

# 计算机组成原理

## DRAM存储器容量扩充、存储器接口设计

王浩宇,教授

haoyuwang@hust.edu.cn

<https://howiepku.github.io/>

# 本节目录

- 存储器容量扩充
- 高级DRAM结构
- 只读存储器ROM
- 存储器接口设计
- 并行存储器

# 存储器的容量扩展

- 存储器芯片的容量是有限的，为了满足实际存储器的容量要求，需要对存储器进行扩展
- 扩展方法
  - 位扩展
    - 只加大字长，而存储器的字数与存储器芯片字数一致
  - 字扩展
    - 给定的芯片存储容量较小（字数少），需要用多个芯片来扩展字数，而位数不变
  - 字位同时扩展
    - 同时扩展位数与字数
    - 例如，一个存储器容量为 $M \times N$ 位，若使用 $X \times Y$ 位的芯片（ $X < M$ ,  $Y < N$ ），需要在字和位同时进行扩展。
      - 需要  $(M/X) * (N/Y)$  个存储器芯片

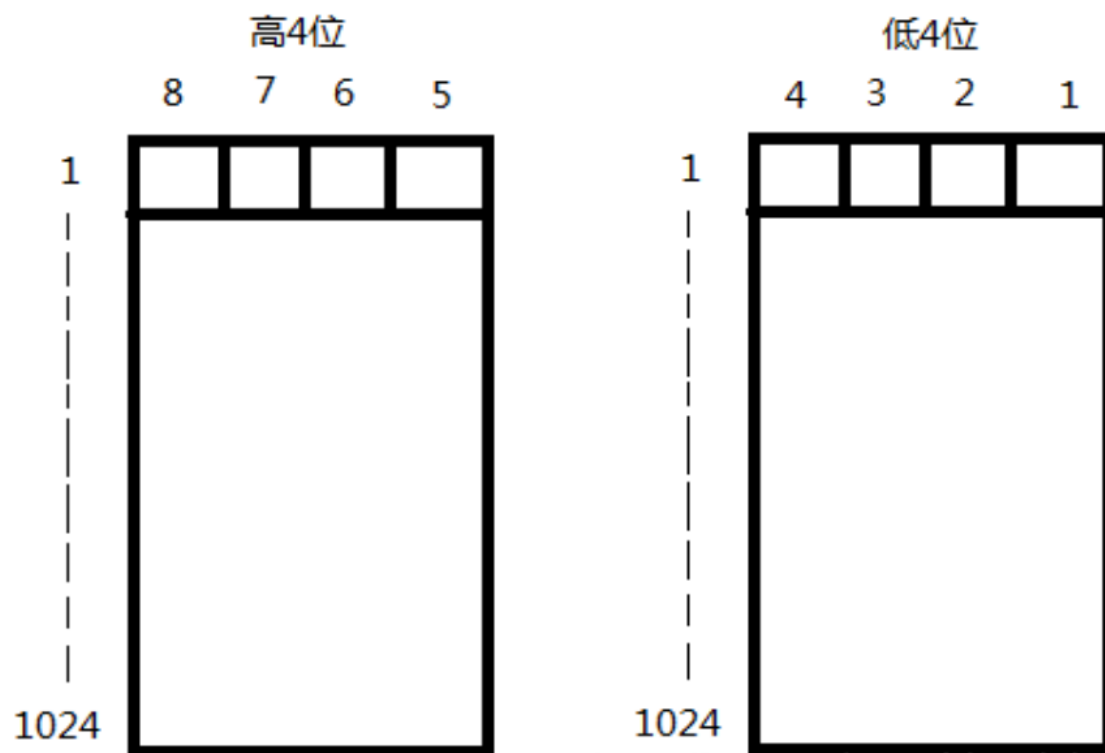
# 字长位数扩展

## ■ 字长位数扩展

- 给定的芯片字长位数较短，不满足设计要求的存储器字长，此时需要用多片给定芯片扩展字长位数
- 三组信号线中，地址线和控制线公用，数据线单独分开连接
- 需要一个片选信号同时选中这些芯片

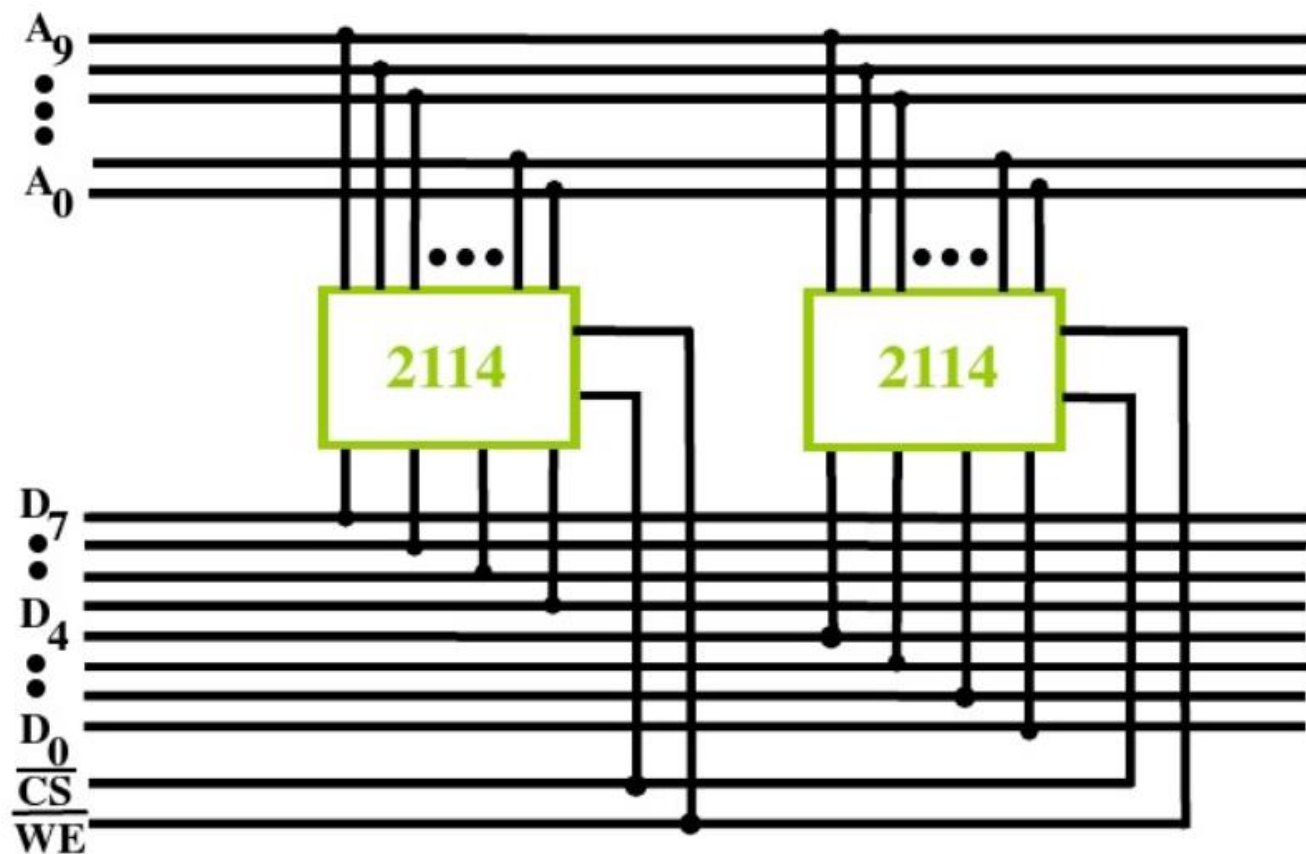
# 字长位数扩展

- 假设现在有1K (1024个) \*4位的存储芯片 (容量为4096bit) 若干, 要想构成一个1K\*8位的存储器
- 我们可以使用两片1K\*4位的存储芯片来构成



# 字长位数扩展

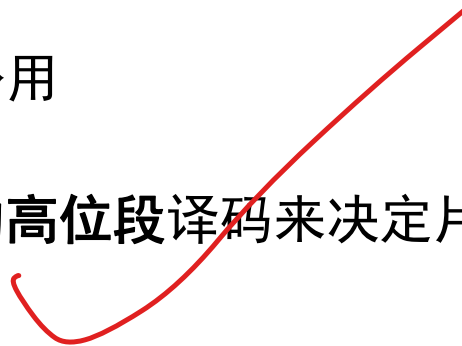
- 通过片选信号CS同时选中两片存储芯片，同时进行8位数据的读出和写入
  - 10根地址线，8根数据线



# 字长位数扩展

- 位扩展的关键就是将两个存储芯片当成一个存储芯片来用
  - 让两个存储芯片同时工作，同时被选中，同时做读操作，同时做写操作，要想保证同时，就是把两个芯片的片选，用相同的信号进行连接。
- 所需芯片数量= 设计要求的存储器容量/选择芯片存储器容量
- 例如：利用 $1\text{M} \times 4$ 位的SRAM芯片，设计一个存储容量为 $1\text{M} \times 8$ 位的SRAM存储器。
  - 所需芯片数量=  $(1\text{M} \times 8) / (1\text{M} \times 4) = 2$ 片

