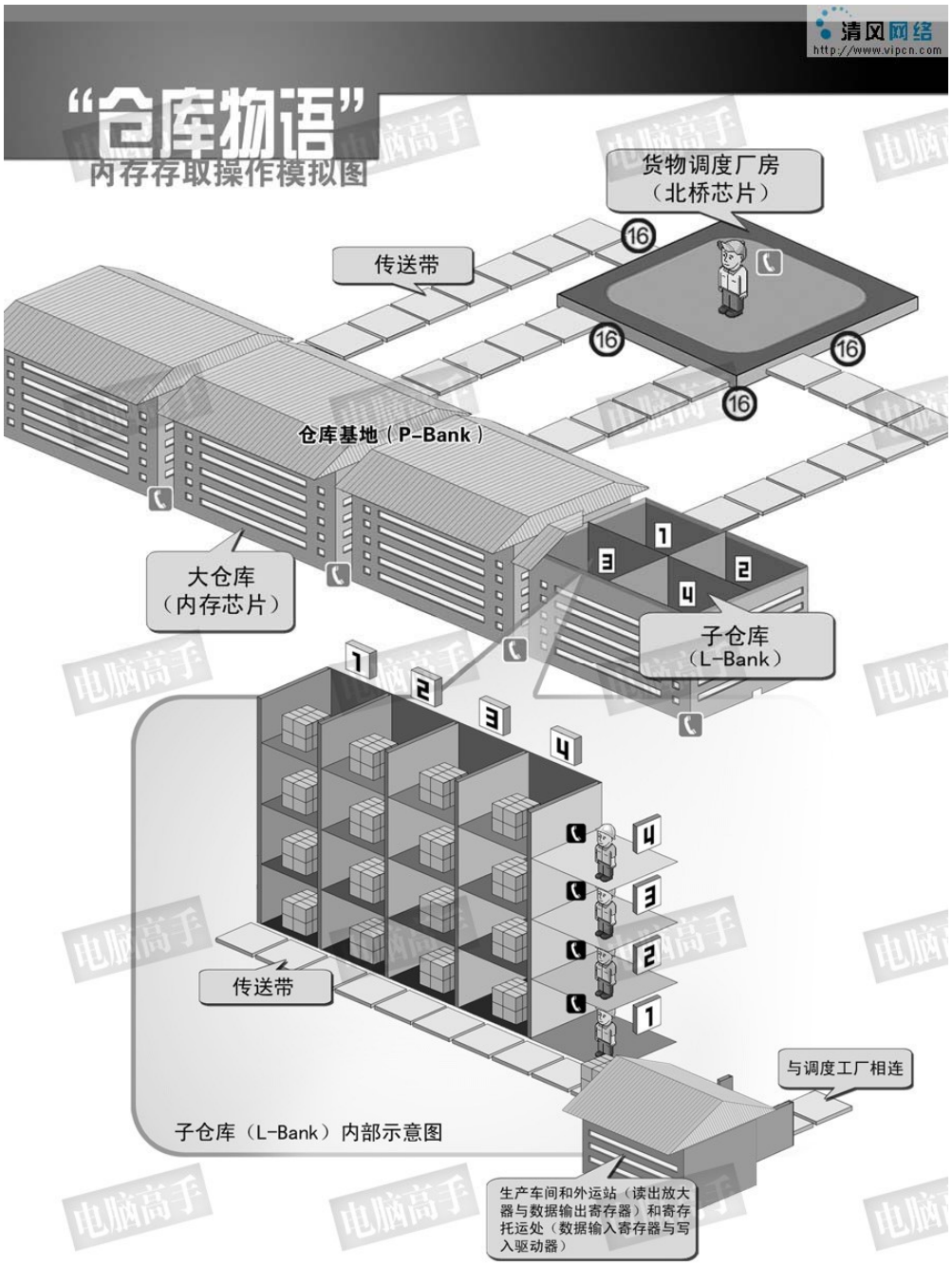
仓库物语

货物基地（主板）连接着物资（数据）的供求方。基地的货物调度厂房（北桥芯片）掌管着若干个用于临时供货/生产与存储的仓库基地（P-Bank），它们通常隶属于某一仓储集团（DIMM），这种基地与调度厂房之间必须由 64 条传送带联系着（P-Bank 位宽），每条传送带一次只能运送一个标准的货物（1bit 数据），而且一次至少要传送 64 个标准货物，这是它们之间的约定，仓库基地必须满足。



