请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

第7章
半导体存储器

Leng Ximo

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

半导体存储器

- 概述
- 只读存储器 ROM
- 用 ROM 实现组合逻辑函数
- 随机存储器 RAM
- 存储器容量的扩展

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

半导体存储器

概述

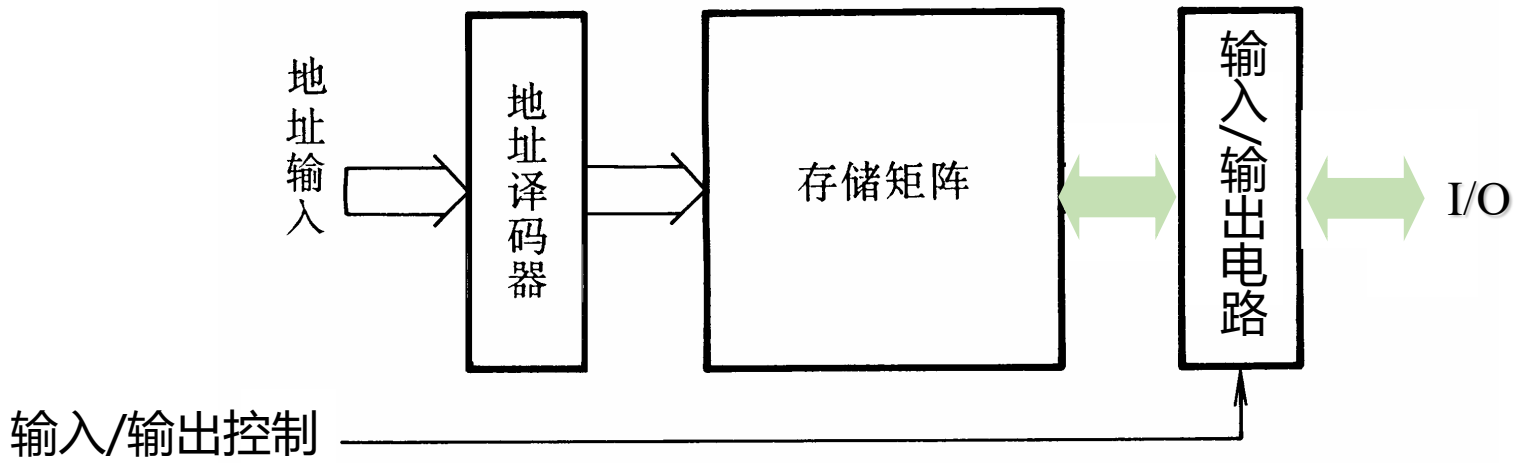
半导体存储器指的是能够存储大量的二值数据（即 0 和 1）的半导体器件；本章研究的内容就是常见的几种类型的半导体存储器的结构、基本原理和工作特点；

因为半导体存储器需要存储的数据数量极其庞大，因此半导体存储器的电路结构不可能是每个存储单元的输入和输出全部引出；因此，半导体存储器设计的基本思想是，为每个存储单元分配一个地址编码，根据输入的地址代码，选择指定的存储单元中的数据，在一个公共的输入/输出端进行读取或写入；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

半导体存储器

半导体存储器的基本结构



不同类型的半导体存储器都以此基本的结构框图作为基础进行变化的，例如地址译码器可以再进一步分为行译码器和列译码器、存储矩阵中的各个存储单元可以使用不同的器件进而有不同的工作特点、只读存储器无法输入最后一级只是一个单向的输出缓冲器等；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

半导体存储器

半导体存储器的分类

按照存/取功能可分为：

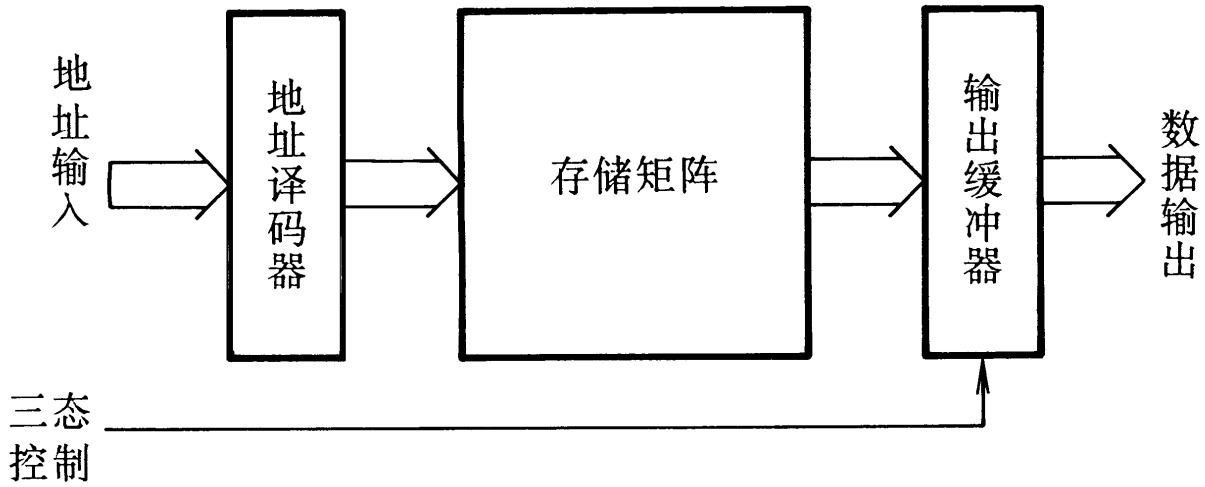
- ROM 只读存储器 (Read-Only Memory)
  - 掩模 ROM
  - PROM（可编程 ROM）(Programmable ROM)
  - EPROM （可擦除的可编程 ROM）(Erasable Programmable ROM)
- RAM 随机存储器 (Random Access Memory)
  - SRAM （静态存储器）(Static RAM)
  - DRAM （动态存储器）(Dynamic RAM)

按照制造工艺（即使用的半导体器件类型）又可分为双极型和 MOS 型；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

ROM 的结构框图



- 地址译码器：将输入的 n 位地址代码译为 2^n 个独立的高低电平，即选择其中一个地址存储的数据；
- 存储矩阵：由基本的存储单元排列而成，存储单元使用的不同工艺、不同性质的器件决定了存储器的类型和工作特点；
- 输出缓冲器：与系统总线连接，实现对输出的三态控制，同时提高带负载能力；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

地址译码器

存储器的输入是 n 位的地址代码，对应 2^n 个状态即 2^n 个地址，地址译码器的作用就是根据输入的地址代码译出独立的高低电平，选择此地址中存储的数据进行输出；而每一个地址中并不一定只存一位数据，可以是多位，我们把每个地址对应的这多位数据整体称为“字”，地址译码器的 2^n 个输出线就称为字线；

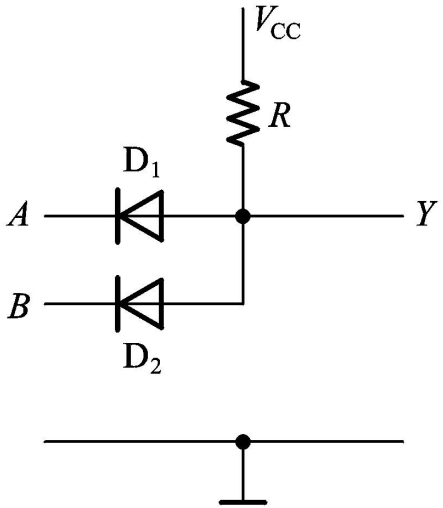
而根据第四章组合逻辑中译码的基础知识，一个二进制译码器（$n/2^n$）的输出是 n 位代码全部的最小项，最小项是各个输入变量的原变量或反变量相与的形式，所以，地址译码器实质上就是一个**与阵列**；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

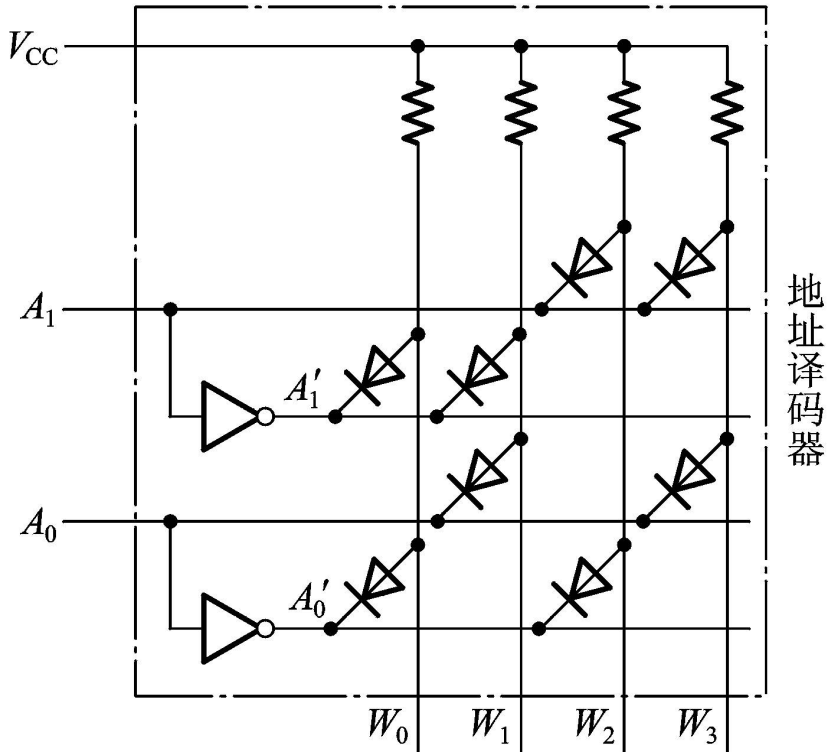
只读存储器 ROM

地址译码器

根据第三章门电路的基础知识，共阳二极管和上拉电阻可以构成与门，因此一个典型的由二极管组成的与门阵列如下（这里以 n = 2，$2^n = 4$ 为例）：



$$Y = A \cdot B$$



译码器的输出是 2^n 条字线，有且仅有一条字线为高电平，即只有一个地址、一个字被选中；

$$W_0 = A_1' A_0'$$
$$W_1 = A_1' A_0$$
$$W_2 = A_1 A_0'$$
$$W_3 = A_1 A_0$$

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

存储矩阵

存储矩阵是由许多存储单元排列而成，每个存储单元中存放一位二值数据即 1 或 0；

根据前面的分析，地址译码器译码的结果是根据输入的地址代码，选择其对应的地址中的数据，即选择将其对应的字线将其拉至高电平（其余的字线为低电平），假设每个地址中对应 m 位数据，即每个字的长度为 m，则存储器的 m 个并行输出端口可以并行输出当前地址的字，即当前地址中存储的 m 位数据，称为 m 条位线，；

