学校代码： 10246

学　　号： 08300720451

学 士 学 位 论 文

SDRAM 内存控制器研究

院　　系： 信息科学与工程学院

专　　业： 微电子系

姓　　名： 梁晨

指 导 教 师： 范益波

完 成 日 期： 2012 年 6 月 7 日

摘要

随着通用 CPU 迈向多核时代、视频处理 ASIC 迈向高清时代，存储器性能对系统整体性能的影响越来越大。然而由于工艺限制，DRAM 的核心频率难以超过 300MHz，各种旨在提高数据吞吐率的 DRAM 接口应运而生。优秀的内存控制器可以充分发挥 DRAM 接口的优势，在相同硬件条件下获得更好的性能，在相同性能要求下降低硬件成本。

在绪论部分，本文讨论了 DRAM 的技术发展历程，重点讲述了 SDRAM 带来的机遇与挑战，以及后 SDRAM 时代的各种技术改进。第一章详细讲解了 SDRAM 的基本操作，并综述了常见的 SDRAM 读写优化技巧，对它们的优略与适用场合做了较为深入的分析。

第三章给出了一种基于 Close Page Policy 的 SDRAM 控制器设计方案，并进行了简单的性能分析。第四章讲述了该设计方案的 verilog 实现、RTL 级仿真、FPGA 综合以及 FPGA 验证。第五章讲解了该设计方案的 SOPC 集成，包括标准化的 Avalon-MM 接口设计、BFM 仿真测试、JTAG 硬件测试。

关键词：SDRAM，内存控制器，SOPC

ABSTRACT

As general purpose CPU enters the multi-core era and video processing AISC enters the high-definition era, memory performance becomes increasingly important to overall system performance. However due to technology restricts, the core frequency of DRAM hasn't exceeded 300MHz. Thus there emerged many DRAM interface protocols aiming at high data throughput. A well-designed DRAM controller can take advantage of such protocols and achieve better performance using the same hardware or lower cost under the same performance requirement.

In the introduction part, this paper reviewed the history of DRAM technology, especially the chances and challenges that SDRAM brought about, as well as new features appeared in the post-SDRAM age. The first chapter starts with the basic operation of SDRAM, and several effective techniques for optimizing SDRAM accesses are summarized.

Then an SDRAM controller design solution based on Close Page Policy is proposed, and basic performance analyses are performed. Next, the Verilog implementation, RTL-level simulation, FPGA synthesis and FPGA verification of the proposed design are described. Finally, the SOPC integration of the proposed design is covered, including standardized Avalon-MM interface designing, BFM simulation and hardware verification through JTAG.

Keyword: SDRAM, memory controller, SOPC

目录

第一章 绪论

随着工艺尺寸的缩小，当代 ASIC 与 CPU 的性能突飞猛进，然而存储器的核心频率任然只有 100~200MHz 左右，SDR/DDR/DDR2/DDR3 等旨在提高数据吞吐率的接口协议应运而生。当下，设计良好的存储控制器，已成为充分发挥系统带宽潜能、降低系统功耗的有力保障。

本章内容：首先，将分三部分简要介绍 DRAM 存储器：前 SDRAM 时代的 DRAM、SDRAM 带来的机遇与挑战、后 SDRAM 时代的 DRAM；然后，将介绍本课题的内容，给出论文结构。

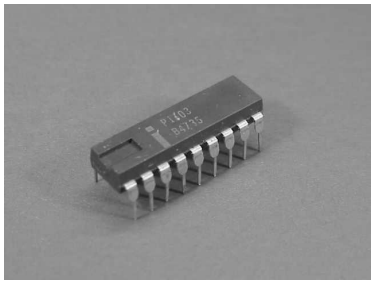
1 前 SDRAM 时期的 DRAM

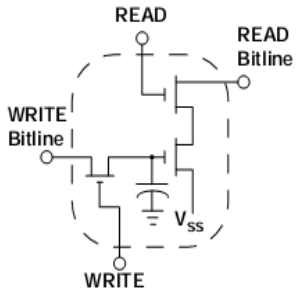
1.1 DRAM 的发明

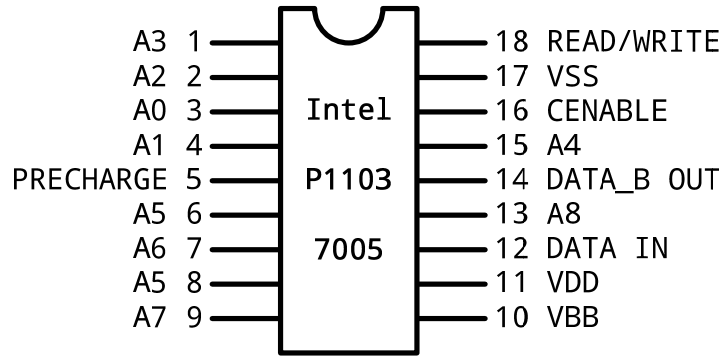
1966 年 Robert Dennard 博士在 IBM 的 Thomas J. Watson 研究中心发明了 DRAM 存储器，其中的存储单元采用了 1T1C 的结构。Robert Dennard 获得了美国专利，专利号为 3,387,286。

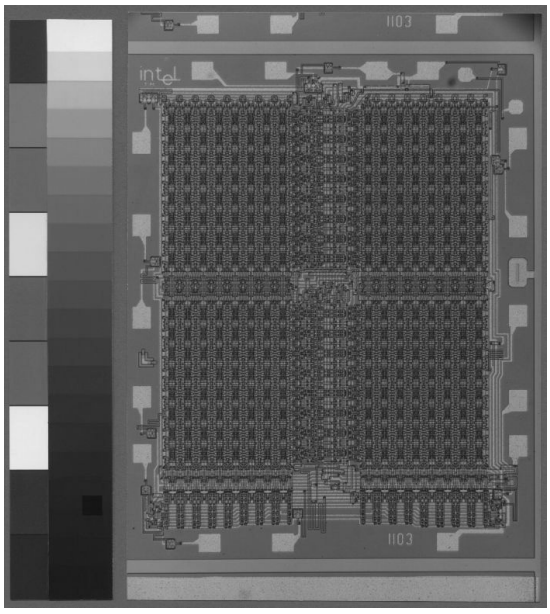
1.2 Intel 1103

1971 年 Intel 发布了 Intel 1103 存储器，容量为 1kbit。它是历史上第一个商业上成功的 DRAM 存储器。与经典 DRAM 不同，它采用 3T1C 的存储单元，有独立的行地址线与列地址线，读与写的数据线分离。







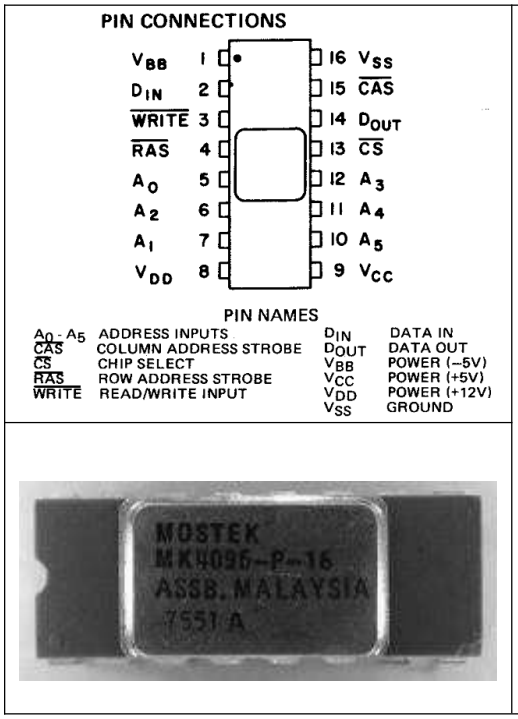


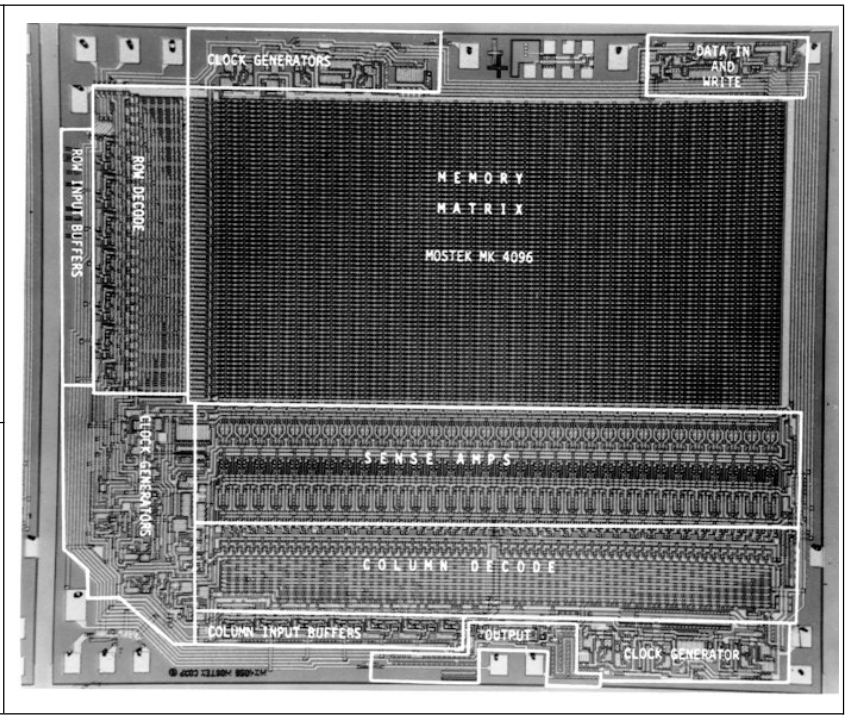
1.3 Mostek MK4096

1973 年 Mostek 发布了 Mostek MK4096 存储器，容量为 4k，由 Robert Proebsting 设计。它是历史上第一个复用行列地址线的 DRAM。行列地址线的复用，减少了封转管脚，降低了成

本，这个传统延续至今。

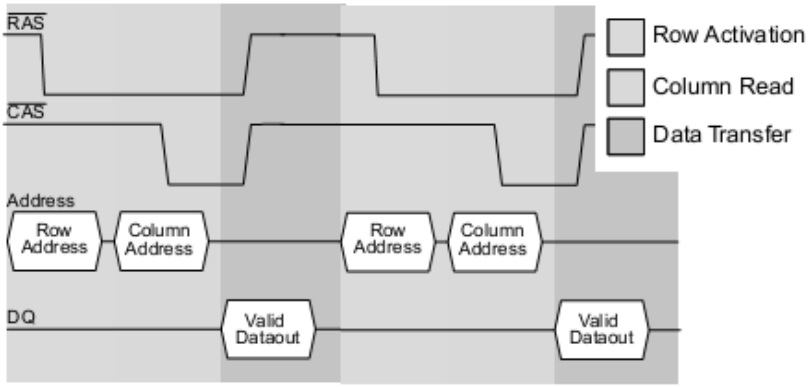
在鼎盛时期，Mostek 曾占有 3/4 的市场份额。





1.4 典型的传统异步 DRAM

在 1970 年代中期，占据主流的是异步接口的 DRAM，那时的的“Clocked DRAM”仅仅昙花一现。这种经典异步接口，与课本《数字集成电路设计透视》中的叙述十分吻合。每次读写前，必须分别进行行选通与列选通，即便读写同一行里的数据，也不能省略任何步骤。一张典型的时序图如下：



1.5 Page Mode、Fast Page Mode(FPM)、Nibble mode 的出现

在异步 DRAM 的时代，出现过许多改进极富有特点的接口改进。

Page Mode 的出现是一个里程碑。Page Mode 的 DRAM 可以把一整行数据保存在集成于片上的灵敏放大器整列中，这访问同一行时，就不必重复进行行选通。Page Mode 的操作，利

用了访问的空间局部性，从而提升了系统的性能。行缓冲的思想，一直延续到了当代的 DRAM。

Fast Page Mode 出现于 1980 年代早期，直到 1990 年代早期任然属于主流。Fast Page Mode 中，只要 RAS_B 有效，就打开列地址缓冲，从而可以在 CAS_B 发起前锁存列地址，这样就提高了读写效率。

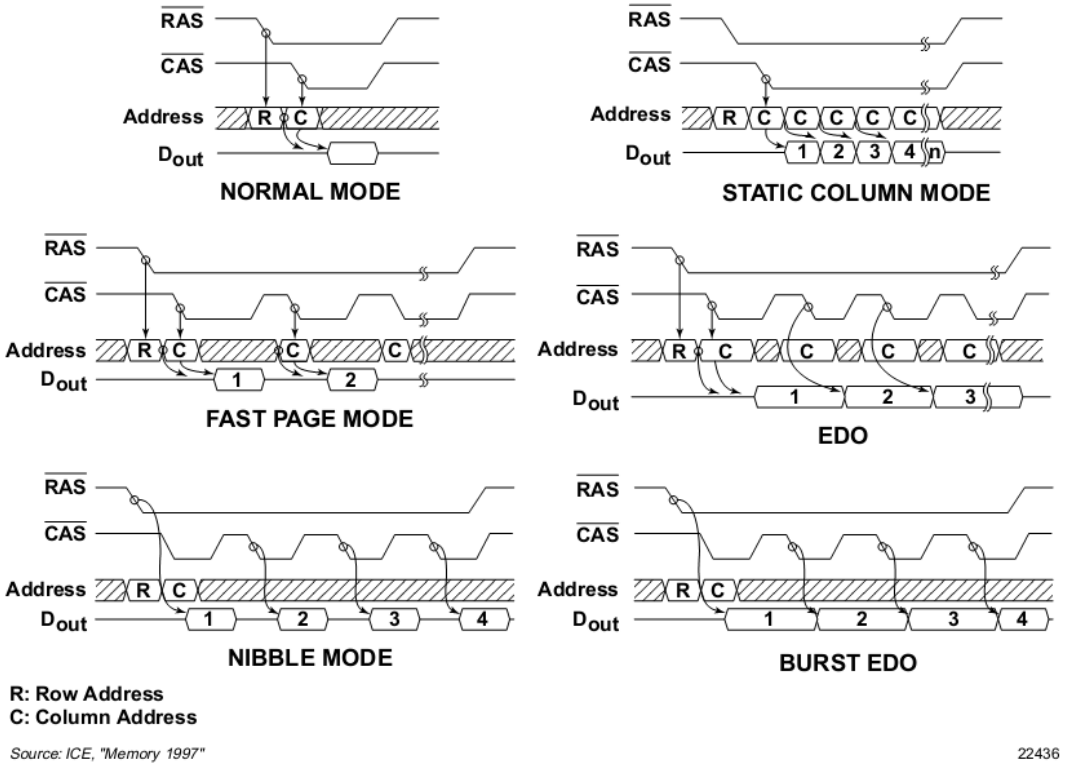
Nibble mode 是 TI 对 FPM DRAM 的一种改进，添加了类似于 4 位突发传送的功能。

1.6 Extended Data Output(EDO)与 Burst EDO 的出现

在 1990 年代中期，出现了 Extended Data Output 的 DRAM。它增加了 OE_B 信号，来代替 CAS_B 去控制输出缓冲。于是，在 CAS_B 上升后，数据可以保持更多时间，故名“Extended Data Output”。EDO 允许 CAS_B 下降沿与数据交叠，缩短了 page mode 下的读写周期，故效率比 FPM 更高。

Burst EDO 增加了列地址自动增加的功能，进一步简化了操作，提高了效率。

几种异步模式的时序对比如下图：



(上图摘自 ICE 公司的《Memory 1997》)

2 SDRAM 带来的机遇与挑战

SDRAM，意为同步的 DRAM，其数据和指令都与时钟上升沿对齐。最早的样品，由三星在 1993 年生产。在 1996-2002 年期间，SDRAM 逐步取代了异步的 FPM DRAM、EDO DRAM，称雄 PC 内存市场。在 2003 年之后，逐渐被 DDR SDRAM 取代。它与前代异步 DRAM 的不同

包括：同步时钟、多 Bank 机制、流水线化的操作、Burst 读写的引入。由于每个时钟周期，只在上升沿传送一次数据，它也被称为 SDR SDRAM，以便与 DDR SDRAM 区别。

2.1 SDRAM 基本硬件参数

SDRAM 的常见容量包括：16Mbit、64Mbit、128Mbit、256Mbit、512Mbit。其中，除了 16Mbit 分为 2 个 Bank 外，其他容量的 SDRAM 都分为 4 个 Bank。

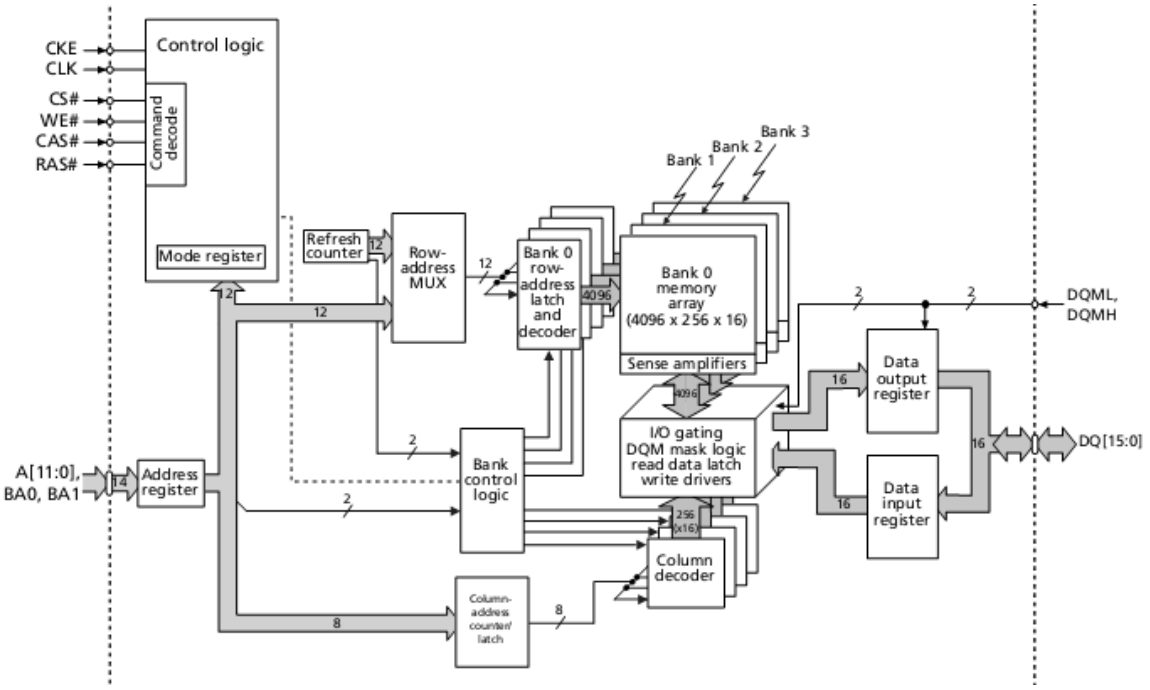
SDRAM 的常见数据位宽包括：4bit、8bit、16bit、32bit。如果系统位宽 64bit，用 16 个 512Mbit 容量 4bit 位宽的 SDRAM 并联，可以实现 8Gbit 的存储空间；而如果用 2 个 512Mbit 容量 32bit 位宽的 SDRAM 并联，只能实现 1Gbit 的存储空间。

SDRAM 的 Burst Length，即读写突发传送的周期数，一般可设置为 1、2、4、8 以及 Full Page，在初始化时通过写模式寄存器来设置。需要注意的是，如果列读写时，地址没有与 BL 对齐，SDRAM 会自行 Wrap 地址，导致意外的结果。

SDRAM 的 Cas Latency，即从发送列读取指令到有效数据出现的延时，一般为 2 或 3 个时钟周期，少数器件可以为 1 个周期，在初始化时通过写模式寄存器来设置。一般 CL=3 时，tAC 参数(即时钟上升沿到有效数据出现的延时)较小，从而使器件可以工作在较高的频率。

SDRAM 的常见速度等级包括：100MHz（T=10ns）、125MHz(T=8ns)、133MHz（T=7.5ns）、143MHz（T=7ns）、166MHz（T=6ns）、183MHz(T=5.5ns)、200MHz（T=5ns）。速度等级一般对应 CL=3 时的最高工作频率。器件的最高工作频率越高，表示其 AC 延时参数越小。

2.2 SDRAM 的三维地址空间



(上图摘自 Micron 公司的 SDRAM 器件手册)

当代 SDRAM 的地址空间由 Bank、Row、Column 三个维度。64Mbit 及以上容量的 SDRAM 一般分为 4 个 Bank。为了与内存模组中 Physical Bank（Rank）的概念区别，SDRAM 的 Bank 有时也叫 Logic Bank。

每个 Bank 都是一个 Row 与 Column 构建的阵列；阵列的每个单元都是一组数据，这组数据的位宽与 SDRAM 数据接口位宽一致。

例如，一个容量 64Mbit 位宽 16bit 的 SDRAM 地址空间为：4bank * (2^12)Row * (2^8)Column * 16 bit = 64Mbit，器件手册一般写为 4 * 1M * 16bit = 64Mbit。

2.3 芯片规格与 SDRAM 芯片组的容量

关于 SDRAM 芯片寻址的术语——Bank、Row、Column 已在上一节给出了解释。下面解释几个 SDRAM 芯片组中常用的术语。

Rank：又叫 Physical Bank，指的是一组被同时操作的 DRAM 芯片。通常，它们的数据总线被合并，以提供系统所需要的数据位宽，如 Intel 处理器要求的 64bit。

Page：一般指同一个 Rank 中 Bank adddress、Row address 相同的空间。可以认为将 Rank 中所有 SDRAM 芯片中的 Row Buffer 合并在一起，就组成了 Rank 中的 Page Buffer。关于 Row Buffer、Page Buffer 对 DRAM 访问的影响将在下一节讲述。

以 64bit 位宽的系统为例，综合列出不同规格 SDRAM 的 3 维地址空间，以及组成 Rank 后的特性：

Chip Size	Chip DQ/ Fab.	Bank Num	Row Num	Col Num	Row Buf Size	64bit Page Size	64bit Rank Size
16Mb	16 ISSI	2	2048	256	4kb	16kb	64Mb
64Mb	32 Micron	4	2048	256	8kb	16kb	128Mb
	16 Micron	4	4096	256	4kb	16kb	256Mb
	8 Micron	4	4096	512	4kb	32kb	512Mb
	4 Micron	4	4096	1024	4kb	64kb	1Gb
128Mb	32 Micron	4	4096	256	8kb	16kb	256Mb
	16 Micron	4	4096	512	8kb	32kb	512Mb
	8 Micron	4	4096	1024	8kb	64kb	1Gb
	4 Micron	4	4096	2048	8kb	128kb	2Gb
256Mb	32 ISSI	4	4096	512	16kb	32k	512Mb
	16 Micron	4	8192	512	8kb	32kb	1Gb
	8 Micron	4	8192	1024	8kb	64kb	2Gb
	4 Micron	4	8192	2048	8kb	128kb	4Gb

512Mb	32 ISSI	4	8192	512	16kb	32kb	1Gb
	16 Micron	4	8192	1024	16kb	64kb	2Gb
	8 Micron	4	8192	2048	16kb	128kb	4Gb
	4 Micron	4	8192	4096	16kb	256kb	8Gb

注：其中 Micron 64Mb: x32 的芯片虽然是 2048 行，计算刷新周期时要按 4096 行算。而 ISSI 16Mb: x16 的芯片也是 2048 行，刷新时确实按 2048 行算。关于刷新的原理见后文。

2.4 Row Buffer 对 SDRAM 读写的影响

SDRAM 的每个 Bank 都有一个 Row Buffer，其实就是一排灵敏放大器(Sense Amplifier)。每次激活(Activate)一个 Row 地址后，就打开 Row Buffer 把整个 Row 的数据保存在其中；Precharge 指令可以关闭指定 Bank 的 Row Buffer，Precharge All 可以关闭所有 Bank 的 Row Buffer；我们称 Row Buffer 关闭的 Bank 为空闲(Idle)的 Bank。

由于 Row Buffer 的存在，SDRAM 的访问分为如下三种情况：

(1) 当前访问的 Row 所在的 bank 中，row buffer 关闭。此时，需要先激活 Row，再发读写指令和 Column 地址。此情况，读写延时中等。

(2) 当前访问的 Row，正好保存在相应 bank 的 row buffer 里。此时，不需要激活 Row，直接发读写指令和 Column 地址，就可以访问。此情况，读写延时最小。

(3) 当前访问的 Row 所在的 bank 中，row buffer 打开，存的是另一个 Row 的数据。此时，必须先发送 Precharge 或 Precharge All 指令关闭 row buffer，再激活 Row，最后发读写指令和 Column 地址。此情况，读写延时最大。

2.5 SDRAM 的新天地

SDRAM 带来的最大机遇就是它流水线化、并行化的接口设计，只要不违反管脚、时序延时的限制，多个 Bank 可以并肩工作。

恰如 Scott Rixner 在 ISCA2000 上指出的，“得益于当代存储器件的三维特性，重新排列存储操作顺序以分时访问 DRAM 大有好处。这样的优化，就好比超标量处理器乱序调度算数操作一样。”

围绕着访问请求重排、SDRAM 指令乱序执行、地址映射、Row Buffer 关闭策略，各种优化方法层出不穷，从简单的多 bank 交错操作，到包含自适应机制的 Row Buffer 管理策略，不胜枚举。

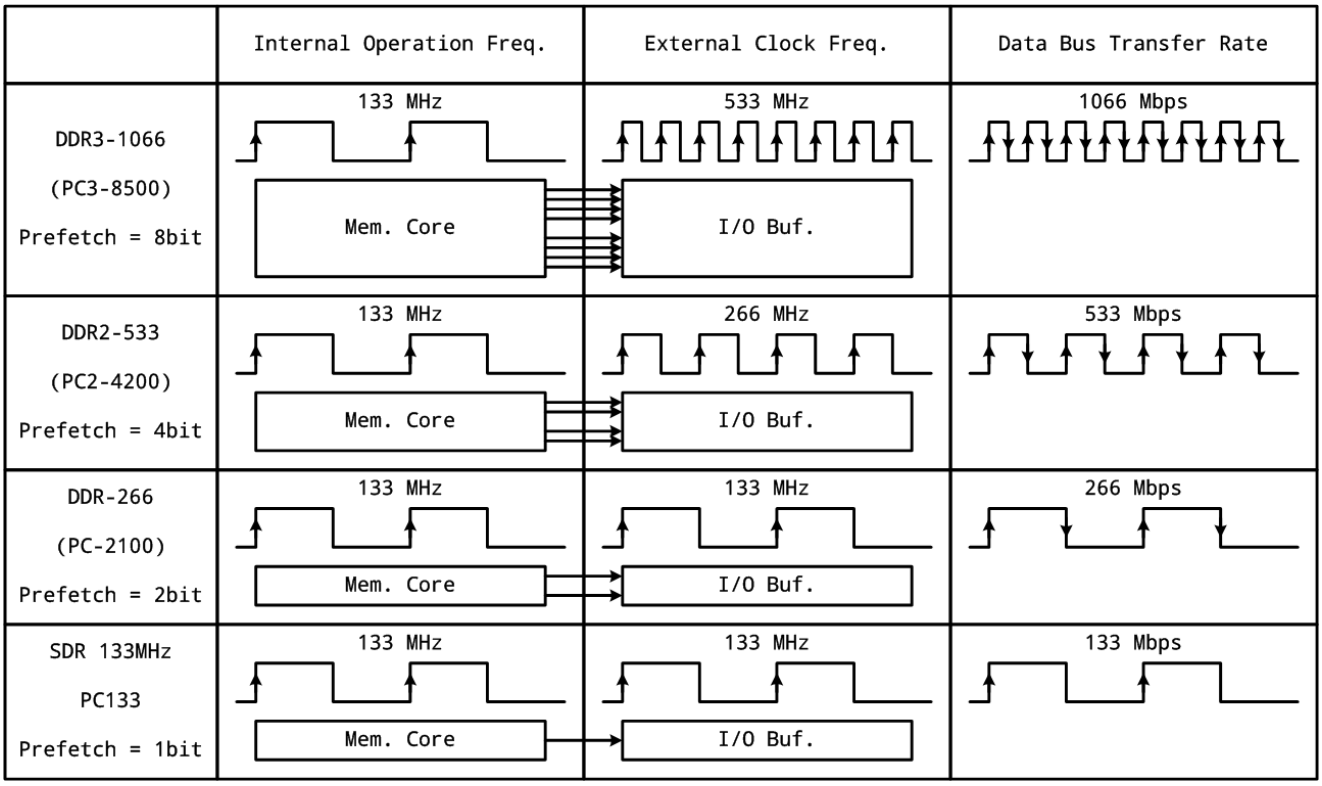
针对视频处理、多核、通信等不同应用，设计 SDRAM 控制器时还需要根据“实时性”的限制，在低延时和高带宽利用率之间有所取舍。

所有这些开启了 DRAM 控制系统的一个新篇章。

3 后 SDRAM 时代的 DRAM

SDRAM 之后，依次出现了 DDR、DDR2、DDR3 这几代 DRAM。和 SDR SDRAM 一样，它们也采用多 bank 流水线化操作的同步架构，其内部半导体存储单元的核心频率也任然在 100~200MHz 之间。DDR、DDR2、DDR3 相对于 SDR SDRAM 最大的区别在于使用了多 bit 预取的技术。从 DDR 开始，同时在上升沿和下降沿对数据进行采样。从 DDR2 开始，I/O 接口的频率开始高于存储单元的核心频率。

下图对 SDR、DDR、DDR2、DDR3 的 SDRAM 做了对比：



不过，也正是多 bit 预取技术，使得 DDR SDRAM 的最小 Burst Length 为 2 而 DDR2 SDRAM 的最小 Burst Length 为 4，DDR3 略有不同，最小 Burst Length 为 4。

另一方面，从 SDR、DDR、DDR2 到 DDR3，指令延时略有差异，SDR 访问的优化算法也可用于 DDR、DDR2 到 DDR3，但需要更具参数的相对关系重新调整，不能生搬硬套。

3.1 DDR SDRAM 引入的其他重要特性

差分时钟 CK 和 CK_B。差分时钟可以抑制温度、电阻等因素对时钟精度的影响。在 CK 上升沿和 CK_B 下降沿的交叉点采样对指令信号采样；把 CK 和 CK_B 的所有交叉点作为数据采样的参考点。

