

SDRAM 的原理和时序

一、SDRAM 内存模组与基本结构

我们平时看到的 SDRAM 都是以模组形式出现，为什么要做成这种形式呢？这首先要接触到两个概念：物理 Bank 与芯片位宽。



1、物理 Bank

传统内存系统为了保证 CPU 的正常工作，必须一次传输完 CPU 在一个传输周期内所需要的数据。而 CPU 在一个传输周期能接受的数据容量就是 CPU 数据总线的位宽，单位是 bit

（位）。当时控制内存与 CPU 之间数据交换的北桥芯片也因此将内存总线的数据位宽等同于 CPU 数据总线的位宽，而这个位宽就称之为物理 Bank（Physical Bank，下文简称 P-Bank）的位宽。所以，那时的内存必须要组织成 P-Bank 来与 CPU 打交道。资格稍老的玩家应该还记得 Pentium 刚上市时，需要两条 72pin 的 SIMM 才能启动，因为一条 72pin -SIMM 只能提供 32bit 的位宽，不能满足 Pentium 的 64bit 数据总线的需要。直到 168pin-SDRAM DIMM 上市后，才可以使用一条内存开机。

不过要强调一点，P-Bank 是 SDRAM 及以前传统内存家族的特有概念，RDRAM 中将以通道（Channel）取代，而对于像 Intel E7500 那样的并发式多通道 DDR 系统，传统的 P-Bank 概念也不适用。

2、芯片位宽

上文已经讲到 SDRAM 内存系统必须要组成一个 P-Bank 的位宽，才能使 CPU 正常工作，那么这个 P-Bank 位宽怎么得到呢？这就涉及到了内存芯片的结构。

每个内存芯片也有自己的位宽，即每个传输周期能提供的数据量。理论上，完全可以做出一个位宽为 64bit 的芯片来满足 P-Bank 的需要，但这对技术的要求很高，在成本和实用性方面也都处于劣势。所以芯片的位宽一般都较小。台式机市场所用的 SDRAM 芯片位宽最高也就是 16bit，常见的则是 8bit。这样，为了组成 P-Bank 所需的位宽，就需要多颗芯片并联工作。对于 16bit 芯片，需要 4 颗（ $4 \times 16\text{bit} = 64\text{bit}$ ）。对于 8bit 芯片，则需要 8 颗了。

以上就是芯片位宽、芯片数量与 P-Bank 的关系。P-Bank 其实就是一组内存芯片的集合，这个集合的容量不限，但这个集合的总位宽必须与 CPU 数据位宽相符。随着计算机应用的发展，

一个系统只有一个 P-Bank 已经不能满足容量的需要。所以，芯片组开始可以支持多个 P-Bank，一次选择一个 P-Bank 工作，这就有了芯片组支持多少（物理）Bank 的说法。而在 Intel 的定义中，则称 P-Bank 为行（Row），比如 845G 芯片组支持 4 个行，也就是说它支持 4 个 P-Bank。另外，在一些文档中，也把 P-Bank 称为 Rank（列）。

提示：SDRAM、SIMM、DIMM、pin 的含义

SDRAM: Synchronous Dynamic Random Access Memory, 同步动态随机存储器。同步是指其时钟频率与 CPU 前端总线的系统时钟频率相同，并且内部的命令的发送与数据的传输都以它为基准；动态是指存储阵列需要不断的刷新来保证数据不丢失；随机是指数据不是线性依次存储，而是自由指定地址进行数据的读写。

pin: 模组或芯片与外部电路联接用的金属引脚，而模组的 pin 就是常说的“金手指”。

SIMM: Single In-line Memory Module, 单列内存模组。内存模组就是我们常说的内存条，所谓单列是指模组电路板与主板插槽的接口只有一列引脚（虽然两侧都有金手指）。

DIMM: Double In-line Memory Module, 双列内存模组。所谓双列是指模组电路板与主板插槽的接口有两列引脚，模组电路板两侧的金手指对应一列引脚。

回到开头的话题，DIMM 是 SDRAM 集合形式的最终体现，每个 DIMM 至少包含一个 P-Bank 的芯片集合。在目前的 DIMM 标准中，每个模组最多可以包含两个 P-Bank 的内存芯片集合，虽然理论上完全可以在一个 DIMM 上支持多个 P-Bank，比如 SDRAM DIMM 就有 4 个芯片选择信号，理论上可以控制 4 个 P-Bank 的芯片集合。只是由于某种原因而没有这么去做。比如设计难度、制造成本、芯片组的配合等。至于 DIMM 的面数与 P-Bank 数量的关系，在 2001 年 2 月的专题中已经明确了，面数 \neq P-Bank 数，只有在知道芯片位宽的情况下，才能确定 P-Bank 的数量，大度 256MB 内存就是明显一例，而这种情况在 Registered 模组中非常普遍。有关内存模组的设计，将在后面的相关章节中继续探讨。

二、SDRAM 内存芯片的内部结构

1、逻辑 Bank 与芯片位宽

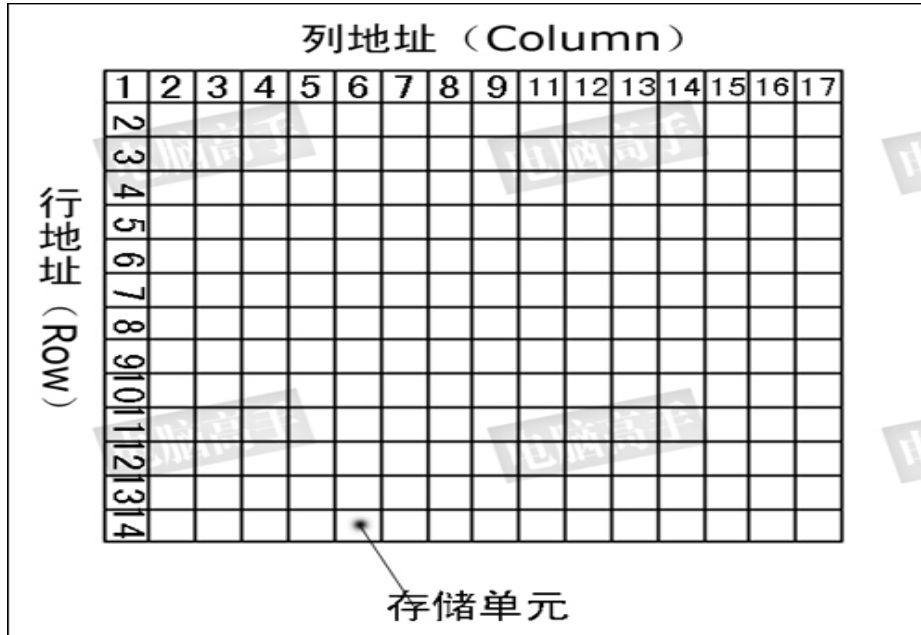
讲完 SDRAM 的外在形式，就该深入了解 SDRAM 的内部结构了。这里主要的概念就是逻辑 Bank。简单地说，SDRAM 的内部是一个存储阵列。因为如果是管道式存储（就如排队买票），就很难做到随机访问了。

阵列就如同表格一样，将数据“填”进去，你可以想象成一张表格。和表格的检索原理一样，先指定一个行（Row），再指定一个列（Column），我们就可以准确地找到所需要的单元格，这就是内存芯片寻址的基本原理。对于内存，这个单元格可称为存储单元，那么这个表格（存储阵列）叫什么呢？它就是逻辑 Bank（Logical Bank，下文简称 L-Bank）。

由于技术、成本等原因，不可能只做一个全容量的 L-Bank，而且最重要的是，由于 SDRAM 的工作原理限制，单一的 L-Bank 将会造成非常严重的寻址冲突，大幅降低内存效率（在后文中将详细讲述）。所以人们在 SDRAM 内部分割成多个 L-Bank，较早以前是两个，目前基本都是 4 个，这也是 SDRAM 规范中的最高 L-Bank 数量。到了 RDRAM 则最多达到了 32 个，在最新 DDR-II 的标准中，L-Bank 的数量也提高到了 8 个。

