SC6800H EMC 及 SDRAM 专题培训

Version: 3.2

Date: 2011-07-08

GSM FAE SZ

Yongjun.Zang

www.spreadtrum.com

前　言

文档说明

本文档为展讯 SC6800H EMC 及 SDRAM 专题培训资料。

阅读对象

本文档针对于所有 SC6800H 平台驱动工程师及硬件工程师。

内容介绍

本文档包括四个章节，分别为：

- 第一章：SDRAM 工作原理和时序分析
- 第二章：SC6800H EMC 简介
- 第三章：SC6800H EMC 寄存器介绍
- 第四章：SC6800H SDRAM 客户化配置
- 第五章：SC6800H EMC 常见问题分析

问题反馈

本文档涉及的内容比较多，难免有错误和遗漏之处。如果有任何的疑问，请联系 yongjun.zang@spreadtrum.com。

目　录

第1章 SDRAM 工作原理和时序分析

1.1 常见 RAM 分类

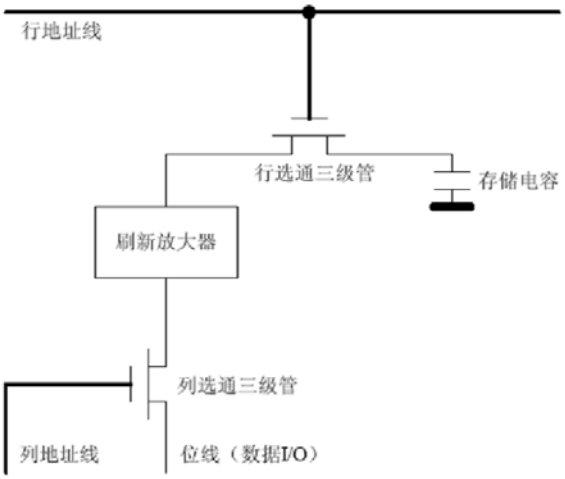
SRAM：Statiic Random Access Memory（静态随机存储器）。SRAM 在加电情况下不需要刷新，数据也不会丢失。一般不是行列地址复用的。

DRAM：Dynamic Random Access Memory（动态随机存储器）。DRAM 需要不断的刷新，才能保证数据的有效性，而且是行列地址复用的。

SDRAM：Synchronous DRAM（同步动态随机存储器）。SDRAM 的读写需要时钟来同步。

随机：指可以自由指定地址进行数据的读写。

动态：指需要不断的刷新来保证数据不丢失。DRAM 的存储电容（如下图）会漏电和缓慢放电，所以需要经常刷新来保存数据。



同步：指其时钟频率与 CPU 前端总线的时钟频率相同，并且其内部命令的发送和数据的读写都以它为基准。

性能：SRAM 中的一个存储位（bit）通常需要六个 MOSFET，对称的电路结构使得 SRAM 的访问速度要快于 DRAM。SRAM 比 DRAM 访问速度快的另外一个原因是 SRAM 可以一次接收所有的地址，而 DRAM 则使用行列地址复用的结构。

1.2 常见 SDRAM 分类

SDRAM 不断的发展演化，出现了以下几种产品形态，分别是：

SDR SDRAM：Single Data Rate。工作电压 3.3V

DDR SDRAM：Double Data Rate。2bit 预读（X2 SDR）。工作电压 2.5V。

DDR2 SDRAM：4bit 预读（X4 SDR）。工作电压 1.8V。

DDR3 SDRAM：8bit 预读（X8 SDR）。工作电压 1.5V。

SDR 与 DDR 均采用单端的时钟信号。DDR2 与 DDR3 由于工作频率比较快，所以采用可降低干扰的**差分时钟信号**作为同步时钟。

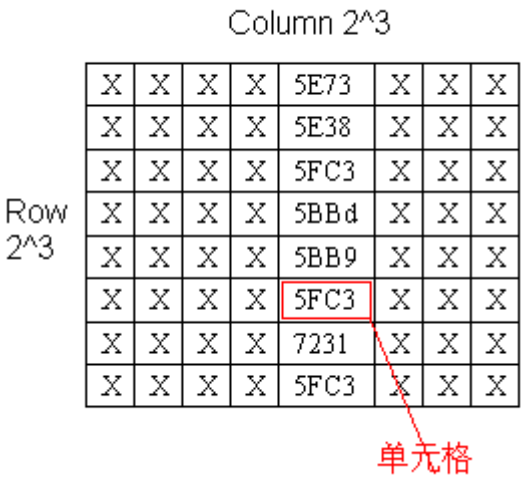
DDR 最重要的改变是在数据传输上，其在时钟信号上升缘与下降缘时各传输一次数据，这使得 DDR 的数据传输速率为传统 SDRAM 的两倍。

SDR 的时钟频率等于数据存储的频率，所以用时钟频率命名。之后的 DDR 则采用数据读写速率作为命名标准。

1.3 SDRAM 的重要概念

1.3.1 L-Bank

SDRAM 是存储阵列的集合。存储阵列如同表格一样，要找到所需要的单元格，先指定一个行（Row），再指定一个列（Column）。这个单元格称为存储单元，这个存储阵列就叫逻辑 Bank（Logical Bank，简称 L-Bank）。



由于 SDRAM 工作原理的限制，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。所以 SDRAM 内部会有多个 L-Bank。目前基本都是 4 个，这也是 SDRAM 规范中的最高 L-Bank 数量。

1.3.2 位宽

L-Bank 存储阵列中一个存储单元的容量就是芯片的位宽。这也是 SDRAM 一次传输的数据量。在某些厂商的表述中，将 L-Bank 中的存储单元称为 Word。如上图的红色部分，存储的是 0x5FC3 一个 16bit 的数字。

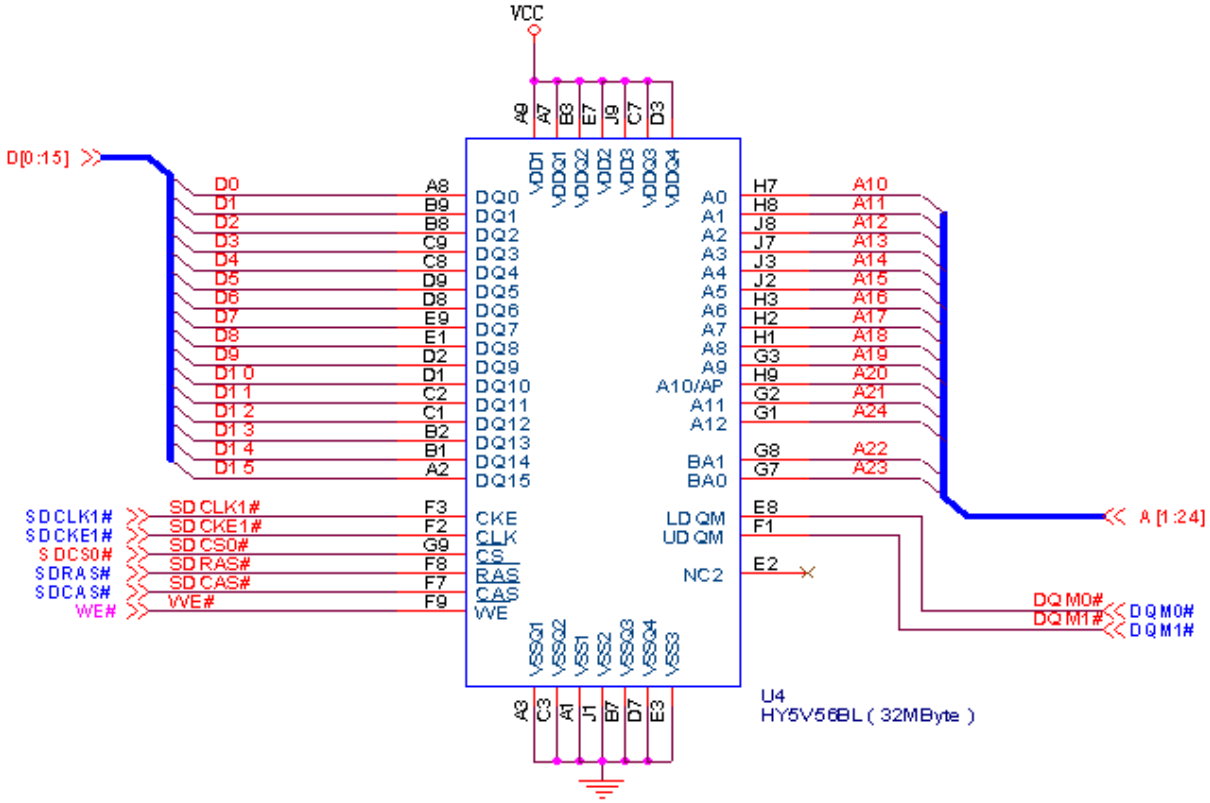
1.3.3 存储容量

SDRAM 的容量 = 2^行数 × 2^列数 × L-Bank 的数量。

如 SC6800H 项目中常用的 row=13, column=9, bank=4, data width=16 的 SDR SDRAM，

其容量=2^13 x 2^9 x 4 x 16 / 1024 / 1024 / 8 = 256MBits = 32MBytes

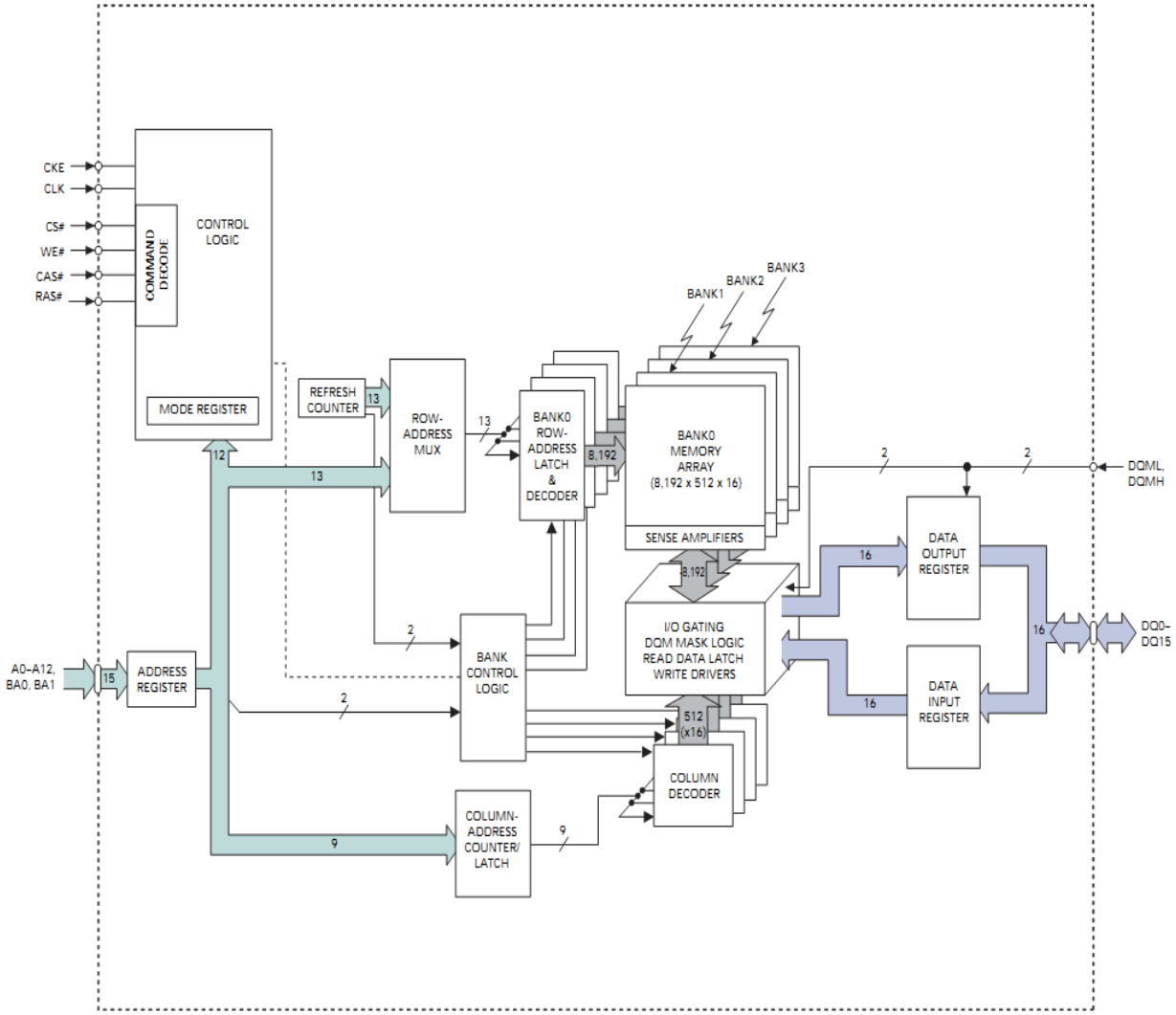
1.4 SDRAM 的外部信号



各外部信号的描述（其中#表示低电平有效）：

CLK	时钟
CKE	时钟有效
CS#	片选
BAn	L-Bank 地址线
CAS#	列地址选通脉冲
RAS#	行地址选通脉冲
An	行列地址线
DQn	数据 I/O 线
DQM	数据掩码
WE#	写允许
Vdd/ VddQ	工作/DQ 电压
Vss/ VssQ	工作/DQ 电压接地

