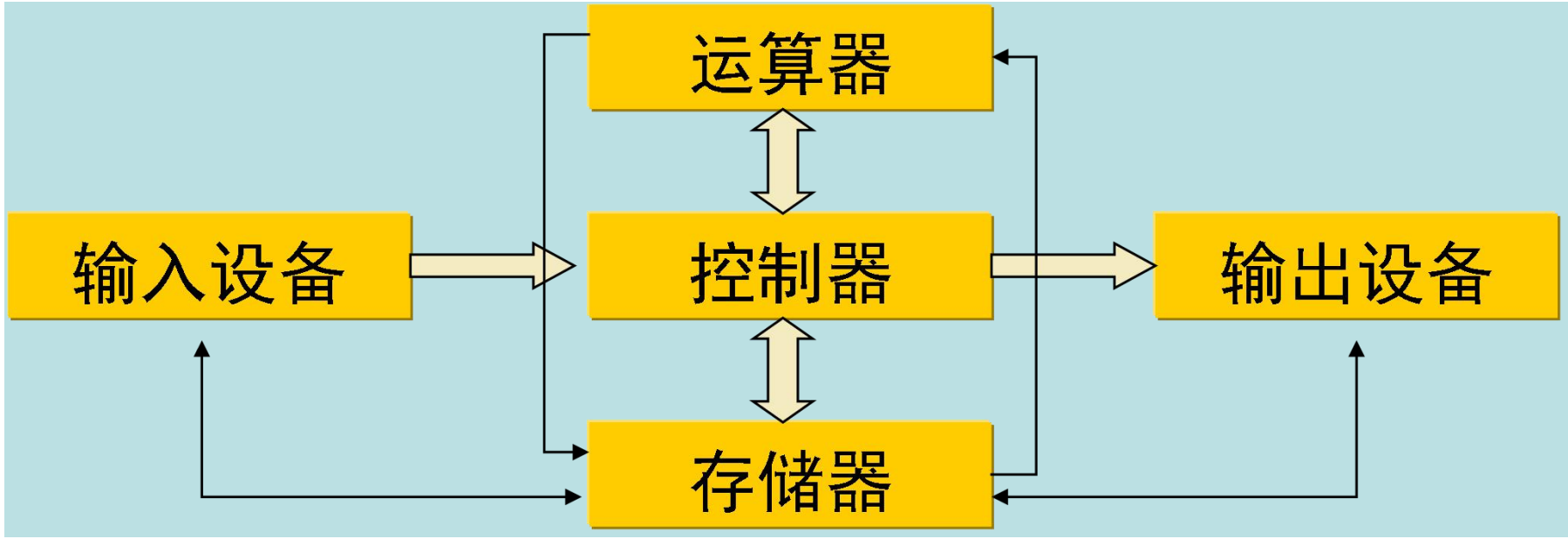
第四章　存储系统

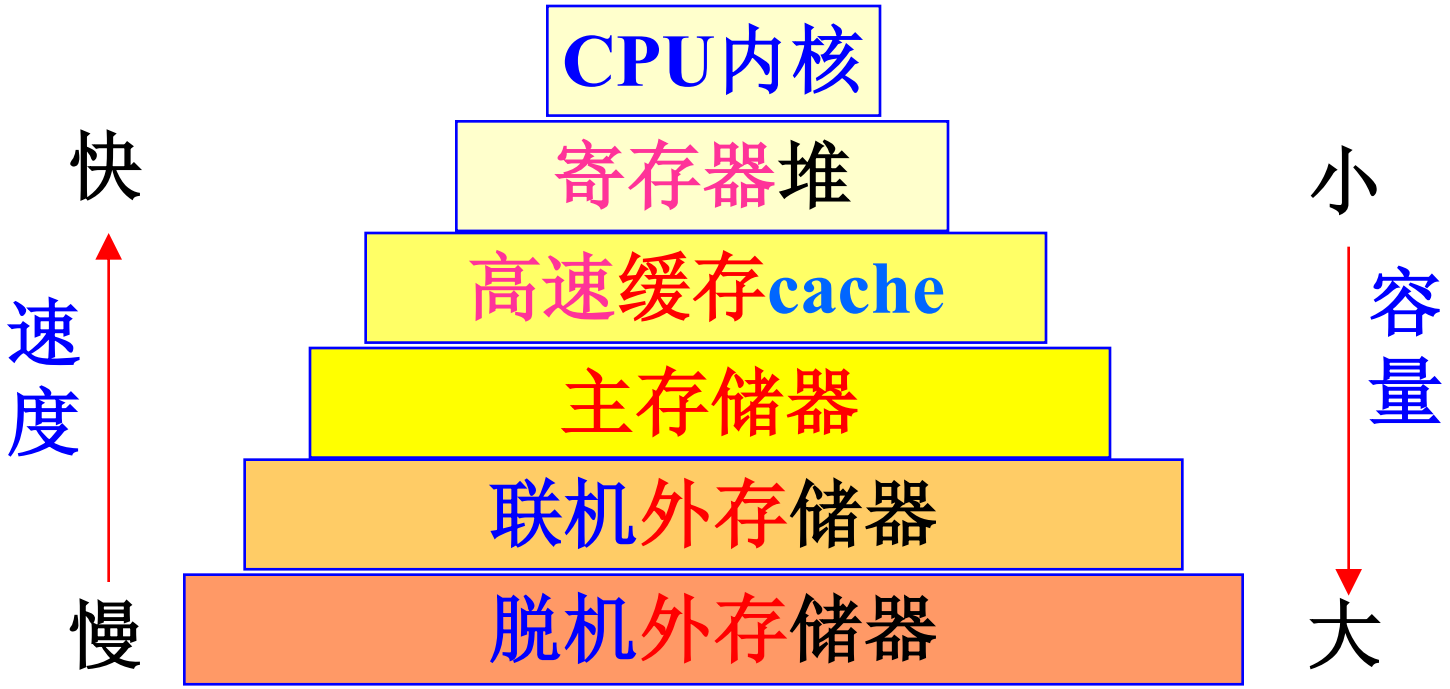
本章主要内容

存储系统、分类及其特点



存储器的层次结构

微机拥有不同类型的存储部件(多层/多级结构)。



§4.1 特殊存储部件

由寄存器（D触发器）构成。CPU内部，用于缓存和传输操作，容量小，速度快。

特殊存储部件
- 通用寄存器堆
- 寄存器队列
- 寄存器栈

4.1.1 寄存器堆（缓冲存储器）

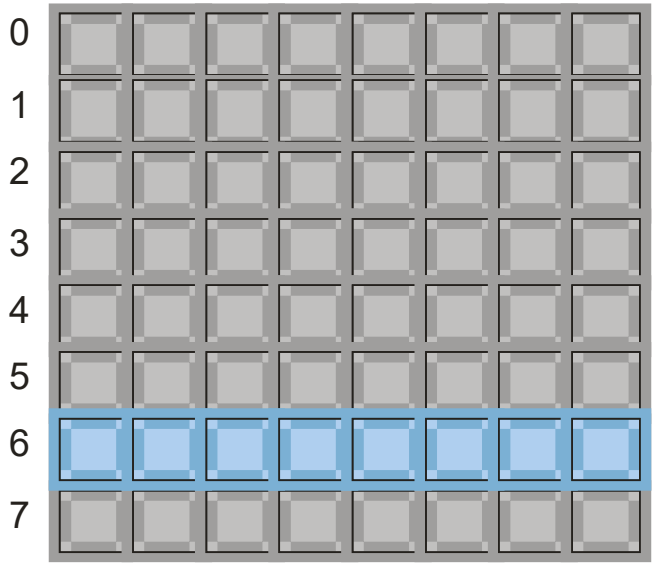
由多个寄存器构成的集合。常用于数据寄存。

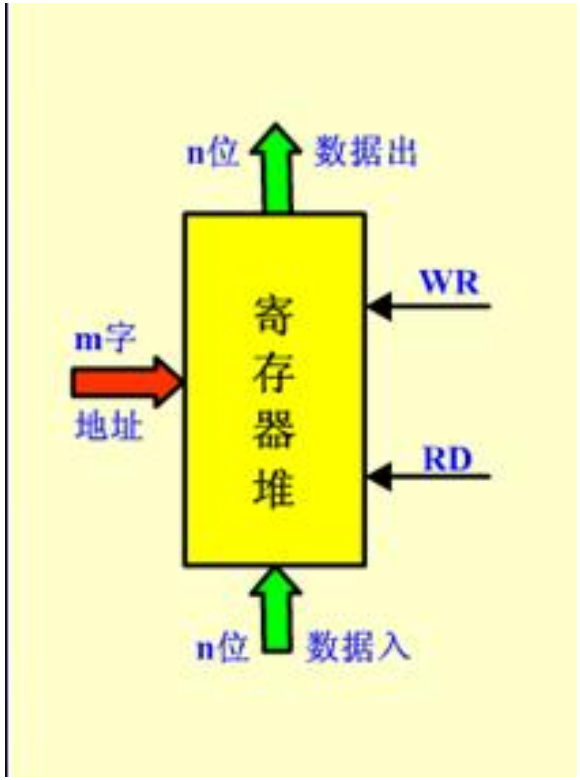
有三组外部信号：地址（短地址）、数据、读/写控制。

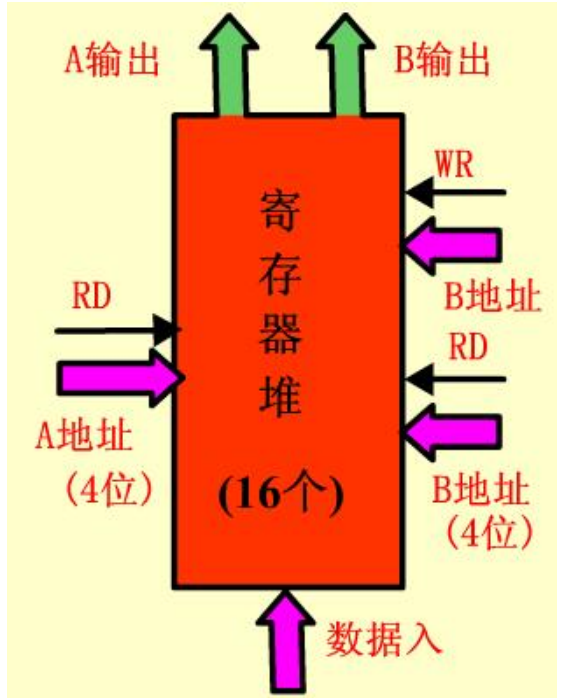
多端口寄存器。

可同时读、写，

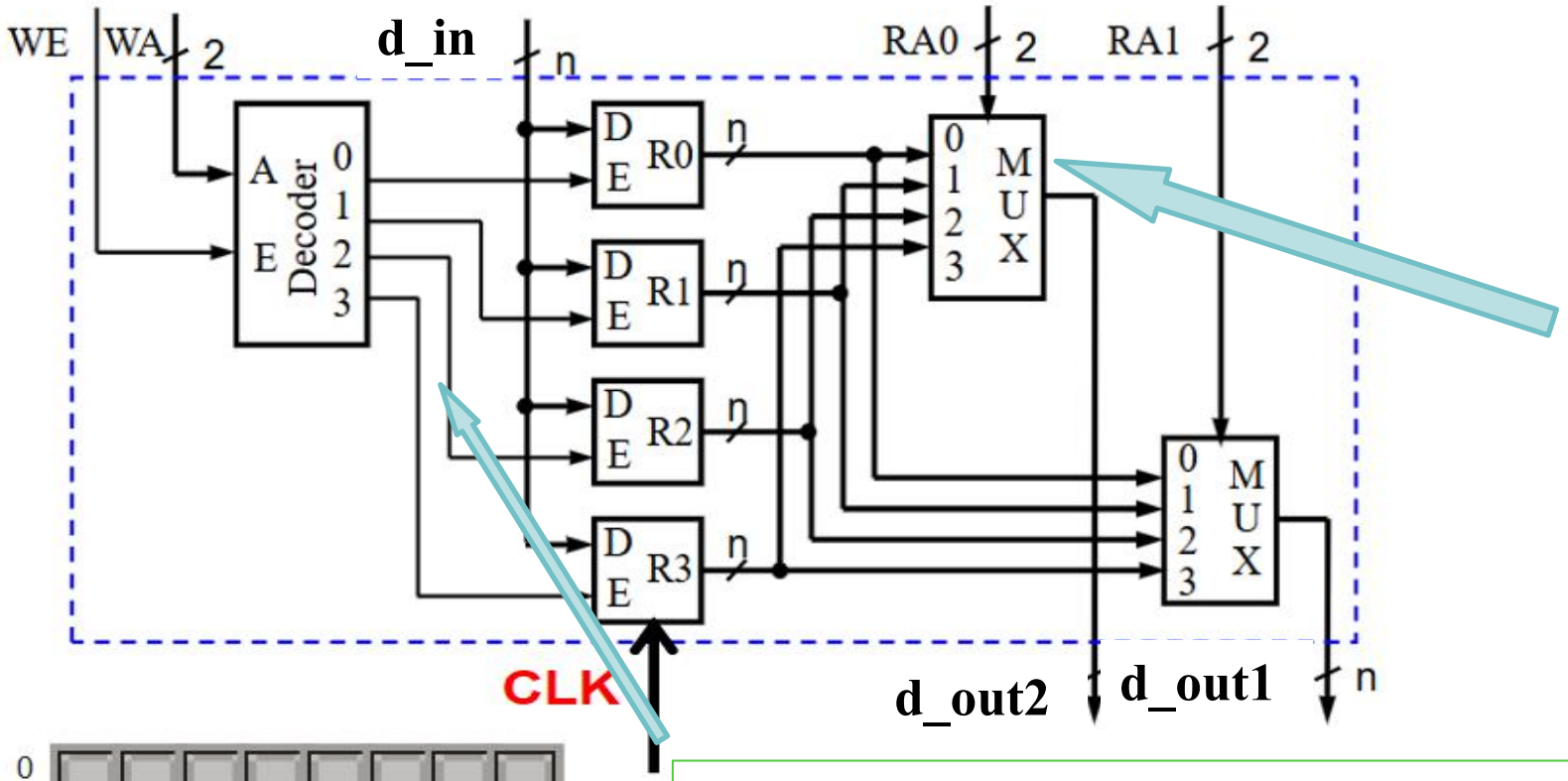
可同时输出两个数。

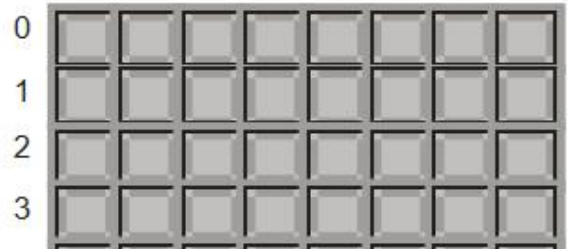







```
reg [7:0] regs [0:7] ;
```

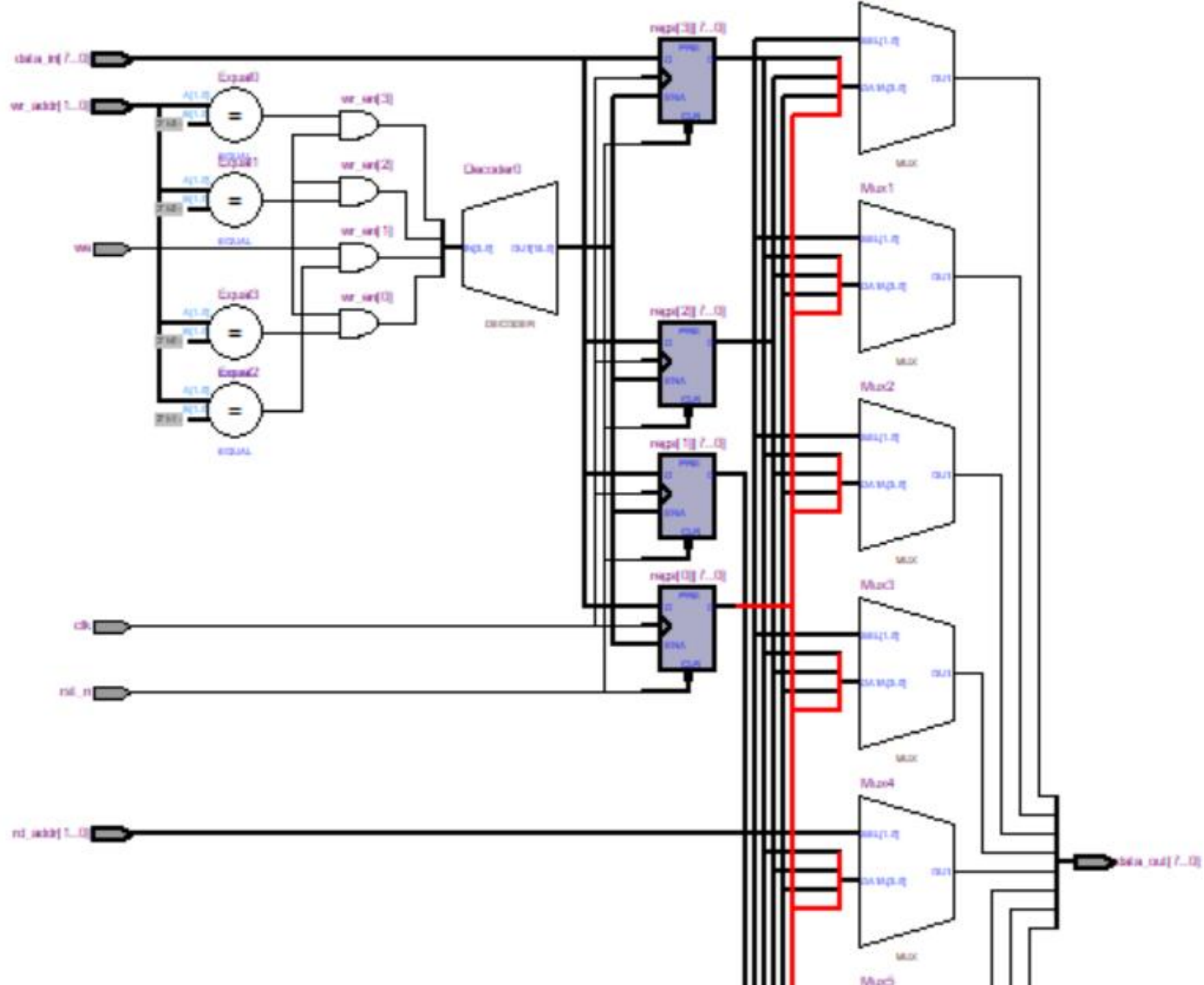





```
input [7:0] d_in, d_out;
reg [7:0] regs [0:3] ;
