中国科学技术大学

硕士学位论文

基于AMBA总线的SPI协议IP核的实现与验证

姓名：赵杰

申请学位级别：硕士

专业：微电子学与固体电子学

指导教师：易波

20090501

摘 要

在 SOC 设计日趋复杂化的今天，其发展的两大挑战是 IP 复用技术和 IP 互联技术，研究 IP 复用技术对于业界具有重要的现实意义。SPI 接口技术是一种高速、全双工、同步的通信总线，并且连线简单，有利于节省 PCB 空间，现在越来越多的芯片集成了这种通信协议。本项目设计了一种可配置为 Master 或 Slave 模式，可设置通信速率并能适用于不同传输模式的 SPI 协议 IP 核。这符合 SOC 设计技术发展的方向，具有重要意义。

综合分析对比了 Silicore 的 Wishbone、IBM 的 CoreConnect 和 ARM 的 AMBA 总线的技术特点。Wishbone 总线配置简单、灵活，有丰富的免费资源。CoreConnect 总线构造完整、通用，功能强大，但是对于嵌入式应用来说可能太复杂。AMBA 总线拥有众多第三方支持，已成为广泛支持的现有互连标准之一。特别是应用于低速系统连接的 APB 总线，技术简单，功耗低，实用性强，具有广阔的应用前景。

根据 Top-Down 的设计思路，论述了基于 AMBA 总线的 SPI IP 核模块的设计方法。首先分析设计目标，定义了模块外围的接口；进而根据设计功能划分子模块，进一步分析其内部互连信号，给出了完整的子模块和信号连接图，以及详细的端口连接和寄存器的设置。随后讨论了本 IP 核的 Verilog HDL 实现过程。首先阐述了整体的设计思路，进一步以时钟分频模块、APB 接口模块、收发逻辑控制模块等关键子模块为例介绍了 Verilog 代码设计过程中的几种典型问题与其解决过程。

使用业界通用的仿真软件 Modelsim 和 QuartusII 对本 IP 核的 Verilog HDL 代码设计进行了功能仿真和 FPGA 时序仿真。测试了模块重启、更换不同时钟分频频率、在所有四种传输时序下以及 Master 和 Slave 模式下的数据传输，结果表明模块的寄存器值和读写数据值均符合预期，仿真顺利通过。

关键词：SOC、IP 核、AMBA 总线、SPI 协议、Verilog HDL、FPGA

Abstract

Nowdays as the SOC design is getting more and more complicated, two major challenges to its development is the multiplexing and interconnect of IP core. SPI is a high-speed, full-duplex, synchronous communication bus, and its simple connection could save PCB space. More and more IC chips are realizinig this protocol. We designed an IP core of SPI protocol, which could be configured as SPI Master or SPI Slave, could set different transmission speed, and could work in any one of the four clock modes. This is very significant and is in line with the development of SOC design technology.

Proposed a comprehensive analysis comparing Silicore's Wishbone, IBM's CoreConnect and ARM's AMBA bus technical characteristics. The bus configuration of Wishbone is simple, flexible, and has much free resources. CoreConnect bus structure is integral, common and powerful, but it is too comlicated for embedded applications. AMBA bus has many third-parties support, and has become one of the standards of the existing interconnection bus. Especially, the APB bus which applies to low-speed connection has a simple structre, low-power, and practicability. It has a wide application prospects.

According to Top-Down design theory, I described the method to design the SPI IP core module based on AMBA bus. Analyzed the design objectives, and defined the input and output signal of the module. Then further divided its sub-modules and analyzed its internal signals, and present a complete module and signal connection diagram as well as detailed registers settings. Afterwards, described the realization of the IP-core using Verilog HDL. Described the design concept, and then introduced the design process of several important modules such as Clock Prescaler, APB interface and Transmit/Receive Logic, and then discussed typical problems in the design process and its solution.

Executed Function Simulation and FPGA Timing Simulation of the IP core using mainstream simulation software Modelsim and QuartusII. Tested the reset process, and checked the data transmission under different Clock speed and under all four clock modes. All the registers value and read/write data are in line with the data expected. Passed the simulation process successfully.

Keywords: SOC、IP core、AMBA bus、SPI protocol、Verilog HDL、FPGA

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：赵杰　　签字日期：2009.12.5

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定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，可以将学位论文编入《中国学位论文全文数据库》等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、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

☑公开　□保密（___年）

作者签名：赵杰　　导师签名：

签字日期：2009.12.5　　签字日期：2009.12.7

第1章 绪论

1.1 SOC 设计的发展现状

随着超大规模集成电路和深亚微米工艺制造技术的发展，集成电路芯片的规模越来越大，目前，数百万门级的电路可以集成在一个芯片上。与此同时，多种兼容工艺技术的开发，可以将差别很大的各种器件集成到同一个芯片上。这就是当前兴起的片上系统 SOC（System On Chip）。

片上系统 SOC（System On Chip）技术研究应用和发展正在微电子及其应用领域引发一场前所未有的变革[1]。SOC 将电路系统设计的可靠性、低功耗等都考虑在 IC 设计之中[2]，把过去许多需要系统设计解决的问题集中在 IC 设计中解决，使系统工程师可以最大限度地简化电路设计，达到整体产品系统的可靠性、精度、稳定等指标。因此，SOC 技术是超大规模集成电路发展的必然趋势和主流。

1.1.1 国内外 SOC 技术的研究及应用现状

随着 SOC 应用的不断普及 市场需要更加广泛的 SOC 设计[1][2][3]。提供商必须拓展系统内部设计能力直接开发和交付设计套件和方法给客户。因此 SOC 设计逐渐向可编程 SOC 方向发展。

中国在高新技术研究发展 863 计划中把 SOC 作为微电子重大专项列入了 2000/2001 年度信息技术领域的重大专项预启动项目，并在 IP 核的开发软硬件协同设计、IP 复用、VDSM 设计、新工艺新器件等方面布置了预研性课题。其中 IP 核的设计和制造是 SOC 技术中最为关键的部分。在中国最适应 SOC 技术应用开发的 SOC 类型是可编程 SOC 技术。

可编程 SOC 是在一块现场可编程芯片上提供产品所需的系统级集成，提供了 ASIC 的高集成度、低功率、小尺寸、低成本，以及 FPGA 的低风险、灵活性和上市快的特性，这也是 SOC 技术在微电子行业受欢迎的最根本的原因。目前，

多家 IC 提供商已经在可编程 SOC 的实现方面提供了很多新的器件，所提供的系统功能包括处理器、存储器和可编程逻辑等，克服了与 ASIC 相关的工程费用高或制造周期太长的问题。

1.1.2 SOC 设计技术的挑战

SOC 设计虽然在过去的十几年中已经取得了长足的发展，但是它所面临的挑战也是不容忽视的[1][4]。SOC 设计的主要挑战，一是系统级设计，这主要关系到 IP 核间的互连性；二是设计可重用的 IP 核，这是 SOC 设计面临的又一个挑战。

IP 核的互连使用片上总线结构解决。典型的 IP 核互连结构见图 1.1。片上总线可以提供 IP 核间的互连，但这又导致了另外一个问题：各 IP 公司使用的总线结构各异，因此不同公司 IP 核之间的互连非常困难。针对此情况，VSIA（Virtual Socket Association）组织一直致力于建立一种通用的片上总线结构。下一节将继续介绍。

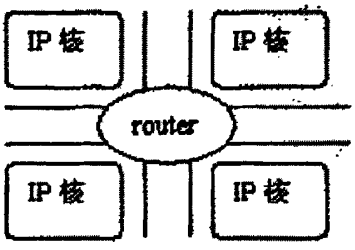

图 1.1 典型的片上网络

此外，SOC 的设计挑战还包括降低 SOC 功率损耗、SOC 的功率预测和最优化、信号完整性、混合信号设计等。上述的这些挑战是长期存在的，影响着当前和未来 SOC 的设计和制造能力。

1.2 IP 核复用技术的产业现状[2][4]

IP 内核复用技术可以有效地缩短 IC 产品设计开发周期并降低成本。然而目前我国整个 IP 产业尚不成熟，缺乏行业规范和交流渠道限制了 IP 产业的发展[5]。

在国外，中小规模的 IP 设计公司、IP 销售公司、对 IP 进行评定和为 IP 进行诉讼调停的公司日益增多，IP 产业发展势头迅猛。我国也十分重视 IP 产业的

发展。IP 产业正在从概念阶段向实用阶段过渡。但总的讲来，现在国内整个 IP 产业尚不成熟，行业规范和交流渠道的缺乏限制了 IP 产业的发展。此外，文档编制的规范性、IP 验证和 IP 评估等问题也有待解决。

1.2.1 IP 内核产业的三类主体

对 IP 内核产业感兴趣的人可分为三类：通用 IP 供应商、通用 IP 用户和专属的 IP 设计人员和用户。

通用 IP 供应商是只对内核开发感兴趣并以此作为最终产品销售或附属产品，这些公司并不关心如何将内核与整体产品设计进行接口。

通用 IP 用户是开展大型设计项目的公司，他们希望通过利用来自外部的内核加速产品的上市时间。IP 用户致力于开发针对特定应用的产品，而对自己开发内核没有兴趣，因为自己重复利用这些内核的机会很小。

最后，专属 IP 设计人员和用户服务于那些针对特定市场进行大规模内部专用产品设计的公司。这些公司的设计主要针对某个特定市场，因此他们的设计复用的机会很多。虽然这些公司设计可复用的内部专用内核，他们也购买一些外部开发的内核，作为内部开发产品的有益补充。

1.2.2 设计复用相关的组织[5]

由于 IP 复用已成为芯片设计的一项重要内容，因此业界成立了不同的组织以推动设计复用标准的发展。他们的目标是开发一套业界标准，促进 IP 使用并简化外部 IP 与内部设计之间的接口。下面将介绍那些致力于标准开发的组织及其作用。

1.VSIA 协会

1996 年 9 月虚拟接口联盟(VSIA)成立，该联盟的成立是为了推动多个来源 IP 内核之间的“混合搭配”而制订开放标准，从而加速 SOC 开发。该联盟的会员由业界各系统公司、半导体公司、IP 公司和 EDA 公司组成。

VSIA 希望通过发布开放的接口标准创建一个环境，这样 VC 就能以最少(甚至不需要)的胶合逻辑电路轻松地满足“虚拟接口”需求，而且是基于功能和物理层面的。VSIA 标准包括业界已有的标准、公开或专有的数据格式，目标是创建可交付使用的内核标准格式，这样内核就完全独立于各个用户的独特设计流程。

2.OpenMORE

Synopsys 公司和 Mentor Graphics 公司合作开展了著名的 OpenMORE(Open Measure of Reuse Excellence)计划，这是建立在两家公司共同发起的“复用方法指南”(RMM)基础上的一项评估计划。OpenMORE 将 IP 内核设计定义为可视为完整 SOC 设计一部分的独立设计。此外，RMM 还定义软核为软宏(softmacro)或以集成 RTL 代码形式交付的内核；而硬核则定义为硬宏或以 GDSII 文件形式交付的内核。硬核可以是完整的设计、布局和布线。