1.5 SDRAM 的内部结构



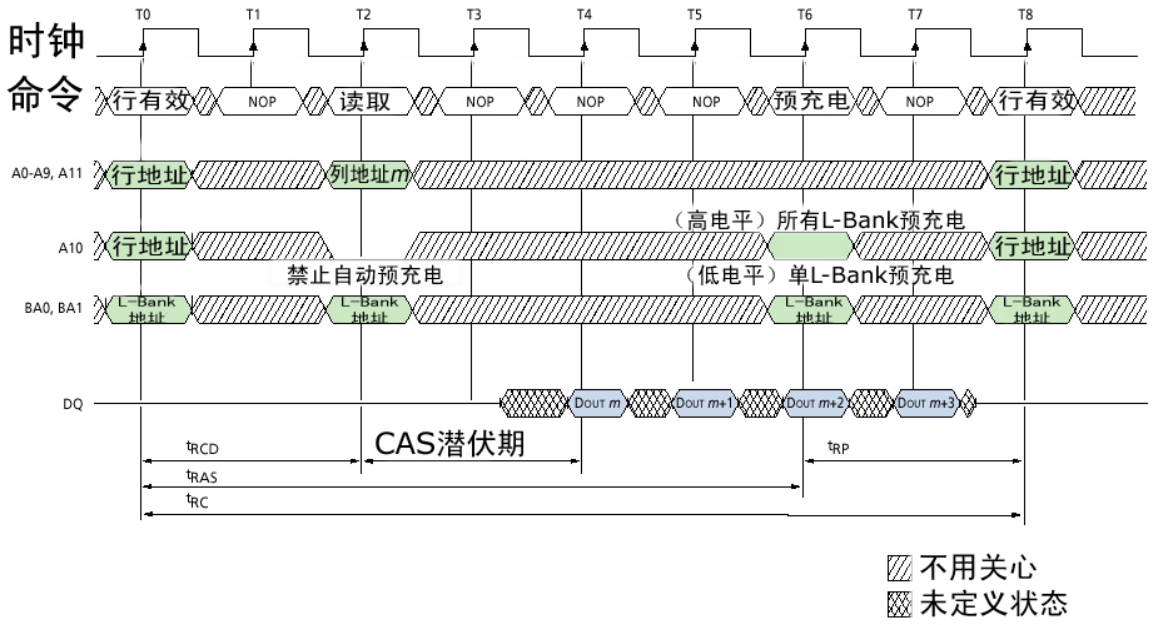
1.6 SDRAM 的基本命令

SDRAM 的基本命令是通过操作各种控制信号/地址信号的组合来完成。下表中除了自刷新命令外，所有命令都是默认 CKE 有效（其中 H 代表高电平，L 代表低电平，X 表示高低电平没有影响）。

Table 3-11 Device Command Truth Table

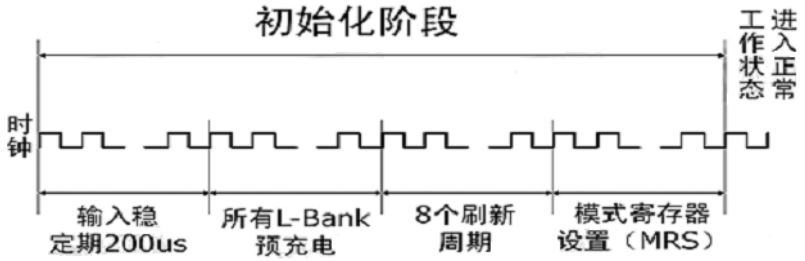
Command	CS#	CKE	RAS#	CAS#	WE#	BA	ADDR
NOP	H	X	X	X	X	X	X
NOP	L	H	H	H	H	X	X
ACTIVE	L	H	L	H	H	bank	row
READ	L	H	H	L	H	bank	column
WRITE	L	H	H	L	L	bank	column
BURST TERMINATE	L	H	H	H	L	X	X
PRECHARGE (one bank)	L	H	L	H	L	bank	A10 low
PRECHARGE (all bank)	L	H	L	H	L	X	A10 high
AUTO REFRESH	L	H	L	L	H	X	X
SELF REFRESH	L	L	L	L	H	X	X
LOAD MODE REGISTER	L	H	L	L	L	Op-Code	

下图为一个完整的从行寻址到行关闭的时序图（其中 tRCD=2、CL=2、tRP=2）。



1.6.1 Initialize

在 SDRAM 内部有一个逻辑控制单元，并且有一个模式寄存器（Mode Register）为其提供控制参数。每次开机时 SDRAM 都要先预充电和对这个逻辑控制单元进行初始化（即设置模式寄存器设置）。



1.6.2 Load Mode Register

模式寄存器设置在 CPU 控制下进行，寄存器的信息由地址线来提供。在设置完模式寄存器之后，就进入了正常的工作状态。

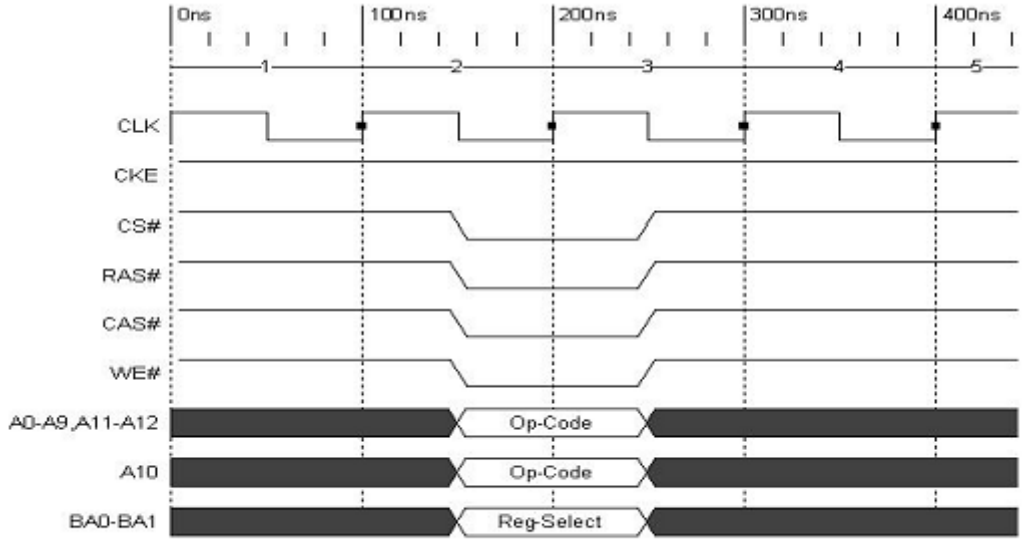


Figure 3-4 Waveform of LOAD MODE REGISTER Command

模式寄存器主要用于设置 Burst Type，Burst Length，CAS Latency 等信息。见下图所示。

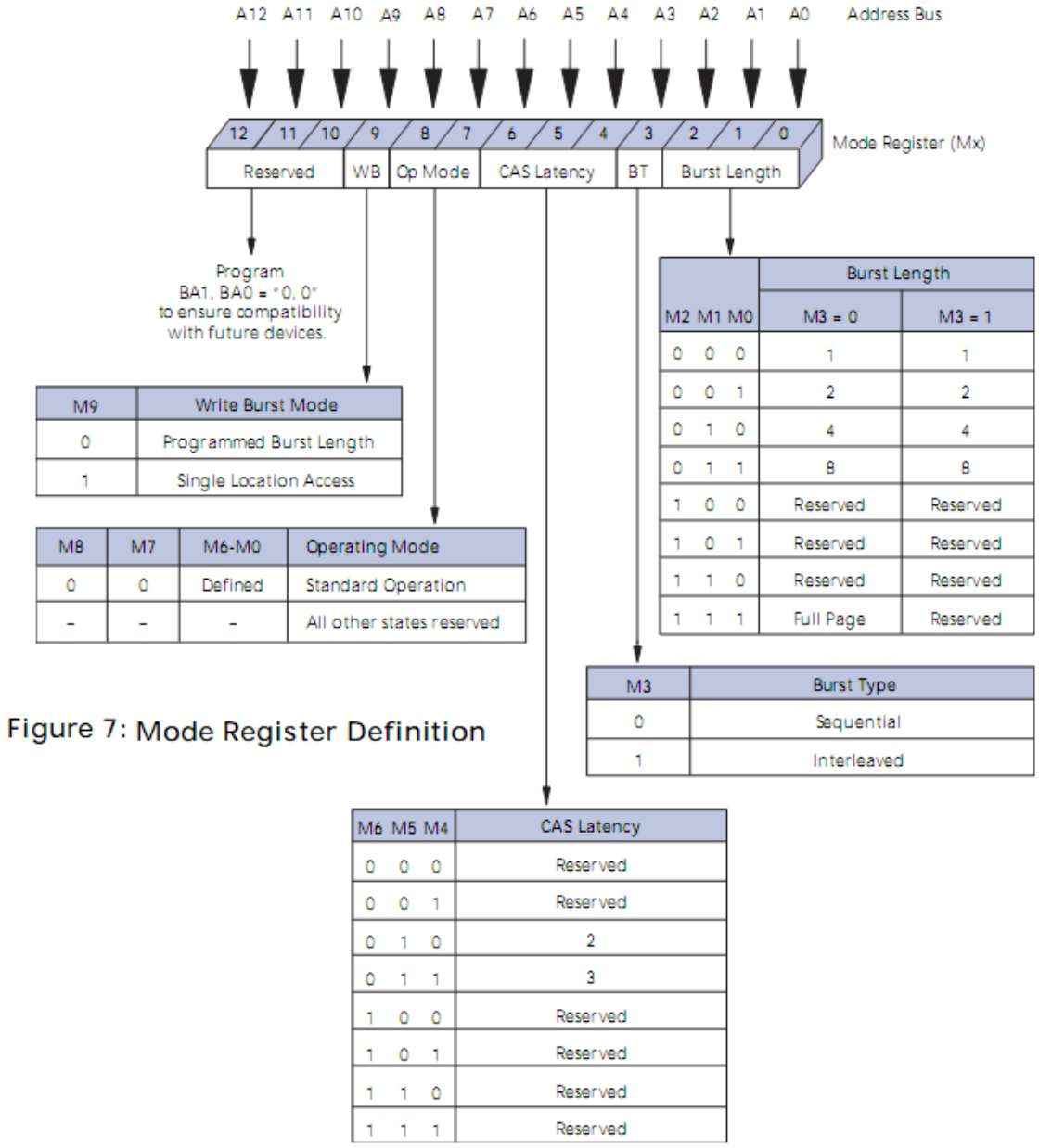


Figure 7: Mode Register Definition

1.6.3 Row Active

要对 L-Bank 中的存储阵列中的某个存储单元进行寻址，首先就要找到所在的行(Row Active)，即行有效。片选和 L-Bank 的寻址与 Active 同时进行。

从行有效时序图中可以看出，在 CS#，L-Bank 有效的同时，RAS#（Row Address Strobe，行地址选通脉冲）也处于有效状态，即此时 An 地址线发送的是行地址。

由于行有效时相应的 L-Bank 也有效，所以行有效也可称为 L-Bank 有效。

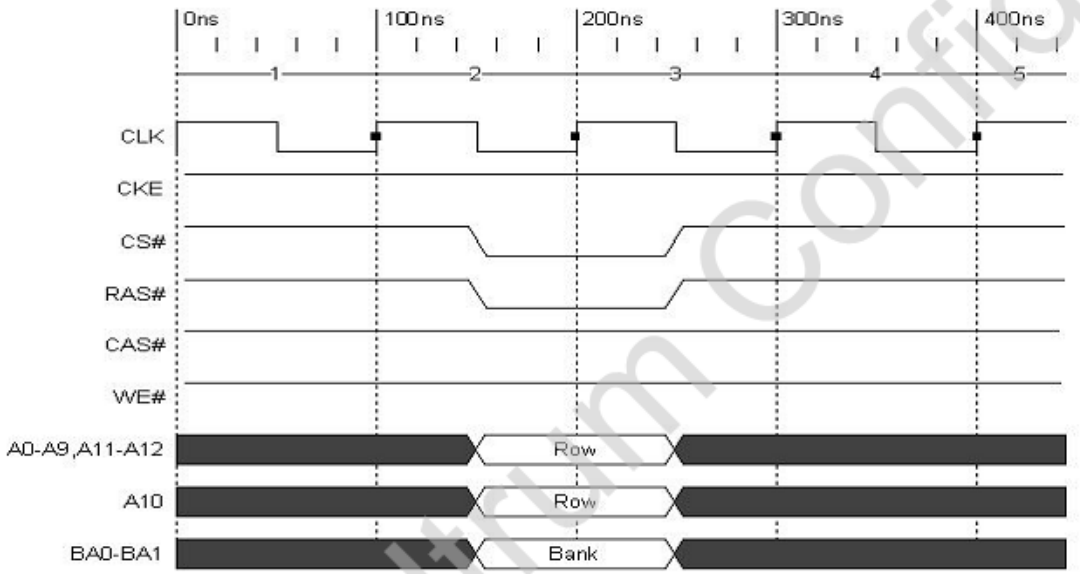


Figure 3-5 Waveform of ACTIVE Command

1.6.4 Column

在某个存储单元的行地址确定之后，就要对其列地址进行寻址了。因为在 SDRAM 中行地址与列地址线是复用的，所以由 CAS#（Column Address Strobe，列地址选通脉冲）决定 An 地址线发送的是列地址。

读/写命令是通过 WE#信号的控制来达到目的。当 WE#为低电平是即为写命令，WE#为高电平就是读取命令。列寻址信号与读写命令是同时发出的。在读写操作时，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。

