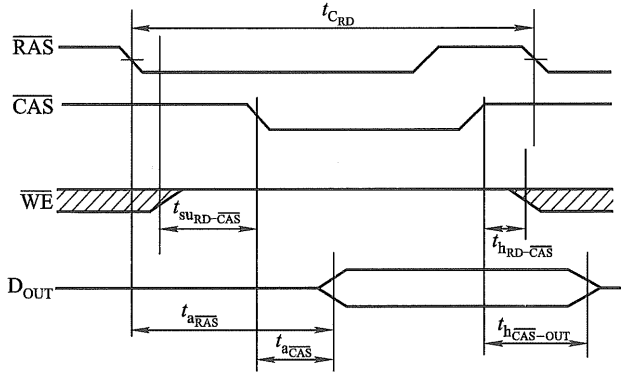


图 4.22 动态 RAM 读工作方式时序图

2）写时序

在写工作方式时（写允许 $\overline{WE}=0$），$\overline{RAS}$ 的一个周期 $t_{C_{WR}}$ 即为写工作周期，如图 4.23 所示。

为了确保写入数据准确无误，$\overline{WE}=0$ 应先于 $\overline{CAS}=0$，而且数据的有效存在时间应与 $\overline{CAS}$ 及 $\overline{WE}$ 的有效相对应，即写入数据应在 $\overline{CAS}$ 有效前的一段时间 $t_{su_{DIN-\overline{CAS}}}$ 出现，它的保持时间应为 $\overline{CAS}$ 有效后的一段时间 $t_{h_{DIN-\overline{CAS}}}$，这是因为数据的写入实际上是由 $\overline{CAS}$ 的下降沿激发而成的。可见，为了保证正常写入，$\overline{WE}$、$\overline{CAS}$ 有效均要大于数据 D_{IN} 有效的时间。

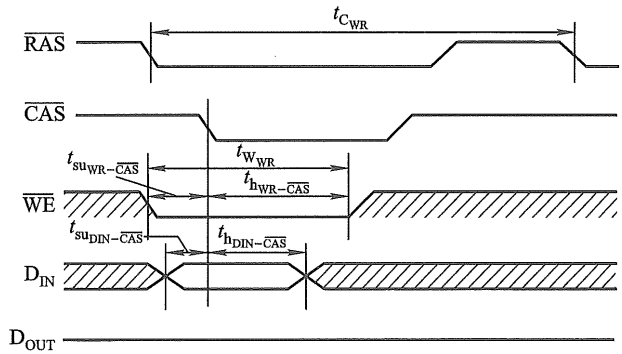


图 4.23 动态 RAM 写工作方式时序图

此外，动态 RAM 还有读-改写工作方式和页面工作方式，本书不再赘述。

（4）动态 RAM 的刷新

刷新的过程实质上是先将原存信息读出，再由刷新放大器形成原信息并重新写入的再生过程（图 4.19 中的刷新放大器及图 4.21 中的读放大器均起此作用）。

由于存储单元被访问是随机的，有可能某些存储单元长期得不到访问，不进行存储器的读/写操作，其存储单元内的原信息将会慢慢消失。为此，必须采用定时刷新的方法，它规定在一定的时间内，对动态 RAM 的全部基本单元电路必作一次刷新，一般取 2 ms，这个时间称为刷新周期，又称再生周期。刷新是一行行进行的，必须在刷新周期内，由专用的刷新电路来完成对基本单元电路的逐行刷新，才能保证动态 RAM 内的信息不丢失。通常有三种方式刷新：集中刷新、分散刷新和异步刷新。

1）集中刷新

集中刷新是在规定的一个刷新周期内，对全部存储单元集中一段时间逐行进行刷新，此刻必须停止读/写操作。例如，对 128×128 矩阵的存储芯片进行刷新时，若存取周期为 0.5 μs，刷新周期为 2 ms（占 4 000 个存取周期），则对 128 行集中刷新共需 64 μs（占 128 个存取周期），其余的 1 936 μs（共 3 872 个存取周期）用来读/写或维持信息，如图 4.24 所示。由于在这 64 μs 时间内不能进行读/写操作，故称为“死时间”，又称访存“死区”，所占比率为 128/4 000×100% = 3.2%，称为死时间率。

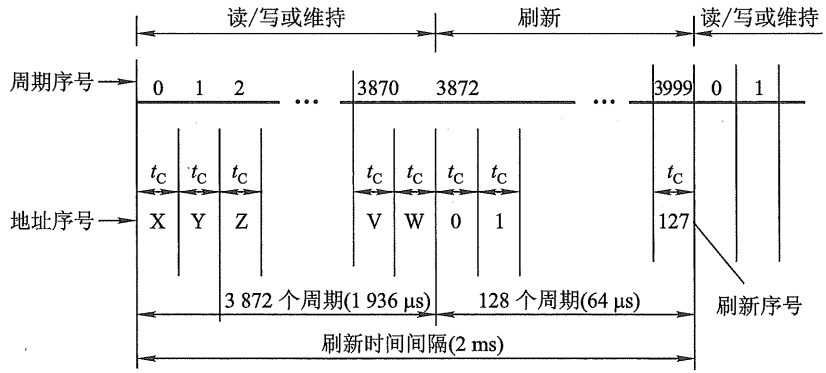


图 4.24　集中刷新时间分配示意图

2）分散刷新

分散刷新是指对每行存储单元的刷新分散到每个存取周期内完成。其中，把机器的存取周期 t_C 分成两段，前半段 t_M 用来读/写或维持信息，后半段 t_R 用来刷新，即 $t_C = t_M + t_R$。若读/写周期为 0.5 μs，则存取周期为 1 μs。仍以 128×128 矩阵的存储芯片为例，刷新按行进行，每隔 128 μs就可将存储芯片全部刷新一遍，如图 4.25 所示。这比允许的间隔 2 ms 要短得多，而且也不存在停止读/写操作的死时间，但存取周期长了，整个系统速度降低了。

3）异步刷新

异步刷新是前两种方式的结合，它既可缩短“死时间”，又充分利用最大刷新间隔为 2 ms 的特点。例如，对于存取周期为 0.5 μs，排列成 128×128 的存储芯片，可采取在 2 ms 内对 128 行各

