

# 第四章 存储器



Institute of Computer Architecture

## 三. 存储器层次结构

### (一)存储器的分类

### (二)层次化存储器的基本结构

### (三)半导体随机存取存储器

#### 1. SRAM存储器

#### 2.DRAM存储器

#### 3. Flash存储器

### (四)主存储器

#### 1. DRAM芯片和内存条

#### 2. 多模块存储器

#### 3.主存储器与CPU的连接

### (五) 外部存储器

### 1. 磁盘存储器

### 2. 固态硬盘 (SSD)

### (六) 高速缓冲存储器(Cache)

#### 1. Cache的基本工作原理

#### 2. Cache和主存之间的映射方式

#### 4. Cache中主存块的替换算法

#### 5.Cache写策略

### (七) 虚拟存储器

#### 1. 虚拟存储器的基本概念

#### 2. 页式虚拟存储器

基本原理，页表，地址转换，TLB（快表）

#### 3. 段式虚拟存储器

#### 4. 段页式虚拟存储器

# Contents

- 1 4.1 概 述
- 2 4.2 主存储器
- 3 4.3 高速缓冲存储器
- 4 4.4 辅助存储器
- 5 4.5 虚拟存储器

# 4.1 概述

## 回顾：存储器基本术语

- 记忆单元（存储基元 / 存储元 / 位元）（Cell）
  - 具有两种稳态的能够表示二进制数码0和1的物理器件
- 存储单元 / 编址单位（Addressing Unit）
  - 主存中具有相同地址的位构成一个存储单元，也称为一个编址单位
- 存储体 / 存储矩阵 / 存储阵列（Bank）
  - 所有存储单元构成一个存储阵列
- 编址方式（Addressing Mode）
  - 字节编址、按字编址
- 存储器地址寄存器（Memory Address Register - MAR）
  - 用于存放主存单元地址的寄存器
- 存储器数据寄存器（Memory Data Register-MDR (MBR)）
  - 用于存放主存单元中的数据的数据的寄存器

# 4.1 概 述

## 一、存储器分类

### 1. 按存储介质分类

(1) 半导体存储器    双极型    静态MOS型    动态MOS型    易失

(2) 磁表面存储器    磁盘 (Disk) 、磁带 (Tape)

(3) 磁芯存储器    硬磁材料、环状元件  
(破坏性读出)

(4) 光盘存储器    激光、磁光材料

非  
易  
失

# 一、存储器分类

4.1

## 2. 按存取方式分类

### (1) 存取时间与物理地址无关（随机访问）

- 每个单元读写时间一样，且与各单元所在位置无关。如：内存。
- 随机存储器 在程序的执行过程中 可读可写
- 只读存储器 在程序的执行过程中 只读

### (2) 存取时间与物理地址有关（串行访问）

#### ◆ 顺序存取存储器 (Sequential Access Memory, SAM )

- 数据按顺序从存储载体的始端读出或写入，因而存取时间的长短与信息所在位置有关。例如：磁带。

#### ◆ 直接存取存储器 (Direct Access Memory, DAM )

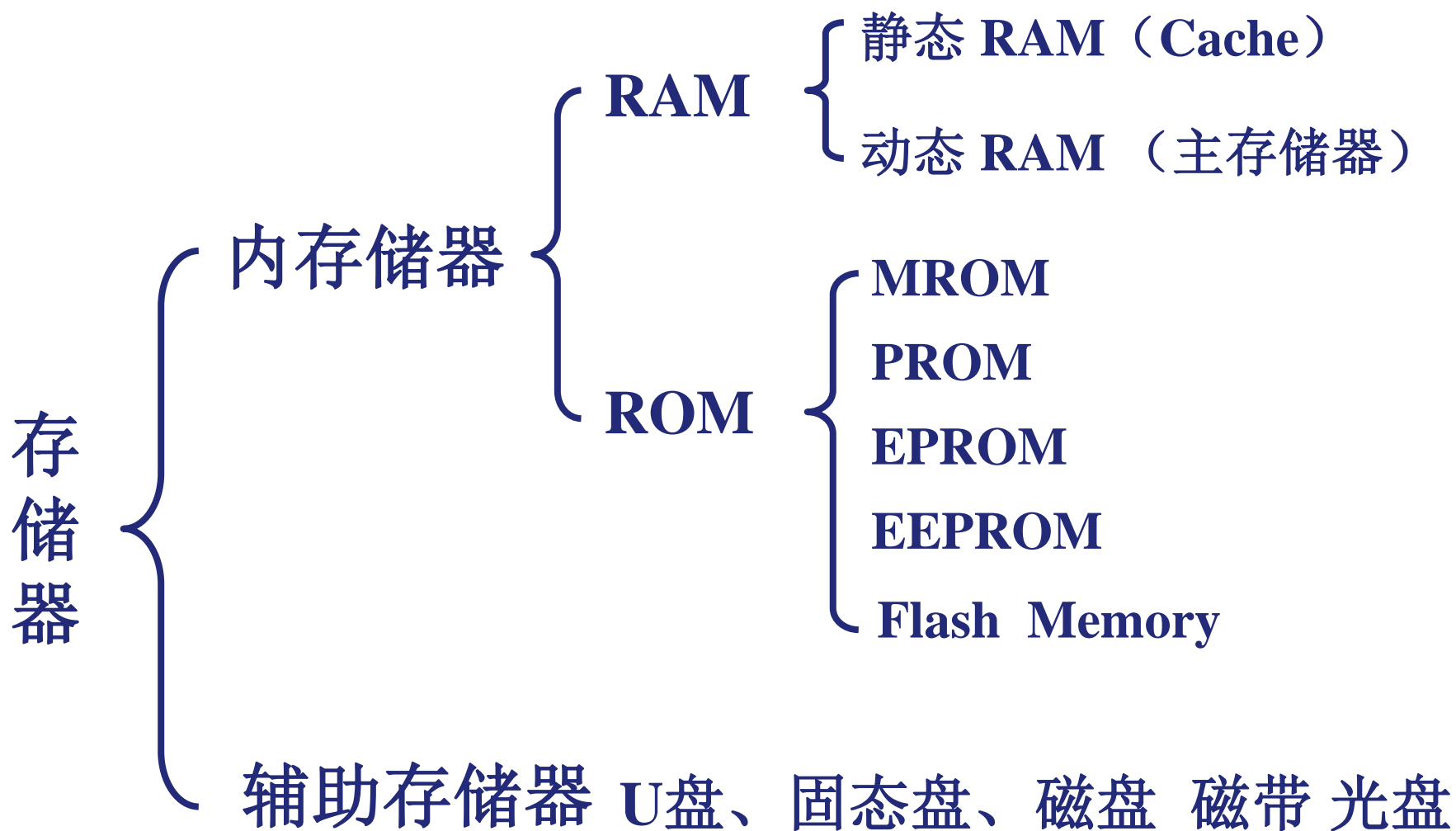
- 直接定位到要读写的数据块，在读写某个数据块时按顺序进行。例如：磁盘。

#### ◆ 相联存储器 (Associate Memory/Content Addressed Memory CAM)

- 按内容检索到存储位置进行读写。例如：快表。

### 3. 按在计算机中的作用分类

4.1



## 4. 按断电后信息的可保存性分类

- 非易失（不挥发）性存储器(Nonvolatile Memory)
  - 信息可一直保留， 不需电源维持。  
(如： ROM、磁表面存储器、光存储器等)
- 易失（挥发）性存储器(Volatile Memory)
  - 电源关闭时信息自动丢失。（如： RAM、Cache等）



## 5. 按功能/容量/速度/所在位置分类

- 寄存器(Register)

- 封装在CPU内，用于存放当前正在执行的指令和使用的数据
- 用触发器实现，速度快，容量小（几十个）

- 高速缓存(Cache)

- 位于CPU内部或附近，用来存放当前要执行的局部程序段和数据
- 用SRAM实现，速度可与CPU匹配，容量小（几MB）

- 内存存储器MM（主存储器Main (Primary) Memory）

- 位于CPU之外，用来存放已被启动的程序及所用的数据
- 用DRAM实现，速度较快，容量较大（几GB）

- 外存储器AM（辅助存储器Auxiliary / Secondary Storage）

- 位于主机之外，用来存放暂不运行的程序、数据或存档文件
- 用磁表面或光存储器实现，容量大而速度慢

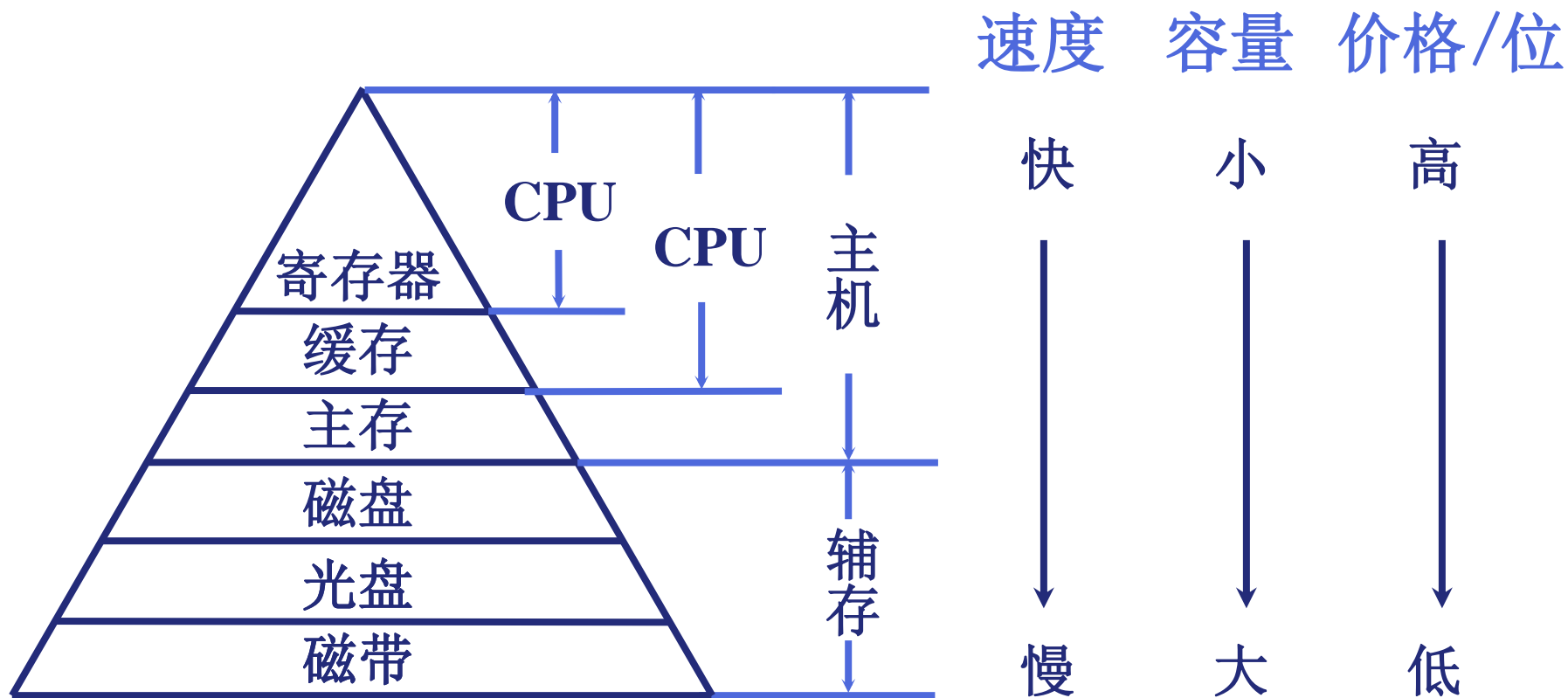
# 内存条（主存）



## 二、存储器的层次结构

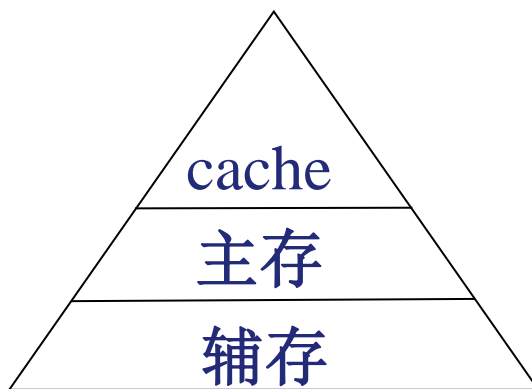
4.1

### 1. 存储器三个主要特性的关系

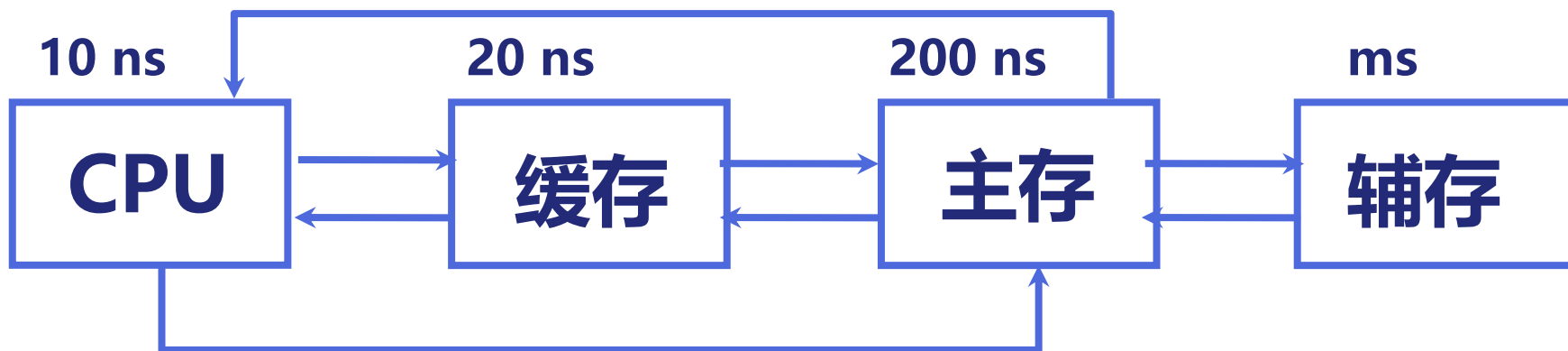


## 2. 三级存储器结构

- 由高速缓冲存储器、主存储器和外存储器组成
  - 内存存储器：CPU能直接访问的存储器
    - 包括高速缓冲存储器和主存储器
  - 外存储器：CPU不能直接访问的存储器，外存储器的信息必须调入内存存储器后才能由CPU进行处理



# 缓存—主存层次和主存—辅存层次



(速度)                      (容量)  
缓存 — 主存      主存 — 辅存

主存储器

虚拟存储器

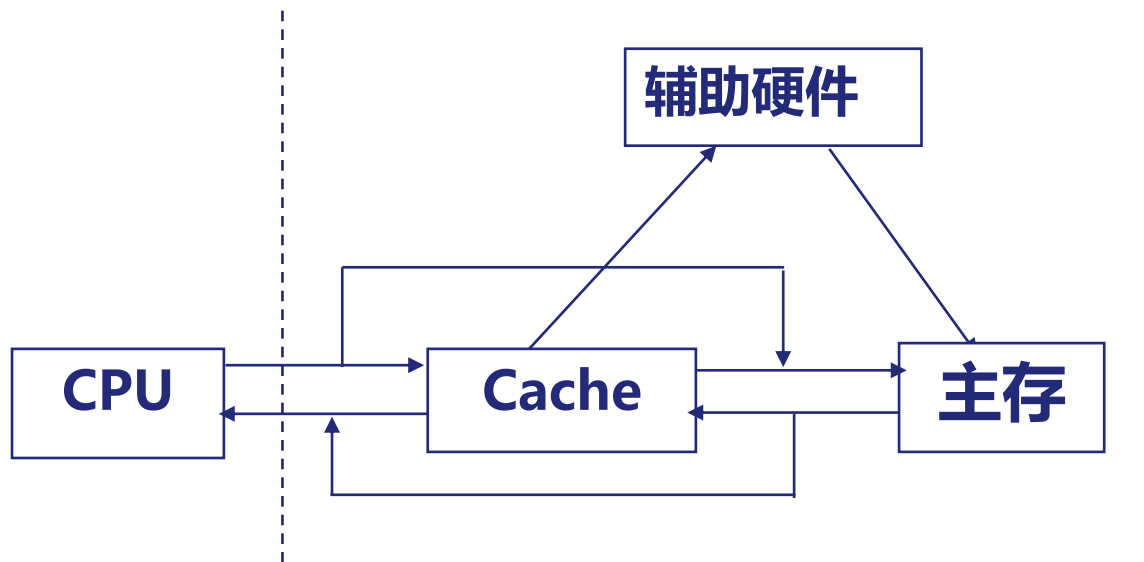
实地址

虚地址

物理地址

逻辑地址

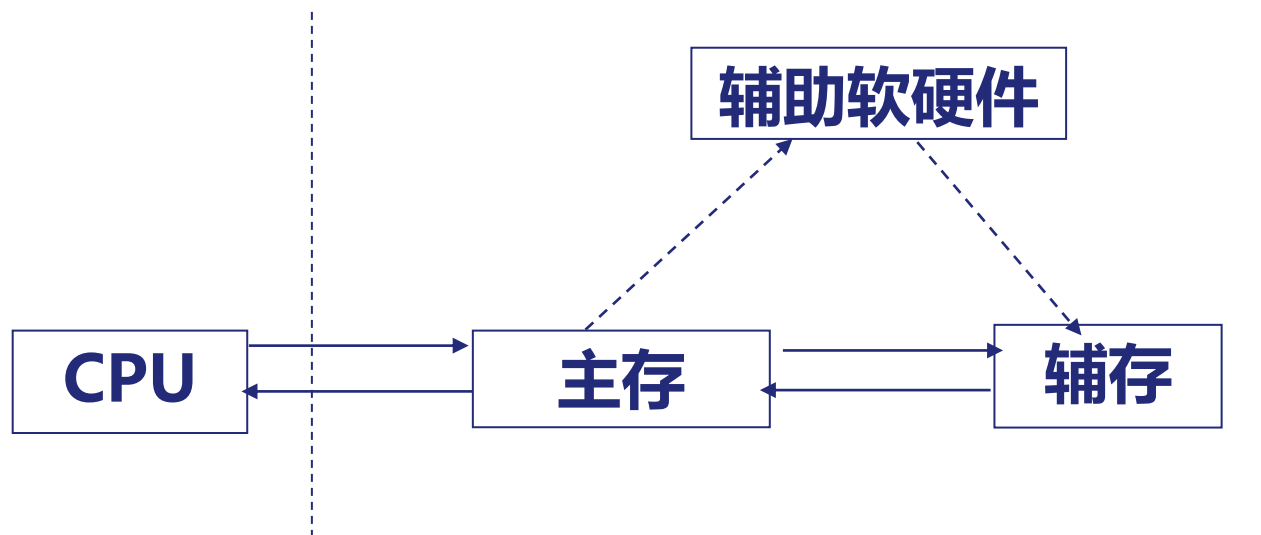
# Cache-主存存储层次（Cache存储系统）



- Cache存储系统是为了解决主存速度不足而提出来的。
- 在Cache和主存之间增加辅助硬件，让它们构成一个整体
- 从CPU看，速度接近Cache，容量是主存的容量，位价格 接近于主存价格
- 由于Cache存储系统全部采用硬件来调度，因此对系统程序员和应用程序员是透明的



# 主存 - 辅存存储层次（虚拟存储系统）

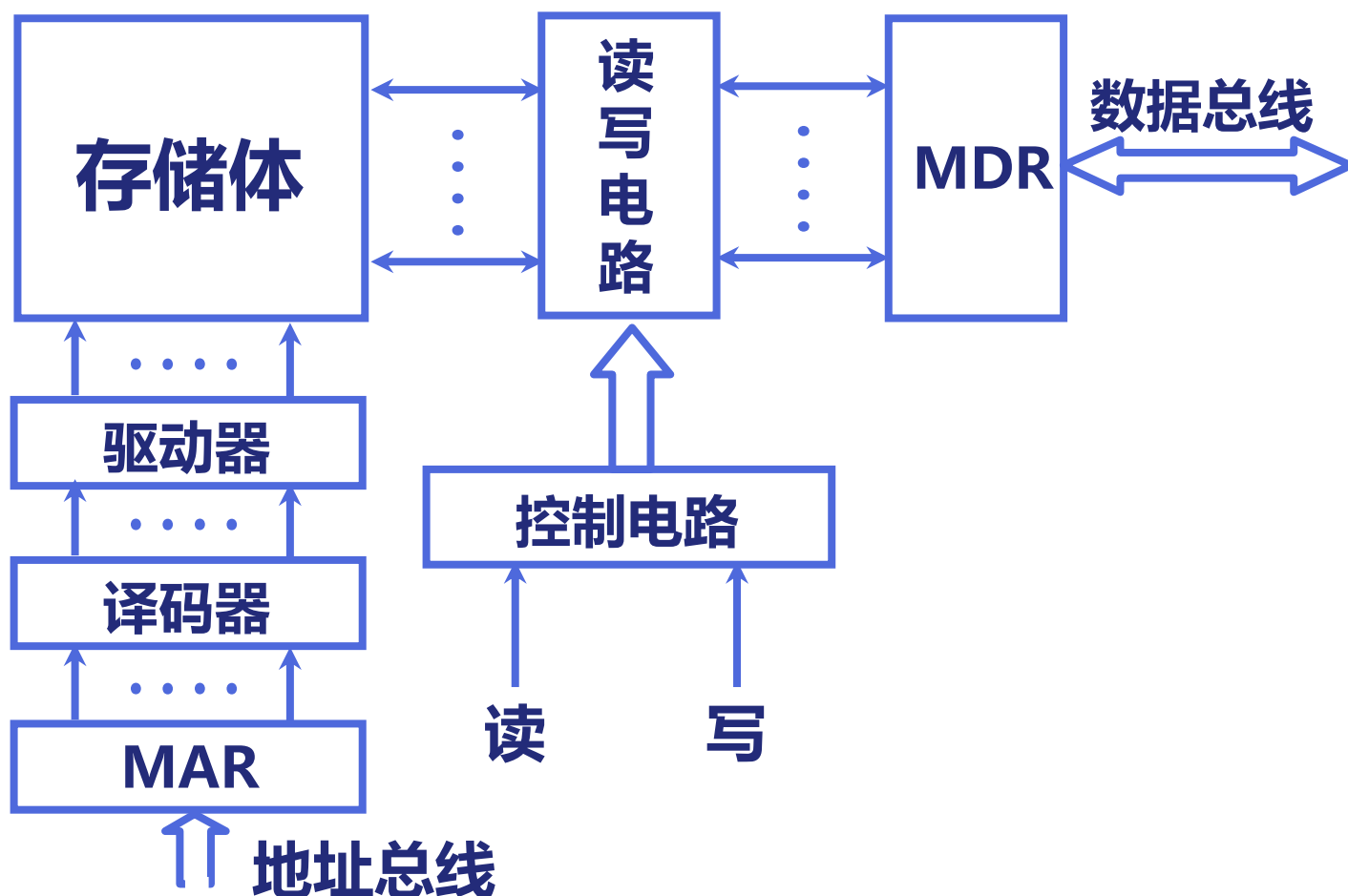


- 虚拟存储器是为了解决主存容量不足而提出来的。
- 在主存和辅存之间，增加辅存的软硬件，让它们构成一个整体
- 从CPU看，速度接近主存的速度，容量是虚拟的地址空间，每位价格是接近辅存的价格
- 由于虚拟存储系统需要通过操作系统来调度，因此对系统程序员是不透明的，但对应用程序员是透明的

## 4.2 主存储器

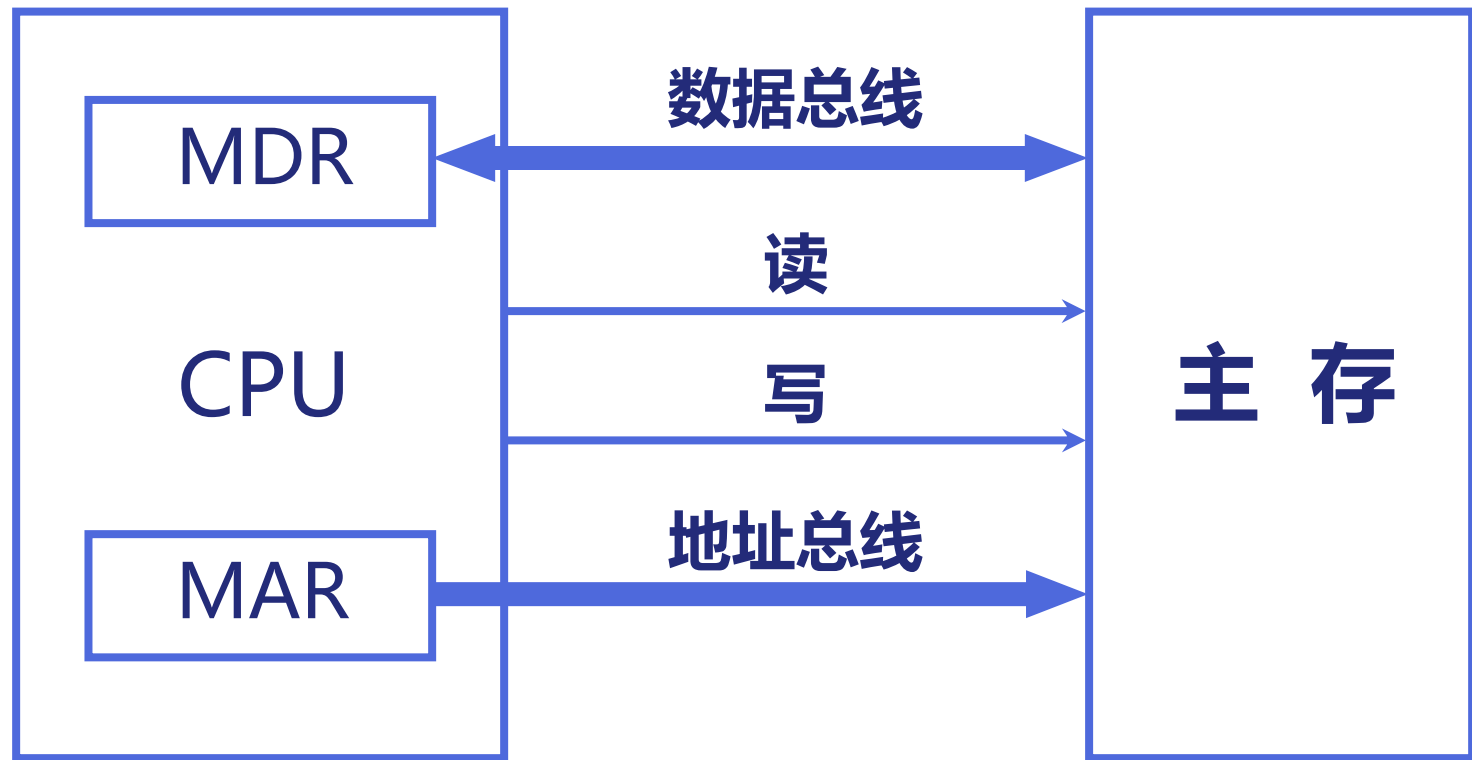
### 一、概述

#### 1. 主存的基本组成





## 2. 主存和 CPU 的联系



### 3. 主存中存储单元地址的分配

高位字节 地址为字地址

低位字节 地址为字地址

字地址	字节地址			
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11

字地址	字节地址	
0	1	0
2	3	2
4	5	4

设地址线 24 根

若字长为 16 位

若字长为 32 位

按 字节 寻址  $2^{24} = 16 \text{ M}$

按 字 寻址 8 M

按 字 寻址 4 M

# 数据的存储和排列顺序

- 80年代开始，几乎所有机器都用字节编址
- ISA设计时要考虑的两个问题：
  - 如何从一个字节地址中取到一个32位的字？ - 字的存放问题
  - 一个字能否存放在任何字节边界？ - 字的边界对齐问题

例如，若  $\text{int } i = 0x01234567$ ，存放在内存100号单元，则用“取数”指令访问100号单元取出  $i$  时，必须清楚  $i$  的4个字节是如何存放的。

Word:	01      23      45      67				little endian word 100
	103	102	101	100	
	msb			lsb	
100    101    102    103				big endian word 100	

大端方式（**Big Endian**）：**MSB**所在的地址是数的地址

e.g. **IBM 360/370, Motorola 68k, MIPS, Sparc, HP PA**

小端方式（**Little Endian**）：**LSB**所在的地址是数的地址

e.g. **Intel 80x86, DEC VAX**

有些机器两种方式都支持，可以通过特定的控制位来设定采用哪种方式。

# BIG Endian versus Little Endian

**Ex1: Memory layout of a number ABCDH located in 1000**

In Big Endian:	→	CD	1001
		AB	1000
In Little Endian:	→	AB	1001
		CD	1000

**Ex2: Memory layout of a number 00ABCDEFH located in 1000**

In Big Endian:	→	00	1000
		AB	1001
		CD	1002
		EF	1003
In Little Endian:	→	00	1003
		AB	1002
		CD	1001
		EF	1000

# BIG Endian versus Little Endian

Ex3: Memory layout of a instruction located in 1000

假定小端机器中指令: **mov AX, 0x12345(BX)**

其中操作码**mov**为**40H**, 寄存器**AX**和**BX**分别为**0001B**和**0010B**,  
立即数占**32**位, 则存放顺序为:

<b>MOV</b>	<b>AX</b>	<b>BX</b>	<b>00012345H</b>
------------	-----------	-----------	------------------

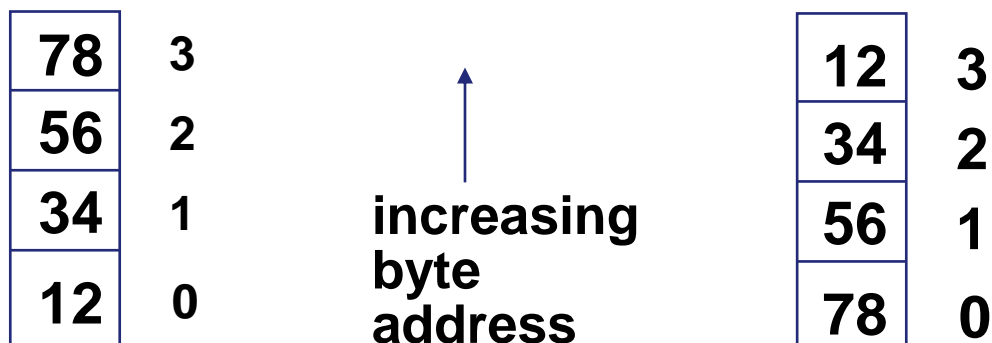
<b>40</b>	<b>1</b>	<b>2</b>	<b>45 23 01 00</b>
-----------	----------	----------	--------------------

若在大端机器上, 则存放顺序如何?