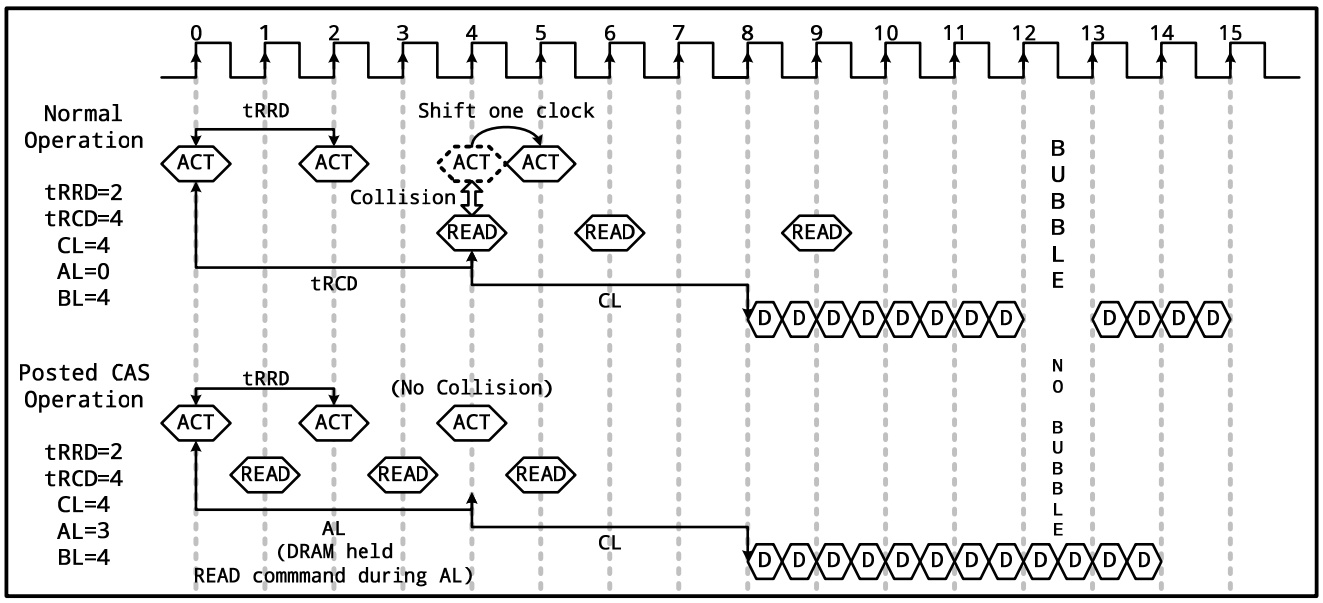
数据采样脉冲 DQS。DDR SDRAM 使用 Source-Synchronous 接口，以确保搞数据传送率下的信号完整性。DQS 是一个双向端口，写数据时，DQS 为 DDR SDRAM 的输入，其上升沿和下降沿分别与两个数据各自的中心对齐；读数据时，DQS 为 DDR SDRAM 的输出，其上升

沿和下降沿分别与两个数据各自的起始边沿对齐。

片上 DLL。随着数据传送频率的提升，时钟树延时(Clock Insertion Delay)变得更加严重，使得片上 DLL 成为了必需。片上 DLL 可以通过模式寄存器来开启和关闭，使用时通常会开启 DLL。

3.2 DDR2 SDRAM 引入的其他重要特性

Post CAS。为提高带宽利用率而作的接口时序改进。在 SDR 和 DDR SDRAM 的时代，行选通、列读取之间必须满足 tRCD 延时；DDR2 接口中可以在行选通后直接进行列读取，而有效数据出现相对于行选通的延时不变。这样可以避免不同 bank 间的行选通、列读取指令碰撞，提高效率。



ODT(On-Die Termination)。终端电阻起着平衡信号完整性和电压摆幅的作用。在 SDR 和 DDR SDRAM 的时代，终端电阻是做在电路板上的，而 DDR2 使用了片上终端电阻(ODT)，从而提高了配置的灵活性，降低了电路板的设计难度与成本。内存控制器通过写 EMR 寄存器来设定或更新 ODT 的值。

OCD(Off-Chip Driver)校准。DDR2 SDRAM 引入了片外驱动校准，以提高信号完整性。校准时，调整的是片内上拉、下拉电阻的等级。由于 DDR2 还引入了差分的数据选取脉冲，基本已满足了信号完整性的要求，所以 OCD 也只在高端场合应用过。目前 Micron 等公司的 DDR2 SDRAM 已经不再支持 OCD 了。

3.3 DDR3 SDRAM 引入的其他重要特性

RESET_B 管脚。DDR3 SDRAM 增加了 RESET_B 管脚，使得存储芯片可以很容易地进入初始化状态。当 RESET_B 有效时，存储芯片内部时钟关闭，处于低功耗的状态；RESET_B

从有效变为无效后，存储芯片自动进入初始化阶段。

Burst-Chop 与 On-The-Fly 技术。DDR3 SDRAM 采用 8n-bit 预取，从 DDR、DDR2 的情况推测应该只支持长度为 8 的突发传送。但是，DDR3 SDRAM 支持 Burst-Chop，能够实现长为 4 的突发传送。可以通过模式寄存器设置突发传送长度，也可以使用 On-The-Fly 技术通过 A12 地址位在发起读写时确定突发传送长度。需要注意的是 Burst-Chop，只相当于一个 Mask 操作，不能节约总线时间。

ZQ 校准。DDR3 SDRAM 的 ZQ 校准提供了更加有效的可控阻抗机制。初始化时的长校准，使 DRAM 能够将驱动电路工艺偏差带来的影响最小化；正常操作时的短校准，可以减小电压、温度漂移带来的阻抗变化。ZQ 校准为良好的信号完整性提供了保障。

动态片上终端电阻(Dynamic ODT)。片上终端电阻(ODT)可以减小当前模组的时钟抖动以及其他模组引起的信号反射。确保信号完整性，就可以提供可预测的有效数据窗口(data eye)。动态片上终端电阻技术，就是可以根据是否进行写操作而在标准电阻、写电阻之间自动切换，免去了像在操作 DDR2 SDRAM 时那样更改模式寄存器的麻烦。

Auto Self-Refresh。JEDEC 标准中的可选特性，如果实现并被启用，SDRAM 会根据自身的温度决定自刷新的频率以降低功耗。

Partial Array Self-Refresh。JEDEC 标准中的可选特性，如果实现并被启用，可一只刷新 SDRAM 中有必要刷新的 Bank，以降低功耗。

4 本课题的内容

本课题研究 SDRAM 内存控制器的设计。从 SDRAM 的结构与基本操作入手，分析各种优化策略，然后给出基于 Close Page Policy 的内存控制器设计方案，并将其集成到 Altera 的 SOPC 系统中。

整个设计流程包括：Spec 编写、模块划分与时序设计，RTL 级的 verilog 设计，RTL 级的 ModelSIM 仿真，Quartus 综合，后仿真，静态时序分析，FPGA 测试。

SOPC 集成的工作包括：Avalon-MM Slave Wrapper 的编写，Avalon-MM Master BFM 仿真测试，基于 Jtag to Avalon Master Bridge 的 FPGA 硬件测试。

5 论文结构

第一章：绪论。

第二章：SDRAM 基本操作与优化策略。

第三章：基于 Close Page Policy 的内存控制器后端设计。

第四章：SDRC_Lite 内存控制器的仿真、综合与硬件测试。

第五章：SDRC_Lite 内存控制器的 SOPC 集成与测试。

第六章：致谢。

6 本章参考文献

[1] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. Memory access scheduling. ISCA2000: p128–138

[2] JEDEC. JEDEC standard No.21-C. 2012

[3] JEDEC. JEDEC standard JESD79F. 2008

[4] JEDEC. JEDEC standard JESD79-2F. 2009

[5] JEDEC. JEDEC standard JESD79-3E. 2010

[6] Computer History Museum. http://www.computerhistory.org

[7] ISSI. http://www.issi.com/

[8] Micron. http://www.micron.com/

[9] Micron. 64Mb x4, x8, x16 SDR SDRAM. 2012

[10] Micron. TN4605 General DDR SDRAM Functionality. 2001

[11] Micron. TN4605 DDR2 Offers New Features and Functionality. 2005

[12] Micron. TN4102 DDR3 ZQ Calibration. 2008

[13] Micron. TN4104 DDR3 Dynamic On-Die Termination. 2008

[14] Elpida. E0678E10 DDR2 SDRAM Technology. 2005

[15] Elpida. E1503E10 New Features of DDR3 SDRAM. 2009

[16] Integrated Circuit Engineering Corporation. Memory 1997. 1997 :Ch7 DRAM Technology

[17] Jan M. Rabaey, A. Chandrakasan, B. Nikolic. Digital Integrated Circuits (2nd Edition). 2003: Ch12 Designing Memory and Array Structures

[18] Brent Keeth, R. J. Baker, B. Johnson, Feng Lin. DRAM Circuit Design: Fundamental and High-Speed Topics. 2007: Ch01 An Introduction to DRAM

[19] Bruce Jacob, Spencer Ng, David Wang. Memory Systems: Cache, DRAM, Disk. 2007: Ch7 Overview of DRAMs

第二章 SDRAM 基本操作与优化策略

由于指令操作的存在，SDRAM 系统的硬件带宽往往得不到充分利用。良好的 SDRAM 访问策略可以提高 SDRAM 系统的硬件带宽的利用率，在相同硬件条件下获得更好的性能或在相同性能要求下降低硬件成本。

SDRAM 具有多 Bank、流水线化等结构特性，如果能够巧妙地排列指令，可以开发 SDRAM 的并行化优势节省总线时间；SDRAM 具有行缓冲，如果根据数据访问的空间局部性、请求延时的限度，适当重排访问请求、控制缓冲策略，可以减少指令的数量，提高总线效率。

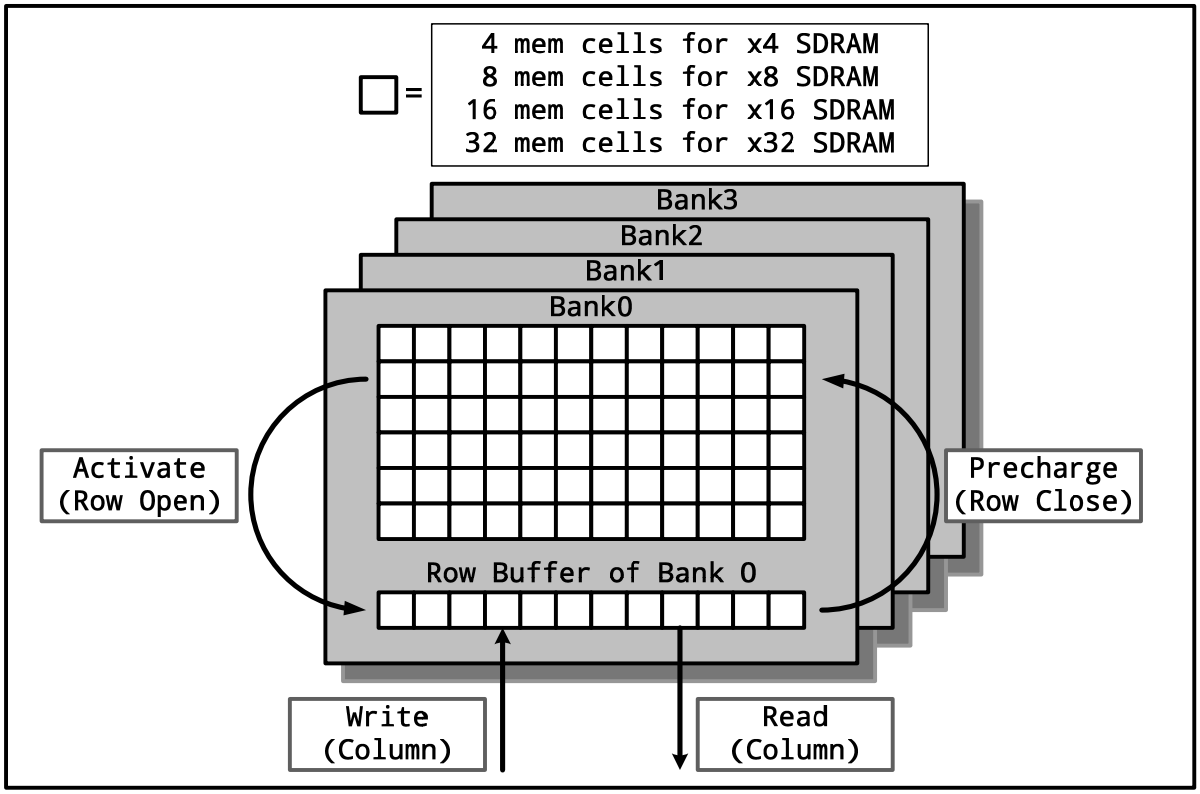
本章内容包括：SDRAM 的特性回顾与基本操作、SDRAM 访问的优化方向与常用策略。由于 SDRAM 的初始化与带宽利用率的优化策略无关，故另行归入控制器设计原理的章节中。从本章开始 SDRAM 专指 SDR SDRAM。

1 SDRAM 的基本操作

1.1 SDRAM 回顾

SDRAM 是同步动态随机存储器的缩写。同步，指它有统一的时钟，在时钟上升沿采样指令、读入或输出数据，与早期的异步 DRAM 不同；动态，指它必须不断刷新以维持数据，与 SRAM 不同；随机存储器，指对它的访问是任意的，而不像 FIFO 或堆栈那样存在访问限制；同时，RAM 一直以来也隐含着掉电后会丢失数据的意思，与 ROM、PROM、Flash 等非挥发存储器相对。

SDRAM 具有由 Bank、行(Row)、列(Column)构成的三维结构，如下图：



如图，SDRAM 分为 4 个 Bank；每个 Bank 都包含一套二维存储阵列，以及一个行缓冲(即一排灵敏放大器)；阵列中的每一行，分为若干列，行激活(Activate)时，对应的行被放入行缓冲中，预充电(Precharge)时，对应的行被写回存储阵列；行中的每一个列，由若干 DRAM 单元构成，单元个数由 SDRAM 芯片的数据总线位宽决定，SDRAM 的读写操作对应的都是行缓冲中的某个列。

SDRAM 的多个 Bank 可以进行不同的工作。例如，Bank0 进行行激活时，Bank1 的数据传输或预充电不会受到影响。实质性的限制只有两方面：SDRAM 的指令、数据总线各只有一套，SDRAM 指令之间必须满足规定的延时条件。

了解 SDRAM 指令与基本操作，是研究 SDRAM 访问优化策略的基础，接下来将对它们作简要介绍。

1.2 SDRAM 的基本指令

首先简要介绍一下 SDRAM 的管脚。

1973 年 Mostek 发布了 Mostek MK4096 存储器，容量为 4k，创新地采用了行列地址线复用技术，从而减少了管脚降低了成本，这个传统延续至今。

典型 SDRAM 的 I/O 管脚包括：

时钟输入(clk)。SDRAM 的最高工作频率一般在 100~200MHz 之间。

时钟使能(cke)。与各种休眠模式有关，不在本课题研究范围之内，相关设计中直接置为 1。

Bank 地址(ba1, ba0)。Bank 地址，选取当前指令对应的 Bank。

地址总线(an~a0)。这就是复用的行列地址总线，行操作时发送行地址，列操作时发送列地址。在 Precharge、Read、Write 时 a10 有特殊含义，见下文表格。

数据总线(dqn~dq0)。在 SDRAM 术语中，dqn~dq0 所有数据的总和也称为 1 个 word，它是三维存储整列访问的最小单元。

数据 Mask(dqm 或 dqm[3:0]或 dqmh、dqml)。4 位和 8 位的 SDRAM 使用 dqm 对应整个 word；16 位 SDRAM 使用 dqmh、dqml 对应高、低字节；32 位 SDRAM 使用 dqm 分别对应 4 个字节。读数据时 dqm 为 1 会使输出高阻态，写数据时 DQM 为 1 表示当前数据不写入。

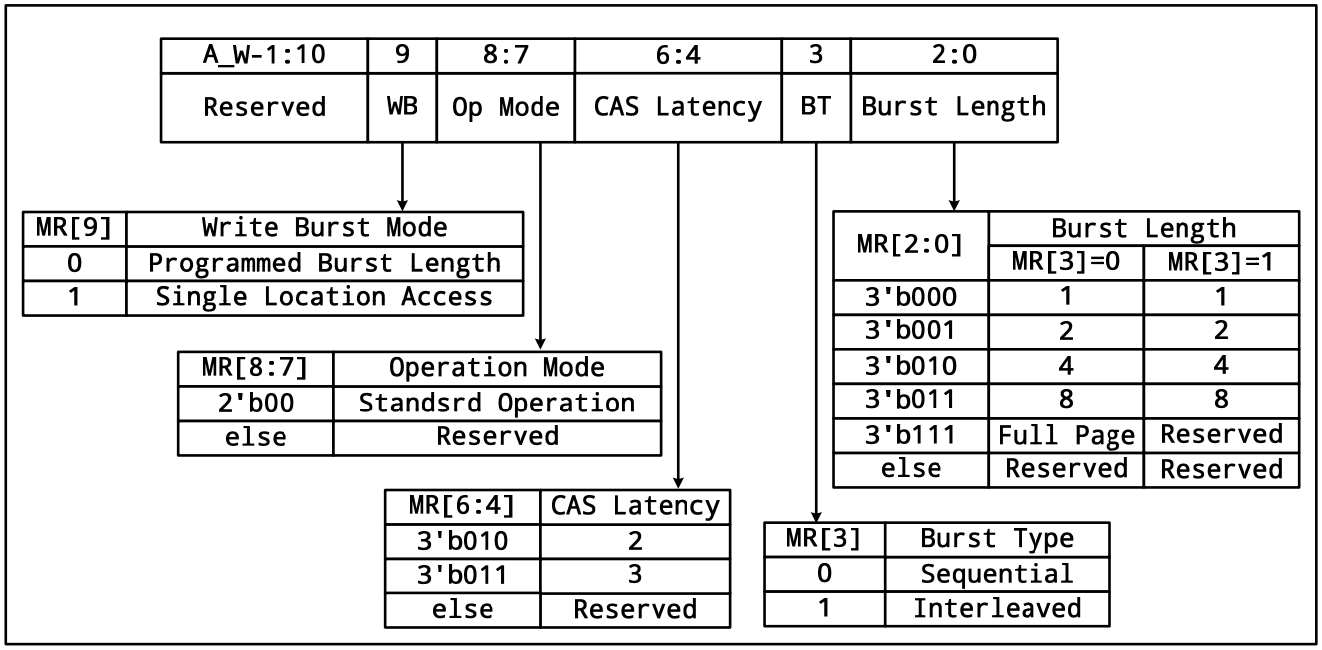
指令输入(cs_n、ras_n、cas_n、we_n、a10)。负责向 SDRAM 发送指令，不考虑休眠模式的指令真值表如下：

指令	缩写	cs_n	ras_n	cas_n	we_n	a10	地址
Load Mode Register	LMR	0	0	0	0	x	Op-Code
Auto Refresh	REF	0	0	0	1	x	x
Precharge	PRE	0	0	1	0	0	Bank/x
Precharge All	PREA	0	0	1	0	1	x
Burst Terminate	BT	0	0	1	1	x	x

Write	WR	0	1	0	0	0	Bank/Col
Write with autoprecharge	WRA	0	1	0	0	1	Bank/Col
Read	RD	0	1	0	1	0	Bank/Col
Read with autoprecharge	RDA	0	1	0	1	1	Bank/Col
Activate	ACT	0	1	0	1	1	Bank/Row
No Operation	NOP	0	1	1	1	1	x
Deselect	DSEL	1	x	x	x	x	x

1.2.1 LMR 指令与模式寄存器

模式寄存器的典型定义如下图：



通常，Write Burst Mode 取 0，Operation Mode 取 2'b00，Burst Type 取 0，而突发传送长度(Burst Length)、CAS 延时(Cas Latency)根据需要来选取。

Burst Length，即读写突发传送的周期数，一般可设置为 1、2、4、8 以及 Full Page，在初始化时通过写模式寄存器来设置。需要注意的是，如果列读写时，地址没有与 BL 对齐，SDRAM 会自行 Wrap 地址，导致意外的结果。

Cas Latency，即从发送列读取指令到有效数据出现的延时，一般为 2 或 3 个时钟周期，少数器件可以为 1 个周期，在初始化时通过写模式寄存器来设置。一般 CL=3 时，tAC 参数(即时钟上升沿到有效数据出现的延时)较小，从而使器件可以工作在较高的频率。

LMR(Load Mode Register)指令用于写模式寄存器，以设定 Burst Length、Cas Latency 等参数。进行 LMR 前，必须关闭所有的 Bank 中的行缓冲。发送 LMR 指令的同时，Bank 地址为 2'b00，模式寄存器的值通过地址总线发送。通常，只在初始化时对模式寄存器设定一次，之后

不再更改。LMR 指令之后，必须隔 tMRD 才能发送其他指令，当代 SDRAM 中 tMRD 的典型值为 2 个时钟周期。

从此处开始的讲解中，如非特殊说明，则默认 Write Burst Mode 取 0，Operation Mode 取 2'b00，Burst Type 取 0，突发传送长度(Burst Length)取 4，CAS 延时(Cas Latency)取 3。

1.2.2 ACT(Activate)指令、WR(write)指令、RD(Read)指令

ACT(Activate)指令。用于把指定 Bank 中的指定行放入该 Bank 的行缓冲里。发送 ACT 指令的同时，ba[1:0]给出 Bank 地址，地址总线发送行地址。如果需要读写的数据不在相应 Bank 的行缓冲里，就需要先发送 ACT 指令，载入相应的行，之后才能完成列读写。

ACT 指令之后可以进行列读写，但必须间隔 tRCD 的时间。不同 Bank 的 ACT 指令之间必须满足 tRRD 的时间间隔；相同 Bank 的 ACT 指令之间比满足 tRC 的时间间隔。

WR(Write)指令。用于向行缓冲中的指定列写入数据。发送 WR 指令的同时，ba[1:0]给出 Bank 地址，地址总线发送列地址，并给出第一个 word 的数据，之后的(Burst Length - 1)个周期里给出剩下数据。给出数据的同时可用 DQM 来做 Mask，1 为不写入，0 为写入。表示列地址时，跳过 a10 位，而 a10 为 0，表示是不在写操作后自动 Precharge。

RD(Read)指令。用于从行缓冲中的指定列读出数据。发送 RD 指令的同时，ba[1:0]给出 Bank 地址，地址总线发送列地址。有效数据在 CL 个周期后输出，在 Burst Length 个周期内，分别输出 Burst Length 个 word 的数据。读操作时，也可给出 DQM，时序与写操作类似，但是不常用到。表示列地址时，跳过 a10 位，而 a10 为 0，表示是不在读操作后自动 Precharge。

如果要访问的列在当前行缓冲中，就不必先执行 ACT 指令了。连续的读或写指令之间要满足 tCCD 间隔，tCCD 一般为 2 个时钟周期，如果 Burst Length 为 4 或 8，则连续的数据输入或输出不受其影响。所以说，开发空间局部性可以提高 SDRAM 访问效率。

从读切换到写的时候，由于数据总线的参数限制，可能要添加一个 Bus Turn Around 的周期，视具体情况而定。

1.2.3 PRE(Precharge)、PREA(Precharge All)、WRA(Write with autoprecharge)、RDA(Read with autoprecharge)指令

PRE(Precharge)指令。用于关闭指定 Bank 的行缓冲，并写回数据。发送 PRE 指令的同时给出 Bank 地址，A10 为 0，表示只对当前 Bank 预充电。

PREA(Precharge All)指令。用于关闭所有 Bank 的行缓冲，并写回数据。发送 PRE 指令的同时，A10 为 1，表示只对当前 Bank 预充电。

PRE、PREA 指令之后必须隔 tRP 后才能进行相应 Bank 的列操作，tRP 典型值略小于 20ns，约对应 2~3 个时钟周期。

当由 WR 指令发起的写操作给出最后一个数据后，间隔 tWR 才能发送 PRE 指令。因为数

据写入行缓冲需要时间。tWR 也称 tDPL，典型值为 2 个时钟周期。因此写操作给出最后一个数据后，相隔 tWR+tRP 才能给出相应 Bank 的下一个列操作指令，这个延时简称 tDAL。由于 SDRAM 的流水线化设计，tWR 对同一行内的写后写、写后读无限制。

当由 RD 指令发起的读操作输出最后一个数据前的 CL-1 个周期可以发送 PRE 指令。后面的列操作只要满足 tRP 就行。

WRA(Write with autoprecharge)指令。特性与 WR 指令一样，只是给出指令时 A10 为 1，表示是不在写操作后自动 Precharge。其效果与“由 WR 指令发起的写操作给出最后一个数据后间隔 tWR 发送 PRE 指令”一致。

RDA(Read with autoprecharge)指令。特性与 RD 指令一样，只是给出指令时 A10 为 1，表示是不在读操作后自动 Precharge。其效果与“在由 RD 指令发起的读操作输出最后一个数据前的 CL-1 个周期发送 PRE 指令”一致。

1.2.4 REF(Refresh)指令

REF(Refresh)指令。用于通知 SDRAM 进行“自动”刷新，每次刷新对应一行(row)。“自动”指的是 SDRAM 内部都包含刷新计数器，发送 REF 指令时，不需要发送对应的 Row 地址，故也把 Refresh 也称为 CBR(Column Before Row)操作。发送 REF 指令前，必须保证每个 Bank 都空闲，如果任意一个 Bank 中有打开的 Row Buffer，必须先发送 Precharge All 或 Precharge 指令。每次 Refresh 指令发送后，必须等待 tRFC 时间才能发送下一条指令，这就是刷新延时。

通用 SDRAM 要求在 64ms 内将所有 Row 刷新一遍，这个周期称为 tREF。SDRAM 的 row 数一般为 4096 或 8192，故如果均匀地刷新，每隔 15.625us 或 7.8125us 要发送一次 REF 指令。有时，即使 SDRAM 的 row 数不到 4096，刷新周期也按 4096 row 的情况计算。只要能满足 tRFC、tREF，连续的突发 Refresh 也是允许的。

2 SDRAM 的优化策略

SDRAM 的优化策略主要包括：行缓冲策略选取、地址重映射、访问请求重排、SDRAM 指令重排。决定 SDRAM 优化策略时的重要因素包括：带宽利用率、最大访问延时、存储器功耗。

行缓冲策略选取，与访问的空间局部性、延时限制有关。核心思想包括：Open Page Policy，Close Page Policy，各类折中优化的策略，以及一些动态调整 Page Policy 的技术。一般而言，Close Page Policy 适用于实时性要求高或者空间局部性差的应用场合；Open Page Policy 适用于实时性要求不高且空间局部性好的应用场合。

SDRAM 指令重排，着重发挥 SDRAM 的多 Bank 并行、流水线化的特性，从而提升总线效

率。

访问请求重排、地址重映射，是为行缓冲策略、SDRAM 指令重排服务的。可以向提高访问请求空间局部性的方向优化，从而提升 Open Page Policy 的效率；也可以使请求目标地址更分散的方向优化，从而便于 SDRAM 指令重排，充分发挥多 Bank 并行工作的优势。

下面对这些技术做简要讨论。

2.1 行缓冲与 SDRAM 优化策略

2.1.1 SDRAM 读写特征与行缓冲回顾

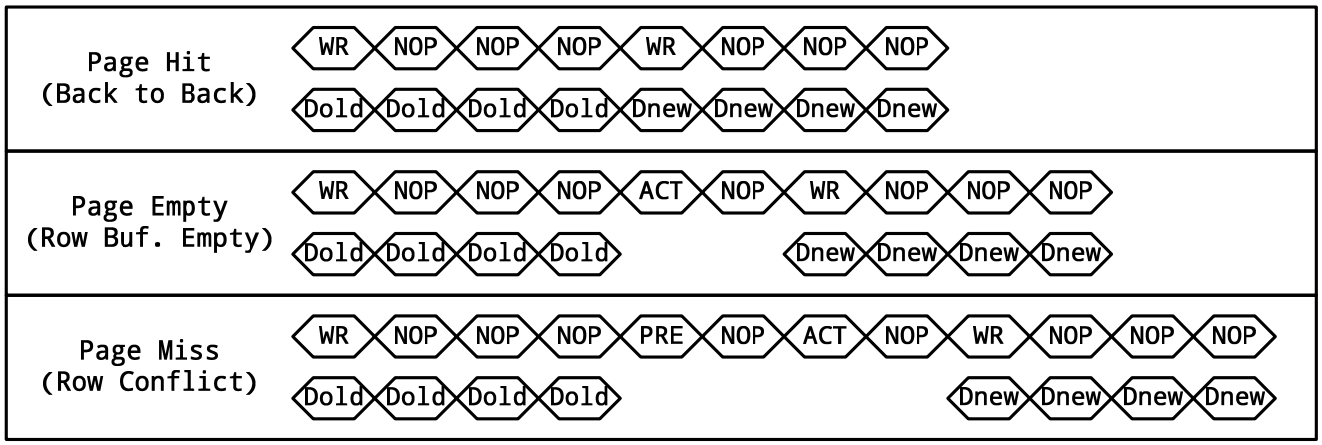
SDRAM 的四个 Bank 各有一个 Row Buffer。ACT 指令会将一行数据放入相应 Bank 的 Row Buffer 中；PRE、PREA 指令会关闭 Row Buffer 写回数据。如果某 Bank 的 Row Buffer 是关闭的，我们称它为空闲(Idle)的。SDRAM 的访问存在三种情况：

(1) Page Hit。当前访问的 Row，正好保存在相应 bank 的 row buffer 里。此时，不需要激活 Row，直接发读写指令和 Column 地址，就可以进行列读写，这就是 Back to Back 读写。此情况，读写延时最小。

(2) Page Empty。当前访问的 Row 所在的 bank 中，row buffer 关闭。此时，需要先激活 Row，再发读写指令和 Column 地址。此情况，读写延时中等。

(3) Page Miss。当前访问的 Row 所在的 bank 中，row buffer 打开，存的是另一个 Row 的数据。此时，必须先发送 Precharge 或 Precharge All 指令关闭 row buffer，再激活 Row，最后发读写指令和 Column 地址。此情况，读写延时最大。

假设 PRE 指令延时 tRP 为 2 个时钟周期、ACT 指令延时 tRCD 为 2 个时钟周期，则以上三种情况的写时序简图如下：



从图中可以看出，Page Hit 不导致带宽损失，Page Empty 的带宽损失较小，Page Miss 导致的带宽损失最大。需要注意的是，在进行指令重排优化的情况下，多 Bank 并行工作，PRE

指令延时、ACT 指令延时可以被“隐藏”，不一定会导致带宽损失，详细情况将在后文叙述。

2.1.2 Close Page Policy 简介

Close Page Policy 指的是尽量保持行缓冲关闭的策略。对当前行访问结束后立刻进行 Precharge 操作，不论后面是否还要访问该行。Close Page Policy 中只发生一种情况：Page Empty(即行缓冲为空，需要先激活行)。其读延时为 tRCD+CL，写延时为 tRCD。

Close Page Policy 的读写延时适中并且恒定，与访问请求的时间空间局部性无关，具有良好的可预测性(Predictabilility)，适用于实时性要求较高的应用场合。

Close Page Policy 的硬件实现比较简洁，因为每次读写都是一个行激活、列读写、预充电的过程。

2.1.3 Open Page Policy 简介

Open Page Policy 指的是尽量保持行缓冲打开的策略。除非需要访问当前同一 bank 中的其它行，则不进行 Precharge 操作。Open Page Policy 中常会发生两种情况：Page Hit(即再次访问 Row Buffer 中的数据)、Page Miss(Row Buffer 中的行不是要访问的行)。前者读延时为 CL，写延时为 0；后者读延时为 tRP+tRCD+CL，写延时为 tRP+tRCD。

如果访问请求的时间空间局部性很好，则发生 Page Hit 的概率远大于发生 Page Miss 的概率，应用 Open Page Policy 可以大大减少读写延时；如果访问请求的时间空间分部很分散，则发生 Page Miss 的概率远大于发生 Page Hit，应用 Open Page Policy 会造成较大的读写延时。

