数字电路
Digital Circuits and System

李文明
liwenming@ict.ac.cn

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES
中国科学院大学
University of Chinese Academy of Sciences

半导体存储器

半导体存储器

- 存储器概述
- 只读存储器（ROM）
- 随机存储器（RAM）
- 存储器容量的扩展
- 用存储器实现组合逻辑函数

半导体存储器

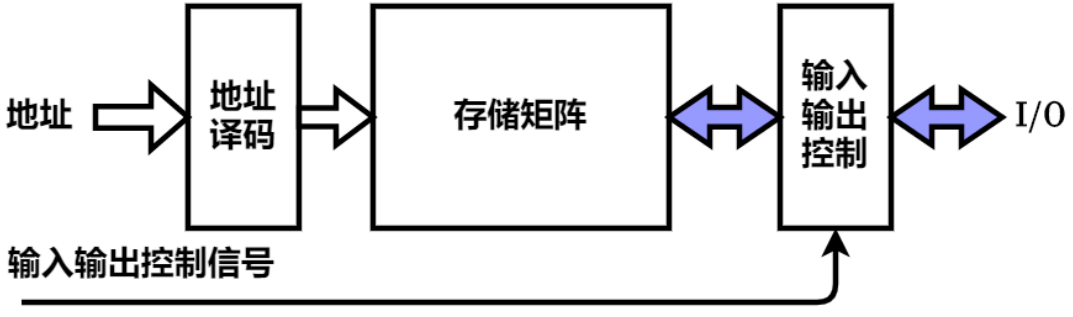
- 存储器概述
- 只读存储器（ROM）
- 随机存储器（RAM）
- 存储器容量的扩展
- 用存储器实现组合逻辑函数

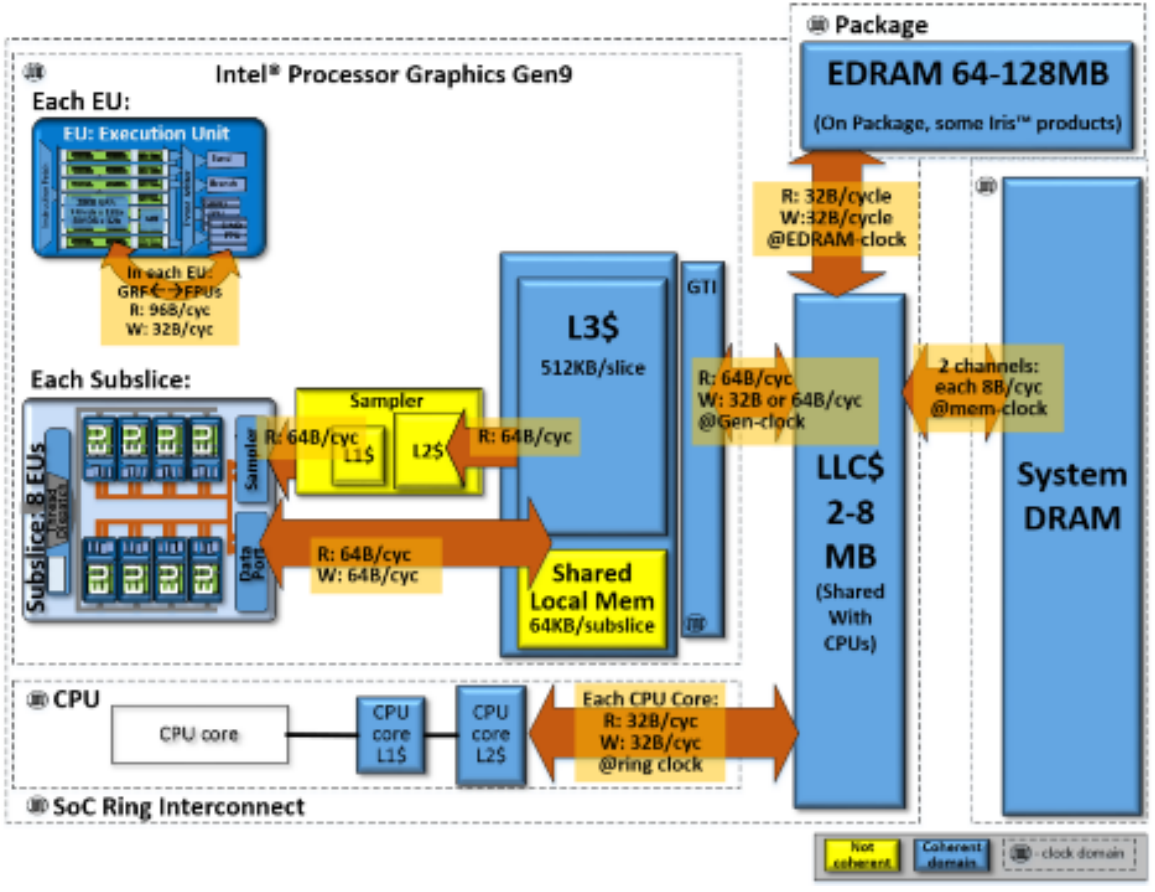
存储器概述

- 定义：能存储大量二值信息（或称为二值数据）的半导体器件

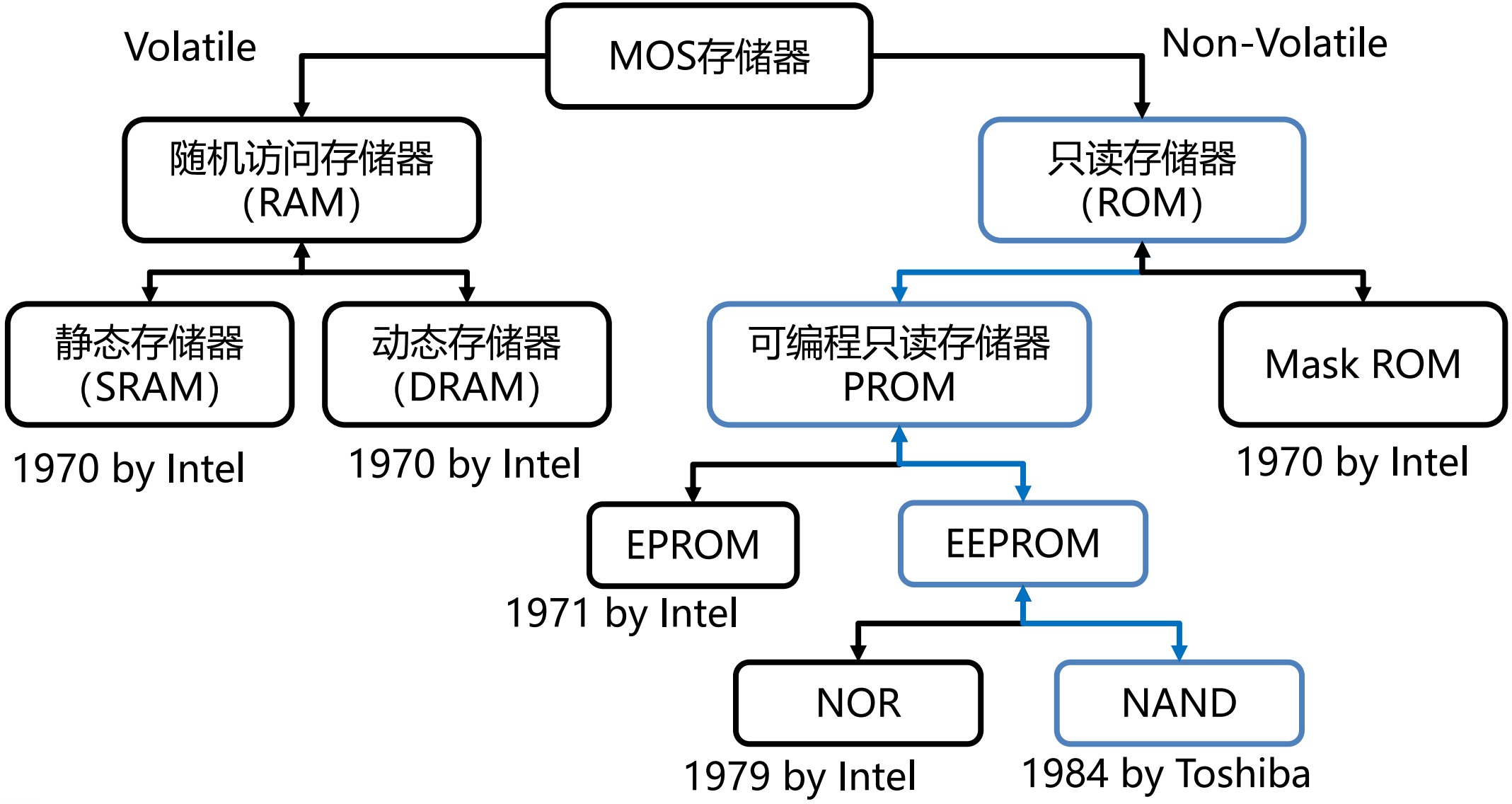
- 一般结构
  - 存储单元多
  - 输出/输入引脚有限
  - 多芯片容量扩展

- 性能指标
  - 存储容量
  - 读写速度
  - 读写次数





半导体存储器分类


MOS存储器
Volatile
Non-Volatile
随机访问存储器
(RAM)
只读存储器
(ROM)
静态存储器
(SRAM)
动态存储器
(DRAM)
可编程只读存储器
PROM
Mask ROM
1970 by Intel
1970 by Intel
1970 by Intel
EPROM
EEPROM
1971 by Intel
NOR
NAND
1979 by Intel
1984 by Toshiba

存储器基本概念

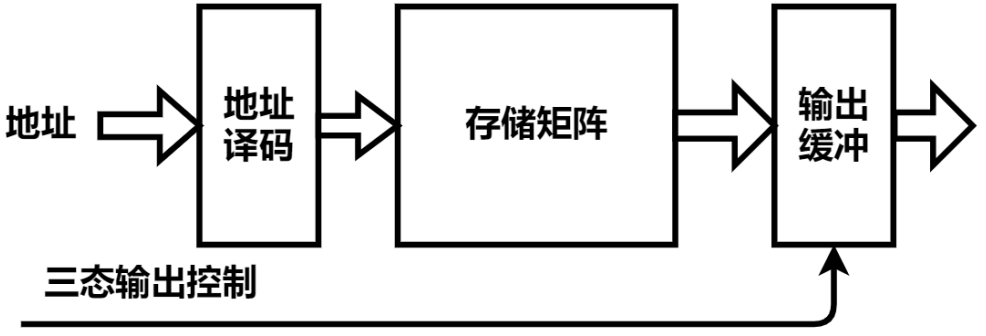
- RAM: 随机存储器 (Random Access Memory)
  - 指一类具备读写功能的存储器
  - 历史上存储器的阵列的每一位都可以独立访问，目前以字节（8bits）为单位
- ROM: 只读存储器 (Read Only Memory)
  - 不具备“在线”写操作功能
  - 写操作一般需要高电压，擦除需要紫外线或高电压
- 易失性存储器 (Volatility of Memory)
  - 易失性存储器掉电后会失去存储的数据，如RAM是易失存储器
  - 非易失存储器的数据会一致存在，不受掉电的影响，如ROM是非易失存储器
- 动态和静态存储器 (Static vs. Dynamic Memory)
  - 静态：只要供电存在，存储器的数据就能保持（SRAM）
  - 动态：存储器需要周期性刷新，否则数据就会丢失（DRAM）