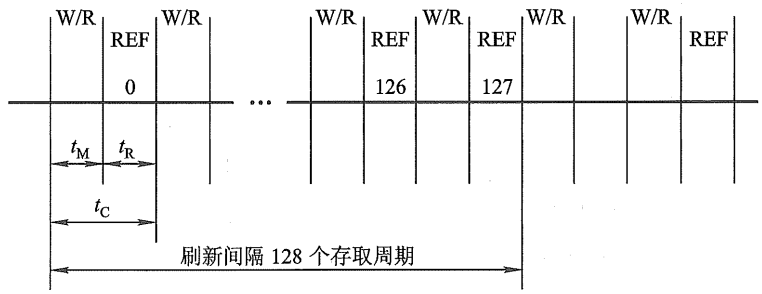


图 4.25 分散刷新时间分配示意图

刷新一遍,即每隔 15.6 μs(2 000 μs÷128≈15.6 μs)刷新一行,而每行刷新的时间仍为0.5 μs,如图 4.26 所示。这样,刷新一行只停止一个存取周期,但对每行来说,刷新间隔时间仍为 2 ms,而"死时间"缩短为 0.5 μs。

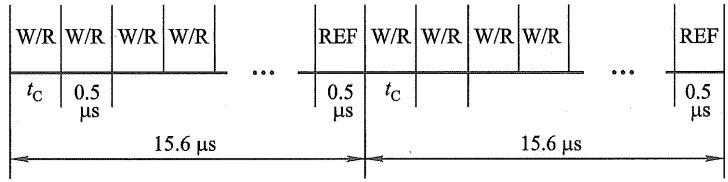


图 4.26 异步刷新时间分配示意图

如果将动态 RAM 的刷新安排在 CPU 对指令的译码阶段,由于这个阶段 CPU 不访问存储器,所以这种方案既克服了分散刷新需独占 0.5 μs 用于刷新,使存取周期加长且降低系统速度的缺点,又不会出现集中刷新的访存"死区"问题,从根本上提高了整机的工作效率。

3. 动态 RAM 与静态 RAM 的比较

目前,动态 RAM 的应用比静态 RAM 要广泛得多。其原因如下:

① 在同样大小的芯片中,动态 RAM 的集成度远高于静态 RAM,如动态 RAM 的基本单元电路为一个 MOS 管,静态 RAM 的基本单元电路可为 4~6 个 MOS 管。

② 动态 RAM 行、列地址按先后顺序输送,减少了芯片引脚,封装尺寸也减少。

③ 动态 RAM 的功耗比静态 RAM 小。

④ 动态 RAM 的价格比静态 RAM 的价格便宜。当采用同一档次的实现技术时,动态 RAM 的容量大约是静态 RAM 容量的 4~8 倍,静态 RAM 的存取周期比动态 RAM 的存取周期快 8~16 倍,但价格也贵 8~16 倍。

随着动态 RAM 容量不断扩大,速度不断提高,它被广泛应用于计算机的主存。

动态 RAM 也有缺点:

①由于使用动态元件(电容),因此它的速度比静态 RAM 低。

②动态 RAM 需要再生,故需配置再生电路,也需要消耗一部分功率。通常,容量不大的高

速缓冲存储器大多用静态 RAM 实现。

4.2.4　只读存储器

按 ROM 的原始定义，一旦注入原始信息即不能改变，但随着用户的需要，总希望能任意修改 ROM 内的原始信息。这便出现了 PROM、EPROM 和 EEPROM 等。

对半导体 ROM 而言，基本器件为两种：MOS 型和 TTL 型。

1. 掩模 ROM

图 4.27 所示为 MOS 型掩模 ROM，其容量为 1K×1 位，采用重合法驱动，行、列地址线分别经行、列译码器，各有 32 根行、列选择线。行选择线与列选择线交叉处既可有耦合元件 MOS 管，也可没有。列选择线各控制一个列控制管，32 个列控制管的输出端共连一个读放大器。当地址为全“0”时，第 0 行、0 列被选中，若其交叉处有耦合元件 MOS 管，因其导通而使列线输出为低电平，经读放大器反相为高电平，输出“1”。当地址 $A_4 \sim A_0$ 为 11111，$A_9 \sim A_5$ 为 00000 时，即第 31 行、第 0 列被选中，但此刻行、列的交叉处无 MOS 管，故 0 列线输出为高电平，经读放大器反相为“0”输出。可见，用行、列交叉处是否有耦合元件 MOS 管，便可区分原存“1”还是存“0”。当然，此 ROM 制成后不可能改变原行、列交叉处的 MOS 管是否存在，所以，用户是无法改变原始状态的。

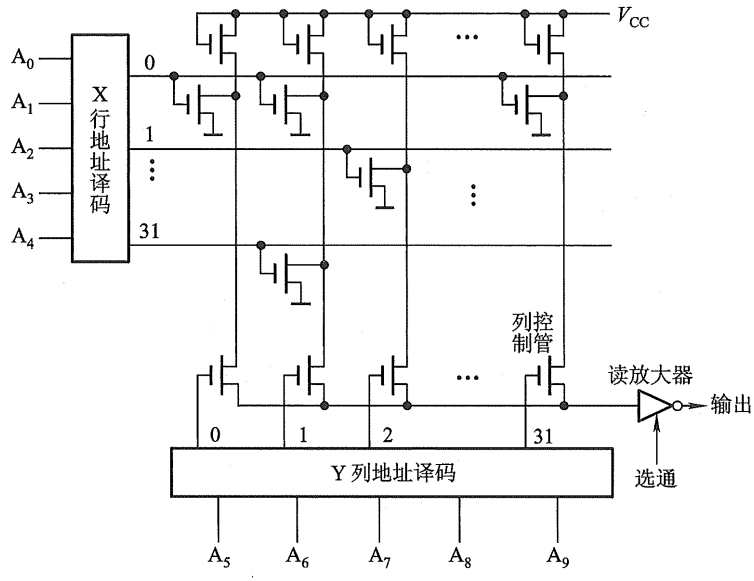


图 4.27　1 K×1 位的 MOS 管掩模 ROM

2. PROM

PROM 是可以实现一次性编程的只读存储器，图 4.28 示意一个由双极型电路和熔丝构成的基本单元电路。在这个电路中，基极由行线控制，发射极与列线之间形成一条镍铬合金薄膜制成的熔丝（可用光刻技术实现），集电极接电源 V_{CC}。熔丝断和未断可区别其所存信息是“1”或“0”。