<b>40</b>	<b>1</b>	<b>2</b>	<b>00 01 23 45</b>
-----------	----------	----------	--------------------

<b>00</b>	<b>1005</b>	<b>45</b>
<b>01</b>	<b>1004</b>	<b>23</b>
<b>23</b>	<b>1003</b>	<b>01</b>
<b>45</b>	<b>1002</b>	<b>00</b>
<b>12</b>	<b>1001</b>	<b>12</b>
<b>40</b>	<b>1000</b>	<b>40</b>
<b>地址</b>		

# Byte Swap Problem (字节交换问题)



Big Endian

Little Endian

上述存放在0号单元的数据（字）是什么？ **12345678H**? **78563412H**?

存放方式不同的机器间程序移植或数据通信时，会发生什么问题？

- ◆ 每个系统内部是一致的，但在系统间通信时可能会发生问题！
- ◆ 因为顺序不同，需要进行顺序转换

音、视频和图像等文件格式或处理程序都涉及到字节顺序问题

**ex. Little endian: GIF, PC Paintbrush, Microsoft RTF, etc**

**Big endian: Adobe Photoshop, JPEG, MacPaint, etc**

# Alignment(对齐)

**Alignment:** 要求数据的地址是相应的边界地址

- 目前机器字长一般为32位或64位，而存储器地址按字节编址
- 指令系统支持对字节、半字、字及双字的运算，也有位处理指令
- 各种不同长度的数据存放时，有两种处理方式：
  - 按边界对齐 （假定字的宽度为32位，按字节编址）
    - 字地址：4的倍数（低两位为0）
    - 半字地址：2的倍数（低位为0）
    - 字节地址：任意
  - 不按边界对齐

**坏处：可能会增加访存次数！**

# Alignment(对齐)

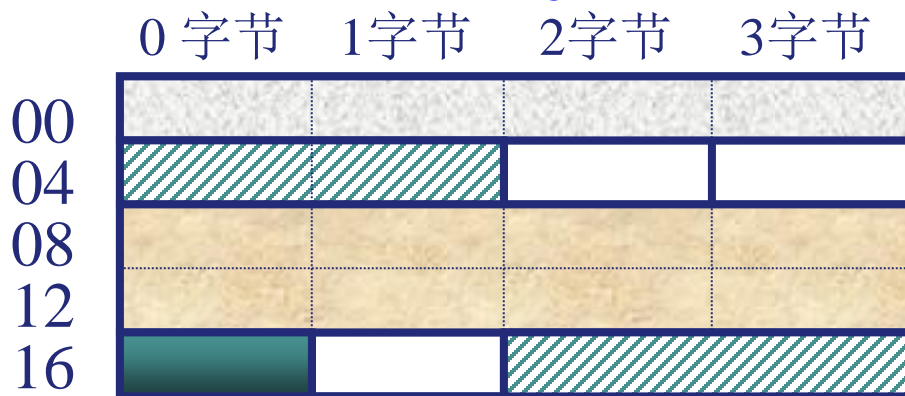
示例 假设数据顺序：字-半字-双字-字节-半字-.....

如：int i, short k, double x, char c, short j,.....

按边界对齐

x: 2个周期

j: 1个周期

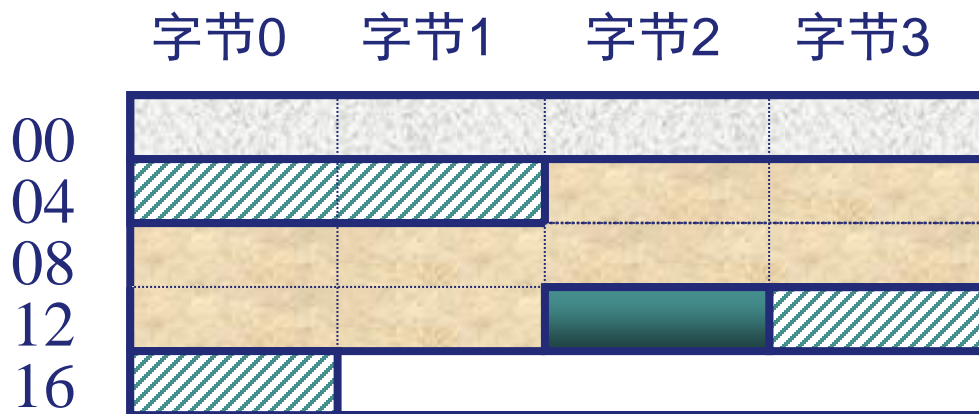


则：&i=0; &k=4; &x=8; &c=16; &j=18;.....

边界不对齐

x: 3个周期

j: 2个周期



增加了访存次数！

则：&i=0; &k=4; &x=6; &c=14; &j=15;.....



## 4. 主存的性能指标

### (1) 存储容量 主存 存放二进制代码的总数量

- 以字编址的计算机：字数与字长的乘积

**存储容量 = 存储单元个数 × 存储字长**

64K×16位：64K个存储单元，每个存储单元字长为16位

- 以字节编址的计算机，以字节数来表示

**存储容量 = 存储单元个数 × 存储字长 / 8**

64K×16位用字节数表示：128K字节 (128KB)

# 4. 主存的技术指标

## (2) 存储速度

- 存取时间 $T_A$  (Memory Access Time) :从CPU送出内存单元的地址码开始, 到主存读出数据并送到CPU (或者是把CPU数据写入主存) 所需要的时间 (单位: ns,  $1\text{ ns} = 10^{-9}\text{ s}$ )

读出时间 写入时间

- 存取周期 $T_{MC}$  (Memory Cycle Time) :连读两次访问存储器所需的最小时间间隔, 它应等于存取时间加上下一存取开始前所要求的附加时间, 因此,  $T_{MC}$ 比 $T_A$ 大 ( 因为存储器由于读出放大器、驱动电路等都有一段稳定恢复时间, 所以读出后不能立即进行下一次访问。)

读周期 写周期

- $T_{MC} > T_A$ , 因为存储器在读写操作后, 要有一段恢复内部状态的复原时间。对于破坏性读出的RAM, 存取周期往往比存取时间要大的多, 甚至可达到 $T_{MC} = 2T_A$ , 这是因为存储器中的信息读出后需要马上进行再生 (重写)。

## 4. 主存的技术指标

**(3) 存储器的带宽:单位时间内存储器存取的信息量**

**单位: 字/秒, 字节/秒, 位/秒**

- **提高主存的带宽的措施**

- 缩短存取周期
- 增加存储字长
- 增加存储体

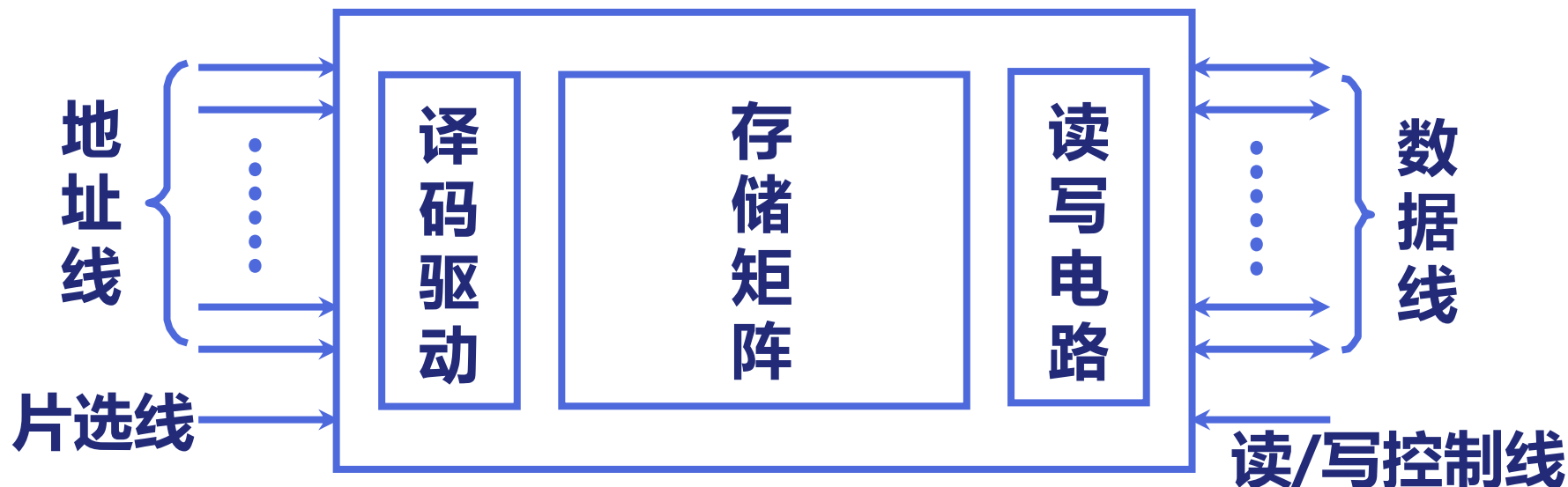
## 4. 主存的技术指标

**(4) 可靠性：**指在规定时间内，存储器无故障读写的概率。

- 平均无故障时间（Mean Time Between Failures, MTBF）用于衡量可靠性。
- MTBF为两次故障之间的平均时间间隔。MTBF越长，说明存储器可靠性越高。

**(5) 功耗：**功耗既反映耗电又反映其发热程度。

### 1. 半导体存储芯片的基本结构



地址线 (单向)

数据线 (双向)

芯片容量

10

4

1K × 4位

14

1

16K × 1位

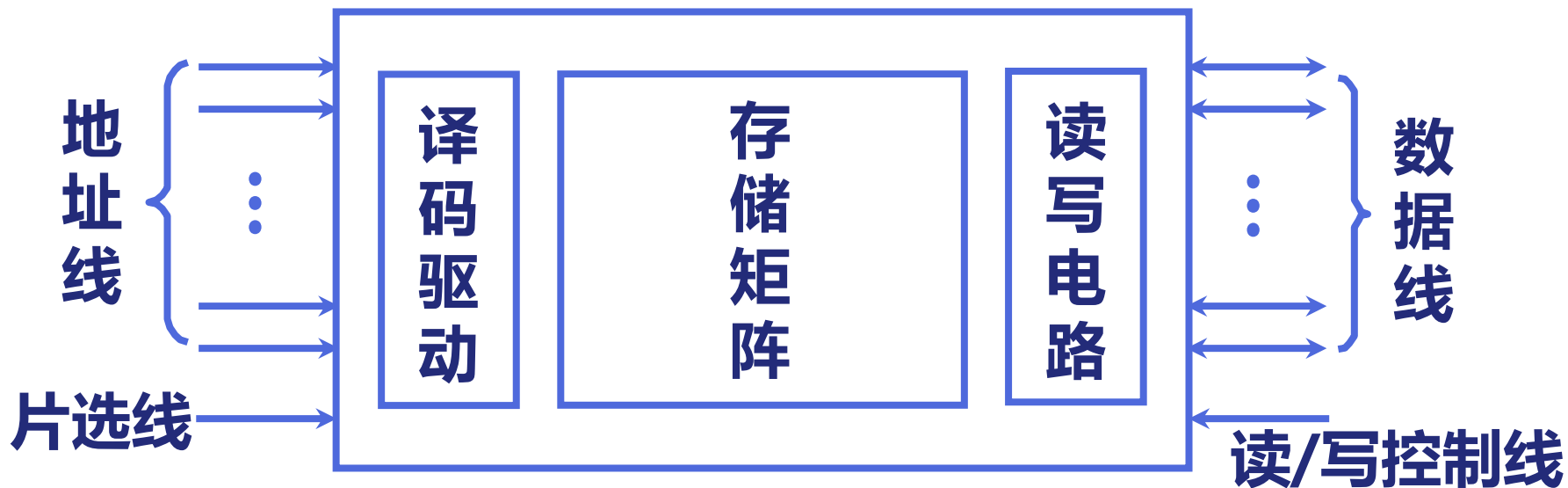
13

8

8K × 8位

## 二、半导体存储芯片简介

### 1. 半导体存储芯片的基本结构



片选线      $\overline{CS}$       $\overline{CE}$

读/写控制线      $\overline{WE}$  (低电平写 高电平读)

$\overline{OE}$  (允许读)      $\overline{WE}$  (允许写)

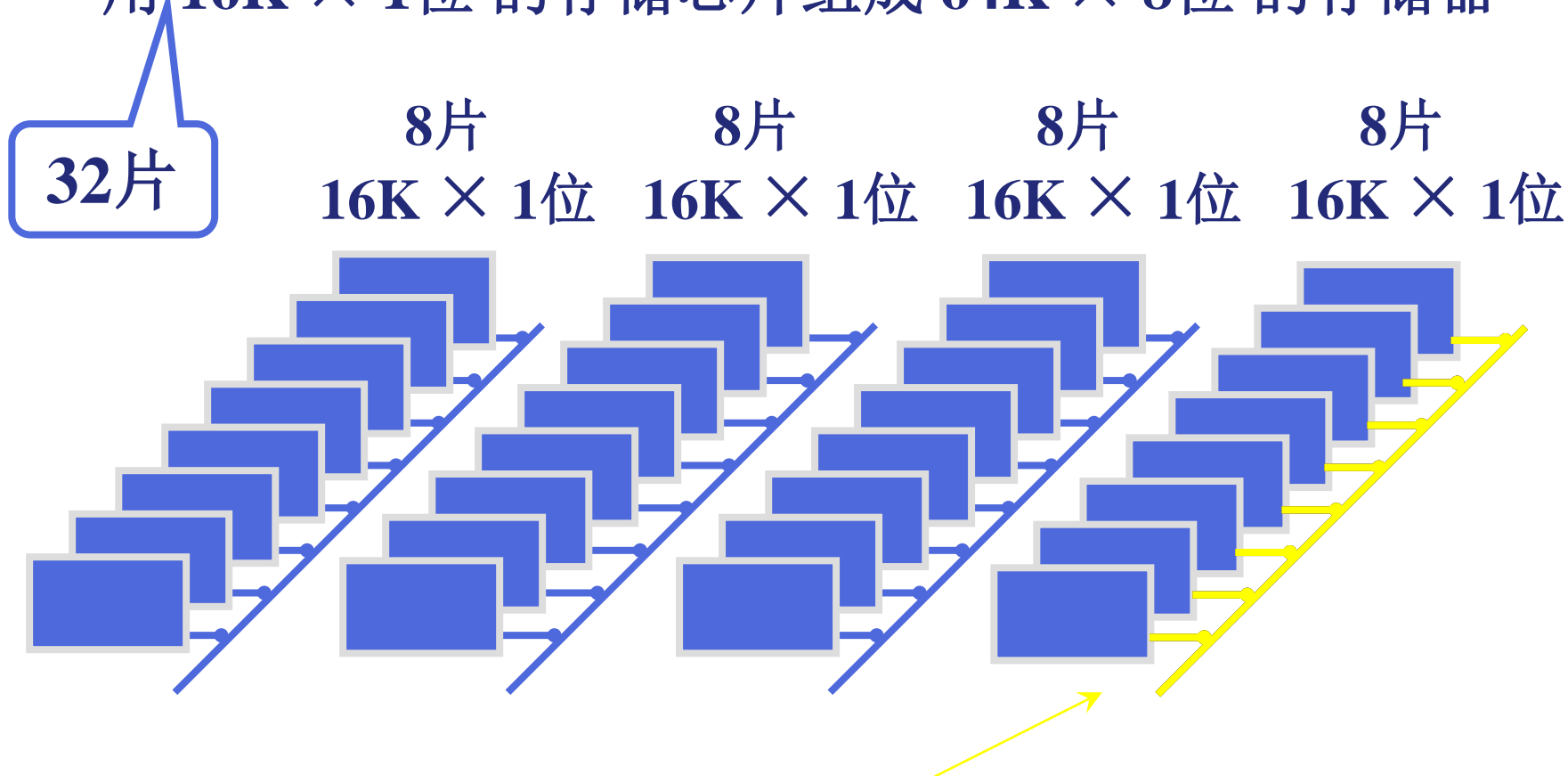
# 存储芯片片选线的作用

- 由于一块集成芯片容量有限，要组成一个大的存储器一般需要将多个芯片连接使用。
- 在使用存储器某个地址时，某些芯片需要使用，其它芯片暂时不用，即片选。
- 当芯片的片选信号 ( $\overline{CS}$ ) 有效时，芯片被选中，此片所连的地址线才有效，才能对它进行读写操作

# 存储芯片片选线的作用

4.2

用  $16\text{K} \times 1$  位的存储芯片组成  $64\text{K} \times 8$  位的存储器



当地址为 65 535 时，此 8 片的片选有效



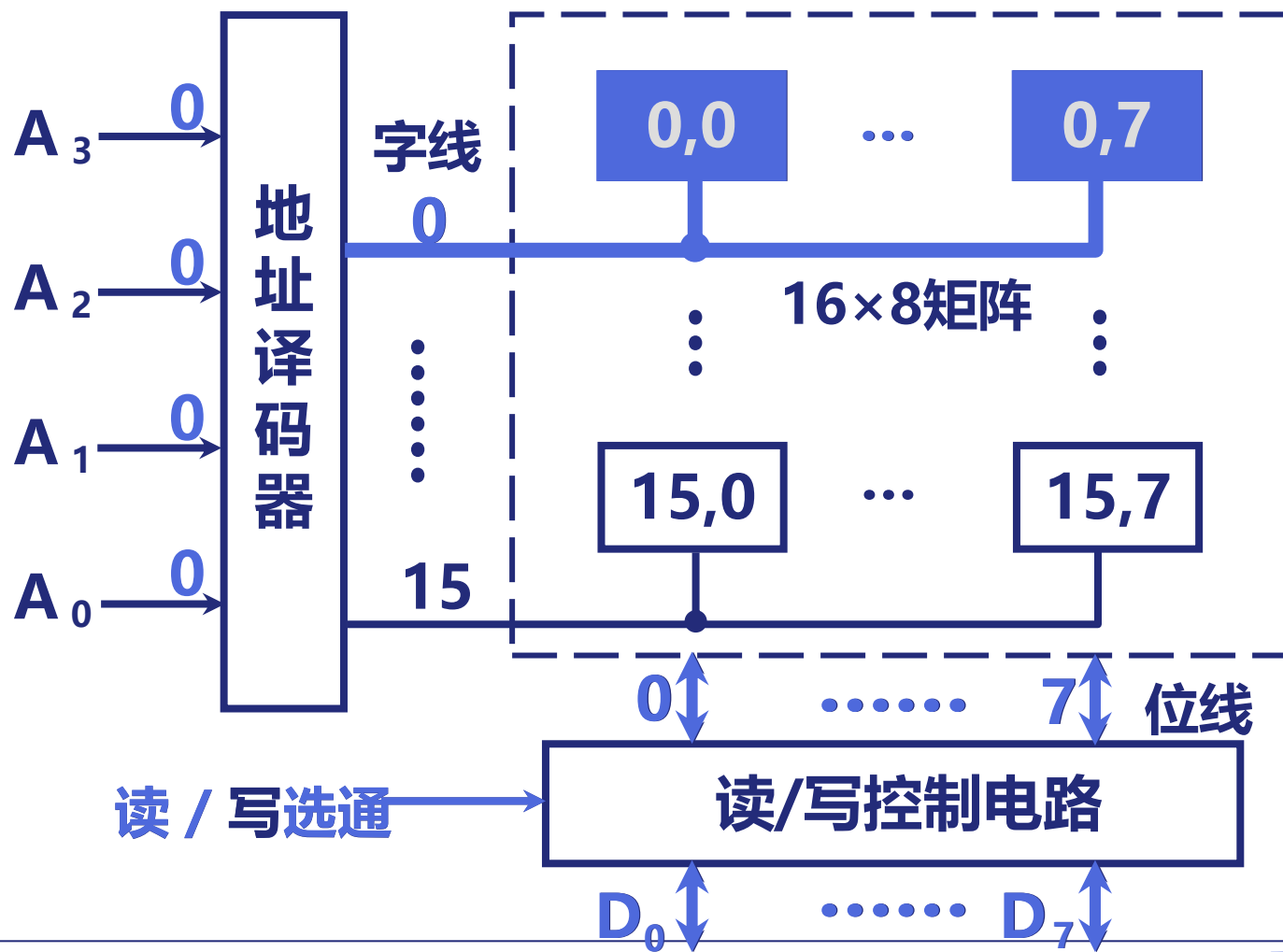


## 2. 半导体存储芯片的译码驱动方式

- 地址译码器设计方案

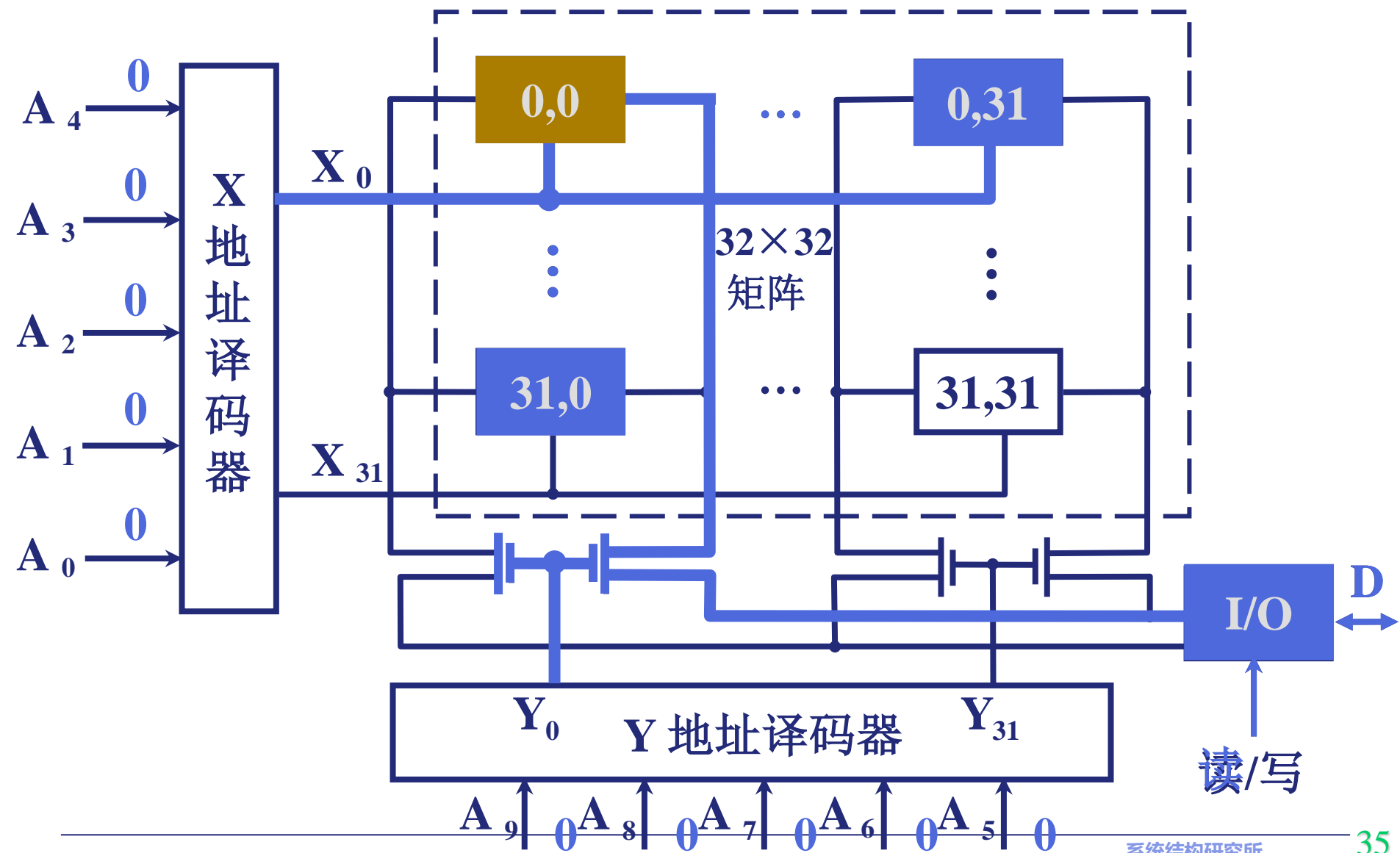
- 线选法（单译码）：地址译码器只有一个，用一根字线直接选中一个存储单元的各位
  - 当地址线数较大，译码器复杂、开销大，使得存储器成本迅速增加，性能下降
- 重合法（双译码）：X地址译码器、Y地址译码器

### (1) 线选法



## (2) 重合法

4.2

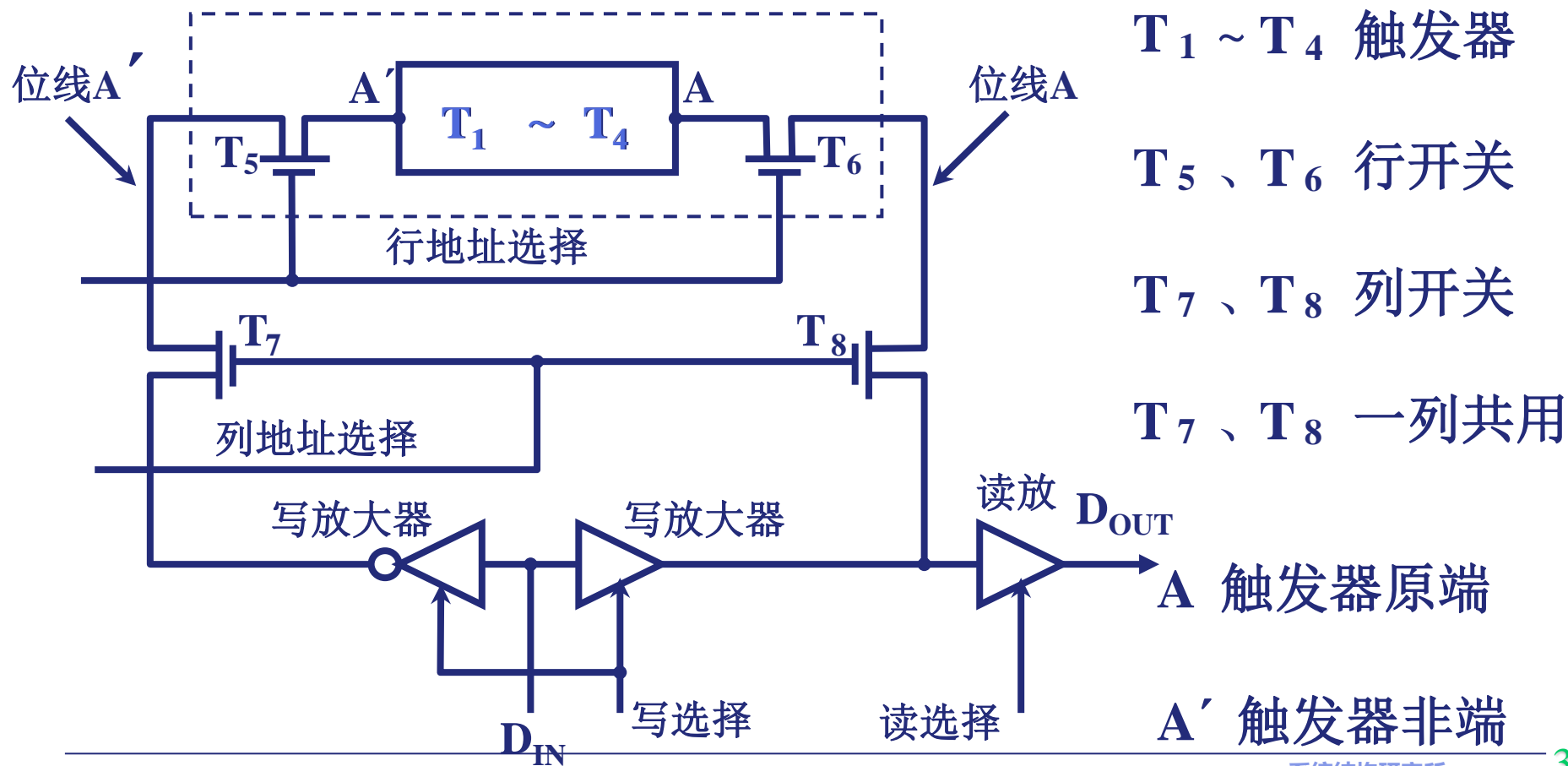


# 三、随机存取存储器 (RAM)

4.2

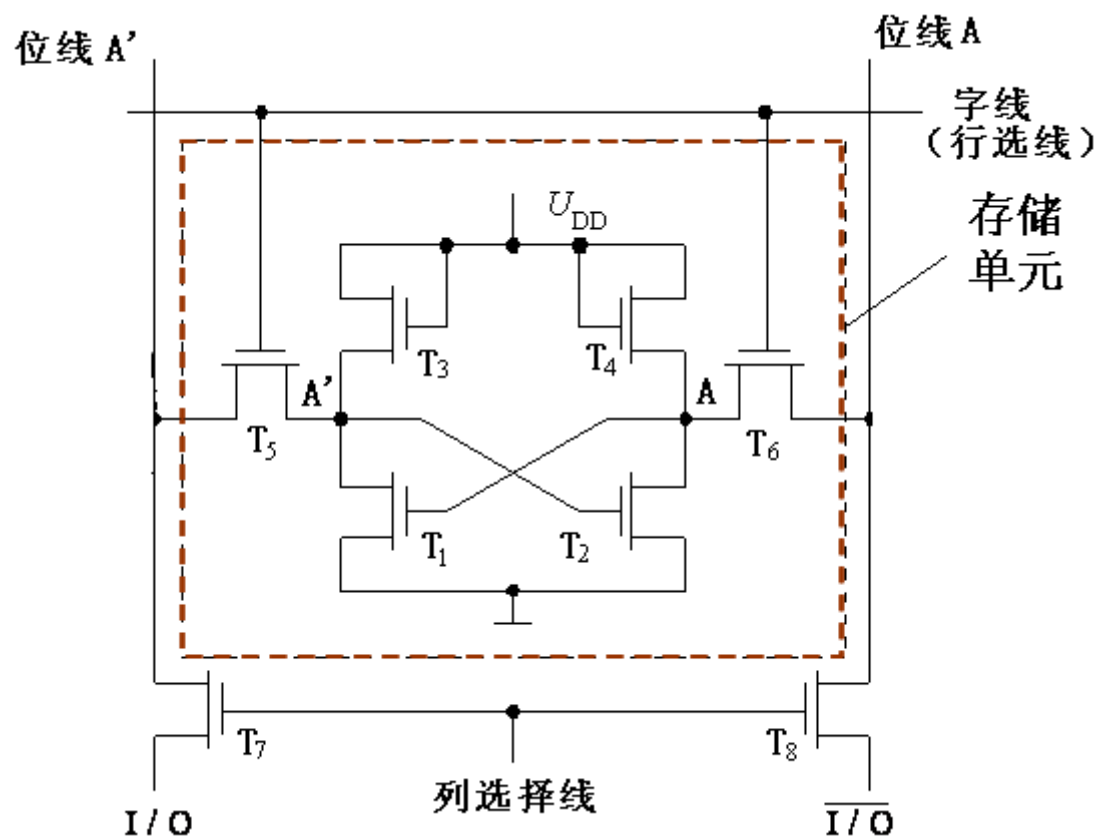
## 1. 静态 RAM (SRAM)