半导体存储器

- 存储器概述
- 只读存储器（ROM）
- 随机存储器（RAM）
- 存储器容量的扩展
- 用存储器实现组合逻辑函数

掩膜只读存储器(ROM) （1）

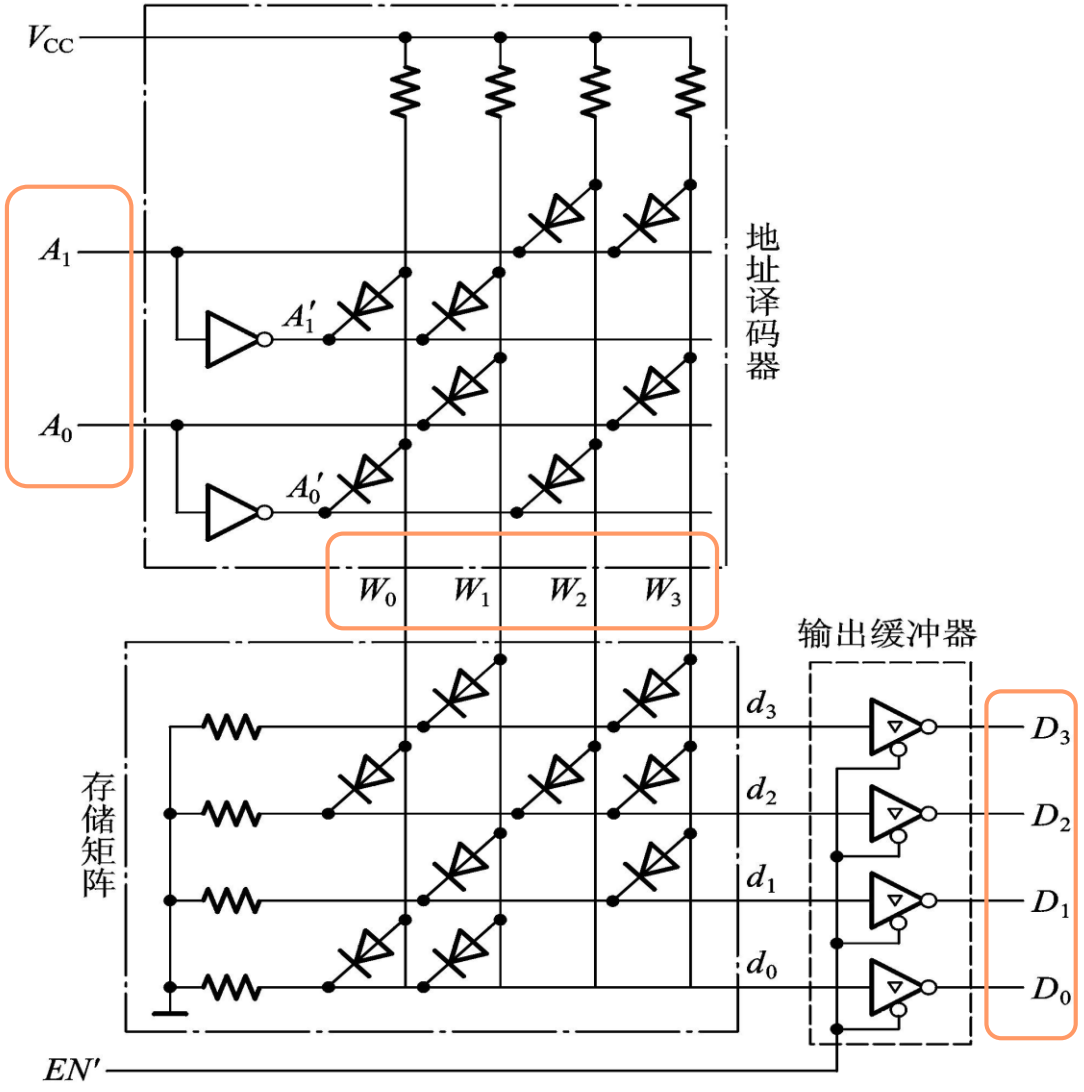
- **存储矩阵**
  - 由许多存储单元排列而成
  - 存储单元可以是二极管、双极型三极管或MOS管，每个单元能存放1位“0”或“1”
  - 而每一个或一组存储单元有一个相应的地址



- **地址译码器**：将输入的地址代码译成相应的控制信号，从存储矩阵中将指定的单元选出，并把其中的数据送到输出缓冲器
  - 存储器容量：地址线为n位，可以译码出2^n个地址单元，如果每个单元是M位二进制数，**则总容量= $2^n \times M$（bits)**
  - 存储器编制：**0~ $2^n - 1$**
- **输出缓冲器**：实现对输出状态的三态控制，以便与系统的总线连接

掩膜只读存储器(ROM) (2)

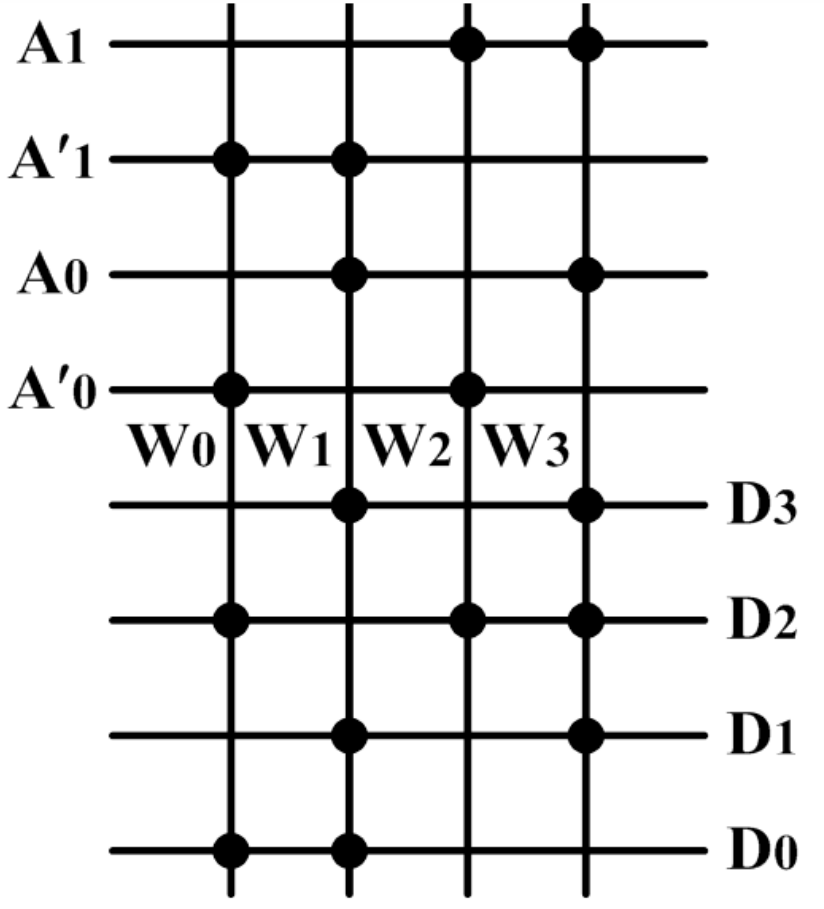
- “与”阵列：译码出字线
- W_0~W_1为字线， D_0~D_3为位线，交叉的点就是一个存储单元，交叉点的数目即为存储单元数，**存储容量=字数×位数**
- “或”阵列：存储“bits”
- 输出缓冲
  - 提高带负载能力
  - 电平变换
  - 对输出的三态控制，以便与总线相连



掩膜只读存储器(ROM) (3)

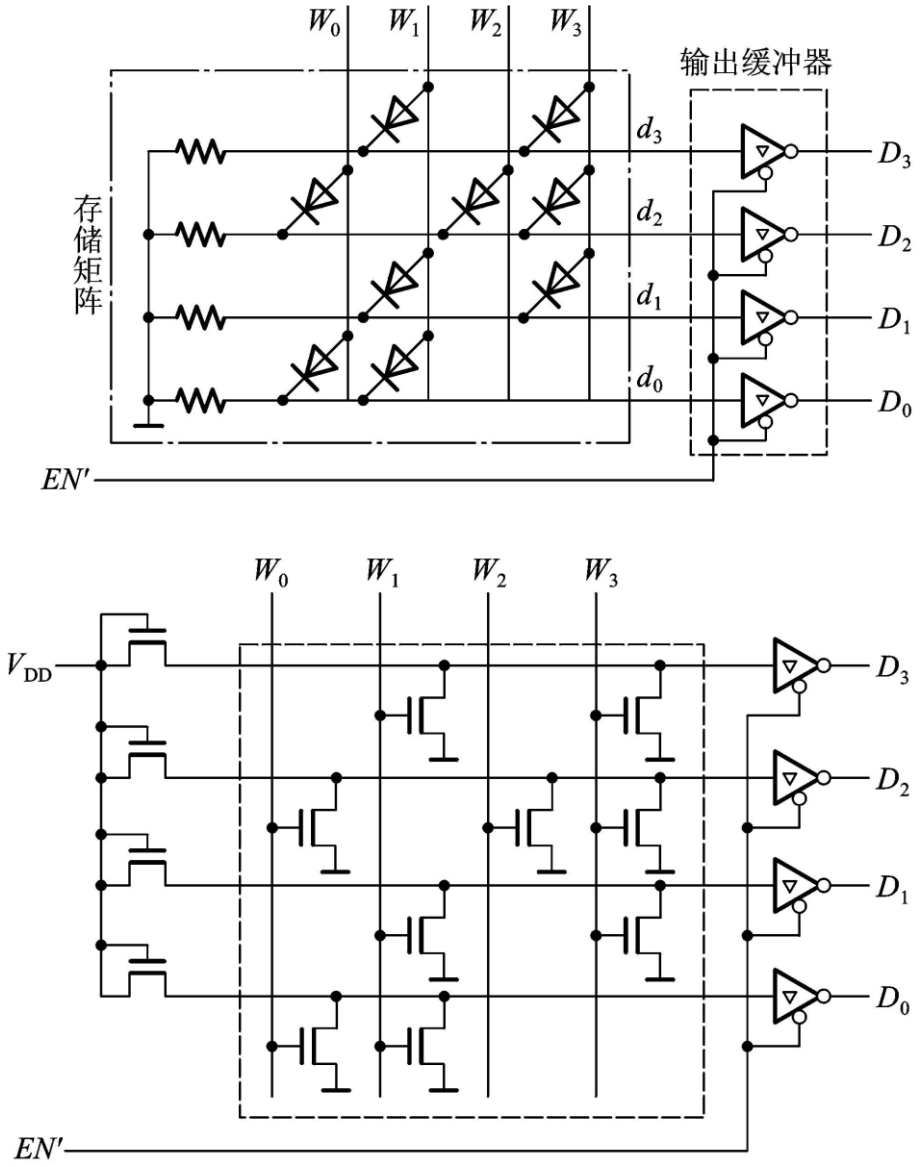
- 译码 $\begin{cases} W_0 = A_1' A_0' \\ W_1 = A_1' A_0 \\ W_2 = A_1 A_0' \\ W_3 = A_1 A_0 \end{cases}$ ， $\begin{cases} D_0 = W_0 + W_1 \\ D_1 = W_1 + W_3 \\ D_2 = W_0 + W_2 + W_3 \\ D_3 = W_1 + W_3 \end{cases}$