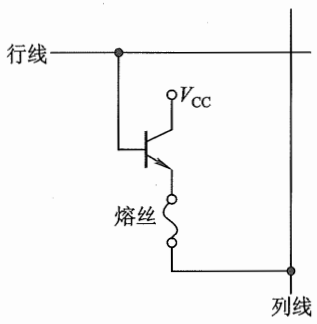


图 4.28 双极型镍铬熔丝式单元电路

图 4.29 是由图 4.28 所示基本单元电路构成的 16×1 位双极型镍铬熔丝式 PROM 芯片。用户在使用前，可按需要将信息存入行、列交叉的耦合元件内。若欲存“0”，则置耦合元件一大电流，将熔丝烧掉。若欲存“1”，则耦合处不置大电流，熔丝不断。当被选中时，熔丝断掉处将读出“0”，熔丝未断处将读出“1”。例如，当地址 $A_3 \sim A_0$ 为 0000 时，第 0 行、第 0 列被选中，此刻行、列交叉的耦合元件熔丝未断，故读出 D = 1；若 $A_3 \sim A_0 = 0001$，则第 1 行、第 0 列被选中，此刻行、列交叉的耦合元件熔丝已断，读出 D = 0。当然，已断的熔丝是无法再恢复的，故这种 ROM 往往只能实现一次编程，不得再修改。

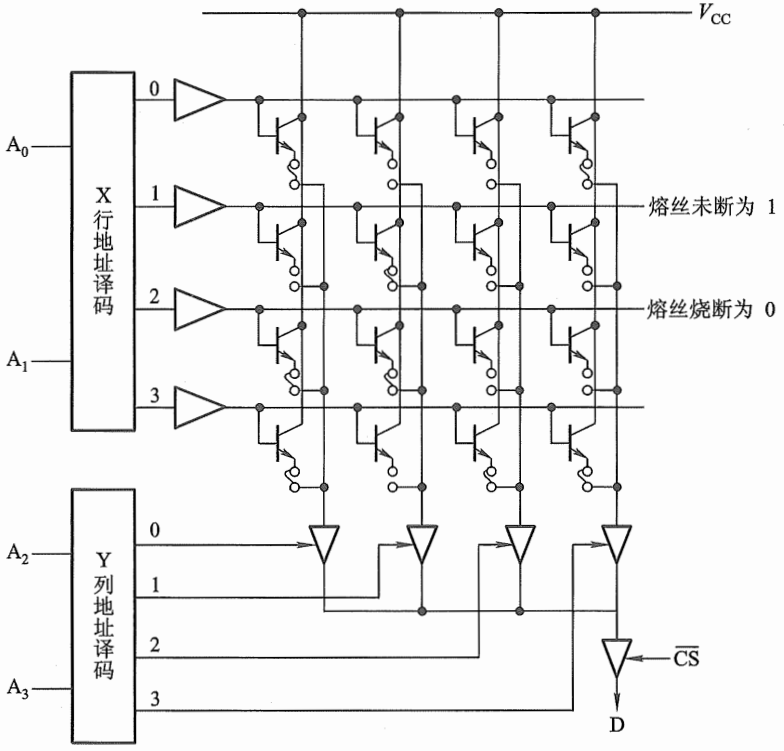


图 4.29 16×1 位双极型镍铬熔丝式 PROM

3. EPROM

EPROM 是一种可擦除可编程只读存储器。它可以由用户对其所存信息作任意次的改写。目前用得较多的 EPROM 是由浮动栅雪崩注入型 MOS 管构成的，又称 FAMOS 型 EPROM，如图 4.30所示。

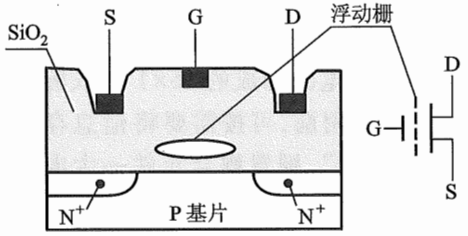
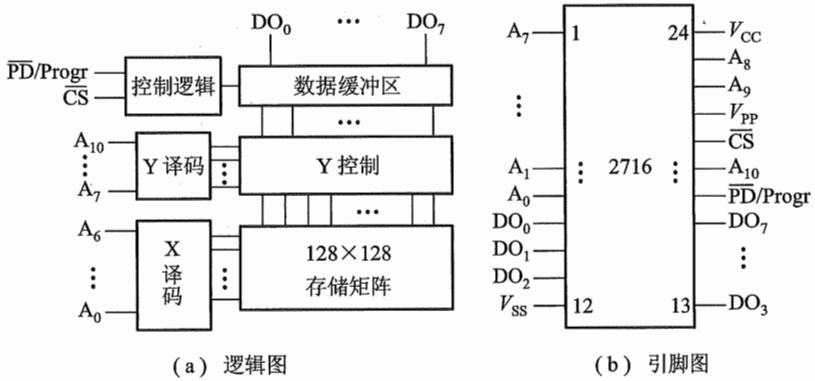


图 4.30　N 型沟道浮动栅 MOS 电路

图中所示的 N 型沟道浮动栅 MOS 电路，在漏端 D 加上正电压（如 25 V、50 ms 宽的正脉冲），便会形成一个浮动栅，它阻止源 S 与漏 D 之间的导通，致使此 MOS 管处于“0”状态。若对 D 端不加正电压，则不能形成浮动栅，此 MOS 管便能正常导通，呈“1”状态。由此，用户可按需要对不同位置的 MOS 管 D 端施正电压或不施正电压，便制成了用户所需的 ROM。一旦用户需重新改变其状态，可用紫外线照射，驱散浮动栅，再按需要将不同位置的 MOS 管 D 端重新置于正电压，又得出新状态的 ROM，故称之为 EPROM。

图 4.31 为 2716 型 EPROM 的逻辑图和引脚图。



（a）逻辑图　　（b）引脚图

图 4.31　2716 型 EPROM 逻辑图及引脚图

这类芯片的外引脚除地址线、数据线外，还有两个电源引出头 V_{CC} 和 V_{PP}。其中 V_{CC} 接 +5 V；V_{PP} 平时接 +5 V，当其接+25 V 时用来完成编程。V_{SS} 为地。$\overline{CS}$ 为片选端，读出时为低电平，编程

