第二节 主存储器

一、概述

1.主存的基本组成

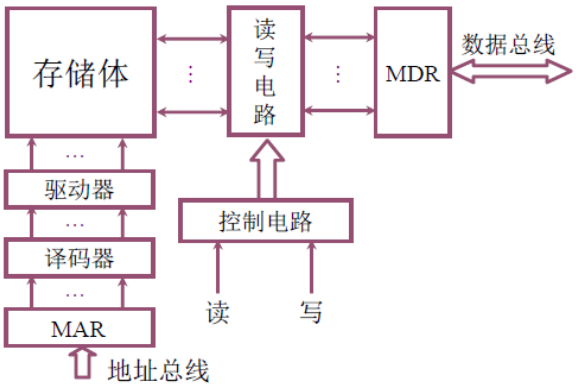

图 2.1 主存储器的框图复杂版

MAR 还是存储地址，经过译码器和驱动器传递给存储体。MDR 存储输入或者写出的数据，由读写电路决定。

和辅存相比，主存的特点是**容量小，速度快，成本高**。

2.主存与 CPU 之间的联系

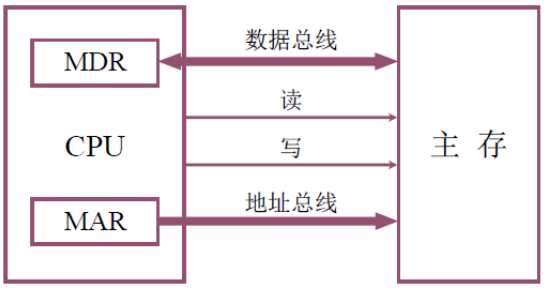

图 2.2 主存与 CPU 之间的联系

依旧是三条总线，数据总线、地址总线和控制总线。数据总线连接主存和 MDR，控制总线确定读还是写入，地址总线由 CPU 里面的 MAR 送给主存储器。

3.主存中存储单元地址的分配

假设我们的存储字长是 32 位，那么这个 16 进制数 12345678H 如何在主存储器中进行存储呢？

一种方式是高位字节地址为字地址，如图 2.3 左所示。0/4/8 是字地址，每个字地址包含 4 个字节，高位字节存放在低地址，地位字节存放在高地址，并且以高位字节的地址作为存储字的地址。这种被称为**大端方式**（或**大尾方式**）。

还有一种方式是低位字节地址为字地址，如图 2.3 右所示。0/4/8 还是存储字的地址，每个字包含 4 个字节，4 个字节的地址分别是 0/1/2/3，把高位存在高地址，低位存在低地址，我们把低位字节的地址称为存储字的地址。这种被称为**小端方式**（或小**尾方式**），X86 用这种方式。

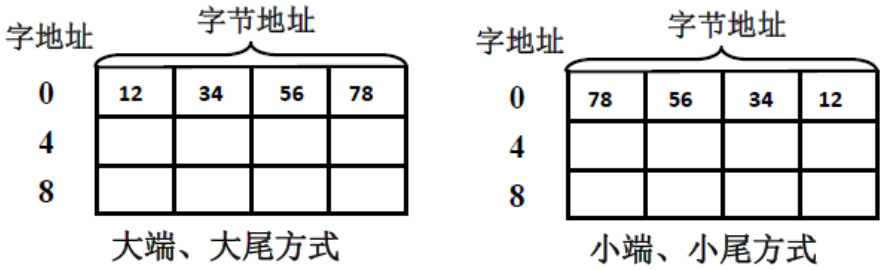

图 2.3 存储单元地址分配

假设我们有 24 根地址线，如果我们按字节寻址，24 根地址线，可以访问 16MB（2 的 24 次方）。如果字长为 16 位，按字寻址，那么就是 8MW；若字长为 32 位，按字寻址，就是 4MW。

4.主存的技术指标

（1）存储容量：主存存放二进制代码的总位数。

（2）存储速度：

• 存取时间：存储器的访问时间、读出时间和写入时间；

• 存取周期：**连续**两次**独立**的存储器操作（读或写）所需要的**最小**间隔时间（包含读周期和写周期）。

（3）存储器的带宽：位/秒。

二、半导体存储芯片简介

1.半导体存储芯片的基本结构

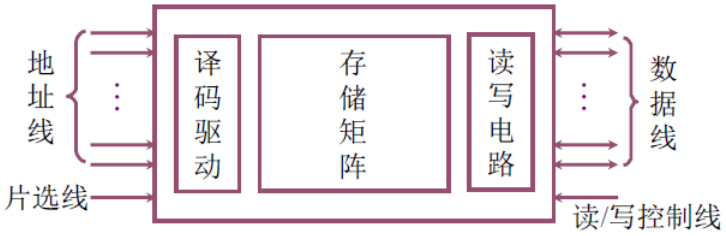

图 2.3 基本结构

片选线选定该芯片之后，地址线输入地址，经过译码驱动电路选定存储矩阵中存储数据的位置，用读写电路完成读写操作（读或写由读/写控制线确定）。

地址线是单向的，数据线是双向的。假如地址线是 10 位，数据线是 4 位的，那么芯片容量就是 $4*2^{10}$ 位，也就是 1K*4 位。

片选线一般有两种表现方式，CS 和 CE，低电平有效就是上面有横杠，和数字电路中已有。

读/写控制线：WE（上面有横杠，低电平写，高电平读），或者 OE（上面有杠，低电平允许读），WE（上面有杠，低电平允许写）。

那么如何用 16K*1 位的存储芯片组成 64K*8 位的存储器呢？

我们可以以 8 片为一组，每次片选信号选中一组（即 8 片），然后共 4 组。如图 2.4 所示。

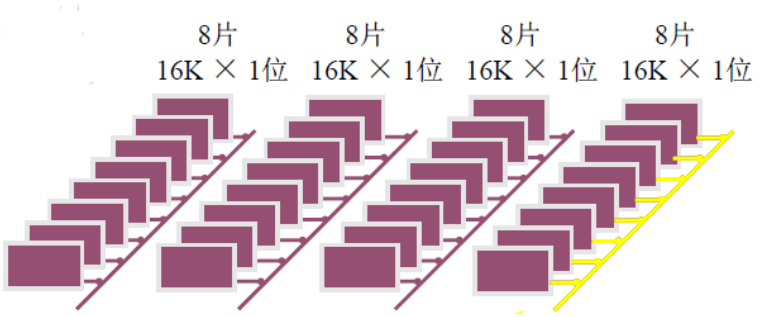

图 2.4 64k*8 位存储器

2.半导体存储芯片的译码驱动方式

（1）线选法

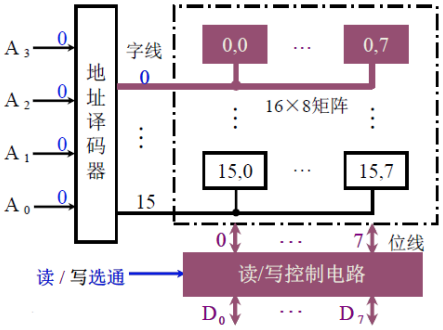

图 2.5 线选法示意图

根据地址线有 4 条来确定是由 16 个存储单元，根据数据线来确定每个存储单元有 8 位。关于译码器的知识点在数字电路已经讲过，在这里不再重述。

因为这种方法在大型存储器中不太合适，所以提出了重合法。

（2）重合法

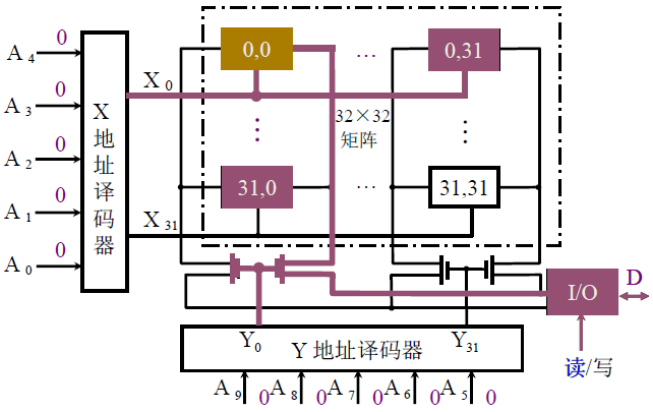

图 2.6 重合法

重合法是将所有的存储单元布局称为一个二维的阵列。我们把地址译码器分为了 X 和 Y，通常情况下，我们称之为**行地址**和**列地址**。行列地址分别译码，每个译码器输出只有一根线有效。

