SDRAM 理论篇之基础知识及操作时序讲解

FPGA 学习笔记 Kevin 2 年前 (2016-01-12) 8931℃ 0 评论

这个星期在进行 SDRAM 的学习，当然关于 SDRAM 的理论知识，在上周讲了一部分，而这周主要的还是写代码及调试。如果有朋友在这之前没有接触过内存的话，可以看下业内写的比较好的扫盲文章《高手进阶 终极内存技术指南》（点击进入下载），虽然 Kevin 在这之前也曾经看过这篇文章，但是看完之后对于写代码还是毫无头绪，所以 Kevin 在这里建议大家，如果是完全不知道内存是神马的朋友可以看下这篇文章，对于一些有基础但是不了解 SDRAM 操作时序的朋友可以直接看 SDRAM 厂商的官方文档，虽然 Kevin 也会在这篇博文中会根据自己对官方手册的理解来讲解 SDRAM 操作时序，但还是建议大家多看英文文档，这样才能体会原汁原味的知识，这也是为了避免所谓的“权威”中文版给大家带来某些误导，毕竟也是会有一些朋友在看着“权威”的中文翻译版本却在不停的骂娘，哈哈。所以大家也不要轻信所谓的中文翻译版手册，因为每个人对于手册中的英文意思的理解都有可能有区别，所以大家要相信厂商的手册才是最权威的。

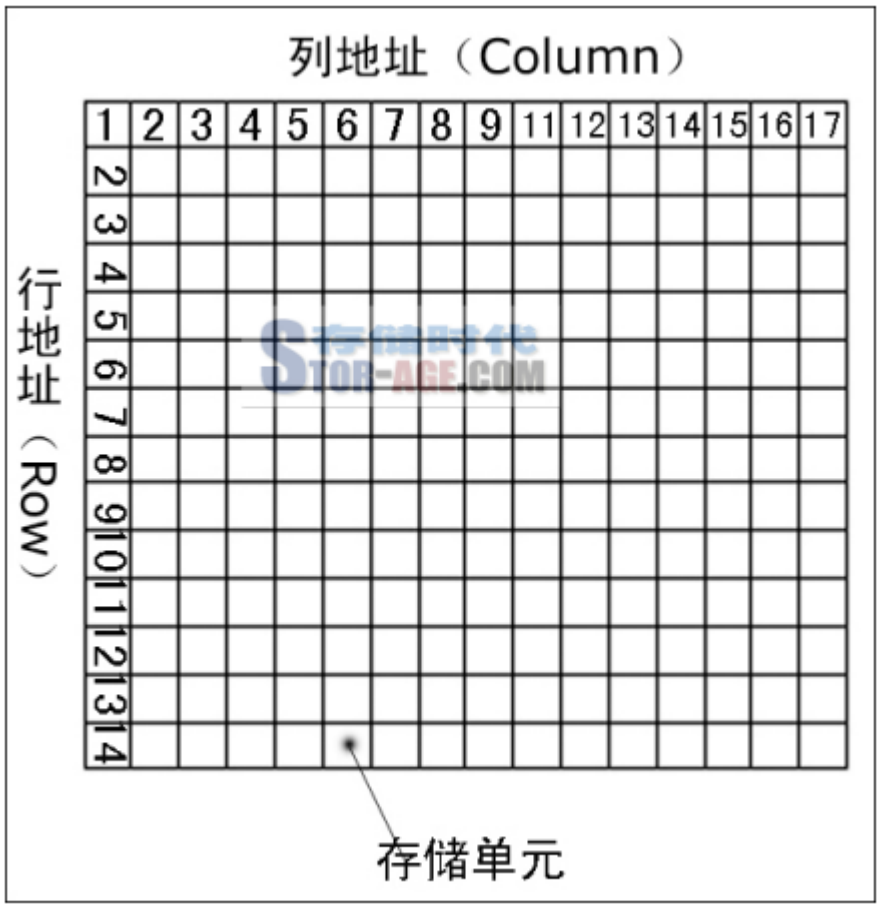
接下来，咱们正式开始讲解 SDRAM 的有关知识。

SDRAM（Synchronous Dynamic Random Access Memory），同步动态随机存储器。同步是指 Memory 工作需要同步时钟，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存储的数据不丢失，因为 SDRAM 中存储数据是通过电容来工作的，大家知道电容在自然放置状态是会有放电的，如果电放完了，也就意味着 SDRAM 中的数据丢失了，所以 SDRAM 需要在电容的电量放完之前进行刷新；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。

关于 SDRAM 的内部结构

我们借用《高手进阶 终极内存技术指南》中的部分文字来进行讲解：

SDRAM 内部其实是一个存储阵列，大家可以想象一下，如果 SDRAM 内部不是以阵列的形式存在而是以管道的形式存在，那 SDRAM 是很难做到随机访问的。阵列就如同一张表格一样，将数据填进去。和表格的检索原理一样，先确定一个行（Row），再确定一个列（Col），我们就可以准确的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，这个单元格可以成为存储单元，那么这个表格（存储阵列）叫什么呢？这就是逻辑 Bank (Logic Bank,下文简称 Bank).