上图就是这样的一个仓库基地（P-Bank），它由 4 个大仓库（内存芯片）组成，它们的规模都相当大，每个大仓库为基地提供 16 条传送带（芯片位宽为 16bit），总共加起来刚好就是 64 条。每个大仓库里都有四个规模和结构相同的子仓库（L-Bank），它们都被统一编了号。而子仓库中有很多层（行），每层里又有很多的储藏间（列），每个储藏间可以放置 16 个标准货物，虽然子仓库的规模很大，但每一层和每一个房间也都编好了号，而且每一层都有一个搬运工在值班。

为了与外界联系方便，仓储集团与调度室设置了专线电话，和一个国家一样，每个仓库基地有一个区号（片选），另外还有四个子仓库号码（L-Bank 地址），是所有大仓库共享的，一个号码对应所有大仓库中编号相同的子仓库。而专线电话的数量也是四个，这样可保证与某个子仓库通话时不会妨碍给其他子仓库打电话。在子仓库的每层则设立分机给搬运工使用。子仓库的楼下就是传送带，找到货物把它扔到上面。但每个大仓库只有一个传送带，也就是说同一时间内只能有一个子仓库在工作。每个子仓库都有一个自己的生产车间（读出放大器）负责指定货物的生产，并且每个大仓库都有一个外运站（数据输出寄存器）和寄存托运处（数据输入寄存器与写入驱动器）与传送带相连，前者负责货物的输出中转，后者负责所接受货物并寄存然后帮助搬运工运送到指定储藏间。那么它是如何与调度厂房协同工作的呢？

1、需求方有货物请求了，这个请求发送到调度厂房，调度人员开根据货主的要求给指定的子仓库打电话，电话号码是：区号+子仓库号码+楼层分机（片选+L-Bank 寻址+行有效/选通）。那一层的搬运工接到电话后就开始准备工作。

2、当搬运工点亮所有储藏间的门牌（tRCD）之后，调度人员会告诉搬运工，货物放在哪个储藏间里（列寻址），如果货物很多，并且是连续存放的，调度员会通知搬运工：“一会儿要搬的时候，从起始房间开始连续将后面的 n 个房间的货物都搬出来，我就不再重复了”（突发传输）。但是，他告诉搬运工要等一下，要求所有大仓库的人员统一行动，先别出货。

3、根据事先的规定，搬运工在经过指定的时间后开始将货物扔到传送带上，传送带开始运转并将货物送到生产车间，由它来复制出全新的货物，然后再送到传送带上通过外运站向调度厂房运去。人们通常把从搬运工找到具体储藏间开始，到货物真正出现在送往调度厂房的传送带上的这段时间称之为“输出潜伏期”（CL），而从值班人把货物扔到传送带到货物开始传向调度厂房的这段时间，被称为“货物输出延迟”（tAC），它体现了值班人员的反应时间和生产车间的效率，也影响着仓库基地所在集团（DIMM）的名声。

4、在这个搬运工工作的同时，由于电话对于编号相同的子仓库是并联的，所以其他子仓库相同楼层的搬运工也收到相同的命令，从相同编号的房间搬出货物，运向各自的生产车间。此时，同一批货物同时出现在各自的 16 条传送带上，并整齐地向调度厂房运去。

5、当货物传送完后，原始货物还要送回储藏间保管，这是必须的，但如果没有要求，货物可以一直保留在生产车间，如果再有需要就再生产，而不用再麻烦搬运工了（读出放大器相当于一个 Cache）。调度人员接着会进行下一批货物的调度，当他发现下一批货物在上次操作的子仓库中，但不在刚才通话的那一层，只能再重新拨电话。这时，他通知各子仓库货物翻新运回，清理生产车间，之后挂断电话（预充电命令），这一切必须要在指定时间里（tRP）完成，然后才能给新的楼层打电话。搬运员接到通知后，就将这一 层中所有房间的货物都拿到生产车间进行翻新（没有货物的就不用翻新），然后再搬回储藏间。干完这一切之后，搬运工挂了电话（关闭行）就可以休息了，他们称这种工作为“货物清理返运”（预充电）。这个工作的速度也要快，否则同样会影响集团名声。当然，这个工作可以让搬运工自动完成（自动预充电），只需调度员在当初下搬运指令时提醒一他：“货物运送完了，就进行货物清理返运吧，我不管了”（用 A10 地址线）。

