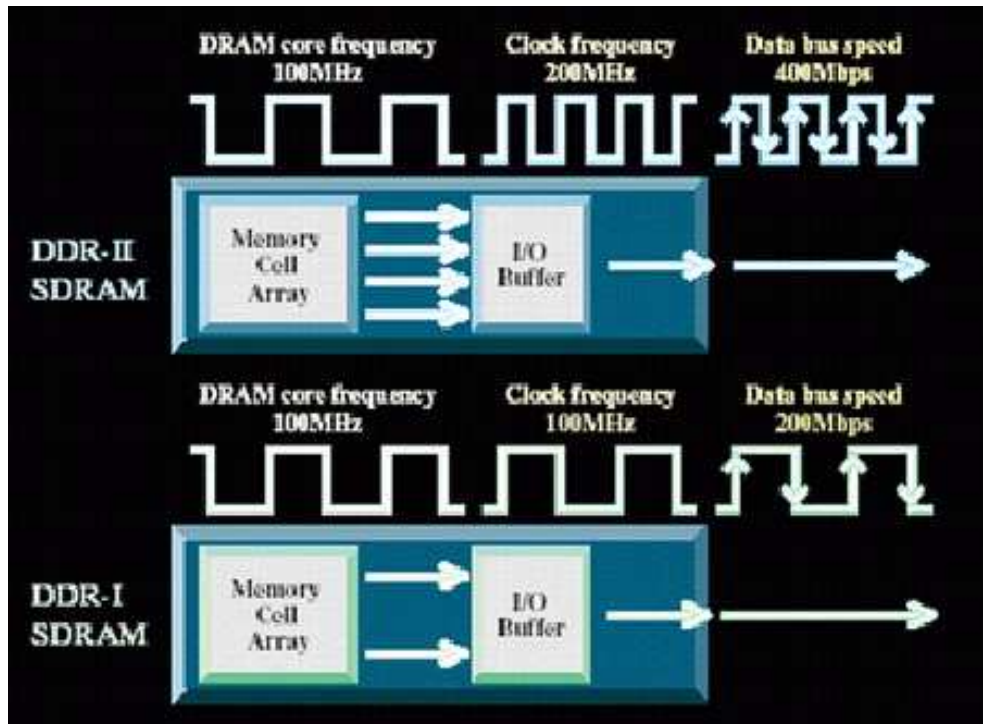


## 1. DDR 的发展：

2003 年秋季 Intel 公布了 DDR2 内存的发展计划。而随着当时 CPU 前端总线带宽的提高和高速局部总线的出现，内存带宽成为系统越来越大的瓶颈。处于主流 DDR 技术已经发展到极至，因此 DDR2 脱颖而出。

DDR2 的实际工作频率是 DDR 的两倍。这得益于 DDR2 内存拥有两倍于标准 DDR 内存的 4bit 预读取能力。下图为 DDR 和 DDR2 预读取能力的对比。



DDR2 内存技术最大的突破点其实不在于用户们所认为的两倍于 DDR 的传输能力，而是在采用更低发热量、更低功耗的情况下（由 2.5V 降为 1.8V），DDR2 可以获得更快的频率提升，突破标准 DDR 的 400MHz 限制。

DDR 内存通常采用 TSOP 芯片封装形式，这种封装形式可以很好的工作在 200MHz 上，当频率更高时，它过长的管脚就会产生很高的阻抗和寄生电容，这会影响它的稳定性和频率提升的难度。这也就是 DDR 的核心频率很难突破 275MHz 的原因。而 DDR2 内存均采用 FBGA 封装形式。不同于目前广泛应用的 TSOP 封装形式，FBGA 封装提供了更好的电气性能与散热性，为 DDR2 内存的稳定工作与未来频率的发展提供了良好的保障。

2007 年中 Intel 表示支持 DDR3 的发展，随后 DDR3 慢慢走上了历史的舞台，根据由 JEDEC 协会所制定的规格来看，由技术面来切入 DDR3 与 DDR2 的异同点，DDR3 拥有高频率低电压的优点，DDR3 可以比 DDR2 运作时省下约 30% 的电力，速

