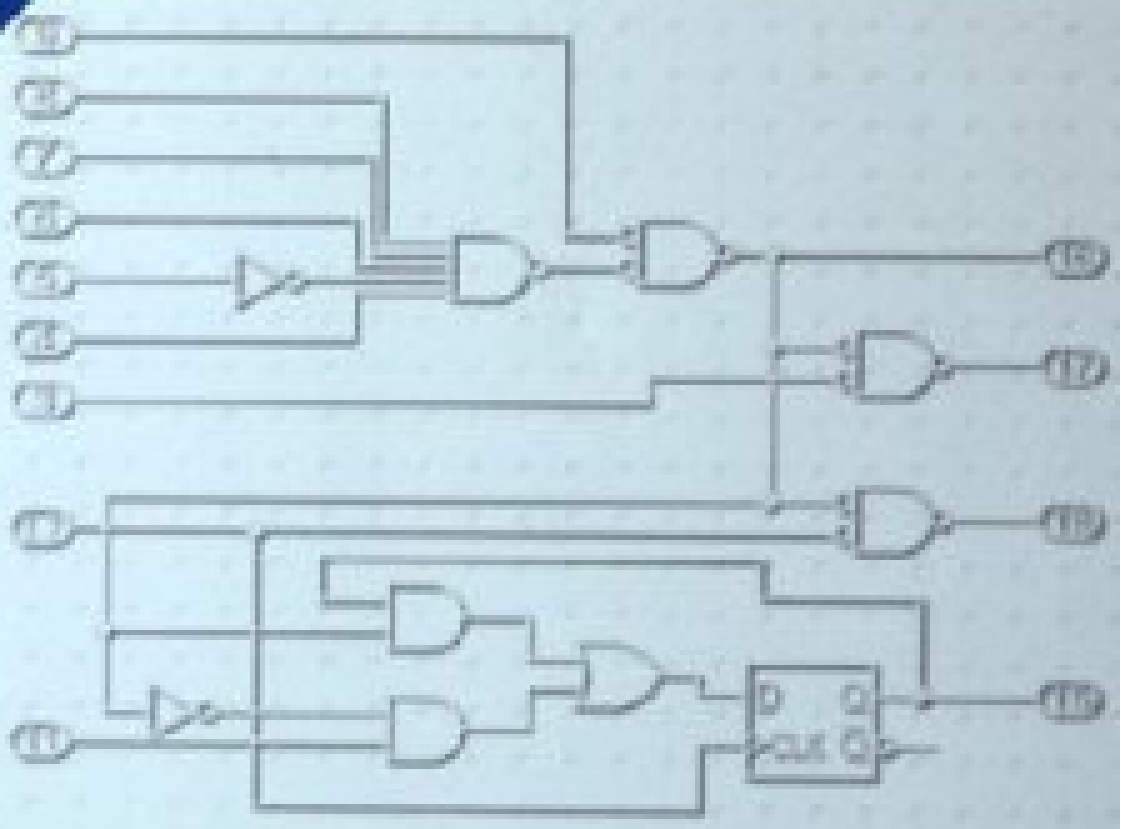
图解实用电子技术丛书

存储器IC的应用技巧

UV-EPROM/EEPROM/SRAM/DRAM的结构与使用方法

[日] 桑野雅彦 著
王庆 译
钱滅 刘滔滔 校



科学出版社
www.sciencep.com

目录

6.6.2 直接总线式 DRAM 的信号连接

6.6.3 直接总线式 DRAM 的操作概况

6.6.4 直接总线式 DRAM 的操作示例

存储器 IC 的应用技巧

第一章

UV－EPROM 的结构与使用方法

1.1　UV－EPROM 的结构与特征

●1.1.1　UV－EPROM 的单元结构

UV-EPROM的单元结构如图所示。其基本结构与在下一章中说明的闪速存储器相同。UV-EPROM的存储单元是由MOSFET（金属氧化物半导体场效应晶体管）构成的，在它的控制栅和N沟道间有一个称为浮置栅的特殊栅极，这是UV-EPROM单元结构的主要特征。

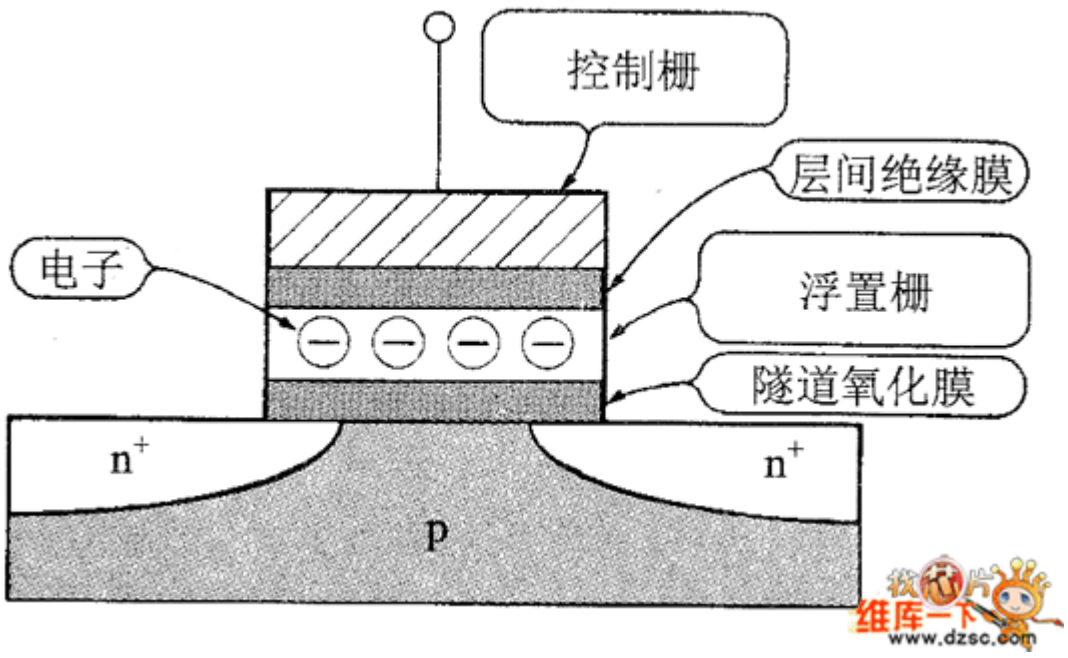


图 UV-EPROM 的单元结构

由于浮置栅利用氧化膜使栅极与基板绝缘，使存储于此处的电荷不能被轻易释放，从而达到持续保存记忆的目的。与闪速存储器相同，通过浮置栅中是否存储电荷，利用 FET（场效应晶体管）的阈

值电压的变化，进行高电平与低电平的判断。一般地讲，UV 一 EPROM 在擦除状态（浮置栅中未存储电荷的状态）时，读出“高电平”；而在存储电荷状态时，读出“低电平”。

●1.1.2　UV－EPROM 的写入与擦除

写入时，通过给栅极加上高电压 vPP，如图 1 所示，向浮置栅注入电荷。注人后的电荷由于不具备穿透硅氧化膜能壁的能量·因而只能维持现状。

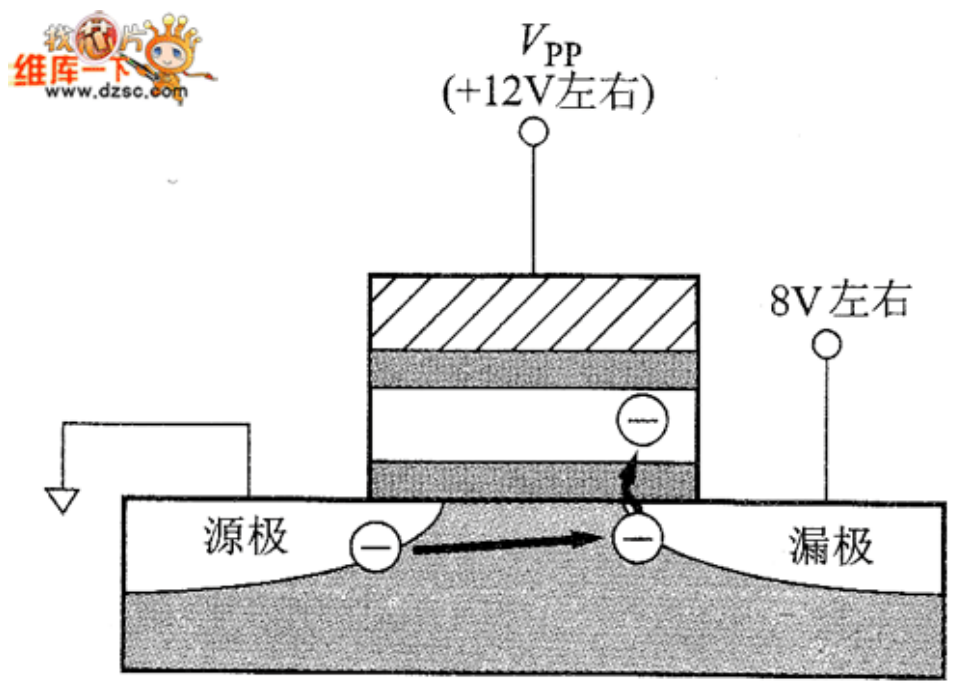


图 1　UV－EPROM 的写入

当浮置栅接收到紫外线的照射，浮置栅中的电子接收了紫外线光量子的能量，则电子变成具有穿透硅氧化膜能壁能量的热电子。如图 2 所示，热电子穿透硅氧化膜，流向基板和栅极，恢复为擦除状态。UV-EPROM 的擦除操作，只能通过接收紫外线的照射来进行，而不能进行电子擦除。也就是说，UV-EPROM 只能够进行由“1”向“0”改变比特数，而在反方向上．除擦除芯片全部内容的方法以外，再没有其他的方法。

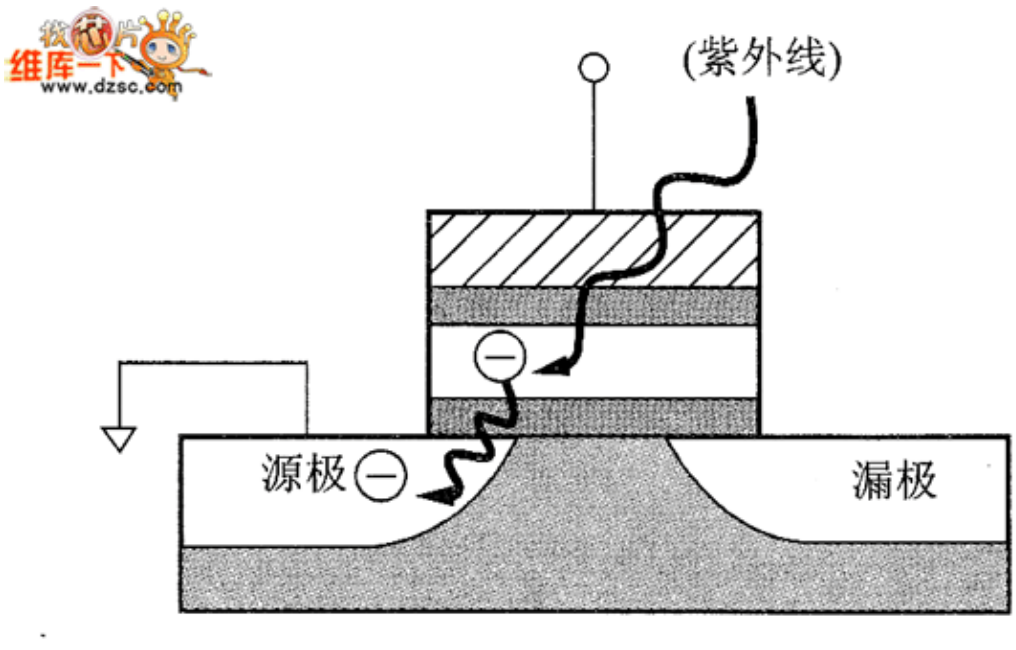


图 2　UV-EPROM 的擦除

我们知道，光的能量与光的波长成反比例关系，为了让电子成为热电子，从而具有穿透氧化膜的能量，就非常需要波长较短的光即紫外线的照射。由于擦除时间决定于光量子的数目，因而即使在波长较短的情况下，也不能缩短擦除时间。一般地，当波长为 400OA（400nm）左右时才开始进行擦除。在3000A左右基本达到饱和。低于 3000A以后，波长即使再短，对于擦除时间也不会产生什么影响。

UV－EPROM 擦除的标准一般为接受波长 2537A、12 000μW / cm2 的紫外线 15～20 分钟左右的照射，即可完成其擦除操作。

根据擦除机制可知，即使得到热能，浮置栅电荷的消失也是发生在某一概率下。其发生的概率是随着器件绝对温度的上升以指数函数递增的。

●1.1.3　一次性 PROM

UV-EPROM为了实现紫外线擦除功能而在封装的中心部位设置了能看到芯片的窗口。如果去掉该窗口的设置，而改为廉价的塑料封装，则不具各擦除（比特由"0"返回"1"）功能，这就是PROM。在将UV EPROM做为产品加以利用时，为防止紫外线照射而擦除数据，一般都在窗口部位贴上遮光片。不同的产品一般写入数据、且擦除后再次利用的情况较少。针对这样的用途，起初就不设置窗口的一次性PROM则具有较大的优势。因为UVEPROM与一次性PROM除封装不同之外，内在机制是完全相同的。所以，ROM写人器的自动识别功能会将这两种类型的器件作为同一类型进行处理。这样，在产品试制期间，可以利用UV-EPROM进行实验。而在制造产品时，则可以顺利地以一次性PROM取代UV-EPROM，这对于产品的生产是非常有利的。

1.2　UV－EPROM 的输入输出信号

作为UV-EPROM的实例，我们以AMD公司 1M位（128K×8 位）UV-EPROM的Am27C010为例进行说明。

Am27C010 的引脚配置如图 1 所示，图 2 是在其内部框图的基础上，根据引脚的功能分组表示的。在图 1 中，NC引脚没有在图 2 中出现，NC是无连接（No-Connection）的缩写，它虽然作为封装的引脚存在，但其内部并不与任何部件相连接。下面我们逐一说明这些引脚所代表的意思。

图 1 Am27C010 的引脚配置

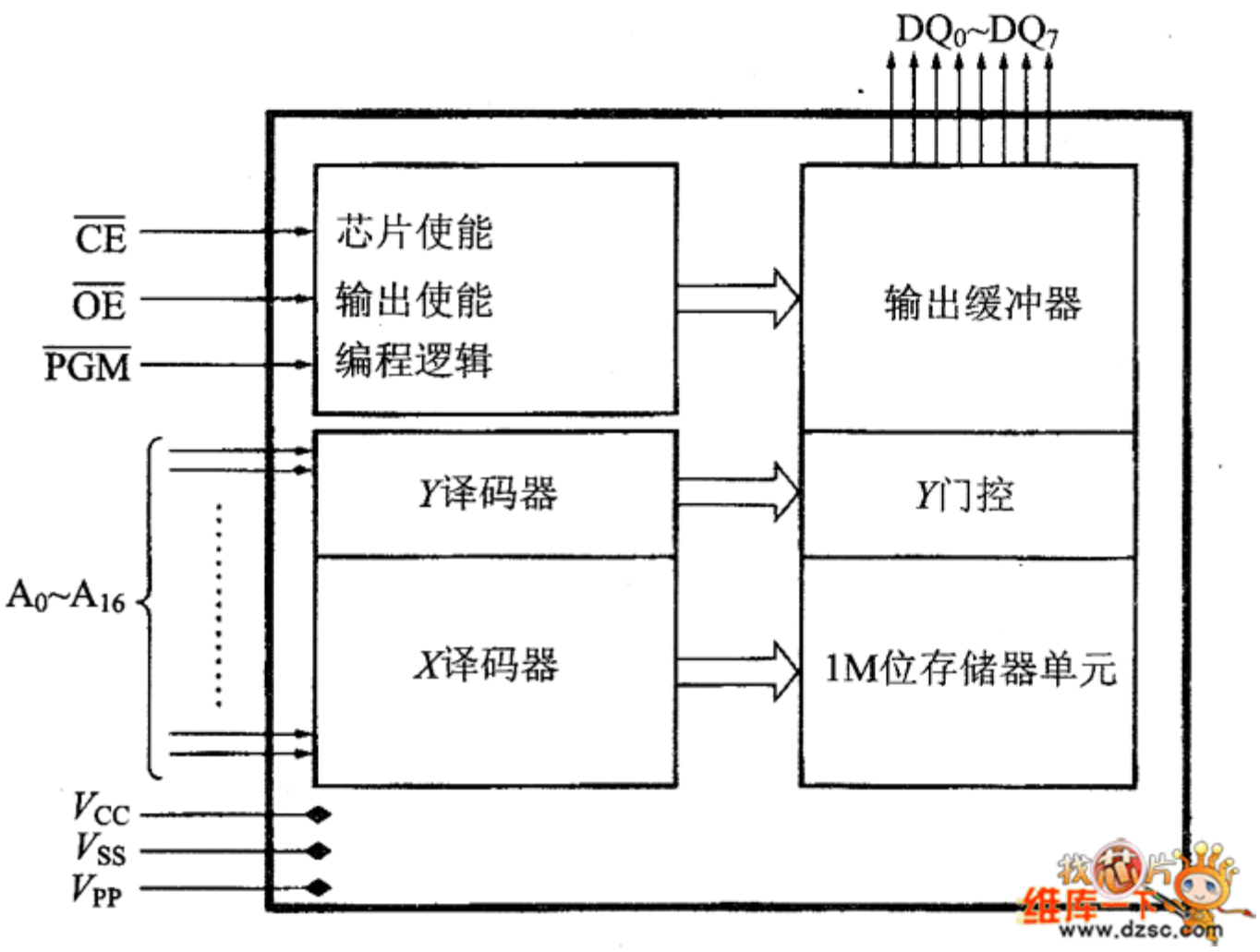


图 2 Am27C010 的内部框图

▲ A0～A16（地址总线）

A0～A16 为地址总线。因为 Am27C010 是 128K×8 位结构的 1M 位的 UV-EPROM，所以地址有 128K、17 根总线。通常 A0 作为 LSB（最低位比特）、A16 作为 MSB（最高位比特）使用。RAM 由于只需要读出与写入时相同的内容，所以即使 A15 与 A16 反向连接也无所谓。而 UV-EPROM

由于写入时使用了 ROM 写入器，因而引脚的顺序几乎就不能替换进行使用（考虑软件解析的方法，有意替换地址来加以利用的例子也存在）。

作为特殊用途的 A9 引脚通过赋予＋12V 的电压，可使其读出制造商名称以及器件的 ID 编码。这应用于 ROM 写人器自动判断插座上所插接的 ROM 类别上，在普通的系统中这样的功能很少用。

▲ DQ0～DQ7（数据总线）

DQ0～DQ7 是数据总线。由于Am27C010 为 128K×8 位的结构，因而数据线也具有 8 位的宽度。与地址总线相同，数据总线虽然为了破解困难也存在有意使用非标准的情况，但通常情况下，将 DQ0 作为LSB，DQ7 作为MSB使用。在一般的操作中，UV－EPROM为ROM，即只读存储器，所以 DQ0～DQ7 为输出专用引脚，只在编程时用于输入。

▲ /OE（输出使能，Output Enable）

/OE是ROM数据输出缓冲器的使能信号，是低电平激活的输人引脚。当与CS一起为有效信号（为低电平）时，向地址总线（A0～A16）所指定的地址中写人的数据将出现于数据总线（DQ0～DQ7）上。另外，本书在表示低电平激活的情况下，信号名上一般标有上划线。但根据产品不同，有时也在前面或者后面加上斜杠（/），或者类似OE#那样在后面加上#的记号。

▲ /CE（芯片使能，Chip Enable）

根据产品的不同，该引脚有时也为CS（芯片选择，简称片选，Chip Select），这是将器件设置为选择状态的信号，通常与/OE共同应用于数据读出中。

▲ /PGM（编程使能）

应用于编程（写入）中，在Vpp端口加上编程电压（＋12V）的状态时是有效信号。除此之外，在一般的操作中该引脚没有任何意义。

▲ Vcc（电源输入）

这是用于UV-EPROM工作的电源。一般情况下，装入系统进行操作时，在此赋予＋5V的电压。编程工作时，Vcc也需要赋予电压。

▲ VPP（编程电源输入）

这是编程时提供编程电压（＋12V左右）电源的引脚，利用该电压向浮置栅注人电荷。在最近的闪速存储器等器件中，大多在芯片内部都具各升压电路，由Vcc所提供的电压生成擦除/编程电压。但是UV-EPROM由于不需要进行在板写人，所以没有内置升压电路，而是在编程时由ROM写人器提供电源。

Am27C010 在进行普通的读操作时，虽然忽略了VPP，但根据UV-EPROM的特性，有时也需要给VPP提供与Vcc相同的电压。这样的数据最好参看关于UV－EPROM的数据手册。

▲ GND（地线）

这是器件确定电压基准的引脚，所有输入输出信号电压的规定都是以该引脚为标准的。

1.3 操作模式

Am27C010 的操作是由 CE、OE、PGM 等输入信号的状态决定的。操作模式与各种引脚状态的关系如表所示。系统使用的一般为数据读、输出禁止、待机三种模式。

操作模式	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	A_0	$A_1 \sim A_8$	A_9	$A_{10} \sim A_{16}$	V_{PP}	$DQ_0 \sim DQ_7$
数据读	“L”	“L”	“X”	“X”	“X”	“X”	“X”	“X”	数据输出
输出禁止	“L”	“H”	“X”	“X”	“X”	“X”	“X”	“X”	高阻抗
待机(TTL)	“H”	“X”	“X”	“X”	“X”	“X”	“X”	“X”	高阻抗
待机(CMOS)	$V_{CC} \pm 0.3V$	“X”	“X”	“X”	“X”	“X”	“X”	“X”	高阻抗
编　程	“L”	“H”	“L”	“X”	“X”	“X”	“X”	V_{PP}	数据输入
编程验证	“L”	“L”	“H”	“X”	“X”	“X”	“X”	V_{PP}	数据输出
编程禁止	“H”	“X”	“X”	“X”	“X”	“X”	“X”	V_{PP}	高阻抗
自动选择	“L”	“L”	“X”	“L”	“L”	V_H	“L”	“X”	制造商代码 (Am27C010 为 01h)
	“L”	“L”	“X”	“H”	“L”	V_H	“L”	“X”	器件代码 (Am27C010 为 0Eh)

表 Am27C010 的操作模式

*: VPP = 12.75V ± 0.25V, VH = 12.0V ± 0.5V, “X”: 无论何信号都无关紧要

在表中，“H”代表 VIH（＋2.0V 以上），“L”代表 VIL（＋0.8V 以下），VIH 与 VIL 分别决定了其电压的上限和下限，Am27C010 的 VIH 上限为 Vcc＋0.5V，VIL 下限为－0.5V（以 GND 引脚为基准）。如果端口加上超过该范围的电压，则器件有可能会毁坏。

●1.3.1　数据读（Data Read）

这是读取ROM内容的操作。一旦/CE有效（低电平），则芯片处于使能（选择）状态，通过使/OE有效，向外部输出ROM内容的输出缓冲器将处于使能状态。利用 17 根地址总线（A0～A16），在128KB中指定所希望读出的那个比特位。

如果在这种状态下等待数据的读出，则DQ0～DQ7 中将出现数据，外部电路的CPU将提取这些数据进行操作。因为并没有利用/PGM及VPP端口，所以无论是高电平还是低电平都无关紧要。关于读操作的细节我们将在后面进行叙述。

●1.3.2　输出禁止（Output Disable）

由于/CE有效，所以芯片本身处于使能状态，但是根据/OE，输出缓冲器将处于禁止状态。因为只要稳定地址总线，使/CE有效，就可以针对ROM内部的存储器单元进行访问，所以在输出禁止模式下，首先进行内部操作，然后使/OE有效，这样一旦进人数据读模式，就可以缩短表面上的存取时间。

采用下面这种连接方法的情况较多，即当 ROM 连接在比读控制信号先确定的地址上时，地址的低位与 ROM 连接，对高位进行解码生成/CE 信号，然后将读信号传人/OE。在这种情况下的 ROM 操作，实际上是由输出禁止模式向数据读模式的转移。

●1.3.3　待机（TTL/CMOS）

/CE一旦变为高电平，则UV-EPROM处于非选择状态。此时，根据/CE的电压，损耗电流将逐渐改变。当/CE是一般的高电平（＋2.0V以上）时，处于TTL待机状态。尽管如此，但如果CE进而增高到Vcc±0.3V，则UV－EPROM处于CMOS待机状态，损耗电流就会变得更小。Am27C010的TTL待机电流最大为 1.0mA，而其CMOS待机时的电流为 100μA，减小了一个数量级。

随着CMOS器件各种形式组合的逐渐增多，有时侯会在无意识的情况下使器件处于CMOS待机状态。最近，器件的电源电压一般都被降低到3.3V以下。如果这种器件的输出与Am27C010 连接，则可以在TTL待机状态下进行操作。

●1.3.4　编程（Programming）

这是写入操作，但并不只是单纯处于这种状态即可完成操作。由于规定了额定电压以及额定时间，因此编程操作是相当麻烦的。在闪速存储器中，由于时序控制是在器件内部的电路中自动进行的，所以主机方面只要发布指令就可以了。而UV-EPROM的时序控制等所有操作都必须来自外部电路。

●1.3.5　编程验证（Program Verify）

这是用于编程时检查写入是否正确的读出模式。在类型较旧的 EPROM 中，是通过单一的长脉冲进行写人操作的，但由于操作时间过长，所以现在的 EPROM 都采用新的方法，即给予短脉冲进行编程，读出的数据如果与写人的数据一致，则进行下一地址的数据写入。

编程中该数据读出的操作称为编程验证。使/PGM 无效（高电平），只要/OE 有效，就可以进行编程验证操作。

●1.3.6　编程禁止（Program Inhibit）

即使处于加载了V_{PP}电压的状态下，一旦/CE为高电平，EPROM也将成为非选择状态，这样的状态是编程禁止模式。使之处于该状态，譬如增加V_{PP}电压，这时即使/PGM有效，也不可能误写人数据c该编程禁止模式应用于向编程操作移动之前以及退出编程操作的时候。

●1.3.7　自动选择（Auto Select）

这里并不是指EPROM被自动选择，而是指当ROM写入器判断器件的制造商及型号时，自动选择ROM的容量及编程算法。不添加V_{PP}电压，在给A_9引脚加上＋12V电压的状态下，如果/CE和/OE有效，则可以读出制造商代码（Manufacture Code）和器件代码（Device Code）。由A_0选择读出哪个编码。

1.4　DC 规定

电源电压以及输人输出电压等是由DC Characteristic（直流特性）规定的。Am27C010的DC规定如表所示。该表中经常出现的V_{OH}等表示方法基本上是这样规定的：开头第 1 个字母V表示额定电压，I表示额定电流；第 2 个字母的O表示输出，I表示输人，L表示漏电流或者负载电流；第 3 个字母的H表示高电平时的规定，L表示低电平时的规定。

符号	意　思	测定条件	min	max	单位
V_{OH}	“H”高电平输出电压	$I_{OH}=-400\mu A$	2.4		V
V_{OL}	“L”低电平输出电压	$I_{OL}=2.1mA$		0.45	V
V_{IH}	“H”高电平输入电压		2.0	$V_{CC}+0.5$	V
V_{IL}	“L”低电平输入电压		−0.5	+0.8	V
I_{LI}	输入负载电流	$V_{IN}=0V\sim V_{CC}$		1.0	μA
I_{LO}	输出漏电流	$V_{OUT}=0V\sim V_{CC}$		5.0	μA
I_{CC1}	工作电流	$\overline{CE}=V_{IL}$，$f=10MHz$，$I_{OUT}=0mA$		30	mA
				60	
I_{CC2}	TTL 待机电流	$\overline{CE}=V_{IH}$		1.0	mA
I_{CC3}	CMOS 待机电流	$\overline{CE}=V_{CC}\pm0.3V$		100	μA
I_{PP1}	V_{PP}提供电源(读操作时)	$\overline{CE}=\overline{OE}=V_{IL}$，$V_{PP}=V_{CC}$		100	μA

表　Am27C010 的 DC 规定

▲　V_{OH}/V_{OL}

这是对输出电压的规定。EPROM数据输出缓冲器的输出电压如果是高电平，则在接近电源电压(VCC)之前无条件输出；如果是低电平，则输出电压为零则是其理想输出。但实际上，器件内部由于存在着某种电阻，输出端电流越大，高电平时的电压输出越低；相反，低电平时的电压输出越高。

测定条件中所表示的电流通过正负号表示电流的方向：负号表示电流由EPROM流向外部的方向，正号表示外部流入EPROM的方向。

▲　V_{IH}/V_{IL}

这是对输人端的电压规定。规定输人多少伏以上的电压必须判断为高电平，多少伏以下的电压必须判断为低电平。当输入电压值为V_{IL}和V_{IH}之间的电压时，由于各种元器件所存在的误差，因而不能够判

断为何种电平。Am27C010 的V_{IH}和V_{IL}是俗称“TTL电平”的一般值，V_{IH}为2.0V以上的值，V_{IL}为 0.8V以下的值。

▲　I_{LI} / I_{LO}

I_{LI}是对输人负载电流的规定。由于普通数字IC的基本操作是采用电压的“H（高电平）”、“L（低电平）”的二进制方法，所以理想的情况是如果信号的状态稳定，电流将为零。但现实情况并不是这样的，而是有某种电流流向输入端，该电流就是I_{LI}，如图（a）所示。

另一方面，I_{LO}是输出漏电流。输出端为高阻抗，也就是输出开关处于断开状态，理想的情况是无论输出与GND还是与V_{CC}连接都完全没有电流。然而，因为开关元器件的阻抗不是无限大，所以实际上还是有某种电流存在。这就是要规定I_{LO}的原因，如图（b）所示。

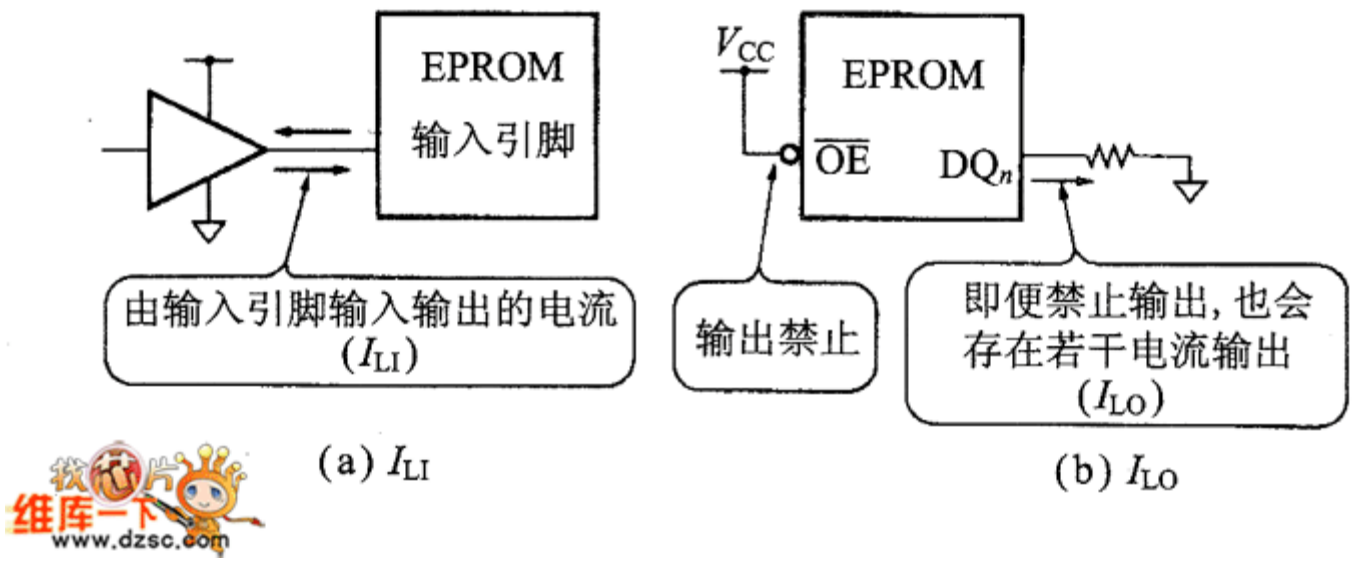


图　I_{LI}与I_{LO}

即使Am27C010 的I_{LI}和I_{LO}值在最大的时候，也不过是 1μA和 5μA这样小的数值，所以，只要不是大量的器件并联，基本上是不会出现问题的。

▲　I_{CC1}

这是一般操作时的损耗电流。因为数字IC特别是CMOS结构的器件根据操作频率，电流会有较大的变化，所以在此测定条件中标明了操作频率。另外，当输出缓冲器连接有负荷时，电流的流向为V_{CC}端→I/O缓冲器→负载，所以感觉上损耗电流变大。

为了避免这种影响，在I_{CC1}的测定条件中，将CE置于低电平，使器件处于使能状态；而将OE置为高电平，使输出缓冲器处于禁止状态。在实际的使用条件下，曲于I_{CC1}值中加上了由缓冲器流向负载的电流，因而需要注意。

另外，由于对于低损耗电流的需求比较多，因此在很多的存储器IC中进行适应I_{CC1}值的分类。Am27C010 也不例外，最大损耗电流为 30mA的与最大损耗电流为 60mA的两种元器件被整合。与其说是制作了不同的产品，不如说是同一产品根据实际测定值或者根据生产计划而改变了用途。

▲　I_{CC2}，I_{CC3}

这是对待机电流的规定。CE的电压如果超过V_{IH}，则EPROM变为禁止，处于损耗电流较小的待机状态。但GE电压如果增加到V_{CC}±0.3V左右，则损耗电流将被控制得更小。

Am27C010 通常待机状态（TTL待机）时的电流为 1mA。如果CE的电压处于更高的电压状态（CMOS待机状态），则此时的待机电流为 100μA，降低了十分之一。

▲　I_{PP1}

这是编程时流向VPP端的电流值。V_{PP}电压虽然较高，但由于编程时存在存储于浮置栅的电流，因此与 100μA相比，该电流值相当小。

1.5　UV-EPROM 的读操作

接下来我们看一下UV-EPROM的读操作。无论怎么说，这只是作为ROM（Read Only Memory，只读存储器）的操作，因而非常简单。

将地址总线（A_0～A_{16}）设置为希望访问的地址，当/CE＝/OE＝低电平时，DQ中出现数据。

现在，让我们来讨论AC特性。

AC特性对时序进行了规定。图显示了Am27C010 读操作时的波形。具体的时序如表所示。在“速度等级”一栏中虽然有许多内容，但这表示即使是相同的器件，也将根据t_{ACC}时间进行区分。器件中标示速度等级的“—45”及“－90”等的数值是与器件型号及批号等一同被打印上去的。

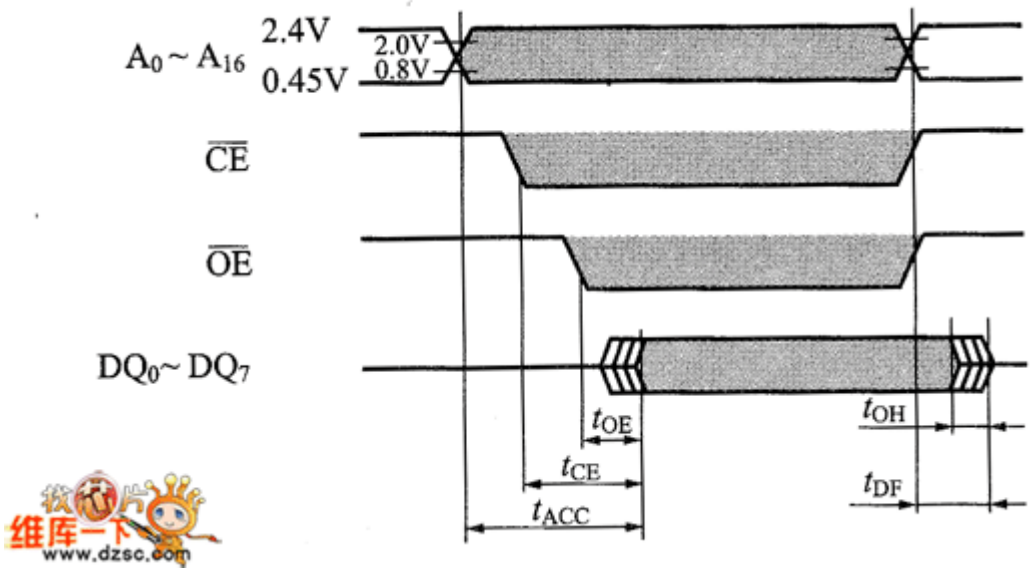


图 Am27C010读操作的波形

符号		内容	测定条件	min/max	Am27C010速度等级								单位
JEDEC式表示	一般表示				-45	-55	-70	-90	-120	-150	-200	-255	
t_{AVQV}	t_{ACC}	从地址确定到数据输出为止	$\overline{CE}$, $\overline{OE}=V_{IL}$	max	45	55	70	90	120	150	200	255	ns
t_{ELQV}	t_{CE}	从$\overline{CE}$有效到数据输出为止	$\overline{OE}=V_{IL}$	max	45	55	70	90	120	150	200	255	ns
t_{GLQV}	t_{OE}	从$\overline{OE}$有效到数据输出	$\overline{CE}=V_{IL}$	max	25	35	35	40	50	65	75	75	ns
t_{EHQZ}/ t_{GHQZ}	t_{DF}	从$\overline{CE}$/$\overline{OE}$无效到数据输出高阻抗		max	25	25	25	25	35	35	40	40	ns
t_{AXQX}	t_{OH}	地址变化，$\overline{CE}$、$\overline{OE}$无效后的数据保持时间		min	0	0	0	0	0	0	0	0	ns

表 Am27C010 的 AC 时序

这并不是根据速度等级的不同而进行不同的设计，即使是以完全相同的设计进行制造，也将根据试验结果和出货计划进行等级分类。例如，“－ 90”器件的t_{ACC}最大是 90ns，它不能归类到“－70”的器件中。也就是说，t_{ACC}的实际测量值如果不是大于 70ns、低于 90ns之间的值，那么根据情况，可能被归类到“－45”及“—70”等的等级中。

下面我们针对图示的时序做少许补充。

▲ t_{ACC}： 地址访问时间（Address Access Time）

在维持/CE＝/DE＝“L”（低电平）的状态下，如果改变地址总线的状态，则一定时间后该地址的

数据将显现于DQ端。从地址确定后到数据被实际确定之间的时间就是t_{ACC}。在t_{ACC}时间形成之前，不能确保DQ端出现数据。

▲ t_{CE}： /CE访问时间

在地址确定、/OE仍然维持“L”低电平的状态下，如果/CE有效（为“L”），则一定时间之后将出现指定地址的数据。从/CE有效到确保数据已确定这一段时间是t_{CE}。Am27C010 的t_{ACC}和t_{CE}的时间是相同的。

▲ t_{OE}

在地址确定、/CE维持“L”低电平的状态下，从/OE有效到数据被确定的时间就是t_{OE}。观察存储器内部，一旦地址确定、/CE有效,则向存储器单元的访问就全部完成，数据输出到EPROM的输出缓冲器前。

在此，只要/OE有效，输出缓冲器就输出数据。为此，比较t_{ACC}和t_{CE}，t_{OE}会更短。

t_{ACC}、t_{CE}和t_{OE}无论哪个都不能单独确定大小，必须在整体中去适应速度最慢的那个时间值。例如，在利用了Am27C010 Qn的系统中，地址确定后，在 5ns后/CE有效，再 5ns后/OE有效。从表中可看出，t_{ACC}＝t_{CE}＝90ns，t_{OE}＝40ns。

如果以地址被确定那一时刻作为起点，则：

根据t_{ACC}的访问时间：90ns

根据t_{CE}的访问时间：5ns＋90ns＝95ns

根据t_{OE}的访问时间：5ns＋5ns＋40ns＝50ns

因而，访问时间的大小取决于t_{ACC}、t_{CE}和t_{OE}中最长的t_{CE}，也就是说地址被指定后数据将在 95ns后被确定。

▲ t_{DF}

/CE及/OE如果无效，则DQ引脚处于高阻抗状态。但这也不能说是瞬间，还是需要一些时间的，该

时间就是t_{DF}。在t_{DF}以内的时间里，如果其他的器件驱动了数据总线，则将会与EPROM的输出发生冲突。因此，在进行硬件设计时必须注意这一点。

▲ t_{OH}

提供给EPROM的地址发生变化，或者即使/CE和/OE无效，瞬间数据也不会消失，而是在非常短的时间内，维持输出的原始状态。规定的这个最小时间就是t_{OH}。

Am27C010 的该时间全部是零，所以不能确保地址发生变化或者/OE及/CE无效后的数据。

1.6 UV－EPROM 的编程方法

●1.6.1 UV－EPROM 写入方式的变迁

UV-EPROM 编程的关键在于向浮置栅中注人的电荷，与读操作相比需要非常长的时间，为此一直在进行着写人方式的改善。

从很早期到 64K 位的 EPROM 时期，经常使用的是 50ms 的固定脉冲方式，这是每写人一个地址（1 字节）就赋予 50ms 的写脉冲的方式。即使在 64K（8K×8）位的情况下也需 8K×50ms，即 410s 的时间。如此简单的处理就需要 5 分钟以上的时间，显然对于稍微复杂的处理根本不具有实用性。

之后，高速的写人方式出现。这种写入方式是在赋予 1ms 左右的较短的写脉冲的同时，确认数据是否被写入，然后在确认已写人的时刻向下一个地址移动。这种写人方式出现后，写入时间被一下缩短到十分之一的程度。接着，在 1M 位的产品出现之前，还出现了更快的页写入方式等，将脉冲缩短到了 0.2ms 的程度，写脉冲宽度一直减小到使用至今的 50μs，是最初 50ms 脉冲的千分之一。

据此，与昔日的 64K 位的 EPROM 相比，目前使用的 4M 位的 EPROM 写人时间实在是缩短了许多。

EPROM 写人方式与写人时间粗略的推测如表所示。

表 UV－EPROM 写入方式的变迁

容量(目标)	典型的写入方式	写入时间的目标
～64K	50ms 固定脉冲方式	64K/7min
64K～512K	1ms 快速写入方式	512K/4min
256K～1M	0.2ms 快速写入方式	1M/1min
1M～4M	0.2ms 快速翻页方式	1M/30s
4M～	50μs 快速翻页方式	4M/30s

●1.6.2　AM27C010 的编程方法

我们列举的Am27C010是AMD公司的UV EPROM。AMD公司UV－EPROM的高速写入方式称为“Flashrite Algorithm”。这种方式是在VPP端加上 12.75、Vcc端加上 6.75V的电压，在赋予 100μs脉冲状写入信号的同时，进行写入操作。

图 1 表示了写人算法。在如图所示的流程图中，在验证之前进行了 25 次的判断，这是 AMD 公司正式资料的原型。一般都认为判断是在编程验证之后进行的。在此的资料，记录了 AMD 正式的操作流程。

具体的操作波形如图 2 所示，各种时序规定如表所示。由于电源电压是用于编程的，所以除了出现VCC及VPP电压建立时间以外，波形自身与通常SRAM等的读 / 写操作等没有变化。但是观察表的单位就可以明白，地址建立等所必要的时间相当于μS数量级等，所以必须注意建立以及保持时间。

表 UV－EPROM 的写入时序

符号		内容	时序		
JEDEC 表示	一般表示		min	max	单位
t_{AVEL}	t_{AS}	地址建立时间	2		μs
t_{DZGL}	t_{OES}	$\overline{OE}$建立时间	2		μs
t_{DVEL}	t_{DS}	数据建立时间	2		μs
t_{GHAX}	t_{AH}	地址保持时间	0		μs
t_{EHDX}	t_{DH}	数据保持时间	2		μs
t_{GHQZ}	t_{DFP}	从输出使能到输出高阻态	0	130	ns
t_{VPS}	t_{VPS}	V_{PP}建立时间	2		μs
t_{ELEH1}	t_{PW}	$\overline{PGM}$编程脉冲宽度	95	105	μs
t_{VCS}	t_{VCS}	V_{CC}建立时间	2		μs
t_{ELPL}	t_{CES}	$\overline{CE}$建立时间	2		μs
t_{GLQV}	t_{OE}	由$\overline{OE}$的数据制定时间		150	ns

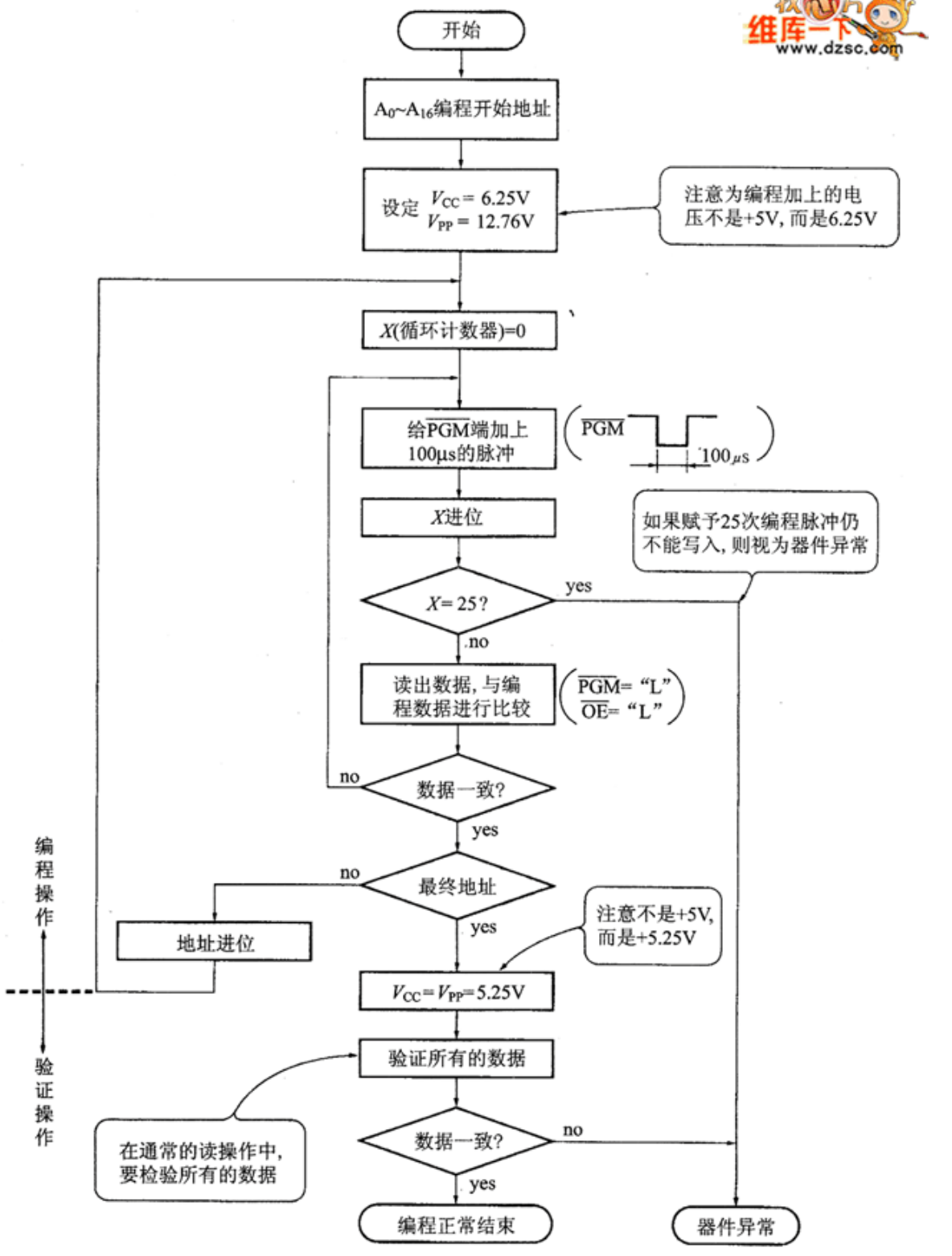


图 1　Am27C010的编程流程

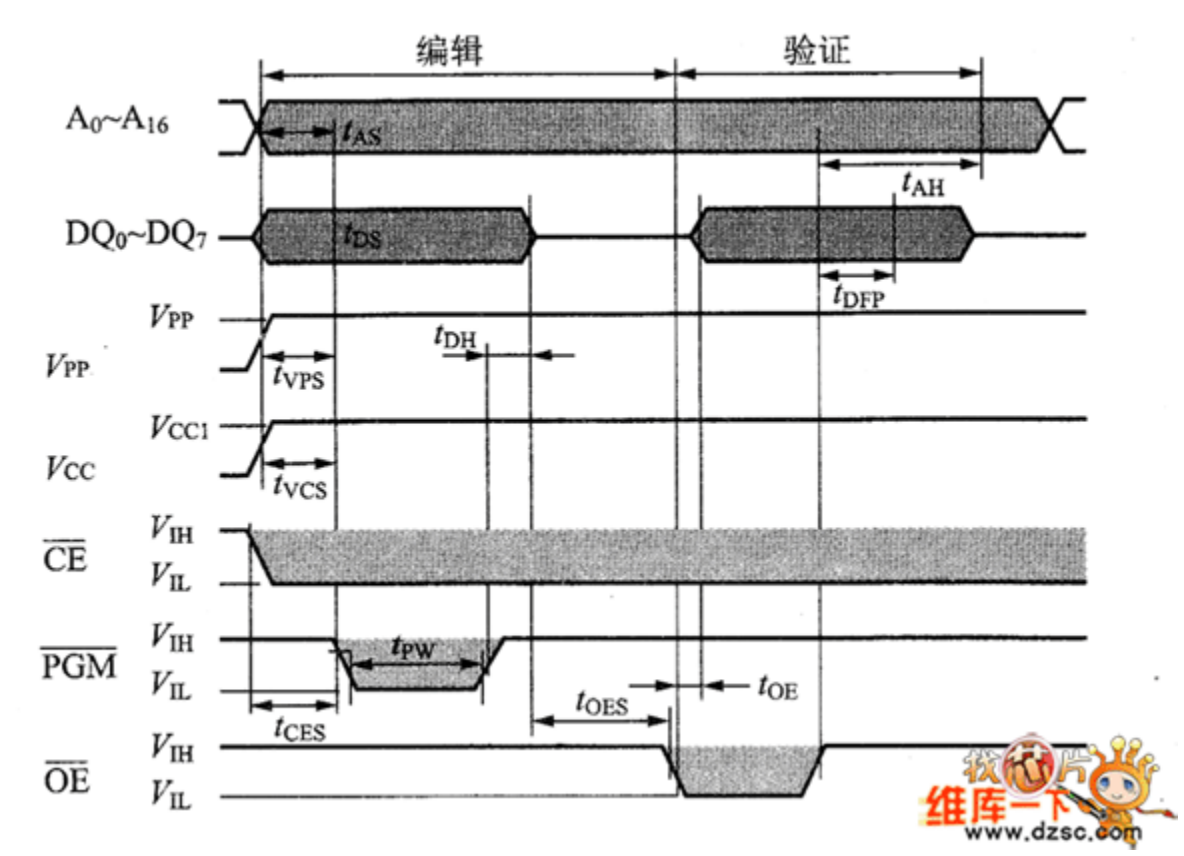


图 2　编程波形

▲　编程电压的添加

为 EPROM 提供编程电压。请注意：不但只给 VPP 提供 12．75V 的电压，还必须为 VCC 提供高于平时操作时电压的 6．25V 电压。在普通的写入装置中一般都搭载稳压器，但也需要考虑在写入过程中对于负载变化的响应。

▲　写入开始地址／数据配置

进行写人操作的地址和写人的数据需要分别配置给各地址总线和数据总线。UV-EPROM 可以针对任意地址进行写人操作。

地址、数据的信号线电平为普通的 TTL 电平。

▲　添加写脉冲

将 PGM 端设置为只有 100μs 的低电平。据此，向写人了“0”的 EPROM 内部存储器单元的浮置栅中注人电荷。

▲　编程验证

如果为/PGM 端提供写脉冲，则将在浮置栅处注入某种电荷。但并不能确认已经存储了使存储器单元确实只为“0”状态的足够的电荷。为此需要进行编程验证。编程验证中其他端口的状态以及电压保持不变，通过使/OE 有效来进行验证。

如果此时读出的数据一致，则表示写人操作完成，如果需要向下一个地址写入，则返回到步骤 2。

如果读出的数据不一致，则表明在先前的脉冲中电荷并没有被完全注人，因此返回步骤 3。但对同一地址的脉冲添加最多只能进行 25 次，如果提供了 25 次脉冲仍没有顺利写入，则表明该器件存在异常，以出错结束操作。

▲　读验证

在希望写入的范围中所有数据被正常写入后，将进行普通操作状态下的读操作，确认数据能否被正常读出。此时，Vcc 及 VPP 设置为 5.25V（普通操作的上限电压）。

如果写入的所有数据都能够被正常读出，则结束操作；如果数据不一致，则作为器件异常以出错结束操作。

●1.6.3　UV－EPROM擦除器的制作

UV-EPROM的擦除操作是通过紫外线进行的。市场上虽然也有擦除器销售，但利用市场上销售的紫外线灯（杀菌灯）可以自己制作擦除器。制作非常简单，笔者曾经尝试着制作了一个。

1.　EPROM擦除器的电路

图是笔者尝试制作时使用的EPROM擦除器电路，这比真正的电路图简明易懂，是以与实际配线图近似的形式描绘的。紫外线灯也与荧光灯一样具有各种不同的大小和瓦数。如果是个人使用的擦除器，则选择较小的紫外线灯即可。我选择了 4W的紫外线灯。另外，镇流器、启辉器的选择也要与紫外线灯相匹配。

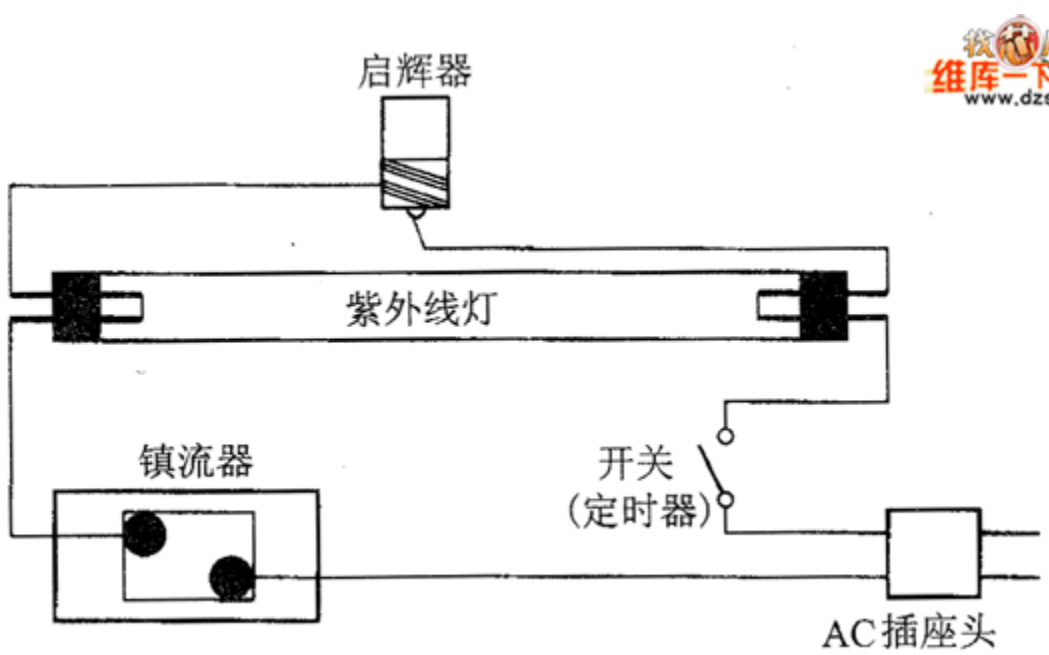


图 UV－EPROM擦除器电路

外壳使用铝盒等现成的盒子即可，只是要注意外壳在擦除过程中能保证紫外线不会直接射人眼中。

正如图中所示，紫外线灯的设置与普通的荧光灯相同。最近，市场上销售较多的是利用转换器缩短从打开开关到点灯这段时间的元器件。但EPROM擦除器不需要那样的功能，所以只利用非常简单的镇流器及启辉器制作即可。

如果为了缩短擦除时间，则应该在开关部位添加定时器等。但业余的可以利用厨房的定时器等，经过一定时间后手工切断开关，这样就可以非常简单地完成制作了。

第二章

闪速存储器的结构与使用方法

2.1　闪速存储器的概要

闪速存储器的基本存储器单元结构如图 1 所示。一眼看上去就是n沟道的MOSFET那样的东西，但又与普通的FET不同，特点是在栅极（控制栅）与漏极 / 源极之间存在浮置栅，闪速存储器利用该浮置栅存储记忆。

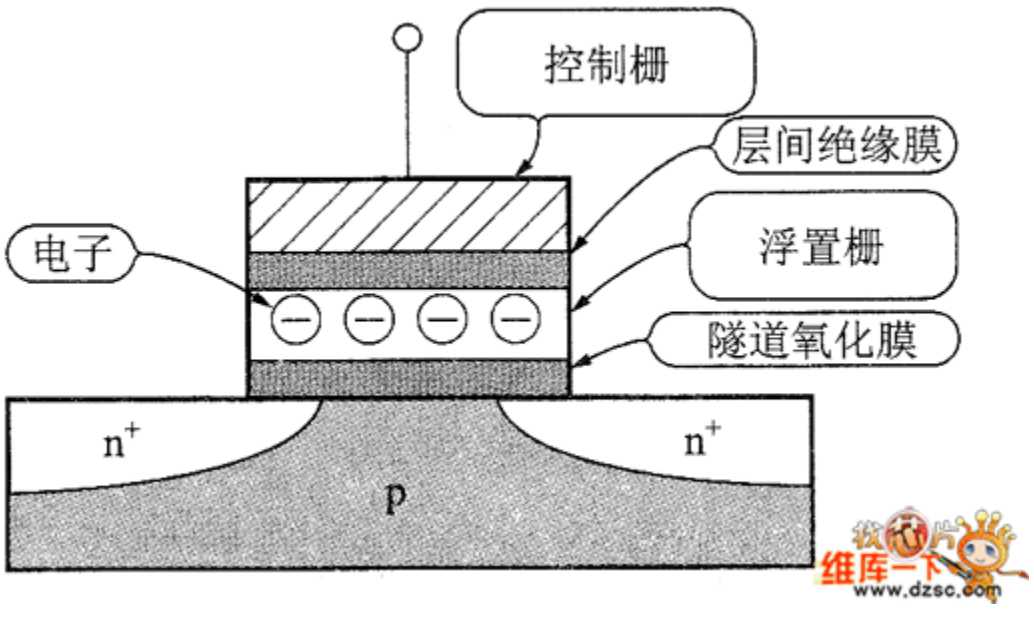


图 1 闪速存储器的单元结构

浮置栅被设计成可以存储电荷的构造，栅极及主板利用氧化膜进行了绝缘处理，一次积累的电荷可以长时间（10 年以上）保持。当然，如果氧化膜存在缺陷，或者由于某种原因使绝缘膜遭到破坏，那么闪速存储器将失去记忆。同时，因为热能必定致使电荷以某概率发生消减，因此数据保存的时间将受到温度的影响。

下面，我们将进一步讨论闪速存储器的擦除与写人的原理。

我们知道，数据的写人与擦除是通过主板与控制栅之间电荷的注人与释放来进行的。例如，一般的NOR闪速存储器在写人时提高控制栅的电压，向浮置栅注人电荷（图 2）。而数据的擦除可以通过两

种方法进行。一种方法是通过给源极加上＋12V左右的高电压，释放浮置栅中的电荷（Smart Voltage Regulator）；另一种方法是通过给控制栅加上负电压（-10V左右），挤出浮置栅中的电荷（负极门擦除法）。各种电压提供方式如图 3 所示。

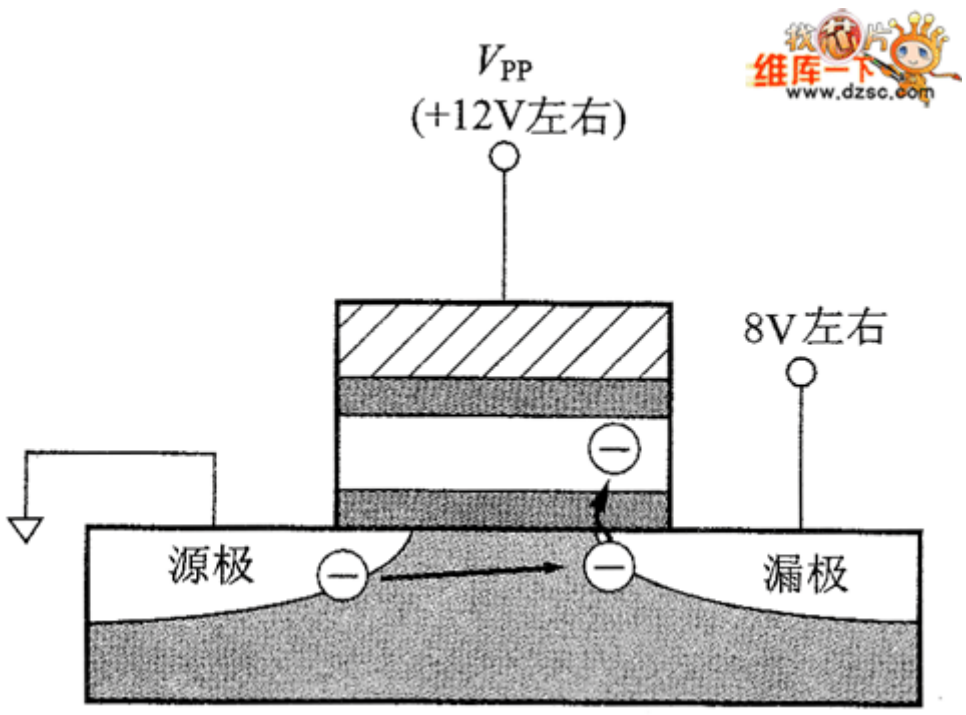
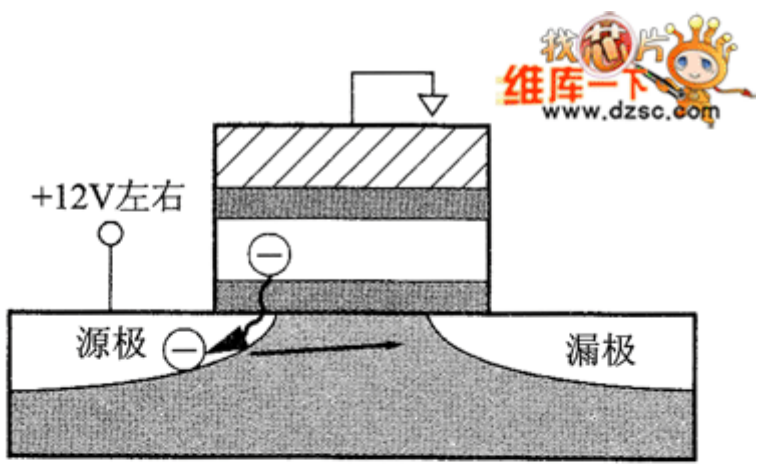
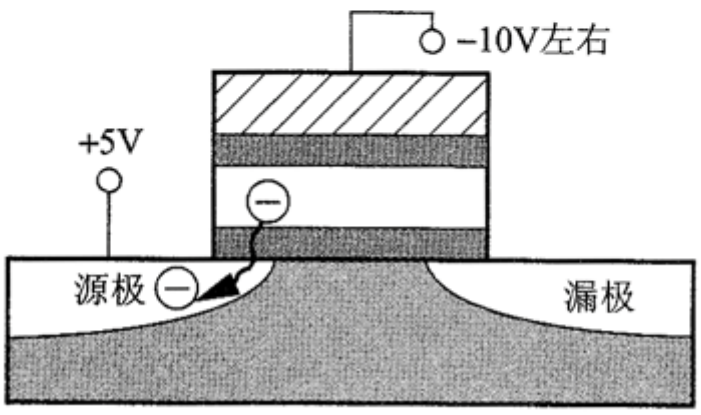


图 2 闪速存储器的写入操作



(a) Smart Voltage Regnlator



(b) 负极门擦除法

图 3 闪速存储器的擦除操作

图 4 图示了闪速存储器单元的电压－电流特性。浮置栅的电荷可抵消提供给控制栅的电压。也就是说，如果浮置栅中积累了电荷，则阈值电压（V_{th}）增高。与浮置栅中没有电荷时的情况相比，如果不给控制栅提供高电压，则漏极－源极间不会处于导通的状态。因此，这是判断浮栅中是否积累了电荷，也就是判断是“1”还是“0”的机制。

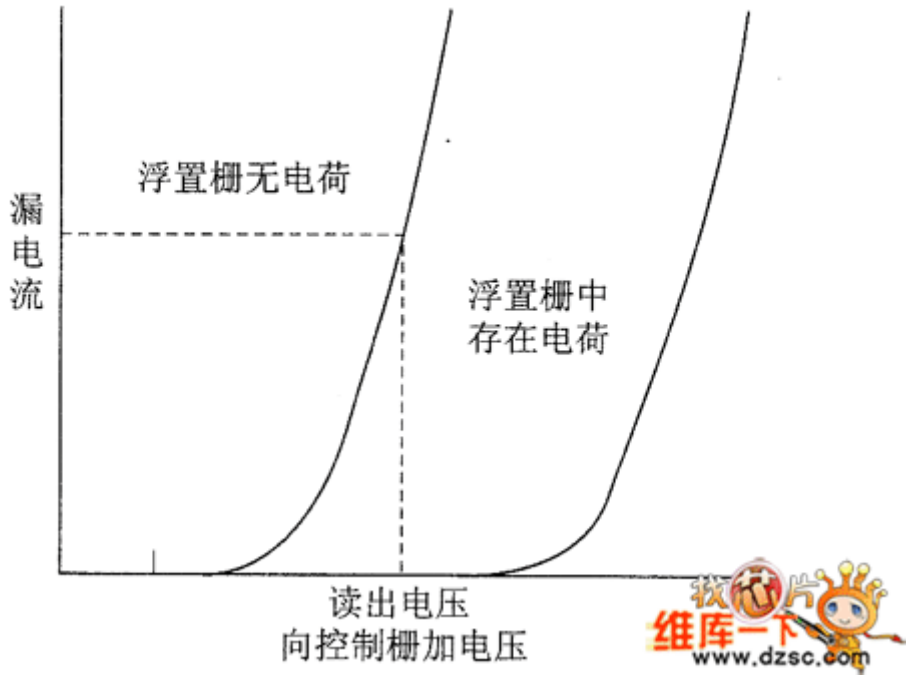


图 4 闪速存储器单元的电压一电流特性变化

那么，写入操作是提高了V_{th}还是降低了V_{th}呢？根据闪速存储器的类型情况也有所不同。作为传统EPROM的一般替代晶的NOR以及硅盘中应用的NAND闪速存储器，在写入时为高V_{th}；而AND及DINOR闪速存储器中，在写人时为低V_{th}。

2.2 闪速存储器的分类及特征

闪速存储器根据单元的连接方式，如表所示，可分成NAND、NOR、DINOR（Divided bit Line NOR）及AND几类。NAND闪速存储器单元的连接方式如图 1 所示，NOR闪速存储器如图 2 所示，DINOR闪速存储器如图 3 所示，AND闪速存储器单元的结构如图 4 所示。市场上销售的闪速存储器基本上就是NOR及NAND两种，其中只有NAND闪速存储器的单元是串联的，其他所有类型的单元都是并联的。

表 闪速存储器的单元方式

种 类	单元的连接方式	逻 辑	写入方法	擦除方法	数据存取
NAND	串联	写入时为高 V_{th}	隧道注入	隧道释放	顺序存取
NOR	并联	写入时为高 V_{th}	热电子注入	隧道释放	随机存取
DINOR	并联(数据线分层)	写入时为低 V_{th}	隧道注入	隧道释放	随机存取
AND	并联(数据/源线分层)	写入时为低 V_{th}	隧道注入	隧道释放	顺序存取

NOR闪速存储器以读取速度 100ns的高速在随机存取中受到人们的青睐。但由于其单元尺寸大于NAND闪速存储器，存在着难以进行高度集成的问题。写人时采用CHE（Channel HotElectron，沟道热电

子）方式，即在栅－漏之间加上高电压，提高通过沟道的电子能量，向浮置栅中注入电荷。这样，由于损耗电流变大，在写入时必须由外部其他途径提供＋12V左右的电源，因而不适合低电压操作。

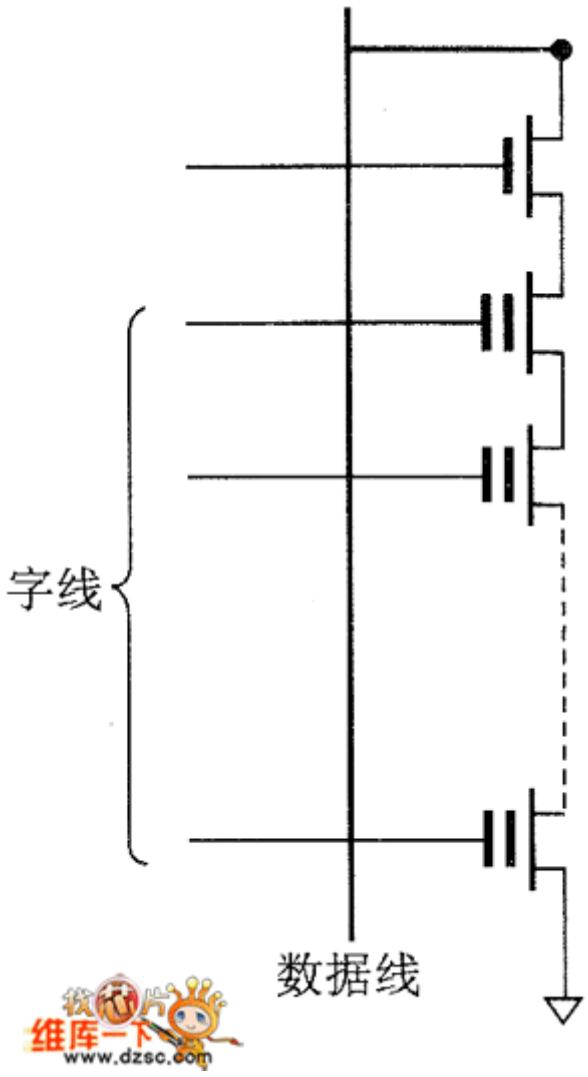


图 1　NAND 闪速存储器的单元结构

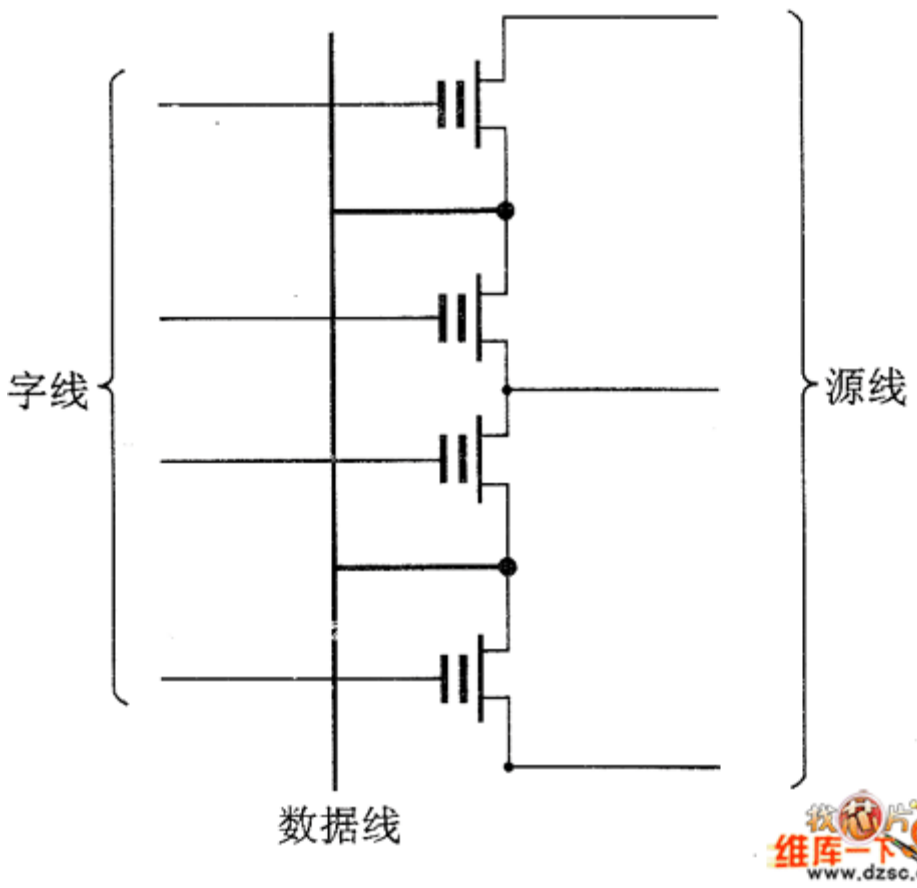


图 2　NOR 闪速存储器的单元结构

与 NOR 闪速存储器相比较，东芝公司开发的 NAND 闪速存储器却能够进行高度集成，写人方式也因采用了被称为隧道的方式，即利用了氧化膜所引起的隧道效应现象，故与 NOR 闪速存储器相比，具有

损耗电流较小的特征。但在另一方面，由于单元是串联连结的，所以面向顺序存取，具有随机存取速度慢的缺点。

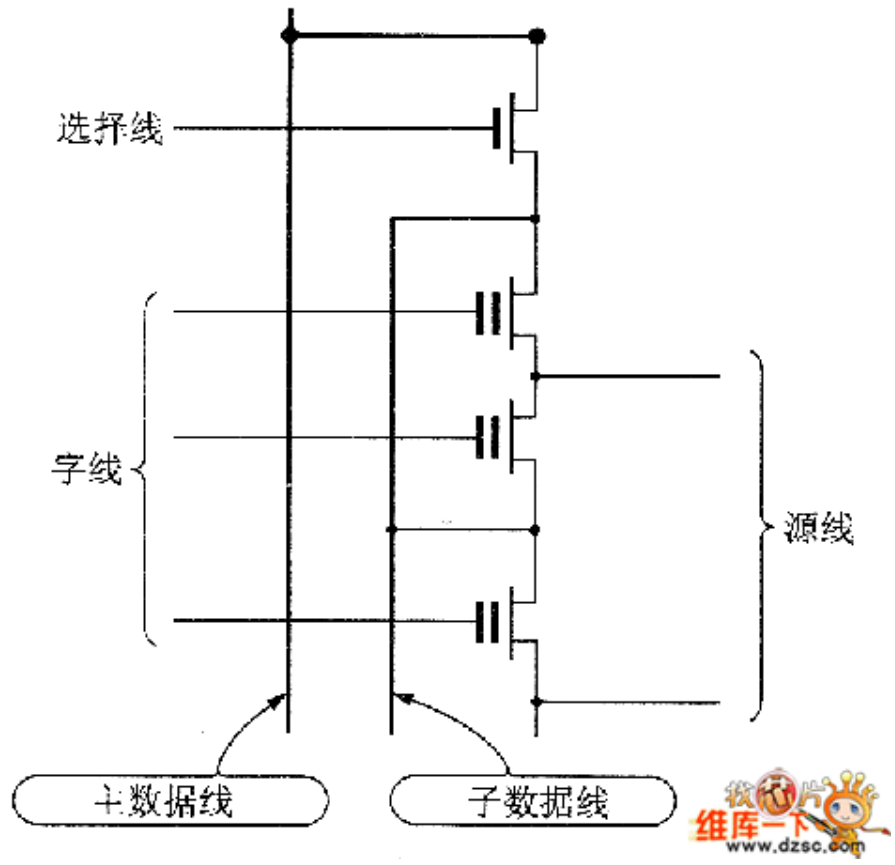


图 3　DINOR 闪速存储器的单元结构

三菱与日立结合NAND及NOR闪速存储器的特点，开发了DINOR（Divided bit－line NOR）闪速存储器以及AND闪速存储器。

DINOR闪速存储器的结构是将数据线（位线）分离成主数据线与子数据线的层次，通过各个存储器单元与子数据线的连接，既可以具有像NAND那样的高度可集成性，又具各与NOR同等程度以上的高速随机存取性。因为写人操作也采用了隧道方式，所以较小的写入电流就可完成写人操作。又因数据置换所需要的高电压升压电路可以设计于芯片内部，因此可以进行低电压的单一电源操作。

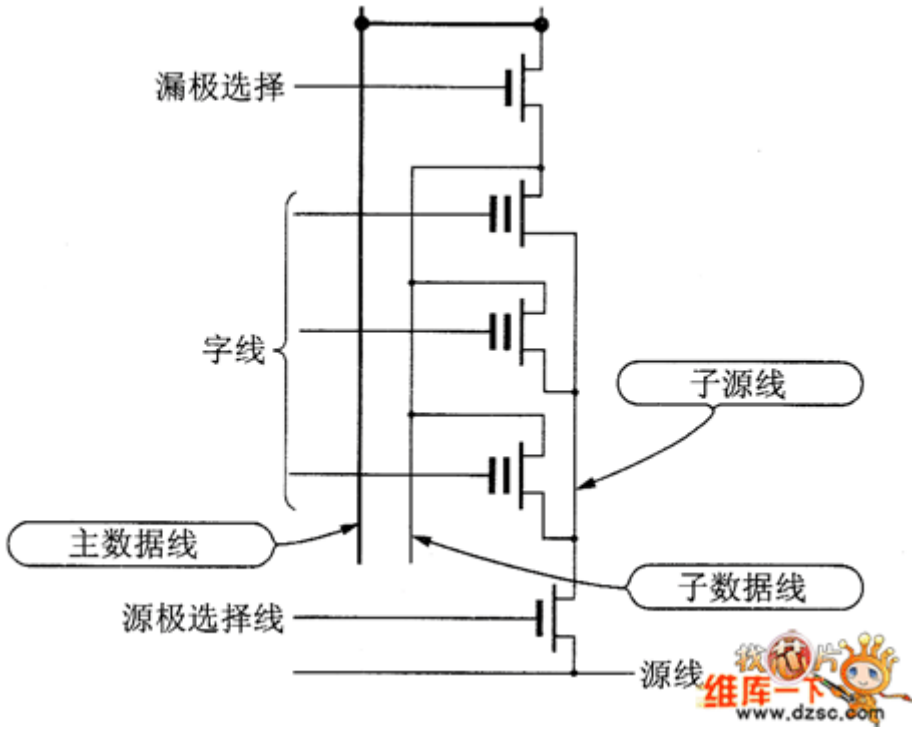


图 4　AND 闪速存储器的单元结构

AND闪速存储器单元的源线也设计了分离的子源线，是倾向于顺序存取的产品。除了能够以与硬盘一个扇区相同的 512 字节大小的小块单位进行写人及读取操作以外，还具有DINOR的低功耗特长，可以面向硅盘等展开应用。单元的连接方式与NOR闪速存储器相同，写入逻辑为反相（NOR写人时V_{th}变高，而AND式则降低），命名为AND式。

现在的NOR闪速存储器也致力于改良，目的在于将写人操作也采用隧道方式以降低功耗，或者通过单元物理结构上的改善等，使低电压单一电源类型的闪速存储器也形成产品。以文件为使用目的的AND及NAND两种类型的闪速存储器目前已在市场上流通，应用于大容量的Flash ATA卡等方面。

2.3　NAND 闪速存储器

●2.3.1　TC58V64 的引脚配置

TC58V64的引脚配置如图所示。在图中未看到地址引脚，这是因为利用数据输人输出引脚（I／O_1～I／O_8），能够以时分方式赋予数据。NAND闪速存储器只能以基本的块单位进行顺序存取，所以不需要以随机存取为前提的地址引脚。

图　TC58V64AFT的引脚配置

接着我们针对各个信号引脚进行简单的说明。

▲ I/O_1～I/O_8

这样的信号引脚进行着地址、指令及数据的输人输出等，与ALE及CLE信号等配合，以时分方式复用。

▲ /CE（Chip Enable，芯片使能）

这是器件的选择信号。电平为低电平时，器件处于选择状态；为高电平时，器件处于非选择状态（低功耗状态）。

▲ /WE（Write Enable，写使能）

将I/O端置于输入（由主机向器件赋予数据等）状态。

▲ /RE（Read Enable，读使能）

这是用于使I/O引脚进行数据输出的信号。/RE也作为内部地址计数器进位的时钟来工作。

RE有效（为低电平），经过存取时间（tREA）后，I/O引脚上的数据得到确定，在/RE的上升阶段，内部地址计数器只向前移动一位。这样，单一的读操作就可以读出连续地址的存储器内容。

▲ CLE（Command Latch Enable，指令锁存使能）

这是向器件内部的指令寄存器中写人给予I/O引脚的操作指令码的控制引脚。在/WE信号上升与下降时，如果CLE有效（为高电平），则作为指令将被锁存。

▲ ALE（Address Latch Enable，地址锁存使能）

这是判断主机给予I/O引脚的数据是地址还是数据的信号。ALE如果有效（高电平），则作为地址来处理；ALE如果无效（低电平），则作为数据进行处理。

▲ /WP（Write Protect，写保护）

强制禁止进行写人及擦除操作。如果/WP 有效（低电平），则芯片内部的升压电路被复位。由于存储器单元不能生成用于写人的高电压，所以即使发出了指令，替换操作也不能被执行。

在上电、断电以及容易使操作不稳定的情况下，使该引脚持续有效，则系统处于安全状态。

▲ R / B（Ready / Busy，就绪 / 忙输出）

这是用于向外部通知器件内部操作状态的信号，是漏极开路输出。如果内部正在进行操作，则该信号有效（为低电平）；如果内部操作完成，则该信号无效（为高电平）。

这些控制信号的组合与工作状态的关系如表所示。

表 TC58V64的操作

操作模式	控制信号线(X:无影响)					
	CLE	ALE	$\overline{CE}$	$\overline{WE}$	$\overline{RE}$	$\overline{WP}$
指令输入	H	L	L	↑	L	X
数据输入	L	L	L	↑	H	X
地址输入	L	H	L	↑	H	X
串行数据输出	L	L	L	H	↓	X
编程期间(Busy)	X	X	X	X	X	H
擦除期间(Busy)	X	X	X	X	X	H
编程擦除禁止	X	X	X	X	X	L

● 2.3.2 NAND 闪速存储器的内部结构

TC58V64的内部结构如图所示。闪速存储器的容量增大，则块数也将增加，但内部的基本结构没有改变。

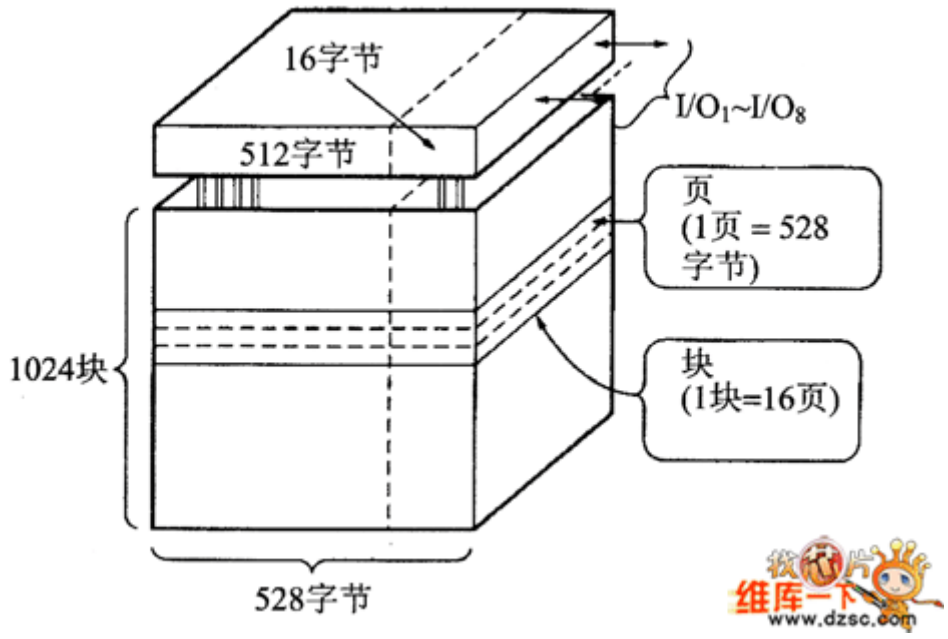


图 TC58V64 的内部结构

NAND 闪速存储器的特点

①按顺序存取数据；

②存储器内部以块为单元进行分割，而各块又以页为单位进行分割；

③以块为单位进行擦除操作；

④以页为单位进行编程（写人）；

⑤页的大小存在尾数（TC28V64 是 528 字节）；

⑥也存在有坏块（bad block）的产品。

以NAND闪速存储器为主的用途是类似硅盘那样的文件保存器件。所谓的 528 写入页大小，就是在 512 字节（相当于一般的硬盘扇区大小）上增加了 16 字节的冗余数据字节，通过在这些冗余字节上添加纠错码，在写人与擦除的重复操作中，即使闪速存储器单元发生异常，数据也可以复原。

