上海交通大学

硕士学位论文

基于Spartan-3 FPGA的DDR2 SDRAM存储器接口设计

姓名：陈良明

申请学位级别：硕士

专业：软件工程

指导教师：韩泽耀;高继业

20071128

基于 Spartan-3 FPGA 的 DDR2 SDRAM 存储器接口设计

摘　要

内部存储器（简称内存）负责计算机系统内部数据的中转、存储与读取，作为计算机系统中必不可少的三大件之一（其余的两个是主板与 CPU），它对计算机系统性能至关重要。内存可以说是 CPU 处理数据的“ 大仓库 ”，所有经过 CPU 处理的指令和数据都要经过内存传递到电脑其他配件上，因此内存性能的好坏，直接影响到系统的稳定性和运行性能。

在当今的电子系统设计中，内存被使用得越来越多，并且对内存的要求越来越高。既要求内存读写速度尽可能的快、容量尽可能的大，同时由于竞争的加剧以及利润率的下降，人们希望在保持、甚至提高系统性能的同时也能降低内存产品的成本。面对这种趋势，设计和实现大容量高速读写的内存显得尤为重要。因此，近年来内存产品正经历着从小容量到大容量、从低速到高速的不断变化，从技术上也就有了从 DRAM 到 SDRAM，再到 DDR SDRAM 及 DDR2 SDRAM 等的不断演进。和普通 SDRAM 的接口设计相比，DDR2 SDRAM 存储器在获得大容量和高速率的同时，对存储器的接口设计也提出了更高的要求，其接口设计复杂度也大幅增加。一方面，由于 I/O 块中的资源是有限的，数据多路分解和时钟转换逻辑必须在 FPGA 核心逻辑中实现，设计者可能不得不对接口逻辑进行手工布线以确保临界时序。而另一方面，不得不处理好与 DDR2 接口有关的时序问题（包括温度和电压补偿）。要正确的实现 DDR2 接口需要非常细致的工作，并在提供设计灵活性的同时确保系统性能和可靠性。

本文对通过 Xilinx 的 Spartan3 FPGA 实现 DDR2 内存接口的设计与实现进行了详细阐述。通过 Xilinx FPGA 提供了 I/O 模块和逻辑资源，从而使接口设计变得更简单、更可靠。本设计中对 I/O 模块及其他逻辑在 RTL 代码中进行了配置、严整、执行，并正确连接到 FPGA 上，经过仔细仿真，然后在硬件中验证，以确保存储器接口系统的可靠性。

关键词：内存，双速率内存，第二代双速率内存，片内终结，模式寄存器，扩展模式寄存器，延时锁定回路

DDR2 SDRAM MEMORY INTERFACE BASED ON SPARTAN-3 FPGA

ABSTRACT

The inner memory is in charge of the data transfer,storage and read/write in the computer system, as one of the three most important parts of the computer system (the other two are main-board and CPU), it plays great role in computer systems. Allegorically the inner memory is the “big storehouse” of the CPU data processing, all the instructions and data processed by CPU must be transferred by inner memory to the other parts of computer system. So that the performance of the inner memory will heavily impact the stability and running performance of computer system.

In current electronics system designing, inner memory is used more and more broadly, meanwhile we are expecting more and more from the inner memory. It’s required to be read/written more quickly, larger in capacity and lower cost as the marketing competent is more and more tight, while keeping the same or seeking higher performance. With this trend, the design and implementation of inner memory with large capacity and high speed are very important. In recent years, the inner memory products evolved from small capacity to big capacity, from low speed to high speed. From the technology point of view, it evolved from DRAM to SDRAM and then DDR SDRAM and DDR2 SDRAM. Compared with the normal SDRAM, the interface design of DDR2 SDRAM is much more complicated as the capacity and speed are increased significantly. On the one hand, as the I/O module resource is limited, the data multiplex and clock transferring must be implemented in the core logic bank of FPGA, the designer have to draw schematic by hand to ensure critical clock scheduling for the interface. On the other hand, we have to pay much attention to the time clock issues

(including temperature and voltage compensation) related to the DDR2 interface. To implementing the DDR2 interface correctly, much careful work is needed and the compatibility design is needed to ensure the performance and stability of the system.

The thesis is the implementation of DDR2 memory interface based on Spartan-3 FPGA of Xilinx corporation. Xilinx FPGA provides I/O blocks and logic resources that make the interface design easier and more reliable. The I/O blocks, along with the logic modules are configured, verified, implemented and properly connected to FPGA in the RTL code, carefully simulated and then verified in hardware to ensure a reliable memory interface system.

KEY WORDS: Inner Memory, DDR, DDR2, ODT, MR, EMR, DLL

图片目录

图 1　内存数据速率趋势……3
图 2　DDR2 内部功能模块 ……4
图 3　模式寄存器 ……9
图 4　写恢复时序图 ……11
图 5　CAS 时序图……12
图 6　扩展寄存器 1 ……13
图 7　读延迟时序图 ……15
图 8　写延迟时序图 ……15
图 9　扩展模式寄存器 2……16
图 10 扩展模式寄存器 3……16
图 11 上电命令仿真时序图……17
图 12 预充电命令仿真时序图……17
图 13 读命令时序图 ……18
图 14 读延迟时序图 ……19
图 15 4 位预取读时序图……20
图 16 写命令时序图 ……20
图 17 4 位预取写时序 ……22
图 18 刷新模式 ……23
图 19 刷新命令仿真图 ……23
图 20 FPGA-DDR2 接口连接图……25
图 21 存储器接口分层图……27
图 22 DDR2 SDRAM 存储接口模块……28
图 23 控制器状态机 ……37
图 24 基础方针波形 ……38
图 25 数据通道方针波形……40
图 26 DDR_EN 信号生成仿真图……40
图 27 DDR_DQ_INOUT 信号生成仿真图……41
图 28 输入数据接收仿真图……41

图 29 初始化波形图 ……42
图 30 初始化波形图-局部 ……42
图 31 写数据波形图 ……43
图 32 板极调试波形图 1……43
图 33 板极调试波形图 2……43
图 34 PC 通过 USB 对 DDR2 实现数据采集系统连接图 ……47

表格目录

表 1　DDR2 与 DDR 比较……2
表 2　DDR2 芯片模组接口信号 ……5
表 3　DDR2 内存命令真值表……7
表 4　模式寄存器真值表……10
表 5　上层基本命令 ……32
表 6　控制器内部基本命令……32

符号说明

SDRAM	Synchronous Dynamic random access memory
SDR SDRAM	Single Data Rate SDRAM
DDR SDRAM	Double Data Rate SDRAM
DDR2 SDRAM	Double Data Rate 2nd Generation SDRAM
ODT	On-Die Termination
MR	Mode Register
EMR	Extended Mode Register
IOB	Input/Output Block
CLB	Clock Logic Block
DLL	Delay-Locked Loop
FIFO	First In First Out
DCM	Digital Clock Manager
DMA	Direct Memory Access

论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：陈宝林

日期：2008年 1月18日

论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：陈宏飞　　指导教师签名：韩泽耕

日期：2008年1月18日　　日期：2008年1月19日

1　课题背景介绍及高带宽数据访问模式的研究

内部存储器是用于存放当前待处理的信息和常用信息的半导体芯片。容量不大，但存取迅速。内部存储器包括 RAM、ROM 和 Cache。RAM（随机存取存储器）是电脑系统的主存储器，人们习惯将 RAM 称为内存。RAM 的最大特点是关机或断电数据便会丢失。内存越大的电脑，能同时处理的信息量越大。我们一般用刷新时间评价 RAM 的性能，单位为 ns（纳秒），刷新时间越小存取速度越快。到目前为止出现过的内存规格主要有 FPM、EDO、SDRAM、RDRAM 以及 DDR/DDR2 SDRAM 等。

DDR 的工作原理是其能在控制时钟触发沿的上、下沿都能进行数据传输（而 SDRAM 只在控制时钟的下降沿进行数据传输），命令在时钟的每个正边沿寄存。因此在一次控制信号过程中，DDR SDRAM 能进行两次的数据交换。双向数据选通脉冲 (DQS) 与接收端中的用于采样的数据一起传输。DQS 是一个选通脉冲，在读取期间由 DDR SDRAM 器件传输，在写入期间由控制器传输。DQS 与用于读取的数据边沿对准，与用于写入的数据中心对准。对 DDR SDRAM 器件的读取和写入访问为突发式；访问以激活命令寄存开始，然后是读取或写入命令。在激活命令下寄存的地址位用于选择要访问的组和行。在读取或写入命令下寄存的地址位用于为突发访问选择组和起始列位置。当前，DDR 作为一种双倍数据速率的 SDRAM 存储器，即双通道内存技术（其实是一种内存控制和管理技术，它依赖于芯片组的内存控制器发生作用）[9]，它在理论上能够使两条同等规格内存所提供的带宽增长一倍。它的技术核心在于：芯片组（北桥）可以在两个不同的数据通道上分别寻址、读取数据可以达到 128 位的带宽。

DDR2 SDRAM 器件是 DDR SDRAM 系列的下一代器件。DDR2 SDRAM 器件使用 SSTL 1.8V I/O 标准（DDR SDRAM 是 SSTL 2.5V I/O），并使用 DDR 架构实现高速运行。所以 DDR2 是在 DDR 技术基础上的改进，可以在较低的频率下具有更高的性能，更良好的稳定性、更低的功耗和更低的制造成本。与 DDR 相比，DDR2 最主要的改进是在内存模块速度相同的情况下，可以提供相当于 DDR 内存两倍的带宽。技术上讲，DDR2 内存上仍然只有一个 DDR SDRAM 核心，但是它可以

并行存取，在每次存取中处理 4 个数据而不是两个数据。DDR2 内存另一个改进之处在于，它采用 FBGA 封装方式替代了传统的 TSOP 方式[3]。对于 DDR 和 DDR2 的主要差别可参见下表 1 所示：

表 1　DDR2 与 DDR 比较

Table1　DDR2 Vs DDR

	项目	DDR	DDR2	DDR2 的优势
1	封装	TSOP(66pins)	FBGA only	可满足较高电气性能和速度要求
2	工作电压	2.5 V I/O	1.8 V I/O	可较少内存系统耗电
3	密度	128Mb-1Gb	256Mb-2Gb	可满足更高容量需求
4	内部组数量	4	4 和 8	可满足高性能、大容量内存的需求
5	读预取位数	2	4	数度更快
6	数据速度	200MHz 266MHz 333 MHz 400 MHz	400MHz 533MHz 667 MHz	总线速度更快
7	信号终结	主板电阻岛终结	片内终结(ODT)	ODT 可降低系统消耗，而且更快速有效。
8	数据选通	单端数据选通	差分数据选通	可提高系统时钟余量

20 世纪 90 年代后期，存储器接口从单倍数据速率 (SDR) SDRAM 发展到了双倍数据速率 (DDR) SDRAM，而今天的 DDR2 SDRAM 运行速率已经达到每引脚 667 Mb/s 或更高。当今的趋势显示，这些数据速率可能每四年增加一倍，到 2010 年，随着 DDR3 SDRAM 的出现，很可能超过每引脚 1.2 Gb/s。见下图 1 所示：

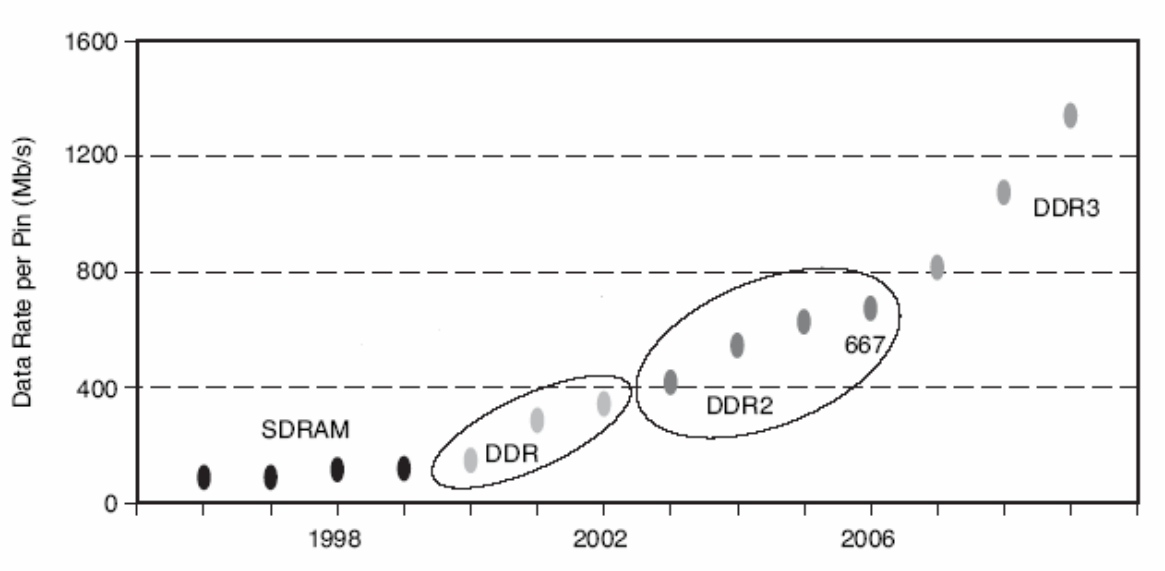


图 1　内存数据速率趋势

Fig1　SDRAM Data Rate Trends

DDR/DDR2 以其高性能、低成本的存储器解决方案，正广泛应用于计算机、手机等电子系统。市场分析表明，在当今所有的电子系统中，超过 50%采用了 DDR/DDR2 存储器，并且预计在接下来的几年中将增长到 80%。

在内存接口实现方面，主要有通过 DSP、单片机、FPGA 实现等几种方式，在实际应用中，DSP 主要用于一些对数据计算、算法实现要求较多的设计中，而且在多任务信号处理系统中，为了提高信号的处理速度，往往要使用几个 DSP 协同工作[16]，为此，必须要解决好几个 DSP 对共享存储器的高速访问问题，因此设计较复杂。并且，DSP 对批量数据的访问一般都是通过启动 DMA 完成[15]，而 DMA 一旦启动，数据的传输就不受 DSP 控制，因此，还要设计专门控制电路和缓冲区来确保高速数据传输的稳定性和可靠性。而通过单片机实现的内存借口，由于受到单片机本身内部资源有限，而且单片机能实现的频率较低，因此只能用于对资源要求不高的低频应用系统中。随着内存容量的不断加大及其速度的不断提高，FPGA 以其丰富的内部逻辑资源和输入输出资源正好满足了电子设计中对大容量、高速度数据读取与存储的需要[1]。

2　DDR2 芯片及相关时序介绍

2.1　芯片介绍

本设计使用的内存芯片是 MICRON 公司的 MT47H32M16 芯片，它的内存模组是 64 位位宽，在缺省设置下，每块内存芯片提供 8 位数据位宽，容量为 64Mbytes，即 512Mbit。内存模组使用共 8 个芯片，共 512Mbytes 容量，位宽为 64bits。这样每个芯片数据容量为 512Mbits， 4 个 logic bank，每次给出数据的位宽为 8bits，每个 logic bank 深度为 16K，宽度为 256x32=8K。这些都是存储量的描述，至于对带宽则为 533MB/s，控制器给出的数据位宽为 8bits，时钟频率为 266Mhz，266*2*8=533MB/s。芯片内部的工作时钟为 266Mhz/2=133Mhz，而数据位宽则扩大一倍为 8*2=16bits。

