分类号_____ 密　级 公 开

UDC _____ 学校代码 10497

武汉理工大学

学　位　论　文

题　目 基于 FPGA 的 SD 卡控制器的设计与实现

英　文 The Design and Implementation of SD
题　目 Card Controller Based on FPGA

研究生姓名 谷　洵

指导教师 姓名 刘 岚 职称 教 授 学位 硕 士
单位名称 武汉理工大学信息工程学院 邮编 430070

申请学位级别 硕 士 学科专业名称 电路与系统

论文提交日期 2010 年 10 月 论文答辩日期 2010 年 11 月

学位授予单位 武汉理工大学 学位授予日期 _____

答辩委员会主席 刘 泉 评阅人 刘 泉
陈 适

2010 年 10 月

独 创 性 声 明

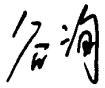
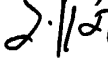
本人声明，所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得武汉理工大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名： 日 期：2010.12.2

学位论文使用授权书

本人完全了解武汉理工大学有关保留、使用学位论文的规定，即：学校有权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。本人授权武汉理工大学可以将本学位论文的全部内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存或汇编本学位论文。同时授权经武汉理工大学认可的国家有关机构或论文数据库使用或收录本学位论文，并向社会公众提供信息服务。

（保密的论文在解密后应遵守此规定）

研究生（签名）：  导师（签名）：  日期 2010.12.2

摘 要

随着存储行业的快速普及和发展，SD 卡的运用日益广泛。在畅销的 U 盘控制芯片之后，SD 存储卡控制芯片领域在市场中占更大份额。与传统的 U 盘读卡器相比，本文设计实现的 SD 卡控制器具备了传输速度更快，运用范围更广的优势。该 SD 卡控制器以 SD3.0 新标准为指导，除支持普通 SD 卡外，特别支持高速 SDXC 卡，使得 SD 卡控制器可以运用在高速领域。同时，本控制器经后端 IC 部门研发可以整合成 SD 卡控制芯片，运用于笔记本电脑中。

SD3.0 标准与原来相比，增加了 UHS-I 模式和 DMA 模式，可以支持 SDXC 卡。通过设计使得 SD 卡在工作频率及性能上有较大提高。本文在介绍了国内外研究情况和 SD 卡规范后，从 SD 卡控制器总体结构入手，简要介绍了 SD3.0 新标准下各模块功能。根据改进内容，主要设计了时钟模块，Tuning 模块和 DMA 模块三个模块，并作了功能仿真、逻辑综合，同时对整体进行 FPGA 验证。

在时钟模块中主要利用 FPGA 特有资源 DCM，可动态提供 25MHz~160MHz 不同的工作频率。利用其优势，不仅为 SD 卡控制器提供了稳定的时钟同时还提供了灵活的频率配置。为了使 SD 卡工作在高频时能满足时序要求和正确采样到数据，通过在 Tuning 模块提出最佳采样时钟选择算法，合理设计状态机，对时钟相位进行调整，得到 SD 卡工作在高频时的最佳采样时钟。该模块通过功能仿真及 FPGA 逻辑综合，达到设计要求，也保证了数据传输的可靠性。DMA 模块打破了 SD 卡与 PC 机进行通信的传统传送方式，不通过 CPU 的处理，直接在外设及内存中进行通信。在逻辑设计中，重点设计双口 RAM 及 RAM 控制部分，把读写过程细化，完成 SD 卡在 DMA 模式下的传输。最后对整体 SD 卡控制器进行 FPGA 验证。先从电源、配置、下载的角度介绍了开发板的设计，然后根据验证场景对其进行验证，特别对 SDXC 卡在高速模式下进行了读写速率及工作频率的测试，也与普通 SD 卡和 U 盘读卡器进行了性能对比，得出验证结果。

通过几个月的研究设计，最终设计实现了一种实用广泛，在高频下工作稳定的 SD 卡控制器。本控制器特别实现了 SDXC 卡在高频时的工作情况。将原来普通 SD 卡工作频率为 25MHz 或者 50MHz，读写速度不超过 25MB/S，15MB/S，提高到工作频率为 160MHz，读写速度约为 60MB/S，35MB/S。

关键词：FPGA，SD 卡，时钟，采样，传输

Abstract

With the rapid spread and development in storage industry, SD Card is widely applied in many fields. After the best-selling controller chips of U-disk, the controller chips of SD memory cards account for greater proportion in the market. Compared with the traditional U-disk reader, the controller of SD card has many advantages, such as the faster transfer speed, the wider range of application. The controller of SD card in this paper based on new criteria of SD3.0.In addition to supporting the universal SD card in the market, it also supports high-speed SDXC card. So the SD card controller can be applied in the high-speed area. The controller in this paper can be integrated into the chip of SD controller by the back-end of IC sector, and then be used in the computers of notebook.

Compared with the original standard, in SD3.0 standard, UHS-I mode and DMA mode are increased. By improvement, it supports high-speed SD cards ---- SDXC card. Therefore, operating frequency and performance are improved greatly. Domestic and foreign research and the SD criteria are firstly described in this paper, and then from the point of overall structure about SD controller, function of all modules based on new standards SD3.0 is introduced briefly. According to the contents of new improvement, the clock module, Tuning module and DMA module are mainly designed, which are simulated, synthetically and verified through FPGA.

In clock module, the main advantage is the specific resources of FPGA----DCM. It can dynamically supply 25MHz~160MHz frequency for the SD card, because make full use of its advantages, it not only supplies a stable clock, but also can configure the different ranges of frequency. In order to meet timing requirements and to sample the data properly, by presenting the algorithm of best sampling clock selection, designing the state machine rationally, adjusting the phase of clock, the best sampling clock is selected in Tuning module. In this module, it meets the design requirements by adoption of FPGA logic synthesis and simulation. Meanwhile, ensure the reliability of data transmission. DMA module makes a breakthrough in area of traditional transmission modes between SD card and PC, which does not through the processing of CPU, communicating between the peripheral and memory directly. In

this module, it focuses on design of dual-port RAM and the control of RAM, making a refinement on the process of reading and writing, and then completing the transmission of SD card in DMA mode finally. At last, the overall controller of SD card passes verification of FPGA. After the design of the development board from the power supply, configuration and download is introduced, validation can be finished by the scene. This paper focuses on testing the speed of read and write, particularly about the frequency and performance in high-speed mode for SDXC cards, meanwhile, makes compare with the U-disk reader and universal SD card on rate. At last, the results are obtained.

Through months of research and design, finally a wide use of controller which works in the high frequency stability has been designed. This controller implements work of SDXC card. Original controller only works at the frequency of 25MHz or 50MHz,the speed of read and write is only 25MB/s, 15MB/s, but now, the frequency of SD card is raised at 160MHz,and the speed of read and write is about 60MB/s,35 MB/s.

Keyword: FPGA, SD card, clock, sample, transmission

目 录

第1章 绪论

1.1 论文研究背景

SD卡（Secure Digital Memory Card）中文翻译为安全数码卡，是一种基于半导体快闪记忆器的新一代，一种使用广泛的移动存储介质。其主要用于数码相机、手持式PC、游戏机、打印机、摄像机，以及其它消费型电子产品。在所有存储卡中占有率最高，而且也是增长最快的一种卡片[1]。

由于目前便携式多媒体电子产品朝小型化、低功耗和高配置发展已是必然趋势，随着无晶圆(Fables)设计公司的兴起和系统厂家对客制化产品要求的提高，市场将更倾向于使用芯片来控制各种接口设备（例如SD卡，MMC卡等）[2,3]。

1.2 论文研究的目的和意义

随着便携式多媒体电子产品的普及和发展，随着人们对便携式产品应用要求的提高，例如对音频，视频内容的海量存储，使得生产厂家所设计出的电子产品必须具有丰富的接口以连接各种移动、便携的存储设备[4]。特别是各类笔记本中的扩展接口不断发展，使得笔记本功能更强大。与传统台式机相比，笔记本除了有传统的USB接口外，还多了不少存储卡的接口，如SD，MS，MMC卡等。为了控制这些卡与主机通信，就少不了各类IO控制芯片。当前笔记本电脑主板中主要使用PCI Express总线，它是新一代的总线接口，比起PCI以及更早期的计算机总线的共享并行架构，每个设备都有自己的专用连接，不需要向总线请求带宽，而且可以提高数据传输率，达到PCI所不能提供的高带宽[5,6]。本文主要研究基于PCI Express总线SD卡的控制，设计了一种支持不同容量，不同频率且读写性能较好的SD卡控制器，经过后期IC部门的研发，可以整合成芯片，用于笔记本电脑中。

随着芯片性能越来越高，规模越来越大，设计复杂度迅速增加；同时，市场对产品设计周期的要求越来越高，因此造成了设计复杂度和设计产能之间的巨大鸿沟，由此导致开发周期越来越长，设计质量利用率越来越难于控制，芯片设计成本越来越趋于昂贵[7-9]。所以，本文基于FPGA进行设计，其意义在于：

FPGA 有着灵活性强，开发周期短，并行处理速度快，可靠性高等优点。除此以外，因为 FPGA 是可编程逻辑器件，用户可对 FPGA 进行重新配置，以实现用户的逻辑。它还具有静态可重复编程和动态系统重构的特性，使得硬件的功能可以像软件一样通过编程来修改，成为专用集成电路（ASIC）领域中的一种半定制电路，FPGA 既克服了定制电路的不足，又解决了原有可编程器件门电路数有限的缺点，可以毫不夸张的讲，FPGA 能实现任何数字器件的功能，上至高性能 CPU,下至简单的 74 电路[10]。

目前一些 SD 卡控制器，例如各类 USB 接口的读卡器，对于 SDSC（SD Standard capacity）1GB,2GB 卡的读写速度一般为 10MB/S,7MB/S；SDHC（SD High capacity）4GB 以上的卡读写速度一般为 20MB/S,15MB/S,并且不支持 DMA 传输，不支持多时钟频率，不支持 SD3.0 中关于 UHI-S 的几种模式。而且因为 USB 接口的传输速度本身受到很大限制，导致 SD 卡的读写速度不可能提高很多。另外，有些设计是基于单片机实现，用单片机实现的弊端：1.因为其本身提供的时钟频率不够，不可能设计高频率的控制器。2.SD 对外实现 2 种模式的访问，一种为 SPI 模式，一种为 SD 模式[11]。单片机采用的 SPI 模式显然没有 4 线并行传输的 SD 模式快，所以在速度上受到了限制。

所以本文设计一种可在 FPGA 上实现的 SD 卡控制器，可以方便移植代码，生成逻辑网表，供后端继续研发使用。同时利用 FPGA 进行验证也节省了开发周期，提高了设计的可靠性。

1.3 国内外研究情况

前几年，尽管中国已成为全球移动存储制造业重要基地，但国内厂商主要依赖海外公司提供的主控芯片和芯片设计方案，进行组装式加工和生产，手中并没真正掌握核心技术；这几年，随着芯邦自主研发的 U 盘和 SD 卡控制芯片的成功推出，海外厂商对移动存储芯片的垄断已被打破[12]。

一批先发企业抓住移动存储行业快速普及的商机，近年来成功实现了跨越式发展，其中引人注目的包括：拥有全球闪存技术基础专利的朗科，目前已迅速成长为国内外最具竞争力的 U 盘龙头厂商；以芯邦微电子为代表的深圳芯片设计公司，也通过自主研发，成功进军移动存储控制芯片这一更为庞大的“上游”市场，成为引领移动存储行业的“黑马”，芯邦微电子总裁张华龙透露，在掌握了 U 盘控制芯片核心专利技术基础上，芯邦已进入了具有更大市场的 SD / MMC

存储卡控制芯片领域，并在成功研发、推出了国内第一款自主知识产权的 SD / MMC 存储卡[13]。

国外，SD 卡 3 大巨头新帝（SanDisk）、松下（Panasonic）、东芝（Toshiba）是目前主流 SD 卡规格创始者，过去都坚持采取自制 NAND Flash 控制芯片，但近年来随着快闪记忆卡产品趋于成熟，台、美系模块厂大举入侵，成本压力排山倒海而来，使得 3 大巨头采购策略开始松动[14]。继东芝将 NAND Flash 控制芯片采购权释放给群联，Panasonic 亦首度采用台厂 SD 卡控制芯片，订单花落擎泰，目前 SD 卡 3 大巨头仅剩下新帝仍坚守采用自制控制芯片策略。

内存业者指出，过去在 SD 卡领域由这 3 家业者掌控市场和规格，每次推出新的技术规格，都是等到自己的控制芯片技术成熟后，才把规格释出给其它业者，导致其它记忆卡厂和控制芯片业者要推出同等级产品，总是要晚上 3~6 个月，但这种情况随着台、美系内存模块厂大举入侵记忆卡市场后已逐渐改善，加上台厂技术支持能力越来越强，促使东芝、Panasonic 纷弃守 SD 卡控制芯片[15]。

凹凸电子在笔记本电脑 IO 控制芯片领域也占有相当的份额，在 Dell、东芝、富士通等中、高端品牌电脑中相当容易找到来自 O2Micro 的读卡器、Pcmcia 控制器、1394 控制器等芯片的身影。特别地，凹凸电子研发了 SD3.0 新标准下有关 SDXC 卡的控制，使得 SD 卡的读写速率得到提升，同时也支持多种 SD 卡。

1.4 论文结构

本论文主要根据 SD3.0 新标准，基于 FPGA 对 SD 卡控制器进行了设计和实现。论文正文分为六个章节，每章内容安排如下：

第 1 章：绪论。主要介绍了 SD 卡的含义，论文研究背景，国内外发展情况，研究的目的及意义。

第 2 章：SD 存储卡的相关规范。主要介绍 SD3.0 新标准下 SD 存储卡物理层规范及 SD 控制器规范，然后说明了本控制器的规格。

第 3 章：SD 卡控制器结构。首先介绍了 SD 卡控制器总体结构，然后重点介绍 SD 控制器中 I/O 接口模块、数据命令控制模块、时钟模块、Buffer 控制模块、Tuning 模块，DMA 接口模块的具体功能。

第 4 章：重点模块的设计。本章重点设计了上章节中的三个模块，分别为时钟模块，Tuning 模块和 Buffer 控制模块中的 DMA 模块。各模块从设计目的，设计原理，设计过程，设计结果详细阐述了 FPGA 的实现过程。

第 5 章：SD 控制器 FPGA 验证结果。首先介绍 FPGA 验证环境，从电源，配置以及下载三个方面阐述了开发板的设计理念。然后罗列出 FPGA 验证的几种场景。特别验证了 SDXC 卡（高速 SD 卡）的高速模式，最后与普通 USB 控制器的传输速度进行了比较。

第 6 章：总结与展望。总结论文内容、个人工作及未来需要继续开发实现的新内容。

第 2 章　SD 存储卡及控制器相关规范

SD 储存卡是由日本的松下、东芝和美国的 SanDisk 公司所组成的 SD 协会（SD Group）联合制定规范和开发的，1998 年首次发布 SD1.0。本文主要研究的基于 SD3.0 规范的控制器。该规范是 2009 年 4 月首次发布。本章主要介绍存储卡规范、控制器规范及所设计的 SD 卡控制器的规格。

