学校代码:10246
学　号: 09300720291

学 士 学 位 论 文

DDR3 SDRAM内存控制器研究

院　　系: 信息科学与工程学院
专　　业: 微电子学
姓　　名: 陆彦珩
指 导 教 师: 范益波
完 成 日 期: 2013年6月7日

摘要

随着数据处理单元时钟频率的不断提高, 数据存取越来越成为整个系统的速度瓶颈, 为提高存储器数据存取速率, 各种旨在提高存储器数据传输速度的DRAM接口协议应运而生, DDR3正是其中最为先进的一代, 也是当下最为流行的内存规格。

存储器控制器是系统与存储器之间交换数据的桥梁和纽带, 一个设计良好的存储器控制器可以最大限度地发挥存储器带宽, 尽最大可能提高系统的数据传输速度。根据存储器的特性及其应用场合, 有针对性地编写存储器控制器, 可以显著提高系统有效带宽, 尽可能发挥存储器的特性。

本文首先回顾了DRAM的发展历程, 接着介绍DDR3　SDRAM的特性及其基本操作, 继而通过分析Altera公司的DDR3 SDRAM存储器控制器IP核了解存储器控制器的总体架构, 然后完成了一个基于AXI总线的DDR3　SDRAM存储器控制器硬件设计。控制器分为前端和后端两部分, 前端内嵌一个双端口SRAM作为高

速缓存并完成时钟域的转换，后端有基于close page policy和基于open page policy的两个版本，可用于不同的场合。最后对硬件设计进行了仿真，并将其部分集成到实验室H.264视频编码系统中。

关键词：DDR3 SDRAM，内存控制器，AXI总线

ABSTRACT

With the rapid increase of the frequency of the data processing unit, data access becomes the bottleneck of the whole system. In order to improve the data access rate of the memory, a lot of fast DRAM protocols come into being. Among those, DDR3 is the most advanced, and also is the most popular.
The memory controller is the bridge between the system and the memory. A well-designed memory controller can maximize the use of the limited memory bandwidth and enhance the data transmission speed. Design the memory controller according to the characteristics and the application situation of the memory, we can significantly improve the system's data access bandwidth and accelerate the data access speed.
This paper reviews the development process of the DRAM at first. Then, I introduce the characteristics of the DDR3 SDRAM and its basic operation. Next, I analyses the DDR3 SDRAM controller IP of the Altera company to comprehend the overall architecture of memory controller. And Then, I complete a DDR3 SDRAM memory controller hardware design based on the AXI bus. This controller can be divided into the front part and the back part. The front part converts the clock domain and uses a two-port SRAM as cache. I finish two versions of the bank-end design. One is based on close page policy, and another based on open page policy. They can be used on different occasions. Finally, I simulate the hardware design and integrate it into the H.264 video encoder system partly.

Keyword: DDR3 SDRAM, Memory Controller, AXI Bus

目录

图目录

表目录

第一章 绪论

随着工艺特征尺寸的减小，当代集成电路芯片的工作频率迅速提高，而存储器核心频率依旧保持在100MHz~200MHz，数据存储越来越成为电路系统的瓶颈。为提高数据存储吞吐率，SDR/DDR/DDR2/DDR3等高速接口协议应运而生。当下，设计良好的存储器控制器，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，简要介绍DRAM发展历程，从早期DRAM到SDRAM、DDR SDRAM、DDR2 SDRAM、DDR3 SDRAM，DRAM在四十多年的发展历程中不断创新变化，最终臻于成熟；紧接着，介绍本课题的内容；最后，给出整篇论文的结构。

1 DRAM发展历程：

1.1 早期DRAM：

1966年，Robert Dennard博士在IBM的Thomas J. Watson研究中心发明了DRAM存储器，五年之后，Intel发布了容量为1kbit 的Intel 1103存储器，DRAM开始在商业上取得成功。[1]

在 1970 年代中期，占据主流的是异步接口的 DRAM，那时的“Clocked DRAM”仅仅只是昙花一现。经典异步接口在每次读写前，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图1.1所示：

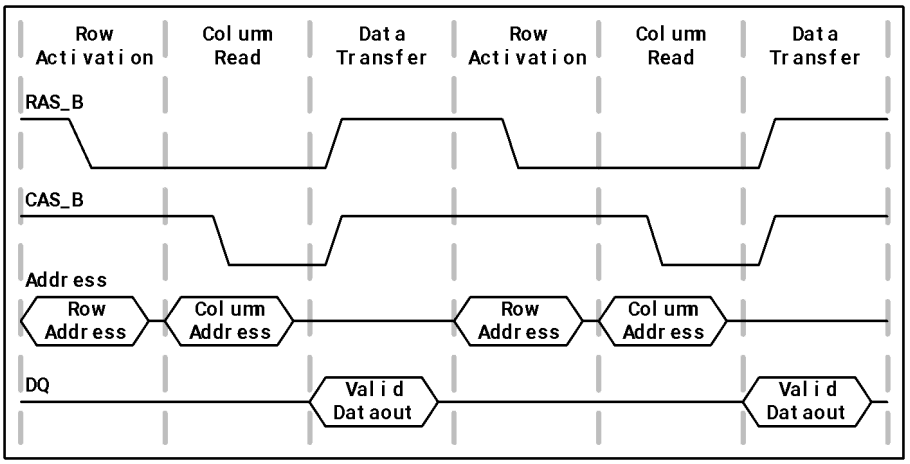
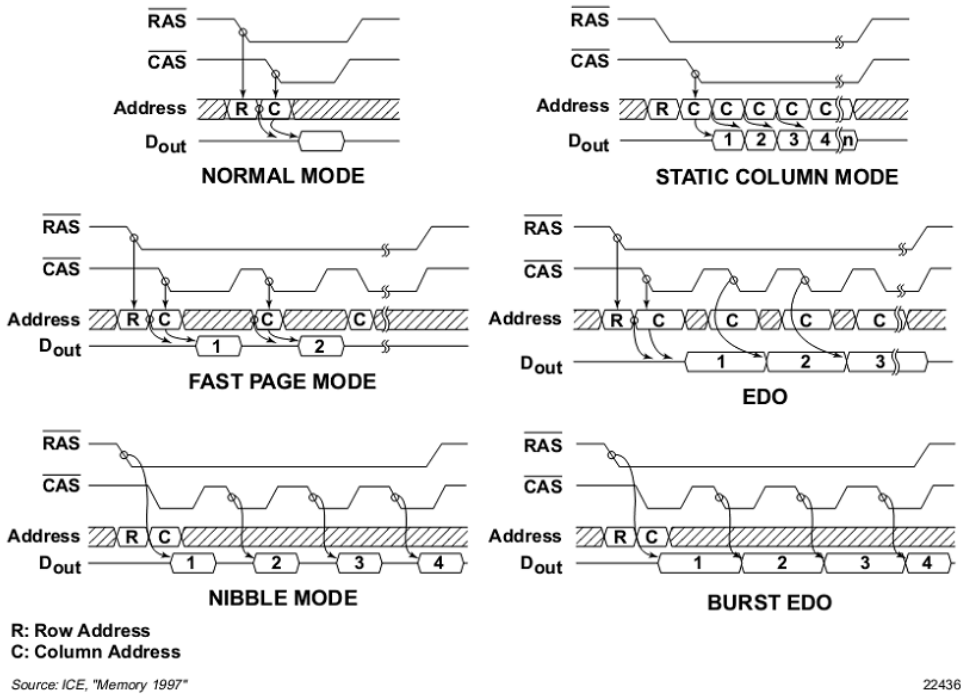


图1.1 异步DRAM读写典型时序

在异步 DRAM的时代，出现过诸如Fast Page Mode、Nibble mode、EDO（Extended Data Output）、Burst EDO等极富特点的接口改进，其中Page Mode 的出现具有里程碑式的意义。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阵列中，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。各种异步模式的时序对比如下图所示：



(上图摘自 ICE 公司的《Memory 1997》)

图1.2 异步DRAM的各种时序改进
(摘自ICE公司的《Memory 1997》[2])

1.2 SDRAM:

由于异步DRAM与同步总线在操作时存在额外等待时间，异步DRAM逐步被同步的SDRAM所取代。SDRAM，即Synchronous DRAM，意为同步的 DRAM，其数据和指令都与时钟上升沿对齐。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。最早的SDRAM由三星在1993 年生产。在 1996-2002 年期间，SDRAM 逐步取代了异步DRAM，逐步占领了内存市场。在 2003 年之后，SDRAM逐渐被存取速度更快的 DDR SDRAM 取代。

SDRAM 引入了同步时钟、多Bank 机制、流水线化的操作、Burst读写等多种机制。它流水线化、并行化的接口设计给控制系统带来了新的机遇，只要不违反管脚、时序延时的限制，多个 Bank 可以并肩工作。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。

所有这些都开启了DRAM控制系统的一个新篇章。[3]

1.3 DDR SDRAM:

DDR SDRAM，即Double Data Rate Synchronous Dynamic Random Access Memory，也就是双倍速率同步动态随机存取存储器，是在SDRAM基础上发展而来的新一代DRAM存储器，2003年取代SDRAM成为主流内存选择，2005年后逐

渐被DDR2 SDRAM取代。

与只在时钟信号上升沿传输数据的SDRAM相比，引入了2bit预取技术的DDR SDARM能够同时在时钟信号的上升沿和下降沿完成数据传输。这也就意味着在存储器核心频率不变的情况下可以将数据传输速率加倍，大大提高了数据传输速率。

除此之外，DDR SDRAM还引入了差分时钟、数据采样脉冲和片上DLL等技术。

1、差分时钟 CK 和 CK#。由于DRAM存储器工作在较高的时钟频率下，故稳定的时钟信号变得相当重要。DDR SDRAM将指令信号的采样点设定在差分时钟 CK 上升沿和 CK# 下降沿的交叉点上，同时将数据信号采样点设定在CK 和 CK# 的所有交叉点上，这样可以抑制温度、电阻等因素对时钟精度的影响，显著提高时钟精度。

2、数据采样脉冲 DQS。为信号完整性，DDR SDRAM引入与数据采样脉冲DQS。DQS 是一个双向端口，写数据时，DQS信号由控制器提供，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由内存器件提供，其上升沿和下降沿分别与两个数据各自的起始边沿对齐。

3、片上延时锁定回路 DLL。DDR SDRAM大大提高了数据传送频率，这也就意味着时钟树延时 (Clock Insertion Delay) 将会明显影响数据传输质量，甚至导致读写出错。为解决这一问题，DDR SDRAM引入片上延时锁定回路DLL（delay lock loop）产生精确的时间延时，调整数据传输时序。在通常情况下DLL处于开启状态，但也可以通过控制模式寄存器关闭DLL。[4,5]

1.4 DDR2 SDRAM

DDR2 SDRAM是在DDR SDRAM基础上发展起来的新一代DRAM存储器，它引入了4-bit预取技术，使得数据传输频率四倍于存储器核心频率，进一步提高了存储器数据传输效率。DDR2标准2003年制定，2005年起DDR2 SDRAM开始逐步取代DDR SDRAM，而2009年之后又逐渐被DDR3 SDRAM所取代。

除了加快数据传输速率，DDR2 SDRAM同样也拥有很多方面的改进：

1、用FBGA封装形式替代DDR的TSOP芯片封装形式，通过减小管脚长度减小寄生阻抗和寄生电容大小，同时提供了更好的电气性能与散热性，为内存的稳定工作与未来频率的发展提供了良好的保障。

2、输入电压由DDR标准的2.5V下降到1.8V电压，从而明显降低了器件的功耗及发热量。

3、引入双通道内存，这需要INTEL芯片组的支持，内存的CAS延迟、容量需要相同。INTEL的弹性双通道的出现使双通道的形成条件更加宽松，不同容量的内存甚至都能组建双通道，提高了内存性能。

4、离线驱动调整OCD（Off-Chip Driver）：DDR2通过离线驱动调整OCD可以提高信号的完整性。DDR2 SDRAM通过调整上拉（pull-up）/下拉（pull-down）的电阻值使两者电压相等。使用OCD通过减少DQ-DQS的倾斜来提高信号的完整性；通过控制电压来提高信号品质。

5、Post CAS：这是一个提高DDR2内存传输效率的改进。在Post CAS操作中，CAS信号（读写/命令）能够被插到RAS信号后面的一个时钟周期，CAS命令可以在附加延迟（Additive Latency）后面保持有效。在DDR2标准中，附加延时AL（Additive Latency）取代了DDR标准中的tRCD（RAS到CAS和延迟），AL周期数可以在0，1，2，3，4中进行设置。由于CAS信号放在了RAS信号后面一个时钟周期，因此ACT和CAS信号永远也不会产生碰撞冲突。

6、片内终结器ODT：关于ODT的详细介绍将在第二章介绍DDR3标准下的动态ODT时给出。[5,6]

1.5 DDR3 SDRAM及未来展望：

由于DDR2依旧不能满足日益加快的数据传输速率要求，所以数据传输速率更快的DDR3便应运而生了。DDR3采用8-bit预取技术，在一个存储器核心周期内可以传输8bit的数据，数据传输能力在DDR2基础上再度加倍。同时DDR3的工作电压进一步下降到1.5V，能够以更低的功耗正常工作。DDR3 SDRAM标准规范2006年开始开发，到2009年DDR3取代DDR2成为内存市场的主打产品，直到现在，DDR3依旧是最为先进和最为流行的内存存储器。

展望未来，DDR4内存的标准规范已经制定完成，未来几年将会逐步取代当下如日中天的DDR3，成为内存领域的主打产品。在可以预见的未来，伴随着处理器速度的不断提高，内存接口技术也将不断更新换代，不断追求更快的数据传输速率。

2 本课题内容介绍：

本课题从实验室视频编解码系统需求的角度出发，研究基于AXI总线的DDR3 SDRAM控制器设计，研究的主要内容包括：

1、DDR3 SDRAM特性及关键技术分析；
2、DDR3 SDRAM基本操作及时序约束分析；
3、Altera DDR3 SDRAM控制器IP核设计原理分析；
4、DDR3 SDRAM存储器控制器的总体功能及结构；
5、基于AXI总线的DDR3 SDRAM存储器控制器设计；
6、存储器控制器的效率改进方法。

3 论文结构：

整个论文的结构安排如下：

第一章为绪论，介绍DRAM发展历程、课题内容及论文结构。

第二章为DDR3 SDRAM特性及DDR3 SDRAM基本操作介绍，主要介绍DDR3 SDRAM引入的一些新特性及其基本操作。

第三章为Altera 公司DDR3 SDRAM 控制器IP核介绍，简单介绍Altera公司DDR3 SDRAM控制器的总体结构及设计思路。

第四章为DDR3 SDRAM控制器后端设计，介绍了一个通用性的DDR3

SDRAM控制器，主要负责在满足DDR3　SDRAM时序要求的前提下完成DDR3 SDRAM的初始化、读写、刷新等基本操作。

第五章为基于AXI总线的内存控制器前端设计，介绍了一个带双端口SRAM的存储器控制器前端设计，读写数据经AXI总线传输到存储器控制器slave后将优先写入一个较小的SRAM中，当行地址不匹配时再对SDRAM进行读写操作，尽最大可能性挖掘读写地址的局部性，提高读写效率。

第六章为DDR3 内存控制器的仿真结果，介绍上述设计的ModelSim仿真结果并加以分析。

第七章为总结及展望，对这一次毕业设计成果进行总结并给出进一步优化性能的方案。

第二章 DDR3 SDRAM特性及DDR3 SDRAM基本操作介绍

DDR3 SDRAM是当下最为流行的内存产品，相比于DDR2，它拥有很多新增特性，同时，它的管脚定义、模式寄存器定义及基本操作也比较复杂。在设计内存控制器之前，我们有必要先了解一下DDR3 SDRAM的新增特性及其他方面的一些基本情况。

本章分为三部分：第一部分介绍DDR3 SDRAM新增的部分特性，了解这些特性是设计性能良好的存储器控制器的必备条件；第二部分介绍DDR3 SDRAM的管脚定义及基本操作；第三部分关注存储器的模式寄存器定义。

1、DDR3 SDRAM特性介绍：

DDR3 SDRAM大大提高了数据传输速率，数据传输频率达到了1333MHz以上，这也就意味着信号的完整性及功耗将成为制约存储器性能的重要瓶颈。出于提高信号完整性及降低功耗的考虑，DDR3 SDRAM将工作电压进一步降低到了1.5V，同时还引入了诸多新技术。[5,7,8]

1.1 动态ODT（on-die termination）技术：

ODT即片内终结器。对存储器而言，为了防止数据线终端反射信号，它需要一些终结电阻以提高数据的完整性。最初的终结电阻都是做在印刷电路板上的，这一方面大大提高了主板的成本，另一方面由于不同内存模组对终结电阻值要求不一，主板上的终结电阻与内存模组间常常存在失配，进而也会影响信号的完整性，同时，主板上的终结电阻也会引入不需要的噪声信号。于是，从DDR2开始，终结电阻开始放置在存储器内部，存储器可以根据自己的特点设置终结电阻值，可以提高信号完整性并降低主板成本。

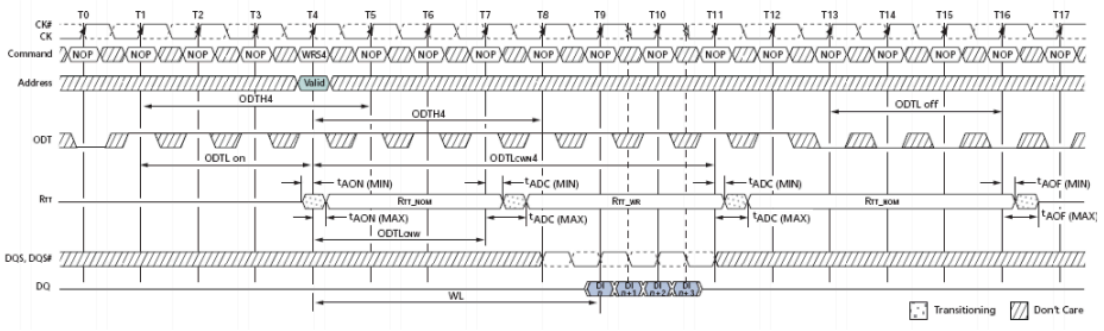


图2.1 DDR3 SDRAM读写时序简图

DDR3在DDR2基础上进一步引入了动态ODT技术，亦即在读写切换时，DDR3内存会在原始ODT和动态ODT做相应的切换。当读取或空闲时，ODT的值会是 20, 30, 40, 60, 120 欧姆之一（由扩展模式寄存器EMR配置）；而写入时会切换至60或120欧姆（由EMR配置）。动态ODT技术的引入一方面进一步提高了DDR3信号稳定性，另一方面也对存储器控制器提出了控制信号要求，在DDR3控制器中必须有ODT模块以产生所需的动态ODT信号。

