# 独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含获得 兰州交通大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：王正宇 签字日期：2012年 6月 7 日

# 学位论文版权使用授权书

本学位论文作者完全了解 兰州交通大学 有关保留、使用学位论文的规定。特授权 兰州交通大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：王正宇 导师签名：杜丽霞

签字日期：2012年 6月 7 日 签字日期：2012年 6月 7 日

硕士学位论文

# DDR3 内存控制器的 IP 核设计及 FPGA 验证

# Design of Memory Controller IP Core and Its Validation on FPGA

作者姓名：王正宇
学科、专业：电路与系统
学　　号：0209739
指导教师：杜丽霞
完成日期：2012 年 4 月

兰州交通大学
Lanzhou Jiaotong University

# 摘　要

作为计算机系统的重要组成部件，内存性能的好坏直接影响计算机系统。由于处理器的访问请求不能被内存直接识别，因此，需要内存控制器来负责完成处理器对内存的控制操作，而内存控制器决定了计算机系统所能使用的最大内存容量、存储体数目、内存类型和速度、内存颗粒的数据深度和数据宽度等重要参数。因此，内存控制器便成为影响内存性能发挥乃至计算机系统整体性能提升的关键因素之一。内存控制器的研究也成为高性能计算、嵌入式系统等领域的研究热点之一。

论文在研究 DDR3 SDRAM JEDEC 规范 JESD79-3E 的基础上，首先对 DDR3 的读写机制和关键技术进行了分析，为控制器的设计提供了理论支撑，然后结合 Altera 公司的外部存储器解决方案，并考虑嵌入式系统的特点，对控制器的设计方案进行了论证，设计出了 DDR3 内存控制器 IP 核的整体架构，接着，采用自顶向下的模块化设计思路，将内存控制器划分为 10 个子模块，并使用 VHDL 语言对各个模块进行编程实现。

在完成控制器 IP 核的设计后，首先使用 VHDL 语言编写了测试平台（Test Bench），在 Quartus 10.0 SP1 和 Modelsim 软件中对内存控制器 IP 核进行软件仿真，接着，论文还给出用户接口模块、初始化模块、指令仲裁模块等关键子模块的 RTL 级仿真结果，并对仿真结果分别进行了分析。最后，在 Altera Stratix IV E 开发板上对控制器 IP 核进行了 FPGA 验证。

本论文所设计的 DDR3 内存控制器 IP 核具有以下特点：

(1) 支持 Unbuffer ECC or Non-ECC 的全系列内存模组，最高可支持容量为 8GB 的内存模组。

(2) 具有内存模组自动识别功能，无需进行参数配置等特点，节省了用户在使用控制器 IP 核时需要配置参数的工作。

(3) 支持最高工作频率为 800MHz，内存带宽为 8500Mbps。

(4) 支持 Altera ALTMEMPHY 数字接口。

(5) 该 IP 核适用于嵌入式系统，具有灵活性好、可移植性强等特点。

**关键词：DDR3；IP 核；内存控制器；FPGA；VHDL**

**论文类型：应用研究**

# Abstract

As a part of the computer system, memory performance directly affects the computer system performance. Because memory cannot recognize the command sent from CPU, memory controller is in charge of processing these commands. It determines some key parameters which are the computer system supporting such as maximum capacity, the number of banks, memory type and speed, the depth and width of SDRAM and so on. Therefore, memory controller determines memory performance or even the overall system performance. Focusing on memory controller research becomes one of the hot points in some fields such as High Performance Computing (HPC), embedded system and so on.

Based on researching on the DDR3 SDRAM JEDEC standard JESD79-3E, the thesis introduces the key technology and the read/write principle of DDR3 SDRAM at first. Then taking the external memory interface solution of Altera Corporation for reference and taking the features of embedded system into consideration, argument on the controller design plan, after that, the top architecture of the memory controller is designed. Continuing, with the top-down and modular design methodology, the DDR3 memory controller is spitted into 10 sub-modules and realized by programming with VHDL language.

After finishing the design of memory controller IP core, a test bench is designed for validation at first. Software simulation is executed in the Quartus II 10.0 SP1 and Modelsim-Altera 6.6c start edition. In addition, the thesis gives the RTL simulation results and result analysis of some key modules such as user interface module, initiation module and execute module and so on. After software simulation, a debugging experiment is executed on the development kit which is equipped with Altera Stratix IV family FPGA.

The memory controller IP core which is designed in the thesis has the following features:

(1) It supports the overall series of Unbuffered ECC or Non-ECC memory modules (UDIMM), which capacity can up to 8 Gigabytes.

(2) It can recognize the module automatically and free to setting parameters, which can save much time and jobs for users.

(3) It supports the peak speed of 800MHz and the peak bandwidth of 8500Mbps.

(4) It supports Altera ALTMEMPHY interface.

(5) The IP core can apply to embedded system. It is smart and easy to transplant.

**Key Words: DDR3; IP Core; Memory Controller; FPGA; VHDL**

# 目　录

# 1 绪论

## 1.1 课题研究背景

作为计算机系统中的重要组件，内存性能的好坏直接影响着计算机系统的性能。随着微电子技术的飞速发展，处理器的性能在成倍的提高，主频和 I/O 带宽都很高，这就需要内存提供很高的数据传输率来配合，这是由计算机组成原理决定的[1-2]。CPU 在运算时所需要的数据都从内存中获取，如果内存系统无法及时给 CPU 供应数据，CPU 不得不长时间处在一种等待状态，硬件资源闲置，性能自然无法发挥。因此，为了适应这种需求，内存的更新换代是必然的。

由于内存不能直接识别处理器的访问请求，内存控制器负责完成处理器对内存的控制操作，而内存控制器决定了计算机系统所能使用的最大内存容量、存储体数目、内存类型和速度、内存颗粒的数据深度和数据宽度等重要参数，内存控制器便成为影响内存性能发挥乃至计算机系统整体性能提升的关键因素之一，所以内存的更新换代呼唤更高性能的内存控制器的出现。因此，如何设计一款高性能的内存控制器始终都是研究的热点。

### 1.1.1 内存发展概述

内存发展大致经历了 SDR（Single Date Rate Synchronous Dynamic RAM，单倍速率同步动态随机存储器）、DDR（Double Date Rate SDRAM，双倍速率同步动态随机存储器）、DDR2 几个阶段。2007 年 JEDEC（Joint Electronic Devices Engineering Council，联合电子器件与设备委员会）颁布了新一代内存规范 JESD79-3A，即 DDR3 内存规范，数据传输率从 667MHz 开始，最高可支持到 2133MHz 的工作频率[3]。DDR3 与 DDR2 的主要区别在于支持 8bit 的预读取技术、支持更多的逻辑段、支持局部自刷新、更低的功耗和新型的绿色引脚封装技术等[4]。

JEDEC DDR3 内存标准一公布，各大内存颗粒厂商以及内存模组厂商都纷纷响应，推出 DDR3 内存的研发计划。从 08 年第三季度开始，各内存厂商开始提供测试样品，采用 70nm 制程工艺，频率主要是 800MHz、1066 和 1333MHz，相比之前的 DDR2，工作电压由 1.8V 降为 1.5V，容量主要是 1GB、2GB 和 4GB。由于 DDR3 内存刚推出时，价格不具备优势，市场份额较小，主要是面向一些高端用户，但是随着工艺制程技术的改进和高密度内存颗粒良品率的提高，到 2009 年时，DDR3 开始成为主流，到 2010 年时，DDR3 开始全面普及，不仅在价格上有所降低，工作电压也从 1.5V 降至 1.35V，内存容量也进一步增加，三星公司开始推出单条 8GB 的内存。进入到 2011 年，DDR3 的

发展继续给力，以三星为首的各大 DRAM 厂商开始于第三季度推出了高容量的 4Gb 内存颗粒，这也进一步使得内存模组的单条容量达到 32GB 成为可能[5-8]。除此之外，工作频率和内存带宽也进一步提高，各大内存厂商也开始推出频率 1600MHz 的内存产品。综观内存市场的发展来看，未来 DDR3 还会朝着大容量、高工作频率、高总线带宽、低工作电压的方向发展。此外，由于内存容量的不断提升，更好的内存纠错和内存数据保护技术也将是内存发展的一个重要方面。

当然，JEDEC 正在制定的 DDR4 内存标准也将于今年的第一季度出炉，Hynix 公司也在 2011 年的 Intel 信息技术峰会上展示了基于 JEDEC DDR4 标准的 4GB SODIMM(Small outline Dual memory module)，其最高工作频率为 2400MHz。这一切均表明，DDR4 时代即将来临，但是，正如 DDR3 面世之初所经历的，DDR4 也会经历刚开始的市场份额不大的局面。然而，此时的 DDR3 将会进入一个全所未有的鼎盛时期，不仅仅是在传统的 X86 领域，也会更一步的渗透到移动多媒体及嵌入式设备领域（目前这一领域处于统治地位的存储产品是 DDR2）。根据半导体市场调研机构 iSuppli 在 2011 年第四季度的预测来看，DDR3 在 2012 年将会达到 71%的市场占有率，而到 2014 年市场占有率将下滑至 49%，与此同时，DDR4 在 2013 年将会占有 5%的市场份额，而到 2015 年时市场份额将超过 50%[9-12]。

### 1.1.2 内存控制器发展概述

内存控制器大致分为传统型和整合型两种。传统型的计算机系统其内存控制器位于主板芯片组的北桥芯片中，CPU 要和内存进行数据交换必须要经过“CPU-北桥-内存-北桥-CPU”五个步骤，在这种模式下，数据经过多级传输而产生的延迟比较大，从而影响计算机的性能。整合型的内存控制器位于 CPU 内，这样，CPU 和内存交换数据时只需经过“CPU-内存-CPU”三级传输，大大减少了数据延迟[13-15]。将内存控制器整合到 CPU 中已经成为行业规范，而且技术也将越来越成熟。

早在 2005 年，内存控制器就被 AMD 集成在 K8 处理器中，但是 Intel 认为在 CPU 中集成内存控制器为时尚早， 坚持将内存控制器集成在北桥中，通过前端总线访问内存控制器。直到 2008 年 Intel 推出 Nehalem 微架构处理器时，才将内存控制器集成到了 CPU 中，该处理器支持三通道 DDR3-1333 内存[16-19]。截止目前，Intel 推出的 SandyBridge LGA2011 将支持四通道 DDR3 内存，而 AMD 于 2011 年第四季度开始推出的推土机架构也将进一步支持更高频率的内存产品。由于内存控制器主要是由芯片组厂商和 CPU 厂商设计和生产，国内涉足这一领域的厂商比较少，最具代表性的是龙芯和威盛。龙芯在 08 年的 6 月份正式发布搭载龙芯 2F 处理器的国产电脑福珑 2F，该处理器集成了 DDR2 内存控制器。威盛在 2010 年的台北电脑展上也展示了双核 Nano CPU，该 CPU 支持双

通道 DD2-800 或 DDR3-1066 内存。从目前内存控制器的技术发展来看，内存控制器主要是向着支持多通道、更高频率、提供更高总线带宽的方向发展[20-21]。当然，随着内存容量的大幅度增加，内存控制的纠错和数据保护技术也将会像内存一样，成为研究的重点。目前具有代表性的技术是 ECC（Error Checking and Correcting）技术、Chipkill 技术、内存热备以及内存镜像技术等[22]。

当然，PC 机以及服务器的发展早已进入联盟化的发展道路，因为任何技术的发展都离不了业界的支持。随着可编程逻辑器件的发展，越来越多的可编程器件的设计厂商开始涉足 X86 领域。Altera 公司在很早就开始致力于将 FGPA 产品应用到高性能计算领域，例如，Altera 与 Intel 合作开发出 FSB 总线、与 AMD 公司合作开发出 HT 总线等。在外部存储器方面，Altera 也提出一系列的解决方案，这些方案涵盖了从高性能 DDR3 到低功耗 DDR 的每种应用。Altera FPGA 通过外部存储器 IP 来提高存储器性能，它包括 PHY 和控制器。设计人员可以选择 Quartus II 软件所列出的默认存储器解决方案，根据存储器要求选择最佳 PHY 和控制器 IP ，也可以选择定制存储器接口。因此，使用 Altera 公司开发出的 FPGA 产品进行 DDR3 内存控制器的设计不失为一种理想的选择[23-27]。

## 1.2 课题研究思路

### 1.2.1 研究目标及适用范围

本课题旨在通过对 JEDEC DDR3 内存规范和 Altera 外部存储器解决方案及 Stratix 家族器件的研究，设计一款兼容 JEDEC 标准的内存控制器 IP 核，完成与 Altera ALTMEMPHY 的接口设计，并在 Altera Stratix IV FPGA 上进行 IP 核的功能验证。设计的 IP 核需要满足以下性能指标：

(1) 支持 DDR3 SDRAM 的所有新特性；

(2) 支持 ALTMEMPHY 数字接口；

(3) 支持单通道 Single/Dual Rank Unbuffer ECC DDR3 的内存模组；

(4) SDRAM 内核工作频率为 100MHz，I/O 时钟频率为 400MHz，数据传输频率最高达到 800MHz，单通道内存带宽为 8500Mbps。

从以上的设计指标，不难发现，本课题所设计的 DDR3 SDRAM 控制器是一款适用于嵌入式系统的内存控制器。有关本课题设计方案的论证将在本论文的第三章中给出。

### 1.2.2 研究内容

本课题主要设计了一款兼容 JEDEC DDR3 的内存控制器 IP 软核，具体工作内容如下：

(1) DDR3 SDRAM 控制器 IP 核的整体架构设计

在对 DDR3 JEDEC 规范进行深入研究并分析 DDR3 SDRAM 工作原理和指令的基础上，进行 IP 核的系统功能分析，确定系统实现的功能，进行各个子模块的功能划分，进而提出内存控制器的整体架构、系统输入与输出等，然后完成控制器主状态机的设计工作。在整个内存控制器的 IP 核设计中，状态机的设计是整个设计的核心。

(2) DDR3 SDRAM 控制器 IP 核的子模块 RTL 级设计

采用自顶向下的模块化设计思路，完成内存控制器的初始化模块、指令执行模块、定时器模块、ECC 模块的代码设计和仿真工作。其中，指令执行模块是该 IP 核设计的重点和难点。

(3) DDR3 SDRAM 控制器 IP 核的数字 PHY 接口设计

将所设计的 DDR3 SDRAM 控制器与 ALTMEMPHY 进行适配，完成与其数字接口的设计。

(4) DDR3 SDRAM 控制器 IP 核的软件仿真及调试

完成整体架构以及所有的子模块的设计后，首先使用 ModelSim 仿真软件对各个子模块进行功能仿真和时序仿真，然后再对整个 IP 核的系统级输入输出进行功能和时序仿真工作，最后，对照 JEDEC DDR3 标准，对部分电路进行相应的微调，从而使得控制器在工作时能够获得最大的时序裕量。

(5) DDR3 SDRAM 控制器 IP 核的板级验证及调试

借助 Quartus 10.1 和 ModelSim 6.6c 软件，完成了对控制器设计的软件仿真之后，把网表文件通过下载电缆下载到 Altera Straix IV 开发板上，进行 IP 核的板级验证与调试。

## 1.3 课题研究意义

虽然 JEDEC 即将于 2012 年的第一季度正式推出 DDR4 SDRAM 的标准，但 DDR3 的发展还尚未达到顶峰，特别是 DDR3 在嵌入式设备上的应用还远远没有达到顶峰，因此，研究 DDR3 SDRAM 及其控制技术，特别是研究如何更好地将 DDR3 SDRAM 和其控制技术应用到嵌入式系统中，仍然是今后很长一段时间的研究热点和重点。本课题是在分析 DDR3 及其控制技术的发展和应用的基础上展开的，采用硬件描述语言编写控制

器的各个子模块，设计出一款具有广阔应用前景的内存控制器的 IP 软核。完成本课题的设计，具有但不限于以下意义：

(1) 虽然 DDR3 内存问世已经有一段时间，国内外也有很多机构对其研究，但这些研究主要集中在传统的 X86 领域，对于如何将 DDR3 内存及其控制器技术应用到嵌入式系统方面的研究并不是很充分，因此，设计一款兼容 JEDEC 标准而且能够更好地应用于嵌入式系统的内存控制器，将具有非常深远的意义。除此之外，由于 DDR3 本身结构复杂、读写操作过程繁杂，对时序的要求较高，因此，高性能内存控制器的设计具有很高的学习和研究价值。

(2) 软 IP 核具有移植性强、参数可调等特点，可以克服设计内存控制器设计周期长、流片成本高、灵活性差等缺点，同时也方便将内存控制器集成到各种嵌入式系统中去。

(3) 本课题设计的 IP 核具有与 ALTMEMPHY 的数字接口。在应用 Altera 提供的外部存储器解决方案和 Altera 提供的高性能 DDR3 存储器控制器（HPC 和 HPC II）进行内存子系统设计时，难免会出现一些兼容性的问题。因此，该 IP 核可以用来进行 Debug 分析，以帮助设计者尽快定位设计过程中出现的一些 Bug，加速设计进程。

(4) DDR3 IP 软核的设计具有产业化启示意义。DDR3 的应用领域相当广泛，不仅涉及传统的 X86 领域，还涉及到通信、移动手持设备、嵌入式系统、工业控制等许多领域。对于设计相当成熟的 IP 核来说，其产业意义和商业价值不容忽视。

(5) 通过本课题的研究，不仅掌握了 DDR3 内存及其控制器方面的相关技术，同时也掌握了 EDA 设计方面的技术，为今后进一步研究 DDR3 内存和内存控制器设计的某些关键技术打下了坚实的基础，同时也为将 DDR3 内存及其控制器技术应用到嵌入式系统中提供了设计思路。

## 1.4 论文结构

整个论文的主体部分共分为五章，各个章节的内容如下：

第一章为绪论部分，主要介绍课题的研究背景、研究思路及研究意义。其中，在研究背景部分分别介绍 DDR3 及其控制器的发展状况，在研究思路部分首先明确了控制器设计的技术指标和应用领域，然后交代了论文工作的研究方法和主要工作内容。

第二章为 DDR3 SDRAM 的技术分析，主要分析了 DDR3 SDRAM 的一些关键技术、系统结构、工作机制、基本操作的指令以及模式寄存器设置的一些内容，为后面章节的控制器设计提供理论支撑。

第三章为 DDR3 内存控制器 IP 核的整体架构的设计，首先通过分析内存子系统的工作机制以及设计方案的论证，给出了整体架构的设计图和主控制逻辑的状态机设计，

然后对各个子模块所要完成的功能进行了描述，并进行 RTL 级设计，给出各个子模块的结构框图、信号定义等。

第四章为 DDR3 内存控制器 IP 核的软件仿真与验证，首先给出了 IP 核仿真的一般方法，其次给出了各个子模块的仿真方法，最后给出了各个模块的仿真结果并对仿真结果进行了分析。

第五章为 DDR3 内存控制器 IP 核的板级仿真和调试，首先介绍了验证平台的资源信息，然后给出验证方案及测试脚本，最后给出了测试结果。

最后是结论和工作展望，主要给出了该 IP 核设计的一些成果，指出了设计的一些不足之处和需要继续完善的地方。最后，对未来的设计和进一步工作提出了一些设想。

# 2 DDR3 SDRAM 存储器技术分析

为了更好的进行 DDR3 内存控制器的设计，首先要对 DDR3 SDRAM 存储器技术进行详细的分析。本章将从以下四个部分展开对 DDR3 的技术分析。

## 2.1 DDR3 概述

DDR3 SDRAM 是第三代双倍数据速率动态同步随机存储器的简称。我们通常所说的 DDR3 内存模组是采用多颗 DDR3 SDRAM 并根据 JEDEC 的相关内存模组设计标准而制作的[28]。JDEEC 根据内存控制器和内存模组上的 SDRAM 的信号连接方式的不同，将内存模组分成三种类型：

(1) UDIMM（Unbuffer Dual Inline Memory Module，无缓冲型内存模组）上的所有总线信号直接和内存控制器的外部引脚相连接。

(2) RDIMM（Register Dual Inline Memory Module，寄存器型内存模组）的数据总线直接和内存控制器相连接，而地址、控制总线通过一个 Register 芯片和内存控制器相连接。这个 Register 芯片一方面可以起到净化信号的目的，另一方面可以使得内存控制器支持到最高 4 Rank 的内存模组。

(3) LRDIMM（Load Reduced Dual Inline Memory Module，低负载型内存模组）有点类似于 DDR2 时代的 FBDIMM,其上的所有总线信号都通过一个 MB（Memory Buffer，内存缓存）芯片和内存控制器相连。该模组使得内存控制器可以支持到 8 Rank，从而可以获得更大容量。

以上三种内存模组根据是否具备 ECC（Error Checking and Correcting）功能又被划分为 ECC 内存和 Non-ECC 内存两大类。对于后两种内存模组，由于涉及到内存容量比较大，市面上见到的一般都是 ECC 的。只有前一种内存模组，市面上才用 Non-ECC 和 ECC 之分，我们通常所见到的 PC 机 DDR3 内存，一般多为 ECC UDIMM。当然，我们见到的笔记本内存的 DDR3 SODIMM 实际上也属于 ECC UDIMM 范畴的，只不过它的引脚数量比一般的 UDIMM 少，为 204 pin，而 DDR2 时代的 PC 机和笔记本内存均是 Non-ECC 的。由于本课题所设计的 IP 核的应用场合是嵌入式系统，因此只支持第一种内存模组，同时支持 ECC 功能。

## 2.2 DDR3 关键技术介绍

最新的 DDR3 SDRAM JEDEC 标准是 2010 年 7 月颁发的 JESD79-3E。在该规范中，明确规定了 DDR3 SDRAM 内置 8 个 Bank，每个 Bank 包含若干行和若干列。相比 DDR2 的 4 个 bank 设计，将 DDR3 存储器划分成 8 个 Bank，使得制作大容量的 SDRAM 芯片

