

计算机系统 I

第三章

Digital Logic Structures

数字逻辑基础



晶体管：构成计算机的最基本单元

- } 微处理器由成千上百万个晶体管组成
 - Intel Pentium 4 (2000): 48 million
 - IBM PowerPC 750FX (2002): 38 million
 - IBM/Apple PowerPC G5 (2003): 58 million

层次：

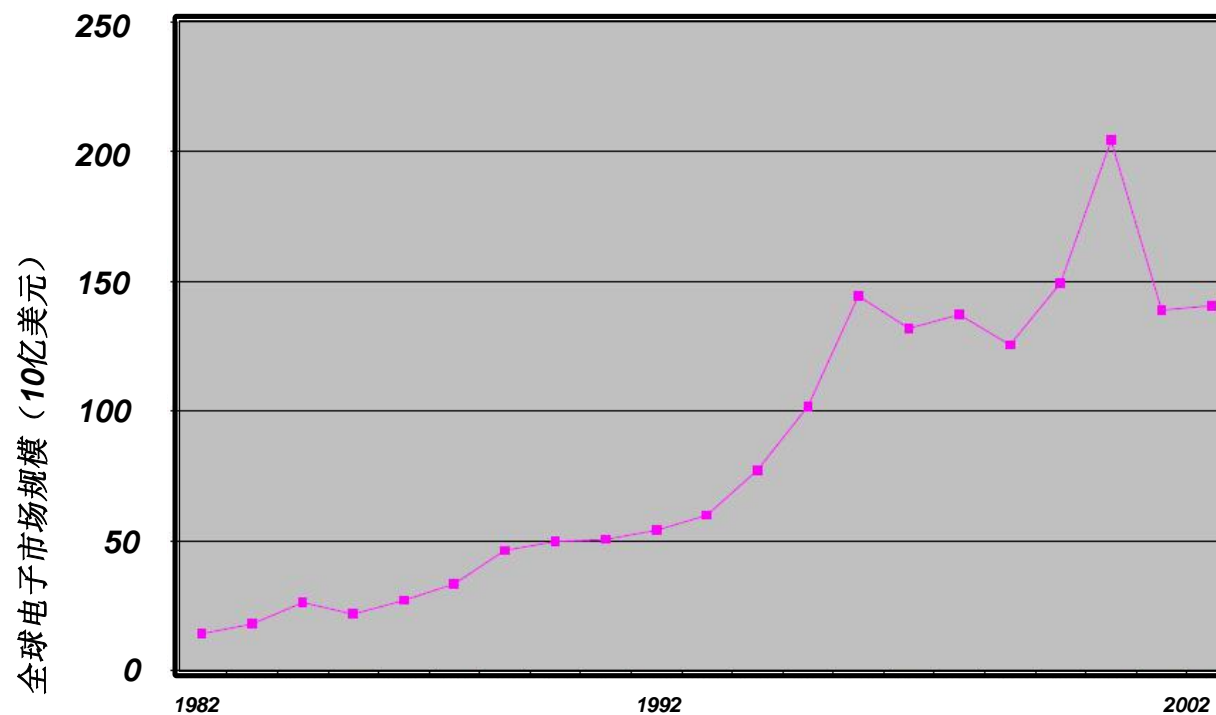
- } 晶体管 **à** 开关(0, 1)
- } 开关 **à** 逻辑门
 - AND, OR, NOT
- } 逻辑门 **à** 逻辑电路
 - Adder, multiplexer, decoder, register, ...
- } 逻辑电路 **à** 微处理器
 - LC-3

CPU与半导体、逻辑门

- } 计算机-CPU发展历史;
- } 摩尔定律(Moore's Law)及局限性;
- } 现有瓶颈及发展趋势;

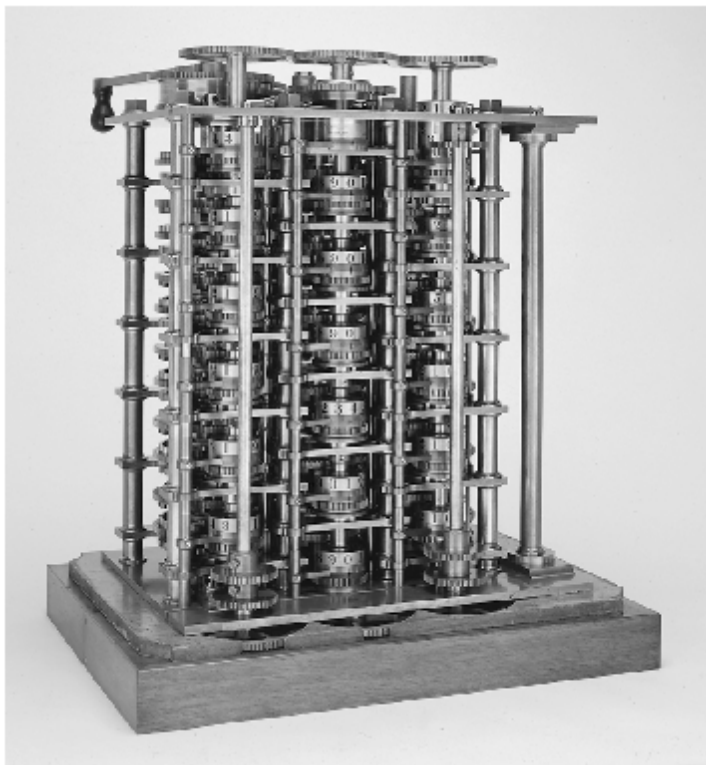


全球半导体市场



- 全球半导体市场发展迅速 (20年间扩展~8倍) ;
- 2000年左右达到~2000亿美元

机械计算器



**Babbage 的机械计算机
(Difference Machine)**

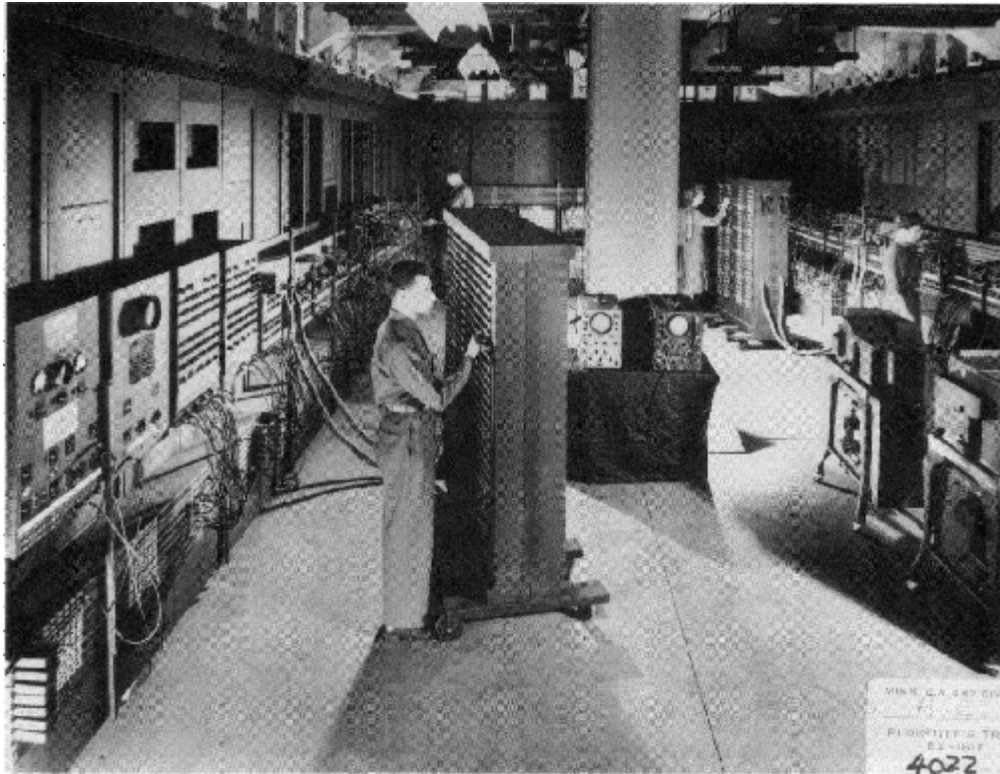
- 十九世纪初（~1834）诞生；
- 采用十进制进位；
- 可进行加减乘除运算；
- 存贮，计算两步处理结构；
- 为了提高计算速度，采用pipeline结构。

- 非常复杂，~25000 部件；
- 非常昂贵，~17470英镑（1834年）

**1900-2000 通货膨胀 ~1000

From : A Century of Change: Trends in UK statistics since 1900

第一台电子计算机

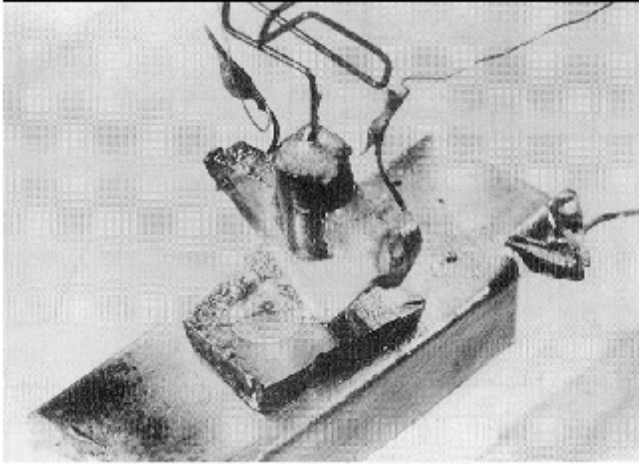


Electronic Numerical Integrator And Computer (ENIAC)

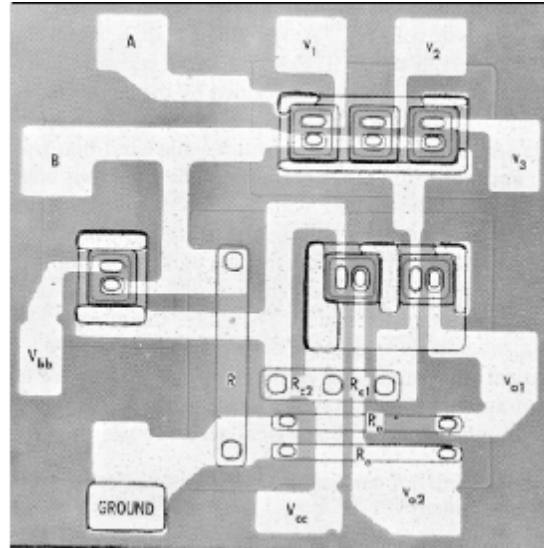
- } 1946年完成;
- } 24米长, 2.6米宽;
- } 18000个真空管

- 可靠性不能保证;
- 不经济 (大量的能源消耗)

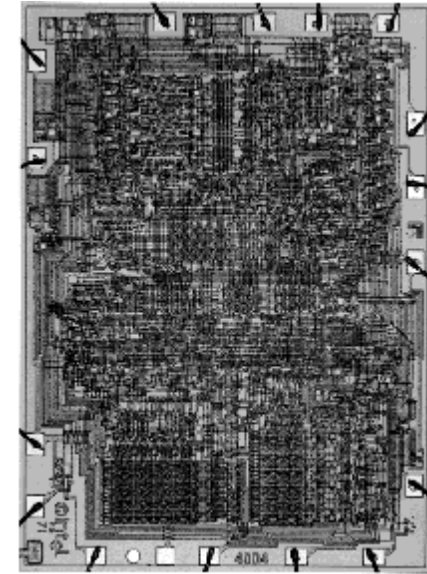
晶体管集成电路



第一个半导体晶体管, Bell Lab, 1948



第一片集成电路, 射机耦合逻辑门 (ECL), Motorola, 1966



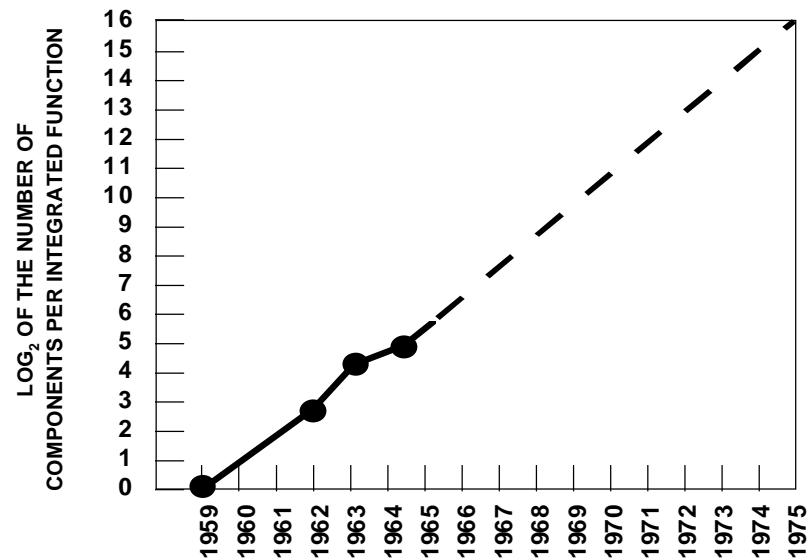
Intel 4004 (1971)

- } 半导体晶体管的发明使集成电路成为可能；
- } 第一片集成电路的成功，是大规模集成电路的开端；
- } 第一个多功能处理器：1000个晶体管，1MHz主频。

摩尔定律(Moore's Law)

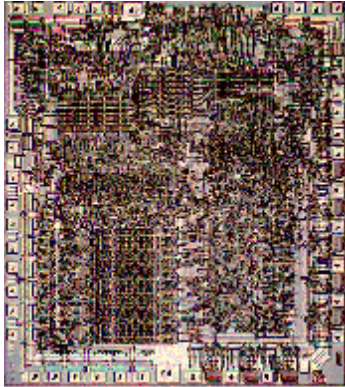
} Gordon Moore, 1965

- 在芯片上集成的晶体管每18-24个月增加一倍。

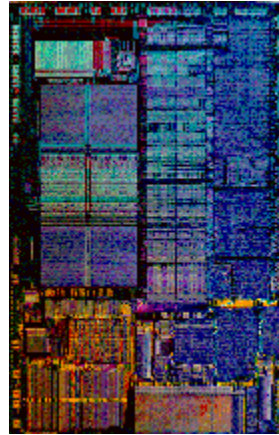


1965年的预测

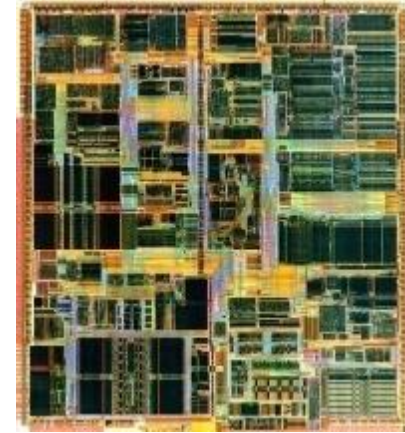
CPU发展历程 (Road map)



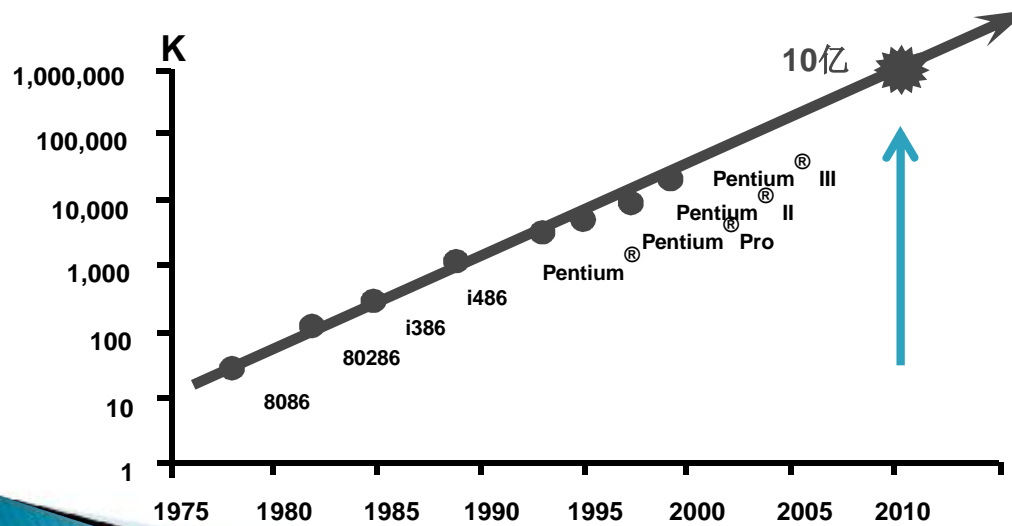
Intel 8080A, 1974
6K 晶体管, 6 μ m



Intel 486, 1989, 81mm²
1.2M 晶体管, 0.8 μ m



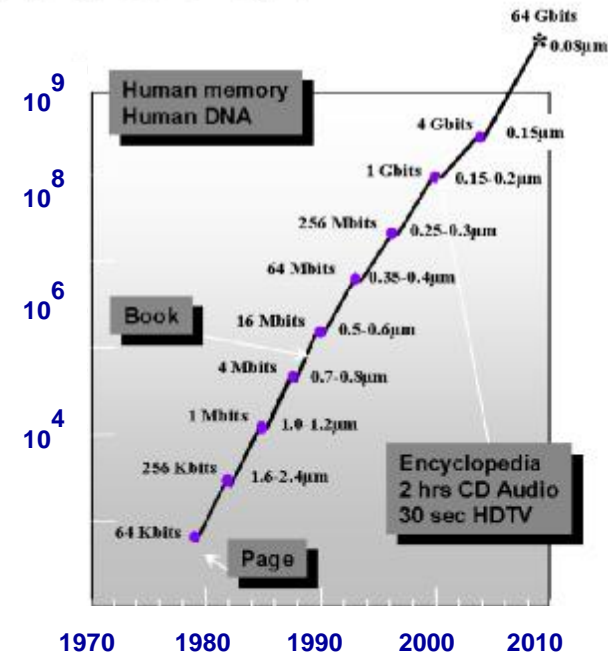
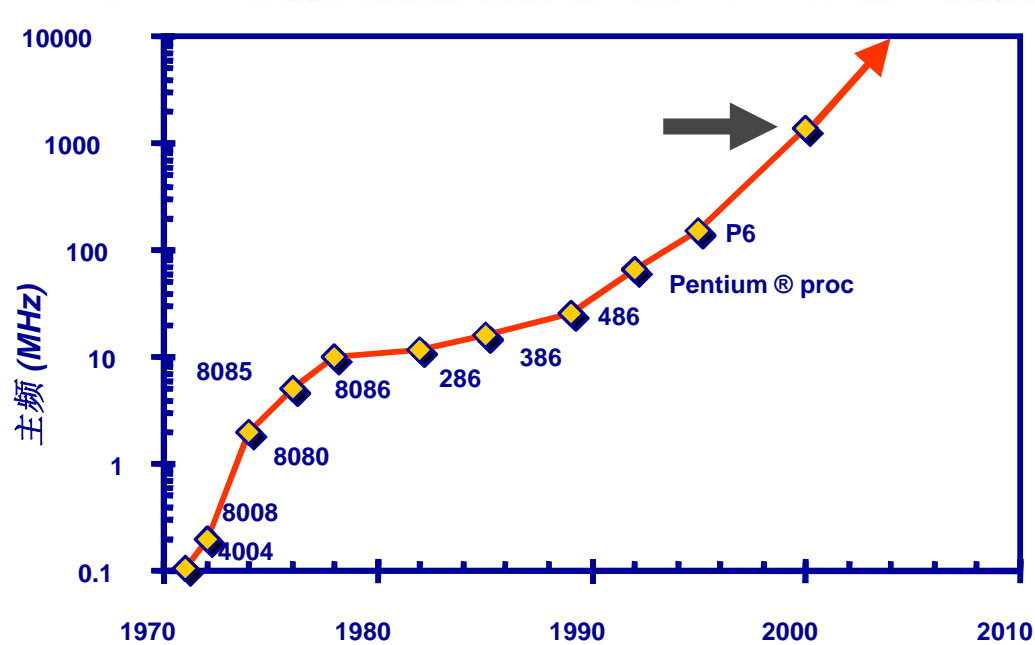
Intel Pentium II, 1997, 203mm²,
7.5M 晶体管, 0.25 μ m