L-Bank 存储阵列示意图

看到这里，大家应该不难想象，我们对 SDRAM 进行寻址的方式为：先确定 Bank 的地址，再确定行和列的地址。

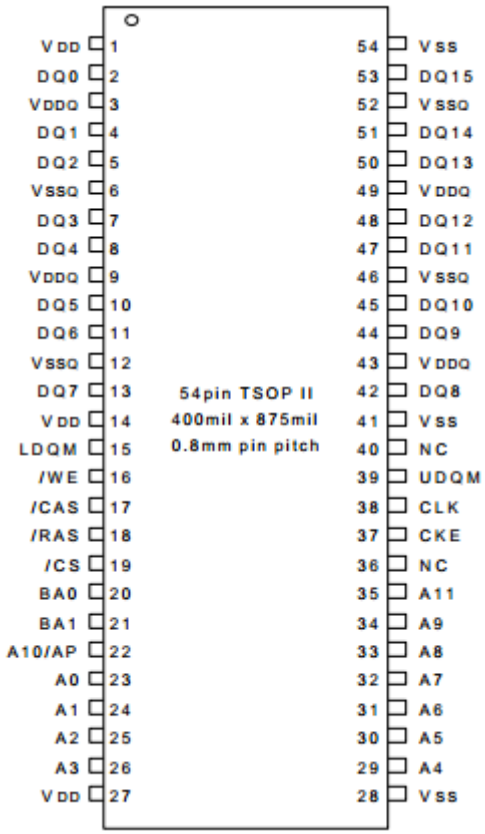
SDRAM 内存容量计算方式

关于内存的计算，我们先看下手册

图中的 128Mb 表示 SDRAM 的总容量，2M：表示 1 个 Bank 共有 2M 个地址，也就是说一个 bank 有 2M 个存储单元；4Bank 表示有 4 个 Bank；16 I/O 表示位宽为 16 位，数据总线用的是 16 位的，1 个存储单元能存储的最大的数为 2^16-1=65535。然后再根据各个参数算下容量：2M x 4 x 16b = 128Mb .

SDRAM 芯片引脚介绍

第一个图是 SDRAM 的引脚图，第二图是各个引脚信号的描述信息


VDD 1
DQ0 2
VDDQ 3
DQ1 4
DQ2 5
VSSQ 6
DQ3 7
DQ4 8
VDDQ 9
DQ5 10
DQ6 11
VSSQ 12
DQ7 13
VDD 14
LDQM 15
/WE 16
/CAS 17
/RAS 18
/CS 19
BA0 20
BA1 21
A10/AP 22
A0 23
A1 24
A2 25
A3 26
VDD 27
54 VSS
53 DQ15
52 VSSQ
51 DQ14
50 DQ13
49 VDDQ
48 DQ12
47 DQ11
46 VSSQ
45 DQ10
44 DQ9
43 VDDQ
42 DQ8
41 VSS
40 NC
39 UDQM
38 CLK
37 CKE
36 NC
35 A11
34 A9
33 A8
32 A7
31 A6
30 A5
29 A4
28 VSS
54pin TSOP II
400mil x 875mil
0.8mm pin pitch

PIN DESCRIPTION

PIN	PIN NAME	DESCRIPT
CLK	Clock	The system clock input. All other inputs ar rising edge of CLK
CKE	Clock Enable	Controls internal clock signal and when de of the states among power down, suspend
$\overline{CS}$	Chip Select	Enables or disables all inputs except CLK
BA0, BA1	Bank Address	Selects bank to be activated during $\overline{RAS}$ a Selects bank to be read/written during CA
A0 ~ A11	Address	Row Address : RA0 ~ RA11, Column Addr Auto-precharge flag : A10
$\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$	Row Address Strobe, Column Address Strobe, Write Enable	$\overline{RAS}$, $\overline{CAS}$ and $\overline{WE}$ define the operation Refer function truth table for details
UDQM, LDQM	Data Input/Output Mask	Controls output buffers in read mode and
DQ0 ~ DQ15	Data Input/Output	Multiplexed data input / output pin
VDD/VSS	Power Supply/Ground	Power supply for internal circuits and input
VDDQ/VSSQ	Data Output Power/Ground	Power supply for output buffers
NC	No Connection	No connection

上面的两个图在 SDRAM 芯片的手册上都是可以找到的，下面 Kevin 来简单的介绍下我们做 FPGA 开发需要重点关注的信号

CLK：SDRAM 工作的时钟，并且所有的输入信号都是在 CLK 的上升沿进行检测的，也就是说我们给 SDRAM 给的任何命令，一定要在 CLK 的上升沿保持稳定，以免 SDRAM 获取我们给出的命令时出现错误。

CKE：时钟使能信号，是用来控制 SDRAM 内部时钟是否工作的一个信号（在 SDRAM 内部也是有时钟的哦）

CS：片选信号，这里需要注意的是，如果要对 SDRAM 进行操作，必须要将片选信号拉低

BA0，BA1：Bank 地址线，用来给 bank 的地址，可以控制 SDRAM 的 4 个 bank

A0～A11：地址线，当我们选择 SDRAM 某个 Bank 的 Row 地址的时候，需要到 12 根地址线（A0～A11）；当选择 Col 地址的时候，只用 A0～A8 这 9 根线；A10 这个信号可以用来控制 Auto-precharge。

RAS、CAS、WE：这三根线就是用来给 SDRAM 发命令的，包括初始化、读、写、自动充电等命令。

UDQM、LDQM：数据输入/输出掩码。

DQ0～DQ15：SDRAM 的数据线，为双向的，向 SDRAM 写数据或者从 SDRAM 中读出来的数据都是在 DQ 上进行传输的

大家看到上面的 A0～A11 这根地址总线既可以来控制 row 也可以控制 col，而且控制 col 地址与控制 row 地址用到的线的数量又不一样，可能会有些疑问，下面 Kevin 向大家解释一下：