成为可能。目前业界已经研制成功了 4Gb 容量的 DDR3 SDRAM。除了更大容量这一特点之外，DDR3 还具有以下特点：

(1) 支持异步复位功能

DDR3 新增了一个 Reset 管脚，这一引脚将使 DDR3 的初始化处理变得更为简单。除此之外，当 Reset 命令有效时，DDR3 内存将停止所有操作，并切换至最少量活动状态，以节约电力。

(2) 新增 ZQ 校准功能

ZQ 校准也是 DDR3 新增的一项功能，它通过在新增的 ZQ 管脚上外接一个 240 欧姆的公差电阻来实现这一功能[1-3]。ZQ 这个引脚先通过一个命令集，然后再通过片上校准引擎（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使用 512 个时钟周期，在退出自刷新操作后使用 256 个时钟周期、在其他情况下使用 64 个时钟周期）对导通电阻和 ODT 电阻进行重新校准。

(3) 突发长度（BL, Burst Length）发生变化

从 DDR 开始，内存数据的读写均采用突发读写的方式来完成。早期架构的 DDR 和 DDR2，突发长度为 4bit。而 DDR3 的突发模式分为三种模式：

第一种是 BL=8，这种模式也是 DDR3 最为常见的模式。第二种是 BL=4，这种模式也被称作 4-bit Burst Chop（突发突变）模式，这种模式主要是为了兼容早期 DDR、DDR2 架构的系统而设计的。第三种是 Fly-By 模式，这一模式的 BL 是不固定的，可以通过 A12 地址线的电平高低来确定 BL 的数值。

(4) 根据温度自刷新和局部自刷新功能

由于 SDRAM 是掉电易失性元器件，所以需要定期刷新操作来维持数据。为了达到节能的目的，DDR3 采用了根据温度自刷新技术。实现这一技术具体做法是这样的，在 DRAM 芯片中内置一个温度传感器，该传感器会自动检测 DRAM 芯片的表面温度，当温度在 0℃至 85℃之间时，DRAM 将以 7.8us 的刷新周期进行自刷新操作；当温度在 85℃到 95℃之间时，DRAM 将以 3.9us 的刷新周期进行自刷新操作，而其他不在此范围的工作温度均超出 JEDEC 规范。此外，DDR3 还可以通过只刷新部分 Bank 的方式实现局部自刷新功能。根据温度自刷新和局部自刷新功能能在一定程度上起到节能的目的。但是，由于这个功能是个可选设计，并非市场上所有的 DDR3 SDRAM 存储器都具备这两种功能，又鉴于本课题设计的 IP 核只是应用于民用场合，所以在本设计中将不提供对此功能的支持。

(5) 点对点连接

这是DDR3区别与DDR2的一个关键区别，它为了提高系统性能而进行的技术改进。在DDR3 SDRAM系统中，控制器与存储器是一一对应的，这样可以大大减轻地址、控制和数据信号的总线负载，提高了信号的完整性。针对单Rank的模组，存储器控制器和DDR3 内存模组之间是点对点（P2P, Point to Point）连接关系；针对双Rank的模组，存储器控制器和内存模组是点对双点（P22P, Point to 2 Points）连接关系。

除了在减少控制器所支持的最大DIMM数量外，DDR3 DIMM在布线方式方面也发生了变化。之前的DDR2 DIMM上，为了减少地址、控制信号到达各个DRAM存储器芯片的时间偏差，采用了菊花链式的连接方式，如图2.1（a）所示。这样的连接方式带来了分支线的问题，影响了信号的传输质量。而在DDR3 DIMM上，采用了新型的“Fly-by”布线结构，信号从DIMM的中部进入，之后到达另一端，依次按顺序连接各个DRAM芯片，取消了信号的分支，如图2.1（b）所示。这样有利于提高信号的完整性，但这样的结构也带来了时间偏差的弊端，于是在DDR3中引入另一技术——读写校准（leveling）技术，从而弥补了这一缺陷。我们也在后续章节中讨论写校准技术。

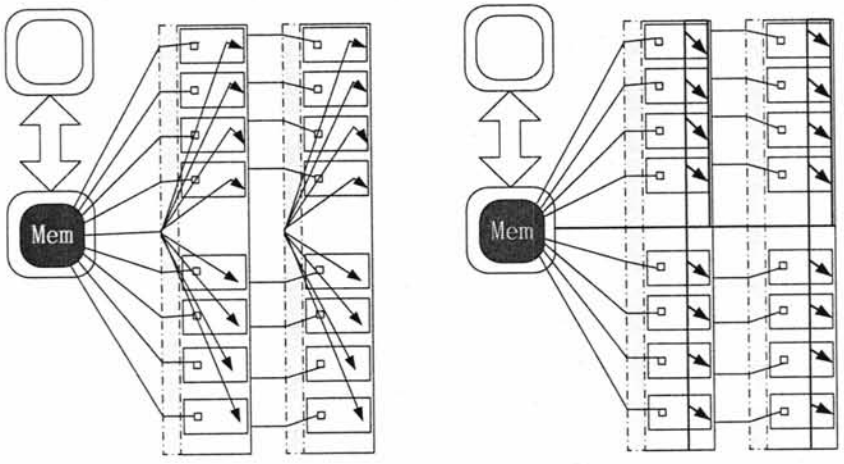


图2.1 DDR2、DDR3的在线拓扑结构

(6) 封装方式（Package）

DDR3 SDRAM采用新型绿色无铅的FBGA封装方式，由于DDR3 SDRAM引入了一些新的功能，因此在引脚方面有所增加。4-bit I/O的SDRAM采用78球（Ball）FBGA封装方式，8-bit I/O的SDRAM采用78球或者82球的两种FBGA封装方式，16-bit I/O的SDRAM采用96球的FBGA封装方式。

(7) 参考电压分成两个

参考电压 $V_{REF}$ 是存储器稳定工作的关键，DDR3 将 SDRAM 工作的参考电压分成两个：一个是为地址和控制总线提供服务的 $V_{REFCA}$，另一个是为数据总线提供服务的 $V_{REFDQ}$。两个不同的参考电压将为 DDR3 存储器提供更好的抗噪能力。

(8) 8-bit 预取架构

提高 SDRAM I/O 的时钟频率有两种方法，一是通过提高 SDRAM 内核的工作频率来实现，二是通过改变 SDRAM 的数据读取的预取架构来实现，前者由于受到工艺制程等方面因素的制约，很难实现。因此，为了获得更高的 I/O 时钟频率，从 DDR 开始，SDRAM 均采用多位数据预取技术，DDR3 采用 8-bit 的预取架构。图 2.2 显示的是从 SDR 到 DDR3 数据预期架构的演变过程。

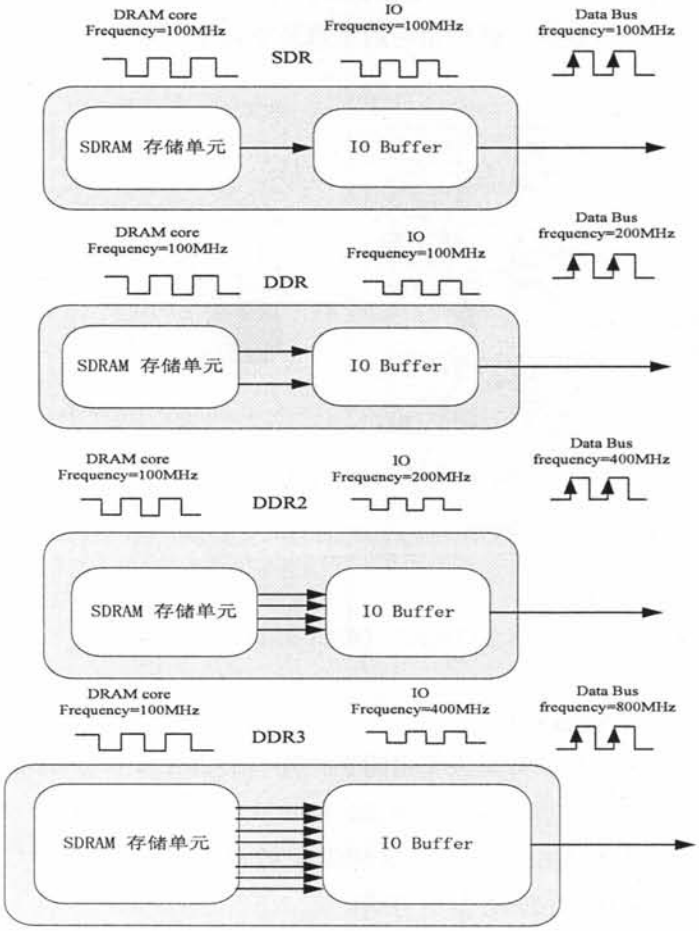


图 2.2 SDR、DDR、DDR2、DDR3 预取设计对比

从图 2.2 中，我们不难发现，DDR3 SDRAM 采用 8 位预取架构，相比 DDR2 的 4 位预取架构，可以获得更高的工作频率。对应同是 100MHz 的内核工作频率，DDR3 可以获得 800MHz 的总线传输频率，大大提高了数据总线的带宽。

从以上总结的 DDR3 诸多优点来看，DDR3 存储器具有明显的技术优势，表 2.1 列出了三代 DDR 技术的相关参数。通过这些技术参数，我们可以对 DDR3 有一个比较全面的了解，这也为后面理解 DDR3 的工作机制及命令奠定了基础。

表 2.1 三代 DDR 技术规格比较

| 技术参数 | DDR | DDR2 | DDR3 |
|---|---|---|---|
| 电压 VDD/VDDQ | 2.5/2.5V(+/-0.125) | 1.8/1.8V(+/- 0.1) | 1.5/1.5V(+/-0.075) |
| I/O 接口 | SSTL_2 | SSTL_18 | SSTL_15 |
| 数据传输率（Mbps） | 200/266/333/400 | 400/533/667/800 | 800/1066/1333/1600/1866/2133 |
| SDRAM 容量标准 | 64Mb-256Mb | 256Mb-4Gb | 512Mb-8Gb |
| CL(CAS 延时) | 1.5/2/2.5 | 3/4/5/6 | 5/6/7/8/9/10/11 |
| AL(附加延时) | 无 | 0/1/2/3/4 | 0/CL-1/CL-2 |
| Bank 数量 | 4 | 4,8 | 8 |
| 预取位数（bit） | 2 | 4 | 8 |
| 突发长度 | 2,4,8 | 4,8 | 4,8 |
| 数据选通时钟 | 单时钟 | 差分时钟 | 差分时钟 |
| 封装方式 | TSOP | FBGA | FBGA |
| 引脚标准 | 184 | 240 | 240 |

## 2.3 DDR3 工作机制

### 2.3.1 DDR3 SDRAM 的工作状态机

根据 JEDEC 规范的规定，DDR3 SDRAM 的工作必须按照一定的规则进行，必须符合相应的状态转换要求[3]。图 2.3 给出了 DDR3 存储器工作的状态机，该状态机描述了 DDR3 各个状态的转换以及转换时所涉及的一些命令。DDR3 内存控制器的设计将以此状态机的设计为重点，确保内存模组上的每个 SDRAM 都按照此状态机进行工作。

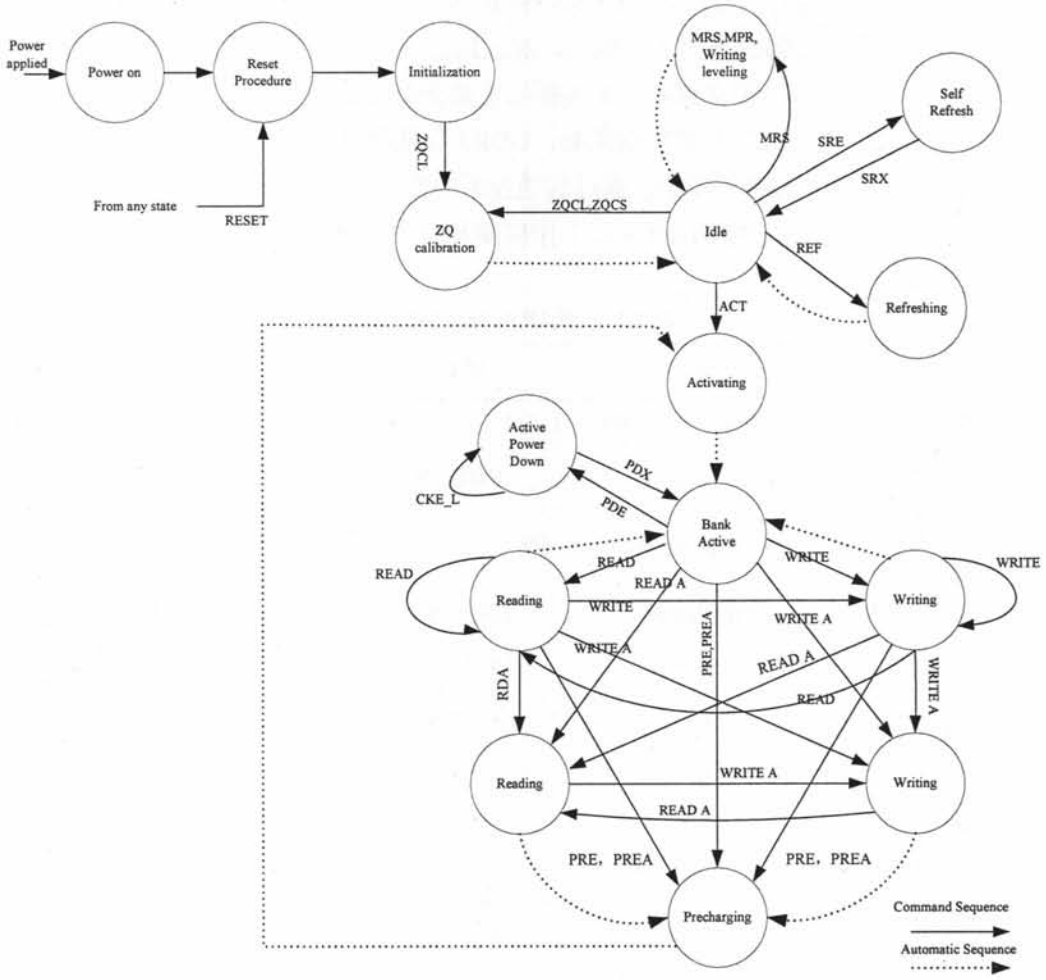


图 2.3 DDR3 SDRAM 工作状态简图

表 2.2 给出了图 2.3 中所涉及到的一些命令和功能描述。

表 2.2 部分 DDR3 命令描述

| 命令缩写 | 功能 | 命令缩写 | 功能 | 命令缩写 | 功能 |
|---|---|---|---|---|---|
| ACT | 激活命令 | READ | 读操作 | PDE | 进入省电模式命令 |
| PRE | 对单一 Bank 预充电 | READA | 带预充电的读操作 | PDX | 退出省电模式 |
| PREA | 对所有 Bank 预充电 | WRITE | 写操作 | SRE | 进入自刷新模式 |
| MRS | 模式寄存器设置 | WRITEA | 带预充电的写操作 | SRX | 退出自刷新模式 |
| REF | 刷新操作 | RESET | 异步复位 | MPR | 多用途寄存器 |

### 2.3.2 DDR3 SDRAM 的上电及初始化过程

DDR3 在被访问前，必须进行初始化操作，完成必要的模式寄存器的配置和 ZQ 校准，然后才能进入空闲的状态，等待控制器的访问[3]。DDR3 完成上电初始化操作必须满足如下时序要求：

(1) 稳定上电后，置复位信号至少 200us，然后在 RESET 信号拉高之前的 10us 时刻或者此时刻之前的任意时刻，拉低 CKE 信号。这里推荐 RESET 信号的电平大小为 $0.3*V_{DD}$，而且上电时电压的上升时间不超过 200ms。同时还要求 $V_{DD}>V_{DDQ}$ 或者 $V_{DD}-V_{DDQ}<0.3$Volts。

(2) 在 RESET 信号失效后，需要继续等待 500us 后再拉高 CKE 信号。在这段时间内，DRAM 芯片将启动内部状态的初始化动作，这一动作是在不需要外部时钟的情况下独立完成的。

(3) 差分时钟对（CK,CK#）需要在 CKE 拉高前，启动并稳定维持至少 10ns 或者 5 个时钟周期。在这期间，至少要发送一个 NOP（空操作）或者 DESEL(反选)命令。

(4) 在复位信号有效的期间，ODT（On Die Termination，片内终结器）信号必须处于高阻态。除此之外，在复位信号失效后到 CKE 信号被拉高的期间，ODT 也必须处于高阻状态。

(5) 执行 MRS 命令，配置模式寄存器 MR2。

(6) 执行 MRS 命令，配置模式寄存器 MR3。

(7) 执行 MRS 命令，配置模式寄存器 MR1，并使能 SDRAM 芯片中的 DLL。

(8) 执行 MRS 命令，配置模式寄存器 MR0，复位 SDRAM 芯片中 DLL。

(9) 执行 ZQCL 命令，进行 ZQ 长校准。

(10) 等待 PLL 进行相位锁定和 ZQ 长校准完成。

完成上述的时序之后，SDRAM 就进入了就绪的状态，等待内存控制器的访问命令。

### 2.3.3 DDR3 模式寄存器配置

为了应用的灵活性、功能的多样性以及获得更多的特性，DDR3 SDRAM 提供了四种模式寄存器供上层用户（这里指内存控制器）通过 MRS 命令进行配置[17-20]。在进行配置之前，模式寄存器的内容是不确定的。必须通过上电和初始化命令或者重装 MRS 命令才能改变模式寄存器的值，让 SDRAM 按照预先设定的方式进行工作，否则会造成不确定的操作出现。

模式寄存器通常用来定义 DDR3 SDRAM 的一些操作模式。不同的模式寄存器定义了不同功能，同时模式寄存器还对突发读写的长度、类型、CAS 大小和操作模式进行了

定义。表 2.3 显示了 DDR3 SDRAM 的四种模式寄存器设置及其所提供的功能，其中，Bank 地址 BA2、BA1、BA0 决定了模式寄存器的选择。

表 2.3 模式寄存器设置及其功能列表

| Mode Register | MR0 | MR1 | MR2 | MR3 |
|---|---|---|---|---|
| BA2 | 0 | 0 | 0 | 0 |
| BA1 | 0 | 0 | 1 | 1 |
| BA0 | 0 | 1 | 0 | 1 |
| A15～A13 | 0 | 0 | 0 | 0 |
| A12 | PRD | Qoff(Output buffer enable/disable) | 0 | 0 |
| A11 | WR(Write Recovery) | TDQS | 0 | 0 |
| A10 | WR(Write Recovery) | 0 | Rtt_WR | 0 |
| A9 | WR(Write Recovery) | Rtt_Nom | Rtt_WR | 0 |
| A8 | DLL | 0 | 0 | 0 |
| A7 | TM(Test mode) | Level | SRT(Self-Refresh Temperature Range) | 0 |
| A6 | CAS Latency | Rtt_Nom | ASR(Auto-Self-Refesh) | 0 |
| A5 | CAS Latency | D. I. C(Output Driver Impedance Control) | CWL(CAS write Latency) | 0 |
| A4 | CAS Latency | D. I. C(Output Driver Impedance Control) | CWL(CAS write Latency) | 0 |
| A3 | RBT(Read Burst Test) | AL(Additional Latency) | CWL(CAS write Latency) | 0 |
| A2 | CL | Rtt_Nom | PASR(Partial Array Self-Refresh (Optional)) | MPR(Multipur pose Register) |
| A1 | BL | D. I. C(Output Driver Impedance Control) | PASR(Partial Array Self-Refresh (Optional) | MPR Location |
| A0 | BL | DLL | PASR(Partial Array Self-Refresh (Optional) | MPR Location |

### 2.3.4 DDR3 SDRAM 指令

DDR3 SDRAM 的工作及状态转换是通过命令来实现的，而这些指令主要是通过片选信号 CS#、行选通信号 RAS#、列选通信号 CAS#、读写控制信号 WE#的不同组合状态来实现的。DDR3 SDRAM 支持的主要指令有：

(1) 预充电命令；

用于关闭 Bank 中打开的行，适用于一个 Bank 或者所有 Bank。由于 SDRAM 的寻址具体独占性，所以在进行完读写操作后，如果要对同一 L-Bank 的另一行进行寻址，就要将原来有效（工作）的行关闭，重新发送行/列地址。L-Bank 关闭现有工作行，准备打开新行的操作就是预充电（Precharge）。预充电可以通过命令控制，也可以通过辅助设定让芯片在每次读写 操作之后自动进行预充电。

(2) 刷新命令；

用于定期刷新以维持其存储的内容，适用整个 SDRAM，而且优先级最高。刷新操作与预充电中重写的操作一样，都是用 S-AMP 先读再写。但是，所不同的是，预充电是对一个或所有 L-Bank 中的工作行操作，并且是不定期的，而刷新则是有固定的周期，依次对所有行进行操作，以保留那些久久没经历重写的存储体中的数据。那么要隔多长时间重复一次刷新呢？目前公认的标准是，存储体中电容的数据有效保存期上限是 64ms（毫秒，1/1000 秒），也就是说每一行刷新的循环周期是 64ms。这样刷新速度就是：行数量/64ms。我们在看内存规格时，经常会看到 4096 Refresh Cycles/64ms 或 8192 Refresh Cycles/64ms 的标识，这里的 4096 与 8192 就代表这个芯片中每个 L-Bank 的行数。刷新命令一次只对某一行有效，发送间隔也是随总行数而变化，4096 行时为 15.625μs（微秒，1/1000 毫秒），8192 行时就为 7.8125μs。

