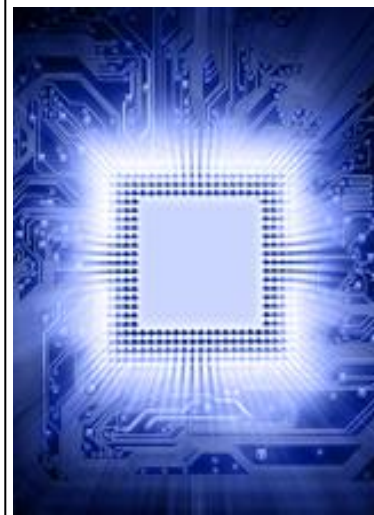


第9章 存储器、CPLD和FPGA

武港山

南京大学人工智能学院

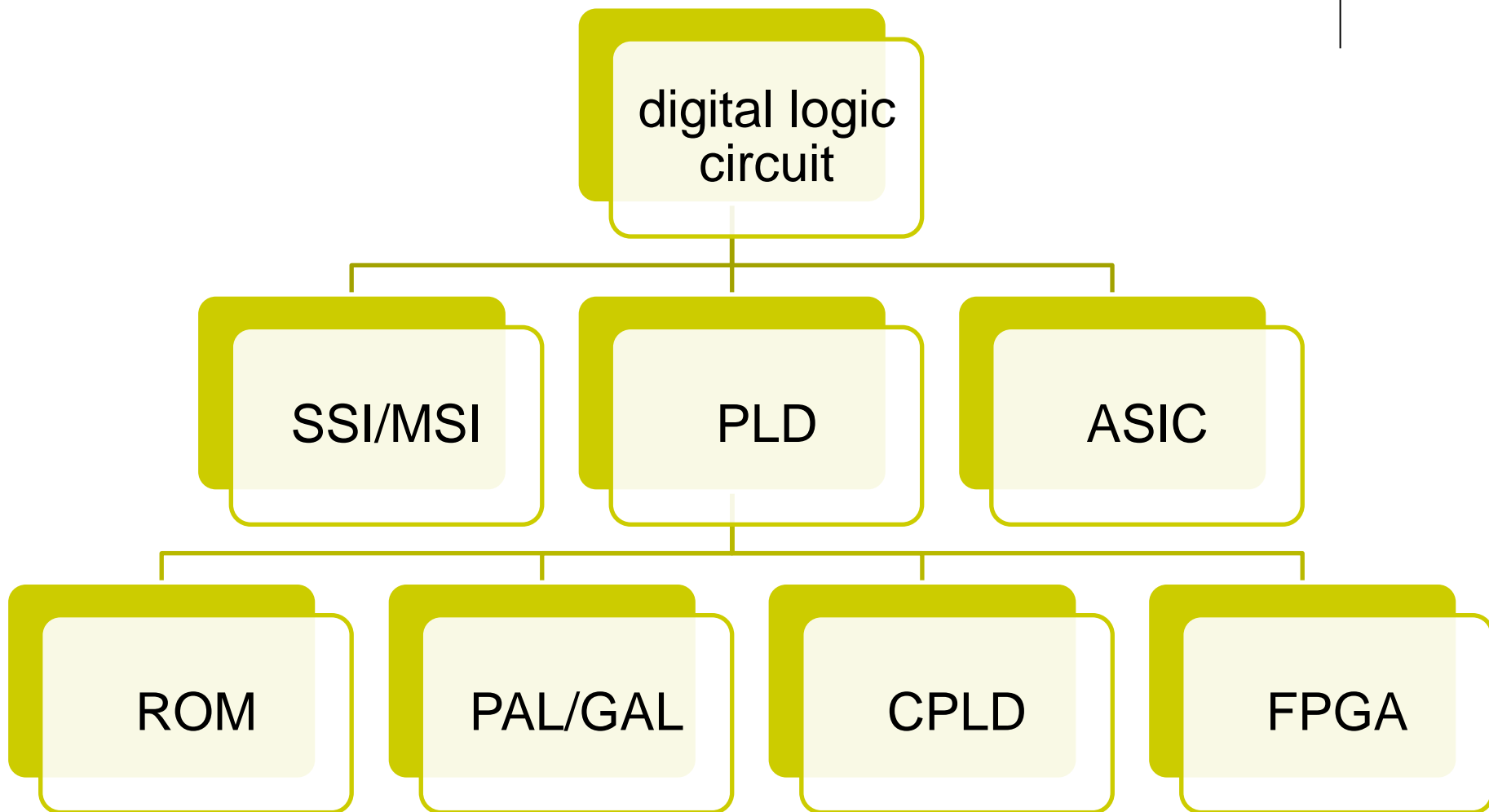




- ROM
- RAM
- CPLD
- FPGA



数字逻辑电路分类





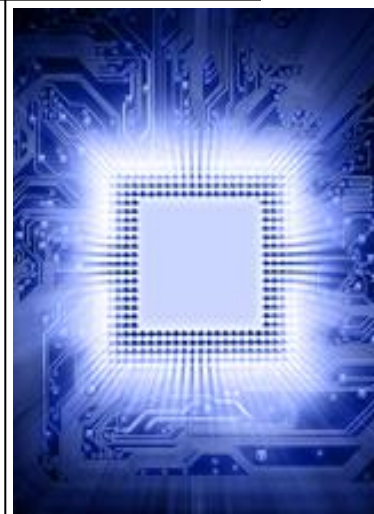
- 技术特征

- 分类：ROM、PAL/GAL、CPLD、FPGA
- 线宽：0.5um → .35um → .25um → .18um → .13um → 90nm → 65nm → 45/32nm
- 门数：千门 → 万门 → 十万门 → 百万门 → 千万门
- 内压：5V → 3.3V → 2.5 → 1.8 → 1.5 → 1.2 → 1.1/0.9V
- 金属线：4层 → 5层 → 6层 → 9层

1 半导体存储器

ROM

RAM





存储器



半导体存储器是一种能**存储**大量**二值数字信息**的**大规模集成电路**，是现代数字系统特别是计算机中的重要组成部分。半导体存储器主要用于电子计算机和某些数字系统，存放程序、数据、资料等。

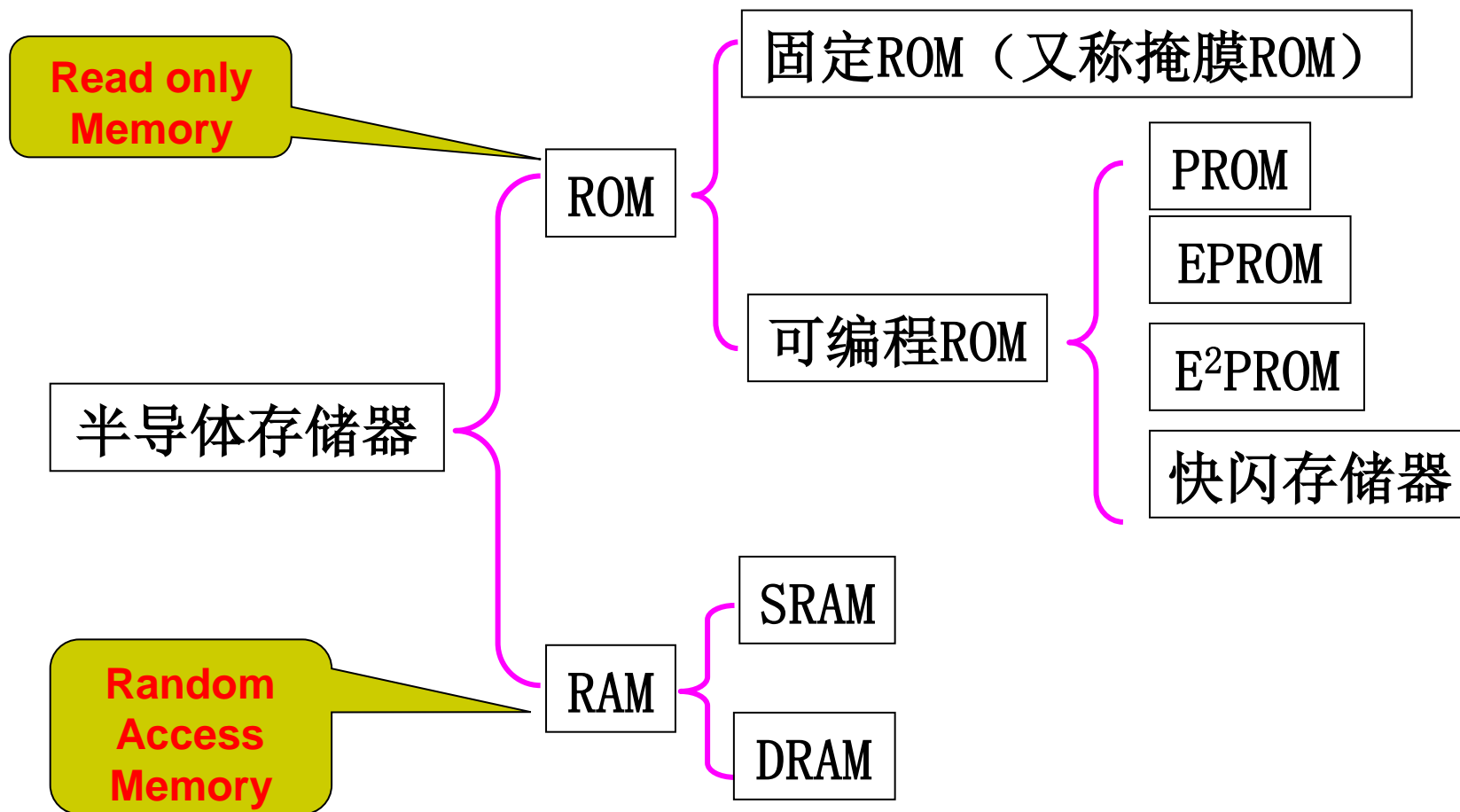
Address	Data
00000000	
00000001	
00000002	
.	
.	
.	
.	
.	
.	
.	
.	
.	
FFFFFFFFD	
FFFFFFFE	
FFFFFFF	



半导体存储器种类



按存取方式来分：





半导体存储器种类



对存储器的**操作**通常分为两类：

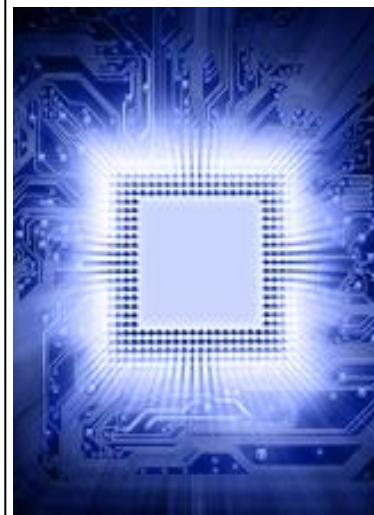
- 写**——即把信息存入存储器的过程。
- 读**——即从存储器中取出信息的过程。

两个重要技术指标：

- 存储容量**——存储器能存放二值信息的多少。单位是**位**或**比特**（bit）。 $1K=2^{10}=1024$ ， $1M=2^{10}K=2^{20}$ 。
- 存储时间**——存储器读出（或写入）数据的时间。一般用读（或写）周期来表示。

只读存储器

ROM

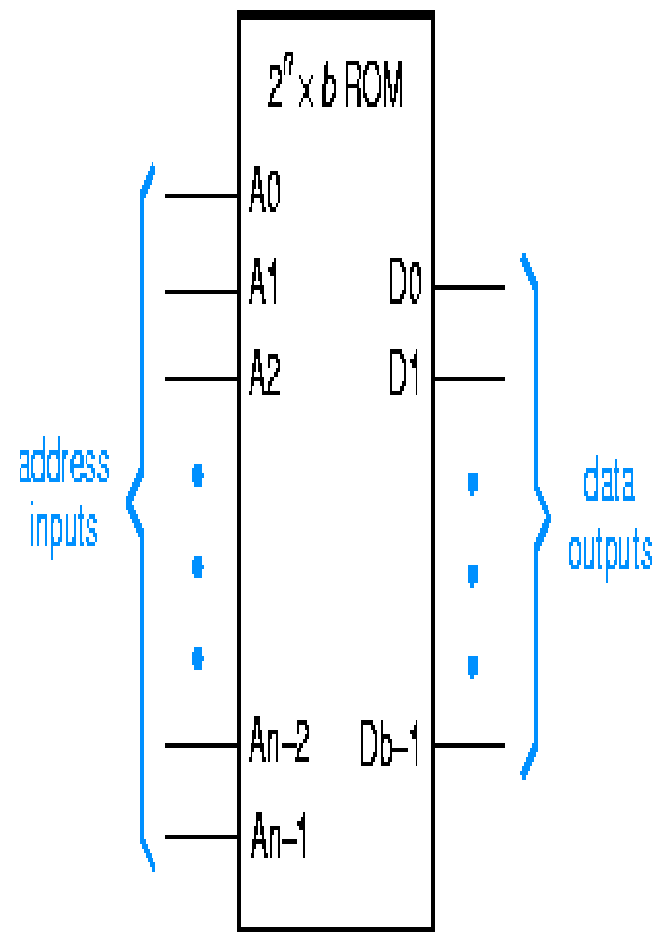




只读存储器



- 只读存储器简称ROM (read-only memory): 是一种具有 n 个输入 b 个输出的**组合逻辑电路**。
 - 包含地址输入（地址译码）、数据输出（输出缓冲）和存储矩阵三部分。
 - 存储矩阵储存了一个 n 输入 b 输出组合逻辑功能的真值表。
 - ROM的数据输出总是等于真值表中有地址输入所选择的那一行的输出位。
- 与真正存储器的区别: **非易失性存储器**(non volatile memory)。即使电源断电, ROM中存储的数据不会丢失。