其详细的内部功能模块描述图[2]如下图 2 所示：

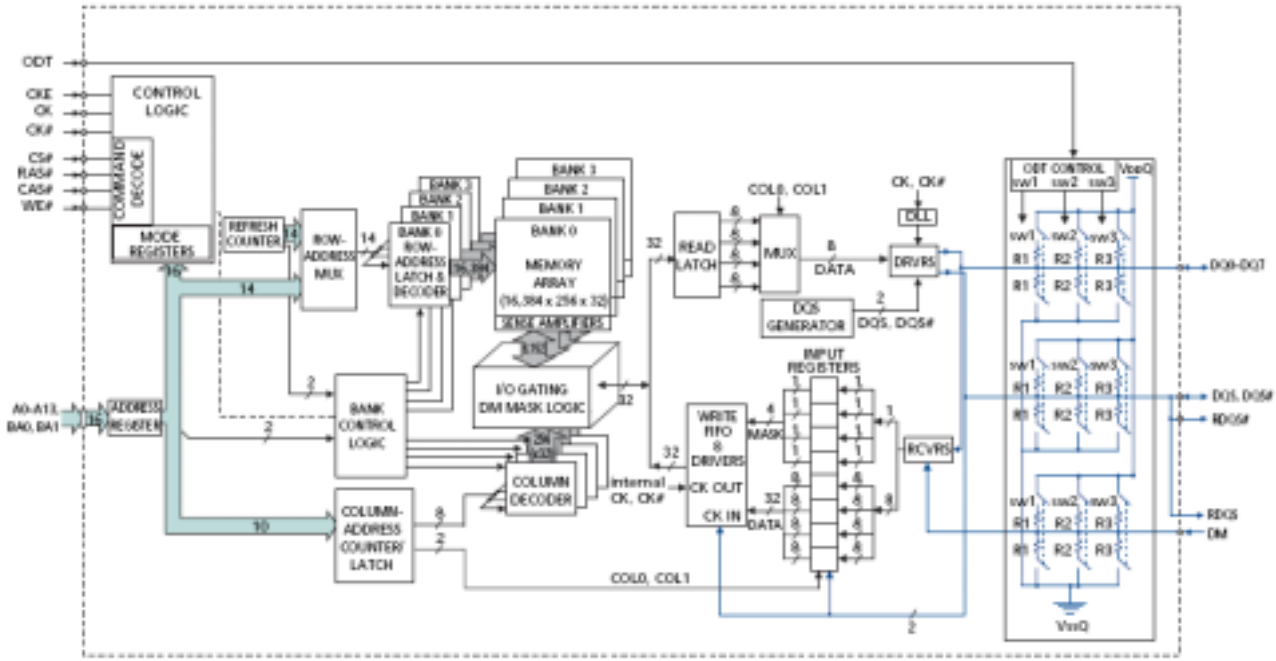


图 2　DDR2 内部功能模块

Fig2　DDR2 functional block diagram

2.2 接口信号说明

每个内存芯片模组的相关接口信号如下表 2 所示:

表 2　DDR2 芯片模组接口信号

Table2　Interface Signals of DDR2 Group

编号	名称	解释
1	CKE	时钟使能
2	CK	时钟输入
3	CK#	反向时钟输入
4	CS#	芯片选择信号
5	RAS#	行访问选择信号
6	CAS#	列访问选择信号
7	WE#	写使能信号
8	BA0	组选择
9	BA1	组选择
10	A0-A13	地址输入
11	DQ0-DQ63	数据输入输出
12	DQS#0-7	数据选通
13	DM0-7	输入数据掩码
14	ODT	片内终结

以上信号最终是连接到模组的内存芯片上去的，下面就一个内存芯片管脚信号进行说明：

▼CKE

时钟使能。高电平激活 DDR2 SDRAM 上时钟电路，低电平禁止该电路。该电路是根据 DDR2 SDRAM 的配置和操作模式来使能或禁止的。

▼CK，CK#

差分时钟输入。所有地址和控制输入信号在 CK 上跳沿及 CK#下跳沿交叉处被采样。输出数据(DQs 和 DQS/DQS#)在 CK 和 CK#交叉处被打出。

▼CS#

芯片选择信号，低电平使能命令解码器，高电平禁止命令解码器。CS#可以被看作是命令码的一部分。

▼RAS#，CAS#，WE#

命令输入 RAS#，CAS#和 WE#（和 CS#一起）定义正在输入的命令。

▼LDM，UDM(DM)

输入数据掩码，DM 是一个针对写数据的掩码信号。在写访问的时候如果 DM 信号为高，则输入数据被掩饰（无效）。DM 在 DQS 的双沿被采样。LDM 是用于 DQ0-DQ7 的数据掩码，UDM 是用于 DQ8-DQ15 的掩码。

▼BA0-BA1

Bank 地址输入，BA0-BA1 定义 ACTIVE，READ，WRITE 或 PRECHARGE 命令作用于哪个 bank 上。在 LOAD MODE 命令中定义是 MR，EMR，EMR（2）和 EMR(3)中的哪一个寄存器。

▼A0-A13

地址输入信号，在 ACTIVE 命令中提供行地址，在 READ/WRITE 命令中提供列地址。在 PRECHARGE 命令中 A10 被采样，低则 PRECHARGE 只应用于一个 bank(BA1-BA0 来选择 bank)，高则应用于所有的 bank。在 LOAD MODE 命令中地址输入提供操作码(op-code)。

▼DQ0-DQ63

数据输入输出，双向数据总线

▼DQS

数据选通(data strobe)，用于源同步操作，读数据时输出，写数据时为输入。读数据时边沿对齐，写数据时中间对齐。

▼DQS#

仅当 LOAD MODE 命令中对差分数据选通模式使能时，DQS#才被使用。

▼RDQS，RDQS#

仅用于 64Megx8 的冗余数据选通，在 EMR 寄存器中设置 RDQS 使能还是禁止。当使能的时候，RDQS 仅作为读数据的输出且在写数据的时候被忽略。RDQS#仅在差分数据选通模式被使能后配合 RDQS 使用。

▼ODT

ODT 即片内终结(On-Die Termination)。所谓的终结，就是让信号在电路的终端被吸收掉，而不会在电路上形成反射，造成对后面信号的影响。在 DDR 时代，控制与数据信号的终结在主板上完成，每块 DDR 主板在 DIMM 槽的旁边都会有一个终结电压岛的设计，它主要由一排终结电阻构成。长期以来，这个电压岛一直是 DDR 主板设计上的一个难点。而 ODT 的出现，则将这个难点消灭了。顾名思义，ODT 就是将终结电阻移植到芯片内部[22]，主板上不再有终结电压岛。ODT 的功能与禁

止由北桥芯片控制，ODT 所终结的信号包括 DQS、RDQS(为 8bit 位宽芯片增设的专用 DQS 读取信号，主要用来简化一个模组中同时使用 4 与 8bit 位宽芯片时的控制设计)、DQ、DM 等。需不需要 ODT 可以由 LOAD MODE 命令来决定。具体的终结操作实现时，首先要确定系统中有几条模组，并因此来决定终结的等效电阻值，有 50 欧姆、75 欧姆和 150 欧姆三档，这在 EMRS 中进行设置。ODT 在很大程度上减少了内存芯片在读取时的 I/O 功率消耗，并简化了主板的设计，降低了主板设计的成本。而且 ODT 也要比主板终结更及时有效[13]，从而也成为了提高信号质量的重要功能，这有助于降低日后 DDR2 进一步提速的难度。

2.3 命令真值表

内存命令真值表如下表 3 所示：

表 3 DDR2 内存命令真值表

Table3 DDR2 command truth table

Function	CKE		CS#	RAS#	CAS#	WE#	BA1 BA0	A13, A12, A11	A10	A9-A0	Notes
	Previous Cycle	Current Cycle									
LOAD MODE	H	H	L	L	L	L	BA	OP Code			2
REFRESH	H	H	L	L	L	H	X	X	X	X	
SELF REFRESH entry	H	L	L	L	L	H	X	X	X	X	
SELF REFRESH exit	L	H	H	X	X	X	X	X	X	X	7
			L	H	H	H					
Single bank PRECHARGE	H	H	L	L	H	L	BA	X	L	X	2
All banks PRECHARGE	H	H	L	L	H	L	X	X	H	X	
Bank activate	H	H	L	L	H	H	BA	Row Address			
WRITE	H	H	L	H	L	L	BA	Column Address	L	Column Address	2, 3
WRITE with auto precharge	H	H	L	H	L	L	BA	Column Address	H	Column Address	2, 3
READ	H	H	L	H	L	H	BA	Column Address	L	Column Address	2, 3
READ with auto precharge	H	H	L	H	L	H	BA	Column Address	H	Column Address	2, 3
NO OPERATION	H	X	L	H	H	H	X	X	X	X	
Device DESELECT	H	X	H	X	X	X	X	X	X	X	
Power-down entry	H	L	H	X	X	X	X	X	X	X	4
			L	H	H	H					
Power-down exit	L	H	H	X	X	X	X	X	X	X	4
			L	H	H	H					

Notes: 1.所有的 DDR2 SDRAM 命令都是由时钟上跳沿时刻的 CS#，RAS#，CAS#，WE 和 CKE 的状态来决定的。

2.BA(BA0-BA1)决定在哪个 bank 上操作。在 LM 命令中 BA 用于选择哪个模式寄存器。

3.BL=4 时的突发读或突发写是不能被中断的。

4.power-down 模式不执行任何 REFRESH 操作，因此 power-down 的持续时间被 refresh 需求限制。

5.ODT 的状态不影响这张表里介绍的状态。ODT 功能在 self refresh 时是不可用的。

6.X 表示 H 或 L

7.SELF REFRESH exit 是异步的。

2.4 寄存器说明

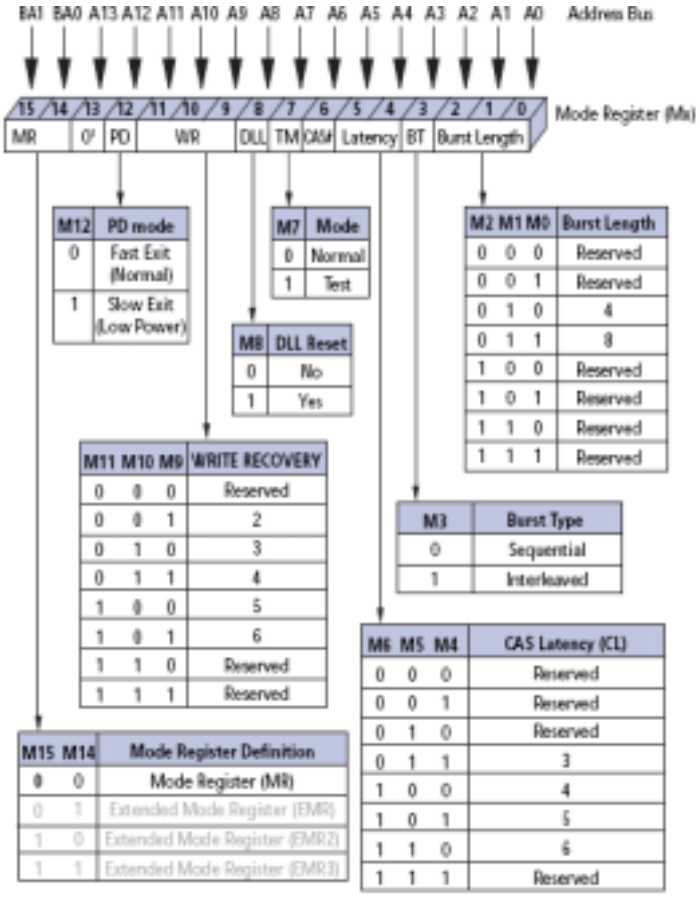
2.4.1 模式寄存器(MR)

模式寄存器(MR)用于定义 DDR2 SDRAM 的特定的操作模式，包括突发长度(burst length)，突发类型(burst type)，CAS 延迟（CAS latency），运行模式（operating mode），延时锁定回路重置（DLL RESET），写恢复（write recovery）和电源关闭（power-down）等模式。

如图 3 所示。模式寄存器中的内容可以通过再次执行 LOAD MODE(LM)命令来改写。如果使用者只是想修改 MR 变量的部分，所有的变量也是需要被编程的（programmed）(M0-M13 for x4 and x8，M0-M12 for x16)。

MR 通过 LM 命令(bits BA1-BA0 = 00)来编程，除去 M8，且其他的比特位(M13-M0 for x4 and x8，M12-M0 for x16)将保存信息直到重新编程或设备掉电，M8 会自清零。对模式寄存器编程不会改变存储单元中的内容。

当所有的 bank 处于 precharged 状态(idle 状态)且没有突发在进行的时候，才能发布 LM 命令。控制器必须等待指定的 tMRD 时间才能开始后续操作如 ACTIVE 命令[12]。违反这些要求中的任何一个，会导致不确定的操作。



Notes: 1. M13 (A13) is reserved for future use and must be programmed to "0." A13 is not used in x16 configuration.
2. Not all listed CL options are supported in any individual speed grade.

图 3　模式寄存器

Fig3　Mode Register

▼Burst Length

Burst Length 由 M0-M3 来定义，如图 3 所示。对 DDR2 SDRAM 的读写操作以突发方式进行，突发长度为 4 或 8。当 READ 或 WRITE 命令发布后，等于突发长度的列 block 将被有效的选中。该突发的所有访问将在该 block 中进行，如果到达了该 block 的边界则 wrap。当 BL=4 的时候，该 block 由 A2-Ai 选中，当 BL=8 的时候，该 block 由 A3-Ai 选中(其中 Ai 是列地址的高位)。剩下的地址（低位地址）用于选择 block 内的起始地址。

▼Burst Type

突发访问有顺序和交织两种方法，通过 M3 来选择。一次突发的访问顺序由 burst length，burst type 及起始列地址来决定，见表 4。

表 4　模式寄存器真值表

Table4　Mode Register Truth Table

Burst Length	Starting Column Address (A2, A1, A0)	Order of Accesses Within a Burst	
		Burst Type = Sequential	Burst Type = Interleaved
4	0 0	0, 1, 2, 3	0, 1, 2, 3
	0 1	1, 2, 3, 0	1, 0, 3, 2
	1 0	2, 3, 0, 1	2, 3, 0, 1
	1 1	3, 0, 1, 2	3, 2, 1, 0
8	0 0 0	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7
	0 0 1	1, 2, 3, 0, 5, 6, 7, 4	1, 0, 3, 2, 5, 4, 7, 6
	0 1 0	2, 3, 0, 1, 6, 7, 4, 5	2, 3, 0, 1, 6, 7, 4, 5
	0 1 1	3, 0, 1, 2, 7, 4, 5, 6	3, 2, 1, 0, 7, 6, 5, 4
	1 0 0	4, 5, 6, 7, 0, 1, 2, 3	4, 5, 6, 7, 0, 1, 2, 3
	1 0 1	5, 6, 7, 4, 1, 2, 3, 0	5, 4, 7, 6, 1, 0, 3, 2
	1 1 0	6, 7, 4, 5, 2, 3, 0, 1	6, 7, 4, 5, 2, 3, 0, 1
	1 1 1	7, 4, 5, 6, 3, 0, 1, 2	7, 6, 5, 4, 3, 2, 1, 0

▼Operating Mode

M7 设为 0 则为普通操作模式，其它的位则按要求设置。M7 设为 1 则为测试式，其它的位则不需要设置，该模式仅给制造商使用。

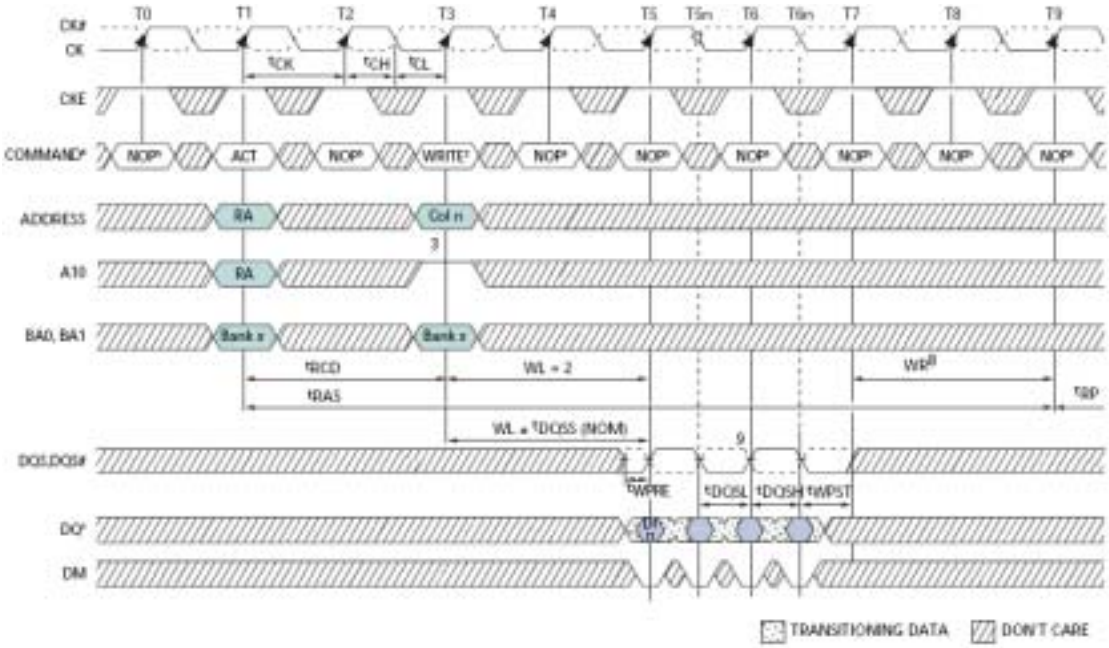
▼DLL RESET

DLL RESET 由 M8 来定义。置 1 则激活 DLL RESET 功能。M8 是自清零的，即在 DLL RESET 功能激活后，该位会自动返回到 0 值。

任何时候，当 DLL RESET 功能被使能，200 个时钟周期后才能发布 READ 命令，这个用于内部时钟和外部时钟的同步。如果没有等到同步的出现，会导致 tAC 或 tDQSCK 参数的冲突。

▼Write Recovery

Write recovery（WR）时间由 M9-M11 定义。WR 寄存器在 WRITE with auto precharge 操作时被使用，在该操作中，DDR2 SDRAM 在上次数据突发后将内部 auto precharge 操作延迟 WR 时钟周期。如图 4 所示：



Notes:
1. DI *n* = data-in from column *n*; subsequent elements are applied in the programmed order.
2. BL = 4, AL = 0, and WL = 2 in the case shown.
3. Enable auto precharge.
4. ACT = ACTIVE, RA = row address, BA = bank address.
5. NOP commands are shown for ease of illustration; other commands may be valid at these times.
6. ^{t}DSH is applicable during ^{t}DQSS (MIN) and is referenced from CK T5 or T6.
7. ^{t}DSS is applicable during ^{t}DQSS (MAX) and is referenced from CK T6 or T7.
8. WR is programmed via MR[11, 10, 9] and is calculated by dividing ^{t}WR (in nanoseconds) by ^{t}CK and rounding up to the next integer value.
9. Subsequent rising DQS signals must align to the clock within ^{t}DQSS.