当设计人员决定为他们的设计购买 IP 内核时，IP 评估将成为设计流程的重要环节。OpenMORE 方案有望通过为内核复用质量提供合理的评估模式而简化 IP 评估流程。IP 开发人员在一个工作表中填写有关硬核和软核的规则描述和应用指南，用户利用该过程得到的最后分数来评估内核设计方法。

大部分 OpenMORE 用户是那些通常在使用公司内部开发的内核和第三方 IP 的公司。IP 提供商也可采用 OpenMORE 以使用户更方便地使用内核，从而减少客户支持。

1.3 课题研究的意义

在 SOC 设计越来越趋向复杂化的今天，研究 IP 复用技术对于业界具有重要的现实意义。

SPI 总线是 Motorola 公司推出的一种同步串行接口技术。它允许 MCU 与各种外围设备以串行方式进行通信和数据交换。它是一种同步的高速、全双工的通信总线，并且只需要四根线，节约了芯片的管脚，同时为 PCB 的布局上节省空间。因为其简单易用的特性，现在越来越多的芯片集成了这种通信协议。因为 SPI 串行通讯总线具有结构简单、通讯速度快等显著优点，在嵌入式系统中应用日趋广泛，例如基于 SPI 总线的数据采集系统[6]，基于 SPI 接口的 DSP 的 Flash 扩展方案[7]，以及利用 SPI 总线连接 DSP 和单片机[8]等。

实现 SPI 协议的 IP 核已经成为业界的设计热点之一，但现有设计功能不够完善。文献[9] 基于龙珠处理器 MC68Ez2328 的并行总线接口，设计了多路 SPI Master 模块，没有实现 SPI Slave 功能；文献[10]基于 Wishbone 总线接口，同

样只设计了 SPI Master 模块。文献[11]介绍一种基于 AMBA 总线的 SPI 协议 IP 核设计，但其重点说明 IP 核设计的规范流程和方法，对于 SPI 模块本身的特性叙述不多。SPI 协议规定通讯需要在 SPI Master 和 SPI Slave 之间进行，并约定了四种传输模式。为了给 SOC 设计和 FPGA 开发提供最大的灵活性，有必要设计一种可配置为 Master 或 Slave 模式，可设置通信速率并能适用于不同传输模式的 SPI 模块。我们设计的基于 AMBA 总线的 SPI 协议 IP 核具备上述优点。选择 AMBA 总线作为接口的原因是它方便灵活，且完全免费，在基于 ARM，SPARC 等处理器系统级芯片设计中已经成为 SOC 的总线标准[12]。

1.4 本文的章节安排和主要内容

本文的主要内容是使用 Verilog HDL 语言，设计和实现了基于 AMBA 总线接口的 SPI IP 核。进行了详细的模块划分和寄存器设置，并使用 EDA 软件对其进行了仿真验证和设计综合。各章的主要内容如下：

第一章概述了国内外 SOC 设计产业和 IP 产业的发展状况，并由此引出了项目研究的意义；

第二章概括介绍了本项目研究所涉及的技术背景：SOC 设计技术、IP 核和 IP 复用技术、以及 IP 软核的设计流程。

第三章分析对比了几种片上总线，并详细介绍了 AMBA 总线和 SPI 协议。为后文的设计内容做好了准备。

第四章叙述了基于 AMBA 总线的 SPI IP 核的设计过程，论述了详细的模块划分和寄存器设置，以及各个模块的 Verilog HDL 实现的思路和重点细节。

第五章分析了 IP 核的时序仿真结果，验证了设计的正确性。

第2章 技术背景

2.1 SOC 的设计方法

2.1.1 SOC 的定义

和传统的功能单一的单元电路 IC 不同，SOC 将信号采集、处理和输出等功能完整的集成在一起，成为一个专用的电子系统。一般认为 SOC 片上系统的定义是:在单一芯片上集成了控制部件(微处理器、存储器)和执行部件(如 I/O,微型开关等)，能够自成体系、独立工作的芯片。

2.1.2 SOC 的设计流程

在 SOC 中，尤其是在面向特定应用领域的 SOC 中，软硬件的结合是非常紧密的，软件和硬件之间的功能划分和实现并没有固定的模式，随不同的应用变化。因此一般来说，面向 SOC 的设计过程是软硬件协同的。图 2.1 给出了一个较为普遍的面向 SOC 软硬件协同设计流程。

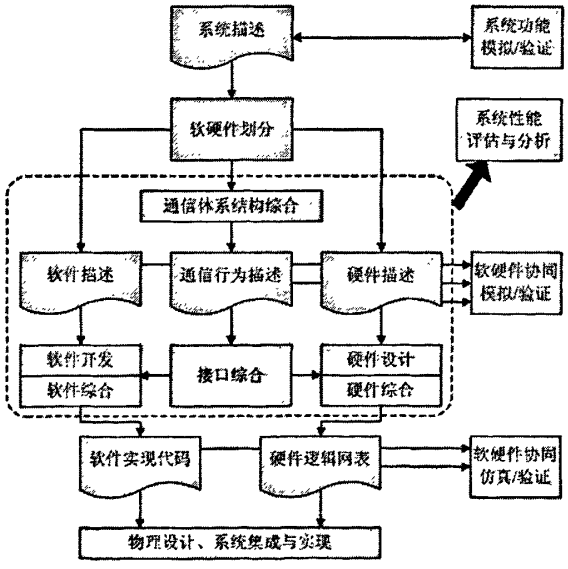

图2.1 SOC软硬件协同设计流程[13]

2.2 IP核和IP复用技术

IP核的含义是指具有知识产权（Intellectual Property）的集成电路芯核，其集合了一组拥有知识产权的电路模块，在设计时可用作为单元模块使用。

2.2.1 IP内核的三种类型

IP内核主要分为三种类型：软核、固核和硬核[14]。这种分类主要是依据不同的产品交付方式和实现方法。

软核的设计完成度最低，通常以可综合的HDL形式提供，具有较高的灵活性，并且与具体的实现工艺无关。但难以预见时序、面积和功耗是其主要缺点。软核以源代码的形式提供，因此其面对着严峻的知识产权保护问题。

硬核的设计完成度最高，以完成布局布线后的网表形式提供，可预见其时序、面积和功耗。尽管硬核可移植性差，但其无须提供寄存器传输级(RTL)文件，易于实现IP保护。

固核是介于软核和硬核之间的设计形态。在完成软核设计的基础上，还进行了门级电路综合和时序仿真等环节。一般固核以门级电路网表的形式提供。

2.2.2 IP 复用技术

IP复用技术是SOC设计的最大挑战之一。当代电子产品的生命期正在不断缩短，这就要求芯片的设计具有更快的速度。为了加快芯片设计过程，人们将已有的IC电路以模块的形式封装起来，在芯片设计中调用，从而达到简化设计流程、缩短设计时间、提高设计效率的目的。

2.3 IP 软核的设计流程

2.3.1 IP 软核的设计过程

基于 SOC设计思想，IP软核设计大多仍采用经典的自顶向下(TOP-DOWN)的设计流程。其设计流程开始于规范制定、功能划分，结束于IP软核的验证和封装交付。下面简单列出主要步骤。

1.分析设计目标，为IP核制定全面的设计规范(Specification);

2.将设计目标分解，将IP核分为各个子模块，各模块间的接口定义以及相互关系都由设计规范给出;

3. 用RTL级代码对IP核的各个子模块进行硬件建模实现，最后将各个子模块集成，开发侧试平台(test bench)，并由EDA(Electronic Design Automation) 工具进行功能验证;

4.在某一FPGA平台下对顶层RTL级代码进行综合，得到与工艺相关的网表文件;

5.使用EDA软件完成布局布线和时序验证，最后在该FPGA平台下进行IP软核的上电验证;

6. 封装IP软核，可交付给用户进行设计复用。(参见图 2.2)

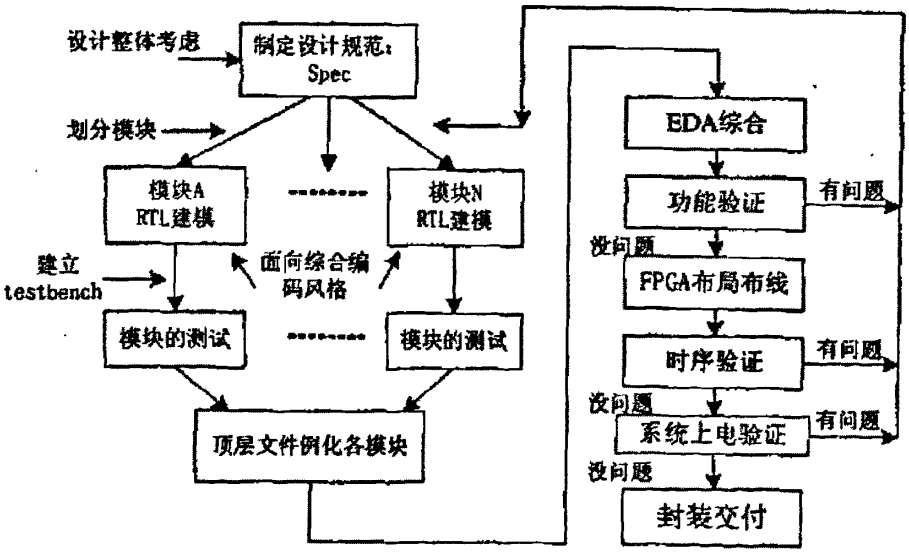

图2.2 自顶向下的软核设计过程[15]

自顶向下的设计过程是一种理想情况，它假定底层各模块均已可完成设计。如果某一模块存在问题，或者影响到整体时序，那么设计过程就需要反复。因此常混合使用自顶向下和自底向上的设计方法来实现IP软核的设计。自底向上的方法是在构建关键子模块时，将各个子模块设计实现时所暴露的具体问题反馈给整体设计，再对整体设计进行修改[16]。

无论IP软核设计采用何种设计过程，最重要的是对IP软核的设计进行总体思考，在设计之前制定全面的设计规范。

2.3.2 IP 核的设计原则

IP 核设计时应尽可能地提供可配置性，对RTL 代码做尽可能全面的仿真，建立标准、规范的文档。为了使IP 核能够适应不同规范的IP 核互连标准，通常的设计中将IP 核分为IP 功能内核和IP 核接口两部分分别进行设计。通过给IP 功能内核配符合不同规范的接口模块，可以在不同场合的应用中保持高度的可互连性和高速的数据交换。

2.4 本章小结

本章主要介绍了设计IP核所用到的背景知识：SOC 的设计流程、IP核和IP复用的概念，以及 IP 软核的设计流程和原则，为下文的设计作好准备。

第 3 章 片上总线及相关协议分析

3.1 片上总线综述

片上系统 SOC 设计的关键问题之一就是采用片上总线。采取适当的片上总线可以使核移植和复用变得容易设计，并且能充分利用外设和处理器，改进自动验证，提高从公共设计平台创建产品的定制化的能力。设计片上总线的时候应该注意以下三点：