地址		译码输出				数据输出			
A_1	A_0	W_3	W_2	W_1	W_0	D_3	D_2	D_1	D_0
0	0	1	0	0	0	0	1	0	1
0	1	0	1	0	0	1	0	1	1
1	0	0	0	1	0	0	1	0	0
1	1	0	0	0	1	1	1	1	0



掩膜只读存储器(ROM) (4)

- ROM是组合逻辑电路，每个字线对应输入变量的最小项，位线是最小项的或，所以ROM可实现逻辑函数的“与 - 或”标准式
- 按用户要求专门设计，出厂时内部数据已固定
- 结构简单，价格便宜，非易失性
- 存储单元可以用二极管，也可以用MOS管实现

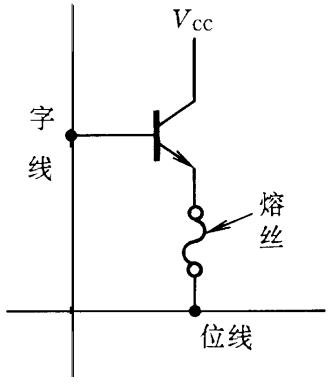


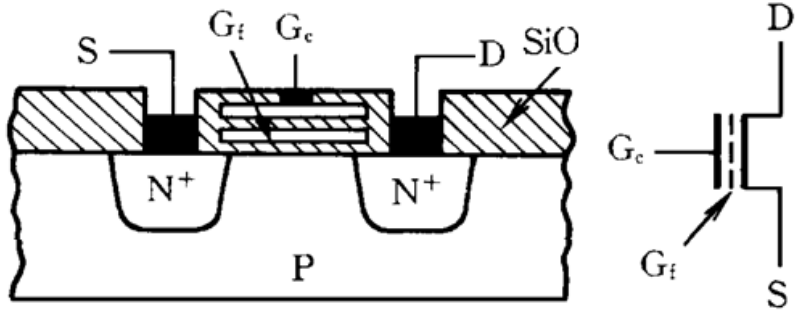
可编程只读存储器PROM(1)

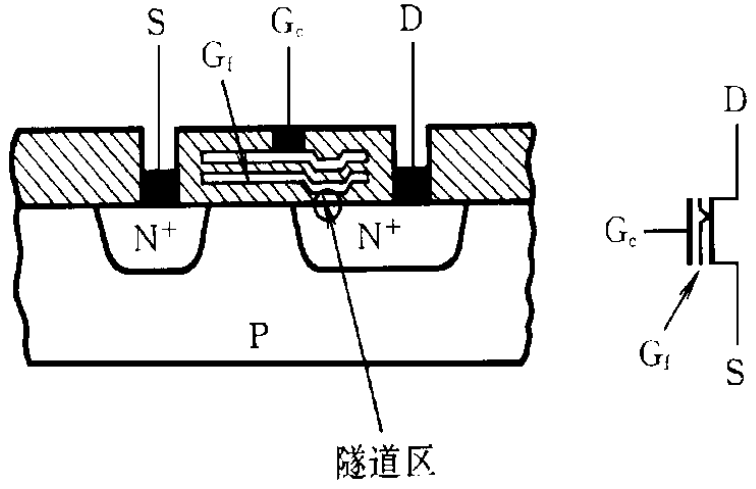
- 整体结构与掩膜ROM一样
  - 地址译码
  - 存储矩阵
  - 输入和编程控制
  - 输出缓冲三态控制
- 在存储矩阵的所有交叉点上全部制作了存储元件，即相当于在所有存储单元中都存入了“1”
- 按存储元件的实现方式不同
  - 熔丝型PROM存储单元
  - 用紫外线擦除的PROM（UVEPROM）
  - 用电信号可擦除的可编程ROM（E^2PROM）

可编程只读存储器(2)

- 可编程单元实现方法







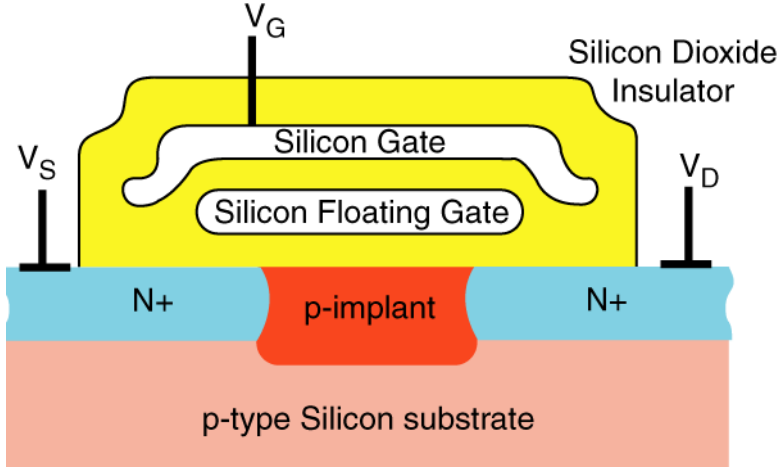
- 快速熔断丝接在发射极
- 当想写入0时，只要把相应的存储单元的熔断丝烧断即可
- 但只可编写一次

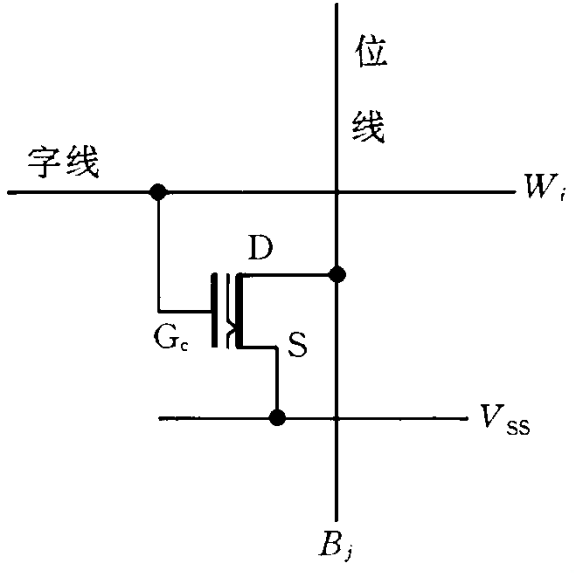
- 若G_f上充以负电荷，则G_c处正常逻辑高电平下不导通
- 若G_f上未充负电荷，则G_c处正常逻辑高电平下导通
- 需要紫外线擦除

- Flotox也是N沟道增强型MOS管；也有两个栅极，控制栅Gc和浮置栅G_f
- 隧道效应：隧道区的电场强度$> 10^7$V/cm时，便在D和G_f之间出现导电隧道，电子可以双向通过，形成电流
- 电可擦除

可编程只读存储器—Flash Memory

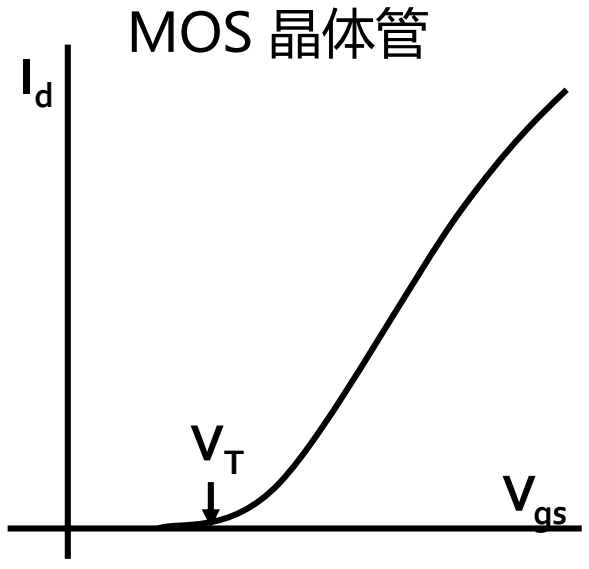
- 与EPROM中的SIMOS管极为相似
- 区别是浮置栅与衬底间氧化层的厚度在EPROM中30~40nm，快闪存储器中10~15nm。
- 浮栅与源区重叠的部分由源区的横向扩散形成，面积小，因而浮置栅-源区间的电容要比浮置栅-控制栅间的电容小得多。当控制栅和源极间加电压时，大部分电压都降在浮置栅与源极之间的电容上