写数据前后ODT信号变化情况图2.1所示，写数据之前存储器处于空闲状态，

ODT置低，写数据时ODT置高以减小信号反射，写数据完成之后ODT继续置低。[9]

1.2 ZQ校准技术：

ZQ是DDR3新增的一个管脚，连接了一个240欧姆的低公差参考电阻。这个引脚通过一个命令集，通过片上校准引擎（On-Die Calibration Engine，ODCE）来自动校验数据输出驱动器导通电阻与ODT的终结电阻值。当系统发出这一指令后，将用相应的时钟周期（在加电与初始化之后用512个时钟周期，在退出自刷新操作后用256个时钟周期、在其他情况下用64个时钟周期）对导通电阻和ODT电阻进行重新校准。ZQ校准技术的引入使DDR3控制器设计中需要引入一个专门的ZQ校准状态，并在有需要时启动以完成ZQ校准。[5,10]

1.3 fly-by技术：

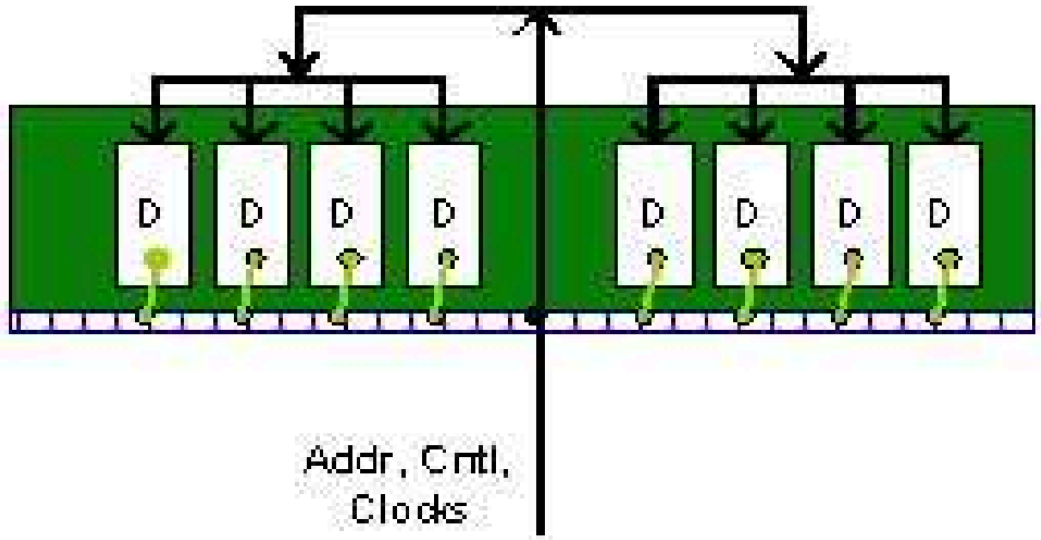


图2.2 DDR2 SDRAM数据写入拓扑结构[12]

DDR3的一个重要技术革新在于地址信号、命令信号及时钟信号采用了fly-by技术（参见图2.3）。对DDR2及更早的内存器件而言，它们采用树状结构（参见图2.2），信号到达每个内存单元的时间都是一样的，这引入了较多的节点和较大的信号反射噪声，同时一个地址命令信号要同时驱动多个内存器件，也要求地址/命令/控制与数据总线拥有更强的驱动能力。为了消除上述弊端，DDR3引入了fly-by拓扑结构，采用P2P连接方式，保证了更好的信号完整性，同时也可以得到更高的时钟频率。

fly-by拓扑结构的引入，也就导致信号到不同DDR3之间的时间延时是不一样的，因而在存储器控制器设计中会有Read leveling及Write leveling机制，它们保证数据信号（DQ、DQS）与时钟信号（CLK）间存在一定的延时。在系统初始化过程中，控制芯片会与SDRAM通过数据信号进行通讯，芯片根据收到的反馈结果信号进行内部延时调节处理，保证以后数据写入时序正确。

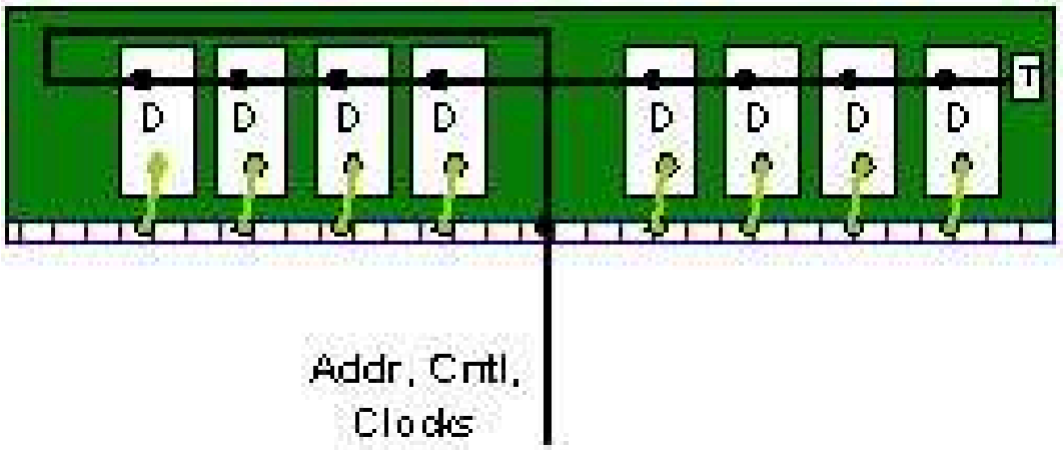


图2.3 DDR3 SDRAM数据写入拓扑结构[12]

1.4 复位(RESET):

复位是DDR3新增的一项重要功能，复位使DDR3的初始化处理变得相对简单。当Reset命令有效时，DDR3存储器将停止所有的操作，并切换至最少量活动的状态，以节省电力。在复位期间，DDR3存储器将关闭内在的大部分功能，所有数据接收与发送器都将关闭。所有内部的程序装置将复位，DLL(延迟锁定回路)与时钟电路将停止工作，并且不理睬数据总线上的任何动静，以使DDR3达到最节省电力的目的。

1.5 根据温度自刷新(ASR)技术:

刷新操作分为两种:自动刷新(Auto Refresh，简称AR)与自刷新(Self Refresh，简称SR)。为了最大的节省电力，DDR3采用了一种新型的自动刷新设计(ASR，Automatic Self—Refresh)。当开始ASR之后，将通过一个内置于DRAM芯片的温度传感器来控制刷新的频率，温度较低时，刷新频率比较慢，随着温度的升高，刷新的频率越来越快。这样在保证数据不丢失的情况下，尽量减少刷新频率，降低功耗。

此外，由于DDR3的ASR是可选设计，并非市场上所有的DDR3存储器都支持这一功能，因此还有一个附加的功能就是自刷新温度范围(SRT，Self-Refresh Temperature)。通过模式寄存器，可以选择两个温度范围，一个是普通的温度范围(例如O℃至85℃)，另一个是扩展温度范围，比如最高到95℃。对于DRAM内部设定的这两种温度范围，DRAM将以恒定的频率和电流进行刷新操作。

1.6 局部自刷新(PASR，PartiaI Array SeIf-Refresh)技术:

局部自刷新技术也是一个DDR3标准新引入的可选设计。在存储器实际使用过程中，并不是每一个Bank都会写入数据，这样在刷新过程中其实没有必要对所有Bank进行刷新操作。局部自刷新技术允许存储器只对部分使用过的Bank进行刷新操作，这样可以最大限度地减少因自刷新产生的电力消耗。

1.7 参考电压变化:

在DDR3标准中，参考电压信号VREF被细分为VREFCA和VREFDQ两部分，前者主要为命令与地址信号服务，而后者主要为数据总线服务。通过细分参考电压信号，可以显著提高参考电压信号的稳定性，进而提高信号完整性。

2、DDR3 SDRAM基本操作：

2.1 DDR3 SDRAM管脚分布：

在介绍DDR3 SDRAM基本操作之前，我们有必要了解一下它的管脚分布情况。单面DDR3 SDRAM一共有120个管脚（双面为240个），其中与存储器控制器设计最为相关的有：

管脚符号	类型	描述
A0-A9,A10/AP,A11,A12/BC#,A13	Input	地址输入。为ACTIVATE命令提供行地址同时为READ及WRITE命令的列地址和自动预充电位（A10），以便从某个bank的内存阵列里选出一个位置。A10在预充电命令期间被采样，以确定预充电命令执行范围：A10为低，则由BA[2:0]来选择预充电的Bank，而若A10为高，则对所有bank进行预充电操作。在LOAD MODE命令期间，地址输入提供了一个操作码。地址输入的参考是VREFCA。A12/BC#：在模式寄存器（MR）使能的时候，A12在READ和WRITE命令期间被采样，以决定burst chop(on-the-fly)是否会被执行（HIGH=BL8执行burst chop），或者LOW-BC4不进行burst chop。
BA0,BA1,BA2	Input	Bank地址输入。定义ACTIVATE、READ、WRITE或PRECHARGE命令是对哪一个bank操作的。BA[2:0]定义在LOAD MODE命令期间哪个模式（MR0、MR1、MR2）被装载，BA[2:0] 的参考是VREFCA
CK,CK#	Input	时钟。差分时钟输入，所有控制和地址输入信号在CK上升沿和CK#的下降沿交叉处被采样，输出数据选通（DQS、DQS#）参考与CK和CK#的交叉点。
CKE	Input	时钟使能。使能（高）和禁止（低）内部电路和DRAM上的时钟。由DDR3 SDRAM配置和操作模式决定特定电路被使能和禁止。CKE为低，提供PRECHARGE POWER-DOWN和SELF REFRESH操作（所有Bank都处于空闲），或者有效掉电（在任何bank里的行有效）。CKE与掉电状态的进入退出以及自刷新的进入同步。CKE与自刷新的退出异步，输入Buffer（除了CK、CK#、RESET#和ODT）在POWER-DOWN期间被禁止。输入Buffer（除了CKE和RESET#）在SELF REFRESH期间被禁止。CKE的参考是VREFCA。
CS#	Input	片选。使能（低）和禁止（高）命令译码，当CS#为高的时候，所有的命令被屏蔽，CS#提供了多RANK系统的RANK选择功能，CS#是命令代码的一部分，CS#的参考是VREFCA。
DM	Input	数据输入屏蔽。DM是写数据的输入屏蔽信号，在写期间，当伴随输入数据的DM信号被采样为高的时候，输入数据被屏蔽。虽然DM仅作为输入脚，但是，DM负载被设计成与DQ和DQS脚负载相匹配。DM的参考是VREFCA。DM可选作为TDQS
ODT	Input	片上终端使能。ODT使能（高）和禁止（低）片内终端电阻。在正常操作使能的时候，ODT仅对下面的管脚有效：DQ[7:0],DQS,DQS#和DM。如果通过LOAD MODE命令禁止，ODT输入被忽略。ODT的参考是VREFCA
RAS#,CAS#,WE#	Input	命令输入，这三个信号，连同CS#，定义一个命令，其参考是VREFCA
RESET#	Input	复位，低有效，参考是VSS，复位的断言是异步的。
DQ0-DQ7	I/O	数据输入/输出。双向数据，DQ[7:0]参考VREFDQ
DQS,DQS#	I/O	数据选通。读时是输出，边缘与读出的数据对齐。写时是输入，中心与写数据对齐。
TDQS,TDQS#	Output	终端数据选通。当TDQS使能时，DM禁止，TDQS和TDDS提供终端电阻。
VDD	Supply	电源电压，1.5V+/-0.075V
VDDQ	Supply	DQ电源，1.5V+/-0.075V。为了降低噪声，在芯片上进行了隔离
VREFCA	Supply	控制、命令、地址的参考电压。VREFCA在所有时刻（包括自刷新）都必须保持规定的电压
VREFDQ	Supply	数据的参考电压。VREFDQ在所有时刻（除了自刷新）都必须保持规定的电压
VSS	Supply	地

VSSQ	Supply	DQ地，为了降低噪声，在芯片上进行了隔离。
ZQ	Reference	输出驱动校准的外部参考。这个脚应该连接240ohm电阻到VSSQ

表2.1 DDR3 SDRAM主要管脚定义[11]

2.2 DDR3 SDRAM基本命令：

在DDR3 SDRAM的管脚中，最为重要的就是命令输入RAS#,CAS#,WE#，它们与片选信号CS#及最高地址位A13一起，共同控制DDR3 SDRAM的行为。不考虑休眠指令的DDR3 SDRAM基本操作真值表如下：

指令	缩写	cs_n	ras_n	cas_n	we_n	A13	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
Activate	ACT	0	0	1	1	1	Bank/ Row
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
ZQ Long	ZQCL	0	1	1	0	1	x
ZQ Short	ZQCS	0	1	1	0	0	x
No Operation	NOP	0	1	1	1	1	x
Deselect	DSEL	1	X	x	x	x	x

表2.2 DDR3 SDRAM基本指令

如上表所示，DDR3 SDRAM的主要操作包括：

配置模式寄存器（LMR）：按照实际应用的需求，完成对DDR3 SDRAM中四个模式寄存器的配置，在本设计中只允许在初始化DDR3 SDRAM时进行配置。

自动刷新（REF）：完成存储器刷新操作以保持数据。

预充电（PRE）：对某一个Bank进行预充电，关闭指定Bank中的行缓冲，在行地址未命中的情况下会用这个指令关闭行缓冲。

全部预充电（PREA）：对所有的Bank进行预充电，关闭所有Bank中的行缓冲，在刷新存储器数据前需要用到这个指令。

行激活（ACT）：根据输入地址将指定Bank中指定行数据全部写入行缓冲中，在行缓冲关闭情况下必须先使用这个指令激活指定行，才能进行后续的读写操作。

写命令(WR):向指定Bank的指定列中写入数据,在此之前必须保证行已被激活。

带预充电的写命令(WRA):在写命令完成后对相应的Bank进行预充电操作,写回数据并关闭行缓冲。

读命令(RD):读取指定Bank中指定列的数据,在此之前必须保证行已被激活。

带预充电的读命令(RDA):在读命令完成后对相应的Bank进行预充电操作,写回数据并关闭行缓冲。

长ZQ校准(ZQCL):用512个时钟周期,通过片上校准引擎自动校验数据输出驱动器导通电阻与ODT的终结电阻值,一般用于初始化过程。

短ZQ校准(ZQCS):用64个周期对导通电阻和ODT电阻进行重新校准,一般用于长时间读写之后对电阻进行较短的校准以提高信号完整度。

空闲(NOP):空操作。

未选择(DSEL):DDR3 SDRAM未被选中。

2.3 DDR3 SDRAM模式寄存器介绍:

DDR3 SDRAM一共有4个模式寄存器,设置DDR3 SDRAM各个方面的行为。在初始化过程中,系统通过LMR命令配置模式寄存器,寄存器的值通过Bank地址及Addr地址管脚写入。Bank地址用于判定模式寄存器序号,Bank地址为000则对应寄存器0,Bank地址为001则对应寄存器1,以此类推。由于DDR3 SDRAM大小不一致,地址管脚可能有13~16个不等,在配置模式寄存器是只有低13位是有意义的,A13以上地址应该输入低电平,在这里为讨论方便,只列出低13位地址。[8]

模式寄存器0的典型定义如下表所示:

A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
PD	WR	WR	WR	DLL	TM	CL	CL	CL	BT	CL	BL	BL

表2.3 模式寄存器0定义表

A12位为DLL Control for Precharge PD,控制预充电时延迟锁相环DLL的行为,设置为0则DLL关闭,设置为1则DLL打开。A11~A9为写恢复时间周期数设定,对应关系如下表所示:

A11~A9	000	001	010	011	100	101	110	111
WR	16	5	6	7	8	10	12	14

表2.4 写恢复时间设置真值表

A8位控制DLL是否复位,设置为0则不进行复位,设置为1进行复位操作。A7控制DDR3 SDRAM工作模式,设置为0则工作在正常状态下,设置为1工作在测试状态下。A6、A5、A4、A2用于设定存储器的弛豫时间(CL),A2为0时,000为接口标准保留位,001~007分别对应5到11个周期,A2为1时,000~010对应12到14个周期,其他为接口标准所保留。A3为读取簇发模式(Read Burst Type)位,A3为0则读取数据时串行输出(Sequential),为1则交错输出(Interleave)。A1、A0为BL(burst length)长度设定,00固定为8,01可在4或8中交换,10固定为4,11为标准保

留位。

模式寄存器1的典型定义如下表所示：

A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Qoff	TDQS	0	RTT	OCD	OCD	RTT	ODS	AL	AL	RTT	ODS	DLL

表2.5 模式寄存器1定义表

A12位输出缓冲使能位，为0时开启，为1时关闭。A11位为TDQS使能位，为0关闭，为1开启。A9、A6、A2控制ODT电阻值，000表示则ODT不开启，001设定为RZQ/4，010为RZQ/2，011为RZQ/6，100为RZQ/12，101为RZQ/8，其他为保留位（RZQ= 240 Ω）。A5、A1控制输出驱动阻抗值，00表示阻抗值为RZQ/6，01表示阻抗值为RZQ/7，其他情况表示阻抗值为RZQ/TBD。A7为写校正使能，设定为0则关闭，反之开启。A4、A3设定附加延时（AL）值，00表示不适用AL，01则设定为CL-1，02设定为CL-2，11为保留状态。A0为DLL使能位，为1关闭，为0开启。

模式寄存器2的典型定义如下表所示：

A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	RTT_WR		0	SRT	ASR	CWL	CWL	CWL	PASR	PASR	PASR

表2.6 模式寄存器2定义表

A10、A9用于设定写入时ODT电阻值，00表示不开启动态ODT，01表示电阻值为RZQ/4，10表示电阻值为RZQ/2，11为保留状态。A7设定自刷新温度范围，为0则采用正常温度范围，1采用拓展温度范围。A6设定自刷新方式，为0则手动刷新，为1则开启自刷新。A5~A3设定写弛豫时间CWL周期数，000~011对应5~8，其他为保留状态。A2~A0配置部分自刷新功能。

模式寄存器3的典型定义如下表所示：

A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	MPR	MPR LOC	

表2.7 模式寄存器3定义表

这个模式寄存器用于配置多功能寄存器功能（MPR，Multi Purpose Register），A2是使能位，A1A0设定PRR的位置（location，LOC）。这一功能在本次设计中并未用到，故这一模式寄存器值为全0。

第三章 Altera 公司DDR3 SDRAM 控制器IP核介绍

由于DDR3 SDRAM控制器硬件设计原理较为复杂，所以在设计自己的存储器控制器前先仔细研究了Altera公司提供的内存控制器IP核，并借此机会了解存储器控制器的基本结构。