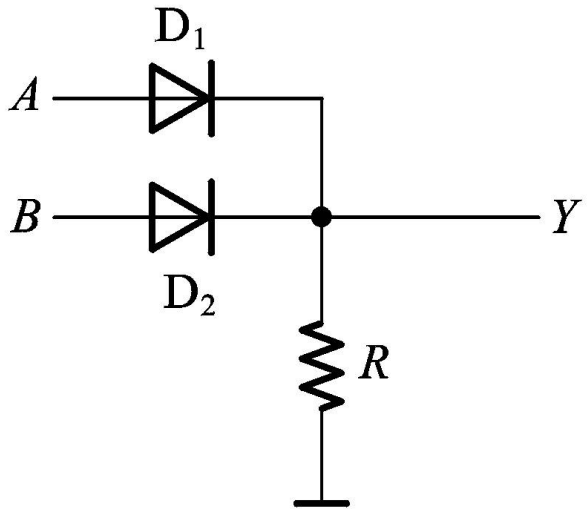
根据第四章组合逻辑中数据选择的基础知识，存储矩阵和前面的地址译码器整体其实就是 m 个并列的数据选择器，这 m 个并列数据选择器的地址输入端为相同的地址代码；对于每个输出端口，其输出一定是某一个字中的对应位，所以是“或”的逻辑关系，即存储矩阵实质上就是一个**或阵列**；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

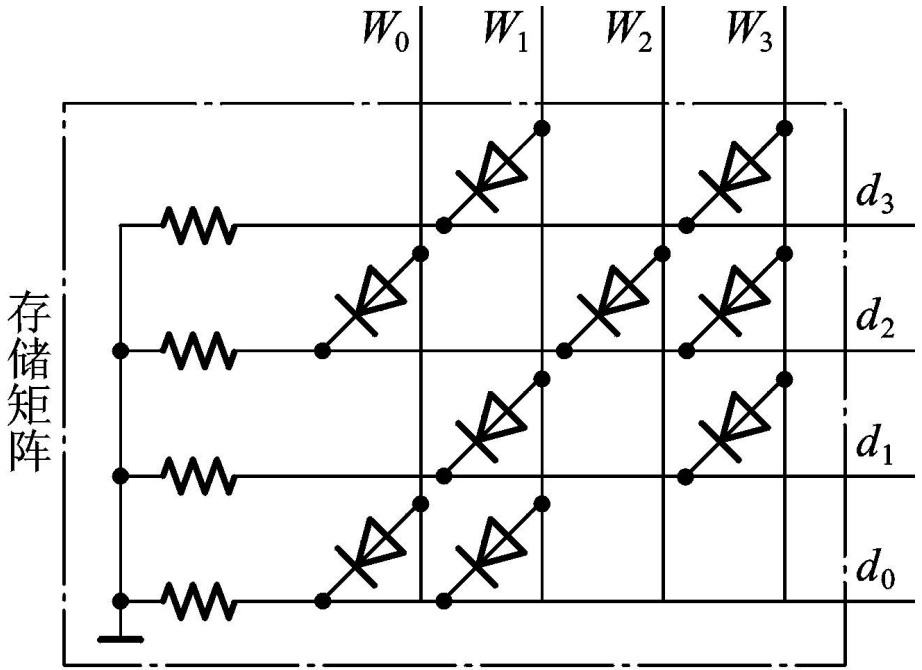
只读存储器 ROM

存储矩阵

根据第三章门电路的基础知识，共阴二极管和下拉电阻可以构成或门，因此一个典型的由二极管组成的或门阵列如下（这里以 m = 4 为例，即每个字含有 4 位）：



$$Y = A + B$$



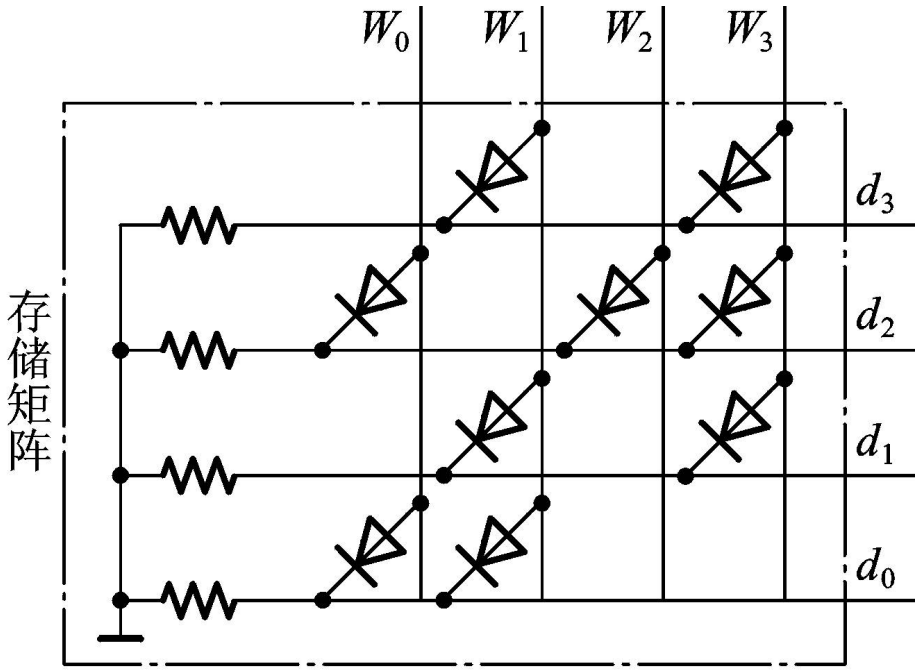
$$d_3 = W_1 + W_3$$
$$d_2 = W_0 + W_2 + W_3$$
$$d_1 = W_1 + W_3$$
$$d_0 = W_0 + W_1$$

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

存储矩阵

根据我们前面的分析，地址译码器的 2^n 条字线只有一个是高电平，其余的都为低电平，即只有一个地址对应的一个字中存储的 m 位数据会被并行输出，我们可以得到此存储矩阵的数据表：



$$d_3 = W_1 + W_3$$
$$d_2 = W_0 + W_2 + W_3$$
$$d_1 = W_1 + W_3$$
$$d_0 = W_0 + W_1$$

$$W_0 = A_1' A_0'$$
$$W_1 = A_1' A_0$$
$$W_2 = A_1 A_0'$$
$$W_3 = A_1 A_0$$

地　址		选中的字线	数　据			
A_1	A_0	W	d_3	d_2	d_1	d_0
0	0	$W_0 = 1$	0	1	0	1
0	1	$W_1 = 1$	1	0	1	1
1	0	$W_2 = 1$	0	1	0	0
1	1	$W_3 = 1$	1	1	1	0

2^n 个地址

每个字的长度：m 位

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

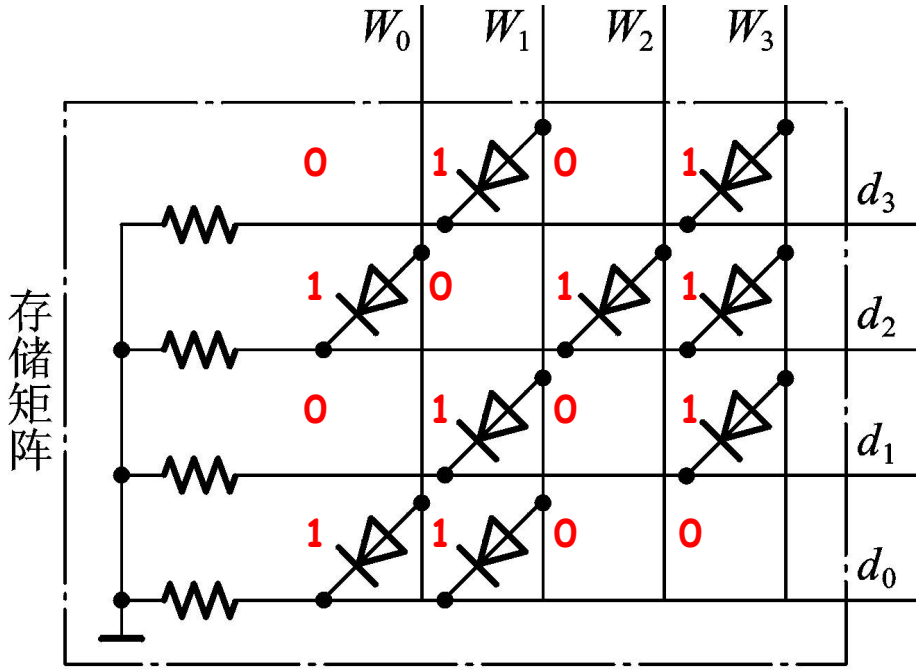
只读存储器 ROM

易错：
存 1 / 存 0 与最后输出 1 / 输出 0 并不一定一致，
取决于输出缓冲级是同相还是反相!