图 4　写恢复时序图

Fig4　WR Cycle

WR 值可以为 2，3，4，5 或 6，由 M9-M11 编程。使用者需要对 WR 编程，计算方法为 WR[cycles]=tWR[ns]/tCK[ns]。

▼Power-Down Mode

Active power-down(PD)模式由 M12 来定义。为 0 时，standard active PD 模式或“ fast-exit ”active PD 模式被使能。tXARD 参数用作 fast-exit active PD exit 时间。DLL 在该模式下要被使能。

当 M12=1，lower-power active PD 模式或“ slow-exit” active PD 模式被使能。tXARDS 参数用作 fast-exit active PD exit 时间。

▼CAS Latency(CL)

CAS latency(CL)由 M4-M6 来定义。CL 是指 READ command 命令发布后到第一个比特的数据出现在数据线上的时间延迟，单位是时钟周期。CL 根据使用的

速度等级可以设置为 3，4，5 或 6。

DDR2 SDRAM 不支持半个时钟周期的延迟。

DDR2 SDRAM 还支持一个叫做 posted CAS additive latency(AL)的特性。CL=3 和 CL=4 的例子见图 5，假定 AL=0。加入 READ 命令在 clock 边沿 n 处发布，CL 为 m 个时钟周期，在数据将在时钟边沿 n+m 处出现(假定 AL=0)。

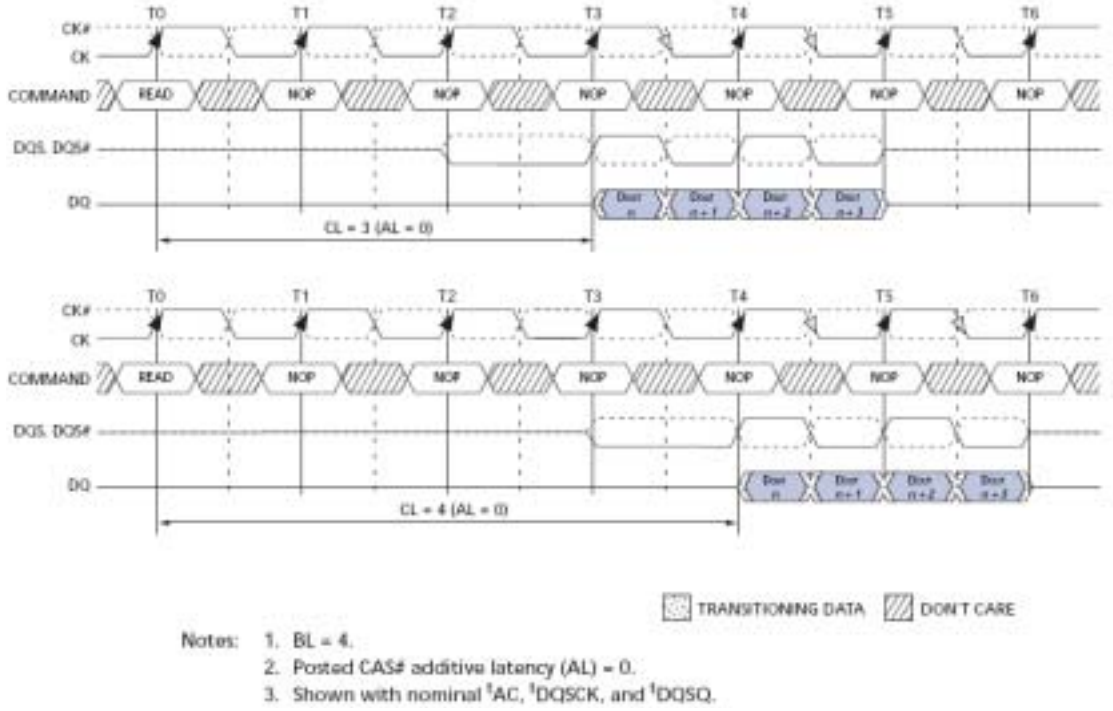


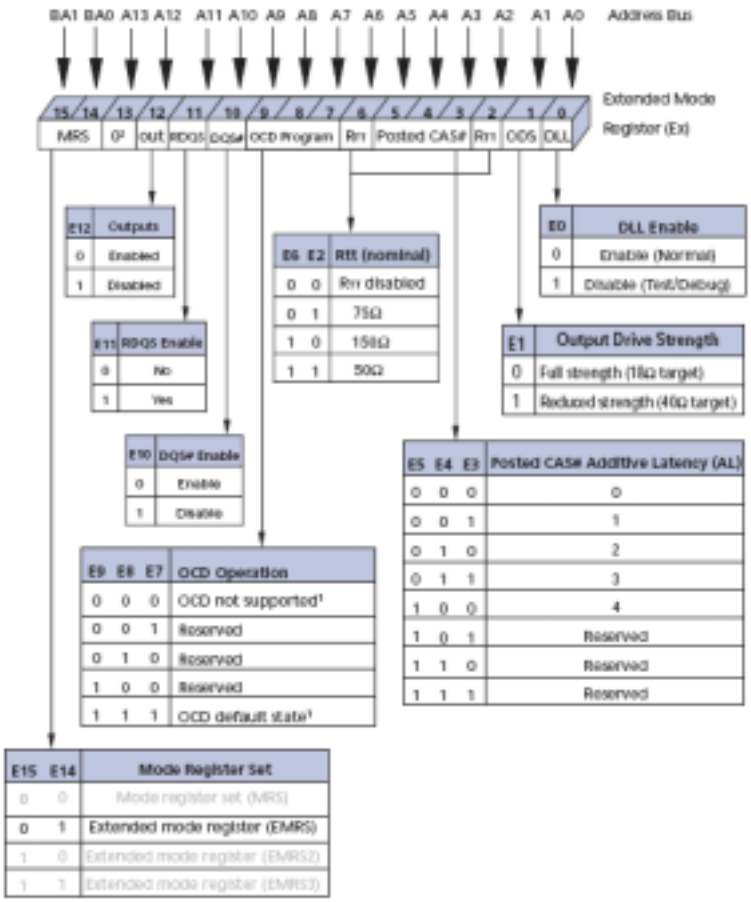
图 5　CAS 时序图

Fig5　CAS Cycle

2.4.2　扩展模式寄存器 1(EMR1)

扩展模式寄存器 1(EMR1)控制模式寄存器控制功能以外的功能，这些额外的功能包括 DLL enable/disable，output drive strength，on-die termination(ODT)(Rtt)，posted AL，off-chip driver impedance calibration(OCD)，DQS# enable/disable，

RDQS/RDQS# enable/disable 和 output disable/enable。这些功能通过图 6 中所示的位来控制。EMR1 通过 LM 命令来编程并一直保存信息直到再次编程或设备掉电。再次编程 EMR1 不会改变存储单元内容。



Notes: 1. During initialization, all three bits must be set to "1" for OCD default state, then must be set to "0" before initialization is finished, as detailed in the notes on pages 17–18.
2. E13 (A13) is not used on the x16 configuration.

图 6　扩展寄存器 1

Fig6 EMR1

▼DLL Enable/Disable

DLL 可以在 LM 命令中通过使能或禁止 E0 来编程。在 normal operation 中 DLL 必须被使能。在 power-up initialization 过程中，及出于诊断或评估目的禁止 DLL 后重新返回 normal operation 时候必须将 DLL 使能。通过 LM 命令复位 DLL 后必须将 DLL 使能。

当进入 SELF REFRESH 操作时 DLL 自动被禁止，并在退出 SELF REFRESH 操作后自动被使能后复位。任何时候 DLL 被使能，在 READ 命令发布前必须有 200 个时钟周期的时间延迟，用于内部时钟与外部时钟的同步。没有同步的话会导致 tAC 或 tDQSCK 参数的冲突。

▼Output Drive Strength

Output Drive Strength 由 E1 来定义。对所有输出而言普通的驱动强度为 SSTL_18。设定 E=0 则为所有输出选择普通的驱动强度(最高强度)；设定 E=1 则将所有输出的强度减少为 SSTL_18 的 60%。这个选择用于对轻负载的支持及点对点的环境中。

▼DQS# Enable/Disable

DQS#由 E10 来使能。当 E10=0，DQS#与 DQS 构成差分信号。当被禁止(E10=1)，DQS 用作单端模式，DQS#被禁止，需要被浮空。该功能也用于对 RDQS#的使能与禁止。如果 RDQS 被使能(E11=1)且 DQS#被使能(E10=0)，则 DQS#和 RDQS#也将被使能。

▼RDQS Enable/Disable

RDQS 焊球被 E11 使能。该功能仅用于 x8 的配置。当被使能(E11=1)，在 READ 期间 RDQS 在功能和时序上等同于 DQS。在写期间，RDQS 被 DDR2 SDRAM 忽略。

▼Output Enable/Disable

OUTPUT ENABLE 功能被 E12 来定义。当被使能(E12=0)，所有的输出(DQs，DQS，DQS#，RDQS，RDQS#)正常工作。当被禁止(E12=1)，所有的 DDR2 SDRAM 输出(DQs，DQS，DQS#，RDQS，RDQS#)将被禁止，因此将去掉输出缓存的电流。

▼On-Die Termination(ODT)

ODT 有效电阻 RTT(EFF)由 EMR 的 E2 和 E6 来定义。通过让 DDR2 SDRAM 控制器独立地为任何或所有的设备打开或关闭 ODT，ODT 功能能够提高存储通道的信号完整性。RTT 有效电阻有 50 欧姆，75 欧姆和 150 欧姆三种选择，应用与 DQ，DQS/DQS#，RDQS/RDQS#，UDQS/UDQS#，LDQS/LDQS#，DM 和 UDM/LDM 信号上。（E6，E2）决定通过打开或关闭“ sw1”，“ sw2 ”或“ sw3 ”来决定 ODT 阻值。ODT 仅用于 active，power-down(fast-exit 和 slow-exit 模式)和 precharge，power-down 操作模式。在进入 self refresh 之前 ODT 必须被关闭。在 DDR2 SDRAM 在 power-up 和 initialization 期间，ODT 应该被禁止直到发布 EMR 命令来使能 ODT 功能。任何时候当 EMR 使能 ODT 功能，在 EMR 被使能后过 8 个时钟周期 ODT 才能被驱动为高。

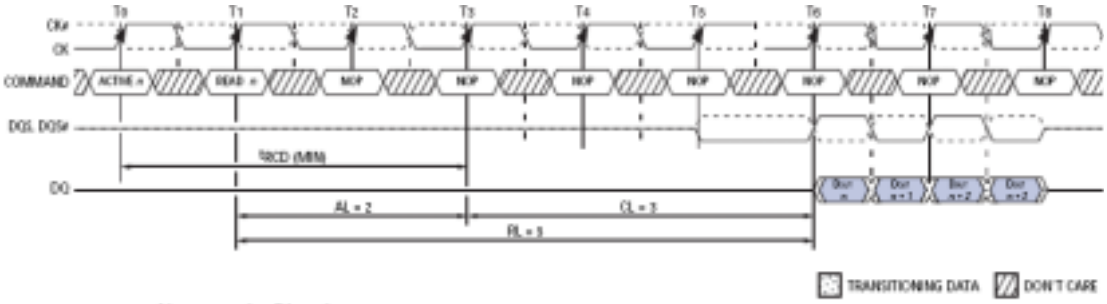
▼Off-Chip Driver(OCD) Impedance Calibration

OFF-CHIP DRIVER 功能不再被支持，一定要设为默认状态。

▼Posted CAS Additive Latency(AL)

Posted CAS additive latency(AL)是命令和数据总线在 DDR2 SDRAM 可以忍受的带宽内有效。E3-E5 定义了 AL 的值，可以为 0，1，2，3 或 4。不能使用保留状态，它为未知操作，会导致不一致的结果。在该操作中，DDR2 SDRAM 允许 READ 或

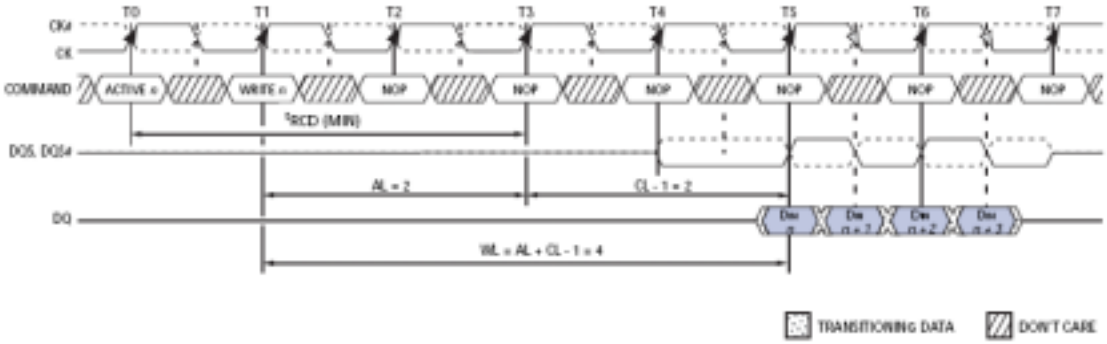
WRITE 命令在 tRCD(MIN)之前发布（要求 AL<=tRCD(MIN)）。该功能的一个典型的应用是设置 AL=tRCD(MIN)-1xtCK。READ 或 WRITE 命令在发布到 DDR2 SDRAM 设备前保留 AL 时间。RL 是 AL 和 CL 之和，RL=AL+CL。WL(write Latency）等于 RL 减去一个时钟周期，WL=AL+CL-1xtCK。RL 的例子见图 7；WL 的例子见图 8。



Notes: 1. BL = 4.
2. Shown with nominal tAC, tDQSCK, and tDQSQ.
3. CL = 3.
4. AL = 2.
5. RL = AL +CL = 5.

图 7　读延迟时序图

Fig7 READ Latency Cycle



Notes: 1. BL = 4.
2. CL = 3.
3. AL = 2.
4. WL = AL + CL - 1 = 4.

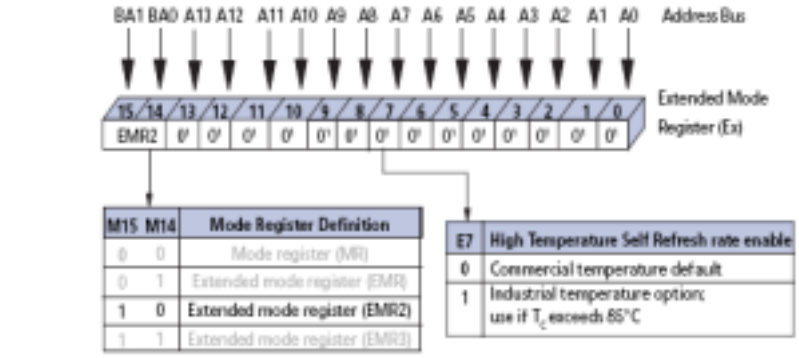
图 8　写延迟时序图

Fig8 Write Latency Cycle

2.4.3　扩展模式寄存器 2(EMR2)

扩展模式寄存器 2(EMR2)控制模式寄存器除外的功能。目前为止所有在 EMR2 中的比特位都为保留位，除了 E7，该比特位作为商用或高温操作的区分。EMR2 通过 LM 命令来编程并保留存储信息直到再次编程或掉电。再次编程不会改变存储内

容。在 IT 设备中 E7(A7)必须编程为 1 以提供更快的刷新率，如果 TCASE 操作 85°C。



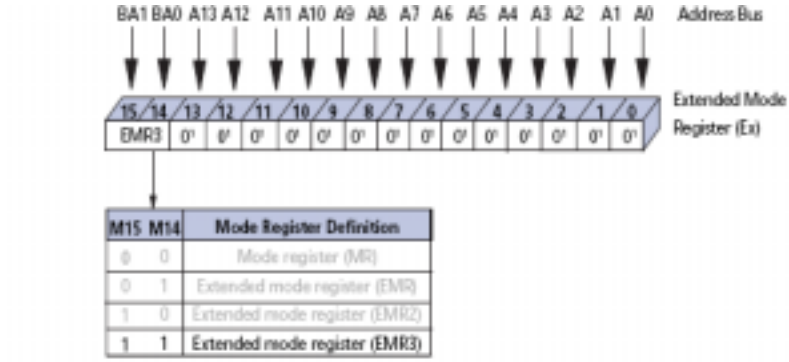
Notes: 1. E13 (A13)–E8 (A8) and E6 (A6)–E0 (A0) are reserved for future use and must all be programmed to "0." A13 is not used in x16 configuration.