本章首先大致介绍了Altera高性能存储器控制器IP核的构成，接着分三部分具体介绍Altera公司DDR3 SDRAM控制器IP核：第一部分主要关注存储器控制器中物理层与控制器间的信号接口协议并由此引出全速率方案、半速率方案与四分之一速率方案，第二部分关注Altera公司提供的两个物理层设计ALTMEMPHY与UniPHY，第三部分则关注IP核控制器部分的硬件设计。

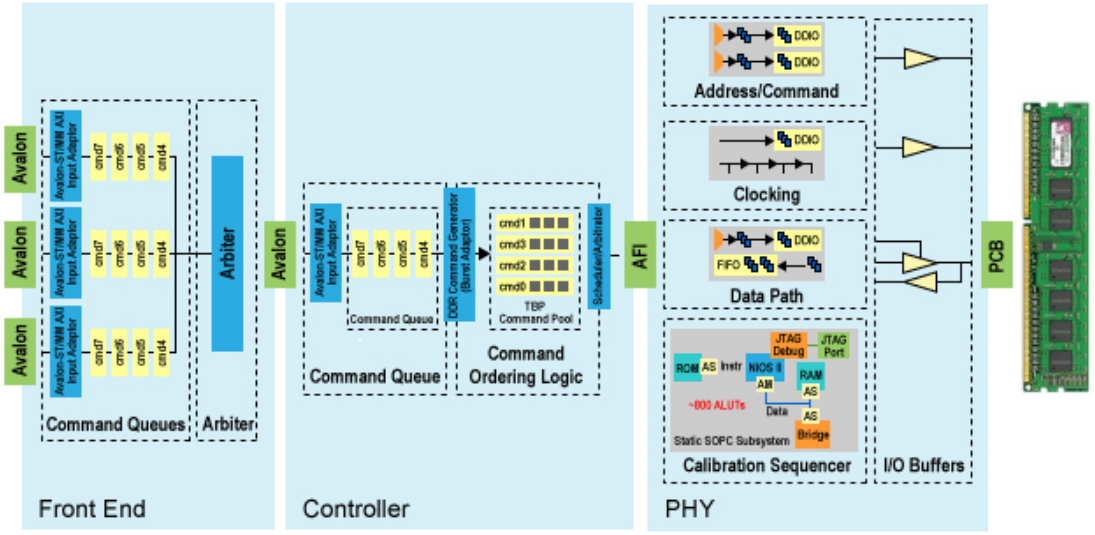


图3.1 Altera内存控制器顶层结构[13]

Altera的高性能存储器控制器基本构成如上图所示。它与上层通过多端口前端(Multi port front-end(MPFE)连接，可以接受多个模块的存储器访问请求，并通过裁决器判断优先级，依次经Avalon灌入控制器。中间是一个信号与数据控制器，与前端通过Avalon接口通信，内置一个命令队列，可以接收8条命令并对其进行排序，尽可能利用Bank交错和局部性原理提高数据传输效率。控制器同时产生控制存储器行为的各种控制信号，并通过AFI(Altera PHY interface)接口与物理接口通信。后端是物理接口部分，内置数据通路、地址与命令通路、PLL、DLL、自检测模块、矫正模块及I/O接口，并最终完成与PCB板上的内存芯片数据通信。

1、AFI接口与DFI接口：

DFI，即DDR PHY interface，是业界通用的DDR接口协议；AFI即Altera PHY interface，是Altera公司自己基于DFI开发的接口协议。与DFI协议相比，AFI去掉了DFI中一些与校正相关的接口，并添加了一些Altera自己的端口。[17,19]

AFI与DFI相比最为显著的差别在于，AFI是一个单一的数据速率接口，这意味着数据只在每个时钟周期的上升沿传输。然而DFI则在双数据速率操作，数据在时钟信号的上升和下降沿均会传输。

AFI的这一特性等于将部分中间控制器的功能移到了Altera自己的物理接口

中，同时Altera的物理接口也允许控制器工作在更低一些的频率上。对DDR3而言，数据及信号主要有全速率(Full)、半速率(Half)与四分之一(Quarter)三种速率方案。

全速率，即指控制器内部逻辑频率与DDR3接口频率一样，此时控制器部分数据宽度加倍，物理接口完成从SDR（单倍数据速率，只在上升沿出数据）到DDR（双倍数据速率，上升下降沿均出数据）的转换。全速率方案读写延时最小，但对控制器内部频率要求最高，实现较为困难。

Latency in Full-Rate Memory Clock Cycles							
Rate	**Controller Address & Command**	**PHY Address & Command**	**Memory Maximum Read**	**PHY Read Return**	**Controller Read Return**	**Round Trip**	**Round Trip Without Memory**
Quarter	20	EWER : 8	5–11	EWER: 16	8	EWER: 57–63	EWER: 52
		EWOR: 8		EWOR: 17		EWOR: 58–64	EWOR: 53
		OWER: 11		OWER: 17		OWER: 61–67	OWER: 56
		OWOR: 11		OWOR: 14		OWOR: 58–64	OWOR: 53
Half	10	EWER: 3	5–11	EWER: 7	4	EWER: 29–35	EWER: 24
		EWOR: 3		EWOR: 6		EWOR: 28–34	EWOR: 23
		OWER: 4		OWER: 6		OWER: 29–35	OWER: 24
		OWOR: 4		OWOR: 7		OWOR: 30–36	OWOR: 25
Full	5	0	5–11	4	10	24–30	19

Notes:

(1) EWER = Even write latency and even read latency
(2) EWOR = Even write latency and odd read latency
(3) OWER = Odd write latency and even read latency
(4) OWOR = Odd write latency and odd read latency

图3.2 基于不同速率模式的时序延时比较
(摘自Altera External Memory Interface Handbook p222[14])

半数据速率方案是指控制器内部逻辑频率是DDR3接口频率的一半。与全速率方案相比，半数据速率方案存储器控制器内部逻辑频率较低，大大简化了控制器内部设计，但延时比全速率方案长三成左右。

四分之一速率方案进一步降低了控制器内部逻辑频率，进一步简化控制器内部设计，但延时也大幅提升，几乎是全速率方案延时的三倍。

Altera的AFI接口，特别是半速率和四分之一速率方案的引入，大大简化了控制器部分的设计，但与此同时也付出了一定的时序代价。目前比较主流的方式都会选择半速率方案，它延时虽然略长一些，但控制器部分的频率要求低了很多，容易更好地完成，同时Altera的物理接口IP（ALTMEMPHY 与 UniPHY）更适宜半速率方案。

2、ALTMEMPHY 与 UniPHY

这是Altera的两个DDR控制器物理接口IP核，均通过AFI接口与前端控制器通信。前者是一个较老的物理接口版本，时序性能稍微差一些，功耗较低，支持DDR、DDR2、DDR3三代DDR内存，适合于较低版本的Altera FPGA板；后者是Altera公司近来主推的一个较新的物理接口，注重高性能，需要更多时钟信号，以面积和功耗的代价换取了更快的读写速度，只支持DDR2与DDR3，一般不用于控制DDR，适合于较高版本的FPGA板。两种物理接口都采用了软核与硬核相结合的设计方式，数据通路等模块被设计为不可更换的硬核，但校准模块等是可以被替换的软核，换言之这部分也是可以用户自行设计的。本论文重点介绍UniPHY物理接口。

2.1 UniPHY物理接口：

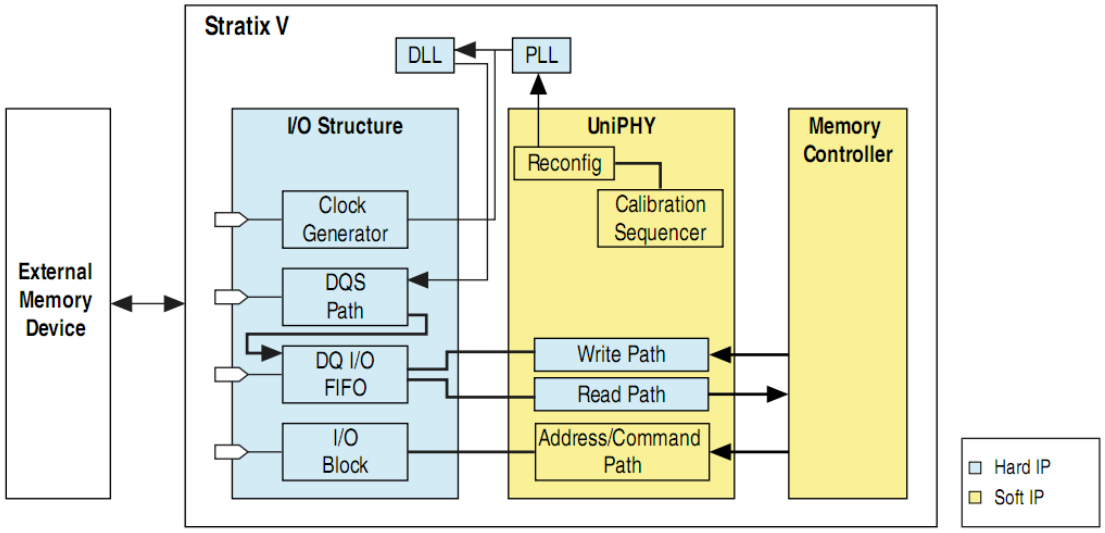


图3.3 UniPHY物理接口 IP核软硬核模块分布
(摘自Altera External Memory Interface Handbook p10[14])

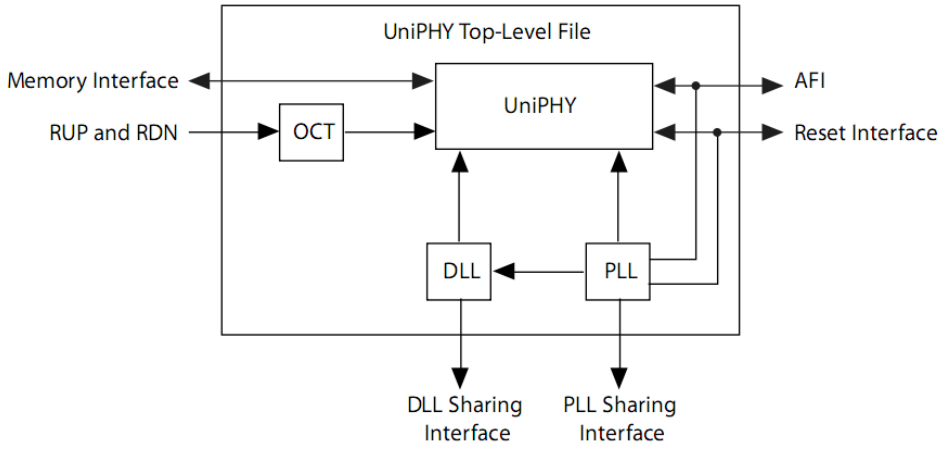


图3.4 UniPHY物理接口顶层模块图
(摘自Altera External Memory Interface Handbook p33[14])

UniPHY顶层结构如上图所示，除了本身的物理接口部分外还有可被共享的DLL（Delay Locked-Loop，延时锁定回路）和PLL（Phase Locked-Loop，锁相环）及OCT部分（On-Chip Termination，用于减低信号反射以及保持信号完整性）

UniPHY结构如下图所示，主要包括I/O端口、复位与时钟产生、地址与命令通路、读通路、写通路及自校准模块，通过Altera自行设计的AFI接口与前端控制器通信。[16]

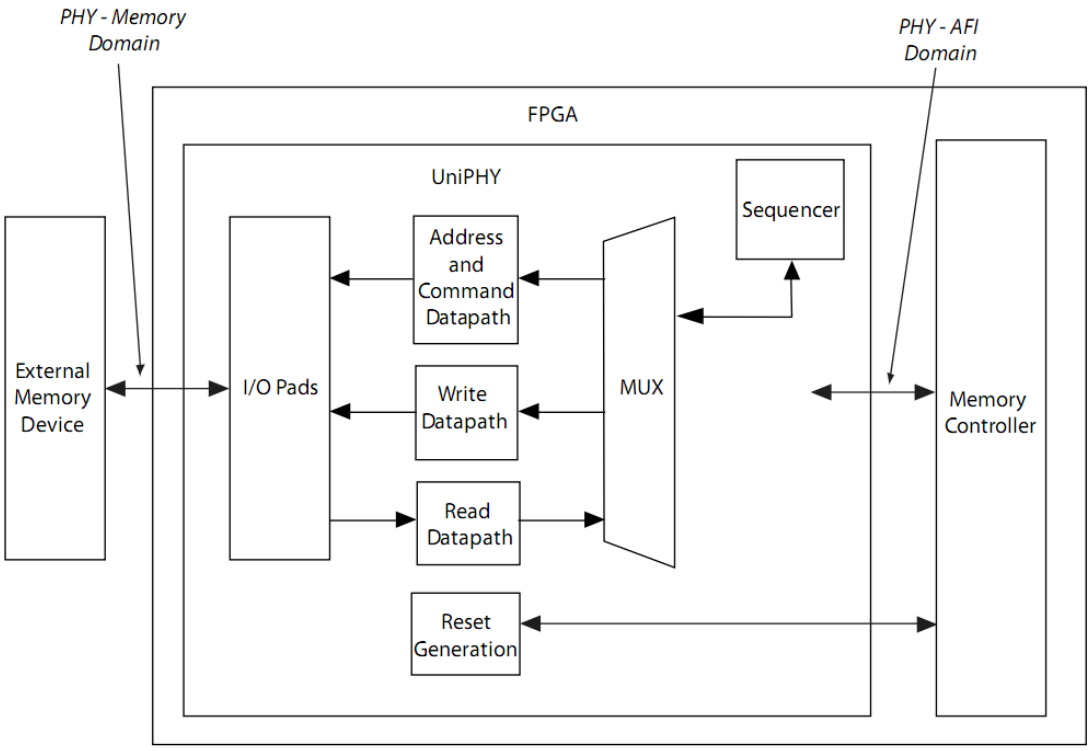


图3.5 UniPHY物理接口模块结构图
(摘自Altera External Memory Interface Handbook p18[14])

I/O端口部分完成输入输出功能。

复位和时钟产生模块完成复位功能，同时产生物理接口所需的时钟信号。由于与内存芯片连接的一侧是全速率数据传输的，而内部控制器一侧有全速率、半速率及四分之一速率三种模式，这就需要物理接口更具需要产生相应的时钟并调整好相应的相位，这就是时钟产生模块需要解决的问题。这一模块同时需要用到高速低偏移的平衡时钟树（DLL、PLL），这既可以是单独为物理接口服务的，也可以是与系统其他部分合用的。

地址与命令通路负责传送控制存储器行为的地址与命令信号。对物理接口而言，地址与命令信号是没有差别的，硬件将用相同的电路将他们传输到I/O端口。对全速率和半速率方案，地址和命令信号将以全速率的方式传输给存储器，而对四分之一速率方案，地址与命令信号将通过半速率方式（只在上升沿传输）传输给存储器。半数据速率转换为全速率的方式如下图所示，将AFI时钟信号相移270度后得到add_cmd_clk时钟信号，经过DDIO（double-data rate input/output）逻辑在时钟信号的上升沿和下降沿分别将低位和高位的地址及命令信号取出，完成数据速率加倍。

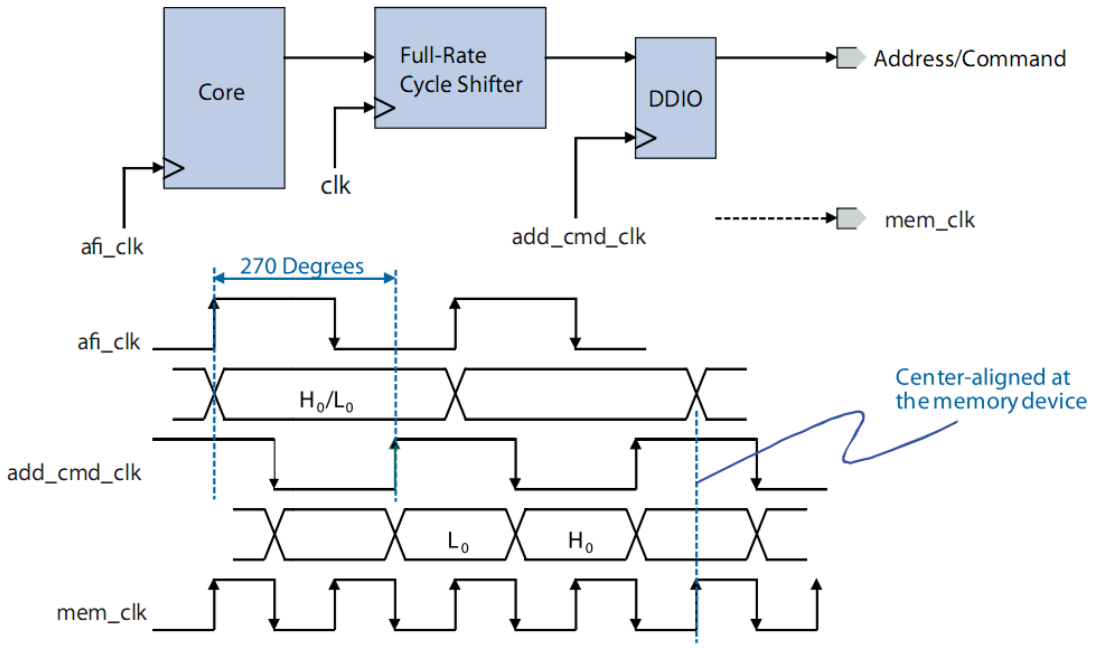


图 3.6 数据速率加倍硬件结构
(摘自Altera External Memory Interface Handbook p20[14])

写通路用来将写数据传输到I/O端口。半速率方案下的写通路结构如下图所示，左边一组DDIO完成从HDR到SDR的转换，右边一组DDIO完成SDR到DDR的数据转换，右下角则是leveling校准模块。对DDR3 SDRAM器件而言，正如前面所讲到的，它的地址与命令信号采用fly-by技术传输，这就要求数据信号与命令信号及时钟间拥有一定的偏移，这也就是需要leveling模块校准时序。在leveling模块中，初始时钟信号经过延迟链产生多个逻辑所需的时钟信号，精确保证数据命令及物理时钟同时到达I/O端口，以极小的时钟偏移完成数据写入，确保数据正确。