存储矩阵

根据我们前面的分析，我们可以看到存储矩阵的特点是，字线和位线的每个交叉点都是一个存储单元，交叉点处接有半导体器件则相当于存 1，没有接半导体器件则相当于存 0；

地址		选中的字线	数据			
A_1	A_0	W	d_3	d_2	d_1	d_0
0	0	$W_0 = 1$	0	1	0	1
0	1	$W_1 = 1$	1	0	1	1
1	0	$W_2 = 1$	0	1	0	0
1	1	$W_3 = 1$	1	1	1	0



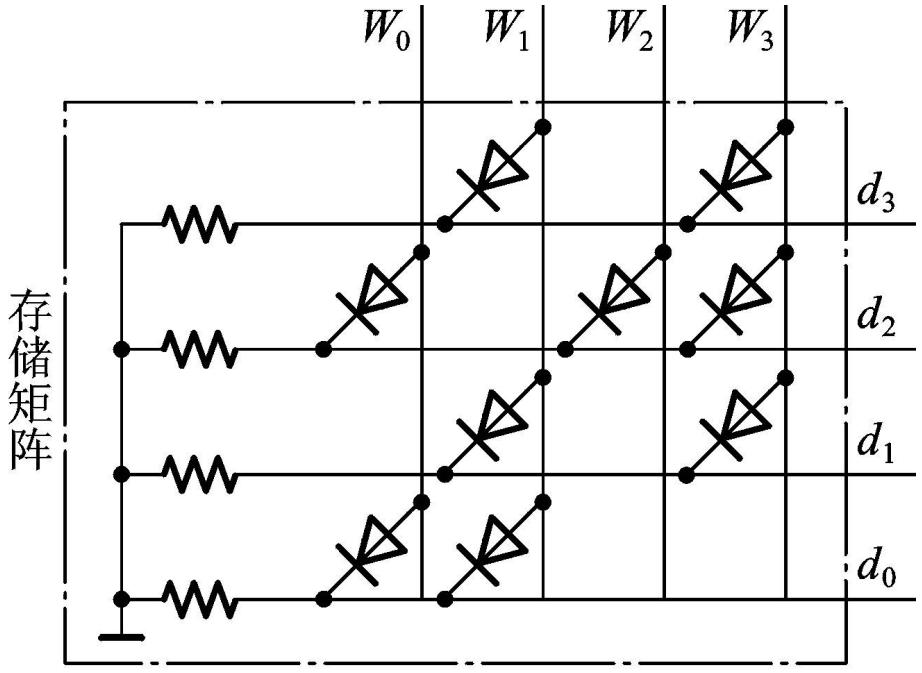
需要注意的是，不能直接将右边的存储单元矩阵直接填入左边的数据表，行和列要进行对调，因为左边的数据表中的每一行代表一个字，每一列代表不同字的一个位；而右边的电路图中每一列是一条字线，每一行是一个位线；

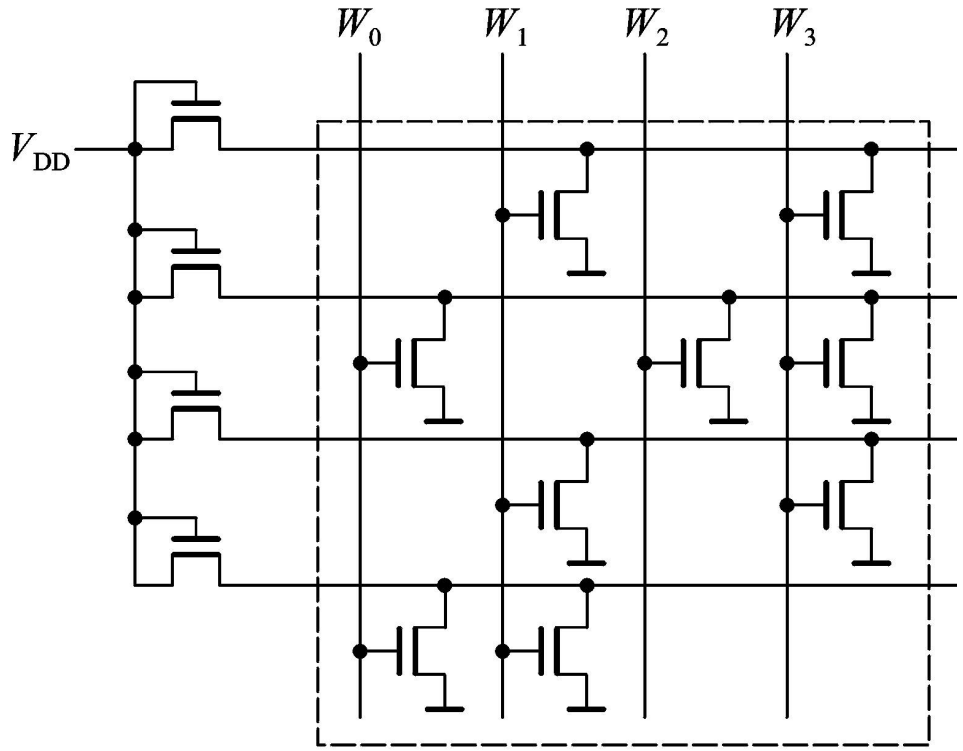
请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

存储矩阵

根据制作工艺的不同，还可以将由二极管构成的存储矩阵替换为 MOS 管存储矩阵；当然也可以替换为晶体三极管存储矩阵，原理相同；





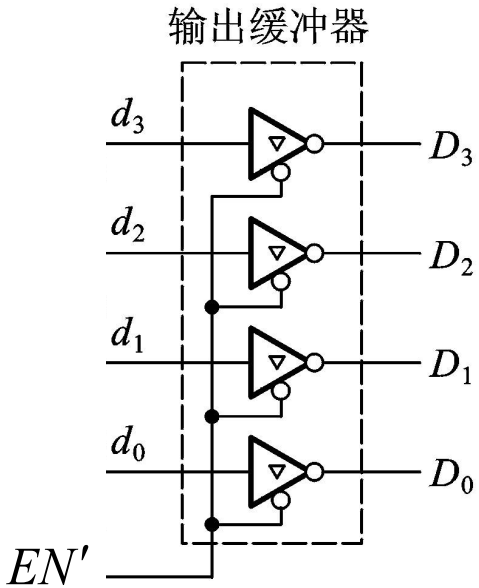
请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

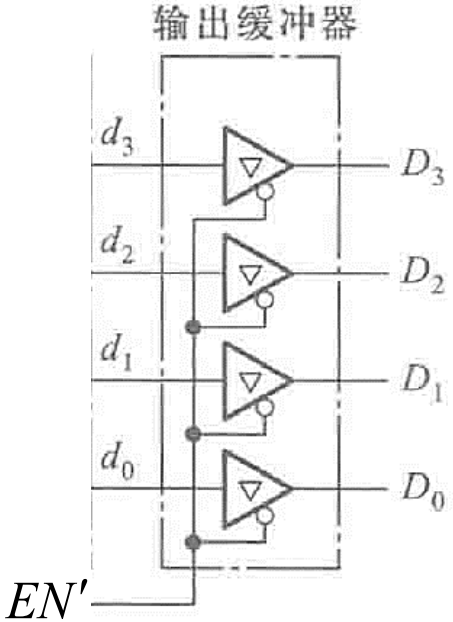
注意：
这里所说的各个输出端“轮流输出”，
针对的是不同的器件，
对于每一个器件，其 m 个端口仍然是同时并行输出的！
（每个器件的 m 个输出端是同一个使能端信号）

输出缓冲器

对于 ROM，只可以读不可以写，因此其最后一级就是一个单向的输出缓冲器，采用 m 个三态门；根据第三章门电路的基础知识，三态门可以使不同器件的各个输出端轮流将数据输送到总线上，通过使能端控制保证互不干扰，且采用这样的一个三态门可以保证输出逻辑电平的标准化，同时提高带负载能力；