度方面 DDR3 从 800Mbps 起跳最高可以至 1600Mbps，几乎是 DDR2 的二倍速度，正因为高传输率的关系，DDR3 可以在一个时序(Clock)之中传出 8bit 的数据，比起 DDR2 的 4bit 也是二倍的数据传输量，低电压更是 DDR3 的优势之一，1.5V 的电压比 DDR2 的 1.8V 降低了 17%。

下面的图表总结了 DDR，DDR2，以及 DDR3 的一些重要的区别：

Figure-1 DDR3 内存的规格比较表

Items	DDR	DDR2	DDR3
Data rate	266/333/400Mbps	400/533/667/800Mbps	800/1066/1333/1600Mbps
Chip size	256Mb/512Mb	512Mb/1Gb	512Mb/1Gb/2Gb
Module Size	256MB/512MB/1GB	256MB/512MB/1GB/2GB	1GB/2GB/4GB
Voltage	2.5V	1.8V	1.5V
Pre-fetch	2bit	4bit	8bit
Package	TSOP	FBGA	FBGA

## 2、认识内存相关工作流程与参数

首先，我们先了解一下内存的大体结构工作流程，这样会比较容易理解这些参数在其中所起到的作用。这部分的讲述运用 DDR3 的简化时序图。

DDR3 的内部是一个存储阵列，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) 就是逻辑 Bank (Logical Bank，下面简称 Bank)。

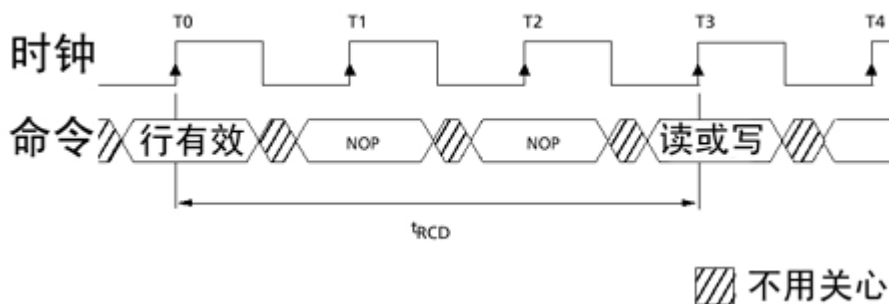
B1	C列地址							
	0	1	2	3	4	5	6	7
R行 地址	1							
	2							
	3							
	4							
	5							
	6							
	7							

DDR3 内部 Bank 示意图，这是一个 NXN 的阵列，B 代表 Bank 地址编号，C 代表列地址编号，R 代表行地址编号。如果寻址命令是 B1、R2、C6，就能确定地址是图中红格的位置

目前 DDR3 内存芯片基本上都是 8 个 Bank 设计，也就是说一共有 8 个这样的“表格”。寻址的流程也就是先指定 Bank 地址，再指定行地址，然后指列地址最终的确寻址单元。

目前 DDR3 系统而言，还存在物理 Bank 的概念，这是对内存子系统的的一个相关术语，并不针对内存芯片。内存为了保证 CPU 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)。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，这个位宽就称为物理 Bank (Physical Bank, 有的资料称之为 Rank) 的位宽。目前这个位宽基本为 64bit。

在实际工作中，Bank 地址与相应的行地址是同时发出的，此时这个命令称之为“行激活” (Row Active)。在此之后，将发送列地址寻址命令与具体的操作命令 (是读还是写)，这两个命令也是同时发出的，所以一般都会以“读/写命令”来表示列寻址。根据相关的标准，从行有效到读/写命令发出之间的间隔被定义为 tRCD，即 RAS to CAS Delay (RAS 至 CAS 延迟，RAS 就是行地址选通脉冲，CAS 就是列地址选通脉冲)，我们可以理解为行选通周期。tRCD 是 DDR 的一个重要时序参数，广义的 tRCD 以时钟周期(tCK, Clock Time)数为单位，比如 tRCD=3，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，DDR3-800，tRCD=3，代表 30ns 的延迟。