这样，在进行寻址时就要先确定是哪个 L-Bank，然后再在这个选定的 L-Bank 中选择相应的行与列进行寻址。可见对内存的访问，一次只能是一个 L-Bank 工作，而每次与北桥交换的数据就是 L-Bank 存储阵列中一个“存储单元”的容量。在某些厂商的表述中，将 L-Bank 中的存储单元称为 Word（此处代表位的集合而不是字节的集合）。

从前文可知，SDRAM 内存芯片一次传输率的数据量就是芯片位宽，那么这个存储单元的容量就是芯片的位宽（也是 L-Bank 的位宽），但要注意，这种关系也仅对 SDRAM 有效，原因将在下文中说明。



2、内存芯片的容量

现在我们应该清楚内存芯片的基本组织结构了。那么内存的容量怎么计算呢？显然，内存芯片的容量就是所有 L-Bank 中的存储单元的容量总和。计算有多少个存储单元和计算表格中的单元数量的方法一样：

存储单元数量=行数×列数（得到一个 L-Bank 的存储单元数量）×L-Bank 的数量

在很多内存产品介绍文档中，都会用 $M \times W$ 的方式来表示芯片的容量（或者说是芯片的规格/组织结构）。M 是该芯片中存储单元的总数，单位是兆（英文简写 M，精确值是 1048576，而不是 1000000），W 代表每个存储单元的容量，也就是 SDRAM 芯片的位宽（Width），单位是 bit。计算出来的芯片容量也是以 bit 为单位，但用户可以采用除以 8 的方法换算为字节（Byte）。比如 $8M \times 8$ ，这是一个 8bit 位宽芯片，有 8M 个存储单元，总容量是 64Mbit（8MB）。

提示：bit、Byte、Word 的关系

bit: 位。二进制数中，一个 0 或 1 就是一个 bit。

Byte: 字节。8 个 bit 为一个字节，这与 ASCII (American Standard Code for Information Interchange, 美国标准信息交换代码) 的规定有关，ASCII 用 8 位二进制数来表示 256 个信息代码，所以 8 个 bit 定义为一个字节。

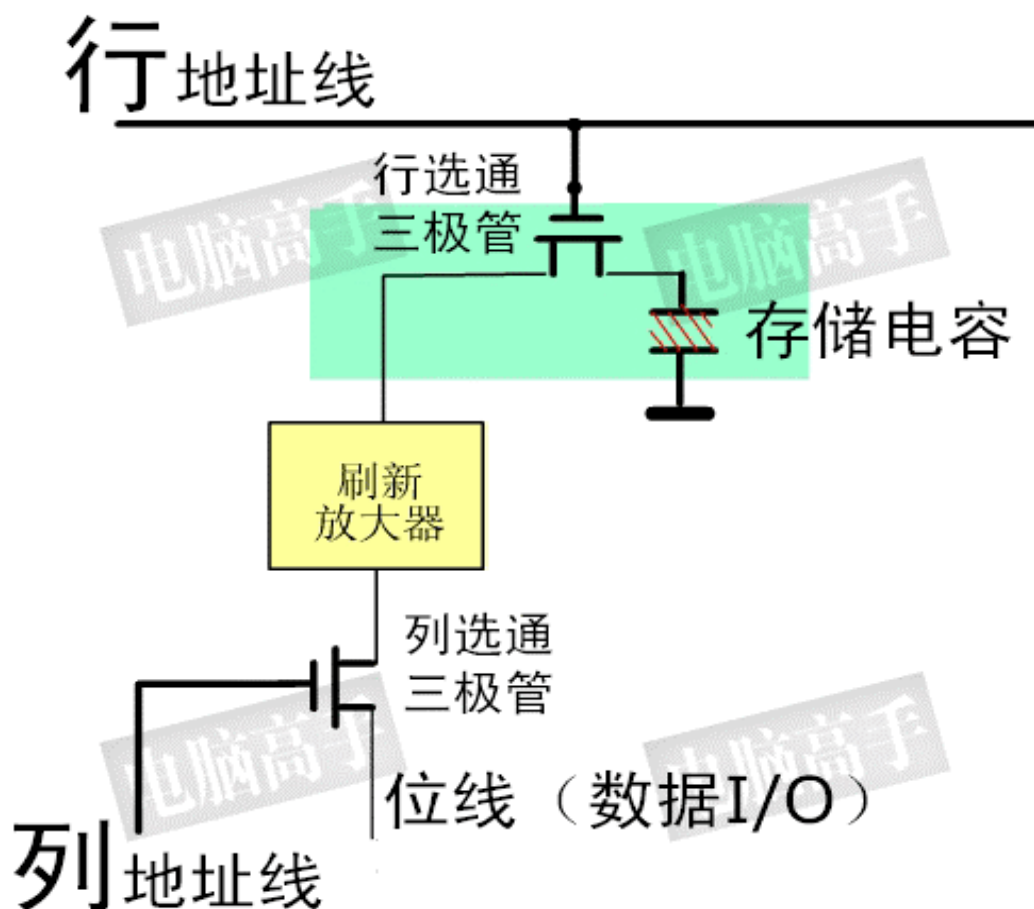
Word: 字：两个字节为一个字，这里的 Word 不是指 L-Bank 中存储单元，此外还有双字 (DWords, Double Words, 4 个字节) 和四字 (QWord, Quad Words, 8 个字节) 的表示法。目前一个 P-Bank 的位宽就是 QWord，这在很多 CPU 与芯片组的介绍中经常用到。

不过， $M \times W$ 是最简单的表示方法。下图则是某公司对自己内存芯片的容量表示方法，这可以说是最正规的形式之一。

2,097,152-WORDS × 4BANKS × 16-BITS
4,194,304-WORDS × 4BANKS × 8-BITS
8,388,608-WORDS × 4BANKS × 4-BITS

提示：DRAM 的存储原理

L-Bank 中的存储单元是基本的存储单位，它的容量是若干 Bit(对于 SDRAM 而言，就是芯片的位宽)，而每个 Bit 则是存放于一个单独的存储体中。这些存储体就是内存中最小的存储单元。你可以用硬盘操作中的簇与扇区的关系来理解内存中的存储形式。扇区是硬盘中的最小存储单元（相当于内存中的存储体），而每个簇则包含有多个扇区（相当于 L-Bank 中的存储单元），数据的交换都是一个簇为单位进行（一次传输一个存储单元的数据）。



