

# 计算机组成原理

## Principle of Computer Organization

### ➤ 第三章 存储系统 第一部分

北京邮电大学  
计算机学院

戴志涛



北京邮电大学

计算机学院

2021/3/30

1

# 本章内容



- 存储系统的基本概念
- 半导体存储器的结构和接口
- 并行存储器
- cache
- 虚拟存储器



# 存储系统的层次结构

## ➤ 早期的计算机系统：存储容量/访问速度要求不高

- ❑ 1970s：中型机，十几kB存储器
- ❑ 个人计算机操作系统MSDOS：1MB



## ➤ 近年来，对存储容量和存储速度的要求不断提升

- ❑ 软件复杂度提高、多媒体技术和网络技术的普及导致对存储容量的要求不断提高
- ❑ CPU的工作速度不断提高
  - ☒ CPU比内存的工作速度提高更快

## ➤ 存储器的价格较高，在整机成本中占有较大的比例

## ➤ 在成本和性能之间存在矛盾，要求使用某种策略解决这一矛盾——

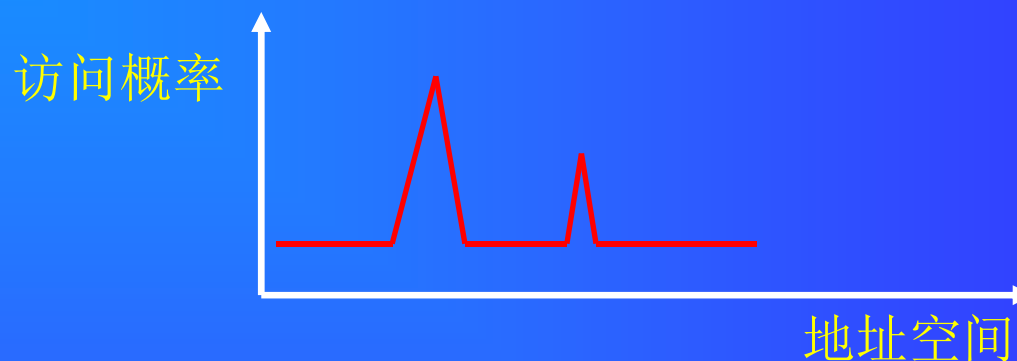
## 存储器分层



# 存储系统的层次结构

## ➤ 程序的局部性原理

- 在一个较短的时间间隔内，程序所访问的存储器地址在很大比例上集中在存储器地址空间的很小范围内——指令 数据



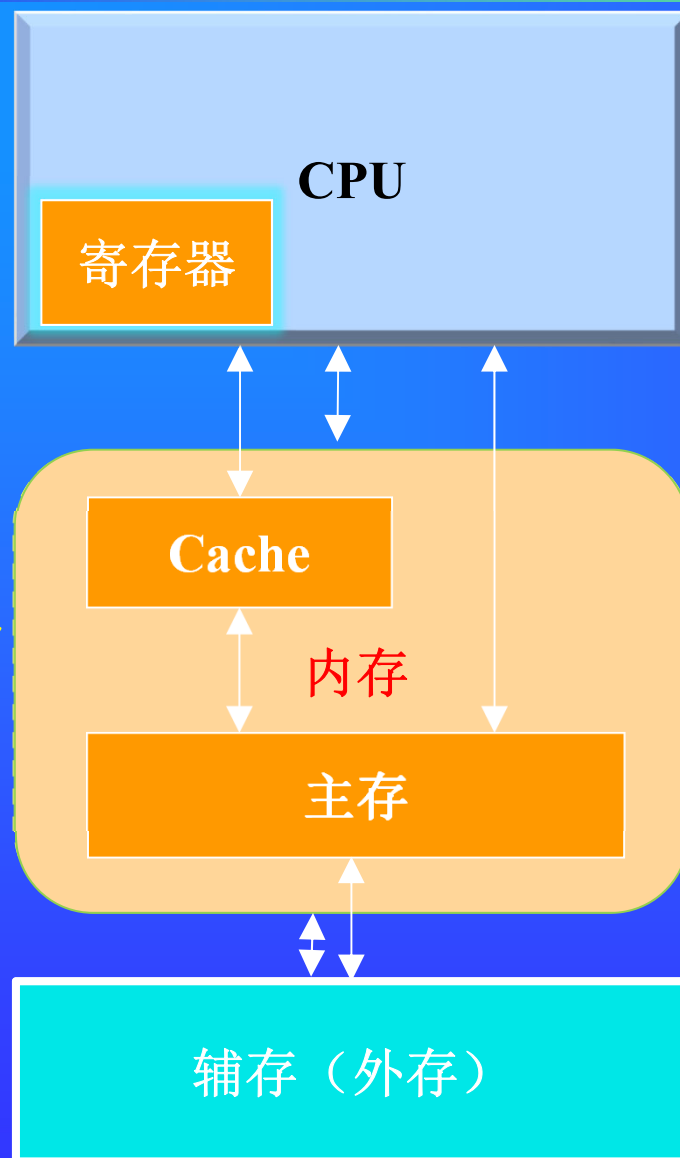
- **程序的局部性原理**：在某一段时间内频繁访问某一局部的存储器地址空间，而对此范围以外的地址空间则很少访问的现象

✉ **时间局部性**：最近被访问的信息很可能还要被访问

✉ **空间局部性**：最近被访问的信息临近的信息也可能被访问



# 存储系统层次结构的组成



内存：速度  
高、容量小、  
价格高，由  
半导体器件  
构成

## ➤ CPU内部寄存器

❑ 速度高、数量少

## ➤ 两级存储体系的组成

❑ 内存

❑ 外存（辅存）

外存：速度低、容量  
大、价格便宜，多由  
非半导体器件构成



# 多级存储体系的组成

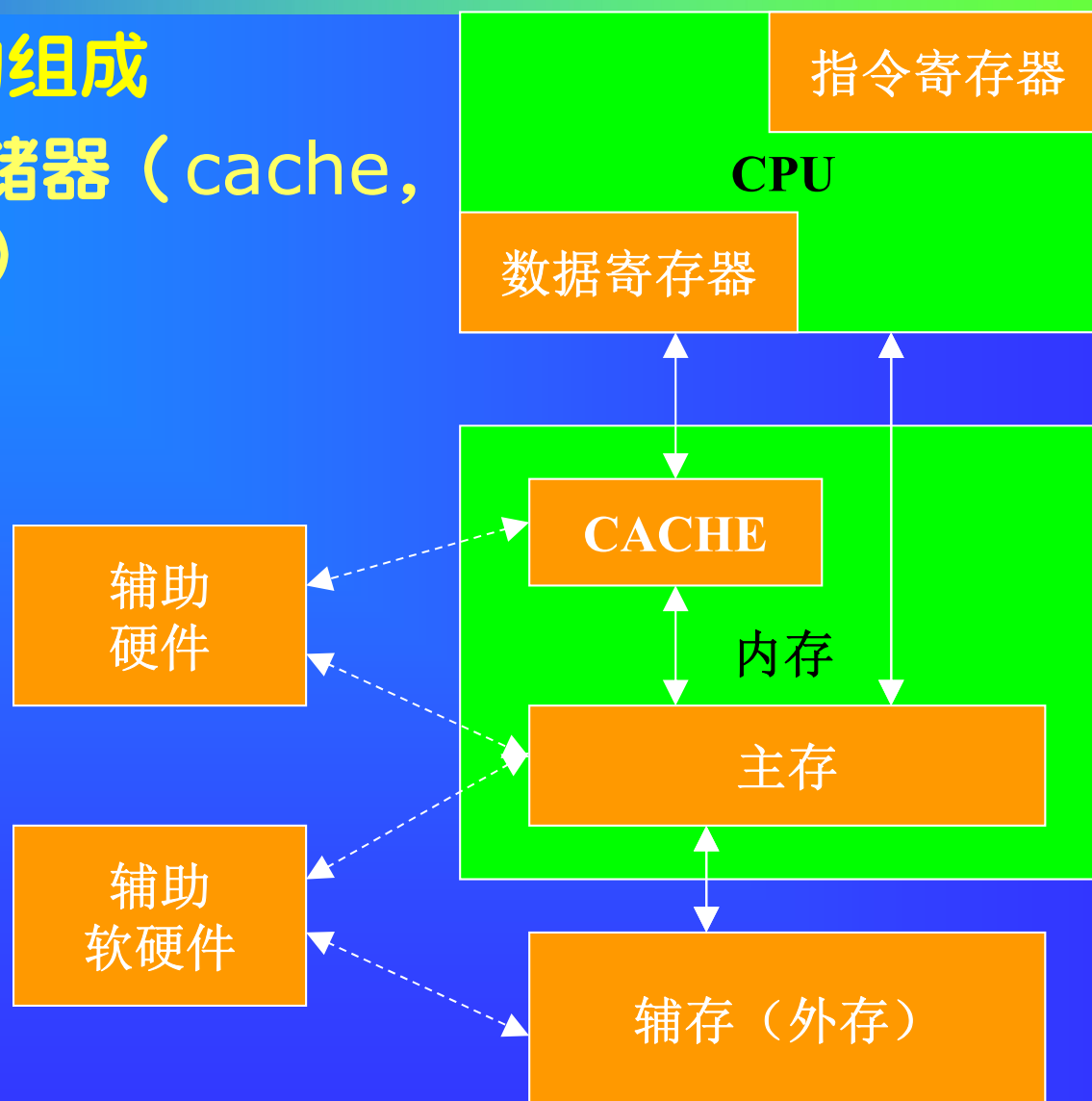
## ➤ 三级存储体系的组成

□ 高速缓冲存储器（cache，缓存、快存）

□ 主存

□ 辅存

➤ 内存 =  
cache + 主存



# Levels of the Memory Hierarchy

Capacity, Access Time  
Bandwidth, Price/GB

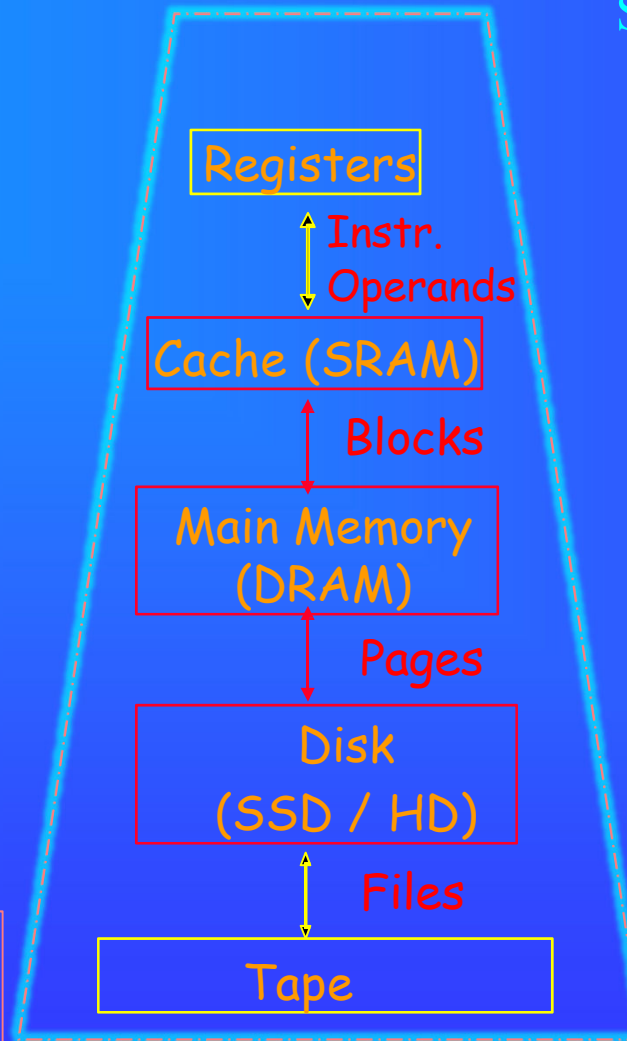
100s Bytes  
<1ns

K-M Bytes, 1-5 ns  
25+ Gb/s, \$100

G Bytes, 10-50 ns  
10 Gb/s, \$10

T Bytes    T Bytes  
0.1 ms    10 ms  
0.5 Gb/s    0.15 Gb/s  
\$1    \$0.15

infinite (1.5 T/tape)  
1 min 0.14 Gb/s, \$0.04



Staging Transfer Unit    **Upper Level**

prog./compiler  
1-8 bytes

cache controller  
8-128 bytes

OS  
512-4K bytes

user/operator  
M-G bytes

faster

Larger



# 多级存储体系的组成



- 外存：主要为解决存储容量要求
- cache：更主要解决速度要求
- 出发点：提高存储系统的性能-价格比——“多快好省”
- 整个存储系统在速度上接近cache，而在容量和价格上接近外存
- 一级（L1）cache + 二级（L2）cache + 三级（L3）cache
- 指令cache（I-cache） + 数据cache（D-cache）





# 内存存储器的编址方式

## ➤ 编址方式——存储系统中存储器地址的组织方式

❑ 编址方式在设计处理机时确定

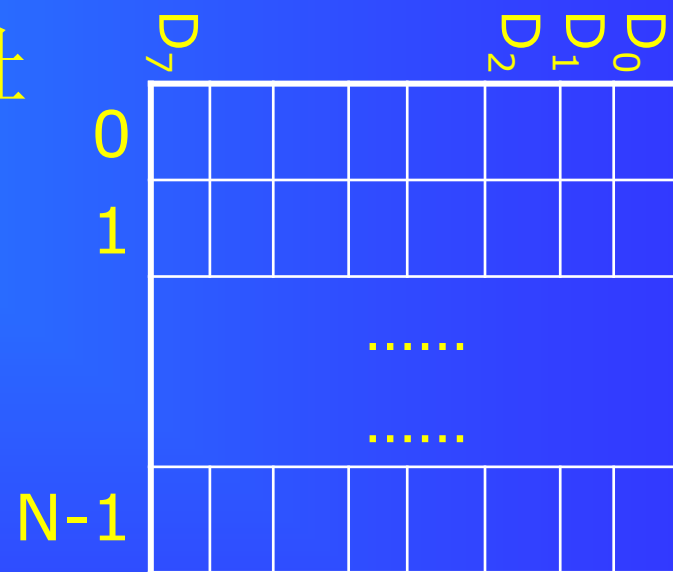
## ➤ 每个存储单元有一个地址

❑ 按字节编址

❑ 按字编址

❑ 按半字编址

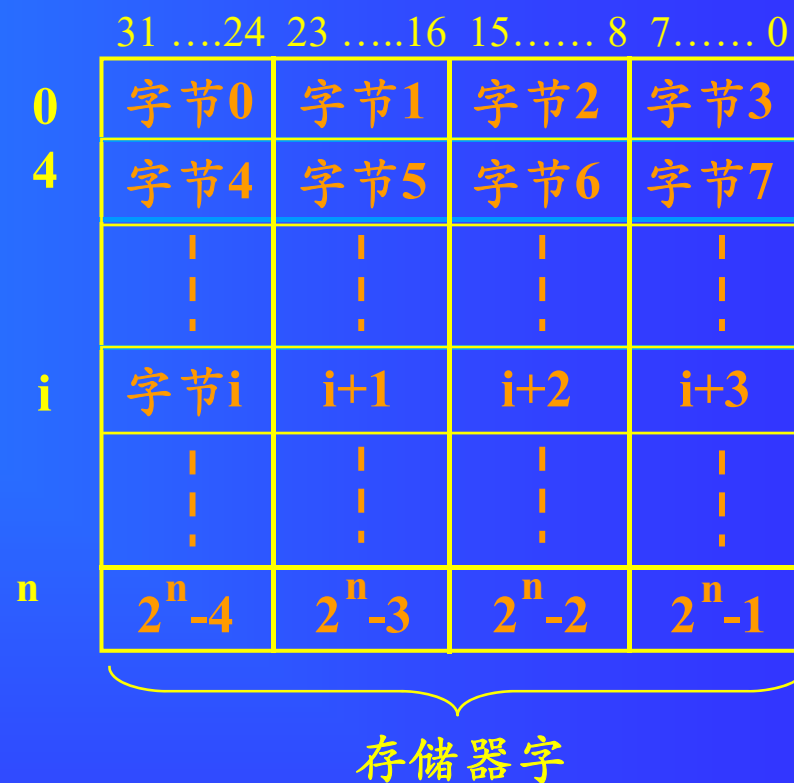
❑ 按1/4字编址



# 存储器访问

## ➤每次访问存储器读写的信息量:

- 字寻址: 每次访问读写一个存储字
- 半字寻址: 每次访问读写半个存储字
- 字节寻址: 每次访问读写一个字节



字长=4字节



# 内存的主要技术指标



## ➤ 存储容量

□ 存储器可存储的信息的字节数或比特数

□ 通常表示为

存储字数 × 存储字长

(存储单元数) (每单元的比特数)

□ 例：1M比特的存储器可以组织成

✉ 1M × 1比特

✉ 128K × 8比特

✉ 256K × 4比特



# 内存的主要技术指标



## ➤ 存取速度

### □ 访问时间（存取时间） $T_A$ :

- ☒ 从存储器接收到读/写命令到信息被读出或写入完成所需的时间
- ☒ 决定于存储介质的物理特性和寻址部件的结构

### □ 存取周期 $T_M$ :

- ☒ 在存储器连续读写过程中一次完整的存取操作所需的时间
- ☒ CPU连续两次访问存储器的最小时间间隔



# 内存的主要技术指标



## ➤存取速度

□ 数据传送速率（频宽） $B_M$ :

☒ 单位时间内能够传送的信息量

☒ 若系统的总线宽度为  $W$  bit, 则  $B_M = W/T_M$   
(b/s)



# 存储器的分类



## ➤ 按存储介质

半导体存储器，磁介质存储器，光存储器

## ➤ 按存储器与CPU的耦合程度

内存（主存+cache），外存

## ➤ 按存储器的读写功能

读写存储器（RWM, Read/Write Memory），

只读存储器（ROM, Read-Only Memory）



# 存储器的分类

## ➤ 按掉电后存储的信息可否保持

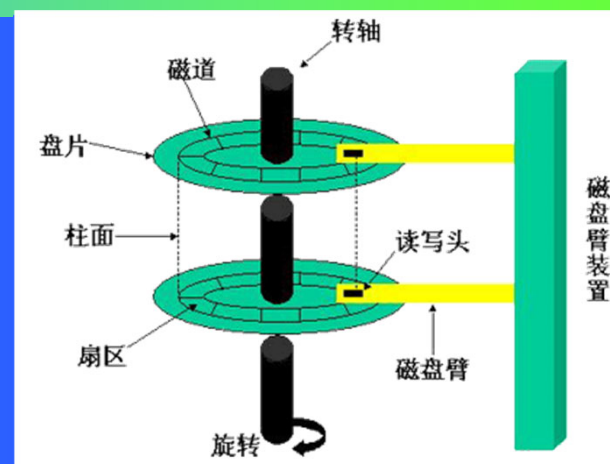
易失性（挥发性）存储器，  
非易失性（不挥发）存储器

## ➤ 按数据存取的随机性

随机存取存储器（RAM: Random Access Memory），  
顺序存取存储器（SAM: Sequential Access Memory），  
直接存取存储器（DAM: Direct Access Memory）

## ➤ 按访问的串并行性

并行存取存储器，串行存取存储器



# 存储器的分类



## ➤ 按时钟特性

同步存储器，异步存储器

## ➤ 按照存储器的访问方式

按地址访问的存储器，

按内容访问的存储器（CAM，相联存储器）

## ➤ 按照半导体存储器的信息储存方法

静态（static）存储器，

动态（dynamic）存储器

## ➤ 按存储器的功能

系统存储器，显示存储器，控制存储器.....





# 内存存储器的习惯分类

## ➤ RAM: 易失性半导体读写存储器

- ❑ 静态RAM (SRAM)

- ❑ 动态RAM (DRAM)

  - ✉ FastPage DRAM, SDR SDRAM, DDR SDRAM.....