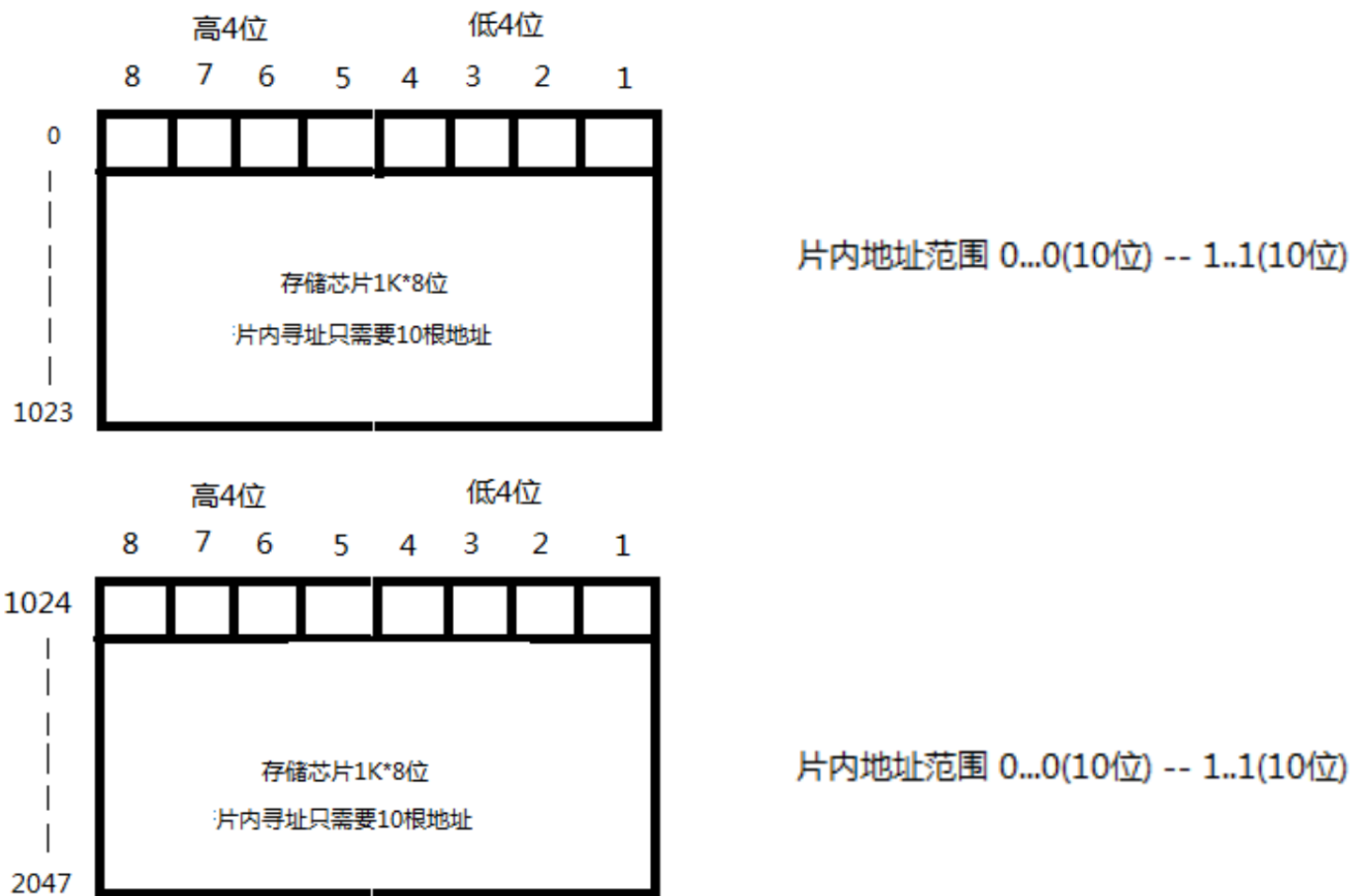
# 字存储容量扩展

- 给定的芯片存储容量较小（字的数量少），不满足设计要求的总存储容量，此时需要用多片给定芯片来扩展字数
  - 三组信号线中
    - 给定芯片的地址总线和数据总线公用
    - 控制总线中R/W公用
    - 使能端不能公用，它由地址总线的高位段译码来决定片选信号
- 



# 字存储容量扩展

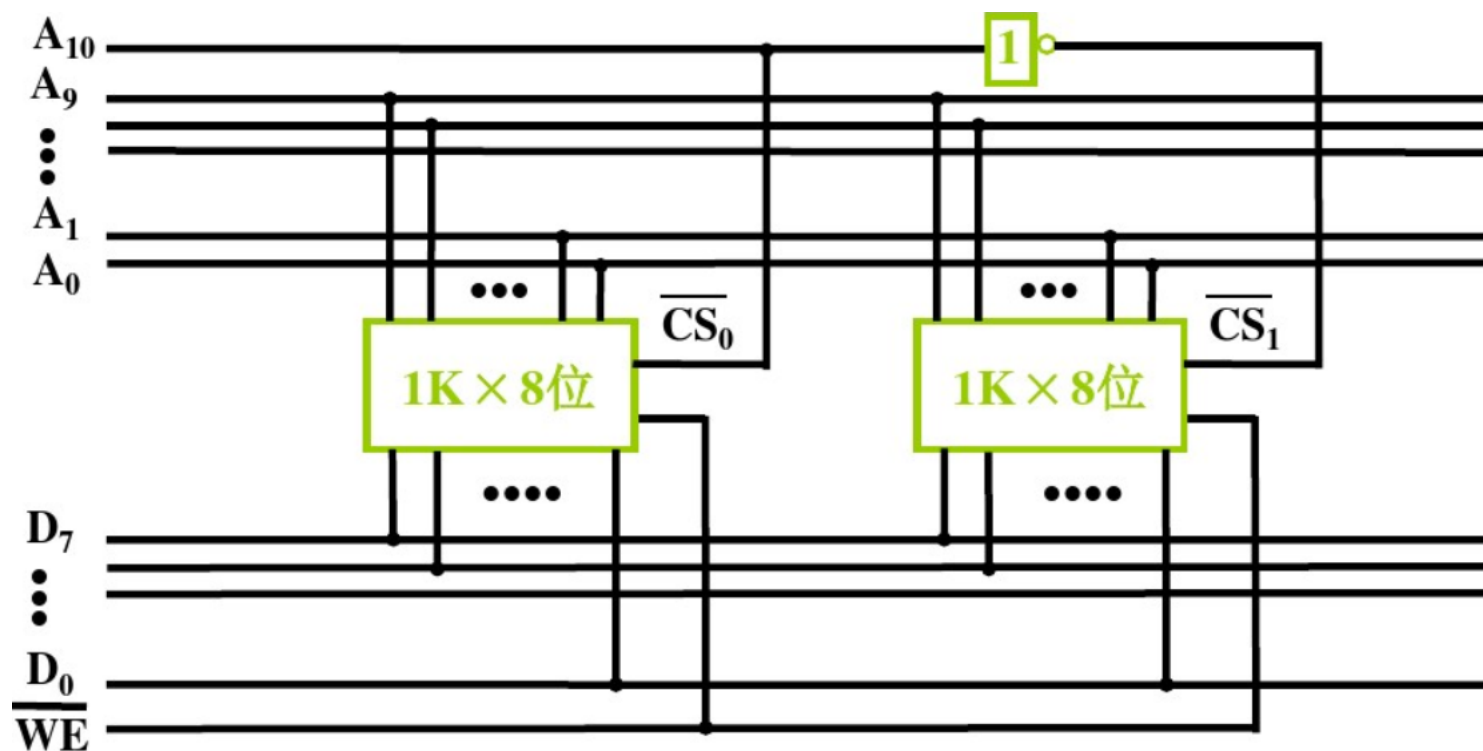
- 1K\*8位的存储芯片，构成一个2K\*8位的存储器



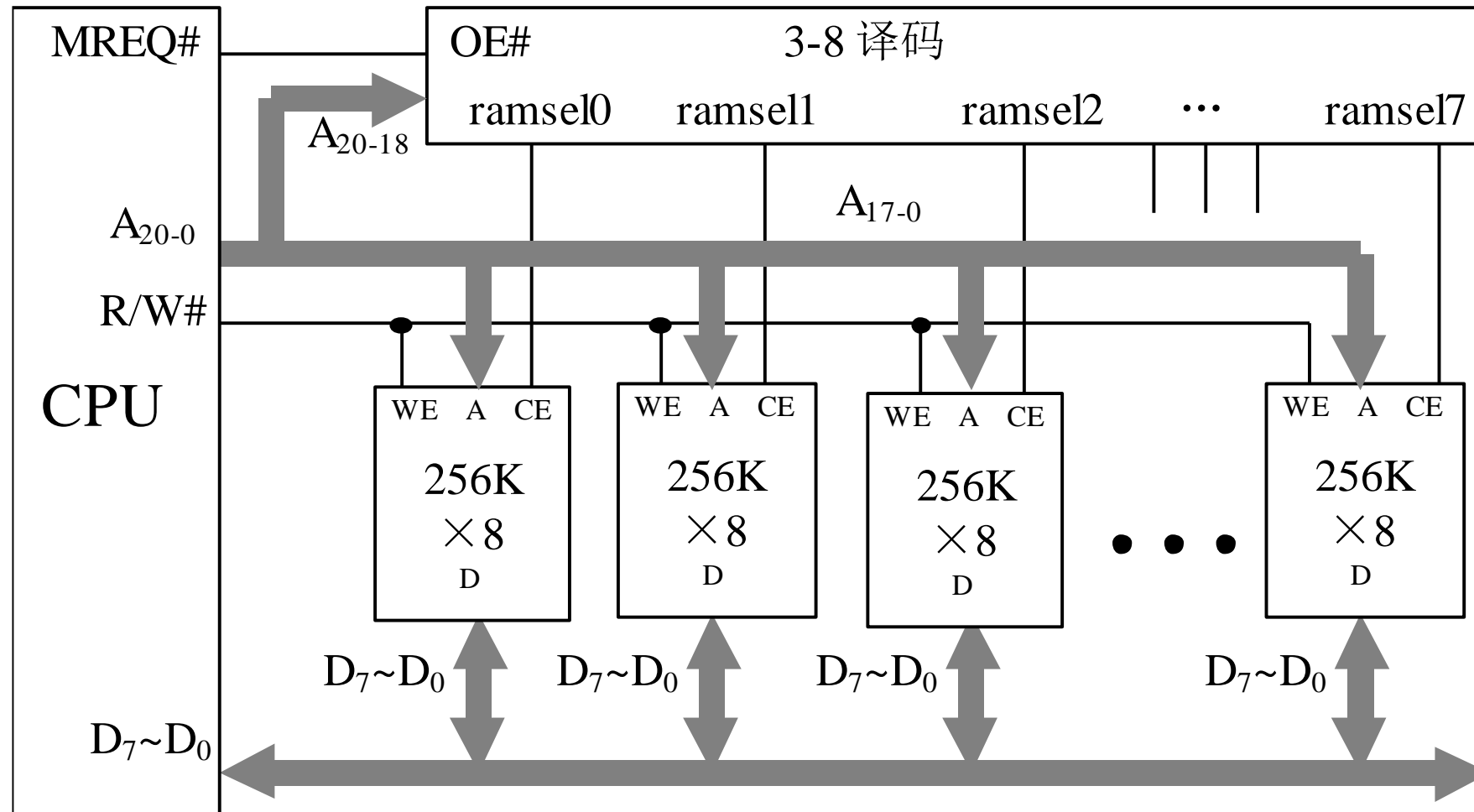
# 字存储容量扩展

## ■ 1K\*8位的存储芯片，构成一个2K\*8位的存储器

- 2K有11位二进制位，而存储芯片的每一片的容量是1K(10位)
- 多出的一位用于做片选信号
- 两个芯片不会同时工作



例：设有若干片 $256\text{K} \times 8$ 位的SRAM芯片，问：采用**字扩展方法**构成 $2048\text{KB}$ 的存储器需要多少芯片？该存储器需要多少地址位？