反相的输出缓冲器
（输出对存储矩阵取反）

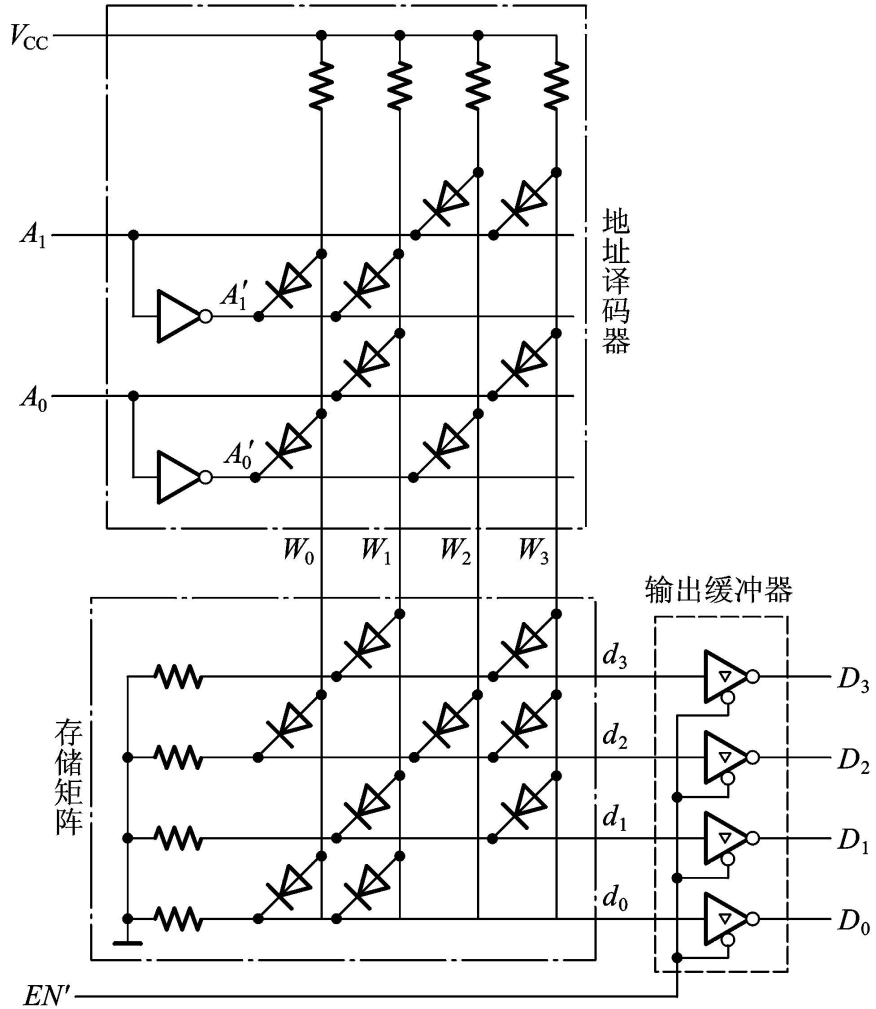


同相的输出缓冲器
（输出与存储矩阵一致）

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

存储器的容量



如图所示的 ROM，其有 n 位地址代码输入，对应 2^n 个字；每个字的长度为 m，即有 m 位输出；存储器能够存储共 $2^n \times m$ 位二值数据（0 或 1）所以记为 $2^n \times m$ 位的 ROM；

2^n 为字数，m 为字长即位数；

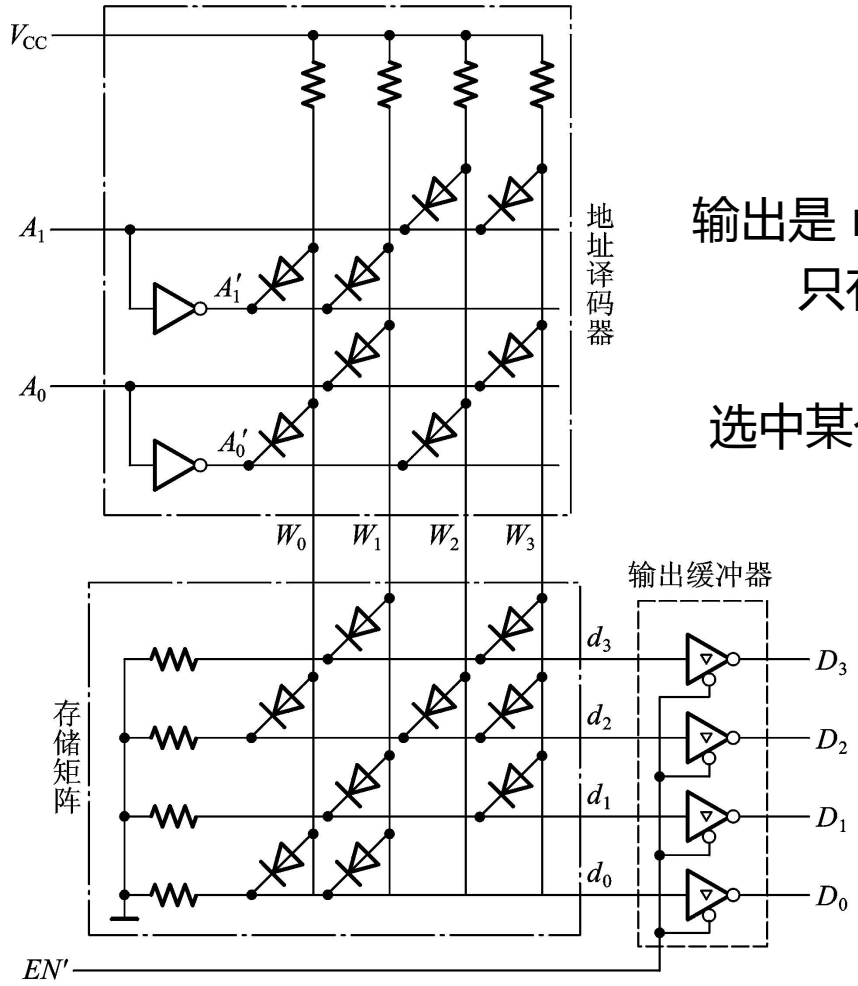
对于大容量的 ROM，1K = 1024，1M = 1024K；

例如左图就是一个 4×4 位的 ROM；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

小结



地址译码器：
与阵列，
输入是 n 位地址代码，
输出是 n 个变量的 2^n 个最小项，对应 2^n 条字线；
只有输入地址代码对应的字线为高电平，
即只选中输入地址代码对应的字；
选中某个地址 = 选中某个字 = 选中该字的所有位

存储矩阵：
或阵列，
有半导体元件表示存 1，
无半导体元件表示存 0，
有 m 行 2^n 列，
每一列对应一个字，
每一行对应一个位；

输出缓冲器：
对于 ROM ，只读不写，
输出为三态门，使能端控制；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

ROM 的类别

- 掩模 ROM：

我们前面给出的这种基本的 ROM 称为掩模 ROM，所谓的掩模 ROM 就是每个存储矩单元的数据即每个存储单元处是否有半导体元件，是厂家生产时固化在存储器内部，用户无法修改的；

- 可编程 ROM —— PROM（Programmable ROM）：

可编程 ROM 的特点是存储器中存储的数据可以由用户自己编程写入，但是只能写入一次，不能够修改；

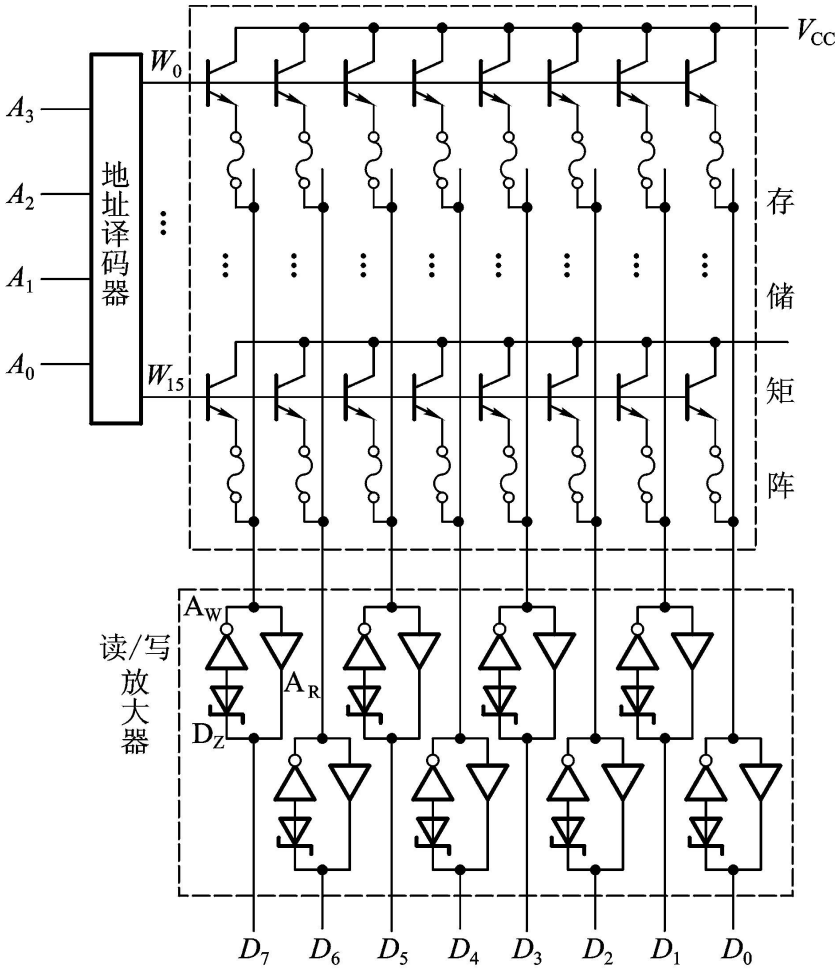
- 可擦除的可编程 ROM —— EPROM（Erasable Programmable ROM）