6、当有货物要运来存储时，调度员在向子仓库发送货物的同时就给指定的楼层打电话，让他们准备好房间，此时货物已经到了寄存托运处，没有任何的运送延迟（写入延迟=0），搬运工在托运间的帮助下，向指定的储藏间运送货物，这可需要一定的时间了，他们称之为货物堆放时间（tWR），必须给足搬运工们这一时间，而不能在这期间里让他们干其他的工作，否则他们会令货物丢失并罢工……

（注：本插栏是对 DRAM 操作的形象性描述，谨供辅助性理解本专题，严谨的操作说明见上文。另外，在此请各位读者注意，将内存比喻为仓库只是为了形象化描述，而不要把内存等同理解为存储，它们是有本质的不同的，在本文的比喻中，它只是一个临时性仓库，这一点请大家分清，不要因此产生新的错误概念。）

SDRAM 的结构、时序与性能的关系（上）

在讲完 SDRAM 的基本工作原理和主要操作之后，我们现在要重要分析一下 SDRAM 的时序与性能之间的关系，它不在局限于芯片本身，而是从整体的内存系统去分析。这也是广大 DIYer 所关心的话题。比如 CL 值对性能的影响有多大几乎是每个内存论坛都会有讨论，今天我们就详细探讨一下，其中的很多内容同样适用于 DDR 与 RDRAM。这里需要强调一点，对于内存系统整体而言，一次内存访问就是对一个页的访问，这个页的定义已经在解释 Full Page 含义时讲明了。由于在 P-Bank 中，每个芯片的寻址都是一样的，所以可以将页访问“浓缩”等效为对每芯片中指定行的访问，这样可能比较好理解。但为了与官方标准统一，在下文中会经常用页来描述相关的内容，请读者注意理解。

一、影响性能的主要时序参数

所谓的影响性能是并不是指 SDRAM 的带宽，频率与位宽固定后，带宽也就不可更改了。但这是理想的情况，在内存的工作周期内，不可能总处于数据传输的状态，因为要有命令、寻址等必要的过程。但这些操作占用的时间越短，内存工作的效率越高，性能也就越好。

非数据传输时间的主要组成部分就是各种延迟与潜伏期。通过上文的讲述，大家应该很明显看出有三个参数对内存的性能影响至关重要，它们是 tRCD、CL 和 tRP。每条正规的内存模组都会在标识上注明这三个参数值，可见它们对性能的敏感性。

以内存最主要的操作——读取为例。tRCD 决定了行寻址（有效）至列寻址（读/写命令）之间的间隔，CL 决定了列寻址到数据进行真正被读取所花费的时间，tRP 则决定了相同 L-Bank 中不同工作行转换的速度。现在可以想象一下读取时可能遇到的几种情况（分析写入操作时不用考虑 CL 即可）：

1、要寻址的行与 L-Bank 是空闲的。也就是说该 L-Bank 的所有行是关闭的，此时可直接发送行有效命令，数据读取前的总耗时为 tRCD+CL，这种情况我们称之为页命中（PH，Page Hit）。

2、要寻址的行正好是前一个操作的工作行，也就是说要寻址的行已经处于选通有效状态，此时可直接发送列寻址命令，数据读取前的总耗时仅为 CL，这就是所谓的背靠背（Back to Back）寻址，我们称之为页快速命中（PFH，Page Fast Hit）或页直接命中（PDH，Page Direct Hit）。

3、要寻址的行所在的 L-Bank 中已经有一个行处于活动状态（未关闭），这种现象就被称作寻址冲突，此时就必须要进行预充电来关闭工作行，再对新行发送行有效命令。结果，总耗时就是 tRP+tRCD+CL，这种情况我们称之为页错失（PM，Page Miss）。

显然，PFH 是最理想的寻址情况，PM 则是最糟糕的寻址情况。上述三种情况发生的机率各自简称为 PHR——PH Rate、PFDR——PFH Rate、PMR——PM Rate。因此，系统设计人员（包括内存与北桥芯片）都尽量想提高 PHR 与 PFHR，同时减少 PMR，以达到提高内存工作效率的目的。