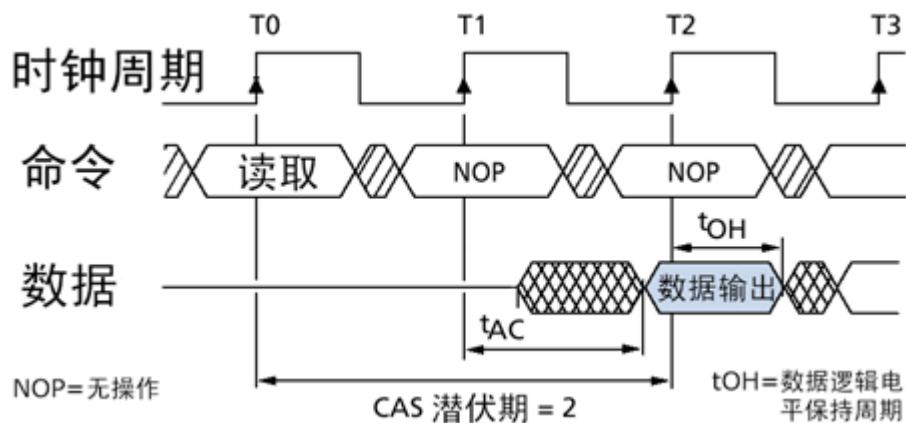


图中显示的是 tRCD=3

接下来，相关的列地址被选中之后，将会触发数据传输，但从存储单元中输出到真正出现在内存芯片的 I/O 接口之间还需要一定的时间 (数据触发本身就有延迟，而且还需要进行信号放大)，这段时间就是非常著名的 CL (CAS

Latency，列地址脉冲选通潜伏期)。CL 的数值与  $t_{RCD}$  一样，以时钟周期数表示。如 DDR3-800，时钟频率为 100MHz，时钟周期为 10ns，如果  $CL=2$  就意味着 20ns 的潜伏期。不过 CL 只是针对读取操作。

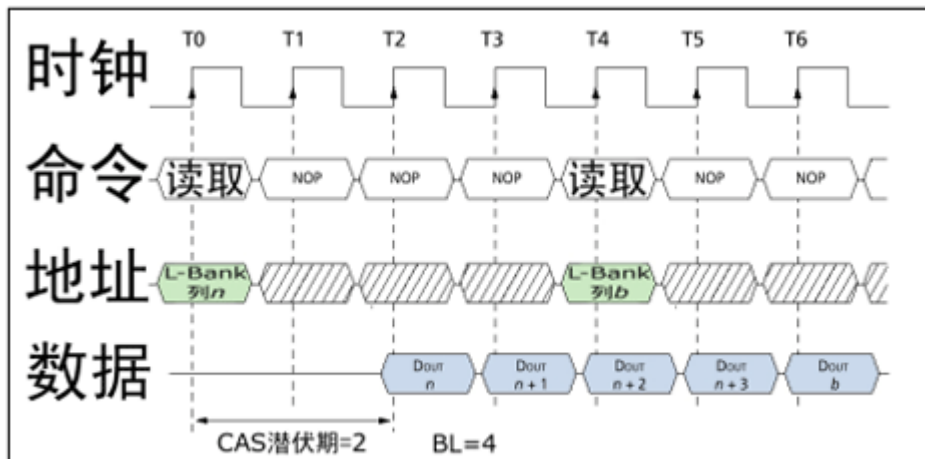
由于芯片体积的原因，存储单元中的电容容量很小，所以信号要经过放大来保证其有效的识别性，这个放大/驱动工作由 S-AMP 负责，一个存储体对应一个 S-AMP 通道。但它要有一个准备时间才能保证信号的发送强度（事前还要进行电压比较以进行逻辑电平的判断），因此从数据 I/O 总线上有数据输出之前的一个时钟上升沿开始，数据即已传向 S-AMP，也就是说此时数据已经被触发，经过一定的驱动时间最终传向数据 I/O 总线进行输出，这段时间我们称之为  $t_{AC}$  (Access Time from CLK，时钟触发后的访问时间)。