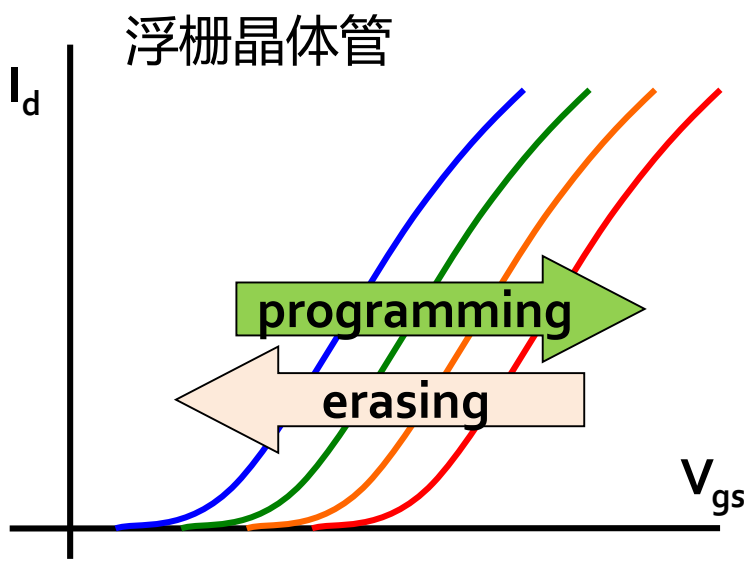




浮栅存储单元

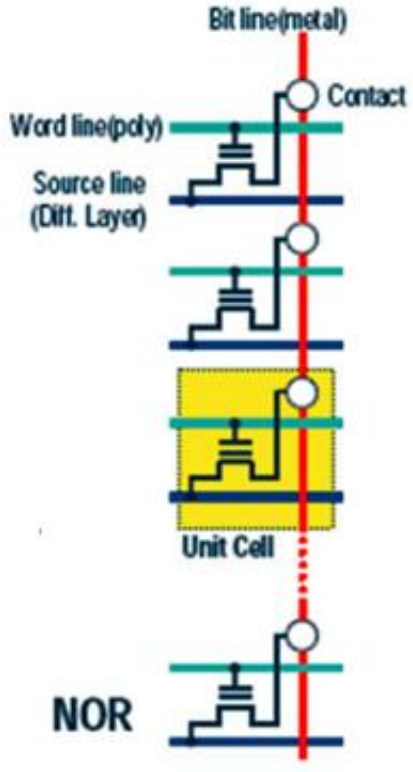
- MOS 管: 一个固定的开启电压
- Flash 存储单元: V_T 能够通过编程和擦除改变

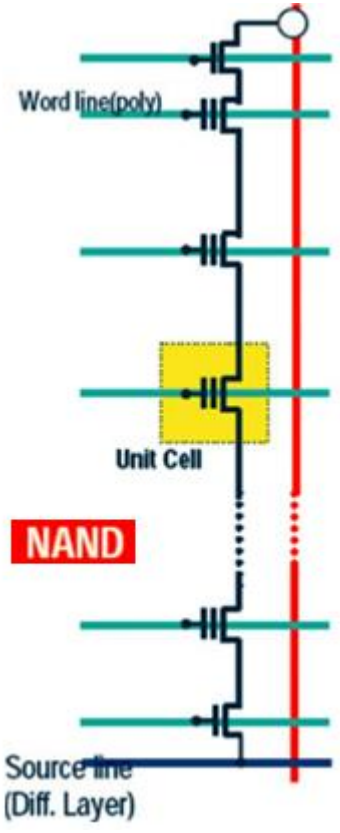




Flash 存储器特点

- NOR
  - 更长的寿命
  - 快速读时间~100ns
  - 慢速写时间~ 10 μs
  - 用与存储代码
- NAND
  - 存储单元小，集成度高
  - 读速度慢 ~ 1 μs
  - 写速度快 ~ 1 μs
  - 用于存储数据





各种PROM比较

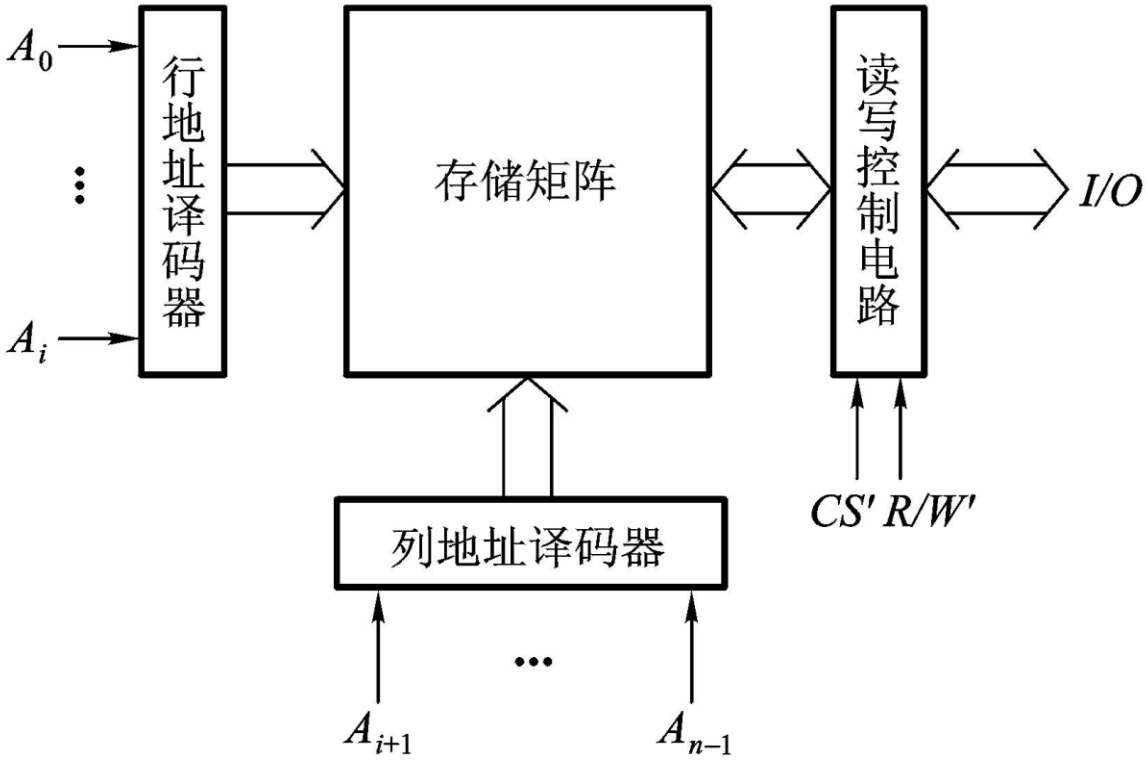
器件	EPROM	EEPROM	Flash EEPROM
通道浮栅尺寸	100 nm	10 nm	10 nm
编程原理	雪崩击穿	隧道效应	雪崩击穿
编程时间	微秒（μs）	毫秒（ms）	微秒（μs）
擦除	紫外线	隧道效应	隧道效应
擦除时间	数千秒	数毫秒/字	数毫秒/字
编程/擦除次数	100	10000	10000

半导体存储器

- 存储器概述
- 只读存储器（ROM）
- 随机存储器（RAM）
- 存储器容量的扩展
- 用存储器实现组合逻辑函数

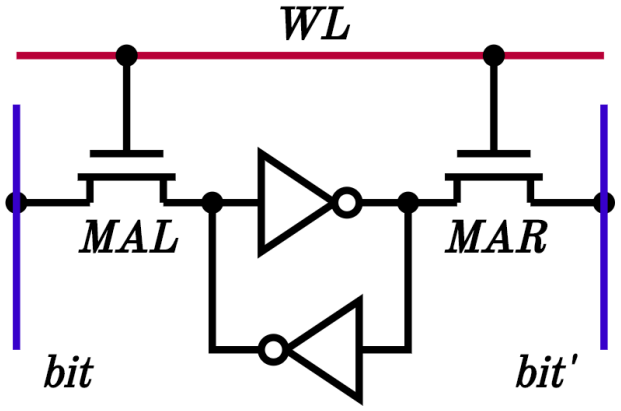
随机存储器（RAM）

- 也称随机读/写存储器，简称RAM
- 特点
  - 可以随时从任何一个指定地址读出数据
  - 可以随时将数据写入任何一个指定的存储单元
  - 优点：读、写方便，使用灵活
  - 缺点：数据易失性
- 分类
  - 静态随机存储器SRAM：每单元(bit)6个晶体管，同步，Cache，读写快~ns
  - 动态随机存储器DRAM：1晶体管/bit，需刷新，容量大，内存，读写~10ns
    - PC100,PC133, DDR, DDR2/3/4, 2.6~4Ghz

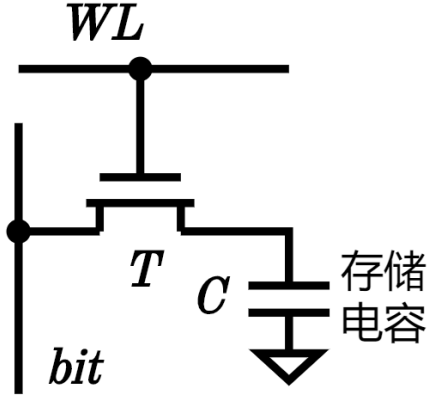


SRAM/DRAM基本概念

- SRAM：静态随机存储器
  - 静态、易失
  - 3个操作状态： hold, write, read
  - 基本存储单元有6个晶体管
    - 基本触发器作为存储部件
    - 访问控制晶体管 MAL和MAR
    - 字线WL控制访问，WL=0（Hold）；WL=1 (Read/Write)