### (1) 静态 RAM 基本电路



# 六管静态MOS管电路

## 6管静态NMOS记忆单元



信息存储原理：看作带时钟的RS触发器

写入时：

- 位线上是被写入的二进制信息0或1
- 置字线为1
- 存储单元(触发器)按位线的状态设置成0或1

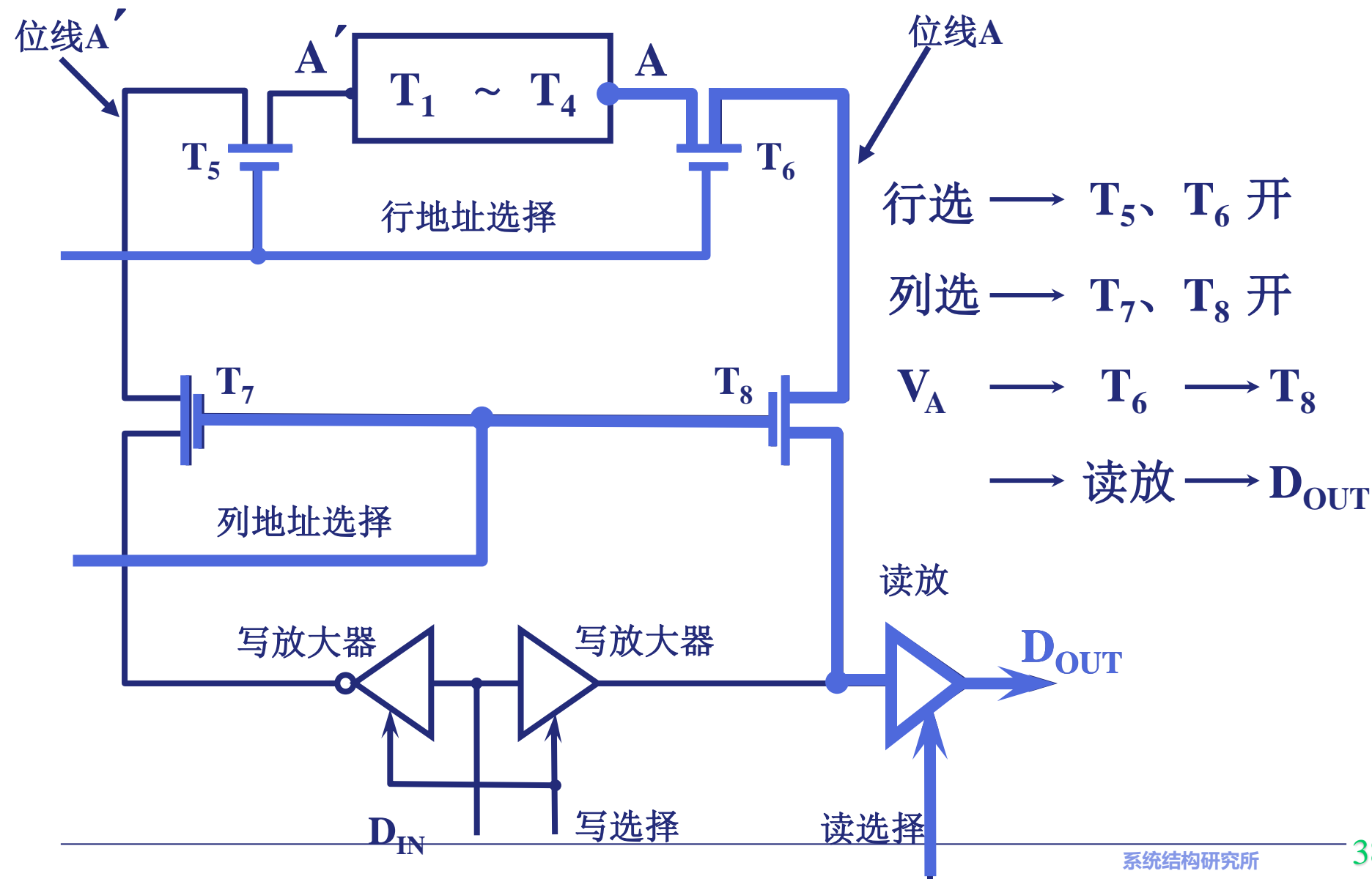
读出时：

- 置2个位线为高电平
- 置字线为1
- 存储单元状态不同，位线的输出不同

SRAM中数据保存在一对正负反馈门电路中，只要供电，数据就一直保持，不是破坏性读出，也无需重写。即：无需刷新！

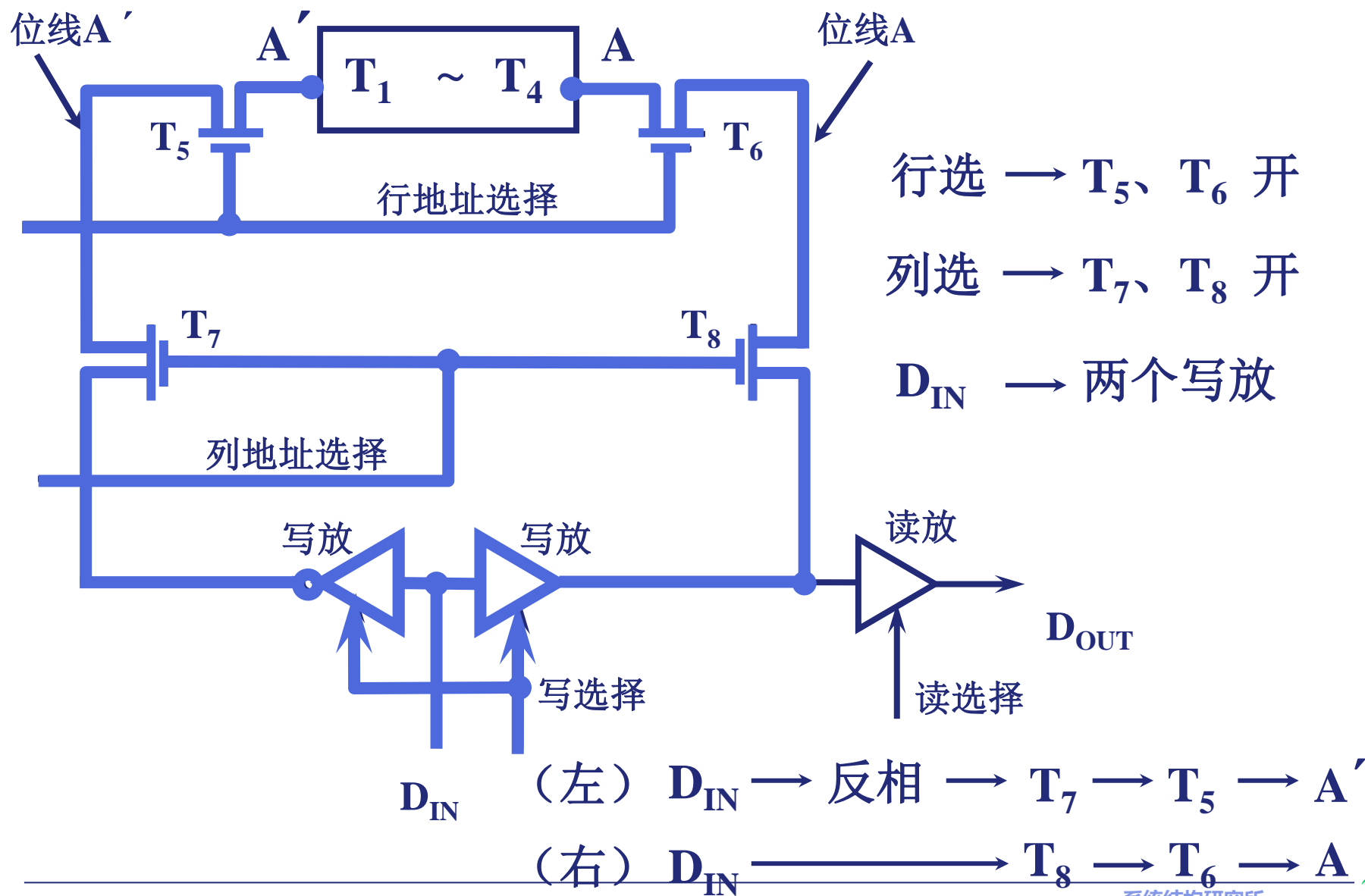
# ① 静态 RAM 基本电路的 读 操作

4.2

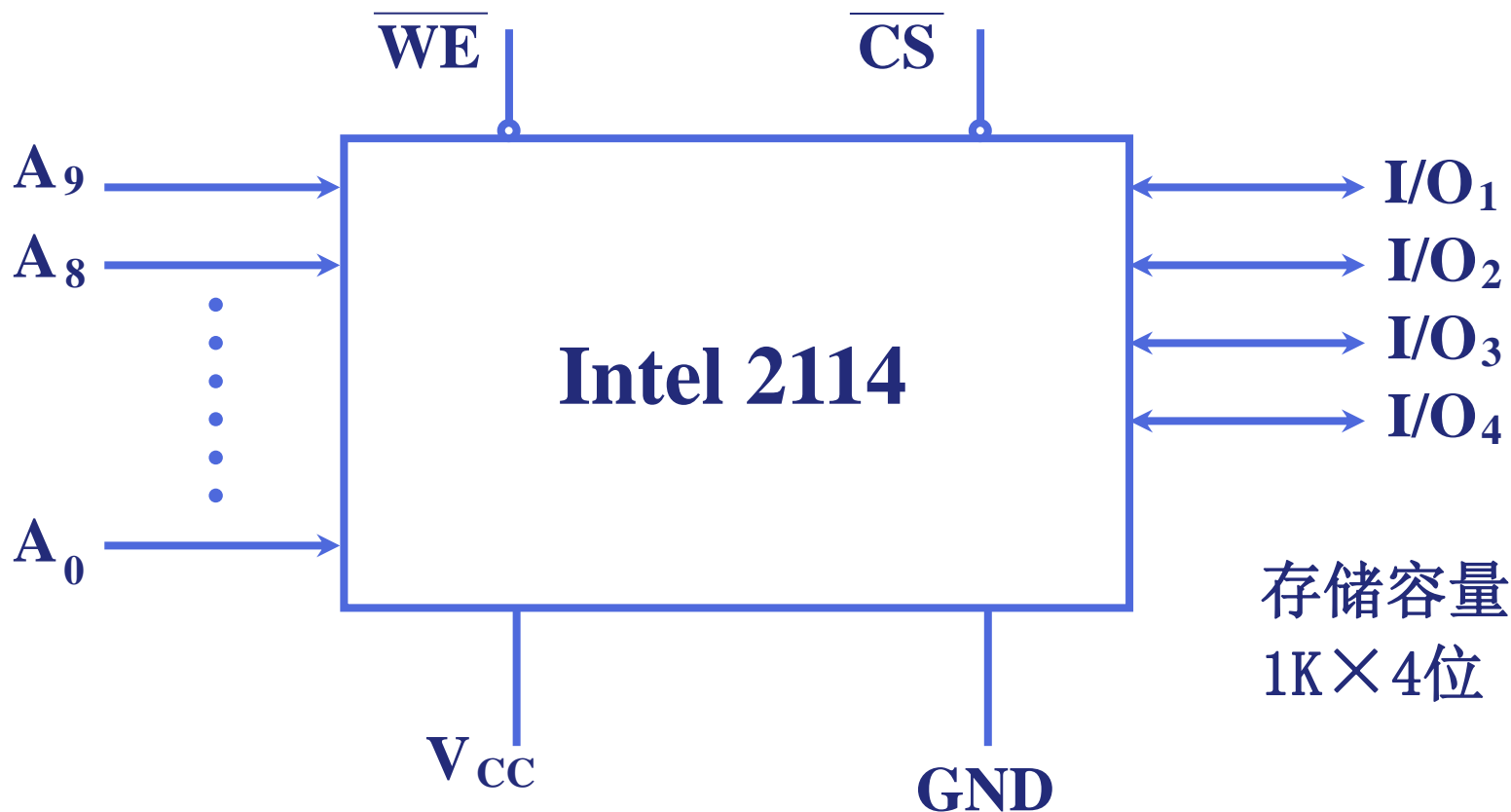


## ② 静态 RAM 基本电路的 写 操作

4.2



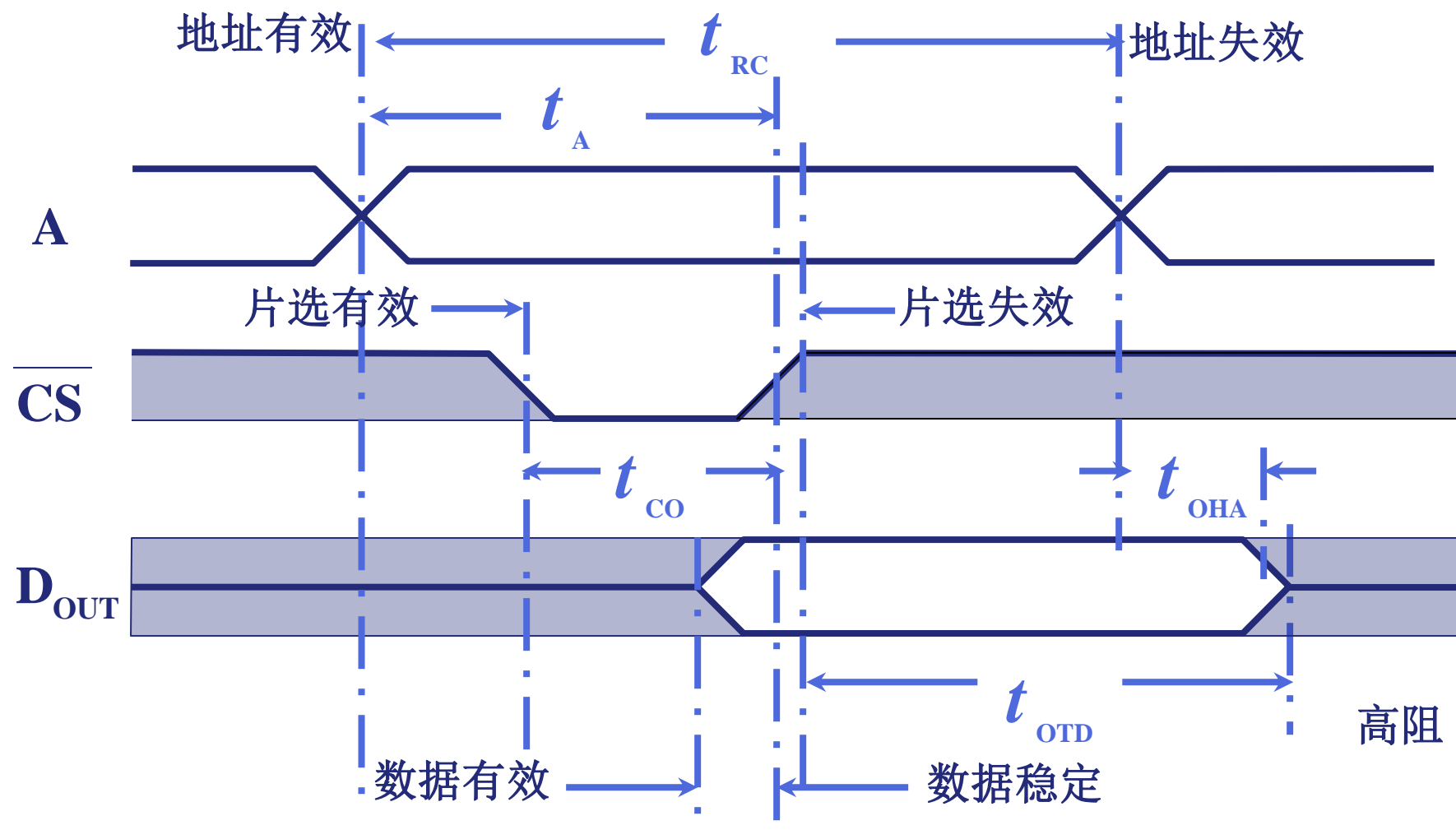
### ① Intel 2114 外特性





### (3) 静态 RAM 读 时序

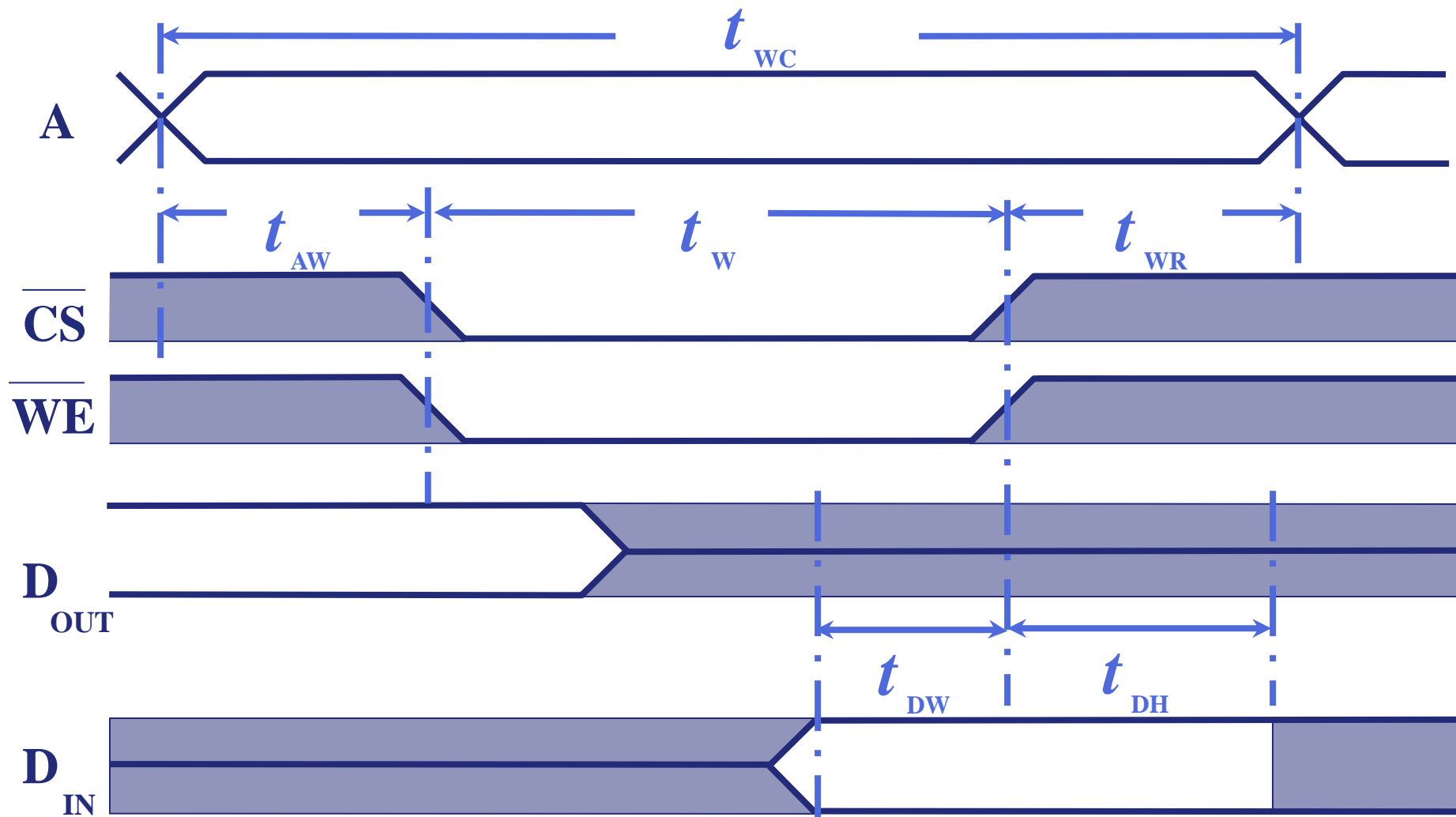
4.2



$t_{OHA}$  地址失效后的数据维持时间

# (4) 静态 RAM (2114) 写时序

4.2

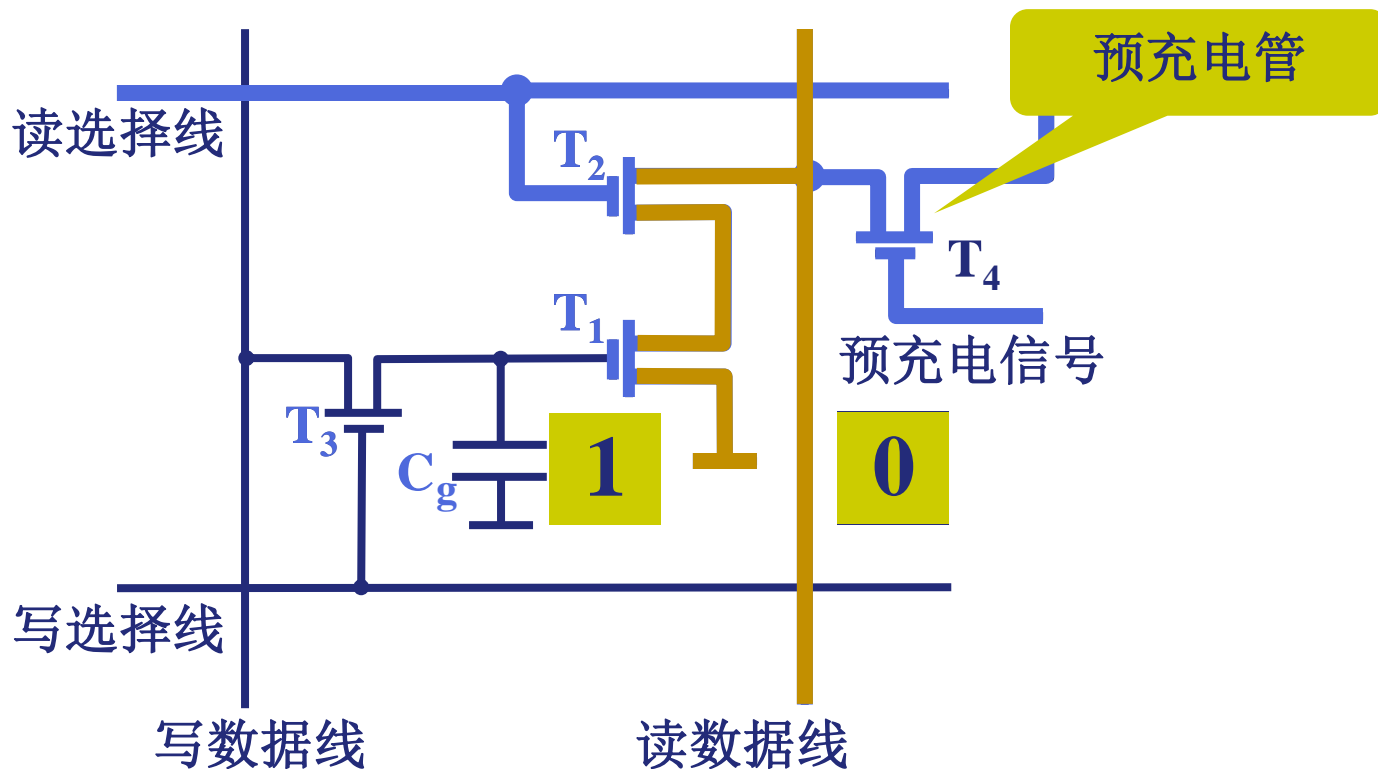


$t_{DH}$   $\overline{WE}$  失效后的数据维持时间

## 2. 动态 RAM ( DRAM )

## 4.2

### (1) 动态 RAM 基本单元电路



读出与原存信息相反

写入与输入信息相同

# (1) 动态 RAM 基本单元电路

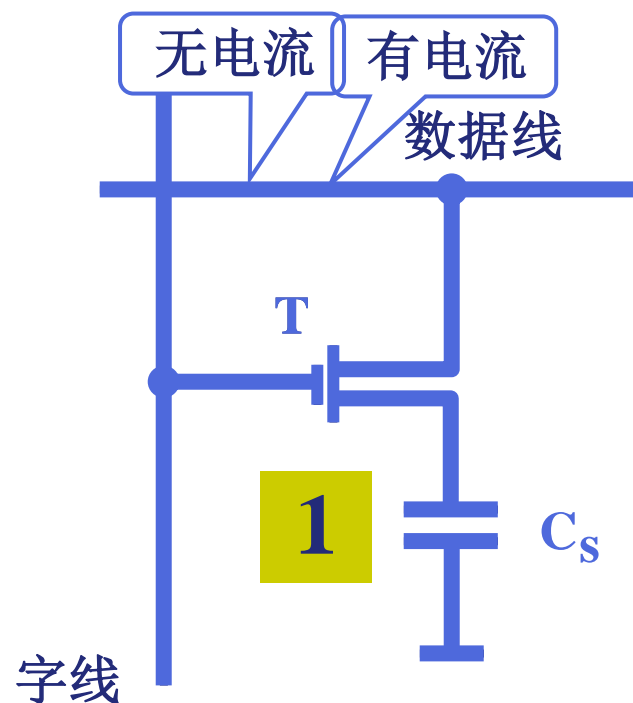
## 4.2

### • 构造和表示:

信息记忆在电容 $C_s$ 上, T为门控管, 控制数据的进出。其栅极接读/写选择线(字线), 漏和源分别接数据线(位线)和记忆电容 $C_s$ 。数据1或0以电容 $C_s$ 上电荷量的有无来判别。

### • 读写原理: 在选择(字)线上加高电平, 使T管导通。

- 写“0”时, 在数据线上加低电平, 使 $C_s$ 上电荷对数据线放电;
- 写“1”时, 在数据线上加高电平, 使数据线对 $C_s$ 充电;
- 读出时, 在数据线上有一读出电压。它与 $C_s$ 上电荷量成正比。



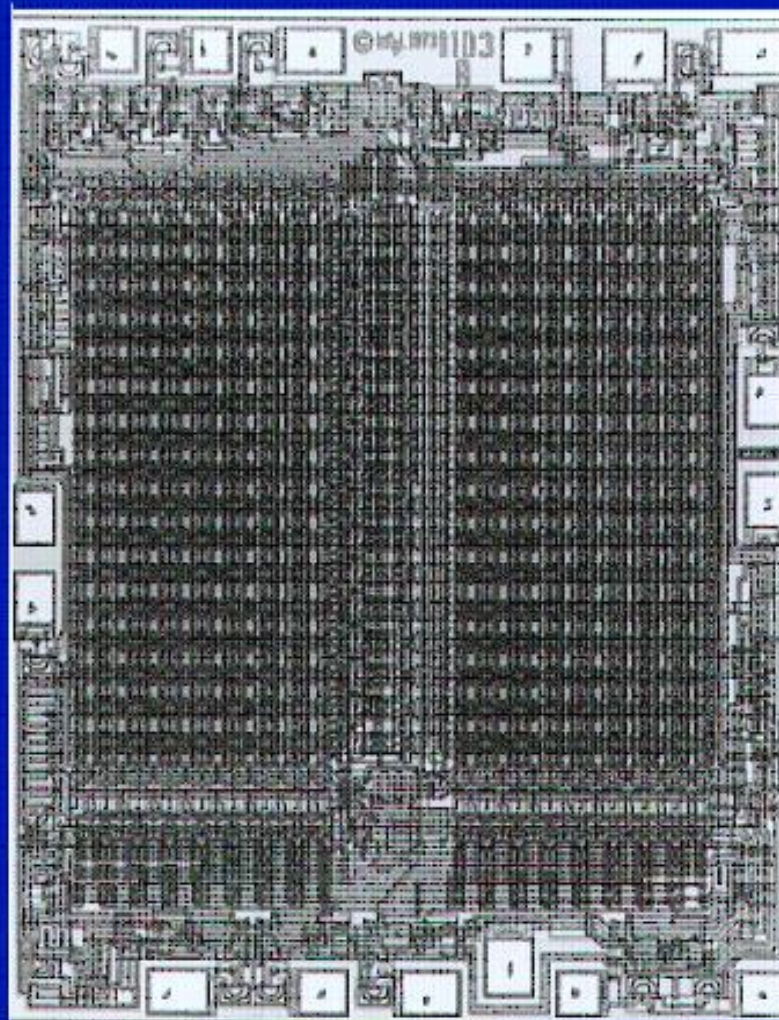
读出时数据线有电流 为 “1”

写入时 $C_s$ 充电 为 “1” 放电 为 “0”