(3) 激活命令；

用于激活要进行读或者写操作所在的 Bank、行和列。

(4) 读命令；

用于从 SDRAM 中的特定地址读取数据。

(5) 写命令；

用于将数据写入到 SDRAM 中的特定地址中去。

表 2.4 和表 2.5 列出了 DDR3 命令的所有种类及其操作真值表。其中，BA 表示 Bank 地址，RA 表示行地址，CA 表示列地址，BC 表示突发 Chop，X 表示任意值，V 表示有效。

表 2.4 DDR3 命令的种类及操作真值表

| 功能 | 缩写 | CKE | | CS# | RAS# | CAS# | WE# | BA0～BA3 | A15～A13 | A12/BC | A10/AP | A9～A0, A11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Previous Cycle | Current Cycle | | | | | | | | | |
| 模式寄存器设置 | MRS | H | H | L | L | L | L | BA | | OP Code | | |
| 刷新 | REF | H | H | L | L | L | H | V | V | V | V | V |
| 自刷新入口 | SRE | H | L | L | L | L | H | V | V | V | V | V |
| 自刷新出口 | SRX | L | H | H | V | V | V | V | V | V | V | V |
| 自刷新出口 | SRX | L | H | L | H | H | H | V | V | V | V | V |
| 单一 Bank 预充电 | PRE | H | H | L | L | H | L | BA | V | V | L | V |
| 所有 Bank 预充电 | PREA | H | H | L | L | H | L | V | V | V | H | V |
| 激活指令 | ACT | H | H | L | L | H | H | BA | | RA (Row Address) | | |
| 写命令（BL8 or BL4） | WR | H | H | L | H | L | L | BA | RFU | V | L | CA |
| 写命令(BL8, on the fly) | WRS4 | H | H | L | H | L | L | BA | RFU | L | L | CA |
| 写命令(BL4, on the fly) | WRS8 | H | H | L | H | L | L | BA | RFU | H | L | CA |
| 带预充电的写命令（BL8 or BL4） | WRA | H | H | L | H | L | L | BA | RFU | V | H | CA |
| 带预充电的写命令 (BL8, on the fly) | WRAS4 | H | H | L | H | L | L | BA | RFU | L | H | CA |
| 带预充电的写命令 (BL4, on the fly) | WRAS8 | H | H | L | H | L | L | BA | RFU | H | H | CA |

表 2.5 DDR3 命令的种类及操作真值表（续表）

| 功能 | 缩写 | CKE | | CS# | RAS# | CAS# | WE# | BA0～BA3 | A15～A13 | A12/BC | A10/AP | A9～A0, A11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Previous Cycle | Current Cycle | | | | | | | | | |
| 读指令（BL4 or BL8） | RD | H | H | L | H | L | H | BA | RFU | V | L | CA |
| 读指令（BL4, on the fly） | RDS4 | H | H | L | H | L | H | BA | RFU | L | L | CA |
| 读指令（BL8, on the fly） | RDS8 | H | H | L | H | L | H | BA | RFU | H | L | CA |
| 带预充电的读指令（BL4 or BL8） | RDA | H | H | L | H | L | H | BA | RFU | V | H | CA |
| 带预充电的读指令（BL4, on the fly） | RDAS4 | H | H | L | H | L | H | BA | RFU | L | H | CA |
| 带预充电的读指令（BL8, on the fly） | RDAS8 | H | H | L | H | L | H | BA | RFU | H | H | CA |
| 空操作 | NOP | H | H | L | H | H | H | V | V | V | V | V |
| 反选 | DES | H | H | H | X | X | X | X | X | X | X | X |
| 进入省电模式 | PDE | H | L | L | H | H | H | V | V | V | V | V |
| 进入省电模式 | PDE | H | L | H | V | V | V | V | V | V | V | V |
| 退出省电模式 | PDX | L | H | L | H | H | H | V | V | V | V | V |
| 退出省电模式 | PDX | L | H | H | V | V | V | V | V | V | V | V |
| ZQ 长校准 | ZQCL | H | H | L | H | H | L | X | X | X | H | X |
| ZQ 短校准 | ZQCS | H | H | L | H | H | L | X | X | X | L | X |

### 2.3.5 DDR3 SDRAM 的写校准操作

为了获得更好的信号完整性，DDR3 存储器采用了新型的 Fly-by 拓扑架构[3]。Fly-by 拓扑架构的好处在于减少了 stub 的数量和长度，但是，它却带来了 DIMM 上每个 DRAM 时钟和数据选通信号之间的时间离散问题。这无疑是加大了内存控制器对 $t_{DQSS}$、$t_{DSS}$、$t_{DSH}$ 规范的控制难度。为此，DDR3 内存控制器需要通过写校准这一技术来弥补 Fly-by 拓扑结构所带来的时间离散的缺陷。

内存控制器可以通过写校准技术来实现调节 DQS-DQS#和 CK-CK#两对差分时钟之间的关系。正在进行写校准的内存控制器通过对 DQS-DQS#时钟进行延迟调节，使得 DQS-DQS#时钟的上升沿和 CK-CK#时钟的上升沿对齐，DRAM 也会异步反馈 CK-CK#时钟，并同抽样的 DQS-DQS#时钟进行比较。图 2.4 所示的为 DDR3 SDRAM 写校准的示意图：

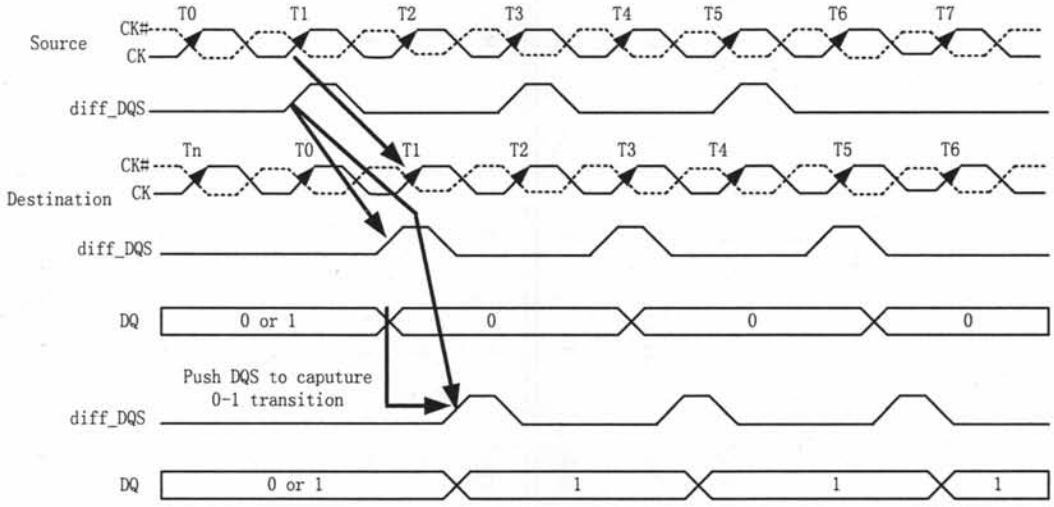


图 2.4 DDR3 SDRAM 写校准概念图

# 3 DDR3 内存控制器的 IP 核设计

本章将介绍内存控制器的 IP 核的设计过程，整个设计过程采用自顶向下的设计思路，首先对 DDR3 内存子系统的工作原理作简单分析，在这之后，参考 Altera 公司提出的外部存储器解决方案，论证设计方案并给出了控制器的整体架构设计及主状态机的设计，最后介绍各个子模块的 RTL 级设计。

## 3.1 DDR3 内存子系统分析

内存子系统是计算机系统的重要组成部分，它包括 CPU、Cache、内存控制器、内存四个部分[29-30]。CPU 在访问指令或数据时，它会先到 Cache 中去找，如果没有找到的话，CPU 才会通过内存控制器去访问内存。由于 CPU 发送的读写指令和地址不能被内存所识别，所以需要通过内存控制器来实现对内存的访问。因此，在进行内存控制器的整体架构设计之前，需要对内存控制器的具体功能进行简单分析。内存控制器连接着 CPU 的运算逻辑单元和内存，主要负责处理 CPU 运算逻辑单元发送过来的访存请求，根据访存请求的优先级进行访存调度，并按照 JEDEC DDR3 规范的时序要求，实现对内存单元的读写操作，此外， DDR3 SDRAM 在进行读写操作之前，必须进行相应的准备工作，包括芯片的初始化、模式寄存器的配置操作、预充电、定期的刷新操作等。因此，内存控制器还担负着对内存模组的管理工作。

基于上面的分析，可以将 DDR3 内存控制器在层次上分成两个部分：传输层和物理层。传输层负责接受来自 CPU 运算逻辑单元的访存请求，并将访存请求按照 DDR3 SDRAM 所需要的节拍发送到物理层。物理层负责将控制和地址信号按照 DDR3 JEDEC 规范所规定的时序传送到 DDR3 SDRAM 芯片，同时在传输层和 SDRAM 芯片之间创建地址和数据通路，负责将数据按照 DDR3 的速率读出或写入 SDRAM 芯片。图 3.1 显示的是 DDR3 内存、内存控制器以及 CPU 逻辑运算单元(ALU)之间连接关系。

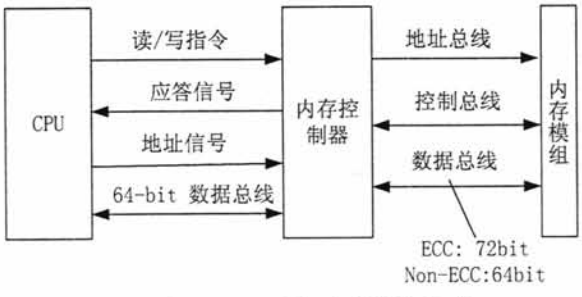


图 3.1 CPU 到内存的连接方式

## 3.2 设计方案论证及内存控制器整体架构设计

内存控制器的设计必须完成两大基本任务，一是所设计的内存控制器要实现对内存模组的管理，并完成 CPU 访存请求，这是衡量内存控制器能否正常工作的重要指标；二是要尽可能的提高访问内存的速度和带宽，这是衡量内存控制器性能的重要指标。除此之外，内存控制器的设计还必须兼顾它与市场主流内存的兼容性、成本等因素[31-35]。在前面第一章中，提到了本课题设计的内存控制器的适用范围是嵌入式系统，而嵌入式系统的特点是灵活性好、可移植性强，因此，嵌入式系统中几乎不可能使用单条超过 8GB 容量的内存模组，而且，为了保持嵌入式设备的便携性，超过 2 个 DIMM 的设计将很少出现。目前市场上主流的嵌入式设备均配置了容量在 2GB 或者更少的 Non-ECC 内存模组。然而，随着三星公司推出 4Gb 的 SDRAM 颗粒，单条 8GB 的 Unbuffer 内存模组也即将问世。鉴于以上几点，设计支持更大内存容量、多个内存通道和 DIMM 的嵌入式内存控制器，似乎并没有太为广阔的市场应用前景，所以，本课题设计的控制器 IP 核将支持最高容量为 8GB，单 Rank 或者双 Rank 的 Unbuffer ECC 内存模组，不支持目前市场上针对大容量和高稳定性而设计的 Register 内存模组和 Load Reduce 内存模组。

由于本课题的 IP 核设计是基于硬件描述语言的设计思路，而且是在 Stratix IV 开发板上进行验证，因此，在深入分析了 Altera 的外部存储器解决方案和 Stratix IV 器件族的特点后，确定本课题所设计的 IP 核将支持的最高数据传输频率为 800MHz，而且，Stratix IV 开发板上只配置了一个 DIMM 插槽，因此，本课题的内存控制器只支持单通道，故理论传输带宽为 8500Mbps。

根据前面对内存以及内存子系统的分析，综合考虑实验条件，将内存控制器设计工作分为两个部分，即内存控制器的控制逻辑部分设计和 ALTMEMPHY 数字接口的设计。图 3.2 给出了内存控制器的整体架构。其中，控制器控制逻辑部分是本设计的重点，它主要包括用户接口模块、指令仲裁调度模块、Bank 管理模块、初始化模块、定时器、刷新控制电路、ECC 模块、ODT 生成逻辑、地址命令解码电路以及写校准电路几个部分。 ALTMEMPHY 则负责在控制器逻辑和 DDR3 SDRAM 之间创建数据和命令通路，完成信号时钟域的转换等工作，该模块的设计工作将在完成控制器的控制逻辑部分的设计之后展开，根据控制逻辑所设计的引脚情况，在 Quartus 10.1 中使用 Megacore 工具例化设计参数，自动生成 ALTMEMPHY。

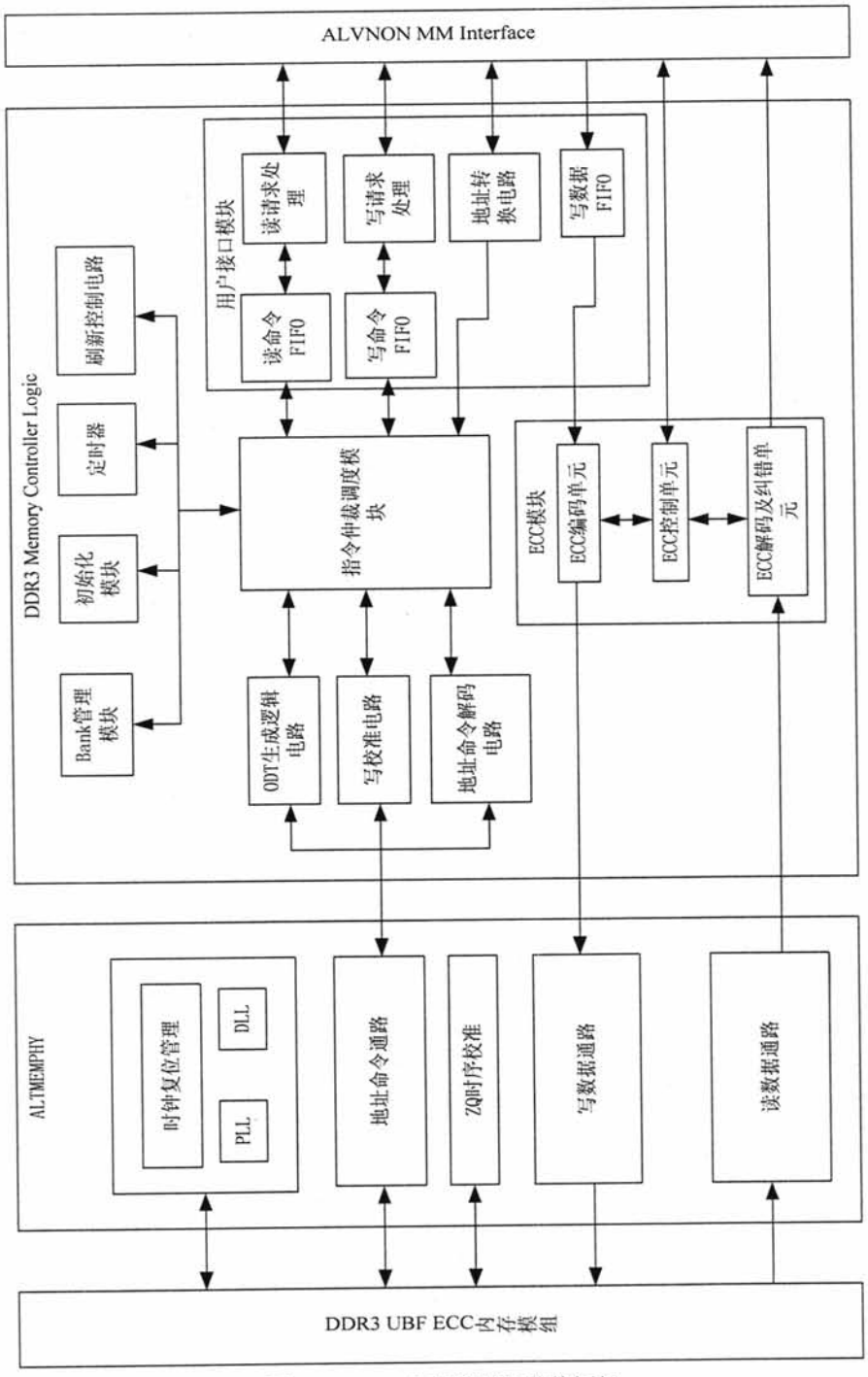


图 3.2 DDR3 内存控制器整体框架

## 3.3 内存控制器主状态机设计

完成上述内存控制器的设计后，下面开始介绍该内存控制器的简易工作流程图，见图 3.3 所示。该流程图介绍了 CPU 访问内存的具体过程。

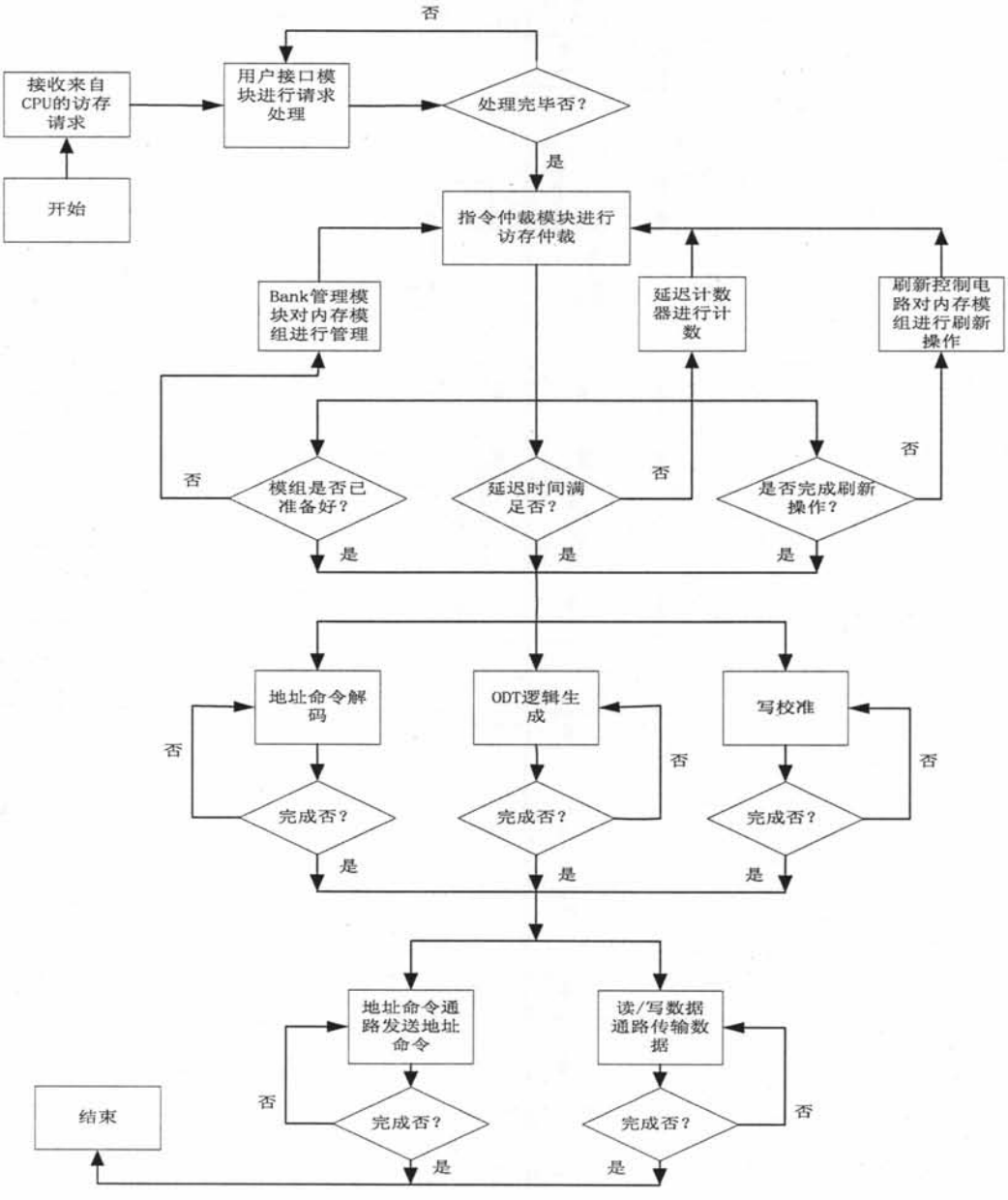


图 3.3 DDR3 内存控制器简易工作流程图

## 3.4 DDR3 内存控制器的 RTL 级设计

### 3.4.1 用户接口模块设计

用户接口模块是内存控制器和 CPU 的主要接口，它包括读请求处理、读命令 FIFO、写请求处理、写命令 FIFO、写数据 FIFO 以及地址转换电路等模块。该接口用来接收上层发送的访存请求指令以及写数据，然后通过地址转换电路将 CPU 发送过来的地址按照控制器设定的方式进行地址映射，并将写入数据交由 ECC 模块进行编码。下面分别介绍用户接口模块中各个子模块的设计。

(1) 读请求处理模块设计

读请求处理模块负责完成对 CPU 发送过来的读命令的处理，包括读请求的接收、应答以及读命令的发送。每一个读请求都会有一个 ID 号，且 ID 号均为奇数。当上层用户（CPU）发送正常的读请求时，该模块首先会查询读命令 FIFO 是否已满，如果 FIFO 已满，该模块将向 CPU 发送忙碌应答信号，表明内存控制器有尚未处理的读命令；如果 FIFO 未满，则其向 CPU 发送空闲信号，接收 CPU 的读指令，并完成时钟域的转换，最后，该模块会将该条指令发送给读命令 FIFO。

(2) 写请求处理模块设计

同读请求处理模块相似，写请求处理模块负责完成对 CPU 发送过来的写命令的处理，包括写请求的接收、应答以及写命令的发送。每一个写请求也会有相应的 ID 号，且 ID 号均为偶数。该模块接收到 CPU 发送过来的写请求信号后，首先会查询写命令 FIFO 是否已满，如果 FIFO 已满，该模块将向 CPU 发送忙碌应答信号，表明内存控制器有尚未处理的写命令；如果 FIFO 未满，则其向 CPU 发送空闲信号，接收 CPU 的写指令，并完成时钟域的转换，最后，该模块会将该条指令发送给写命令 FIFO。