# 字存储容量扩展

- 所需芯片数仍

$d = \text{设计要求的存储器容量} / \text{选择芯片存储器容量}$

- 利用  $1\text{M} \times 8$  位的 DRAM 芯片设计  $2\text{M} \times 8$  位的 DRAM 存储器，所需芯片数

$$d = (2\text{M} \times 8) / (1\text{M} \times 8) = 2 (\text{片})$$

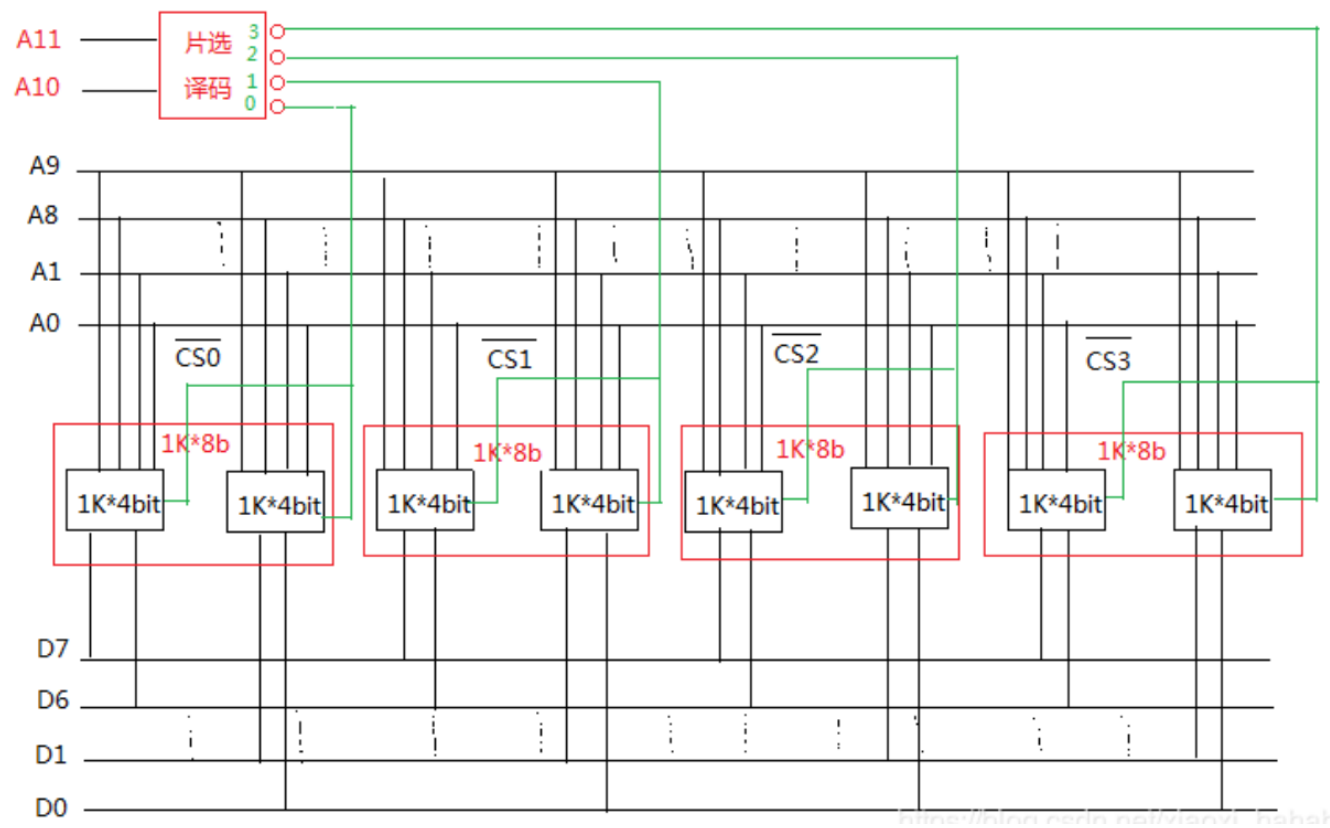
# 字位扩展结合

- 同时扩展位数与字数
- 例如，一个存储器容量为 $M*N$ 位，若使用 $X*Y$ 位的芯片（ $X < M$ ,  $Y < N$ ），需要在字和位同时进行扩展
  - 需要  $(M/X) * (N/Y)$  个存储器芯片

# 字位扩展结合

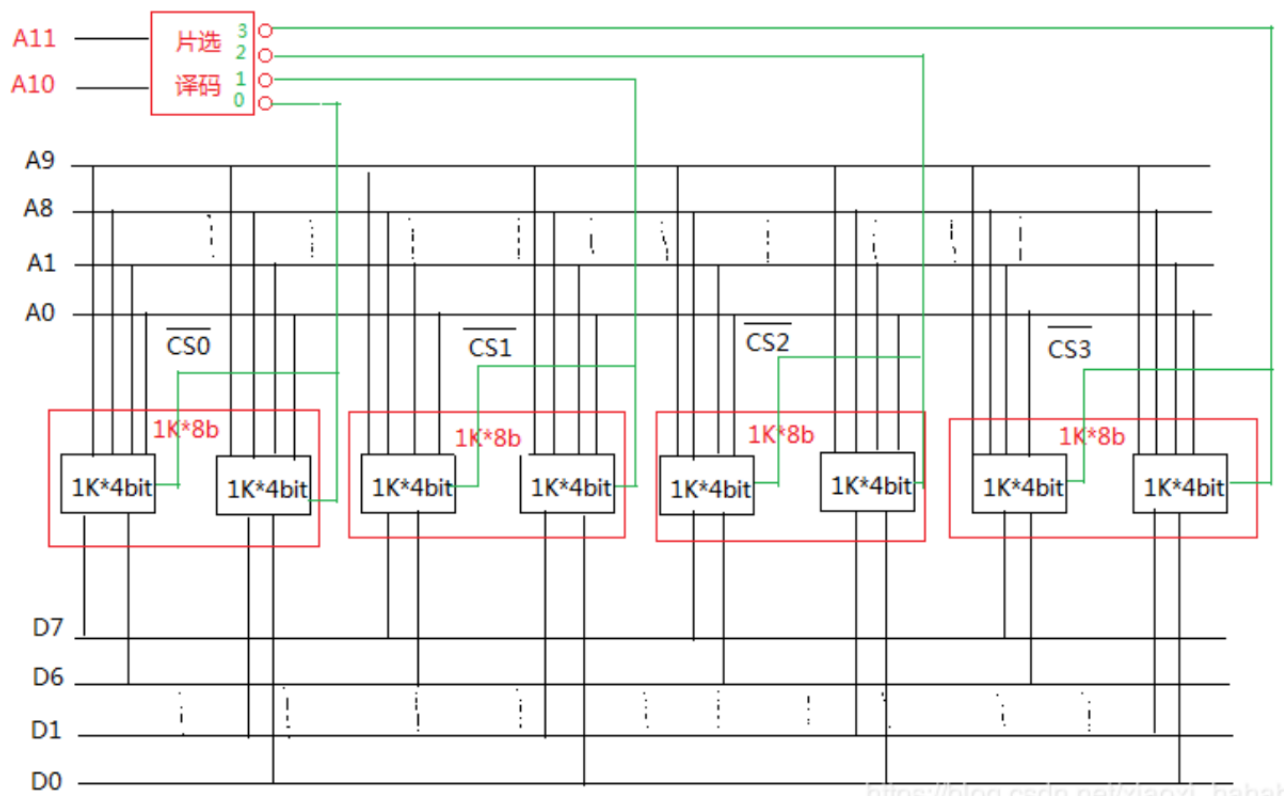
- 假设1K\*4位的存储芯片若干，组成4K\*8位的存储器
- **可用八片1K\*4位的存储芯片来构成**
  - 两片来构成1K\*8bit，4组来构成4K\*8bit的存储器
- **如何连接？**
  - 4K\*8的存储器有12根地址线和8根数据线
  - A0-A9用于1K\*8存储器寻址，A10-11用于做片选信号

# 字位扩展结合



- 4K的空间分配到4个小存储器，每个小存储器含两片1K\*4bit芯片
  - 第一个小存储器的范围为 00 0...0(10个0)~00 1...1(10个1)
  - 第二个小存储器的范围为 01 0...0(10个0)~01 1...1(10个1)
  - 第三个小存储器的范围为 10 0...0(10个0)~10 1...1(10个1)
  - 第四个小存储器的范围为 11 0...0(10个0)~11 1...1(10个1)

# 字位扩展结合



- A11和A10作为2-4译码器的输入
  - A11 = 0   A10=0 选择第一个存储器
  - A11 = 0   A10=1 选择第二个存储器
  - A11 = 1   A10=0 选择第三个存储器
  - A11 = 1   A10=1 选择第四个存储器



# 本节目录

- 存储器容量扩充
- 高级DRAM结构
- 只读存储器ROM
- 存储器接口设计
- 并行存储器

# 高级DRAM结构

- 主存的基本构件为DRAM芯片
- 如何解决DRAM主存性能？
  - 一种方法是在主存和CPU之间插入一级或者多级SRAM组成的高速缓冲存储器cache
- 但SRAM比DRAM贵的多
  - 扩展cache超过一定限度时，将得不偿失
- 对基本DRAM的增强手段
  - 提高时钟频率和带宽
  - 缩短存取周期

## (1) FPM-DRAM 快速页模式动态存储器

- 回顾：DRAM读周期和写周期中，如何寻找一个确定的存储单元地址？
  - 首先由低电平的行选通信号RAS确定行地址，然后由低电平的列选信号CAS确定列地址
  - 下一次寻找操作，也是由RAS选定行地址，CAS选定列地址，依此类推

## (1) FPM-DRAM 快速页模式动态存储器

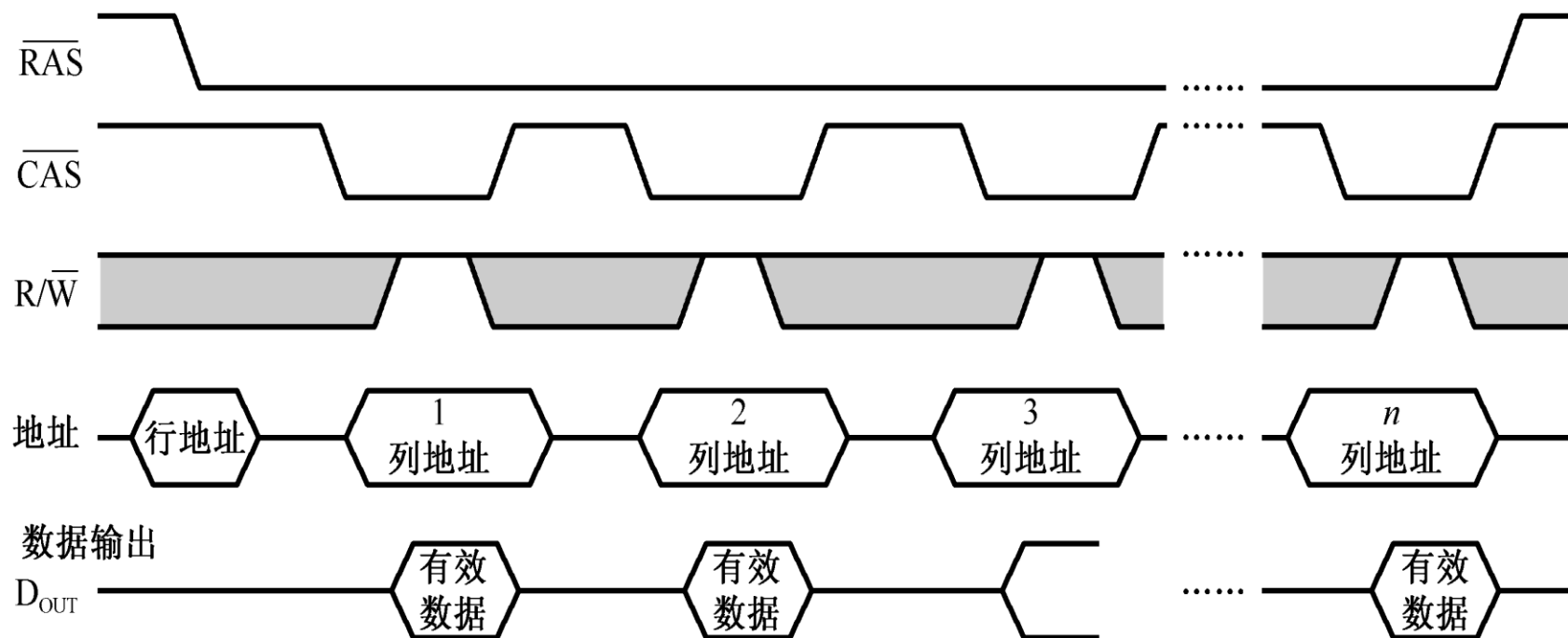
### ■ 根据程序的局部性原理实现

### ■ 程序访问局部性原理

- 根据计算机中对大量典型程序运行情况的结果，当前要立即执行的程序和数据往往局限在一个小的范围内
- 在一段时间间隔内，CPU对局部范围的存储器进行频繁访问，而对此外的地址很少访问。这种现象称为程序访问的局部性
- 时间局部性、空间局部性