2.1 SD 存储卡的介绍和规范

SD 卡的物理特点如管脚分配和数据传输协议和 MMC（Multi Media Card）兼容并略有增加，因此其大小和 MMC 卡差不多，仅比 MMC 卡厚了 0.7mm，以容纳更大的存储单元，重仅约 1.6 克，在外观上，SD 储存卡接口除了保留 MMC 卡 7 个管脚外，还在两边加了 2 个管脚作为数据线，并且带了物理写保护开关[16]。下图为 SD 卡内部结构图，如图 2-1 所示。

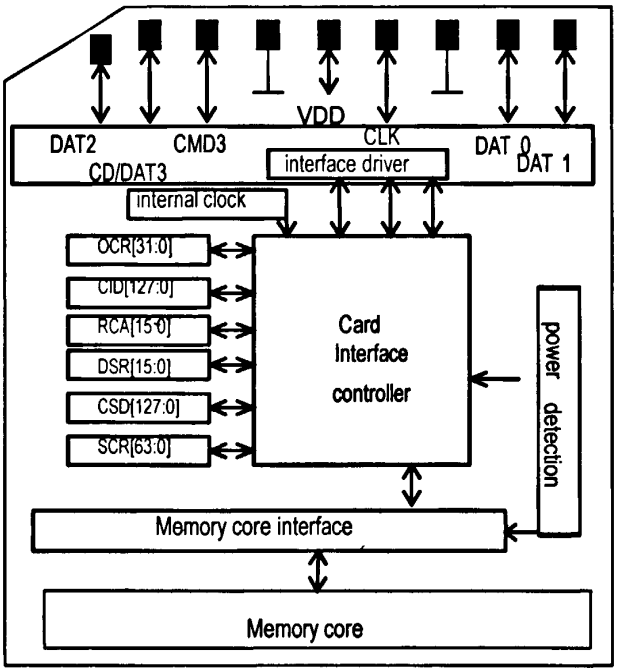


图 2-1 SD 卡内部结构图

其 9 个管脚中，CD/DAT3 为卡检测数据位，CLK 为时钟，CMD 为命令，

DAT0~DAT3 为 4 根数据双向输入线，VDD,VSS1,VSS2 为电源和地。

SD 储存卡内有 6 个字节长短不等的寄存器，其定义如下：

CID：128 比特，用于证明卡身份的识别号码。为强制型。

RCA：16 比特，相关卡地址：卡的本地系统地址，由卡和控制器在初始化时确定。强制型。

DSR：16 比特，驱动进阶寄存器：用于设置卡的输出驱动能力。可选型。

CSD：128 比特，卡特定数据：有关卡的操作条件的数据信息。强制型。

SCR：64 比特，SD 配置寄存器：有关 SD 卡的特定功能信息。强制型。

OCR：32 比特，操作条件寄存器：强制型。

OCR 操作条件寄存器存储着卡的 VDD 电压信息，此寄存器包含一些状态信息位。当卡上电完成后，卡的上电状态位被置起。另一个状态位表明卡的容量，当卡为高容量时，该位置 1；为标准容量时，为 0。当卡上电完成后，控制器读该状态位来识别卡的容量。特别地，针对双电压卡，当卡收到 CMD8 命令时，该寄存器比特 7 置 1，否则为 0。CID 寄存器包含厂商 ID 号，产品名称，产品版本，产品序列号等信息。CSD 寄存器定义了数据格式，错误正确类型，最大数据访问时间，卡命令级以及最大数据传输速率等。RCA 寄存器在卡识别阶段携带卡地址信息。DSR 寄存器用来改进总线速率或者扩大操作条件。SCR 寄存器提供卡的安全支持，数据总线宽度支持，SD3.0 版本支持等。

SD 卡有两种工作模式，分别为卡识别模式和数据传输模式。主机上电复位和卡复位后都将处于卡识别模式，它会在总线上等待卡，直到 SEND_RCA(CMD3) 命令到来。卡收到 SEND_RCA(CMD3)命令后进入数据传输模式，主机识别到卡后也进入此模式[17]。在卡识别模式时主机复位总线所有的卡，验证工作电压，询问卡的地址，这个模式下所有数据的传输都是只通过 CMD 线来完成。进入数据传输模式后，主机先通过发 SEND_CSD (CMD9)命令获取卡的 CSD 信息。SET_DSR (CMD4)用于设置卡的 DSR 寄存器，包括数据总线宽度，总线上卡的数目，总线频率，当设置成功后，卡的工作频率也随之改变。传输模式下所有的数据传输都是点对点的，并且所有有地址的命令都需要有响应。

2.2 SD 控制器的规范

SD3.0 控制器规范中主要对 SD 卡的时钟，电源，读写等进行控制，介绍标准寄存器的含义以及一些新增加的部分，这里主要对 SD3.0 新增部分进行介绍。

SD3.0 中提出了一个新的 UHS-I 模式，UHS-I 的全称即 Ultra High Speed Phase I[18]。UHS-I 操作的模式分为 6 种。

- ✧ DS：缺省时工作频率为 25MHz，电压为 3.3V
- ✧ HS: 高速模式，工作频率为 50MHz，电压为 3.3V
- ✧ SDR12:工作频率为 25MHz，电压为 1.8V
- ✧ SDR25:工作频率为 50MHz，电压为 1.8V
- ✧ SDR50:工作频率为 100MHz，电压为 1.8V
- ✧ SDR104:工作频率为 208MHz，电压为 1.8V
- ✧ DDR50:工作频率为 50MHz，电压为 1.8V

对于 DS 及 HS 两种模式在 SD2.0 中已经支持，SD3.0 中特别规定了另外 5 种新的模式，其中，SDR12 及 SDR25 因为工作频率未变，只是电压变化为 1.8V，所以将电压从 3.3V 转化为 1.8V 即可。后四种模式，不但频率变化，电压也要切换至 1.8V。其中 DDR50 因为是利用时钟双沿采集数据，在同样频率为 50MHz 的情况下比 SDR25 速率要快。特别地，因为 SDR50 和 SDR104 两种模式工作频率很高，对时序要求较高，所以采用 Tuning 进行时钟相位调整。采样点的选择成为重点难点，针对这种情况，后文也给出了解决方案并实现其功能。UHS-I 只支持 SD 卡的 4 比特模式。一旦进入到 UHS-I 模式，卡会自动转变为 4 比特模式。另外，满足 UHS-I 模式的卡也是 SD 卡中的一种，这种卡必须支持双电压，同时工作频率最高可达 208MHz，即 SDXC 卡。这种卡的读写性能及操作也是本文研究的重点。SD3.0 中因为增加了 DMA 传输模式，与以往没有 DMA 传输的情况相比也加大了传输速率。

2.3 SD 控制器的规格

本文设计的 SD 卡控制器具备如下功能：

- ✧ 支持 SD3.0 存储卡协议；
- ✧ 支持多时钟频率：25~160MHz；
- ✧ 支持 DMA 传输；
- ✧ 最大读写速率为：60MB/s,35MB/s；
- ✧ 存储卡写保护；
- ✧ 支持高频率数据传输；
- ✧ 支持 SDXC 卡（最新一种 SD 卡）。

第 3 章　SD 卡控制器结构

SD 卡控制器主要用来控制 SD 卡的读写功能，相当于一个读卡器。但与普通 USB 读卡器不同的是，本文所述的读卡器基于 PCI Express 总线，是笔记本专用存储卡读卡器。本章主要从整体结构的角度来介绍 SD 卡控制器，并详述了各功能模块的功能及联系。

3.1 总体结构

图 3-1 为 SD 卡控制器总体结构图。

图的左边是连接到 PC 机的 PCI Express 总线，右边是 SD 总线，中间为 SD 控制器部分。本文重点研究 SD 控制器部分。该控制器由几个模块组成，分别为 SDIO、DMA 接口、时钟控制、Buffer 控制、Buffer RAM、Tuning，SD 数据命令控制模块。当进行读写操作时，SD 时钟控制模块提供时钟给其他模块。数据通过 PCI Express 总线经过 SDIO 或者 DMA 接口模块进入控制器的存储模块中，从 SDIO 模块进入为驱动模式，从 DMA 接口模块进入为 DMA 模式。此过程需要 Buffer 控制模块对 Buffer RAM 进行控制。同时将输出值（如 DMA 传输错误）传输到 SD 数据命令控制模块进行处理。

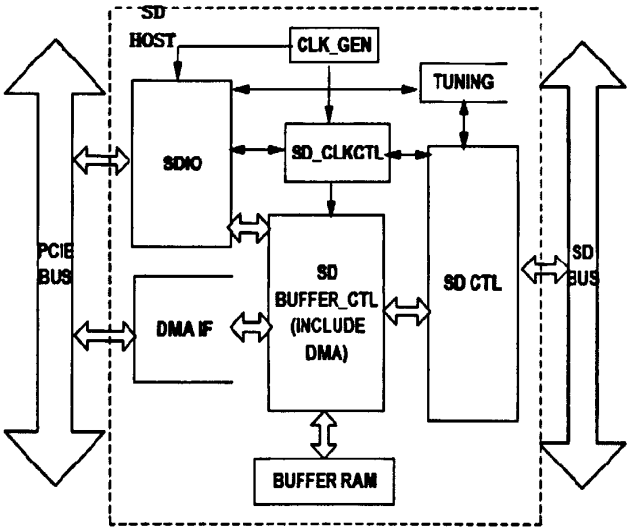


图 3-1　SD 控制器总体结构

然后 SD 数据命令控制模块与 SD 卡进行通信。此外，Tuning 模块涉及到时钟相位，其部分输入信号来源于 SDIO 模块，输出信号影响 SD 数据命令控制模块。

3.2 PCI Express 总线及 SD 总线

3.2.1 PCI Express 总线

PCI Express 是新一代的总线接口，它采用了目前业内流行的点对点串行连接，比起 PCI 以及更早期的计算机总线的共享并行架构，每个设备都有自己的专用连接，不需要向整个总线请求带宽，而且可以提高数据传输率，达到 PCI 所不能提供的高带宽[19]。PCI Express 的接口根据总线位宽不同而有所差异，包括 X1、X4、X8 以及 X16，PCI Express X1 规格支持双向数据传输，每向数据传输带宽 250MB/s[20,21]。本文采用 PCI Express X1 的规格。

3.2.2 SD 总线

卡有两种总线模式, 即 SD 模式和 SPI 模式，SD 模式采用 4 根数据线传输数据，数据传输快，但协议相对复杂；SPI 模式采用 1 根数据线传输数据，速度传输相对较慢, 但传输协议比较简单，操作比较简单[22,23]。本文采用 SD 模式进行数据传输。SD 模式下,SD 有 9 个引脚，分别为时钟、命令、数据、电源和地。

另外，SD 总线支持动态数据线的动态配置，在上电以后，SD 卡仅用 DAT0 作为数据传输，初始化后，控制器可以改变数据线的宽度，当 DAT1~DAT3 没有使用时，相关的输入数据线为三态。

3.3 控制器各模块功能说明

3.3.1 SD I/O 接口模块

这个模块主要完成 SD 控制器与 CPU 总线接口的功能及 SD 控制器标准寄存器的设置。

✧ 接口功能时主要产生以下信号

1) 产生总线宽度控制信号，总线宽度分别 32 比特，16 比特和 8 比特，通过总线宽度选择分别完成这三种宽度总线下数据的输出。

2) 产生字节使能信号，字节使能宽度为 4，分别使能 1 字节，2 字节和 4 字节数据。使能 1 字节数据有四种情况，分别为 0001,0010,0100,1000；使能 2 字节数据有两种情况，分别为 0011,1100；使能 4 字节数据有一种情况，即 1111。通过字节使能，使输入数据有效的载入。

3) 系统地址寄存器译码逻辑：根据地址输入的值选择寄存器地址，载入到 tmpDEC 中；CPU 寄存器读逻辑：将寄存器的值载入到 i_HD_O 中作为读出的数据输出；CPU 寄存器写逻辑：将 tmpDEC 的值载入到寄存器 DRV_SEL 作为要写入的寄存器地址，从 host 端输入的数据载入到 i_C_BUFD_O 后，根据 XBE[3:0]写入到选中的数据寄存器中。

4) 产生 SD 总线电压信号，电压分别为 1.8V 和 3.3V。其中 1.8V 主要用于 SDXC 卡在 UHS-I 五种模式下的传输。3.3V 主要为普通 SD 卡工作时电压。通过电压选择信号分别产生这两种电压。

5) 产生中断信号，例如 DMA 错误中断，数据和命令 CRC 错误校验，数据传输超时中断，插卡拔卡中断，命令完成中断，读写操作忙中断等。

✧ SD 控制器标准寄存器

1) 命令控制寄存器

此寄存器主要是命令寄存器，其中命令寄存器包含三种特殊命令，即暂停/恢复/停止命令，当设置为 01 暂停命令时，控制器需要读等待或者停止检测写忙碌。此时将会产生周期性中断。当设置为 10 恢复命令时，恢复数据传输，控制器也会在写操作前进行是否忙碌的检测。当设置为 11 停止命令时，控制器将停止一切读写操作，驱动软件复位。在响应类型选择时，针对三种特殊命令，分别返回不同长度的返回值。如表 3-1 所示。

表 3-1 命令控制寄存器

名称	比特位	类型	描述
Reserved	15:14	--	--
Command index	13:8	RW	命令索引作为命令的一部分被发送
Command type	7:6	RW	三种特殊命令，暂停/恢复/停止命令
Data pre selc	5	RW	1：当前数据可传 0：没有数据可传
CMD idx chk en	4	RW	1：检测命令索引返回值是否正确 0：不检测
CMD crc chk en	3	RW	1：检测响应命令中 crc 部分 0：不检测
reserved	2	--	--
Reps tyte selc	1:0	RW	响应类型选择

2) 缓存数据端口寄存器如表 3-2 所示。

通过 32 位的数据端口寄存器，控制器内部缓存可以被访问。

表 3-2 缓存数据端口寄存器

名称	比特位	类型	描述
Buffer data	31:0	RW	32 位数据端口寄存器

3) 软件复位寄存器如表 3-3 所示。

当写 1 到寄存器的每一个 bit 时便会产生软件复位脉冲，完成复位后，控制器将清除每一位。对于该寄存器的最低位，除了卡检测电路复位整个控制器。如果这一位为 1，驱动将发复位命令重新初始化 SD 卡。

表 3-3 软件复位寄存器

名称	比特位	类型	描述
Reserved	7:3	--	--
Soft dat rst	2	RW	缓存的读写，DMA 操作等复位
Soft CMD rst	1	RW	命令寄存器的复位
Soft rst for all	0	RW	所有复位除卡初始化检测阶段

除此以外，还有一些寄存器的设置，例如中断状态寄存器等等。在这里由于篇幅，不一一列举。

3.3.2 SD 控制模块

这一模块主要实现命令发布，CRC 校验（包括 CRC7 和 CRC16 的计算），SD 发命令控制，包括自动发 CMD12 的功能，不同状态下数据传输控制。在数据传输中，为了保证传输的正确性，采用 CRC 循环冗余校验码，其特征是信息字段和校验字段的长度可以任意选定。