加入我们行地址和列地址都给出的是 00000，那么作用的就是（0,0）存储单元被通过 I/O 口操作。其实我们在选通之后，(0,31)和(31,0)都是被选通的，但是由于其另一条地址线不通，所以其数据无法进行操作。

我们来对比下线选法和重合法，如果有 20 个地址线，由线选法操作，输出是 **1M** 条线；

如果重合法的话，我们将其分为行地址线和列地址线，每部分 10 条线，那么行地址线输出是 1K 条，列地址线输出是 1K 条，总共只需要 **2K** 条，比线选法少太多了，因此集成度可以很高。

三、随机存取存储器（RAM）

1.静态 RAM（SRAM）

解决问题：

- 保存 0 和 1 的原理是什么？
- 基本单元电路的构成是什么？
- 对单元电路如何读出和写入？
- 典型芯片的结构是很么样子的？
- 静态 RAM 芯片的如何进行读出和写入操作？

（1）静态 RAM 基本电路

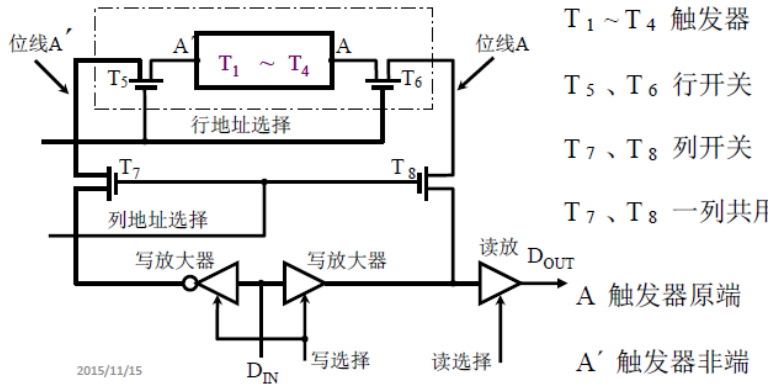

图 2.7 静态 RAM 基本电路

静态 RAM 核心是利用触发器来实现 0/1 存储的。这个存储器是 4 个管子构成，是双稳态的触发器，一端是 0，另一端是 1。T1 到 T4 解决的是用什么样的电路存放 0/1，T5 和 T6 解决对存储软件是读还是写。如果 T5 和 T6 倍选通，那么位线 A 和位线 A’被选通。这 6 个管子就是静态 RAM 的基本组成部分。这又被称为**六管静态 RAM**。

实际电路中，虚线的部分是一个存储单元，而整个电路是由这个存储单元组成的一列的多列组成，T7 和 T8 就是一列上共用的管子，称为列开关。如果某一列被选中，那么 T7 和 T8 就会被选通。

实际中，读数据是直接读，而写数据则是在一端输入我们需要的数据（0 或 1），另一端输入其非。

①静态 RAM 基本电路的**读**操作

一共是由 6 个晶体管构成的。如果要进行读操作，要给出行选（T5/T6 开）和列选（T7/T8 开），那么读有效这个管子就会导通。存放在 A 中的数据就会通过 T6 这个晶体管送到位线上，T8 会往前送，一直送到最后的 Dout。我们知道一旦开始读操作，非端也在进行操作，信号通过 T5 到 T7 最后到达左侧的写放大器，阻止了。

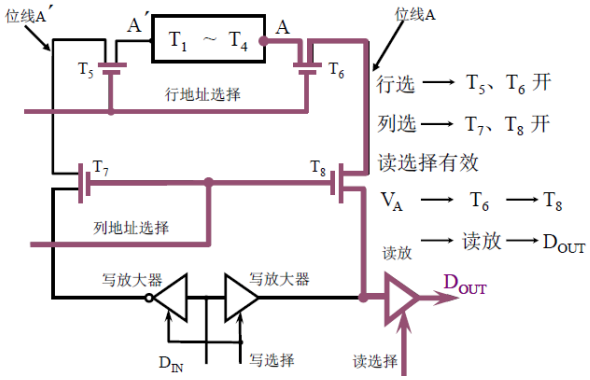

图 2.8 静态 RAM 基本电路的读操作

②静态 RAM 基本电路的**写**操作

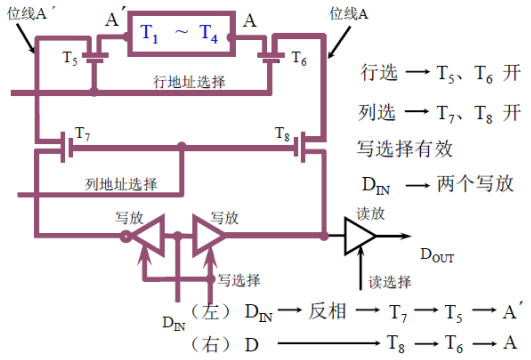

图 2.9 静态 RAM 的写操作

注意一下，左侧的写放大器是取非后进行输出的，这样保证在由 T1 到 T4 组成的双端稳态的触发器两端的信号是相反的。还是一样，先行选（T5/T6 开），列选（T7/T8 开）。要进行写入，写选择信号有效，两个写放大器导通，通过 DIN 输入。左侧通过 DIN 后被**取反**，然后通过 T7 到 T5 最后到达 A’，D 通过 T8 经过 T6 到达 A。

（2）静态 RAM 芯片举例

①Intel2114 外特性

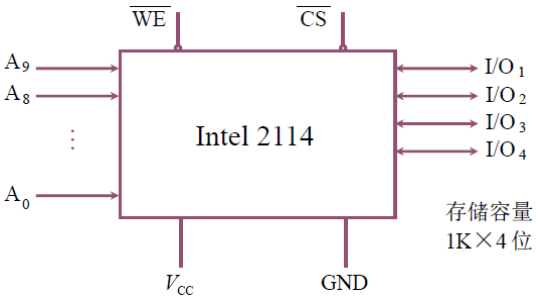

图 2.10 Intel2114 外特性图

WE 是读写控制信号，低电平是写操作，高电平是读操作。CS 是片选信号，低电平有效。左侧 A0 到 A9 是地址线（1K 个存储单元），右侧 I/O1 到 I/O4 是数据线，说明每个存储单元有 4 个基本电路。