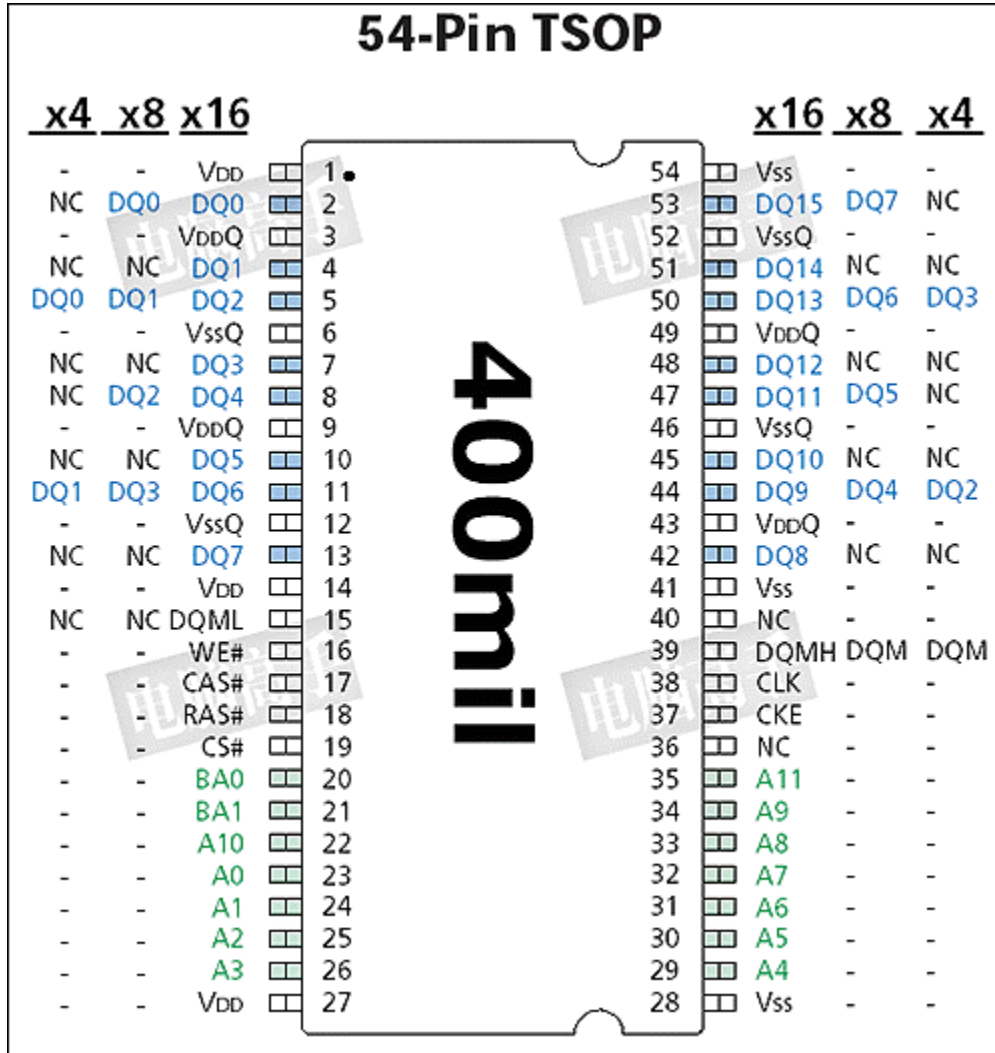
DRAM 的存储原理示意图：行选与列选信号将使存储电容与外界间的传输电路导通，从而可进行放电（读取）与充电（写入）。另外，图中刷新放大器的设计并不固定，目前这一功能被并入读出放大器（Sense Amplifier，简称 S-AMP），具体操作在下文中详细讲述。

我们可以计算一下，结果可以发现这三个规格的容量都是 128Mbits，只是由于位宽的变化引起了存储单元的数量变化。从这个例子就也可以看出，在相同的总容量下，位宽可以采用多种不同的设计。

3、与芯片位宽相关的 DIMM 设计

为什么在相同的总容量下，位宽会有多种不同的设计呢？这主要是为了满足不同领域的需要。现在大家已经知道 P-Bank 的位宽是固定的，也就是说当芯片位宽确定下来后，一个 P-Bank 中芯片的个数也就自然确定了，而前文讲过 P-Bank 对芯片集合的位宽有要求，对芯片集合的容量则没有任何限制。高位宽的芯片可以让 DIMM 的设计简单一些（因为所用的芯片少），但在芯片容量相同时，这种 DIMM 的容量就肯定比不上采用低位宽芯片的模组，因为后者在一个 P-Bank 中可以容纳更多的芯片。比如上文中那个内存芯片容量标识图，容量都是 128Mbit，合 16MB。如果 DIMM 采用双 P-Bank+16bit 芯片设计，那么只能容纳 8 颗芯片，计 128MB。但如果采用 4bit 位宽芯片，则可容纳 32 颗芯片，计 512MB。DIMM 容量前后相差出 4 倍，可见芯片位宽对 DIMM 设计的重要性。因此，8bit 位宽芯片是桌面台式机上容量与成本之间平衡性较好的选择，所以在市场上也最为普及，而高于 16bit 位宽的芯片一般用在需要更大位宽的场所，如显卡等，至于 4bit 位宽芯片很明显非常适用于大容量内存应用领域，基本不会在标准的 Unbuffered 模组设计中出现。

三、SDRAM 的引脚与封装



图注：128Mbit 芯片不同位宽的引脚图（NC 代表未使用，-表示与内侧位宽设计相同）

引脚代号	定义	引脚代号	定义
Vdd/ VddQ	工作/DQ 电压	CAS#	列地址选通脉冲
Vss/VssQ	相应电压的接地	RAS#	行地址选通脉冲
DQn	数据 I/O 线	CK	时钟信号
An	行/列地址线	CKE	时钟有效
DQM	数据掩码	BAn	L-Bank 地址线
CS#	片选	WE#	写允许

注：#表示低电平有效

根据 SDRAM 的官方规范，台式机上所用的 SDRAM 在不同容量下的各种位宽封装标准如下：

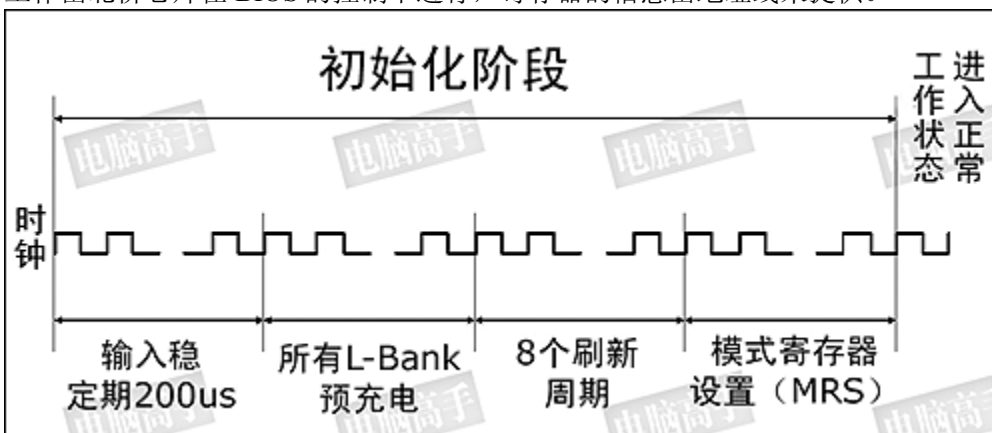
容量 \ 位宽	位宽		
	4bit	8bit	16bit
16Mbit	44pin TSOP-II		50pin TSOP-II
64/128Mbit	54pin TSOP-II (400mil) ^注		
256/512Mbit			

