存储芯片

笔记源文件：Markdown，长图，PDF，HTML

1. 几种存储芯片

1.1. 存储器分类(按存取方式)

1.1.1. 存取方式

存取方式	本质特征	访问存储单元的时间
随机存取	基于地址，CPU给出地址，读写地址单元	与物理位置无关
顺序存取	基于顺序，数据读写顺序=数据的物理排列顺序	取决于数据位置，读写头位置
按内容存取	基于内容，CPU给出关键字*，读出匹配单元	N/A，但可基于硬件快速检索

*AM/CAM一个单元存储的信息=关键字+数据

1.1.2. 总述

存储器类型	本质特征	细分	用途
RAM(随机)	随机存取	SRAM(静态)，DRAM(动态)	#
ROM(只读)	随机存取，只读不写	PROM/EPROM/EEPROM/闪存	#
SAM(顺序)	顺序存取	#	磁带
AM/CAM(相联)	按内容读，按地址写	#	块表/页表

1.1.3. 细分

1️⃣RAM种类

RAM类型	本质特征	用途
SRAM	加电情况下，存储信息稳定	Cache
DRAM	存储单元内容长时间不访问就会失真，需要定期刷新	内存条

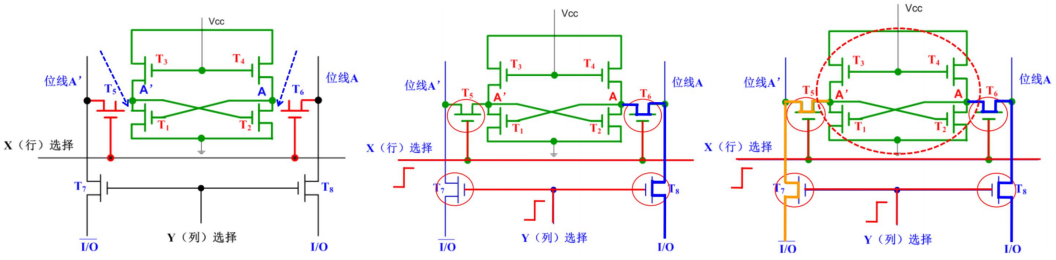
2️⃣ROM种类：按照写入方式不同

类型	本质特征	用途
Mask ROM	不可编程，生产时就烧入了不可改变的程序	#

ROM类型	本质特征	用途
PROM	可一次性编程，生产时不烧入程序，供用户编程后烧入	#
EPROM	可编程可擦除，能够多次编程了，但是一擦除就要全部擦掉	#
EEPROM	电可编程可擦除，相比EPROM进步在于，可分块/分字擦除	IC卡
闪存	快速版本的EEPROM	SSD，U盘

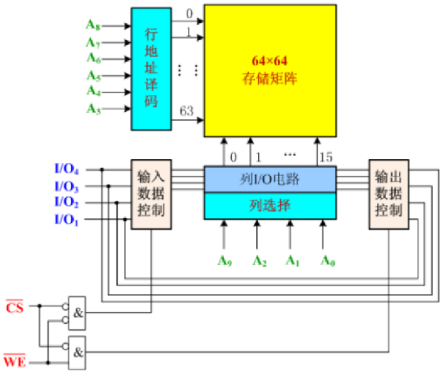
1.2. SRAM概述

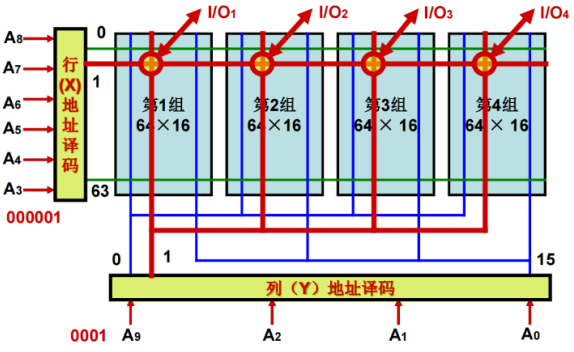
1.2.1. 六管SRAM存储元



状态	行列选择	操作
保持	$X \wedge Y = 0$	T_5, T_6断开，存储元与外界隔离
读出	$X \wedge Y = 1$	X连通T_6，Y连通T_8，数据经过右侧单边放大后读出
写入	$X \wedge Y = 1$	X连通T_5T_6，Y连通T_7T_8，双边写入(改变T_1T_2状态)

1.2.2. SRAM芯片举例：Intel 2114





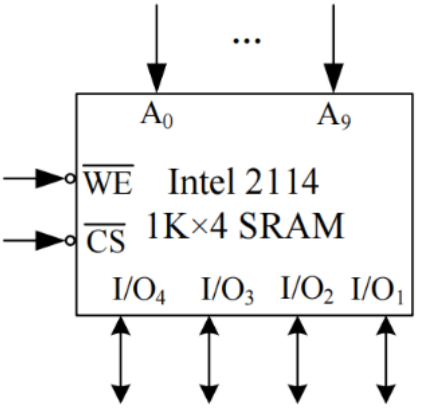
1. 芯片大小：1K×4位或者64×64位
   0. 64×64芯片由四颗一样的64×16组成
   1. 64×64位说明芯片一共有2^{12}个六管SRAM
   2. 4位说明一个地址单元由4个存储元构成，需要四根数据线，即图中$\mathrm{I/O_1} \rightarrow \mathrm{I/O_4}$
   3. 1K说明一共有1024个存储单元，需要10根地址线，即图中$\mathrm{A_1} \rightarrow \mathrm{A_{10}}$
2. 芯片布线：采用双译码结构

1. 横排布线(地址译码线):
    - 一共64条，通过译码器由6根地址线完成64选1
    - 64条横排布线的第k条，对应第1/2/3/4组芯片的第k行
2. 竖排布线(地址译码线):
    - 一共16条，通过译码器由4根地址线完成16选1
    - 每条竖排布线，一分为四插入四个芯片的同一列，所以一组芯片中也是16条竖排布线
    - n\text{16}条横排布线的第k条，对应第$1/2/3/4$组芯片的第k$列
3. 交点：每个芯片中的横竖排布线交点，对应一个六管SRAM存储元