曾经讲过重合法，那么怎么实现选一次四列呢？

②Intel 2114 RAM 矩阵（64*64）**读**操作

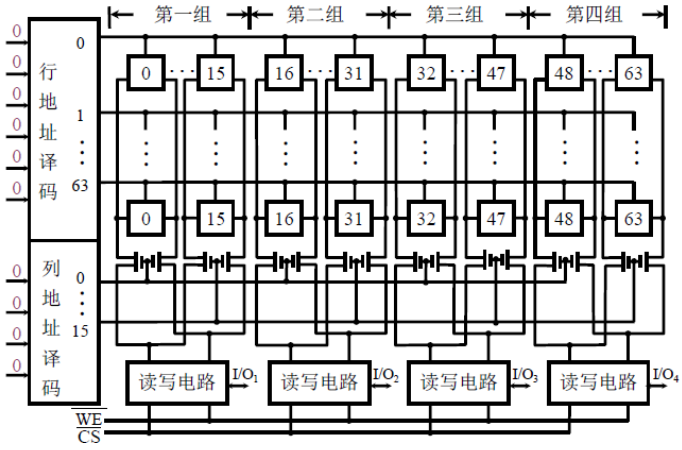

图 2.11 读操作示意图

行地址有 6 位，译码后可以产生 0 到 63 一共 64 个行地址信号。列地址有 4 位，有 16 个列地址信号，没给出一共信号，同时 4 片被选中。我们把 64 列分为了 4 组，每一组包含了 16 列存储单元，每一个列选信号控制每一组的一列存储单元。

假设行地址和列地址输入均为 0.行地址经过译码后，所有的第 0 行都会被选中（0/15/16/31/32/47/48/63），经过列地址译码，每组中第 0 列都会被选中（0/16/32/48）。如果进行读操作，第 0 行的每组的第 0 列都会进行数据输出，经过列控制管和读写电路输出其存储信息。

③Intel 2114 RAM 矩阵（64*64）写操作

同样地址均给 0，那么经过行地址译码后，第 0 行上的存储单元被选中，列地址译码器译码后，每组第 0 列就会被选中。这样交叉点上的存储单元就会被进行写入操作。

2.动态 RAM（DRAM）

解决问题：

- 保存 0 和 1 的原理是什么？
- 基本单元电路的构成是什么？
- 对单元电路如何读出和写入？
- 典型芯片的结构是很么样子的？
- 动态 RAM 芯片的如何进行读出和写入操作？
- 动态 RAM 为什么要刷新，刷新方法？

（1）动态 RAM 基本单元电路

如图 2.12 所示，我们利用的是**电容**。如果电容中保存有电荷，我们认为是 1；否则我们认为是 0。

左侧是第一种方法，称为**三管动态 RAM**。信息就是保存在 Cg 上。T1、T2 和 T3 是控制管，上面这条横线是读选择线，读选择线被选中的时候，T2 这个管子会导通；下面这个横线是写选择线，如果被选中，那么 T3 这个管子会导通，外部数据可以通过 T3 进行操作。

如果进行读出信号。T4 是一个预充电信号，如果预充电信号有效，T4 就会被打开，VDD 会通过 T4 给读数据线进行充电，使读数据线变成高电平，如果读出的话，读选择先有效，那么 T2 导通，如果这个时候 Cg 有信息为 0，也就是 Cg 没有进行充电，那么 T1 管栅极是低电平，T1 不会导通，读数据线是高电平，所以读数据线上读出的是 1。如果保存信息是 1，那么 Cg 是有电容，T1 栅极就是高电平，T1 就是导通的，刚开始读数据线是高电平，但是这时就会通过 T2 和 T1 进行放电，这个时候读出信息就是 0。所以，**读出信息与存储信息是**

相反的；写入与输入信息相同。

右侧是第二种方法，称为**单管 RAM**。信息保存在 Cs 上。如果相应的行选通，那么字线控制的这个 T 就会被打开，电容就可以通过 T 充电或放电。如果 Cs 保存为 0，那么数据线上就不会有电流，如果电容保存的是 1，那么数据线上就会有电流。如果写入的话，写 1 位充电，写 0 为放电。

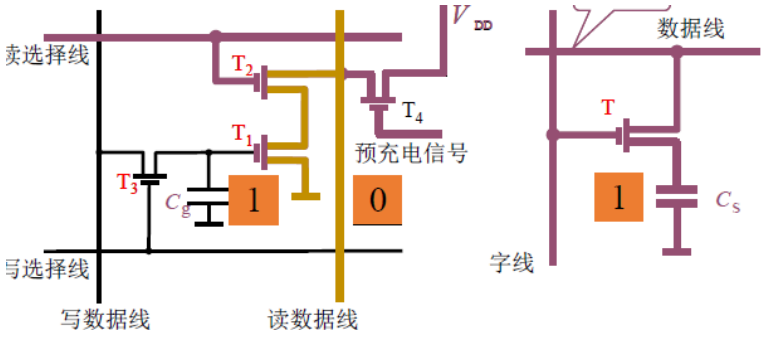

图 2.12 动态 RAM 基本单元电路

（2）动态 RAM 芯片距离

①三管动态 RAM 芯片（Intel1103）**读**操作

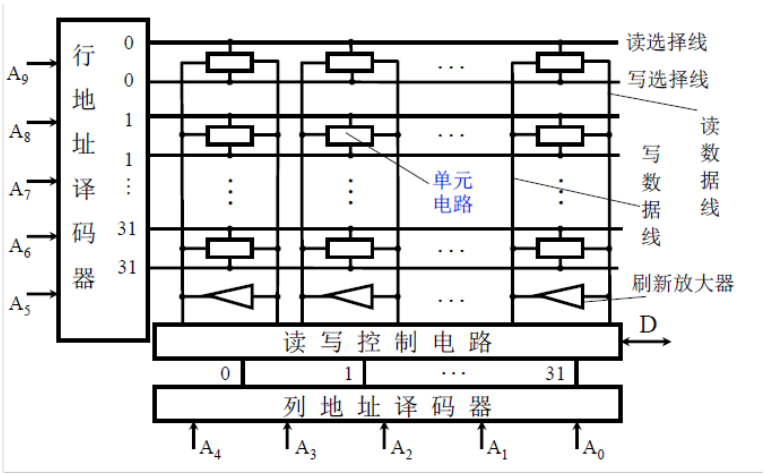

图 2.13 Intel1103 芯片电路图

地址输入依然是由行地址和列地址组成。每次读出和写入只有 1 位数据。这个芯片的容量是 1K*1 位。每一个行地址译码器的译码结果都对应着两个输出，分别对应着读选择线和写选择线，也就是说在行地址译码器不仅仅是地址参与，还有读写控制信号参与。

下面我们来看看如何进行读操作假设行地址译码器为 00000，那么第 0 行的读选择线有效，第 0 行所有单元都被选中进行读操作，列地址依旧是 00000，第 0 列被选中，第 0 行第 0 列的单元通过读数据线把数据送到读写控制电路，同时输出。看图 2.13 中读写控制电路上面有一个三角形的器件，这是一个**刷新放大器**。

之所以采用刷新放大器是因为，**我们采用电容存储电荷原理进行存储信息的，电容会漏电，经过一段时间之后，电容上信号会消失，所以采用刷新放大器就是对电容中保存的信息进行重现。**每隔一段时间，都需要对给定存储单元中的信息进行刷新。

②三管动态 RAM 芯片（Intel1103）**写**操作

假设行地址是 11111，第 31 行被选中，写选择线有效，第 31 行所有存储单元准备好接受信息的写入。列地址信号为 00001，那么第 1 列被选中。也就是被选中的单元为第 31 行第 1 列的单元。数据通过 D 端进行输入，通过读写控制电路，会被写入到该单元。