①尽可能简单。首先是采用简单的结构，可以节约逻辑单元；其次采用简单的时序，能提高总线的速度；第三采用简单接口，可减少与 I P 核连接的复杂度。

②有较大的灵活性。片上系统应用广泛，对总线的速度、位宽、时序等等参数，不同种类的应用要求各不相同，因此需要片上总线具有较大的灵活性。

③尽可能降低功耗。在实际应用时，总线上各种信号应尽量保持不变，并且可以考虑在闲置时置于低电平；此外尽量采用单向信号线，这样不但降低了功耗，同时也简化了时序。

有两种方法可以实现片上总线，一是选用国际上公开通用的总线结构；二是根据特定领域自主开发片上总线。

下面分析比较一下目前 SOC 上使用较多的三种片上总线标准——Silicore 的 Wishbone、IBM 的 CoreConnect 和 ARM 的 AMBA。

3.2 三种常用片上总线介绍

3.2.1 Wishbone 总线

Silicore 公司最先提出 Wishbone 总线，现在由 OpenCores 组织负责维护。由于其开放性，结构简单、灵活，现在已经拥有不少用户群。Wishbone 总线规范是一种片上系统 IP 核互连体系结构。它定义了一种 IP 核之间公共的逻辑接口，减轻系统组件集成的难度，能够提高系统组件的可重用性、可靠性和可移植性。Wishbone 总线规范可用于软核、固核和硬核，可以用多种开发工具和目标硬件实现。

结构简单是 Wishbone 总线的一个优点。它仅定义了一条高速总线。在一个复杂的系统中，可以采用两条 Wishbone 总线的多级总线结构：其一用于高性能系统部分，其二用于低速外设部分，使用一个接口连接二者。Wishbone 的一种互连结构如图 3.1 所示。

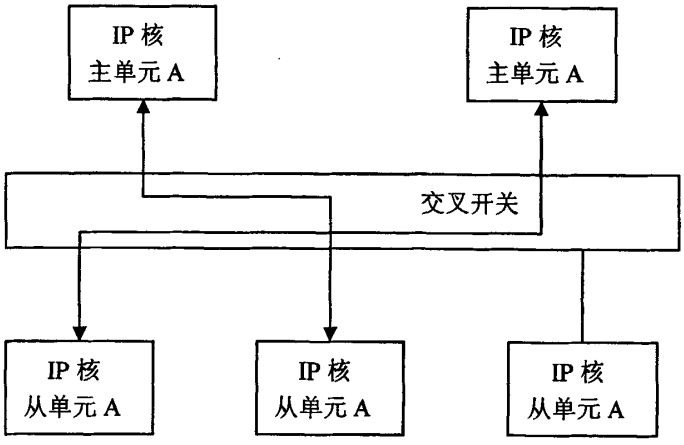

图 3.1　交叉开关总线方式

灵活性是 Wishbone 总线的另一个优点。并没有一种统一的间接方式来连接多种多样的 IP 核。Wishbone 总线提供了四种不同的 IP 核互连方式，来满足不同系统的需要：

◇用于两 IP 核直接互连的点到点（point-to-point）方式；

◇用于多个串行 IP 核之间的数据并发传输的数据流（data flow）方式；

◇用于多个 IP 核共享的共享总线（shared bus）方式；

◇用于同时连接多个主从部件，提高系统吞吐量的交叉开关（crossbar switch）（图 3.1）方式。

3.2.2 CoreConnect 总线

IBM 公司的 CoreConnect 总线在技术上可行性较强，它拥有完备的一整套技术文档。CoreConnect 具有三种基本类型的连接总线，即处理器内部总线 PLB(Processor Local Bus)、片上外围总线 OPB(On-Chip Peripheral Bus)和设备控制总线 DCR(Device Control Register)[19]。CoreConnect 总线的逻辑结构如图 3.2 所示，它定义了所有的系统构成部件以及它们是如何连接的。下面分别介绍 PLB 和 OPB 的主要技术特征。

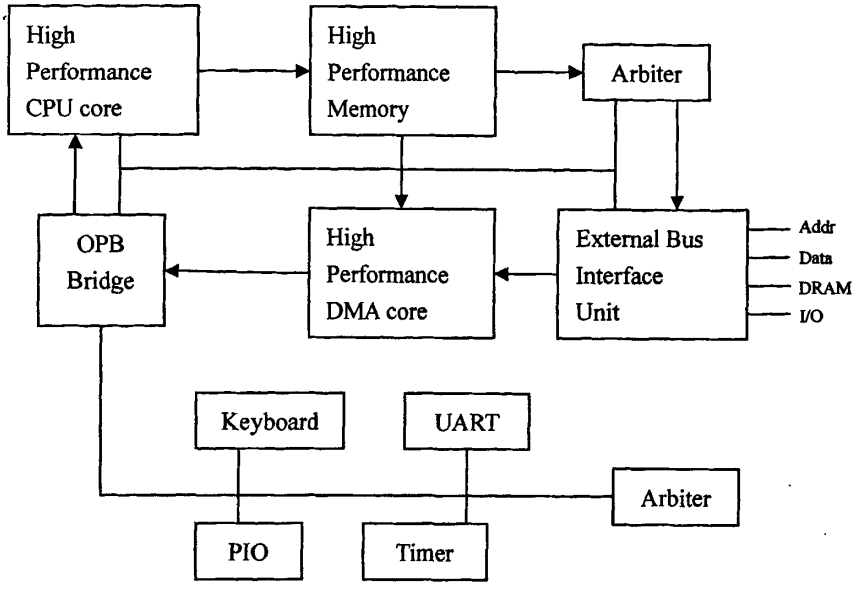

图 3.2 Core Connect 总线逻辑

PLB 标准的目的是为总线传输的主要发出者和接受者之间提供高带宽、低延迟的连接。主要特性如下:

◆高性能处理器内部总线;

◆交叠的读和写功能(最快每周期两次传输);

◆可以分段传输;

◆ 32 或 64 位数据总线:

◆ 32 位地址空间;

◆支持字节使能(非对准和 3 字节传输);

◆支持仲裁、REQ,GNT 和 LOCK;

◆特 殊 DMA(DirectM emoryA ccess)模式，如快速的从内存到内存;

◆地 址和数据状态扼制(Addressan dd ataphaseth rottling);

◆延迟计数器(保证保持延迟到预想的程度)。

OPB 标 准 的目的是连接具有不同的总线宽度及时序要求的外设和内存，并做到尽量保持 PLB 的性能。主要特征为:

◆片上外围总线;

◆多个主设备;

◆ 32 位地址空间;

◆读和写数据总线分开;

◆ 8^-32 位数据总线;

◆支持重试模式(如果主设备要求的从设备忙，主设备隔一段时间再次请求):

◆支持突发(burst)传输模式;

◆支持 DMA;

◆设备可以是内存映射(支持 DMA);

◆检测总线超时功能(在仲裁器中);

CoreConnect 是一套精心设计、构造完整、通用的解决方案，可以为类似于工作站这样的高性能系统提供性能良好的连接，但是对于简单的嵌入式应用来说可能有点过于复杂，许多特性无法用到，但其优点是可适用未来更庞大、更复杂系统的连接。

3.2.3 AMBA 总线

AMBA（Advanced Microcontroller Bus Architecture）总线规范是 ARM 公司设计开发的一种用于高性能嵌入式系统的总线标准[20]。它的优点是独立于处理器和制造工艺技术，增强了各种应用中的外设和系统宏单元的可重用性。并且 AMBA 总线规范是一个开放标准，可免费从 ARM 获得。

目前，AMBA 总线被 ARM 公司 90%以上的合作伙伴所采用，在基于 ARM 处理器内核的 SOC 设计中，已经成为拥有广泛支持的现有互联标准之一。AMBA 总线规范 2.0[21]于 1999 年发布，该规范引入了先进的高性能总线（AHB）。为了实现在任何工艺条件下均能实现接口和互连的最大带宽的目标，AHB 对接口和互连均进行了定义。AHB 已经不仅仅是一种总线，而是一种带有接口模块的互连体系。

AMBA 总线规范主要设计目的如下：

① 满足具有一个或多个 CPU 或 DSP 的嵌入式系统产品的快速开发要求；

② 确保可重用的多种 IP 核可以成功地移植到不同的系统中，适合全定制、标准单元和门阵列等技术，增加设计技术上的独立性；

③ 为了增加处理器的独立性，促进系统模块化设计；

④ 为了使片外的操作和测试通信更加有效，减少对底层硅的需求。

AMBA 总线是一个多总线系统，其定义了三种不同类型的可以组合使用的总线：AHB（Advanced Highperformance Bus）、ASB（Advanced System Bus）和 APB（Advanced Peripheral Bus）。图 3.3 示意了典型的基于 AMBA 的 SOC 核心部分。其中， CPU 和存储器、DMA 控制等高速模块之间通过高性能系统总线（AHB 或 ASB）连接。，而系统的大部分低速外部设备则连接在低带宽总线 APB 上。可以用一个桥接器（AHB/ASB-APB-Bridge）来连接 AHB 和 APB。

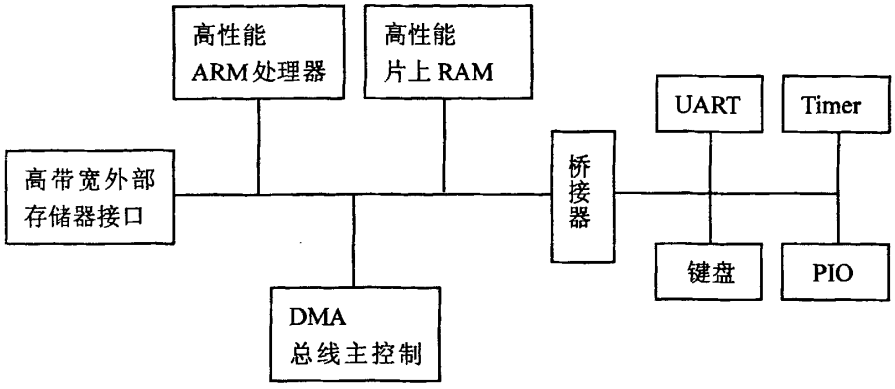

图 3.3 基于 AMBA 的主控制器[]

AMBA 的 AHB 是高性能系统的骨干总线，主要用于连接高性能和频率设备之间的连接，如 CPU、DMA 设备、片上存储器和 DSP 等。其主要特性如下：

- 支持多个总线主设备控制器；
- 单周期总线主设备控制权转换；
- 32~128 位数据总线宽度；
- 具有访问保护机制，以区分特权模式和非特权模式访问， 指令和数据读取等；
- 数据猝发传输最大为 16 段；
- 地址空间为 32 位；
- 支持字节、半字和字传输。

AMBA 的 ASB 同样适用于高性能的系统模块。在可以不选择 AHB 的高速特性的场合，可使用 ASB 作为系统总线。它支持处理器、片上存储器，以及片外处理器接口与低功耗外部宏单元之间的连接。其主要特点与 AHB 类似，而主要的不同是它采用同一条双向数据总线来读数据和写数据。

AMBA 的 APB 拥有完善的低功耗特性，其主要作用是连接系统的低速外部设备。它已经过优化，可以减少功耗和对外设接口的复杂度；它可连接在 AHB 和 ASB 两种系统总线上。其主要特征为：

