	1
1	0	1	0	0	0
1	0	1	0	1	0
1	1	0	1	0	1
1	1	0	1	1	1

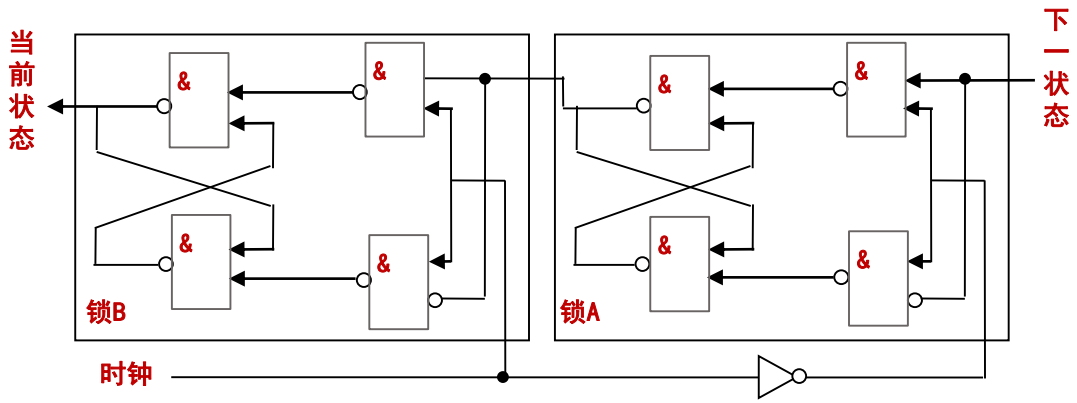
门控D锁存器

- 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



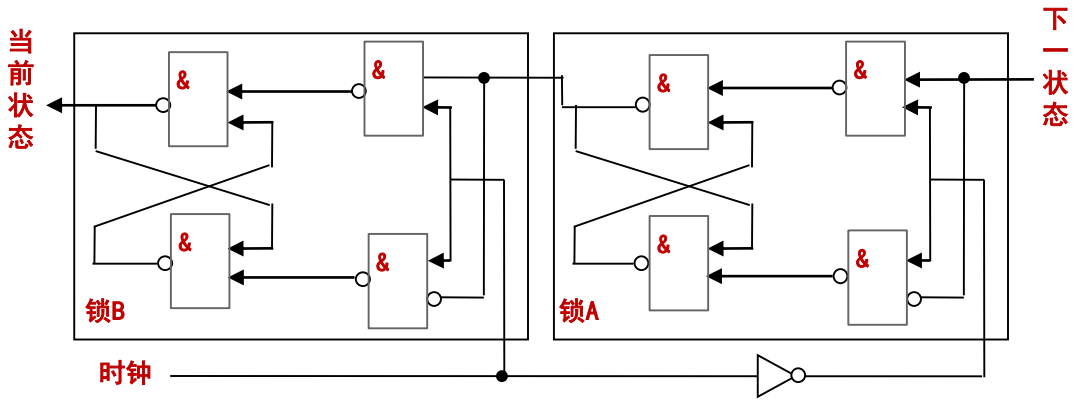
主从D触发器

- 主从触发器，由2个门控D锁存器构成
- A为主锁存器，B为从锁存器，使能信号相反
- 实现存储数据和输入信号之间的隔离



主从D触发器

- 时钟信号为0，WE_A=1，WE_B=0，A锁存器原样输出数据D的值（门控D锁存器性质），B锁存器保持原来状态的值，输出不变
- 时钟信号由0变1瞬间（上升沿），WE_A=0，WE_B=1，A之前锁存的D值保持不变，B锁的输出相应变为A中锁存的D值
- 时钟信号为1，A锁保持，B锁输出也不变