(3) FIFO 设计

FIFO(First In First Out，先进先出)是一种采用环形存储结构的先进先出存储器。其使用一个双端口存储器存放数据，数据发送方在一端写入数据，接收方在另一端读出数据，能够协调好两个时钟域的工作，满足高时钟频率的要求。FIFO 在 FPGA 设计中主要用来缓冲数据和隔离时钟或相位差异。访问 FIFO 时不需要地址线，只需要数据线和读写控制信号线，且数据地址由内部读写指针自动加 1 完成，因此利用 FIFO 实现数据的缓存具有接口简单、读写方便的优点。FIFO 可以分为同步 FIFO 和异步 FIFO 两种，由于用于用户接口的写入时钟和读出时钟频率不同，因此本模块中的所有 FIFO 均为异步 FIFO。图 3.4 所示的是一个异步 FIFO 的内存结构图。

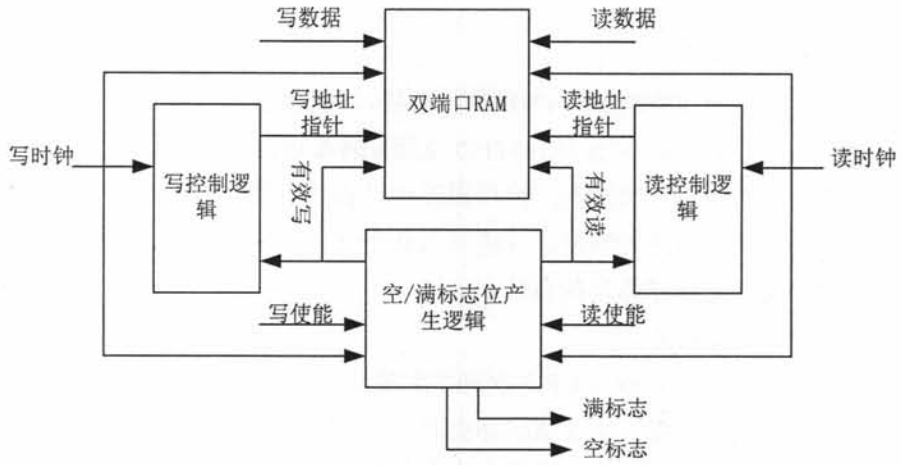


图 3.4 异步 FIFO 内存结构图

下面首先介绍一下用户接口模块中各个 FIFO 的具体功能如下：

① 读命令 FIFO 设计

读命令 FIFO 负责接收由读请求处理模块发送过来的读命令，并暂存读命令，同时，负责将读命令排队发送给指令仲裁模块。该 FIFO 可以暂存连续 8 条读指令。

② 写命令 FIFO 设计

写命令 FIFO 负责接收来自写请求处理模块的写命令，并暂存写命令，此外，它还负责将写指令排队发送给指令仲裁模块。该 FIFO 同样可以暂存 8 条连续写指令。

③ 写数据 FIFO 设计

CPU 在执行写内存操作时，写数据 FIFO 负责暂存 CPU 发送过来的待写入数据，当指令仲裁模块仲裁可以执行写操作时，该 FIFO 中的数据会排队发送到 ECC 模块进行编码加工，然后在通过写数据通路写入到内存。

(4) 地址转换电路设计

由于 CPU 发送过来的地址是一个线性地址，并不能为内存模组直接识别，所以，需要一个地址转换电路来实现对其的地址映射。该模块主要包括两个部分：一个是 SPD（Serial Presence Detection，模组存在串行检测）信息解码电路，另一个是地址映射电路。

对于任何一个计算机系统而言，系统在上电自检过程中，BIOS（Basic Input/Output System，基本输入输出系统）或者引导程序会通过 SMBus（System Management Bus，系统管理总线）去读取存储在内存模组上的 SPD 信息。SPD 信息中包含有该内存模组诸如行、列地址线数目、Bank 数、Rank 数、容量大小等一切信息 [18-19]。SPD 信息解码

电路负责解码出这些存储在 SPD 中得信息，然后地址映射电路依据这些信息进行线性地址和内存物理地址的一一映射，从而完成地址映射。表 3.1 列出了本论文中解码的 SPD 信息及其在 JEDEC 规范中的对应的字节位置。

表 3.1 SPD 信息解码内容

| JEDEC 中对应的字节位置 | 功能描述 |
|---|---|
| 2 | DRAM器件类型 |
| 3 | 模组类型 |
| 4 | SDRAM的芯片密度及Bank数量 |
| 5 | SDRAM的地址定义 |
| 6 | SDRAM的工作电压 |
| 7 | 模组组织结构 |
| 8 | 模组总线宽度 |
| 12 | SDRAM最小时钟周期 |
| 14 | CAS延迟，低字节 |
| 15 | CAS延迟，高字节 |
| 18 | RAS到CAS的时间延迟 |
| 19 | 两个行选通命令之间的时间延迟 |

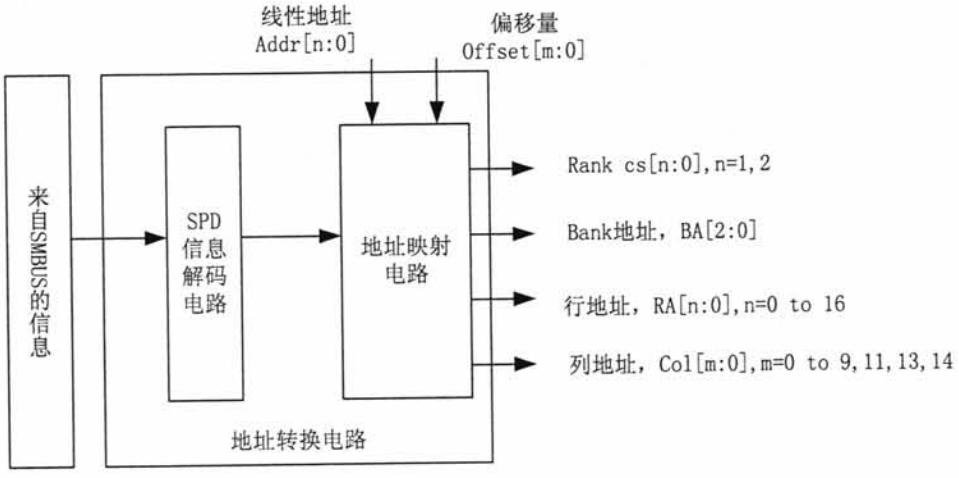


图 3.5 地址转换电路内部结构图

图 3.5 所示的就是地址转换电路的内部结构图，该地址映射电路实际上是一个编码器，对于不同的内存模组，其地址映射的方式会有所不同，表 3.2 分别列出了不同内存组织结构的地址映射方式，表中涵盖了目前市场所有的 UDIMM 的组织结构。

表 3.2 DDR3 地址映射方式

| DDR3 模组类型 | | | 芯片物理地址 | | | 系统线性地址 | | | |
|---|---|---|---|---|---|---|---|---|---|
| 模组容量 | 芯片密度 | 模组结构 | Bank 地址 | 行地址 | 列地址 | 片地址 | 段地址 | 行地址 | 列地址 |
| 1GB | 512Mb | 1R×4 | BA[2:0] | A[12:0] | A[9:0], A[11] | Addr[27] | Addr[26:24] | Addr[23:11] | Addr[10:0] |
| 1GB | 512Mb | 2R×8 | BA[2:0] | A[12:0] | A[9:0] | Addr[27:26] | Addr[25:23] | Addr[22:10] | Addr[9:0] |
| 2GB | 1Gb | 1R×4 | BA[2:0] | A[13:0] | A[9:0], A[11] | Addr[28] | Addr[27:25] | Addr[24:11] | Addr[10:0] |
| 2GB | 1Gb | 2R×8 | BA[2:0] | A[13:0] | A[9:0] | Addr[28:27] | Addr[26:24] | Addr[23:10] | Addr[9:0] |
| 2GB | 2Gb | 2R×8 | BA[2:0] | A[14:0] | A[9:0] | Addr[28] | Addr[27:25] | Addr[24:10] | Addr[9:0] |
| 4GB | 1Gb | 2R×4 | BA[2:0] | A[13:0] | A[9:0], A[11] | Addr[29:28] | Addr[27:25] | Addr[24:11] | Addr[10:0] |
| 4GB | 2Gb | 1R×4 | BA[2:0] | A[14:0] | A[9:0], A[11] | Addr[29] | Addr[28:26] | Addr[25:11] | Addr[10:0] |
| 4GB | 2Gb | 2R×8 | BA[2:0] | A[14:0] | A[9:0] | Addr[29:28] | Addr[27:25] | Addr[24:10] | Addr[9:0] |
| 8GB | 2Gb | 2R×4 | BA[2:0] | A[15:0] | A[9:0] | Addr[30:29] | Addr[28:26] | Addr[25:10] | Addr[9:0] |
| 8GB | 4Gb | 1R×4 | BA[2:0] | A[15:0] | A[9:0] , A[11] | Addr[30] | Addr[29:27] | Addr[26:11] | Addr[10:0] |
| 8GB | 4Gb | 2R×8 | BA[2:0] | A[15:0] | A[9:0] | Addr[30:29] | Addr[28:26] | Addr[25:10] | Addr[9:0] |

### 3.4.2 初始化模块设计

SDRAM 在进行正常的读写操作之前，需要对其进行初始化操作，以避免在进行正常读写过程中出现一些不可预测的错误。初始化模块负责完成内存器件的初始化，使得器件进入空闲等待状态，等待被访问。初始化操作只发生在系统上电和复位之后。根据 DDR3 JEDEC 规范，SDRAM 在进行初始化时，首先要完成一系列的等待延迟操作，然后再完成配置模式寄存器操作，之后再进行 PLL 的锁定，最后执行 ZQCL（ZQ 长校准）。因此，将初始化模块分成两个更小的子模块，分别是延迟计数器以及配置模块。由于 ALTMEMPHY 已集成了时钟复位管理电路和 ZQ 校准引擎，因此，将初始化过程分成两个阶段，第一阶段进行延迟等待、模式配置，待完成后报告指令仲裁模块，第二阶段先由 ALTMEMPHY 的时钟复位管理模块中内嵌的 DLL 执行 DLL 锁定开始，接着指令仲裁模块发送一个 ZQCL 指令给 ALTMEMPHY，最后 ALTMEMPHY 完成 ZQ 校准工作。完成以上操作后，SDRAM 便进入空闲等待状态。下面将分别介绍初始化模块中两个子模块的设计思路如下：

(1) 延迟计数器

考虑到初始化模块中需要进行等待操作的指令较多，如果每个指令都使用一个延迟计数器进行延迟操作的话，将会增加很多冗余代码，为此，将延迟计数器分为通用延迟计数寄存器（cd_cnt，Common Delay Counter）、专用延迟计数寄存器（sd_cnt,Special Delay Counter）以及等待完成延迟计数寄存器 Init_Post。通用延迟计数寄存器用于初始化每一个操作过程所需要的延迟的计数，专用延迟计数寄存器用于配置模块所需的计时和 DLL 锁定所需延迟的计数，等待完成延迟计数寄存器用于产生延迟计数等待初始化完成。

(2) 配置模块

配置模块是为适应 SDRAM 的使用灵活性而设计的。在前面章节中也介绍了模式寄存器的设定是通过 BA[2:0]和 A[15:0]地址线的状态来设定。同时，为了保证对模式寄存器的可靠写入，两个模式寄存器设定命令（MRS）之间必须有 $t_{MRD}$ 的时间间隔；同样，为了保证 SDRAM 能够可靠地更新模式寄存器设定的新特性，在 MRS 和 Non-MRS 命令之间也必须有 $t_{MOD}$ 的时间间隔。所以，配置模块必须完成对地址线状态（也即 OP-Code）的解码工作和两个连续指令的时间间隔的控制工作。图 3.6 给出了配置模块的工作流程图。

在初始化阶段，配置模块要依次完成对 MR2、MR3、MR1、MR0 四个模式寄存器的设置。同时，由于执行 MRS 命令并不会对 SDRAM 中的数据造成破坏，因此，在非初始化阶段，同样可以通过置 MRS 指令来访问该配置模块，从而完成相应的配置工作。

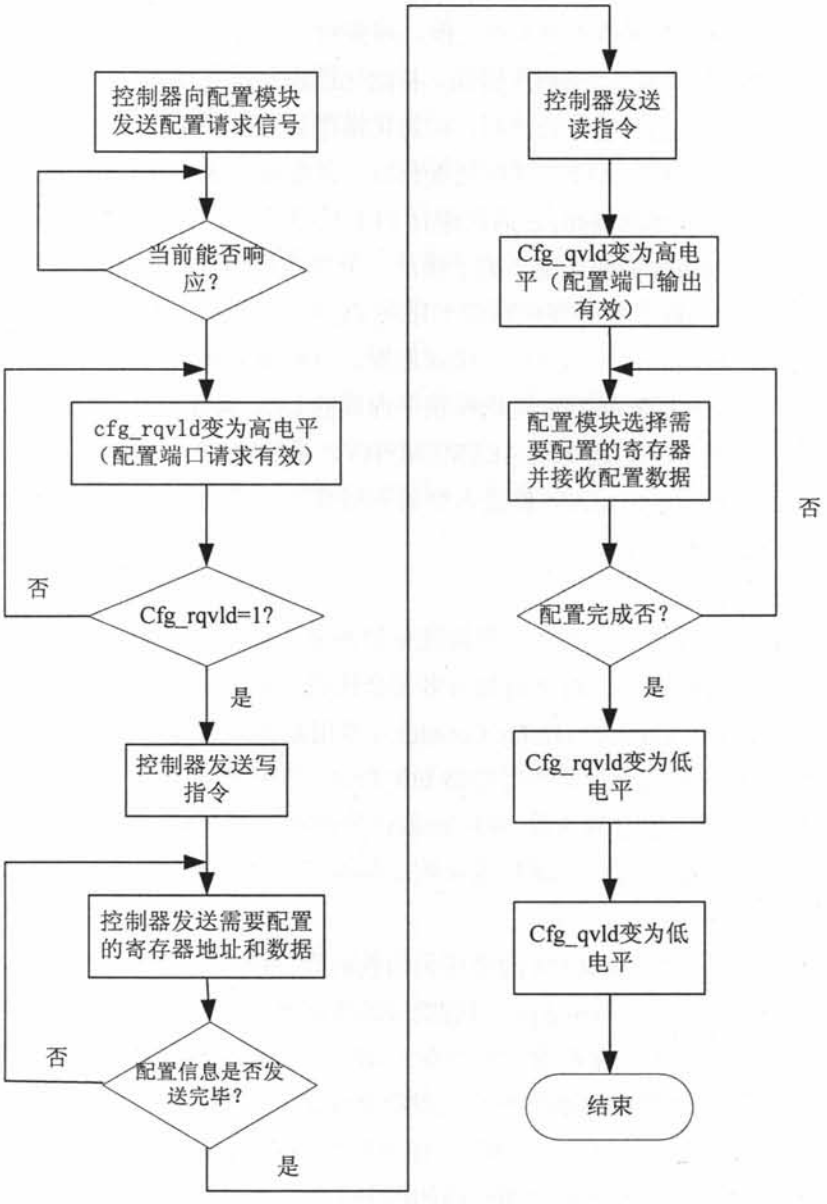


图 3.6 配置模块流程图

综上所述，给出本设计的初始化模块的内部结构图，如图 3.7。

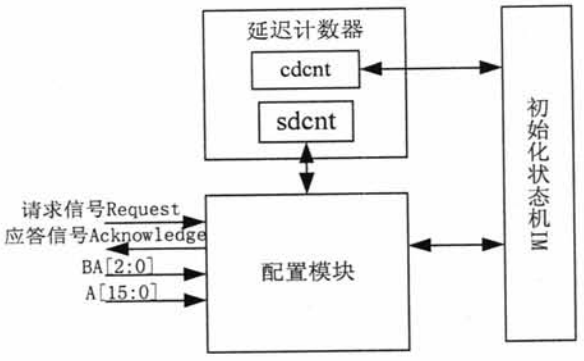


图 3.7 初始化模块内部结构图

最后，给出初始化模块的工作流程图，如图 3.8 所示。

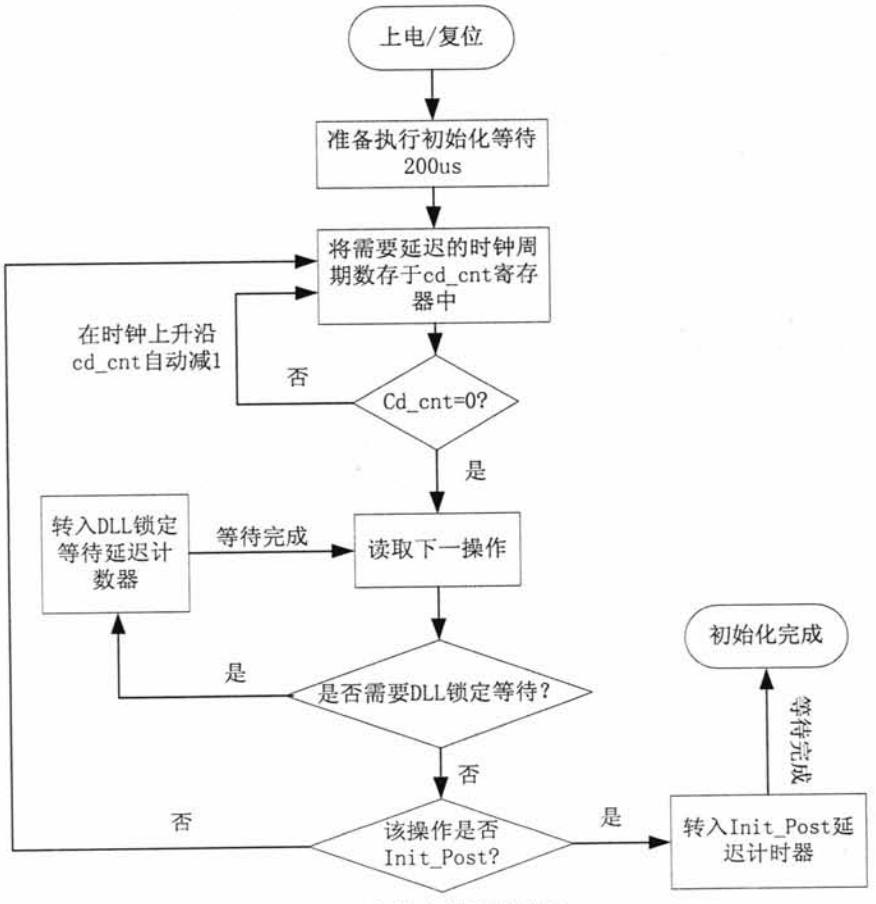


图 3.8 初始化过程流程图

### 3.4.3 Bank 管理模块设计

内存控制器不仅担负着响应 CPU 的访存请求的任务，还负责对内存模组进行管理的工作，而实现这个工作，Bank 管理模块扮演着非常重要的角色。在前面章节中，已经介绍了 SDRAM 进行读写操作的过程。如果要访问某个存储单元，首先必须通过激活指令（ACT）来打开待访问单元的 Bank 和行，然后再选通列并发送读写指令。因此，需要 Bank 管理模块负责监测内存模组上所有 SDRAM 芯片内所有 Bank 的工作状态，同时将这些状态信息及时的报告给指令仲裁模块，以方便仲裁模块判断是否在发送读写指令前先发送一个预充电指令。

### 3.4.4 定时器模块设计

根据 DDR3 JEDEC 规定，SDRAM 在执行两个连续命令之间需要一定的延迟。定时器模块实际上是一个延迟计数器组，用于产生这些延迟。它包括读到写延迟计数器 Rd_to_Wr[n:0]、写到读延迟计数器 Wr_to_Rd[n:0]、读到预充电延迟计数器 Rd_to_Pre[n:0]、写到预充电延迟计数器 Wr_to_Pre[n:0]等各种需要的延迟操作。

### 3.4.5 刷新控制电路设计

刷新控制电路实际上也是一个刷新计数器。前面也介绍到，DDR3 SDRAM 在正常温度下的行刷新周期为 7.8us。这个刷新控制电路就每隔 7.8us 报告指令仲裁模块，请求执行刷新命令。刷新命令具有最高优先级。

### 3.4.6 指令仲裁模块设计

指令仲裁模块的设计是整个控制器逻辑设计中最为重要的设计单元。它负责接收用户接口模块发送来的指令，然后根据 Bank 管理模块、初始化模块等模块的输入，来仲裁生成相应的指令，并将地址和指令发送给地址命令解码电路。该模块由主状态机和指令寄存器两部分组成。

DDR3 有 16 类指令，共计 25 条，负责实现对 DDR3 SDRAM 的操作。指令仲裁模块使用4-bit二进制对16 类指令进行统一编码并存于指令寄存器中。当系统正常工作时，控制器各个管理部件（子模块）会向指令仲裁模块发出请求，请求其发送指令以完成相应的功能。因此，为了合理的控制和管理这些请求，就必须有一个特别的仲裁算法，以便在多个设备提出请求时进行仲裁以决定哪个设备优先获得响应。图 3.9 是指令仲裁模块的主状态机。

