计算机组成原理与接口技术——基于MIPS架构

Apr, 2022

第6讲 半导体存储器接口

杨明
华中科技大学电信学院
myang@hust.edu.cn

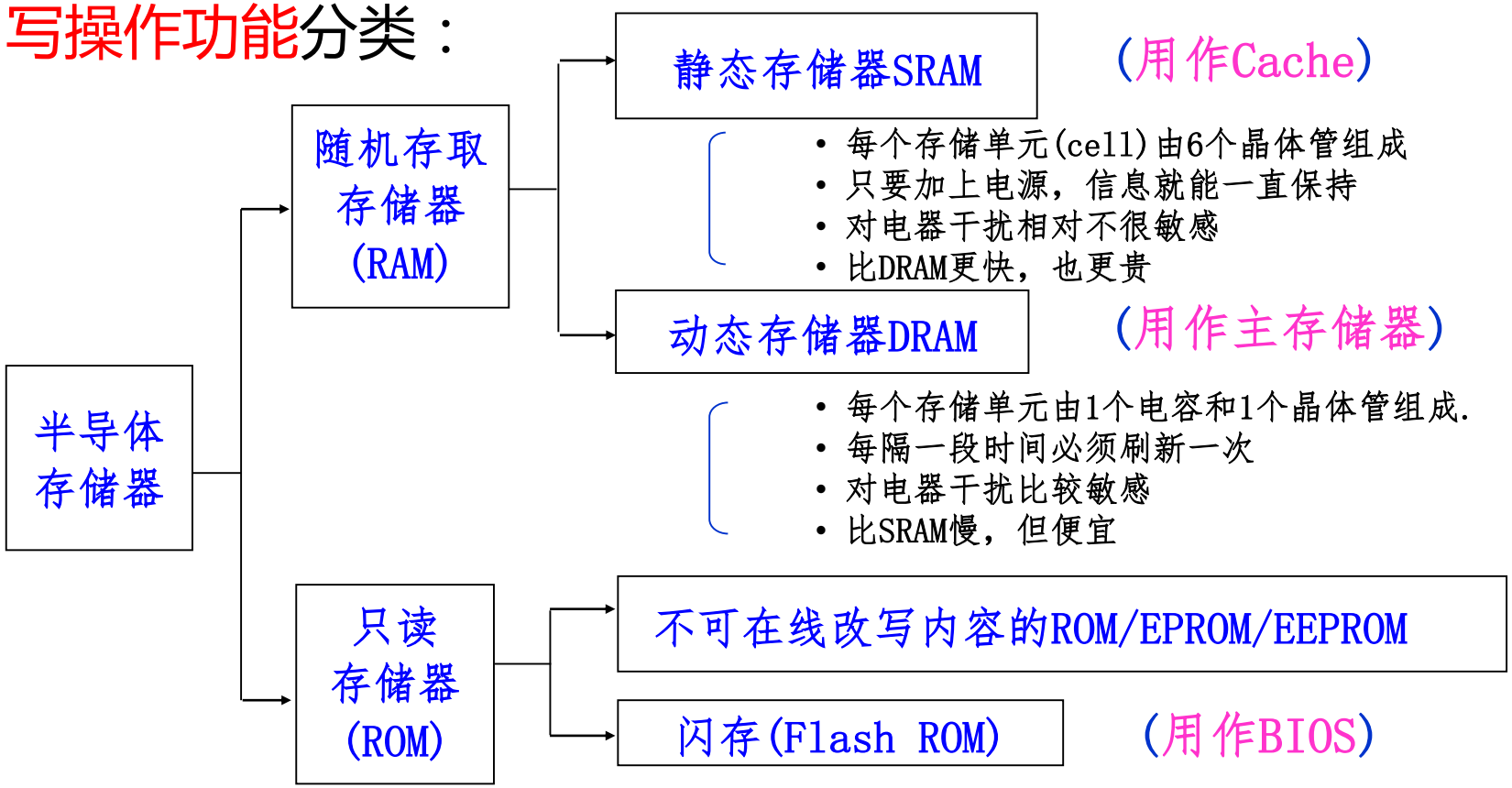
Agenda

- 内容
  - 半导体存储芯片分类
  - 典型存储芯片的结构特点、读写时序
  - 接口的基本概念
  - 存储器接口设计
- 目的
  - 了解存储芯片分类
  - 了解典型存储芯片的结构特点、读写时序
  - 了解接口的基本构成、数据传送方式以及控制方式
  - 掌握存储器容量、字长扩展接口设计
  - 掌握支持不同类型数据访问的存储器接口电路设计

6.1 存储器分类

▶ 半导体存储芯片分类

- 根据读、写操作功能分类：



- 根据读写操作时序分类：异步存储器和同步存储器
- 根据数据传输的方式分类：并行存储器与串行存储器

6.2 典型存储芯片

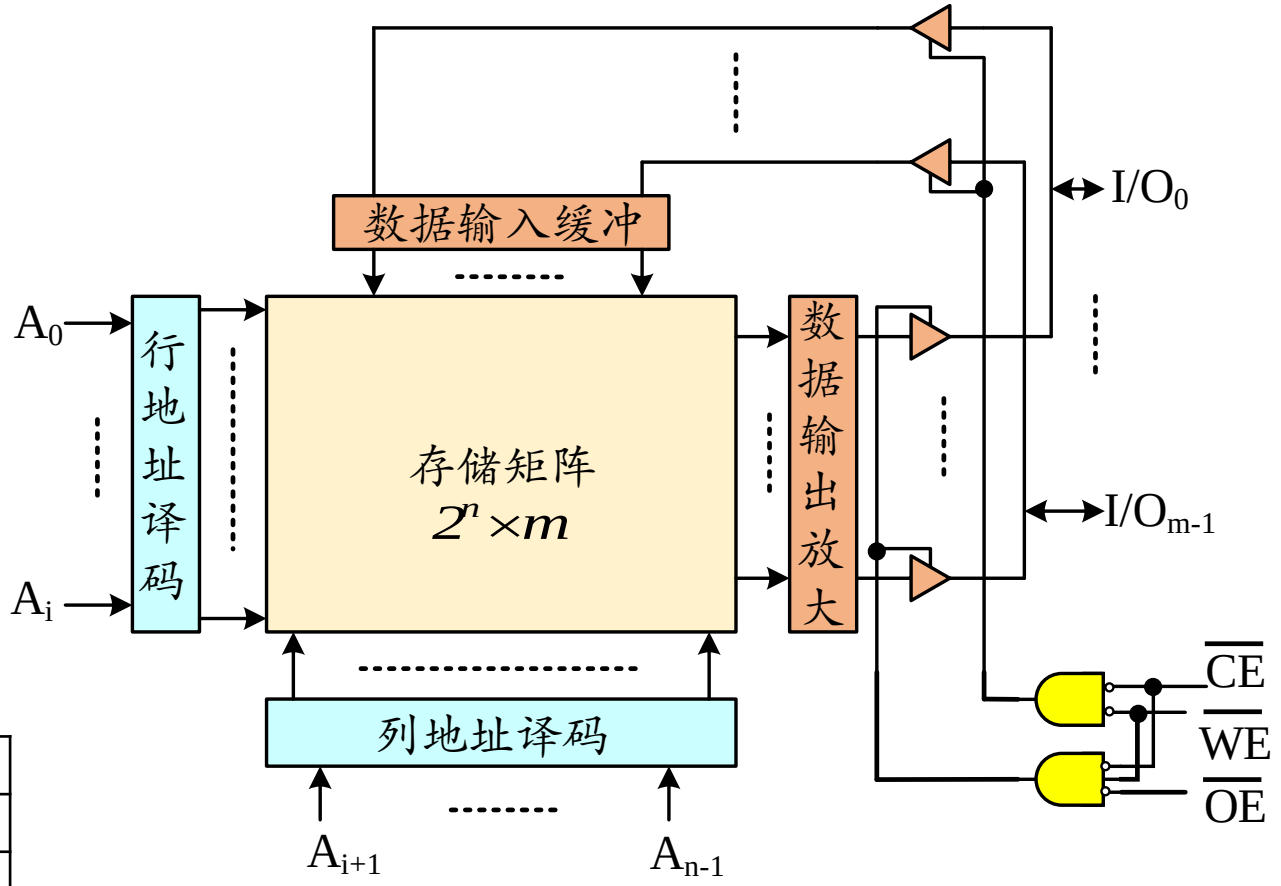
- 异步SRAM
- 同步SRAM（SSRAM）
- 同步DRAM（SDRAM）
- DDR2-SDRAM
- NOR FLASH
- NAND FLASH

6.2 典型存储芯片

▶ 6.2.1 异步SRAM

- 结构框图
  - 存储矩阵
  - 地址译码器
  - 数据I/O
  - 读写控制逻辑

采用线性寻址，地址线根数和存储容量相关，通过地址信号直接寻址内部的每一个字节



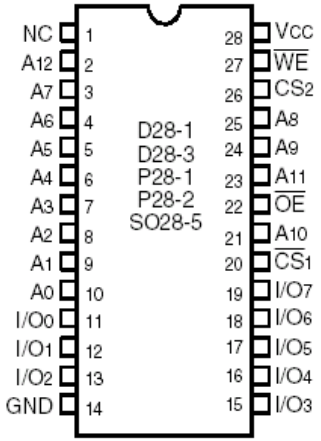
- 工作模式

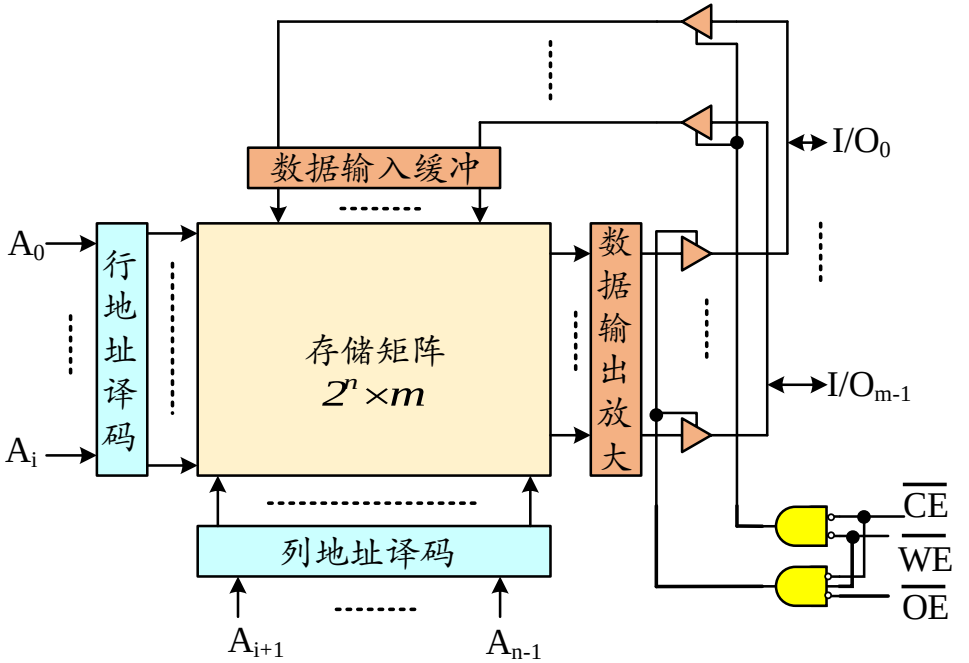
工作模式	$\overline{CE}$	$\overline{WE}$	$\overline{OE}$	$I/O_0 - I/O_{m-1}$
保持(微功耗)	1	X	X	高阻
读	0	1	0	数据输出
写	0	0	1	数据输入
输出无效	0	1	1	高阻

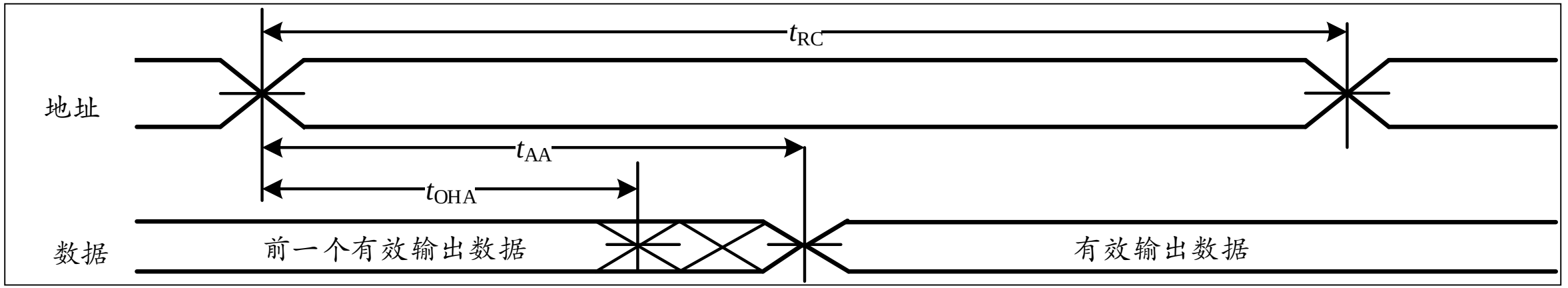
6.2 典型存储芯片

▶ 6.2.1 异步SRAM

- 结构框图
- 工作模式
- 读写时序
  - 读操作1：#CE=#OE=0,#WE=1时
    - 地址改变、输出数据就改变
    - 数据输出之和地址变化有关
    - 地址和有效数据之间有时延 tAA



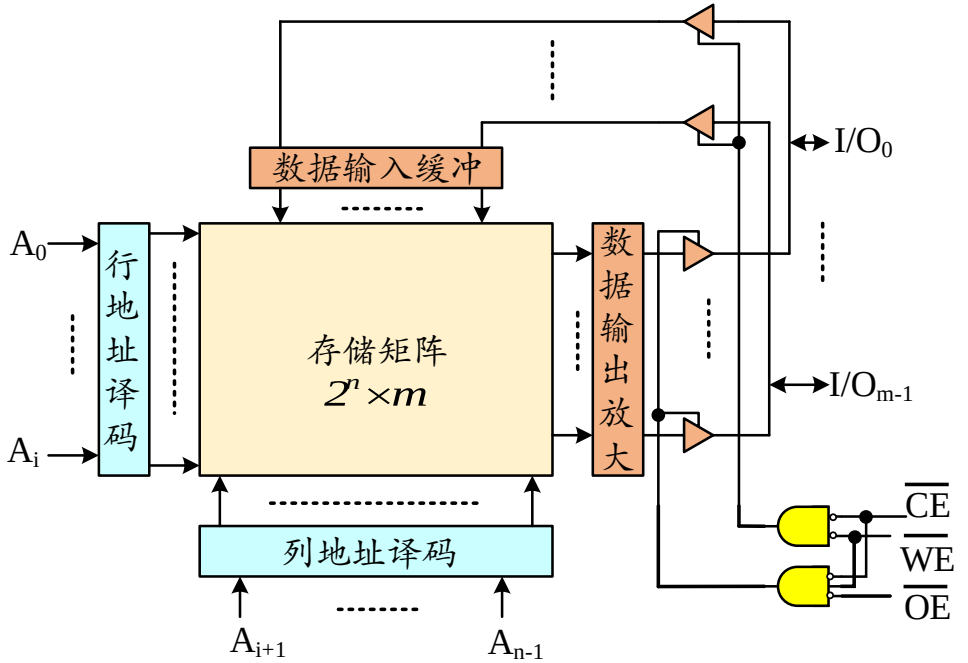


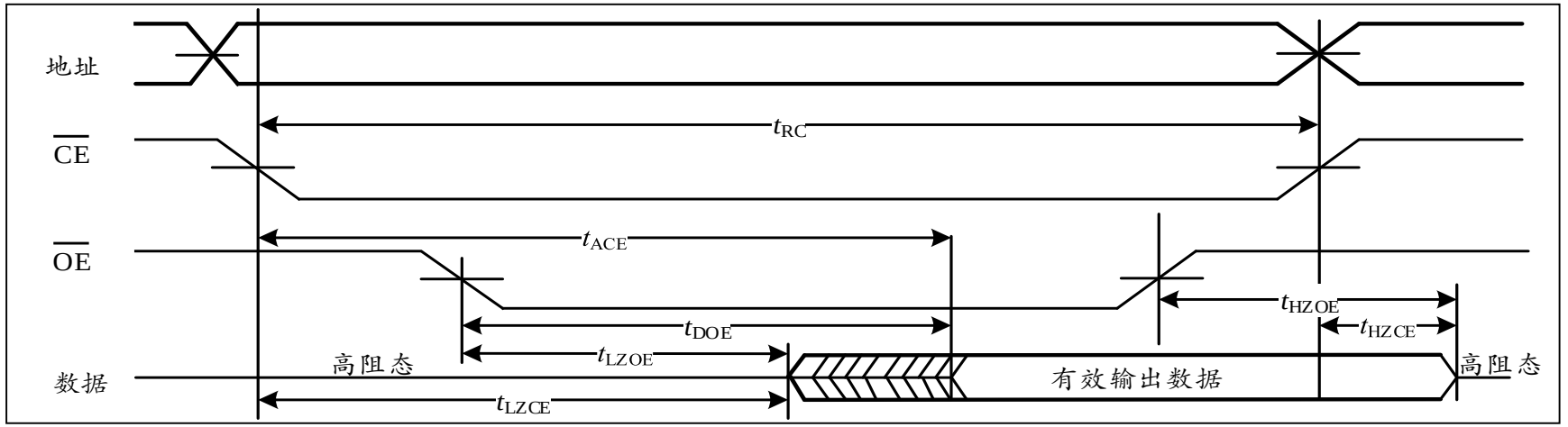


6.2 典型存储芯片

▶ 6.2.1 异步SRAM

- 结构框图
- 工作模式
- 读写时序
  - 读操作2：#WE=1，地址先有效或与#CE同时有效，数据输出由#OE控制
    - 地址在#CE下降沿有效
    - 地址和有效数据之间有时延 tACE

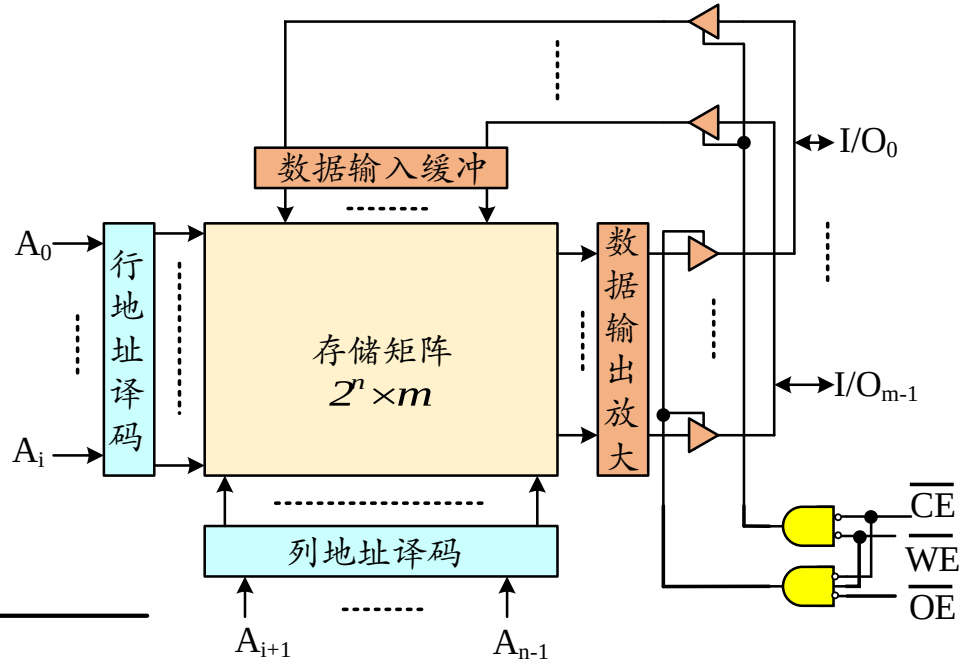


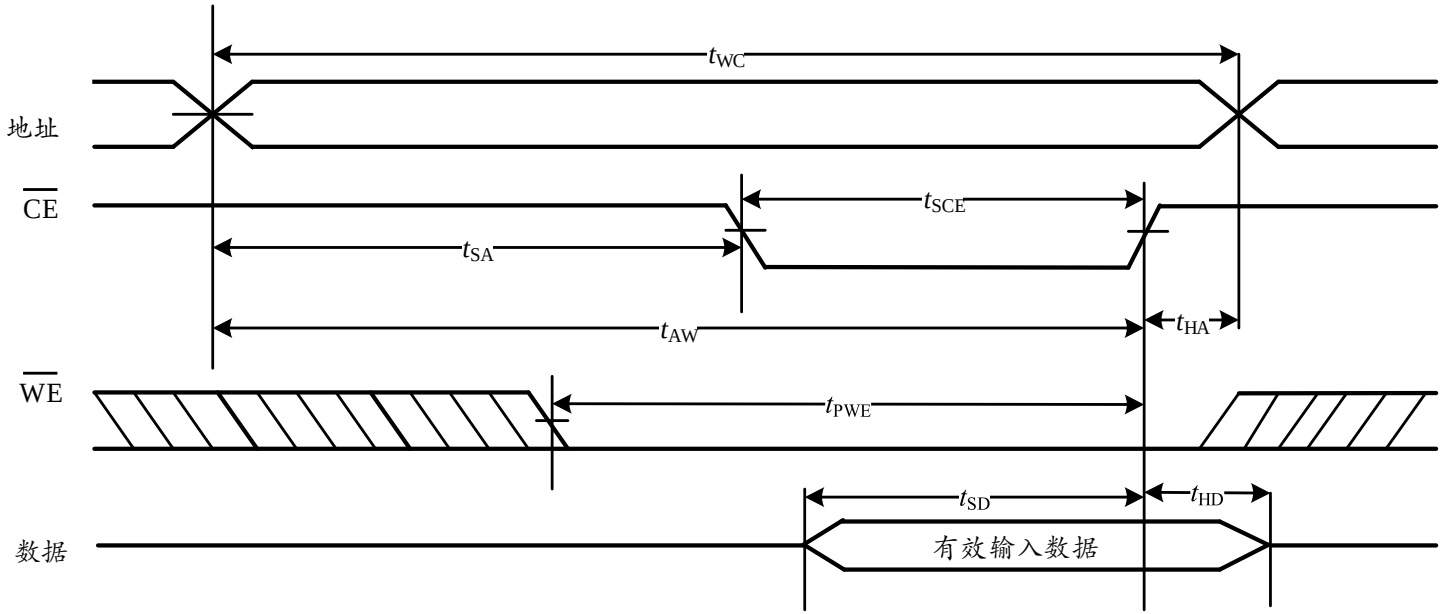


6.2 典型存储芯片

▶ 6.2.1 异步SRAM

- 结构框图
- 工作模式
- 读写时序
  - 写操作1： #OE=1， #CE控制写入
    - 在#CE上升沿写入数据
    - 需要 tPWE > tSCE

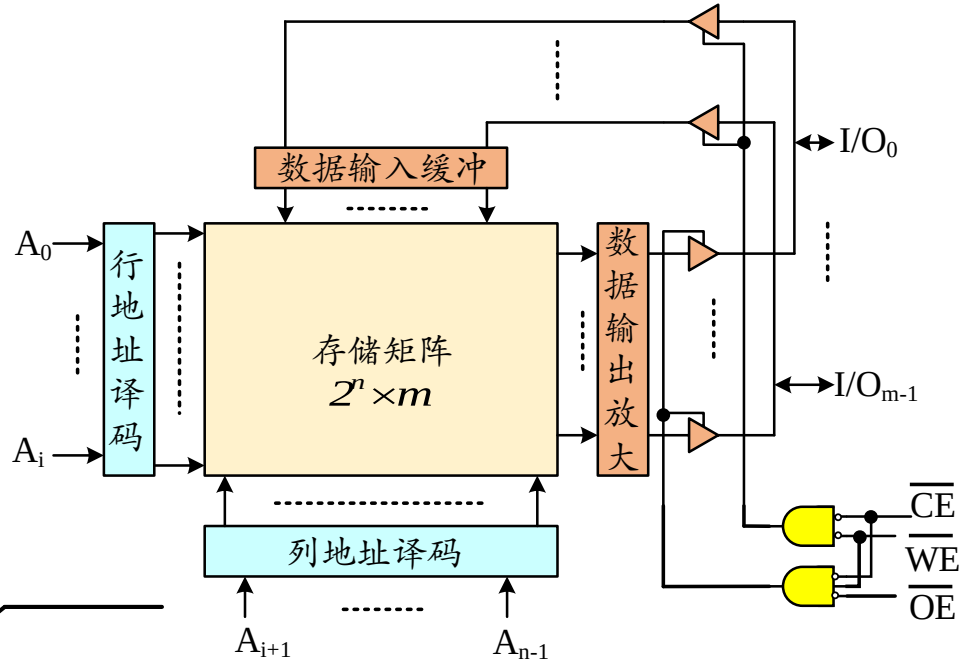


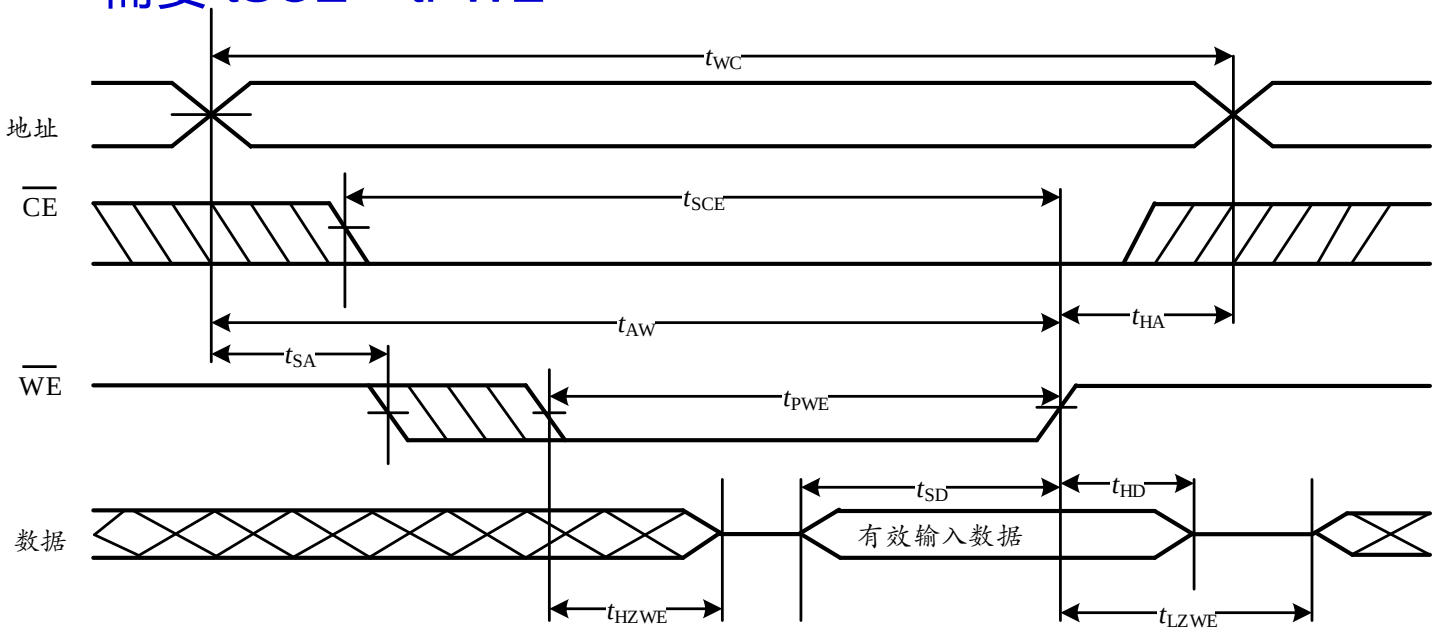


6.2 典型存储芯片

▶ 6.2.1 异步SRAM

- 结构框图
- 工作模式
- 读写时序
  - 写操作2：#OE=1，#WE控制写入
    - 在#WE上升沿写入数据
    - 需要 tSCE > tPWE