```

```
always @( ra )
// 数据选择器，用于选择输出数据
    case (ra)
        2'b00: d_out = regs[0];
```

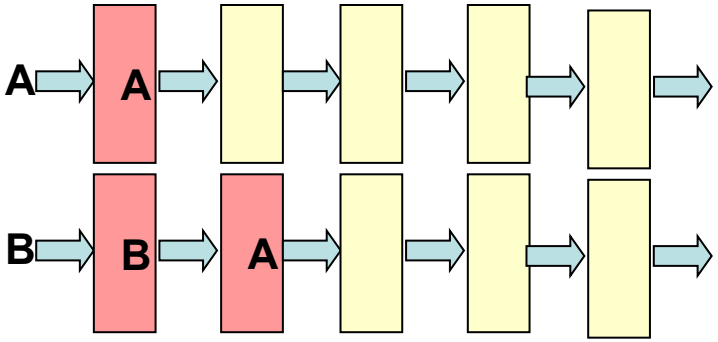
```
wire [3:0] we_wa;
assign we_wa = {we&&(wa==2'b11), we&&(wa==2'b10), we&&(wa==2'b01), we&&(wa==2'b00) };  //写使能与地址译码
always ( posedge clk , negedge rst_n)
   ……                  // 寄存器堆异步复位
   case (we_wa)      // 根据写使能及地址写入数据
         4'b0001: regs[0] <= d_in;
```


Equal0
Equal1
Equal3
Equal2
wr_en[3]
wr_en[2]
wr_en[1]
wr_en[0]
Decoder0
reg4[3] [7...0]
reg4[2] [7...0]
reg4[1] [7...0]
reg4[0] [7...0]
Mux1
Mux2
Mux3
Mux4
clk
rst_n
rd_addr[1...0]
data_out[7...0]

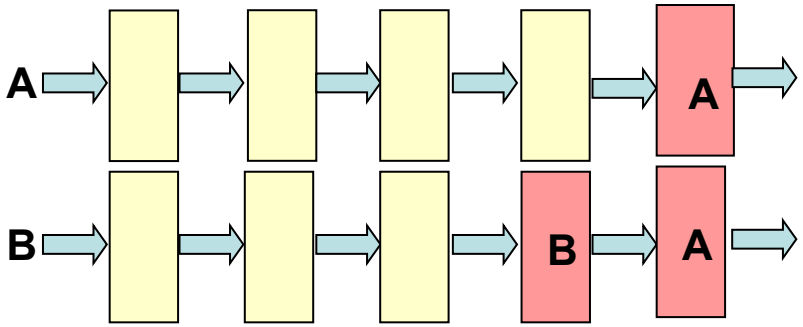
4.1.2 寄存器队列（FIFO)