- 低速、低功耗外部总线；

- 单个总线主设备控制器；
- 非常简单，加上 CLOCK 和 RESET，总共只有 4 个控制信号；
- 32 位地址空间；
- 最大 3 2 位数据总线；
- 读数据总线与写数据总线分开。

APB 总线上的信号主要包括：时钟信号 PCLK；地址线 PADDR；写入使能 PWRITE 和 PENABLE；片选信号 PSEL；写入数据信号 PWDATA 和读取数据信号 PRDATA。时钟的上升沿是所有传输信号的触发沿，主要包括写信号和读信号的传输，如图 3.4 所示。图中的读写数据的时序，第一个时钟周期为 SETUP 周期，第二个周期为 ENABLE 周期。信号的传输在 ENABLE 周期中完成。

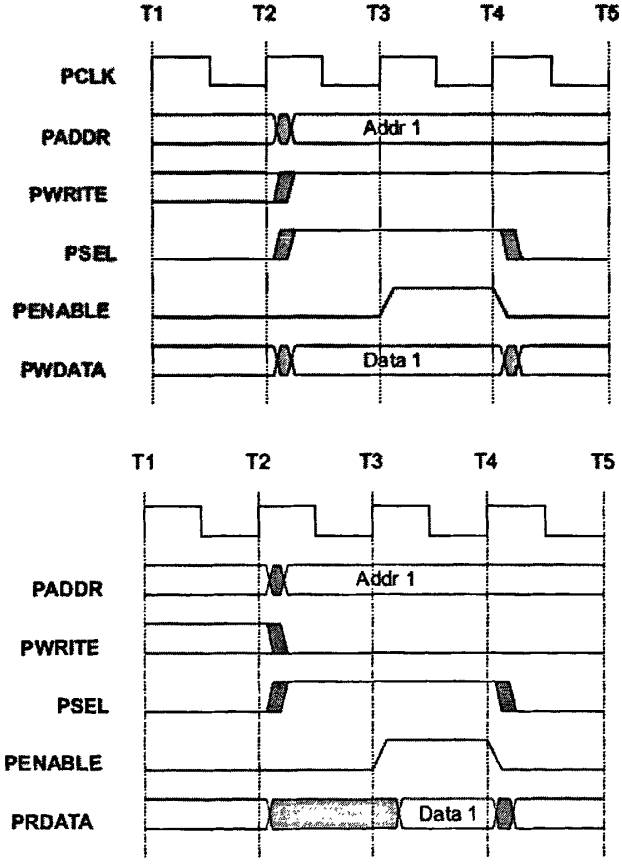

图 3.4 APB 总线写入和读取数据时序图[28]

上文介绍的三种总线中，Wishbone 总线配置简单、灵活，有丰富的免费口资源。CoreConnect 总线构造完整、通用，功能强大，但是对于嵌入式应用来说可能太复杂。AMBA 总线拥有众多第三方支持，已成为广泛支持的现有互连标

准之一。特别是应用于低速系统连接的 APB 总线，技术简单，功耗低，实用性强，具有广阔的应用前景。因此在本项目的设计中，将 AMBA 中的 APB 总线作为了所设计 SPI 核对外连接的接口。

3.3 SPI 协议

SPI(Serial Peripheral Interface— — 串行外设接口)总线是Motorola公司推出的一种同步串行接口技术。它允许 MCU 与各种外围设备之间以串行方式进行通信和数据交换。它是一种同步的高速、全双工的数据通信总线。它只使用四根信号线，节约了芯片的管脚，为 PCB 的布局上节省空间。因为 SPI 总线具有简单易用的特性，现在越来越多的芯片内部集成了这种通信协议。

3.3.1 SPI 信号线

SPI 是一种同步传输接口，具有同步时钟信号、数据输入信号和数据输出信号。另外每个从芯片还需要一个片选信号，主器件通过片选信号选择与其通信的从器件。SPI 总线由 1 根片选线、1 个时钟信号线和两条数据线共四根线所组成，如表 3.1 所示。

信号名	描述
SCK（serial clock）	串行时钟线，由 SPI 主模块产生，在主从交换数据时使用，确保数据交换的同步性
MOSI（master out slave in）	主机输出/从机输入线，串行数据传输
MISO（master in slave out）	主机输入/从机输出线，串行数据传输
SS（slave select）	从机选择线，由主机发送至从机，当输入时表示该从机被选中，与主机进行通信，否则未选中，与主机进行通信

表 3.1 SPI 外围端口表

3.3.2 SPI 的主从模式

SPI 有主模式和从模式两种工作模式[23]。工作在主模式的是主器件，工作在从模式的是从器件。主器件负责对数据传输进行初始化。同步时钟信号 SCK 由主器件产生，每一位数据的发送 / 接收需要 1 个时钟周期。主器件通过对数据寄存器的写操作可以开始数据传输。如果移位寄存器为空，待发送的数据会立即写入移位寄存器，然后 8 位发送数据在串行时钟的控制下通过 MOSI 引脚依次送出，同时从机发送过来的数据通过 MISO 引脚依次进入主机。

从器件片选信号 SS 由主器件输出，在低电平有效，此时主器件和从器件开始通信；当 SS 信号处在高电平时，说明片选信号无效，系统空闲。如果在数据传输的中途 SS 信号由低变高，系统会中止数据传输。

3.3.3 SPI 系统构成

一个典型的 SPI 系统包括 1 个主器件和 1 个或几个从器件，在本文设计采用的是片上总线 APB 接口和上层的 CPU 通信，如图 3.5 所示。

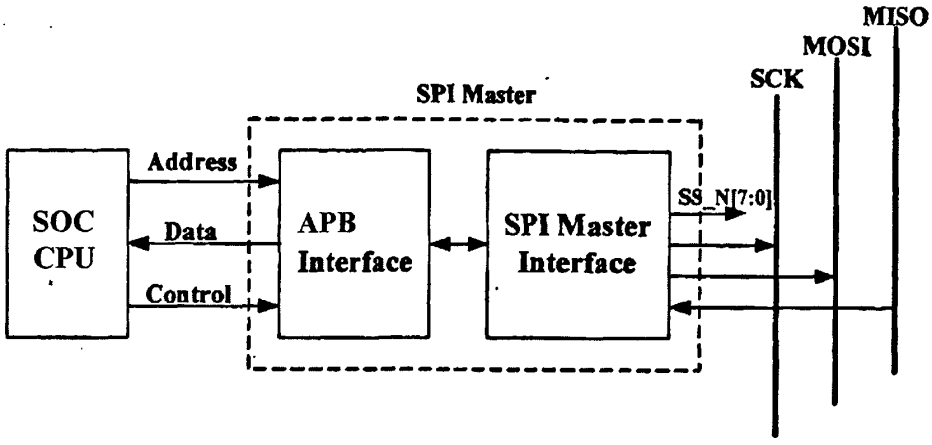

图 3.5 典型 SPI 系统(SOC)结构

SOC CPU 通过 SPI 接口模块与外围器件相连。当该 SOC 以主机模式运行时，可以通过片选信号选择一个从机器件进行通信，而当它以从机模式工作时，等待被主机选中进行通信。系统内同一时刻只能有一个主机，否则系统就会进入异常状态。本文设计的 SPI IP 核模块可以通过寄存器把模块配置为主机模式或者从机模式。

3.3.4 SPI 时序

串行时钟 SPICLK 负责同步 SPI 总线的数据传输，每个时钟脉冲传送 1 比特数据[24][25]。主机输出 SPICLK 信号并输入从机。时钟极性（CPOL）可以控制时钟的高电平或低电平为有效状态。CPOL=“0”表示 SCK 高电平有效；而 CPOL=“1”则表示 SCK 低电平有效。

此外，时钟相位信号 CPHA=“0”，表明数据在信号 SS 声明后的第一个 SCK 边沿有效；而 CPHA=“1”表明数据在信号 SS 声明后的第二个 SCK 边沿有效。SPI 主机与从机时钟相位和极性设置必须相同。图 3.6 为四种不同相位与极性的

时钟控制下的数据传输模式：

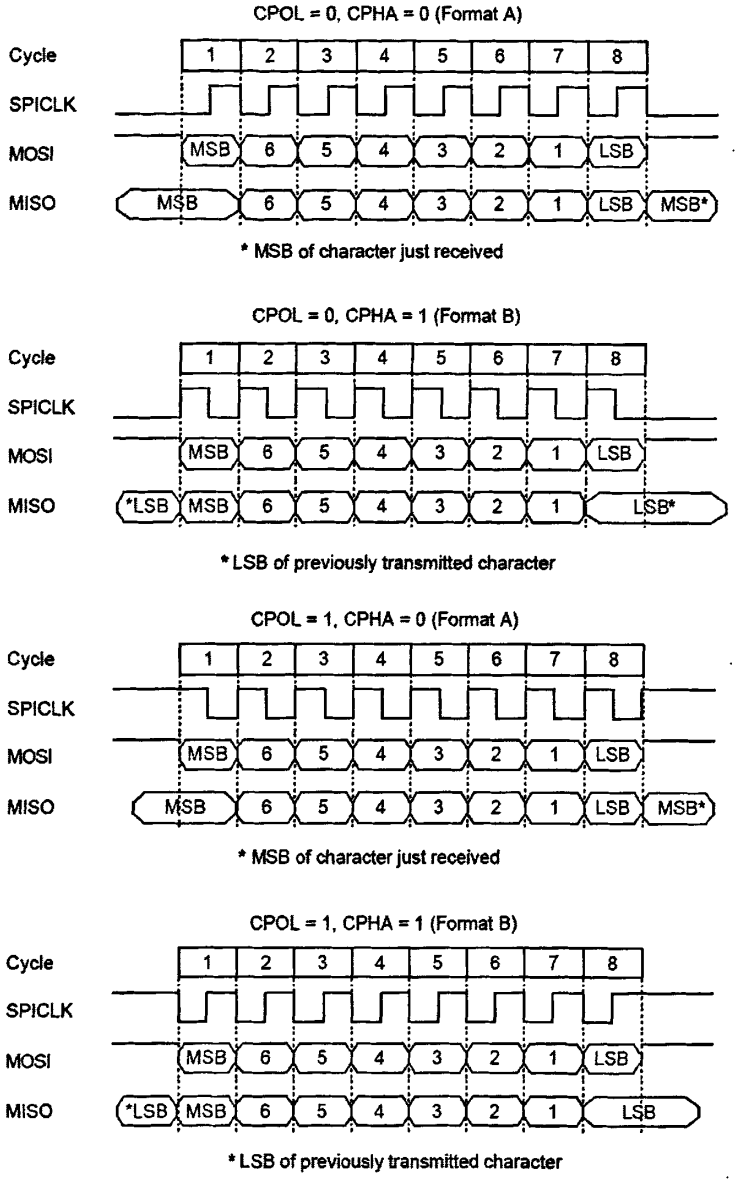

图 3.6 SPI 的数据传输模式[33]

3.4 本章小结

本章主要内容是介绍了几种目前常用且影响较大的片上总线规范，通过对比分析说明了本文设计为何以 AMBA APB 总线作为接口，并且给出了 APB 总线的读写操作时序图。之后详细介绍了 SPI 协议的信号线和时序，为下文的 IP 核设计做好了准备。