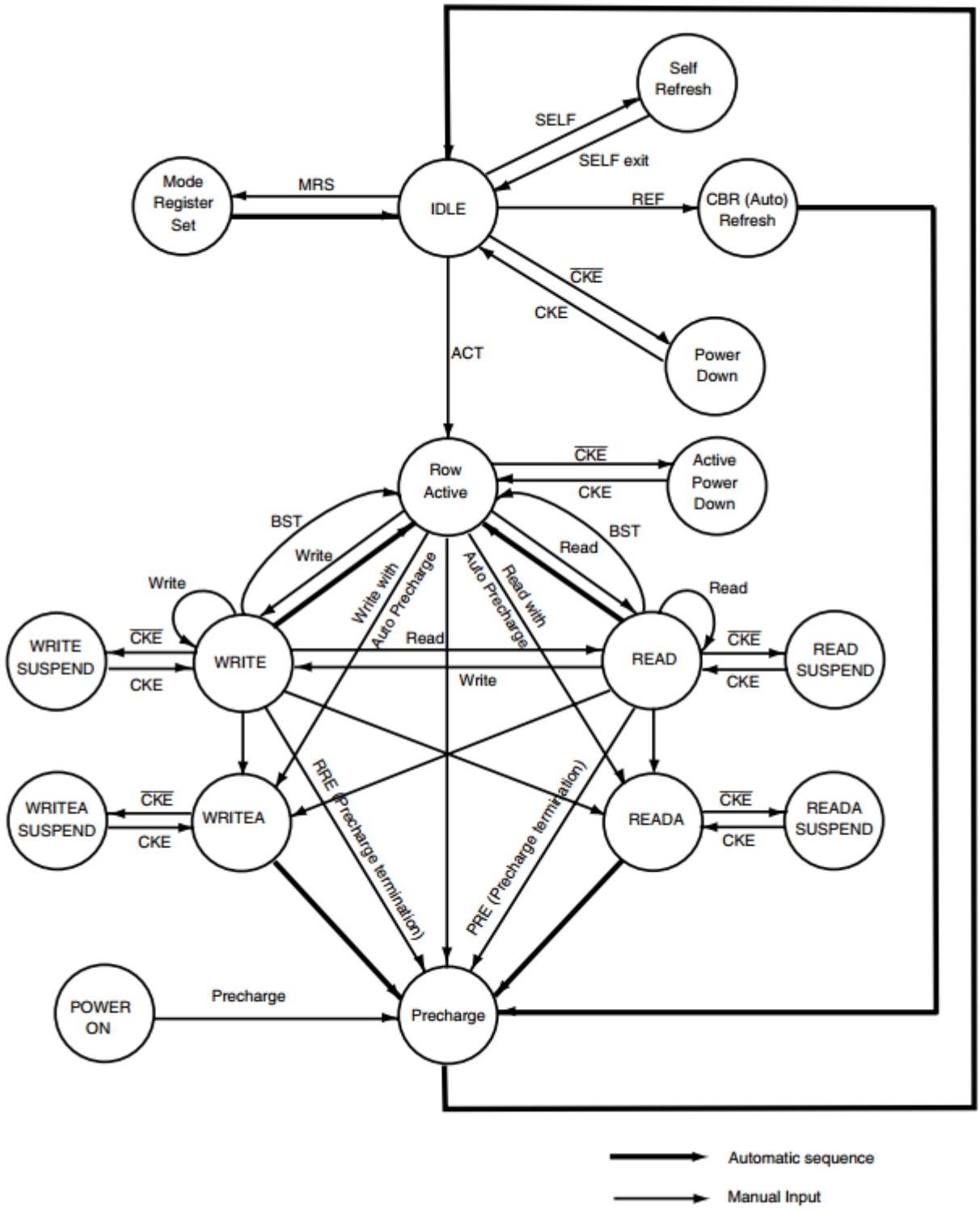
SDRAM 的厂商一般为了节约成本，采用同一总线来对 SDRAM 进行寻址是无可厚非的，对于 Row 地址用到了 12 根线，也就是总共有 2^12=4096 个 Row 地址，而 col 地址使用 9 根线，也就是有 2^9=512 个 col 地址，而总共加起来的话，一个 Bank 就有 4096×512=2097152 (2x1024x1024) ，也就是有 2M 个地址，这样的话，就与我们前面计算 SDRAM 的容量想吻合了。

另外一个的话，可能大家不太明白掩码（UDQM、LDQM）的作用，下面请您耐着性子听 Kevin 的解释：

就拿咱们使用的这块 SDRAM 芯片来讲，数据线有 16 根，也就是说明我们数据的位数可以达到 16 位，但是呢，请注意，也许在我们使用 SDRAM 的时候，也许我们在向 SDRAM 写数据的时候，我们生成的数据只有 8 位，但 FPGA 是与 SDRAM 的 16 根数据线连在一起的，这个时候，存到 SDRAM 中的数据还是 16 位的，所以为了避免这个问题，我们就可以使用掩码来屏蔽掉高 8 位了。当然掩码在读数据的时候起到的作用也是类似的。