下文将分开讨论 Read 和 Write 命令操作。

下面是一个读写操作示意图，读命令与列地址一块发出。

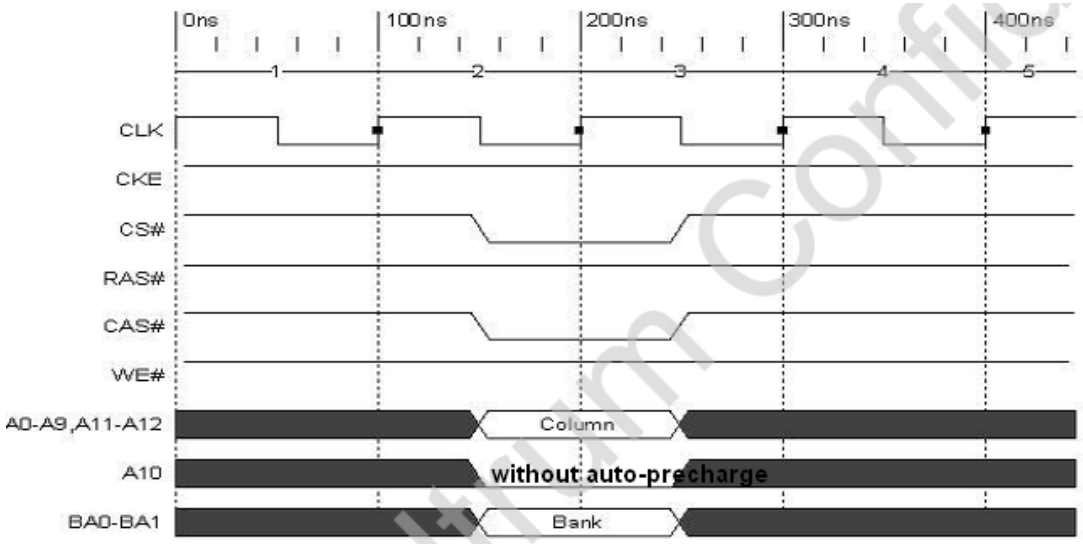
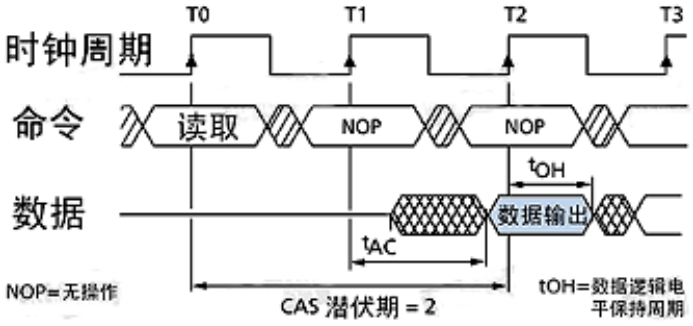


Figure 3-11 Waveform of READ Command

1.6.5 Read

在选定列地址后，就已经确定了具体的存储单元，但要经过一定的时间（存储体中晶体管的反应时间），数据通过 DQ 输出到内存总线上。

从 CAS#与读取命令发出到数据输出的时间为 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（Read Latency）。CL 的单位为时钟周期数，在开机初始化过程中的 Load Mode Register 阶段进行设置。



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 S-AMP 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，此时数据重写操作则可在预充电阶段完成。

1.6.6 Write

写操作的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。如下图：

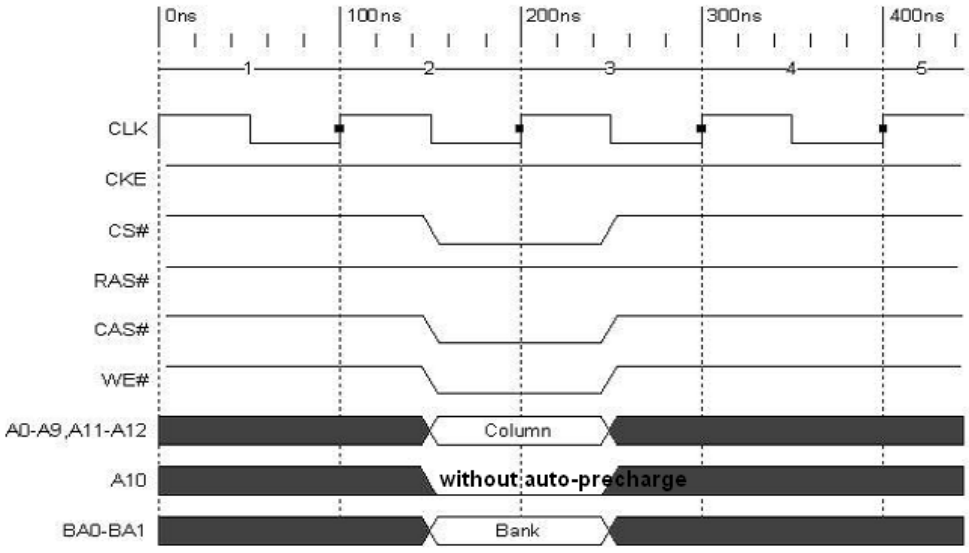


Figure 3-12 Waveform of WRITE Command

由于数据由主控制端发出，输入时 SDRAM 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，即没有了 CL。

因为选通三极管与电容的充电必须要有一段时间，所以数据写入存储电容需要一定的时间。为了保证数据的可靠写入，都会留出足够的写入/校正时间 tWR（Write Recovery Time），这个操作也被称作写回（Write Back)。

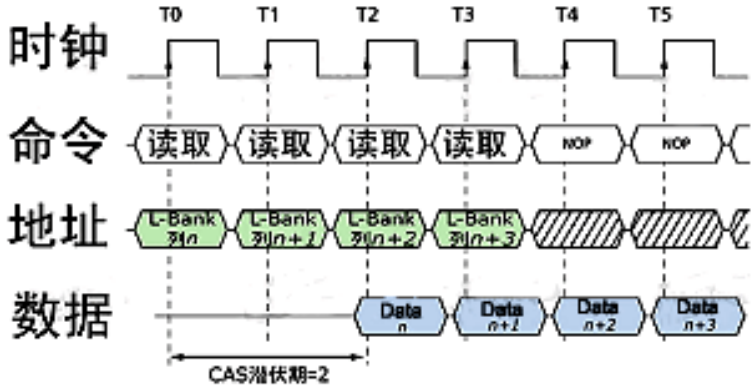
1.6.7 Burst

突发操作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，连续传输所涉及到存储单元的数量就是突发长度（Burst Lengths，简称 BL）。

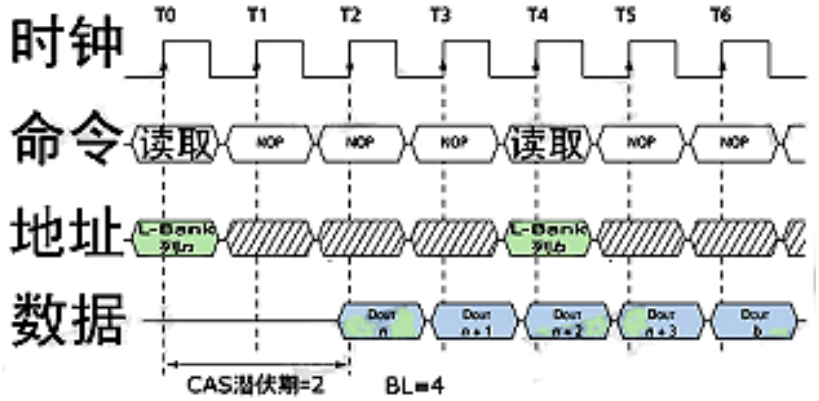
上文讲到的读/写操作，都是一次对一个存储单元进行寻址，如果要连续读/写就还要对当前存储单元的下一个单元进行寻址，也就是要不断的发送列地址与读/写命令（行地址不变，所以不用再对行寻址）。虽然由于读/写延迟相同可以让数据的传输在 I/O 端是连续的，但它占用了大量的内存控制资源，在数据进行连续传输时无法输入新的命令，效率很低。

为此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（一般的是 tRCD+CL）外，其后每个数据只需一个周期的即可获得。

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。如下图：



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。如下图：



在初始化过程中的 Load Mode Register 阶段需要设置 BL 的数值和确定读/写操作的模式以及突发传输的模式。

突发读/突发写：读与写操作都是突发传输的，这也是常规的设定。

突发读/单一写：读操作是突发传输，写操作则只是一个个单独进行。

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：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：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）

1.6.8 Precharge

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（A10）让 DRAM 在每次读写操作之后自动进行预充电。

预充电是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。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。

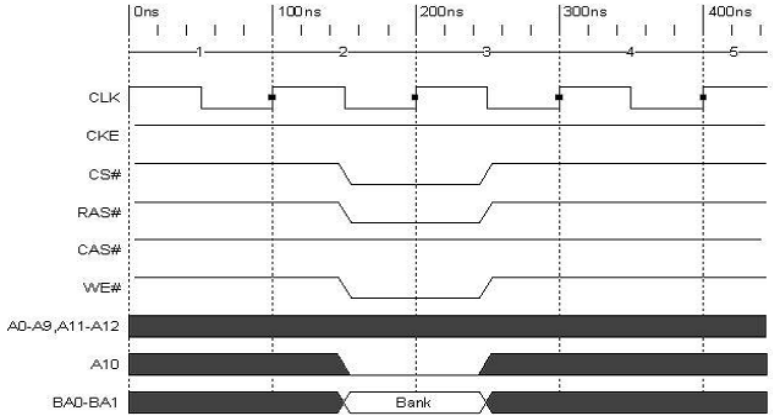


Figure 3-6 Waveform of PRECHRGE(one bank) Command

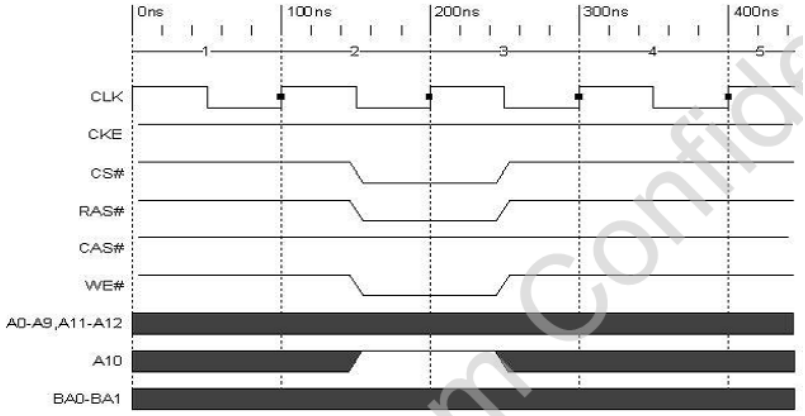
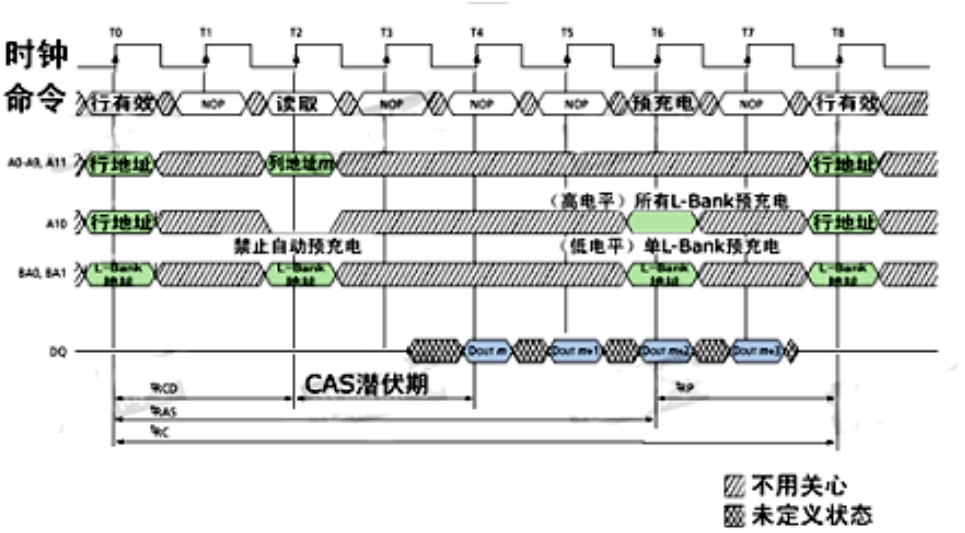


Figure 3-7 Waveform of PRECHARGE(all bank) Command

在预充电命令中，A10 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）。tRP 的单位也是时钟周期数。

下图为读取时预充电时序图，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。



1.6.9 Auto Refresh

DRAM 要不断进行刷新才能保留住数据，因此刷新是 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。预充电和刷新区别:

1．刷新则是有固定的周期，依次对所有行进行操作。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。

2．刷新时进行操作的行是指所有 L-Bank 中地址相同的行。预充电中各 L-Bank 中的工作行地址并不是一定是相同的.