习题（4）

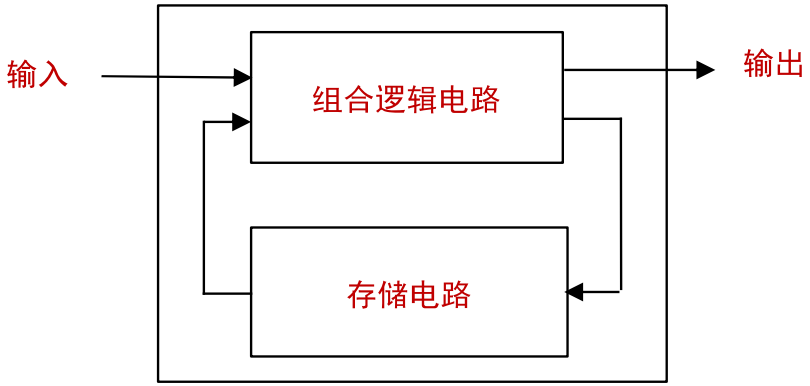
- 书面作业
- 7. 13

时序逻辑电路

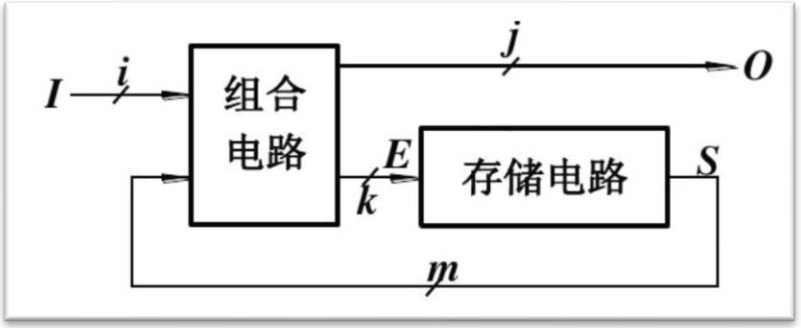
- 基本介绍
- 有限状态机
- 时序逻辑电路设计

时序逻辑电路

- 既能处理信息，也能存储信息
- 任何时刻的电路输出不仅取决于当前时刻输入，还取决于电路原来的状态
- 由**组合逻辑电路**和**存储电路**组成
  - **存储电路**必不可少，由锁存器或触发器组成

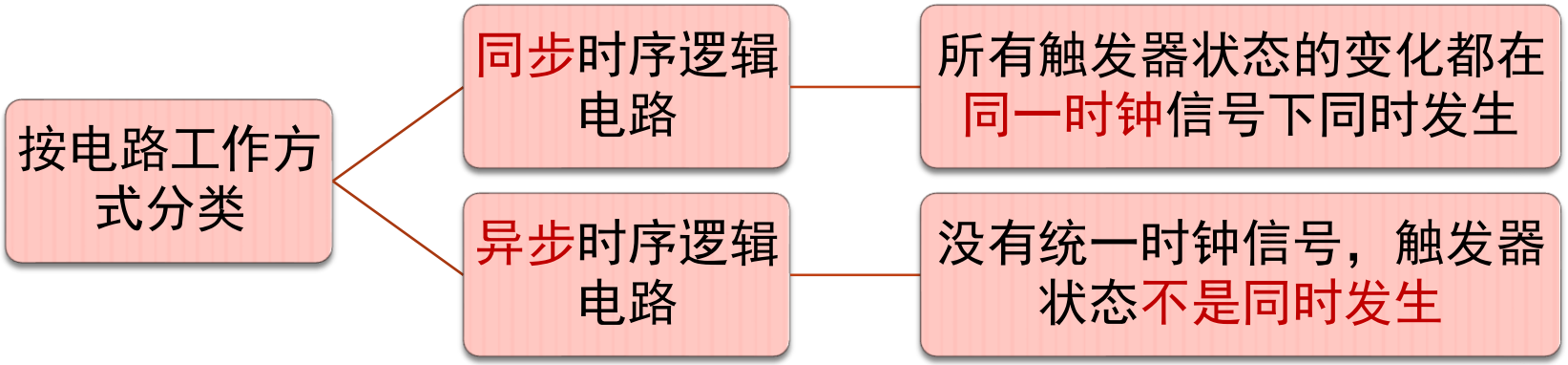


时序逻辑电路



- $I(I_1, \ldots, Ii)$：输入信号
- $O(01, \ldots, Oj)$：输出信号
- $E(E_1, \ldots, Ek)$：存储电路的激励或驱动信号
- $S(S_1, \ldots, Sm)$：存储电路的状态信号/变量，现态
- 输入、 输出及存储电路信号之间的逻辑关系：
  - $E = \mathrm{f}(I, S)$ —激励/驱动方程
  - $S^{n+1} = \mathrm{g}(\mathrm{E}, \mathrm{Sn})$ —状态转换方程
  - $O = h(I, S)$ —输出方程

