

# Chapter3.存储器(Part1)

## 1. 概述

### 1.1. 存储器分类

#### 1.1.1. 概述

**1** 按在计算机中的作用：

内部存储器(内存)=主存+Cache，和CPU交换信息

外部存储器(外存，储存)，位于主机外，按照IO方式管理

控制存储器，位于CPU内，是控制单元CU的核心，用于存储微程序

**2** 按照存储器的截止：基本要求是只要能区别两种物理状态(0/1)，所以有半导体/磁材料/磁光(光盘)

**3** 按存取方式：

RAM(随机存取存储器)：可读可写，任何一个单元内容都可随机存取，访问时间一样

ROM(只读存储器)：例如主板上的BIOS固件就放在ROM，一般情况只读不写，非电易失

SAM(顺序存取存储器)：典型例子如磁带，读/写操作时只能顺序地对数据所在物理地址进行查找

DAM(直接存取存储器)：半顺序存取存储器，如磁盘，读数据要经过两个过程

1. 找磁道，把磁头移过去，这一过程速度慢，相当于随机寻址/直接寻址
2. 转动主轴，按照顺序寻址，在磁道里找到一块数据，这一过程速度取决于转速

AM(相联存储器)：也叫CAM(按内容存储器)，按照存储关键字是否匹配来访问，实现快速检索

#### 1.1.2. 半导体存储器的分类

**1** 按制造工艺：双极型存储器，MOS型(最常用)

**2** 按信息可保存性：电易失，非电易失

#### 1.1.3. RAM存储器的分类

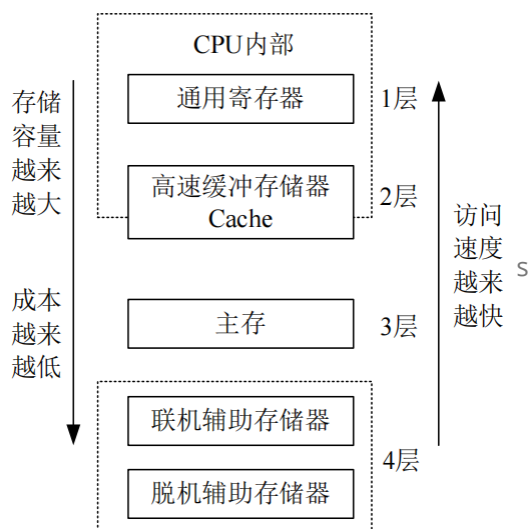
**1** SRAM：静态随机存取存储器，速度快/稳定/不需刷新/集成低/功耗大/成本高/常用于Cache

**2** DRAM：动态随机存取存储器，集成高功耗低、价格便宜，需定时刷新。常用于大容量主存

**3** NVRAM：非易失性随机存取存储器，结合SRAM和EEPROM(或者硬盘)，掉电时通过电池给ARM供电继续将数据写入至EEPROM

## 1.2. 存储系统的层次结构：解决容量/价格/速度的矛盾

### 1.2.1. 现代计算机存储结构层次



注意1：辅助存储器诸如硬盘

注意2：主存和硬盘之间实际上有一个虚存，可以提高内存容量

### 1.2.2. 层次间的原则

- 1 包含性：内层信息一定被包含在其外层的存储器中，例如以上M1的信息必须被包含在M2当中
- 2 一致性原则：不同层次中同一信息的值必须保持相同，举个例子例如在Cache里 $a=1$ ，当CPU对 $a$ 写入后 $a=2$ 时，为保持一致Cache会继续写储存中的 $a$ 使得贮存中 $a=2$

### 1.2.3. 存储系统工作原理

#### 1.2.3.1. Cache—主存层间：Cache硬件实现，对程序员不可见

- 1 程序运行局部性：程序倾向于一段时间内访问集中一小片区域，最近被访问的数据可能再次被访问
- 2 设计原理：将当前使用最多的一小段程序(程序I-Cache)和数据(数据D-Cache)放在cache中
- 3 总体效果：CPU访存速度接近于cache速度，寻址空间仍为主存空间，价格接近主存

#### 1.2.3.2. 主存—辅存层间：虚存对系统程序员可见

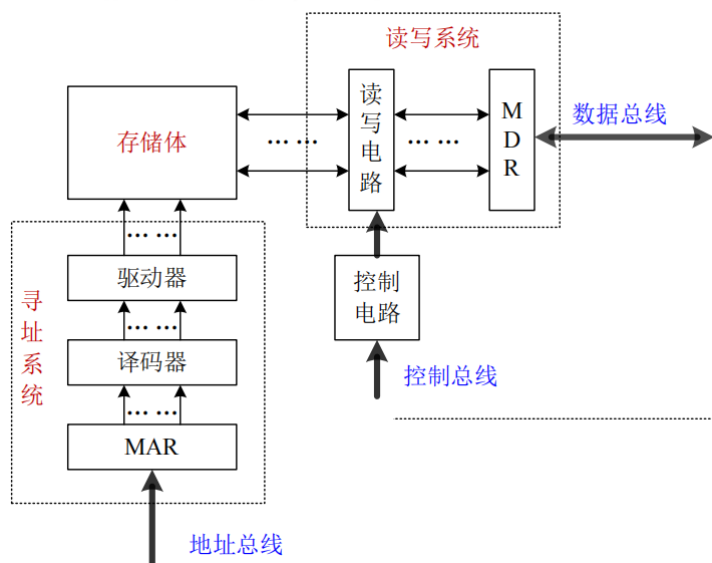
- 1 虚存：主存+辅存的一部分  $\xrightarrow{\text{软硬结合技术}}$  虚拟存储器
- 2 总体效果：容量&位价接近辅存，速度接近主存

### 1.2.4. 存储系统的设计问题

- 1 地址映射：哪些东西放主存/放Cache)
- 2 数据一致性
- 3 地址变换：CPU访存只给出主存单元，这样的话如何访问不同层次单元

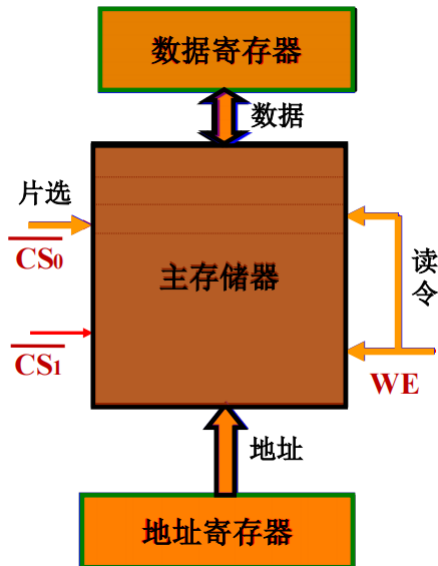
## 2. 主存储器

### 2.1. 主存储器的基本结构



MAR/MDR在逻辑上属于主存，但在物理上一般都集成在CPU中。驱动器/读写电路实质上是一个放大器

#### 2.1.1. 主存储器的读写过程



##### 1 信号含义：

1. 片选信号 $\overline{CS_0}$ ：低电平有效，低电平存储器工作，高电平不工作
2. 写使能信号 $WT$ ：高电平有效，高电平写，低电平读

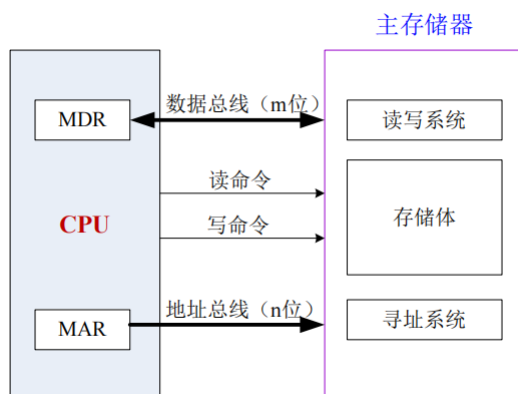
##### 2 读过程：

1. 地址写到MAR地址寄存器当中，MAR把地址送主存储器
2. 满足 $\overline{CS_0}$ 低电平， $WT$ 低电平条件后，数据可以读出，随后送MDR数据寄存器

##### 3 写过程：

1. 地址放到MAR地址寄存器当中，要写入的内容放到MDR数据寄存器当中
2. 满足 $\overline{CS_0}$ 低电平， $WT$ 高电平条件后，把数据写入相应的单元