③单管动态 RAM4116（16K*1 位）外特性

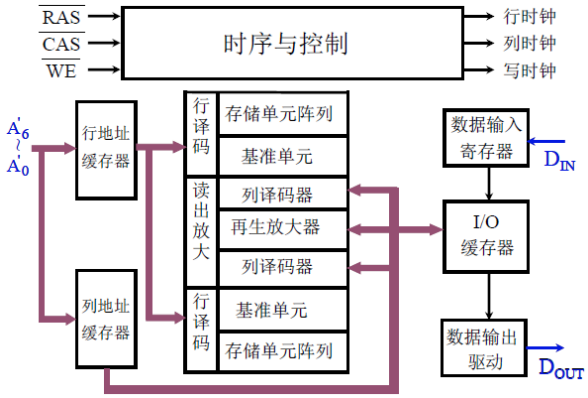

图 2.14 4116 的外特性图

这里面有一点和我们前面讲的不一样，4116 是 16K*1 位的，从这个数字上看，16K 需要 14 根地址线，但是 4116 只有 7 根地址线，实际上他需要 14 位地址，这 14 位地址是**分两次**进行传送的。第一次接收到 7 位行地址，第二次接收 7 位列地址，分别存放在行地址缓存器和列地址缓存器，经过译码后选择存储单元。图 2.14 下右侧有一个 I/O 缓存器，是输入输出数据的缓存，其两端连接一个数据输入寄存器和数据输出驱动，可以完成数据的输入和输出。RAS 是行选通信号，CAS 是列选通信号，WE 是读写控制信号。右侧的行时钟、列时钟和写时钟控制了芯片内部的操作。

④4116（16K*1 位）芯片**读**原理

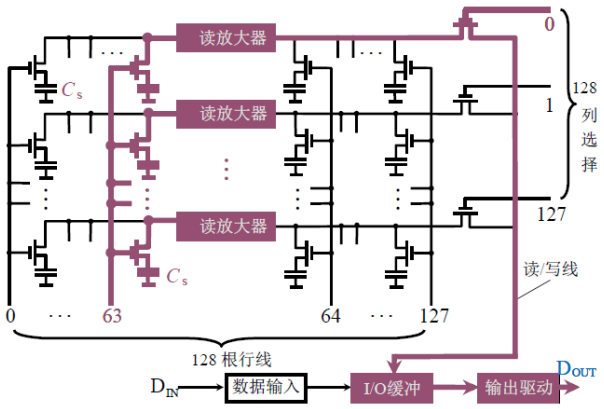

图 2.15 4116 的读原理图

128 根行线，128 根列选择，在这个图上**横的方向是列，竖的方向是行**。我们看在第 63 行和 64 行之间，每一列上都连接一个读放大器，这个读放大器是一个“跷跷板电路”，也就是说如果我们在其一端强制为 1，那么另一端则为 0。

如果给出行地址为 0111111，那么第 63 行被选中，第 63 行所有的晶体管都被打开，电容中的电荷就会被放出，到达读放大器的左侧。我们以第一个晶体管为例，如果这里面有电的话，那么这个读放大器左侧就是 1，右侧就是 0；如果电容没有电的话，那么读放大器左侧是 0，右侧就是 1。

假如列地址为 0000000，那么就是第 0 列被选中，相对应晶体管打开，那么第 0 列响应的信号经过右上角的晶体管送到读写线上，经过 I/O 缓冲、输出驱动被送出。

由于读放大器的存在，所以会有这样一个现象：**在读放大器左侧，电容有电是 0，电容无电是 1；在读放大器右侧，电容有电是 1，电容无电是 0。**

⑤4116（16K*1 位）芯片**写**原理

假设行地址依旧是 0111111，那么第 63 行被选中，63 行的所有控制管打开，数据输入

通过 DIN 到数据输入到 I/O 缓冲，最后送到读/写线上。假设列选择为 0000000，那么依然是第 0 列有效，第 0 列管子被打开，其它的管子都是截止的，那么信号只能被送到第 0 列读放大器的右侧，通过读放大器左端写入到指定单元。假设写入信号是 1，那么读放大器右端是 1，左端就是 0，电容当中就没有电；假设写入信号是 0，那么读放大器左侧是 0，右侧就是 1，电容充电。

所以存在这样一个现象：**在读放大器左侧，电容有电是 0，电容无电是 1；在读放大器右侧，电容有电是 1，电容无电是 0。**

（3）动态 RAM 刷新——刷新只与**行地址**有关

①集中刷新（存取周期为 0.5μs），以 128*128 矩阵为例

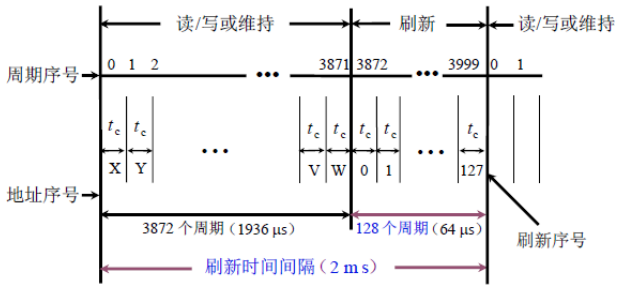

图 2.16 集中刷新示意图

所谓集中刷新，就是将刷新存放在相对集中的一段时间来进行操作。假设存取周期为 0.5μs，刷新时间间隔为 2ms（所有的电容在这 2ms 内都要完成一次信息的再生），以 128*128 矩阵为例来讲解。2ms 一共是 4000 个存取周期，前 3872 个周期（1936μs）可以供 CPU 和 I/O 对动态 RAM 进行读写操作，后面 128 个周期（64μs）专用于芯片刷新操作，也就是说在这段时间中无论是 CPU 还是 I/O 都无法对动态 RAM 进行操作，这段时间称为死区。

死区事件率为 128/4000*100%=3.2%。

这种方法存在的问题就是如果在死区时，CPU 或 I/O 想对动态 RAM 操作是不可能的。

②分散刷新（存取周期为 1μs）

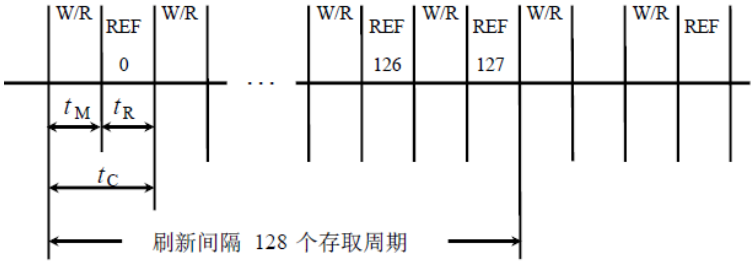

图 2.17 分散刷新示意图

整个存取周期 tc 包含两部分，为 tM 和 tR，前者为原来的读写周期，后者为也是 0.5μs，专门用于某一行的刷新。这种方法 128μs 就把 128 行刷新完了。这实际是一种过度的刷新，动态 RAM 不需要这么多次刷新。优点是**没有死区**，但是把读写周期加长了，性能下降了。

③分散刷新与集中刷新相结合（异步刷新）

若每经过 15.6μs 刷新一行，那么可以将这个 2ms 分隔为若干个区段，如图 2.18 所示。每个区段里面设置 0.5μs 进行刷新，也就是说死区为 0.5μs。刷新可以安排在任意位置，但是如果**将刷新安排在指令译码阶段，就不会出现“死区”**。