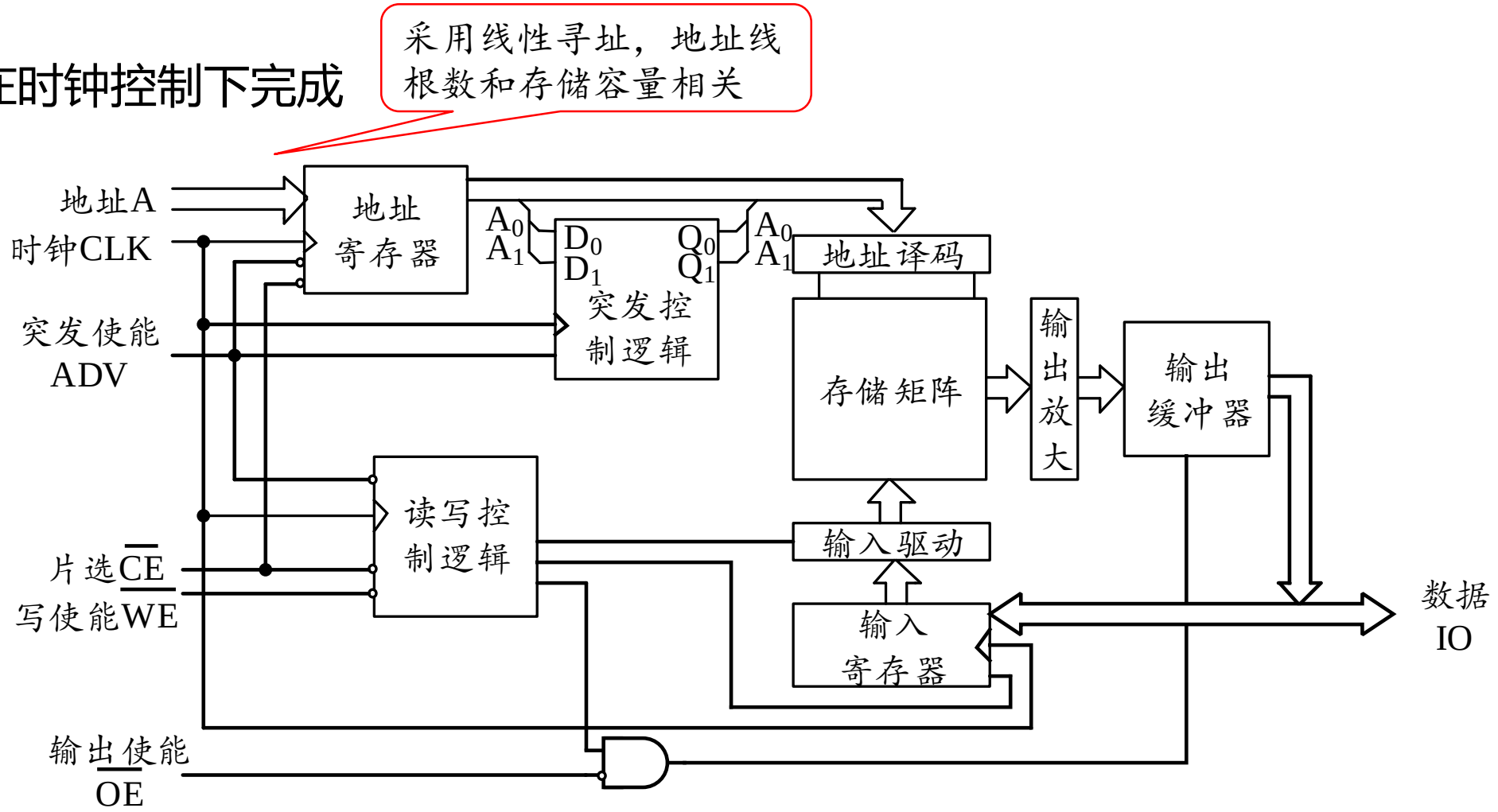


6.2 典型存储芯片

- 异步SRAM
- 同步SRAM（SSRAM）
- 同步DRAM（SDRAM）
- DDR2-SDRAM
- NOR FLASH
- NAND FLASH

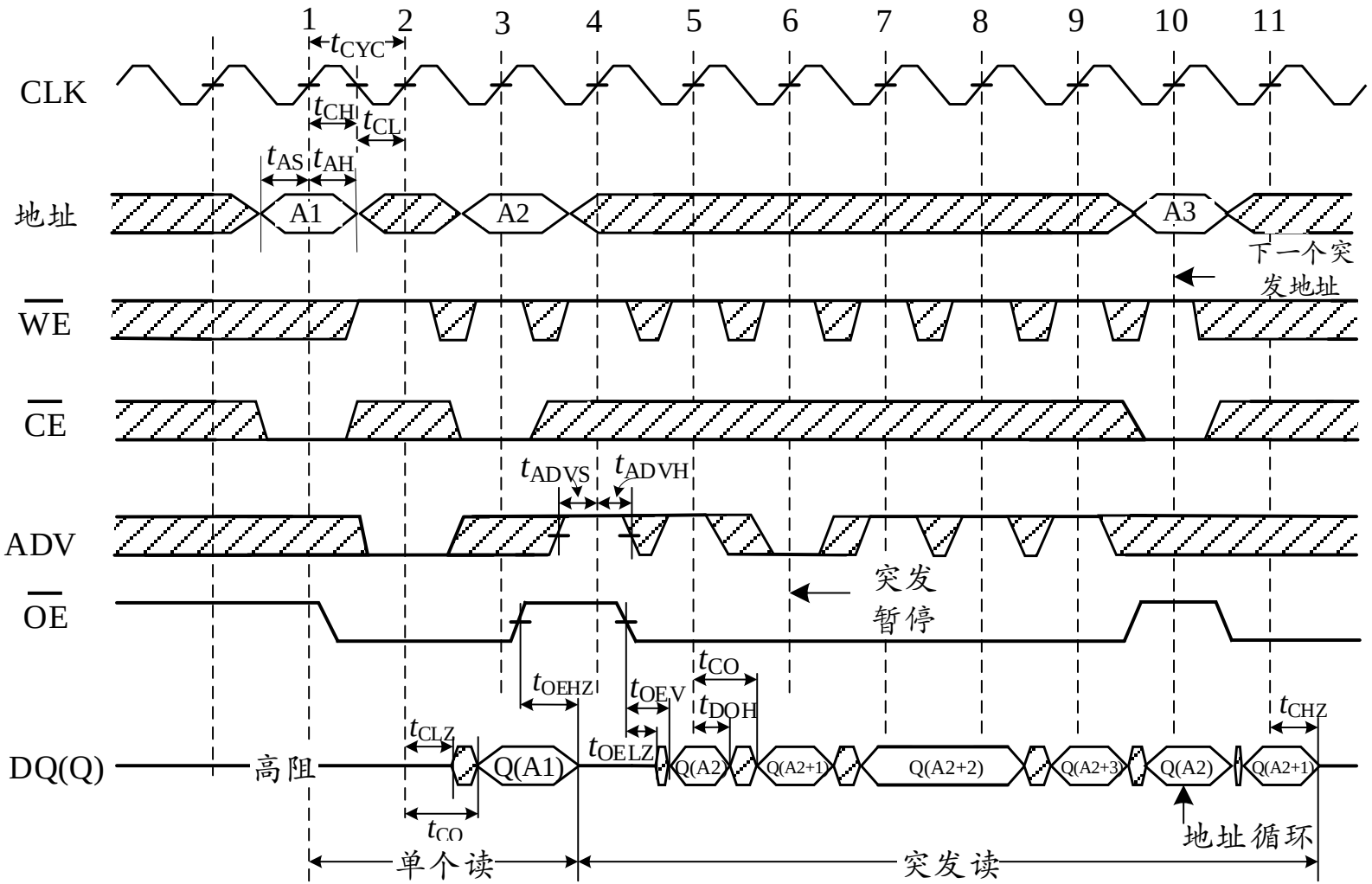
▶ 6.2.2 同步SRAM（SSRAM）

- 结构框图
  - 读写都在时钟控制下完成



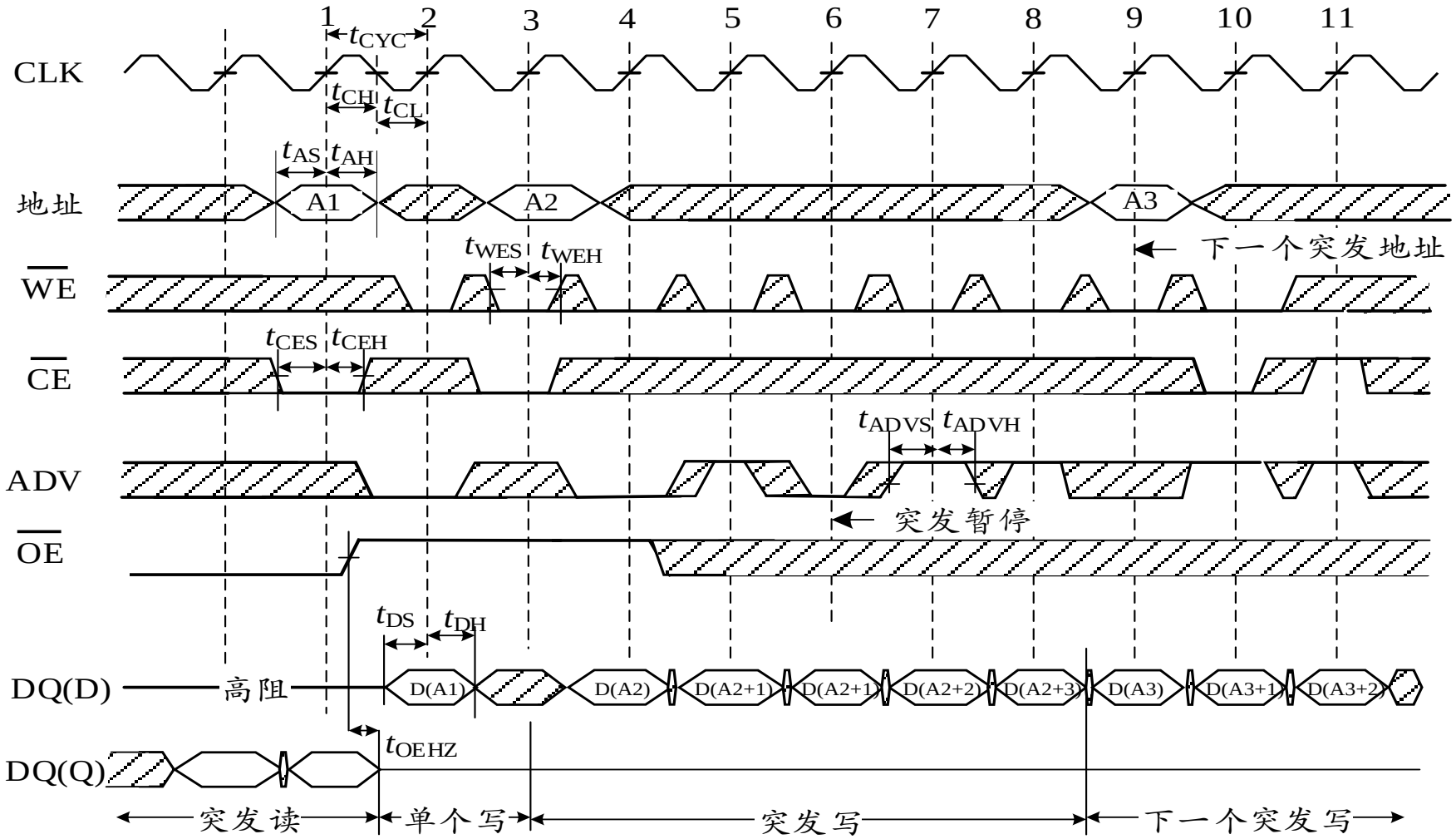
▶ 6.2.2 同步SRAM（SSRAM）

- 结构框图
- 读操作



▶ 6.2.2 同步SRAM（SSRAM）

- 结构框图
- 读操作
- 写操作

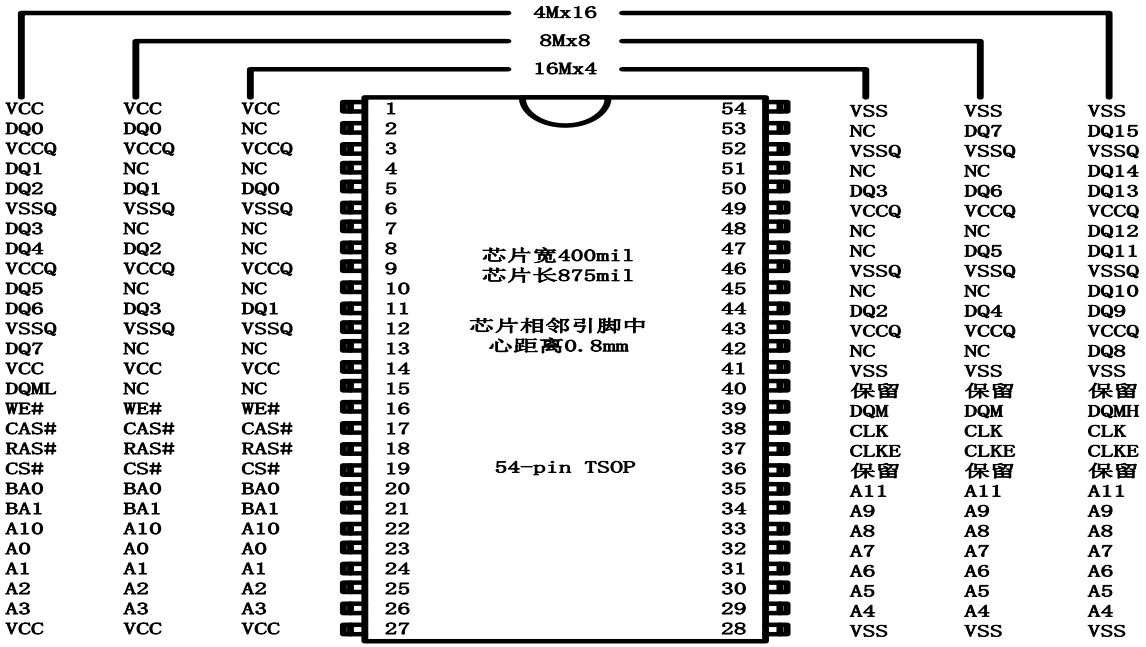


6.2 典型存储芯片

- 异步SRAM
- 同步SRAM（SSRAM）
- 同步DRAM（SDRAM）
- DDR2-SDRAM
- NOR FLASH
- NAND FLASH

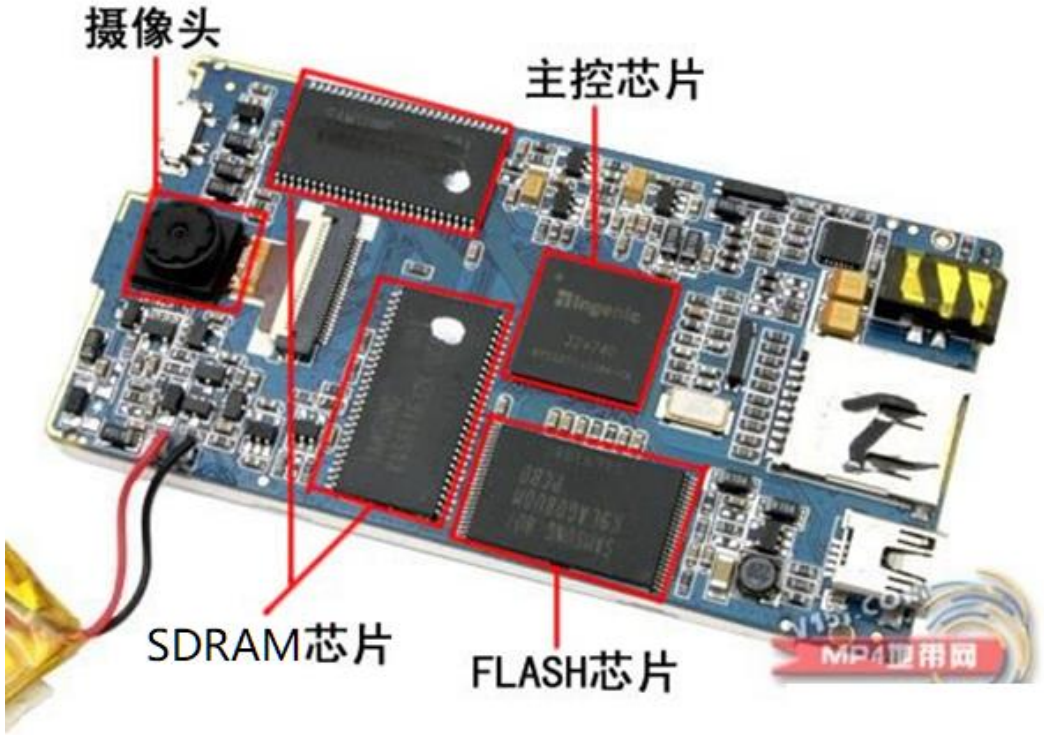
▶ 6.2.3 同步DRAM（SDRAM）

- 外观与管脚



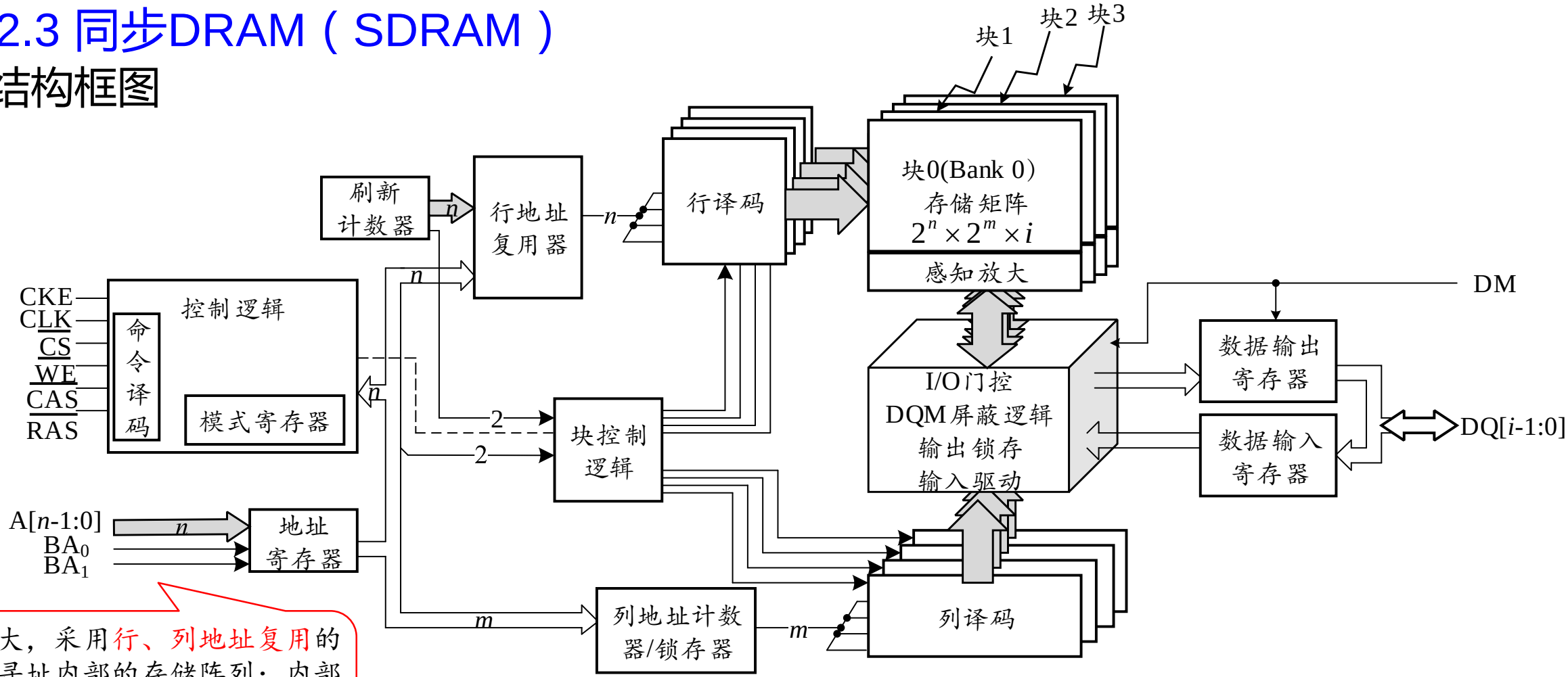
VCC	VCC	VCC	1	54	VSS
DQ0	DQ0	NC	2	53	NC
VCCQ	VCCQ	VCCQ	3	52	VSSQ
DQ1	NC	NC	4	51	NC
DQ2	DQ1	DQ0	5	50	DQ3
VSSQ	VSSQ	VSSQ	6	49	VCCQ
DQ3	NC	NC	7	48	NC
DQ4	DQ2	NC	8	47	NC
VCCQ	VCCQ	VCCQ	9	46	VSSQ
DQ5	NC	NC	10	45	NC
DQ6	DQ3	DQ1	11	44	DQ2
VSSQ	VSSQ	VSSQ	12	43	VCCQ
DQ7	NC	NC	13	42	NC
VCC	VCC	VCC	14	41	VSS
DQML	NC	NC	15	40	保留
WE#	WE#	WE#	16	39	DQM
CAS#	CAS#	CAS#	17	38	CLK
RAS#	RAS#	RAS#	18	37	CLKE
CS#	CS#	CS#	19	36	保留
BA0	BA0	BA0	20	35	A11
BA1	BA1	BA1	21	34	A9
A10	A10	A10	22	33	A8
A0	A0	A0	23	32	A7
A1	A1	A1	24	31	A6
A2	A2	A2	25	30	A5
A3	A3	A3	26	29	A4
VCC	VCC	VCC	27	28	VSS

VSS	VSS	VSS
NC	DQ7	DQ15
VSSQ	VSSQ	VSSQ
NC	NC	DQ14
DQ3	DQ6	DQ13
VCCQ	VCCQ	VCCQ
NC	NC	DQ12
NC	DQ5	DQ11
VSSQ	VSSQ	VSSQ
NC	NC	DQ10
DQ2	DQ4	DQ9
VCCQ	VCCQ	VCCQ
NC	NC	DQ8
VSS	VSS	VSS
保留	保留	保留
DQM	DQM	DQMH
CLK	CLK	CLK
CLKE	CLKE	CLKE
保留	保留	保留
A11	A11	A11
A9	A9	A9
A8	A8	A8
A7	A7	A7
A6	A6	A6
A5	A5	A5
A4	A4	A4
VSS	VSS	VSS



▶ 6.2.3 同步DRAM（SDRAM）

- 结构框图



容量大，采用行、列地址复用的方式寻址内部的存储阵列；内部一般分为多个BANK，对应有BA信号实现BANK的选择

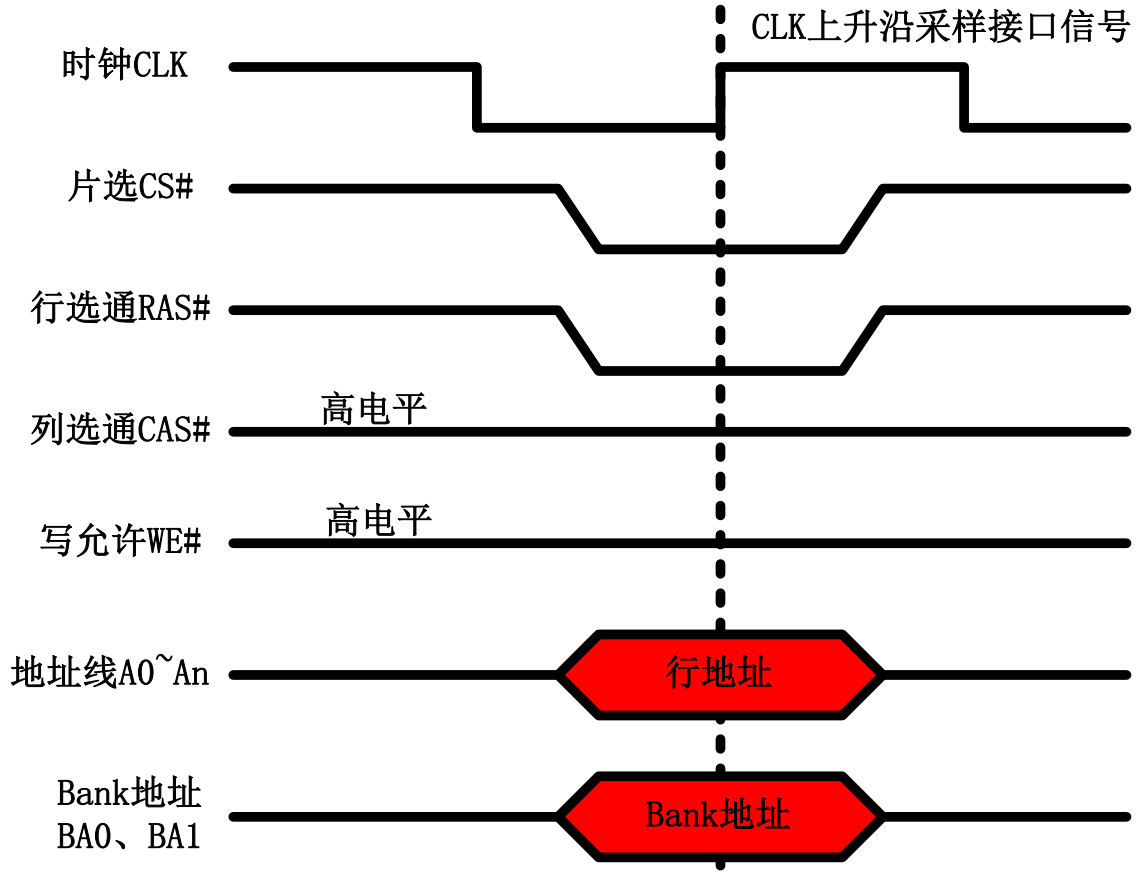
▶ 6.2.3 同步DRAM（SDRAM）

- 读写时序
  - 读写命令

命令类型	CS#	RAS#	CAS#	WE#	DQM	地址线
无操作	H	X	X	X	X	X
	L	H	H	H	X	X
行有效（激活Bank的某行）	L	L	H	H	X	Bank/行地址
列有效与读命令	L	H	L	H	L/H	Bank/列地址
列有效与写命令	L	H	L	L	L/H	Bank/列地址
突发传输终止命令	L	H	H	L	X	X
模式寄存器设置命令	L	L	L	L	X	寄存器值

▶ 6.2.3 同步DRAM（SDRAM）

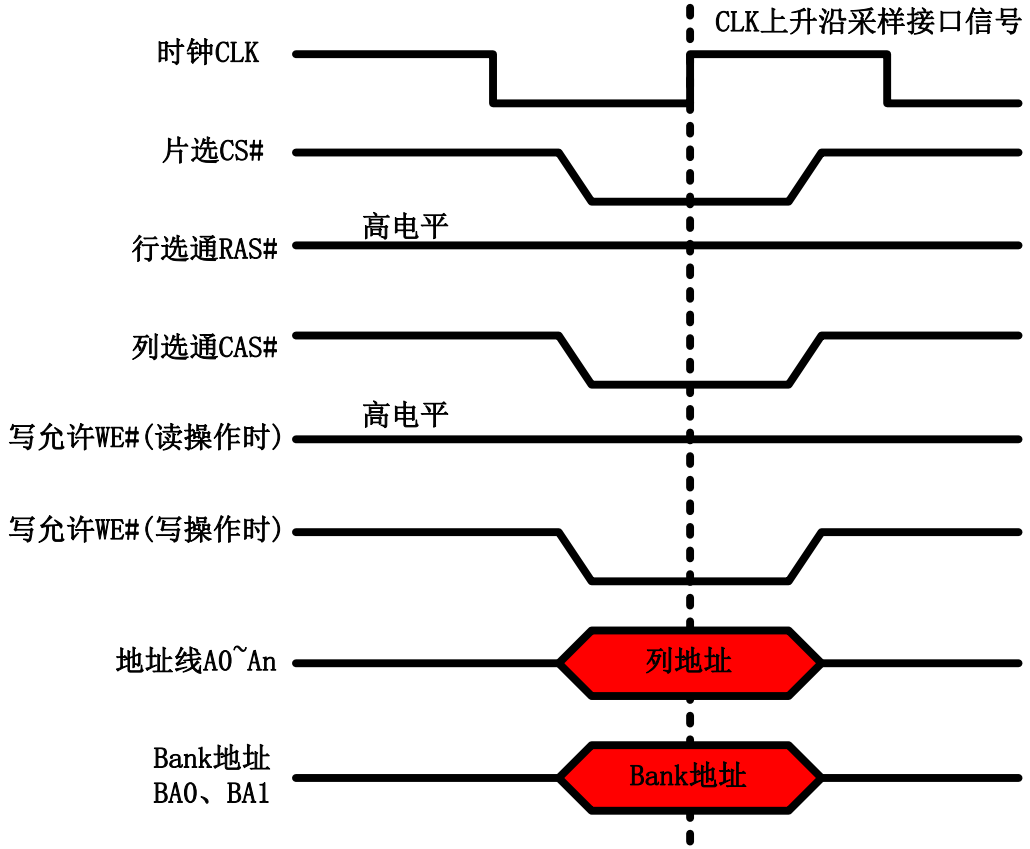
- 读写时序
  - 行有效命令



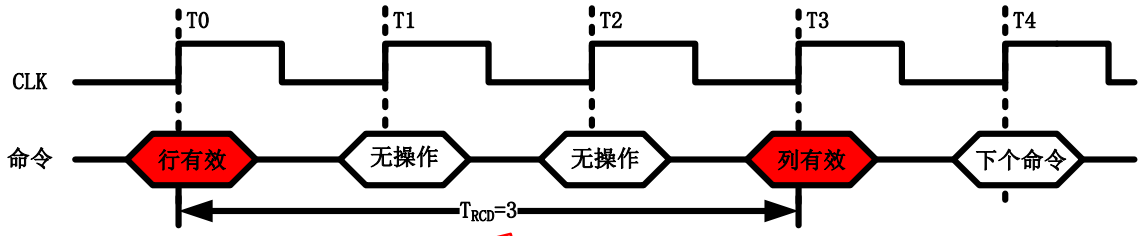
命令类型	CS#	RAS#	CAS#	WE#	DQM	地址线
无操作	H	X	X	X	X	X
	L	H	H	H	X	X
行有效（激活Bank的某行）	L	L	H	H	X	Bank/行地址
列有效与读命令	L	H	L	H	L/H	Bank/列地址
列有效与写命令	L	H	L	L	L/H	Bank/列地址
突发传输终止命令	L	H	H	L	X	X
模式寄存器设置命令	L	L	L	L	X	寄存器值

▶ 6.2.3 同步DRAM（SDRAM）

- 读写时序
  - 列有效与读/写命令时序



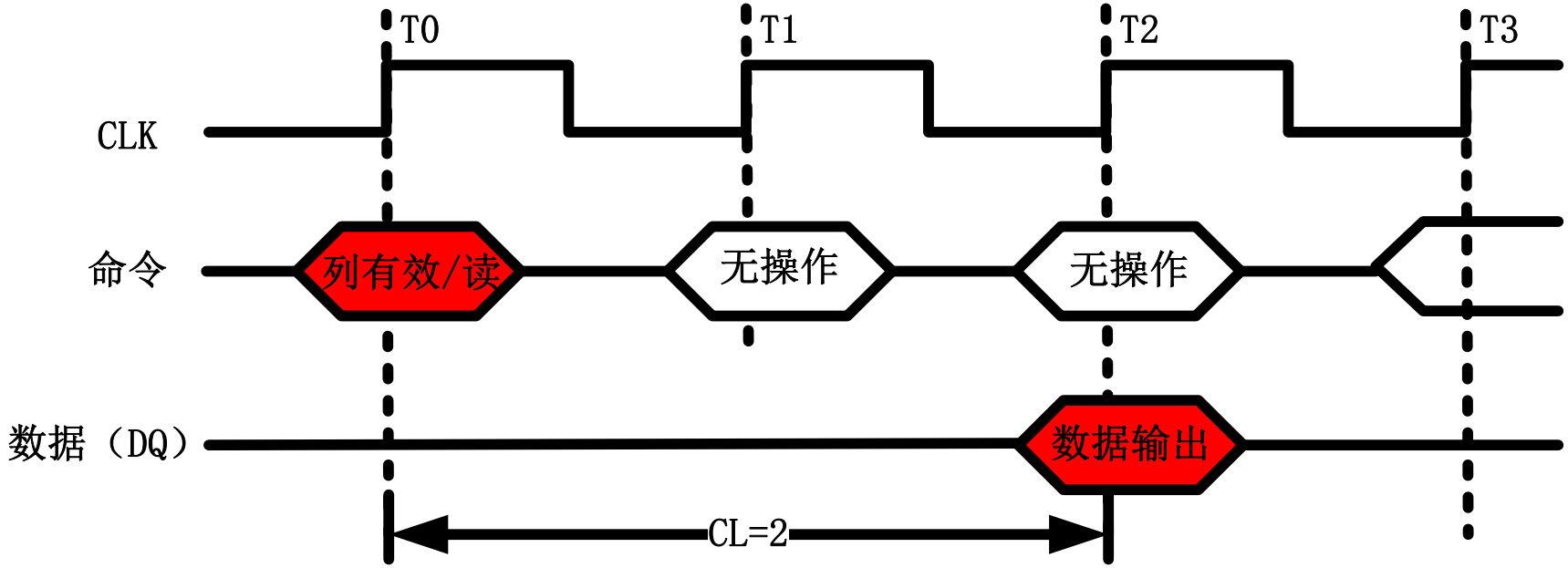
命令类型	CS#	RAS#	CAS#	WE#	DQM	地址线
无操作	H	X	X	X	X	X
	L	H	H	H	X	X
行有效（激活Bank的某行）	L	L	H	H	X	Bank/行地址
列有效与读命令	L	H	L	H	L/H	Bank/列地址
列有效与写命令	L	H	L	L	L/H	Bank/列地址
突发传输终止命令	L	H	H	L	X	X
模式寄存器设置命令	L	L	L	L	X	寄存器值