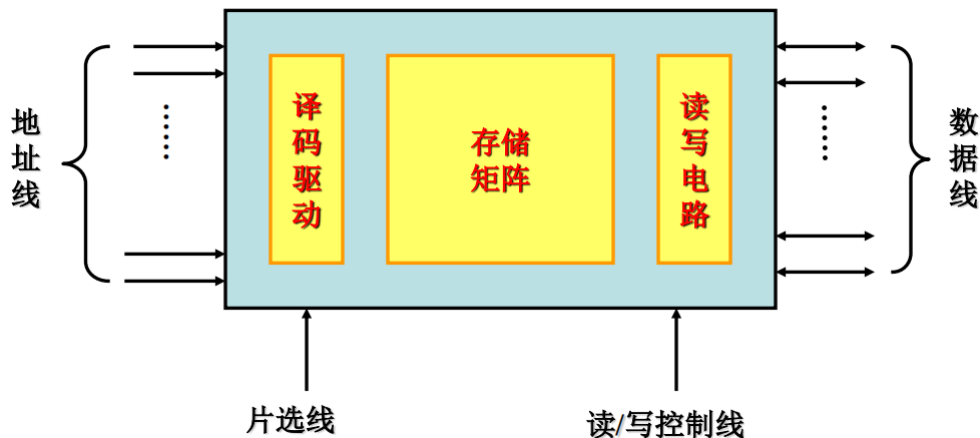
## 2.1.2. 主存储器与CPU的连接



- 1 数据总线：多少位计算机，数据总线就是多少位，现在一般64位
- 2 地址总线：通常少于计算机位数，其宽度决定了CPU可以直接寻址的内存大小
- 3 MAR：暂存CPU正在访问的存储单元的地址
- 4 MDR：暂存CPU和主存间交互的数据

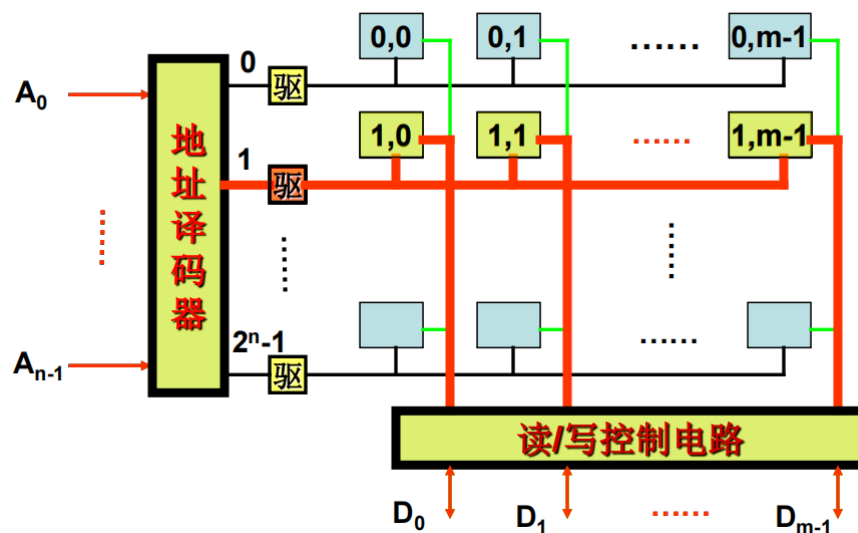
## 2.1.3. 半导体芯片(内存颗粒)：内存条的基本单元

### 2.1.3.1. 基本结构



### 2.1.3.2. 半导体芯片的地址译码方式

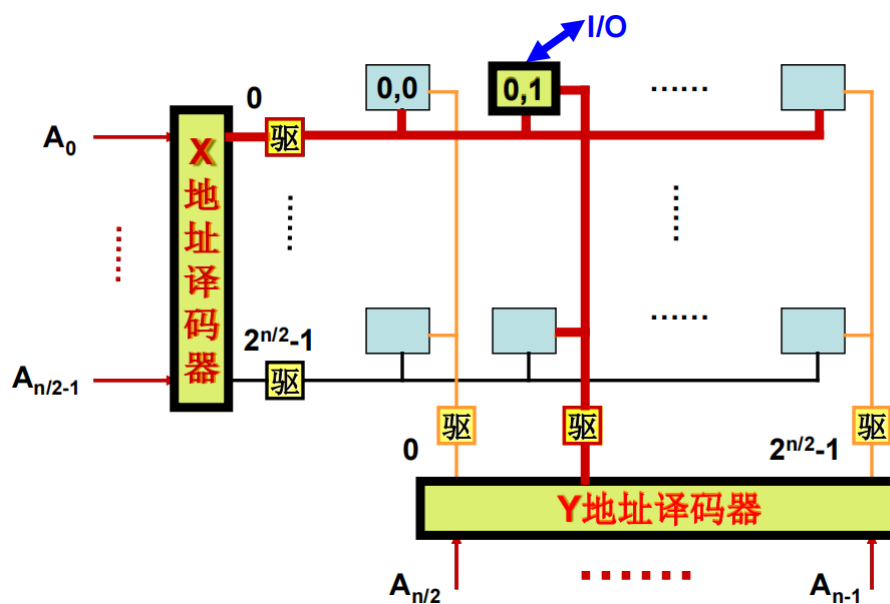
- 1 线选法(单译码结构)
  1. 原理：地址译码后直接选中一个存储单元的所有位，导致 $n$ 位地址需要 $2^n$ 个地址译码线，而每根地址译码线后面都需要加一个驱动器
  2. 结构示意图：容量为 $2^n$ ，每个字中有 $m$ 位。 $n$ 根地址线代表 $n$ 位地址，译码后输出 $2^n$ 根线，每根线后接一驱动器，驱动器可以选中 $m$ 个存储元，然后读出/写入选择单元



3. 特点：速度快，但是成本太高了(译码输出线太多)；把这个特点反过来就是下面的特点

## 2 重合法(双译码结构，同样的结构甚至可以拓展到三维)

1. 原理：以16位为例，将地址分为行列两组每组八位，每组分别用各自的一套译码器译码，然后(举个例子)高位给X低位给Y，行列译码的重合点就是所选单元，所以此时只需要地址译码线根数： $2^{\frac{n}{2}} + 2^{\frac{n}{2}}$
2. 结构示意图：可以看出，重合点的充要条件是对于两个驱动器都有效



## 2.2. 主存的性能指标

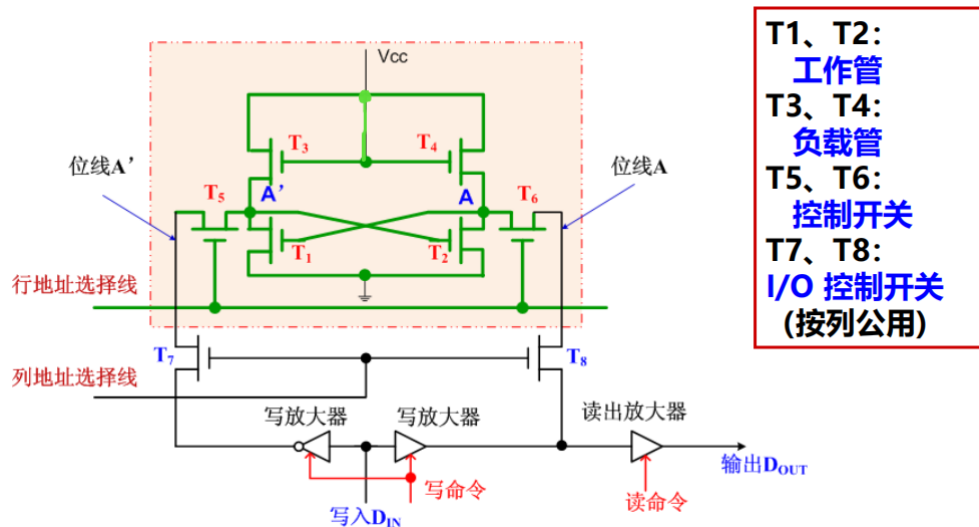
- 1 存储容量 = 存储单元个数 × 存储字长（按字）= 字节数（按字节编址）
- 2 存取速度：存取周期(两次操作时间间隔) = 存取时间(存储器操作启动到完成所需时间) + 恢复时间
- 3 存储器带宽：单位时间内存储器存取的信息量，数值上=1字/存取周期(单位秒)