刷新操作分为两种：自动刷新（Auto Refresh）与自刷新（Self Refresh）。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 Auto Refresh，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。

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，之后进入正常的工作状态，也就是说在这期间所有工作指令只能等待而无法执行。

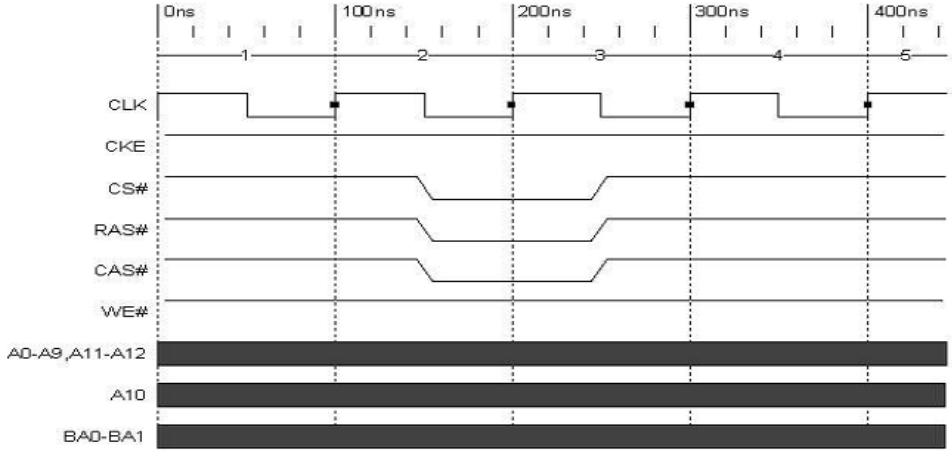


Figure 3-8 Waveform of AUTO REFRESH Command

1.6.10 Self Refresh

Self Refresh 则主要用于休眠模式低功耗状态下的数据保存。

在发出 Self Refresh 命令时，将 CKE 置于无效状态，就进入了相应的模式，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 Self Refresh 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。

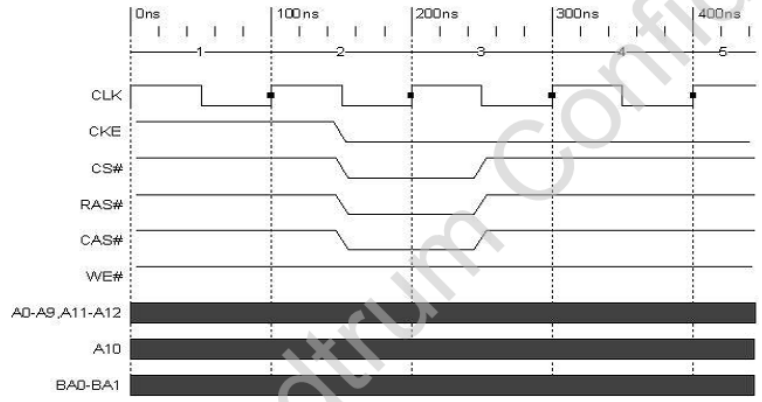


Figure 3-9 Waveform of Enter SELF REFRESH Command

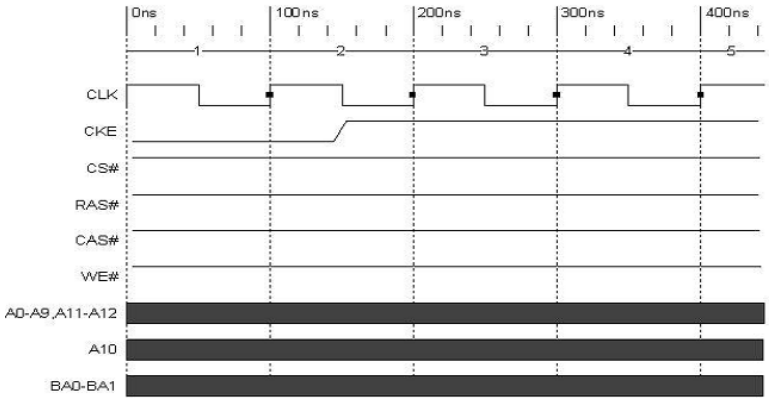


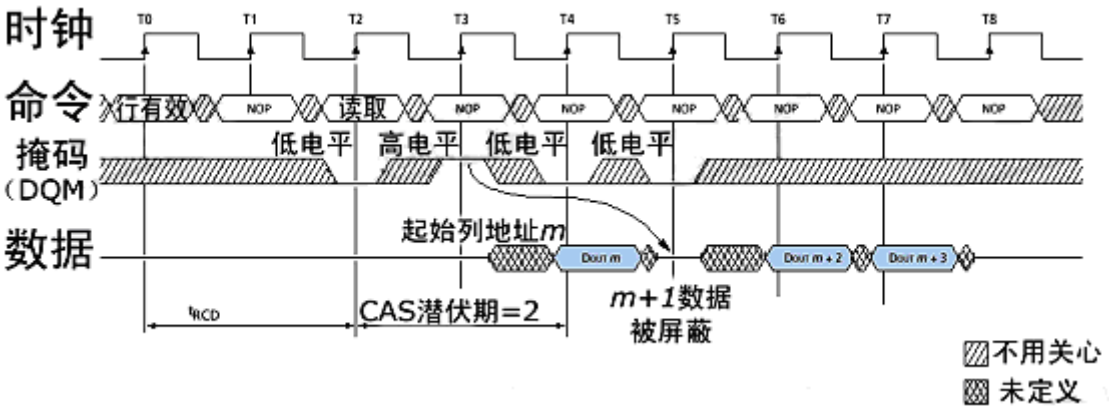
Figure 3-10 Waveform of Exit SELF REFRESH Command

1.6.11 DQM

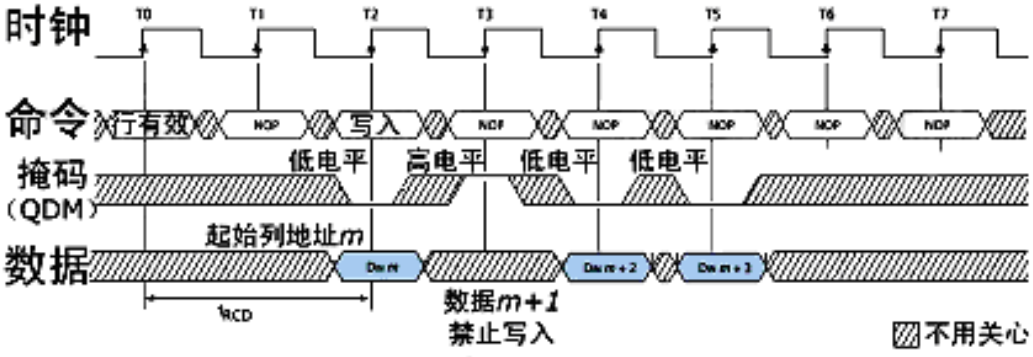
为了屏蔽不需要的数据，采用了数据掩码（简称 DQM）技术。内存可以通过 DQM 控制 I/O 端口取消哪些输出或输入的数据。这里需要强调的是，在读取时被屏蔽的数据仍然会从存储体传出，只是在掩码逻辑单元处被屏蔽。

DQM 由 CPU 控制，每个信号针对一个字节。所以对于 8bit 位宽芯片需要一个 DQM 信号，而对于 16bit 位宽芯片则需要两个 DQM 引脚。

读取时，DQM 在两个周期后生效，突发周期的第二笔数据被取消。如下图：



写入时，DQM 立即生效，突发周期的第二笔数据被取消。如下图：



1.7 SDRAM 的时序参数

1.7.1 tMRD(Mode Register Delay)

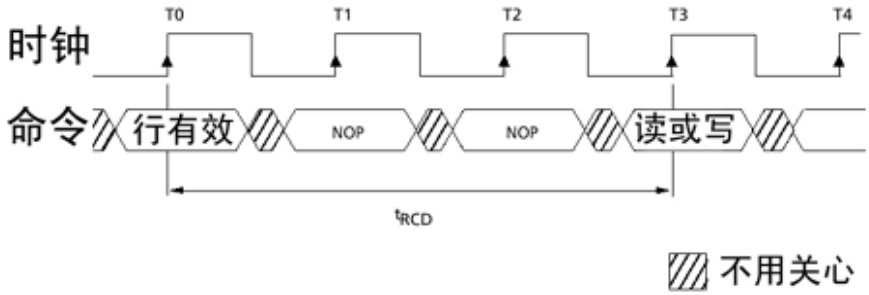
表示在进行初始化时，从 LOAD MODE REGISTER 命令发出到可以执行 ACTIVE / REFRESH 命令的时间。

1.7.2 tRP(Row Precharge Time / PRECHARGE to ACTIVE delay)

表示行预充电时间。tRP 用来设定在另一行能被激活之前，现有的工作行需要的充电时间。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 tRP。

1.7.3 tRCD(RAS to CAS Delay / ACTIVE to READ/WRITE delay)

表示行寻址到列寻址延迟时间。从行有效到读/写命令发出之间的间隔被定义为 tRCD，也叫行选通周期。这是根据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。tRCD 是排第二的重要时序参数。



1.7.4 tCL(CAS Latency)

CAS latency 表示内存读写操作前列地址选通的潜伏时间。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（数据的触发有延迟，而且还需要进行信号放大）。

这个参数控制内存接收到一条数据读取指令后要等待多少个时钟周期。同时该参数也决定了在一次内存突发传送过程中完成第一部分传送所需要的时钟周期数。在保证稳定性的前提下，CAS 值越小，则内存的速度越快。

CAS 是找到数据的最后一个步骤，也是内存参数中最重要的参数。

1.7.5 tWR(Write Recovery Time)

表示写恢复延时。指在一个激活的 bank 中完成有效的写操作及预充电前，必须等待多少个时钟周期。

这段必须的时钟周期用来确保在预充电发生前，写缓冲中的数据可以被写进内存单元中。过低的 tWR 虽然提高了系统性能，但可能导致数据还未被正确写入到内存单元中，就发生了预充电操作，会导致数据的丢失及损坏。

1.7.6 tRAS(Row Active Time / Active to Precharge Delay)

表示从内存行有效命令发出至预充电命令发出之间的最短时间。

如果 tRAS 的周期太长，系统会因为无谓的等待而降低性能。如果 tRAS 的周期太短，则可能因缺乏足够的时间而无法完成数据的突发传输，这样会引发丢失数据或损坏数据。

1.7.7 tRRD(Row to Row Delay / RAS to RAS delay / ACTIVE to ACTIVE delay)

表示行单元到行单元的延时。该值也表示向相同 bank 中的同一个行单元两次发送激活指令之间的时间间隔。tRRD 值越小越好，延迟越低表示下一个行能更快地被激活，进行读写操作。

1.7.8 tWTR(write to read turn around delay)

表示读到写延时。在最后一次有效的写操作和下一次读操作之间必须等待的时间。

1.7.9 tREF(Row Refresh Time)

表示行刷新所需要的时间。

1.7.10 tRFC(AUTO REFRESH Command period)

表示 Auto Refresh 所需要的时间。

1.7.11 tXSR(exit SELF REFRESH to ACTIVE command time)

表示 Exit Self Refresh 到 Row Actice 需要的时间。

1.8 SDRAM 的性能分析

1.8.1 SDRAM 时序对性能的影响

有三个参数对内存的性能影响至关重要：tRCD、CL 和 tRP。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。写操作不用考虑 CL。

现在分析一下读取时可能遇到的几种情况：

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，这种情况称之为页快速命中（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。总耗时就是 tRP+tRCD+CL，这种情况称之为页错失（PM，Page Miss）。

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)。因此尽量提高 PHR 与 PFHR，同时减少 PMR，以达到提高内存工作效率的目的。

1.8.2 SDRAM 减少延迟的方法

由于 Bank 内的行与行之间具有关联性，因此当其中一个 Bank 的读或写操作结束后，必须执行一次预充电命令以关闭正在操作的行。预充电命令执行后，会有 tRP 的延时，延时完成后才能向同一 Bank 的其它行发出新的激活命令。

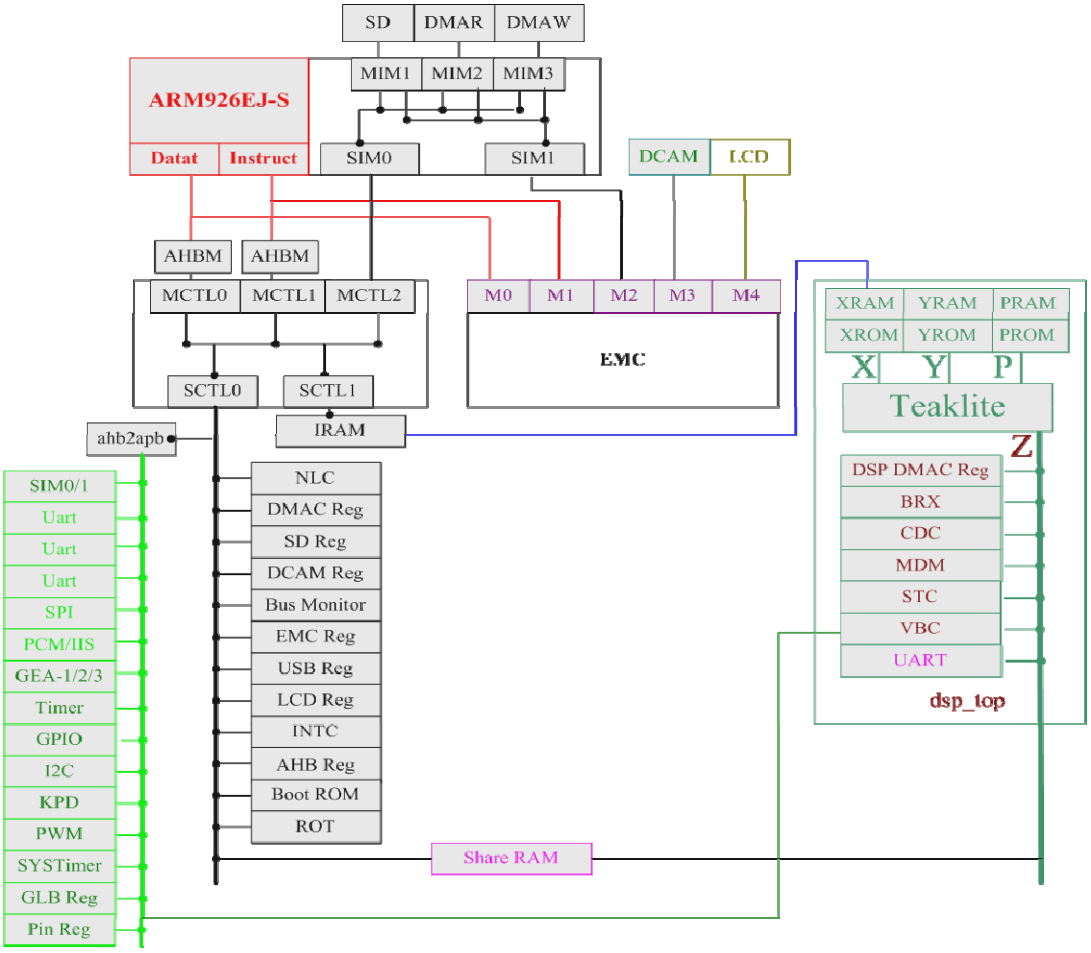
自动预充电技术是一种有效的减少延迟的方法，它通过自动在每次行操作后进行预充电操作来减少对同一 Bank 内的不同行寻址时发生冲突的可能性。