图 9　扩展模式寄存器 2

Fig9　EMR2

2.4.4　扩展模式寄存器 3(EMR3)

扩展模式寄存器 3(EMR3)控制模式寄存器除外的功能。目前为止所有在 EMR3 中的比特位都为保留位。



Notes: 1. E13 (A13)–E0 (A0) are reserved for future use and must all be programmed to "0." A13 is not used in x16 configuration.

图 10 扩展模式寄存器 3

Fig10　EMR3

2.5 基本命令时序图

2.5.1 上电命令(power_up)

系统上电后，电源和时钟经过 200 微秒的稳定时间后，通过空操作命令（NOP）来拉高时钟使能信号（CKE），从而使其处于有效状态。在 CKE 信号有效后，再等待 400ns 后[8]，通过预充电命令（precharge all）对模式寄存器进行初始化操作。

上电命令时序图如下所示：

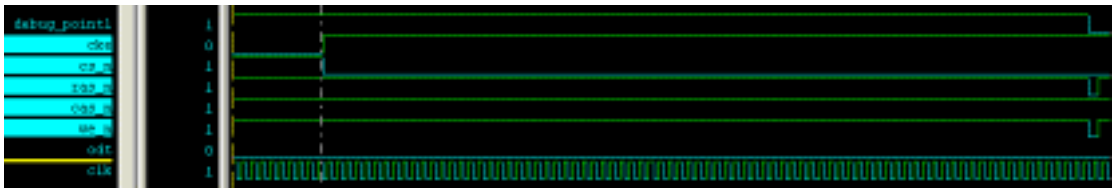


图 11 上电命令仿真时序图

Fig11 Power Up Waveform

2.5.2 预充电命令(precharge all)

预充电命令用于取消特定组中活动行的活动状态。在预充电命令发出的一定时间 (tRP) 后，此组在接下来的行激活命令中可以使用。输入 A10 确定组（一个或全部）是否需要预充电。

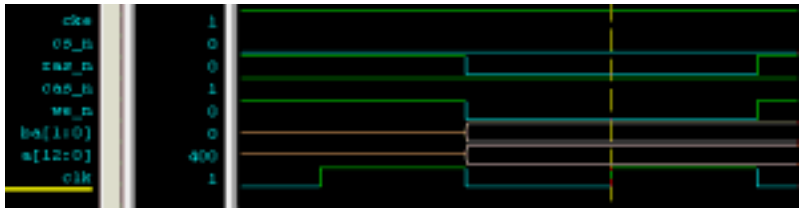


图 12 预充电命令仿真时序图

Fig12 Precharge Waveform

2.5.3 读数据命令(read)

读命令用于发起对激活行的读访问。BA1-BA0 值选择 bank，A0-Ai(其中 i=A9 用于 x16，A9 用于 x8 或 A9，A11 用于 x4)选择开始列位置。A10 上的值用于决定是否使用 auto precharge。如果选中了 auto precharge，读突发结束时将对正在被访问的行 precharge；如果没有选中，则该行仍然打开供后续访问。

▼READ Operation(读操作)

读突发由 READ 命令发起，如图 13 所示。开始列和 bank 地址由 READ 命令提

供，auto precharge 可以被使能或禁止。如果 auto precharge 被使能，在突发结束的时候正在被访问的行会自动被刷新；如果被禁止，则突发结束后该行将仍处于打开状态。

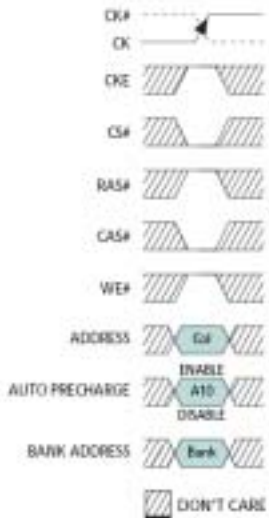


图 13 读命令时序图

Fig13 Read Cycle

读突发期间，开始列地址处的有效输出数据将在 READ latency(RL)个时钟周期后可得。RL=AL+CL。AL 和 CL 的值通过 MR 和 EMR 命令来编程。随后的每个输出数据将依次在随后的时钟上跳或下降沿给出(即在随后的 CK 和 CK#交叉处)。图 14 显示了不同的 AL 和 CL 设置下的 RL。

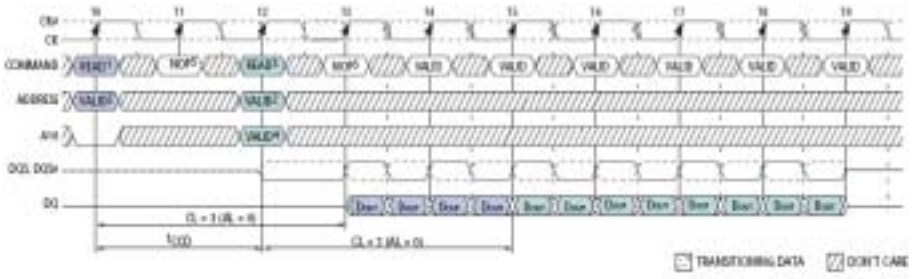
Notes: 1. DO *n* = data-out from column *n*.
2. BL = 4.
3. Three subsequent elements of data-out appear in the programmed order following DO *n*.
4. Shown with nominal ^{t}AC, ^{t}DQSCK, and ^{t}DQSQ.

图 14 读延迟时序图

Fig14 Read Cycle

DQS/DQS#由 DDR2 SDRAM 随输出数据一道给出。DQS 的初始低状态和 DQS#的高状态是读预同步(read preamble)(tRPRE) [9]。最后的输出数据单元后的 DQS 的低状态和 DQS#的高状态称为 read postamble(tRPST)。突发结束后，假定没有其它命令被发起，DQ 将进入高阻状态。

DDR2 SDRAM 使用 BL=4 的时候不允许被中断或截断。一旦 BL=4 的读命令发布，则一定要完成全部的读突发。然而，使用 BL=8 的读操作(auto precharge 被禁止)可以只被另外一个读突发中断或截断，只要该中断出现在 4-bit 边界，因为 DDR2 SDRAM 的 4 位预取结构。如图 15 所示。



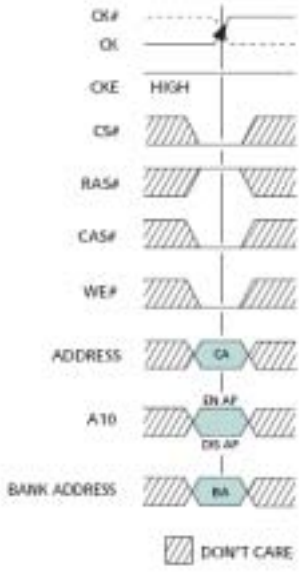
Notes:
1. BL = 8 required; auto precharge must be disabled (A10 = LOW).
2. READ command can be issued to any valid bank and row address (READ command at T0 and T2 can be either same bank or different bank).
3. Interrupting READ command must be issued exactly 2 x ^{t}CK from previous READ.
4. Auto precharge can be either enabled (A10 = HIGH) or disabled (A10 = LOW) by the interrupting READ command.
5. NOP or COMMAND INHIBIT commands are valid. PRECHARGE command cannot be issued to banks used for READs at T0 and T2.
6. Example shown uses AL = 0; CL = 3, BL = 8, shown with nominal ^{t}AC, ^{t}DQSCK, and ^{t}DQSQ.

图 15 4 位预取读时序图

Fig15 4bit-prefetch Read Cycle

2.5.4 写数据命令(Write)

WRITE 命令用于发起对一个激活行的突发写访问。BA1-BA0 选择 bank，地址信号 A0-i(其中 i=A9 for x8 and x16;A9，A11 for x4)用于选择开始访问的列地址。A10 则用于决定是否自动预充电(auto precharge)。如果选择了 auto precharge，则在写突发结束时正在被访问的行将被预充电;如果没有选择 auto precharge，该行将仍然保持打开状态给后续的访问使用。写命令的时序图见图 16。



Note: CA = column address; BA = bank address; EN AP = enable auto precharge; and DIS AP = disable auto precharge.

图 16 写命令时序图

Fig16 Write Command Cycle

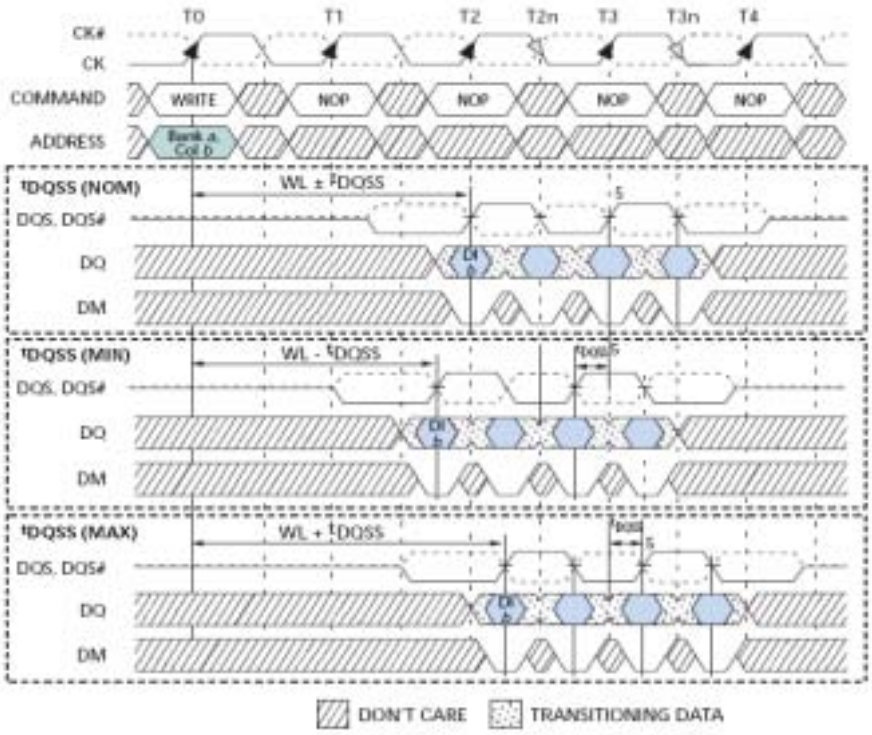
出现在 DQ 上的输入数据根据相应的 DM 的值逻辑电平来决定是否写入 memory。假如相应的 DM 为低，则数据将被写入 memory；如果 DM 为高，则数据将被忽略，不会在相应的列位置执行相应的写操作。

▼WRITE Operation(写操作)

写突发由 WRITE 命令发起，如图 17 所示。WL=RL – 1CK = AL + (CL – 1CK)。开始的列及 bank 地址由 WRITE 命令提供，且在该访问中 auto precharge 可以使能也可以禁止。如果 auto precharge 被使能，则在突发结束时当前被访问的列将被 prech-arged。对一般的用于后面叙述的 WRITE 命令，auto precharge 将被禁止。

在写突发中，第一个有效的数据单元将在跟随 WRITE 命令后面的 DQS[17]的第一个上升沿处被打出，随后的数据将在连续的 DQS 跳变沿被打出。在 WRITE 命令和 DQS 第一个上升沿之间的 DQS 的低状态称为写预同步(write preamble);而在最后一个数据单元后面的 DQS 的低状态被称为写后同步(write postamble)。

WRITE 命令和第一个 DQS 上升沿的时间间隔为 WL±tDQSS。随后的 DQS 上跳沿就被同步了，相对于相关的时钟跳变沿，±tDQSS 的变化。tDQSS 被指定了一个相对宽泛的范围(一个时钟周期的 25%)。



Notes:
1. DI *b* = data-in for column *b*.
2. Three subsequent elements of data-in are applied in the programmed order following DI *b*.
3. Shown with BL = 4, AL = 0, CL = 3; thus, WL = 2.
4. A10 is LOW with the WRITE command (auto precharge is disabled).
5. Subsequent rising DQS signals must align to the clock within tDQSS.

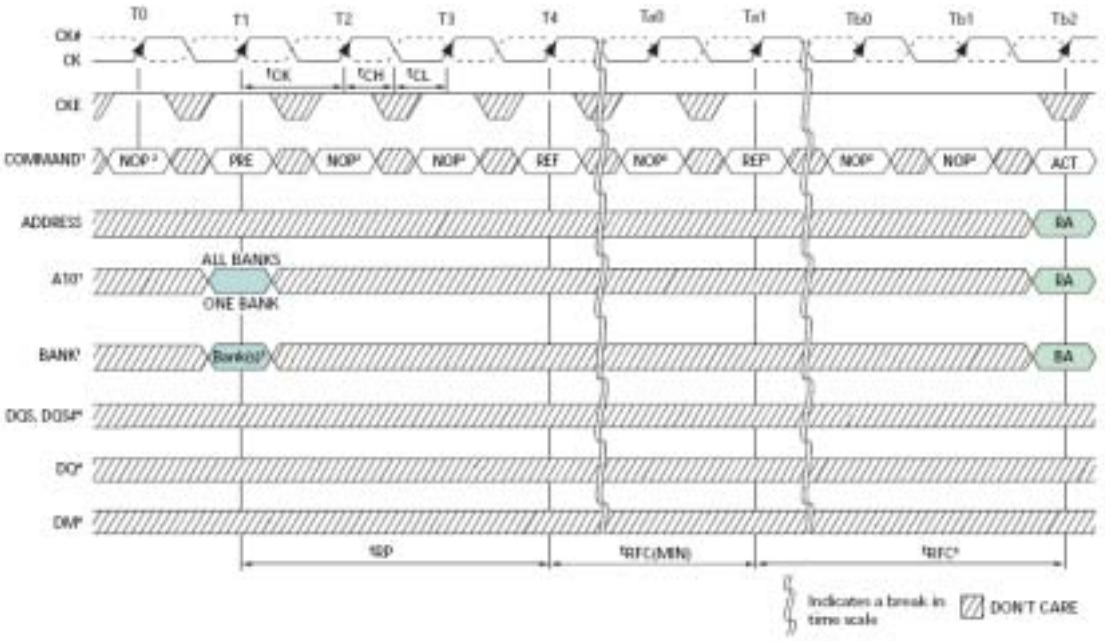
图17 4位预取写时序

Fig17 4Bits-prefetch Write Cycle

2.5.5 刷新命令(REFRESH)

▼REFRESH 命令

REFRESH 用于 DDR2 SDRAM 的普通操作中，类似于 CAS#-Before-RAS#(CBR)REFRESH。该命令不能持久，所以它必须在每次需要 refresh 的时候发布。地址由内部刷新控制器产生，这使得在 REFRESH 命令中地址位为“ Don’t Care ”。512Mb DDR2 SDRAM 每隔 7.8125us(MAX)需要一个 REFRESH 周期。为了提高进度安排和任务间切换的效率，在所提供的绝对刷新间隔内有些灵活性。对任何一个 DDR2 SDRAM 最多可以发布 8 个 REFRESH 命令(来推迟发布 REFRESH 命令)，这意味着 REFRESH 命令键最大的绝对间隔是 9x7.8125us(70.3us;高温操作为 3.9us) [18]。刷新周期在 REFRESH 命令登记时开始，在 tRFC(MIN)后结束。



Notes:
1. PRE = PRECHARGE, ACT = ACTIVE, AR = REFRESH, RA = row address, BA = bank address.
2. NOP commands are shown for ease of illustration; other valid commands may be possible at these times. CKE must be active during clock positive transitions.
3. "Don't Care" if A10 is HIGH at this point; A10 must be HIGH if more than one bank is active (i.e., must precharge all active banks).
4. DM, DQ, and DQS signals are all "Don't Care"/High-Z for operations shown.
5. The second REFRESH is not required and is only shown as an example of two back-to-back REFRESH commands.