注意速度上1M=1000K，容量上才是1M=1024K

## 2.3. SRAM存储器(单位容量极其贵, Cache)

### 2.3.1. 存储元结构：六管SRAM存储元

#### 2.3.1.1. 结构概览

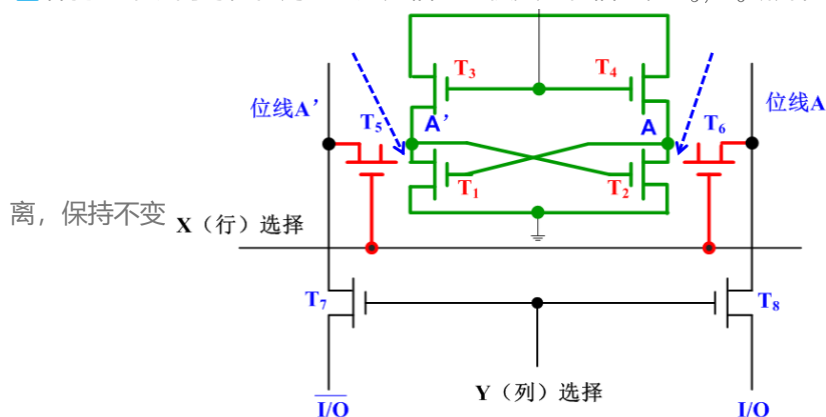


$T_3, T_4$ 相当于两个电阻, 提供默认的高电平

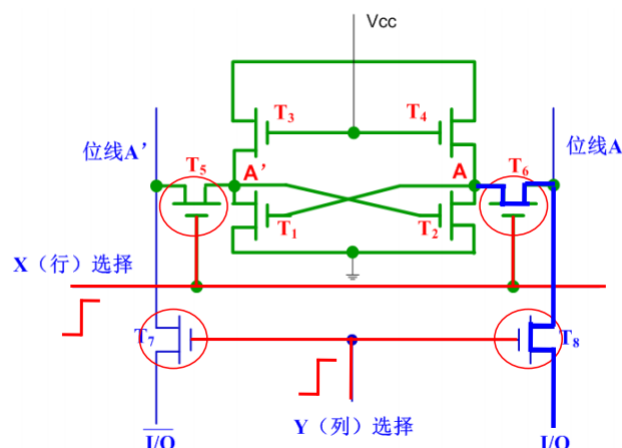
$T_1, T_2$ 构成一个锁存器, 能锁存住0/1

#### 2.3.1.2. 六管SRAM三种状态

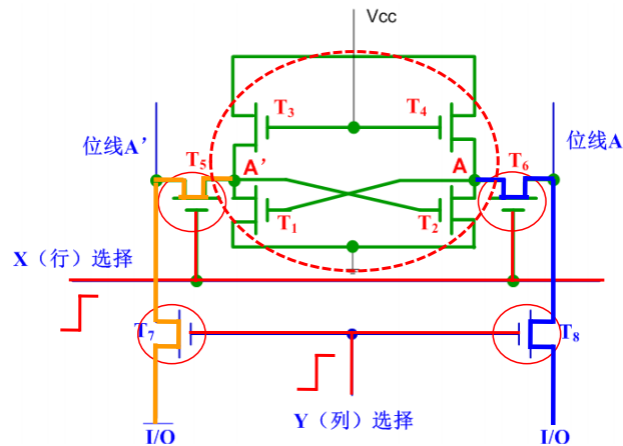
1 保持: 未访问时, 行列地址选无信号or仅其一有信号,  $T_5, T_6$ 断开, 存储元与外界隔离, 保持不变



2 读出:  $X$ 有效则打开 $T_6$ ,  $Y$ 有效则打开 $T_8$ (行列信号均有效), 存储元所存信息经 $T_6 \rightarrow$  位线 $A \rightarrow T_8 \rightarrow$  读放大器, 且读出的方式是单边的, 例如右边读出1左边就会读出0, 只要读一边就行

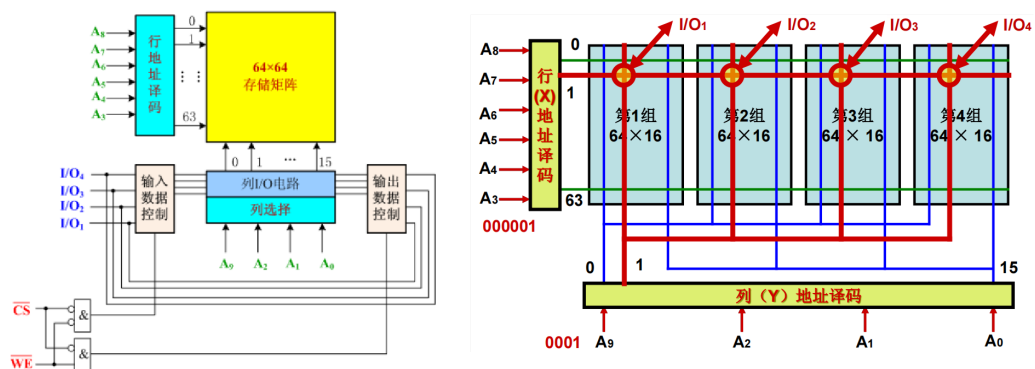


**3** 写入：行列信号均有效，所有开关导通，顺序为  $D_{in} \rightarrow$  写放大器  $\rightarrow T_8/T_7 \rightarrow$  位线  $\rightarrow T_5/T_6 \rightarrow A'/A$  迫使  $T_1/T_2$  改变状态，必须双边写，一边为1一边为0，两边都为1时状态不确定



## 2.3.2. SRAM示例：Intel 2114

容量1K×4位，存储矩阵64×64



**1** 容量1K×4位：共有1K个存储元，1K对应10个地址线；每个单元的数据有4位，对应4个数据线

**2** 64×64：等于 $2^{12}$ ，编成地址需要12个地址线？没有冲突，因为一个单元有四位，也就是说一次性要读四位出来，这也决定了Y列的设计

**3** 存储矩阵结构：分为四组，具体读的时候，1234四组中每个都读一位出来

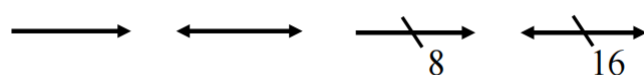
**4** 示例：X=000001，Y=0001，如图所示，X行对应一条线划过去，Y列对应一条线分成四条划过去，得到四个交叉的点，这四个合在一起就是要读的数据

## 2.3.3. SRAM的外特性

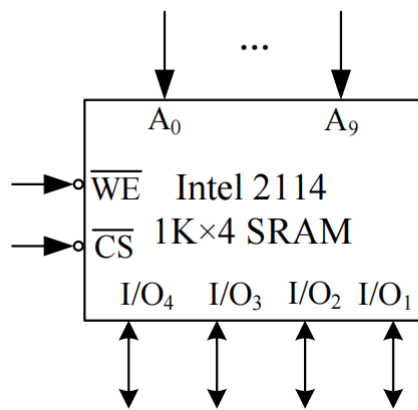
**1** 逻辑符号方框图表示：方框内标注芯片引脚/信号名(引脚上所加信号名)，外标注型号/片容量/引脚名(引脚的逻辑定义)

**2** 引脚表示方法

1. 电气引脚：电源线，地线等，逻辑符号中可省
2. 逻辑引脚：单向的是地址线，双向的是数据线(可写可读)，然后右边的分别是8or16位数据的总线



**3** Intel 2114外特性(小圈圈表示低电平有效)



**A9~A0:** 地址线

**I/O<sub>4</sub>~I/O<sub>1</sub>:** 数据线, 双向

**$\overline{\text{WE}}$ :** 读写读令

= H: 读令

= L: 写令

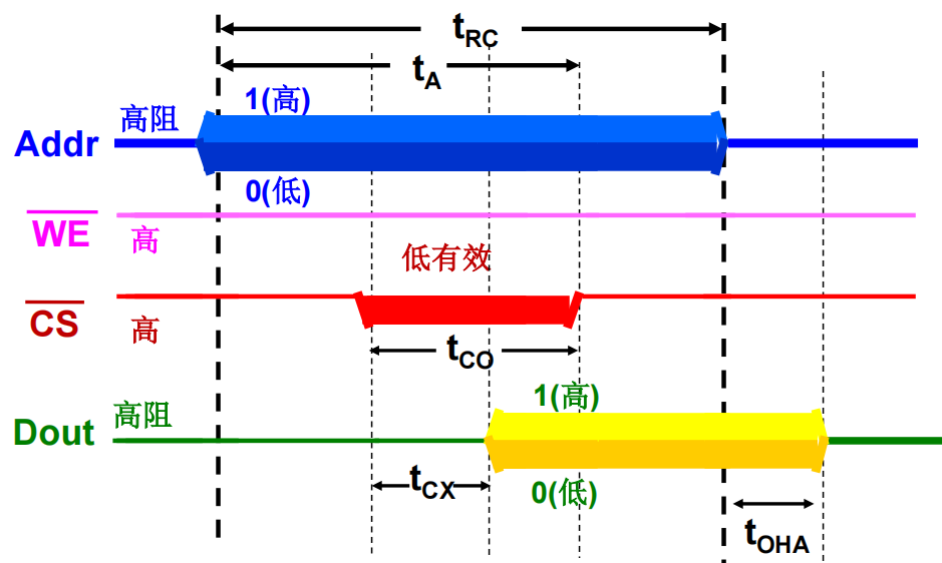
**$\overline{\text{CS}}$ :** 片选

低有效 (选中)