注：400mil（400 千分之一英寸）是指芯片封装的宽度，约合 10.16 毫米

四、SDRAM 的内部基本操作与工作时序

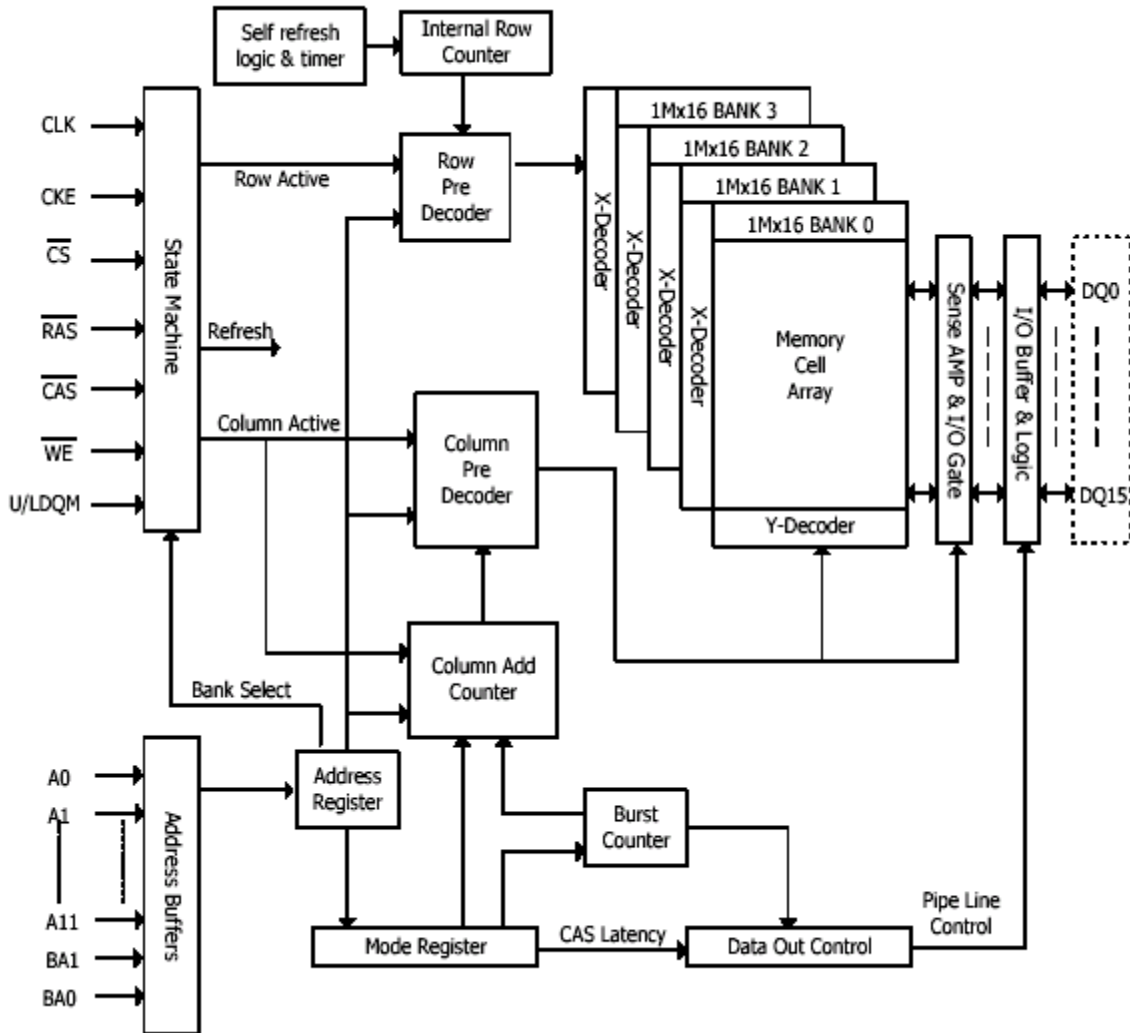
1、芯片初始化

在 SDRAM 芯片内部还有一个逻辑控制单元，并且有一个模式寄存器为其提供控制参数。因此，每次开机时 SDRAM 都要先对这个控制逻辑核心进行初始化。有关预充电和刷新的含义在下文有讲述，关键的阶段就在于模式寄存器（MR，Mode Register）的设置，简称 MRS，这一工作由北桥芯片在 BIOS 的控制下进行，寄存器的信息由地址线来提供。



SDRAM 在开机时的初始化过程

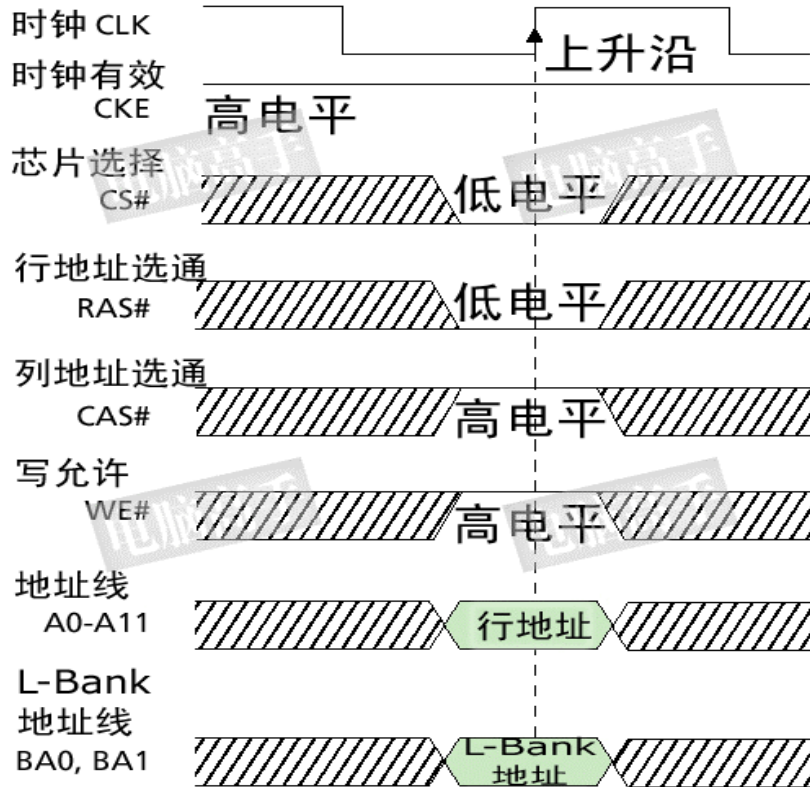
SDRAM 模式寄存器所控制的操作参数：地址线提供不同的 0/1 信号来获得不同的参数。在设置到 MR 之后，就开始了进入正常的工作状态。



64Mbit (4M×16bit) SDRAM 内部结构图

2、行有效

初始化完成后，要想对一个 L-Bank 中的阵列进行寻址，首先就要确定行（Row），使之处于活动状态（Active），然后再确定列。虽然之前要进行片选和 L-Bank 的定址，但它们与行有效可以同时进行。



行有效时序图

在 CS#、L-Bank 定址的同时，RAS（Row Address Strobe，行地址选通脉冲）也处于有效状态。此时 An 地址线则发送具体的行地址。如图中是 A0-A11，共有 12 个地址线，由于是二进制表示法，所以共有 4096 个行（ $2^{12}=4096$ ），A0-A11 的不同数值就确定了具体的行地址。由于行有效的同时也是相应 L-Bank 有效，所以行有效也可称为 L-Bank 有效。

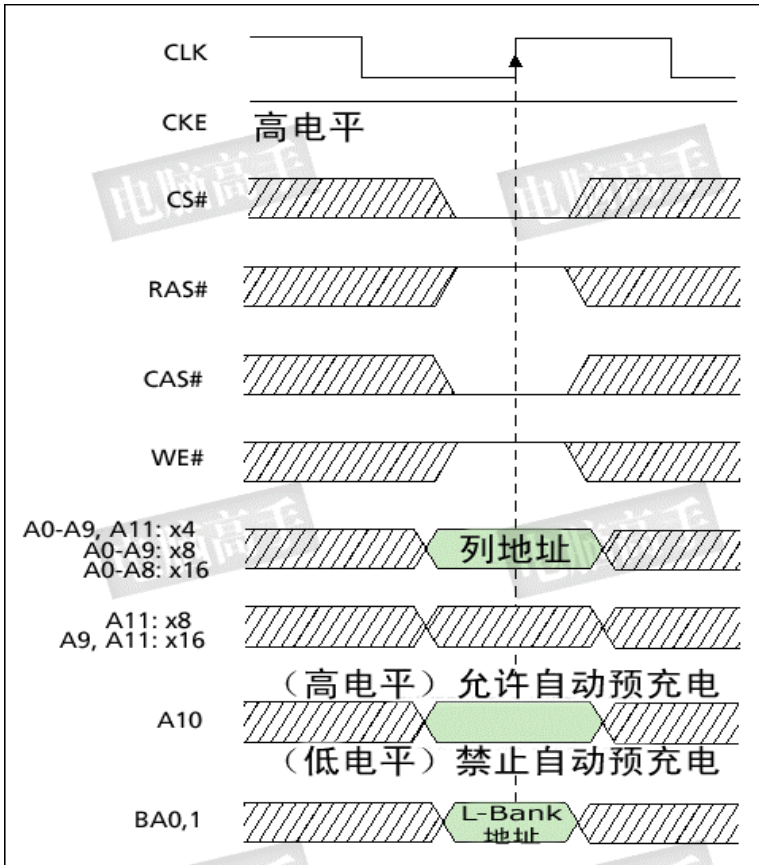
3、列读写

行地址确定之后，就要对列地址进行寻址了。但是，地址线仍然是行地址所用的 A0-A11（本例）。没错，在 SDRAM 中，行地址与列地址线是共用的。不过，读/写的命令是怎么发出的呢？其实没有一个信号是发送读或写的明确命令的，而是通过芯片的可写状态的控制来达到读/写的目的。显然 WE# 信号就是一个关键。WE# 无效时，当然就是读取命令。