## (1) FPM-DRAM 快速页模式动态存储器

- **分页技术：保持行地址不变，只改变列地址**，对同一行的所有内存单元进行访问
- 经过一个快速页周期后，根据读写命令R/W信号，该页中的所有存储单元都进行了读/写操作

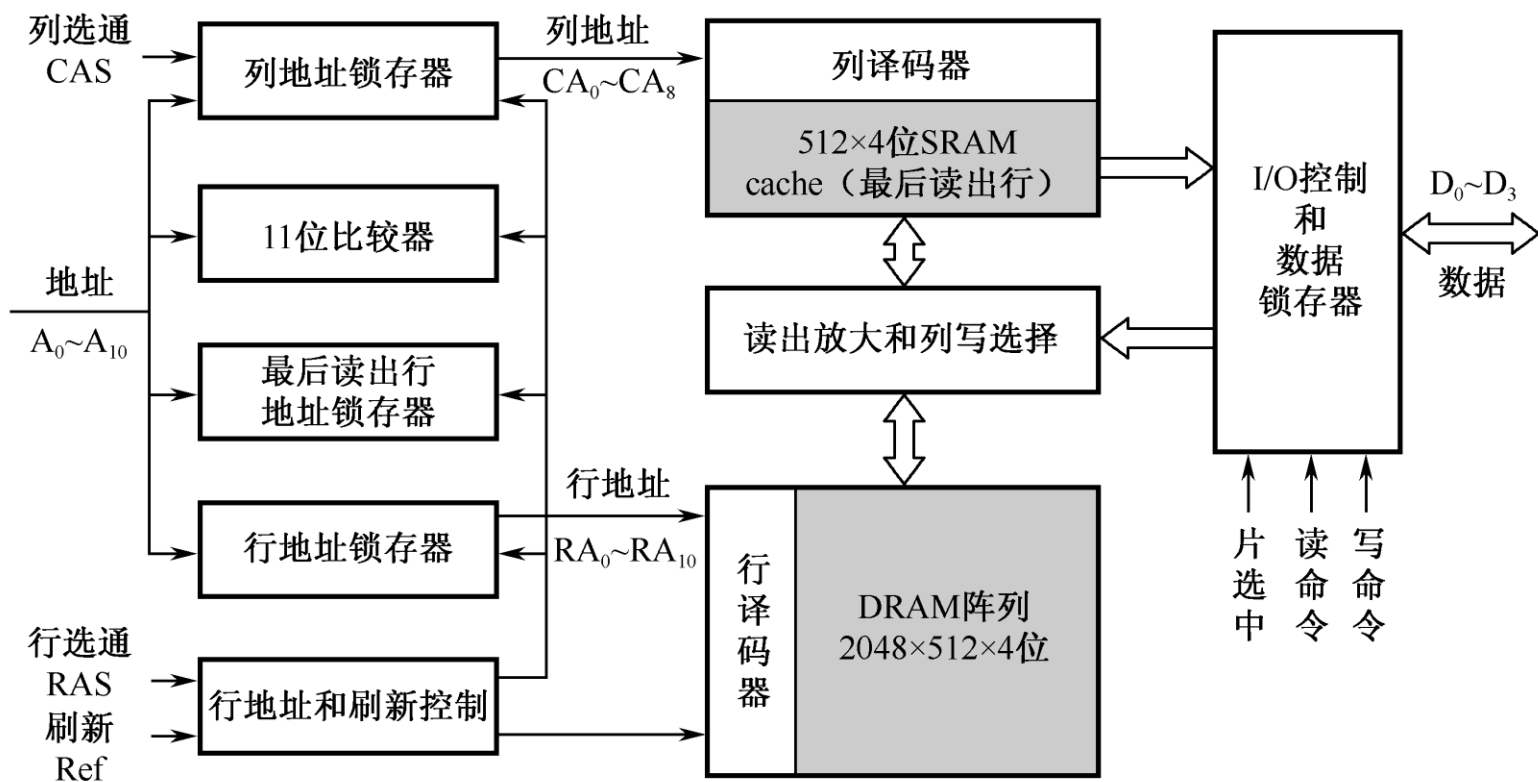


## (2) CDRAM存储器

- 带高速缓冲存储器（cache）的动态存储器
- 在DRAM芯片内集成了一个小容量的SRAM，从而使DRAM芯片的性能得到显著改进

## (2) CDRAM存储器

- 1M×4位CDRAM芯片的结构框图，其中SRAM为512×4位，保存最后读出行的数据
- SRAM是DRAM的一个小副本



## (2) CDRAM存储器

### ■ SRAM保存最后读出行的数据

### ■ 下一次读取

- 输入行地址立即与最后读出行锁寄存器的内容进行11位比较
- 若比较相符，则SRAM命中，由输入的列地址从SRAM中选择某一位组送出
- 如比较不相符，则需要驱动DRAM阵列，更新SRAM和最后读出行地址锁寄存器的内容



## (2) CDRAM存储器

### ■ 猝发式读取

- SRAM保存一行内容的方法，对成块传送非常有利
- 如果连续的地址高11位相同，意味着属于同一行地址，那么连续变动的9位列地址就会使SRAM中相应位组连续读出，成为猝发式读取
- 猝发式访问：在对同一行的连续单元进行访问时，减少额外的延迟和等待周期

### ■ CDRAM结构的另外两个优点

- SRAM读出期间可以同时DRAM阵列进行刷新
- 芯片内的数据输出路径(SRAM-IO)与数据输入路径(IO到列写选择和读出放大器)是分开的，允许在写操作完成的同时来启动同一行的读操作

### (3) SDRAM存储器

#### ■ SDRAM同步型动态存储器

- 计算机系统CPU使用的是系统时钟，SDRAM的操作要求与系统时钟相同步，在系统时钟控制下从CPU获得地址、数据和控制信息
- 它与CPU的数据交换同步于外部的系统时钟信号，并且以CPU/存储器总线的最高速度运行，而不需要插入等待状态
- SDRAM支持与系统同步的连续单元猝发式访问

# 本节目录

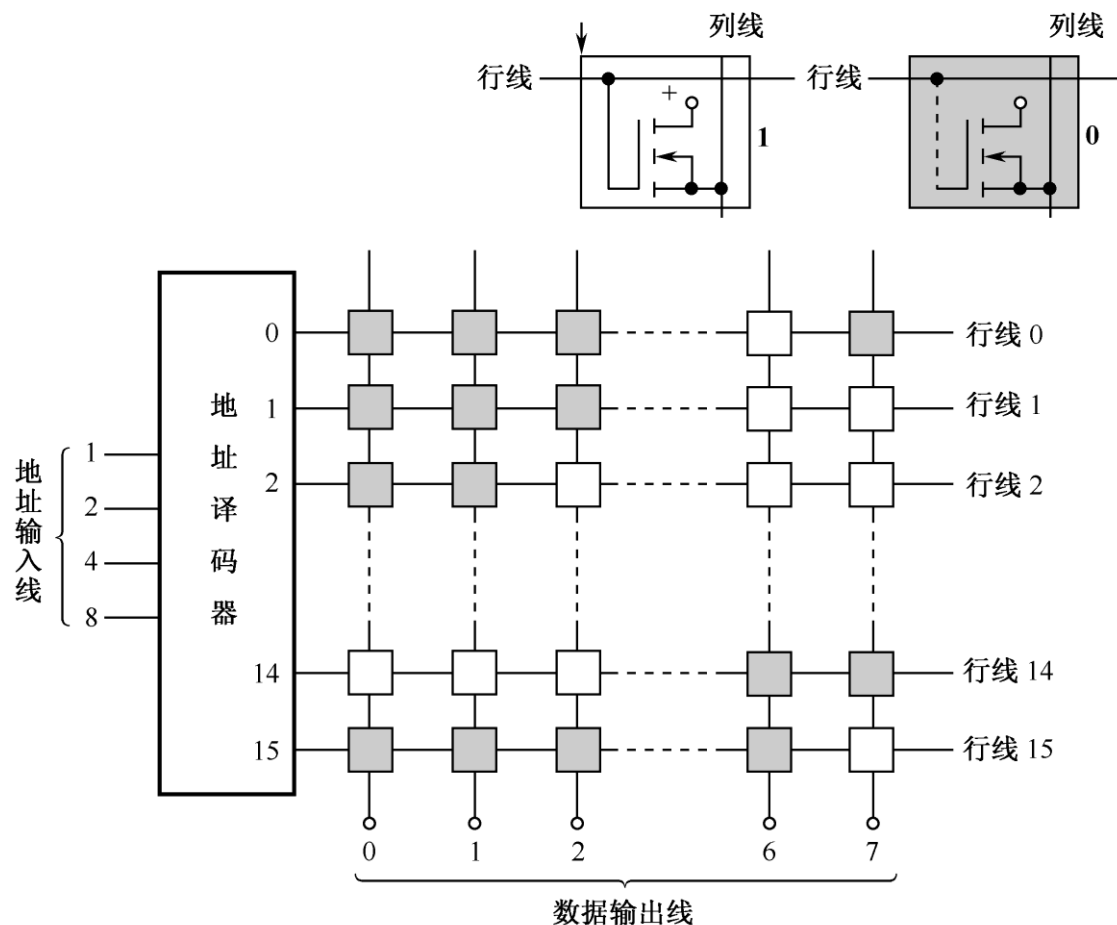
- 存储器容量扩充
- 高级DRAM结构
- 只读存储器ROM
- 存储器接口设计
- 并行存储器

# 只读存储器ROM

- SRAM和DRAM都是随机读写存储器，数据可读可写
- ROM只读存储器
  - 在它工作时只能读出，不能写入。
  - 然而其中存储的原始数据，必须在它工作以前写入只读存储器由于工作可靠，保密性强，在计算机系统中得到广泛的应用。
- 主要有两类：
  - 掩模ROM：掩模ROM实际上是一个存储内容固定的ROM，由生产厂家提供产品。
  - 可编程ROM：用户后写入内容，有些可多次写入
    - 一次性编程的PROM
    - 多次编程的EPROM和E<sup>2</sup>PROM。