## 2.3.4. SRAM读写时序

### 2.3.4.1. 读周期时序

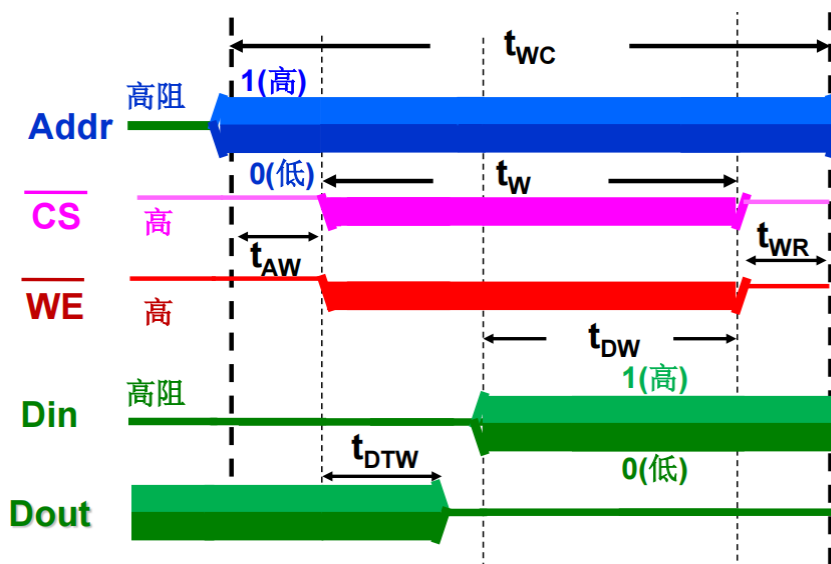
$t_{RC}$ 为存取时间(无法太短),  $t_{CO}$ 为片选时间



- 1 先改地址, 然后战术停顿一下防止信号干扰
- 2 然后让片选地有效, 存储器开始工作
- 3 一段时间后SRAM就会通过Dout(数据线)将数据送出来
- 4 存取时间结束后, 数据线还会输出一会, 这就是所谓的恢复时间



### 2.3.4.2. 写周期时序

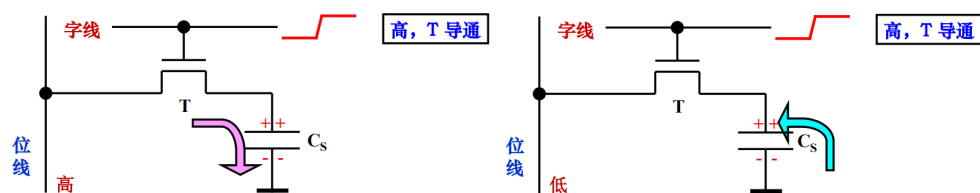


注意数据是双向的，线写入一个数据，数据写完终止后会把这个数据输出

## 2.4. DRAM(内存条用的多)

### 2.4.1. 单管DRAM存储元

#### 2.4.1.1. 概览



- 1 位线连接数据线，字线连接地址线经过译码器之后的线
- 2 电容CS容量特别小，通过其有/无存储电荷来区分信号1/0

#### 2.4.1.2. 写操作

- 1 写入1操作(左)：选中该地址(字线高电平)从而MOS管导通，再设置位线为高电平，由此CS电容充电，1写入/保持原有的1
- 2 写入0操作(右)：选中该地址(字线高电平)从而MOS管导通，再设置位线为低电平，由此CS电容放电，0写入/保持原有的0

#### 2.4.1.3. 读操作

如右图，选中该地址(字线高电平)从而MOS管导通

- 1 若CS无电荷，位线上无电流，读出为0
- 2 若CS有电荷，则发生放电，位线上的读出放大器检测到这种变化，读出1

但是问题在于这样操作的话会使读出后电荷放掉，即被读单元的内容一定被清为零，因此需要把读出

的内容重写写回，所以需要再生

#### 2.4.1.4. DRAM的再生与保持

**1** 再生：读放大器同时又是再生放大器，读出过程中建立起稳态，然后该稳态再自动写回存储元。

**2** 保持：存储元未选中时，CS电容与外界隔离(功耗近似为0)以保持原电荷状态不变

但是这里只是近似为0，一直放那里还是会有漏电的，所以需要刷新

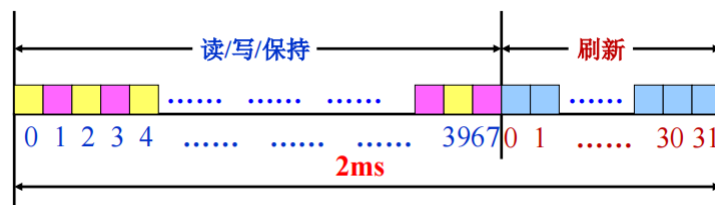
#### 2.4.2. DRAM的定期刷新

定时补充CS的漏电，大致2-8ms一次

一下以Intel 1103 1K×1 DRAM为例，内部矩阵为32行×32列，存取周期500ns，最大刷新间隔2ms(4000个存取周期)

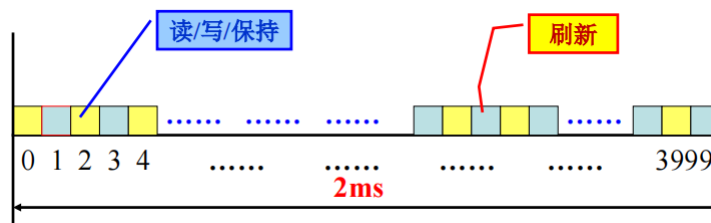
##### 2.4.2.1. 集中式刷新

在2-8ms内，对芯片内全部存储元逐行刷新一遍。缺点在于刷新的时候不能读写，全部刷新会造成全部不能读写



##### 2.4.2.2. 分散式刷新

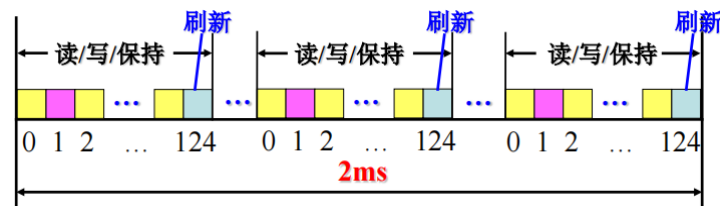
每读/写一次我就刷新一次，每次只刷新一行，每行轮着来。如图蓝块时刷新，黄块时读写。缺点是刷新的次数太多了



##### 2.4.2.3. 异步刷新

最大刷新间隔(2ms)内，对芯片内的全部存储元逐行轮流刷新一遍。刷新周期平均分散在最大刷新间隔中

在Intel 1103 1K×1 DRAM中，要求2ms内刷新32行(次)，也就是 $2\text{ms}/32=125$ 存储周期



### 2.4.3. DRAM存储器的结构特点

1 分为行地址列地址：解决DRAM内存较大地址线太多

使得一个地址线分时传递两个地址信号， $\overline{RAS}$ 信号有效时送行地址， $\overline{CAS}$ 信号有效时送列地址

由此增加一根地址线，容量能够增加 $2 \times 2 = 4$ 倍

PS, 行地址一般为高位地址

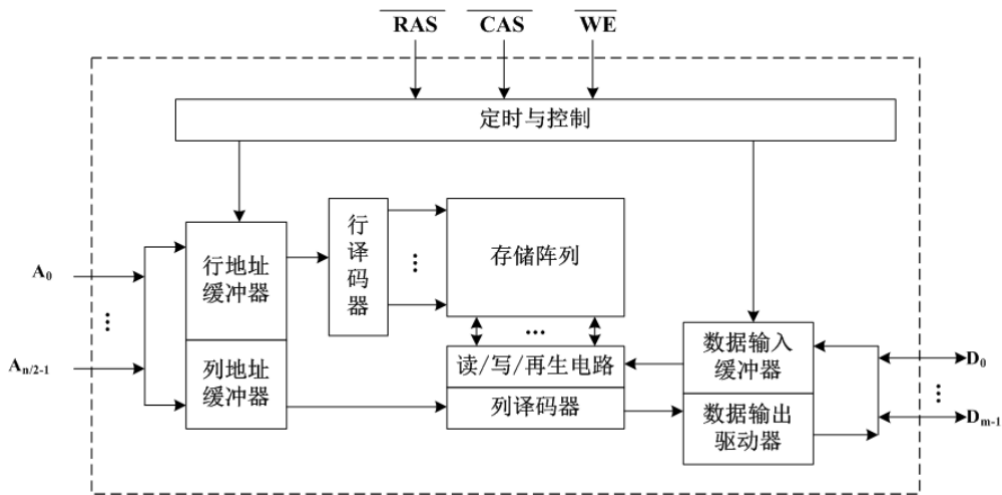
2 无 $\overline{CS}$ 引脚：进一步减少引脚，片选功能被兼并进 $\overline{RAS}$ ，所以

送行地址时： $\overline{RAS}$ 有效(行地址，片选)

送列地址时： $\overline{RAS}$ 有效(片选)， $\overline{CAS}$ 有效(列地址)