发送列读写命令时必须与行有效命令之间保持的间隔，即RAS to CAS Delay，用tRCD = 时钟周期数 表示

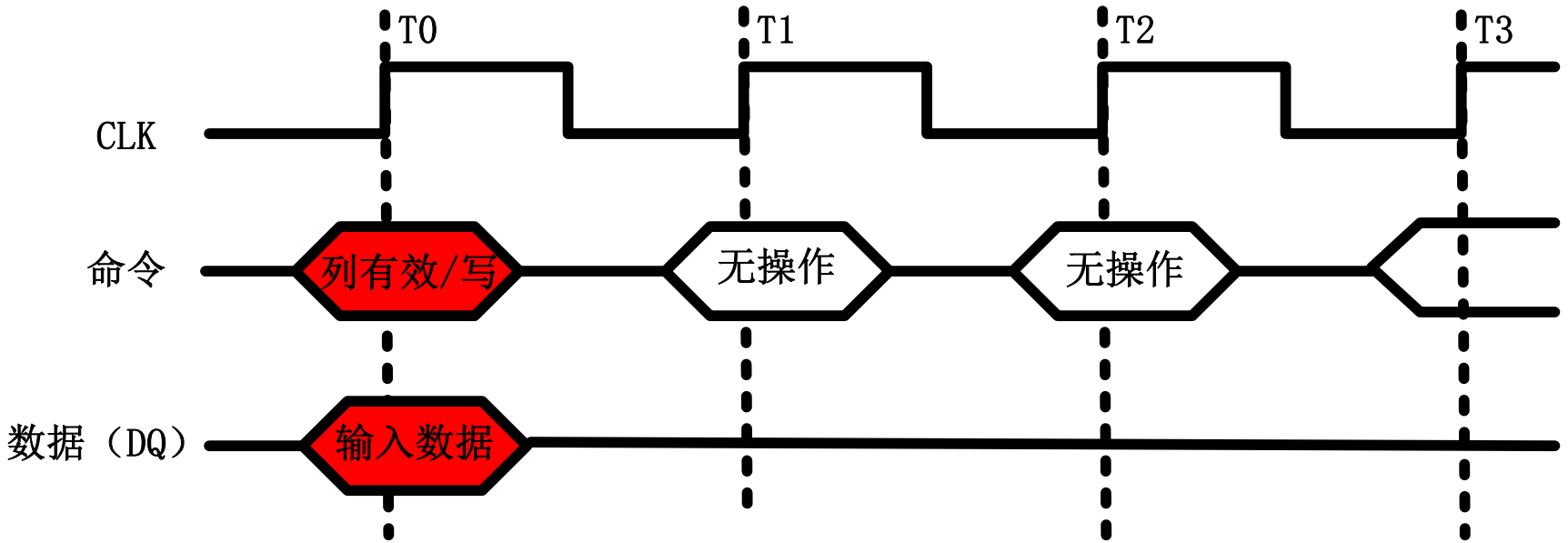
▶ 6.2.3 同步DRAM（SDRAM）

- 读写时序
  - 读
    - 对读操作，选定列地址后，数据输出到数据线上有一个延迟，即CAS Latency，用CL=周期数表示



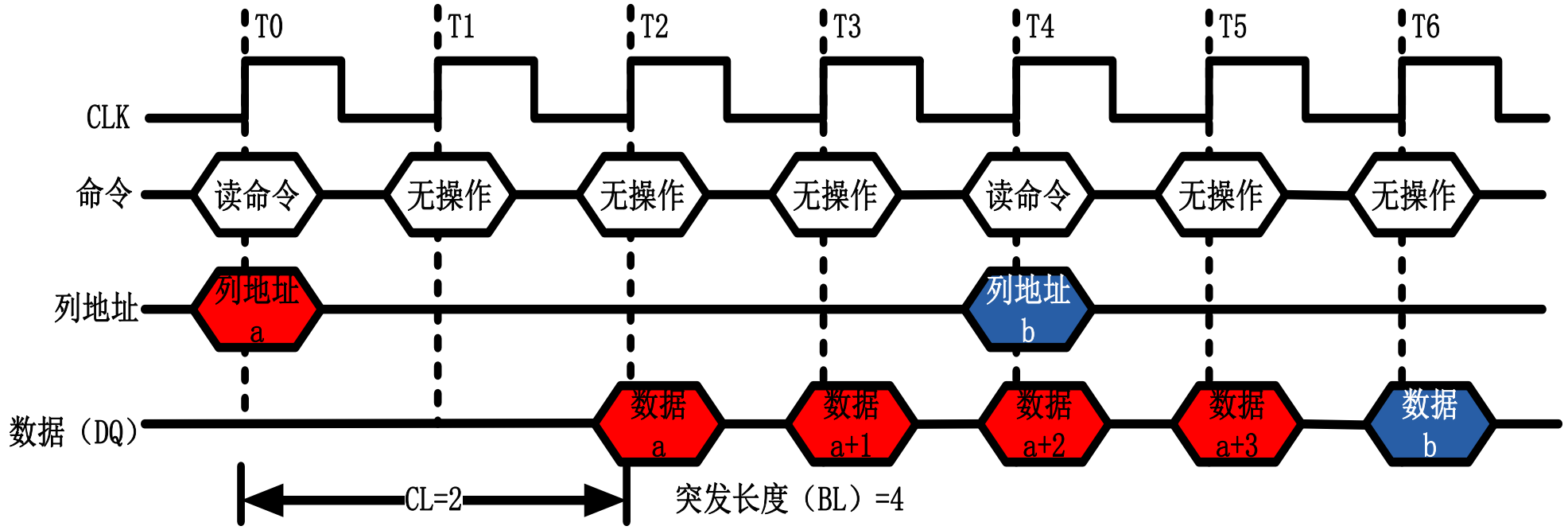
▶ 6.2.3 同步DRAM（SDRAM）

- 读写时序
  - 写
    - 写操作时，数据由CPU发到数据总线，可与CAS同时发生，不存在读操作中的CAS Latency



▶ 6.2.3 同步DRAM（SDRAM）

- 读写时序
  - 突发模式读



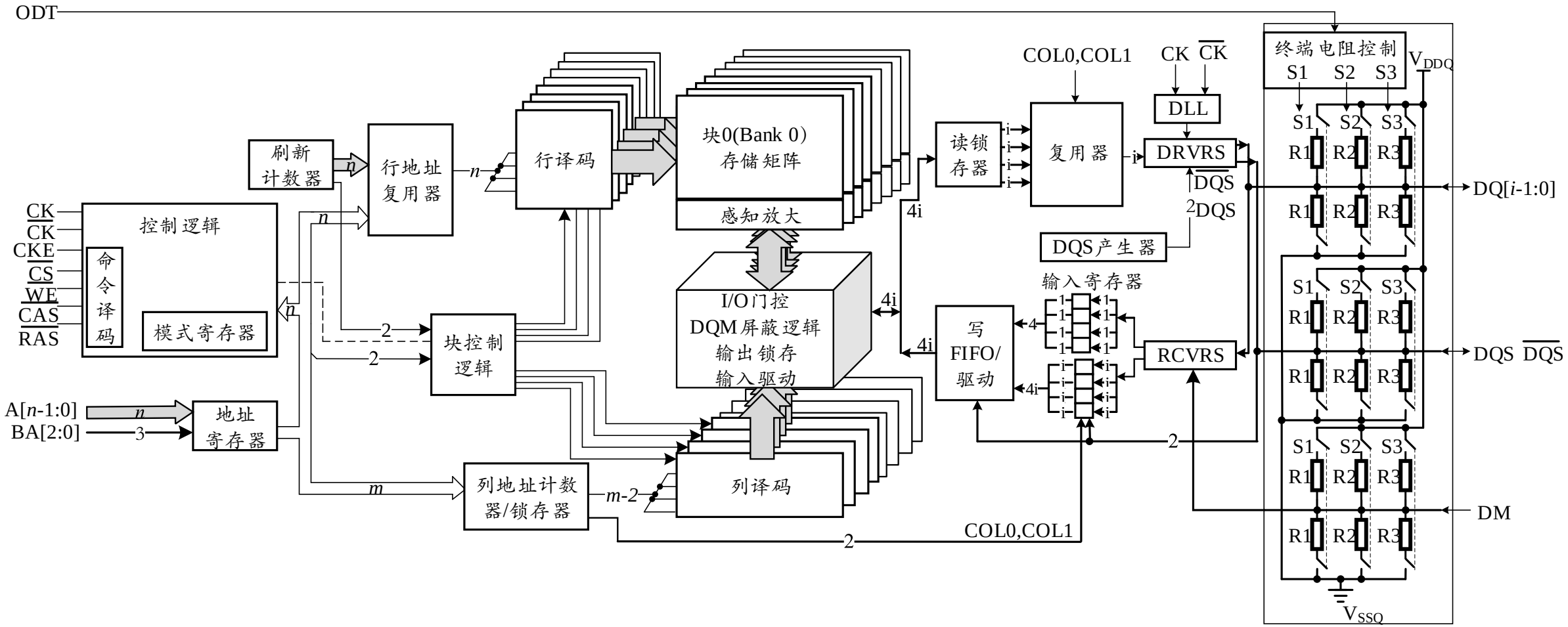
6.2 典型存储芯片

- 异步SRAM
- 同步SRAM（SSRAM）
- 同步DRAM（SDRAM）
- DDR2-SDRAM
- NOR FLASH
- NAND FLASH

6.2 典型存储芯片

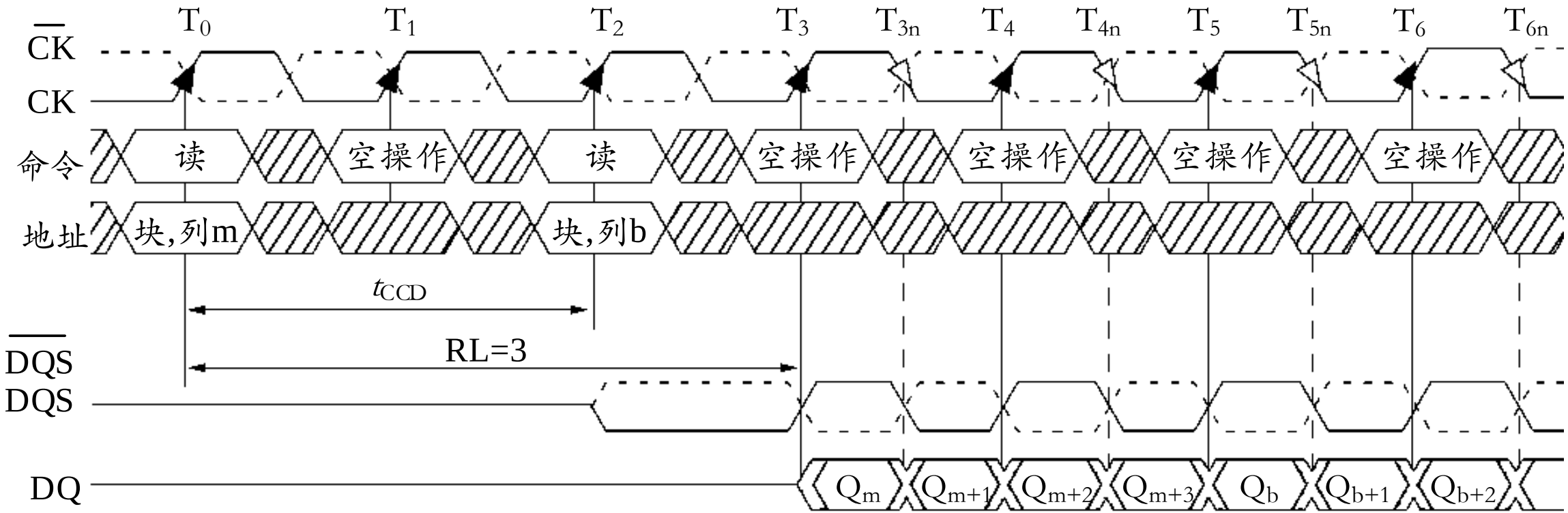
▶ 6.2.4 DDR-SDRAM

- DDR：Double Data Rate，双倍速率，在时钟上、下沿都能进行数据传输



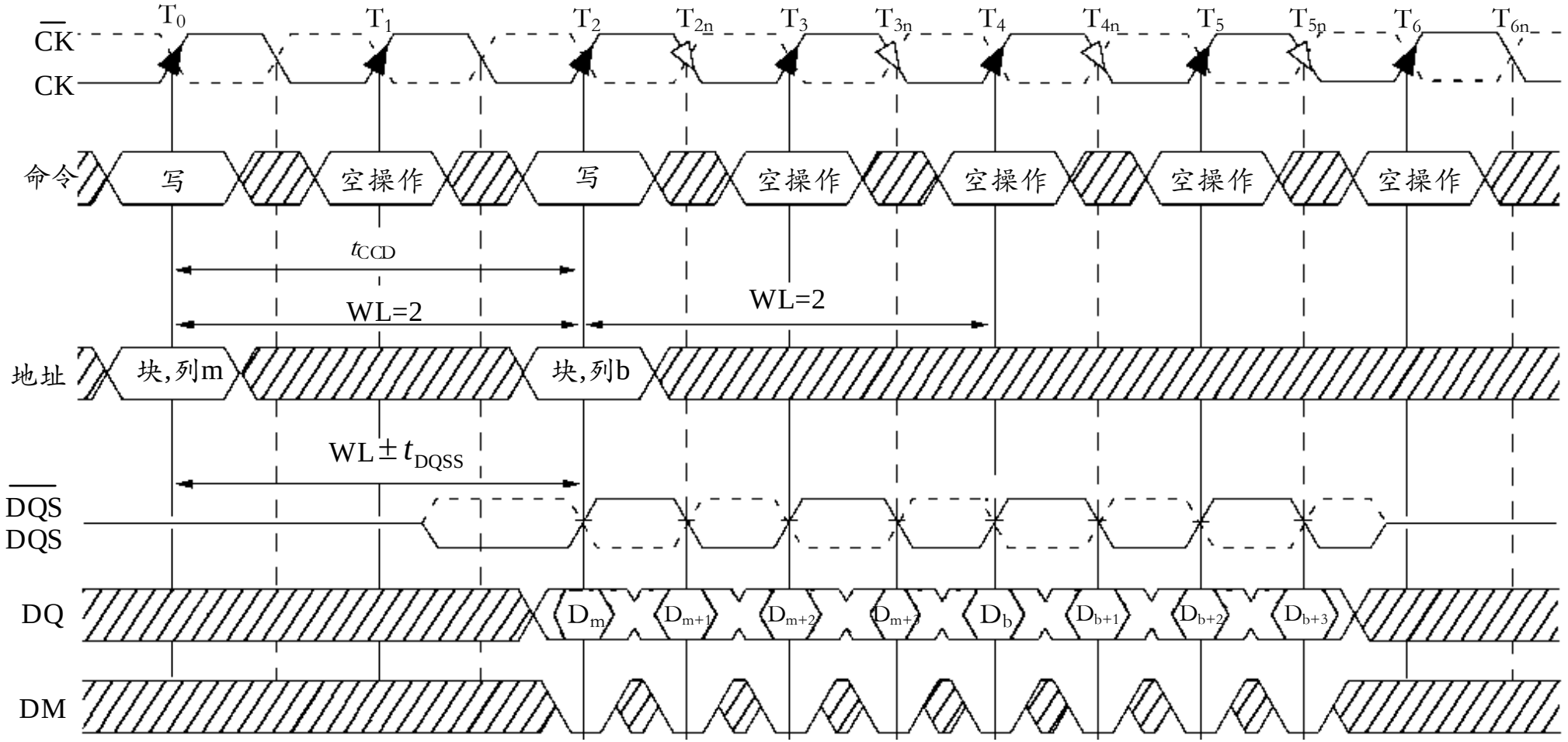
▶ 6.2.4 DDR-SDRAM

- 连续突发读时序



▶ 6.2.4 DDR-SDRAM

- 连续突发写时序

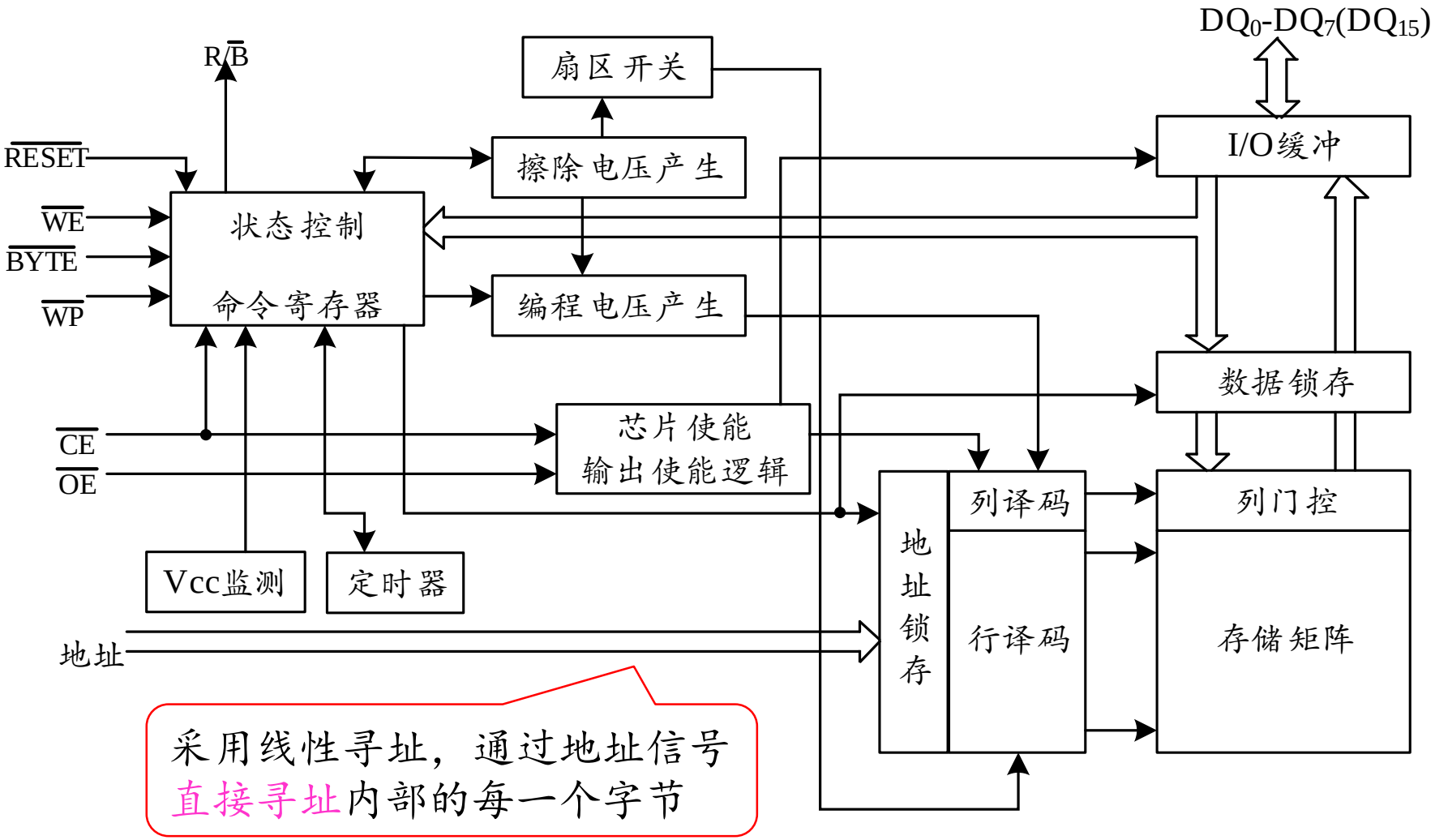


6.2 典型存储芯片

- 异步SRAM
- 同步SRAM（SSRAM）
- 同步DRAM（SDRAM）
- DDR2-SDRAM
- NOR FLASH
- NAND FLASH

▶ 6.2.5 Nor Flash

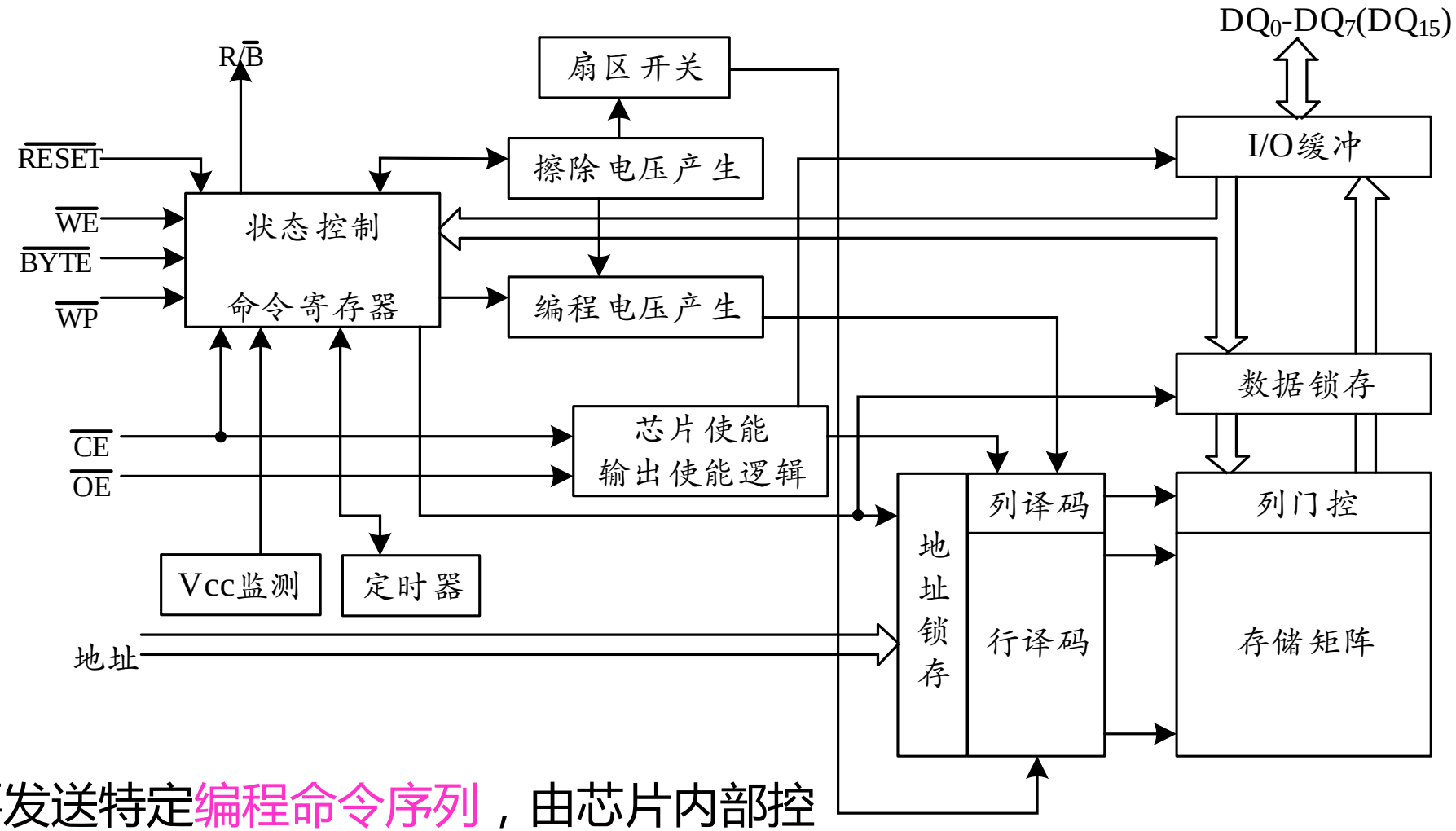
- 结构框图
  - 和SRAM类似
  - 多电压产生电路
- 读操作
  - 和SRAM类似
- 用途
  - 和NAND Flah相比，速度稍快，但容量稍小，主要存储bootloader，也叫Code Flash。



6.2 典型存储芯片

▶ 6.2.5 Nor Flash

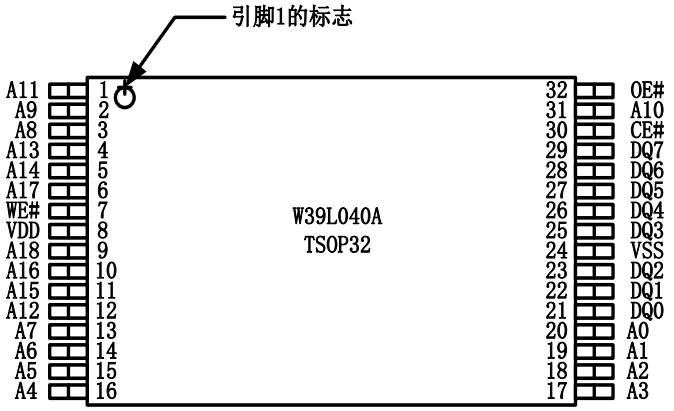
- 结构框图
  - 和SRAM类似
  - 多电压产生电路
- 读操作
  - 和SRAM类似
- 用途
  - 和NAND Flah相比，速度稍快，但容量稍小，主要存储bootloader。
- 写操作
  - 不能直接写，需要发送特定编程命令序列，由芯片内部控制电路自动完成写操作。且写入前需要先擦除原来存储值。



6.2 典型存储芯片

▶ 6.2.5 Nor Flash

- 结构框图
  - 和SRAM类似
  - 多电压产生电路
- 读操作
  - 和SRAM类似
- 用途
  - 和NAND Flah相比，速度稍快，但容量稍小，主要存储bootloader。
- 写操作
  - 不能直接写，需要发送特定编程命令序列，由芯片内部控制电路自动完成写操作。且写入前需要先擦除原来存储值。

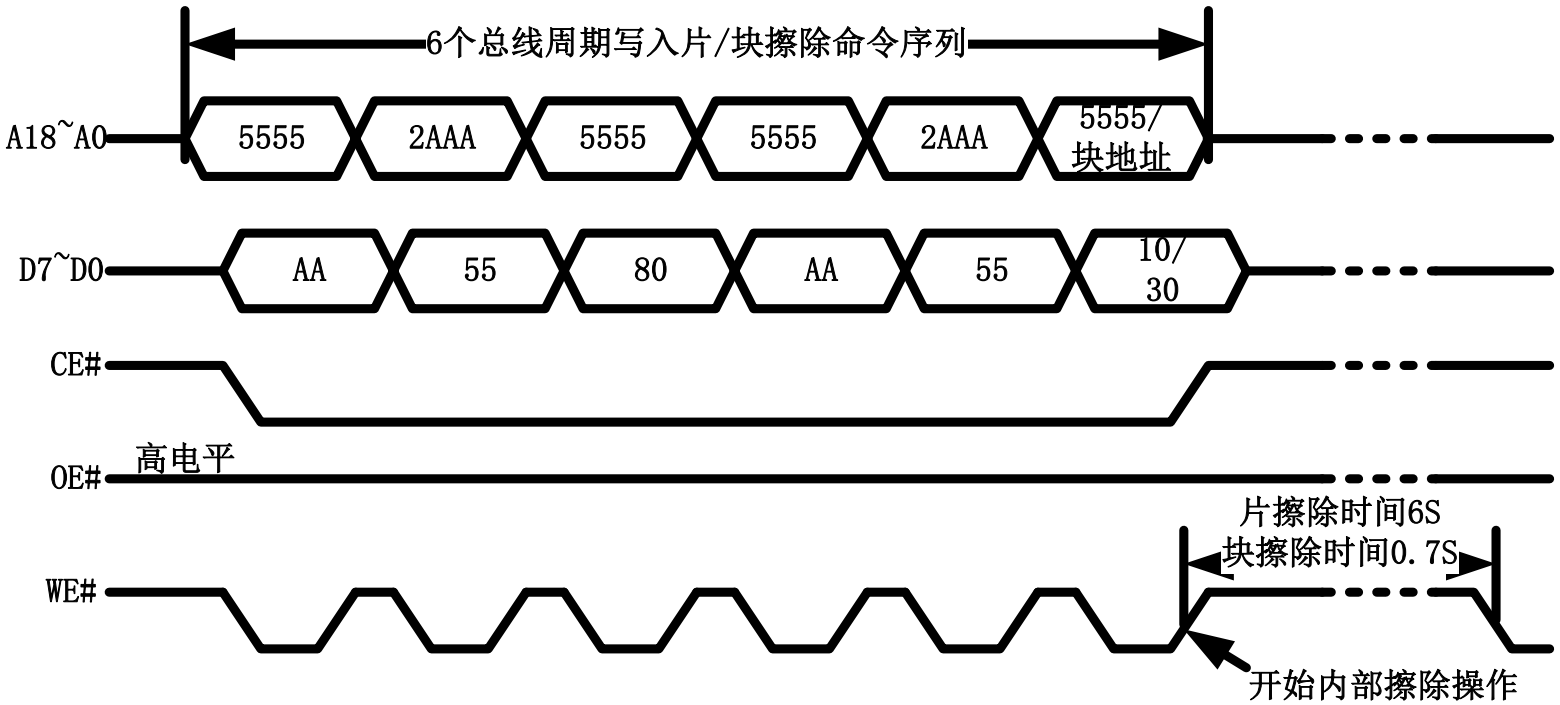


COMMAND DESCRIPTION	NO. OF Cycles	1ST CYCLE		2ND CYCLE		3RD CYCLE		4TH CYCLE		5TH CYCLE		6TH CYCLE	
		Addr. (1)	Data	Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data
Read	1	A_{IN}	D_{OUT}										
Chip Erase	6	5555	AA	2AAA	55	5555	80	5555	AA	2AAA	55	5555	10
Sector Erase	6	5555	AA	2AAA	55	5555	80	5555	AA	2AAA	55	SA(3)	30
Byte Program	4	5555	AA	2AAA	55	5555	A0	A_{IN}	D_{IN}				
Product ID Entry	3	5555	AA	2AAA	55	5555	90						
Product ID Exit (2)	3	5555	AA	2AAA	55	5555	F0						
Product ID Exit (2)	1	XXXX	F0										

▶ 6.2.5 Nor Flash

- 结构框图
  - 和SRAM类似
  - 多电压产生电路
- 读操作
  - 和SRAM类似
- 用途
  - 和NAND Flah相比，速度稍快，但容量稍小，主要存储bootloader。

擦除操作时序

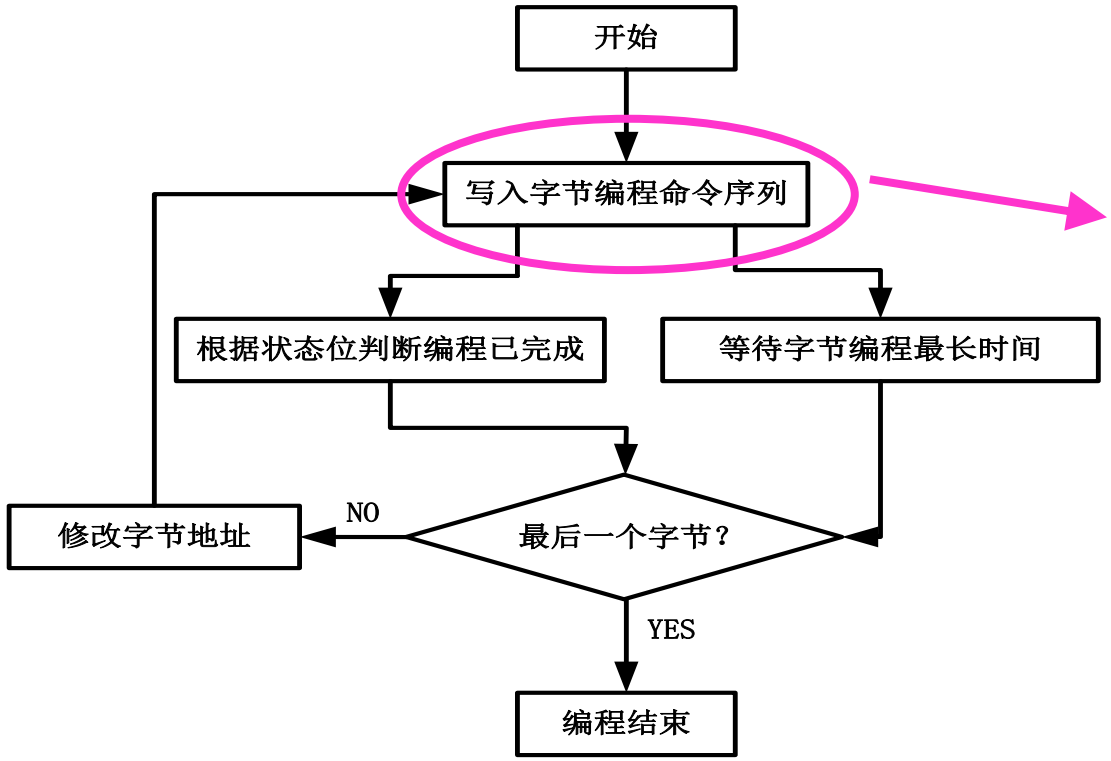


- 写操作
  - 不能直接写，需要发送特定编程命令序列，由芯片内部控制电路自动完成写操作。且写入前需要先擦除原来存储值。

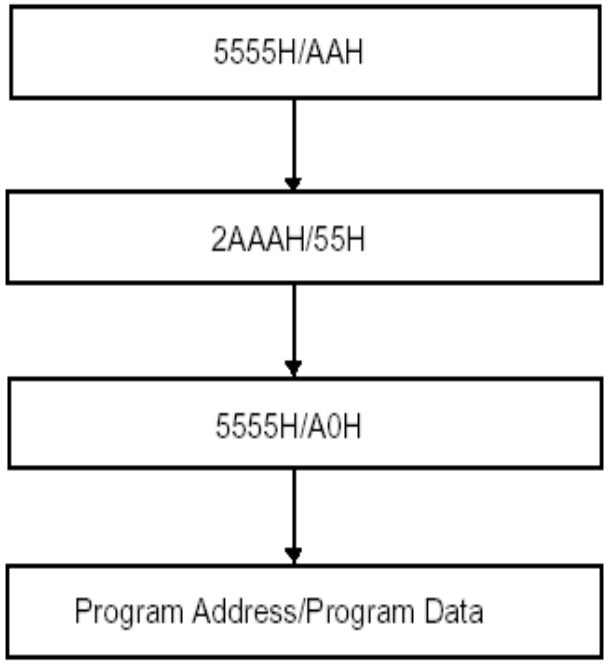
▶ 6.2.5 Nor Flash

- 结构框图
  - 和SRAM类似
  - 多电压产生电路
- 读操作
  - 和SRAM类似
- 用途
  - 和NAND Flah相比，速度稍快，但容量稍小，主要存储bootloader。
- 写操作
  - 不能直接写，需要发送特定编程命令序列，由芯片内部控制电路自动完成写操作。且写入前需要先擦除原来存储值。

数据烧写流程



写操作命令序列



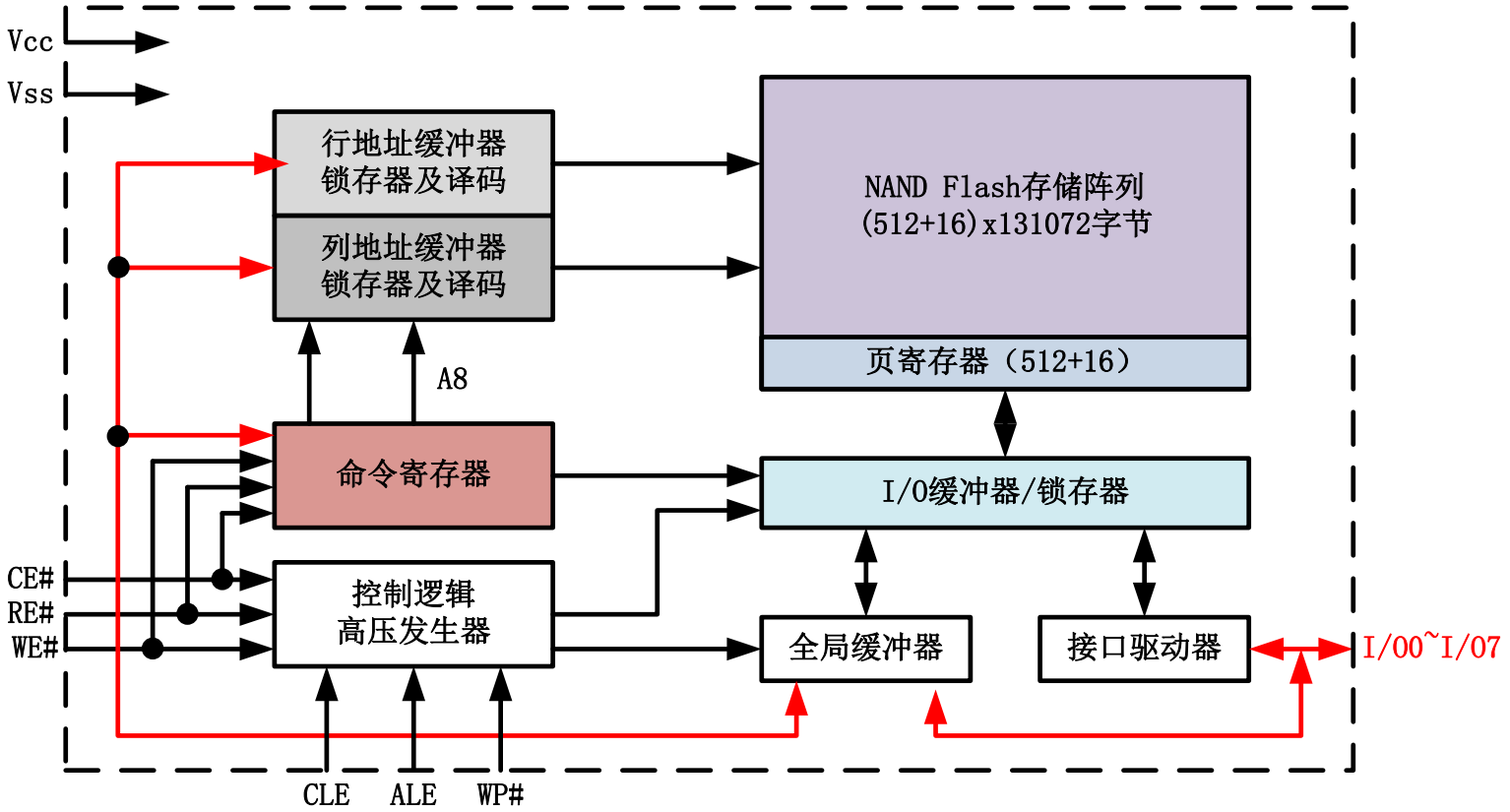
6.2 典型存储芯片

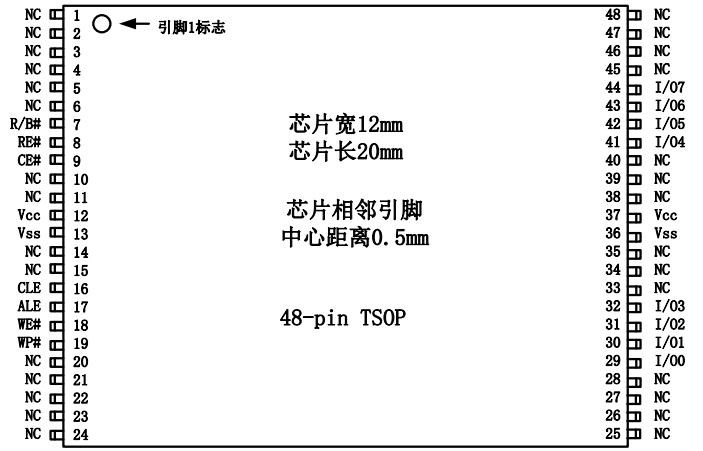
- 异步SRAM
- 同步SRAM（SSRAM）
- 同步DRAM（SDRAM）
- DDR2-SDRAM
- NOR FLASH
- NAND FLASH