另外，作为擦除的单位，即一块 16 页（转变为数据大小就是 512 字节×16＝8K）最好与主机端的文件管理单位一致。

当读者看到特点⑥，即和普通存储器器件特点相同的最后一条，应该稍微有些吃惊吧。在产品供货期间，所有数据未成为FFH的块都作为坏块不进行擦除操作，必须做到在主机的文件管理软件中不利用这些坏块。TC58V62 最多容许有 10 个坏块，由于TC58V64 具有1024个块，所以它有时只能利用1014个块。

●2.3.3　操作指令

TC58V64的操作指令（通过指令输人所给予的编码）如表所示。只有自动块擦除是 2 字节指令，其他都是 1 字节指令。

表　TC58V64的操作指令

指　令	指令数据		备　注
	第一周期	第二周期	
串行数据输入	80h		
读模式(1)	00h		
读模式(2)	01h		
读模式(3)	50h		
复位	FFh		Busy 信号有效时可发布
自动编程	10h		
自动块擦除	60h	D0h	
状态读	70h		Busy 信号有效时可发布
ID 读	90h		

下面我们就针对这些指令与存取操作进行说明。

▲数据读操作

TC58V64的读操作如图 1 所示。随着指令，利用 3 个字节指定读出开始地址，开始进行数据读操作，地址及指令在WE信号的上升沿被锁存。

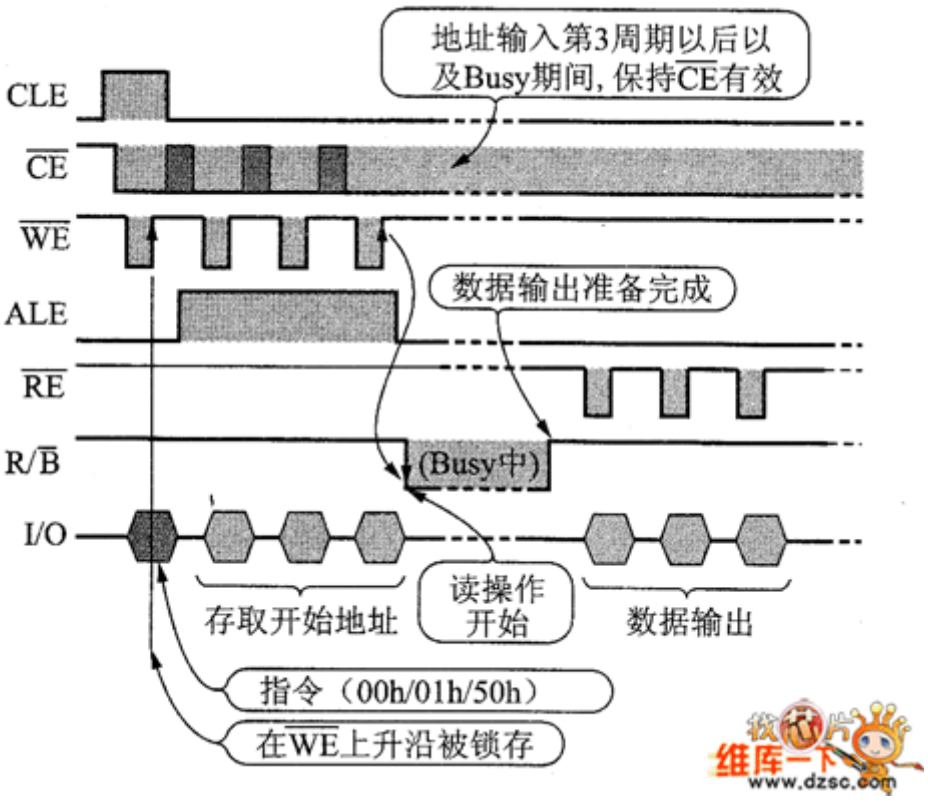


图 1　TC58V64 的读操作

读操作时地址的指定方法如表 1 所示。最先给予的数据称为列地址，后面的两个周期中所给予的数据称为页地址。TC58V64 的存储器具有1024个块，由于各个块又分割成 16 个页，所以A22～A13 为块号（块地址），A12～A19 为块内的页号（块内的NAND地址）。

表 1 指定地址的方法

地址周期	I/O 引脚							
	I/O_8	I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1
第一周期	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
第二周期	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9
第三周期	“0”	“0”	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}

读指令如果是 00h，则 A8 = “0”，如果读指令为 01h，则 A8 = “1”，所以没有指定 A8 的位。

A22～A13：块地址

A12～A9：块内 NAND 地址

A7～A0：列地址

这里有意思的是 A8 没有指定，而 1 页的大小为 528 字节，大于 512。这样 A9 就成为了页地址。

在此，将一页（528 字节）分成 256＋256＋16 这样的结构（假设称为段）可能更容易理解。根据从一页的哪个段开始读，指令将被分离，读指令具有 3 个种类（00h，01h，50h）正是因为这个原因。而从各个段中的什么部位开始读是由列地址过渡的。当然，在最后的 16 字节的地方，列地址只能取得 0～15 的值。

在指令代码为 00h，01h 的情况下，将从读取开始地址到一页的最后地址传输数据，一旦到达最终的地址，则输出下一页的起始数据。仅当指令代码为 50h[读指令(3)]的时候，如果到达页的最后部分，则由下一页的第 512 个字节（第 3 部分）开始输出，而不返回到页的起始。

这些操作模式如图 2、图 3、图 4 所示。

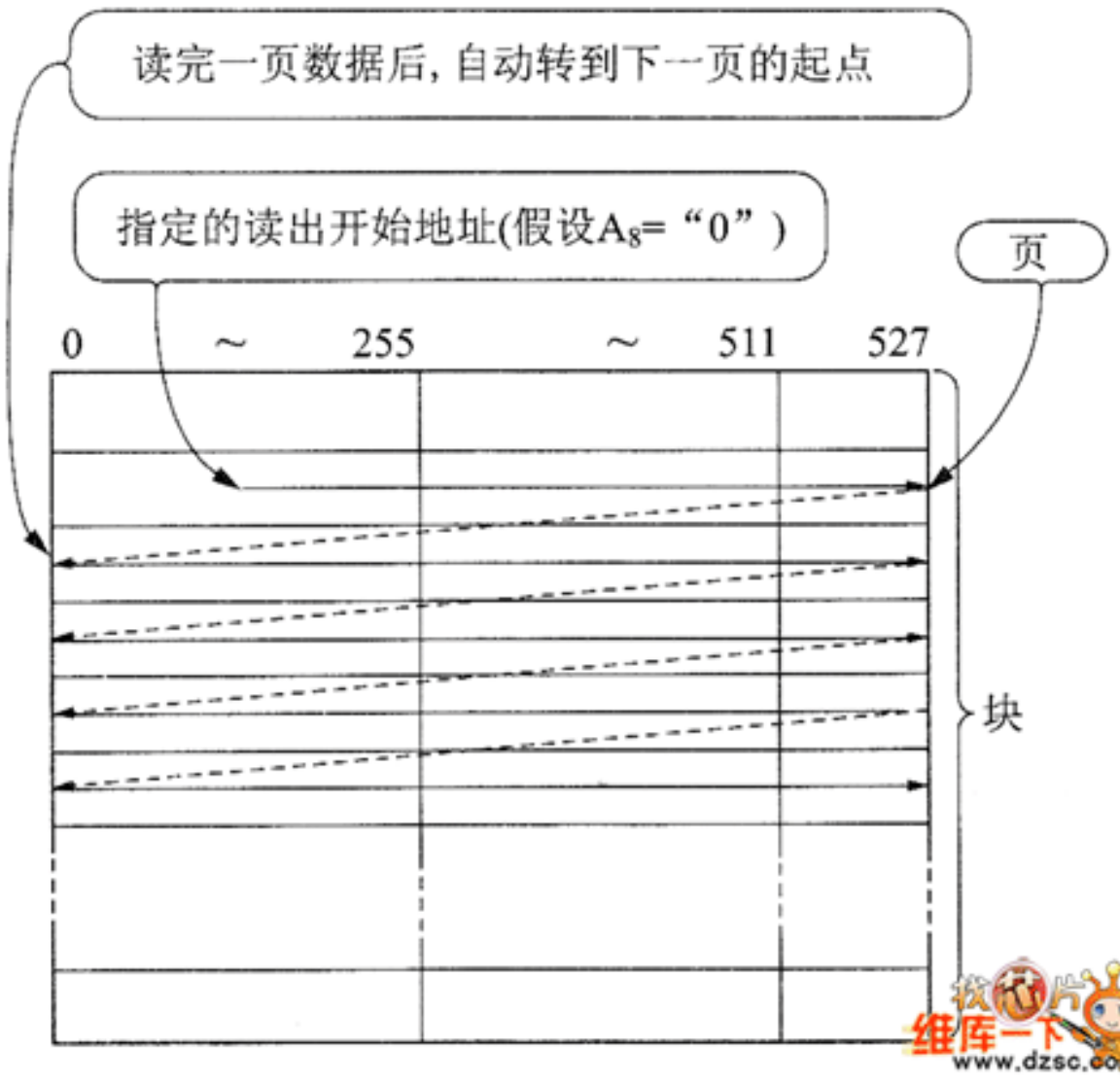


图 2 顺序读（1）

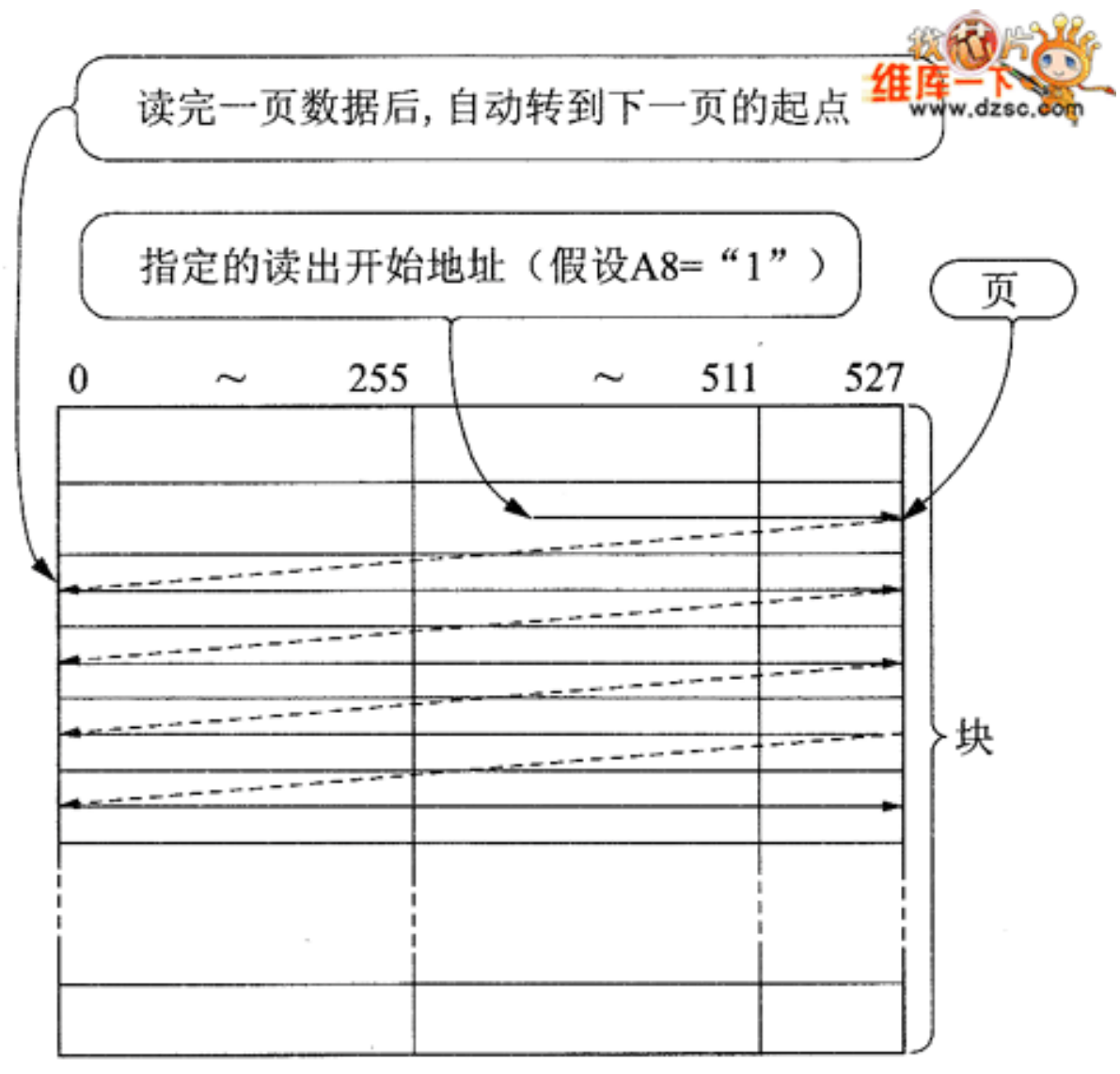


图 3 顺序读（2）

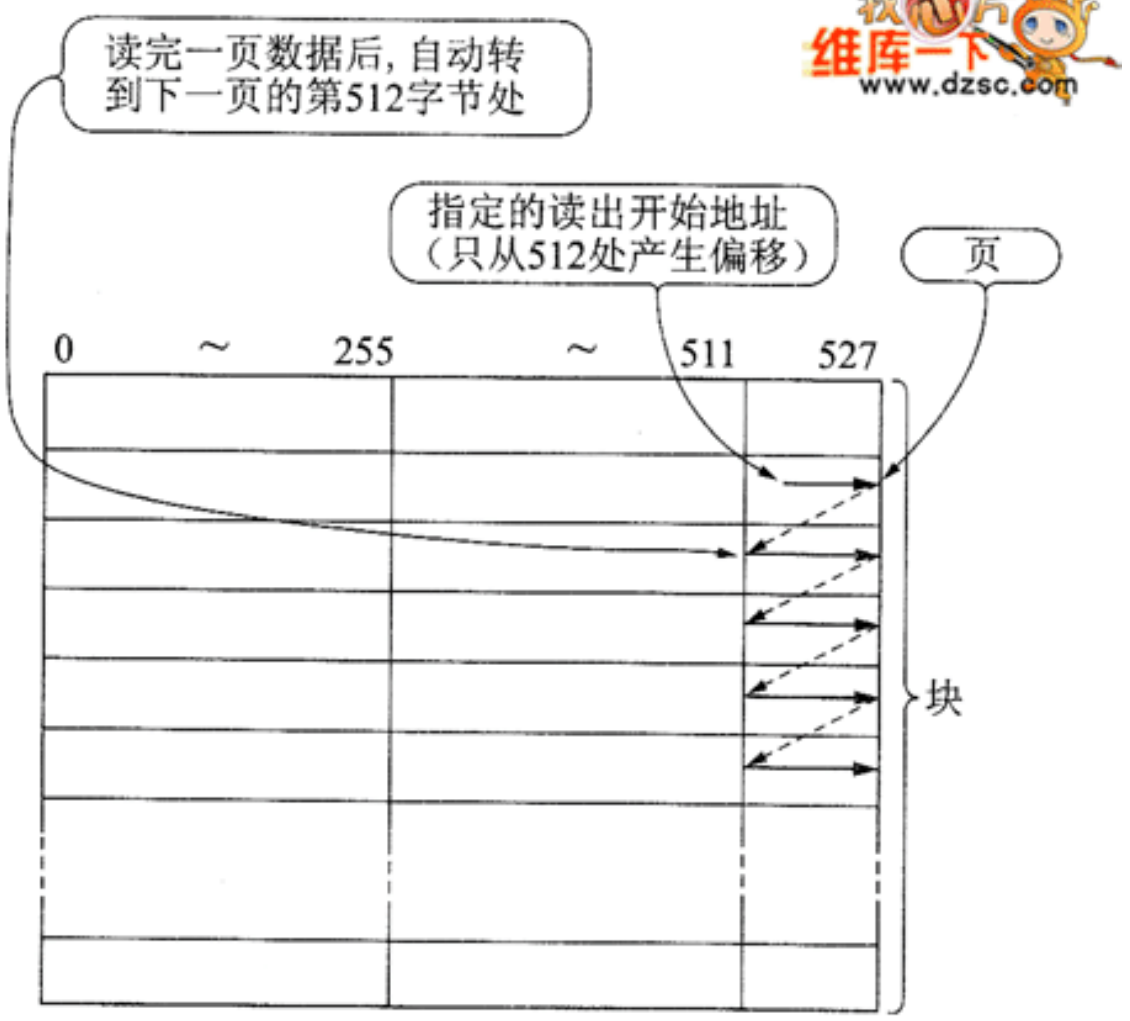


图 4 顺序读（3）

如图 5 所示，当进行读操作时，在指定地址后到读取最初数据之间以及到达页的结尾部位后。需要一定的时间返回到下一页的起始部分（顺序读（3）的情况下为第 3 部分的起始）。这期间Busy信号有效，需要让来自主机的访问等待。这段时间可以认为是内部一页的数据（528 字节）传输到页缓冲器上的时间。

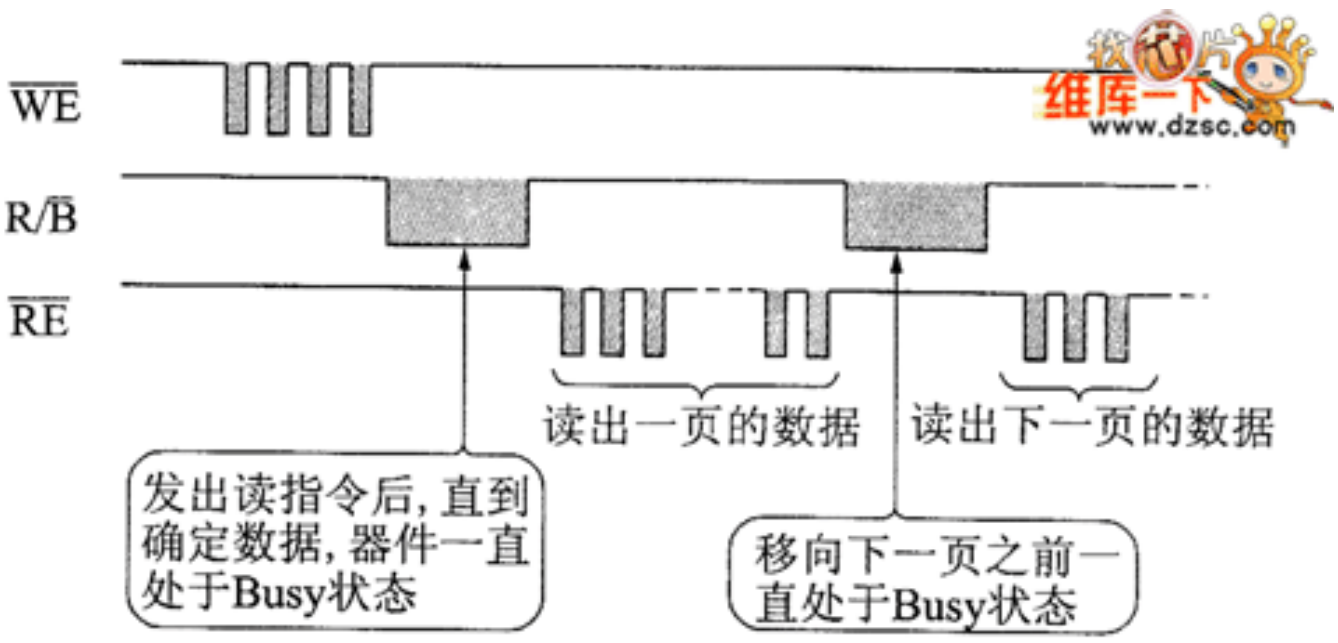


图 5 读操作中的 Busy 状态

▲状态读

状态读的操作如图 1 所示，如果跟着指令 70h进行读取操作，则读出存储器的状态。存储器的状态数据如图 2 所示。

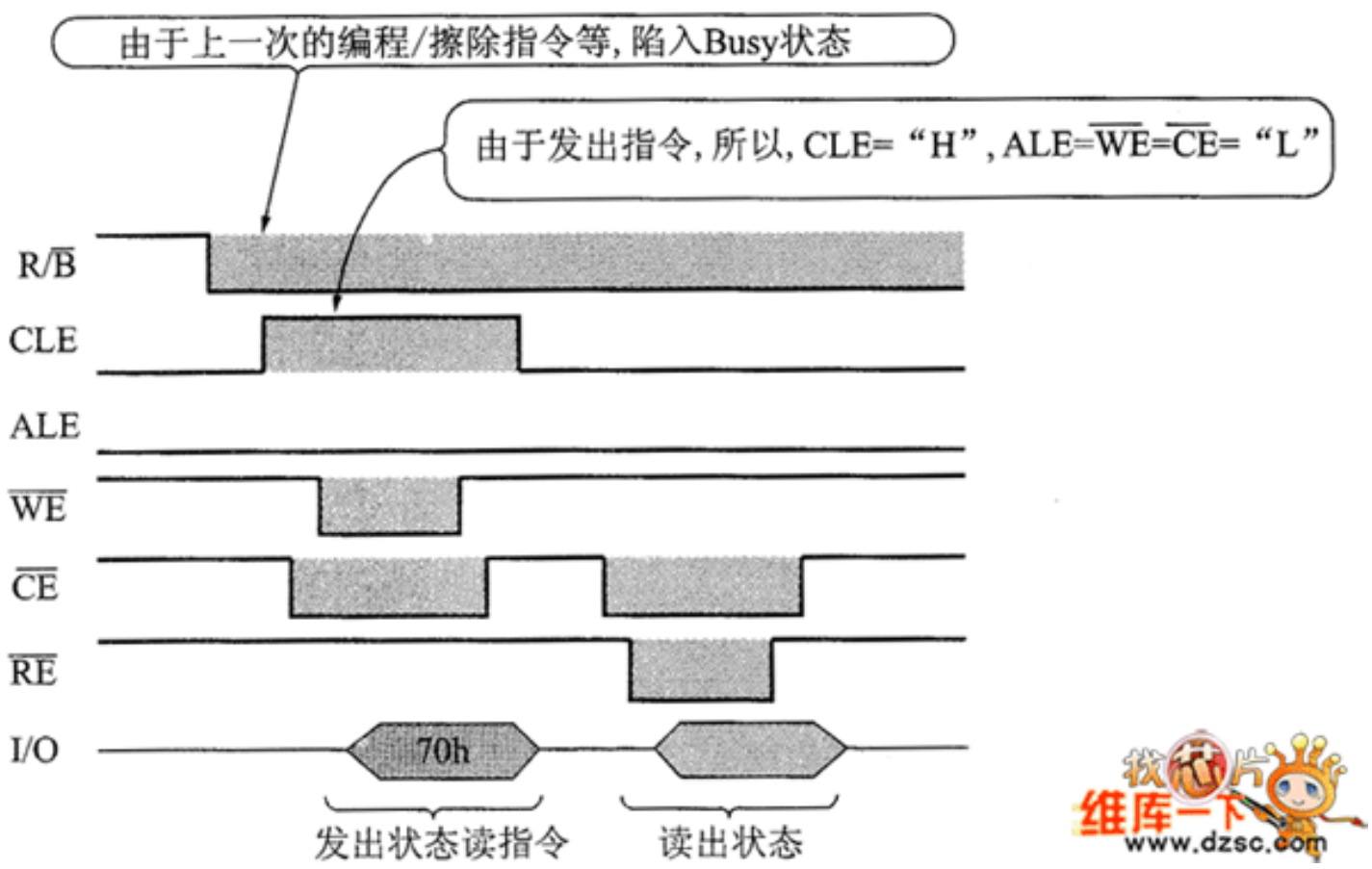


图 1　状态读操作

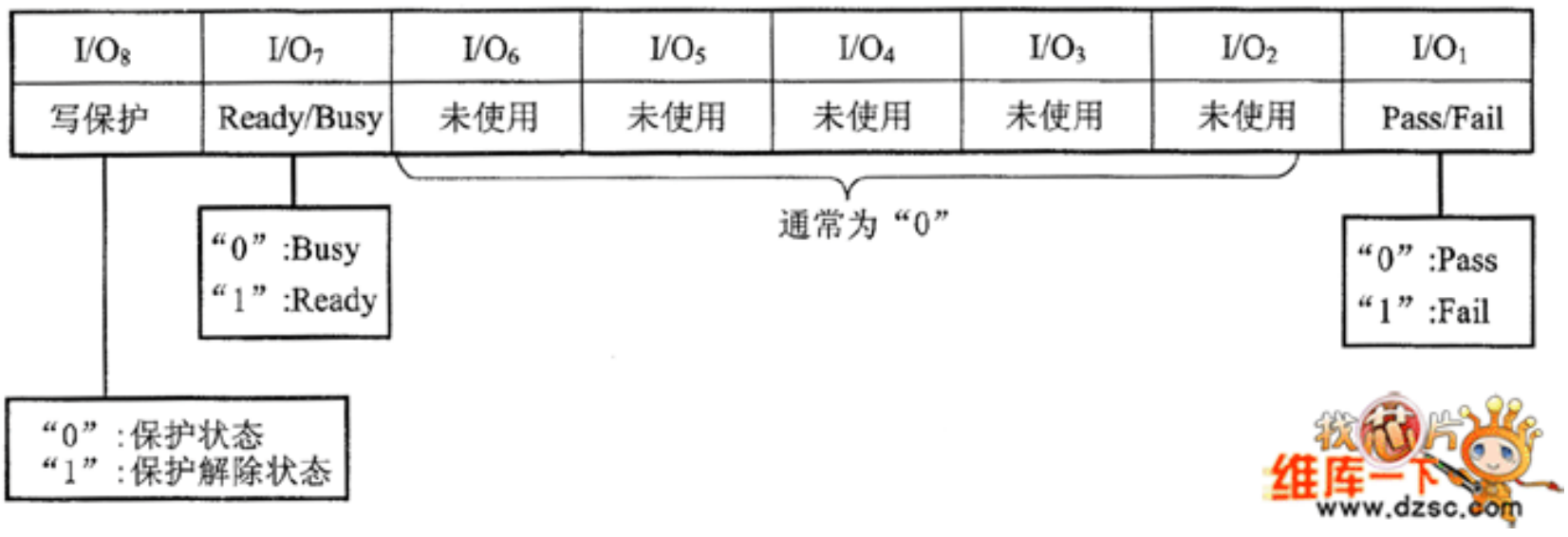


图 2　TC58V64 的状态

▲自动页编程操作

自动页编程操作按照如下的程序进行：

①发出数据输人指令（80h）；

②地址 / 数据输入；

③发出自动编程指令（10h）。

图对上述流程进行了图示。在器件的内部工作中，在②中所给予的数据不是直接写人存储器中的，而是先暂时存人页缓冲器中。通过③开始送出写入指令，然后转发到②所提供的页地址处。

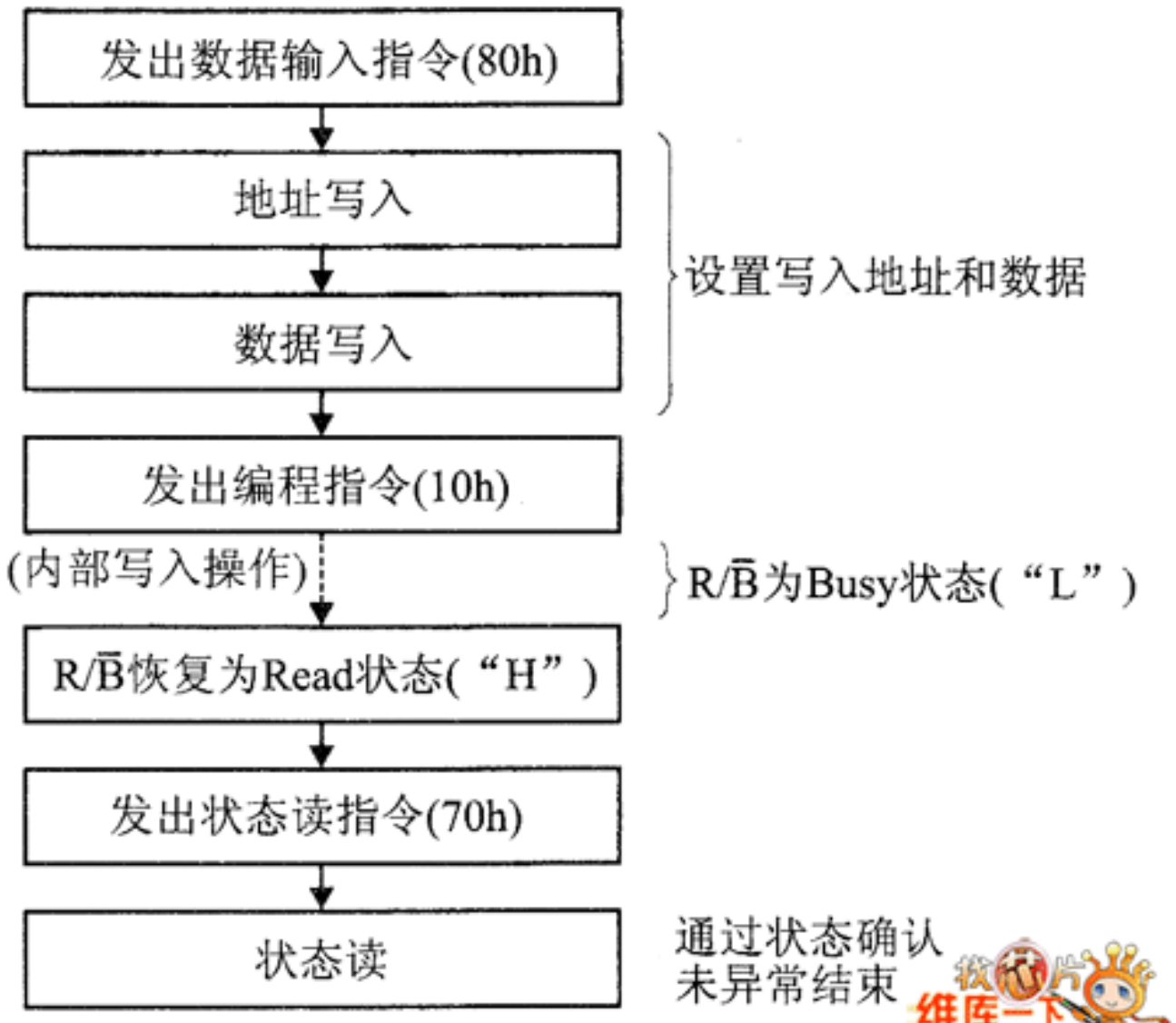


图 自动页编程的操作流程

此时，在芯片内部进行写人验证。如果正常，则恢复R / B设置为Ready状态（为高电平）后正常结束。如果由于某些异常导致写人验证未正常进行的，则在内部自动进行重试，如果重试达到所规定的次数，则将R / B设置为Ready状态的同时结束异常。

▲自动块擦除

自动块擦除是将指定的块的内容一次性设为 FFh。正如先前所描述的那样，虽然写入操作是以页（528 字节）为单位进行的，但擦除只能以块为单位进行。由于块擦除所提供的地址是块地址，所以占有 2 个字节。

自动块擦除的工作流程如图所示。与自动页编程的操作相同，在内部也进行自动重试，如果重试规定次数以后仍然存在异常，则将 R / B 设置为 Ready 状态的同时结束异常。

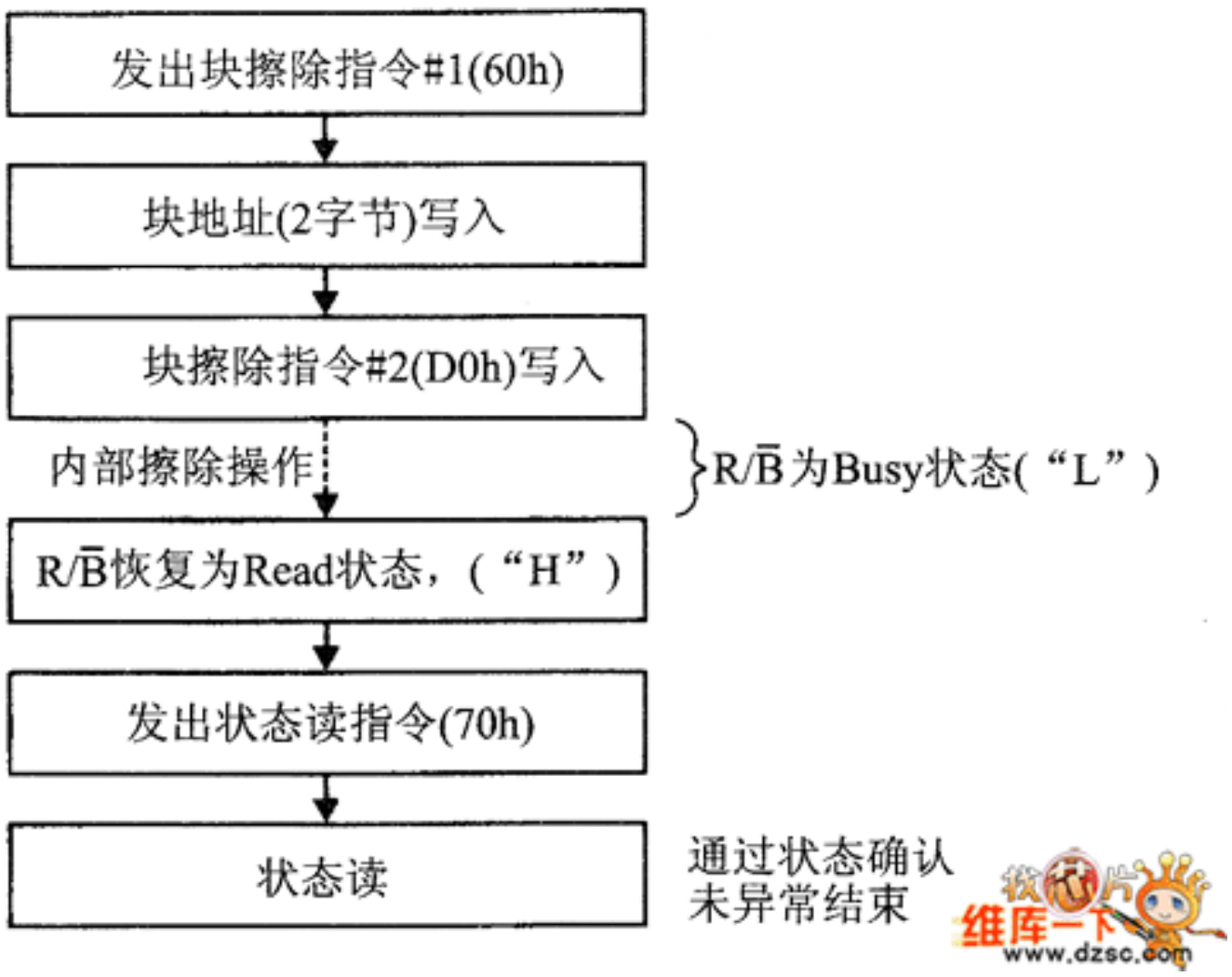


图 自动块擦除的操作流程

▲复位操作

复位操作通过指令代码FFh的写入来进行。一旦发出该指令，内部用于编程 / 擦除的升压电路将断开，编程 / 擦除电压放电直至电压为 0V。在放电这段时间内，R / B信号维持为Busy状态（低电平）。

利用复位指令，地址寄存器的各个位全部为“0”，数据寄存器中的位都为“1”。

▲ID 读操作

ID代码是为了主机自动识别与总线连接的器件的制造商名称 / 器件型号而设置的。ID读指令用于读取该ID代码。

图 1 图示了ID读操作的流程。图 2 图示了发送ID读指令的操作。

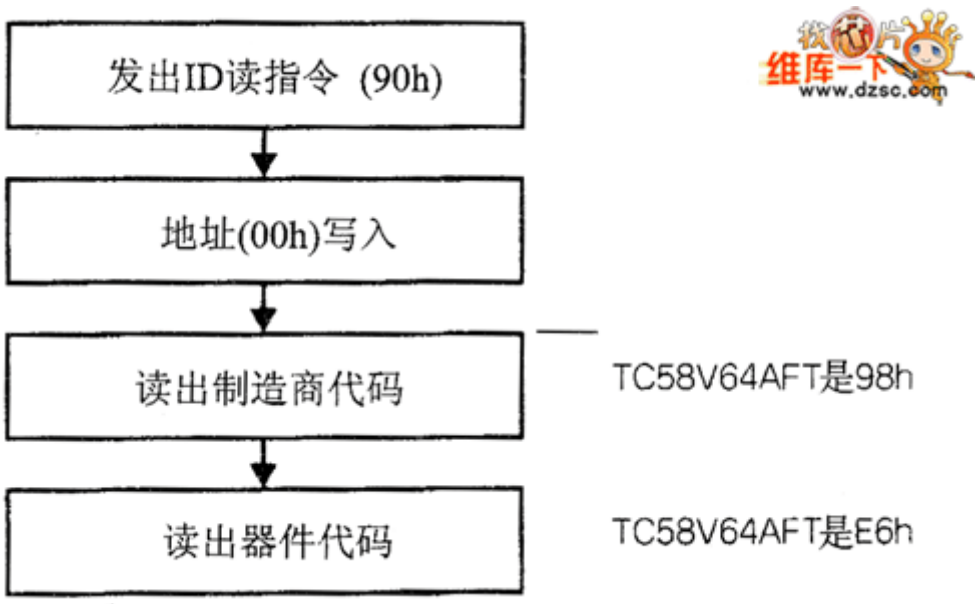


图 1 ID 读指令的躁作流程

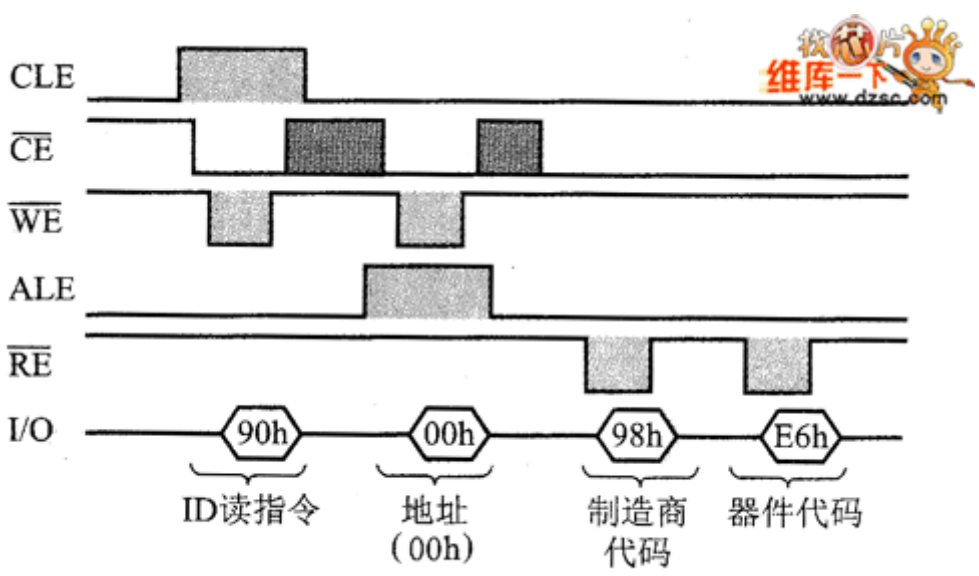


图 2 ID 读指令的操作

在读取ID时，发送ID读指令（90h）之后，写人 00h作为地址。由于 90h是指令，所以虽然将CLE设为有效（高电平），但由于紧跟着的 00h是地址值，所以CLE无效（低电平），而ALE有效（高电平）。

完成这样的流程后，通过下面 2 次的读取，可以读出制造商代码（TC58V64为 98h）和器件代码（TC58V64 为E6h）。

2.4　NOR 闪速存储器

●2.4.1　引脚配置

在器件的使用方面，必须了解引脚配置以及各个引脚所代表的意思，因此我们现在首先调查引脚的配置。下载Am29F010A的数据手册后，没有关于DIP封装的记载。但是，因为事实上Am29F010 的DIP类型是存在的，所以引脚配置肯定已经确定了。我们试着与相同系列的前一产品Am29F040B的引脚配置进行比较。Am29F010A的PLCC式的引脚配置如图 1 所示，Am29F040B的PLCC式与DIP式的引脚配置如图 2 所示（Am29F010A与 040B除此之外都还有TTSOP式的封装）。

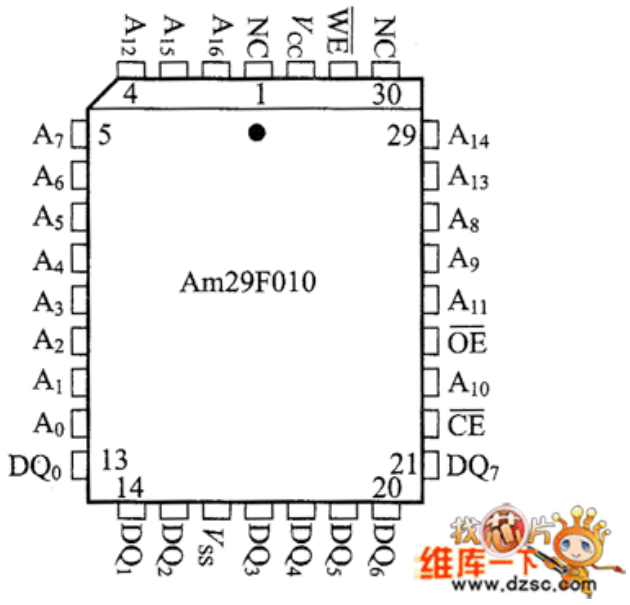


图 1 Am29F010 的引脚配置（PLCC 封装）

将 Am29F010 与 Am29F040B 所具有的 PLCC 式的引脚配置进行比较，我们可以知道，前者中为 NC(Not—Connected，无连接）的引脚 6 与 9 只是配置了地址的 2 个高位（A17 和 A18）。

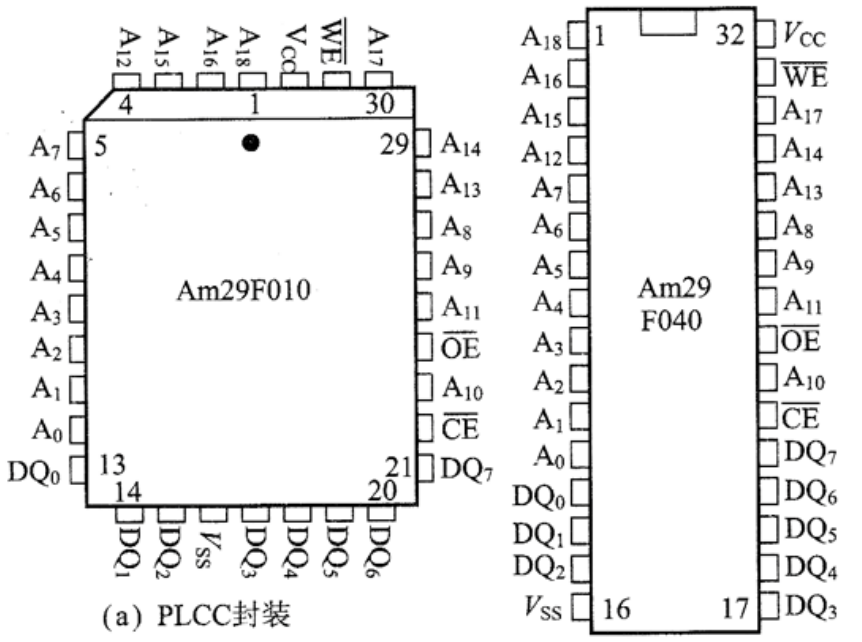


图 2 Am29F040 的引脚配置

因此，对于Am29F010A的DIP式的引脚配置，也可以认为只是将Am29F040B的DIP式的引脚 30 和引脚 1 作为NC来使用。

目前我们都是在研究封装器件的内部结构，封装的外形虽然存在各种各样的形式，但取出其中芯片的管芯（die）则是完全一样的。管芯周围填充了为引出信号而布置的垫片，由垫片引出细电线（接合线）与封装的引脚相连接，由于不会有人故意将引线交叉设置，所以信号按序排列的管芯即使封装改变也不会有任何影响。

当然，接合线本身尽可能短，不但在成本方面，而且在特性方面也是有利的，所以封装中管芯的方向需要根据封装形式而加以改变。想象芯片中管芯以哪种方向进行配置以及如何布线是一件有意思的事情。

●2.4.2　信号的种类

Am29F010引脚的分组如图所示。Am29F040只是地址总线增加到 18 根，没有NC引脚，其他方面与Am29F010 完全相同。闪速存储器的操作与地址、数据、/CE、/OE、/WE的组合如表所示。

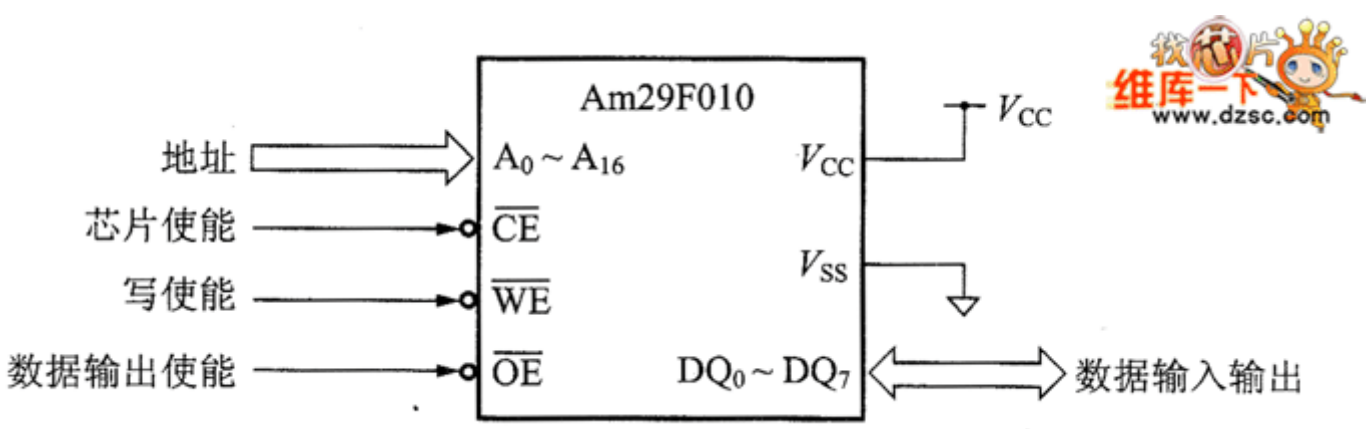


图 Am29F010 的引脚分组

操　　作	$A_0 \sim A_{16}$	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	$DQ_0 \sim DQ_7$
读	地址输入	“L”	“L”	“H”	数据输出
写	地址输入	“L”	“H”	“L”	数据输入
待机	X	$V_{CC} \pm 0.5V$	X	X	高阻抗
输出禁止	X	“L”	“H”	“H”	高阻抗
硬件复位	X	X	X	X	高阻抗

表 Am29F010A的操作模式

▲　VCC / Vss

这是电源引脚。因为Am29F010 和Am29F040 都是＋5V单一电源工作的FlashROM，因此，给Vcc加上＋5V电压，Vss作为标准电位为 0V。

▲　A0～A16（地址）

A0～A16 是地址引脚。虽然 Am29F010 是容量为 1M 位的闪速存储器，但由于是以 DQ0～DQ7 的 8 位为单位进行数据输入输出，因此地址就是 1M 位÷8 位＝128K，具有 17 根地址线。

▲　DQ0～DQ7（数据）

这是与外部进行数据传输的引脚。因为 Am29F010 通常是以 8 位为单位进行输人输出的，所以数据通常是以 8 位为单位进行读取与写人的。

▲　/CE（芯片使能）

这是器件的选择信号。只有该引脚为低电平时，下面所描述的/OE及/WE信号才有效。当与CPU等连接时，对CPU的高位地址进行解码，然后输人到该引脚。

当存在多个器件时，通常是/CE以外的引脚全部并联，由/CE引脚确定访问那个器件。

▲　/OE（输出使能）

只有在上面所说明的/CE引脚为低电平时/OE才有效。从Flash ROM读出数据时，如果/OE与/CE都为低电平，则一定时间后DQ0～DQ7 引脚上将出现数据。因为没有表示收集有效数据的信号，所以在外围电路中，根据数据手册中的值，预测数据已被确定的时间，然后读出出现在DQ0～DQ7 中的数据。

▲　/WE（写使能）

该引脚只是在先前说明的/CE 引脚为低电平时才有效。在赋予闪速存储器指令及写人数据等时，该信号与/CE 引脚一起同为低电平。

事实上数据是在/CE 及/WE 信号的上升沿（由低电平上升为高电平的时候）被提取到芯片内部的。

●2.4.3　与处理器的连接实例

表示闪速存储器与CPU的连接模式如图 1 所示。该图中信号名与信号的意思吻合ISA总线，通过地址译码器对CPU地址的高位进行解码，如果属于闪速存储器范围．则使/CE信号有效，将地址的低位赋予闪速存储器。

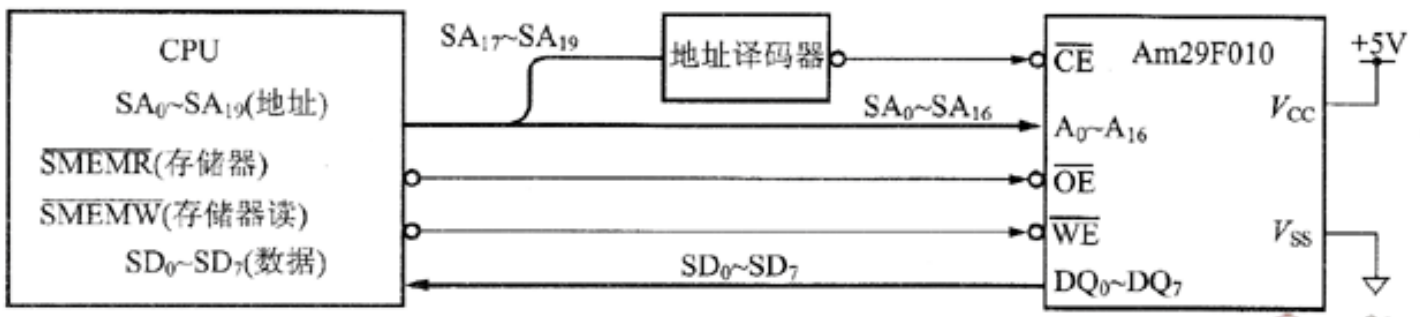


图 1　　8 位 CPU 与闪速存储器的连接思路

进而/OE与SMEMR信号、/WE与SMEMW信号相连，DQ_0～DQ_7连接于CPU的数据总线。

该图中特意考虑到时序的关系。根据CPU的总线工作时间，或者需要仔细计算时间，或者使其等待、延长总线周期等。ISA总线与近期的闪速存储器工作等相比，其速度是足够缓慢的，所以该图可以在添加缓冲器的电路中工作。

如果CPU的数据总线超过 8 位（16 位或者 32 位宽等），则需要排列若干个闪速存储器。但是一般的CPU即使数据总线具有 16 位或者 32 位，作为命令一般设计时也是以 8 位为单位进行输入输出的。为了适应这样的命令，一般CPU的外部数据总线也以 8 位为单位进行分组，并且设计了在读写时标明存取所对应的分组信号。

以图 2 为例，表示在 16 位的CPU中连接了 2 个Am29F010 的电路。这里的信号种类与名称是以ISA总线为基准的（事实上在ISA上进行 16 位的存取需要使MLMCS16 有效，在此省略）。与前图相比，我们知道增加了SBHE信号，该信号用于表示是否使用数据总线的高 8 位，对低 8 位的访问是由A0 确定的。表总结了ISA总线上存取操作与A0、SBHE信号的操作。信号名存在若干不同，但处理器基本上都采用相同的方法。

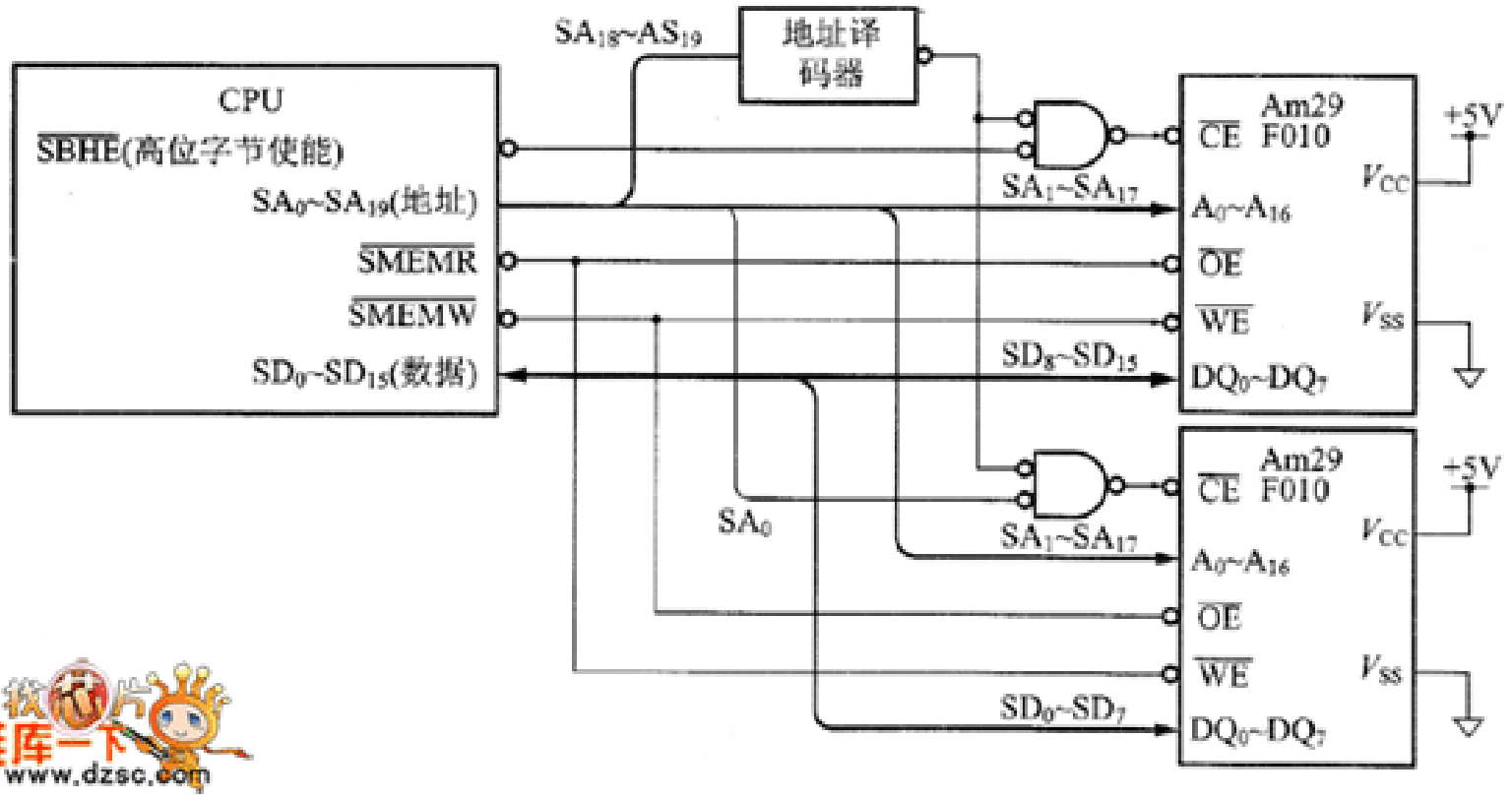


图 2　16 位 CPU 与闪速存储器的连接思路

表　在 16 位总线上的存取操作实例

总线操作	信号状态		数据有效/无效	
	A_0	$\overline{SBHE}$	$SD_0 \sim SD_7$	$SD_8 \sim SD_{15}$
偶数地址的字节(8 位)存取	“L”	“H”	有效	无效
奇数地址的字节(8 位)存取	“H”	“L”	无效	有效
偶数地址的字(16 位)存取	“L”	“L”	有效	有效

注：进行由奇数地址的字存取时，是分割奇数地址与偶数地址的字节存取（2 次存取）来运行的.

由此表我们知道，由于 A0 与低位字节的选择信号等价，所以地址偏移一个，将 A1～A17 赋予了闪速存储器。当地址一致、A0 为低电平时，低 8 位的闪速存储器的面有效；当 SBHE 为低电平时，高 8 位的闪速存储器的 CE 有效。

上述的例子为 16 位的 CPU。在 32 位以上 CPU 的情况下，没有 A0 及 A1，而是准备了以字节为单位的使能信号（大多命名为/BE0，/BE1，/BE2，/BE3 等名称）作为替代。

●2.4.4　读周期的概要

下面我们来看看闪速存储器读周期的时序。基本的存取方法的思路如图所示。

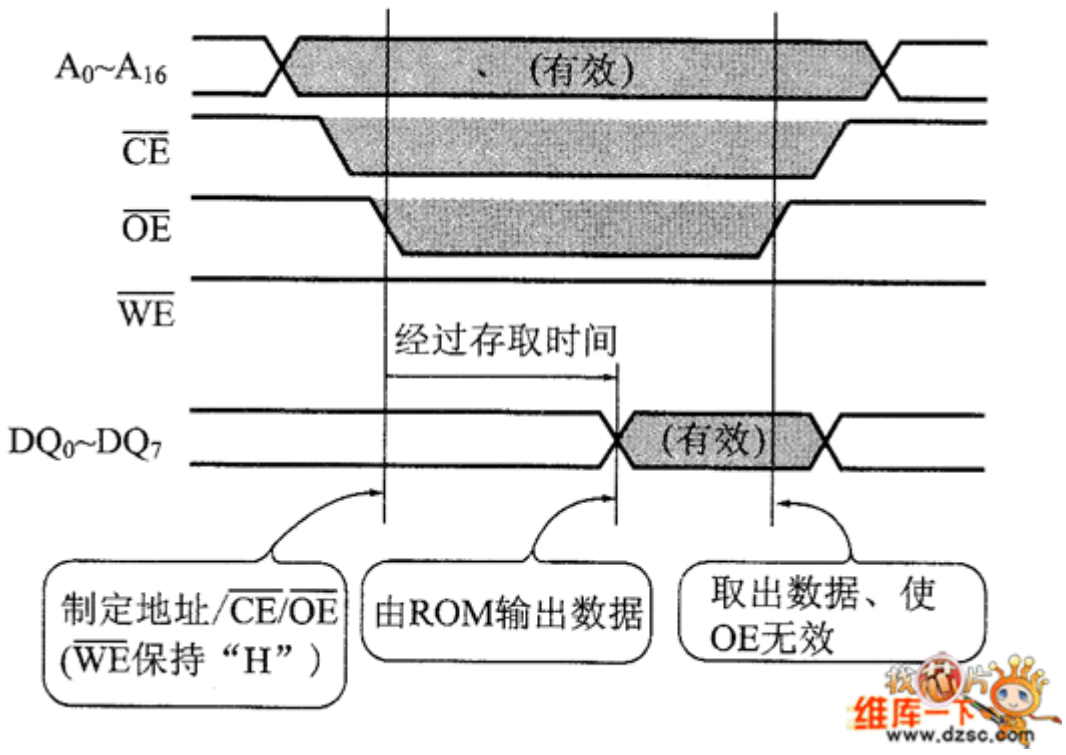


图　闪速存储器的读操作

将希望访问的地址提供给 A0～A16，一旦/CE、/OE 有效（低电平），则由闪速存储器开始读出数据（因为是读操作，所以/WE 保持高电平）。

数据在何时被确定是由地址．/CE．/OE 各自确定后的延迟时间（存取时间）规定的，是由最迟的时间确定数据的。

例如，Am29F010A-55 由地址及/CE 的存取时间为 55ns，由“/OE 的存取时间为 30ns。如果地址确定了的同时，/CE、/OE 同时有效，则 55ns 后将出现有效数据；在地址确定、CE 一直有效的稳定状态时，只要/OE 有效，则 30ns 后数据确定。

●2.4.5　写周期的概要

写周期的基本思路如图所示。因为这次是写的方向，所以/OE 保持高电平，由主机方面赋予数据（DQ0～DQ7）。

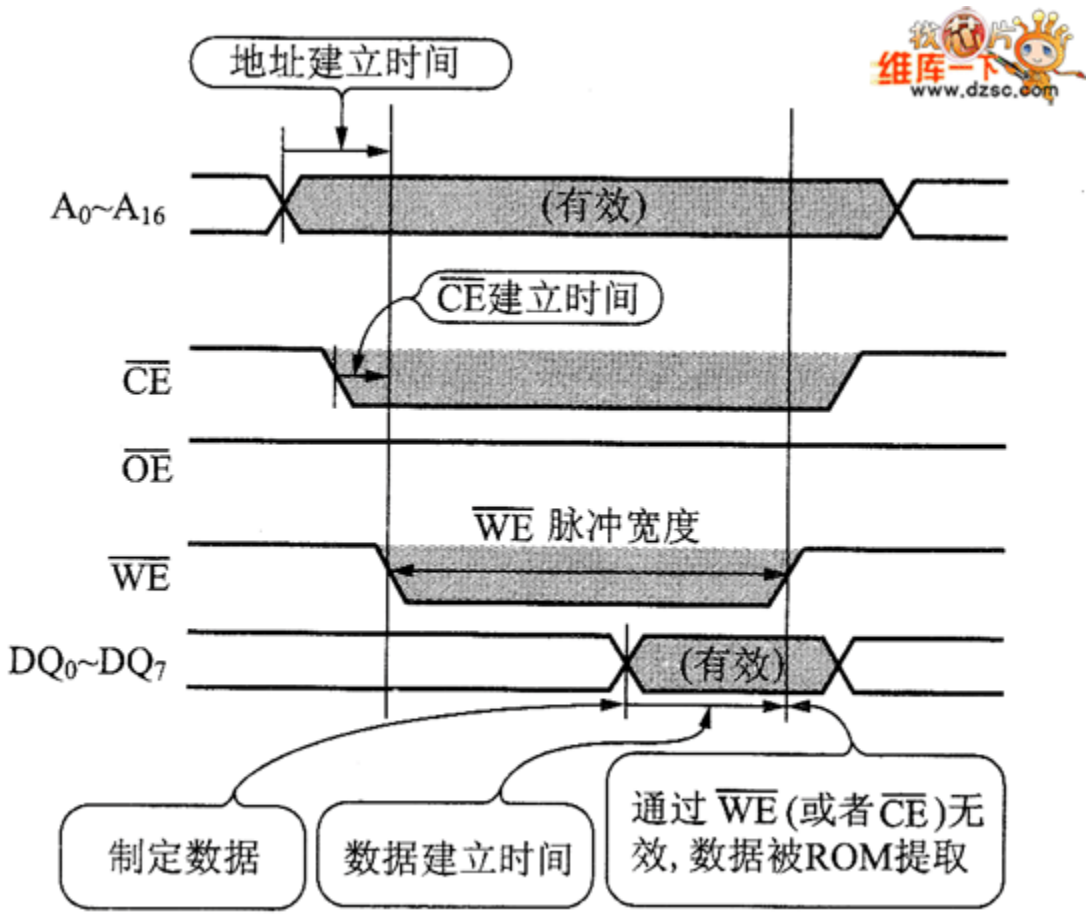


图　闪速存储器的写操作

当/WE 和/CE 双方都为低电平时，进行写操作的地址被提取到闪速存储器内部。而在/WE 和/CE 双方都变成低电平后，在任何一方上升为高电平阶段（由低电平到高电平的变化中），数据被提取到闪速存储器内部。通过/WE 进行的写操作称为/WE 写控制；通过/CE 进行的写操作称为/CE 写控制。一般情况下利用/WE 写控制的较多，所以图示的也是/WE 写控制的过程。

写操作时必须要注意的是地址及数据等各信号的建立时间。进行读操作时，只要等待地址、/CE、/OE 确定．从目的地址中就可以发出数据，因此对于确定的顺序就没什么必要给予注意。但是在写操作的情况下，如果不考虑地址、数据以及控制信号的时序，将会发生写人到错误的地址中或者没有正确接收数据的情况。

需要特别注意的是建立时间。如图所示，既需要在/WE 下降之前确定/CE 及地址，又需要在/WE 上升的一定时间之前确定数据。

详细的数值我们以后再进行说明。例如 Am29F010-55 的地址及/CE 的建立时间（分别为 tAs、tCS）最小都为 0。建立时间最小为 0 可能有些难以理解，解释为关键不能为负数可能比较容易理解。如果/CE 的建立时间为负，即/CE 是在以后成为低电平的，这样就变成了/CE 写控制。所以说，/WE 写控制的最小值为 0 是理所当然的。

数据在/WE 上升之前被确定，Am29F010-55 最小需要 20ns（tDS）。也就是说。在/WE 上升至少 20ns 之前，数据就必须被确定。

●2.4.6　读周期的时序

下面我们看一下数据手册中原有的、具体的读周期时序。图是Am29F010A的读操作的时序图，表是时序规定。与前所概述的内容相比较，在时序图当中，作为基点的电压存在两处，比概略图具有更多的时间规定。

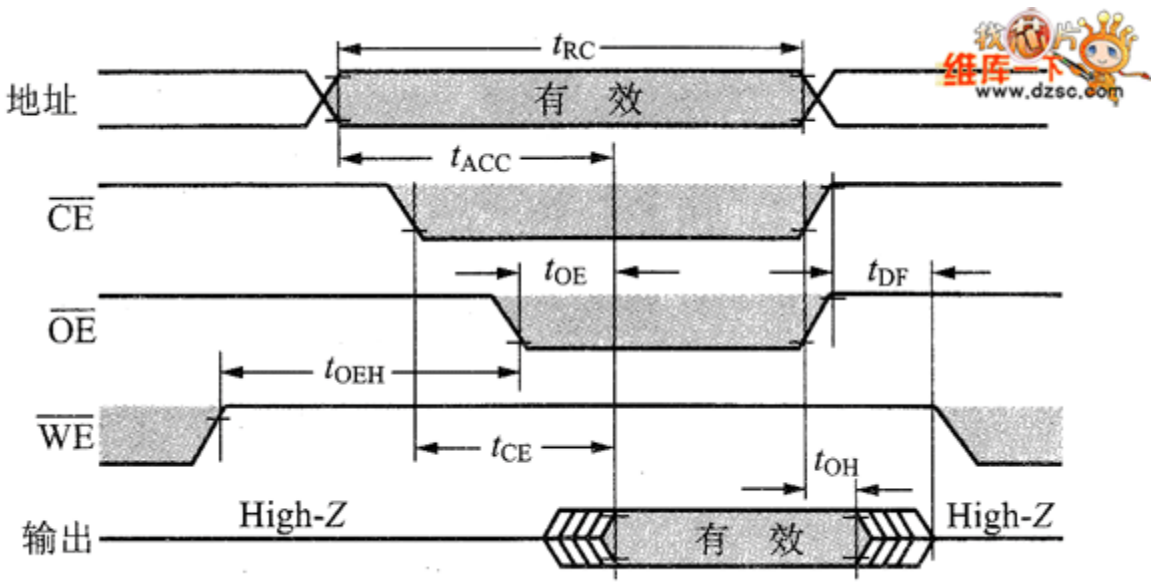


图 Am29F010A 的读操作时序

表 时序规定

符号		参数	条件		速度选项					单位
JEDEC	Std				−45	−55	−70	−90	−120	
t_{AVAV}	t_{RC}	Read Cycle Time		min	45	55	70	90	120	ns
t_{AVQV}	t_{ACC}	Addresee to Output Delay	$\overline{CE}=V_{IL}$ $\overline{OE}=V_{IL}$	max	45	55	70	90	120	ns
t_{ELQV}	t_{CE}	Chip Enable to Output Delay	$\overline{OE}=V_{IL}$	max	45	55	70	90	120	ns
t_{GLQV}	t_{OE}	Output Enable to Output Delay		max	25	30	30	35	50	ns
t_{EHQZ}	t_{DF}	Chip Enable to Output High Z		max	10	15	20	20	30	ns
t_{GHQZ}	t_{DF}	Output Enable to Output High Z		max	10	15	20	20	30	ns
	t_{OEH}	Output Enable Hold Time	Read	min	0					ns
			Toggle and Data Polling	min	10					ns
t_{AXQX}	t_{OH}	Output Hold Time From Addresses $\overline{CE}$ or $\overline{OE}$ Whichever Occurs First		min	0					ns

两处的电压基点表示如果低于下面的基点则为低电平，如果高于上面的基点则为高电平。处于两者之间的电压则根据器件的特性及其他，或者识别为高电平，或者识别为低电平。具体的电压将在后面的说明（DC 规定）中出现。

通常在波形下降阶段，处在下方的基点表示起始于确定了低电平那个时刻；而上方的基点则表示不能认为是高电平的那个时刻（根据器件，有时候也认为是可以识别为低电平的那个时刻）。一般情况下，时序规定都尽可能地严格。

现在，我们对上图中出现的 tRC、tOEH、tOH、tDF4 个参数加以说明。

▲ tRC（读周期时间）

这是确定（稳定）地址的时间。通过数据手册我们了解到，由于该时间与 tACC（从地址到数据输出的时间）是同一的，所以采用通常的方法一般不会出现问题。

▲ tOEH（输出使能保持时间）

这是在刚进行完写操作的情况下，从/WE 成为高电平开始到/OE 成为低电平之间的时间规定。由于在数据手册上最小为 0ns，所以可以解释为只要/WE 和/OE 不同时为低电平即可。

▲ tOH（输出保持时间）

在/OE 信号恢复为高电平后，只要进行严格的测定，就可测出 DQ 输出仍在持续进行。从/OE 不是低电平开始到 DQ 不再输出正确的数据这一段时间就是 tOH。由于 Am29F010A 的这个时间最小为 0，所以一旦 OE 变成高电平，则就不可再相信数据的准确性。

在存储器存取时间较长的情况下，利用该参数，在读取数据之前，先将/OE 恢复为高电平来争取时间，这样的设计事例曾经出现过。但实践证明放弃这样的做法才是明智的。

▲ tDF（从/CE、/OE 到输出高阻抗）

/CE 或者/OE 成为高电平之后．闪速存储器的 DQ 输出完全成为高阻抗所需要的时间就是 tDF。因为这一时刻中闪速存储器可能会持续输出某些数据,所以，如果此刻其他的驱动器以及缓冲器被启动，则可能会发生数据冲突。在缓冲器的方向控制方面，如果是在/OE 与缓冲器的方向控制等同时进行的电路中，则需要对缓冲器的切换时间是否过早进行判断。

●2.4.7　写周期的时序

写周期的详细情况将在后面进行描述，这里要说明的是，写周期在闪速存储器中利用/WE 的存取操作不是类似 RAM 那样的向指定地址的直接写人操作，而是通过指令对闪速存储器进行操作。与 NAND 闪速存储器相同，通过一连串的指令序列，可以进行编程（数据写人）和芯片擦除操作。

为此，时序图也以包括用于编程及擦除操作时间的形式被记录。图 1 为编程操作，图 2 为擦除操作。

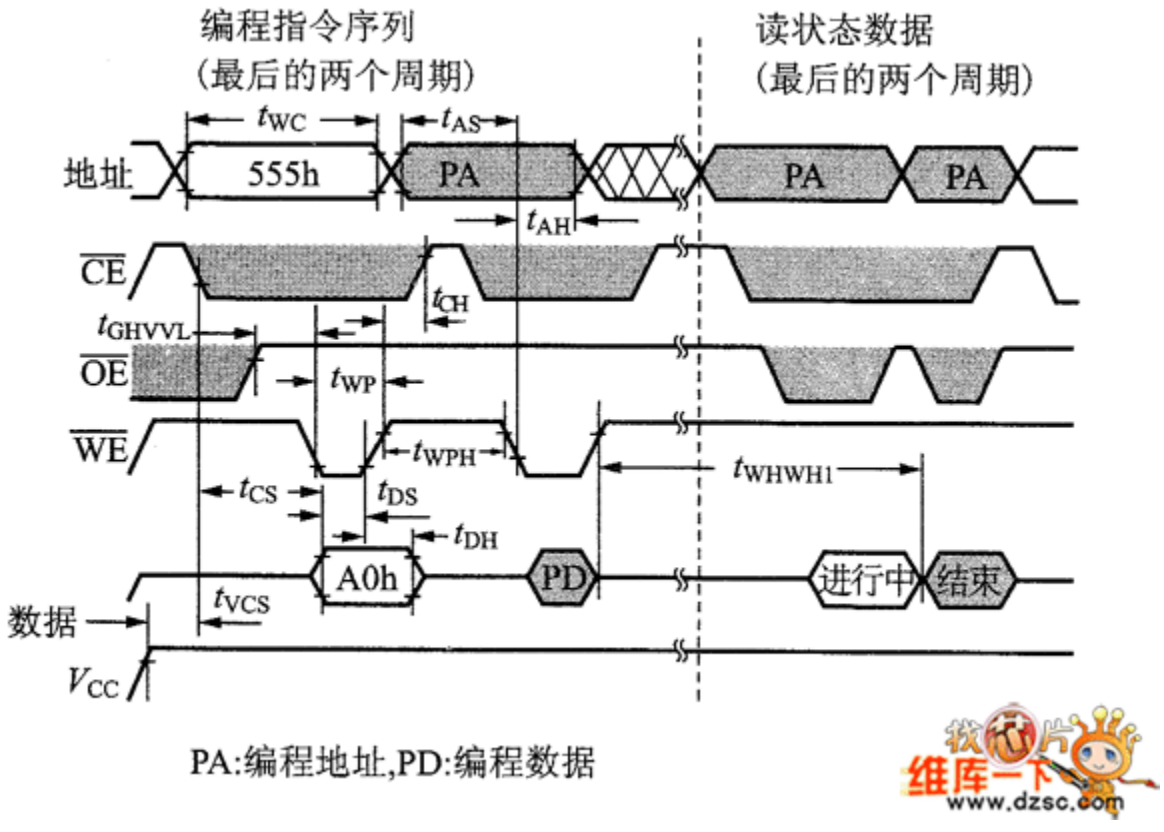


图 1　编程操作

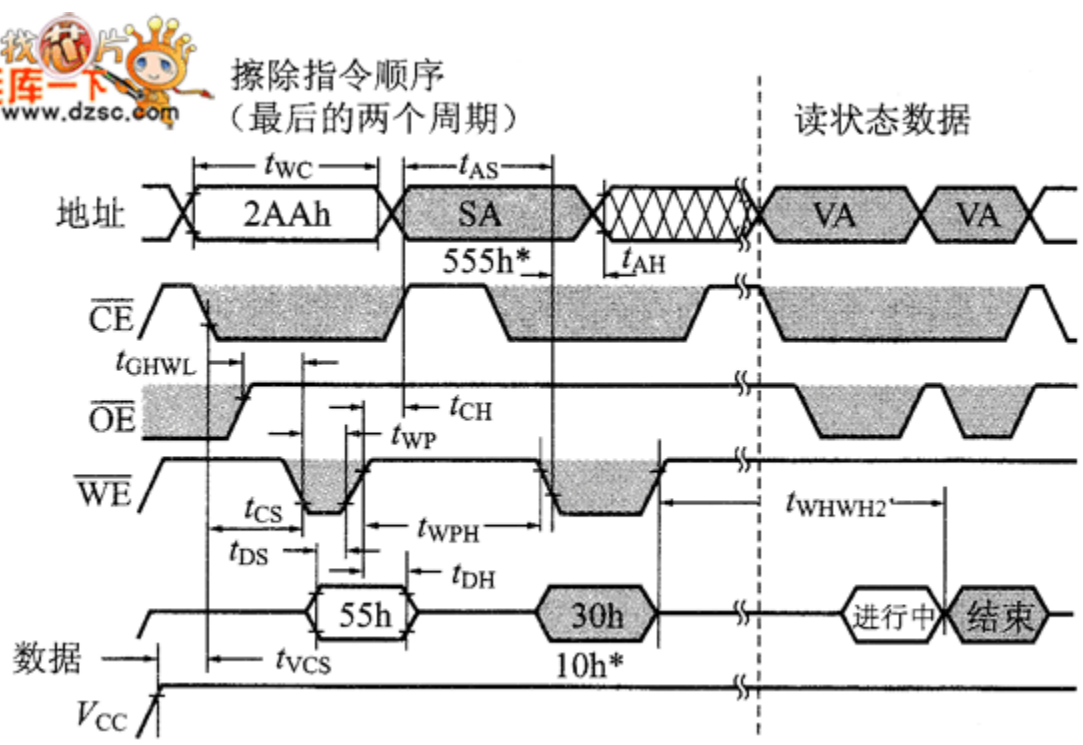


图 2　擦除操作

另外，在图中，如编程操作，是通过向 555h 写人 AOh、然后给予 PA（编程地址）和 PD（编程数据）的写人方法。但实际上，在访问 555h 之前，就进行几次写入操作，闪速存储器将此识别为编程操作。关于这种顺序将在以后详细说明。如果将该过程全部写下来，将会占据非常大的篇幅，困此省略大部分，只对最后部分进行说明。擦除操作也以相同的形式进行说明。

由于编程及擦除的内部操作相当花费时间，因此，需要从闪速存储器中取出其状态等，以便 CPU 检验操作是否结束。图的右半部分描述的就是这个验证操作。

无论怎样的操作时序，作为写操作基本上都是相同的，我们在说明编程操作之后，将简单地描述擦除时序。

■ 编程操作

首先，我们先看如图 1 所示的编程操作。因为时序规定的线太多，可能看起来非常复杂，但只要按照顺序追踪下去，就不会觉得困难。基本的时序思路刚才已经概括说明。

▲ t_{VCS}（Vcc建立时间）

由于闪速存储器在内部具有对来自主机的指令进行解释的定序程序，因此电源一旦达到额定电压值,则立即不能进行存取。电源达到额定电压值之后到发出最初的指令之前,必须获得的时间就是tvcs。Am29F010A的这个时间为 50μs。

通常，作为CPU的外部记忆加以使用的时候，一般都具备加电重启电路。电源接通后，解除CPU的复位，CPU到开始最初的存取一般都需要相当长的时间。因此，出现问题的时候较少，但在使用闪速存储器写人器的情况下，都在电源引脚上准备半导体开关等，插拔插座时关闭电源，在开始存取时，进行接通电源的操作。为此,在电源控制板上接通电源后,必须经过达到额定电压值所需要的时间＋50μs以后，才能访问闪速存储器，这是需要注意的地方。

▲ tCS（/CE 建立时间）

/WE 写控制的情况下，/CE 需要提前于/WE 有效，这个提前的时间就是 tCS。因为时序规定最小为 0ns，所以也就意味着它不能为负值，也就是/CE 不可能在/WE 之后有效。如果/CE 在/WE 之后有效，就成为了/CE 写控制。

▲　tWC（写周期时间）

这是在写操作中确定地址的时间。Am29F010A-55 需要保持 55ns 以上的时间。

▲　tGHWL（读恢复时间）

这是在写周期的前面为读周期的情况下，/OE 必须在/WE 多少时间之前无效的规定。

▲　twp（写低电平脉冲宽度）

规定了/WE 有效的期间。Am29F010A-55 最小为 30ns。

▲　tDS（数据建立时间）

在向闪速存储器中进行写操作时，虽然数据是在/WE 的上升阶段被提取。但是对于正确数据的锁定，需要在/WE 上升之前确定数据（DQ0～DQ7），这个时间就是 tDS。Am29F010-55 为 20ns，所以需要在 20ns 以上的时间之前确定数据。

▲　tDH（数据保持时间）

这是在/WE 上升后必须保持数据被确定的时间。以前根据器件内部的情况，某种程度上需要这个时间。但在最近的器件中．这个时间为 0。也就是说，到/WE 上升，数据大多能被保持。Am29F010A 的 tDH 就为 0ns。

▲　tAS（地址建立时间）

将写操作时的地址提取到闪速存储器是在/WE 的下降时刻，所以在达到该下降时刻之前必须确定地址。tAS 表示在/WE 下降之前的多少时间前确定地址需要。

Am29F010A 的 tAS 为 0，因此，最慢也要在/WE 的下降时刻确定地址。

▲　tAH（地址保持时间）

虽然写操作时的地址是在/WE 下降时锁存的，但这也与 tDH 相同，需要规定持续保持的时间。Am29F010A－55 需要的 tAH 时间为 45ns。

▲　tWPH（写高电平脉冲宽度）

这是/WE 无效后到下一次/WE 有效的时间。Am29F010A 最少需要 20ns 的时间，所以连续存取时必须获取超过 20ns 的间隔时间。

▲　tWHWH1（字节编程操作时间）

在进行编程操作时，虽然用最后的写操作指定写人地址及数据，但闪速存储器并不是在完成写操作的同时，完成向闪速存储器单元的写入。一看芯片内部的框图就可知道，在闪速存储器内部具有状态控制器及写入电压生成电路，编程指令只不过是状态机的启动脉冲向单元发出开始写人操作的指示而已。写人电路工作直到实际完成向存储器单元的写人操作，这一段时间就是 tWHWH1。Am29F010A 的 tWHWH1 时间为 7μs，但因为这毕竟是典型值，所以多少存在一些变化。判断是否真正完成写入操作不是通过时间管理，而是必须根据数据读的状态验证来进行。

在 tWHWH1 期间，如果 CPU 针对闪速存储器进行数据读操作，则在 DQ 中将出现内部的操作状态，这是理所当然的事情。只要完成读取就成为一般的读周期，数据就会被读出。图中表示了最后切换的时序。

关于具体的状态内容等，我们将在后面进行说明。

■　擦除操作

▲　tWHWH2

擦除操作与编程操作一样，由主机发出一连串的指令序列，器件从识别出擦除指令的时刻起，开始内部的操作。

写操作方面的时序与编程操作完全相同，但与编程操作相比，擦除操作却需要较长的时间。该时间就是 tWHWH2。Am29F010A 为 1s，该值也是典型值，实际上与编程操作相同，也将查询状态进行验证。

●2.4.8　闪速存储器指令

通过地址与数据特定组合的若干次写人序列，向闪速存储器发出指令。利用这样的序列，防止由于编程错误及开通电源的暂时不稳定等因素所引发偶然擦除及写人操作。

Am29F010A 的指令定义如表所示。例如，当编程指令（向闪速存储器特定地址写入数据）下达时，如下所述：

①向 555h 地址写人 AAh；

②向 2AAh 地址写人 55h；

③向 555h 地址写入 A0h；

④向希望写人地址（PA）处写人希望写入的数据（PD）。

表 Am29F010A 的指令定义

指令顺序		周期	总线周期											
			第1次		第2次		第3次		第4次		第5次		第6次	
			Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Read		1	RA	RD										
Reset		1	XXXX	F0										
Reset		3	555	AA	2AA	55	555	F0						
Autoselect	Manufacturer ID	4	555	AA	2AA	55	555	90	X00	01				
	Device ID	4	555	AA	2AA	55	555	90	X01	20				
	Sector Protect Verify	4	555	AA	2AA	55	555	90	(SA) X02	00				
										01				
Program		4	555	AA	2AA	55	555	A0	PA	PD				
Chip Erase		6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10
Sector Erase		6	555	AA	2AA	55	555	80	555	AA	2AA	55	SA	30
Erase Suspend		1	XXX	B0										
Erase Resume		1	XXX	30										

RA：读地址；PA：编程地址；SA：扇区地址；RD：读数据；PD：编程数据

通过上述 4 次写入序列完成写人操作。最后的写操作完成后，根据读取状态来判断内部操作是否结束。

第三章

EEPROM 的结构与使用方法

3.1　EEPROM 的概要

EEPROM 从大方面分为并行 EEPROM 和串行 EEPROM。并行 EEPROM 通过与闪速存储器相同的引脚配置可以并行输人输出数据；而串行 EEPROM 通过 8 引脚等较小的封装一位一位进行数据传输。表整理并比较了一般闪速存储器、并行 EEPROM 以及串行 EEPROM 的特点。

表 闪速存储器与 EEPROM 的比较

	闪速存储器(并行)	并行 EEPROM	串行 EEPROM
容　量	大	中	小
封　装	大～中	大～中	小
擦除单位	整个芯片或者块	以 1 字为单位	以 1 字为单位
编程单位	1 字(只有“1”→“0”的方向)	以 1 字为单位	以 1 字为单位
编程方法	需要指令序列	只可以进行写存取操作	发出指令
读 速 度	快	快	慢
操　作	异步	异步	时钟同步
控制信号	地址、数据、片选、读、写		I^2C:时钟、数据
			Microwire:时钟、片选、数据
			SPI:时钟、数据 IN/OUT、片选、保持

并行 EEPROM 与闪速存储器的不同只是在以 1 字节为单位进行替换这一点上，其他基本可以进行相同的操作处理。虽然可以连接一般的微型处理器并应用于编程中，但由于前面所叙述的不能增加容量的原因，主要在必须以 1 字节为单位进行替换的情况下应用。

而串行 EEPROM，包括地址及指令等在内需要一位一位地进行传输，故数据的传输速度不能够提高。但因为利用较少的引脚就可完成存取，因此与单芯片微型计算机之类的、只需要输入输出引脚的处理器就可以实现简单地连接。依靠这样的特性，它被专门应用于小规模系统的周围器件的信息设置。除

此之外，还用于存放初始化 FPGA 等的数据。目前，一般的串行 EEPROM 容量最大也就是 512K 位，而小容量 EEPROM 目前存在的现有产品有不到 1K 的，这是在其他的存储器器件中几乎没有的产品。