FIFO（First In First Out，先进先出），用若干个移位寄存器构建的小型存储部件。用于指令队列。

无地址线，双端口存储器，可同时读写。



常规移位寄存器



FIFO寄存器

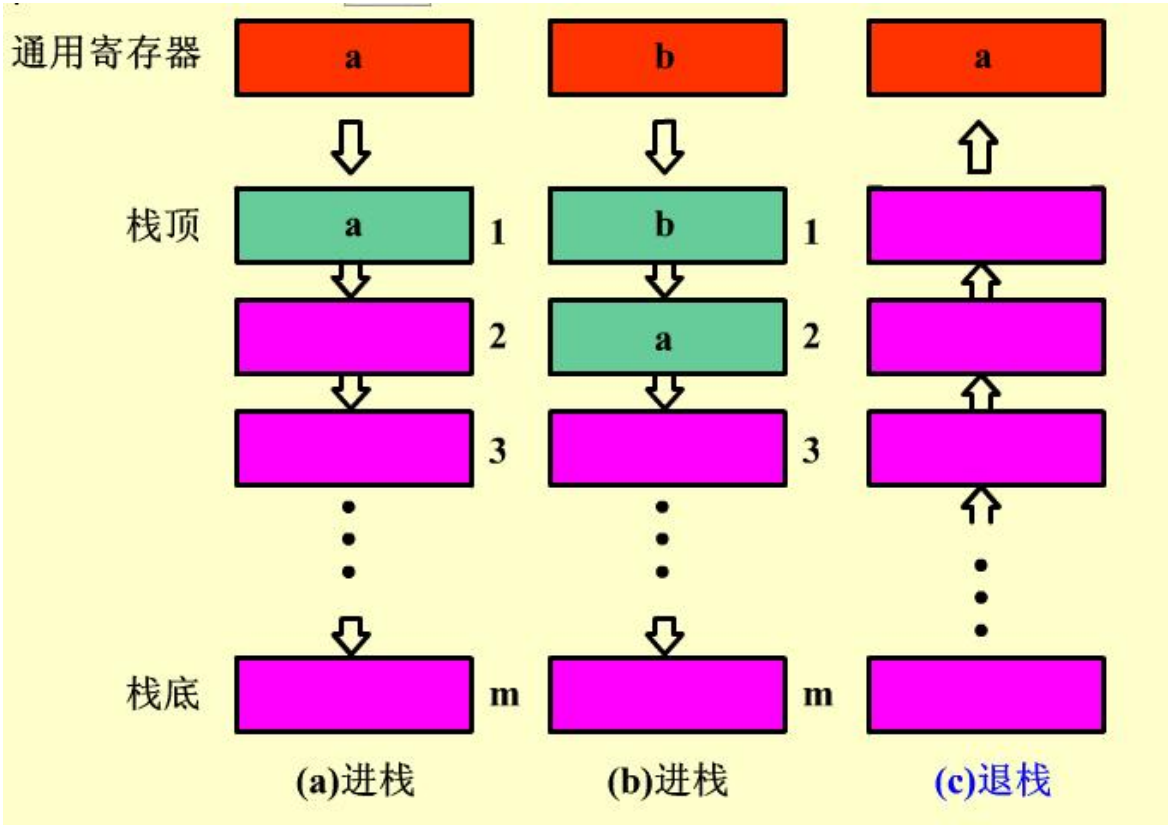
FIFO用于两个不同系统通信、数据采集传送、串并转换。

4.1.3 寄存器栈（LIFO)