图 18 刷新模式

Fig18 Refresh Mode Cycle

刷新命令相关 verilog 代码如下：

```
debug_point3 =1;
refresh;
debug_point3 =0;
nop       (trfc-1);
debug_point3 =1;
```

刷新命令仿真波形如下图所示：

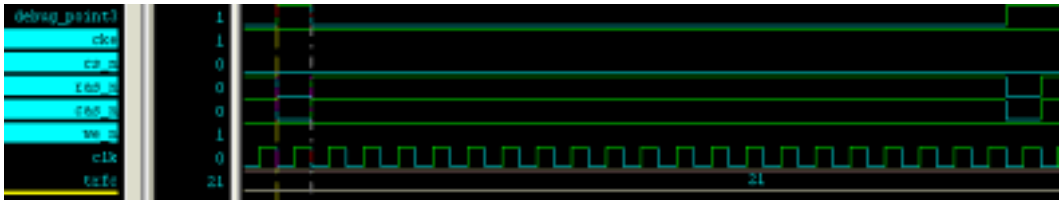


图 19 刷新命令仿真图

Fig19 Refresh Waveform

3 DDR2 接口设计代码介绍

3.1 整体介绍

本设计采用 Xilinx 的 spartan3 FPGA 中一款，具体型号为 xc3s4000-4fg900，实现对 DDR2_400_512M DIMM 内存模组的控制，该模组接口部分工作频率 200Mhz，存储容量为 512Mbytes，位宽为 64bits[4]。包含 8 个 DDR2 400 芯片，每个芯片容量为 512Mbits，4 个 logic bank，每次给出数据的位宽为 8bits，每个 logic bank 深度为 16K，宽度为 256x32=8K(bits)。具体的接口连接如下图所示：

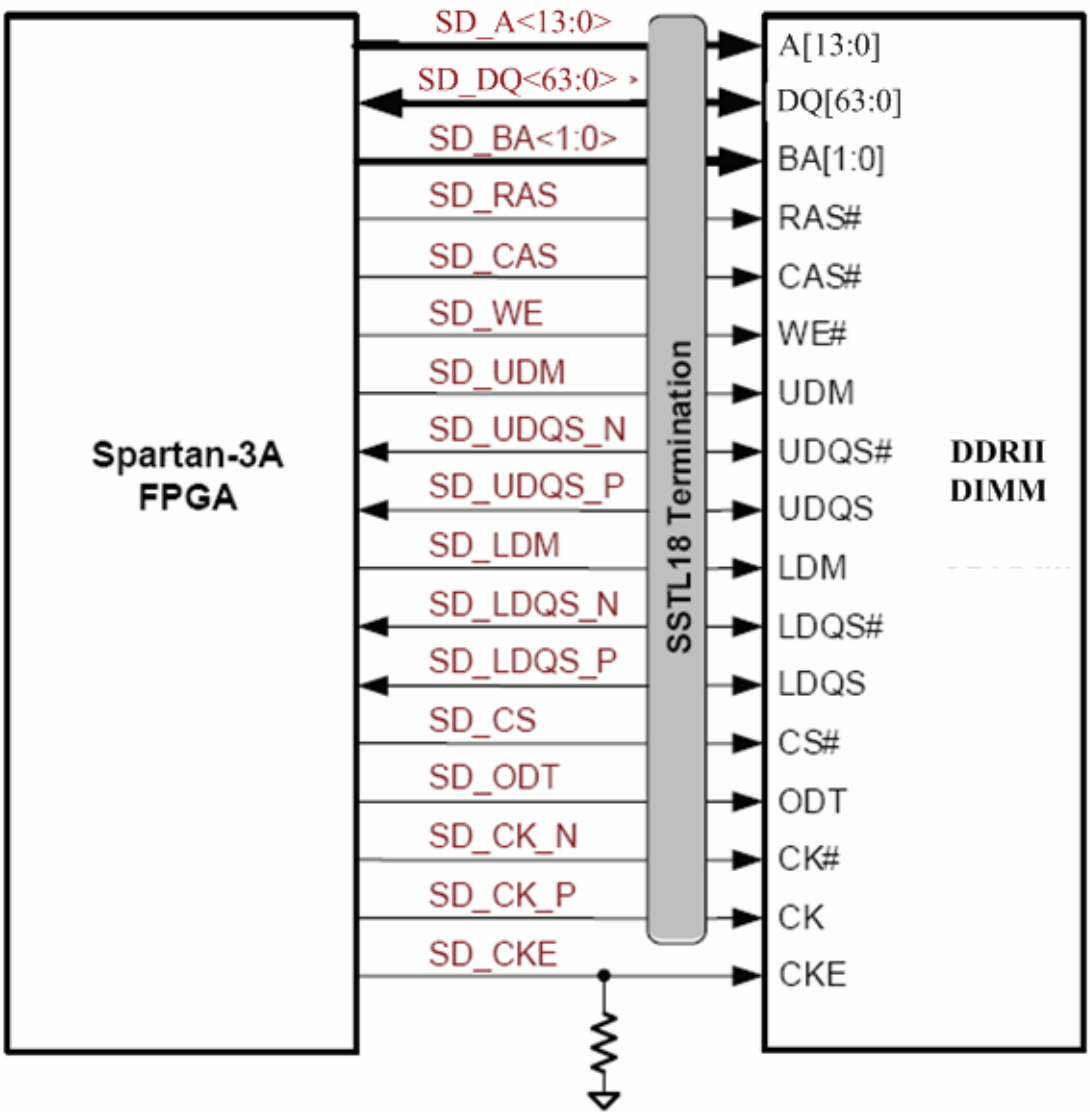


图 20 FPGA-DDR2 接口连接图

Fig20 FPGA-DDR2 Interface diagram

与内存模组的接口信号主要有

ddr2_clk0 和 ddr2_clk0b	输出，接到内存条上去的差分时钟信号
ddr2_cke	输出，时钟使能信号，高有效
ddr2_cs0b	输出，片选信号，选中某个模组，低有效
ddr2_rasb	输出，行选择，低有效
ddr2_casb	输出，列选择，低有效
ddr2_web	输出，写信号，低有效
ddr2_address[13:0]	输出，地址信号

ddr2_ba[1:0]	输出，bank 地址
ddr2_dm[7:0]	输出，数据掩码，1bit 对应一个字节数据
ddr2_ODT0	输出，片内总结开关信号，高打开
ddr2_dq[63:0]	双向信号，数据信号，每个芯片 8 根线，共 8 颗芯片
ddr2_dqs[7:0]	双向信号，数据选通信号，每个芯片一根线
	与上层应用的接口
reset_in	输入，同步复位信号，低电平有效
clk_4n	输入，内存控制器工作时钟，200Mhz
DDR2_MEM_INIT	输入，DDR2 芯片初始化信号，一个时钟周期的高脉冲
refresh_req	输入，刷新请求，刷新的处理和其他命令不同，不是由 u_cmd 命令提供，因为 u_cmd 只有一个时钟周期，容易漏掉刷新
refreshed	输出，刷新应答，刷新请求已经被处理后给出该 信号
basic_cmd[3:0]	输入，应用层发起的对 DDR2 操作的单个命令，每个总线周期发送一个命令
basic_tsk_bank[1:0]	输入，和命令配合的相关数据，bank
basic_tsk_row[12:0]	输入，和命令配合的相关数据，行
basic_tsk_col[9:0]	输入，和命令配合的相关数据，列
basic_tsk_ap	输入，和命令配合的相关数据，是否自动预充电[5] (auto precharge，和 write，read 命令配合)
basic_tsk_dq[255:0]	输入，和命令配合的相关数据，写入数据，4 次突发(burst)，每次 64bits，共 256bits

在具体设计中将 DDR2 SDRAM 存储接口分层以简化设计并使设计模块化。图 21 显示了层次化的存储接口。共三层，分别为应用层，实现层和物理层。

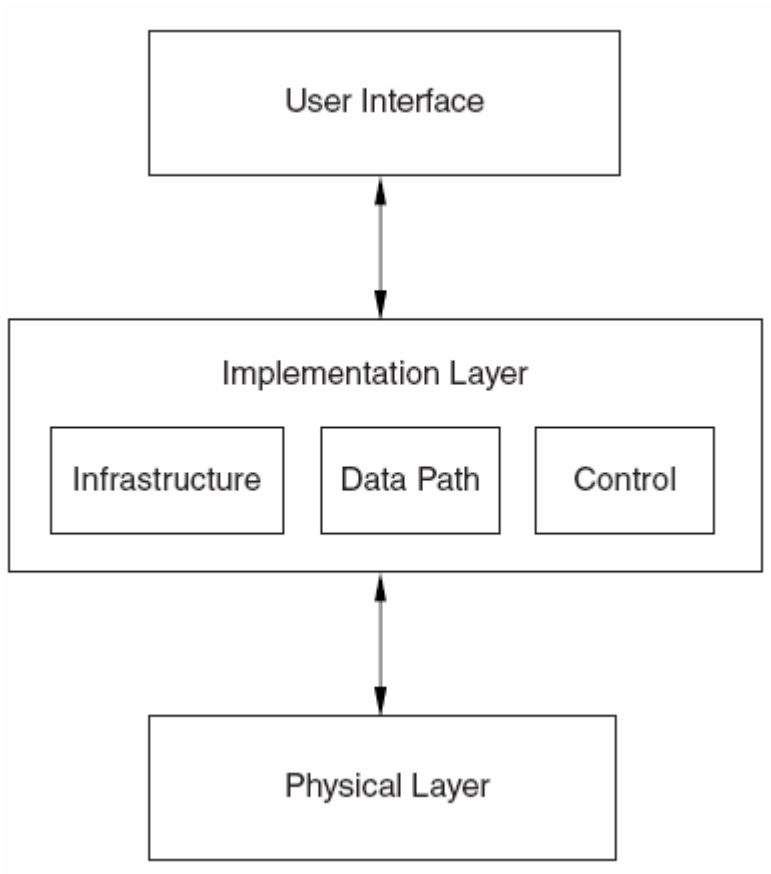


图 21 存储器接口分层图

Fig21 Interface Layering Model

DDR2 SDRAM 控制器模块

图 22 是 Spartan-3 DDR2 SDRAM 存储接口方块图。图中的所有 4 个方框是 ddr2_top 模块的子模块，每个模块的功能将在后面解释。

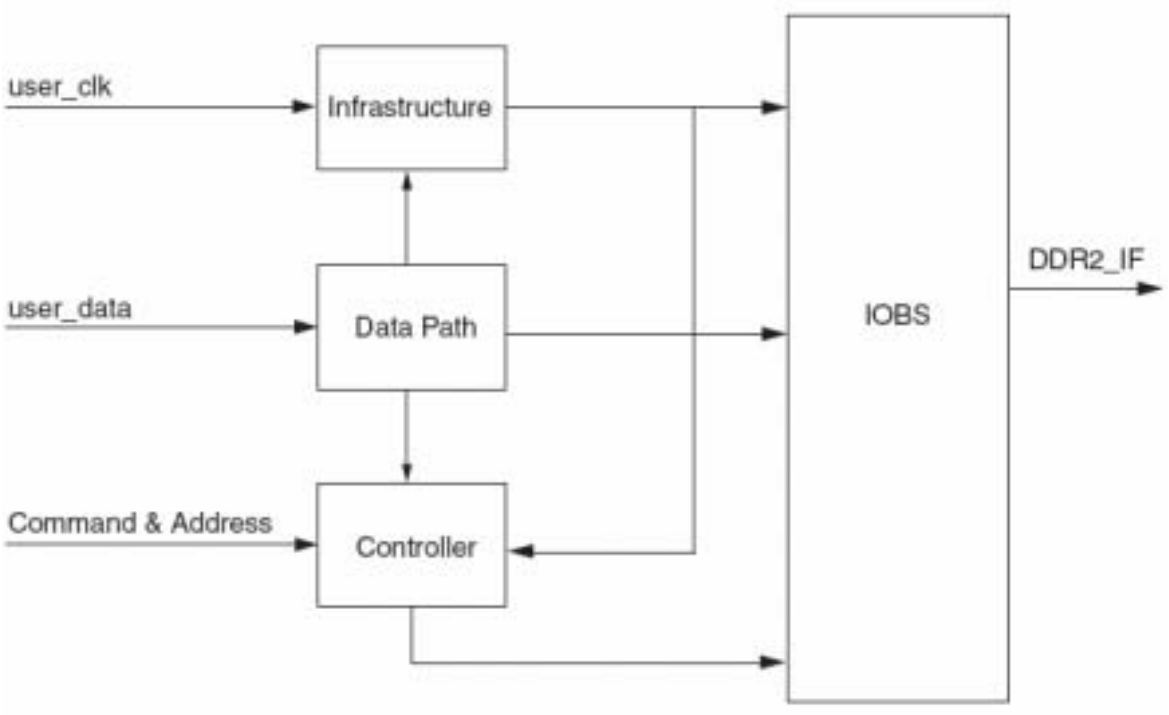


图 22 DDR2 SDRAM 存储接口模块

Fig22 DDR2 SDRAM Interface Modules

3.1.1 控制器(Controller)

控制器的设计是基于 XAPP253 设计(可综合 400Mb/s DDR SDRAM 控制器)，但作了修改以适应 DDR2 SDRAM 的变化。该设计支持突发长度 4，CAS latencies 为 3 或 4。该设计作了修改来实现 DDR2 SDRAM 写延迟特征。该控制器在 Load Mode 命令中初始化 EMR(2)和 EMR(3)寄存器并产生差分 data strobes。

控制器接收用户的命令，对命令解码并对 DDR2 SDRAM 产生读、写和刷新命令。该控制器也产生其它模块所需信号。可参考 XAPP253 以获得更详细的设计和时序分析。

3.1.2 数据路径(Data Path)

数据路径模块负责对存储器发送或接收数据。主要功能包括:

1)往 memory 中写数据

2)从 memory 中读数据

3)将读取数据从 memory 时钟域送往 FPGA 时钟域。

数据写和读的捕获技术的描述可以参见 XAPP768c，Spartan-3 器件和 166Mhz 或 333Mhz DDR SDRAM 存储器接口。写数据和选通(strobe)由 FPGA 发出，在写的时候选通的中部和数据对齐。对 DDR2 SDRAM，选通不是自由运行。为了满足上述要求，使用去往 memory 的基本时钟的 90°或 270°相移时钟将写数据打出[11]，而数据选通信号则使用去往 memory 的基本时钟打出。

Memory 读数据和源同步时钟边沿对齐。DDR2 SDRAM 时钟是非自由运行选通。使用这个非自由运行选通来接收数据并将其发送到 FPGA 时钟域。数据的输入端使用类似于选通输入端的资源。这能确保数据和选通信号的匹配延迟，直到选通在选通延迟电路中被延迟。

3.1.3 基础(Infrastructure)

基础模块产生 FPGA 的时钟和复位信号。数字时钟管理(DCM)用于产生时钟及其反相时钟。一个延迟校准电路也应用于该模块。

延迟校准[6]电路用于选择延迟单元的数目，用于根据读数据延迟选通线。延迟校准电路计算各方面都完全等同于选通延迟电路的一个电路的延迟。延迟的各个方面都被考虑进了校准，包括所有的部件和路由延迟。校准电路在一个给定的时间内选择延迟单元数目。校准完成后，assert 延迟电路的选择线。参考 XAPP768 获得详细延迟校准信息。

3.1.4 输入输出端口模块(IOBS)

所有的 FPGA 输入输出信号在 IOBS 模块中实现。所有的地址和控制信号 are registered 进入 IOBS 模块或从 IOBS 模块输出。

3.2 设计层次结构

tb_spartan3_ddr2
. activate
. ddr2_top(ddr2_top)
. . . ddr2(ddr2)
. . . . controller(controller)
. . . . . ODT_ACK_REG_INST1(FD)
. . . . . rst_calib0(FD)

. . . . . rst_iob_out(FD)
. . . . data_path(data_path)
. . . . . data_path_rst0(data_path_rst)
. . . . . data_read0(data_read)
. . . . . data_read_controller0(data_read_controller)
. . . . . data_write0(data_write)
. . . . infrastructure(infrastructure)
. . . . . cal_top0(cal_top)
. . . . . clk_dcm0(clk_dcm)
. . . . iobs(iobs)
. . . . . controller_iobs0(controller_iobs)
. . . . . datapath_iobs0(data_path_iobs)
. . . . . infrastructure_iobs0(infrastructure_iobs)
. . . . ddr2_cmd_parser(ddr2_cmd_parser)
. . . . . data_construct(data_construct)
. micron_ddr2_testmodule(ddr2_testmodule)