SDRAM 的相关操作时序

前面铺垫了这么多 SDRAM 的相关知识，相信大家已经迫不及待想知道怎么来操作 SDRAM 了，各位看官且听 Kevin 详细道来。



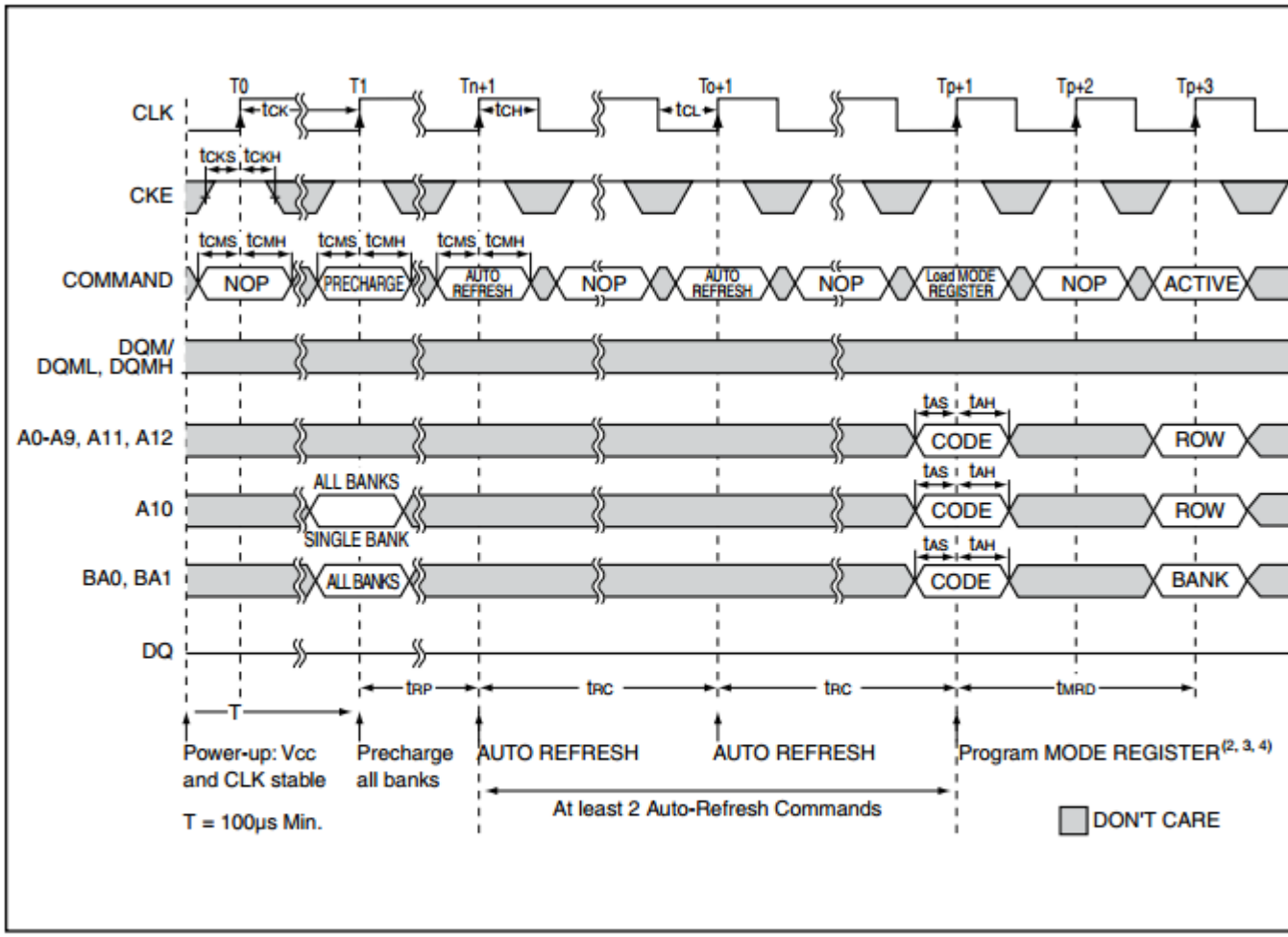
上图是 SDRAM 内部的状态跳转图（可能在网页上看到的图片不太清晰，大家可以在导航菜单“福利”下的“文档手册”中下载这些文档），粗黑线表示在该状态下会自动跳转到另一个状态，细黑线表示需要给命令才会跳转。

SDRAM 初始化

我们先找到 SDRAM 最开始的状态“POWER ON”，这是刚上电的状态。在“POWER ON”状态给‘Precharge’命令之后就会跳转到“Precharge”状态，然后自动跳转至“IDLE”状态。在“IDLE”状态下，我们需要给 SDRAM 两次 Auto-refresh 命令，然后接着需要进行模式寄存器设置，对模式寄存器设计完毕之后，我们的初始化过程也就结束了。说了这么多，我们还是不知道怎么给 SDRAM 进行初始化，当时 Kevin 看过《高手进阶 终极内存技术指南》之后也是依然不知道怎么写代码对 SDRAM 进行初始化。在这篇博文里边，Kevin 暂时不会讲如何写代码，我只会把初始化这个过程相关的时序及原理给大家讲清楚，原理理解透彻了，相信大家对于代码如何进行书写也是会有一定的想法的，关于代码如何书写，也可以参考 Kevin 的另一篇博文。

我们继续上一张官方文档中的图片，这次来的可以波形图哦，各位看官要看仔细咯，对于你写代码是很有帮助的：

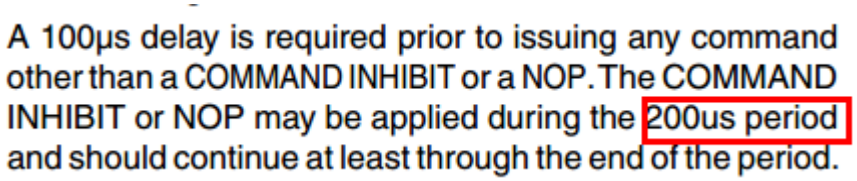
INITIALIZE AND LOAD MODE REGISTER(1)



Notes:
1. If $\overline{CS}$ is High at clock High time, all commands applied are NOP.
2. The Mode register may be loaded prior to the Auto-Refresh cycles if desired.
3. JEDEC and PC100 specify three clocks.
4. Outputs are guaranteed High-Z after the command is issued.