CRC 码集选择的原则：若设码字长度为 N，信息字段为 K 位，校验字段为 R 位(N=K+R)，则对于 CRC 码集中的任一码字，存在且仅存在一个 R 次多项式 g(x)，使得

$$V(x)=A(x)g(x)=xRm(x)+r(x) \quad (3\text{-}1)$$

其中: m(x)为 K-1 次信息多项式， r(x)为 R-1 次校验多项式。

g(x)称为生成多项式：

$$g(x)=g0+g1x^{1}+g2x^{2}+...+g(R-1)x^{(R-1)}+gRx^{R} \quad (3\text{-}2)$$

发送方通过指定的 g(x)产生 CRC 码字，接收方则通过该 g(x)来验证收到的 CRC 码字[24]。

CRC 校验码软件生成方法： 借助于多项式除法，其余数为校验字段。可以除尽，即传输正确，否则，错误。

在本文中，通过 CRC 保护 SD 的命令和响应，传输的命令都会产生 CRC，同时也用于检测每一条响应。在这里，采用 CRC7（7 比特）和 CRC16（16 比特）。7 比特的 CRC 被用于所有的指令，所有的响应（除了 R3 类），和 CSD 及 CID 寄存器。CRC7 是一个 7 位的值，其计算方式如下：

生成器多项式:

$$G(x)=x^7+x^3+1 \tag{3-3}$$

主多项式：

$$M(x)=(第一位)*x^n+(第二位)*x^{n-1}+...+(最后位)*x^0 \tag{3-4}$$

$$CRC[6...0]=余数[(M(x)*x^7)/G(x)] \tag{3-5}$$

注：第一位是相关比特流（指令，响应，CID 或 CSD）的最左位比特。多项式中的 n 是 CRC 的保护比特位数减 1，对于指令和响应的保护比特数是 40，即 n=39；对于 CSD 和 CID 的保护比特数是 120，即 n=119。

对于数据传输，一个 16 比特的 CRC 会被用于每一个传输的数据块里。检查的结果将被记录在状态寄存器中。为了避免 CRC16 占据数据总线，其主要用于块传输模式中的过载保护。CRC 检查总数是一个 16 位值，其计算方式类似于 CRC7。

3.3.3 SD 时钟模块

时钟模块分三种情况，当插入 SD 卡后，先有一段卡识别过程，这个过程的时钟频率大概为 100K---400K；当识别卡后，进入卡的正常工作阶段，因为本文设计的时钟域较广，卡的工作频率范围为 25MHz---160MHz，所以可以提供较多的时钟频率供各种不同的 SD 卡工作。当拔卡后，时钟停止。插拔卡的过程可以通过状态机实现。当 SD 频率选定并且内部时钟寄存器的值有效时，通过判断内部时钟是否稳定，如果稳定则提供 SD 时钟。否则，还要继续等待。

为了保证 SD 时钟不会在紧急情况下出错，比如 FPGA 的 DCM 无法工作了，此时无法产生时钟供 SD 卡工作。或者因为时序问题，导致 SD 卡无法正常读写。在这些情况下，可以利用时钟备选方案。

方案一：可以设置一个带有分频参数的寄存器，设置其默认值为 2，直接对

目的时钟进行 2 分频处理。即原来目的时钟为 50MHz，可以让其工作在 25MHz。虽然频率降低，但是不会影响卡的正常读写工作。

方案二：设置 300M 备选时钟，通过内部分频得到 200MHz，100MHz 或 50MHz 频率。寄存器设置如表 3-4 所示。

表 3-4 150/300M DM/DN 控制选择寄存器

Bit(s)	Name	Access	Definition
31:30	RSVD	RW	Default 2'h0
29:24	dn_300m_cfg	RW	300M DN PLL configure selection, Default 6'h3B
23:21	RSVD	RW	Default 3'h0
20:16	dm_300m_cfg	RW	300M DM PLL configure selection, Default 5'h18
15:14	RSVD	RW	Default 2'b0

3.3.4 SD Buffer 控制模块

这一模块主要针对驱动和 DMA 操作两种模式下对外部 Buffer 的控制。分为三部分，一为 DMA 模式下的控制，二为驱动模式下的控制，三为这两种模式控制信号的选择。驱动模式主要产生一些标志位信号，同时有 CPU 的参与，会严重影响 SD 卡的传输速度。控制信号的选择主要功能是当选择 DMA 传输模式时，将 DMA 模块的输出信号作为外部 Buffer 的控制信号；当选择驱动传输模式时，将驱动模块的输出信号作为外部 Buffer 的控制信号。本文后面重点研究了 DMA 模式下的 Buffer 控制情况。

3.3.5 Tuning 模块

这一模块是本文研究的重点，Tuning 的功能即进行时钟相位的调整和选择，以便在高频下满足时序要求，完成数据传输。Tuning 只需要在 SDR50，SDR104 两种模式下操作。在进行 Tuning 前需要对 UHS-I 模式和电源进行选择。特别地，当选择 UHS-I 中的 SDR50，SDR104，DDR50 时，电源都要切换至 1.8V，一旦卡切换至 1.8V 模式，将不会转化到 SPI 模式。提供给卡的电压为 3.3V，但是提供给 SD 控制器的时钟、数据和信号线的电压从 3.3V 转化成 1.8V。为了避免卡和控制器电压不匹配，卡电压在初始化时可以通过命令 ACMD41 切换成 1.8V，如果此时卡和控制器的电压均为 1.8V 了，那么进入到 UHS-I 模式。当切换至 1.8V 模式后，还要判断提供的时钟是否已稳定，稳定后才能作为电压提供给 SD 卡。

3.3.6 DMA 接口模块

这个模块主要产生 DMA 与系统的接口信号，如 DMA 传输中的读写数据信号，DMA 请求，DMA 结束等信号。通过 DMA 数据总线选择信号来构建 DMA 字节使能信号。为了与 DMA 传输块 block 大小匹配，有字节计数器，选择一字节时，传输字节计数自动加 1，选择 2 字节加 2。依此类推。

第 4 章　重点模块的设计

前一章已简要介绍了 SD 控制器的总体结构及各功能模块，本章主要针对 SD3.0 新规范中更新的一些模块作详细的设计实现。从设计的目的、原理、过程及结果详细阐述了三个重点模块的设计过程。该部分的设计使 SDXC 卡的控制得以实现，也使 SD 卡控制器功能更加完整，读写性能得到提高。

4.1 时钟模块

4.1.1 采用 DCM 的目的

普通 SD 卡的工作频率为 25MHz 或者 50MHz，由于本控制器可支持市场上所有的 SD 卡，那么需要通过设计满足各种 SD 卡工作的频率。本文采用动态配置时钟的方法，可任意设置倍频以及分频系数，动态生成 25MHz~200MHz 的时钟，比如 SD 小容量卡（低于 32MB）的工作频率最高是 25MHz；SDHC 卡主要工作在 50MHz，对于最新的 SDXC 卡（64GB）中的 SDR50 和 SDR104 模式分别可以工作在 100MHz 和 160MHz。FPGA 有其特别的时钟管理模块即 DCM，本文采用 Xilinx VirtexⅣ160 芯片，通过逻辑控制 DCM 达到动态配置时钟频率的目的。

采用 DCM 的作用：1.实现零时钟偏移（Skew），消除时钟分配延迟，并实现时钟闭环控制；2.时钟可以映射到 PCB 上用于同步外部芯片，这样就减少了对外部芯片的要求，将芯片内外的时钟控制一体化，以利于系统设计[25]。3.可以任意设置寄存器分频系数，得到不同的频率，当实际工程中遇到问题使得 SD 卡不能工作在规定频率下时，可以通过降频的方式使 SD 卡继续工作。降频的方法：可以减小 M 值，增加 D 值或单独减小 M 值和增加 D 值。

4.1.2 时钟管理模块 DCM 原理

DCM 由四部分组成，分别为数字频率合成器(DFS)，DLL，数字移相器（DPS）和数字频谱扩展器(DSS)[26]。本文主要利用 DLL 和数字频率合成器来进行设计。

DLL 原理：DLL 主要由一个延时线和控制逻辑组成，延时线对时钟输入端 CLKIN 产生一个延时，时钟分布网络将该时钟分配到器件内的各个寄存器和时

钟反馈端 CLKFB；控制逻辑在反馈时钟到达时采样输入时钟以调整二者之间的偏差，实现输入和输出的零延时，锁定环路进入“锁定”状态，只要输入时钟不发生变化，输入时钟和反馈时钟就保持同步[27]。

数字频率合成器的原理：DFS 可以为系统产生丰富的频率合成时钟信号，输出信号为 CLKFB 和 CLKFX180，可提供输入时钟频率分数倍或整数倍的时钟输出频率方案，输出频率范围为 1.5～320MHz（不同芯片输出频率范围是不同的）[28]。这些频率是基于用户自定义的两个整数比值，一个是乘因子（CLKFX_ MULTIPLY），另外一个是除因子（CLKFX_ DIVIDE），输入频率和输出频率之间的关系为

$$F_{CLKFX}=F_{CLKIN}\times\frac{CLKFX\_MULTIPLY}{CLKFX\_DIVIDE} \tag{4-1}$$

4.1.3 DCM 配置

根据 DCM 原理，可以通过设置，生成不同频率的时钟信号，例如选择 SDXC 卡中的 SDR50 模式，输出频率为 100MHz，而选择 SDR104 模式时输出频率变成 200MHz。

下图为 Virtex-4 的 DCM 模块，具体设置如图 4-1 所示。

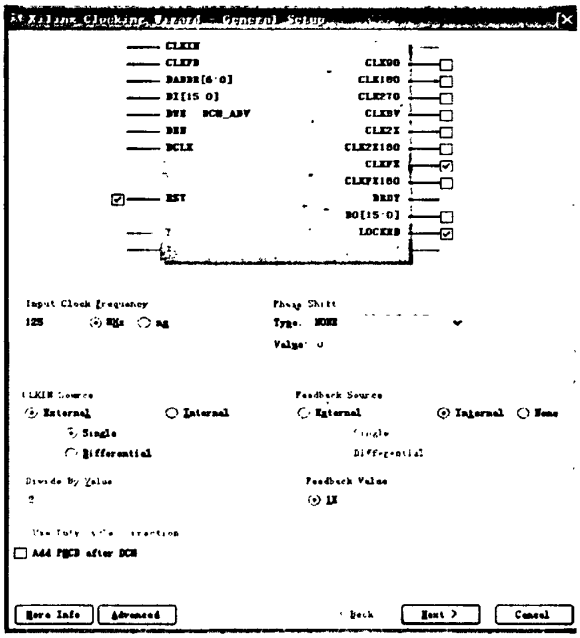


图 4-1 DCM 配置

本设计中利用其中以下引脚。引脚说明如下：

CLKIN（源时钟输入）：DLL 输入时钟信号，通常来自 IBUFG 或 BUFG。

CLKFB（反馈时钟输入）：DLL 时钟反馈信号，该反馈信号必须源自 CLK0 或 CLK2X，并通过 IBUFG 或 BUFG 相连。

RST（复位）：控制 DLL 的初始化，通常接地。

CLK0（同频信号输出）：与 CLKIN 无相位偏移；CLK90 与 CLKIN 有 90 度相位偏移；CLK180 与 CLKIN 有 180 度相位偏移；CLK270 与 CLKIN 有 270 度相位偏移。

LOCKED（输出锁存）：为了完成锁存，DLL 可能要检测上千个时钟周期。当 DLL 完成锁存之后，LOCKED 有效。

CLKFX(完整时钟输出)：通过设置 M(Multiply)和 D(Divide)的值，可以输出范围内的所有时钟值。

DO：当没有使用动态配置时，表示 DCM 的状态，使用动态配置时，用于数据输出。

DRDY:表示 DCM 动态配置是否 ready 的状态。

DI:数据的动态配置输入端口。

DADDR:动态地址配置。

DWE:写使能信号，控制往 DADDR 中写入 DI 数据。

DEN:动态配置使能信号。

DCLK：DCM 动态配置的时钟输入，其可以被任意时钟驱动。本设计中采用 CLKIN 驱动。

其中，IBUFG 和 BUFG 为输入时钟全局变量，FPGA 全局时钟资源一般使用全铜层工艺实现，并设计了专用时钟缓冲与驱动结构，从而使全局时钟到达芯片内部的所有可配置单元(CLB)、I/O 单元(IOB)和选择性块 RAM(Block Select RAM)的时延和抖动都为最小[29]。

根据数字频率合成器的原理，可以通过输入乘因子（M）和除因子（D），来改变不同的分频系数，这样就可以得到不同的频率。根据 Xilinx Virtex-4 中关于 DCM 模块中 M 和 D 参数的要求，M 取 1~64 的整数，D 取 1~32 的整数。所以为了得到本设计中 SD 卡的多种工作频率，需要计算出多个 M,D 的值。

由此可见，基本的输入信号只需要为 clkin，clkfb，rst，daddr，di，drdy，dwe，den，dclk，输出信号为 clkfx，locked，do 及 clk0。输入时钟频率为 125MHz，输出时钟假如为 100MHz，根据公式可以算出 M=4,D=5。当输出频率为 200MHz 时，M/D 为 1.6。

4.1.4 寄存器设置

所以在设计中需要输出不同频率的时钟信号，只需要改变 M/D 的值即可。在这里，M 位宽为 6，D 位宽用 5 来表示。特别地对于 SDR104 模式，寄存器配置如表 4-1 所示。

表 4-1 200MHz 时钟 SDR104 模式 M/D 寄存器设置

location	attrib	Register Field Explanation
15:14	Rsvd	Rsvd
13:8	RW	200M_M_sdr104_drv Default value is 6'h27 (6'd39).
7:5	Rsvd	Rsvd
4:0	RW	200M_D_sdr104_drv Default value is 5'h18 (5'd24).