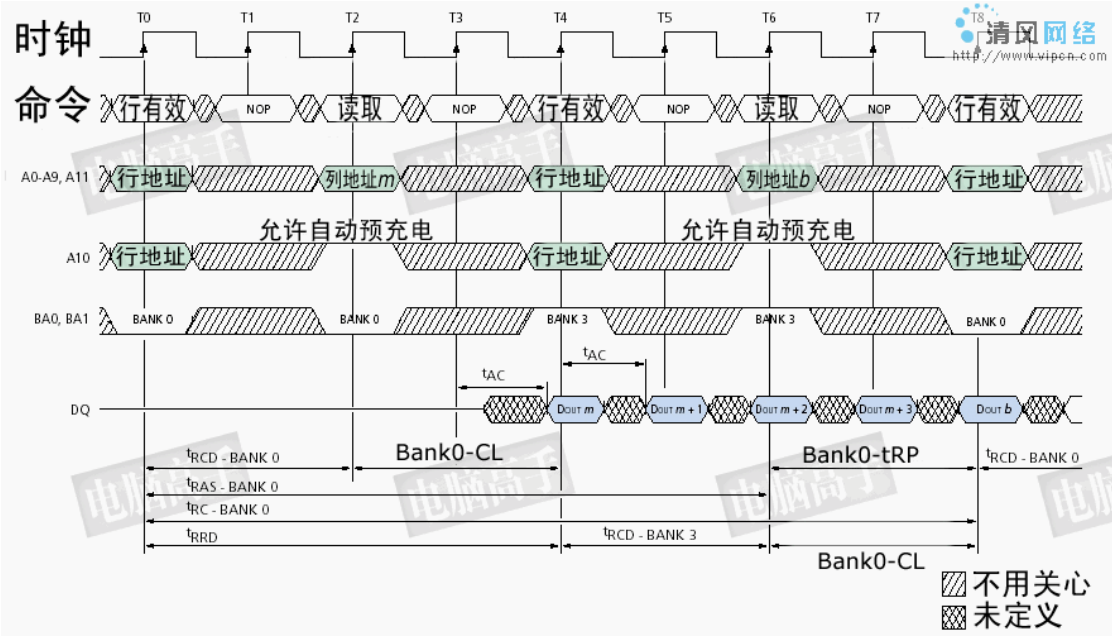
二、增加 PHR 的方法

显然，这与预充电管理策略有着直接的关系，目前有两种方法来尽量提高 PHR。自动预充电技术就是其中之一，它自动的在每次行操作之后进行预充电，从而减少了日后对同一 L-Bank 不同行寻址时发生冲突的可能性。但是，如果要在当前行工作完成后马上打开同一 L-Bank 的另一行工作时，仍然存在 tRP 的延迟。怎么办？ 此时就需要 L-Bank 交错预充电了。

VIA 的 4 路交错式内存控制就是在一个 L-Bank 工作时，对下一个要工作的 L-Bank 进行预充电。这样，预充电与数据的传输交错执行，当访问下一个 L-Bank

时，tRP 已过，就可以直接进入行有效状态了。目前 VIA 声称可以跨 P-Bank 进行 16 路内存交错，并以 LRU 算法进行预充电管理。

有关 L-Bank 交错预充电（存取）的具体执行在本刊 2001 年第 2 期已有详细介绍，这里就不再重复了。



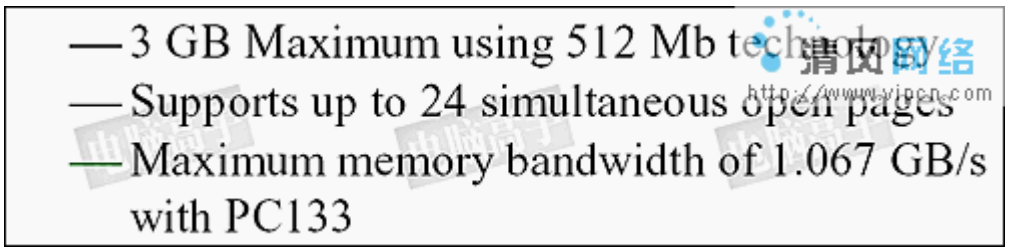
L-Bank 交错自动预充电/读取时序图：L-Bank 0 与 L-Bank 3 实现了无间隔交错读取，避免了 tRP 对性能的影响