可能大家第一次接触到这种时序图的时候会有点头疼，不过没关系，请听 Kevin 给大家慢慢解释，在看文档解释的同时，与上边的状态图结合起来会更容易理解一点，对于图中出现的每一个指令，我们都是可以在我们所使用的器件手册中找到：

根据官方文档的介绍，SDRAM 在刚上电的 100us 期间，除了“COMMAND INHIBIT”或“NOP”是不可以给 SDRAM 发其他指令的，所以大家能看到发送“Precharge”命令与刚上电的时间间隔“T”最小值为 100us，这个 100us 其实也相当于 SDRAM 的一个稳定期。看过《高手进阶 终极内存技术指南》这篇文章的朋友可能会发现，在《高手进阶 终极内存技术指南》上写的稳定器是 200us，而这里出现的却是 100us，这样岂不是出现了矛盾，Kevin 刚看手册的时候，也出现了这个疑问。我们可以肯定的一点是，官方的手册出错的可能性是很小的，如果连官方的文档都出错了，那还让我们这些使用他们器件的人怎么活呢 ╭(T □ T)╮ 。那会不会是《高手进阶 终极内存技术指南》的作者写错了呢？这么久经考验的经典文章，如果有错误，肯定是早就提出来了。最终，Kevin 在官方手册上找到了答案：



对于这段英文，Kevin 的理解是：在 200us 期间，SDRAM 是都有可能执行“COMMAND INHIBIT”或“NOP”命令的，所以这里我们就索性把稳定器弄成 200us。（如果理解出错，请大家批评指出）

稳定器过了之后，我们先给“Precharge”命令，然后再过“tRP”的时间再给“Auto Refresh”命令，然后再过“tRC”的时间再给“Auto Refresh”命令，然后再经过“tRP”的时间进行模式寄存器设置。在给命令的时候，我们需要注意，“Precharge”、“Auto Refresh”、”Load Mode REGISITER“这三个命令都是只出现了一次的，没有给这些命令的时候都是给的”NOP“命令。

给”Precharge”命令时，我们需要指定 A10 及 Bank 地址，如果 A10 为高(All Banks)，就意味着是给所有的 Bank 进行预充电，此时不需要给 Bank 地址，如果 A10 为低（SINGLE BANK）,就需要指定某一个 bank 的地址。

给”Auto Refresh“命令时，是不需要指定 bank 地址的（大家注意看右下角有说明，灰色部分的数据我们是不需要关心的）。

进行模式寄存器设置的时候，需要给的指令会稍微复杂一点，手册上显示 A0～A11 及 BA0，BA1 都用到了，下面我们来看下模式寄存器应该怎么进行设置，请看图：

Mode Register

BA1	BA0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	OP Code	0	0	CAS Latency			BT	Burst Length		

OP Code

A9	Write Mode
0	Burst Read and Burst Write
1	Burst Read and Single Write

Burst Type

A3	Burst Type
0	Sequential
1	Interleave

CAS Latency

A6	A5	A4	CAS Latency
0	0	0	Reserved
0	0	1	Reserved
0	1	0	2
0	1	1	3
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

Burst Length

A2	A1	A0	Burst Length	
			A3 = 0	A3=1
0	0	0	1	1
0	0	1	2	2
0	1	0	4	4
0	1	1	8	8
1	0	0	Reserved	Reserved
1	0	1	Reserved	Reserved
1	1	0	Reserved	Reserved
1	1	1	Full Page	Reserved

A9：用来指定操作模式：为高表示突发读/突发写，为低表示突发读/单写；

A4~A6：指定潜伏期的长度，关于潜伏期，《高手进阶 终极内存技术指南》上有非常详细的介绍，不过潜伏期设置的实际效果就是，如果潜伏期设置的是 3，在我们进行 SDRAM 读操作的时候，读出来的数据就会相对于”READ”命令延后 3 个周期出来，如果潜伏期是 2，那就会延后 2 个周期。

A3：设置突发的类型，连续型和非连续型。

A2～A0：用来指定突发的长度。

对于上面的解释，大家可能会由于不懂突发是什么会觉得 Kevin 这里边写得可能还是显得不够直白，下面 Kevin 举个例子：A9 设置为 0，潜伏期设置为 3，突发类型 A3 为 0，突发长度（A2～A0）设置为 4，在我们进行写操作的时候，数据是每 4 个数据写一次的，就是说我们给一次写指令，就会向 SDRAM 写进去 4 个数据，而且四个地址是连续的（如果突发类型设置的是非连续，则地址不会连续，具体的地址会是怎么样的一种规律，Kevin 自己也没搞懂）；在我们进行读操作的时候，在给完一次读命令的 3 个周期（潜伏期为 3）后，会有 4 个数据连续的崩出来.