## FIRST DRAM ( 1103 by Intel 1970 )

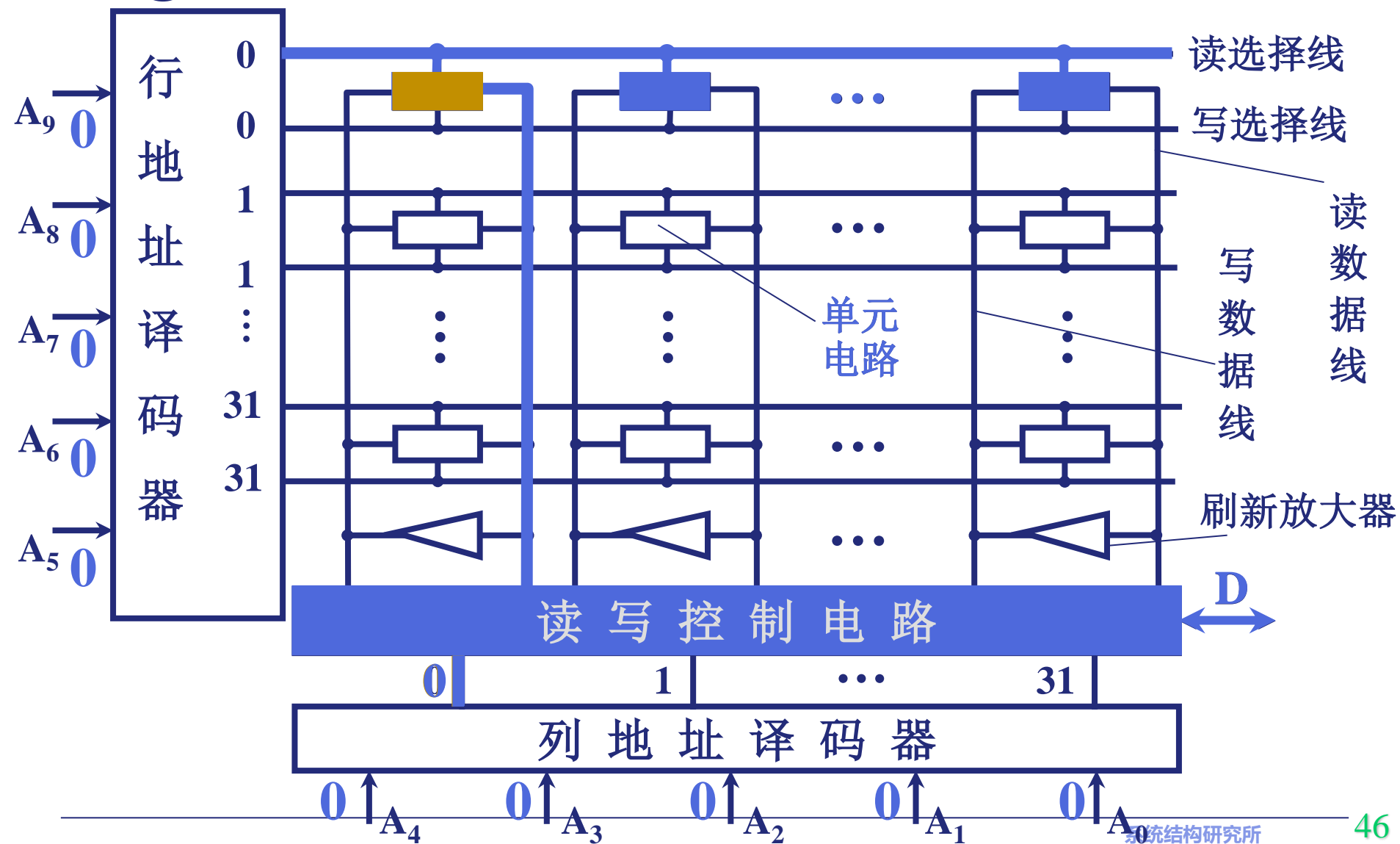
( 5 V      8  $\mu\text{m}$       3  $\text{mm}^2$       2600  $\mu\text{m}^2/\text{cell}$  )



## (2) 动态 RAM 芯片举例

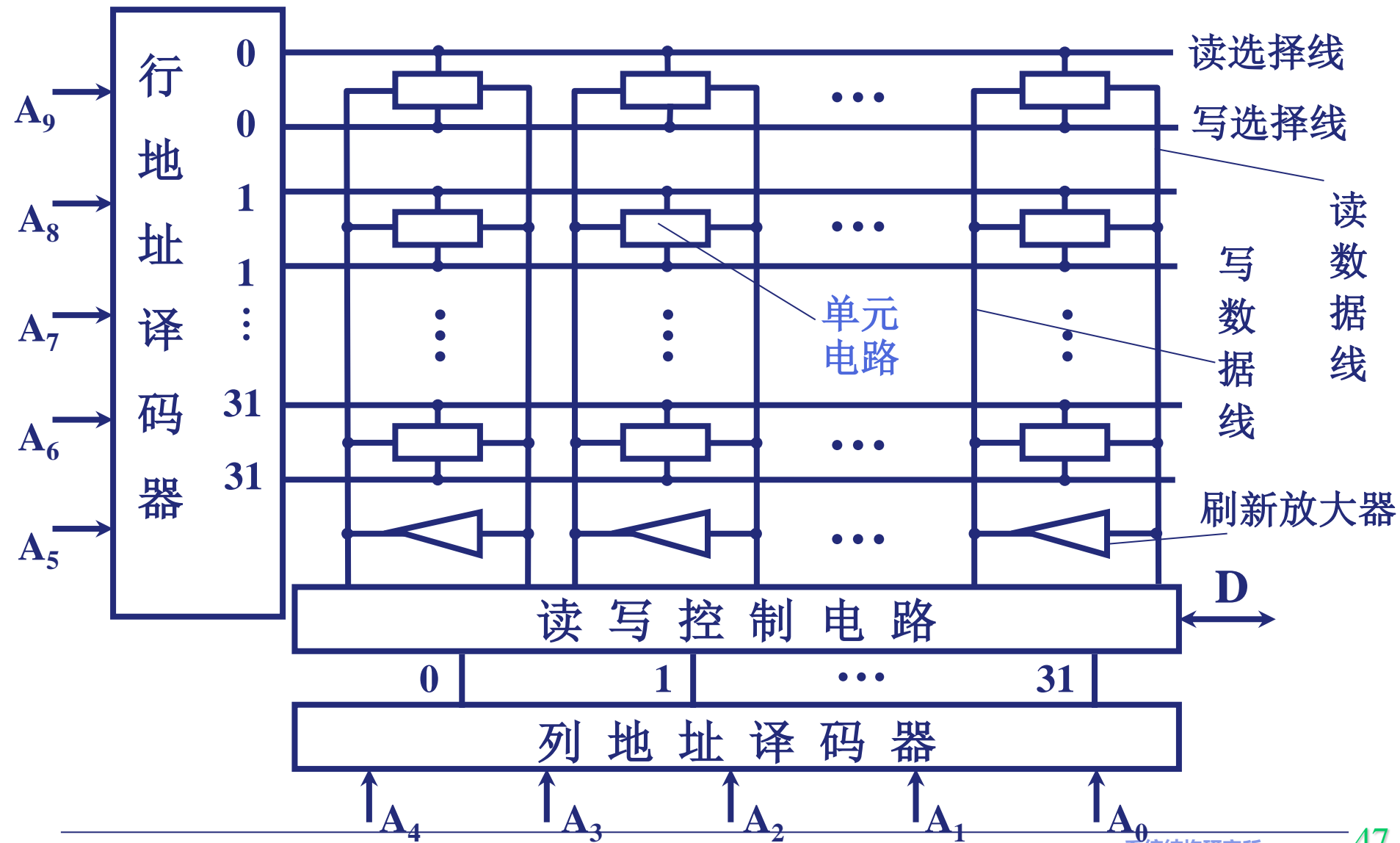
4.2

### ① 三管动态 RAM 芯片 (Intel 1103) 读

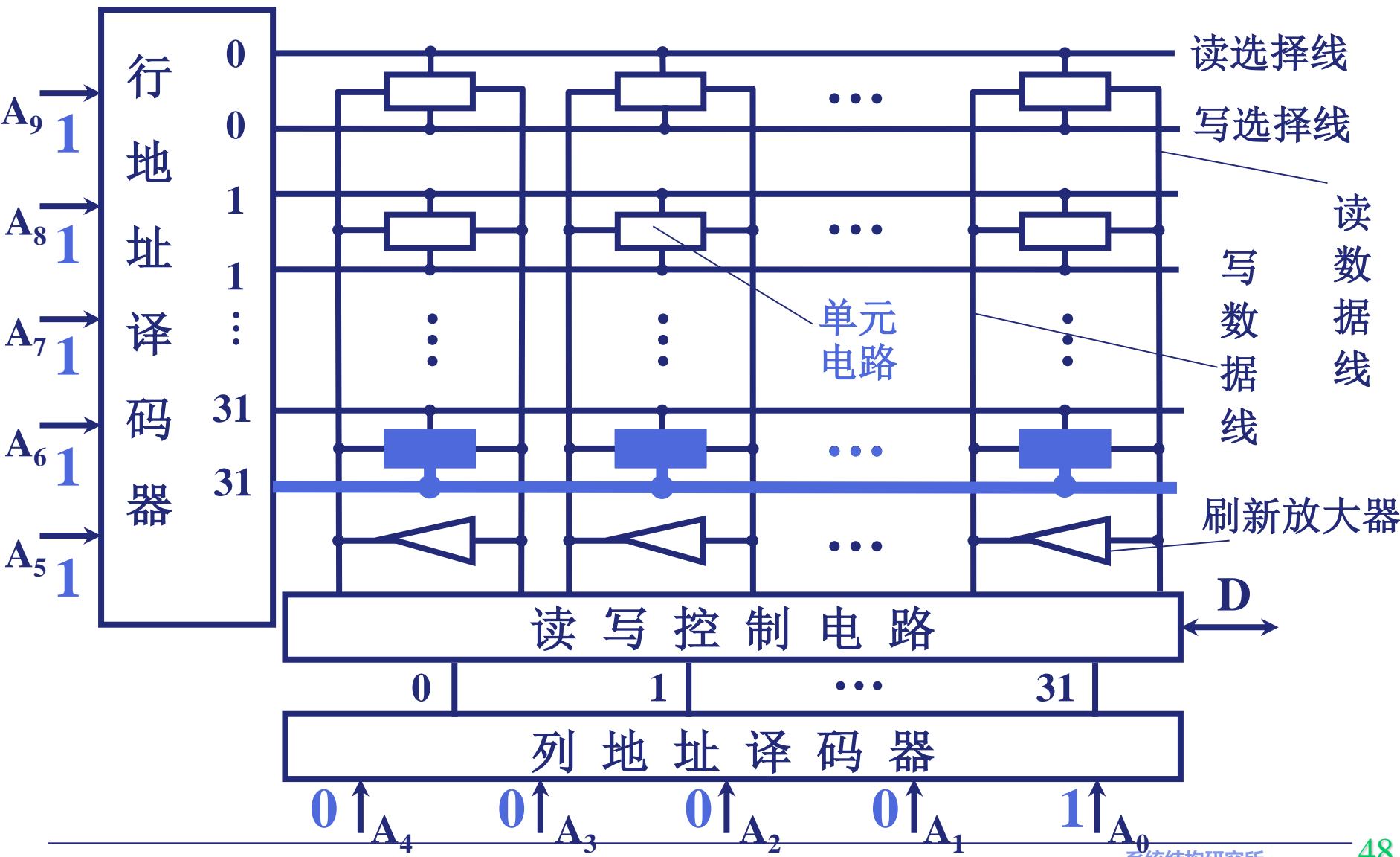


## ② 三管动态 RAM 芯片 (Intel 1103) 写

4.2

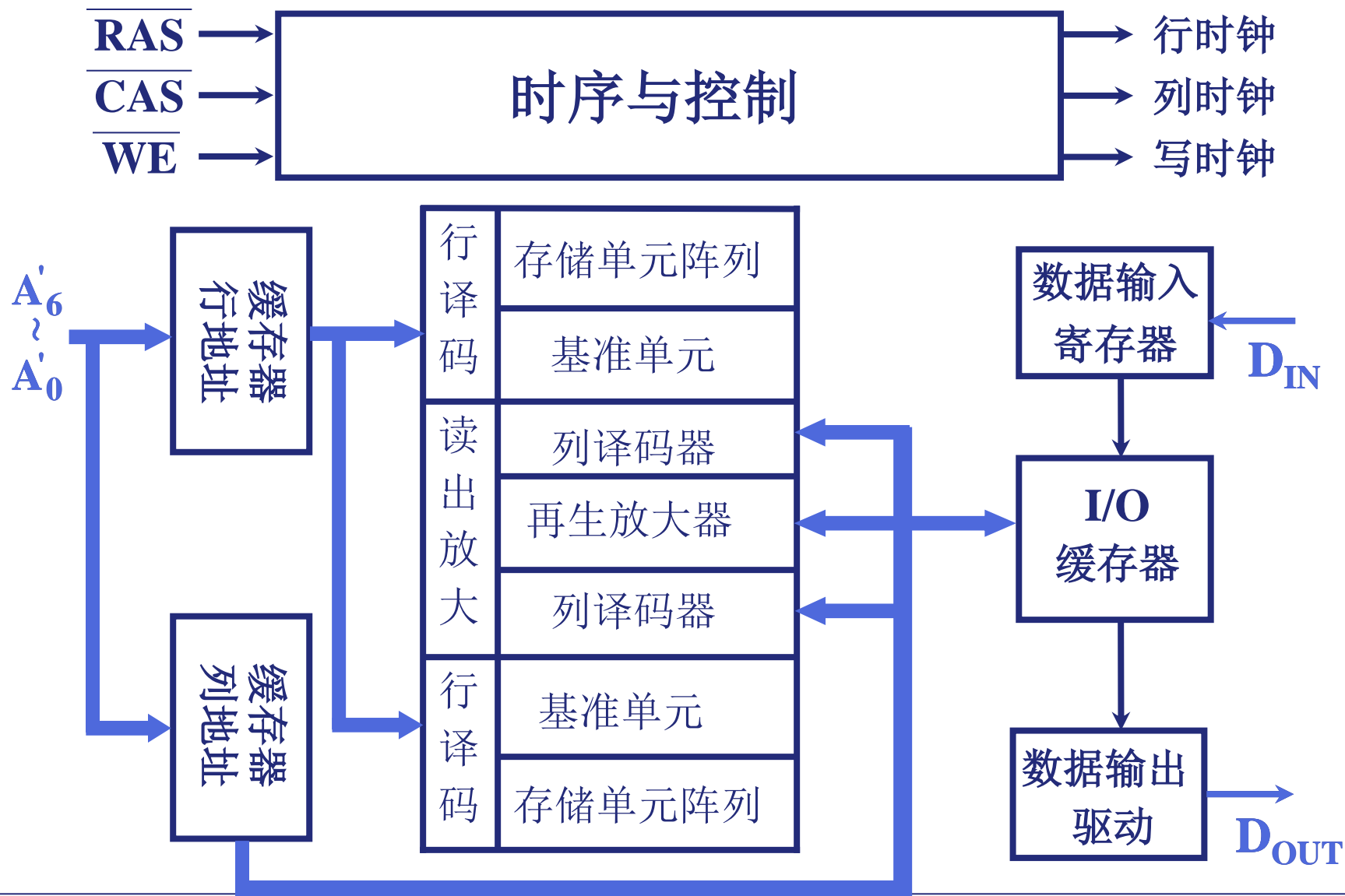


# ② 三管动态 RAM 芯片 (Intel 1103) 写 4.2



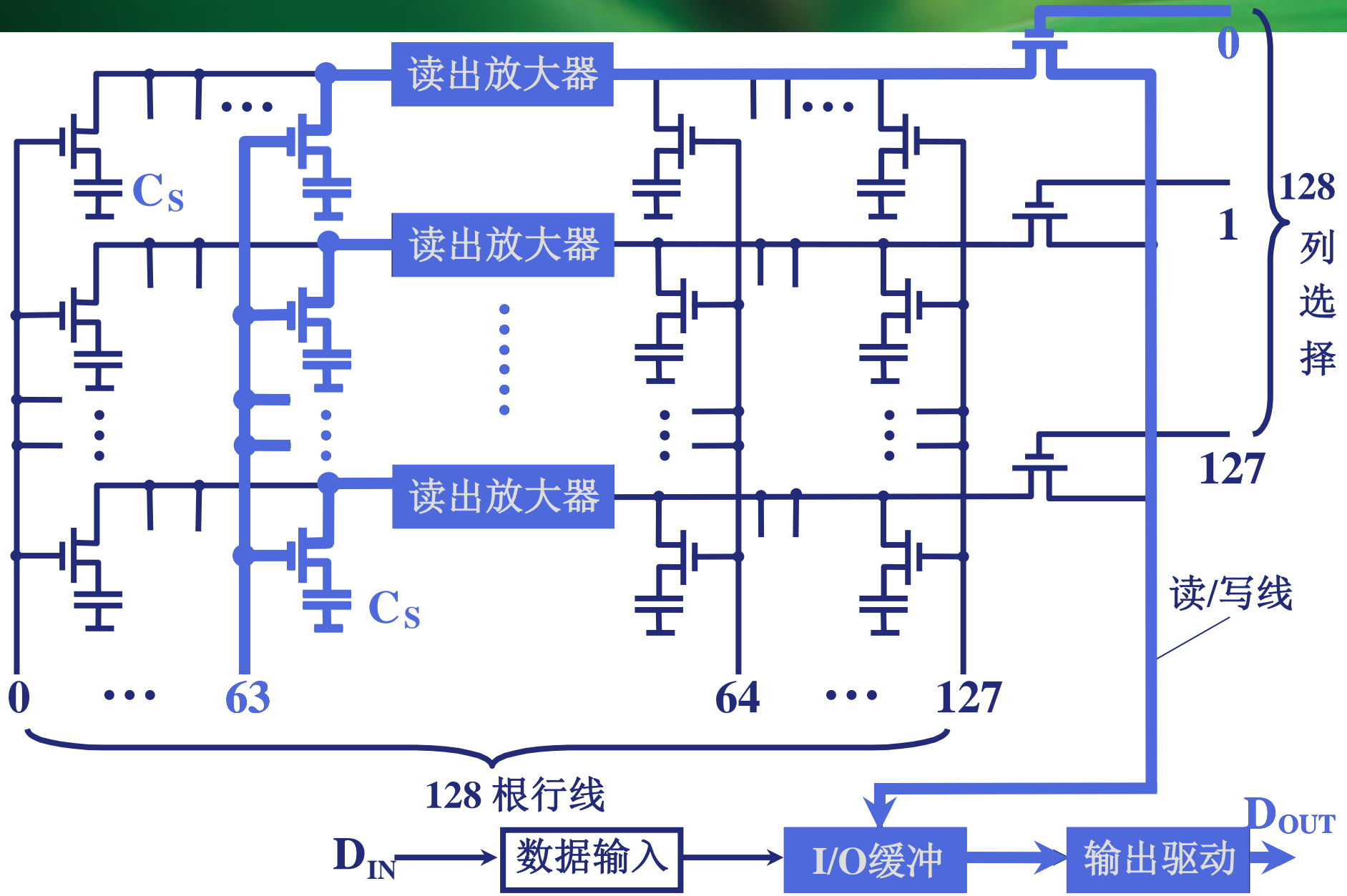


### ③ 单管动态 RAM 4116 (16K × 1位) 外特性



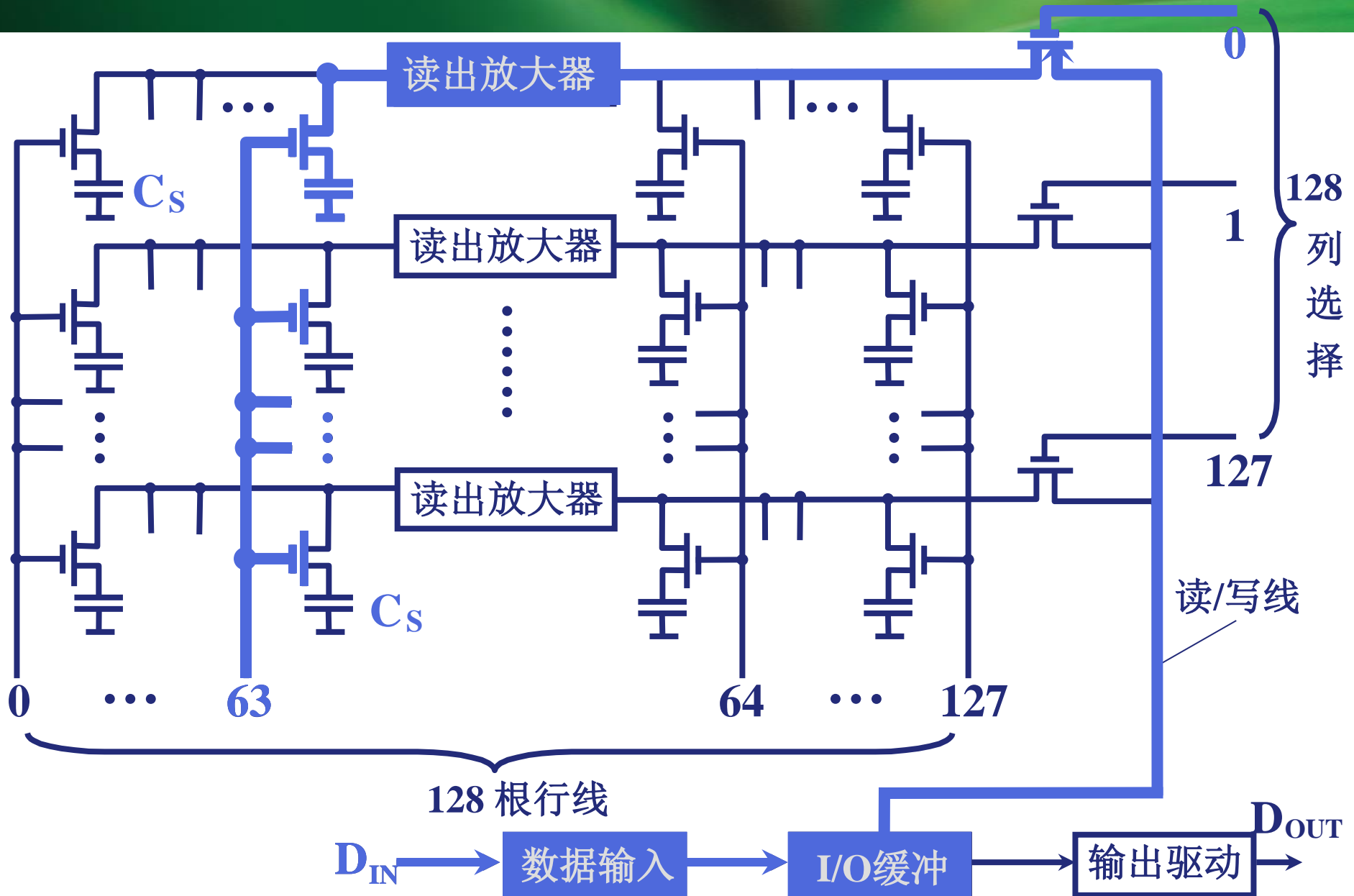


# ④ 4116 (16K × 1位) 芯片 读 原理





# ④ 4116 (16K × 1位) 芯片 写原理



### (3) 动态 RAM 时序

4.2

#### 行、列地址分开传送

##### 读时序

行地址  $\overline{\text{RAS}}$  有效

写允许  $\overline{\text{WE}}$  有效(高)

列地址  $\overline{\text{CAS}}$  有效

数据  $\text{D}_{\text{OUT}}$  有效

##### 写时序

行地址  $\overline{\text{RAS}}$  有效

写允许  $\overline{\text{WE}}$  有效(低)

数据  $\text{D}_{\text{IN}}$  有效

列地址  $\overline{\text{CAS}}$  有效

## (4) 动态 RAM 刷新

- 刷新是先将原存信息读出，再由刷新放大器形成原信息并重新写入的再生过程。
  - 刷新过程对CPU是透明的
  - 刷新是按行进行的，每行中的记忆单元同时被刷新，故刷新操作时仅仅需要行地址，不需要列地址。
  - 刷新类似读操作，但有所不同。刷新操作只给电容补充电荷，不需要信息输出。另外，刷新不用加片选，整个存储器中所有芯片同时被刷新。

## (4) 动态 RAM 刷新

- 刷新周期：从上次对整个存储器刷新结束到下次对整个存储器全部刷新一遍为止的时间间隔，为电容数据有效保存期的上限（64ms）。
- 有三种刷新方式：集中式、分散式、异步刷新。

### ① 集中刷新：

前一段时间正常读/写，后一段时间停止读/写，集中逐行刷新。

特点：集中刷新时间长，不能正常读/写（死区），很少使用。

### ② 分散刷新：

一个存储周期分为两段：前一段用于正常读/写操作，后一段用于刷新操作。

特点：不存在死区，但每个存储周期加长。很少使用。

### ③ 异步刷新：

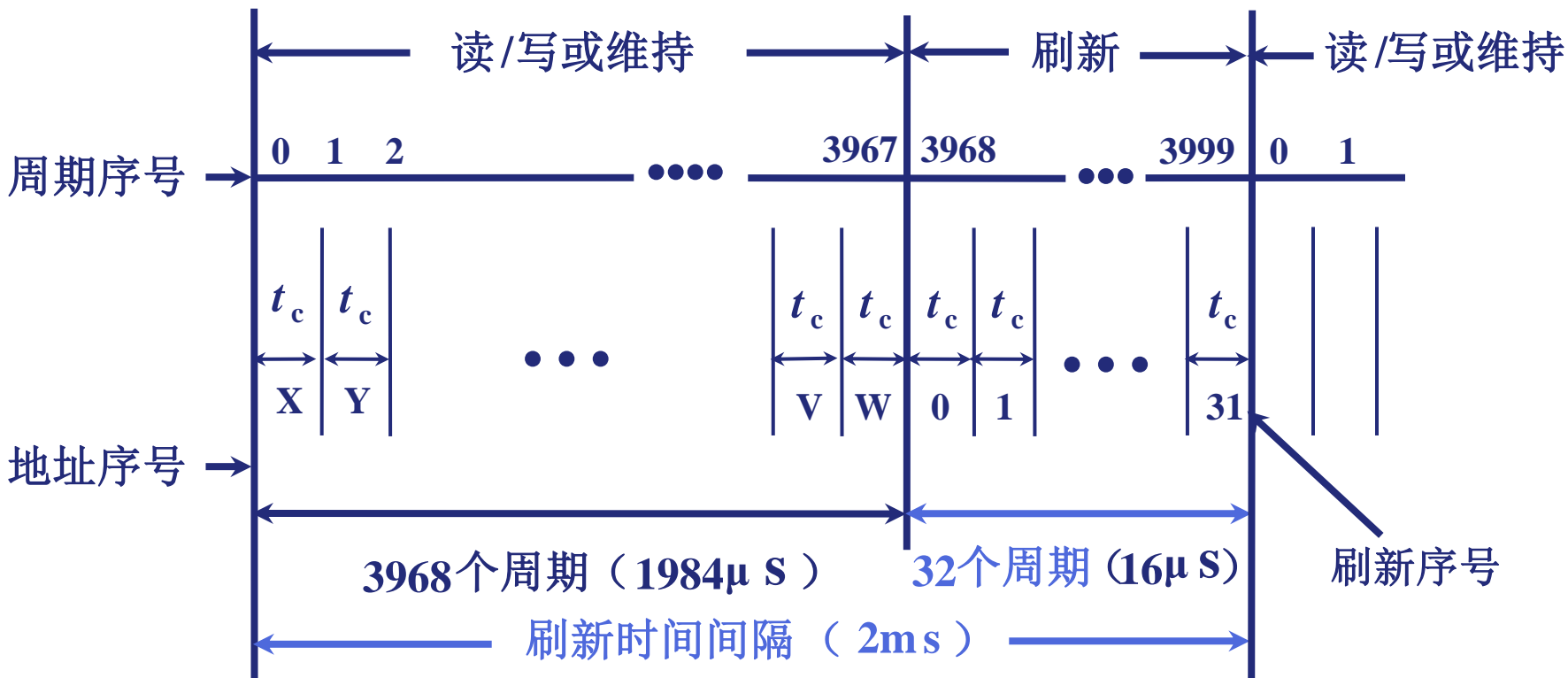
结合上述两种方式。以4096行为例，在64ms时间内必须轮流对每一行刷新一次，即每隔 $64\text{ms}/4096=15.625\mu\text{s}$ 刷新一行。

特点：结合前两种，效率高，用得较多。

## (4) 动态 RAM 刷新

4.2

### ① 集中刷新（存取周期为 $0.5\mu\text{s}$ ）以 $32 \times 32$ 矩阵为例

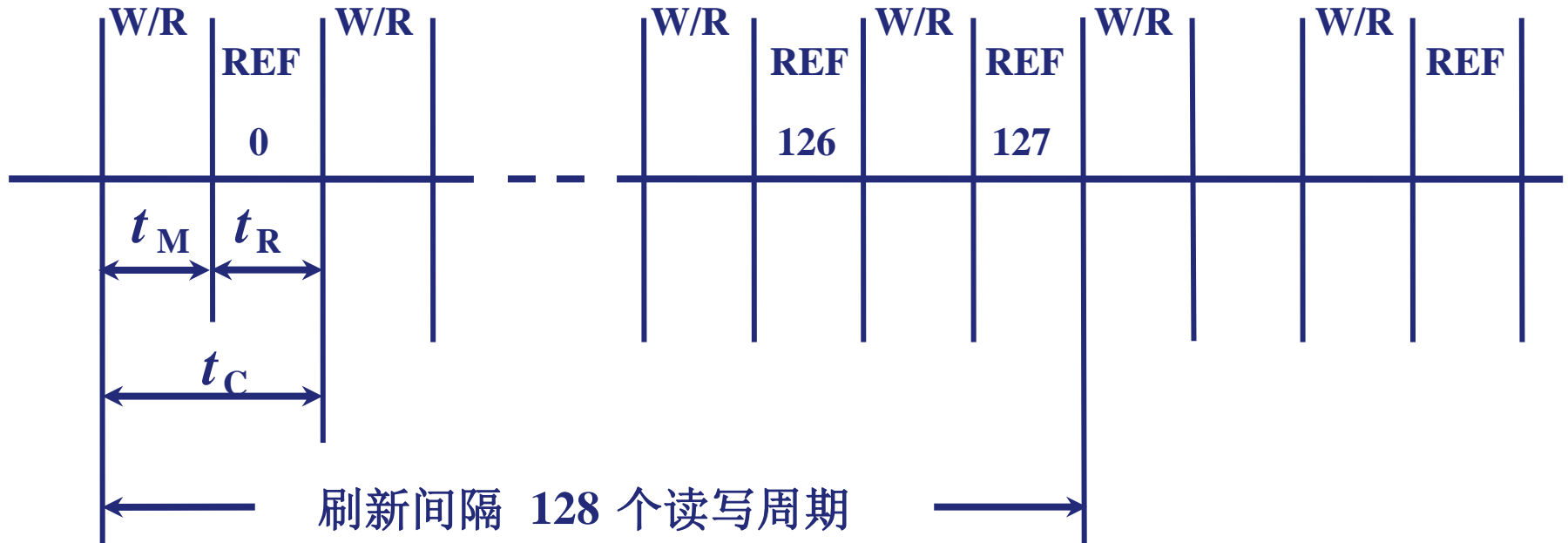


“死区” 为  $0.5\mu\text{s} \times 32 = 16\mu\text{s}$

“死时间率” 为  $32/4000 \times 100\% = 0.8\%$

## ② 分散刷新 (存取周期为 $1\mu\text{s}$ )

## 以 $128 \times 128$ 矩阵为例



$$t_C = t_M + t_R$$

## 无“死区”

## 读写 刷新

(存取周期为  $0.5\ \mu\text{s} + 0.5\ \mu\text{s}$ )