三、增加 PFHR 的方法

无论是自动预充电还是交错工作的方法都无法消除 tRCD 所带来的延迟。要解决这个问题，就要尽量让一个工作行在进行预充电前尽可能多的接收多个工作命令，以达到背靠背的效果，此时就只剩下 CL 所造成的读取延迟了（写入时没有延迟）。

如何做到这一点呢？这就是北桥芯片的责任了。在上文的时序图中有一个参数 tRAS（Active to Precharge Command，行有效至预充电命令间隔周期）。它有一个范围，对于 PC133 标准，一般是预充电命令至少要在行有效命令 5 个时钟周期之后发出，最长间隔视芯片而异（基本在 120000ns 左右），否则工作行的数据将有丢失的危险。那么这也就意味着一个工作行从有效（选通）开始，可以有 120000ns 的持续工作时间而不用进行预充电。显然，只要北桥芯片不发出预充电（包括允许自动预充电）的命令，行打开的状态就会一直保持。在此期间的对该行的任何读写操作也就不会有 tRCD 的延迟。可见，如果北桥芯片在能同时打开的行（页）越多，那么 PFHR 也就越大。需要强调的是，这里的同时打开不是指对多行同时寻址（那是不可能的），而是指多行同时处于选通状态。我们可以看到一些 SDRAM 芯片组的资料中会指出可以同时打开多少个页的指标，这可

以说是决定其内存性能的一个重要因素。



Intel 845 芯片组 MCH 的资料：其中表明它可以支持 24 个页面同时处于打开状态

但是，可同时打开的页数也是有限制的。从 SDRAM 的寻址原理讲，同一 L-Bank 中不可能有两个打开的行（S-AMP 只能为一行服务），这就限制了可同时打开的页面总数。以 SDRAM 有 4 个 L-Bank，北桥最多支持 8 个 P-Bank 为例，理论上最多只能有 32 个页面能同时处于打开的状态。而如果只有一个 P-Bank，那么就只剩下 8 个页面，因为有几个 L-Bank 才能有同时打开几个行而互不干扰。Intel 845 的 MHC 虽然可以支持 24 个打开的页面，那也是指 6 个 P-Bank 的情况下（845MCH 只支持 6 个 P-Bank）。可见 845 已经将同时打开页数发挥到了极致。

不过，同时打开页数多了，也对存取策略提出了一定的要求。理论上，要尽量多地使用已打开的页来保证最短的延迟周期，只有在数据不存在（读取时）或页存满了（写入时）再考虑打开新的指定页，这也就是变向的连续读/写。而打开新页时就必须要关闭一个打开的页，如果此时打开的页面已是北桥所支持的最大值但还不到理论极限的话，就需要一个替换策略，一般都是用 LRU 算法来进行，这与 VIA 的交错控制大同小异。

提示： LRU 算法与替换策略

LRU 是 Least Recently Used（近期最少使用）的简称。一般常用于 CPU 的 Cache 中每个 Cache 行（Cache 中基本的存储单元）的替换策略。以 Cache 中的操作为例，它为每个 Cache 行设置一个计数器，Cache 每命中一次，命中行计数器清零，其它各行计数器加 1，因此它是未访问次数计数器。当需要替换时，比较各特定行的计数值，将计数值最大的行换出。内存页面的替换与预充电控制原理也基本一样，设立相应的地址寄存器以存储打开页面（L-Bank）的地址，然后给每个寄存器设立访问计数器，需要替换时，将最近最少用到的页面关闭，并打开新的页面。但是，对于 845 这样的北桥，由于它能打开的页面数已经达到理论的最大值，替换策略反而简单，因为每个 L-Bank 都有一个工作行，所以打开新行也就是就关闭了同一 L-Bank 中的工作行，而无需 LRU 的帮助，因为别无选择。

SDRAM 的结构、时序与性能的关系（下）

四、内存结构对 PHR 的影响

这是结构设计上的问题，所以单独来说。在我们介绍 L-Bank 时，曾经提到单一的 L-Bank 会造成严重的寻址冲突。现在，当我们了解了内存寻址的原理后，就不难理解这句话了。如果只有一个 L-Bank，那么除非是背靠背式的操作（PFH），