时序逻辑电路

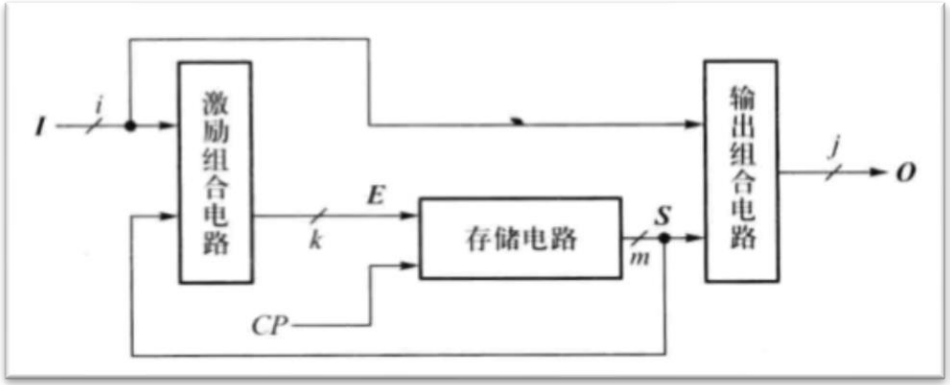


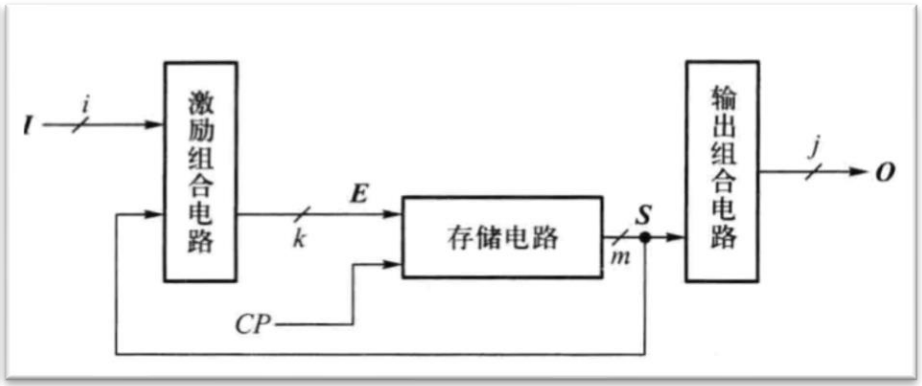
- 同步时序电路一般由触发器组成，应用较广泛
  - 寄存器、计数器、序列信号发生器等
- 异步时序电路可由锁存器或触发器组成
  - 电平异步时序电路：锁存器
  - 脉冲异步时序电路：触发器

时序逻辑电路

按输出信号特点分类

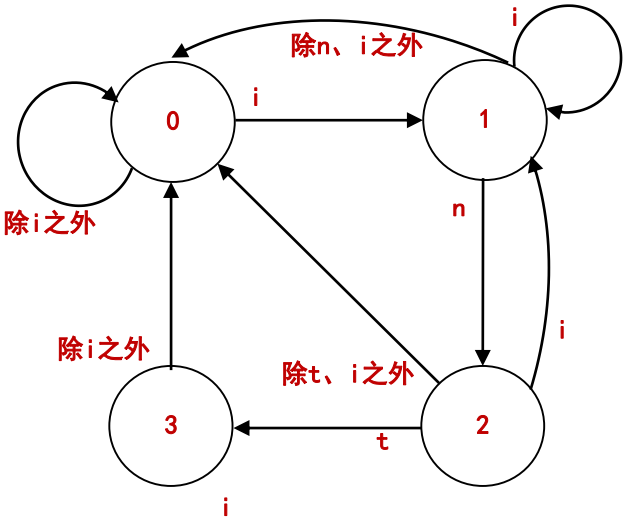
- Mealy型（下图左）
  - 输出状态与存储电路的状态和外部输入均有关 $O = h(I, S)$
- Moore型（下图右）
  - 输出状态只与存储电路的状态有关，与外部输入无关 $O = h(S)$





有限状态机

- 实现被称为**有限状态机**的机制
  - **有限个状态**以及**状态之间的转移和动作**等行为的数学模型
  - 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**
  - 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，如交通信号灯控制器



有限状态机

- 寄存器容量是有限的，状态的数目是有限的
- 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
- 可以用时序逻辑电路来实现