命令名（功能）	CS#	RAS#	CAS#	WE#	DQM	地址线	DQs
命令禁止（NOP）	H	X	X	X	X	X	X
无操作（NOP）	L	H	H	H	X	X	X
有效/活动（使指定L-Bank中的指定行有效）	L	L	H	H	X	Bank/行	X
读取（从指定L-Bank中的指定列开始读取数据）	L	H	L	H	L/H	Bank/列	X
写入（从指定L-Bank中的指定列开始写入数据）	L	H	L	L	L/H	Bank/列	有效
突发传输终止	L	H	H	L	X	X	活动
预充电（关闭指定或全部L-Bank中的工作行）	L	L	H	L	X	相应编码	X
自动刷新或自刷新（后者进入自刷新模式）	L	L	L	H	X	X	X
模式寄存器加载（写入）	L	L	L	L	X	操作码	X
写允许/输出允许	-	-	-	-	L	-	有效
写禁止/输出屏蔽（High-Z）	-	-	-	-	H	-	屏蔽

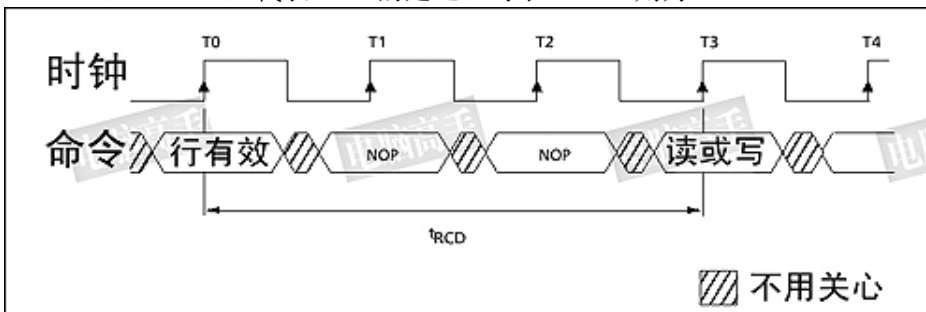
SDRAM 基本操作命令, 通过各种控制/地址信号的组合来完成 (H 代表高电平, L 代表低电平, X 表示高低电平均没有影响)。此表中, 除了自刷新命令外, 所有命令都是默认 CKE 有效。对于自刷新命令, 下文有详解

列寻址信号与读写命令是同时发出的。虽然地址线与行寻址共用, 但 CAS (Column Address Strobe, 列地址选通脉冲) 信号则可以区分开行与列寻址的不同, 配合 A0-A9, A11 (本例) 来确定具体的列地址。



读写操作示意图, 读取命令与列地址一块发出 (当 WE# 为低电平时即为写命令)

然而, 在发送列读写命令时必须要与行有效命令有一个间隔, 这个间隔被定义为 t_{RCD} , 即 RAS to CAS Delay (RAS 至 CAS 延迟), 大家也可以理解为行选通周期, 这应该是根据芯片存储阵列电子元件响应时间 (从一种状态到另一种状态变化的过程) 所制定的延迟。 t_{RCD} 是 SDRAM 的一个重要时序参数, 可以通过主板 BIOS 经过北桥芯片进行调整, 但不能超过厂商的预定范围。广义的 t_{RCD} 以时钟周期 (t_{CK} , Clock Time) 数为单位, 比如 $t_{RCD}=2$, 就代表延迟周期为两个时钟周期, 具体到确切的时间, 则要根据时钟频率而定, 对于 PC100 SDRAM, $t_{RCD}=2$, 代表 20ns 的延迟, 对于 PC133 则为 15ns。



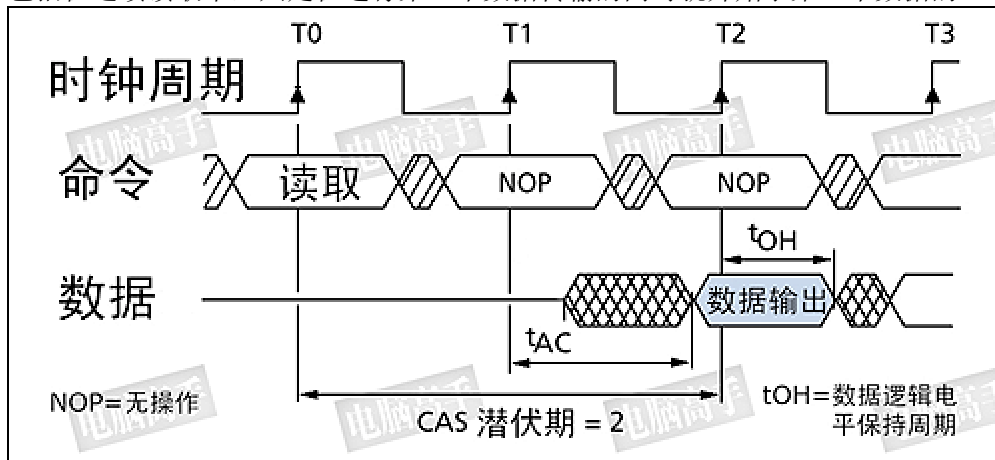
$t_{RCD}=3$ 的时序图

4、数据输出（读）

在选定列地址后，就已经确定了具体的存储单元，剩下的事情就是数据通过数据 I/O 通道（DQ）输出到内存总线上了。但是在 CAS 发出之后，仍要经过一定的时间才能有数据输出，从 CAS 与读取命令发出到第一笔数据输出的这段时间，被定义为 CL（CAS Latency，CAS 潜伏期）。由于 CL 只在读取时出现，所以 CL 又被称为读取潜伏期（RL，Read Latency）。CL 的单位与 tRCD 一样，为时钟周期数，具体耗时由时钟频率决定。

不过，CAS 并不是在经过 CL 周期之后才送达存储单元。实际上 CAS 与 RAS 一样是瞬间到达的，但 CAS 的响应时间要更快一些。为什么呢？假设芯片位宽为 n 个 bit，列数为 c ，那么一个行地址要选通 $n \times c$ 个存储体，而一个列地址只需选通 n 个存储体。但存储体中晶体管的反应时间仍会造成数据不可能与 CAS 在同一上升沿触发，肯定要延后至少一个时钟周期。

由于芯片体积的原因，存储单元中的电容容量很小，所以信号要经过放大来保证其有效的识别性，这个放大/驱动工作由 S-AMP 负责，一个存储体对应一个 S-AMP 通道。但它要有一个准备时间才能保证信号的发送强度（事前还要进行电压比较以进行逻辑电平的判断），因此从数据 I/O 总线上有数据输出之前的一个时钟上升沿开始，数据即已传向 S-AMP，也就是说此时数据已经被触发，经过一定的驱动时间最终传向数据 I/O 总线进行输出，这段时间我们称之为 tAC（Access Time from CLK，时钟触发后的访问时间）。tAC 的单位是 ns，对于不同的频率各有不同的明确规定，但必须要小于一个时钟周期，否则会因访问时过长而使效率降低。比如 PC133 的时钟周期为 7.5ns，tAC 则是 5.4ns。需要强调的是，每个数据在读取时都有 tAC，包括在连续读取中，只是在进行第一个数据传输的同时就开始了第二个数据的 tAC。