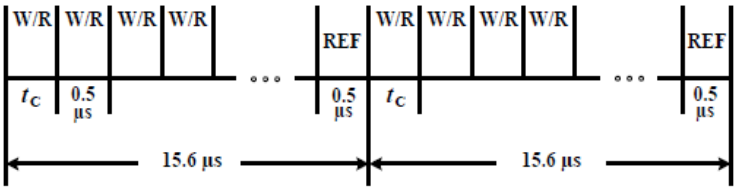

图 2.18 异步刷新示意图

3.动态 RAM 和静态 RAM 的比较

表 2.1 动态 RAM 和静态 RAM 的比较

	DRAM（主存）	SRAM（缓存）
存储原理	电容	触发器
集成度	高	低
芯片引脚	少	多
功耗	小	大
价格	低	高
速度	慢	快
刷新	有	无

四、只读存储器（ROM）

只读存储器的发展历程：

•早期的只读存储器——在厂家就写好了内容；

•改进 1——用户可以自己写——一次性；

•改进 2——可以多次写——要能对信息进行擦除；

•改进 3——电可擦写——特定设备；

•改进 4——电可擦写——直接连接到计算机上。

1.掩模 ROM（MROM）

行列选择先交叉处有 MOS 管为 1，无 MOS 关为 0。（数电课程已经讲过）

2.PROM（一次性编程）

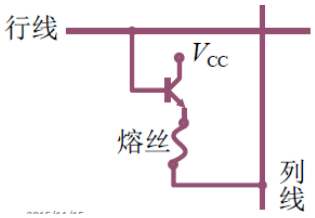

图 2.19 PROM 示意图

熔丝断为 0，熔丝未断为 1,。一旦编程，无法修改。（数电课程已经讲过）

3.EPROM（多次性编程）

它采用的是 N 沟道浮动栅 MOS 电路。对这种类型的电路来说，G 是栅极，S 为源，D 是漏。这种电路如果在 D 端加上正电压，源和漏形成浮动栅，是源和漏不导通，保存信息我们认为是 0；如果不加电压，源和漏导通，不形成浮动栅，保存信息我们认为是 1。

这种方法如果我们需要对其修改，我们就要驱散这个浮动栅，用紫外线直接照射电路，

清洗浮动栅。

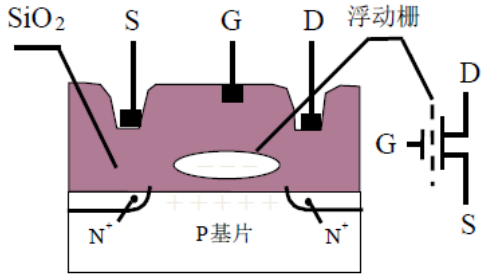

图 2.20 N 沟道浮动栅 MOS 电路

但是信息擦除比较麻烦。

4.EEPROM（多次性编程）

支持：电可擦写、局部擦写和全部擦写。

5.Flash Memory（闪速型存储器）

EPROM 价格便宜，集成度高；EEPROM 电可擦洗重写，FLASH MEMORY 比 EEPROM 快，具备 RAM 功能。

五、存储器与 CPU 的连接（重点和难点）

1.存储器容量的扩展

（1）位扩展

用 **2 片** 1K*4 位存储芯片组成 1k*8 位的存储器。即 10 根地址线，8 根数据线。如图 2.21 所示。

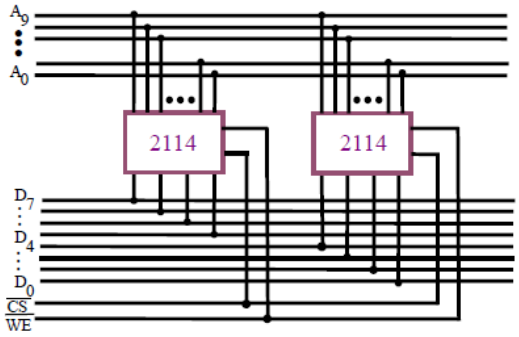

图 2.21 实现位扩展的框图

（2）字扩展

用 **2 片** 1K*8 位存储芯片组成 2K*8 位的存储器。如图 2.22 所示。

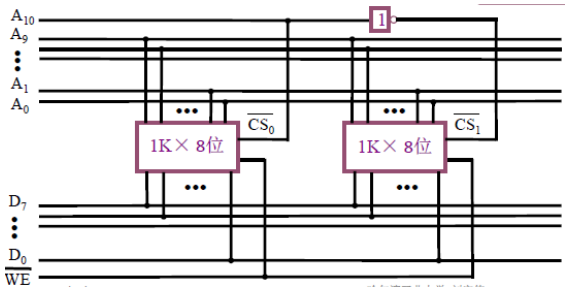

图 2.22 字扩展示意图

如图所示，D0 到 D7 分别连接到芯片的数据口，然后 A0 到 A9 依旧是分别连接到两个

芯片对应的地址入口，A10 连接芯片 1 的片选接口，A10 取反以后连接到芯片 2 的片选接口。

（3）同时扩展

用片 1K*4 位存储芯片组成 4K*8 位的存储器，即 12 根地址线，8 根数据线。可以分析得知，既然是 4 位扩展成 8 位，那就需要两个该存储芯片组成一组。由于是 12 根地址线，多出来了 2 根，所以需要接译码器，可以得出 4 个结果来执行片选操作，所以要有 4 组。故其连接图如图 2.23 所示。

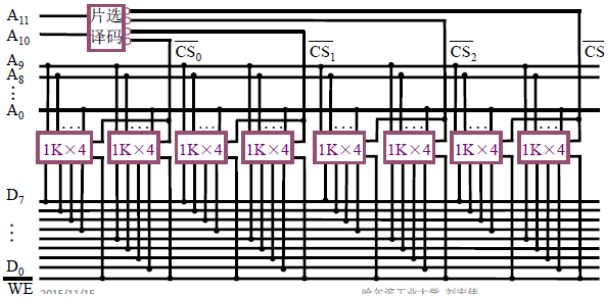

图 2.23 同时拓展示意图

2.存储器与 CPU 的连接

包括：

①地址线的连接；

②数据线的连接；

③读/写命令线的连接；

④片选线的连接：要确定 CPU 访问的是存储器还是 I/O，如果是存储器一定要在片选信号中体现，每一个芯片都有自己的地址范围，必须满足 CPU 的要求；

⑤合理选择存储芯片，ROM 和 RAM 合理选择，如果保存系统程序和配置信息则用 ROM，保存用户程序用 RAM，要保证芯片数量尽可能少，片选参数尽可能少；

⑥其它：时序、负载等。

例 4.1：

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：

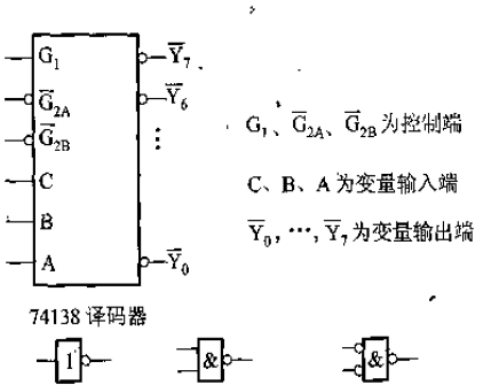

图 4.36 译码器和门电路

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。

解析：

CPU 有 16 根地址线说明其寻址范围为 64K，8 根数据线说明其读写数据位数为 8 位。

具体解题步骤如下：

（1）写出对应的二进制地址码

A15	A14	……													A0	
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM 2K*8
……																
0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	RAM 1K*8
……																
0	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	

（2）确定芯片的数量及类型

可以看在系统程序区中，前 5 位不发生变化，后面 11 位从全 0 变成全 1。所以可以忽略前面 5 位，只考虑后面 11 位地址，因此可以是 **1 片 2K*8 位 ROM** 来实现。