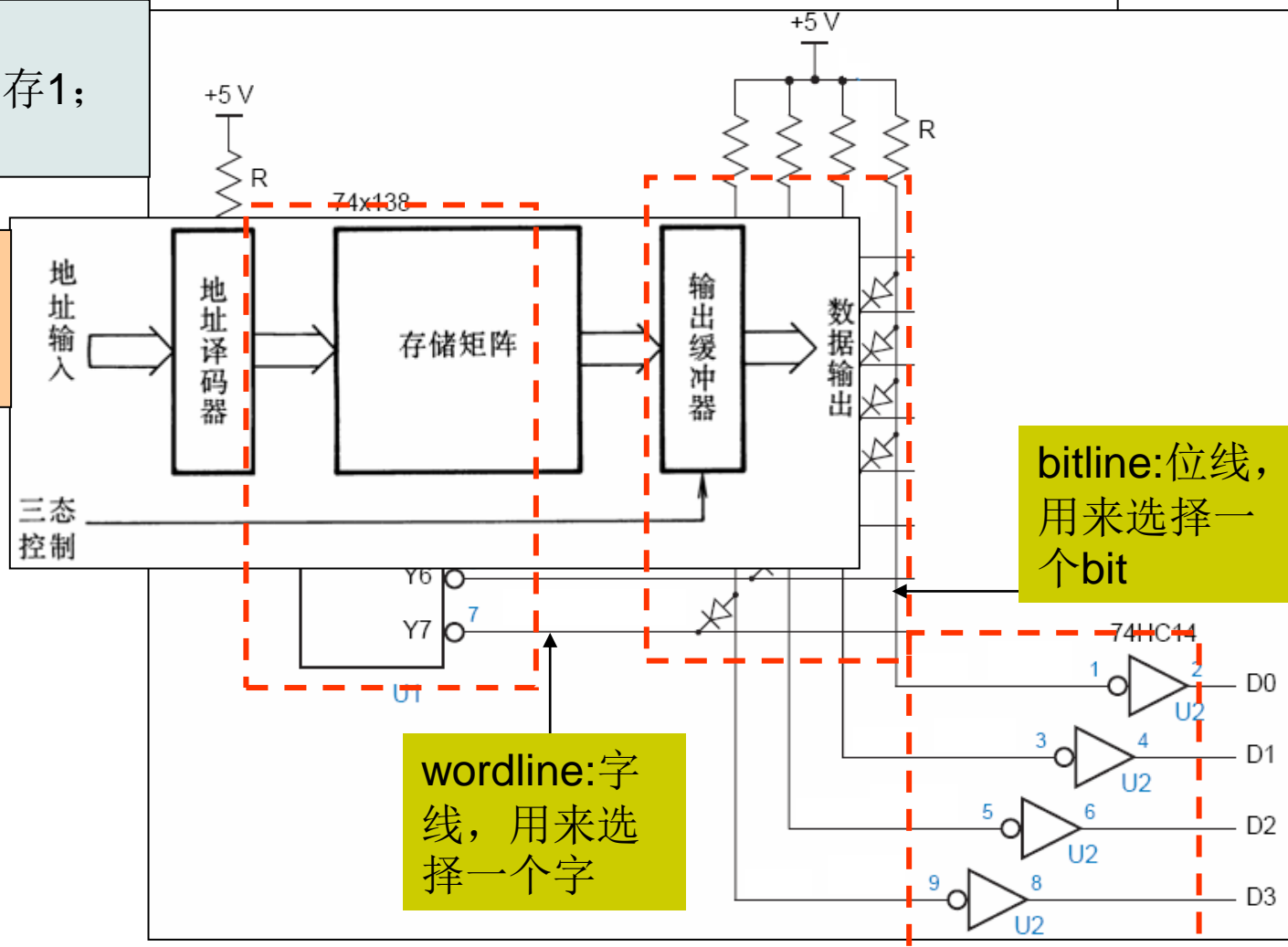


ROM结构



在存储矩阵中，
有二极管的点相当于存1；
否则是存0。

该存储器的容量
= 字线数 × 位线数
= 8个4bit数



bitline: 位线，
用来选择一个bit

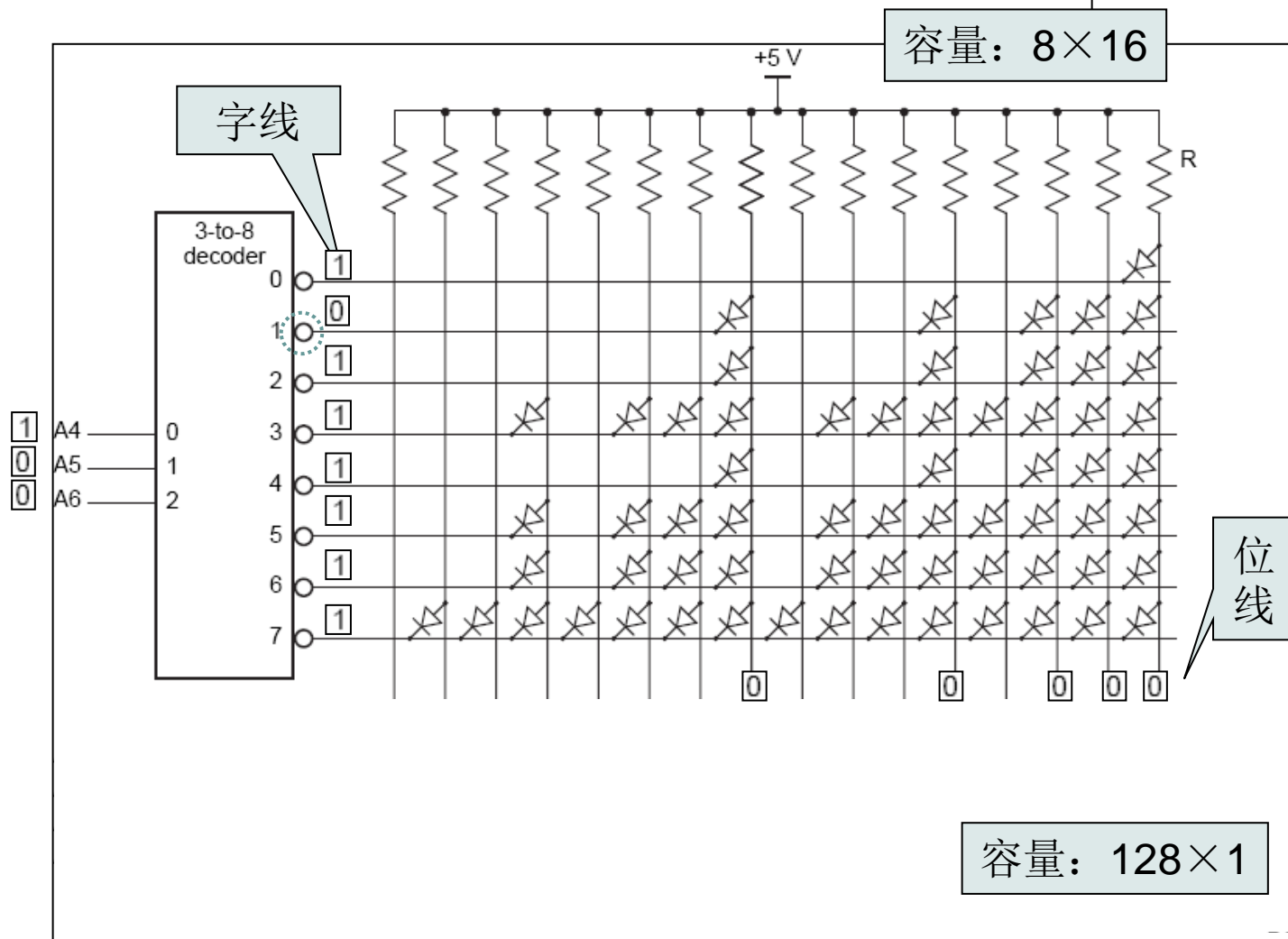
wordline: 字
线，用来选
选择一个字



使用二维译码的ROM

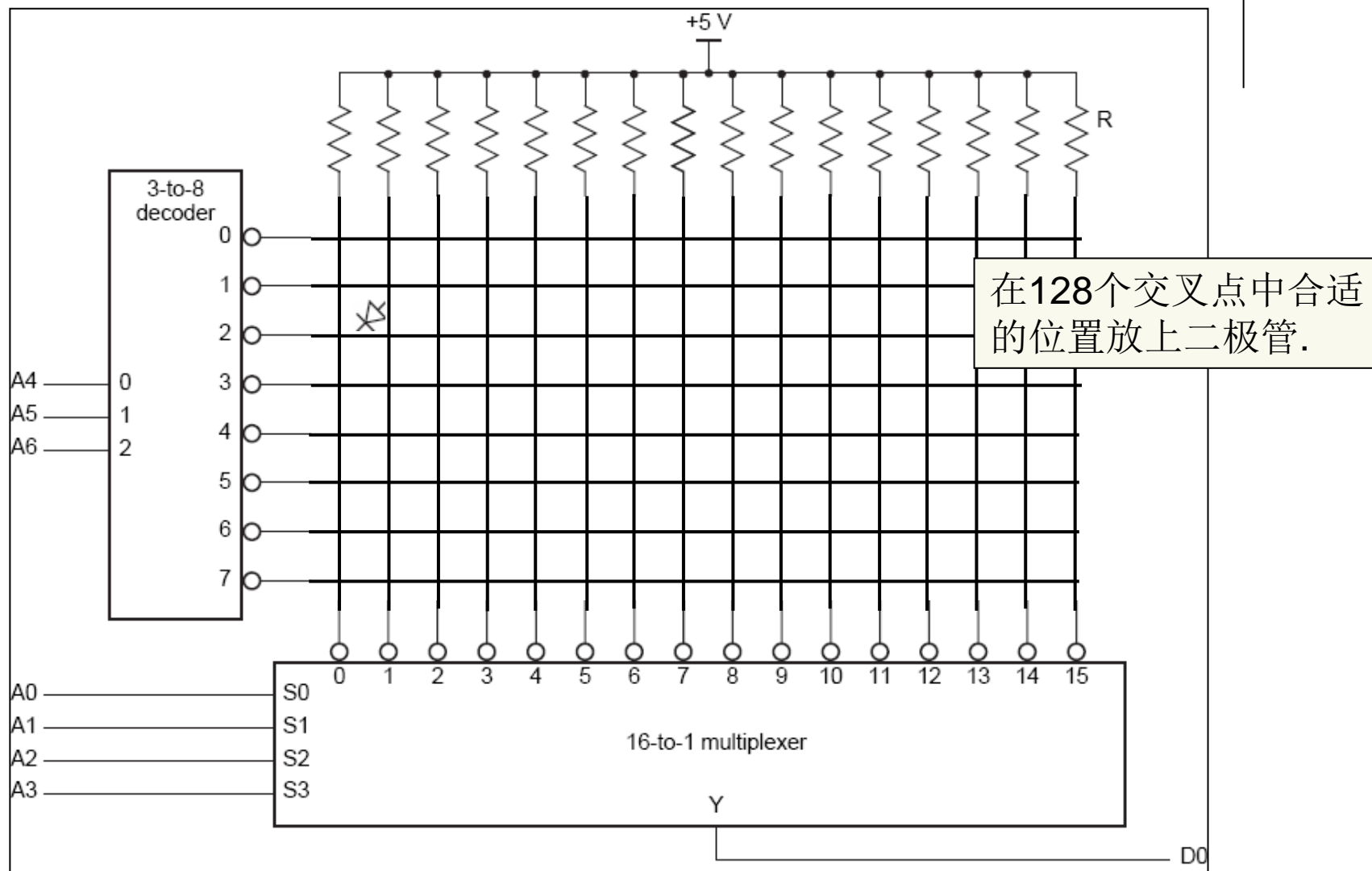
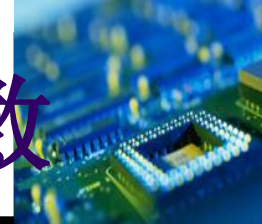