# 掩模ROM只读存储器

- 存储内容固定，由生产厂家提供产品
- 一旦ROM做成，就不能改变其中的存储内容



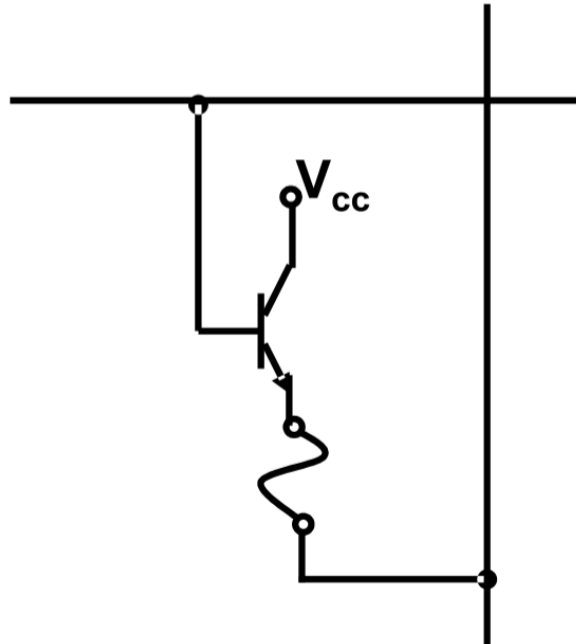
大部分ROM利用在行选线和列选线交叉点上的晶体管是导通或者截止来表示存1或0

# 可编程ROM

- 可编程ROM由PROM， EPROM和E<sup>2</sup>PROM三种
- PROM是一次性编程
- EPROM叫做光擦除可编程可读存储器。它的存储内容可以根据需要写入，当需要更新时将原存储内容抹去，再写入新的内容，解决了PROM芯片只能写入一次的弊端
- EEPROM（电可擦写可编程只读存储器）是可用户更改的只读存储器（ROM），其可通过高于普通电压的作用来擦除和重编程（重写）。不像EPROM芯片，EEPROM不需从计算机中取出即可修改

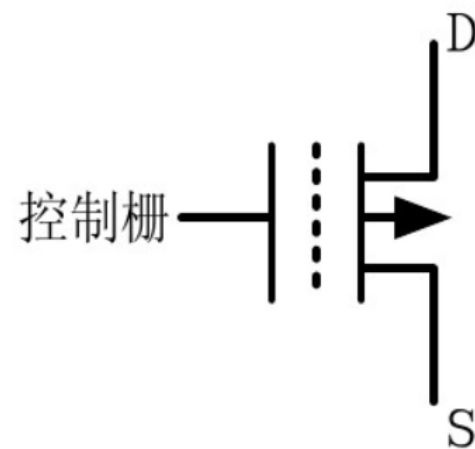
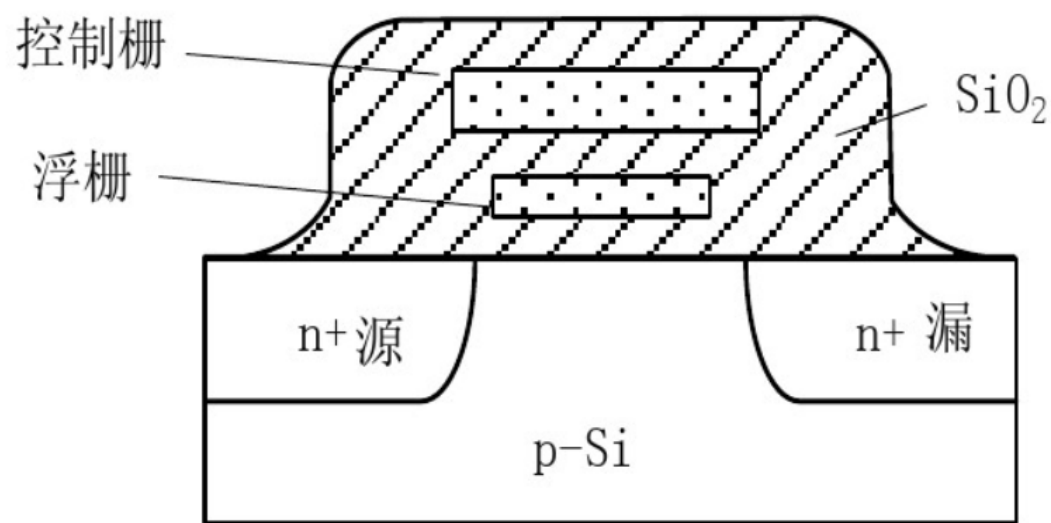
# PROM

- PROM是可以实现一次性编程的只读存储器
- 该图是一个由双极型电路和熔丝构成的基本单元电路
  - 写“0”：烧断熔丝
  - 写“1”：不烧断熔丝



# EPR0M

- EPR0M的信息可重复擦写
- 通过编程器把信息写入存储器，该信息可长久保持，可作为只读存储器
- 当内容需要改变时，利用擦抹器（如紫外光源）将其擦除，再重新写入信息
- 目前用得较多的EPR0M的存储单元由浮动栅雪崩注入型MOS管构成





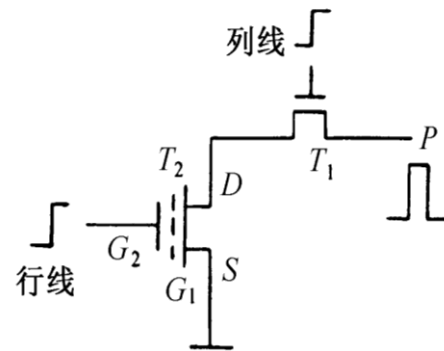
# EPROM

## ■ 这种EPROM出厂时为全“1”状态

- 使用者可根据需要写“0”

## ■ 写“0”电路

- P端加20多伏的正脉冲，脉冲宽度为0.1~1ms
- EPROM允许多次重写
- 抹去时，用40W紫外灯，相距2cm，照射几分钟即可



(f) 写0时电路

# EEPROM

## ■ 电可擦除只读存储器EEPROM

- 在EPROM基础上开发，可以在加电的情况下擦除存储器的全部或某一部分内容，然后在电路上直接改写其擦除过的单元内容

## ■ 基本原理

- EEPROM的内部电路与EPROM电路类似，但其结构进行了一些调整，在浮栅上增加了一个隧道二极管，在编程时可以使电荷通过它流向浮栅，而擦除时可使电荷通过它流向漏极，不需要紫外光激发放电，即擦除和编程只须加电就可完成，且写入的电流很小。

## ■ EPROM擦除时，将整个芯片全部信息都擦去；EEPROM可只擦除部分信息

---

# 本节目录

- 存储器容量扩充
- 高级DRAM结构（简单讲述，不要求掌握）
- 只读存储器ROM
- 存储器接口设计
- 并行存储器（简单讲述，不要求掌握）

# 存储器接口设计举例

## ■ 接口基本步骤

- 选择芯片（ROM, RAM）
- 地址分配
- 真值表（芯片范围）
- 写出逻辑表达式
- 给出电路

例 某计算机的主存地址空间中，从地址 $0000_{16}$ 到 $3FFF_{16}$ 为ROM存储区域，从 $4000_{16}$ 到 $5FFF_{16}$ 为保留地址区域，暂时不用，从 $6000_{16}$ 到 $FFFF_{16}$ 为RAM地址区域。RAM的控制信号为CS#和WE#，CPU的地址线为A15~A0，数据线为8位的线路D7~D0，控制信号有读写控制R/W#和访存请求MREQ#。

要求：

- (1) 画出地址译码方案
- (2) 如果ROM和RAM存储器芯片都采用 $8K \times 1$ 的芯片，试画出存储器与CPU的连接图
- (3) 如果ROM存储器芯片采用 $8K \times 8$ 的芯片，RAM存储器芯片采用 $4K \times 8$ 的芯片，试画出存储器与CPU的连接图
- (4) 如果ROM存储器芯片采用 $16K \times 8$ 的芯片，RAM存储器芯片采用 $8K \times 8$ 的芯片，试画出存储器与CPU的连接图

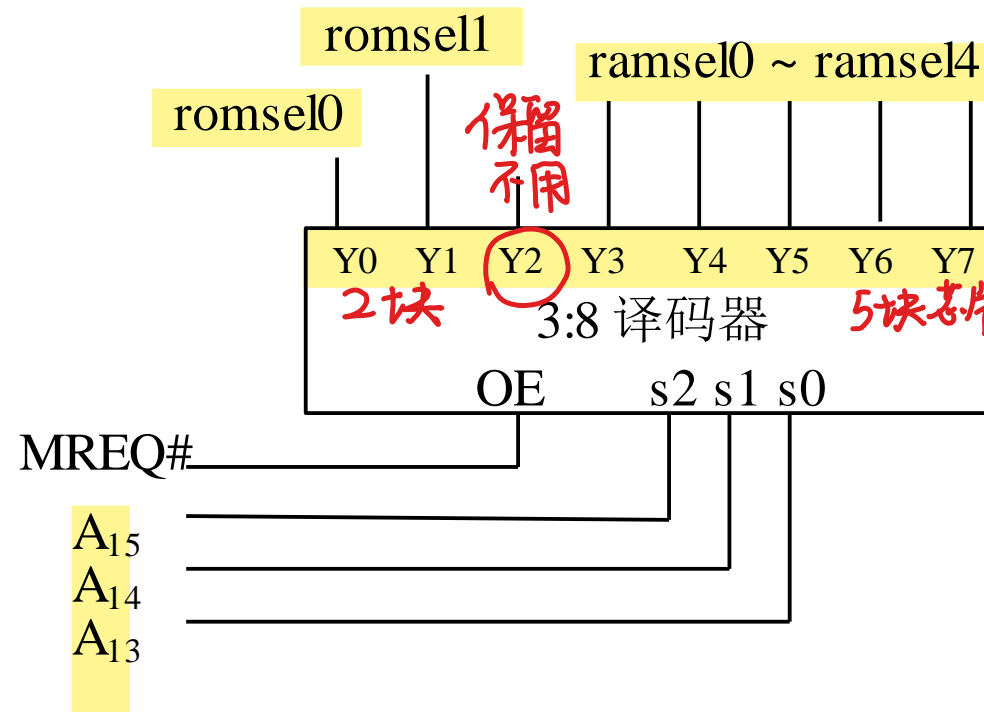
## (1) 画出地址译码方案

一个地址是一个字节

- 全部地址空间为 $2^{16} = 64\text{KB}$ 
  - ROM存储区域的容量为16KB
  - 保留存储区域容量为8KB
  - RAM的存储区域为40KB
  - 地址译码采用以8KB为区域单位，将64KB的存储空间分为8个8KB的区域
  - 用地址的高3位作为区域选择译码信号

$$4 \times (2^4)^3 = 2^4 \times 2^{10}$$

# (1) 画出地址译码方案



- Y0和Y1的输出作为ROM的选择信号，因为ROM的地址区域为0000到3FFF，其地址的A15-13位为000~001
- Y3到Y7的这5条输出信号作为RAM的选择信号，因为RAM的地址区域为3FFF到FFFF，其地址的A15-13位为011-111