3.3 模块介绍

tb_spartan3_ddr.v 位整体仿真的最上层文件，ddr2_testmodule.v 为 micron 公司提供的仿真模型。ddr2_cmd_parser.v 为 DDR2 的仿真提供发起一个过程，包括上电，初始化，写，读，刷新等，这些命令送给 ddr2 模块，后者完成相应的动作。原来的设计只有一个连续的操作过程且固定不可变，为上电，初始化，然后就是不停地写读若干次后停止，且写入的数据和地址都是固定的。我要将在此设计的基础上进行修改，做到可以发布基本的命令，如激活(active)、不操作(nop)、写(write)、读(read)、刷新(refresh)、预充电(precharge)、odt 打开(odt on)和 odt 关闭(odt off)。这些基本的操作命令可以任意封装组合。

DDR2 的每个命令都是一个时钟周期内完成的，这里要求做到每个时钟周期 ddr2 的动作都是被控制的。

外部对 ddr2_top 的控制是通过 basic_cmd[3:0]、basic_tsk_bank[BA_BITS-1:0]、basic_tsk_col[COL_BITS-1:0] 、 basic_tsk_ap 、 basic_tsk_dq[8*DQ_BITS-1:0] 和 basic_tsk_row[ROW_BITS-1:0]来完成的。原来这些信号是通过 SYS_CLK 打出，而

设计内部处理时钟为 clk，两者都是 167Mhz，但由于没有相关性[7]，会出现问题，现在决定外部用 bus_clk(266/4Mhz)的时钟来打出这些信号，然后在内部先用 clk 时钟将这些信号识别出来，以后大的设计中 clk 由 bus_clk 四倍频得到。

由于 dqs，dq 等信号必须依赖原有设计，否则工作量太大了，而原有的 controller.v 中的状态机只是按照固定的流程跑完一遍然后停止，仅仅起到演示的作用，不具备使用性，必须作修改，这就是问题的难点。所以只能在 controller.v 模块中增加内容或修改内容，而不能任意删减内容，修改时一定要小心。

3.3.1 DDR2 命令解释器(ddr2_cmd_parser.v)

命令解释器的功能是负责解释上层模块传递过来的对内存操作的 basic 命令，将其翻译成 controller 模块所能认识的基本命令形式[19]。同时完成地址及数据的构造工作，将上层软件送过来的数据构造成 u_data_i，送给 data_path.data_write 模块使用，将行地址，列地址，自动预充电(ap)信号统一组织为地址信号 u_address 送给 controller 模块使用。

接口信号为:

basic_cmd[3:0]	输入，上层软件下达的基本命令
basic_tsk_bank[1:0]	见上节
basic_tsk_col[9:0]	见上节
basic_tsk_ap	见上节
basic_tsk_dq[255:0]	见上节
basic_tsk_row[12:0]	见上节
init_val	输入，初始化完成信号，即 controller.v 给出的 init_done
power_up_ok	DDR2 power_up 工作完成，由 controller.v 给出
u_cmd[3:0]	输出，用户命令，送给 controller.v
u_data_i[255:0]	输出，用户数据，送给 data_path.data_write.v
task_bank[1:0]	输出，bank 号，送给 controller.v
u_address[23:0]	用户地址，送给 controller.v
u_config_parms1	用户配置参数，送给 controller.v，DDR2 初始化所需的参数
u_config_parms2	用户配置参数，送给 controller.v，DDR2 初始化所 需的参数

basic_cmd 命令具体解释

表 5　上层基本命令

Table5　Basic Command From Uper Level

basic_cmd	描述
4’b0001	ACTIVE_NOW，激活命令
4’b0010	NOP_NOW，空操作
4’b0011	REFRESH_NOW，刷新命令
4’b0100	WRITE_NOW，写命令
4’b0101	READ_NOW，读命令
4’b0110	PRECHARGE_NOW，预充电命令
4’b0111	ODT_ON_NOW，ODT 打开命令
4’b1000	ODT_OFF_NOW，ODT 关闭命令
4’b1011	LOAD_MODE，配置模式寄存器命令

u_cmd 命令解释

表 6　控制器内部基本命令

Table6　Basic Command In Controller

u_cmd	描述
4’b1011	load_mode，配置模式寄存器命令
4’b0001	power_up，上电命令
4’b0010	init，初始化命令
4’b0100	active，激活命令
4’b0101	nop，空操作
4’b0110	write，写命令
4’b0111	read，写命令
4’b1000	precharge，预充电命令
4’b1001	odt_on， odt 打开命令
4’b1010	odt_off，odt 关闭命令

该模块代码的核心是 next_state 状态机，该状态机接受基本的命令及 init_val，power_up_ok 命令，根据这些命令信号而转入不同的状态中去。根据这些状态构造出 u_cmd 命令信号。该模块包含一个子模块 data_construct-数据构造模块， 这个模块用来对数据信号及地址信号进行构造。其输入信号有 u_cmd，task_dq，task_row，task_col，task_ap，输出信号则为 u_data_i，u_address。根据 u_cmd 命令，在相应的时刻给出所需要的数据及地址信号。对数据信号而言，其实就是直接将 task_dq 信号给出即可[10]，送到 data_write 模块。对地址信号而言，在 u_cmd 为 active 时给 task_row 信号(行地址);为 precharge 命令的时候给{13'b0，task_ap，10'b0}信号;为 write 或 read 时给列地址

▼端口信号说明

clk	166Mhz 工作时钟
clk90	clk 工作时钟相移 90 度的时钟
rst	复位信号
rst180	相差 180 度的复位信号
rst90	相差 90 度的复位信号
cmd_ack	
cnt_roll	每次写或读完成，该信号有两个时钟周期的拉高
dip1	1 允许 DDR2 被使用，0 禁止 DDR2 被使用
dly_tc	一个时钟周期的退出 dly 状态标志信号
r_w	即为 r_w_dly 中的 read_write 信号，周期性高低转换，变化时刻为退出 dly 状态时刻
refresh_done	一个时钟周期的拉高信号，auto_refresh 结束标志信号
init_val	即 controller.v 中的 INIT_DONE 信号，初始化完成后便老高且维持高电平
power_up_ok	一个时钟周期的拉高信号，上电(power up)结束标志
u_data_val	
addr_inc	地址增加信号，在写读周期内维持高电平
addr_rst	一个时钟周期的高电平，在刚进入写读周期后拉高，地址复位信号
u_cmd	用户命令，上电、初始化、读、写、刷新等命令，这些命令送达 controller.v 模块，启动 ddr2 相应动作。

u_ODT_ack

dly_inc　　dly 状态中延迟计数使能信号，在该状态中维持高电平

▼详细说明

将对 DDR2 进行怎样的操作完全由 next_state 状态机决定，且固定不可变，具体为：

rst_state->power_up_start->power_up->init_start->init->wr->dly->auto_ref_start->auto_ref->rlfsr->rd->dly->auto_ref_start->auto_ref->rlfsr->wr->dly……..->rd->dly->auto_ref_start->auto_ref->rlfsr->rst_state

其中 wr 到 rd，再到 wr，最后再到 rd，然后进入 rst_state

这里是后续需要修改的部分，要改成和 512Mb_ddr2 设计中的风格一样，将相关命令新等进一步独立并封装，以便于自由调用，当然还有做成可综合。

这里的写及读与协议要求的想去甚远，需要按照协议要求的进行改造，暂时不考虑 burst 情况，一个一个的地址来写。

修改后的设计如下:

上电后自动完成 rst_state->power_up_start->power_up->init_start->int 过程，ddr2.controller 会对相应的过程做出完成应答，对 power_up_start 的应答为 power_up_ok_dly，得到该应答后状态机由 power_up 状态转入 init_start 状态，而相应的应答为 init_done 信号，得到了应答后便回到 rdy_state 状态，等待来自外部的命令了。

这些基本命令由端口信号 basic_cmd[3:0]通过上层模块给出，目前做了 6 个，分别为 active_now、nop_now、refresh now、write_ap_now、read_ap_now、precharge_now、odt_on_now 和 odt_off_now (由基本命令组合起来的组合命令将在上层的文件中得到体现[14])。状态机不会连续动作，而是每个命令得到一个动作，每个动作完成后都会回到 rdy_state 状态，这样写便于上层的自由控制。这里没有加写寄存器的命令，寄存器在上电的时候写完，以后便固定不可更改。

动作流程为，首先是 basic_cmd 来基本命令信号，送给 next_state1 状态机，进入相应的状态，从而 cmd_p[10:0]得到相应的值，这个值延迟一拍给 cmd[10:0]，由该信号得到 u_cmd_p[3:0]信号，延迟一拍得到 u_cmd[3:0]，送给 ddr2.v 模块，模式寄存器配置数据直接在该模块生成并送到 ddr2.v 模块中使用，分别为 u_config_parms1 和 u_config_parms2。

3.3.2　控制器(controller.v)

DDR2 控制器的功能是完成 ddr2_cmd_parser 过来的命令的进一步解释，将其转换为内存模组所能识别的控制及地址信号。DDR2 SDRAM 器件支持的最小突发长度为 4，每隔一个时钟周期请求一个命令。突发长度为 4 时，控制器在每个控制器时钟（慢时钟）周期发送一个命令。突发长度为 8 时，控制器每隔一个控制器时钟（慢时钟）周期发送一个命令。

端口信号说明

address[23:0]	输入，地址信号
bank_address[1:0]	输入，bank 地址
config_register1[14:0]	输入，模式寄存器配置数据 1
config_register2[12:0]	输入，模式寄存器配置数据 2
command_register[3:0]	输入，命令，即 ddr2_cmd_parser 的 u_cmd 命令
refresh_req	输入，刷新请求信号，直接由上层应用模块下达
refreshed	输出，刷新应答信号，送给下达刷新请求的上层应用模块
ddr2_rasb_cntrl	输出，行选择
ddr2_casb_cntrl	输出，列选择
ddr2_ba_cntrl[1:0]	输出，bank 地址
ddr2_address_cntrl[13:0]	输出，地址
ddr2_cke_cntrl	输出，时钟使能
ddr2_csb_cntrl	输出，片选
ddr2_ODT_cntrl	输出，ODT 使能
dqs_enable	输出，dqs 使能，送往 iobs 模块，配合产生 dqs 信号
init	输出，初始化命令，送往 ddr2_cmd_parser 模块

该模块对送过来的命令信号 command_register 进行分析，并相应产生下列一个命令周期的命令信号：

PRECHARGE_CMD	预充电
power_up_start	开始上电
initialize_memory	初始化
refresh	刷新
write_cmd_in	写

ld_mode	配置模式寄存器
read_cmd	读命令
active_cmd	激活命令
nop_cmd	空操作
ODT_en	ODT 使能
ODT_disable	ODT 禁止

将 config_register1 和 config_register2 则解释成具体的模式寄存器配置内容

Burst_length = config_register1[2:0]

Cas_latency = config_register1[6:4]

WR = config_register1[13:11] + 3’d2

EMR_OCD_not_support ={config_reg2[12:7] ， config_reg2[3] ， config_reg2[6:4] ，config_reg2[2:0]};

EMR_OCD_default = {config_reg2[12:10]， 3'h7，config_reg2[3]， config_reg2[6:4]，config_reg2[2:0]}

Command_register 解析后的信号及 refresh_req 信号等则产生：

start_power_up ， start_odt_turn_off ， start_odt_turn_on ， start_read_wait ，start_burst_read

，start_burst_write ， start_write_wait，start_first_write ， start_precharge_after_write ， start_precharge_after_write2， start_active ， start_refresh ，start_load_mode_reg ， start_precharge 信号

这些信号用于相应的状态机中，产生如下状态信号:

state_power_up ， state_odt_turn_off ， state_odt_turn_on ， state_read_wait ，state_burst_read

，state_burst_write ， state_write_wait，state_first_write ， state_precharge_after_write ， state_precharge_after_write2， state_active ， state_refresh ，state_load_mode_reg ， state_precharge

而这些状态信号则产生本模块的系列输出信号。具体的控制器状态图如下所示：

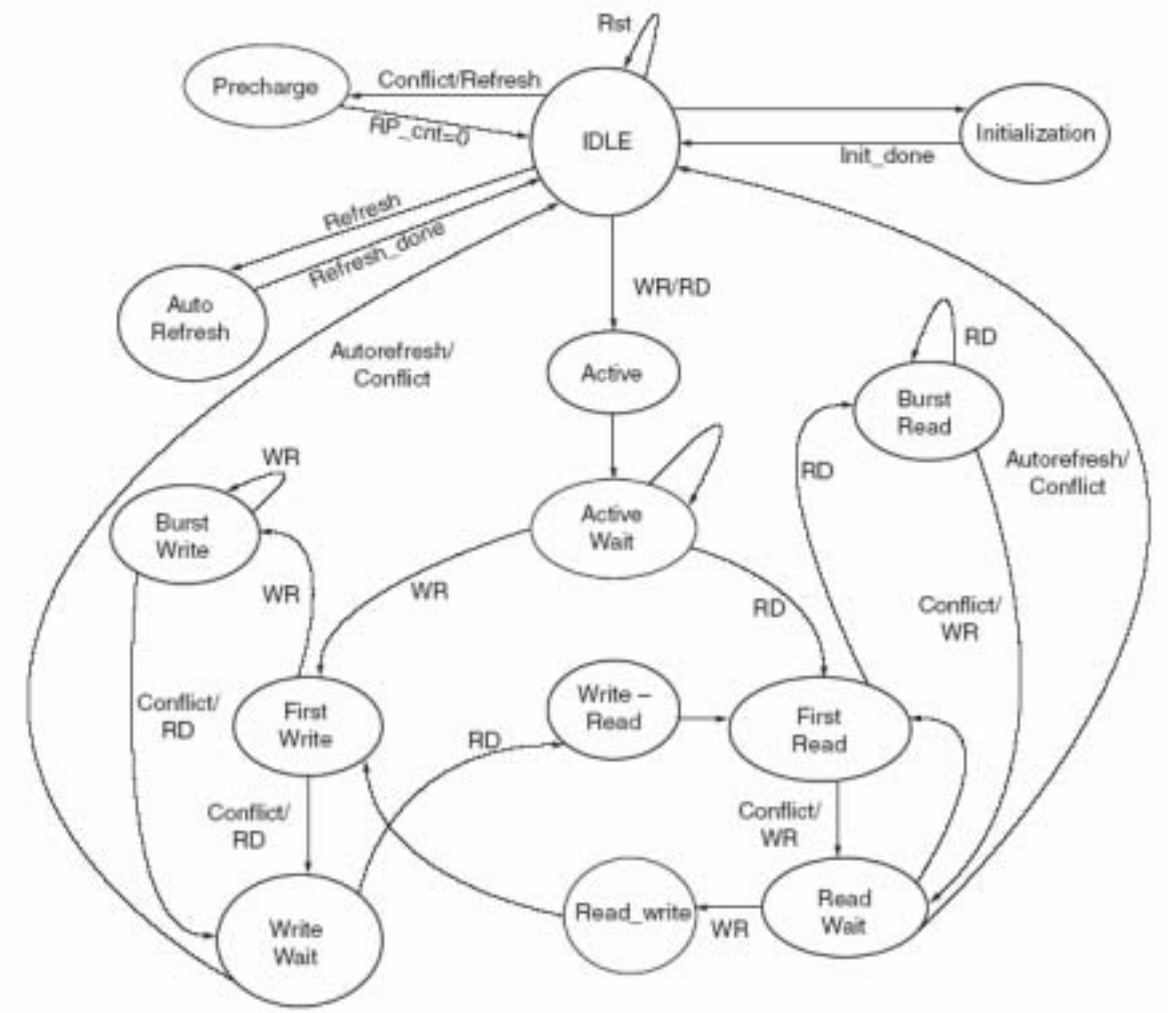


图 23 控制器状态机

Fig23 Controller State Machine

控制器开始设置在空闲(idle)状态，从 idle 状态只能跳入 Initialization 状态，在下达初始化命令的时候跳转，否则将一直处于 idle 状态，初始化完成后，内存模组便处于工作状态，需要定期进入 Auto Refresh 状态进行刷新。当需要写数据的时候，根据接收到的写入命令，首先进入激活状态(Active)，同时完成行的选择，随后进入写状态，并给出列地址，完成写入工作[20]。在进行读数据的时候，根据读命令同样进入激活状态，并给出行地址，随后进入读状态并给出列地址，然后内存模组将读出的数据放到数据线上。读和写都可以突发，第一次写进入 First Write 状态，然后由 Write Wait 状态跳转到 Burst Write 状态进行突发写，写结束后再由 Write Wait 转到 Idle 状态，第一次读则进入 First Read 状态，然后由 Read Wait 状态跳转到 Burst Read 状态，然后进入 Burst Read，读结束后再由 Read Wait 状态转到 Idle 状态。