假设 Row Buffer 中的数据能被下次访问用到的概率为 x，则 Open Page Policy 优于 Close Page Policy 的判断条件如下：

$$x*0+(1-x)*(t_{RP}+t_{RCD})>1*t_{RCD}$$

即当 x 大于 tRP/(tRP+tRCD)时，Open Page Policy 的平均读写延时低于 Close Page Policy。

Open Page Policy 的硬件实现比 Close Page Policy 略复杂一些，需要判断 Page Hit 与否，还需要保证 Row Buffer 开启时间不超过 tRAS(max)的限制以防数据丢失。

2.1.4 Open Page Policy 与 Close Page Policy 的进一步讨论

由于 Row Buffer 的开启与关闭消耗的能量比较多，Close Page Policy 的功耗大于 Open Page Policy，且访问的局部性越好，差距越明显。

对于轻量级的应用，Close Page Policy 具有读写延时固定、电路简洁的优势，Xilinx 公司在《Synthesizable High Performance SDRAM Controller》中给出了一个不错的设计方案。

PC 应用以及单核 SOC，往往具有较好的访问请求局部性，多采用 Open Page Policy 或类似的衍生方案。Altera 公司为其 NIOS 软核设计的 SDRAM IP 就是一个典型的例子。

对于图像处理的应用，存储器的访问是很有规律性的，这给访问的优化带来了方便。在每次访问的同时给出连续访问长度等信息，即可实现存储器的高效访问，这也是接下来的几章中将要详细讨论的设计方案。

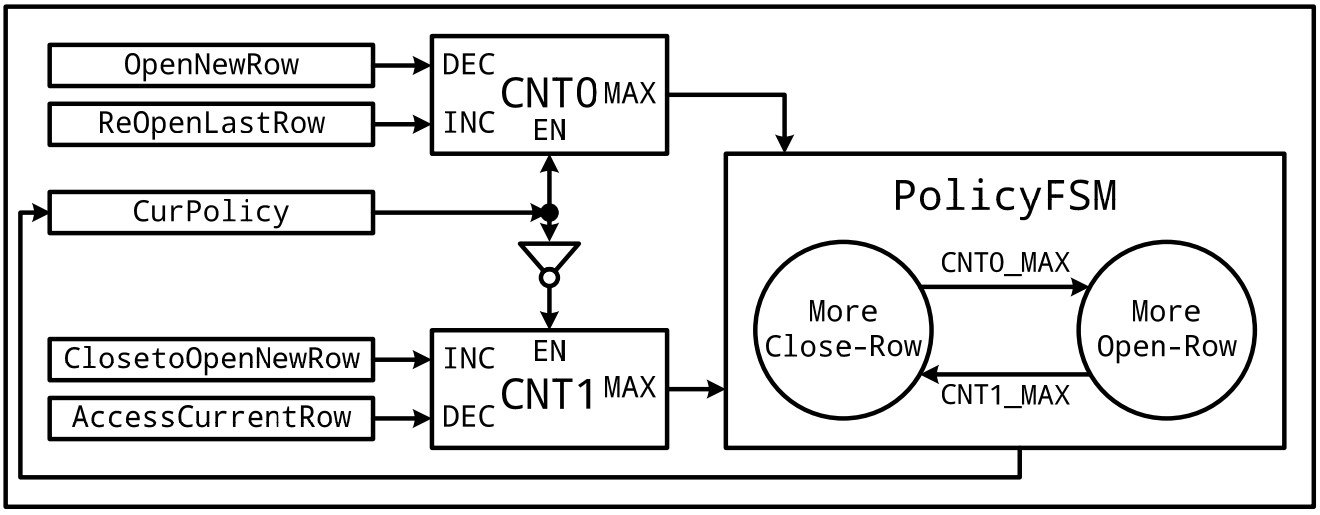
在多核应用、网络服务器应用中，由于访问请求的来源较为分散导致局部性下降，似乎更倾向于 Close Page Policy，但实际应用中往往采用更复杂的动态行缓冲策略。

2.1.5 动态的行缓冲策略

动态的行缓冲策略，即根据运行中前一段时间的访问历史动态决定 SDRAM 行缓冲策略的技术。

Intel 在其 2003 年的一份专利中提出了如下思想：假设存在偏向于 Open-Row 和偏向于 Close-Row 的两种策略。若当前策略偏向于 Open-Row，则对下列情况进行双向计数：关闭当前行以打开新行、访问当前行，如果前者接二连三地出现并超过阈值，则更换策略；若当前策略偏向于 Close-Row，则对下列情况进行双向计数：打开新行、打开上一次操作中被关闭的行，如果后者接二连三地出现并超过阈值，则更换策略。

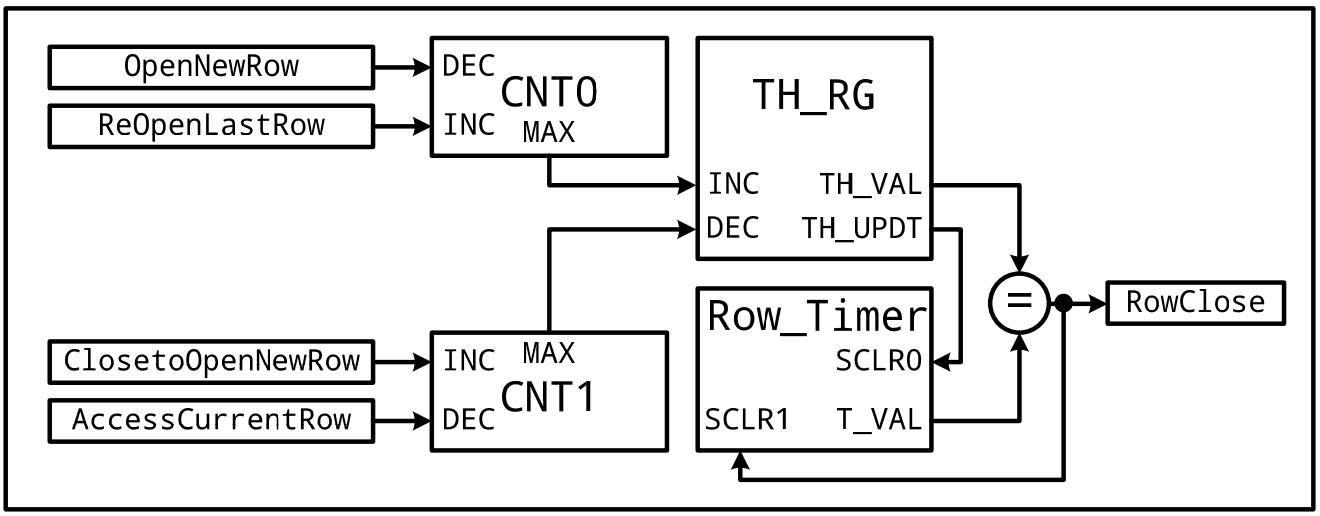
不难发现，如果两个计数器的阈值都设为 2，则整个结构与计算机体系结构课上讨论的 2-bit 分支跳转预测器类似。这种设计的简化框图如下：



AMD 在其 2005 年的一份专利中，阐述了另一种典型的动态行缓冲策略。它是基于定时器和动态计数阈值实现的，依然采样如下四个事件：关闭当前行以打开新行、访问当前行、打开新行、打开上一次操作中被关闭的行。如果“关闭当前行以打开新行”接二连三地发生，则下调动态计数阈值；如果“打开上一次操作中被关闭的行”接二连三地发生，则上调动态计数阈值。

每当定时器数值到达动态计数阈值时，定时器清零。另外，更新动态计数阈值时，也要清零定时器。行缓冲关闭操作，在定时器数值到达动态计数阈值时触发，其他时间不进行任何关闭行缓冲的操作。

基于定时器的设计有两个好处：SDRAM 的行缓冲开启有时间极限，此设计中只需给定动态计数阈值的上限即可满足此要求；此设计对应的电路相对简单，不必为各种策略设计不同的状态机与控制逻辑。此设计的简单框图如下。



Utah 大学的研究人员在 2011 年提出了一种基于请求预测的动态行缓冲策略。请求预测存储在行地址连续访问次数表中。首次访问时保持行缓冲打开，记下行地址、连续访问次数；如果又一次访问该行，则预测将发生相同次数的连续访问。

如果连续访问次数预测过低(发生同一行关了又开的情况)，则保持行缓冲打开直到连续访问结束，更新连续访问预测值；如果连续访问次数预测过高(提前开始访问其他行)，则将连续访问预测值减 1。

行地址连续访问次数表，采取 FIFO 形式，只对最近访问过的行做记录，以减少硬件开销。

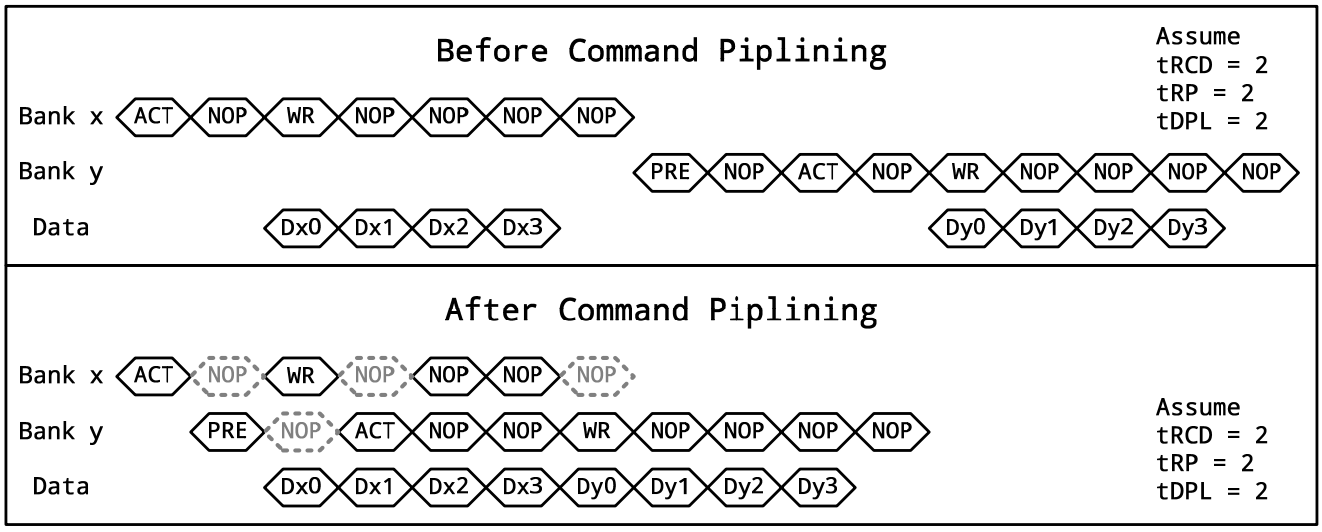
Utah 大学的研究人员认为此方法优于之前提到的基于动态策略选择的方案和基于可调阈值定时器的方案，但是我认为这需要进一步硬件验证与跟具体的 Benchmark 测试。

当然不论采用什么样的动态行缓冲策略，都是在以控制器的功耗、面积为代价，换取性能(或者说带宽利用率、最大访问延时)的改善，小型应用往往不需要这么复杂的策略。动态行缓冲策略的实现粒度可分为：全局、Bank、行三个层次。以全局或 Bank 为粒度的实现十分普遍；以行为粒度，效果最好，但硬件代价巨大，可以改进为只对最近用到过的行做预测。

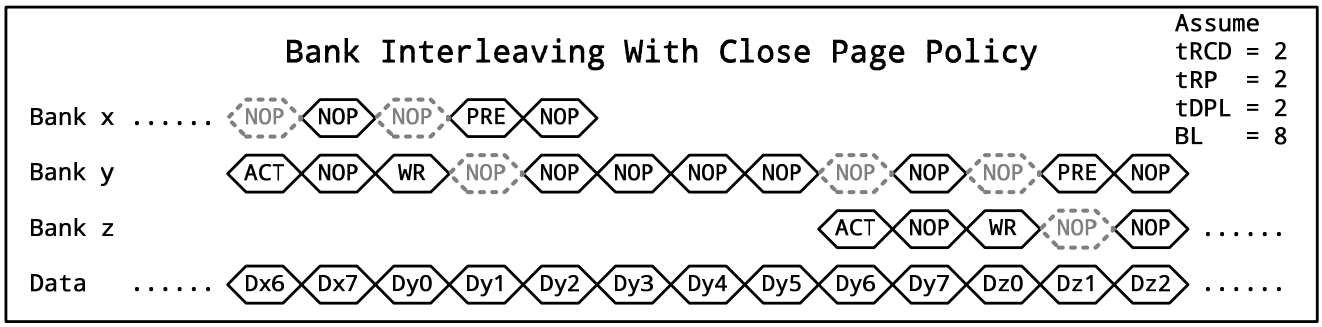
2.2 SDRAM 指令重排

SDRAM 具有四个 Bank，可以同时进行不同的操作。SDRAM 的操作也存在一些资源限制：每个 Bank 只有一个 Row Buffer；指令总线只有一条；指令之间要满足规定的延时；数据总线只有一条；数据总线从读切换到写可能需要等待。

SDRAM 的指令重排，与超标量 CPU 的乱序执行十分类似，都是在不产生数据冲突的前提下，尽快把可用资源调配给待执行的指令，用有意义的指令去替代 NOP 指令。下图给出了一个通过指令重排节省总线时间的实例。

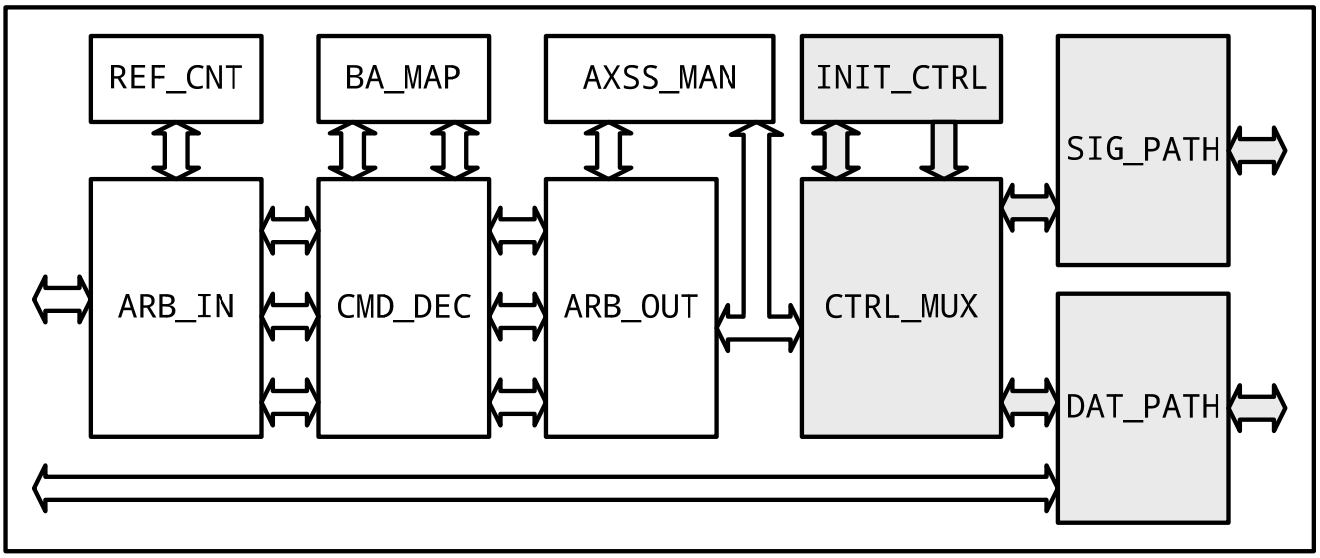


Bank 交错是一种简单有效的指令重排，常常与 Close Page Policy 合用。如果能保证相邻请求访问的 Bank 不同且读写方向一致，则通过此技术可以达到几乎 100%的带宽利用率。下图给出 Bank 交错的写时序示例。



Denis Shekhalev 在 OpenCore 网站上给出了一种结合 Open Page Policy 以及 SDRAM 指令流水线化的 SDRAM 控制器设计实例。该设计的读写请求预取深度为 3，在满足指令延时的

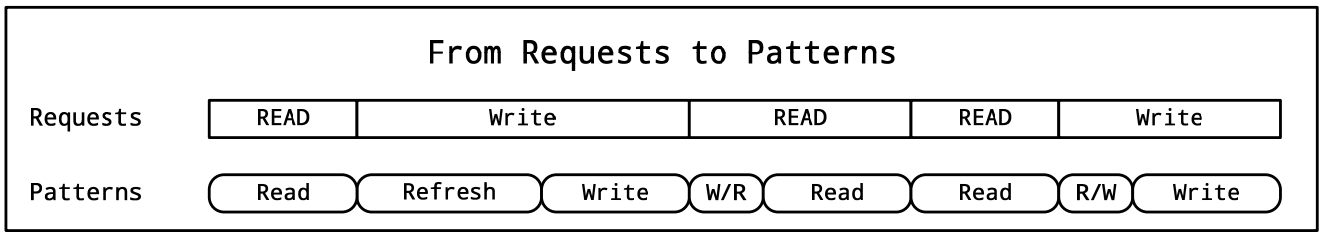
情况下，将第 1 条请求未用到的资源分配给第 2 条请求，将前两条请求都没用到的资源分配给第 3 条请求。其基本结构如下图。



此设计的关键部件为 Bank Map 以及 Access Manager。Bank Map 为 Open Page Policy 服务，记录 4 个 Bank 是否空闲，若不空闲则 Row Buffer 中存的是哪一行，从而判断当前读写请求需要用什么样的指令实现。Access Manager 用以判断 SDRAM 总线资源的可用性并保证指令之间满足规定的延时，有了它，就可以合理地实现 SDRAM 指令流水线化。

然而，像这样见缝插针的指令重排虽然可以有效地提高 SDRAM 带宽利用率，降低平均等待时间，但是也带来了请求响应时间的不可预测性，不利于大型系统中的整体性能分析。

荷兰的研究人员提出了一种基于“预先计算的访问 pattern”的解决方案，在保证响应时间可预测性的同时，达到了较高的性能。通过 Bank 交错构建较短的读 pattern、写 pattern、读写切换 pattern、写读切换 pattern。所有的读写请求都用这些 pattern 来实现，由于这些 pattern 的形式固定，读写请求的相应实现有较高的可预测性。读 pattern 和写 pattern 有近乎 100%的带宽利用率，并且同类 pattern 之间可以无缝拼接并保持 100%的带宽利用率。拼接 pattern 以完成系统端读写请求后，带宽利用率可能由于读写方向变化略有退化，但任然保持了较高的水平。请求响应简图如下：



2.3 SDRAM 地址映射方案

SDRAM 地址映射方案由两个优化方向：提高读写请求的时间空间局部性、增加 SDRAM

操作的可并行性。良好的地址映射方案，不会把原本空间连续的请求拆分到不同的 SDRAM 行中；良好的地址映射方案，会使连续的操作面向不同 Bank 中的不同行，从而为指令流水线化或多 Bank 交错读写提供可能。

地址映射方案一般是固定的，不可以在系统运行时更改。Utah 大学的研究人员在 2001 年提出了一种创新的 Impulse 内存控制器。一般计算机系统中，进程的 Virtual Address 需要在 TLB(Translation Lookaside Buffer)的帮助下转换为 Physical Address，再由内存控制单元转化为 DRAM 硬件地址。Impulse 内存控制器旨在直接完成 Virtual Address 到 DRAM 硬件地址的转换，并根据正在运行的应用动态选择优化的映射方案。该设计思路新颖、系统性强，可惜由于实现复杂，至今未得到广泛应用。

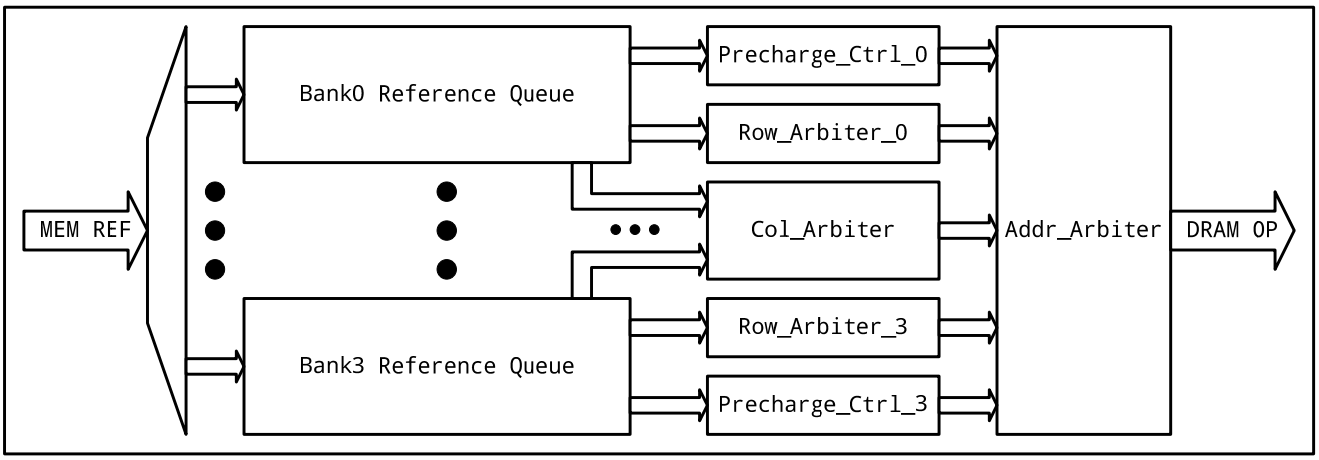
2.4 访问请求调度

实际应用中，往往存在一个 SDRAM 访问请求队列，这使得 SDRAM 控制器可以提前知道接下来的若干条指令。如果合理地重排这些指令，不但可以提高访问的时间空间局部性，而且能够促进 SDRAM 指令的并行化执行。以下是两种常见的访问请求调度方案。

以 Bank 为单位的 Round Robin 调度。即为每个 Bank 建立一个请求队列，然后依次从每个队列中取出一个请求来执行，没有就跳过，如此循环往复。这种调度方案简洁明了、实现简便、可预测性强、不会导致“请求饥饿”，而且由于它利于多 bank 的交错工作，特别适合于流水线化的 Close Page Policy 设计。

以 Bank 为单位的加权调度。依然为每个 Bank 建立一个请求队列，然后循环执行，但是等待请求较多的队列被优先执行。此方案更适合在访问局部性较好的情况下，与 Open Page Policy 配合。

Scott Rixner 在 ISCA2000 上提出了一种结合了访问请求调度与 SDRAM 指令重排的优化方案。该方案任然每个 Bank 建立一个请求队列，然后分行激活、列访问、预充电三个指令单元进行仲裁，最后重排指令发给 SDRAM。其结构框图如下：



该研究中，提到了三个仲裁优先级的考虑因素：

[1] 等待阈值，指令单元等待超过一定阈值后优先级提升，以防“请求饥饿”；
[2] 次序，旧的指令单元有较高的优先级；
[3] 读优先，优先满足读请求，减少系统延时。
两种预充电策略：
[1] Open，预充电的条件为：队列中不再有对当前行的访问，且存在对其他行的访问。
[2] Closed，预充电的条件为：队列中不再有对当前行的访问。
两种行、列仲裁策略：
[1] Most Pending，选择队列中访问请求最多的行或列进行操作，发掘局部性优势。
[2] Fewest Pending，选择队列中访问请求最少的列，尽快释放资源。

The University of Texas at Austin 的研究人员，在 MICRO2004 中指出了 Scott Rixner 方案的两个不足：做短期优化不考虑长期影响、与 CPU 需求结合不紧密。该团队针对 IBM Power5 处理器，提出了基于“访问历史”、“CPU 读写平均比例”的动态调整的请求仲裁方案，在实测中获得了更好的效果。

提高请求队列的规模，可以使访问请求调度的效果更好，但是这也加大了硬件开销、使数据冲突的检测更为复杂，所以实际应用中需要折中考虑。

需要指出的是，访问请求仲裁一般适用于访问请求密度较大、可用 bank 较少的场合；如果请求密度较小、可用 bank 较多，直接依照访问请求队列的顺序执行，也不会有大的性能损失。

2.5 访问请求仲裁

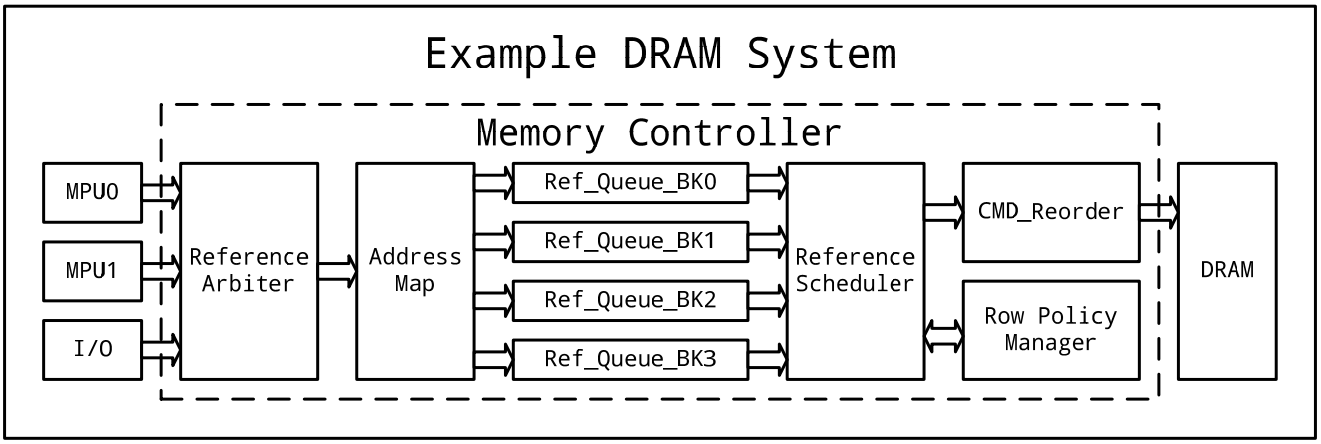
在多核或较复杂的系统中，同一块 SDRAM 硬件需要响应不同来源的请求，而这些请求的带宽要求与优先级互不相同，此时就需要进行请求仲裁。请求仲裁有很多形式，公平性、平均性能、最差情况性能、可预测性各有千秋，该话题超出了本课题的研究范围，此处不做深入讨论。

2.6 SDRAM 访问优化策略回顾

在前面的小节里，我们讨论了行缓冲策略、SDRAM 指令重排、SDRAM 地址映射方案、访问请求调度、访问请求仲裁五个主题。实际上，SDRAM 的各种访问优化技术不是孤立的，往往要相互配合才能获得较好的效果。

这里的讨论不仅适用于 SDRAM，也适用于 DDR、DDR2、DDR3 SDRAM；Rambus 公司的 DRAM 以其 Bank 数量多著称，更易于通过并行化提高效率，也更适合采用 Open Page Policy，但要知道 Bank 中的 Row Buffer 都是由灵敏放大器构成的，高昂的成本使其产品至今未得到广泛应用。

作为回顾，此处给出一张 SDRAM 控制系统的示意图：



3 本章参考文献

[1] Micron. 64Mb x4, x8, x16 SDR SDRAM. 2012

[2] B. Fanning. Method for Dynamically Adjusting a Memory System Paging Policy. 2003. United States Patent, Number 6604186-B1

[3] B. Sander, P. Madrid, and G. Samus, Dynamic Idle Counter Threshold Value for Use in Memory Paging Policy. 2005. United States Patent, Number 6976122-B1

[4] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. Memory access scheduling. ISCA2000: p128–138

[5] Ibrahim Hur, C. Lin. Adaptive History-Based Memory Schedulers. MICRO37: p343-354

[6] Lixin Zhang, Zhen Fang, M. Parker, B. K. Mathew, L. Schaelicke, J. B. Carter, W. C. Hsieh, S. A. McKee. The Impulse memory controller. IEEE Transactions on Computers 2001: p1117-1132

[7] B. Akesson, K. Goossens, M. Ringhofer. Predator: A predictable SDRAM memory controller. (CODES+ISSS)2007: p251-256

[8] B. Akesson, K. Goossens. Architectures and Modeling of Predictable Memory Controllers for Improved System Integration. DATE2011: p1-6

[9] Chitra Natarajan, Bruce Christenson, Fayé Briggs. A Study of Performance Impact of Memory Controller Features in Multi-Processor Server Environment. WMPI2004: p80-87

[10] Seong-Il Park, In-Cheol Park. History-Based Memory Mode Prediction For Improving Memory Performance. ISCAS2003: volumn5, pV185-V188

[11] M. Awasthi, D. W. Nellans, R. Balasubramonian, A. Davis. Prediction Based DRAM Row-Buffer Management in the Many-Core Era. PACT2011: p183-184

[12] Denis Shekhalev. High Speed SDRAM Controller With Adaptive Bank Management and Command Pipeline. 2008

[13] Xilinx. xapp134, Synthesizable High Performance SDRAM Controller, 2000

[14] Altera. Embedded Peripherals IP User Guide. 2010.12: Ch2 SDRAM Controller Core

[15] Bruce Jacob, Spencer Ng, David Wang. Memory Systems: Cache, DRAM, Disk. 2007: Ch13 DRAM Memory Controller

[16] Memory Controllers for Real-Time Embedded Systems: Predictable and Composable Real-Time Systems. 2011: Ch2 Proposed Solution

第三章 基于 Close Page Policy 的内存控制器后端设计

本次的 SDRAM 控制器主要是为了配合实验室的 H.264 编码硬件。由于系统端的请求具有较强的规律性，决定采用轻量级的 close page policy 设计方案，并略作改进，以实现“可变长度的突发传送”。

本章分为三大部分 ：第一部分讨论 Close Page Policy 的实现，并做简单的带宽利用率分析；第二部分讨论 SDRAM 的初始化原理；第三部分给出基于 Close Page Policy 的内存控制器后端设计方案。

1 Close Page Policy 的原理与有效带宽分析

Close Page Policy 是一种简单有效的 SDRAM 控制器设计策略。它的性能与访问地址的空间分部无关，带宽利用率固定，不考虑 Refresh 影响时读写延时恒定，适合于实时性要求较高的应用。

它保证每次 SDRAM 访问都是读写延时中等的 Page Empty 情况，手段很简单：每次读写后自动 Precharge，关闭 row buffer。这样不会因为同一 Bank 内 Row 地址切换而有 Penalty，也不会因为访问地址的空间局部性有 Gain，所以性能稳定性好。但另一方面，大量的 Precharge 给 SDRAM 带来较大的功耗。

1.1 Close Page Policy 的实现与性能分析

典型的 Close Page Policy 操作，只涉及 Refresh、Activate & Read with Autoprecharge、Activate & Write with Autoprecharge 三种情况。以下讲解这三种基本操作，相关的时序参数，并推导出理想情况下的有效读写带宽公式。

注：此处先假设控制器实现时未使用 SDRAM 指令流水线。

1.1.1 Refresh 操作和它对有效带宽的影响

由于 DRAM 的 1T1C 结构存在漏电，必须周期性地刷新(Refresh)以持久保持数据。

Refresh 指令，简称 REF 指令，用于通知 SDRAM 进行“自动”刷新，每次刷新对应一行(row)。“自动”指的是 SDRAM 内部都包含刷新计数器，发送 REF 指令时，不需要发送对应的 Row 地址，故也把 Refresh 也称为 CBR(Column Before Row)操作。每次 Refresh 指令发送后，必须等待 tRFC 时间才能发送下一条指令，这就是刷新延时。