LIFO（ **Last In First Out**，后进先出）方式，用若干个双向移位寄存器构建的小型存储部件。

用于减少函数调用时对内存的访问。

双向移位寄存器构成



§4.2 随机读写存储器RAM

存储器：大容量存储，由TTL或MOS构成。

内存/主存储器

随机读写存储器

Random Access Memory （RAM）

特性：能读能写、易失

作用：存放编写的程序和数据

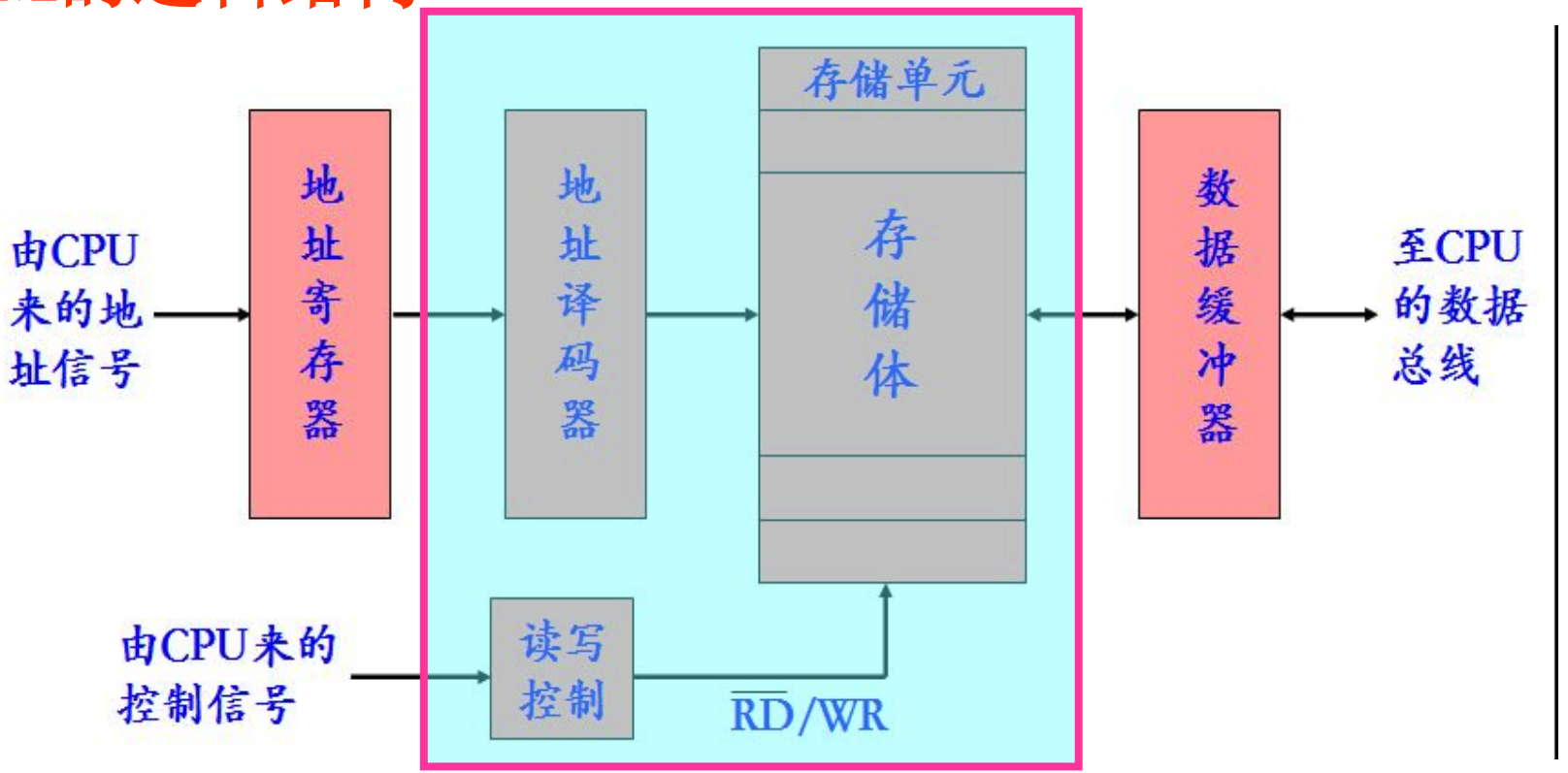
只读存储器

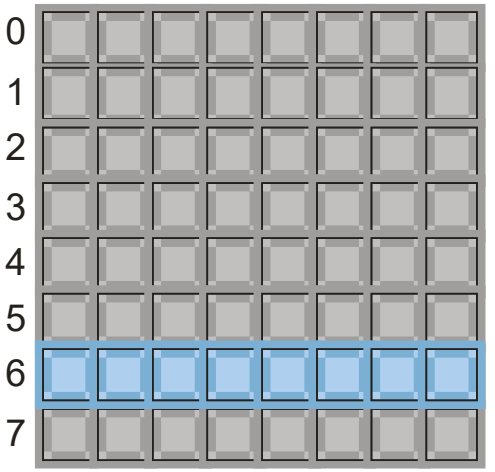
Read Only Memory （ROM）

特性：只能读不能写、不易失

作用：存放固定的程序和数据

4.2.1 RAM的逻辑结构



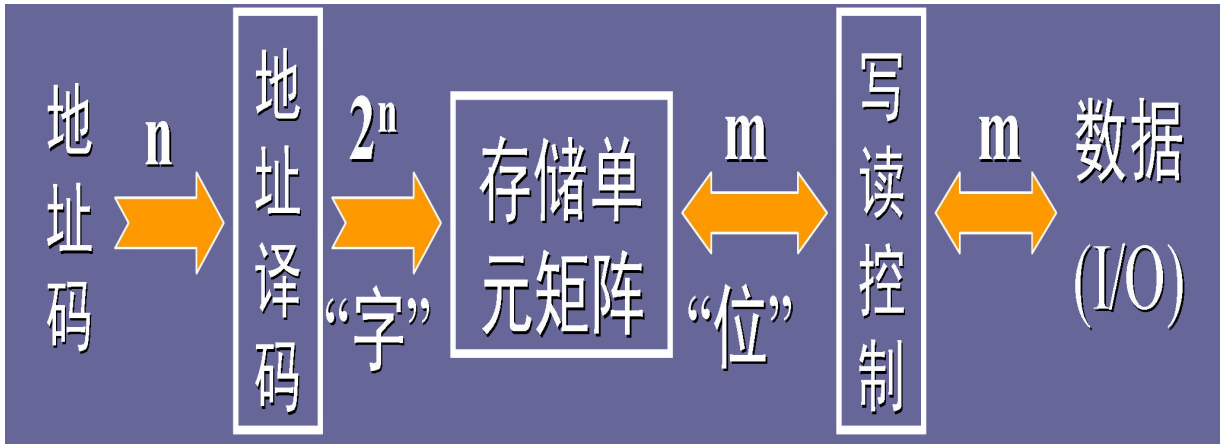


存储器的外部连线：

- 地址线（单向）
- 数据线是（双向）
- 控制线（单向且读、写互斥）
- 片选线

4.2.2 存储器地址译码方式

- 单译码方式
- 双译码方式

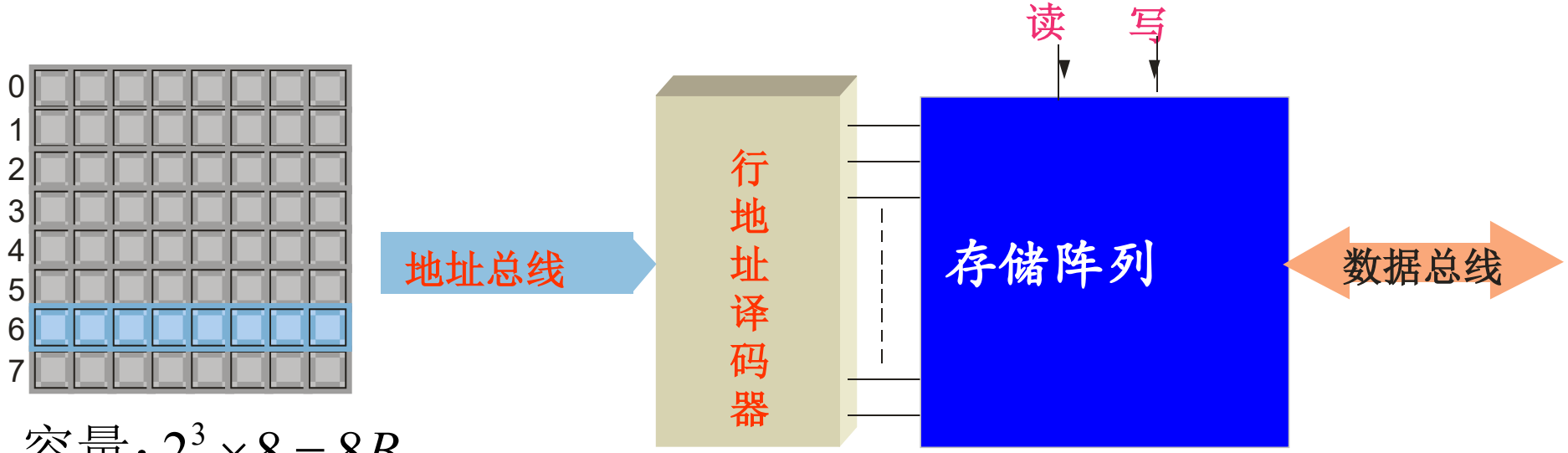


1 单译码方式

一个译码器，选中一个字的所有存储元。（二维存储阵列）

位线：数据线

字线：存储单元选通线

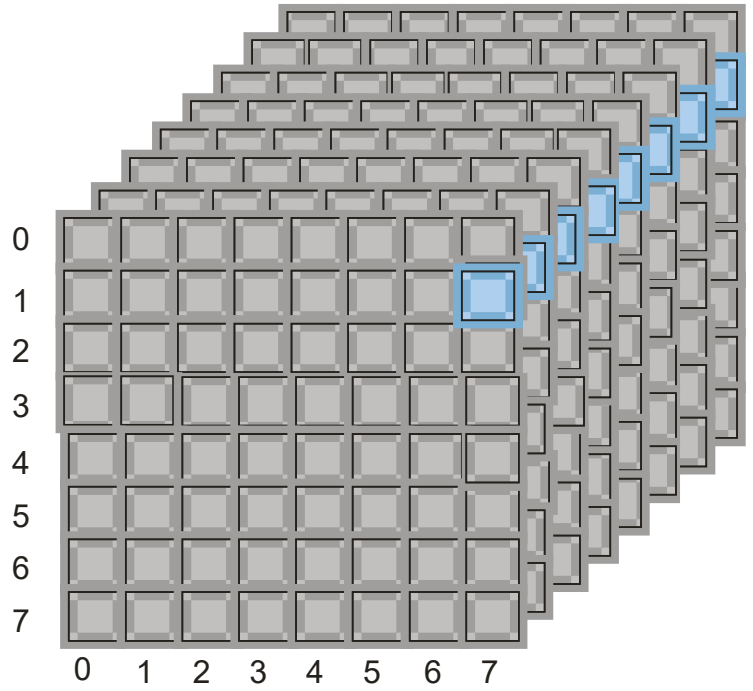


容量：$2^3 \times 8 = 8B$

2 双译码方式

两个译码器，选中交叉点的存储元。（三维存储阵列）

构成大容量存储器。



容量：$2^6 \times 8 = 64B$

3 存储器容量

单元数×每单元的位数

字数　　字长

单位：KB、MB、GB等，

其中K=2^{10}，M=2^{20}、G=2^{30}、B(Byte)=8bit

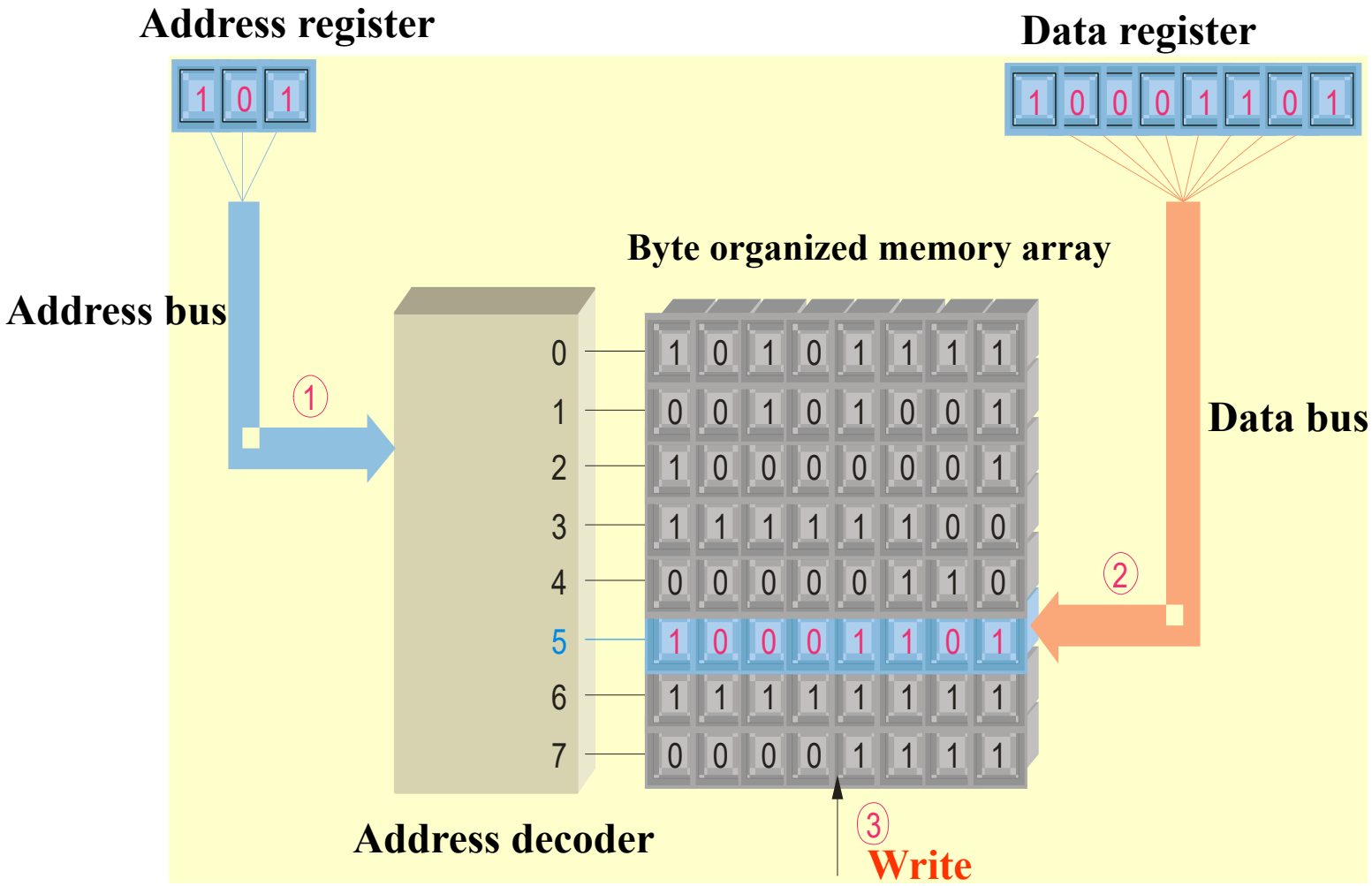