3️⃣ 读写过程

1. 行选择:
    - 经过$A_8A_7A_6A_5A_4A_3$地址线输入信号
    - 经过译码后选择64条横布线中的第m条
    - 命中所有芯片的第m行
2. 列选择
    - 经过$A_9A_2A_1A_0$地址线输入信号
    - 经过译码后选择16条横布线中的第n条
    - 命中所有芯片的第n列
3. 命中
    - 横竖交叉，命中每种所有四组芯片的第m行第n列，共四位数据
    - 记该处的六管SRAM存储元值为$(\mathrm{m,n})$
4. 读写
    - 输出：$(\mathrm{m,n})_1 \xrightarrow{\text{输出}} \mathrm{I/O_1}$，$(\mathrm{m,n})_2 \xrightarrow{\text{输出}} \mathrm{I/O_2}$，$(\mathrm{m,n})_3 \xrightarrow{\text{输出}} \mathrm{I/O_3}$，$(\mathrm{m,n})_4 \xrightarrow{\text{输出}} \mathrm{I/O_4}$
    - 输出：$(\mathrm{m,n})_1 \xleftarrow{\text{写入}} \mathrm{I/O_1}$，$(\mathrm{m,n})_2 \xleftarrow{\text{写入}} \mathrm{I/O_2}$，$(\mathrm{m,n})_3 \xleftarrow{\text{写入}} \mathrm{I/O_3}$，$(\mathrm{m,n})_4 \xleftarrow{\text{写入}} \mathrm{I/O_4}$

❎ 补充：芯片的外特性



A9~A0：地址线

$\mathrm{I/O_4}$~$\mathrm{I/O_1}$：数据线，双向

$\overline{\mathrm{WE}}$：读写读令

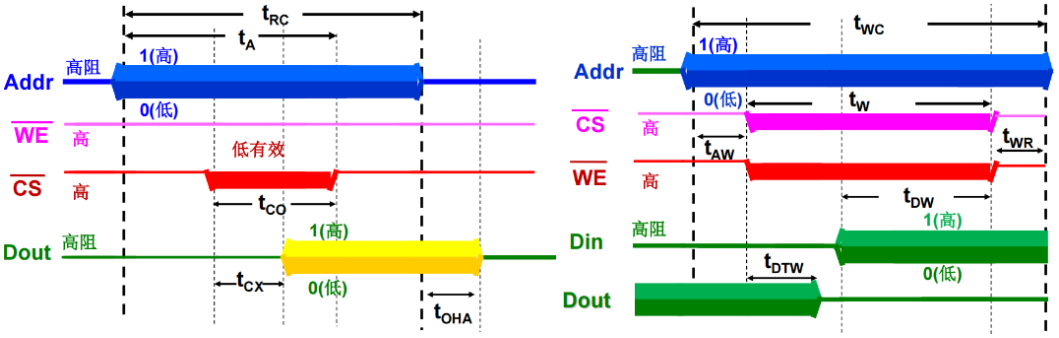
=H：读令

=L：写令

$\overline{\mathrm{CS}}$：片选

低有效（选中）

1.2.3. SRAM的读写时序(左读/右写)

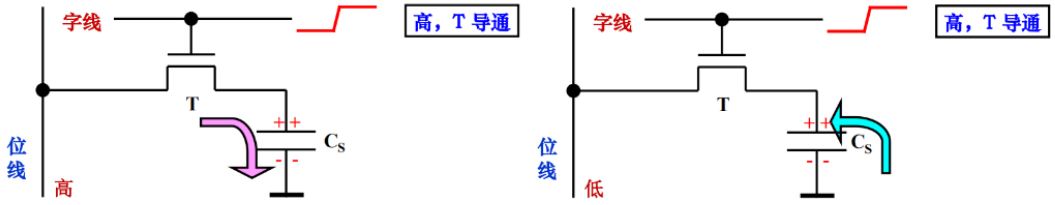


以读时序为例

1. CPU发送有效地址：t_A用于给出地址，$t_{RC}-t_A$的时间里依然给出有效地址目的在于防干扰
2. 片选：t_{CO}内片选信号有效，在这期间命中要读出的存储单元
3. 输出：D_{out}有效代表数据通过数据线输出，t_{OHA}持续输出一会称为恢复时间

1.3. DRAM概述

1.3.1. 单管DRAM存储元



1. 读写操作

操作	字线行为	位线行为
写入1	高电平(选中)	高电平
写入0	高电平(选中)	低电平
读出1	高电平(选中)	放电
读出0	高电平(选中)	无电流

2. 再生操作

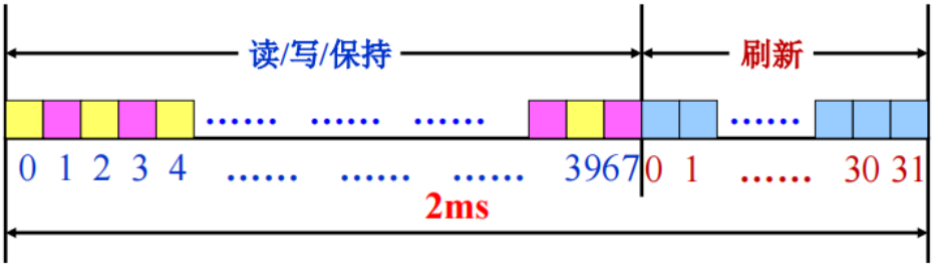
    1. 读出1过程中：
        - 单元中的数据被读走，清零
        - 读出过程中经过再生/读放大器，数据在放大器中维持
    2. 再生：放大器中的数据自动写回被清零的存储单元