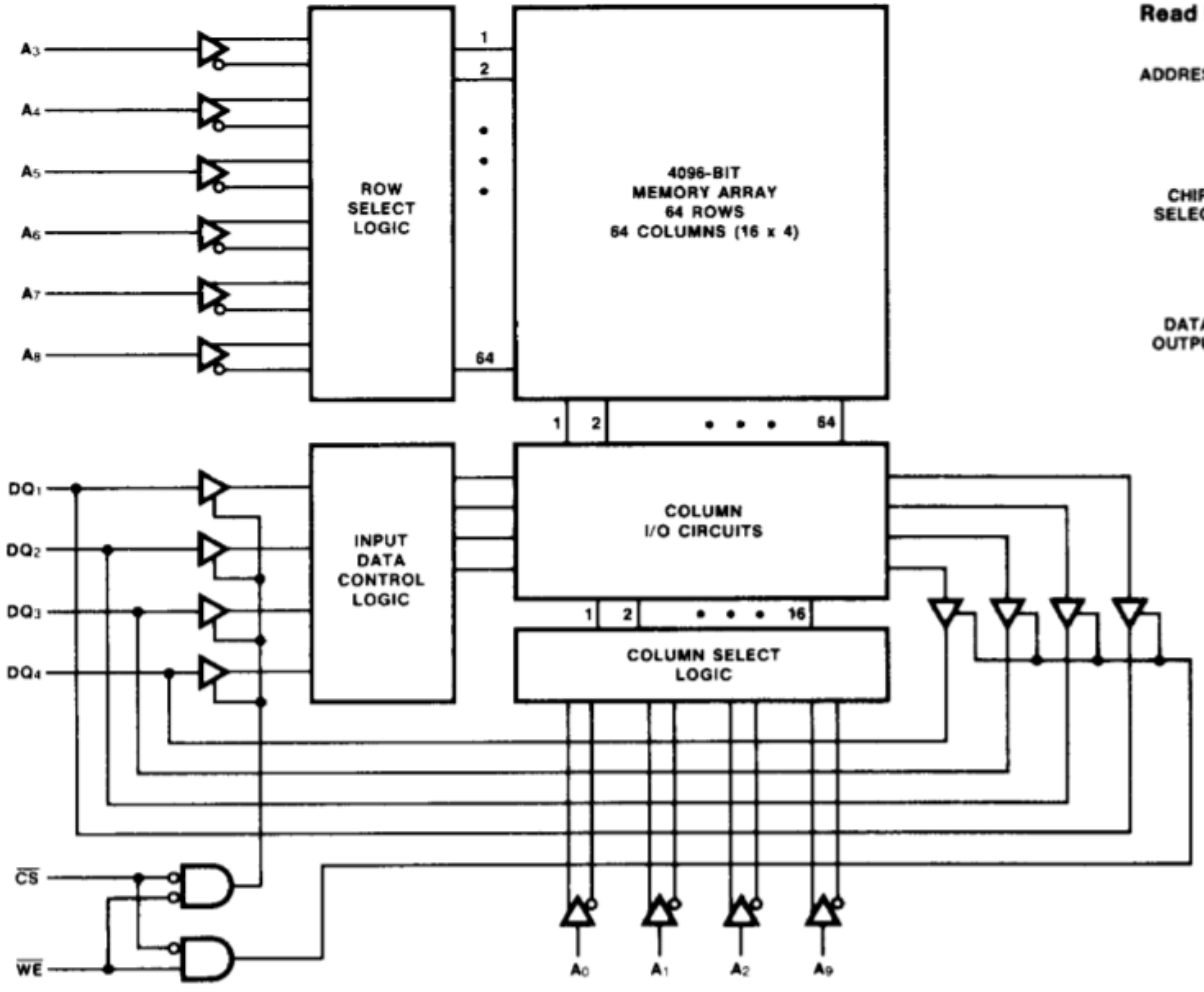


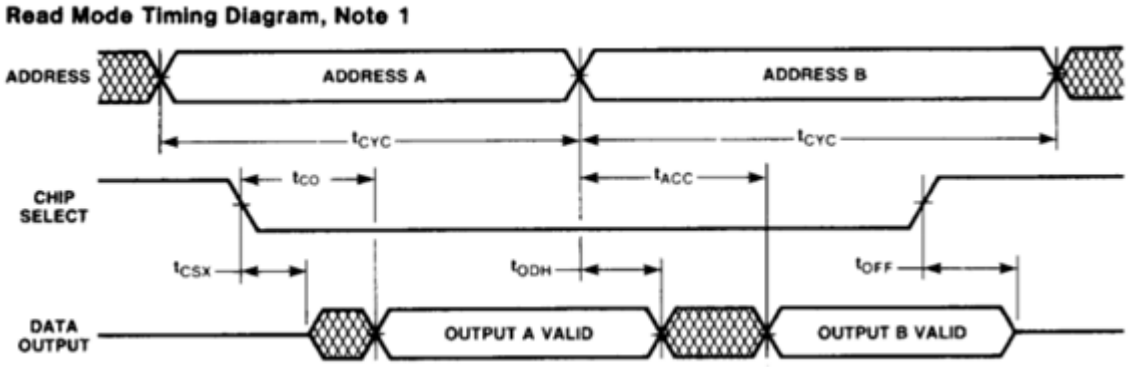
- DRAM: 动态随机访问存储器
  - 动态、易失
  - 单晶体管存储单元
    - 一个访问控制晶体管，一个存储数据的电容
    - 控制输入：字线 (WL)，数据I/O; 位线

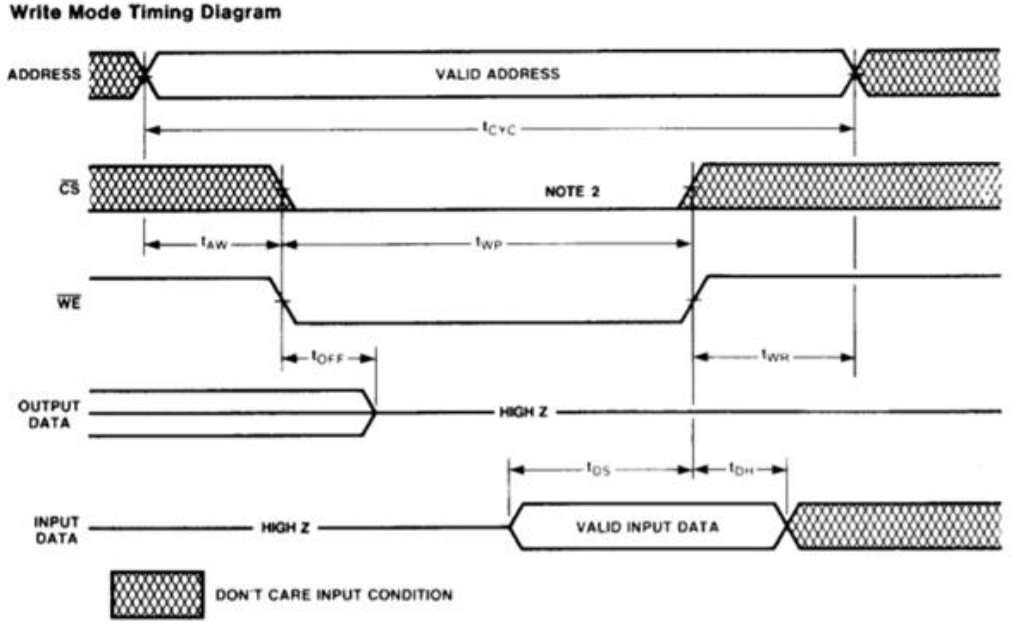


- DRAM 与 SRAM 比较
  - DRAM：每位的占用面积和成本更低，接口电路复杂，需要周期刷新
  - SRAM：成本高，集成度低，不需要刷新，访问速度快

SRAM结构和工作时序

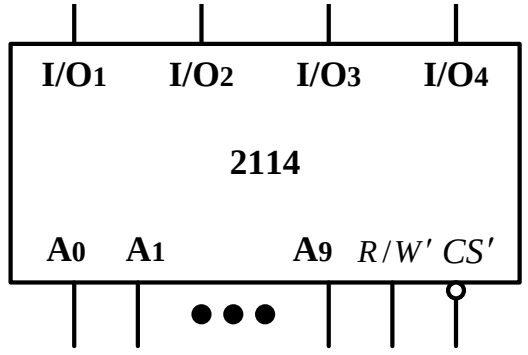


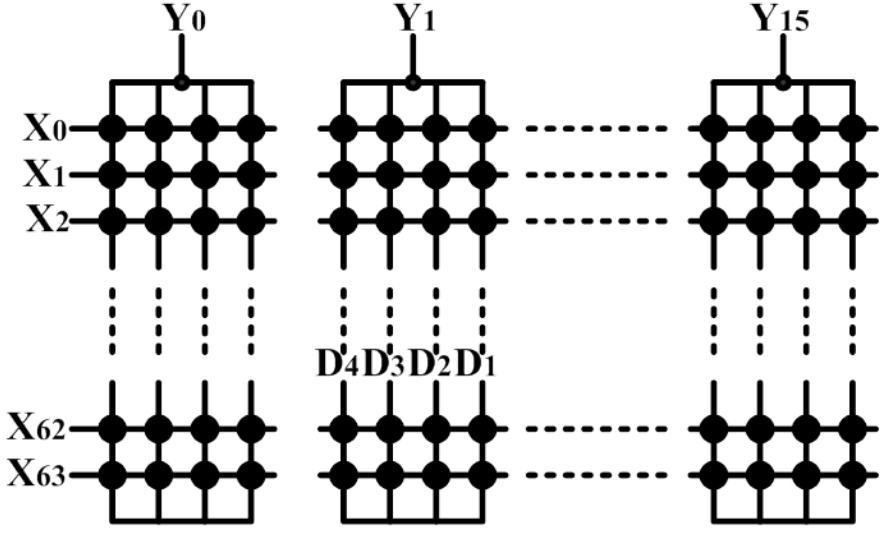




SRAM地址和控制信号

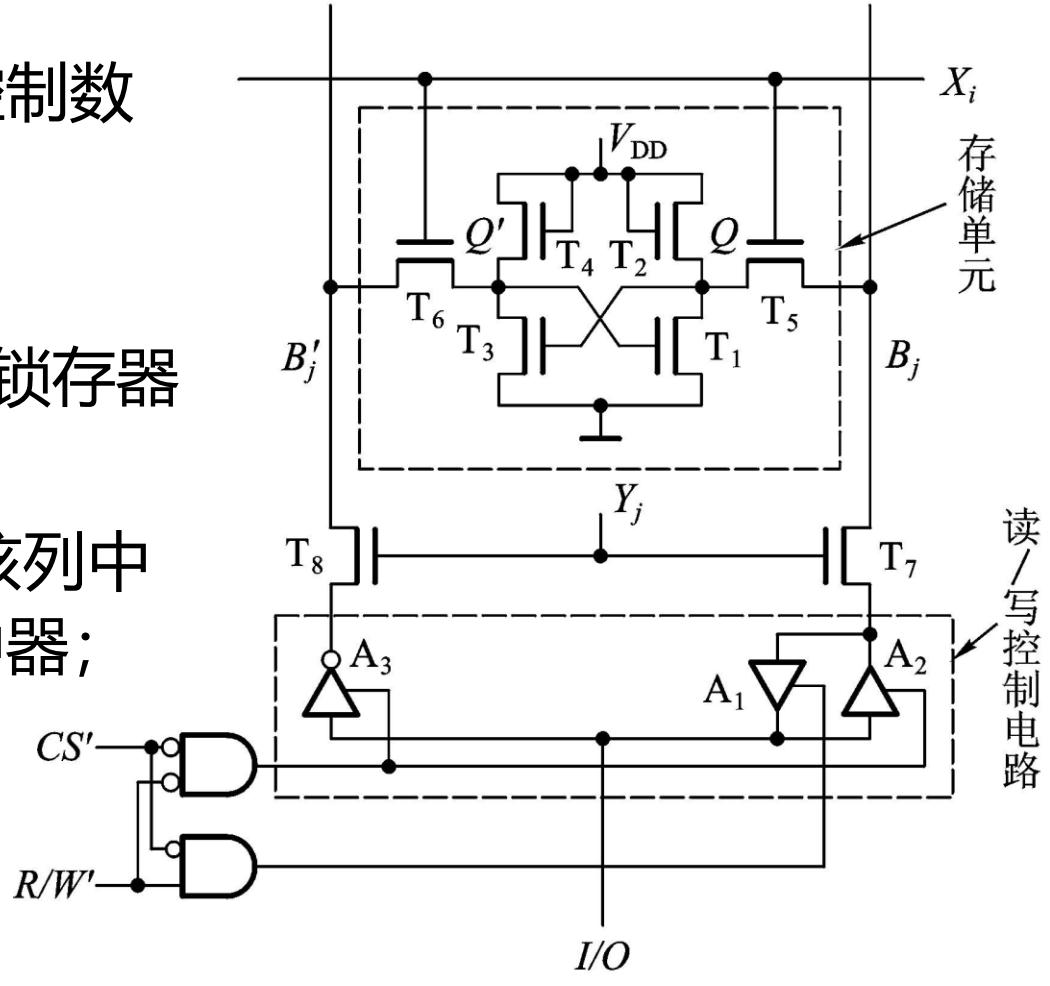
- 存储单元：64×64 = 4096个存储单元，排列成64行和64列的矩阵
- 每个存储单元都是由6个NMOS管组成
- 地址译码器：地址线$A_0 \sim A_9$
  - $A_8 \sim A_3$行地址译码器输入，输出$2^6 = 64$ 根行地址$X_0 \sim X_{63}$
  - $A_9\ A_2 \sim A_0$列地址输入，输出$2^4 = 16$ 根列地址$Y_0 \sim Y_{15}$
  - 如：$A_9\ A_2 \sim A_0 = 0001, A_8 \sim A_3 = 111110$时，则$Y_1 = 1, X_{62} = 1$，可对其交点$D_4 \sim D_1$进行读写操作
  - 每个字线包含4位，$I/O_0 \sim I/O_3$
- 控制信号
  - CS'片选信号，为低时选中该芯片，可读写，否则I/O三态
  - R/W'读写控制信号，高电平读操作，低电平写操作