第 4 章 SPI 可复用 IP 核的设计和实现

前面章节介绍了 IP 核设计的产业背景、技术背景和设计流程，并且详细叙述了 AMBA APB 总线以及 SPI 协议的技术规范。以此为基础，本章将详细介绍基于 AMBA 总线 SPI 可复用 IP 核的设计过程，以及使用 Verilog HDL 硬件描述语言实现本 IP 核的思想、方法和关键代码分析。

4.1 设计目标

本 SPI 协议 IP 核的作用是向外提供满足 SPI 协议要求的输入/输出信号，可以和其它设备按照 SPI 协议的时序进行通信；可以通过 APB 总线连接 APB HOST。APB HOST 通过 APB 总线接口对 SPI 模块进行控制、数据读写、状态察看等操作。同时 SPI 也可向 APB HOST 发出中断请求。

本 IP 核模块采用自顶向下的设计方法。首先设计模块的顶层，然后进行功能划分细化为子模块。之后确定子模块之间的信号互联，以及数据格式等技术细节，得到较为完善的模块结构图。

4.2. 模块划分

4.2.1 顶层设计

根据设计目标的要求，顶层的对外信号接口应该包括：APB 总线的接口信根据设计目标的要求，顶层的对外信号接口应该包括 APB 总线的接口信号：时钟 PCLK，片选 PRESETn、PSEL，数据传输使能 PENABLE，写入使能 PWRITE，地址线 PADDR,写入数据线 PWDATA,读取数据线 PRDATA；SPI 协议规定的通信线：MOSI, MISO, SCLK, 片选信号 SS，以及中断信号 INTR。

从模块的角度看，数据传输信号 MOSI 和 MISO 可以等效为数据输入信号 Rx 和数据输出信号 Tx；同步时钟 SCLK 是 SPI 主模块输出信号、从模块的输入信号，而本模块的一个设计目标是和根据需要配置成主模块或从模块，既需要发送又需要接收时钟信号，因此将 SCLK 设计成 CLKOUT 和 CLKIN 接口。同理，片选信号设计成用于主模式的输出信号 FSSOUT 和用于从模式的输入信号 FSSIN。得到 SPI 顶层模块如图 4.1 所示。

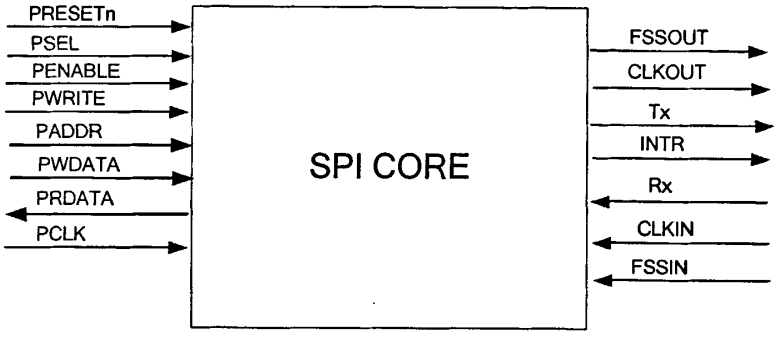

图 4.1 SPI 顶层模块

当模块被配置为主模式 Master 时，可按图 4.2 所示和 SPI slave 模块连接。

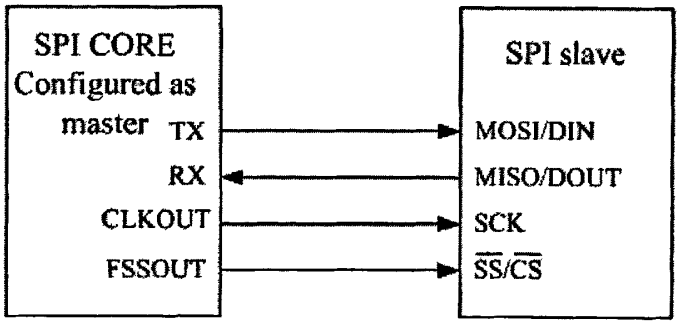

图 4.2 模块配置为 SPI master 时的连接方式

当模块被配置为 Slave 时，可按图 4.3 所示和 SPI master 模块连接。

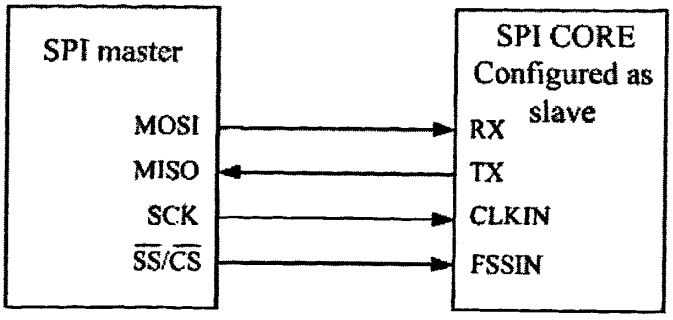

图 4.3 模块配置为 SPI Slave 时的连接方式

其中从模式下 FSSIN 如果不接 Master 的片选信号则必须接 0。

4.2.2 模块划分和接口信号描述

完成顶层模块设计之后，下一步的目标是将 SPI 顶层模块划分为若干子模块。划分依据是通过分析和分解设计目标，根据功能需求分别确定各子模块的作用。

1. 为了和 APB HOST 按照 APB 总线的时序逻辑进行通信，需要一个 APB 总线的接口模块 APB interface；

2. 需要一个逻辑控制模块 Transmit/Recieve Logic，按照 SPI 协议规定的时序逻辑来发送和接收数据；

3.为了匹配 APB 总线和高速时钟和外部设备的低速时钟，需要时钟分频模块 Clock Prescaler，从 APB 总线的时钟信号 PCLK 分频产生控制发送和接收数据的时钟信号 SCLK；

4.考虑到因为 SPI 的数据传输速度和 APB 总线速度不同，为了与 APB 总线速度匹配，对应发送和接收功能各需要一个 FIFO 模块，用于读写操作时缓存数据。由 APB 总线向外设写数据时，数据先被写入发送 TxFIFO 缓存，然后，再从 TxFIFO 读出送到负责发送数据的逻辑控制模块；读数据时，则先从逻辑控制模块读取道 RxFIFO，再送到 APB 接口模块。

此外，还需要中断控制模块，以及寄存器控制模块等。进一步分析确定以上各内部模块之间的互联信号，可以得到图 4.4 所示的详细的模块结构图。

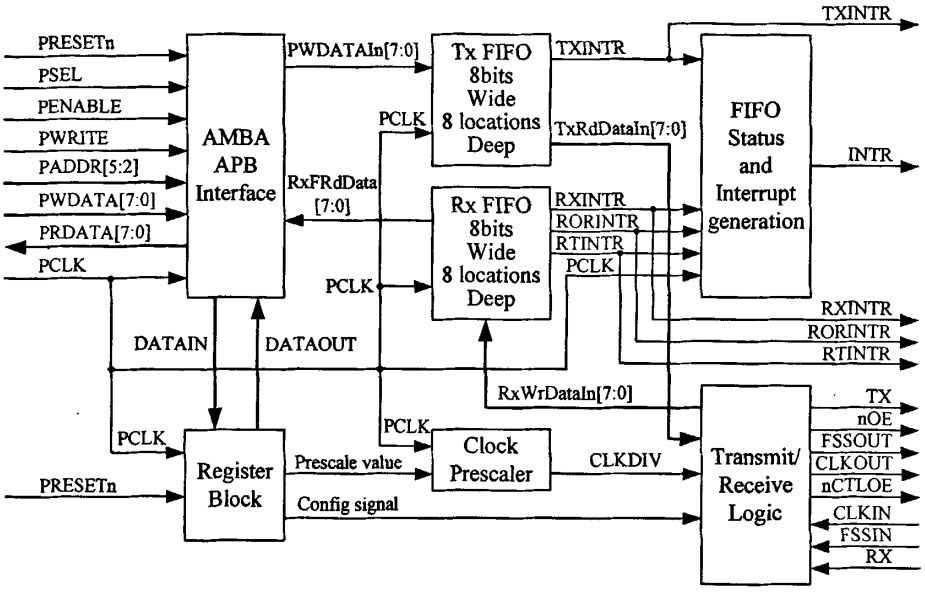

图 4.4 SPI IP 核模块结构框图

对各个模块功能的简单描述如下：

AMBA APB Interface:

AMBA APB interface 负责地址解码，控制各个寄存器的读写。

Register Block:

Register Block 存储控制寄存器、状态寄存器的数据。

Clock Prescaler:

当配置为 Master 时，Clock prescaler 将 PCLK 分频产生输出时钟信号 CLKOUT。

Transmit FIFO:

Transmit FIFO 是数据位宽为 8，深度为 8 的先进先出缓存，APB HOST 通过 AMBA APB 总线接口写入的待传输数据将缓存在这里直到发送电路将其读出。

Receive FIFO:

Receive FIFO 是数据位宽为 8，深度为 8 的先进先出缓存，接收电路接收到的数据存储在缓存里直到 APB HOST 通过 AMBA APB 总线接口将数据读出。

Transmit and receive logic:

当配置为 Master 时，发送电路从 Transmit FIFO 读出待发送数据，进行并

串转换后将串行数据从 TX 端口发出。

当配置为 Slave 时，接收电路在 CLKIN 的控制下完成串并转换，并将并行数据写入 Receive FIFO 等待 APB HOST 读取。

Interrupt generation logic:

SPI 模块产生四个独立可屏蔽中断，发送请求、接收请求、接收溢出以及接收超时。中断控制模块 Interrupt generation logic 产生一个综合中断信号，为以上四个信号的或函数。

接口信号描述：

PRESETn：复位信号，有效时模块回到初始状态。
PSEL：片选信号，有效时选中模块，方可进行操作。
PENABLE：操作使能信号。读写操作需该信号有效。
PWRITE：写入使能信号。当 PSEL 有效且该信号为高电平，表示写入操作；PSEL 有效且该信号为低电平表示读取操作。
PADDR[5:2]：地址信号。指定操作的目标寄存器。
PWDATA[7:0]：写入数据信号。可写入控制命令或待发送的数据。
PRDATA[7:0]：读取数据信号。可读取寄存器的值或接收到的数据。
PCLK：APB 总线时钟。

以上为 APB 总线接口信号。下面是和 SPI 串行通信相关的信号。

RX：串行数据输入信号。
TX：串行数据输出信号。
CLKOUT：SPI Master 时钟输出信号。若配置为 Slave 则该信号无用。
CLKIN：SPI Slave 时钟输入信号。若配置为 Master 则该信号无用。
FSSOUT：SPI Master 帧控制信号，当数据传输开始时变为低电平，结束后回复高电平。若配置为 Slave 则该信号无用。可作为 SPI 从机选择信号 SS 输出端。
FSSIN：SPI Slave 帧控制输入信号，若配置为 Master 则该信号无用。可作为 SPI 从机选择信号 SS 输入端。
nOE：TX 信号的输出使能信号，低电平有效。
nCTLOE：FSSOUT 和 CLKOUT 的输出使能信号。
中断输出信号：
TXINTR：发送服务请求中断，高电平有效。当发送 FIFO 中的待发送数据个数小于等于 4 时该中断发出。
RXINTR：接收服务请求中断，高电平有效。当接收 FIFO 中的待读取数据个数大于等于 4 时该中断发出。
RORINTR：接收溢出中断，高电平有效。当接收 FIFO 已满时又有新的数据接收到时该中断发出。
RTINTR：接收超时中断，高电平有效。当接收 FIFO 中仍有数据未被读出，并且传输停止的超过 32 个 PCLK 周期时，该中断发出。