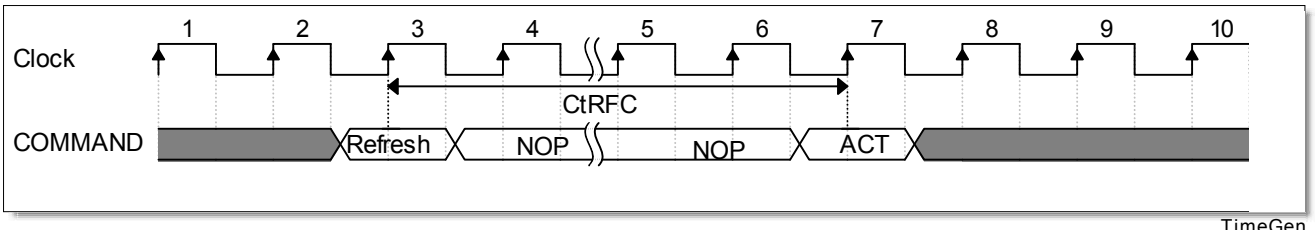
通用 SDRAM 要求在 64ms 内将所有 Row 刷新一遍，这个周期称为 tREF。SDRAM 的 row 数一般为 4096 或 8192，故如果均匀地刷新，每隔 15.625us 或 7.8125us 要发送一次 REF 指令。有时，即使 SDRAM 的 row 数不到 4096，刷新周期也按 4096 row 的情况计算。只要能满足 tRFC、tREF，连续的突发 Refresh 也是允许的。

发送 REF 指令前，必须保证每个 Bank 都空闲，如果任意一个 Bank 中有打开的 Row Buffer，

必须先发送 Precharge All 或 Precharge 指令。因为 Close Page Policy 以读写后立刻关闭 Row Buffer 为特点，所以不用考虑发送 REF 指令前关闭 Row Buffer 的问题。

按照器件手册的规定，AC 参数折合为时钟周期数时要向上取整。下文中，将以 CtREF 表示 tREF 对应的周期数，以 CtRFCm1 表示 tREF 对应的周期数减 1，其他以此类推，不再赘述。

刷新操作的时序图如下：



按时序要求，每 CtREF 个周期中，要有 SDRAM 行数乘以 CtRFC 个周期用于刷新或刷新延时，故推得刷新导致的带宽损失计算如下：

$$BWavailable = BW \times (1 - \frac{CtRFC \times RowNum}{CtREF})$$

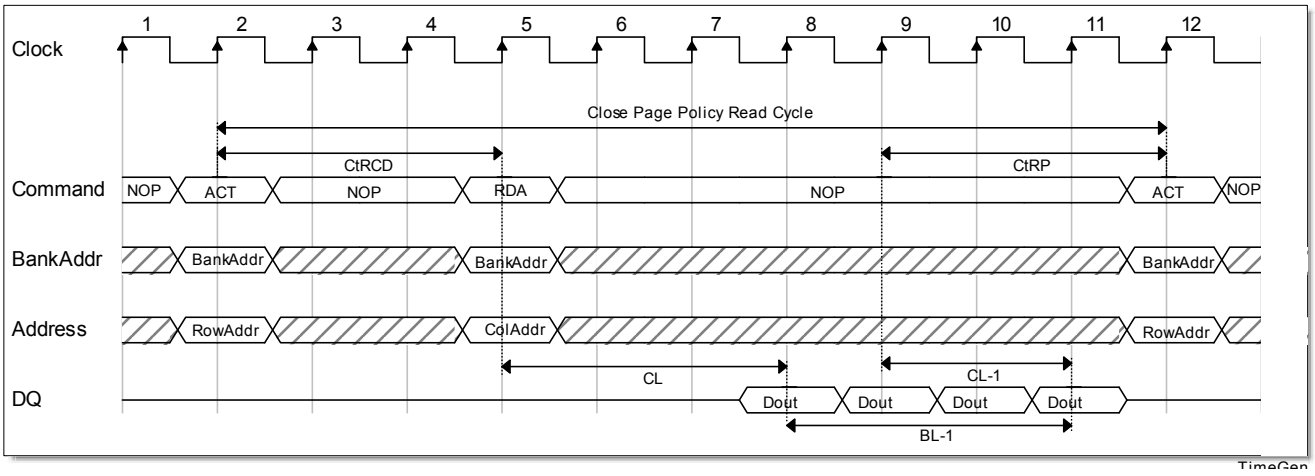
1.1.2 Activate & Read with Autoprecharge 操作与 Close Page Policy 的读效率分析

Close Page Policy 中，连续的读操作可以认为是 Activate 和 Read with Autoprecharge 操作的循环。

Activate 命令，简称 ACT 指令，用于激活 Bank 中的一个 Row。发此命令的同时，要给出 Bank 地址和 Row 地址。ACT 命令后要隔 tRCD 才能发送列读写指令。

Read with Autoprecharge 命令，简称 RDA 指令，作用是：读取数据，并自动执行 Precharge 关闭 Row Buffer。发此命令的同时，要给出 Bank 地址和 Column 地址。RDA 命令后，隔 CL 个周期才会有有效数据输出，输出是连续的。自动 precharge 发生在出现最后一个有效数据的时钟上升沿前的(CL-1)个周期。与单独的 Precharge 指令一样，自动 precharge 后要隔 tRP 才能发送下一条指令。

以 CL=3，BL=4 为例，画出整个读操作周期的时序图如下：



Close Page Policy 在每次读写后都关闭 Row Buffer，故其读延时恒定为 ACT 指令发送到第一个有效数据输出的延时，即 ACT 延时加 Cas Latency，也即 CtRCD+CL。

如上图，Close Page Policy 读周期中，ACT 指令到 RDA 指令的延时为 CtRCD 个周期；RDA 指令到下一个 ACT 指令的延时周期数为 CL + (BL - 1) - (CL - 1) + CtRP，即 BL + CtRP。故 Close Page Policy 连续读操作中，一个 ACT 指令到下一个 ACT 指令的延时周期数为 CtRCD + BL + CtRP，这就是 Close Page Policy 的读周期。而 Close Page Policy 的读周期中，只有 BL 个周期在传送数据，结合前文中 Refresh 操作的影响，推得理想的读有效带宽公式如下：

$$BWeffrd = BW \times \frac{BL}{CtRCD + BL + CtRP} \times (1 - \frac{CtRFC \times RowNum}{CtREF})$$

需要注意的一点是，SDRAM 允许指令与数据重叠，即使(CL - 1) < CtRP，前文中的公式依然成立。

更普适的带宽计算需要补充 4 点条件：

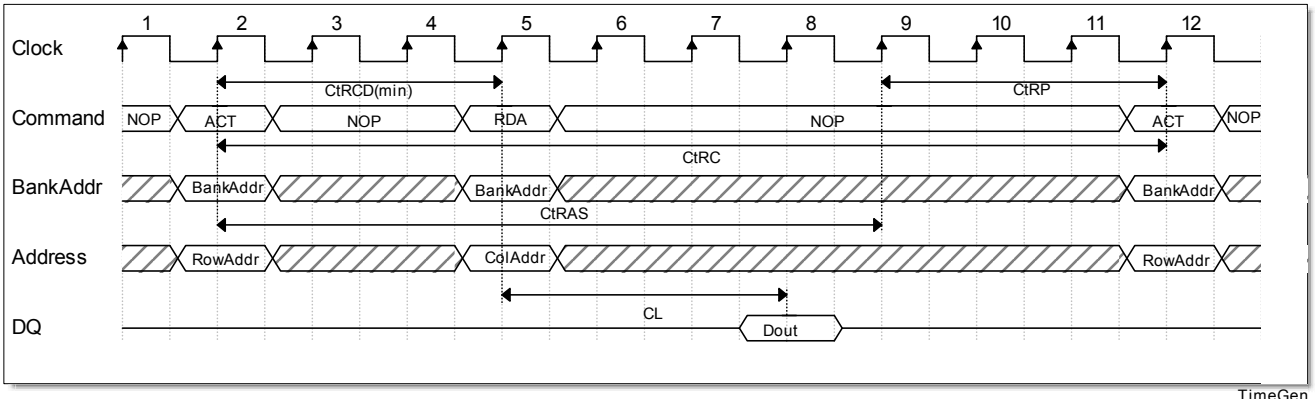
[1] ACT 指令与 PRE 指令之间必须相隔 tRAS(min)的时间，在 BL=1 或 2 时可能会成为限制条件；

[2] 一个 Row Buffer 被打开的时间不可超过 tRAS(max)。tRAS(max)远大于其它 AC 延时参数，除非时钟频率低到离谱，对 Close Page Policy 不产生约束；

[3] 同一 Bank 的两条 ACT 指令之间，必须相隔 tRC 的时间。一般 tRC 约等于 tRAS(min)+tRP，这就是 tRAS(min)起限制作用时，Close Page Policy 的“Read Cycle”，此时只考虑 tRAS(min)就行了。但是，此关系不一定总是成立，为了计算精确，也将其列入考虑范围；

[4] 不同 Bank 的两条 ACT 指令之间必须相隔 tRRD 的时间，这个时间与 tRCD 大致相同，对非流水线化的 Close Page Policy 不产生约束。

以 CL=3，BL=1 为例，画出整个读操作周期的时序图，以体现 tRAS(min)和 tRC 的影响：



从前文 CL=3，BL=4 的读时序图中，我们看到 ACT 指令到自动 Precharge 相隔的时钟周期数为：CtRCD + CL + (BL - 1) - (CL - 1) ，即 CtRCD + BL。如果这个值小于 CtRAS(min)，则自动 Precharge 会推迟以满足 tRAS(min)延时。同样，如果读时钟周期数 max(CtRAS, CtRCD + BL) + CtRP 超过了 CtRC，也必须延迟下一个 ACT 指令，以满足 SDRAM 的时序要求。综合

考虑后，推得修正后的理想读有效带宽公式如下：

$$BWeffrd = BW \times \frac{BL}{\max(CtRC, \max(CtRAS, CtRCD + BL) + CtRP)} \times (1 - \frac{CtRFC \times RowNum}{CtREF})$$

分析以上公式，可得出以下结论：

[1] **Close Page Policy 中 SDRAM 的读带宽利用率与 Cas Latency 无关！**

[2] **提高 Burst Length(BL)可以提高 Close Page Policy 的读带宽利用率。**

[3] 选择较高的时钟频率，可以提高硬件带宽，但也可能导致命令延时对应的周期数增加，故需要综合考虑。

[4] 因为器件速度等级越高，tRCD、tRP 参数越小，如果系统输出到 SDRAM 的时钟频率已定，tRCD、tRP 对应的时钟周期数 CtRCD、CtRP 也可能会小一些，故可以选择速度等级较高的器件，以提高带宽利用率。

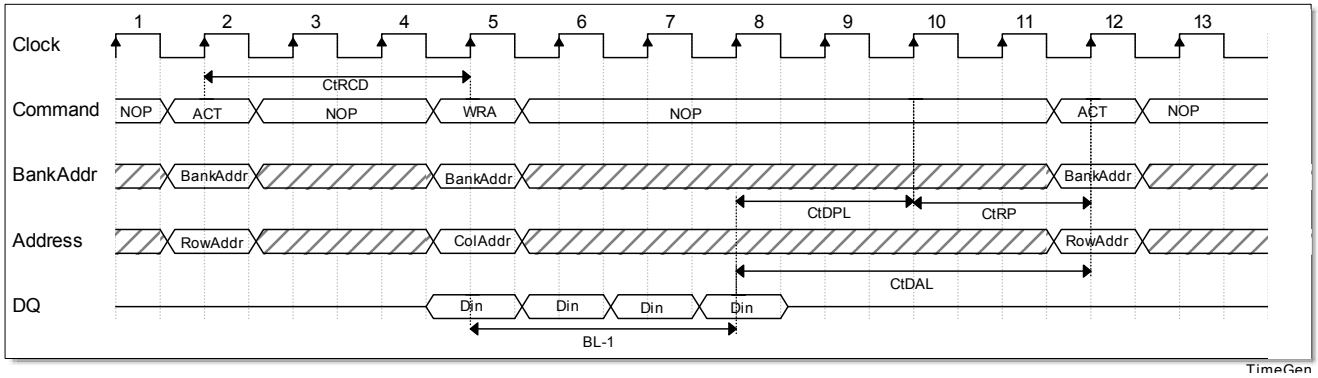
1.1.3 Activate & Write with Autoprecharge 操作与 Close Page Policy 的写效率分析

Close Page Policy 中，连续的写操作可以认为是 Activate 和 Write with Autoprecharge 操作的循环。

Activate 命令，简称 ACT 指令，用于激活 Bank 中的一个 Row。发此命令的同时，要给出 Bank 地址和 Row 地址。ACT 命令后要隔 tRCD 才能发送列读写指令。

Write with Autoprecharge 命令，简称 WRA 指令，作用是：写入数据，然后自动执行 Precharge 关闭 Row Buffer。发此命令的同时，要给出 Bank 地址和 Column 地址，以及第一笔数据；并在接下来的 BL-1 个周期内依次完成数据发送。自动 Precharge 发生在出现最后一个有效数据的时钟上升沿后的 CtDPL 个周期(也叫 tWR 延时)，之后 Precharge 延时参数 tRP 也必须被满足。有时候用 tDAL 标示 tDPL+tRP。

以 CL=3，BL=4 为例，画出整个写操作周期的时序图如下：



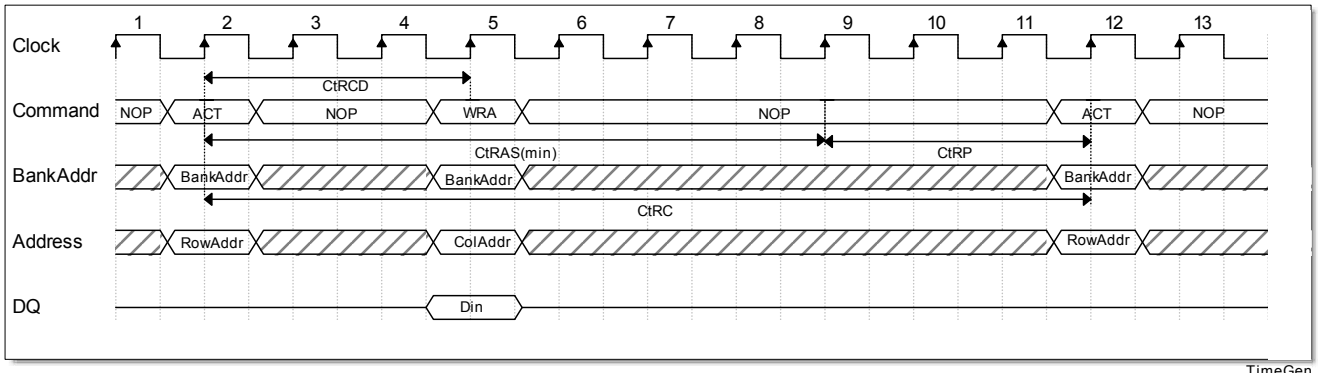
SDRAM 在发送 WRA 写指令的同时就可以发送第一笔数据。Close Page Policy 在每次读写后都关闭 Row Buffer，故其写延时恒定为 Activate 延时，即 CtRCD。

如上图，Close Page Policy 写周期中，ACT 指令到 WRA 指令的延时为 CtRCD 个周期；WRA 指令到下一个 ACT 指令的延时周期数为 (BL - 1) + CtDAL。故 Close Page Policy 连续

写操作中，一个 ACT 指令到下一个 ACT 指令的延时周期数为 CtRCD + BL - 1 + CtDAL，这就是 Close Page Policy 的写周期。而 Close Page Policy 的写周期中，只有 BL 个周期在传送数据，结合前文中 Refresh 操作的影响，推得理想的写有效带宽公式如下：

$$BWeffwr = BW \times \frac{BL}{CtRCD + BL - 1 + CtDAL} \times (1 - \frac{CtRFC \times RowNum}{CtREF})$$

与读操作一样，Close Page Policy 的写操作也受 tRC 和 tRAS(min)的制约。以 CL=3，BL=1 为例，画出整个写操作周期的时序图，以体现 tRAS(min)和 tRC 的影响：



从前面 CL=3，BL=4 的写时序图中，我们看到 ACT 指令到自动 Precharge 相隔的时钟周期数为：CtRCD + (BL -1) + CtDPL。如果这个值小于 CtRAS(min)，则自动 Precharge 会推迟以满足 tRAS(min)延时。同样，如果读时钟周期数 max(CtRAS, CtRCD + (BL -1) + CtDPL) + CtRP 超过了 CtRC，也必须延迟下一个 ACT 指令，以满足 SDRAM 的时序要求。综合考虑后，推得修正后的理想写有效带宽公式如下：

$$BWeffwr = BW \times \frac{BL}{\max(CtRC, \max(CtRAS, CtRCD + BL - 1 + CtDPL) + CtRP)} \times (1 - \frac{CtRFC \times RowNum}{CtREF})$$

分析以上公式，可得出以下结论：

[1] **提高 Burst Length(BL)可以提高 Close Page Policy 的写带宽利用率。**

[2] 选择较高的时钟频率，可以提高硬件带宽，但也可能导致命令延时对应的周期数增加，故需要综合考虑。

[3] 因为器件速度等级越高，tRCD、tRP 参数越小，如果系统输出到 SDRAM 的时钟频率已定，tRCD、tRP 对应的时钟周期数 CtRCD、CtRP 也可能会小一些，故可以选择速度等级较高的器件，以提高带宽利用率。

1.2 Close Page Policy 性能计算实例

1.2.1 Samsung 64Mb：x16 单片 SDRAM 计算示例

1.2.1.1 系统设定

系统只是用一片 Samsung 64Mb：x16 的 SDRAM 作为内存。

系统时钟 166MHz，要求 BL=4 或 8。SDRAM 速度等级为 166MHz，若取 CL=3，它的 AC

延时参数如下表：

tXX	tXX(ns)	CtXX(clks)	CtXXm1(clks)
tRCD	18	3	2
tRP	18	3	2
tRAS(min)	42	7	6
tWR(a.k.a. tDPL)	2clk	2	1
tDAL	2clk+tRP	5	5
tRC	60	10	9
tRFC	60	10	9
tRRD	12	2	1

此 SDRAM 的三位地址空间为：4bank * (2^12)Row * (2^8)Column * 16 bit = 64Mbit。需要在 64ms 内将 4096 行刷新一遍。

1.2.1.2 非流水线 Close Page Policy 性能计算

读延时 = CtRCD + CL = 3 + 3 = 6 clks

写延时 = CtRCD = 3 clks

回顾读带宽公式：

$$BWeffrd = BW \times \frac{BL}{\max(CtRC, \max(CtRAS, CtRCD + BL) + CtRP)} \times (1 - \frac{CtRFC \times RowNum}{CtREF})$$

BL = 4 时：

BWeffrd = BW * 4/10 * (1- 4096*60/64000000) = 39.85% * BW

BL = 8 时：

BWeffrd = BW * 8/14 * (1- 4096*60/64000000) = 56.92% * BW

回顾写带宽公式：

$$BWeffwr = BW \times \frac{BL}{\max(CtRC, \max(CtRAS, CtRCD + BL - 1 + CtDPL) + CtRP)} \times (1 - \frac{CtRFC \times RowNum}{CtREF})$$

BL = 4 时：

BWeffwr = BW * 4/11 * (1- 4096*60/64000000) = 36,22% * BW

BL = 8 时：

BWeffwr = BW * 8/15 * (1- 4096*60/64000000) = 53.13% * BW

1.2.1.3 一个有意思的假设

假设 Close Page Policy 中的 BL 可以为 16，重新做以上计算：

BWeffrd = BW * 16/22 * (1- 4096*60/64000000) = 72.45% * BW

BWeffwr = BW * 16/23 * (1- 4096*60/64000000) = 69.30% * BW

显然，当 BL=16，Close Page Policy 的带宽利用率进一步上升。虽然 SDRAM 的 BL 不可以设置为 16，但是连续传送 16 个 word 的数据并不是不可能，这也就是“系统端看见的突发传送长度”。详细的实现机理将在第三部分内存控制器设计方案中说明。

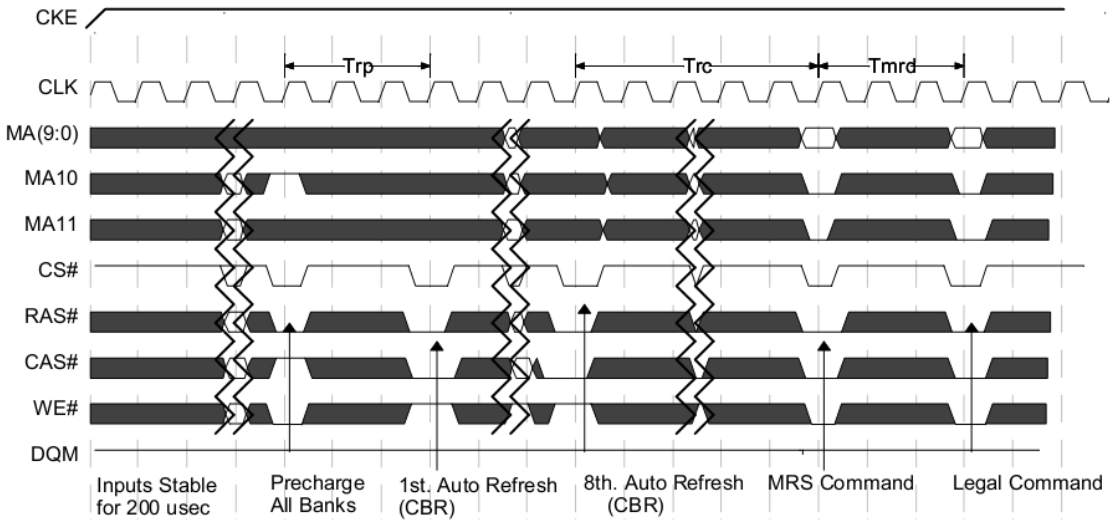
2 SDRAM 的上电初始化原理

SDRAM 必须要用规定的方式进行上电与初始化，以保证器件正常工作。初始化期间通过写模式寄存器指令(LMR)来完成 Burst Length、CAS Latency 等重要参数的设置。

2.1 Jedec 21-C 标准与 Intel PC100 标准的规定

早期的 SDRAM、当代的韩系、日系、台系 SDRAM 都符合这里的描述。为保证兼容性，本次设计采用这个传统初始化标准，但 tMRD 周期取为 2 以匹配当代的器件。

下图为 1999 年 Intel 定义的初始化时序图，取自 Intel PC100 标准：



上电步骤：

[1] 同时施加 VDD 与 VDDQ，并把 CKE 置为低。

[2] 等待电源稳定。

[3] 等待时钟稳定后把 CKE 置为高。

初始化步骤：

[1] 保持 200us 的 NOP 指令发送。

[2] 发送 Precharge All 指令。

[3] 发送至少 8 次 Refresh 指令。

[4] 发送 Mode register set 指令，设置模式寄存器。

备注：

[1] 初始化时不可忽视交流延时参数(trp、trfc、tmrs)，必要时要添加 NOP 指令。

[2] 当代器件一般规定 tMRS 为 2 个时钟周期，而不是 Jedec 或 PC100 中定义的 3 个周期。

[3] 对于 Samsung、Zentel 的器件，可以只 Refresh 两次。

[4] Elpida、Zentel 等公司建议在初始化时把 DQM 置为 1，以防止可能的数据冲突。这一点在 Intel 定义的时序图中也有体现。PC100 规定在第一条有效指令发送后，DQ 自动变为高阻，但这之前的状态未规定。

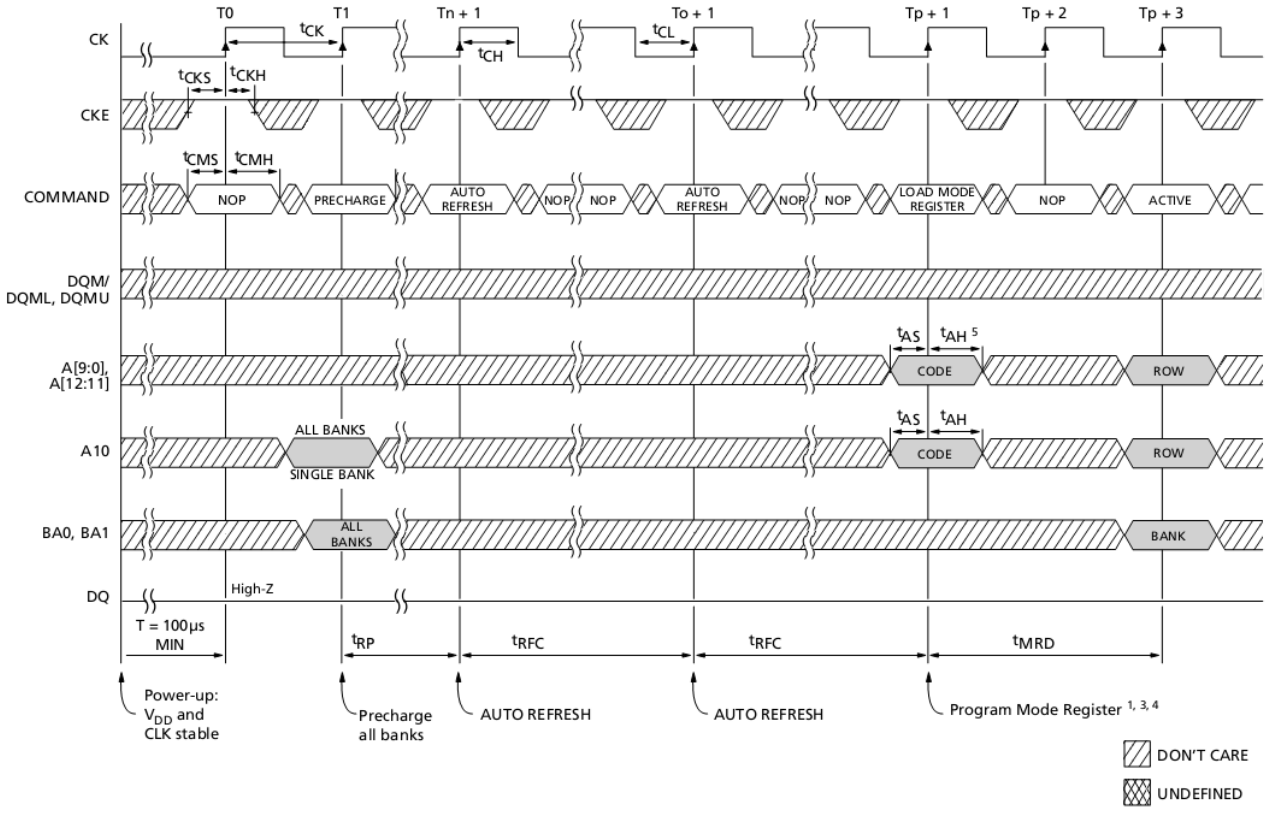
[5] 理论上只要所有 Bank 都空闲，就可以设置 Mode register。但一般情况下，初始化后 Mode register 不再需要改变。Mode register 中设置了 CAS Latency、Burst Length 等重要参数，这些配置信息通过地址总线传送，详细操作已在第二章的“LMR 指令与模式寄存器”部分说明。

2.2 Micron、ISSI 等当代美系 SDR SDRAM 的特点

Micron、ISSI 等公司的 SDRAM 产品中，时钟稳定后的初始等待时间缩短为 100us，同时 refresh 的次数减少为至少 2 次。对于 Micron 器件，时钟稳定后，CKE 可以在 100us 的等待期间上升，然后保持 NOP 发送；而对于 ISSI 器件，时钟稳定后，CKE、DQM 必须上升为 1，然后才开始 100us 的 NOP 发送。

Micron 为 SDRAM 做的 verilog 模型流传甚广，Xilinx、Altera、Lattice 等 FPGA 厂商提供的参考设计一般都遵循 Micron 手册中的初始化步骤。

下图为 Micron SDRAM 的初始化时序图，取自 Micron 的器件手册：



3 基于 Close Page Policy 的内存控制器后端设计方案

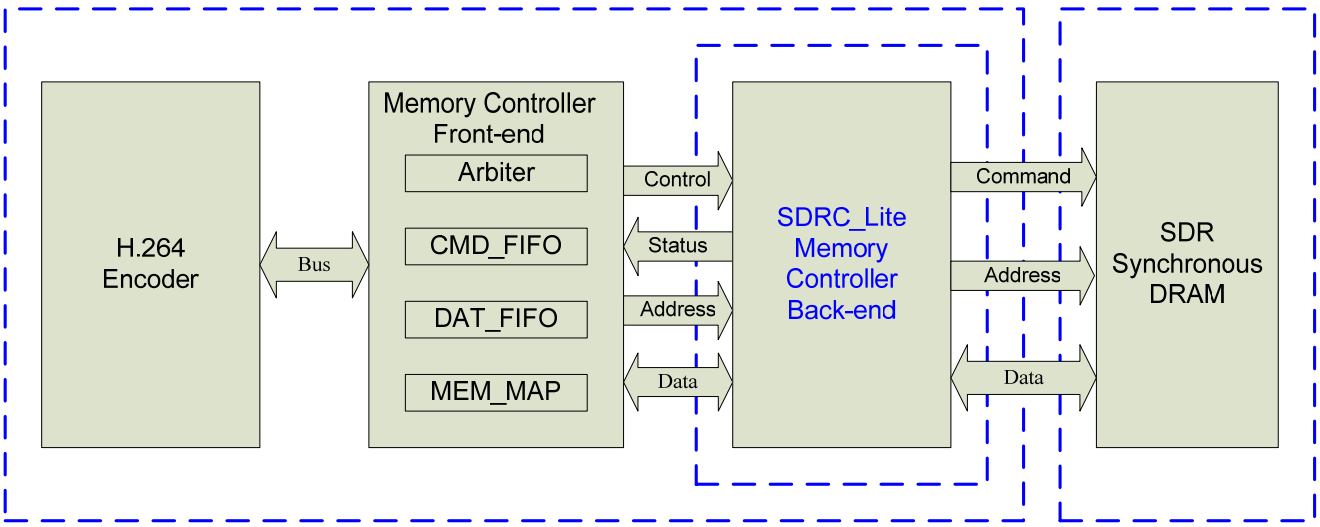
3.1 系统概述

本次 SDRAM 控制器设计的目的是为实验室的 H.264 编码系统提供 DRAM 存储接口支持。

本节描述的是底层 SDRAM 控制器，它相当于存储控制器的后端。在连接 H.264 视频编码系统前，还需要连接存储控制器的前端。存储控制器的前端，主要包括：地址映射、请求仲裁、数据 FIFO 和指令 FIFO，是一个跨时钟域的单元；存储控制器的后端，主要控制 SDRAM 的初始化、刷新与高效读写，工作频率与 SDRAM 操作频率相同。