## ➤ ROM: 非易失性半导体只读存储器

- ❑ 掩膜ROM (MASK ROM)

- ❑ 可编程ROM (PROM)

  - ✉ 一次性可编程ROM (OTP ROM)

  - ✉ 可擦除PROM (EPROM)

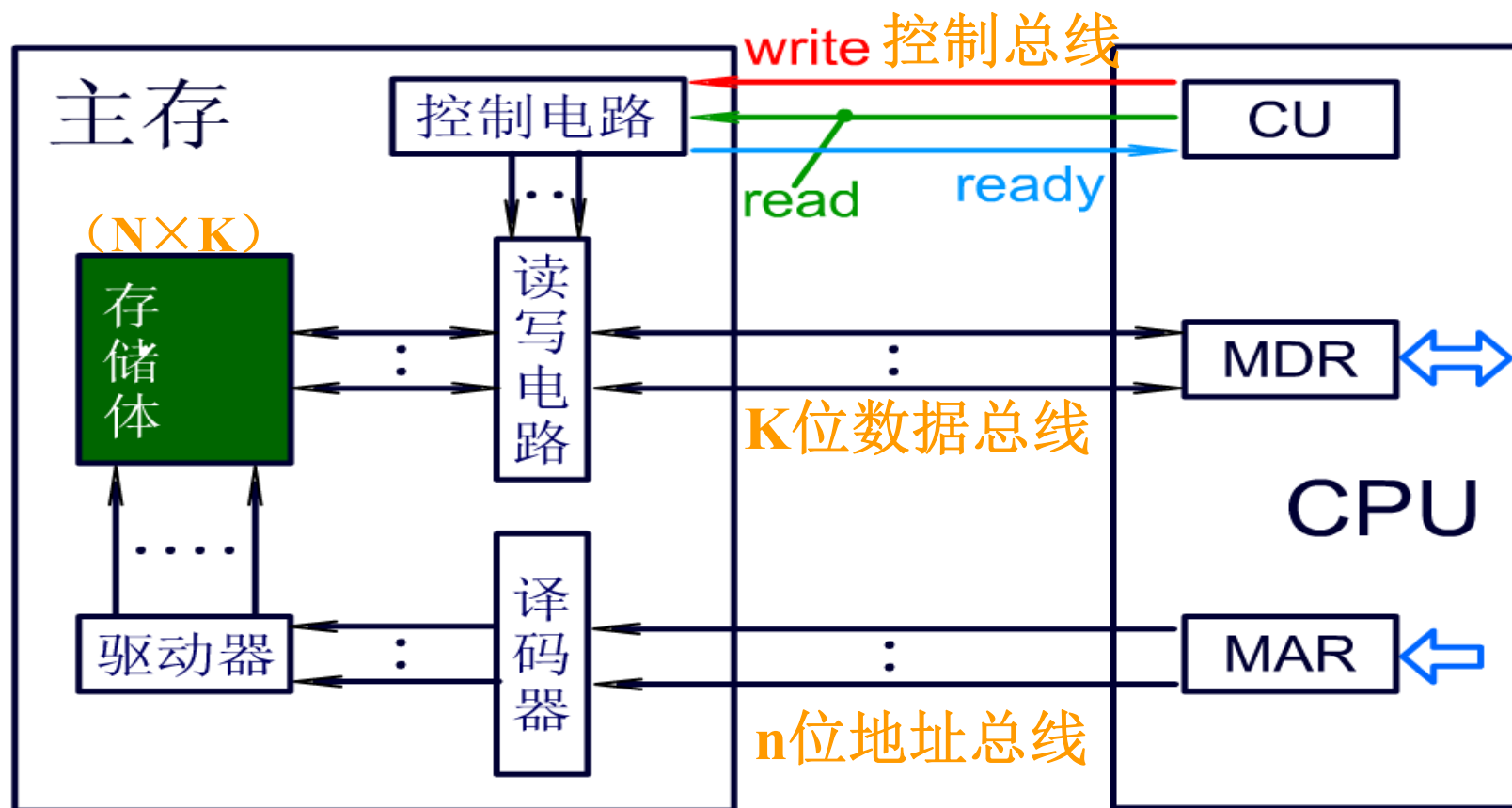
    - ❑ 紫外线擦除EPROM (UV EPROM)

    - ❑ 电擦除EPROM (EEPROM, E<sup>2</sup>PROM)

    - ❑ 闪速存储器 (FLASH ROM)



# 内存的总体结构



$$n = \log_2 N$$

MAR: 内存地址寄存器  
MDR: 内存数据寄存器



# 随机读写存储器 (RAM)



## ➤ 随机读写存储器

- ❑ 双极型半导体存储器

- ❑ 金属氧化物半导体 (MOS) 存储器

  - ✉ 静态MOS存储器 (SRAM)

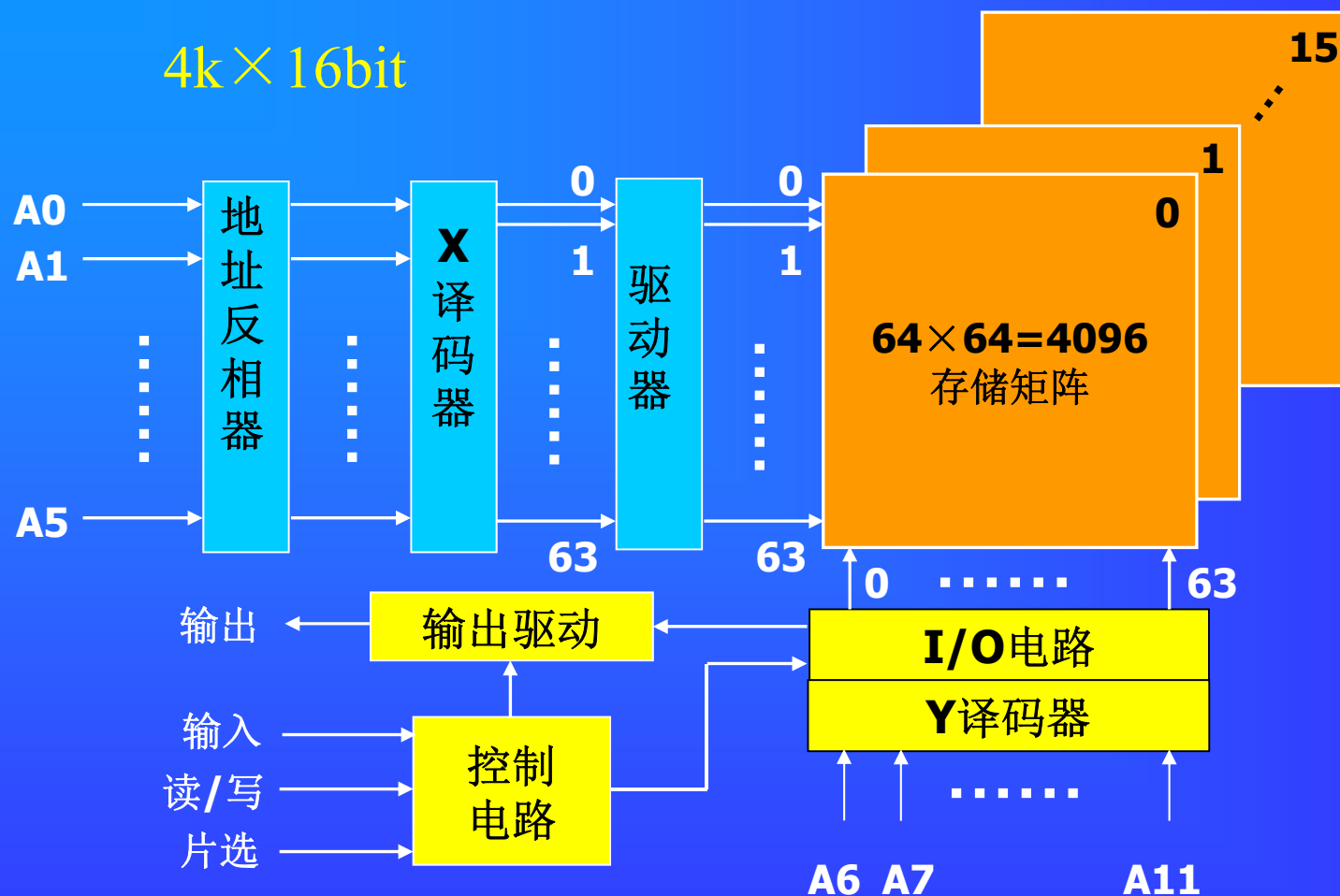
  - ✉ 动态MOS存储器 (DRAM)



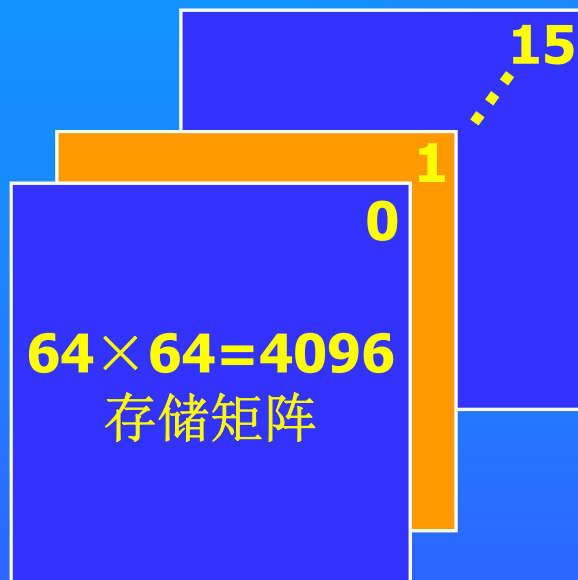
# 静态随机存取存储器 (SRAM)



# SRAM存储器的组成



# 存储矩阵的组织



	$D_{15}$	.....	$D_2$	$D_1$	$D_0$
0					
1					
...		...			
...		...			
4095					

➤ 每条行选择线 $X_i$ 要控制多少个基本存储元?

✓  $64 * 16 = 1024$

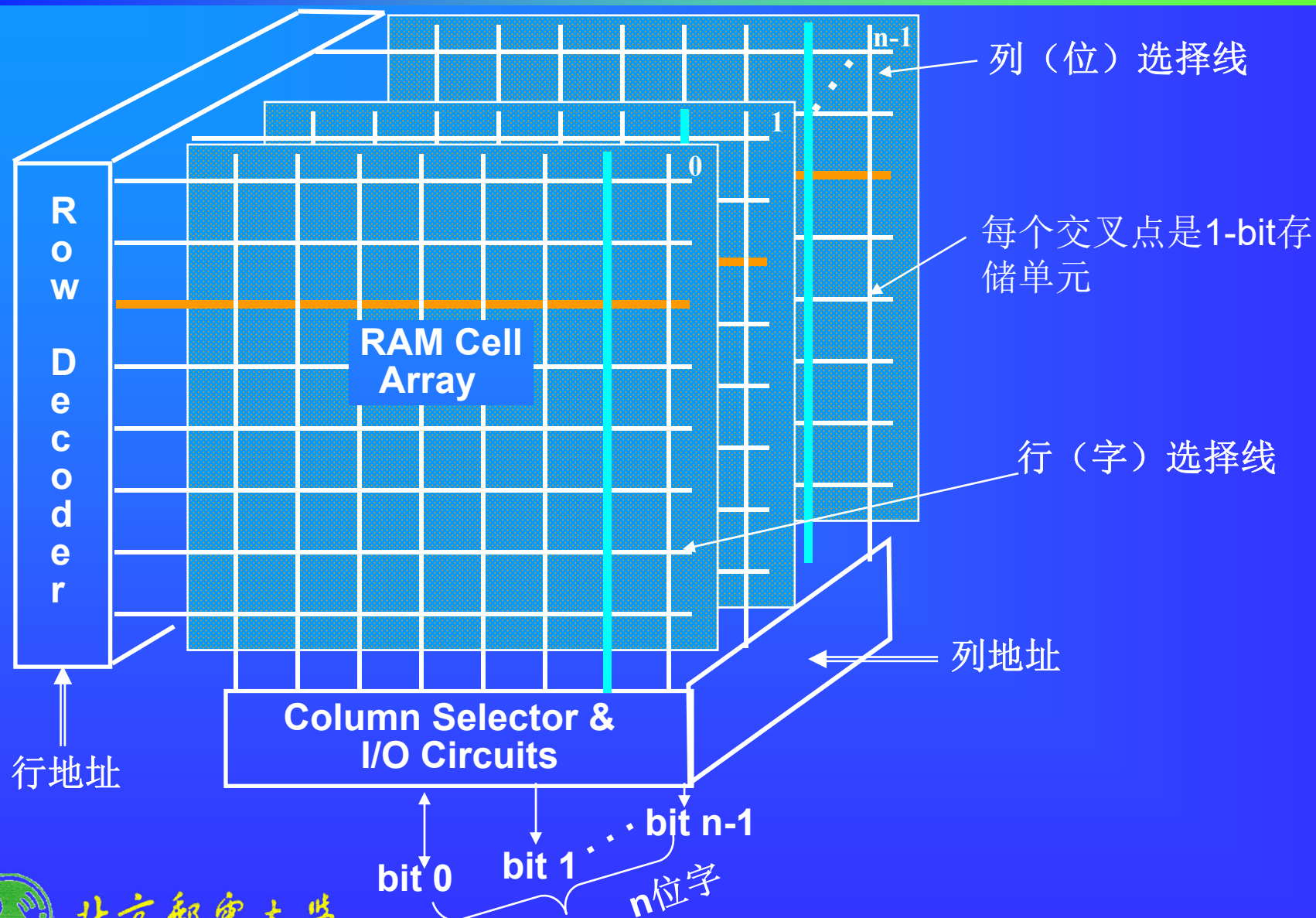
➤ 每条列选择线 $Y_i$ 要控制多少个基本存储元?