- Moore's Law 准确预言了集成电路的发展;
- 集成度在今年有望达到10亿/芯片。

[**http://www.intel.com/intel/intelis/museum/exhibit/hist_micro/hof/hof_main.htm](http://www.intel.com/intel/intelis/museum/exhibit/hist_micro/hof/hof_main.htm)

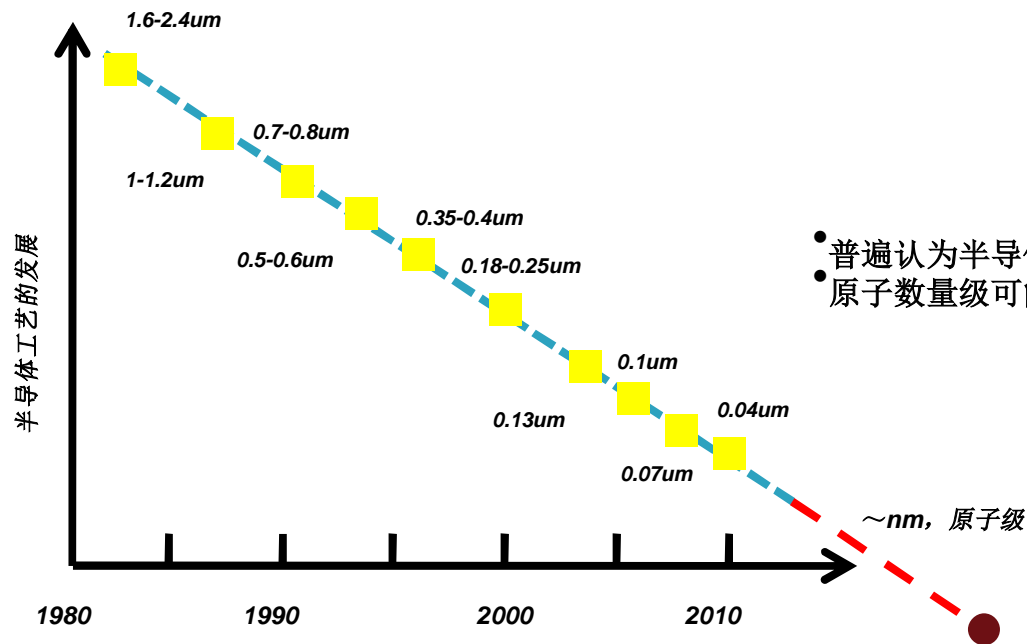
CPU发展历程（性能的提高）



- } 主流多功能处理芯片的工作频率每两年增加一倍；
- } 可集成的信息量以几何级数增加。

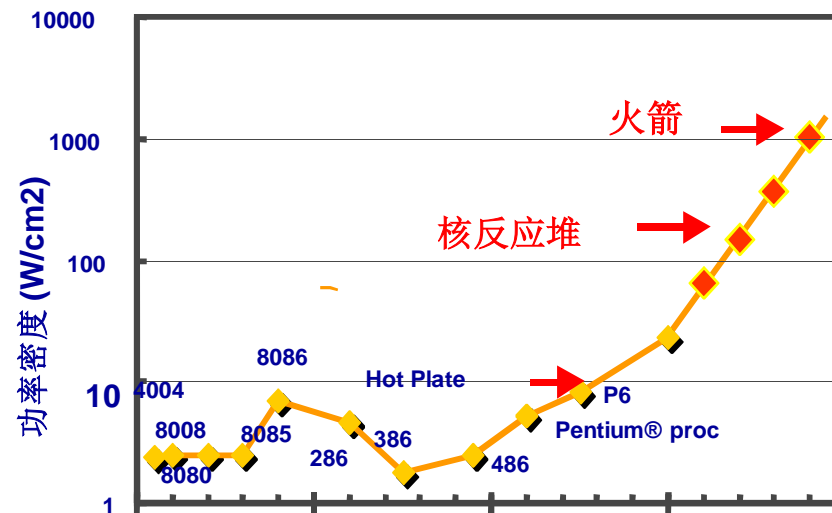
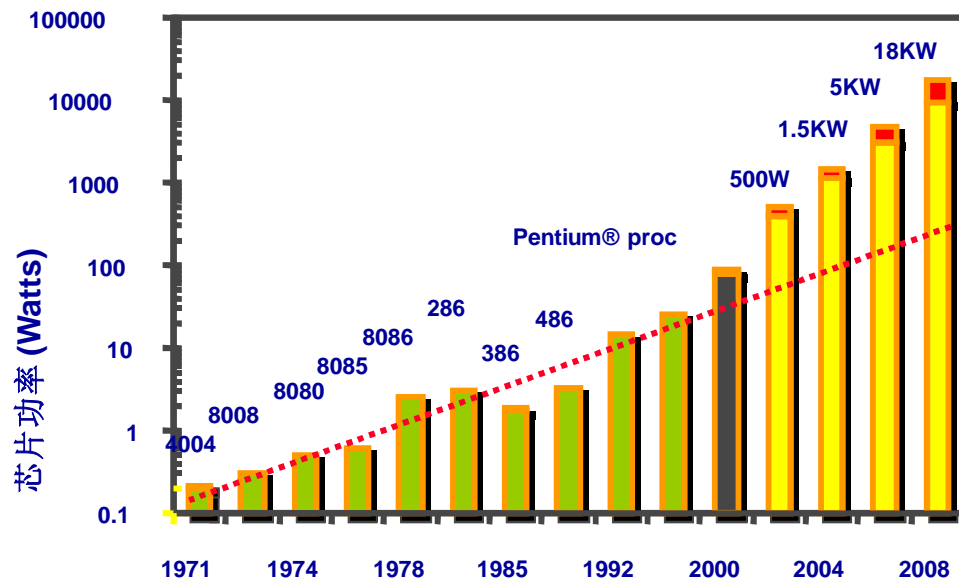
摩尔定律的局限性-scale down

- 半导体制作工艺每两年换代一次，以集成更多的器件；
- 每一代器件长度为上一代的 ~ 0.7 ；
- 同样芯片面积可以多容纳一倍的器件。



- 普遍认为半导体工艺可以在发展几代；
- 原子数量级可能是先有制作方法的终结。

瓶颈-能量密度

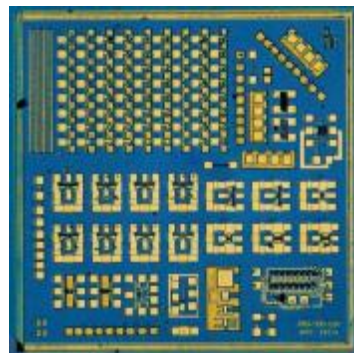


- 更多的集成器件，更快的运算速度，所需功率也成几何级数增加；
- 在极高的温度下，半导体器件不能正常工作。

发展趋势

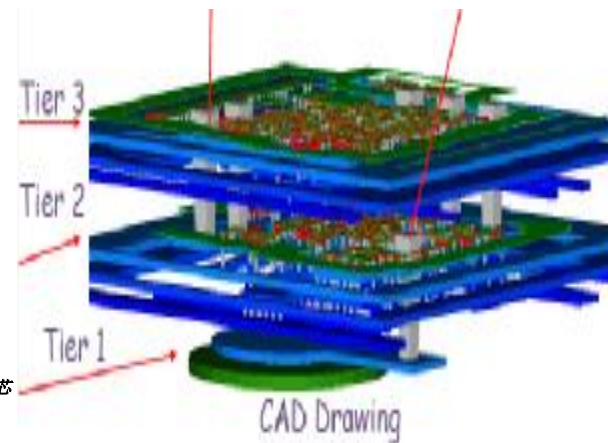
} 新器件和工艺

- 新材料，如III-V族半导体；
- 新工艺，如3D架构。



磷化铟镓锗 (InAaGsP) 制作的逻辑门，比目前最快的硅芯片快 ~2.5倍

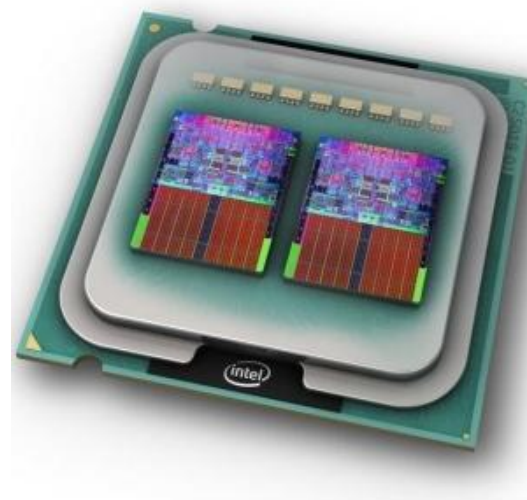
MIT, Microsystems Technology Laboratories



3D生产工艺

● 系统与电路设计

- 采用新的电路架构，如并行处理。



Intel 四核CPU

最基本开关单元:MOS晶体管

1)双极型集成电路（双载子：空穴和电子同时参与导电）以通常的**NPN或PNP型双极型晶体管**为基础的单片集成电路。它是1958年世界上最早制成的集成电路。双极型集成电路主要以硅材料为衬底，在平面工艺基础上采用埋层工艺和隔离技术，以双极型晶体管为基础元件。

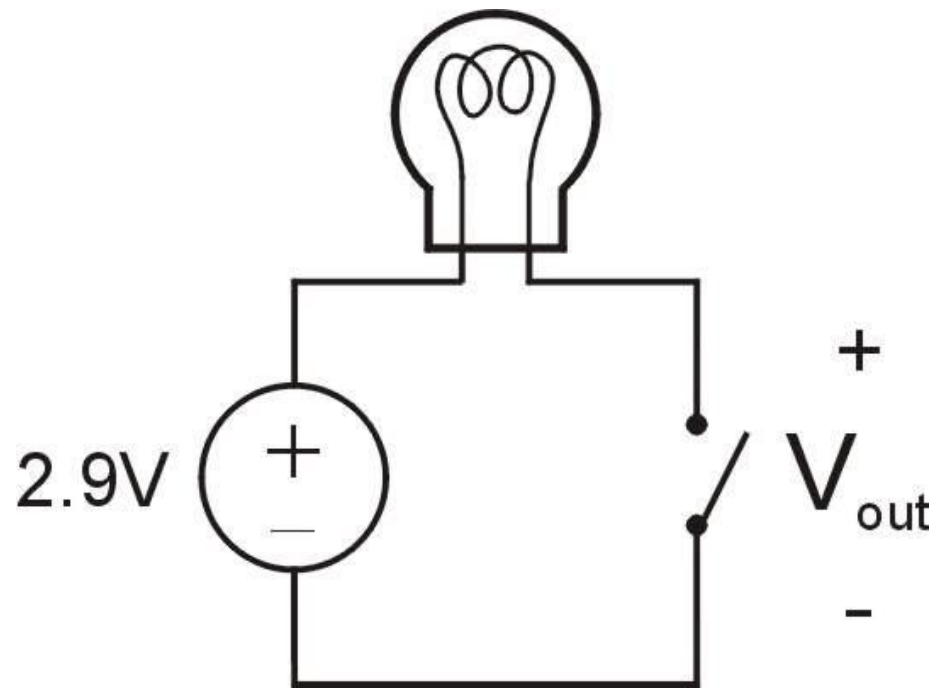
2)单极型集成电路（金属氧化半导体**MOS晶体管**）

只有一种载流子（多数载流子）参与导电。

1963年，仙童半导体（Fairchild Semiconductor）的Frank Wanlass发明了MOS电路。到了1968年，美国无线电公司（RCA）一个由亚伯·梅德温（Albert Medwin）领导的研究团队成功研发出第一个MOS集成电路（Integrated Circuit）。经过长期的研究与改良，今日的MOS元件无论在使用的面积、操作的速度、耗损的功率，以及制造的成本上，都比另外一种主流的半导体制程双极型集成电路要有优势。

MOS有**PMOS和NMOS**之分，对应的分别是PMOS电路和NMOS电路，主流的由两者共同做成的互补型电路（CMOS）。

工作原理:简单的开关电路



} 开关 **打开**:

- 电路无电流流过
- Light is **off**
- V_{out} is **+2.9V**

} 开关 **闭合**:

- 电流通过开关
- Light is **on**
- V_{out} is **0V**

基于开关的电路能简单的表征两个状态:
on/off, open/closed, voltage/no voltage.

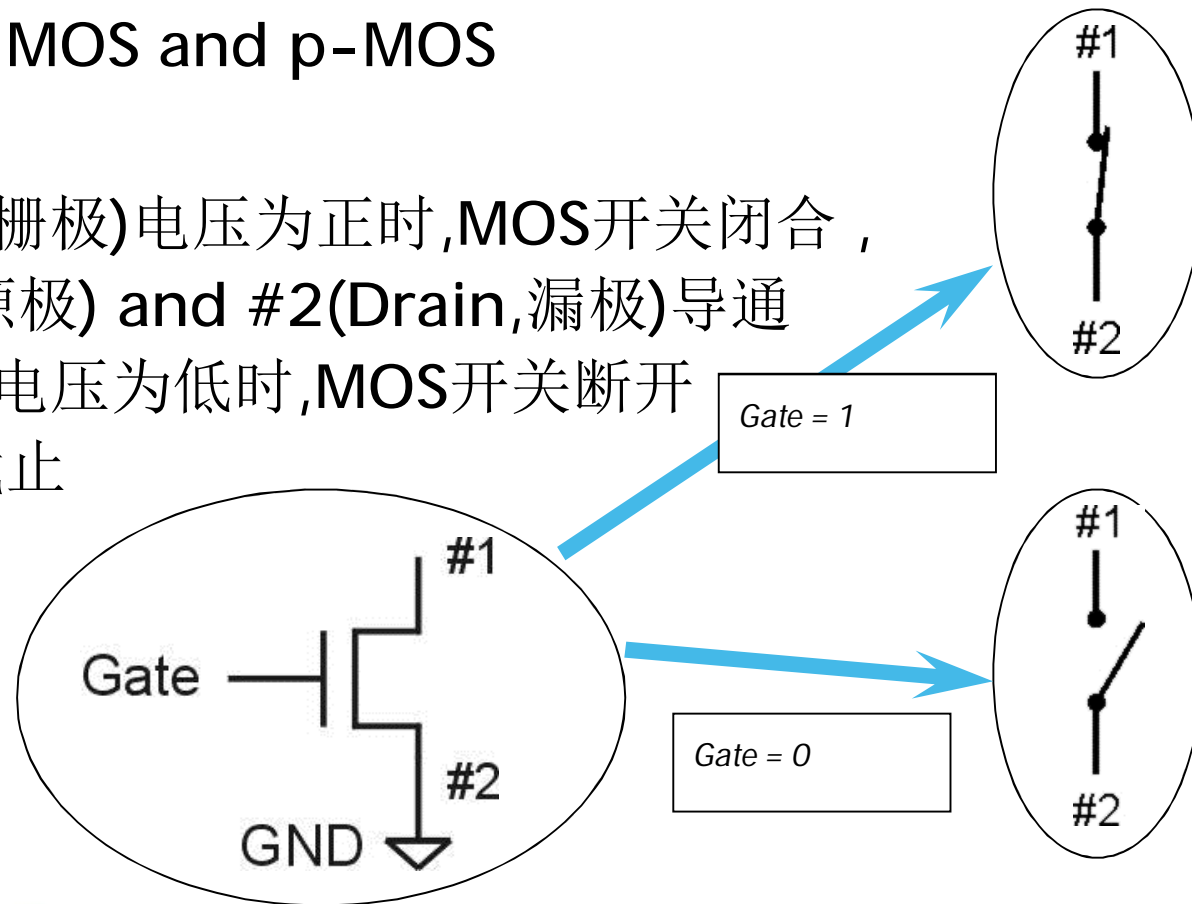
N型金属氧化半导体管 (NMOS)

} MOS = Metal Oxide Semiconductor(金属氧化半导体)

- 两种类型: n- MOS and p-MOS

} **n-MOS**

- 控制门(Gate,栅极)电压为正时,MOS开关闭合 ,
#1(Source,源极) and #2(Drain,漏极)导通
- 控制门(Gate)电压为低时,MOS开关断开
#1 and #2截止