- 减少译码器的大小到地址数目的平方根数量级
- ROM单元排列成一个矩形，使用译码器和多路选择器
- 高阶地址决定行数据，低价地址决定位数据





使用二维译码ROM实现逻辑函数





3 ROM的应用



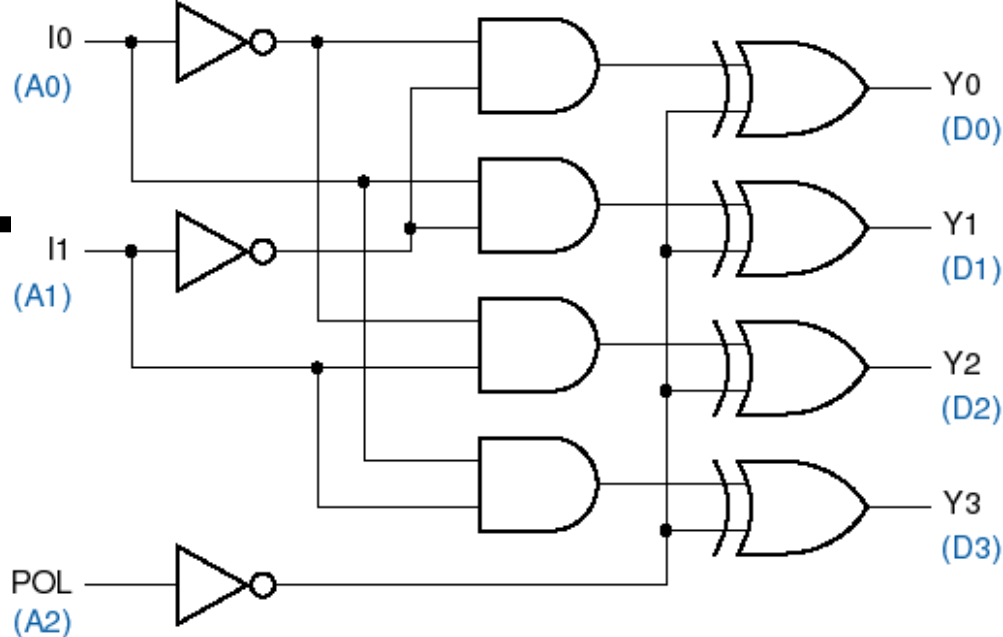
- **ROM = 最小项译码器+ 可编程或矩阵**
- **ROM = 存真值表的存储器**
- 通用可编程逻辑器件
- 组合逻辑设计
 - 由于**ROM**的地址译码器输出是全部输入变量的**最小项**，每一位数据的输出是这些最小项之**和**，因此任何形式的组合逻辑函数均能通过向**ROM**写入数据来实现。

2-4译码器实例

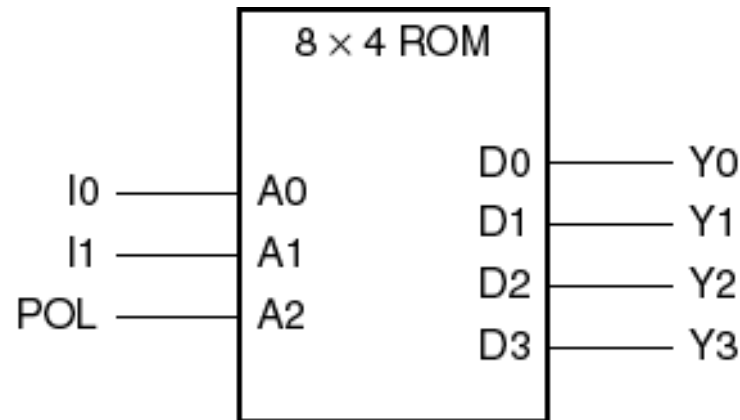
- 一个3输入4输出的组合逻辑真值表

Inputs			Outputs			
A2	A1	A0	D3	D2	D1	D0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