但如果要在正在读写的行完成操作后马上打开同一 Bank 的另一行时，仍然存在 tRP 的延迟。交错式控制是另一种更有效的减少延迟的方法，即在一个 Bank 工作时，对另一个 Bank 进行预充电或寻址，预充电和数据的传输交错执行，当访问下一个 Bank 时，tRP 已过，这样就可以直接进入行有效状态，如果配合的比较理想，那么就可以实现无间隔的 Bank 交错读写。

第2章 SC6800H EMC 简介

2.1 SC6800H 内部架构

下图为 6800H 芯片内部结构图。从中可以看到，EMC 等高速设备挂在 AHB 总线上，其它慢速设备挂在 APB 总线上。ARM 通过 AHB，APB 总线访问各种外部设备。



2.2 SC6800H EMC Features

6800H EMC 是一个有多个 AHB channels 的外部存储器控制器(external memory controller)。

支持 16-bit SDR-SDRAM。

支持 16-bit NOR-Flash, pSRAM, SRAM, MCU 接口的 LCM, 其它 SRAM-timing 兼容设备。

因为 6800H 目前使用存储设备的以 NAND+SDR-SDRAM 为主。所以本文主要讲解与 SDR SDRAM 相关的知识。而 SRAM/pSRAM/NOR Flash 部分只简单提下。

2.2.1 SDRAM

2.2.1.1 支持的功能

Support up to 2 CS. And all SDRAM must have the same type.
Support from 16Mbit to 8Gbit SDRAM.
Support 4-bank.
Support 11/12/13/14-bit row width.
Support 8/9/10/11/12-bit column width.
support 2/4/8/16/32-burst;
Support real wrap mode burst. That is, one wrap burst is implemented by one SDRAM burst stead of divided into two SDRAM bursts.
Support 1/1.5/2/2.5/3/3.5 CAS latency.
Support 0/0.5/1/1.5 WRITE latency.
Only support sequential type, not support interleave type.
Support configurable auto-precharge bit location.
Support configurable mode register and extended mode registers.

2.2.1.2 支持的时序参数

tREF: Auto refresh interval time.
tRP: Minimum PRECHARGE(m) to ACTIVE(m) delay.
tRCD: Minimum ACTIVE(m) to READ/WRITE(m) delay.
tRRD: Minimum ACTIVE(m) to ACTIVE(n) delay.
tRC: Minimum ACTIVE(m) to ACTIVE(m) delay.
tRAS: Minimum ACTIVE(m) to PRECHARGE(m) delay.
tWR: Minimum WRITE(m) to PRECHARGE(m) delay.
tRFC/tRC/tARFC: AUTO REFRESH command period.
tXSR/tSREX/tSRFX: Exit SELF REFRESH to ACTIVE command time.
tMRD: LOAD MODE REGISTER command to ACTIVE or REFRESH command delay.
tRTW: Minimum read to write turn around delay.
tWTR: Minimum write to read turn around delay.
tRTR: Minimum different-CS read to read turn around delay.

2.2.2 SRAM

2.2.2.1 支持的功能

Support 3 CS.
Support ADP and ADM.
Only support 16-bit data width.
Read: Asynchronous single / Asynchronous page (length 2/4/8/16/32) / Synchronous burst.
Write: Synchronous single / Synchronous burst.
Support Wrap burst and increase burst / Burst length: 2, 4, 8, 16, 32 and continuous.
Support clk_emc/2 mode and clk_emc/4 mode.
Support Fix latency and variable latency.
Support WAIT/RDY for variable latency.
Support row boundary auto-detecting for fix latency.
Support dedicated CRE/P-MODE.

2.2.2.2 支持的时序参数

Common timing features:
Read to read turn around: 1-cycle, …, 7-cycle
Read to write turn around: 1-cycle, …, 7-cycle
Write to read turn around: 1-cycle, …, 7-cycle
Write to write turn around: 1-cycle, …, 7-cycle

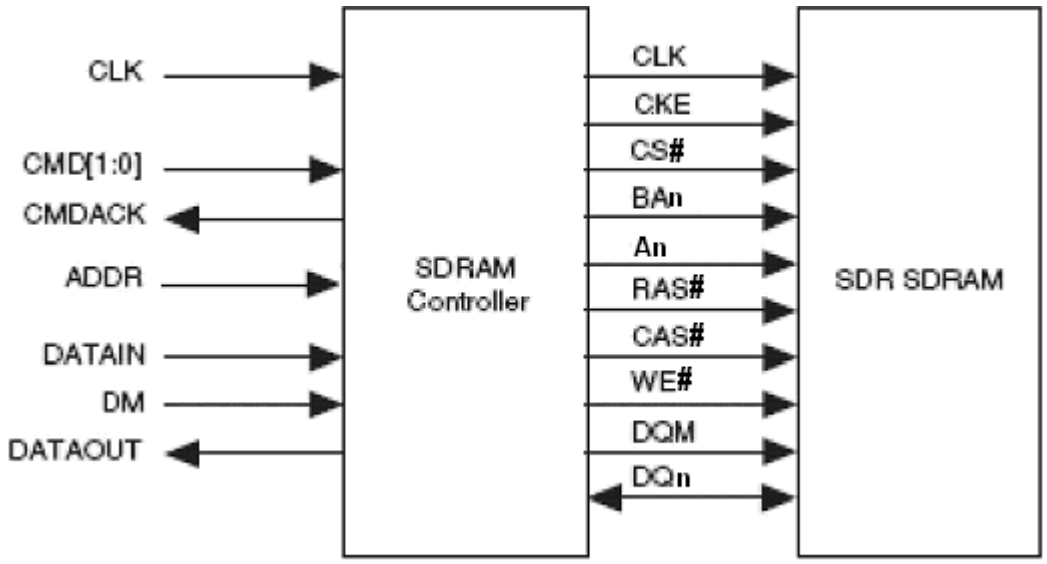
Each CS timing features:
Asynchronous single read time or CE# width: 1-cycle, …, 15-cycle
Asynchronous page read first delay time: 1-cycle, …, 15-cycle
Asynchronous page read next delay time: 1-cycle, …, 15-cycle
Synchronous burst read first delay time: 1-cycle, …, 15-cycle
Synchronous burst read next delay time: 1-cycle, …, 15-cycle
Asynchronous single write time or CE# width: 1-cycle, …, 15-cycle
Synchronous burst write first delay time: 1-cycle, …, 15-cycle
Synchronous burst write next delay time: 1-cycle, …, 15-cycle
ADM: access start to address/data switch point delay time: 0-cycle, 1-cycle, 2-cycle, 3-cycle
Access start to OE# low delay time: 0-cycle, 1-cycle, …, 7-cycle
Access start to WE# low delay time: 0-cycle, 1-cycle, …, 7-cycle
WE# width: 0-cycle, 1-cycle, …, 15-cycle
Access start to CE# low delay time: 0-cycle, 1-cycle
Access start to ADV# low delay time: 0-cycle, 1-cycle
ADV# width: 0-cycle, 1-cycle, 2-cycle, 3-cycle

2.3 SC6800H EMC 外部信号

PAD name	**16-bit SRAM**	DIR	**16-bit SDR**	DIR
EMRSTN_M	**RST#**	O		
EMCLKDP_M	**CLK1**	O	**CLK**	O
EMCLKDM_M	**CLK0**	O		
EMCKE_M			**CKE**	O
EMCSN0_M	**CE#[0]**	O	**CS#[0]**	O
EMCSN1_M	**CE#[1]**	O	**CS#[1]**	O
EMCSN2_M	**CE#[2]**	O	**CS#[2]**	O
EMCSN3_M	**CE#[3]**	O	**CS#[3]**	O
EMRASN_M	**ADV#**	O	**RAS#**	O
EMCASN_M	**OE#**	O	**CAS#**	O
EMWEN_M	**WE#**	O	**WE#**	O
EMDQM0_M	**LB#**	O	**DQM#[0]**	O
EMDQM1_M	**UB#**	O	**DQM#[1]**	O
EMDQM2_M				
EMDQM3_M				
EMDQS0_M				
EMDQS1_M				
EMDQS2_M				
EMDQS3_M				
EMA0_M~ EMA13_M	**A[0]~ A[13]**	O	**A[0]~ A[13]**	O
EMA14_M	**A[14]**	O	**BA[0]**	O
EMA15_M~ EMA28_M	**A[15]~ A[28]**	O		
EMCRE_M	**CRE**	O	**BA[1]**	O
EMWAIT_M	**WAIT**	I/O		
EMD0_M~ EMD15_M	**D[0]~ D[15]**	I/O	**D[0]~ D[15]**	I/O

2.4 常见 SDRAM 控制器简介

2.4.1 SDRAM 控制器外部信号

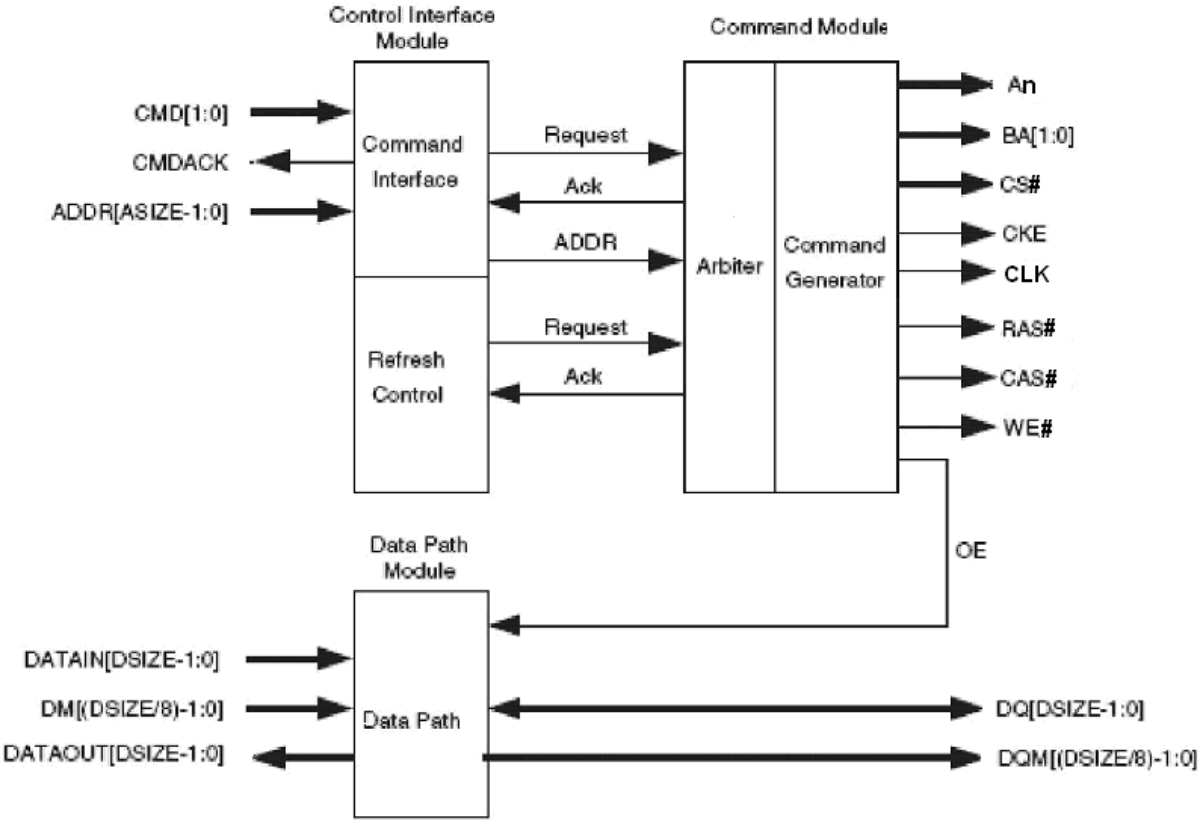


SDRAM 控制器作为 AHB 总线上的一个 slave，其接口必须遵循 AMBA 的规范。

控制器右侧的接口信号为与 SDRAM 对应管脚相连的信号。

控制器左侧的接口信号为与 CPU 相连的系统控制接口信号。其中 CLK 为系统时钟信号，ADDR 为系统给出的 SDRAM 地址信号，DATAIN 是系统用于写入 SDRAM 的数据信号， DATAOUT 是系统用于从 SDRAM 读出数据信号，CMD 和 CMDACK 为系统和控制器的命令交互信号，DM 是数据 Mask 信号。