写入时为高电平。$\overline{PD}$/Progr 是功率下降/编程输入端,在读出时为低电平;当此端为高电平时,可以使 EPROM 功耗由 525 mW 降至 132 mW;当需编程时,此端需加宽度为 50~55 ms、+5 V的脉冲。

EPROM 的改写可用两种方法,一种用紫外线照射,但擦除时间比较长,而且不能对个别需改写的单元进行单独擦除或重写。另一种方法用电气方法将存储内容擦除,再重写。甚至在联机条件下,用字擦除方式或页擦除方式,既可局部擦写,又可全部擦写,这种 EPROM 就是 EEPROM。

进入到 20 世纪 80 年代,又出现了一种闪速存储器(Flash Memory),又称快擦型存储器,它是在 EPROM 和 EEPROM 工艺基础上产生的一种新型的、具有性能价格比更好、可靠性更高的可擦写非易失性存储器。它既有 EPROM 的价格便宜、集成度高的优点,又有 EEPROM 电可擦除重写的特性。它具有整片擦除的特点,其擦除、重写的速度快。一块 1 M 位的闪速存储芯片的擦除、重写时间小于 5 μs,比一般标准的 EEPROM 快得多,已具备了 RAM 的功能,可与 CPU 直接连接。它还具有高速编程的特点,例如,采用快速脉冲编程算法对 28F256 闪速存储芯片每字节的编程时间仅需 100 μs。此外,该器件具有存储器访问周期短,功耗低及与计算机接口简单等优点。

在需要周期性地修改存储信息的应用场合,闪速存储器是一个极为理想的器件,因为它至少可以擦写/编程 10 000 次,这足以满足用户的需要。它比较适合于作为一种高密度、非易失的数据采集和存储器件,在便携式计算机、工控系统及单片机系统中得到大量应用,近年来已用于微型计算机中存放输入输出驱动程序和参数等。

非易失性、长期反复使用的大容量闪速存储器还可替代磁盘,例如,在笔记本手掌型袖珍计算机中都大量采用闪速存储器做成固态盘替代磁盘,使计算机平均无故障时间大大延长,功耗更低,体积更小,消除了机电式磁盘驱动器所造成的数据瓶颈。

4.2.5 存储器与 CPU 的连接

1. 存储容量的扩展

由于单片存储芯片的容量总是有限的,很难满足实际的需要,因此,必须将若干存储芯片连在一起才能组成足够容量的存储器,称为存储容量的扩展,通常有位扩展和字扩展。

(1) 位扩展

位扩展是指增加存储字长,例如,2 片 1 K×4 位的芯片可组成 1 K×8 位的存储器,如图 4.32 所示。图中 2 片 2114 的地址线 $A_9 \sim A_0$、$\overline{CS}$、$\overline{WE}$ 都分别连在一起,其中一片的数据线作为高 4 位 $D_7 \sim D_4$,另一片的数据线作为低 4 位 $D_3 \sim D_0$。这样,便构成了一个 1 K×8 位的存储器。

又如,将 8 片 16 K×1 位的存储芯片连接,可组成一个 16 K×8 位的存储器,如图 4.33 所示。

(2) 字扩展

字扩展是指增加存储器字的数量。例如,用 2 片 1 K×8 位的存储芯片可组成一个 2 K×8 位

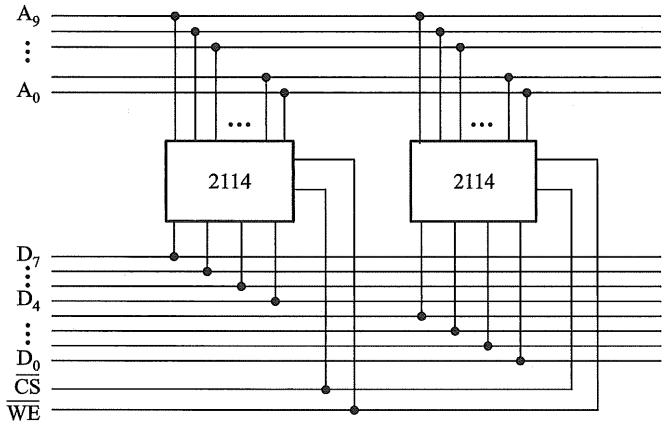


图 4.32　由 2 片 1 K×4 位的芯片组成 1 K×8 位的存储器

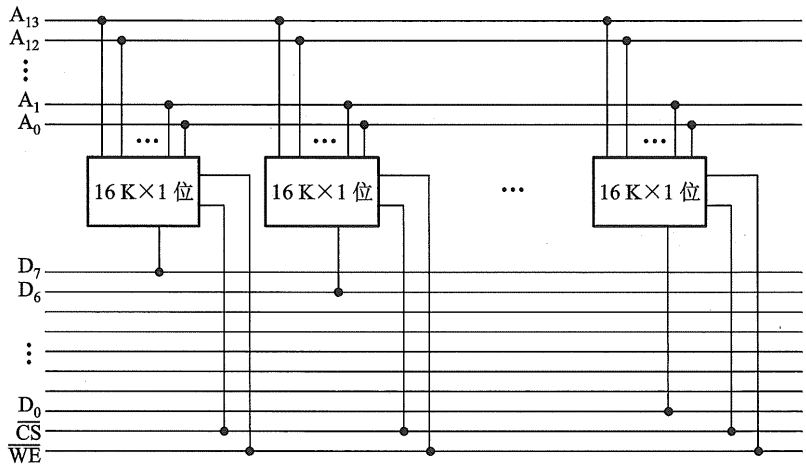


图 4.33　由 8 片 16 K×1 位的芯片组成 16 K×8 位的存储器

的存储器，即存储字增加了一倍，如图 4.34 所示。

在此，将 A_{10}用作片选信号。由于存储芯片的片选输入端要求低电平有效，故当 A_{10}为低电平时，$\overline{CS_0}$ 有效，选中左边的 1 K×8 位芯片；当 A_{10}为高电平时，反相后 $\overline{CS_1}$ 有效，选中右边的1 K×8 位芯片。