4.1.5 逻辑控制 DCM

此部分主要利用状态机来实现对 DCM 的动态控制即 DRP，具体状态机如图 4-2 所示。

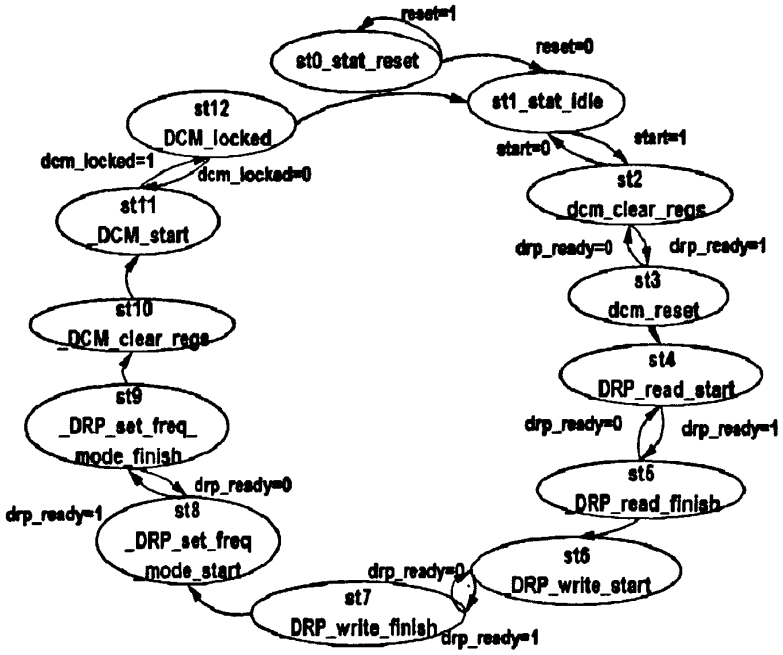


图 4-2 DCM 的动态配置状态机

状态机说明：

st0_stat_reset：状态机复位；

st1_stat_idle： 空闲状态；

st2_dcm_clear_regs：DCM 清除寄存器；

st3_dcm_reset：DCM 复位；

st4_DRP_read_start：DCM 动态配置读开始；

st5_DRP_read_finish：DCM 动态配置读完成；

st6_DRP_write_start：DCM 动态配置写开始；

st7_DRP_write_finish：DCM 动态配置写完成；

st8_DRP_set_freq_mode_start：DCM 动态设置频率模式开始；

st9_DRP_set_freq_mode_finish：DCM 动态设置频率模式完成；

st10_DCM_clear_regs：DCM 清除寄存器；

st11_DCM_start：DCM 开始；

st12_DCM_locked：DCM 锁定；

状态机复位，当 start 信号到来后，状态机从空闲态跳至清除寄存器状态，然后由于 DCM_DRP 动态配置 ready 信号置位，进入 DCM 复位状态，之后进入一系列的读写状态，在读写状态中，乘数(M)和除数(D)的值是通过 DRP 往地址 50h 写数来实现的，读写开始状态使能信号均为 0，读写完成后置 1。新的频率模式通过往地址 41h 写数来实现，同样，在开始设置频率模式状态，使能信号清 0，设置完成后，使能信号置 1，同时清除地址寄存器值。在清除寄存器状态后，进入 DCM 开始状态，当 DCM 锁存信号置 1 时，表明 DCM_DRP 完成，然后状态跳至空闲状态。

通过 ISE 设计工具可将此状态机例化成以下代码：

```
drp_stmach Inst_drp_stmach (
.clk(clk_in),
.start(drp_start),
.reset(~resetn),
.new_freq_mode(drp_change_mode),
.drp_ready(DRDY),
.drp_en(DEN_IN),
.drp_we(DWE_IN),
.drp_addr(DADDR_IN),
```

```
.drp_sel(DRP_SEL),
.dcm_locked(LOCKED),
.dcm_reset(dcm_reset),
.drp_done(drp_done)
);
```

4.1.6 4 分频设计

通过 DCM 产生了 200MHz 时钟，当不选择 UHS-I 模式时，只需要 SDHC 卡工作频率在 50MHz 或者 25MHz 时，可以通过逻辑设计，将 200MHz 时钟 4 分频，得到 50MHz 时钟；或者进行 8 分频，得到 25MHz 时钟。如图 4-3 所示。

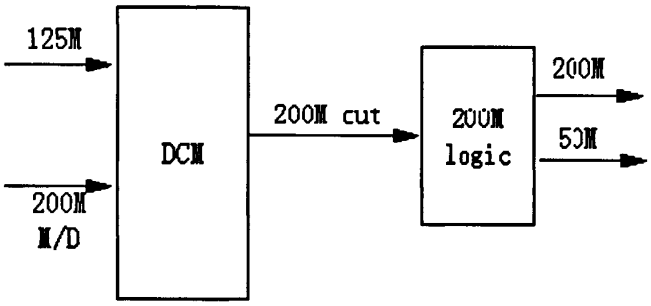


图 4-3 时钟分配

分频的思想可以通过一个脉冲计数器和 D 触发器构成。这里实现偶数分频，如实现 N 分频，计数到 N/2-1 时，触发器翻转一次即可。例如：4 分频即计数到 1 时，输出时钟翻转一次，利用 FPGA 设计工具 ISE，综合后的传输级结构及仿真图如图 4-4、图 4-5 所示。

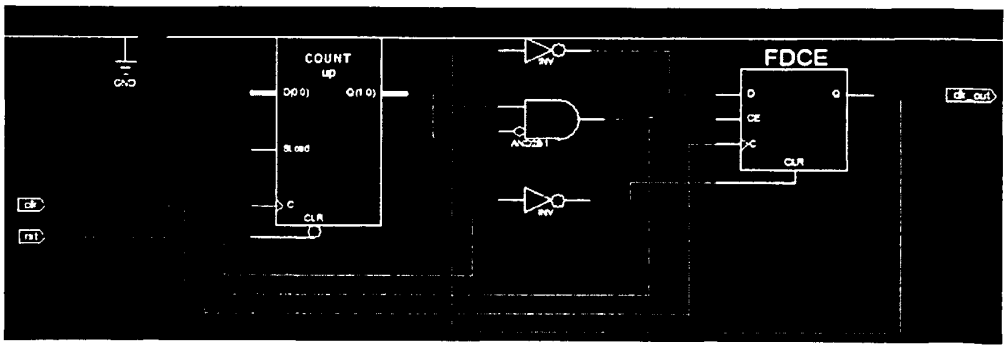


图 4-4 传输级结构

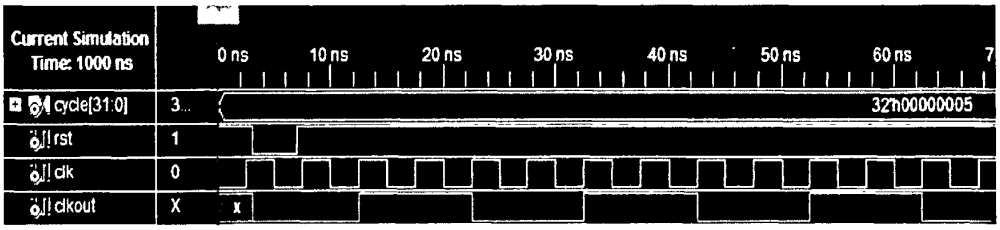


图 4-5 仿真图

由此可见，输入时钟为 200MHz，输出时钟为其四分频，即 50MHz。

4.2 Tuning 模块

4.2.1 Tuning 的目的

所谓 Tuning 即来源于它的英文意思"调整"，在此主要指调整时钟相位。本文主要研究和设计高频时钟下如何有效利用时钟来获得正确数据及如何在众多不同的相位时钟中选择一个最佳时钟。本设计主要针对 UHS-I 模式中的 SDR50 和 SDR104 模式，对应的工作频率分别为 100MHz，200MHz。当工作频率比较高时，相应时钟周期就会变小，对数据的建立和保持时间要求就比较严格。建立时间即触发器在时钟沿来到前，其数据输入端的数据必须保持不变的时间；保持时间即触发器在时钟沿来到后，其数据输入端的数据必须保持不变的时间。因为触发器内部数据的形成是需要一定的时间的，如果不满足建立和保持时间，触发器将进入亚稳态，进入亚稳态后触发器的输出将不稳定，其值在 0 和 1 之间变化，从而影响本级触发器的输出，同时也会导致亚稳态在下级触发器的传播，就不能保证系统输出的稳定性。所以，对于 SD 卡工作在高频并保证数据或命令信号在高频传输满足时序的情况下，本文提出一种选择相位时钟的算法来确定最佳采样时钟，保证能够满足数据的建立和保持时间。

4.2.2 Tuning 原理

主要原理即控制器先复位采样控制模块，驱动会发 CMD19 到指定寄存器。每当发一次 CMD19，会得到一组卡响应的数据，控制器选用内部一个时钟去采样，若采样得到的数据与已知原始数据（协议中规定的）一致，则认为数据采样成功，同时也表明内部出现一个正确的相位时钟。然后通过逻辑从众多正确时钟中选择一个最佳采样时钟作为接口采样时钟。15 个不同的相位时钟可通过

底层编辑器做延时产生。Tuning 这块逻辑的功能即从 15 个不同的相位时钟中选择一个时钟作为最佳采样时钟。

4.2.3 Tuning 流程

图 4-6 为 Tuning 过程的整体流程图。

首先 driver 控制器需要选择 UHS-I 模式（driver 控制），然后 driver 通过设置寄存器 execute Tuning 为 1 让 host 控制器开始执行 Tuning。driver 去写 CMD 寄存器，接着 driver 不断的发 CMD19 这个命令直到 host 控制器将 execute Tuning 复位为 0.而在发 CMD19 的过程中，卡会发一些数据到控制器中，控制器将这些数据与协议中的数据对比，如果相等则表明数据采样成功，即此时钟满足数据时序要求，为其中一个正确时钟。相应将该位的值(1 有效，表响应了 CMD19)作为输入端口传给硬件。然后是硬件控制器对选择正确采样时钟的逻辑设计。当 Tuning 完成或者 Tuning 没有在 40 次以内完成时，host 控制器都将 execute Tuning 复位为 0。直到 CMD19 命令达到 40 次或者超过 150ms，Driver 会停止发 CMD19，在这种情况下，通过逻辑设计必须选择一个正确的采样时钟（sampling clock select）。当其值为 1 时表明 Tuning 过程成功。否则，Tuning 失败。所以整个过程的关键在于如何通过硬件逻辑设计选择一个正确的采样时钟。下面将详细介绍硬件逻辑设计部分。

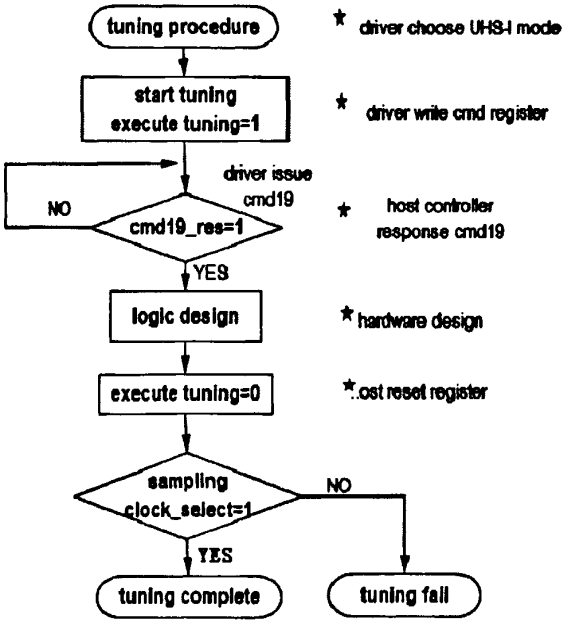


图 4-6 Tuning 流程图

在 Tuning 序列正在进行时，控制器不可能产生中断，除非 CMD19 响应错误。当发现 Tuning 错误中断信号时，driver 会通过清除采样时钟选择为 0 来复位 Tuning 电路，然后再重新执行 Tuning 这个过程，即 retuning。本文不重点研究 retuning 的情况。

4.2.4 硬件逻辑设计

✧ Tuning 过程硬件逻辑部分 I/O 端口介绍，如表 4-2 所示。

表 4-2 I/O 端口

I/O 端口名称	宽度	输入、输出	说明
XRST	1	I	复位信号
OPELK_I	1	I	输入操作时钟
C19_STRT	1	I	在 CMD19 开始发送后，这个信号作为逻辑部分的输入信号。
C19_COUNT	[7:0]	I	发 CMD19 命令的次数。
C19_END	1	I	CMD19 发完后也作为逻辑设计的输入信号
C19_RES	1	I	当控制器正确提取一组 Tuning 的数据后产生信号输入到逻辑设计
UHSI_MODE	[2:0]	I	控制器选择 UHS-I 模式，也是逻辑设计输入信号
C19_MAXCNT_O	[7:0]	O	Tuning 过程发 CMD19 最多的次数(40 次)
TAP_COMP	1	O	TAP setting complete. 当前一个 CMD19 发完后，延时一个周期，产生此信号作为发下一个 CMD19 的标志位.
TUNE_END	1	O	当发完 40 次 CMD19，Tuning 过程结束
TUNE_OK	1	O	当最佳采样相位时钟找到，用这一位作为输出信号

✧ 最佳采样时钟选择算法

对于逻辑设计部分，即如何完成 Tuning 操作这个过程变为设计的重点，其中最佳采样时钟的选择更是重中之重。在选取最佳采样时钟的过程中，主要设计算法如下：每当发现正确的采样时钟记为 1，错误记为 0，从连续的 1 中选取最中间的一个 1，选取时需要确定正确时钟的起始点和连续正确时钟个数的一

半，然后相加得到的值对应其相位就可以知道哪一个时钟是最佳采样时钟了。

✧ 15 个相位时钟正确与否的判断

Host 最多发 40 次 CMD19，当 CMD19 发完后每次都将返回一个值（C19_RES）即逻辑设计部分的输入端，将每次的返回值放至返回寄存器的最高位，依次循环右移一位，最后 40 位的返回寄存器存放 40 个返回值。

代码如下：

```
if (C19_END)
begin
      TUNING_RES[38:0] <= TUNING_RES[39:1];
      TUNING_RES[39] <= C19_RES;
    end
```

因为只有 15 个不同的相位时钟，而第 0,15,30 次返回的响应值对应同一个时钟，15 个相位时钟的正确与否用“与”逻辑进行判断。将对应同一个时钟的返回值相与，最后得到一个用 15 位表示的返回值（TUNING_RES_SUM[14:0]）。

部分代码如下：

```
if (TUNING_ST == `TUN_CALC) begin

TUNING_RES_SUM[0]<=TUNING_RES[0]&TUNING_RES[15]&TUNING_RES[
30];

TUNING_RES_SUM[1]<=TUNING_RES[1]&TUNING_RES[16]&TUNING_RES[
31];
 .....（后面类似）
TUNING_RES_SUM[14]<=TUNING_RES[14]&TUNING_RES[29];
end
```

✧ 连续正确时钟的判断

正确相位时钟的判断即从以上 15 个返回值中判断连续 1 的情况，确定一个正确时钟的有效窗口。100M 和 200M 时钟用不同的寄存器值来表示有效窗口。比如设置 200M 时钟用 5 来表示出现 5 个连续 1 的区域即为采样的有效窗口。

有效窗口判断的部分代码如下：

```
 if ((TUNING_ST == `TUN_PRE_SELC || TUNING_ST == `TUN_SELC ||
TUNING_ST == `TUN_SELC1) && !TUNE_SEL_DONE) begin
     case (TUN_WINDOW_SET)
       4'h1 : begin
         if (TUNING_RES_SUM[0] == 1'b1)  TUN_WINDOW_CHECK = 1;
```

```
else   TUN_WINDOW_CHECK = 0;
        end
        4'h2 : begin
          if (TUNING_RES_SUM[1:0] == 2'b11)   TUN_WINDOW_CHECK = 1;
else   TUN_WINDOW_CHECK = 0;
        end
 ..........（类似省略）
        4'hf : begin
         if(TUNING_RES_SUM[14:0]==15'b111_1111_1111_1111)
TUN_WINDOW_CHECK = 1;   else   TUN_WINDOW_CHECK = 0;
        end
end
```

由此可见，根据不同频率有效窗口设置的大小，连续 1 的个数分为 15 种情况，两个 0 之间一定存在连续的 1，在状态机中不同状态的控制下不断通过循环左右移，将得到的连续 1 移至低位端，直到 TUNING_RES_SUM 从 0 位起一直为 1，那么就判断找到了连续的 1，例如：发完 40 次 CMD19 后，得到的 TUNING_RES_SUM[14:0]="1110 1011 1111 100"那么通过循环右移位，得到"0011 1010 1111 111" TUNING_RES_SUM[8:0]均为 1，很明显找到了连续的 1。

✧ 最佳采样时钟的选择

以上得出的 TUNING_RES_SUM 的值是随机的，有可能其值中只有连续 3 个 1 或者连续 7 个 1，所以要分 2 种情况考虑。一种为加调整，一种为减调整。此时与开始设置的有效窗口进行比较，多加少减，同时根据在不同状态的移位情况，就可以确定连续 1 的个数。直到通过循环移位发现不满足连续 1 的条件时，Tuning 完成，例如窗口有效寄存器设置的值为 5，那么 TUNING_RES_SUM[4:0]="1110 0111 0101 111"时结束。通过设置变量可以得到起始采样点及连续 1 个数，用连续 1 的个数除以 2， 再通过相加即可得出最佳的采样时钟对应相位值，即确定了最佳采样时钟。