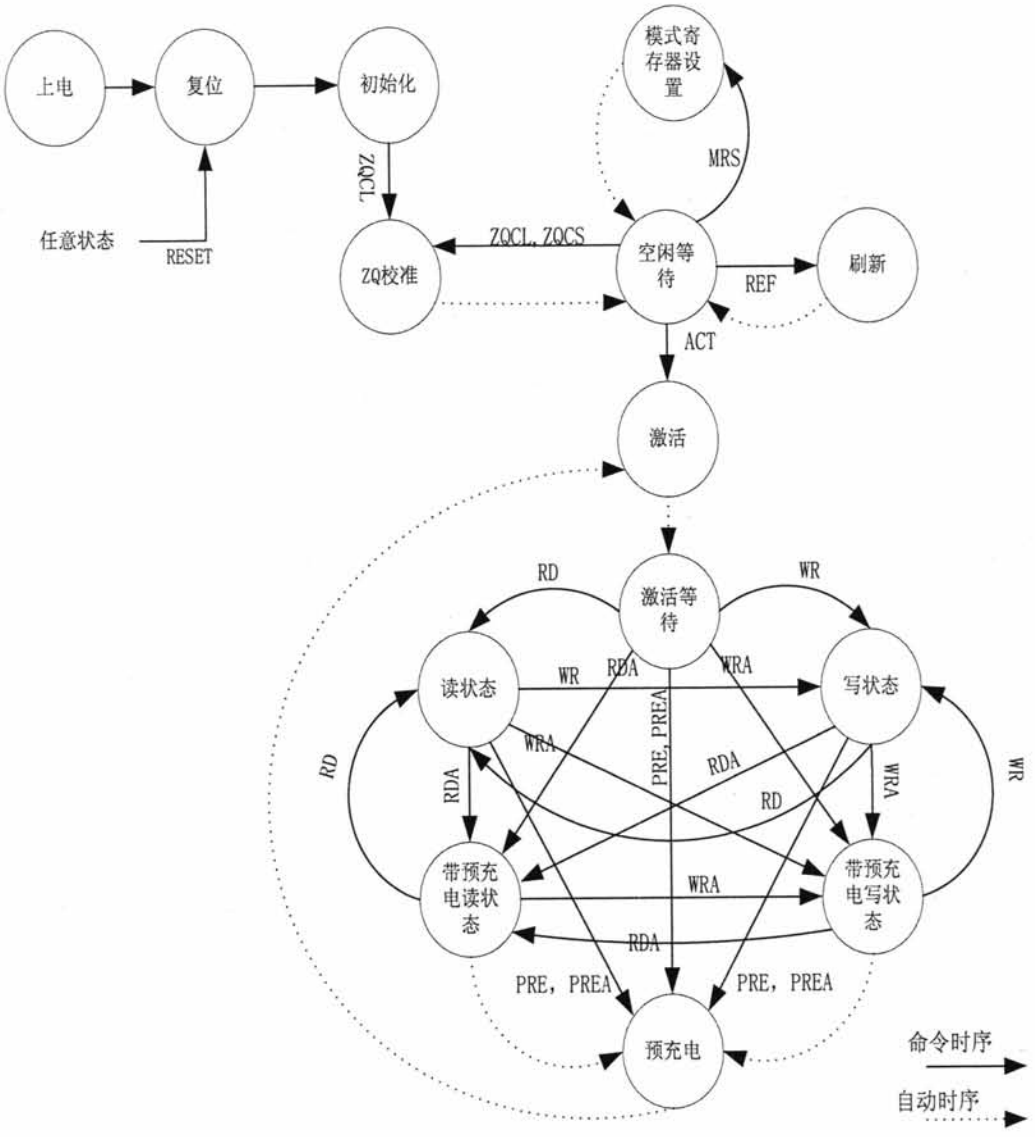


图 3.9 指令仲裁模块主状态机设计

### 3.4.7 地址命令解码电路设计

地址命令解码电路接收来自指令仲裁模块经过仲裁后的指令，并将指令按照 DDR3 JEDEC 规定，转换成能够表达 RAS、CAS、WE 等指令的对应时序逻辑之后，交由地址命令通路发送给内存模组。

### 3.4.8 ODT 生成逻辑设计

ODT (On Die Termination，片内信号终结器）是为了提高 SDRAM 信号完整性而设计的。它允许内存控制器通过 ODT 引脚来生成相应的逻辑。

### 3.4.9 写校准电路设计

前面章节中也曾提到，基于 DDR3 SDRAM 的内存模组在设计上采用了 Fly-by 的拓扑架构，虽然减少了信号的衰减，但也带来了时序约束的问题。因此，就需要设计一个写校准电路来消除时序不收敛的现象。

### 3.4.10 ECC 模块设计

(1) ECC 模块的构成

ECC 模块是为了提高内存子系统的工作稳定性而设计的，它能发现 2 位错误并能够改正 1 位错误。ECC 模块主要由编码器、译码和纠错电路以及 ECC 控制逻辑三部分组成，图 3.10 所示的是 ECC 模块的结构图。编码器负责将上层用户传送过来的 64-bit 的数据通过汉明编码将其编码成 72-bit 后再发送给内存。译码及纠错电路负责将内存传送过来的 72-bit 数据解码成 64-bit 的数据后发送给上层用户。如果解码后发现 1 位错误，纠错电路会将其纠正，同时纠错电路会将错误的类型和地址报告给相应寄存器；如果解码后发现 2 位错误，那么纠错电路将无法纠正错误，这时只报告给 ECC 控制逻辑，由控制逻辑产生中断，报告给上层用户。ECC 控制逻辑负责控制多个编码器和译码纠错电路并行工作，以满足该模块可以适应不同的数据宽度（当然，这里的数据宽度必须是 64 的整数倍）。它和上层用户之间有一个 32-bit 的接口，用于产生中断并报告相关寄存器的状态。

ECC 控制逻辑内有几类寄存器，用于指示 ECC 模块的状态，下面将介绍这几类寄存器：

① 配置寄存器，它用于记录 1 位错误和 2 位错误的所设定的阈值，一旦计数器达到这个阈值，会立即产生一个中断。

② 状态寄存器，它用于记录发生错误的类型、地址以及对应字节的 ECC 错误的校正子（Syndrome）。

③ 计数器，它用于记录已经被检测到 1 位错误或者 2 位错误和已经被修正的 1 位错误。

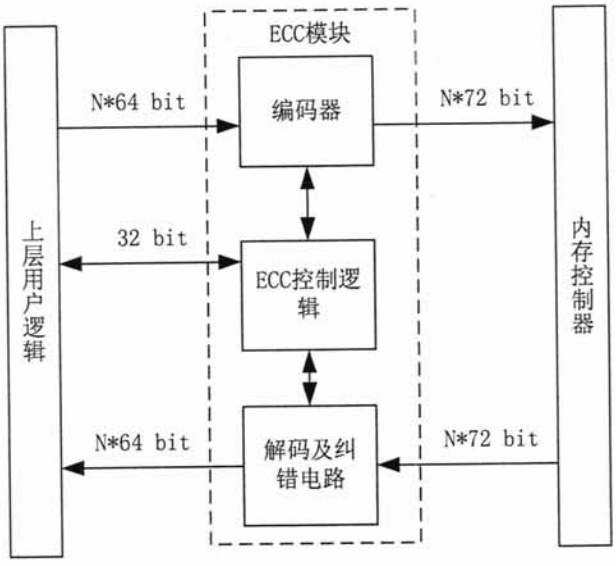


图 3.10 ECC 模块结构图

(2) ECC 编码算法

ECC 编码算法采用增强型汉明编码（Enhance Hamming Coding），又称为 SEC-DED（Single Error Correction-Double Error Detection）编码，即该编码能自动更正 1 位错误并检测 2 位错误，它是一种 FEC(Forward Error Correction)编码，所谓 FEC，是指在不需要向发送方请求更多信息或者重发的情况下，接收方就能自动纠正接受数据的错误。根据汉明的理论，对于任意长度为 k 位的信息矢量 u 进行编码，添加一个长度为 m 位的校验矢量 p，组成一个长度为 m+k 位的码字（Codeword）[36]。如果将矢量 p 转化为整数时，它的值必须是在 0~m+k 之间，而 m 位的二进制能区分出$2^m$种情况，因此，m 的大小必须满足：

$$2^m \geq m + k + 1 \tag{3.1}$$

这个公式也被称之为汉明定律。所以对于 64bit 的数据矢量进行汉明编码，需要 7bit 的校验矢量，这时的编码被称为 SEC 编码。ECC 编码正是再此基础上再增加一位校验码，形成了 SEC-DED 编码。因此，每 64bit 数据矢量 u 进行 ECC 编码需要一个长度为 8 位的校验矢量 q。下面将具体介绍校验矢量 q 的生成方式：

① 依据汉明定律计算出码字的总长度。

② 标记码字中所有是 2 的幂次方的位置分配给校验矢量 p。除已标记位置之外，其他的所有位置均分配给信息矢量 u。

③ 计算 SEC 所需的校验矢量 p，校验矢量 p 的计算方法如下：

记校验矢量的最低位为 p0，其值为 1,3,5,7...等位（即隔一位校验一位，这些位置信息转换成二进制，其第 0 位均为 1）的异或运算结果（做偶奇偶校验）。

记次低位位 p1,其值是 2,3,6,7,10,11...等位（即隔两位校验两位，这些位置信息转换成二进制，其第 1 位均为 1）的异或运算结果。

类似地，p2 是 4,5,6,7,12,13,14,15,20,21,22,23（隔四位校验四位，这些位置信息转换成二进制，其第 2 位均为 1）的异或结果。依次类推，可以计算出校验矢量 p 中的其他几位。

④ 最后，再对已完成 SEC 编码的 m+k+1 位长度的码字增加 1 位奇偶校验位，完成 SEC-DED 编码，形成长度为 m+1 位的 ECC 码（也就是校验矢量 q）。64bit 数据 ECC 码的生成方式可以用如表 3.3 的校正子表来表示：

表 3.3 校正子表

| | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|---|
| 1000 | D63 | D62 | D61 | D60 | D59 | D58 | D57 | CB7 |
| 0111 | D56 | D55 | D54 | D53 | D52 | D51 | D50 | D49 |
| 0110 | D48 | D47 | D46 | D45 | D44 | D43 | D42 | D41 |
| 0101 | D40 | D39 | D38 | D37 | D36 | D35 | D34 | D33 |
| 0100 | D32 | D31 | D30 | D29 | D28 | D27 | D25 | CB6 |
| 0011 | D25 | D24 | D23 | D22 | D21 | D20 | D19 | D18 |
| 0010 | D17 | D16 | D15 | D14 | D13 | D12 | D11 | CB5 |
| 0001 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | CB4 |
| 0000 | D3 | D2 | D1 | CB3 | D0 | CB2 | CB1 | No error |

通过这个校正子表，我们可以算出校验矢量 P 的每一位（即表中 CB1～CB7）。例如：

CB1=D0⊕D1⊕D3⊕D4⊕D6⊕D8⊕D10⊕D11⊕D13⊕D15⊕D17⊕D19⊕D21⊕D23⊕D25⊕D26⊕D28⊕D30⊕D32⊕D34⊕D36⊕D38⊕D40⊕D42⊕D44⊕D46⊕D48⊕D50⊕D52⊕D54⊕D56⊕D57⊕D59⊕D61⊕D63；

当使用上述算法计算出 CB1～CB7，再对 D0～D63 以及 CB1～CB7 共 71bit 数据进行一次奇偶校验得出 CB8，从而完成 ECC 编码工作。

(3) ECC 解码及纠错算法

解码及纠错电路会首先根据内存传送过来的信息矢量 u（即数据线 DQ0-DQ63 上面的信息）进行 ECC 编码，编码后形成新的 ECC 码（校验矢量 r）,然后再同内存传送过

来的校验矢量 q（即数据线 CB0-CB7 上的信息）进行异或运算，运算结果被称为校正子(Syndrome)。如果 Syndrome 全为 0，则表明没有错误发生；如果 Syndrome 有且仅有 1 个‘1’，则表示校验矢量有 1bit 错误，由于校验矢量只是用来校验数据的，并不是真正被需要的数据，纠错逻辑将不会有动作发生。如果 Syndrome 有奇数个‘1’发生，则表明有 1bit 的错误发生，解码逻辑会通知纠错逻辑来修正错误；如果有偶数个‘1’发生，则表明有 2bit 的错误发生，此时只报告错误信息，但无法更正错误。

## 3.5 ALTMEMPHY 数字接口设计

### 3.5.1 ALTMEMPHY 功能介绍

ALTMEMPHY 是 Altera 公司专为外部 SDRAM 存储器定义的物理层接口，它主要负责将控制器发送过来命令、地址信号发送至内存，同时为读写数据创建通路，完成信号时钟域的转换。它主要包括以下几个子模块：

(1) 写数据通路：负责将控制器发送过来的写数据发送至内存器件。在执行写操作时，内存控制器会以半倍数据速率（HDR）连续送出 4n bit 的数据至写数据通路，写数据通路首先会通过串并转换将 4n bit 的数据并行地送入信号时钟域转换模块，从而完成数据传输率由半倍数据速率（HDR）向双倍数据速率（DDR）的转化。最后，DDR 数据将被送往内存的 DQ 管脚。

(2) 内嵌 PLL 和 DLL 的时钟复位管理：负责时钟的生成及相位调整、复位管理，同时，它还负责管理不同类型的时钟网络，包括在初始化阶段的 DLL 锁定等。

(3) 读数据通路：负责将 DQ 上的数据发送至内存控制器。在执行读操作时，它首先要完成读数据捕获，然后将捕获数据由 DDR 速率转换至 SDR 速率，最后通过并串转换，将读数据同步到 HDR 时钟域，最后，HDR 速率的读数据将被发送到内存控制器。

(4) 地址命令通路：负责接收来自控制器的地址命令信号，并对其完成时钟域的转化。

(5) ZQ 时序发生器：根据 DDR3 Specification 的规定，在内存器件初始化时，SDRAM 需要进行 ZQ 校准，从而使得 SDRAM 在进入 idle 状态时获得最大时序裕量。

### 3.5.2 ALTMEMPHY 数字接口介绍

ALTMEMPHY 的接口信号主要包括以下几个部分：

(1) 至 SDRAM 的 I/O 接口；

(2) 时钟复位信号；

(3) 外部 DLL 信号；

(4) 用户定义的校准 ODT 控制信号；

(5) 写数据接口信号；

(6) 读数据接口信号；

(7) 地址命令接口信号；

(8) 校准控制及状态接口信号；

(9) 调试接口信号。

其中，这些接口信号中，带有 mem_前缀的信号是连接到内存模组上的，而带有 ctl_前缀的信号是连接到控制器上的。

### 3.5.3 ALTMEMPHY 设计

当完成了控制器所以子模块的设计后，根据控制器到内存端的端口定义，在 Quartus 软件中使用 MegaCore 设计 ALTMEMPHY。图 3.11 是使用 Quartus 软件的宏功能模块自动生成的 ALTMEMPHY 的元件符号图。

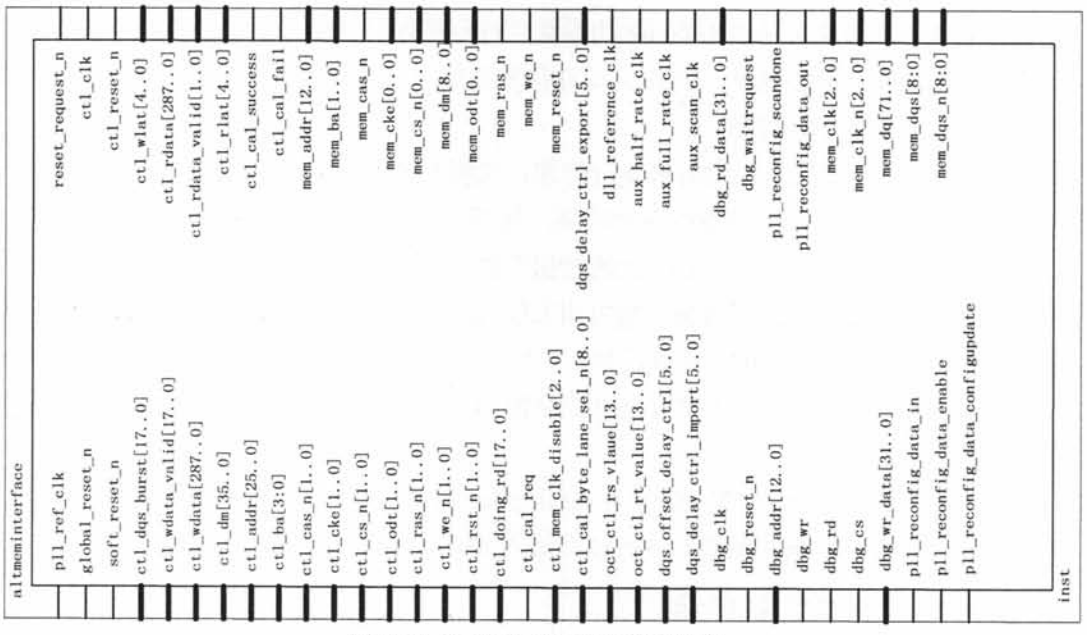


图 3.11 ALTMEMPHY 元件符号图

# 4 DDR3 内存控制器 IP 核的软件仿真

仿真与验证是集成电路设计过程中的一个重要环节。验证过程是证明设计正确的过程，验证的目的是为了保证设计实现与设计规范是一致的，而仿真是在物理实现之前进行验证工作的一种方法。目前，验证一般是通过仿真实现的[37]。本章首先介绍软件仿真的验证平台搭建、软件仿真验证的方法，然后给出 IP 核的核心子模块的 RTL 级仿真结果并对仿真结果进行分析。

## 4.1 验证平台（Test Bench）设计

### 4.1.1 验证平台的组成

验证平台通常包括确定的输入序列和期望输出响应两部分，它是一个代码的集合。仿真程序提供设计的输入激励并监控设计的响应。典型验证平台的结构如图 4.1 所示，它包括待验证设计、输入激励、参考值、输出响应和比较 5 个部分。

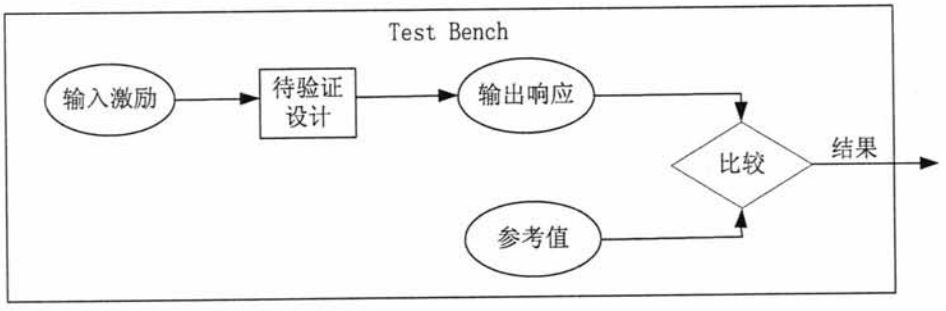


图 4.1 验证平台示意图

下面简单介绍这 5 个组成部分的作用：

(1) 待验证设计

待验证设计（DUV，Device Under Verification）可以是 RTL 级代码，也可以是网表，一般是 HDL 语言描述的 RTL 级代码。

(2) 输入激励

就是使得 DUV 工作的输入激励，它是不含任何延时的源代码程序，通过添加外部激励，观察信号在寄存器之间的传输情况，并由此来判断寄存器之间的连接逻辑是否实现了系统的行为功能。

(3) 参考值

通常使用参考模型或期望响应两种方法来产生验证平台参考值。参考模型就是用于和 DUV 进行比较用的设计，它可以是行为模型，还可以是已经验证过的设计。期望响应法是以参考文件代替参考模型。参考文件中存放着参考模型的标准输出。

在本论文中使用期望响应法，假设 DUV 的逻辑符合设计的要求，根据输入激励，应当输出的期望结果。

(4) 输出结果模块

此模块为仿真验证的实际输出响应结果。

(5) 比较模块

比较模块的主要功能是实现期望输出（期望值）和实际输出（输出结果）进行比较。采用逐个比较的方法，一旦发现数据不一致，就进行报错。

### 4.1.2 平台搭建

在 DDR3 控制器的 IP 核软件仿真中，最重要的是仿真平台的搭建[38-40]。该平台的搭建是建立在充分理解内存控制器所要完成的功能的基础之上的。搭建该平台还需综合考虑以下几个方面：

(1) 确定所设计模块需要进行测试的特性；

(2) 确定测试方法和策略，搭建应用于验证工作的完整测试平台；

(3) 采用硬件描述语言编写验证平台的各个组件；

(4) 建立测试向量库；

(5) 测试软件 EDA 工具的使用，主要使用 Altera Quartus 10.1 SJ Full Version 和 Modelsim _Altera 6.6c 两个工具。

下面开始 IP 验证平台的搭建工作：

首先在 Quartus 软件中调用 DDR3 存储器模型和 ALTMEMPHY IP 核，然后连接所设计的控制器和 DDR3 存储器模型。再编写测试向量，用于控制对 DDR3 存储器模型中数据的读写，构成 RTL 级仿真的测试向量。最后编写顶层测试代码，例化控制器 IP 核、ALTMEMPHY IP 核、DDR3 存储器模型。该验证平台的测试软件顶层伪码如下：

```
module test_example_drive
{
function lfsr();     //利用 lfsr 生成用于读写验证的随机数据
function write();    //将 lfsr 生成的随机数据写入 DDR3 的指定地址
function read();     //将 DDR3 指定地址内的数据读出
function compare();  //比较写入的数据与读出的数据是否一致
function out();      //输出比较结果
```

```
display;    //显示验证结果
}
```

下面简单介绍上述代码段中涉及到的几个子函数的功能：

(1) Lfsr( ) 函数，用于读写操作的测试激励是采用 LFSR（Linear Feedback Shift Register，线性反馈移位寄存器）生成的随机数据。该函数的核心代码实际上一个带反馈的线性移位寄存器。