可以看在用户程序区中，前 6 位不发生变换，后面 10 位从全 0 变成全 1。所以可以忽略前面 6 位，选择 1K*8 位的 RAM。所以可以用 **2 片 1K*4 位 RAM** 来实现。

（3）分配地址线

对于 ROM 来说，前面 5 位不发生变化，因此选择后 11 位作为内部地址线，即将 A10~A0 接 2K*8 位 ROM 的地址线，剩余这些作为芯片选择信号。

对于 RAM 来说，前面 6 位不发生变化，因此选择后 10 位作为内部地址线，即将 A9~A0 接 1K*4 位 RAM 的地址线，剩余这些作为芯片选择信号。

74LS138 译码器有 C/B/A 三个输入端，有 8 个输出端来进行片选。由于 A15~A14 完全相同，因此这里我们可以用 A13~A11 来连接 138 译码器的三个输入端，即 A13 连接 C，A12 连接 B，A11 连接 A。

由于 G1 是高电平有效，所以可以把 A14 连接到 G1 上。由于 G2A 和 G2B 是低电平有效，所以可以将 A15 连接到 G2A 和 G2B 上。另外，**MREQ 一定要用，连接到片选信号上**。

（4）确定片选信号

最后连接图如图 2.24 所示。

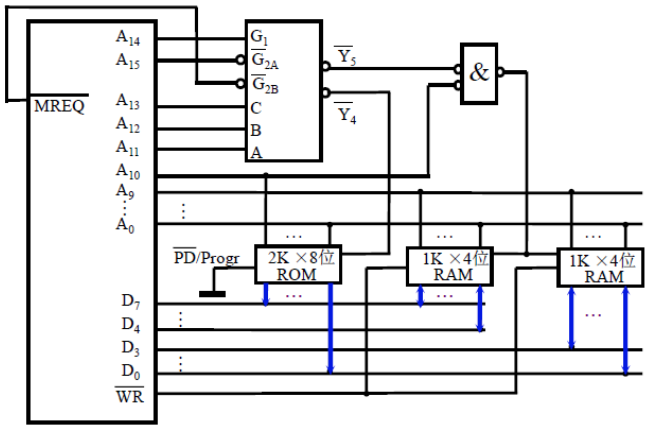

图 2.24 例 4.1 结果图

在图 2.24 里面，MREQ 连接到 G2B 上（因为 MREQ 一定要用），A13/A12/A11 分别连接到 C/B/A 上，相应地址线也分别连接到相应的位置。根据上表所示的结果，输出 Y4 的话就连接到 ROM 上（100），输出是 Y5，那么就连接到 RAM 上，由于在连接 RAM 时 A10 没有用，而 A10 必须要用，并且 A10 为零，所以取一个与连接到 RAM 上。因为 ROM 是只读，所以 CPU 的 WR 只跟 RAM 相连接。数据线直接连接到 ROM 中，而对于 RAM 来说，则两片分别连接 4 个。**由于 ROM 这里只读，所以将 PD 口置零。**

例 4.2：假设同前，要求最小 4K 为系统程序区，相邻 8K 为用户程序区。步骤依然同样：

（1）写出对应的二进制地址码（2）确定芯片的数量及类型

A15	A14	……													A0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM
……																4K*8
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1 片
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	RAM
……																4K*8
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1 片
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	RAM
……																4K*8
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1 片

（3）分配地址线

A14/A13/A12 对应 C/B/A。

（4）确定片选信号

（5）确定片选逻辑

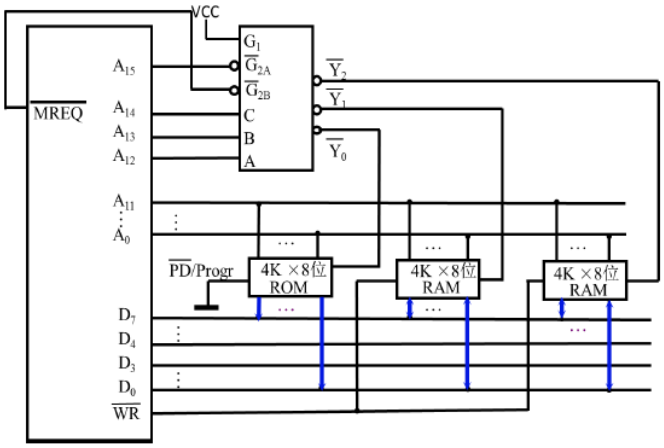

图 2.25 例 4.2 结果图

由于 ROM 这里只读，所以将 PD 口置零。

六、存储器的校验

•为什么要对存储器的信息进行校验？

为了防止存储器信息出现错误带来的灾难性后果。

•为了能够校验出信息是否正确，如何进行编码？

•纠错或检错能力与什么因素有关？

•校验出信息出错后是如何进行纠错？

1.合法代码集合

检错能力和任意两组合法代码之间二进制位的**最少差异数**有关，差异越多，检错和纠错能力越强。

2.编码的最小距离

任意两组合法代码之间**二进制位数**的**最小差异**。编码的纠错、检错能力与编码的最小距离有关系。公式如下：

L-1=D+C(D>=C)

其中，L 为编码的最小距离，D 为检测错误的位数，C 是纠正错误的位数。

3.汉明码的组成

汉明码采用奇偶校验、分组校验。汉明码的分组是一种**非划分**方式，组和组之间是有交叉的。

例如有：1 2 3 4 5 6 7。我们将其分为 3 组，每组有 1 位校验位，共包括 4 位数据位。我们可以这样分组：

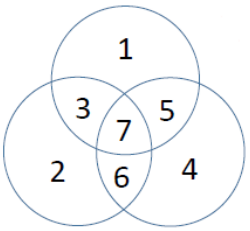

图 2.26 汉明码分组

以 1,2,4 为三个圆（2^i），1 和 2 的和放在 1 圆和 2 圆的交叉处，一次类推，7 放在正中间。这样就形成了 3 组，每一组都是偶校验方式，每一组 4 个数据，其中一个数据为校验位。这三组会出现下面的情况：

表 2.2 校验位结果

P3	P2	P1	结果	结果说明
0	0	0	无	无错误
0	0	1	1	说明错误出现在第一组，且为第一组独有的元素
1	0	1	5	第一组和第三组错了，只错了 1 位，说明为其公共位置错误。看图中是 5，正好也是 101 编码的位置。
1	1	0	6	第二组和第三组错了，只错了 1 位，说明为其公共位置错误，看图中是 6，编码 110 正好也是 6。
1	1	1	7	第 1/2/3 组都错了，只错了一位，说明为其公共位置错误，看图中是 7，编码 111 正好也是 7。

这种方法的好处是给出的二进制数确定了出错的位置。

那么校验位放在哪里呢？由于校验位只对这一组进行校验，所以校验位不能放在与其他组共享的位置，只能放在自己独有的位置，也就是 2^i 位置上。

如何分组呢？第一组编码二进制化后最右侧的位置是 1，第二组编码二进制化最右侧第二个位置是 1，以此类推下去。如果从最右侧数第 1 和第 2 位均为 1，那么就是第 1 组和第 2 组共有的，如果第 1 位和第 3 位均为 1，那么就是第一组和第三组共有的，以此类推……

我们来看一下校验过程。如果我们对每一组进行校验，假设第一组出错了，那么最后 1 位一定是 1，假设第 1 组和第 3 组都出错了，那么最后 1 位和倒数第三位都是 1，说明是 5 的这个位置出错了。