### ③ 异步刷新

4.2

- 是分散刷新与集中刷新的结合

对于  $128 \times 128$  的存储芯片（存取周期为  $0.5\mu\text{s}$ ）

若每隔  $2\text{ ms}$  集中刷新一次 “死区” 为  $64\mu\text{s}$

若每隔  $15.6\mu\text{s}$  刷新一行

而且每行每隔  $2\text{ ms}$  刷新一次 “死区” 为  $0.5\mu\text{s}$

将刷新安排在指令译码阶段，不会出现 “死区”

### 3. 动态 RAM 和静态 RAM 的比较

4.2

	主存 DRAM	SRAM 缓存
存储原理	电容	触发器
集成度	高	低
芯片引脚	少	多
功耗	小	大
价格	低	高
速度	慢	快
刷新	有	无

### 1. 掩膜 ROM (MROM)

- 以元件的“有/无”表示“1”、“0”。
- 内容由半导体制造厂按用户提出的要求在芯片的生产过程中直接写入的，写入后内容无法改变。

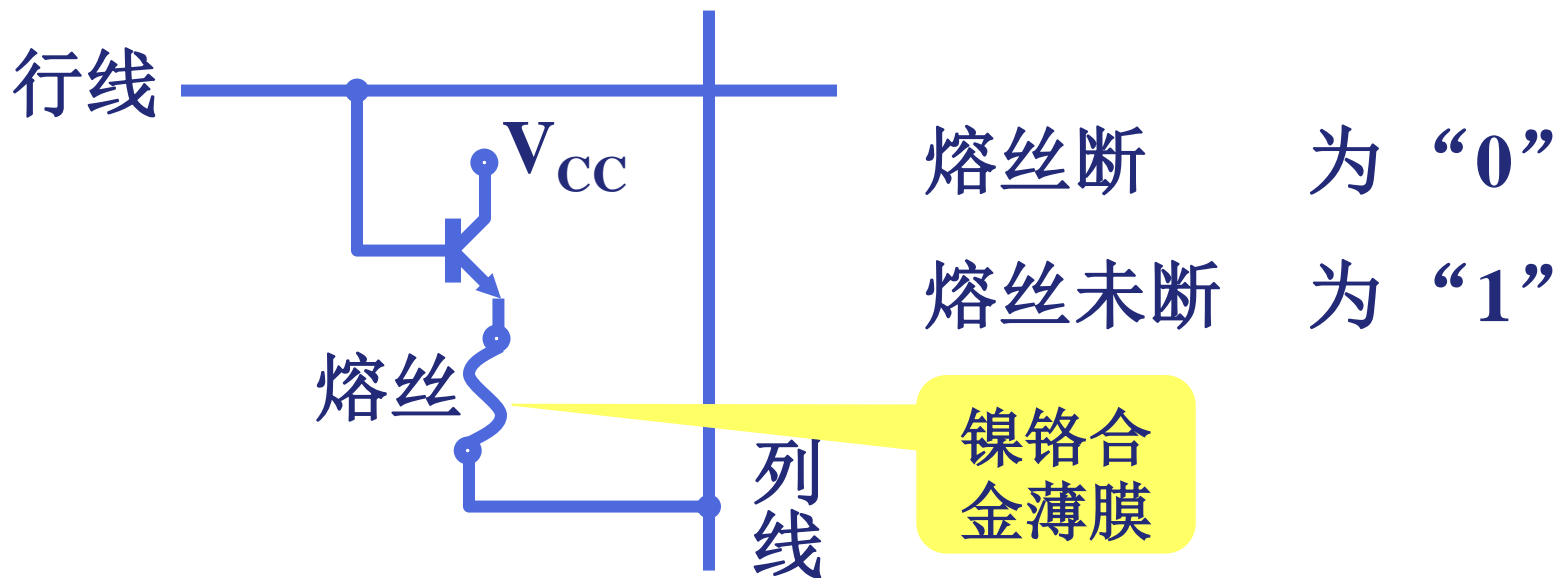
行列选择线交叉处有 MOS 管为“1”

行列选择线交叉处无 MOS 管为“0”

- 优点：可靠性高，形成批量之后价格便宜；
- 缺点：用户对制造厂的依赖性过大，灵活性差。

### 2. PROM (一次性编程)

- 允许用户利用专门的设备（编程器）写入自己的程序，但一旦写入后，其内容将无法改变。



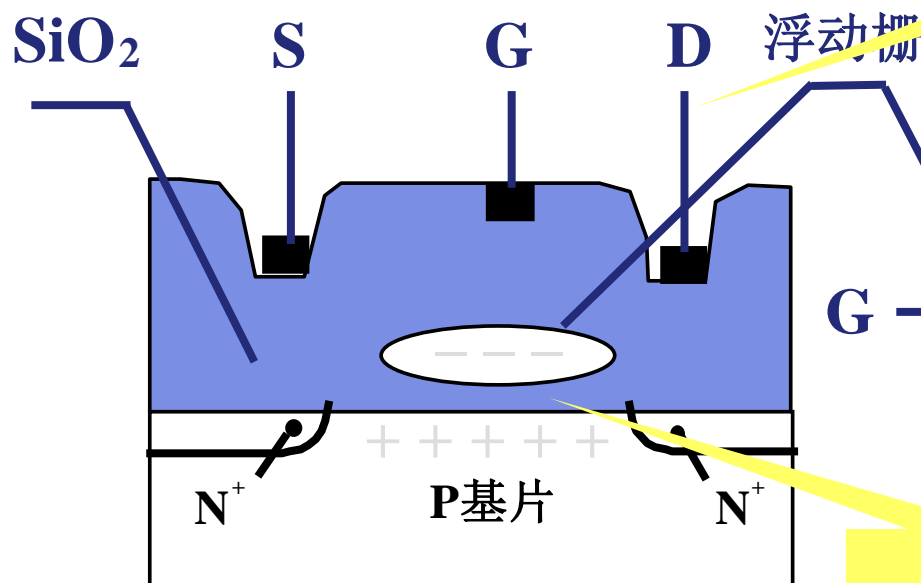
- 产品出厂时，所有记忆单元均制成“0”（或制成“1”），用户根据需要可自行将其中某些记忆单元改为“1”（或改为“0”）。
- 双极型PROM有两种结构，一种是熔丝烧断型，另一种是PN结击穿型，

### 3. EPROM (多次性编程)

4.2

#### 浮动栅雪崩注入型MOS管 (1) N型沟道浮动栅 MOS 电路

25V (写入电压)、  
50ms宽正脉冲



埋在氧化物绝缘层中，不与外部导通。  
写入信息前，浮栅上没有电荷，D、S  
间不导通，MOS呈“1”态

S 源

D 漏

紫外线全部擦洗

$0.05\sim 0.1\mu\text{m}$

D 端加正电压

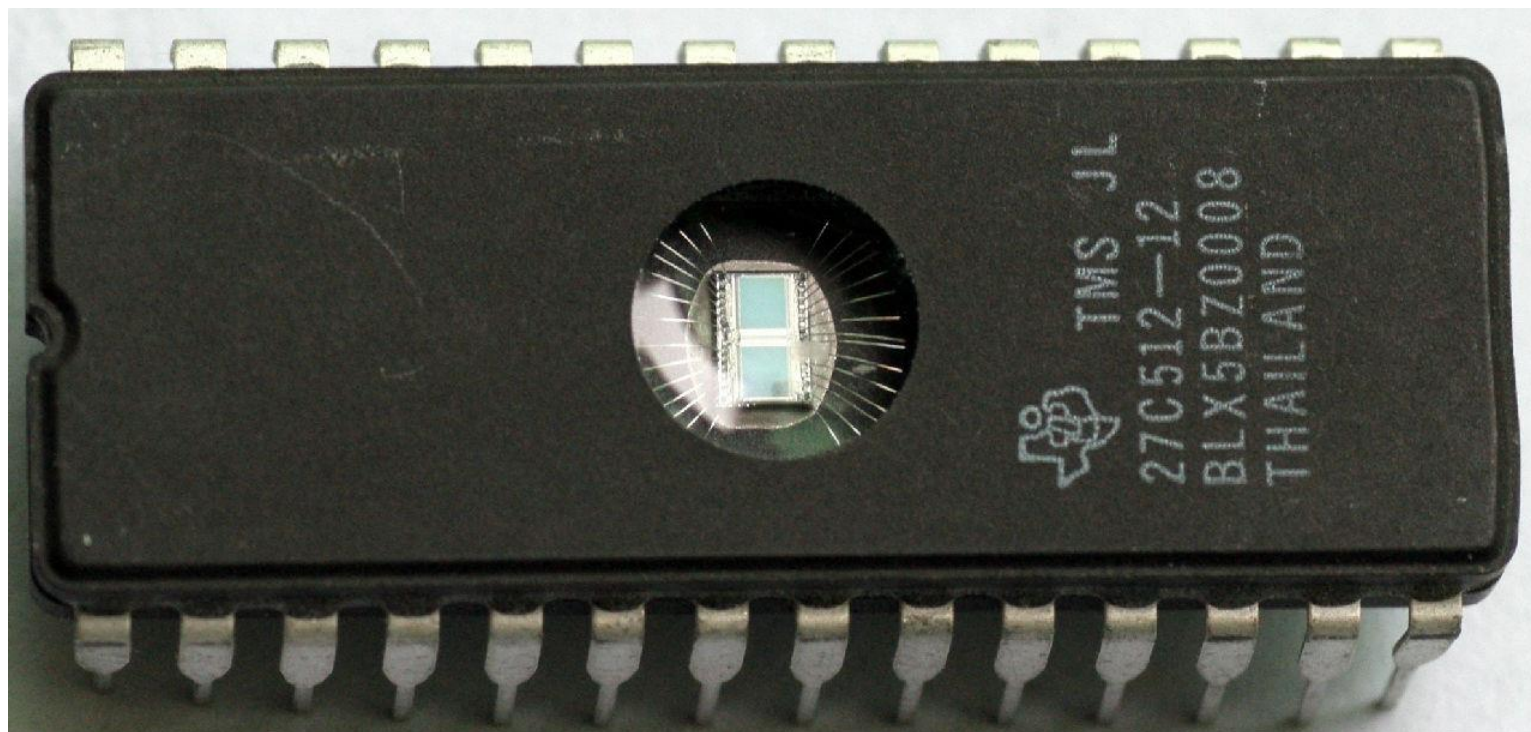
形成浮动栅

S 与 D 不导通为“0”

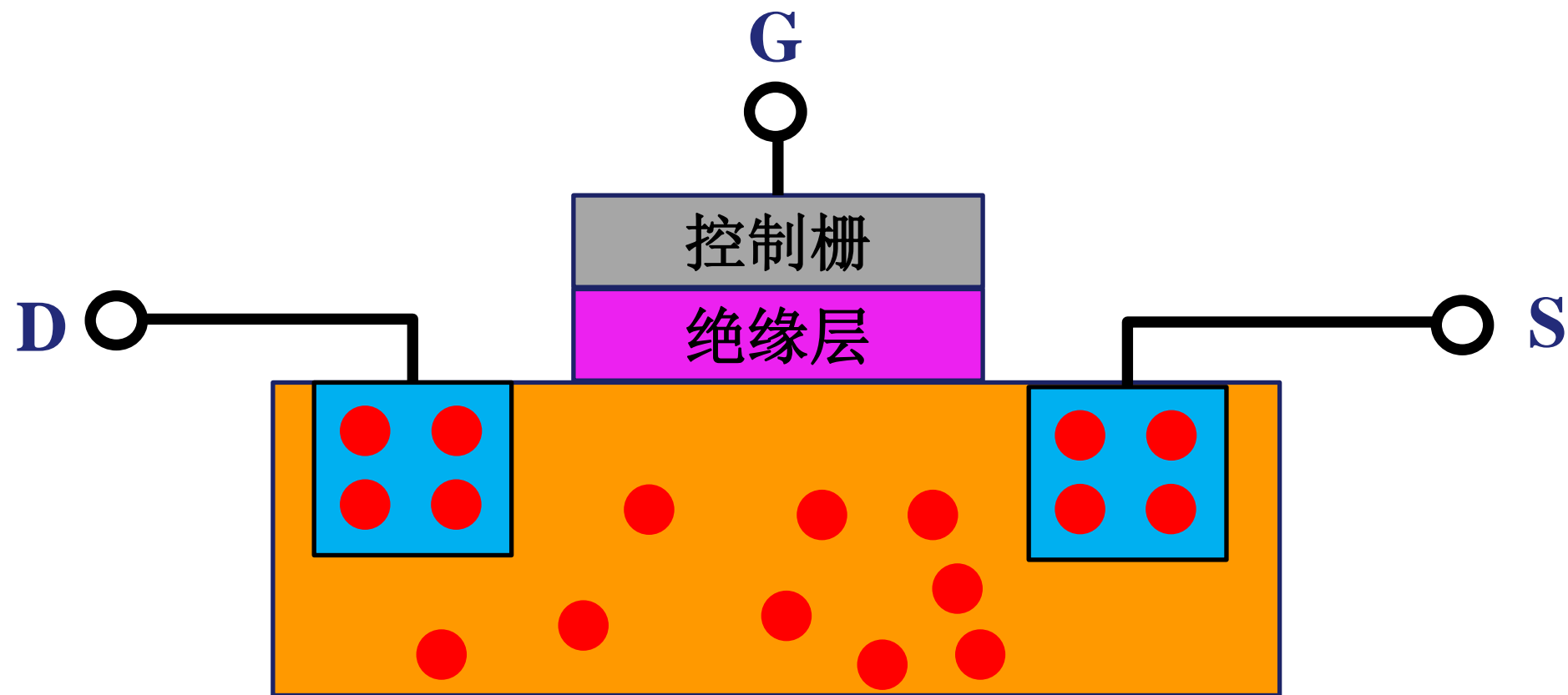
D 端不加正电压

不形成浮动栅

S 与 D 导通为“1”