至此，对于初始化的过程，到这里算是基本讲完了。

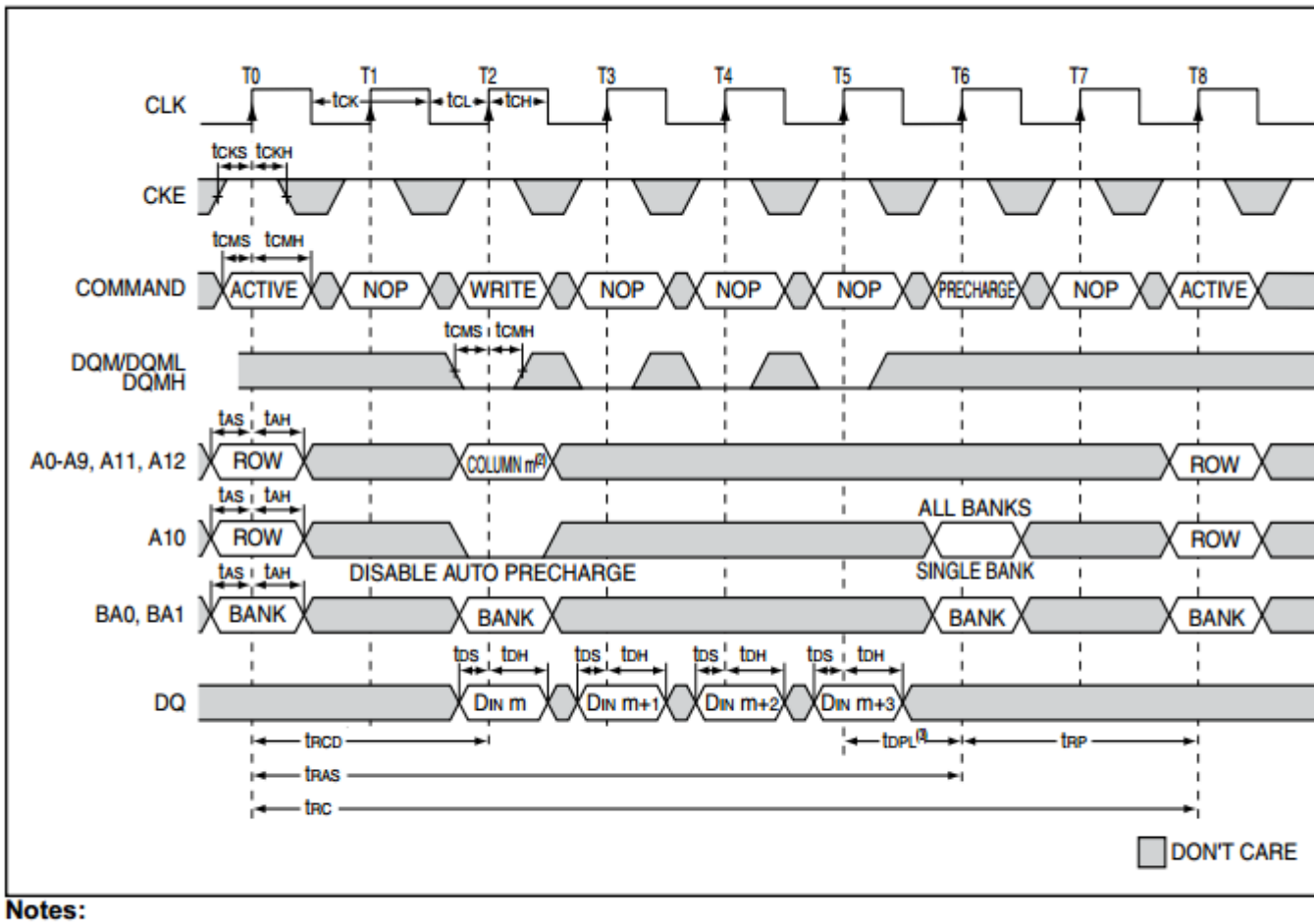
SDRAM 写操作

在状态转移图中，我们可以发现在初始化完成之后，对 SDRAM 进行读或者写操作之前，我们还需要有一个命令”ACT”（这个命令在我们的初始化时序图中也出现了，只是我们是拿到了这里来讲），这个命令的意思说成大白话就是”行有效”命令，就是让 SDRAM 中的某一行活动起来，以便我们进行读或写。

在正式讲写操作之前，还有一点希望大家能注意一下，可能细心的朋友已经发现了，在我们的状态转移图中有”WRITE”和”WRITEA”这两个状态，处于这两个状态时，我们都可以对 SDRAM 进行写操作，只是说在”WRITEA”状态，我们每写完一个突发长度的数据之后，SDRAM 会自动跳出这个状态进行刷新，而在”WRITE”状态，是需要给相应的指令之后才会跳出”WRITE”状态的，所以为了提高 SDRAM 的运行速度，我们一般采用不让 SDRAM 进入”WRITEA”状态来提高速度。当然”READ”和”READA”这两个状态的区别也是这样的。

下面我们继续上一张写操作的时序图，这个时序图是不进入”WRITEA”状态的：

WRITE - WITHOUT AUTO PRECHARGE



Notes:
1) Burst Length = 4
2) x16: A9, A11, and A12 = "Don't Care"
x8: A11 and A12 = "Don't Care"
3) tRAS must not be violated.

按照老套路，Kevin 继续和大家一起来分析写操作的时序：

在初始化完成后的 “tMRD” 之间之后，给一个”ACT”命令，同时指定是哪一个 bank 的哪一行，然后再经过 “tRCD”的时间给“WRITE”指令，同时指定是 bank 地址和列地址并把 A10 拉低，然后再根据设定好的突发长度将对应数量的数据写进去，这样就完成了我们的写操作。写完四个数据之后干嘛呢，这个就完全是我们自己说了算，如果我不给 SDRAM 发任何指令，那 SDRAM 内部还是处于“WRITE”状态的，如果我还想让 SDRAM 进行写，那我可以在给 SDRAM 发送写指令，如果此时刷新 SDRAM 的时间到了，那我们就必须去执行 SDRAM 的刷新操作来保证 SDRAM 的数据不被丢失。

可能看到这里大家会想，如果我正在让 SDRAM 写数据，是不是 SDRAM 刷新的时间到了，我就必须是让 SDRAM 马上执行刷新操作吗？这样的话肯定不是现实的。大家试想一下，SDRAM 写数据是每四个每四个数据进行的，如果正在写第二数据却马上让 SDRAM 执行刷新操作，那必然会把还没写的剩下的两个数据丢失，On My God，我们是断然不能让我们的数据丢失的。不能让我们的数据丢失，又要保证 SDRAM 进行刷新来保证我们整个