3. 保持操作：

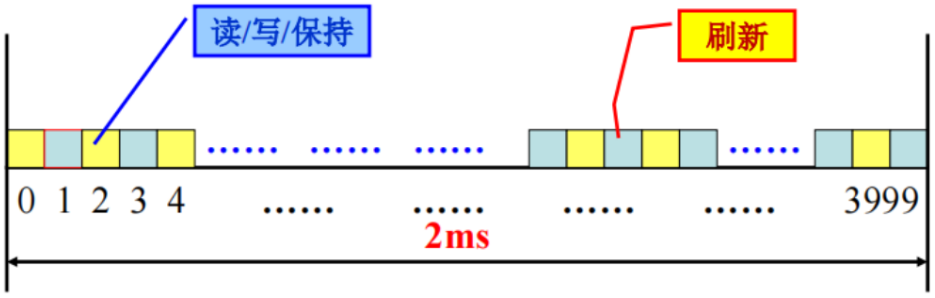
    1. 字线低电平时(未选中)，隔绝外界使得电容不漏电
    2. 不漏电是不可能的，所以需要定时刷新

1.3.2. DRAM刷新

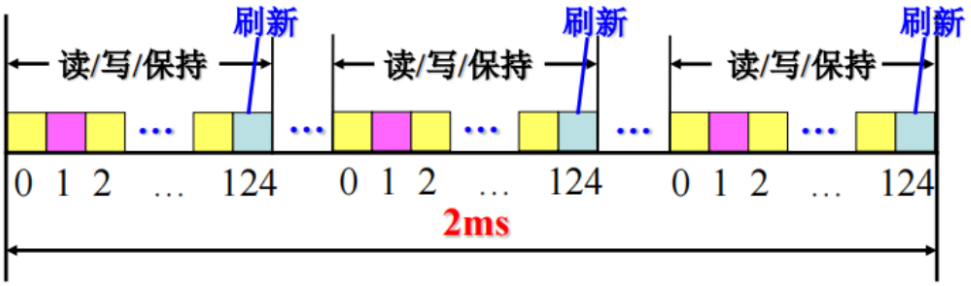
1️⃣集中式：最大刷新间隔(2ms)内，对芯片内全部存储元逐行刷新一遍，刷新时不读写



2️⃣分散式：每读/写一次我就刷新一次(太频繁了！)，每次只刷新一行，每行轮着来



3️⃣异步刷新：逐行刷新，但是刷新某一行的时间点平均分散在最大刷新间隔(2ms)内

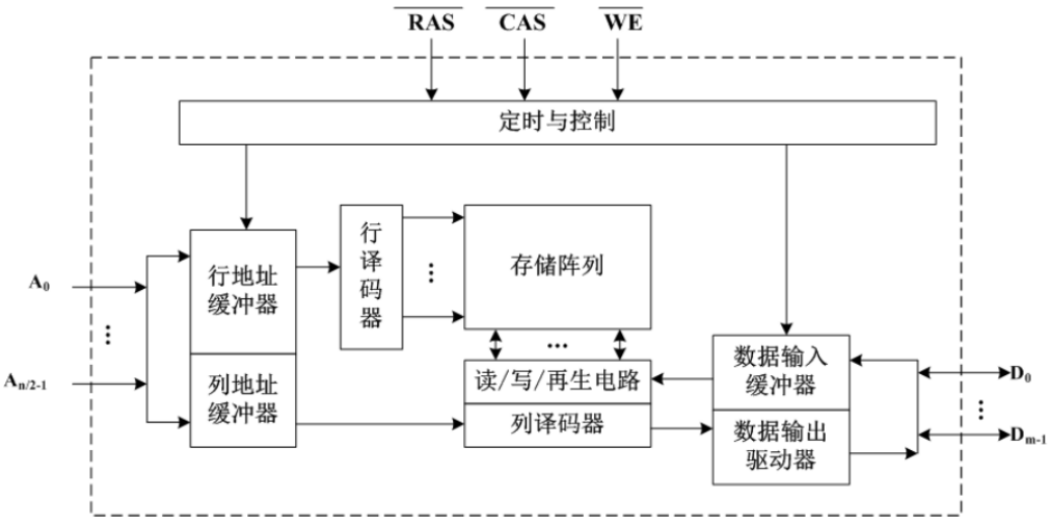


例题：

1. 条件
   - 用64K×1位的DRAM芯片构成1M×16位的主存储器阵列，
   - 芯片内部存储元排列成正方形整列
   - 其刷新周期最大时间间隔为4ms，采用异步刷新时
2. 提问：两次刷新操作的最大时间间隔
3. 结构分析
   - 64K×1位：表示DRAM其实是256位×256位的
   - 所需芯片总数：1M/64K=16，16/1=16，共需要16×16=256块DRAM
   - 芯片阵列：DRAM在阵列中的排布为16×16
4. 刷新时间计算：
   - 由以上分析可得，对单个DRAM而言需要在4ms内完成对256行的刷新
   - 所以时间间隔不超过：4ms/256=15.625us(千万别四舍五入)

1.3.3. DRAM结构与举例

1 外部结构



1. 结构特点：分行列地址

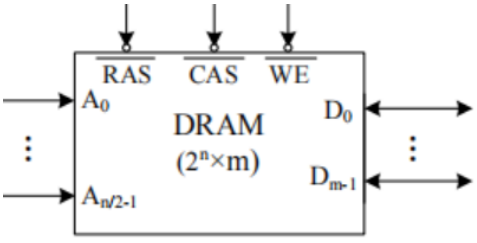
操作	$\overline{RAS}$	$\overline{CAS}$
送行地址	有效，表示行地址&片选	无效
送列地址	有效，表示片选	有效，表示列地址