3.3.3 数据路径(data_path.v)

数据构造模块完成 bank address，address，data 等的构造，以配合 parser 的命令一起送给 ddr2 模块处理。dm 信号没有，这里将数据掩码(dm)一律固定为 0，即 ddr2_top 发出的数据所有位都是有效的。dqs 信号是 ddr2 模块自己生成的，而不是这里给出的。

在实现上 nop 命令是个特殊的命令，由于每个命令都是一个时钟周期完成，然后跳转到 rdy_state 状态，而在 controller.v 中相应的状态是跳转到 idle 状态，而这个状态发出的命令就是 nop 命令[21]。因此，对 nop 命令 nop(task_delay)的处理，等效于等待 task_delay 个时钟周期后再发布后续的命令。

由于需要尽量利用原有设计，目前为止 ddr2 还没有做大的修改。需要改动的接口部分及说明如下 user_input_data 写数据，现在由 data_construct.v 提供 user_input_address 读写地址，现在由 data_construct.v 提供 user_bank_address bank 地址，现在由 dta_contruct.v 提供。burst_done，ddr2 的输入信号，这里的 burst 和 datasheet 里面的不一样，可能不对，不用，接低电平。user_output_data，ddr2 的输出信号，读数据。user_ODT_ack，ddr2 的输出信号，具体如何用待后面研究。代码内部的修改就是 controller.v 模块，因为送达的 u_cmd 信号和以前不一样了。

3.3.4 基础(infrastructure.v)

本模块的作用为产生整个内存控制器设计所需的 4 个相同频率不同相位的时钟，通过 dcm(数字时钟管理器)产生。频率都为 200Mhz。(实际调试运行在 126Mhz)

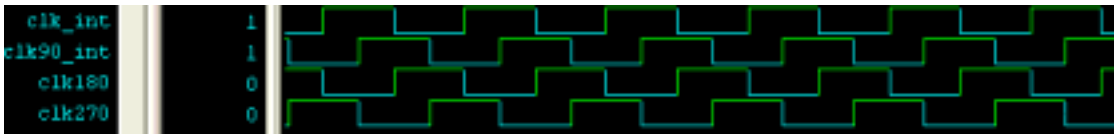


图 24 基础方针波形

Fig24 Infrastructure Waveform

3.3.5 输入输出端口模块(iobs.v)

这个模块的作用是将输入输出信号用 xilinx 可编程逻辑器件(spartan3)的 IO block 中的原语结构来进行处理，以便于时序能够很好的满足要求。这些输出信号主要有 3 类，分别为控制信号，数据信号和时钟信号。对应的则有 3 个子模块。

- 控制信号输入输出端口模块(controller_iobs.v)
- 数据通道输入输出端口模块(data_path_iobs.v)
- 基础信号输入输出端口模块(infrastructure_iobs.v)

▼控制信号输入输出端口模块(controller_iobs.v)

输入信号有

ddr2_rasb_cntrl

ddr2_casb_cntrl

ddr2_web_cntrl

ddr2_csb_cntrl

ddr2_address_cntrl

ddr2_ddr2_ba_cntrl

ddr2_cke_cntrl

ddr2_ODT0_cntrl

将这些信号经过 OBUF 原语结构进行处理，从而得到如下输出信号

ddr2_rasb

ddr2_casb

ddr2_web

ddr2_csb

ddr2_address

ddr2_ba

ddr2_cke

ddr2_ODT0

▼数据通道输入输出端口模块(data_path_iobs.v)

本模块对输出到 ddr2_dimm 的信号进行处理，相应的输入信号为 write_data_falling[127:0] 和 write_data_rising[127:0]， ddr2_dqs_en，mask_falling[7:0]和 mask_rising[7:0]，分别用于产生 dq，dqs 和 dm 输出信号。这里以产生 dq 信号为例，对相关设计作以说明。输入的数据信号经过称为 FDDRRSE 的原语结构，得到 ddr_dq_q 信号。

```
FDDRRSE DDR_OUT
(.Q  (ddr_dq_q);
 .C0 (clk270);
 .C1 (clk90);
```

```
.D0 (write_data_rising);
.D1 (write_data_falling);
.R  (GND);
.S  (GND));
```

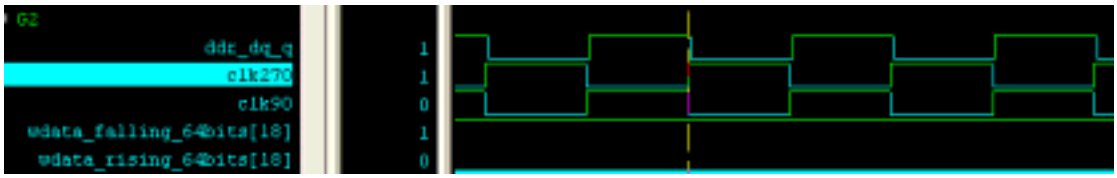


图 25 数据通道方针波形

Fig25 Data Path Waveform

由于 ddr2_dimm 数据信号为双向信号，因此需要知道何时为 ddr2_controller 控制，这个控制信号就是一个名为 write_en_val 的信号，高则表示当前由 ddr2_controller 控制该信号线输出。经过一个 FDCE 结构得到本模块所需要的控制信号 ddr_en。

```
assign enable_b = ~write_en_val ;
FDCE DQ_T(
   .D     (enable_b) ;
   .CLR   (RESET) ;
   .C     (clk270);
   .Q     (ddr_en);
   .CE    (CLOCK_EN)) ;
```

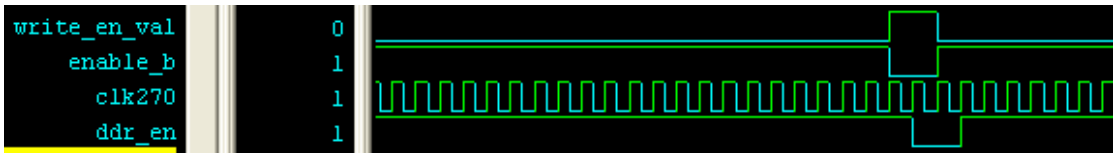


图 26 DDR_EN 信号生成仿真图

Fig26 DDR_EN Generation Waveform

最后，ddr2_dq_q 经过 OBUFT 结构产生 ddr_dq_inout 信号，这个信号就是将要输出到 ddr2_dimm 接口上去的信号。

```
OBUFT DQ_OBUFT(
      .I  (ddr_dq_q);
      .T  (ddr_en) ;
      .O  (ddr_dq_inout)) ;
```

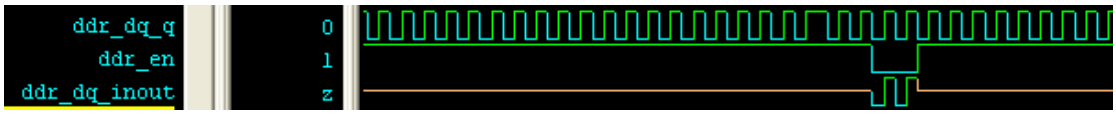


图 27 DDR_DQ_INOUT 信号生成仿真图

Fig27 DDR_DQ_INOUT Generation Waveform

对从 ddr2_dimm 过来的输入数据信号进行处理。dq 及 dqs 信号为双向信号，当由 ddr2_dimm 输入过来，则要对其进行接收，然后才可处理。在本模块中使用 IBUF 结构进行接收。

```
IBUF   DQ_IBUF(
    .I  (ddr_dq_inout);
    .O  (read_data_in));
```

图 28 输入数据接收仿真图

Fig28 Input Data Received Waveform

▼基础信号输入输出端口模块(infrastructure_iobs.v)

本模块用于产生 ddr2_clk0，ddr2_clk0b 信号给 ddr2_dimm。由设计内部 clk0 和 clk180 时钟信号，通过 FDDRRSE 结构产生 ddr2_clk0b 和 ddr2_clk0b_q 信号，然后将这两个信号送给 OBUF 结构，最终产生所需要的输出时钟。

```
FDDRRSE DDRCLK0_INST   (
    .Q   (ddr2_clk0_q);
    .C0  (clk0);
    .C1  (clk180);
    .CE  (vcc) ;
    .D0  (vcc) ;
    .D1  (gnd) ;
    .R   (gnd) ;
    .S   (gnd));
 OBUF  r1 (.I(ddr2_clk0_q);
    .0(ddr2_clk0));
```

4　仿真及实际运行调试

▼仿真及实际运行结果与波形

通过 Modsim 软件和 Debussy 软件进行仿真和察看波形，初始化仿真波形如下图所示：

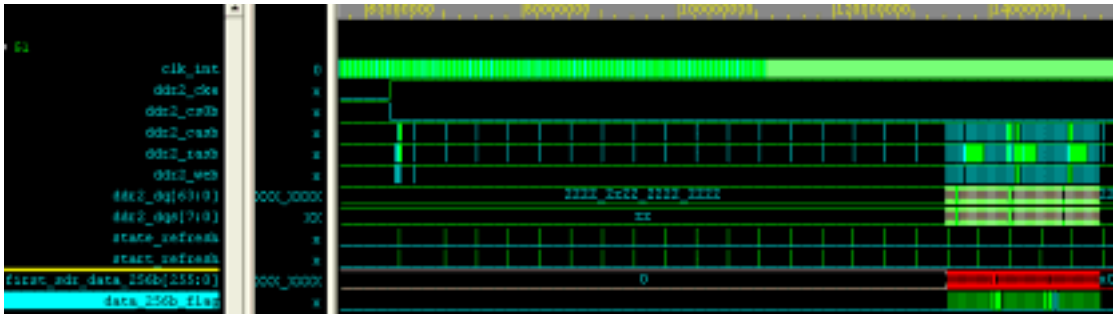


图 29 初始化波形图

Fig29　Initialize Waveform

注：仿真波形，之所以有红线，就是做板子时候将有些 dq 信号和 dqs 信号放到了差分对管脚，导致不能用，从而在代码中将相应的 dqs 禁用，导致数据不能正确接收导致。

局部波形如下图所示：

图 30 初始化波形图-局部

Fig30　Initialize Waveform Details

写数据方针结果如下图所示：

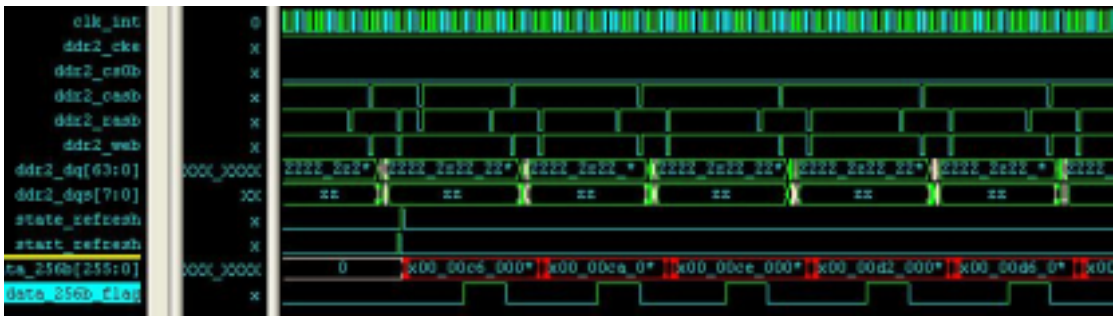


图 31 写数据波形图

Fig31 Write Data Waveform

下图是实际板级调试，chipscope 抓取到的信号。Flagb_n，fifo_data[15:0]和 slwr_n 为 usb 的接口部分信号 Data_256b_flag 则为从 ddr2 内存条读到的数据的标志信号，每个标志对应 256bits 的数据。Usb_dump_trigger_reg 则为从 ddr2 内存条读取数据的触发信号。

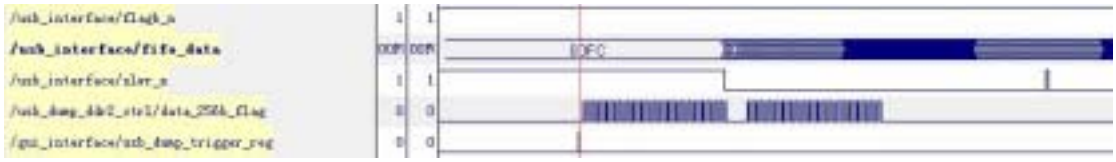


图 32 板极调试波形图 1

Fig32 On-board Emulation Waveform1

下图给出了一些内部的信号，其中的 state_refresh 则内存控制器内部的表示对内存进行刷新的状态信号。

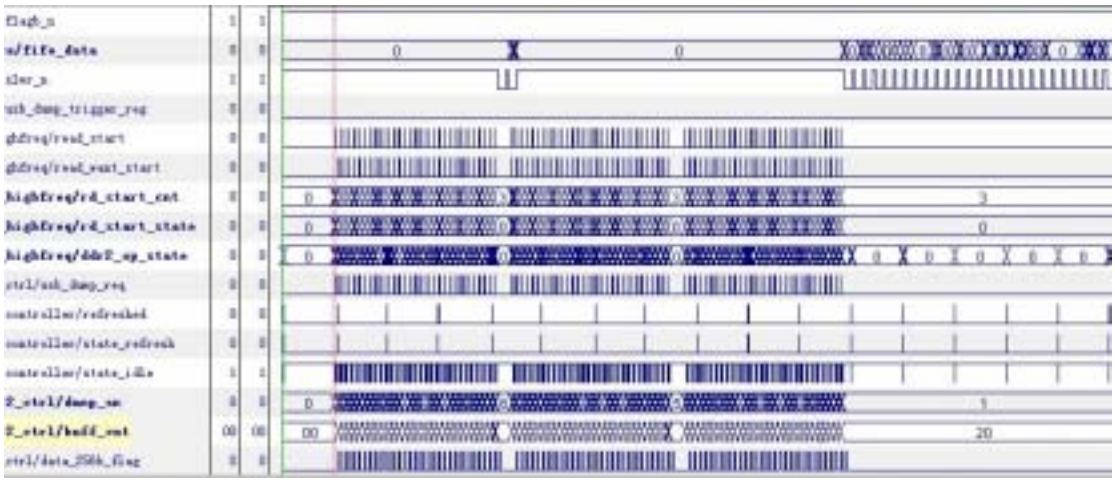


图 33 板极调试波形图 2

Fig33 On-board Emulation Waveform2

▼通过工具约束提高编译性能

由于 xilinx 的 spartan3 芯片性能所限，本设计最大的难点是提高工作频率，所以约束主要围绕提高编译时钟频率，约束在 ucf 文件中添加，也可通过菜单操作完成，ucf 中则会自动增加相应的内容。主要如下：

1）工作时钟频率的约束

设计工作的主时钟为 26Mhz 的 clk_26m_pin，DDR2 的工作时钟则为 200Mhz，降频工作在 126Mhz，由片外通过管脚输入 clk_4n_pin，并通过锁相环提供 DDRII 控制器所需要的 4 路时钟 clk0,clk90,clk180,clk270。因此对 clk26m_pin 和 clk_4n_pin 要进行时钟周期约束，如下：

```
NET "clk_26m_in" TNM_NET = "clk_26m_pin";
TIMESPEC "TS_clk_26m_pin" = PERIOD "clk_26m_pin" 38 ns HIGH 50 %;
NET "clk_pin_4n" TNM_NET = "clk_4n_pin";
TIMESPEC "TS_clk_4n_pin" = PERIOD "clk_4n_pin" 7.7 ns HIGH 50 %;
```

2）无关路径 set false parth

这一点也很重要，使得工具不对无关路径进行关注，从而提高编译的速度，同时也有利于需要关注的路径性能能够更好的提高。

时钟与复位信号之间，上述各时钟之间都是不相关的，因此要加上约束。

```
PIN "ddr2_top/ddr2/infrastructure/clk_dcm/DCM_INST.CLK0" TNM = "clk0";
PIN "ddr2_top/ddr2/infrastructure/clk_dcm/DCM_INST.CLK90" TNM = "clk90";
PIN "ddr2_top/ddr2/infrastructure/clk_dcm/DCM_INST.CLK180" TNM = "clk180";
PIN "ddr2_top/ddr2/infrastructure/clk_dcm/DCM_INST.CLK270" TNM = "clk270";
NET "sys_rst" TNM_NET = "reset";
NET "ddr2_top/ddr2/controller/rst0_r" TNM_NET = "reset";
NET "ddr2_top/ddr2/controller/rst180_r" TNM_NET = "reset";
NET "ddr2_top/ddr2/controller/rst180_r_inv" TNM_NET = "reset";
NET "ddr2_top/ddr2/controller/rst_dqs_div_r" TNM_NET = "reset";
NET "ddr2_top/ddr2/infrastructure/sys_rst180_1" TNM_NET = "reset";
NET "ddr2_top/ddr2/infrastructure/sys_rst180_o" TNM_NET = "reset";
NET "ddr2_top/ddr2/infrastructure/sys_rst270_1" TNM_NET = "reset";
NET "ddr2_top/ddr2/infrastructure/sys_rst270_o" TNM_NET = "reset";
NET "ddr2_top/ddr2/infrastructure/sys_rst90_1" TNM_NET = "reset";
NET "ddr2_top/ddr2/infrastructure/sys_rst90_o" TNM_NET = "reset";
```