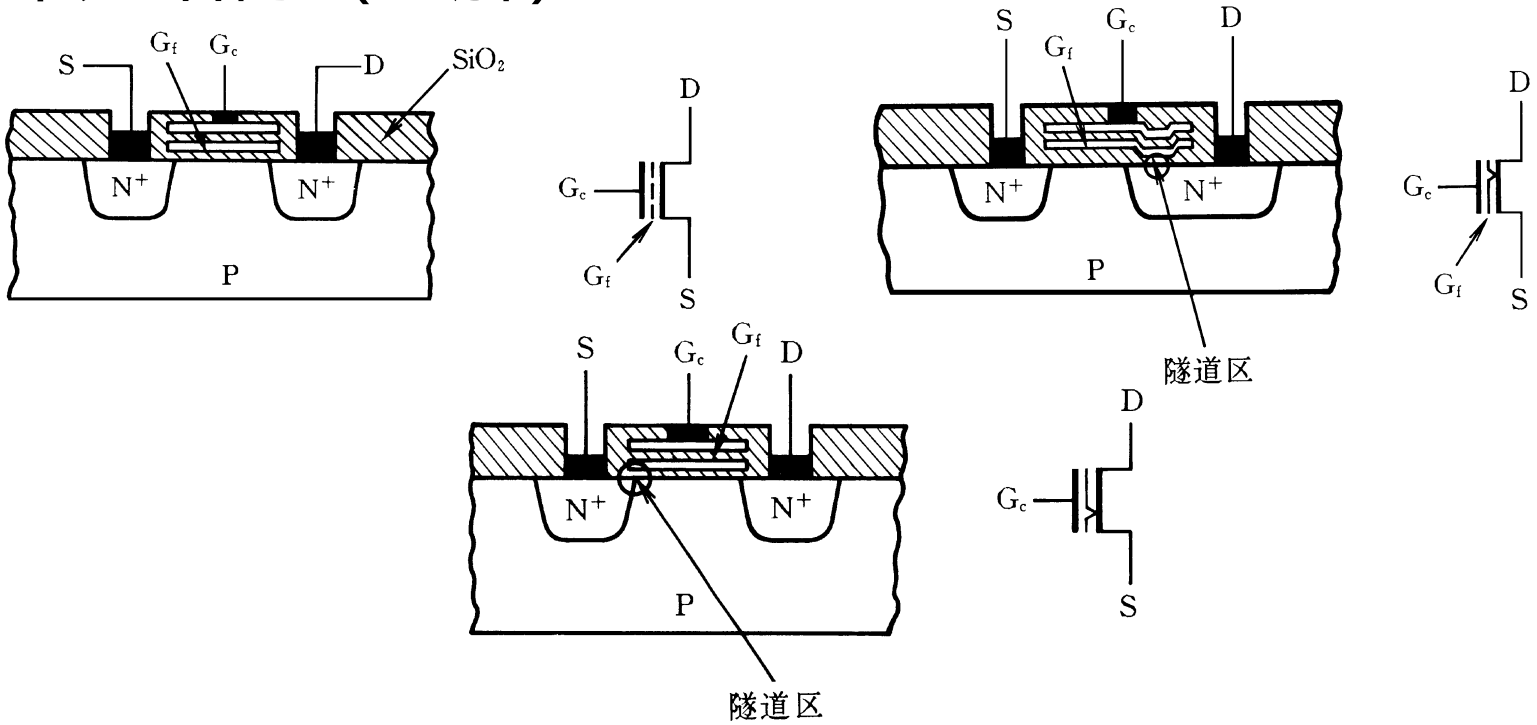
相较于 PROM，EPROM 中的数据可以擦除重新写入，用于需要经常修改 ROM中数据的场合；
特别地，EPROM 又可分为紫外线擦除的可编程 ROM —— UVE-PROM、电信号擦除的可编程 ROM —— E^2PROM、快闪存储器；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

其他类型的 ROM 的存储单元结构（了解）



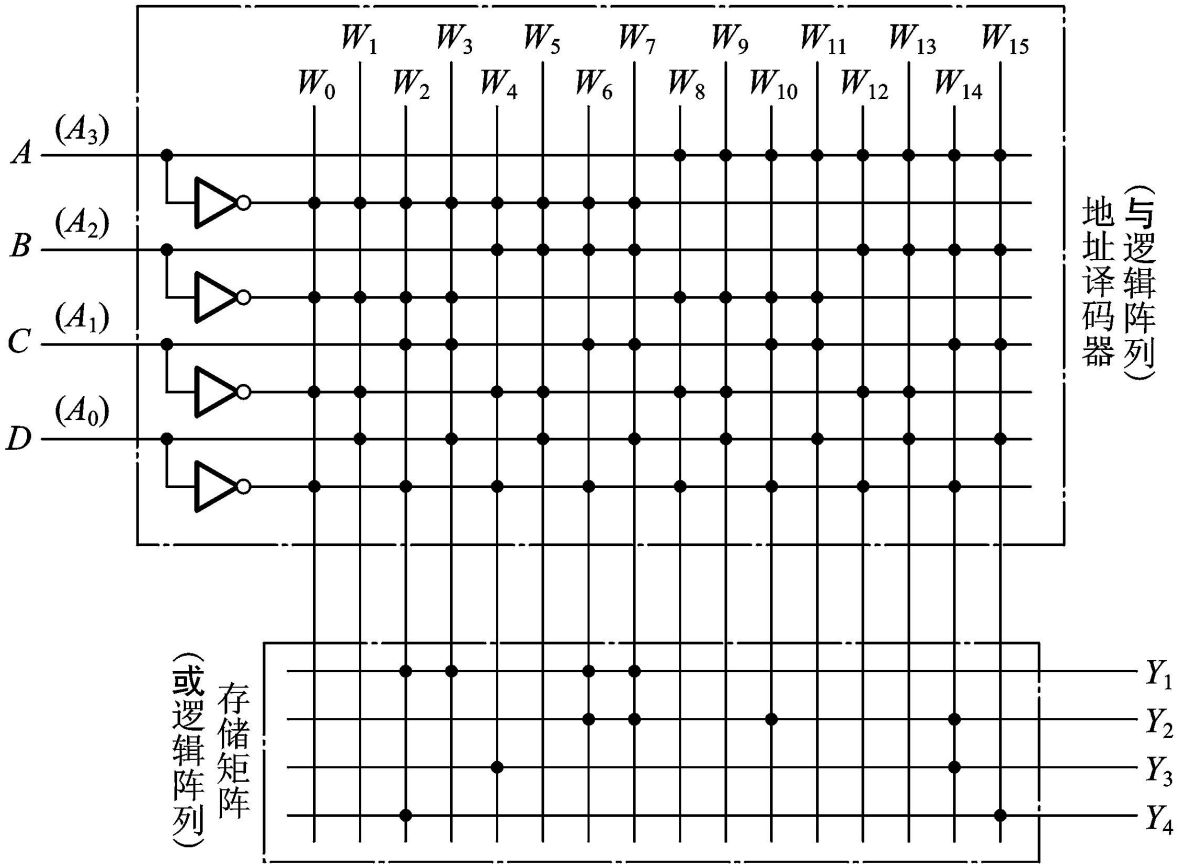


无论是哪种类型的 ROM，其整体结构框图并没有变化，区别只是在于存储矩阵中的存储单元采取了特殊工艺的半导体器件或是特殊的结构！！！

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

只读存储器 ROM

ROM 的点阵图表示方法



因为不同 ROM 的存储矩阵中的每个存储单元可以采用不同的半导体器件，因此，为了简化作图，在接入存储器件的矩阵交叉点上画一个圆点“ ·”或是一个叉“×”以代替存储器件；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

用 ROM 实现组合逻辑函数的原理

根据我们前面的分析，ROM 本质上就是一个组合逻辑电路，其地址译码器是一个与阵列，能够得到 n 位输入变量全部 2^n 个最小项；而根据第二章逻辑代数基础知识，任何一个组合逻辑函数都可以写成最小项之和的形式，而存储矩阵就是一个或阵列；因此，任何一个组合逻辑函数都可以通过向 ROM 中写入相应的数据（如果是可编程的 ROM ）实现；

事实上，换一种理解方式，事实上 ROM 可以看成是 m 个并行连接的数据选择器，它们的地址代码输入端是公共的，每个数据选择器有 n 位地址输入端，2^n 个数据输入端；第 i 个数据选择器代表每个字的 i 位；m 个数据选择器的共 $m \times 2^n$ 个数据输入端的预置数，就是 ROM 存储矩阵中每个存储单元存入的数据；根据输入的地址代码选中一个字，选中一个字就是选中该字的所有位，m 个数据选择器并行输出的就是该字的所有位；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

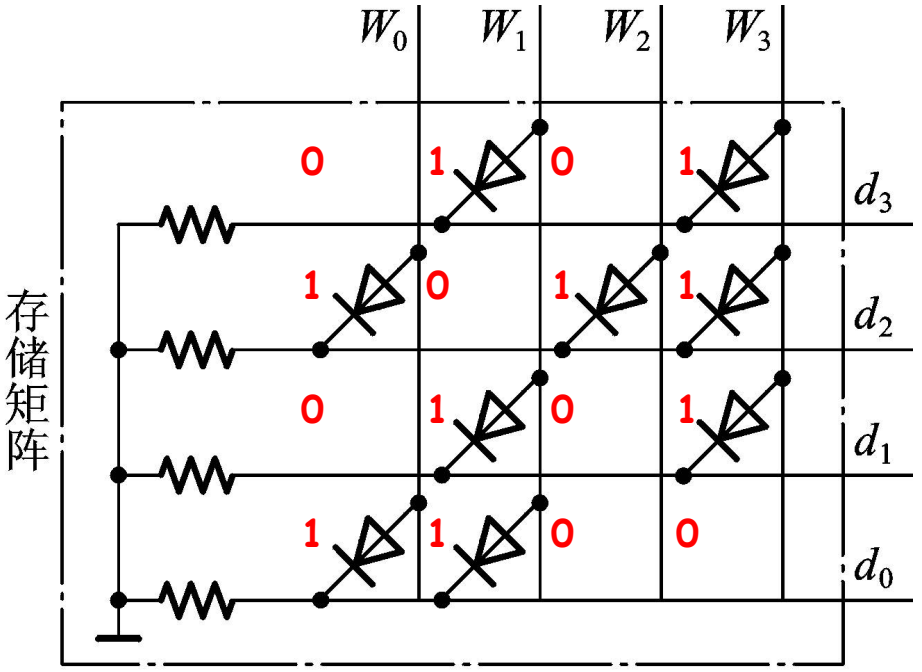
用 ROM 实现组合逻辑函数

这里假设输出缓冲级为同相输出！

存储矩阵与真值表的对应关系

根据我们前面的分析，真值表中的每一行对应的是一个最小项即一个字，共 2^n 行；真值表的输出侧每一列每一个输出变量对应的是就是一位；因此 ROM 的存储矩阵与真值表是一个行列倒置的关系（如果存储矩阵中列线为字线，行线为位线）

地址		选中的字线	数据			
A_1	A_0	W	d_3	d_2	d_1	d_0
0	0	$W_0 = 1$	0	1	0	1
0	1	$W_1 = 1$	1	0	1	1
1	0	$W_2 = 1$	0	1	0	0
1	1	$W_3 = 1$	1	1	1	0



请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

不需要去背!
通过题目练习去理解!