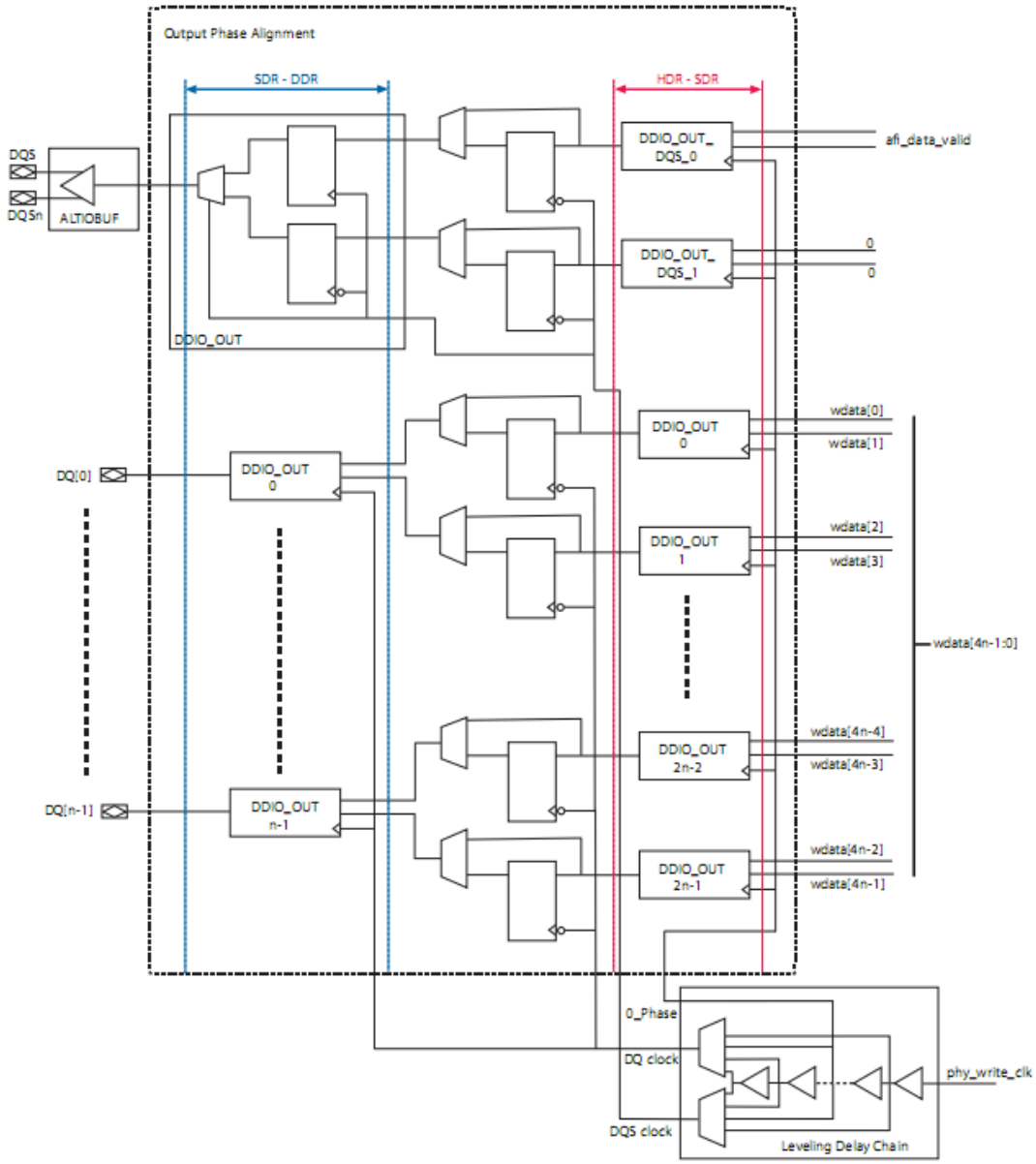


图3.7 写数据通路硬件结构
(摘自Altera External Memory Interface Handbook p21[14])

半速率方案下的读数据通路部分如下图所示，与写通路类似，这部分硬件最重要的功能之一就是将信号从双倍数据速率额DDR（double data rate）转换为半速率HDR（half-rate single data rate）。读通路中首先通过DDIO将DDR转换为SDR（single data rate），而后通过异步读FIFO（first in first out，先入先出队列）完成从全速率到半速率的转换。

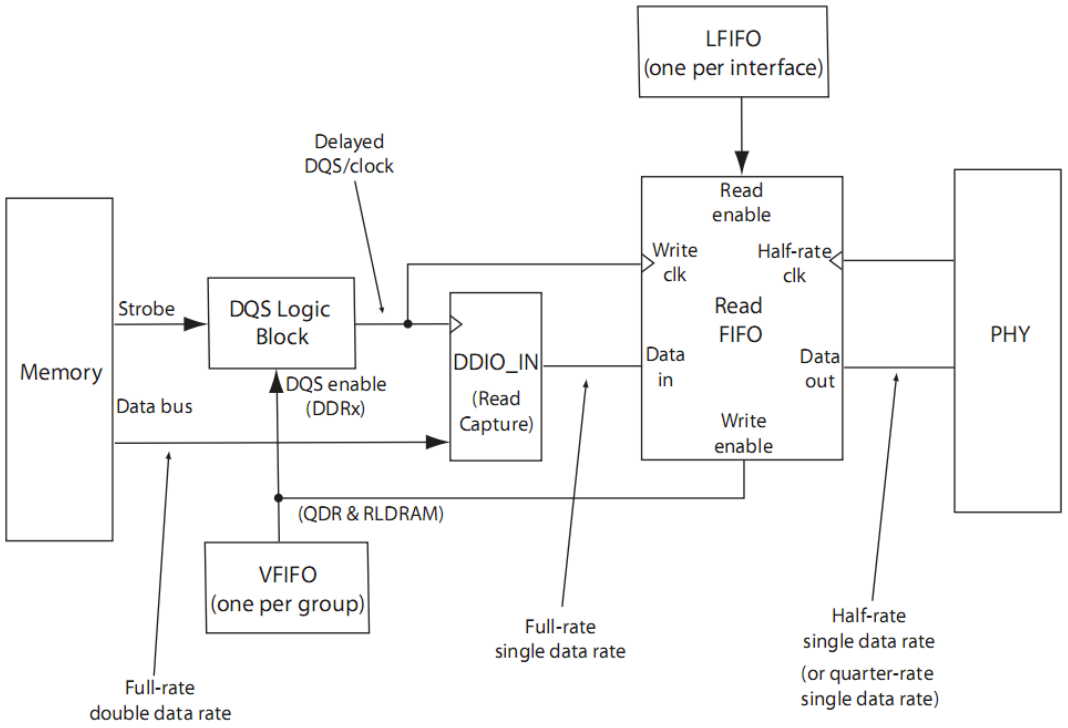


图3.8 读数据通路硬件结构
(摘自Altera External Memory Interface Handbook p23[14])

sequencer部分可以说是整个UniPHY IP核中最核心的部分了。

由于从控制器逻辑到DDR3存储器需要在让信号在主板上跑一段时间，这就将不可避免地引入一些干扰延时。在频率较低的时候这样的延时不会造成太大的影响，但对DDR3这种数据传输频率极高的存储器件而言类似的延时就是致命的。特别当时钟周期与传输延时大小可比拟时，它将导致大量的读写出错，进而影响这个系统运行，这是我们就需要一个硬件模块来完成时序的校正。sequencer模块正是用于完成这一功能。

sequencer模块通过调整延时链补偿传输延时以矫正物理接口的时序，同时这里的sequencer模块还完成DDR3的初始化与刷新功能，并能在任何需要的时候对时序进行重新校准。在sequencer收到请求校准的信号后，它将向DDR3 SDRAM器件发送试探性的读写命令信号，根据读写的结果判断在信号在主板上的延时时间，并据此调整延时链完成时序校准。

Altera的物理接口允许使用基于Nios Ⅱ设计的sequencer或者基于RTL设计的sequencer，两者拥有完全不同的接口管脚，所以相互之间不易于移植。

2.2 ALTMEMPHY物理接口：

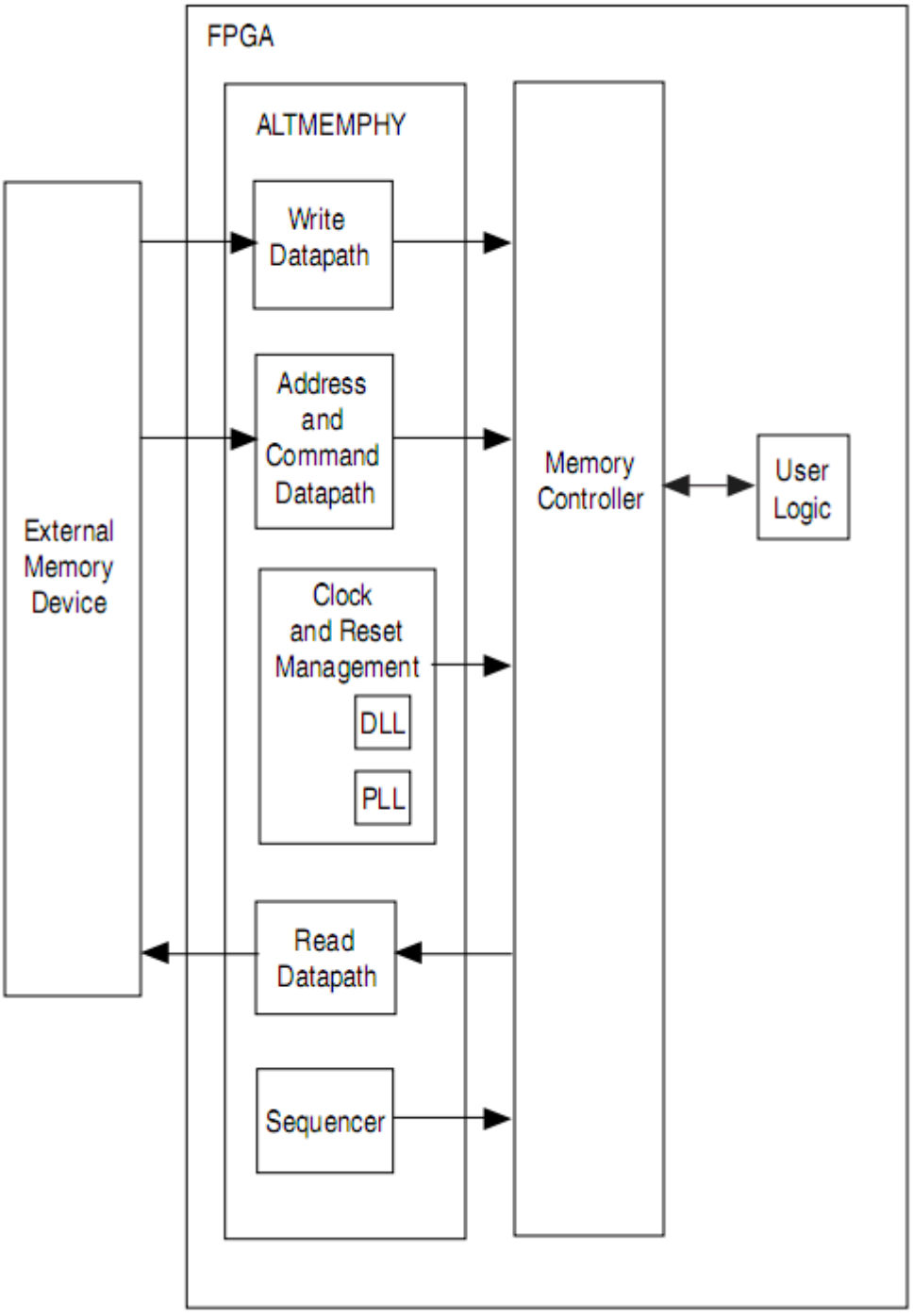


图3.9 ALTMEMPHY物理接口顶层结构
(摘自Altera External Memory Interface Handbook p74[14])

上图是ALTMEMPHY的系统应用及模块结构图，在接口方面ALTMEMPHY与UniPHY并无太多差别，它们都使用Altera自行开发的AFI接口与前端控制器通信，在内部结构上ALTMEMPHY比UniPHY明显要简单一些，性能也差一些。在本文中不对ALTMEMPHY做过多介绍。[15]

3、Altera HPC Ⅱ控制器IP介绍：

HPCⅡ控制器（High Performance controller 2），是Altera公司的第二代高性

能存储器控制器IP，它前端连接MPEE，后端连接物理接口ALTMEMPHY或UniPHY，是整个存储器控制器电路中最核心的部分。此IP核的模块结构如下图所示：

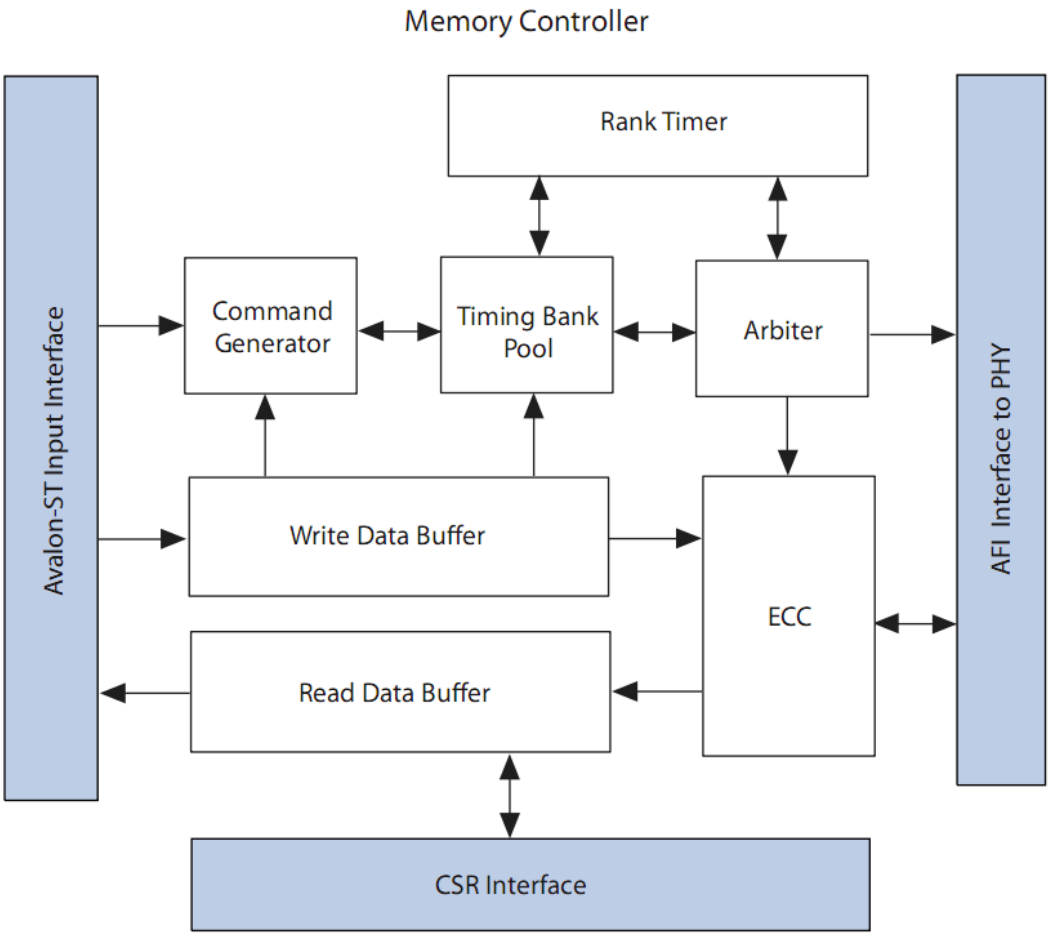


图3.10 Altera高性能内存控制器顶层结构[13]

图中蓝色部分是IP核的三个信号接口模块。前段接口主要负责存储器与总线或系统前端的通信，符合Avalon-ST总线接口协议。前段接口模块还内置一个AXI到Avalon的转化器，可将AXI接口转换为Avalon接口。后端接口是用Altera自创的AFI接口协议完成与后端UniPHY或ALTMEMPHY物理接口的通信。CSR接口即control and status register（标志寄存器）接口，它主要与系统端直接通信。CSR接口可用于设置整个controller的工作状态，同时，在ECC模块报错后也是通过CSR接口与上级系统进行通信。

上图中上方的四个模块主要负责控制DDR3 SDRAM的指令时序。其中命令产生模块（command generator）负责接收从前端Avalon总线或内部ECC模块传来的命令，并将这些命令处理后交给Timing Bank Pool模块暂存。Timing Bank Pool模块本质上这是个指令并行队列，它接收前面的数据读写请求并与Arbiter一起对前面的请求进行顺序排列，当从写数据缓冲器中收到写数据准备好信号就将写请求高效有序的传递到Arbiter模块。裁决器（Arbiter）模块裁决读写请求的前后顺序，当只有一个请求时请求将被迅速满足，而当有多个请求出现时裁决器采用先相应读请求，而后按时间顺序相应的裁决方案。Rank Timer模块保证了具体时

间信息的排序。它完成以下三个功能：1、保证在一个时间窗口内只有四个活跃请求；2、管理从读到写及从写到读的总线转换时间；3、管理不同Bank之间之间的激活延时时间。通过这四个模块的协同工作，控制器可以有效完成指令的重排，同时可通过Bank交错操作的思想提高数据传输效率。

下方的三个模块主要负责数据的传输处理。读数据缓冲（read data buffer）模块主要用于将ECC模块校正后的数据传送给Avalon接口。写数据缓冲模块主要用于放置等待写入的写数据，等到收到写入命令后输入ECC模块进行编码后写入物理接口。ECC（error-correcting code）模块则完成数据的校验。

由于信号干扰的存在，存储器读写过程中可能会出现部分bit数据的错误，而DDR3超过1GHz的频率导致信号干扰及失配导致部分数据出错的情况难以避免，这就需要存储器控制器模块能够及时检测和纠正这样的错误。ECC模块正是基于这一需要产生的。

ECC模块包括编码和解码两部分，后者同时完成单个bit数据错误的检测和纠正及两个bit数据错误的检测。每一笔数据在写入存储器前都会经过ECC模块进行汉明编码，一个周期后64、32、16及8bit数据会拓展为72、40、24及16bit数据，而后编码前数据和ECC数据分别写入存储器。当数据被读出时，ECC模块取出数据耗费一个周期进行解码，如果数据正确则直接送出，数据检测出一个错误则进行纠正并计数，如果检测出多个错误则ECC模块无法完成数据更正，此时直接报错中止数据读取，等待系统处理。

ECC模块以一个周期的时序代价，至多一倍的存储代价（一般为25%），换取了系统一定的自动纠错能力，有效提高了数据存储系统的稳定性，在当下的存储器控制器设计中应用广泛。