2. 输送地址过程：

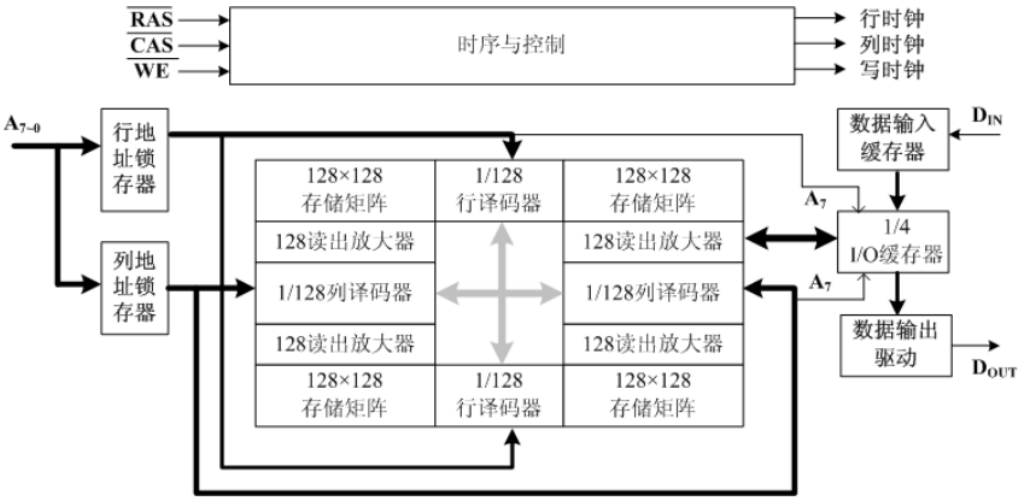
   地址信号→高位冲入行地址缓存/低位冲入列地址缓存→行列分别译码→命中地址单元

3. 数据读写：经过输入/输出缓冲器，实现写/读

4. 外特性



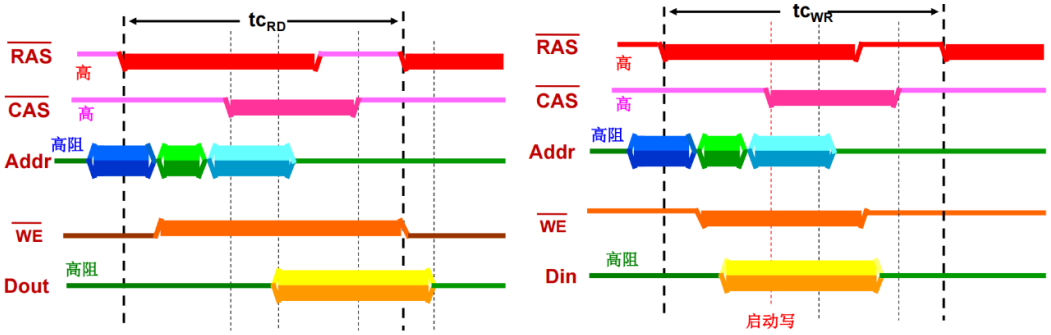
2 Intel 2164A DRAM



1. 结构：
    - 大小：64K×1位
    - 阵列：一分为四后2×2二维排列，每两块之间共用一个译码器
2. 译码过程
    - 输入地址后分为三部分——最高位A_7/行地址/列地址，送各自缓存
    - 最高位A_7冲入I/O缓存器，负责选定四个芯片中的一个
    - 选定芯片后，其连接的行/列译码器启动，通过输入的行/列地址完成译码，命中

1.3.4. DRAM读写时序

注意：读快写慢，Addr中蓝色是送行地址/青色是送列地址

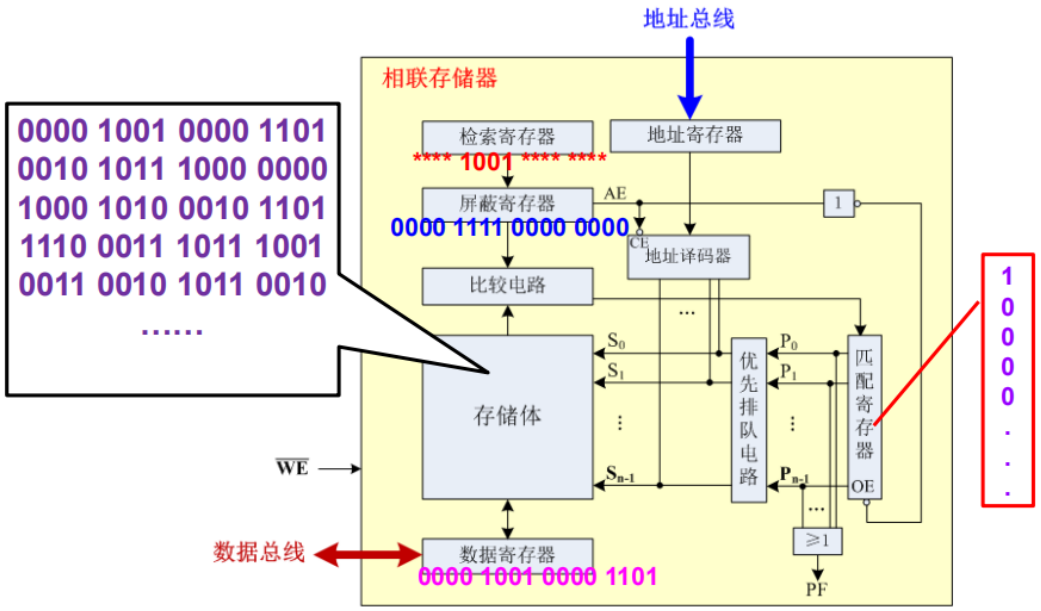


以读时序为例

1. Addr处行地址先准备好，RAS下降，读取行地址/片选
2. RAS保持片选，CAS下降，读取Addr处的列地址(已经准备好)
3. 行列地址都有了，WE高电平开始读取数据
4. 一段时间后Dout将数据输出

1.4. CAM/AM概述

1.4.1. 组成



1.4.2. 读过程：可以1:N比较

1 关键字的获取

1. 检索寄存器：暂存检索字，其中$a \to b$位是关键字，如
   XXXX 1001 XXXX XXXX
2. 屏蔽寄存器：提取检索字中的关键字，$a \to b$位是1其余为0，如
   0000 1111 0000 0000
3. 关键字 = 检索寄存器∧屏蔽寄存器，本例中关键字=0000 1001 0000 0000