容量＝8KB=8K×8=2^{13}×8

说明：13根地址线

8根数据线

4 存储器的读写操作

1）写操作



1) 地址有效

2)数据放总线

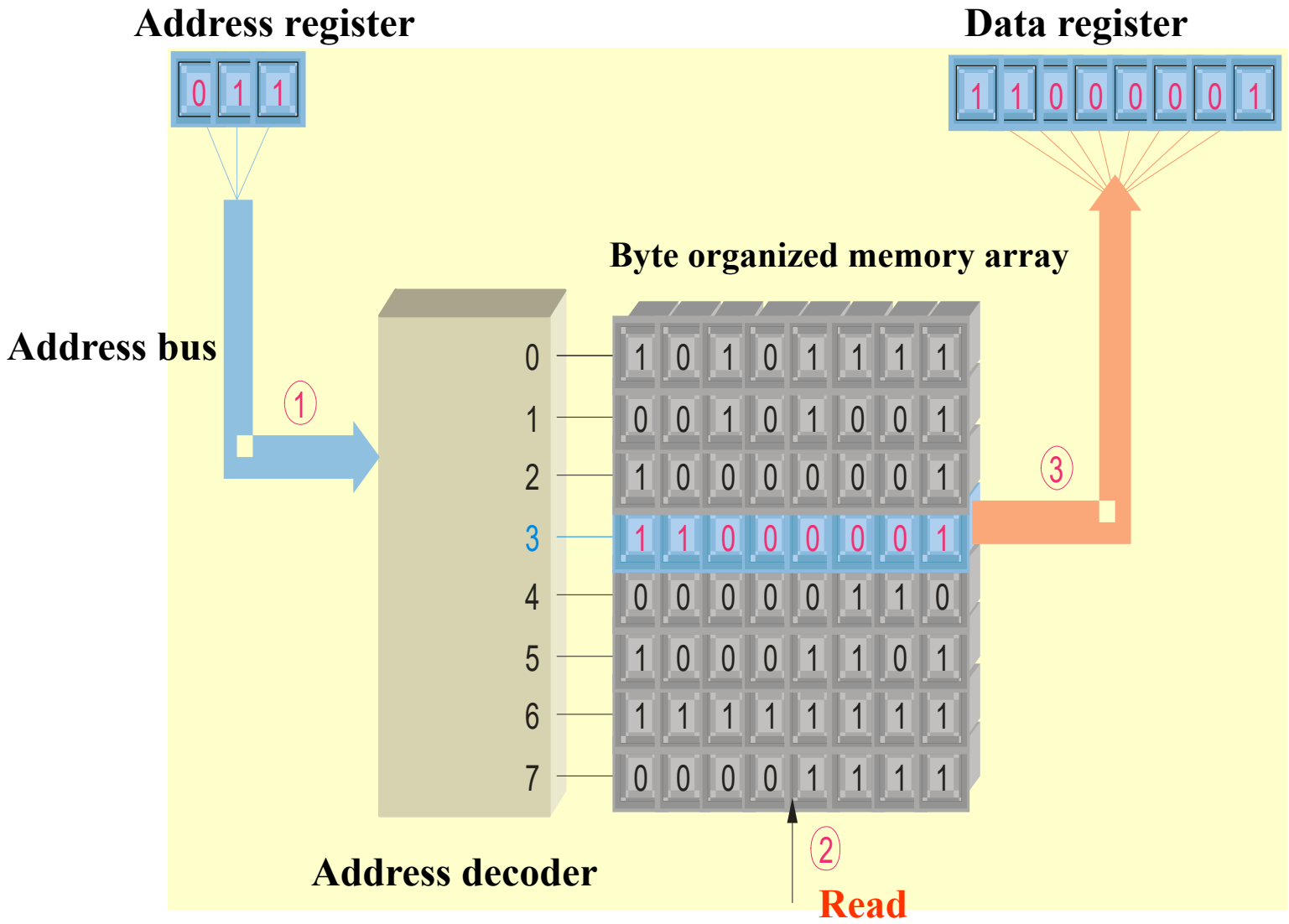
3)写命令有效

2）读操作

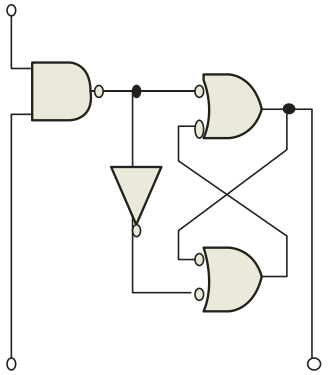
1) 地址有效

2) 读命令有效

3) 数据被复制到总线并打入寄存器



5 RAM的分类



Memory cell

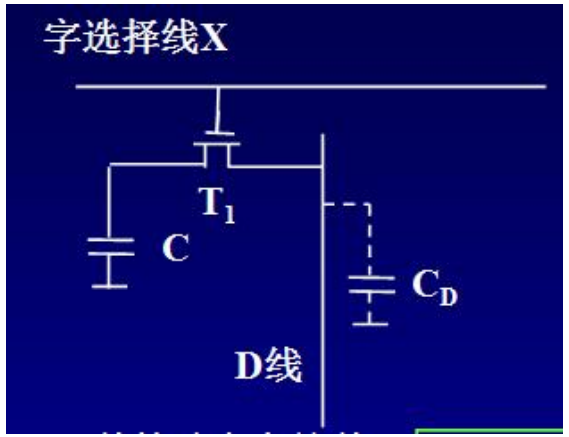
MOS型 RAM

SRAM （Static RAM）

单元：锁存器（6管MOS），存储信息稳定

特点: 存取快\功耗较高\价格贵

应用： PC机中的Cache



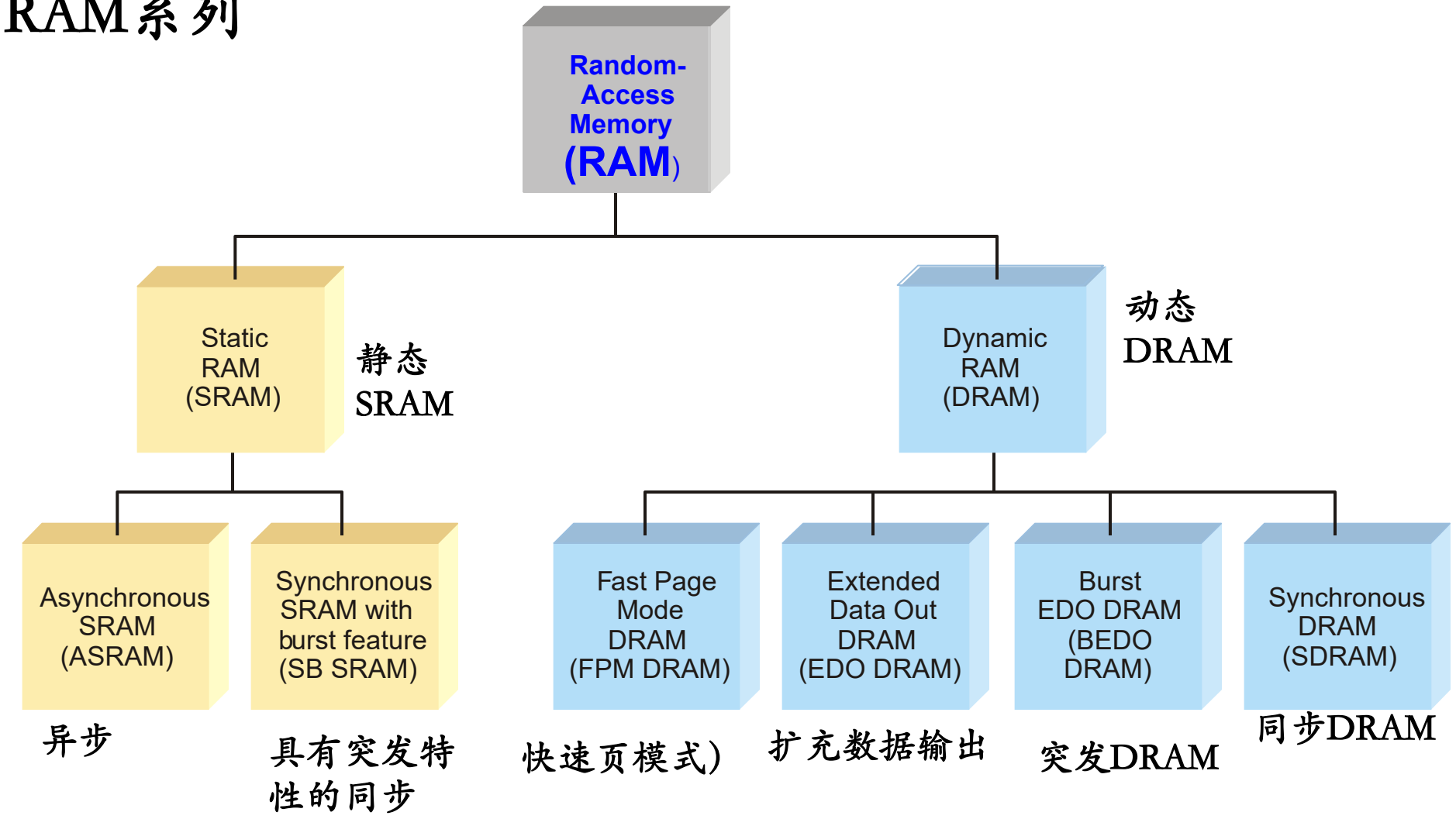
DRAM（Dynamic RAM）

单元：电容存储信息（1管＋1电容），漏电不稳定，需定时刷新，外围电路较复杂。

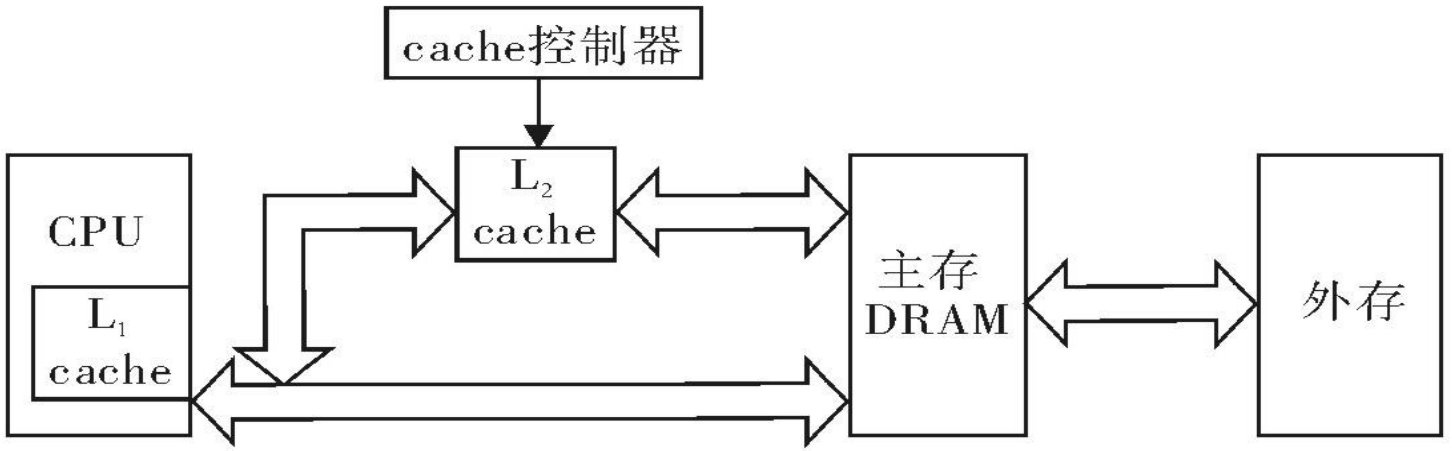
特点: 集成度高\功耗低\价格便宜

应用：微机中的内存条、显卡上的显存

RAM系列


Random-Access Memory (RAM)
Static RAM (SRAM)
静态SRAM
Dynamic RAM (DRAM)
动态DRAM
Asynchronous SRAM (ASRAM)
Synchronous SRAM with burst feature (SB SRAM)
Fast Page Mode DRAM (FPM DRAM)
Extended Data Out DRAM (EDO DRAM)
Burst EDO DRAM (BEDO DRAM)
Synchronous DRAM (SDRAM)
异步
具有突发特性的同步
快速页模式)
扩充数据输出
突发DRAM
同步DRAM