图中标准  $CL=2$ ， $t_{AC}=1$

目前内存的读写基本都是连续的，因为与 CPU 交换的数据量以一个 Cache Line（即 CPU 内 Cache 的存储单位）的容量为准，一般为 64 字节。而现有的 Rank 位宽为 8 字节 (64bit)，那么就要一次连续传输 8 次，这就涉及到我们也经常能遇到的突发传输的概念。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，连续传输的周期数就是突发长度 (Burst Lengths，简称 BL)。

在进行突发传输时，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。

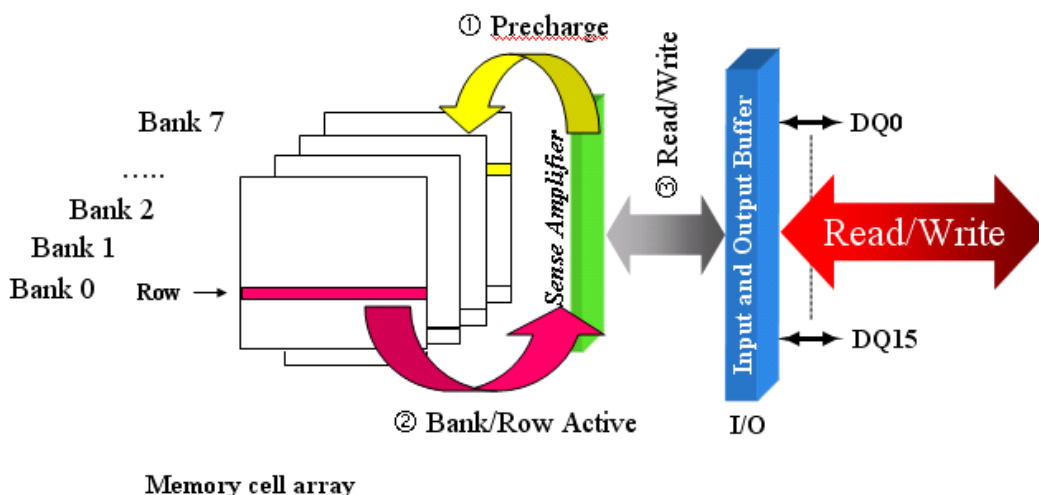


突发连续读取模式：只要指定起始列地址与突发长度，后续的寻址与数据的读取自动进行，而只要控制好两段突发读取命令的间隔周期（与 BL 相同）即可做到连续的突发传输。

谈到了突发长度时。如果  $BL=4$ ，那么也就是说一次就传送  $4 \times 64\text{bit}$  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。

在数据读取完之后，为了腾出读出放大器以供同一 Bank 内其他行的寻址并传输数据，内存芯片将进行预充电的操作来关闭当前工作行。还是以上面那个 Bank 示意图为例。当前寻址的存储单元是 B1、R2、C6。如果接下来的寻址命令是 B1、R2、C4，则不用预充电，因为读出放大器正在为这一行服务。但如果地址命令是 B1、R4、C4，由于是同一 Bank 的不同行，那么就必须要先把 R2 关闭，才能对 R4 寻址。从开始关闭现有的工作行，到可以打开新的工作行之间的间隔就是  $t_{RP}$ （Row Precharge command Period，行预充电有效周期），单位也是时钟周期数。