状态

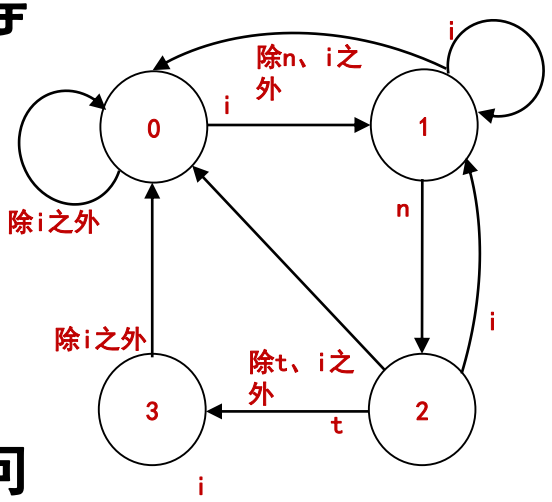
- 示例一：电话应答机
- 可以根据响铃的次数（如3次），决定是否开启录音机录音
- 电话应答机的4个状态：
  - A. 不开启录音机，还未响铃；
  - B. 不开启录音机，但已响铃1次；
  - C. 不开启录音机，但已响铃2次；
  - D. 开启录音机。
- 这4种情况分别被标记为A、B、C和D，每一种情况都被称为应答机的一种状态

状态

- 示例二：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

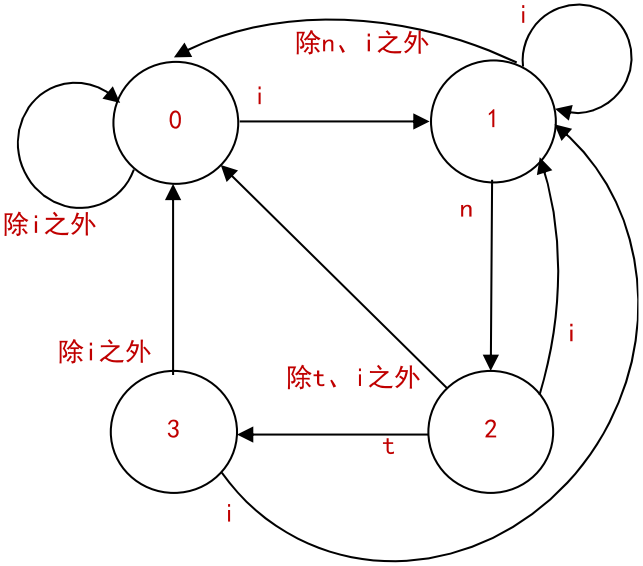
状态图

- 描述有限状态机的方法之一
  - 其他：逻辑函数表达式、状态表等
- 一组圆：对应状态
- 一组有向弧线
  - 每一条弧线确定一个状态的转换
  - 箭头表示状态的转换方向
  - 剪头起始状态称为当前状态，指向的状态称为下一状态/次态



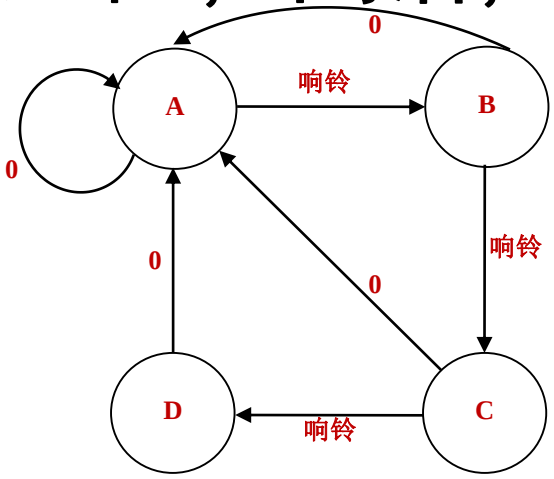
状态图

- 计算“int”字符串出现次数
- 4个状态，10个状态转换
  - 外部输入是读到的字符
  - 次态由当前状态和当前的外部输入的组合决定
  - 系统的输出为计数器的值：
    - 当系统状态是0、1和2时，计数器不变；
    - 当系统状态是3时，计数器加1