（3）字、位扩展

字、位扩展是指既增加存储字的数量，又增加存储字长。图 4.35 示意用 8 片 1 K×4 位的芯片组成 4 K×8 位的存储器。

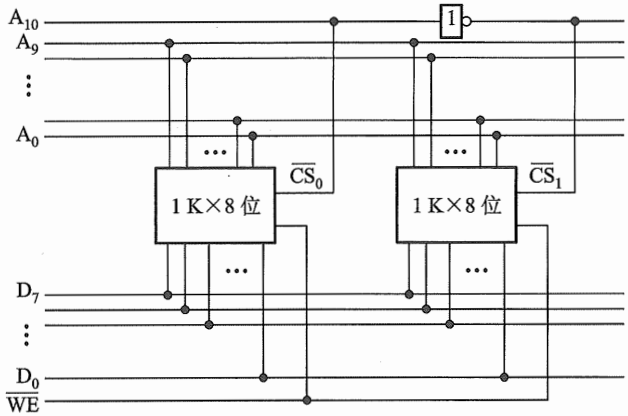


图 4.34　由 2 片 1 K×8 位的芯片组成 2 K×8 位的存储器

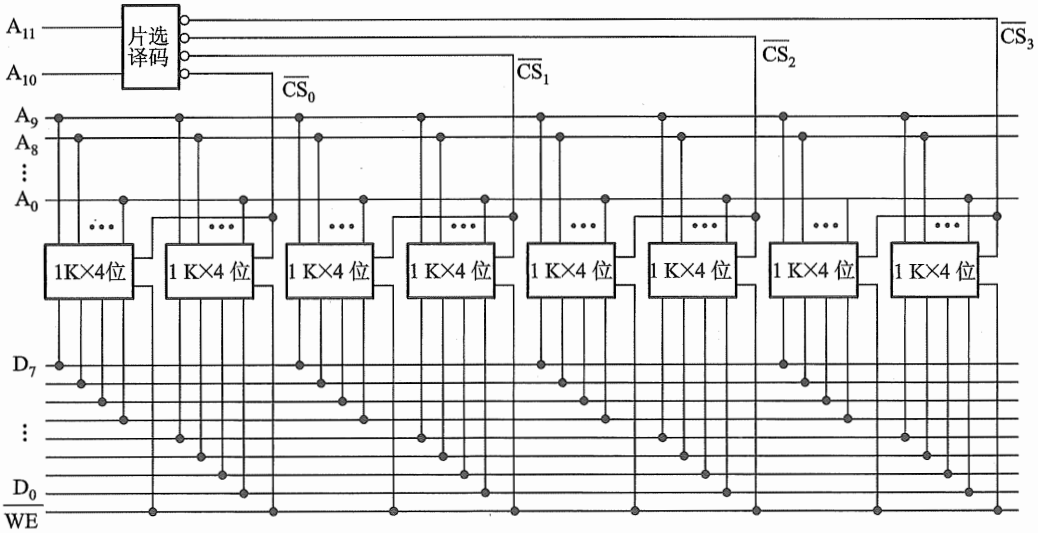


图 4.35　由 8 片 1 K×4 位的芯片组成 4 K×8 位的存储器

由图中可见，每 2 片构成一组 1 K×8 位的存储器，4 组便构成 4 K×8 位的存储器。地址线 A_{11}、A_{10}经片选译码器得到 4 个片选信号 $\overline{CS_0}$、$\overline{CS_1}$、$\overline{CS_2}$、$\overline{CS_3}$，分别选择其中 1 K×8 位的存储芯片。$\overline{WE}$ 为读/写控制信号。

2. 存储器与 CPU 的连接

存储芯片与 CPU 芯片相连时，特别要注意片与片之间的地址线、数据线和控制线的连接。

（1）地址线的连接

存储芯片的容量不同，其地址线数也不同，CPU 的地址线数往往比存储芯片的地址线数多。

通常总是将 CPU 地址线的低位与存储芯片的地址线相连。CPU 地址线的高位或在存储芯片扩充时用,或做其他用途,如片选信号等。例如,设 CPU 地址线为 16 位 $A_{15}\sim A_0$,1 K×4 位的存储芯片仅有 10 根地址线 $A_9\sim A_0$,此时,可将 CPU 的低位地址 $A_9\sim A_0$ 与存储芯片地址线 $A_9\sim A_0$ 相连。又如,当用 16 K×1 位存储芯片时,则其地址线有 14 根 $A_{13}\sim A_0$,此时,可将 CPU 的低位地址 $A_{13}\sim A_0$ 与存储芯片地址线 $A_{13}\sim A_0$ 相连。

(2) 数据线的连接

同样,CPU 的数据线数与存储芯片的数据线数也不一定相等。此时,必须对存储芯片扩位,使其数据位数与 CPU 的数据线数相等。

(3) 读/写命令线的连接

CPU 读/写命令线一般可直接与存储芯片的读/写控制端相连,通常高电平为读,低电平为写。有些 CPU 的读/写命令线是分开的,此时 CPU 的读命令线应与存储芯片的允许读控制端相连,而 CPU 的写命令线则应与存储芯片的允许写控制端相连。

(4) 片选线的连接

片选线的连接是 CPU 与存储芯片正确工作的关键。存储器由许多存储芯片组成,哪一片被选中完全取决于该存储芯片的片选控制端 $\overline{CS}$ 是否能接收到来自 CPU 的片选有效信号。

片选有效信号与 CPU 的访存控制信号 $\overline{MREQ}$ (低电平有效)有关,因为只有当 CPU 要求访存时,才需选择存储芯片。若 CPU 访问 I/O,则 $\overline{MREQ}$ 为高电平,表示不要求存储器工作。此外,片选有效信号还和地址有关,因为 CPU 的地址线往往多于存储芯片的地址线,故那些未与存储芯片连上的高位地址必须和访存控制信号共同产生存储芯片的片选信号。通常需用到一些逻辑电路,如译码器及其他各种门电路,来产生片选有效信号。

(5) 合理选择存储芯片

合理选择存储芯片主要是指存储芯片类型(RAM 或 ROM)和数量的选择。通常选用 ROM 存放系统程序、标准子程序和各类常数等。RAM 则是为用户编程而设置的。此外,在考虑芯片数量时,要尽量使连线简单方便。