汉明码组成的三要素：

（1）汉明码的组成需要添加 k 位检测位（也就是分成组），这个 k 满足 $2^k>=n+k+1$

$$2^k \geq n+k+1 \tag{2.1}$$

其中信息位有 n 位，校验位有 k 个，再加上没有错误的是 1 种情况；

（2）检测位的位置为：2^i（i=0,1,2,3,...）；

（3）检测位的取值：检测位的取值与该位所在的检测“小组”中承担的奇偶校验任务有关系。

假设我们后面都以 1 为偶数个来检测，那么各检测位 Ci 承担的检测小组为：

- C_1 检测的 g1 小组包含第 1/3/5/7/9/11……，位置的二进制编码为 XXX…XXX1

- C_2 检测的 g1 小组包含第 2/3/6/7/10/11……，位置的二进制编码为 XXX…XX1X
- C_4 检测的 g1 小组包含第 4/5/6/7/12/13……，位置的二进制编码为 XXX…X1XX
- C_8 检测的 g1 小组包含第 8/9/10/11/12/13……，位置的二进制编码为 XXX…1XXX

gi 小组独占第 2^{i-1} 位，位置的二进制编码为 0…10…0，gi 和 gj 小组共同占第 $2^{i-1}+2^{j-1}$ 位，位置的二进制编码为 0…010…010…0。以此类推。

例 4.4 求 0101 按“偶校验”配置的汉明码

解析：因为 n=4，要满足公式（2.1），所以 k=3。故汉明码排序如下：

二进制序号	1	2	3	4	5	6	7
名称	C1	C2	0	C4	1	0	1
编码结果	**0**	**1**	0	**0**	1	0	1

C1 的位置要使第 1 组 1 的个数为偶数个，第 1 组包含了第 1/3/5/7 个元素，这里面 1 为 2 个，已经为偶数，所以 C1 为 0。

C2 的位置要使第 2 组 1 的个数为偶数个，第 1 组包含了第 2/3/6/7 个元素，这里面 1 为 1 个，需要补为偶数个，所以 C2 为 1。

C4 的位置要使第 4 组 1 的个数为偶数个，第 4 组包含了第 4/5/6/7 个元素，这里面 1 为 2 个，已经为偶数，所以 C4 为 0。

故最后的编码结果为：**0100101**。

练习 1：按配偶原则配置 0011 的汉明码。

解析：因为 n=4，同理可得 k=3。故汉明码排序如下：

二进制序号	1	2	3	4	5	6	7
名称	C1	C2	0	C4	0	1	1
编码结果	**1**	**0**	0	**0**	0	1	1

C1 的位置要使第 1 组 1 的个数为偶数个，第 1 组包含了第 1/3/5/7 个元素，这里面 1 为 1 个，所以 C1 为 1。

C2 的位置要使第 2 组 1 的个数为偶数个，第 1 组包含了第 2/3/6/7 个元素，这里面 1 为 2 个，所以 C2 为 0。

C4 的位置要使第 4 组 1 的个数为偶数个，第 4 组包含了第 4/5/6/7 个元素，这里面 1 为 2 个，已经为偶数，所以 C4 为 0。

所以 0011 的汉明码为 **1000011**。

4.汉明码的纠错过程

形成新的检测位 Pi，其位数与增添的检测位有关，如果增添 3 位（k=3），新的检测位为 P4P2P1.以 k=3 为例，Pi 的取值为图 2.27 所示。

$$P_1 = \overset{C_1}{1} \oplus 3 \oplus 5 \oplus 7$$
$$P_2 = \overset{C_2}{2} \oplus 3 \oplus 6 \oplus 7$$
$$P_4 = \overset{C_4}{4} \oplus 5 \oplus 6 \oplus 7$$

图 2.27 k=3 时 pi 取值

对于按偶校验配置的汉明码，不出错时 P1=0/P2=0/P4=0。

例 4.5 已知接收到的汉明码为 0100111（按配偶原则配置），试问要求传送的信息是什么？

解析：纠错过程如下：

$$P_1 = 1 \oplus 3 \oplus 5 \oplus 7 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$
$$P_2 = 2 \oplus 3 \oplus 6 \oplus 7 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$
$$P_4 = 4 \oplus 5 \oplus 6 \oplus 7 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

所以 P4P2P1=110，即第 6 位出错，纠正后结果为 0100101，也就是传送信息为 0101。

练习 2：写出偶校验配置的汉明码 0101101 的纠错过程。

解析：纠错过程如下：

$$P_1 = 1 \oplus 3 \oplus 5 \oplus 7 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$
$$P_2 = 2 \oplus 3 \oplus 6 \oplus 7 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$$
$$P_4 = 4 \oplus 5 \oplus 6 \oplus 7 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

所以 P4P2P1=100，也就是第 4 位错误，可以不纠正。

练习 3：按配奇原则配置 0011 的汉明码

解析：因为 n=4，同理可得 k=3。故汉明码排序如下：

二进制序号	1	2	3	4	5	6	7
名称	C1	C2	0	C4	0	1	1
编码结果	**0**	**1**	0	**1**	0	1	1

C1 的位置要使第 1 组 1 的个数为奇数个，第 1 组包含了第 1/3/5/7 个元素，这里面 1 为 1 个，所以 C1 为 0。

C2 的位置要使第 2 组 1 的个数为奇数个，第 1 组包含了第 2/3/6/7 个元素，这里面 1 为 2 个，所以 C2 为 1。

C4 的位置要使第 4 组 1 的个数为奇数个，第 4 组包含了第 4/5/6/7 个元素，这里面 1 为 2 个，所以 C4 为 1。

所以配置的汉明码为：**0101011**。

遗留问题：

（1）汉明编码最小距离（汉明距离）为：

（2）如果是采用配置原则，纠错过程和按偶配置有什么区别？

七、提高访存速度的措施

- 采用高速器件
- 采用层次结构：Cache-主存
- 调整主存结构

1.单体多字系统

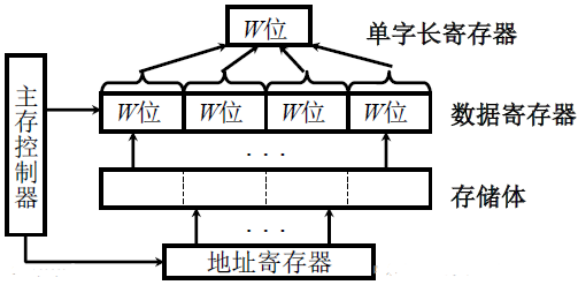

图 2.26 单体多字系统

把存储器的存储字长加长。比如说 CPU 是 32 位，存储器的存储字可以是 64 位。CPU 一次性把这些字的信息从存储器中取出放到数据寄存器中，然后来使用，这种方法可以**增加存储器的带宽**。但是这种结构也存在问题，比如说 CPU 想往存储体中写信息，首先要把一

个字长的信息写到单字长寄存器，然后再写到数据寄存器中，然后写入存储器中，这样的问题是如果我们只需要一个字长的信息，但是数据寄存器还是会把 4 个字长的信息写到存储体中，会造成错误的修改；如果要取数据不是连续存放在一个存储字中，比如第一条是跳转指令且跳转较远，那么放在数据寄存器的 4 个存储字只有 1 个是有用处的，其它都没有作用。虽然每一个存储单元有 4 个存储字，但是它们是连续的，会有问题。