# DDR3 Architecture



## *Bank Active Procedure*

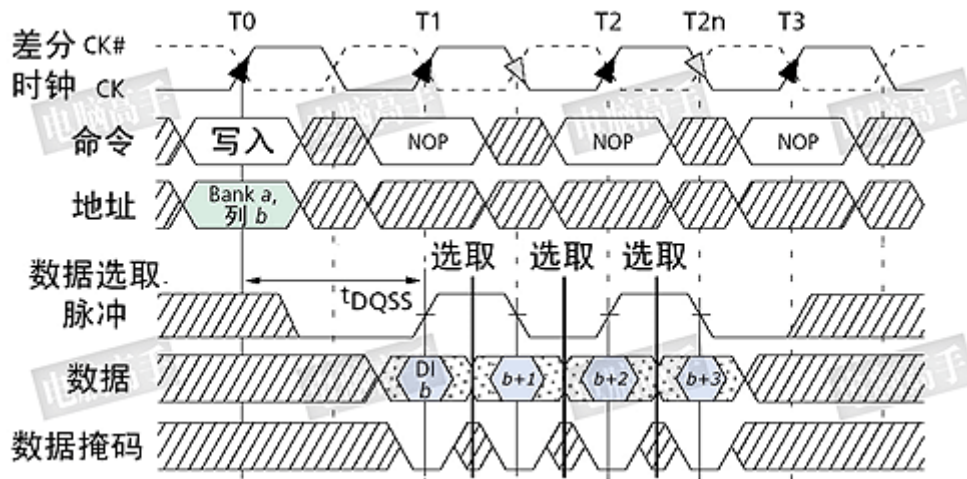
在不同 Bank 间读写也是这样，先把原来数据写回，再激活新的 Bank/Row。

数据选取脉冲（DQS）

DQS 是 DDR 中的重要功能，它的功能主要用来在一个时钟周期内准确的区分出每个传输周期，并便于接收方准确接收数据。每一颗芯片都有一个 DQS 信号线，它是双向的，在写入时它用来传送由北桥发来的 DQS 信号，读取时，则由芯片生成 DQS 向北桥发送。完全可以说，它就是数据的同步信号。

在读取时，DQS 与数据信号同时生成（也是在 CK 与 CK#的交叉点）。而 DDR 内存中的 CL 也就是从 CAS 发出到 DQS 生成的间隔，DQS 生成时，芯片内部的预取已经完毕了，由于预取的原因，实际的数据传出可能会提前于 DQS 发生（数据提前于 DQS 传出）。由于是并行传输，DDR 内存对 tAC 也有一定的要求，对于 DDR266，tAC 的允许范围是 $\pm 0.75\text{ns}$ ，对于 DDR333，则是 $\pm 0.7\text{ns}$ ，有关它们的时序图示见前文，其中 CL 里包含了一段 DQS 的导入期。