H.264 编码应用与 SDRAM 存储系统的整体框图如下：

SDRC_Lite Memory Controller Back-end Beta2
System Block Diagram



3.2 SDRC_Lite 存储控制后端概述

本 SDRAM 控制器后端设计被定名为 SDRC_Lite，因为它是一个高效的轻量级设计。它的基本特点如下：

[1] 基于略作改进的 Close Page Policy 操作原理，能稳定地保持低读写延时、高带宽利用率。

[2] 全自动的 SDRAM 初始化、刷新控制。“初始化完成信号”引出，以供系统端参考。

[3] 为系统提供简化的读写接口，隐藏 SDRAM 操作细节。

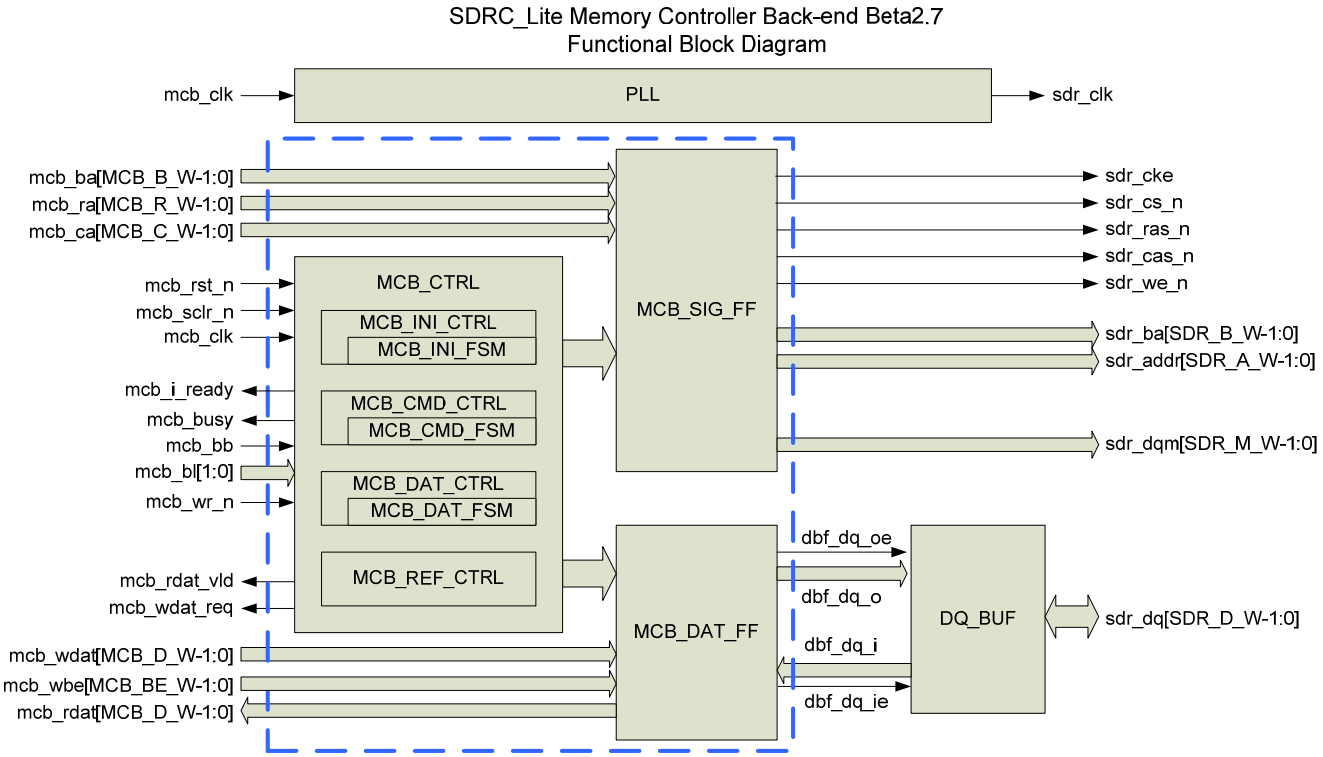
[4] 提供 Byte enable 接口，可以实现数据的选择性写入。

[5] 提供可动态调整的 Burst Length，每次发起读写时，都可以设置 Burst Length 为 4、8、12、16 中的任意值。

[6] 操作时序兼容 PC100 标准和工业界实施标准。

[7] 将数据位宽、地址位宽、SDRAM 器件的时序参数作为源代码中的 parameter，轻松适应不同的系统设置以及各厂商各型号的 SDR SDRAM 芯片。

SDRC_Lite 存储控制后端的功能框图如下：



3.3 存储控制后端顶层信号

SDRC_Lite 一端与片外的 SDRAM 芯片相连，另一端与存储控制前端或 Avalon Wrapper 等片上单元相连。其宽口位宽参数可调，基本参数包括 SDR_M_W、SDR_B_W、SDR_A_W、SDR_D_W，分别表示数据 Mask 位宽、Bank 地址位宽、行列地址位宽、数据位宽。

详细的端口列表如下：

Signal Name	Direction	Description
(Interface to SDR SDRAM)		
sdr_clk	Output	SDRAM clock
sdr_cke	Output	SDRAM clock enable
sdr_cs_n	Output	SDRAM chip select
sdr_ras_n	Output	SDRAM row address strobe
sdr_cas_n	Output	SDRAM column address strobe
sdr_we_n	Output	SDRAM write enable
sdr_dqm[SDR_M_W:0]	Output	SDRAM input/output mask
sdr_ba[SDR_B_W:0]	Output	SDRAM bank address inputs
sdr_addr[SDR_A_W:0]	Output	SDRAM address inputs
sdr_dq[SDR_D_W:0]	I/O	SDRAM data input/output

(Interface to Memory Controller Front-end or Avalon Wrapper)		
mcb_clk	Input	MCB clock
mcb_rst_n	Input	MCB asynchronous reset
mcb_sclr_n	Input	MCB synchronous reset
mcb_bb	Input	MCB burst begin
mcb_rw_n	Input	MCB data direction: 0-write, 1-read
mcb_bl[1:0]	Input	MCB burst length: 00-4, 01-8, 10-12, 11-16
mcb_ba[MCB_B_W-1:0]	Input	MCB bank address
mcb_ra[MCB_R_W-1:0]	Input	MCB row address
mcb_ca[MCB_C_W-1:0]	Input	MCB column address
mcb_busy	Output	MCB sdr sdram controller busy
mcb_rdat_vld	Output	MCB read data valid
mcb_wdat_req	Output	MCB data to write request
mcb_wdat[MCB_D_W-1:0]	Input	MCB write data
mcb_rdat[MCB_D_W-1:0]	Output	MCB read data
mcb_wbe[MCB_BE_W-1:0]	Output	MCB write data byte enable
mcb_i_ready	Output	MCB initialization done

3.4 设计备注

3.4.1 SDRC_Lite 的核心与外部模块的划分

SDRC_Lite 功能框图中蓝色方框内的为核心模块，在它外面还包括 DQ 三态缓冲、相移 PLL 这两个模块。

DQ 三态缓冲、相移 PLL 的实现形式，与实现内存控制器的工艺或器件有关，在 ASIC、Xilinx FPGA、Altera FPGA 上各不相同。比如，如果要用 50MHz 的时钟源产生两个相位相差为 3ns 的 100MHz 时钟。在 Xilinx Spartan-3E FPGA 中，需要将两个 DCM 串联使用；在 Altera Cyclone II FPGA 中，只需要调用一个 ALTPLL 的 megacore。

为了提高 SDRC_Lite 设计的可移植性，决定将 DQ 三态缓冲、相移 PLL 放在核心模块外部。

3.4.2 连接 SDRC_Lite 的片上单元

本次设计主要考虑以下几种 SDRC_Lite 连接情况：

[1] 与 H.264 编码系统中的存储控制前端相连，响应其读写请求。

[2] 与一个简单的跨时钟域 Wrapper 相连，通过若干 FIFO 实现可靠的跨时钟域的 SDRAM 读写。

[3] 通过 Avalon Wrapper 连接到 SOPC 系统中，为 NIOS II 软核或其他 Avalon 总线设备提

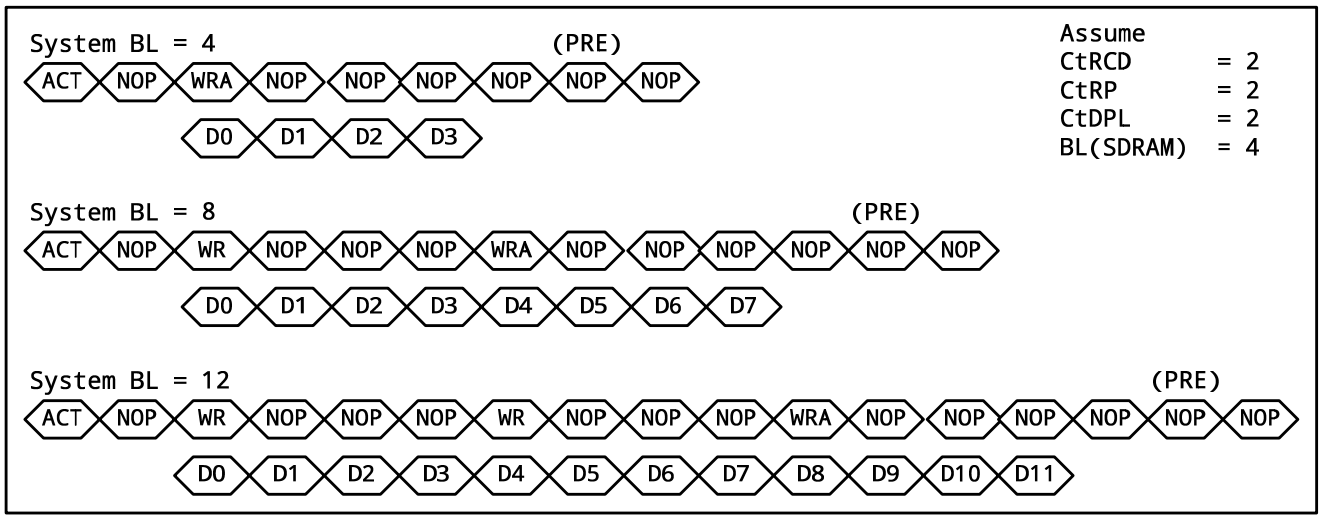
供存储支持。SDRC_Lite 核心与 Avalon Wrapper 一起构成一个通用的 Avalon Custom IP，可以在不同的 SOPC 设计中调用。关于 SOPC 集成的设计、测试、硬件验证，将在后续的章节中给出。

3.4.3 SDRAM 的 Burst Length 与 SDRC_Lite 核心(MCB)的 Burst Length

SDRC_Lite 设计中，SDRAM 的 CL(Cas Latency)确定为 3，BL(Burst Length)确定为 4，模式寄存器初值不是可调节参数。

系统端的 Burst Length，在每次系统端发起读写请求时通过 mcb_bl 设置，共有四个可选的值：4、8、12、16，分别用 2'b00、2'b01、2'b10、2'b11 表示。这 4 种 Burst Length 都是用长为 4 的 SDRAM 突发传送拼接而成的。

下面给出系统端 Burst Length 为 4、8、12 时的 SDRAM 写时序简图。



数据位宽为 64 bit(例如四块 x16 的 SDRAM 并联)时，长为 4、8、12、16 的突发传送，已足以满足 H.264 编码系统的单次数据请求，而该系统多次数据请求之间相隔的时间较大，不要求长时间的连续传输，故以上设计足以满足需求。

从广义上讲，这任然是基于 Close Page Policy 的设计，只是实现了“可变的突发传送长度”，提高了传送效率。

3.4.4 关于地址对齐

SDRAM 收到读写指令后，Row Buffer 中数量与 Burst Length 一致的一组列被选中，所有访问都发生在这些列中，如果遇到边界则会发生地址的 wrap。

假设 BL 为 4，列地址为 A[8:0]，则首先用[8:2]选出对应的 4 列。接着，从其中的第 A[1:0]列开始依次向高地址方向访问。如果遇到这 4 列的边界，则返回 A[1:0]=2'b00 的地址，之后继续向高地址方向访问。

整个过程如下表：

A[1:0]	Order of Burst(BL=4)
2'b00	0-1-2-3
2'b01	1-2-3-0
2'b10	2-3-0-1
2'b11	3-0-1-2

SDRC_Lite 中，将 SDRAM 的 Burst Length 设置为 4、Burst Type 设置为顺序，故也受到这样的限制，为了避免不可预测的结果，应当保证系统端给出的 Column Address 的最低 2 位为 0。

在后续的 Avalon Wrapper 设计中，通过 Mask 操作，避免了这一限制，详细原理将在后续章节讨论。

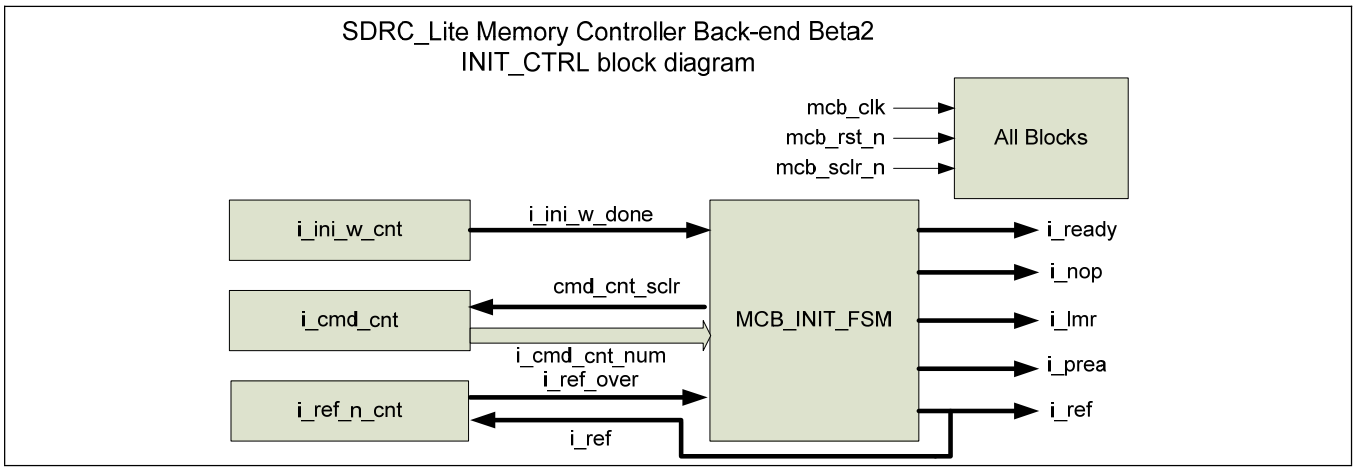
3.5 SDRC_Lite 存储控制后端核心控制逻辑的设计概要

MCB_CTRL 单元，负责存储控制后端核心控制逻。

它由 MCB_INI_CTRL、MCB_CMD_CTRL、MCB_DAT_CTRL、MCB_REF_CTRL 四个子模块构成，分别控制 SDRAM 的初始化、指令控制、数据通路控制、刷新控制，在这里给出简要的介绍。

3.5.1 上电初始化控制电路(MCB_INI_CTRL)的设计概要

本次设计使用的初始化控制电路，包括初始化状态机(MCB_INI_FSM)、初始化等待计数器(i_ini_w_cnt)、初始化 refresh 次数计数器(i_ref_n_cnt)、初始化命令延时计数器(i_cmd_cnt)四个模块。它们的连接关系如下图：



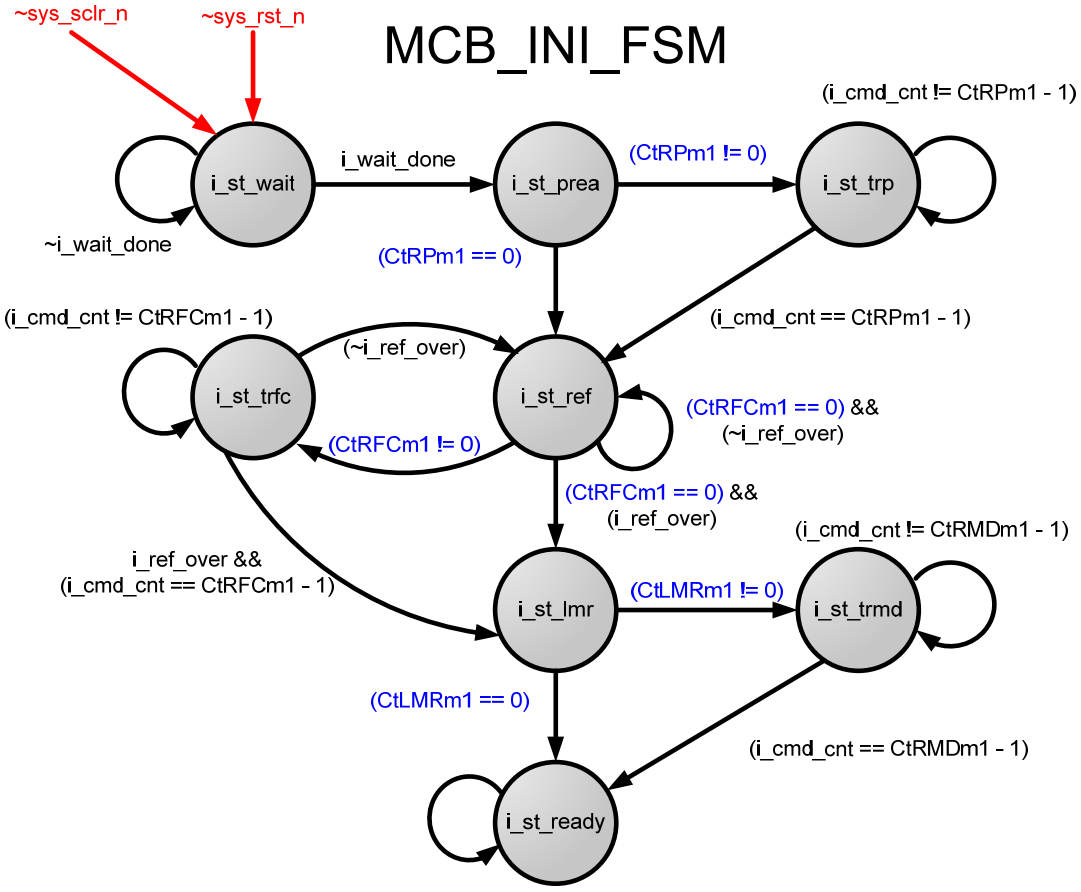
初始化等待计数器(i_ini_w_cnt)、初始化 refresh 次数计数器(i_ref_n_cnt)、初始化命令延时计数器(i_cmd_cnt)，与初始化状态机协同工作，共同完成初始化的控制。初始化等待计数器控制 100us 或 200us 的初始化等待；初始化 refresh 次数计数器控制 refresh 的次数；初始化命令延时计数器保证 trp、trfc、tmrd 等指令延时参数被满足。用户可以通过 verilog 参数改变这三

个计数器的设置。

初始化状态机(MCB_INI_FSM)。采用 Moore 模型，输出只与当前状态有关，以保证输出控制信号与时钟同步。状态机代码风格将采用 Altera 推荐的 2 段式写法，时序的状态更新为一段，组合的 Next State Logic 与 Output Logic 为一段。初始化状态机的状态跳转顺序基本按照 Jedec 的 SDRAM 初始化规定。此处假设在系统复位后，电源与时钟信号已经稳定，状态机主要控制初始化部分而非上电部分。各个状态与对应的输出如下表：

状态名	输出的指令信号	输出的其他控制信号
i_st_nop	(延时状态)	
i_st_prea	i_prea	i_cmd_cnt_sclr
i_st_trp	(延时状态)	
i_st_ref	i_ref	i_cmd_cnt_sclr
i_st_trfc	(延时状态)	
i_st_lmr	i_lmr	i_cmd_cmr_sclr
i_st_tmrd	(延时状态)	
i_st_ready	(初始化完成状态)	i_ready

初始化状态机(MCB_INI_FSM)的状态转换图如下：



注：上图中，蓝色的条件与器件的交流延时参数、系统工作频率有关，不会映射为电路逻辑。

trp、trfc、tmrd 分别是 Precharge All、Refresh、Mode Register Set 后的延时，在此期间不能发送 NOP 以外的指令。按照器件手册的规定，交流延时参数的单位从 ns 折算为时钟周期时，应该向上取整。假设 Trp 折合为 x 个时钟周期，那么从 i_st_prea 进入 i_st_trp 后，就应停留 x-1 个时钟周期（因为 i_st_prea 自身也占用了一个周期）。如果 x 恰好为 1，则 i_st_trp 状态不存在，电路参数决定了，x 的大小就决定了，是否可能进入 i_st_trp 只与电路参数有关，不会映射为电路逻辑。trfc、tmrd 对状态跳转的影响同理于 trp。

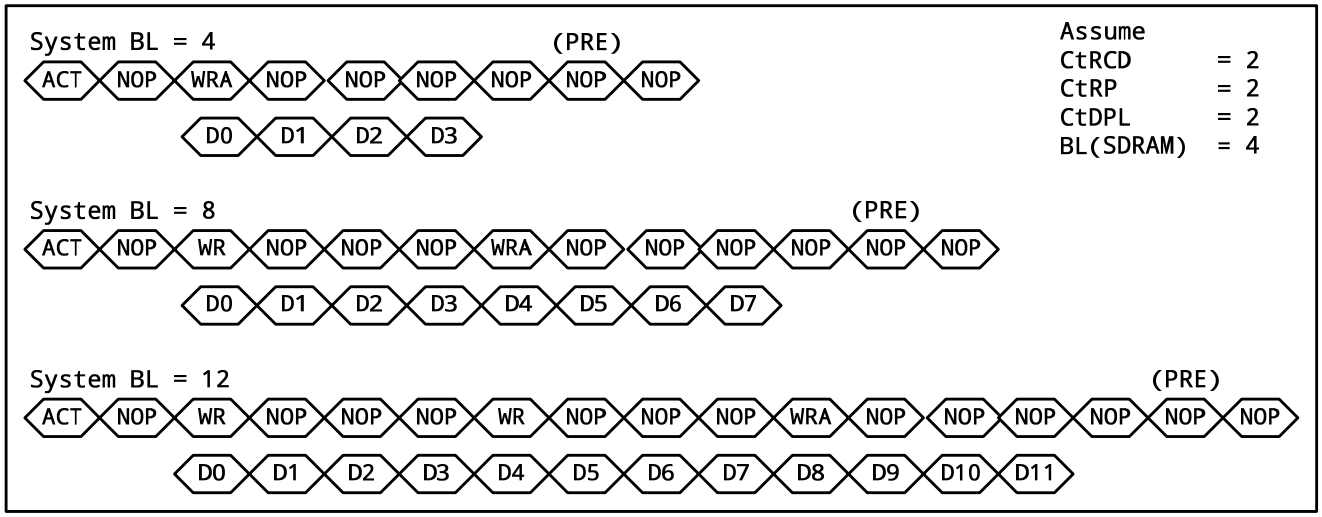
在系统复位时，MCB_INI_FSM 进入 IDEL 状态(即 i_st_nop)；完成 100us 或 200us 的初始化等待后，进行 Precharge All 操作；满足 trp 延时后，进行 Refresh 操作；满足 trfc 延时后，如果达到需要的 Refresh 次数，就进行 Mode Register 设置，否则，继续 Refresh 操作；Mode Register 设置后，经过 tmrd 延时，最终进入 ready 状态，初始化完成。

所有的延时等待状态中，都不发指令信号；i_st_prea 状态中发起 i_prea；i_st_ref 状态中发起 i_ref；i_st_lmd 状态中发起 i_lmr；这些指令信号将通过信号通路(MCB_SIG_FF)编码为 SDRAM 命令。初始化完成后，发出 i_ready，这之后用户的请求才会被响应。

3.5.2 命令控制电路(MCB_CMD_CTRL)的设计概要

命令控制电路，负责初始化完成后的 SDRAM 指令发送控制。它接受三类请求：系统端的读请求、系统端的写请求、刷新控制单元的刷新请求，根据当前状态将它们转换为指令信号，再由信号通路(MCB_SIG_FF)编码为 SDRAM 命令。

MCB_CMD_CTRL 依照改进的 Close Page Policy 设计而成。每次读写，都一个先行激活、再列访问、最后预充电(写回)的过程。所谓“可变的系统端突发长度”，其实就是在发送请求时指定了“连续列访问的数目”。不妨回顾一下之前给出的 SDRC_Lite 写时序简图：

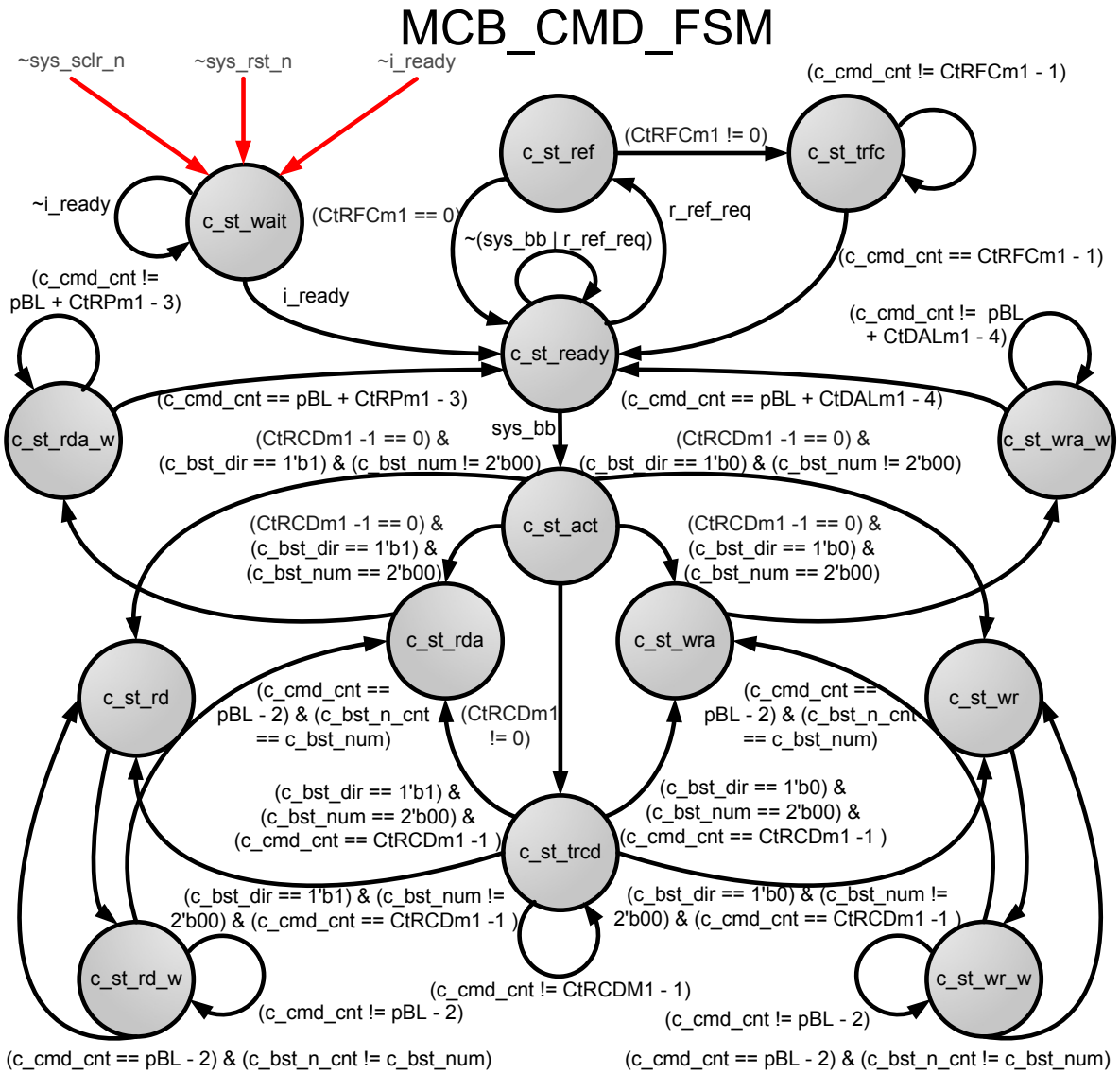


MCB_CMD_CTRL 接收到读写请求后，会把系统端突发传送长度(mcb_bl)、读写方向锁存到 c_bst_num、c_bst_dir 寄存器中。而 MCB_CMD_CTRL 中的 burst 计数器(c_bst_n_cnt)，用于记录当前系统请求中剩余的 SDRAM 突发传送数目。命令状态机(MCB_CMD_FSM)，根据

c_bst_n_cnt、c_bst_dir 决定发送什么样的指令。如果不是最后一次的 SDRAM 突发传送，则发送 RD 或 WR 指令；如果是最后一次的 SDRAM 突发传送，则发送 RDA 或 WRA 指令，以便在读写后自动完成 Precharge 操作。

MCB_CMD_CTRL 总是先完成当前的系统读写请求之后，再响应 MCB_REF_CTRL 的刷新请求。而当存在 MCB_REF_CTRL 的刷新请求时，SDRC_Lite 的 mcb_busy 有效，系统端不再发送新的读写请求直到刷新完成。这样的设计保证了读写请求一旦发起，便能被高效地执行。

命令状态机(MCB_CMD_FSM)。采用 Moore 模型，包含若干指令状态和指令等待状态，原理与 MCB_INI_FSM 类似此处不再赘述。在初始化完成之前，处于 c_st_wait 状态，不接受读写或刷新的请求；初始化、刷新、系统端突发读或系统端突发写结束后，进入 c_st_ready 状态，准备接受新请求。完整的状态转化图如下：



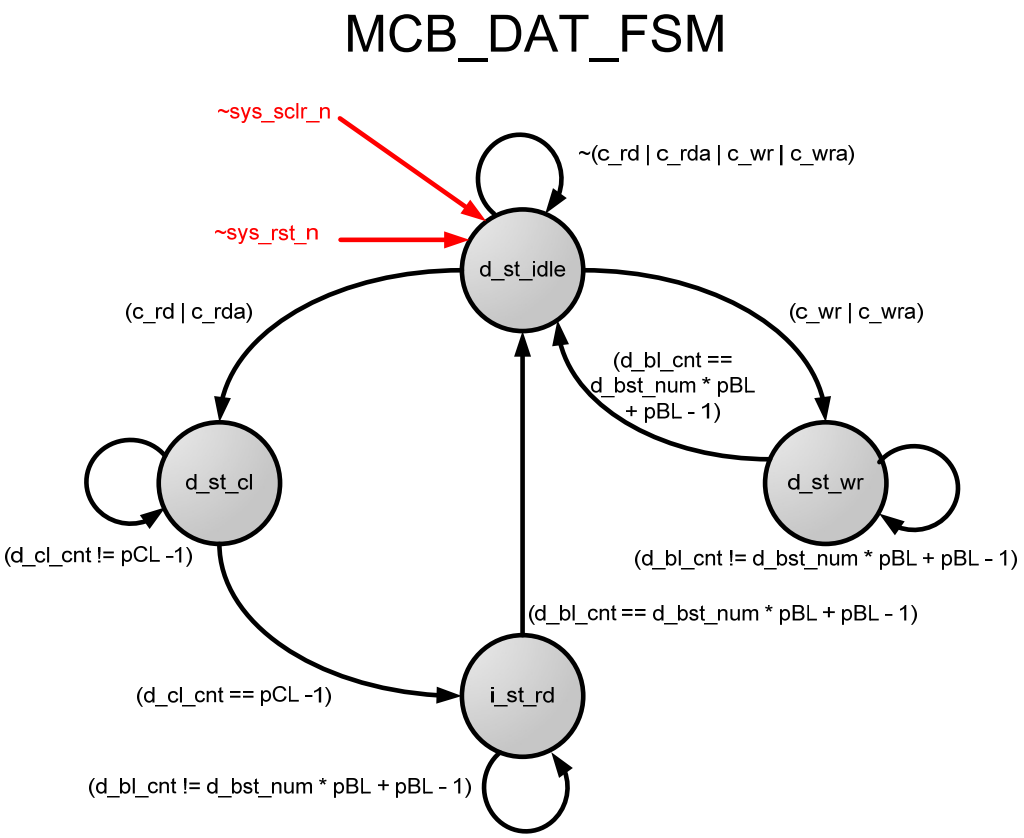
注：上图中，蓝色的条件与器件的交流延时参数、系统工作频率有关，不会映射为电路逻辑。