▶ 6.2.6 NAND Flash：容量大，用于存储数据，Data Flash

- 结构框图
  - 命令、地址、数据线复用
    - 命令、地址、数据都由I/O引脚输入/输出

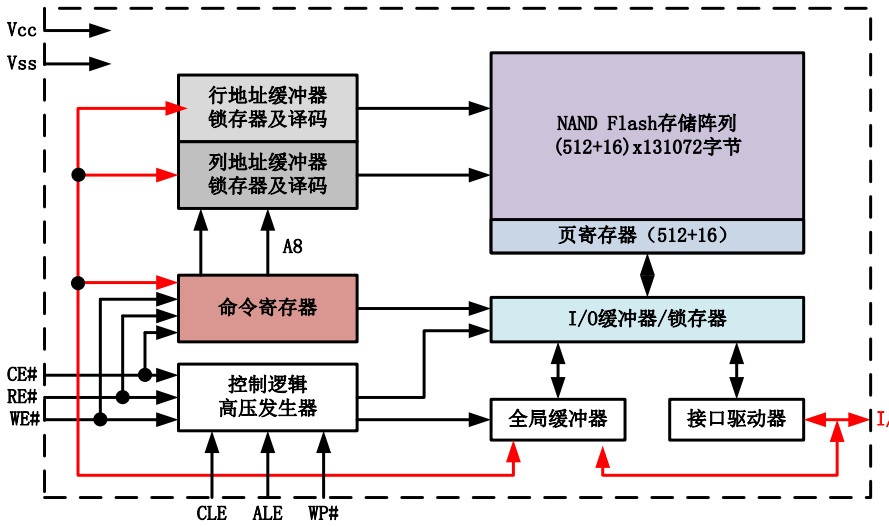
只有8位IO管脚，一个地址需要分多次写入

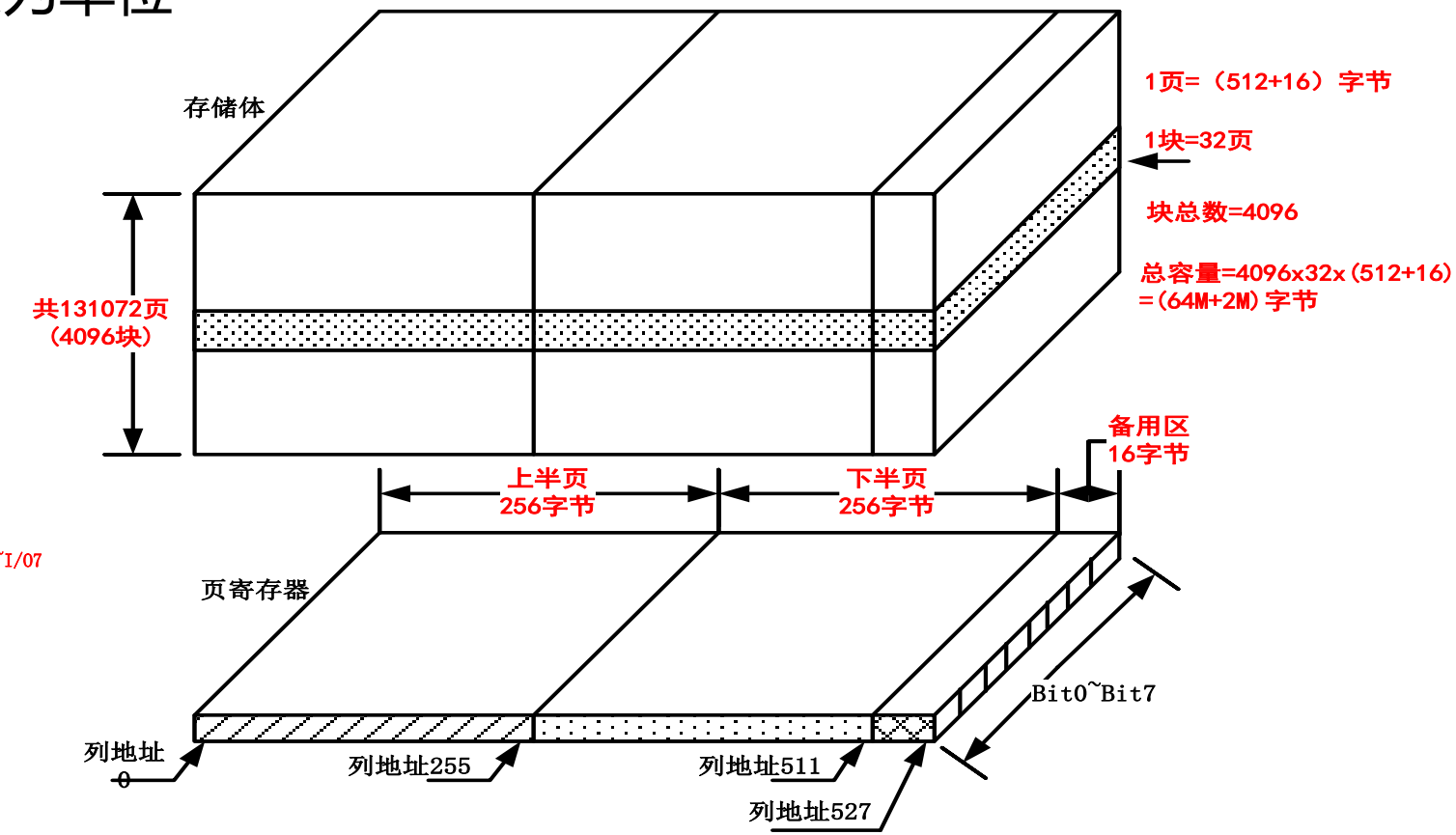




▶ 6.2.6 NAND Flash

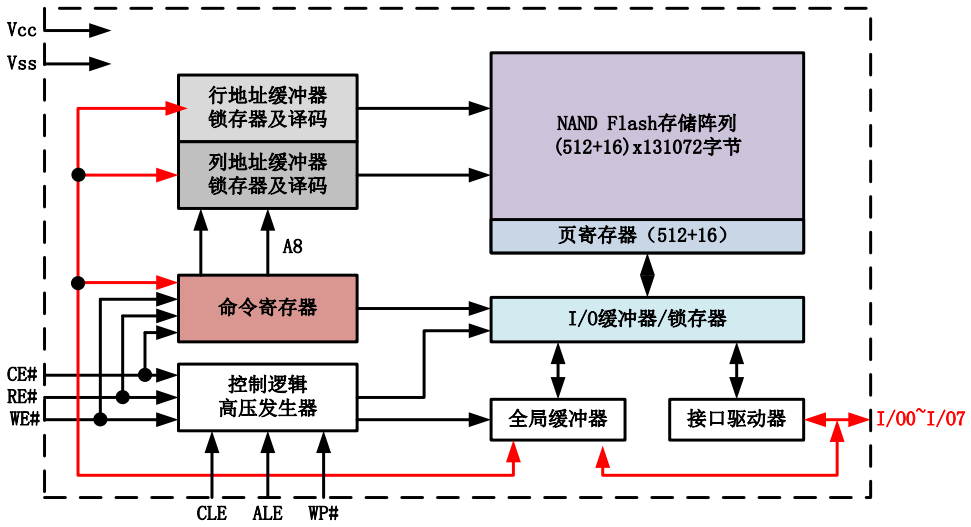
- 读写操作
  - 读、写以页为单位；擦除以块为单位





▶ 6.2.6 NAND Flash

- 读写操作
  - 读、写以页为单位；擦除以块为单位

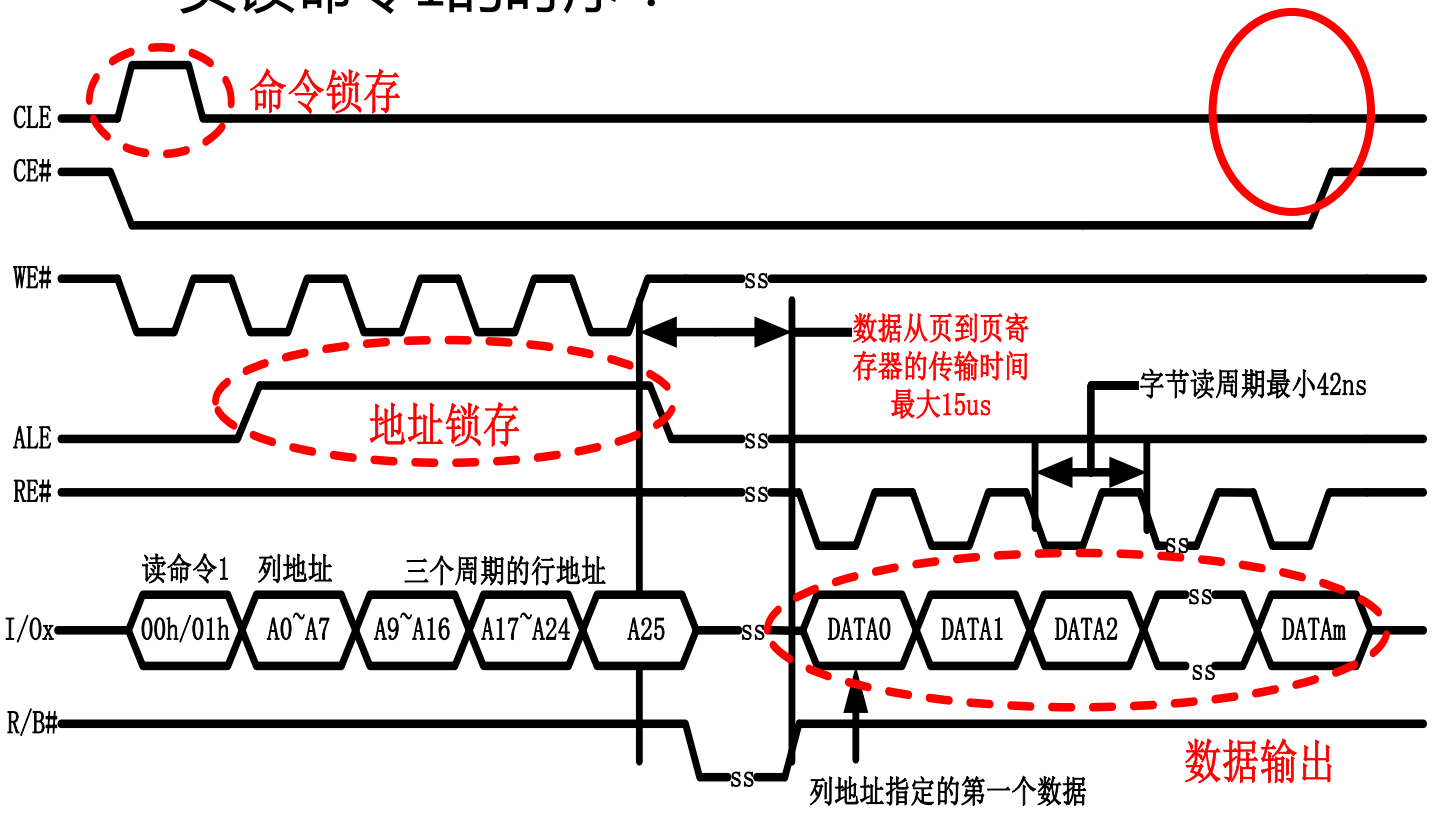


模式		CLE	ALE	$\overline{CE}$	$\overline{WE}$	$\overline{RE}$	$\overline{WP}$
读	命令输入	1	0	0	↑	1	X
	地址输入	0	1	0	↑	1	X
编程擦除	命令输入	1	0	0	↑	1	1
	地址输入	0	1	0	↑	1	1
数据输入		0	0	0	↑	1	1
数据输出		0	0	0	1	↓	X
数据输出暂停		X	X	X	1	1	X
读忙		X	X	X	1	1	X
编程忙		X	X	X	X	X	1
擦除忙		X	X	X	X	X	1
写保护		X	X	X	X	X	0
空闲		X	X	1	X	X	0V/V_{CC}

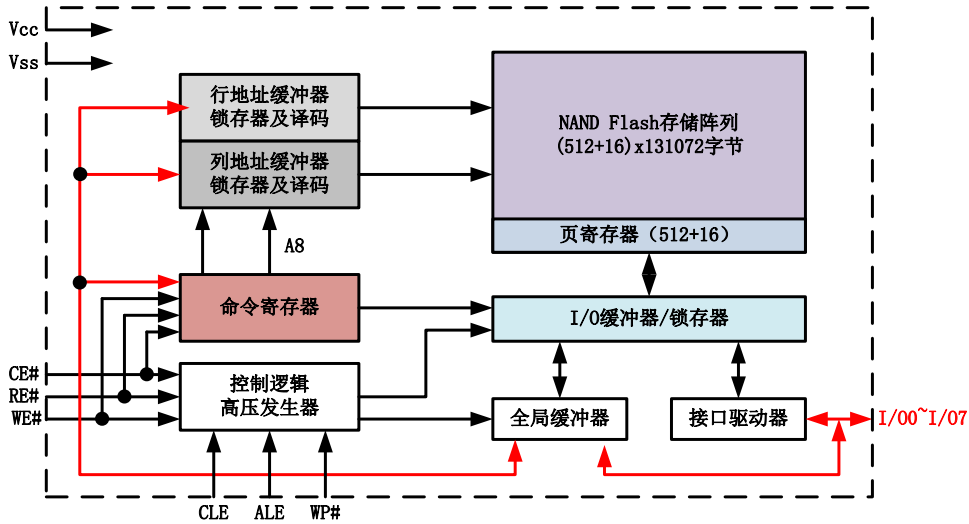
6.2 典型存储芯片

▶ 6.2.6 NAND Flash

- 读写操操作
  - 读、写以页为单位；擦除以块为单位
  - 页读命令1的时序：

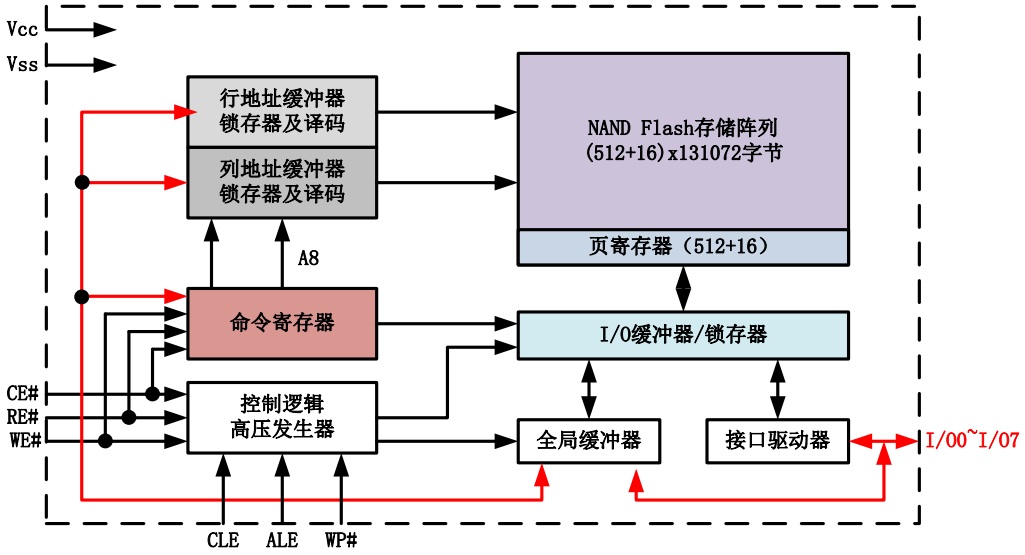


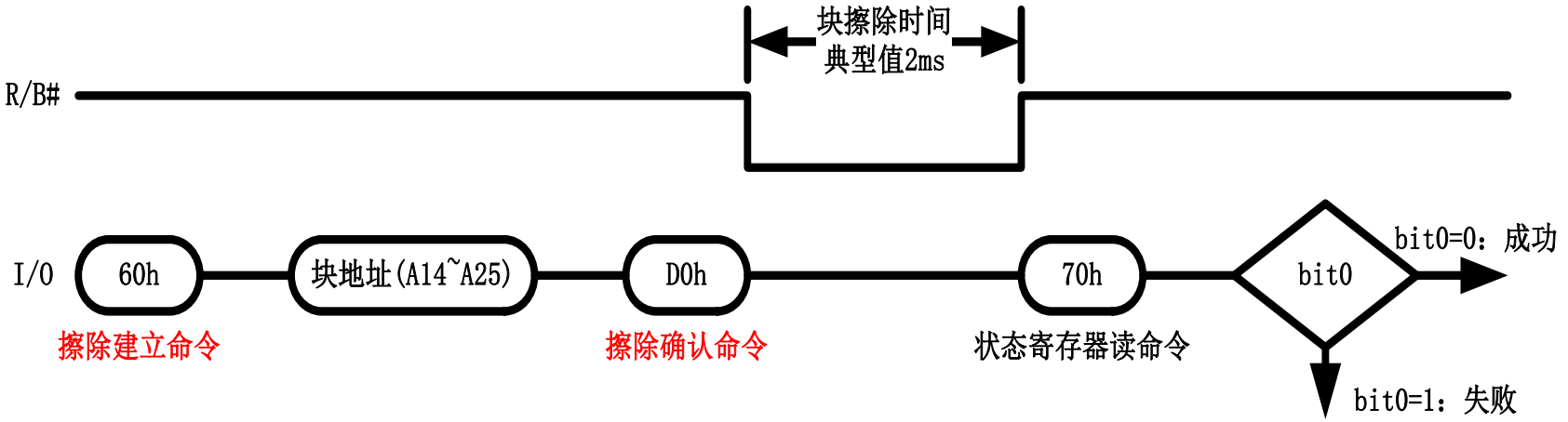
模式		CLE	ALE	$\overline{CE}$	$\overline{WE}$	$\overline{RE}$	$\overline{WP}$
读	命令输入	1	0	0	↑	1	X
	地址输入	0	1	0	↑	1	X
编程擦除	命令输入	1	0	0	↑	1	1
	地址输入	0	1	0	↑	1	1
数据输入		0	0	0	↑	1	1
数据输出		0	0	0	1	↓	X
数据输出暂停		X	X	X	1	1	X
读忙		X	X	X	1	1	X
编程忙		X	X	X	X	X	1
擦除忙		X	X	X	X	X	1
写保护		X	X	X	X	X	0
空闲		X	X	1	X	X	0V/V_{CC}



▶ 6.2.6 NAND Flash

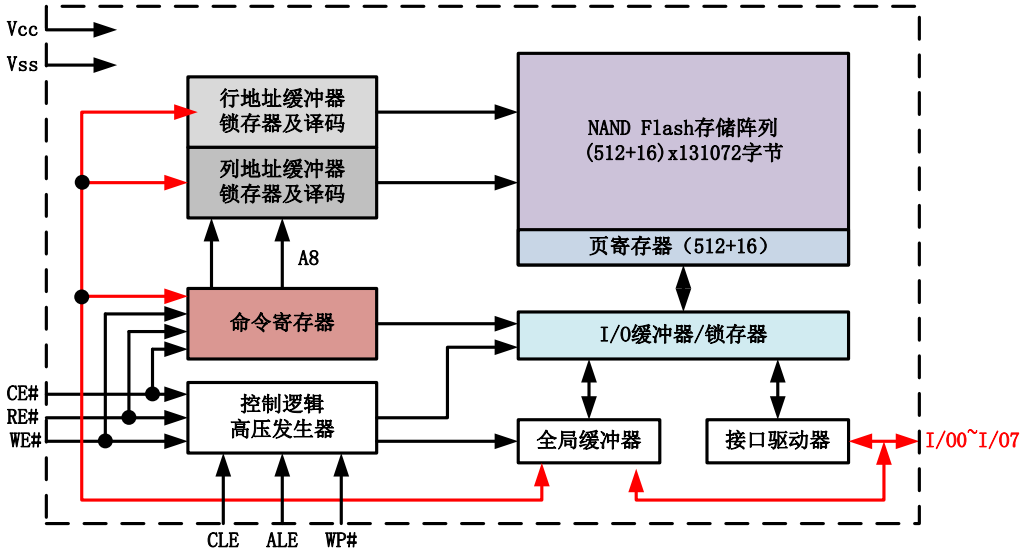
- 读写操操作
  - 读、写以页为单位；擦除以块为单位
  - 块擦除时序：

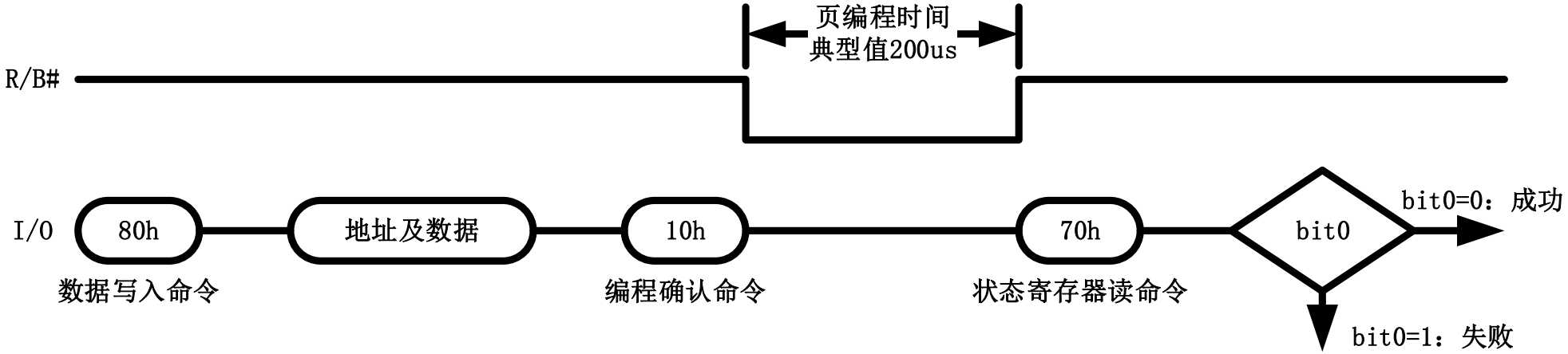




▶ 6.2.6 NAND Flash

- 读写操操作
  - 读、写以页为单位；擦除以块为单位
  - 页编程时序：



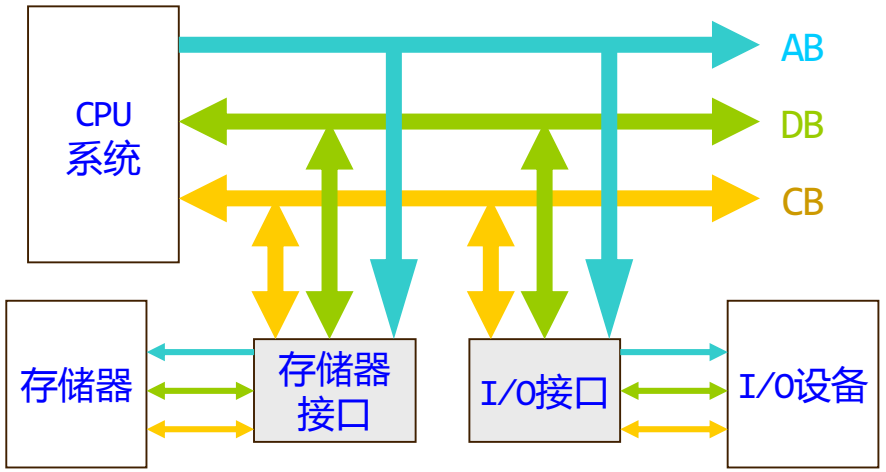


Agenda

- 内容
  - 半导体存储芯片分类
  - 典型存储芯片的结构特点、读写时序
  - 接口的基本概念
  - 存储器接口设计
- 目的
  - 了解存储芯片分类
  - 了解典型存储芯片的结构特点、读写时序
  - 了解接口的基本构成、数据传送方式以及控制方式
  - 掌握存储器容量、字长扩展接口设计
  - 掌握支持不同类型数据访问的存储器接口电路设计

6.3 接口的基本概念

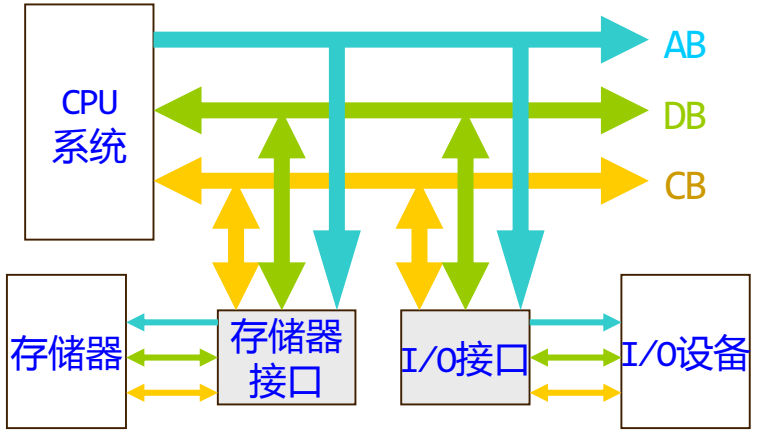
- 概念
  - 接口包含两方面的含义：为实现两个设备间数据交换的电子线路（硬件）及其通信协议（软件）
  - 接口并不局限在中央处理器与存储器或外设之间，也可在存储器与外设之间



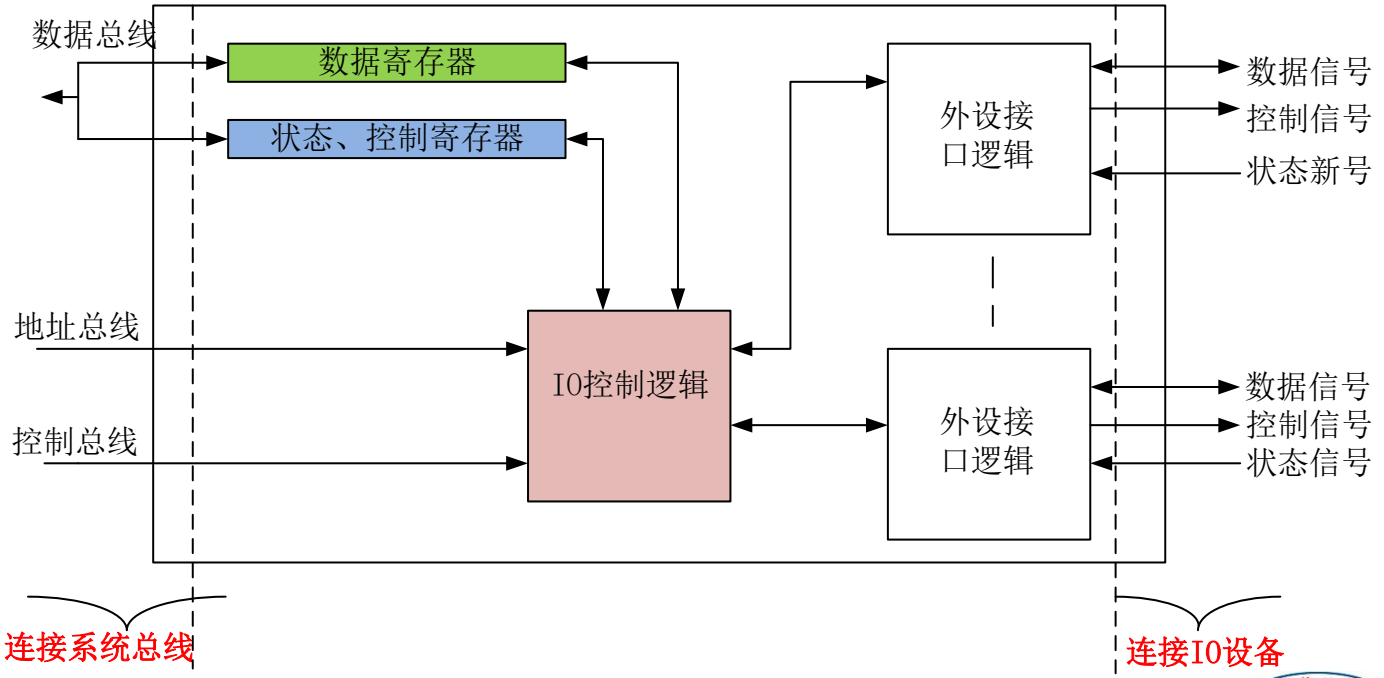
- 接口功能：任何接口通常都包括以下5种功能
  - 控制和定时
  - 与微处理器通信
  - 与外设通信
  - 数据缓冲
  - 错误检测

6.3 接口的基本概念

- CPU与接口之间通信包括以下几个方面：
  - 命令译码
  - 数据缓冲
  - 状态反馈
  - 地址译码



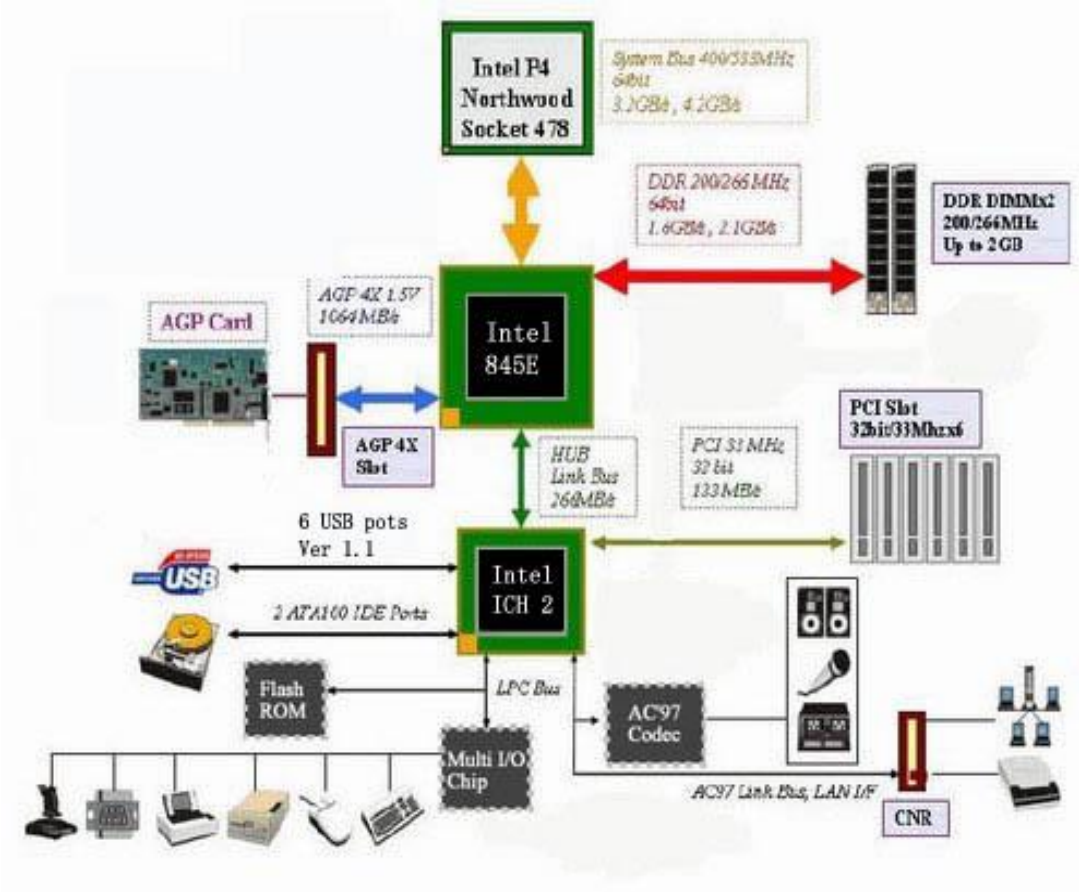
- 接口基本结构
  - 控制逻辑电路
  - 状态设置和存储电路
  - 数据存储和缓冲电



6.3 接口的基本概念

▶ 接口规范

- CPU和各种设备具有固定的接口规范（硬件、软件），如串口-RS232、PCI-PCI2.1、USB-USB1.1/2.0/USB3.0/USB3.1等，只要遵循相应的接口规范，用户就可以进行设备之间的接口开发。
- 现代PC系统中CPU和各种设备之间通过主板上的芯片组(Chipset)进行管理
- CPU和各种I/O设备的接口、CPU和存储器的接口都已高度集成到了芯片组芯片里

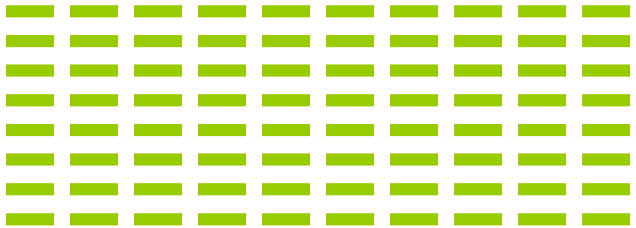


6.3 接口的基本概念

▶ 接口与外设间的数据传送方式

- 接口主要是起设备之间进行数据传送的桥梁作用，设备间数据传送方式有：
  - 并行数据传送
    - 并行数据的每一位都对应独立的传输线路
    - 如：ISA、PCI、AXI、IDE
    - 数据有多少位就要有同样数量的传送线，速度快，但连线多，只适用于短距离传送。

并行



  - 串行数据传送
    - 将构成字符的每个二进制数据位，按一定的顺序逐位进行传送
    - 如：RS232、IIC、SPI、USB
    - 连线简单、控制复杂、速度慢(USB很快)、距离远。
    - 每位电平的时间长短T则由通信双方事先约定