INTR：综合中断信号，为上述四个中断信号的或逻辑。

4.3 寄存器设置

寄存器设计是本 IP 核模块设计中的基础环节。 APB HOST 通过读取寄存器就可以知道本 IP 核所处的状态，通过写寄存器可以对模块进行控制。

现将本 IP 核使用的寄存器说明如下：

1、SPI Control Register：SCR （0x00）

Bits	Name	R/W	Reset Value	Description
4	SE	R/W	0x0	SPI 使能信号。0：禁用 1：使能
3	SOD	R/W	0x0	Slave Output Disable 当配置为Slave模式时如果不需要输出数据则可将此位置1。 0：输出数据 1：输出禁用
2	MS	R/W	0x0	Master或Slave配置信号0：Master 1：Slave
1	CPHA	R/W	0x0	Clock Phase
0	CPOL	R/W	0x0	Clock Polarity

2、SPI Data Register：SDR （0x04）

Bits	Name	R/W	Reset Value	Description
[7:0]	SDR	R/W	X	SPI数据寄存器。写该寄存器数据将写入发送FIFO，读该寄存器数据将从接收FIFO中读出。

3、SPI Status Register: SSR (0x08)

Bits	Name	R/W	Reset Value	Description
4	BSY	RO	0x0	SPI busy标志位 0：SPI空闲 1：SPI传输中
3	RFF	RO	0x0	接收FIFO是否已满标志位 0：未满 1：已满
2	RNE	RO	0x0	接收FIFO是否为空标志位 0：为空 1：未空
1	TNF	RO	0x1	发送FIFO是否已满标志位 0：已满 1：未满
0	TFE	RO	0x1	发送FIFO是否为空标志位 0：未空 1：为空

4、Clock Prescale Register：CPSR （0x0c）

Bits	Name	R/W	Reset Value	Description

[7:0]	CPSR	R/W	0x00	$CLKOUT = \frac{PCLK}{2 \times (1 + CPSR)}$

5、Interrupt Mask Set or Clear Register: IMSC (0x10)

Bits	Name	R/W	Reset Value	Description
3	TXIM	R/W	0x0	TXINTR中断屏蔽信号 0：被屏蔽 1：未屏蔽
2	RXIM	R/W	0x0	RXINTR中断屏蔽信号 0：被屏蔽 1：未屏蔽
1	RTIM	R/W	0x0	RTINTR中断屏蔽信号 0：被屏蔽 1：未屏蔽
0	RORIM	R/W	0x0	RORINT中断屏蔽信号 0：被屏蔽 1：未屏蔽

6、Raw Interrupt Status Register：RIS （0x14）

Bits	Name	R/W	Reset Value	Description
3	TXRIS	RO	0x1	给出TXINTR未被屏蔽时的值
2	RXRIS	RO	0x0	给出RXINTR未被屏蔽时的值
1	RTRIS	RO	0x0	给出RTINTR未被屏蔽时的值
0	RORRIS	RO	0x0	给出RORINTR未被屏蔽时的值

7、Masked Interrupt Status Register： MIS （0x18）

Bits	Name	R/W	Reset Value	Description
3	TXMIS	RO	0x0	给出TXINTR被屏蔽之后的值
2	RXMIS	RO	0x0	给出RXINTR被屏蔽之后的值
1	RTMIS	RO	0x0	给出RTINTR被屏蔽之后的值
0	RORMIS	RO	0x0	给出RORINTR被屏蔽之后的值

8、Interrupt Clear Register： ICR （0x1c）

Bits	Name	R/W	Reset Value	Description
1	RTIC	WO	-	写入1时将RTINTR清除，写0无效
0	RORIC	WO	-	写入1时将RORINTR清除，写0无效

4.4 IP 核的 Verilog HDL 实现

完成 IP 核的模块划分和寄存器定义之后，需要对系统进行 HDL 语言建模实现，并进行综合和仿真，以验证 IP 模块设计的正确性。本项目使用 Verilog HDL 语言对设计的 SPI 协议 IP 核模块进行建模和实现。

4.4.1. Verilog HDL 简介

Verilog HDL 硬件描述语言(Verilog Hardware Description Language)是目前应用最为广泛的硬件描述语言[28]。Verilog HDL 可以用来进行各种层次的逻辑设计，也可以进行数字系统的逻辑综合，仿真验证和时序分析等。Verilog HDL 进行设计最大的优点是其工艺无关性. 因此在功能设计，逻辑验证阶段可以不必过多考虑门级及工艺实现的具体细节。

Verilog 语言风格和 C 语言有点接近。其设计的基本单元是模块 Module，对应于数字电路中的电路模块。每个模块按照其输入输出信号分别建模。一个模块的基本要素包括两部分：端口和逻辑功能——输入如何影响输出。

4.4.2 设计思路

进行本 IP 核的 Verilog HDL 建模时，基本的思路是分别设计各个模块的 Verilog HDL 代码，对各子模块进行建模，然后将各个子模块进行集成。在顶级层次只需定义所有的输入输出端口、对各个子模块进行实例化，并定义好子模块之间的连线即可。

子模块设计时，和整体模块设计思路类似，首先分析功能，明确所有输入和输出信号，如有必要，可以继续划分出下一级子模块。直到不再继续划分时，该子模块的 Verilog 代码设计主要目标是完成端口定义、内部连线定义、设计输入信号如何影响输出量的逻辑转换功能。

下面结合具体子模块叙述本 IP 核的设计过程。

4.4.3 时钟、寄存器和中断模块的设计

时钟分频模块：

当模块被配置为 SPI Master 时，时钟分频模块 Clock prescaler 将 APB 总线时钟 PCLK 分频，产生内部时钟信号 CLKDIV。分频功能利用倒数计数器实现，计数初值保存在寄存器 CPSR。CLKDIV 的空闲状态为低电平，每个 PCLK 上升沿时计数器值减 1，减到 0 时 CLKDIV 置为高电平，持续一个 PCLK 周期，然后回归低电平。CLKDIV 经过收发逻辑模块，产生频率为其一半的 CLKOUT 信号，该信号是 SPI 通信的主时钟信号。分频公式为：

$$CLKDIV = \frac{PCLK}{1+CPSR}$$

$$CLKOUT = \frac{PCLK}{2\times(1+CPSR)}$$

模块部分代码如下：

```
//=============================================================//
`timescale 1ns/1ps   //仿真的时间精度是 1ps，代码中数字 1 代表 1ns.

module SspScaleCntr
  (
        SSPCLK          ,
        nSSPRST         ,
        SSESync         , // SPI 模块综合判断后的使能信号
        SSPCPSR         ,
        SSPCLKDIV           // SPI 模块内部分频得到的时钟
  );


input       SSPCLK      ;     // SPI 通讯主时钟
input       nSSPRST     ;     //APB 总线重启信号
input       SSESync     ;     // SPI 模块综合判断后的使能信号
```

```
input  [7 :0]   SSPCPSR          ;        //寄存器 CPSR
output          SSPCLKDIV        ;        // SPI 模块内部分频得到的时钟
reg [7:0]   SSPCPSC;        // SPI 时钟分频计数器
reg [7:0]   NextSSPCPSC;  // SSPCPSC 下个输入值

always @(negedge nSSPRST or posedge SSPCLK)  //每个 PCLK 计数值变化一次
  begin
    if(!nSSPRST)
      SSPCPSC <= 8'b0;
    else
      SSPCPSC <= NextSSPCPSC;
  end


always @(SSESync or SSPCPSC or SSPCPSR)
begin
  if (!SSESync | (~|SSPCPSC) )
    NextSSPCPSC = SSPCPSR;
//如果片选失效，或 CPSC 减小到 0，则下个值重新装载 CPSR 初值
  else
    NextSSPCPSC = SSPCPSC - 8'b1;  //计数值递减 1
end

assign SSPCLKDIV = (SSESync)? (~|SSPCPSC) : 1'b0;
// SSPCPSC 减到 0 时 CLKDIV 置为高电平，持续一个 PCLK 周期，然后回归
低电平。
endmodule

//========================= End ==========================//
```

寄存器模块

Register Block 存储控制寄存器、状态寄存器和数据寄存器的数据。其中控制寄存器 SCR0,SCR1 可以设置四种 SPI 时序，并设定模块为 SPI Master/Slave 模

式；状态寄存器 IMSC 指示各中断的屏蔽状态；数据寄存器 SDR 保存发送/接收数据。

中断控制模块：

SPI 模块产生四个独立可屏蔽中断：发送服务请求中断，当发送 FIFO 中的待发送数据个数小于等于 4 时该中断发出。接收服务请求中断，当接收 FIFO 中的待读取数据个数大于等于 4 时该中断发出。接收溢出中断，当接收 FIFO 已满时又有新的数据接收到时该中断发出。接收超时中断，当接收 FIFO 中仍有数据未被读出，并且传输停止的超过 32 个 PCLK 周期时，该中断发出。

由 Interrupt generation logic 产生一个综合中断信号，该信号是以上四个信号的或函数。

4.4.4 子模块 APB interface 的设计

APB 总线接口模块 APB interface 的外部连接比较繁多，包括 APB 总线的数据线、地址线、控制线等，同时，它与内部其它模块如 FIFO 模块、寄存器模块等许多模块相联系，因而其设计是一个重点。

其主要功能为：地址译码、读取和写入寄存器，并实现根据片选信号选通或关闭内部数据 PWDATAIn[7:0]的功能。

实现地址译码和读取/写入寄存器操作的思路是：首先将各个寄存器的地址保存到参数数组中，例如寄存器 SSPCR1 的地址保存为常量 PA_SSPCR1,寄存器 SSPDR 的地址则保存为常量 PA_SSPDR; 当需要读取或写入时，将 APB 总线发送过来的地址和 这些地址常量一一对比，如果和某个地址常量相符，则可以知道是要对和该常量相应的寄存器进行读写操作。其中，读写状态的判定方法是：当片选信号 PSEL 有效时且写入信号 PWRITE、写操作使能信号 PENABLE 均有效时判定为写状态，当 PSEL 有效，且 PWRITE 无效时判断为读取状态。

此外，本项目设计了由 APB 总线的片选信号 PSEL 控制选通或者关闭内部数据总线 PWDATAIn[7:0] 的功能。实现方法是，当片选 PSEL 且写入状态 PWRITE 有效时，将数据总线 PWDATA[7:0] 赋值给内部数据总线