✓  $64 * 16 = 1024$

平面  
1

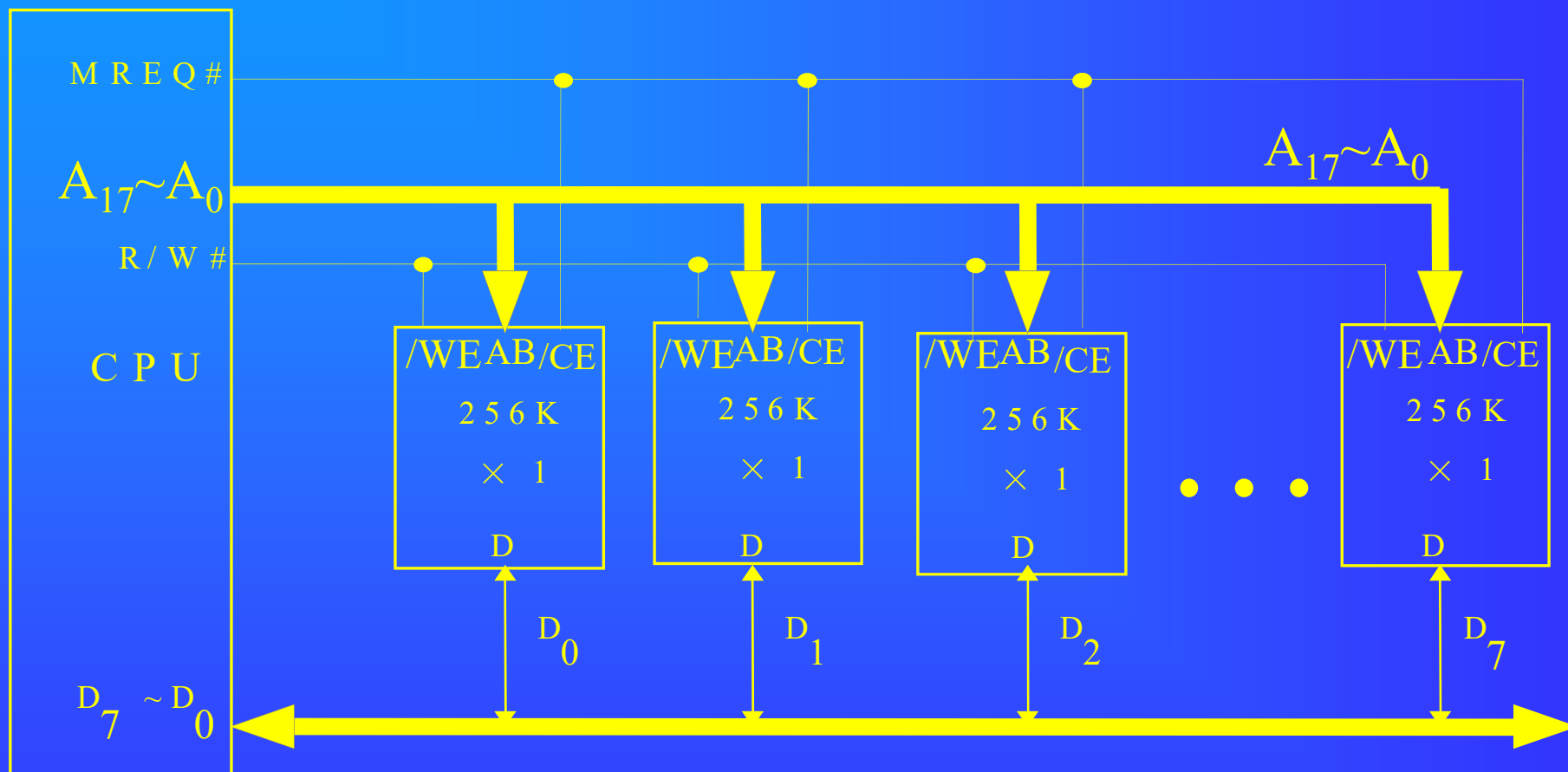


# n位存储器芯片的结构



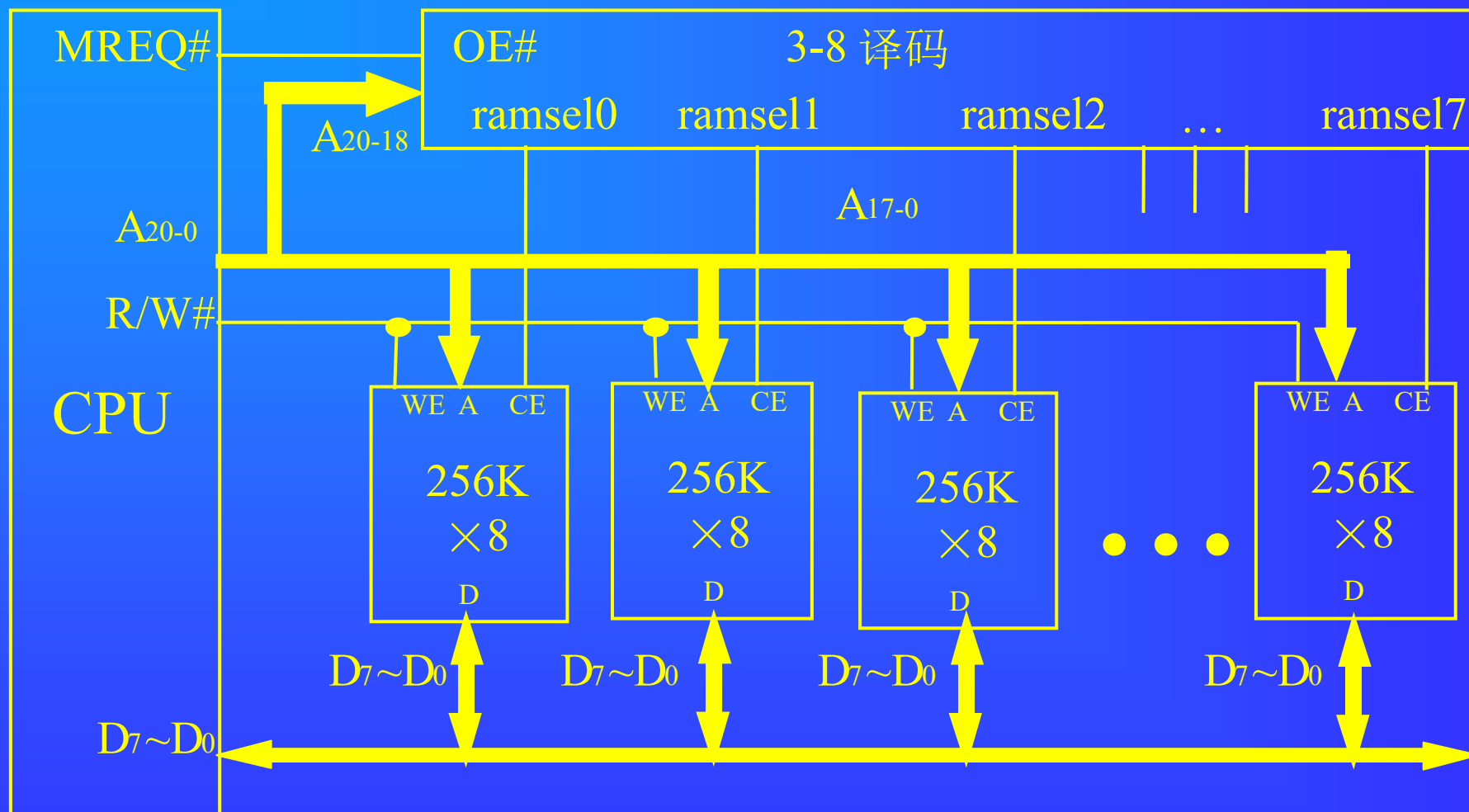
# 位扩展实例

256K × 1bit → 256K × 8bit





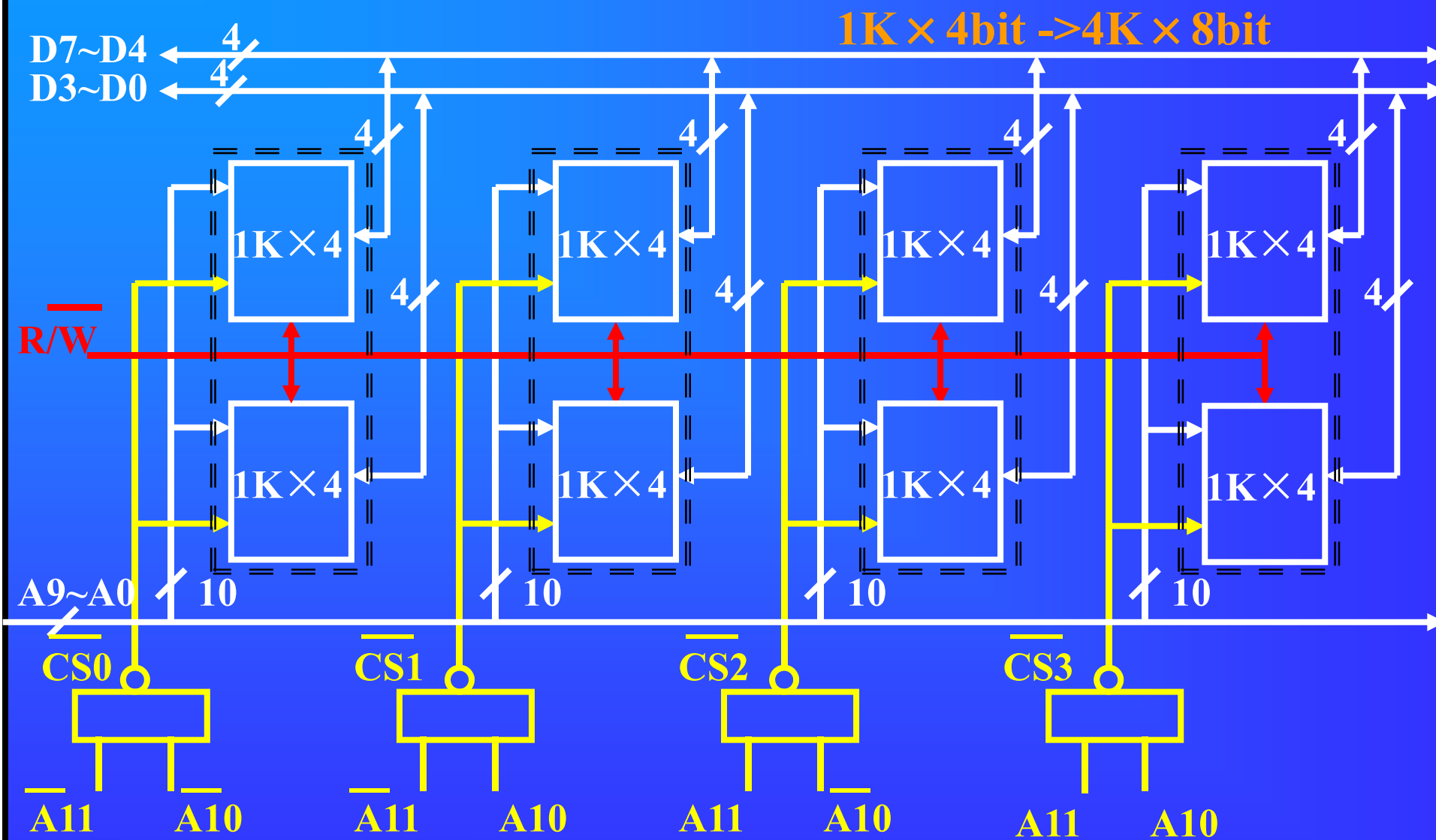
# 字扩展实例



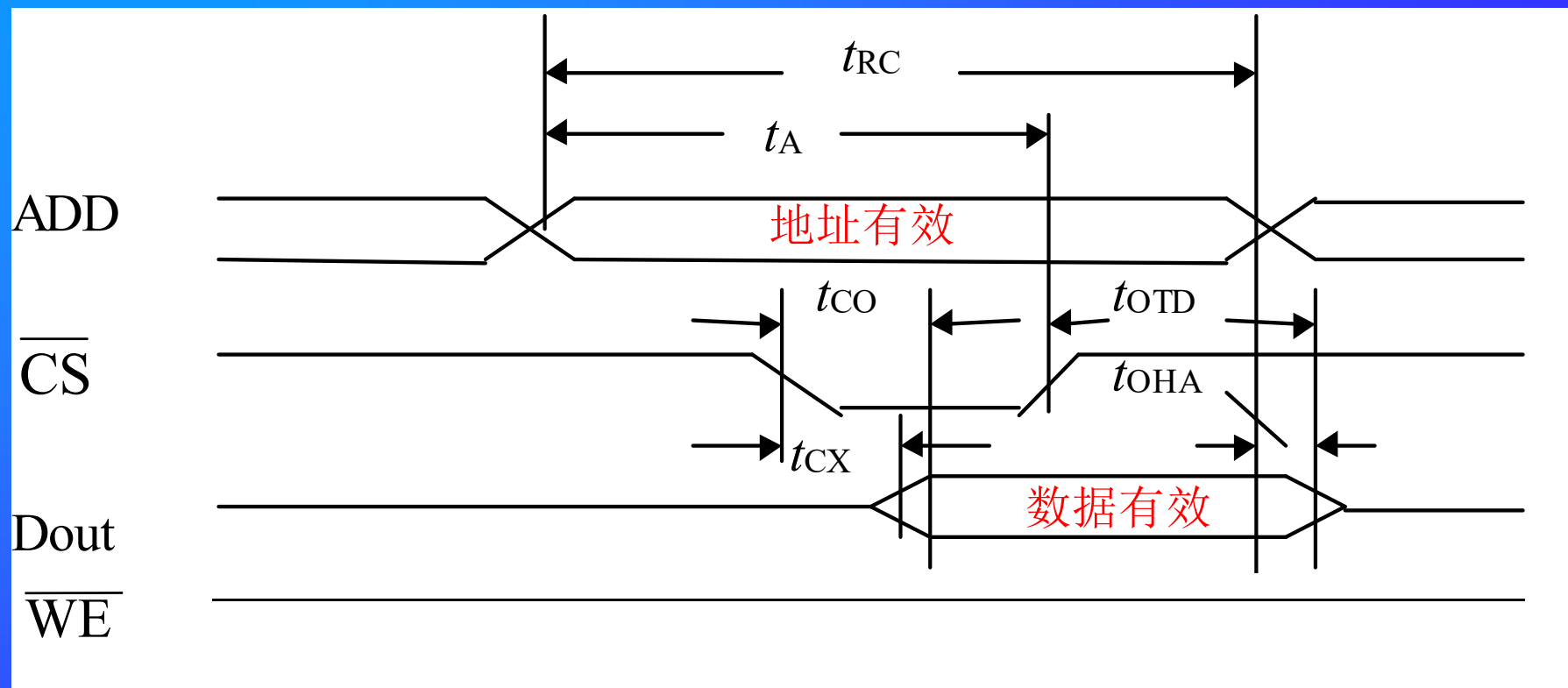
256K × 8bit → 2048K × 8bit



# 字位扩展实例



# SRAM时序——读周期



$t_{RC}$

读周期

$t_{CO}$

片选到数据输出延迟

$t_{OTD}$

从断开片选到输出变为三态

$t_A$

读出时间

$t_{CX}$

片选到输出有效

$t_{OHA}$

地址改变后的维持时间



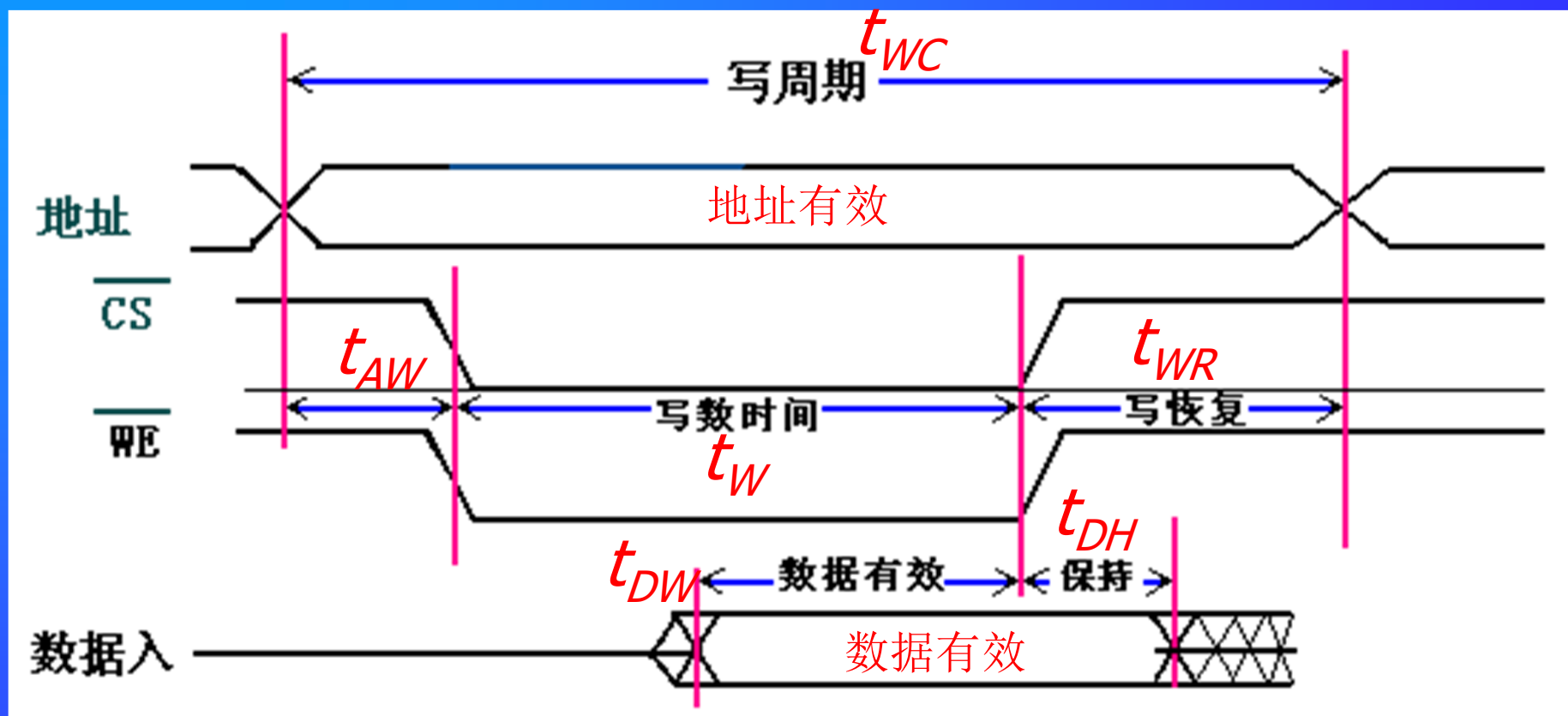
北京邮电大学

计算机学院

2021/3/30

59

# SRAM时序——写周期



$t_{WC}$  写周期时间  
 $t_{WR}$  写恢复时间  
 $t_{DW}$  数据有效时间

$t_W$  写数时间  
 $t_{DTW}$  从写信号有效到输出三态的时间  
 $t_{DH}$  写信号无效后数据保持时间



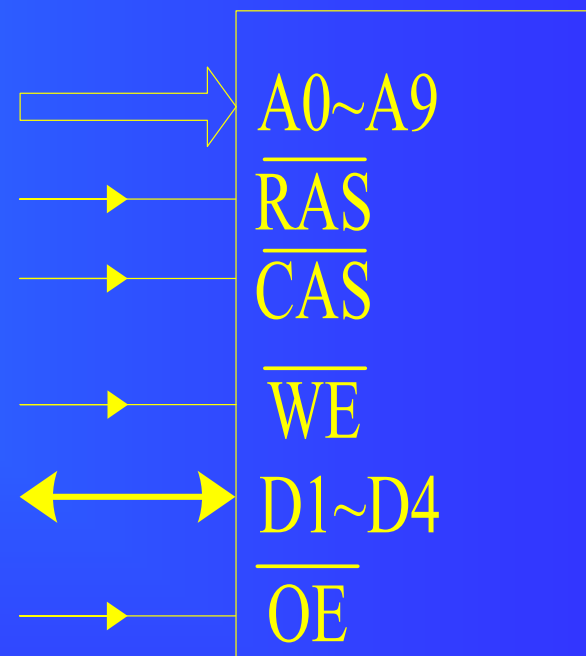
# 动态随机存取存储器 (DRAM)



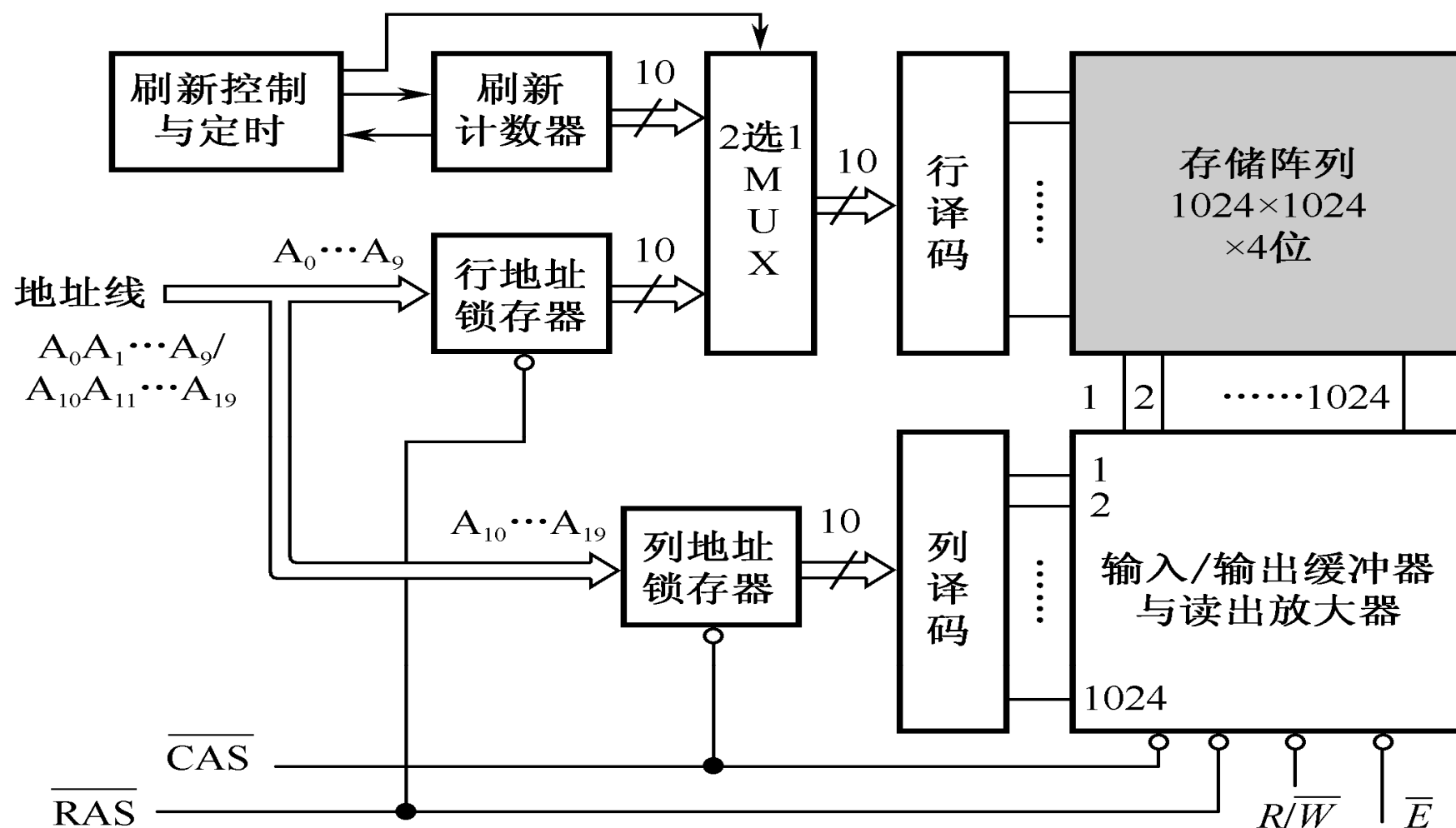
# DRAM的管脚信号

某1M × 4比特的DRAM芯片的外部信号:

- nWE: 写使能, 高电平读, 低电平写
- D<sub>1</sub> ~ D<sub>4</sub>: 数据输入/输出
- A<sub>0</sub> ~ A<sub>9</sub>: 地址线, 分时传送低10位行地址A<sub>0</sub> ~ A<sub>9</sub>和高10位列地址A<sub>10</sub> ~ A<sub>19</sub>
- nRAS: Row Address Strobe (行地址选通信号)
- nCAS: Column Address Strobe (列地址选通信号)
- nOE: Output Enable