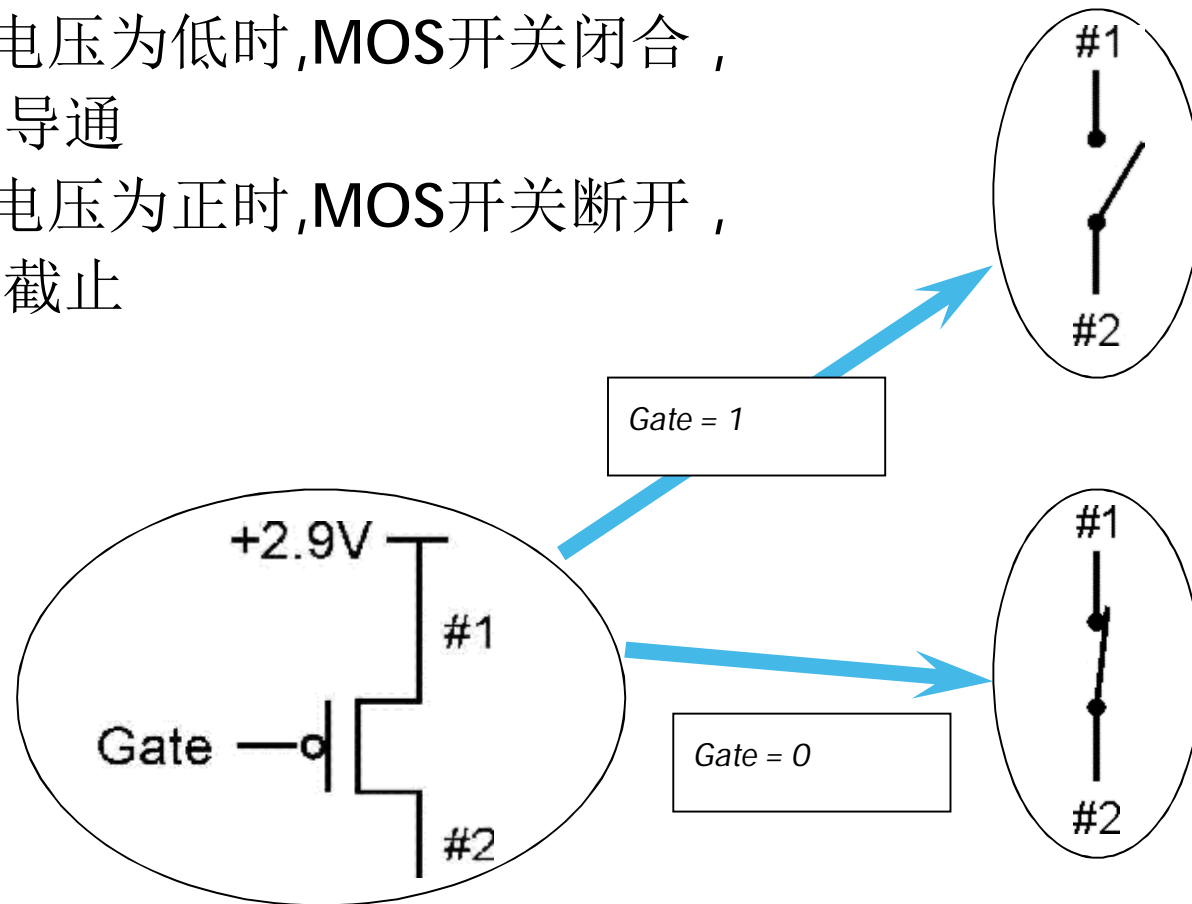


Terminal #2 must be connected to GND (0V)

P型金属氧化半导体管 (PMOS)

} p-MOS 工作机制和 n-MOS互补(相反)

- 门极(Gate)电压为低时,MOS开关闭合 ,
#1 and #2导通
- 门极(Gate)电压为正时,MOS开关断开 ,
#1 and #2截止



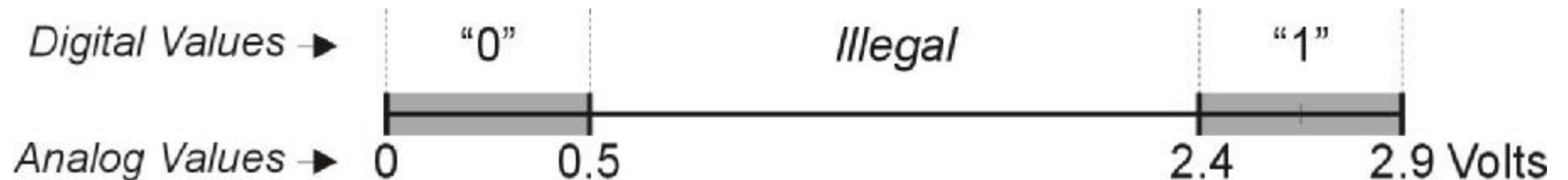
Terminal #1 must be
connected to +2.9V

基本逻辑门

} 如何利用电子开关实现基本的逻辑操作: **AND, OR, NOT.**

} 逻辑值与模拟电压:

- 用不同范围的模拟电压来表示‘0’和‘1’




- 模拟电压范围取决于制造晶体的工艺
- 通常代表“1”的高电压有: +5V, +3.3V, +2.9V, +1.8V, +1.35V, +1.0V
 - 教材教学使用 +2.9V

CMOS 电路：互补金属氧化半导体

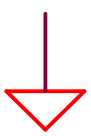

} CMOS : **Complementary** MOS

} 特点：在电路中成对使用 **n-MOS** 和 **p-MOS** 两种晶体管

- **p-MOS**

- 一端连接到 代表高电平的正电压(+), 符号 
- 当控制门输入为低时另一端输出为高电压('1').

- **n-MOS**

- 一端连接到 代表低电平的0电压(GND), 符号 
- 当控制门输入为低时另一端输出为低电压('0'). 

} 电路输出要不通过开关连接到正电压,要不通过开关连接到0电压(GND)

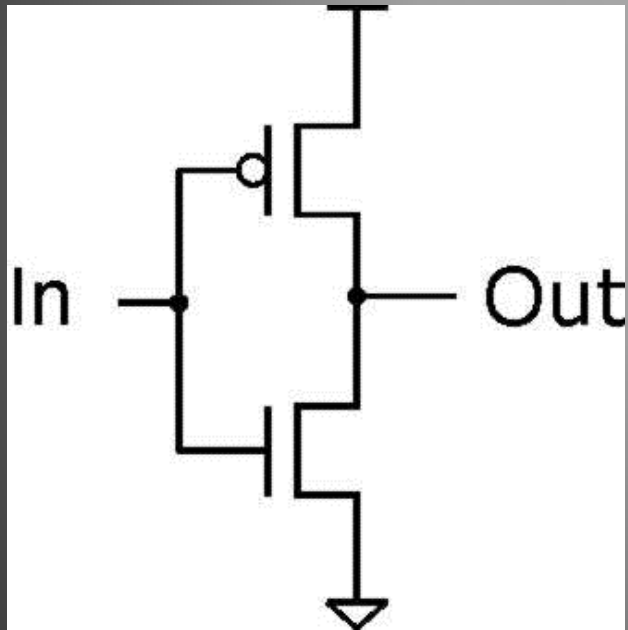
真值表：Truth Table

逻辑函数的一种最基本的表达方法
用列表的方式列出所有输入和输出的对应值
真值表的行数怎么确定？

Inputs		Outputs	
A	B ...	X	Y ...

$2^{\#inputs}$ {

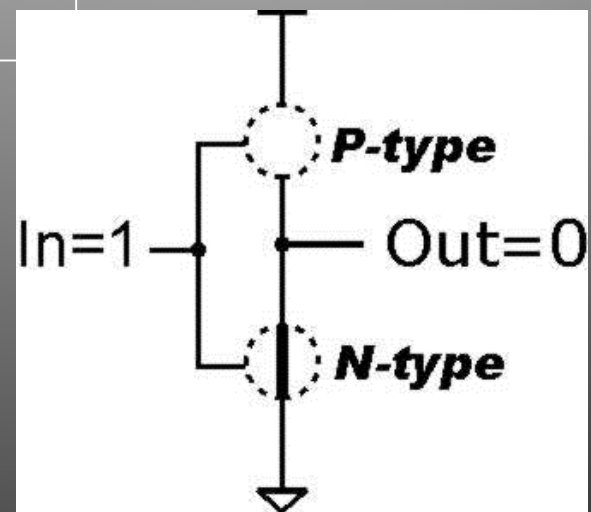
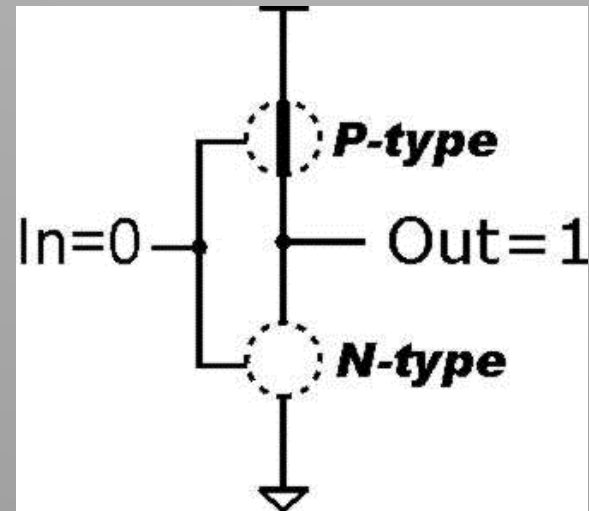
反门:NOT



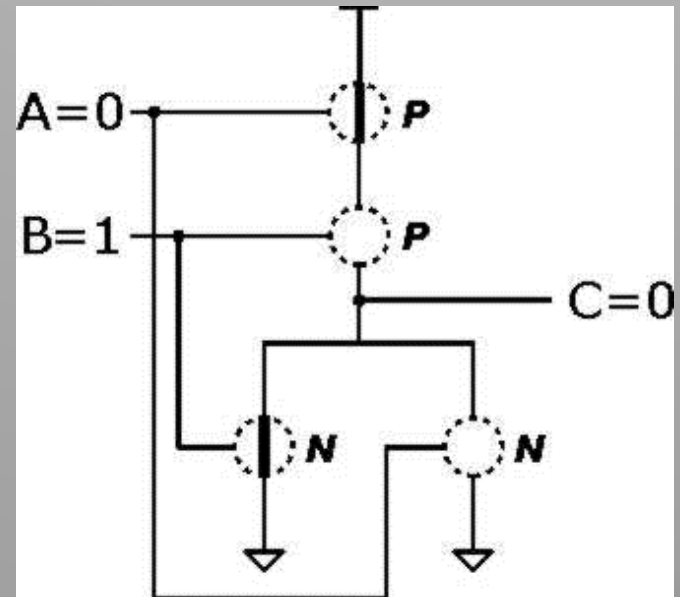
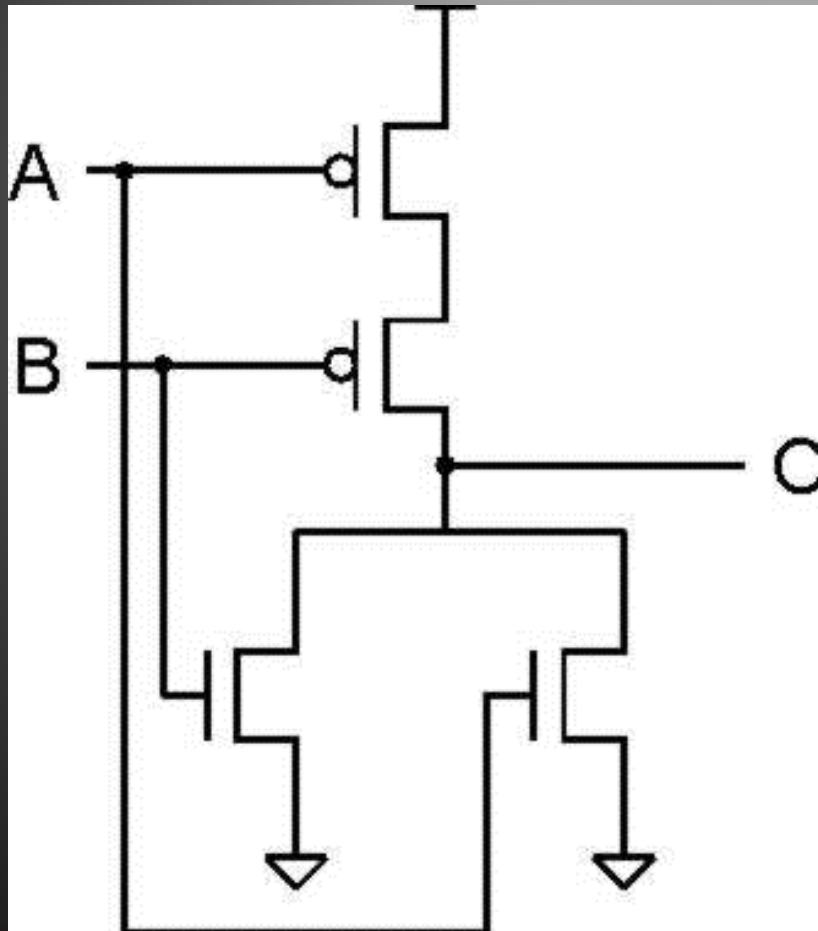
Truth table

In	Out
0 V	2.9 V
2.9 V	0 V

In	Out
0	1
1	0



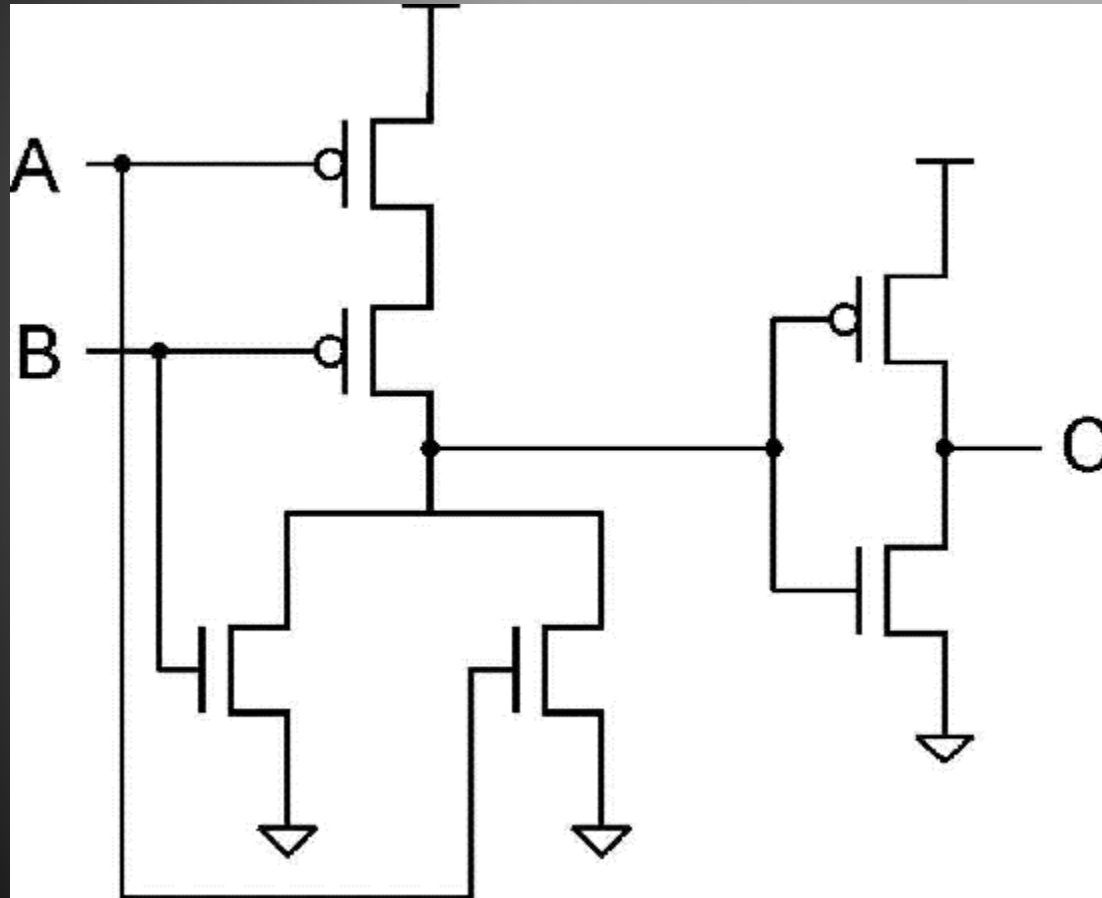
或非门(NOR)



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

Note: Serial structure on top, parallel on bottom.

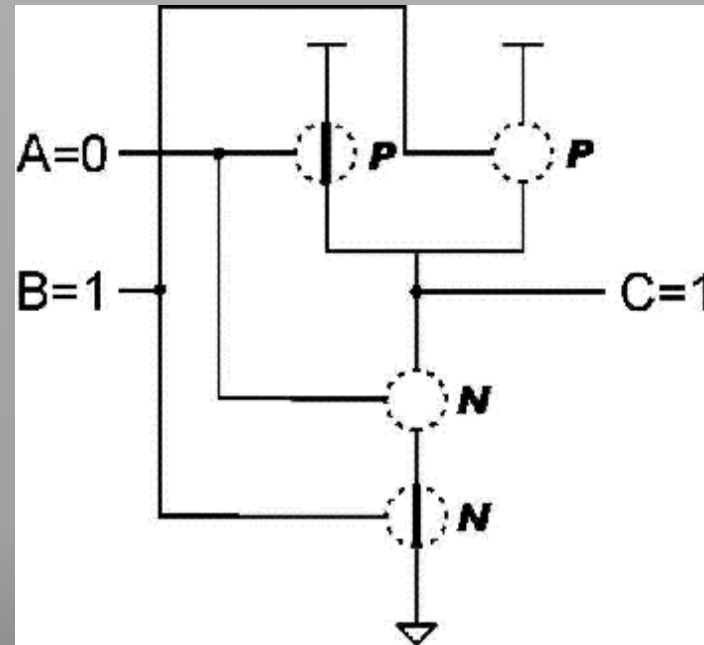
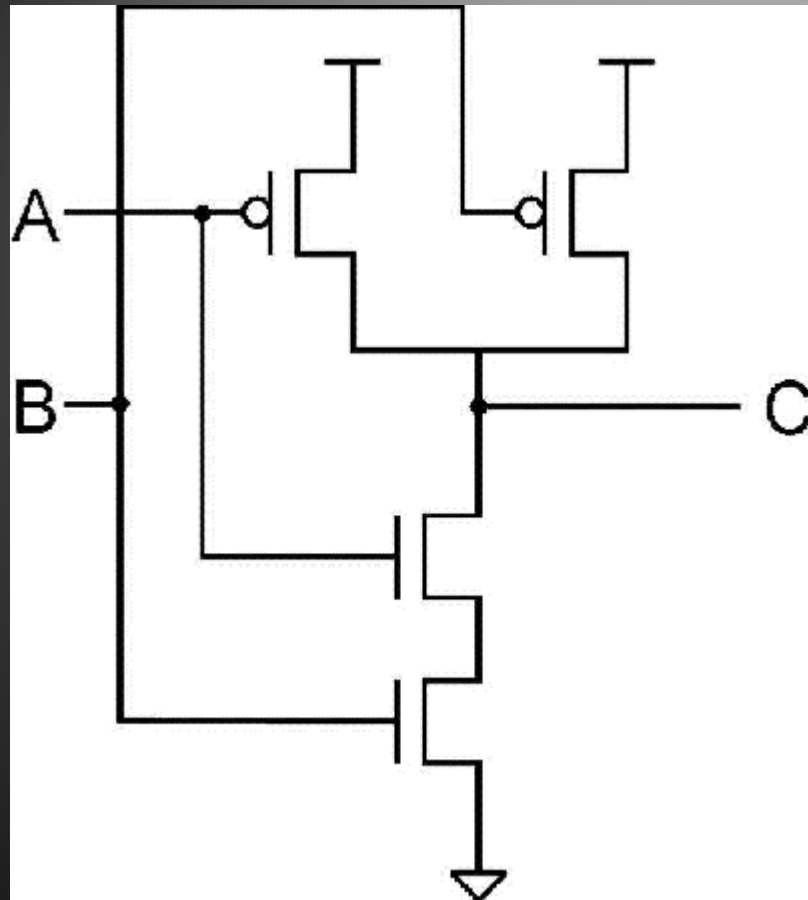
或门(OR)



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

Add inverter to NOR.