2.4.2 SDRAM 控制器内部结构



SDRAM 控制器内部结构包括三个模块：系统控制接口模块、命令解析响应模块、数据通路模块。
系统控制接口模块：用于接收系统的控制信号，进而产生不同的命令组合。
命令解析响应模块：用于接收命令并解码成操作指令，并产生 SDRAM 的操作动作。
数据通路模块：用于控制数据的有效输入输出。

2.5 SC6800H DMEM 控制器简介

SC6800H 的 DMEM controller 是一个支持 two-burst 的 SDR SDRAM controller。

因为可以在 SDRAM 不同的 bank 执行并行的操作，所以为了改善性能，DMEM controller 被设计为同时支持 2 个 burst (PRECHARGE/ACTIVE/READ/WRITE)，分时轮流操作。

DMEM controller 内部维护了一个操作时序的状态机，如下图所示：

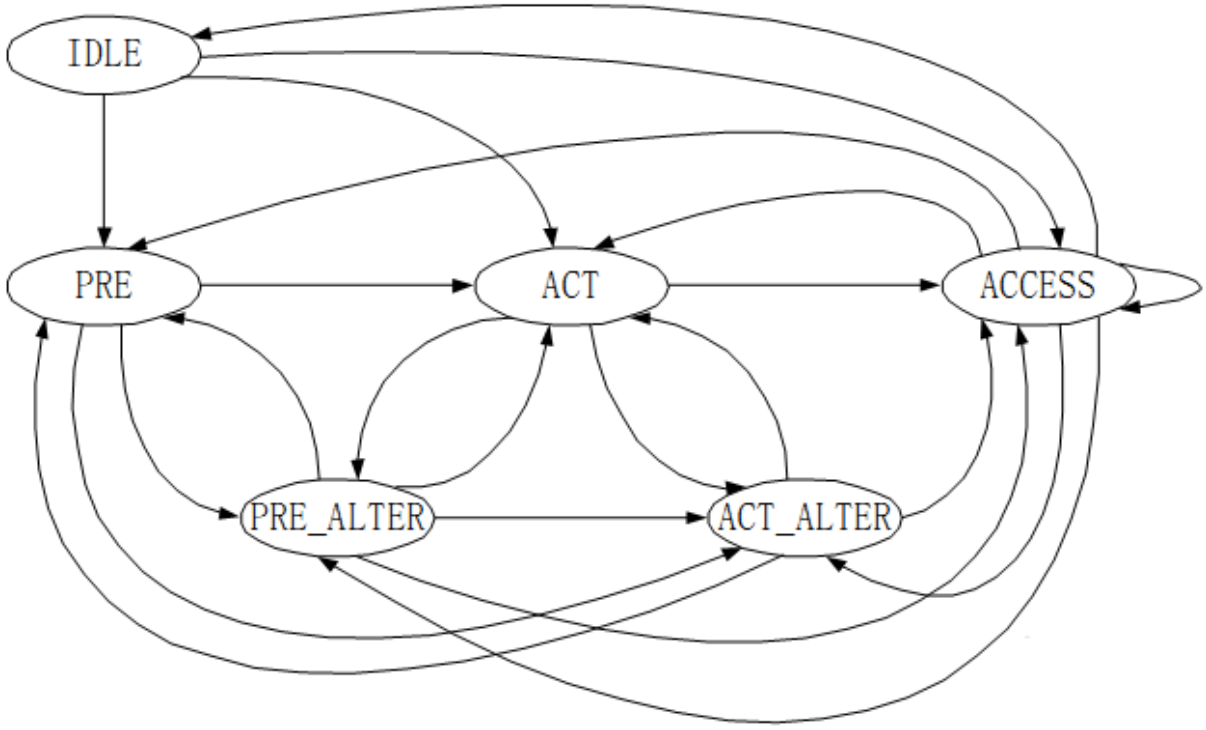


Figure 3-20 DMEM Controller State Transfer

IDLE: idle state
PRE: precharge current burst bank
ACT: active current burst row
PRE_ALTER: precharge next burst bank
ACT_ALTER: active next burst row
ACCESS: access (read or write) current burst

Notes:
1. Not support any burst is terminated / interrupted by any command before it completes. For example, it isn't supported that one burst is terminated by BURST TERMINATE cmmand, or by new READ / WRITE, or by PRECHARGE, etc.
2. Not support CONCURRENT AUTO PRECHARGE.
3. Not support BURST READ / SINGLE WRITE mode.

第3章 SC6800H EMC 寄存器介绍

3.1 寄存器列表

General Control Registers:

Address Offset	Register Name	Default Value
0x0000	EMC_CFG0	32'h00000141
0x0004	EMC_CFG1	32'h0000000C
0x0008 – 0x000C	Reserved	32'h00000000
0x0010	EMC_CFG0_CS0	32'h00000000
0x0014	EMC_CFG0_CS1	32'h00000000
0x0018	EMC_CFG0_CS2	32'h00000000
0x001C	EMC_CFG0_CS3	32'h00000000
0x0020 – 0x002C	Reserved	32'h00000000
0x0030	EMC_CFG0_CH0	32'h0001031C
0x0034	EMC_CFG0_CH1	32'h0001031C
0x0038	EMC_CFG0_CH2	32'h0001031C
0x003C	EMC_CFG0_CH3	32'h0001031C
0x0040	EMC_CFG0_CH4	32'h0001031C
0x0044	EMC_CFG0_CH5	32'h0001031C
0x0048	EMC_CFG0_CH6	32'h0001031C
0x004C	EMC_CFG0_CH7	32'h0001031C
0x0050	EMC_CFG0_CH8	32'h0001031C
0x0054	EMC_CFG0_CH9	32'h0001031C
0x0058	EMC_CFG0_CH10	32'h0001031C
0x005C	EMC_CFG0_CH11	32'h0001031C
0x0060	EMC_CFG0_CH12	32'h0001031C
0x0064	EMC_CFG0_CH13	32'h0001031C
0x0068	EMC_CFG0_CH14	32'h0001031C
0x006C	EMC_CFG0_CH15	32'h0001031C
0x0070	EMC_STS0	32'h00000000
0x0074	EMC_STS1	32'h00000000
0x0078	EMC_STS2	32'h8007FFFF
0x007C	EMC_STS3	32'h8007FFFF
0x0080	EMC_STS0_CH0	32'h00000058
0x0084	EMC_STS0_CH1	32'h00000058
0x0088	EMC_STS0_CH2	32'h00000058
0x008C	EMC_STS0_CH3	32'h00000058
0x0090	EMC_STS0_CH4	32'h00000058
0x0094	EMC_STS0_CH5	32'h00000058
0x0098	EMC_STS0_CH6	32'h00000058
0x009C	EMC_STS0_CH7	32'h00000058
0x00A0	EMC_STS0_CH8	32'h00000058
0x00A4	EMC_STS0_CH9	32'h00000058
0x00A8	EMC_STS0_CH10	32'h00000058
0x00AC	EMC_STS0_CH11	32'h00000058
0x00B0	EMC_STS0_CH12	32'h00000058
0x00B4	EMC_STS0_CH13	32'h00000058
0x00B8	EMC_STS0_CH14	32'h00000058
0x00BC	EMC_STS0_CH15	32'h00000058

Delay Line Control Registers:

Address Offset	Register Name	Default Value
0x00C0	EMC_DL0	32'h00000000
0x00C4	EMC_DL1	32'h00000000
0x00C8	EMC_DL2	32'h00000000
0x00CC	EMC_DL3	32'h00000000
0x00D0	EMC_DL4	32'h00000000
0x00D4	EMC_DL5	32'h00000000
0x00D8	EMC_DL6	32'h00000000
0x00DC	EMC_DL7	32'h00000000
0x00E0 – 0x00FC	Reserved	32'h00000000
0x0100	EMC_DL16	32'h00000000
0x0104	EMC_DL17	32'h00000000
0x0108	EMC_DL18	32'h00000000
0x010C	EMC_DL19	32'h00000000
0x0110	EMC_DL20	32'h00000000
0x0114	EMC_DL21	32'h00000000
0x0118	EMC_DL22	32'h00000000
0x011C	EMC_DL23	32'h00000000
0x0120	EMC_DL24	32'h00000000
0x0124	EMC_DL25	32'h00000000
0x0128	EMC_DL26	32'h00000000
0x012C	EMC_DL27	32'h00000000
0x0130 – 0x013C	Reserved	32'h00000000

DMEM Control Registers:

Address Offset	Register Name	Default Value
0x0140	EMC_DCFG0	32'h100F3A12
0x0144	EMC_DCFG1	32'hCFFFFF3F
0x0148	EMC_DCFG2	32'h003FBFFF
0x014C	EMC_DCFG3	32'h00000000
0x0150	EMC_DCFG4	32'h00000000
0x0154	EMC_DCFG5	32'h00000000
0x0158	EMC_DCFG6	32'h00000000
0x015C	EMC_DCFG7	32'h00000000
0x0160 – 0x016C	Reserved	32'h00000000

SMEM Control Registers:

Address Offset	Register Name	Default Value
0x0170	EMC_SCFG0	32'h00FF0000
0x0174	EMC_SCFG1	32'h00003333
0x0178 – 0x017C	Reserved	32'h00000000
0x0180	EMC_SCFG0_CS0	32'h00100030
0x0184	EMC_SCFG1_CS0	32'h00007F7F
0x0188	EMC_SCFG2_CS0	32'h00A0744F
0x018C	EMC_SCFG3_CS0	32'h01224114
0x0190	EMC_SCFG0_CS1	32'h00100030
0x0194	EMC_SCFG1_CS1	32'h00007F7F
0x0198	EMC_SCFG2_CS1	32'h00A0744F
0x019C	EMC_SCFG3_CS1	32'h01224114
0x01A0	EMC_SCFG0_CS2	32'h00100030
0x01A4	EMC_SCFG1_CS2	32'h00007F7F
0x01A8	EMC_SCFG2_CS2	32'h00A0744F
0x01AC	EMC_SCFG3_CS2	32'h01224114
0x01B0	EMC_SCFG0_CS3	32'h00100030
0x01B4	EMC_SCFG1_CS3	32'h00007F7F
0x01B8	EMC_SCFG2_CS3	32'h00A0744F
0x01BC	EMC_SCFG3_CS3	32'h01224114
0x01C0 – 0x01FC	Reserved	32'h00000000

3.2 寄存器详解

3.2.1 通用控制寄存器

3.2.1.1 EXT_MEM_CFG0

在 SDRAM_GenMemCtlCfg 配置了 EXT_MEM_CFG0：

```
REG32 (EXT_MEM_CFG0) = (BIT_0|BIT_1|BIT_3|BIT_6|BIT_8|BIT_9|BIT_10|BIT_11);
```

这个 EMC 通用寄存器主要用来设置：software address 的映射，SDR/DDR 选择，command queue 的模式，SMEM/ DMEM 选择，使能 auto-sleep，使能 auto-gate，大小端等。

3.2.1.2 EXT_MEM_CFG1

在 SDRAM_GenMemCtlCfg 配置了 EXT_MEM_CFG1：

```
REG32 (EXT_MEM_CFG1) = 0x00000008; //EMC phy set
```

这个 EMC 物理通路选择寄存器主要用来设置：PIN 脚功能（SMEM/SDR/DDR），DQS，ADP/ADM 选择，CLK 的模式等。

3.2.1.3 EXT_MEM_CFG0_CSx

在 SDRAM_GenMemCtlCfg 配置了 EXT_MEM_CFG0_CS0：

```
REG32 (EXT_MEM_CFG0_CS0) = ( (0x1<<12) | (dburst_wlength<<8) | (dburst_rlength<<4) | (0x3));
```

这一组寄存器主要用来设置每个 CS 的特有的参数。包括 SMEM/DMEM 各自的读/写的 Burst/page/single 选择，Burst/page length 选择，当前 SMEM/ DMEM 选择等。

3.2.1.4 EXT_MEM_CFG0_CHx(0~15)

在 SDRAM_GenMemCtlCfg 配置了 EXT_MEM_CFG0_CH0：

```
REG32 (EXT_MEM_CFG0_CH0)  = 0x0003c31c;
```

这一组寄存器主要用来设置每个 channel 的特有的参数。包括 Channel 优先级，数据大小端切换，AHB INCR 读/写的 sub Burst length 选择，Asynchronous/Synchronous 选择，Single buffer/Double buffer，Channel 使能，Channel auto-sleep 使能等。

3.2.1.5 EXT_MEM_STS3

这个寄存器主要用来获取 EMC 的状态。包括 15 个 channel、CH MUX、SMEM、DMEM、EMC 的 idle 状态指示。

3.2.2 Delay Line 控制寄存器

SDRAM output clock features:

1. Clock frequency is fixed to the half of clk_emc frequency.
2. Support two phase-inverted options for delay coarse adjustment.
3. Support delay fine adjustment by a dedicated delay line.