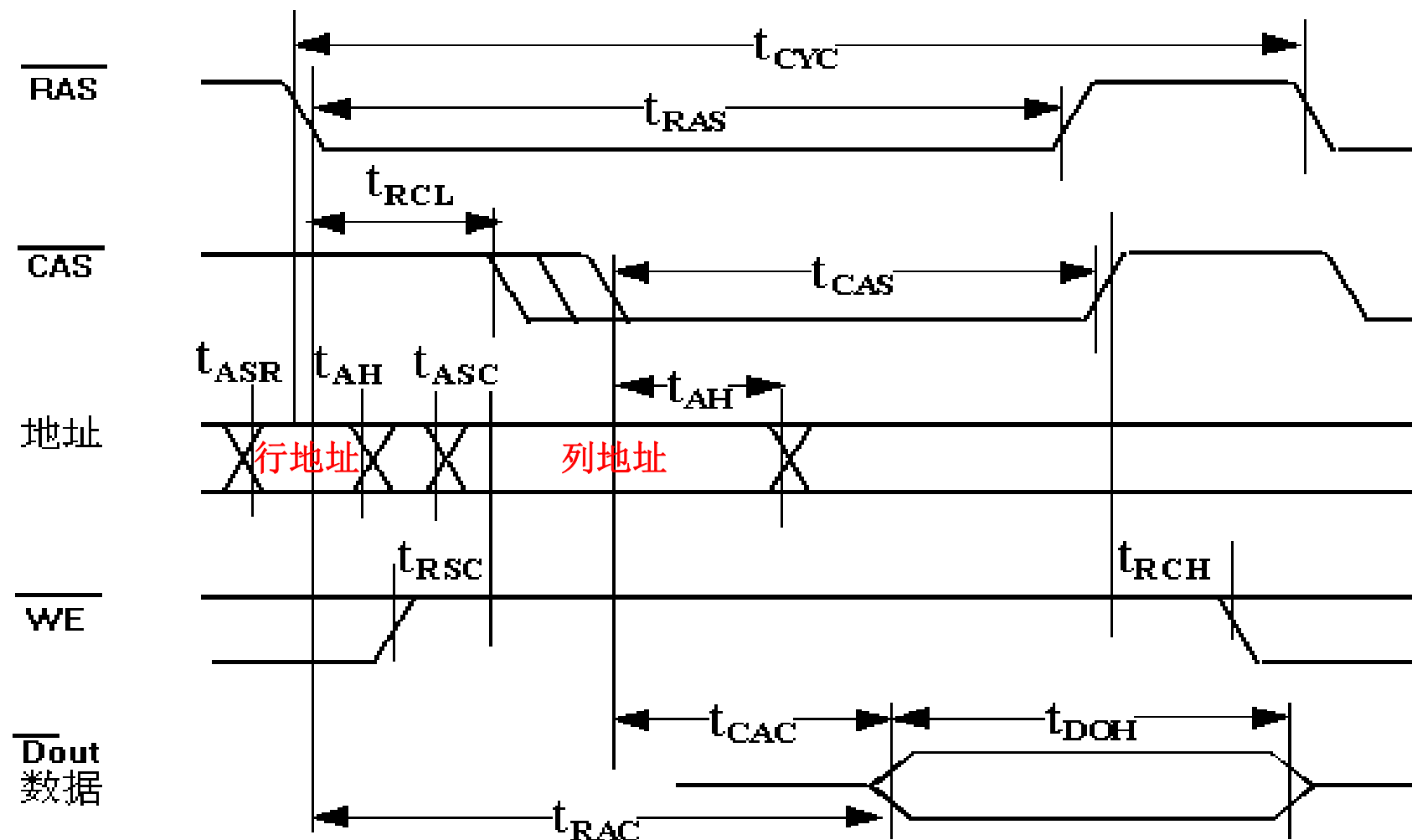
# 动态存储器框图



(b) 逻辑结构图



# DRAM时序——读周期

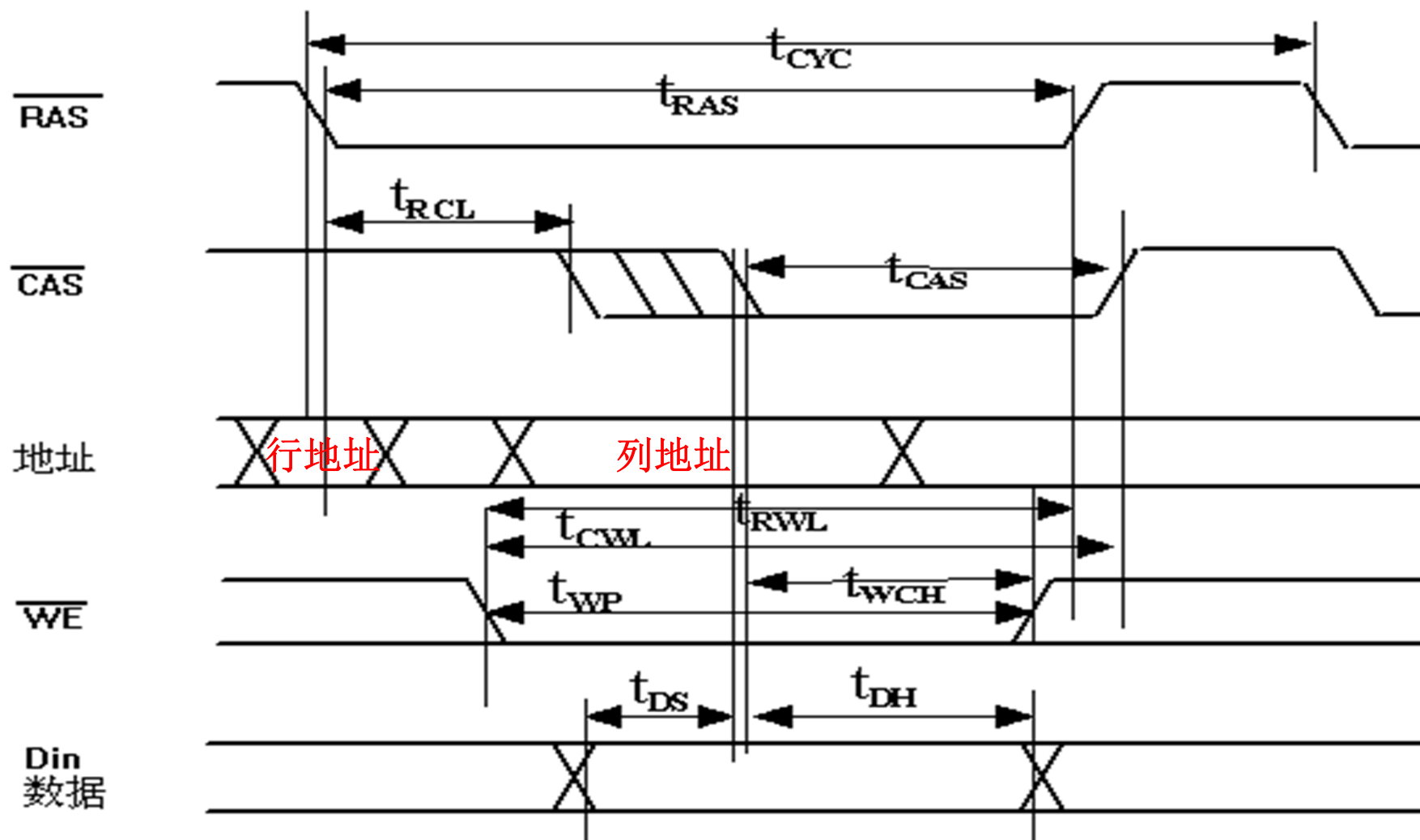


行地址有效→行地址选通→列地址有效→列地址选通→数据输出→行选通、列选通及地址撤销





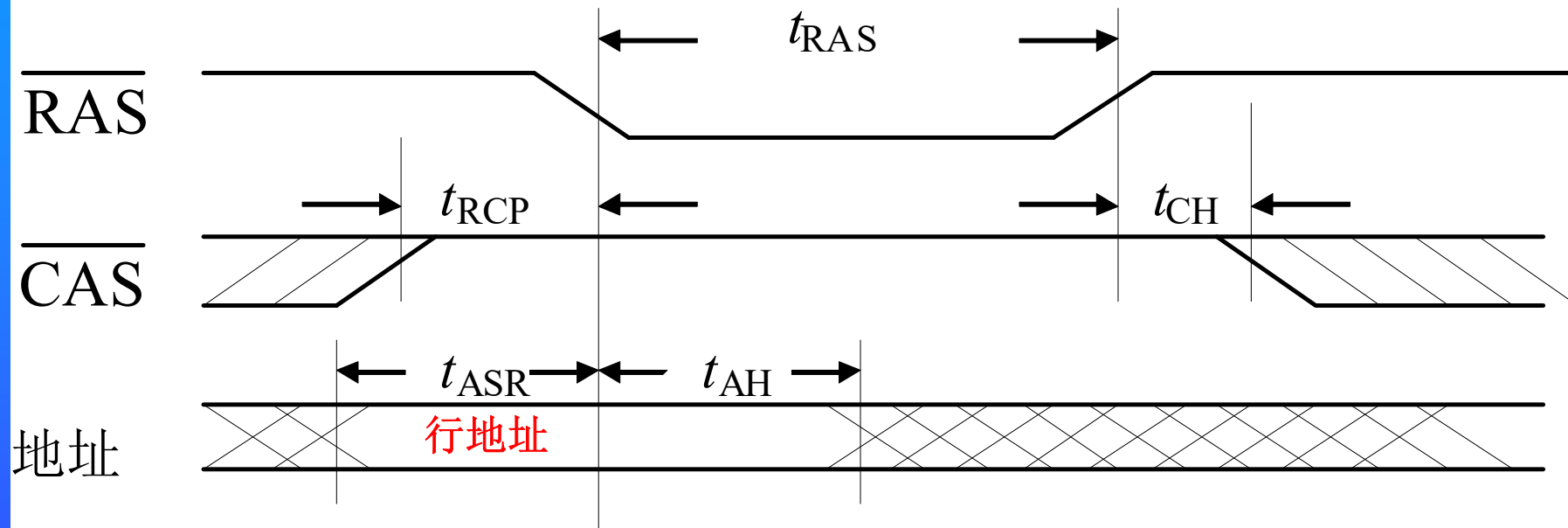
# DRAM时序——写周期



行地址有效→行地址选通→列地址、数据有效→列地址选通→数据输入→行选通、列选通及地址撤销



# DRAM时序——刷新周期

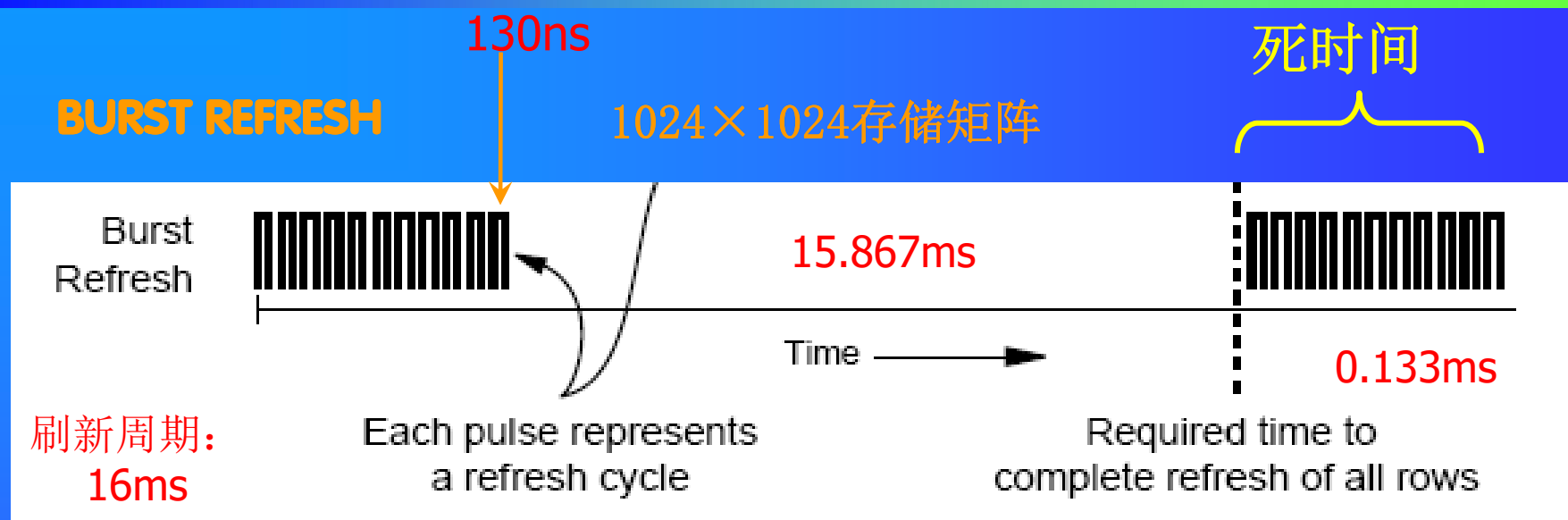


(a) 只用 RAS\*的刷新

nRAS only: 刷新行地址有效→nRAS有效→刷新行地址和nRAS撤销



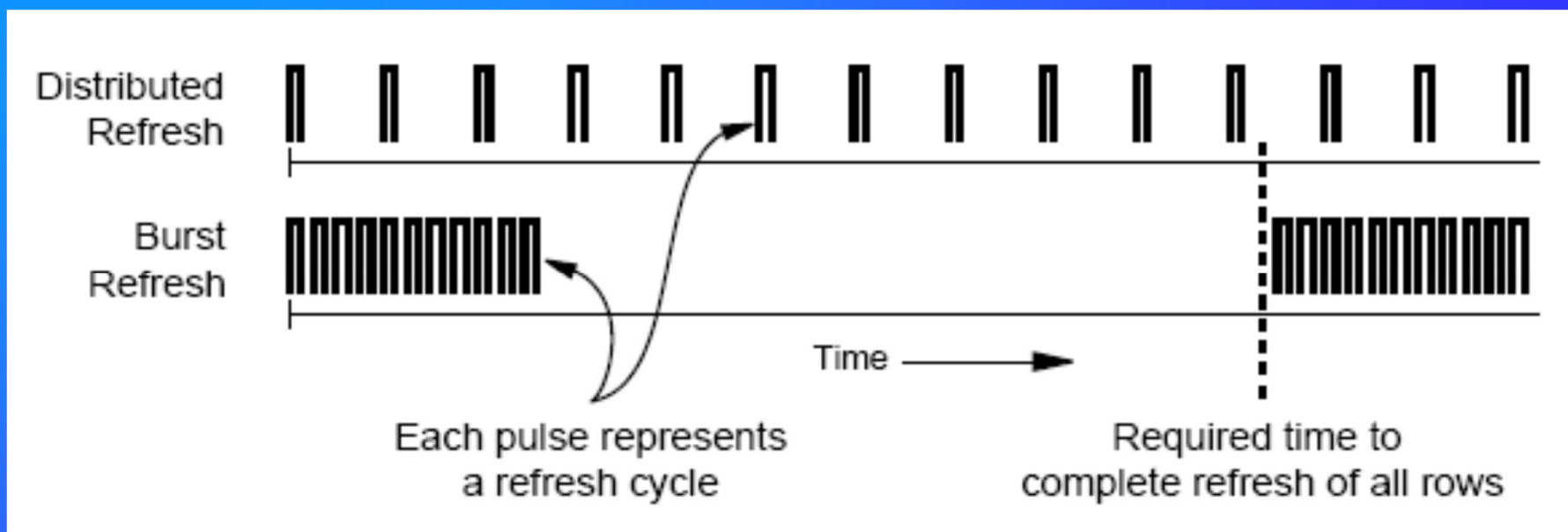
# DRAM的刷新策略：集中式刷新



- 刷新占用时间:  $1024 \text{ cycles} \times 130\text{ns} = 133,120\text{ns} = 0.133\text{ms}$
- 正常读/写/维持时间:  $16\text{ms} - 0.133\text{ms} = 15.867\text{ms}$
- 优点: 存储器系统的平均读写周期接近于存储器件的读写周期
- 缺点: 在刷新的过程中不允许读写, 存在“死时间”
  - ❑ 死时间率  $D = \text{刷新时间} / \text{总时间} = 0.133 / 16 = 0.83\%$



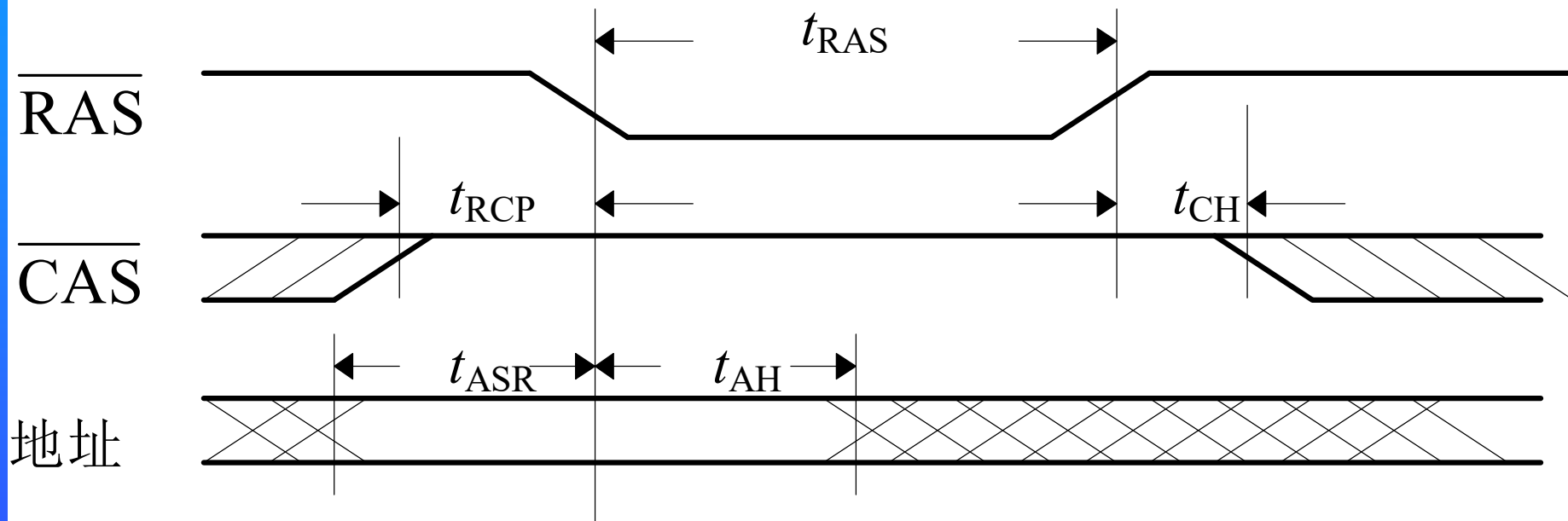
# DRAM的刷新策略：分散式刷新



- $16\text{ms}/1024 = 0.015625\text{ms} = 15.6\mu\text{s}$
- perform a refresh cycle every  $15.6\mu\text{s}$