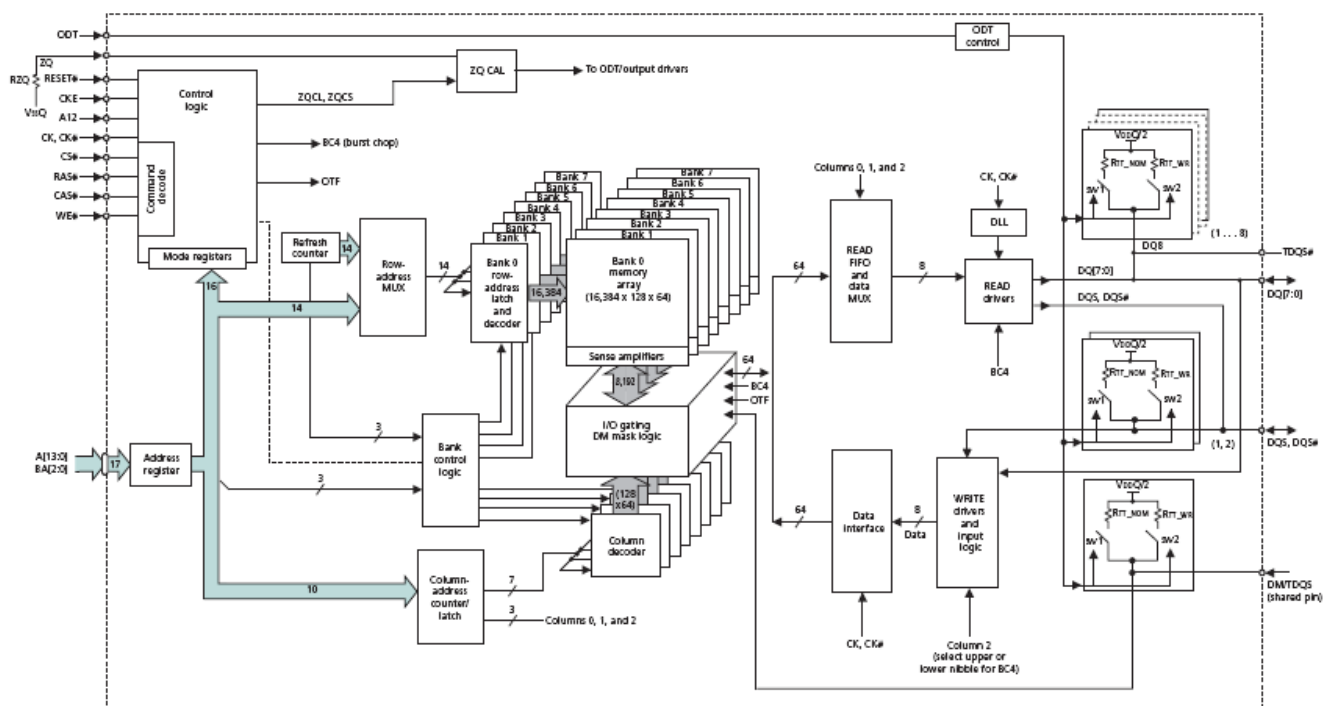
DQS 在读取时与数据同步传输，那么接收时也是以 DQS 的上下沿为准吗？不，如果以 DQS 的上下沿区分数据周期的危险很大。由于芯片有预取的操作，所以输出时的同步很难控制，只能限制在一定的时间内，数据在各 I/O 端口的出现时间可能有快有慢，会与 DQS 有一定的间隔，这也就是为什么要有一个 tAC 规定的原因。而在接收方，一切必须保证同步接收，不能有 tAC 之类的偏差。这样在写入时，芯片不再自己生成 DQS，而以发送方传来的 DQS 为基准，并相应延后一定的时间，在 DQS 的中部为数据周期的选取分割点（在读取时分割点就是上下沿），从这里分隔开两个传输周期。这样做的好处是，由于各数据信号都会有一个逻辑电平保持周期，即使发送时不同步，在 DQS 上下沿时都处于保持周期中，此时数据接收触发的准确性无疑是最高。



在写入时，以 DQS 的高/低电平期中部为数据周期分割点，而不是上/下沿，但数据的接收触发仍为 DQS 的上/下沿

### 3. 容量的计算

**Figure 4: 128 Meg x 8 Functional Block Diagram**



上图为 X8data 的单颗 DDR3 架构图，行 (Row) 地址线复用 14 根，列 (Column) 地址线复用 10 根，Bank 数量为 8 个，IO Buffer 通过 8 组数位线 (DQ0-DQ7) 来完成对外的通信，故此单颗 DDR3 芯片的容量为  $2^{14} \times 2^{10} \times 8$ ，结果为 1Gbit，因为 1B 包含 8bit， $1GB/8=128MB$ 。

如果我们要做成容量为 1GB 的内存条则需要 8 颗这样的 DDR3 内存芯片，每颗芯片含 8 根数位线 (DQ0-DQ7) 则总数宽为 64bit，这样正好用了一个 Rank。

假果还用 128MB 的 DDR3 芯片去做 2GB 内存条，结果就会有所不同。我们最好选用 4 根数位线 (DQ0-DQ3)，数量是 16 颗，这样也是用了一个 Rank。

在 K2 的项目中我们要做容量为 8GB 的内存条，则数量用 64 颗 128M 的 DDR3，这样位宽高达  $64 \times 4 = 256\text{bit}$ ，要做成 4 个 Rank。