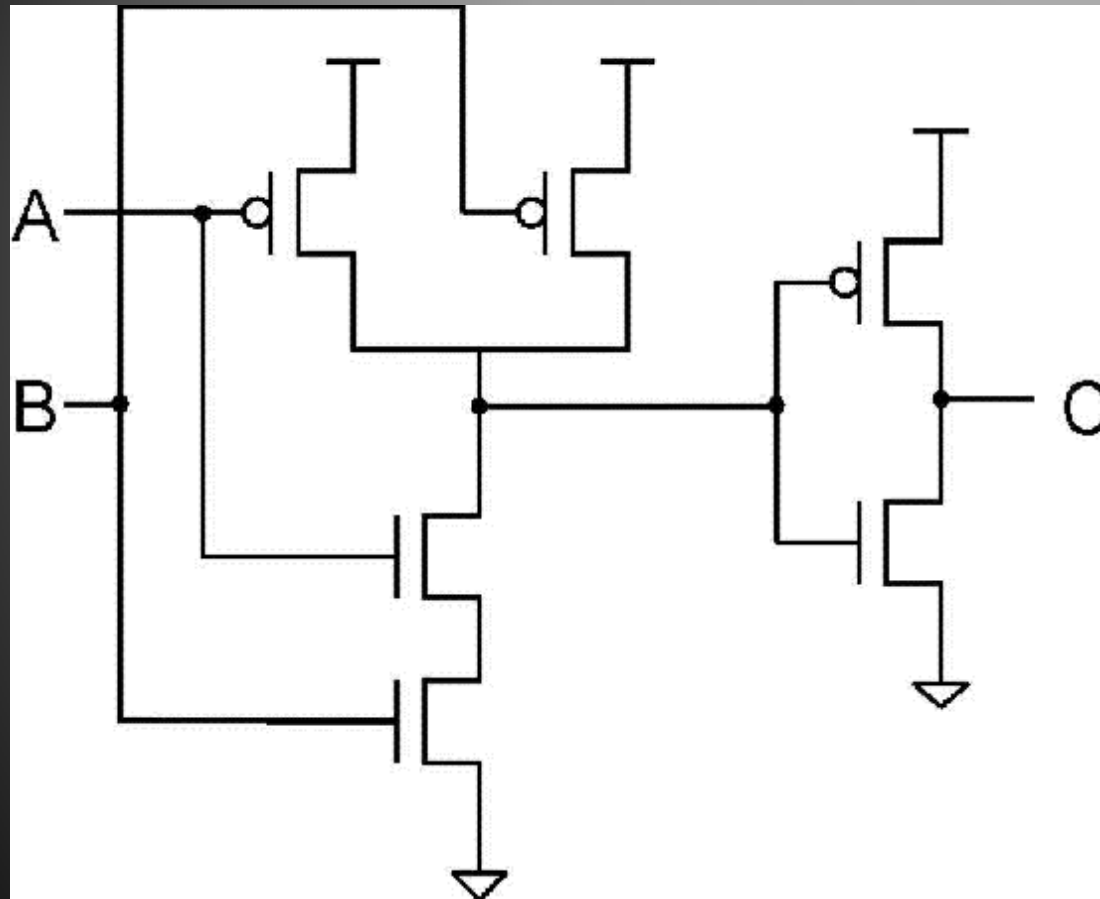
与非门(NAND)



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

Note: Parallel structure on top, serial on bottom.

与门(AND)



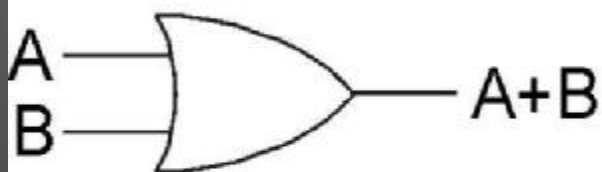
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Add inverter to NAND.

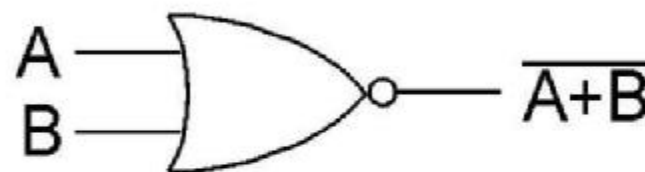
基本逻辑门符号



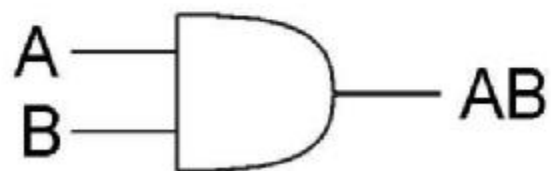
NOT



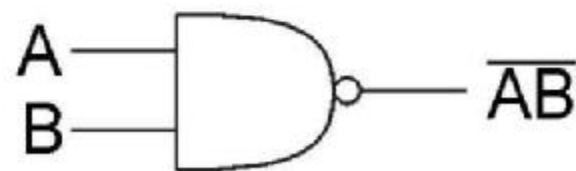
OR



NOR



AND



NAND

与或的转换：摩根定律（反演律）

} 摩根定律（反演律）：

$$\overline{A+B} = \overline{A} \cdot \overline{B} \quad (\text{证明：真值表})$$

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

} 可扩展

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

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

$$\overline{A_1 + A_2 + \mathbf{L} + A_n} = \overline{A_1} \cdot \overline{A_2} \cdot \mathbf{L} \cdot \overline{A_n}$$

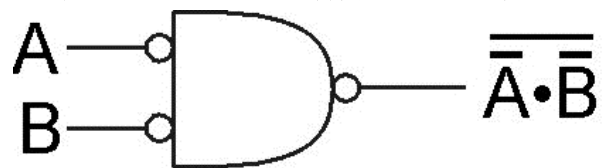
$$\overline{A_1 \cdot A_2 \cdot \mathbf{L} \cdot A_n} = \overline{A_1} + \overline{A_2} + \mathbf{L} + \overline{A_n}$$

与或的转换

} 利用 AND门,反门实现OR门

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

} 电路（A/B输入端前的小圆圈代表求反）



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

与或的转换

- } 思考：利用 OR门,反门实现AND门
- } 思考：只用 NAND门,能实现AND,NOT和OR吗?
- }

多输入的情况

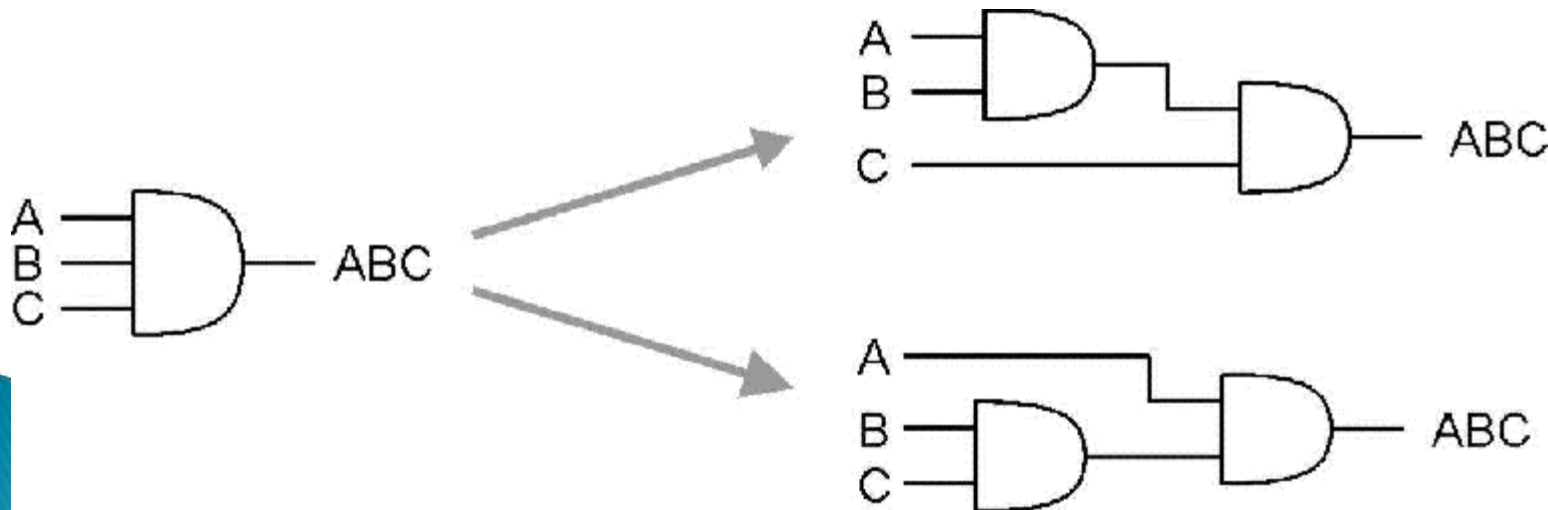
} **AND/OR** 门可以有多个的输入信号.

- **AND** : 所有输入 = 1 , 输出才为1。
- **OR** : 所有输入 = 0 , 输出才为0。

} **NAND/NOR**操作类似

- **NAND** : 所有输入 = 1 , 输出才为0。
- **OR** : 所有输入 = 0 , 输出才为1。

} 使用二输入实现三输入的的实现方法



小结

- } MOS 晶体管是最基本的电子开关器件，用来实现最基本的逻辑门电路。
 - p-MOS
 - 一端连接到 代表高电平的正电压(+)
 - 当控制门输入为低时另一端输出为高电压('1').
 - n-MOS
 - 一端连接到 代表低电平的0电压(GND)
 - 当控制门输入为低时另一端输出为低电压('0').
- } 基本门电路: NOT, NOR, NAND（分别用2， 4， 4mos晶体管）
 - 逻辑功能通常用AND, OR, and NOT表示
- } 摩根定律（反演律）
实现与或操作的转换

利用基本门电路设计功能电路

} 组合逻辑电路

- 输出只依赖当前的输入
- 无状态（无存储电路）

} 时序逻辑电路

- 输出不仅依赖当前的输入还取决于电路过去的状态
- 利用存储电路存储电路过去的状态信息
- 时序逻辑电路 = 组合逻辑电路 + 存储电路

} 我们先学习些常用的组合逻辑的功能电路,然后再学习时序电路。

简单组合逻辑电路设计

} 真值表 \rightarrow 逻辑表达式的方法

可以将任意真值表转换成基于AND, OR, NOT操作的逻辑表达式(逻辑完备性)

(1) 找出输出 $F = 1$ 的行;

(2) 对每个 $F = 1$ 的行, 取值为1的变量用原变量表示, 取值为0的变量用反变量表示, 形成与表达式;

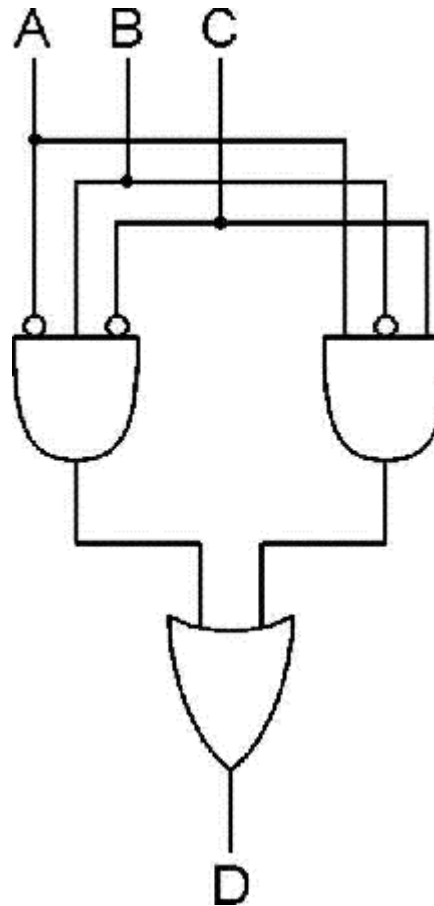
(3) 将各个与表达式进行逻辑或, 得到最终的逻辑表达式。



例子:

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

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



1. AND combinations that yield a "1" in the truth table.

2. OR the results of the AND gates.

简单组合逻辑电路设计

组合逻辑电路设计流程

- 1) 问题 **→** 确定输入输出
- 2) 形成真值表
- 3) 真值表 **→** 逻辑表达式
- 4) 逻辑表达式化简
- 5) 用基本逻辑门电路实现



【例】某厂有A、B、C三个车间和Y、Z两台发电机。如果一个车间开工，启动Z发电机即可满足使用要求；如果两个车间同时开工，启动Y发电机即可满足使用要求；如果三个车间同时开工，则需要同时启动Y、Z两台发电机才能满足使用要求。试仅用与非门和异或门两种逻辑门设计一个供电控制电路，使电力负荷达到最佳匹配。

解 用“0”表示该厂车间不开工或发电机不工作，用“1”表示该厂车间开工或发电机工作。为使电力负荷达到最佳匹配，应该根据车间的开工情况即负荷情况，来决定两台发电机的启动与否。因此，此处的供电控制电路中，A、B、C是输入变量，Y、Z是输出变量。由此列出电路的真值表如表1所示。



表1

A	B	C	Y	Z
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

例 设计一个有三个输入、一个输出的组合逻辑电路，输入为二进制。当输入二进制能被3整除时，输出为1，否则，输出为0。

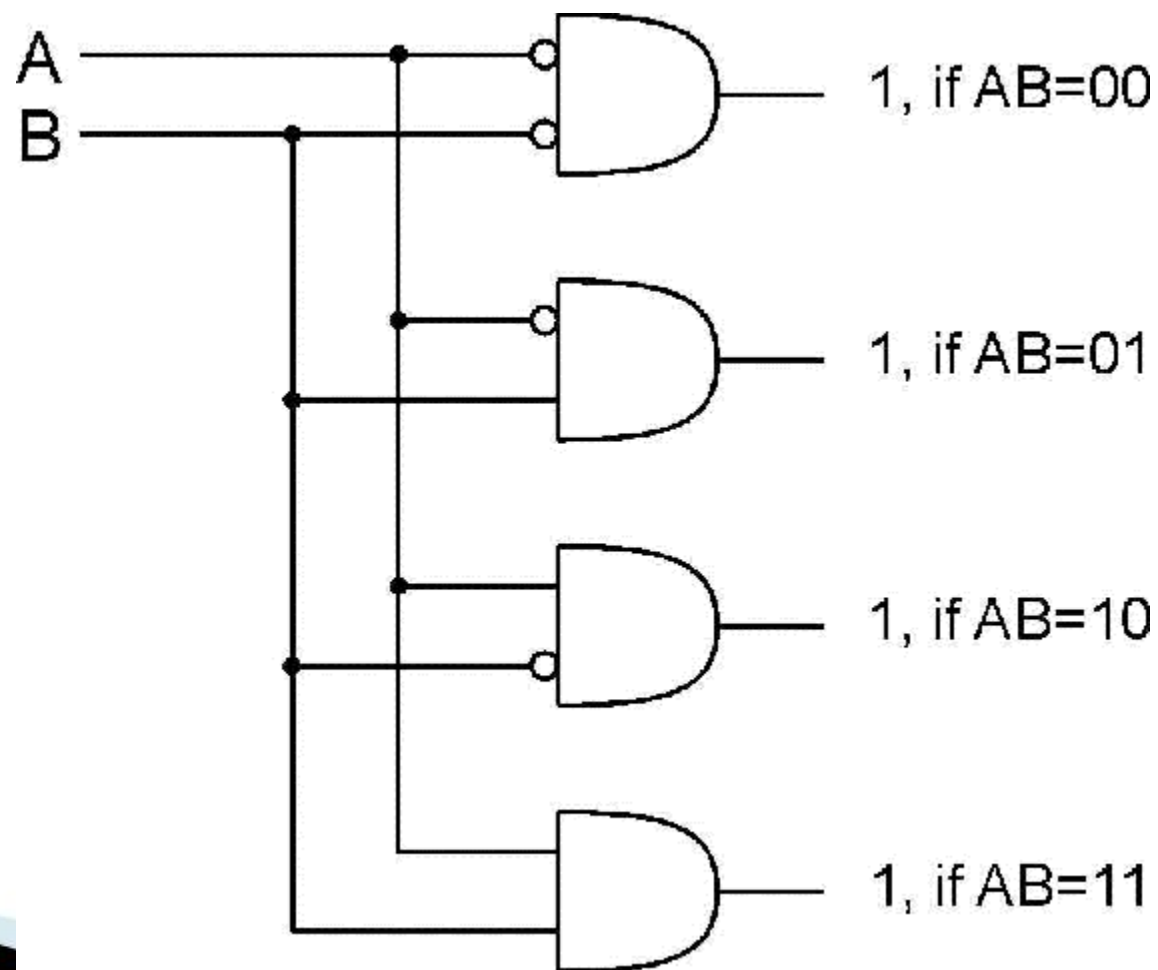


译码器

} n 输入, 2^n 输出

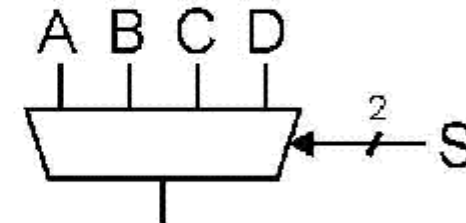
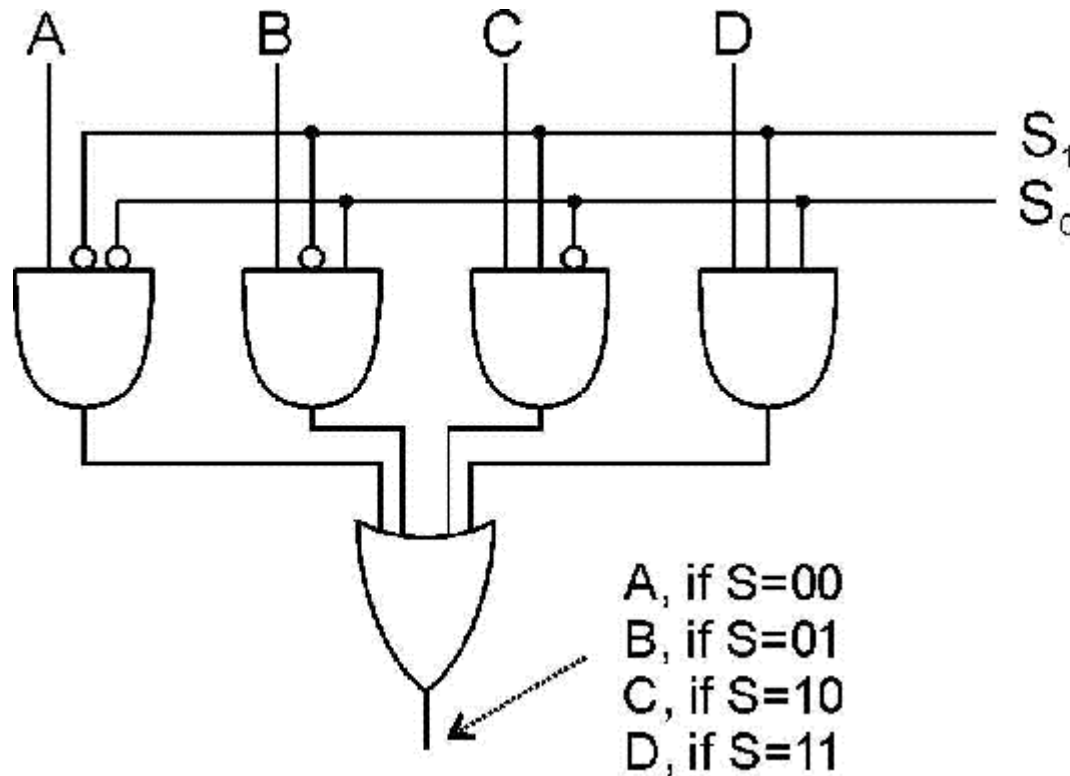
- 解释一个二进制数（将二进制的位置信息-物理位置）