✧ 逻辑部分状态机如图 4-7 所示。

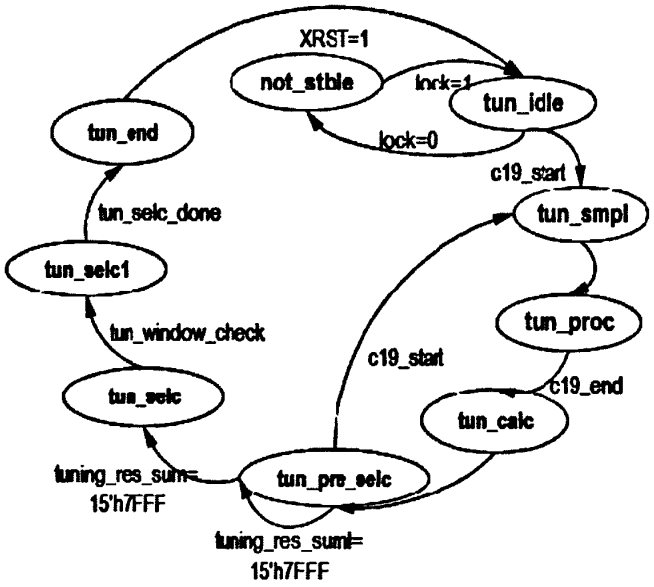


图 4-7 状态机

状态机设计说明：

NOT_STBL（ST0）：状态机不稳定状态；

TUN_IDLE（ST1）：Tuning 空闲准备状态；

TUN_SMPL（ST5）：Tuning 开始阶段；

TUN_PROC（ST2）：Tuning 过程阶段；

TUN_CALC（ST3）：Tuning 计算过程；

TUN_PRE_SELC（ST8）：Tuning 找到连续时钟的过程；

TUN_SELC（ST4）：Tuning 采样判断阶段（主要判断正确时钟的起点位置）；

SELC1（ST7）：Tuning 采样确定采样点阶段；

TUN_END（ST6）：Tuning 过程结束；

具体过程如下：

复位信号有效时，先进入到状态机不稳定的状态，当 dll 的 lock 信号为高时证明 dll 输出稳定，此时进入到 Tuning 的空闲状态，准备开始进行 Tuning 操作。一旦 CMD19 信号已经发出后，进入到 Tuning 开始阶段，否则当 lock 信号不稳定时仍然回到状态机不稳定状态。当 lock 信号稳定，则进入到 Tuning 过程阶段。当 40 次 CMD19 均发完后，进入到 Tuning 计算阶段。计算阶段即把 c19 的响应信号相与得出一个 15 位的响应结果(前面有具体说明)。接着进入到 Tuning 的时钟选择阶段，之所以设置这个状态，是因为要分两种情况来考虑。一，如果 15

个采样时钟都能采到正确数据，即 15 个 1（特殊情况）。二，得到的响应中出现过 0，即不是 15 个相位时钟均满足。在没有出现 15 个相位时钟均满足的时候，又需要考虑如何找到连续正确时钟的问题，所以又分两种情况考虑：一，找到有效窗口即有效窗口判断为 1，二，没有找到有效窗口即有效窗口判断为 0；当有效窗口判断为 1 时，还需要考虑，是否有更多的 1 可以通过循环移位放到一起，比如“1110 1001 1111 111”，从这个序列中可以发现最高位的 3 个 1 可以通过循环左移移至到低位，形成一个连续 12 位的 1。所以把这个阶段设置成时钟选择前确定正确连续时钟的过程。直到发现有效窗口判断为 0 即正确时钟不再连续后进入下一个阶段（判断正确时钟的起点）。这个过程比较简短，此时若有效窗口从 0 变为 1，进入到选择最佳采样点阶段，在这个阶段主要通过移位的次数来确定连续 1 的个数，将移位次数除以 2 即找到连续 1 的中点位置。当移位后有效窗口再次为 0 时，判断结束。此时也通过设计思路中提到的计算得出了采样点。

✧ Tuning 模块仿真结果及分析

本文采用 FPGA 设计软件 ISE 自带仿真器进行仿真，仿真长度采用 1000ns，为了能够模拟出输入端口的信息，在测试代码中直接给定了一些值，并且为了方便，本仿真将发 40 次的 CMD19 改为了 15 次，同时为了能够看出状态机如何跳变以及采样点如何选择，在做仿真时直接给出了 15 位返回值的初始值即 TUNING_RES_SUM[14:0] =” 111010111111100”。总体仿真结果如图 4-8 所示。

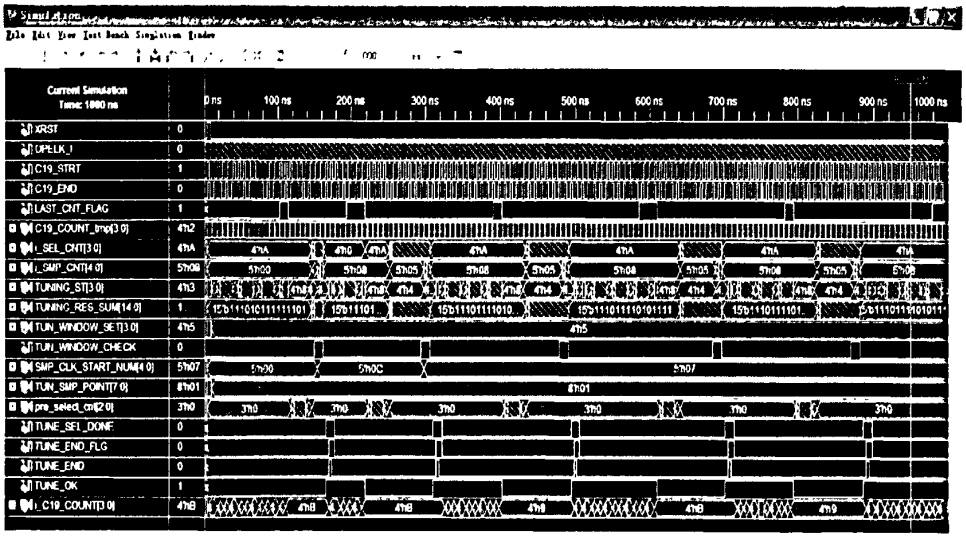


图 4-8 总体仿真图

如图 4-9 所示为相位时钟采样中部分状态机的变化及具体采样的结果。

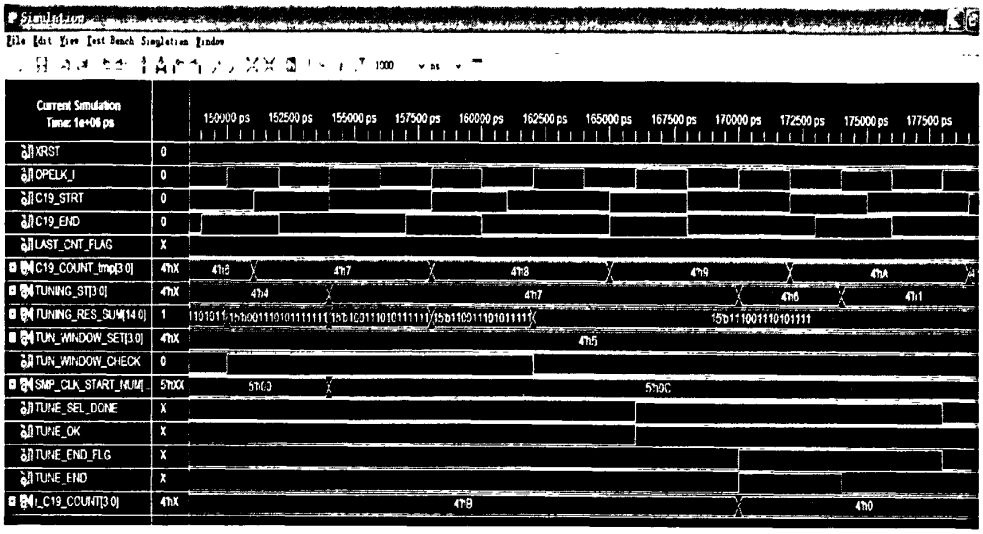


图 4-9 具体采样图

从仿真图可以看出，选择的结果在第七个状态也就是(TUN_SELC1)时，当窗口有效信号变为 0 时，采样结束；当下一个时钟沿到来的时候，可以得出相位的值。同时状态机运转完全正确，通过对比可以得出，0 即对应第五个时钟（规定好的值）。开始这个序列为“111010111 1 11100 ”，从连续的 7 个正确时钟中我们便选出了最中间的一个时钟，把它作为最佳采样时钟。

4.2.5 逻辑综合结果及分析

本设计采用的芯片为 Xilinx Virtex4 系列的芯片，芯片型号为 XC4VLX160, 封装为 FF1148,速度等级为 10。本设计采用第三方专用综合软件 Synplify Pro9.6.1 对设计进行逻辑综合。

综合后的时序分析如图 4-10 所示。

Performance Summary

Worst slack in design: -1.893

Starting Clock	Requested Frequency	Estimated Frequency	Requested Period	Requested Period	Slack	Clock Type	Clock Group
OPELK_I	200.0MHz	115.1MHz	5.000	6.803	1.893	declared	default_clkgroup_0

图 4-10 综合时序报告

通过时序报告，如图所示，可以看出，设计需要的时钟周期为 5ns，即频率为 200MHz,但是当加入周期约束后，预估计的频率仅仅为 145MHz，报告也给出了 slack 值即超出规定时钟周期的值为-1.893。所以没有符合设计的时序要求。可以通过添加约束的方法来达到要求。

约束有以下 3 种作用：

1) 提高设计的工作频率

对很多数字电路设计来说，提高工作频率非常重要，因为高工作频率意味着高处理能力。通过附加约束可以控制逻辑的综合、映射、布局和布线，以减小逻辑和布线延时，从而提高工作频率。本设计主要利用这个作用。

2) 获得正确的时序分析报告

几乎所有的 FPGA 设计平台都包含静态时序分析工具，静态时序分析工具以约束作为判断时序是否满足设计要求的标准，因此要求设计者正确输入约束，以便静态时序分析工具输出正确的时序分析报告。

3) 另外通过约束还可以指定 IO 引脚所支持的接口标准和其他电气特性，以便与外围设备较好连接。

时序约束的一般要求：

附加时序约束的一般策略是先附加全局约束，然后对快速和慢速例外路径附加专门约束，附加全局约束时，首先定义设计的所有时钟，对各时钟域内的同步元件进行分组；对分组附加周期约束，然后对 FPGA/CPLD 输入输出 PAD 附加偏移约束、对全组合逻辑的 PAD TO PAD 路径附加约束；附加专门约束时，首先约束分组之间的路径，然后约束快、慢速例外路径和多周期路径，以及其他特殊路径[30-33]。

本文使用 FPGA 综合工具 Synplify9.6.1 的约束工具 SDC 进行约束。Synplify SDC 提供了丰富的词表来定义并编译逻辑相关的设计单元，包括 I/O 端口、时钟网络以及同步 RTL 结构推出的寄存器单元，它提供了针对时序逼近的重要检测点，是满足最终速度目标的一个好的指示器。

下面通过比较添加约束前后的差别作详细分析：

通过查看综合时序报告，可以看出最差路径的信息（包括路径起点及终点，路径延时时间，数据建立时间，路径总延时等等）。所以便于添加约束条件。

本设计先加全局约束即周期约束（在本模块设计中，只有一个时钟域，所以不需要对时钟进行分组）。再使用“FROM_TO”的语句对局部电路附加较松的约束即附加专门约束。一般情况下局部电路附加的约束都要比全局约束松。

利用 synplify 添加时序约束后时序报告，如图 4-11 所示。

Performance Summary

Worst slack in design: 0.686

Starting Clock	Requested Frequency	Estimated Frequency	Requested Period	Required Period	Slack	Clock Type	Clock Group
OPELK_I	200.0MHz	231.8MHz	5.000	4.314	0.686	declared	default_clkgroup_0

图 4-11 添加约束后的时序报告

通过添加时序约束后，可以明显看出，此时的预测频率为 231.8MHz，高于 200MHz，所以可以满足设计的时序要求。同时也可以发现 slack 值变为正值，即时序正常。

再来看一下资源利用报告：

添加约束后与之前相比如表 4-3 所示。

表 4-3 前后资源报告比较

	添加约束后	添加约束前
Total LUT	244 个	281 个
BUFGP	1 个	1 个
Register	77 个	86 个
IBUF	58 个	58 个
OBUF	65 个	65 个
Total load per clock	77 次	86 次

通过对比，可以发现在逻辑资源上，添加约束后，也相应变少了。所以该模块设计，即完成了逻辑功能同时也满足了时序要求。

4.3 DMA 模块

4.3.1 DMA 的目的

SD 卡控制器的设计除了时钟是个重点外，数据传输也是一个关键问题，如何提高数据传输速率从而满足数据处理的准确性成为设计的重点。

传统的 PC 机与 SD 卡通信中，有如下两种传送方式。

a) 查询传送方式，CPU 与 I/O 设备往往是异步的，很难保证 CPU 执行输入操作时，外设已把要输入的信息准备好了，而当 CPU 执行输出时，外设寄存器一定是空的，所以，在程序控制下的传送方式，必须要查询外设的状态，接口部分除了数据传送的端口外，还必须有传送状态信号的端口[34]。

b) 在查询传送方式中，CPU 要不断地询问外设，当外设没有准备好时，CPU 要等待，不能干别的操作，这样就浪费了 CPU 的时间，而且外设的速度一般都比较低，所以产生了中断的传送方式。但中断方式仍然是由 CPU 通过程序传送的，每次要保护断点。保护现场需要多条指令，每条指令要有取指和执行时间。对于一个高速的 I/O 设备，就显得速度太慢了。

综上，为了提高外设与系统内存通信的速度，希望用硬件在外设与内存之间直接进行数据交换，而不通过 CPU。这种方式即为 DMA------直接存储访问。

4.3.2 DMA 的原理

DMA 的主要原理：通常系统的地址和数据总线以及一些控制信号线（例如：/RD./WR 等）都是由 CPU 管理的。在 DMA 方式时，就希望 CPU 把这些总线让出来（即 CPU 连到这些总线上的线都处于第三态--------高阻状态），而由 DMA 控制器接管[35,36]。本文主要通过硬件来实现 SD 控制器部分的 DMA 控制。DMA 方式主要利用共享式存储器实现，但不能同时访问存储器。本文采用 SDMA 的方式进行传输，SDMA 即每当到达系统存储区的边界时，控制器产生 DMA 中断来请求驱动（软件完成）更新系统地址寄存器的值，并且在系统边界处产生 DMA 等待信号。

4.3.3 两个重要寄存器的配置

表 4-4 block 大小寄存器

D15	D14 D12	D11 D00
Rsvd	HostDMA Buffer boundary	Transfer block size

如表 4-4 所示，其中 D15 为保留位，D[14:12]为 SDMA 缓冲器块长度界限的设置，表示系统存储器的连续缓冲区的大小。D[11:0]即传输块大小的设置。SDMA 传输必须在该位指定的边界处等待。具体设置如表 4-5 所示。