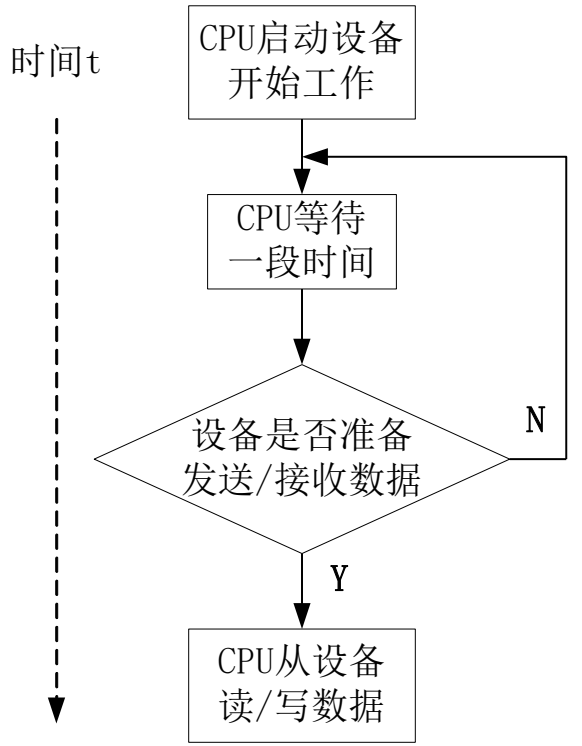
串行

▶ 接口控制方式

- 不同的设备，采用不同的控制方式实现数据的传送：
  - “无条件”传送方式
    - CPU与外设同步工作，CPU不需要查询外设的状态，直接与外设交换数据。此时接口简单，适用于 CPU 与外设同步工作的情况。
    - 更多情况下， CPU 与外设并不同步工作，很难保证CPU在执行读/写操作时，外设是准备好的，所以在数据传送前，CPU需要首先通过下面两种方式(查询、中断)知晓外设的状态，只有在外设准备好时，才能进行传送。
  - 查询
  - 中断
  - DMA

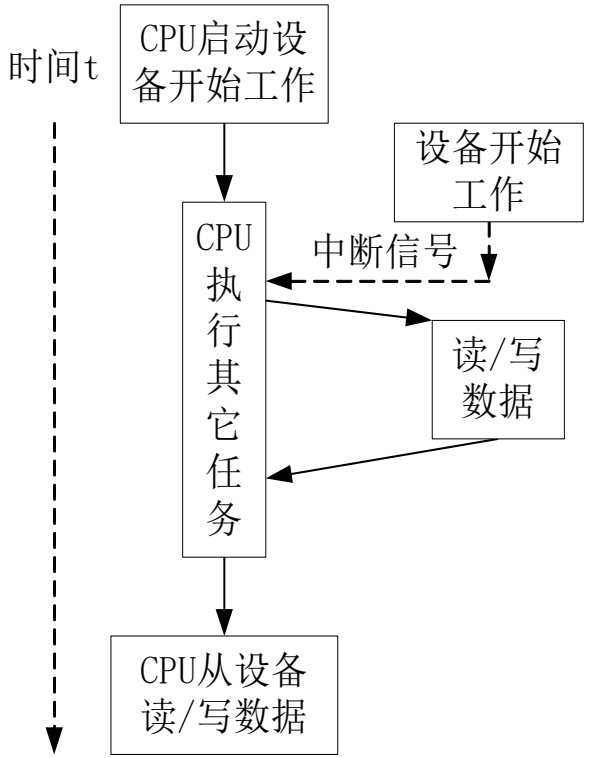
▶ 接口控制方式

- 不同的设备，采用不同的控制方式实现数据的传送：
  - 查询
    - 查询方式中，CPU处于主动地位，CPU启动设备开始工作后，CPU需要不停地查询设备的状态，不能做别的操作，只有设备准备好之后才能读/写数据；
      - ▶在数据传送之前通过接口的状态查询电路询问外设，只有外设允许传送数据后才能传送数据。
    - 特点
      - ▶CPU运行效率不高(CPU要等待)；
      - ▶ 实时性不好(已准备好却没及时查到)；
    - 查询方式只用于对实时性要求不高的场合；
    - 为了提高CPU运行效率，保证系统实时性，通常采用中断方式。



▶ 接口控制方式

- 不同的设备，采用不同的控制方式实现数据的传送：
  - 中断
    - 外设要与CPU传送数据时，外设向CPU发出请求，CPU响应后再传送数据
    - 中断方式中，CPU处于被动地位，CPU启动设备开始工作后，CPU执行其它的程序，设备一旦准备发送/接收数据就会事先向CPU发送一个中断请求信号，CPU接收到中断请求信号后，停止当前的操作，保护好现场，进入中断服务子程序读/写设备的数据，完成后恢复现场，继续执行原来的程序；
    - 特点
      - ▶CPU运行效率高(CPU无需等待)；
      - ▶ 实时性好(有中断就读/写)；
    - 不少接口芯片既可用查询方式，又可用中断方式，可以根据实际情况灵活设计。

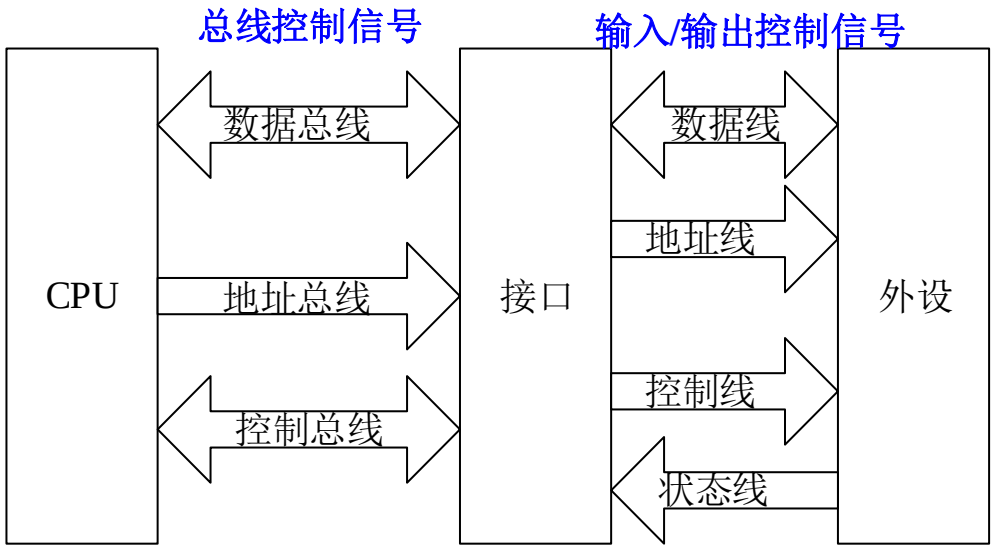


6.3 接口的基本概念

▶ 接口控制方式

- 不同的设备，采用不同的控制方式实现数据的传送：
  - DMA
    - 数据不经过CPU在存储器和外设之间直接传送
    - DMA方式的基本思想是在外设和存贮器之间开辟直接的数据交换通道，由DMA 硬件装置控制，直接由外设和内存之间传送数据，而不通过CPU。通常用于高速外设、成批交换数据。
    - 如现代PC中光驱、硬盘与内存交换数据，就使用了 DMA传输控制方式：DMA33/66/100/133
  - 实际应用中，几种方式均可使用。如 A /D 实验中：延迟等待（无条件传送）、查询、中断，若使用高速A /D 也可用 DMA 方式直接将数据送内存。

- 接口控制方式
  - 接口信号
    - 无论什么控制方式，接口在硬件电路实现上，都是通过信号来实现控制
    - 信号可以分为两大类
      - 总线信号
      - 输入/输出控制信号



Agenda

- 内容
  - 半导体存储芯片分类
  - 典型存储芯片的结构特点、读写时序
  - 接口的基本概念
  - 存储器接口设计
- 目的
  - 了解存储芯片分类
  - 了解典型存储芯片的结构特点、读写时序
  - 了解接口的基本构成、数据传送方式以及控制方式
  - 掌握存储器容量、字长扩展接口设计
  - 掌握支持不同类型数据访问的存储器接口电路设计

6.4 存储器接口设计

► 存储器接口设计需解决的问题

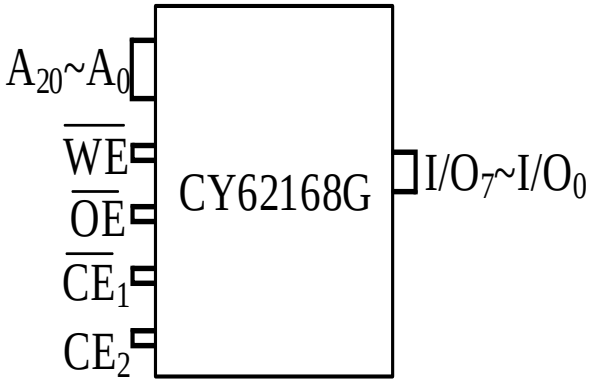
- (1) 存储容量扩展：由小容量存储芯片构建大容量的存储器
- (2) 存储空间映射：将物理存储空间映射到指定的逻辑存储空间
  - 计算机系 统微处理器能访问 的 存储空 间 通常称为 逻辑存储空 间
    - 如 32 位微处理器能访问 的逻辑存储空间 为 4GB,
  - 由 存储芯片 构成的 存储空 间 称为 物理存储空间
    - 如一片 1GB 存储芯片物理存储空间 仅为 1GB。
- (3) 不同类型数据访问兼容：低位宽存储芯片构建统一的支持多种不同类型数据访问的存储器
- (4) 总线操作时序匹配：总线与存储芯片的操作时序匹配
  - 由专门模块（CPU内部的各种存储器控制模块）完成

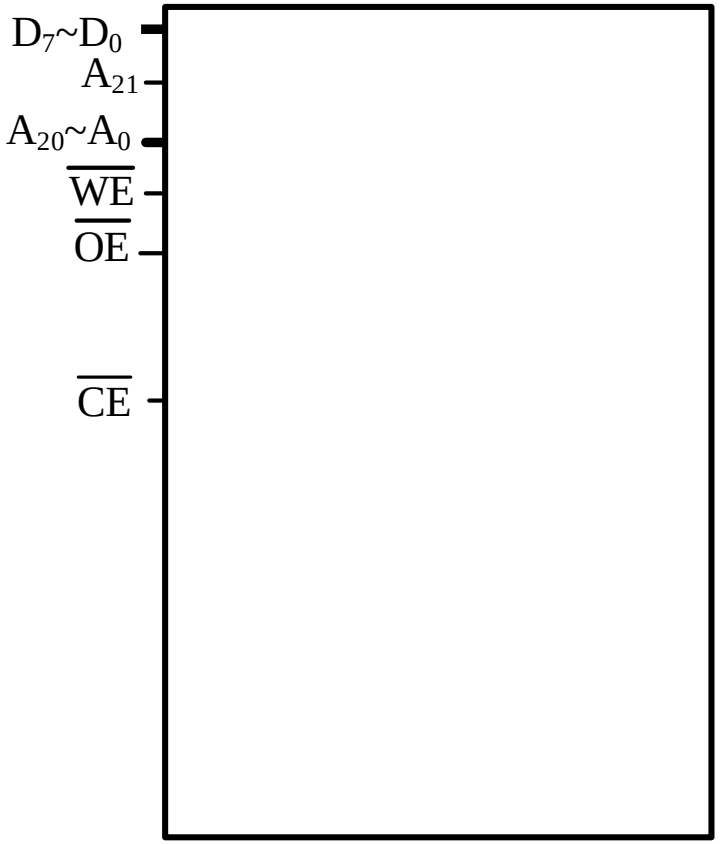
▶ 6.4.1 存储器容量扩展

- 字数扩展
  - 可寻址存储单元数增多，地址线增多
- 字长扩展
  - 存储单元位宽增大，数据线增多
- 字长、字数联合扩展
  - 地址线、数据线均增多

▶ 6.4.1 存储器容量扩展

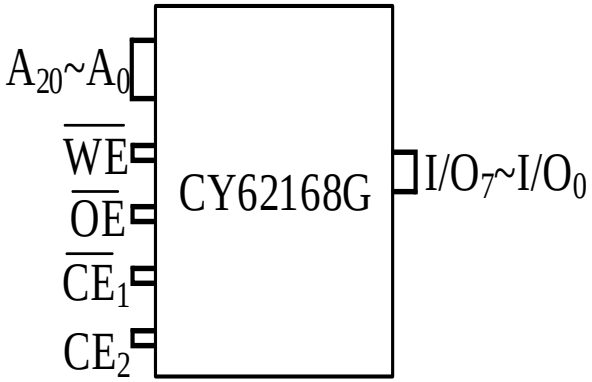
- 字数扩展举例：
  - 例6.1 基于异步SRAM存储芯片CY62168G(2M×8b）设计一容量为4M×8b的存储器

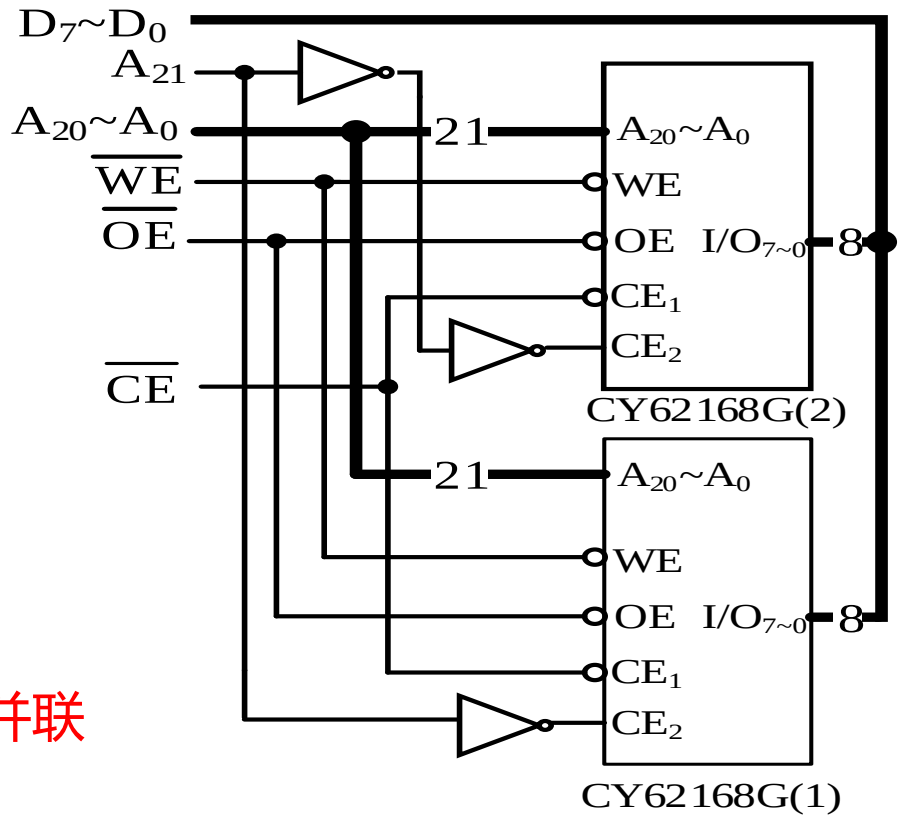




▶ 6.4.1 存储器容量扩展

- 字数扩展举例：
  - 例6.1 基于异步SRAM存储芯片CY62168G(2M×8b）设计一容量为4M×8b的存储器

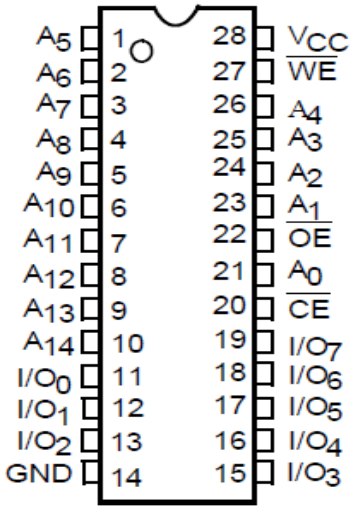


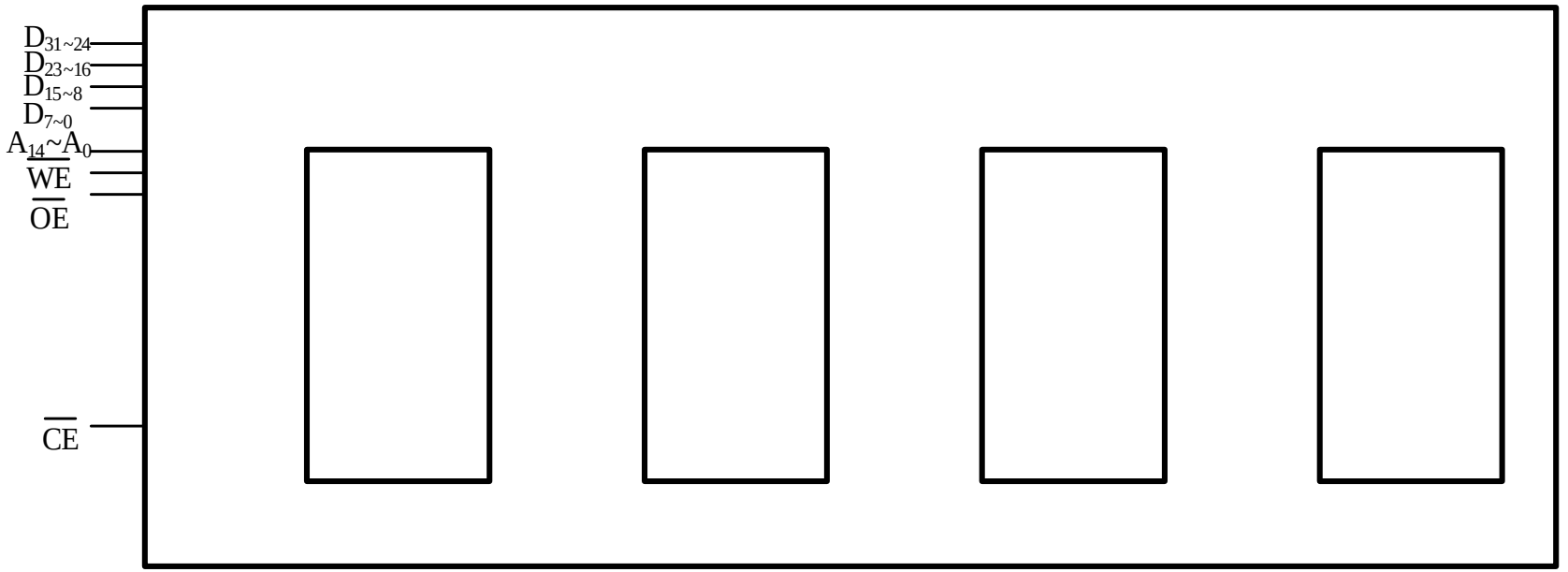


- 需存储芯片数为：$\frac{4M \times 8b}{2M \times 8b} = 2$
- 芯片的数据线、地址线、读写控制线：并联
- 增加一位地址，选择两个芯片的片选

▶ 6.4.1 存储器容量扩展

- 字长扩展举例：
  - 用异步SRAM存储芯片62256（32K x 8b）设计一个32K×32b的存储器

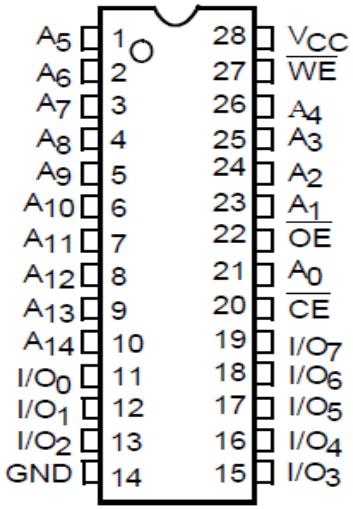


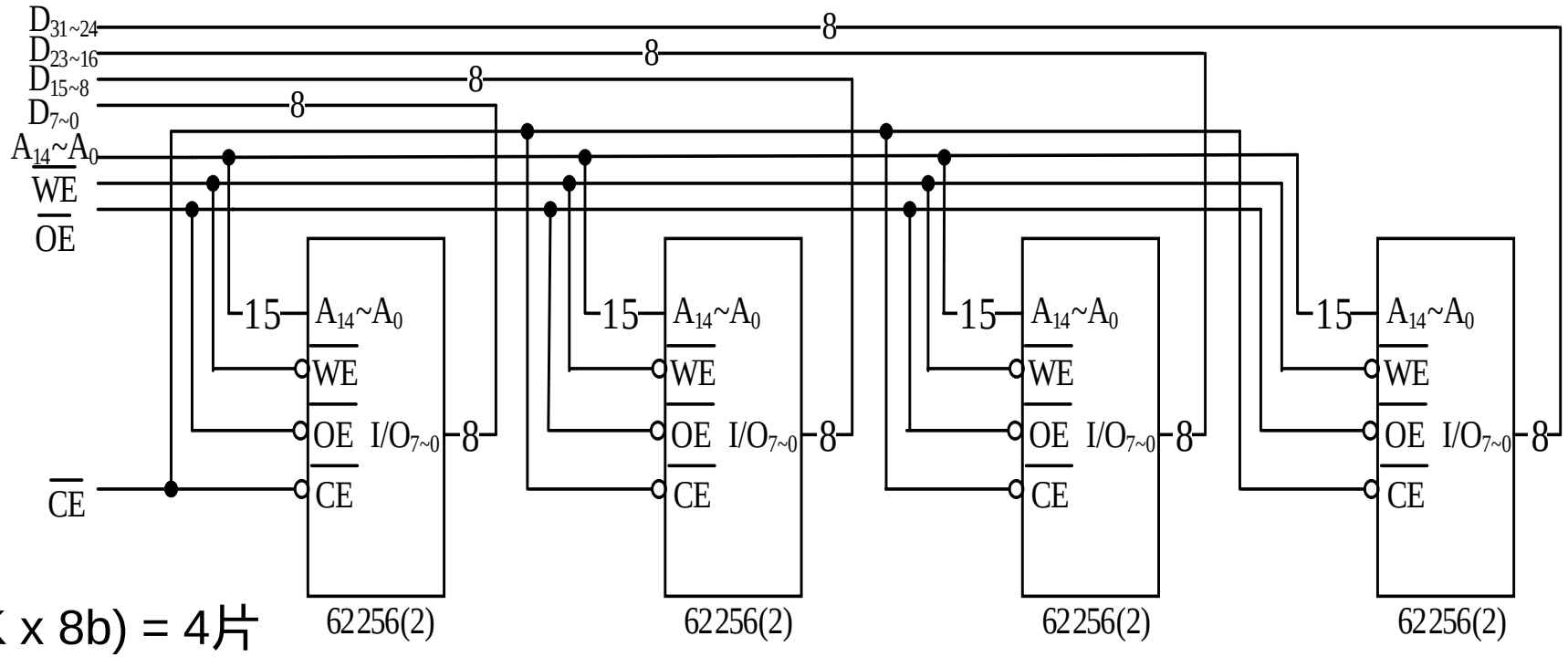


- 32K x 32b / (32K x 8b) = 4片
  - ▶地址线、片选线、控制线并联
  - ▶每片芯片8位数据线各自连接不同4组8位数据线

▶ 6.4.1 存储器容量扩展

- 字长扩展举例：
  - 用异步SRAM存储芯片62256（32K x 8b）设计一个32K×32b的存储器

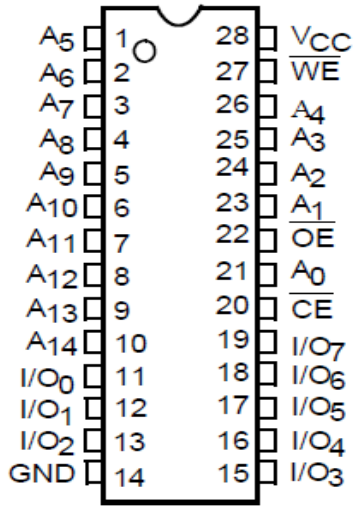


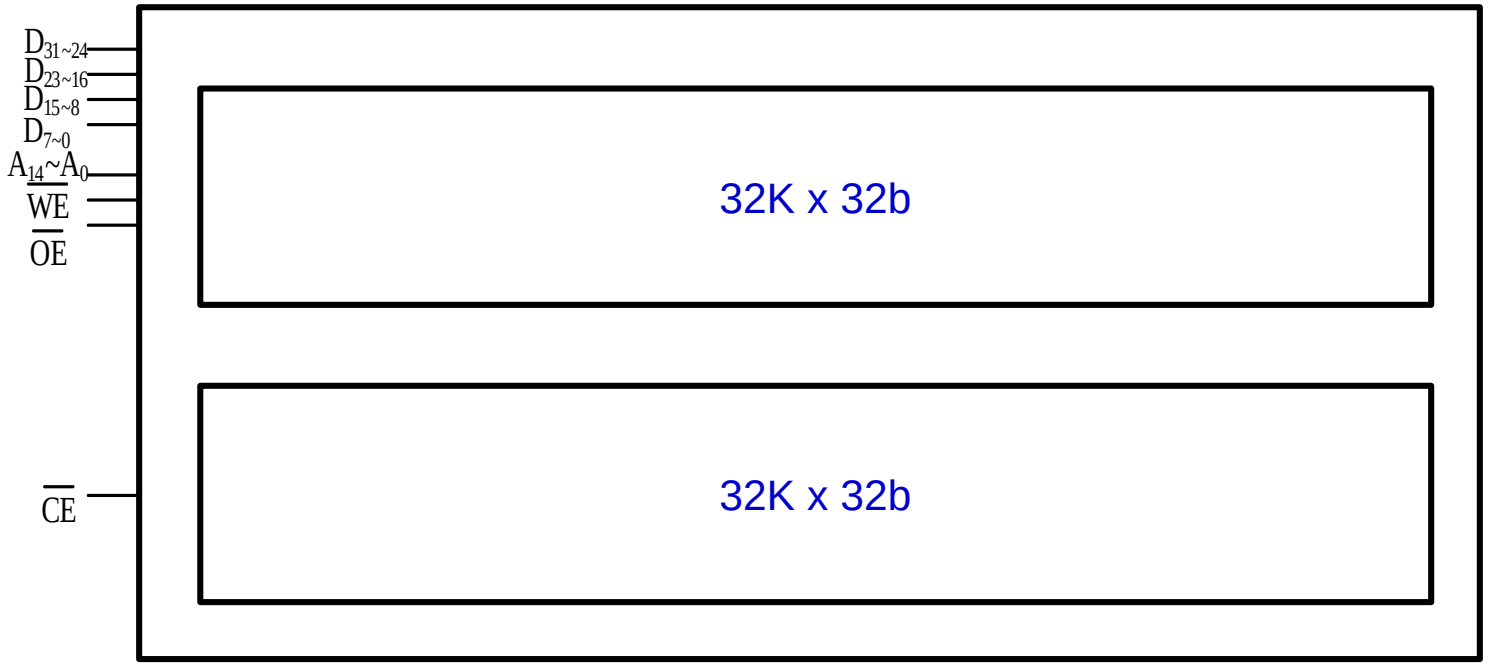


- 32K x 32b / (32K x 8b) = 4片
  - ▶地址线、片选线、控制线并联
  - ▶每片芯片8位数据线各自连接不同4组8位数据线

▶ 6.4.1 存储器容量扩展

- 字数、字长扩展举例：
  - 例6.2 用异步SRAM存储芯片62256（32K x 8b）设计一个64K×32b的存储器

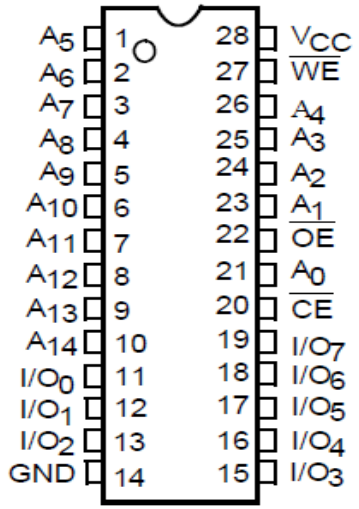




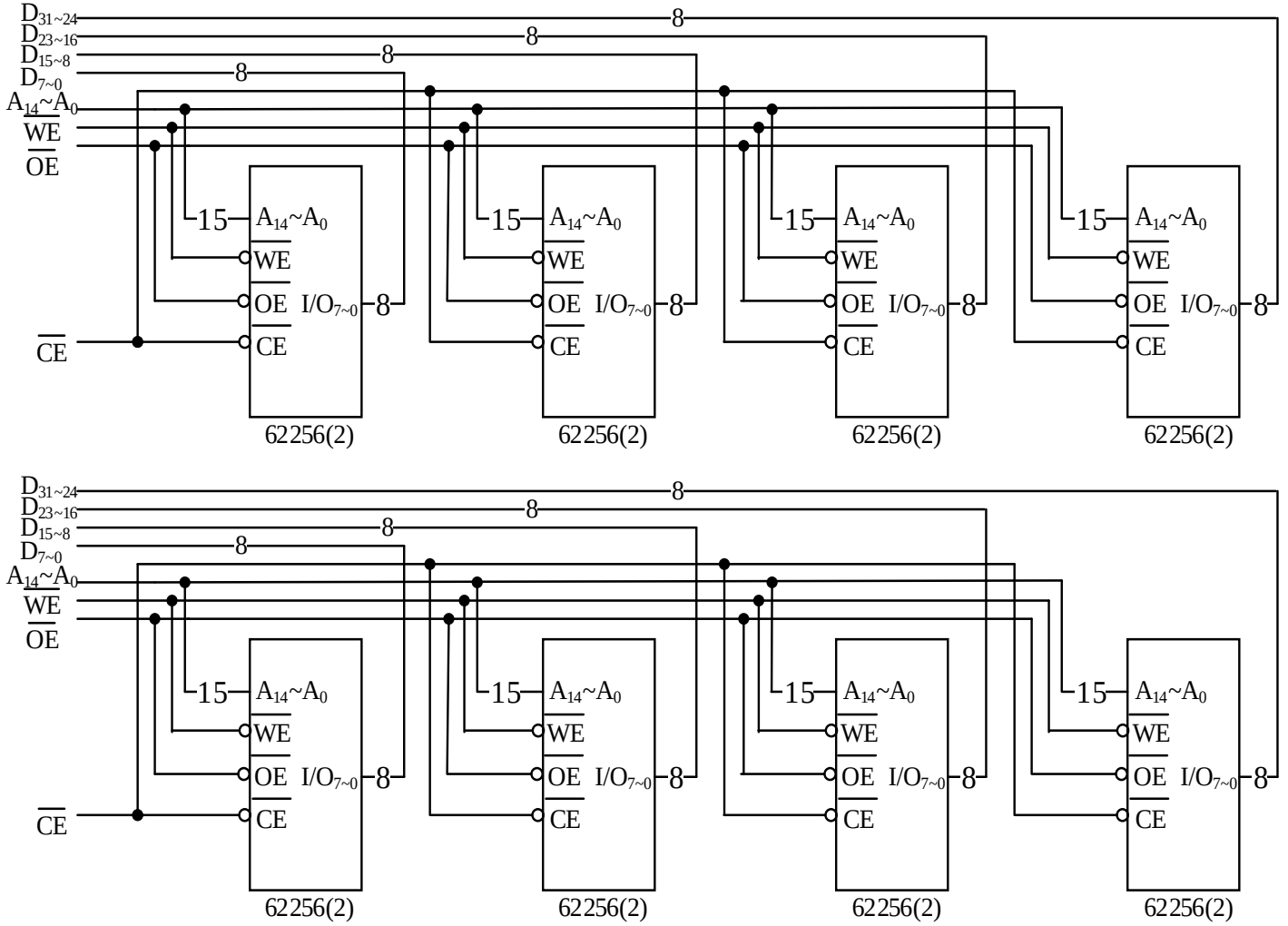
  - 先字长扩展：32K x 32b / (32K x 8b) = 4片，4片构成一组32K x 32b数据宽度存储器
  - 再字数扩展：64K x 32b / (32K x 32b) = 2组