否则 tRP、tRCD、CL（读取时）一个也少不了。

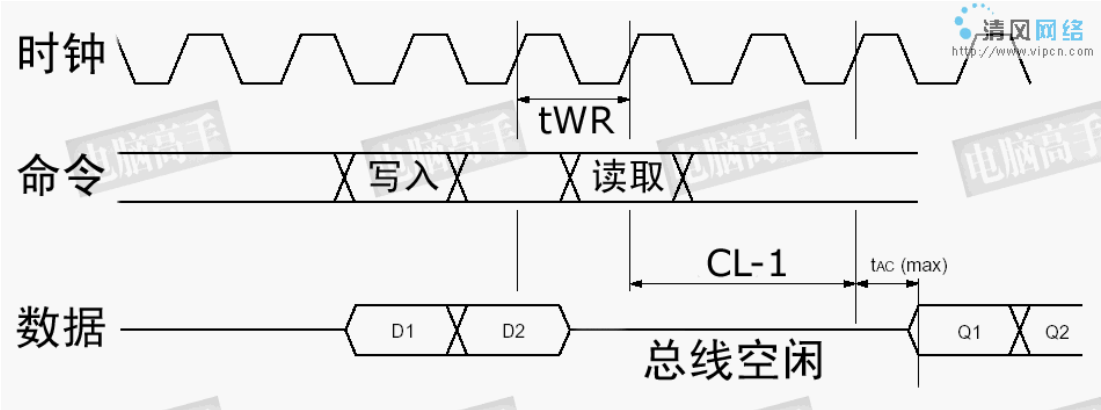
上文中，内存交错之所以能实现就是因为有多个 L-Bank，从这点就可以看出 L-Bank 数量与页命中率之间的关系了。PHR 基本上可以等于“（L-Bank 数-1）/L-Bank 数”。

SDRAM 有 4 个 L-Bank，那么页命中率就是 75%，DDR-Ⅱ SDRAM 最多将有 8 个 L-Bank，PHR 最高为 87.5%。而 RDRAM 则最多有 32 个 L-Bank，PHR 到了惊人的 96.875%，这也是当时 RDRAM 攻击 SDRAM 的一主要方面。

不过，从内存的结构图上可以看出，L-Bank 多了，相应外围辅助的元件也要增加，比如 S-AMP，L-Bank 地址线等等。在 RDRAM 的介绍中，我会讲到 L-Bank 数量增多后所带来的一些新问题。