CL=2 与 tAC 示意图

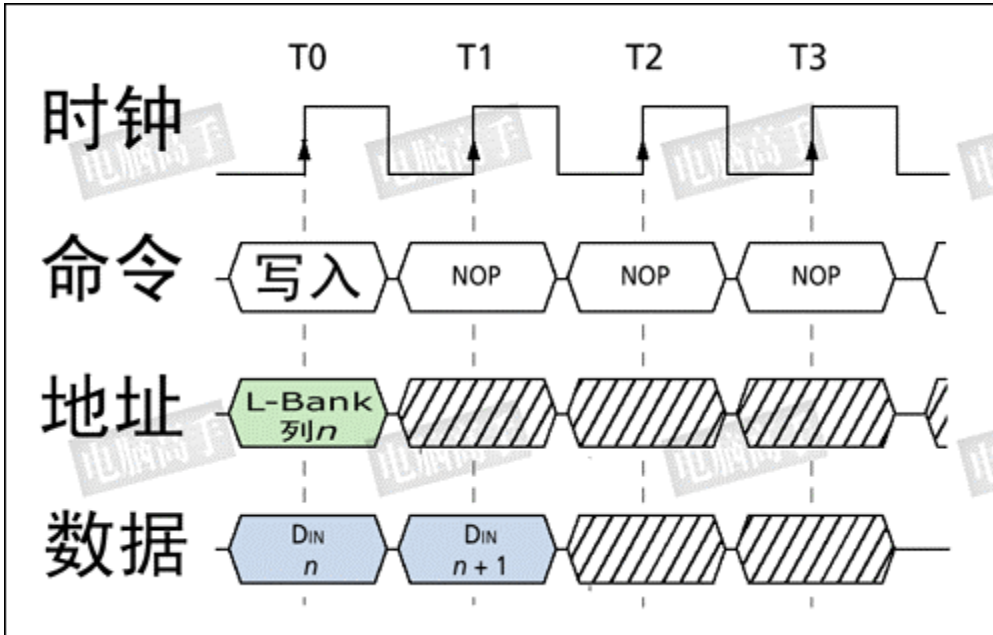
不过，从存储体的结构图上可以看出，原本逻辑状态为 1 的电容在读取操作后，会因放电而变为逻辑 0。所以，以前的 DRAM 为了在关闭当前行时保证数据的可靠性，要对存储体中原有的信息进行重写，这个任务由数据所经过的刷新放大器来完成，它根据逻辑电平状态，将数据进行重写（逻辑 0 时就不重写），由于这个操作与数据的输出是同步进行互不冲突，所以不会产生新的重写延迟。后来通过技术的改良，刷新放大器被取消，其功能由 S-AMP 取代，因为在读取时它会保持数据的逻辑状态，起到了一个 Cache 的作用，再次读取时由它直接发送即可，不用再进行新的寻址输出，此时数据重写操作则可在预充电阶段完成。

5、数据输入（写）

数据写入的操作也是在 tRCD 之后进行，但此时没有了 CL（记住，CL 只出现在读取操作中），行寻址与列寻址的时序图和上文一样，只是在列寻址时，WE# 为有效状态。

从图中可见，由于数据信号由控制端发出，输入时芯片无需做任何调校，只需直接传到数据输入寄存器中，然后再由写入驱动器进行对存储电容的充电操作，因此数据可以与 CAS 同时发送，也就是说写入延迟为 0。不过，数据并不是即时地写入存储电容，因为选通三极管（就如读取时一样）与电容的充电必须要有一段时间，所以数据的真正写入需要一定的周期。为了保

证数据的可靠写入，都会留出足够的 写入/校正时间（ t_{WR} , Write Recovery Time），这个操作也被称作写回（Write Back）。 t_{WR} 至少占用一个时钟周期或再多一点（时钟频率越高， t_{WR} 占用周期越多），有关它的影响将在下文进一步讲述。

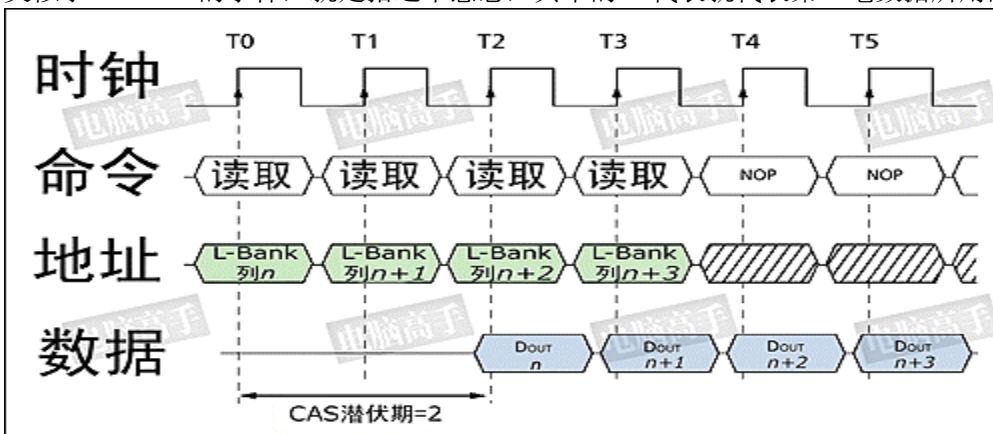


数据写入的时序图

6、突发长度

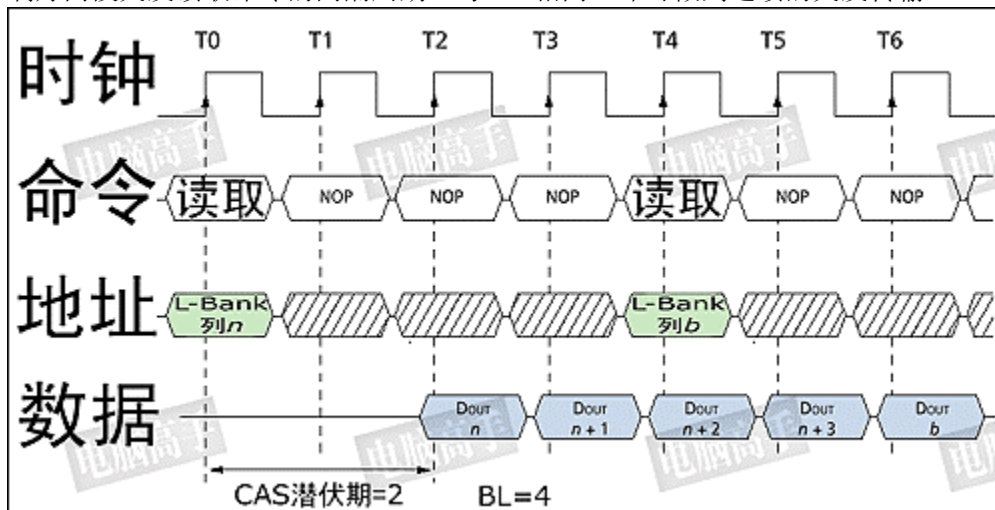
突发（Burst）是指在同一行中相邻的存储单元连续进行数据传输的方式，连续传输所涉及到的存储单元（列）的数量就是突发长度（Burst Lengths, 简称 BL）。

在目前，由于内存控制器一次读/写 P-Bank 位宽的数据，也就是 8 个字节，但是在现实中小于 8 个字节的数据很少见，所以一般都要经过多个周期进行数据的传输。上文讲到的读/写操作，都是一次对一个存储单元进行寻址，如果要连续读/写就还要对当前存储单元的下一个单元进行寻址，也就是要不断的发送列地址与读/写命令（行地址不变，所以不用再对行寻址）。虽然由于读/写延迟相同可以让数据的传输在 I/O 端是连续的，但它占用了大量的内存控制资源，在数据进行连续传输时无法输入新的命令，效率很低（早期的 FP E/EDO 内存就是以这种方式进行连续的数据传输）。为此，人们开发了突发传输技术，只要指定起始列地址与突发长度，内存就会依次地自动对后面相应数量的存储单元进行读/写操作而不再需要控制器连续地提供列地址。这样，除了第一笔数据的传输需要若干个周期（主要是之前的延迟，一般的是 $t_{RCD}+CL$ ）外，其后每个数据只需一个周期的即可获得。在很多北桥芯片的介绍中都有类似于 X-1-1-1 的字样，就是指这个意思，其中的 X 代表就代表第一笔数据所用的周期数。