*2-bit
decoder*



多路选择器 (Multiplexer : MUX)

- } 从 2^n 输入中选择一个信号输出，选择信号n个
- } 4选1的电路

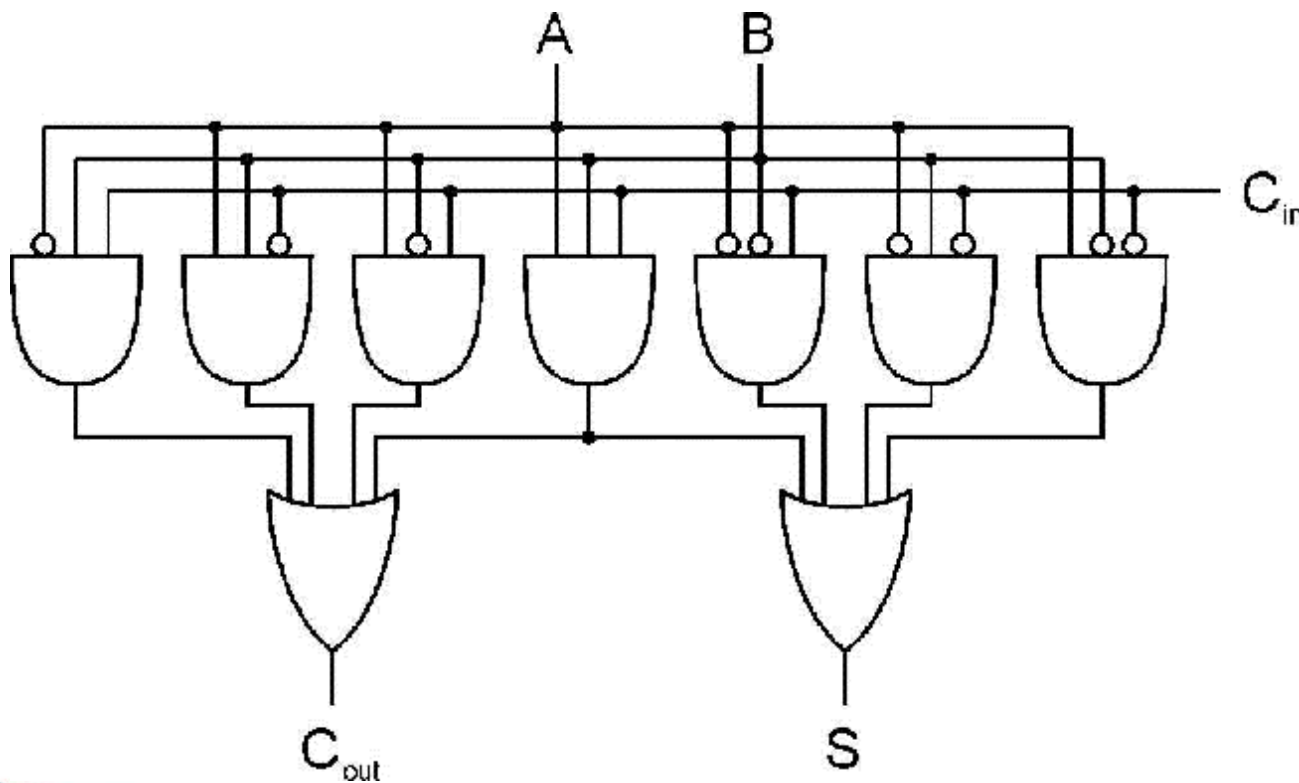


4-to-1 MUX

- } 8选1的实现?

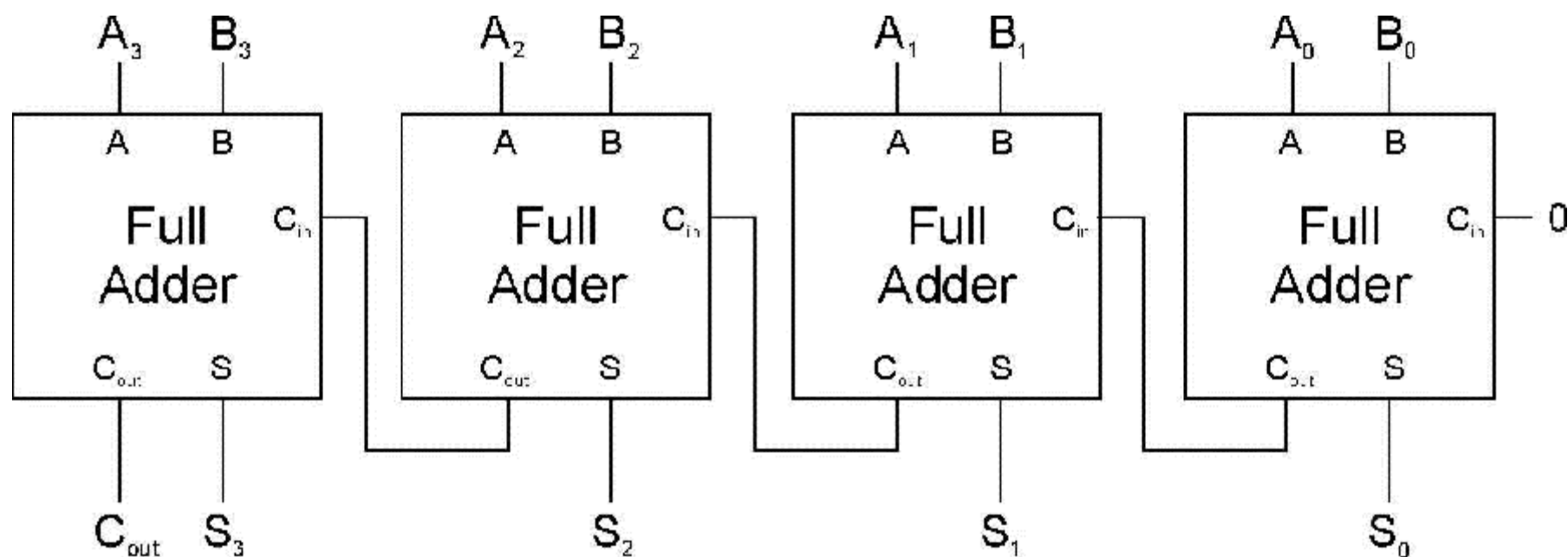
1 位全加器

} 对两个**1Bit**二进制位做加法同时加上下一级传递过来的进位。



A	B	C_{in}	S	C_{out}
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

4位加法器的实现（串行进位）



PLA:可编程逻辑阵列

- 利用软件设计硬件电路。
(PLA/CPLD/FPGA)
- 原理：实现真值表可能的所有
利用可编程的连接点选择连接
输入或门上。

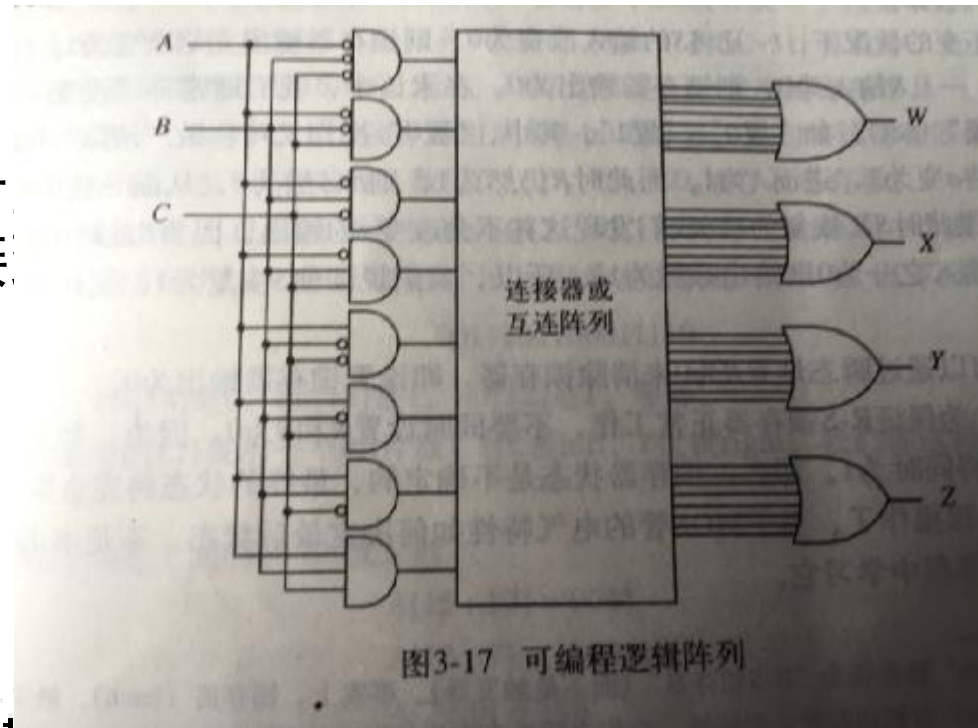
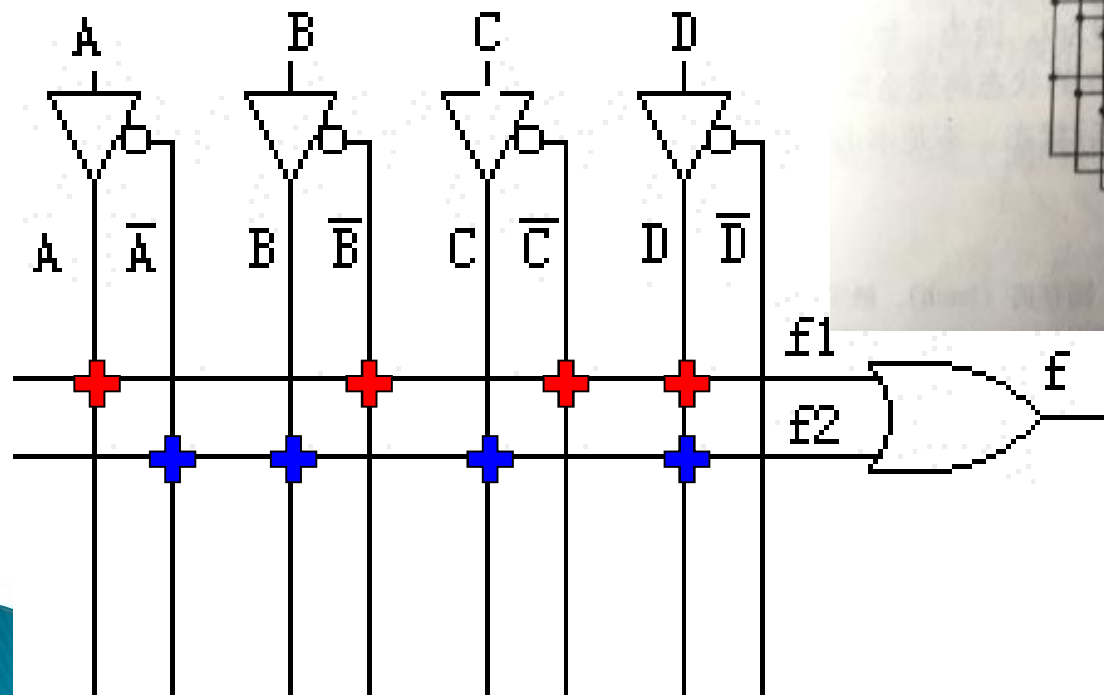


图3-17 可编程逻辑阵列

作业

Ex 3.5, 3.6, 3.7, 3.8, 3.9

Ex 3.11, 3.12, 3.18

Ex 3.20, 3.22, 3.23

Ex 3.30, 3.31



Combinational vs. Sequential

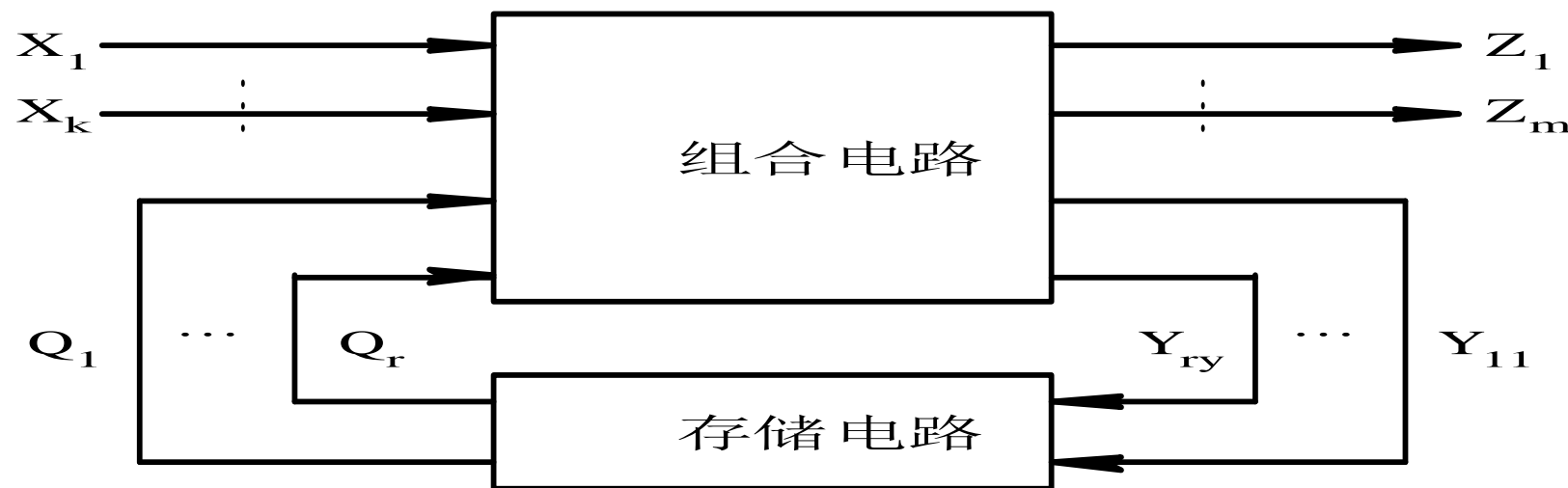
} 组合逻辑电路

- 输出只依赖当前的输入
- 无状态（无存储电路）

} 时序逻辑电路

- 输出不仅依赖当前的输入还取决于电路过去的状态
- 利用存储电路存储电路过去的状态信息
- 时序逻辑电路 = 组合逻辑电路 + 存储电路

时序逻辑电路由组合逻辑电路和起记忆作用的存储电路组成。



时序逻辑电路组成见上图所示，它由组合逻辑电路，记忆电路(锁存器组)两部分组成。其中：

X_1, X_2, \dots, X_n ——外部输入信号

Q_1, Q_2, \dots, Q_k ——存储器的输出，称为状态变量

Z_1, Z_2, \dots, Z_m ——对外输出信号

Y_1, Y_2, \dots, Y_k ——触发器的激励信号

次态与现态的概念

Q_1 、...、 Q_k 信号表示电路现在所处的状态，简称现态(Present State);

在对电路功能进行研究时，通常将某一时刻的状态称为“现态”， Q 的位数决定了状态数量.

1位: Q_0 , 电路有两个状态 0,1

2位: Q_0, Q_1 , 电路有四个状态 00,01,10,11

.....