3.2　串行 EEPROM

串行EEPROM的接口一般包括I^2C（Inter IC Communication）总线、Microwire总线以及SPI（Serial Peripheral Interface）总线三种。无论哪种总线都使用时钟信号和数据 / 控制线，时钟信号由主机进行控制。用于控制的信号线中，I^2C为 2 根、Microwire为 3 根、SPI为 4 根。

还需要考虑的是，在这些总线中，无论哪种总线的同一条总线上连接多个串行EEPROM器件的情况。I^2C总线是把由主机传递过来的器件地址与目标对象所设定的地址进行比较，如果一致就是目标对象。

Microwire和SPI总线除了具有用于传输的信号外，还具有片选信号，通过该信号的有效与否，主机指定哪个器件作为目标对象。

3.3　Microwire 总线对应的存储器——M93Cx6

● 3.3.1　M93Cx6 的引脚配置

M93Cx6 系列是 8 引脚的封装，其引脚配置如图 1 所示，包括 3 根 Microwire 总线信号及用于数据宽度（以 16 位为单位还是以 8 位为单位）选择的 ORG 输入和片选输人。

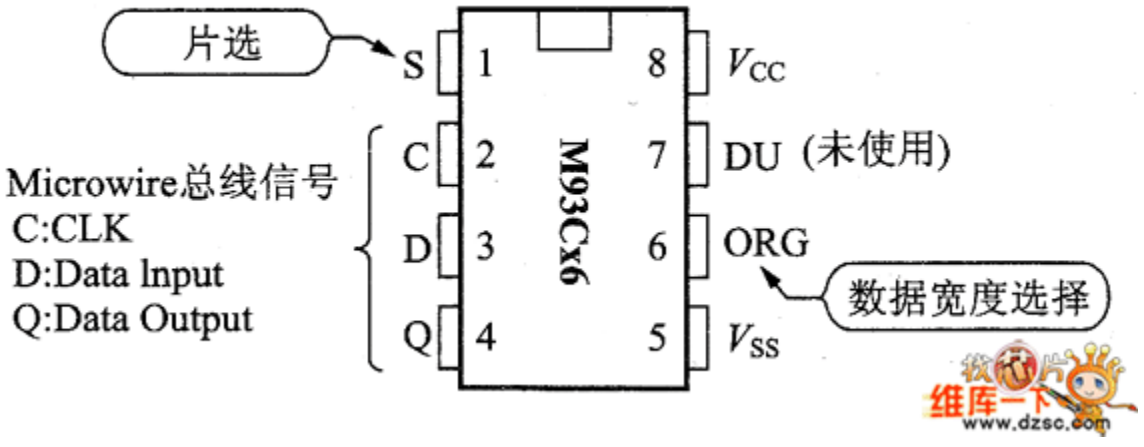


图 1　M93Cx6 的引脚配置

由于 ORG 不是普通动态变化的，所以与主机相连的是 C、D、Q 和 S 这 4 根线。图 2 显示了主机与多个 Microwire 存储器的连接实例。

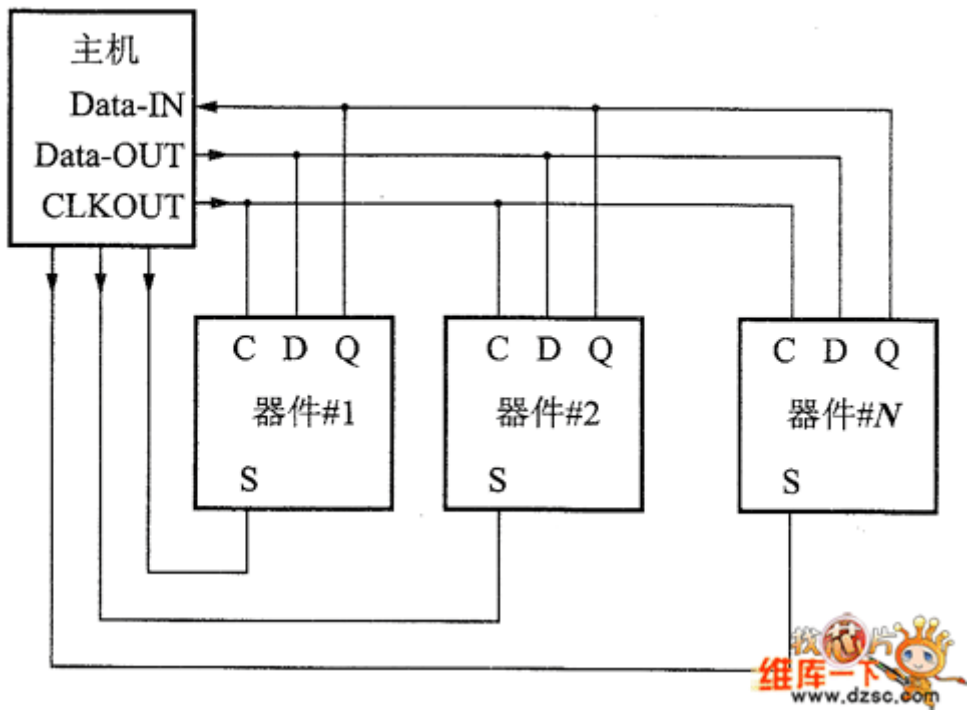


图 2　Microwire 存储器与主机的连接

下面我们简述 Microwire 总线的各个信号。

▲　S（Chip Select Input，片选输入）

这是片选输人。由于 Microwire 总线也与其他的串行总线一样，是可以与多个存储器器件连接的，因此，通过该引脚主机可以通知要访问那个器件。该引脚为高电平时，M93Cx6 为使能状态，可以与主机进行传输。

▲　D（Data Input，数据输入）

在 Microwire 总线的情况下，存储器的数据输人引脚与输出引脚是分离的。该引脚是主机向目标（存储器）方向的数据输人引脚，存储器在时钟（C）上升沿将数据提取到内部。

▲ Q（Data Output：数据输出）

这是数据输出引脚。如果时钟（C）到达上升沿，则将新的数据存放于 Q 中。

▲ C（Clock）

这是时钟信号。Microwire总线的操作是以该时钟信号为基准进行的。在写操作的方向时，主机存放于D的数据在时钟的上升沿被存储器提取；而在进行读操作时，存储器在时钟上升沿来到时，将欲在下一次提取到主机的数据存放于Q中。该操作如图 3 所示，M93Cx6 的最高时钟频率为1MHz。

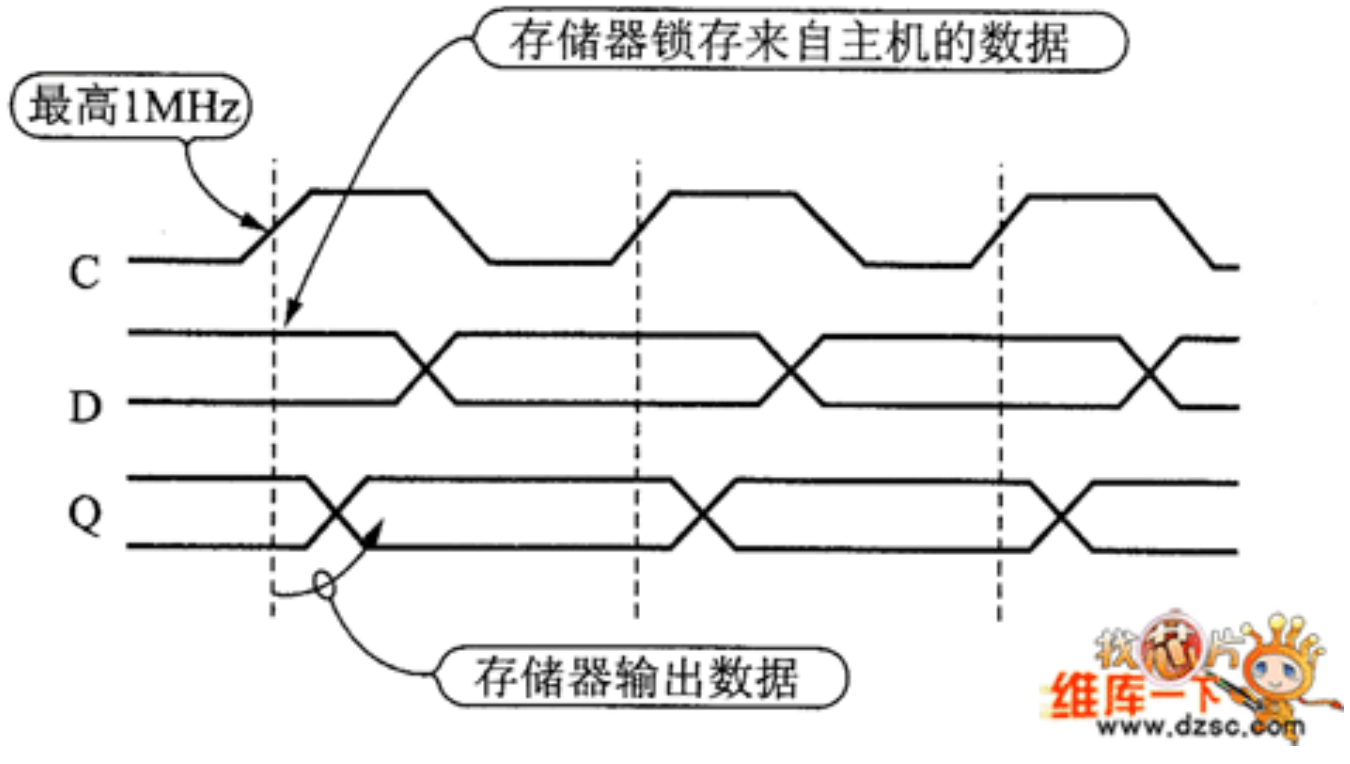


图 3　时钟与 D／Q 的关系

▲ ORG（Organization）

其他的串行EEPROM也一样，即使是“串行”也还是输入输出总线，读取以及替换通常是以 8 位乃至 16 位为字单位进行的。如果是 8 位宽度的，则读操作时一旦赋予地址就要读出 8 位的数据，替换时也必须传递 8 位的数据。

M93Cx6 系列的字长可以通过 ORG 引脚改变，ORG 如果是高电平而且是打开的，则为 16 位宽度，必须从所指定的地址以 16 位为单位输入输出数据。反之，如果 ORG 为低电平，则数据宽度为 8 位，将以 8 位为单位输人输出数据。

● 3.3.2　Microwire 总线的存取操作

如前所述，Microwire总线的存取操作是以时钟的上升沿为基准进行的。总线的大概工作情况如图 1 所示，时钟上升时，如果S输入为高电平，则器件处于选择状态。D输人一旦成为高电平，则识别为起始位，开始进行工作。在起始位之后的 2 位为操作代码，然后是地址位。地址位的位数不固定，根据存储器的容量而发生变化。

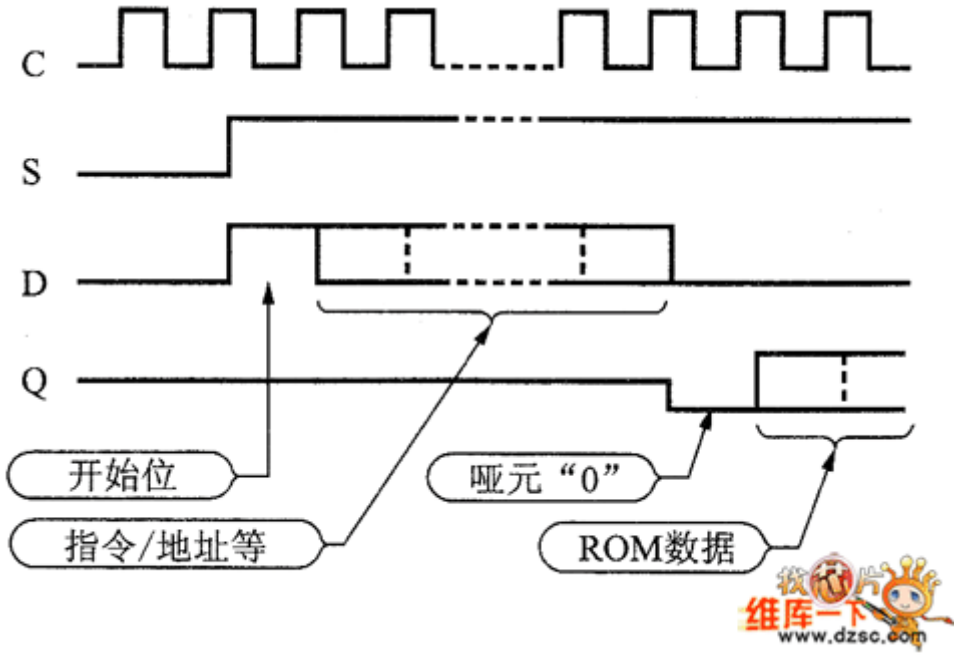


图 1　Microwire 总线的存取操作示例

若干个指令只能由这 2 位来指定，对于不能通过这 2 位完全表示的部分，将挪用紧随其后的地址位的一部分来表示指令。在这些指令中，虽然地址字段的值实际上只能使用 2 位，但需要传输与READ / WRITE指令相同的位数。

M93Cx6 具有 7 种指令，例如M93C06 和M93C46的指令格式如表 2 所示。作为实例，M93C06 的读操作如图 2 所示，写操作如图 3 所示。另外EWEN指令的发布实例如图 4 所示。

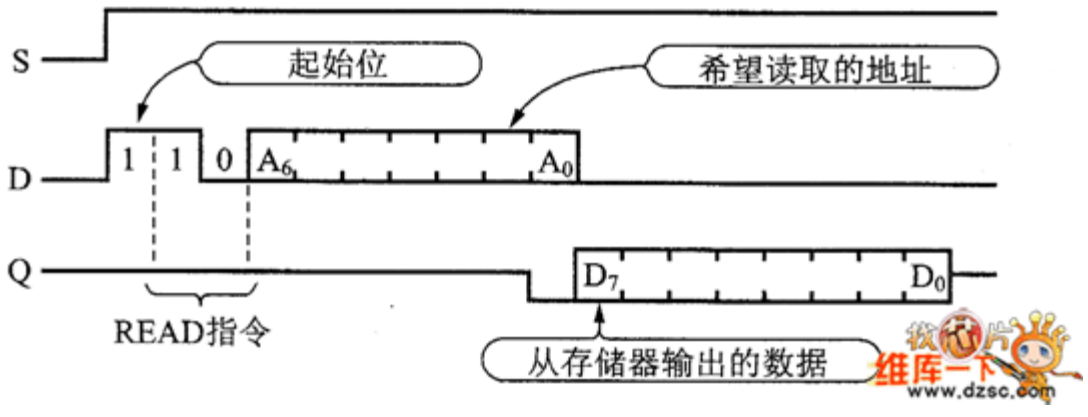


图 2　M93C06 的读操作

表　M93C06 的指令设置

指令名称	内　容	指令/数据					
		起始位	操作代码	ORG=L(8 位宽度时)		ORG=H(16 位宽度时)	
				地址字段值	数据	地址字段值	数据
READ	来自存储器的数据读	“1”	“10”	$A_6 \sim A_0$	$Q_7 \sim Q_0$	$A_5 \sim A_0$	$Q_{15} \sim Q_0$
WRITE	对存储器的数据写	“1”	“01”	$A_6 \sim A_0$	$D_7 \sim D_0$	$A_5 \sim A_0$	$D_{15} \sim D_0$
EWEN	擦除/写使能	“1”	“00”	“11XXXXXX”	- - - -	“11XXXXXX”	- - - -
EWDS	擦除/写禁止	“1”	“00”	“00XXXXXX”	- - - -	“00XXXXXX”	- - - -
ERASE	特定地址的擦除	“1”	“11”	$A_6 \sim A_0$	- - - -	$A_6 \sim A_0$	- - - -
ERAL	芯片整体的擦除	“1”	“00”	“10XXXXXX”	- - - -	“10XXXXXX”	- - - -
WRAL	向整个芯片写入同一数据	“1”	“00”	“01XXXXXX”	$D_7 \sim D_0$	“01XXXXXX”	$D_{15} \sim D_0$

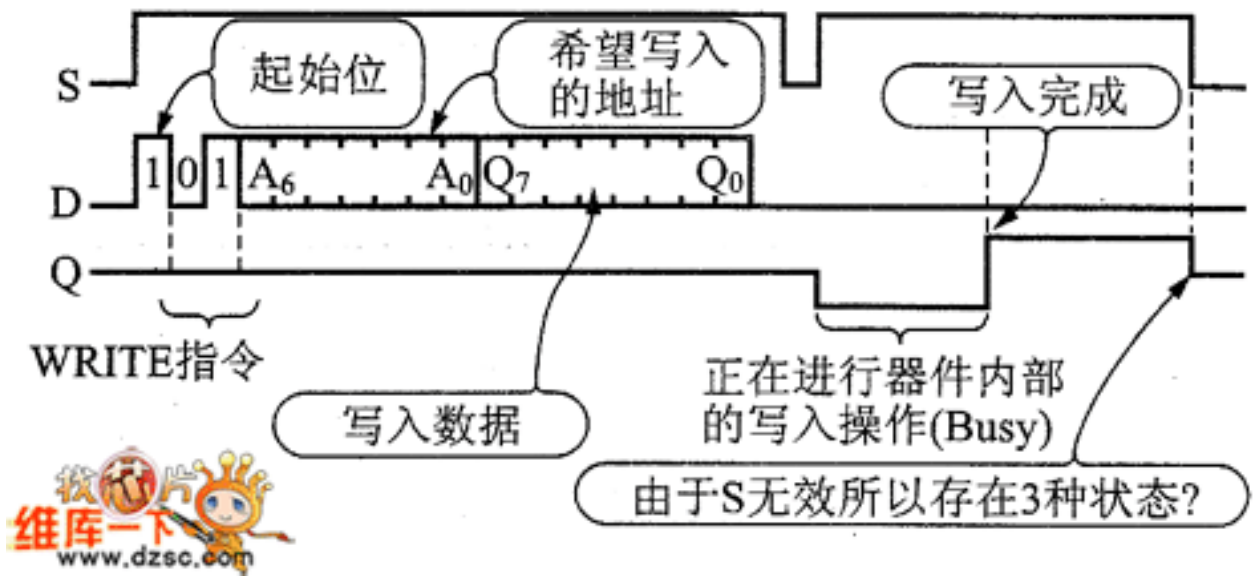


图 3　M93C06 的写操作

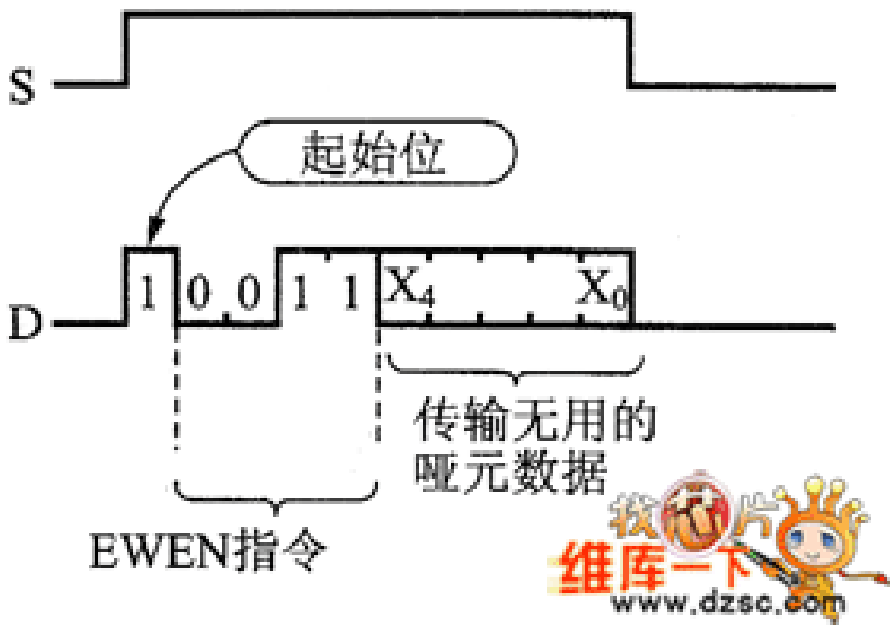


图 4　EWEN 指令的发布实例

▲ READ

数据读指令是在操作代码“10”之后通过指定地址来进行的。一旦地址位发送完毕，存储器于时钟上升沿同步从Q按顺序输出数据。由于开始之后将输出 1 位哑元“0”，所以需要丢弃该位。地址 / 数据起始于MSB这一点是与E1A－232 等串行传输不同的地方。

EEPROM内部存在地址寄存器，如果读出 1 字节，则自动增加地址计数。读取指令之后，S保持高电平，如果继续给予时钟，则自动出现下一地址的数据，但此时不插人哑元“0”数据位。

▲ WRITE

写操作也是在操作代码“10”之后指定地址，进而利用 D 发送写人数据。一旦这些信息传送完毕，必将在下次时钟的上升沿将 S（片选）设置为低电平，暂时中断选择。器件如果知道 S 为低电平，则开始内部的写人操作；如果 S 为高电平，则指令被忽略。

与闪速存储器等相同，即使接受指令和数据，内部的写入操作也不是立即就能完成的。这期间，如果 S 设置为高电平．则选择器件；如果内部尚处于写人操作，则 Q 输出为低电平；如果写人操作已完成，则 Q 输出为高电平。

闪速存储器在编程操作中进行的操作只是在位由“1”变为“0”的方向上，由“0”变为“1”的方向上只能进行擦除操作。而 EEPROM 如果发出 WRITE 指令，则内部可自动进行擦除操作。因此，只要发出 WRITE 指令就可以进行替换。

▲ EWEN（ERASE / WRITE Enable，擦除 / 写使自包）

当操作代码“00”紧接着地址位前 2 位为“11”时，就是EWEN指令。M93Cx6 已经具有禁止写入及擦除操作的功能，它是可以防止不经意的替换操作的。由于M93Cx6 在提供电源后写人及擦除操作是被禁止的，因此在希望进行替换操作时，需要发出该指令，允许进行替换操作。

▲ EWDS（ERASE / WRITE Disable，擦除 / 写禁止）

当操作代码“00”紧接着地址位的前两位为“00”时，就是 EWDS 指令，发出该指令后，存储器就不能够接受替换及擦除指令。

为了防止不经意的存储器的替换和擦除操作，最好在发出 WRITE 指令后再发出该指令。

▲　ERASE

当操作代码为“11”时，由随后地址位所指定的地址的内容将被擦除。虽然与通过WRITE写入所有“1”的数据的结果相同，但擦除指令由于不需要传输数据的时间，因而较简单。在后面将要叙述的I^2C总线对应存储器的情况下，没有擦除指令，通过写人FFh就能达到擦除的功能。

▲　ERAL（Erase All Memory）

当操作代码“00”紧接着地址位的前两位为“10”时，就是ERAL指令。它将擦除整个芯片的内容。与闪速存储器的ChipErase指令相同。

▲　WRAL（Write All Memory with same Data）

利用指定的数据写满整个芯片，ERAL 指令可以指定数据的。

3.4　SPI 总线存储器——M95256

● 3.4.1　M95256 的引脚配置

M95256也仍然是 8 引脚的封装，虽然也存在TSSOP的 14 引脚的封装，但其中 6 个引脚未加以利用。引脚配置如图所示。

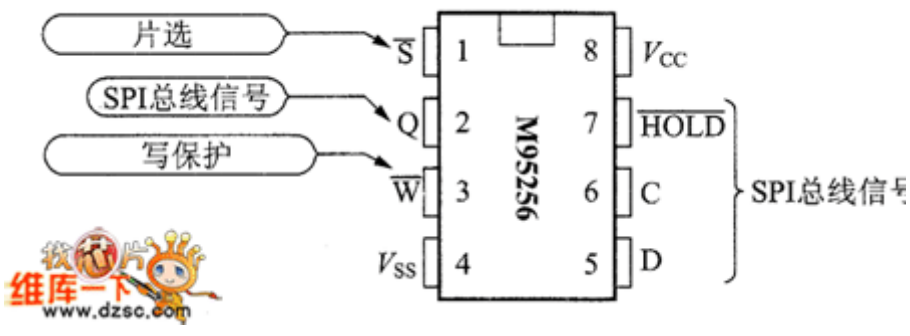


图　SPI总线对应的存储器M95256的引脚配置

与Microwlre相同，都是通过S（Chip Select，片选）输人选择器件，D与Q也被分离，而且也是与时钟（C）同步进行指令及地址、数据的传输的。但与Microwire相比较，改善了传输结构，操作处理起来更加简便。

与Microwlre相比，较大的改变如下所述。

▲　片选引脚的逻辑

在Microwlre总线中，片选为高电平激活，而在SPI总线中变更为一般的低电平激活。

▲　/HOLD（保持）引脚的添加

在Microwlre总线中，存储器器件必须能够追随主机的传输时钟速度；而在SPI总线中，由于增加了/HOLD引脚，这样对于主机就可以使之处于等待状态。而最高时钟频率由M93Cx6 的1MHz提高到5MHz，也可以说是增加/HOLD引脚的理由吧。

▲　指令代码的 8 位化

相当于SPI总线指令代码的Microwire总线的操作代码只有 2 位，利用这 2 位代码表示 4 个命令显然不够，因此采用了所谓一直到地址位为止都作为命令代码的变通方法。而SPI的指令代码为 8 位．因而即使不采用上述的变通办法也足够表示命令。

▲　状态寄存器的添加

在Microwire总线的情况下，称为存储器端的状态只是替换及擦除操作时Q的状态，相当于表示Ready / Busy的程度。而SPI在器件内部，安装了状态寄存器，可以统一处理芯片的状态以及利用软件实现其保护功能等。

▲　指令的整理

在SPI总线的EEPROM中，删除了认为使用频率较低的擦除方面的指令以及WRAL指令等，取而代之的是添加了状态寄存器的读 / 写指令。

▲　W（Write Protect，写保护）引脚的添加

SPI不但具有通过软件实现的写保护功能，而且还添加了通过硬件实现的写保护功能。

● 3.4.2　SPI 总线对应的存储器的操作

SPI总线的工作本身与Microwire总线基本上没有太大的变化。图 1 表示M95256的存储器读周期。读命令的指令代码为“00000011”，但传送完该代码后，16 位的地址由高位开始按顺序被传送，一旦接收到该地址位，就会从存储器将数据输出到Q。M95256 的容量为256K位，也就是 32KB，因而A15 被忽略，但作为地址必须传输 16 位。另外还需要注意一点，SPI总线与Micorwire总线相同，地址与数据都是由高位开始传输的。

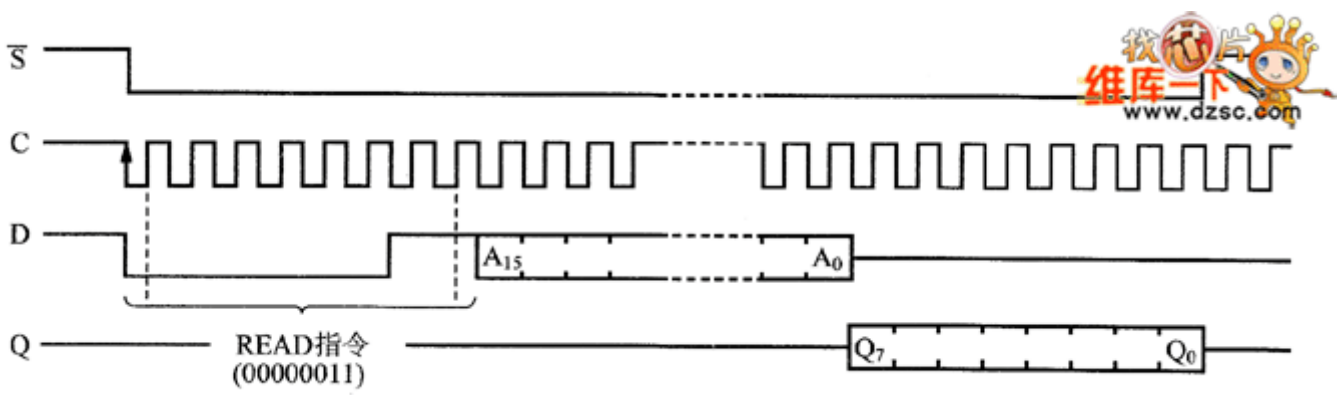


图 1　M95256 的读周期

写操作的方向如图 2 所示，在 8 位的指令代码后，紧接着是 16 位的地址及 8 位的数据被连续传输，之后开始EEPROM内部的替换操作。

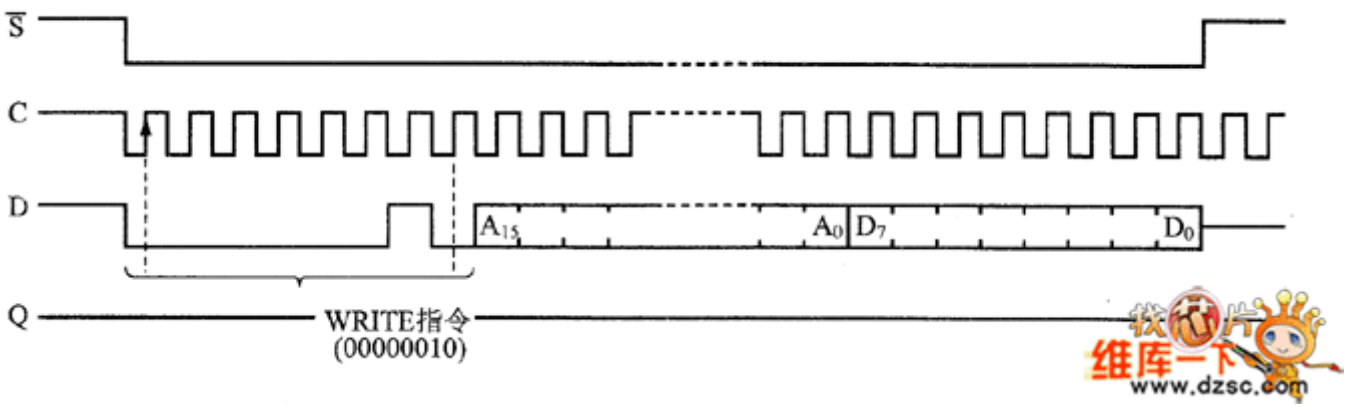


图 2　M95256 的写周期

图中所表示的只是 1 字节的替换操作。M95256 内部是以 64 字节为 1 页的单位进行分割的，可以汇总 1 页内的数据进行替换。而且M95256 内部也具有地址计数器，写操作之后可自动进位。因此，通过保持/S有效、连续传送多字节的数据，可以一次最多进行 64 字节（1 页大小）的数据替换。

状态寄存器的读 / 写操作除了不必赋予地址外，其他与存储器的读 / 写操作相同。当状态寄存器连续进行读操作时，将一个个读出新的状态。发出WRITE指令后，在不需要访问其他SPI存储器的情况下，通过保持S有效连续读取状态寄存器，就可了解其目前的状态，而不必每次都发出读取状态寄存器的指令，可以说这是相当方便的吧。

● 3.4.3　指令设置

M95256 所拥有的指令如表所示，共有 6 种。与对存储器单元的读 / 写操作相关的只有READ和WRITE。我们已经了解到Microwire总线所拥有的擦除等指令已被删除。

表　M95265 的指令设置

指令名	内　容	指令代码
WREN	写使能锁的设置	“00000110”
WRDI	写使能锁的复位	“00000100”
RDSR	状态寄存器的读取	“00000101”
WRSR	状态寄存器的写入	“00000001”
READ	数据读	“00000011”
WRITE	数据写	“00000010”

RDSR（Read Status Register，读状态寄存器）与 WRSR（WriteStatus Register，写状态寄存器）是分别用于状态寄存器的读 / 写操作的；WREN（Set Write Enable Latch，设置写使能锁）与 WRDI（Reset Write Enable Latch，复位写使能锁）分别用于器件写入的禁止与许可。因此，这些指令所设定的状态将反映于状态寄存器的 WEL（Write Enable Latch，写使能锁）位上。

● 3.4.4　状态寄存器

M95256 的状态寄存器的位配置如图 1 所示。其中，WEL 和 WIP 是只读的不能替换，其他的位可以替换。各个比特位的简单描述如下：

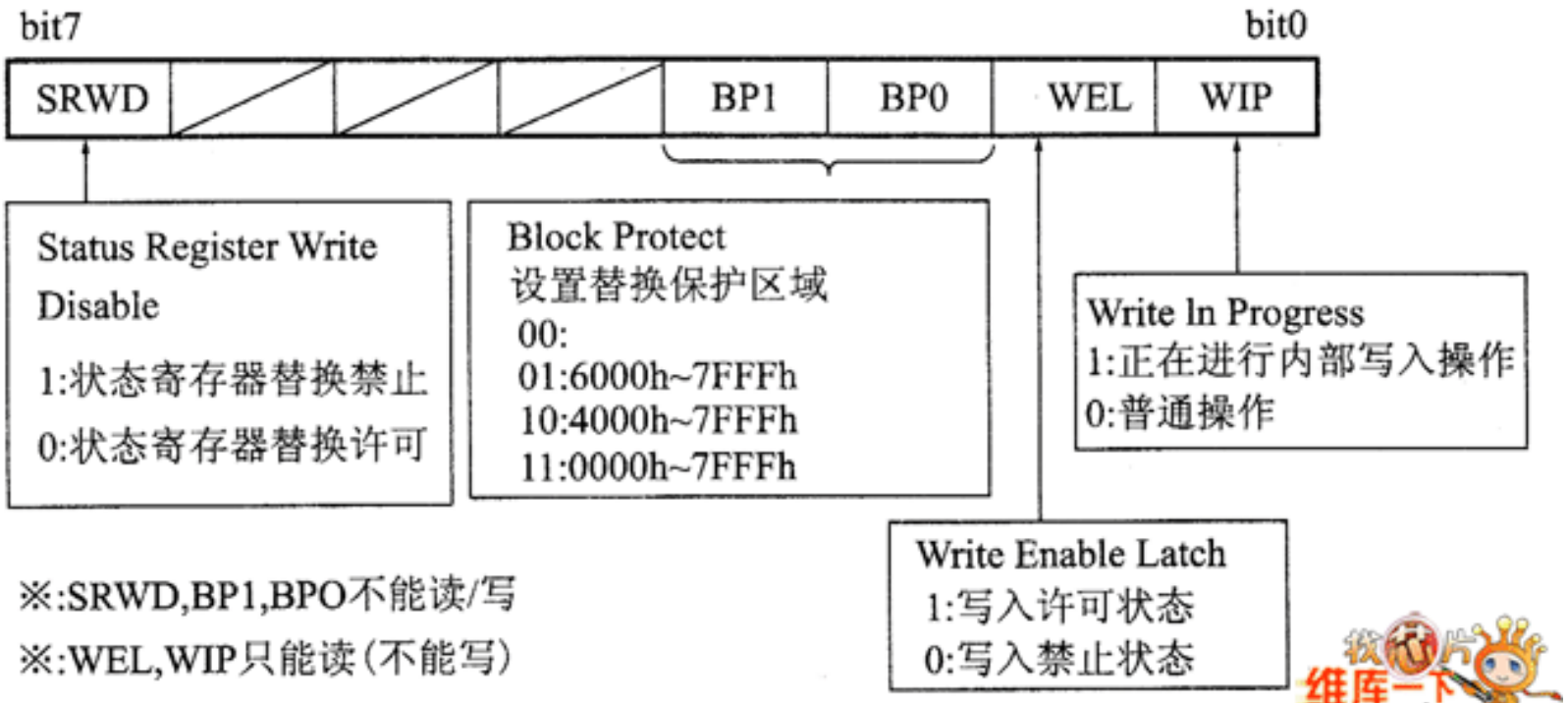


图　M95265 的状态寄存器

▲　SRWD（Status Register Write Disable，状态寄存器写使能）

该比特位进行状态寄存器的更新控制。如果该比特位为“0”，进而设置 WEL 位（通过 WREN 命令），就能够进行状态寄存器的替换。

该比特位为“1”时的操作依赖于/W 引脚的状态。如果/W 为高电平，也就是 WEL 位（通过 WREN 命令）被设置，则可以进行状态寄存器的更新；而如果/W 为低电平，则不能更新状态寄存器。

▲　BP1 / BP0（Block Protect，区域保护）

M95256 通过 BPI 和 BP0 这两位，就可以确定写保护范围和许可范围。其关系如表示。

表 M95265BP 位的分配

BP1	BP0	保护区域	地址区域
0	0	无	无
0	1	高位 1/4	6000h～7FFFh
1	0	高位 1/2	4000h～7FFFh
1	1	整体	0000h～7FFFh

通过范围的设定，对被保护范围发出的 WRITE 指令被忽略。关于未被保护的范围，只在下面说明的当 WEL 位为“1”时可以进行写人操作。

由于 BP 位的变换只能通过状态寄存器的替换进行，所以通过前面描述的 SRWD 位的功能，状态寄存器的更新作为对器件的保护功能也能进行。

▲　WEL（Write Enable Latch，写使能锁）

这是只读（专用于读出）位。该位为“1”时，可以对状态寄存器进行写人操作。而只有该比特位为“1”、并且根据BP位并未被保护的时候，才能对存储器单元进行写人操作。在提供电源后，该比特位自动变为“0”。

该比特位在发出了WREN指令时为“1”。操作过程中变为“0”需具备以下 3 个条件：

①WRDI指令完成；

②WRSR指令完成；

③WRITE指令完成。

也就是说，利用WREN使写入操作被许可后，一旦对状态寄存器及存储器单元进行写人操作，则自动恢复为写人禁止的状态。

但是，正如前面讨论过的那样，当具各SRWD为“1”且/W输人为低电平这样的条件、且M95256处于硬件保护模式时，通过软件无论怎样控制（即使发出WREN指令），也不能变为写入许可，而且也不能更新状态寄存器。只有将/W引脚变为高电平才能恢复。

▲　WIP（Write In Progress）

与闪速存储器及Microwlre对应的EEPROM等相同，从发出WRITE指令，至完成对单元的写人操作还需要相当的时间。因此，为确认由主机对芯片内部的写人处理是否完成，还设计了WIP位。WIP位为“1”时，芯片内部的替换操作正在进行，因而主机不能发出READ及WRITE指令等。

3.5　I^2C总线对应的存储器——M24Cxx

I^2C总线是Philips公司所提倡的两线式的简单接口。它不但应用于EEPROM，而且还应用于LCD驱动器以及RAM、I／O端口等。I^2C总线上不但能连接若干个从属控制器，还可以连接多主控器，因而可以

在总线上连接多个主机，共享该总线。总线速度由Version1.0所定义的标准模式（最高时钟为100kHz），再加上快速模式（至400kHz），提高到1998年上市的Version2.0的3.4Mbps速度。由于我们所举出的实例M24C01等是基于版本Version1.0的，因此时钟速度最高至400kHz。

I^2C总线上数据的传输单位为8位。因为在发送完8位的数据后，接收方将返回1位的状态值（ACK／NoACK），因此I^2C总线是以共计9时钟周期为一个传输单位的。

I^2C总线传输的流程如图所示。在I^2C总线上，从起始条件到结束条件是一个传输操作的单位。通过在通常的数据传输中设置不出现在总线工作模式中的起始／结束条件，可以避免总线上因为其他器件传输的数据所造成的误操作。

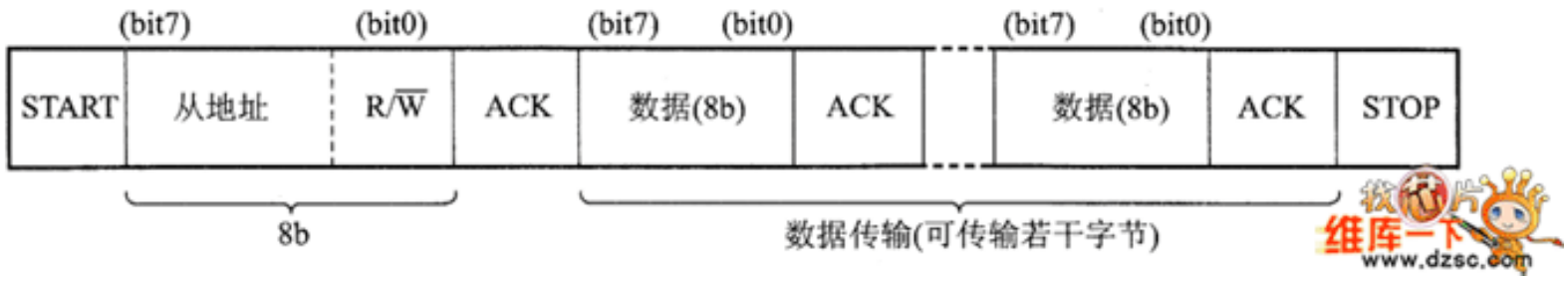


图　I^2C总线的数据传输流程

起始条件之后，开始进行传输的起始字节的格式如表所示。最低位是用于区分读／写操作的，前7位是规范上称为“从地址（Slave Address）”的字段，从名称上猜测是希望在此指定总线上的S从地址编号（0～127），但事实上该字段也包括指定目标类型。

表　I^2C总线起始字节的格式

从地址							R/W	意义
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
0	0	0	0	0	0	0	0	General Call Address
0	0	0	0	0	0	0	1	START byte
0	0	0	0	0	0	1	X	CBUS address
0	0	0	0	0	1	0	X	Reserved for different bus format
0	0	0	0	0	1	1	X	Reserved for future purposes
0	0	0	0	1	X	X	X	Hs-mode master code
1	1	1	1	1	X	X	X	Reserved for future purposes
1	1	1	1	0	X	X	X	10-bit Slave addressing
1	0	1	0	E_2	E_1	E_0	R/W	（M24Cxx就是如此使用）

正如M24C01 等在表中所表示的那样，地址字段的前 4 位（位 7～4）是“1010”时，表示为目标对象，紧接着的 3 位作为器件编号来使用。

我们并不是要故意略去I^2C总线的介绍，只是想将内容限定于I^2C总线在M24Cxx系列的使用方法上。

● 3.5.1　I^2C总线与串行EEPROM

在I^2C总线上，数据是串行传输的，但数据的读 / 写是以 8 位为单位的，不可以只指定存储器内部某特定的比特位进行读或者替换操作，至少需要以 8 位为单位进行存取。

M24Cxx与主机的连接实例如图 1 所示，简单的传输格式实例如图 2 所示。在I^2C总线对应的EEPROM中，认为最多可以连接 8 个地址，每个地址至多为 8 位（256 字节，2K位）的存储器，基于I^2C总线的规范标准，利用起始字节的 0 位区分读或写操作，利用 1～4 位指定器件编号。这样，就可以指定接收数据的器件。然后接着的第 2 字节是存储器地址，第 3 字节以后为数据。

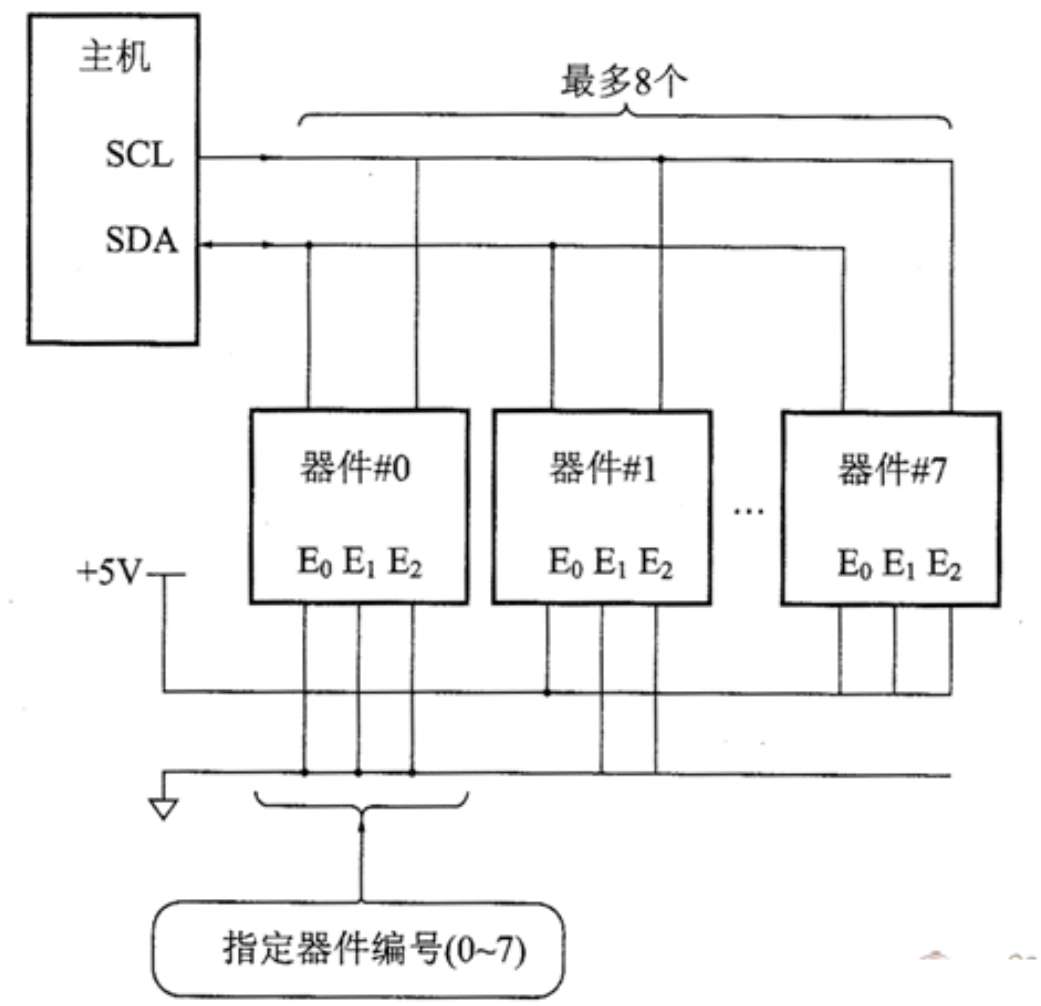


图 1　I^2C总线对应的存储器与主机的连接

因为可以连接地址为 8 位（256 字节，2K位）的 8 个器件，因此总线上的最大存储器空间为 16K 位。这样的空间容量已不能满足目前的需求，容量不足的情况逐渐增多。为此，将地址字段设置为 2 字节，将I^2C总线扩展做成最多可连接 8 个 64KB（512K位）的存储器，这样一般就可应用于超过 16K位容量的EEPROM上。由于存储器空间最大为 512KB，从I^2C总线的传输能力上看，这个程度应该足够了。

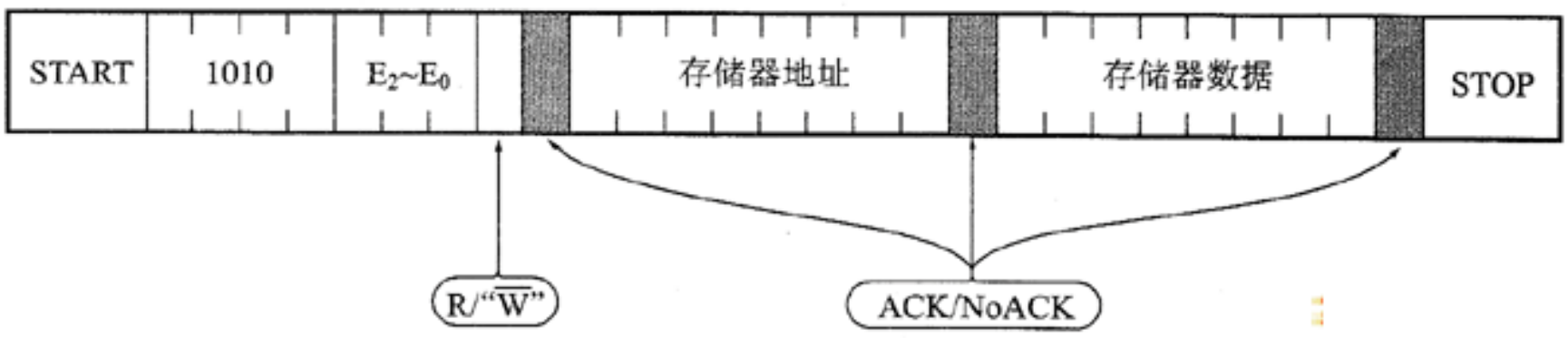


图 2　　I^2C总线的存储器存取操作示例

● 3.5.2　I^2C总线存储器M24C01～M24C16

存储器包括M24C01，M24C02，M24C04，M24C08 以及M24C16 等，但无论哪种存储器，其引脚配置都是相同的，如图所示。我们在介绍I^2C总线的操作之前，先针对这些引脚进行简单的说明。

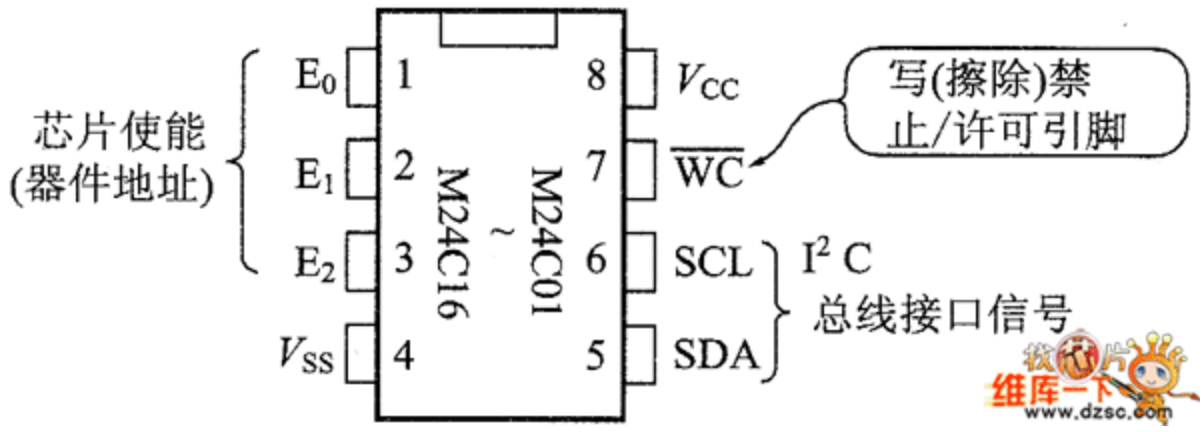


图　　M24Cxx 的引脚配置

▲ Vcc / Vss

这是电源引脚。Vss 是基准电位（0 伏），Vcc 电压不但存在＋5V（4.5～5.5V）的产品，还存在 2.5～5.5V、甚而 1.8～3.6V 电压更低的产品。

▲ SCL（Serial Clock，串行时钟）

这是时钟输人引脚。说是时钟，但由于不需要特殊的固定周期，所以只要认为是来自主机的数据选通信号即可。利用该时钟信号和 SDA 信号，主机可以赋予地址，或者进行数据的读 / 写操作。

▲ SDA（Serial Data Input / Output，串行数据输入 / 输出）

这是用于传输地址和数据的引脚，可双向使用。因为 M24C01 的输出是漏极开路输出，因此需要上拉电阻。

▲ E0 / E1 / E2（Chip Enable Input，芯片使能输入）

在M24Cxx的手册上是Chip Enable，但由于称为总线上的器件编号比较容易理解，因此在这里我们就称之为器件编号。如前所述，在I^2C总线上最多可以连接 8 个最大为 256 字节（2K位）的存储器，主机通过I^2C总线送出所访问的器件编号和地址。I^2C总线上的器件将主机送来的器件编号与通过E0 / E1 / E2 引脚所提供的值进行比较，如果一致则知道自己被选中。

由于在I^2C总线上每个器件编号所相当于的范围最多为 256 字节，所以在超过该范围容量较大的器件上器件编号也作为地址使用，形成一个器件独占连续范围的格局。也就是说，从主机看去是具有多个 256 字节的器件。

例如，在M24C04（4KB）中E0 是无效的，如果设定E1＝“L（低电平）”、E2＝“H（高电平）”，那么器件编号的 4 号和 5 号将由 1 个M24C04 所专用。如果是M24C16 则容量为 16K位，一个M24C16 将占用I^2C总线上的所有空间，因此E0～E2 引脚不具有任何意义

● 3.5.3　I^2C总线的基本操作

I^2C总线上只具有SCL（时钟）和SDA（数据）2 根信号线。如果是单纯的串行传输，一旦因为某种原因造成引脚的偏差，则可能会造成不能区分总线上传输的是数据还是地址信息的后果。解决上述问题的简单办法就是附加独立于总线的Reset（复位）信号，由主机控制该信号。因为I^2C至少利用 2 根

线进行所有的操作，因此在数据传输时，通常当SCL为低电平时，设置下一个数据；当SDA变化后，SCL为高电平，这可以解释为一连串操作的开始 / 结束。

▲　起始条件

起始条件表示一系列操作的开始。图 1 表示起始条件以及随后数据传输的开始操作。在I^2C总线的空闲状态下，SDA及SCL通过上拉电阻都为高电平。在这样的状态下，如果SCL仍保持高电平，而SDA变为低电平,则成为开始指令。

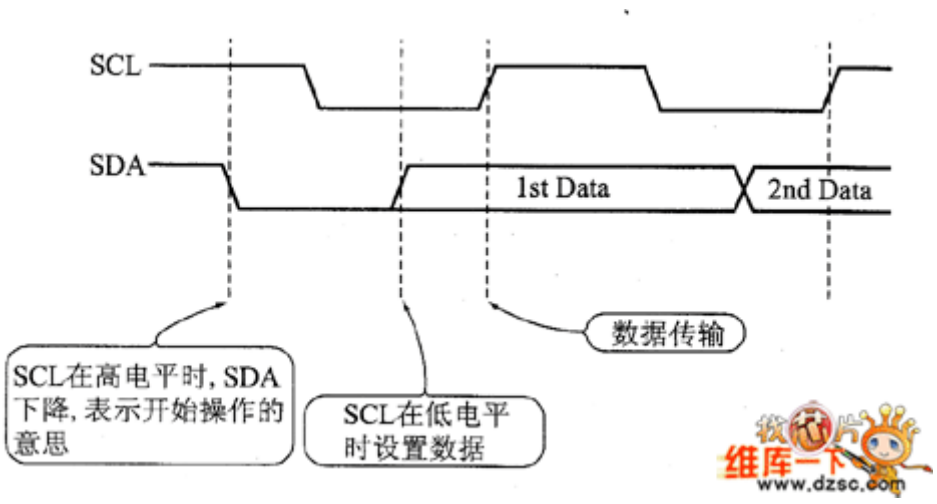


图 1　　I^2C总线的起始条件

由于该状态并不出现于地址及数据的发送与接收过程中，因此，即使在途中发生异常，只要检测出该状态，初始化内部的状态机，就可以使其恢复。

▲　结束条件

在一系列操作的最后是结束条件。结束条件如图 2 所示。当 SCL 为高电平时，一旦 SDA 由低电平变化为高电平，即成为结束条件，主机与器件之间的通信将停止，器件恢复为空闲状态。进行写操作时的结束状态是开始进行 EEPROM 内部单元写操作的指示标志。

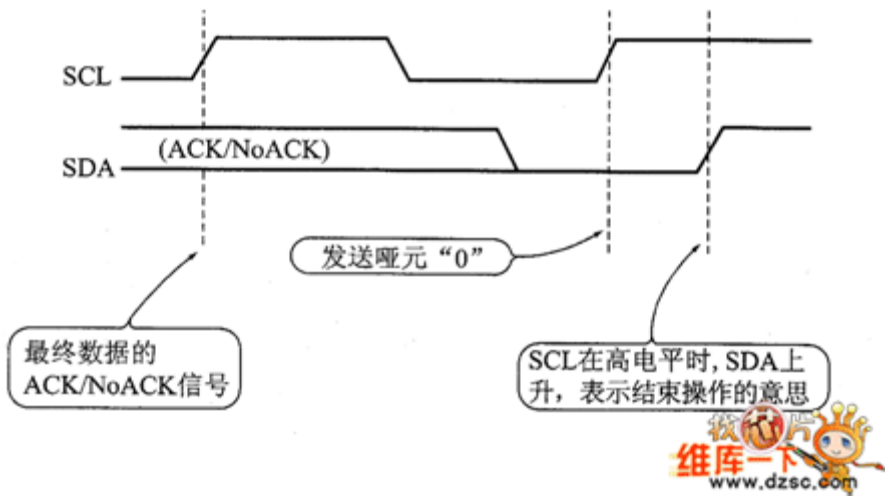


图 2　　I^2C总线的结束条件

结束前所传输的数据是 ACK / NoACK 的状态位，如果是 ACK，则该状态位为低电平。但如果发生某种错误时，则表示为 NoACK 的高电平。读操作时的最后字节是主机向器件返回 NoACK 信息，所以 SDA 为高电平。这样就不能形成结束条件所需要的 SDA 的上升沿，因此在结束之前需要加入哑元“0”数据位作为解决办法。

在最终数据的 ACK/NoACK 之后，主机通过下述的流程，形成结束条件，如下所述：

①SCL 变为低电平；

②SDA 变为低电平；

③SCL 变为高电平（发送哑元数据）；

④SDA 变为高电平（结束条件）。

▲　数据传输

数据传输的流程如图 3 所示。除去开始与结束条件，在传输包含地址指定等数据时，能够使 SDA 发生变化的条件只能是在 SCL 为低电平时。因此，总线操作以如下的步骤进行：

①SCL 变为低电平；

②为 SDA 设置数据（主机或者器件）；

③SCL 变为高电平。

进行数据读操作时，主机在 SCL 恢复为高电平之前读取数据。

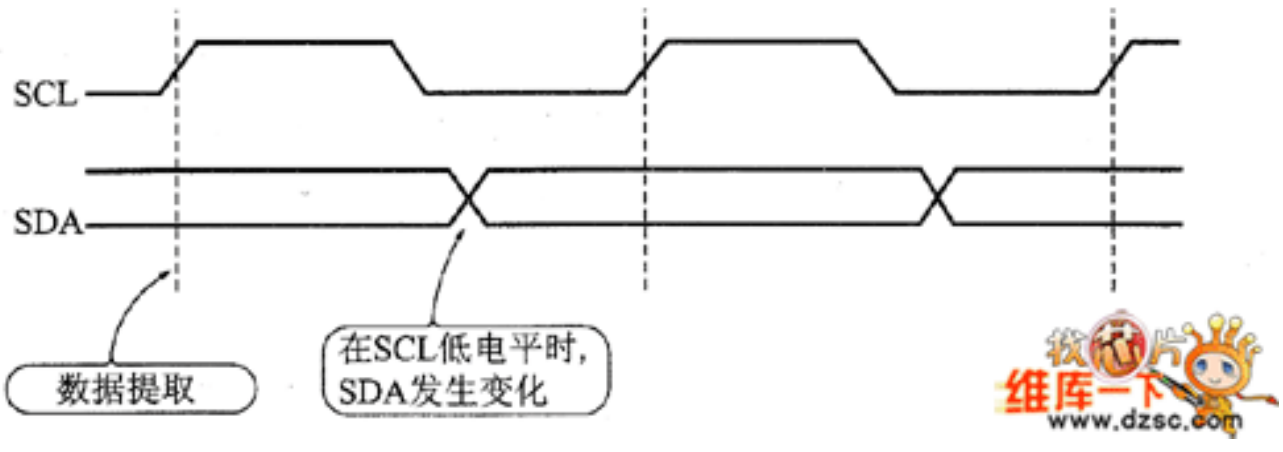


图 3　I2C 总线的数据传输

● 3.5.4　写操作的流程

在I^2C总线上传输操作是以 8 位＋ACK / NoACK共计 9 位为单位进行的。发送是从位 7（MSB）开始进行的，虽然一般的串口（PC机的COM端口等）是由位 0（LSB）开始发送的，但I^2C总线是相反的，这一点需要注意。接收 8 位数据或者指令的接收方在下一个时钟输出ACK / NoACK位，如果是低电平则表示ACK；如果是高电平则表示NoACK。

写操作包括字节写及页面写两种。字节写是只置换特定的 1 个地址；而页面写可置换汇总了 16 字节界限内的连续的地址范围（页）。各种写操作的流程图如图所示。

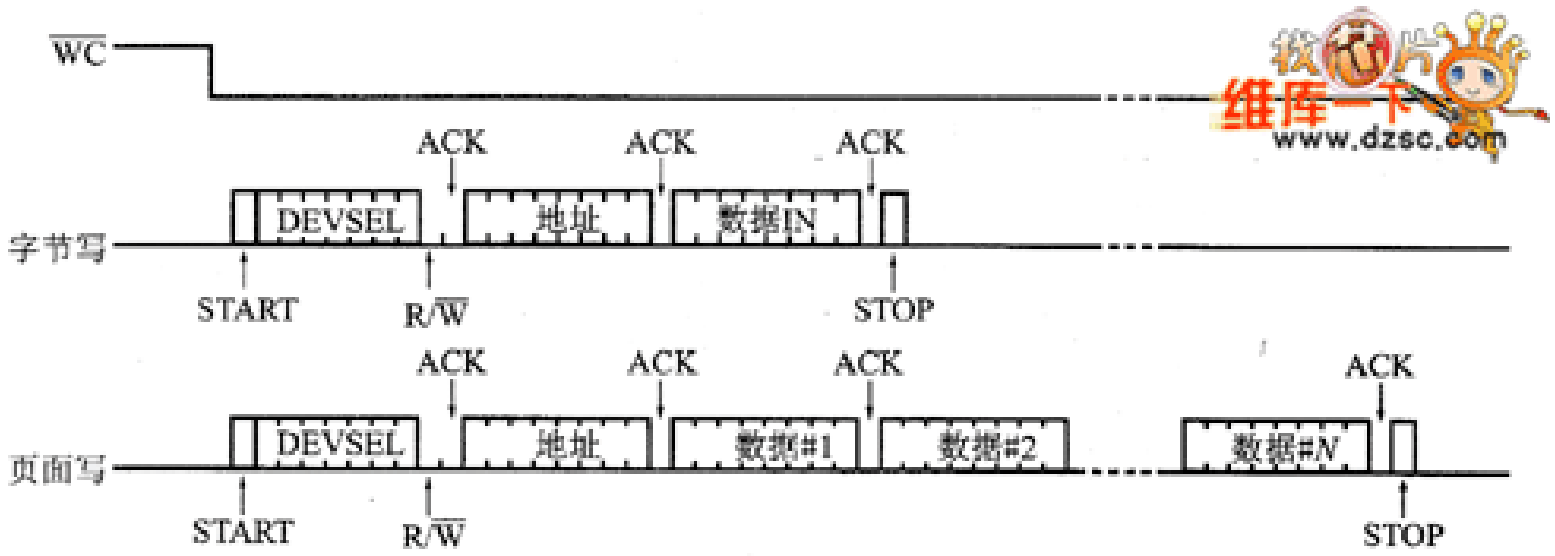


图　　I^2C存储器的字节写与页面写

▲　字节写

这是指定任意的地址（8 位）写人数据的方式。

起始字节的 DEVSEL 按照前面描述的起始数据的格式，位 7～4 是“1010”的固定模式，读操作（“1”）还是写操作（“0”）。

如果存储器处于写保护状态（/WC 引脚为高电平等），则在接收地址之前一直返回 ACK 信号，对于之后传输的数据，则返回 NoACK 信号。

数据发送完毕后，如果检测出来自主机的结束条件，则 EEPROM 内部开始进行写入操作。根据数据手册可查出完成写入操作所需要的时间。＋5V 的产品需要 5ms，其他的产品大约需要 10ms 左右的时间。

EEPROM 内部的替换周期在进行过程中，即使发送下一个指令也将返回 NoACK 信号。因此，利用该 NoACK 信号就可判断内部的操作是否完成。

▲　页面写

页面写操作本身与字节写是相同的。由于在存取后地址自动进位，因而可以按序发送 1 页（16 字节）以内的数据。实际上向存储器单元的写人操作，与字节写相同，都是在检测出结束条件后进行的，所以需要等待，直到操作结束，然后再进行下一个操作。

● 3.5.5　读操作的流程

M24Cxx 的读操作模式及其各种模式的操作流程如图 1、图 2 所示。

▲　当前地址读

EEPROM 内部具有保持当前地址的寄存器。读取当前地址的数据时，不需要指定地址。只要单纯给出读指令就可读出数据。读取完毕后，内部所保持的当前地址将自动进位。

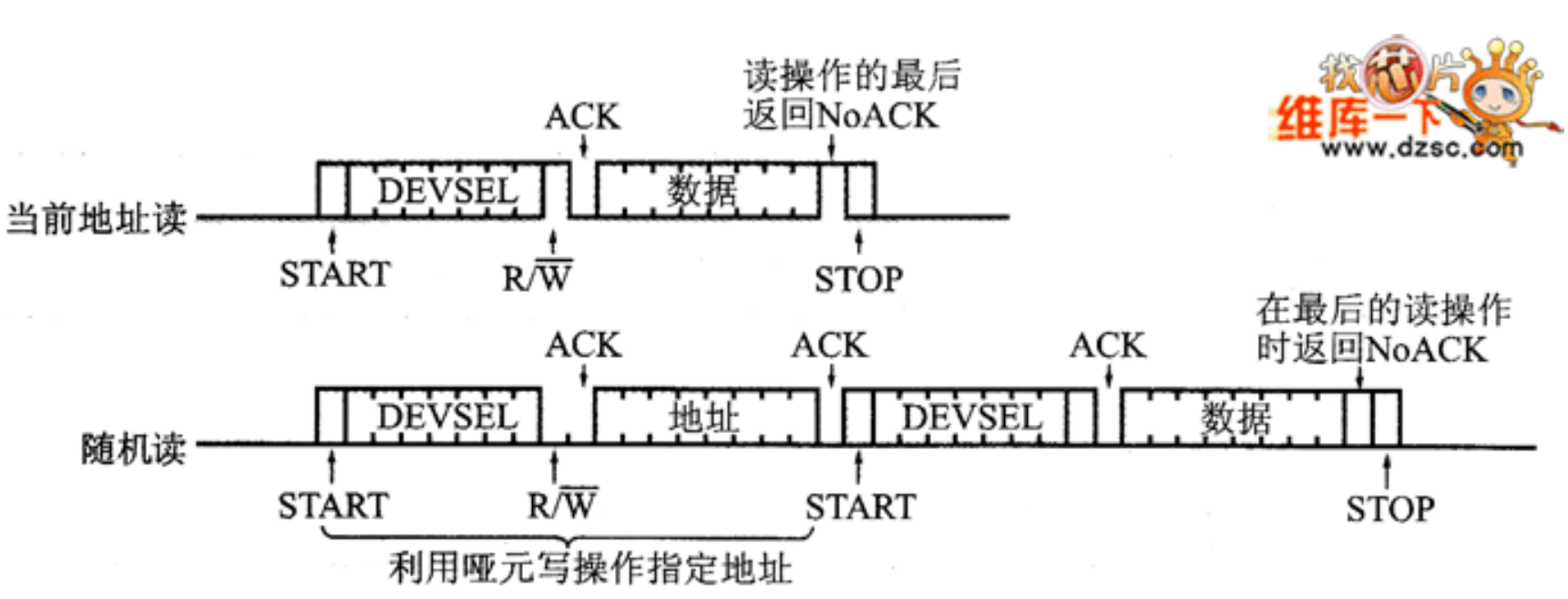


图 1　I²C存储器的读操作（1）

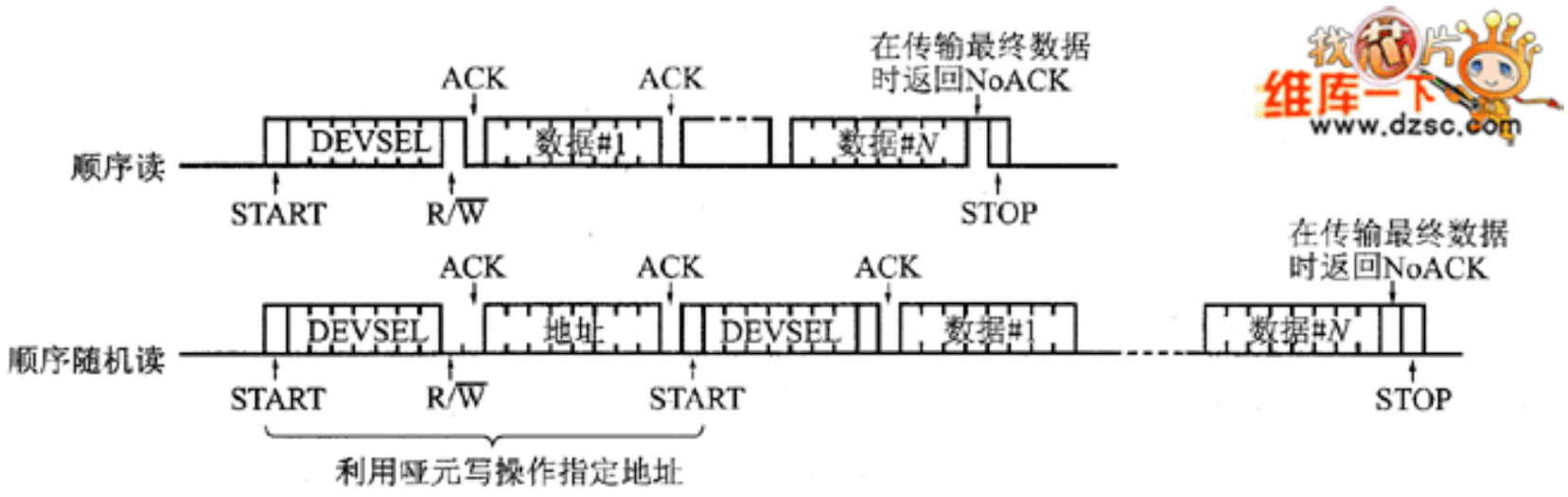


图 2　I²C存储器的读操作（2）

数据读操作后的 ACK / NoACK 信号由主机返回，但必须返回 NoACK 信号。

▲　随机读

随机读是由主机指定任意的地址读取的。利用写指令设定地址，如果赋予读指令则可以读出当前地址。所以，与字节写操作时相同，在第 1 字节的数据后面给出地址。在这里，一旦发送出数据就成为写操作，在此设置开始条件，取消向写操作的迁移而发出读指令，将从事先设定的地址中读出数据。

此时，DEVSEL 数据（前 7 位数据）必须设定与最初写指令所发送的相同的值。

▲　顺序读

在当前地址读操作之后，如果主机返田 ACK 信号，则为顺序读模式，器件将准备下一个地址的数据，主机取回该数据。一旦到达要读出的最终地址，主机将返回 NoACK 信号，通知器件这已是最后的数据。

▲　顺序随机读

当指定任意地址、希望由此连续读出数据时，可利用该模式。只要认为这是与对应于当前读的顺序读相同的模式即可。

顺序随机读模式与随机读同样进行读操作，接收到数据后如果是 ACK 应答，则器件将准各下一个地址的数据；如果是最终数据，则返回 NoACK 信号，结束数据的传输。

● 3.5.6　扩展I^2C总线存储器

如前所述，因为 8 位地址不够用而导人的总线称为扩展I^2C总线。除了地址指定为 2 字节以外，其他的处理过程完全相同。图 1 和图 2 分别表示写和读时的操作。地址是按高位（A_{15}～A_8）和低位（A_7～A_0）的顺序发送的。

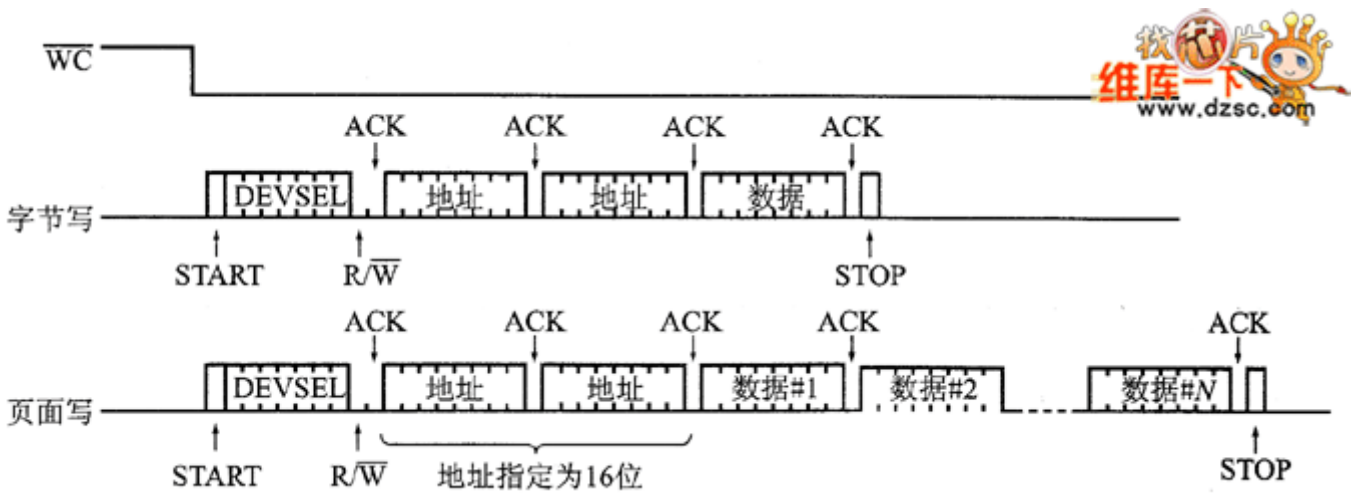


图 1　扩展I^2C存储器的写操作

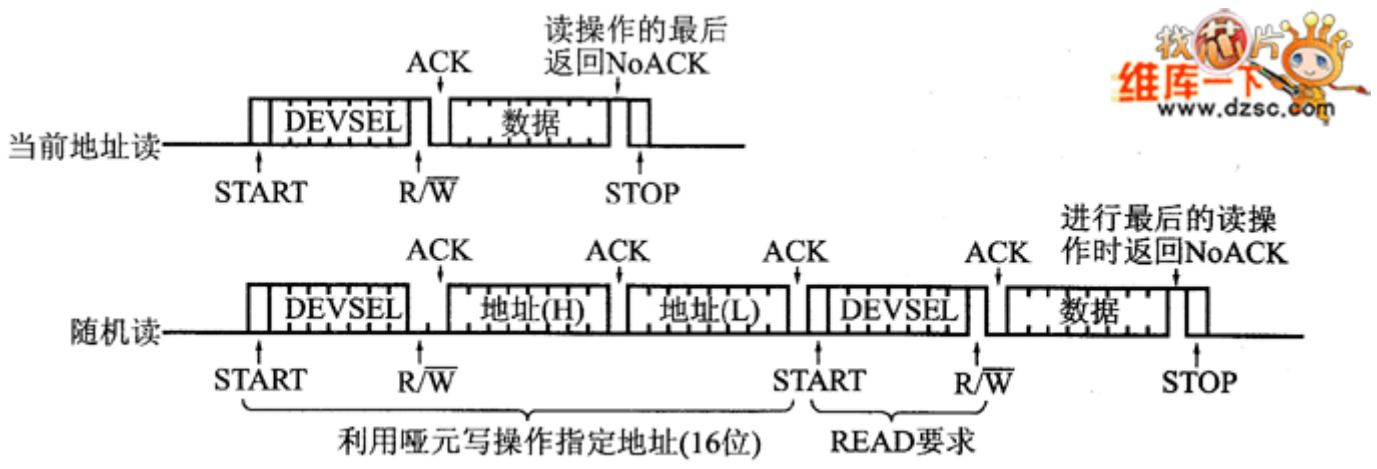


图 2　扩展I^2C存储器的读操作

● 3.5.7　M24C64 的时序

M24C64 的时序图如图及表所示。I^2C总线只有 2 根信号线，基本上是以时钟信号为基准进行操作的，所以大多数是时钟上升沿与下降沿的规定。表中大多也是这样详细的规定，我们只介绍重要的部分。

图　M24C64 的时序图

表　M24C64 的 AC 时序规定

符　号	意　义	min	max	单　位
t_{CH1CH2}	SCL(时钟)上升时间		300	ns
t_{CL1CL2}	SCL 下降时间		300	
t_{DH1DH2}	SDA 上升时间	20	300	
t_{DL1DL2}	SDA 下降时间	20	300	
t_{CHDX}	从 SCL 上升到 SDA 输入变化所需要的时间	600		
t_{CHCL}	SCL 高电平的时间	600		
t_{DLCL}	从 SDA 输入下降到 SCL 为低电平的时间(START 时)	600		
t_{CLDX}	从 SCL 上升沿到 SDA 输入发生变化所需要的时间	0		
t_{CLCH}	SCL 低电平时间	1300		
t_{DXCX}	从输入变化到时钟变化的时间	100		
t_{CHDH}	从 SCL 上升到 SDA 输入高电平(STOP 时)	600		
t_{DHDL}	从 SDA 上升到下一个下降的时间(总线自由期间)	1300		
t_{CLQV}	从 SCL 的下降到 SCA 输出确定的时间	200	900	
t_{CLQX}	从 SCL 下降开始的 SDA 输出保持时间	200		
f_c	SCL 频率		400	kHz
t_W	写入操作时间		10	ms

▲ 时钟规定

关于时钟，可以说有五大限制，即tCHCL（时钟为高电平时间）和tCLCH（时钟为低电平时间）、图中未出现的tC（时钟频率）、tCH1CH2（时钟的上升时间）以及tCL1CL2（时钟的下降时间）。

tCLCH为 1.3μs，tCHCL最小为 600ns，fc最高为 400kHz。所谓的 400kHz就是如果占空比为 50%，低电平 / 高电平时间分别为 1.25μs，因此，要在最高 400kHz时钟下提高传输速度，则占空比就不是 50%，这样就需要在稍微延长低电平时间以及缩短高电平时间上下一番功夫。如果作为ISA总线的 8 位 I / O来利用，那么每存取一次就需要不到 1μs的时间。只根据SDA设置及时钟高电平这样的周期计算，所需要的时间不到 2μs，因此这不会成为问题。

虽然时钟的下降确实不需要 300ns，但上升如果是通过漏极开路输出的，则将出现上拉电阻及配线容量的问题。关于上拉电阻的选定，I^2C总线的规格说明书上都有记载，但应该结合为减少功耗而做的计算和波形观测的结果来确定。

▲ SDA（输入）建立

在数据传输中，SDA 在时钟的上升沿被提取。因此必须在时钟的上升沿之前确定 SDA。这个时间就是 tDXCX，为 100ns。

▲ SDA（输入）保持

在I^2C总线上，当SCL为高电平时SDA发生变化，将会形成开始条件或者结束条件。因此，数据传输过程中，能让SDA状态发生变化的只限于SCL为低电平的期间。因此，在SCL下降之前，需要保持SDA的状态，只有在SCL为低电平之后，SDA才能发生变化。

这个持续保持状态的时间就是保持时间，简写为tCLDX，为0μs。在这里，我们只要明白该时间不可为负值即可。

▲ 开始条件的规定

困为开始条件是 SCL 为高电平时 SDA 变为低电平而形成的，所以需要注意以下两点：

①从 SCL 成为高电平到 SDA 成为低电平的时间（tCHDX）；

②从 SDA 成为低电平到 SOL 成为低电平的时间（tDLCL）。

无论哪个期间都为 600ns。因此，与 ISA 总线的每个存取低于 1μs 相比，可以说时间是足够短的。

▲　结束条件的规定

与开始条件相同，结束条件也需要相同规定的验证。

①从 SCL 成为高电平到 SDA 成为高电平的时间（tCHDH）；

②从 SDA 成为高电平到 SDA 成为低电平重新生成开始条件的时间（tDHDL）。

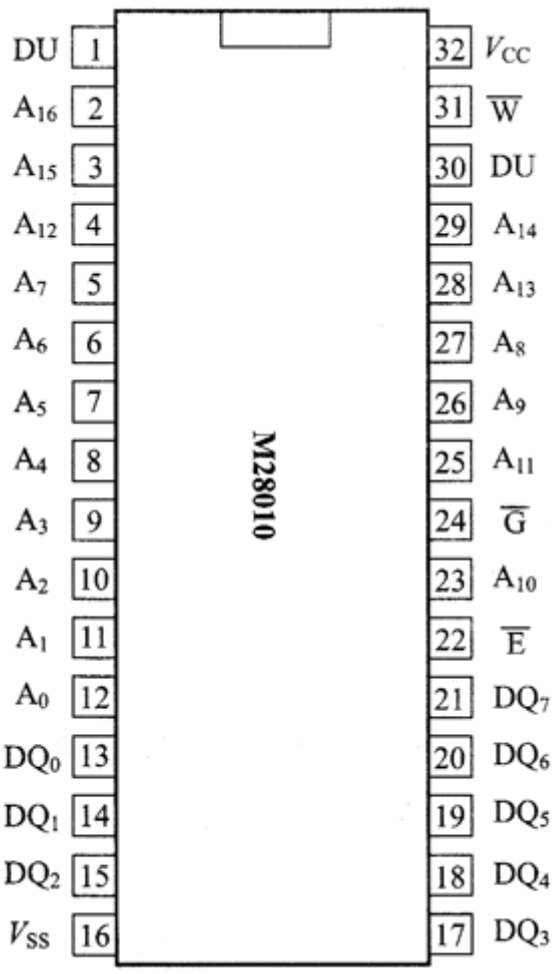
这两个时间分别为 600ns 和 1.3μs。在数据传输等过程中，SDA 与 SCL 的交互操作需要不到 2μs 的时间，而从开始条件到结束条件，只有 SDA 的操作是连续的，所以存在未满足 1.3μs 这一规定的可能性。因此，可在 SDA 成为高电平之后需要在哑元的 I／O 存取上争取时间，可在同一数据 2 次写人时争取时间。

▲　数据存取时间／保持时间

进行读操作时，EEPROM的数据是在SCL的下降时被开始输出的。在开始输出之前，SDA上保持着上回传输的数据，确保这一数据的时间就是tCLQX，确保新的数据被确定之前所需要的时间是tCLQV。tCLQX 最小为 200ns；tCLQV最小为 200ns，最大为 900ns，所以从SCL成为低电平到在ISA总线等的I／O读周期上读出数据为止，感觉其中时间还是非常紧张的。在此之间最好插入哑元周期。

3.6　并行 EEPROM

作为并行 EEPROM 的实例，我们举出 ST MicroElectronics 公司 1MB 的 M28010 来进行说明。DIP 式的引脚配置如图所示：与 AMD 闪速存储器命名信号名称的方式不同，DU 等同于 NC，其他/W，/G，/E 分别等同于/WE，/OE 和/CE。我们知道引脚配置本身具各兼容性。



DU:Don't Used(未使用的)

图　M28010 的引脚配置

并行 EEPROM 的内部也同闪速存储器一样，包括获得高电压用于替换的升压电路以及对内部单元的写入控制电路。写人操作可以针对任意地址，与闪速存储器相同，在从 CPU 端进行写人操作到内部的替换操作结束．需要一定的时间。因此，和闪速存储器一样，如果在替换中要求读取数据，则将读出状态值，根据这个特点可以判断替换操作是否已完成。还有一个相同点就是可以汇总对同一页的替换进行处理。

与闪速存储器较大的不同之处在于以下两点：

①写人操作只要简单地写即可；

②写保护功能没有特殊的电压控制也可利用。

虽然闪速存储器可以写入任意的地址，但为了进行写入操作，必须发出多次指令序列。与此相对应，只要 EEPROM 没有被执行保护功能，就可以对该地址进行写人操作。

但是，像这样简单进行的写人操作，反过来可具有因电气异常而导致误替换的危险性。为此，EEPROM 也要具各写保护功能，以避免不经意的替换。

● 3.6.1　M28010 的信号

我们在了解 M28010 的操作之前，先简单介绍各种信号。

▲ A0～A16（Address Input，地址输入）

这是地址总线，由这些引脚指定进行写人 / 读取操作的地址。

▲ DQ0～DQ7（Data Input / Output，数据输入 / 输出）

这是双向数据总线。进行数据读操作时，如果/E 和/G 都有效，则在这些引脚上将输出数据；而在进行写操作时，在这些引脚上事先设置数据，如果/E 和/W 都有效，则数据将锁存于存储器中。

▲ /E（Chip Enable，芯片使能）

这是片选信号，该引脚无效时，忽略/W 和/G 等。

▲ /W（Write Enable，写使能）

写人数据及主机赋予指令时，/W 与/E 信号都有效。在/E 和/W 都有效之后，无论使哪个信号无效，数据都将被提取到内部。

▲ /G（Output Enable，输出使能）

这是数据输出使能引脚。在进行存储器读 / 写操作之后的状态读操作时，/G 与/E 都有效。

● 3.6.2　基本的存取操作

M28010 基本的存取过程与闪速存储器完全相同，只包括对器件的读操作及写操作两种。擦除以及保护功能等可以通过写操作发出一系列的指令序列来进行。

▲ 读操作

EEPROM 的读操作如图 1 所示。注意事项与闪速存储器的基本相同，因此这方面请参考闪速存储器的注意事项。赋予地址，使/E 和/G 有效，然后等待一定时间后出现数据。这样，主机提取该数据，然后使/E 和/G 无效，结束存取周期。