(2) Write( ) 函数，该函数将 lfsr 函数生成的随机数据（设计期望值）通过控制器的控制，写入 DDR3 存储芯片的指定地址。

(3) Read( ) 函数，该函数用于将 DDR3 存储芯片的特定地址中的数据读出来（输出的响应结果）。

(4) Compare( ) 函数，该函数用于比较实际输入的数据和读出的数据是否完全一致。

(5) Out( ) 函数，该函数用于输出比较，如果一致，显示为 Success，否则显示为 Fail。

特别值得一提的是，这里所使用的 DDR3 存储器模型仅供软件仿真阶段使用，下载到 FPGA 中将会被开发板上的内存模组所替代。本设计的验证平台模型如图 4.2 所示。

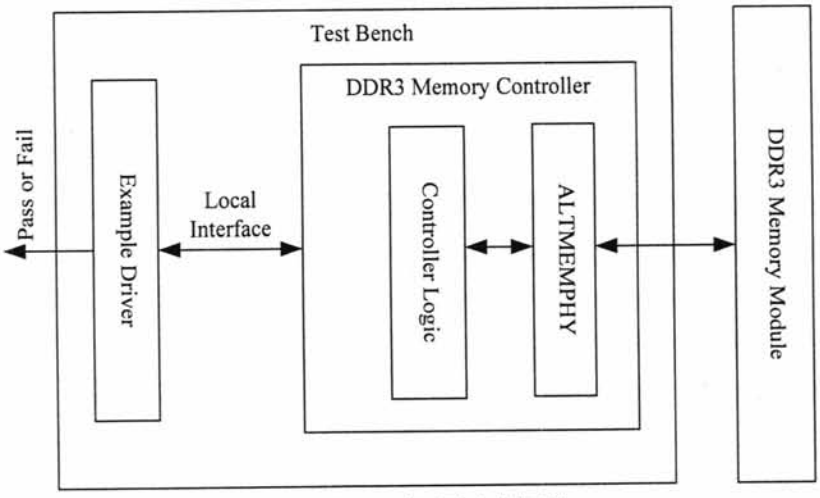


图 4.2 验证平台框架图

## 4.2 软件验证流程

整个软件验证均在 Quartus 10.1 和 Modelsim 6.6c 中进行。Quartus10.1 和 Modelsim 6.6c 联合编译、调试及仿真的的流程如下：

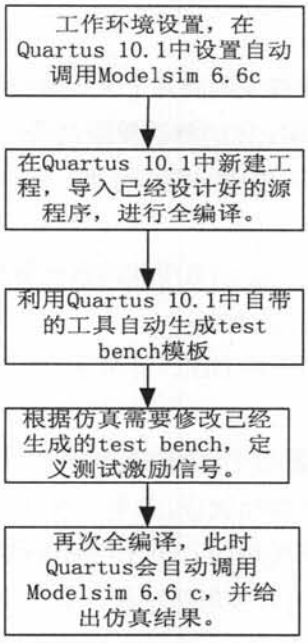


图 4.3 软件仿真流程图

## 4.3 RTL 级仿真测试结果及分析

RTL 级仿真的任务是完成对所设计的各个子模块的功能验证，是进行 IP 核验证的必经之路。它是在设计物理实现之前进行设计验证的一种方法。RTL 级仿真采用动态仿真技术，从设计规范出发，开发测试向量集和标准输出，然后对设计施加激励，观察信号在寄存器之间的传输情况来判断寄存器之间的连接逻辑是否正确；观察实际输出和理想输出是否一致，判断该数字系统是否实现了预期的功能[38-40]。本小节将给出内存控制器重要子模块仿真的结果及其分析，主要包括用户接口模块、初始化模块、指令仲裁模块、ECC 模块以及地址命令解码电路，对于其他模块，由于内部结构比较简单，鉴于篇幅，不在一一给出每个模块的仿真波形。

### 4.3.1 用户接口模块仿真测试

用户接口模块主要负责处理上层用户的读写请求，而读写请求在被送到内存控制器之前，都会暂存在用户接口模块的 FIFO 中，因此，FIFO 的可靠性至关重要。本小节将给出 FIFO 的仿真结果。

FIFO 设计采用 Mega Pulg-in Megfuction 直接生成。用户接口模块一共包含 4 个 FIFO，每个 FIFO 的差异仅表现在 FIFO 的大小上，结构上完全相同。因此，这里给出数据 FIFO

分别在读 FIFO 操作和写 FIFO 操作时的波形及其分析。图 4.4 的中间部分是该 IP 核所设计的 FIFO，为了 FIFO 的 RTL 级验证，辅助设计了写控制逻辑和读控制逻辑，用于读写 FIFO 时的操作。

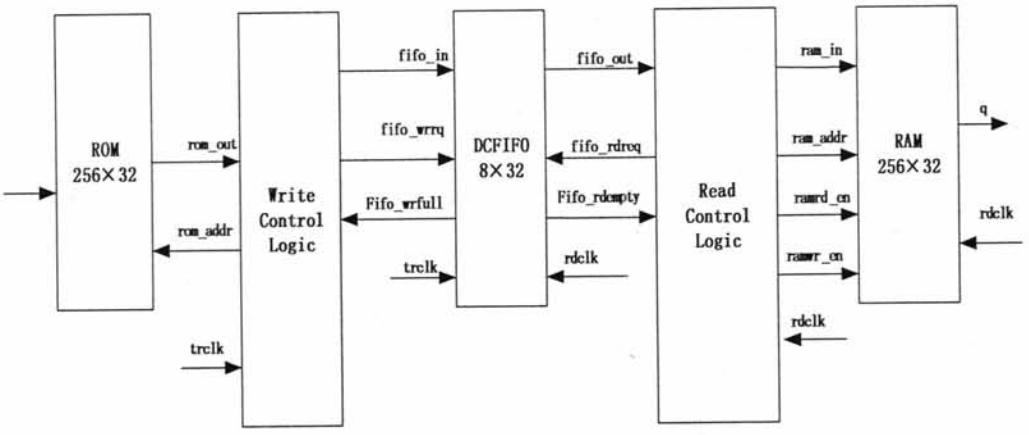


图 4.4 FIFO 仿真原理图

图 4.5 和 4.6 分别是对 FIFO 进行写操作和读操作时的仿真波形。

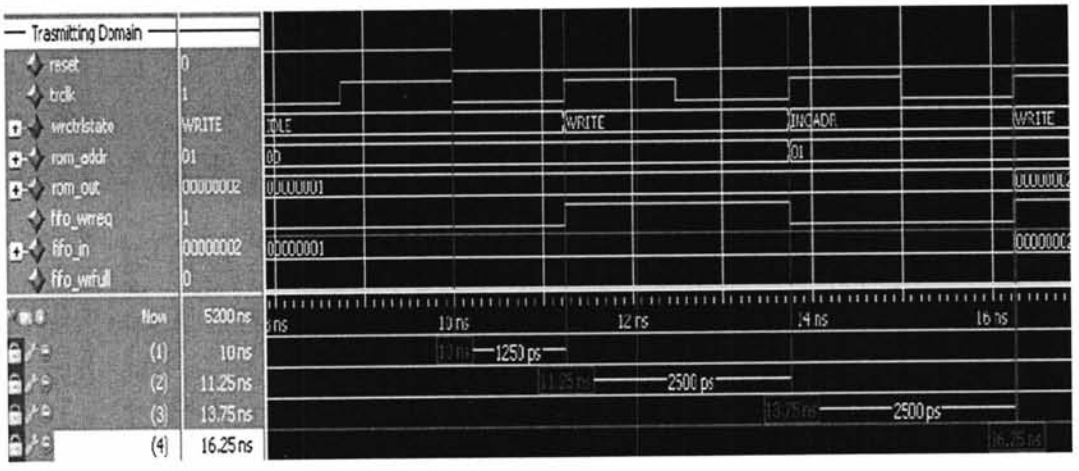


图 4.5 数据 FIFO 写操作波形图

在图 4.5 中，在到达 10 ns 之前，reset 信号被拉高，此时，FIFO 的写控制器处于空闲（Idle）状态。在空闲状态下，fifo_wreq 信号被拉低，请求数据从地址 rom_addr="00" 的存储单元读出。由于 ROM 没有数据缓冲，因此，数据会立即输出到 rom_out 上。

当到达 11.25 ns 时，在 reset 信号和 fifo_full 信号都被反选的情况下，fifo_wreq 信号拉高，rom_addr 地址保持不变，写控制器将由空闲状态进入写（Write）状态。此时数据将由 rom_out 送入写控制逻辑并经 fifo_in 写入到 FIFO 中去。

在 13.75 ns 时刻，fifo_wreq 信号被拉低，此时只要 rom_addr 不是“FF”且 fifo_wrfull 信号为低电平，写控制器将进入地址指针下移阶段（INCADR），rom_addr 由“00”变成“01”。之后，写控制逻辑的状态转换重复进行，只到 fifo 被写满为止。

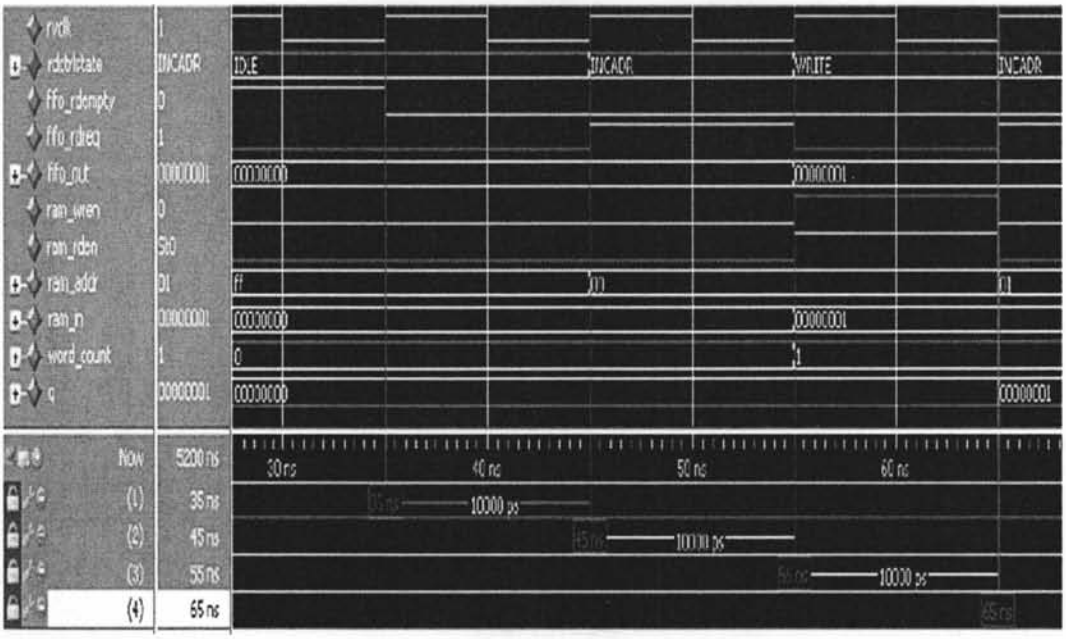


图 4.6 FIFO 读操作波形图

在图 4.6 中，在 35 ns 之前，由于 fifo_empty 信号被拉高，此时不管 reset 信号是否有效，读控制器处于空闲（Idle）状态，ram_addr 地址为“FF”，表示 FIFO 为空。在 45ns 时刻，fifo_empty 信号被拉低，读控制器进入地址指针下移阶段（INCADR）。ram_addr 自动加 1，ram_addr 变成“00”。在下一个时钟时，读控制器进入写（Write）状态，从“00”处开始读取 FIFO 中的数据至 fifo_out，并使能 ram_wen 信号，将数据通过 ram_in 写到 RAM 中去，与此同时，读控制器还使能 ram_ren 信号，以便数据可以通过 q 输出。此外，写控制器还会对 word_out 自动加‘1’，以表明正确读出 FIFO 数据的个数。之后，读控制器的状态转换便重复，只到 fifo_empty 信号被拉高为止。

### 4.3.2 初始化模块仿真测试

由于初始化过程中涉及到的操作比较多，且持续时间比较长，因此，无法用一张波形图来描述整个初始化过程。在这里， 只截取了初始化过程中的两个片段，见图 4.7 和图 4.8。

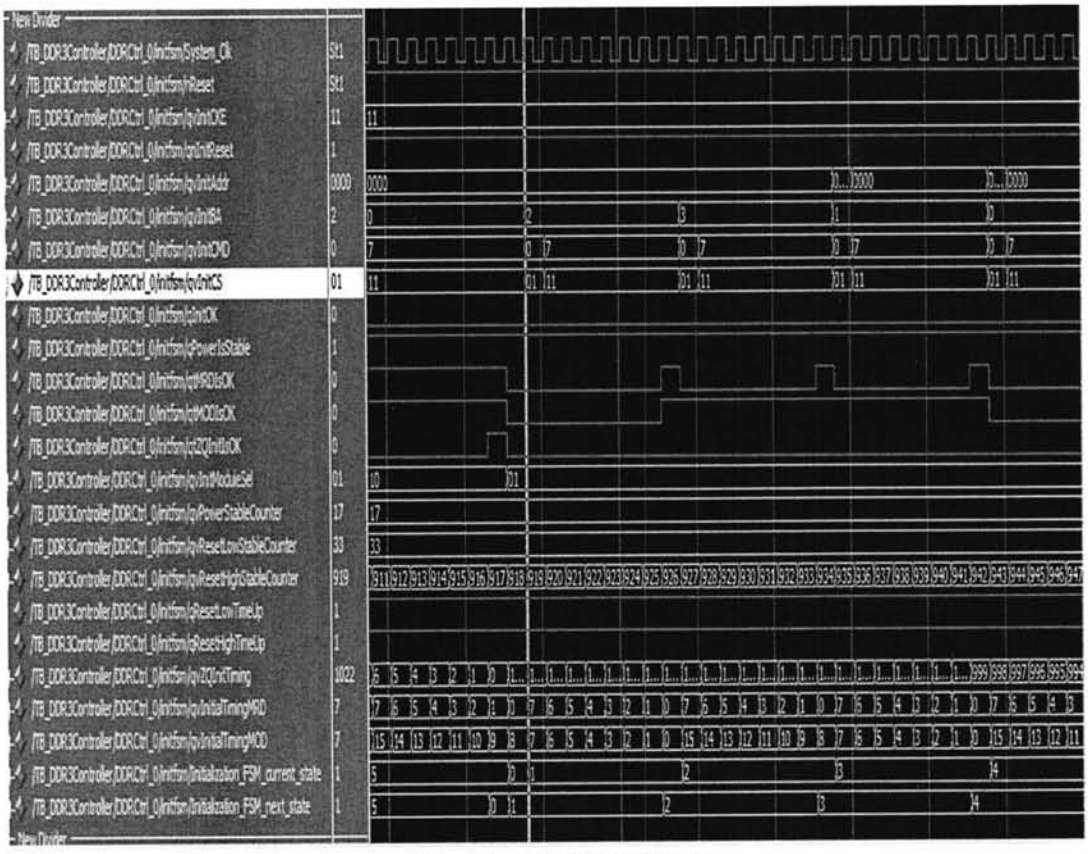

图 4.7 初始化模块仿真结果

从图 4.7 和图 4.8 中可以看到，初始化模块是按照 DDR3 SDRAM 规范进行代码编写的。在初始化过程开始时，qvPowerStableCounter、qvResetLowStableCounter、qvResetHighStableCounter 三个和上电复位有关的计数器开始计数延迟，当延迟时间到时，拉高如 qPowerIsStable 等相应的指示信号。初始化顺利进入模式寄存器配置过程，在这一过程中，控制器会对 MR0，MR1，MR2，MR3 四个寄存器进行参数配置，也就是执行 4 条 MRS 指令，而根据 JEDEC 规范，连续执行 2 条 MRS 指令必须有 $t_{MRD}$ 的时间间隔，另外，MRS 和 Non-MRS 指令直接也必须有 $t_{MOD}$ 的时间间隔，因此，设置了

qvInitialTimingMRD 和 qvInitialTimingMOD 两个延迟寄存器进行延迟计数，当满足延迟计数时，内部信号 qtMRDIsOK 和 qtMODIsOK 将被拉高，使得初始化过程进入到 ZQ 校准阶段。这一阶段，同样有 qvZQIninitTiming 寄存器进行计数延迟，之后拉高 qtZQInitIsOK 信号，至此，整个初始化过程结束，对外端口 qInitOK 将被拉高，初始化模块任务完成。图 4.7 和图 4.8 中所涉及到的信号定义见表 4.1 和表 4.2，内部信号以及相关寄存器定义见表 4.3。

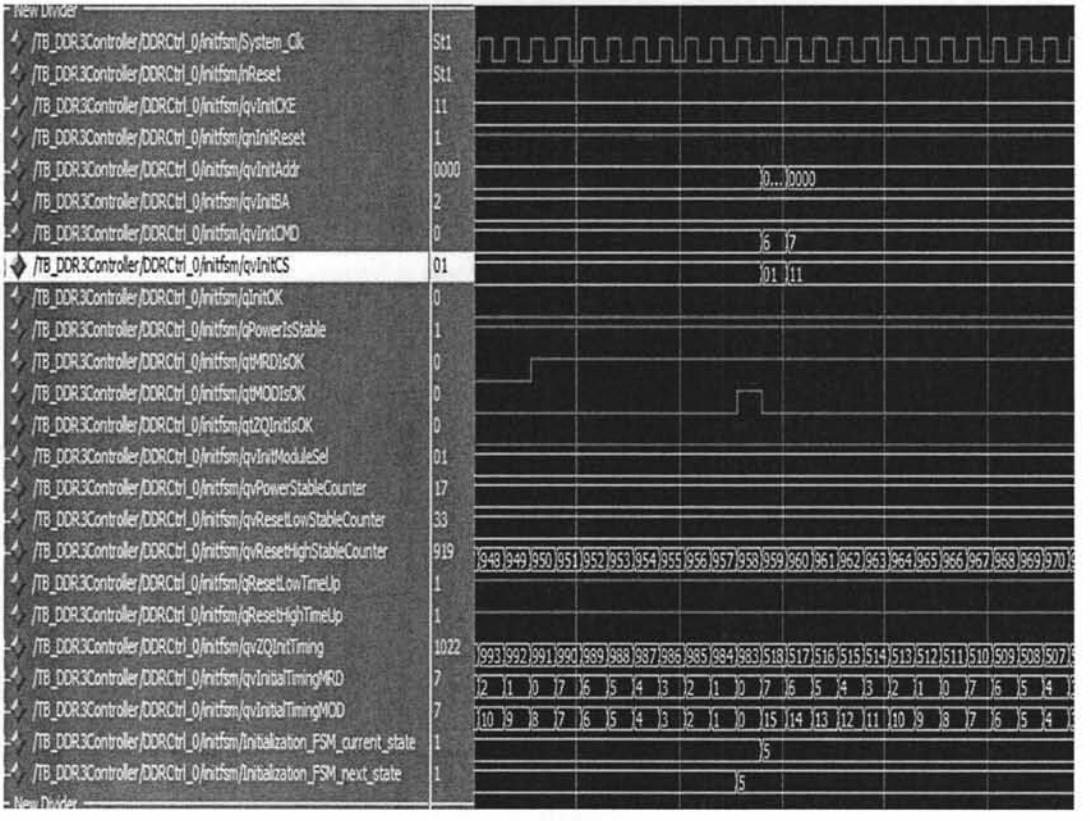


图 4.8 初始化模块仿真结果（续）

表 4.1 初始化模块端口信号定义

| 信号名称 | 类型 | 信号定义 |
|---|---|---|
| System_clk | Input | 时钟信号 |
| qvInitCKE[1:0] | Output | 时钟信号使能 |
| nReset | Input | 初始化模块复位信号 |
| qvInitAddr[14:0] | Output | 初始化单元行列地址 |

表 4.2 初始化模块端口信号定义（续）

| 信号名称 | 类型 | 信号定义 |
| --- | --- | --- |
| qvInitBA[2:0] | Output | 初始化单元 Bank 地址 |
| qvInitCMD[2:0] | Output | 初始化命令，用于执行初始化过程 |
| qvInitCS[1:0] | Output | 片选信号，用于选择待初始化的 DRAM |
| qInitOK | Output | 初始化完成信号，用于指示初始化过程结束 |
| qnInitReset | Output | 置复位信号 |

表 4.3 初始化模块内部信号定义

| 信号名称 | 信号定义 |
| --- | --- |
| qPowerIsStable | 稳定上电指示 |
| qtMRDIsOK | MRD 延迟时间到，可以置下一条 MRS 指令 |
| qtMODIsOK | MOD 延迟时间到，可以置 Non-MRS 指令 |
| qtZQInitIsOK | ZQ 校准完成指示 |
| qvInitModuleSel | 初始化模块选择 |
| qvPowerStableCounter | 稳定上电时间计数器 |
| qvResetLowStableCounter | 复位信号低电平时间计数器 |
| qvResetHighStableCounter | 复位信号高电平时间计数器 |
| qResetLowTimeUp | 复位信号维持低电平时间到 |
| qResetHighTimeUp | 复位信号维持高电平时间到 |
| qvZQIninitTiming | ZQ 校准时间计数器 |
| qvInitialTimingMRD | MRD 延迟计数器 |
| qvInitialTimingMOD | MOD 延迟计数器 |
| Initialization_FSM_Current_state | 初始化模块当前状态 |
| Initialization_FSM_Next_State | 初始化模块下一状态 |

### 4.3.3 指令仲裁模块仿真测试

指令仲裁模块是整个控制器设计的核心模块，它是整个控制器的调度中心，所有的指令均由其仲裁调度后发其他模块。指令仲裁模块的仿真时序比较复杂，图 4.9 所示的是该模块在初始化状态开始前的仿真时序。