PWDATAIn[7:0]；当二者有一个无效时，则关闭内部总线，将其各位全部置为 0 以达到省电效果。

以上功能的对应的实现代码是：

```
module SspApbif
(
        PCLK                    , // APB Bus Clock
.....
}
.....

parameter [4:2] PA_SSPCR1    = 3'b000,    //0x00,
                PA_SSPDR     = 3'b001,    //0x04
                PA_SSPSR     = 3'b010,    //0x08
                PA_SSPCPSR   = 3'b011,    //0x0c
                PA_SSPIMSC   = 3'b100,    //0x10
                PA_SSPRIS    = 3'b101,    //0x14
                PA_SSPMIS    = 3'b110,    //0x18
                PA_SSPICR    = 3'b111;    //0x1c

//保存各个寄存器的地址到常量数组中
.....
assign GatedPA = (PSEL == 1'b1) ? PADDR : 3'b000;
//如果 PSEL 为高电平，将地址线 PADDR 上的数据赋值给 GatedPA; 否则
//GatedPA 置 0.

assign PWDATAIn = ((PSEL ==  1'b1) & (PWRITE == 1'b1 )) ? PWDATA :
8'b0000_0000;
//如果未被选中或不允许写入，内部数据总线 PWDATAIn[7:0] 置零。否则
将//PWDATA[7:0] 赋值给内部数据总线 PWDATAIn[7:0]

assign WrEn       = PENABLE & PSEL & PWRITE;    //判定写入状态；
```

```
//以下实现寄存器写操作
// SSPCR1
assign SSPCR1Wr = ((WrEn == 1'b1) & (GatedPA == PA_SSPCR1)) ? 1'b1 : 1'b0;
//如果处于写入状态，且 APB 总线发送的地址 和常量 PA_SSPCR1 相同，则寄//存器 SCR1 的写入状态标志 SSPCR1Wr 置为 1（写入），否则为 0（不写入）
// SSPDR
assign SSPDRWr = ((WrEn == 1'b1) & (GatedPA == PA_SSPDR)) ? 1'b1 : 1'b0;
……

assign RdEn = PSEL & (!PWRITE); //判定为读取状态
//以下实现寄存器读取操作
// SSPCR1
assign SSPCR1Rd = ((RdEn == 1'b1) & (GatedPA == PA_SSPCR1)) ? 1'b1 : 1'b0;
//如果处于读取状态，且 APB 总线发送的地址 和常量 PA_SSPCR1 相同，则寄//存器 SCR1 的读取状态标志 SSPCR1Rd 置为 1（读取），否则为 0（不读取）
// SSPDR
assign SSPDRRd = ((RdEn == 1'b1) & (GatedPA == PA_SSPDR)) ? 1'b1 : 1'b0;
……
endmodule
```

4.4.5. 子模块 RxFIFO 的设计

RxFIFO 和 TxFIFO 是先进先出缓存，分别执行读写操作，其功能相对应，内部实现的结构也类似。选择一个为代表即可。另一个的设计过程与其类似。

首先，明确设计功能和接口。RXFIFO 是先进先出缓存，接收电路接收到的数据存储在其中，直到 APB HOST 通过 AMBA APB 总线接口将数据读出。同时，

还需要向外提供接收请求中断 RXINTR 等中断。

分析其有两项主要功能：缓存数据，产生中断。一般来说，中断由控制模块产生。根据控制模块和数据传输模块分开的原则，将 RxFIFO 进一步划分为专门负责状态控制的 RxCtnl 和负责数据缓存的 Regfile.

RxFIFO 模块的部分代码如下：

```
module SspRxFIFO
(
        PCLK            ,  // APB bus clock
......
        RxFRdData       // RX FIFO read data
);
input           PCLK          ;  // APB bus clock
input           PRESETn       ;  // Muxed reset (from BnRES)
......
wire [2:0]      WrPtr         ;  // RXFIFO Write Pointer
wire [2:0]      RdPtr         ;  // RXFIFO Read Pointer
//以上定义模块的输入/输出信号，和模块内部的连接线

SspRxFCntl
uSspRxFCntl //实例化状态控制子模块 RxFCntl.
  (

            .PCLK          ( PCLK ),
            .PRESETn       ( PRESETn ),
            .RXIM          ( RXIM ),
            .RORIM         ( RORIM ),
            .RORIC         ( RORIC ),
           ......

            .RegFileWrEn ( RegFileWrEn ),
```

```
            .WrPtr          ( WrPtr ),
            .RdPtr          ( RdPtr )
        );
```

//括号中是 RxFCntl 的端口定义，其中包含 RXIM, RORIM 等中断寄存器的位信号，用于实现状态判断和控制。

```
SspRxRegFile
uSspRxRegFile //实例化数据缓存子模块 RxRegFile.
    (
            .PCLK           ( PCLK ),
            .MS             ( MS ),
//寄存器 MS 位，用于选择 MASTER 模式或者 SLAVE 模式

            .SRxFWrData     ( SRxFWrData ),
            .MRxFWrData     ( MRxFWrData ),
            .RegFileWrEn    ( RegFileWrEn ),
            .WrPtr          ( WrPtr ),
            .RdPtr          ( RdPtr ),
            .RxFRdData      ( RxFRdData )
        );
//
endmodule
```

模块 RXFIFO 的代码中，除了进行端口定义和内部连线定义，剩下的部分就只是实例化了两个子模块。

可以看出，模块控制子模块 RxFCntl 的端口定义中包含了 RXIM, RORIM 等中断寄存器的位信号，用于实现状态判断和控制。而缓存子模块 RxRegfile 的端口定义中不含中断信号，含有的只是读取数据信号和写入数据信号，以及它们的数据指针，专门负责传送数据。

此外，RxRegfile 的端口信号中还含有寄存器 MS 位信号，此信号置 0 或置 1 可以将模块配置为 Master 模式或 Slave 模式。在这两种模式下，ReRegfile 中读取的数据 RxFRdData 会被赋予不同的值。Master 模式下 MRxFWrData 会被

赋值给 RxFRdData，而 Slave 模式下 SRxFWrData 会被赋值给 RxFRdData。而 MRxFWrData 和 SRxFWrData 分别是来自于在 MASTER 和 SLAVE 模式下完成数据收发逻辑的模块。也就是说，为了实现 Master 和 Slave 两种配置模式，本 IP 模块的收发数据逻辑子模块 Transmit/Receive Logic 被设计成了分别在两种状态下有效的相互独立的两个模块。对此在后文会进一步介绍。

4.4.6 子模块 Transmit/Receive Logic 的设计

收发数据逻辑子模块 Transmit/Receive Logic 的主要设计功能如下：

1. 数据发送时，将来自发送 TxFIFO 核内的 8 位并行数据转换成串行数据 TXD 输出。

2. 数据接收时，将来自 RXD 的串行数据转换成 8 位并行数据，并传送到 RxFIFO。

3. 按照 SPI 协议时序的要求，在每个 SCLK 周期传送一位数据。

4. 实现 Master/Slave 选择的功能，可以被设置为 以 SPI 主模式或 SPI 从模式方式和外界通信。

该模块是所有子模块中功能要求最多、结构最复杂的模块，是设计的难点。下面重点介绍其设计的思想和方法。

Master/Slave 模式选择的功能的实现的方法是，分别设计两个独立的模块 MTxRxCntl 和 STxRxCntl，各自完成在 Master 模式和 Slave 模式下的数据收发逻辑的功能，然后利用寄存器位 M/S 作为片选信号，同一时间只选通其中一个模块的输出信号。这两个模块的功能和结构在主要方面是类似的，只是输入输出信号稍有差异。在此以 Master 模块 MTxRxCntl 为例来说明收发逻辑模块的设计方法。

发送和接收数据的串并转换设计：待发送数据，会被从 TxFIFO 拷贝到一个内部的发送移位寄存器中。然后从该移位寄存器逐位移出到发送数据信号线 TXD 上。而从 RXD 得到的接收数据被送到一个内部移位寄存器。当收到一个完整的祯时，接收到的数据会从移位寄存器被复制到接收缓存 RxFIFO 中。

实现 SPI 协议时序的方法是分析协议时序，将数据传输过程分解为若干阶段，每个阶段用有限状态机的一个状态描述。主模式下数据传输有限状态机如图 4.5 所示：

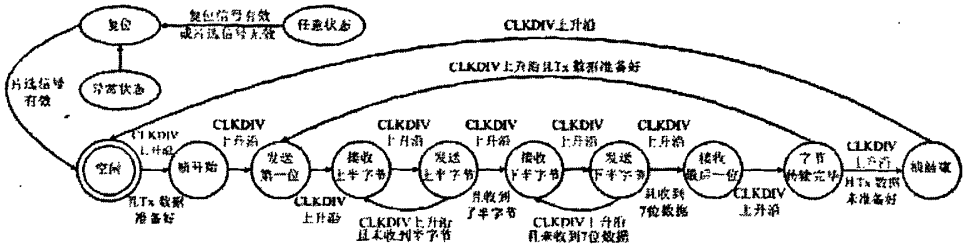

图 4.5 主模式下数据传输的有限状态机

初始状态为空闲状态。Tx 发送数据准备好后，在下个 CLKDIV 上升沿进入帧开始状态，并将帧控制输出信号 FSSOUT 置为低电平。之后从发送第一位数据状态开始数据传输，状态转换由 CLKDIV 上升沿触发。发送状态和接收状态相互交替，每个状态发送或接收一位数据，持续一个 CLKDIV 周期。发送和接收状态下将 SPI 时钟输出信号 CLKOUT 置为相反电平，因此收发一位数据构成一个完整的 CLKOUT 周期，CLKOUT 的频率是 CLKDIV 的一半。字节传输完毕后，若 Tx 数据未准备好则判定为帧结束，FSSOUT 恢复至高电平，并回到空闲状态；若 Tx 准备好则判定为连续字节传输过程，转移到发送第一位状态开始新字节的传输。

此外，在任意状态下 APB 总线的复位信号 nSSPRST 有效，或片选信号 PSEL 失效，或检测到模块处于异常状态，则下一状态置为复位状态。之后若片选信号有效则转换到空闲状态。