第四章 DDR3 SDRAM控制器后端设计

本次毕业设计主要期望完成一个适用于实验室视频编解码系统的DDR3 SDRAM存储器控制器设计。考虑到视频编解码系统段读写请求具有较强的规律性，地址局部性特征明显，较为适合使用基于open page policy的存储器控制器设计。但同时基于open page policy的存储器控制器设计较为艰难，故先行完成了一个基于close page policy的轻量级存储器控制器，进而完成基于open page policy的存储器控制器设计。

本章分为五部分，第一部分介绍存储器控制器设计中的行缓冲策略，第二部分介绍存储器初始化原理及硬件控制模块的实现，第三部分介绍存储器刷新原理及硬件控制模块的实现，第四部分介绍一个基于close page policy的存储器控制器设计，第五部分介绍一个基于open page policy的存储器控制器设计。

1、行缓冲策略：

1.1 行缓冲简介：

一般而言，DDR3 SDRAM有8个Bank，每个Bank都有一个行缓冲（Row Buffer）。行缓冲本质上就是一组灵敏放大器，DRAM将一整行数据保存在集成于片上的灵敏放大器整列中，访问同一行时，就不必重复进行行选通，这样就利用访问的空间局部性，提高了系统的传输性能。在实际操作中，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。显然，DDR3 SDRAM的访问存在以下三种情况：

1、行空闲：当前访问的行所在Bank行缓冲处于关闭状态，此时需要先用ACT指令激活相应行，将其数据写入行缓冲，继而才能发送读写指令及列地址完成数据读写。

2、行命中：当前访问的行正好保存在相应Bank的行缓冲中，这时不需要任何操作，直接发送读写指令及列地址就可完成数据读写。

3、行未命中：当前访问的行并不是保存在相应Bank行缓冲中的行，这时需要先发送PRE或PREA指令关闭行缓冲，等待预定时间后发送ACT指令激活相应行，接下来才能发送读写指令及列地址进行数据读写。

在上面三种情况中，行命中所需等待时间最短，行空闲次之，行未命中最长。需要注意的是，通过不同Bank之间的指令重排，在某一Bank进行预充电激活等操作的间隙，可以插入其他Bank的读写命令，这样某一笔读写所耗费的时间多少未必会真正影响整个系统的读写效率。在本次设计中限于能力有限并未涉及Bank交错操作及指令重排方面的问题，具体的介绍将在第七章总结与展望中给出。

1.2 close page policy和open page policy简介及比较：

在对待行缓冲的问题上，有close page policy与open page policy两种最基本的行缓冲策略，前者在每一次读写完成之后都给出PREA或PRA指令关闭所有

Bank的行缓冲，后者每一次读写完成之后都不进行预充电操作，保持行缓冲开启。

采用close page policy时，每一次读写访问中均只发生行空闲这一种情况，每一次读写均需要先激活指定行再进行数据读写，读写完成后都需要进行预充电操作，读写延时恒定，读延时为AL+CL，写延时为AL+CWL。

采用open page policy时，读写访问可能遭遇行空闲、行命中、行未命中三种情况。行空闲时读写延时延时与上文中讨论一致，行命中时无需进行任何操作就可以读写数据，这时读延时为CL，写延时为CWL，行未命中时在发送读写指令前需要先发送预充电指令和激活指令，所需读延时为tRP+AL+CL，写延时为tRP+AL+CWL。

从读写性能角度上来说，如果系统访问的空间时间局部性比较好，那么行命中的概率将比较高，此时采用open page policy比较合适，如果系统读写请求比较离散，那么行命中的概率将非常低，此时比较适合采用close page policy。对实验室视频编解码系统来说，系统访问的空间时间局部性会非常好，此时比较适合采用open page policy。

从硬件设计角度上来说，close page policy每一次访问都是行激活、列读写、预充电的操作，硬件较为简洁，所需的硬件面积也比较小。而open policy存在行空闲、行命中、行未命中三种情况，同时在判断行是否命中是必须关注行缓冲开启时间限制tRAS以及在刷新操作时必须先关闭所有行缓冲，故硬件设计复杂度将大大提高，所需逻辑元件数量也将大大增加。

从功耗角度上考虑，由于行缓冲本质上就是一系列灵敏放大器，所以行缓冲的关闭和开启将耗费相当大的能量，因而close page policy的功耗将明显大于open page policy，且访问的局部性越好，功耗的差别越大。

综上所述，close page policy硬件较为简洁、功耗相对较大（若出现完全没有行命中的极端情况，则两种策略功耗水平一样）、在读写请求空间时间局部性比较差的情况下效率较高，故一般适用于多核、网络服务器等访问请求来源较为分散的场合。而open page policy硬件复杂度较高、功耗相对较小，在读写请求空间时间局部性比较好的情况下读写效率较高，故一般适用于PC应用、单核SOC等访问请求较为集中的场合。

需要特别指出的是，这里的close page policy和open page policy只是两个比较极端的基本行缓冲策略，在实际应用中还会有各种糅合两种行缓冲策略的衍生策略，或者采用动态预测的方式轮换采用两种行缓冲策略。当然不论采用什么样的行缓冲策略，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。

在本次设计中，由于实验室视频编解码系统在读写请求上具有相当高的时间空间局部性，故最终决定采用open page policy，但考虑自身能力，先行完成了一个基于close page policy的存储器控制器后端硬件实现方案。

2、DDR3 SDRAM初始化原理及硬件控制模块的实现：

2.1 DDR3 SDRAM初始化原理：

DDR3 SDRAM必须以预先确定的顺序加电和初始化，配置一些基本的操作参数，否则会导致不确定的操作。系统初始化的过程如下：

1、电源电压和时钟稳定后，维持复位信号有效至少200微秒，拉高CKE(时钟使能)为高电平；

2、CKE变为低电平，10ns后复位信号失效，再等待500微秒，直到CKE变为高电平；

3、至少发送一个NOP(空操作)命令或DESEL(非选中)命令；

4、执行扩展模式寄存器EMR2设置命令；

5、执行扩展模式寄存器EMR3设置命令；

6、运行扩展模式寄存器EMR1设置命令，使能存储器芯片中的DLL；

7、执行扩展模式寄存器EMR设置命令，复位存储器芯片中的DLL；

8、执行ZQCL命令，进行ZQ校准；

9、等待锁相环相位锁定和ZQ校准完成。[20]

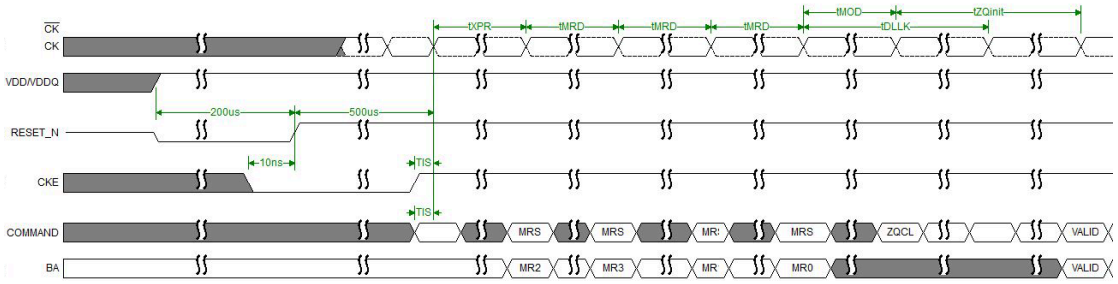


图4.1 DDR3 SDRAM初始化时序[21]

需要注意的是，在设置模式寄存器和发送空命令之后需要等待指定周期，以保证存储器的正常工作，DDR3 SDRAM初始化时序图如图所示。完成上述初始化步骤之后，存储器便进入就绪状态，等待控制器的访问命令。(关于模式寄存器的配置可参见本文第2章模式寄存器定义部分)

2.2初始化模块设计概要：

无论对哪一种行缓冲策略，初始化过程都是相同的，因而在两个版本的控制器后端设计中初始化模块相同。

初始化过程需要遍历以下状态：

I_IDLE	空闲状态
I_RESET	复位信号置低
I_RST_CKE	复位信号和时钟使能信号同时为低
I_CKE	时钟使能信号为低
I_CMD	发送空写入命令
I_MR2	装载扩展模式寄存器2
I_MR3	装载扩展模式寄存器3
I_MR1	装载扩展模式寄存器1
I_MR0	装载模式寄存器0
I_ZQ	进行长时间的ZQ电阻校准

表4.1 初始化模块状态表

当收到初始化开始命令init_begin后初始化模块状态从空闲(I_IDLE)跳转到(I_RESET)，继而按照时序要求遍历9个状态即可完成初始化过程，具体的初始化过程可参见第六章中队仿真结果的详细介绍。

3、DDR3 SDRAM 自动刷新原理及硬件控制模块的实现：

3.1 DDR3 SDRAM自动刷新原理：

由于DRAM采用电容保存数据，所以DRAM需要定期刷新以保持内部的数据。DDR3 SDRAM中，刷新操作是通过依次刷新所有Bank的同一行来完成的。在每一次刷新命令发送之前，我们需要预充电所有Bank，保证所有Bank的行缓冲均处于关闭状态，而后发送刷新命令(不需要发送行地址)，则DDR3 SDRAM会根据内部刷新计数器的值刷新某一行的数据并更新刷新计数器的值。因此存储器控制器只需要隔一定时间给DDR3 SDRAM发送刷新命令，DDR3 SDRAM就会自动将所有行一次刷新一遍，从而完成数据的保持。

目前公认的标准是，存储体中电容的数据有效保存期上限为64ms，也就是说我们需要在64ms中将DDR3 SDRAM所有行刷新一遍。如果我们采用均匀刷新的方式(这也是最为常用的刷新方式)，那么刷新时间间隔为64ms/行数，在这里我们取行数为8192，这就要求我们每7.8125us就必须发送一次刷新命令，JEDEC标准也要求我们在正常情况下每7.8us发送一次刷新命令。[8]

值得一提的是，由于刷新命令的存在，DDR3 SDRAM的有效带宽必然会有所降低，由于刷新时需要预充电所有Bank，故在采用open page policy时影响尤甚(采用close page policy时每次读写完成后都会预充电所有Bank，所以采用close page policy策略时，刷新命令前不用发送预充电指令)，这也是DRAM相对SRAM取得成本优势的同时必须要付出的代价。需要指出的是，在满足tREF(数据保持的最长时间，一般为64ms)和tRFC(两次刷新命令最小间距时间）两个时序约束条件的情况下，也可以采用在DDR3 SDRAM空闲时连续刷新的策略以提高整体的存取效率，但这也需要引入更加复杂的控制逻辑。在本次设计中我选择了最为基本的均匀刷新策略。

3.2 自动刷新控制模块设计概要：

这里的自动刷新控制模块仅仅只负责计数并发出刷新请求，故在下面两个版本的存储器后端设计中自动刷新控制模块也是一致的。在初始化过程中，自动刷新模块不启动，当初始化完成后，计数器开始计数。当计数值达到阈值(CtREFi)后计数器清零并重新开始计数，同时发出刷新请求，当刷新请求被相应之后刷新请求清零等待下一次计数值达到阈值，如此循环往复，即可完成刷新控制功能。

值得一提的是，本模块中选择在计数值达到阈值后立马清零重新计数，这虽然保证了发出刷新请求的时间是绝对均匀的，但一旦前一个刷新请求未被响应而计数器已经达到阈值，则会出现少刷新一次的情况引发不确定性。由于刷新在所有操作中拥有最高的优先级且最长的读写过程(Full Page)也不会耗费数千

个周期的时间，故这样的不确定性在实际应用中不存在。

4、基于close page policy的存储器控制器后端设计：

基于close page policy的内存控制器后端设计共分为控制模块（DDR3_ctl）、信号传输模块（DDR3_sig_ff）、数据传输模块（DDR3_dat_ff）三大部分，控制模块负责提供控制信号，信号传输模块负责将控制模块传来的指令按照DDR3接口协议要求转换成控制信号传递给DDR3 SDRAM器件，数据传输模块负责在时钟上升沿和下降沿均传输数据给器件，完成数据收发功能。

控制器模块中内置四个控制器：初始化控制模块（init_ctl）、刷新控制模块（ref_ctl）、命令控制模块（cmd_ctl）以及数据控制模块（dat_ctl），初始化模块与刷新模块已经在前两节介绍过了，这里重点介绍命令控制模块。

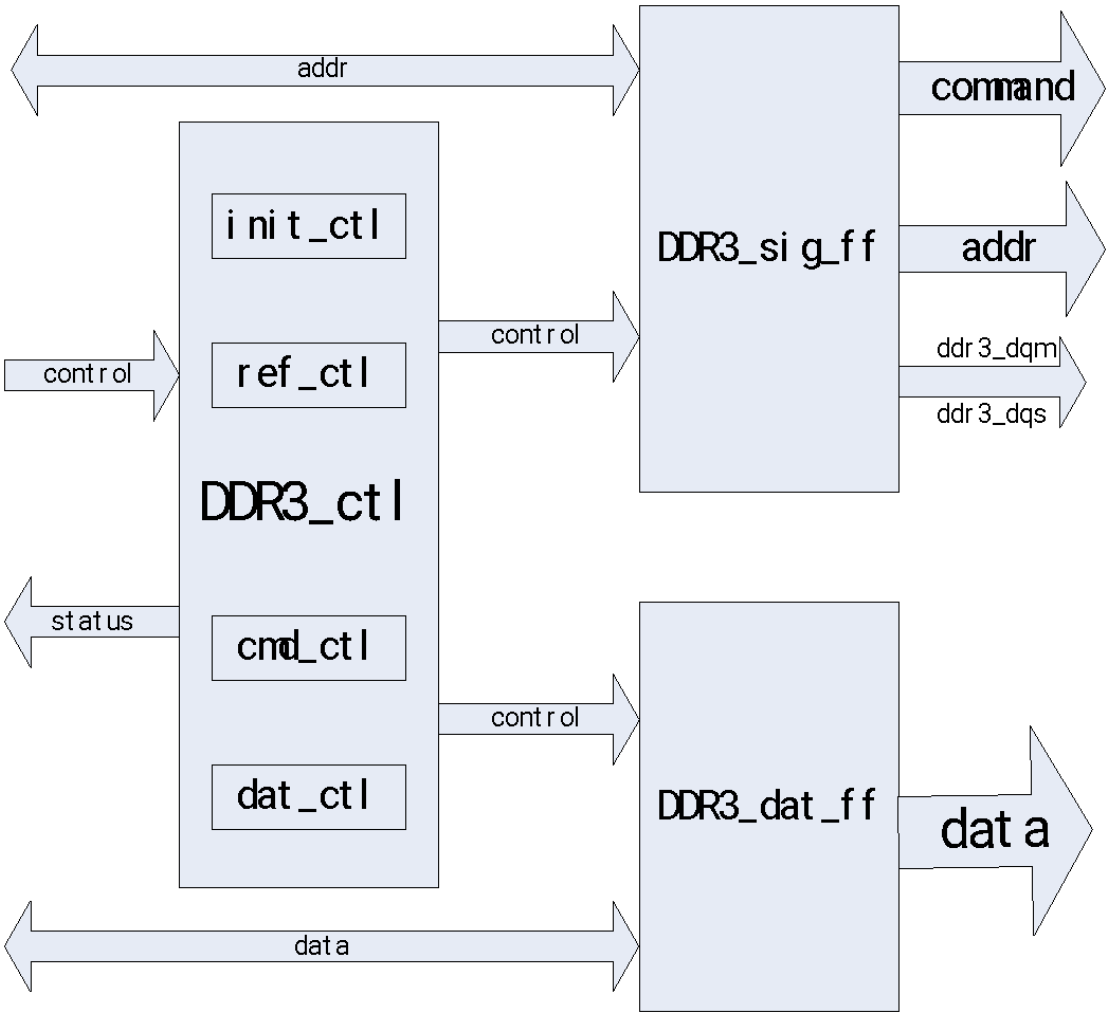


图4.2 基于close page policy策略控制器后端模块结构

命令控制模块负责控制整个控制器在初始化完成之后的工作。它最为核心的部分就是一个控制状态机。状态机共有6个状态：等待初始化状态（c_wait）、预备状态（c_ready）、刷新状态（c_ref）、激活状态（c_act）、带预充电的写状态（c_wra）、带预充电的读状态（c_rda），在后四个状态后面需要加上等待状态以满足等待的时序要求。

在初始化开始后，系统自动进入初始状态（c_wait），当收到初始化结束信号

ddr3_mcb_i_ready后转入预备状态(c_ready)。在预备状态下刷新请求信号具有最高优先级，一旦出现刷新请求信号ref_req，则系统进入刷新状态c_ref，刷新操作一般需要8个周期的延时，故系统进入刷新等待状态c_ref_w等待8个周期后才能进入c_ready状态等待下一个操作指令。

若系统处于预备状态且没有刷新请求，则一旦收到读写开始信号就进入激活状态c_act将所需读写的行放入行缓冲。继而按时序要求经过数个周期的等待状态c_act_w后按照读写方向信号ddr3_mcb_wr_n进入对应的读写状态(c_rda或c_wra)，完成读写过程后重新回到预备状态(c_ready)等待下一个指令。