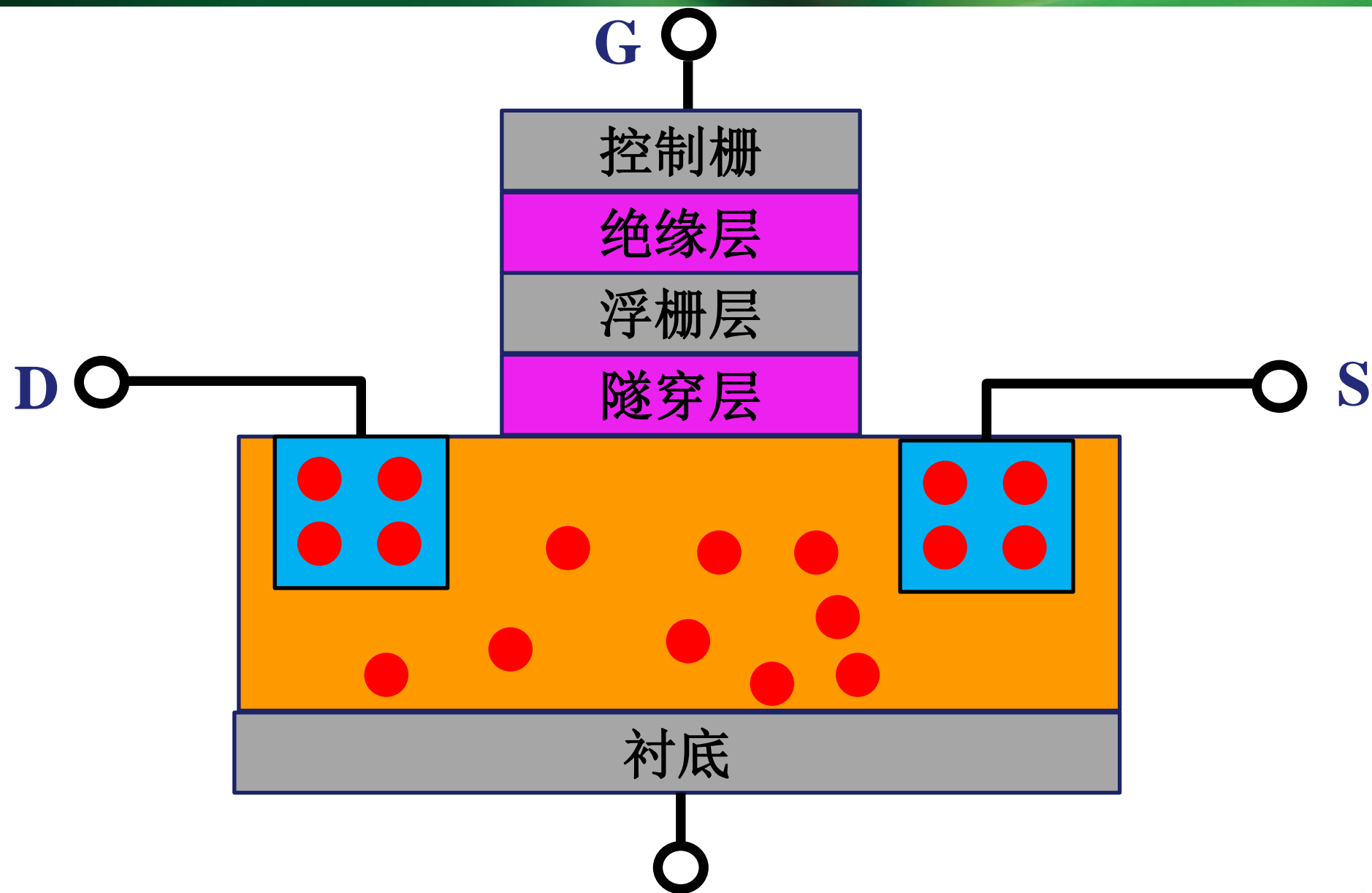


# 普通晶体管





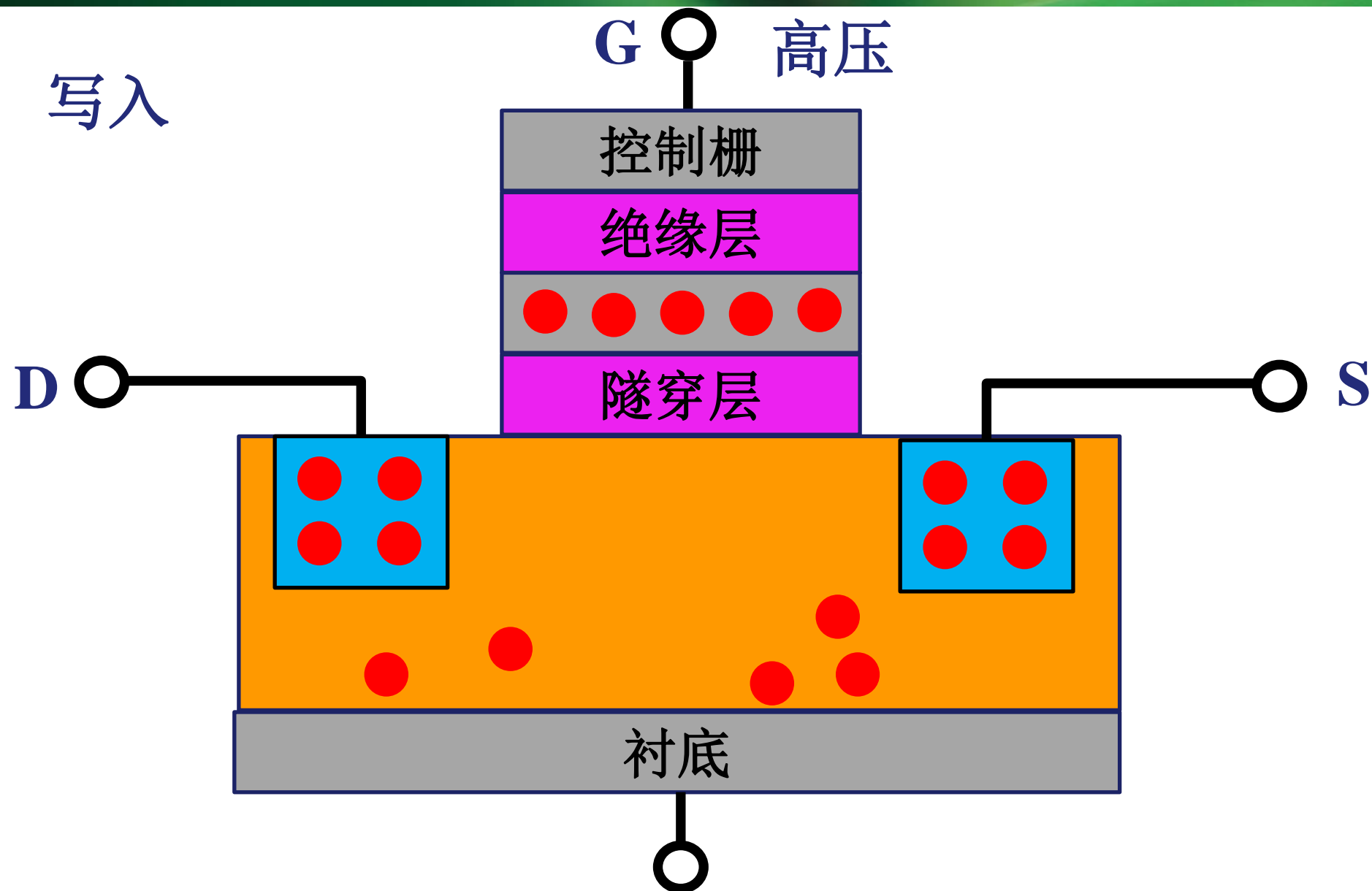
# 浮栅晶体管



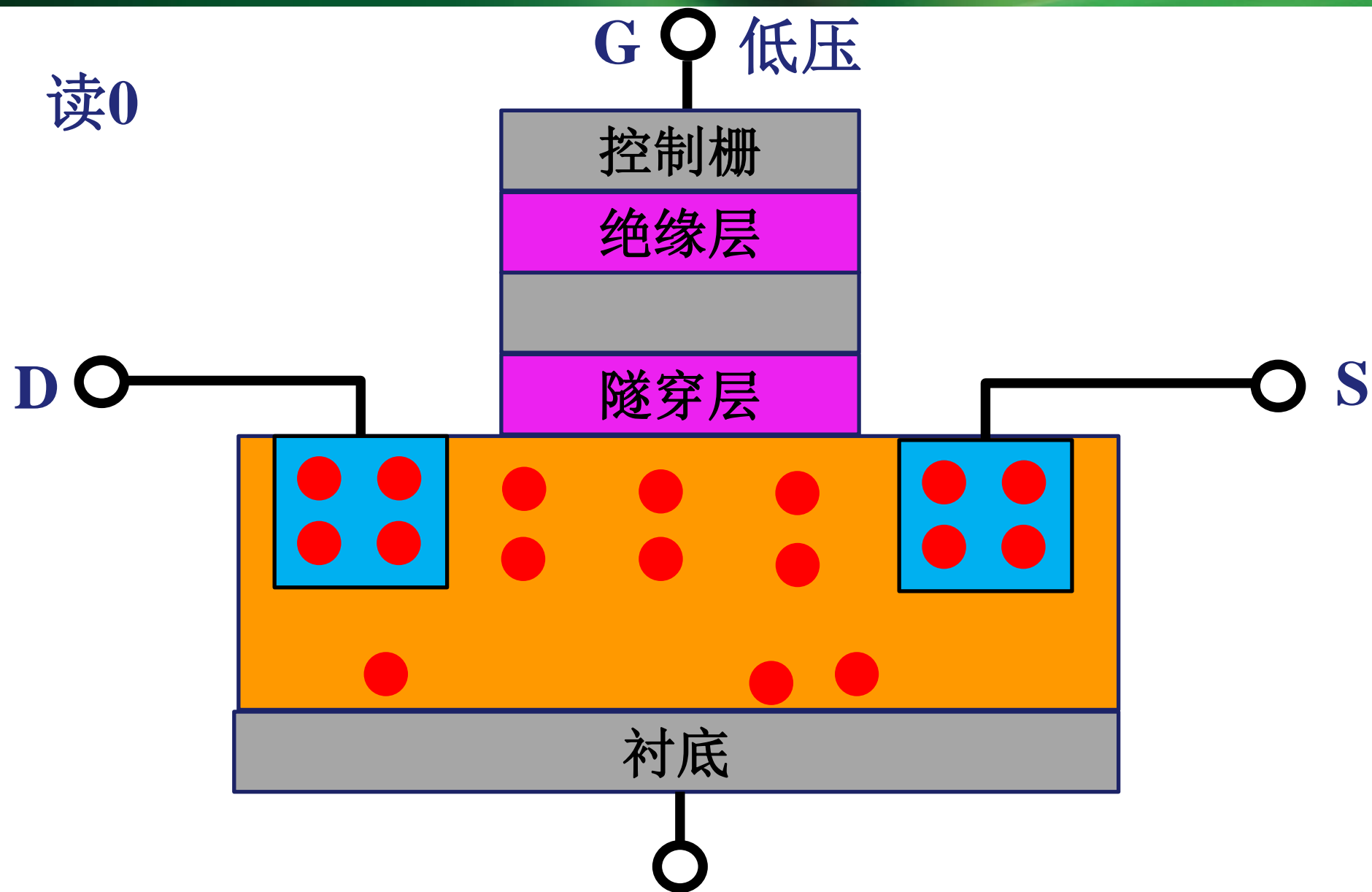


# 浮栅晶体管

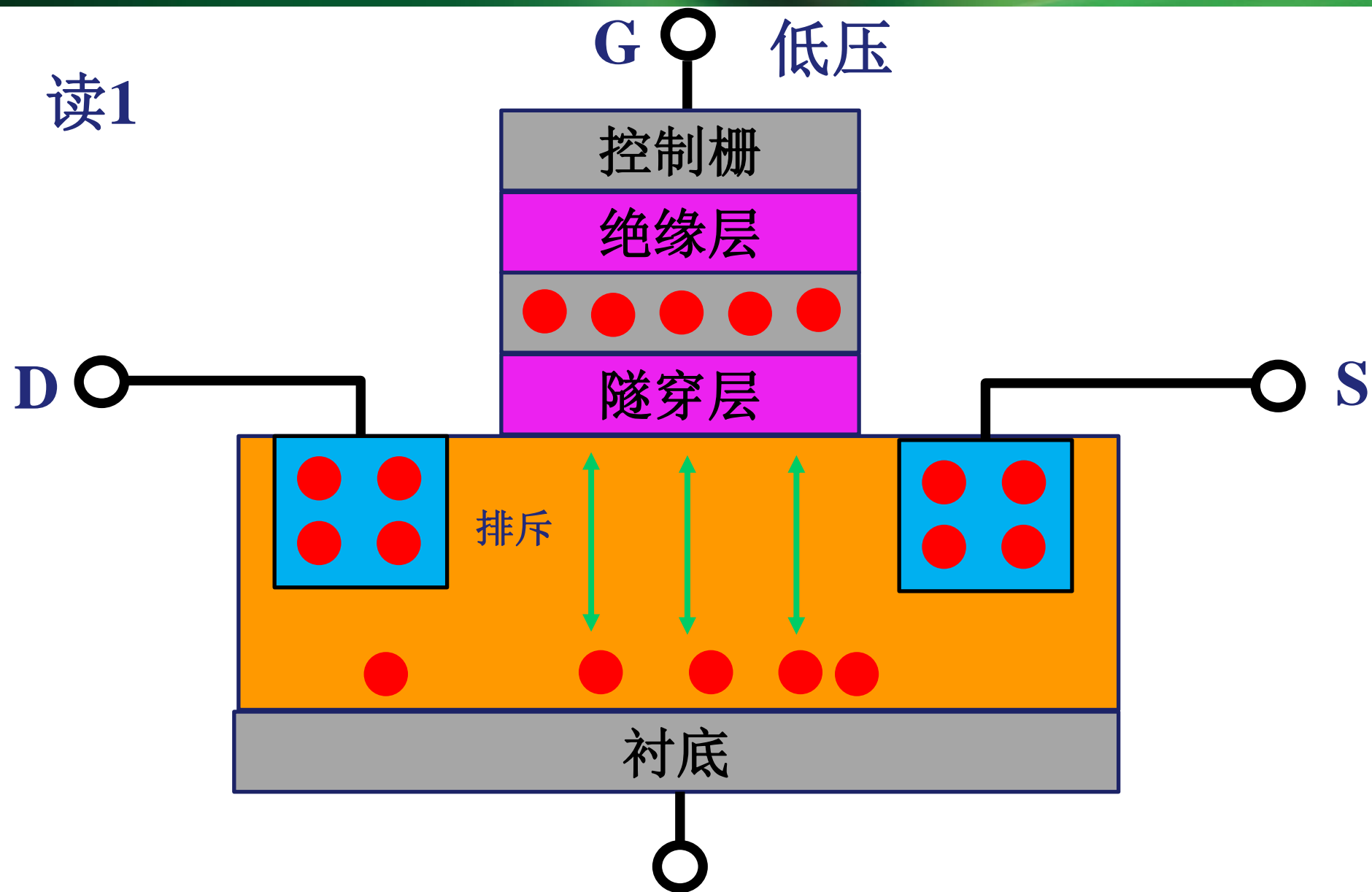
写入



# 浮栅晶体管

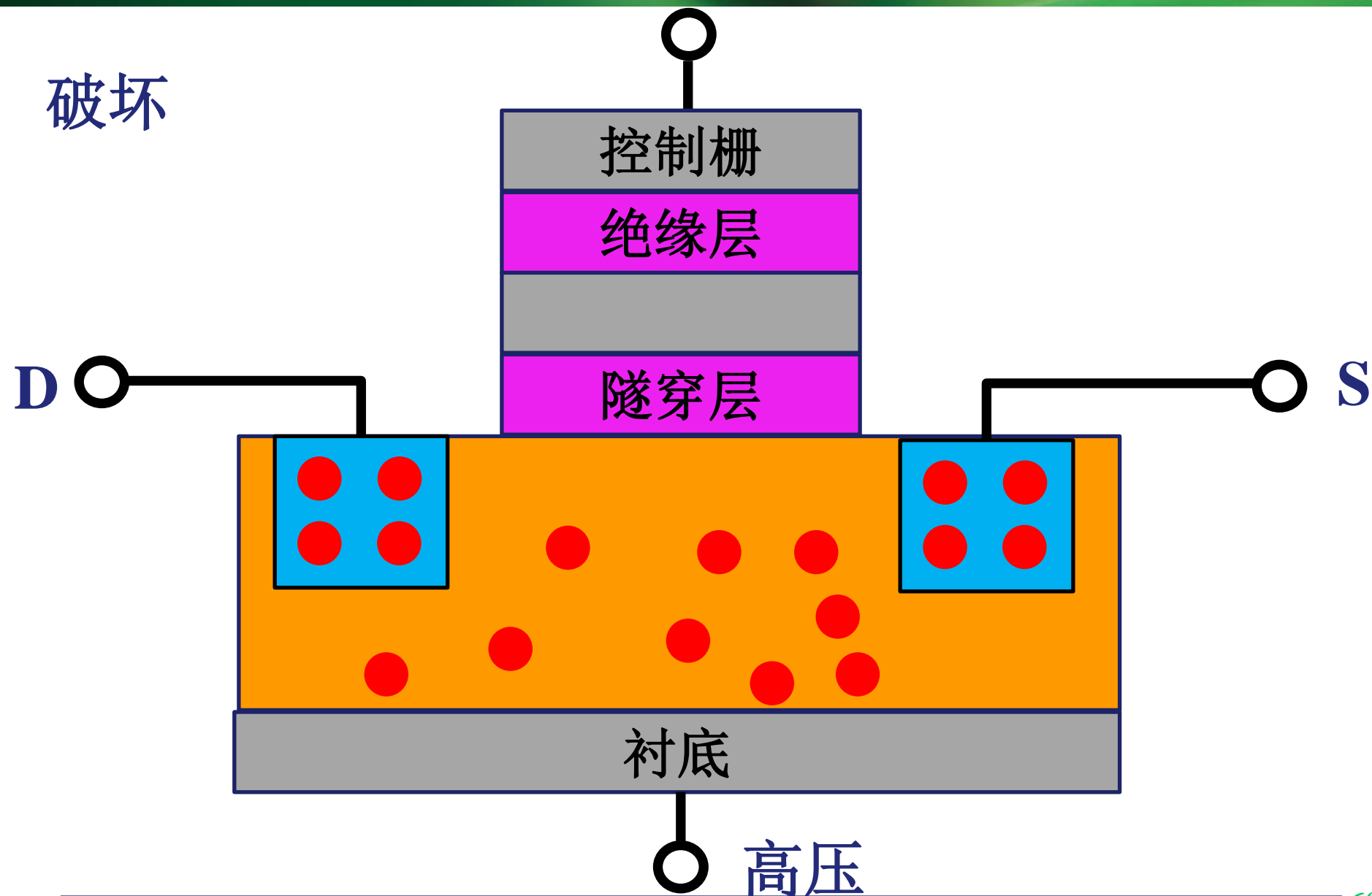


# 浮栅晶体管



# 浮栅晶体管

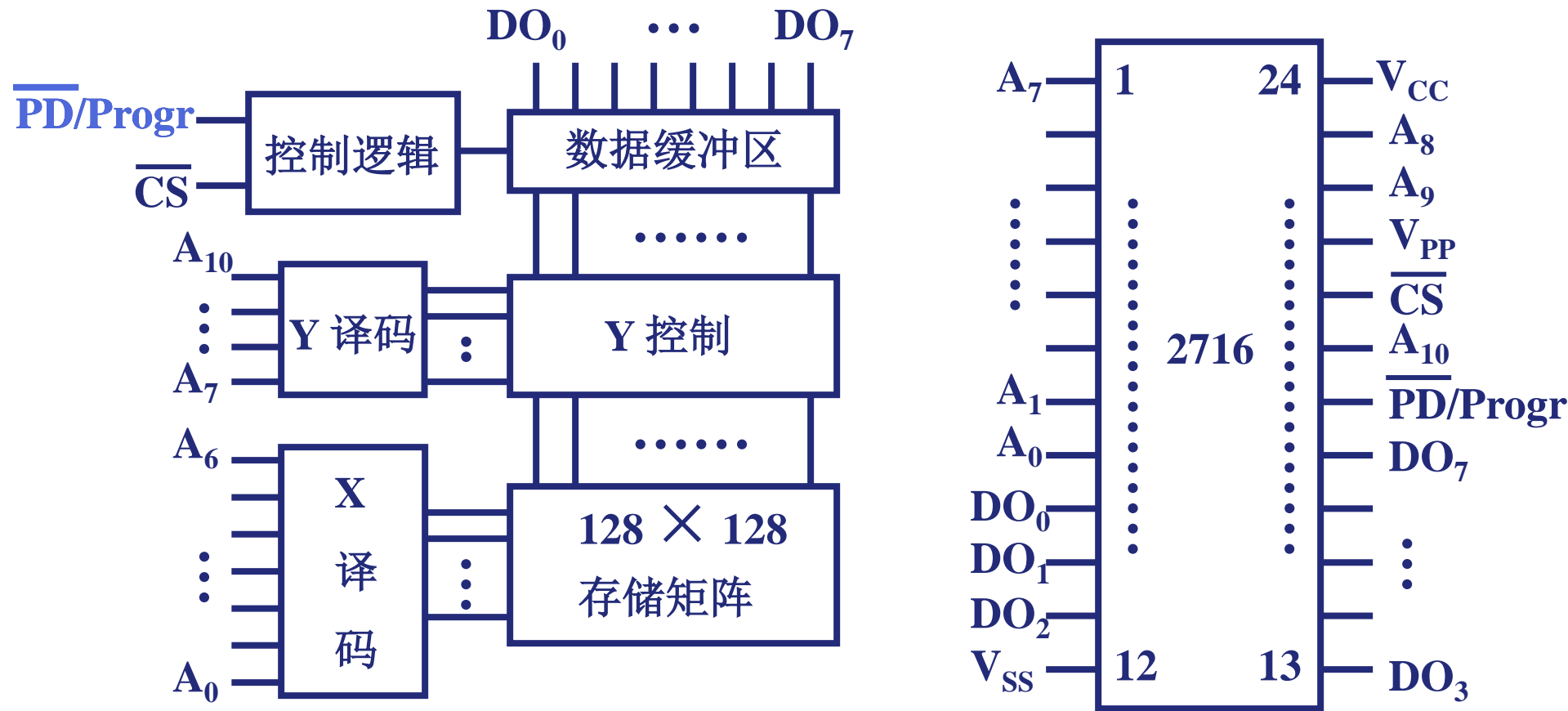
破坏



### 3. EPROM (多次性编程)

- 可由用户利用编程器写入信息，可以多次改写。
- EPROM出厂时，存储内容为全“1”，用户可以根据需要将其中某些记忆单元改为“0”。当需要更新存储内容时可以将原存储内容擦除（恢复全“1”），以便再写入新的内容。
- EPROM（UVEPROM）：用紫外线灯制作的擦除器照射存储器芯片上的透明窗口，使芯片中原存内容被擦除。
  - 由于是用紫外线灯进行擦除，所以只能对整个芯片擦除，而不能对芯片中个别需要改写的存储单元单独擦除。
  - 为了防止存储的信息受日光中紫外线成分的作用而缓慢丢失，在写入完成后，必须用不透明的黑纸将芯片上的透明窗口封住。
- E<sup>2</sup>PROM是采用电气方法来进行擦除的。
  - 在联机条件下既可以用字擦除方式擦除，也可以用数据块擦除方式擦除。以字擦除方式操作时，能够只擦除被选中的那个存储单元的内容；在数据块擦除操作时，可擦除数据块所有单元的内容。

## (2) 2716 EPROM 的逻辑图和引脚



$\overline{\text{PD/Progr}}$  功率下降 / 编程输入端    读出时为低电平

# EPROM不能取代RAM

- 原因

① EPROM的编程次数（寿命）是有限的：

EEPROM因为氧化层被磨损，重复改写次数一般10万次

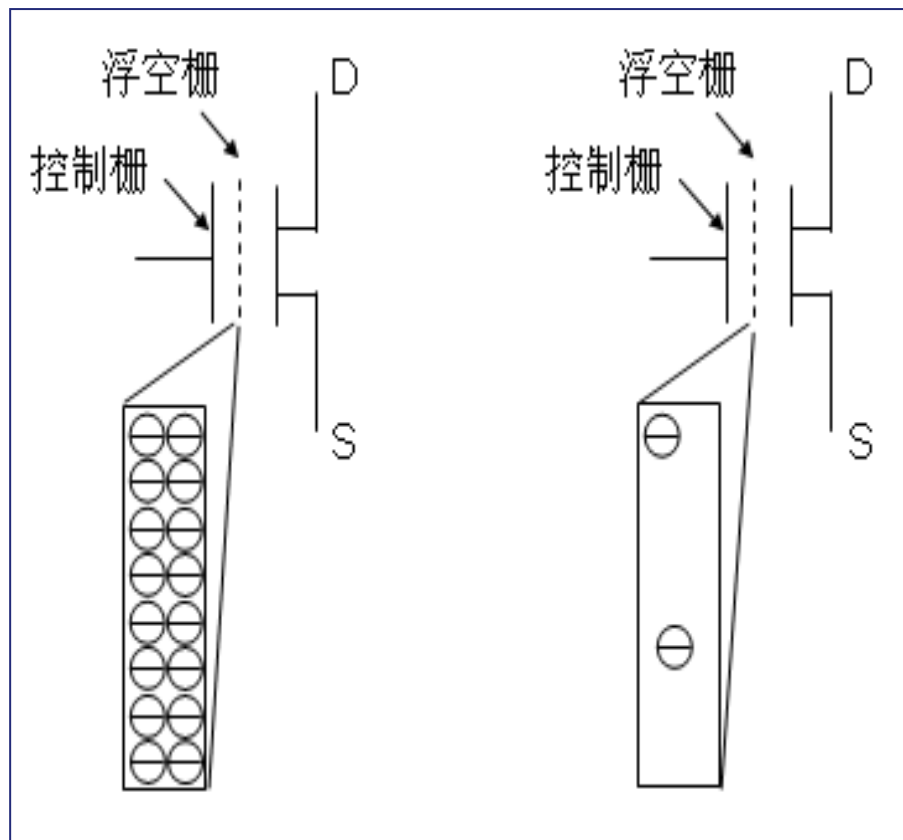
② 写入时间过长，即使对于EEPROM，擦除一个字节大约需要10ms，写入一个字节大约需要10us，比SRAM或DRAM的时间长100—1000倍。

## 4. Flash Memory (快擦型存储器)

- 闪速存储器是20世纪80年代中期出现的一种快擦写型存储器，它的主要特点是既可在不加电的情况下长期保存信息，又能在线进行快速擦除与重写，兼备了EEPROM和RAM的优点。
- 一般，微型计算机的主板采用闪速存储器来存储BIOS程序，由于BIOS的数据和程序非常重要，不允许修改，故早期主板BIOS芯片多采用PROM或EPROM。



# 闪存 (Flash Memory)

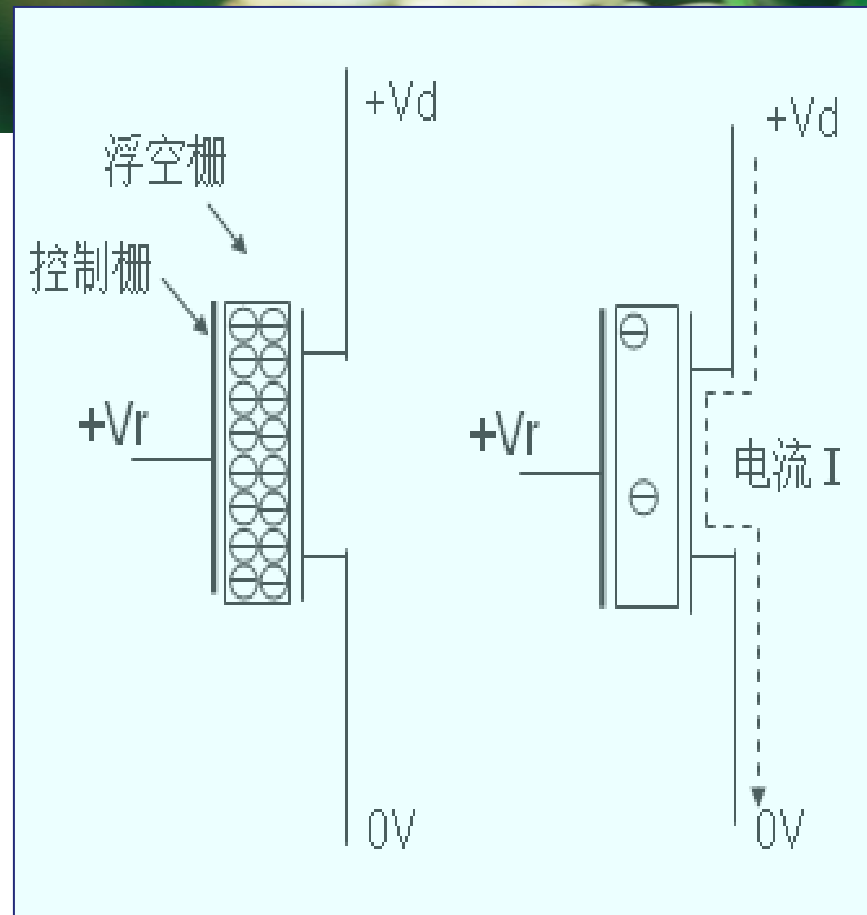
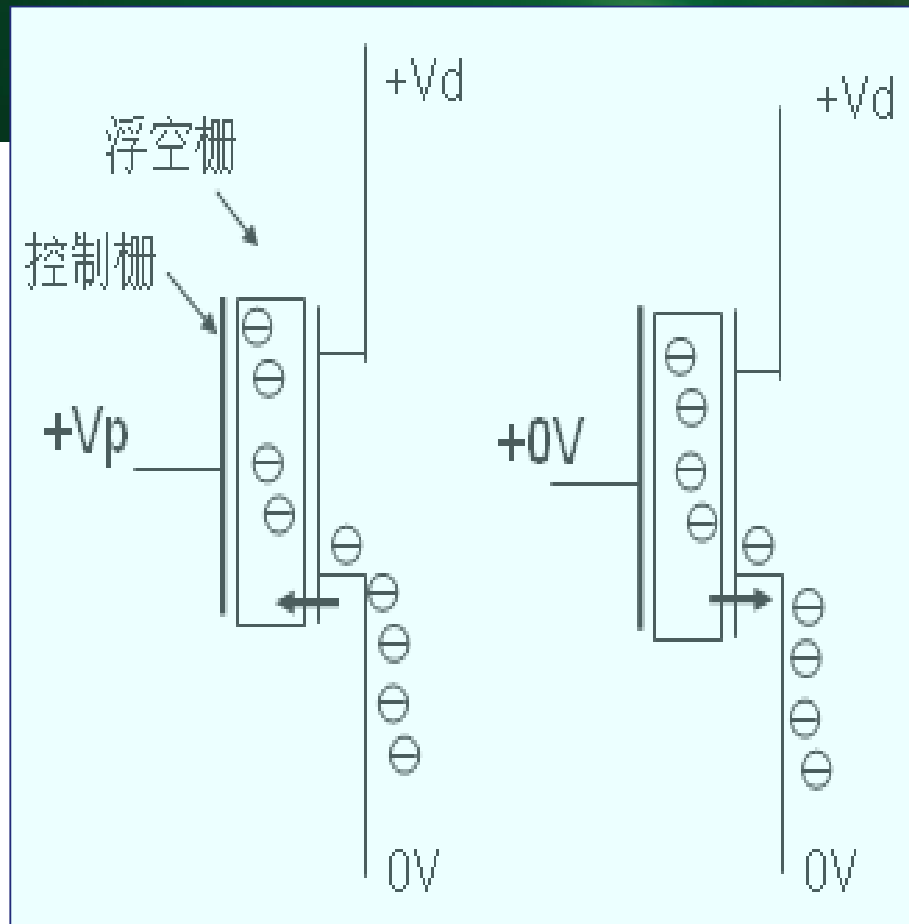


(a) “0”状态

(b) “1”状态

控制栅加足够正电压时，浮空栅储存大量负电荷，为“0”态；控制栅不加正电压时，浮空栅少带或不带负电荷，为“1”态

闪存的读取速度与DRAM相近，是磁盘的100倍左右；写数据（快擦—编程）则与硬盘相近



(a) 编程:写 “0”      (b) 擦除:写 “1”

(a) 读 “0”      (b) 读 “1”

有三种操作：擦除、编程、读取

读快、写慢！

{ 写入：快擦（所有单元为1）－ 编程（需要之处写0）  
 读出：控制栅加正电压，若状态为0，则读出电路检测不到电流；若状态为1，则能检测到电流。

# 施敏 (Simon M • Sze)

美国籍，微电子科学技术、半导体器件物理专业，台湾交通大学电子工程学系毫微米元件实验室教授，美国工程院院士。

1936年出生。1957年毕业于台湾大学。1960年、1963年分别获得华盛顿大学和斯坦福大学硕士与博士学位。

施敏博士是国际知名的微电子科学技术与半导体器件专家和教育家。他是非挥发MOS场效应记忆晶体管（MOSFET）的发明者，这项发明已成为世界集成电路产业主导产品之一(Flash)。90年代初其产值已达100亿美元。此外，他还有多项创造性成果，如80年代初首先以电子束制造出线宽为 $0.15\mu\text{m}$  MOSFET器件，首先发现崩溃电压与能隙的关系，建立了微电子元件最高电场的指标，如此等等，他也是手机发明人之一。



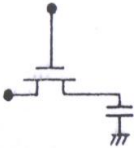
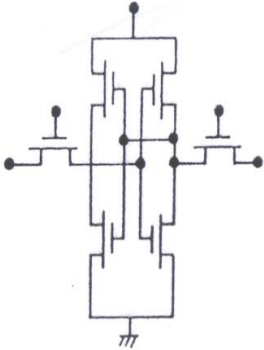
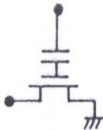
# 续

由于他在微电子器件及在人才培养方面的贡献，先后被选为台湾中央研究院院士和美国国家工程院院士；1991年他得到IEEE电子器件的最高荣誉奖（Ebers奖），称他在电子元件领域做出了基础性及前瞻性贡献。获得过三次诺贝尔奖提名。施敏教授著作无数，其中之一《Physics of Semiconductor Devices》被翻译成六国文字，发行量逾百万册并广泛用作教科书与参考书。

# SEMICONDUCTOR MEMORIES

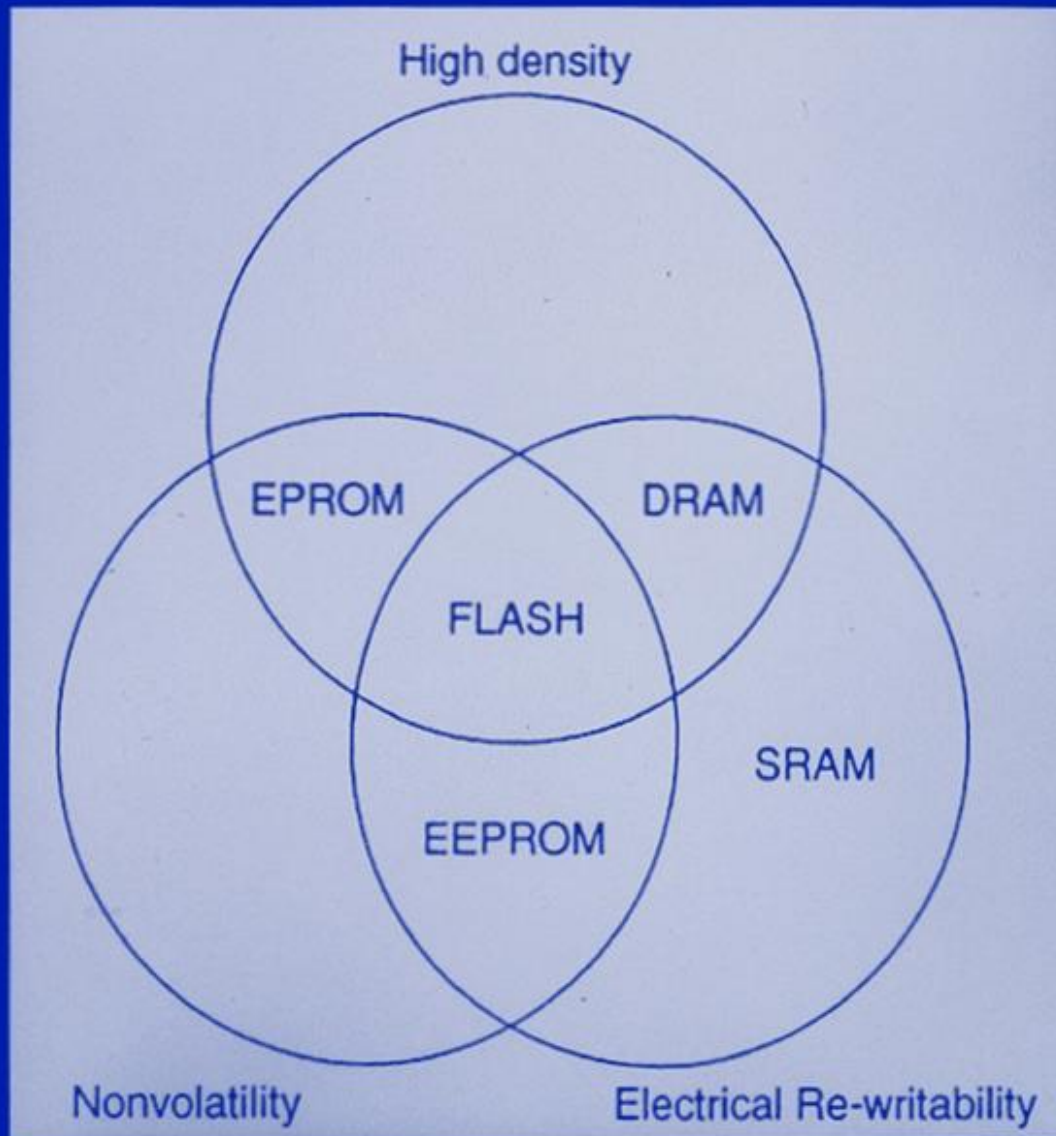
- **VOLATILE MEMORIES**— Lose stored information once supply is switched off
  - DRAM ( Dynamic Random Access Memory)
  - SRAM ( Static Random Access Memory)
- **NONVOLATILE MEMORIES**— Keep stored information when the power supply is switched off
  - Flash MEMORY
  - EEPROM (Electrically Erasable-Programmable Read Only Memory)
  - EPROM ( Erasable-Programmable Read Only Memory)

# COMPARISON OF SEMICONDUCTOR MEMORIES

DRAM	SRAM	Flash
		
1T + 1C	6T / 4T+2L	1T
Volatile	Volatile	Nonvolatile

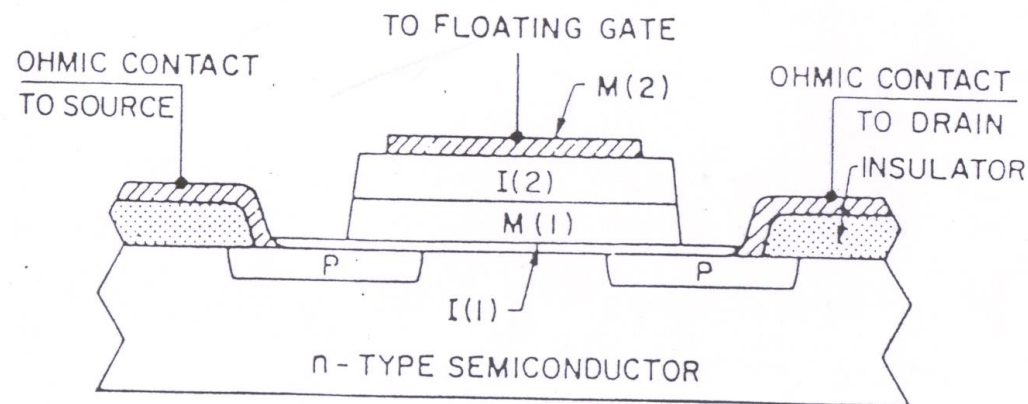


# COMPARISON OF MEMORY ATTRIBUTES



# THE FIRST NONVOLATILE SEMICONDUCTOR MEMORY

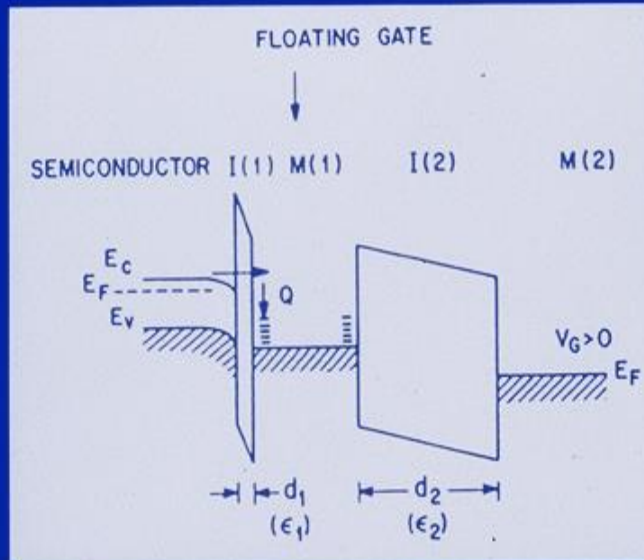
## —The Floating-Gate Concept (1967)



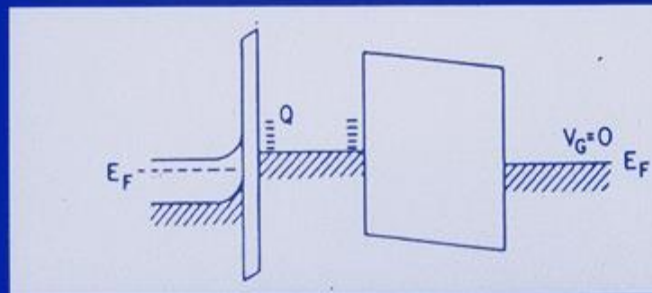
D. Kahng, S. M. Sze "A Floating Gate and its Application to Memory Device" Bell Syst. Tech. J., 46, 1288 (1967)



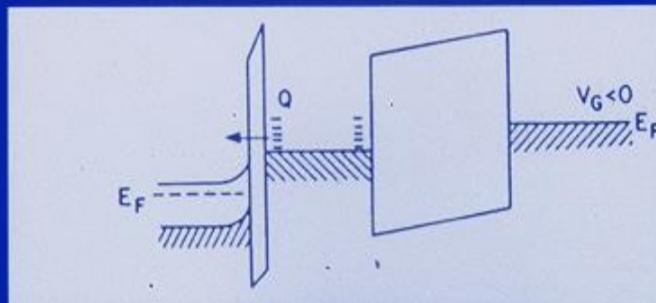
# FLOATING-GATE MEMORY DEVICE OPERATION



(a) Programming Mode  
(Fowler-Nordheim or  
Direct Tunneling)



(b) Storage Mode

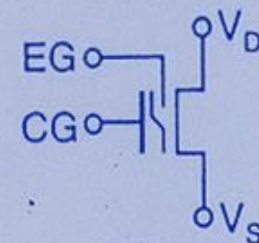
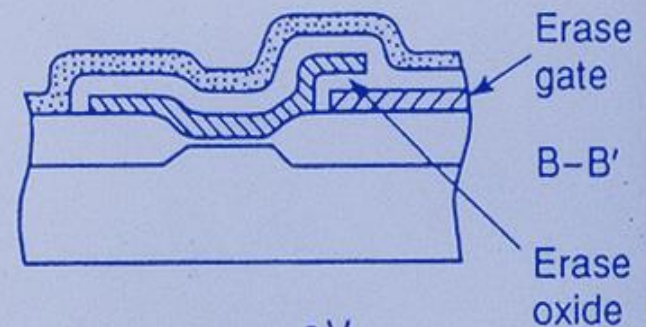
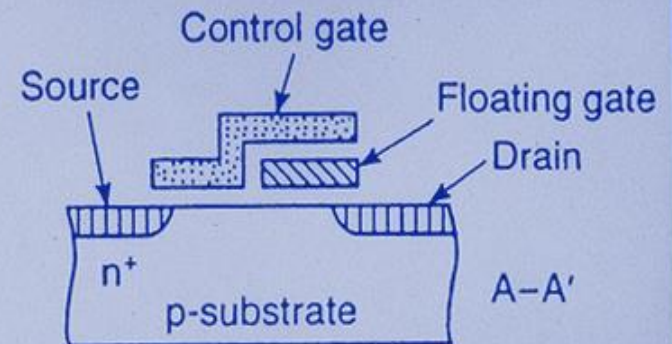
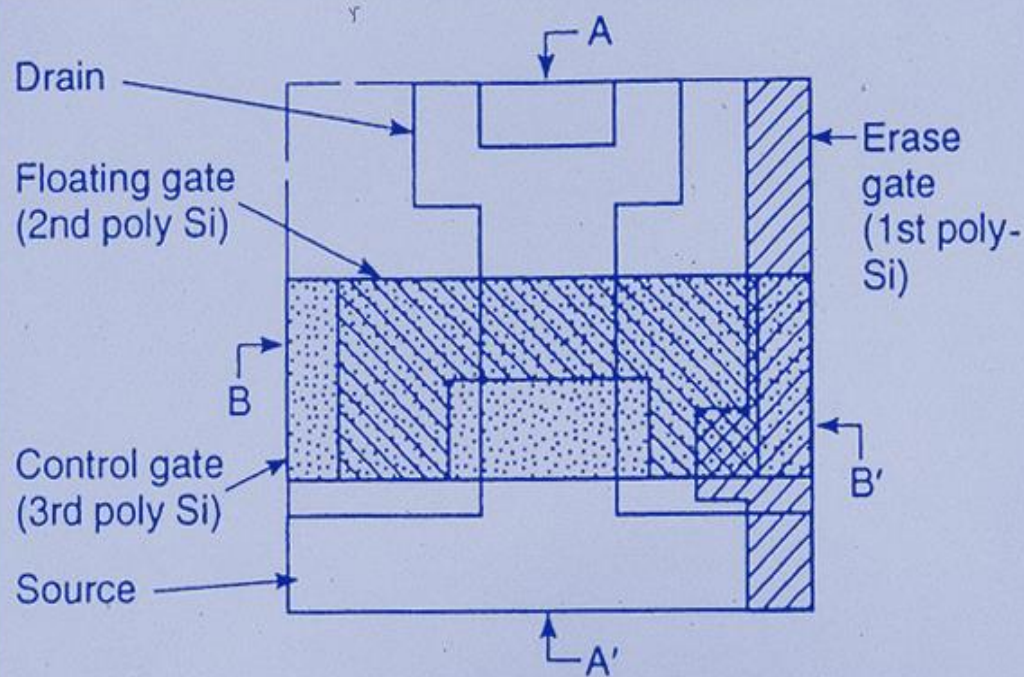