# (1) 画出地址译码方案

译码器的输出信号逻辑表达式为：

$$\text{romsel0} = \overline{A15} * \overline{A14} * \overline{A13} * \overline{MREQ\#}$$

000

$$\text{romsel1} = \overline{A15} * \overline{A14} * A13 * \overline{MREQ\#}$$

001

$$\text{ramsel0} = \overline{A15} * A14 * A13 * \overline{MREQ\#}$$

011

$$\text{ramsel1} = A15 * \overline{A14} * \overline{A13} * \overline{MREQ\#}$$

100

$$\text{ramsel2} = A15 * \overline{A14} * A13 * \overline{MREQ\#}$$

101

$$\text{ramsel3} = A15 * A14 * \overline{A13} * \overline{MREQ\#}$$

110

$$\text{ramsel4} = A15 * A14 * A13 * \overline{MREQ\#}$$

111

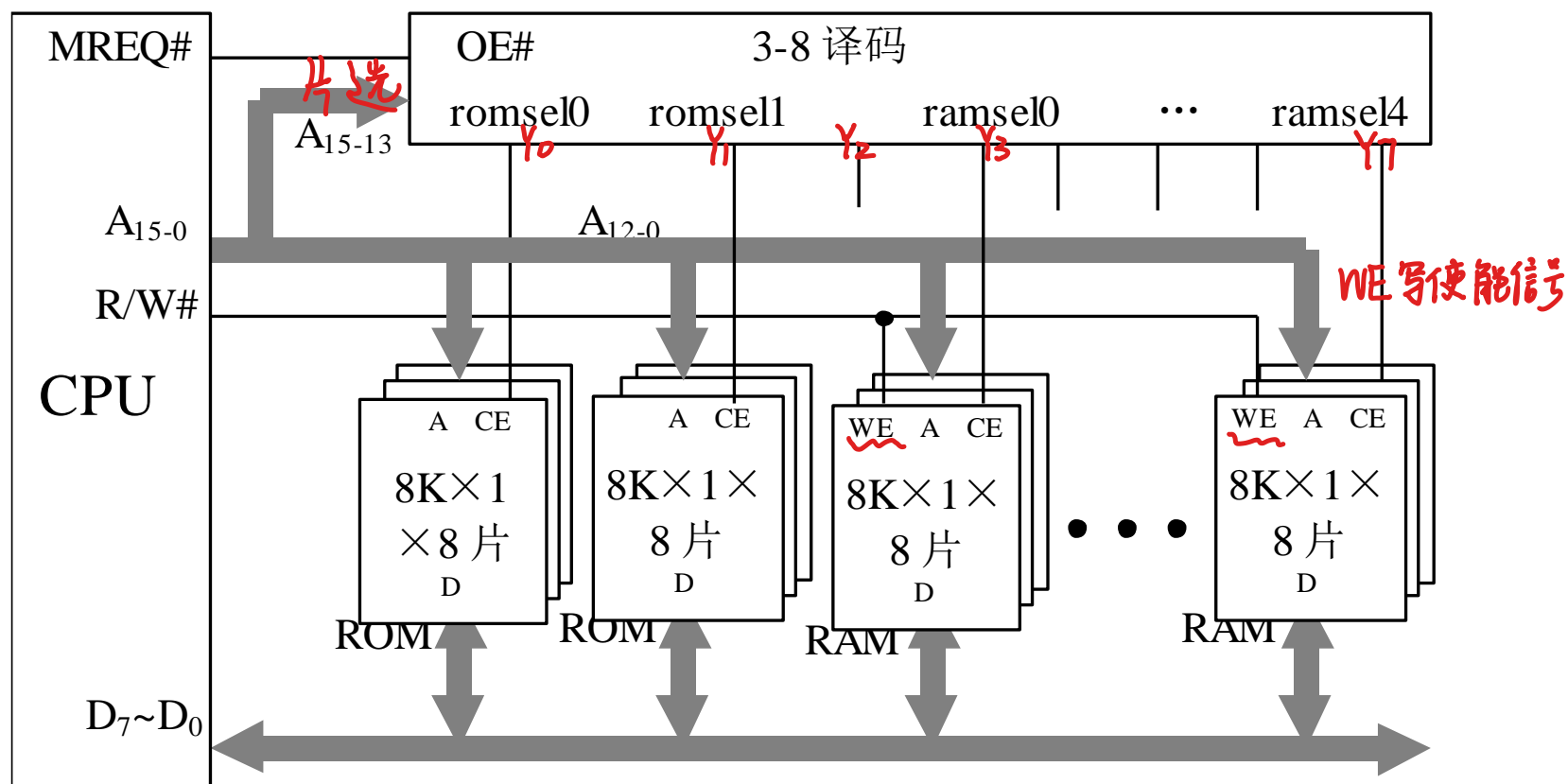
中间的不用



(2) 如果ROM和RAM存储器芯片都采用 $8K \times 1$ 的芯片，试画出存储器与CPU的连接图。

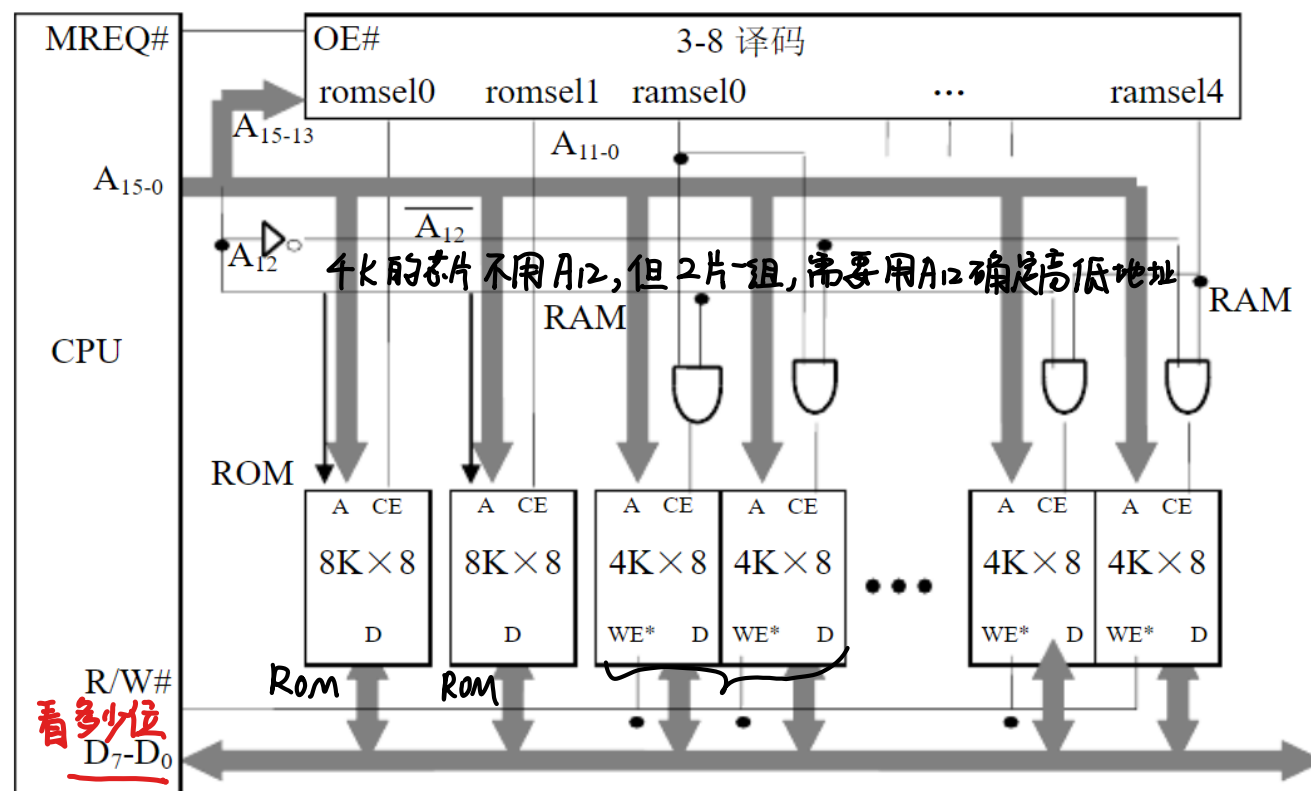
8KB的存储区域可以用8片存储器芯片构成一组实现 $8K \times 1$ 的存储器芯片的地址线需要13条，即 $A_{12} \sim 0$  1B =  $2^8$  位 故使用位扩展叠8片

ROM芯片没有R/W#控制信号输入



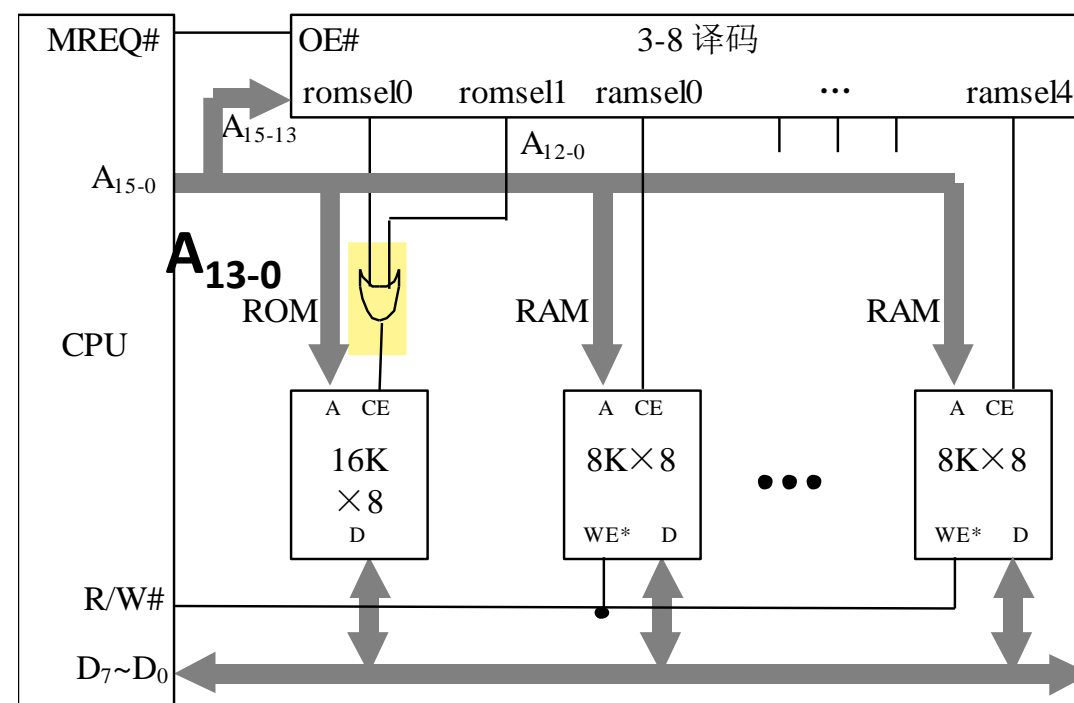
(3) 如果ROM存储器芯片采用 $8K \times 8$ 的芯片，RAM存储器芯片采用 $4K \times 8$ 的芯片，试画出存储器与CPU的连接图