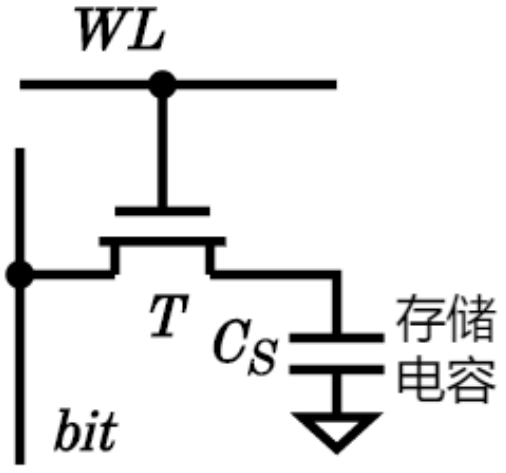
SRAM存储单元

- 靠锁存器的自保功能存储数据，门控管控制数据的输入和输出
- SR锁存器 T_1, T_2, T_3, T_4 ， 存储数据
- 行线门控管 T_5, T_6，行选中时，导通，SR锁存器输出连接到列线的门控管上
- 列线门控管T_7, T_8，列选中时，导通，把该列中已被行选中的锁存器输出连接到输出缓冲器；或者该列线被写入缓冲器所驱动
- CS' 与R/W'共同决定位数据处于输入状态，还是输出状态，通过三态门控制



动态随机存储器(DRAM)

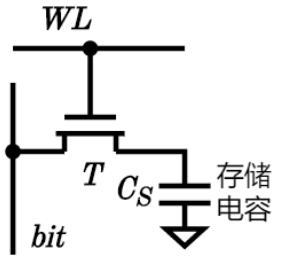
- MOS管栅极电容可以存储电荷
- 需辅以刷新必要的控制电
  - 单管存储单元
    - N沟道增强型MOS管T，存储电容C
    - 外围电路复杂，但能提高集成度

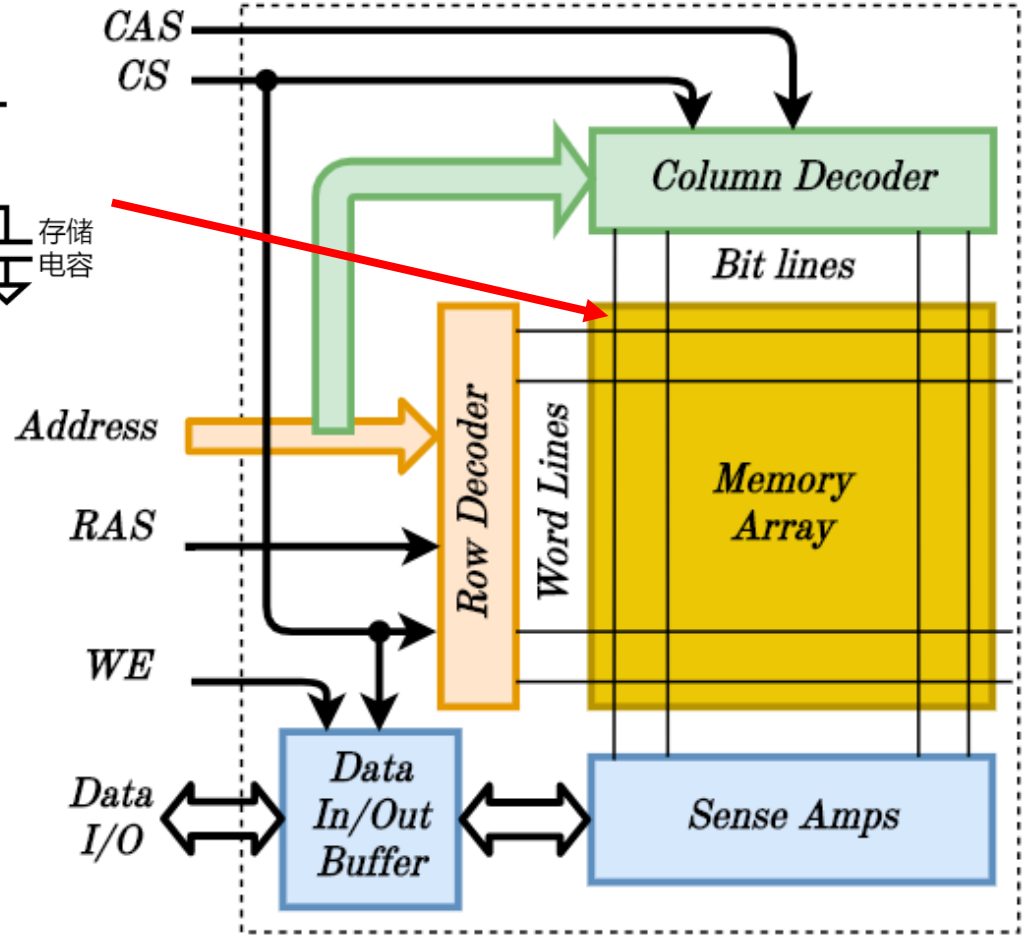


- 写
  - 字线给出高电平，使T导通，位线上的数据便经过T存入C_S.
- 读
  - 字线给出高电平，使T导通。 C_S经T向位线上的电容C_B提供电荷，使位线获得读出的信号电平，(位线读出的电压信号很小)
  - 破坏性读出，故需要设置灵敏的读出放大器
- 刷新
  - 按行读出再写回，需周期性操作，给电容充电，避免电容上的电荷消失

DRAM总体结构

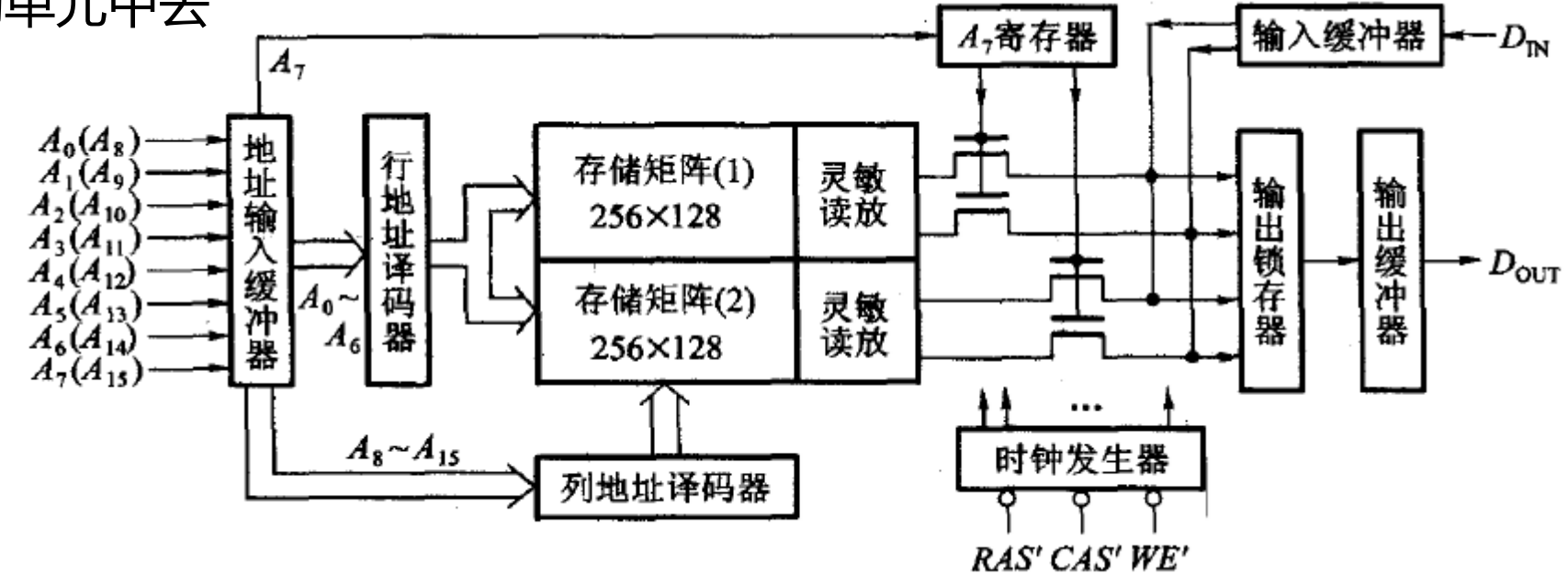
- 行地址、列地址分两次传送
  - Address是地址线
  - RAS信号锁定行地址
  - CAS信号锁定列地址
- Data I/O是双向数据总线
  - 写操作：作为数据输入线
  - 读操作：作为数据输出线
- WE是读写控制信号，区分总闲的读、或者写请求
- CS是片选信号，有效时选中芯片，可读写，无效时，芯片数据总闲高阻
- RAS、CAS、WE的时序组合，可以控制DRAM的多种工作模式