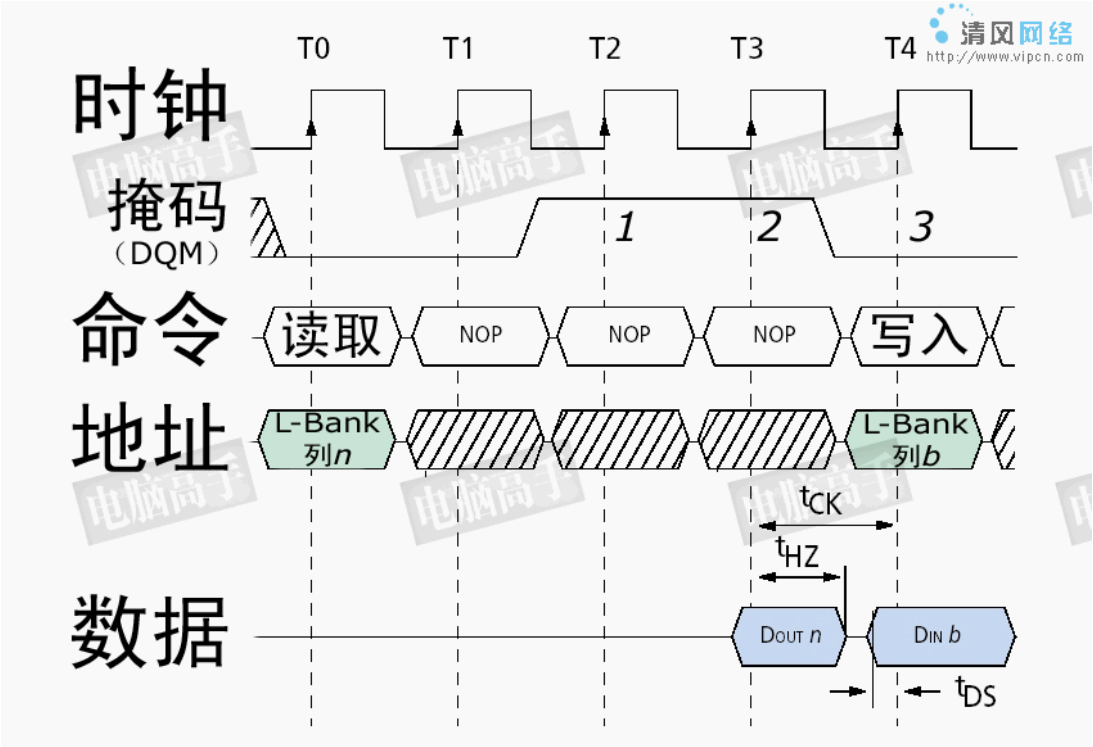
五、读/写延迟不同对性能所造成的影响

SDRAM 在读取操作时会有 CL 造成的延迟，而在写入时则是 0 延迟。这样，在读操作之后马上进行写操作的话，由于没有写延迟，数据线不会出现空闲的时候，保证了数据总线的利用率。但是，若在写操作之后马上进行读操作的话，即使是背靠背式进行，仍然会由于 tWR 与 CL 的存在而造成间隔，这期间数据总线将是空闲的，利用率受到了影响。



在先写后读的操作中，由于保证写入的可靠性，读取命令在 tWR 之后发出，并再经过 CL 才能输出数据，本例中 CL=3，造成了两个时钟周期的总线空闲

这里需要着重说明一下，在突发读取过程中，想立刻中断并进行新的读操作，和读后读模式（见“突发连续读取模式图”）一样，只是新的读命令根据需要提前若干个周期发出，经过 CL 后就会自动传输新的数据。但是，若想中断读后立即进行写操作，就需要数据掩码（DQM）来屏蔽写入命令发出时的数据输出，避免总线冲突。根据芯片设计的不同，有时可能会浪费一个周期进行总线 I/O 的调转，此时一个周期的总线空闲也是不可避免的。



突发读后写时的操作，以本图为例，在最后一个所需数据（本例为第一笔数据）输出前一个周期使 DQM 有效，屏蔽第二笔数据的输出；2、发出写入命令，此时所读取的第二笔数据被屏蔽。3、继续 DQM 以屏蔽第三笔数据的输出。其中 tHZ 表示输出数据与外部电路的连接周期，tDS 表示数据输入准备时间，如果 tHZ+tDS>tCK，那么写入操作就要延后一个周期，这要视芯片的具体设计而定

提示：为什么在中断突发读取时需要 DQM，而中断写入时不用
我们知道行列选通之后，被选中的存储体的电容所存储的信息要被自动读输出，这是不可阻止的。为避免与输入的数据发生总线冲突，要在内部的数据屏蔽逻辑单元中通过 DQM 的控制来屏蔽掉输出的数据。在写命令发出的同时，已经有数据从 S-AMP 输出，而且下一个存储单元（列）也已被打开导通（开始 tAC 的过程），因此一般需要两个 DQM 周期来屏蔽掉已经传出的数据和将要传出的数据，以避免发生总线冲突。由于写入命令的新的列地址生效，下一个计划要读取的列就自动关闭了，所以最多只需两个 DQM。而在中断突发写操作时，之所以不需要 DQM，是因为读命令会让写允许（WE#）无效，而且数据由北桥芯片控制，在发出读取命令的同时数据的传送就会被中断，这也是为什么新的读取命令可以在上一个读取过程中发过以避免延迟的影响，但不能在上一个写过程中发出的原因。

六、BL 对性能的影响

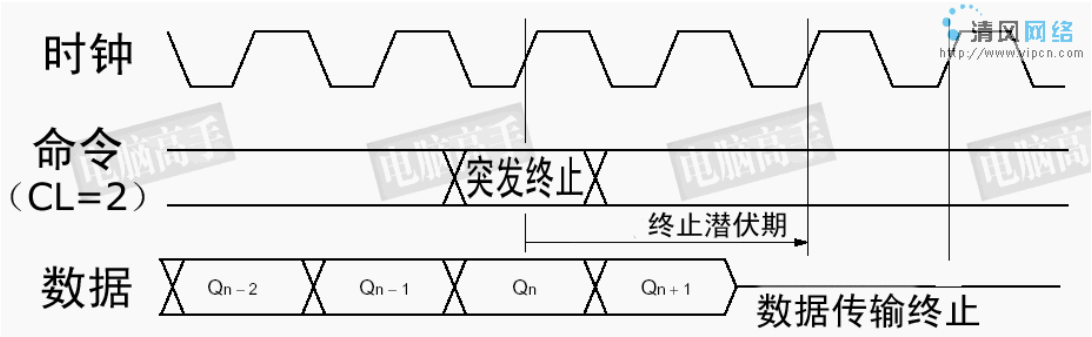
从读/写之间的中断操作我们又引出了 BL（突发长度）对性能影响的话题。首先，BL 的长短与其应用的领域有着很大关系，下表就是目前三个主要的内存应用领域所使用的 BL，这是厂商们经过多年的实践总结出来的。