非突发连续读取模式：不采用突发传输而是依次单独寻址，此时可等效于 BL=1。虽然可以让数据是连续的传输，但每次都要发送列地址与命令信息，控制资源占用极大。

突发连续读取模式：只要指定起始列地址与突发长度，寻址与数据的读取自动进行，而只要控制好两段突发读取命令的间隔周期（与 BL 相同）即可做到连续的突发传输。



至于 BL 的数值，也是不能随便设或在数据进行传输前临时决定。在上文讲到的初始化过程中的 MRS 阶段就要对 BL 进行设置。目前可用的选项是 1、2、4、8、全页（Full Page），常见的设定是 4 和 8。顺便说一下，BL 能否更改与北桥芯片的设计有很大关系，不是每个北桥都能像调整 CL 那样来调整 BL。某些芯片组的 BL 是定死而不可改的，比如 Intel 芯片组的 BL 基本都为 4，所以在相应的主板 BIOS 中也就不会有 BL 的设置选项。而由于目前的 SDRAM 系统的数据传输是以 64bit/周期进行，所以在一些 BIOS 也把 BL 用 QWord（4 字，即 64bit）来表示。如 4QWord 就是 BL=4。

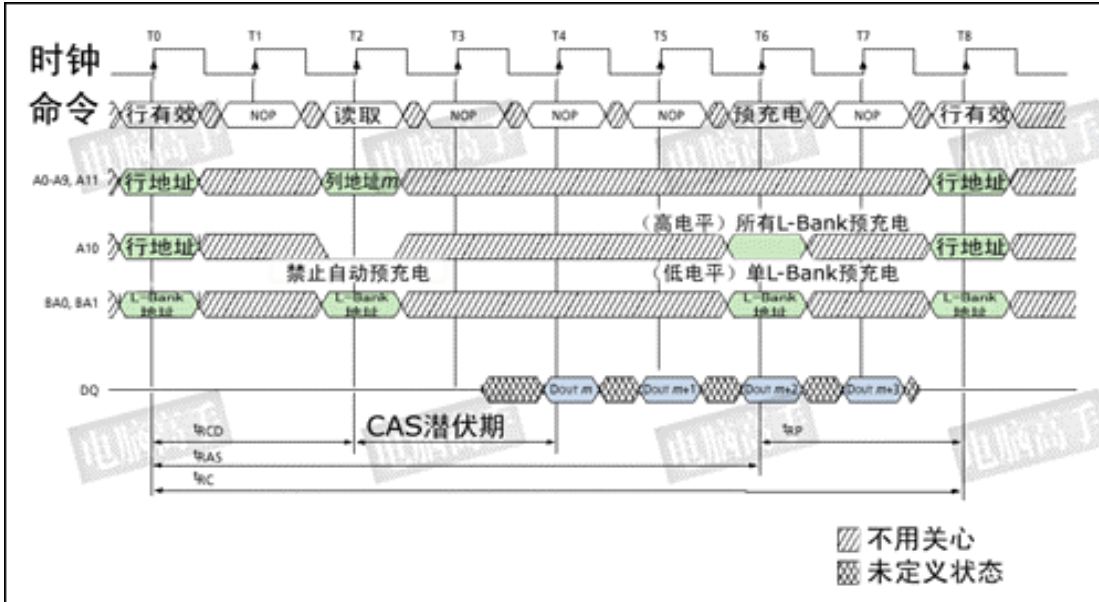
Full Page（全页）突发传输是指 L-Bank 里的一行中所有存储单元从头到尾进行连续传输。

另外，在 MRS 阶段除了要设定 BL 数值之外，还要具体确定读/写操作的模式以及突发传输的模式。突发读/突发写，表示读与写操作都是突发传输的，每次读/写操作持续 BL 所设定的长度，这也是常规的设定。突发读/单一写，表示读操作是突发传输，写操作则只是一个一个单独进行。突发传输模式代表着突发周期内所涉及到的存储单元的传输顺序。顺序传输是指从起始单元开始顺序读取。假如 BL=4，起始单元编号是 n，顺序就是 n、n+1、n+2、n+3。交错传输就是打乱正常的顺序进行数据传输（比如第一个进行传输的单元是 n，而第二个进行传输的单元是 n+2 而不是 n+1），至于交错的规则在 SDRAM 规范中有详细的定义表，但在这此出于必要性与篇幅的考虑就不列出了。

7、预充电

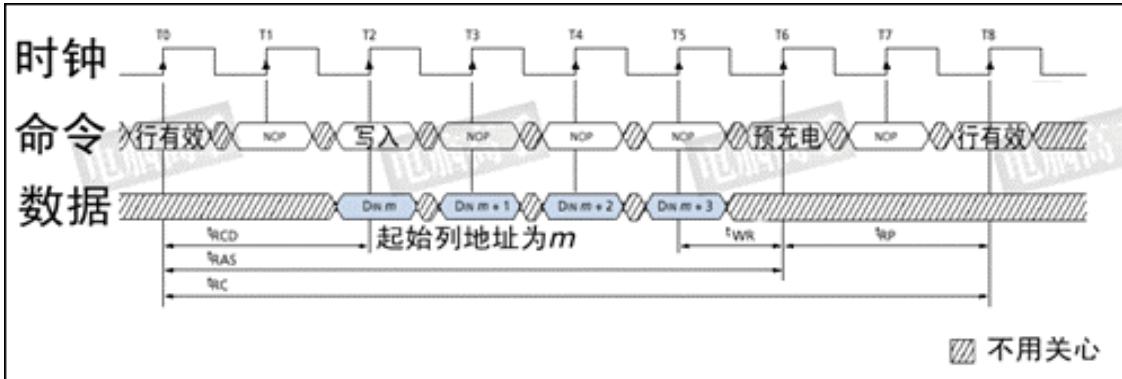
由于 SDRAM 的寻址具有独占性，所以在进行完读写操作后，如果要对同一 L-Bank 的另一行进行寻址，就要将原来有效（工作）的行关闭，重新发送行/列地址。L-Bank 关闭现有工作行，准备打开新行的操作就是预充电（Precharge）。预充电可以通过命令控制，也可以通过辅助设定让芯片在每次读写操作之后自动进行预充电。实际上，预充电是一种对工作行中所有存储体进行数据重写，并对行地址进行复位，同时释放 S-AMP（重新加入比较电压，一般是电容电压的 1/2，以帮助判断读取数据的逻辑电平，因为 S-AMP 是通过一个参考电压与存储体位线电压的比较来判断逻辑值的），以准备新行的工作。具体而言，就是将 S-AMP 中的数据回写，即使是没有工作过的存储体也会因行选通而使存储电容受到干扰，所以也需要 S-AMP 进行读后重写。此时，电容的电量（或者说其产生的电压）将是判断逻辑状态的依据（读取时也需要），为此要设定一个临界值，一般为电容电量的 1/2，超过它的为逻辑 1，进行重写，否则为逻辑 0，不进行重写（等于放电）。为此，现在基本都将电容的另一端接入一个指定的电压（即 1/2 电容电压），而不是接地，以帮助重写时的比较与判断。

现在我们再回过头看看读写操作时的命令时序图，从中可以发现地址线 A10 控制着是否进行在读写之后当前 L-Bank 自动进行预充电，这就是上文所说的“辅助设定”。而在单独的预充电命令中，A10 则控制着是对指定的 L-Bank 还是所有的 L-Bank（当有多个 L-Bank 处于有效/活动状态时）进行预充电，前者需要提供 L-Bank 的地址，后者只需将 A10 信号置于高电平。在发出预充电命令之后，要经过一段时间才能允许发送 RAS 行有效命令打开新的工作行，这个间隔被称为 tRP（Precharge command Period，预充电有效周期）。和 tRCD、CL 一样，tRP 的单位也是时钟周期数，具体值视时钟频率而定。