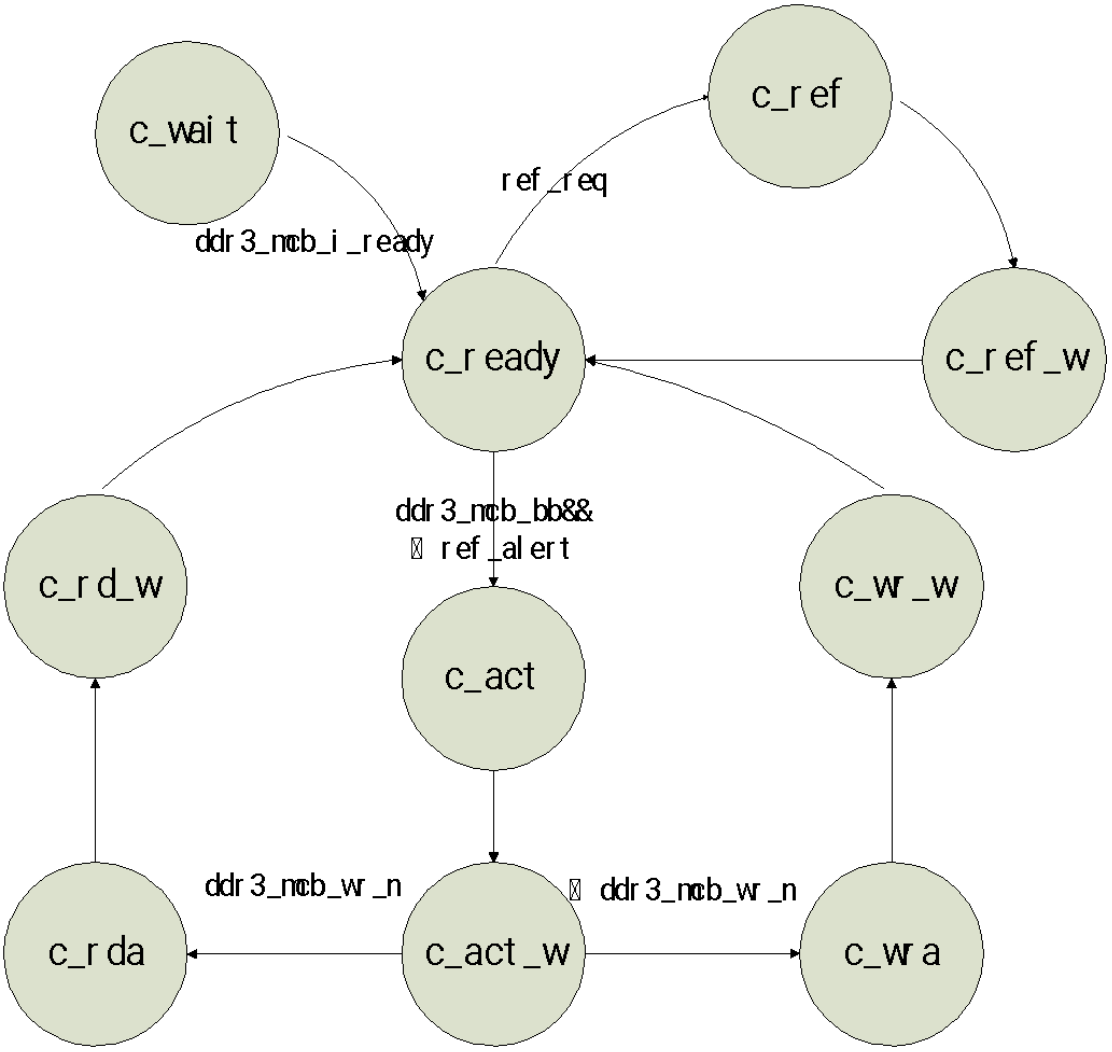


图4.3 基于close page policy策略控制器后端状态机

基于close page policy的内存控制器设计是比较轻量级的，整体状态转换并不复杂，按照时序要求一步步完成各项工作就可以完成设计，相对比较简单，面积也较小，但并不适用于局部性非常好的视频硬件编解码系统，所以后面将给出一个更为合适的基于open page policy的内存控制器后端设计方案。

5、基于open page policy的存储器控制器后端设计：

基于open page policy的存储器控制器后端设计在系统框架上与基于close page policy大致一样，但由于需要判断输入Bank与row地址与行缓冲当下的状态

是否匹配，所以在控制模块之前需要加上行地址控制模块（row_ctl）。

行地址控制模块负责判断当下读写请求的类型，并发出发出判断信号行命中row_hit、行未命中row_miss或行空闲row_empty。它内部有8个小的行地址控制模块，每个小模块内部有1个行地址寄存器及1个空闲标记，行地址寄存器记录当下某个Bank中行缓冲中寄存的数据的行地址，空闲标记则记录行缓冲是否打开。当出现一个读写请求时，行地址控制模块开始依据输入Bank地址将这个请求送至某一个小行地址控制模块中，而后这个模块根据空闲标记及行地址寄存器和输入行地址发出行命中、行未命中或者行空闲判断信号，并依据判断结果对空闲标记及行地址寄存器进行必要的操作。同时，当出现刷新信号时，由于刷新时需要将所有行缓冲关闭，故此时所有空闲标记全部置低。

基于open page policy的内存控制器后端与基于close page policy的内存控制器后端另一个重要区别在于其命令控制模块。

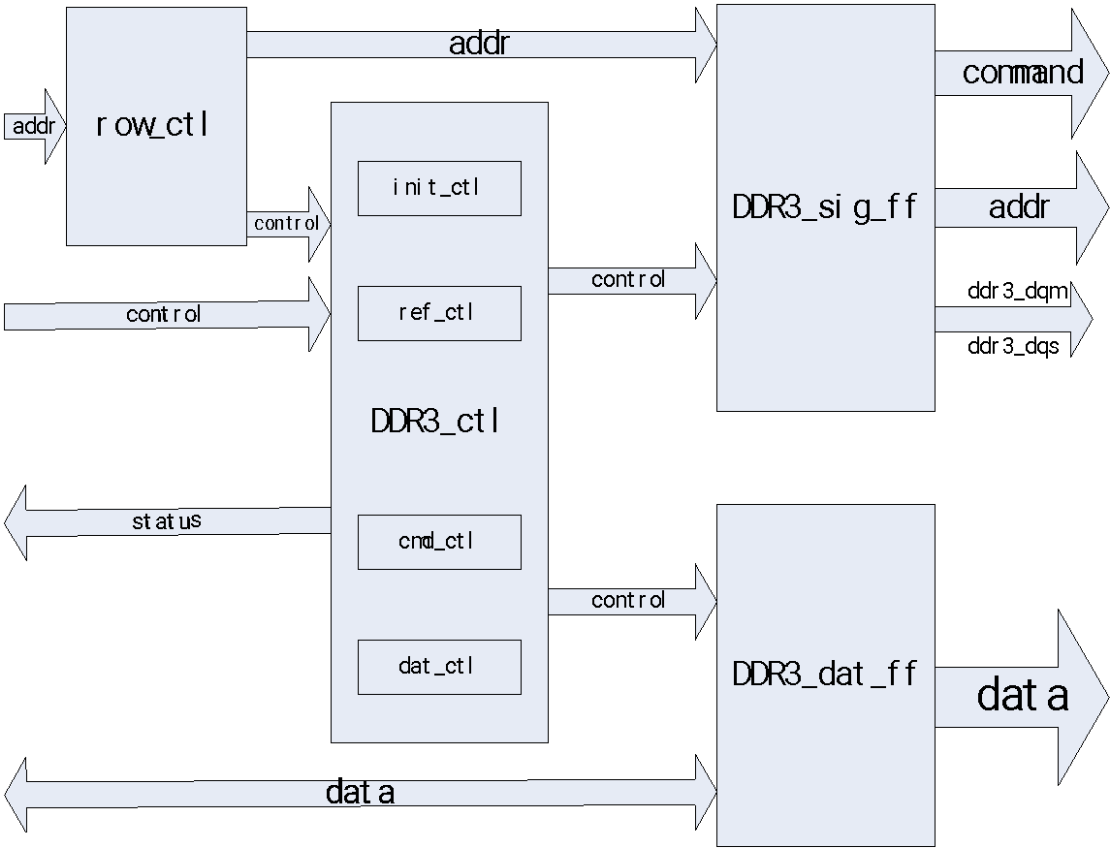


图4.4 基于open page policy策略控制器后端模块结构

对采用open page policy的方案而言，其状态机复杂度将明显提高，它拥有等待状态（c_wait）、预备状态（c_ready）、全部预充电状态（c_prea）、刷新状态（c_ref）、预充电状态（c_prec）、激活状态（c_act）、读状态（c_rd）、写状态（c_wr）八个状态，同时后六个状态还会有等待状态（图中未标出）。

系统初始化时状态机处于等待状态（c_wait），初始化完成后进入预备状态（c_ready），继而刷新请求到来时进入全部预充电状态（c_prea），继而进入刷新状态（c_ref）完成刷新操作。

系统处于预备状态且没有刷新指令时，收到行命中（row_hit）信号则直接按照ddr3_mcb_wr_n信号判断读写方向进入读或写状态（c_wr或c_rd）；收到行空

闲(row_empty)信号则先进入激活状态(c_act)激活对应行，再进行读写操作;收到行未命中(row_miss)信号则先进入预充电状态(c_prec)关闭对应Bank的行缓冲，继而进行激活操作和读写操作。

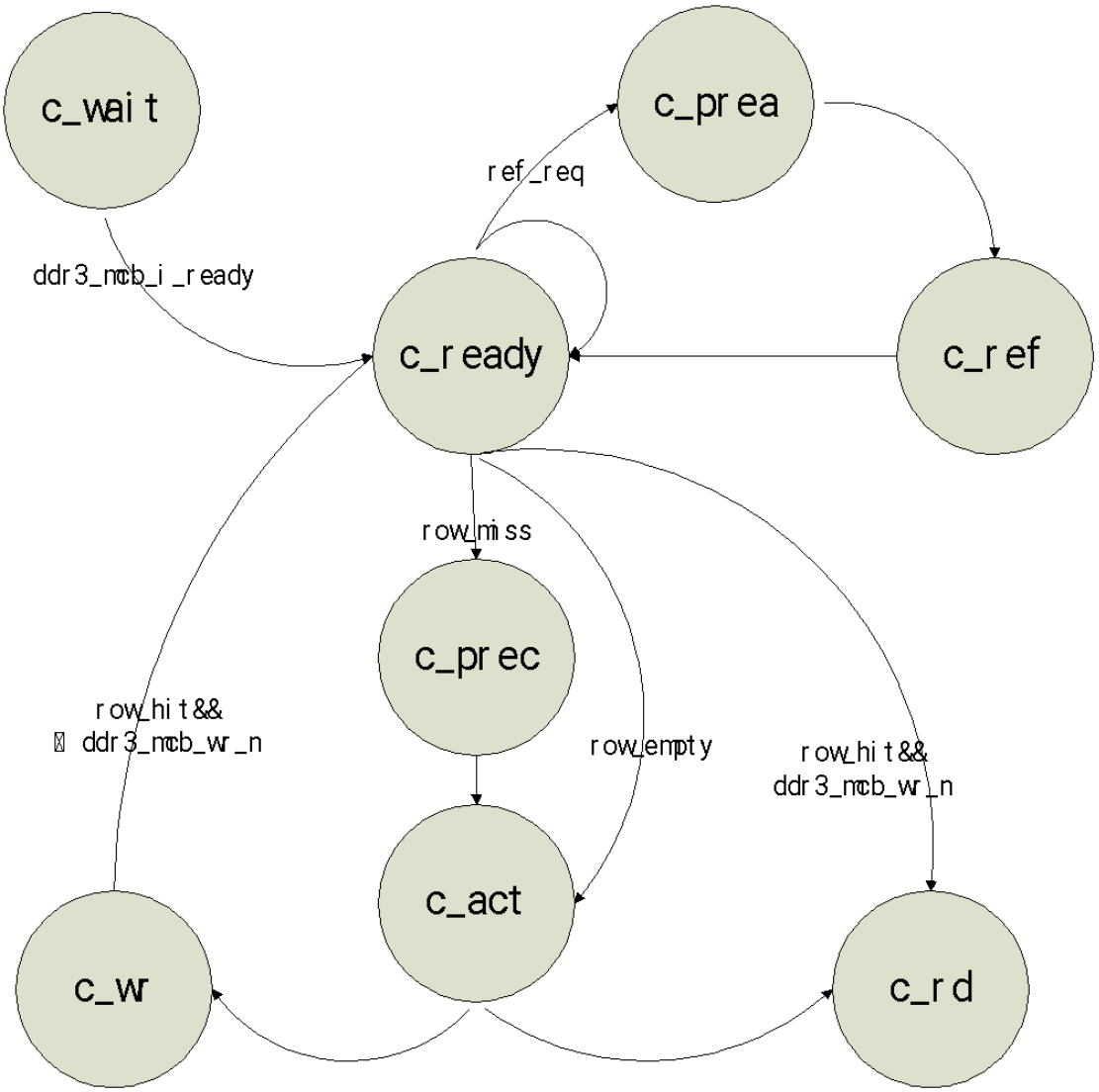


图4.5 基于open page policy策略控制器后端状态机

第五章 基于AXI总线的内存控制器前端设计

在前面的后端设计中，面向系统端是一个比较简单的接口，不符合任何一种通用接口协议要求，同时后端设计中的时钟频率与DDR3 SDRAM的时钟频率保持一致，与系统端频率存在一定的差别，这就意味着如果要将DDR3 SDRAM控制器真正集成到应用系统中，还需要一个前端模块完成接口的转换、时钟的转换以及数据传输等工作。

本章重点介绍内存控制器前端设计，主要分为两部分：第一部分简要介绍内存控制器前端的功能，第二部分简要介绍本次设计的基于双端口SRAM的一个内存控制器前端硬件。

1、内存控制器前端介绍：

从整个实验室系统的角度来看DDR3 SDRAM存储器控制器，可以大致将控制器分为三个部分。H.264 encoder通过AXI总线向存储器控制器发出各种读写指令，控制器前端完成总线的通信、内存访问请求的管理以及时钟域的转换，并控制存储器后端的操作，存储器后端根据接收数据的地址向DDR3 SDRAM发送控制命令并按存储器时序要求收发地址及数据，DDR3 PHY是一个调整时序的模块，它依据所处的环境调整命令地址及数据的时序并发送给片外DDR3 SDRAM器件。由于DDR3物理层设计涉及模拟设计，故本次毕业设计并未涉及，存储器控制器后端设计已经在前面第四章介绍过了，而本章介绍的就是控制器前端设计。

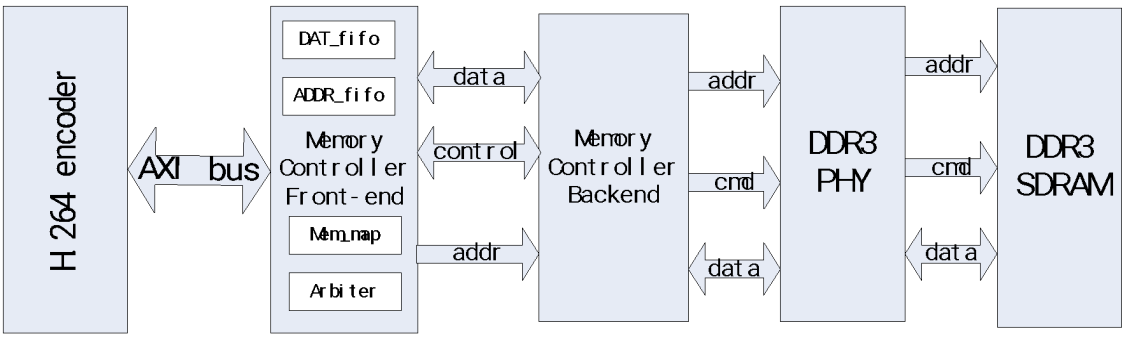


图5.1 控制器系统应用框图

控制器前端实际上是相对于控制器后端提出的一个概念，一般而言控制器后端是通用的，在很多场合由于接口不匹配等原因还需要加上控制器前端来将控制器后端集成到大的系统中。在真正完成的成熟设计中经常会把前端和后端结合在一起，这样可以大大减少冗余逻辑并提高性能。

一个控制器前端设计通常包括数据FIFO、地址FIFO、Mem_map、判决器以及前后两个接口转换模块。其中数据FIFO和地址FIFO主要完成时钟域的转换，接口转换模块主要负责与前后端的数据通信，判决器主要负责解决各种数据冲突，并且尝试着通过各种方式挖掘局部性提高数据传输速率，Mem_map负责合理分配数据写入存储器的地址，系统仅仅发送给控制器一个虚拟地址，控制器根据虚拟地址从提高读写效率的角度生成一个新的地址，并将数据写入这个新的

地址中。Mem_map一方面减轻了系统端的压力，另一方面也有利于尽最大可能发掘局部性提高数据传输速率。

在本次设计中，并没有采用数据FIFO转换时钟域的方式，而是选择将数据写入双端口SRAM中，这样的设计有以下特点：

1、双端口SRAM两端输入频率可以不一致，通过双端口SRAM可以顺利完成时钟域的转换。

2、读写双端口SRAM需要地址，这也就意味着设计中不需要引入地址FIFO，直接使用SRAM的地址即可。

3、双端口SRAM可以作为DDR3 SDRAM的一个高速缓存，正常情况下系统只需要读写双端口SRAM就可以顺利完成数据通信，这将大大降低系统的读写时间开销。

4、每一笔数据不直接与DDR3 SDRAM打交道，而是在更新数据时大批写入DDR3 SDRAM，双端口SRAM实际上部分承担了判决器的功能。

5、引入双端口SRAM可以大大减轻存储器控制器后端的压力，只有在双端口SRAM需要更新数据的时候才会访问DDR3 SDRAM，且这种数据读写非常规律，一个非常简单的存储器控制器后端就可以高效率的完成这样的数据读写，同时根据数据读写的特征可以有针对性的设计控制器后端。

本次设计同时引入AXI接口模块及MCB接口模块分别负责对前后端的接口转换。控制器前端也并没有对总线发送的地址进行过多地改变，仅仅只是出于提高局部性的考虑划分了Bank地址、Row地址及Column地址对应的地址位。AXI总线地址一共有32位，总线给Slave发送地址时并未规定哪几位对应Row地址，所以我们可以灵活选择Row地址以尽最大可能挖掘数据的局部性，提高存储器的存取效率。

2、存储器控制器前端设计：

本次毕业设计中我引入了一个双端口SRAM来构建存储器处理器前端，其模块结构如图5.2所示，本次设计的存储器控制器前端共分为三部分，第一部分是AXI接口模块，它与AXI总线工作在同样的频率上，负责完成与AXI总线的通信并判断是否需要更新双端口SRAM；第二部分为双端口SRAM，它的两端工作在不同的时钟频率上，通过它可以完成时钟域的转换；第三部分为存储器控制器后端接口模块，它与控制器后端工作在同一个频率上，只在更新双端口SRAM数据时启动。

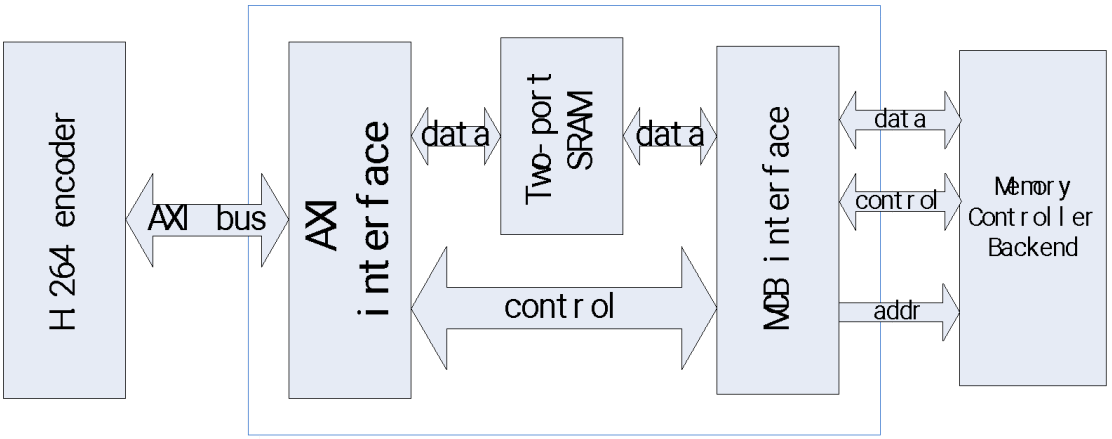


图5.2 控制器前端模块结构图

2.1 AXI interface模块:

这一模块主要就是若干组合逻辑和D触发器的组合，它完成与AXI的基本数据指令通信，在整个编码系统中，DDR3 SDRAM片外存储器是作为系统的一个Slave口挂在AXI总线上的，同时在AXI总线与DDR3 SDRAM控制器之间还存在一个gs（general slave）模块用于简化AXI接口时序，AXI总线模块需要符合的时序要求如下图所示[26]:

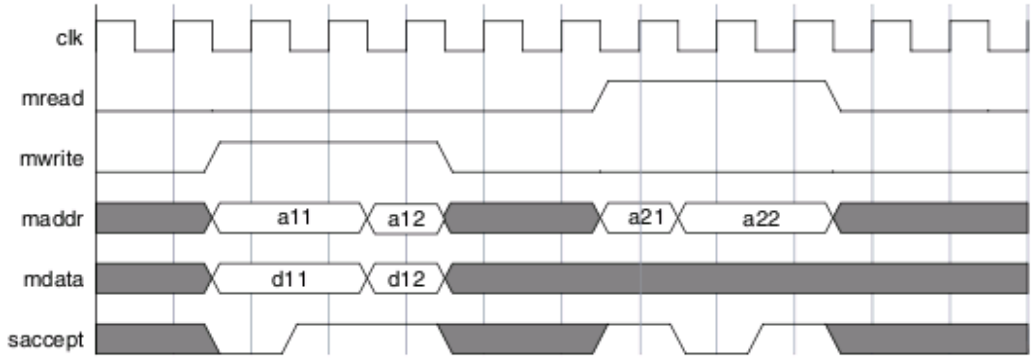


图5.3 AXI指令发送时序

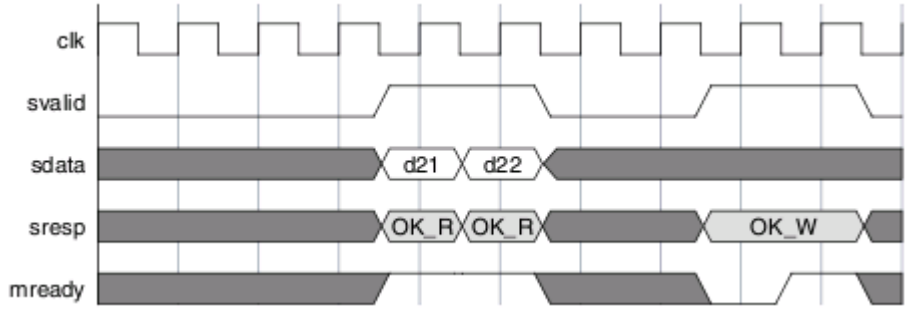


图5.4 AXI返回接收时序

图5.3显示AXI接口模块接收指令时Slave需要满足的时序要求，当AXI发出读写命令（mread或mwrite）时会同时发送地址和数据（需要写入数据时），如果此时saccept信号为低，则总线等待，若为高则总线认为指令已被接收，这是可以发送下一个指令或者执行其他操作。

图5.4显示Slave完成读写指令后需要给AXI的返回信号时序，当总线空闲信号mready为高时，拉高svalid，sresp发出读完成信号并送出数据，则完成读操作，同理，mready为高时拉高svalid且sresp发出写完成信号则代表写操作完成。当mready信号为低时，Slave发出的信号不会被总线接收到，此时slave需要保持原有信号等待总线接收。

除了完成与AXI接口的通信根据AXI发送的地址判断其Bank地址及Row地址是否与当前地址相符（row_now），若相符则直接读写双端口SRAM，若不相符则向MCB interface发送更新双端口SRAM的指令（mcb_begin），并在收到更新完成的信号（refresh_finish）后继续读写双端口SRAM。需要注意的是mcb_begin和refresh_finish信号需要工作在两个不同的时钟域之下，在本次设计中AXI总线端工作频率明显低于控制器后端，这样我只需要控制mcb_begin的宽度为一个AXI总线时钟周期就可以保证MCB接口模块肯定能够收到，而MCB接口模块送来的refresh_finish信号需要AXI接口模块收到后返回一个清除信号（refresh_clear）才能清除。

2.2 双端口SRAM：

这一模块借用了实验室的一个双端口SRAM模型，根据AXI数据位宽（256位）及DDR3 SDRAM位宽（16位，时钟上下沿均可传输数据）将SRAM的数据位宽定为64位，同时根据DDR3 SDRAM的列地址位宽，将SRAM的地址位宽定为10位，也就是说本次设计中使用的是一个64kbit大小的双端口SRAM。由于AXI总线数据位宽是双端口SRAM数据宽度的四倍，所以至少需要四个周期才能完成一笔AXI总线数据读写。

双端口SRAM由于其两端的时钟频率可以不一致，故可以用作时钟域的转换，完成从AXI总线时钟域（一般工作在100MHz左右）向DDR3 SDRAM时钟域（数据传输速率的一半，一般在833MHz左右）的转换。

2.3 MCB interface模块：

MCB（memory controller bankend，存储器控制器后端）接口模块是双端口SRAM与控制器后端之间数据通信的桥梁，它主要负责在cache未命中时完成SRAM的数据刷新工作。

当接收到AXI接口模块传来的启动信号（mcb_begin）后，MCB借口模块启动存储器控制器后端并将SRAM中所有数据读出并写入DDR3 SDRAM中，紧接着将所需Bank及Row地址（row_new）的全部数据导入SRAM中，并发出更新完成信号（refresh_finish）提示AXI接口模块可以开始读取SRAM中数据了，完成SRAM数据刷新功能。

3、控制器前端设计小结：

本次毕业设计完成了一个满足AXI总线时序、基于双端口SRAM的控制器前端设计，具体设计方案已在前一节中给出，这里从对控制器后端的要求、对数据局部性要求及双端口SRAM大小设定三个方面对本次完成的控制器前段进行

总结和讨论。

3.1 对控制器后端的要求：

本次设计的控制器前端对控制器后端要求极为简单，需要控制器后端完成的操作仅仅只是在数据刷新时将双端口SRAM的全部数据直接写入DDR3 SDRAM中的同一行，接着将DDR3 SDRAM中一行的数据全部读出，这也就意味着只需要连续完成大量的同一行数据的读写就可以了，数据读写方式非常固定，在读写时可以很长时间保持行缓冲打开，直到整个一行的读写完成。

这样的要求前面给出的两个存储器后端设计方案都可以轻松满足，但出于时序及芯片面积的考虑，可以将基于close page policy的方案稍作修改，支持连续读写并去掉大量的冗余逻辑，完成一个可连续读写的存储器后端设计方案。这样完成的存储器后端方案本质上基于close page policy策略，正常情况下不开启行缓冲，在收到开始读写命令后打开行缓冲，并从列地址为0处开始连续读写存储器，由于设定的BL为8，所以中间在不关闭行缓冲的基础上还需要不断的发送读写命令，直到将整个一行数据全部读写完毕并关闭行缓冲，从而完成数据读写。

这样的存储器后端完成一行数据读(64kbit)大致需要1035个周期(读写不一致)，其中绝大部分时间都在进行数据传输(1024周期)，应该说尽最大可能应用了DDR3 SDRAM的理论带宽。

一个设计简单的存储器后端另一个重要的优点是比较容易满足频率要求。存储器后端与DDR3 SDRAM工作在同一频率上，一般来说都在833MHz左右，一个设计简单的存储器后端在后端设计时会更容易满足这一时钟频率。

3.2 对数据局部性的要求：

本次设计的控制器前端适用于数据局部性非常好的场合。

使用双端口SRAM作为cache在命中时确实可以明显提高读写数据的速度，但在未命中时需要付出非常大的时序上的代价。正如前一小节分析的，完成整个一行数据(64kbit)的读写大致需要1035个DDR3 SDRAM数据传输周期(一般为833MHz)，假定AXI总线工作在100MHz，这也就意味着需要125个AXI总线周期才能完成一次数据刷新，这样的时序代价显然是非常大的，这也就意味着只有在数据局部性非常明显、能够长时间保持cache命中的情况下这样的方案才是合适的，视频编解码系统确实正是数据局部性非常好的一种系统。

同时，如果能够设计更好的Mem_map完成数据地址的合理重排也可以尽可能挖掘数据的局部性，这也是进一步优化的一个重要方向。

3.3 双端口SRAM大小设定

在初步完成的设计方案中，SRAM的大小被设定为与DDR3 SDRAM行缓冲一致，为64kbit，这样的安排主要考虑设计方便，在读写效率上未必是最优选择，在未来优化性能是可能会考虑增大或者减小cache大小。

如果增大cache的大小，那么考虑数据更新时更新一行比较方便，在设计时会采用部分更新cache的方案，也就是用所需的一行数据替代最长时间没有使

用的一行数据(具体的替代方案可以进一步选择)。同时，由于视频编解码系统数据读写非常有规律，可以在并没有数据更新需求时就根据某些逻辑完成部分行的数据更新，这样的更新可以作为后台进程与前台的数据读写同时进行，进而提高数据传输速率。

如果减小cache的大小，那么虽然cache命中的概率会有所减小，但完成数据更新时需要付出的时序代价也会明显下降，这可能也会提高整体的数据传输效率。

第六章 DDR3 SDRAM设计内存控制器的仿真与综合

本章重点介绍本次内存控制器硬件的RTL仿真结果，由于基于open page policy的存储器控制器后端设计可以说是基于close page policy版本的加强版，故在本章中只介绍基于open page policy的仿真结果。

本章主要分为两部分，第一部分详细介绍基于open page policy后端设计的RTL仿真结果，第二部分介绍放在实验室H.264视频硬件编解码系统中进行仿真的内存控制器前端仿真结果。

1、后端设计的RTL仿真结果：

由于没有合适的物理层verilog模型，故后端设计的仿真直接通过编写testbench文件完成。本次仿真使用ModelSim仿真，用debussy工具观察波形。

1.1 初始化仿真结果：

当硬件上电之后，将瞬间收到开始初始化（init_begin）信号，这时初始化进程开始。得到仿真波形如下图所示：

图6.1 初始化仿真波形图

在收到初始化命令后，复位信号（ddr3_rst）置低至少200us，而后在复位信号消失前10ns置低时钟使能信号（ddr3_cke）并保持至少500us，接下来才可以进行其他配置模式寄存器等操作。

图6.2 初始化命令发送仿真波形图

放大初始化最后一段波形，可以看到清晰的六次发送命令，第一次为发送空指令，接下来是四次装载模式寄存器，最后是一次ZQ长校准命令，ZQ校准完成后输出初始化完成信号（ddr3_mcb_i_ready）.

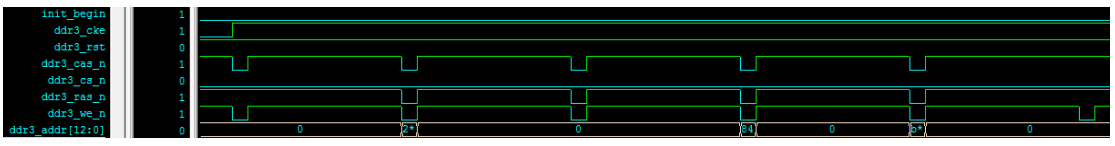


图6.3 初始化配置模式寄存器命令仿真波形图

放大装载模式寄存器部分的波形，可以看到有规律的四次模式寄存器的载入，每次装载模式寄存器之间都有固定的DDR3 SDRAM时序要求的等待时间。

由以上分析可以看出硬件在仿真中可以顺利完成初始化任务。

1.2 写入数据仿真结果：

这里只分析基于open page policy的仿真结果，因为基于close page policy的读写过程就相当于行空闲时基于open page policy硬件的读写过程，唯一的区别仅仅在于前者最后发送的是读写命令都带预充电功能（WRA、RDA），而后者不带（WR、RD），但在时序上没有差别。

当地址判断结果为行空闲时，首先发送ACT指令与Bank及行地址，继而发送WR指令和列地址，即可完成写入操作。

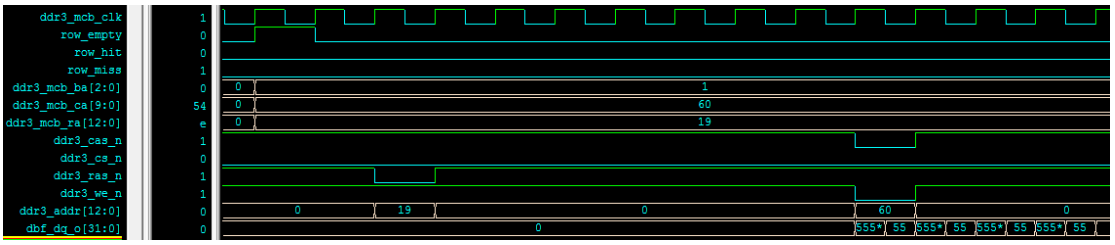


图6.4 行空闲时写入操作仿真波形图

当地址判断结果为行命中时，直接发送WR指令和列地址即可连续4周期八个上下沿写入数据。

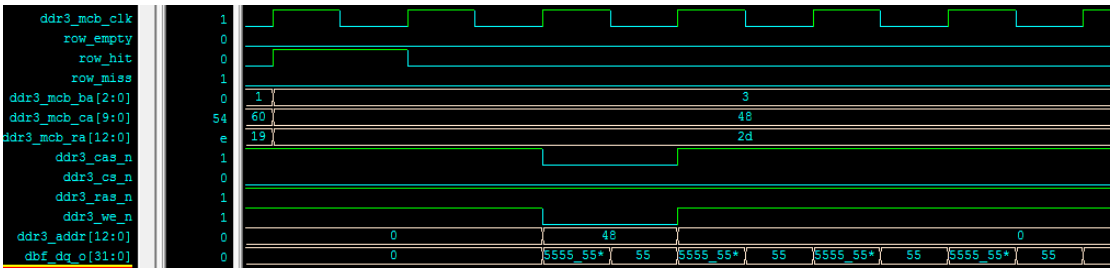


图6.5 行命中时写入操作仿真波形图

当地址判断结果为行未命中是，首先需要发送预充电命令及Bank地址关闭指定Bank的行缓冲，继而发送激活命令ACT及Bank及行地址将指定行放入行缓冲（row_buffer），最终发送WR命令及列地址写入数据。需要注意的是，由于DDR3 SDRAM标准中的时序要求，所有的PRE、ACT、WR时间之间都有一定的时间间隔要求，所以在两个命令之间都必须插入相应的无操作NOP命令。

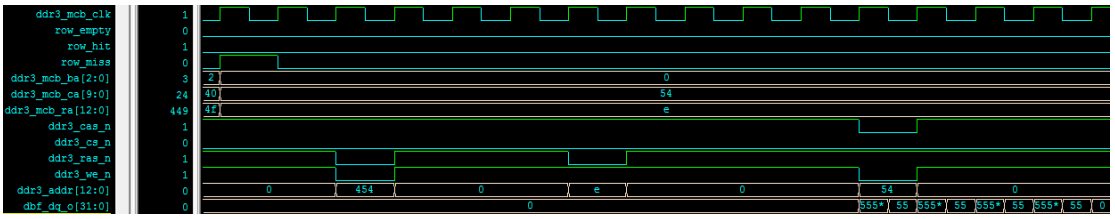


图6.6 行未命中时写入操作仿真波形图

1.3 读取数据仿真结果：

读取过程与写入过程类似，但读取过程需要等待弛豫时间收到dqs信号后才能得到读取的数据。行空闲时首先需要激活所需行，继而发送读指令等到收到dqs信号后便开始在时钟上升下降沿分别读取数据。行命中及未命中时读取过程也与写类似。

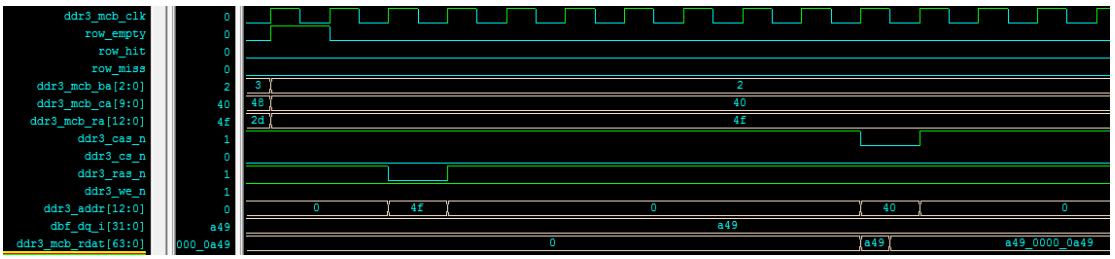


图6.7 行空闲时读取操作仿真波形图

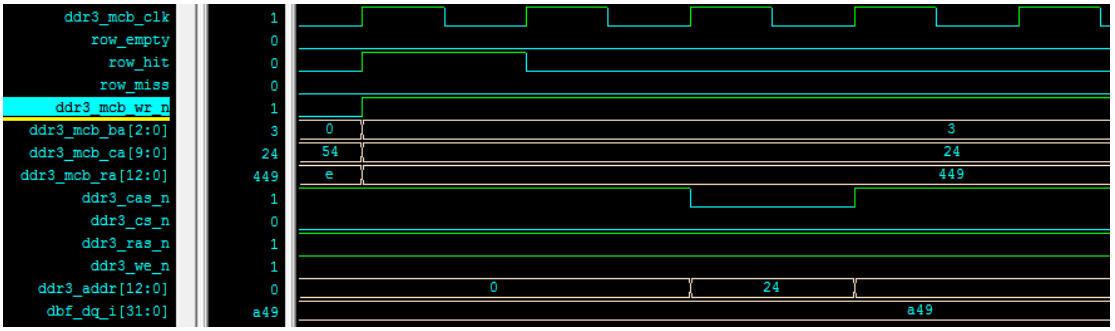


图6.8 行命中时读取操作仿真波形图

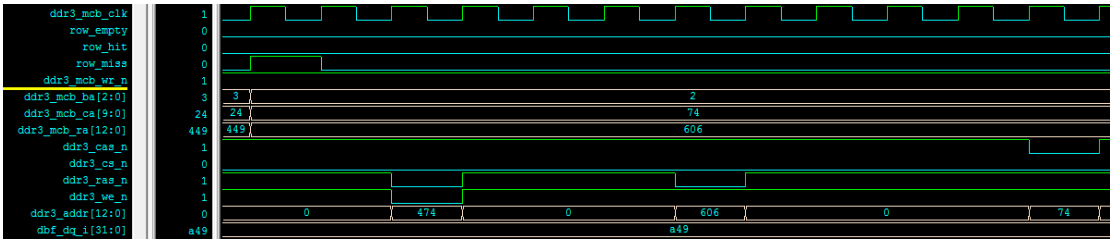


图6.9 行未命中时读取操作仿真波形图

1.4 刷新操作仿真结果：

当刷新计数器计数到固定值后，刷新模块发出刷新警报信号(ref_alert),此时若系统处于空闲状态(c_ready)，则刷新模块紧接着发出刷新请求(ref_req)，控制器继而发出PREA指令，关闭所有Bank的行缓冲，紧接着发出刷新指令(c_ref)，控制DDR3 SDRAM进行刷新操作。