3.5.3 数据控制电路(MCB_DAT_CTRL)的设计概要

数据控制电路(MCB_DAT_CTRL)，控制数据通路的读写方向，并给出写数据更新请求信号(mcb_wdat_req)、读数据有效信号(mcb_rdat_vld)。它减轻了主控制电路(命令控制电路)的压力，使指令发送和数据发送互不干扰，效率更高。

MCB_DAT_CTRL，包括数据状态机(MCB_DAT_FSM)、Cas Latency 计数器(d_cl_cnt)、SDRAM Burst 数目计数器(d_bst_n_cnt)、SDRAM Burst 长度计数器(d_bl_cnt)。

数据状态机(MCB_DAT_FSM)是其中的主要单元，其状态转换图如下：



3.5.4 刷新控制电路(MCB_REF_CTRL)的设计概要

刷新控制电路(MCB_REF_CTRL)。DRAM 必须不断刷新才能维持数据，MCB_REF_CTRL 就是定时发起刷新请求的控制电路。MCB_REF_CTRL 在初始化时不工作，因为初始化过程中不需要维持 DRAM 存储单元中的数据；初始化完成后，MCB_REF_CTRL 中的刷新计数器(r_ref_i_cnt)开始计数，达到刷新阈值(CtREFi)后停止计数，发起刷新请求，刷新被响应后，计数器清零，继续继续计数，如此循环往复。

刷新阈值(CtREFi)是一个可配置的 verilog 设计参数，理论上应该取 SDRAM 器件手册上的 tREF 除以 SDRAM 行数后折合的时钟周期数。但是由于 MCB_CMD_CTRL 总是先完成当前的系统读写请求之后，再响应 MCB_REF_CTRL 的刷新请求，CtREFi 应该取得大一些以确保万无一失。

3.6 SDRC_Lite 存储控制后端 SDRAM 接口逻辑的设计概要

SDRC_Lite 的 SDRAM 接口逻辑主要分为两个部分：信号通路(MCB_SIG_FF)、数据通路(MCB_DAT_FF)。前者主要把 MCB_CTRL 发出的指令信号翻译为 SDRAM 的命令，必要时同时给出相应的地址；后者依照 MCB_CTRL 的控制信号，对 SDRAM 的输入、输出数据进行锁存，同时也对 SDRAM 的数据 MASK(sdr_dqm)进行适当的控制。

此处对 SDRAM 指令真值表略作回顾：

指令	缩写	cs_n	ras_n	cas_n	we_n	a10	地址
Load Mode Register	LMR	0	0	0	0	x	Op-Code
Auto Refresh	REF	0	0	0	1	x	x
Precharge	PRE	0	0	1	0	0	Bank/x
Precharge All	PREA	0	0	1	0	1	x
Burst Terminate	BT	0	0	1	1	x	x
Write	WR	0	1	0	0	0	Bank/Col
Write with autoprecharge	WRA	0	1	0	0	1	Bank/Col
Read	RD	0	1	0	1	0	Bank/Col
Read with autoprecharge	RDA	0	1	0	1	1	Bank/Col
Activate	ACT	0	1	0	1	1	Bank/Row
No Operation	NOP	0	1	1	1	1	x
Deselect	DSEL	1	x	x	x	x	x

3.7 SDRC_Lite 核心中所涉及的基本参数

信号位宽参数：SDR_A_W(SDRAM 地址位宽)、SDR_B_W(Bank 地址位宽)、SDR_R_W(行地址位宽)、SDR_C_W(列地址位宽)、SDR_D_W(数据位宽)、SDR_M_W(数据 Mask 位宽)；

系统时钟频率参数：MCB_tCK(系统时钟周期(ns))；

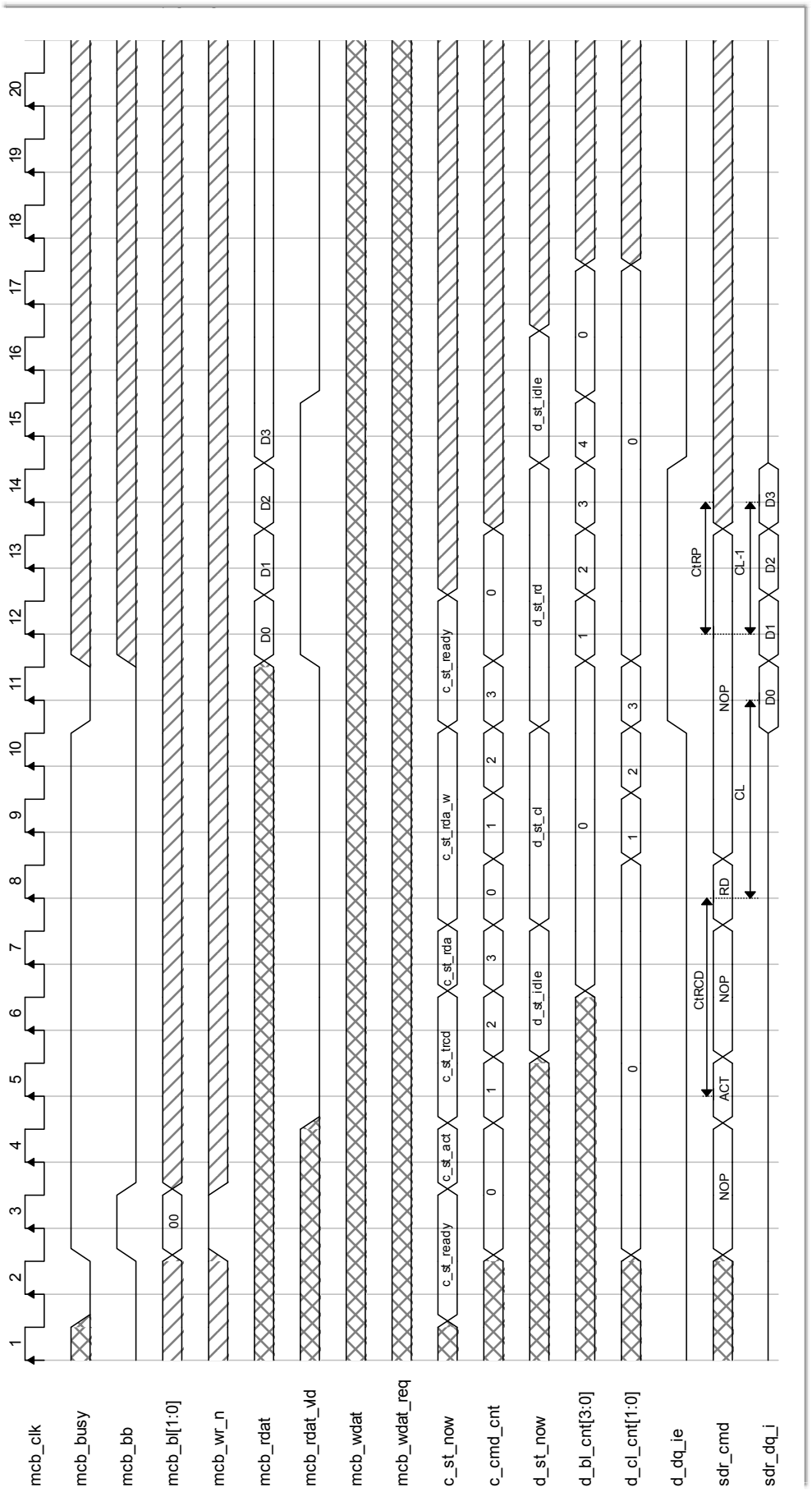
交流延时参数：tRP(预充电延时(ns))、tRFC(刷新延时(ns)，一般等于 tRC)、tRCD(ACT 到列读写的延时(ns))；

初始化控制参数：tINIw(初始化等待时间(ns))、IrefN(初始化刷新次数)。

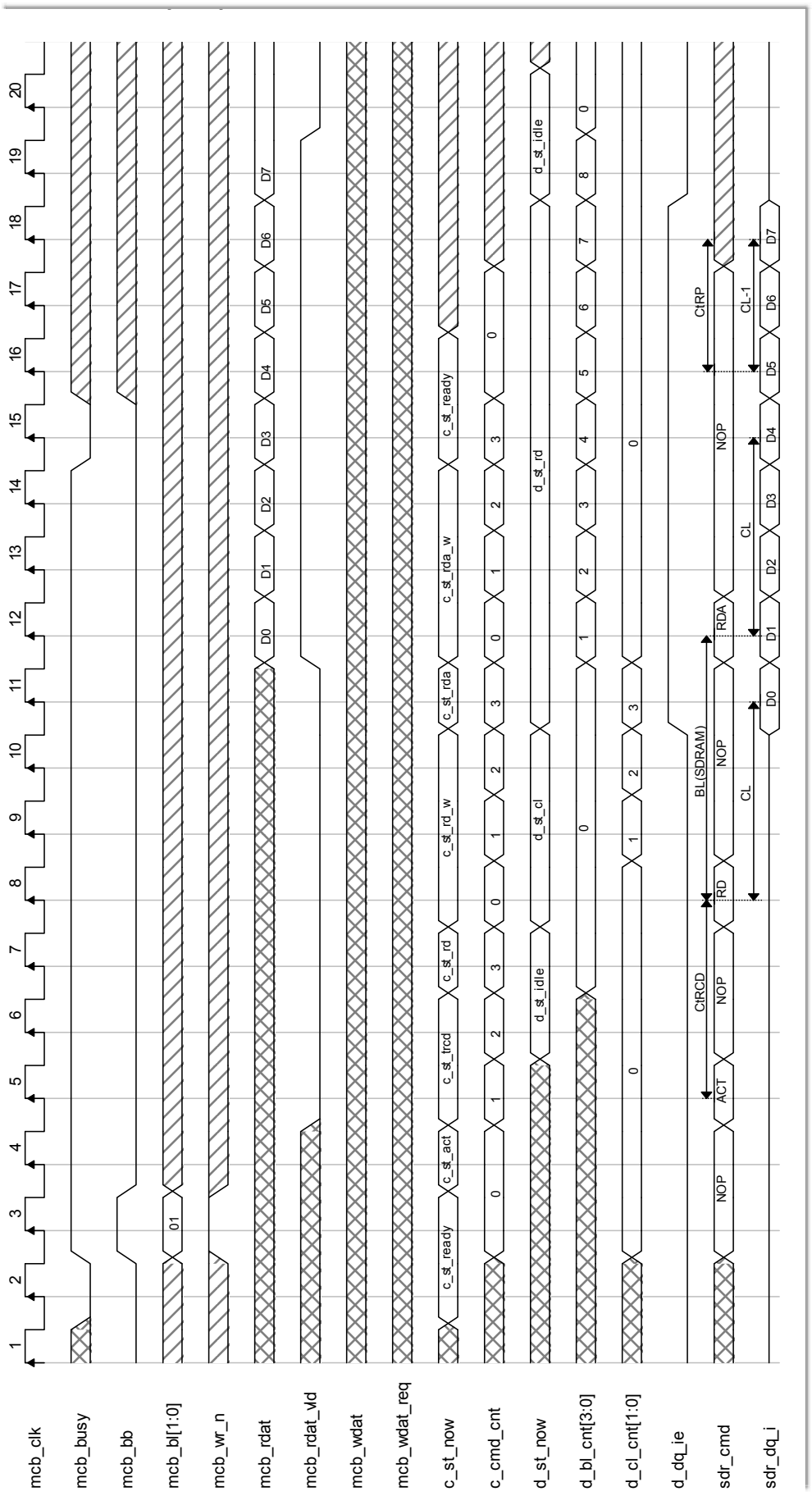
3.8 SDRC_Lite 基本读写操作的时序图

本次设计将严格按照，先出时序图，再出源代码，最后再仿真对比的原则，尽力杜绝“时序靠凑、时序靠仿”的不良设计习惯。时序图使用 TimeGen 3.2 软件全手工绘制；绘制前，根据信号间的逻辑关系，参照 SDRAM 手册中的时序限制，打过草稿；电子版完成后，又反复检查，更正了一些不合理之处。时序图包括 4 张：读取(Burst Length = 4)、读取(Burst Length = 8)、写入(Burst Length = 4)、写入(Burst Length = 8)。下面将依次给出。

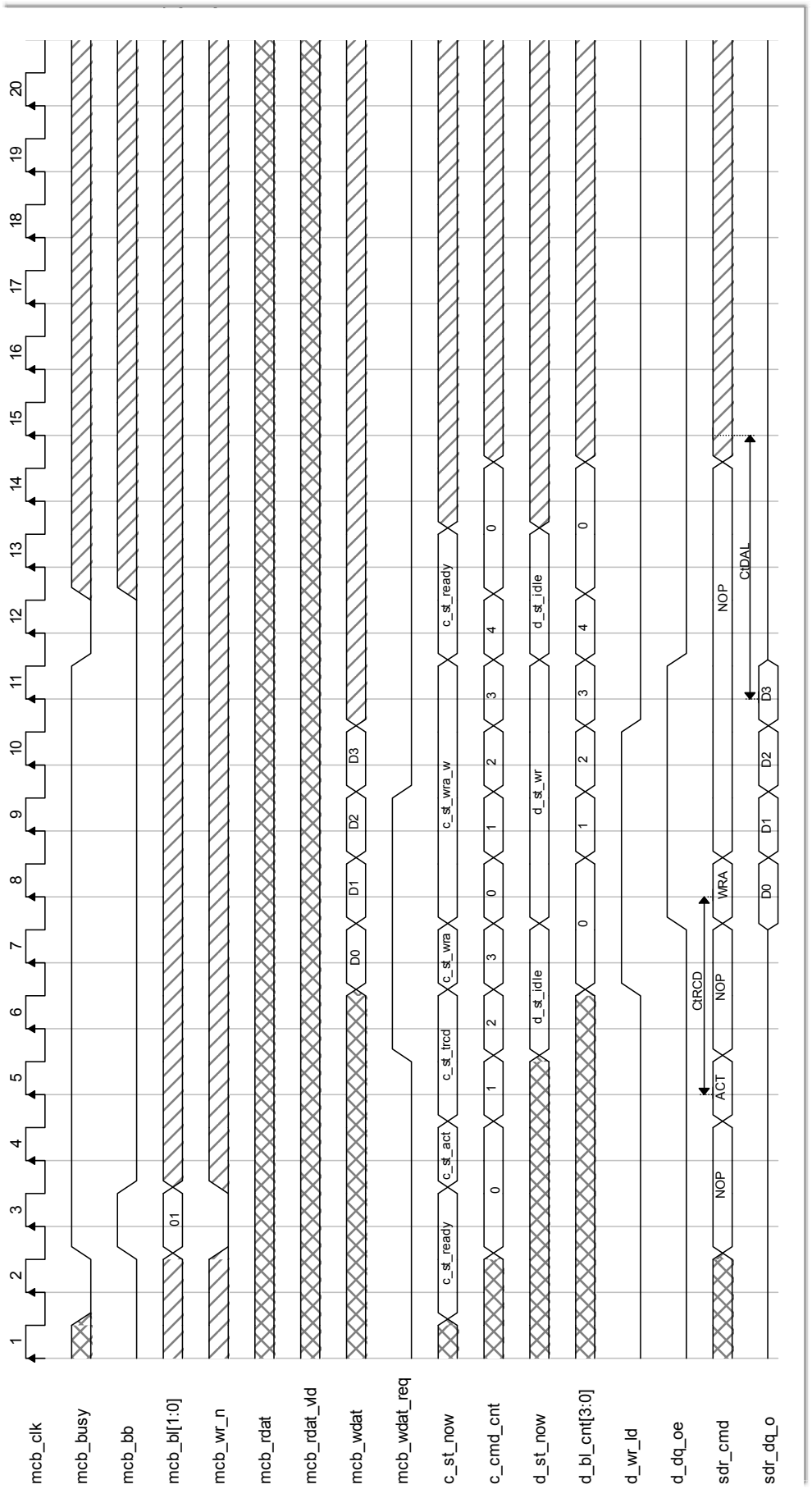
3.8.1 时序图**01**：读时序，系统突发长度为**4**
SDRAM相关的参数如下：**CL=3，BL=4，CtRP=3，CtRCD=3。**



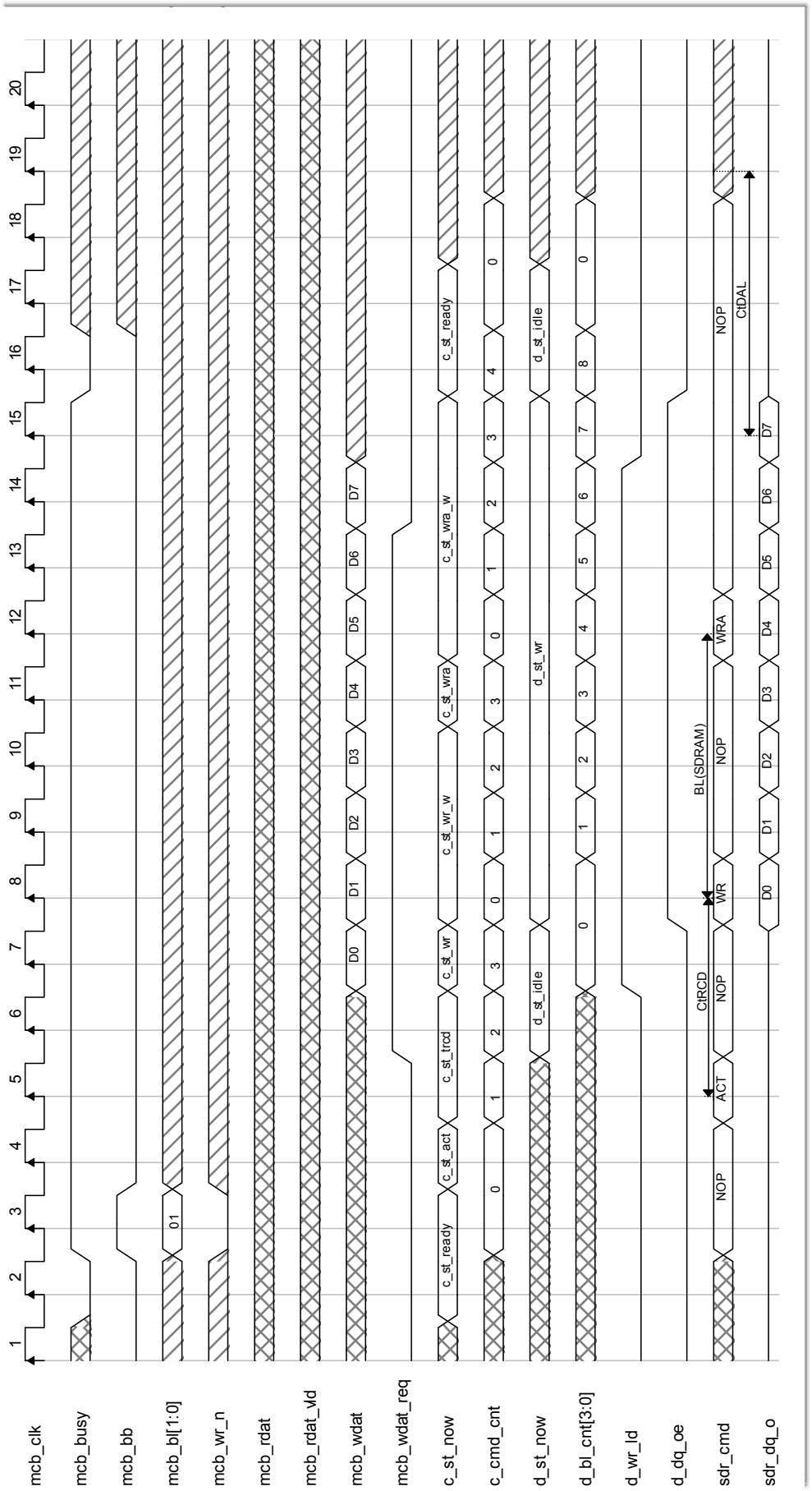
3.8.2 时序图02：读时序，系统突发长度为8
SDRAM相关的参数如下：CL=3，BL=4，CtRP=3，CtRCD=3。



3.8.3 时序图03：写时序，系统突发长度为4
SDRAM相关的参数如下：CL=3，BL=4，CtRP=3，CtRCD=3。



3.8.4 时序图04：写时序，系统突发长度为8
SDRAM相关的参数如下：CL=3，BL=4，CtRP=3，CtRCD=3。



4 本章参考文献

[1] Intel. PC SDRAM Specification, Revision 1.7. November 1999

[2] JEDEC. JEDEC standard No.21-C. 2012

[3] TI. TMS626812 datasheet. 1997

[4] Micron. 64Mb x4, x8, x16 SDR SDRAM. February 2012

[5] ISSI. 1 Meg Bits x 16 Bits x 4 Banks (64-MBIT) SYNCHRONOUS DYNAMIC RAM, Rev. I, Dec. 2011

[6] Elpida. User's manual HOW TO USE SDRAM, rev 8.1. March 2009

[7] Hynix. SDRAM DEVICE OPERATION, rev1.1. Sep.2003

[8] Samsung. CMOS SDRAM Device Operations, rev 0.2. 1999

[9] Windbond. W9864G6JT 1M * 4 BANKS * 16 BITS SDRAM, Revision A01. Dec.2011

[10] Nanya. NT5SV32M8CS NT5SV16M16CS 256Mb Synchronous DRAM, REV 1.4. Dec.2011

[11] Xilinx. xapp134, Synthesizable High Performance SDRAM Controller. 2000

[12] Altera. white paper SDR SDRAM Controller, version 1.1. February 2001

[13] Lattice. RD1010, SDR SDRAM Controller, rev 04.6. April 2011

[14] Zentel. A3V64S40FTP 64Mb Synchronous DRAM Specification, rev 1.0. Aug.2011

[15] Altera. Quartus II Handbook, rev 10.1. Dec. 2010: Ch10 Recommended HDL Coding Styles

[16] Xilinx. UG627 XST user guide, rev 12.4. Dec. 2010: Ch 03:XST HDL Coding Techniques

第四章 SDRC_Lite 内存控制器的仿真、综合与硬件测试

在第三章中讲述了 SDRC_Lite 内存控制器的实现原理与设计概要。本章将讲述它的 Verilog 实现，RTL 仿真，FPGA 综合以及 FPGA 硬件验证。

1 SDRC_Lite 的 Verilog 实现

SDRC_Lite 设计使用 Verilog 语言描述，兼容 IEEE Std 1364-2001 给出的 verilog-2001 标准。用于综合的 RTL 级设计文件中，只涉及 assign、always、if、case 四种语句；状态机的设计风格使用的是 Altera 推荐的风格；用于仿真的行为级测试文件中，使用了 task 语句，使测试更加面向任务；复杂且反复用到的逻辑编写为 function，以增加代码的可读性；所有基本参数和导出参数放在独立文件中，其他设计或仿真文件使用 include 调用它，从而实现“全局参数”。

SDRC_Lite 的设计概要已在第三章明确，Verilog 语言描述完全以设计概要为基础。

2 SDRC_Lite 的 RTL 仿真

本次仿真使用 ModelSIM 6.5 进行，dump 出来的波形使用 Debussy 5.4v9 观察。

SDRC_Lite 的大部分子模块都有单独的 Testbench，经过了单独测试，此处不再赘述。这里，主要讨论 SDRC_Lite 的整体仿真。

2.1 仿真模型与参数设置

SDRC_Lite 的整体仿真中，用到了 Micron 公司提供的 SDRAM 模型，以便模拟与 SDRAM 器件的交互：比如 Mode Register 设置、读出先前写入的数据等等。

该 Micron SDRAM 模型，对应型号为 mt48lc4m16a2-7E 的 SDRAM 芯片。它是一款 64Mbit 的 SDRAM，其中与 SDRC_Lite 设计相关的参数值如下表：

SDRC_Lite 参数名	含义	mt48lc4m16a2-7E 的参数值
SDR_A_W	SDRAM 地址位宽	12
SDR_B_W	SDRAM Bank 地址位宽	2
SDR_R_W	SDRAM 行地址位宽	12
SDR_C_W	SDRAM 列地址位宽	8
SDR_D_W	SDRAM 数据位宽	16
SDR_M_W	SDRAM 数据 Mask 位宽	2
tRP	预充电延时(ns)	15
tRFC	刷新延时(ns)	66
tRCD	ACT 到列读写的延时(ns)	15

仿真的最小时间单位 1ns，为精度为 100ps。系统时钟定为 100MHz，即周期为 10ns，这

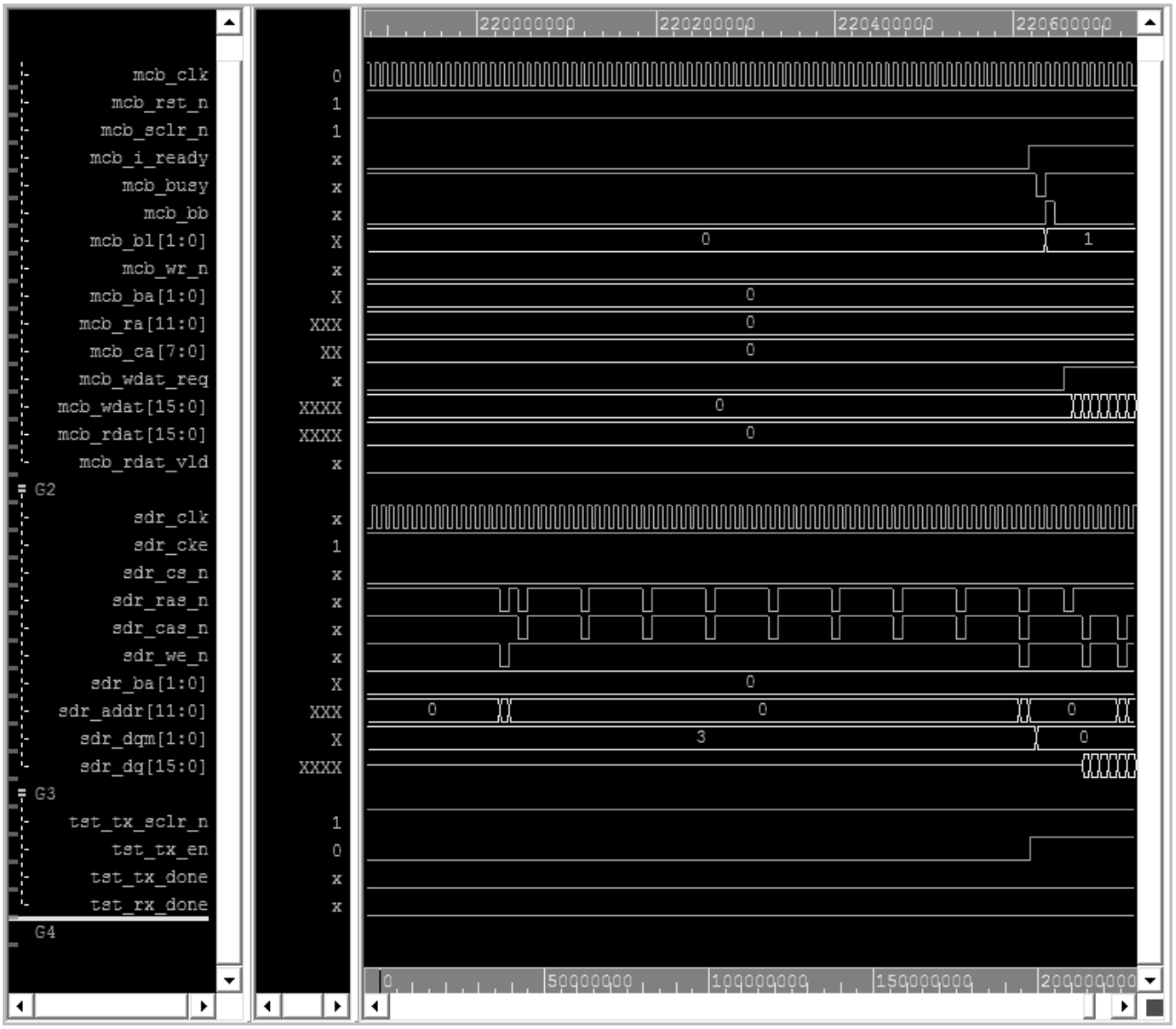
个数值与实际应用的频率接近，也便于仿真。

为了保证指令和数据被 SDRAM 有效接收，一般要求 SDRAM 时钟相对于系统时钟有 3ns 的滞后。在 Testbench 中，此延时用延时赋值语句实现；FPGA 验证中，用 PLL 核实现。

2.2 包含 Micron Model 的 Testbench 与仿真波形

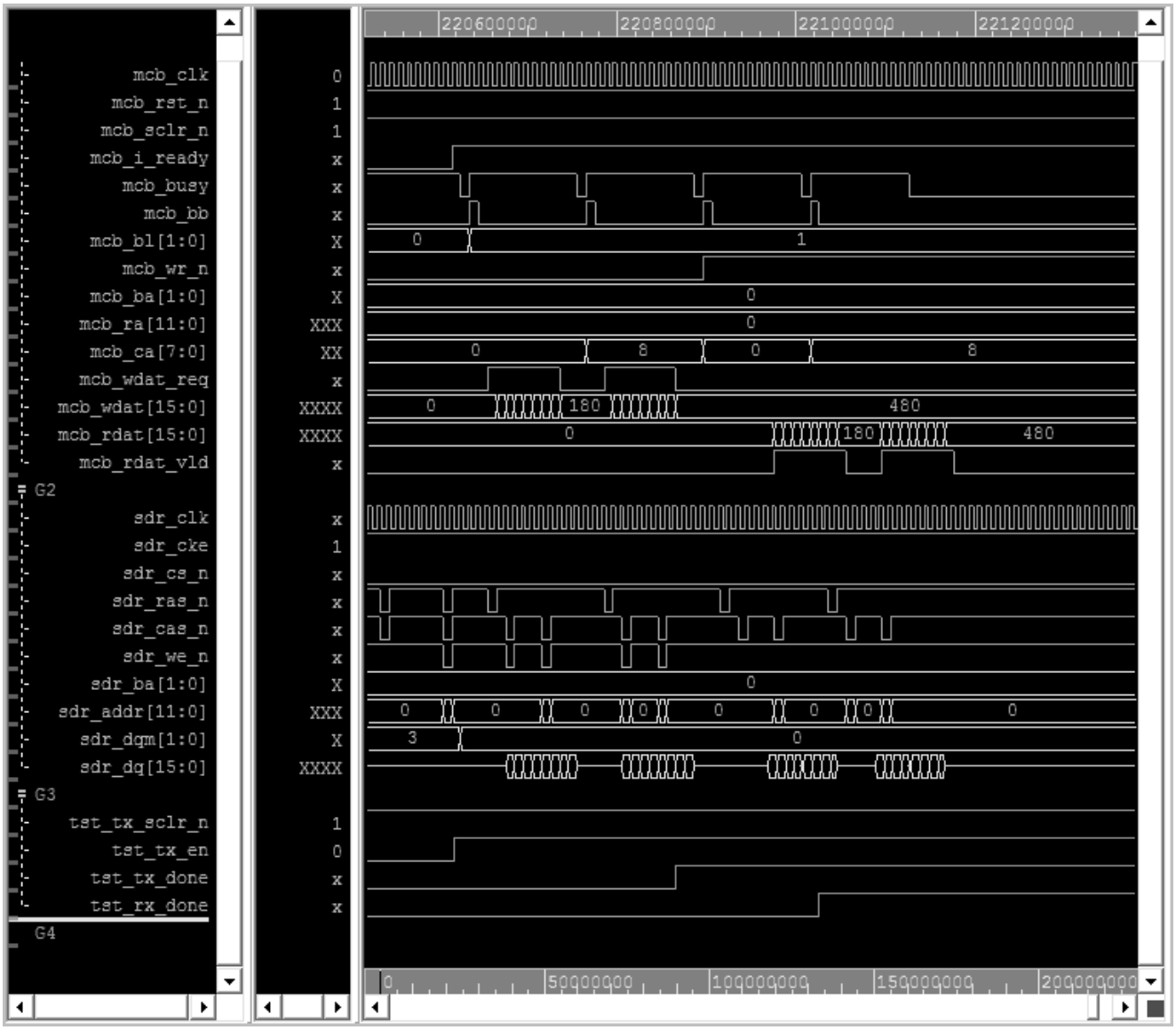
Testbench 包括读写控制单元(SDRC_MCB_TOP_TST_TRX16)、SDRC_Lite(MCB_TOP)、Micron Model 三个部分。读写控制单元，使用 readmemb 语句从位流文件中载入位流，写入通过 SDRC_Lite 写入 SDRAM，再回读出来。