- 可看成一个具有极性控制的2-4译码器。



门电路实现

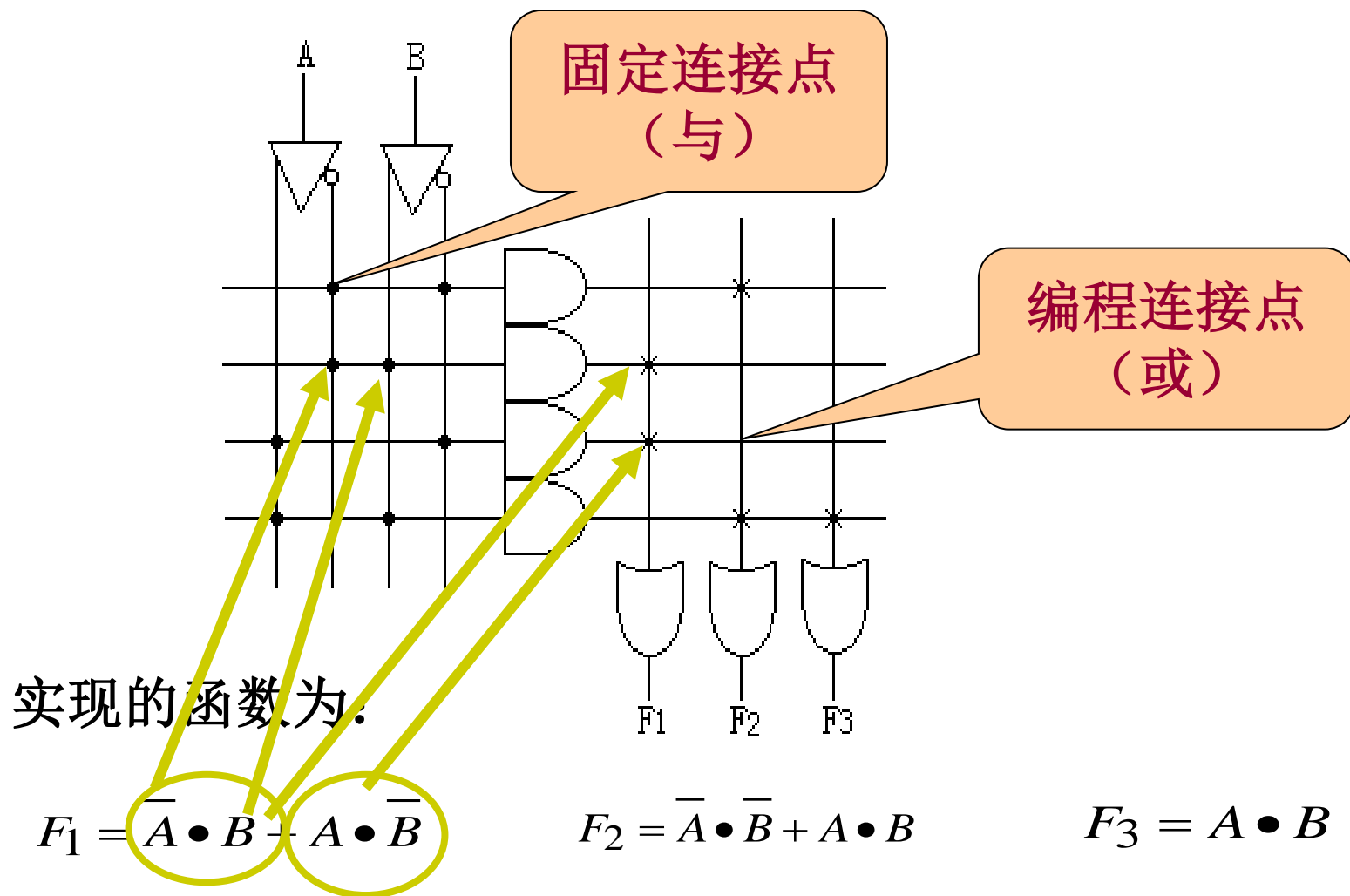


ROM实现

地址0~7存储E、D、B、7、1、2、4、8。



用PROM实现组合逻辑电路功能





3 ROM的应用



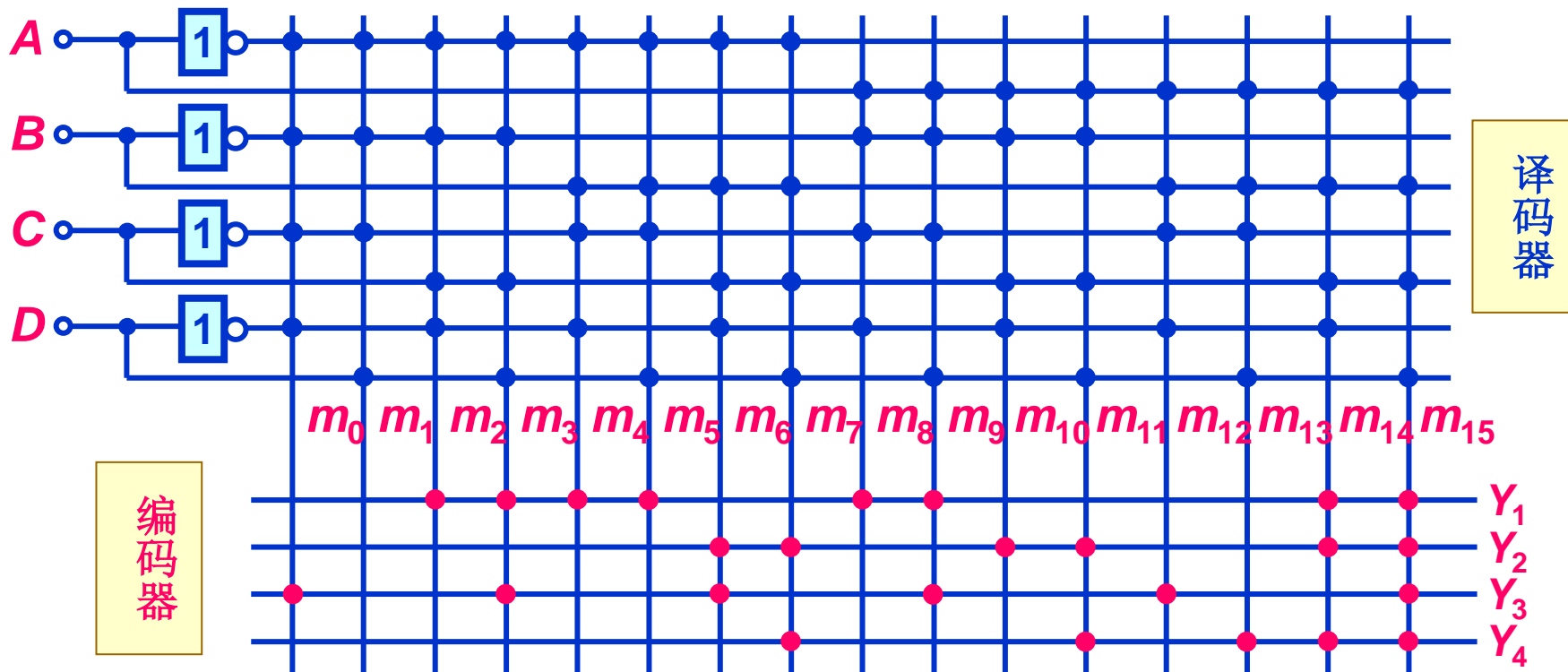
[例] 用 ROM 实现以下逻辑函数

$$Y_1 = \sum m(2, 3, 4, 5, 8, 9, 14, 15)$$

$$Y_2 = \sum m(6, 7, 10, 11, 14, 15)$$

$$Y_3 = \sum m(0, 3, 6, 9, 12, 15)$$

$$Y_4 = \sum m(7, 11, 13, 14, 15)$$





3 ROM的应用



【例】用ROM设计一个四位二进制码转换为格雷码的代码转换电路。

解：① 输入是四位自然二进制码 $B_3 \sim B_0$ ，输出是四位格雷码 $G_3 \sim G_0$ ，故选 $2^4 \times 4$ 的ROM。

② 四位二进制码转换为格雷码的真值表，即ROM的编程数据表如表所示。由此可写出输出函数的最小项之和式为：

$$G_3 = \sum m(8,9,10,11,12,13,14,15)$$

$$G_2 = \sum m(4,5,6,7,8,9,10,11)$$

$$G_1 = \sum m(2,3,4,5,10,11,12,13)$$

$$G_0 = \sum m(1,2,5,6,9,10,13,14)$$



二进制码转换为格雷码的真值表

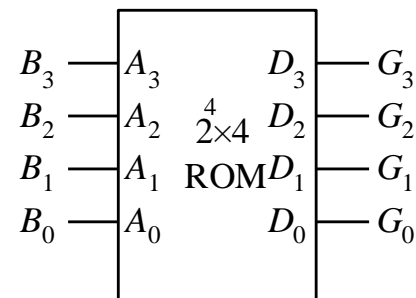
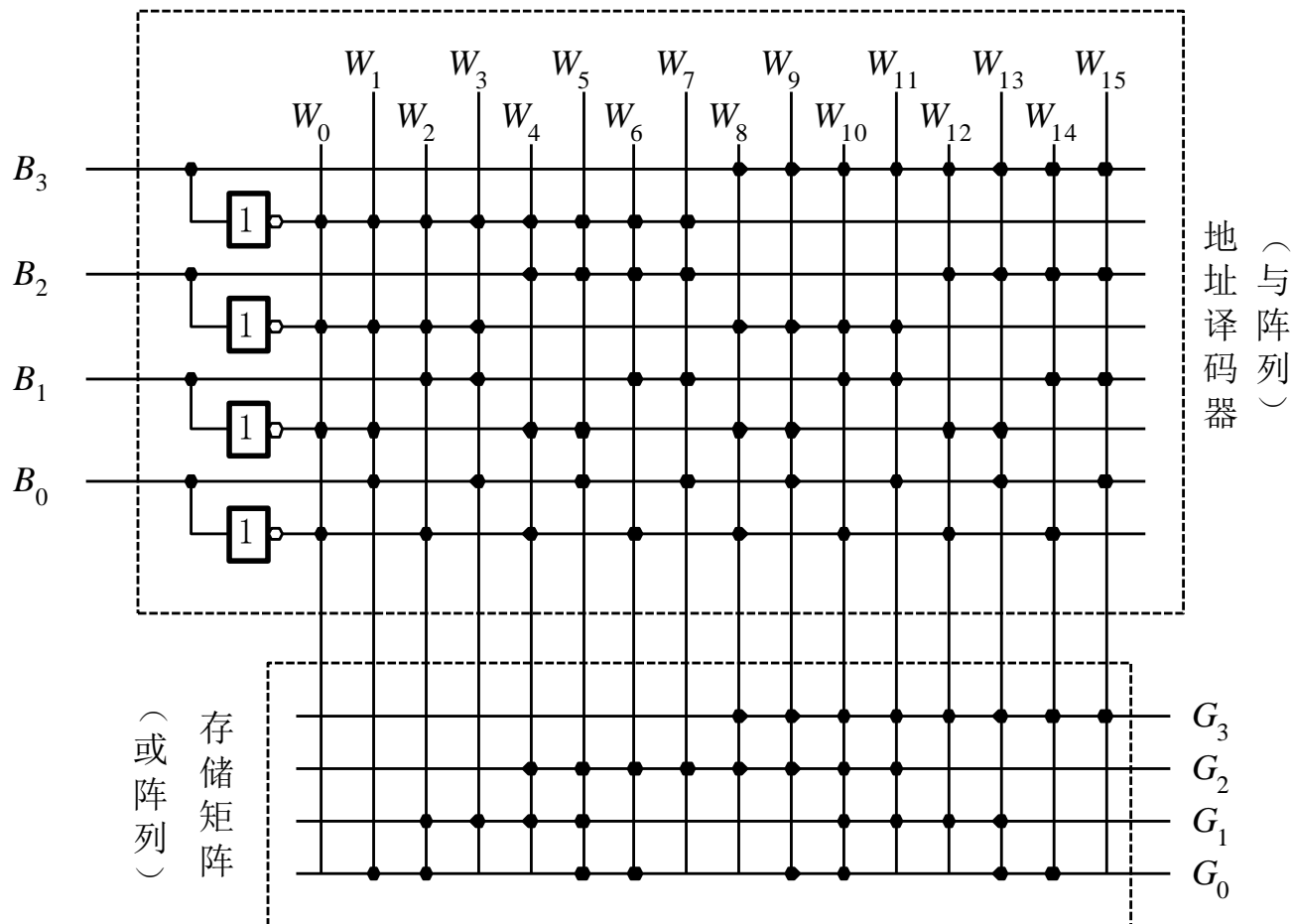
字	二进制码				格雷码			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
W_0	0	0	0	0	0	0	0	0
W_1	0	0	0	1	0	0	0	1
W_2	0	0	1	0	0	0	1	1
W_3	0	0	1	1	0	0	1	0
W_4	0	1	0	0	0	1	1	0
W_5	0	1	0	1	0	1	1	1
W_6	0	1	1	0	0	1	0	1
W_7	0	1	1	1	0	1	0	0
W_8	1	0	0	0	1	1	0	0
W_9	1	0	0	1	1	1	0	1
W_{10}	1	0	1	0	1	1	1	1
W_{11}	1	0	1	1	1	1	1	0
W_{12}	1	1	0	0	1	0	1	0
W_{13}	1	1	0	1	1	0	1	1
W_{14}	1	1	1	0	1	0	0	1
W_{15}	1	1	1	1	1	0	0	0



3 ROM的应用



③ 用ROM实现码组转换的阵列图及逻辑符号图分别如图 (a)、(b)所示。



(b)

动画展示

(a) 二进制码^(a)转为格雷码的阵列图; (b) 逻辑符号图



3 ROM的应用

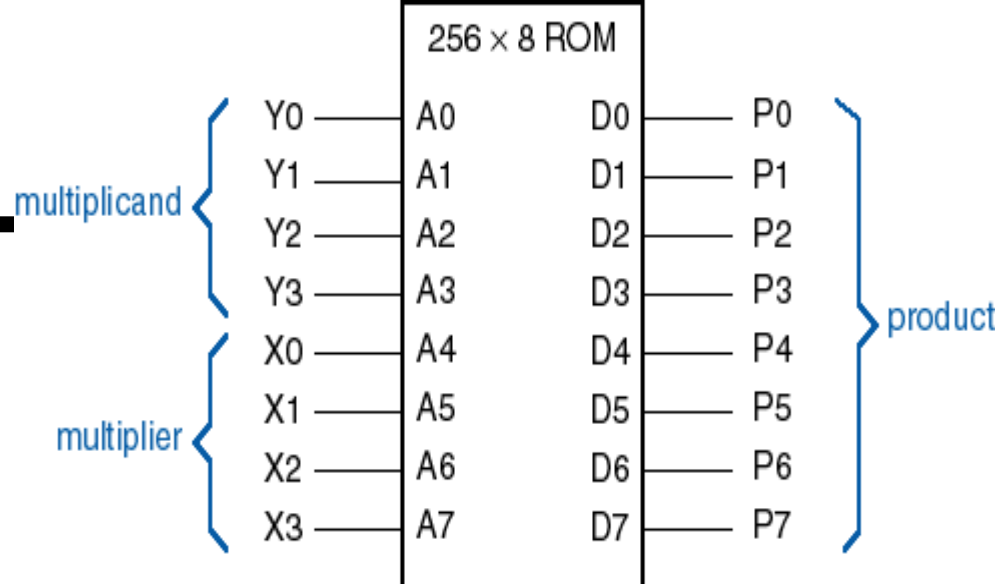


- 字符发生器
 - 实现字符发生器的基本原理是:将字符的点阵预先存储在**ROM**中, 然后顺序给出地址码, 从存储矩阵中逐行读出字符的点阵, 并送入显示器即可显示出字符。
- 数学函数表p582页表9-3
 - 实现计算机中的运算有两种方法, 一种是编写运算程序, 存入**ROM**中, 通过计算机执行运算程序。另一种方法是把因变量和自变量的函数关系存在**ROM**中, 好像查函数表一样。



4x4 乘法器

0X0,0X1,...,FXF



00:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10:	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
20:	00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
30:	00	03	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
40:	00	04	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
50:	00	05	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
60:	00	06	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
70:	00	07	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
80:	00	08	10	18	20	28	30	38	40	48	50	58	60	68	70	78
90:	00	09	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A0:	00	0A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B0:	00	0B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C0:	00	0C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D0:	00	0D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E0:	00	0E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F0:	00	0F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

2 读/写存储器

RAM

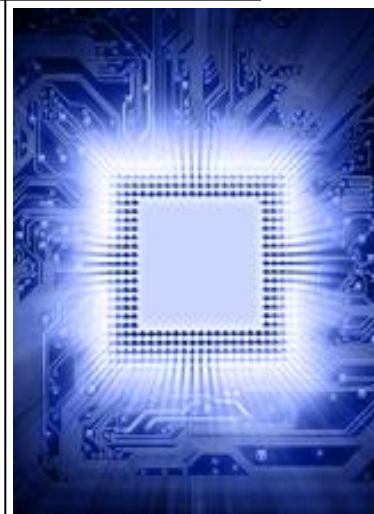
RAM的功能和特点

RAM的结构

地址译码方法

RAM的存储元

存储器容量的扩充





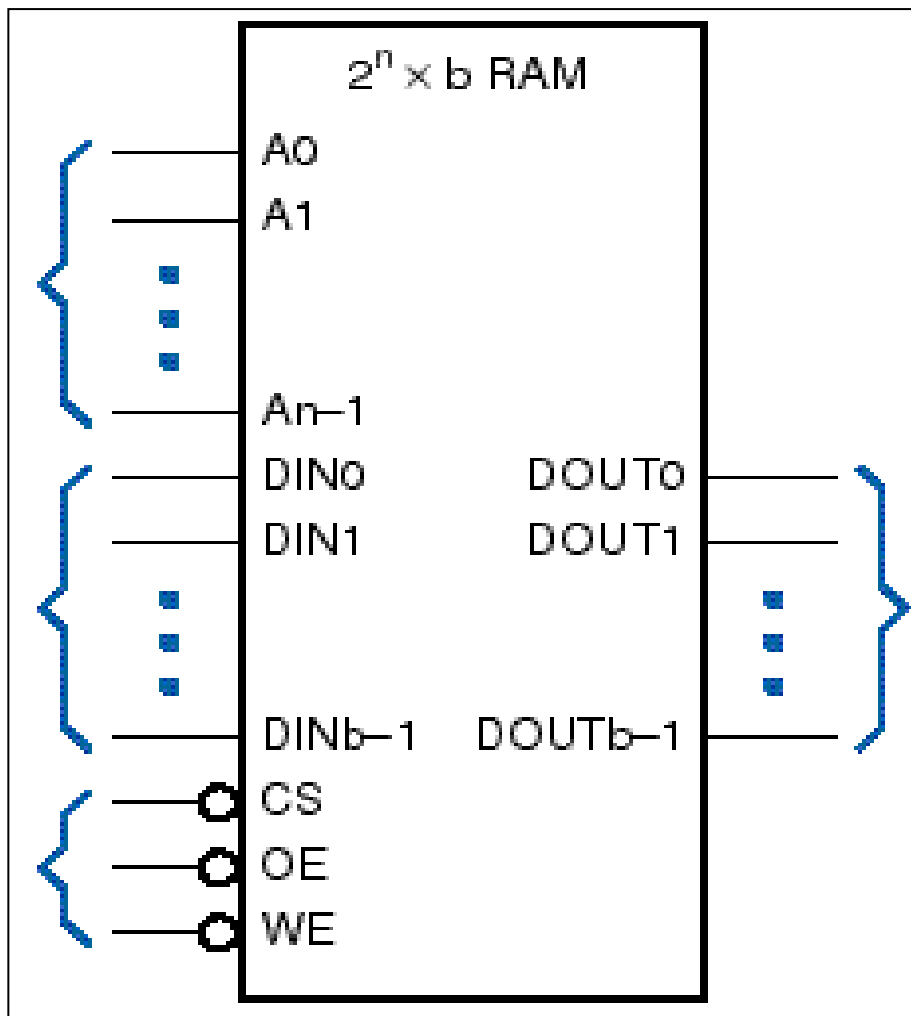
2、读写存储器



- 读/写存储器(Read/Write Memory RWM)是指可以在任何时候存储和检索信息的存储器阵列
- 现在数字系统中的读写存储器大多数是随机存取存储器 (Random-access Memory, RAM)，意思是读或写存储器的1个位所花费的时间与该位在RAM中的位置无关。
- **静态存储器**Static RAM,SRAM: 一旦在某个存储位置写入数据，**只要电源不被切断**，其存储内容保持不变，除非重新写入新内容。
- **动态存储器**Dynamic RAM,DRAM: 必须对存储数据进行**周期性读出和写入刷新**，否则存储器中的数据将会消失。