2 匹配：

1. 比较电路：获得的关键字$\xleftarrow[\text{比较电路}]{\text{匹配}}$存储单元对应位的内容，在此处就是找出4→7位是1001的单元
2. 匹配寄存器OE：
   - 匹配寄存器中的一位，对应一个CAM的存储单元
   - 某个存储单元匹配上了后，匹配寄存器相应位被置1
   - PF是匹配寄存器所有位与在一起的结果，PF=1表示有匹配
3. 以此类推匹配完所有单元，匹配寄存器为1的位对应的单元为匹配成功的单元

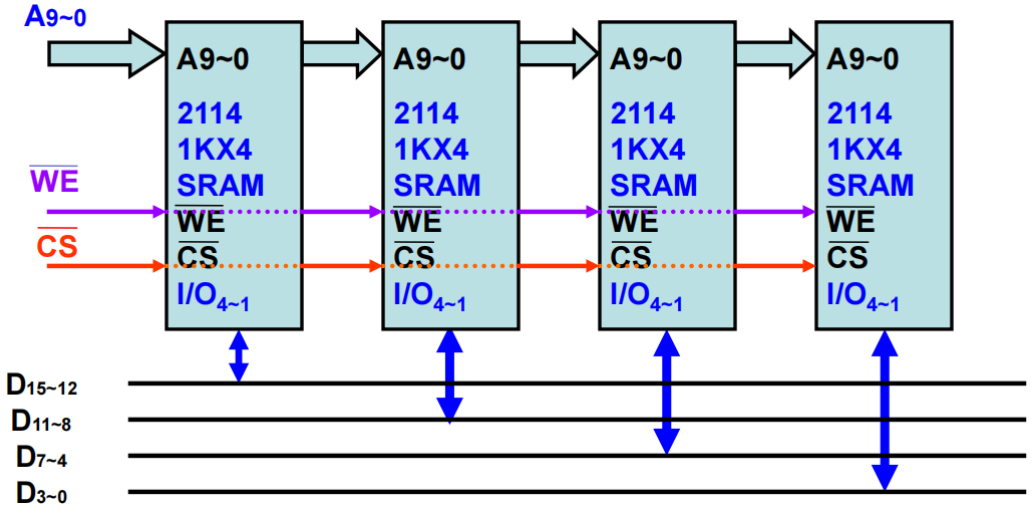
3 排队输出：适用于多个单元匹配成功

1. 匹配寄存器中有多位1，仅保留最靠前的1将其对应的单元数据输出到数据寄存器，后面清0
2. 优先排队电路：$S_i = \overline{P_0 P_1 \dots P_{i-1}} \cdot P_i$
   - $P_0 \to P_i$是指匹配寄存器的$0 \to i$位
   - $S_i = 1$时，匹配寄存器第i位，对应的存储单元就输出

2. 容量扩展：存储芯片的连接

2.1. SRAM容量扩展

2.1.1. 位扩展：芯片串联(输出拼接)



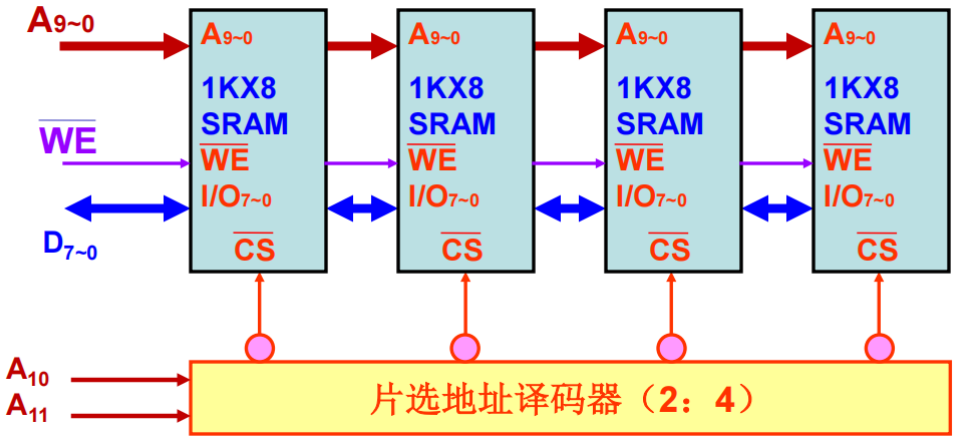
1 信号

符号	含义
$A_{0\sim9}$	10根地址线，每根A_i同时连接四个芯片
$\overline{CS}$	片选线，四个芯片共用同一片选
$\overline{WE}$	读写控制线，四个芯片共用；高电平为读/低电平为写
$D_{15\sim12}/D_{11\sim8}/D_{7\sim4}/D_{3\sim0}$	数据线，分别接收来自第一/第二/第三/第四片芯片的四位数据

2 扩展逻辑

1. 原有的芯片为$1\mathrm{K}\times4$只能输出4位的数据，而存储器是16位的(数据必须是16位)
2. 4个芯片横排，每个分别提供四位，拼成16位

2.1.2. 位扩展：芯片并联(片选挑一)