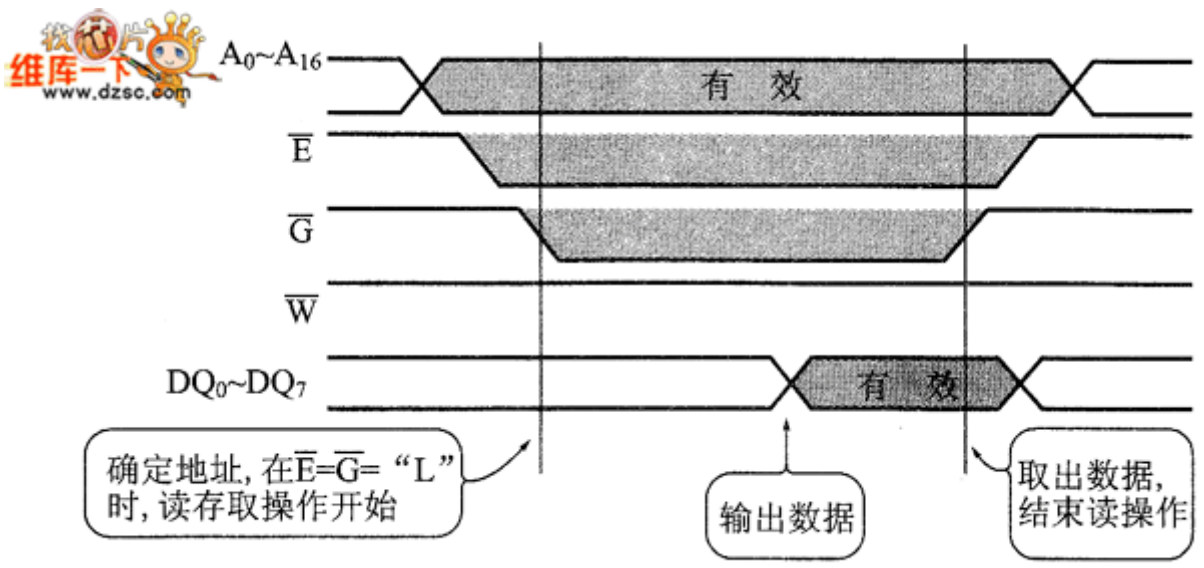


图 1　EEPROM 的读操作

因为与串行 EEPROM 不同，它不与时钟同步运行，所以需要参考数据手册中的存取时间等，设计成在只经过认为数据确实被确定的时间后，主机再进行数据读操作。虽然不必太介意像 ISA 总线那样的低速总线，但像目前的处理器那样，如果外部总线的操作较快，则在外部需要增加让 CPU 等待的电路，以调整时序。

▲　写操作

EEPROM 的写操作也与闪速存储器相同，包括利用/W 信号控制（/W 写控制）和利用/E 信号控制（/E 写控制）两种方法，图 2 和图 3 简单说明了这两种方法的操作。

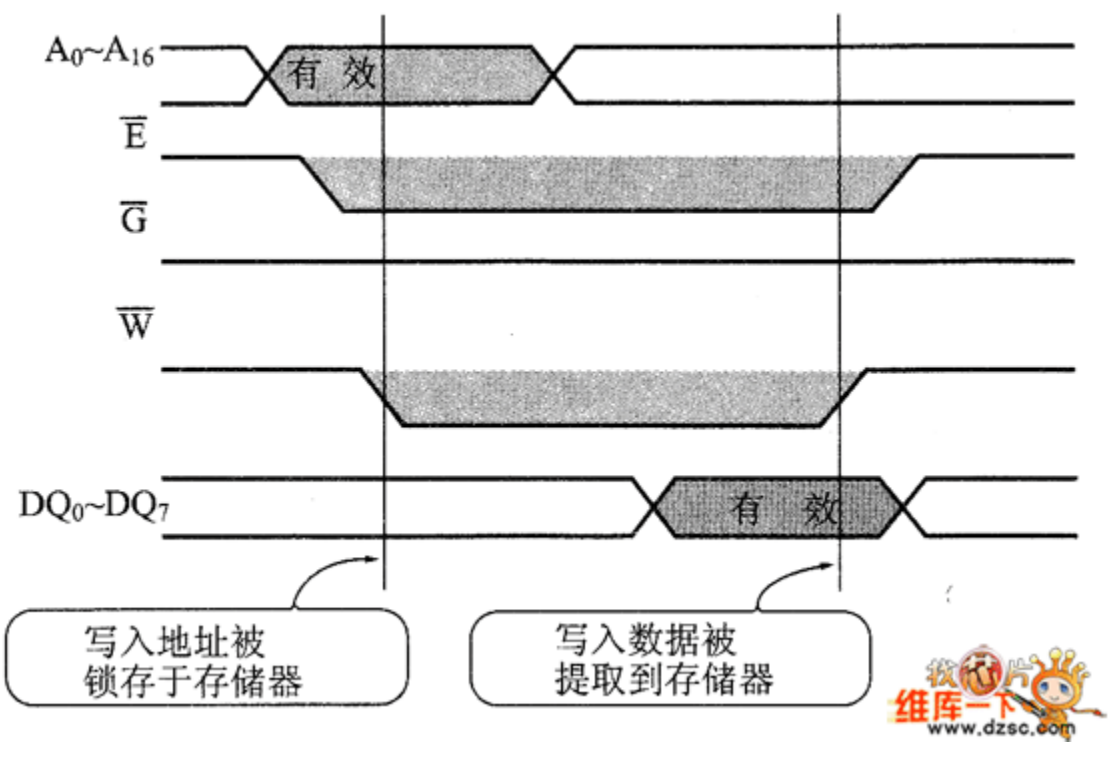


图 2　EEPROM 的写操作（1）（/w 写控制）

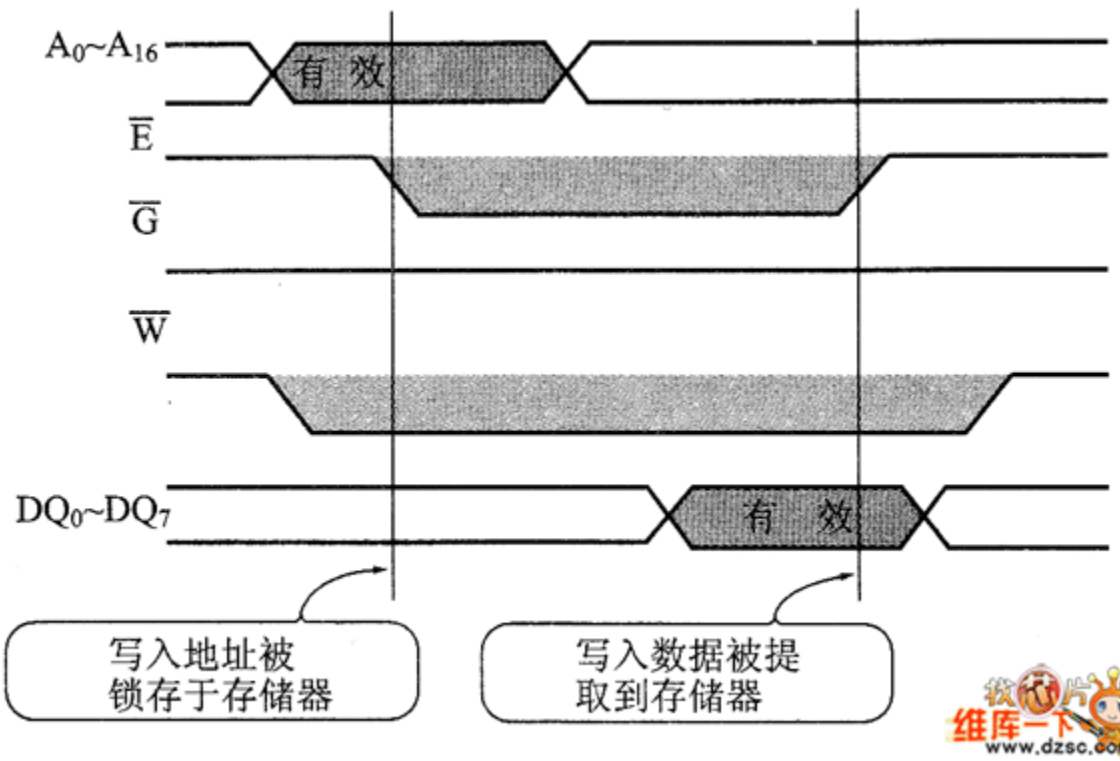


图 3　EEPROM 的写操作（2）（/E 写控制）

无论哪种方法，在/E 或/W 延迟有效的下降沿，地址被提取，而在提前无效的一方的上升沿，数据被提取。

● 3.6.4　状态寄存器

与闪速存储器相同，EEPROM 一旦开始向芯片的写人操作，则存储器单元将与外部总线分离，对于其间的读取操作将返回状态。M28010 的状态寄存器如图所示。下面我们说明这些引脚的作用。

DQ_7	DQ_6	DQ_5	DQ_4	DQ_3	DQ_2	DQ_1	DQ_0
DP	TB	PLTS	×	×	×	PWA	SDP

DP　：Data Polling (替换操作中将写入数据反相)
TB　：Toggle Bit (替换操作中，每读出一次数据就将其反相)
PLTS　：页面负载定时器状态
×　：未定义
PWA　：中断页面写操作
SDP　：软件数据保护状态

图　M28010 的状态寄存器

▲　位 7：DP（Data Polling）

与闪速存储器相同，在完成内部写操作之前的这段时间内，将反相读出最后写入数据的位 7。如果看到写人的数据与读取的数据一致，则可检测出替换操作已经完成。

▲　位 6：TB（Toggle Bit）

这也与闪速存储器中的 Toggle Bit 相同，是用于了解替换操作是否完成的位。位 7 是将写人数据反相后读出的，而位 6 是在完成替换操作之前，每进行一次读操作，就读出反相的数据。在发出写及擦除等指令后的读操作中，读出“0”，以后将为“1”，“0”，“1”，“0”这样只要读取就将其反相。

因为只要完成替换操作就成为普通的读周期，所以数据将不再反相，这样就可以检测出替换操作是否完成。

▲　位 5：PLTS（Page Load Timer Status）

前面曾经叙述过，EEPROM 可以连续进行同一页的写入操作。这样的页面写是在从一次写操作到下一次写操作之间不超过一定时间（tWLQ5H）内连续进行的。芯片具有监测这段时间的定时器，如果超时，则开始进行存储器单元的替换操作。

PLTS 位显示该定时器的状态，在超时之前一直读出“0”，而如果超时，则开始内部操作，PLTS 位将变为“1”。

▲　位 1: PWA（Page Write Abort）

页面写时能够进行置换的只限于同一页，因为 M28010 一页大小为 128 字节，所以 A7～A16 不允许发生变化。如果在页面写的过程中发生向不同于 A7～A16 值的地址的写操作，则将取消全部的页面写操作，不进行替换操作。

此时，EEPROM 端在 tWLQ5H 期间或者在 W 保持高电平状态的 2 次读周期之间，在能够读出状态的基础上，将该位变为“1”，然后向主机发送页面写操作出错的通知。

▲　位 0：SDP（Software Data Protection）

根据软件数据保护机制，该位表示是否进行了保护操作。该位如果是“1”，则为保护状态；如果是“0”，则为保护解除状态。

第四章

SRAM 的结构与使用方法

4.1　SRAM 的单元结构

SRAM 利用电路自身的状态进行存储。虽然电路种类很多，但基本上都是构成称为“触发器”的电路。在这里，作为存储器单元结构，我们只介绍普通常见的存储器单元。

●4.1.1　RS 触发器

作为触发器，具有少许逻辑电路知识的读者会马上联想到 RS 触发器，如图 1 所示。若将 NOR 门利落地布线就成为了 RS 触发器。简单的操作示例如图 2 所示。一般地，首先将 R（Reset）和 S（Set）都设置为低电平，Q 及/Q 的初始状态虽然不明确，但在此，若 S 成为高电平，则 Q 为高电平，/Q 为低电平。

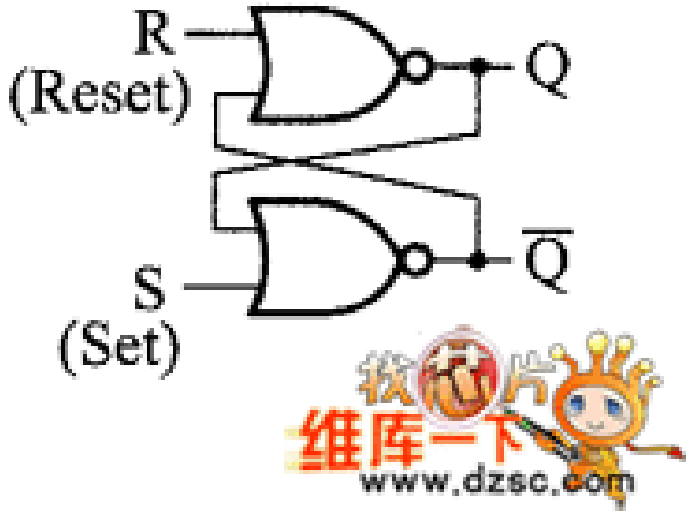


图 1　RS 触发器

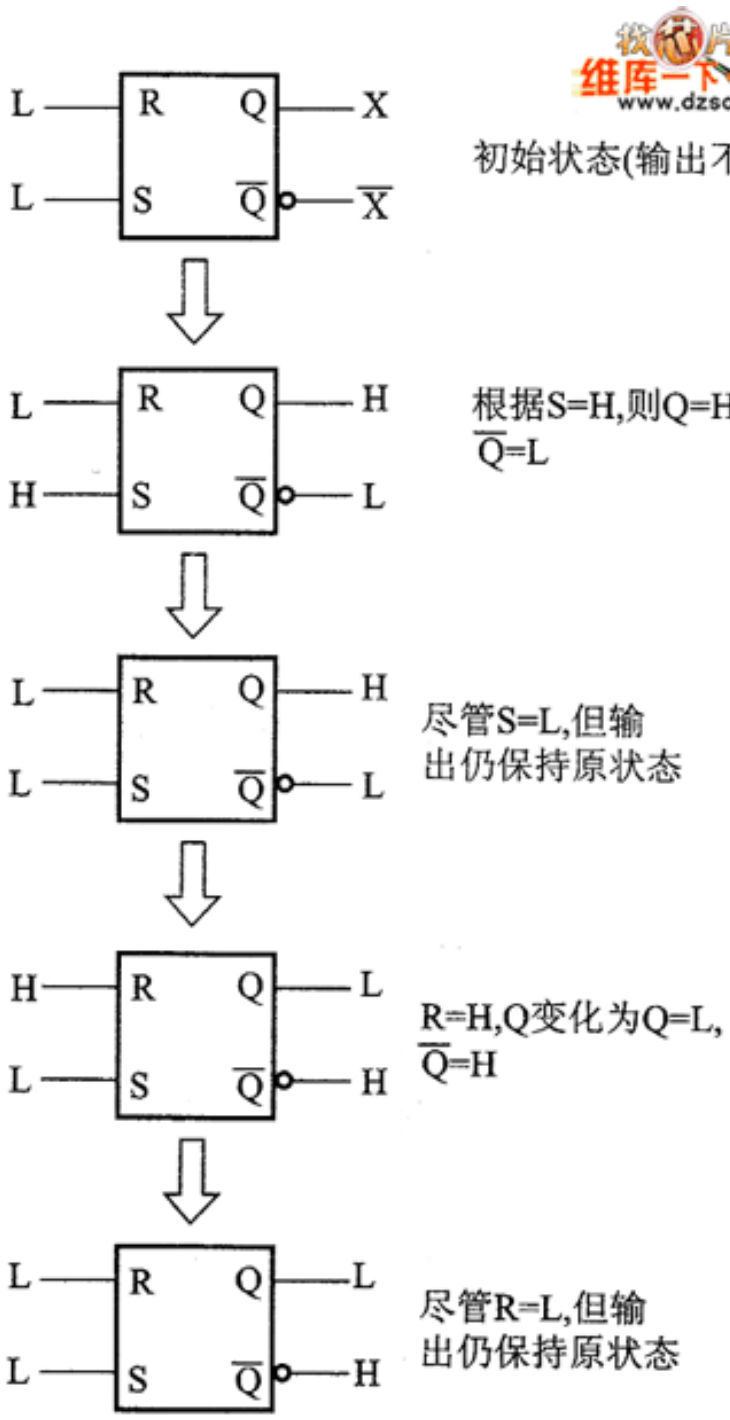


图 2　RS 触发器的操作示例

在这样的状态下，即使 S 恢复为低电平，Q 及/Q 的状态也不变化。由于 Q 为高电平，其下方 NOR 门的输出（/Q）就为低电平。这样再看上面的 NOR 门，由于 R 输人也为低电平，所以输出（Q）也保持高电平。

在此，如果 S 保持低电平，一旦 R 成为高电平，则 Q 将为低电平，下面的 NOR 门的输出将为高电平，触发器就是按照这样的情况运行的，且稳定在反相状态。即使 R 恢复为低电平，也将保持这种状态。

只要按这种情况将多个晶体管集合起来，就可以做成存储器。但在这个电路中，由于晶体管的数目太多，现实中几乎不能应用。图 3 是将 NOR 门以晶体管的形式描述的，每个 NOR 门需要 4 个晶体管。也就是说，构成 RS 触发器需要 8 个晶体管。现实中，利用如下所示的 4 晶体管或者 6 晶体管单元。

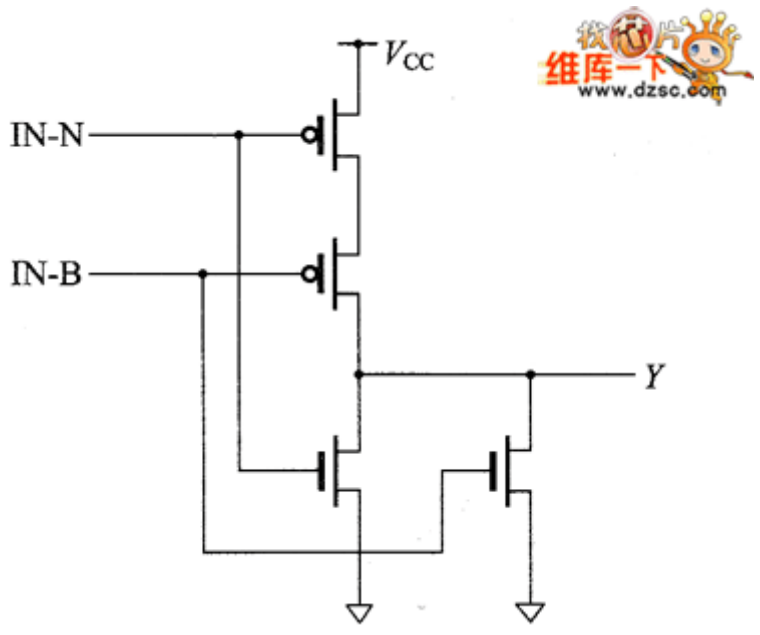


图 3　NOR 门

●4.1.2　4 晶体管单元

4 晶体管的 SRAM 存储器单元的电路构成如图所示，这是利用 2 个晶体管的保持电路以及用于存取的晶体管进行设计的。因为晶体管附加了电阻负载，因而有时也称为高电阻负载式晶体管。

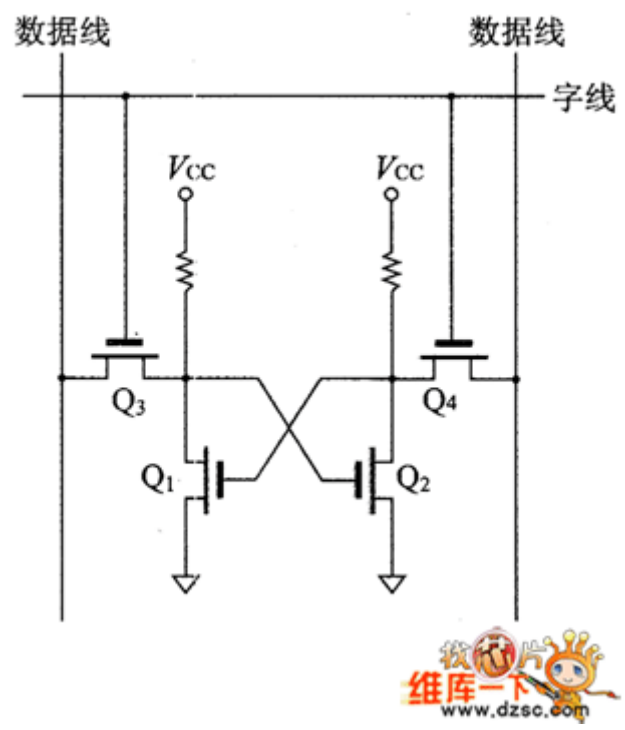


图　　4 晶体管 SRAM 存储器单元

将 Q1 和 Q2 的反相器与各个相对应的门进行交叉连线，就构成了触发器，这是实际的数据存储电路。Q3 和 Q4 是用于读出数据的晶体管开关，如果在字线上被选择，则将通过在数据线上出现触发器状态的形式进行数据读操作。

●4.1.3　6 晶体管单元

4 晶体管式单元在集成方面是有优势的，但在功耗及低电压驱动上却存在问题。为了解决这样的问题，利用P沟道的MOSFET置换 4 晶体管单元的电阻部分，构成 6 晶体管单元。

6 晶体管单元的电路构成如图所示，与 4 晶体管单元相比，单元变得较大。图中 Q1 和 Q5 以及 Q2 和 Q6 分别形成 CMOS 结构，这样，漏电流也变小，就可以将待机电流控制得很小。

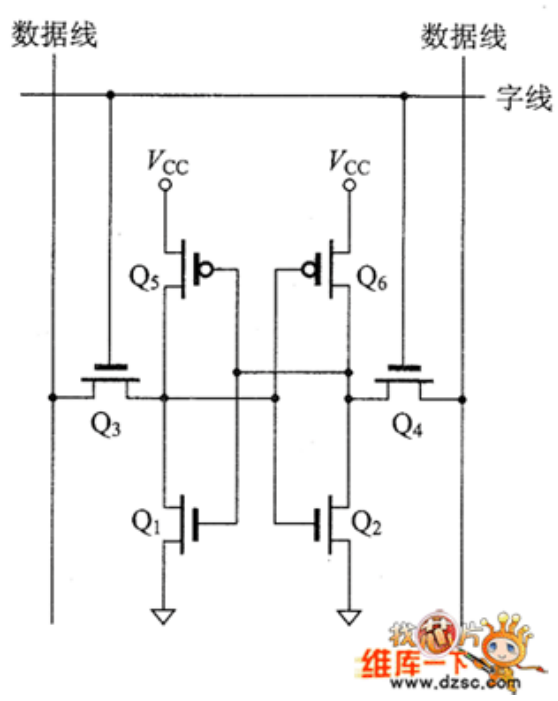


图　　6 晶体管SRAM存储器单元

4.2　SRAM 的分类

SRAM 存在很多的变型，可以说其种类多区分在周围接口上。在利用了 SRAM 单元的器件中，能够举例说明的主要为如下类型。

●4.2.1　异步 SRAM

最一般的就是具备地址总线及数据总线的 SRAM，最具代表性的如图所示。根据用途，可以化分为两种，一种是需要低功耗 / 大容量化的 SRAM；另一种是注重随机存取速度的 SRAM。

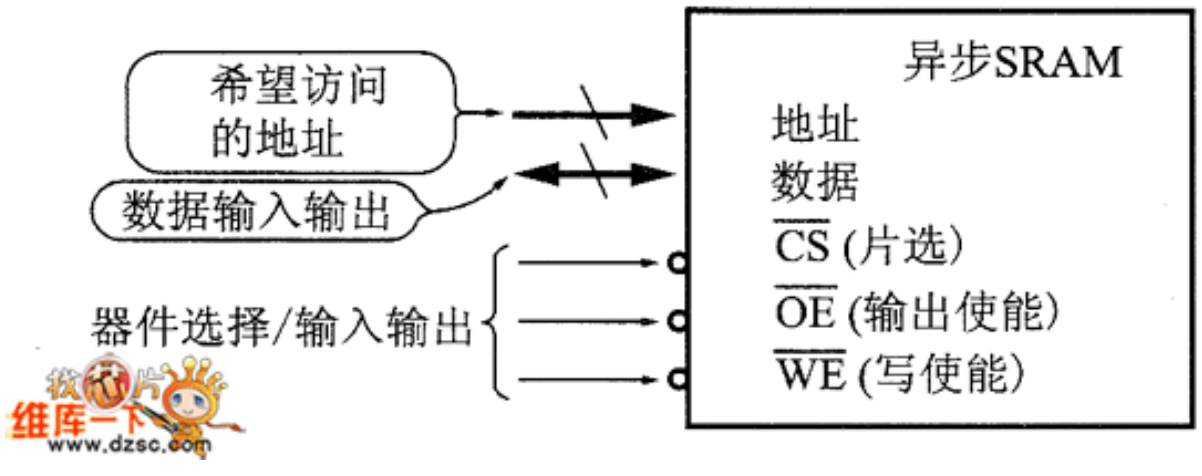


图　　异步 SRAM 的输人输出信号示例

前者或者应用于备用电池和记录各种设置信息中，或者作为组装的微型计算机系统的主存储器来使用。后者最典型的应用大概就是个人计算机的主板中经常使用的高速缓冲存储器的标记 RAM。

●4.2.2　同步 SRAM

同步 SRAM 是与时钟同步运行的意思，一般应用于作为高速缓冲存储器使用的、称为同步管道突发式（Synchronous PipelineBurst）SRAM 中，其他还包括同步突发式（Follow Through，直通）SRAM。同步 SRAM 的信号示例如图所示。

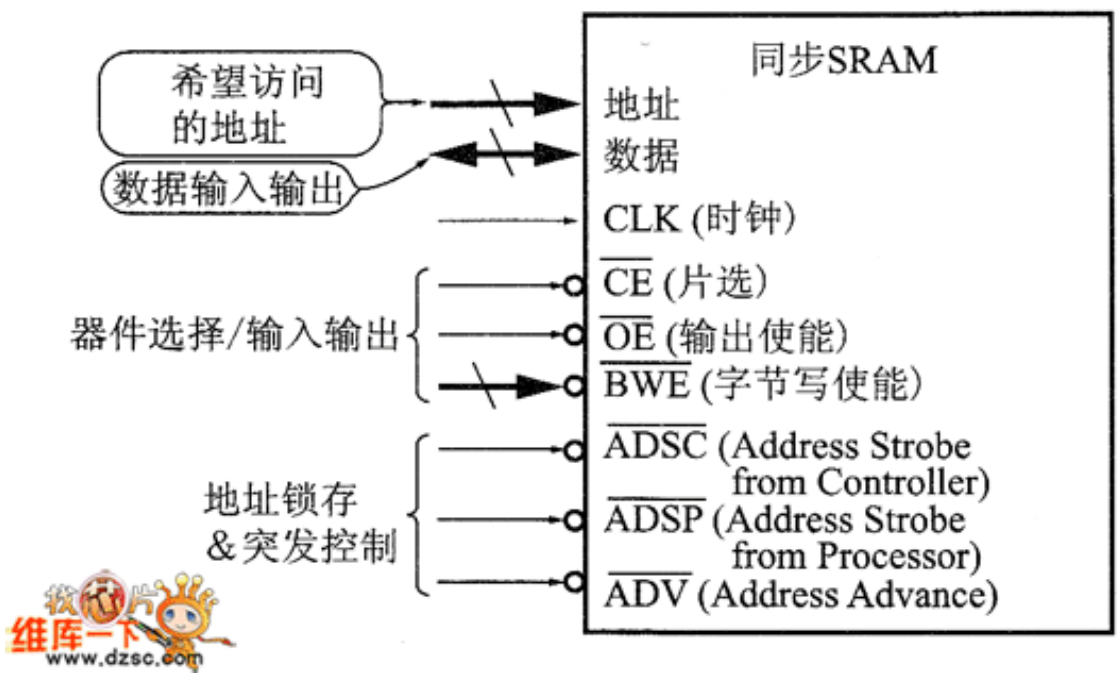


图　　同步 SRAM 的信号示例

无论哪种同步 SRAM，其操作都是与时钟同步进行的。之所以称为“突发”是因为不需要修正地址，可以连续读 / 写连续的区域（一般为 4 次存取大小），所谓的 4 次吻合了普通处理器的突发传输周期。

●4.2.3　双端口 SRAM

普通的存储器器件为单端口，也就是数据的输入输出只利用一个端口，设计了两个输入输出端口的就是双端口 SRAM。虽然还具有扩展系列的 4 端口 SRAM，但双端口 SRAM 已经非常不错了。图 1 表示双端口 SRAM 的信号示例。

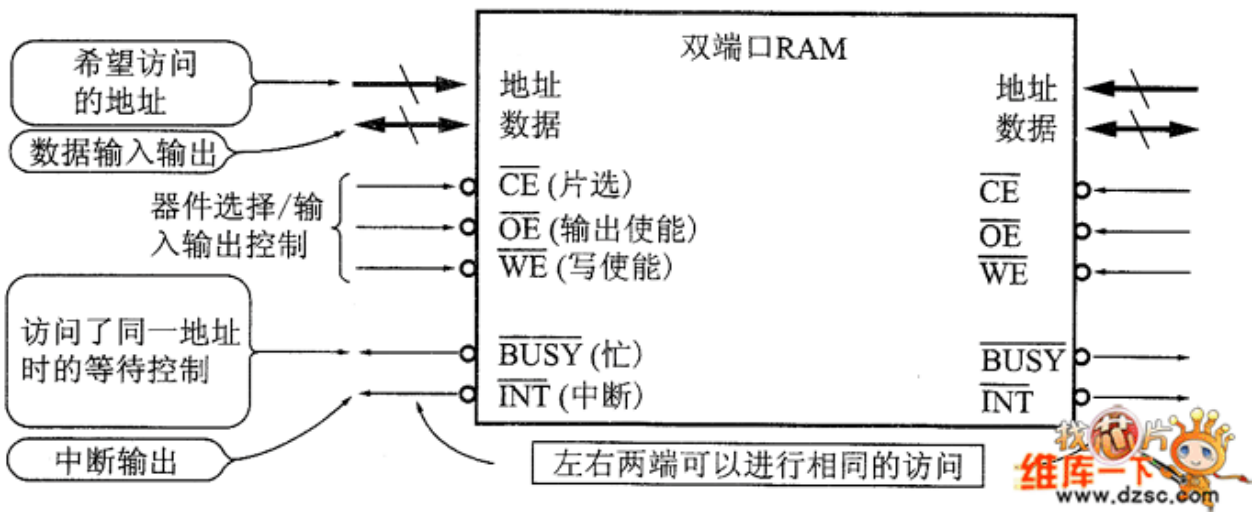


图 1 双端口 SRAM 的信号示例

双端口 SRAM 经常应用于 CPU 与其周边控制器等类似需要直接访问存储器或者需要随机访问缓冲器之类的器件之间进行通信的情况。

在多个 CPU 进行分散处理的情况下，CPU 之间为了传递数据，经常共享同一存储器。但这样的机制如果利用单端口 SRAM 来实现，就成为如图 2 所示的存储器，采用的方法是在双方的 CPU 之间设计一仲裁电路（Arbiter），当要求访问时，打开任意一端与存储器之间的接口就可以进行访问。当双方同时需要访问时，一方需要等待另一方完成访问，这样就需要仲裁电路以及总线缓冲器等。而访问发生冲突、出现问题的可能性较高，因此不能说这是个易于控制的 SRAM。

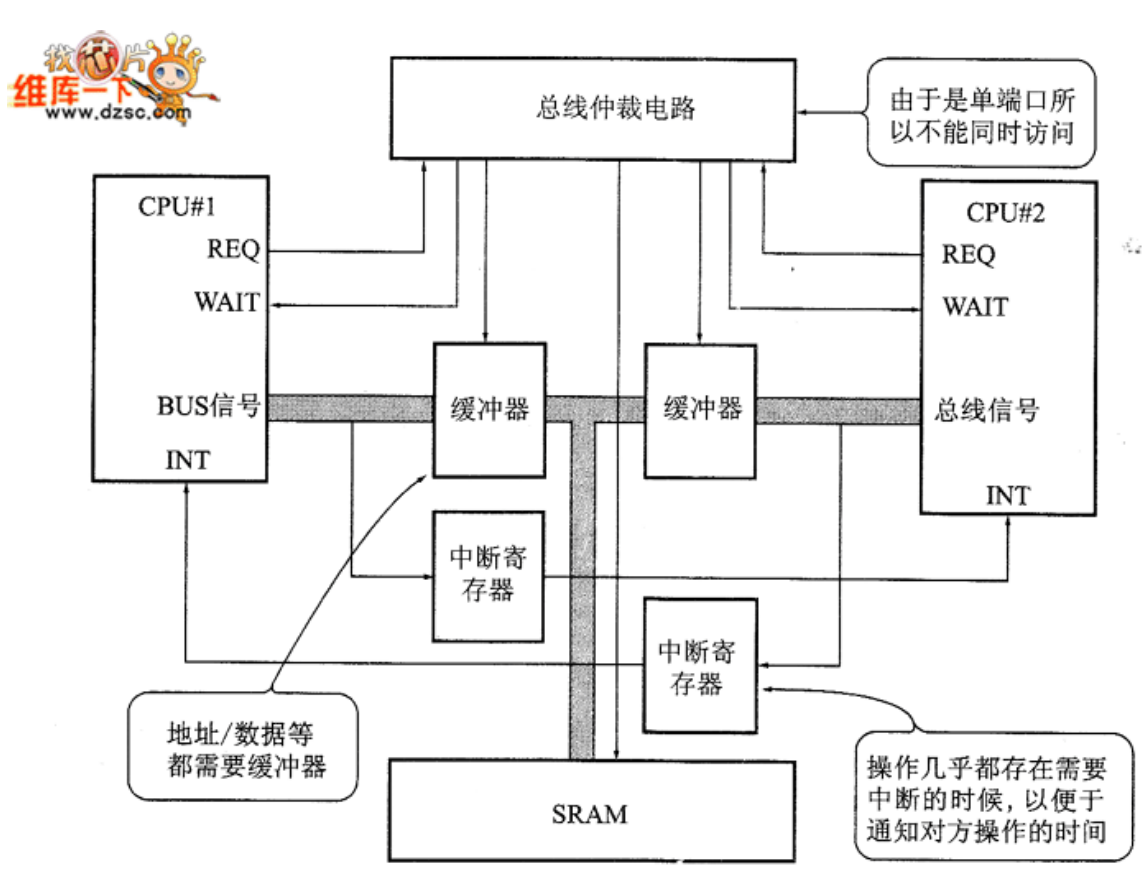


图 2　利用单端口 SRAM 的共享存储器

双端口存储器是为解决上述的问题而制作的。双端口存储器具有两套地址总线、数据总线以及控制信号等，可以从任意一端开始自由访问。与单端口存储器加仲裁电路不同的地方在于即使访问相同的器件，也不一定是对同一地址的访问，因而不必让访问等待。也就是说，即使是对同一存储器器件进行访问，只要是不同的地址，访问就不需要等待，可以在任意时间进行访问。而且在双端口存储器中，作为附加电路，还可以具有用于相互切人对方的电路等。

因此，如图 3 所示的存储器确实简单且性能优秀。

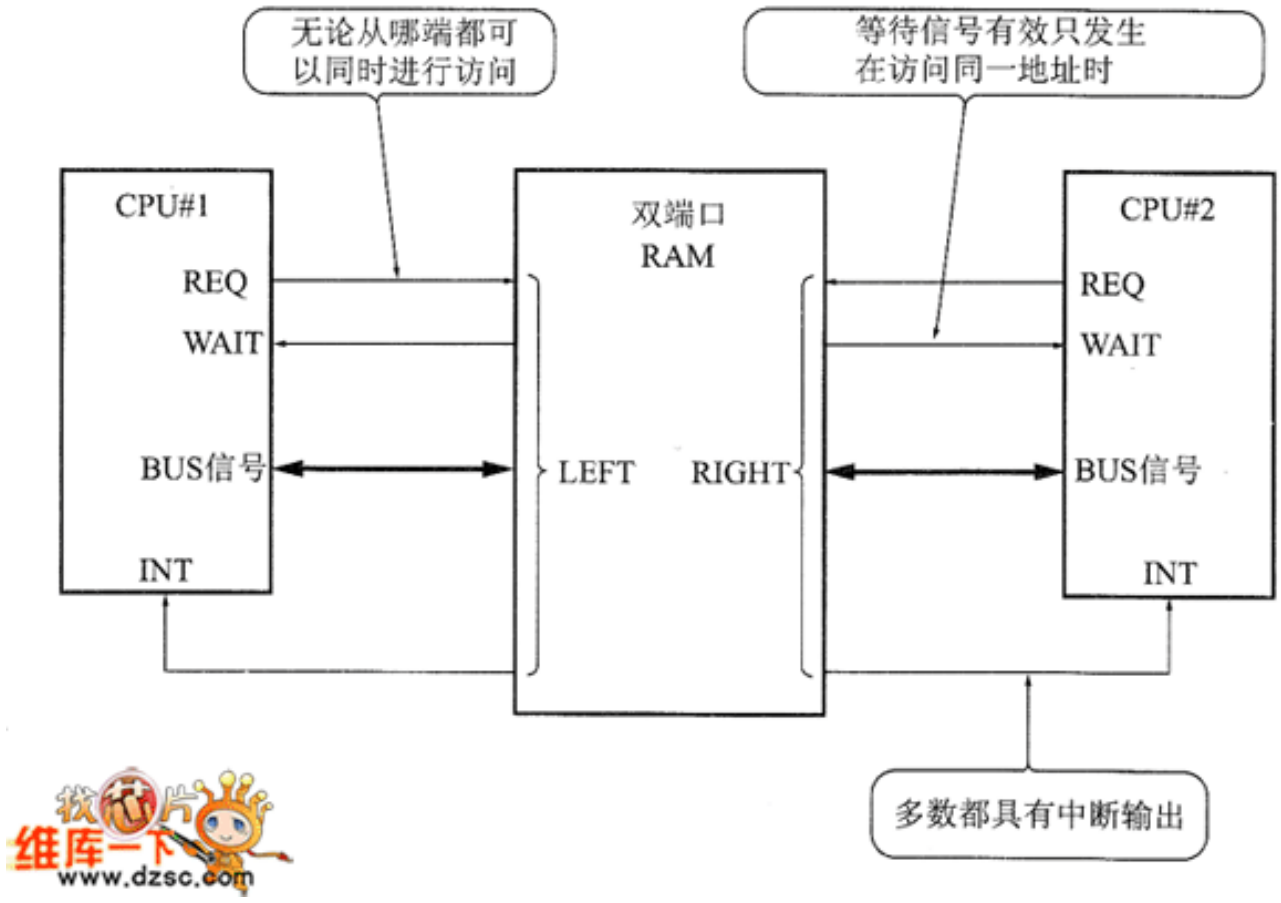


图 3　使用双端口 RAM 的连接

●4.2.4　FIFO

FIFO 是 First In / First－Out 的缩写，是先入先出的意思。FIFO 存储器分为写入专用区和读取专用区。读操作与写操作可以异步进行，写人区上写入的数据按照写入的顺序从读取端的区中读出，类似于吸收写人端与读出端速度差的一种缓冲器。计算机的串口，一般也都具有 FIFO 缓冲器（不是单一的 FIFO 存储器，而是嵌入在设备内部）。

FIFO 存储器的连接模式如图所示。在 FIFO 存储器而不是地址总线上附加了表示内部缓冲器状态（Buffer Full，缓冲器已满；Buffer Empty，缓冲器为空）的状态引脚，连接于 FIFO 的双方利用该状态进行操作的控制。另外，还设计了在接通电源及复位（Reset）或由于操作中的某些异常等原因而重新初始化（无数据状态）FIFO 的复位引脚，这可以说是 FIFO 存储器的特点。

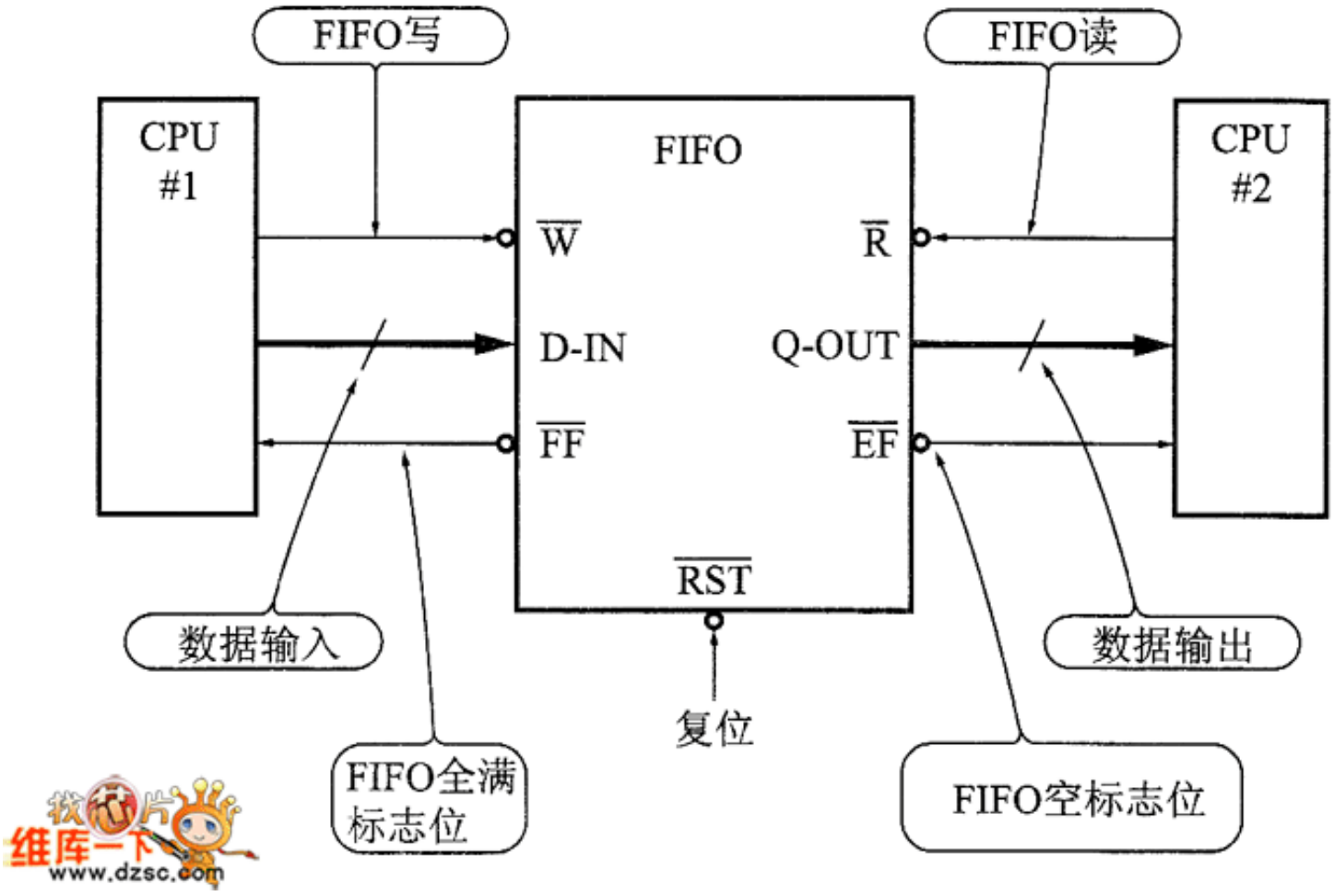


图　FIFO 存储器的连接

4.3　异步 SRAM

异步 SRAM 是 128K×8 位结构的 1M 位 SRAM，我们以 CY62l28 为例进行说明。引脚配置如图所示，这是非常标准的配置，在其他生产商的许多产品中都能见到这种配置。在自制的 SRAM 主板上就使用了现成的 ISSI 引脚兼容产品。

● 4.3.1　异步 SRAM 的信号

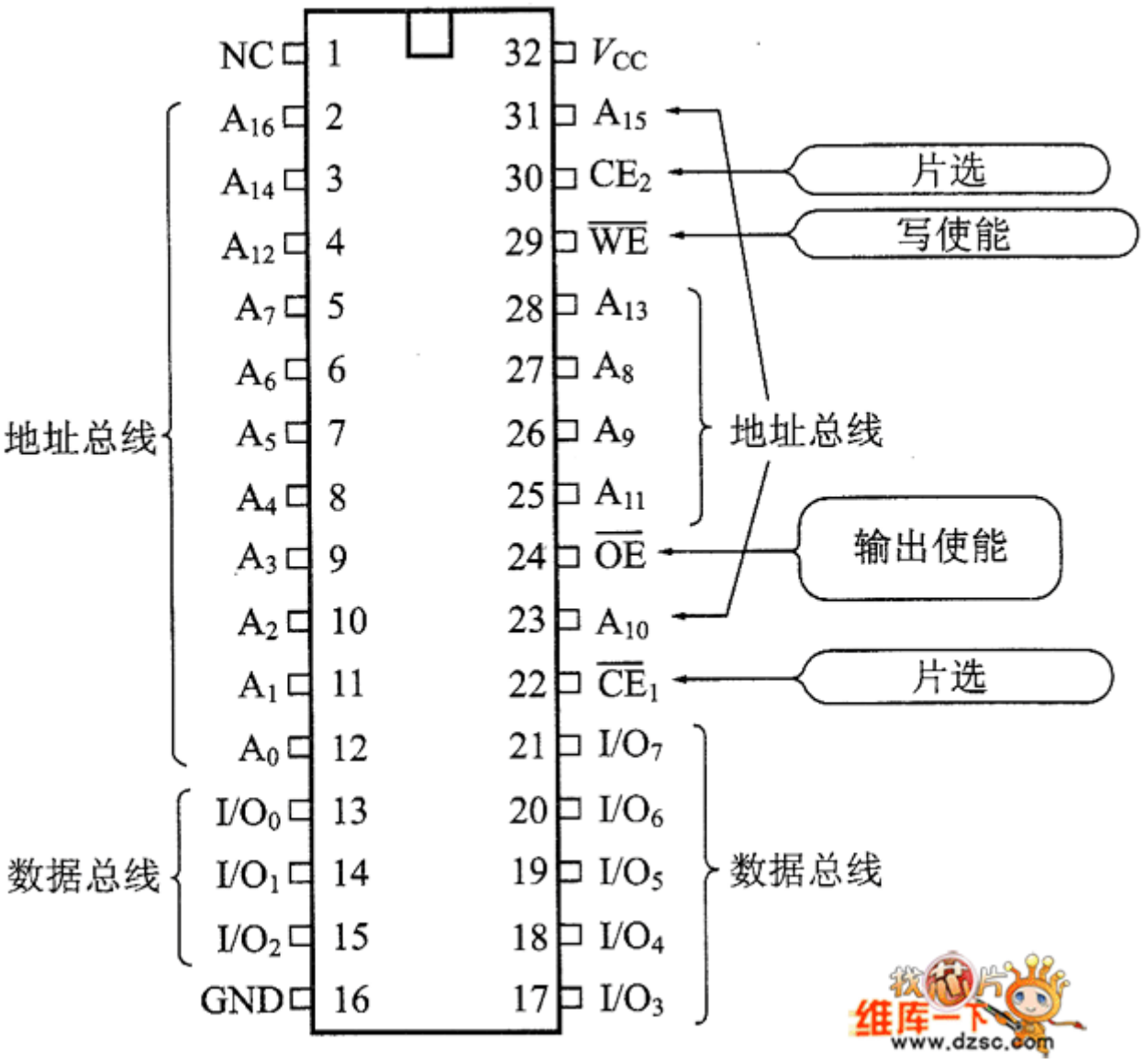


图　CY62128 的引脚配置

异步 SRAM 的各个引脚所表示的意思如下所述。各个控制输人与操作状态的关系如表所示。

表　SRAM 的控制输入与操作

$\overline{CE_1}$	CE_2	$\overline{OE}$	$\overline{WE}$	$I/O_0 \sim I/O_7$	操作模式
“H”	“X”	“X”	“X”	高阻抗	断电状态
“X”	“L”	“X”	“X”	高阻抗	断电状态
“L”	“H”	“L”	“H”	数据输出	读操作
“L”	“H”	“X”	“L”	数据输入	写操作
“L”	“H”	“H”	“H”	高阻抗	选择状态(输出禁止)

▲　A0～A16（地址）

用于指定希望访问的地址。由于是以 128K×8 位的结构作为对象的，所以地址线具有 17 根。SRAM 不是通过特殊的存储器写人器写入的，而且对于地址也没有类似 DRAM 的刷新功能那样的操作。因此，虽然可以更换地址，例如将 CPU 地址的最低位加到 SRAM 的 A16 那样没有什么问题，但一般是将 A0 作为最低位、A16 作为最高位来使用的。

▲　I／O0～I／O7（数据）

这是数据输入输出引脚，可双向使用。与地址引脚的更换相同，虽然可以更换数据引脚来使用，但一般将 I／O0 作为最低位、将 I/O7 作为最高位使用。

▲　/CE1、CE2（Chip Enable，芯片使能）

这是器件的选择信号，虽然存在/CE1 和 CE2 两个信号，但它们是以 AND 条件选择的。也就是说，只有当/CE1 为低电平且 CE2 为高电平时，器件才处于选择状态。如果不在选择状态，则其他输人引脚的状态将全部被忽略。存在极性不同的 Chip Enable 信号是为了处理方便。例如，电池备份时，因为主电源本身下降，与主电源连接的电路中就不能形成高电平，但是通过简单的下拉电阻，可以相对简单地保持低电平。

需要注意的是，在以相同引脚排列形式、且 CE2 引脚处于高位的 4M 位容量的 CY62148 中，因为 CE2 引脚被作为地址引脚使用了，因而 Chip Enable 就只能利用/CE1。

如果将 SRAM 置于低功耗的待机状态，则需要注意/CE1 及 CE2 的电压。与在 CMOS 电平的使用时相比，在使用 TTL 电平输人（高电平为 2.4 伏；低电平为 0.8 伏）时，损耗电流变得相当的大。例如，对于 CY62128-55 来讲，如果是 CMOS 电平，则典型值为 0.4μA，而 TTL 电平就为 25mA，相差一个数量级。因此，进行普通的电池备份时，将/CE1、CE2 设计在 CMOS 栅极进行驱动，以便确保其为高电平。

▲　/OE（Output Enable，输出使能）

这是打开 SRAM 数据输出缓冲器的信号。读操作时，利用片选后的状态（/CE1＝低电平；CE2＝高电平）指定地址，如果/OE 为低电平，则在 I／O 引脚上将出现存储器的内容，但必须事先将/WE 设为高电平。

▲　/WE（Write Enable。写使能）

这是向 SRAM 写人的信号。在/WE 上升时刻，数据被写入到存储器中。当/WE 和/OE 双方都为低电平时，/WE 优先进行操作。也就是说，在/OE 为低电平、保持向 I／O 引脚输出数据的状态中，如果将/WE 设为低电平，I／O 引脚将转换为输人模式。

●4.3.2　异步 SRAM 的基本操作

异步 SRAM 正如其名称，不是与特定的时钟信号同步运行，而是根据输人信号的状态运行的。因为没有信号表示读取时已确定了有效数据，也没有信号表示写入时已接收到数据，所以，需要获取制造商的数据手册，根据时序图，按“应该已读出有效数据”及“应该能接收数据”这样的条件，进行存储器的设计。

▲　读操作：/OE 读控制

异步 SRAM 的基本读操作如图 1 所示。首先指定地址，然后使 CE2＝/WE＝高电平，/CE1＝/OE＝低电平，此时将在 I / O 引脚出现数据。如果保持该状态而改变地址，则将出现新地址的数据。另外，如果/CE1，CE2,/WE 和/OE 没有满足读状态的条件，则 SRAM 中止驱动 I / O 引脚，成高阻抗。

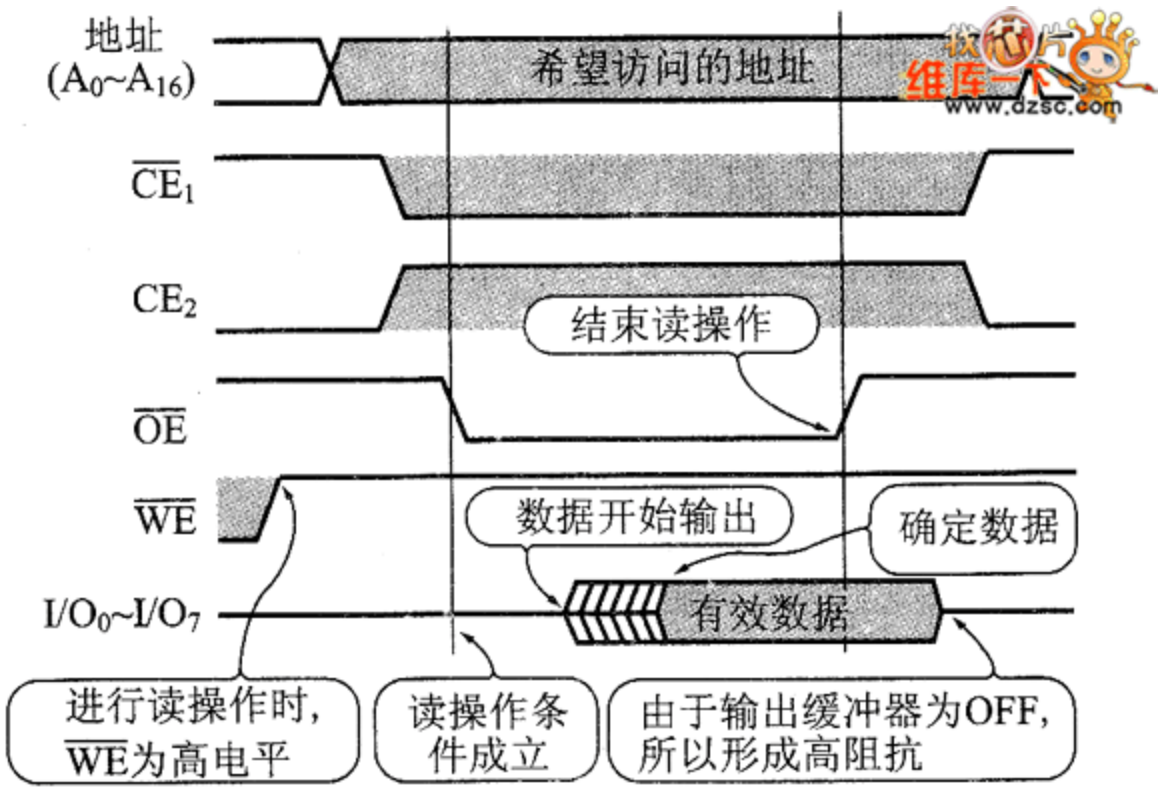


图 1　　异步 SRAM 的读操作

读操作时，使/CE1，CE2，/WE,/OE 等保持读状态，也允许改变地址（也就是保持存取的状态，只改变地址，读取不同地址的数据）。但是一部分高速 SRAM 也存在这样的情况，即当器件处于选择状态（CE 有效）时，如果改变其地址，则设备判断发生了误操作，所以事先需要确认是否允许这样的应用。

▲　写操作 1：/WE 写控制

异步 SRAM 的基本写操作如图 2 所示。首先指定地址，如果 CE2＝高电平，/CE1＝低电平，则器件处于选择状态。只要/OE 一直有效（低电平），则在此临时输出数据。但因为/WE 具有优先权，所以一

旦/WE 有效，则 I／O 引脚变为高阻抗状态。进行写人操作的地址必须在/WE 下降之前就要确定，数据的写人操作是在/WE 的上升沿进行的。

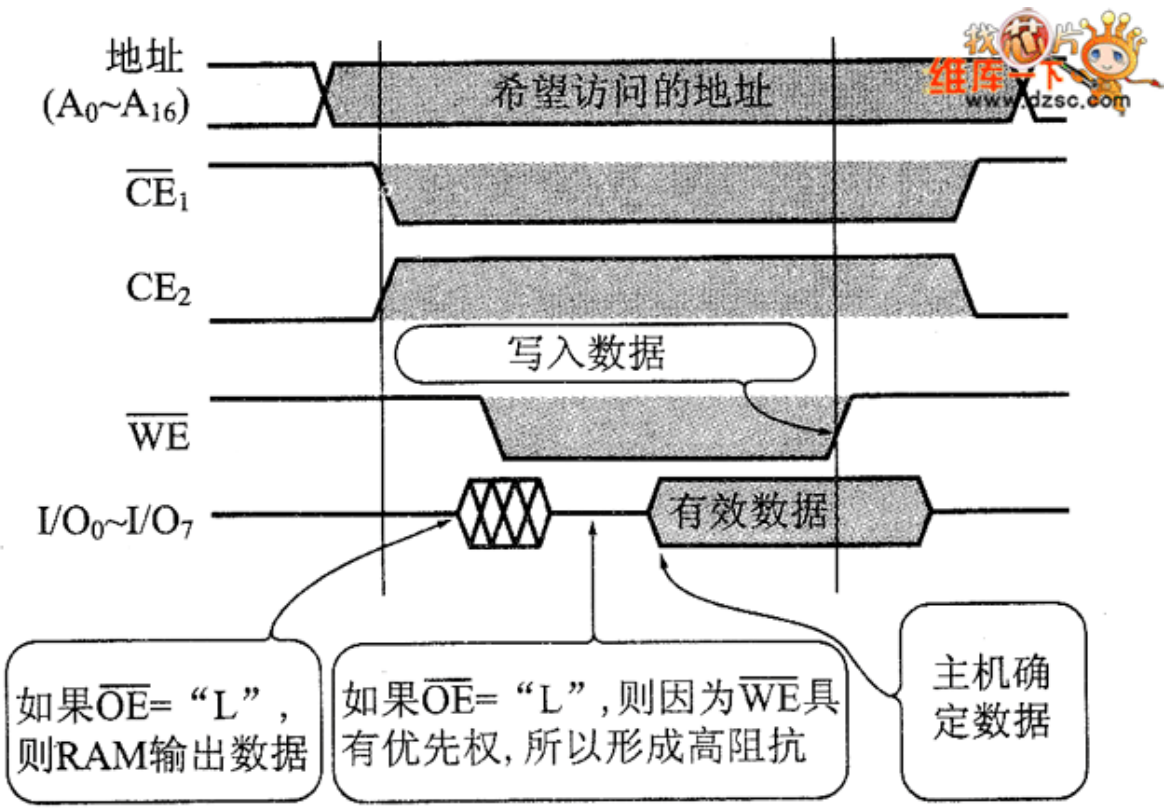


图 2　异步 SRAM 的写操作 1（/WE 写控制）

在首先进行读操作、然后修正读出的数据、最后再向同一地址写人（读－修改一写）的时候，如果保持/OE 有效，则输出数据的操作是一种方便的操作。

▲　写操作 2：CE 写控制

CE 写控制的操作如图 3 所示。CE 写控制在/WE 已经有效的状态下，利用/CE1 和 CE2 写入数据。因为/WE 已经是有效的，因此器件变为选择状态的同时也变为写状态。

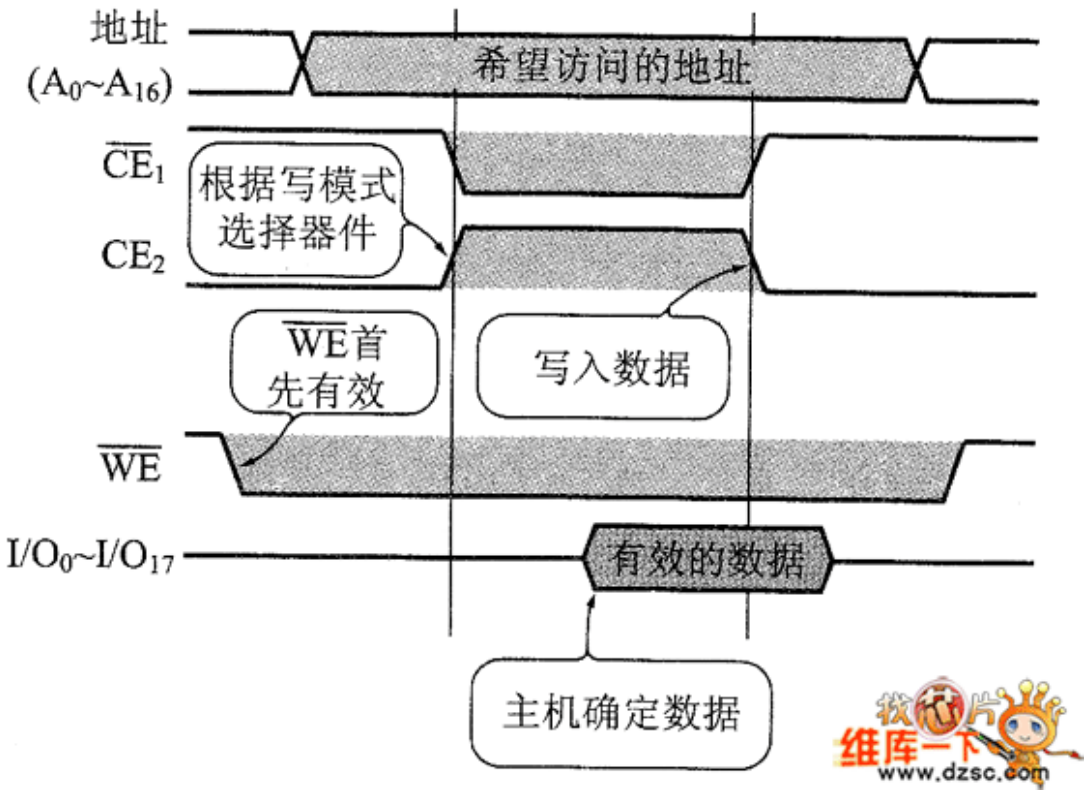


图 3　异步 SRAM 的写操作 2（CE 写控制）

●4.3.3　时序的解析

在 CY62128 数据手册的时序图中，/OE 读控制操作、CE 写控制操作以及/WE 写控制操作的时序分别如图 1 至图 3 所示。其各种时序规定如表所示。

★读操作的时序规定

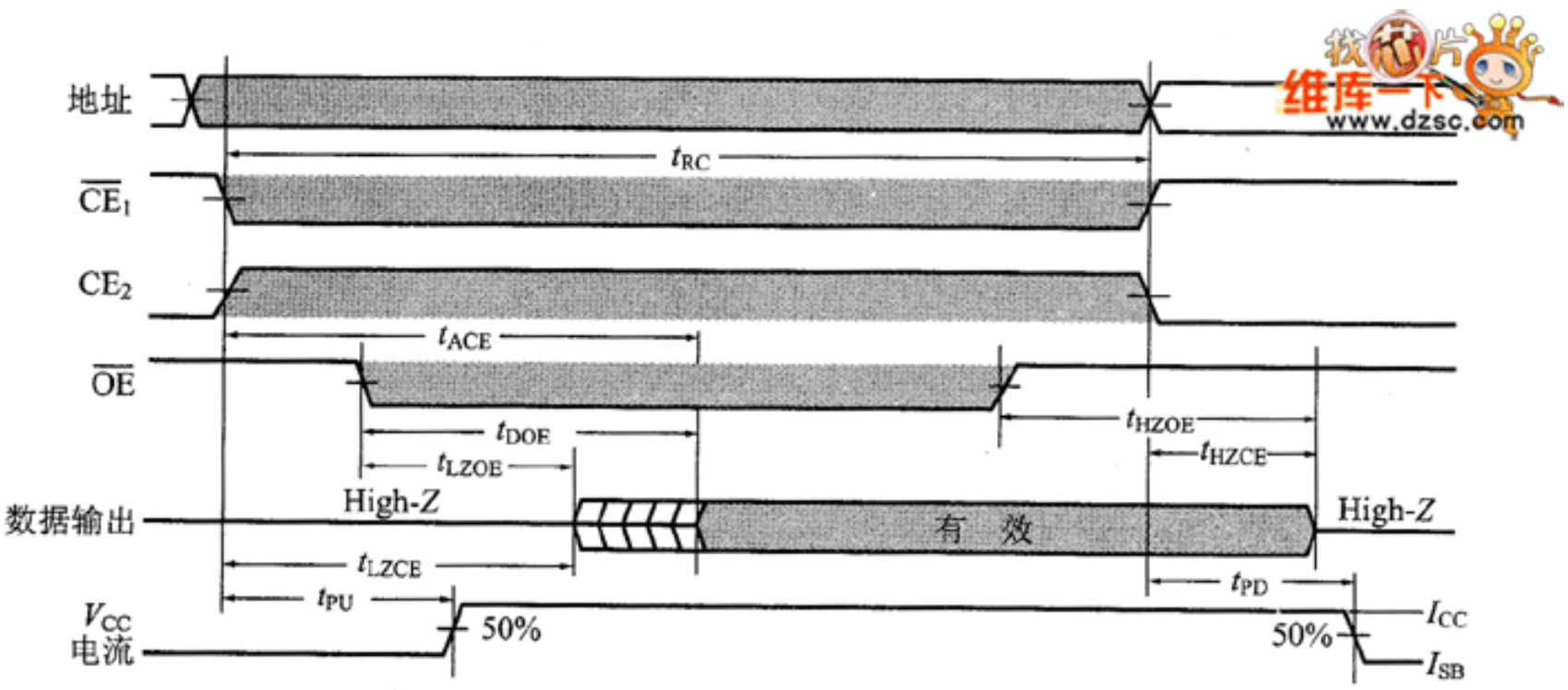


图 1　/OE 读控制操作的时序

读操作的时序规定如下所述。

▲　tAA（Address to Data Valid）

该时间值在图中未出现，它是指从指定地址到确定数据所需要的时间。CY62128 中的 tAA 与下一个 tACE 相同，可以作为一个值来进行处理。

▲　tACE

tACE 是来自/CE1、CE2 的存取时间。从/CE1 和 CE2 全都为有效的状态开始，经过 tACE 时间，需要确定 I / O 引脚的数据。在读时序中，需要注意存取时间存在两个，除 tACE 之外还包括下一个将要说明的来自/OE 的存取时间（tDOE），数据的确定要遵循 tACE 和 tDOE 中较迟的那个时序。

例如，CY62128-55 的 tACE 为 55ns，tDOE 为 20ns，在地址被指定的同时，当/CE1，CE2 和/OE 全部同时有效时，由 tACE 的 55ns 确定时序。如果地址及/CE1、CE2 在/OE 前 35ns 以上被确定，则/OE 有效 20ns 以后，数据被确定。

▲ tDOE

这是从/OE 有效到确定数据所需要的时间。曾在讲解 tACE 时接触过，实际数据的输出时序是由 tAA、tACE 和 tDOE 之中最慢的一个时序来决定的。

▲ tLZOE

从/OE 有效到确定数据所需要的时间是 tDOE，但从/OE 有效到 I / O 被开始驱动的时间是 tLZOE。因为 CY62128-55 的 tLZOE 为 0ns（min），所以一旦/OE 有效，则可能会立即输出某些数据。

▲ tLZCE

tLZOE 相同,这是从/CE1 及 CE2 开始有效到 I / O 引脚被开始驱动的时间。CY62128 的该时间为 5ns。

▲ tHZOE

如果/OE 无效，则输出缓冲器变为禁止，I / O 引脚为高阻抗状态，tHZOE 就是成为这种状态所需要的时间。因为 CY62128-55 的该时间为 20ns，因此，即使/OE 无效，在 20ns 左右的时间内，I / O 引脚也仍然处于被驱动的状态。

▲ tHZCE

与 tHZOE 相同，如果使/CE1、CE2 无效，I / O 引脚也将为高阻抗状态，这一过程所需要的时间就是 tHZCE。CY62128－55 的 tHZCE 为 20ns（max），与 tHZOE 值相同。

▲ tRC

这是对读操作一个周期的时间规定。由于 tAA 及 tACE 值为最小值，所以，一般认为在实际的设计中不会低于该 tRC 时间，但是需要注意不要产生未满足 tRC 规定的读周期。

▲ tPU / tPD

如果/CE1、CE2 一起有效后处于选择状态，则 SRAM 将处于操作状态，损耗电流变大（加电）；反之，如果/CE1、CE2 一起无效，则成为待机状态，损耗电流变小（断电）。tPU / tPD 表示该加电 / 断电的时间，tPU 最小为 0，tPD 最大为 55ns。

在处于选择状态的同时开始有较大的电流，即使结束选择状态，55ns 以内也会持续消耗电流。所

以在设计电源切换电路时需要对此加以注意。

还有一点需要注意，尽管在图中没有表示，但如前所述，/CE1 和 CE2 的电压水平会使损耗电流发生较大的改变。

★CE 写控制操作的时序规定

CE 写控制操作的规定比读操作的规定要稍微麻烦些，读操作只是等待确定各种信号，而写操作则与之不同，如果不满足地址及数据的建立 / 保持时间以及写操作的时间等，SRAM 将不能正确接收地址及数据。

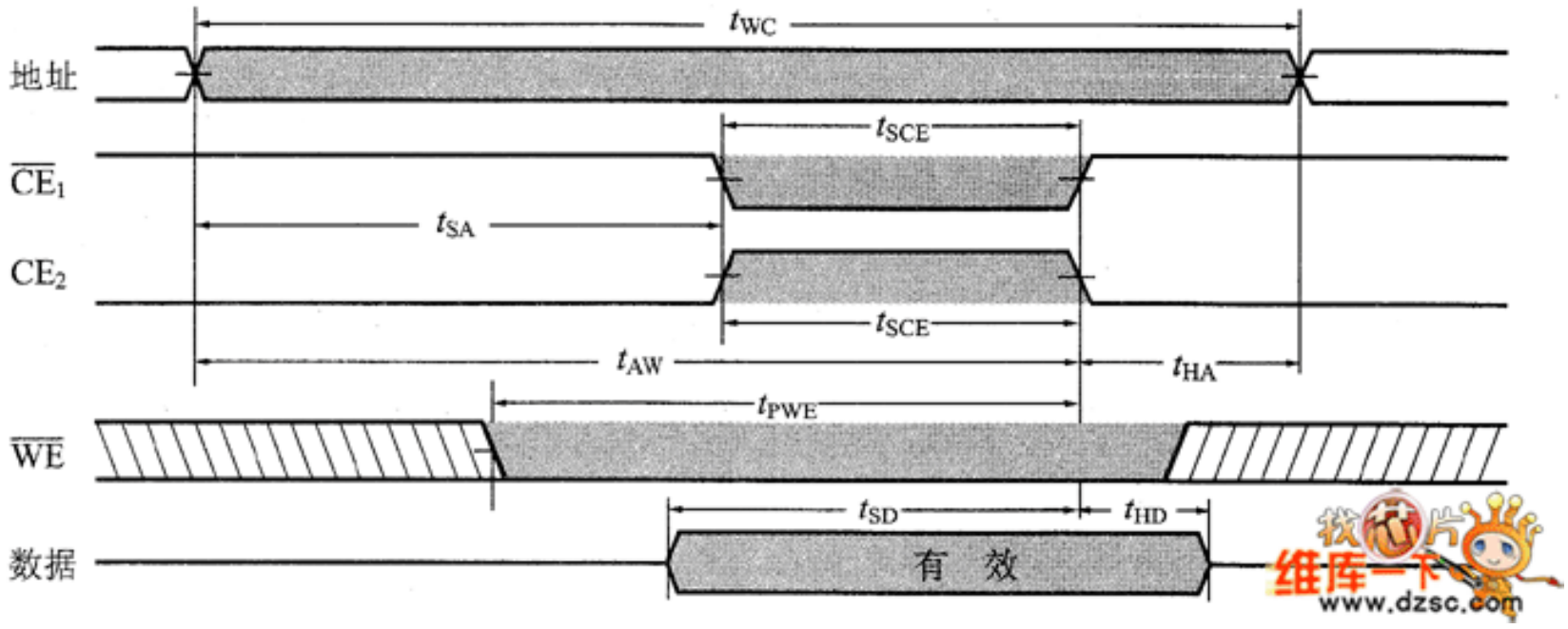


图 2　CE 写控制操作的时序

▲　tSA

与读操作时不同，当进行写操作时，在/CE1、CE2 有效的过程中，地址必须被指定，所需要的时间就是 tSA。CY62128 的 tSA 最小为 0，所以不能反向运行，也就是说在/CE1、CE2 有效的瞬间之后，地址不改变即可。

▲　tHA

这是写操作时从/CE1、CE2 无效到可以让地址改变成其他状态所需要的时间。也是因为 CY62128 的该时间为 0，因而不可反向运行。

▲　tSD / tHD

在/CE1、CE2 无论哪个信号无效的过程中，输入到 I / O 引脚的数据被写人到存储器内部，此时.在写人之前所需要的确定数据的时间就是 tSD（数据建立时间）；而在信号无效之后数据所必须保持的

时间就是 tHD（数据保持时间）。CY62128-55 的这两个时间分别为 25ns 和 0ns，也就是说，CY62128-55 需要在/CE1 或 CE2 无效的 25ns 之前确定数据，需要保持数据一直到信号无效。

▲ tSCE

tSCE 是规定从/CE1、CE2 双方有效之后到任意一方无效的时间。如果不能满足该时间规定，那么向 SRAM 内部的存储器单元的写人操作可能不能被正常执行。CY62128-55 的 tSCE 时间为 45ns。

▲ tAW

这是针对写操作完成（/CE1、CE2 无论那个信号无效）地址的建立时间。CY62128 的这个时间需要 45ns，但稍微观察一下就可明白，tSCE 与 tAW 是相同的值，而且 tSA 虽然可以为 0，但由于在实际的电路中采用不可反向运行的机器，因此只要正确设计，这个时间规定是不会出现问题的。

▲ tPWE

这是/WE 信号有效到/CE1、CE2 任意一个无效的时间。从时序规定上看，是与 tSCE 相同的值，但既然是希望利用“CE 写控制”，那么/WE 信号的有效时间就要设计得比 tSCE 时间长，这样才不会出现问题。

★/WE 写控制操作的时序规定

/WE 写控制的时序表示符号本身与 CE 写控制是相同的，但在时序规定上，写操作的结束就是/WE 的上升。

tHZOE 表示从/OE 无效到 I / O 引脚成为高阻抗状态所需要的时间，正如在读操作中所描述的那样。

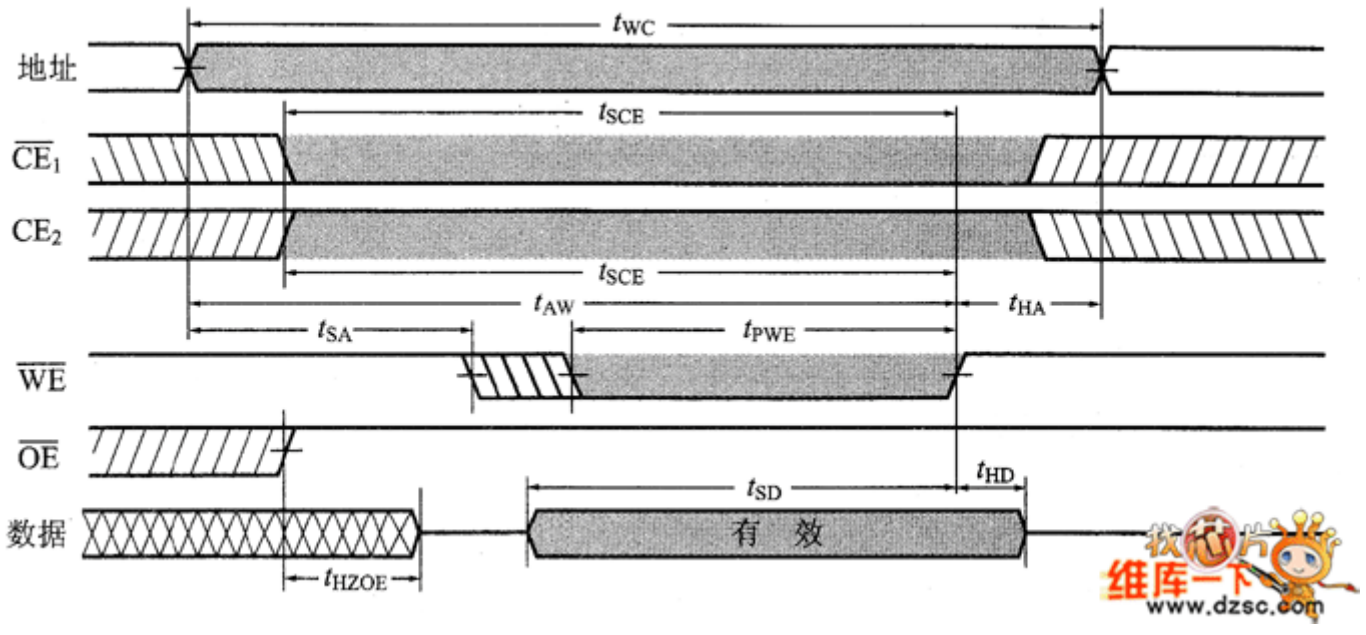


图 3 /WE 写控制操作的时序

表　时序规定

符　号	条　　件	62128-55		6218-07		单位
		min.	max.	min.	max.	
读周期						
t_{RC}	Read Cycle Time	55		70		ns
t_{AA}	Address to Data Valid		55		70	ns
t_{OHA}	Data Hold from Address Change	5		5		ns
t_{ACE}	$\overline{CE_1}$ LOW to Data Valid, CE_2 HIGH to Data Valid		55		70	ns
t_{DOE}	OE LOW to Data Valid		20		35	ns
t_{ZOE}	OE LOW to LOW Z	0		0		ns
t_{HZOE}	OE HITH to High Z		20		25	ns
t_{LZCE}	$\overline{CE_1}$ LOW to Low Z, CE_2 HIGH to Low Z	5		5		ns
t_{HZCE}	$\overline{CE_1}$ HITH to Hight Z, CE_2 LOW to High Z		20		25	ns
t_{PU}	$\overline{CE_1}$ LOW to Power-Up, $\overline{CE_1}$ HIGH to Power-Up	0		0		ns
t_{PD}	$\overline{CE_1}$ HIGH to Power-Down, CE_2 LOW to Power-Down		55		70	ns
写周期						
t_{WC}	Write Cycle Time	55		70		ns
t_{SCE}	$\overline{CE_1}$ LOW to Write End, CE_2 HIGH to Write End	45		60		ns
t_{AW}	Address Set-Up to Write End	45		60		ns
t_{HA}	Address Hold from Write End	0		0		ns
t_{SA}	Address Set-Up to Write Start	0		0		ns
t_{PWE}	$\overline{WE}$ Pulse Width	45		50		ns
t_{SD}	Data Set-Up to Write End	25		30		ns
t_{HD}	Data Hold from Write End	0		0		ns
t_{LZWE}	$\overline{WE}$ HIGH to LOW Z	5		5		ns
t_{HZWE}	$\overline{WE}$ LOW to High Z		20		25	ns

4.4　同步 SRAM

同步 SRAM 意如字义，是与时钟同步运行的 SRAM。由于地址的提取以及数据的输出全部是与时钟同步，所以没有必要像异步 SRAM 那样必须分别考虑基于各种信号的时序，这是其最大的优势。

同步这样的名字容易让人产生误解的是，容易想象成如图所示的、在普通的异步 SRAM 外部添加时钟同步电路。这种情况是为了确保地址及数据等的建立 / 保持时间，以一个时钟单位进行调整。

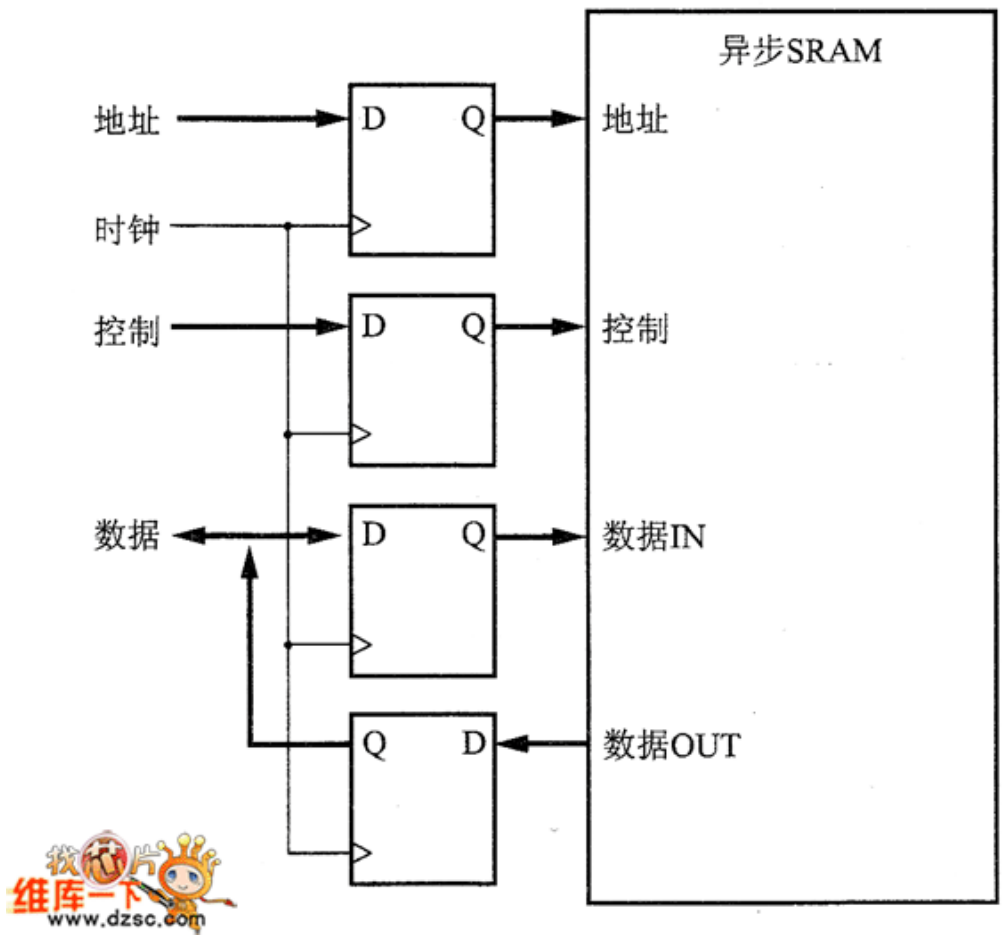


图　这也是同步 SRAM

事实上，这是与异步 SRAM 相同的。从图中就可看出，异步 SRAM 比较麻烦的是必须遵守各处的时序规定。如果总线的操作时钟缩短为 66MHz（一周期 16 ns）及 100MHz（一周期 10ns），则调整时间本身就是一大技术难点。

同步 SRAM 比这种异步 SRAM 的同步化又先进了一步，它采用这样的方法，即在第 1 个时钟中接收地址及数据（写操作时）以及指令类，在第 2 个时钟以后根据所给予的指示、进而根据所给予的信号的指示进行运行，确保操作以时钟单位进行。例如，如果是读操作，则锁存地址及指令以后，决定在第几个时钟中读出数据。这样，只要存储器以及主机方面都与时钟同步进行操作即可，在设计上也非常简便易行。

●4.4.1　同步管道突发式 SRAM

同步管道突发式 SRAM 大体的结构如图所示。这种类型的 SRAM 需要考虑适应 CPU 的突发传输模式，图中“突发控制”部分就是为此设计的电路。

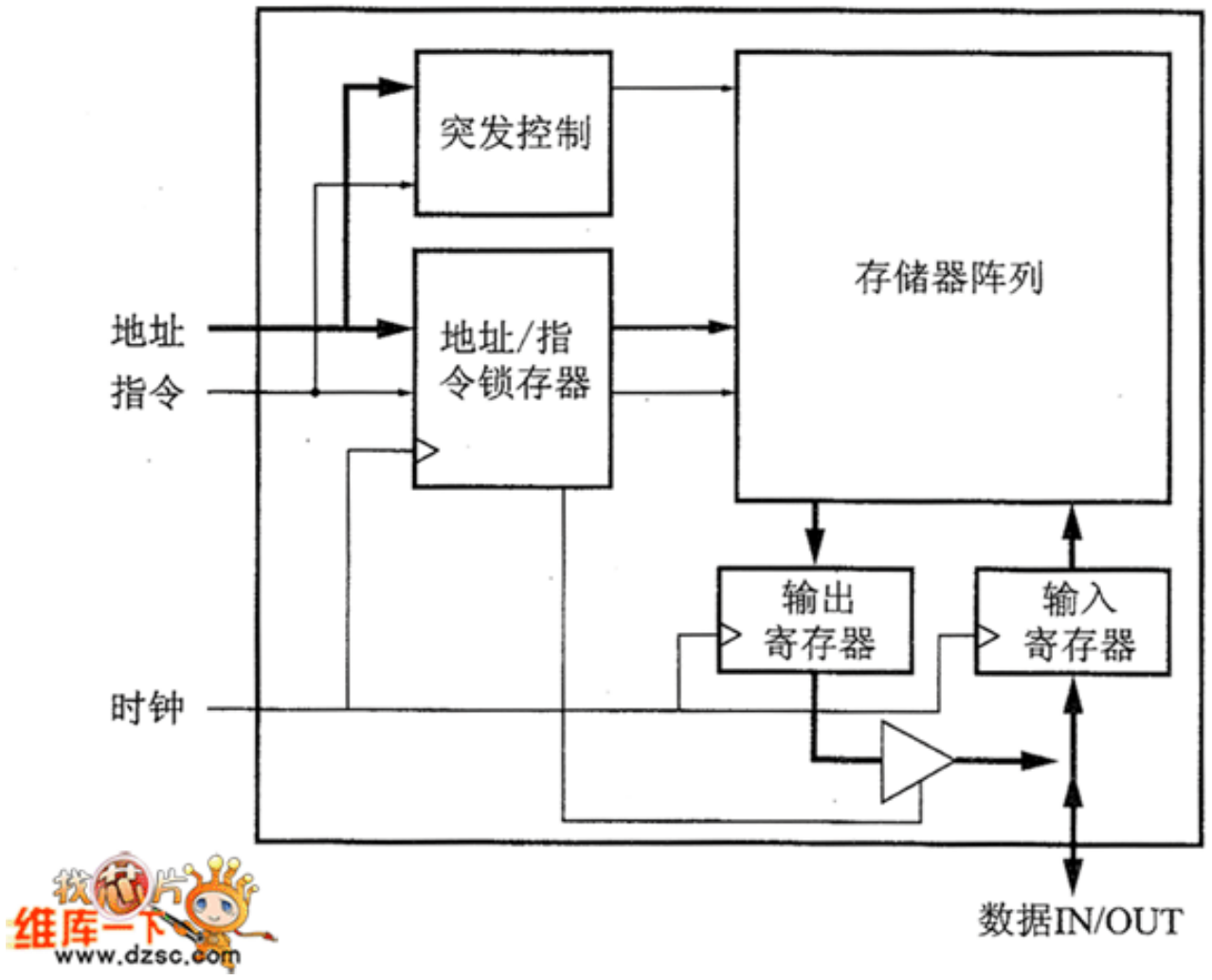


图　同步管道突发式 SRAM 的内部框图

现在的 CPU 都在内部装载高速缓冲存储器，以便于提高对连续区域的存取效率。即使针对外部总线，也设计了类似提高对连续区域存取效率的总线周期，该总线周期称为突发传输周期。