2.多体并行系统

（1）高位交叉——顺序编址

假设我们有 4 个存储体，从 M0 到 M3。所谓高位交叉，就是我们对这 4 个存储体构成的存储单位编址的时候，比如说从 M0 开始，顺序对 M0 进行编址，里面就是从 0000 到 1111，那么在前面加 2 位作为对存储体进行编址，M0 为 00，M1 为 01，M2 为 10，M3 为 11。其内部都是从 0000 到 1111，如图 2.27 所示。

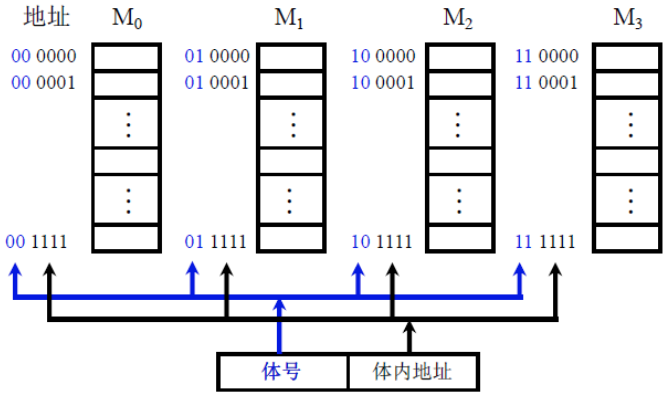

图 2.27 高位交叉图 1

如果说每个存储体都有自己的 MDR 和 MAR，那么 4 个存储体就可以进行并行工作了。这种方法使的每个存储体都有自己的控制电路进行操作，4 个存储体就有了并行操作的基础。这种方法的问题如果用户使用一个程序，程序在计算机中是按顺序进行存储，如果某条指令存在一个位置，那么下一个指令就存在接下来的位置，指令按序执行，在执行过程中，m0 被 CPU 不停访问，其它存储体都是空闲。所以这种方法会造成某个存储体非常繁忙，别的空闲。

在这个方法中，高 2 位为选择存储器，而后面 4 位是选择存储器中的存储单元，这种就是我们之前所说的存储器的扩展，所以它适用于存储器容量扩展，不适合提高存储器的带宽。图 2.28 给出了这种方法更为通用的方式。

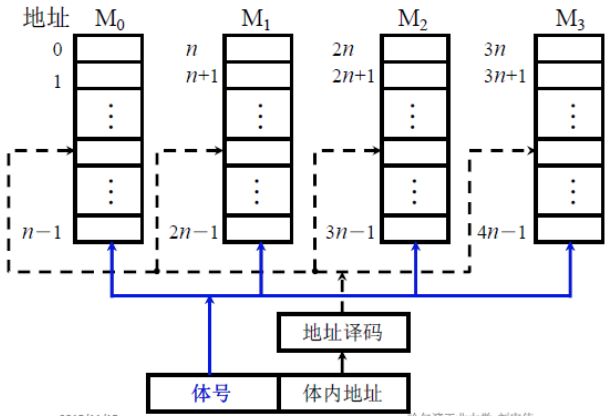

图 2.28 高位交叉通用方式

（2）低位交叉——各个体轮流编址

这种方式就是解决高位交叉存在的问题。编码采用低位编码的方式，如图 2.29 所示。采用横向编码方式，M0 第一个地址为 000000，M1 第一个地址为 000001，M2 第一个地址为000010，M3 第一个地址为000011；M0 第二个地址为000100，M1 的第二个地址为000101，M2 的第二个地址为 00110，M3 的第二个地址为 000111……以此类推。对于 M0 来说，后两位是 00，对于 M1 来说，后两位是 01，对于 M2 来说后两位是 10，对于 M3 来说后两位是 11。可以这样说，地址的后两位给出了存储体的地址，前面 4 位给出了每个存储体内部地址。

CPU 给出一个地址，低位（2 位）表示存储体的地址，高位（4 位）表示存储体内部地址。如果 4 个存储体完全独立，每个控制体都有自己的电路，可以进行地址、数据、控制的锁存，那么 4 个存储体就可以并行轮流操作。

那么，如果进行存放，第一个部分存储在 000000，第二部分就存放在 000001，第三部分存放在 000010……以此类推。这样 4 个存储体都得到了使用。如果进行读取的时候也是同样进行读取，如果 M0 准备好了数据，就可以向 CPU 进行数据传送。称为**分离式通信。**

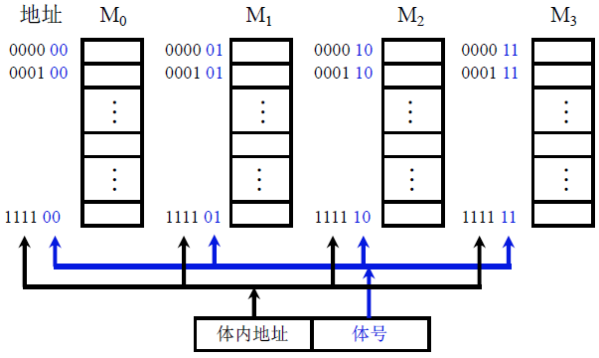

图 2.29 低位交叉图 1

低位交叉的特点：**在不改变存取周期的前提下，增加存储器的带宽。**如图 2.30 所示。

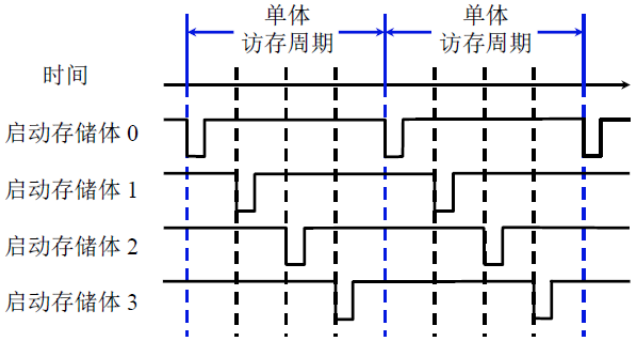

图 2.30 低位交叉时序图

在一个单体访存周期，我们将其分为 4 个时间段。第一个周时间段启动存储体 M0，在第 2 个时间段启动 M1，第 3 个时间段启动 M2，第 4 个时间段启动 M3。

实际上这个方式相当以流水方式来访问。

设四体低位交叉存储器，存取周期为 T，总线传输周期为 τ，为实现流水线方式存取，应满足 **T=4 τ** 。连续读 4 个字所需要的时间为 T+（4-1） τ 。

低位交叉用于带宽提高和访问速度。

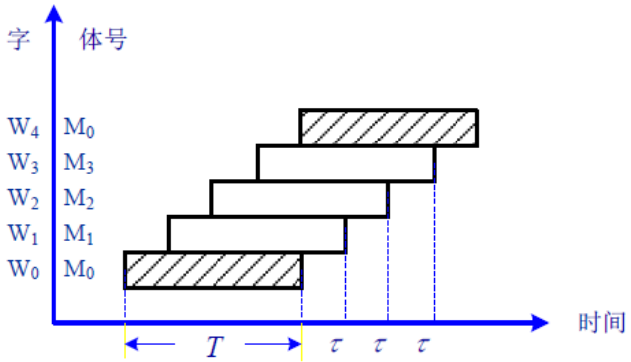

图 2.31 示意图

3.高性能存储芯片

（1）SDRAM（同步 DRAM）：在系统时钟的控制下进行读出和写入，CPU 无需等待。

（2）RDRAM：解决存储器带宽问题。

（3）带 Cache 的 DRAM。在 DRAM 的芯片中**集成**了一个由 **SRAM** 组成的 **Cache**，有利于**猝发式读取**。