在实际应用 CPU 与存储芯片时,还会遇到两者时序的配合、速度、负载匹配等问题,下面用一个实例来剖析 CPU 与存储芯片的连接方式。

例 4.1　设 CPU 有 16 根地址线、8 根数据线,并用 $\overline{MREQ}$ 作为访存控制信号(低电平有效),用 $\overline{WR}$ 作为读/写控制信号(高电平为读,低电平为写)。现有下列存储芯片:1 K×4 位 RAM、4 K×8 位 RAM、8 K×8 位 RAM、2 K×8 位 ROM、4 K×8 位 ROM、8 K×8 位 ROM 及 74138 译码器和各种门电路,如图 4.36 所示。画出 CPU 与存储器的连接图,要求如下:

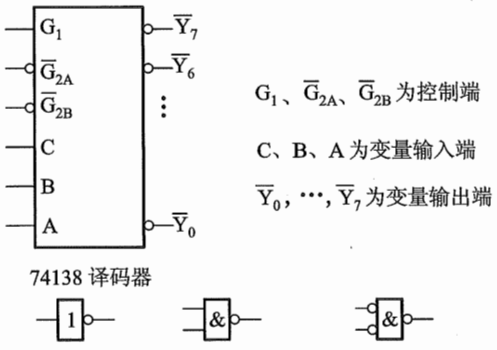


图 4.36　译码器和门电路

① 主存地址空间分配：

6000H～67FFH 为系统程序区。

6800H～6BFFH 为用户程序区。

② 合理选用上述存储芯片，说明各选几片。

③ 详细画出存储芯片的片选逻辑图。

解：第一步，先将十六进制地址范围写成二进制地址码，并确定其总容量。

A_{15} … A_{12} A_{11} … A_8 A_7 … A_4 A_3 … A_0

0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ┐
…　　　　　　　　　　　　　　　　├ 系统程序区
0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 ┘ 2 K×8 位

0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 ┐
…　　　　　　　　　　　　　　　　├ 用户程序区
0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 ┘ 1 K×8 位

第二步，根据地址范围的容量以及该范围在计算机系统中的作用，选择存储芯片。

根据 6000H～67FFH 为系统程序区的范围，应选择 1 片 2 K×8 位的 ROM，若选择 4 K×8 位或 8 K×8 位的 ROM，都超出了 2 K×8 位的系统程序区范围。

根据 6800H～6BFFH 为用户程序区的范围，选 2 片 1 K×4 位的 RAM 芯片正好满足 1 K×8 位的用户程序区要求。

第三步，分配 CPU 的地址线。

将 CPU 的低 11 位地址 A_{10}～A_0 与 2 K×8 位的 ROM 地址线相连；将 CPU 的低 10 位地址 A_9～A_0 与 2 片 1 K×4 位的 RAM 地址线相连。剩下的高位地址与访存控制信号 $\overline{MREQ}$ 共同产生存储芯片的片选信号。

第四步，片选信号的形成。

由图 4.36 给出的 74138 译码器输入逻辑关系可知，必须保证控制端 G_1 为高电平，$\overline{G}_{2A}$ 与 $\overline{G}_{2B}$ 为低电平，才能使译码器正常工作。根据第一步写出的存储器地址范围得出，A_{15} 始终为低电平，A_{14} 始终为高电平，它们正好可分别与译码器的 $\overline{G}_{2A}$（低）和 G_1（高）对应。而访存控制信号 $\overline{MREQ}$（低电平有效）又正好可与 $\overline{G}_{2B}$（低）对应。剩下的 A_{13}、A_{12}、A_{11} 可分别接到译码器的 C、B、A 输入端。其输出 $\overline{Y}_4$ 有效时，选中 1 片 ROM；$\overline{Y}_5$ 与 A_{10} 同时有效均为低电平时，与门输出选中 2 片 RAM，如图 4.37 所示。图中 ROM 芯片的 $\overline{PD}$/Progr 端接地，以确保在读出时低电平有效。RAM 芯片的读写控制端与 CPU 的读写命令端 $\overline{WR}$ 相连。ROM 的 8 根数据线直接与 CPU 的 8 根数据线相连，2 片 RAM 的数据线分别与 CPU 数据总线的高 4 位和低 4 位相连。

例 4.2 假设 CPU 及其他芯片同例 4.1，画出 CPU 与存储器的连接图。要求主存的地址空间满足下述条件：最小 8 K 地址为系统程序区，与其相邻的 16 K 地址为用户程序区，最大 4 K 地址空间为系统程序工作区。详细画出存储芯片的片选逻辑并指出存储芯片的种类及片数。