#### 4. 下面我们来介绍一下 DDR3 的工作情况和一些基本 command

内存要保存信息就要不断的 refresh，

refresh 操作与预充电操作一样，都是用 S-AMP 先读再写。预充电是对一个或所有 Bank 中的工作行操作，并且是不定期的，而 refresh 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 Bank Precharge 不同的是，这里的行是指所有 Bank 中地址相同的行，而 Precharge 中各 Bank 中的工作行地址并不是一定是相同的。

refresh 操作分为两种：(Auto Refresh, 简称 AR) 与 (Self Refresh, 简称 SR)。不论是何种 refresh 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 refresh 计数器）用来自动的依次生成行地址。refresh 涉及到所有 Bank，因此在 refresh 过程中，所有 Bank 都停止工作，而每次 refresh 所占用的时间为 9 个时钟周期 (PC133 标准)，之后就可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。显然，refresh 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 refresh 操作。在 SR 期间除了 CKE 之外的所有外部信号都是无效的，只有重新使 CKE 有效才能退出自 refresh 模式并进入正常操作状态。





与发送器都将关闭。所有内部的程序装置将复位，DLL（延迟锁定回路）与时钟电路将停止工作，而且不理睬数据总线上的任何动静，将使 DDR3 达到最节省电力的目的。

## DDR3 Command

Command	CS#	RAS#	CAS#	WE#	BA#	Addr
Row/Bank Active	L	L	H	H	BA	A <sub>10</sub> = "L" A <sub>11</sub> = "L"
Read	L	H	L	H	BA	CA, A <sub>10</sub> /AP
Write	L	H	L	L	BA	CA, A <sub>10</sub> /AP
Precharge	L	L	H	L	BA	A <sub>10</sub> /AP
Refresh	L	L	L	H	x	x
MRS*	L	L	L	L	OPCode	OPCode