▶ 6.4.1 存储器容量扩展

- 字数、字长扩展举例：
  - 例6.2 用异步SRAM存储芯片62256（32K x 8b）设计一个64K×32b的存储器



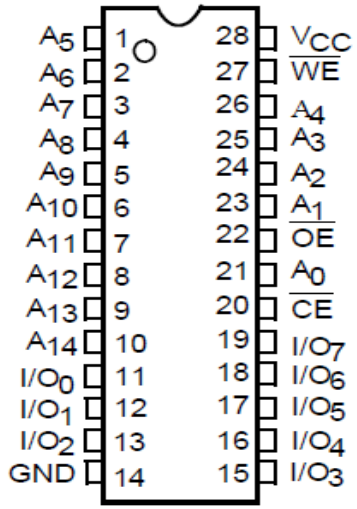
- 先字长扩展：4片
- 再字数扩展：2组

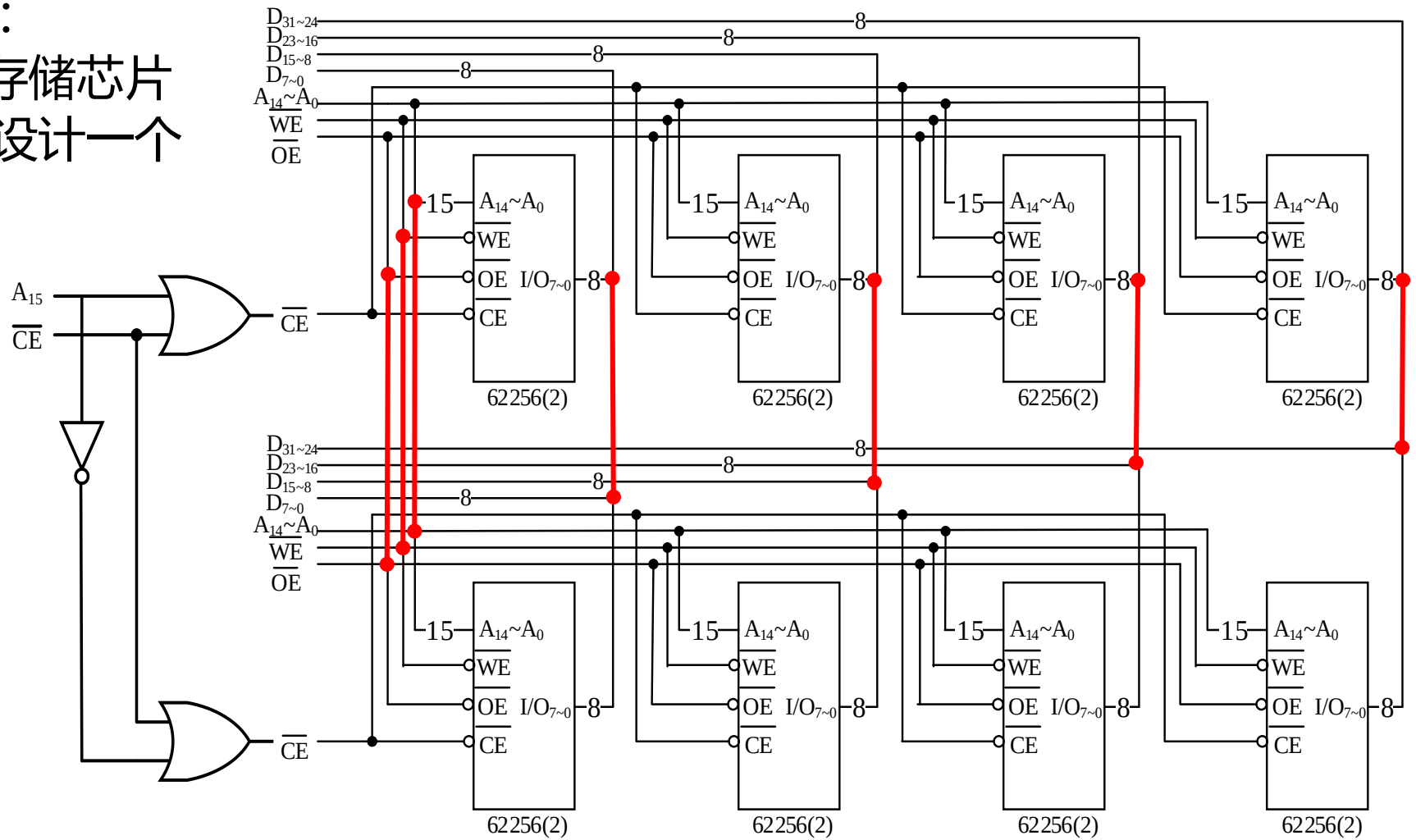


6.4 存储器接口设计

▶ 6.4.1 存储器容量扩展

- 字数、字长扩展举例：
  - 例6.2 用异步SRAM存储芯片62256（32K x 8b）设计一个64K×32b的存储器





- 先字长扩展：4片
- 再字数扩展：2组

▶ 存储器接口设计需解决的问题

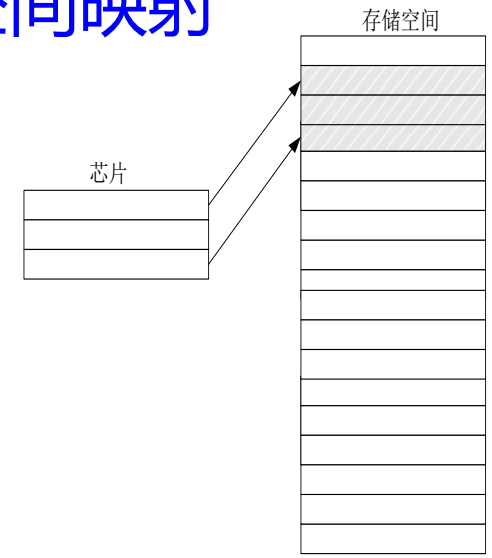
- (1) 存储容量扩展：由小容量存储芯片构建大容量的存储器

- (2) 存储空间映射：将物理存储空间映射到指定的逻辑存储空间
  - 计算机系 统微处理器能访问 的 存储空 间 通常称为 逻辑存储空 间
    - 如 32 位微处理器能访问 的逻辑存储空间 为 4GB,
  - 由 存储芯片 构成的 存储空 间 称为 物理存储空间
    - 如一片 1GB 存储芯片物理存储空间 仅为 1GB。

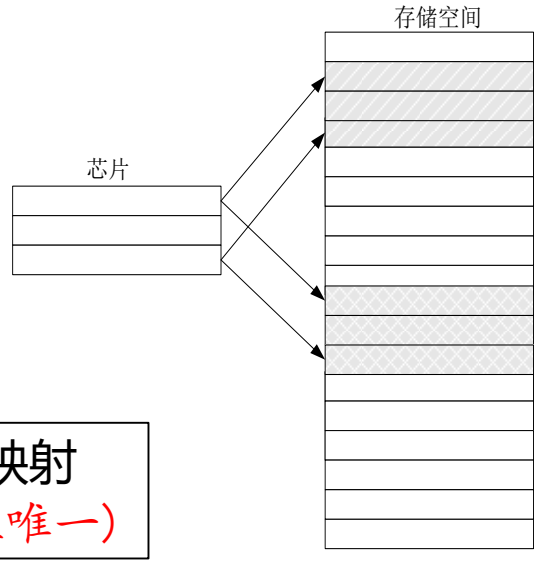
- (3) 不同类型数据访问兼容：低位宽存储芯片构建统一的支持多种不同类型数据访问的存储器

- (4) 总线操作时序匹配：总线与存储芯片的操作时序匹配
  - 由专门模块（CPU内部的各种存储器控制模块）完成

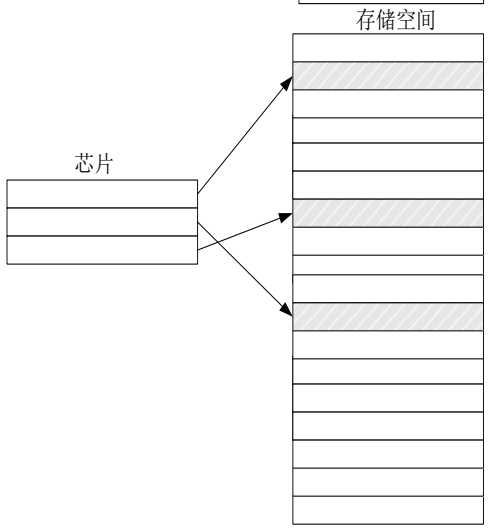
▶ 6.4.2 存储空间映射



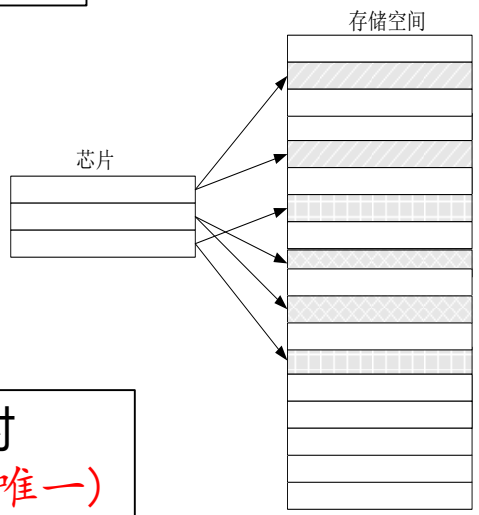
一对一整体映射
（地址连续且唯一）



一对多整体映射
（地址连续但不唯一）



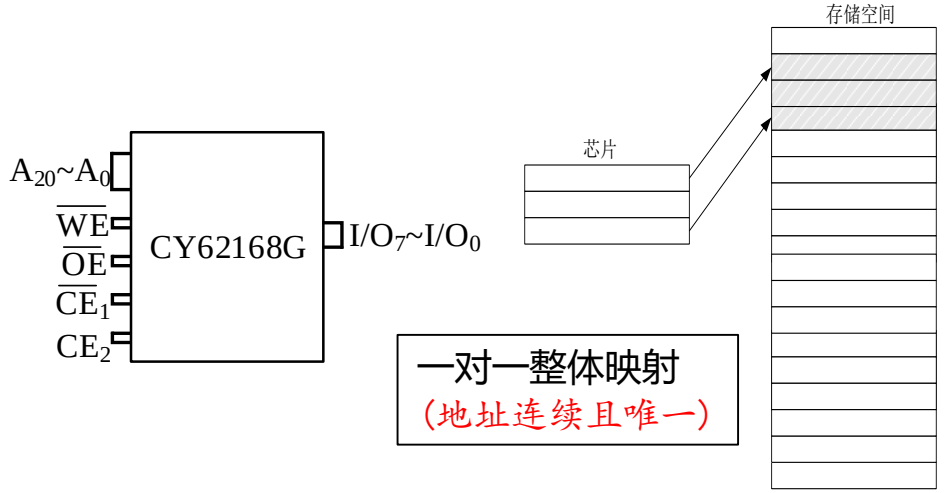
一对一离散映射
（地址不连续但唯一）



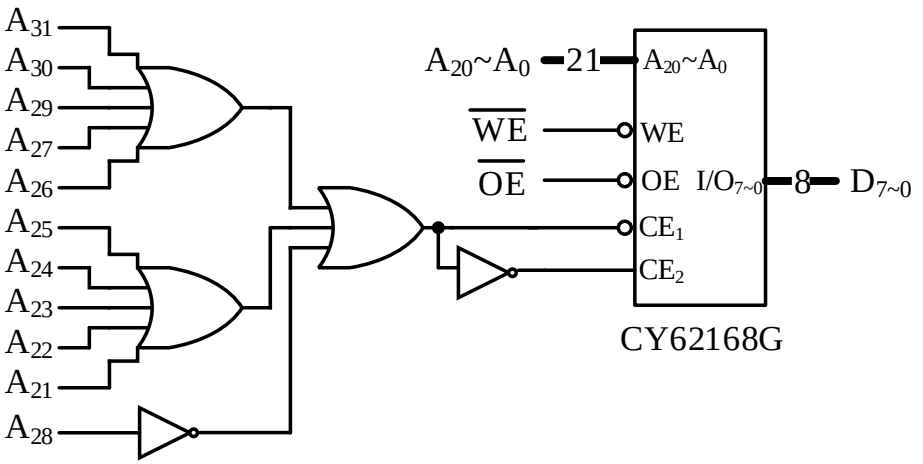
一对多离散映射
（地址不连续且不唯一）

▶ 6.4.2 存储空间映射

- 一对一整体映射举例
  - 例1 2M×8b的存储芯片CY62168G。将该芯片地址唯一映射到逻辑存储空间范围为0x00000000~0xffffffff的计算机系统的物理存储空间0x10000000~0x101fffff内，设计存储器接口电路。
    - 2MB的存储芯片具有21位地址总线：A20~A0。
    - 4GB存储空间的计算机系统具有32位地址总线：A31~A0。其中A20~A0与存储芯片的18位地址总线A20~A0直接相连。
    - 剩余的高9位地址线为A31~A21，由于地址范围为0x10000000~0x101fffff。由此可知在整个地址范围内A[31:21] =0001 0000 000 。因此需要将A31~A21译码产生存储芯片片选信号，假设片选信号为CS，且低电平有效，那么
    - $\overline{CS} = A31 + A30 + A29 + \overline{A28} + A27 + A26 + A25 + A24 + A23 + A22 + A21$
  - 系统地址总线所有剩余高位地址都必须经过译码之后才能连接到芯片的片选控制端叫做全译码法。

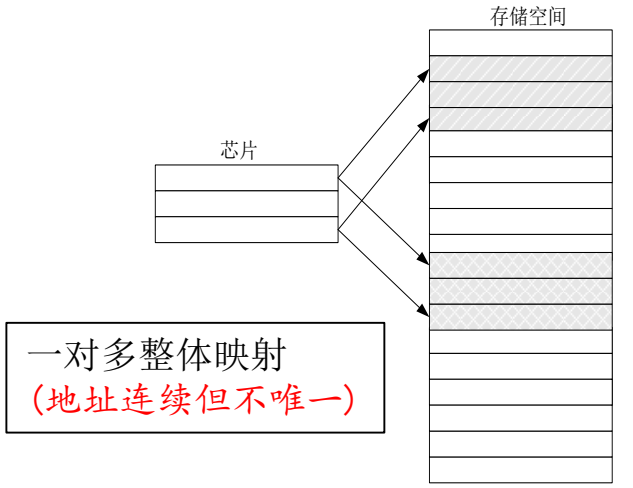


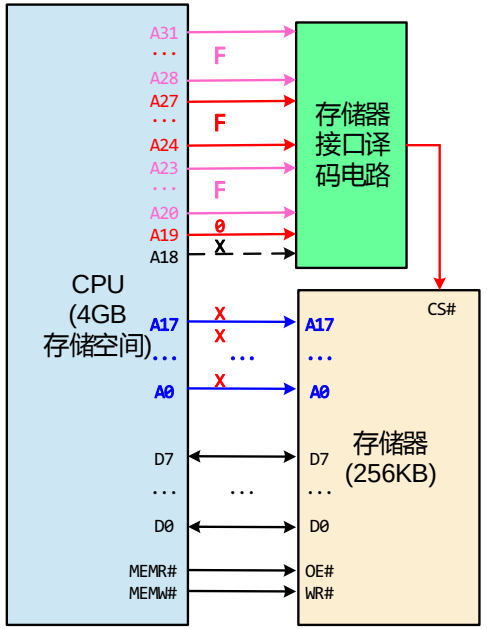
物理地址	A_{31} ~A_{21}	A_{20}~A_0
0x10000000	$(0001\ 0000\ 000)_2$	寻址芯片内部存储单元
0x101fffff	$(0001\ 0000\ 000)_2$	



▶ 6.4.2 存储空间映射

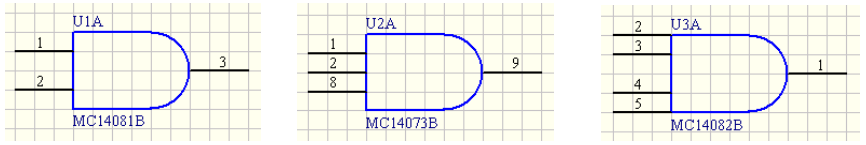
- 一对多整体映射举例
  - 部分高位地址线没有参与译码形成芯片片选控制信号。
    - 仅一位高位地址线参与译码，就叫做线选法；
    - 多位但不是全部高位地址线参与译码，就叫做部分译码法。
  - 例2 将容量为256KB的存储芯片一对多整体映射到存储空间为4GB的计算机存储系统中，且要求地址范围为0xfff00000~0xfff3ffff或0xfff40000~0xfff7ffff，该如何实现地址译码？
    - 由于地址范围可以为0xfff00000~0xfff3ffff或0xfff40000~0xfff7ffff，
    - 那么A[31:20] 固定为1、A19固定为0，而A18既可以是0也可以是1，因此可以不考虑A18这根地址线，
    - 仅用A31~A19译码产生存储芯片的片选信号，假设片选信号为CS，且低电平有效，那么
    - $\overline{CS} = \mathbf{A19} + \overline{A20 \cdot A21 \cdot A22 \cdot A23 \cdot A24 \cdot A25 \cdot A26 \cdot A27 \cdot A28 \cdot A29 \cdot A30 \cdot A31}$



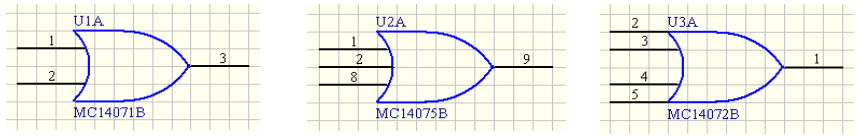


▶ 6.4.3 接口译码电路

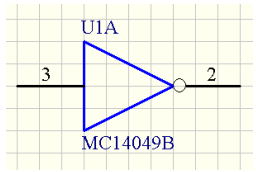
- 逻辑门电路
  - AND , OR , NOT, NAND ,NOR….
    - 与门：多位相与，输出——见0为0，全1为1；
    - 或门：多位相或，输出——见1为1，全0为0；
    - 非门：1变0，0变1；
    - 与非门：见0为1，全1为0；
    - 或非门：见1为0，全0为1；



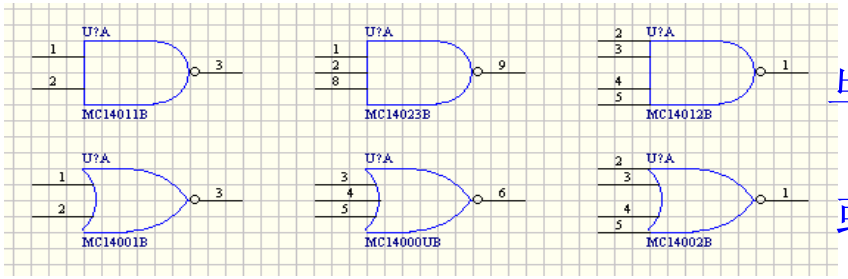
与门



或门

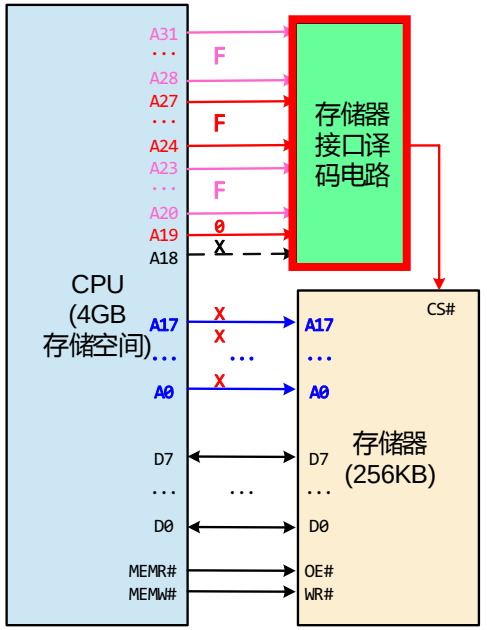


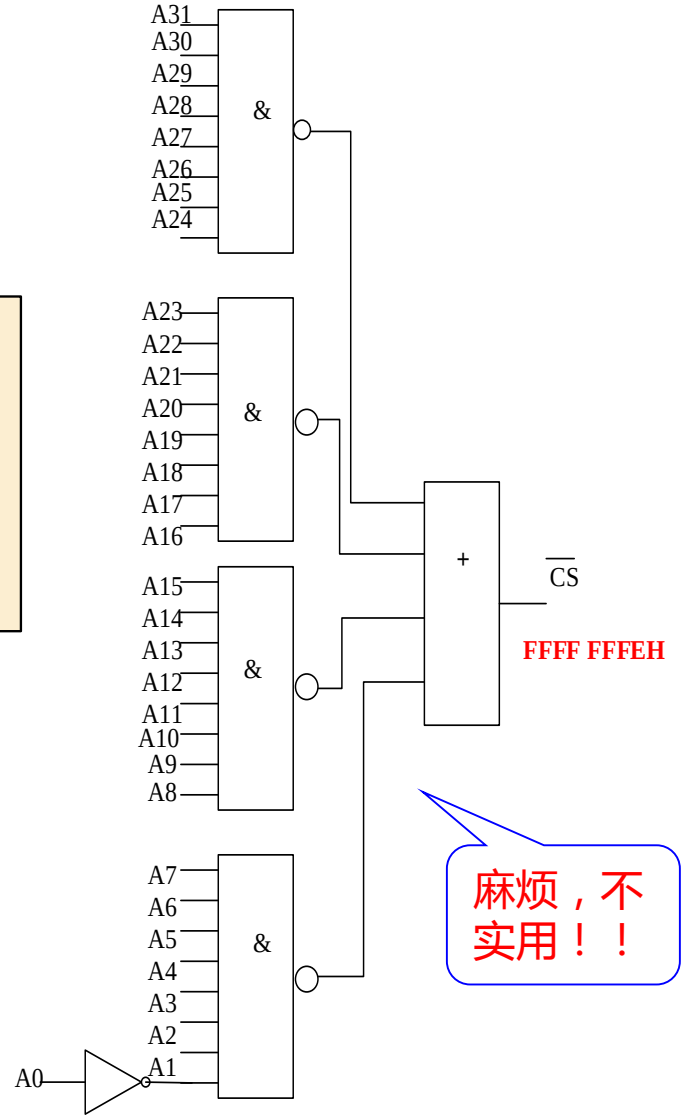
非门



与非门

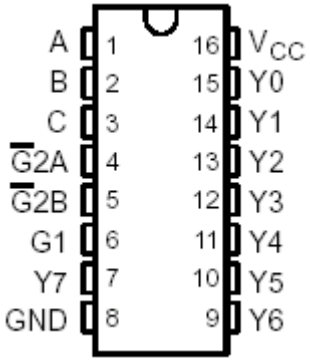
或非门





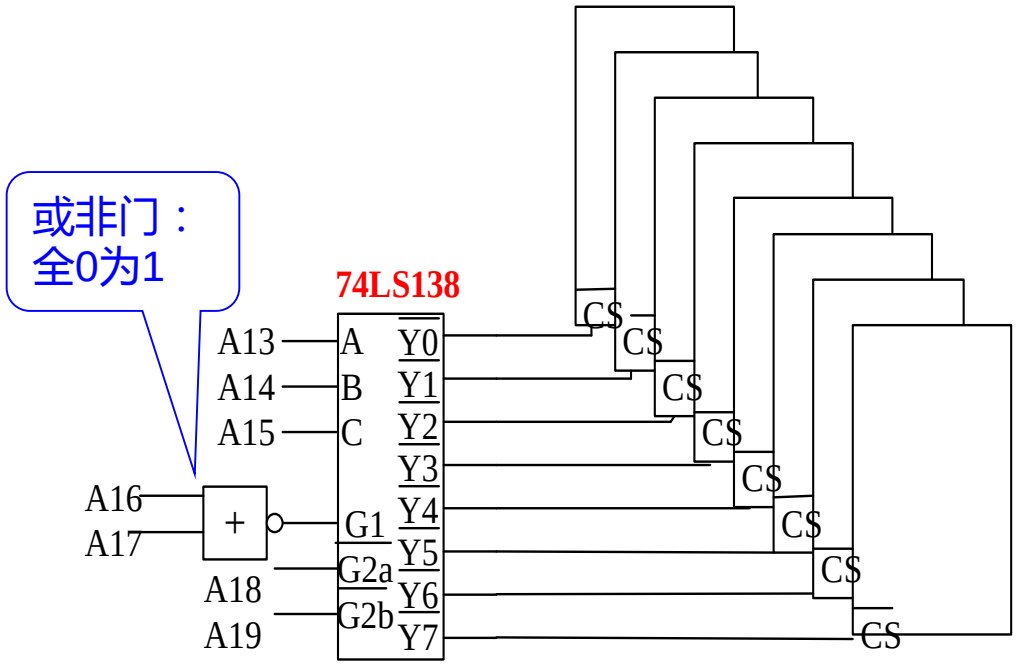
▶ 6.4.3 接口译码电路

- 专用译码芯片
  - 138，…
    - 20地址、8数据系统；
    - 用8KBX8构成64KB；
    - 地址0000h-FFFFh。



FUNCTION TABLE

ENABLE INPUTS			SELECT INPUTS			OUTPUTS							
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

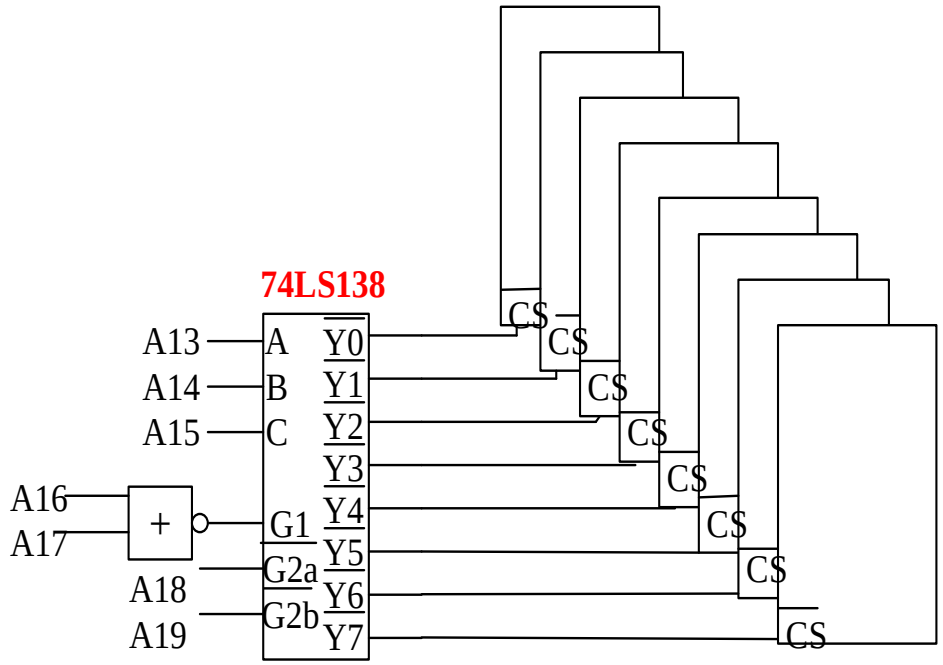


输出	对应的地址范围					地址16进制值
	A_{19-16}	A_{15-12}	A_{11-8}	A_{7-4}	A_{3-0}	
Y0	0 0 0 0 ,	0 0 0 X ,	X X X X ,	X X X X ,	X X X X	00000H ~ 01FFFH
Y1	0 0 0 0 ,	0 0 1 X ,	X X X X ,	X X X X ,	X X X X	02000H ~ 03FFFH
Y2	0 0 0 0 ,	0 1 0 X ,	X X X X ,	X X X X ,	X X X X	04000H ~ 05FFFH
Y3	0 0 0 0 ,	0 1 1 X ,	X X X X ,	X X X X ,	X X X X	06000H ~ 07FFFH
Y4	0 0 0 0 ,	1 0 0 X ,	X X X X ,	X X X X ,	X X X X	08000H ~ 09FFFH
Y5	0 0 0 0 ,	1 0 1 X ,	X X X X ,	X X X X ,	X X X X	0A000H ~ 0BFFFH
Y6	0 0 0 0 ,	1 1 0 X ,	X X X X ,	X X X X ,	X X X X	0C000H ~ 0DFFFH
Y7	0 0 0 0 ,	1 1 1 X ,	X X X X ,	X X X X ,	X X X X	0E000H ~ 0FFFFH

▶ 6.4.3 接口译码电路

- 可编程逻辑器件

```
module DECODER(
    input [19:13]       A,          //输入地址信号
    output [7:0]        CS          //输出片选信号
    );
reg [7:0] CS1;                      //设置输出寄存器
assign CS[7:0]=CS1[7:0];            //输出引脚与寄存器相连
always @(A) begin
        if (A[19:16] != 4'b0000)
                CS1[7:0] <= 8'b11111111;     //A[19:16]!=0000，所有CS无效
        else
        case (A[15:13])
                3'b000  : CS1[7:0] <= 8'b11111110;
                3'b001  : CS1[7:0] <= 8'b11111101;
                3'b010  : CS1[7:0] <= 8'b11111011;
                3'b011  : CS1[7:0] <= 8'b11110111;
                3'b100  : CS1[7:0] <= 8'b11101111;
                3'b101  : CS1[7:0] <= 8'b11011111;
                3'b110  : CS1[7:0] <= 8'b10111111;
                3'b111  : CS1[7:0] <= 8'b01111111;
        endcase
end
endmodule
```

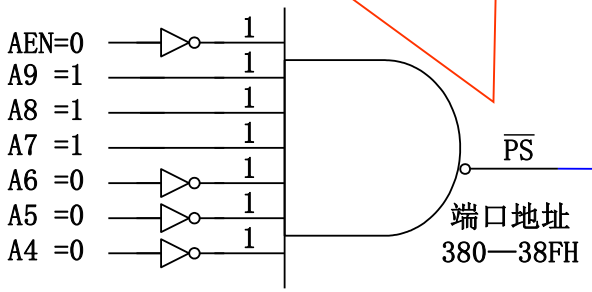


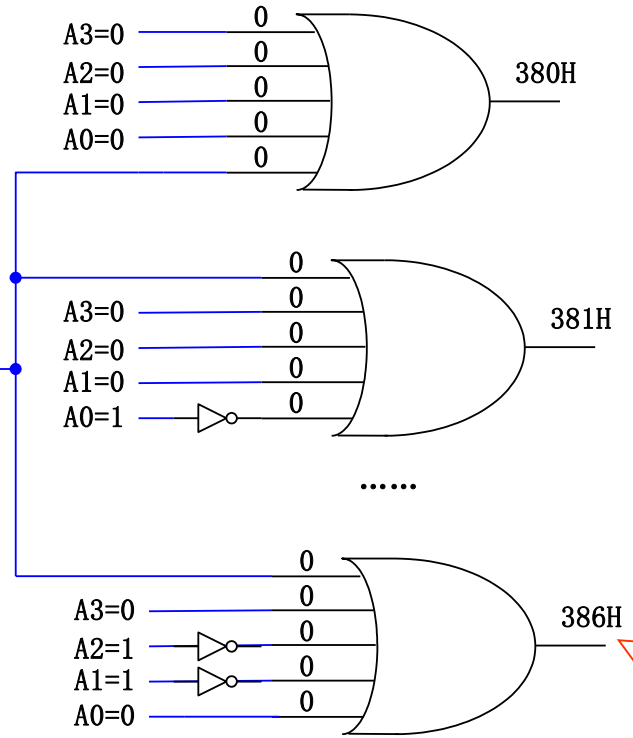
灵活：无需改变硬件电路

▶ 6.4.3 接口译码电路

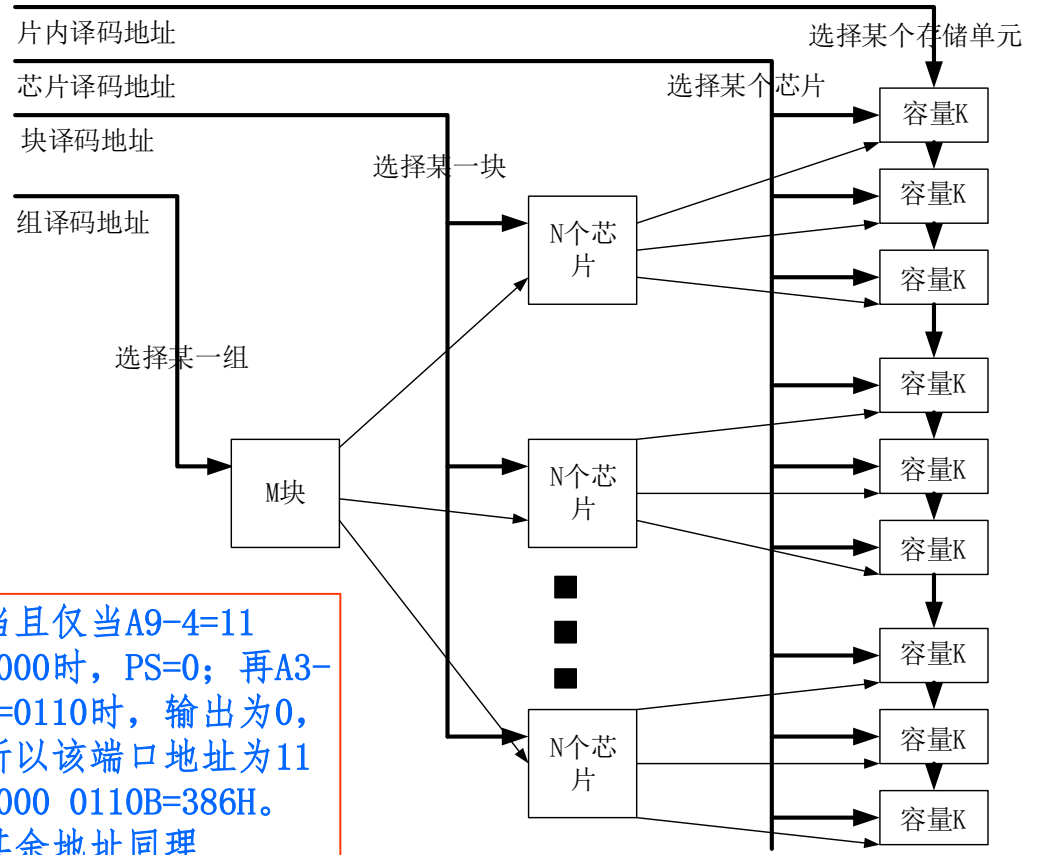
- 分级译码
  - 在实际应用中，通常先用高位地址线产生一组地址，然后再用剩下的低位地址线进一步组内寻址