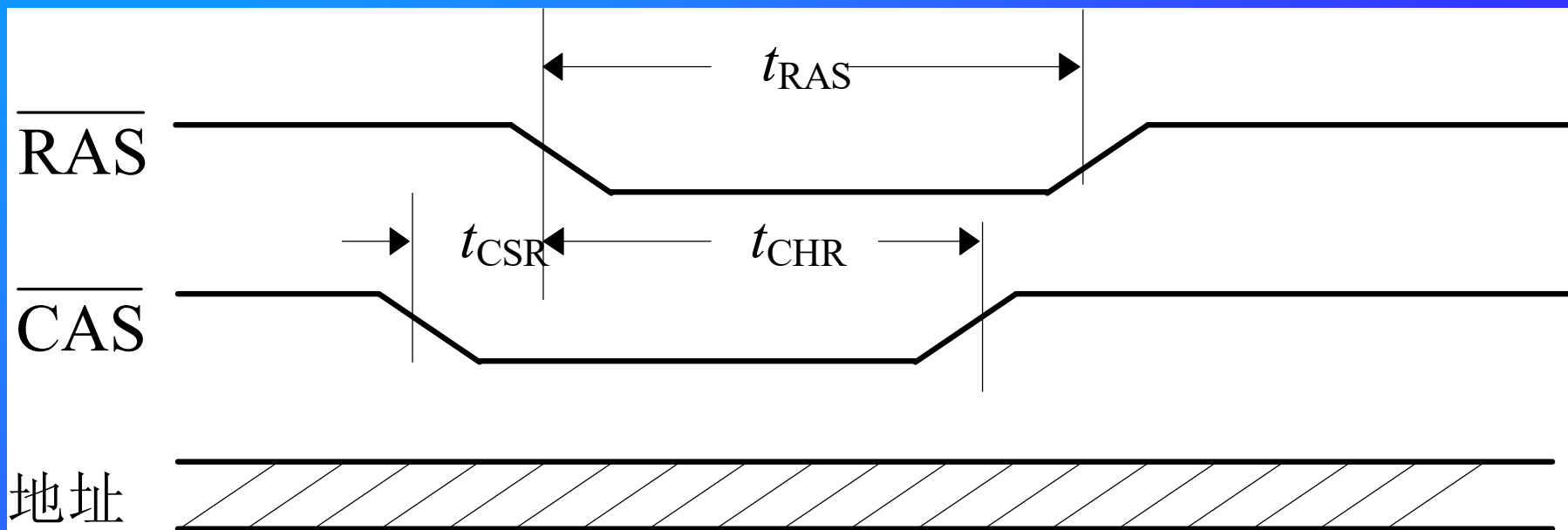
# DRAM的刷新控制方式



只用nRAS信号的刷新模式



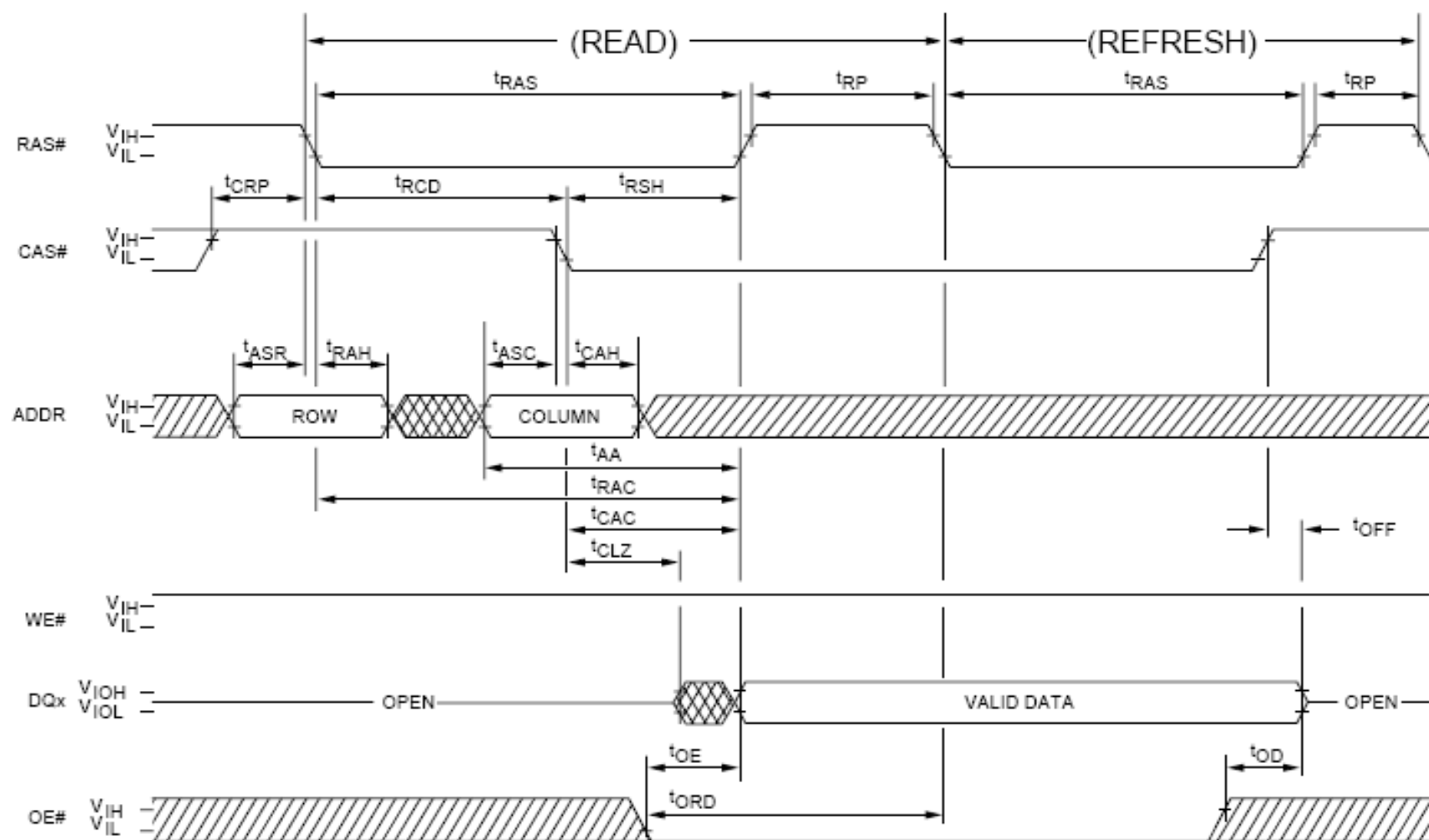
# DRAM的刷新控制方式



nCAS在nRAS之前的刷新（自动刷新）模式  
CBR (nCAS-BEFORE-nRAS) REFRESH



# DRAM的刷新控制方式



隐含式刷新



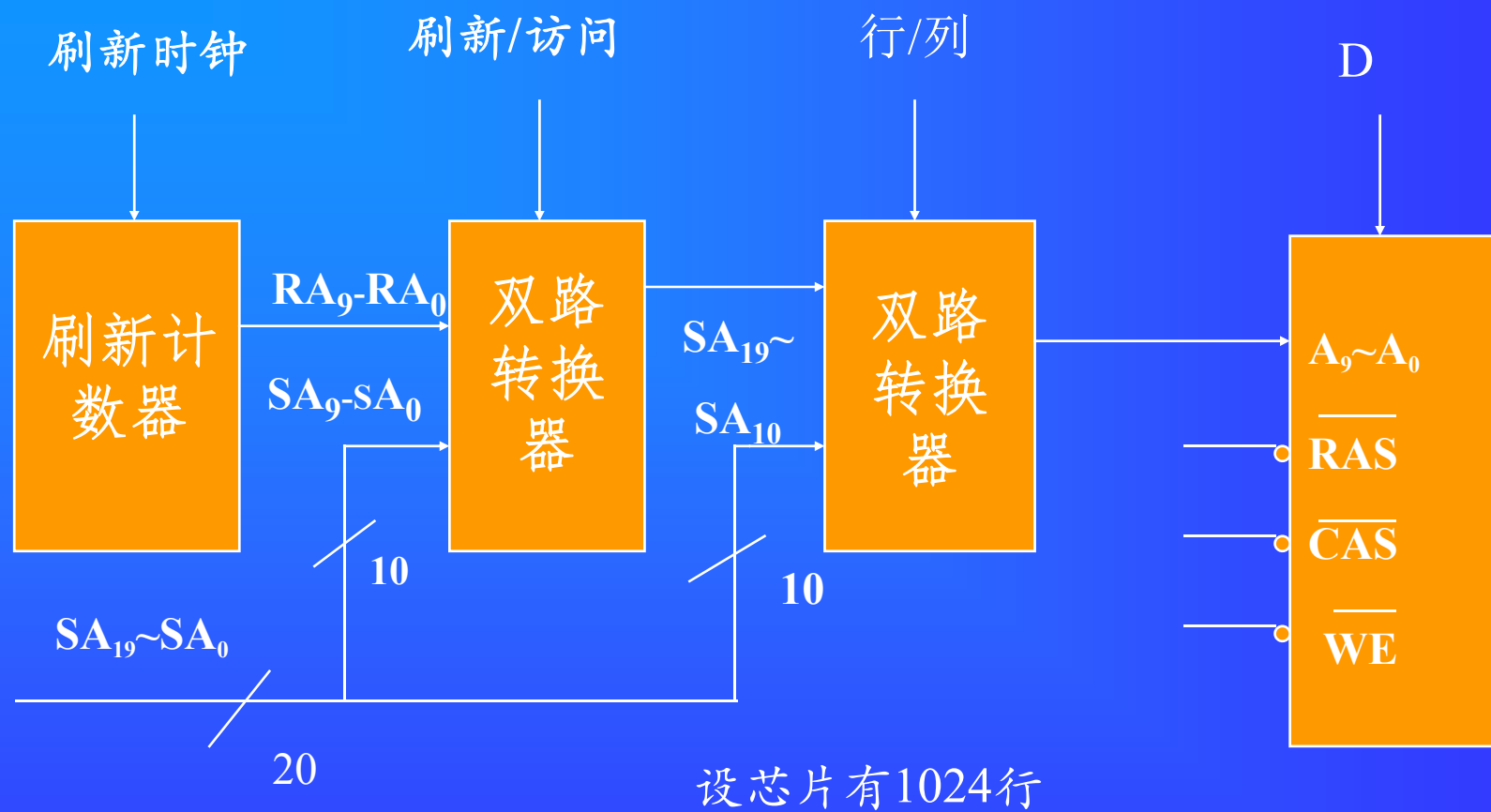
北京邮电大学

计算机学院

2021/3/30

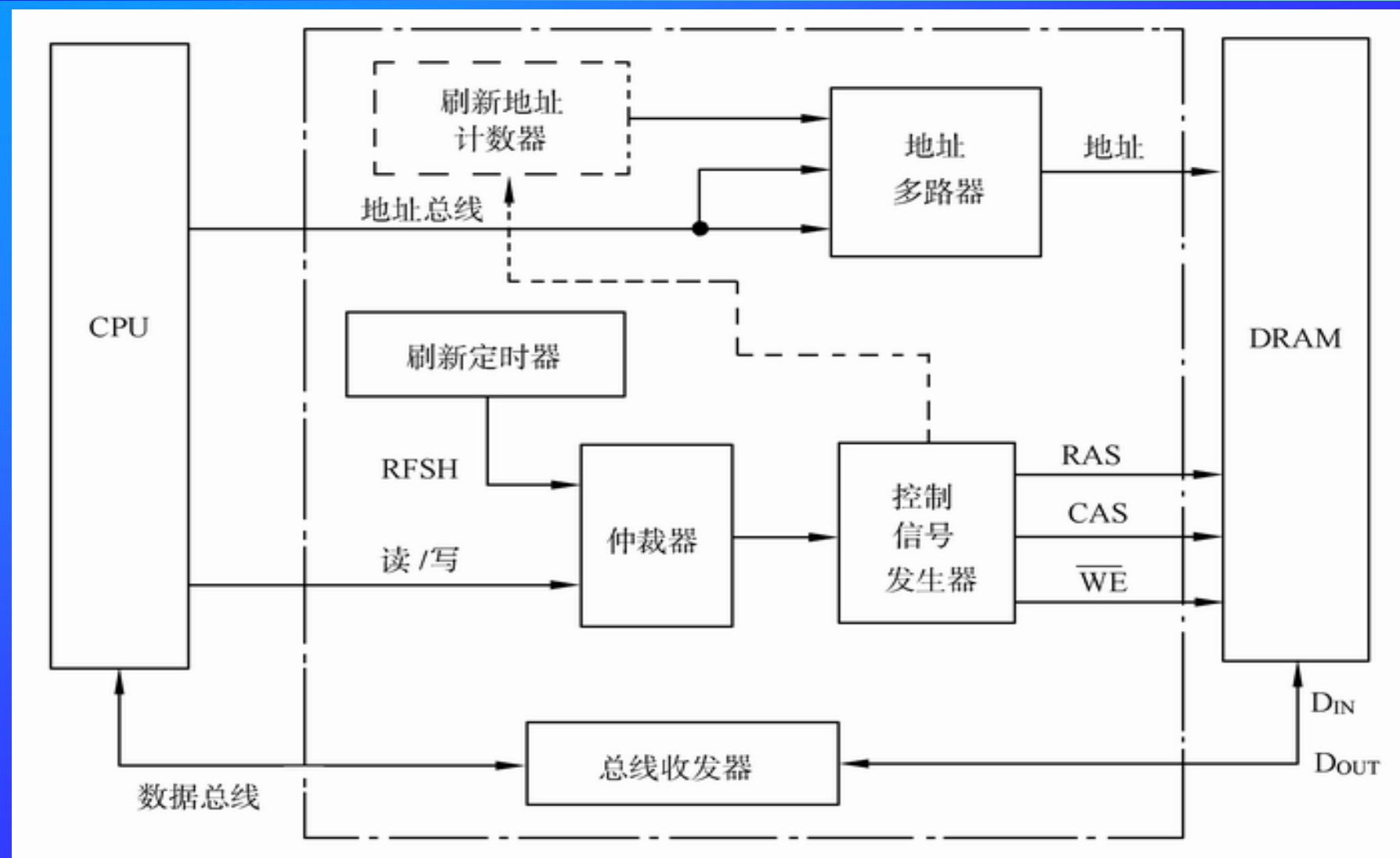
93

# DRAM存储器与CPU的连接



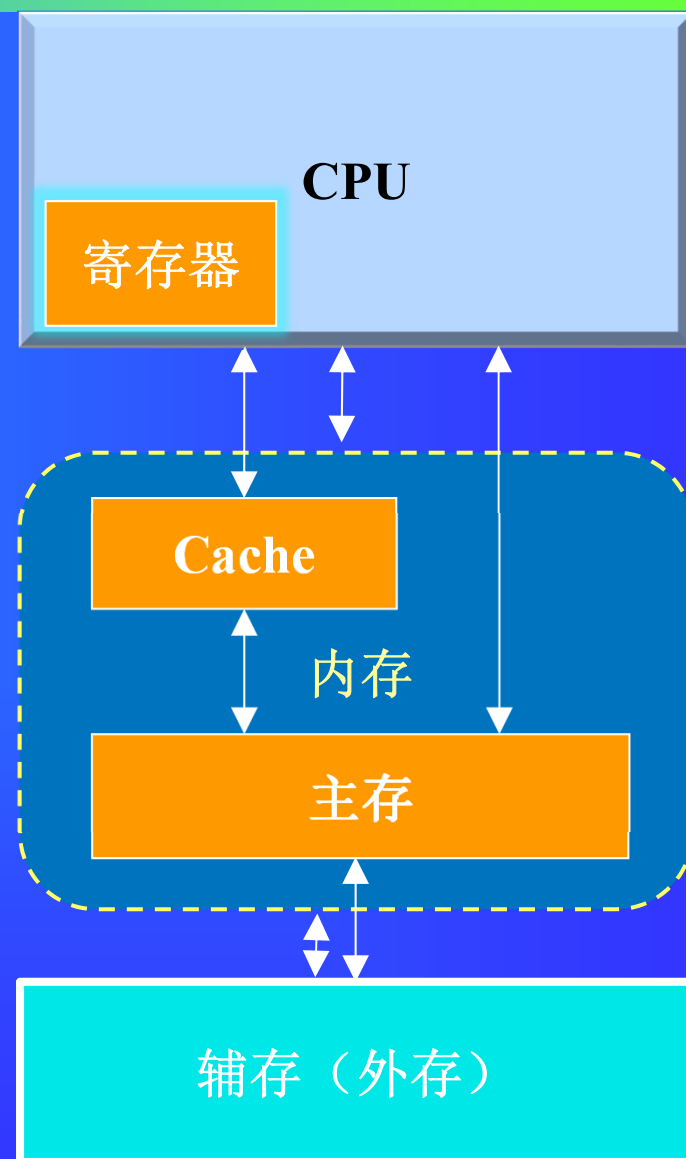


# DRAM控制器

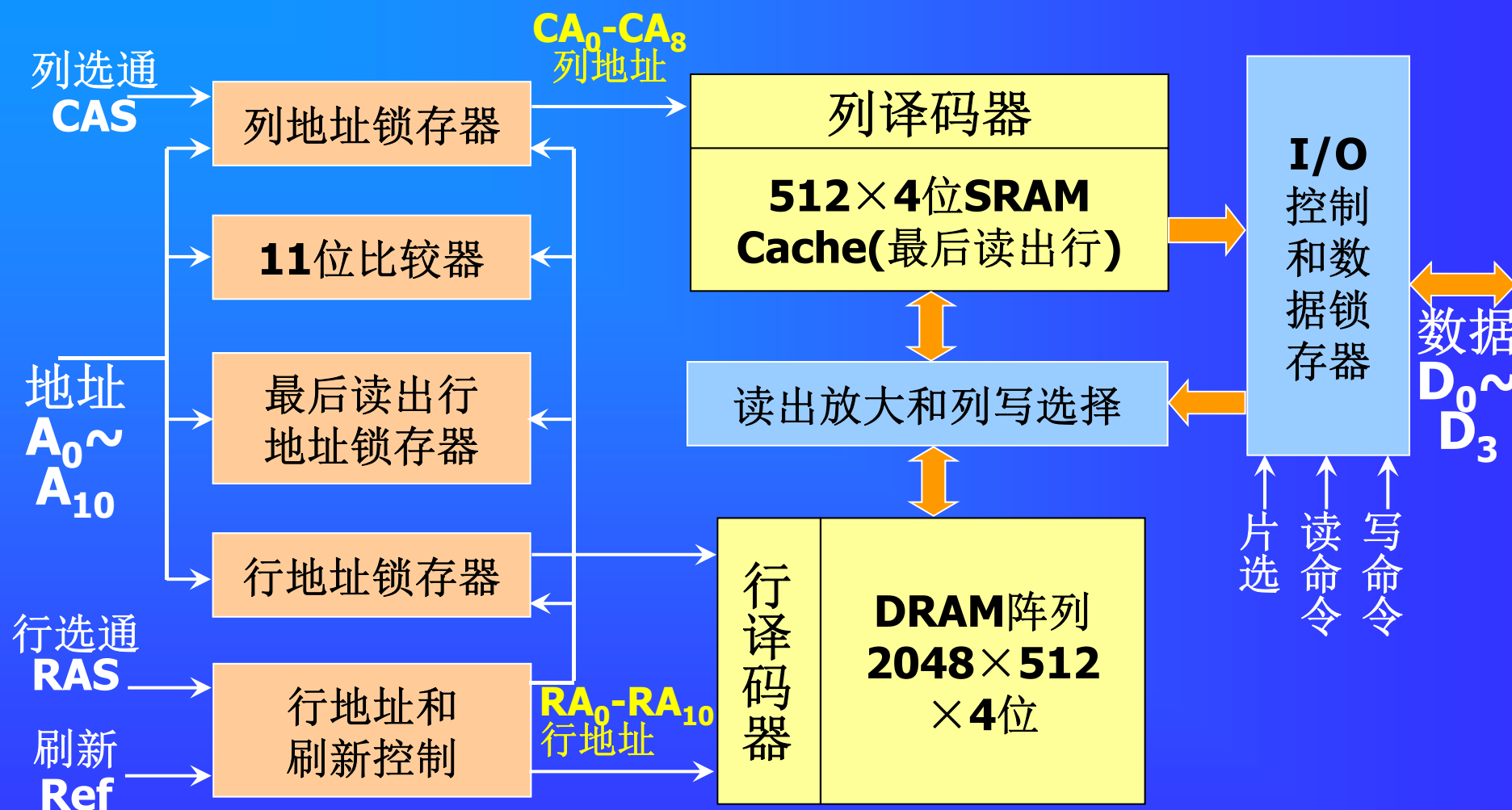


# 缓冲DRAM (Cached DRAM)

- 在DRAM芯片内部增加SRAM构成“Cache”，保存最近访问的一行
- 增加行地址比较器：如果下次访问的是最近访问过的行，则直接从Cache中读出

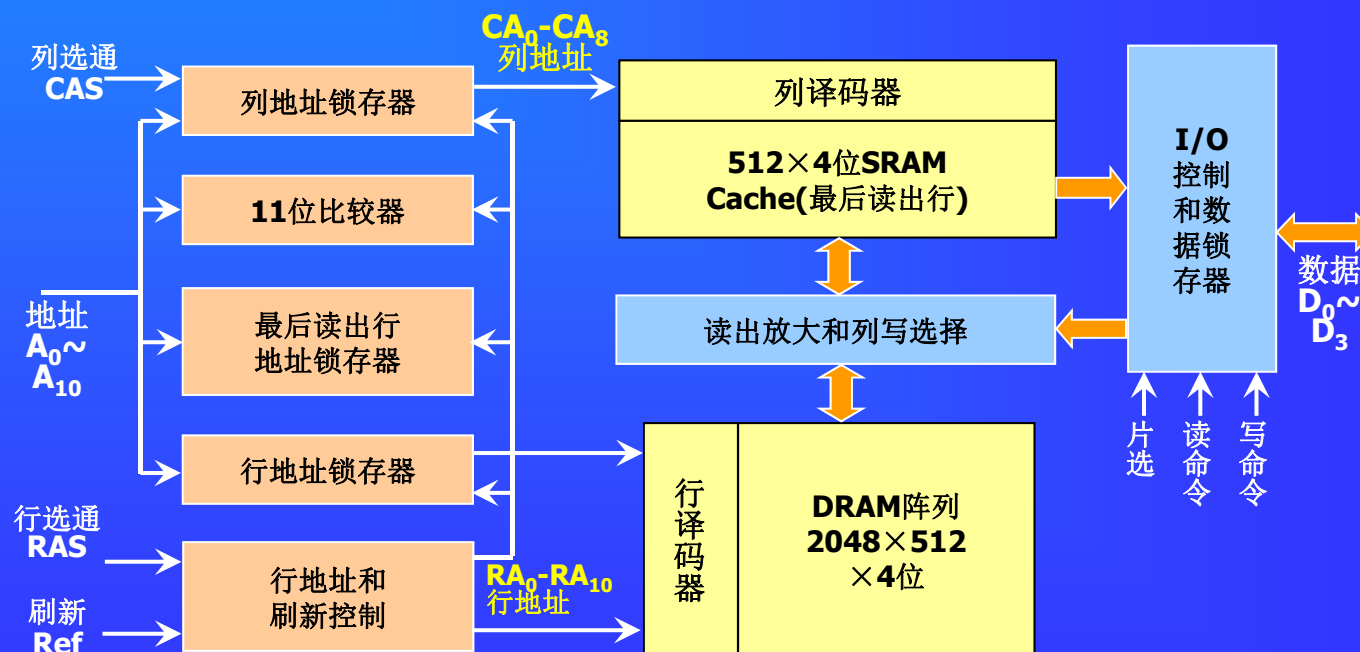


# 1M × 4bit CDRAM

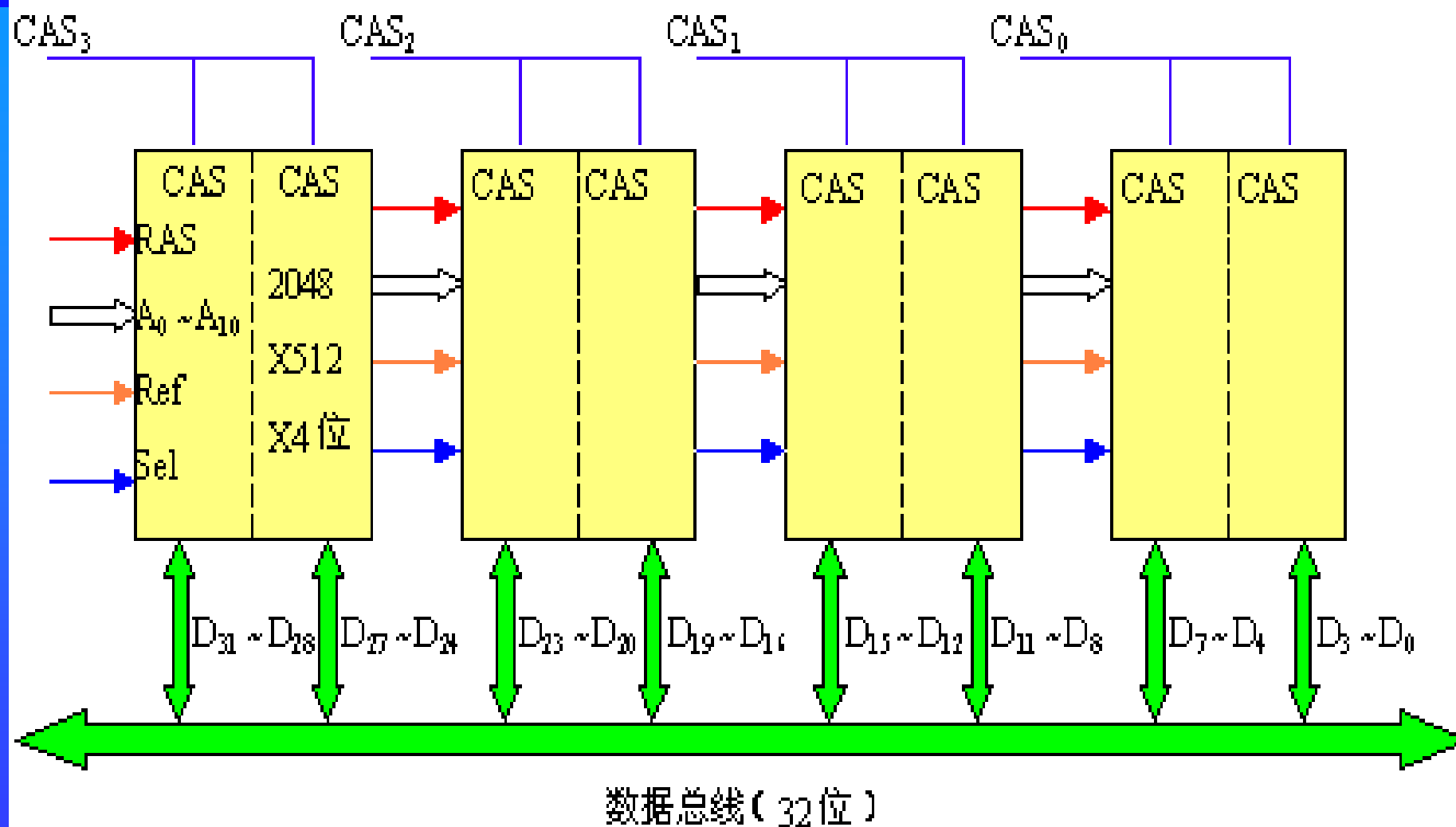


# CDRAM (cached DRAM) 的优点

- 高速突发 (burst) 传送
- Cache读出的同时进行刷新操作
- 读、写操作可并行



# 1M × 32位 CDRAM内存条

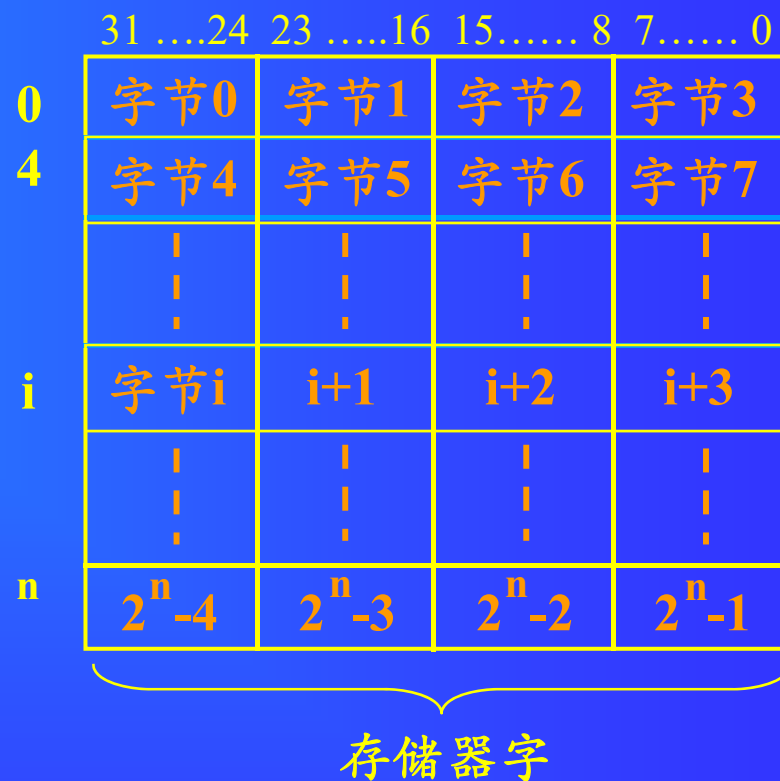


# 存储器的寻址宽度



➤每次访问存储器读写的信息量:

- 字寻址: 每次访存读写一个存储字
- 半字寻址: 每次访存读写半个存储字
- 字节寻址: 每次访存读写一个字节



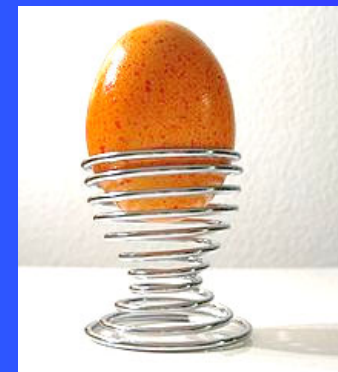
字长=4字节



# Big Endian and Little Endian

## ➤ 端模式 (Endian Mode)

- ❑ Jonathan Swift: 格利佛游记
- ❑ 多字节数据的字节顺序



## ➤ 含义

An egg in an egg cup with the little-endian portion oriented upward

### ❑ 大端 (Big-Endian)

- ✉ 高有效字节放在内存的低地址端，低有效字节放在内存的高地址端

### ❑ 小端 (Little-Endian)

- ✉ 低有效字节放在内存的低地址端，高有效字节放在内存的高地址端



# Big Endian and Little Endian

## ➤ 含义

- ❑ 大端 (Big-Endian) : 高字节在低地址, 低字节在高地址
- ❑ 小端 (Little-Endian) : 低字节在低地址, 高字节在高地址

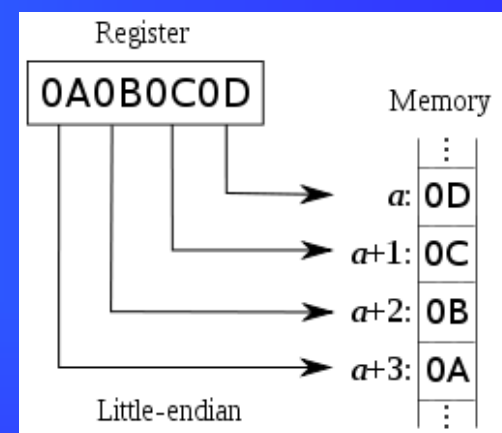
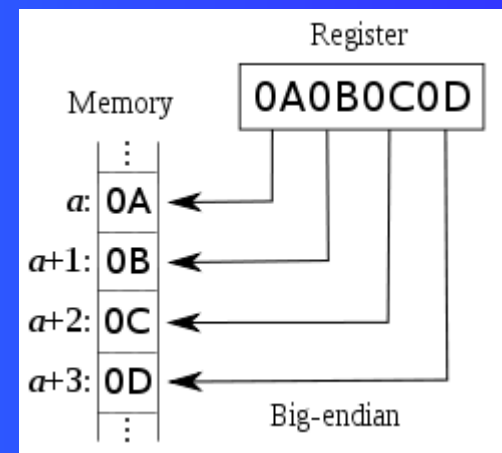
## ➤ 例: 数字 0x0A0B0C0D 在内存中