- 8KB的存储区域可以用1片ROM芯片或2片RAM存储器芯片构成一组实现
- $8K \times 8$ 的ROM芯片的地址线需要13条，即A12-A0 地址线多少由容量决定
- $4K \times 8$ 的RAM芯片的地址线需要12条，即A11-A0  $8KB = 2^{13}B$  用13根



(4) 如果ROM存储器芯片采用 $16K \times 8$ 的芯片，RAM存储器芯片采用 $8K \times 8$ 的芯片，试画出存储器与CPU的连接图

- 16KB的ROM存储区域可以用1片ROM芯片实现，其余RAM存储区域可以用RAM芯片实现。
- 16K\*8的ROM芯片地址线需要14条，即A13-A0；8K\*8的RAM芯片的地址线需要13条，即A12-A0 *地址线看芯片容量*



# 本节目录

- 存储器容量扩充
- 高级DRAM结构\*\*\*（简单讲述，不要求掌握）
- 只读存储器ROM
- 存储器接口设计
- 并行存储器\*\*\*（简单讲述，不要求掌握）

# 并行存储器\*\*\*

- 由于CPU和主存储器之间在速度上是不匹配的，这种情况便成为限制高速计算机设计的主要问题。
- 解决途径
  - 设置各种缓冲器
    - 通用寄存器
  - 采用分层的存储系统
    - Cache
    - 虚拟存储系统
  - 多个存储器并行工作
    - 并行访问和交叉访问

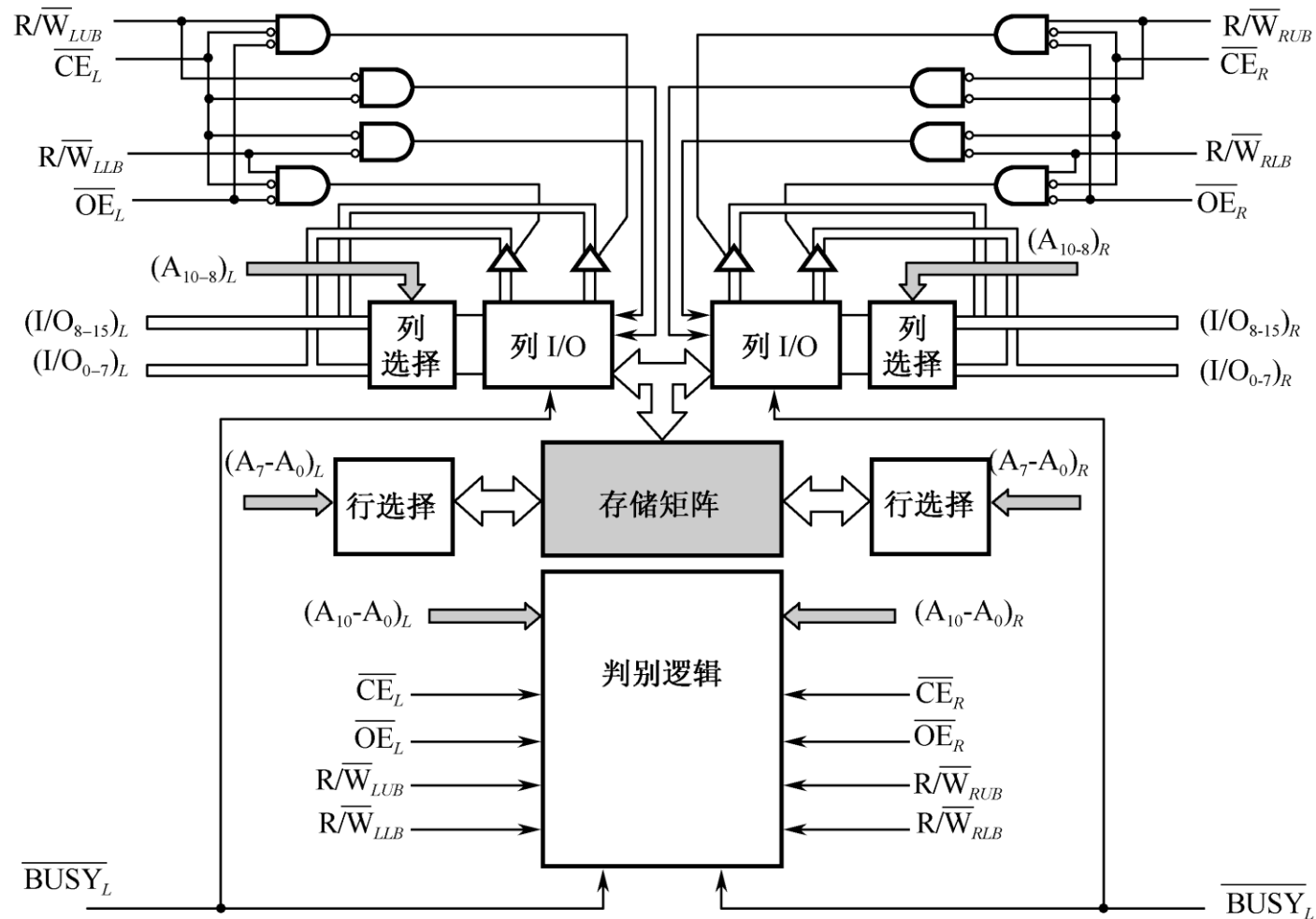
# (1) 双端口存储器

## ■ 双端口存储器

- 双端口存储器由于同一个存储器具有两组相互独立的读写控制电路而得名。
- 由于进行并行的独立操作，因而是一种高速工作的存储器。

# (1) 双端口存储器

## ■ 双端口存储器 IDT7133 的逻辑框图



# (1) 双端口存储器

## ■ 无冲突读写控制

- 当两个端口的地址不相同时，在两个端口上进行读写操作，一定不会发生冲突
- 当任一端口被选中驱动时，就可对整个存储器进行存取，每一个端口都有自己的片选控制(CE)和输出驱动控制(OE)
- 读操作时，端口的OE(低电平有效)打开输出驱动器，由存储矩阵读出的数据就出现在I/O线上



# (1) 双端口存储器

## ■ 有冲突读写控制

- 当两个端口同时存取存储器同一存储单元时，便发生读写冲突
- 为解决此问题，特设置了BUSY标志
- 在这种情况下，片上的判断逻辑可以决定对哪个端口优先进行读写操作，而对另一个被延迟的端口置BUSY标志(BUSY变为低电平)，即暂时关闭此端口。

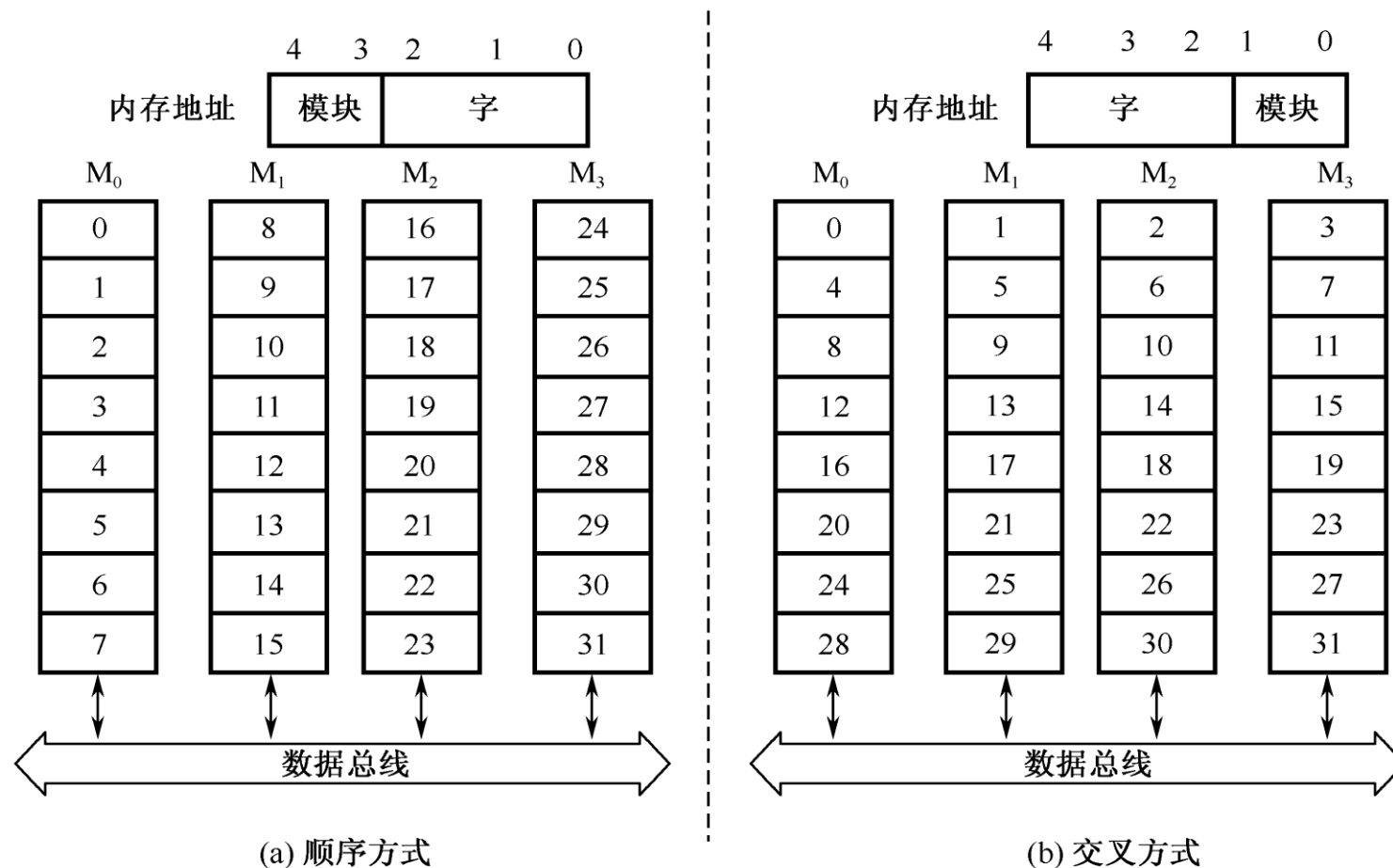
# (1) 双端口存储器

## ■ 有冲突读写控制判断方法

- 如果地址匹配且在CE之前有效，片上的控制逻辑在 $CE_L$ 和 $CE_R$ 之间进行判断来选择端口(CE判断)
  - 如果CE在地址匹配之前变低，片上的控制逻辑在左、右地址间进行判断来选择端口(地址有效判断)
- 
- 无论采用哪种判断方式，延迟端口的BUSY标志都将置位而关闭此端口
  - 当允许存取的端口完成操作时，延迟端口BUSY标志才进行复位而打开此端口