(c) Erase Mode  
(FN or DT)

# HISTORY OF FLOATING-GATE NVSM

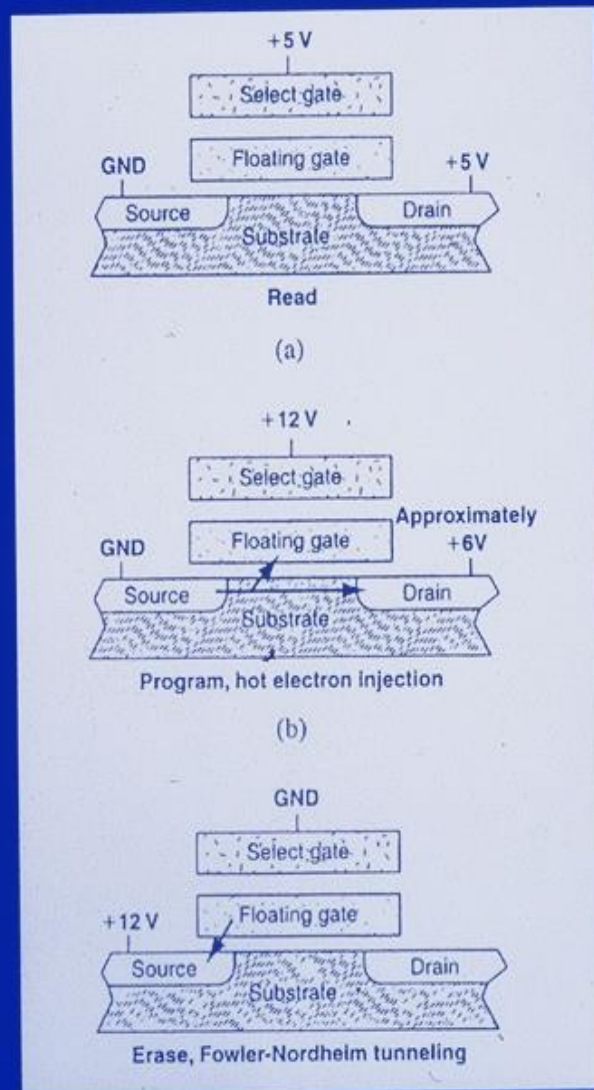
YEAR	DEVICE	INVENTOR(S)/AUTHOR(S)	ORGANIZATION
1967	Floating-Gate Concept	D. Kahng, S. M. Sze	Bell Labs
1971	EPROM-FAMOS	D. Frohman-Bentchkowsky	Intel
1976	EEPROM-SAMOS	H. Iizuka et al.	Toshiba
1978	NOVRAM	E. Harari et al.	Hughes
1984	Flash Memory	F. Masuoka et al.	Toshiba
1988	ETOX Flash Memory	V. N. Kynett et al.	Intel
1994	Room-Temperature Single-Electron Transistor	K. Yano et al.	Hitachi
1995	Multilevel Cell	M. Bauer et al.	Intel

# TOP AND CROSS-SECTIONAL VIEW OF A FLASH MEMORY (1984)





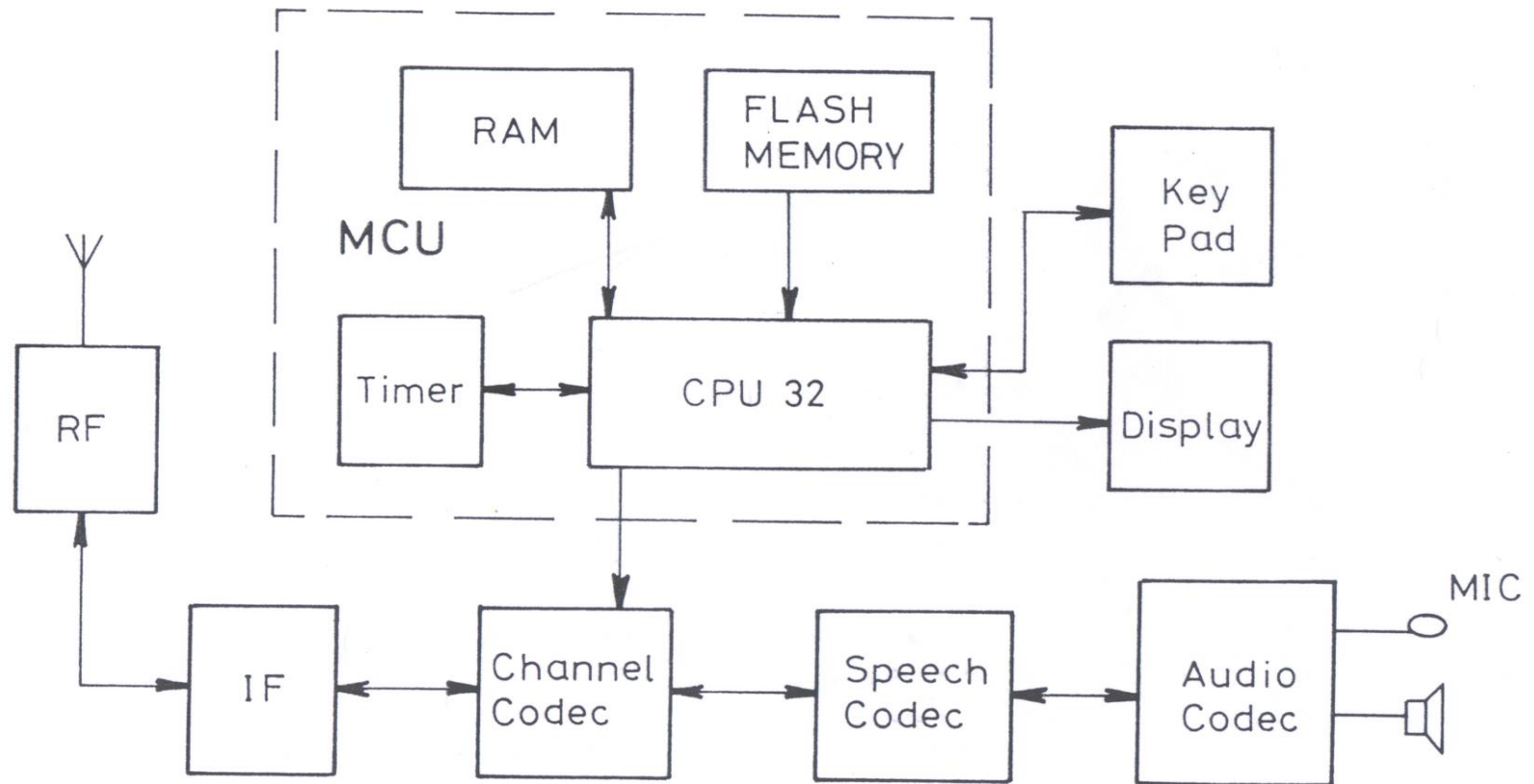
# AN ETOX CELL OPERATION (EPROM with Tunnel Oxide) in 1988



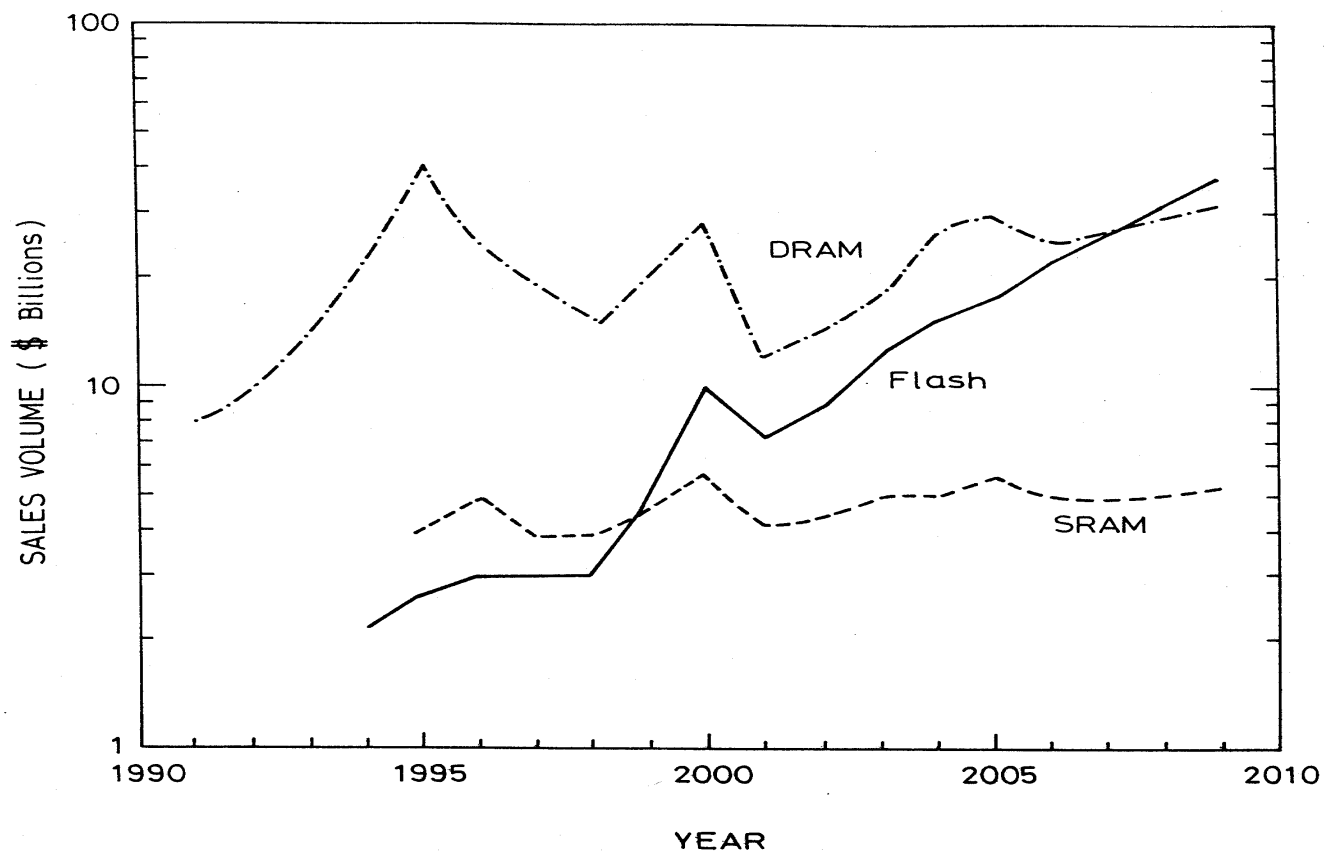
# FLASH APPLICATIONS



# CELLULAR PHONE BLOCK DIAGRAM



# SEMICONDUCTOR MEMORY MARKET



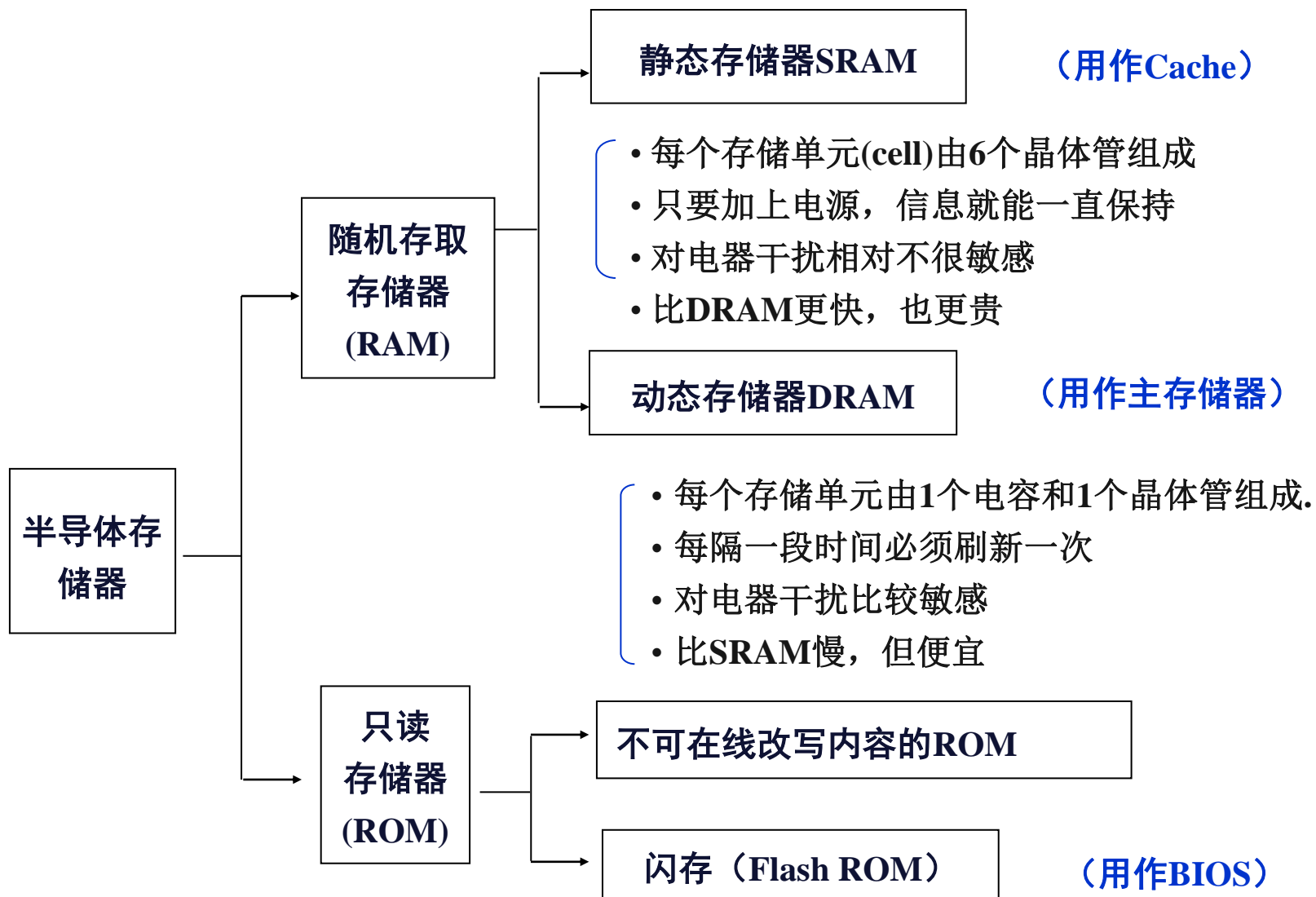


# CONCLUSION

- The nonvolatile semiconductor memory (NVSM) was invented in 1967 based on the floating-gate concept for charge storage and nonlinear transport processes for programming and erase.
- Because of its attributes of high density, low-power consumption, nonvolatility, and electrical rewritability, NVSM has surpassed SRAM in global memory market in 1999, and is poised to eclipse DRAM in the near future.
- NVSM has revolutionized electronic technology and enabled tremendous advances in portable electronic systems such as the mobile phone, notebook computer, digital photography, PDA, GPS, Smart IC card, etc.
- Many emerging NVSM technologies have been investigated to extend the floating-gate scaling to sub-10 nm, regime. A major candidate is the nanocrystal memory.



# 内存芯片的多种类型



# 五、存储器与 CPU 的连接

1. 存储器容量的扩展：由于单个芯片的存储器容量有限，为了组成较大容量的存储器，需要多个芯片进行组合。

- 位扩展（增加存储字长）：由  $mK \times n_1$  的存储器芯片组成  $mK \times n_2$  的存储器，需  $(n_2/n_1)$  片  $mK \times n_1$  的存储器芯片。
- 字扩展（增加存储字数量）：由  $m_1K \times n$  的存储器芯片  $m_2K \times n$  的存储器，需  $(m_2/m_1)$  片  $m_1K \times n$  的存储器芯片。
- 字位同时扩展：由  $m_1K \times n_1$  的存储器芯片组成  $m_2K \times n_2$  的存储器，需  $(m_2/m_1) \times (n_2/n_1)$  片  $m_1K \times n_1$  的存储器芯片。

# 五、存储器与 CPU 的连接

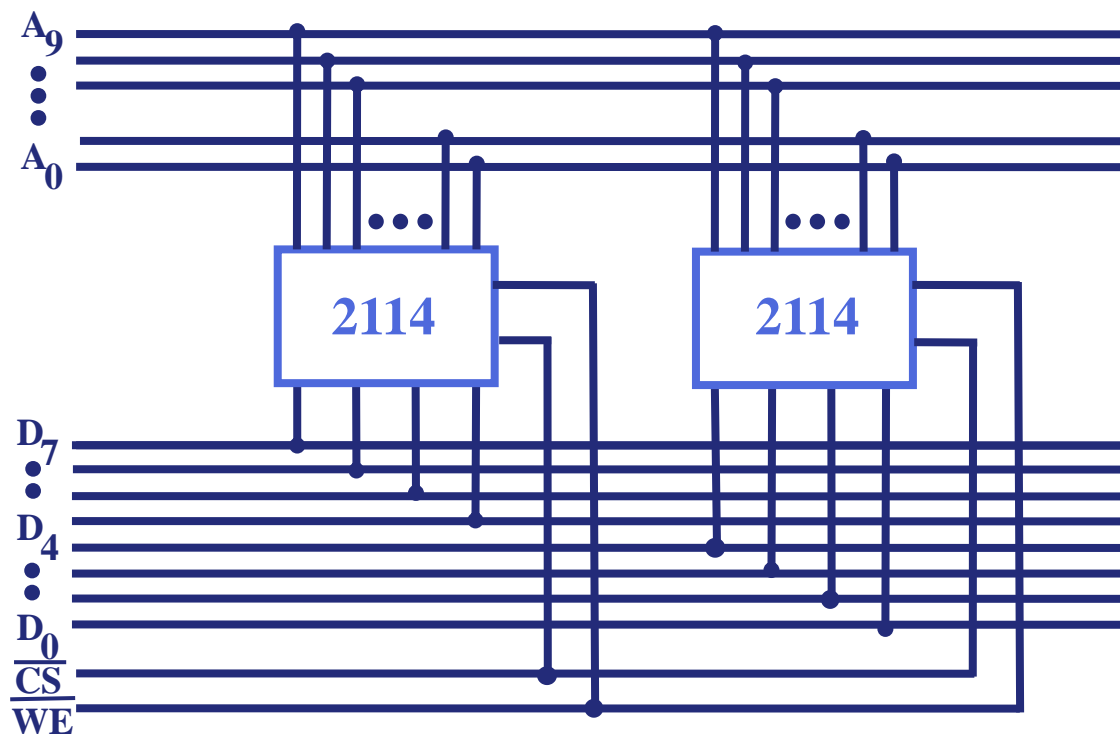
4.2

## (1) 位扩展（增加存储字长）

用 2片  $1K \times 4$  位 存储芯片组成  $1K \times 8$  位的存储器

10根地址线

8根数据线



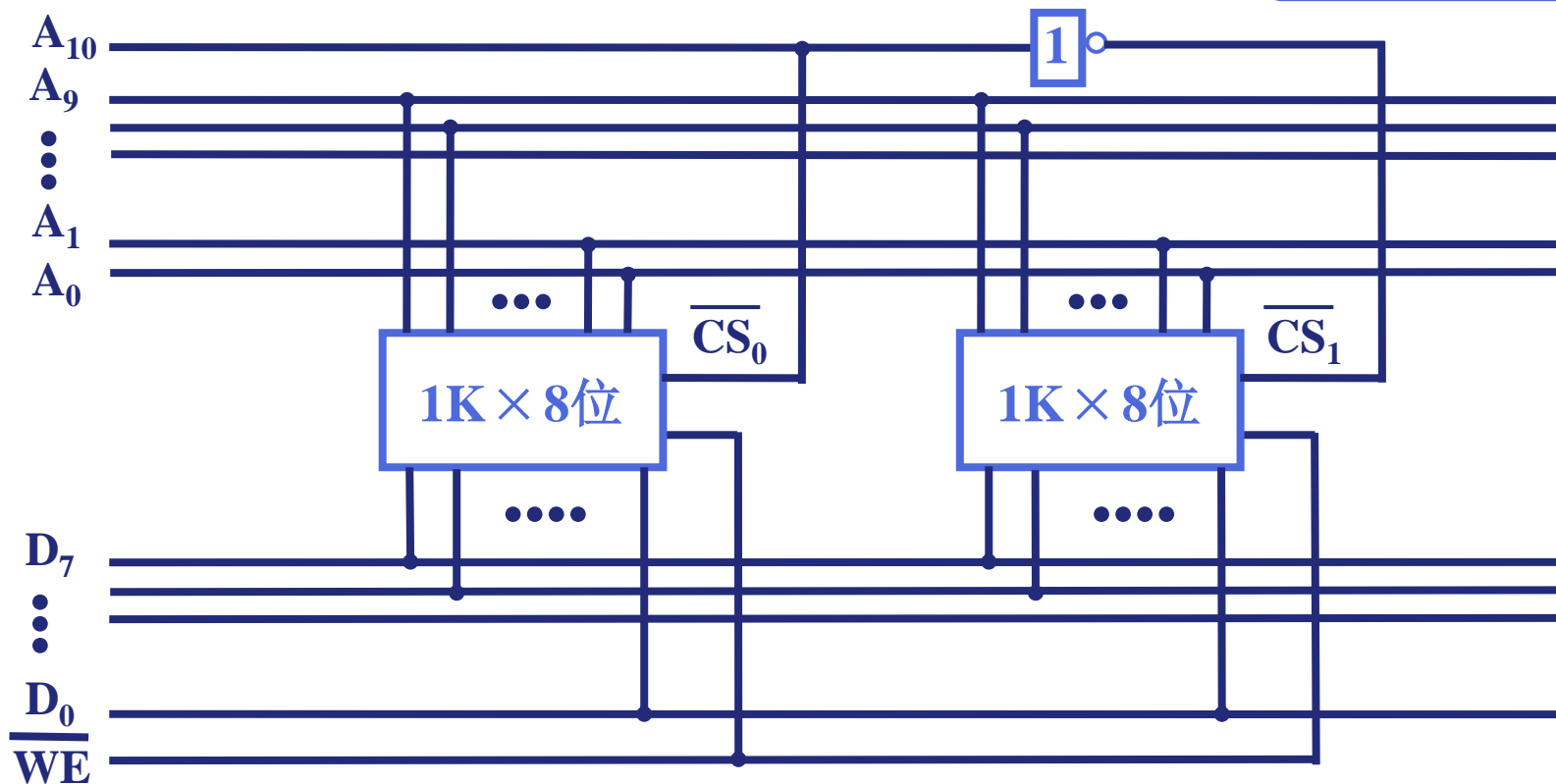
## (2) 字扩展（增加存储字的数量）

4.2

用 2片  $1\text{K} \times 8$ 位 存储芯片组成  $2\text{K} \times 8$ 位 的存储器

11根地址线

8根数据线



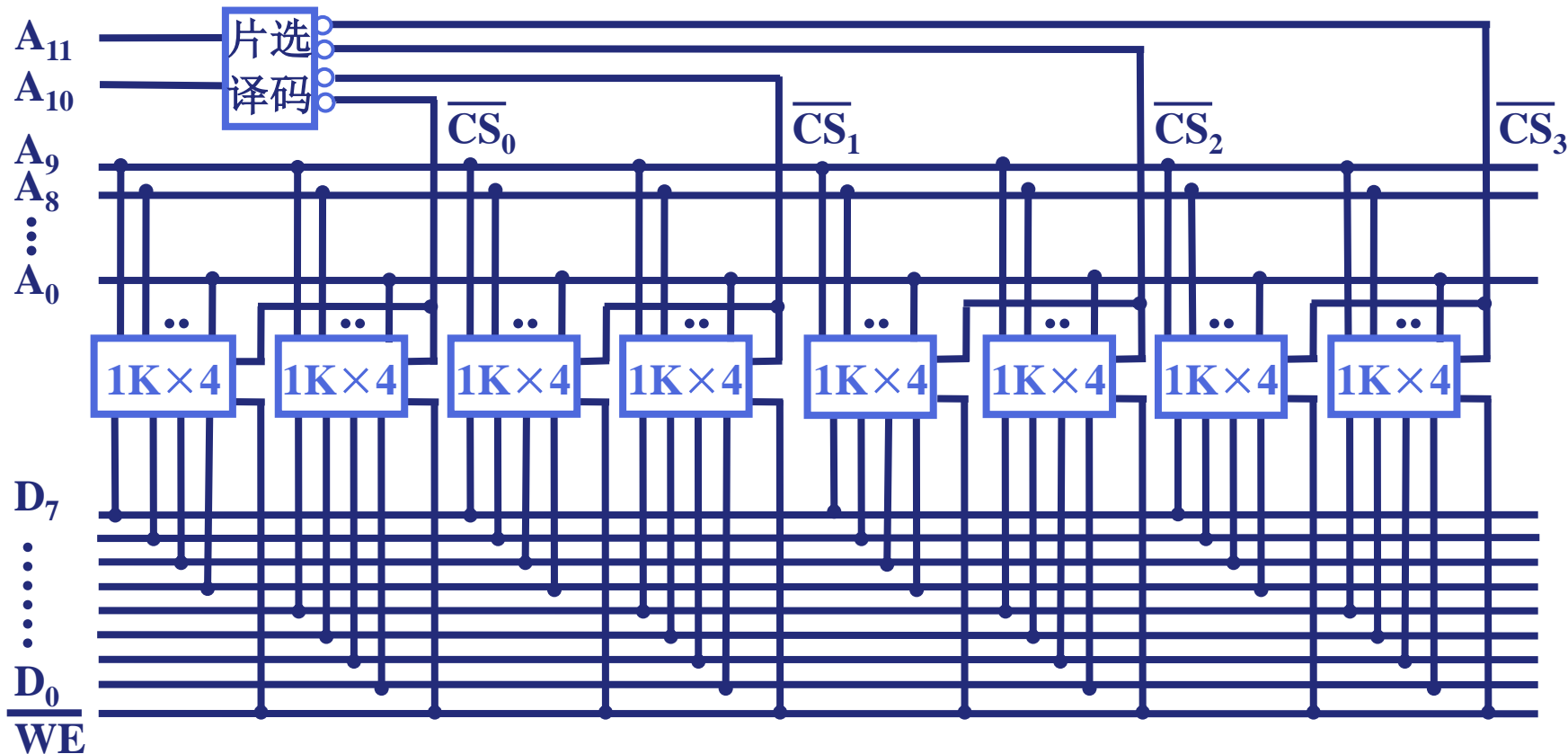
### (3) 字、位扩展

4.2

用 8片  $1\text{K} \times 4$ 位 存储芯片组成  $4\text{K} \times 8$ 位 的存储器

12根地址线

8根数据线



## 2. 存储器与 CPU 的连接

### 4.2

(1)地址线的连接:内部地址线、芯片选择线

CPU地址线往往比存储器地址线多，所以将CPU地址线低位与存储芯片地址相连，CPU地址线高位或做存储芯片扩充，或做他用。

(2) 数据线的连接:数据线对应连接

(3) 读/写线的连接： 对应连接

(4) 合理选用芯片

(5) 其他      时序、负载

## 例4.1

**例1** 设CPU有16根地址线、8根数据线，并用 $\overline{\text{MREQ}}$ 作为访存控制信号（低电平有效），用 $\text{WR}$ 作为读/写控制信号（高电平为读，低电平为写）。现有下列存储芯片：1 K $\times$ 4位RAM、4 K $\times$ 8位RAM、8 K $\times$ 8位RAM、2 K $\times$ 8位ROM、4 K $\times$ 8位ROM、8 K $\times$ 8位ROM及74138译码器和各种门电路，画出CPU与存储器连接图，要求