整个过程中，SDRC_Lite 自动完成 SDRAM 初始化的波形如下：



观察图中 sdr_cs_n、sdr_ras_n、sdr_cas_n、sdr_we_n 四个信号，可知 SDRC_Lite 依次向 SDRAM 发送了如下指令：PREA、8 次 REF、LMR。之后 mcb_i_ready 变为 1，表示初始化完成。指令的间隔也没有问题，初始化完成前 dqm 为 1，整个过程符合设计要求。

整个过程中，系统端通过 SDRC_Lite 读写 SDRAM 的波形如下：



如上图，每当 SDRC_Lite 的 mcb_busy 为 0 时，读写控制单元通过 mcb_bb 发起突发传送开始信号。图中 mcb_bl 为 2'b01 表示突发传送长度为 8，读写操作的指令依次为：ACT、WR、WRA、ACT、WR、WRA、ACT、RD、RDA、ACT、RD、RDA，可见从系统端看来：发生了两次突发传送长度为 8 的读和两次突发传送长度为 8 写。指令的间隔也没有问题，指令序列输出符合指令状态机的设计意图。可以观察到写入的数据与读出的数据一致，限于篇幅此处不再贴放大的波形。

2.3 其他仿真测试

在另一个 SDRC_Lite 的 testbench 中，对长为 4 的读、长为 8 的读、长为 4 的写、长为 8 的写这四种请求进行了排列组合，观察了状态机的状态跳转。同时，还观察了刷新操作对系统端连续读写请求的影响。所有行为与设计要求符合，限于篇幅此处不再贴该 Testbench 的波形。

3 SDRC_Lite 的 FPGA 综合

本次 FPGA 综合使用的是 Quartus 10.1sp1，目标器件是 Cyclone II ep2c8pq208c8。

综合时开启的优化选项包括：寄存器延时平衡(register retiming)、寄存器复制(register duplication)、时序驱动的综合(timing-driven synthesis)、向提高速度方向优化等等。

综合时，任然是用仿真时用的面向 mt48lc4m16a2-7E 的参数。

综合后 SDRC_Lite 使用了 221 个 LE(组合逻辑 182 个，寄存器 152 个)。以 mcb_clk 为时钟，进行简单的静态时序分析，得出此设计的最高工作频率为 164.12MHz，这个频率已可以满足一般 SDRAM 应用的需要。

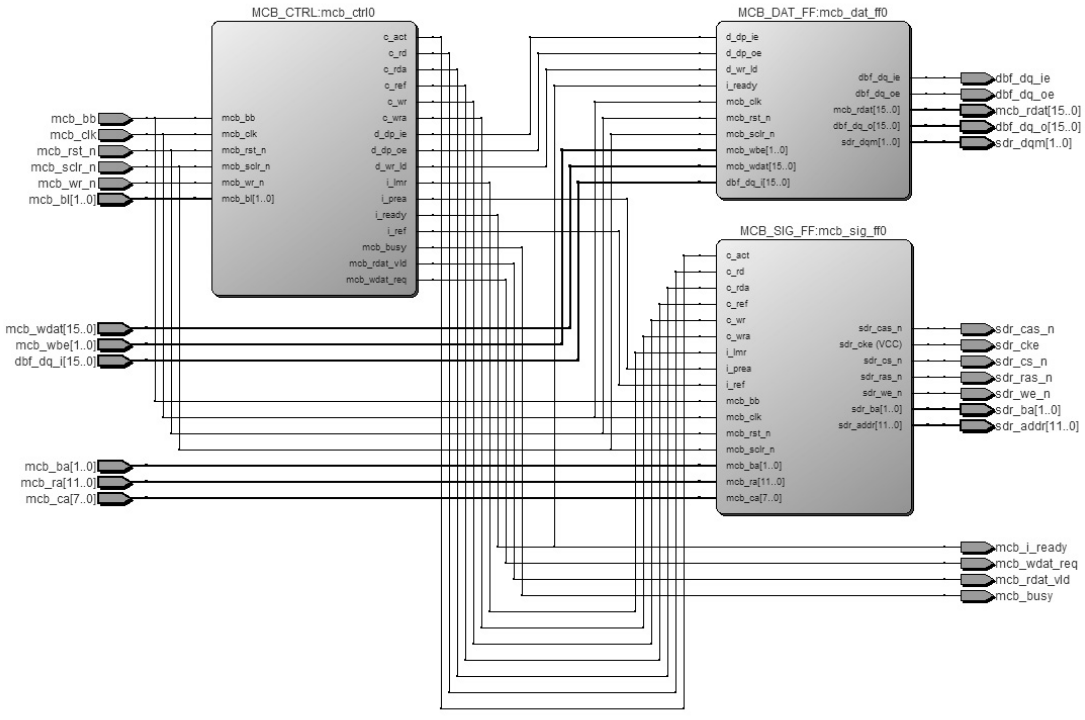
作为对比，在相同的参数设置、目标器件、综合条件下，对 OpenCore 上的 HSSDRC 内存控制 IP 进行综合。发现 HSSDRC 占用了 710 个 LE(组合逻辑 602 个，寄存器 399 个)。对 HSSDRC 做了简单的时序分析，发现同等条件下它的最高工作只有 95.72MHz，不能很好地满足实际应用的频率需求。

当然这样的不同，是由于 SDRC_Lite 与 HSSDRC 的不同设计原理造成的。SDRC_Lite 使用了 Close Page Policy，HSSDRC 使用了 Open Page Policy；SDRC_Lite 不进行 SDRAM 指令流水线化，HSSDRC 实现了 SDRAM 指令流水线化。

由于 SDRC_Lite 已足以满足 H.264 编码所需的带宽要求，故采用它而不是 HSSDRC，可以降低芯片的面积和功耗，提升芯片的最高工作频率。

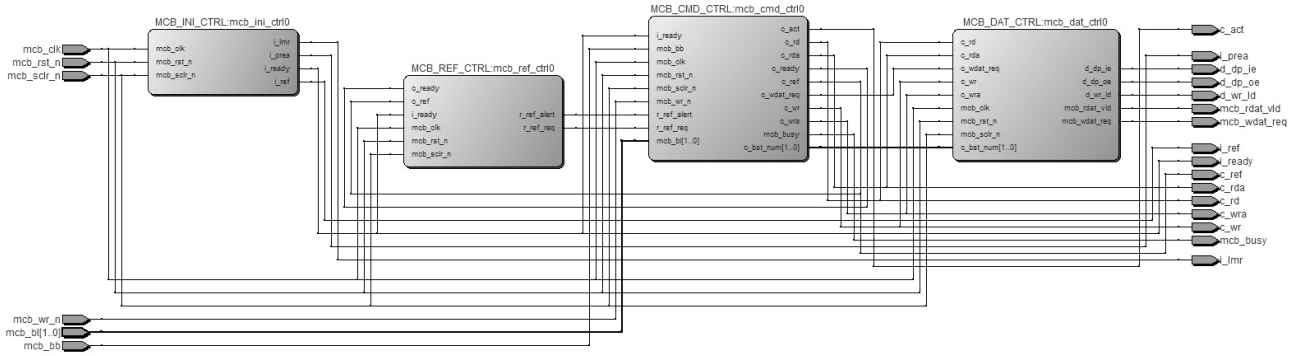
下面列出一些综合后得到的图像：

SDRC_Lite 顶层模块：MCB_TOP　(包含 MCB_CTRL、MCB_SIG_FF、MCB_DAT_FF)

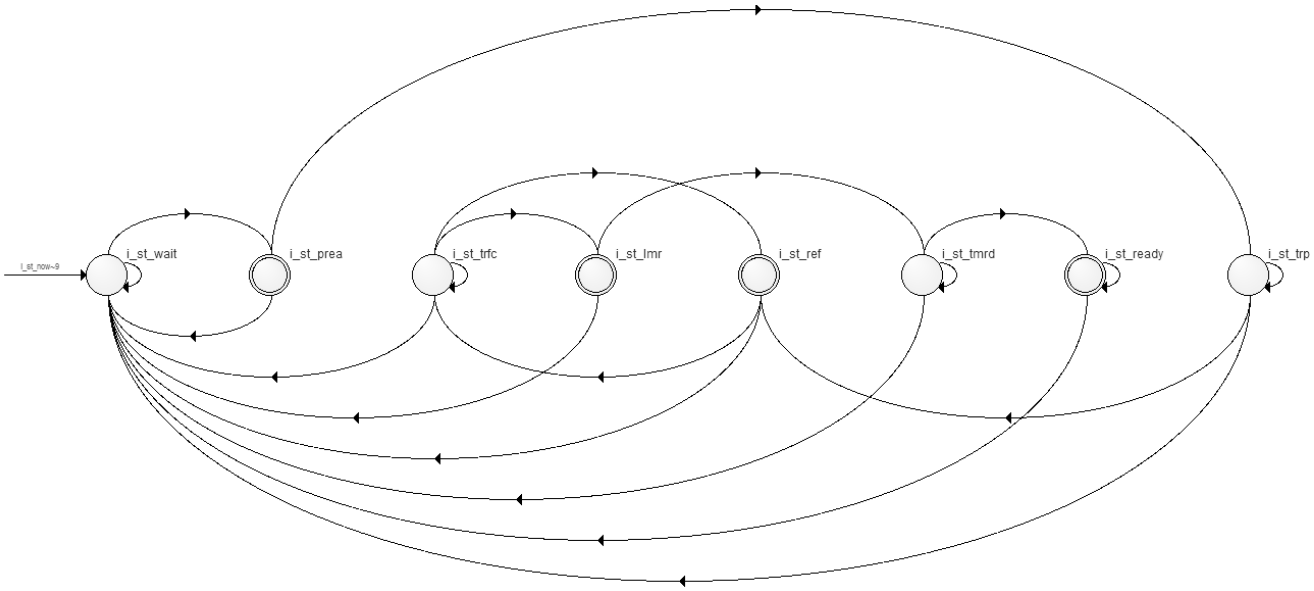


SDRC_Lite 控制模块：MCB_CTRL

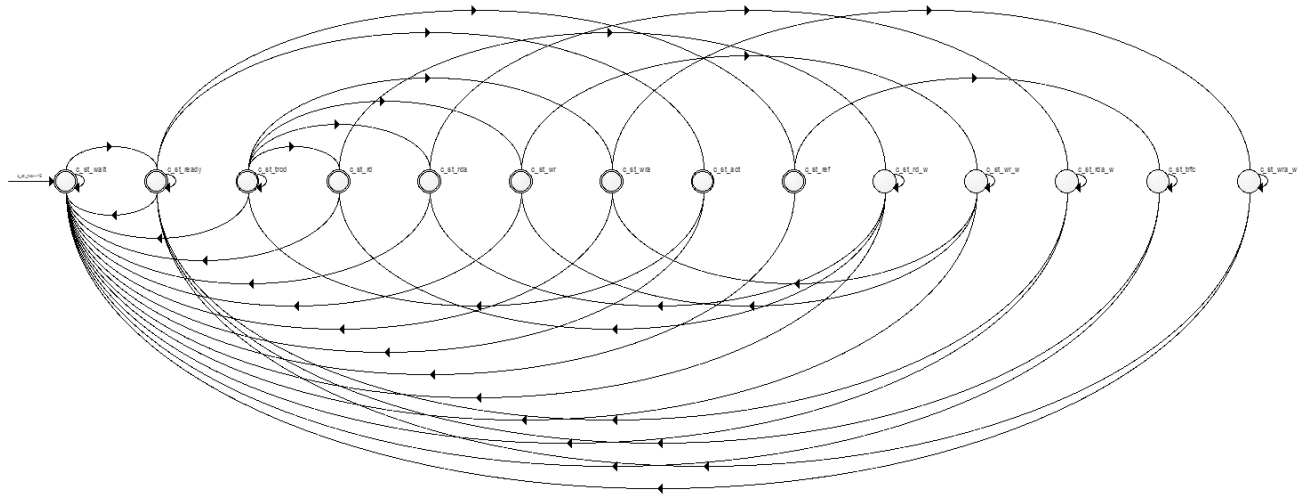
(包含 MCB_INI_CTRL、MCB_REF_CTRL、MCB_CMD_CTRL、MCB_DAT_CTRL)



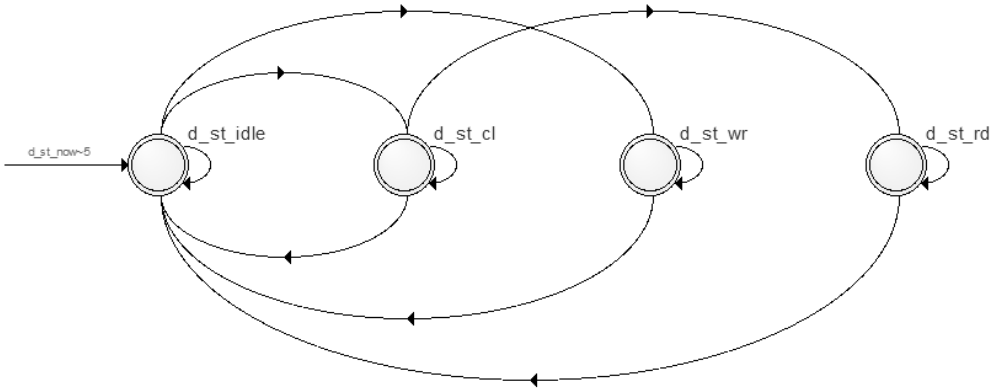
初始化状态机(MCB_INI_FSM)



命令状态机(MCB_CMD_FSM)



数据状态机(MCB_DAT_FSM)



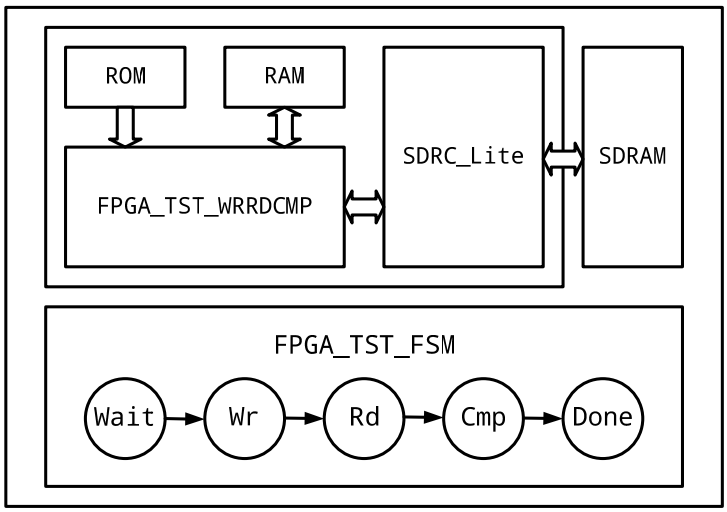
4 SDRC_Lite 的 FPGA 硬件验证

SDRC_LIte 的 FPGA 验证主要使用了 Altera Cyclone II ep2c8pq208c8 FPGA，以及 Hynix HY57V641620FTP-7 SDRAM。

FPGA 验证时的顶层设计文件主要包括：测试控制单元(FPGA_TST_WRRDCMP)、内存控制器 SDRC_Lite(MCB_TOP)、PLL 模块(MC_PLL)、IO 三态缓冲、片上 RAM、片上 ROM。

PLL 模块(MC_PLL)。用于把开发板上 20MHz 的时钟倍频到 100MHz 作为系统时钟，并产生相位相差负 3ns 的另一个 100MHz 时钟供 SDRAM 使用。系统时钟超前于 SDRAM 时钟，可以保证发送给 SDRAM 的指令和数据被有效地接收。

整个 FPGA 验证的测试思路如下图：



ROM。用 FPGA 的 Block RAM 实现，存储一组随机的数据，用于写入 SDRAM。

RAM。用 FPGA 的 Block RAM 实现，存储从 SDRAM 中读出的数据。

测试控制单元(FPGA_TST_WRRDCMP)。验证过程在 FPGA_TST_WRRDCMP 的控制下完成。上电后，FPGA_TST_WRRDCMP 首先等待 SDRAM 初始化完成；之后，从 ROM 读出

数据通过 SDRC_Lite 控制器写入 SDRAM；写入完成后，从 SDRAM 中读出数据写入 RAM 中；最后，对比 RAM 中的数据和 ROM 中的数据，如果完全吻合，发起操作成功信号。

经过硬件验证，SDRAM 读写正常，说明 SDRC_Lite 的设计合理可靠。同时，也发现 PLL 的作用巨大，如果 SDR_clk 和 MCB_clk 的相对相位不适当，读写无法正确完成。

5 下一步的验证

前面讲述的硬件验证方案比较死板，不易于通过软件控制，下一章中我们将讲述 SDRC_Lite 的 SOPC 集成以及进一步验证。

6 本章参考文献

[1] Micron. 64Mb x4, x8, x16 SDR SDRAM. February 2012

[2] Pong P. Chu. FPGA Prototyping Using Verilog Examples, Wiley-Interscience. June 2008: 全文参考

第五章 SDRC_Lite 内存控制器的 SOPC 集成与测试

本章主要讨论 SDRC_Lite 的 SOPC 集成与测试。包括：Avalon-MM Wrapper 的设计、基于 Avalon-MM Master BFM 的仿真测试、基于 Jtag to Avalon Master Bridge 的 FPGA 硬件测试。

1 Avalon-MM Wrapper 的设计

为了将 SDRC_Lite 内存控制后端集成到 SOPC 系统中，我设计了一个 Avalon-MM Wrapper 把 SDRC_Lite 的接口协议转化为 Avalon-MM 接口协议。Avalon-MM Wrapper 与 SDRC_Lite 一起构成了一个 SDRC_Lite Avalon IP，具有可配置的参数，可以方便地在不同的 SOPC 设计中调用。

1.1 SOPC 与 Avalon 总线简介

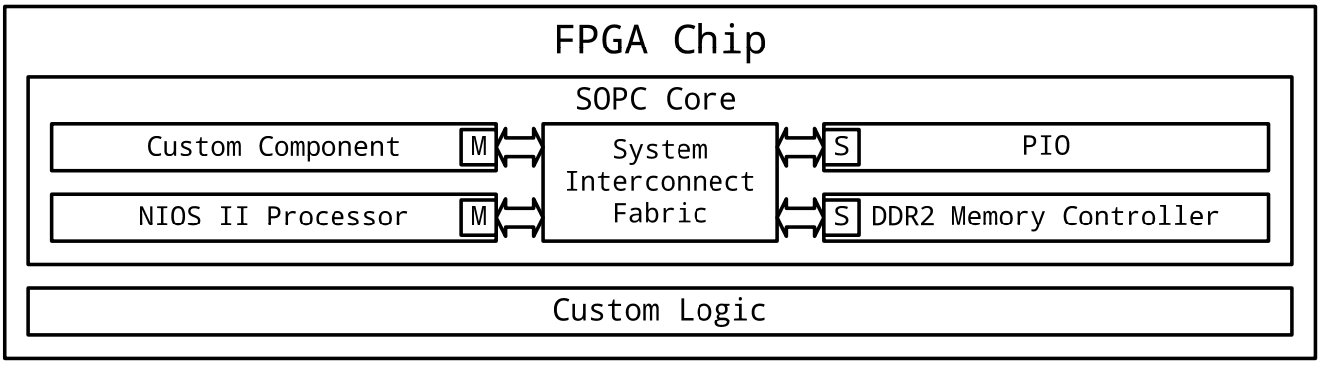
SOPC 是一个使用不同功能单元快速搭建 SOC 系统的半自动化工具，用户可以使用 Altera 提供的 IP 核，也可以使用自己设计的 Custom IP。SOPC 中的互连使用 Avalon 总线协议，如果用户设计的接口不符合 Avalon 总线协议，则需要添加 Wrapper 以便集成。

Avalon 总线协议主要包括 Avalon-ST 协议和 Avalon-MM 协议。Avalon-ST 协议主要完成单向数据包的传输；Avalon-MM 协议主要完成基于地址映射的读写。两者各有 Master 接口、Slave 接口之分。

SOPC 工具最强大的一个方面是可以自动生成各个功能单元之间复杂的逻辑。这些逻辑包括：地址译码、数据选通、流水线化的读传输、等待状态发起、中断信号控制等等。较有特色的一点是，SOPC 的互联中使用了 Slave 端仲裁，每个连接了多个 Master 的 Slave 前都有一个仲裁器，所以同一时间可以有多个 Master 操作多个不同的 Slave，这是传统的 Master 端仲裁做不到的。

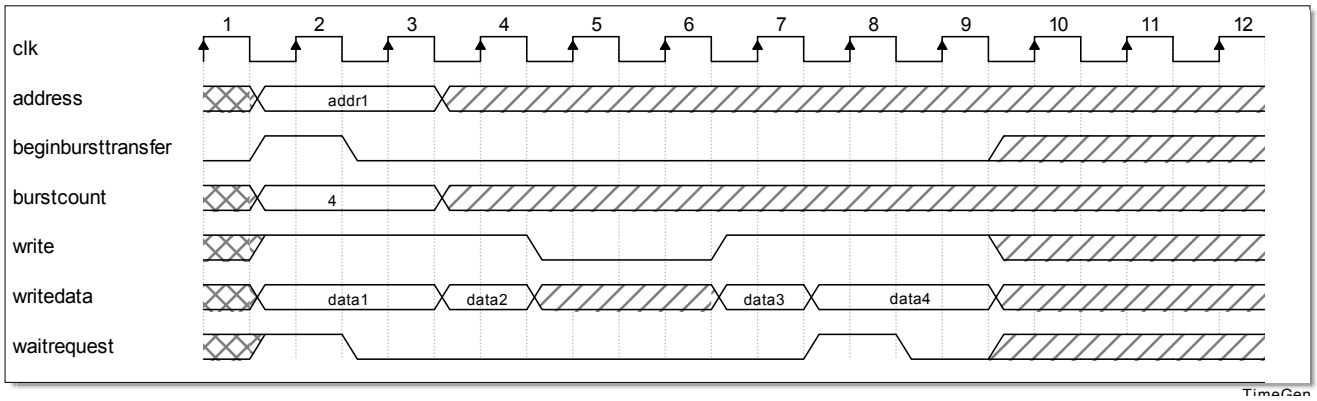
需要指出的是，SOPC 的高度自动化也带来了一定不预知性，经验不足的用户不一定能很好地控制其中复杂系统中的数据传送。

SOPC 工具生成的系统，可以作为一个软核被更高层级的设计调用，下图给出了一个 FPGA 设计示意图：



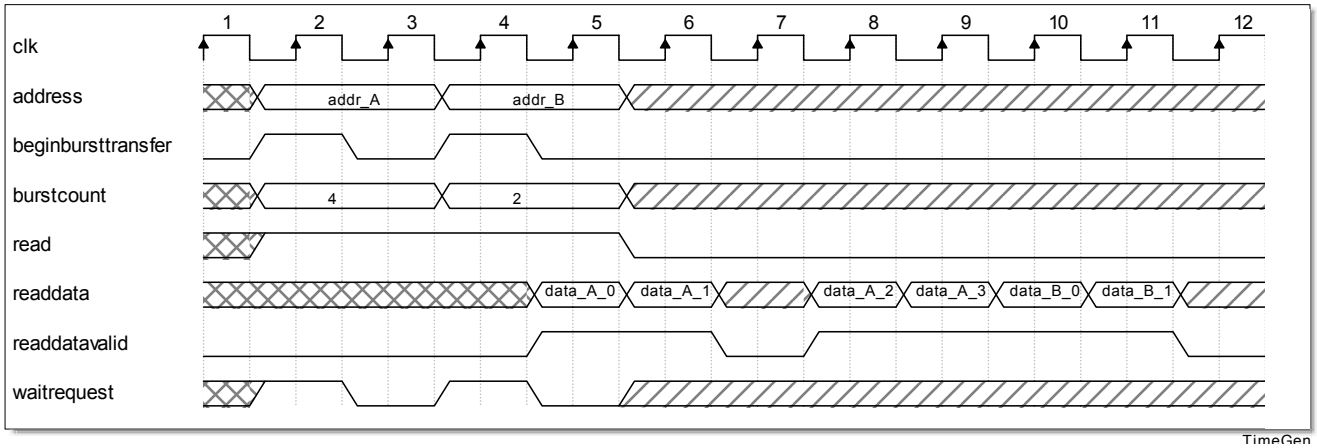
1.2 Avalon-MM 协议简介

支持 Burst 传输的 Avalon-MM Slave 接口的典型写时序图如下：



系统端发起为时 1 个周期的 beginbursttransfer 信号，同时，write 信号为有效，并给出地址(address)和突发传送长度(burstcount)。收到 beginbursttransfer 后，waitrequset 立刻变为有效。从 waitrequset 变为无效后的下一周期 address、burstcount 被忽略。如果上一周期 waitrequset 无效，则 Master 在下一周期更新写数据，否则 Master 会对 Slave 进行等待；burst 完成前 write 为 0，表示 Master 要求 Slave 进行等待。

支持 Burst 传输的 Avalon-MM Slave 接口的典型读时序图如下：



系统端发起为时 1 个周期的 beginbursttransfer 信号，同时，read 信号为有效，并给出地址(address)和突发传送长度(burstcount)并保持到 waitrequset 变为无效后的下一周期为止。收到 beginbursttransfer 后，waitrequset 立刻变为有效。waitrequset 变为无效后的下一周期，既可以发起下一次突发读操作。当 readdatavalid 为 1，表示当前的 readdata 来自突发读操作并且应当被 master 接收。

“maxiumPendingReadTransaction”属性描述了一个 Slave 接口能在有多少个读请求的数据尚未完全传回 Master 的情况下接受下一个读请求。虽然可以用 waitrequset 拖延 Master 的读操作，但是在 SOPC 集成时正确设置此参数可以提高 SOPC 系统互连的效率。

1.3 Avalon Wrapper 的设计

本次设计中的 Avalon-MM Wrapper 主要实现两方面的功能。基础功能，提供 SDRC_Lite 系统端接口到 Avalon-MM Slave 接口的转化；高级功能，通过内部逻辑掩盖 SDRAM 的地址特性，提供更加简单易用的存储访问接口。

其信号与简要说明如下表：

Signal Name	Dir.	Description
(Interface to SOPC interconnect fabric)		
csi_clockreset_clk	Input	sopc clock input
csi_clockreset_reset_n	Input	sopc reset input
avs_s1_address[AVL_A_W-1:0]	Input	avalon-MM address
avs_s1_read	Input	avalon-MM read
avs_s1_write	Input	avalon-MM write
avs_s1_beginbursttransfer	Input	avalon-MM burst begin
avs_s1_waitrequest	Output	avalon-MM wait request to master
avs_s1_burstcount[4:0]	Input	avalon-MM burst length
avs_s1_readdatavalid	Output	avalon-MM read data valid
avs_s1_readdata[AVL_D_W-1:0]	Output	avalon-MM read data
avs_s1_byteenable[AVL_BE_W-1:0]	Input	avalon-MM write mask (per byte)
avs_s1_writedata[AVL_D_W-1:0]	Input	avalon-MM write data
(Interface to SDRC_Lite Memory Controller Back-end)		
mcb_clk	Input	MCB clock
mcb_rst_n	Input	MCB asynchronous reset
mcb_sclr_n	Input	MCB synchronous reset
mcb_bb	Input	MCB burst begin
mcb_rw_n	Input	MCB data direction: 0-write, 1-read
mcb_bl[1:0]	Input	MCB burst length: 00-4, 01-8, 10-12, 11-16
mcb_ba[MCB_B_W-1:0]	Input	MCB bank address
mcb_ra[MCB_R_W-1:0]	Input	MCB row address
mcb_ca[MCB_C_W-1:0]	Input	MCB column address
mcb_busy	Output	MCB sdr sdram controller busy
mcb_rdat_vld	Output	MCB read data valid
mcb_wdat_req	Output	MCB data to write request
mcb_wdat[MCB_D_W-1:0]	Input	MCB write data

mcb_rdat[MCB_D_W-1:0]	Output	MCB read data
mcb_wbe[MCB_BE_W-1:0]	Output	MCB write data byte enable
mcb_i_ready	Output	MCB initialization done

1.3.1 任意长度的 Burst、不与 Burst 边界对齐的 Burst

此次设计的 Avalon Wrapper，支持 1、2、3、4、5、6、7、8 共八种 Burst 长度，Burst 传输的起始地址不一定要在 Burst 的边界上。

回顾 SDRC_Lite 的设计，SDRC_Lite 要求请求中列地址的最低两位为 0，并且支持 4、8、12、16 四种 Burst 长度，而不支持“长为 3”或者“起始列地址为 11”这类的 Burst 请求。

任意长度的 Burst、不与 Burst 边界对齐的 Burst，通过“内部读写长度扩展”以及“Mask 操作”协同实现。每次接收到请求后就根据 Burst 长度和起始位置，计算出对应的 SDRC_Lite 突发传送长度和相应的 Mask。

对于写操作，有两个交替更新的寄存器，用于记录当前 avalon 操作和下一 avalon 操作对应的 SDRC_Lite 突发传送长度和相应的 Mask。

对于读操作，在把数据输出给 avalon master 时还需要用 Mask 来计算 readdatavalid 的值，以保证返回的数据符合原先的 burst 长度，所以使用了一个长度为 4 的同步 FIFO 记录读操作的 SDRC_Lite 突发传送长度和相应的 Mask。

1.3.2 跨 SDRAM 行的 Burst

SDRC_Lite 设计时，假设单次读写请求不会从 SDRAM 的一个行跨越到另一个行。如果出现这样的情况，前一行的开头会被覆盖而后一行不会被写入。

此次设计的 Avalon Wrapper，会自动检测单次 Burst 是否跨越了 SDRAM 的行边界。如果跨越了行边界，则自动将它拆分为 2 个 SDRC_Lite 突发读写请求。