AP: Auto Precharge

BA: Bank Address

CA: Column Address

RA: Row Address

MRS: Mode Register Set

上表中 MRS 可以设置 Mode 寄存器值

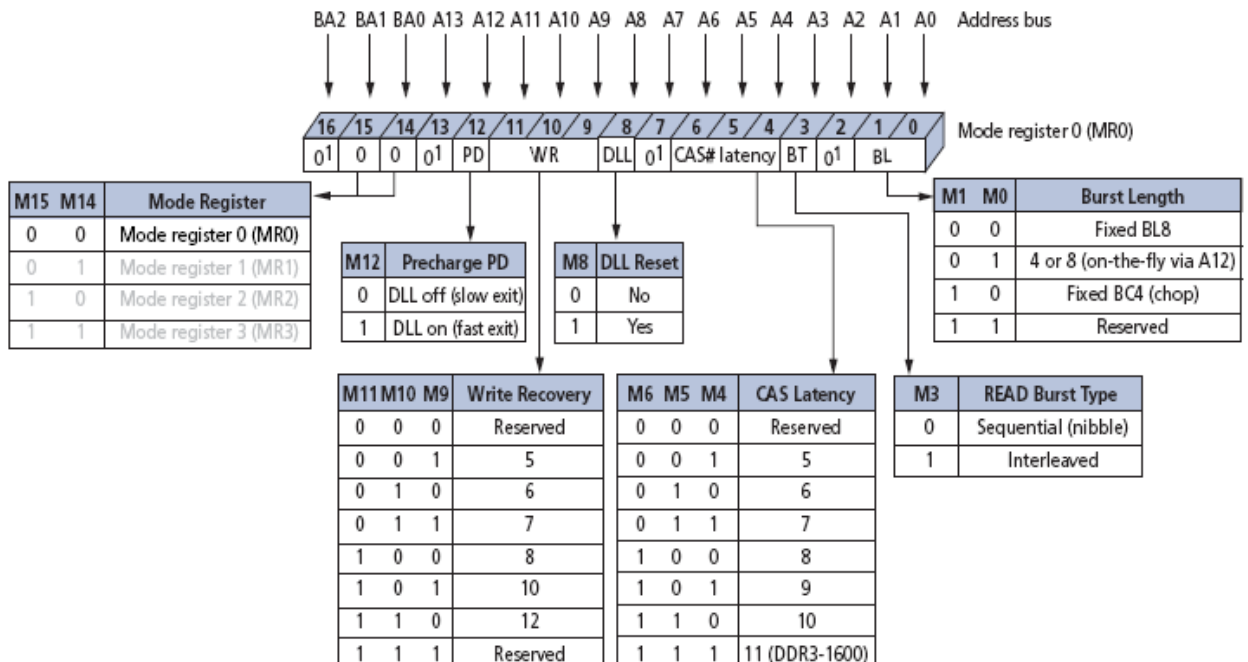
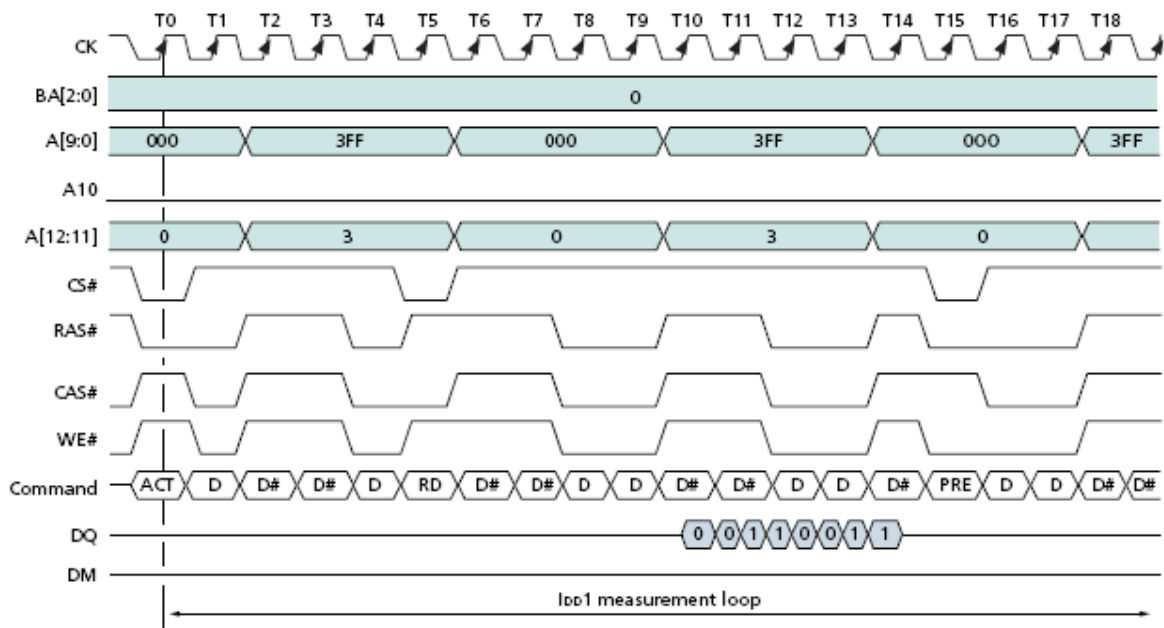


Figure 13: Idd1 Example – DDR3-800, 5-5-5, x8 (-25E)



Notes: 1. Data DQ is shown, but the output buffer should be switched off (per MR1[12] = 1) to achieve IOUT = 0mA (MR1[12] = 0 is reflected in this example; however, test conditions are MR1[12] = 1). Address inputs are split into three parts.

以上图为例 CS#, RAS#, CAS#, WE#为 L, L, H, H。则指令为 Row/Bank Active; 随后 CS#拉高, command 无效, 在第 4 个时钟周期这 4 个信号变为 L, H, L, H, 对照表格, 指令为 Read, 经过几个时钟周期延迟, 在 3CLK 后读数据。