用 ROM 实现组合逻辑函数的步骤

Step 1：进行逻辑抽象（与一般的组合逻辑设计题目一致），得到真值表（可以写成最小项之和的形式或者画成卡诺图的形式快速得到真值表）；

Step 2：选择存储器芯片的容量，即输入端口和输出端口的个数；存储器的地址输入端应该大于等于（最好是等于）输入变量数，芯片的数据输出端应该大于等于（最好是等于）输出变量数；如果给定的芯片无法满足上述要求，则需要用多片存储器通过扩展的方法；

Step 3：将输入变量接至存储器的地址输入端，取存储器的数据输出端作为组合逻辑函数的输出端，从函数的真值表得到对应的存储器的数据表；

Step 4：将得到的数据表写入存储器中，根据“行列倒置”特点绘制存储矩阵的点阵图；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

例：

用 ROM 产生以下组合逻辑函数，列出 ROM 的数据表，画出存储矩阵的点阵图；

$$\begin{cases} Y_1 = A'BC + A'B'C \\ Y_2 = AB'CD' + BCD' + A'BCD \\ Y_3 = ABCD' + A'BC'D' \\ Y_4 = A'B'CD' + ABCD \end{cases}$$

输入变量 4 个，输出变量 4 个，选择 $2^4 \times 4$ 的 ROM；首先列出此组合逻辑函数的真值表，或是写成最小项之和的形式；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

（接上页例题）

$$\begin{cases} Y_1 = A'BC + A'B'C \\ Y_2 = AB'CD' + BCD' + A'BCD \\ Y_3 = ABCD' + A'BC'D' \\ Y_4 = A'B'CD' + ABCD \end{cases}$$

$$\begin{cases} Y_1 = \sum m(2,3,6,7) \\ Y_2 = \sum m(6,7,10,14) \\ Y_3 = \sum m(4,14) \\ Y_4 = \sum m(2,15) \end{cases}$$

地址				数据			
A	B	C	D	Y_1	Y_2	Y_3	Y_4
0	0	0	0				
0	0	0	1				
0	0	1	0	1			1
0	0	1	1	1			
0	1	0	0			1	
0	1	0	1				
0	1	1	0	1	1		
0	1	1	1	1	1		
1	0	0	0				
1	0	0	1				
1	0	1	0		1		
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0		1	1	
1	1	1	1				1

（真值表输出侧空白的即为 0）

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

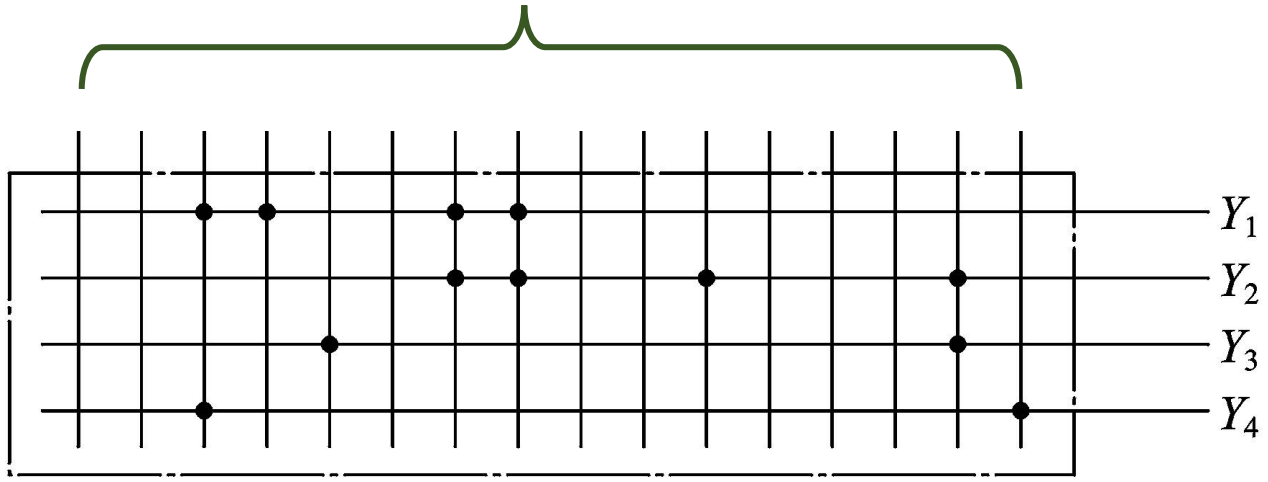
用 ROM 实现组合逻辑函数

(接上页例题)

地址				数据			
A	B	C	D	Y_1	Y_2	Y_3	Y_4
0	0	0	0				
0	0	0	1				
0	0	1	0	1			1
0	0	1	1	1			
0	1	0	0			1	
0	1	0	1				
0	1	1	0	1	1		
0	1	1	1	1	1		
1	0	0	0				
1	0	0	1				
1	0	1	0		1		
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0		1	1	
1	1	1	1				1

倒置

字线 W_0 ~ W_{15}
实质上就是最小项 m_0 ~ m_{15}



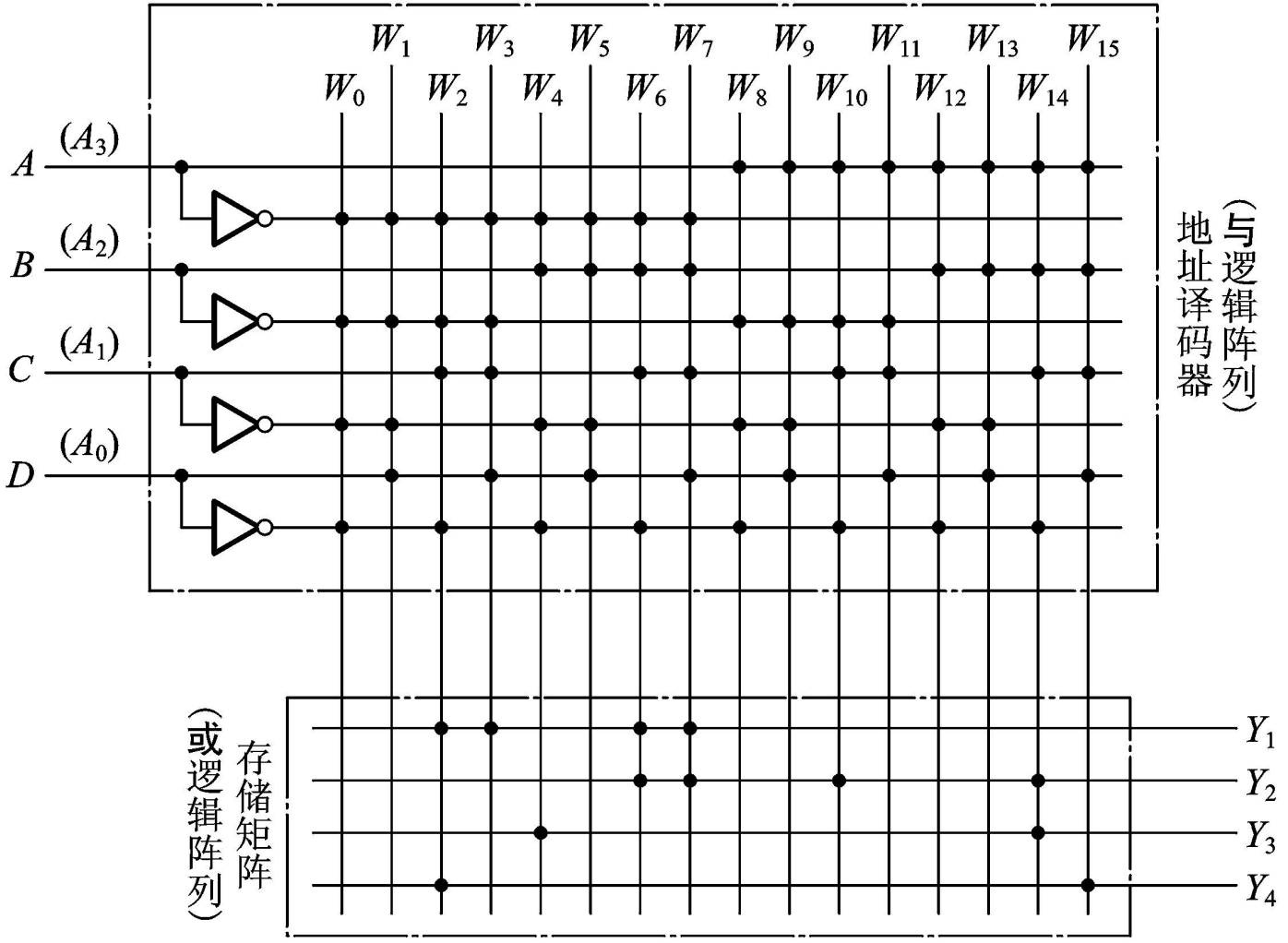
存储矩阵点阵图
(打点即为 1，不打点即为 0)

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

（接上页例题）

本题最终的结果如右图所示；
（这里没有考虑输出缓冲器）



请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

变式：

变式 1：如果对于前面的例题，给定的 ROM 的容量是 256 × 8 ，如何修改设计?