### 2.4.4. DRAM内部结构与外部特性

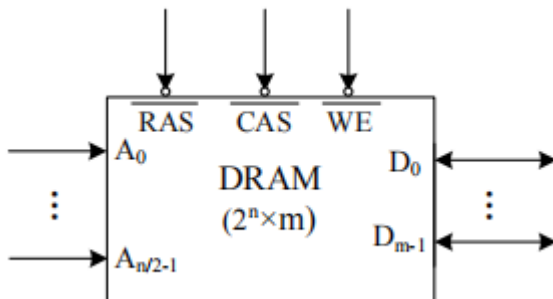
#### 2.4.4.1. 外部结构



读取地址的大致过程：

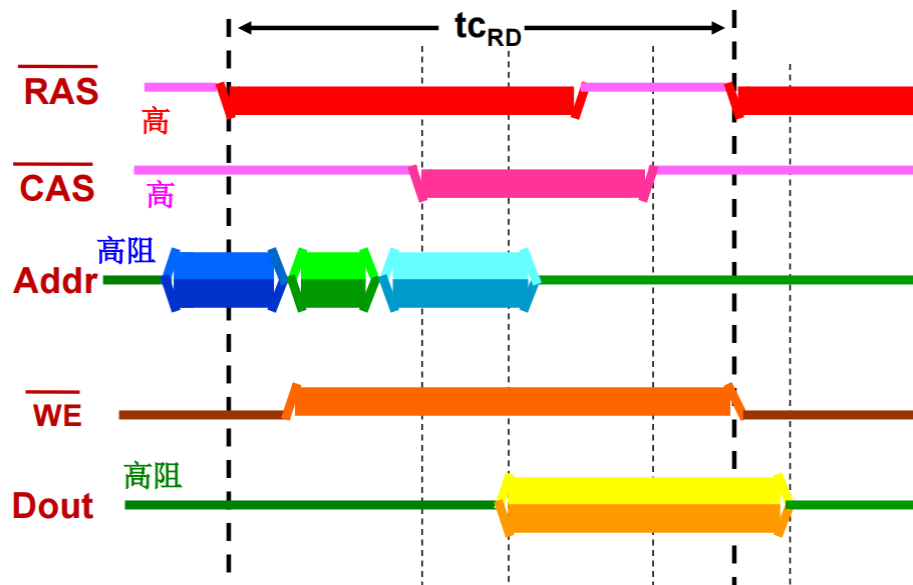
行地址列地址冲进行/列地址缓冲器(寄存器)→在送入行/列译码器→存储阵列中读取地址

#### 2.4.4.2. 存储器的外特性



## 2.4.5. DRAM读/写时序

### 2.4.5.1. DRAM读周期时序(更快)



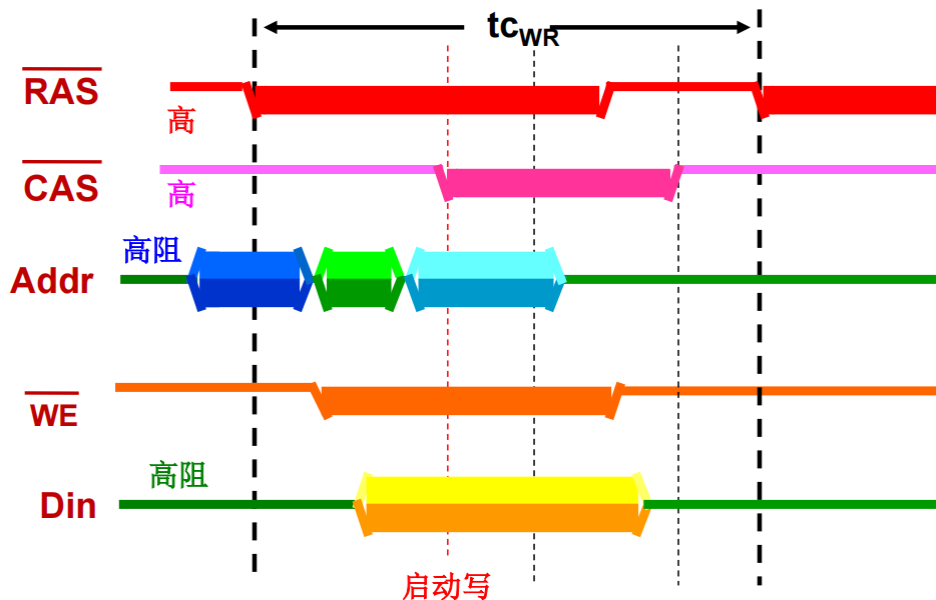
$\text{Addr}$ 处行地址先准备好,  $\text{RAS}$ 下降, 读取行地址, 片选

$\text{CAS}$ 下降,  $\text{RAS}$ 保持片选, 读取 $\text{Addr}$ 处的列地址(已经准备好)

行列地址都有了,  $\text{WE}$ 高电平开始读取数据

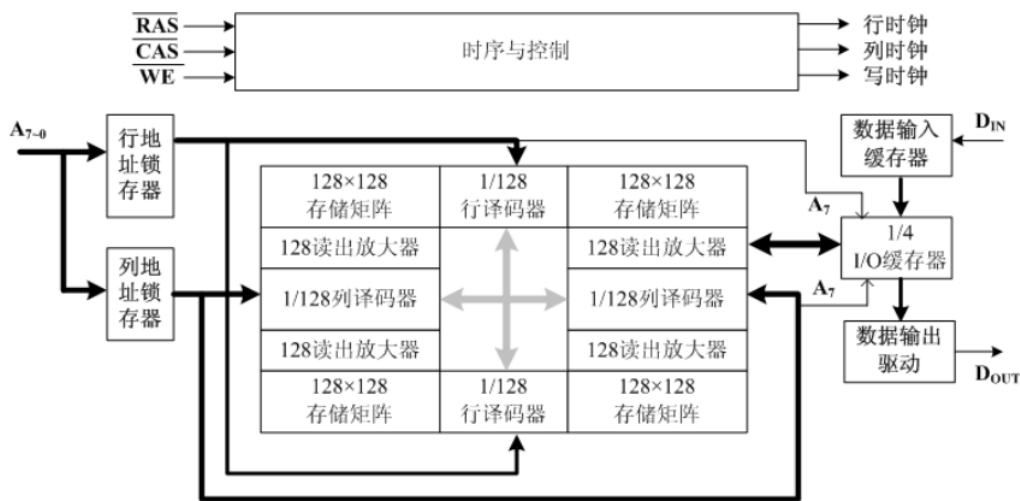
一段时间后 $\text{Dout}$ 将数据输出

### 2.4.5.1. DRAM写周期时序(更慢)



行列地址按照同样方式读取, 与读不同, 在行列地址读取好之前,  $\text{Din}$ 要准备好写入的数据

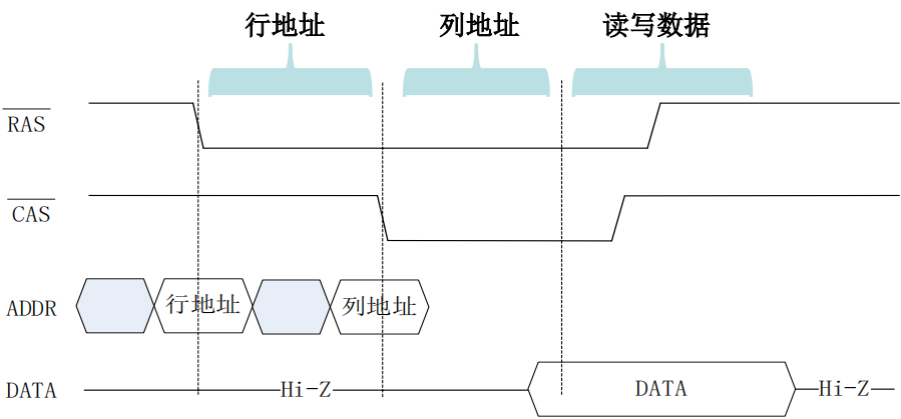
### 2.4.6. DRAM示例：Intel 2164A DRAM(64K×1位)



- 1 行列地址的最高位送IO缓存器，其余位送行/列译码器
- 2 读取数据：行列地址最高位(IO缓存器)决定读哪一块128\*128，然后行列译码器再决定读选中块中的哪个内存