解：第一步，根据题目的地址范围写出相应的二进制地址码。

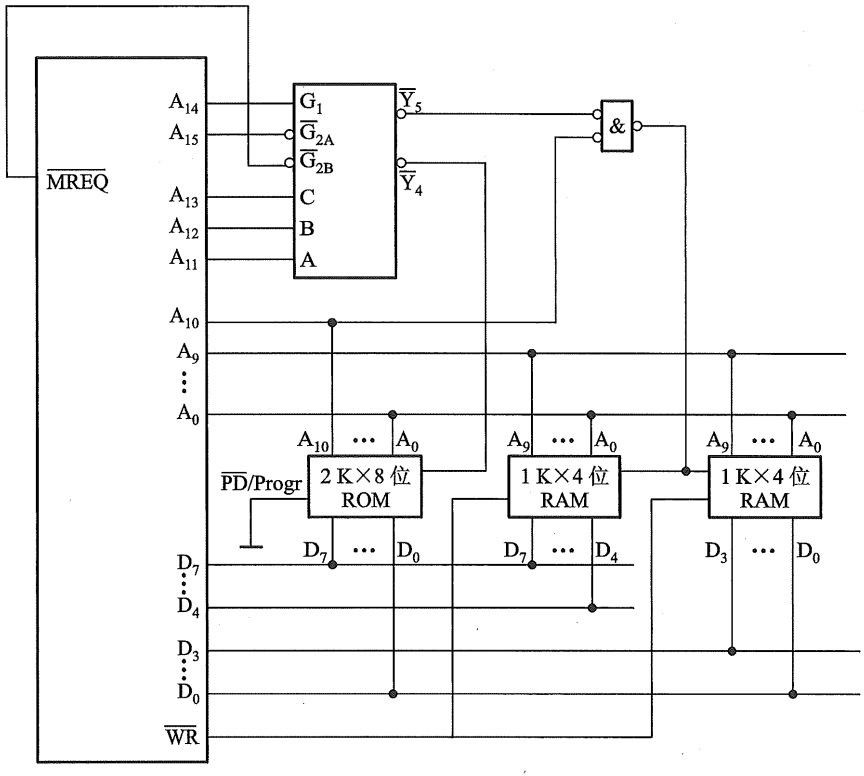


图 4.37　例 4.1 CPU 与存储芯片的连接图

A_{15} ··· A_{12}	A_{11} ··· A_8	A_7 ··· A_4	A_3 ··· A_0	
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	最小 8 K×8 位系统程序区
···				
0 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1	
0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	相邻 16 K×8 位用户程序区
···				
0 0 1 1	1 1 1 1	1 1 1 1	1 1 1 1	
0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	
···				
0 1 0 1	1 1 1 1	1 1 1 1	1 1 1 1	

A_{15} … A_{12} A_{11} … A_8 A_7 … A_4 A_3 … A_0

1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 } 最大 4 K×8 位

… } 系统程序工作

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

第二步,根据地址范围的容量及其在计算机系统中的作用,确定最小 8 KB 系统程序区选择 1 片 8 K×8 位 ROM;与其相邻的 16 KB 用户程序区选择 2 片 8 K×8 位 RAM;最大 4 KB 系统程序工作区选择 1 片 4 K×8 位 RAM。

第三步,分配 CPU 地址线。

将 CPU 的低 13 位地址线 A_{12}~A_0 与 1 片 8 K×8 位 ROM 和 2 片 8 K×8 位 RAM 的地址线相连;将 CPU 的低 12 位地址线 A_{11}~A_0 与 1 片 4 K×8 位 RAM 的地址线相连。

第四步,形成片选信号。

将 74138 译码器的控制端 G_1 接 +5V,$\overline{G}_{2A}$ 和 $\overline{G}_{2B}$ 接 $\overline{MREQ}$,以保证译码器正常工作。CPU 的 $A_{15}A_{14}A_{13}$ 分别接在译码器的 C、B、A 端,作为变量输入,则其输出 $\overline{Y}_0$、$\overline{Y}_1$、$\overline{Y}_2$ 分别作为 ROM、RAM_1 和 RAM_2 的片选信号。此外,根据题意,最大 4 K 地址范围的 A_{12} 为高电平,故经反相后再与 $\overline{Y}_7$ 相"与",其输出作为 4 K×8 位 RAM 的片选信号,如图 4.38 所示。

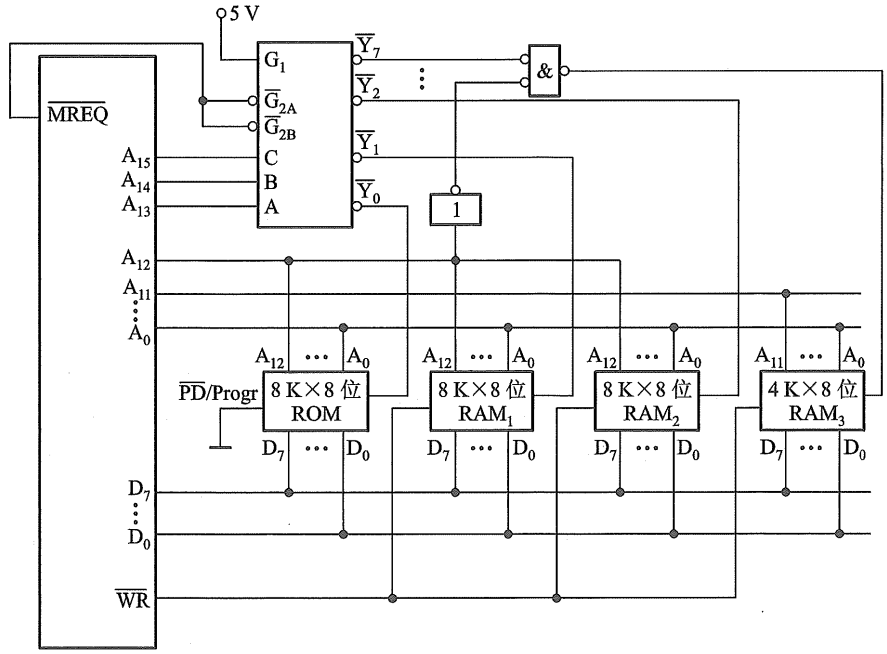


图 4.38 例 4.2 CPU 与存储芯片的连接图

例 4.3 设 CPU 有 20 根地址线和 16 根数据线,并用 IO/$\overline{M}$ 作为访存控制信号,$\overline{RD}$ 为读命

令，$\overline{WR}$ 为写命令。CPU 可通过 BHE 和 A_0 来控制按字节或字两种形式访存（如表 4.1 所示）。要求采用图 4.39 所示的芯片，门电路自定。试回答：

（1）CPU 按字节访问和按字访问的地址范围各是多少？

（2）CPU 按字节访问时需分奇偶体，且最大 64 KB 为系统程序区，与其相邻的 64 KB 为用户程序区。写出每片存储芯片所对应的二进制地址码。

（3）画出对应上述地址范围的 CPU 与存储芯片的连接图。

表 4.1　例 4.3 CPU 访问形式与 BHE 和 A_0 的关系

BHE	A_0	访问形式
0	0	字
0	1	奇字节
1	0	偶字节
1	1	不访问

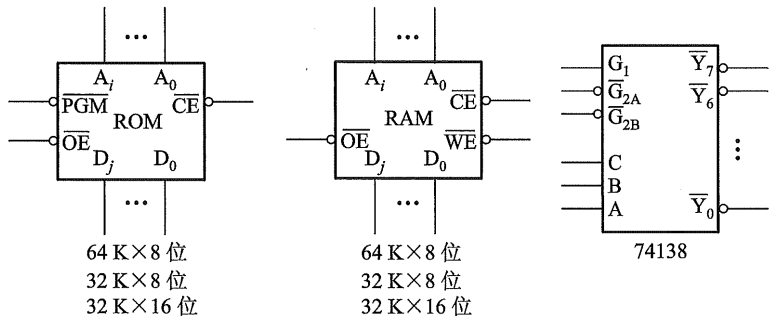


图 4.39　例 4.3 芯片

解：（1）CPU 按字节访问的地址范围为 1 M，CPU 按字访问的地址范围是 512 K。

（2）由于 CPU 按字节访存时需区分奇偶体，并且还可以按字访问，因此如果选择 64 K×8 位的芯片，按字节访问时体现不出奇偶分体；如果选择 32 K×16 位的芯片，虽然能按字访问，但不能满足以字节为最小单位。故一律选择 32 K×8 位的存储芯片，其中系统程序区 64 KB 选择 2 片 32 K×8 位 ROM，用户程序区 64 KB 选择 2 片 32 K×8 位 RAM。它们对应的二进制地址范围如下：