将在某一现态下，外部信号发生变化后到达的新的状态称为“次态”。

现态和次态不是一成不变的。电路一旦从现态变为次态，对于下一个变化来讲，这个次态就变成了现态。一般时序电路中有一个控制系统状态统一变化的信号称为时钟信号，时钟信号按固定的时钟节拍变化。



与组合逻辑电路相比，时序逻辑电路具有以下两个**特点**：

- ① 结构上存在输出到输入的反馈通道，且有存储器件；
- ② 因为有存储器件，所以电路具有记忆功能。

如果仅就输入输出关系来看，也可以说时序逻辑电路具有一个特点，即电路在任何时刻的输出不仅和该时刻的输入有关，而且和过去的输入也有关系。

时序逻辑电路中可用的存储器件种类很多，可以是延迟元件，也可以是锁存器，其中以集成锁存器的使用最为广泛。



时序逻辑电路=组合逻辑电路+存储电路

存储电路实现：磁性材料（磁盘、硬盘），电子器件（内存条）。

利用电子电路来制作的记忆器件被称为锁存器。

锁存器：具有存储功能的器件，能够存储一位二进制信息的基本单元。

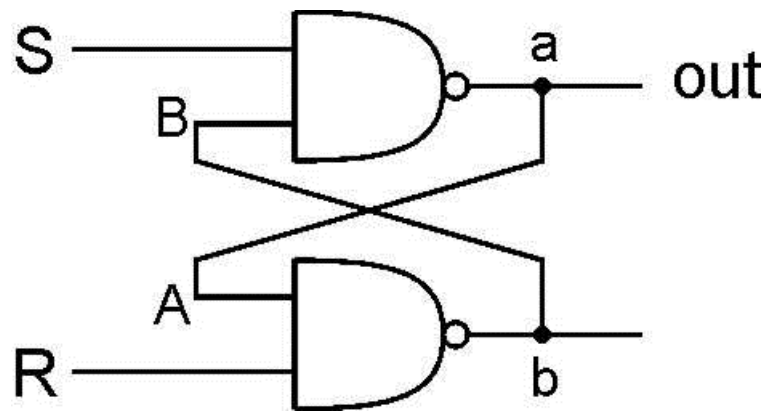
锁存器特点：

- 1、有两个稳定状态“0”态和“1”态；
- 2、能根据输入信号将锁存器置成“0”或“1”态；
- 3、输入信号消失后，被置成的“0”或“1”态能保存下来，即具有记忆功能。



基本R-S锁存器: 简单的存储单元（存储1 bit信息）

- } R信号（低有效）：“Reset” 清零信号， $R=0$ ， $S=1$ 将存储单元内容设置为‘0’
- } S信号（低有效）：“Set” 设置信号， $S=0$ ， $R=1$ 将存储单元内容设置为‘1’
- } a,b输出互反（ $a=1$ 时 $b=0$ 或 $a=0$ 时 $b=1$ ）
- } R,S同时为‘0’？



基本R-S锁存器的工作过程

- 1) 输入端 $R=0$, $S=1$: 当锁存器R端为低电平、S端为高电平时, 锁存器为0态, a端为0, 而b端为1。
- 2) 输入端 $R=1$, $S=0$: 当锁存器R端为高电平、S端为低电平时, 锁存器为1态, a端为1, 而b端为0。
- 3) 输入端 $R=1$, $S=1$: 当锁存器的两个输入端都同时为高电平时, 锁存器维持原状态不变;
- 4) 输入端 $R=0$, $S=0$: 当锁存器两输入端同时变1后, 锁存器究竟是什么状态取决于门₁和门₂的反应速度, 输出不确定。



基本R-S锁存器总结

} $R = S = 1$

- 保持当前值,存储状态

} $S = 0, R = 1$

- set value to 1

} $R = 0, S = 1$

- set value to 0

} $R = S = 0$

- 禁止态, 不允许同时写入 '0' 和 '1'
- 回到保持态后输出不确定, 可能为 '1' 也可能为 '0'
- *Don't do it!*

基本R-S锁存器的问题

} 基本RS触发器的特点:

1) S: 直接置位端

R: 直接复位端

2) 无统一控制信号, 状态转化由RS端直接控制

} 基本触发器问题:

1) 输入电平R、S直接控制触发器输出状态, 多个存储器位读写时转态变化不统一, 使用不便, 抗干扰能力差;

2) R、S 输入有约束。R、S 不能同时有效 ($R+S=1$)



解决：门控D锁存器（Gated D-Latch）

} 2个输入： D (data) , WE (write enable)

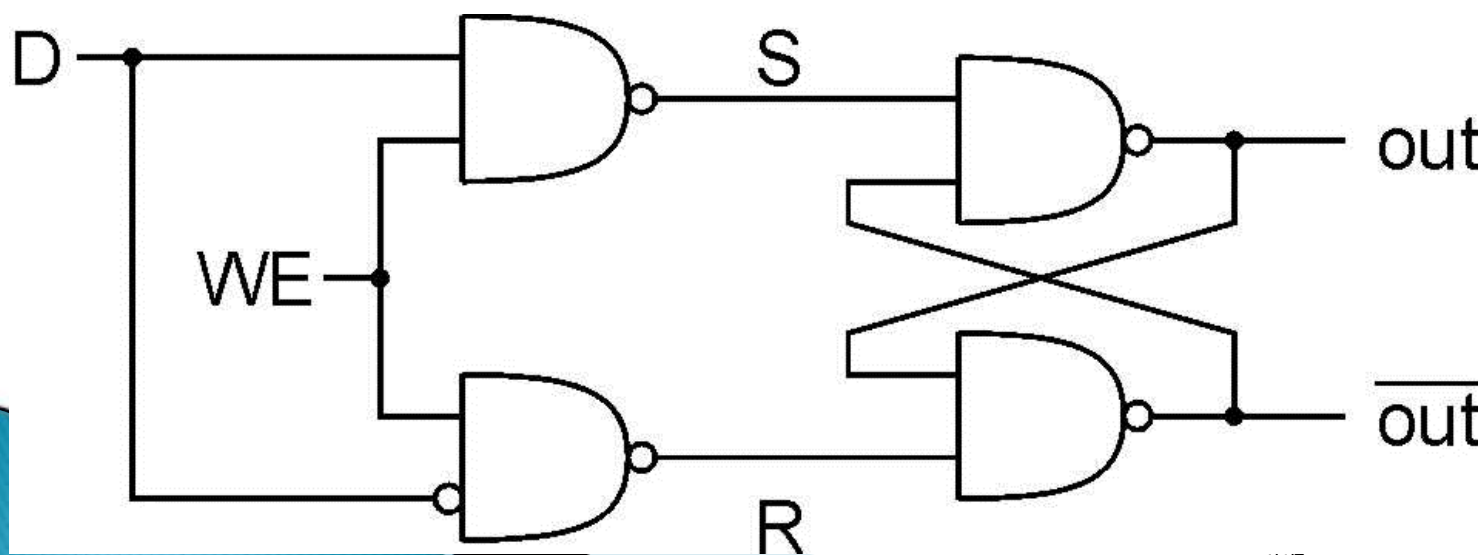
- 当 **WE = 0**, 锁存器保持。(此时D信号变化不影响输出)

 - $S = R = 1$

- 当 **WE = 1** (此时D信号变化影响输出, 锁存器输出为D的值)

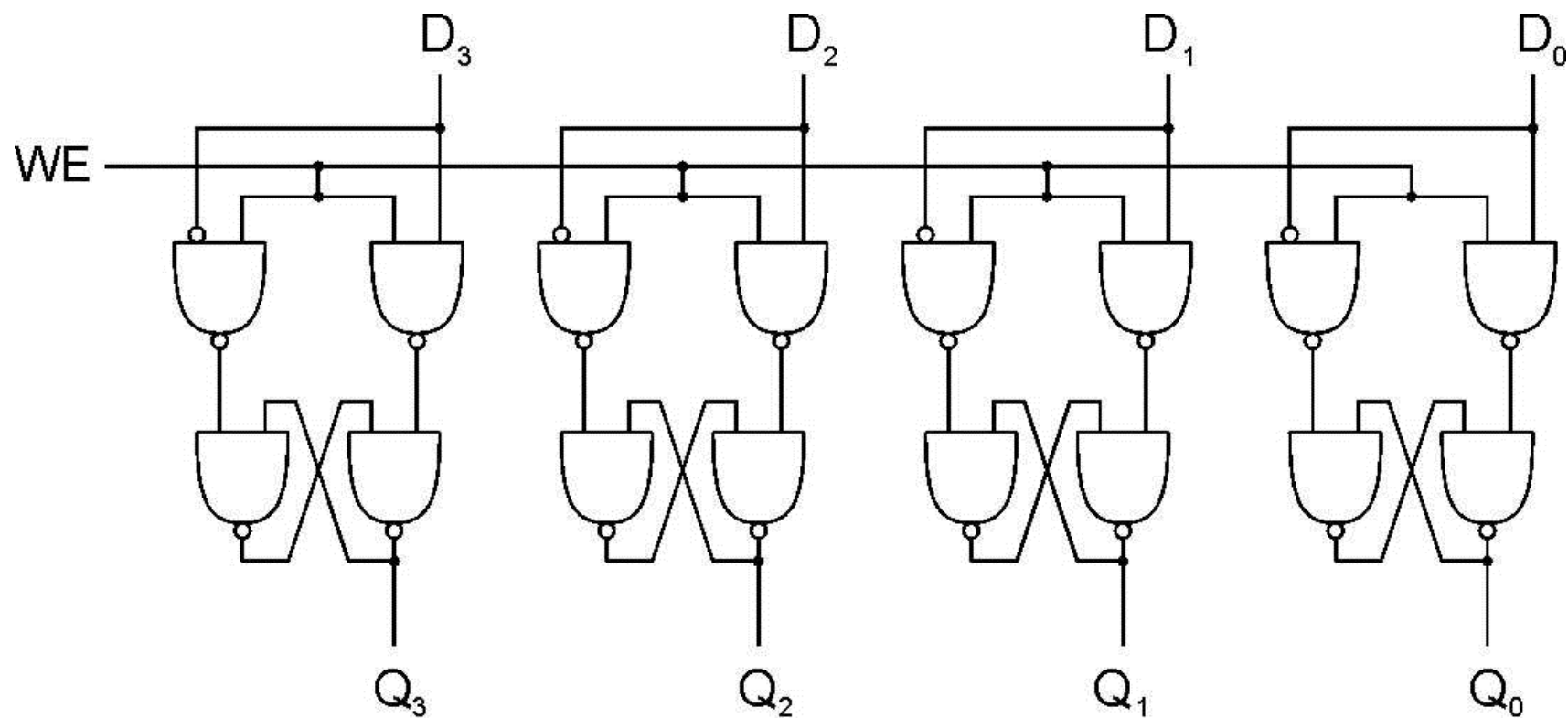
 - D=0: $S=1, R=0$ out=0

 - D=1: $S=0, R=1$ out=1 消除了约束。



寄存器

} 寄存器存放多位数据，有多个锁存器组成。



4-bit寄存器

多位数据表示(例: 3.4.3)

- 1. 从右到左表示数据位 从(0)到(n-1)位
- 2. 可使用位地址表示

$$A = \overset{15}{0} \underset{14}{1} \underset{13}{0} \underset{12}{1} \underset{11}{0} \underset{10}{0} \underset{9}{1} \underset{8}{1} \underset{7}{0} \underset{6}{1} \underset{5}{0} \underset{4}{1} \underset{3}{0} \underset{2}{1} \underset{1}{0} \underset{0}{1}$$

$A[14:9] = 101001$ $A[2:0] = 101$



内存的概念

} 内存由一定数目的存储单元组成，其中每个存储单元可以被单独识别(具有唯一的地址)并存放一个数据。

寻址空间:

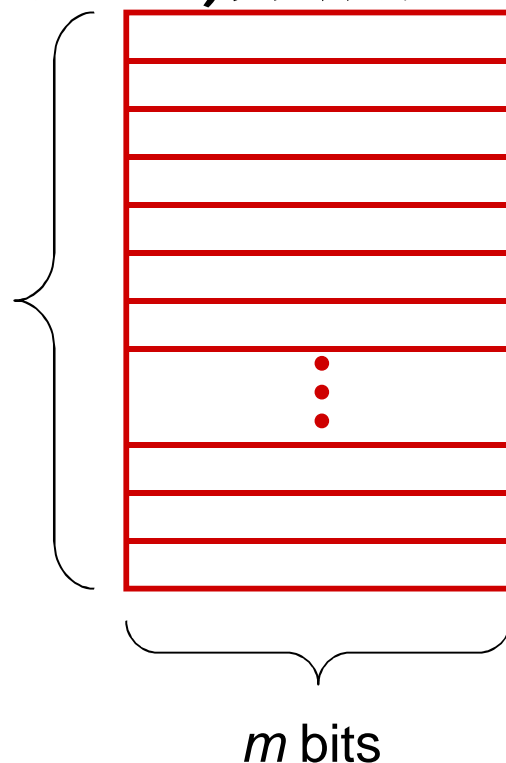
可独立识别的存储单元总数

(通常为2的整数次幂)

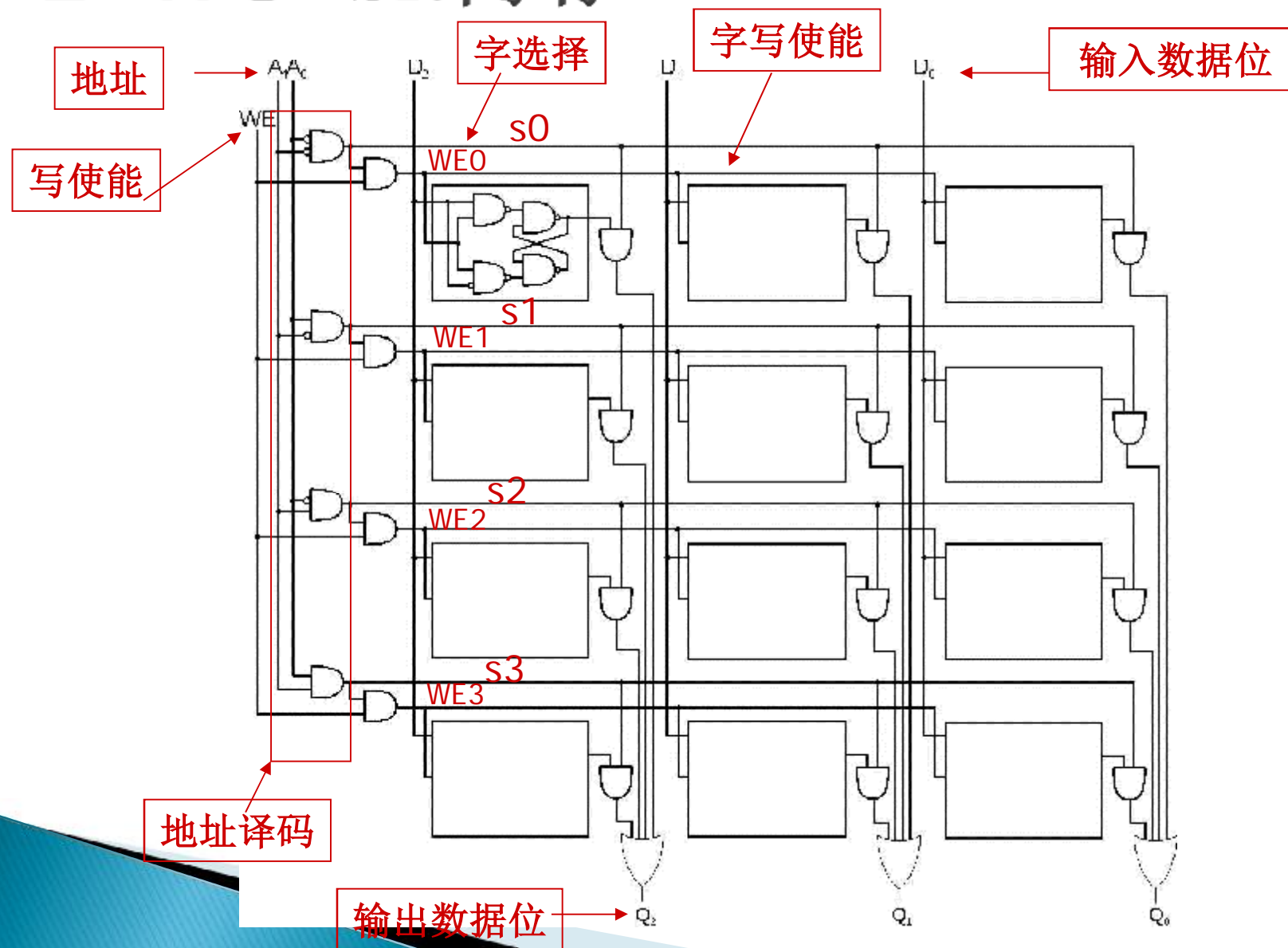
$k = 2^n$
locations

寻址能力:

每个内存存储单元中包含的bit数目
(例如: 每个位置存取一个字节)



$2^2 \times 3\text{-bit}$ 内存

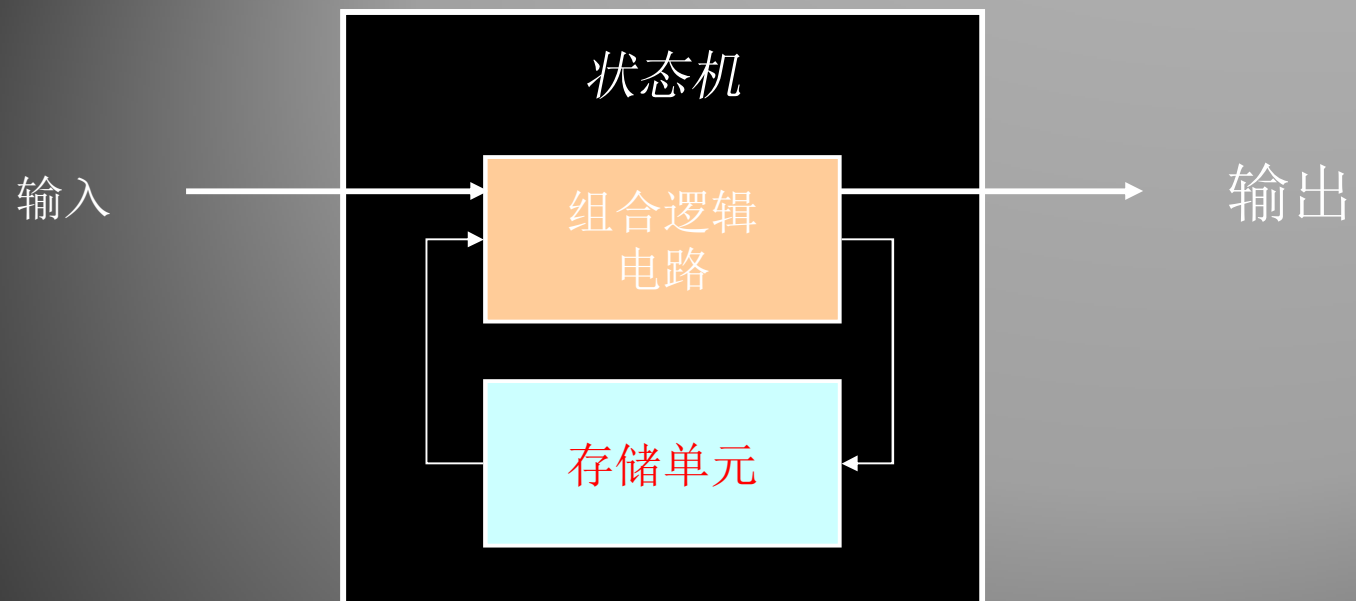


对内存的更多了解

- } This is a not the way actual memory is implemented.
 - fewer transistors, much more dense, relies on electrical properties
- } But the logical structure is very similar.
 - address decoder
 - word select line
 - word write enable
- } Two basic kinds of **RAM** (Random Access Memory)
- } **Static RAM** (SRAM)
 - fast, maintains data as long as power applied
- } **Dynamic RAM** (DRAM)
 - slower but denser, bit storage decays – must be periodically refreshed

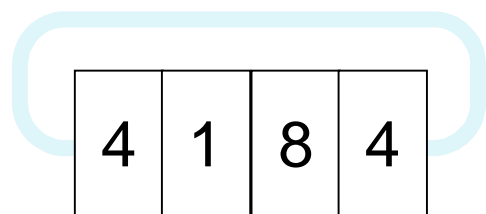
Also, non-volatile memory: ROM, PROM, flash, ...

3.6时序电路

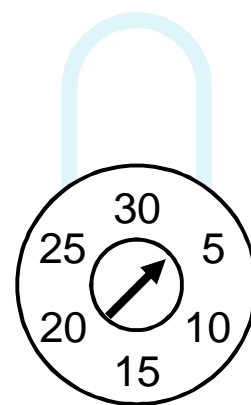


时序与组合

两类密码锁



组合:
数值正确即可开锁.



时序:
数值和顺序都正确才能开锁.