表 4-5 SDMA 具体设置

000b	4K bytes	(Detects A11 carry out)
001b	8K bytes	(Detects A12 carry out)
010b	16K bytes	(Detects A13 carry out)
011b	32K bytes	(Detects A14 carry out)
100b	64K bytes	(Detects A15 carry out)
101b	128K bytes	(Detects A16 carry out)
110b	256K bytes	(Detects A17 carry out)
111b	512K bytes	(Detects A18 carry out)

例如：当该位设置为 000b 时，缓冲器的大小为 4k 字节。那么字节地址的低 12 位指向连续缓冲区中的数据。高 20 位指向系统存储区中页所在地址。当控制器检测到执行的地址从第 11 位增加到第 12 位时，将停止 SDMA 传输。显而易见，D[11:0]也可以任意设置传输数据块的大小。本文设置的边界为 512K 字节。

另外一个重要的寄存器即传输块计数寄存器。如表 4-6 所示。

表 4-6 block 数量寄存器

D15	D00
Blocks count for current transfer	

此寄存器由 16 位组成。当数据量较大需要多块 block 一起传输并且块计数使能为 1 时，驱动会设置此寄存器的值。用 0000h~ffffh 来表示 0~65535 个 block。控制器在每块数据传输之后使块计数自减 1，当减到 0 时停止。当此寄存器设置为 0000h 时表示没有数据块传输。只有当没有传输进行时，该寄存器才可以访问（如，传输停止后）。

在数据传输过程中，数据长度的确定也是通过这两个寄存器来设置的，控制器与 SD 卡的缓冲器大小十分重要。卡的缓冲器大小由卡的特殊数据寄存器（CSD）最大块长度来设置。尽管卡能够提供的最大块长度大于 512K 字节，但是传输数据的块长度仍然要为 512K 字节，因为控制器规定的最大 DMA 传输块长度即为 512K 字节。传输数据的最大长度有如下公式

$$\text{Max data size=block size *block count} \tag{4-2}$$

即数据传输的大小等于块数量与块大小的乘积。

4.3.4 DMA 控制结构

如图 4-12 所示，该图左边部分连接系统（A 端），右边连接 SD 卡（B 端），中间为 SD 卡 DMA 控制部分。中间由三部分组成，分别是 DMA 总线宽度控制，双口 RAM 及 Buffer RAM 控制部分。

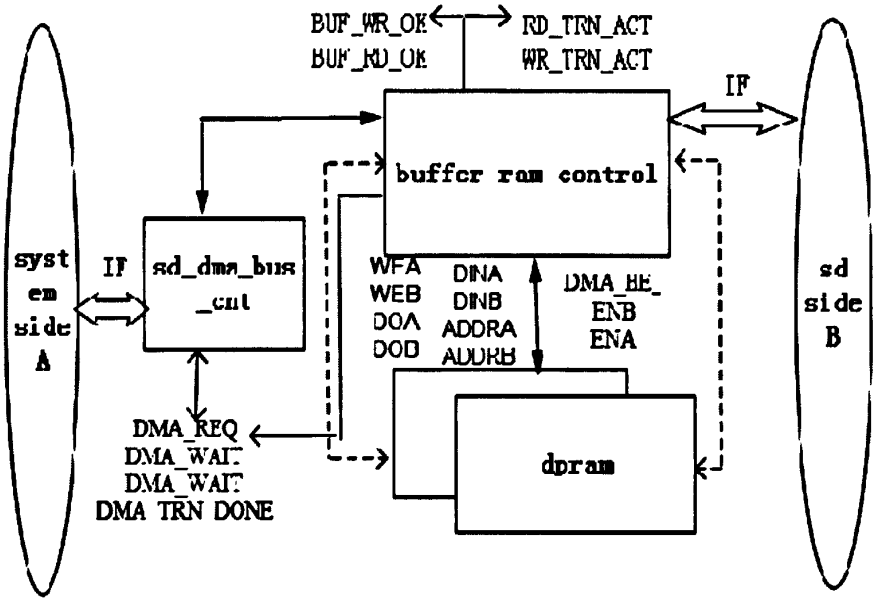


图 4-12 DMA 控制结构图

DMA 总线宽度控制主要控制不同总线宽度的输入输出数据以及 DMA 字节使能。输入输出数据受 DMA 字节使能控制。

总线宽度分为 32 位，16 位，和 8 位。16bit 模式时，分两种情况，一种为高两位字节使能，一种为低两位字节使能。即 DMA_BE=4'b0011、4'b1100。8bit 模式时，则有四种情况，即分别只有一位字节使能，即 DMA_BE =4'b0001、4'b0010、4'b0100、4'b1000。

双口 RAM 主要用作数据存储。本文采用两片 512*32 的双口 RAM 存放数据，由两片相同大小的双口 RAM 构成。一片 RAM 的大小为 512*32，由 512*8 的基本RAM扩展而来。所以总共大小为 1024*32 的RAM需要 8 片 512*8 的RAM 进行扩展。

最重要的部分即 Buffer RAM 的控制，其主要功能是对双口 RAM 进行控制，产生读写使能信号，标志信号，DMA 接口信号等。同时还有同步控制，因为两边时钟不同，需要同步数据指针，即将 A 端的写指针同步到 B 端时域内。A 端

判断数据 block 标志信号同步至 B 端。除此以外，还产生 DMA 错误，DMA 等待，DMA 完成等信号供 DMA 总线宽度控制模块用。

由此可见，双口 RAM 的设计及其控制部分是设计的重点。所以本文用硬件描述语言对数据存储及其控制部分作了详细设计并实现其功能。

4.3.5 双口 RAM 的设计

✧ 双口 RAM 的原理

双口 RAM 是在一个 SRAM 存储器上具有两套完全独立的数据线、地址线和读写控制线，并允许两个独立的系统同时异步地对该存储器进行随机性的访问，即共享式多端口存储器，双口 RAM 最大的特点是存储数据共享[37-39]。

✧ 双口 RAM 读写操作

双口 RAM verilog 读写过程代码如下所示：

1. 写操作：

```
always@(posedge CLKA)
    begin:DAT_WR_A_Side
        if (ENA && WEA)        memory[ADDRA]   <= DINA;
    end
always@(posedge CLKB)
    begin:DAT_WR_B_Side
        if (ENB && WEB)        memory[ADDRB]   <= DINB;
    end
```

2. 读操作：

```
always@(posedge CLKA)
    begin:DAT_RD_A_Side
        if (ENA)               DOUTA<= memory[ADDRA];
    end
always@(posedge CLKB)
    begin:DAT_RD_B_Side
        if (ENB)               DOUTB<= memory[ADDRB];
    end
```

双口 RAM 通过 FPGA 设计工具 ISE 综合后的传输级结构如图 4-13 所示。

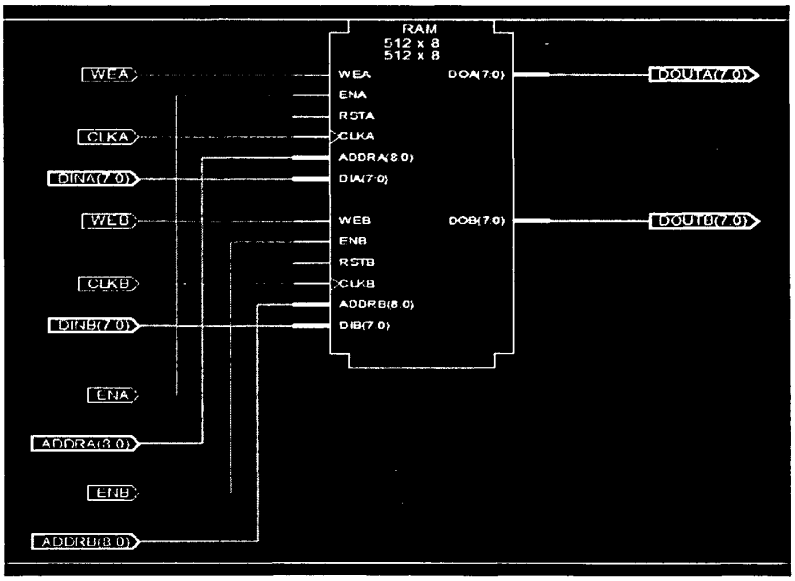


图 4-13 双口 RAM RTL 结构图

双口 RAM 通过 FPGA 设计工具 ISE 仿真后的结果如图 4-14 所示。

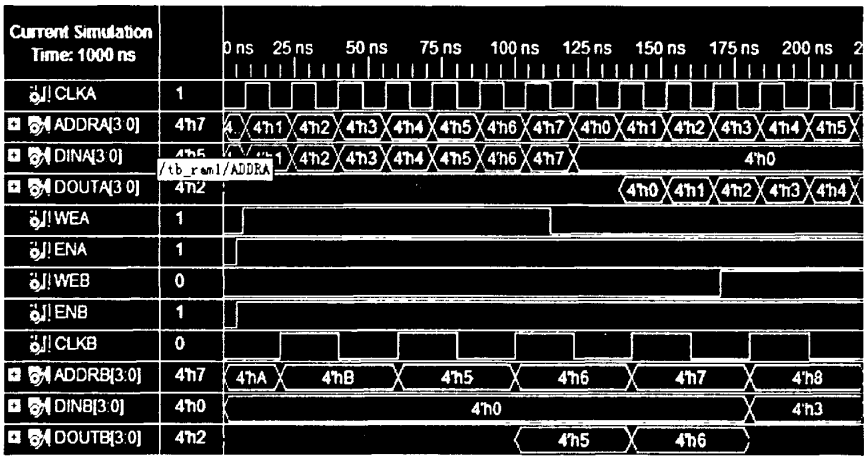


图 4-14 双口 RAM 仿真图

为快速观看功能仿真是否正确，我们假设地址宽度为 4，数据宽度也为 4，CLKA, CLKB 周期分别为 8ns，20ns 即对应 125MHz 和 50MHz。从图中可以看出，先向 ADDRA 写数据，然后从其中读出数据，B 端先从与 A 相同的地址中读出数据，然后进行写操作。从波形来看，符合设计要求。

✧ 双口 RAM 的扩展

因为本设计是在系统内存与 SD 卡之间进行通信，为使两边可以很好解决高速可靠性传输问题，需要设计一个 RAM 可以满足在两边时钟不同步的情况下，所以采用 RAM，双口 RAM 的深度为 512，宽度为 8。所以 A,B 口各需要 9 位地址线来表示，又因为数据量较大，且数据总线的宽度为 32 位，所以要进行 RAM 字和位的扩展。将其扩展成 1024*32 大小的 RAM。所以共需要 8 片 512*8 的 RAM 构成或者 2 片 512*32 的 RAM，本文先通过位扩展成 512*32 的 RAM，然后再通过字扩展成 1024*32 的 RAM。扩展后 ISE 综合传输级图如图 4-15 所示。

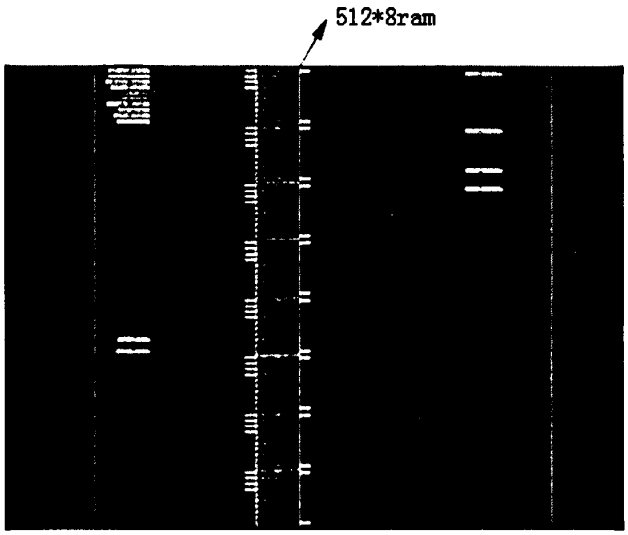


图 4-15 RAM 扩展后 RTL 图

由此可见，利用 8 片 512*8 的 RAM 扩展成为了 1024*32 的 RAM。

4.3.6 Buffer RAM 控制

✧ Buffer RAM 逻辑 I/O 端口说明，如表 4-7 所示。

表 4-7 I/O 端口

I/O 端口名称	宽度	输入、输出	说明
XRST_ALL	1	I	复位信号
SYSCLK	1	I	System side 时钟
OPECLK	1	I	SD side 时钟
DMA_EN	1	I	DMA 操作使能
BUF_WR_OK	1	O	表示 Buffer 可写，即 Buffer 没有满
BUF_RD_OK	1	O	表示 Buffer 可读，即 Buffer 没有空
DMA_BE	4	I	DMA 模式下 system side 字节访问使能

I/O 端口名称	宽度	输入、输出	说明
SECT_ADR_A SECT_ADR_B	1	O	A 端 RAM 选择指针，0 表示选择 RAM0,1 表示选择 RAM1，B 端类似
WEA_01	2	O	A 端写使能，低位使能 RAM0，高位选择 RAM1
WEB_0，WEB_1	4	O	B 端写使能，WEB_0 使能 RAM0，WEB_1 使能 RAM1
ENB_0，ENB_1	4	O	B 端读、写可控使能
SY_BUFD_I	32	O	System side Buffer 读数据
SD_BUFD_I	32	O	SD side Buffer 读数据
DOA_0，DOA_1 DOB_0，DOB_1	32	I	A、B 端数据输出端口
DMA_REQ	1	O	DMA 请求
DMA_WAIT	1	O	传完一个 block 数据的标志位
DMA_TRN_DONE	1	I	DMA 传输完毕

✧ 使能端控制

当向 SD 卡进行写操作时，B 端相当于读 RAM，在 RAM 未空的情况下 ENB_0，ENB_1 作为读使能。反之，当从 SD 卡读数据时，B 端相当于写 RAM，此时，ENB_0，ENB_1 与 B 端写使能 WEB_0，WEB_1 相等。而从上述双口 RAM 的写控制来看，只有当写使能和使能均有效时，才可以对 RAM 进行写操作。当使能信号有效时，进行读操作。对于 WEA,WEB 写使能，均有 A，B 端的写标志位控制，所以不可能存在同时向同一地址写数据。同时对于同一段空间，写完后才能进行读操作。例如对于 RAM0：

```
assign  WEB_0[0]     =  (SD_BUF_WR[0] & (~SECT_ADR_B));
.....
assign  WEB_0[3]     =  (SD_BUF_WR[3] & (~SECT_ADR_B));
assign WEA_01[0]  =  (SYD_BUF_WR & (~SECT_ADR_A));
```

✧ 数据流控制

设置 A,B 端数据指针，向 SD 写数据时，用 A 端数据指针低位来判断用哪个 RAM，0 表示用 RAM0,1 表示用 RAM1。当检测到已是 block 的结尾，A 端数据指针加 1，指向 RAM1。从 SD 读数据时，B 端指针当读完一个 block 数据后自加 1，用 B 端数据指针低位来判断用哪个 RAM，0 表示用 RAM0,1 表示用 RAM1.