## 2.5. 新型DRAM存储器

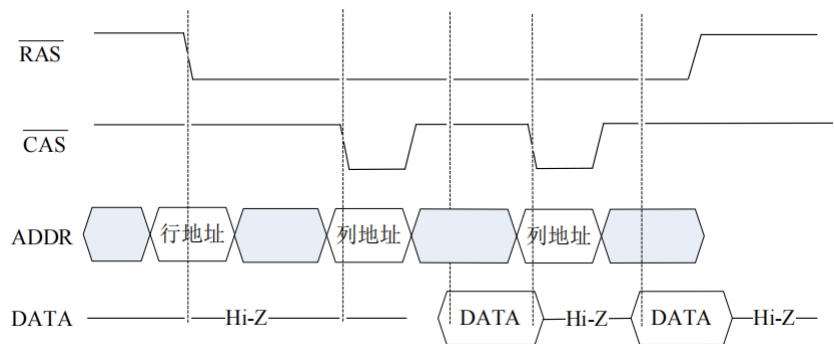
### 1 DRAM



### 2 FPM DRAM：基于局部性原理

大概率：第一次送入行地址/列地址然后保留行地址；第二次只送入列地址然后读取上一次的行地址

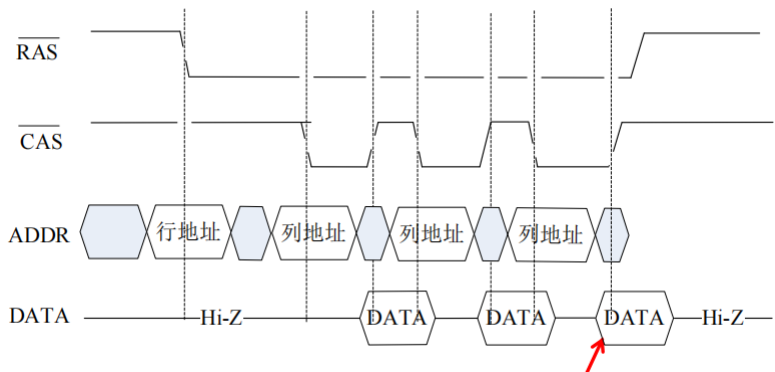
小概率：退化为两次都和DRAM一样



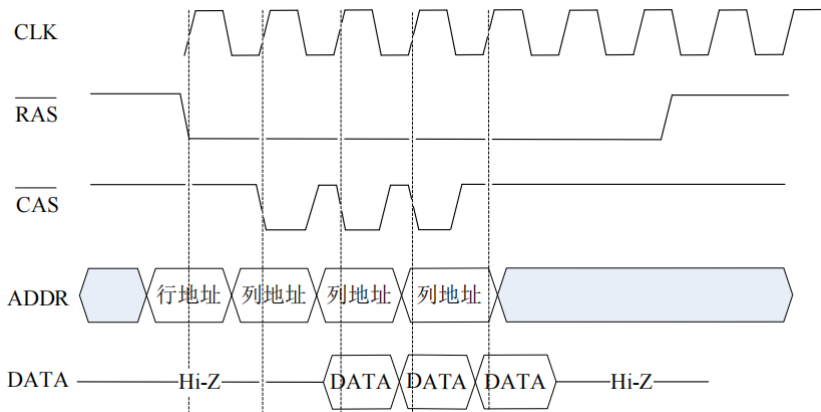
### 3 EDO DRAM：基于FPM的改进

FPM是送完行地址列地址，等待数据，再送新列地址

EDO是送完行地址列地址，等待数据的同时就再送新的列地址



3 SDRAM：加一个控制时钟，同步CPU与主存(送地址和送数据)的操作  
时钟上升沿进行操作(送行/列地址，传数据)



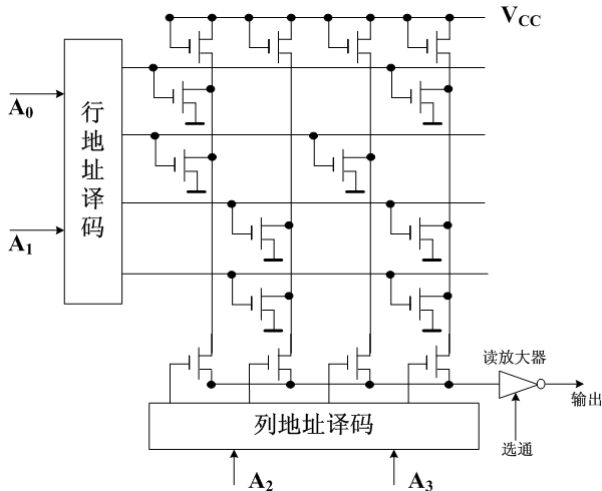
4 DDR：不仅时钟上升沿操作，下降沿也施加一定的操作

## 2.6. 只读存储器/闪速存储器

### 2.6.1. 各种ROM

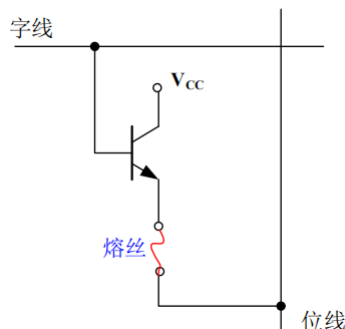
#### 2.6.1.1. Mask ROM(MROM)

生产的时候就将程序刻入ROM中，如下结构中行列地址交叉有MOS管的代表1，交叉无任何东西的代表0，好处是成本低



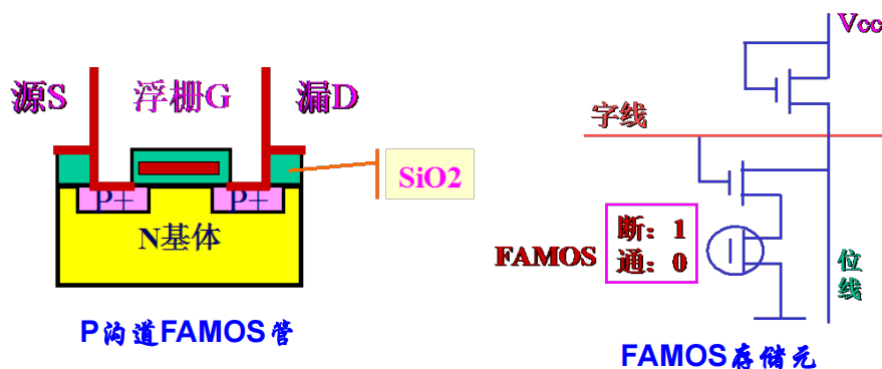
### 2.6.1.2. 可编程ROM(PROM)

分为熔丝/反向二极管式，如图熔丝不断为1断了为0，通过烧断熔丝来一次性编程



(字线为地址线，位线为数据线)

### 2.6.1.3. 可擦除的PROM(EPROM): 基于FAMOS



1 FAMOS: 中间有一个空腔保存电子，当有电子时电流会击穿空腔过去(0)，无电子绝缘过不去(1)

2 通过注入电子(高电压)来编程，但是编程后可擦除(紫外灯)电子

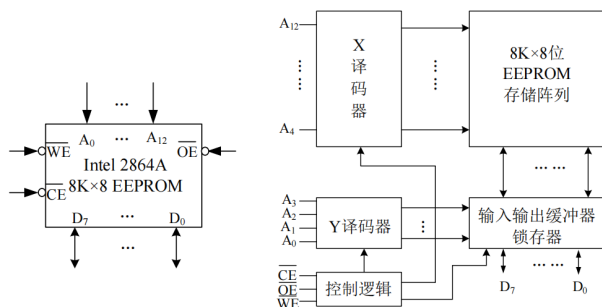
### 2.6.1.4. 可擦除可编程ROM(EEPROM)

1 原理：浮栅上加一个控制栅，直接通过控制栅控制电子的擦写

2 工作方式：正常情况只读，高电压时写入，控制栅接地源极加正高压浮栅上的自由电子越过绝缘层进入源极释放(擦除)

3 主要用于IC卡存储信息

## 2.6.2. EEPROM示例：Intel 2864

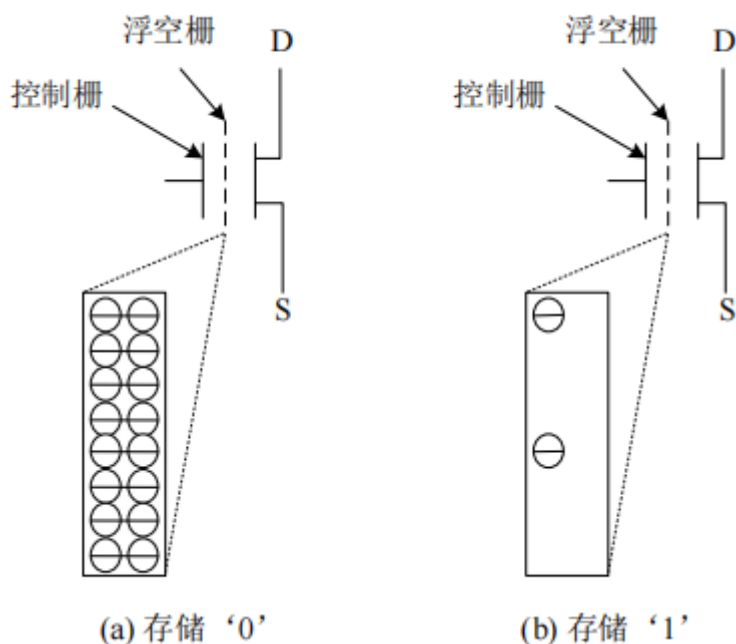


方式	控制引脚			
	$\overline{\text{CE}}$	$\overline{\text{OE}}$	$\overline{\text{WE}}$	I/O <sub>0</sub> ~I/O <sub>7</sub>
读出	L	L	H	数据输出
写入	L	H	L	数据输入
不工作	H	X	X	高阻
禁止写	X	L	X	—
禁止写	X	X	H	—