### ❑ 大端:

☒ 低地址 -----> 高地址  
0x0A | 0x0B | 0x0C | 0x0D

### ❑ 小端:

☒ 低地址 -----> 高地址  
0x0D | 0x0C | 0x0B | 0x0A

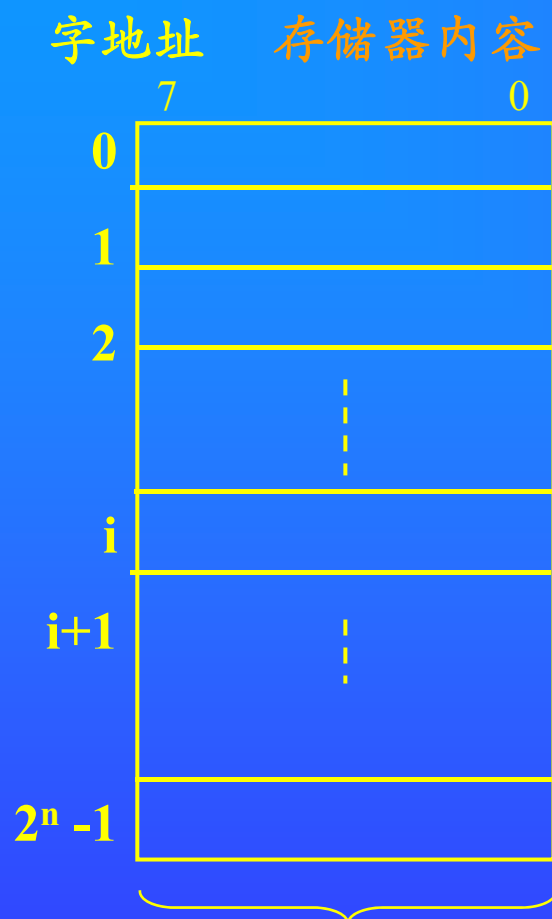


Mapping registers to memory locations



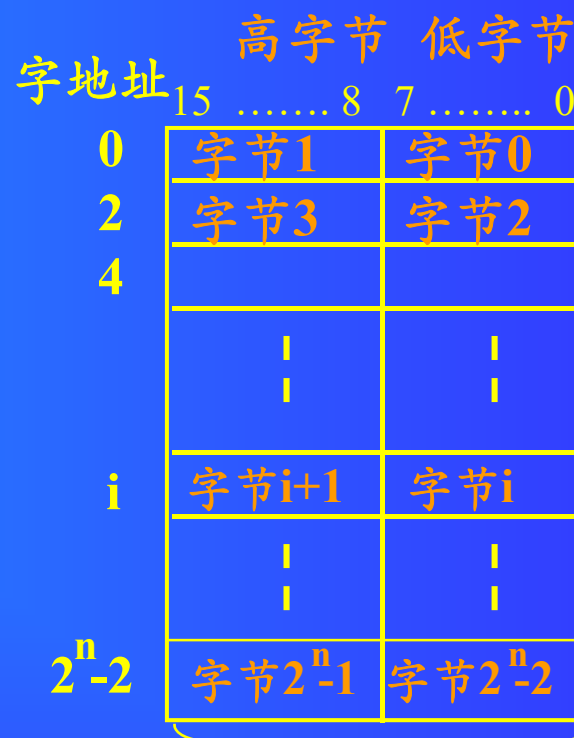


# 总线宽度与字节地址



存储器字

字长=1字节



存储器字

低字节——低地址

高字节——高地址

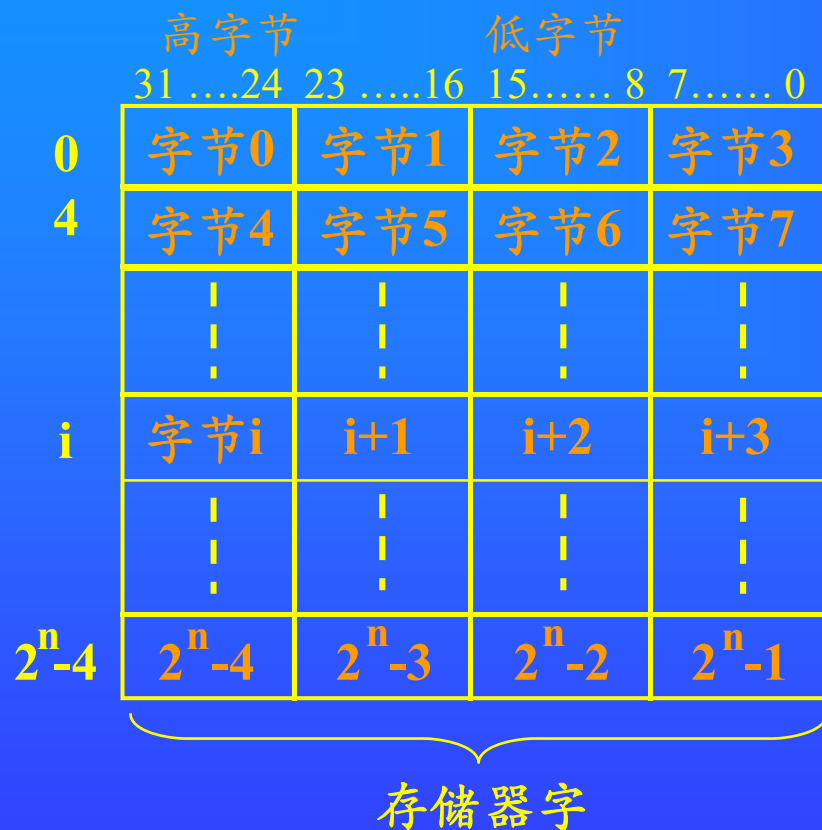
字地址=低字节地址

字长=2字节

小端  
Little Endian



# 总线宽度与字节地址



字长=4字节

低字节——高地址

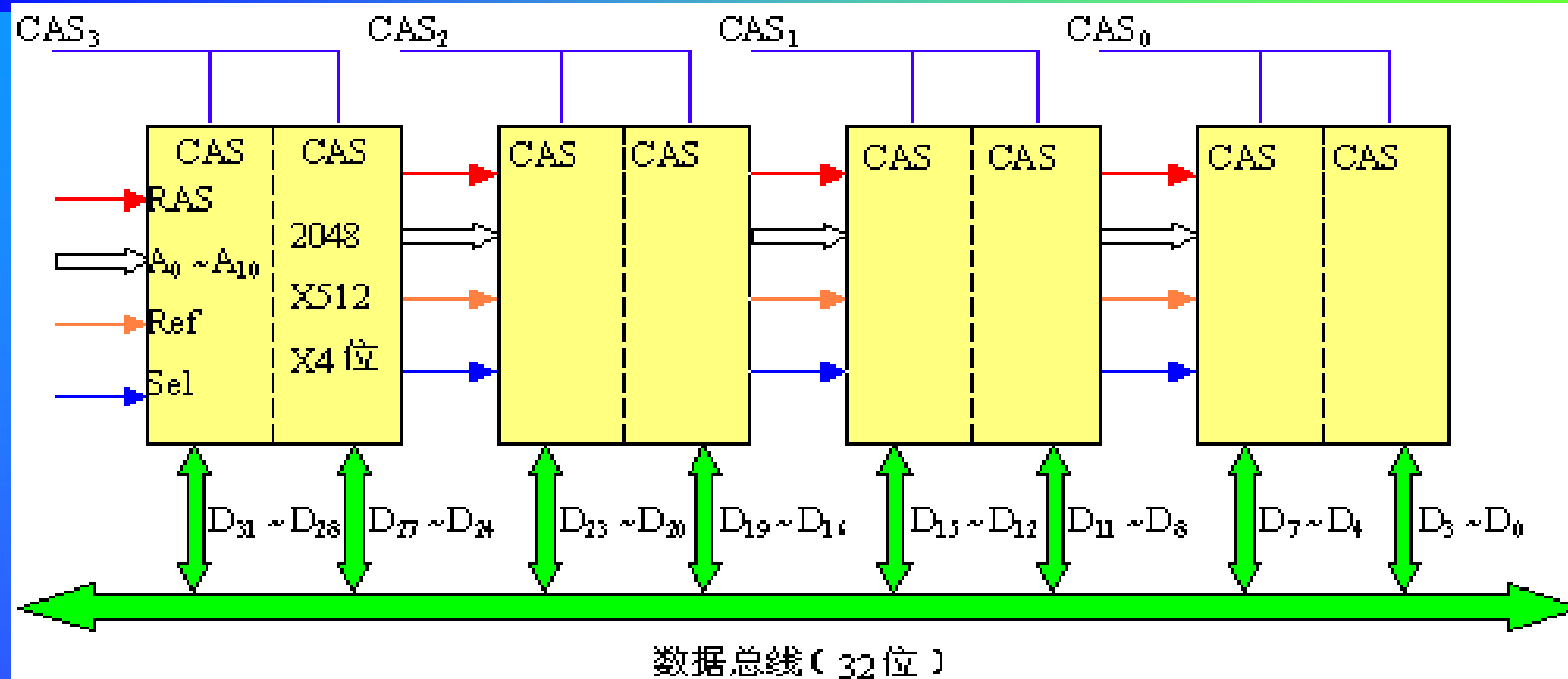
高字节——低地址

字地址=高字节地址

大端  
Big Endian



# 1M × 32位 CDRAM内存条



# CDRAM内存条地址分配

字地址	$/CAS_3 = /BE_3$	$/CAS_2 = /BE_2$	$/CAS_1 = /BE_1$	$/CAS_0 = /BE_0$
000000	000011	000010	000001	000000
000100	000111	000110	000101	000100
001000	001011	001010	001001	001000
001100	001111	001110	001101	001100
010000	010011	010010	010001	010000

- 小端Little Endian: 低字节放在低地址，高字节放在高地址
- 32位总线，按字节编址

➤BE=byte enable



# 课堂练习：单项选择题

某计算机存储器按字节编址，采用小端方式存放数据，假定编译器规定int型和short型长度分别为32位和16位，并且数据按边界对齐存储。某C语言程序段如下：

```
struct{  
    int  a;  
    char b;  
    short c;  
} record;  
record.a=273;
```

若record变量的首地址为0xC008，则地址0xC008中内容及record.c的地址分别为

A. 0x00、0xC00D

B. 0x00、0xC00E

C. 0x11、0xC00D

D. 0x11、0xC00E



# 课堂练习：单项选择题

存储器按字节编址，小端方式，int型和short型长：32位和16位，数据按边界对齐存储。record变量的首地址为0xC008，则地址0xC008 中内容及record.c的地址分别为

- A. 0x00、0xC00D    B. 0x00、0xC00E  
C. 0x11、0xC00D    D. 0x11、0xC00E

```
struct{ int  a;  
        char b;  
        short c;  
} record;  
record.a=273;
```

C008	11	a
C009	01	
C00A	00	
C00B	00	b
C00C		
C00D	-	
C00E		c
C00F		
C010		

【答】

- ✓  $273 = 256 + 16 + 1 = 1\ 0001\ 0001B = 0000\ 0111$
- ✓ 小端：高地址存高半字/字节；低地址存低半字/字节
- ✓ 0xC008 中内容:0x11, record.c的地址: 0xC00E
- ✓ 故答案为D



# 【 DRAM存储器的发展 】

## （第三章第二部分）



# Nonvolatile Memories

- **Masked ROM (ROM)**
  - ❑ Data are written permanently during the manufacturing
  - ❑ High density and low cost
- **One Time Programmable ROMs (OTP ROM, PROM)**
  - ❑ User programmable only once, Flexible and low cost
- **UV Erasable PROM (EPROM)**
  - ❑ User programmable multiple times, Flexible, erased by UV
- **Electrically Erasable PROM (E<sup>2</sup>PROM or EEPROM)**
  - ❑ Erased by high voltage
- **Flash Memories**
  - ❑ Block/page erasable by regular supply voltage





# 课堂练习



CPU的地址总线A15-A0，数据总线D7-D0，访存允许信号 $\overline{\text{MREQ}}$ ，读写选择信号 $\text{R}/\overline{\text{W}}$ 。

➤ 主存地址空间分配：（十进制地址，按字节编址）

❑ 0-8191为系统程序区，由ROM芯片组成；

❑ 8192-32767为用户程序区；

❑ 最后2K地址空间为系统程序工作区

➤ 现有如下存储器芯片：

❑ EPROM：8K×8位(控制端仅有 $\overline{\text{CS}}$ )；

❑ SRAM：16K×1位，2K×8位，4K×8位，8K×8位。

➤ 请从上述芯片中选择适当芯片设计该计算机主存储器，画出主存储器逻辑框图。说明选哪些存储器芯片，选多少片。



# 课堂练习

解：根据给定条件，选用：

EPROM: 8K × 8位芯片1片

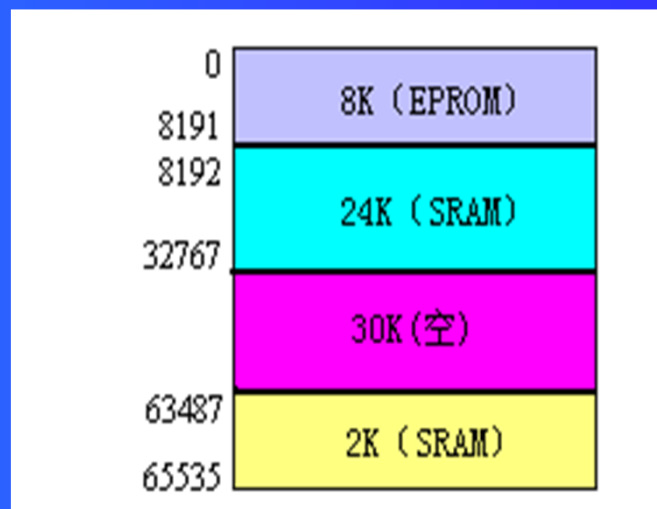
SRAM: 8K × 8位芯片3片  
2K × 8位芯片1片

地址区间：

0 – 8191 = 0000 0000 0000 0000 – 0001 1111 1111 1111

8192 – 32767 = 0010 0000 0000 0000 – 0111 1111 1111 1111

63488 – 65535 = 1111 1000 0000 0000 – 1111 1111 1111 1111



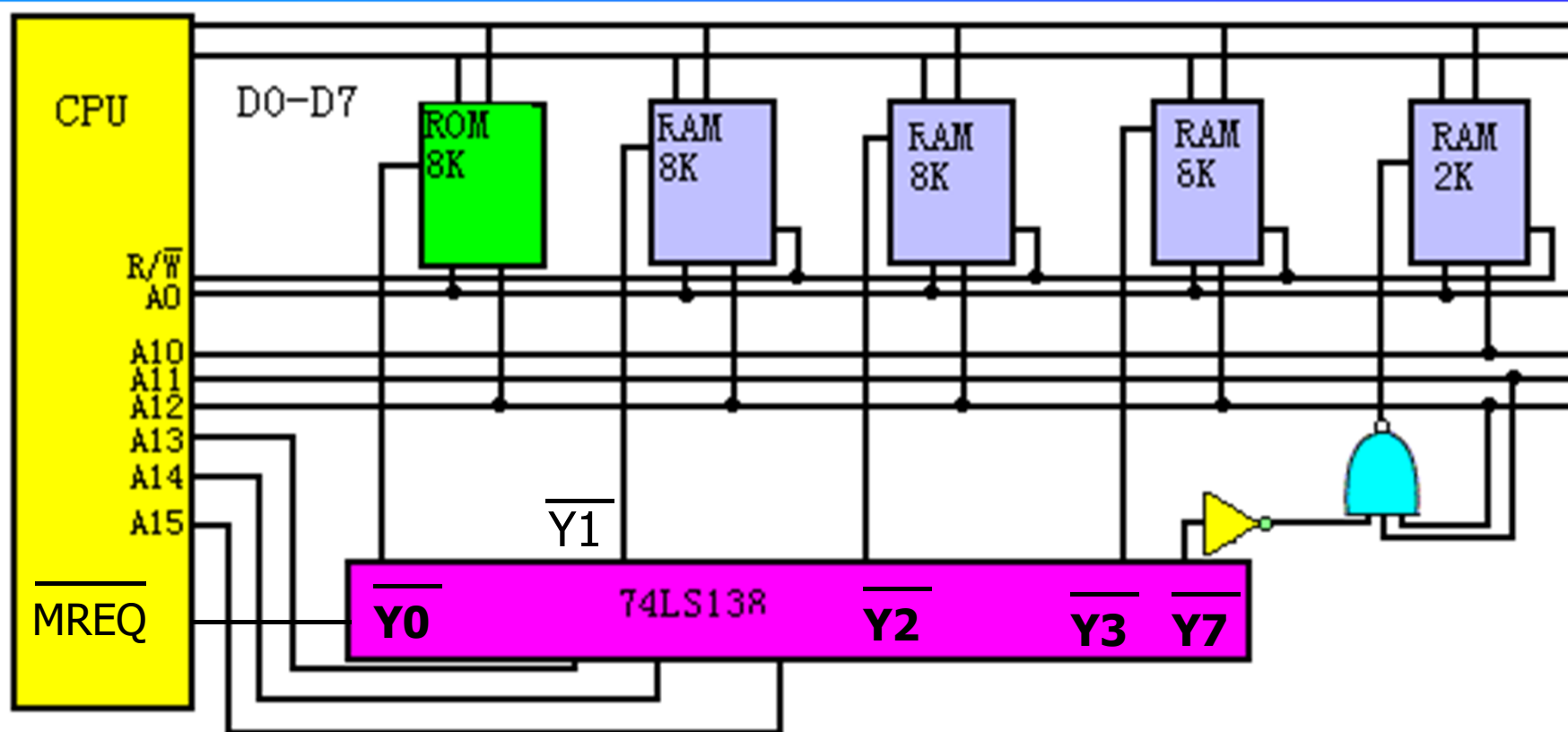
选择EPROM，用3:8译码器的Y0输出端

选择3片8K × 8位SRAM，用Y1、Y2、Y3输出端

选择2K × 8位SRAM时，用Y7输出端以及A11和A12地址线



# 课堂练习



# 闪速 (flash) 存储器概述

- 二十世纪八十年代出现、九十年代中后期开始应用
- 电可擦、非易失性只读存储器

## Connecting



Mobile Phone



Screen Phone



Smart Mobile Phone



PDA



Auto PC



Palm PC



Set Top Box



Handheld PC



Thin Client



Handheld Game



Game Console



Net TV



DSC



Audio recorder



MP3

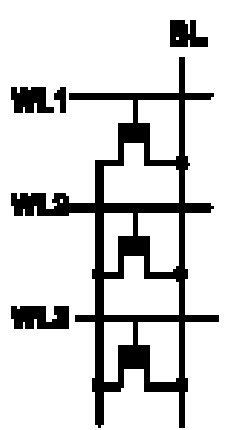
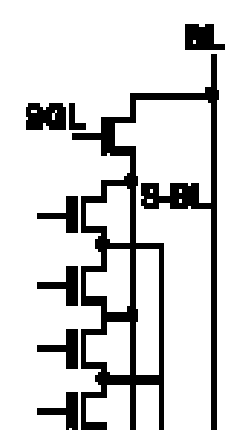
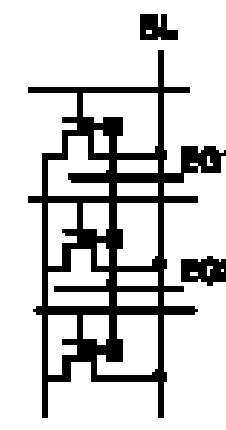
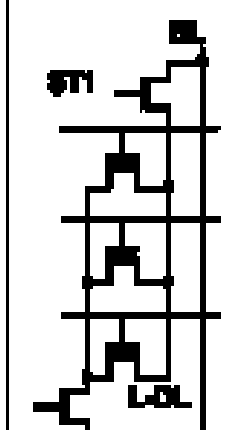
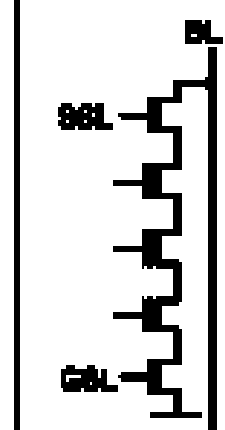
## Consumer

## Computing



# 闪速存储器技术分类



Technology	<u>NOR</u>	DINOR	T-Poly	AND	<u>NAND</u>
Structure					
Program Method	CHE	F-N	CHE	F-N	F-N
Erase Method	F-N	F-N	F-N	F-N	F-N
Layers	2P2M	3P2M	3P1M	3P2M	2P1M
Company	Intel, AMD	Mitsubishi	SanDisk	Hitachi	Samsung Toshiba

Reference : ISSCC 94, 95, 96 Flash Session



# 两种主要的闪存技术：线性闪存（NOR Flash）

- 能快速随机读取，读取速度高
- execute-in-place (XIP)
- 单字编程
- 在对存储器进行重新编程之前需要对区或整片进行擦除操作
  - ❑ 以区块（sector）或芯片为单位执行擦除操作
- 接口方式：拥有独立的数据总线和地址总线
- 信息存储可靠性高
- 适用于擦除和编程操作较少而直接执行代码的场合，尤其是纯代码存储应用
- 不适用于纯数据存储和文件存储的应用
  - ❑ 擦除和编程速度较慢
  - ❑ 区块尺寸较大



## 两种主要的闪存技术：非线性闪存（NAND Flash）

- 以页为单位进行读和编程操作
- 以块（block）为单位进行擦除操作
- 快速编程和快速擦除
- 接口方式：数据、地址采用同一总线
- 位成本低、位密度高
- 较高的比特错误率，需软件处理坏块（失效块）
- 10倍于NOR flash的擦除次数
- 适用于大容量存储设备：存储卡、固态硬盘
  - ❑ 增加NAND FLASH控制器后也可用于程序存储



# 线性闪存

## NOR FLASH MEMORY



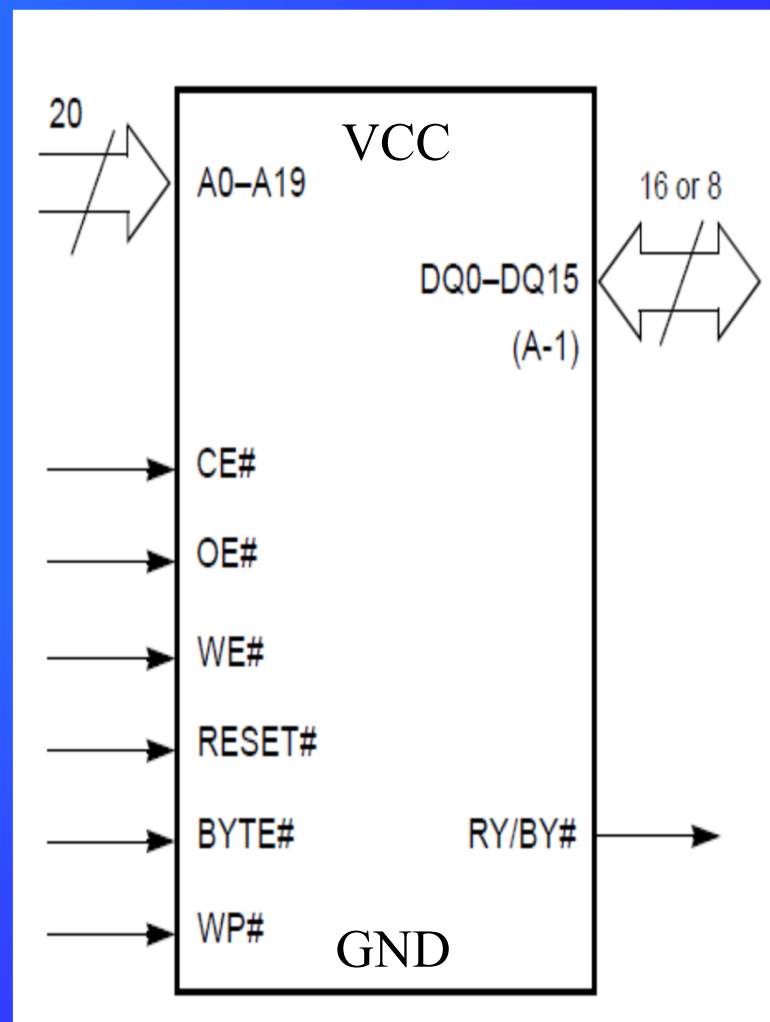


# S29AL016J——2M × 8/1M × 16 NOR闪存

➤ 赛普拉斯公司（原AMD/Spansion）

➤ 接口信号

- ☐ A19~A0: 20位地址
- ☐ DQ0 ~ DQ15: 16位数据输入/输出（字方式）
- ☐ CE#: 片选
- ☐ WE#: 写使能
- ☐ OE#: 输出允许
- ☐ BYTE#: 8bit/16bit选择
- ☐ RESET#: 硬件复位线
- ☐ RY/BY#: Ready/Busy#
- ☐ VCC = 3.3 volt