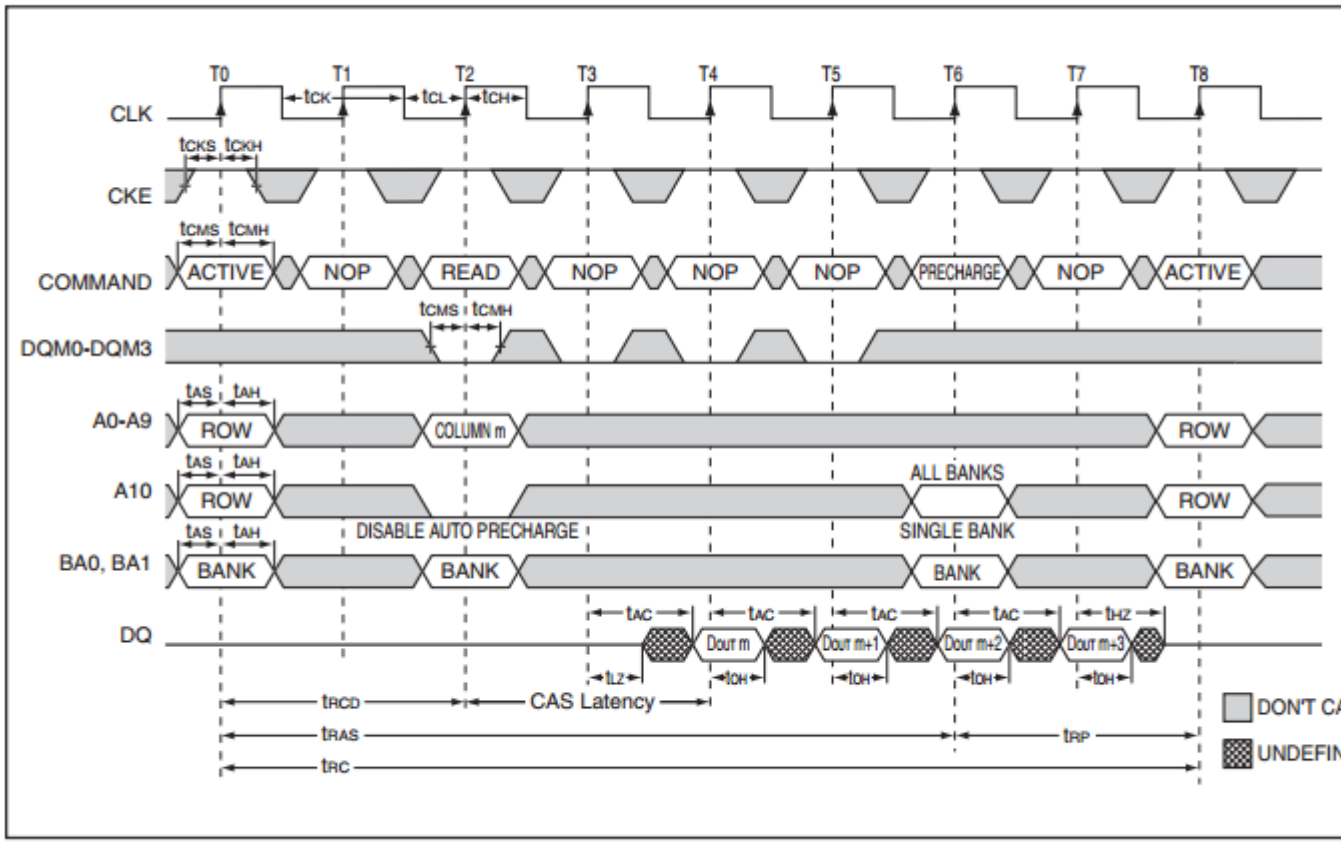
SDRAM 相应 BANK 中的数据不被丢失，我们应该怎么来写代码呢？天啊，想到这里，是不是已经有朋友头都要快炸了呢？大家不要慌，关于如何写代码，Kevin 会在下一篇博文中进行讲解的，大家先耐着性子把 SDRAM 的时序彻底弄明白。关于刷新的时序，会在这篇文章的后续部分进行介绍。

写操作到这里就已经基本上说完了。

SDRAM 读操作

下面来说说我们的读操作，其实读操作和写操作是类似的，相信大家有了写操作的基础，对于 SDRAM 的读操作理解起来还是会比较轻松的，咱们继续上图：

READ WITHOUT AUTO PRECHARGE



CAS latency = 2, Burst Length = 4

首先，依然要给“ACT”命令，记住不要忘了指定 ROW 地址和 BANK 地址，然后在给“READ”指令，经过设定好的潜伏期之后，咱们的数据就出来了。大家也不要忘了，在给相关命令的时候，千万不要忘记其他信号线上的信号。