3.2.2.1 EXT_MEM_DLx

其中[4:0] 表示 delay line 的 delay 值。[5]表示 delay line 是否使用 inverting 选项。

寄存器	功能描述

EMC_DL0	clksmem_in_r_dl
EMC_DL1	clksmem_in_w_dl
EMC_DL2	clksmem0_out_dl
EMC_DL3	clksmem1_out_dl
EMC_DL4	clkdmem_out_dl
EMC_DL5	dqs_ie_dl
EMC_DL6	dqs_out_dl
EMC_DL7	clkwr_dl
EMC_DL16	dqs_gate_pre_dl_0
EMC_DL17	dqs_gate_pre_dl_1
EMC_DL18	dqs_gate_pre_dl_2
EMC_DL19	dqs_gate_pre_dl_3
EMC_DL20	dqs_gate_pst_dl_0
EMC_DL21	dqs_gate_pst_dl_1
EMC_DL22	dqs_gate_pst_dl_2
EMC_DL23	dqs_gate_pst_dl_3
EMC_DL24	dqs_in_dl_0
EMC_DL25	dqs_in_dl_1
EMC_DL26	dqs_in_dl_2
EMC_DL27	dqs_in_dl_3

3.2.3 DMEM 控制寄存器

3.2.3.1 EXT_MEM_DCFG0

此寄存器提供了下面的几个参数来配置 SDRAM type：

drf_data_width:
0: 16-bit
1: 32-bit

drf_column_mode:
0: 8-bit
1: 9-bit
2: 10-bit
3: 11-bit
4: 12-bit

drf_row_mode:
0: 11-bit
1: 12-bit

2: 13-bit
3: 14-bit

drf_auto_pre_position:
0: A[10]
1: A[11]
2: A[12]
3: A[13]

另外还包括 row-hit detect logic 使能，clock output 使能，alternative bank pre-charge mode0/1 使能，auto-refresh 使能及模式选择，tREF，tRFC 等参数。

3.2.3.2 EXT_MEM_DCFG1

此寄存器用于配置 tRP,tRCD,tRRD,tWR,tXSR,tRAS-MIN,rtw(read-to-write turn-around time)，wtr(write-to-read turn-around time)，rtr(read-to-read turn-around time)，tMRD 等时序参数。

3.2.3.3 EXT_MEM_DCFG2

此寄存器用于配置 auto-sleep mode 及 refresh counter 选择，read data FIFO 的配置。

3.2.3.4 EXT_MEM_DCFG3

在 SDRAM_Device_Init 函数中，有 load mode register 代码：

```
REG32 (EXT_MEM_DCFG3) &= ~ (0xFFFF);
REG32 (EXT_MEM_DCFG3) |= (mode_reg | 0x40040000);
```

此寄存器用于 mode register 的配置,软件命令相关配置(PRECHARGE ALL BANK/AUTO REFRESH/LOAD MODE REGISTER/ENTER SELF REFRESH/ EXIT SELF REFRESH），每个 CS 的 Sleep status 等。

3.2.3.5 EXT_MEM_DCFG4

此寄存器用于 read data latency，write data latency，read DM latency，write DM latency，DQS output latency，sample clock latency 等 latency 时序参数的配置。

3.2.3.6 EXT_MEM_DCFG5

此寄存器用于 DQS gate pre signal delay select，DQS gate pst signal delay select 等时序参数的配置。

3.2.3.7 EXT_MEM_DCFG6

此寄存器用于 DQS PAD OE signal delay select，DQS PAD IE signal delay select 等时序参数的配置。

3.2.3.8 EXT_MEM_DCFG7

此寄存器用于 data PAD OE signal delay select，data PAD IE signal delay select 等时序参数的配置。

第4章 SC6800H SDRAM 客户化配置

Sdram 的配置，主要在主 make，sdram_prod.c(s_sdram_timing_param 和 sdram_config_info)和 mem_cfg.c。

4.1 project_sc6800h_XXX.mk

4.1.1 BB_DRAM_TYPE

```
BB_DRAM_TYPE      = 32MB_16BIT          ### DRAM Type:
                                        # Option1:NONE                #define BB_DRAM_TYPE_NONE
                                        # Option1:32MB_16BIT          #define BB_DRAM_TYPE_32MB_16BIT
                                        # Option2:32MB_32BIT_13R      #define BB_DRAM_TYPE_32MB_32BIT_13R
                                        # Option3:32MB_32BIT_12R      #define BB_DRAM_TYPE_32MB_32BIT_12R
                                        # Option4:64MB_16BIT          #define BB_DRAM_TYPE_64MB_16BIT
                                        # Option5:64MB_32BIT          #define BB_DRAM_TYPE_64MB_32BIT
```

此宏定义在 project makefile 中，用于配置 SDRAM 的大小，位宽，Row 宽，Column 宽等， 详细用法见 sdram_prod.c。

BB_DRAM_TYPE 暂时无法做到随 Nand/SDRAM 的自适应，这个宏必须随项目实际使用的 SDRAM 芯片的 spec 配置。

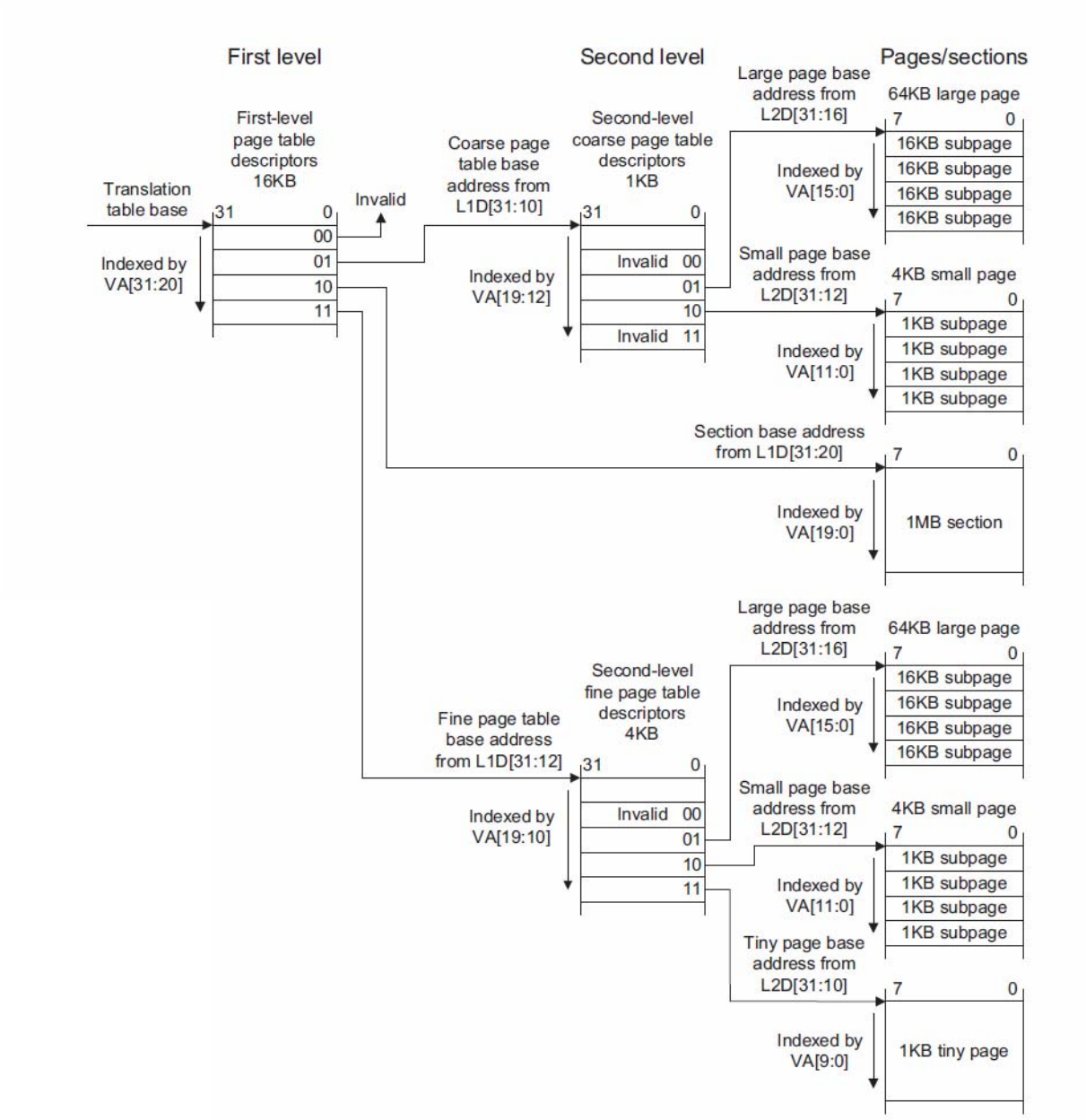
很多客户会在更换 MCP 后没有意识到要修改这个宏，导致出现无法下载、开机等情况。

4.1.2 VM_SUPPORT & MMU

此宏定义在 project makefile 中，用于打开虚拟内存的功能。

MMU 的基本原理就是把程序用的虚拟内存地址，通过 Demand Paging Table 中的 2 张表（First level, Second level）映射到相应的某个物理内存块(section or large page or small page or tiny page)中的某一个物理地址。如果没有找到这个物理内存块，那就会产出缺页中断，然后把缺的页加载进来。

如下图所示：



4.2 sdram_prod.c

4.2.1 s_sdram_timing_param

```
LOCAL CONST SDRAM_TIMING_PARA_T s_sdram_timing_param
{
    64,  //row_ref_max;    //ROW_REFRESH_TIME, Refresh interval time, ns
    30,  //row_pre_min;    //ROW_PRECHARGE_TIME, ns
    30,  //rcd_min;        //T_RCD,ACTIVE to READ or WRITE delay, ns
    20,  //wr_min;         //T_WR, WRITE recovery time, ns
    110, //rfc_min;        //T_RFC, AUTO REFRESH command period, ns
    150,//xsr_min;         //T_XSR, ns
    60,  //ras_min;        //T_RAS_MIN, row active time, ns
    20,  //rrd_min;        //T_RRD
    2,   //mrd_min;        //T_MRD, cycle
    2,   //wtr_min;        //T_WTR, cycle
};
```

s_sdram_timing_param 中的各个项都是时间参数，单位是 ns 或 cycle。以上参数均可以在 SDRAM SPEC 上查到。

4.2.2 s_sdram_config_info

```
LOCAL const SDRAM_CFG_INFO_T s_sdram_config_info =
{
    BK_MODE,                    //can only be set as BK_MODE_1, 2, 4, 8
    SDRAM_ROW_MODE,             //can only be set as ROW_MODE_11~13
    SDRAM_COL_MODE,             //can only be set as COL_MODE_8~12
    SDRAM_DATA_WIDTH,           //can only be set as DATA_WIDTH_8, 16
    BURST_LEN,                  //can only be set as BURST_LEN_2, 4, 8, 16, full page
    CAS_LATENCY,                //can only be set as CAS_LATENCY_1~3
    SDRAM_EXT_MODE_VALUE,       //can config extend register mode in SDRAM
    SDRAM_SIZE,                 //can only be set as 8, 16, 32, 64,128M
    SDRAM_CLK_DLY               //can only be set as 0X2120, 0X2220, 0X2320
};
```

sdram_config_info 用来配置 SDRAM 初始化所需要的基本参数。不同参数的配置是用宏 BB_DRAM_TYPE_XX_XX 配置，在 sdram_prod.c 中，如：

```
#if defined (BB_DRAM_TYPE_32MB_16BIT)
    #define SDRAM_ROW_MODE      ROW_MODE_13
    #define SDRAM_COL_MODE          COL_MODE_9
    #define SDRAM_DATA_WIDTH        DATA_WIDTH_16
    #define SDRAM_SIZE          SDRAM_32M
#elif defined(BB_DRAM_TYPE_64MB_16BIT)
    #define SDRAM_ROW_MODE      ROW_MODE_13
    #define SDRAM_COL_MODE          COL_MODE_10
    #define SDRAM_DATA_WIDTH        DATA_WIDTH_16
    #define SDRAM_SIZE          SDRAM_64M
    …
#endif
```

4.2.3 Userdef_XXX_sdram_init

为了能够支持更多的 SDRAM 以及让用户能够有更加灵活的配置，还提供了两个函数：

```
void Userdef_before_sdram_init ()
void Userdef_after_sdram_init()
```

这两个函数分别置于 SDRAM 初始化前和后。如果出现一些和通常初始化过程不一样的 SDRAM，就可以在这些函数辅助完成 SDRAM 的初始化。如果用户想在 FDL1 和 Bootloader 中扩展自己应用的话也可以填写这些函数。

4.3 Mem_cfg.c

6800H 的 memory config 更加复杂,配置时需要特别地小心。

4.3.1 MAX_PTE_TABLE_NUM

```
#define    MAX_PTE_TABLE_NUM              32     //32M Bytes
#define    MAX_PTE_TABLE_NUM              64     //64M Bytes
```