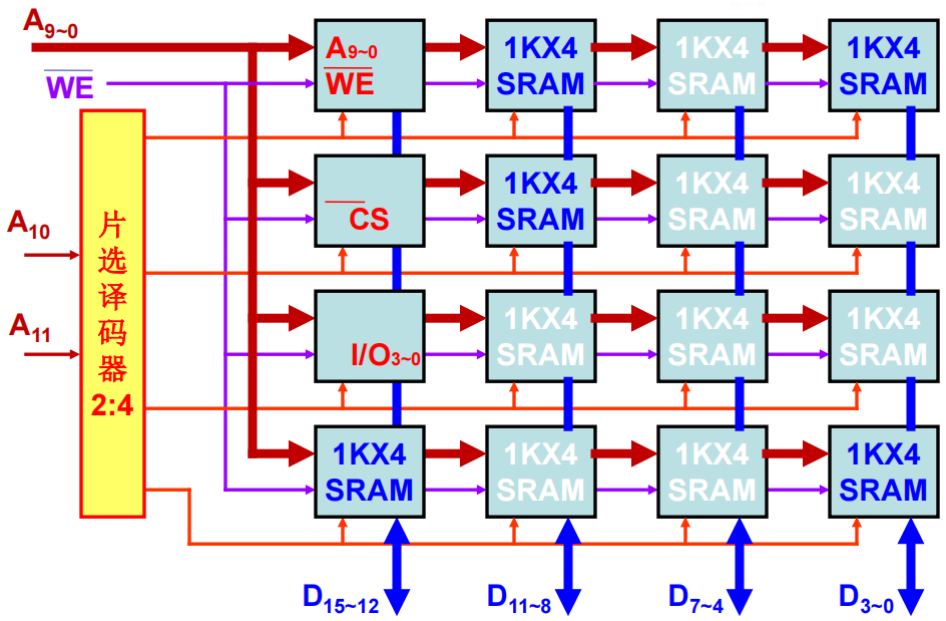
1 信号

符号	含义
$A_{0\sim9}$	10根地址线，每根A_i同时连接四个芯片
$A_{10\sim11}$	两根地址线，作为译码器的输入，完成片选信四选一
$\overline{CS}$	片选线，四个芯片四个片选，由译码器四选一决定用哪个片选
$\overline{WE}$	读写控制线，四个芯片共用
$D_{7\sim0}$	数据线，四个芯片共用，但同一时刻只能有四个中的一个通过这堆数据线输出数据

2 扩展逻辑

1. 地址信号高两位$A_{10\sim11}$：用于选定四个芯片之一，非选定的芯片保持不活跃
2. 地址信号其他位$A_{0\sim9}$：用于在选定的芯片内，命中要读写的地址单元

2.1.3. 字位扩展：芯片二维阵列

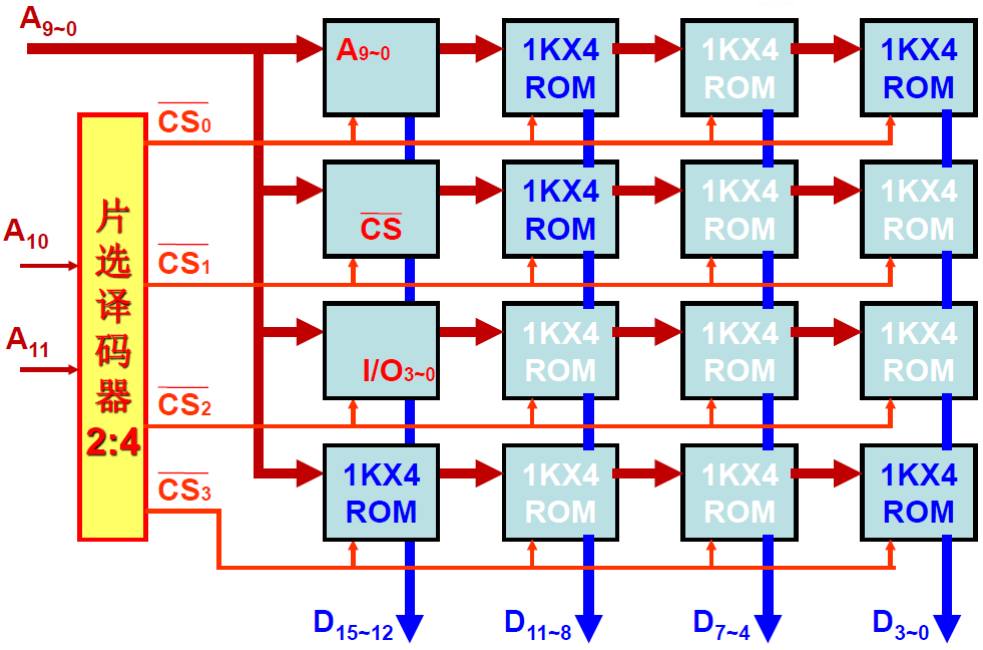


1 先用字扩展逻辑，选定一行的芯片

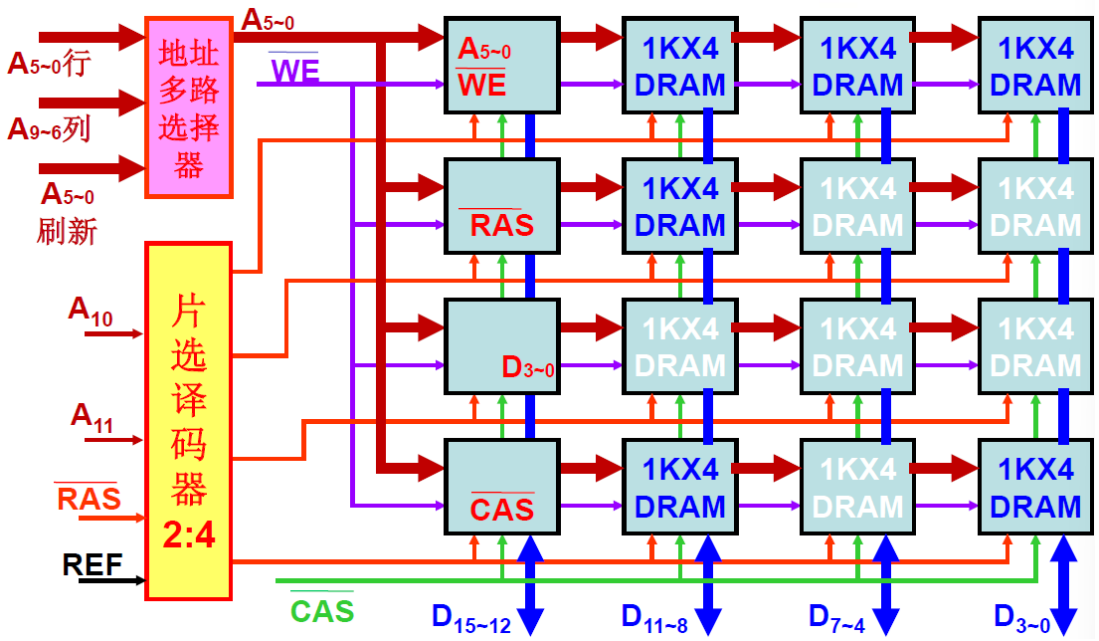
2 再用位扩展逻辑，让选定行的芯片， 每个芯片输出四位，拼成12位输出

2.2. ROM容量扩展

直接参考SRAM的，剔除WE引脚即可



2.3. DRAM容量扩展



直接参考SRAM的，改变如下

1. 地址线A_i分行/列地址，经过多路选择器分时送往芯片地址端
2. 加入送行地址信号$\overline{RAS}$，送列地址信号$\overline{CAS}$
   1. $\overline{RAS}$加到片选译码器上
   2. $\overline{CAS}$直接按照并联逻辑加到每个芯片上
3. 片选译码器加入REF
   1. REF有效时，四行芯片同时刷新
   2. 配合$A_{0\sim5}$行地址，以此来决定刷新芯片的哪一行