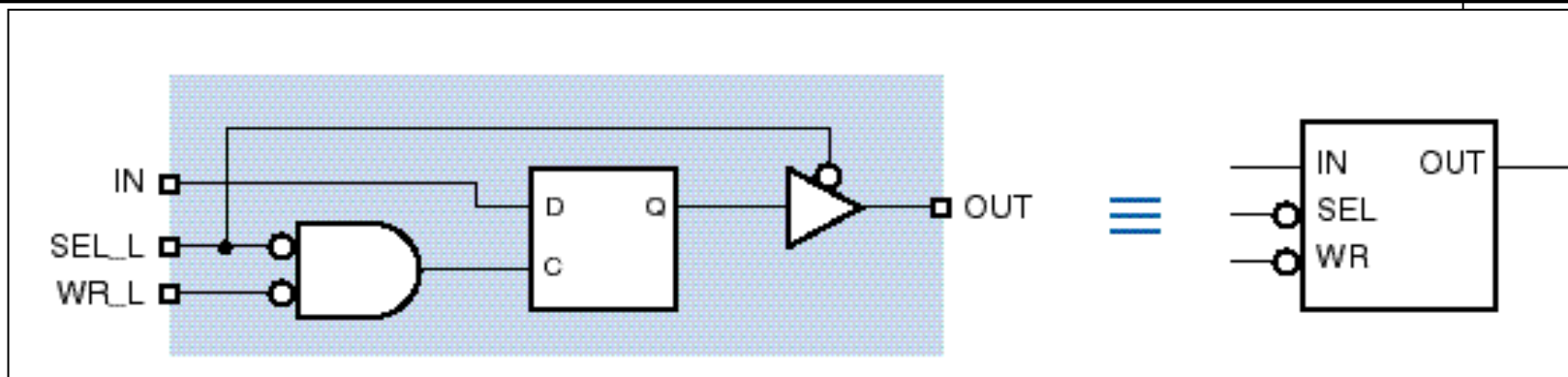


静态RAM(SRAM)

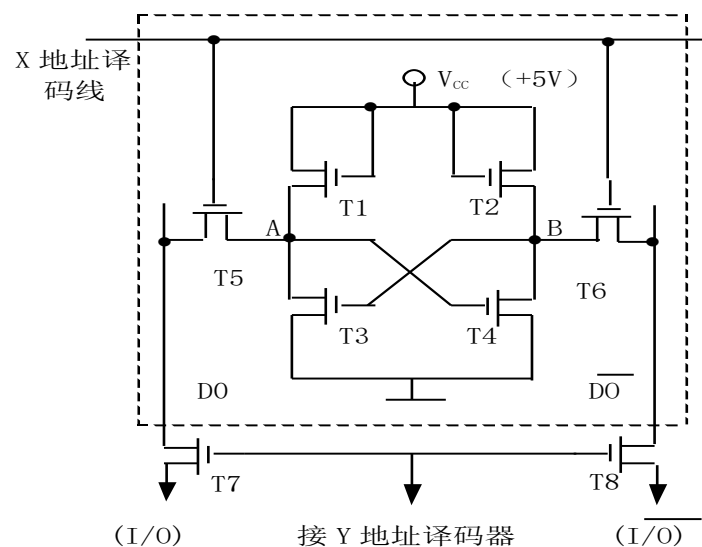


- 地址输入端 $A_{n-1} \dots A_0$
- 数据输入端 $DIN_{b-1..0}$
- 数据输出端 $DOUT_{b-1..0}$
- 片选 CS
- 写使能 WE
- 输出使能 OE

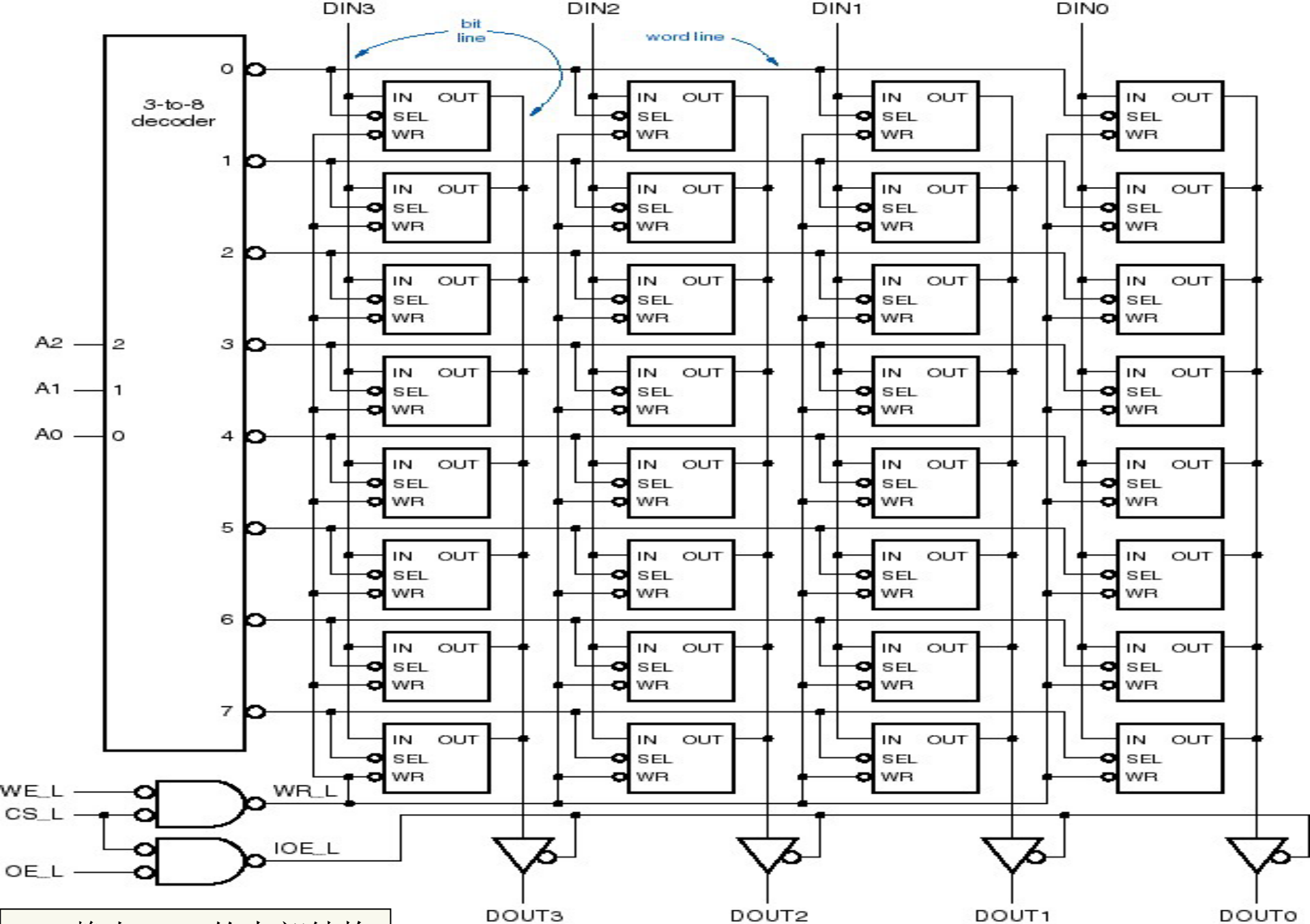
SRAM的基本单元



- SRAM中每个单元中存储器件为 **D锁存器**。
- IN: 数据输入
- SEL_L: 片选, 低有效
- WR_L: 写使能, 低有效
- 当SEL_L有效时, 数据被输出
- 当SEL_L和WR_L同时有效时, 数据被写入单元



六管基本存储电路单元

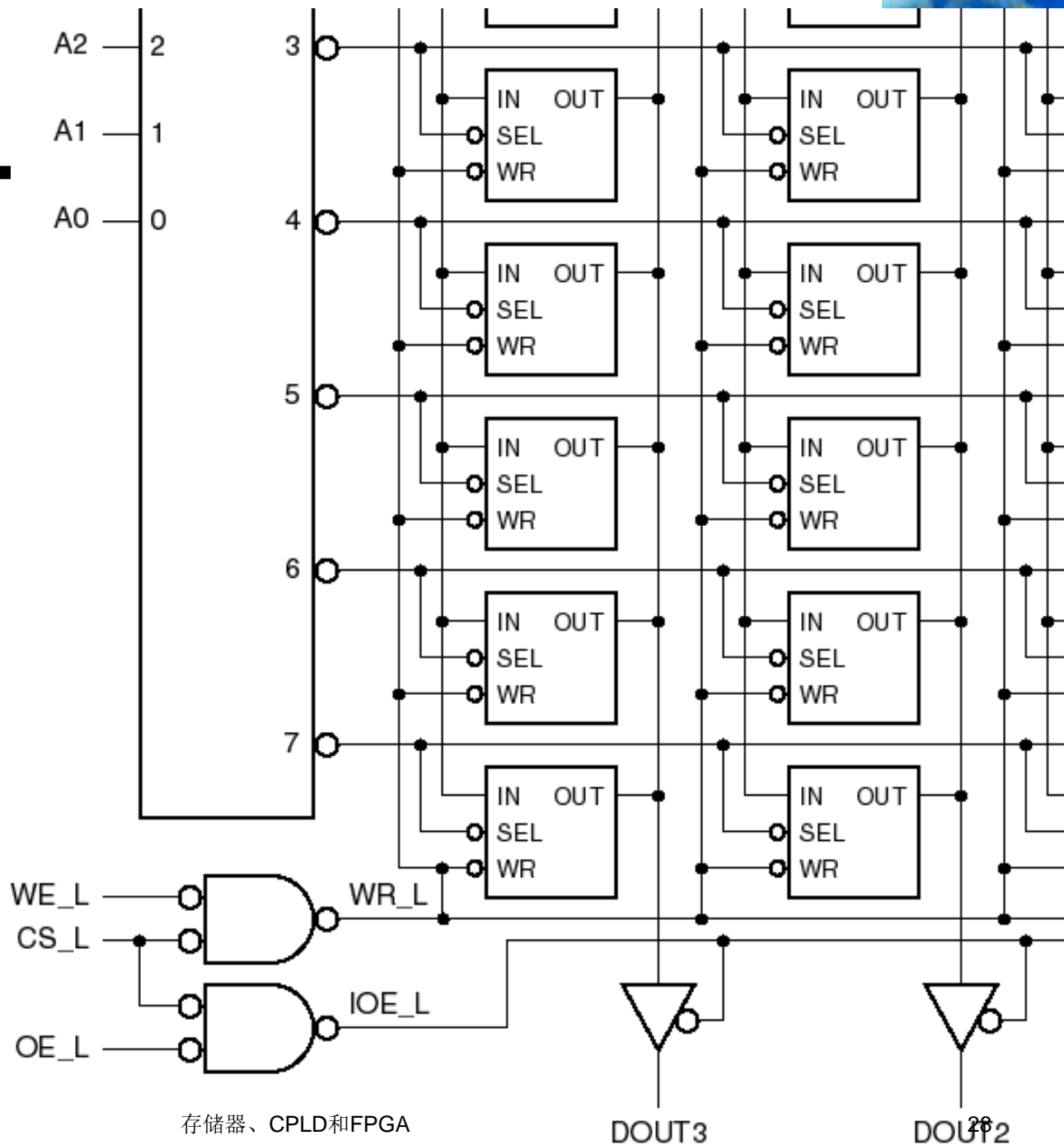


8X4静态RAM的内部结构

SRAM control lines

- 读操作，输出数据是地址输入的组合函数
- 写操作过程中，输入数据存储在锁存器中
- 当CS_L、WE_L同时有效时，WR_L才有效。
- 地址输入在WR_L有效前和失效后都必须保持一段时间稳定