在突发传输周期中，虽然通常是将连续 4 字的数据汇总后进行传输，但只要知道开始地址，之后就可确定存取地址的顺序。因此，像通常存储器的存取方式那样，不必输出每次的地址，只要输出最初的地址，之后与时钟同步，就可连续输出数据，这样就可以实现高速化的目标。

对此，在同步 SRAM 上，对应于该突发传输周期，只要给予最初的地址，就可以实现通过自身自动生成下一个地址，然后进行数据的读 / 写操作。之所以称为“同步突发”及“同步管道突发”，是为了表示这是对应于突发传输操作的。

在个人计算机的世界中，同步管道突发式 SRAM 在奔腾（Pentium）类处理器成为主流之前经常作为二级高速缓冲存储器使用。近期的 CPU 为了提高性能都内置了二级高速缓冲存储器。在外部即使附加高速缓冲存储器作为三级缓存，其性能也未提高多少。因此，在个人计算机的母板上已经很少能看到同步管道突发式 SRAM。

●4.4.2　实际的同步管道突发式 SRAM

下面我们看一下实际的同步管道突发式 SRAM，这次我们作为实例的产品是 Cypress 公司的 128K×36 位的 CY7C1347B。之所以采用 36 位而不是 32 位，是因为考虑到每隔 8 位（一个字节）能进行验证的情况。

CY7C1347B 的内部框图如图 1 所示，信号种类如图 2 所示。这些信号除了以一字节为单位进行写人操作的/BWn 信号以外，还包括进行 32 位整体写入操作的/GW。在 CPU 的突发周期中，当可以一次性更新 1 字大小（36 位）的数据时使用/GW；当从外部更新 1 字节或 2 字节大小的数据时使用/BWn 信号，这样就可以只更新相应的字节数据。另外，用于地址锁存的信号包括/ADSC 和/ADSP 两个信号，/ADSC 用于来自缓存控制器的存取；/ADSP 用于来自处理器的存取。/ADSP 与/ADSC 在写存取时的处理上存在若干不同，这将在以后进行说明。

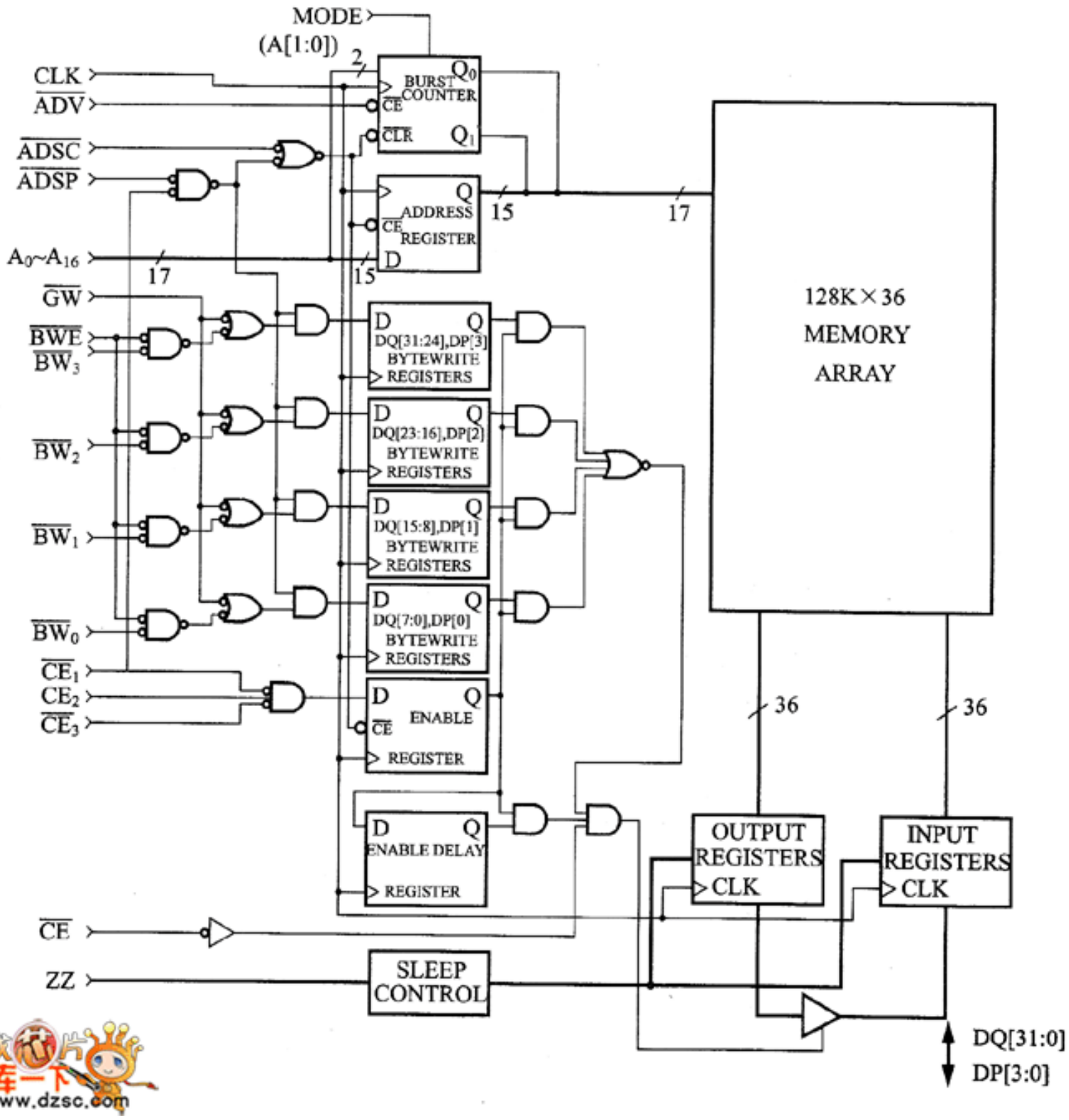


图 1　　CY7C1347B 的内部框图

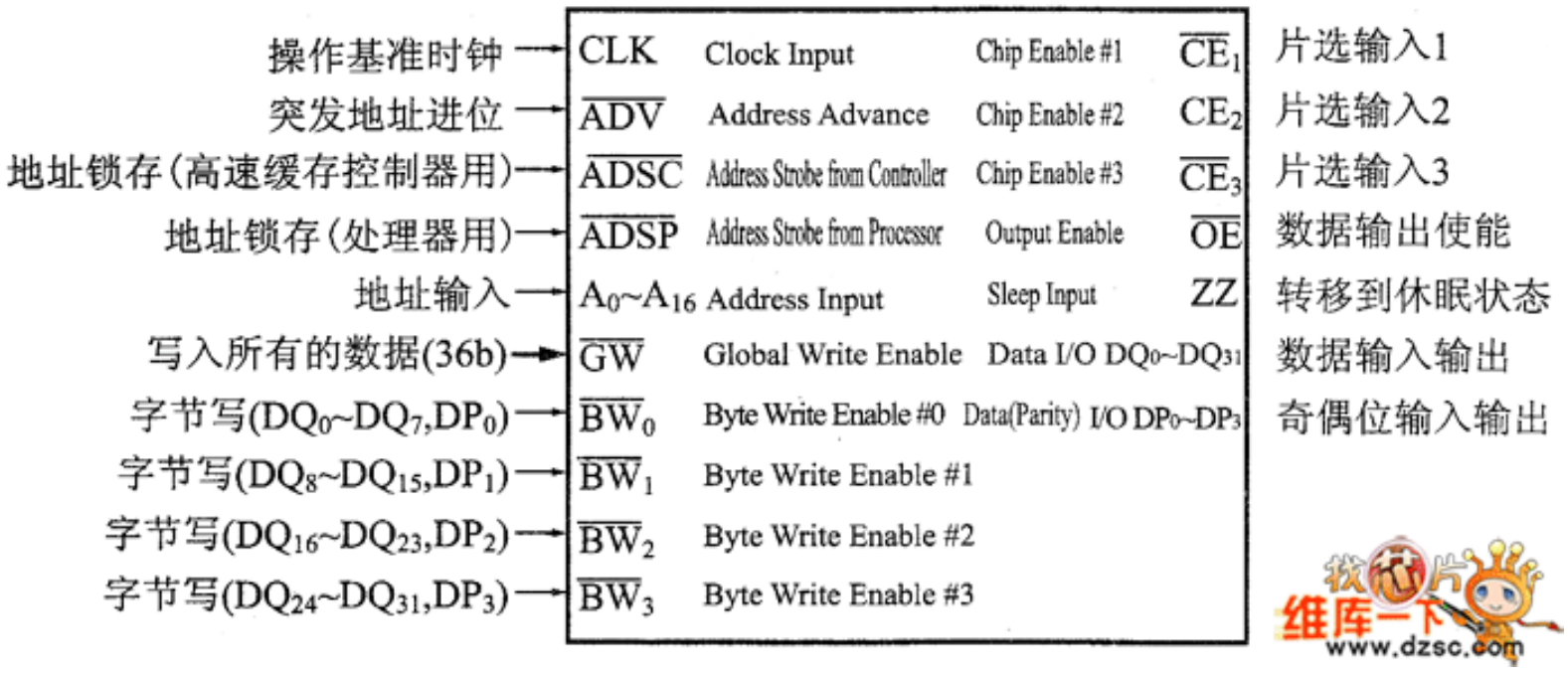


图 2　CY7C1347B 的信号

与主存储器相比，高速缓冲存储器被要求快速操作。因此为了尽可能不在外部对控制信号进行处理，所以准备了多个类似这样用于存取的信号。

例如，通常情况下，CPU 的 ADS（地址选通）信号与/ADSP 信号直联，而缓存控制器与/ADSC 信号直联。这样，当 CPU 访问外部总线时，其地址也被同步管道突发式 SRAM 提取。

进行读操作时，缓存控制器判断其范围内的数据是否被存放（是否命中？）于同步管道突发式 SRAM（缓存数据 RAM）中，如果数据存在则操作控制信号，以便读出同步管道突发 SRAM 的数据。如果是写操作，则使/GWE 信号有效，进行数据的更新。

当成为外部总线主控器的器件要读取存储器时，缓存控制器利用/ADSC 信号，为同步管道突发式 SRAM 提供外部总线主控器读出的地址。

●4.4.3　同步管道突发式 SRAM 的各种信号

CY7C1347B 所具有的各种信号及其意义如下所述，基本上各种信号都是在时钟（CLK）信号的上升沿被采样的。

▲　A0～A16（地址）

这是地址输入。CY7C1347B 的数据具有 36 位，这是每 8 位数据＋1 位验证的结构，共有 4 字节大小。由于普通的处理器是以 8 位为单位进行输人输出的，所以一般都是 A0 连接 CPU 的 A2、A1 连接 A3 这样的形式。

突发传输时，根据内部的突发式计数器，A0 与 A1 被自动更新。异步 SRAM 的情况下，由于只要能从写入的地址中读出数据即可，所以地址引脚即使颠倒连接也不会出现问题。但是在同步管道突发式 SRAM 的情况下，如果 A0 及 A1 引脚颠倒连接，则在突发传输中将出现异常，因此一定要明确地将 A0 作为 LSB 使用。

▲　/BW0～/BW3（Byte Write Select）

这是 1 字节（实际上为 9 位）的数据写控制信号。时钟上升时，当/BWE 信号有效（低电平）时，其中对应于有效（已成为低电平）信号字节部分的数据将成为要更新的对象。/BW0 对应于 LSB 一端（DQ0～DQ7 及 DP0），/BW3 对应于 MSB 一端（DQ24～DQ31 及 DP3）。

▲　/GW（Globle Write Enable）

/BWn 是以 1 字节为单位的写入控制，而/GW 是汇集 4 字节（正确地说为 36 位）进行写入的信号，是低电平激活信号。

/GW 有效时，/BWn 及/BWE 是无效的。

▲　/BWE（Byte Wirte Enable）

这是用于控制/BWn 使能与禁止的信号，如果在时钟沿上为低电平，则/BWn 为有效。

▲　CLK（Clock Inpat）

这是存储器的操作标准时钟，控制信号、地址等的提取以及数据的输人输出都是与时钟的上升沿同步进行的。

▲　/CE1（Chip Enable 1）

这是低电平激活的Chip Enable信号。如果CE2 及/CE3 全部有效，则器件被选择。

CE1 也作为ADSP的屏蔽信号使用，如果/CE1 无效，即使ADSP有效，内部也不锁存地址。

▲ CE2（Chip Enable 2）

这是高电平激活的Chip Enable信号，如果/CE1 及/CE3 全都有效，则器件被选择。

▲ /CE3（Chip Enable 3）

这是低电平激活的 Chip Enable 信号，如果/CE1 及 CE2 全都有效，则器件被选择。

▲ /OE（Output Enable）

这是与低电平激活的时钟信号异步的输人信号，当希望读取数据时，使/OE 有效。/OE 虽然是异步输人，但从其内部框图可以看出，通过时钟同步的 Chip Select（片选）以及/WEn 信号等，可对/OE 进行屏蔽。

从图中可以看出，由于写操作的方向具有优先权，所以/OE 即使保持有效，在进行写操作时也会自动地关闭输出缓冲器。

▲ /ADV（Advance）

这是对应于突发传输，指示“下一地址”的信号。如果在时钟的上升过程中/ADV 有效，则突发式计数器变为使能状态，自动生成下一地址。

突发传输时的地址递推方式（称为 burst order 或 burst sequence，突发顺序）从大方面分为交叉存取突发顺序和线性突发顺序两种。所谓的交叉存取突发顺序就是将最初地址的下一地址位 0（A0）反相，然后再将其下一地址位 1 和位 0 反相，最后位 0 反相。而线性突发顺序是以位 0 / 1 按照 00→01→10→11 的顺序推进的。各种突发顺序整理如表所示。

表　突发顺序

突发模式	A(1：0)			
	第 1 次	第 2 次	第 3 次	第 4 次
线性突发	00	01	10	11
	01	10	11	00
	10	11	00	01
	11	00	01	10
交叉存取突发	00	01	10	11
	01	00	11	10
	10	11	00	01
	11	10	01	00

当地址的低位 2 位为“00”时，虽然无论哪种方式都进行相同的操作，但顺序却是不同的。例如，当从“01”开始时，交叉存取突发顺序为 01→00→11→10，而线性突发顺序为 01→10→11→00。

80486 以及奔腾系列等 Intel 的处理器采用交叉存取突发顺序，而其他的 RISC 系列的微型计算机等采用线性突发顺序。

▲ /ADSP（来自处理器的地址选通）

如果/ADSP 在时钟沿有效，则 A0～A16 将被锁存于地址寄存器及突发式计数器中。由框图我们可以知道，/GW 及/BWn 等写信号在/ADSP 有效的时钟沿上是无效的，/WE 及写数据最快也要在/ADSP 的下一个时钟中赋予。例如，在进行写回高速缓存（WriteBack Cache）操作的情况下，CPU 在进行写操作的时候，会暂时将缓存的内容写出（从缓存进行读操作）主存储器，然后为了将 CPU 所读出的数据写人，在来自 CPU 的存取中一般都只暂时锁存地址。

/ADSC 对于地址锁存也具有相同的功能，但它并不屏蔽写人相关的控制信号。由于是控制器进行的操作，因此同时确定地址可以争取一个时钟的时间。

▲ /ADSC（来自控制器的地址选通）

与/ADSP 相同，如果在时钟沿/ADSC 有效，则 A0～A16 以及/GW 和/WE 信号将被锁存于地址寄存器和突发式计数器中。

▲ ZZ（Sleep）

这是异步的高电平激活的输入。如果该引脚成为高电平，则处于断电状态，功耗变小。一般都设置为低电平使用。在台式计算机中同步管道突发式 SRAM 的功耗对整体具有很大的影响，所以大多数的情况是一直设置为低电平进行使用。

▲ DQ0～DQ31、DP0～DP3（双向数据输入输出线）

这是数据总线，其中 DQ0～DQ7 与 DP0、DQ8～DQ15 与 DP1、DQ16～DQ23 与 DP2 以及 DQ24～DQ31 与 DP3 分别成对。

当在时钟沿上芯片使能（/CE0、CE1、/CE2 都有效）以及与写操作相关的信号（/GW 及/BWn、/BWE）全都无效时，如果/OE 有效，则操作是对存储器单元的访问，2 个时钟后将输出数据。

如果在时钟沿上芯片使能，写信号有效，则 DQn、DPn 为输人，数据将与下一个时钟沿同步被提取到内部的锁存器中，进而在下一个时钟写入存储器单元。

▲　MODE（突发顺序选择）

这是为了进行突发顺序的选择。如果与 GND 连接则选择为线性突发顺序；如果是 VDDQ 引脚及开放状态则是交叉存取突发顺序被选择。为了根据处理器的种类决定通过哪种模式进行操作，一般都将该引脚的状态提前固定，禁止在器件操作过程中更改该引脚的状态。

●4.4.4　同步管道突发式 SRAM 的基本操作

同步管道突发式 SRAM 的操作基本上都是与时钟的上升沿同步进行的，因此最好能够看到时钟沿的状态。在功能方面看上去非常复杂，但时序的读取比异步 SRAM 还是简单的。

▲　同步管道突发式 SRAM 的周期定义

由于都是与时钟同步的，所以可以通过时钟沿中各个控制线的状态确定下一个状态。

表　表示 CY7C1347B 的周期定义。

表　同步管道突发式 SRAM 的周期定义

下一周期	使用地址	ZZ	$\overline{CE_3}$	CE_1	$\overline{CE_1}$	$\overline{ADSP}$	$\overline{ADSC}$	$\overline{ADV}$	$\overline{OE}$	DQ	Read/Write
非选择状态	不使用	L	X	X	H	X	L	X	X	Hi-Z	X
非选择状态	不使用	L	H	X	L	L	X	X	X	Hi-Z	X
非选择状态	不使用	L	X	L	L	L	X	X	X	Hi-Z	X
非选择状态	不使用	L	H	X	L	H	L	X	X	Hi-Z	X
非选择状态	不使用	L	X	L	L	H	L	X	X	Hi-Z	X
开始读	从外部锁存	L	L	H	L	L	X	X	X	Hi-Z	X
开始读	从外部锁存	L	L	H	L	H	L	X	X	Hi-Z	Read
连续读	下一地址	L	X	X	X	H	H	L	H	Hi-Z	Read
连续读	下一地址	L	X	X	X	H	H	L	L	DQ	Read
连续读	下一地址	L	X	X	H	X	H	L	H	Hi-Z	Read
连续读	下一地址	L	X	X	H	X	H	L	L	DQ	Read
固定读	现在的地址	L	X	X	X	H	H	H	H	Hi-Z	Read
固定读	现在的地址	L	X	X	X	H	H	H	L	DQ	Read
固定读	现在的地址	L	X	X	H	X	H	H	H	Hi-Z	Read
固定读	现在的地址	L	X	X	H	X	H	H	L	DQ	Rrad
开始写	现在的地址	L	X	X	X	H	H	H	X	Hi-Z	Write
开始写	现在的地址	L	X	X	H	X	H	H	X	Hi-Z	Write
开始写	从外部锁存	L	L	H	L	H	L	X	X	Hi-Z	Write
连续写	下一地址	L	X	X	X	H	H	L	X	Hi-Z	Write
连续写	下一地址	L	X	X	H	X	H	L	X	Hi-Z	Write
固定写	现在的地址	L	X	X	X	H	H	H	X	Hi-Z	Write
固定写	现在的地址	L	X	X	H	X	H	H	X	Hi-Z	Write
Sleep	未使用	H	X	X	X	X	X	X	X	Hi-Z	X

从器件的内部框图以及/CE3、CE2、/CE1 一栏可以看出，这些使能引脚是在操作开始时刻被利用的，而一旦开始进行读或者写操作，则这些引脚就不再被利用。

▲　读操作 1：单一读

所谓的单一读操作，就是读出所希望读取的地址数据，与异步 SRAM 的处理相同。其操作波形如图 1 所示，如果在最初的时钟沿上让芯片发挥使能，并且赋予地址，那么将在 2 个时钟后读出数据，只要在外部电路锁存该数据即可。

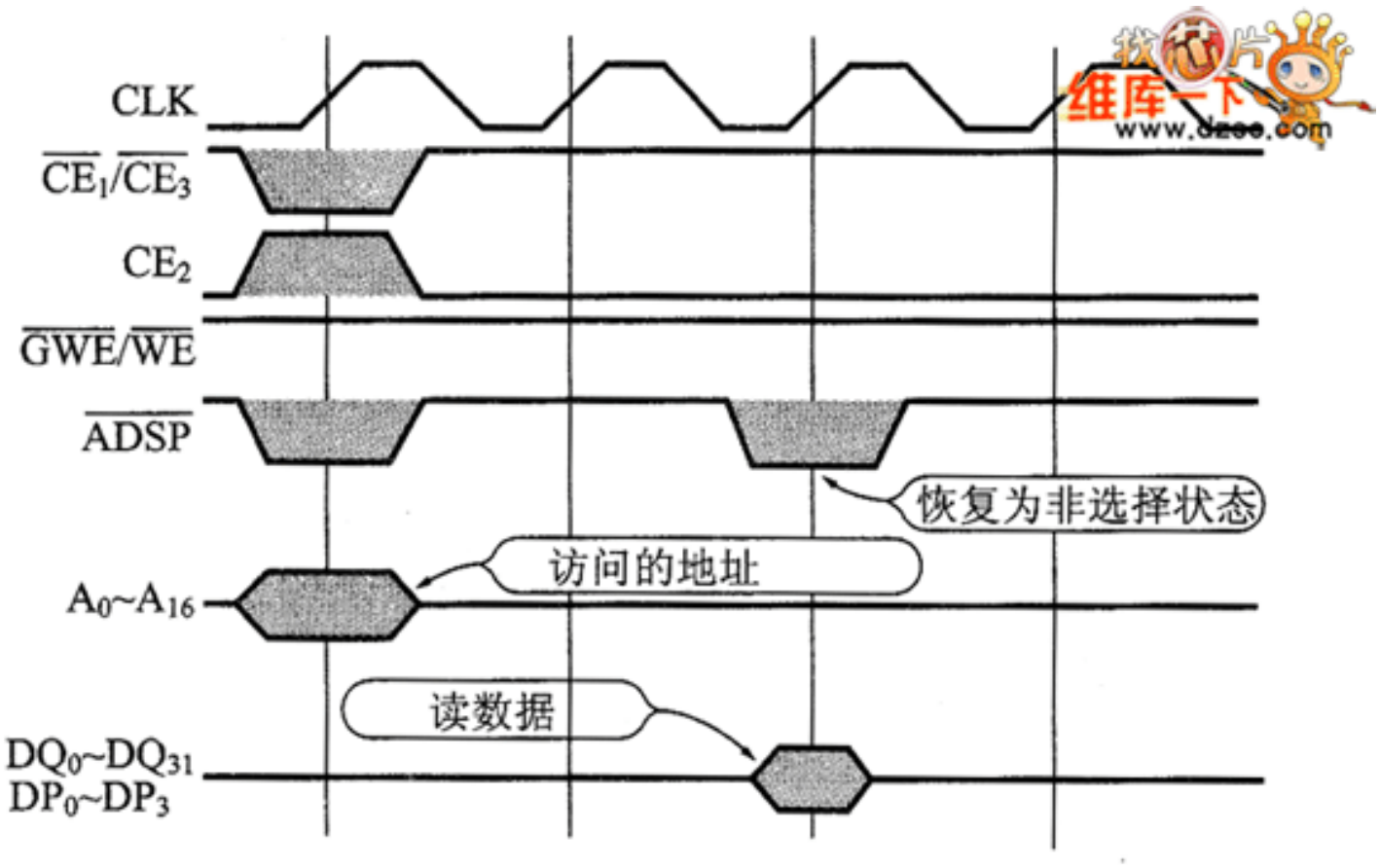


图 1　单一读操作

起始地址在最初的/ADSP 时钟时被锁存，开始对存储器单元进行存取操作，然后在下一个时钟从存储器中输出数据，进而在下一个时钟将数据提取至输出缓冲器的锁存器中。对于单一读只要具有这个印象即可。

▲　读操作 2：突发读

突发读时的操作如图 2 所示，在开始时刻与单一读的操作相同，但第 2 个时钟以后，在/ADV 有效方面具有独到之处。

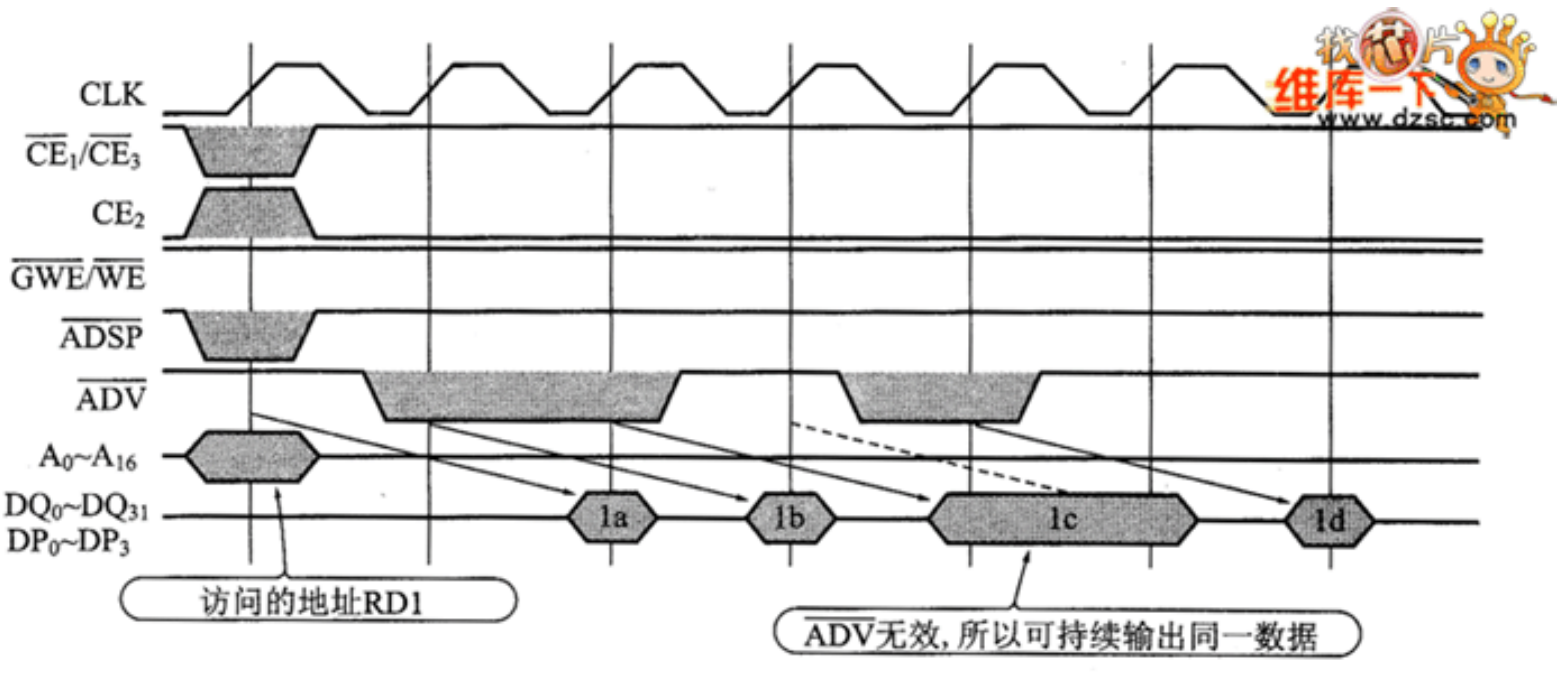


图 2　突发读操作

在锁存了地址的过程中，因为对存储器单元的存取操作已经开始，所以只能在下一个时钟更改地址。在此，如果/ADV 有效，则通过内部的突发式计数器，将更新低位的 2 位地址，2 个时钟后再读出新地址的数据。

图中也表示了在中途/ADV 无效时的操作，由于即使/ADV 无效，读操作本身也将继续，所以数据将被连续输出。如果/ADV 再次有效，则突发式计数器向前进位，在 2 个时钟后读出下一个数据。

▲　写操作 1：单一写

单一写的操作如图 3 所示。由于在此例中利用了/ADSP 信号，所以写控制信号以及数据将在第 2 时钟中赋予。利用/ADSC 信号时可以同时赋予数据及写控制信号。

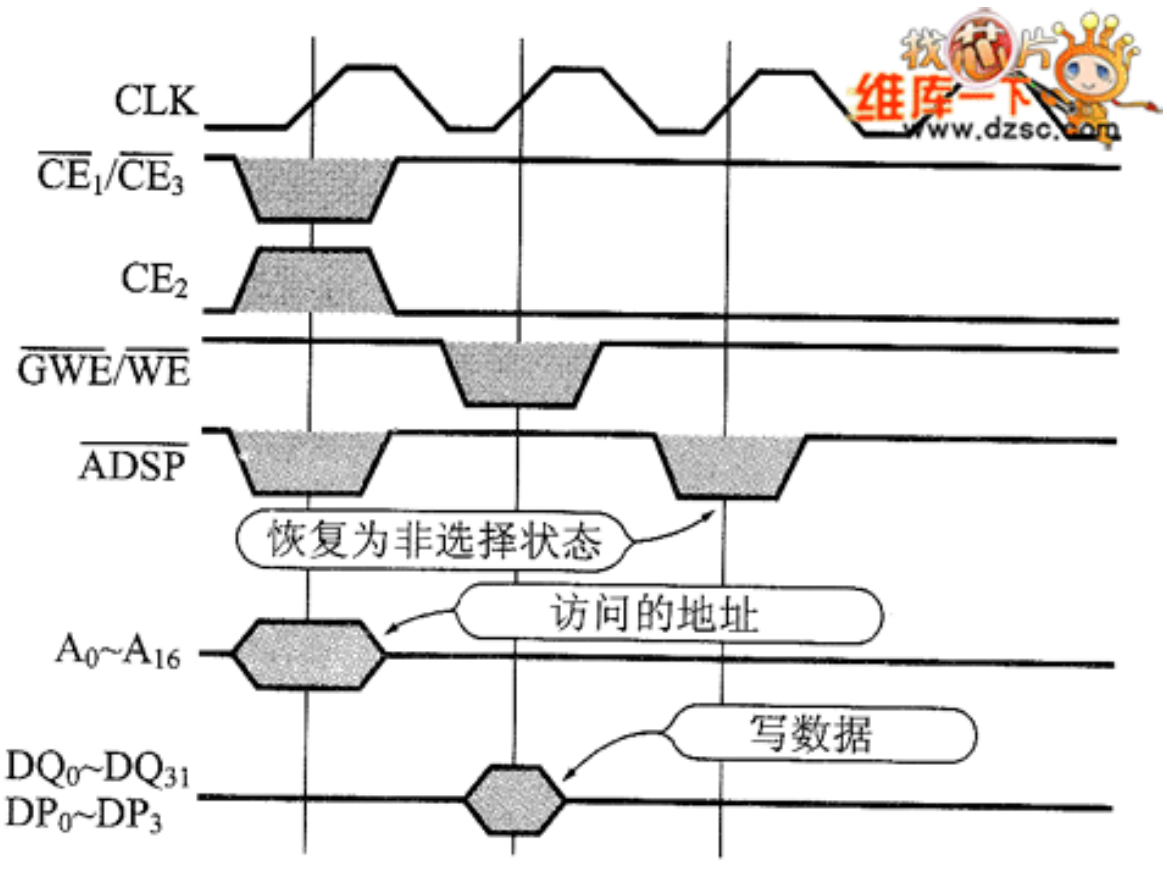


图 3 单一写操作

▲　写操作 2：突发写

表示突发写操作的如图 4，最初的写操作与单一写相同，第 2 次写操作以后由于与/ADV 信号同时赋予数据，因而成为推进数据传递以及地址的操作。与进行读操作时不同的是可以同时赋予信号/ADV 与数据。

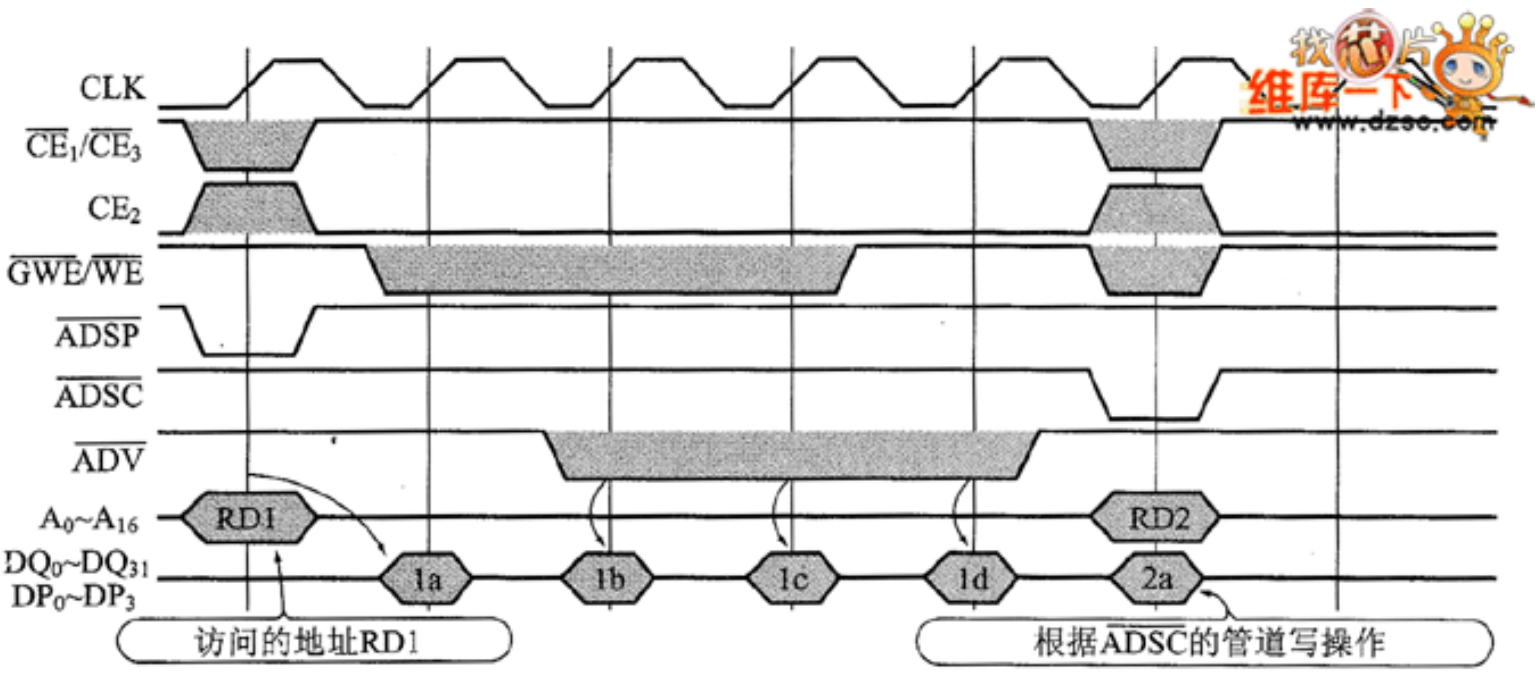


图 4　突发写操作

观看内部框图就可明白，此时，数据端被提取到输人锁存器中，地址进位，等待在下一个周期完成写人操作。

不可以在/ADSP 的下一个周期、即起始数据写人的时刻使/ADV 有效，如果/ADV 有效，而起始数据的写人操作还没有进行，那么将造成地址向前进位的后果。

在图中，对利用/ADSP 进行写操作之后利用/ADSC 的写入周期也进行了描述。虽然在这个时钟时刻，在同步管道突发式 SRAM 内部也在进行写入操作，但由于外部锁存器已经处于接受下一指令的状态，所以可以对利用/ADSC 信号的地址及数据进行锁存。

●4.4.5　同步突发式 SRAM

同步突发式 SRAM 的内部框图如图所示，它与同步管道突发式 SRAM 基本相同，不同之处只是在输出缓冲器中没有配置锁存器。

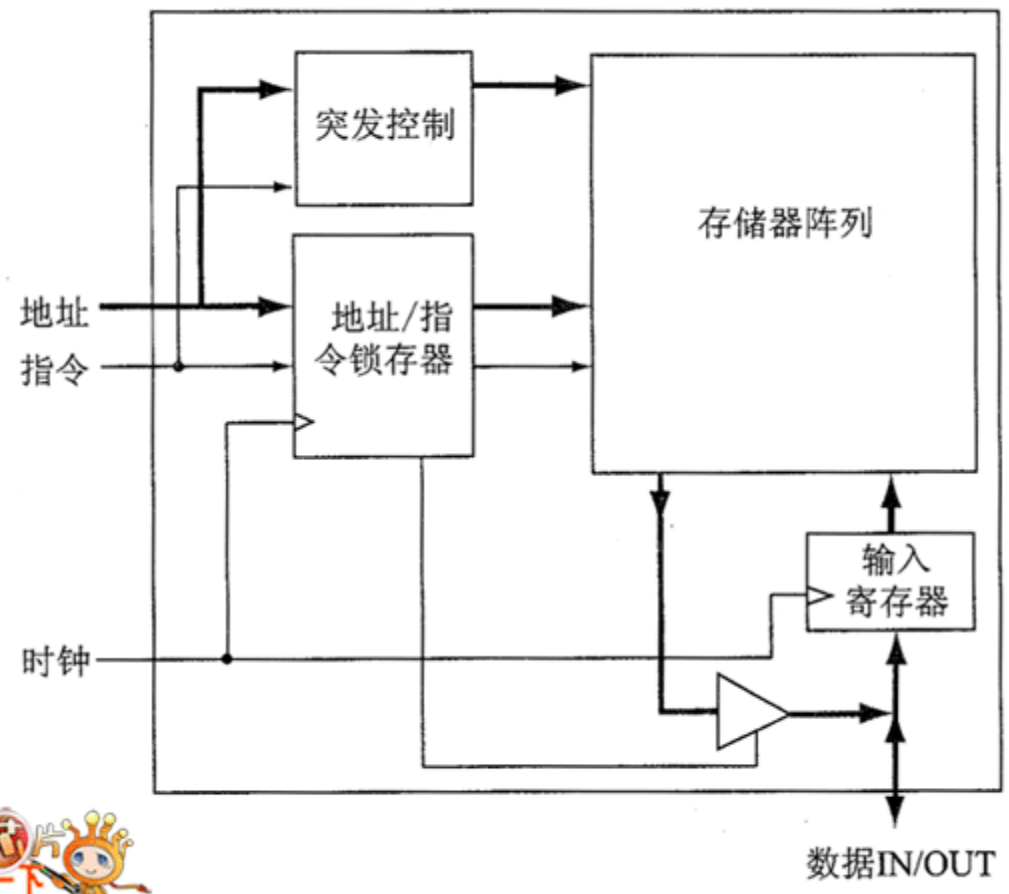


图　同步突发式 SRAM 的内部框图

因为没有锁存器，所以数据比管道突发类型的 SRAM 提前一个时钟输出。但另一方面，它很难提高时钟频率。前面曾例举的 Cypress 公司的同步管道突发式 SRAM 的时钟频率最高为 166MHz，但相同系列的同步突发式 SRAM 的频率最高却只有 117MHz。

在个人计算机的外部缓冲器中一般都使用了同步管道突发式 SRAM，而同步突发 SRAM 几乎没有被利用。

●4.4.6　实际的同步突发式 SRAM

我们将 Cypress 公司的 CY7C1345B 作为同步突发式 SRAM 的实例，它具有与 CY7C1347B 相同的 128K×36 位的结构，其内部框图如图所示。由框图可知，与 CY7C1347B 相比，除了没有输出寄存器以外，其他完全相同，只是将控制信号等/BWn 改名为/BWSn，其他方面完全相同，请参照同步管道突发式 SRAM 的说明。

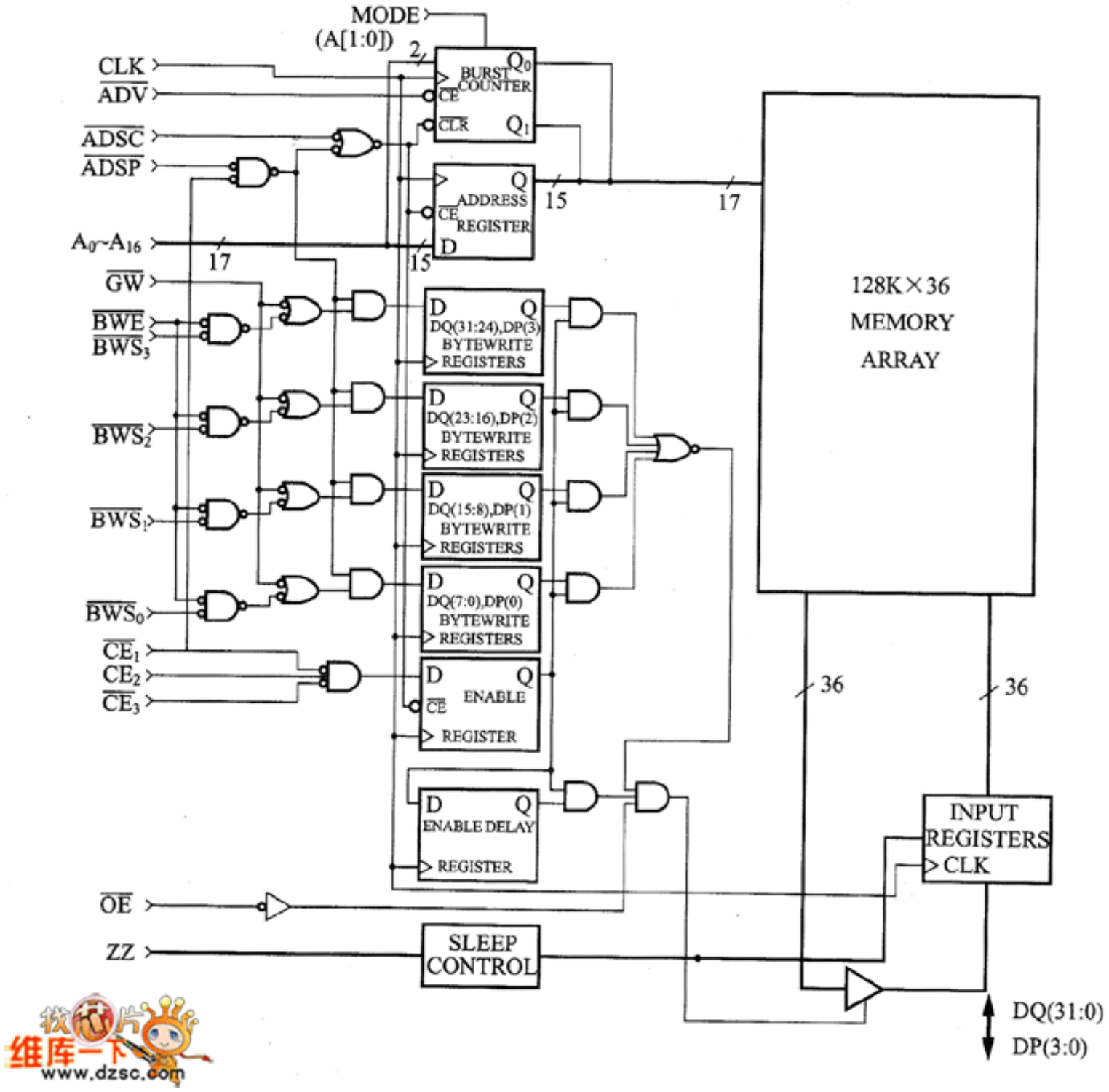


图　CY7C1347B 的内部框图

另外，从框图我们可知，同步突发式 SRAM 的写人周期与同步管道突发式 SRAM 的相同，所以在此我们只说明读操作。

●4.4.7　同步突发式 SRAM 的单一读操作

同步突发 SRAM 的单一读操作如图所示，与同步管道突发式 SRAM 不同的是，数据是在/ADSP 有效的下一个时钟中输出的。

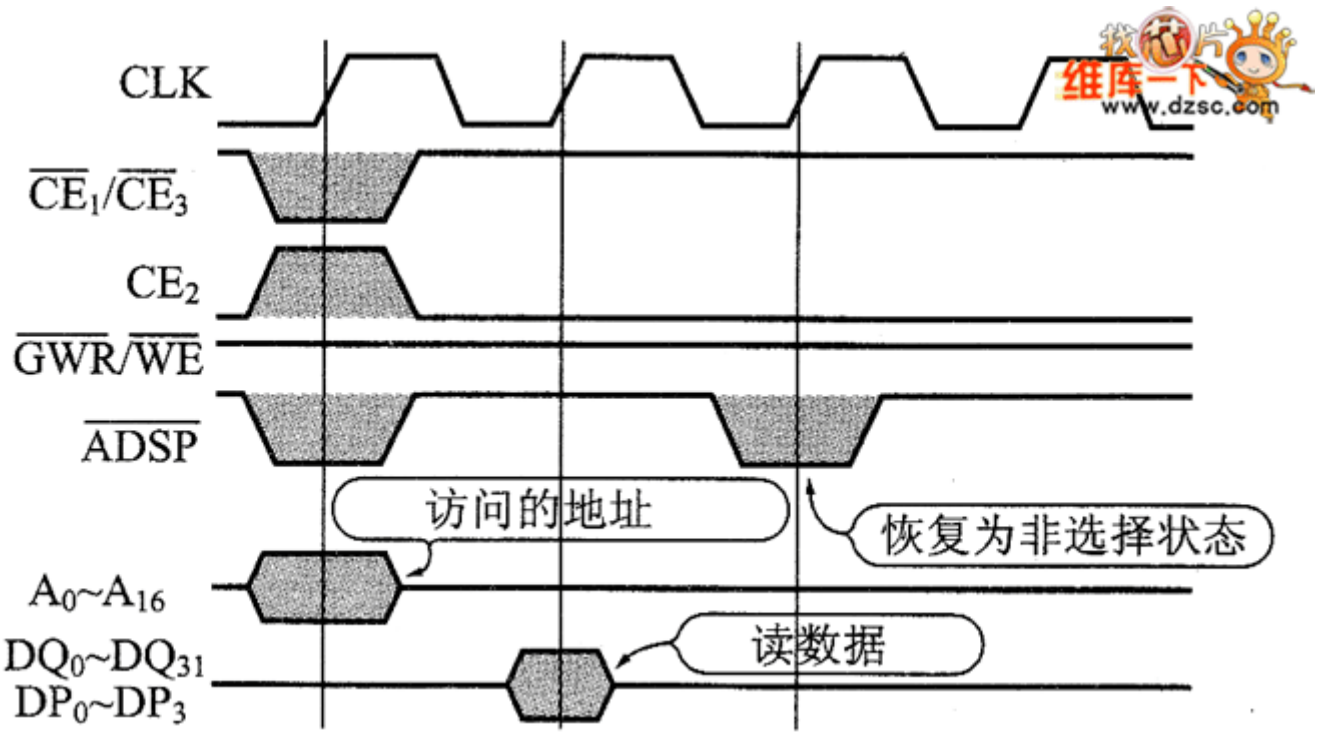


图　　同步突发 SRAM 的单一读操作

●4.4.8　同步突发式 SRAM 的突发读操作

同步突发式 SRAM 的突发读操作如图所示，与单一读操作相同，因为突发读操作也是在赋予地址的下一个时钟且/ADV 有效时，在一个时钟之后输出下一地址的数据。所以，与管道突发类型相比，整体上形成缩短了一个时钟的操作波形。

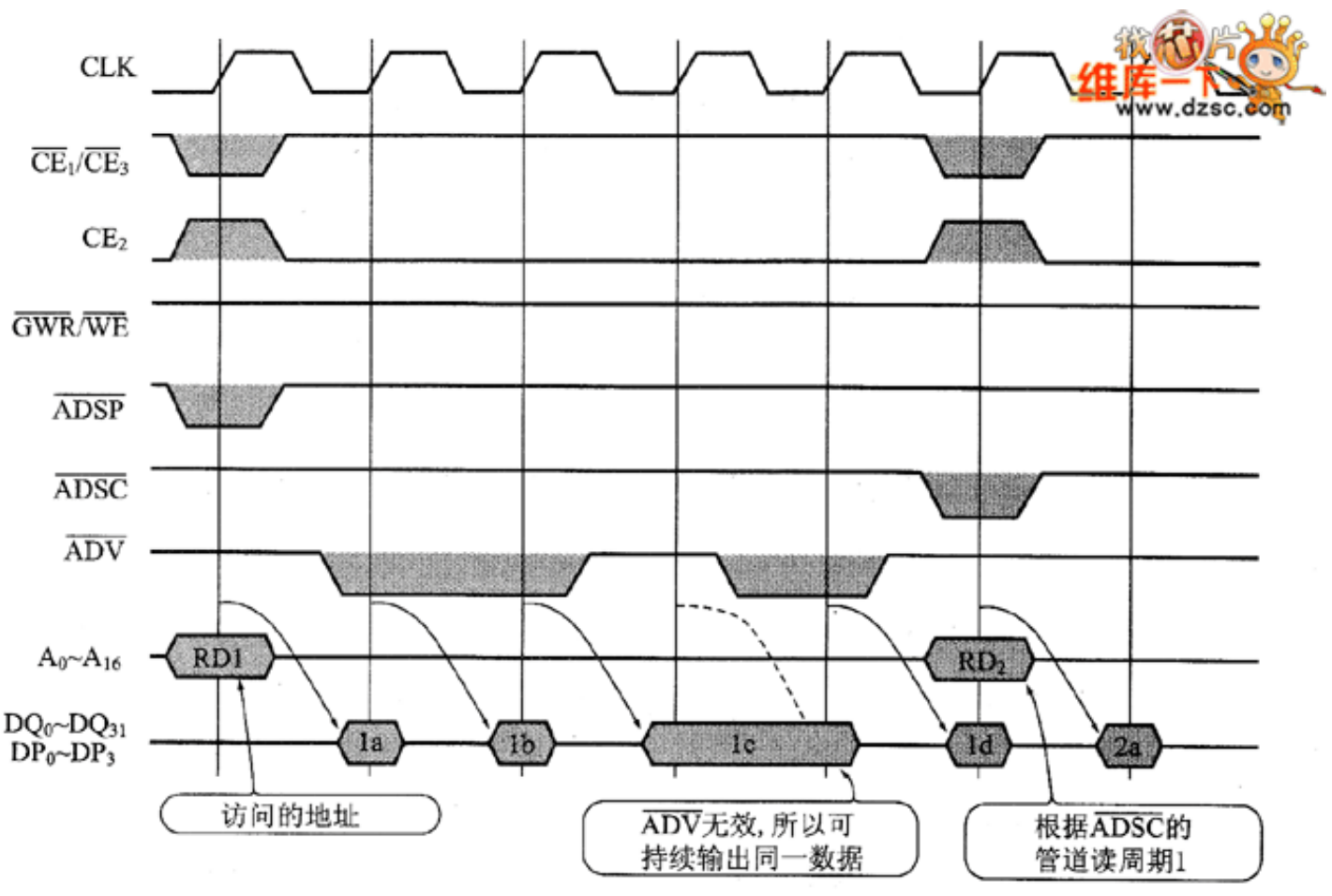


图　同步突发式 SRAM 的突发读操作

图中，在利用/ADSP的突发读操作之后，追随利用/ADSC的单一读周期。在不需要（已经指示到最终地址）/ADV有效的过程中，由于下一地址被赋予，因此第 4 次的数据的读操作与赋予下一个存取地址的/ADSC将同时进行。

4.5　SRAM 主板的制作

为了实际使用一次 SRAM，我们尝试制作附加了与 ISA 总线（PC104）相连接的电池备份的异步 SRAM 主板。

本次制作的电路如图 1 所示，电路中 PLD（MEMDEC）的内部电路如图 2 所示。存储器打算专用 8 位幅宽、ISA 总线 D0000h～DFFFFh 的 64K 字节的区域，但由于目前的个人计算机都安装了各种各样的适配卡，不知道该范围内是否空闲。为此，先启动 Windws，选择“我的电脑→属性→设备管理器→计算机→属性→内存”，看看所显示的内容，确认是否存在空余空间。

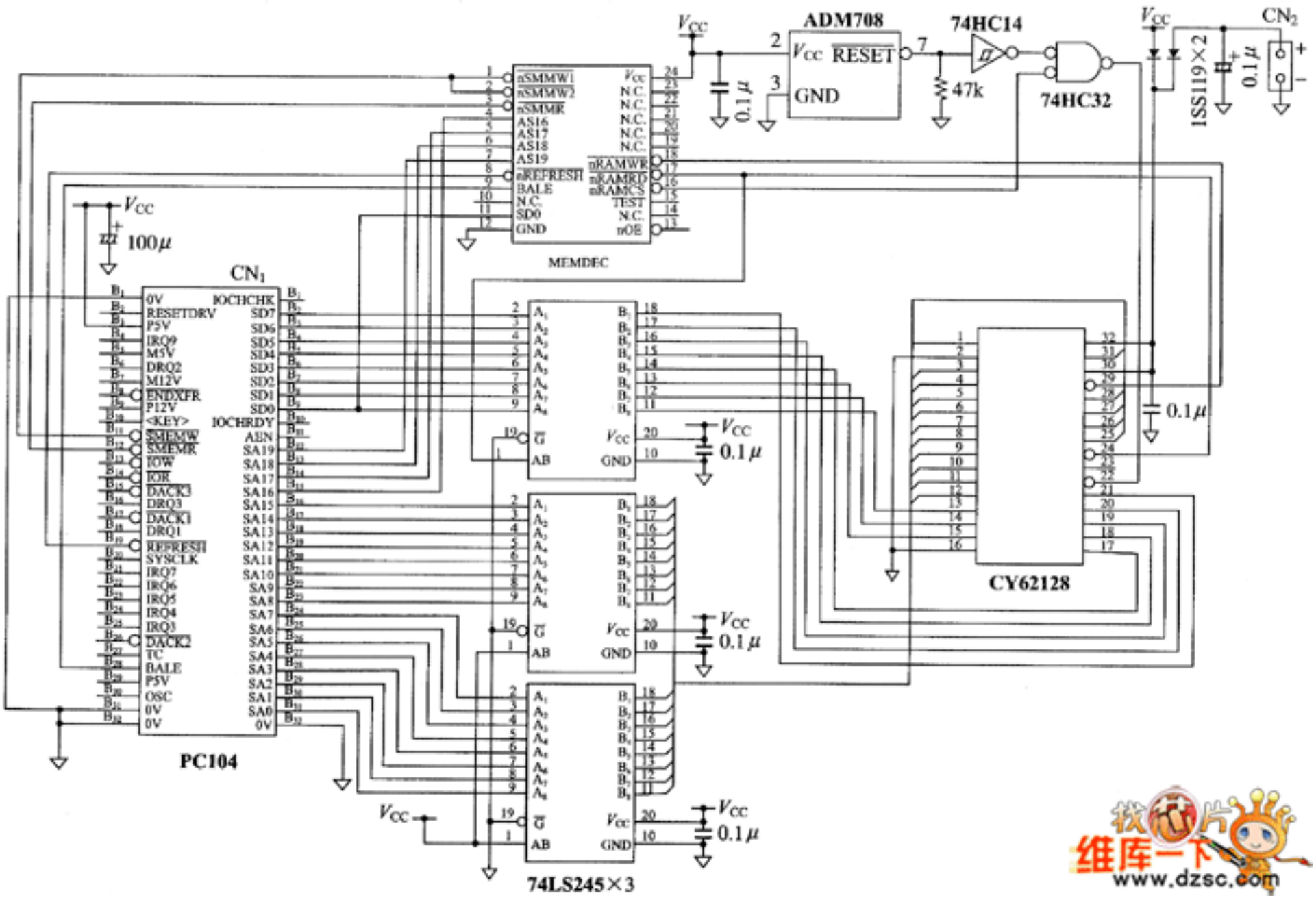


图 1　用于 ISA 总线的 SRAM 主板的电路图

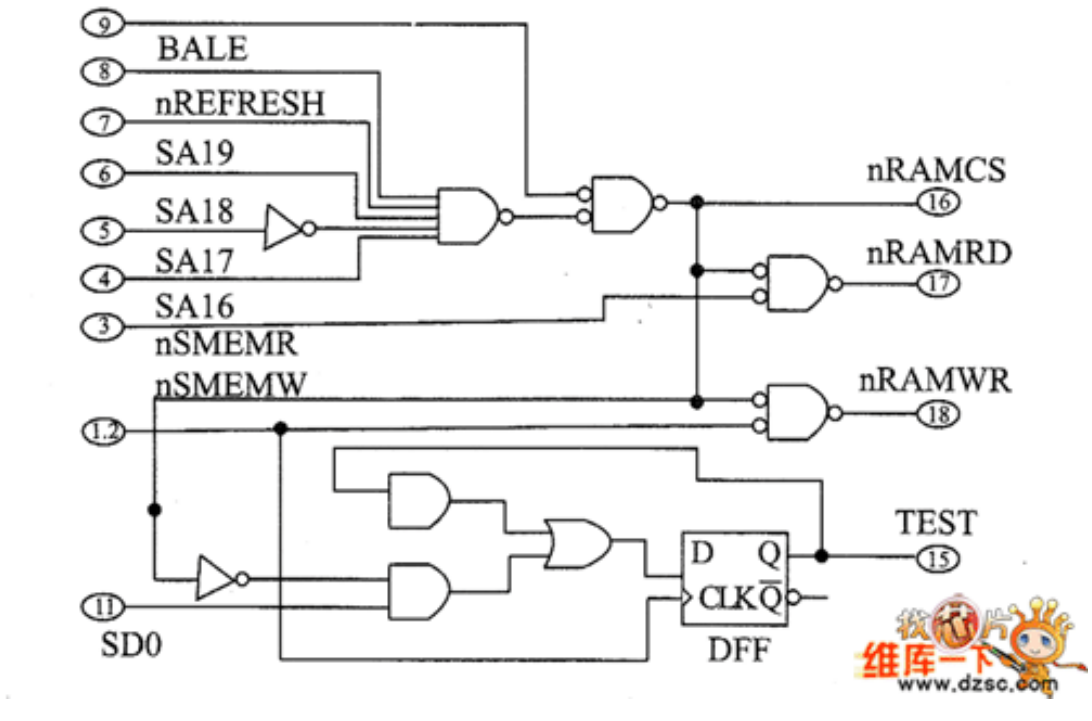


图 2　　PLD 的电路

●4.5.1　ISA 总线存储器周期的注意事项

ISA 总线基于 IBM 鼻祖计算机所具有的 1MB 空间的 8 位总线，扩展成为具有 16MB 空间的 16 位总线。在兼容性保持方面给予了足够的重视，具体地说就是将 ISA 总线分成两个插口（CardEdge），与 16 位扩展相关的信号都被分配到小的插口（添加的）上。除此之外，我们还可看到在信号关系方面，为了保持兼容性也做了相当细致的工作。

下面，我们将在利用扩展部分及存储器空间的基础上针对必须注意的信号进行解说。在说明中，我们假设将 ISA 总线的插口中靠近面板一侧（较宽的一侧）的称为 8 位总线部分，将另一个插口称为 16 位扩展部分。

▲　地址

地址总线以不同的信号名称交叠存在,8 位总线部分为 SA0～SA19,16 位扩展部分为 LA17～LA23。PC / AT 的思路是将主存储器也扩展到 ISA 总线上，因此，只要认为可以在 1M 字节（100000h 地址）以上的范围内简单配置以 128K 字节为单位的扩展存储器卡、拥有到 LA17 为止的地址即可。

▲　存储器读 / 写信号

存储器的读 / 写信号在 8 位总线部分具有/SMEMR及/SMEMW信号，而在 16 位扩展部分具有/MEMR

及/MEMW信号。

两者虽然具有完全相同的意思，但有效的范围不同。/MEMR和/MEMW在进行ISA总线的存储器存取操作中必须有效，而/SMEMR及/SMEMW只在存取 1M字节以内的范围（000000h～0FFFFFh）时有效。

这是为了保持低位的兼容性。由于 8 位总线的存储器空间为 1M字节，所以地址总线只有 20 根（SA0～SA19）。因此，单从 8 位总线的地址看，不能区分CPU连接了 0 地址、100000h地址还是 200000h地址等。如果将原来的 8 位总线卡插人ISA总线，则当访问 1M字节以上的空间时，若/SMEMR及/SMEMW也有效，就是非常糟糕的事情了。因此，设计时就需要将其设计成只能在 1M字节以内的空间访问时有效。

由于本次将SRAM主板放置在 000000h～0DFFFFh地址，所以利用了/SMEMR及/SMEMW信号。

▲ 刷新

由于曾有过在 ISA 总线上利用 DRAM 对主存储器进行扩展的想法,所以刷新周期大约为 15.6μs。该刷新操作与/REFRESH 信号一起有效，在地址低位的 8 位（SA0～SA7）上附上刷新地址，就可以形成存储器读取周期。

这类似于哑元的存储器读取周期，本次为了以防万一，将其设为不应答。

▲ 等待关系

本次不利用与等待相关的信号，仅做一些说明。

ISA 总线还包括为了延长 CPU 总线周期的等待信号（IOCHRDY）以及缩短总线周期、提升速度的/SRDY（有时也表示为/ZWS 及/OWS）信号。

在目地端针对主机的要求不能立即应答的情况下，利用 IOCHRDY 等待总线周期的结束，以低电平表示 Not Ready，也就是等待的意思。由于借助 ISA 总线的上拉电阻通常将其设置为高电平，所以只要不做任何更改，它将不会处于等待状态，只是执行普通的总线周期。

/SRDY 信号则相反，是能够缩短总线周期的信号。ISA 总线的情况下，16 位存储器存取操作（/MEMCS16 有效）虽然能够在 3 个周期内完成，但为了能让通常需要 6 个周期完成的 8 位存储器存取操作缩短到 3 个周期，就可以利用/SRDY 信号。

▲ 8 位存储器周期

ISA 总线的 8 位存储器存取周期如图所示，并列了标准周期、利用了 IOCHRDY 信号的插人一个等待的示例以及利用了/SRDY 信号的不等待存取的操作示例。

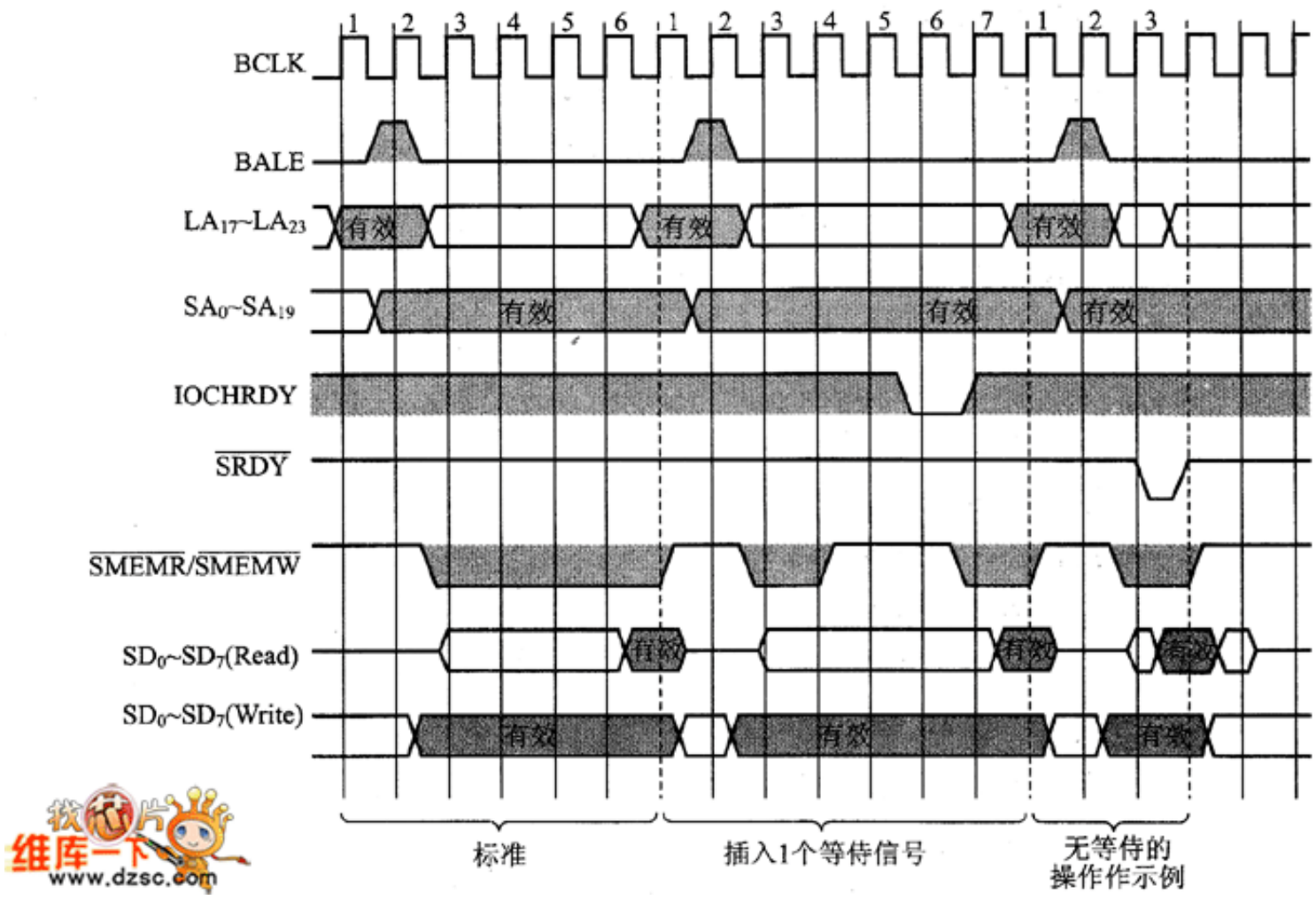


图　ISA 总线的 8 位存储器存取周期

由于 SYSCLK 通常为 8MHz，所以一个周期为 125ns。现如今 SRAM 的存取时间在 100ns 以内的非常普通，所以可以说这是非常缓慢的总线周期。8 位存储器周期如果不采用等待信号，则可以在 6 个周期内完成存取操作。

BALE 在高电平期间地址发生变化，地址（SA0～SA19）确定后，BALE 变为低电平，指令（/SMEMR、/SMEMW）有效后开始存取操作。

因为作为高位地址的 LA17～LA23 在 BALE 变为低电平后，在规格上是不定的，所以需要提前在 BALE 上锁存译码结果，锁存 LA。事实上，由于在主板上特意改变 LA 没有任何意义，所以，LA 并不是不定的，一般是与 SA 同样保持输出状态。尽管如此，在这方面也需要加以注意。由于本次配置的地址在 1M 字节以内的范围（0D0000h）内，不会利用到 LA，因而不必注意此处。

在写操作时，指令（/SMEMW）有效之前尽早确定数据，存储器由于是在/SMEMW 的上升沿提取数据的，所以对于建立时间，可以说有足够的富余时间。指令有效之后，主机与时钟的上升同步监视 IOCHRDY 信号，如果 IOCHRDY 变为低电平，就插入等待信号。

本次不是特别需要等待信号，因此只按照原来的默认时序进行。在结束第 6 个周期时指令无效。在进行读操作时，以这样的时序提取数据。

●4.5.2　SRAM存储器主板的基本设计

▲　地址缓冲器

在提供给存储器的SA0～SA15地址中加人缓冲器。缓冲器利用74LS244也可以，但因为741LS245布线简单，所以通过74LS245可单向使用。

▲　数据缓冲器

因为数据需要双向进行，所以要利用74LS245进行接收。将栅极一直打开，通过对存储器的读信号来进行方向控制。本次我们采用将PLD上存储器的读信号设置为只在/CS1有效时才输出的方法。

▲　PLD（MEMDEC）

PLD应用于生成对存储器的片选、/OE以及/WE信号中。片选信号是在刷新周期以外、当地址高位（SA16～SA19）为Dh（将D0000h～DFFFFh设置在SRAM主板空间）、且BALE为低电平时被选择的。

将存储器的读/写信号设置为当片选和/SMEMR、/SMEMW有效时输出。

▲　备份电源的切换

电池备份的重点在于电源切换和片选信号的控制。本次为了简单起见，只单纯获取Vcc和电池（为CN2提供3.6V的电池）的二极管OR，但需要注意二极管正向电压降。如果电源电压比所提供的电压低很多，则可能发生超出操作电压或者输入引脚的电压高于电源电压的情况。

▲　片选控制

为了电池备份，必须使存储器的片选信号无效。本次我们虽然只利用/CE1进行控制，但为了保持较低的损耗电流，必须使/CE1保持与电源电压相近的值（CY62128为VCC-0.2V以上）。为了进行片选控制，将利用作为电源监视IC的ADM708（模拟器件）和74HC系列的CMOS门组成电路。

ADM708本来是CPU用于生成复位信号的器件，这种用于电源监视的IC具有几个种类，还包括用于SRAM的电池各份的电源切换电路及内置片选控制功能的IC。利用这种IC的电路虽然非常简单，但器件的价格有些高，这是其缺点所在。

我们本次利用的ADM708引脚配置以及内部框图如图1所示。电源的切换关键在于电源电压下降到何种程度才能使之成为忽略主机信号的备份状态，由个别零部件进行这样的控制是相当麻烦的。

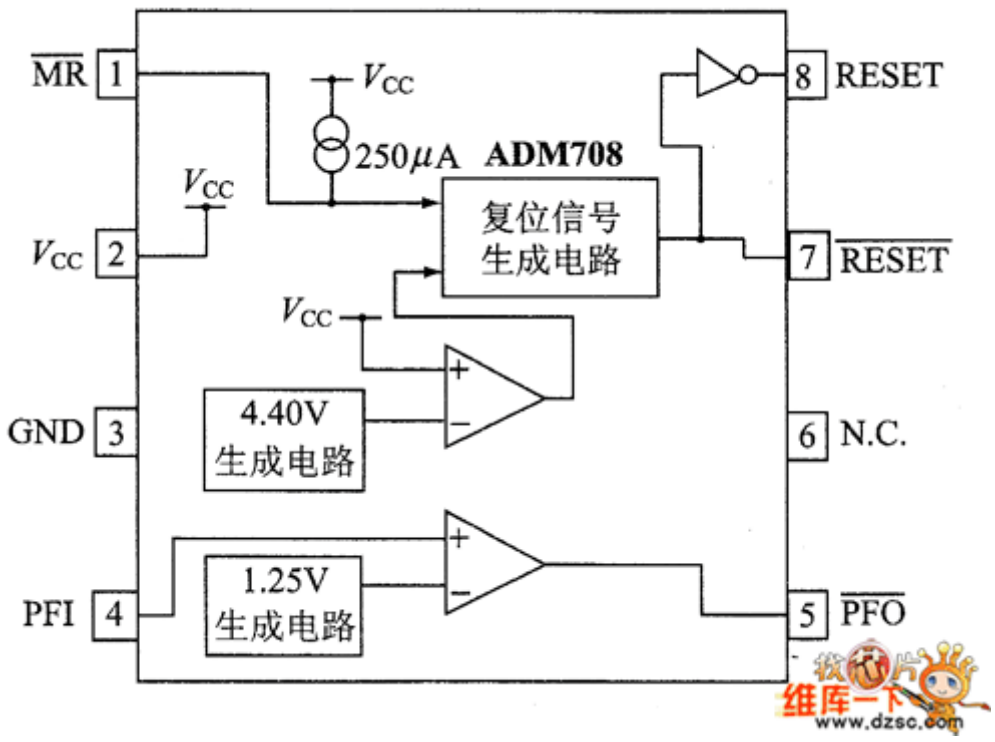


图 1 ADM708 的引脚配置与框图

从框图上可以明白，ADM708 内部具各 4.40V 和 1.25V 的生成电路，4.40V 的生成电路与 Vcc 相比较，增加了一个复位生成电路。当电源电压低于 4.40V 时，RESET、/RESET 信号有效（RESET 为高电平，/RESET 为低电平）。

电路的操作如图 2 所示。因为 Vcc 自身将逐渐降低，而 RESET 方面的输出电压也将随之一块降低，为此我们这次将利用/RESET 的输出。当电源电压超出 ADM708 的操作范围时，为了确保低电平而增加下拉电阻，由 74HC14 的施密特触发器的栅极接受。74HC14 以及下一阶段的 74HC32 的电源引脚与 SRAM 的电源引脚公用。

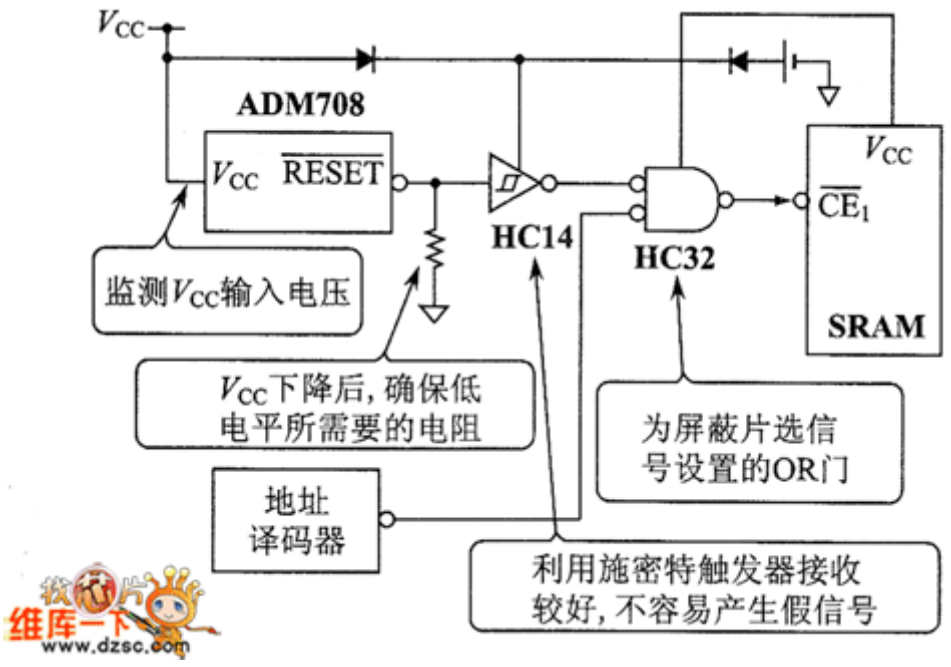


图 2 备份电路的操作

因此，如果 ADM708 的/RESET 为低电平，则 74HC32 的输出引脚被强制为高电平，又因为 SRAM 的/CE1 无效，因而变为待机状态。

●4.5.3　SRAM存储器主板的操作确认

让我们对已完成的SRAM存储器主板进行操作。在MSDOS模式下启动，利用DEBUG指令，从D0000h开始试着进行数据的读/写操作。如果确认了主板能够正常运行，则为备份电源连接器（CN2）提供电源，去掉个人计算机的电源，损耗电流在40μA左右。重新启动MS－DOS模式，读取刚才写人的地址，因为能够读出所写人的数据，因而可知备份电源是起到了相应的作用的。

由于从D0000h开始的领域为PC/AT的扩展BIOS领域，所以，如果SRAM上事先写入了附加头信息等的数据，则在操作系统启动前将被调用。在SRAM上安装各种经过仔细研究的程序进行试验，你就会有非常有趣的发现。

SRAM与闪速存储器等不同，它的替换操作是非常简单的，可以以1字节为单位进行替换，并且不需要替换时间。一旦拔掉电池数据将丢失，因而在实施ROM化之前的阶段，可以进行各种各样的实验，这是其方便之处。

第五章

特殊的 SRAM 的结构与使用方法

5.1　双端口 SRAM

双端口 SRAM 包括与时钟异步的类型和与时钟同步的类型两种。同步类型的产品并不是简单的在异步产品的外部增加了锁存器，而是具有了类似同步突发式 SRAM 的、自动地址进位的功能。

在异步类型的产品中，在从左右两个端口向同一地址访问发生冲突时，利用/BUSY 信号让稍后到达的访问等待，在同步类型的产品中没有这样的等待控制，因而两个端口可以异步进行访问。

●5.1.1　异步类型的双端口 SRAM

作为异步类型的双端口 SRAM，我们以 Cypress 公司的 CY7C019 为例，CY7C019 的内部框图如图所示。

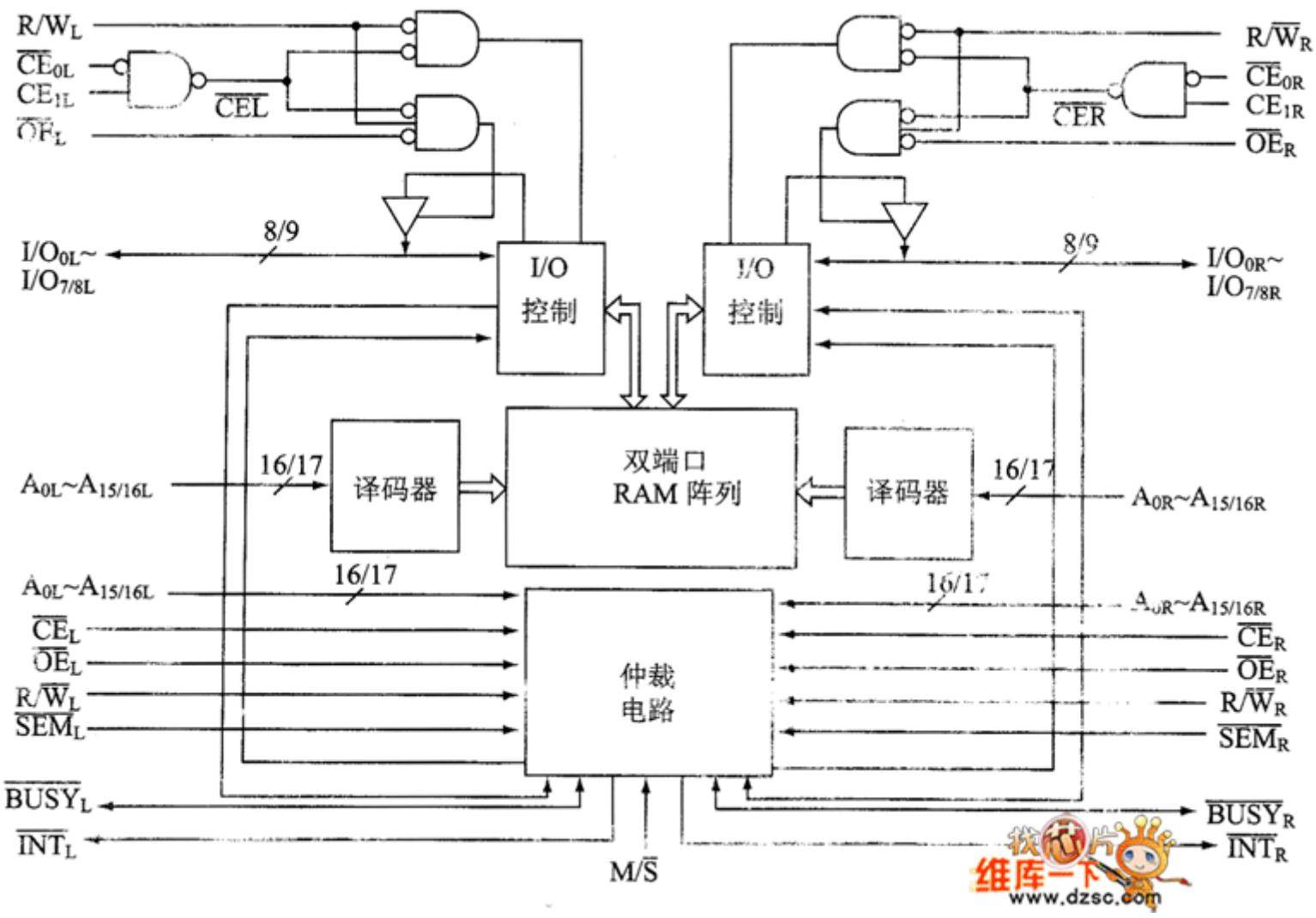


图 CY7C019 的内部框图

中央部分为双端口存储器阵列，并列着能同时设置两个地址的存储元器件。下面的框图是实现了控制信号的部分，这些控制信号用于当两个端口的访问发生了冲突时而进行的仲裁以及连接多个所谓中断及信号灯的附加功能而进行的扩展位宽度中。

双端口 SRAM 的两端何时进行访问是不可预测的，在一端正在更新存储器单元的内容而另一端希望读出同一地址的情况下，后一个访问需要等待。为此，需要预备/BUSY 信号。

当连接多个双端口 SRAM 时，如果各个访问仲裁逻辑单独进行仲裁，那么在非常接近的时间内双方的访问发生了冲突的情况下，有的器件将赋予 LEFT 端口访问的权限，而向 RIGHT 端口返回/BUSY 信号；相反，有的器件会赋予 RIGHT 端口访问的权限，而向 LEFT 端口返回/BUSY 信号。为此，利用主/从（Mater/Slave）功能、以从属器件追随主器件的仲裁功能的判断结果而进行设计。

决定器件的主操作/从操作的是 M / S 信号。如果 M / S 为高电平，则为主器件；如果 M / S 为低电平，则为从属器仵。主器件的/BUSY 信号为输出引脚，而从属器件的/BUSY 信号为输入引脚。

● 5.1.2　CY7C019 的引脚配置

CY7C019 的引脚配置如图所示，它是 100 引脚的 TQFP 封装，是左右对称的。

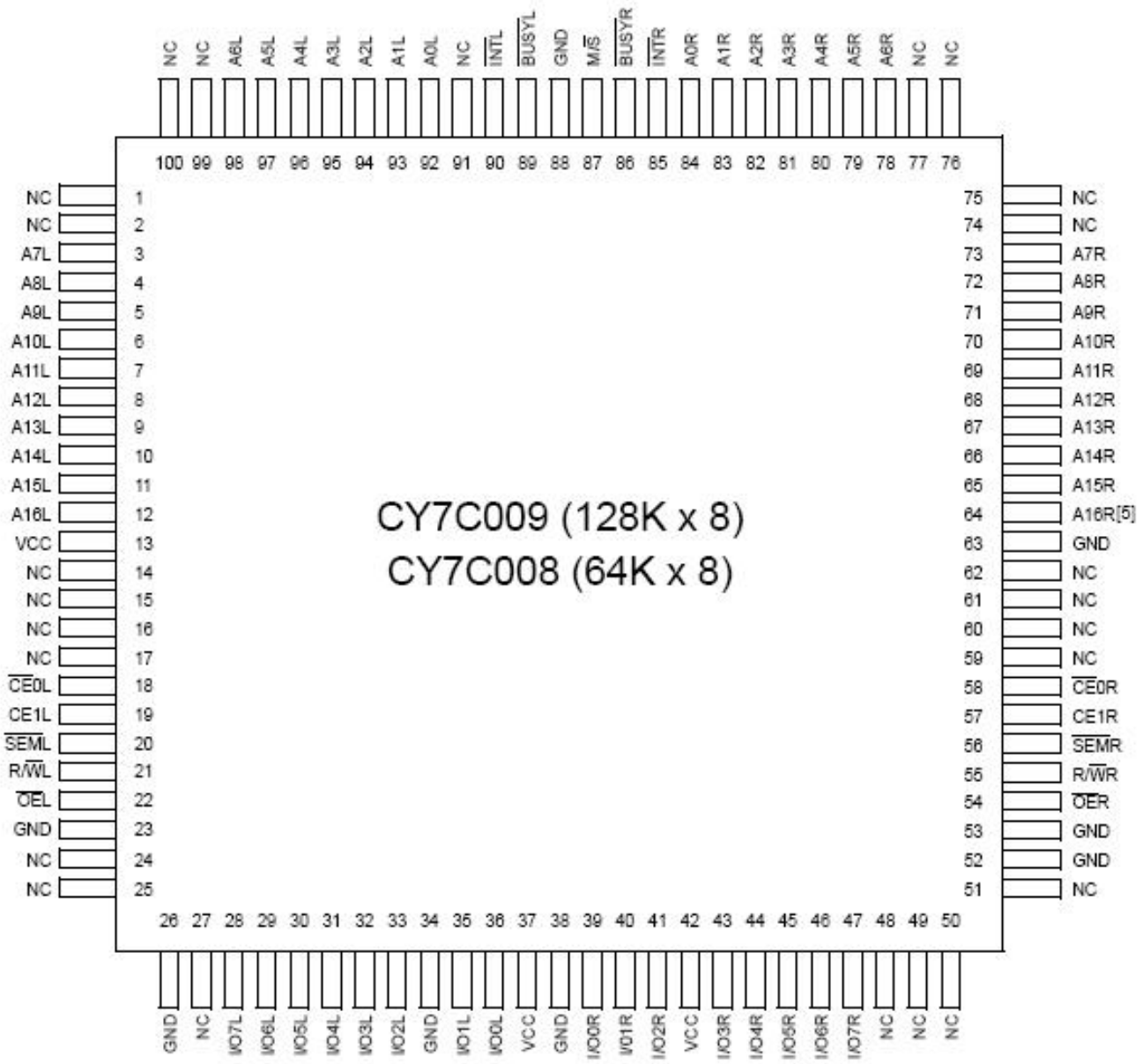


图　　CY7C019 的引脚配置

● 5.1.3　CY7C019 的信号线

双端口 SRAM 具备所有用于存储器存取的信号，分别成对地存在，平等地处理两端口的操作。因此，双端口 SRAM 为了使用方便，两个端口分别称之为 LEFT 端口和 RIGHT 端口，而它们的信号名称以 L 及 R 字符加以区分。但是，因为在功能上两个端口是完全等价的，所以在以下的说明中，我们省略了 L 及 R 符号。