3. 存储芯片系统和CPU的连接

3.1. 连接要点

1. 三种线的连接：地址线/数据线/控制线

连线	单个芯片	字/位扩展后
数据线	芯片数据线<-一一对应->总线*数据线	扩展芯片数据线<-一一对应->总线数据线
地址线	芯片地址线<-一一对应->总线地址线	扩展芯片地址线<-连接->总线地址线低位 片选信号<-连接->总线地址线高位，如上例中的$A_{10\sim11}$
控制线**	芯片控制端<-一一对应->总线控制线	芯片控制端<-多对一->总线控制线

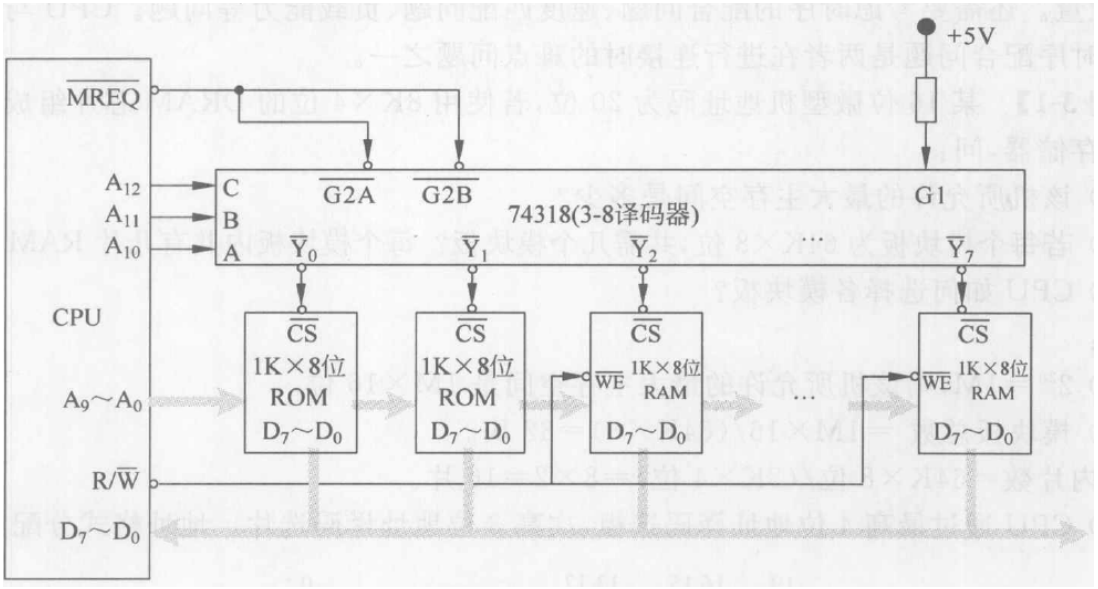
*总线一律指系统总线

**控制线，主要是读写控制线

2 CPU的$\overline{\text{MRQE}}$控制信号

1. $\overline{\text{MRQE}}$有效时译码器才能产生有效的片选信号
2. $\overline{\text{MRQE}}$可用于区分访问内存/访问IO

3.2. SRAM/ROM与CPU的连接



1 SRAM需要连接R/$\overline{\text{W}}$信号，而ROM不用，这一点在图中得以体现

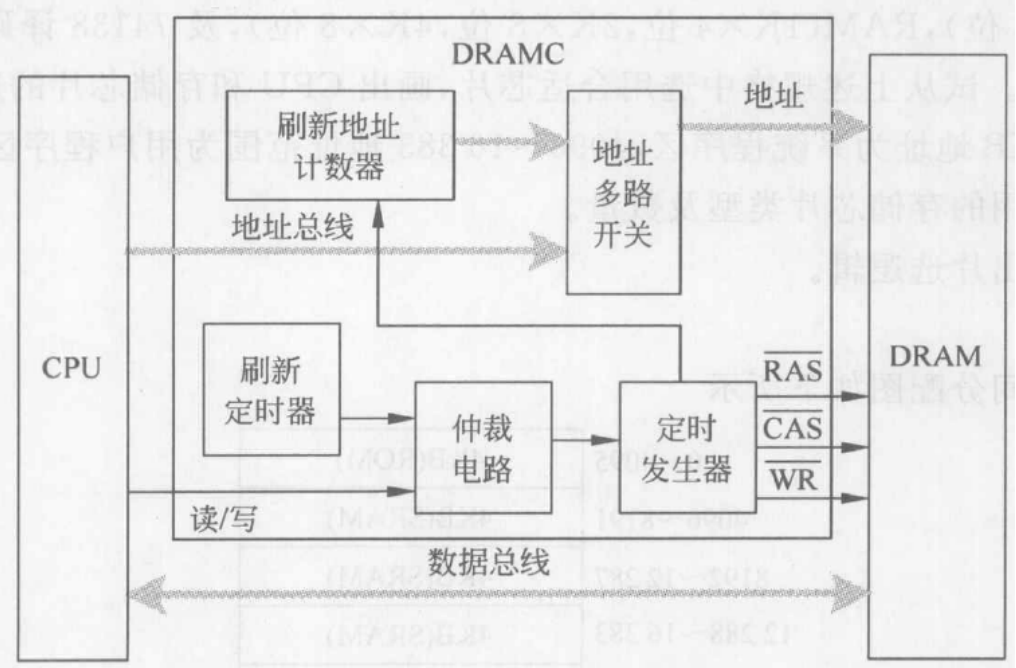
2 74318译码器的工作逻辑

信号	含义
ABC	输入地址的高位$A_{12}A_{11}A_{10}$
$\overline{Y_0}\overline{Y_1}\overline{Y_2}\overline{Y_3}\overline{Y_4}\overline{Y_5}\overline{Y_6}\overline{Y_7}$	输出的片选信号$\overline{CS}$