2020年6月3日



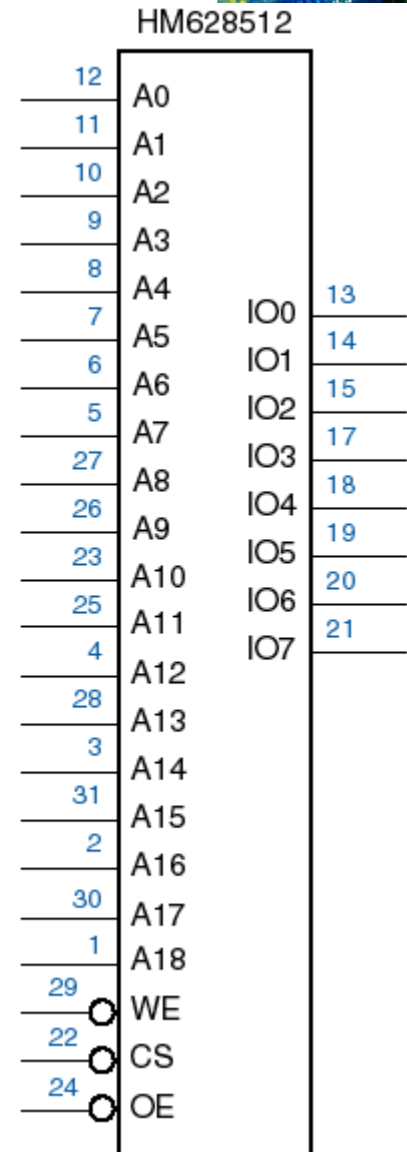
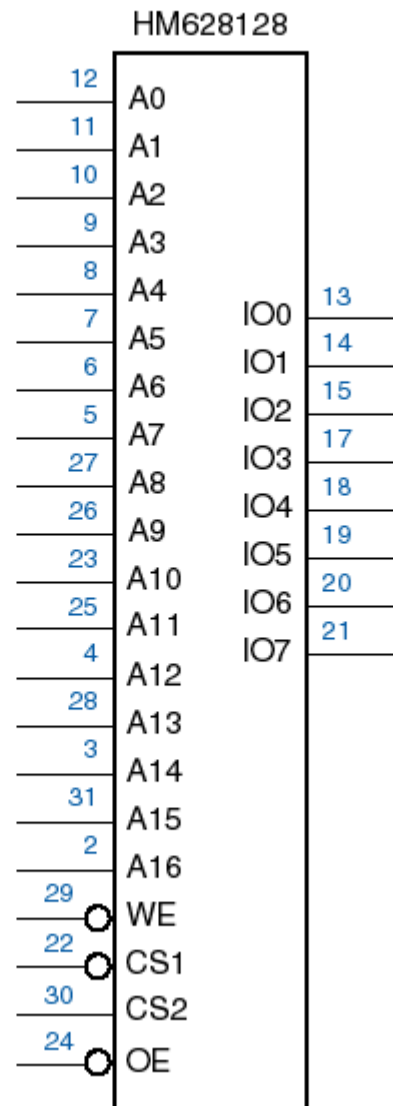
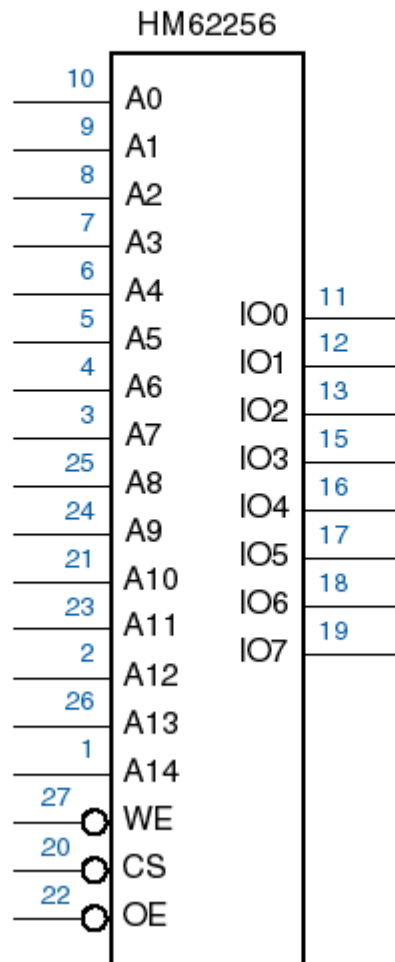
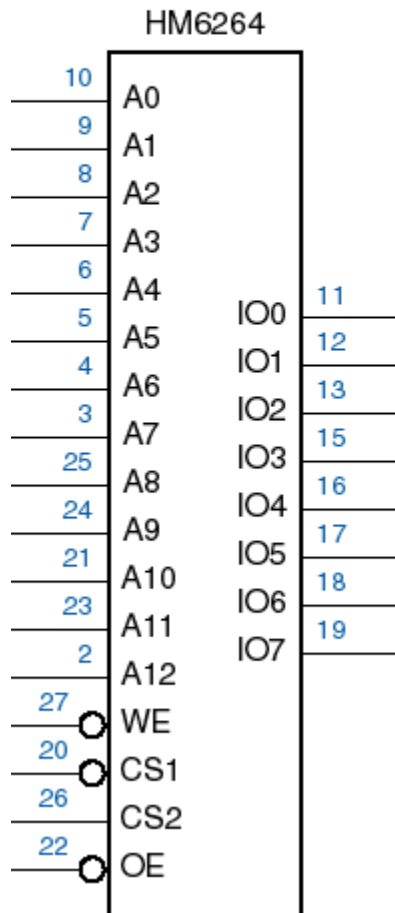
存储器、CPLD和FPGA

DOUT3

DOUT2

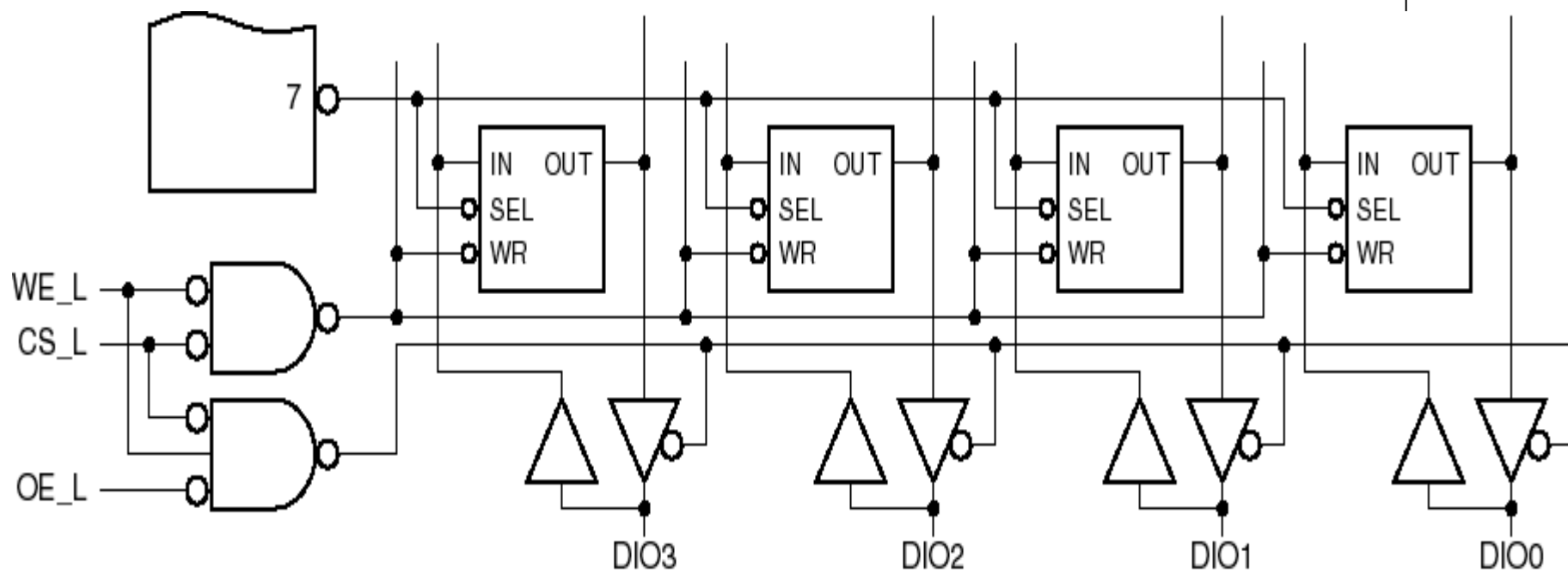
SRAM devices

- Similar to ROM packages





双向数据总线的SRAM

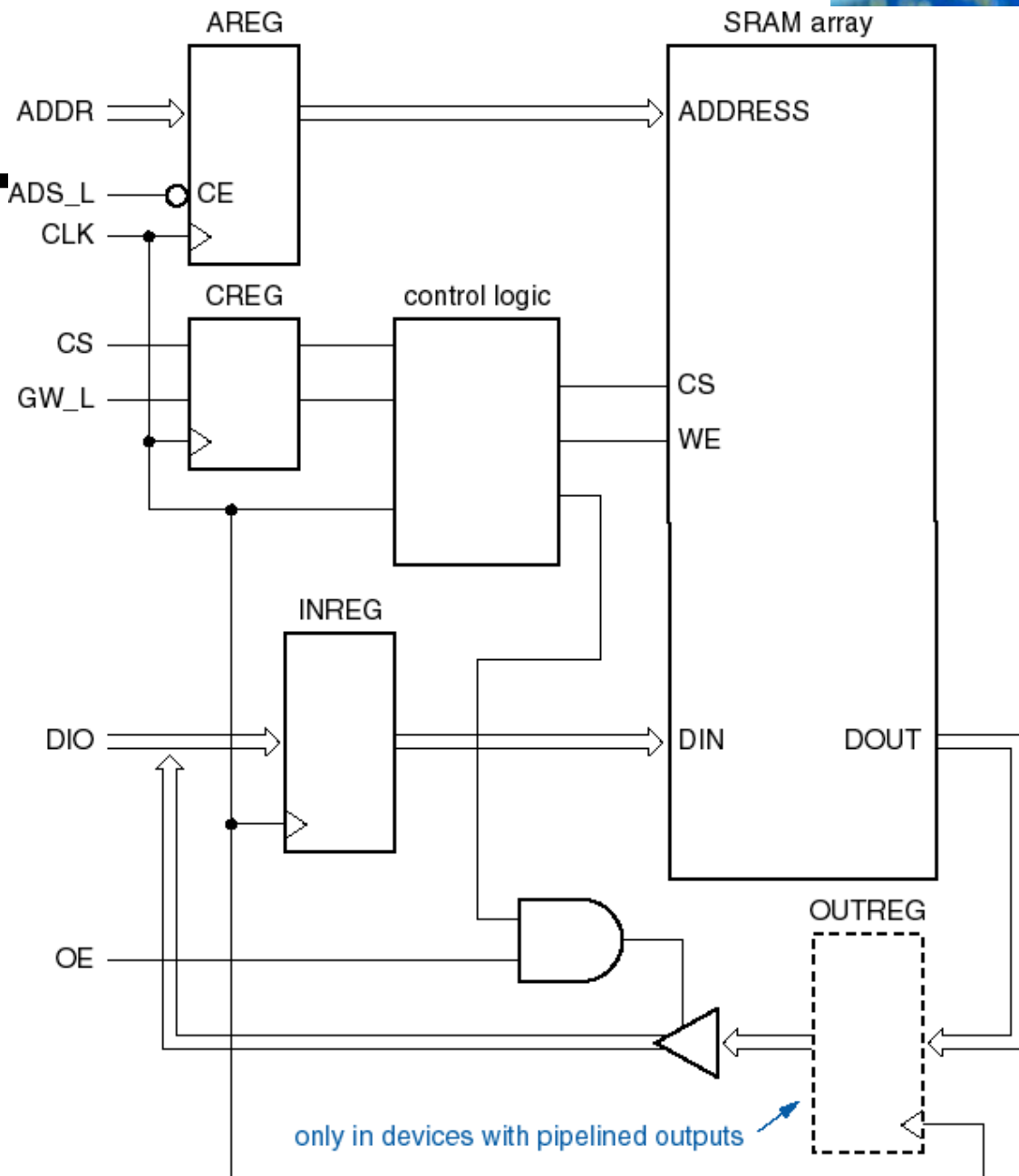


- 同一数据引脚既可用于读操作又可用于写操作。



同步SRAM

- SSRAM: 使用锁存器
- 具有一个用于控制、地址和数据与时钟控制端口
- E.g., Pentium cache RAMs





SSRAM类型



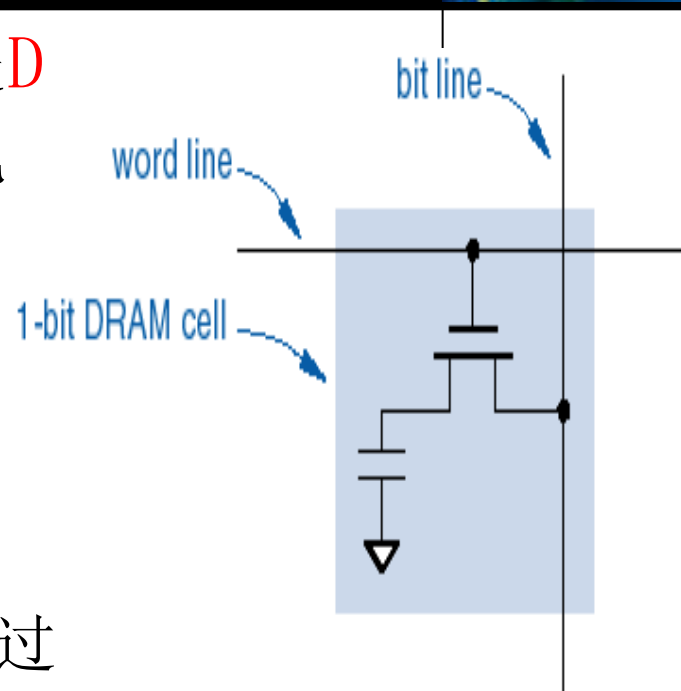
- 具有流过式输出的后写SSRAM
- 具有流水线输出的后写SSRAM
- 零总线周转ZBTSSRAM
- 四倍数据速率QDRSSRAM