▲ A0～A16（地址总线）

这是地址总线，CY7C019 因为是 128K×9 位的双端口 SRAM，因而有 17 根地址总线。与异步 SRAM 相同，没有突发传输及刷新等功能，因此可以任意使用地址引脚作为 LSB，但一般都将 A0 作为 LSB 使用。

▲ I／O0～I／O8（数据总线）

这是双向数据总线，CY7C019 的数据宽度为 9 位，因而具有 9 根数据线。相同系列的 CY7C018 数据宽为 8 位，所以除了 I/O9 引脚被省略外其他都是一样的。

▲ /CE0、CE1（Chip Enable0／1，芯片使能 0／1）

这是芯片使能信号。当/CE0 为低电平（低于 VIL）且 CE1 为高电平（VIH 以上）时，器件处于选择状态，针对所赋予的地址进行读／写操作。

▲ /OE（Output Enable，输出使能）

将 I／On 设置为输出模式，读出所指定地址的数据。当/BUSY 为低电平时，即使/OE 有效，数据也是不确定的，因此数据读操作必须等待/BUSY 为高电平。

▲ R／W（读／写）

进行写操作时，R／W 为低电平，与/CE0、CE1（以下总称为 CE）同时使用。当 CE 有效并且 R／W 为低电平时，进行写操作；当 CE 无效或者 R／W 为高电平时，数据总线上的数据被写入。

R / W 可以在 CE 之后变为低电平，但是，因为只有到 R / W 成为低电平才可进行写操作，所以此时数据总线上将输出数据。为使数据不起冲突，在将 R / W 设置为低电平之后，需要等待数据总线变为高阻抗状态，所以应该设计为由外部电路驱动数据总线。

▲ /BUSY（忙）

这是用于总线仲裁的信号，该信号的方向根据 M / S 引脚的状态而变化。当 M / S 为高电平时，为主模式，/BUSY 引脚为输出。此时，在其中一个端口要进行存取而另一端口已经在同一地址进行存取操作的情况下，后至的那个端口的/BUSY 信号有效。

▲ /SEM（信号灯寄存器存取，Semaphere Register Access）

这是与双端口存储器本身的功能没有直接关系的附加功能，当 CY7C019 中组合了双端口处理器系统时，除了拥有方便的存储器的功能外，还具各 8 个信号灯寄存器。

在对信号灯寄存器进行存取时/SEM 是有效的。此时，地址低位的 3 位作为信号灯寄存器编号来使用。/SEM 有效时 CE 信号不得有效（必须具备/CE0 为低电平且 CE1 为高电平这样的条件）。我们将在后面详细介绍信号灯功能。

▲ /INT（中断输出）

这也是与双端口存储器本身的功能没有直接关系的附加功能，在组合了双端口处理器系统的情况下，大多利用中断功能互相传递状态变化以及请求等，因此，在 CY7C019 中预置了相互中断的功能。

如果从 LEFT 端口向存储器的最高位地址（1FFFFh 地址）进行写人操作，则 RIGHT 端口的/INT 有效；如果从 RIGHT 端口读取该地址，则/INT 无效。相反，如果从 RIGHT 端口向最高位－1 地址（1FFFEh 地址）进行写人操作，则 LEFT 端口的/INT 有效；一旦 LEFT 端口读取该地址，则/INT 无效。

因为除了/INT 有效 / 无效以外，其他地址处理方式相同。所以，对于传递 1 字节的指令，利用该地址是非常方便的。如果不利用中断功能，那么该地址不进行特殊的处理，可作为普通的双端口存储器使用。

●5.1.4　CY7C019 的基本操作功能

CY7C019 的基本操作可以分为读操作、写操作、BUSY 状态、中断功能、信号灯功能以及主/从操作六种，下面我们对 CY7C019 的基本操作进行说明。

▲　读操作

图表示了双端口 SRAM 读操作的波形。与异步 SRAM 相同，确定地址后，在/CE0 为低电平、CE1 为高电平时器件被选择，通过 R / W 为高电平和/OE 为低电平，确定读操作状态，从而读出数据。而主机方面只要提取该数据即可。

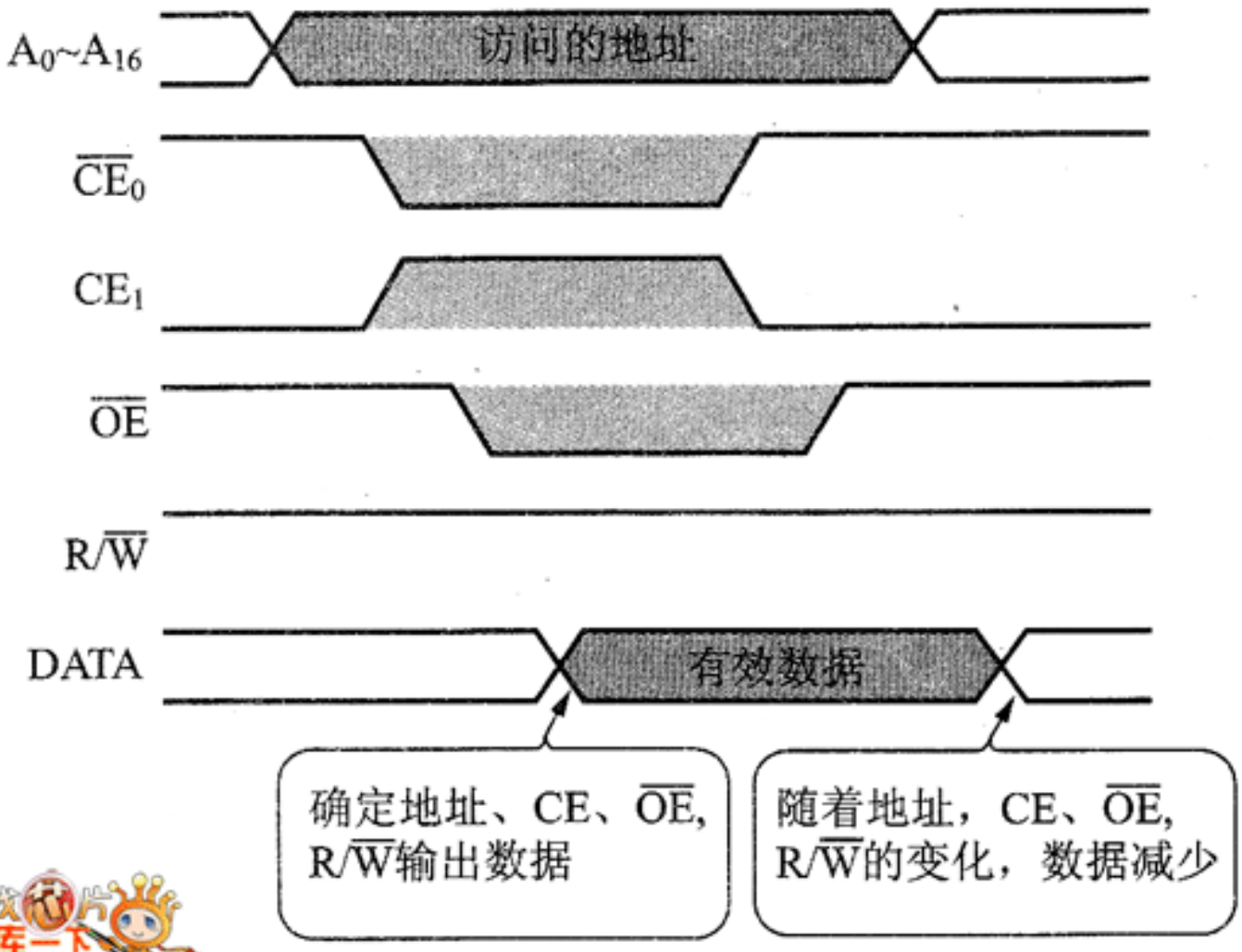


图 双端口 SRAM 的读周期

▲　写操作

图表示了双端口 SRAM 写操作的波形，从图形可知，也是同异步 SRAM 相同的操作。在该示例中，/OE 仍然无效，先确定 R / W 信号后，通过 CE 信号进行写入操作。图中/CE0、CE1 虽然同时发生变化，但也可以其中一个信号保持有效，另一个信号有效或者无效都行，可以在无效的时序中进行写人操作。

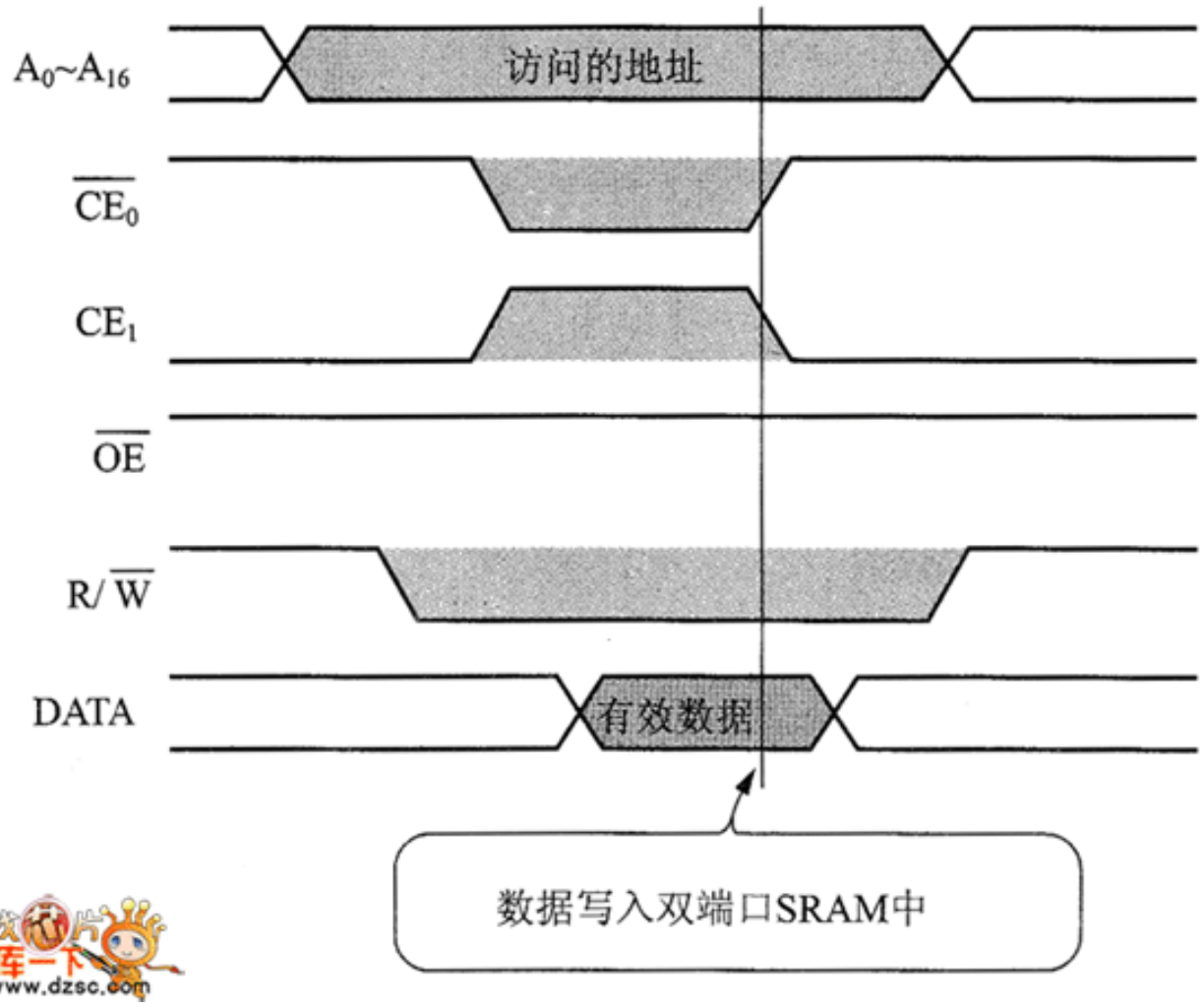


图　双端口 SRAM 的写周期

首先让 CE 有效，然后通过 R / W 进行写入的方法当然也是可以的，在这种情况下，是在 R/W 的上升沿进行写入操作的。

▲　/BUSY 状态

双端口 SRAM 虽然可以同时进行来自两个端口的存取操作，但不可以在同时对同一地址进行存取操作（发生冲突）。在一个端口完成存取操作之前，另一个端口必须等待。我们利用/BUSY 信号完成这种功能。

基本的操作准则是先到者优先，先进行了存取操作的一端优先进行操作，后到的端口的/BUSY 信号有效。一定时间以内发生了来自两个端口的存取请求时，CY7C019 虽然只能有一个端口的/BUSY 信号有效，但此时并不能保证哪一个端口的/BUSY 信号有效。

图 1 表示了由左右两个端口同时对双端口 SRAM 同一地址进行存取操作时的操作概况。在示例中，假定由 LEFT 端口首先访问、在其访问过程中又发生来自 RIGHT 端口的访问，此时，RIGHT 端口的/BUSY 信号有效，开始等待 LEFT 端口完成操作。

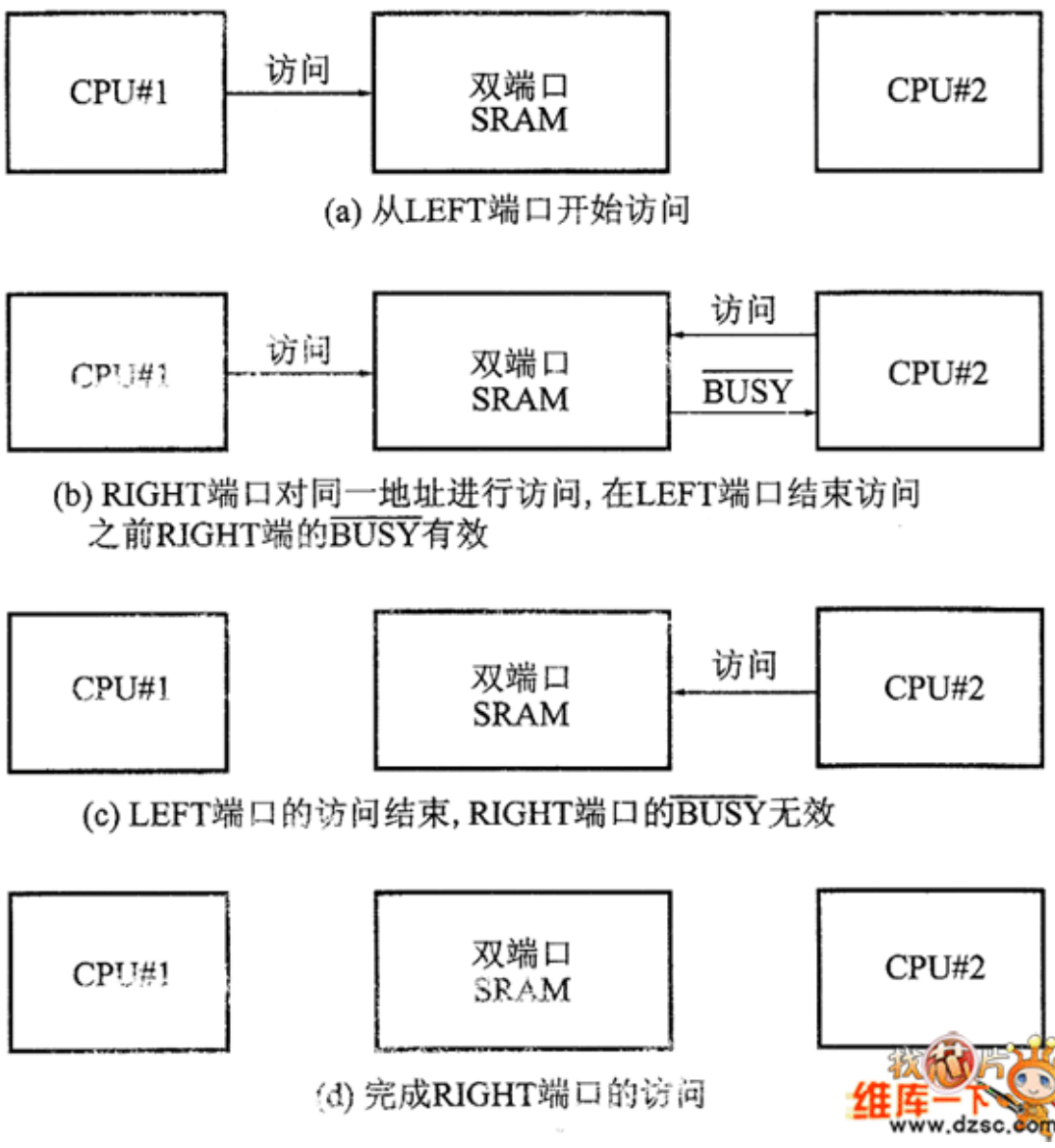


图 1　对双端口 SRAM 同时进行存取操作

更加具体的波形显示如图 2 所示，与刚才的示例相同，是 LEFT 端口首先进行访问，在使 RIGHT 端口等待之后，又发生了来自 LEFT 端口的访问，图 2 显示了这一过程的波形，由此可知 LEFT、RIGHT、LEFT 端口交互进行访问的情况。

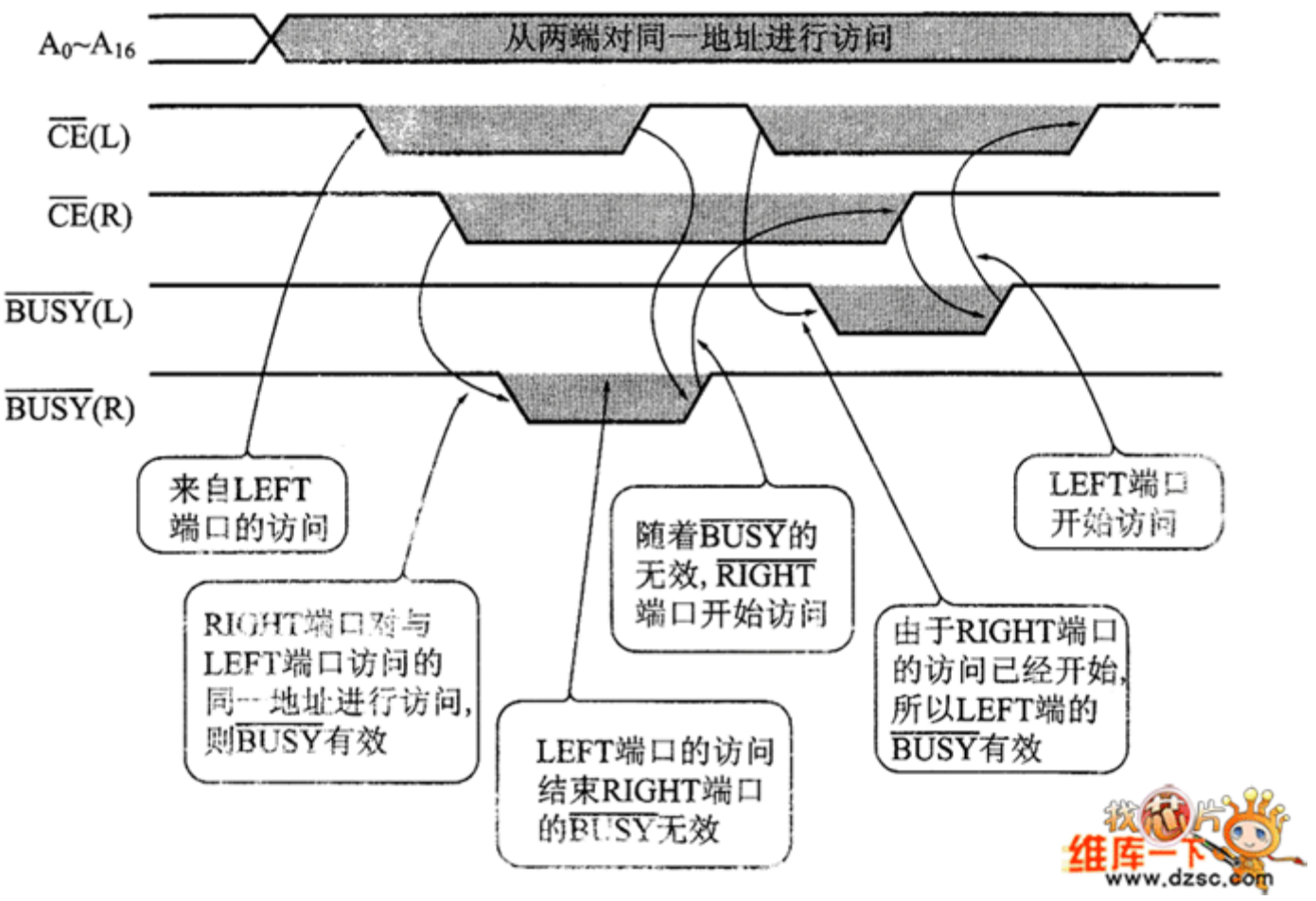


图 2　双端口 SRAM 的仲裁操作

▲　中断功能

在利用双端口进行多个处理器间通信的情况下，为了传递开始处理请求以及结束的通知等信息，经常相互间中断某操作。CY7C019 就是为了这个目的而增加了中断功能。

说是中断功能，其实与已经描述过的写操作、读操作没有什么不同。

图是中断操作的示例。在该示例中，LEFT 端口中断了 RIGHT 端口的操作。如果由 LEFT 端口向 1FFFFh 地址写人数据（数据为任意），则 RIGHT 端的/INT 输出有效（为低电平）。如果连接于 RIGHT 端的 CPU 等接收到该指令，由 RIGHT 端口读取 iFFFFh 地址，那么，/INT 输出无效，此时 RIGHT 端读出由 LEFT 端口写入的数据。

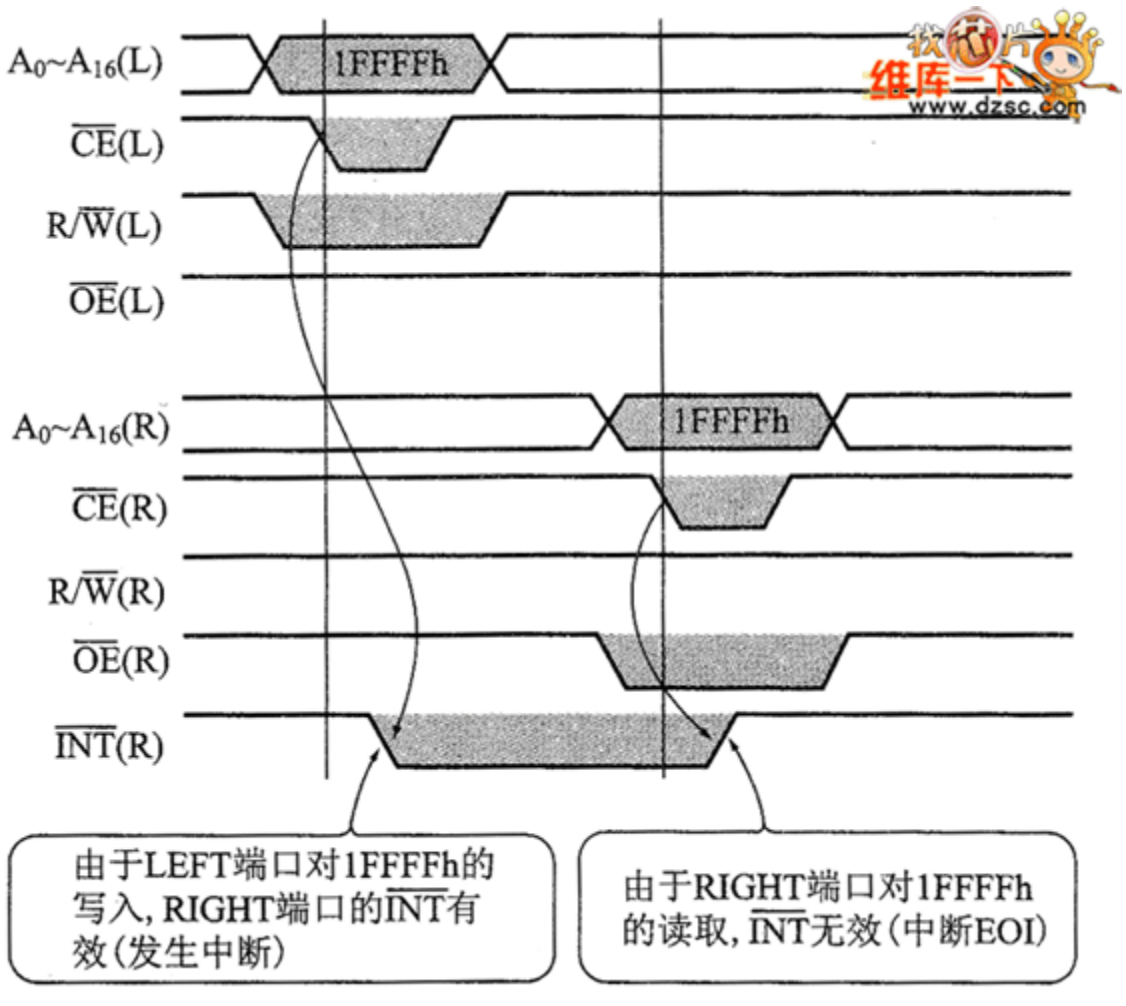


图 CY7C019 中断功能的操作

●5.1.5　同步类型的双端口 SRAM

作为同步双端口 SRAM，我们以 CY7C09199 为例进行说明。CY7C09199 与 CY7C019 相同，都是 128K×9 位结构的双端口存储器，其框图如图所示，由图可知，各个信号引脚都是利用时钟进行采样操作的。

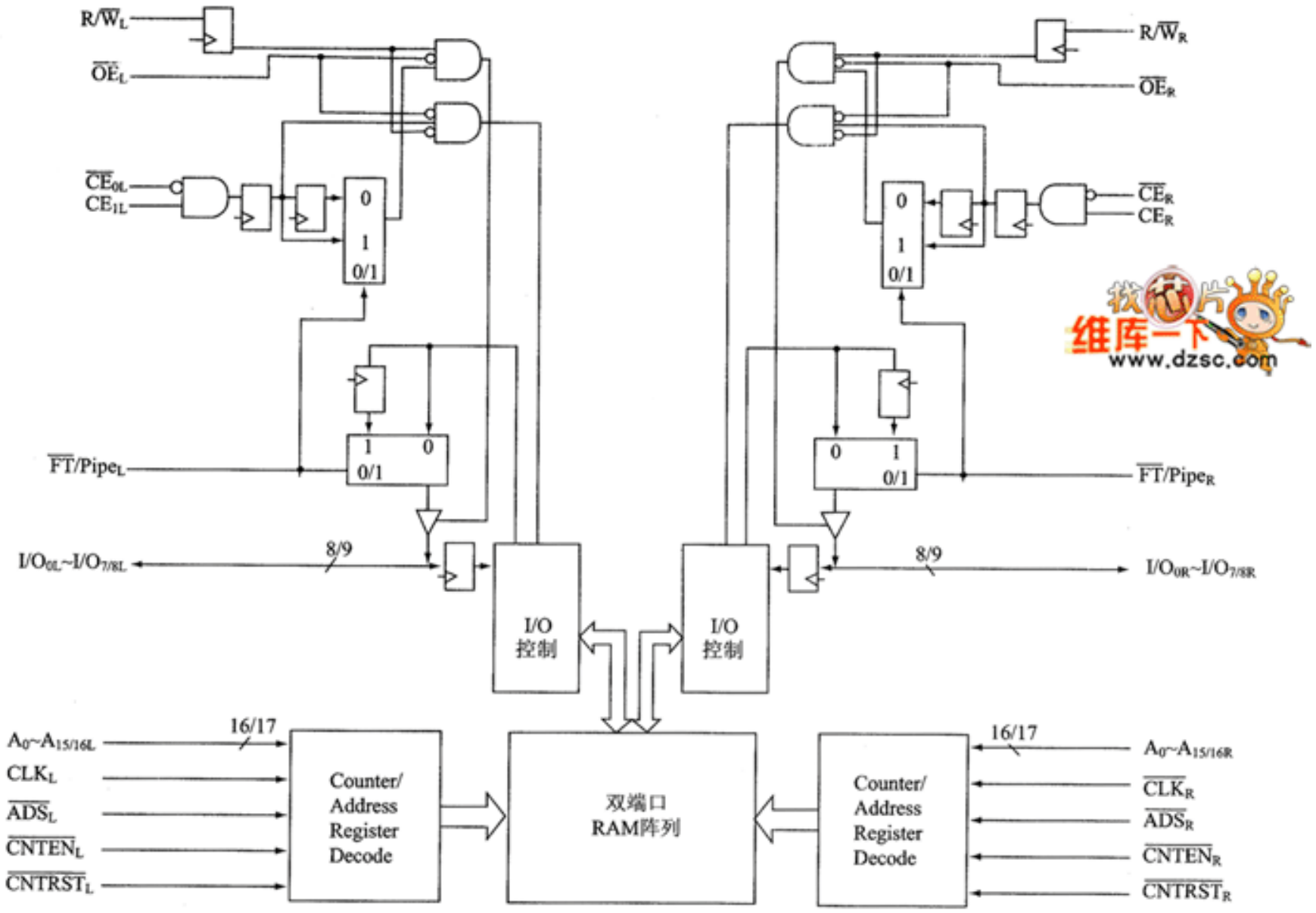


图　CY7C09199 的内部框图

其功能仍然是双端口存储器的功能，只是因为与时钟同步运行，所以不具备异步类型所拥有的仲裁机制。而且信号灯功能也被删除，取而代之的是增加了从最初所赋予的地址开始能够进行一系列读/写操作的计数器功能（Counter/Address Register Decode，计数器/地址寄存器译码器）。FT/Pipe 引脚是同步管道突发式 SRAM 与同步突发式 SRAM 不同的地方，在进行数据读操作时，锁存一次数据之后，通过该引脚选择在下一个时钟中输出（管道操作）还是直接输出（Flow Through，直流）。与同步 SRAM 时的情况相同，管道类型的最高时钟可以把频率取得较高。

● 5.1.6　CY7C09199 的引脚配置

CY7C09199 的引脚配置如图所示，与 CY7C019 相同，是 100 引脚的 TQFP 封装，是左右对称的。

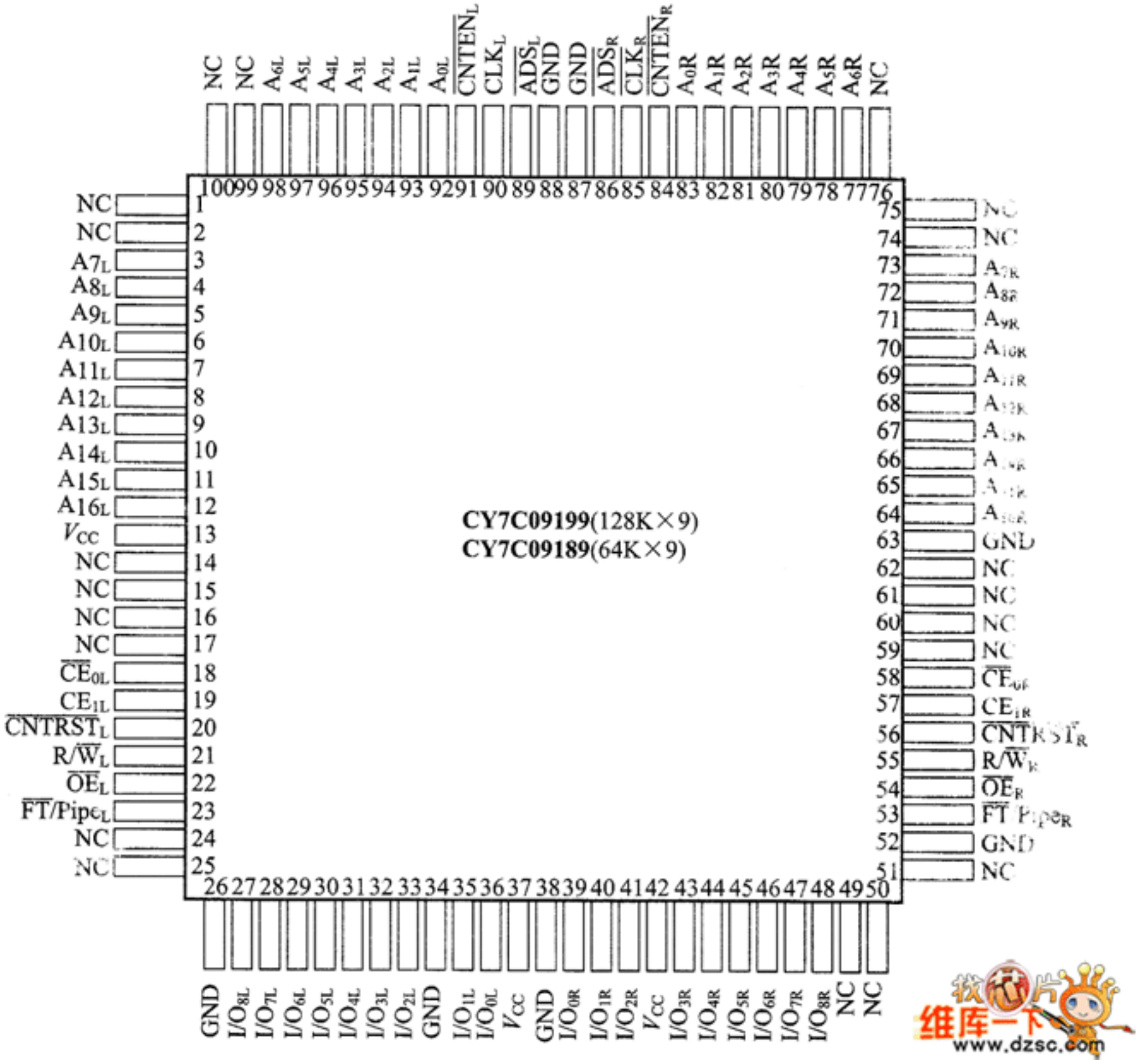


图 CY7C09199 的引脚配置

●5.1.7 CY7C09199 的信号

CY7C09199 的信号也与 CY7C019 的信号相同，具有以左右两个端口为对象的 2 组信号。在此，只要没有特殊的说明，就不区分 LEFT 和 RIGHT 端口进行记述。

CY7C09199 的各个信号时序是由时钟的上升沿规定的，下面只要没有特殊的说明，时钟沿都是表示时钟上升沿的意思。

▲　A0～A16（地址总线）

这是地址总线。正如先前接触过的，CY7C09199 因为是 128K 字×9 位结构的双端口存储器，所以具有 17 根地址总线。

在/CE0、CE1 信号以及/ADS 有效时的时钟沿上地址被锁存。

▲　I／O0～I／O8（数据总线）

这是数据总线。进行数据写操作时，在时钟沿上数据被提取；

在进行数据读操作时，从时钟沿开始读出数据。因此外部电路在下一个时钟沿上提取数据。读操作时，根据 FT／Pipe 引脚，需要注意数据输出时序的改变。

▲　/CE0、CE1（Chip Enable0／1，芯片使能 0／1）

这是芯片使能信号。时钟沿来临后，当/CE0 为低电平（VIL 以下）且 CE1 为高电平（VIH 以上）时，器件处于选择状态，可以进行读／写操作。通过在同一时钟沿上所赋予的 R／W 信号的状态确定是读操作还是写操作，由/ADS 及/CNTEN 决定所访问的地址。

▲　/OE（Output Enable，输出使能）

将 I／On 设为输出模式，然后读出指定地址的数据。只要在数据输出的时序内/OE 为低电平，数据就可以输出。但如果“为高电平，则数据总线（I／O0～I／O8）不被驱动，一直处于高阻抗状态。

▲　R／W（读／写）

进行写操作时，R／W 为低电平与/CE0 及 CE1（以下将/CE0 和 CE1 总称为 CE）一起被应用。在时钟沿上，如果 CE 有效、并且 R／W 为低电平，则为写操作；如果 CE 有效、而 R／W 为高电平，则为读操作。与异步双端口 SRAM 不同，R／W 由于是在时钟沿上被采样的，因此必须与 CE 信号确定在同一个时钟沿上。

▲　/CNTEN（Counter Enable，计数器使能）

CY7C09199 的地址锁存器也具有计数器的功能，可以从最初所访问的地址开始按序访问连续的区域。与同步突发式 SRAM 的突发传输相似，同步突发式 SRAM 只能对低位的 2 位进行计数，而 CY7C09199 能够对所有的地址进行计数，这是两者的不同之处。

当需要该功能时可以利用/CNTEN 信号，在上升沿上 CE（/CE0 和 CE1）有效的状态下，如果/CNTEN 有效，则处于这种功能模式，每当时钟沿来临，地址就自动进位，然后输出下一地址的数据。如果/CNTEN

无效，则地址将不再进位。

另外，当/ADS 有效时，/CNTEN 无效。

▲ /ADS（Address Strobe，地址选通）

该信号用于将所赋予 A0～A16 的地址作为要访问的地址存储。在时钟沿上如果 CE 有效、并且/ADS 也有效，则双端口 SRAM 将 A0～A16 作为访问地址锁存于内部。

如果在 CE 有效的时钟沿上/ADS 无效，则将目前的地址作为访问对象，此时，只要/CNTEN 信号有效,地址就可自动进位。

▲ /CNTRST（Counter Reset，计数器复位）

在时钟沿上如果 CE 有效、并且/CNTRST 也有效，则访问地址恢复为 0。/CNTRST 的操作与/CNTEN 及/ADS 没有关系，地址被强制恢复为 0。因此，例如保持 CE 及/CNTEN 一直有效，只要使定期/CNTRST 有效，就可以像循环缓冲器那样，连续访问同一地方。

▲ /OE（Output Enable，输出使能）

这是在读取时必须有效的信号。根据框图可知，该信号是与时钟异步的。当读 / 写操作交互发生时，需要注意有效 / 无效的时序。

▲ FT / Pipe（直流 / 管道）

根据框图可知，CY7C09199 在进行读操作时，是将由存储器单元输出的数据原封不动地输出到 I / O 引脚的，它具有直流（Flow - Through）和管道两种模式。直流模式是在赋予地址的下一个时钟输出数据；而管道模式是将从存储器单元输出的数据暂时提取到内部的锁存器然后输出的。

输出数据的时序包括了通过内部锁存器的那部分时间，因此管道模式要慢一个时钟。在时序方面，管道模式较严格，相对来说直流模式在时序规定上比较宽松。在存取速度方面，当进行单次访问时，直流模式需要一个时钟，而管道模式所需要的 2 个时钟就是其弱点。当利用/CNTEN 功能连续访问某一区域时，由于两种模式只差一个时钟（例如，如果传输 16 字节，则直流模式需要 16 个时钟，与之相对应，管道模式需要 17 个时钟），因而从整体来看，其性能差别不大。

● 5.1.8　CY7C09199 的存取操作

CY7C09199 的操作，将根据除/OE 以外其他的各个信号在时钟沿上处于何种状态而确定。

表 1 表示读 / 写操作，表 2 表示地址锁存 / 进位机制的操作条件。

表 1 CY7C09199 的读 / 写操作模式

输入引脚					输出输出引脚	操作模式
$\overline{OE}$	CLK	$\overline{CE_0}$	CE_1	R/$\overline{W}$	$I/O_0 \sim I/O_8$	
X	↑	H	X	X	高阻抗	非选择状态
X	↑	X	L	X	高阻抗	非选择状态
X	↑	L	H	L	数据输入	写操作
L	↑	L	H	H	数据输出	读操作
H	X	L	H	X	高阻抗	输出禁止

表 2 CY7C09199 的地址计数器控制

输入引脚						输入出引脚	模式	操作状态
$A_0 \sim A_{16}$	上一个时钟沿时的 $A_0 \sim A_{16}$	CLK	$\overline{ADS}$	CNTEN	$\overline{CNTRST}$	$I/O_0 \sim I/O_8$		
X	X	↑	X	X	L	D(0)	复位	地址计数器恢复为 0
A(n)	X	↑	L	X	H	D(n)	负载	锁存外部地址
X	A(n)	↑	H	H	H	D(n)	保持	保持上一时钟的地址
X	A(n)	↑	H	L	H	D(n+1)	进位	启动计数器

所谓读 / 写操作的输出禁止，表示虽然在内部进行读操作，但因为外部输出缓冲器关闭而不能输出数据。

另外，地址计数器控制表中的 D（n）表示地址 A（n）的数据，表中表示的操作都是直流模式下的操作。在管道模式的情况下，请将“上一个时钟沿上的 A0～A16”替换为“再上一个时钟沿上的 A0～A16。

▲读／写搡作

图中表示了存取操作中的一个例子，该示例中的操作是管道模式（FT／Pipe 引脚为高电平）下的操作，它按照读／写／读这样的顺序进行存取。

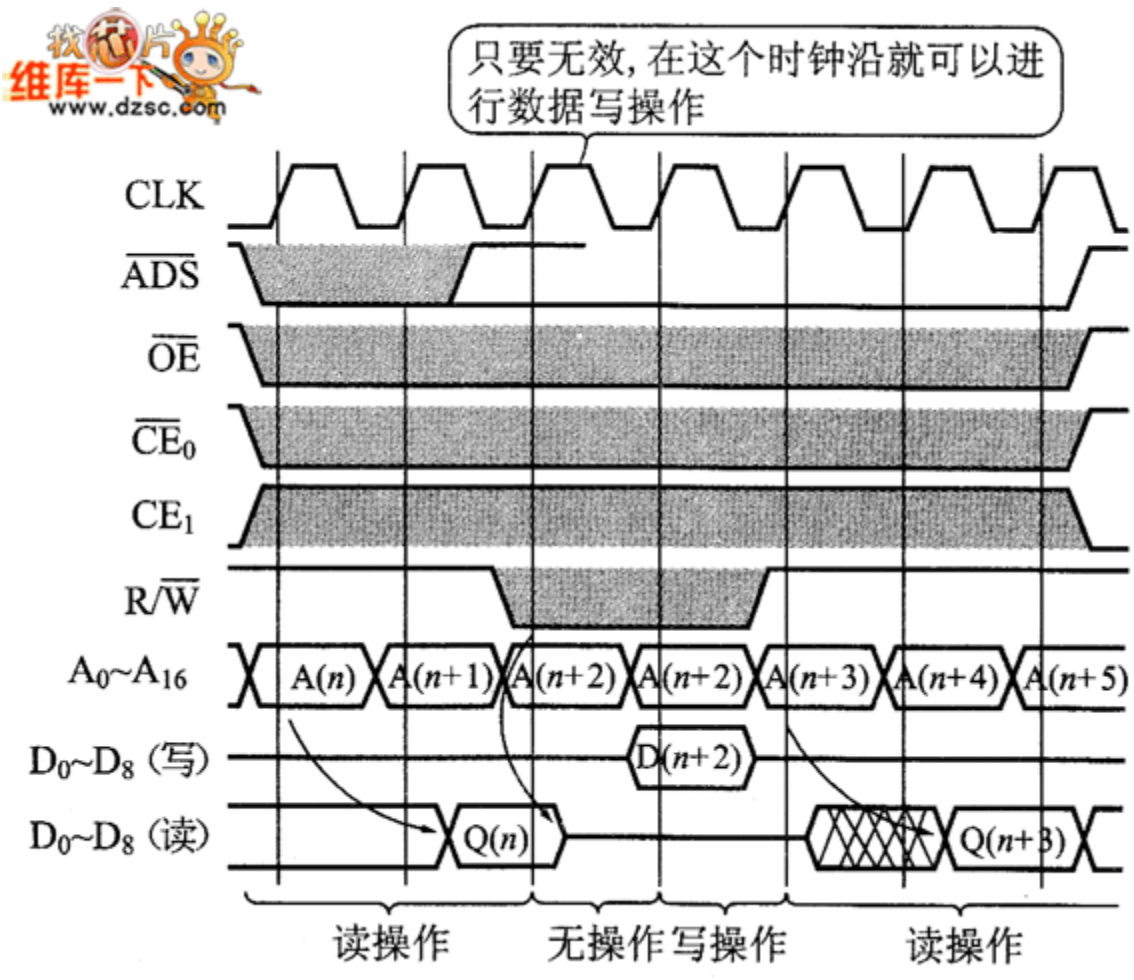


图　　同步双端口 SRAM 的存取操作示例

直流模式下的读／写操作，因为数据的输出是被一个个时钟前置的，所以在赋予地址的下一个时钟沿上确定数据。

首先，在最初的时钟上 CE 有效，器件处于被选择的状态。因为 R／W 为高电平，所以操作是读模式，而又因为/ADS 有效，因而将 AO～A16 作为所访问的地址进行提取。在这个例子中，是在下一个时钟中改变地址的，这只是希望表示管道模式操作才这样描述的。

因为是在赋予地址的下一个时钟沿开始输出数据，所以，外部电路在所赋予指定地址的 2 个时钟周期后的时钟沿上提取数据。

在这次的示例中，在 2 个时钟之后的时序内将 R／W 设为低电平，则转换为写模式。如果在这个时钟沿之前 CE 无效，则因为不能输出 Q（n）的数据，所以在外部电路中驱动 I／OO～I／O8，可以在该时钟沿上写人数据。但是此示例中，由于 CE 一直有效，所以正在输出来自双端口 SRAM 的数据，因而不能写入数据。只能在此等待一个时钟，在下一个时钟沿中写入数据。

写操作结束后，如果 R／W 再恢复为高电平，则变为读模式。从高电平的 R／W 被采样的时钟沿开始，2 个时钟之后确定数据。

▲ 地址计数器模式

如前所述，CY7C09199 的地址锁存器也可以作为计数器进行操作。当器件处于读模式、/ADS 为高电平时，如果/CNTEN 有效，则地址进位，将自动访问下一个地址。

图表示了上述过程，该图是 CY7C09199 为管道模式（FT / Pipe 为高电平）下的操作示例。

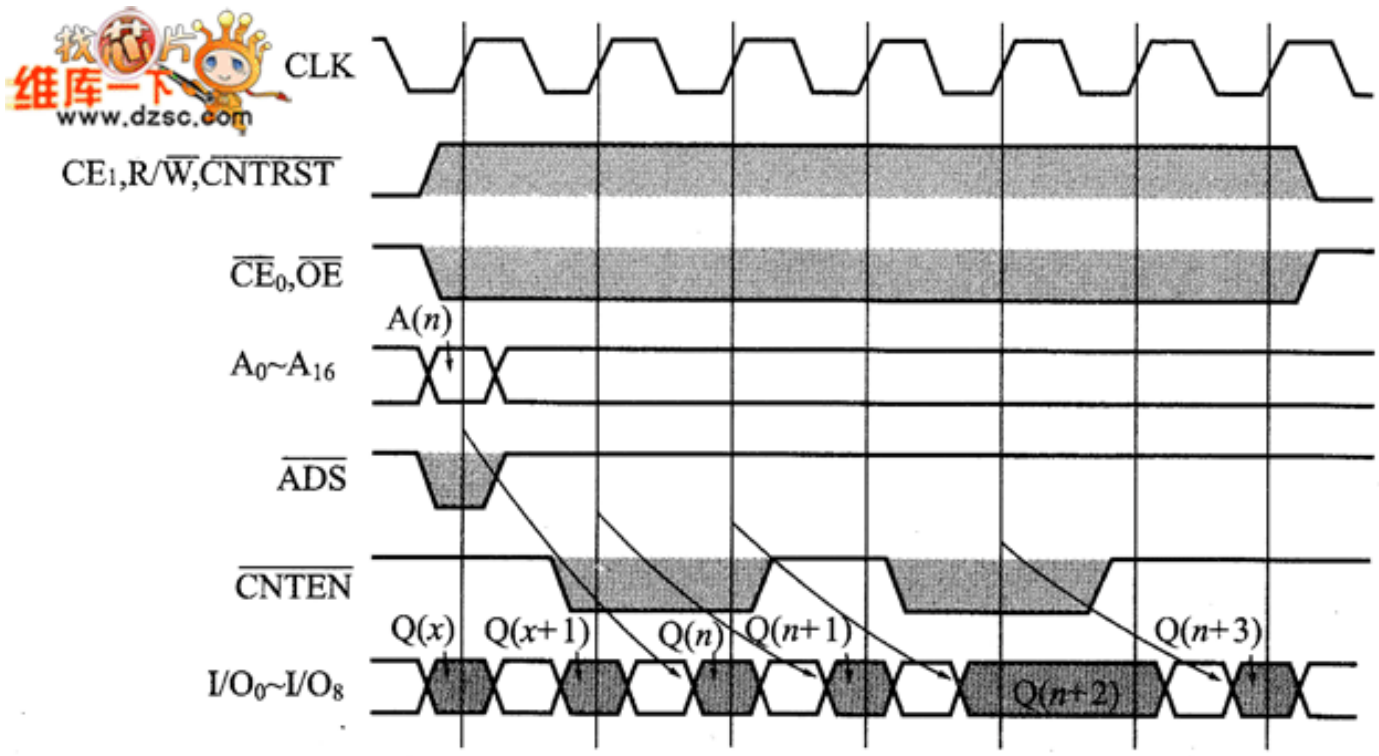


图　CY7C09199 的连续区域的读操作

首先，在最初的时钟内/ADS 有效，赋予地址的初始值（n 地址）（在此，只要/CNTRST 有效，则地址将自动为 0）。在这里所赋予的地址的数据将在 2 个时钟后输出。

然后，在下一个时钟内因/CNTEN 有效，使下一个地址（n＋1 地址）成为访问对象，而后在其下一个时钟内指示（n＋2）地址的访问。

在此，如果取消/CNTEN，则因为将中止地址的进位，所以如图所示，将持续输出（n＋2）地址的数据。/CNTEN 再次有效，则将在 2 个时钟之后输出（n＋3）地址的数据。

5.2　FIFO 存储器

FIFO 是 First－In First-Out 的缩写，翻译成中文就是“先入先出”。FIFO 存储器可以独立进行输人输出，也可以看成是一种双端口存储器，它确实与双端口存储器相同，具有两个端口，但它与双端口存储器最大的不同就是一个端口专门用于写入操作，而另一个端口专门用于读取操作。而且，

因为数据是按照写人的顺序被读出的，因而没有地址引脚，这也是与双端口存储器的不同之处。

从软件的角度观察 FIFO，如图 1 所示，类似底部装有水龙头的箱槽，由上部写人数据，下面输出数据。无论箱槽是空是满都有各自相应的标识，可以从外部了解其状况。

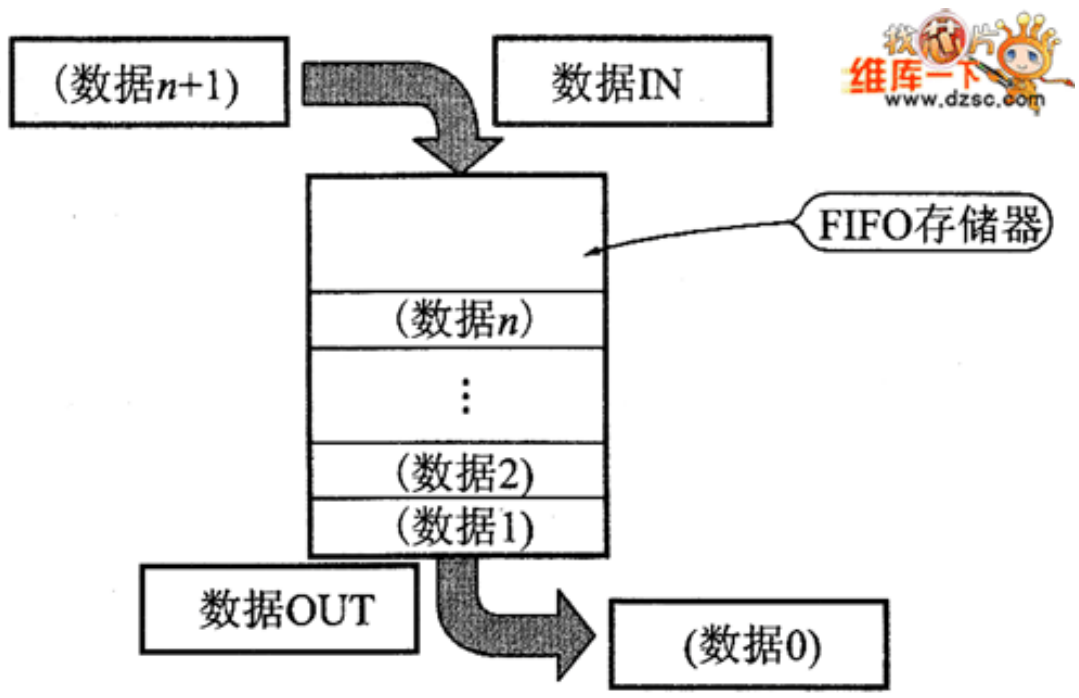


图 1　FIFO 存储器的印象（之一）

但是，作为实际上的硬件产品，并不存在这样类似移位寄存器的产品。如图 2 所示，将 FIFO 存储器设想为环形就比较容易理解。它包括表示读取数据位置的指针以及表示写人数据位置的指针，读 / 写操作分别完成一次存取操作后，地址向前进一位。当然，如果读指针超越写指针、或者写指针环绕一周越过读指针，则操作将出现问题，因此将其分块进行处理。

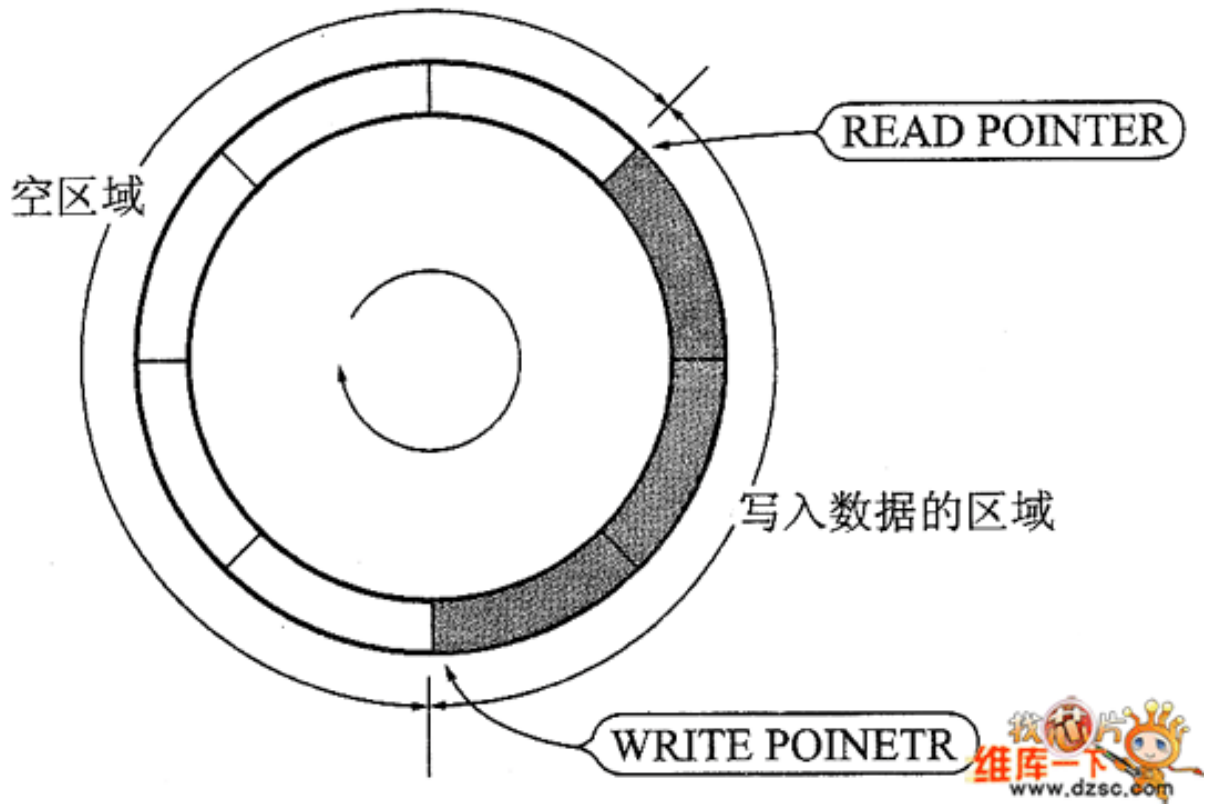


图 2　FIFO 存储器的印象（之二）

● 5.2.1　实际的 FIFO 存储器

作为实际的 FIFO 存储器，我们以 Cypress 公司的 CY7C419 为例进行说明。CY7C419 是 256 字×9 位结构的 FIFO 存储器，其引脚配置如图 1 所示。在与 CY7C419 相同的系列中，还包括内部结构为 512 字×9 位（CY7C421）以及 1K 字×9 位、2K 字×9 位、4K 字×9 位（分别为 CY7C425 / 429 / 433）的产品。由于 FIFO 存储器没有地址引脚，因而无论哪种产品都具有完全相同的引脚配置，所以该存储器可以互换使用。

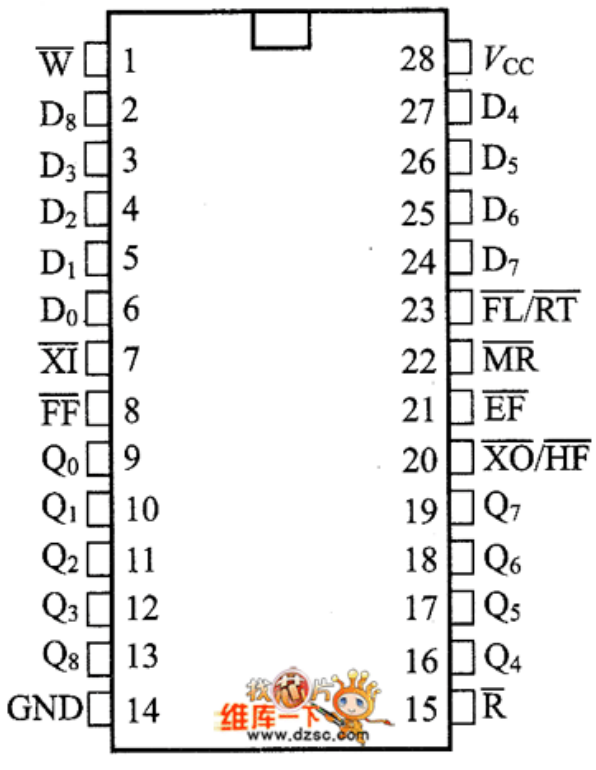


图 1 CY7C419 的引脚配置

CY7C419 的框图如图 2 所示，可以看出这是与刚才的 FIFO 印象图非常相似的形式。在前面的图中没有出现的是处于下半部分的复位逻辑、标识控制电路以及扩展逻辑这三项。对此，我们将进行简单的补充说明。

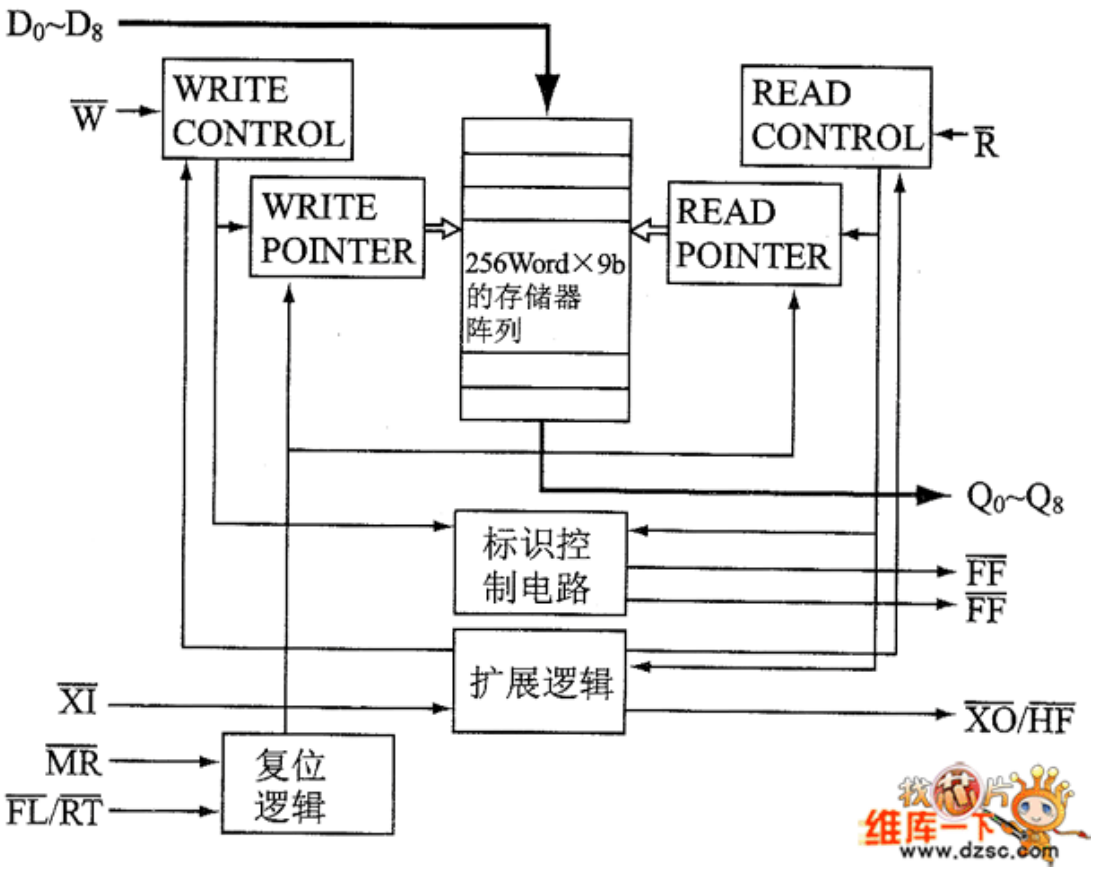


图 2 CY7C419 的内部框图

▲　复位逻辑

如前所述，FIFO 存储器没有地址引脚，访问的地址由存储器内部的读指针及写指针进行管理，从外部既不能读出也不能置换该指针。

所以，在接通电源及系统复位的情况下，决定 FIFO 存储器初始状态的就是复位逻辑,通过/MR（Master Reset，主复位）及 FL / RT（First Load，优先加载 / Retransmit，重传）引脚进行各个指针的初始化。

▲　标识控制电路

虽然 FIFO 存储器的读指针及写指针等不能被读出，但如果不了解数据是否已经写人或者数据是否已满，那么读取端可能进行没有任何数据的读操作，而写人端也可能在数据已满的地方仍在写人数据。

为此，在 FIFO 存储器中设计了表示“缓冲器为空（BufferEmpty）”及“缓冲器已满（Buffer Full）”的信号。由标识控制电路控制这些信号。

▲　扩展逻辑

设计扩展逻辑是为了在连接若干 FIFO 存储器情况下可以更多地存放数据。CY7C419 通过/XI（Expansion IN，扩展输入）、/XO（Expansion OUT，扩展输出）以及/FL（First Load）信号进行控制，/XO 输出与相邻器件的/XI 输入相连接，最后的器件的/XO 输出与最前面器件的/XI 输出相连接，以这样的形式构成环形缓冲器。

● 5.2.2　CY7C419 的信号

下面我们来观察 CY7C419 的各个信号。

▲ D0～D8（数据 IN）

这是数据输人引脚。FIFO 存储器类似单向缓冲器，所以这种引脚是输人专用的引脚。提供给 D0～D8 的数据在/W 的上升沿被存放于 FIFO 存储器中。

▲ /W（Write）

这是向 FIFO 存储器的数据写信号。在/W 的上升沿（由低电平跳变为高电平的过程中）赋予给 D0～D8 的数据被添加到 FIFO 缓冲器的有效数据的末尾，此时也可以进行 WRITE POINTER（写指针）及标识类信号（/EF、/FF 及/HF）的更新。/EF 在 W 的上升沿更新，/HF 及/EF 在 W 的下降沿更新。

另外，/FF（Full Flag，全满标志位）标识有效时，不能写人新的数据（忽略写入数据）。

▲ Q0～Q8（数据 OUT）

这是数据的输出引脚，专门用于输出数据。如果/R 有效，则 FIFO 存储器的先头数据将出现在 Q0～Q8 上。

▲ /R（Read）

这是 FIFO 存储器的读信号。如果/R 有效，则 FIFO 存储器前的先头数据将出现于 Q0～Q8 上，此时，也更新器件内部的 READPOINTER（读指针）及被外部输出的标识类信号（/FF、/BF 及/HF）。/HF 及/FF 在/R 的上升沿更新，/EF 在 R 的下降沿更新。

▲ /MR（Master Reset，主复位）

对 FIFO 存储器内部所有的指针及标识类信号进行初始化。由于 FIFO 存储器为空，所以/EF 有效，/FF 无效。CY7C419 包括独立 / 宽度扩展模式（Standalone / Width Expansion Modes）及深度扩展模式（Depth Expansion Mode）两种，在哪一种模式下进行操作，是由/MR 有效时/XI 及/FL 引脚的状态决定的。

▲ /FL、/RT（First Load / Retransmit，优先加载 / 重传）

这是根据操作模式不同而具有不同功能切换的引脚，如果是深度扩展模式则为/FL 引脚，决定在所连接的若干个器件中从哪个器件开始操作（复位后，是否处于最初存放数据的位置）。如果是独立 / 宽度扩展模式，则为/RT（Retransmit，重传）引脚，读指针返回物理上的起始位置，可以重新输出数据。

▲ /EF（Empty Flag，空标志位）

这是表示 FIFO 存储器状态的标识引脚。/EF 是在 FIFO 存储器为空时有效的信号，在读取最终数据时的/R 下降沿，/EF 有效，之后在/W 上升时无效。

▲ /FF（Full Flag，全满标志位）

这是 FIFO 存储器已满时有效的信号。在写人最终数据时的/W 的下降沿，/FF 有效，在之后数据读取时的/R 的上升沿无效。

▲ /XI（Expansion IN）

该引脚除了应用于深度扩展模式中接受后面存储器的/XO 信号以外，还应用于复位时决定操作模式中。如果/XI 为低电平，/FL 为高电平，则确定为独立模式。

▲ /XO、/HF（Expansion OUT / Half Full，半满标志位）

这是根据操作模式具有不同功能切换的信号。深度扩展模式的情况下为/XO 输出，为了让成为数据访问对象的器件转移到一个相邻的器件，利用该信号作为传递访问权限的信号进行操作。例如在写操作的方向上，在自己就是访问对象的情况下，当缓冲器已满时，相邻的一个器件就必须切换为写操作的目的地。因此，在对缓冲器的最终地址进行写操作时，与/W 联动，改变/XO 输出，向相邻的器件转移写操作的目的地。

独立模式时为/HF 输出，当向第“存储器的容量÷2＋1”字节进行写人操作时，/HF 在此时刻的/W 的下降沿有效；在已经加人了“存储器容量÷2＋1”字节的数据时，/HF 在此时刻的读操作（读操作之后为存储器容量÷2）时的/R 的上升沿无效。

● 5.2.3　CY7C419 的操作

下面我们来观察 CY7C419 的存取操作。

▲ 读 / 写操作

CY7C419 的读 / 写操作波形如图所示。读操作时的存取时间（10ns）与写操作时的数据建立时间（6ns）是 CY7C419-10 器件的时序。

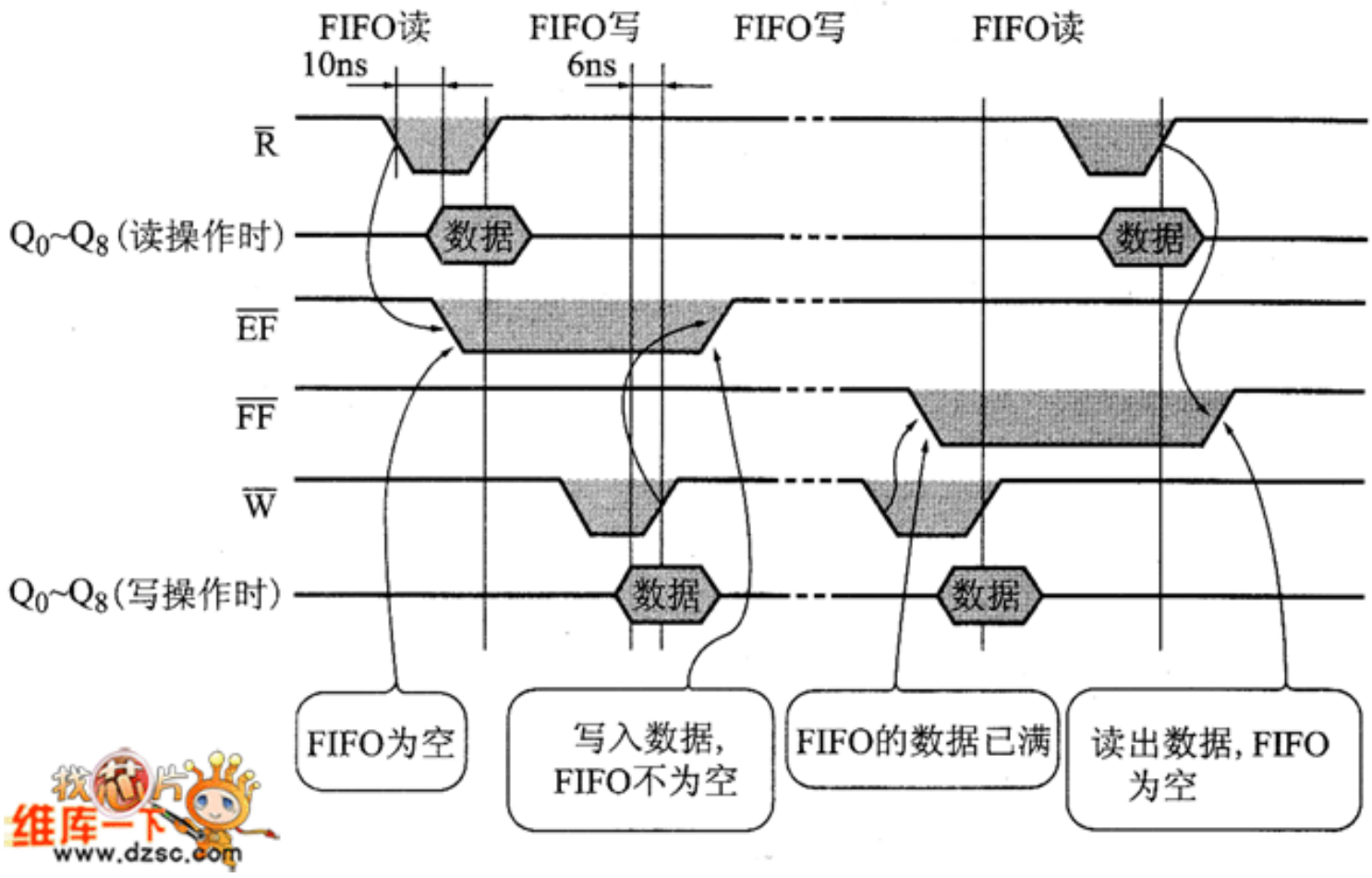


图　　对 FIFO 存储器的存取及标识操作

图的左半部分表示通过读操作使缓冲器成为空的状态后进行数据写操作时的操作。右半部分则是通过写操作，在缓冲器变为已满的状态后进行读操作时的操作。/EF 及/FF 标识的操作可能已经非常明白了，如果读操作单纯让/R 有效，则在 Q0～Q8 引脚上出现 FIFO 存储器的起始数据；而在进行写操作时，是在/W 的上升沿上提取 D0～D8 的数据（在建立时间之前需要确定数据）。另外，在独立模式下的/HF（Half Full）信号除了有效 / 无效的时序是处于缓冲器容量的一半位置以外，其他的与/FF 标识的操作相同。

▲ 深度扩展模式的操作

CY7C419 深度扩展时的连接如图 1 所示，/XO 输出与相邻的/XI 输入连接，复位后只将 FIFO 前面器件的/FL 引脚预先设为低电平，其他都设为高电平。除此之外，数据以及 R / W 等信号都是公用的。CY7C419 在复位中（/MR 有效时）确定是独立模式还是深度扩展模式。

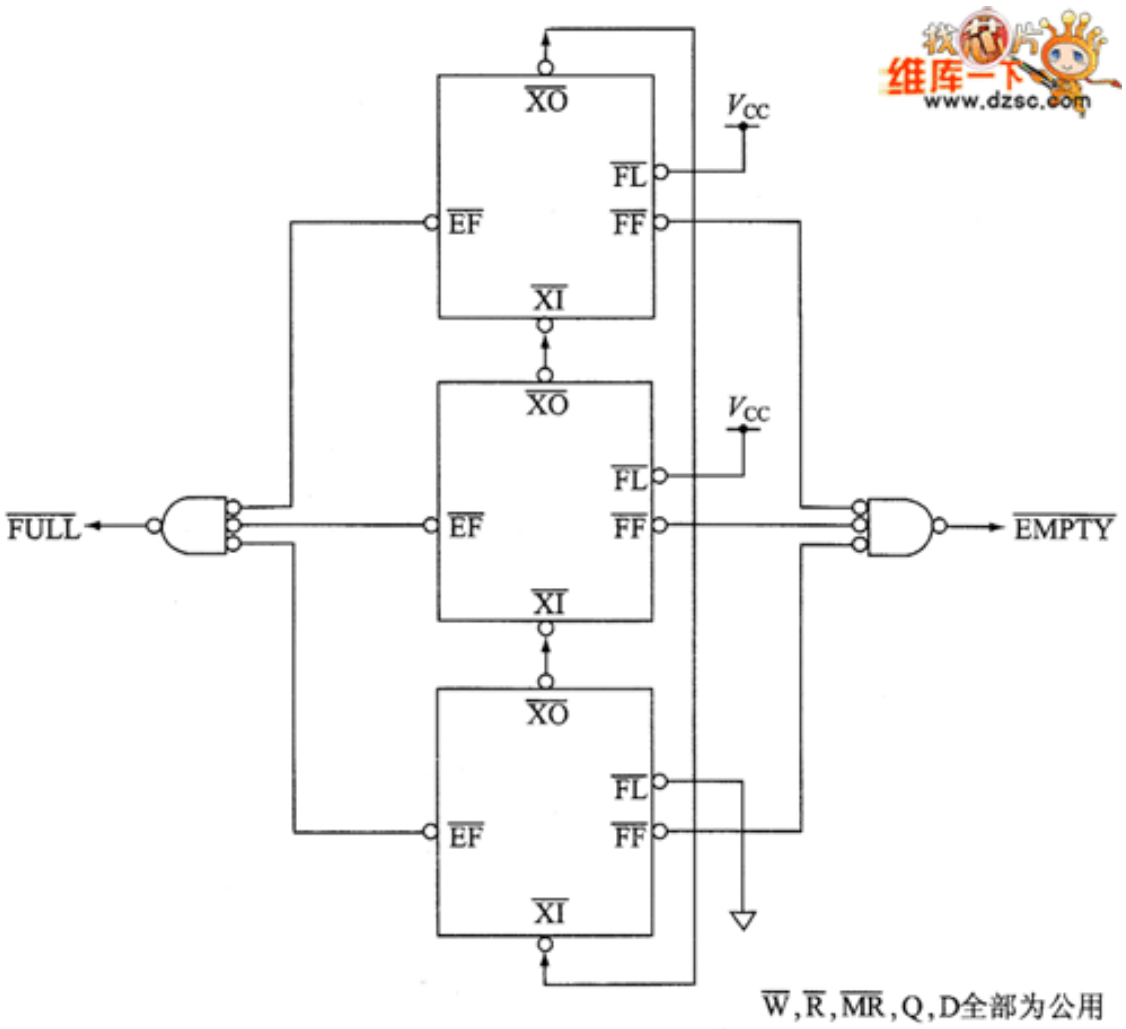


图 1 深度扩展时的操作

复位时，如果/XI 为低电平，而/FL 为高电平，则为独立模式；而当/XI 为高电平时，为深度扩展模式。

图 2 表示深度扩展模式时/XO 信号的操作。最初的读操作是从目前正在访问的器件的最终区域开始进行的波形，可看出/XO 输出是与/R 联动进行的。相邻的器件通过/XI 输入接收/XO 输出，使其移向成为访问对象的器件。

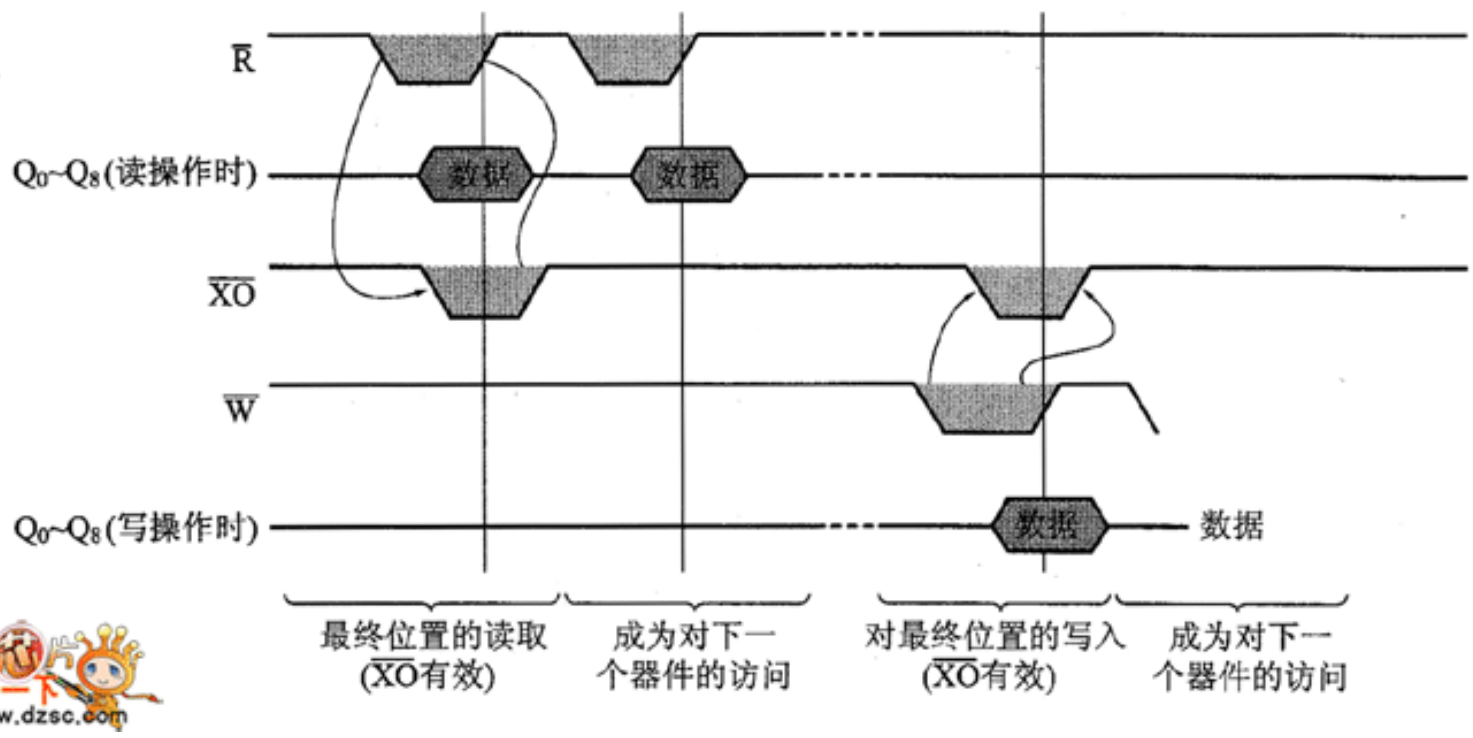


图 2 FIFO 存储器的深度扩展模式的操作

写操作时也是相同的，当向最终区域进行了写操作时，/XO 输出有效，传递着从下一个写访问开始相邻的器件已成为访问对象的信息。

/XO 是写 / 读操作共同使用的，但由于各个器件的/R 与/W 是公用的，所以只要看到/XI 引脚和/R、/W 信号，就可以判断下一个访问对象是否是自己。当然，成为读对象还是写对象是被分别记录的。

例如，当一个 FIFO 存储器的容量为 512 字时，写 / 读 / 写 / 读这样一连串连续的操作中，最初的 512 次是由最前面的器件进行，然后下一个 512 次访问由相邻的器件进行，这样访问目的地逐渐改变，一旦结束最终器件的第 512 次访问，则下一步的操作又将以最前面的器仵作为访问对象开始进行。因为 D 及 Q 等所有都是公用的，所以从外部可以将深度为 n 倍的 FIFO 存储器作为一个器件进行处理。

此时的操作如图 3、图 4 所示。首先，假设读操作与写操作的目的地都位于 FIFO＃1，每当向 FIFO 进行写人操作，FIFO＃1 的写指针前进，与向最终位置（与 FIFO—FULL 无关，只是到 FIFO 缓冲器的末尾位置之前写指针的前进状态）的写入操作同步，/XO 输出有效，这样的过程表示于图的左侧。在此，FIFO#1 已经有效的/XO 信号传递给 FIFO＃2 的/XI，因为 FIFO＃2 也接受了/W 信号，所以可以知道写指针环绕到了自己（读指针没有环绕）。因此，FIFO#2 接受下一个 FIFO 的写操作（图的右侧）。