3 74318译码器的控制逻辑：译码器只能在如下状态，才能启动工作

$\overline{MRQE}:\overline{G2A}$	$\overline{MRQE}:\overline{G2B}$	G_1
0	0	1

3.3. DRAM与CPU的连接：基于DRAMC



结构	描述
刷新地址计数器	记住这次刷新的行，然后下一次让DRAM刷新下一行
地址多路选择器	选行/列/刷新地址之一送给DRAM
刷新定时器	定时给DRAM信号，让DRAM执行刷新
仲裁电路	当CPU/IO要来访存，同时DRAM有需要刷新了，执行二选一操作
定时发生器	产生各种控制信号如RAS/CAS/WE等

4. 存储扩展-CPU例题1

4.1. 条件

1 CPU

1. 有16根地址线(供$2^{16}=65536$个地址单元)，8根数据线(数据为8位)
2. $\overline{\text{MRQE}}$为访存控制信号，低电平有效
3. $\text{R}/\overline{\text{W}}$为读写命令信号，高电平读/低电平写

2 存储芯片

1. ROM三个：2K×8位，4K×4位，8K×8位
2. RAM三个：1K×4位，2K×8位，4K×8位

3 其他电路元件：

1. 74318译码器一个
2. 各种门电路，数量不限

4.2. 要干啥

1️⃣设计合适的存储器-CPU芯片

2️⃣满足最小4KB地址为系统程序区(只读)，$4096 \sim 16383$地址范围为用户程序区(可读可写)

4.3.三小问&解答

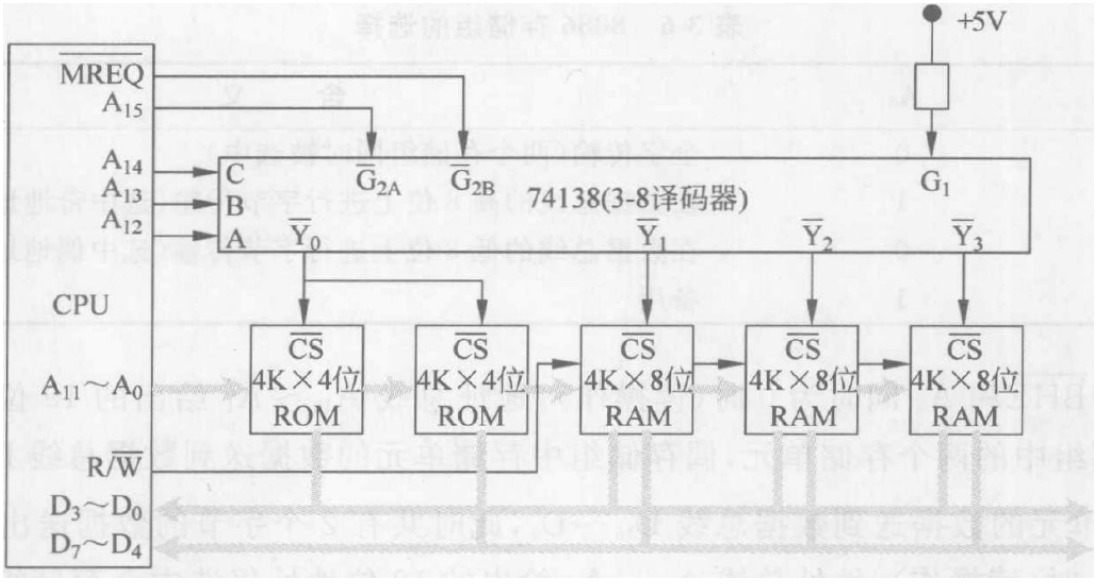
1️⃣画出地址空间分配图：

地址	大小	存储芯片
$00000 \sim 04095$	4KB	ROM
$04096 \sim 08191$	4KB	SRAM
$08192 \sim 12287$	4KB	SRAM
$12288 \sim 16383$	4KB	SRAM
$16384 \sim 65535$	#	#

2️⃣给出选用的存储芯片类型&数量

1. 注意事项
   - 芯片一样多时优先采用位扩展，因为字扩展还要译码，位扩展只需分片与数据线连接
   - ROM/RAM混用时，应该尽量选择外特性(如容量)一致的芯片
   - 避免二级译码
2. 芯片选择：存储系统 $\xleftarrow{\text{四片芯片四选一}}$ $\begin{cases} 4\text{K}\times8\text{位,一片} \xleftarrow{\text{位扩展}} \text{ROM, } 4\text{K}\times4\text{位, 两片} \\ \text{SRAM, } 4\text{K}\times8\text{位, 三片} \end{cases}$

3️⃣芯片与CPU的连接



1. 地址线：所有芯片无论多少位，都有$4\text{K}=2^{12}$个地址单元，分配低12个地址线用于芯片内寻址
2. 译码器：
   - 分配中间三个地址线用于输入译码器，最高地址线/$\overline{\text{MRQE}}$作为控制信号

    - 译码器输出只采用前四个端口，其余废掉
3. $\mathrm{R}/\overline{\mathrm{W}}$：只需要连接$\mathrm{SRAM}$
4. 数据的读写
    - 译码器选定$\overline{\mathrm{Y}}_0$后：两个ROM一个贡献低四位数据，一个贡献高四位数据，拼成8位
    - 译码器选定$\overline{\mathrm{Y}}_1\overline{\mathrm{Y}}_2\overline{\mathrm{Y}}_3$之一后：选定的$\mathrm{SRAM}$将8位数据，与八条数据线一一对应输出