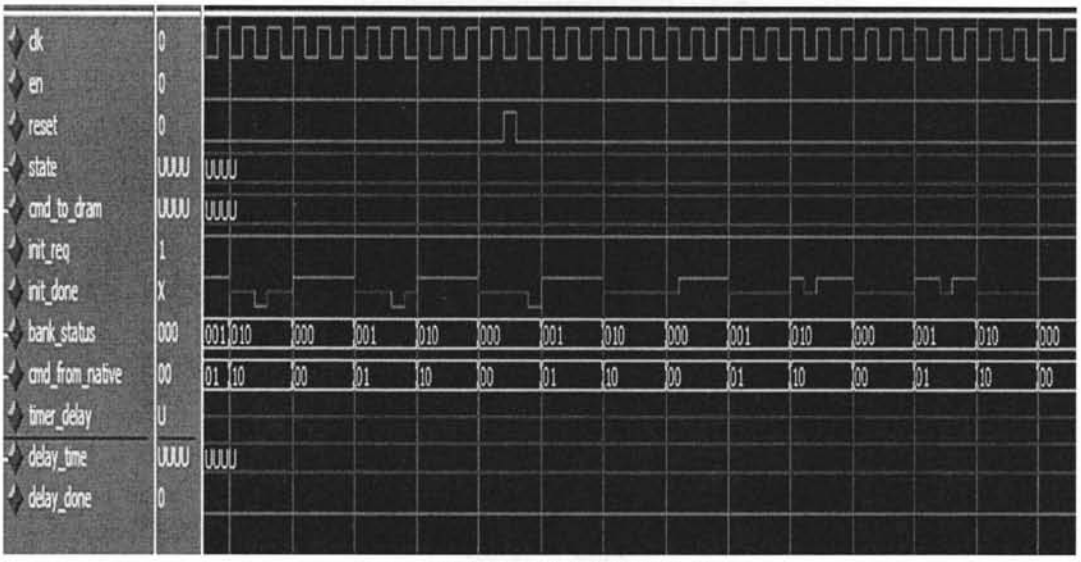


图 4.9 指令仲裁模块仿真时序图

在图 4.9 中，各信号的定义见表 4.4 所示：

表 4.4 指令仲裁模块信号定义

| 信号名称 | 端口方向 | 信号定义 |
|---|---|---|
| clk | Input | 时钟信号，上升沿有效 |
| en | Input | 使能信号，高电平有效 |
| reset | Input | 复位信号，用于内存器件初始化，高电平有效。 |
| state[0:3] | Output | 指令仲裁器状态指示，用于 debug |
| cmd_to_dram[0:3] | Output | 指令生成结果，发往地址命令解码模块。 |
| init_req | Input | 初始化请求信号，发往初始化模块 |
| init_done | Output | 初始化完成信号，发自初始化模块 |
| bank_status[0:2] | Input | Bank 状态信息，发自 bank 管理模块 |
| cmd_from_native[0:1] | Input | 上层用户访存请求信号，发自用户接口模块。 |
| timer_delay | Output | 请求定时器延迟，高电平有效，发往定时器。 |
| delay_time[0:3] | Output | 延迟时间，发往定时器模块，定时器才用减法计数。 |
| delay_done | Input | 延迟时间到，发自定时器模块。 |

图 4.9 显示的是内存模组处在未定义状态下的仿真结果。在这一状态下，所有的输出信号均为未定义。

### 4.3.4 ECC 模块仿真测试

由于 ECC 模块的设计分为两个部分，即编码器及解码纠错电路。因此，ECC 模块的仿真测试也分为两个部分展开。

编码器只有在数据从 CPU 端读入时（即执行写内存操作）才工作。图 4.10 是 ECC 编码器和解码纠错电路的仿真结果：

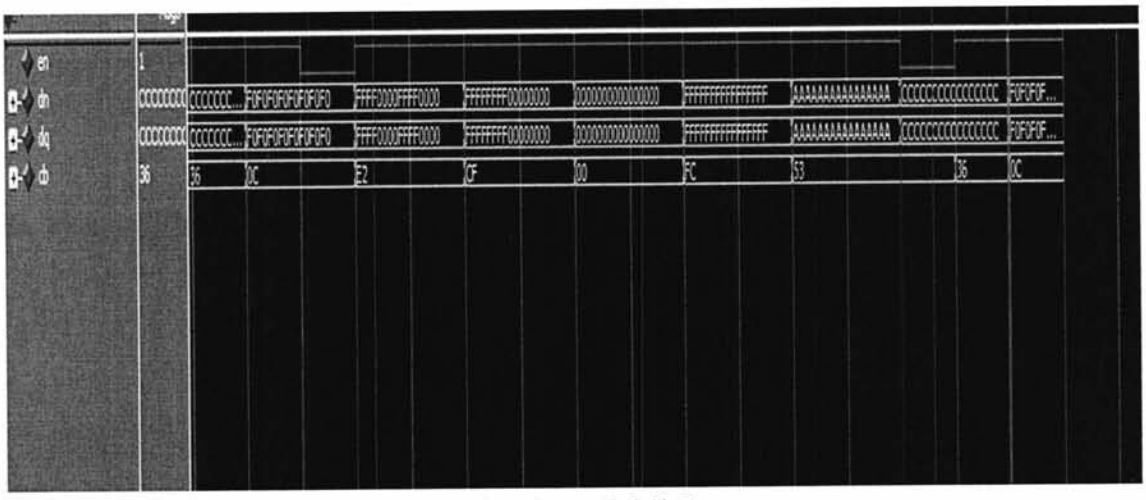


图 4.10 ECC 编码器仿真结果

在图 4.10 中，各信号的定义见表 4.5：

表 4.5 ECC 编码器信号定义

| 信号名称 | 类型 | 信号定义 |
|---|---|---|
| en | Input | 使能信号 |
| dn | Input | 输入数据，64 bit |
| dp | Output | 输出数据，64bit |
| cb | Output | 输出 ECC 码，8bit |

从图中可以看到，在 en 被拉高的情况下（图中 en 为高电平的时间范围是 246832ps to 440000ps），64bit 输入数据 dn 经过编码器后，被编码成由原始数据和 ECC 码组成的 72bit 的信息，其中，输出数据 dp 和 dn 完全一致，这点可以从仿真图中看到。而 cb 则是按照 ECC 算法得出的校验码。

当有数据从内存端读入时（即执行读内存操作），解码及纠错电路才会工作。解码电路在仿真是采用错误注入的方法验证解码电路的可靠性，错误注入的源代码如下：

```
data: process
begin --inject single bit error, Dp (0) flip
wait for 40 ns;
Dp<=X"0000000000000001"; --inject single bit error, Dp(4) flip
wait for 40 ns;
Dp<=X"FFFFFFFFFFFFFFEF"; --inject double bit error; both occurred at the data bits. Dp (0), Dp (2) both flip
wait for 40 ns;
Dp<=X"AAAAAAAAAAAAAAAF"; --inject double bit error; one occurred at the data bit, the other occurred at the check bit. Dp (0), cb (0) both flip.
wait for 40 ns;
Dp<=X"CCCCCCCCCCCCCCCD"; --inject double bit error, both occurred at the check bits, cb(1), cb(2) both flip
wait for 40 ns;
Dp<=X"F0F0F0F0F0F0F0F0"; --inject single bit error, cb(3) flip
wait for 40 ns;
Dp<=X"FFFF0000FFFF0000"; -- no error
wait for 40 ns;
Dp<=X"FFFFFFFF00000000";
end process;
check_bit: process
begin --inject single bit error,Dp(0) flip
wait for 40 ns;
cb<=X"08"; --inject single bit error,Dp(4) flip
wait for  40 ns;
cb<=X"EF"; --inject double bit error,both occurred at the data bits. Dp (0), Dp (2) both flip
wait for 40 ns;
cb<=X"58"; --inject double bit error,one occurred at the data bit,the other occurred at the check --bit.Dp(0),cb(0) both flip
wait for 40 ns;
cb<=X"30"; --inject double bit error, both occurred at the check bits,cb(1),cb(2) both flip
wait for 40 ns;
cb<=X"0A"; --inject single bit error, cb(3) flip
wait for 40 ns;
cb<=X"E6";
```

```
no error
wait for 40 ns;
cb<=X"CE";
end process;
```

上述程序使用 2 个状态机来编写测试向量，用于驱动 ECC 解码纠错电路工作。从表中的代码可以看到，测试向量基本覆盖了 2bit 和 1bit 错误的所有类型。由于超出 2bit 的错误将无法被 ECC 模块检测到，因此，对于超出 2bit 的错误，本模块一律按 2bit 错误处理。将编写好的 test bench 和源程序导入到 ModelSim 中进行仿真，得到如图 4.11 所示的仿真结果：

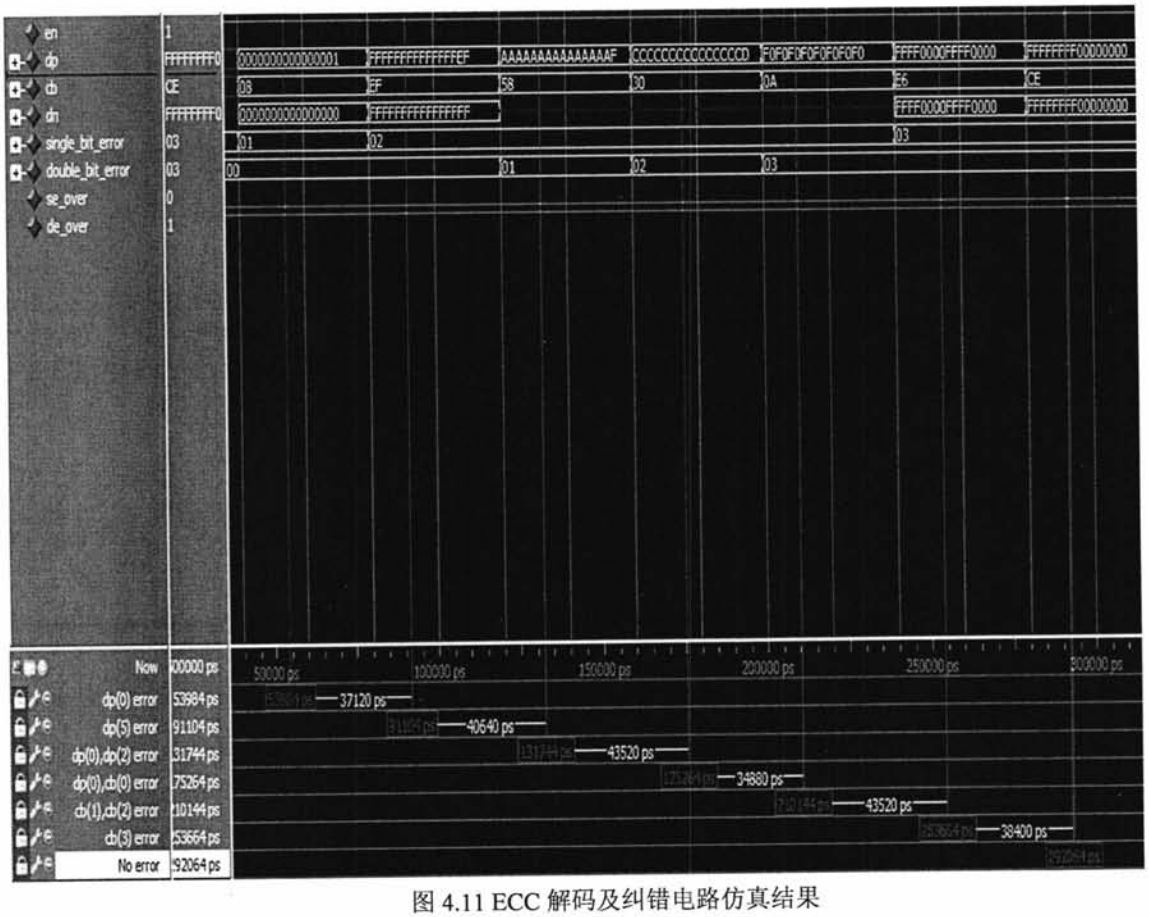


图 4.11 ECC 解码及纠错电路仿真结果

在图中，各信号的定义如表 4.6 所示：

表 4.6 ECC 解码及纠错电路信号定义

| 信号名称 | 类型 | 功能定义 |
|---|---|---|
| en | Input | 使能信号 |
| dp | Input | 输入数据，64bit |
| cb | Input | ECC 码，8bit |
| dn | Output | 输出数据，64bit |
| single_bit_error | Output | 1-bit 错误寄存器，寄存器上限阈值为 3F，超过阈值将产生中断。 |
| double_bit_error | Output | 2-bit 错误寄存器，寄存器上限阈值为 1F，超过阈值将产生中断。 |
| se_over | Output | 1-bit 阈值中断 |
| de_over | Output | 2-bit 阈值中断 |

从图 4.11 中可以看到，72-bit 信息被重新解码为 64-bit。所有数据位发生 1 位错误的数据均被改正过来，并且 single_bit_error 寄存器在每次发生 1-bit 错误时，都会自动计数。所以数据发生 2-bit 错误时，译码器将无法译码出正确数据，故输出为高阻态，见图中的 131744ps 至 210144ps 时间段。在这个时间段，连续发生 3 次 2-bit 错误，double_bit_error 寄存器计数至 03。

### 4.3.5 地址命令解码电路仿真测试

地址命令解码电路的实际工作是将指令仲裁模块发出的指令的表现形式转换为 JEDEC 规范规定的表现形式（也即 cs、cas、ras、we 四种信号的组合表现形式），它的仿真结果如图 4.12 所示。图 4.12 中涉及到的信号定义见表 4.7 所示：

表 4.7 地址命令解码电路信号定义

| 信号名称 | 类型 | 功能定义 |
|---|---|---|
| clk | Input | 时钟信号 |
| command | Input | 命令信号，来自指令仲裁模块的输出 |
| en | Input | 地址命令解码电路使能信号 |
| ba_from_native | Input | Bank 地址信号，来自上层用户 |
| addr_from_native | Input | 行列地址信号，来自上层用户 |
| ba_to_memory | Output | Bank 地址信号，发往地址通路 |
| addr_to_memory | Output | 行列地址信号，发往地址命令通路 |
| cs | Output | 内存模组片选信号，发往地址命令通路 |
| cas | Output | 内存模组列选通信号，发往地址命令通路 |
| ras | Output | 内存模组行选通信号，发往地址命令通路 |
| we | Output | 内存模组读写信号，发往地址命令通路 |

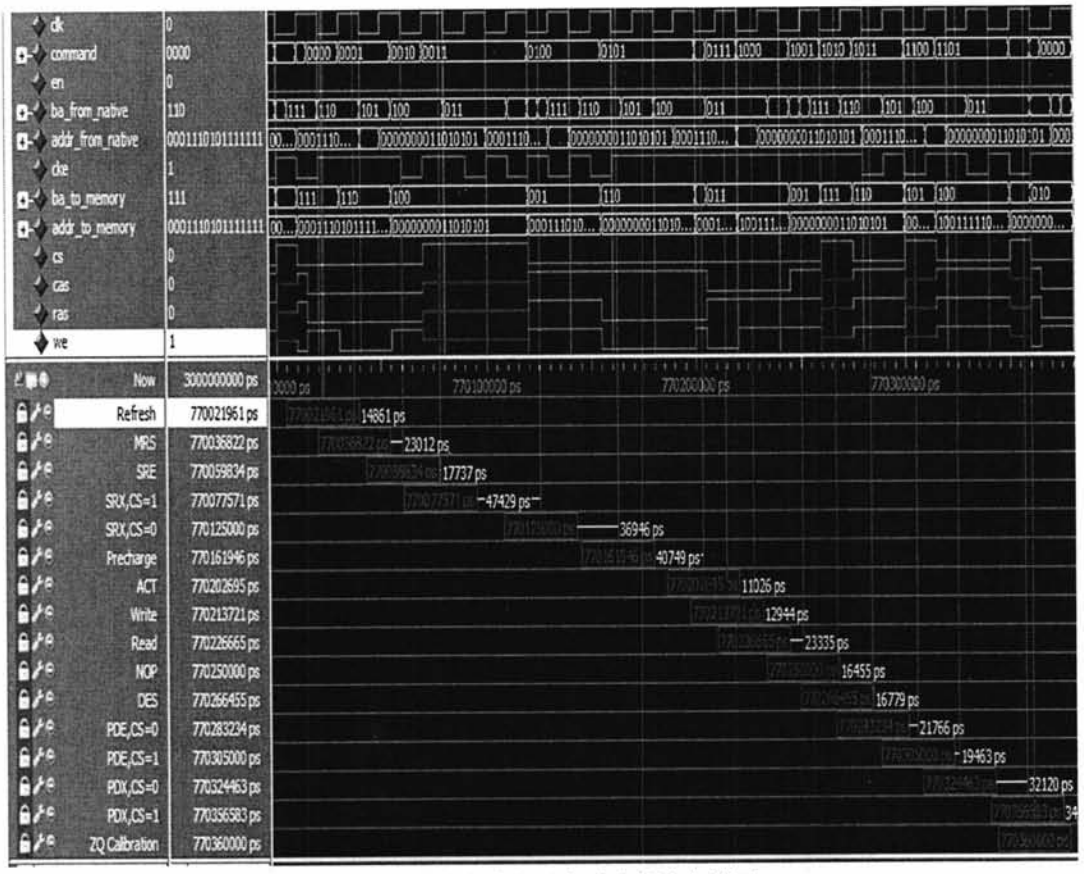


图 4.12 地址命令解码电路功能仿真结果

在图 4.12 中，command 信号发自指令仲裁模块，是经过仲裁调度后的指令，用 4-bit 二进制表示，command 信号经过地址命令解码电路译码后被译码 cs、cas、ras、we 四种信号。这四种信号将通过地址命令通路被发往内存模组。图中各个时刻对应的 command 信号译码后的指令见图的左下部分，共计 16 类指令，已完全涵盖了 DDR3 SDRAM 的所有指令。

# 5 DDR3 内存控制器 IP 核的板级调试及验证

本章介绍如何实现设计的内存控制器对 Stratix IV 开发板上内存模组的控制，从而完成 DDR3 内存控制器 IP 核的原型验证。

## 5.1 硬件验证平台介绍

本课题所设计的 DDR3 内存控制器 IP 核的原型验证，使用的是 Altera 公司的 Stratix IV FPGA 开发板。同时使用 Altera 公司 Quartus II 设计工具和支持 Altera 库的 Mentor Graphics 公司 Modelsim 仿真工具协同完成。图 5.1 是本次原型验证使用的平台[41-42]。

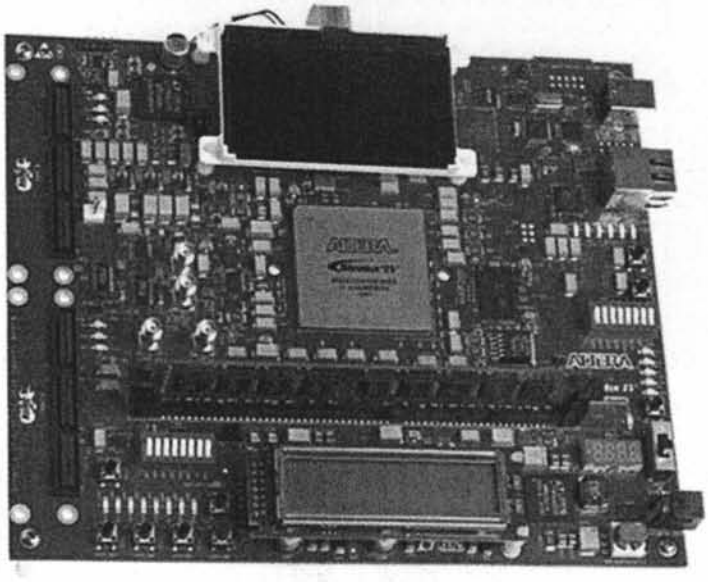

图 5.1 Stratix IV E FPGA 开发板

Stratix IV E FPGA 开发板为基于 Altera Stratix IV E FPGA 的高性能产品开发以及原型验证提供了一个硬件平台。它具有众多的外围接口以及内存接口，此外，它还有两个高速 Mezz 卡（HSMC，High Speed Mezzanine Card）连接器，方便设计者连接 Altrea 第三方合作伙伴的 Mezz 卡以达到扩展功能的作用。开发板的内部结构如图 5.2 所示[41-42]。该开发板具有非常丰富的硬件资源。在本次 FPGA 的验证过程中，主要使用了以下硬件资源：

(1) FPGA 资源：Altera Stratix IV EP4SE530H35。该型号 FPGA 包含 531200 个 LE(Logic Element)，212480 个 ALM(Adaptive Logic Module) ，8 个 PLL，1024 个 18bit ×18bit DSP 乘法器。