组地址为380H-38FH，十位地址中的低四位可变，说明仅有高六位地址线参与了组地址译码，再在组内译码，则需要低4位地址也参与译码。



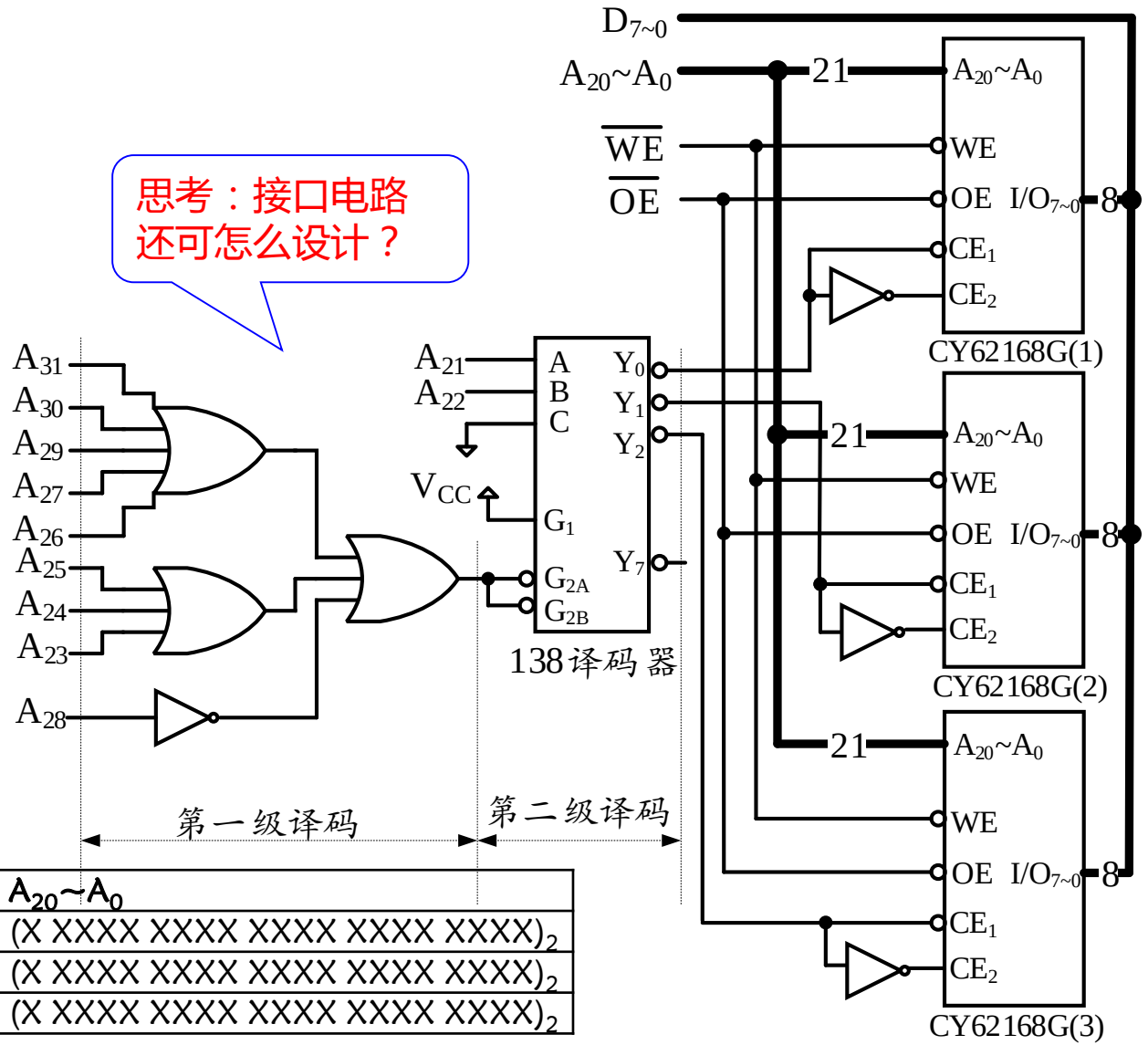


当且仅当A9-4=111000时，PS=0；再A3-0=0110时，输出为0，所以该端口地址为111000 0110B=386H。其余地址同理



▶ 6.4.4 存储空间映射示例

- 例6.6 一片SRAM存储芯片CY62168G的容量为2MB，用它设计一容量为6M×8b的存储器，且该存储器唯一映射到逻辑存储空间范围为0x00000000~0xffffffff的计算机系统的物理存储空间0x10000000~0x105fffff，试设计该存储器接口电路。
  - 共需要：6MB / 2MB = 3 片，各自地址范围：



物理地址范围	A_{31} ~A_{21}	A_{20}~A_0
0x10000000~0x101fffff	(0001 0000 000)2	$(X\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX)_2$
0x10200000~0x103fffff	(0001 0000 001)2	$(X\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX)_2$
0x10400000~0x105fffff	(0001 0000 010)2	$(X\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX)_2$

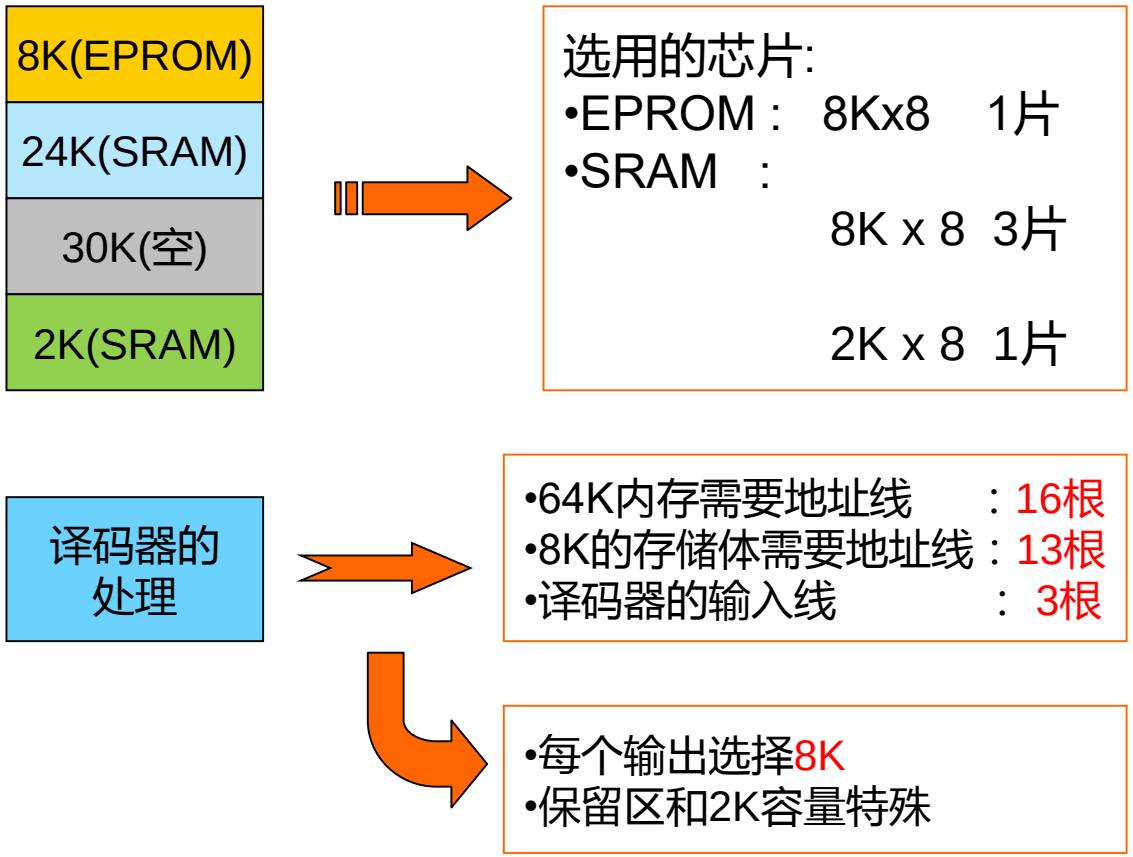
▶ 6.4.4 存储空间映射示例

- 【例】CPU的地址总线共有A15 – A0，双向数据总线8根(D7 – D0)，控制信号有R/W#和访存请求MREQ#。主存地址空间分配如下：
  - 0 – 8191：为系统程序区，由ROM组成
  - 8192 - 32767：为用户数据区，
  - 最后2K：为系统数据区。

上述地址为十进制，按字节编址。现有如下存储器供选用：

- EPROM ：8K x 8位(控制端仅有CS)
- SRAM ：16K x 1、2K x 8、 4K x 8 、 8K x 8

试设计该存储器，并画出与CPU连接图



▶ 6.4.4 存储空间映射示例

- 【例】CPU的地址总线共有A15 – A0，双向数据总线8根(D7 – D0)，控制信号有R/W#和访存请求MREQ#。主存地址空间分配如下：
  - 0 – 8191：为系统程序区，由ROM组成
  - 8192 - 32767：为用户数据区，
  - 最后2K：为系统数据区。

上述地址为十进制，按字节编址。现有如下存储器供选用：

- EPROM ：8K x 8位(控制端仅有CS)
- SRAM ：16K x 1、2K x 8、 4K x 8 、 8K x 8

试设计该存储器，并画出与CPU连接图

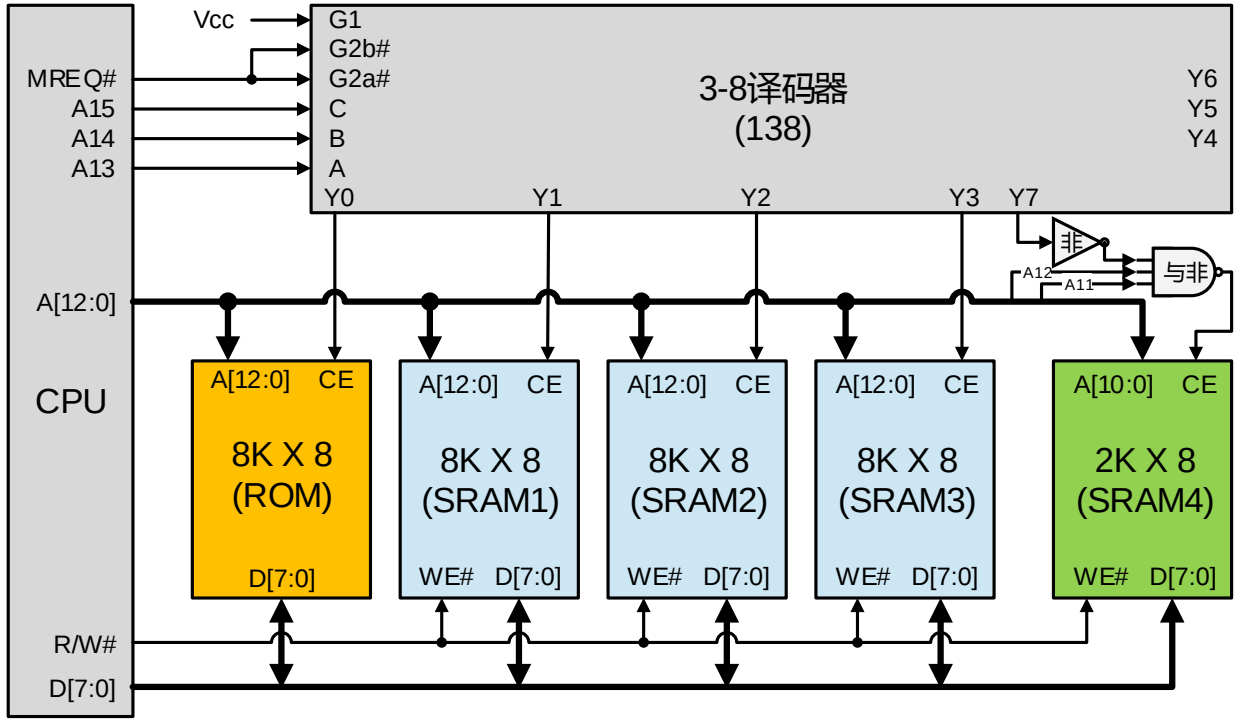
片选	地址范围				地址十六进制数	对应存储器
	A_{15}	A_{14}	A_{13}	A_{12}, A_{11-8}, A_{7-4}, A_{3-0}		
Y0	0	0	0	X, XXX, XXX, XXX	0000H ~ 1FFFH	ROM
Y1	0	0	1	X, XXX, XXX, XXX	2000H ~ 3FFFH	SRAM1
Y2	0	1	0	X, XXX, XXX, XXX	4000H ~ 5FFFH	SRAM2
Y3	0	1	1	X, XXX, XXX, XXX	6000H ~ 7FFFH	SRAM3
Y4	1	0	0	X, XXX, XXX, XXX	8000H ~ 9FFFH	
Y5	1	0	1	X, XXX, XXX, XXX	A000H ~ BFFFH	
Y6	1	1	0	X, XXX, XXX, XXX	C000H ~ DFFFH	
Y7	1	1	1	0, 0XX, XXX, XXX	E000H ~ E7FFH	
				0, 1XX, XXX, XXX	E800H ~ EFFFH	
				1, 0XX, XXX, XXX	F000H ~ F7FFH	
				1, 1XX, XXX, XXX	F800H ~ FFFFH	SRAM4

▶ 6.4.4 存储空间映射示例

- 【例】CPU的地址总线共有A15 – A0，双向数据总线8根(D7 – D0)，控制信号有R/W#和访存请求MREQ#。主存地址空间分配如下

片选	地址范围				地址十六进制数	对应存储器
	A_{15}	A_{14}	A_{13}	A_{12}, A_{11-8}, A_{7-4}, A_{3-0}		
Y0	0	0	0	X, XXXX, XXXX, XXXX	0000H ~ 1FFFH	ROM
Y1	0	0	1	X, XXXX, XXXX, XXXX	2000H ~ 3FFFH	SRAM1
Y2	0	1	0	X, XXXX, XXXX, XXXX	4000H ~ 5FFFH	SRAM2
Y3	0	1	1	X, XXXX, XXXX, XXXX	6000H ~ 7FFFH	SRAM3
Y4	1	0	0	X, XXXX, XXXX, XXXX	8000H ~ 9FFFH	
Y5	1	0	1	X, XXXX, XXXX, XXXX	A000H ~ BFFFH	
Y6	1	1	0	X, XXXX, XXXX, XXXX	C000H ~ DFFFH	
Y7	1	1	1	0, 0XXX, XXXX, XXXX	E000H ~ E7FFH	
				0, 1XXX, XXXX, XXXX	E800H ~ EFFFH	
				1, 0XXX, XXXX, XXXX	F000H ~ F7FFH	
				1, 1XXX, XXXX, XXXX	F800H ~ FFFFH	SRAM4

试设计该存储器，并画出与CPU连接图



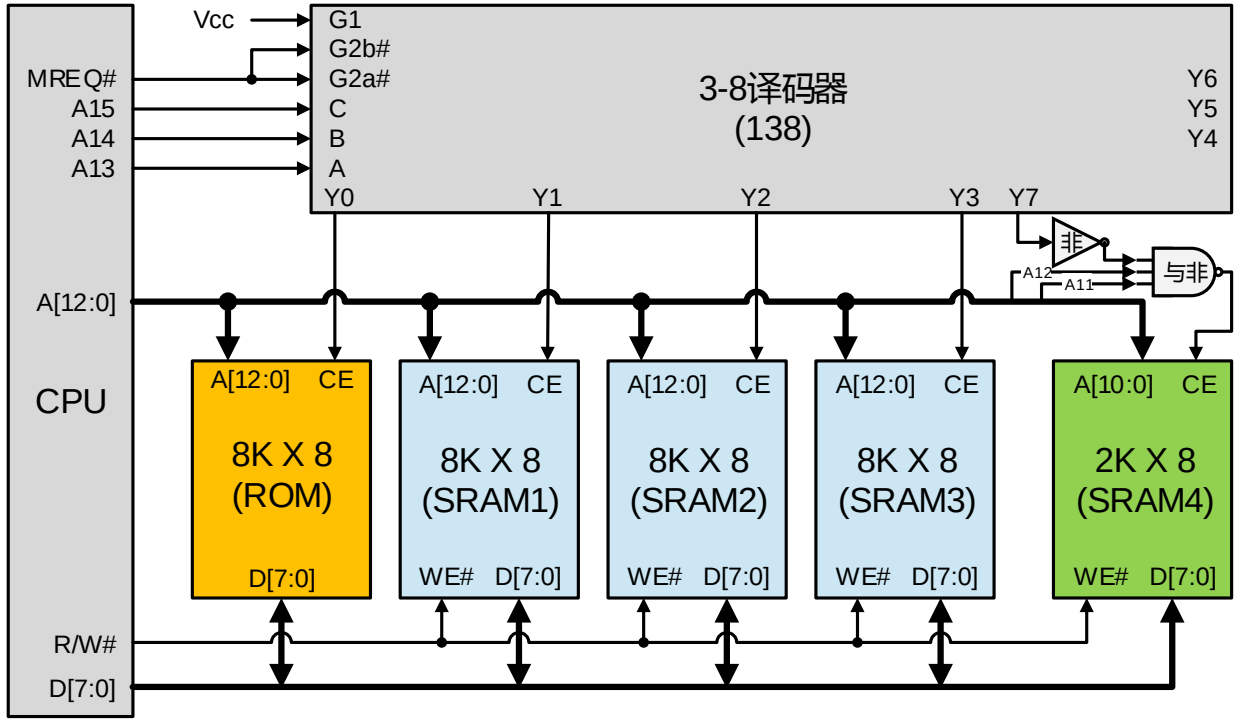
▶ 6.4.4 存储空间映射示例

- 【例】CPU的地址总线共有A15 – A0，双向数据总线8根(D7 – D0)，控制信号有R/W#和访存请求MREQ#。主存地址空间分配如下：
  - 0 – 8191：为系统程序区，由ROM组成
  - 8192 - 32767：为用户数据区，
  - 最后2K：为系统数据区。

上述地址为十进制，按字节编址。现有如下存储器供选用：

- EPROM ：8K x 8位(控制端仅有CS)
- SRAM ：16K x 1、2K x 8、4K x 8、8K x 8

试设计该存储器，并画出与CPU连接图



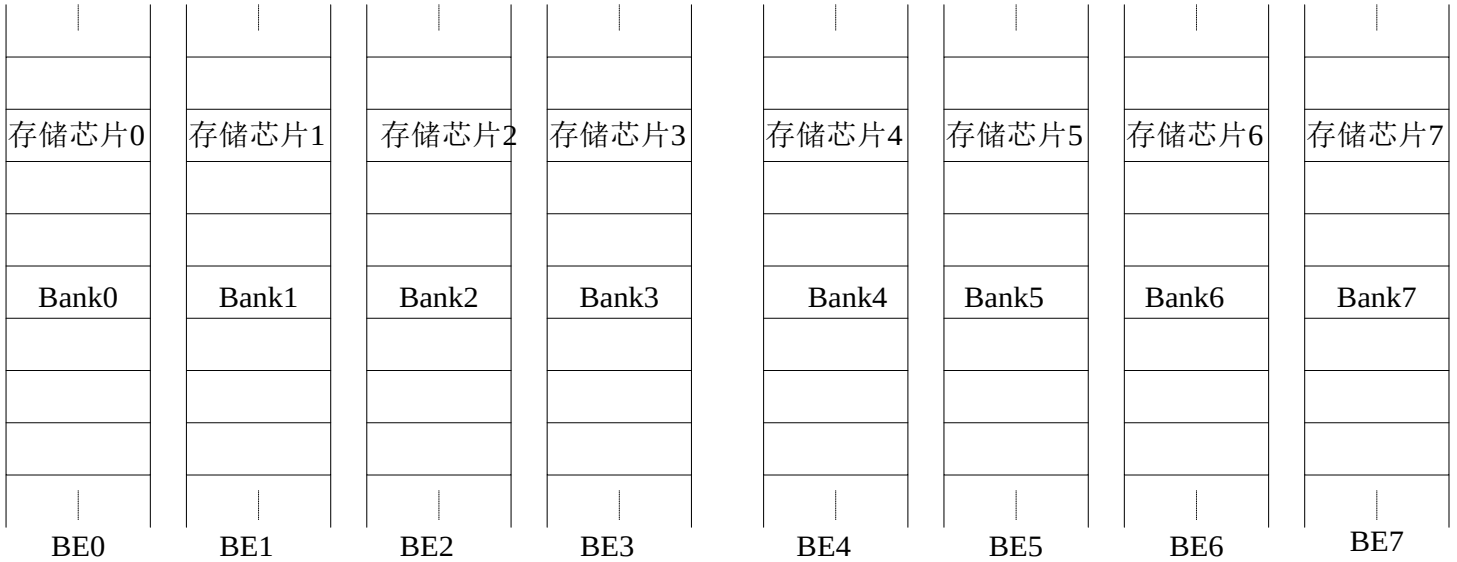
▶ 存储器接口设计需解决的问题

- (1) 存储容量扩展：由小容量存储芯片构建大容量的存储器
- (2) 存储空间映射：将物理存储空间映射到指定的逻辑存储空间
  - 计算机系 统微处理器能访问 的 存储空 间 通常称为 逻辑存储空 间
    - 如 32 位微处理器能访问 的逻辑存储空间 为 4GB,
  - 由 存储芯片 构成的 存储空 间 称为 物理存储空间
    - 如一片 1GB 存储芯片物理存储空间 仅为 1GB。
- (3) 不同类型数据访问兼容：低位宽存储芯片构建统一的支持多种不同类型数据访问的存储器
- (4) 总线操作时序匹配：总线与存储芯片的操作时序匹配
  - 由专门模块（CPU内部的各种存储器控制模块）完成

6.4 存储器接口设计

▶ 6.4.5 存储器组织结构

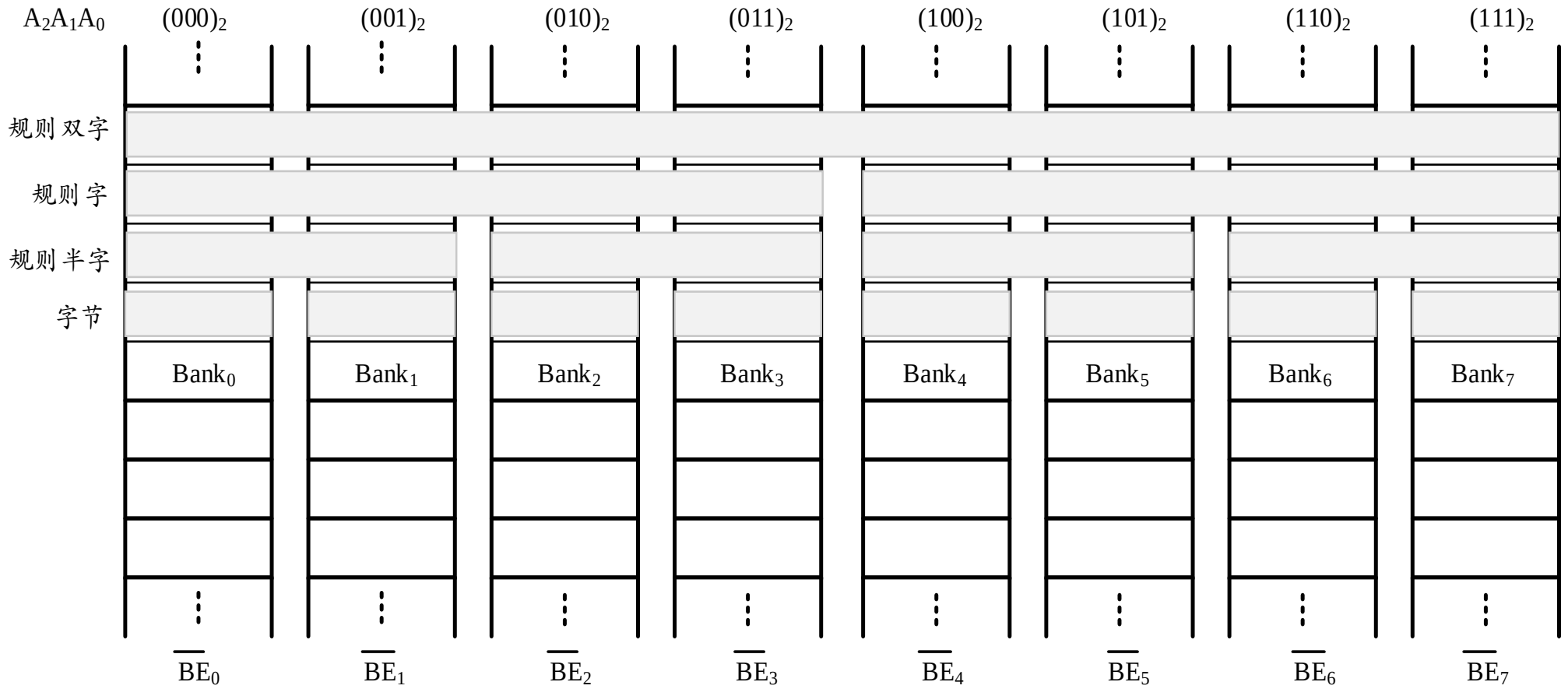
- 大部分微处理器支持访问8位，16位，32位到64位不等位宽数据
  - 用字节使能信号$\overline{BE}$(Byte Enable)来选择某一字节，构成对8/16/32/64位不同类型数据的访问



- 每个存储芯片数据宽度为8bit（1Byte）；
- 若想访问字节类型数据，给出BE0——BE7中任意一个字节使能信号有效，就可以访问对应存储单元的字节数据；
- 若想访问半字类型数据，给出BE0——BE7中连续两个字节使能信号有效，就可以访问对应的两个连续存储单元的半字数据；（实际还需半字地址对齐）
- 若想访问字类型数据，给出BE0——BE7中连续四个字节使能信号有效，就可以访问对应的四个连续存储单元的字数据；（实际还需字地址对齐）

▶ 6.4.5 存储器组织结构

- 用字节使能信号$\overline{BE}$选择某一字节，构成对8/16/32/64位不同类型数据的访问
  - $\overline{BE}$和存储域、地位地址的关系

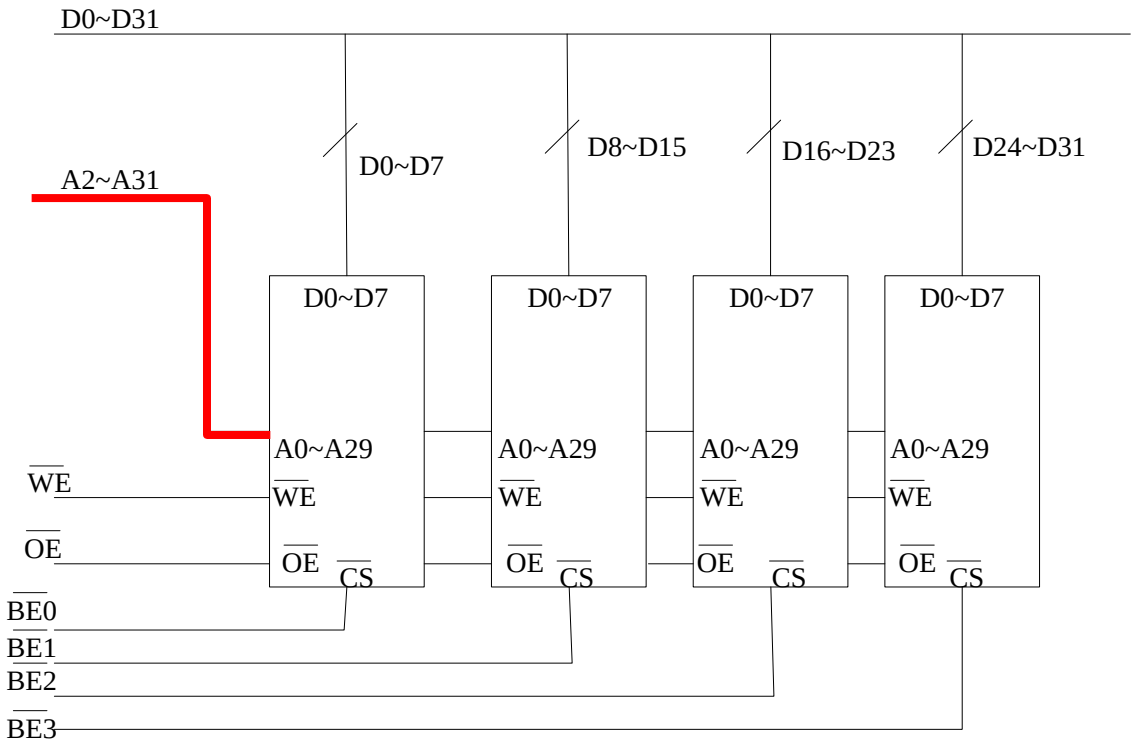


▶ 6.4.5 存储器组织结构

- 使用字节使能信号与高位地址译码产生存储芯片片选信号。
- 使用字节使能信号与存储器写控制信号译码产生存储芯片写控制信号。

● 例 为一个32位的微处理器设计一个存储空间为4GB的SRAM存储器，要求支持字节、半字、字类型数据访问，采用1G*8bit的SRAM存储芯片，如何设计该存储器？

- ➢ 用4片8位芯片构成32位数据宽度；
- ➢ 32位的字访问需地址对齐，地址信息A1~A0无效，存储芯片的A29~A0对应连接到CPU的A31~A2上
- ➢ 采用BE0~BE3分别控制4片芯片的片选线



未选中芯片不工作，处于低功耗状态

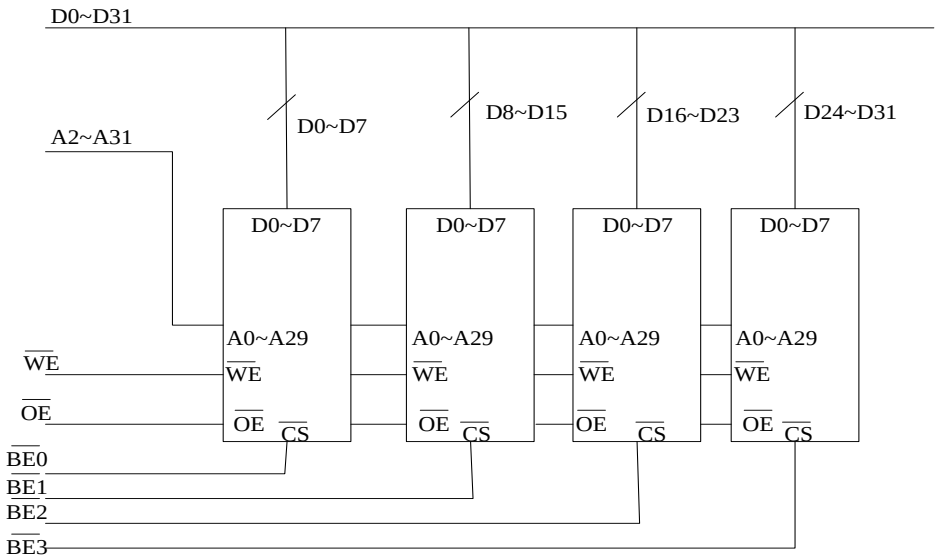
6.4 存储器接口设计

▶ 6.4.5 存储器组织结构

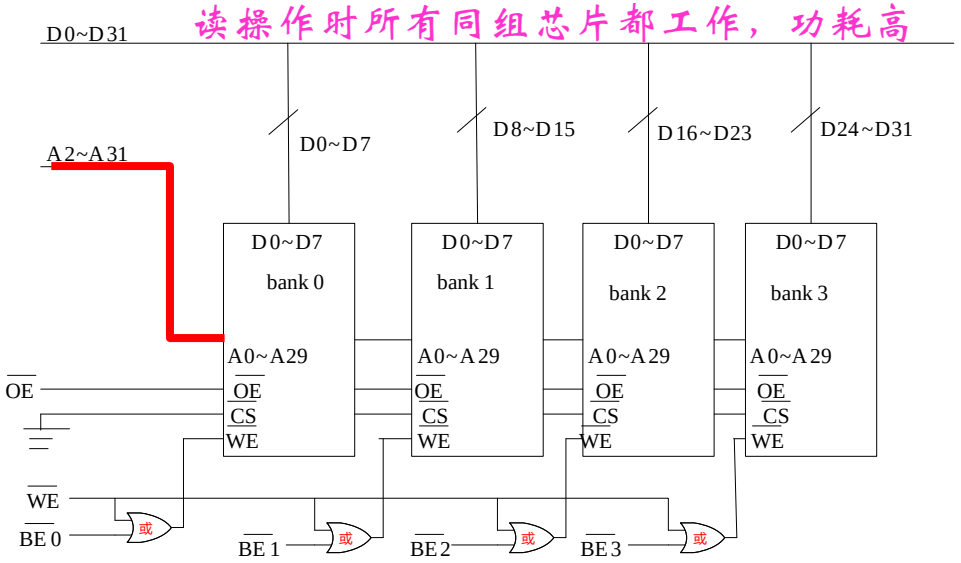
- 使用字节使能信号与高位地址译码产生存储芯片片选信号。
- 使用字节使能信号与存储器写控制信号译码产生存储芯片写控制信号。

● 例 为一个32位的微处理器设计一个存储空间为4GB的SRAM存储器，要求支持字节、半字、字类型数据访问，采用1G*8bit的SRAM存储芯片，如何设计该存储器？

- ➢ 32位的字访问需地址对齐，地址信息A1~A0无效，存储芯片的A29~A0对应连接到CPU的A31~A2上
- ➢ 采用BE0~BE3分别与存储器写控制信号译码产生存储芯片写控制信号