最右边三个信号分别为：片选，读使能，写使能

## 2.6.3. 闪存存储器(U盘/SSD)

与E2PROM类似，但浮栅的绝缘层更薄。因此擦写速度更快，但耐损耗性(读写次数)稍差



上图其实是Single LC结构浮空栅只能代表0/1

还有注入Triple LC结构，就是浮空栅可以代表000-111一共8个状态，分别对应电子填满0, 1/8,..., 7/8, 1

## 2.7. 存储容量扩展

### 2.7.1. 容量扩展：用多个SRAM芯片构成存储器

#### 2.7.1.1. 位扩展(现在内存条基本都是位扩展)

**1** 适用情况：芯片字数够但位数不够，例如SRAM芯片是1024字x8位而存储器是1024字x16位

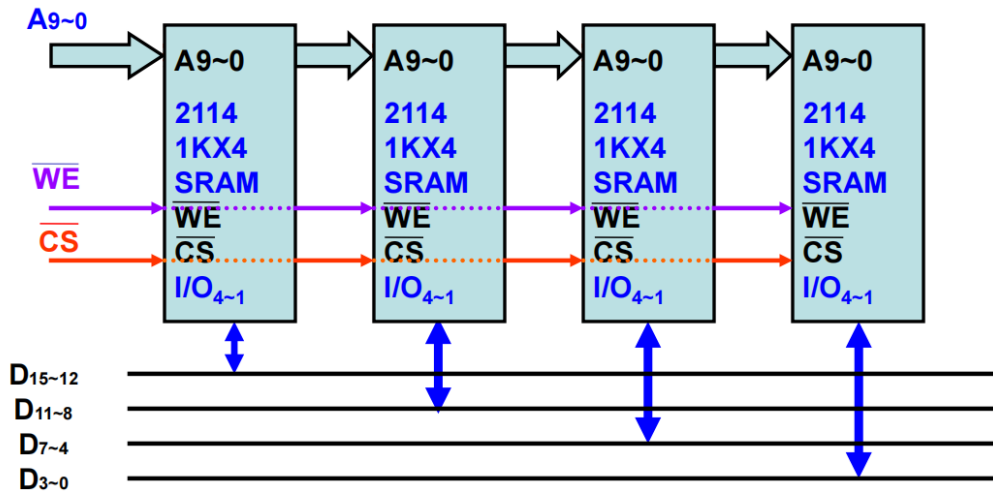
**2** 解决办法：在上面的例子中，可以用两片SRAM并连，第一片SRAM提供存储器地址前8位，第二片提供后8位，两个1024x8 SRAM一起构成1024x16的存储器

PS: 在一般的情况中，芯片用量=存储器字长/芯片字长

**3** 芯片位扩展的排列：水平方向一维排列，别叠加



4 芯片间的连接：Ai(地址)、WE(读/写控制)、CS(片选控制)线同名端并连；Di(数据)线各片按位引出



### 2.7.1.2. 字扩展

1 适用情况：芯片位数够但字数不够，例如SRAM芯片是1024字x8位而存储器是2048字x8位

2 解决办法：在上面的例子中，可以用两片芯片串联，然后：

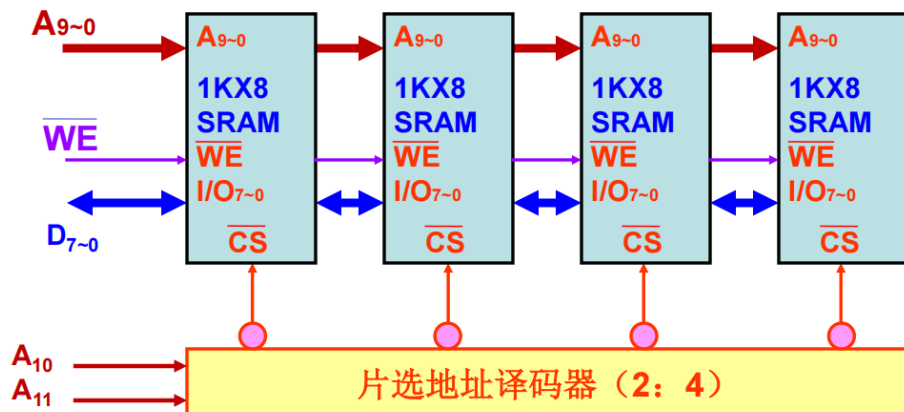
1. 访问第一片SRAM的地址时(范围0-1023)，第二片SRAM保持不活跃。
2. 访问第二片SRAM的地址时(范围1024-2047)，第一片SRAM保持不活跃。

PS: 在一般的情况下，芯片用量=存储器字/芯片字

3 芯片字扩展的排列：竖直(当然也可以水平)方向一维排列，别叠加

4 片间连接：

1. WE(读/写控制线)、Di(数据)线同名端并连
2. Ai(地址)线分两部分：低位地址部分与各芯片同名端并连；设片选地址译码器；高位地址部分接片选地址译码器输入端
3. CS(片选控制)线：各片的片选线分别接片选地址译码器的各输出端



在这个示例中，地址线12位，每个SRAM芯片10位，地址的高两位被用来选择哪个芯片保持活跃，地址的低10位用来选择芯片中的地址

### 2.7.1.3. 字位扩展

1 适用情况：芯片位数/字数都不够，例如SRAM芯片是1024字×4位而存储器是2048字×8位

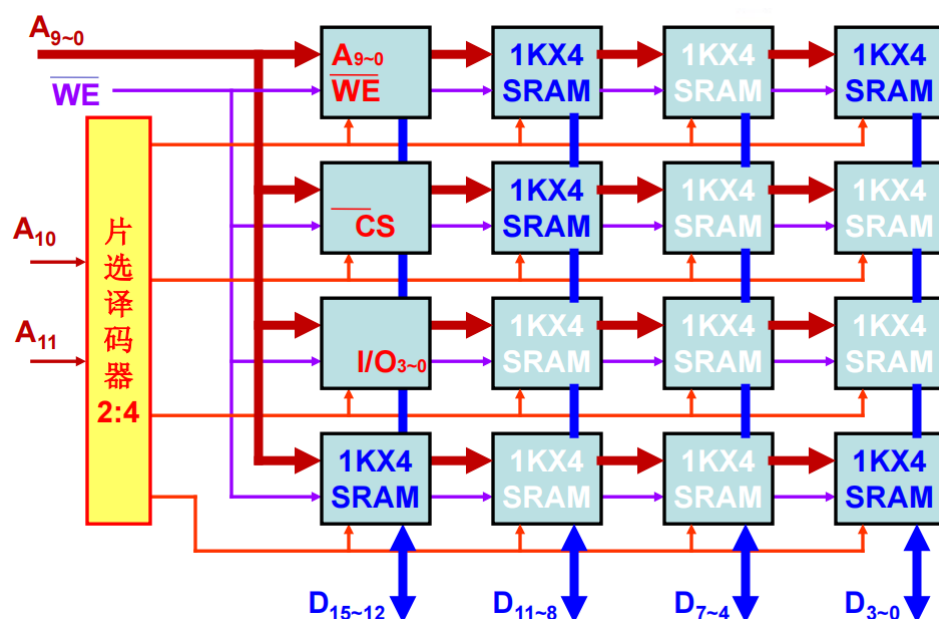
芯片用量=存储器容量/芯片容量

2 芯片排列：二位平面矩阵

3 片间连接

1. WE(读/写控制)线同名端并连；Di(数据)线沿位向同名端并连引出；
2. Ai(地址)线仍分两部分：低位地址部分与各芯片同名端并连；设片选地址译码器；高位地址部分接片选地址译码器输入端；
3. CS(片选控制)线沿字向同名端并连引出，分别接片选地址译码器的各输出端