1.3.3 Avalon Wrapper 的地址映射

此次设计的 Avalon Wrapper，并未采取复杂的地址映射，直接把 bank address、row address、column address 拼接为 avalon address。该 avalon address 是以 SDRAM 的 word 而不是字节为对象的，在连接其他 Avalon IP 时需要注意。

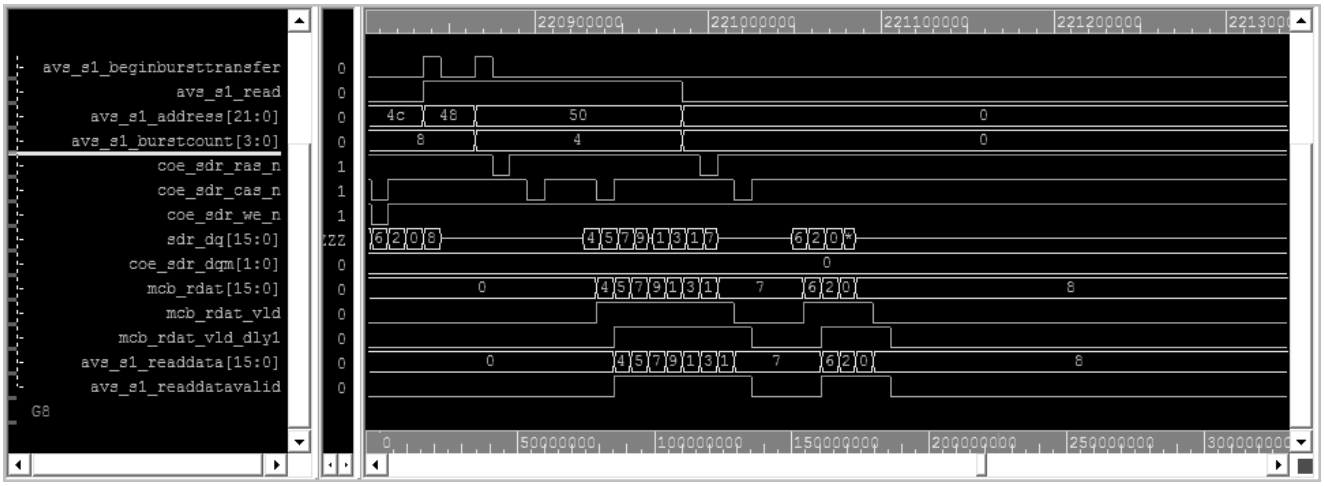
通过前面讲述实现任意长度的 Burst、不与 Burst 边界对齐的 Burst、跨 SDRAM 行的 Burst，Avalon Wrapper 掩盖了 SDRAM 的地址访问特性，把它抽象为一个简单的存储整列，方便了系统集成。

当然，如果使用者想提高 SDRAM 的读写效率，任然应该使用长为 4 或 8、与 Burst 边界对齐、不跨越 SDRAM 行的 Burst，以减少不必要的(被 mask 掉的)SDRAM 读写。

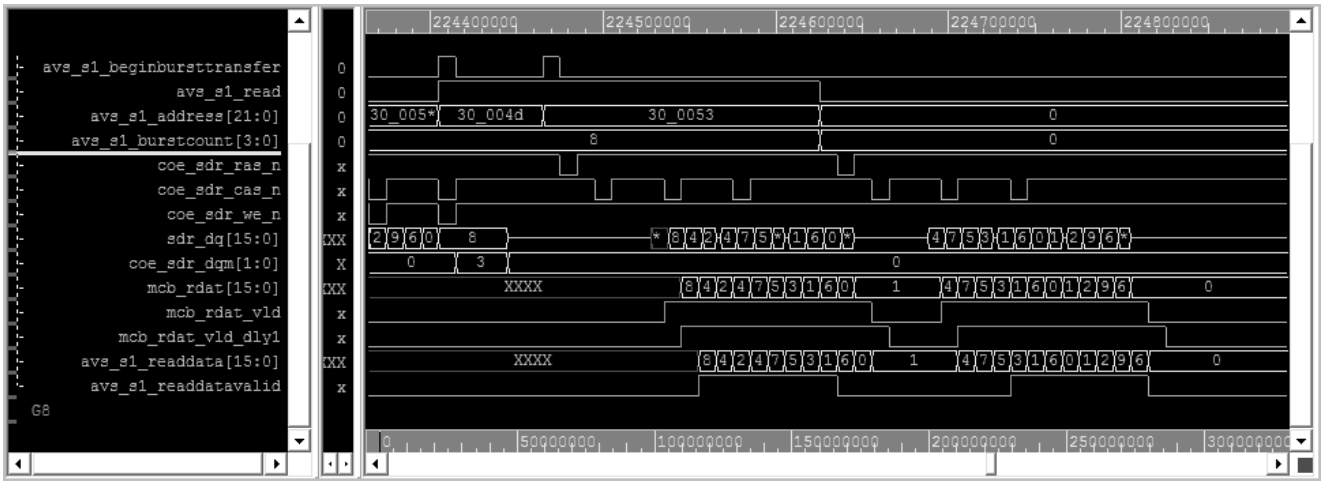
1.4 Avalon Wrapper、SDRC_Lite、Micron SDRAM Model 的协同仿真

在完成 RTL 设计后，对 Avalon Wrapper、SDRC_Lite、SDRAM Micron Model 进行了协同仿真，以验证其功能。这里列出不同情况下的 Burst 读操作仿真图，做一个对比。

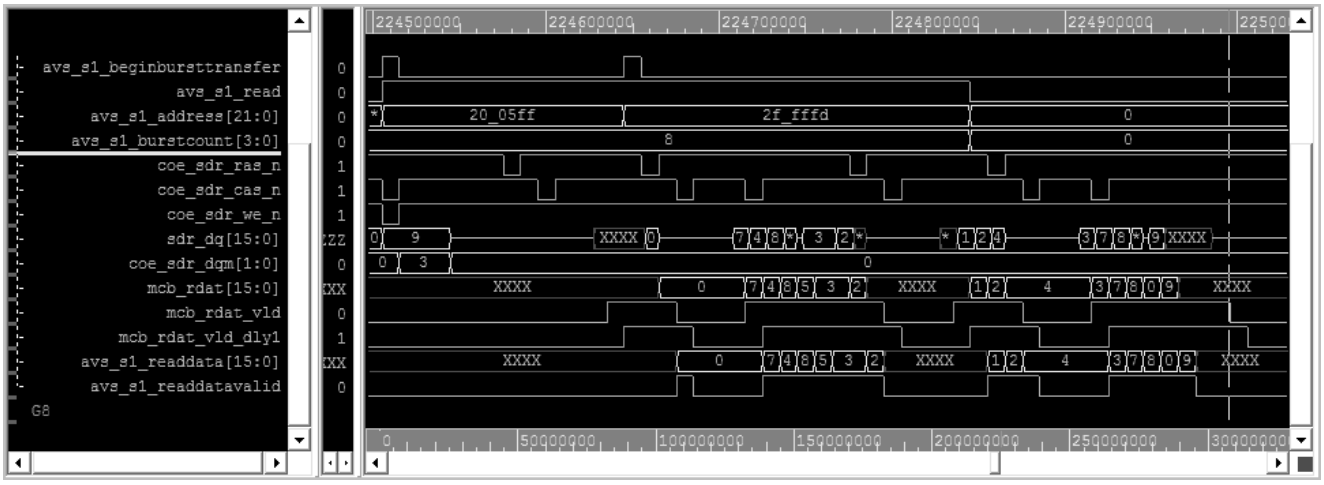
下图展示了长分别为 4 和 8、不跨越 SDRAM 行、与 Burst 边界对齐的 Burst 读。



下图展示了长度为 8、不跨越 SDRAM 行、不与 Burst 边界对齐的 Burst 读。



下图展示了长度为 8、跨 SDRAM 行、不与 Burst 边界对齐的 Burst 读。



其他方面的仿真还有很多，包括了写读数据对比等等此处从略。

从仿真中可以看出，本设计实现了通过 Avalon 协议访问 SDRAM 的功能。对于访问地址的也没有特殊要求，而使用符合 SDRAM 特性的地址可以提高访问效率。整体而言，仿真结果让人满意。

1.5 Quartus 综合

本次工作中，将 Avalon Wrapper 与 SDRC_Lite 做为一个整体(MCB_AVL_IP_TOP)，做了一次 Quartus II 综合。综合条件与之前综合 SDRC_Lite 时类似，任然面向 Altera Cyclone II ep2c8pq208c8。

MCB_AVL_IP_TOP 综合后，共占用了 430 个 LE(其中 Avalon Wrapper 占用了 227 个 LE，SDRC_Lite 占用了 203 个 LE)。经过简单的静态时序分析，发现其最高工作频率为 163.93MHz，与单独实现 SDRC_Lite 时的 164.12MHz 基本持平，Avalon Wrapper 没有拖低最高工作频率。

作为对比，我从 SOPC 中提取出了其自动生成的 SDRAM 控制器，以类似的条件对其做了单独的综合。该免费 IP 在内部使用了 Transaction FIFO，综合后共占用了 364 个 LE，最高工作频率只有 105.61MHz。

相比之下，本次设计的 SDRC_Lite Avalon IP(MCB_AVL_IP_TOP)比 Altera 的免费 IP 面积稍大，而最高工作频率却要高很多。

1.6 从 RTL 代码到 SOPC Custom IP

SOPC 有一个 Custom IP 生成工具，可以自动分析 Verilog 代码，提取接口与参数信息，之后用户可以对自动分析的结果作手动修改。“maxiumPendingReadTransaction”属性，就是在此时设置的。

详细的软件细节可以参考 Altera 的用户手册，此处从略。需要注意的是，生成的 IP 信息文件(XXX_hw.tcl)与 Quartus 版本有关。只要用同一版本的 Quartus 生成了 IP 信息文件，就可以拷贝到其他工程中，直接在 SOPC 类调用自己设计的 IP 核。

2 基于 Avalon-MM Master BFM 的仿真测试

2.1 Avalon-MM Master BFM 简介

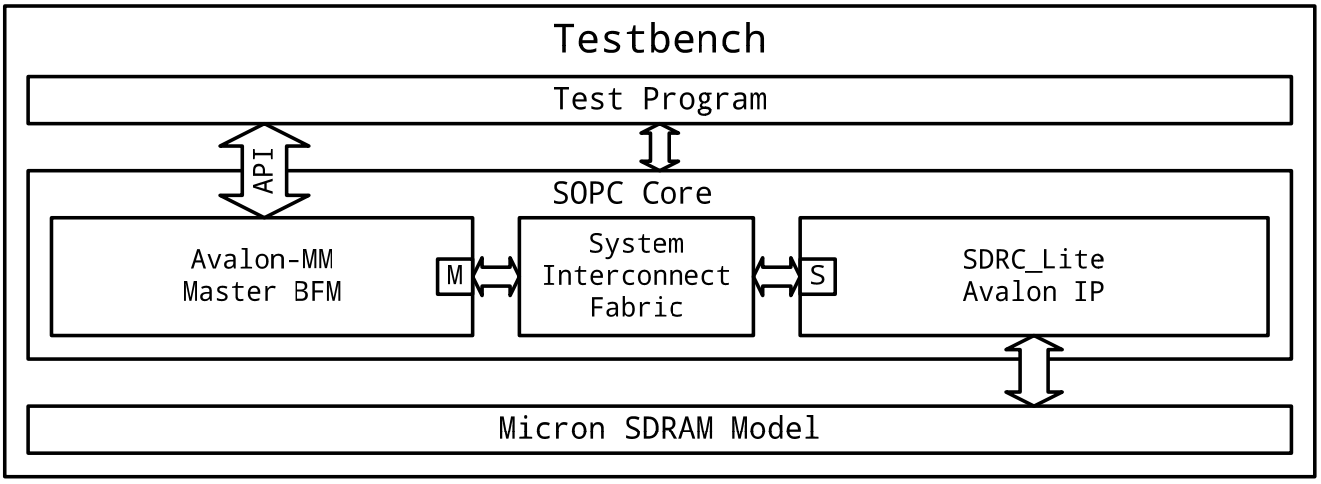
Avalon-MM Master BFM 实现了 Avalon-MM 协议，并提供了用于构建 Avalon-MM 操作和读取 Avalon-MM 相应的 API，用于对用户设计的具有 Avalon-MM Slave 接口的 IP 核进行验证。这些 API 简化了 testbench 的编写。同时，由于 Avalon-MM Master BFM 由 Altera 提供、Avalon-MM Slave 接口由用户编写，保证了测试的独立性，可以检测出用户对 Avalon 协议的误读。

Avalon-MM Master BFM 只用于仿真，不能用于硬件验证。

2.2 环境搭建

进行 Avalon-MM Master BFM 时，需要建立 Quartus 工程，搭建包含待测 IP 以及 Avalon-MM Master BFM 的 SOPC 系统，然后编写 testbench 调用该 SOPC 系统，通过 API 进行测试。

本次测试中搭建的 testbench 的框图如下：

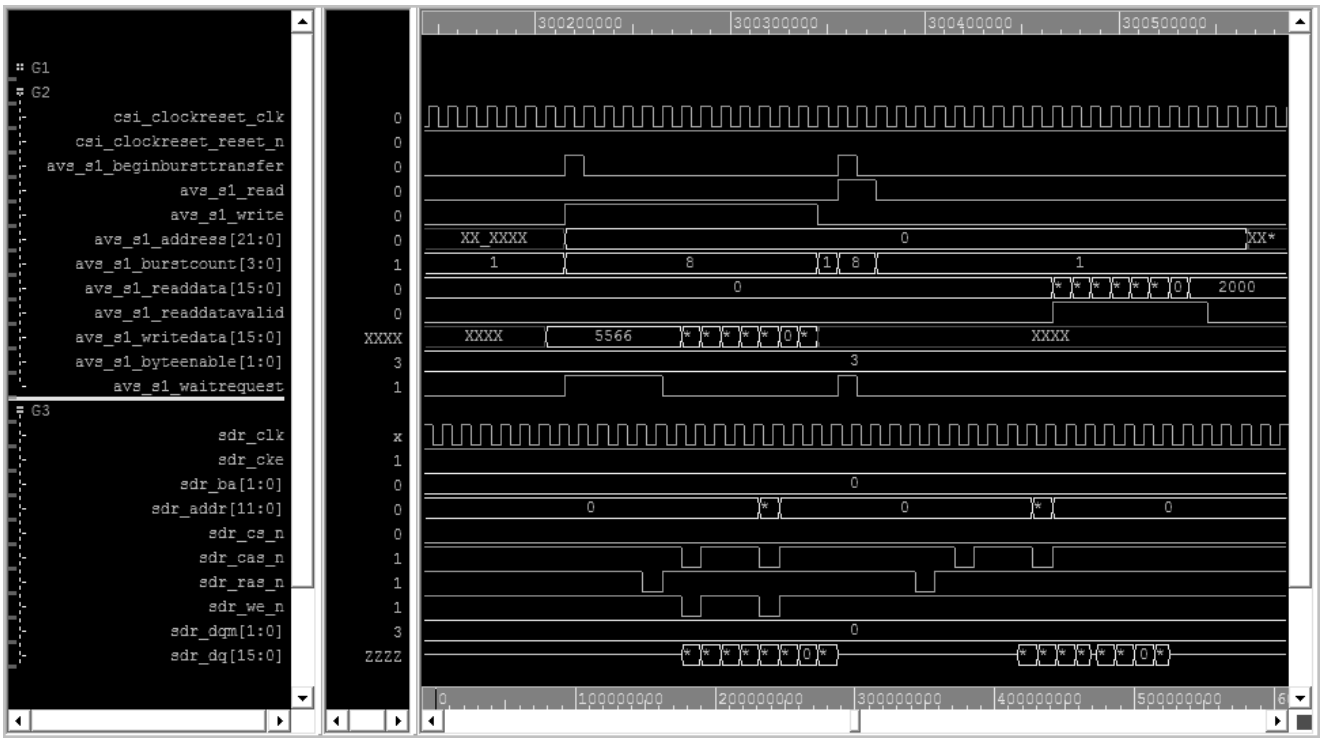


为了模拟与 SDRAM 的交互，依然加入了 Micron 的 SDRAM 模型。

测试程序通过 API 控制 SOPC 系统中的 Avalon-MM Master BFM，Master BFM 通过系统互连向 SDRC_Lite Avalon IP 发起 avalon-MM 操作，SDRC_Lite Avalon IP 负责与 SDRAM Model 交互。

2.3 仿真测试结果

下面的仿真图中显示了 SDRC_Lite Avalon IP 响应写、读请求各一次，结果符合要求。



分析更详细的仿真结果发现 SDRC_Lite Avalon IP 的设计符合 Avalon-MM 接口的要求，可以集成到 SOPC 系统中。此处不再罗列波形。

3 基于 Jtag to Avalon Master Bridge 的硬件测试

3.1 Jtag to Avalon Master Bridge 简介

Jtag to Avalon Master Bridge 提供了通过 JTAG 协议与 SOPC 系统交互的接口。

PC 机，可以使用 Altera 的 System Console 软件，经由 USB-Balster 下载线，与 FPGA 进行基于 JTAG 协议的通信。而通过 JTAG 协议可以控制 Jtag to Avalon Master Bridge 的 Avalon MM 接口操作，从而对 SOPC 中的待测单元进行读写。

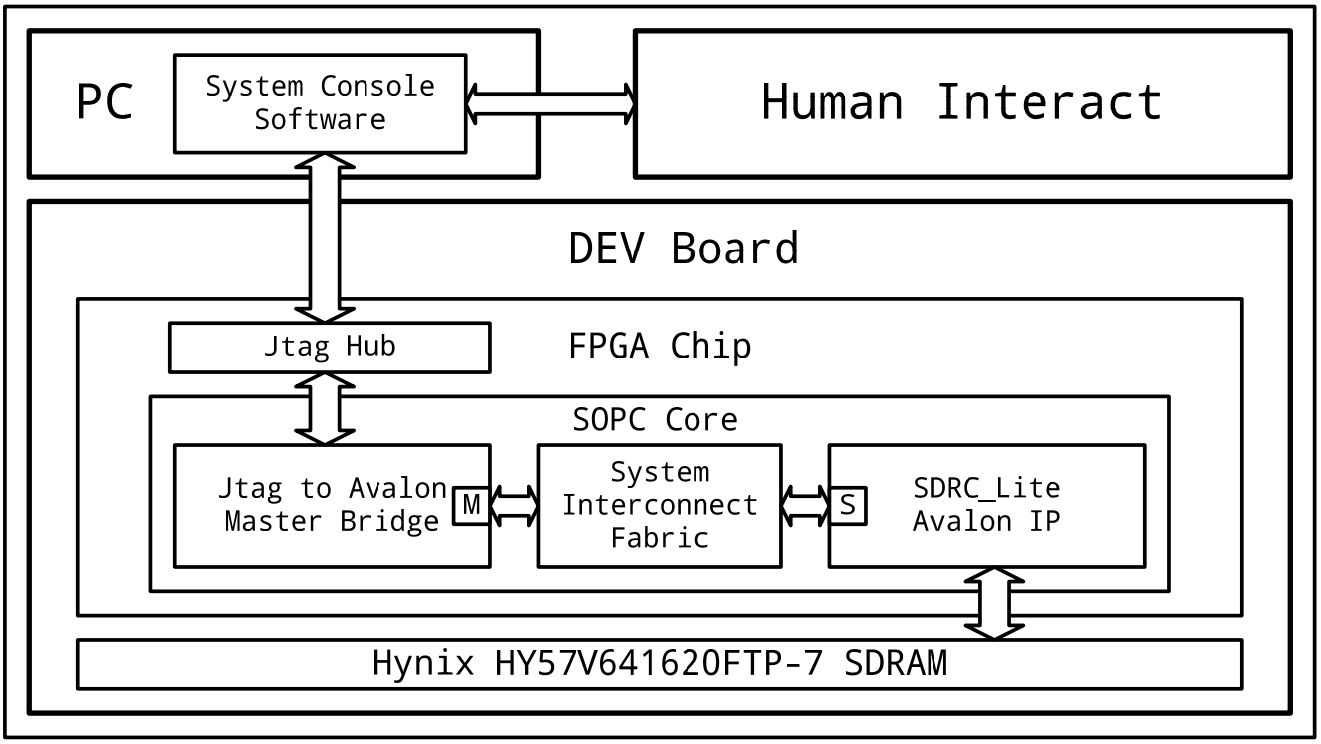
简而言之，就是可以通过 PC 机上的 System Console 软件对实现在 FPGA 上的具有 Avalon 接口的待测单元进行读写测试。

Jtag to Avalon Master Bridge 测试属于硬件测试。由于涉及 JTAG 协议与 FPGA 的下载电路，Jtag to Avalon Master Bridge 一般不用于仿真测试。

3.2 环境搭建

Jtag to Avalon Master Bridge 验证，依然使用 Altera Cyclone II ep2c8pq208c8 FPGA，以及 Hynix HY57V641620FTP-7 SDRAM。

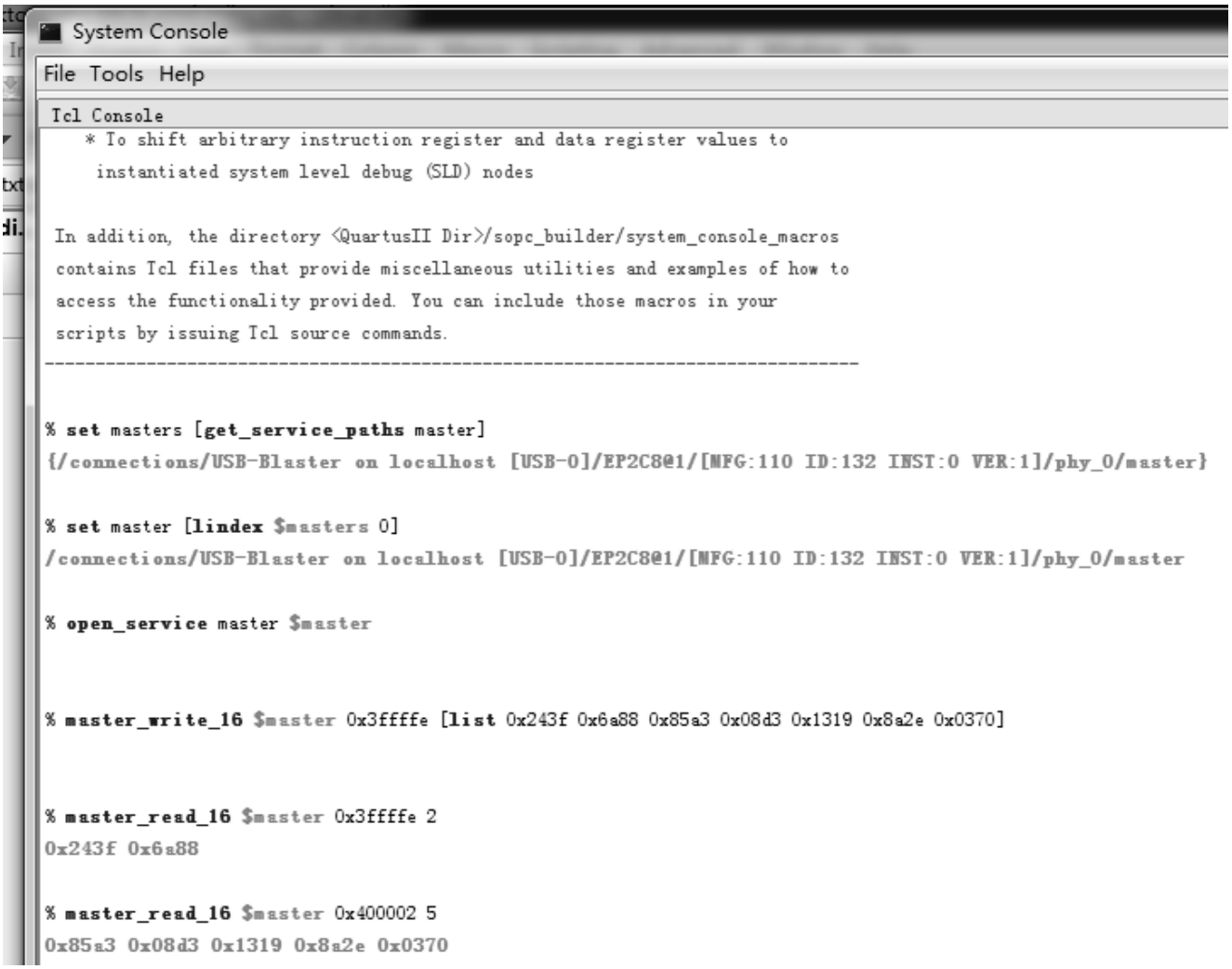
测试系统的框图如下：



用户可以操作电脑向 SDRAM 发送读写指令，并实时地观测到结果。System Console 的操作方法可以参考 Altera 的 Quartus 用户手册。

3.3 Jtag to Avalon Master Bridge 硬件测试结果

此处给出一张 System Console 的操作截图。



3.3.1 早期的小挫折

在我的早期设计版本中，SDRC_Lite Avalon IP 尚未支持任意长度的 Burst、不与 Burst 边界对齐的 Burst。那时，基于 Jtag to Avalon Master Bridge 的硬件读写总是不能正确完成。改进 Avalon Wrapper 后，硬件测试就通过了。

这种问题的出现与 SOPC 互连中的地址变换与 Burst 拆分有关。Jtag to Avalon Master Bridge 的地址是按字节(8 bit)来算的，而 SDRC_Lite Avalon IP 的地址输入是按 SDRAM Word 来算的(本例中为 16bit)，这之间存在一个转换关系。Jtag to Avalon Master Bridge 可以发起多种长度的 Burst，而且又要经过 SOPC 互连的拆分，导致给到 SDRC_Lite Avalon IP 的 Burst Length 变得不易预测。

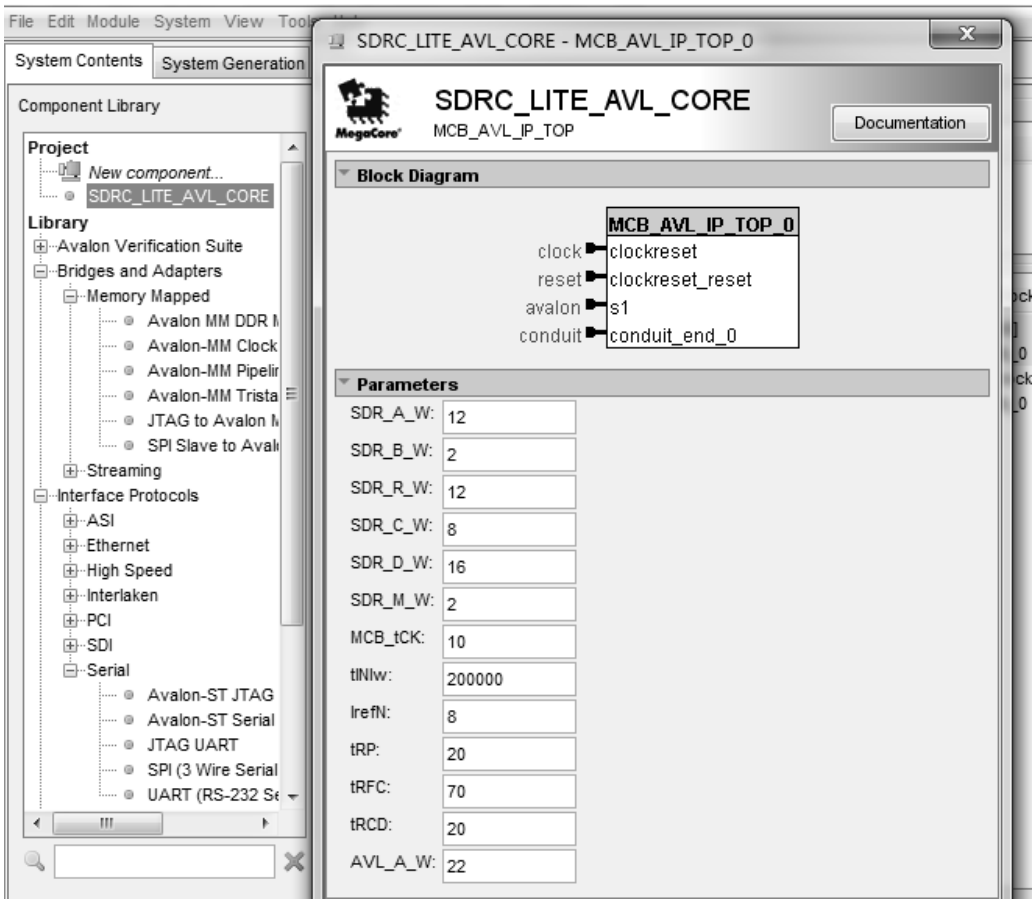
3.3.2 结果分析

在 System Console 中，我测试了多种 Burst Length 以及各种 Burst 情况，写与读的数据都

十分吻合，表明 SDRC_Lite 的功能符合要求。

写入后，等待若干分钟再读出，发现写入的数据没有丢失。这说明 SDRC_Lite 的刷新操作得当，能够维持住 SDRAM 中的数据。

至此，SDRC_Lite Avalon IP 通过了 Avalon-MM Master BFM 的仿真测试、Jtag to Avalon Master Bridge 的硬件测试。它作为基于 Close Page Policy 的 SDRAM 控制方案，可以替代基于 Open Page Policy 的 Altera 免费 SDRAM IP，承担起 SOPC 系统中的 SDRAM 存储控制任务。放一张在 SOPC 系统中实例化 SDRC_Lite Avalon IP 的截图如下：



4 本章参考文献

[1] Altera. Avalon Interface Specifications, ver 1.3. August 2010: Ch3 Avalon Memory-Mapped Interfaces

[2] Altera. SOPC Builder User Guide, ver 1.0. Dec. 2010: 全文参考

[3] Altera. Embedded Peripherals IP User Guide, ver 10.1.0. Dec. 2010: Ch2 SDRAM Controller Core

[4] Altera. Quartus II Handbook, ver 10.1. Dec. 2010: Ch12 Analyzing and Debugging Designs with the System Console

[5] Altera. Avalon Verification IP User Guide, ver 2.0. Jan. 2011: Section III Avalon-MM BFMs

[6] Altera. Making SOPC Builder Components. August 2011

[7] Altera. Using the SDRAM on Altera's DE2-115 Board with Verilog Designs. July 2010

[8] D. W. Hawkins. JTAG-to-Avalon-MM Tutorial, ver 1.0. March 2012

致谢

在毕业论文完成之际，我要对所有曾经关心、指导和帮助过我的领导、老师和同学表示最诚挚的感谢。

首先，要感谢我的导师范益波老师。他治学严谨、勤于科研。在本次毕业设计的过程中，从题目选取、资料整理，到设计概要讨论、测试方案选择，范老师都给了我最前瞻性的建议、最无私的支持和最热切的鼓励。在范老师的指导下，我的毕业设计，不但涉及了国内外论文中的优秀算法，而且还能与实验室项目紧密结合，既成为了一次自我锻炼的机会，也成为了一次向师兄们学习实验室项目基础知识并相互了解的机会。

接着，我要感谢一直指导我的袁兴师兄，在与师兄的接触中，我不但明确了科研的具体内容，而且也了解了实验室的历史，并被师兄们的好学与勤奋深深感染。

最后，我还要感谢我的母校，他“博学而笃志，切问而近思”的校训熏陶了我，使我能够在人生的道路上不畏艰险、勇敢向前。

梁晨

2012.6