①主存地址空间分配

6000H~67FFH为系统程序区

6800H~6BFFH为用户程序区

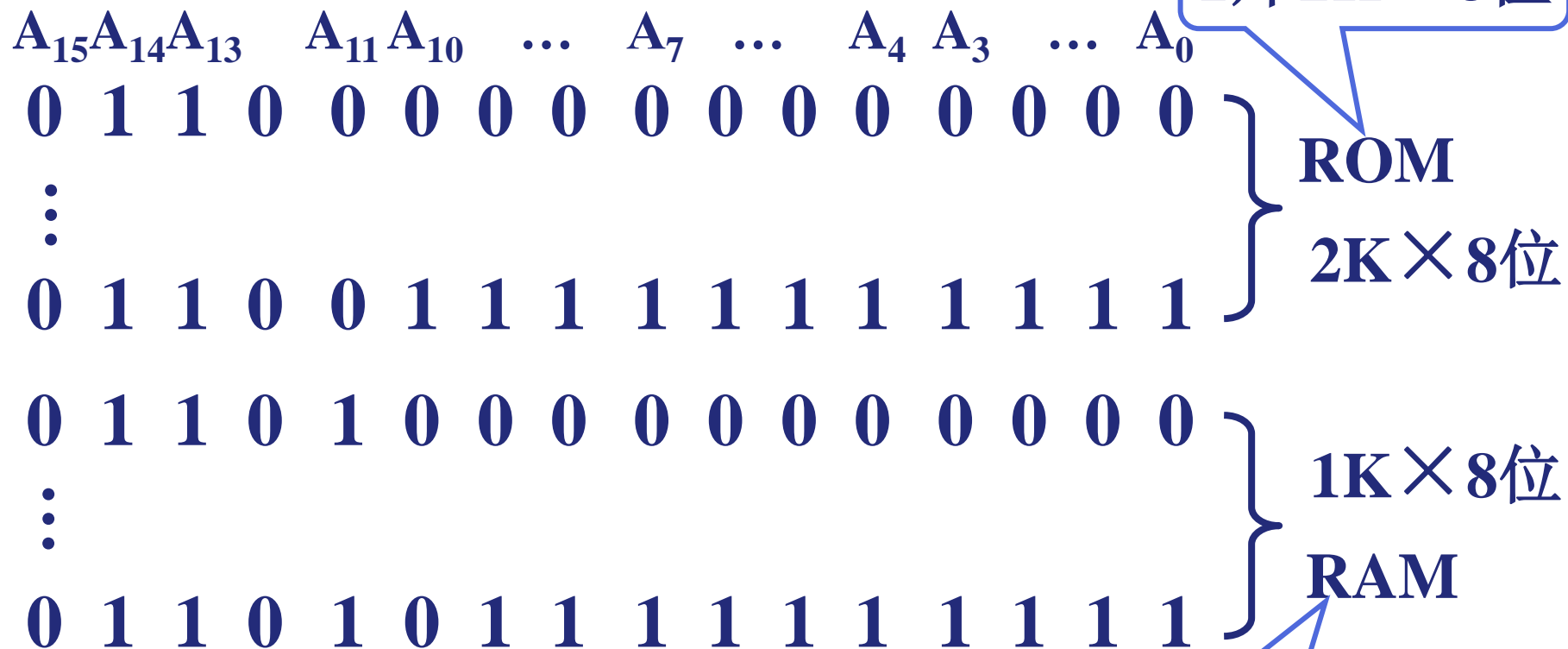
②合理选用上述存储芯片，说明各选几片

③详细画出存储芯片的片选逻辑图

## 例4.1 解:

4.2

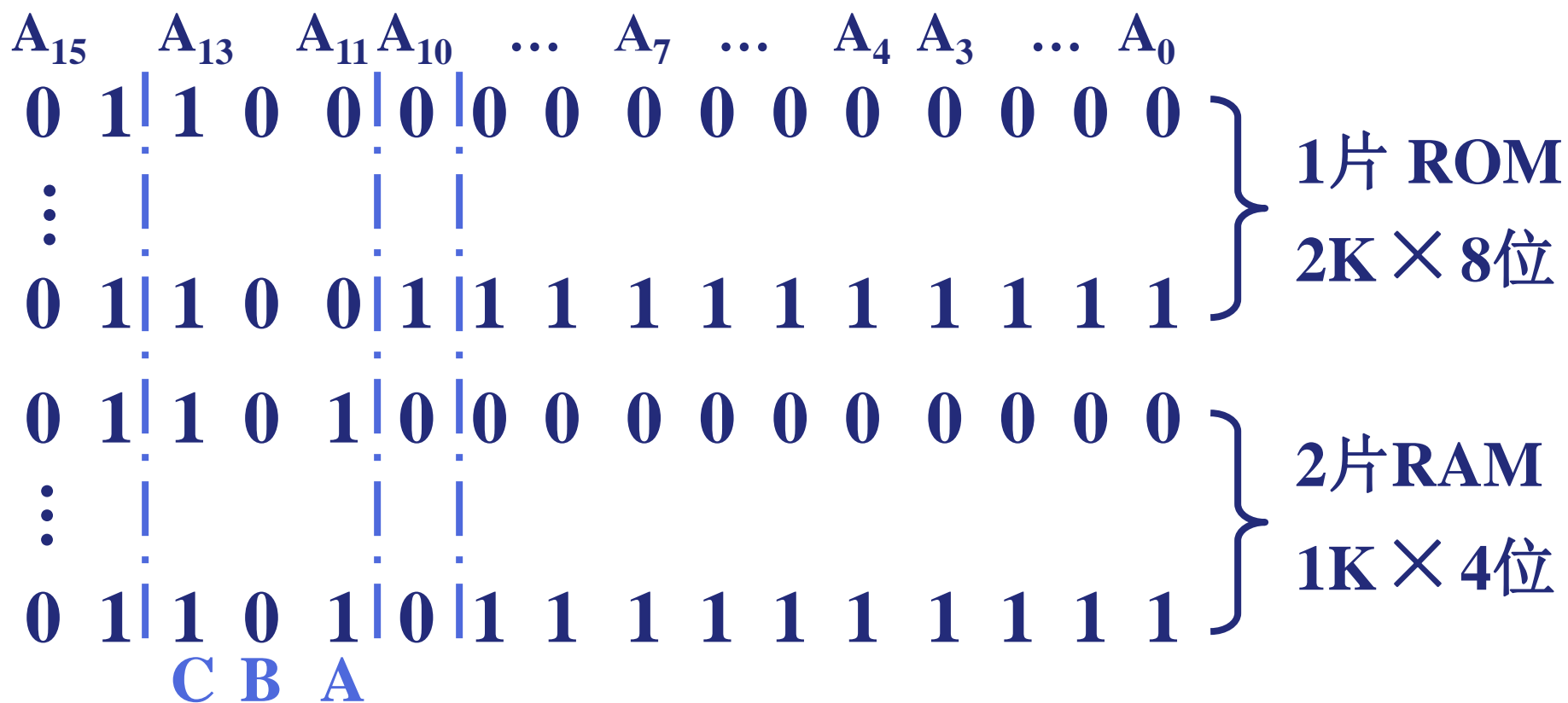
(1) 写出对应的二进制地址码



(2) 确定芯片的数量及类型



### (3) 分配地址线

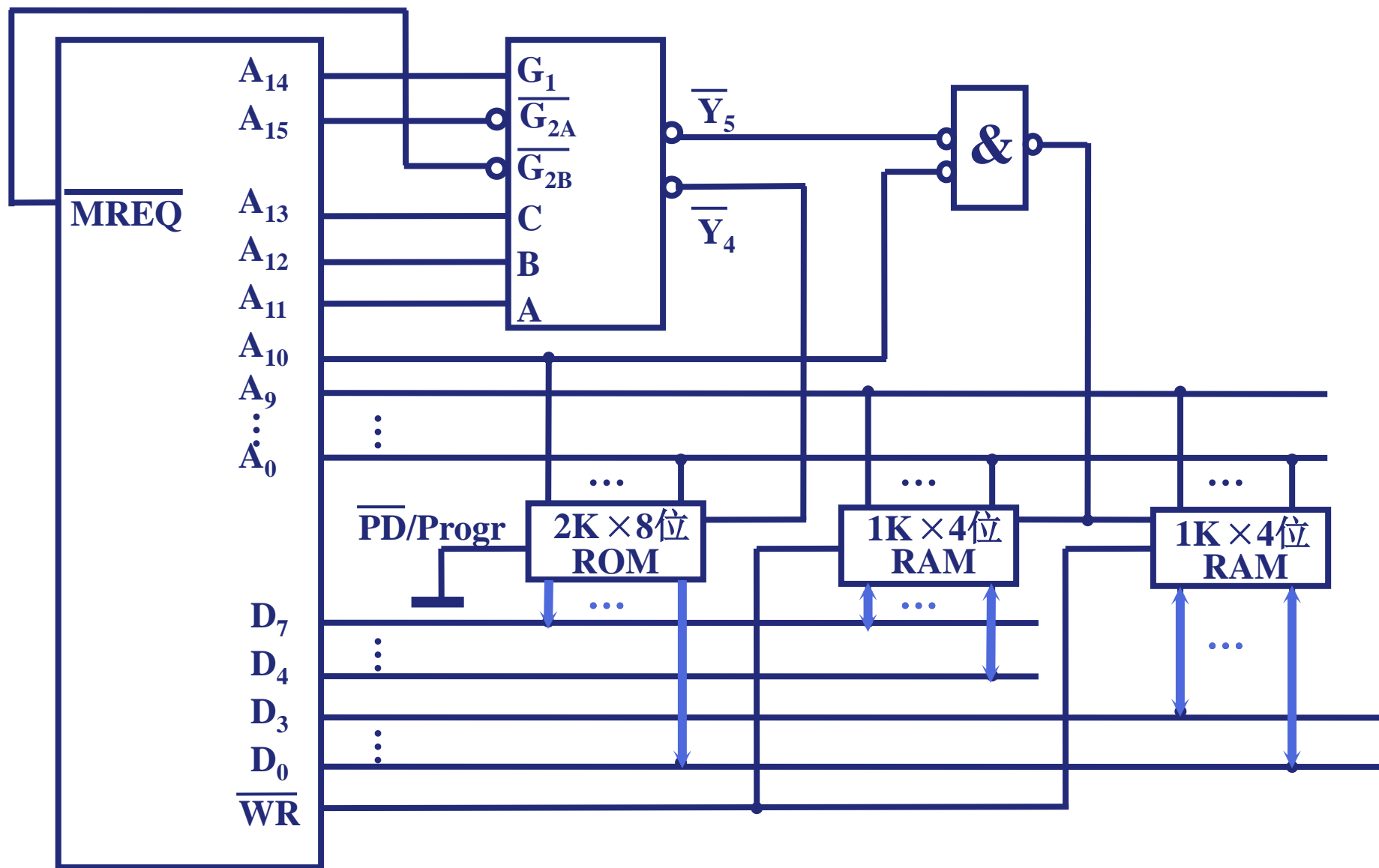


$A_{10} \sim A_0$  接  $2K \times 8$ 位 ROM 的地址线

$A_9 \sim A_0$  接  $1K \times 4$ 位 RAM 的地址线

### (4) 确定片选信号

# 例 4.1 CPU 与存储器的连接图



例4.2 假设同前，要求最小 8K为系统程序区，相邻 16K为用户程序区。最大 4K为系统程序区工作区。

(1) 写出对应的二进制地址码

(2) 确定芯片的数量及类型

1片  $8K \times 8$ 位 ROM 2片  $8K \times 8$ 位 RAM

1片  $4K \times 8$ 位 RAM

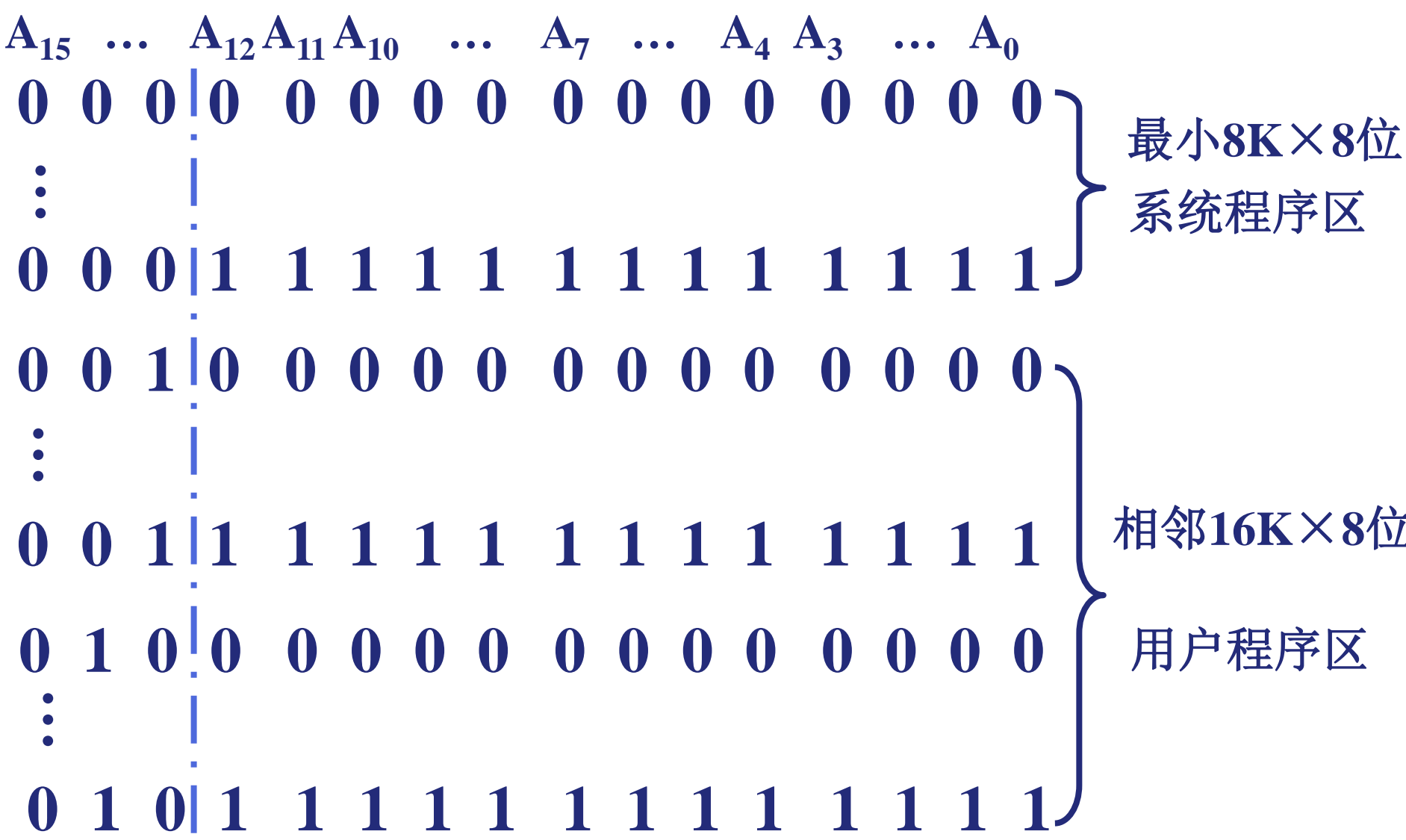
(3) 分配地址线

$A_{12} \sim A_0$  接 ROM 和  $8K \times 8$ 位 RAM 的地址线

$A_{11} \sim A_0$  接  $4K \times 8$ 位 RAM 的地址线

(4) 确定片选信号

# (1) 写出对应的二进制地址码



## (1) 写出对应的二进制地址码

$A_{15}$	...	$A_{12}$	$A_{11}$	$A_{10}$	...	$A_7$	...	$A_4$	$A_3$	...	$A_0$	
1	1	1	1	0	0	0	0	0	0	0	0	} 最大4K×8位 系统程序工作区
⋮				⋮								
1	1	1	1	1	1	1	1	1	1	1	1	

## (2) 确定芯片的数量及类型

- 最小8KB系统程序区：1片8K×8位ROM；
- 与其相邻的16KB用户程序区：选择2片8K×8位RAM；
- 最大4KB系统程序工作区：选择1片4K×8位RAM。

## 例3：CPU和主存的连接

CPU地址线A15~A0，数据线D7~D0， $\overline{WR}$ 为读/写信号， $\overline{MREQ}$ 为访存请求信号。0000H~3FFFH为BIOS区，4000H~FFFFH为用户程序区。用8K×4位ROM芯片和16K×8位RAM芯片构成该存储器，要求说明地址译码方案，并将ROM芯片、RAM芯片与CPU连接。

解：因为0000H~3FFFH为BIOS，故ROM区高两位总是00，低14位为全译码。

ROM区大小为： $2^{14} \times 8\text{位} = 16\text{K} \times 8\text{位} = 16\text{KB}$ ，ROM芯片数为：

$16\text{K} \times 8\text{位} / 8\text{K} \times 4\text{位} = 2 \times 2 = 4$ ，字方向扩展2倍，位方向扩展2倍。

ROM芯片内地址位数为13位，连到CPU低13位地址线A12~A0。

因为4000H~FFFFH为用户程序区，故RAM区高两位是01、10、11，低14位为全译码。RAM区大小为： $3 \times 2^{14} \times 8\text{位} = 3 \times 16\text{K} \times 8\text{位} = 48\text{KB}$ 。

RAM芯片数为：

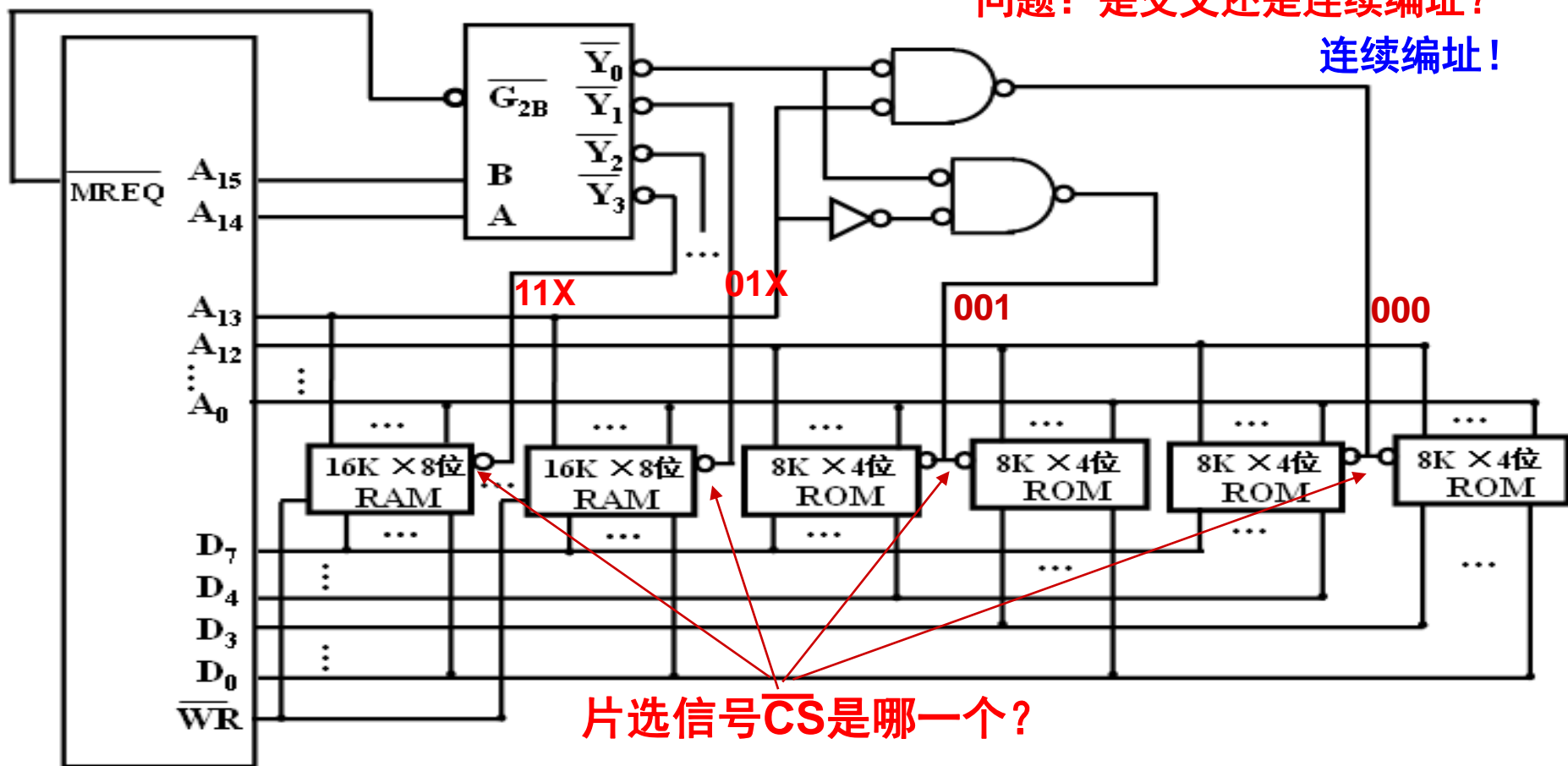
$48\text{K} \times 8\text{位} / 16\text{K} \times 8\text{位} = 3 \times 1 = 3$ ，字方向上扩展3倍，位方向上不扩展。

RAM芯片内地址位数为14位，连到CPU低14位地址线A13~A0。

ROM片选由最高三位确定。  
RAM片选由最高两位确定。

问题：是交叉还是连续编址？

连续编址！



问题：为什么 $\overline{WR}$ 不连到ROM芯片上？

ROM芯片只读不写

问题： $\overline{MREQ}$ 信号的作用是什么？

有效(低电平)时，表示选中主存读写。

# 练习

CPU有16根地址线（A15~A0），8根数据线（D7~D0）， $\overline{\text{MREQ}}$ 作为访问存储器的控制电平（低电平有效）， $\overline{\text{WE}}$ 作为读写控制电平（ $\overline{\text{WE}}=0$ 时，写允许， $\overline{\text{WE}}=1$ 时，读允许）。现有芯片：2114（1K×4位），需要扩展为2KB内存，地址范围为2000H~27FFH，片选信号由74LS138（3—8译码器）译码进行，试画出CPU与3—8译码器及存储芯片的连接。



# 存储器接口

## (1) 8位存储器接口

如果数据总线为8位，则存储器只能按字节编址

## (2) 16位存储器接口

16位存储器系统由2个存储体组成，存储体选择通过选择信号BHE实现。

如果要传送一个16位数，则2个存储器被选中

如果要传送一个8位数，则1个存储器被选中

## (3) 32位存储器接口

32位存储器系统由4个存储体组成，存储体选择通过选择信号 $\overline{BE}_3 \sim \overline{BE}_0$ 实现。

如果要传送一个32位数，则4个存储器被选中

如果要传送一个16位数，则2个存储器被选中

如果要传送一个8位数，则1个存储器被选中

# 存储器接口

## (4) 64位存储器接口

64位存储器系统由8个存储体组成，存储体选择通过选择信号BE7~BE0实现。

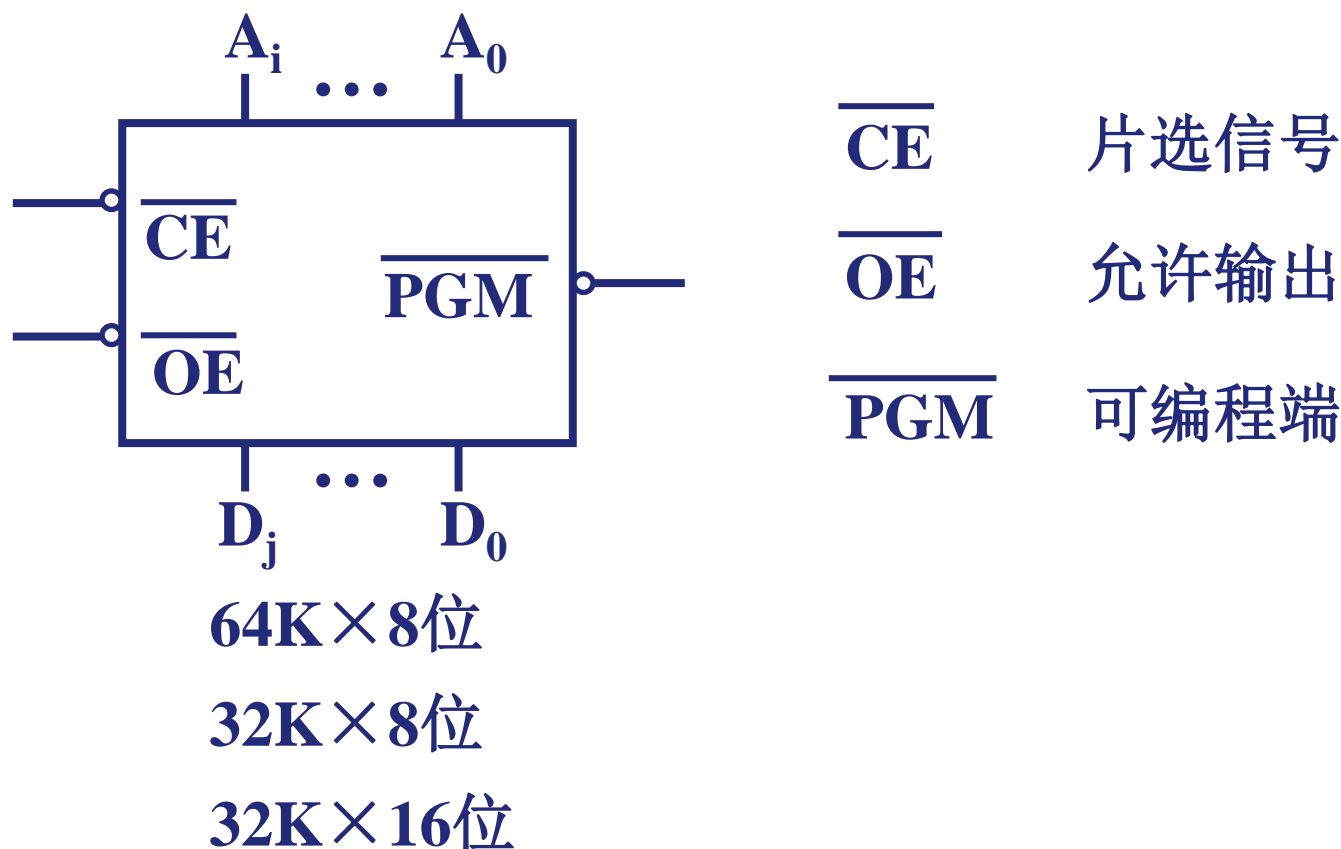
如果要传送一个64位数，则8个存储器被选中

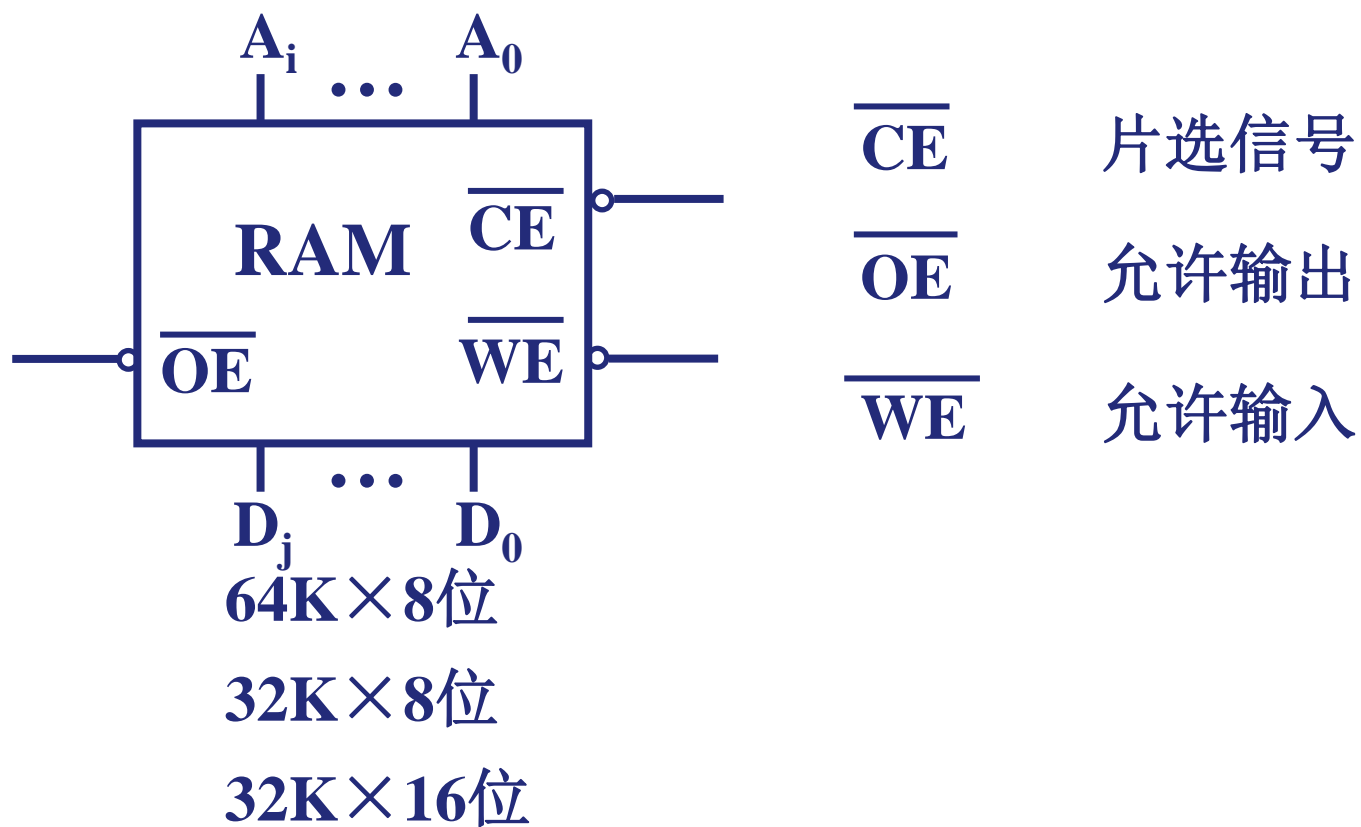
如果要传送一个32位数，则4个存储器被选中

如果要传送一个16位数，则2个存储器被选中

如果要传送一个8位数，则1个存储器被选中

**例 4.3** 设 CPU 有 20 根地址线，16 根数据线。并用  $\overline{\text{IO/M}}$  作访存控制信号。 $\overline{\text{RD}}$  为读命令， $\overline{\text{WR}}$  为写命令。现有 PROM，外特性如下：





用 138 译码器及其他门电路（门电路自定）

- (1) CPU按字节访问和按字访问的地址范围各多大？
- (2) CPU按字节访问时需分奇偶体，且最大64KB为系统程序区，与其相邻的64KB为用户程序区，写出每片存储芯片对应的二进制地址码。
- (3) 画出CPU与存储芯片连接图。

BHE	A0	访问形式
0	0	字
0	1	奇字节
1	0	偶字节
1	1	不访问