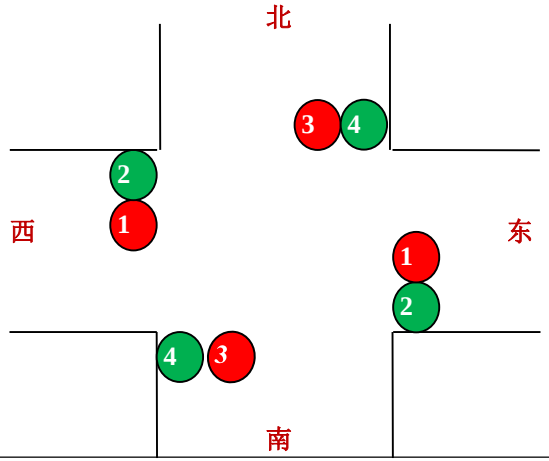
状态图

- 电话应答机
- 外部输入是响铃，0表示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



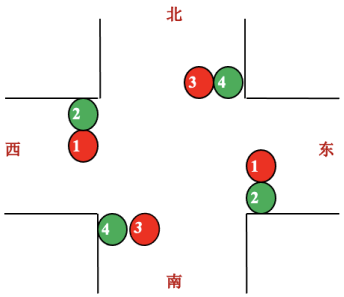
示例：交通灯控制器

- 东西向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）
  - 相同颜色的灯互为对角线
- 通行按钮，行人按下控制红灯亮



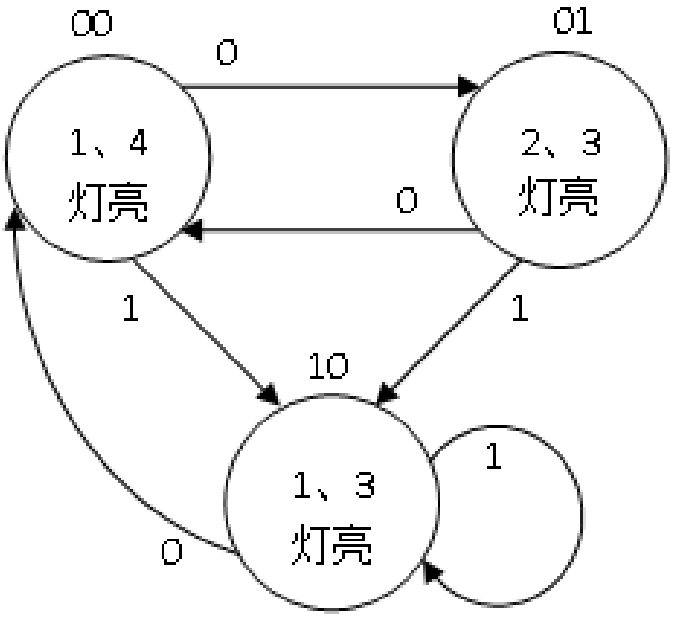
一、问题分析

- 当没有行人时
  - 前一时钟周期，1红、4绿灯亮；
  - 下一时钟周期，3红、2绿灯亮；
  - 重复这个顺序……
- 当有行人按下通行按钮时，
  - 当前时钟周期结束时，1、3号红灯亮
  - 保持一个时钟周期
  - 下一周期，1红、4绿号灯亮
  - 重复……



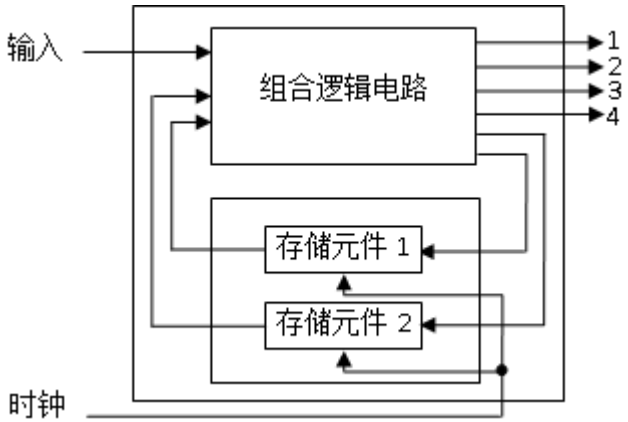
二、状态图

- 状态图共有3个状态：00/01/10
  - 无行人按通行按钮（用0表示）
    - 按时钟周期从状态00转换到状态01
    - 再转换到状态00
    - 重复进行
  - 有行人按下按钮（用1表示）
    - 当前时钟周期结束，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