# S29AL016J 的组织结构



- 外部按 $2M \times 8$ 或 $1M \times 16$ 比特组织
- 顶部引导区和底部引导区两种类型
- 内部组织为若干个区块

□ 底部引导区区块划分如表

区块	$A_{19}$	$A_{18}$	$A_{17}$	$A_{16}$	$A_{15}$	$A_{14}$	$A_{13}$	$A_{12}$	区块大小 (K字节/K字)	地址范围 (16进制)	
										字节模式 (x8)	字模式 (x16)
SA0	0	0	0	0	0	0	0	X	16/8	00000-003FFF	00000-01FFF
SA1	0	0	0	0	0	0	0	1	8/4	004000-005FFF	02000-02FFF
SA2	0	0	0	0	0	0	0	1	8/4	006000-007FFF	03000-03FFF
SA3	0	0	0	0	0	0	1	X	32/16	008000-00FFFF	04000-07FFF
SA4	0	0	0	0	0	1	X	X	64/32	010000-01FFFF	08000-0FFFF
SA5	0	0	0	0	1	0	X	X	64/32	020000-02FFFF	10000-17FFF
SA6	0	0	0	0	1	1	X	X	64/32	030000-03FFFF	18000-1FFFF
SA7	0	0	1	0	0	0	X	X	64/32	040000-04FFFF	20000-27FFF
SA8	0	0	1	0	1	0	X	X	64/32	050000-05FFFF	28000-2FFFF
SA9	0	0	1	1	0	0	X	X	64/32	060000-06FFFF	30000-37FFF
SA10	0	0	1	1	1	0	X	X	64/32	070000-07FFFF	38000-3FFFF
SA11	0	1	0	0	0	0	X	X	64/32	080000-08FFFF	40000-47FFF
SA12	0	1	0	0	1	0	X	X	64/32	090000-09FFFF	48000-4FFFF
SA13	0	1	0	1	0	0	X	X	64/32	0A0000-0AFFFF	50000-57FFF
SA14	0	1	0	1	1	0	X	X	64/32	0B0000-0BFFFF	58000-5FFFF
SA15	0	1	1	0	0	0	X	X	64/32	0C0000-0CFFFF	60000-67FFF
SA16	0	1	1	0	1	0	X	X	64/32	0D0000-0DFFFF	68000-6FFFF
SA17	0	1	1	1	0	0	X	X	64/32	0E0000-0EFFFF	70000-77FFF
SA18	0	1	1	1	1	0	X	X	64/32	0F0000-0FFFFF	78000-7FFFF
SA19	1	0	0	0	0	0	X	X	64/32	100000-10FFFF	80000-87FFF
SA20	1	0	0	0	1	0	X	X	64/32	110000-11FFFF	88000-8FFFF
SA21	1	0	0	1	0	0	X	X	64/32	120000-12FFFF	90000-97FFF
SA22	1	0	0	1	1	0	X	X	64/32	130000-13FFFF	98000-9FFFF
SA23	1	0	1	0	0	0	X	X	64/32	140000-14FFFF	A0000-A7FFF
SA24	1	0	1	0	1	0	X	X	64/32	150000-15FFFF	A8000-AFFFF
SA25	1	0	1	1	0	0	X	X	64/32	160000-16FFFF	B0000-B7FFF
SA26	1	0	1	1	1	0	X	X	64/32	170000-17FFFF	B8000-BFFFF
SA27	1	1	0	0	0	0	X	X	64/32	180000-18FFFF	C0000-C7FFF
SA28	1	1	0	0	1	0	X	X	64/32	190000-19FFFF	C8000-CFFFF
SA29	1	1	0	1	0	0	X	X	64/32	1A0000-1AFFFF	D0000-D7FFF
SA30	1	1	0	1	1	0	X	X	64/32	1B0000-1BFFFF	D8000-DFFFF
SA31	1	1	1	0	0	0	X	X	64/32	1C0000-1CFFFF	E0000-E7FFF
SA32	1	1	1	0	1	0	X	X	64/32	1D0000-1DFFFF	E8000-EFFFF
SA33	1	1	1	1	0	0	X	X	64/32	1E0000-1EFFFF	F0000-F7FFF
SA34	1	1	1	1	1	0	X	X	64/32	1F0000-1FFFFF	F8000-FFFFF



# S29AL016J 的组织结构



区块	$A_9$	$A_{18}$	$A_{17}$	$A_{16}$	$A_{15}$	$A_{14}$	$A_{13}$	$A_{12}$	区块大小 (K字节/K字)	地址范围 (16进制)	
										字节模式 (x8)	字模式 (x16)
SA24	1	0	1	0	1	X	X	X	64/32	150000-15FFFF	A8000-AFFFF
SA25	1	0	1	1	0	X	X	X	64/32	160000-16FFFF	B0000-B7FFF
SA26	1	0	1	1	1	X	X	X	64/32	170000-17FFFF	B8000-BFFFF
SA27	1	1	0	0	0	X	X	X	64/32	180000-18FFFF	C0000-C7FFF
SA28	1	1	0	0	1	X	X	X	64/32	190000-19FFFF	C8000-CFFFF
SA29	1	1	0	1	0	X	X	X	64/32	1A0000-1AFFFF	D0000-D7FFF
SA30	1	1	0	1	1	X	X	X	64/32	1B0000-1BFFFF	D8000-DFFFF
SA31	1	1	1	0	0	X	X	X	64/32	1C0000-1CFFFF	E0000-E7FFF
SA32	1	1	1	0	1	X	X	X	64/32	1D0000-1DFFFF	E8000-EFFFF
SA33	1	1	1	1	0	X	X	X	64/32	1E0000-1EFFFF	F0000-F7FFF
SA34	1	1	1	1	1	X	X	X	64/32	1F0000-1FFFFF	F8000-FFFFF



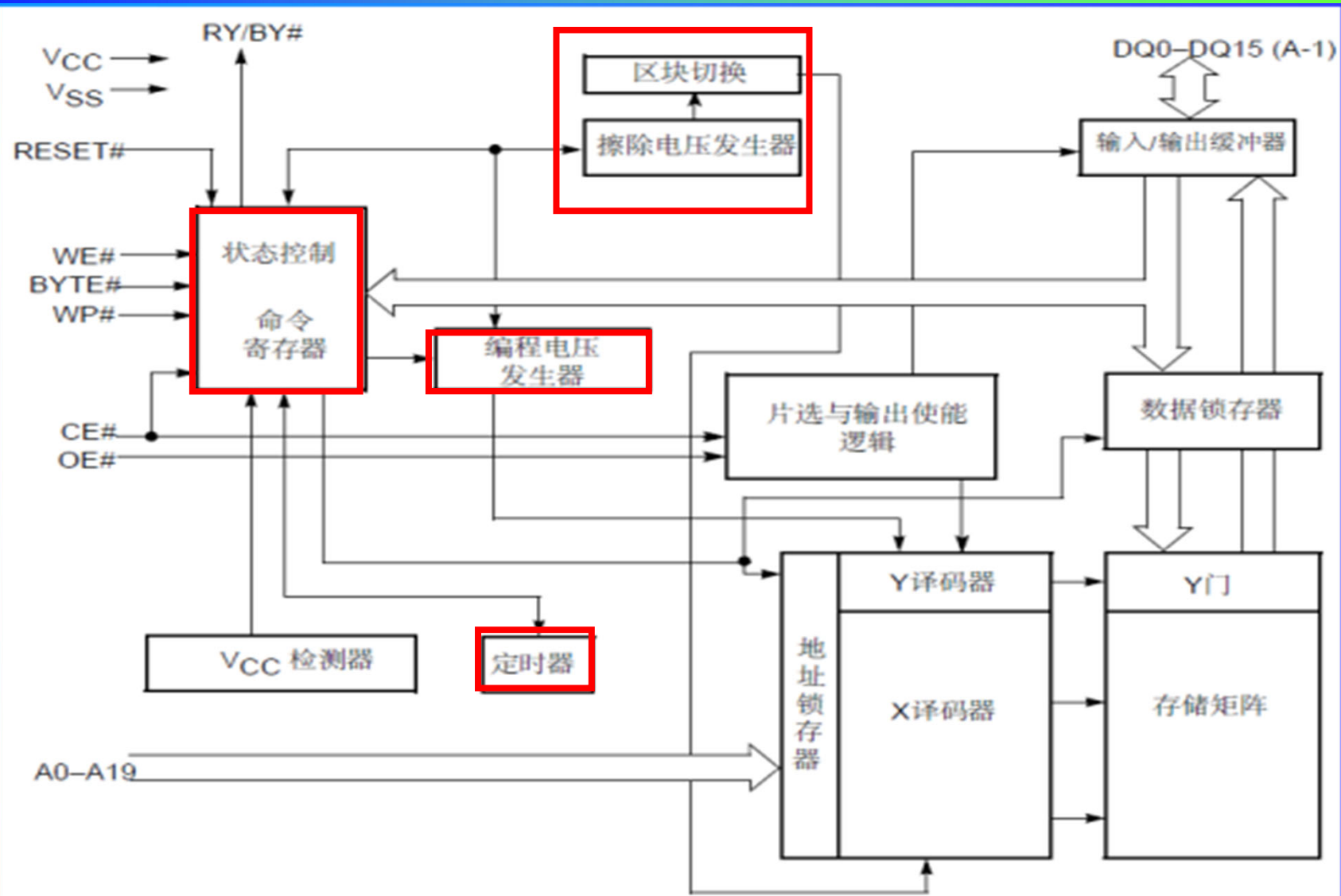
# S29AL016J 的组织结构



区块	$A_{19}$	$A_{18}$	$A_{17}$	$A_{16}$	$A_{15}$	$A_{14}$	$A_{13}$	$A_{12}$	区块大小 (K字节/ K字)	地址范围 (16进制)	
										字节模式 (x8)	字模式 (x16)
SA0	0	0	0	0	0	0	0	X	16/8	000000-003FFF	00000-01FFF
SA1	0	0	0	0	0	0	1	0	8/4	004000-005FFF	02000-02FFF
SA2	0	0	0	0	0	0	1	1	8/4	006000-007FFF	03000-03FFF
SA3	0	0	0	0	0	1	X	X	32/16	008000-00FFFF	04000-07FFF
SA4	0	0	0	0	1	X	X	X	64/32	010000-01FFFF	08000-0FFFF
SA5	0	0	0	1	0	X	X	X	64/32	020000-02FFFF	10000-17FFF
SA6	0	0	0	1	1	X	X	X	64/32	030000-03FFFF	18000-1FFFF
SA7	0	0	1	0	0	X	X	X	64/32	040000-04FFFF	20000-27FFF
SA8	0	0	1	0	1	X	X	X	64/32	050000-05FFFF	28000-2FFFF



# 闪速存储器的逻辑结构



# Bus Operations

操作	CE#	OE#	WE#	RESET#	WP#	地址	DQ <sub>0</sub> -DQ <sub>7</sub>	DQ <sub>8</sub> -DQ <sub>15</sub>	
								BYTE# = 0	BYTE# = 1
读	0	0	1	1	任意	地址输入	数据输出	数据输出	DQ <sub>8</sub> -DQ <sub>14</sub> = 高阻 DQ <sub>15</sub> = A-1
写	0	1	0	1	*	地址输入	数据输入/输出	数据输入/输出	
保持	$V_{CC} \pm 0.3V$	任意	任意	$V_{CC} \pm 0.3V$	任意	任意	高阻	高阻	高阻
输出禁止	0	1	1	1	任意	任意	高阻	高阻	高阻
复位	任意	任意	任意	0	任意	任意	高阻	高阻	高阻



# 闪存的工作方式



➤ 上电后，芯片内部的状态机使器件处于读存储矩阵操作状态

□ 读出操作与其它的各种ROM芯片相同

➤ 只有在执行了特定的命令序列之后，才可进入其它状态，进行芯片擦除、区块擦除、编程写入、软件数据保护或者读标识码等操作



# 命令序列



- 为防止状态机的误动作，闪存的各种命令是以向特定地址写入特定内容的“命令序列”方式定义的
- 命令寄存器和状态寄存器本身并不占据单独存储器片内地址
- 不同命令通常要占用长短不一的若干个总线写周期
- 在每一次命令操作之后，可以查询状态寄存器，以使CPU能够了解命令的执行情况





# 公共闪存接口



## ➤ 公共闪存接口（Common Flash Interface, CFI）

- ❑ 从闪存器件中读取数据的接口的公开标准

## ➤ 命令兼容性：

- ❑ 同一系列的FLASH产品提供兼容的命令集

- ❑ FLASH支持的命令集通常与Intel或AMD之一兼容

- ✉ AMD/Fujitsu标准命令集

- ✉ Intel/Sharp扩展命令集



# S29AL016J的部分命令定义（字模式）

命令序列	总线周期数	总线周期											
		周期1		周期2		周期3		周期4		周期5		周期6	
		地址	数据	地址	数据	地址	数据	地址	数据	地址	数据	地址	数据
复位	1	XXX	F0										
读	1	读地址	读数据										
芯片擦除	6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10
区块擦除	6	555	AA	2AA	55	555	80	555	AA	2AA	55	区块地址	30
编程	4	555	AA	2AA	55	555	A0	编程地址	编程数据				



# WRITE OPERATION STATUS

- 有多个比特可以用于判定编程和写入的状态
  - ❑ DQ2, DQ3, DQ5, DQ6, DQ7, and RY/BY#
  - ❑ 每个比特提供一种判断编程或擦除操作是否结束的方法
- RY/BY#: Ready/Busy#
  - ❑ 漏极开路的输出线，指明编程或擦除的内部操作是否已经完成
  - ❑ 输出为低(Busy)时，编程或擦除正在进行中
  - ❑ 输出为高(Ready) 时，编程或擦除已经结束



# 闪存的应用



- 特点：可在线“写”入数据，又具有ROM的非易失性
- 可以取代全部的UV EPROM和大部分的E<sup>2</sup>PROM
- 闪存的主要用途：
  - ❑ 存储监控程序、引导程序等基本不变或不经常改变的程序  
NOR闪存
  - ❑ 储存在掉电时需要保持的系统配置等不常改变的数据  
NOR闪存
  - ❑ 固态硬盘  
NAND闪存



# 闪速存储器 vs RAM



- NOR Flash比SRAM成本低，比SRAM集成度高，且具有非易失性
- NOR FLASH：非易失性在线可编程只读存储器
- FLASH取代DRAM？
  - ❑ 可重复擦写次数约为 $10^6$ ，【DRAM：约 $10^{15}$ 次】
  - ❑ 编程速度远低于DRAM写入速度
  - ❑ 读出速度远低于DRAM



# Memory Comparison

Memory type	Non-volatile	High density	Low power	Rewrite	Fast read
Flash	√	√	√	√	√
SRAM	—	—	—	√	√
DRAM	—	√	—	√	√
EEPROM	√	—	√	√	√
PROM/EPROM	√	√	√	—	√
ROM	√	√	√	—	√
Hard-disk	√	√	—	√	—
Floppy	√	—	—	√	—
CD	√	√	—	—	—



# 作业题

➤ P 115 5 (存储器按16位编址)

➤ 补充:

某机器中, CPU地址总线为 $A_{15}—A_0$ , 数据总线为 $D_7—D_0$ , 控制信号为 $R/nW$  (读/ $n$ 写),  $/MREQ$ (存储器请求), 当且仅当 $/MREQ$ 和 $R/nW$ 同时有效时, CPU才能对有存储器进行读(或写)。已配有一片ROM芯片, 地址空间为 $0000H—3FFFFH$ 。现再用几片 $16K \times 8$ 的芯片构成一个 $32K \times 8$ 的RAM区域, 使其地址空间为 $8000H—FFFFH$ 。假设此RAM芯片有 $/CS$ 和 $WE$ 信号控制端。试画出此CPU与上述ROM芯片和RAM芯片的连接图。



# 加速存储系统访问速度的途径



- 芯片技术——提高单个芯片的访问速度
  - ❑ 选用更高速的半导体器件
  - ❑ 改善芯片内部结构和对外接口方式
- 结构技术——改进存储器与CPU之间的连接方式
  - ❑ 双口存储器，多口存储器
  - ❑ 多体交叉存储器（多模块交叉存储器）
- 系统结构技术——从整个存储系统的角度采用分层存储结构
  - ❑ cache
  - ❑ 虚拟存储器





# 双端口存储器



## ➤ 早期：计算机系统以CPU为中心

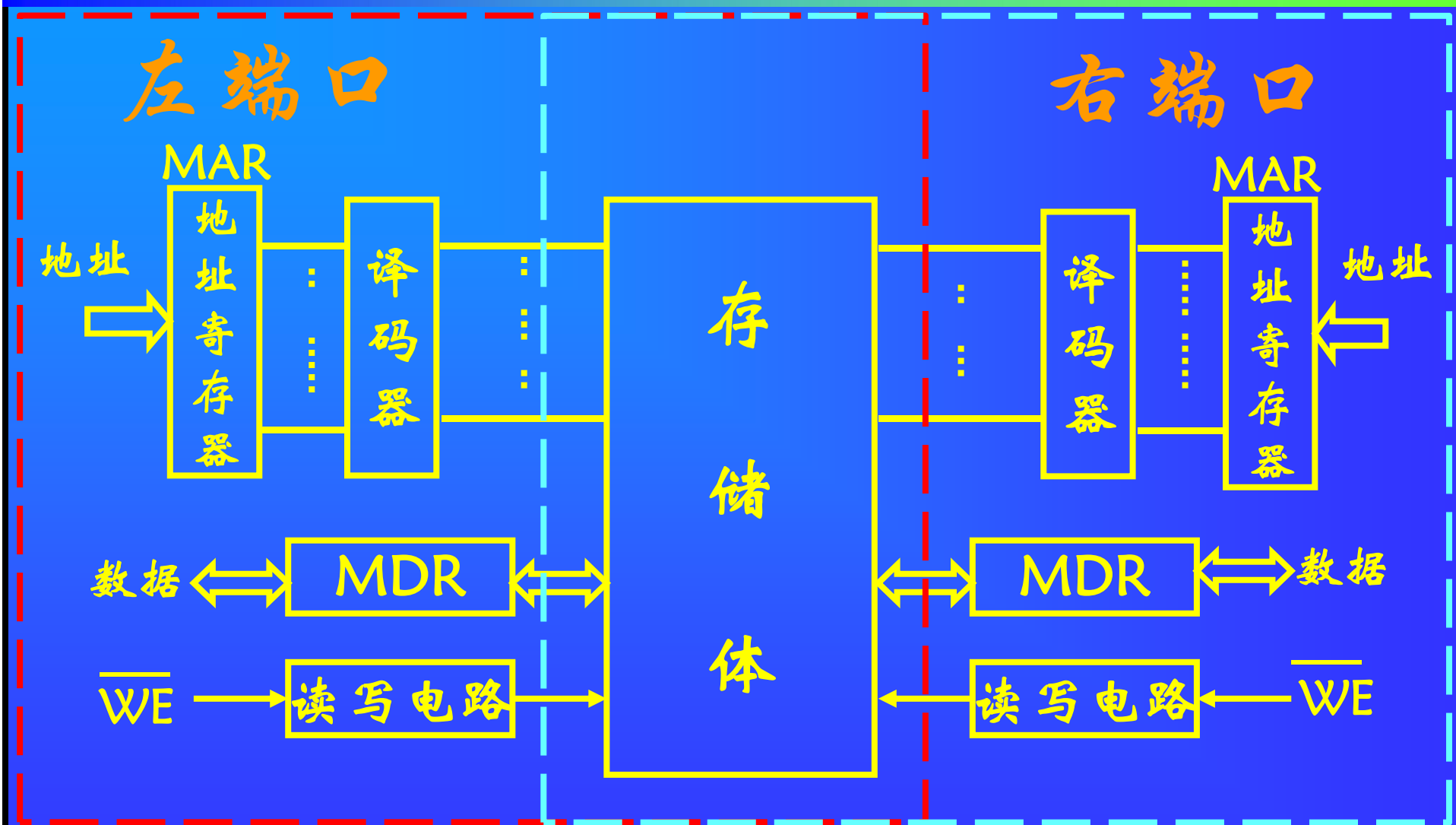
- ❑ 部件之间的信息传递受CPU控制
- ❑ I/O设备与主存之间的信息交换经过CPU的运算器