NET "ddr2_top/ddr2/infrastructure/sys_rst_1" TNM_NET = "reset";

NET "ddr2_top/ddr2/infrastructure/sys_rst_o" TNM_NET = "reset";

TIMESPEC "TS_4" = FROM " clk_26m_pin " TO " clk_4n_pin " TIG;

TIMESPEC "TS_2" = FROM "clk0" TO "clk90" "TS_clk_26m_pin" * 1;

TIMESPEC "TS_4" = FROM "clk0" TO "clk180" TIG;

TIMESPEC "TS_5" = FROM "clk0" TO "clk270" TIG;

TIMESPEC "TS_6" = FROM "clk90" TO "clk180" TIG;

TIMESPEC "TS_7" = FROM "clk90" TO "clk270" TIG;

TIMESPEC "TS_8" = FROM "clk180" TO "clk270" TIG;

TIMESPEC "TS_9" = FROM "reset" TO "FFS" TIG;

3）在ISE的综合(synthesis)属性中，Optimization Goal选择”speed”,Optimization Effort中选择”High”。

对设计实现(Implement Design)，在映射属性(Map Properties 中，Map Effort Level 选 High，Map Effort Level 选择 Continue on Impossible，Optimization Strategy(Cover Mode)选择 Speed；在布局布线属性(Place&Route Timing Report Properties)中,Place And Route Mode 选择 Multi Pass Place and Route，Place&Route Effort Lever(Overall)选择 High，Place Effort Lever(Overrides Overall Level)选择 High，Router Effort Level(Overrides Overall Level)选择 High，Extra Effort(Highest PAR level only)选择 High。

时钟约束编译结果如下：

Constraint	Requested	Actual	Logic Levels	Absolute Slack	Number of errors
* TS_ddr2_top_ddr2_infrastructure_clk_dcm_CLK0_BUF = PERIOD TIMEGRP "ddr2_top_ddr2_infrastructure_clk_dcm_CLK0_BUF" TS_clk_4n_pin HIGH 50%	7.700ns	7.468ns	2	0.232ns	0
TS_ddr2_top_ddr2_infrastructure_clk_dcm_CLK90_BUF = PERIOD TIMEGRP "ddr2_top_ddr2_infrastructure_clk_dcm_CLK90_BUF" TS_clk_4n_pin PHASE 1.925 ns HIGH 50%	7.700ns	7.668ns	0	0.032ns	0
TS_ddr2_top_ddr2_infrastructure_clk_dcm_CLK180_BUF = PERIOD TIMEGRP "ddr2_top_ddr2_infrastructure_clk_dcm_CLK180_BUF" TS_clk_4n_pin PHASE 3.85 ns HIGH 50%	7.700ns	6.780ns	0	0.920ns	0
TS_ddr2_top_ddr2_infrastructure_clk_dcm_CLK270_BUF = PERIOD TIMEGRP "ddr2_top_ddr2_infrastructure_clk_dcm_CLK270_BUF" TS_clk_4n_pin PHASE 5.775 ns HIGH 50%	7.700ns	6.304ns	0	1.396ns	0
TS_clk_26m_pin = PERIOD TIMEGRP "clk_26m_pin" 38 ns HIGH 50%	38.000ns	19.516ns	6	18.484ns	0
TS_2 = MAXDELAY FROM TIMEGRP "clk0" TO TIMEGRP "clk90" TS_clk_26m_pin	38.000ns	5.857ns	0	32.143ns	0

▼本设计所用 FPGA 为 Xilinx 的 spartan3 中一款，具体为 xc3s4000-4fg900。

综合编译之后的资源消耗情况如下图所示，其中 Flip Flop 用了 5392 个，Block RAM 用了 4 个，GCLK 用了 6 个，DCM 用了 1 个。

Logic Utilization	Used	Available	Utilization	Note(s)
Total Number Slice Registers	5,400	55,296	9%	
Number used as Flip Flops	5,392			
Number used as Latches	8			
Number of 4 input LUTs	3,119	55,296	5%	
Logic Distribution				
Number of occupied Slices	4,618	27,648	16%	
Number of Slices containing only related logic	4,618	4,618	100%	
Number of Slices containing unrelated logic	0	4,618	0%	
Total Number 4 input LUTs	3,372	55,296	6%	
Number used as logic	3,119			
Number used as a route-thru	146			
Number used for Dual Port RAMs	102			
Number used as Shift registers	5			
Number of bonded IOBs	132	633	20%	
IOB Flip Flops	76			
IOB Dual-Data Rate Flops	93			
Number of Block RAMs	4	96	4%	
Number of GCLKs	6	8	75%	
Number of DCMs	1	4	25%	
Total equivalent gate count for design	342,306			
Additional JTAG gate count for IOBs	6,336			

5　应用实例

通过 FPGA 实现的 DDR2 接口设计以其灵活性和较高性价比，可以在通信、编写电子设备等很多方面展开应用。本文仅以该接口在数据的实时抓取和显示应用为例做一简要介绍，该应用需要大数据量高带宽的数据采集和传输。

其系统连接图如下所示：

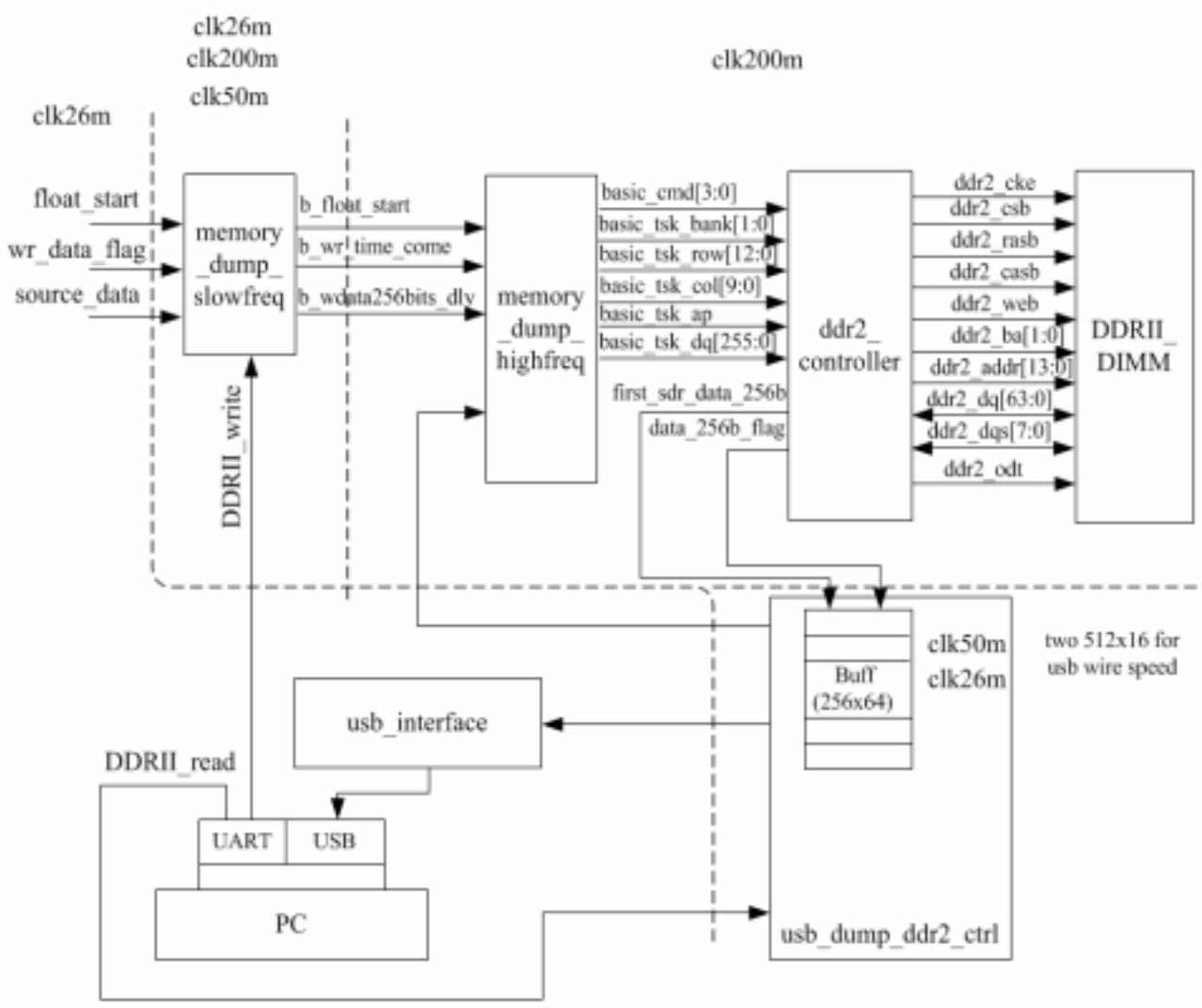


图 34 PC 通过 USB 对 DDR2 实现数据采集系统连接图

Fig34　PC Read/Write Data to DDR2 Through USB Diagram

在该系统中写数据时，只要触发了开始写信号，便从 0 地址开始，连续不断的写入指定的长度，然后停止。读数据则从 USB 口读取而不是 UART 口，也不需要通过总线的方式了。(当然，如果以后也用 USB 口来进行配置，则要考虑将 USB 口加到总线上去，但由于时间限制，目前 USB 只用作将 DDR2 中的数据读取出来送给 PC，因而做成点对点通信，且只有读取功能)。通过基于 FPGA 的 DDR2 接口设计，PC 可以实现快速的数据读取，从而大大提高效率。

在 FPGA 设计中，可以用内置逻辑分析仪进行内部信号的实时显示，其实现原理就是将需要显示的内部数据保存到 FPGA 的内部 RAM 中去，并通过 JTAG 口实时送达 pc，在显示器上显示出来。内置逻辑分析仪的最大缺点就是 RAM 容量小，导致抓取信号数量有限，存储深度有限。

本设计的作用就是利用片外 RAM 来存储数据，并通过 usb 口送达 pc 显示器显示，从而完全摆脱了 RAM 容量的限制。由于要做到实时显示，所以需要很高的带宽才能保证大量信号数据能够实时传递。所以采用 DDR2 内存条及 usb2.0。实现原理如图所示。uart 口为配置端口，可以将需要采取的操作通过该口写入设计内部相关的寄存器中，这些操作涉及的配置数据包括：写开始，写入何种数据，写入起始位置，写入长度；读开始，读取起始位置，读取长度。写命令由 uart 口下达，需要存储显示的信号为 source_data，开始存储的触发信号为 floag_start,经过跨时钟域处理后为 b_float_start 和 b_wdata256bits_dly，通过相应的命令形式将数据及要求的操作告知 ddr2_controller 模块，该模块于是完成对内存条的写动作。读命令也有 uart 口下达，命令送达 usb_dump_ddr2_ctrl 模块，该模块将命令送给 memory_dump_highfreq 模块，这个模块则通过控制 ddr2_controller 模块完成对内存条的读工作，读取的数据由 ddr2_controller 模块送达 usb_dump_ddr2_ctrl 模块内部的 buff 中去，并通过 usb_interface 模块，由 usb2.0 实时传输给 pc，从而完成了实时读取工作。

当然，最终数据能以逻辑分析仪的形式在 pc 的显示器上显示出来，还需要软件的支持。

6 全文总结

本论文对通过 FPGA 实现 DDR2 接口设计进行了模块化实现，并对每个模块进行了代码设计与仿真实现。由于非自由运行 DQS 和存储器提供的边沿对齐的读数据使得再 FPGA 中实现度数据采集接口十分困难，对于较高频率（大于 100MHz）的接口，必须使用读存储器 DQS 信号，以实现较高的余量。为了将 DQS 置于数据采集窗口的中心，必须将其延迟。被延迟的 DQS 利用本地时钟资源分布到 FPGA 中。随着频率的不断提高，如何将 DQS 信号准确地置于数据采集窗口的中心，我们将面临更大的挑战。

FPGA 设计人员在满足关键时序余量的同时力争实现更高性能，在这种情况下，存储器接口的设计是一个一向构成艰难而耗时的挑战[23]。FPGA 提供 I/O 模块和逻辑资源，从而使接口设计变得更简单、更可靠。通常而言，电子应用可分为两类：一类是低成本应用，降低器件成本为主要目的；另一类是高性能应用，首要目标是谋求高带宽。运行速率低于每引脚 400 Mb/s 的 DDR SDRAM 和低端 DDR2 SDRAM 已能满足大多数低成本系统存储器的带宽需求。本设计采用 Spartan3 FPGA，具有较高的性价比，因此这一接口设计具有较强的灵活性和应用性，同时也可以作为系统级设计的参考。

参考文献

[1] 赵雅兴，“ FPGA 原理及应用 ”，天津大学出版社，1999

[2] Micron Corporation，“ DDR2 Datasheet ”,2005

[3] Xilinx Corporation，“ Spartan-3 Complete Data Sheet ”,April 2006

[4] Xilinx Corporation，“ Spartan-3 Generation Configuration User Guide ”2006

[5] Xilinx Corporation，“ Spartan-3 Generation FPGA User Guide ”2006

[6] Howard W.Johnson，PHD.，Martin Graham，PHD.，“ HIGH-SPEED DIGITAL DESIGN---A Handbook of Black Magic ”，2001

[7] Clifford E.Cummings，“ Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs ”，2001

[8] J.BHASKER，“ Verilog HDL Syntesis---A Practical Primer ”，1998

[9] 边计年，薛宏熙，“ 用 VHDL 设计电子线路 ”，北京：清华大学出版社，2000

[10] Samir Palnitkar, “ Verilog HDL, A Guide to Digital Design and Synthesis ”,Sunsoft Press A Prentice Hall Title,1996

[11] Veendrick,Harry J.M., “ The Behavior of Flip-Flop Used as Synchronizers and Prediction of Their Failure Rate ”,IEEE Journal of Solid-State Circuits,Vol. SC-15,No.2.,April 1980

[12] E.A.Lee and D.G.Messerschmidt, “ Digital Communication ”.Norwell，1994

[13] F.Krummenacher and N.Joehl, “ A4-MHz CMOS continuous-time filter with on-chip automatic tuning ”,IEEE J.Solid-state Circuits,vol23,June 1988

[14] Y.-T.Wang and B.Razavi, “ An 8-bit 150-MHz CMOS A/D converter ”,IEEE J.Solid-State Circuits,vol.35,Mar 2000

[15]“ TMS626162,TMS626812 16Mb Synchronous DRAMs Technical Reference ”, TI Corporation,1996

[16]“ Shared Memory Interface With the TMS320C54x DSP ”，TI Corporation,1998

[17] 侯伯亨，顾新著，“ VHDL 硬件描述语言与数字逻辑电路设计 ” 西安电子科技大学出版社，2001

[18] Douglas L. Perry，“ 电子设计硬件描述语言 VHDL ”，北京：学苑出版社，1994

[19] J. Bhasker，“ Verilog HDL Synthesis-- A practical Primer ”， Star Galaxy Publishing, ISBN

[20] 潘松，黄继业. “ EDA 技术实用教程 ”.北京：科学出版社，2002

[21] 王金明，杨吉斌. “ 数字系统设计与 Verilog HDL ”.北京：电子工业出版社，2002

[22] 黄正瑾. “ 在系统编程技术及其应用 ”.南京：东南大学出版社，1997

[23] David Comer Donald Comer， “ 电子电路设计 ”， 中国计量出版社，2005

致谢

衷心感谢我的导师韩泽耀老师，尤其在论文准备期间，韩老师给予我悉心指导和帮助。韩老师以他渊博的知识、严谨的治学态度和忘我的工作精神给我树立了学习的榜样，将使我终身受益。

感谢我的企业导师高继业和倪安臣及张亿宾同学，在我需要帮助的关键时期，他们都给予我无私的帮助和指导，使得我能顺利完成论文，在此表示由衷的敬意。同时还要感谢我家人在我就读研究期间给予我的支持和理解！

攻读学位期间发表的学术论文

1. 陈良明、韩泽耀，“ 浅析 OFDM-第四代移动通信的主流技术 ”，《 计算机技术与发展》，已录用，将于 2008 年第 4/5 期发表。