此宏在没有开启 VM_SUPPORT 时会用到。需要根据实际的 SDRAM size 配置。

4.3.2 mem_all_size

```
CONST uint32 mem_all_size          = 0x02000000;
```

表示 SDRAM 的容量。此宏根据实际的 DRAM size 配置。

4.3.3 MEM_RO_CODE_SIZE

```
#define    MEM_RO_CODE_SIZE          0x00200000(当 VM_SUPPORT 开启时)
```

此宏应配置为大于 kernel image size

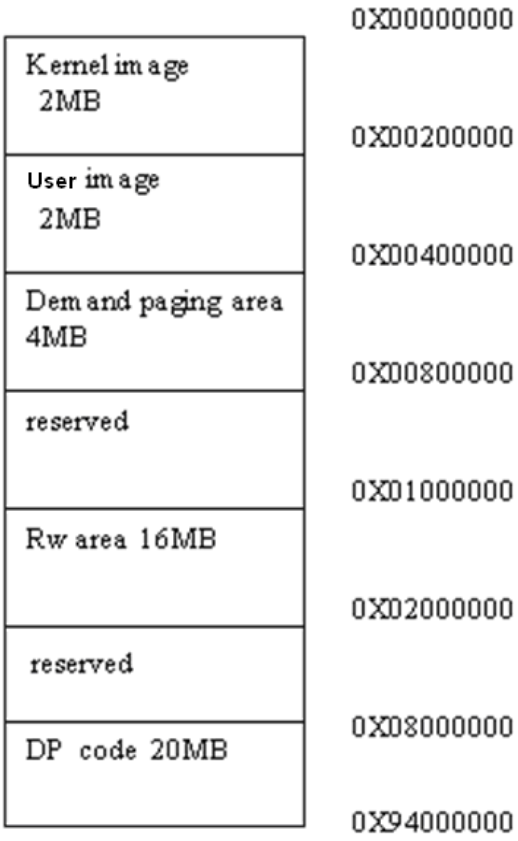
4.3.4 MEM_DP_VIR_AREA_SIZE

```
#define    MEM_DP_VIR_AREA_SIZE      0x01700000
```

此宏应配置为大于 user image size

4.3.5 vm_region

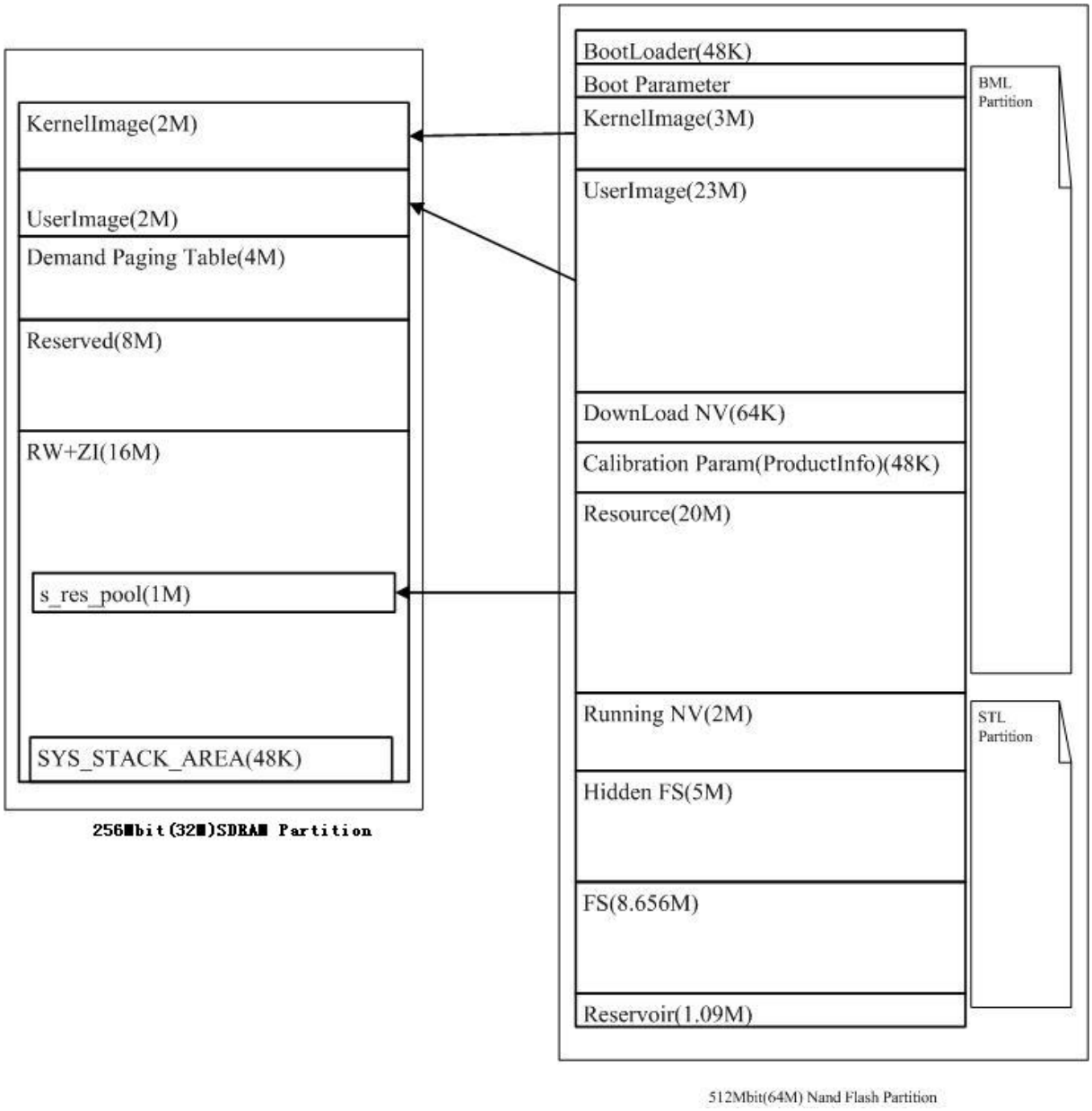
在开启 VM_SUPPORT，使用 32M SDRAM 的情况下，根据对应于项目的 vm_region 和 scatterfile 文件，SDRAM 空间的配置分布如下图。



其中 SDRAM 中的 DP Code(0x08000000~0x94000000)分区的地址，是 UserImage 应用程序看到的逻辑上的虚拟地址，不是真实的 SDRAM 的物理地址。

4.4 SDRAM 和 NAND Partition

在开启 VM_SUPPORT，使用 32M SDRAM + 64M NAND MCP 情况下，各个使用分区情况如下图。



其中 NAND Flash 中的 UserImage 会根据需要，按页动态加载到 SDRAM 中的 UserImage 中。NAND Flash 中的 Resource 会根据需要，动态加载到 SDRAM 中的 s_res_pool 中。

其中 NAND Flash 中的 STL Partition 是可以读写的，包括 Running NV，Hidden FS，FS 等分区，XSR 要求每个分区不得小于 2M bytes。当然如果 NAND 不够用，可以选择不要 FS 分区（即 UDISK）。

4.5 sdram_init.c 代码分析

在 fdl1，bootloader 的汇编初始化中，均调用了 Chip_Init 函数，Chip_Init 函数主要是调用了 Chip_ConfigClk 和 SDRAM_Init 函数。而 SDRAM_Init 函数配置了 SDRAM 的时序等参数。

4.6 sdram_phy.c 代码分析

在大 code 端的代码中，手机开机后在 PROD_Init 函数中调用了 HAL_SDRAM_InitPara(SDRAM_GetTimingPara())初始化了 SDRAM 的时序等参数。

第5章 SC6800H EMC 常见问题分析

涉及到下载的问题，都可以把下载工具下的 Channel.ini 里的 LogMode=0 修改为 5，打印出详细 Log 来分析下载失败的原因。

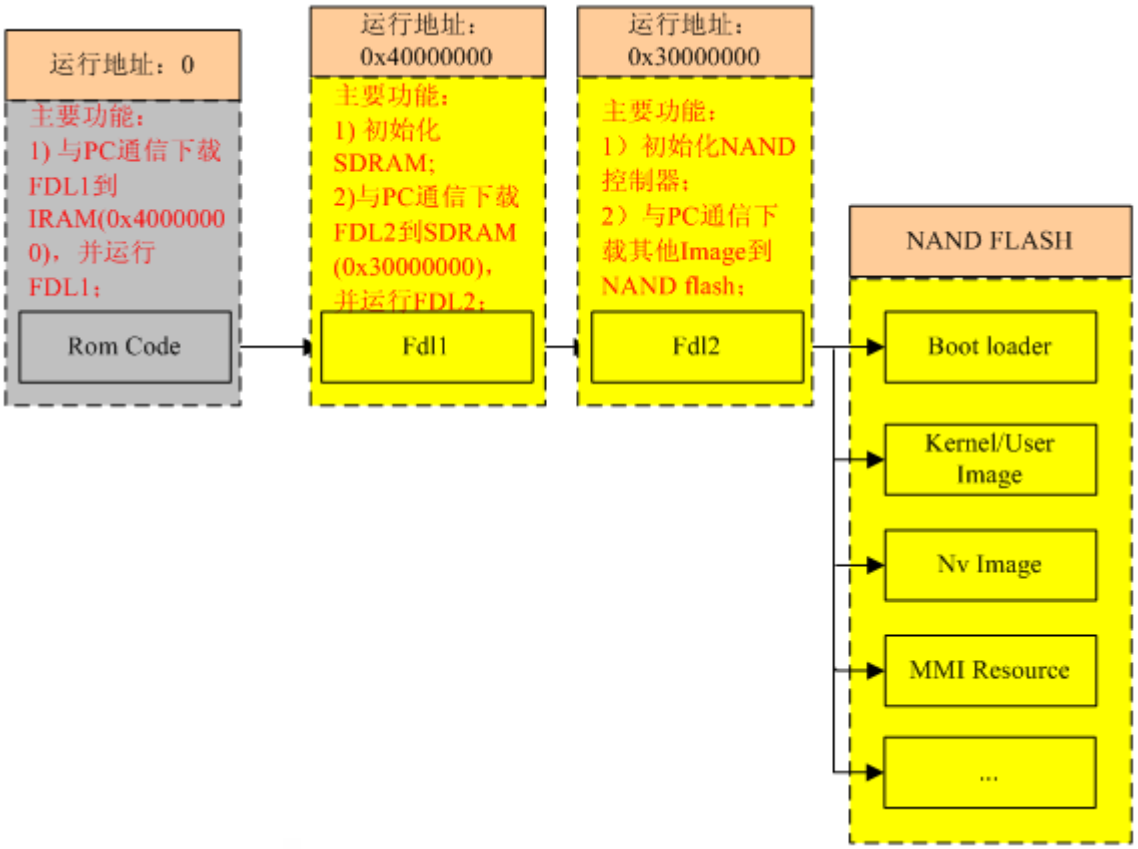
```
[Log]
; Text log level
; 0, No text log
; 1, Log errors,default value
; 2, Log warnings
; 3, Log runtime information
; 4, Log data only
; 5, Log everything
Level = 0 ----->5 修改此处
```

涉及到开机初始化（init.c）的问题，都可以直接通过 UART 和超级终端，打印出详细 Log 来定位死机的位置，分析开机失败的原因。

5.1 硬件关键检查项

1. BB 是否虚焊或损坏。
2. MCP 是否虚焊或损坏。
3. BB strappig pin 设置
4. Core 电压，I/O 电压
5. MCP PIN 设置是否正确。

5.2 RomCode/FDL1/FDL2 下载流程



5.3 FDL1 无法下载

此处指 FDL1 自身无法下载，或者下载失败。BB Rom Code 会负责把 FDL1 下载到 BB IRAM 中，所以要重点关注 BB。一般是硬件问题（但不涉及 MCP）。需要检查：

1. 电池：电量过低，电源不稳，需要换电池或电源。
2. UART 线：驱动过老需更新；线路或电平转换芯片不稳定需降低波特率；UART 接口有问题。
3. USB 线：驱动未安装；USB 线过长；下载操作流程不对。

5.4 FDL1 无法执行/FDL2 无法下载

此处指 FDL1 下载完成后，无法正常执行。FDL1 会负责把 FDL2 下载到 SDRAM 中，所以要重点关注 SDRAM。可以看下载工具的 log，测试 fdl1 的执行情况。一般是硬件问题。需要检查：

1. MCP 虚焊或损坏。
2. BB strappig pin 设置
3. Core 电压，I/O 电压设置问题。
4. MCP PIN 设置是否正确。

5.5 FDL2 无法执行

此处指 FDL2 下载完后，自身无法执行起来。FDL2 会负责把 PS，res 等下载到 NAND Flash 中，所以要重点关注 NAND Flash。

5.6 开机死机

正常的机器,不按 POWER 键,上电后,电流上去后,几秒后会自动归零。若发生了异常或者 ASSERT，电流会固定在一个几十 mA 上。

无法开机可能的原因：

1. 检查 BB_DRAM_TYPE 和时序配置。
2. 检查 GPIO 配置。
3. 开机 logo 时死掉，可能是 RUNNING NV 被破坏，可以使用 DLoaderR 等读出 RUNNING NV 与正常的 RUNNING NV 相比较。
4. 开机动画后出现的 assert，可以按普通的 assert 分析方法分析。

5.7 概率性死机

1. EMC PIN 的驱动能力不足。
2. 时序配置过高。
3. 射频干扰到 CLK 等，可以用飞行模式测试，或者导电布包一下。
4. 其他 26M 干扰问题。

5.8 系统反映速度慢

1. 先分析是否纯软件问题：
   如同时开启太多的应用。
   使用了不恰当的图片格式。
   中断，timer 来的太频繁。
   其他执行大量运算的程序。
2. Nand 或 SDRAM 自身慢：
   需要检查时序配置和硬件。

附录A Revision History

Version	Date	Owner	Notes
1.0	2011-6-7	臧永军	创建文档。
2.0	2011-6-12	臧永军	新增内容。
3.2	2011-7-8	臧永军	Release.