读取时预充电时序图：图中设定：CL=2、BL=4、tRP=2。自动预充电时的开始时间与此图一样，只是没有了单独的预充电命令，并在发出读取命令时，A10 地址线要设为高电平（允许自动预充电）。可见控制好预充电启动时间很重要，它可以在读取操作结束后立刻进入新行的寻址，保证运行效率。

有些文章强调由于写回操作而使读/写操作后都有一定的延迟，但从本文的介绍中写可以看出，即使是读后立即重写的操作，由于是与数据输出同步进行，并不存在延迟。只有在写操作后进行其他的操作时，才会有这方面的影响。写操作虽然是 0 延迟进行，但每笔数据的真正写入则需要一个足够的周期来保证，这段时间就是写回周期（tWR）。所以预充电不能与写操作同时进行，必须要在 tWR 之后才能发出预充电命令，以确保数据的可靠写入，否则重写的的数据可能是错的，这就造成了写回延迟。



数据写入时预充电操作时序图：注意其中的 tWR 参数，由于它的存在，使预充电操作延后，从而造成写回延迟。

8、刷新

之所以称为 DRAM，就是因为它要不断进行刷新（Refresh）才能保留住数据，因此它是 DRAM 最重要的操作。

刷新操作与预充电中重写的操作一样，都是用 S-AMP 先读再写。但为什么有预充电操作还要进行刷新呢？因为预充电是对一个或所有 L-Bank 中的工作行操作，并且是不定期的，而刷新则是有固定的周期，依次对所有行进行操作，以保留那些久久没经历重写的存储体中的数据。但与所有 L-Bank 预充电不同的是，这里的行是指所有 L-Bank 中地址相同的行，而预充电中各 L-Bank 中的工作行地址并不是一定是相同的。

那么要隔多长时间重复一次刷新呢？目前公认的标准是，存储体中电容的数据有效保存期上限是 64ms（毫秒，1/1000 秒），也就是说每一行刷新的循环周期是 64ms。这样刷新速度就是：行数量/64ms。我们在看内存规格时，经常会看到 4096 Refresh Cycles/64ms 或 8192 Refresh Cycles/64ms 的标识，这里的 4096 与 8192 就代表这个芯片中每个 L-Bank 的行数。刷新命令一次对一行有效，发送间隔也是随总行数而变化，4096 行时为 15.625 μ s（微秒，1/1000 毫秒），8192 行时就为 7.8125 μ s。

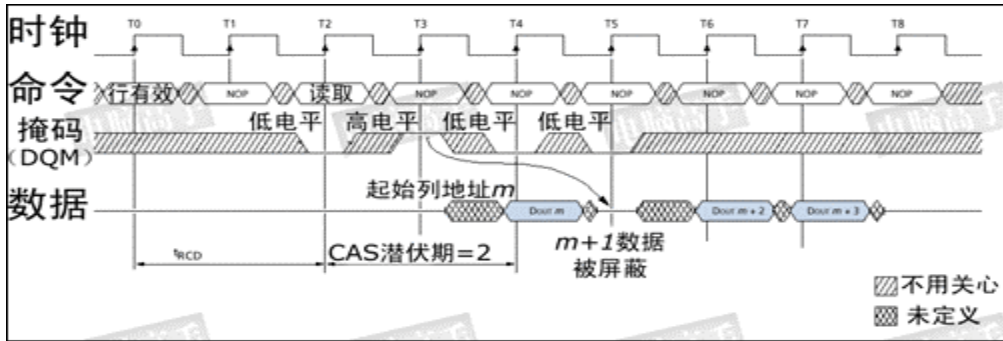
刷新操作分为两种：Auto Refresh，简称 AR 与 Self Refresh，简称 SR。不论是何种刷新方式，都不需要外部提供行地址信息，因为这是一个内部的自动操作。对于 AR，SDRAM 内部有一个行地址生成器（也称刷新计数器）用来自动的依次生成行地址。由于刷新是针对一行中的所有存储体进行，所以无需列寻址，或者说 CAS 在 RAS 之前有效。所以，AR 又称 CBR（CAS Before RAS，列提前于行定位）式刷新。由于刷新涉及到所有 L-Bank，因此在刷新过程中，所有 L-Bank 都停止工作，而每次刷新所占用的时间为 9 个时钟周期（PC133 标准），之后就进入正常的工作状态，也就是说在这 9 个时钟期间内，所有工作指令只能等待而无法执行。64ms 之后则再次对同一行进行刷新，如此周而复始进行循环刷新。显然，刷新操作肯定会对 SDRAM 的性能造成影响，但这是没办法的事情，也是 DRAM 相对于 SRAM（静态内存，无需刷新仍能保留数据）取得成本优势的同时所付出的代价。

SR 则主要用于休眠模式低功耗状态下的数据保存，这方面最著名的应用就是 STR（Suspend to RAM，休眠挂起于内存）。在发出 AR 命令时，将 CKE 置于无效状态，就进入了 SR 模式，此时不再依靠系统时钟工作，而是根据内部的时钟进行刷新操作。在 SR 期间除了 CKE 之外的所有外部信号都是无效的（无需外部提供刷新指令），只有重新使 CKE 有效才能退出自刷新模式并进入正常操作状态。

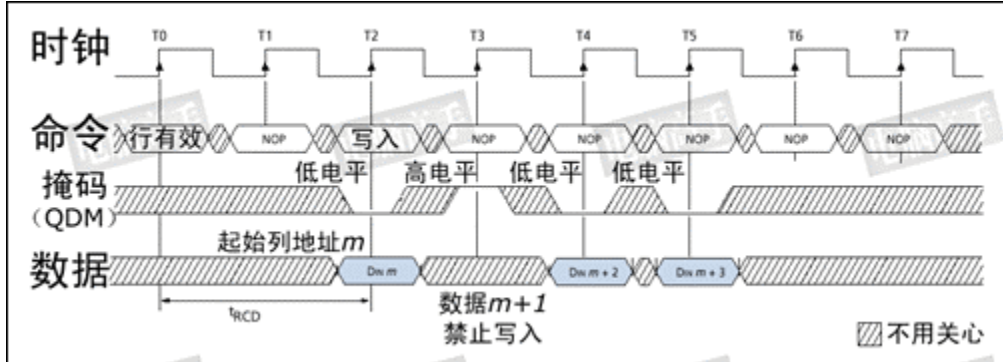
9、数据掩码

在讲述读/写操作时，我们谈到了突发长度。如果 BL=4，那么也就是说一次就传送 4×64bit 的数据。但是，如果其中的第二笔数据是不需要的，怎么办？还都传输吗？为了屏蔽不需要的数据，人们采用了数据掩码（Data I/O Mask，简称 DQM）技术。通过 DQM，内存可以控制 I/O 端口取消哪些输出或输入的数据。这里需要强调的是，在读取时，被屏蔽的数据仍然会从存储体传出，只是在“掩码逻辑单元”处被屏蔽。DQM 由北桥控制，为了精确屏蔽一个 P-Bank 位宽中的每个字节，每个 DIMM 有 8 个 DQM 信号线，每个信号针对一个字节。这样，对于 4bit 位宽芯片，两个芯片共用一个 DQM 信号线，对于 8bit 位宽芯片，一个芯片占用一个 DQM 信号，而对于 16bit 位宽芯片，则需要两个 DQM 引脚。

SDRAM 官方规定，在读取时 DQM 发出两个时钟周期后生效，而在写入时，DQM 与写入命令一样是立即生效。



读取时数据掩码操作，DQM 在两个周期后生效，突发周期的第二笔数据被取消。



写入时数据掩码操作，DQM 立即生效，突发周期的第二笔数据被取消。