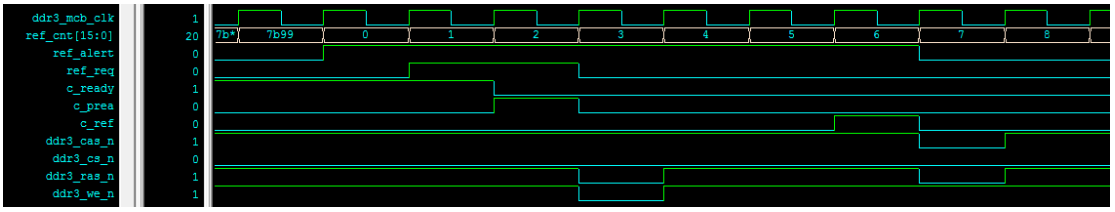


图6.10 刷新操作仿真波形图

2、存储器控制器前端RTL仿真结果：

前端设计由于并不直接与DDR3 SDRAM发生联系，所以前端设计的仿真验证可以使用SDRAM模型完成，这需要前端设计对数据位宽、与后端连接的通信接口及双端口SRAM的大小(SDRAM的row buffer比DDR3 SDRAM小)进行些许改动，但本质上并不影响功能实现。

前端设计仿真通过与实验室H.264视频编解码系统的集成联合仿真来完成，H.264视频编码器通过AXI总线给片外内存发送读写指令，本次设计的前端接收指令并对后端SDRAM进行相应的操作，如果读写出现错误，则会主动报错。前

端设计仿真在cygwin软件环境下进行，使用ModelSim软件进行功能仿真，使用Debussy观察波形并调试代码，在仿真时特别将双端口SRAM两端频率设为不一致，以检验存储器控制器前端能否完成时钟域的转换。经过数小时运行，硬件系统完成10帧视频编码，编码系统与片外内存SDRAM数据通信正常未出错。

需要注意的一点是，实验室AXI总线的数据位宽为256位，使用的双端口SRAM数据位宽为64位，预期的SDRAM和DDR3　SDRAM数据位宽均为16位，这也就意味着我们需要级联多个SDRAM（DDR3 SDRAM）来达到64位的位宽。对SDRAM而言，它只在上升沿传输数据，所以需要4个SDRAM来达到64位的位宽，而对DDR3 SDRAM而言，由于它在时钟的上升沿和下降沿同时传输数据，这样只需要级联两个DDR3 SDRAM就能在一个周期内传输64bit的数据。

下面截取几个系统运行中的代表性片段简要介绍存储器控制器前端的仿真结果。

2.1 双端口SRAM写操作：

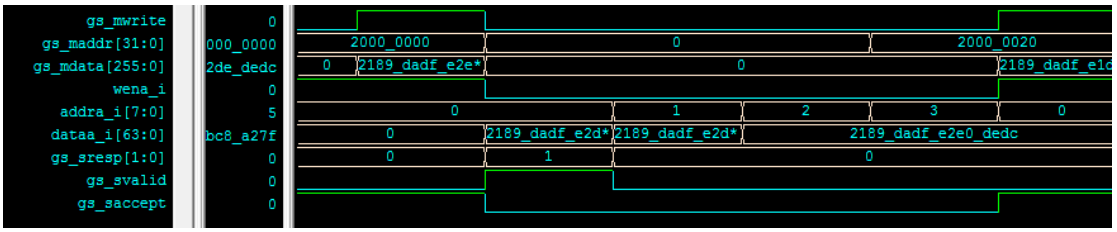


图6.11 双端口SRAM写操作仿真波形图

AXI总线向片外内存Slave发出写命令（gs_mwrite）及地址数据，而后数据复制到buffer中，AXI接口模块发出写入完成信号（gs_svalid、gs_sresp），同时拉低gs_saccept信号表示Slave处于繁忙状态。与此同时，判断Bank及行地址命中后拉低双端口SRAM总线端写使能信号，同时发送地址和数据，将总线发过来的数据写入双端口SRAM中。由于总线数据位宽为256位，而双端口SRAM数据位宽为64位，所以需要四个周期完成数据的写入。写入完成后拉高gs_saccept信号表示Slave处于空闲状态。

2.2 双端口SRAM读操作：

AXI总线向片外内存Slave发送发出读命令（gs_read）及地址，继而AXI接口模块拉低gs_saccept信号。判断Bank及行地址命中后，拉低双端口SRAM总线端的读使能信号开始读取数据，需要注意的是由于最开始地址没有准备好，所以第一笔读出的是地址为0的数据，故必须舍去，继而经过4个周期凑齐256位数据，向AXI总线发出读取完成信号（gs_svalid、gs_sresp）并送出数据（gs_sdata）。完成读操作后拉高gs_saccept准备进行下一次读写。

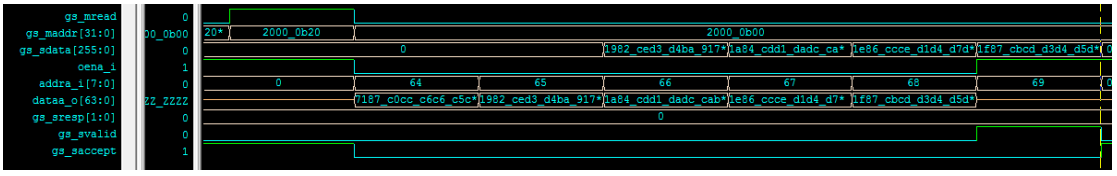


图6.12 双端口SRAM读操作仿真波形图

2.3 双端口SRAM数据更新操作：

当AXI输入的Bank及行地址与原先的地址不一致时，就必须进行SRAM的数据更新，此时AXI接口模块向MCB接口模块发出开始工作信号（mcb_begin），开始进行双端口SRAM的数据更新。

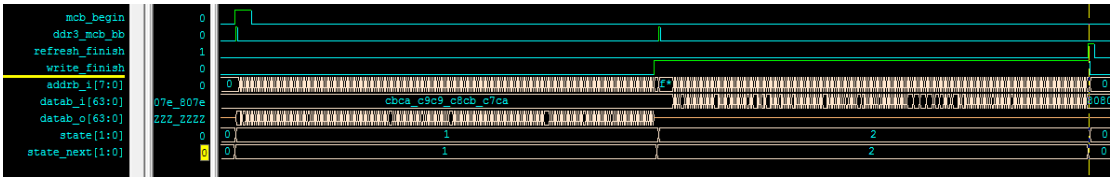


图6.13 双端口SRAM数据更新操作仿真波形图

图6.13是刷新操作全过程的仿真截图。数据刷新分两部分，收到开始工作信号（mcb_begin）后MCB接口模块拉高启动后端信号(ddr3_mcb_bb)，启动存储器控制器后端将双端口SRAM内部的所有数据全部写入外部存储器SDRAM，并发出写入结束信号（write_finish），紧接着state从写状态（01）转换为读状态（10），继而再度启动存储器控制器后端，控制SDRAM完成预充电激活等操作后将所需行的数据读出并全部写入双端口SRAM中，从而完成双端口SRAM的更新操作。

图6.14给出的是收到mcb_begin信号后MCB接口模块操作的仿真结果。收到启动信号后，模块状态（state）从空闲变为写，继而拉高ddr3_mcb_bb并拉低oenb_i开始读取SRAM中数据，当SDRAM控制器后端请求数据信号（ddr3_mcb_wdat_req）拉高之后写入数据信号（ddr3_mcb_wdat）读取SRAM输出信号（datab_i）并将双端口SRAM地址（addrb_i）加一，这样就可以开始持续读取SRAM数据并写入SDRAM中。

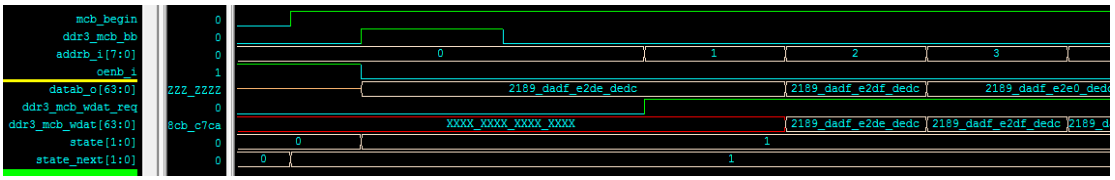


图6.14 MCB接口模块读SRAM写SDRAM操作仿真波形

图6.15显示的是完成第一段数据传输过程前后的仿真波形。当地址信号加到ff之后，双端口SRAM数据被完全导出，继而输出写完成信号write_finish。由于存储器后端此时出现了一次刷新操作，故繁忙标志（ddr3_mcb_busy）延迟了一个周期才拉低，继而state状态从1（写）转变为2（读）并再度拉高启动存储器控制器后端，等到控制器后端拉高数据可读信号（ddr3_mcb_wdat_req）后，读取SDRAM输出数据并按地址逐一写入SRAM中。

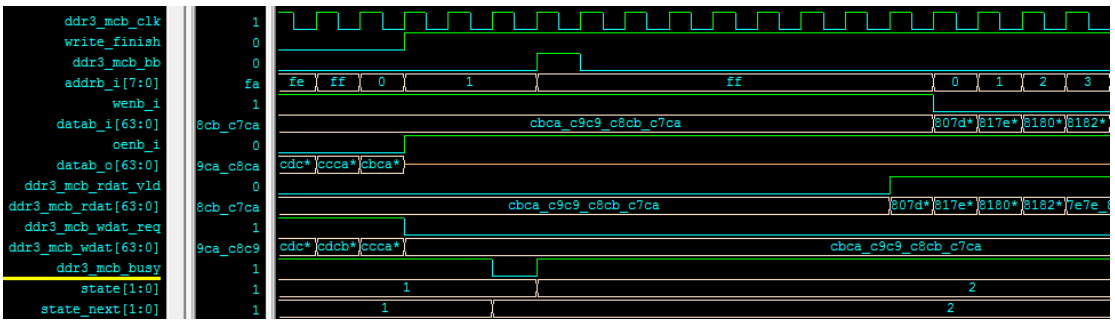


图6.15 MCB接口模块读SDRAM写SRAM操作仿真波形

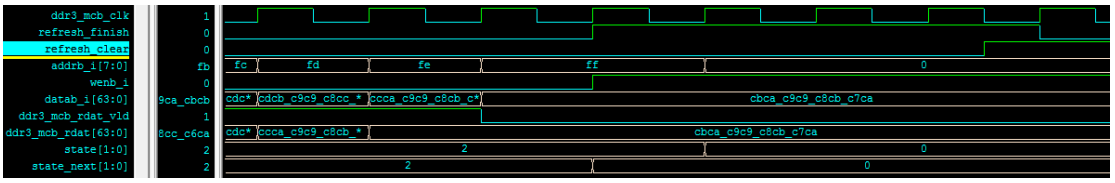


图6.16 MCB接口模块刷新完成操作仿真波形

图6.16显示的是完成刷新操作前后的仿真波形，完成数据写入后，state转为空闲，同时向AXI接口模块发送刷新完成信号（refresh_finish）。等接收到AXI接口模块发送的清除信号（refresh_clear）后清除刷新完成信号，完成整个刷新过程。由于AXI接口模块和MCB接口模块工作在不同的时钟域，且AXI接口模块时钟频率较低，故上图中等待了一段时间才等到清除信号。

第七章 总结及展望

在本次毕业设计中，我概述了DRAM的发展历程并重点介绍了DDR3 SDRAM的特性及基本操作，介绍了Altera公司官方内存控制器IP核及其物理层IP核，给出了我的DDR3 SDRAM存储器控制器设计方案并最终与实验室项目集成。

本次设计最终完成了一个基于AXI总线的DDR3 SDRAM存储器控制器。最终完成的DDR3 SDRAM控制器大致可以分成前端和后端两部分。前端分为三块，第一块与系统工作在相同的频率上，完成AXI总线接口转换并根据地址是否命中决定是读写双端口SRAM还是完成SRAM的数据更新，第二块即为双端口SRAM，作为cache完成数据存取并作为时钟域转换的载体，第三块是面向后端的接口，负责控制存储器控制器后端在刷新SRAM时将双端口SRAM中所有数据写入DDR3 SDRAM中并把DDR3 SDRAM中所需行的数据读出传送到SRAM中。后端则作为DDR3 SDRAM的命令产生模块依照时序要求完成对内存的控制。

在本次设计中，由于DDR3 SDRAM物理层模型的缺失，存储器控制器后端验证不够充分，前端设计也只是通过实验室原先的SDRAM模型完成验证，在设计验证方面不够完备，这也是后面需要继续进行的工作。

本次设计完成的DDR3 SDRAM的内存控制器由于内置一个16kbit的双端口SRAM作为cache，故对应用中读写请求的时间空间局部性要求较高。在实际运行中清空并重新装满SRAM的时间开销会相当的大，因此，只有在时间空间局部性比较好的情况下这样的方案才会更为适合，而实验室视频编解码系统恰好满足这样的条件。

在接下来的实验室工作中，我可以从以下几个方面进一步完善整个DDR3 SDRAM存储器控制器：

1、进一步完成DDR3 SDRAM控制器的设计验证工作，完成与带物理层的DDR3 SDRAM模型的仿真。由于实验室没有带物理层的DDR3 SDRAM模型，所以本次毕业设计的验证工作相当不完备，这也是本次毕业设计最为遗憾的地方。

2、进一步完善整个控制器设计。本次设计依旧有很多可以完善的地方，比如后端中可以加上提高稳定性的ECC模块、后端可以添加让DDR3 SDRAM进入休眠状态的控制逻辑、前端可以设计成异步FIFO的模式完成时钟域的转换等，同时从整个系统的角度而言，将整个视频编解码系统的memory map放在前端中以便按照控制器的习惯分配地址能够进一步提高存储器的使用效率。上面所有这些都是可以进一步完善的地方。

3、进一步完成控制器前端和控制器后端的整合并提高整体的性能。在本次设计中控制器前端与控制器后端是先后设计的，两者之间虽然最终强行捏合在了一起，可以共同写作完成数据读写任务，但其中（特别是在后端）依旧存在一些冗余逻辑，比如AXI接口发送给前端的命令是读写分开的，而由于后端的接口读写是同一根信号线，故前端将分开的读写信号整合到了一起，但到了后端依旧还必须把这个信号分开处理，这里面存在明显的不必要的逻辑。

4、进一步改进DDR3 SDRAM的访问优化算法，并做到硬件上的低成本实现。在本次毕业设计中过于纠结于行缓冲策略的选择，而忘记了所有的所谓策略最终需要达到的目的都是提高最终的存取效率。除了选择合适的行缓冲策略，利用Bank交错或者指令重排的方式也可以提高存储器的存取效率。所谓Bank交错就是在某一Bank进行预充电激活等操作的间隙，可以插入其他Bank的读写命令，这样的间插操作可以明显提高存取效率。所谓指令重排就是根据系统读写请求的地址对指令进行重新排序以提高整体的存取效率，比如存在多个写请求时（一般而言读请求具有最高的优先级），优先相应可以行命中的写请求，这样可以提高整体的存取效率。

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、与实验室项目的整合，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，我的毕业设计能够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，我要感谢一直帮助我的实验室师兄弟们，特别是袁兴师兄和梁晨师兄，在与师兄弟的接触中，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，并被师兄弟们的好学与勤奋深深感染。

然后，我还要感谢自己的父母和祖父母，他们养育我长大，不管潮起潮落一如既往地在背后支持我。

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。

参考文献

[1] Bruce Jacob, Spencer Ng, David Wang. Memory Systems: Cache, DRAM, Disk. Morgan Kaufmann, 2007: Ch12 Evolutionary Developments of DRAM Device Architecture.
[2] Integrated Circuit Engineering Corporation. Memory 1997. 1997: Ch07 DRAM Technology
[3] 梁晨 SDRAM内存控制器研究，复旦大学学士学位论文 2012
[4] Micron. TN4605 General DDR SDRAM Functionality . 2001
[5] 百度百科. http://baike.baidu.com/.
[6] Elpida. E0678E10 DDR2 SDRAM Technology . 2005
[7] Elpida. E1503E10 New Features of DDR3 SDRAM . 2009
[8] JEDEC. JEDEC standard JESD79-3E . 2010
[9] Micron. TN4102 DDR3 ZQ Calibration . 2008
[10] Micron. TN4104 DDR3 Dynamic On-Die Termination .2008
[11] http://blog.csdn.net/shanghaiqianlun/article/details/6976804
[12] Altera. Challenges in Implementing DDR3 Memory Interface on PCB Systems—— A Methodology for Interfacing DDR3 SDRAM DIMM to an FPGA. 2008
[13] Altera. http://www.altera.com.cn/
[14] Altera.External Memory Interface Handbook. 2012
[15] Altera Introduction to ALTMEMPHY IP. 2012
[16] Altera UniPHY Functional Description. 2012
[17] Altera. MegaCore IP Library Release Notes and Errata. 2012
[18] Altera.DDR3 SDRAM High-Performance Controller User Guide.2012
[19] DENALI SOFTWARE, INC.DDR PHY Interface (DFI) Specification Version 2.1 2009.1
[20]万佚. 高性能DDR3存储控制器的研究与实现:(硕士学位论文). 北京:国防科技大学, 2008.
[21] Micron. http://www.micron.com/
[22] Gyounghwan Hyun, Yongseok Jin, Jinsu Jung, Seongyoon Kim, and Hyuk-Jae Lee .A Synchronous DRAM Controller for an H.264/AVC Encoder.2008 International SOC Design Conference.
[23] 王正宇.DDR3内存控制器的IP核设计及FPGA验证.兰州交通大学硕士学位论文,2012
[24] 黄云翔.DDR3 SDRAM控制器的设计及验证。华南理工大学硕士学位论文, 2012.
[25] 邓丽. 高带宽低延迟的DDR2存储器控制器的研究与实现. 国防科技大学硕士学位论文, 2006. 11
[26] Synopsys. General Slave Data Book.2012