Di/CS线引出方向垂直



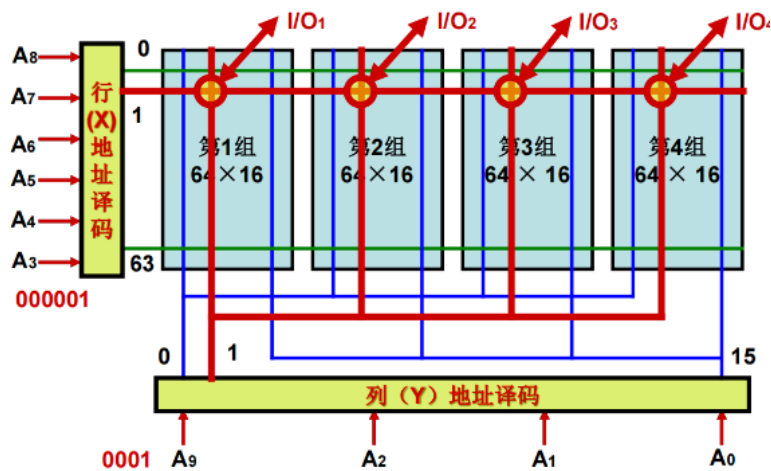
横向是位扩展(每列芯片提供4位一共16位)，纵向是字扩展(译码器选择哪一列芯片，低地址选择内存)

### 2.7.2. DRAM容量扩展

1 DRAM字/位扩展与SRAM一样，但地址线和片选信号的连接需要作特殊处理：

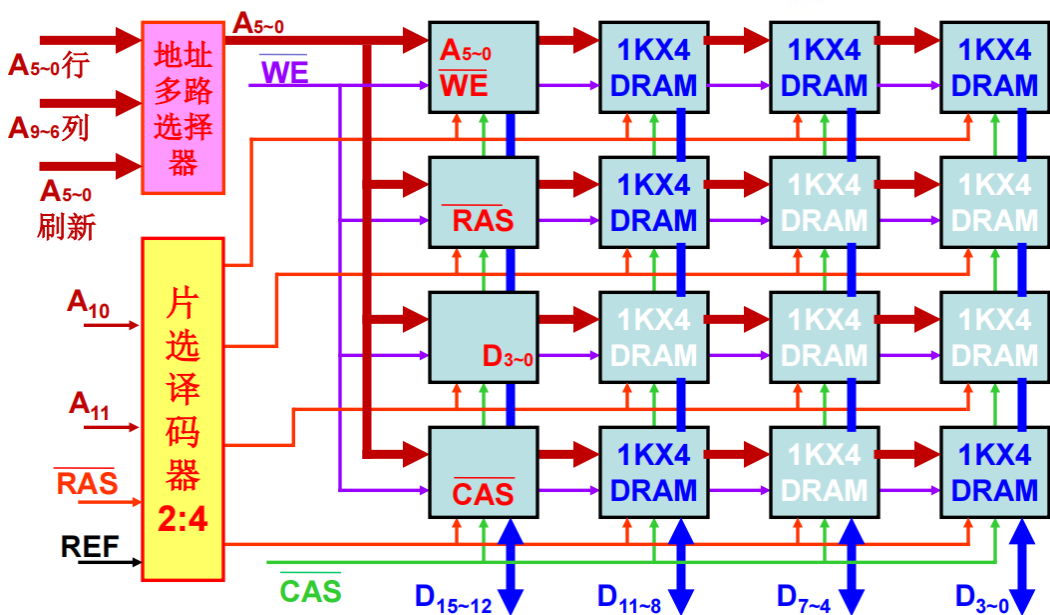
1. Ai(地址)线需分行/列送各芯片地址端；
2. 片选译码器输入需加 RAS 信号，以使片选译码器输出的时间与 RAS 一致；
3. 片选译码器输入还需加刷新定时信号(REF)，以便在刷新时，强制同时刷新所有芯片内部存储矩阵
4. 采用唯RAS有效刷新时，CAS可简单地按片并联

2 示例：1K×4 DRAM组成4K×16位存储器(芯片内部64×64存储矩阵)，结构如图所示



行地址6位，列地址4位

### 3 片间连接



与SRAM的主要区别：

1. 需要一个多路选择器将CPU发出的地址一分为二为行地址/列地址
2. 有关刷新：
  - 特殊2/4译码器中当REF有效时四条输出都会有效从而刷新所有DRAM
  - 地址多路选择器中A0-5的刷新信号决定了每一个DRAM是刷新64行中的哪一行

## 2.7.3. ROM容量扩展

处理ROM不需要写引脚WE，其他都一样。就是SRAM字位扩展那个图去掉WE

## 2.8. 存储器与CPU的连接：总线

### 2.8.1. SRAM连接方法

- 1 地址线：CPU通过地址总线向存储器发送地址，地址总线低位直连存储芯片地址引脚，高位连接片选译码器输入端
- 2 数据线：CPU通过数据总线与存储器交换数据，存储器数据引出线与数据总线按位连通

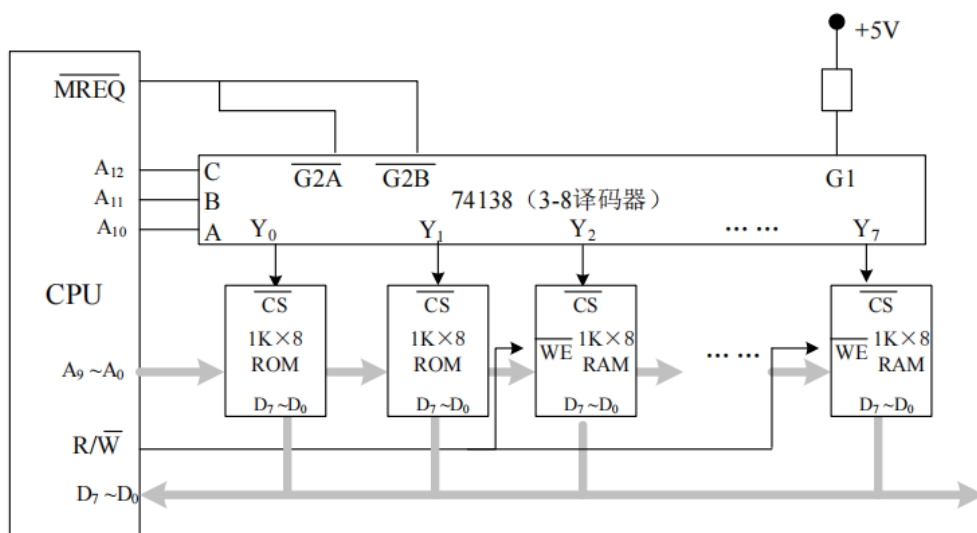
3 读/写控制线：CPU通过控制总线中的信号线向存储器发读/写令，存储器WE线与控制总线中的读/写命令线连通即可

4 片选时间控制：CPU控制总线中的MREQ(访存请求)信号与片选译码器使能输入端相连即可

PS: MREQ有效表示访问存储器(只有他低电平才可以访问存储器)，IOREQ有效表示访问IO设备

## 2.8.2. ROM连接方式

与SRAM基本相同，大门时读写控制线不连，无WE端

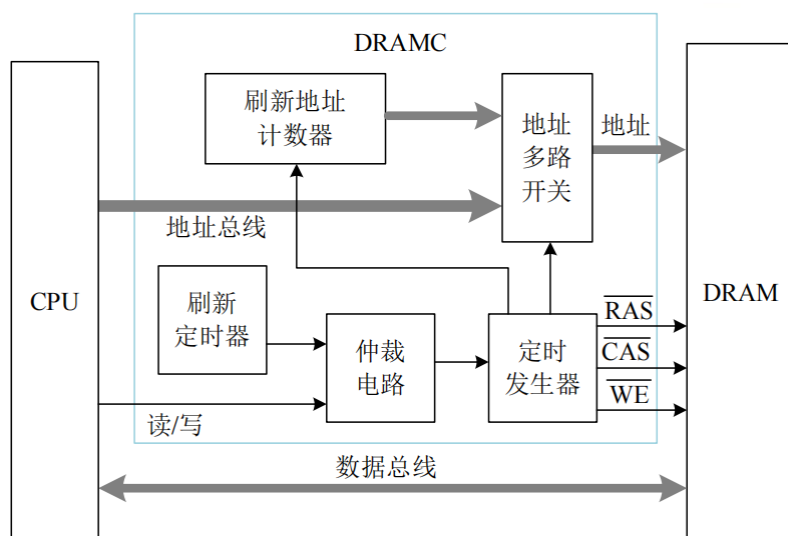


1 这里有三个使能端，左边两个低电平有效，右边一个高电平有效，可以起到译码作用

2 左边两个为ROM不用连读写信号，右边两个为RAM所以要连


## 2.8.3. DRAM连接方式

1 概述：DRAM不直连CPU总线，而通过DRAMC与CPU总线连接。CPU送一次地址，DRAMC分两次送给DRAM。DRAMC定期刷新DRAM



2 DRAMC结构

1. 刷新地址计数器：记住这次刷新的行，然后下一次让DRAM刷新下一行
2. 地址多路选择器：选行/列/刷新地址之一送给DRAM

- 
3. 刷新定时器：定时给DRAM信号，让DRAM执行刷新
  4. 仲裁电路：当CPU/IO要来访存，同时DRAM有需要刷新了，执行二选一操作
  5. 定时发生器：产生各种控制信号如RAS/CAS/WE 等