2. 动态随机存储器 (DRAM)

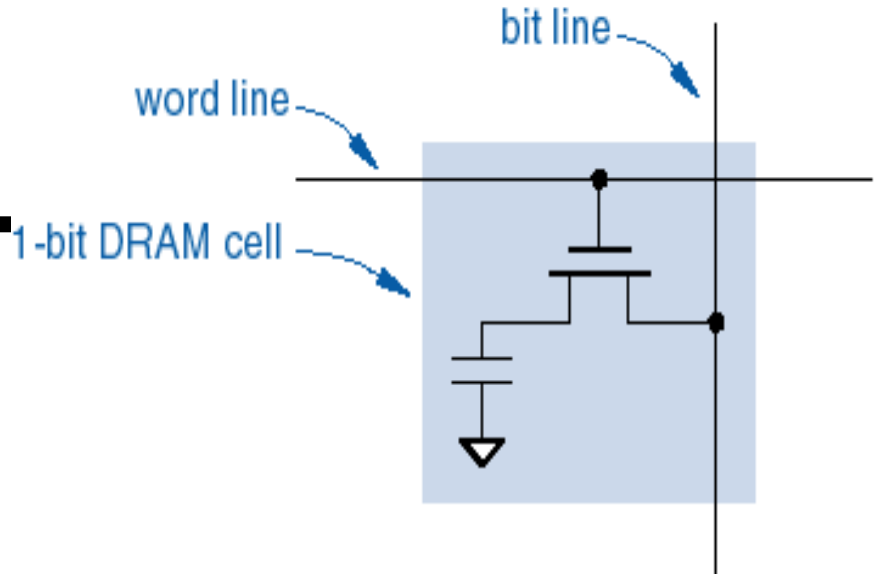


- 静态RAM中最基本的存储器单元是D锁存器，需要4-6个门电路来实现。
- 为提高集成度，动态RAM中每位只用一个晶体管的存储器单元。
- 动态RAM结构
 - 在微小的电容器上存储信息，并通过一个晶体管来存储信息。
 - 字线设置为高电平，导通晶体管
 - 存储1，位线设置为高电压，电容器充电
 - 存储0，位线设置为低电压，电容器放电





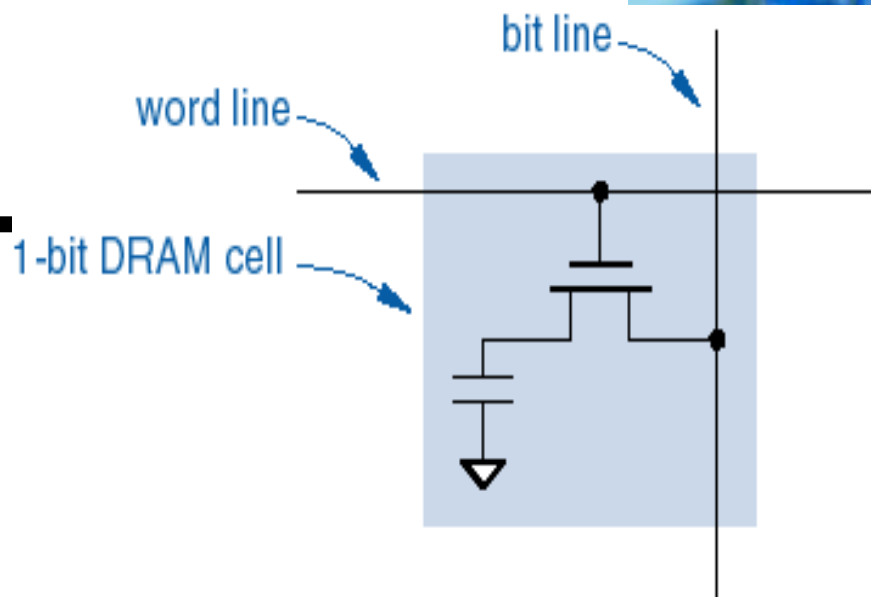
DRAM读操作



- 位线被预充电电压到 $V_{DD}/2$ 。
- 字线设置为高电平。
- 根据电容器电压是高电平还是低电平，检测预充电的位线是被推高了还是被推低了。（通过读出放大器来检测）
- 注意：
 - 读出一个单元会破坏存储在电容器上的原始电压。
 - 需要在读出数据后重新写入原来单元中



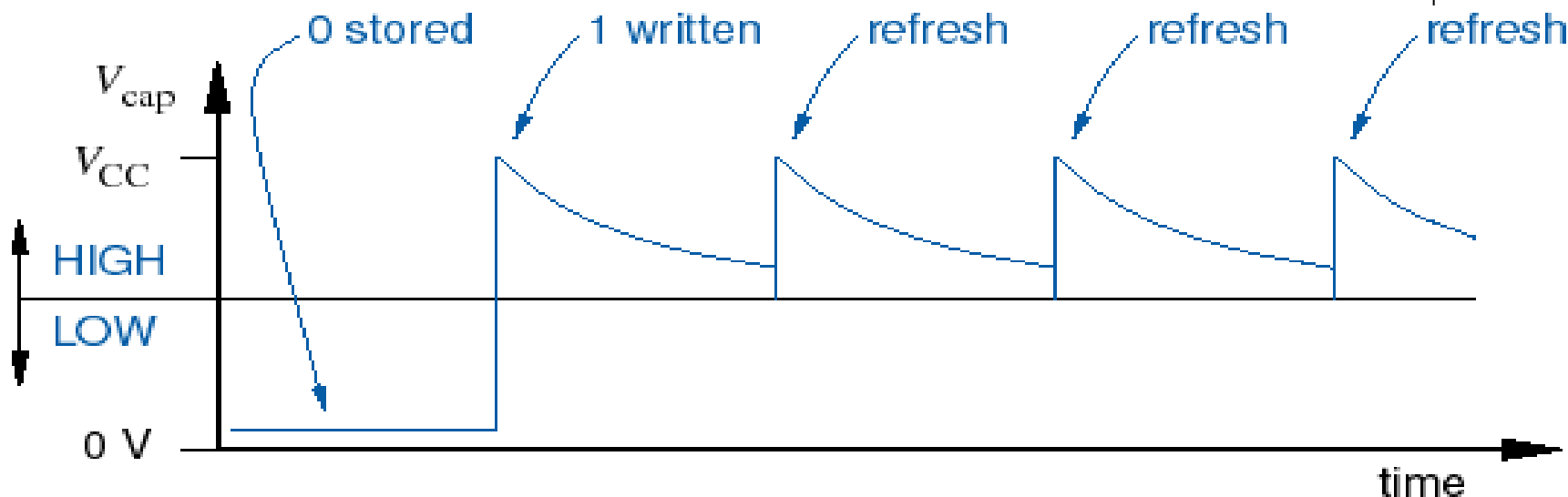
DRAM 写操作



- 使得字线为高电平。
- 设置位线为低或高电平来存储0或1，通过电容的充放电来实现。
- 使得字线为低电平。
- 注意：存储为1，最终会逐渐放电而消失。



DRAM 刷新操作



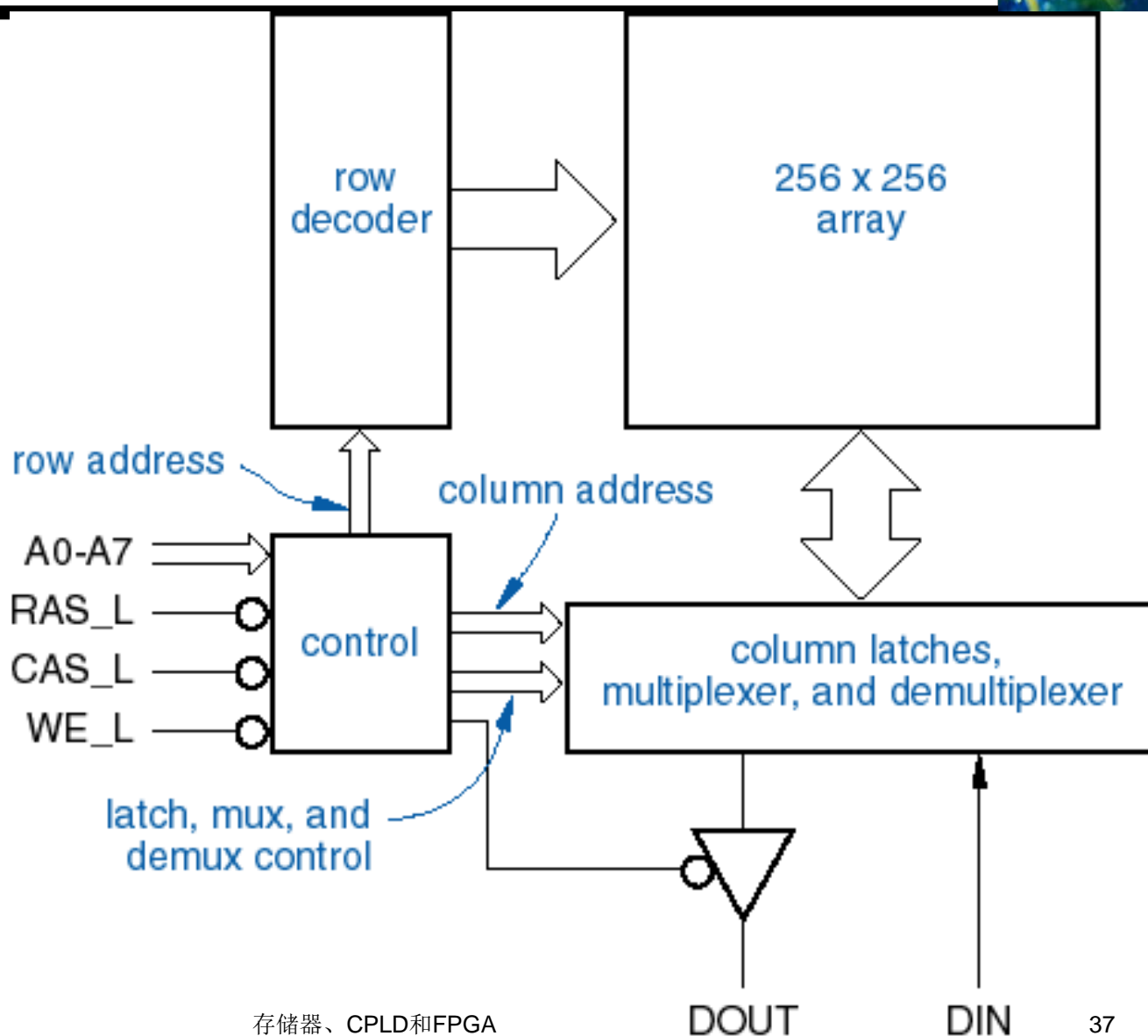
- 顺序地将每一个单元中电压有点下降的内容读入到D锁存器，然后写回一个来自锁存器的固定低电平或高电平
- 典型器件每隔4-64ms刷新存储单元一遍。
- 在笔记本休眠期间主要的功耗就在于保持DRAM的刷新。



DRAM-chip内部结构



64K x 1
DRAM





SDRAM定时



- 完成一个读周期的步骤：
 1. 选择把所期望的地址段，发出**PRE**命令。对该段中的所有位线预充电，使其达到 $U_{DD}/2$ 。
 2. 等待几个时钟沿，直到预充电操作完成。
 3. 再次选择所期望的段，并把期望地址的高位输入到输入端**A[11:0]**，并发出**ACTV**命令。
 4. 等待几个时钟沿（**RAS-CAS**延迟），使得读出的4096位的字在内部稳定下来
 5. 将期望地址的低位送到输入端**A[11:0]**，并发出**READ**命令。
 6. 在等待几个时钟沿（**CAS**等待），4位地址从列多路选择器传送到**DQ[1:4]**
 7. 读取输入输出引脚**DQ[1:4]**上的数据。