控制写过程：当从系统内存向 SD 卡写数据时，首先通过 A 端 RAM 选择指针选择 RAM0，打开 A 端写使能，数据从 RAM0 A 口的输入数据端口进入，此时，B 端读使能打开，数据从 RAM1 B 端的输出数据端口输出。当 A 端检测到一个 block 的数据即（512byte）已传完，指针指向 RAM1，开始对 RAM1 进行类似操作。RAM0 和 RAM1 轮流进行。

控制读过程：当从 SD 卡向系统内存读数据时，首先通过 B 端 RAM 选择指针选择 RAM0，打开 B 端写使能，数据从 RAM0 B 口的输入数据端口进入，此时，A 端读使能打开，数据从 RAM1 A 端的输出数据端口输出。当 B 端检测到一个 block 的数据即（512byte）已传完，指针指向 RAM1，开始对 RAM1 进行类似操作。RAM0 和 RAM1 轮流进行。

✧ Buffer RAM 空满状态控制

分为空和满两种状态的判断。通过比较 A、B 端指针，若两端指针的低位相同，即读指针追上写指针，表示缓冲器已空，则不能从中读数据，若两端指针低位相同，仅高位不同，即写指针领先读指针一个周期，表示缓冲器已满，则不能向其中写数据。

✧ 同步控制

因为 RAM 两边时钟周期不同，系统时钟快于 SD 时钟，部分信号需要作同步处理，同步的搭建比较简单，只需用 2 个 D 触发器连接即可完成。用 FPGA 设计工具 ISE 综合后的 RTL 结构如图 4-16 所示。

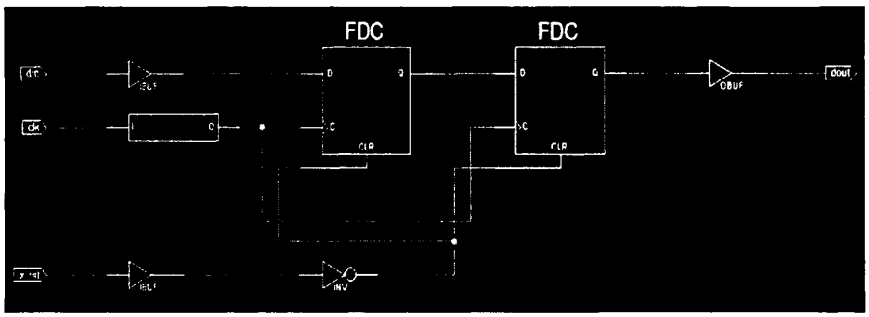


图 4-16 Buffer RAM RTL 结构图

✧ Buffer RAM 仿真图及结果分析

Buffer RAM 仿真图如图 4-17 所示。

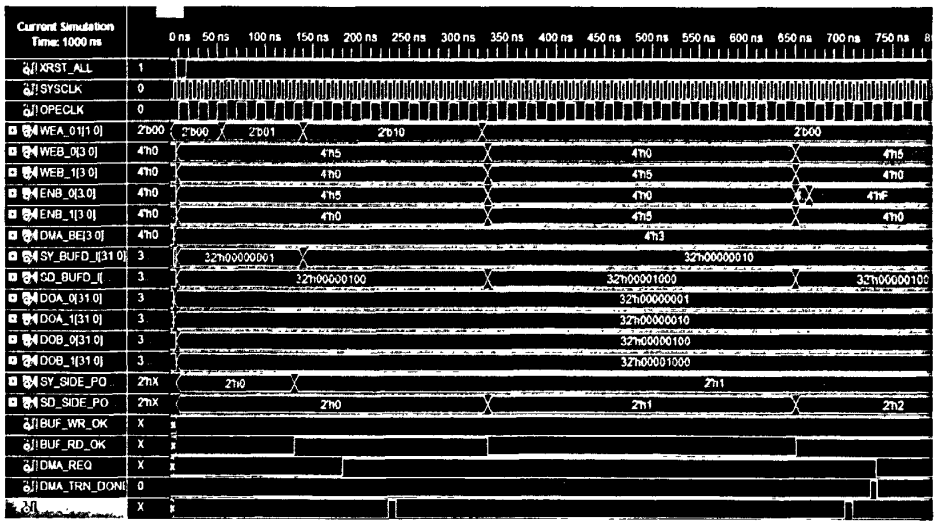


图 4-17 Buffer RAM 仿真图

本设计给出的激励为：SYSCLK 为 125MHz，OPECLK 为 50MHz。DMA 操作一直使能。字节使能 DMA_BE=4'b0011 即 2 字节有效，DOA_0=32'h00000001，DOA_1=32'h00000010，DOB_0=32'h00000100，DOB_1=32'h00001000 等。从图 4-17 可以看出，两片 RAM 的 A，B 端使能信号交替出现，ENB_0，ENB_1 两个信号分别当SD卡在进行读操作时(对RAM相当于写操作)，其值即为WEB_0，WEB_1，符合使能端控制的要求。当 RAM 为读操作时，通过指针正反值来区别使能哪个 RAM。所以在读操作时，当 RAM1 为 4'b1111 时，RAM0 必为 4'b0。SY_BUFD_I 的值分别由 DOA_0，DOA_1 交替给出。SD_BUFD_I 分别由 DOB_0，DOB_1 交替给出，数据传送正确。BUF_WR_OK 和 BUF_RD_OK 由两边指针来确定满空状态也符合设计要求，随着 DMA 完成信号有效，DMA 请求信号无效。符合设计要求，使 DMA 传输在 SD 卡中得以实现。

第 5 章　SD 控制器 FPGA 验证结果

前一章对几个重点模块进行了仿真、综合和实现，对于总体部分，本文采用自己设计的开发板进行总体验证。本章主要先介绍 FPGA 验证环境，通过下载比特流文件到开发板，按照验证场景进行验证，特别对 SDXC 卡的两种高速模式及速率进行了验证。

5.1 FPGA 验证环境介绍

本文因为考虑到有高频时钟的需求和开发板后期较高的重复利用率，选择了 Xilinx FPGA 芯片 VirtexⅣ160 系列。此板设计如图 5-1 所示。

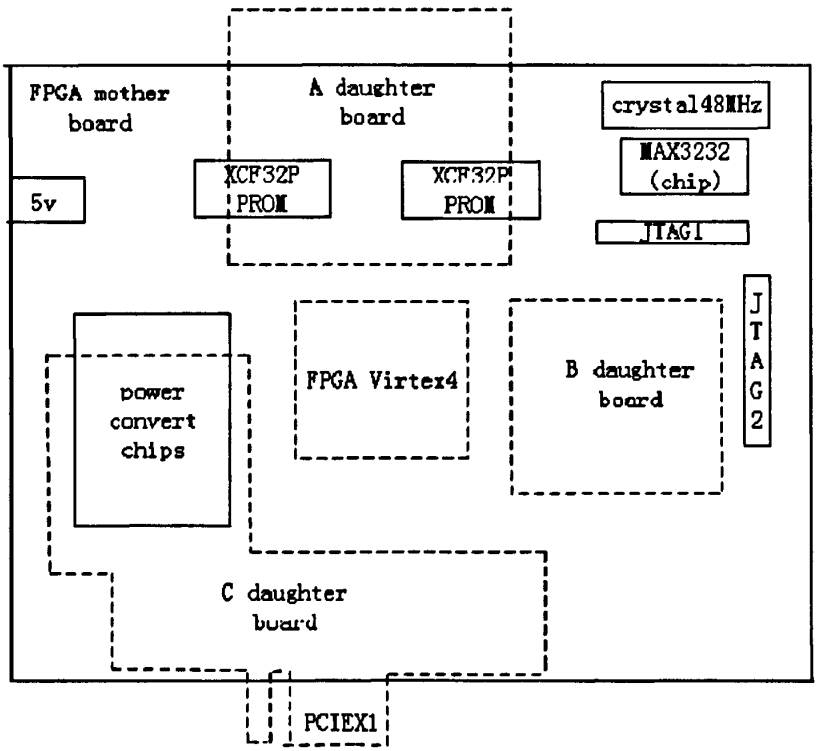


图 5-1 FPGA 验证版

.开发板由 FPGA 母板和几个子板构成，母板主芯片为 Xilinx VirtexⅣ系列及两片 PROM XCF32P 组成。每片 XCF32P 大小为 32M。图中有三个子板，A 板

主要用于普通 SD 卡，C 板主要用于 SDXC 卡，B 板主要为了检测内部信号以便调试测试所用。该开发板直接通过 C 板的 PCIEX1 插到主板的 PCI ExpressX1 插槽上实现 FPGA 与主机的通信。

5.1.1 电源

此开发板有两种供电模式，一种为外用 5V 直接供电，一种主板通过 PCI ExpressX1 插槽给板供 12V 电压及 3.3V 电压，再由母板的电压转换芯片，分别将 PCIE12V 电压转化成母板所需供电电压 5V，同时将 PCIE3.3V 电压转化成 FPGA 的核电压，FPGA 的核电压为 1V，2.5V，3.3V。各子板供电电压为 3.3V。因为 PCI Express 是一个多层协议，由传输层，数据链路层和物理层构成。为了与 FPGA 的数字特性匹配，在 C 板有一片特殊的芯片 TI PHY 用来连接并转换 PCIE 中物理层的信息。

5.1.2 配置

此开发板采用的配置方式为较常用的 JTAG 模式，JTAG 的全称是 Joint Test Action Group，即联合测试行动小组，目前，JTAG 已成为一种国际标准测试协议，主要用于各类芯片的内部测试，现在大多数高级器件（包括 FPGA、MCU、DSP 以及 CPU 等）都支持 JTAG 协议[40,41]。在 JTAG 模式中，PC 和 FPGA 通信的时钟为 JTAG 接口的 TCLK，数据直接从 TDI 进入 FPGA，完成相应功能的配置。

一般情况，在 FPGA 正常工作时，配置数据存储在 SRAM 中，这个 SRAM 单元被称为配置寄存器。由于 SRAM 是易失性存储器，因此重新上电后，外部电路需要将配置数据重新载入到芯片内的配置 RAM。因为考虑到掉电后每次需要重新配置数据比较繁琐，在开发板中另外设置了两片 32M 的 PROM 用来存放数据，PROM 不会因为 FPGA 的掉电而丢失数据，所以此开发板有两个 JTAG 的连接，一种将数据放置 SRAM，另一种将数据放到 PROM，然后下载到 FPGA 芯片中。

5.1.3 下载

当设计完成后，ISE 调用 BitGEN 程序把布局布线后的.ncd 文件转化成.bit 文件，包括了配置数据和配置指令。如果使用 JTAG 模式，可直接将.bit 文件或.mcs

文件通过 iMPACT 文件配置到 FPGA 芯片中。

通过原理图设计及 PCB 制板，硬件验证环境如图 5-2、图 5-3 所示。

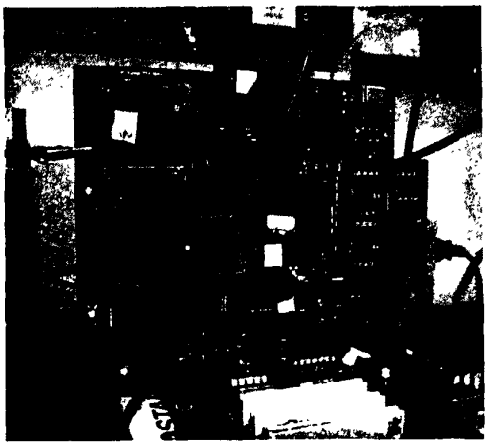


图 5-2 开发板

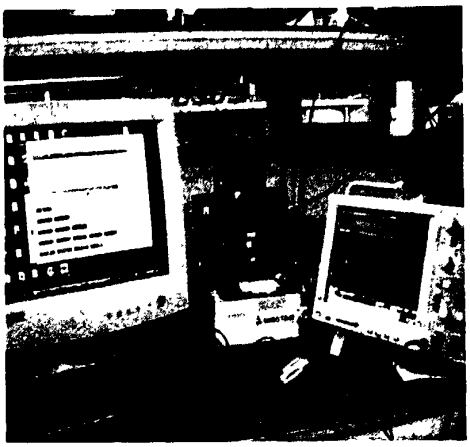


图 5-3 整体验证环境

5.2 验证场景

表 5-1 验证场景

	验证功能点
1. 普通 SD 卡（16MB~32GB）	基本功能读写操作 格式化 插拔卡 读写速度 写保护 安全退出
2. SDXC 卡 (64GB)	SDR25 模式读写 SDR50 模式读写 SDR104 模式读写 DDR50 模式读写 电压转换 不同模式下频率值
3.所有卡	插拔卡抖动时间 修改配置空间寄存器值，相应功能是否有效，如改变 M/N 系数，改变频率 时钟备用方案 插卡中断 上下拉控制信号是否正确当插拔卡和掉电

5.3 SDXC 卡的验证

采用 TOSHIBA SDXC64GB 卡，如图 5-4 所示。

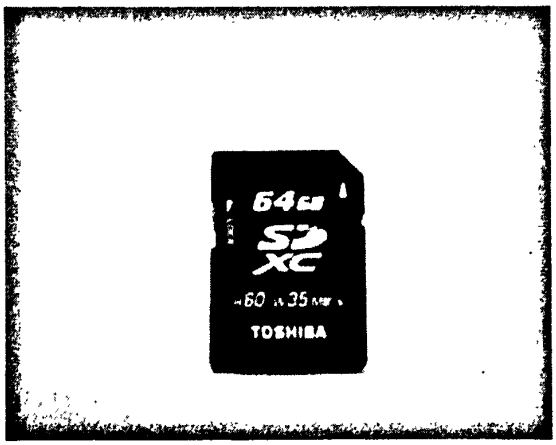


图 5-4 SDXC 卡实物图

特别地，对其在 SDR104 和 SDR50 模式下，验证了其时钟频率及读写速度，用示波器对其频率进行了测试及用 crystaldiskmark（测试速度读写软件）对读写速度进行了测试，测试结果满足设计要求。如图 5-5 至图 5-9 所示。