只需要选择 8 个输入端口的任意 4 个作为输入变量的输入端，其余不用的输入端输入 0 即可；选择 8 个输出端口的任意 4 个作为输出变量的输出端；设计方案不唯一，区别只是在于地址代码不同即被选中的字线不同，对应到数据表即可；（一般先用低位）

变式 2：如果对于前面的例题，解题时发现真值表输出侧“ 1 多 0 少”，可以如何优化设计?

可以在后级使用一个反相的输出缓冲器（三态门），前面存储矩阵中为 0 的打点，为 1 的不打点；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

例：

已知某 ROM 逻辑电路图如下，请写出此 ROM 实现的组合逻辑函数最简与或式；

16 条字线对应的是 16 个最小项；

存储矩阵是或阵列；

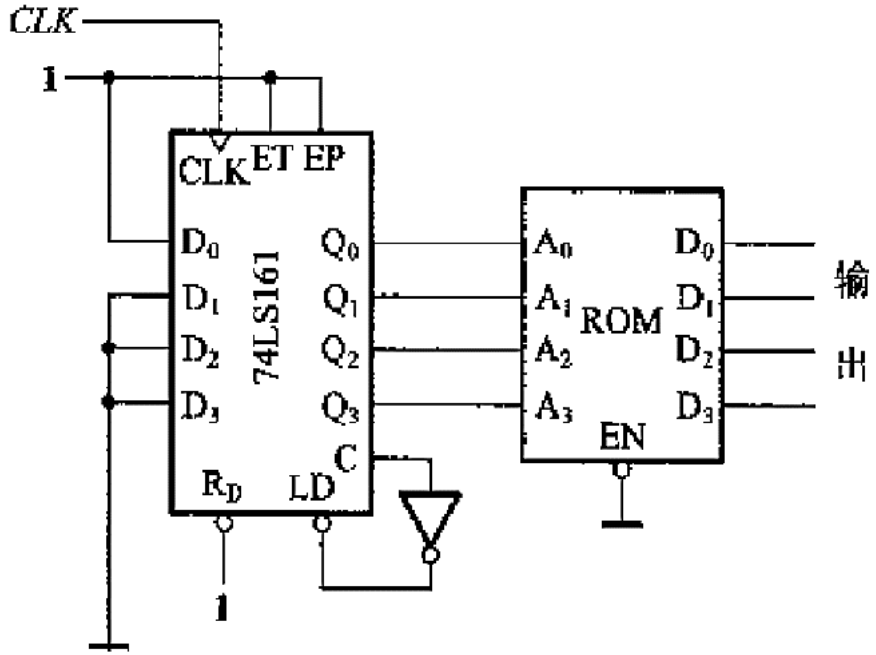
将 D_3、D_2、D_1、D_0 根据存储矩阵点阵图写成最小项之和的形式，再用卡诺图化简法进行化简即可；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

例：

已知用 16× 4 位 ROM 和同步十六进制加法计数器 74LS161 组成的脉冲分配电路如下，ROM 的数据表如下表所示（在这里没有给出，本题只强调思路）；试画出在 CLK 信号连续作用下 D_3、D_2、D_1、D_0 的输出波形；



74161 通过置数法设计为十五进制的计数器；
其状态在 0001-1111 之间循环变化；

ROM 的地址输入端在 0001-1111 之间变化，也就选择字线 1 ~ 字线 15 的数据输出，即并行输出数据表中除了 $A_3A_2A_1A_0$ = 0000 一行对应的数据；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

用 ROM 实现组合逻辑函数

习题小结：

用 ROM 实现组合逻辑函数的习题类型无非就是设计（给定组合逻辑函数，或者是实际应用背景中的逻辑功能，求解数据表和绘制点阵图）和分析（给定 ROM 点阵图，求出组合逻辑函数式），事实上只要理解一些基本的概念和原理，例如译码器（与阵列）引出的 2^n 条字线对应的是 2^n 个最小项、存储矩阵是一个或阵列、存储矩阵数据表与真值表是倒置的关系等，这种类型题就非常容易；因为 ROM 实质上是一个组合逻辑电路，所以理论上在第四章组合逻辑电路中的任何一道设计题都可以用 ROM 实现（一般难度并不高）；除此之外，可能会与其他章节例如时序逻辑电路、A/D 和 D/A 转换等联系考察综合性题目；

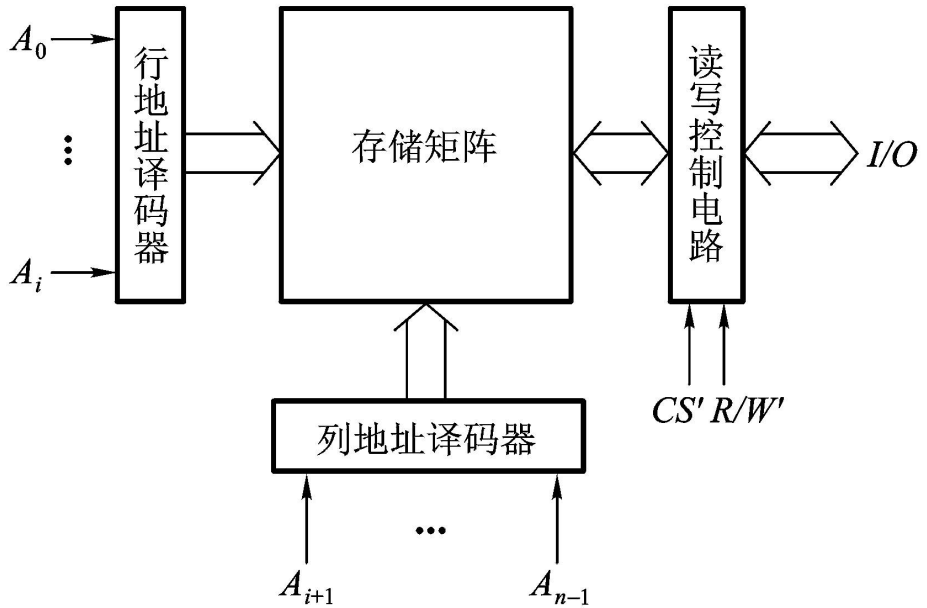
参考练习题目：

教材 P383~385 习题 7.7，7.8，7.9，7.10，7.11，7.12，7.14；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

随机存储器 RAM

RAM 的结构框图



- 行地址译码器和列地址译码器：对于容量较大的存储器，为了减少字线数目，可以将一个 n 位地址代码的地址译码器分为行地址译码器和列地址译码器；
- 存储矩阵：由基本的存储单元排列而成，存储单元使用的不同工艺、不同性质的器件决定了存储器的类型和工作特点；
- 读写控制电路：根据控制端控制数据读取（输出）或写入（输入）；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

随机存储器 RAM

行地址译码器和列地址译码器

事实上，ROM 中也可以使用这种行地址译码器和列地址译码器的设计方式；只是通常的习题中几乎不会考察行、列译码形式的 ROM ；

行地址译码器和列地址译码器就是将原本的 n 位地址译码器分为 n_1 位的行地址译码器和 n_2 位的列地址译码器，存储矩阵中的某一个字是否被选中，由行地址译码的结果和列地址译码的结果共同决定；这样设计的优点在于节省了字线的数量；例如对于一个 $n = 10$ 的存储器，如果使用传统的单译码方式，则其地址译码器会输出 $2^{10} = 1024$ 个字线；如果将 $n = 10$ 分为 $n_1 = 6$ 的行地址译码器和 $n_2 = 4$ 的列地址译码器，这样只需要有 $2^6 + 2^4 = 80$ 条字线；

这种行译码和列译码方式与后面单片机课程中的“矩阵键盘”原理类似；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

随机存储器 RAM

RAM 的存储矩阵

RAM 与 ROM 存储矩阵的区别就是存储矩阵的内部结构即存储单元不同；RAM 的存储单元是触发器（SRAM）或动态 MOS 存储单元（DRAM），而 ROM 的存储矩阵是或阵列；本质上 RAM 属于时序逻辑电路，而 ROM 属于组合逻辑电路；

至于 SRAM 和 DRAM 的存储单元结构和原理，不作要求；

RAM 的读写控制电路

RAM 相较于 ROM 的区别在于其即可以读也可以写，所以其输出级是一个双向的读写控制电路，由读写控制端 R/W’ 控制；除此之外，类似于 ROM 的使能端，读写控制电路存在一个片选端 CS’，低电平有效，当 CS’ = 1 时输出被封锁在高阻态；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

随机存储器 RAM

RAM 与 ROM 的比较

相同点：整体的结构框图类似，都含有地址译码器和存储矩阵，寻址的原理都相同；

不同点：

- 存储矩阵的内部结构即存储单元不同：ROM 本质上是组合逻辑电路，其存储矩阵是或阵列；而 RAM 本质上是时序逻辑电路，其存储矩阵由触发器或动态存储单元构成；
- ROM 工作时只能读取不能写入，而 RAM 工作时可以读也可以写，由读写控制端控制；即体现在电路的输出级不同；
- ROM 断电后数据不会丢失，而 RAM 断电后数据会丢失；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

存储器容量的扩展

存储器容量扩展的原理

存储器容量的扩展主要分为两种情况：

① 位扩展：当每一片 ROM 或 RAM 的字数够用而位数不够用，则只需要增加存储器的数量，多个存储器使用相同的地址输入端以及附加端（使能端、片选端、读写控制端等），即每一片的地址线、读写控制线、片选线分别并接即可；