状态图 (时序逻辑电路的“真值表”)

系统在特定时刻和特定条件下的快照。

例如:

- 1. 记分牌可以代表篮球比赛的状态
 - 比分, 剩余时间, 控球方等。
- 2. X或者O的位置可以描述游戏“三子连珠”的状态。



密码锁状态

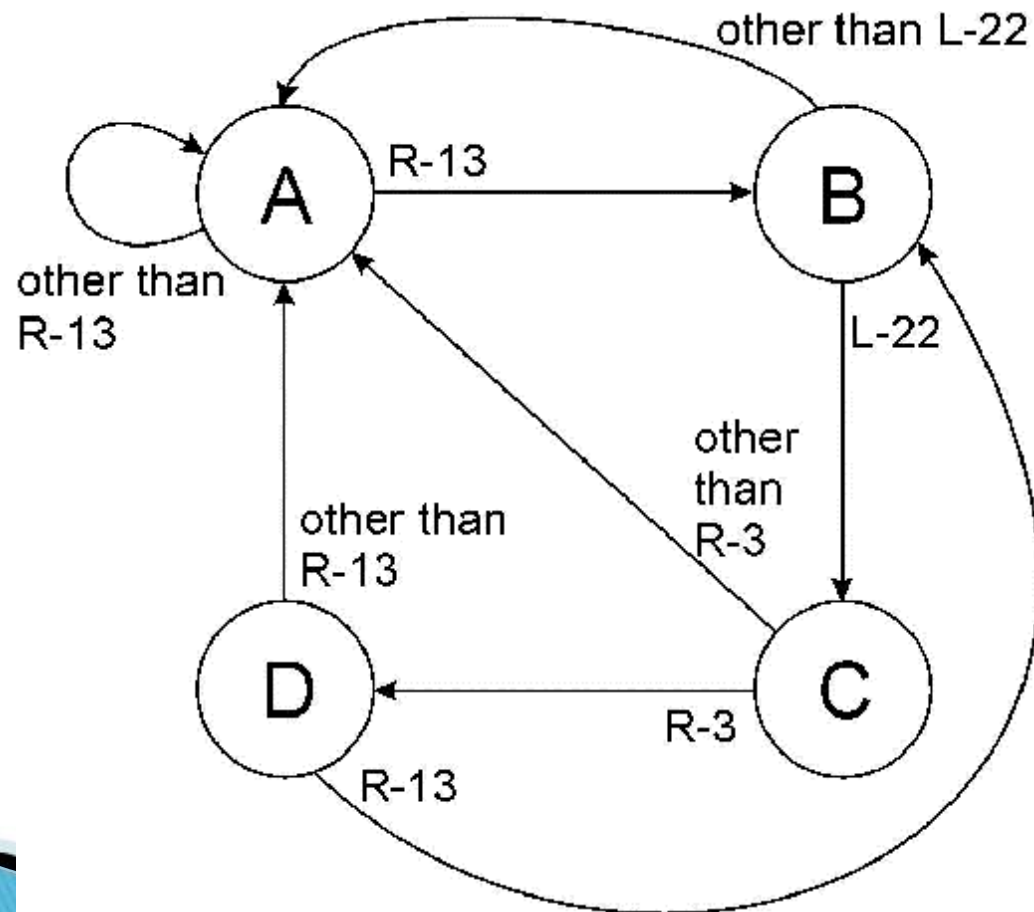
密码锁的4种状态，以A-D表示：

- A:** 闭锁状态，且无外部操作。
- B:** 闭锁状态，刚完成R13操作。
- C:** 闭锁状态，已完成R13-L22操作序列。
- D:** 开锁状态，已完成R13-L22-R3操作序列。




状态图:时序电路的一种描述方法

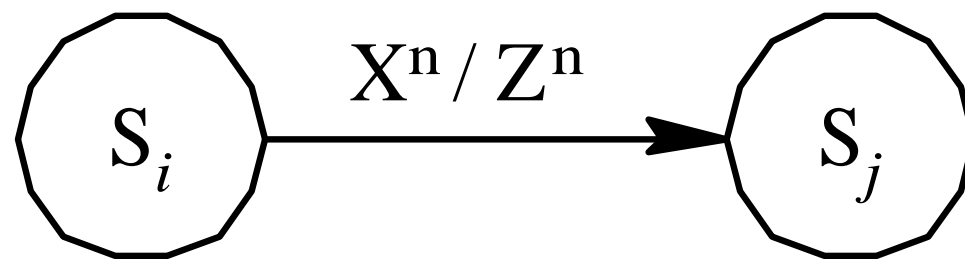
状态节点和节点间的连线组合。



状态图（有限状态机）描述法

状态图(State Diagram)是时序逻辑电路状态转换图的简称，它能够直观地描述时序逻辑电路的状态转换关系和输入输出关系，是分析和设计时序逻辑电路的一个重要工具。在状态图中，电路的状态用状态名符号外加圆圈(称为状态圈)来表示，状态转换的方向用箭头来表示，箭头旁以X/Z的形式标出转换的输入条件X和相应的电路输出Z，如图所示。该图读法如下：当电路在时刻 t^n 处于现态 S_i 而输入为X时，电路输出为Z；在时刻 t^{n+1} ，电路将转换到次态 S_j 。





状态图





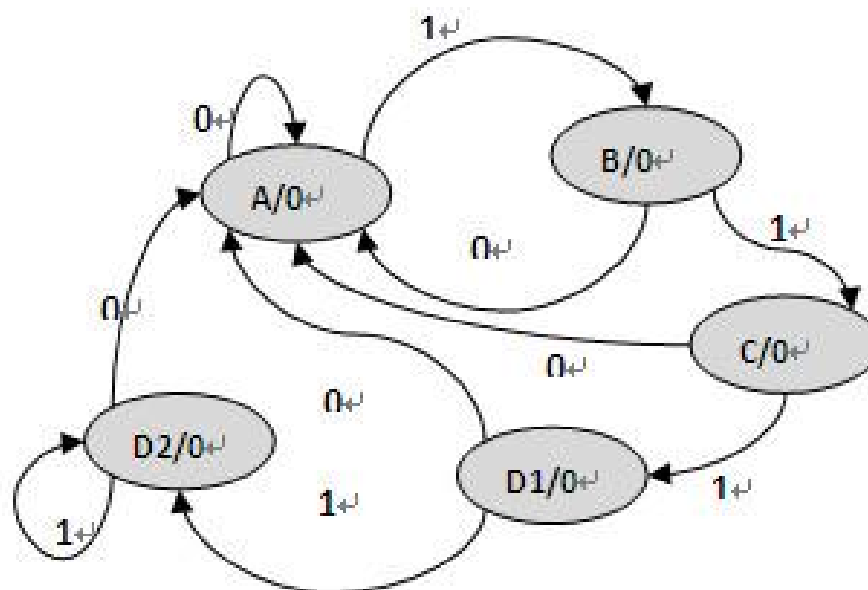
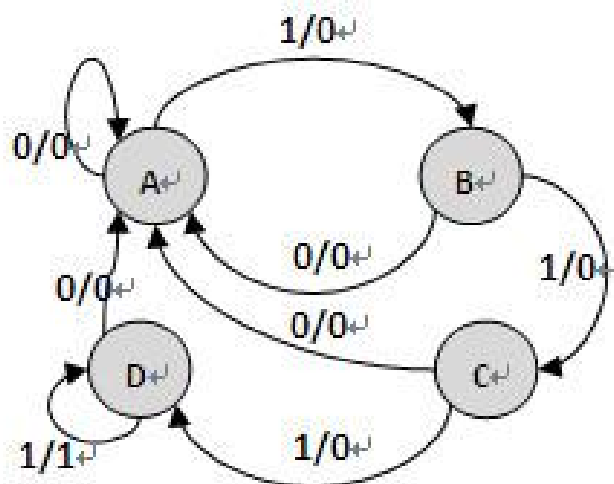
有限状态机

有限状态机的5个组成部分:

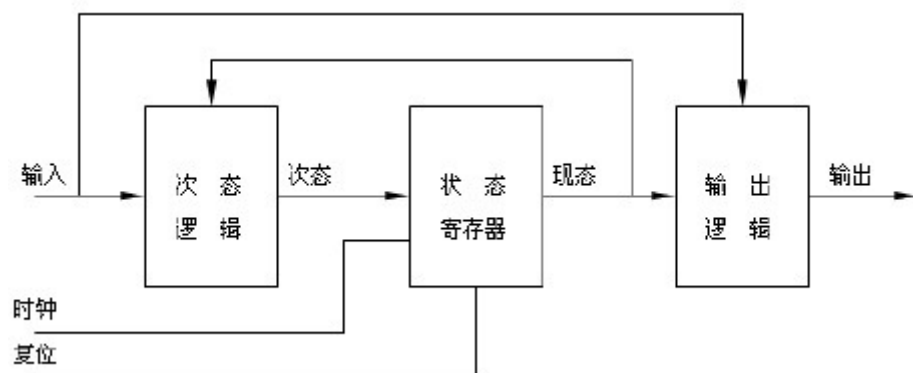
1. 状态(有限数目)
2. 外部输入(有限数目)
3. 对外输出(有限数目)
4. 任意状态间转移(显式注明)
5. 对外输出操作(显式注明)
 - 输入可产生状态转移
 - 输出可以由1)当前状态决定
 - 2)当前状态+输入



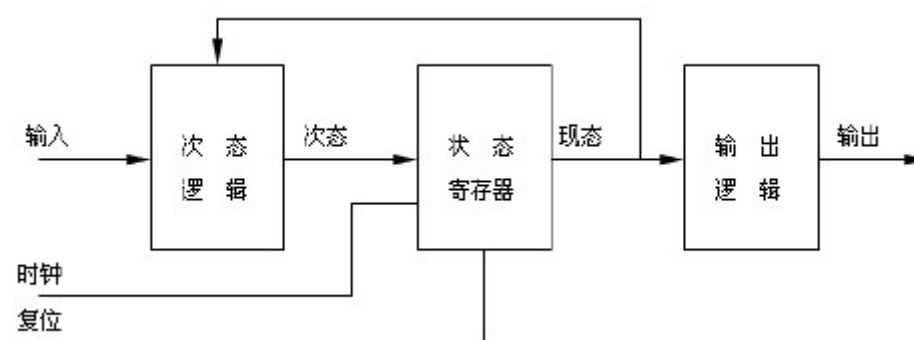
输出可以由1)当前状态决定
2)当前状态+输入



Mealy 型状态

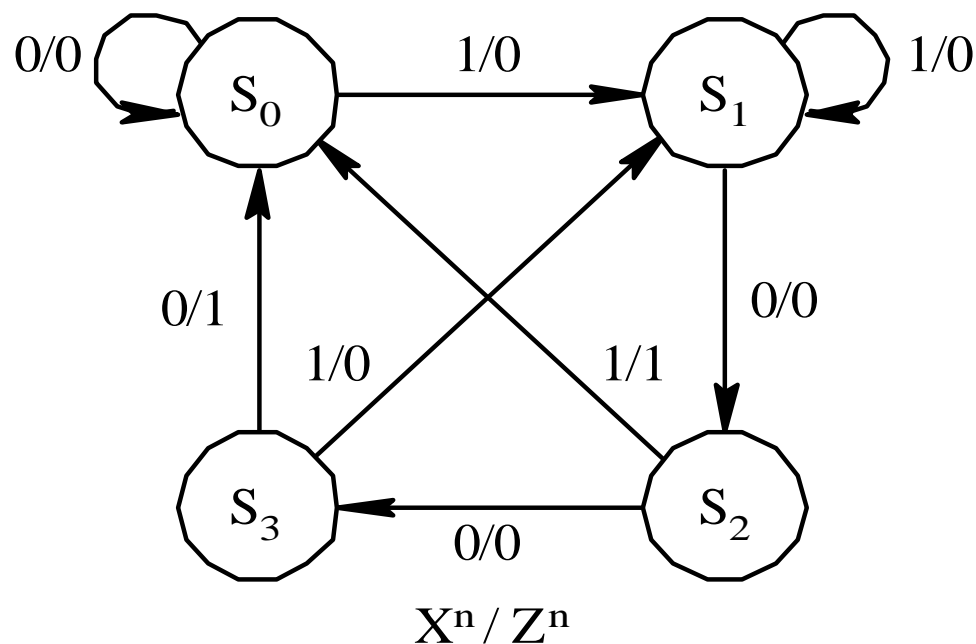


Moore 型状态



【例】 某时序逻辑电路的状态图如图所示。假定电路现在处于状态 S_0 ，试确定电路输入序列为 $X=1000010110$ 时的状态序列和输出序列，并说明最后一位输入后电路所处的状态。

解 根据电路的状态图、初始状态及输入序列，可以推导如下：



状态图

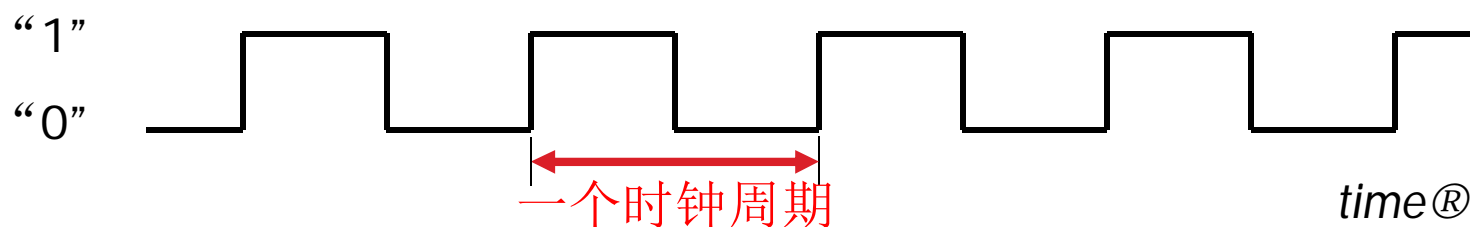
时刻	0	1	2	3	4	5	6	7	8	9
输入X	1	0	0	0	0	1	0	1	1	0
现态	S0	S1	S2	S3	S0	S0	S1	S2	S0	S1
次态	S1	S2	S3	S0	S0	S1	S2	S0	S1	S2
输出Z	0	0	0	1	0	0	0	1	0	0

可见，当电路处于初始状态S0且输入序列X=1000010110时，状态序列为S1S2S3S0S0S1S2S0S1S2，Z输出序列为0001000100，最后一位输入后电路处于S2状态。



时钟

} 通常，状态转移是通过**时钟**来触发的。



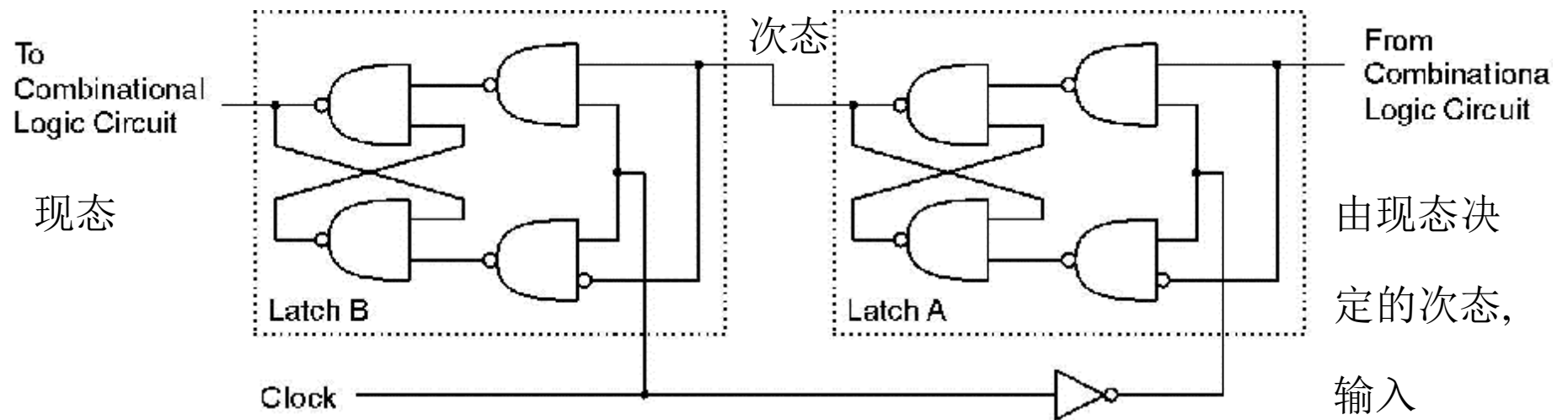
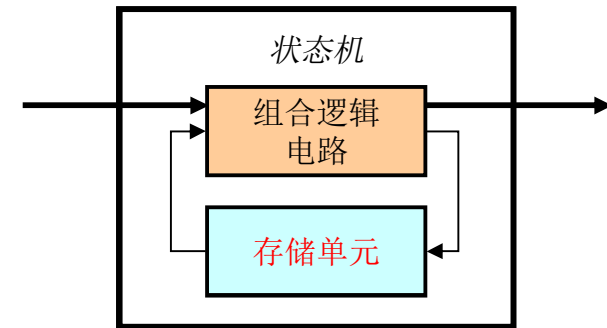
} 在每个时钟周期的开始，基于当前状态和外部输入，状态机进行转换。



存储单元：主从锁存器

一对D锁存器构成一个主从锁存器，将当前状态和下一状态分离。

（每个时钟只动作一次）



During 2nd phase (clock=0), *next* state, computed by logic circuit, is stored in Latch A.

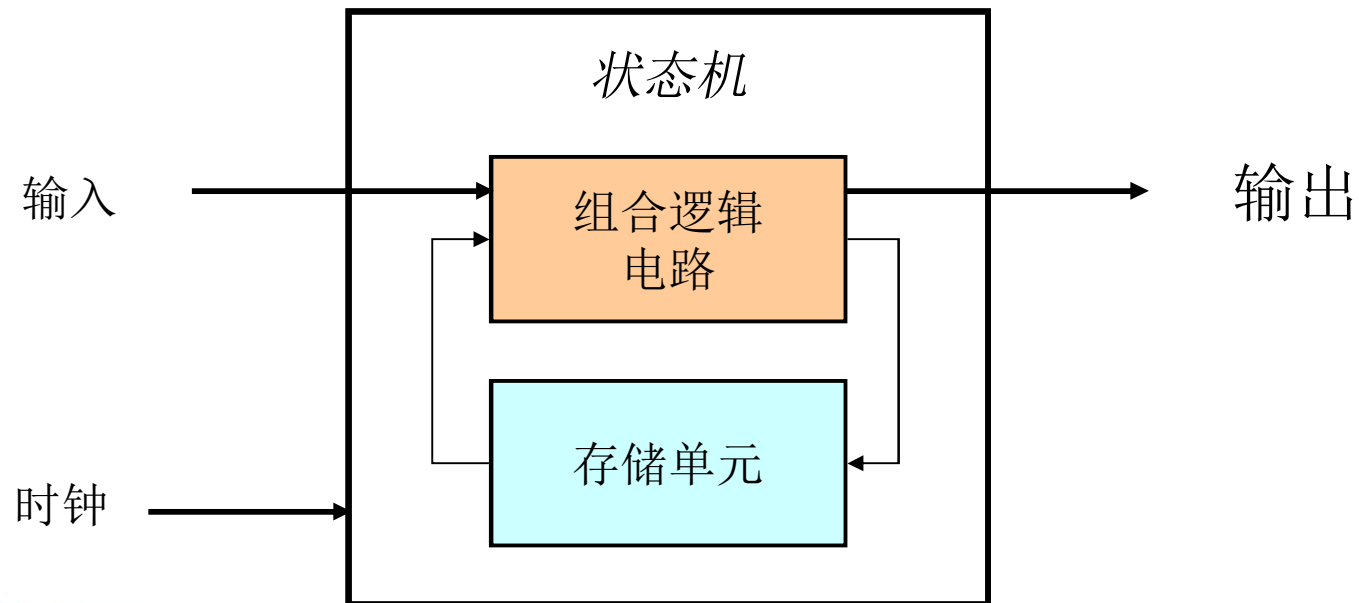
During 1st phase (clock=1), previously-computed state becomes *current* state and is sent to the logic circuit.