4164DRAM举例（1）

- 存储容量：65536 bits，Data I/O一位
  - 两个128行、256列的存储矩阵
  - RAS′ =0，输入地址代码的A_0~A_7，A_0~选中两块中的2行， A_7 选中2行中的一行
  - CAS′ =0，输入地址代码A_8~ A_{15}，选中256列中选中一列
  - 当WE′ =1时，进行读操作，输出三态缓冲器到达数据输出端D_{OUT}
  - 当WE′ =0时，进行写操作，加到数据输入端D_{IN}的数据经过输入缓冲器写入输入地址指定的单元中去



半导体存储器

- 存储器概述
- 只读存储器（ROM）
- 随机存储器（RAM）
- 存储器容量的扩展
- 用存储器实现组合逻辑函数

存储器容量扩展

- 当使用一片ROM或RAM器件不能满足对存储容量的需求时，则需要将若干片ROM或RAM组合起来，构成更大容量的存储器。

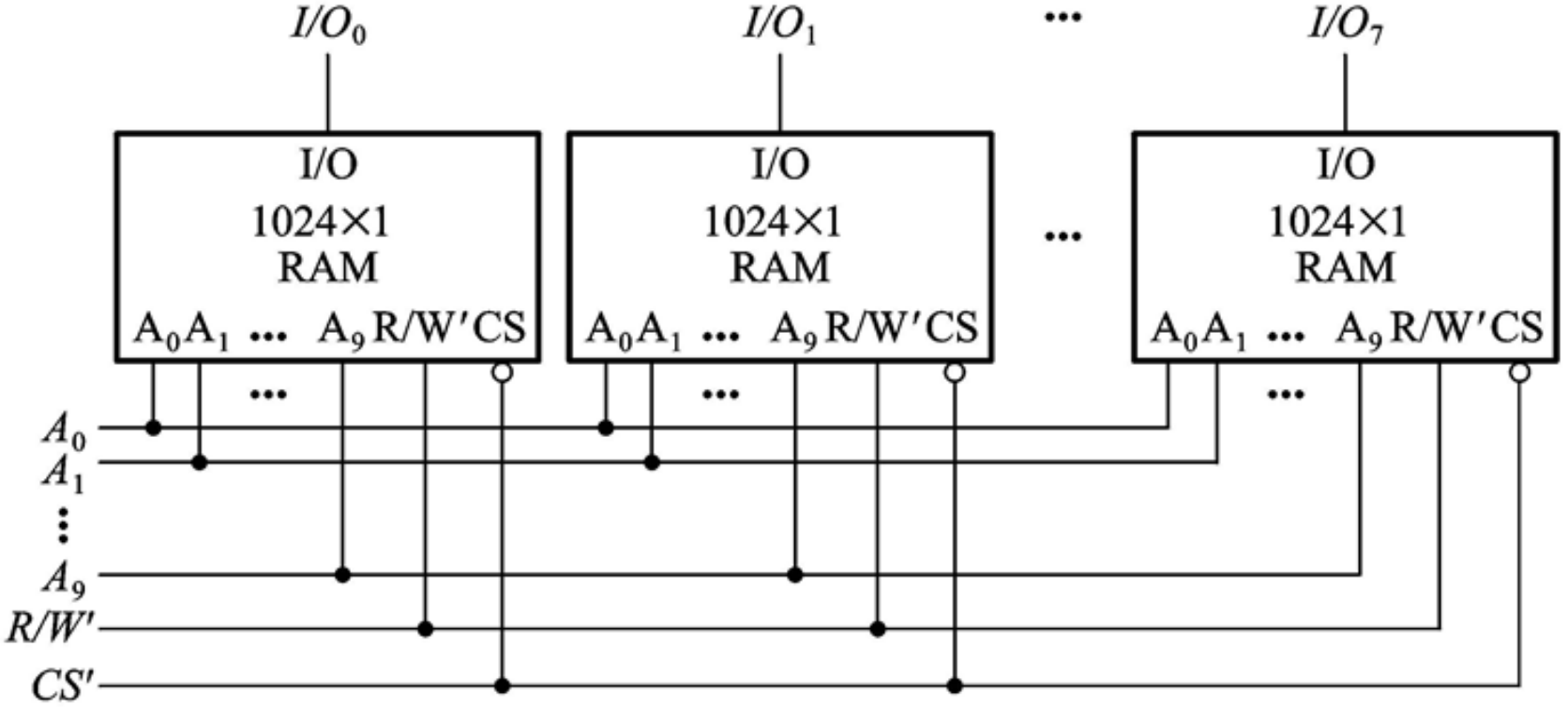
- 存储容量的扩展方式有两种

  – 位扩展方式：获得更宽的存储器字，8位，16位，32位，64位，72位（64+8）

  – 字扩展方式：获得更大存储器字容量（KB，MB，GB）

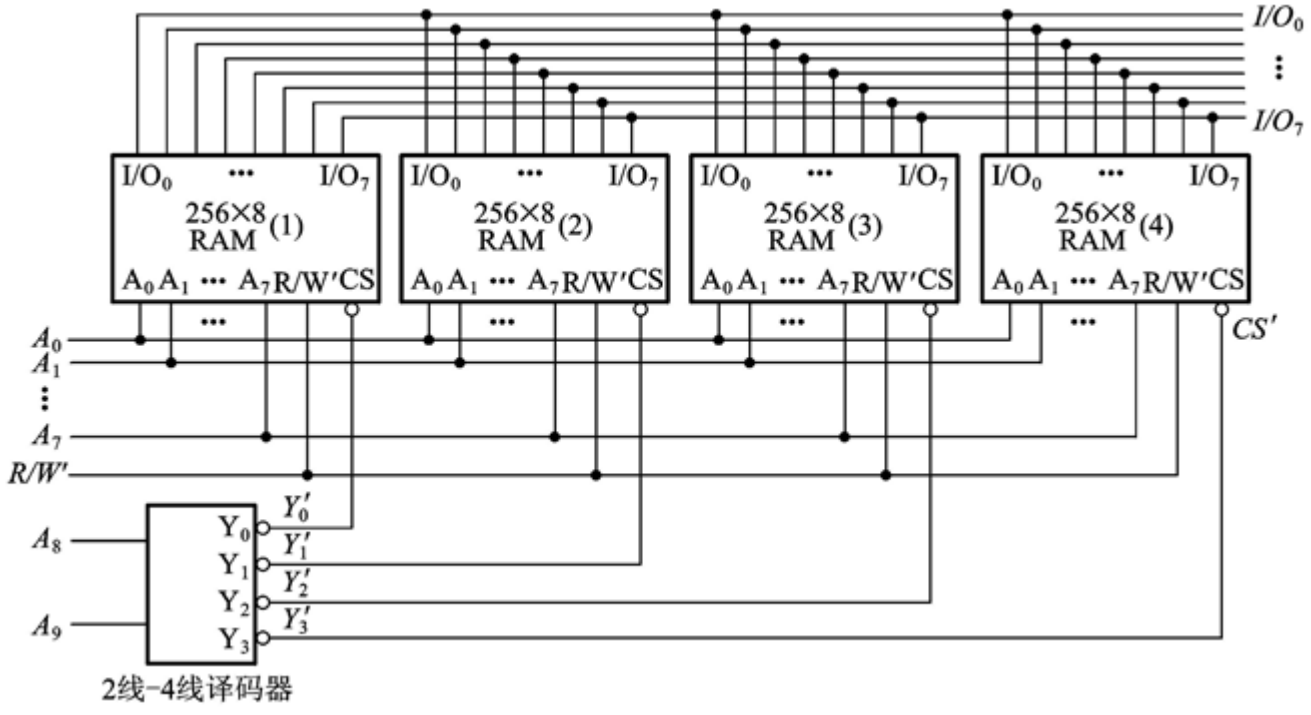
存储器位容量扩展

- 将所有地址线、R/W’ 、CS’ 分别并联
- 每一片I/O端作为整个RAM输入/输入端的一位
- 总的存储容量为每一片存储容量的8倍



存储器字容量扩展

- 例，用4片256 × 8 位RAM接成1024 × 8 位RAM



地址		片选输出				地址空间
A_9	A_8	CS_4	CS_3	CS_2	CS_1	
0	0	1	1	1	0	000~0FF
0	1	1	1	0	1	100~1FF
1	0	1	0	1	1	200~2FF
1	1	0	1	1	1	300~3FF

半导体存储器

- 存储器概述
- 只读存储器（ROM）
- 随机存储器（RAM）
- 存储器容量的扩展
- 用存储器实现组合逻辑函数

用存储器实现组合逻辑函数

- 以地址线为输入变量，则数据线即为一组关于地址变量的逻辑函数
- 例,输入地址A_0、A_1为输入逻辑变量，输出数据D_0~D_3为输出逻辑变量，则D_0~D_3即为一组A_0、A_1的组合逻辑函数，并得到真值表和逻辑函数如下：

$$\begin{cases} D_3 = m_1 + m_3 \\ \quad = A_1' A_0 + A_1 A_0 \\ D_2 = m_0 + m_2 + m_3 \\ \quad = A_1' A_0' + A_1 A_0' + A_1 A_0 \\ D_1 = m_1 + m_3 = A_1' A_0 + A_1 A_0 \\ D_0 = m_0 + m_1 = A_1' A_0' + A_1' A_0 \end{cases}$$

最小项	A_1	A_0	D_3	D_2	D_1	D_0
m_0	0	0	0	1	0	1
m_1	0	1	1	0	1	1
m_2	1	0	0	1	0	0
m_3	1	1	1	1	1	0

- 任何形式的组合逻辑函数均能通过向ROM中写入相应的数据来实现
- 具有n位输入地址,m位数据输出的ROM可以获得一组(最多m个)任何形式的n变量逻辑函数，只要根据函数的形式向ROM中写入相应的数据即可
- 该原理也适用于RAM

用存储器实现七段显示译码器

- 采用4位地址，8位数据的ROM来实现
- $A_3A_2A_1A_0$做地址输入端BCD码的$DCBA$，得到了所需的4-16线译码器
- 以数据输出端$D_6 \sim D_0$作为$a \sim g$的输出端
- 用EPROM实现译码器，只需把表中右侧的数据写入相应的地址单元即可