	组成单元	速度	集成度	应用
SRAM	触发器	快	低	小容量系统
DRAM	极间电容	慢	高	大容量系统



4.2.3 SRAM存储器

靠触发器记忆数据。

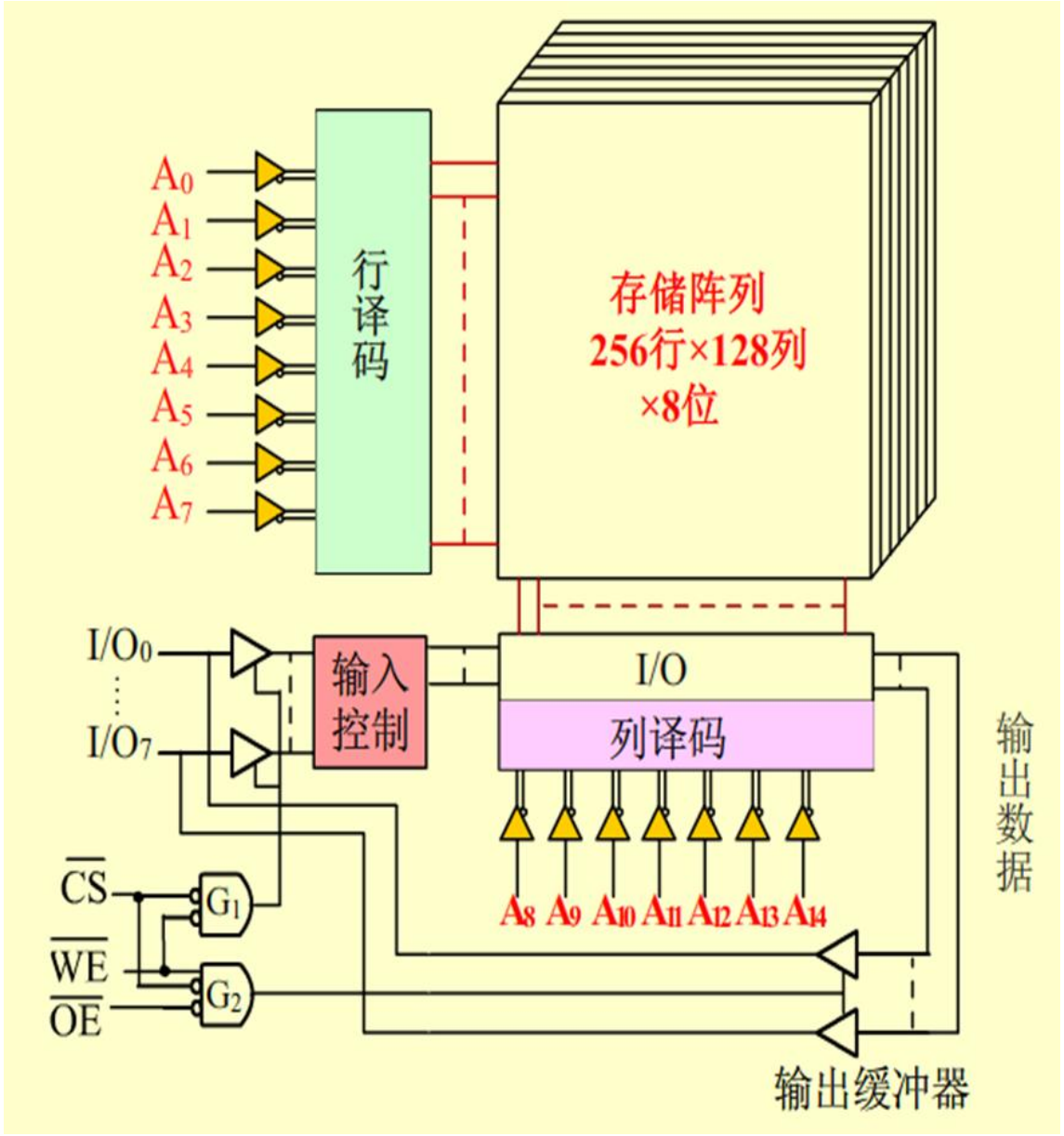
SRAM 工作过程：

读周期：

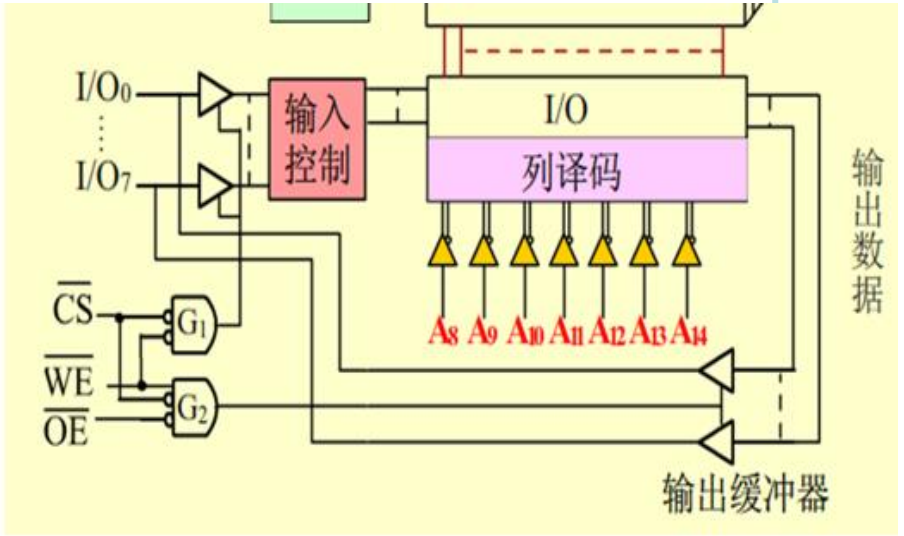
- 地址有效
- 片选信号CS有效　Chip select
- 输出使能OE有效
- 数据输出（三态门）到总线

写周期：

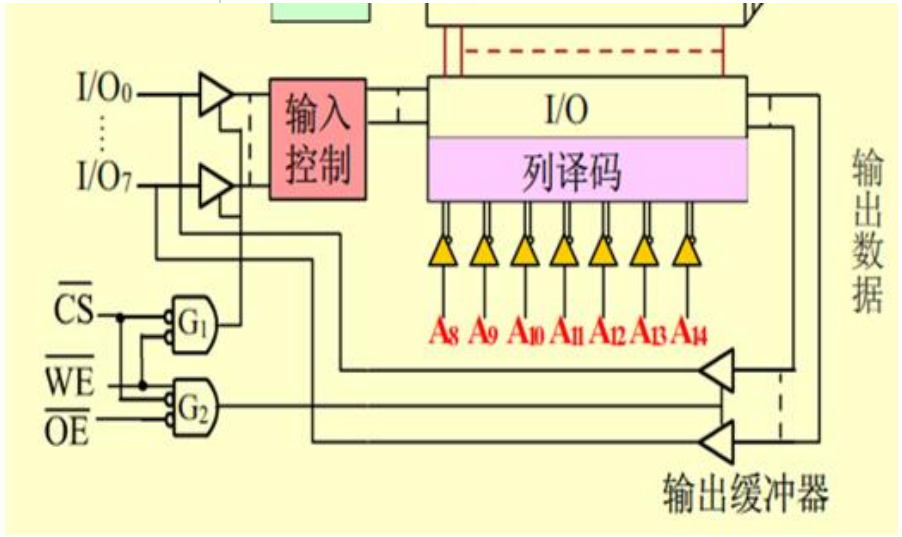
- 地址有效
- 片选信号CS有效
- 数据放置（三态门）到总线
- 写信号WE有效



```
module memory_e (  cs ,oen ,wen ,addr, data );
  parameter   DATA_WIDTH = 8;        // 数据宽度
  parameter   ADDR_WIDTH = 2;        // 地址宽度
  parameter MEM_WIDTH = 4 ;      // 存储器个数
  input  cs , oen, wen  ;
  input  [ADDR_WIDTH-1:0]     addr;   // 地址输入
  inout  [DATA_WIDTH-1:0]    data ;   // 双向数据端口
  reg [DATA_WIDTH-1:0]  memory  [ 0:MEM_WIDTH-1 ];   // 存储器数组声明
  reg [DATA_WIDTH-1:0] data_out;        // 数据输出寄存器
  wire [DATA_WIDTH-1:0] data_in ;      // 数据输入缓冲

   assign data = data_out ;
   assign data_in = data ;    // 双向数据端口驱动逻辑
```



```
always @ ( cs, oen  )     // 存储器读操作
   begin
       if ( !cs && !oen )
          data_out = memory[addr]; // 从指定地址读
       else
          data_out =  {DATA_WIDTH{1'bz}} ;
   end

 always @ ( cs, wen )    //存储器写操作
   begin
       if ( !cs && !wen )
             memory[addr] = data_in;   / / 将数据写入指定地址
       end

endmodule
```



4.2.4 DRAM存储器

靠MOS管栅极电容记忆数据。

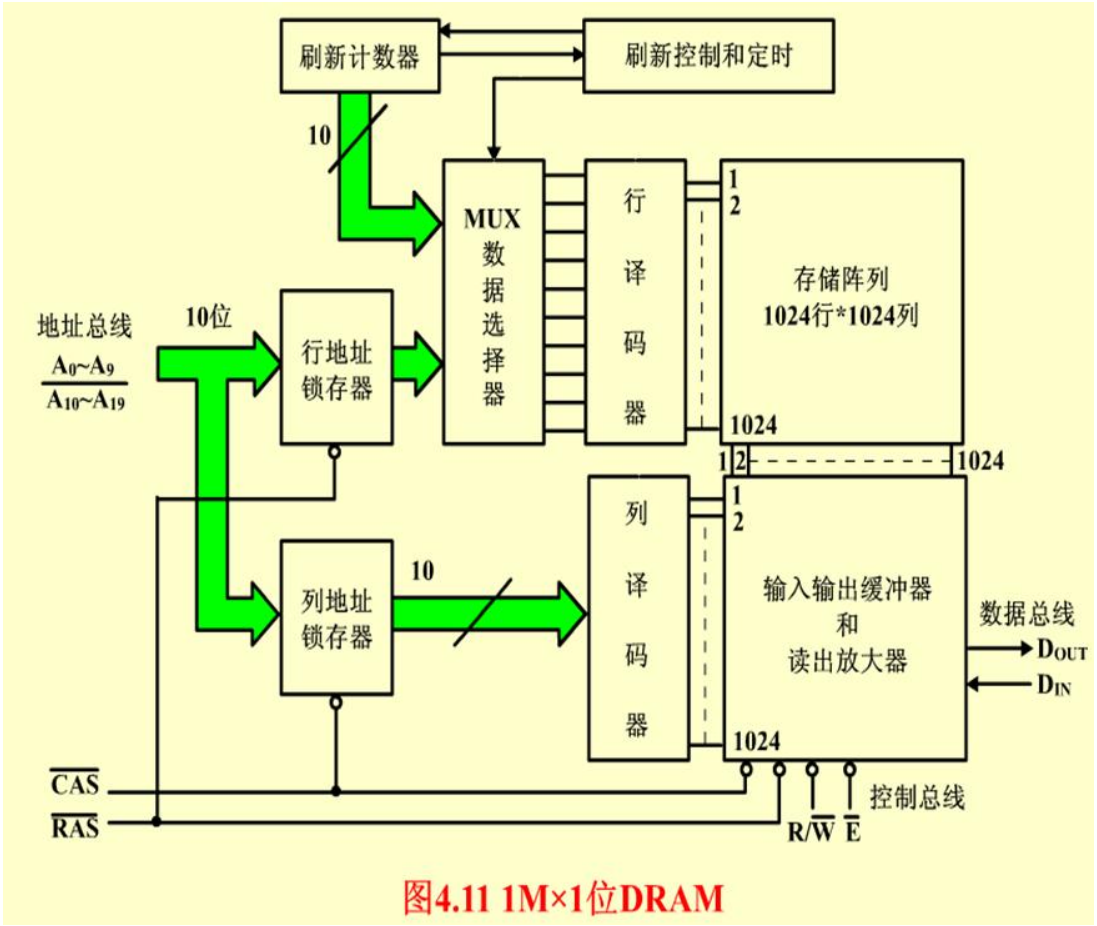
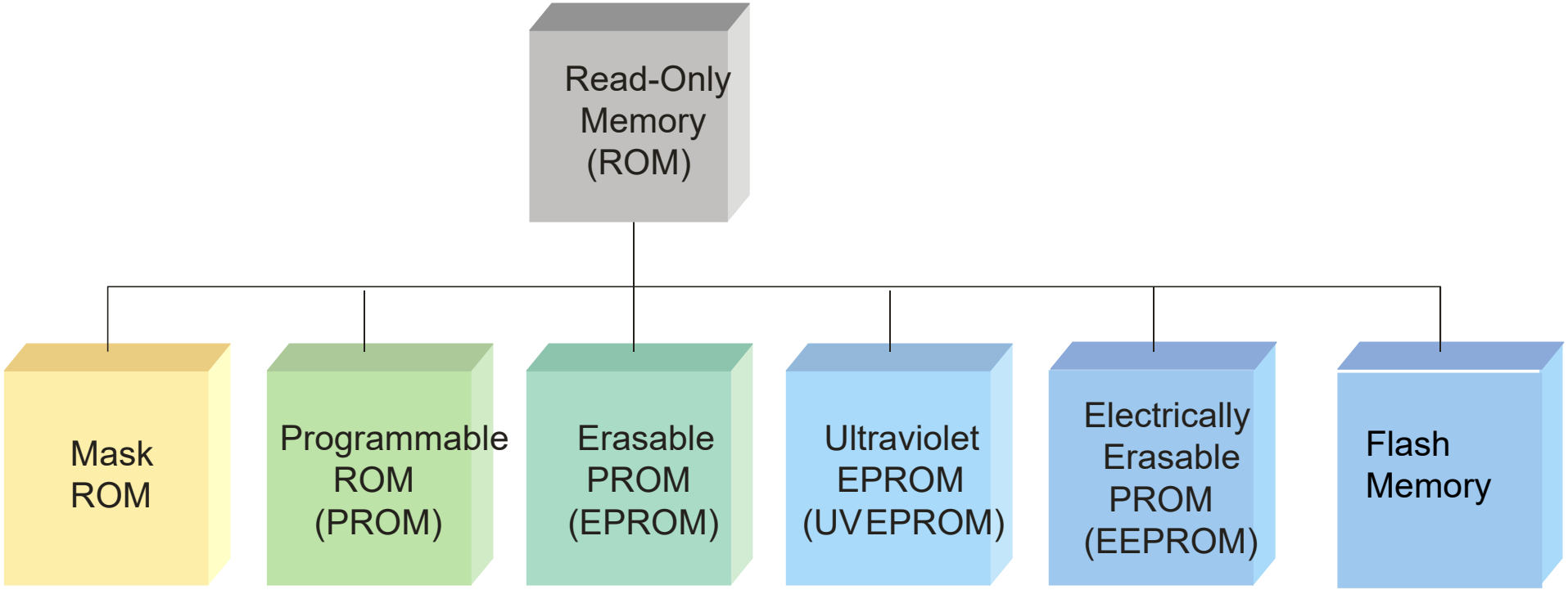