EDO DRAM: 采用快速页面模式（FPM），对地址连续的多个单元进行读写；另一方面，在输入下一个地址时，仍然允许数据输出 → 扩展了数据输出的时间

SDRAM: 采用同步的方式进行存取，传统DRAM采用异步的方式进行存取。

送往SDRAM的地址、数据及控制信号都是在一个时钟信号的上升沿被采用和锁存的，SDRAM输出的数据也在时钟的上升沿锁存到芯片内部的输出寄存器。

DDR SDRAM: 双倍数据速率同步内存（DDR（double data rate）SDRAM），是在SDRAM基础上发展的。

DDR不仅在时钟上沿进行操作，在时钟脉冲的下沿也可以进行一次对等的操作（读或写）。



双通道 DDR DRAM: 双通道内存体系包含了两个独立的、具备互补性的64位智能内存控制器，两个内存控制器都能够在彼此间零等待时间的情况下同时运作，形成了128位宽度的内存数据通道，使内存的带宽翻了一番（理论上，实际为3%~18%），如P4处理器

DDR2 SDRAM: 是在DDR SDRAM基础上发展的，采用锁相技术，可以在一个时钟周期内传输4次数据。

如：Intel 915P Express芯片组支持DDR2 内存，其频率为533MHz/400MHz

DDR3 SDRAM: 提供了相较于DDR2 SDRAM更高的运行效能与更低的电压，是DDR2 SDRAM的后继者（增加至八倍），也是现时流行的内存产品。



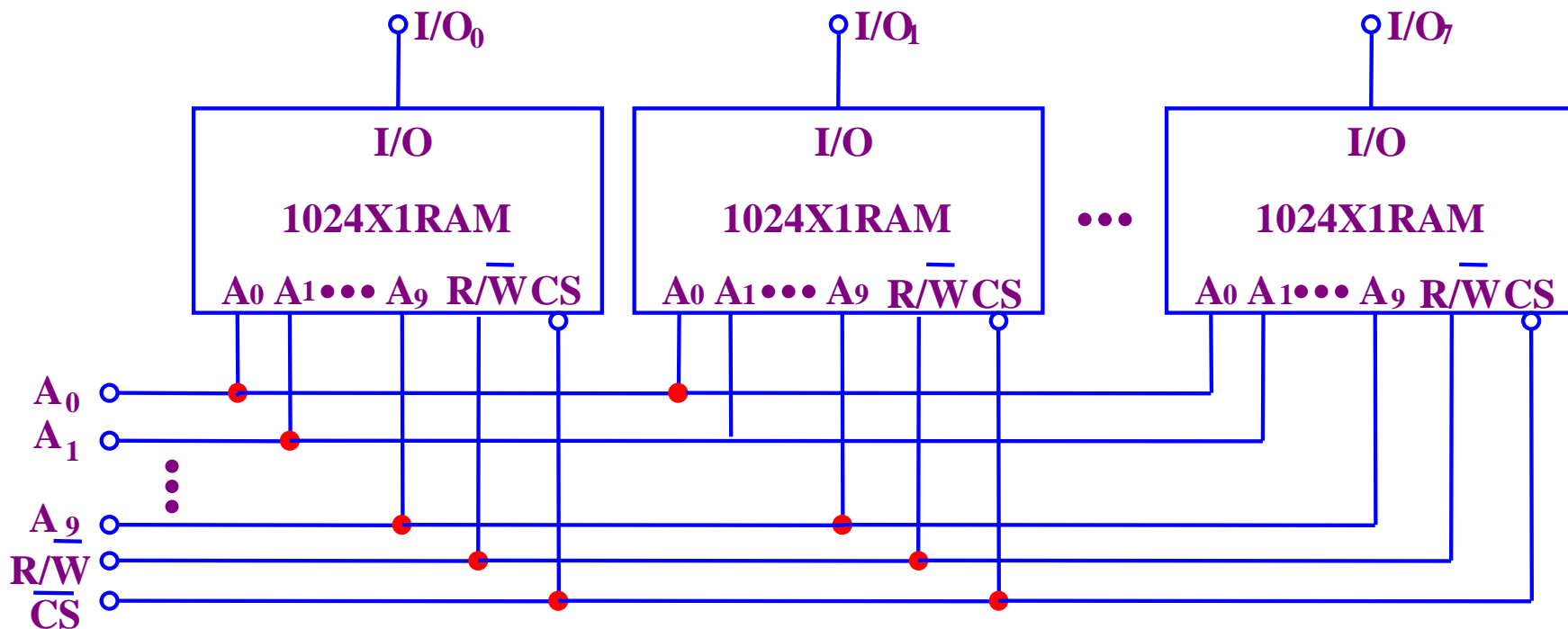
存储器的容量扩展（一）



一、位扩展

将8片 1024×1 的RAM扩展成 1024×8 的RAM

要点：所有RAM使用相同的控制线和地址线。





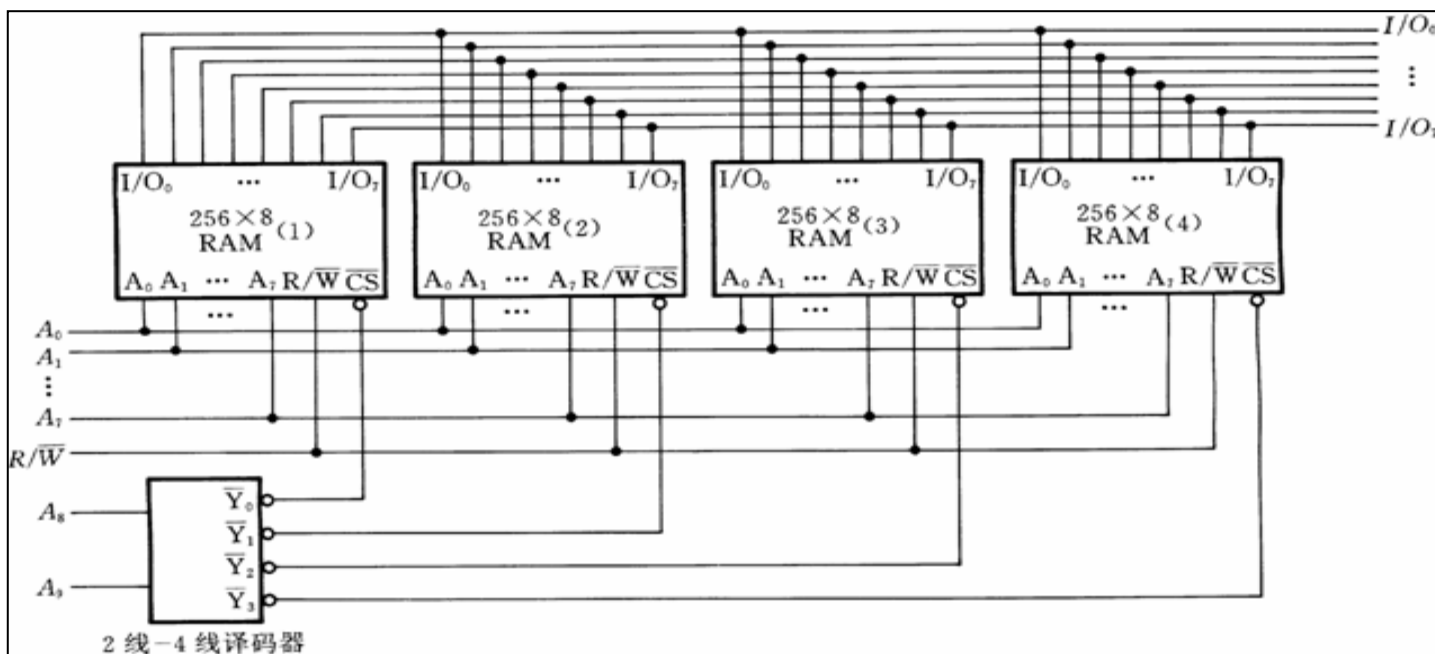
存储器的容量扩展（二）

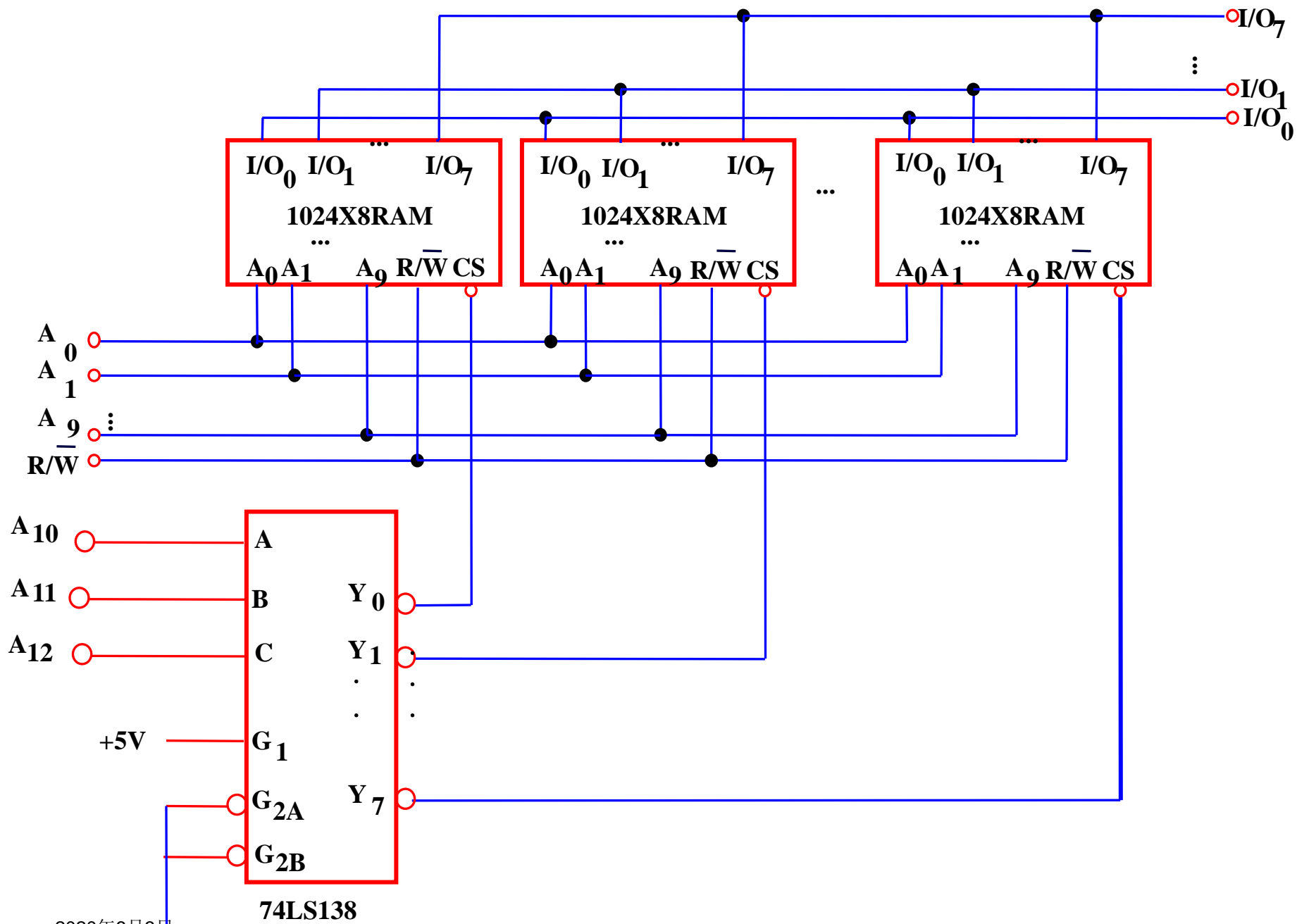


二、字扩展

将4片 256×8 的RAM扩展成 1024×8 的RAM

要点：所有RAM使用相同的地址线和读写控制线，但是片选信号不同，由译码器产生。







存储器容量的扩充



- 3、字位同时扩展
 - 增加地址译码电路
 - 合并位扩展数据
- 【例题】
 - 有 256×4 位芯片，问地址线多少位，数据线多少位？
 - 使用上述芯片组成 1024×4 位存储器，问要多少芯片？
 - 使用上述芯片组成 256×16 位存储器，问要多少芯片？
 - 使用上述芯片组成 2048×32 位存储器，问要多少芯片？