A_{19} … A_{16} A_{15} … A_{12} A_{11} … A_8 A_7 … A_4 A_3 … A_0

1 1 1 1　1 1 1 1　1 1 1 1　1 1 1 1　1 1 1 1　} 64 K×8 位 ROM，

…　} 其中 1 片 32 K×8 位（奇）

1 1 1 1　0 0 0 0　0 0 0 0　0 0 0 0　0 0 0 0　} 1 片 32 K×8 位（偶）

A_{19} … A_{16} A_{15} … A_{12} A_{11} … A_8 A_7 … A_4 A_3 … A_0

1 1 1 0　1 1 1 1　1 1 1 1　1 1 1 1　1 1 1 1　} 64 K×8 位 RAM，

…　　} 其中 1 片 32 K×8 位(奇)

1 1 1 0　0 0 0 0　0 0 0 0　0 0 0 0　0 0 0 0　} 1 片 32 K×8 位(偶)

该题的难点在于片选逻辑。由于 CPU 按字访问还是按字节访问受 BHE 和 A_0 的控制，因此可用 BHE 和 A_0 分别控制 74138 译码器的输入端 B 和 A，而 A_{15}～A_1 与存储芯片的地址线相连，余下的 A_{16}接 74138 译码器的输入端 C。A_{19}、A_{18}、A_{17}作为与门的输入端，与门输出接至 74138 译码器的 G_1 端，$\overline{G}_{2A}$和 $\overline{G}_{2B}$与 IO/$\overline{M}$相连，以确保 74138 译码器正常工作。具体连接图如图 4.40 所示。

图中译码器输出$\overline{Y}_4$有效时，同时选择 ROM_1 和 ROM_2，CPU 以字形式访问；$\overline{Y}_5$有效时选择 ROM_1(奇体)，$\overline{Y}_6$有效时选择 ROM_2(偶体)，CPU 以字节形式访问。同理，译码器输出$\overline{Y}_0$控制 CPU 可按字形式访问 RAM_1 和 RAM_2。$\overline{Y}_1$和 $\overline{Y}_2$分别按字节访问 RAM_1(奇体)和 RAM_2(偶体)。CPU 的读命令 $\overline{RD}$ 直接和 ROM、RAM 的 $\overline{OE}$(允许输出端)相连，CPU 的写命令 $\overline{WR}$ 直接和 RAM 芯片的 $\overline{WE}$(允许写输入端)相连。ROM 芯片的 $\overline{PGM}$ 端低电平时可编程，接高电平 V_{CC}时可按只读方式工作，$\overline{CE}$ 为片信号，分别与不同的译码输出端相连。

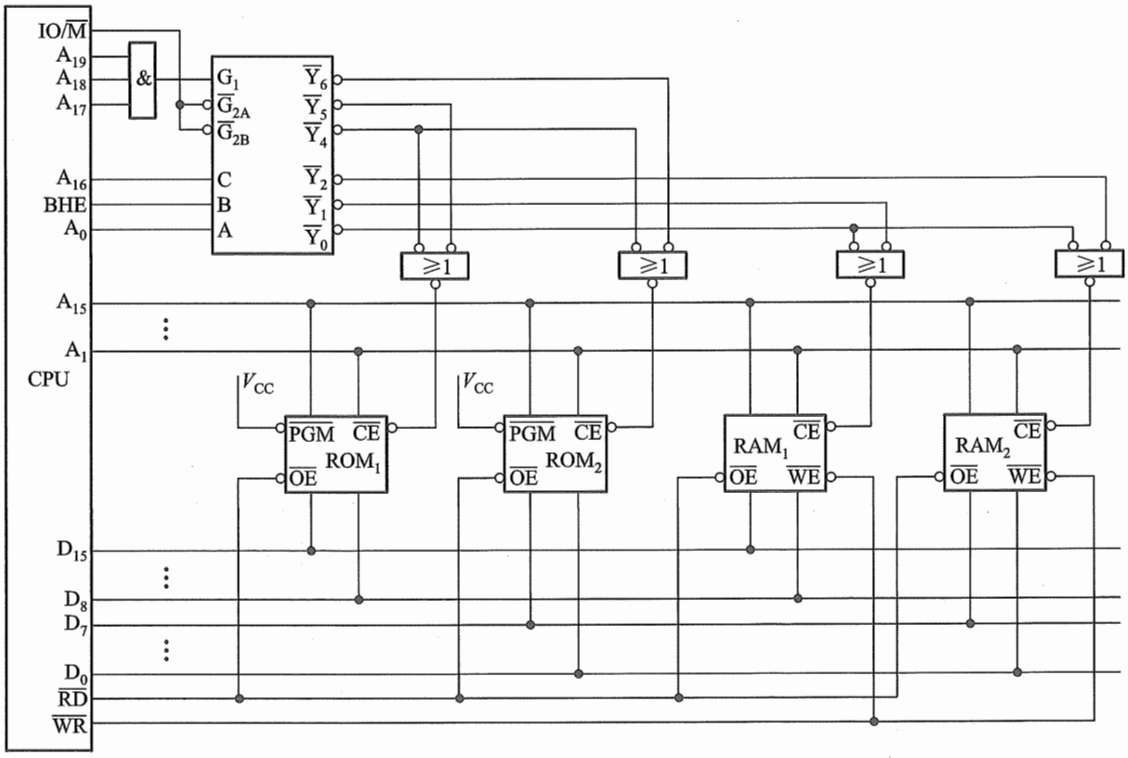


图 4.40　例 4.3 CPU 与存储芯片的连接图