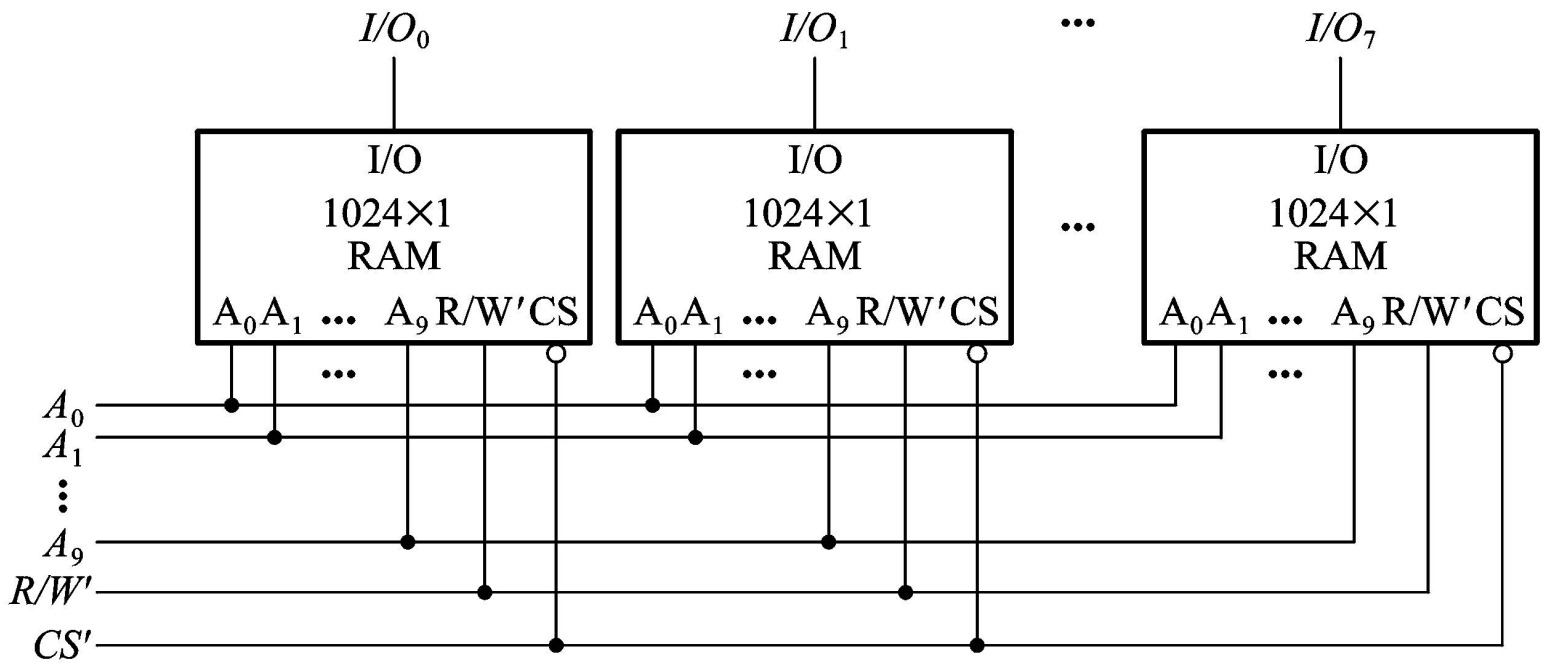
② 字扩展：当每一片 ROM 或 RAM 的位数够用而字数不够用，则需要增加地址代码的位数；低位的地址线仍然并接至每一片的地址输入端；而高位的地址线，需要通过一个译码器译为独立的高低电平，片选每个存储器，即连接至每一片的片选控制端（使能端）；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

存储器容量的扩展

位扩展方式

下面是将 8 片 1024× 1 扩展为 1024× 8；非常容易分析和设计；

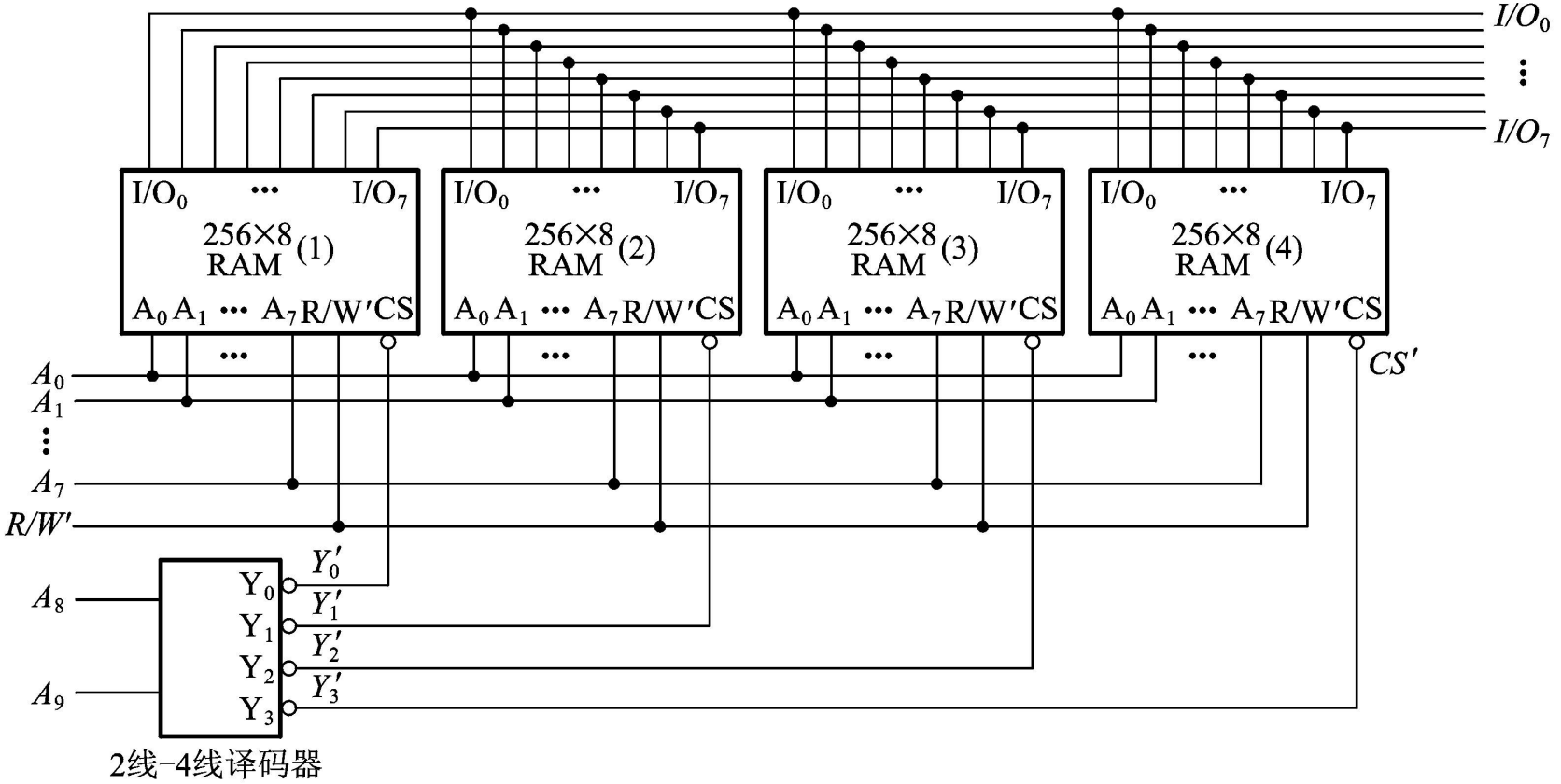


请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

存储器容量的扩展

字扩展方式

下面是将 4 片 256 × 8 扩展为 1024× 8;



请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

存储器容量的扩展

字扩展方式

字扩展方式并不是唯一的，如果按照前一页，用 A_9、A_8 的二位代码状态片选各个器件，而 A_7 ~ A_0 作为公共的地址输入端的话，每一片覆盖的地址仍然是 2^8 = 256 个，每一片的地址范围应该是 00xxxxxxxx、 01xxxxxxxx、 10xxxxxxxx、 11xxxxxxxx，也就是对应 0 ~ 255，256 ~ 511，512 ~ 767，768 ~ 1023；

如果用 A_1、A_0 的二位代码状态片选各个器件，而 A_9 ~ A_2 作为公共的地址输入端的话，则每一片覆盖的地址仍然是 2^8 = 256 个，每一片的地址范围是 xxxxxxxx00、 xxxxxxxx01、xxxxxxxx10、 xxxxxxxx11，xxxxxxxx 为 00000000~11111111，最终的地址范围与前面的结果不同；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

存储器容量的扩展

自测：

（1）用两片 1024 × 8 位的 ROM 组成 1024 × 16 位的存储器；

位扩展；地址线、片选线直接并行连接即可；

（2）用四片 1024 × 4 位的 RAM 和 3/8 译码器 74LS138 组成 4K × 4 位的存储器，并写出每一片的寻址范围；

字扩展；4 K = 4 × 1024 = 4096 ，可以选择 A_{11}、A_{10} 作为译码器的输入；译码器只需用 2 个输入端和 4 个输出端即可；译码器输出连接每一片的片选端；低位地址线和读写控制端公用；

（3）用十六片 1024 × 4 位的 RAM 和 3/8 译码器 74LS138 组成 8K × 8 位的存储器，并写出每一片的寻址范围；

先位扩展，再字扩展；（有思路即可，作为考试题概率较小）

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

存储器容量的扩展

习题小结：

参考练习题目：

教材 P383~385 习题 7.3，7.4，7.5，7.6；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com

半导体存储器——小结

- 半导体存储器的分类；
- 半导体存储器的基本结构；
- ROM 的工作原理；
- 用 ROM 实现组合逻辑函数；
- RAM 与 ROM 的比较；
- 存储器容量的扩展；

请尊重个人版权，请勿上传至其他公共平台，谢谢理解！
内容问题可联系 bow33@163.com