数字	输入				输出							16进制数
	A_3	A_2	A_1	A_0	Y_a	Y_b	Y_c	Y_d	Y_e	Y_f	Y_g	
0	0	0	0	0	1	1	1	1	1	1	0	7E
1	0	0	0	1	0	1	1	0	0	0	0	30
2	0	0	1	0	1	1	0	1	1	0	1	6D
3	0	0	1	1	1	1	1	1	0	0	1	79
4	0	1	0	0	0	1	1	0	0	1	1	33
5	0	1	0	1	1	0	1	1	0	1	1	5B
6	0	1	1	0	0	0	1	1	1	1	1	1F
7	0	1	1	1	1	1	1	0	0	0	0	70
8	1	0	0	0	1	1	1	1	1	1	1	7F
9	1	0	0	1	1	1	1	0	0	1	1	73
10	1	0	1	0	0	0	0	1	1	0	1	0D
11	1	0	1	1	0	0	1	1	0	0	1	19
12	1	1	0	0	0	1	0	0	0	1	1	23
13	1	1	0	1	1	0	1	1	0	1	1	5B
14	1	1	1	0	0	0	0	1	1	1	1	0F
15	1	1	1	1	0	0	0	0	0	0	0	00

用ROM实现组合逻辑函数 (1)

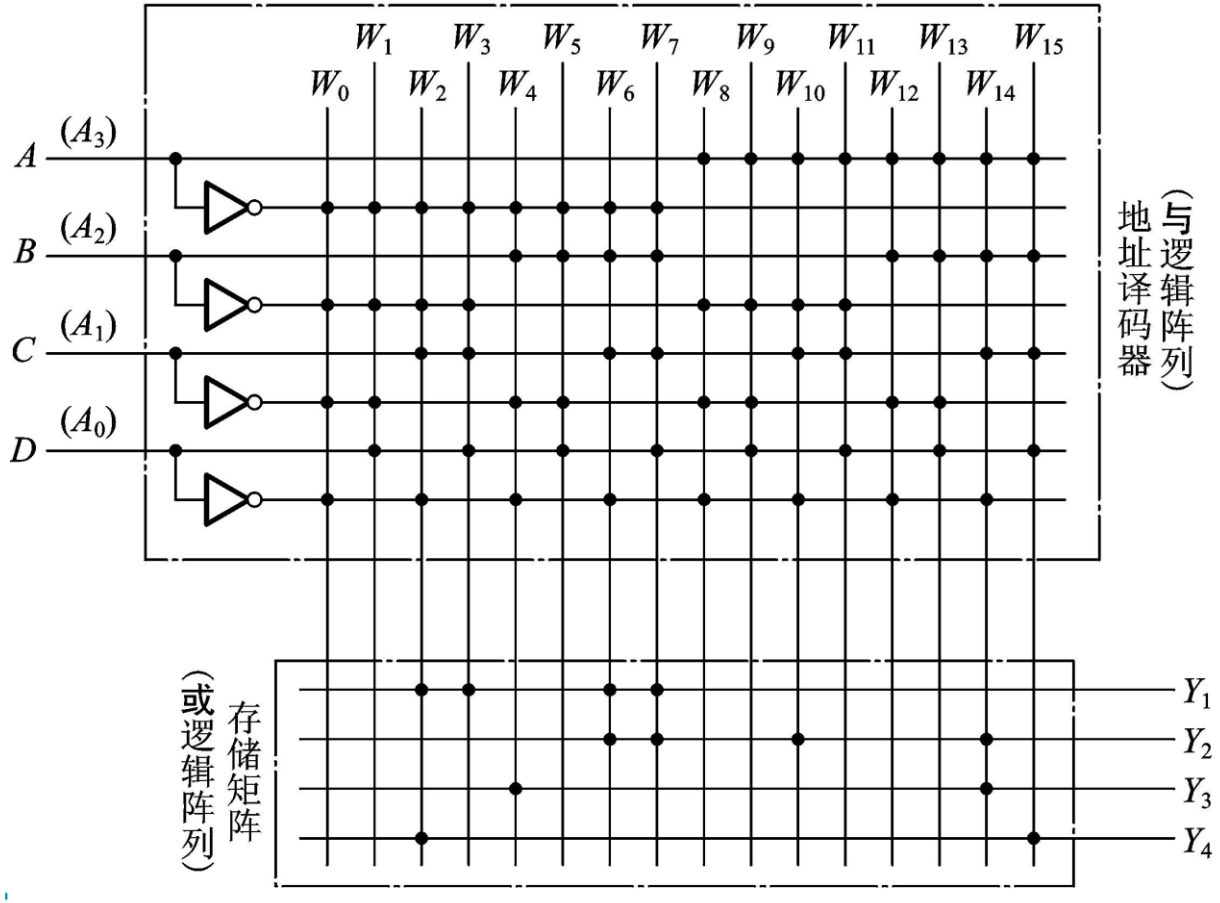
- 试用ROM产生下列组合逻辑函数
$$\begin{cases} Y_1 = A'BC + A'B'C \\ Y_2 = AB'CD' + BCD' + A'BCD \\ Y_3 = ABCD' + A'BC'D' \\ Y_4 = A'B'CD' + ABCD \end{cases}$$

- 将所给的逻辑函数展成最小项之和的形式
$$\begin{cases} Y_1 = \sum m(2,3,6,7) \\ Y_2 = \sum m(6,7,10,14) \\ Y_3 = \sum m(4,14) \\ Y_4 = \sum m(2,15) \end{cases}$$

- 要实现的是4个逻辑函数，且逻辑函数为4变量的，所以需要4位地址输入和4位数据输出，故选16×4的ROM实现

用ROM实现组合逻辑函数 (2)

- 按右表把数据写入ROM就可实现前述逻辑函数，下图是存储矩阵连接关系

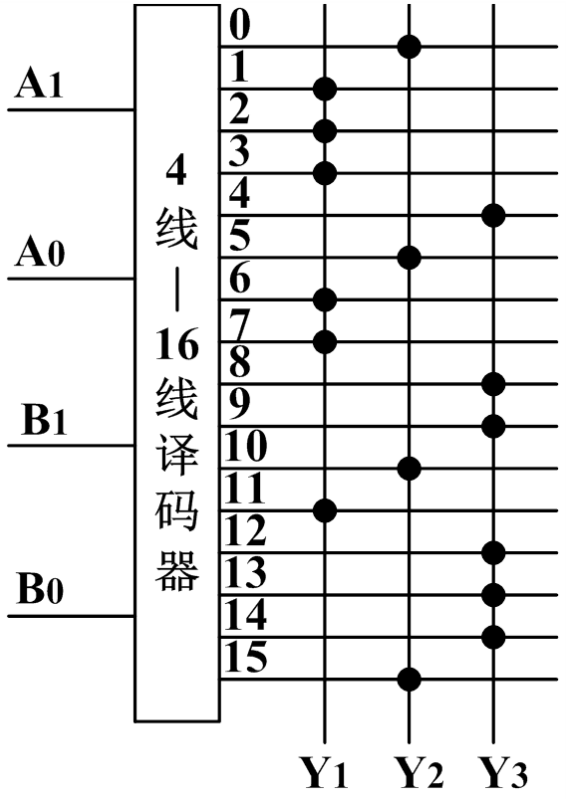


数字	输入				输出				16进制数
	A	B	C	D	Y_4	Y_3	Y_2	Y_1	
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0
2	0	0	1	0	1	0	0	1	9
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	1	0	0	4
5	0	1	0	1	0	0	0	0	0
6	0	1	1	0	0	0	1	1	3
7	0	1	1	1	0	0	1	1	3
8	1	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0
10	1	0	1	0	0	0	1	0	2
11	1	0	1	1	0	0	0	0	0
12	1	1	0	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0
14	1	1	1	0	0	1	1	0	6
15	1	1	1	1	1	0	0	0	8

用ROM实现2位二进制数比较器

- 设这两个2位数分别为
  $A = A_1A_0, B = B_1B_0$
  - 当$A < B$时，$Y_1 = 1$
  - 当$A = B$时，$Y_2 = 1$
  - 当$A > B$时，$Y_3 = 1$

- 以A_1A_0, B_1B_0作为ROM地址，$Y_3Y_2Y_1$作为ROM内数据的取值，列出真值表

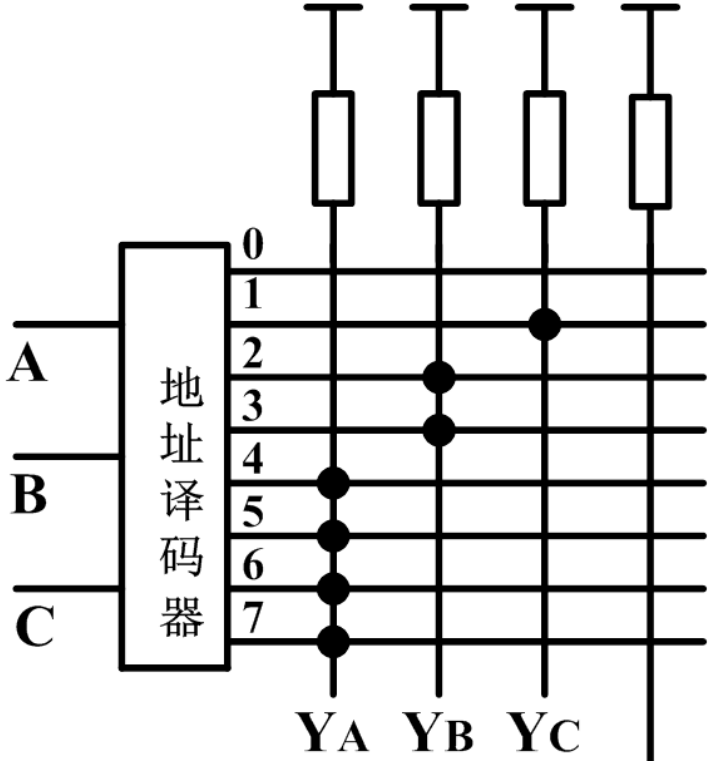


输入							16进制数
A_1	A_0	B_1	B_0	Y_3	Y_2	Y_1	
0	0	0	0	0	1	0	2
0	0	0	1	0	0	1	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	1
0	1	0	0	1	0	0	4
0	1	0	1	0	1	0	2
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	1
1	0	0	0	1	0	0	4
1	0	0	1	1	0	0	4
1	0	1	0	0	1	0	2
1	0	1	1	0	0	1	1
1	1	0	0	1	0	0	4
1	1	0	1	1	0	0	4
1	1	1	0	1	0	0	4
1	1	1	1	0	1	0	2

用存储器实现排队电路

- 用8×4位ROM实现一个排队组合电路，电路的功能是输入信号A、B、C通过排队电路后分别由Y_A, Y_B, Y_C 输出。在同一时刻只能有一个信号通过，如果同时有2个以上信号通过时，则按A、B、C的优先顺序通过

A	B	C	Y_A	Y_B	Y_C
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0



问题和建议?