## ➤ 当前：计算机系统以内存为中心（信息交换角度）

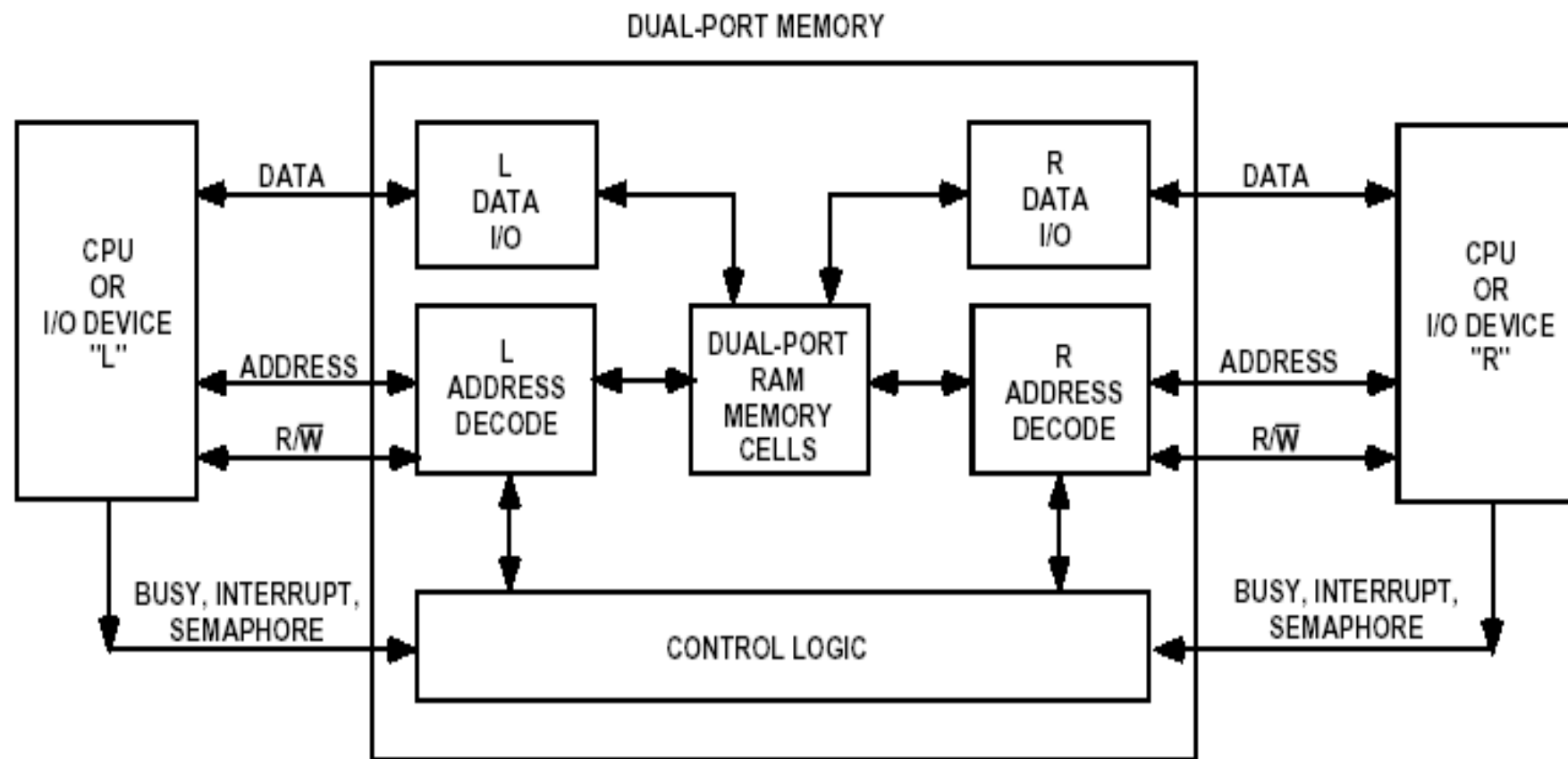
- ❑ 不仅CPU可以访问主存，其他部件也可不经CPU而直接与主存交换信息



# 双口存储器的逻辑结构



# DPRAM的应用方式



2648 drw 01

Figure 1. Dual-Port Memory Block Diagram

## DPRAM: Dual Port RAM



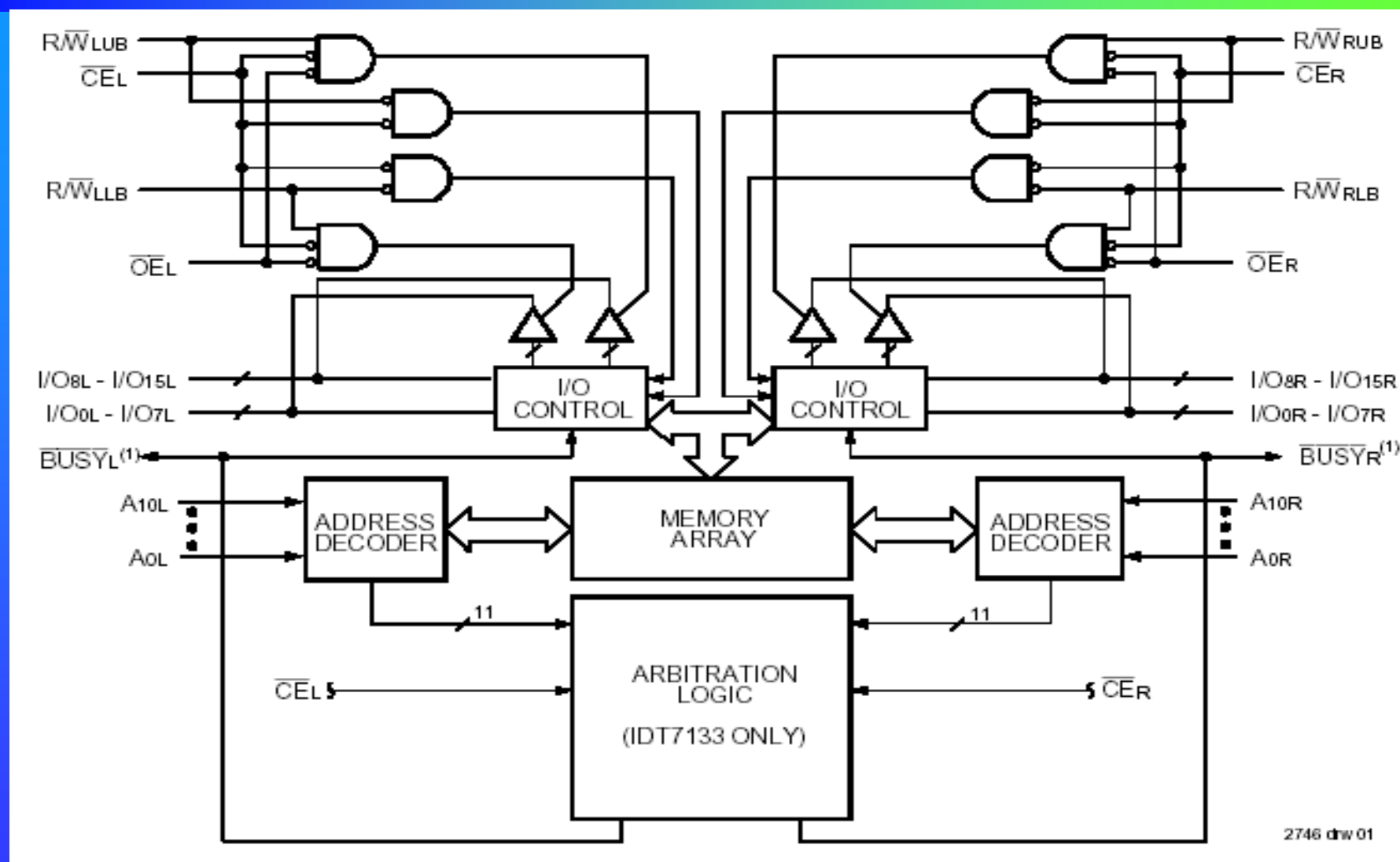
# IDT7133: 2K × 16 DPRAM的管脚

## Pin Names

Left Port	Right Port	Names
$\overline{CE_L}$	$\overline{CE_R}$	Chip Enable
$R/\overline{W}_{LUB}$	$R/\overline{W}_{RUB}$	Upper Byte Read/Write Enable
$R/\overline{W}_{LLB}$	$R/\overline{W}_{RLB}$	Lower Byte Read/Write Enable
$\overline{OE_L}$	$\overline{OE_R}$	Output Enable
$A_0L - A_{10}L$	$A_0R - A_{10}R$	Address
$I/O_0L - I/O_{15}L$	$I/O_0R - I/O_{15}R$	Data Input/Output
$\overline{BUSY_L}$	$\overline{BUSY_R}$	Busy Flag
$V_{CC}$		Power
GND		Ground



# IDT7133功能方框图



# IDT7133的读写控制方式：无冲突

LEFT OR RIGHT PORT <sup>(1)</sup>						Function
R/W <sub>LB</sub>	R/W <sub>UB</sub>	$\overline{CE}$	$\overline{OE}$	I/O <sub>0-7</sub>	I/O <sub>8-15</sub>	
X	X	H	X	Z	Z	CE <sub>R</sub> = CE <sub>L</sub> = V <sub>H</sub> , Power Down Mode, IsB1 or IsB3
L	L	L	X	DATA <sub>IN</sub>	DATA <sub>IN</sub>	Data on Lower Byte and Upper Byte Written into Memory <sup>(2)</sup>
L	H	L	L	DATA <sub>IN</sub>	DATA <sub>OUT</sub>	Data on Lower Byte Written into Memory <sup>(2)</sup> , Data in Memory Output on Upper Byte <sup>(3)</sup>
H	L	L	L	DATA <sub>OUT</sub>	DATA <sub>IN</sub>	Data in Memory Output on Lower Byte <sup>(3)</sup> , Data on Upper Byte Written into Memory <sup>(2)</sup>
L	H	L	H	DATA <sub>IN</sub>	Z	Data on Lower Byte Written into Memory <sup>(2)</sup>
H	L	L	H	Z	DATA <sub>IN</sub>	Data on Upper Byte Written into Memory <sup>(2)</sup>
H	H	L	L	DATA <sub>OUT</sub>	DATA <sub>OUT</sub>	Data in Memory Output on Lower Byte and Upper Byte
H	H	L	H	Z	Z	High Impedance Outputs

2746 13 13

## NOTES:

1. A<sub>0L</sub> - A<sub>10L</sub> ≠ A<sub>0R</sub> - A<sub>10R</sub>
2. If  $\overline{BUSY}$  = LOW, data is not written.
3. If  $\overline{BUSY}$  = LOW, data may not be valid, see two and two timing.
4. "H" = HIGH, "L" = LOW, "X" = Don't Care, "Z" = High-Impedance, "LB" = Lower Byte, "UB" = Upper Byte

当两个端口存取的存储单元的地址不相同时，读写操作不会产生冲突



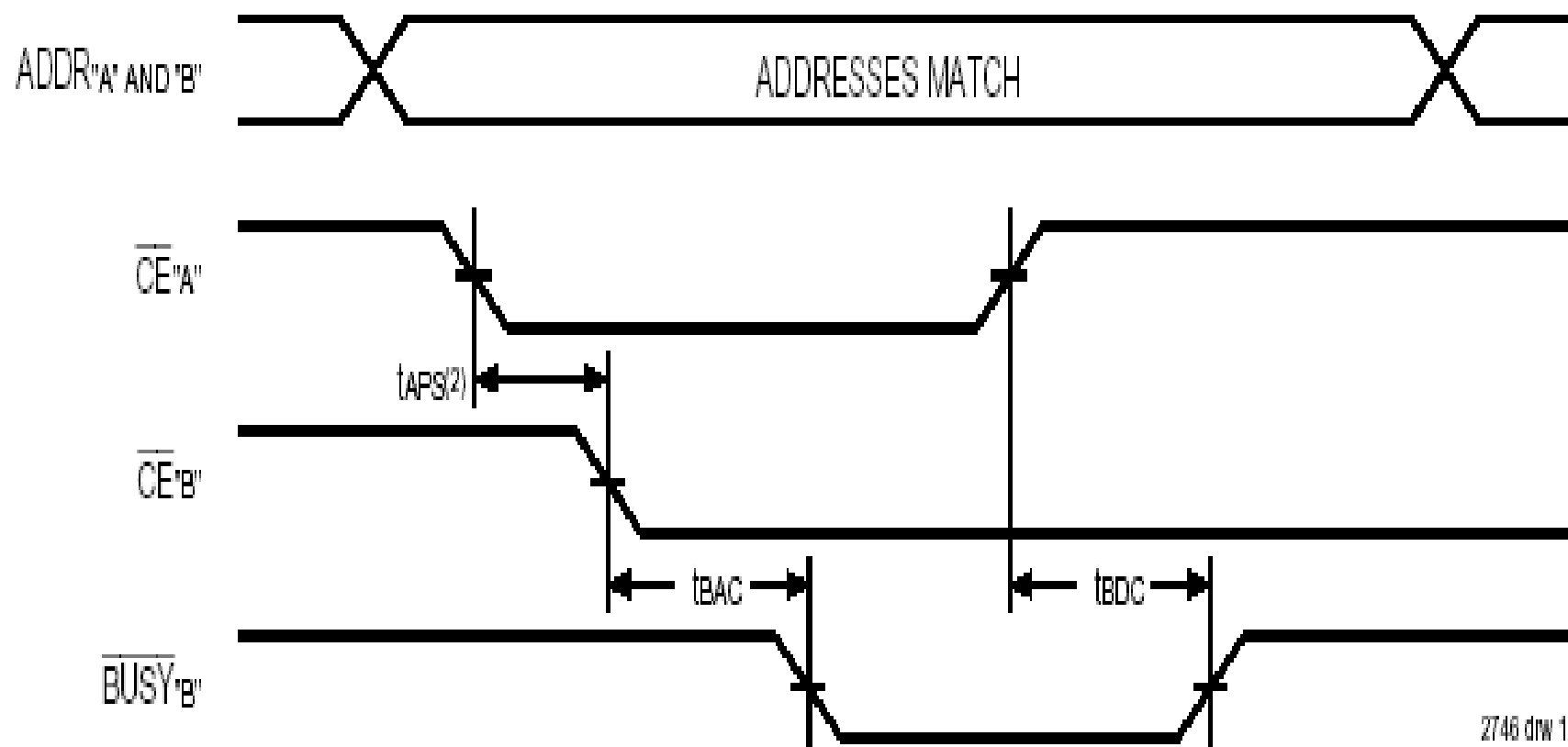
# IDT7133的读写控制方式：有冲突

- 当两个端口同时存取存储器同一存储单元时，发生读写冲突
  - ❑ 当某端口的BUSY有效时，该端口写操作被内部禁止
  - ❑ 由仲裁逻辑决定对哪个端口优先进行写操作，并延迟另一个端口的写操作
  - ❑ 仲裁逻辑的判断方式
    - ✉ CE控制的仲裁
    - ✉ 地址控制的仲裁



# /CE控制的仲裁

## Timing Waveform of $\overline{\text{BUSY}}$ Arbitration Controlled by $\overline{\text{CE}}$ Timing<sup>(1)</sup>



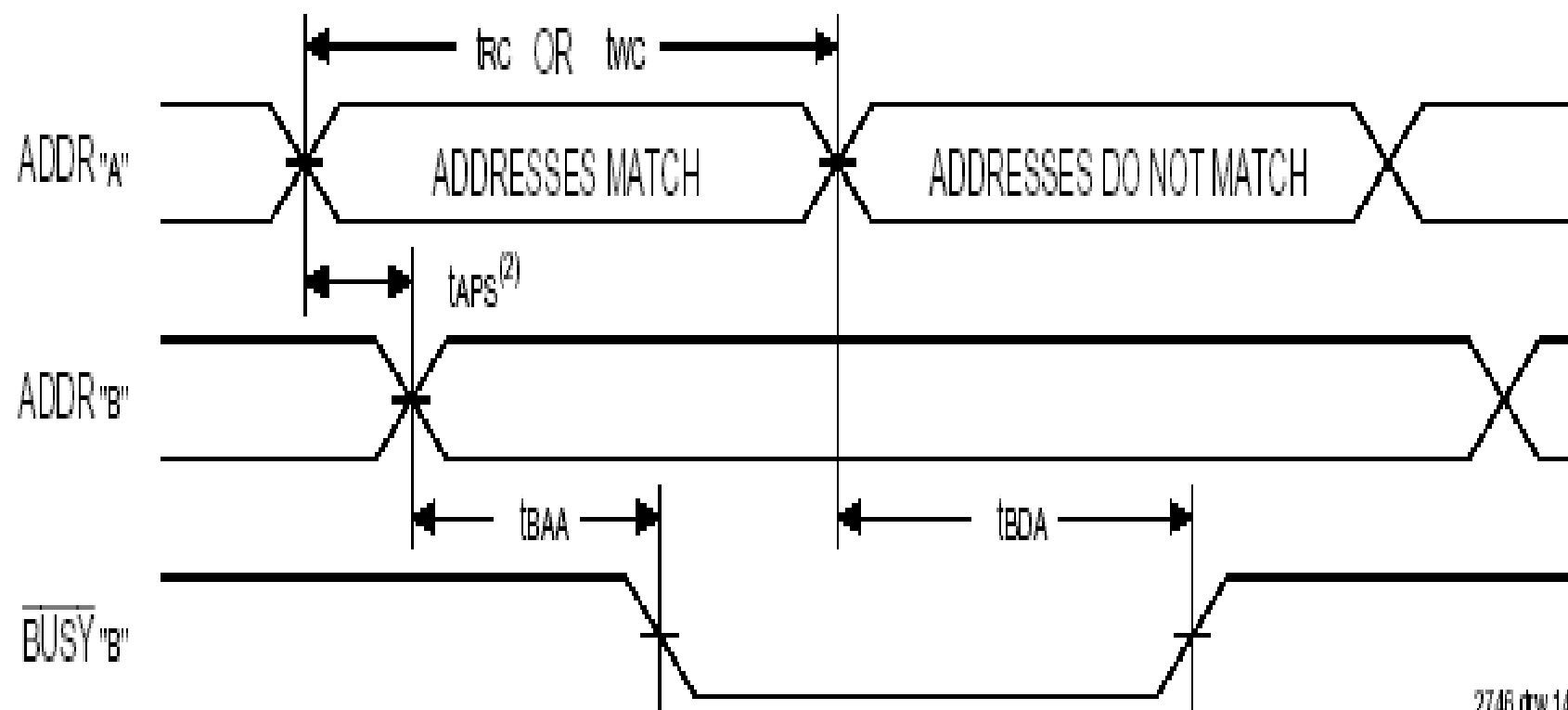
如果地址在CE有效之前匹配，则控制逻辑在 $\text{CE}_L$ 和 $\text{CE}_R$ 之间进行判断来选择端口





# 地址控制的仲裁

## Timing Waveform of $\overline{\text{BUSY}}$ Arbitration Controlled by Addresses<sup>(1)</sup>



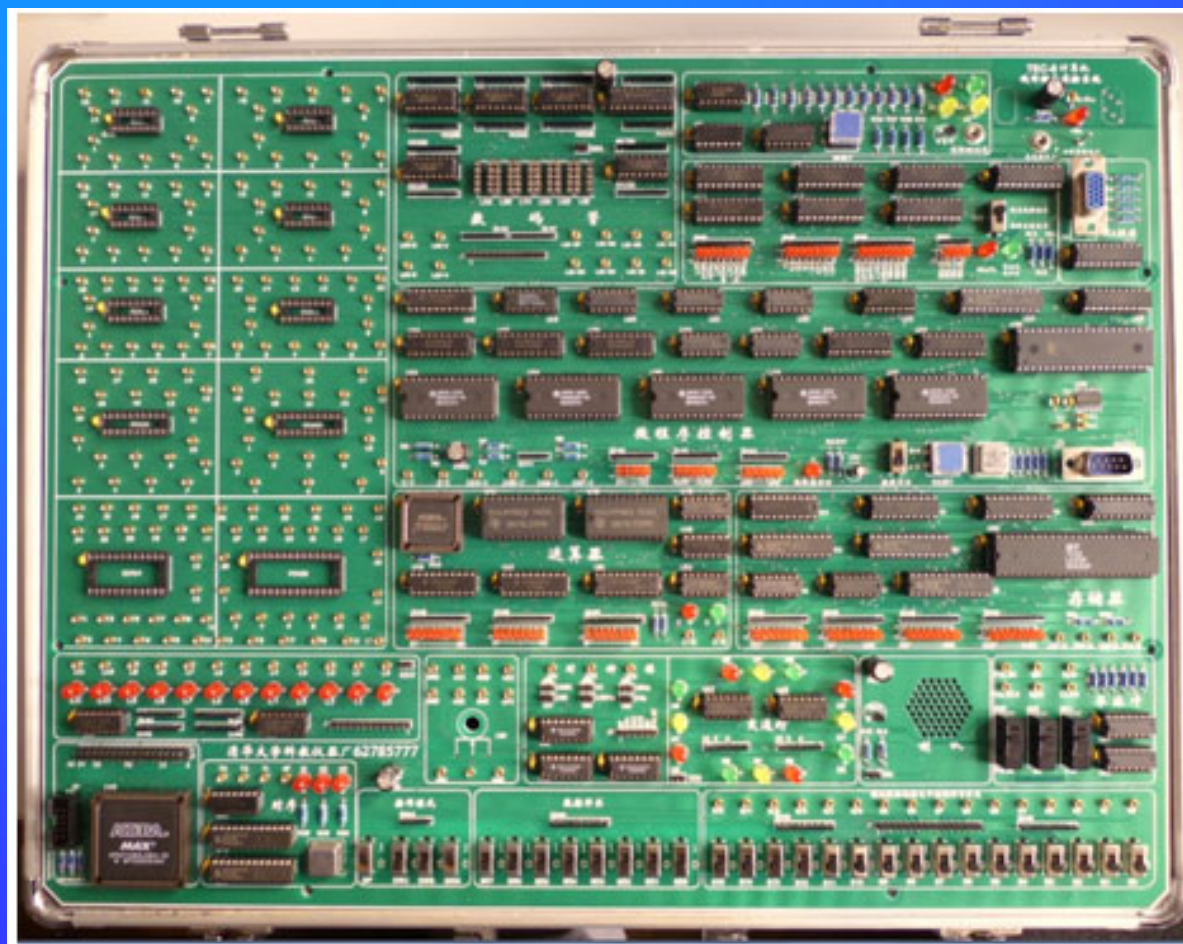
2746 drw 14

如果CE在地址匹配之前变低，则控制逻辑根据左、右地址进行判断来选择端口



# 实验二 双端口存储器

## TEC-8 计算机硬件综合实验系统



主楼  
723室  
电话：  
62283297  
张杰老师



# 【多模块交叉存储器】 (第三章第二部分)



# 本章内容（第一部分）

- 存储系统的基本概念
- 随机存取存储器
- 只读存储器
- 并行存储器



# 本章重点（第一部分）



- 存储系统的基本概念与性能指标
- 存储系统接口
- 提高存储系统性能的方法



# 计算机组成原理

## Principle of Computer Organization

### ➤ 第三章 存储系统

第一部分

# 结束