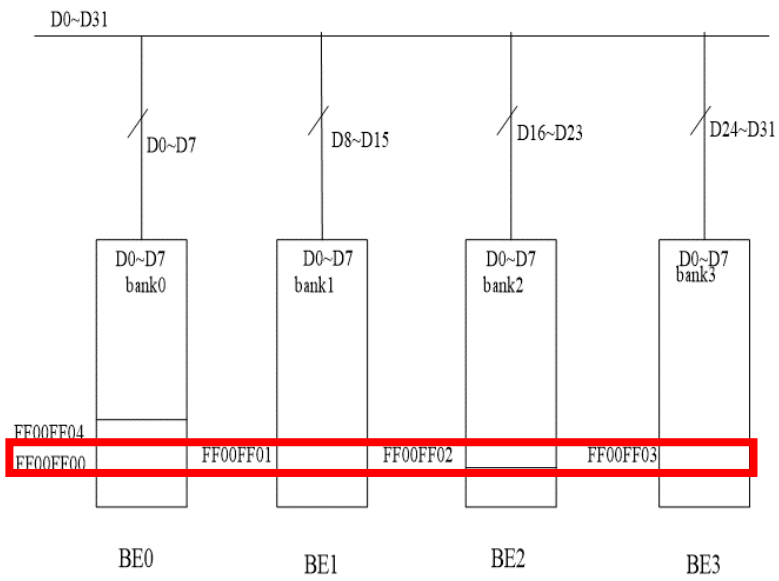
▪ 采用BE0~BE3分别控制4片芯片的片选线



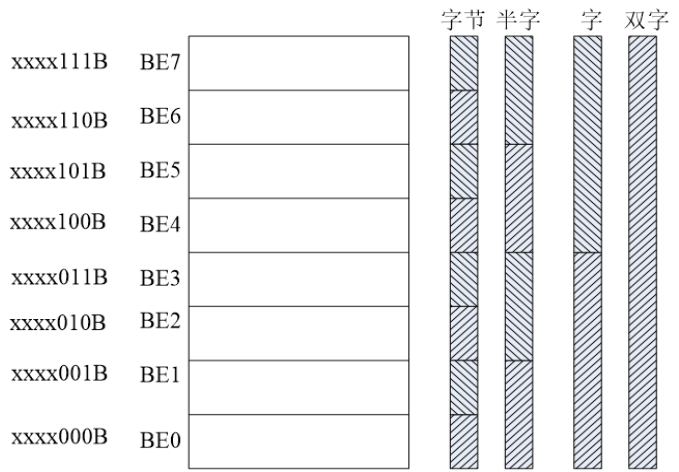
▪ BE信号与存储器写信号控制存储芯片写信号实现不同类型的数据访问

▶ 6.4.5 存储器组织结构

- 内存访问边界对齐
  - 一次总线操作实现半字类型的数据访问要求字节使能信号从偶字节使能信号开始的连续两个字节有效，
    - 即半字数据访问，要求最低地址位为0
  - 一次总线操作实现字类型的数据访问要求字节使能信号从4字节使能信号开始的连续4个字节有效，
    - 即字数据访问，要求低2位地址位为00
  - 一次总线操作实现双字类型的数据访问要求字节使能信号从8字节使能信号开始的连续8个字节有效
    - 即双字数据访问，要求低3位地址位为000

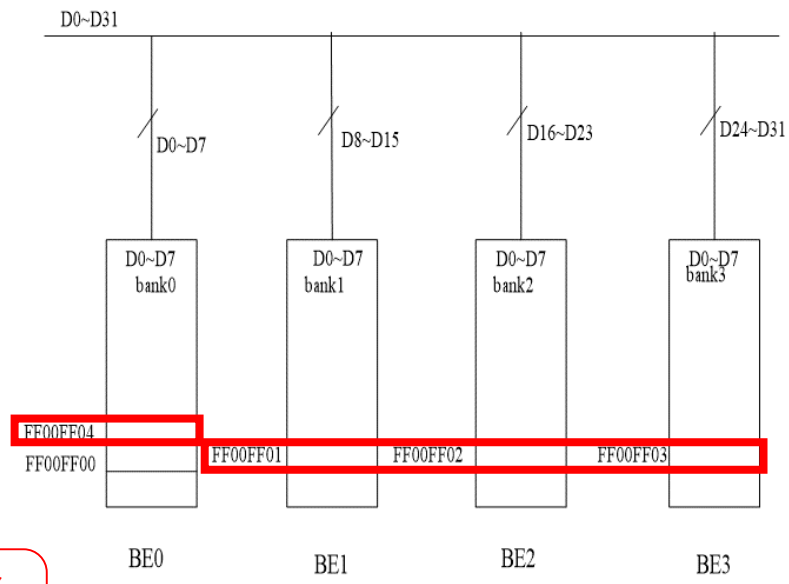


- 如从FF00FF00地址读取一个字类型的数据。
  - FF00FF00的低2位为0，已经字边界对齐
  - 字类型的数据D0—D31直接从bank0—bank3读取，bank0提供数据的D0—D7， bank1提供数据的D8—D15， bank2提供数据的D16—D23，bank4提供数据的D24—D31，通过微处理器的一次读总线操作就可以读入到微处理器中。

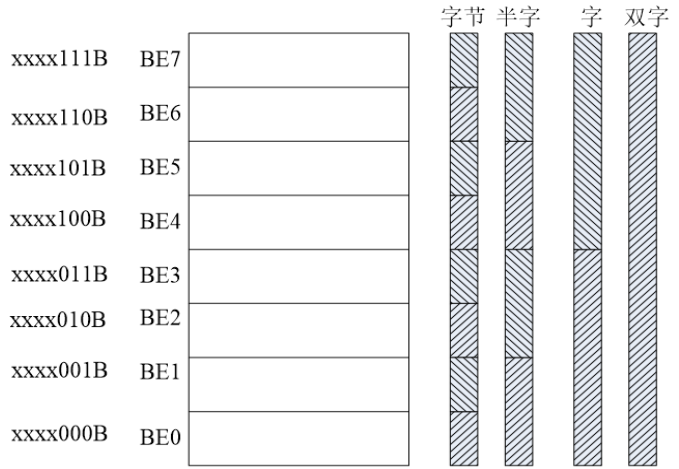


▶ 6.4.5 存储器组织结构

- 内存访问边界对齐
  - 一次总线操作实现半字类型的数据访问要求字节使能信号从偶字节使能信号开始的连续两个字节有效，
    - 即半字数据访问，要求最低地址位为0
  - 一次总线操作实现字类型的数据访问要求字节使能信号从4字节使能信号开始的连续4个字节有效，
    - 即字数据访问，要求低2位地址位为00
  - 一次总线操作实现双字类型的数据访问要求字节使能信号从8字节使能信号开始的连续8个字节有效
    - 即双字数据访问，要求低3位地址位为000



对一个内存未对齐的数据进行了这么多额外的操作，大大降低了CPU性能；所以需要采用边界对齐的访问方式



- 如从FF00FF01地址读取一个字类型的数据。
  - FF00FF01的低2位为01，没有字边界对齐
  - 若此时使BE0-BE3=0，则CPU读取的D31-D0 = $[D7\text{-}0]_{bk3}$ $[D7\text{-}0]_{bk2}$ $[D7\text{-}0]_{bk1}$ $[D7\text{-}0]_{bk0}$，而非期望D31-D0 = $[D7\text{-}0]_{bk0}$ $[D7\text{-}0]_{bk3}$ $[D7\text{-}0]_{bk2}$ $[D7\text{-}0]_{bk1}$。所以处理器就不能通过一次读总线操作正确获取32位字数据。
  - 实际需要分两次读取：首先读取FF00FF00地址，使BE1-BE3有效，读取D31-D8，并右移到D23-0；然后读取FF00FF04地址，使BE0有效，读取D7-D0，并左移到D31-24；最后再把两次结果合并成32位字数据。

► 6.4.5 存储器组织结构

- 内存访问边界对齐
  - 内存对齐的主要作用是：
    - 1、平台原因(移植原因)：不是所有的硬件平台都能访问任意地址上的任意数据的；某些硬件平台只能在某些地址处取某些特定类型的数据，否则抛出硬件异常。
    - 2、性能原因：经过内存对齐后，CPU的内存访问速度大大提升。

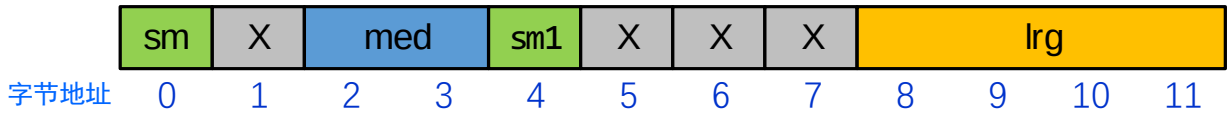
▶ 6.4.5 存储器组织结构

- 内存访问边界对齐
  - 例4. 2 已知某32位计算机系统，编译器采用边界对其方式为数据分配内存空间，若定义了以下数据结构：

```
strict foo {
    char    sm;       /*1字节*/
    short   med;      /*2 字节*/
    char    sm1;      /*1字节*/
    int     lrg;       /*4字节*/
}
```

```
struct foo1 {
    char    sm;       /*1字节*/
    char    sm1;      /*1字节*/
    short   med;      /*2 字节*/
    int     lrg;      /*4字节*/
}
```

  - foo边界对齐的内存映像：浪费内存空间？？

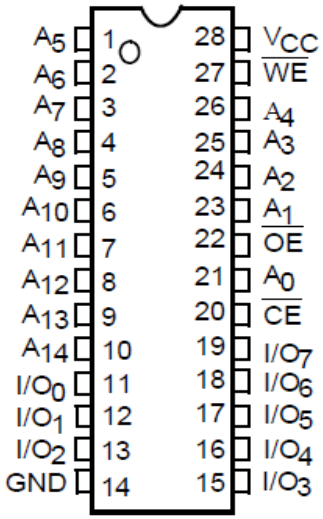


  - foo非边界对其的内存映像：增大访问时延？？

如要节省内存存储空间，提高访问效率，可以修改该数据结构的定义方式为foo1，其内存映像：

▶ 6.4.6 多类型数据访问存储器接口

- 【例】SRAM 64M × 8 b构造 128M × 32b，支持字节、字、半字类型数据访问。
  - 字长扩展：4片
  - 容量扩展：2组
  - 字节使能信号参与译码控制



$$\text{SRAM0: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot\overline{A28}} + \overline{BE0}$$

$$\text{SRAM1: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot\overline{A28}} + \overline{BE1}$$

$$\text{SRAM2: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot\overline{A28}} + \overline{BE2}$$

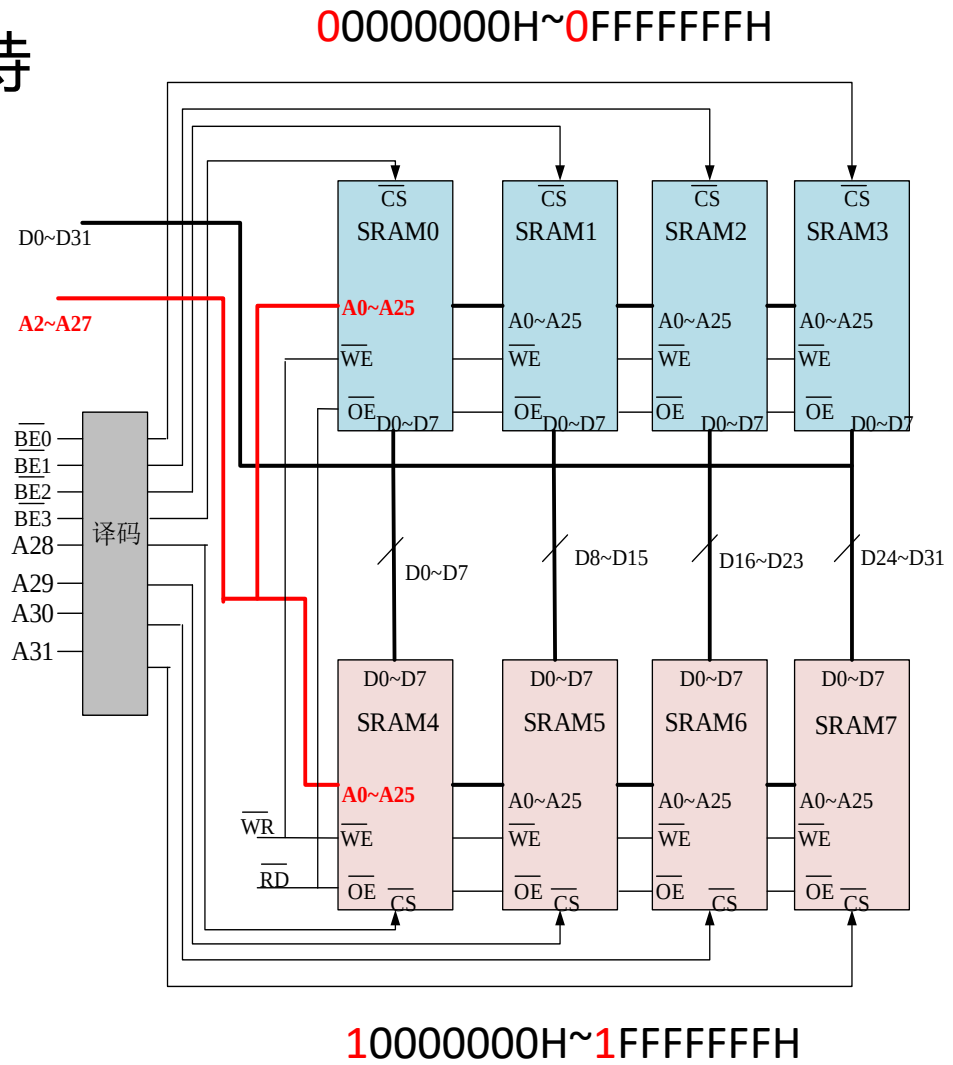
$$\text{SRAM3: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot\overline{A28}} + \overline{BE3}$$

$$\text{SRAM4: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot A28} + \overline{BE0}$$

$$\text{SRAM5: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot A28} + \overline{BE1}$$

$$\text{SRAM6: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot A28} + \overline{BE2}$$

$$\text{SRAM7: } \overline{CS} = \overline{\overline{A31}\cdot\overline{A30}\cdot\overline{A29}\cdot A28} + \overline{BE3}$$



第6章 作业

- 作业题（第二版，第六章P244）
  - 22
  - 26

- 要求
  - 微助教作业提交
  - 下次课前提交

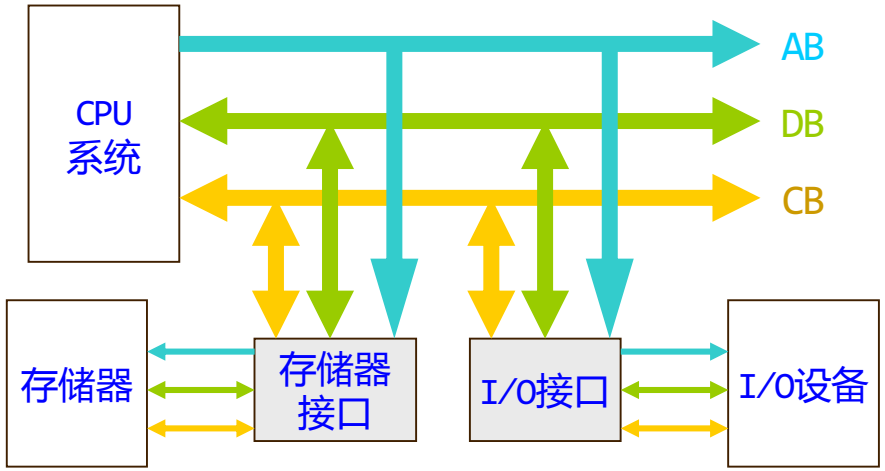
► 两类存储器接口：

- 简单存储器接口
  - SRAM与CPU的连接及扩展
    - 1)需要连接的线
      - ►地址线、数据线、读 / 写控制线、片选信号
    - 2)存储器扩展的种类
      - ►①位扩展 ：当数据位不足时
      - ►②字扩展 ：当容量不足时
      - ►③字位同时扩展：当数据位和存储体的容量均不足时使用

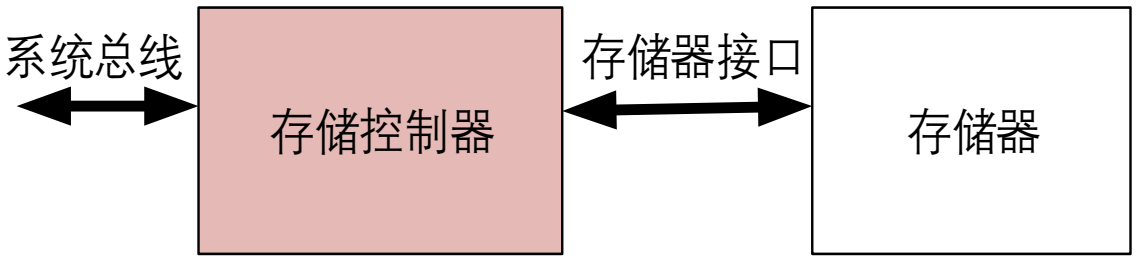
- 基于存储控制器的接口
  - 实现存储芯片与总线的接口；
  - 通过对存储芯片的多次读写操作以及数据缓冲实现不同字节的访问

▶ 基于存储控制器的接口

- 存储控制器

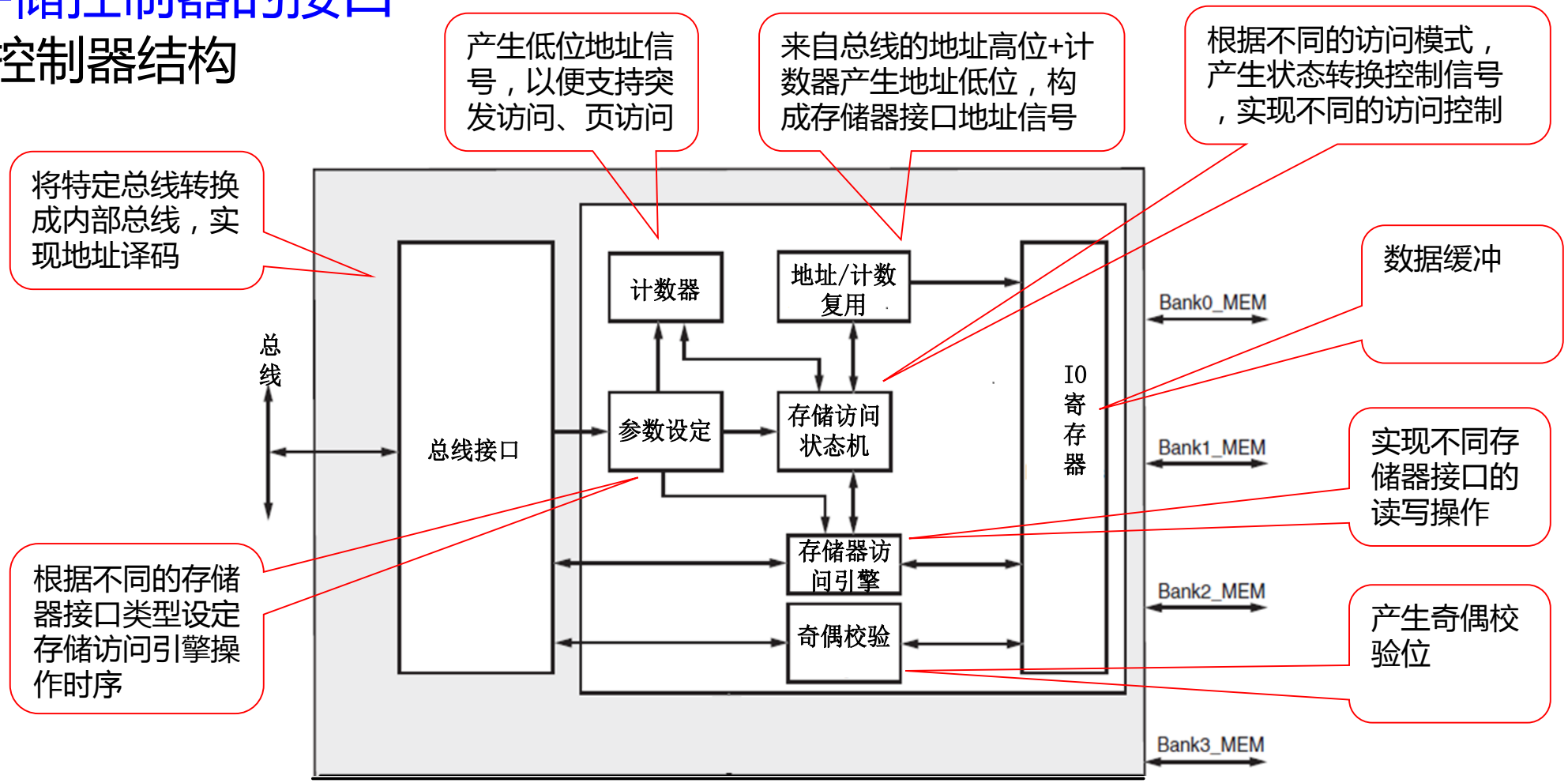


- 将系统总线转换为适合访问各类存储器总线信号的接口设备



基于存储控制器的接口

- 存储控制器结构



▶ 基于存储控制器的接口

- Xilinx AXI存储控制器：SRAM接口信号

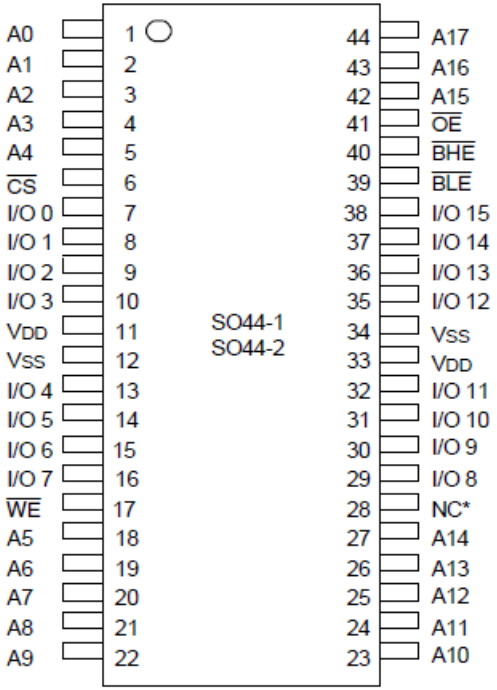
信号类型描述	存储控制器引脚名称	存储芯片引脚名称
数据线	MEM_DQ(((DN+1)*DW)-1:DN*DW)	D(DW-1 : 0)
地址线	MEM_A(HAW-AS-1:HAW-MAW-AS)	A(MAW-1 : 0)
芯片使能线（低电平有效）	MEM_CEN(BN)	CEN
读使能线（低电平有效）	MEM_OEN	OEN
写使能线（低电平有效）	MEM_WEN	WEN (有字节使能的芯片)
写使能线（低电平有效）	MEM_QWEN(DN*DW/8)	WEN (无字节使能的芯片)
字节使能线（低电平有效）	MEM_BEN((((DN+1)*DW/8)-1):(DN*DW/8))	BEN(DW/8-1 : 0)

变量名称	具体含义	变量名称	具体含义
DN	存储块内芯片序号(Memory bank number)	HAW	总线地址宽度(Width in bits of AXI address bus)
BN	子存储系统存储块序号(Memory device number within a bank)	MW	存储块数据位宽(Width in bits of memory subsystem)
DW	存储芯片数据总线位宽(Width in bits of data bus for memory device)	AS	地址偏移宽度=$\log_2(\frac{AU*MW}{DW}/8)$, Address shift for address bus
AU	存储芯片可寻址最小数据位宽(Width in bits of smallest addressable data word on the memory device)	MAW	存储芯片地址总线宽度(Width in bits of address bus for memory device)

  - 存储芯片、存储模块、子存储系统之间的关系为：
    - 存储芯片（device）通过字长以及字数扩展构成存储模块（bank），
    - 不同的存储模块通过存储控制器组成子存储系统（sub-system）。

▶ 基于存储控制器的接口

- Xilinx AXI存储控制器：SRAM接口信号
  - 【例6.7】已知异步SRAM芯片IDT71V416S为256K X 16b的存储芯片，支持字节访问。要求采用2片芯片通过存储控制器构建一个32位的存储模块，试设计存储控制器与存储芯片之间的接口。
    - IDT71V416S管脚和功能

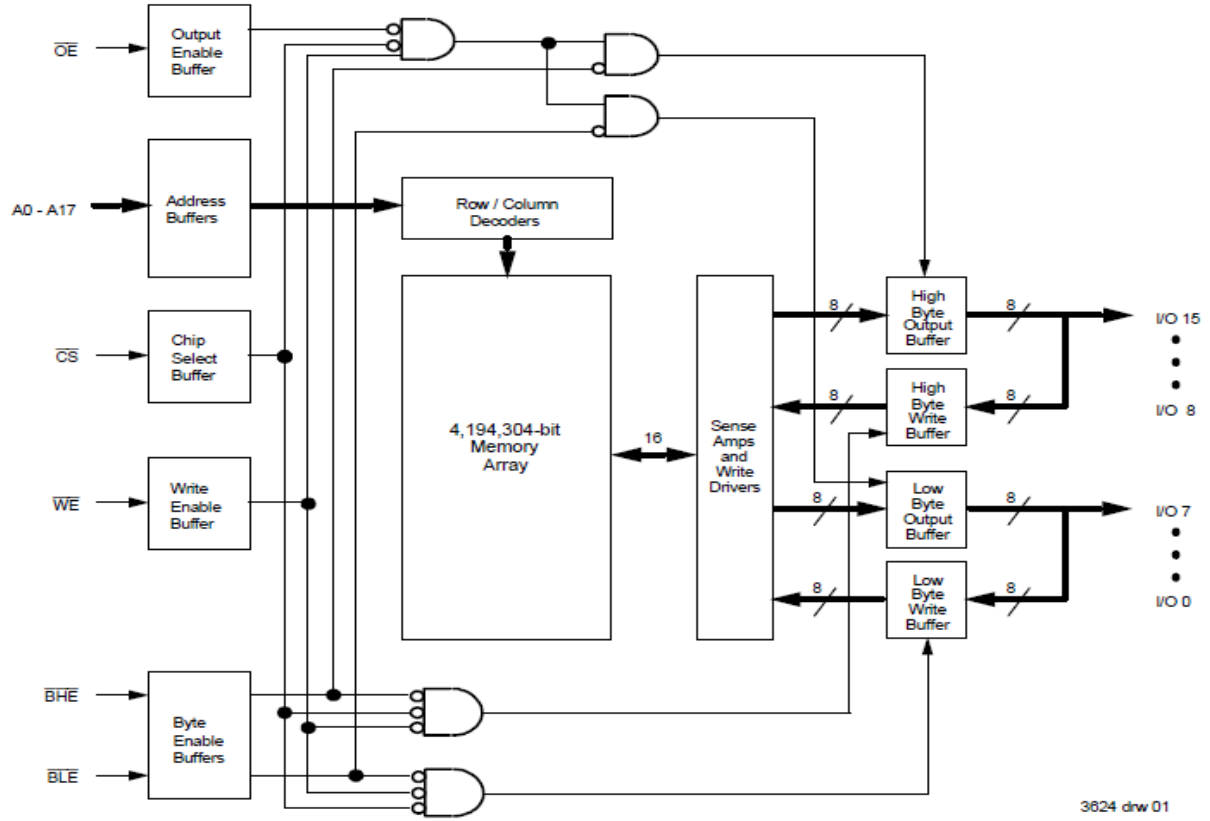


Pin Descriptions

A_0 - A_{17}	Address Inputs	Input
$\overline{CS}$	Chip Select	Input
$\overline{WE}$	Write Enable	Input
$\overline{OE}$	Output Enable	Input
$\overline{BHE}$	High Byte Enable	Input
$\overline{BLE}$	Low Byte Enable	Input
I/O_0 - I/O_{15}	Data Input/Output	I/O
V_{DD}	3.3V Power	Pwr
V_{SS}	Ground	Gnd

▶ 基于存储控制器的接口

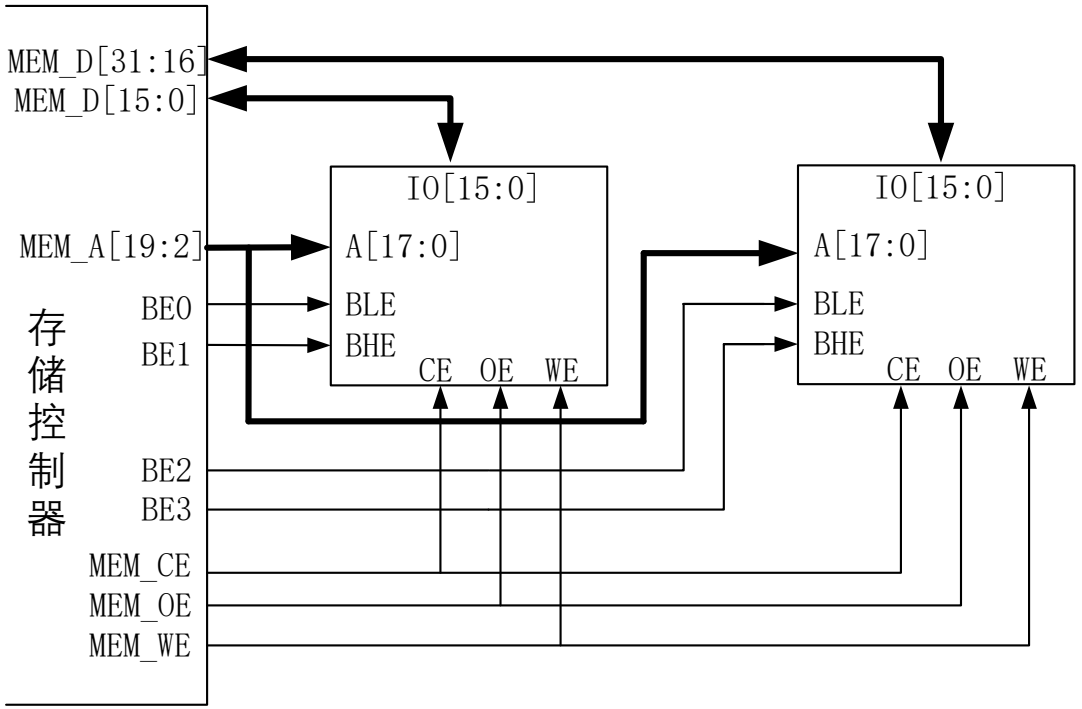
- Xilinx AXI存储控制器：SRAM接口信号
  - 【例6.7】已知异步SRAM芯片IDT71V416S为256K X 16b的存储芯片，支持字节访问。要求采用2片芯片通过存储控制器构建一个32位的存储模块，试设计存储控制器与存储芯片之间的接口。
    - IDT71V416S内部结构



▶ 基于存储控制器的接口

- Xilinx AXI存储控制器：SRAM接口信号
  - 【例6.7】已知异步SRAM芯片IDT71V416S为256K X 16b的存储芯片，支持字节访问。要求采用2片芯片通过存储控制器构建一个32位的存储模块，试设计存储控制器与存储芯片之间的接口。
    - 变量取值

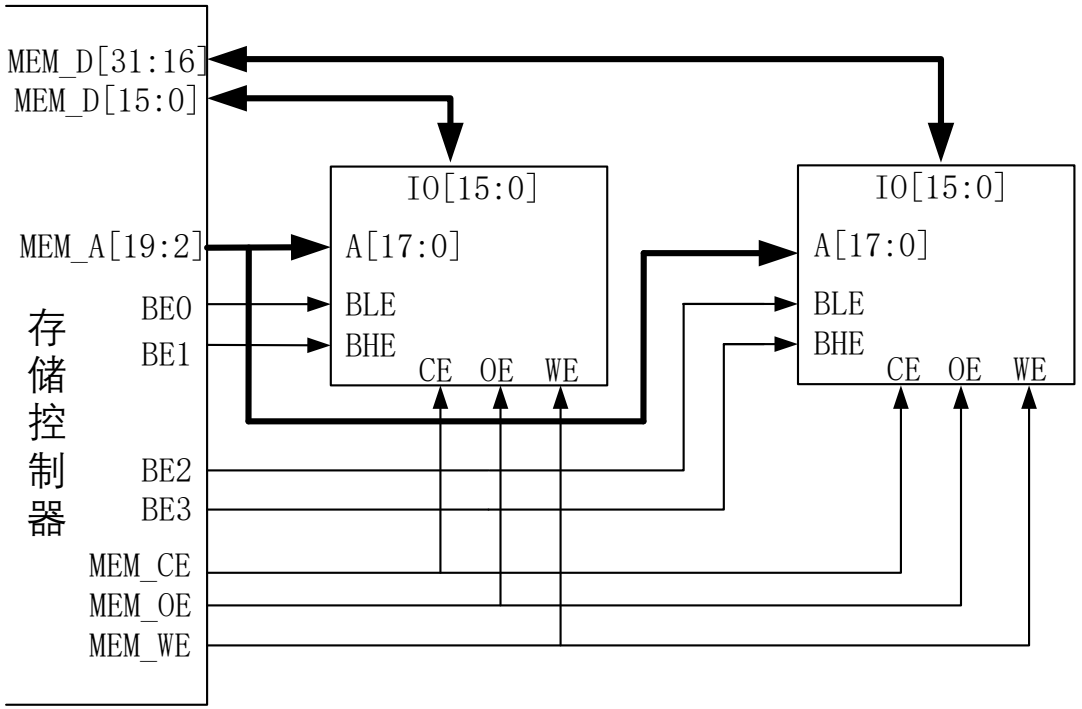
变量名	值	含义
BN	0	一个存储块，序号为0
DN	0，1	两个存储芯片，序号为0，1；其中0号为高位数据，1号为低位数据
MW	32	子存储系统数据位宽32位
DW	16	存储芯片数据位宽16位
MAW	18	存储芯片地址总线18位
AU	16	存储芯片字长16位
AS	2	地址偏移2位=$\log_2(32*16/16)/8$
HAW	32	地址总线32位（AXI总线）



▶ 基于存储控制器的接口

- Xilinx AXI存储控制器：SRAM接口信号
  - 【例6.7】已知异步SRAM芯片IDT71V416S为256K X 16b的存储芯片，支持字节访问。要求采用2片芯片通过存储控制器构建一个32位的存储模块，试设计存储控制器与存储芯片之间的接口。
    - 存储控制器与存储芯片之间的引线连接

芯片号	引脚含义	存储控制器段引脚名称	存储芯片引脚名称
1	数据线	MEM_DQ(15 : 0)	I/O(15 : 0)
	地址线	MEM_A(19 : 2)	A(17 : 0)
	芯片使能	MEM_CEN(0)	CS
	读使能	MEM_OEN	OE
	写使能	MEM_WEN	WE
	字节使能	MEM_BEN(1 : 0)	BHE:BLE
0	数据线	MEM_DQ(31 : 16)	I/O(15 : 0)
	地址线	MEM_A(19 : 2)	A(17 : 0)
	芯片使能	MEM_CEN(0)	CS
	读使能	MEM_OEN	OE
	写使能	MEM_WEN	WE
	字节使能	MEM_BEN(3 : 2)	BHE:BLE



Thanks