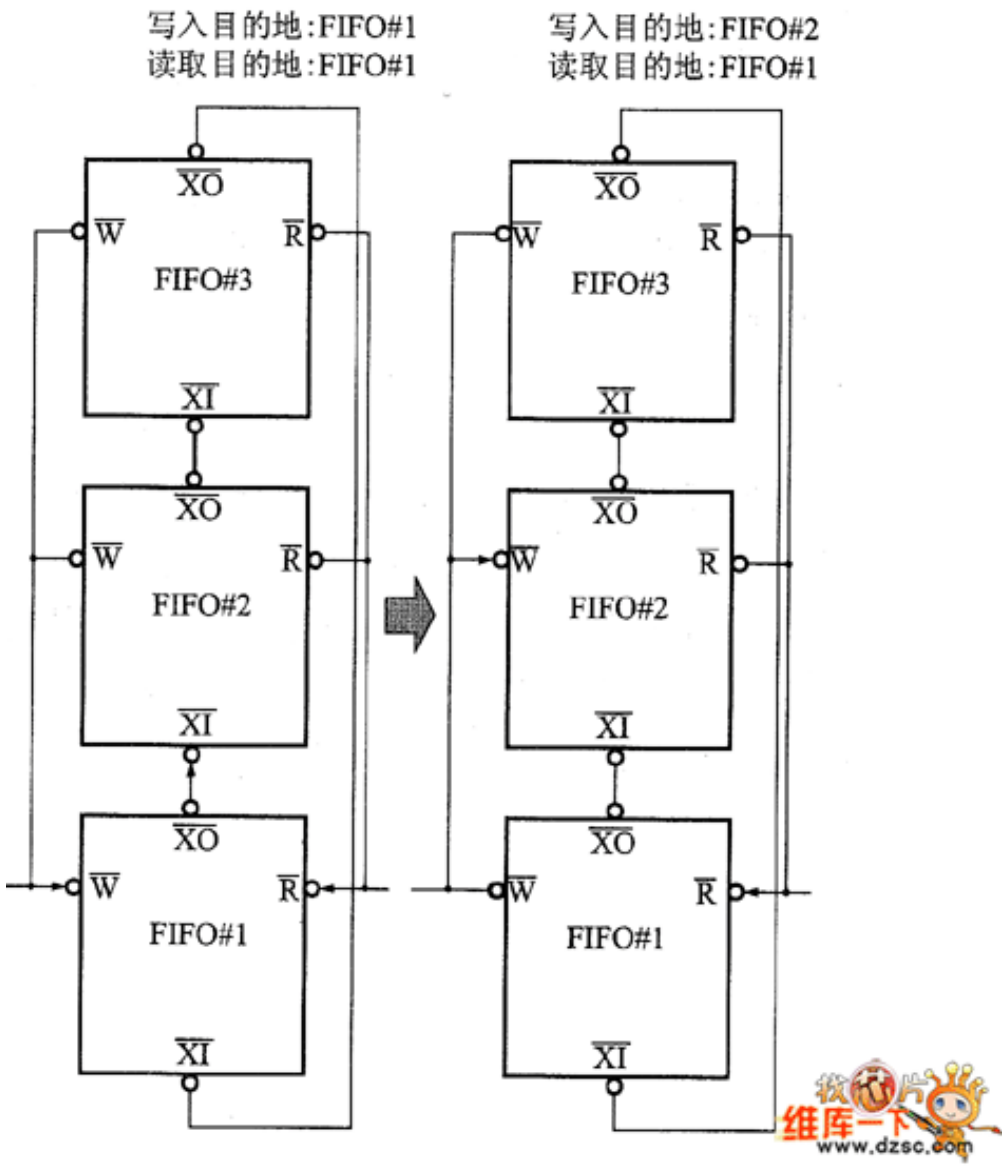


图 3 FIEO 存储器的深度扩展操作（写操作）

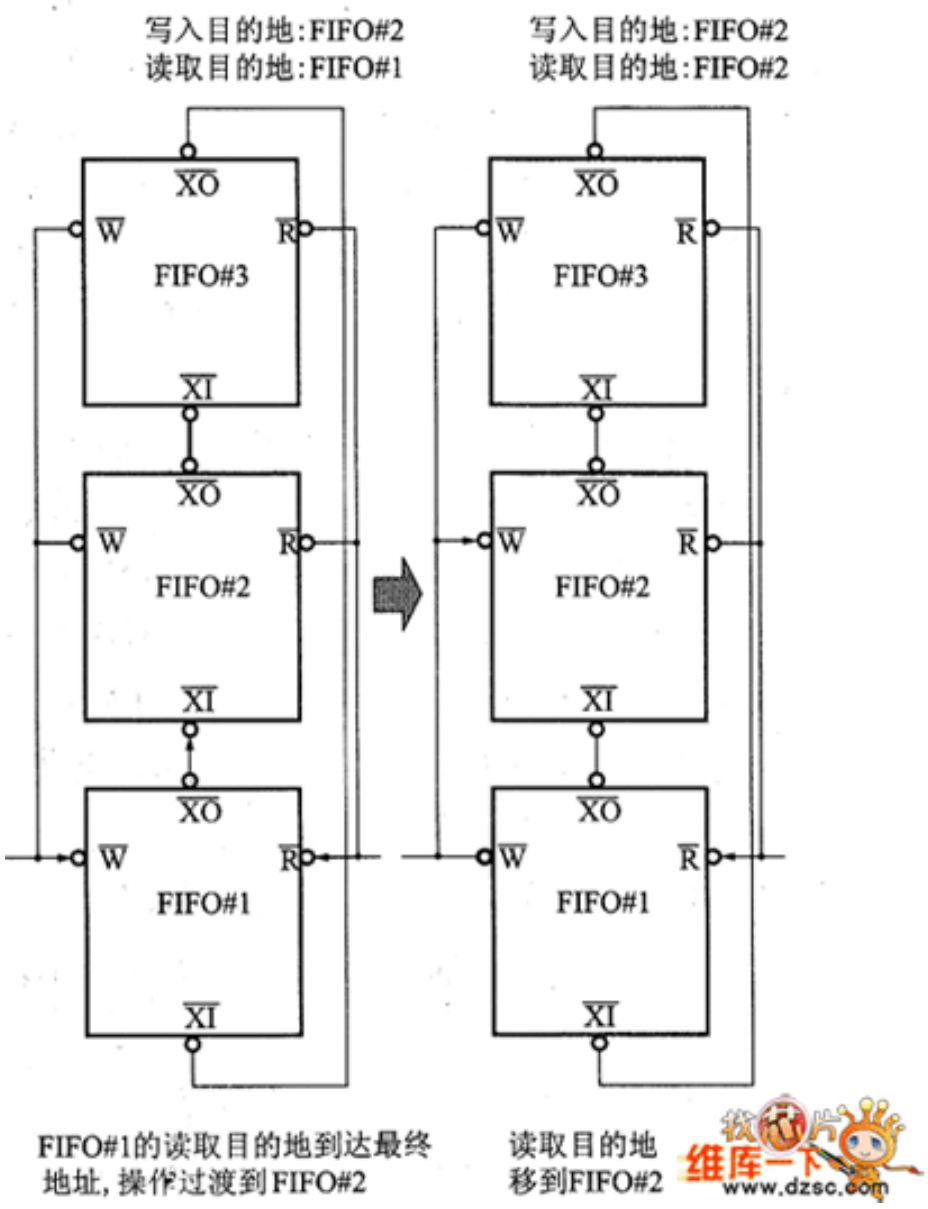


图 4 FIFO 存储器的深度扩展操作（读操作）

进行读操作时也是相同的。读指针如果到达前进中的 FIFO＃ 1 的最终位置（不一定 FIFO 为空，在 FIFO＃2 和 FIFO＃3 都为已满状态下，有时也从 FIFO＃1 的起始位置开始加人数据），则与写操作时相同，/XO 有效，FIFO#2 可知读指针已被传递，这个过程表现在图 4 的左侧。从下一个读操作开始，如图右侧所示，FIFO＃2 将作为存取对象。

第六章

DRAM 的结构与使用方法

6.1　DRAM 的单元结构

DRAM 单元的基本结构如图所示，负责数据存储的是图中的电容器，根据是否存储电荷来判断数据的“0”、“1”。图中电容器的一端接地，因为是交流接地，所以不是形成 GND 电平的意思。电容器的另一端与用于存取开关 FET 的漏极相连接。

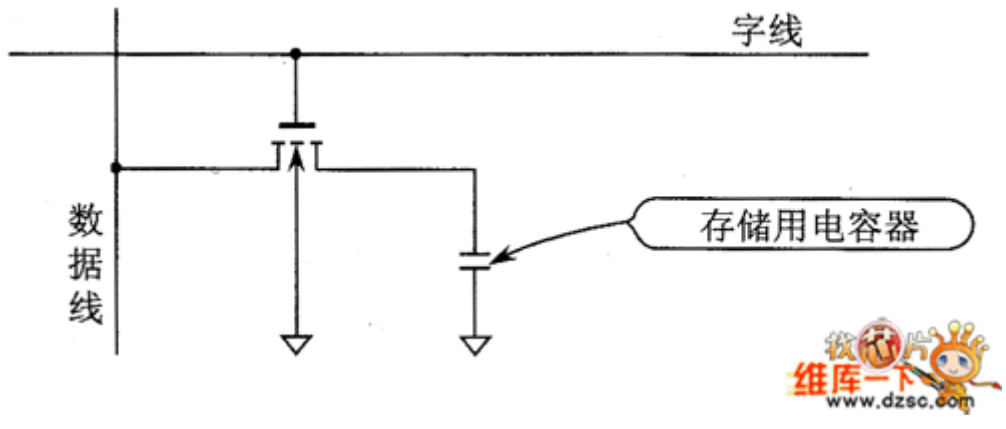


图　DRAM 的单元

读操作时，通过连接于 FET 栅极的字线，将 FET 处于 ON 状态，电容器与数据相连接。由于在电容器与数据线之间发生电荷的移动，数据线的电压发生变化，因此，只要检测出该电压的变化，就可以判定数据（“1”还是“0”）。因为电容器中存储的电荷发生了移动，所以如果进行读操作，其存储的内容就将消失。存取操作的详细过程将在后面说明，像这样因为存取而失去存储状态，亦即破坏性读出也是 DRAM 的特征之一。虽然在数据读操作之后必须恢复同一数据，但该操作是 DRAM 内部自动进行的，通常不必特别在意。

●6.1.1　DRAM 单元结构的概况

图对 DRAM 单元结构进行了较为详细的描述，这是平面式的最基本的结构，在 1MB 的 DRAM 占据主导地位之前一般都是这样结构的单元。与刚才的图相比较更容易理解，图的左侧为 FET 部分，右侧为电容器部分。氧化膜为电介质，多晶硅为电极，从而形成电容器。

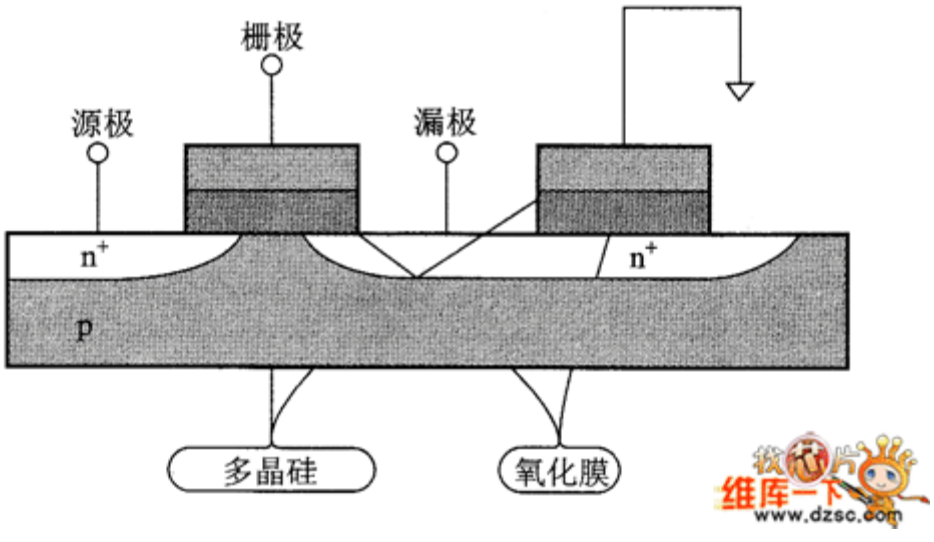


图　DRAM 的单元结构

●6.1.2　刷新

当 DRAM 的电容器存储了电荷时，对于 FET 来说，形成反偏置状态，必然会发生漏电流，图 1 中图示了这一点。因为在如图所示的方向上存在电流，因此 DRAM 单元的电容器将必然进行放电。所以，需要定期将单元的状态恢复为初始状态，这称为刷新操作。

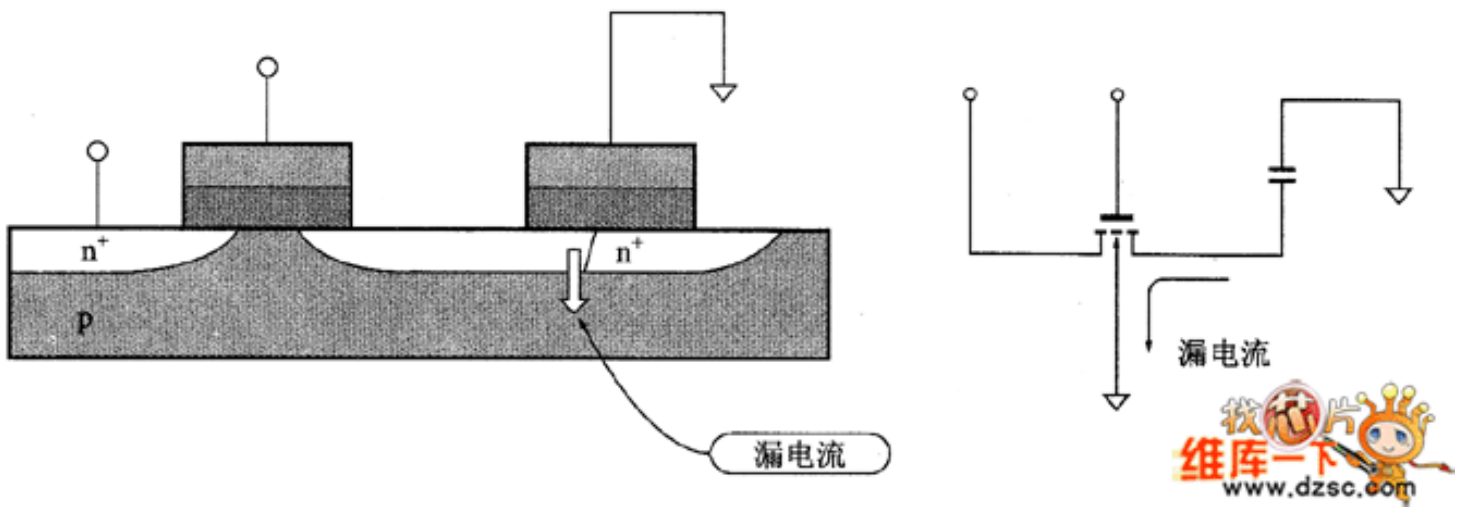


图 1　DRAM 单元的漏电流

通过时间过程与单元电压的关系表示的刷新操作思路如图 2 所示。如果放置电容器则电容器极间电压将如图中虚线那样按指数函数下降，一旦超过阈值，则存储状态将发生反相。因此，在超越阈值之前，必须定期将存储状态恢复为原始的水平，即进行刷新。执行刷新操作的方式存在若干种，这些将在后面进行叙述。

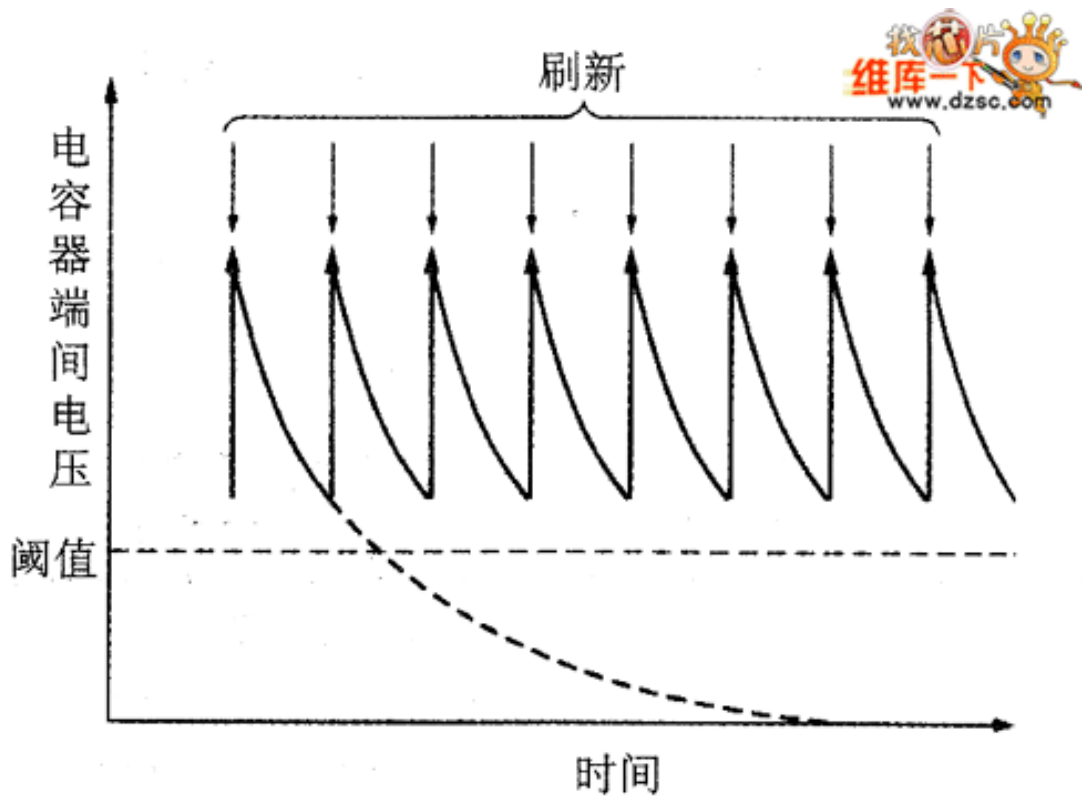


图 2　刷新操作

●6.1.3　软错误

如果为了提高 DRAM 的存储容量而提高集成度，那么当然要减少连接到 FET 的电容器的容量，所以用于存储的电荷量也将随之减少。随着容量的减少，数据的正确读取将变得困难，除此之外，还会存在由于 α 射线而破坏存储的问题。在射线中，α 射线虽然是氦的原子核，但如果它飞人 DRAM 的电容器部分，则电荷将消减，存储将丢失，这称为软错误。为了防止即使是一页纸的 α 射线，需要注意封装所利用的材料中包含的放射线同位素中所放射出的 α 射线，而不必在意在封装状态下来自外界的 α 射线。

一般来说，因为完全除去同位素是困难的，所以，为了降低软错误的概率，无论如何需要在 DRAM 单元上确保某种程度的容量。但是，即使那样也不能使软错误发生的概率为 0。为此，在工作站级别以上的计算机中，使用大量的存储器元件，而且在可靠性要求较高的系统中，对 DRAM 进行 FCC 校验，使之具有即使发生软错误也会自动更正的机制。因为个人计算机等设备中不需要如此高的可靠性，所以大多都没有进行 DRAM 的出错校验。

●6.1.4　电容器的设计

因为电容器的容量不能无限小，所以既要进行小型化处理又要保持其容量是 DRAM 高集成化的重点。基本电介质的介电常数为 ε，电极面积为 S，电极间距离为 d，假设电容器的容量为 C，则：

$$C = \varepsilon \times S \div d$$

成立，因此，为了增大电容器的容量，需要增大 ε 值（使用介电常数较大的绝缘体）、减小d值（使电极间的电介质层尽可能薄）或者增大S值（扩大电极面积）。

介电常数是根据材料确定的，当初曾使用过SiO2 氧化膜，从 1M位的DRAM时代开始，使用称为NO（Si3N4-SiO2）的氮化膜。之后曾经专门在扩大电极面积S的方向上发展过，因为逐渐达到其界限，继而又向使用高介电常数的材料上发展，利用了Ta_2O_5（介电常数约为 50）、BST（介电常数为 250）等。

在增大面积S上，曾经从 4M位的DRAM开始积极发展过，因为在平面上增大面积存在一定的限制，因而采用了立体结构，其中有一些结构相当复杂，图表示了若干结构复杂的例子。

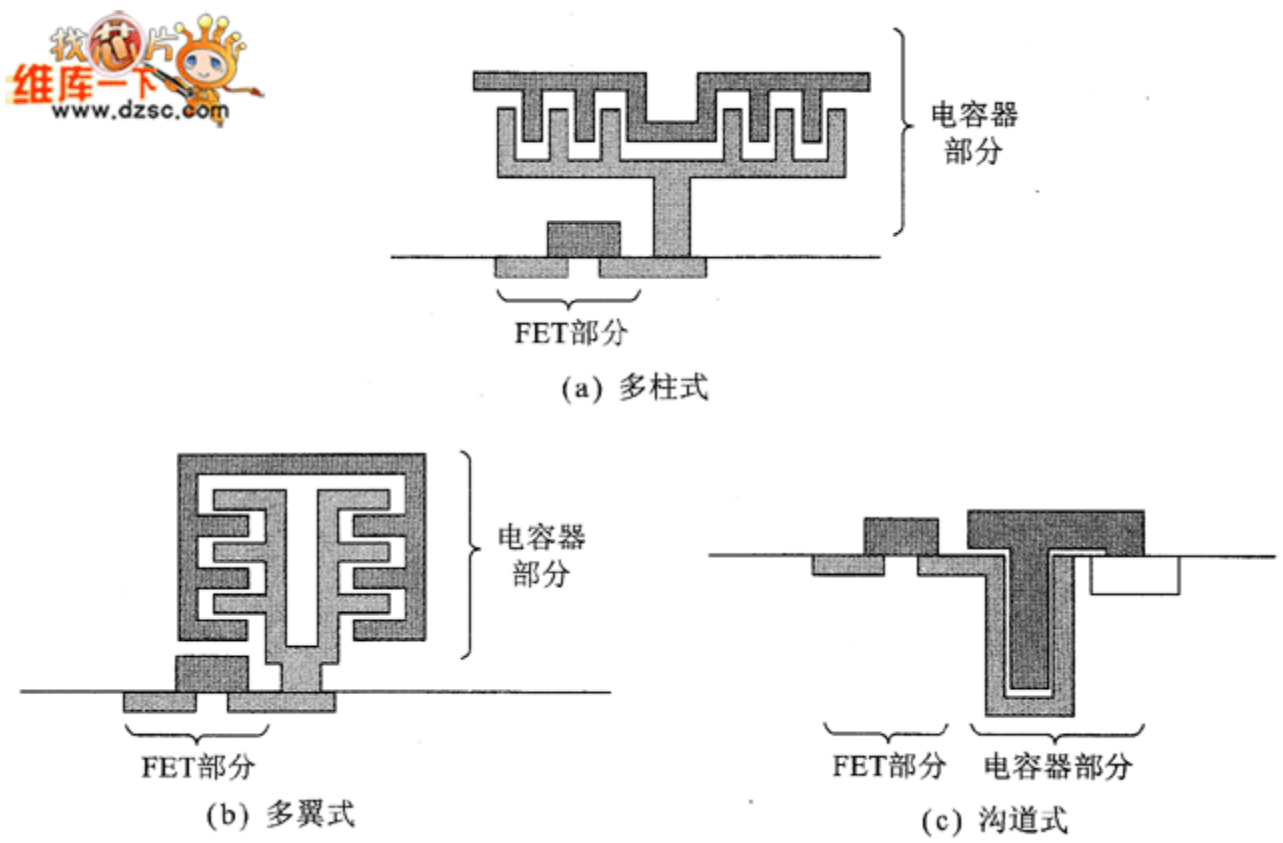


图　DRAM 电容器的设计

从大方面上分类，电容器的结构分为在底板上叠加电容器结构的堆叠式结构和在底板上穿孔、利用其孔洞的沟道式结构这两种。前者还包括如图（a）所示的形成凹凸的圆柱式以及如图（b）所示的水平方向上形成凹凸的翼状式。最初只有一个凹凸面，然后为了增大面积又发展成两端具有多个凹凸的多柱式及多翼式结构，图中也表示了这些类型。正如先前表示的那样，通过利用高介电常数的材料，即使减少面积也可以得到相同的容量，因此，根据这样的方法，减少堆叠式结构的凹凸数，姑且认为

可形成扁平器皿状的电容器。

穿孔的沟道式结构如图（c）所示，在洞的纵深方向没有什么变化，但在纵横比（深度 / 宽度）上存在较大的变化，目前纵横比为 30 左右。

在沟道式结构上，电介质材料的利用还没有什么进展，但这应该是将来发展的方向。

6.2　DRAM 内部电路

DRAM 单元部分的布线如图所示，具有用于单元选择的字线，并且各个单元与数据线相连。数据线通过列选择开关或者通过预充电开关分别与公用数据线或者预充电电源相连接。预充电电源的电压大多采用器件电源电压一半左右的电压值。

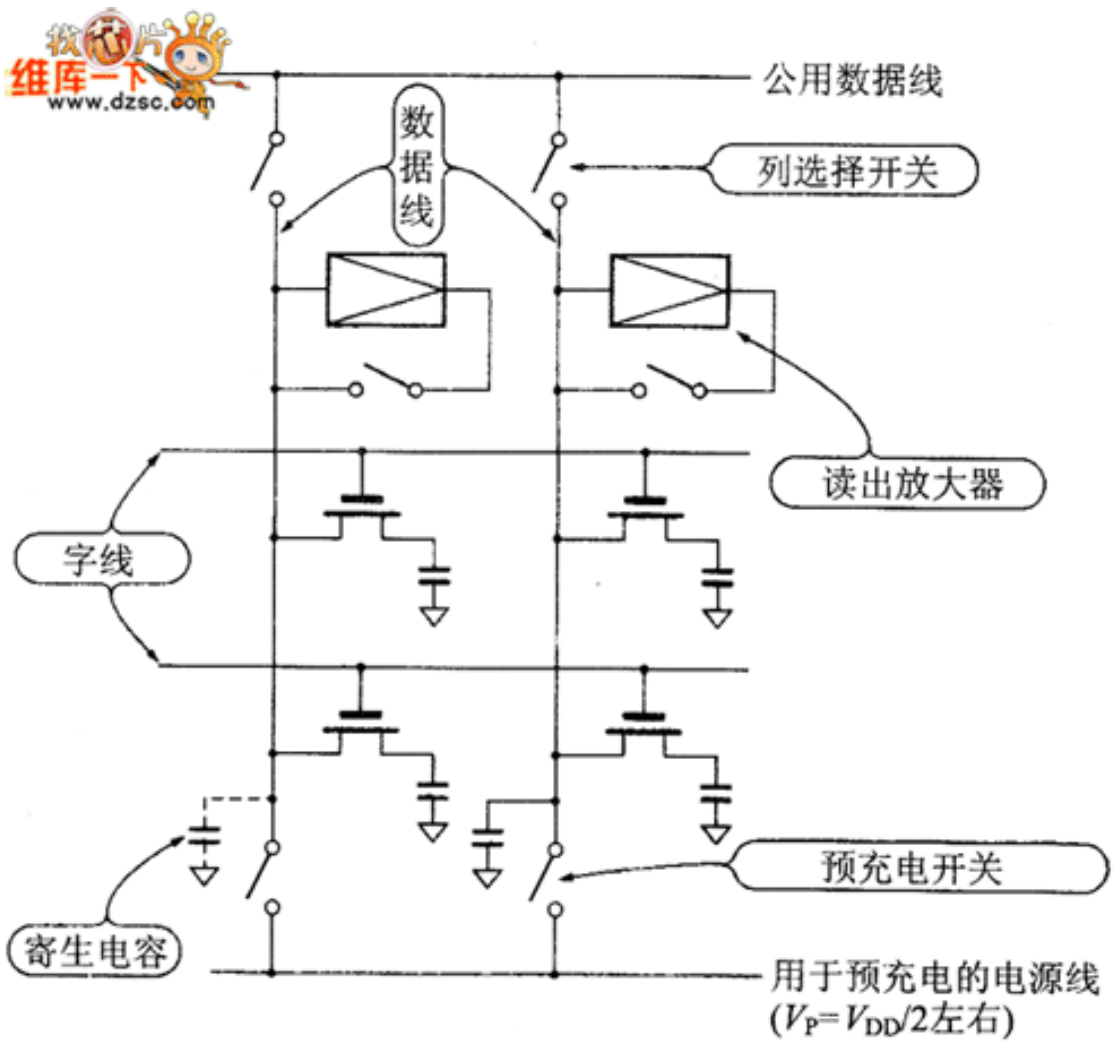


图　DRAM 的基本结构

数据线上具有读出放大器，在这里对数据线上的状态“1” / “0”进行判定以及放大数据线上的电压电平。

图中虚线表示的电容器符号是数据线的寄生电容，正如后面将要叙述的那样，在读出 DRAM 上的数

据时，这个寄生电容具有较大的作用。

下面我们来探讨 DRAM 单元的读操作的思路。DRAM 单元的写人操作是通过确定数据线的状态，将 FET 设置为 ON，然后通过电容器的充放电来完成的。读操作上稍有些麻烦，因为用于存储的电容器的容量非常之小，所以不可能一下子驱动公用数据线。

▲数据线的预充电

图表示可以说是读取 DRAM 之前的准备状态，数据线与预充电电源相接，将数据线的电压设置为预充电电压，数据线借助寄生电容，即使将预充电开关设置为 OFF，数据线的电压也会保持预充电电压（当然，由于存在漏电流因而会逐渐下降）。这样的操作称为预充电。

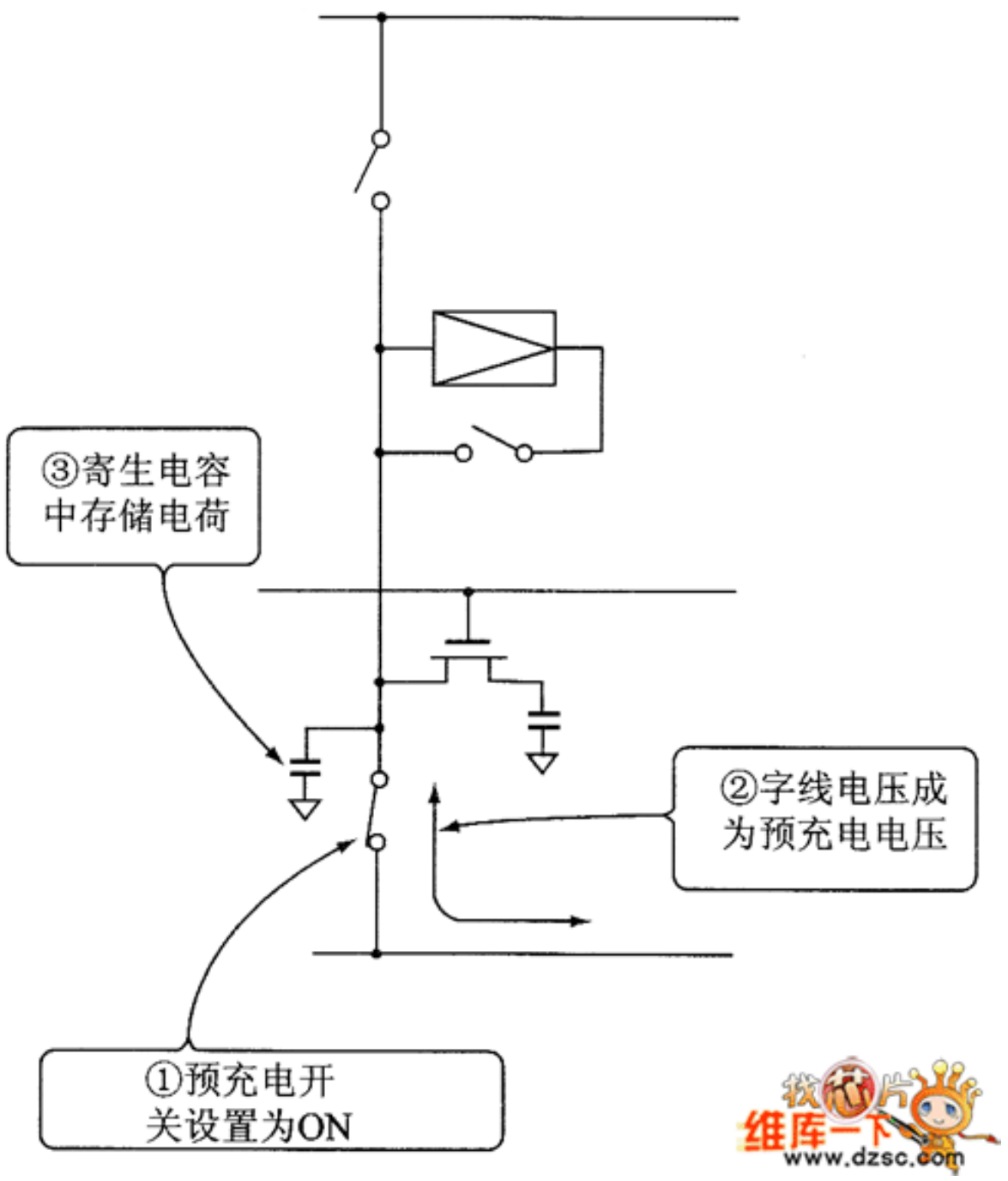


图　DRAM 的读取（预充电）

▲数据的提取与放大

只要完成了预充电，预充电开关就处于 OFF 状态。之后，选择数据线，一旦 FET 为 ON，特定单元的电容器与寄生电容则形成并联的格局，这样，根据数据的“1”/“0”，预充电电压可进行高低调整。这样的变化并不是很大，所以利用读出放大器进行放大。

读出放大器与其说它是类似音频放大器之类的器件，不如说它类似数字 IC 的缓冲器更容易理解。以预充电电压为基准，根据电压的高低确定输出是高电平还是低电平。图表示这一过程。

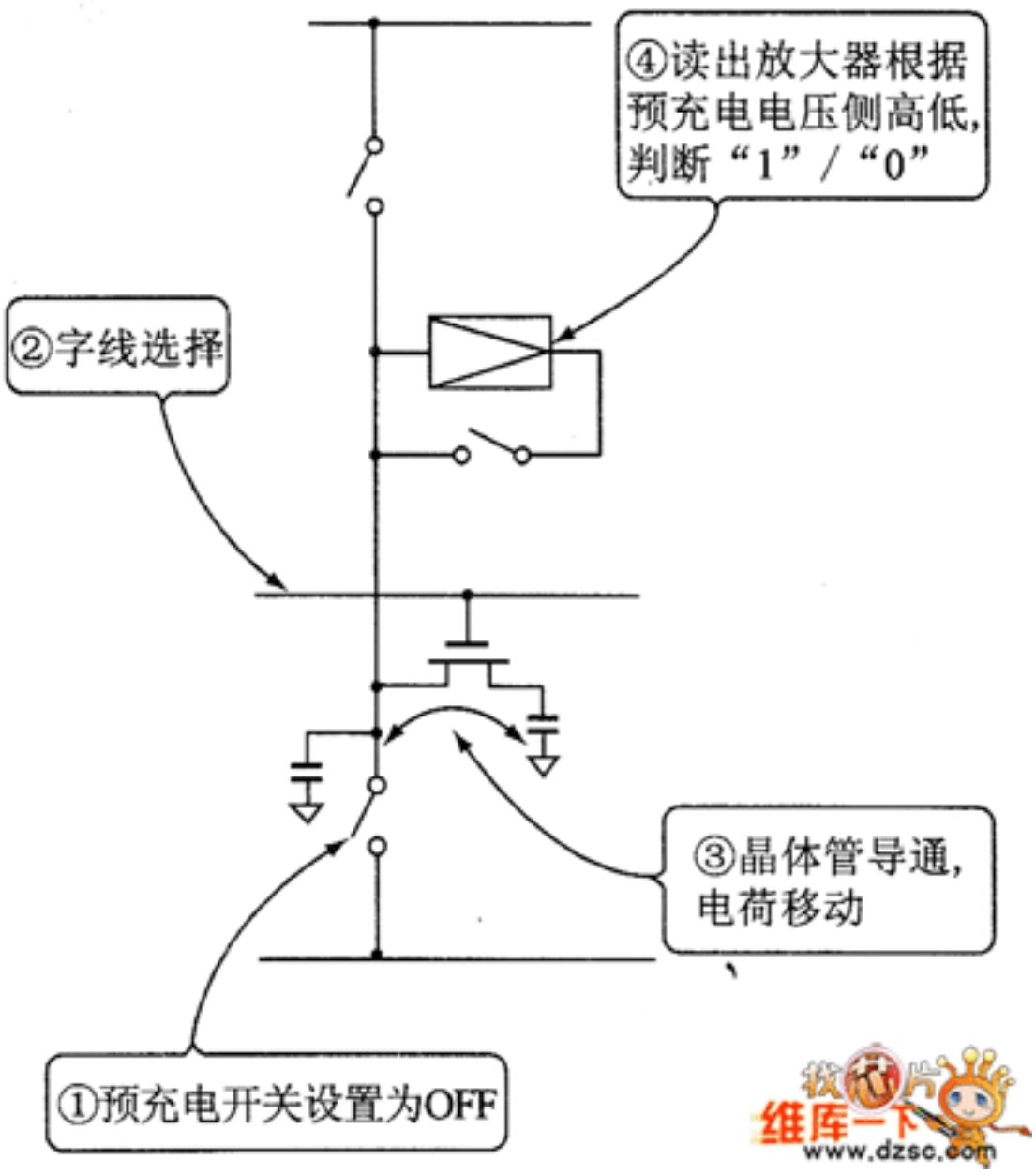


图　DRAM 的读取（数据的提取与放大）

▲读出放大器的连接与读出

在利用读出放大器结束放大的过程中，读出放大器的输出与数据线相连接，如图所示。读出放大器的输出被连接，所以数据线上的电压可在读出放大器的输出电压之内变化。读出放大器的输人也通过读出放大器自身的输出而被驱动，也就是形成自维持电路的形式。

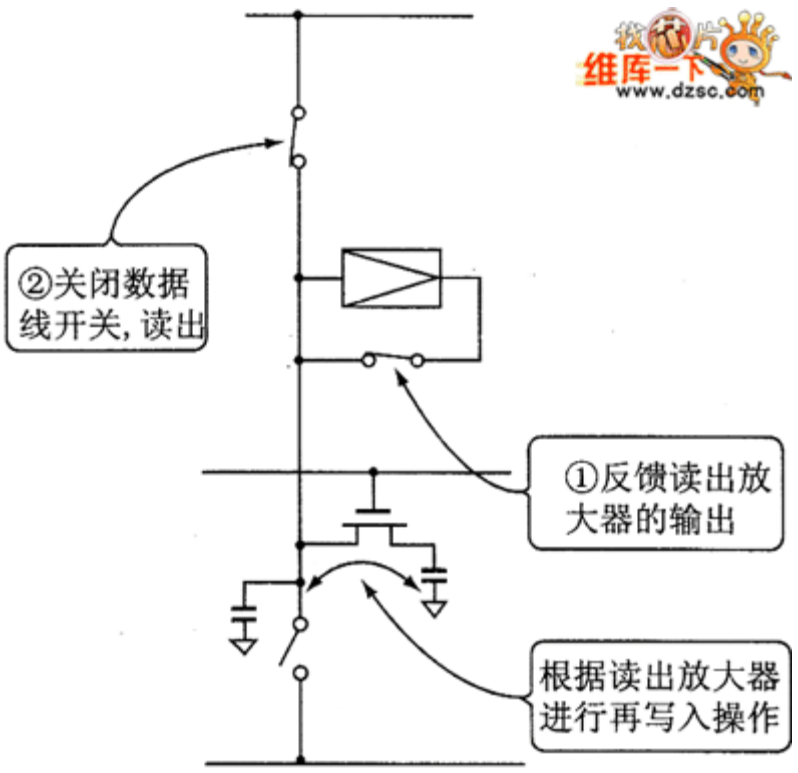


图　DRAM 的读取（反馈放大器的连接与读取）

由于驱动数据线的不是 DRAM 单元的电容器而是读出放大器，所以可以充分地驱动公用数据线。在此，公用数据线的开关也设置为 ON 状态，这样公用数据线被驱动，从而可以向与外部的接口处传送数据。

此时，由于字线被选择而 FET 处于 ON 状态，所以 DRAM 单元处于数据线与单元电容器相连接的状态，因此，电容器的状态恢复为初始状态。刷新操作除了数据线与公用数据线不相连这一点不同外，其他基本操作都是相同的。

6.3　DRAM 的外部接口

●6.3.1　DRAM 的基本信号

DRAM 从最初时期的产品至今所利用的信号类型如图所示。DRAM 的主导产品已经逐步移向与时钟同步的同步 DRAM 及 Direct Rambus DRAM（直接总线式 DRAM），所以我们所说明的这种类型的 DRAM 将逐渐变得稀少，我们只将基本的信号举例说明一下。

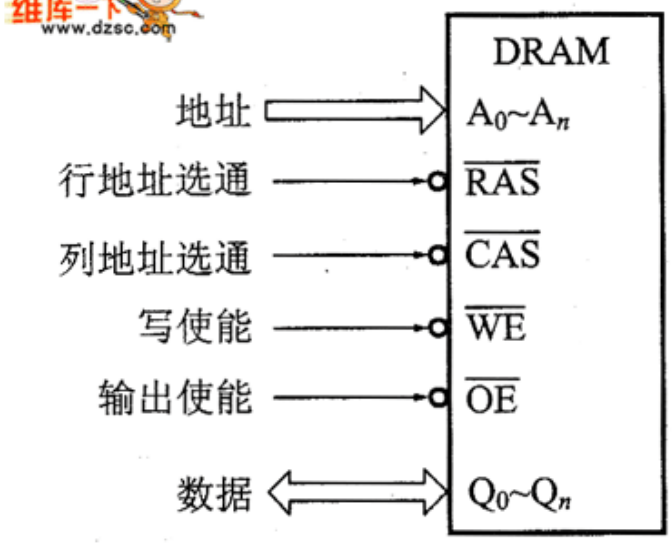


图　DRAM 的基本类型

▲　地址（A0～An）

这是赋予 DRAM 的地址，由于 DRAM 是利用后面将要说明的/RAS 及/CAS 信号分 2 次赋予地址的，所以所拥有的地址引脚数为寻址所需要的数目的一半左右。例如，1M×1 位的 DRAM，其地址需要20.位，但它分 2 次、每次赋予 10 位的地址。由于容量较大的 DRAM 并不一定每次都给予一半的地址，因此也存在实际引脚数目超过寻址所需要的半数的器件。

▲　/RAS（行地址选通）

当指定 DRAM 单元时，根据行地址与列地址选择所访问的单元。在/RAS 的下降（由高电平向低电平的变化时刻）过程中，地址引脚作为行地址锁存于 DRAM 内部。

▲　/CAS（列地址选选通）

行地址之后，如果/CAS 有效，则 DRAM 将地址引脚的状态作为列地址提取到内部。根据列地址，其中一个列选择开关被选择而处于 ON 状态，公用数据线上出现数据。

像这样利用行地址与列地址指定所访问单元的方法是 DRAM 基本的访问方法。

▲　/WE（写使能）

这是指定是读数据还是写数据的信号，进行写操作时，/WE 有效。

▲　/OE（输出使能）

启动 DRAM 的数据输出缓存，当/WE 无效时，DRAM 内部在读模式下运行，但如果/OE 无效，则数据引脚（DQ0～DQn）不能被驱动，保持高阻抗的状态。

▲　DQ0～DQn（数据）

这是数据输入输出引脚，可双向使用。

● 6.3.2　DRAM 的读/写操作

DRAM 基本的存取操作如图所示，结合/RAS 及/CAS 的有效，分割为行地址和列地址赋予地址。进行读操作时，如果在这里/OE 有效，则 DQn 引脚被驱动，读出数据。另一方面，进行写操作时，在/CAS 有效之前/WE 有效，然后 DQn 上设置数据，如果/CAS 有效，则在其下降沿写入数据。

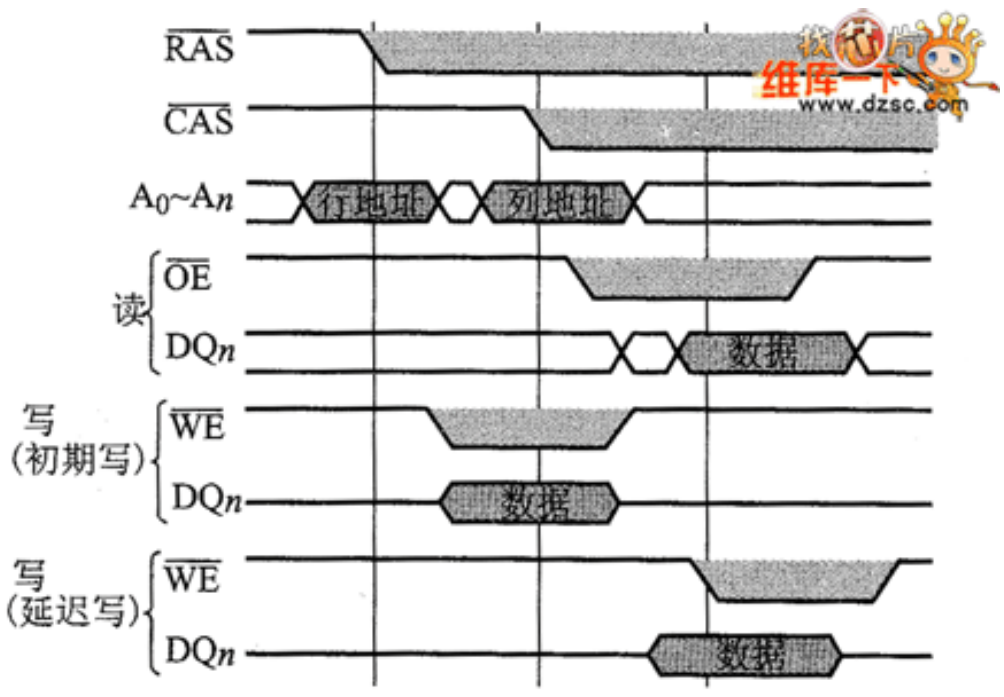


图　DRAM 的存取操作

这是所谓的初期写的一般方法，除此之外，还具有称为延迟写的方法，此方法在/RAS 及/CAS 有效的状态下设置（由于/OE 已经无效，因而 DQn 不能被驱动）数据，在/WE 的下降沿写人数据。这些方法都是在进行读－修改－写操作时方便的方法，所谓的读－修改－写操作也就是从存储器读出数据后、更改部分比特位、然后写回到同一地址的操作。

其过程就是/RAS、/CAS 有效之后/OE 有效，一旦读出数据即将其提取，然后/OE 无效，在 DQn 上设置新的数据，再使/WE 有效。/RAS 及/CAS 保持原状态也可以。与连续生成读周期和写周期相比，数据修改操作是效率更高的好办法。

● 6.3.3　DRAM 的刷新操作

▲惟 RAS 有效刷新

正如在 DRAM 的存取操作中所说明的，如果进行 DRAM 的读操作，则因为读出放大器的输出被返回到电容器，所以可兼容刷新操作。但是，如果只考虑刷新操作，那么就不需要赋予列地址读出数据，因此，不赋予列地址，只赋予行地址的方法就是惟 RAS 有效刷新。

惟 RAS 有效刷新操作如图所示。设定行地址（刷新地址）后/RAS 有效，然后只要设定列地址、/CAS 有效，就是读操作，如果此时/CAS 无效，/RAS 无效，就是刷新操作。

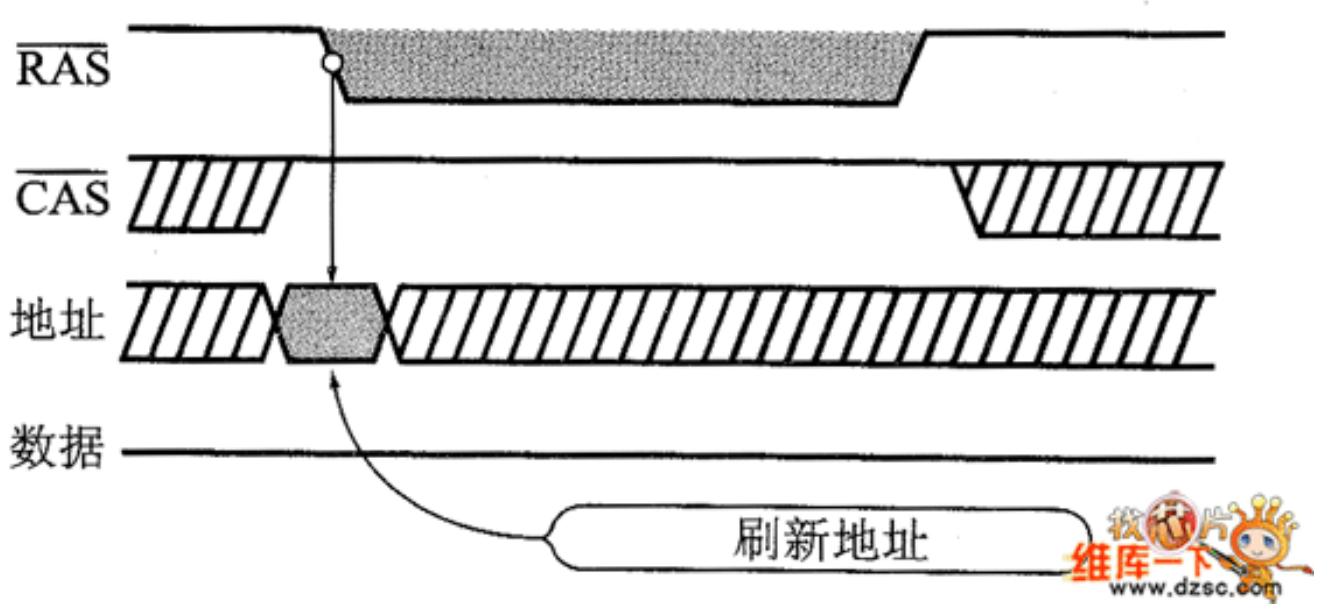


图　惟 RAS 有效刷新

DRAM 内部不需要进行特殊的设计，而且作为 DRAM 控制器端，当设计用于刷新的时序生成电路时，当利用惟 RAS 有效刷新时，则只要从普通的访问电路屏蔽 CAS 即可，所以相对来说，这是经常利用的方法。惟 RAS 有效刷新的方法虽然简单，损耗电流与进行读操作时的损耗电流相比也较小，但与其他的刷新方式相比，其损耗电流却较大，这是其缺点。

▲CAS 先于 RAS 有效刷新

在惟 RAS 有效刷新的操作中，DRAM 控制器必须知道个别的 DRAM 具有多少刷新地址，这是非常不方便的，因而又设计了 CAS 先于 RAS 有效刷新的方法。该方法在 DRAM 内部内置刷新地址的发生电路，由 DRAM 控制器来指示开始刷新操作。

在普通的存取操作中，是按照/RAS 先有效、/CAS 再有效的顺序进行的。改变这种顺序，通过/CAS 先有效、/RAS 再有效的顺序，指示刷新操作。如图所示，改变/RAS、/CAS 的顺序需要切换电路，但刷新地址是在 DRAM 内部自动生成的，外部不需要准各用于刷新地址的计数器，也不需要地址的多路转换，因而存在优势。而且，与惟 RAS 有效刷新操作相比，损耗电流一般较小，不需要在存储器控制器中生成刷新地址，只要管理周期即可，因此，这种刷新方法在个人计算机等设各中应用最广泛。

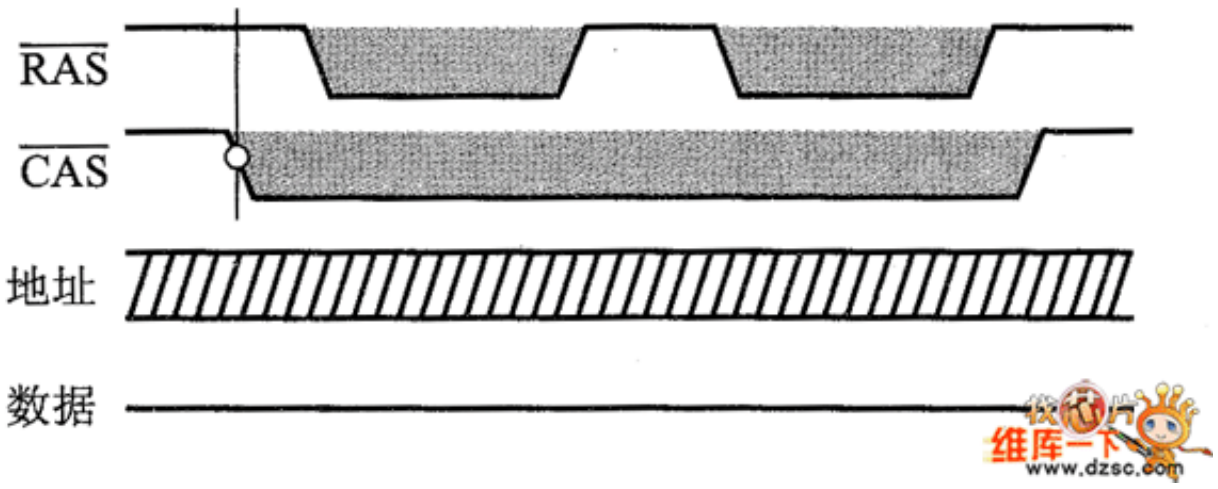


图　CAS 先于 RAS 有效刷新

▲隐藏刷新

一般地，DRAM 控制器内部都设计成在一定周期内要请求 DRAM 刷新操作，协调该请求与来自主机（一般为 CPU）的访问，然后进行 DRAM 刷新操作或者存取操作。简单的如图 1 所示。

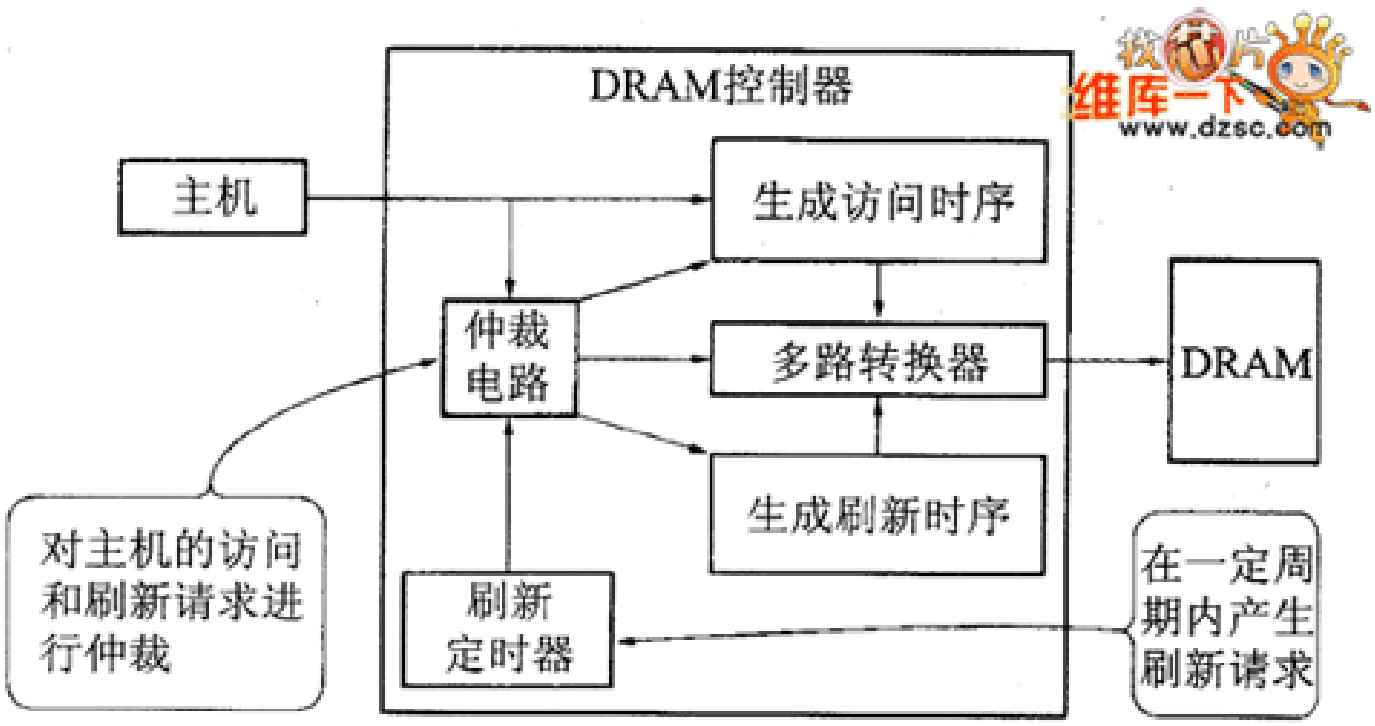


图 1　DRAM 控制器的内部框图示例

因为对 DRAM 的访问是互斥的，所以如果在刷新过程中存在来自主机的访问，那么保持该访问请求直到刷新操作结束，这样，越增加存储器访问频率，刷新操作与来自主机的访问之间冲突发生的概率就越高，导致的结果就是性能下降。隐藏刷新操作就是为抑制这种性能下降而设计的一种刷新方法。

隐藏刷新操作如图 2 所示。图的最初过程是/RAS 最先有效的、普通的读访问操作，在此通过/RAS 先无效而后再有效，形成与 CAS 先于 RAS 有效刷新相同的波形，完成刷新操作。

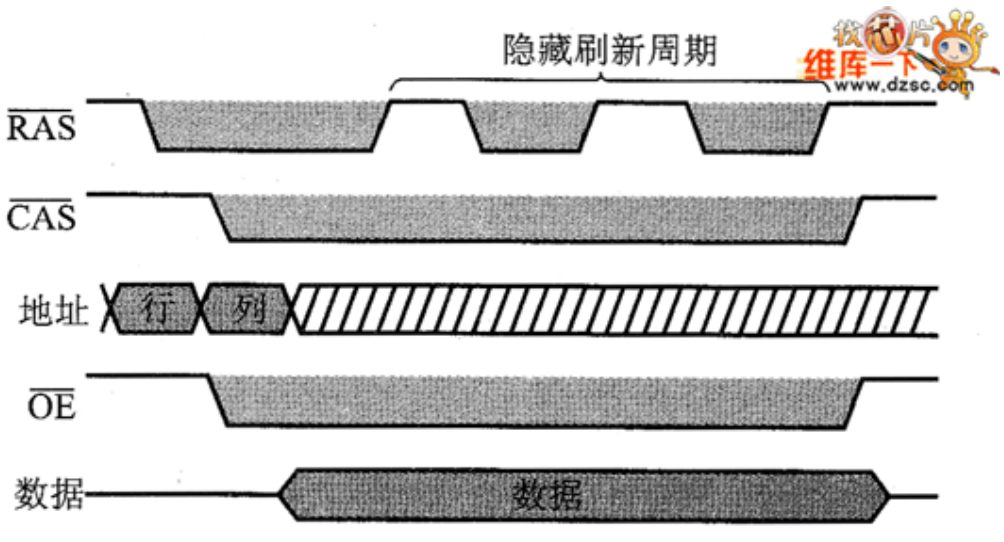


图 2　隐藏刷新操作

此时，保持数据输出的状态，利用该状态，可以在来自主机的一个访问之间嵌人刷新周期。由于将刷新周期隐藏于普通的访问之中，因而称这种方式为隐藏刷新（Hidden Reflesh）。

▲自刷新

这是为适应低功耗等需求而设计的模式。由于 DRAM 的刷新电路一般都设计在外部，因而即使在待机状态下，为了进行刷新操作也需要运行 DRAM 控制器电路。

对此，在 DRAM 内部嵌人刷新计时器以及刷新地址生成电路，使 DRAM 自身可以自动地进行刷新操作，这就是自刷新操作。

自刷新的操作如图所示。最初与 CAS 先于 RAS 有效刷新操作相同，但如果将/RAS 及/CAS 保持有效状态持续 100μs 后，DRAM 内部的自刷新电路开始运行，然后自动进行刷新操作。如果/RAS 及/CAS 无效而开始存取操作，则自刷新操作停止，恢复为一般的操作模式。

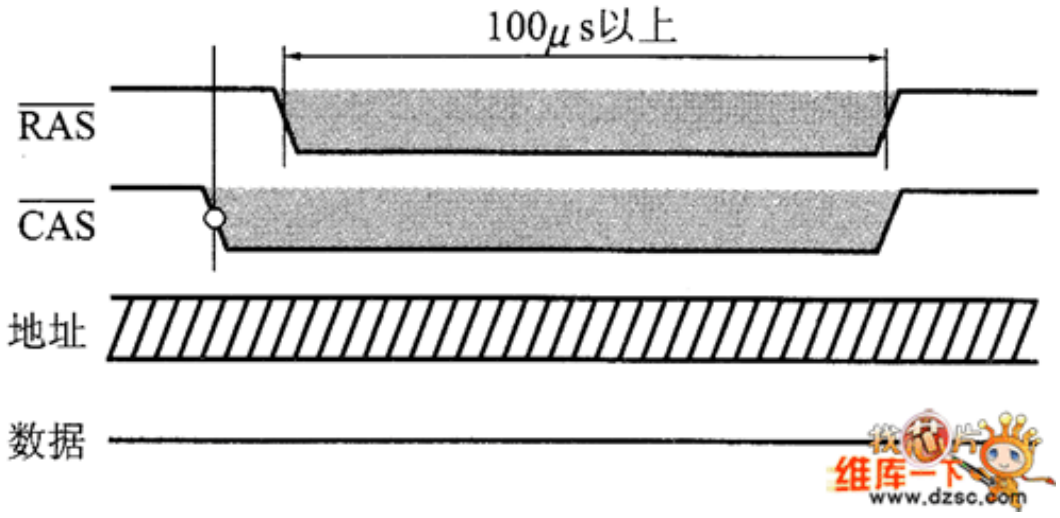


图　自刷新操作

由于在普通的操作中，一般没有停止超过 100μs 的存取操作的情况，所以自刷新操作毕竟是在待机状态下所利用的模式，能够较低地控制损耗电流，因而对于电池各份等非常有利。

●6.3.4　DRAM 的快速访问模式

观察对 DRAM 单元的访问方式即可明白，在多路复用地址以及读操作之前必须进行预充电，以及利用读出放大器进行放大等，所以不太擅长随机访问。但是，在实际中的存储器访问中，持续访问连续的区域是很常见的，而且在安装了高速缓冲存储器的情况下，由于高速缓冲存储器与主存储器（一般以 DRAM 构成）之间的传输是以块为单位进行的传输。所以，完全的随机访问是罕见的，大多是连续的区域或某狭窄的区域被集中访问。因此，着重设计了在 DRAM 端可连续访问已确定区域的机制。

以前访问模式具有页模式、静态列模式和半字节（nibble）模式这三种，之后页模式被随后出现的快速翻页模式（Fast PageMode）所取代，进而又被 EDO 模式（也称为超页模式，HyperPageMode）所取代一直到现在。期间，具有静态列模式及半字节模式的 DRAM 逐渐从市场上消失。

这几种访问模式基本的思路类似。如果利用读操作进行说明，那么正如在 DRAM 的单元结构中所说明的那样，在 DRAM 读访问中，在选择了字线的时刻，在各个数据线上确定数据，该数据由列地址进行选择。因为在所有的数据线上都将一定出现数据，所以，与每次都要重新赋予行地址的操作相比，在此只要切换列地址，就可以进行快速的访问。只要能将来自主机地址中的高位分配给行地址，将低位分配给列地址，那么就可以进行连续区域的快速访问。

▲静态列模式

静态列模式操作的概况如图所示。在一般的存取操作中，如果通过/CAS 指定地址，那么就只出现其列地址的数据。但如果保持/CAS 有效而切换地址，则成为切换列地址的模式。在 DRAM 内部，只要切换列选择开关就可以输出数据。

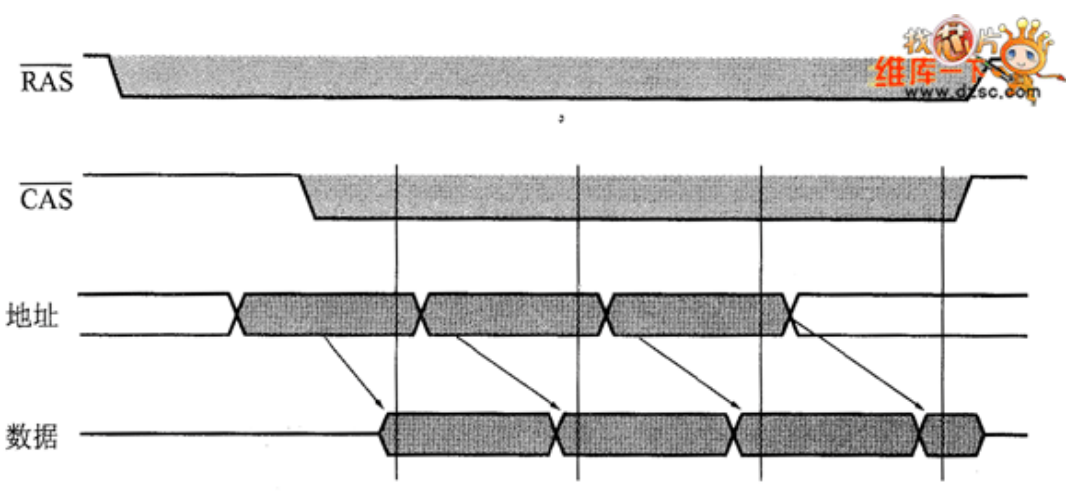


图　静态列摸式

在设计上不需要列地址的选通信号，只要简单地切换地址即可达到目的，这是其简单方便之处，但地址的误差将影响到 Qn，这又是所存在的缺陷。在个人计算机领域，有相当一部分机型采用了该模式。

具有这种模式的 DRAM 现今几乎不存在了。

▲半字节（nibble）模式

半字节模式的 DRAM 如图 1 所示，在 DRAM 的输出缓冲器部位设计了 4 字（word）锁存器。通过这个锁存器，对于起始地址的 4 字数据，可以不赋予列地址而进行连续的输出。只要认为这正好类似于管线突发式 SRAM 的突发传输模式即可，图 2 表示半字节模式 DRAM 的操作。

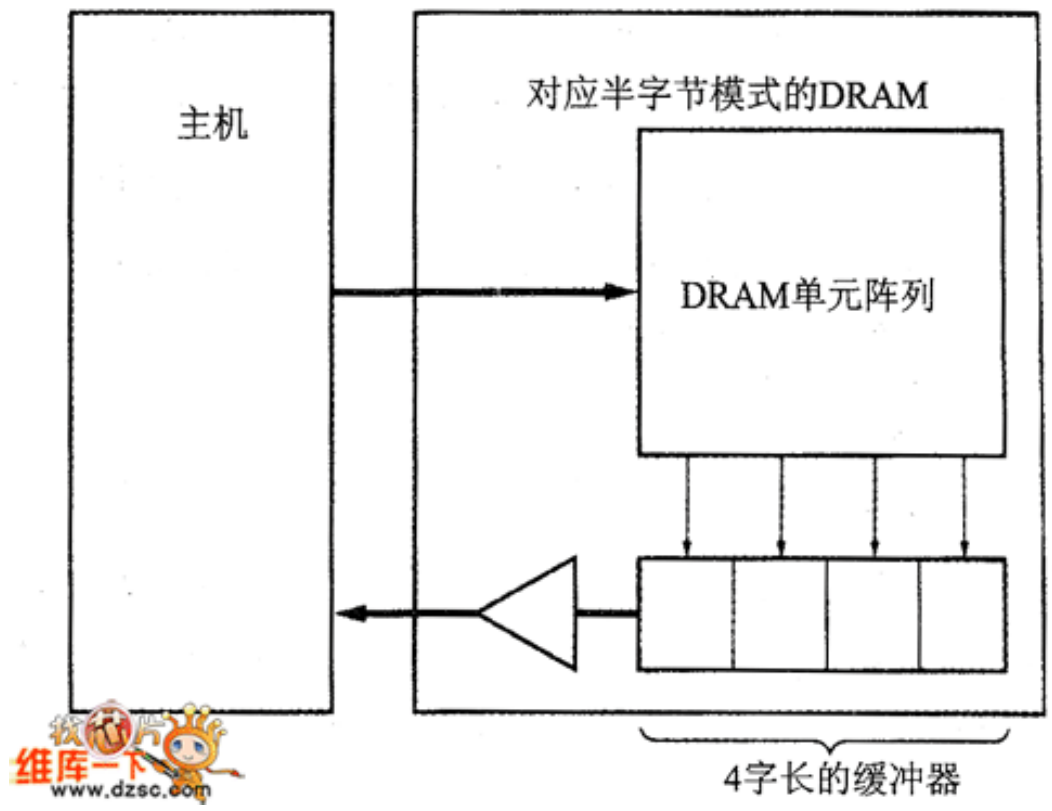


图 1　半字节 DRAM 的思路

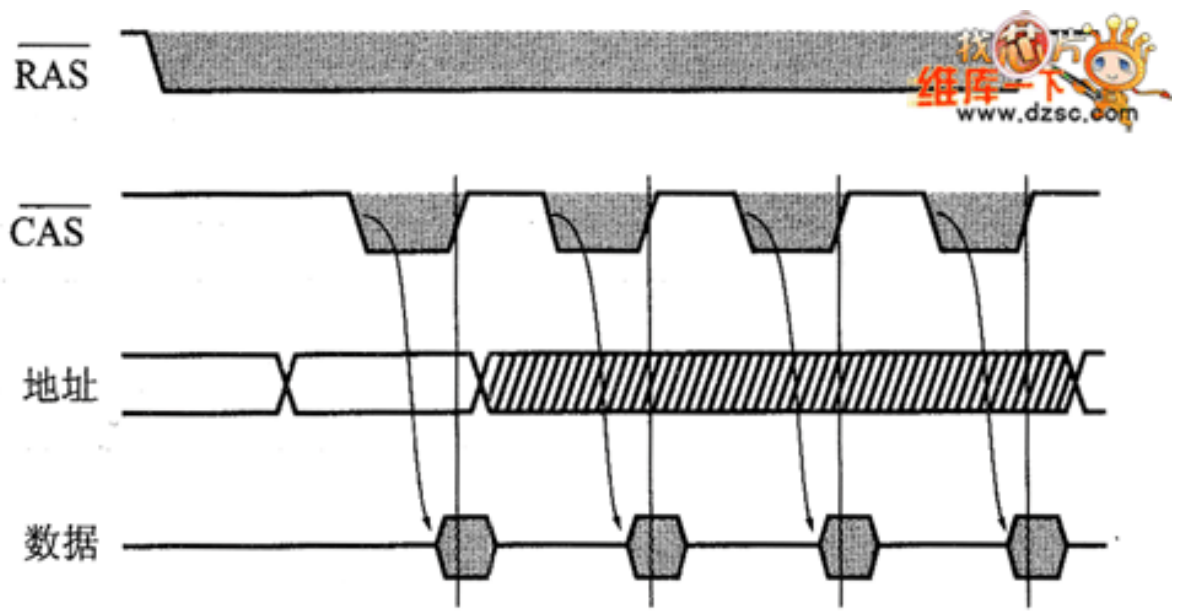


图 2　半字节模式

由于地址的顺序（相当于管道突发式 SRAM 的突发序列）是固定的以及只能处理 4 字数据从而存在较大的局限等原因，使得该模式没有像页模式 / 快速翻页模式那样得到普及。

现在具有该模式的 DRAM 也几乎不存在了。

▲页模式

页模式是在通常利用/RAS 及/CAS 的访问后，通过每次重新赋予/CAS 和列地址，对同一行地址中任意列地址进行访问的方式，图表示页摸式的操作。

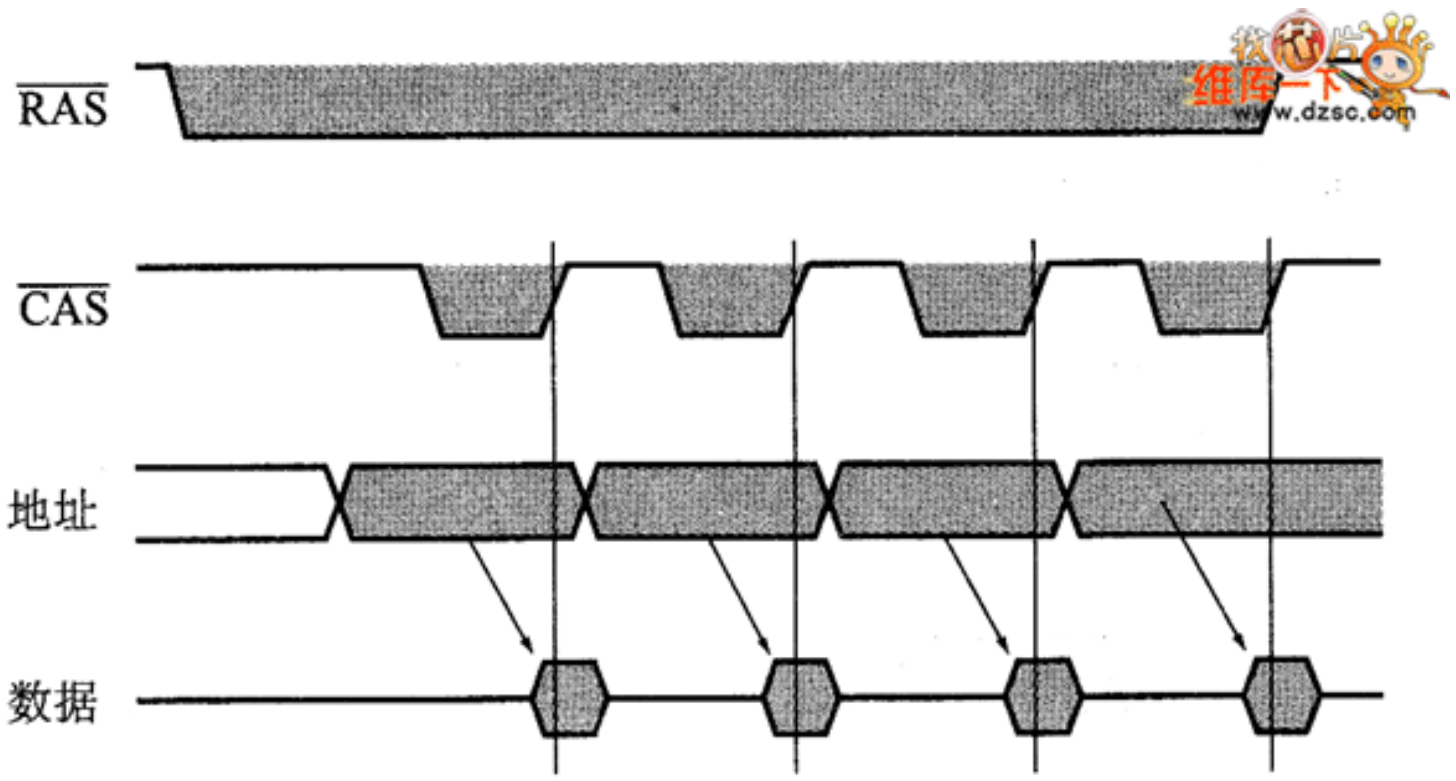


图　页模式

最初第一回的访问与通常的访问没有什么不同，但当访问同一行地址区域时，/CAS 暂时无效之后，将重新设置列地址，然后再使/CAS 有效，这样将输出下一列地址的数据。因为只要是同一页内（行地址为同一区域内）就能进行随机访问，所以，让 CPU 进行管线操作，判断下一个地址是否在同一页内。只要在同一页内，那么不使/RAS 无效，只根据列地址和/CAS 进行访问，可以说这种方法是非常方便的。目前页模式已经让位于号称其升级版的快速翻页模式。

▲快速翻页模式

在页模式中，因为当/CAS 有效时是不能改变地址的，所以 DRAM 控制器需要锁存数据后使/CAS 无效，然后切换地址。将这种方式进行改善，通过在/CAS 下降沿锁存列地址，在/CAS 有效期间进行下一列地址的切换，这就是快速翻页模式。CPU 总线在管道模式下运行时，在最初存取操作的途中将输出下一个地址，此时的地址可以原封不动地传送给 DRAM，可以说这是利用价值很高的一种模式，其操作如图所示。

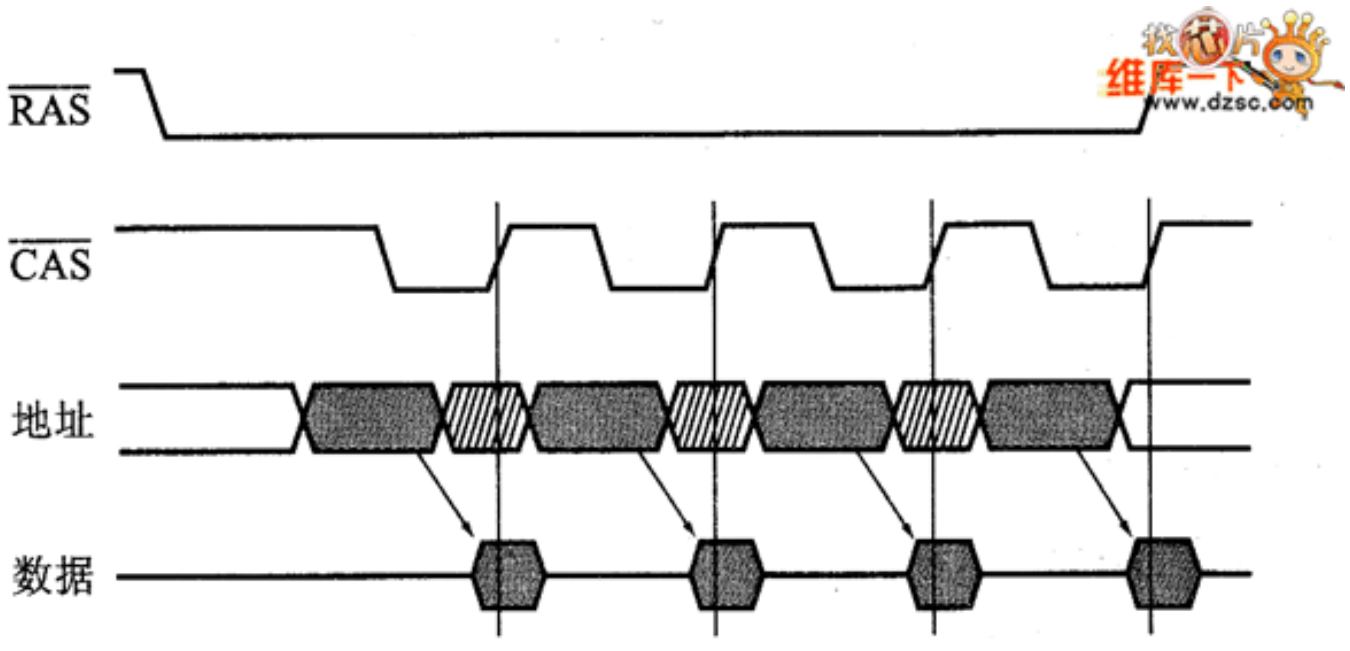


图　快速翻页模式

快速翻页模式的使用一度达到鼎盛，但随着更易于提高性能的 EDO 模式（超级页模式）的出现，而逐渐被淘汰。

▲EDO 模式

在快速翻页模式中，如果/CAS 无效，则将停止 DQn 的驱动，数据将随之消减。取代这种方法，而采用即使/CAS 无效，也能保持数据输出的方法，即采用 EDO（Expansion Data Output，扩展数据输出）模式。因为可以将/CAS 无效的时间隐藏于访问之中，因此可以提高速度几乎接近于极限值。

图表示 EDO 模式的操作。即使/CAS 无效数据也可持续输出，因此，在此期间可以设置下一地址，然后使/CAS 再次有效，这样就可以赋予下一地址。

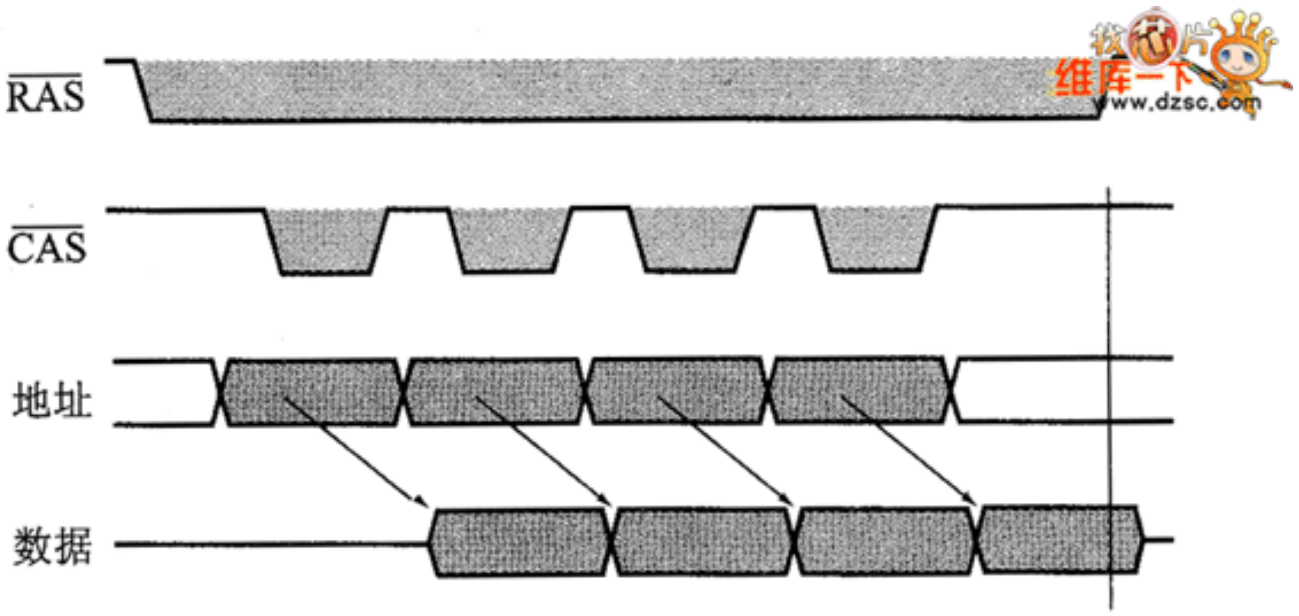


图　EDO 模式

6.4　同步 DRAM

● 6.4.1　同步 DRAM 的信号

同步 DRAM 的信号类型如图 1 所示，其中存在时钟（CLK）、时钟使能（CKE）以及存储块（Bank）编号指定等若干信号的更改，但可以看出，同步 DRAM 沿用了异步 DRAM 的信号。SDRAM 将内部分割为若干个存储块，这是 SDRAM 的一大特征。

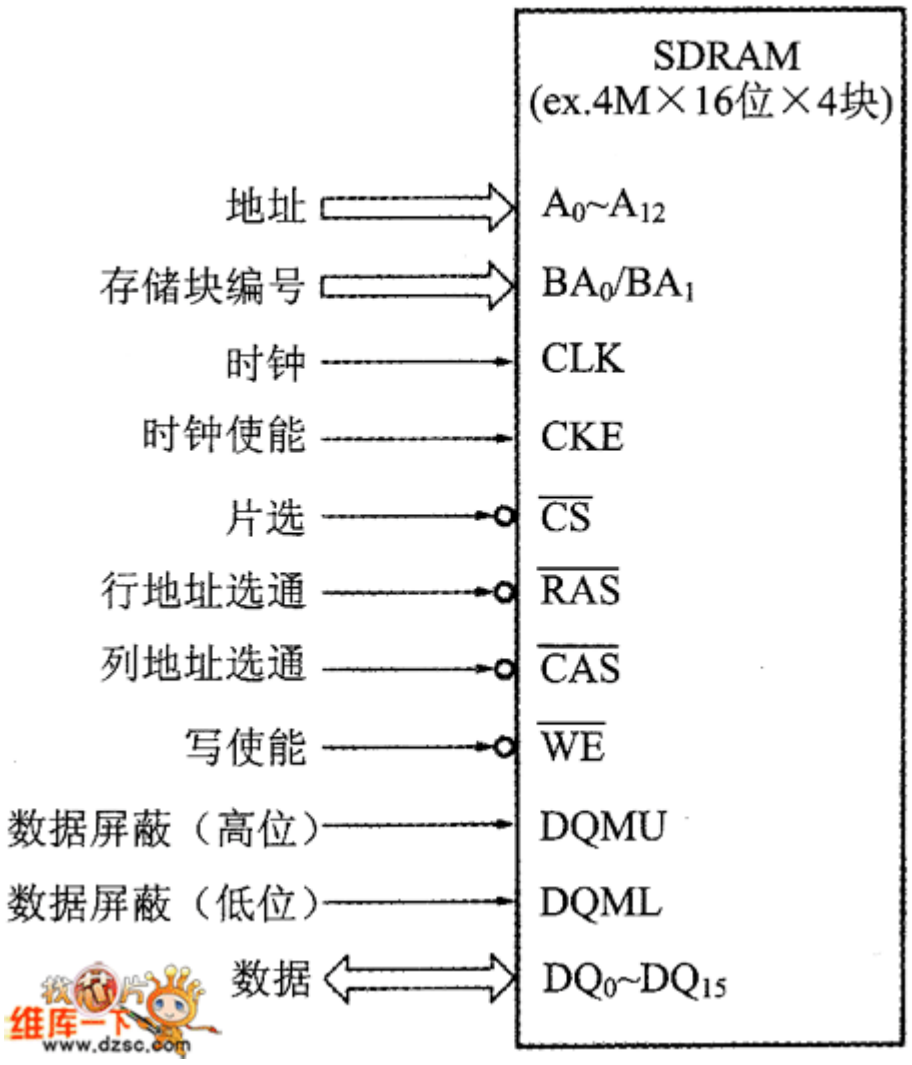


图 1　SDRAM 的信号

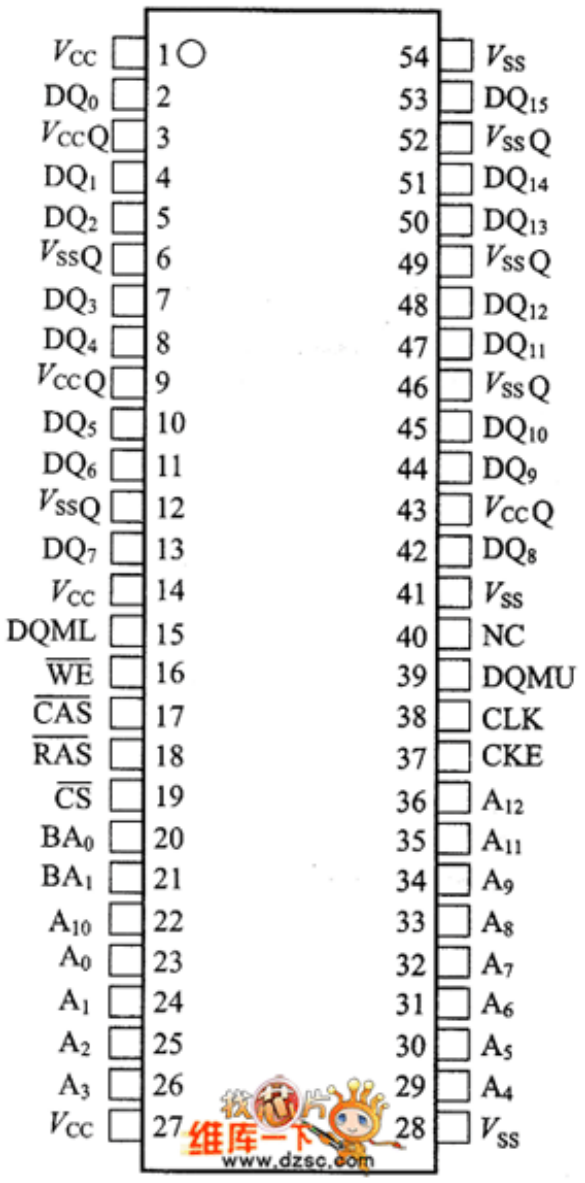


图 2　HM5225165B 的引脚配置

作为 4M 字×16 位×4 块（156M 位）结构的 SDRAM 的例子，日立的 HM5225165B 的引脚配置与框图分别如图 2 及图 3 所示。

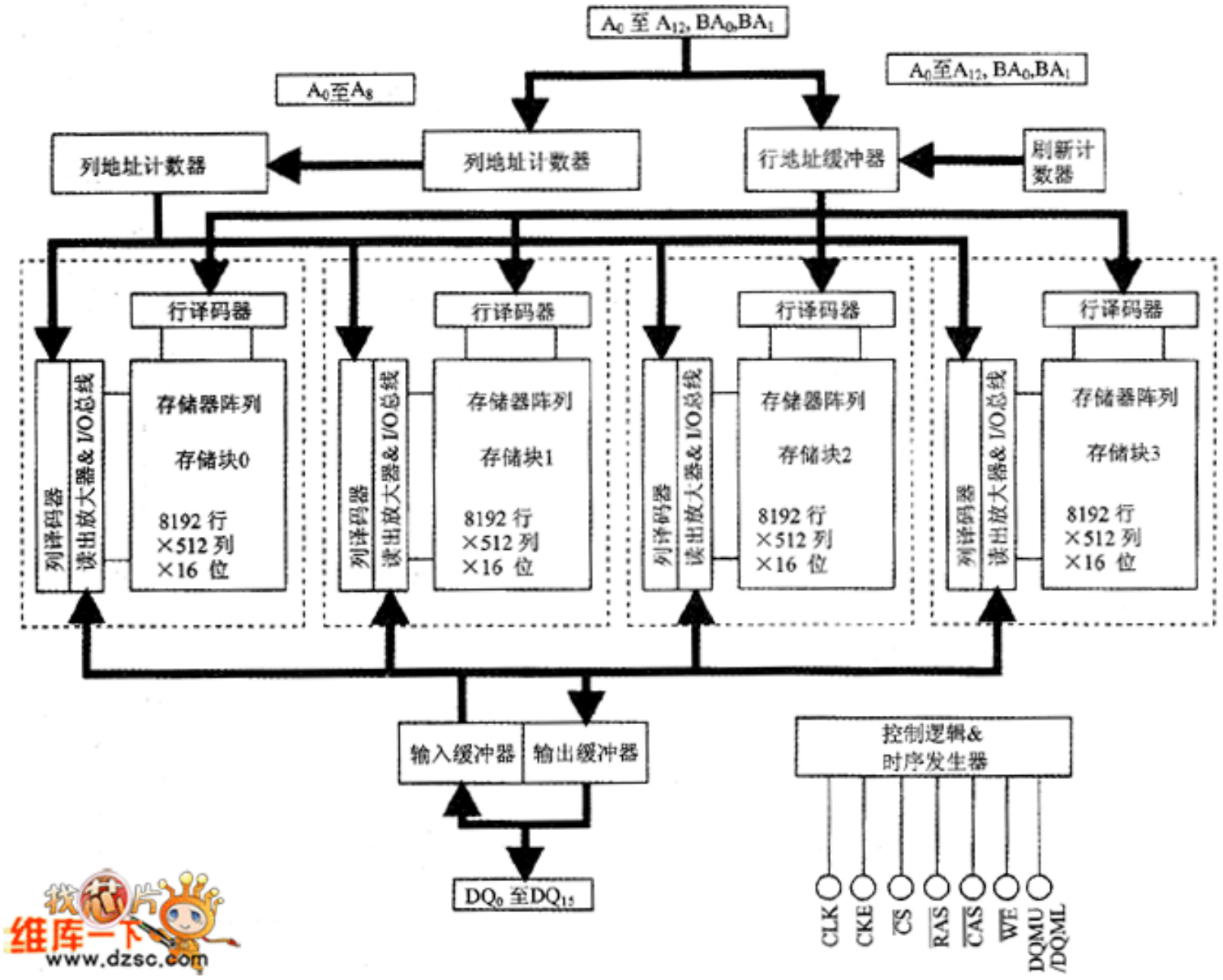


图 3　HM5225165B 的框图

接着，我们针对这些信号进行简单的说明。

▲　A0～A12（地址）

这是地址总线，与异步 DRAM 相同分为行地址与列地址。当赋予行地址时，使用 A0～A12；当赋予列地址时，使用 A0～A8（列地址时的 A9～A12 为无效），一页具有 512 字（Word），而且由于具有 4 个存储体，所以在同一行地址可以访问 2K 字的区域。