三、时序逻辑问题

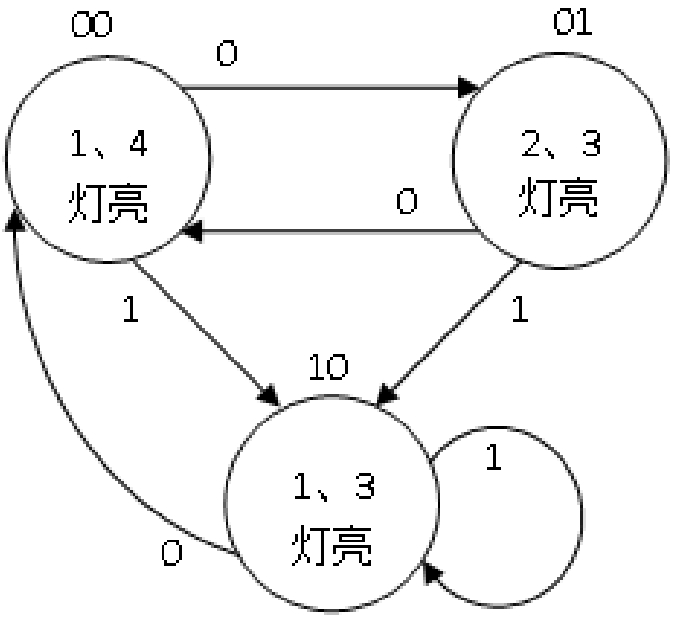
- 1个输入：行人的按钮行为
- 4个输出：控制1、2、3和4号灯何时亮
- 寄存器：2个存储元件，记录控制器处于哪一个状态（00，01，10），由交通信号的过去的行为决定的（存储电路的状态）
- 时钟：使状态转换每隔0.5分钟即可发生



四、真值表

- 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]），也就是次态
  - 由当前状态和当前的外部输入的组合决定