图4.11 1M×1位DRAM

- 刷新电路（将刚读出的数据再写回去）
- 写、读、刷新三个操作
- 地址线分两次送（减少总线数，减少芯片引脚）
- 读、写数据线分开

	SRAM	DRAM
存储信息	触发器	电容
破坏性读出	非	是
需要刷新	不要	需要
送行列地址	同时送	分两次送
运行速度	快	慢
集成度	低	高
发热量	大	小
存储成本	高	低

§4.3 只读存储器ROM



掩模ROM

PC机中存储BIOS程序、机器码、出厂信息。

可编程ROM

可擦除ROM

紫外线PROM

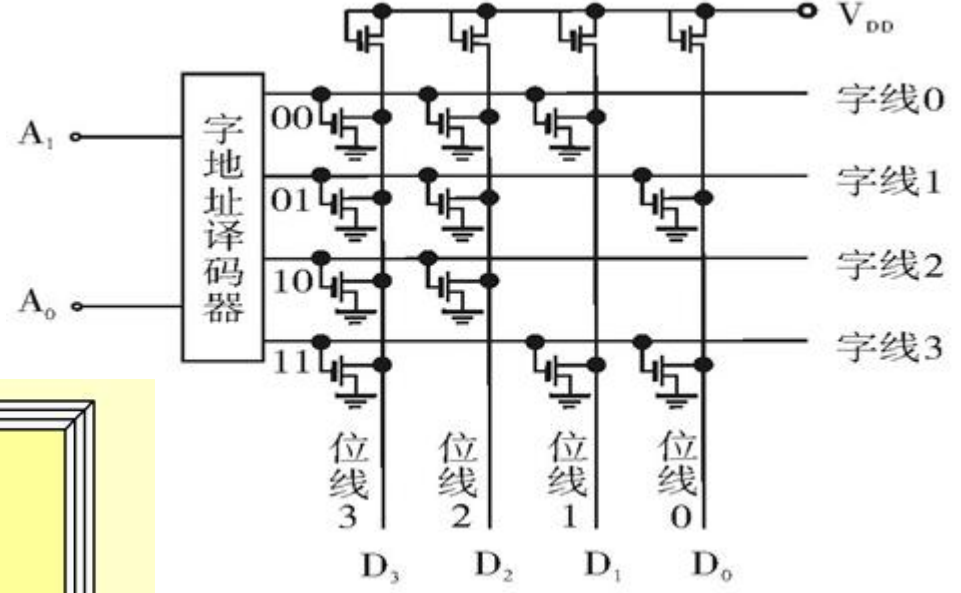
电可擦除E^2PROM

闪存

U盘

USB flash disk

1 ROM的内部组成



(a)掩模ROM逻辑符号

(b)内部逻辑框图

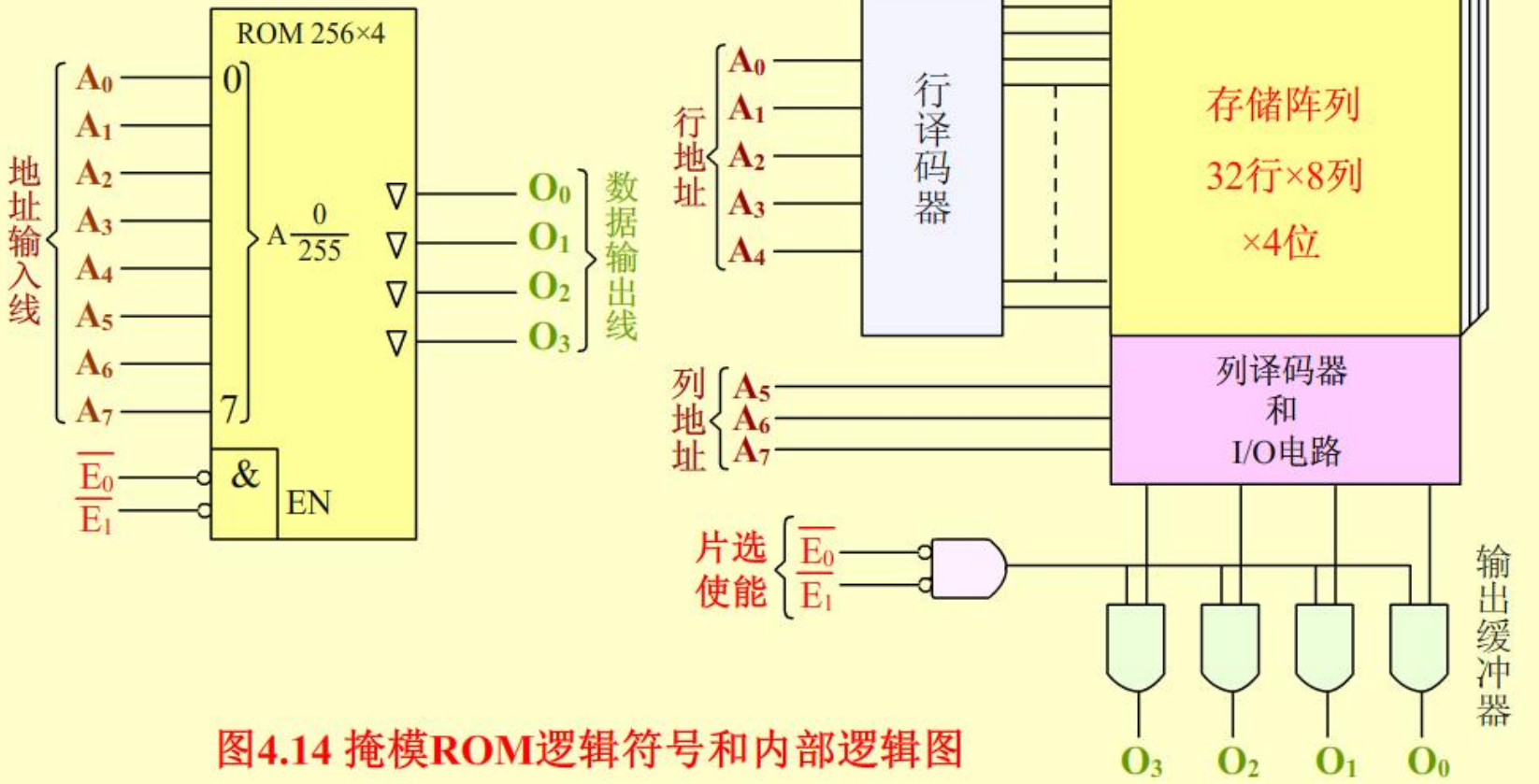
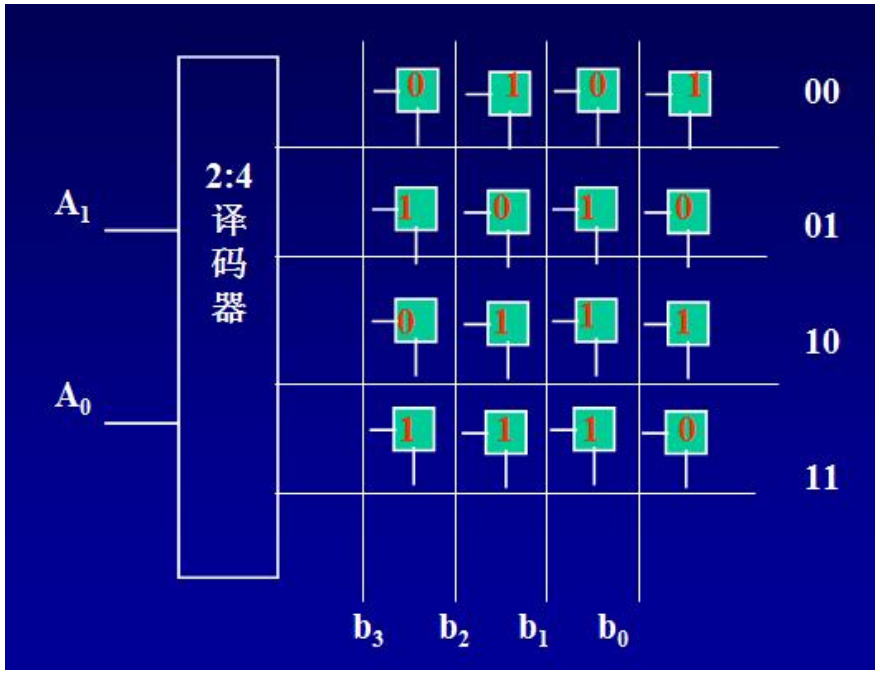