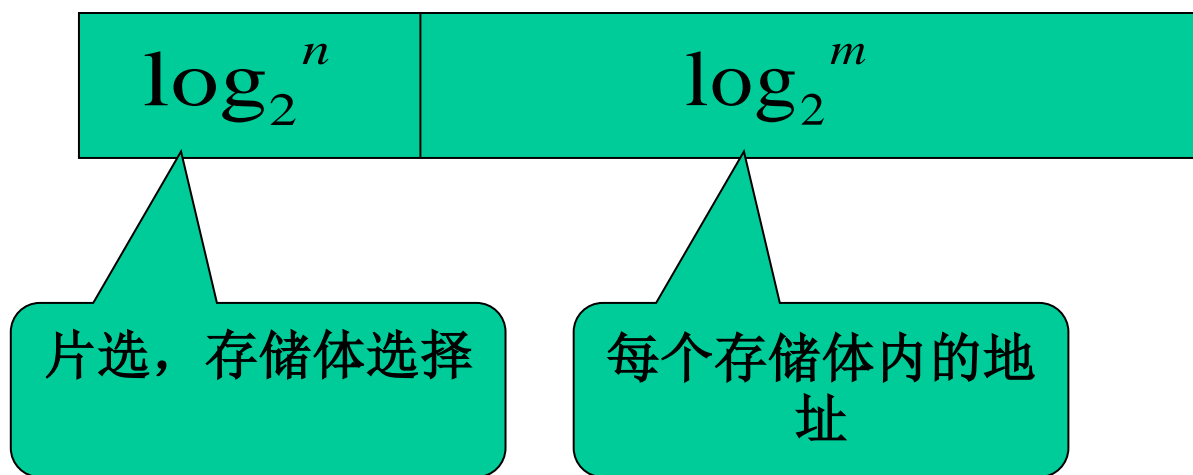
## (2) 多模块交叉存储器

- 多模块交叉存储器：一个由若干个模块组成的主存储器是线性编址的。这些地址在各模块中如何安排，有两种方式：一种是顺序方式，一种是交叉方式



## (2) 多模块交叉存储器

- 假设有 $n$ 个存储体，每个存储体的容量为 $m$ 个存储单元
- 顺序方式：



# 多模块交叉存储器

## ■ 顺序方式

[例] M0—M3共四个模块，则每个模块8个字

顺序方式： M0： 0—7

M1： 8—15

M2： 16—23

M3： 24—31

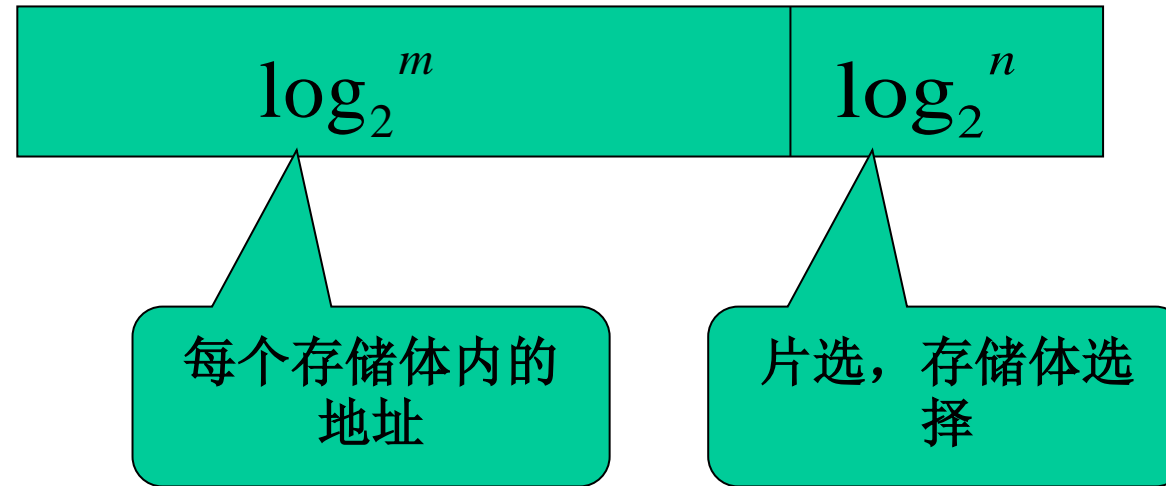
## ■ 高位选模块，低位选块内地址

## ■ 特点

- 某个模块进行存取时，其他模块不工作
- 某一模块出现故障时，其他模块可以照常工作
- 通过增添模块来扩充存储器容量比较方便
- 缺点是各模块串行工作，存储器的带宽受到了限制

# 多模块交叉存储器

- 交叉方式（可以实现多模块流水式并行存取）



# 多模块交叉存储器

[例] M0—M3共四个模块，则每个模块8个字  
交叉方式：

M0: 0, 4, ... 除以4余数为0

M1: 1, 5, ... 除以4余数为1

M2: 2, 6, ... 除以4余数为2

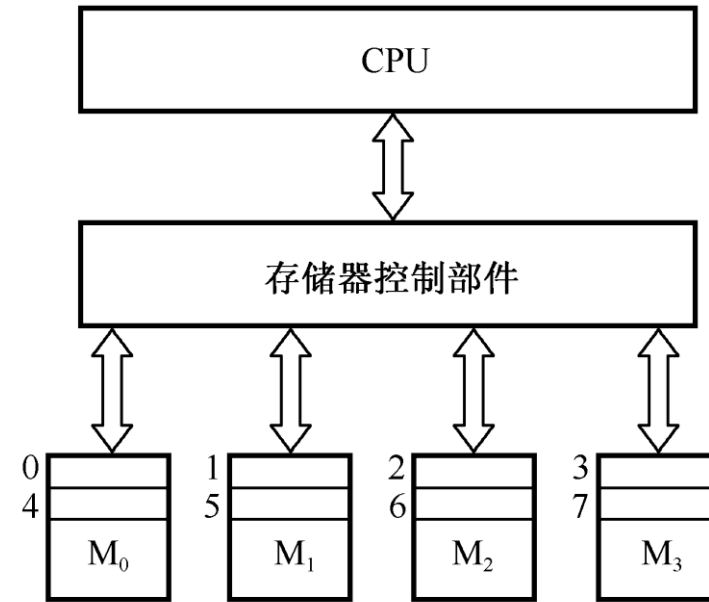
M3: 3, 7, ... 除以4余数为3

高位选块内地址，低位选模块

## ■ 特点

- 连续地址分布在相邻的不同模块内
- 同一个模块内的地址都是不连续的
- 优点是对连续字的成块传送可实现多模块流水式并行存取，大大提高存储器的带宽
- 使用场合为成批数据读取

# 多模块交叉存储器的基本结构



- 四模块交叉存储器结构框图
- 主存被分成4个相互独立、容量相同的模块 $M_0$ ,  $M_1$ ,  $M_2$ ,  $M_3$
- 每个模块都有自己的读写控制电路、地址寄存器和数据寄存器，各自以等同的方式与CPU传送信息
- 在理想情况下，如果程序段或数据块都是连续地在主存中存取，那么将大大提高主存的访问速度



# 多模块交叉存储器的基本结构

## ■ 并行存储器结构

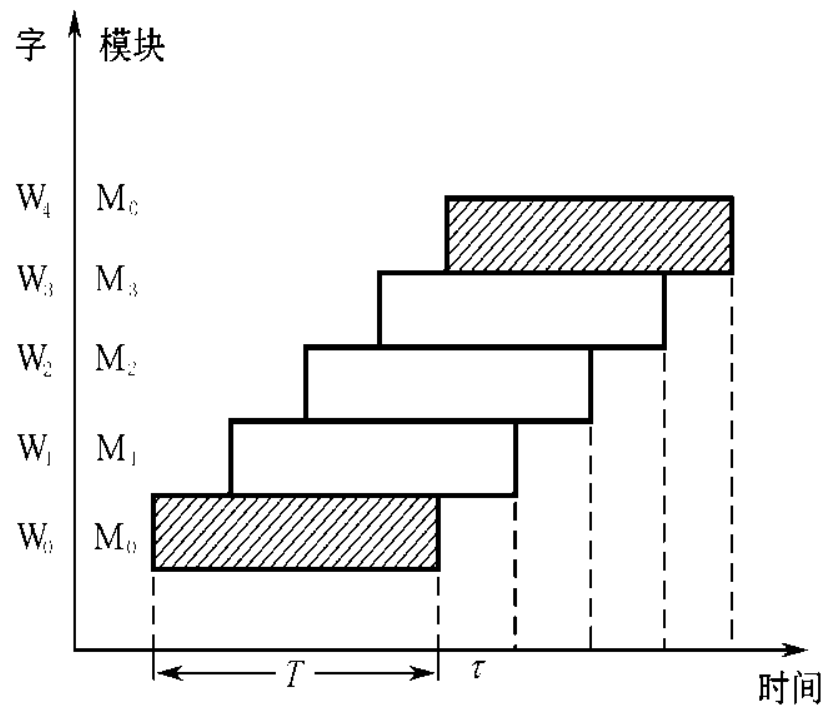
- CPU同时访问四个模块，由存储器控制部件控制它们分时使用数据总线进行信息传递
- 对每一个存储模块来说，从CPU给出访存命令，到读出信息仍然使用了一个存储周期时间
- 对CPU来说，可以在一个存取周期内连续访问四个模块
- 各模块的读写过程将重叠进行

# 多模块交叉存储器的基本结构

## ■ 流水线方式存取

- 模块字长等于数据总线宽度
- 模块存取一个字的存储周期为 $T$ ，存储器的交叉模块数为 $m$
- 通常在一个存储器周期内， $m$ 个存储体必须分时启动，则各个存储体的启动间隔为  $\tau = T / m$
- $m$ 为交叉存取度
- 连续取 $m$ 个字的时间为  

$$t_{\text{交叉}} = T + (m-1) \tau$$
- 顺序方式存储器连续读取  
 $m$ 个字所需时间为
 
$$t_{\text{顺序}} = mT$$



例 设存储器容量为32字，字长64位，模块数 $m=4$ ，分别用顺序方式和交叉方式进行组织。存储周期 $T=200\text{ns}$ ，数据总线宽度为64位，总线传送周期 $=50\text{ns}$ 。若连续读出4个字，问顺序存储器和交叉存储器的带宽各是多少？

- 顺序存储器和交叉存储器连续读出 $m=4$ 个字的信息总量是  
 $q=64\text{b} \times 4=256\text{b}$
- 顺序存储器和交叉存储器连续读出4个字所需时间分别是：  
 $t_2=mT=4 \times 200\text{ns}=800\text{ns}=8 \times 10^{-7}\text{s}$   
 $t_1=T+(m-1) \times 50\text{ns}=200\text{ns}+(4-1) \times 50\text{ns}=350\text{ns}=35 \times 10^{-7}\text{s}$
- 顺序存储器和交叉存储器的带宽分别是：  
 $W_2=q/t_2=256\text{b} \div (8 \times 10^{-7})\text{s}=320\text{Mb/s}$   
 $W_1=q/t_1=256\text{b} \div (35 \times 10^{-7})\text{s}=730\text{Mb/s}$