(2) I/O 资源：USB Blaster，供下载使用。

(3) 存储器资源：x72 DIMM 插槽，Ramaxel RMS1761EC58E8F-1333 内存条 1 根。

(4) 其他资源：100MHz 晶振。

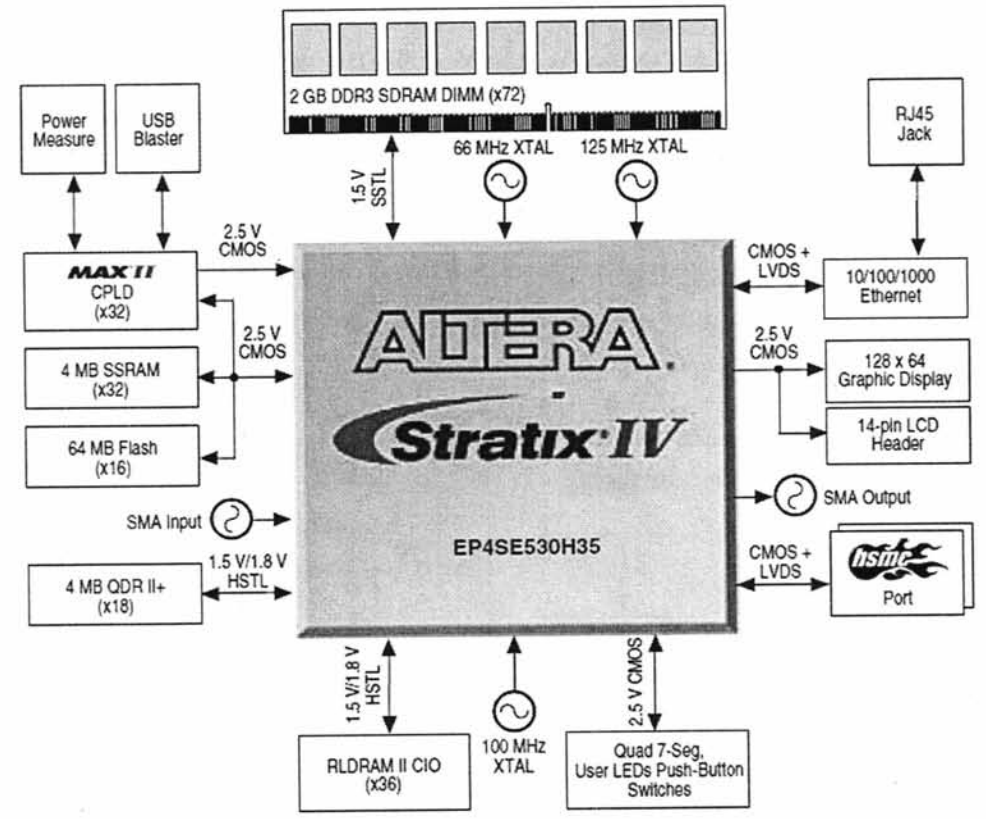


图 5.2 Stratix IV E FPGA 开发板内部结构图

## 5.2 验证方案、流程及结果

### 5.2.1 验证方案介绍

整个控制器 IP 核的板级验证与调试将分三个步骤完成：

(1) 在第四章中，已经在 Quartus 10.1 软件中使用 SOPC Builder 搭建了软件仿真的平台。在软件仿真阶段，使用了基于 DDR3 SDRAM 的内存模组原型进行仿真。在硬件仿真阶段，使用 Ramaxel RMR1761EC58E8F-1333 2GB Unbuffer ECC 内存进行仿真测试。由于该控制器 IP 核具有 ECC 功能，因此，首先将关闭 ECC 功能，对控制器 IP 核进行基本功能的测试，待测试通过后，再开启 ECC 功能，进行一些 ECC 方面的测试。

(2) 为了进一步考量该控制器 IP 核的兼容性以及参数自适应性，需要在完成上述测试之后，再分别更换几种不同规格的内存模组进行基本功能的测试。

(3) 为了验证使用该控制器 IP 核的内存子系统的实际工作性能，需要移植一个 linux 内核到之前已经搭建好的基于 NIOS II 的嵌入式系统中，然后在此 linux 系统中安装内存标准性能测试软件 Stream 进行内存带宽的测试。鉴于论文工作周期及设计复杂度，这一步骤的验证工作在本论文工作中未启动。

### 5.2.2 验证流程介绍

板级验证是基于 FPGA 的 EDA 设计工作的最后一个验证环节，它是在时序仿真的基础上加入了实际的线延迟而进行的仿真验证工作。整个验证流程如图 5.3 所示：

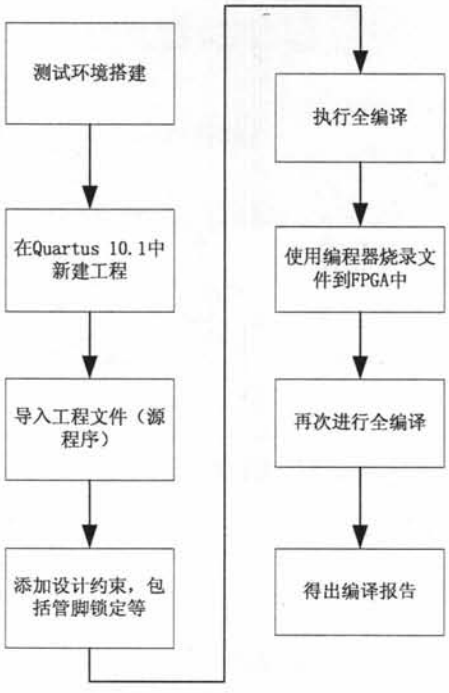


图 5.3 板级验证流程图

### 5.2.3 验证结果及分析

下面将分别介绍各个验证环节的实施细节及结果：

(1) 设计输入

在 Quartus 10.1 中调入之前用于软件仿真的所有工程文件即可。

(2) 设计约束

工程中添加了设计文件后，需要给设计加以时序约束和分配管脚。设计的约束是为了使高速数字电路的设计满足运行速率方面的要求，在综合、布局布线阶段附加约束。时序分析工具是以用户的时序约束判断时序是否满足设计要求的标准，因此要求设计者输入合适的约束，以便得到正确的时序分析报告[41]。分配管脚是将设计文件的输入输出信号指定到 FPGA 器件的某个管脚，设置此管脚的电平标准、电流强度等。分配管脚时，除了仔细将计划分配的管脚和端口连接准确外，还应注意复位、输入和输出等信号的高低电平。否则很可能会导致系统始终处于复位状态，或者 LED 灯亮灭情况与预期的完全相反。完成管脚分配后，执行 Save & Compile，进行编译，即可生成 SOF 文件。

(3) 分析与综合（Analysis &Synthesis）

在 Quartus 10.1 中执行“Start Analysis &Synthesis”分析并综合整个工程。图 5.4 是整个分析综合的结果。综合的基本功能是将使用硬件描述语言设计的代码转换成基本的门电路的逻辑连接关系，也即网表文件，供下一步的布局布线使用。随着 FPGA 越来越复杂、性能要求越来越高，综合在设计流程中也成了一个很重要的部分[40-42]。综合的结果直接影响后续的布局布线，进而影响整个设计的好坏。因此，综合工具的性能将直接影响整个设计的结果，但是好的综合工具的成本也较高。本设计采用折中方案，直接使用 Quartus 10.1 自带的综合工具进行综合，综合也能满足设计的要求。

| | |
|---|---|
| Analysis & Synthesis Status | Successful - Tue Dec 06 15:42:13 2011 |
| Quartus II Version | 10.1 Build 153 11/29/2010 SJ Full Version |
| Revision Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Top-level Entity Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Family | Stratix IV |
| Logic utilization | N/A |
| Combinational ALUTs | 12,816 |
| Memory ALUTs | 405 |
| Dedicated logic registers | 13,969 |
| Total registers | 14683 |
| Total pins | 174 |
| Total virtual pins | 0 |
| Total block memory bits | 111,652 |
| DSP block 18-bit elements | 4 |
| Total GXB Receiver Channel PCS | 0 |
| Total GXB Receiver Channel PMA | 0 |
| Total GXB Transmitter Channel PCS | 0 |
| Total GXB Transmitter Channel PMA | 0 |
| Total PLLs | 1 |
| Total DLLs | 1 |

图 5.4 DDR3 IP 核分析综合结果

(4) 布局布线 (Fitter)

布局布线（Fitter）使用由 Analysis&Synthesis 建立的数据库，将工程的逻辑和时序要求与器件的可用资源相匹配。它将每个逻辑功能分配给最好的逻辑单元位置，进行布线和时序，并选择相应的互连路径和引脚分配[41]。本设计的布局布线的结果如图 5.5 所示：

| | |
|---|---|
| Fitter Status | Successful - Tue Dec 06 15:56:44 2011 |
| Quartus II Version | 10.1 Build 153 11/29/2010 SJ Full Version |
| Revision Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Top-level Entity Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Family | Stratix IV |
| Device | EP4SE530H35C2 |
| Timing Models | Final |
| Logic utilization | 5 % |
| Combinational ALUTs | 12,886 / 424,960 ( 3 % ) |
| Memory ALUTs | 470 / 212,480 ( < 1 % ) |
| Dedicated logic registers | 14,076 / 424,960 ( 3 % ) |
| Total registers | 14790 |
| Total pins | 176 / 744 ( 24 % ) |
| Total virtual pins | 0 |
| Total block memory bits | 110,720 / 21,233,664 ( < 1 % ) |
| DSP block 18-bit elements | 4 / 1,024 ( < 1 % ) |
| Total GXB Receiver Channel PCS | 0 |
| Total GXB Receiver Channel PMA | 0 |
| Total GXB Transmitter Channel PCS | 0 |
| Total GXB Transmitter Channel PMA | 0 |
| Total PLLs | 1 / 8 ( 13 % ) |
| Total DLLs | 1 / 4 ( 25 % ) |

图 5.5 DDR3 IP 核布局布线结果

(5) 装配（Assembler）

装配的目的是根据布局布线的结果生成编程文件 SOF，以方便下载到配置芯片中去。图 5.6 是本设计装配的结果。

| | |
|---|---|
| Assembler Status | Successful - Tue Dec 06 15:57:39 2011 |
| Revision Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Top-level Entity Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Family | Stratix IV |
| Device | EP4SE530H35C2 |

图 5.6 DDR3 IP 核 Assembler 结果

(6) TimeQuest Timing Analysis

TimeQuest Timing Analysis 的目的是为了分析和验证设计的时序是否满足要求。图 5.7 是本设计时序分析的结果：

| | |
|---|---|
| Revision Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Top-level Entity Name | ddr3_Controller_x72_or_x64_design_by_Aden |
| Family | Stratix IV |
| Total Critical Violations | 0 |
| Total High Violations | 3 |
| - Rule R101 | 1 |
| - Rule D101 | 2 |
| Total Medium Violations | 1 |
| - Rule R102 | 1 |
| Total Information only Violations | 364 |
| - Rule T101 | 314 |
| - Rule T102 | 50 |

图 5.7 时序分析结果

(7) 下载配置

首先使用 Quartus 软件对工程文件进行全编译，待编译通过后，就可以对 Startix IV E 器件进行下载配置了。将布局布线后的结果（*.POF 文件）通过 Quartus 自带的 Programmer 下载到开发板上的 FPGA 中去，得到与设计相符的实际硬件电路，对实际的电路用测试系统进行调试，从而验证设计的正确性。

# 结　论

本论文采用自顶向下的设计思路，设计了一个兼容 JEDEC DDR3 SDRAM 标准的内存控制器 IP 核，并搭建了完整的内存控制器仿真验证平台，制定了详细的测试计划，完成了对部分重要子模块的 RTL 级功能仿真工作。然后，在此基础之上，针对 Altera 提出的外部存储器解决方案，完成 ALTMEMPHY 数字接口的设计，提出控制器板级验证方案，并在 Altera Startix IV E FPGA 开发板上完成验证工作。以下是本论文完成的具体工作：

(1) 完成对 DDR3 SDRAM JEDEC 标准 JESD79-3E 研究；

(2) 构建了 DDR3 内存控制器的整体架构；

(3) 完成了 ALTMEMPHY 数字接口的设计；

(4) 搭建了完整的软件仿真平台（Test Bench），完成了对 DDR3 控制器部分子模块的 RTL 级仿真；

(5) 提出了 FPGA 板级验证方案，完成了对设计代码下载、配置以及 FPGA 仿真。

在整个论文设计工作过程中，具有以下创新之处：

(1) 本文设计的 IP 核支持 Unbuffer ECC or Non-ECC 的全系列内存模组，具有内存模组自动识别功能，无需进行参数配置等特点，节省了用户在使用控制器 IP 核时需要配置参数的工作。

(2) 本文设计的 IP 核支持 ALTMEMPHY 数字接口，通用性较好。特别适用于使用 Altera 存储器解决方案的用户。

(3) 本文所设计的 IP 核具备可扩展性。由于该 IP 核是采用 VHDL 语言所设计的 Soft IP 核，因此非常方便进行代码移植复制，从而设计出支持多通道的内存控制器 IP 核。

由于论文工作周期以及 IP 核设计的复杂度等原因，本课题所设计的 IP 核仅仅是一个实验级的 IP 核，与商用的 IP 核相比，还有较大的差距，因此，为了进一步巩固研究成果，提高研究水平，还需要从以下几个方面展开进一步的研究工作：

(1) 所设计的 IP 核后续的板级验证时序优化工作。可以从两个方面入手，一是代码优化，二是架构优化。

(2) 所设计的 DDR3 内存控制器在嵌入式系统中的集成应用研究。可以考虑移植 linux 内核和一些必要的驱动程序，构建一个小型的计算机系统，从系统层面来考量所设计的内存控制器的性能。

(3) 开展对 DDR3 RDIMM 和 LRDIMM 的控制技术的研究。

(4) 开展对内存多通道控制技术的研究。

# 致　谢

时光飞逝，三年的研究生学习生活即将结束。在这三年的研究生学习生涯中，我得到很多老师和同学帮助。在此，谨向所有关心和帮助过我的老师和同学们表示由衷的感谢，特别要感谢我的导师杜丽霞教授。感谢杜老师在学习和生活上对我的亲切关怀和悉心指导，是您的远见卓识和对 EDA 技术发展状况的准确把握，促使我对 EDA 技术的研究，特别是对 EDA 方法的研究，从而最终确定了论文的研究手段和方法；是您的谆谆教诲与严格要求，促使我顺利地完成学位论文。您高深的学术造诣、纵观全局的洞察能力、幽默乐观的生活态度、对事业的不懈追求以及平易近人的处世态度拓宽了我的人生视野，指引我前进的方向!衷心祝愿您身体健康、一生平安! 此外，还要感谢王紫婷教授、吴蓉教授以及所有研究生学习阶段的授课老师们，你们循循善诱的教导，拓宽了我的知识面。感谢实验室的师弟师妹们、宿舍的室友以及共同生活学习的同学们，和你们在一起的时光非常愉快。

非常感谢我的实习单位曙光信息产业（北京）有限公司，特别要感谢产品中心的沙超群、朱越、赵雷以及张迎华等领导的关怀，感谢产品中心所有同事老师们的帮助，正是在你们的影响下，我有幸接触到高性能计算领域，进而将目光聚焦在内存及其控制技术上，从而确定课题研究方向。感谢研发中心李静老师在控制器整体架构设计方面所给予的指导，您的指导和帮助促使我的课题研究得以顺利进行。

最后，在论文完成之际，我要深深地感谢我的父母，你们的关心、支持与理解，永远是我学习的最大动力。在我人生成长的每一个足迹中，都倾注了你们的心血和汗水。谨以此文，向你们表达我难以言表的感激之情。

# 参考文献

[1] 邓丽. 高带宽低延时的 DDR2 内存控制器的研究与实现:(硕士学位论文). 北京:国防科技大学, 2006.

[2] 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.

[3] JEDEC. DDR3 SDRAM Specification JESD79-3E. http://www.jedec.org/standards-documents/docs/jesd-79-3d, 2010.

[4] Shen, Dr.Willian Wu. DDR3 Fuctional Outlook. JEDEX, SanJose, 2007: 28-32.

[5] T. Fukushima, Y. Yamada, H. Kikuchi etc. New Three-Dimensional Integration Technology Based on Reconfigured Wafer-on-Wafer Bonding Technique. IEDM Tech, Washington, DC, 2007: 985-988.

[6] 韦喜波. DDR SDRAM 控制器的设计与验证:(硕士学位论文). 哈尔滨:哈尔滨工业大学, 2009.

[7] 高挺挺. 适应于 DDR SDRAM 和 DDR2 SDRAM 控制器的设计:(硕士学位论文). 合肥:合肥工业大学, 2009.

[8] 张凯, 李云岗. 基于 AMBA 总线的 DDR2 SDRAM 控制器研究与实现. 微电子学与计算机. 2005, 22(9):117-122.

[9] You-Qing Wang, Dong-hua Zhou, Li-Heng Liu. Reliable Memory Feedback Design for a Class of Nonlinear Fuzzy Systems with Time-varying Delay. International Journal of Automation and Computing, Tsinghua University, P.R. China, 2007:169-176.

[10] 汤彦. 片上内存控制器性能评估和优化:(硕士学位论文). 北京:中科院计算研究所, 2006.

[11] 李瑞. 基于 SOC 的存储器 IP 核的分析与设计:(硕士学位论文). 成都:电子科技大学, 2008.

[12] 张宇, 时龙兴, 王学香等. 面向片上系统的高性能 SDRAM 控制器设计. 固体电子学研究与发展. 2007, 27(3):408-413.

[13] 刘培文, 陈祥, 颜伟光. 计算机组装与维护教程. 北京:北京科海电子出版社, 2009.

[14] (美) 布莱恩特(Bryant, R. E), 奥哈拉伦(O' Hallaron, D, R.)著, 龚奕利, 雷迎春译. 深入理解计算机系统. 北京:机械工业出版社, 2010.

[15] M. Morris Mano. Computer System Architecture(计算机系统体系结构第 3 版). 北京:清华大学出版社, 1998.

[16] Pat Conway, Bill Hughes. The AMD Opteron Northbridge Architecture, IEEE Computer Society.2007, 27(2): 10-21.

[17] Radhakrishan S, Chinthamani S, Kai Cheng.The Blackford Northbridge Chipset for the Intel 5000. IEEE Computer Society.2007, 27(2): 22-33.

[18] Jonathan Owen, Maurice Steinman. Northbridge Architecture of AMD Griffin Microprocessor Family. IEEE Computer Society.2008, 28(2): 10-18.

[19] T. Ono, T. Mizukusa, T. Nakamura et al. Three-Dimensional Processor System Fabricated by Wafer Stacking Technology. IEEE COOL Chips V Tech, San Francisco, CA, 2002: 186-193.

[20] Ware F •A, Hampel C. Improving Power and Data Efficiency with Threaded Memory Modules. ICCD, San Jose, CA, 2007:417-424.

[21] 江先阳, 刘新春, 张佩珩等. 计算机密集型体系集成 DDR SDRAM 控制器设计. 计算机工程与科学. 2006, 28(3):96-101.

[22] 陆阳, 王强, 张本宏等. 计算机系统容错技术研究. 计算机工程. 2010, 36(13):230-235.

[23] 吴继华, 王诚. Altera FPGA/CPLD 设计(初级篇). 北京:人民邮电出版社, 2005.

[24] 吴继华, 王诚. Altera FPGA/CPLD 设计(高级篇). 北京:人民邮电出版社, 2005.

[25] 刘延飞, 郭锁利等. 基于 Altera FPGA/CPLD 的电子系统设计及工程实践. 北京:人民邮电出版社, 2009.

[26] 刘欲晓, 方强, 黄宛宁等. EDA 技术与 VHDL 电路开发应用实践. 北京:电子工业出版社, 2009.

[27] Nikolov • H, Stefanov • T etc. Efficient External Memory Interface for Multi-Processor Platforms Realized on FPGA Chips.Conference of Field Programmable Logic and Applications, Amsterdam, 2007: 580-584.

[28] 张永志. DDR2 内存控制器的模块设计和验证平台技术研究:(硕士学位论文). 合肥: 合肥工业大学, 2009.

[29] Xiao-Long Chen, Yun Zhang, Zhi Liu. Design a memory Variable Structure AQM Controller with Input-Delay Estimation. Proceedings of the Seventh International Conference on Machine Learing and Cybernetics, Kunming, 2008:1936-1942.

[30] 艾丽华, 罗四维. 计算机体系结构简明教程. 北京:电子工业出版社, 2008.

[31] 程晓东, 郑为民, 唐志敏. 基于 DDR SDRAM 控制器时序分析的模型. 计算机工程. 2005, 31(17):182-184.

[32] W. Wulf, S. McKee. Hitting the Memory Wall: Implications of the Obvious.ACM Computer Architecture News. 1995, 23(1): 20-24.

[33] Altera Corporation. External Memory Interface Handbook. http://www.altera.com.cn/literature/hb/external-memory/emi.pdf, 2010.

[34] Micron Corporation. DDR3 SDRAM Tech Note, http://www.micron.com/products/dram/ddr3-sdram/ddr3-sdram-tech-notes, 2010.

[35]Altera Corporation. External Memory PHY Interface (ALTMEMPHY) (nonAFI) Megafunction User Guide. http://www.altera.com.cn/literature/ug/ug_altmemphy.pdf, 2010.

[36] Li Zhanghui, Ni Xiaoqiang, Wang Yongwen. Implementation and Desgin Of Error-correcting Codes in High Performance Processor. the 15th Computer Engineering and Industrial Meeting, Beijing, 2011: 65-69.

[37] 舒展. DDR2 控制器 IP 的设计与 FPGA 实现:(硕士学位论文). 合肥:合肥工业大学, 2009.

[38] 冯子军, 肖俊华, 胡伟武. 龙芯 1 号 IP 验证方法. 计算机工程. 2008, 34(5):31-35.

[39] 方苗, 陈泽文, 彭澄廉. SOPC 设计中的用户自定义逻辑. 计算机工程. 2004, 32(17):32-35.

[40] 须文波. DDR2 SDRAM 控制器的 FPGA 实现. 江南大学学报. 2006, 5(2):145-148.

[41] Altera Corporation. Stratix IV E FPGA Deveplopment Board Reference Mannul. http://www.altera.com.cn/literature/manual/rm_sive_fpga_dev_board.pdf, 2010.

[42] Altera Corporation. Stratix IV E FPGA Development Kit User Guide. http://www.altera.com.cn/literature/ug/ug_sive_fpga_dev_kit.pdf, 2010.

# 攻读学位期间的研究成果

[1] 王正宇. 基于 VHDL 的 TLC5615 串行 D/A 转换器设计. 中国科技博览. 2011(16):57-57.

[2] 魏萍, 杜丽霞, 王正宇. 基于 SOPC 的公共区域智能照明监控系统. 自动化与仪器仪表. 2011(5):132-133.

# DDR3内存控制器的IP核设计及FPGA验证

作者：王正宇
学位授予单位：兰州交通大学