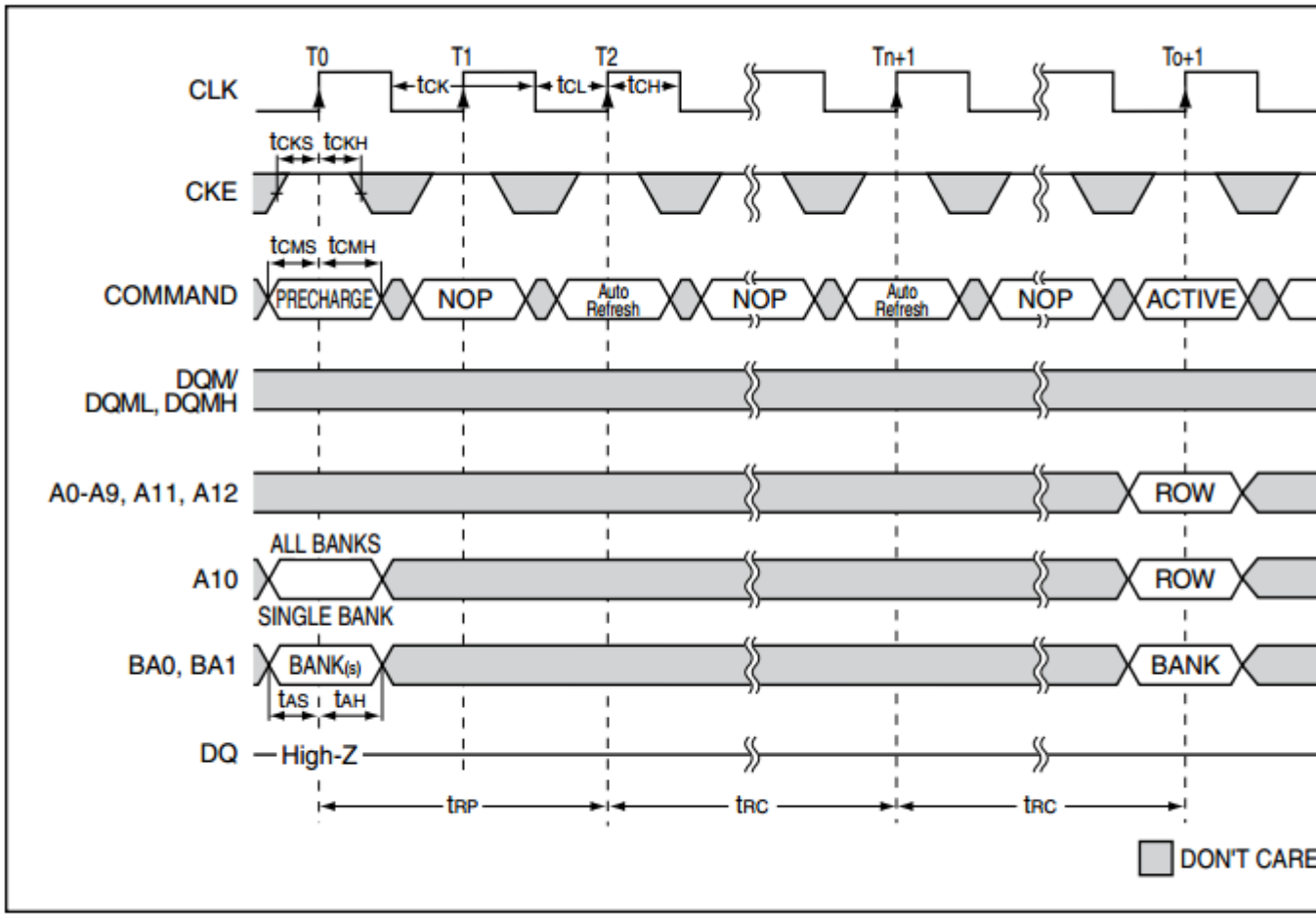
SDRAM 的自动刷新操作

Kevin 已经在前面多次提到了 SDRAM 的自动刷新，SDRAM 为什么要自动刷新，相信大家通过前面的介绍已经知道原因了，可以说如果 SDRAM 到了该刷新的时间没有进行刷新，那不管我们给 SDRAM 写进去了多少数据都是白搭，所以大家一定要高度重视 SDRAM 的刷新操作，到了该刷新的时候就一定要给刷新的命令。SDRAM 内部电容保存数据的最长时

间是 64ms，而我们一个 BANK 有 4096 行，64ms/4096～=15us，也就是说为了保证 SDRAM 内部的数据不被丢失，两次刷新之间的最大时间间隔为 15us，所以为了能让 SDRAM 有更多的时间进行读或者写，我们就设定 SDRAM 刷新的周期为 15us.

在这里 Kevin 还想补充一个小知识，SDRAM 每进行一次刷新，是对每一行进行操作的，并不是单独针对每一个电容进行充电，所以每进行一次刷新，该行中的电容进行充电我们可以理解为是同步发生的。

AUTO-REFRESH CYCLE



Notes:
1. CAS latency = 2, 3

这就是 SDRAM 进行自动刷新的时序图了，相信大家有了前面的基础，对于这个图应该不会害怕了吧。

在每次自动刷新时，我们需要给一个“Precharge”命令，这个命令有什么作用呢？大家可以看下在前面的那张状态图，如果此时 SDRAM 正处于“WRITE”或“READ”状态时，这个“Precharge”命令可以使 SDRAM 跳出“WRITE”或“READ”状态从而进入“IDLE”状态。接下来，经过“tRP”的时间，给一个“Auto-Refresh”命令，注意，我们这里只需要给一次“Auto-Refresh”命令**（在电子发烧友论坛和正点原子论坛中的帖子中，Kevin 写成了给两次“Auto-Refresh”命令，是错误的，希望看到了这篇帖子的朋友注意一下）**，至此，自动刷新操作就完成了。关于这里的自动刷新操作，和我们的初始化过程有点类似，只是在这里没了模式寄存器的设置。

好了，对于 SDRAM 的初始化、读、写以及自动刷新，Kevin 在这里根据自己的理解已经全部讲完了，如果在这篇文章中出现了某些错误，还希望您及时批评指出。另外，Kevin 也会在接下来的一篇博文中分享怎么来写 SDRAM 的控制器，也就是怎么来把这篇文章中的这些操作用代码写出来。