有限状态机的实现

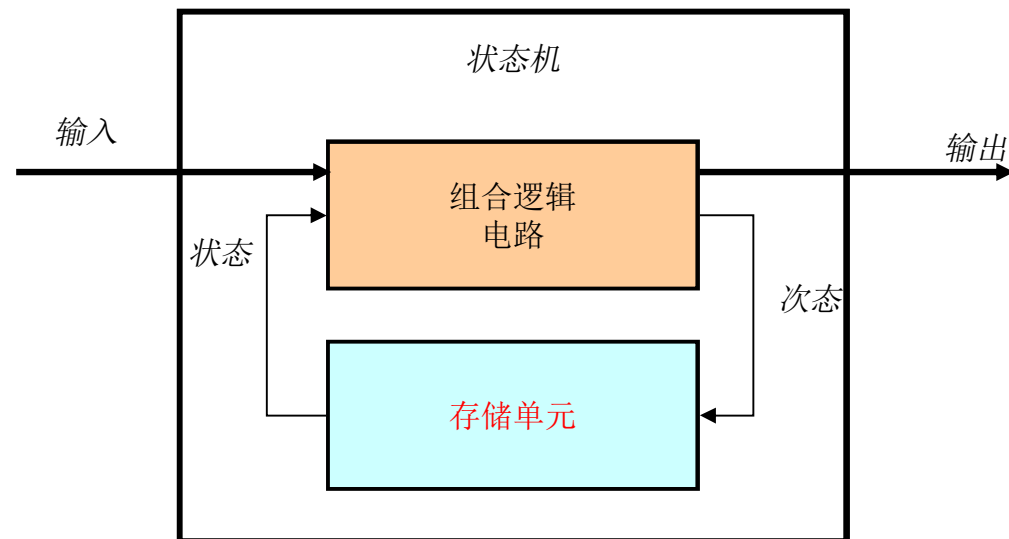
- 组合逻辑电路：
- 决定输出和下一个状态。

存储单元

存储各种状态。



两个真值表（输出，次态）



状态	输入	输出	。	。

状态	输入	次态

输出可以由1)当前状态决定
2)当前状态+输入

例子

交通警告牌:当通电工作时候的状态
(Switch=on)

- S00: No lights on ($Z=0, Y=0, X=0$)
- S01: light1 & 2 on ($Z=1, Y=0, X=0$)
- S02: light1, 2, 3, & 4 on ($Z=1, Y=1, X=0$)
- S03: light1, 2, 3, 4, & 5 on ($Z=1, Y=1, X=1$)

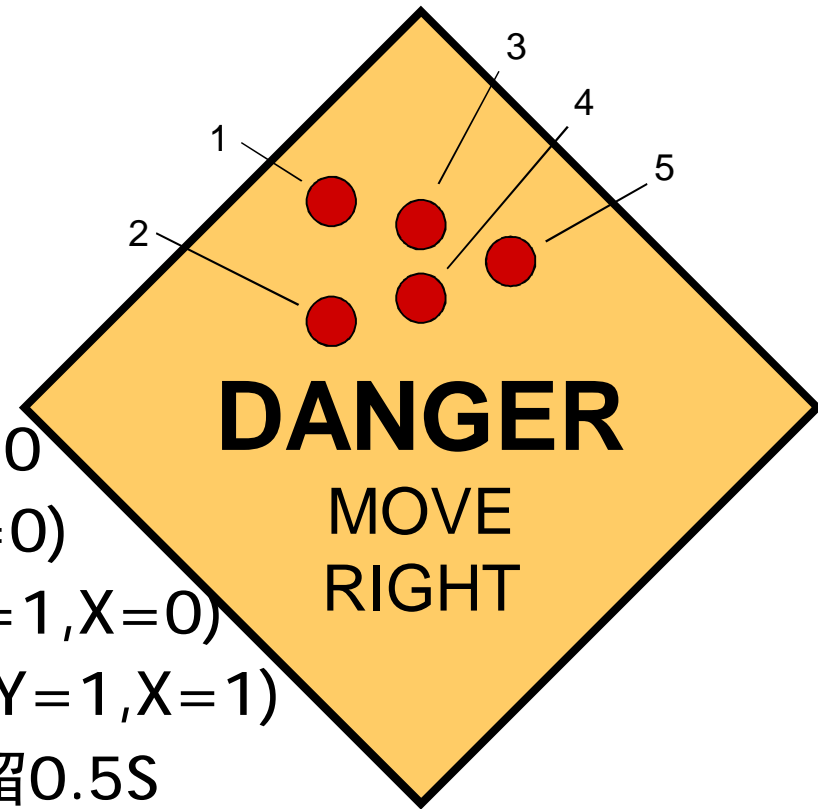
开关接通时反复循环, 每个状态停留0.5S
(Switch=off)

- S00 开关断开时状态

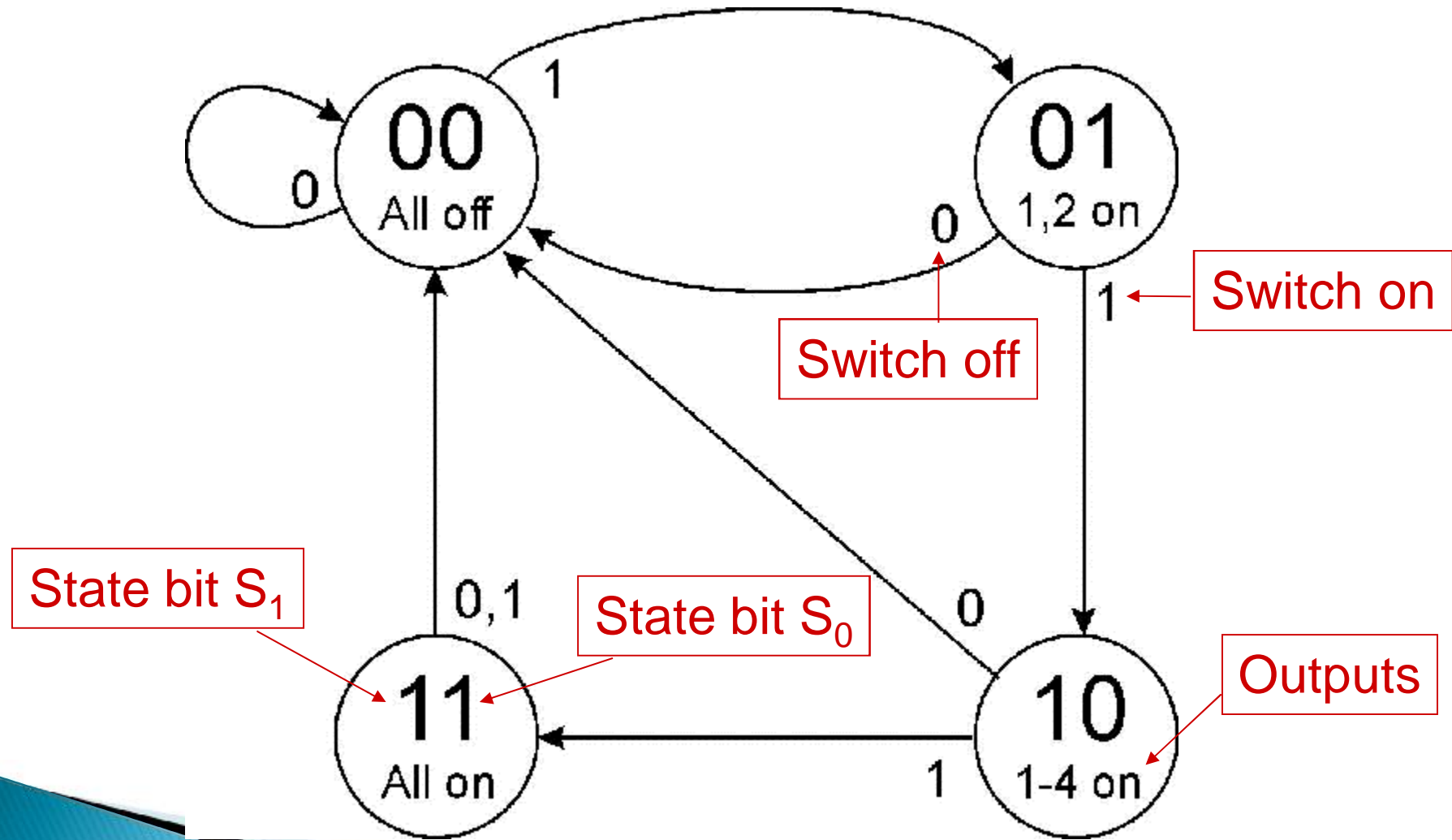
LED1,2 controlled by Z

LED3,4 controlled by Y

LED5 controlled by X



状态图



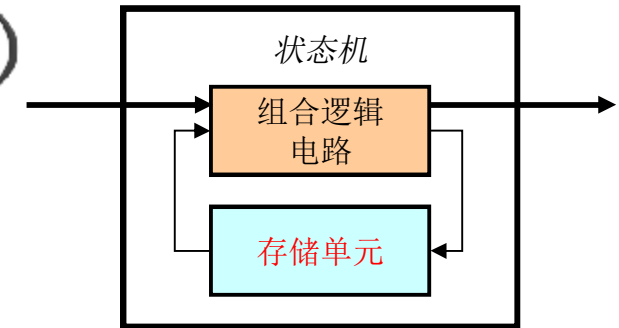
Transition on each clock cycle.

两个真值表（输出/次态）

Outputs
(depend only on state: S_1S_0)

S_1	S_0	Z	Y	X
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

Lights 1 and 2 → Z
Lights 3 and 4 → Y
Light 5 → X



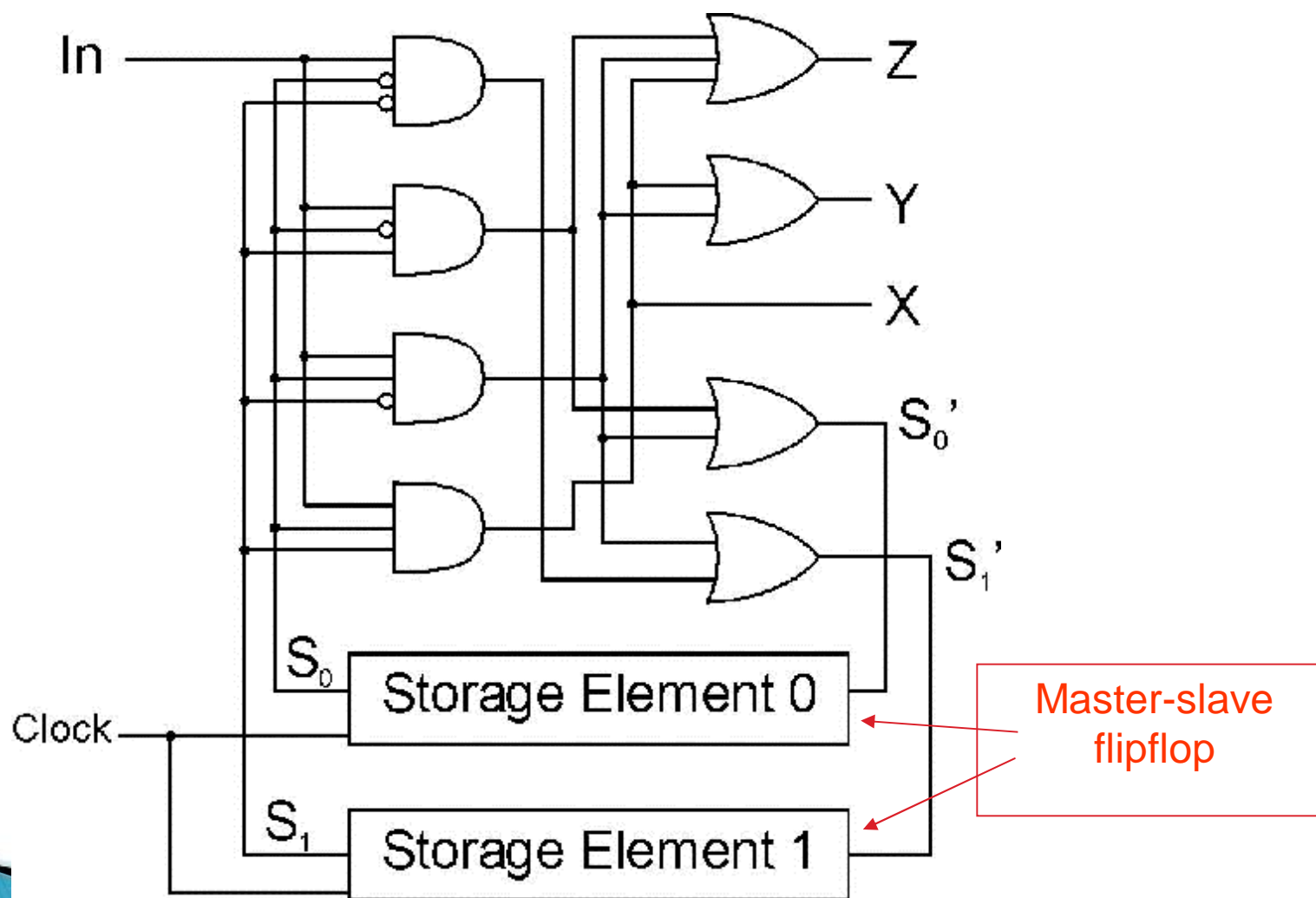
In	S_1	S_0	S_1'	S_0'	Z
0	X	X	0	0	
1	0	0	0	1	
1	0	1	1	0	
1	1	0	1	1	
1	1	1	0	0	

Switch → In

Whenever In=0, next state is 00.

Next State: $S_1'S_0'$
(depend on state and input)

逻辑实现



从逻辑电路到数据通路

} 数据通路：计算机用来处理信息所用到的所有逻辑电路

Example: LC-3 的数据通路（下一页）.

} 组合逻辑

- Decoders（译码器） -- 把指令转换成控制信号
- Multiplexers（选择器） -- 选择输入和输出
- ALU (算术和逻辑运算单元) - 对数据进行运算操作

} 时序逻辑

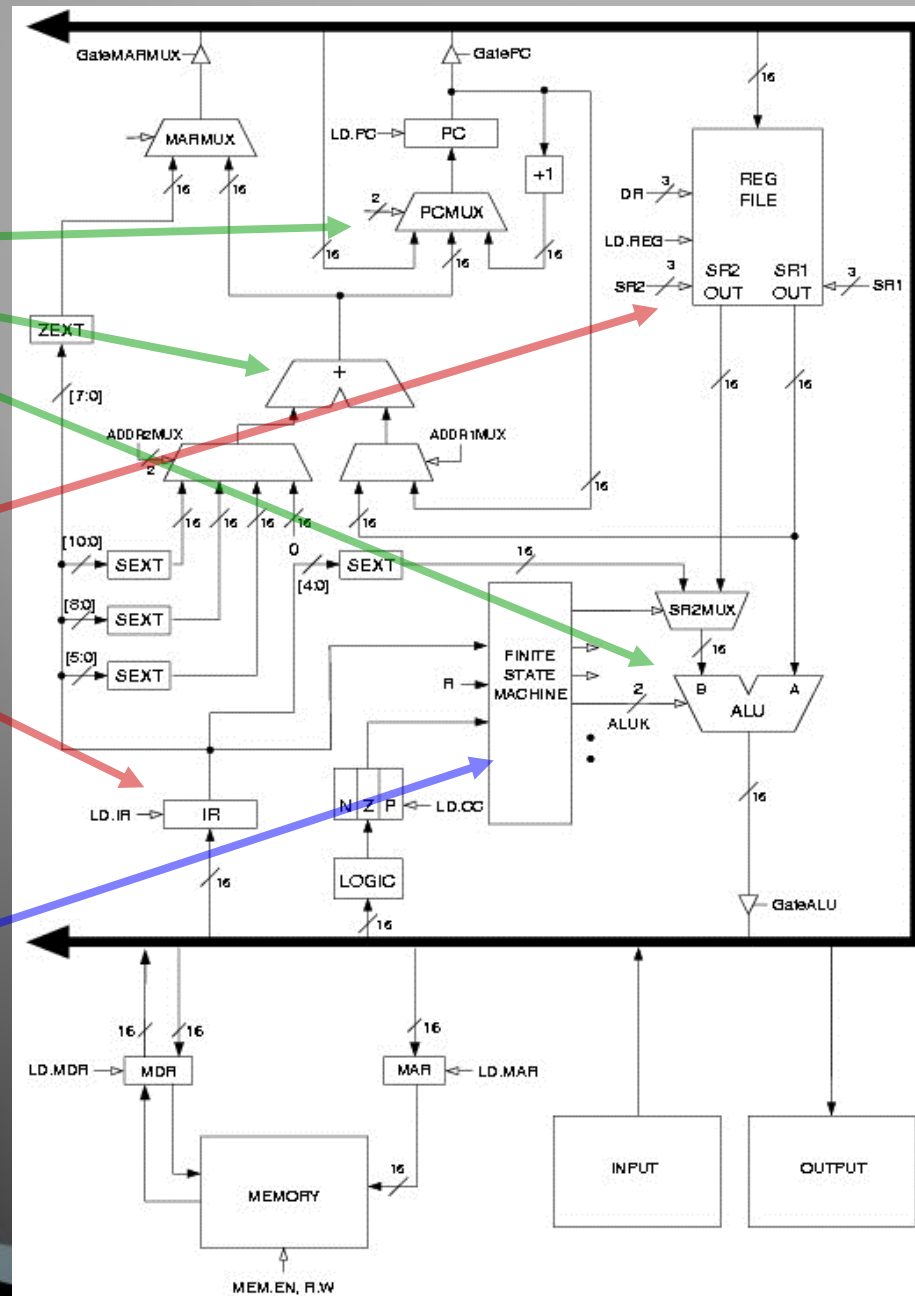
- 状态机 -- 产生控制信号，负责数据移动
- 寄存器和锁存器 - 存储部件

LC-3 数据通路

Combinational
Logic

Storage

State Machine



作业

} Ex 3.19, Ex 3.21, 3.34, 3.35

} Ex 3.40, 3.41, 3.43