图4.14 掩模ROM逻辑符号和内部逻辑图

2 ROM的逻辑构成

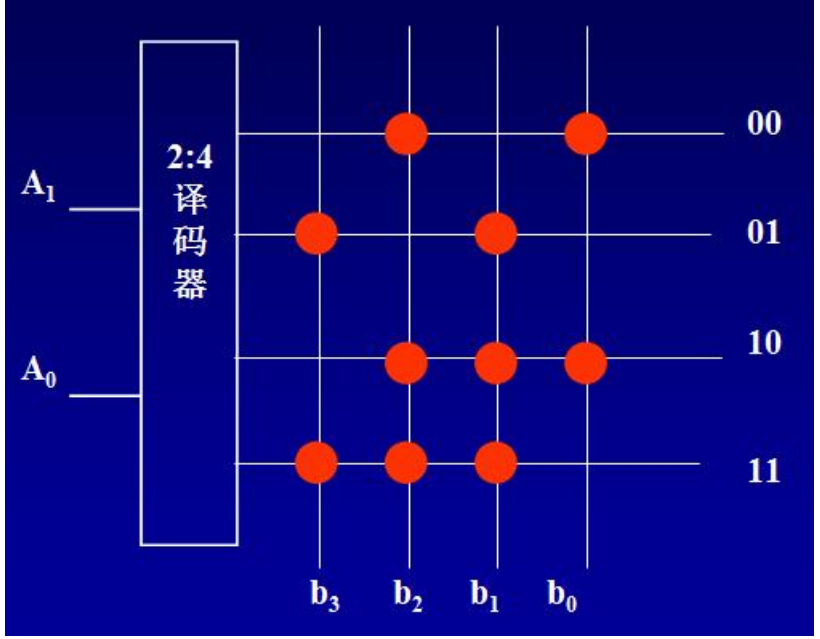
ROM ＝ 与门阵列（地址译码器）＋或门阵列（存储矩阵）构成的组合逻辑电路。

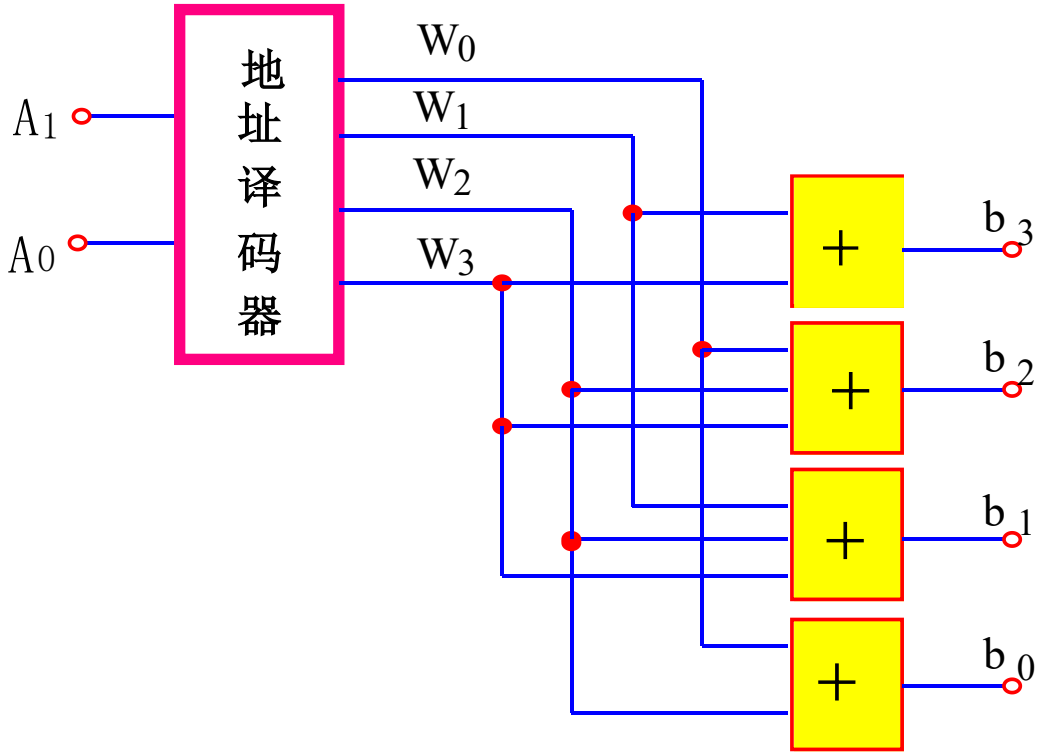


ROM真值表

地　址		存 储 内 容			
A_1	A_0	b_3	b_2	b_1	b_0
0	0	0	1	0	1
0	1	1	0	1	0
1	0	0	1	1	1
1	1	1	1	1	0

$$b_3 = \overline{A_1}A_0 + A_1A_0$$

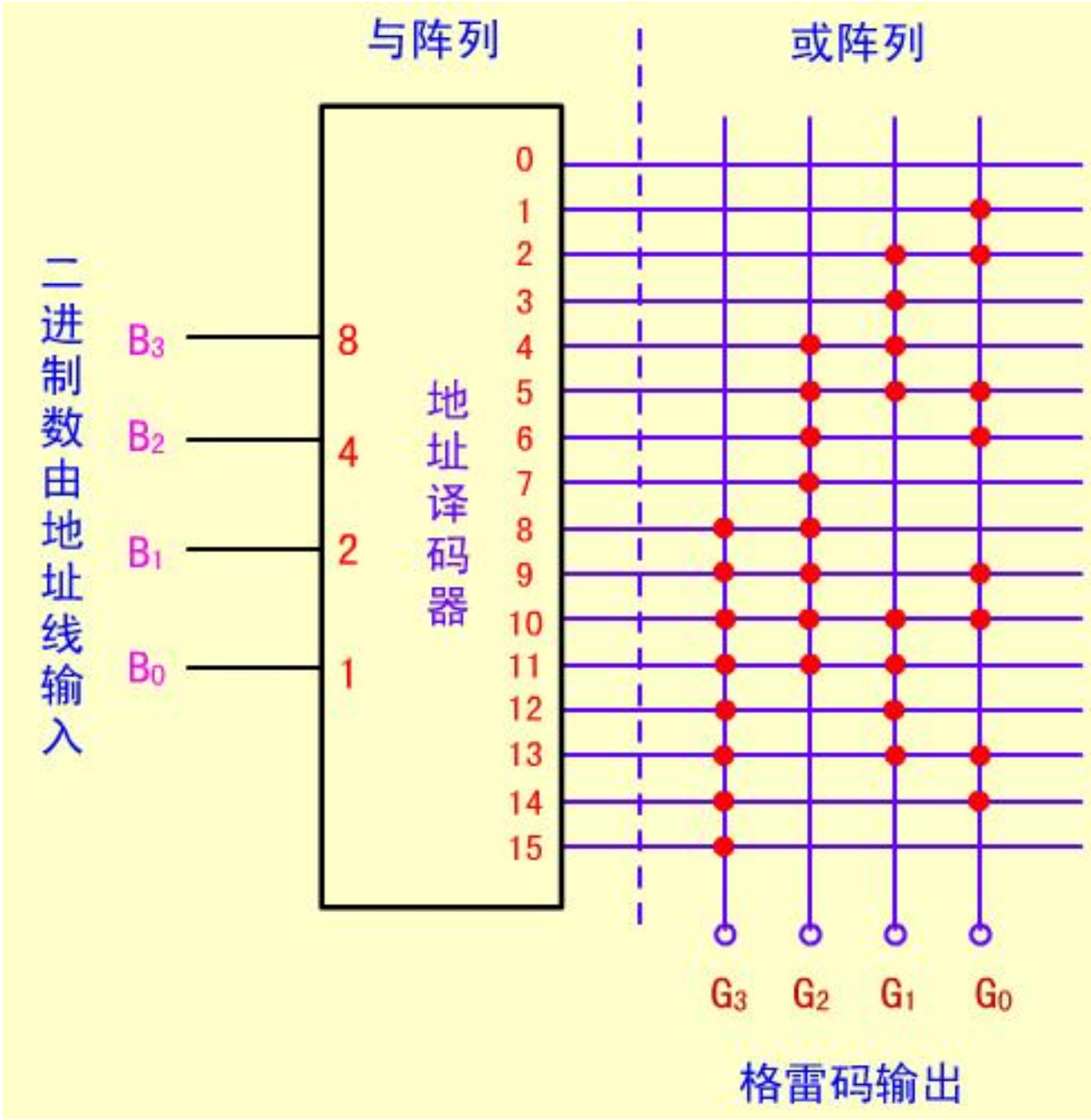




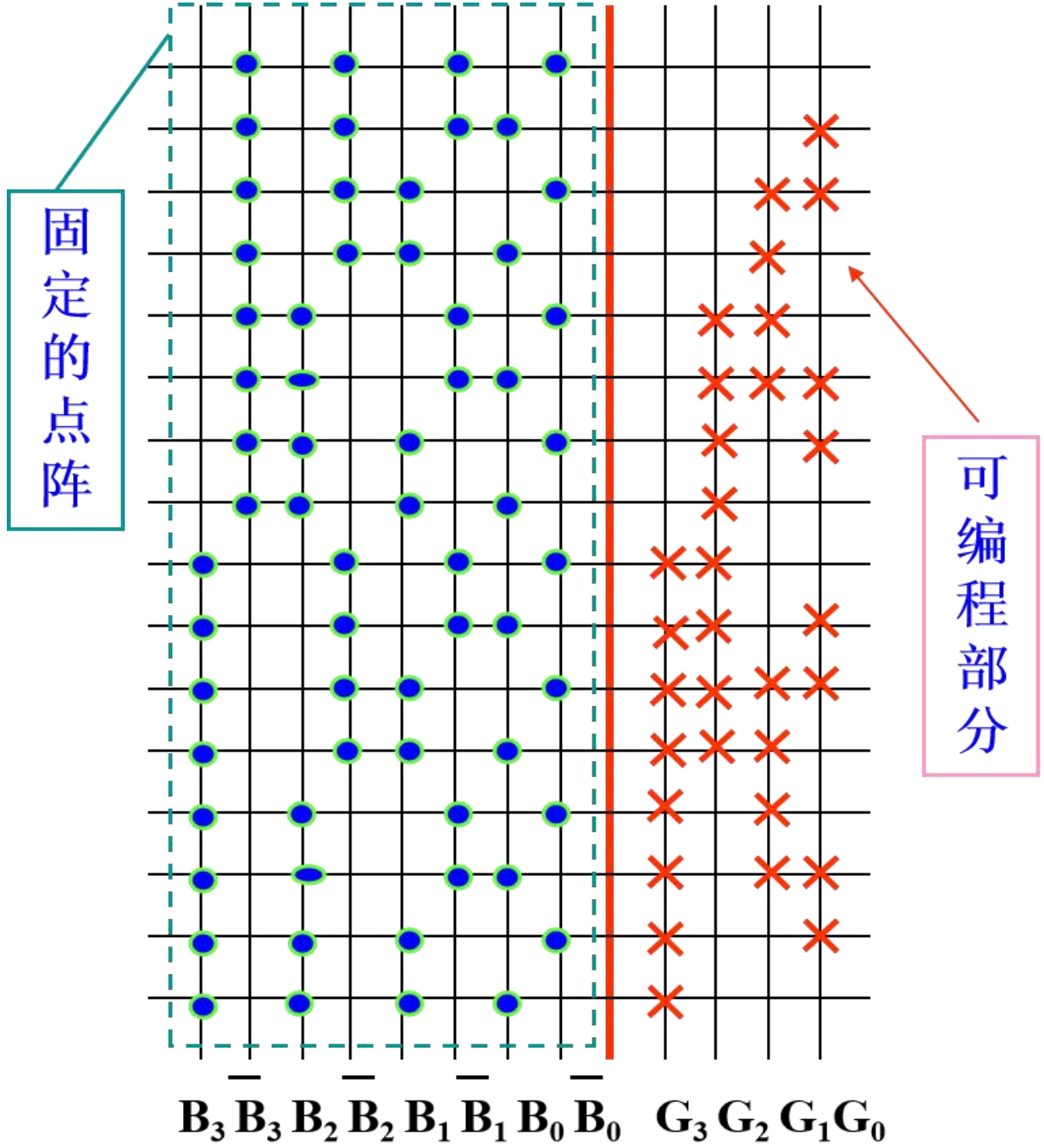
ROM真值表

地　址		存 储 内 容			
A_1	A_0	b_3	b_2	b_1	b_0
0	0	0	1	0	1
0	1	1	0	1	0
1	0	0	1	1	1
1	1	1	1	1	0

二进制码				格雷码			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



二进制码				格雷码			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



思考题：

1、32K × 16表示什么意思？
(32K=2^{15},15根地址线,16根数据线)

2、构成 4M× 32存储器需要16K × 8的芯片多少片？
（1024）

3 ：奔腾CPU的数据总线宽度为64位，地址总线宽度32位，问：奔腾主存的最大物理地址空间为多少？

（ $2^{32}\times64=4\times2^{30}\times64=4\times2^{30}\times8\times8=32GB$ ）

已知存储器的地址空间:

ROM 地址为0000H~3FFFH

PROM 地址为4000H~7FFFH

EPROM 地址为C000H~DFFFH

RAM 地址为E000H~FFFFH

用2:4译码器、适当的门设计地址译码电路。

A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
0000	0000	0000	0000	ROM
0011	1111	1111	1111	
0100	0000	0000	0000	PROM
0111	1111	1111	1111	
1100	0000	0000	0000	EPROM
1101	1111	1111	1111	
1110	0000	0000	0000	RAM
1111	1111	1111	1111	

A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
0000	0000	0000	0000	ROM
0011	FFFF	FFFF	FFFF	
0100	0000	0000	0000	PROM
0111	FFFF	FFFF	FFFF	
1100	0000	0000	0000	EPROM
1101	FFFF	FFFF	FFFF	
1110	0000	0000	0000	RAM
1111	FFFF	FFFF	FFFF