BL 与相应的工作领域

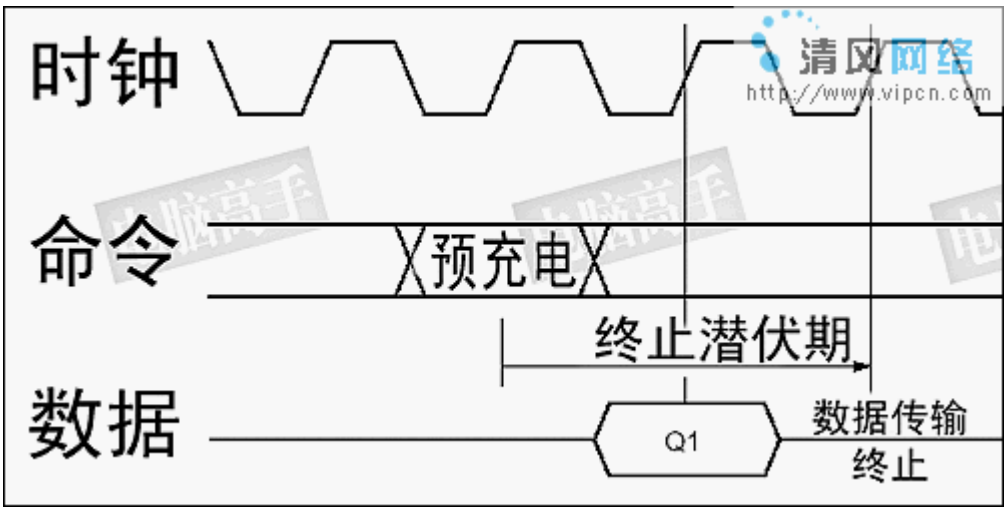
BL（突发长度）	典型应用
1 或 2 （短）	网关/路由器
4 至 8 （中）	PC 内存
8 至 256 （长）	显卡

BL 越长，对于连续的大数据量传输很有好处，但是对零散的数据，BL 太长反而会造成总线周期的浪费。以 P-Bank 位宽 64bit 为例，BL=4 时，一个突发操作能传输 32 字节的数据，但如果只需要前 16 个字节，后两个周期是无效的。如果需要 40 字节，需要再多进行一次突发传输，但实际只需要一个传输周期就够了，从而浪费了三个传输周期。而对于 2KB 的数据，BL=4 的设置意味着要每隔 4 个周期发送新的列地址，并重复 63 次。而对于 BL=256，一次突发就可完成，并且不需要中途再进行控制。不少人都因此表示了 BL 设定对性能影响的担心。

但设计人员也不是傻瓜，通过上文的介绍，可以看出他们在这方面的考虑。通过写命令、DQM、读命令的配合/操作，完全可以任意地中断突发周期开始新的操作，而且 DQM 还可以帮我们在 BL 中选择有用的数据，从而最大限度降低突发传输对性能带来的影响。另外，预充电命令与专用的突发传输终止命令都可以用来中断 BL，前者在中断后进行预充电，后者在中断后不进行其他读/写操作。



专用的突发停止命令可用来中断突发读取，其生效潜伏期与 CL 相同。对于写入则立即有效



用预充电命令来中断突发读取，生效潜伏期与 CL 相同，要小于或等于 tRP。写入时预充电在最后一个有效写入周期完成，并经过 tWR 之后发出，同时立即中断突发传输

所以，突发周期的中断并不难，但用短 BL 应付大数据量存取需要不断的指令与列寻址配合，而为了取消不需要的传输周期，由于需要运用额外的控制，也将占用不少的控制资源。所以 BL 针对不同应用领域有不同设计的主要目的，就是在保证性能的同时，系统控制资源也能得到合理的运用。