A10 也作为指令被应用，是比较特殊的引脚。当进行读 / 写操作时，在赋予列地址的时候，A10 成为是否进行自动预充电操作（后述）的选择信号的输人引脚。HM5225165B 不利用 A10 作为列地址，但相同容量的 16M 字×4 位×4 块结构的 HM5225405B 则利用列地址 A0～A9 以及 A11 共计 11 位，A0 为指定自动预充电。

另外，同步 DRAM 具有模式寄存器，可以进行突发传输操作的设定以及 CAS 延迟（发出读指令后，数据被输出前的时钟数）的指定等，指定时，为了进行寄存器值的设定，A0～A12 以及 BA0，BA1 被利用。

▲　BA0、BA1（存储块地址）

HM5225165 内部被分割为 4 个存储块，各个存储块可独立进行操作。例如，它可以采用这样的方法进行访问，即为一个存储块提供行地址后，再为其他的存储块提供其他的行地址，然后再一次返回最初的存储块，提供列地址从而进行访问。

利用 BA0、BA1 指定存储块，双方都是低电平时，存储块 0 被选择；当 BA0 是高电平而 BA1 为低电平时，存储块 1 被选择；相反，BA0 是低电平而 BA1 是高电平时，存储块 2 被选择；当双方都是高电平时，存储块 3 被选择。

▲　CLK（时钟输入）

这是时钟输人信号。所有的信号输入输出都是与该时钟的上升沿同步进行的。

▲　CKE（时钟使能）

这是决定下一周期的时钟是否有效的引脚，一般保持在高电平状态，但加人到省电模式及自刷新中时，可将其设置为低电平，以使系统处于非操作状态。

▲　/CS（片选）

这是片选输入信号。当该引脚无效（成为高电平）时，输入信号被忽略。内部操作（存储块激活及突发操作）本身即使当/CS 处于高电平状态时，也将被执行。

当该引脚有效（成为低电平）时，所赋予的控制信号及地址等是有效的。

▲　/RAS、/CAS、/WE

虽然名称本身与以前的异步 DRAM 相同，在某种程度上感觉是在异步 DRAM 中的处理方式，但功能上具有相当大的差别，它采用结合 3 条信号线指示操作的方法，详细说明将在后面进行。

▲　DQMU / DQML（DQ Mask High / Law）

利用该信号进行数据位的屏蔽，DQMU 对应于 DQ8～DQ15，DQML 对应于 DQ0～DQ7。读操作时，如果该信号为高电平，则数据位被屏蔽，输出缓冲器变为高阻抗状态，不能进行数据输出。写操作时，如果该信号为高电平，则不能向相应位的内部存储器单元进行写入。如果该信号为低电平，则读操作时 DQn 被驱动，写操作时可向内部单元进行写人操作。

▲　DQ0～DQ15（数据）

这是数据输人输出引脚，DQ0～DQ7 是低位字节，DQ8～DQ15 为高位字节，分别通过 DQML 及 DQMU 进行存取屏蔽，因此可以以 8 位为单位进行输人输出。

● 6.4.2　SDRAM 指令

SDRAM 也具有/RAS、/CAS 及/WE 信号，其名称与异步 DRAM 相同，功能上也存在相似的地方，但其实际的处理方式是通过 3 条线的结合，对 SDRAM 发出指令。在异步 DRAM 的情况下，例如，如果使/CAS 信号在/RAS 信号前有效，则成为/CAS 先于/RAS 有效刷新，形成按顺序赋予指令的形式。但在 SDRAM 的情况下，通过各个控制线状态的组合而形成指令，这是 SDRAM 与异步 DRAM 较大的不同。表示各信号的组合与操作列表。

表　SDRAM 的指令表

指　令	简　称	CLKE		输入信号						
		$(n-1)$	(n)	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA_n	A_{10}(AP)	A_n
模式寄存器设置	MRS	H	X	L	L	L	L	操作代码		
自动刷新	REF	H	H	L	L	L	H	X	X	X
开始自刷新	SELF	H	L	L	L	L	H	X	X	X
自刷新结束	SELX	L	H	L	H	H	H	X	X	X
		L	H	H	X	X	X	X	X	X
激活 & 行地址锁存	ACTV	H	X	L	L	H	H	存储块	行地址	
数据读	READ	H	X	L	H	L	H	存储块	L	列地址
附加自动预充电数据读	READ_A	H	X	L	H	L	H	存储块	H	列地址
数据写	WRITE	H	X	L	H	L	L	存储块	L	列地址
附加自动预充电的数据写	WRITE_A	H	X	L	H	L	L	存储块	H	列地址
指定存储体预充电	PRE	H	X	L	L	H	L	存储块	L	X
所有存储体预充电	PALL	H	X	L	L	H	L	X	H	X
断电状态突发		H	L	H	X	X	X	X	X	X
		H	L	L	H	H	H	X	X	X
由断电状态恢复		L	H	H	X	X	X	X	X	X
		L	H	L	H	H	H	X	X	X
输入忽略	DESL	H	X	H	X	X	X	X	X	X
无操作	NOP	H	X	L	H	H	H	X	X	X

通过这些指令，SDRAM 内部发生状态的迁移，如图所示，不能发送在该状态迁移图中与当前状态无关的指令（如在 IDLE 状态不能直接发出 WRITE 指令）。

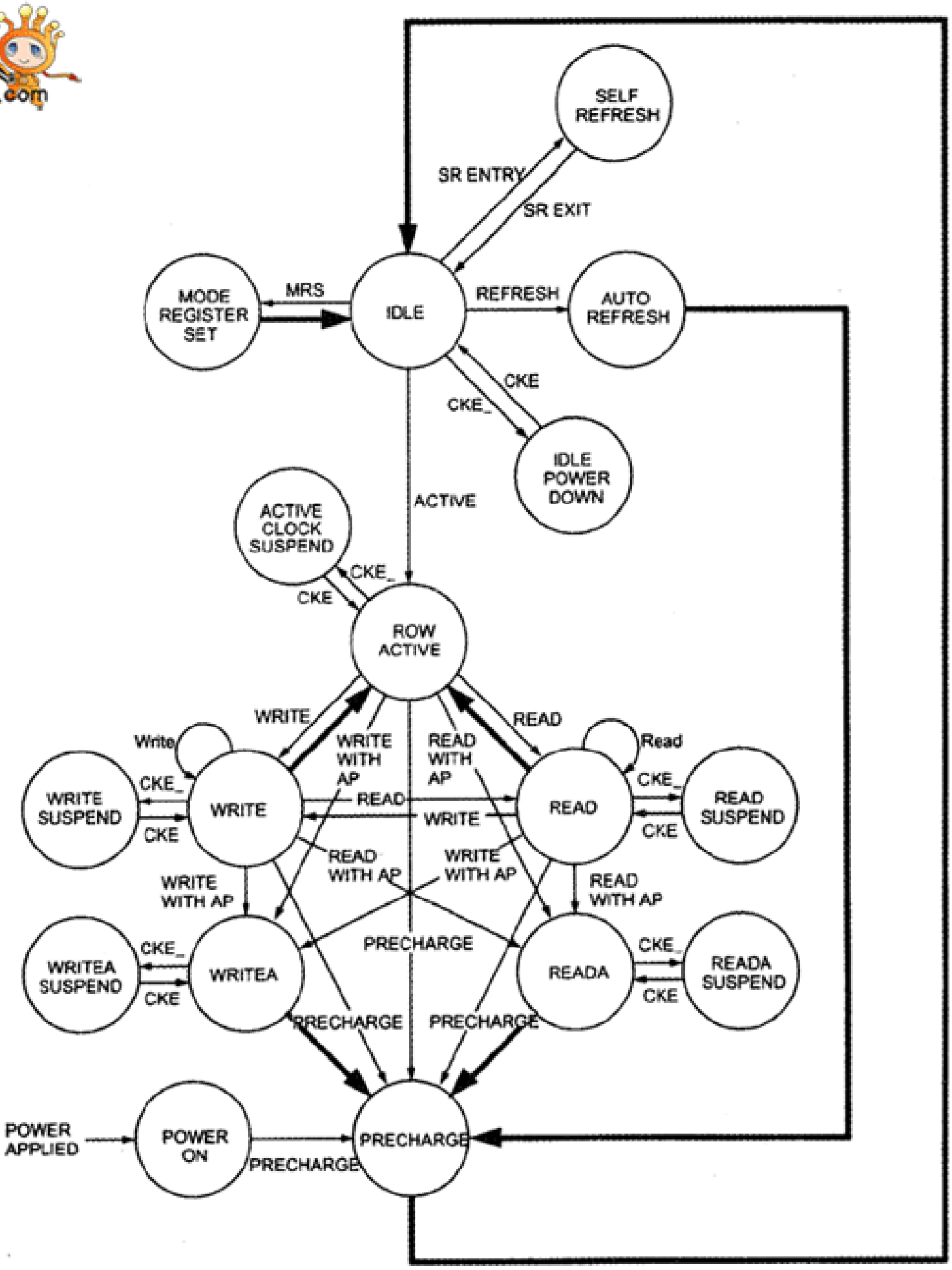


图 SDRAM 的状态迁移

▲ 模式寄存器设置（MRS）

SDRAM 具有模式寄存器，通过该模式寄存器，可以切换 SDRAM 的操作模式。模式寄存器的设置如图 1 所示，可以说不是通过改变数据而是通过改变地址进行操作的。

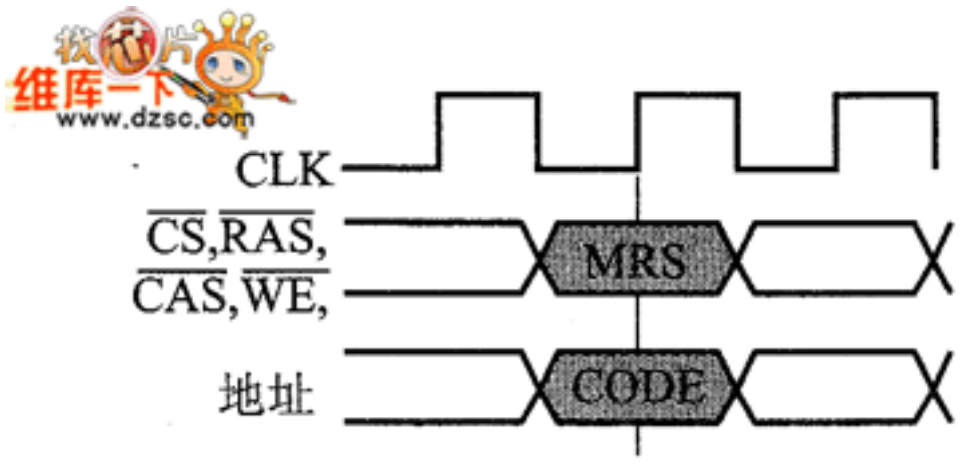


图 1 SDRAM 的模式寄存器存取操作

模式寄存器的引脚配置如下图 2 所示：

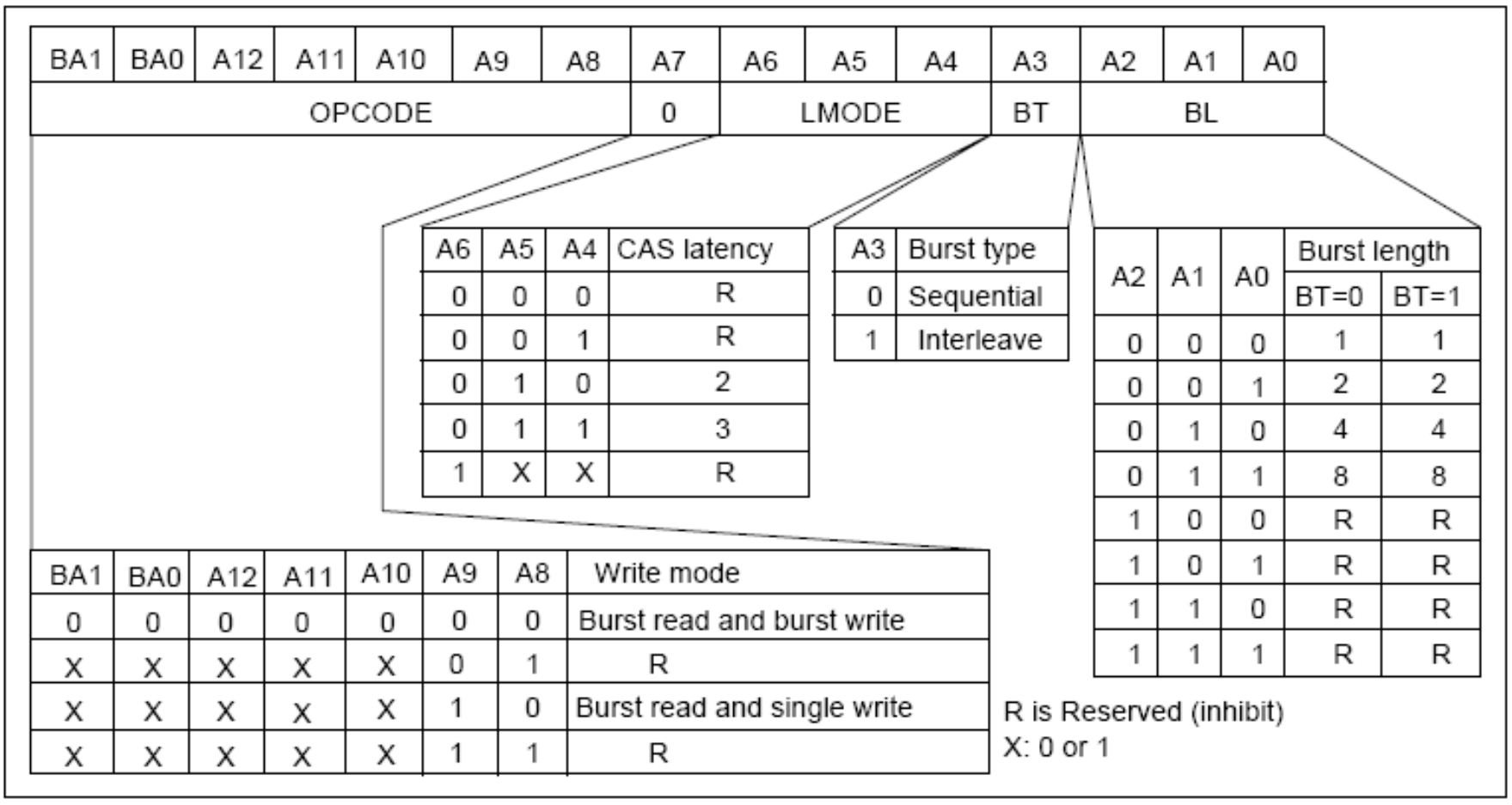


图 2　模式寄存器的引脚配置示列

（1）OPCODE（操作代码：BA0 / BA1、A8～A12）

这是写入模式的设置。

• Burst read and burst wrlte（突发读与突发写）

写操作时进行突发传输，起始地址是写操作开始时的列地址突发传输的字数是由突发长度（BL：A0～A2）指定的大小。

• Burst read and single wrlte（突发读与单一写）

写操作时不进行突发传输，只能是相当于一个字的写操作。

（2）LMODE（/CAS 延迟时间设定：A4～A6）

异步 DRAM 的情况下，从/RAS 及/CAS 有效到数据输出所需要的时间是规定以 ns 为单位。而在同步 DRAM 的情况下，是通过第几个时钟指定是否输出的。

虽然 CAS 延迟时间（CL）越小理所当然存取速度就越快，但由于与 DRAM 内部的操作关联，因而不能随便缩短 CAS 延迟时间。在判断以多少 MHz 操作以及 CAS 延迟时间取多大的值合适等问题时，需要查阅数据手册。

例如，HM5225165BTT-75 的时钟频率最高可为 133MHz，但以 133MHz 操作时的 CAS 延迟时间为 3；而以 100MHz 进行操作时的 CAS 延迟时间为 2。

当以 100MHz 使之操作时，在发出读指令后的第 2 个时钟（20ns 后）提取数据。而以 133MHz 进行操作时，由于将在第 3 个时钟（约 22.6ns 后）确定数据，因而如果只考虑单一的读操作传输速度，那么以 100MHz 进行操作的情况比较有利。事实上，利用突发传输的情况是非常普遍的，存在反常现象的情况只限于此。例如，传输 4 字时，由于从第 2 个字后是每隔一个时钟输出的，因此需要 CAS 延迟时间＋3 个时钟的时间。

当以 100MHz 进行操作、CAS 延迟时间为 2 时，存取速度为 50ns，而当以 133MHz 进行操作、CAS 延迟时间为 3 时，存取速度约为 45ns，因而 133MHz 的操作速度快了将近 10％。

（3）BT（突发类型：A3）

同步 DRAM 与管道突发式 SRAM 等相同，具有对应连续存取主机某一连续区域的突发传输的操作模式。由该引脚指定突发操作的顺序（突发顺序）是线性突发顺序还是交叉存取突发顺序。

突发传输时，主机只需要提供所存取的起始地址，以后的地址将由同步 DRAM 端自动生成。

突发传输中低位地址如何变化的总结如图 3 所示。最具代表性的 x86 系列的奔腾处理器采用的是交叉存取突发顺序，而其他处理器一般都是以线性突发顺序进行操作的。

脉冲时间＝2

地址(10 进制)	起开始地址	
A_0	线性	交叉存取
0	0,1	0,1
1	1,0	1,0

脉冲时间＝4

起始地址		地址(10 进制)	
A_1	A_0	线性	交叉存取
0	0	0,1,2,3	0,1,2,3
0	1	1,2,3,0	1,0,3,2
1	0	2,3,0,1	2,3,0,1
1	1	3,0,1,2	3,2,1,0

脉冲时间＝8

起始地址			地址(10 进制)	
A_2	A_1	A_0	线框	交叉存取
0	0	0	0,1,2,3,4,5,6,7	0,1,2,3,4,5,6,7
0	0	1	1,2,3,4,5,6,7,0	1,0,3,2,5,4,7,6
0	1	0	2,3,4,5,6,7,0,1	2,3,0,1,6,7,4,5
0	1	1	3,4,5,6,7,0,1,2	3,2,1,0,7,6,5,4
1	0	0	4,5,6,7,0,1,2,3	4,5,6,7,0,1,2,3
1	0	1	5,6,7,0,1,2,3,4	5,4,7,6,1,0,3,2
1	1	0	6,7,0,1,2,3,4,5	6,7,4,5,2,3,0,1
1	1	1	7,0,1,2,3,4,5,6	7,6,5,4,3,2,1,0

图 3 突发顺序

（4）BL（突发长度：A0～A2）

该引脚设定在突发传输操作中进行多少字的传输，HM5225165 如图所示，可以从 1，2，4，8 中进行选择。在目前个人计算机所使用的 CPU 中，突发长度一般为 4 字。

●6.4.3　同步 DRAM 的存取操作示例

▲同步 DRAM 的读操作

同步 DRAM 的读操作示例如图所示，所有操作都是以时钟的上升沿为基准进行的，与前面的状态迁移图结合相信会更容易明白。

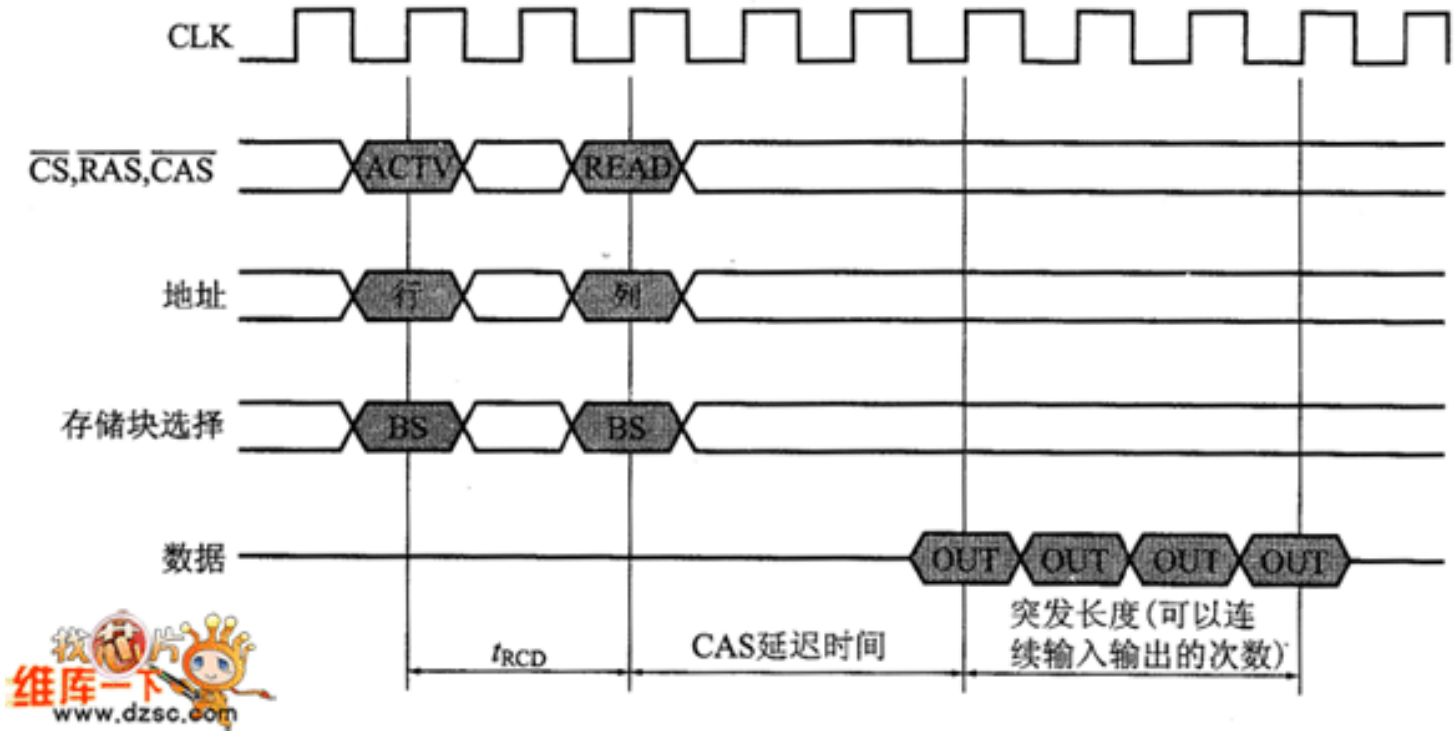


图 SDRAM 的读操作

（1）行地址与存储块编号的指定

首先，因为同步 DRAM 处于 IDLE 状态，所以在此发出利用了/RAS、/CAS、/WE 及/CS 信号的 ACTV 指令。同时分别赋予 A0～A12、BA0 / BA1 行地址和存储块编号，据此激活所指定的存储块，移向 ROW ACTIVE 状态。此后如果等待 tRCD 时间，则可以接受下一指令。这个 tRCD 时间记录于数据手册中，HM5225165B 的该时间为 20ns，因此，如果以 133MHz 进行操作则需要 3 个时钟；如果以 100MHz 进行操作则需要 2 个时钟。

（2）列地址的指定与读指令的发出

与经过了 tRCD 时间后的时钟上升沿同步，提供 READ 指令、列地址以及要进行读操作的存储块编号。在读操作中，可以进行相当于模式寄存器所设定的突发长度的连续读操作。

（3）CAS 延迟时间

发出 READ 指令后，经过模式寄存器所设定的 CAS 延迟时间（CL）的时钟周期后，将输出数据。图中表示了 CAS 延迟时间为 3 时的操作。

（4）数据的提取

经过 CAS 延迟时间后，模式寄存器所指定的突发长度（BL）的数据将连续输出，这样就可以提取数据。图中表示了 BL＝4 时的操作。如果相当于突发长度的数据输出结束，则 DRAM 的输出缓冲器自动变为高阻抗状态。

▲同步 DRAM 的写操作

同步 DRAM 的写操作如图所示，与读操作相同，都是与时钟上升沿同步地赋予指令及数据的。

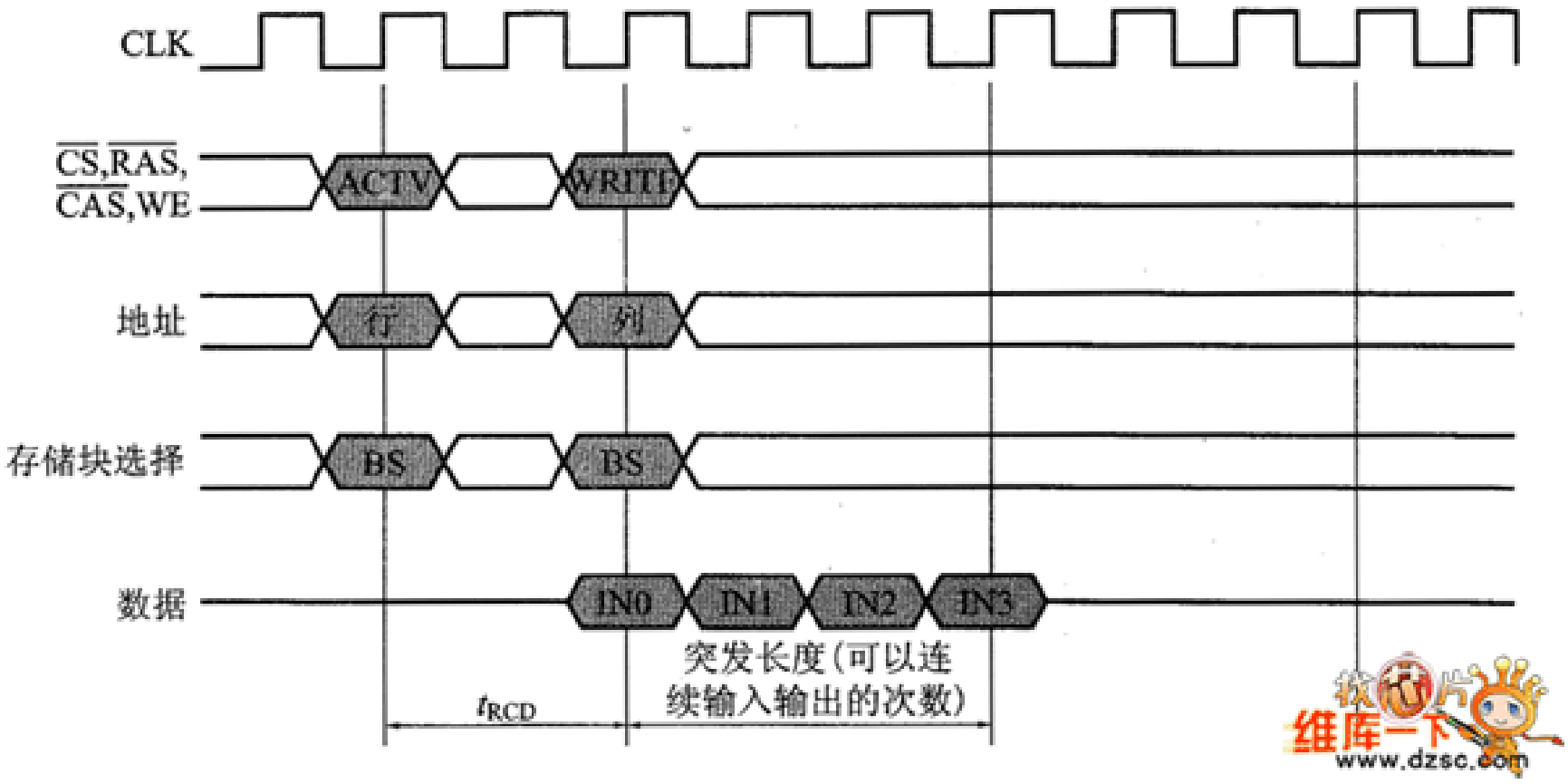


图 SDRAM 的写操作

（1）行地址与存储块编号指定

向处于 IDLE 状态的同步 DRAM 发出 ACTV 指令，同时赋予行地址和存储块编号，据此，激活相应的存储块，使之处于能够接受下一写指令的状态。

（2）发出写指令

发出 ACTV 指令后，只要经过 tRCD 时间的等待，就可以处于能接受下一指令的状态，与列地址、写入数据一起发出 WRITE 指令。与读操作时不同的是，不必在意延迟时间，可与指令同时赋予数据。

（3）数据的连续写入

当进行突发写操作时，可在此之后连续赋予数据。只要连续赋予模式寄存器的突发长度（BL）所指定的长度，写人目的地的地址就可在 DRAM 内部自动更新，进行突发写操作。图中表示了 BL＝4 时的操作。

6.5　DDR-SDRAM

●6.5.1　DDR-SDRAM 的信号

DDR SDRAM 的信号例如图 1 所示，在这里，作为 4M×16 位×4 块结构的 256M 位的 DDR SDRAM，我们以 ELPIDA 公司（NEC 与日立的合资公司）的 HM5425161B 为例进行说明。在同步 DRAM 的基础上添加的信号标注了※符号，与 DRAM 控制器的连接如图 2 所示。首先我们针对这些信号进行说明。

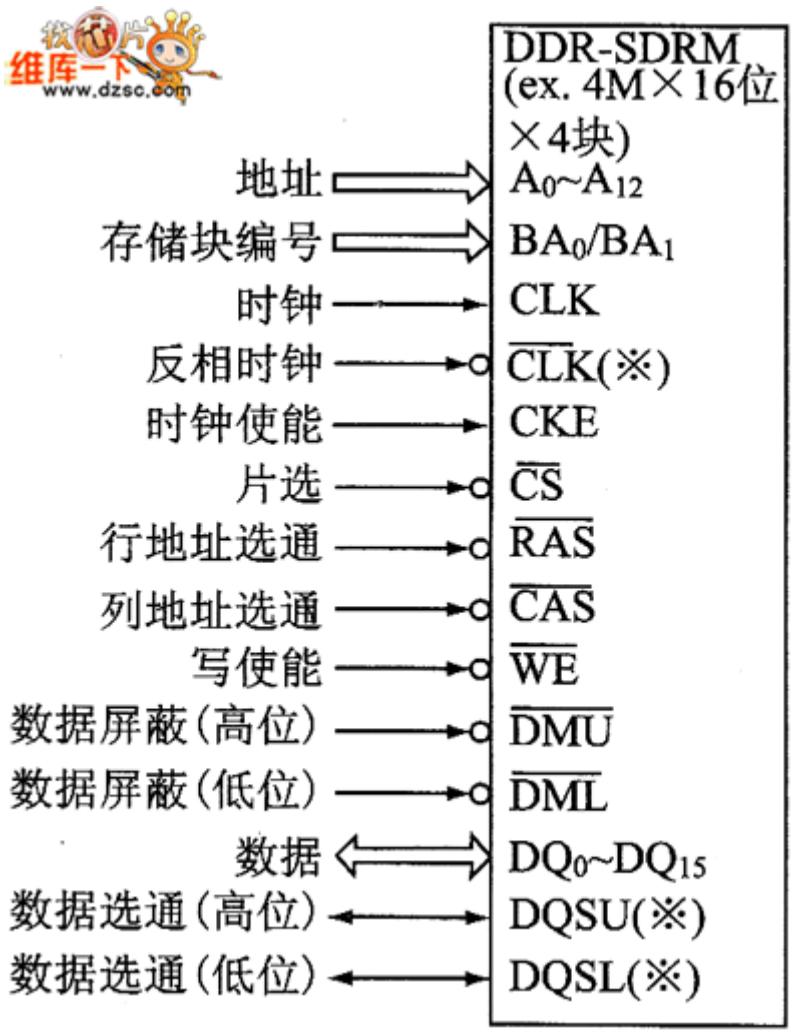


图 1　DDR－SDRAM 的信号

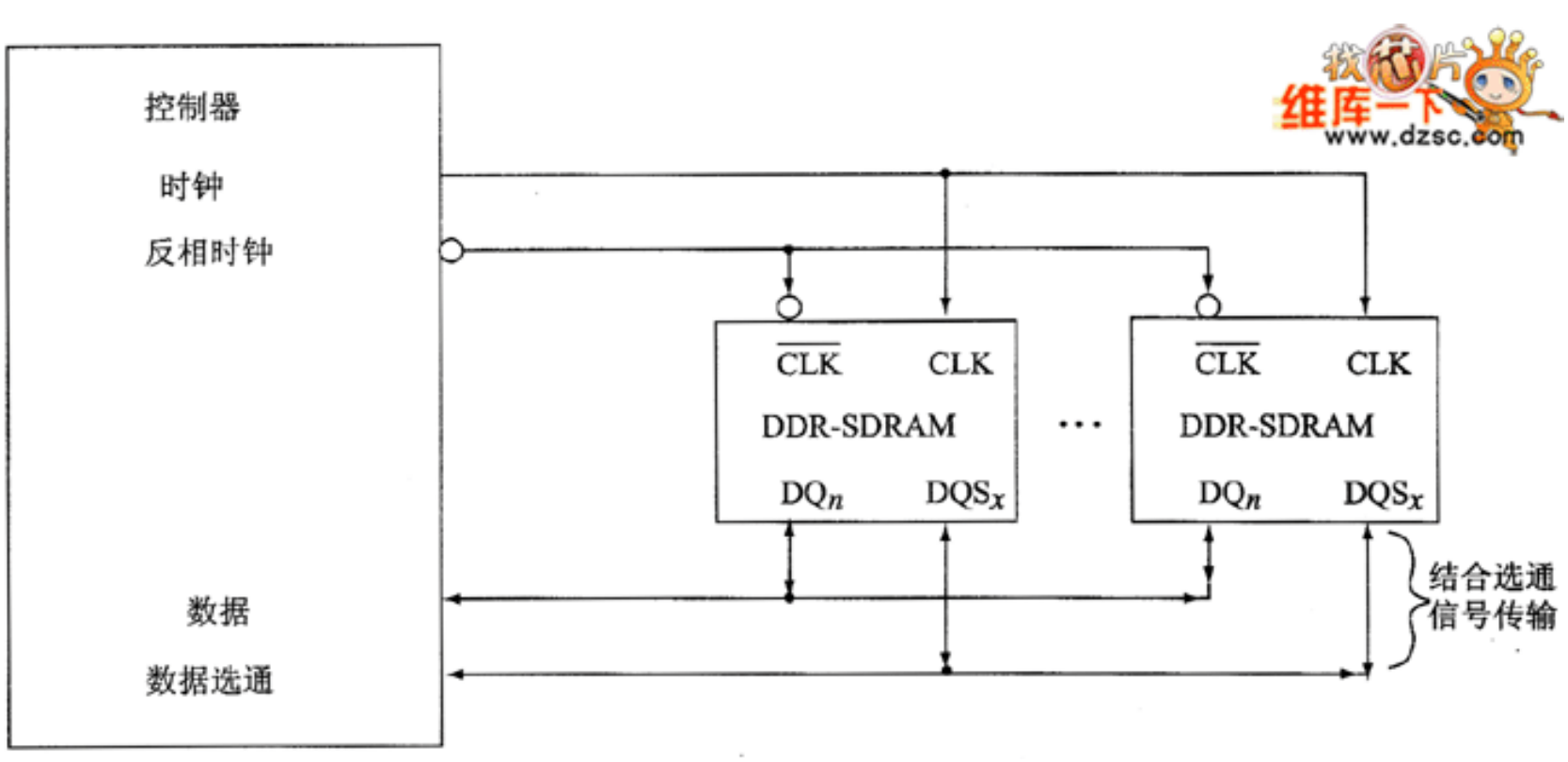


图 2　DDR－SDRAM 的连接

▲　/CLK（反相时钟）

同步 DRAM 只有一个时钟输入，与上升沿同步进行操作，而 DDR－SDRAM 同时也利用反相时钟。在 DMU / DML（数据屏蔽）、DQSU / DQSL（数据选通）和 DQn（数据）的采样时利用 CLK、/CLK 两种时钟。

因为在上述以外信号输入的采样时只利用 CLK，所以认为该信号只应用于数据传输中即可。

▲　DQSU / DQSL

在 DDR-SDRAM 的情况下，因为数据传输是非常快的，因此在 DRAM 控制器与 DRAM 元件之间存在信号偏移的问题。为此，在数据传输时，我们利用 DQSU / DQSL 判断数据是否确定。该信号可双向使用。

读操作时，如果接收到来自 DRAM 控制器的 READ 指令，则 DDR-SDRAM 将 DQS 信号设为低电平，然后结合数据切换 DQS。虽然 DDR-SDRAM 与同步 DRAM 在指令的传输上是相同，都在 CLK 的上升沿进行，但 DDR－SDRAM 的 CAS 延迟时间值采用整数或者整数＋0．5 的值，所以当 CAS 延迟时间是整数时，DQS 与 CLK 同相；当 CAS 延迟时间是整数＋0．5 时，DQS 与/CLK 同相。在主机方面，不是单纯地与时钟同步接受数据，而是根据是否切换了 DQS 信号来提取数据。

写操作时，DRAM 控制器在数据传输开始之前将 DQS 设置为低电平，数据确定后再进行切换 DQS 的操作。DDR－SDRAM 是要结合 DQS 信号提取数据的。

●6.5.2　DDR-SDRAM 的操作

▲DDR SDRAM 的读操作

DDR SDRAM 的读操作如图所示。发出 ACT 指令后，只经过 tRCD 时间后发出所要进行的 READ 指令，这一流程是与同步 DRAM 相同的，是与 CLK 的上升沿同步进行的。从图中我们可知道，之后的数据传输是以 2 倍的速率进行的。

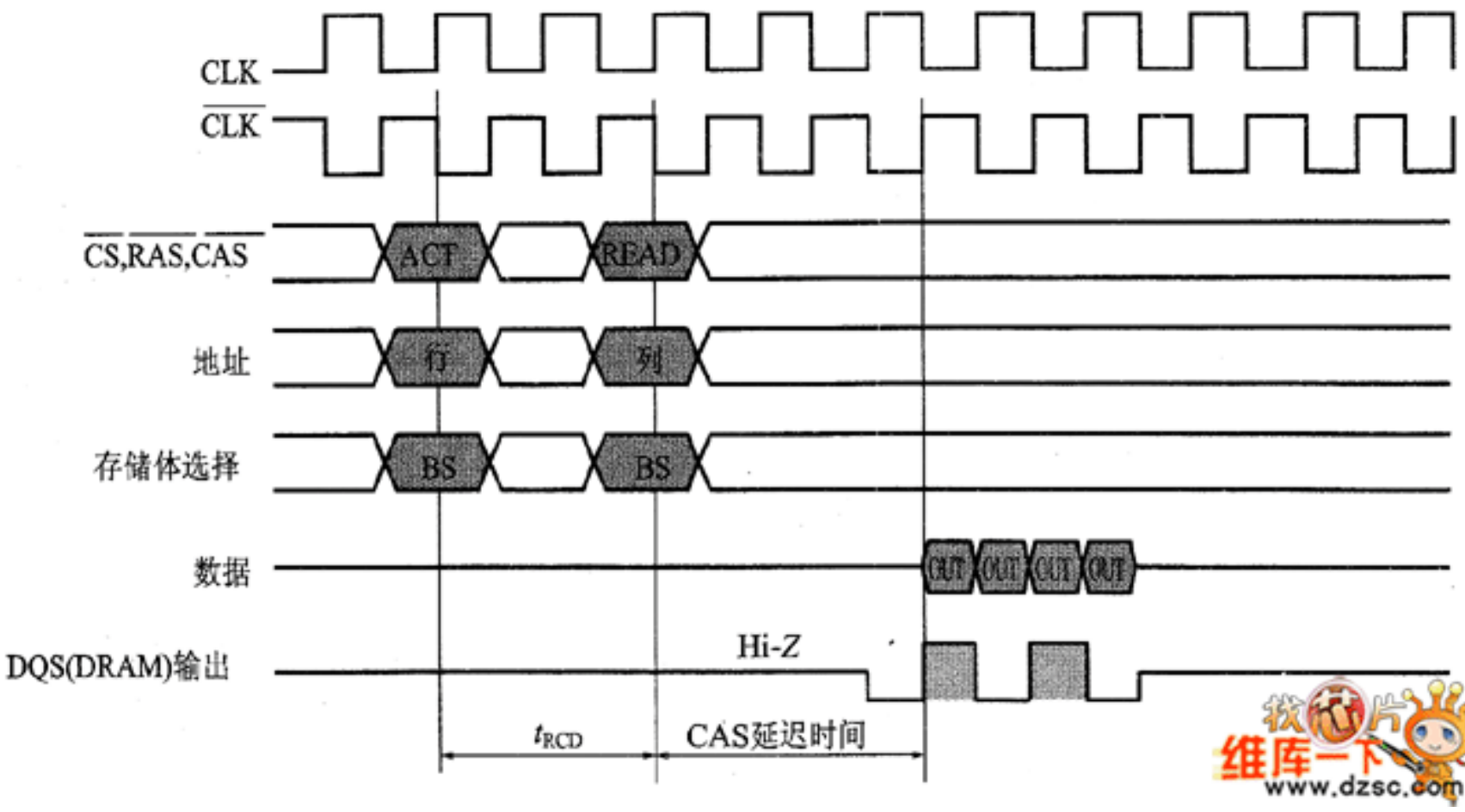


图　DDR SDRAM的读操作

由于可以利用1/2时钟，所以CAS延迟时间也不只是整数值，也有＋0.5的取值。该图中的操作实例的CAS延迟时间为2.5。

如图所示，DDR－SDRAM与数据一起驱动DQS信号，DQS与数据一起变化，所以主机方面等待这个变化若干个延迟时间后提取数据。

可以对相当于模式寄存器所设定的突发长度的数据进行连续读操作也是DDR－SDRAM与同步DRAM相同的地方。

▲DDR SDRAM的写操作

DDR SDRAM的写操作如图所示。仍然是与同步DRAM相同，随着ACT指令的发出而发出WRITE指令。但DDR－SDRAM数据不是与WRITE指令同时发出的，而是在一个时钟后赋予数据，这是与同步DRAM的不同之处。

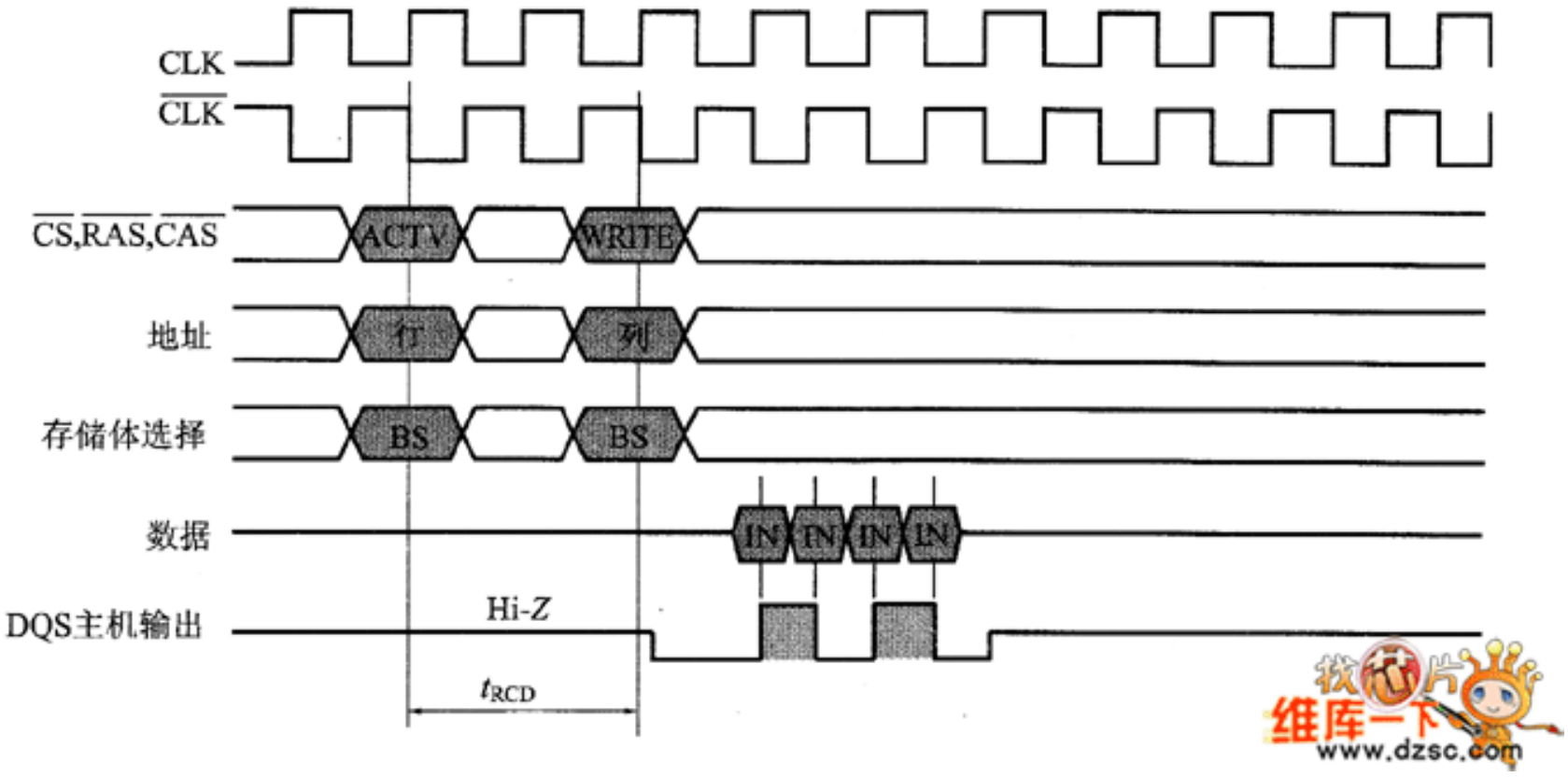


图 DDR－SRAM 的写操作

另外，DDR－SDRAM 锁存数据的时序不是利用 CLK，而是利用 DQS 信号。在 DRAM 控制器端确定数据后等待若干个延迟时间，选通 DQS。而 DDR SDRAM 端则在该变化沿处锁存数据。

进行读操作时，DDR SDRAM 输出的 DQS 具有与数据同步的状态信号；而进行写操作时 DQS 却成为选通信号，这是两者最大的不同。

进行读／写操作时的时序微调是在 DRAM 控制器上进行的，这也可以说是 DDR－SDRAM 使用上的特征。

6.6　直接总线式 DRAM

Direct Rambus DRAM（直接总线式 DRAM）可以说是原来的 Rambus　SDRAM 的升级版，在今后个人计算机的广泛应用上，人们对它给予较大的期望。

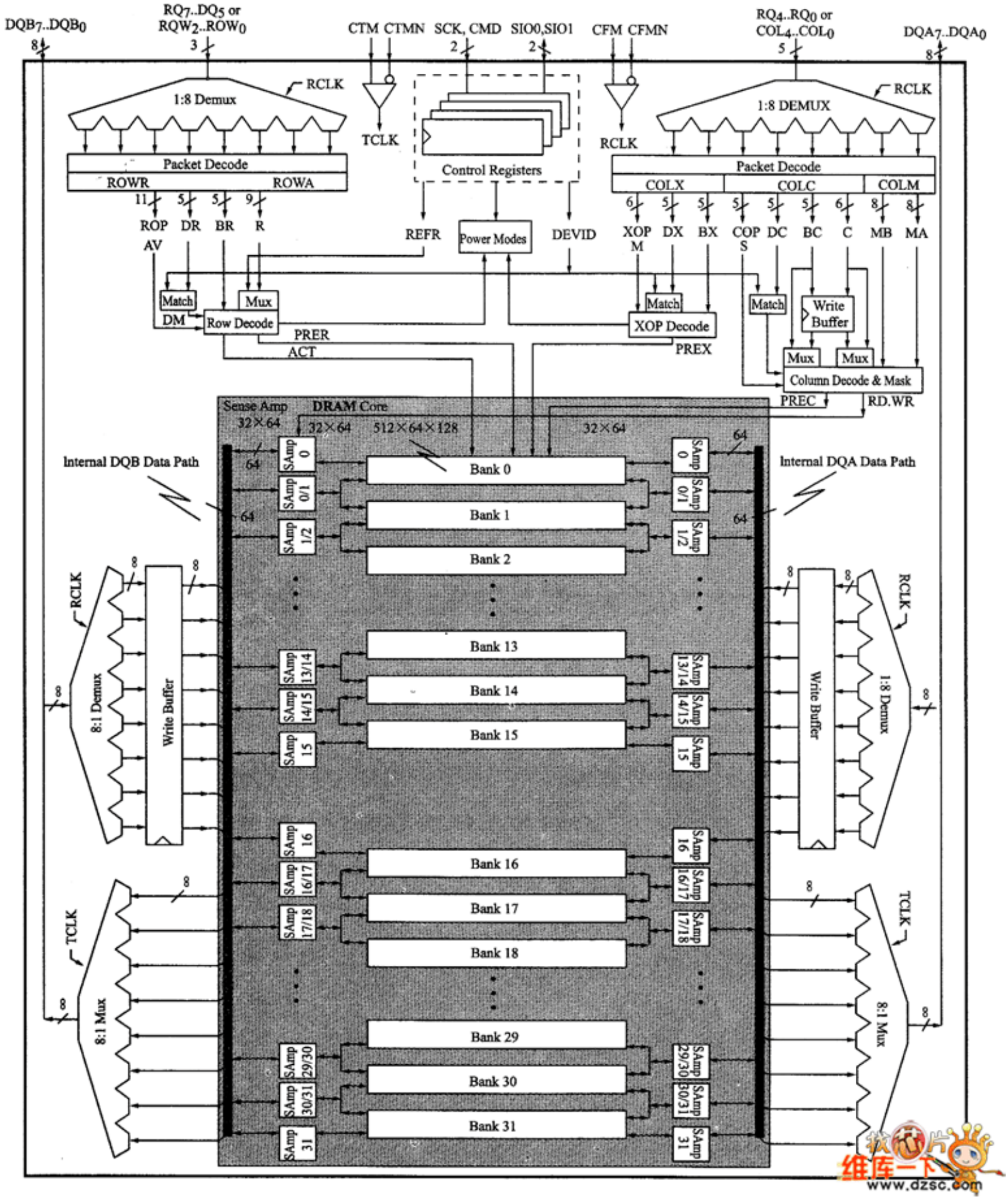


图 μPD488448 的内部框图

Direct Rambus DRAM 内部 DRAM 单元本身与其他的 DRAM 器件没有区别，通过将外部接口设置成指令包方式以及在信号电平等电气的接口部分的设计，使得在 400MHz 的高时钟频率上也可以利用时钟的两个沿进行操作。

●6.6.1　直接总线式 DRAM 的信号

作为 Direct Rambus DRAM，我们以 NEC（现在的 ELPIDA 公司）的 uPD488448 为例进行说明。该 DRAM 的结构为 8M 字×16 位×32 块。与 DDR SDRAM 等相比较，采用较多的存储块是 Direct Rambus DRAM 的特征之一。内部框图如上图所示，是相当复杂的，这也是成本提高的一个因素。

	A	B	C	D	E	F	G	H	J
12	GND		V_{DD}				V_{DD}		GND
11									
10	DQA7	DQA4	CFM	CFMN	RQ5	RQ3	DQB0	DQB4	DQB7
9	GND	V_{DD}	GND	GNDa	V_{DD}	GND	V_{DD}	V_{DD}	GND
8	GND	DQA5	DQA2	V_{DDa}	RQ6	RQ2	DQB1	DQB5	SIO1
7									
6									
5	SCK	DQA6	DQA1	V_{REF}	RQ7	RQ1	DQB2	DQB6	SIO0
4	VCMOS	GND	V_{DD}	GND	GND	V_{DD}	GND	GND	VCMOS
3	N.C.	DQA3	RQ0	CTMN	CTM	RQ4	RQ0	RQB3	N.C.
2									
1	GND		V_{DD}				V_{DD}		GND

图 1　μPD488448 的信号配置（俯视图）

信号配置如图 1 所示，以前的 DRAM 大多采用 TSOP 等封装，而 Direct Rambus DRAM 采用 BGA 封装。

Direct Rambus DRAM 的信号整理如图 2 所示，可知与 DDR－SDRAM 相比具有相当多的不同。

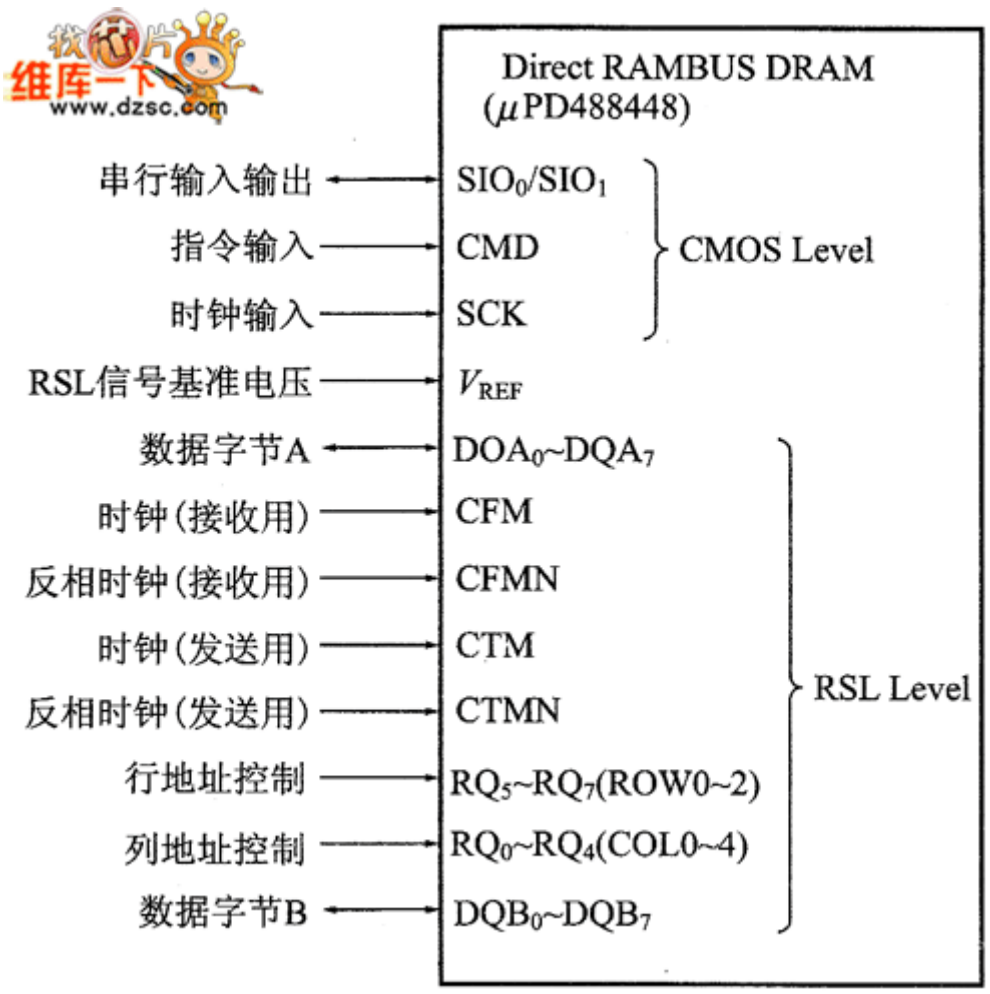


图 2 Direct Rambus DRAM 的信号种类

这些信号中，实际用于数据传输的只是标记为RSL Level的信号，看起来比较复杂。但事实上，由于时钟存在 4 个系统、数据存在 2 个系统，所以将这些进行整理，就是时钟、行地址控制、列地址控制以及数据这 4 种类型，因此信号的种类本身可以说不是那么复杂。下面我们针对这些信号进行简单的说明。

▲ CMD / SIO0 / SIO1 / SCK

为了对Rambus DRAM内部进行操作控制，组装了 30 个以上的控制寄存器（框图中央的上部）。为了访问这些寄存器而设计了CMD / SIO0 / SIO1 / SCK这 4 个信号。

这些信号是用于进行配置的，所以速度都相当的慢。SCK的周期时间最小为 1000ns（1μs），因而需要在1MHz以下进行操作。

▲ CTM / CTMN / CFM / CFMN（时钟）

CTMN、CFMN是分别与CTM（Clock To Master）、CFM（Clock From Master）配对的反相时钟信号。利用CTM、CTMN在器件内部生成发送时钟（TCLK），利用CFM、CFMN在器件内部生成接收时钟（RCLK），以便提取来自写人数据及ROW / COL引脚的指令等。

▲ DQA0～DQA7、DQB0～DQB7

这是进行读数据 / 写数据操作的信号。Direct RambusDRAM的数据宽为 8 位或者 16 位，μPD488448 是 16 位宽的Direct Rambus DRAM。由于在DRAM内部数据传输单位为 64 位，所以μPD488448 具有两个

64 位通道，通过 8 个周期（DirectRambus DRAM由于可利用时钟的两个变化沿所以是 4 个时钟周期）进行传输。

▲ RQ0～RQ7

这些引脚用于赋予控制指令及地址信息等，RQ0～RQ4、RQ5～RQ7 又分别称为COL0～COL4、ROW0～ROW2，这是为DirectRambus DRAM将指令及数据分组打包而形成的组合。关于封装将在后面详细叙述。

▲ VREF

在通常的数据传输中所使用的Direct Rambus DRAM信号是以称为RSL（Rambus Signaling Level，Rambus信号电平）的信号电平工作的，赋予这个标准电压的就是VREF引脚，VREF电压是由规范决定的，为 1.4V±0.2V。

●6.6.2 直接总线式 DRAM 的信号连接

Direct Rambus DRAM 的信号连接关系如图所示。与 DDR-SDRAM 最大的不同在于信号线是漏极开路输出以及时钟是以连续不断的方式往复的。

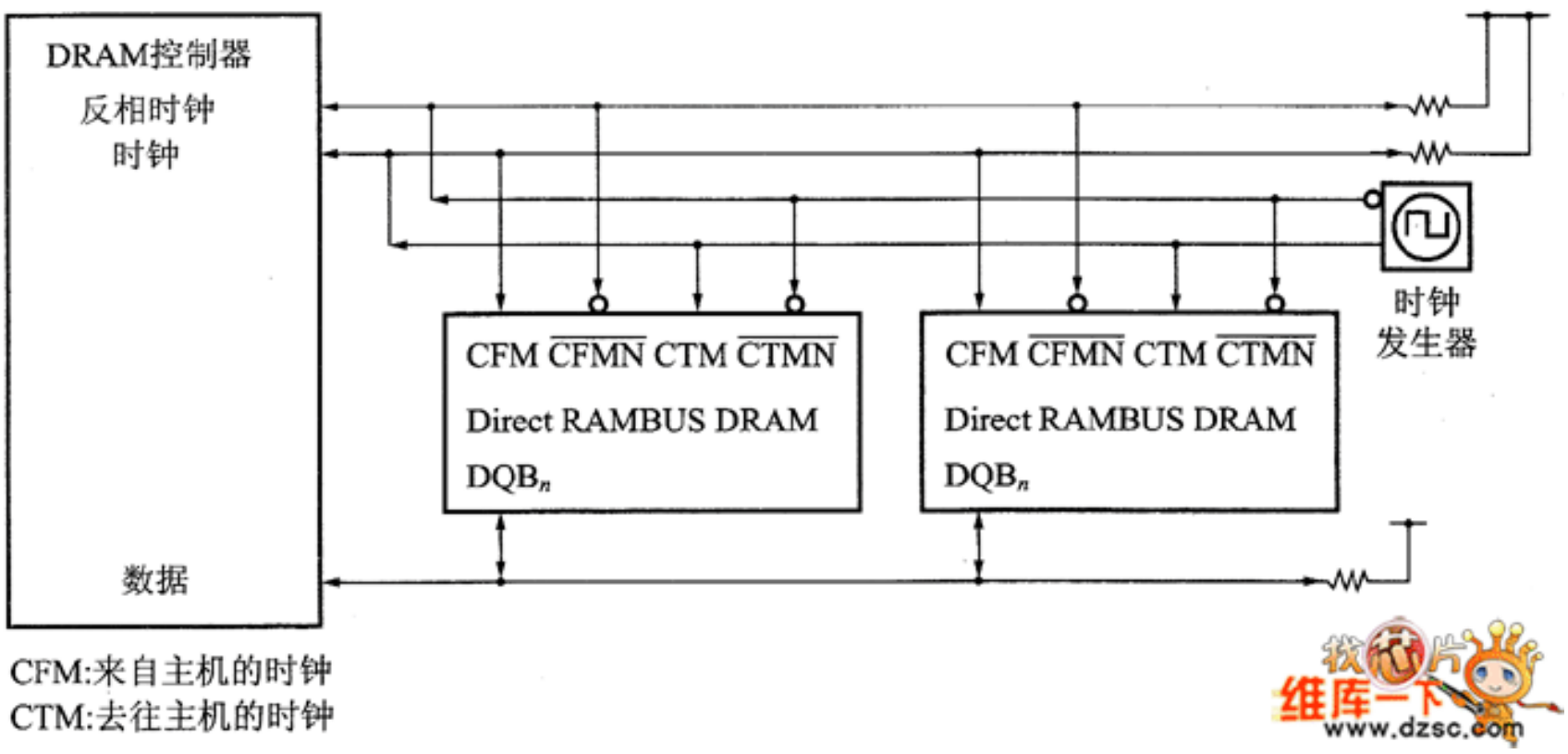


图 Direct Rambus DRAM 的信号连接关系

异步 DRAM、同步 DRAM 以及 DDR－SDRAM 等无论哪个输出都是有极性输出，是属于在高电平时驱动高电平的机制，而 Direct Rambus DRAM 利用漏极开路输出，可以对应于快速操作。

另外，采用 2 个系统时钟是处理时钟相位偏移的对策，DDRSDRAM 利用双向的选通信号实施时钟相位偏移的处理对策，而 Direct Rambus DRAM 预备了由 DRAM 向 DRAM 控制器方向的时钟及由控制器向 DRAM 方向的时钟两个系统，通过改变读操作与写操作时所利用的时钟，实施时钟相位偏移的处理对策，基本上采用了接近理想的处理方式。

图的右上部是时钟发生器，物理上放置于离 DRAM 控制器最远的 DRAM 的旁边，从这里按顺序给时钟布线，到 DRAM 控制器处再折回到最远的 DRAM 处，结束布线。

在这两个系统的时钟输人中，由 DRAM 向主机方向的称为 CTM（Clock To Master）；相反，由主机向 DRAM 方向的称为 CFM（Clock From Vaster），以便于区分。读操作时，也就是从 DRAM 输出数据时，Direct Rambus DRAM 与 CTM 时钟同步输出数据。如果时钟与数据信号的布线长度等相同，则时钟与数据具有相同的延迟时间到达 DRAM 控制器，所以 DRAM 控制器可以与时钟同步接受数据。

另一方面，当向 DRAM 进行写操作时，DRAM 控制器与时钟同步输出数据，由于该数据是与作为由 DRAM 控制器向 DRAM 方向的时钟 CPM 时钟一起传输的，所以只要 DRAM 端能与 CFM 时钟同步接受数据即可。

●6.6.3　直接总线式 DRAM 的操作概况

Direct RambuS DRAM 与同步 DRAM 等最大的不同在于将指令分组后进行若干次传输。在同步 DRAM 的情况下，说起指令，也就是通过/RAS、/CAS、/WE 信号操纵内部定序器。而在 DirectRambus DRAM 的情况下，指令以及器件地址等所有都通过分组汇总，以处理器间通信的形式进行传输。

为此，Direct Rambus DRAM 不存在像同步 DRAM 那样的地址及指令专用的引脚，因而可以减少引脚数目。

▲ 分组格式

Direct Rambus DRAM 的分组格式示例如图 1 和图 2。

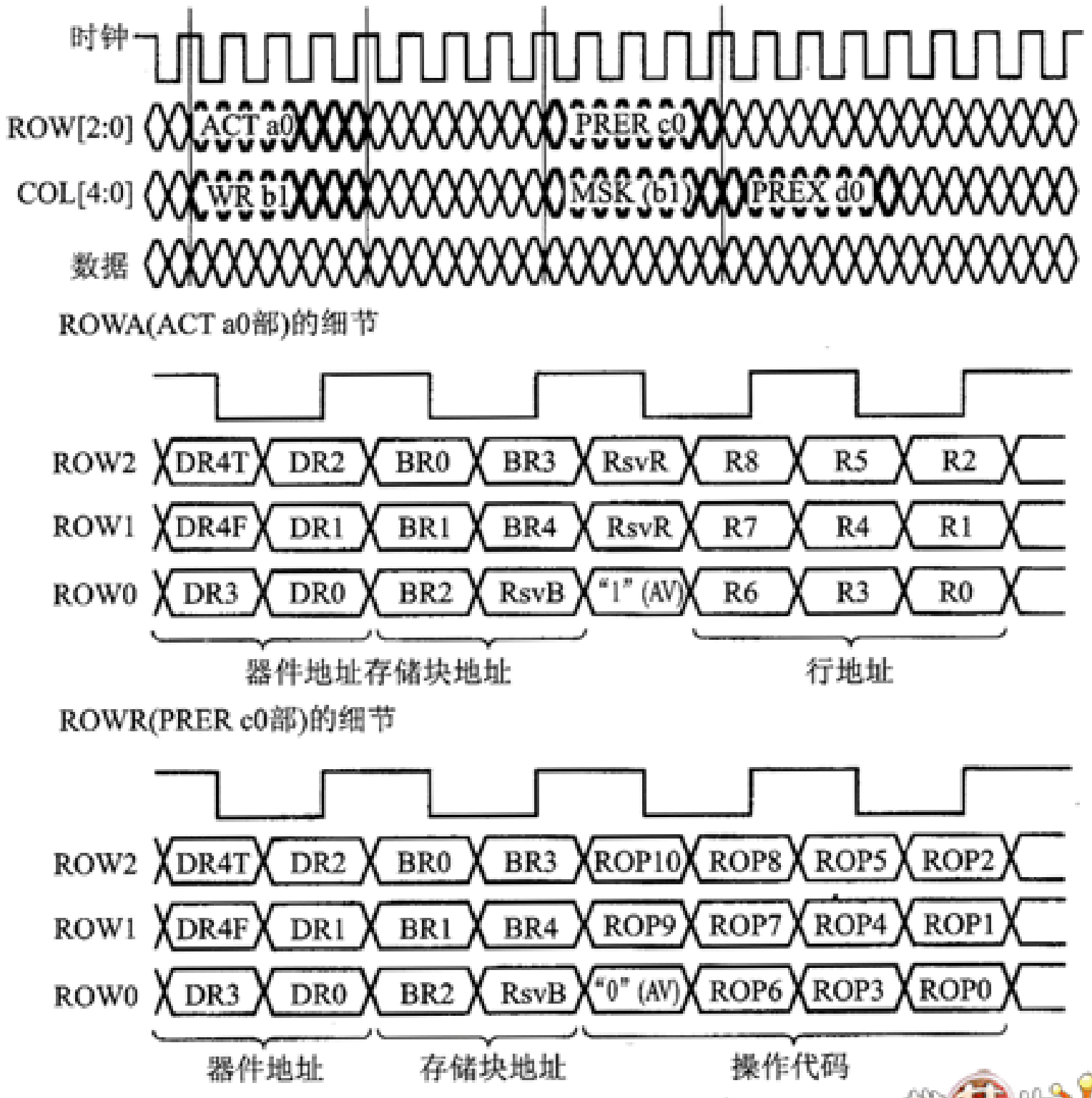


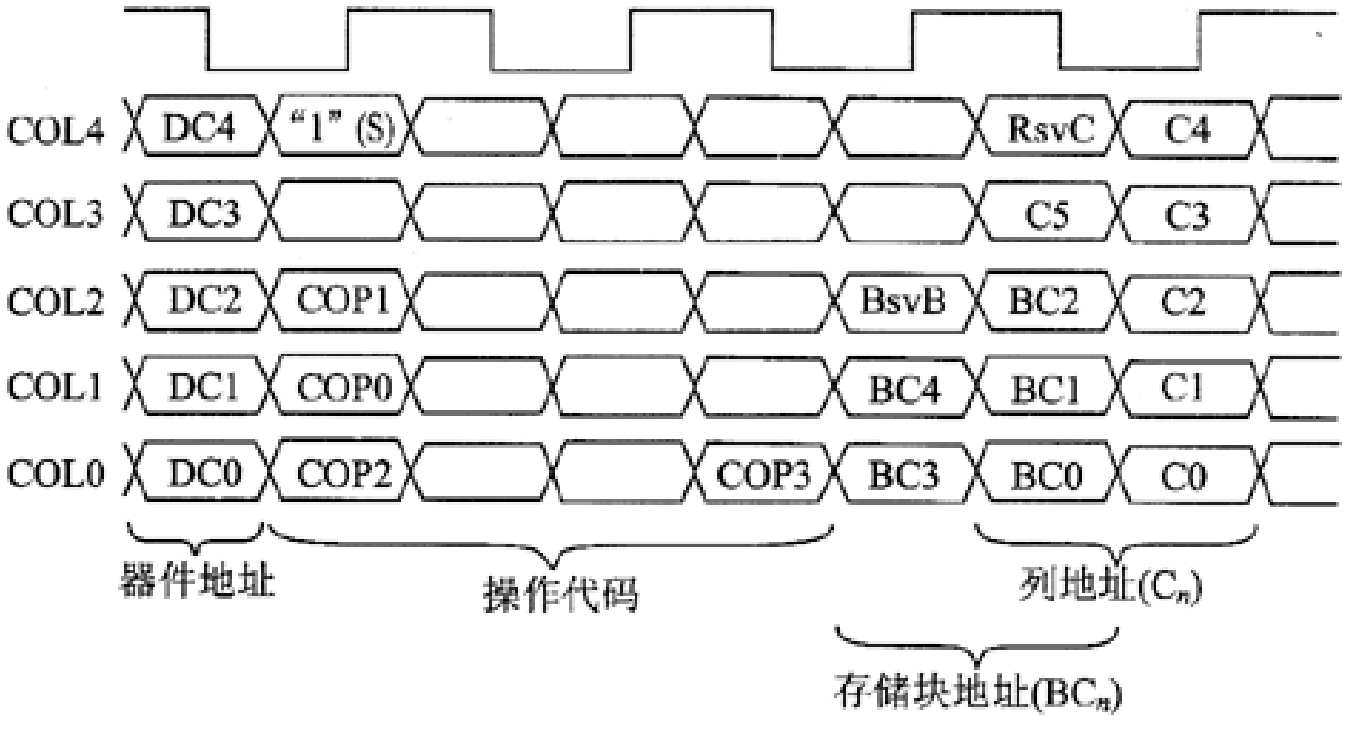
图 1 Direct Rambus DRAM 的分组格式（之一）

▲ ACT 指令

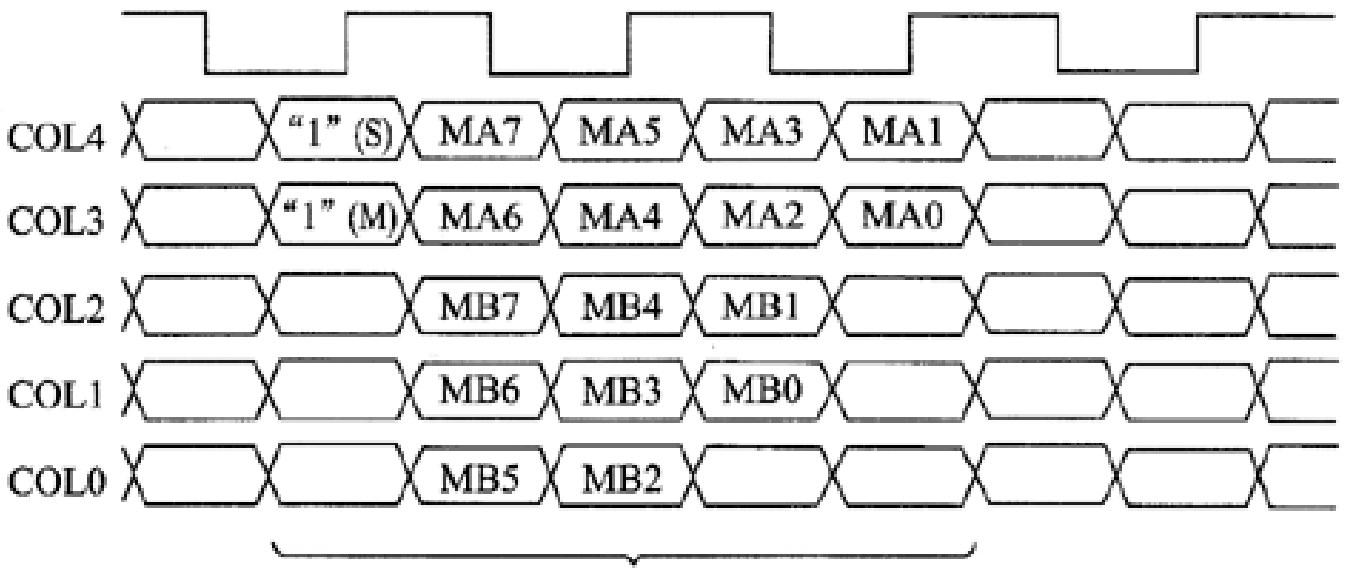
这是激活（Activate）指令。与同步 DRAM 的 ACT 指令相同，是指定行地址及存储块编号、激活器件的指令，图中写为 ACT a0。需要注意与同步 DRAM 不同的一点，就是 Direct Rambus DRAM 利用 ROW0～ROW0（RQ5～RQ7）以分组形式进行指定。

Direct Rambus DRAM 为各个器件分配器件编号（由控制寄存器设定），访问时根据该分组中的器件编号进行选择，进而指定存储块编号和行地址准备访问。行地址具有 9 位，需要从 512 根行地址中选择一根。

COLC分组(WR b1部)的细节



COLM分组(MSK (b1)部)的细节



COLX分组(PREX d0部)的细节

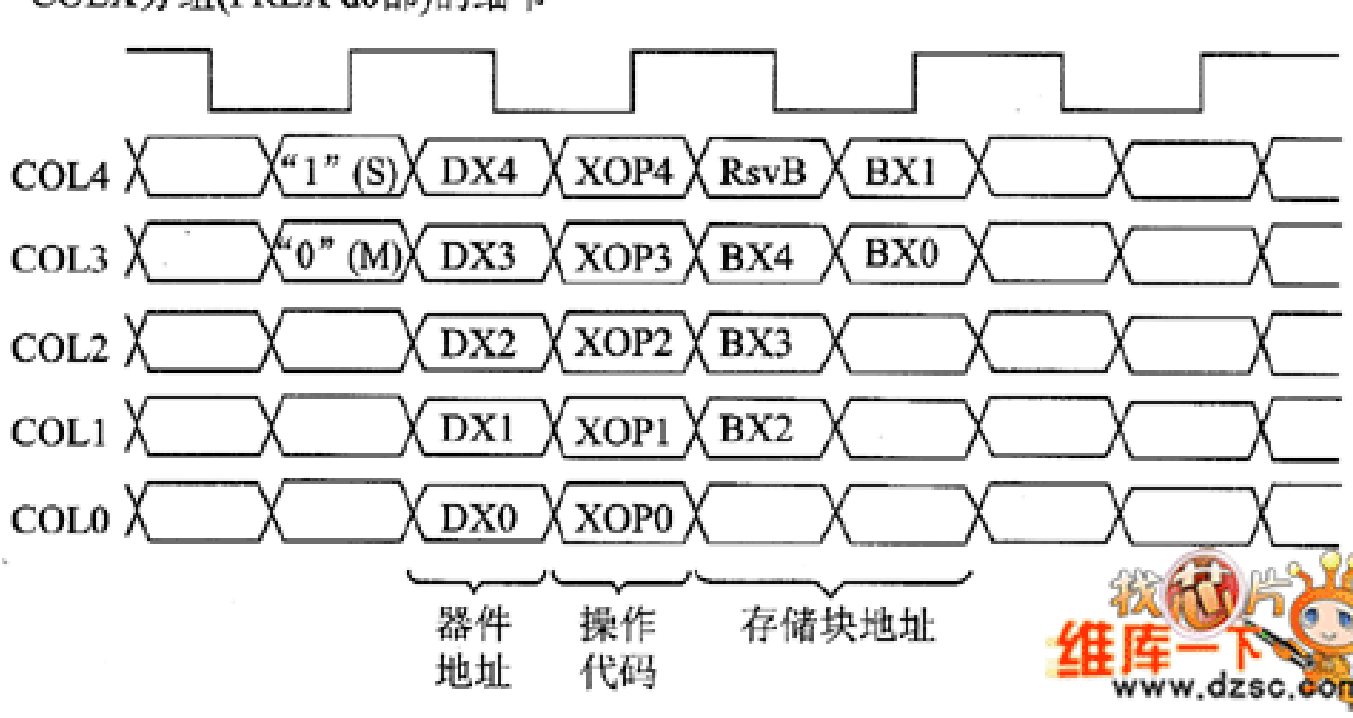


图 2 Direct Rambus DRAM 的分组格式（之二）

▲　PRER 指令

这是预充电指令。它应用于指定器件编号及存储块编号、释放读出放大器以及激活其他的行地址等方面。

▲　WR 指令

这是写指令。如果是读操作状态，则当然为读指令。在此，指定作为访问对象的器件地址以及开始访问的列地址（行地址在 ACT 指令时已经赋予）等，以便开始进行实际的存取操作。

▲　MSK

这是访问屏蔽指令。uPD488448 因为 8 位宽度的数据总线具有 2 个系统（DQA 与 DQS），所以分别具有各自的屏蔽信号。

▲　PREX

这也是预充电信号。在读指令及没有字节屏蔽的写指令后，为进行扩展操作而使用 COLX 分组。在该 COLX 分组中最经常使用的就是 PREX 指令，根据该指令，进行 DRAM 内部的预充电。

●6.6.4　直接总线式 DRAM 的操作示例

Direct Rambus DRAM 实际的操作示例（写操作）如图所示。可以看出，是以每 4 个时钟周期的分组为单位传输指令及数据等的。

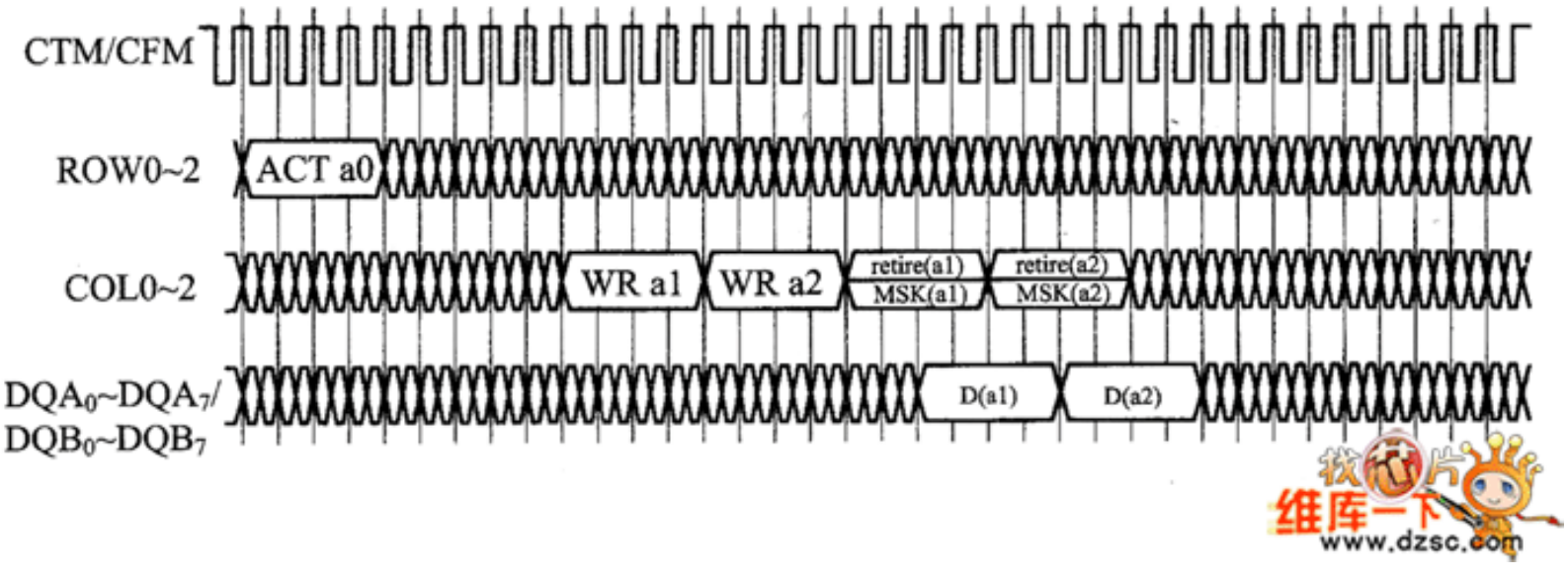


图　Direct Rambus DRAM 的写操作示例

最初，利用 ACT 指令赋予行地址，接着利用 WR 指令发送希望访问的地址及屏蔽数据。此例中是边切换地址边进行传输的，能够进行这样复杂的操作可以说是因为 Direct Rambus 以分组形式传输数据而取得的成果。