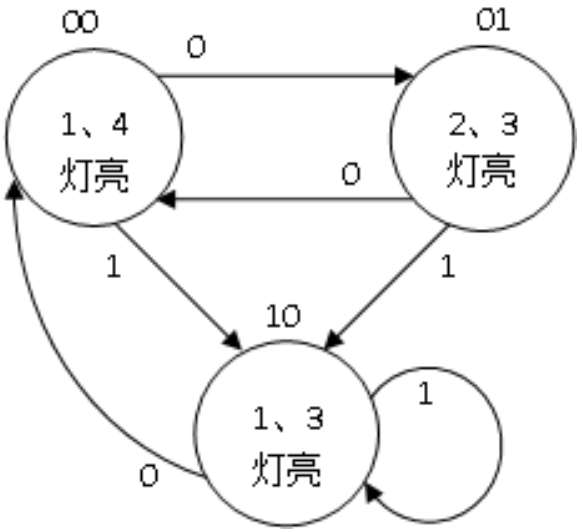
四、真值表



- 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

四、真值表



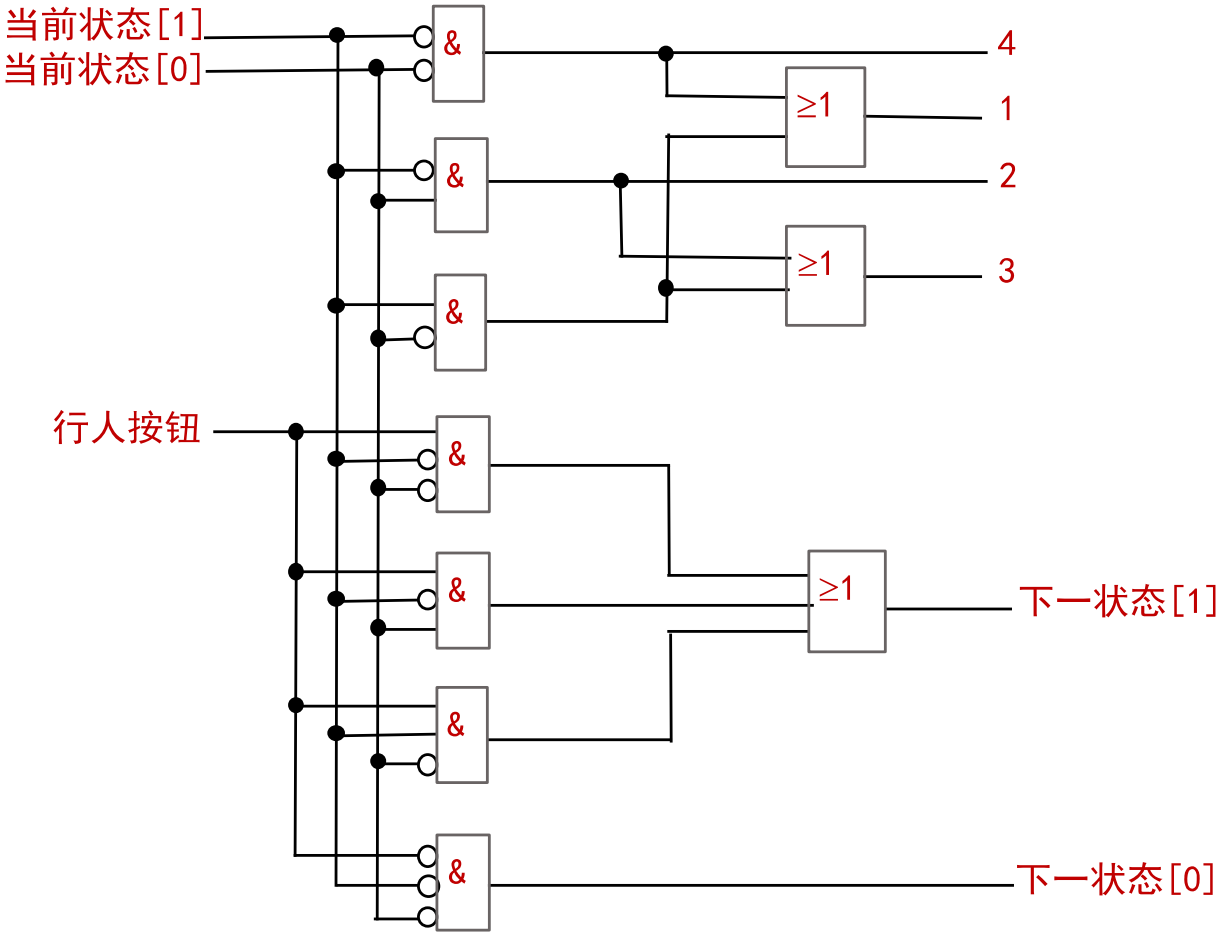
- 下一状态/次态表
  - 由当前状态和当前的外部输入的组合决定

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

五、组合逻辑电路

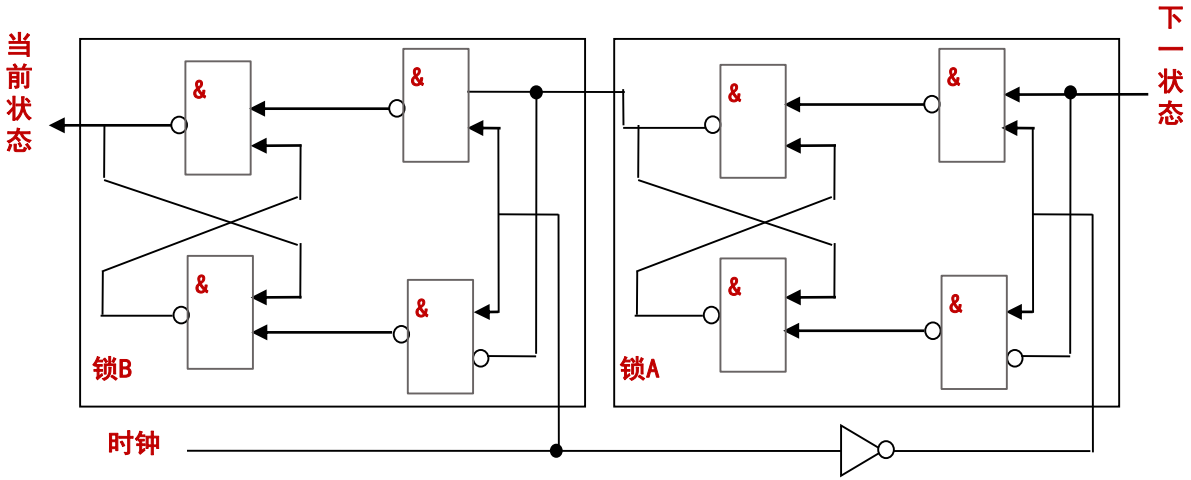
[1]	[0]	1	2	3	4
0	0	1	0	0	1
0	1	0	1	1	0
1	0	1	0	1	0

行人	当前[1]	当前[0]	次态[1]	次态[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



六、存储电路

- 使用：主从D触发器
- 注意：寄存器此处不能由门控D锁存器组成
- 原因：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



习题（5）

- 上机作业
- 7. 17