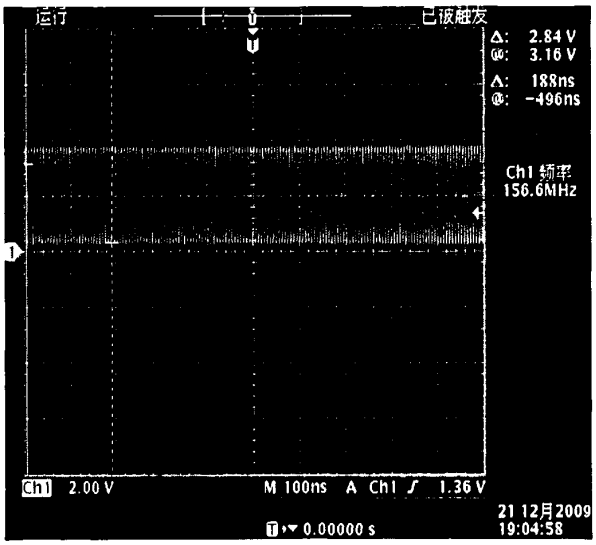


图 5-5 SDR104 模式

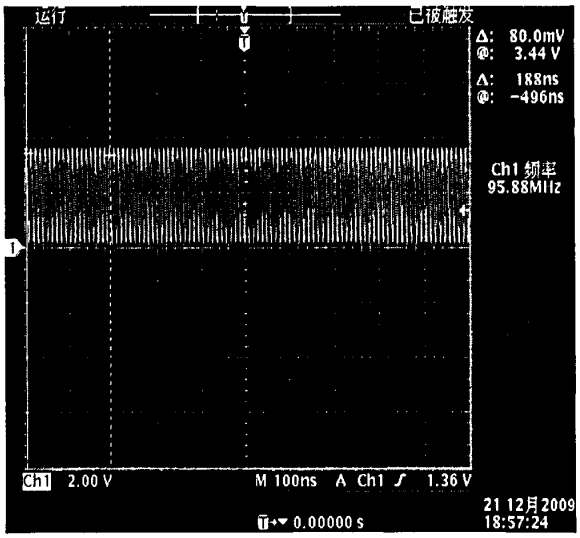


图 5-6 SDR50 模式

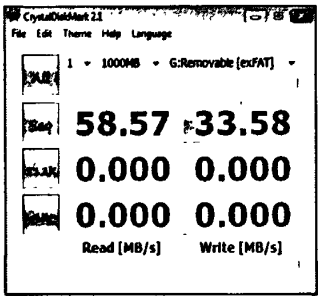


图 5-7 SDR104 模式

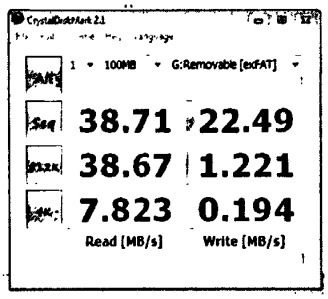


图 5-8 SDR50 模式

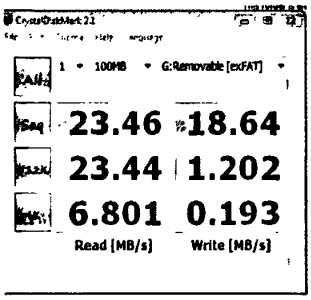


图 5-9 SDR25 模式

普通 SD 卡的时钟频率及速率如图 5-10、图 5-11 所示。

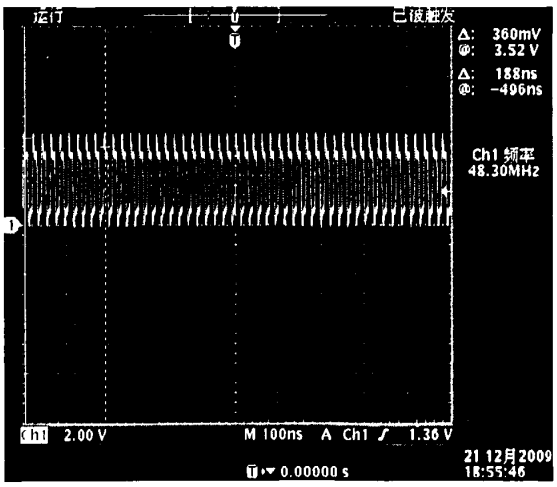


图 5-10 Sandisk 2GB 频率

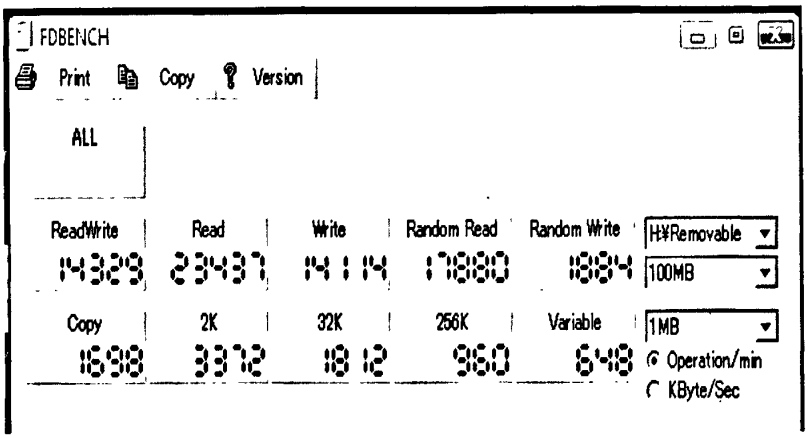


图 5-11 Sandisk 2GB 速率

由此可见，相对于普通 SD 卡（Sandisk 2GB）SDXC 卡在工作频率和传输速率上得到了很大提高。

5.4 速度比较

通过将普通 SD 卡读卡器与本文研究的 SD 卡控制器对 4 种工作频率为 50MHz 的 SD 卡进行速度比较，如表 5-2 所示。

表 5-2 速度比较

	USB 读卡器		SD 控制器	
	W(MB/S)	R(MB/S)	W(MB/S)	R(MB/S)
Sandisk 2GB	12.621	19.646	14.114	23.437
Sandisk extreme III 4GB	15.087	19.022	17.409	22.550
PNY 16GB	9.854	19.471	10.568	23.357
KINGSTON 32GB	10.270	19.530	11.124	23.272

从表 5-2 中可以看出，不同种类的普通 SD 卡在工作频率为 50MHz 时，本文研究的 SD 控制器在读写性能上都优于普通 USB 读卡器，提高了 SD 卡的读写速度。

第 6 章　总结及展望

SD 卡正在发展成为目前消费类电子产品市场上炙手可热的产品，如何使其速度越来越快，稳定性越来越好，对其进行控制显得尤为重要。笔者有幸参与了关于 SD 卡控制器实际项目的开发设计，完成了新标准下通用 SD 卡控制器设计。

本文首先介绍了 SD3.0 新规范，并阐述了新规范中的几个改进点。然后对整体 SD 控制器结构及各模块具体功能进行介绍。重点介绍了时钟模块，Tuning 模块及 DMA 模块的设计实现过程。最后介绍 FPGA 验证环境，对整体 SD 控制器进行验证，特别列出 SDXC 卡的验证结果及与普通 SD 卡在工作频率和性能上的比较结果。在整个研发过程中，笔者主要对以下内容进行了具体研究：

1. 时钟模块的设计，此模块主要充分利用 FPGA 内部资源 DCM(时钟管理模块)对时钟频率进行计算和设置。特别通过逻辑动态配置 DCM，利用 DCM 中的数字频率合成器的原理得出相关参数，灵活生成多种时钟频率供各类 SD 卡工作。
2. Tuning 模块的设计，此模块针对 UHS-I 模式中的 SDR50 和 SDR104 模式进行时钟相位调整，使 SD 卡在 160MHz 的高频下能够工作稳定。不仅拓宽了 SD 卡工作频率，同时提高了 SD 卡的工作速度。在硬件逻辑设计中，主要提出了最佳采样时钟算法对不同的相位时钟进行选择，通过状态机的具体设计和 FPGA 仿真及综合，完成此模块的设计任务。
3. DMA 模块的设计，此模块是 Buffer 控制模块中的一部分，在原来仅有驱动模式控制的基础上，提出用 DMA 进行控制，提高了 SD 卡的工作速度。在 DMA 控制中重点研究了双口 RAM 部分的设计，RAM 的控制，并通过 FPGA 仿真。
4. SD 控制器 FPGA 验证，此章节对 FPGA 开发板进行设计，并通过对几个场景进行验证，特别对 SDXC 卡的 2 种高速模式进行频率及速率测试，也与普通 SD 卡（Sandisk 2GB）在频率和速率上进行了对比，最后将 U 盘读卡器与本文设计的控制器作了速度对比，从整体上验证了本论文的 SD 卡控制器。

尽管实现了现有这个 SD 控制器，但并不是研究开发工作的终点，相反是更多工作和研究的起点，在进行了这些分析和研究之后，仍然是有很多方面值得进一步去讨论：

1. SD 卡控制器时钟方面可以设计停钟方案，在不需要 SD 卡工作的时候可以关闭 SD 卡时钟，这样可以降低功耗。例如对 SD 卡进行读写操作完成后，可以进入省电模式，关闭时钟，当对其再次进行读写操作后重新开启 SD 卡的时钟。
2. 对 FPGA 验证场景作更全面的设置，对控制器硬件作全面评估。在进行场景验证时，除了考虑常规的功能及性能测试外，应该多考虑一些突发情况。例如，在对 SD 卡进行读写操作时，突然拔掉 SD 卡，观察系统会不会瘫痪，SD 卡之后是否工作正常等。在对控制器进行硬件评估时，考虑功耗评估等。
3. 可进一步提高 SD 卡控制器的工作频率，目前 FPGA 采用芯片是 Xilinx Virtex4 系列，今后可采用 Xilinx Virtex5 系列，因为其内部逻辑资源更加丰富，可设计完成对时序要求更高的 SD 卡控制器。

致 谢

饮其流者思其源，学有成时念吾师，衷心感谢我的导师刘岚教授。本论文的研究工作是在刘老师的精心指导下完成的。刘老师治学严谨、踏实的工作作风以及高尚的人格将是我今后工作的榜样。在此，我向我的导师致以最衷心的感谢。

感谢所有教过我的老师。是你们辛勤的耕耘使我收获了扎实的专业基础知识，同时在专业技能上也得到提高。

感谢四凸电子有限公司项目组主管、带我的师傅和项目组其他成员，是你们让我把理论与实际联系起来，在实际工作中给予我很多帮助，使我个人能力得到提高，为以后工作打下良好的基础。

感谢同窗两年多的钟琴、邬巧燕、闫慧、李莎、吴芳等同学，感谢你们在学习和生活上给予我的帮助，与你们一起成长的日子将永远成为我生命中美好的回忆。

感谢尊敬的专家在百忙中抽出时间评阅我的论文。

最后，我要感谢我的父母，因为有你们默默的支持和鼓励，我才能坚强的克服各种困难，探索和解决新的问题。

谷 洵

2010 年 10 月 武汉理工鉴湖

参考文献

[1] 赵威. SD 存储卡的设计与实现[D].上海：上海交通大学微电子学院，2007.

[2] 左源，刘新宁，师超. 一种 SD 卡控制器的硬件实现[J]. 电子器件，2007,30（2）：126-128.

[3] 王清. SD 卡硬件启动和数据存储的控制逻辑的设计实现[J]. 电脑知识与技术，2008,4（4）：55-57.

[4] 胡菲，卢益民. 基于 FPGA 的 PCI 接口控制器的设计与实现[J]. 电子元器件应用，2006,1(10)：21-23.

[5] 李贵山，陈金鹏等. PCI 局部总线及其应用[M]. 西安：电子科技大学出版社，2006. 78-81.

[6] G. Borriello, R. Want. Embedded Computation meets the World Wide Web. Journalof the ACM, 2007, 43(5): 59-66.

[7] Samsung Electronics. S3C2410A–200MHz & 266MHz 32-Bit RISC MicroprocessorUser's Manual. 2004, 31-36.

[8] Brahim Doqan. Microcontroller and SD-card based multichannel data logger[J]. Electronics World,2008,11(2)：56-57.

[9] Bate I. Real-Time Embedded System. Computing & Control Engineering Journal,2007, 22(1): 72-76.

[10] 谢京华. 基于 FPGA IPcore 的接口设计与应用[D].成都：电子科技大学信息学院，2006.

[11] 张洁. 通用串行总线的SD卡读写器的设计.自然科学韶关学院学报，2008,29(3),125-127.

[12] 刘晓梅，董仲博. SD 卡 I/O 接口设计[J]. 现代计算机，2007,10(271),55-59.

[13] 李锡武. 掌上电脑 SD卡接口技术的研究与实现[D].武汉：华中科技大学计算机系，2006.

[14] X Ravi Badruk,Don Anderson,Tom Shanley. PCI Express system Architecture[J],Addison Wesley,2008,5(10):10-13.

[15] 董铁庄，吴晴. PCI Express(下一代内部互联技术)在嵌入式系统中的应用[J]. 微计算机信息，2005,6(03)：47-48.

[16] Physical Layer Specification Ver3[1].00 (SD GROUP: Panasonic, Toshiba,SanDisk),2009.

[17] SD Host Controller Standard Specification Version 2.00(SD GROUP:Panasonic, Toshiba,SanDisk),2005.

[18] SD Host Controller Standard Specification Version 3.00(SD GROUP:Panasonic, Toshiba,SanDisk),2009.

[19] 徐志军，徐光辉. CPLD/FPGA 的开发与应用[M]. 西安：电子工业出版社，2005.89-90.

[20] 周俊峰，黄建国，王志刚. 基于 PCI Express 接口的数据采集存储系统设计[J].设计参考，2009,11(9)：11-12.

[21] 黄辉，孙肖子. 基于 PCI_A MegaCore 的 PCI 总线接口设计[J]. 现代电子技，2006,4(147)：45-46.

[22] 4C Entity. Content Protection for Recordable Media Specification. SD Memory Card Book Common Part,2006.

[23] 张惠国，于宗光. FPGA 时钟分配网络设计技术[J]. 微计算机信息，2008,1(2)：187-190.

[24] Han-joon Kim etc. New Flash Memory Management for Flash Storage System,2005.

[25] 李丙玉，王晓东，吕宝林，刘文光. FPGA 设计中 DCM 的原理分析及应用研究[J]. 微计算机信息，2009,25(12)：22-23.

[26] 周盛雨，孙辉先，陈晓敏，安军社，张健. 基于模块化设计方法实现 FPGA 动态部分重构[J]. 微计算机信息，2008,2(2)：164-166.

[27] 邱云周. 基于 DSP 和 FPGA 运动控制技术的研究[D].四川大学信息工程学院系，2005.

[28] Wen-Tzeng Huang etc. A Compression Layer for Nand Type Flash Memory Systems,2007.

[29] Janick Bergeron. Writing Testbench functional verification of HDL models[M].Kluwer Academic Publisher,2005.

[30] Zeidman Bob. Introduction to Verilog.Piscataway. NJ:Institute of Electrical and Electronic Engineers,2005.

[31] 褚振勇. FPGA 设计与应用[M]. 西安：西安电子科技大学出版社，2006. 56-60.

[32] 黄智伟. FPGA 系统设计与实践[M]. 北京：电子工业出版社，2005. 45-48.

[33] 刘韬，楼兴华. FPGA 数字电子系统设计与开发导航[M]. 北京：人民邮电社，2005.78-81.

[34] 吴杰，张保平. 基于 FPGA 的 PCI 总线接口多通道 DMA 控制器的设计与实现[J]. 微型机与应用，2007,8(10)：15-17.

[35] 刘义峰. 基于 FPGA 的多通道 DMA 控制器的 IP 核设计[D].保定：华北电力大学信息学院，2008.

[36] 蔡菲娜，曹祁. 利用 FPGA 实现 DMA 方式的高速数据采集[J]. 微电子学与计算机，2005,3(22)：239-241.

[37] Santa Clara. Logic Design Manual for ASICs.CA:LSI Logic Corporation,2007.

[38] Michanel Keating. Pierre Bricaud Reuse Methodology for system on a chip design.Second Edition.Kluwer Academic Publisher,2005.

[39] [美]Bob Zeidman，赵宏图译. 基于 FPGA&CPLD 的数字 IC 设计方法[M].北京：北京航空航天大学出版社，2006.46-49.

[40] 师超. SDIO 接口的软硬件实现及性能评估[D].广州：东南大学电信系，2006.

[41] Xilinx. The ProgRAMmable Logic Data Book，2007.

攻读硕士学位期间发表的学术论文

发表的论文

[1] 谷洵，刘岚．SD 卡控制器中 DMA 传输的设计实现 科技论文在线，2010.7

基于FPGA的SD卡控制器的设计与实现

作者：谷洵
学位授予单位：武汉理工大学