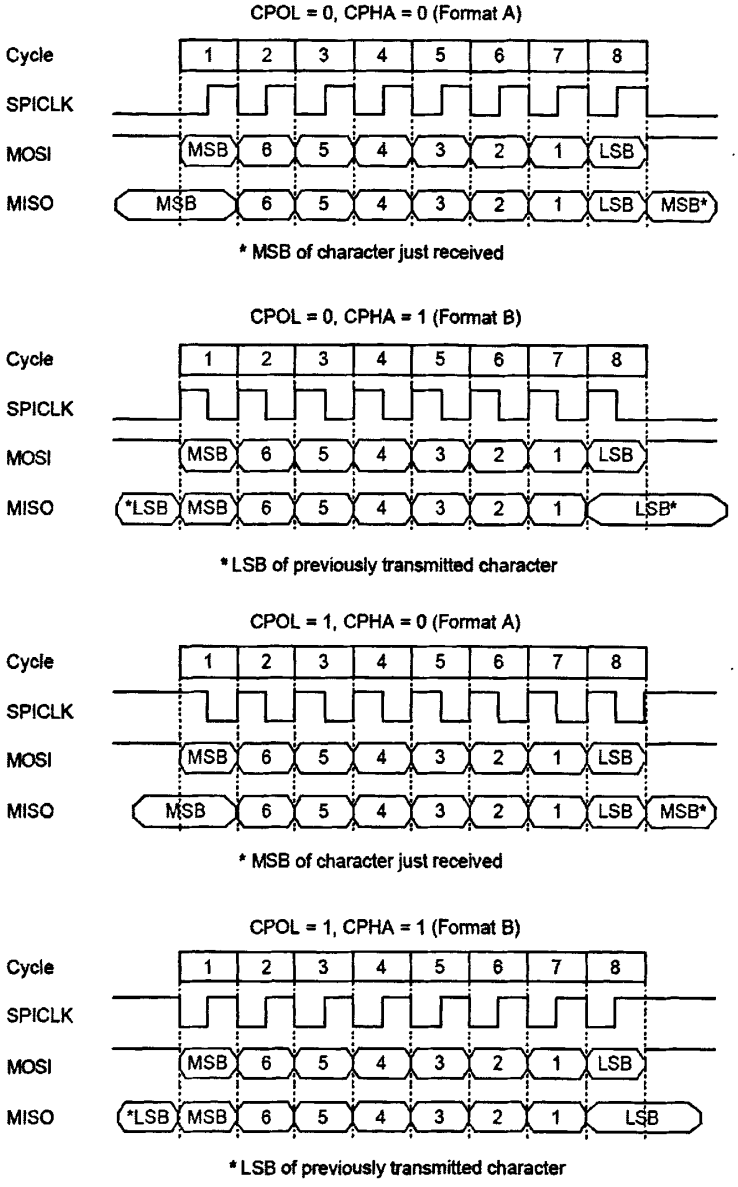

图 4.6 SPI 的数据传输模式[33]

如图 4.6 所示，SPI 协议规定，在每位数据传输时钟周期的第一个时钟沿发送数据，在第二个时钟沿（即中间的时钟沿）接收数据，这样可确保接收数据稳定。分析所有四种传输模式发现，在每位数据传送周期的前半周都符合 SPICLK=CPOL^CPHA, 而后半周 SPICLK=!CPOL^CPHA。而 Master 模块的状态机的设定是，前半个 CLKOUT 周期为发送状态，后半周期为接收状态，因此可在发送状态将 CLKOUT 电平置为 SPO^SPH，而在接收状态将 CLKOUT 置为 !SPO^SPH，其中 SPO 和 SPH 分别对应 CPOL 和 CPHA 的寄存器设置值。

这样就能兼容 SPI 协议规定的四种传输模式。

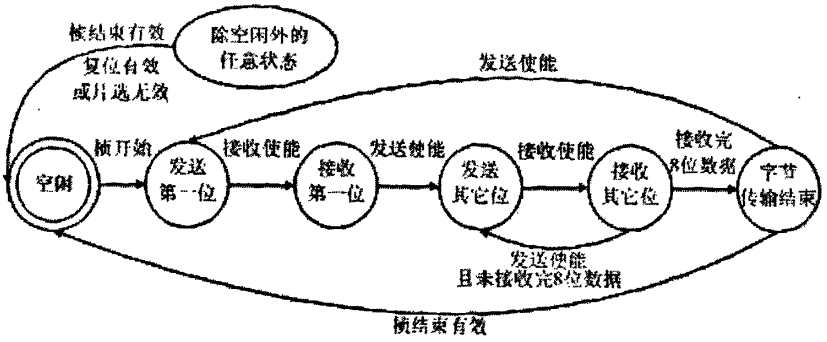

图 4.7 从模式下数据传输的有限状态机

如图 4.7 所示，Slave 和 Master 模式的相同点是发送和接收交替，每个状态发送或接收一位数据；在完成当前字节传输后同样可返回空闲或进行连续传输。不同点主要在于触发条件。Master 模式开始传输的触发条件是 Tx 数据准备好，而 Slave 模式则是收到来自 Master 的桢开始命令——FSSIN 低电平；Master 模式发送和接收的状态转换由其内部时钟信号 CLKDIV 的上升沿驱动，而 Slave 模式下则由 Master 输入的 CLKIN 信号的边沿派生的发送使能和接收使能信号驱动。此外 Master 拥有对数据传输的的控制权，若 Slave 在传输中途收到来自 Master 的桢结束命令，会中止传输并返回空闲状态。

Slave 模式为兼容四种传输时序，定义发送使能为 CLKIN 下降沿且 SPO^SPH=0（即 CPOL,CPHA=00,11）；或 CLKIN 上升沿且 SPO^SPH=1（即 CPOL,CPHA=10,01）。定义接收使能为 CLKIN 上升沿且 SPO^SPH=0（即 CPOL,CPHA=00,11）；或 CLKIN 下降沿且 SPO^SPH=1（即 CPOL,CPHA=10,01）。对照图 1 可知，这样可确保发送使能对应于每位数据传输周期的第一个边沿，而接收使能对应于中间的时钟沿。

以上三个小节选取了三个重要的子模块，各自详细介绍了其如何进一步划分子模块、确定关键函数的实现细节、以及面对复杂问题的设计思路问题。以上的三个侧重点，是所有子模块设计过程中需要考虑的典型问题。由于篇幅所限，本文无法详述本 IP 核在 Verilog HDL 代码实现过程中的每一个步骤。其它子模块的详细设计过程可以参考对以上的三个子模块的介绍。

4.5 小结

本章根据 Top-Down 的设计思路，描述了基于 AMBA 总线的 SPI IP 核模块的设计方法。首先分析设计目标，定义了模块外围的接口；进而根据设计功能划分子模块，进一步分析其内部互连信号，给出了完整的子模块和信号连接图，以及详细的端口连接和寄存器的设置。

本章的后半部分描述了本 IP 核的 Verilog HDL 实现过程。首先阐述了整体的设计思路，进一步以关键子模块为例介绍了 Verilog 代码设计过程中的几种典型问题与其解决过程。并给出了模块的综合电路图。

第5章 仿真验证

仿真验证是 IP 核设计过程中非常重要的环节。通过仿真，可以运行设计代码，加载设定好的测试输入量，分析输出量是否符合预期，以此来验证模块设计的正确性。

5.1 仿真原理概述

本设计采用软件仿真以及业界通用的 Modelsim+ QuartusII 的模式。验证流程是用 Verilog 语言写出测试向量，首先在 Modelsim 中进行功能仿真，然后选择 FPGA 芯片，利用 QuartusII 综合，得到网表文件和延时文件后再回到 Moldelsim 中进行时序仿真。

5.2 测试代码设计

总体的测试思路是，建立一个顶层的 Testbench 模块，在其内部实例化两个 SPI 模块 u0SPI 和 u1SPI，分别配置为主模块和从模块，相互连接进行数据传输。在 Testbench 模块中把 APB 总线信号设为寄存器变量，向两个 SPI 模块发送命令并读写数据。在测试代码主任务中调用各子任务，并在子任务完成后给出完成时间。主要有三项子测试任务：

1、reset_test：重启后读取各个寄存器，判定其初值是否正确。

2、CPSR_test：测试不同的时钟分频数值下的数据传输是否正确。将 u0 设为 Master 模块，u1 设为 Slave 模块，每次给 u0 的寄存器 CPSR 设置不同的初值，共设置 11 次，每种分频数值下传输一个字节数据。

3、txrx_test：发送和接收数据测试。测试 CPOL、CPHA 为 00、10、01、11 四种时钟模式下的数据发送和接收。首先将 u0 设为 Master 模块，u1 设为 Slave 模块，每种时钟模式下一次性传输 8 个字节（装满缓冲区）。完成四种模式的传输后将 u0 设为 Slave 模块，u1 设为 Master 模块，再次测试四种时钟模式下的数据传输。

测试中所有的传输数据都是利用系统函数$random 产生的随机数，CPSR 测试下的各时钟分频数值同样采用随机数。采用随机数作为传输数据的优点是，能

尽可能多的覆盖各种情况，提高测试的完备程度[]。通过 APB 总线信号 PWDATA 写入寄存器数值或待传输数据，传输完成后从 PRDATA 总线读取接收数据，并和初始写入的发送数据比较，如果不一致就报错并中断仿真。关键的实现写入和读取功能的测试代码以及 APB 总线读取时序和写入时序图如下（每次写入或读取一个字节）：

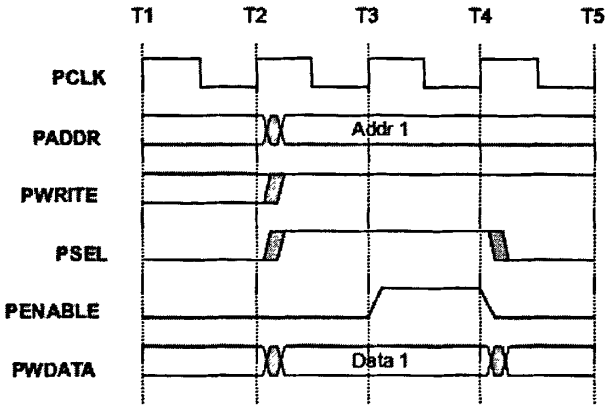

图 5.1　APB 总线写入数据时序图[28]

```
task ahb_write8;      //APB 总线写入任务
input[7:0] REG_AD; //输入量寄存器名，在测试代码开始处所有的寄存器//名被声明为该寄存器的地址。
input[7:0] writedata;//输入量，待写入的数据
begin
@(posedge PCLK)       //第一个 PCLK 上升沿
begin
PADDR<=REG_AD;          //将目标寄存器地址赋值给 APB 地址总线 PADDR
PWRITE<=1'b1;             //PWRITE=1 代表写入操作
PSEL<=1'b1;             //片选信号有效，选中目标模块
PENABLE<=1'b0;        //PENABLE 置0,确保本PCLK周期为非操作周期。
PWDATA<= writedata; //将待写入的字节数据赋值给 APB 数据总线
end
@(posedge PCLK)       //第二个 PCLK 上升沿
PENABLE<=1'b1;        //PENABLE 置 1，本 PCLK 周期为操作周期。
@(posedge PCLK)       //第三个 PCLK 上升沿
begin
```

```
PENABLE<=1'b0;          //操作周期完成，PENABLE 归 0
PSEL<=1'b0;             //片选信号失效
end
end
endtask
```

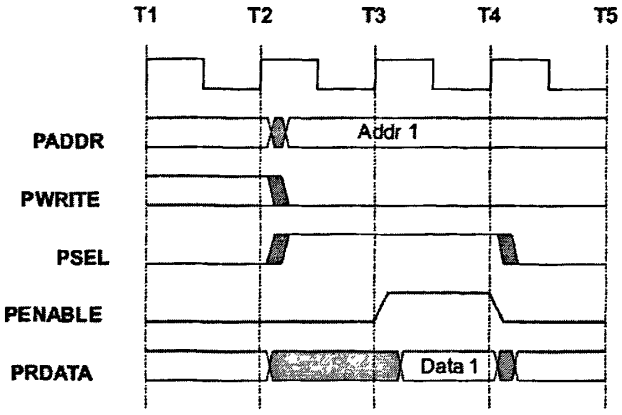

图 5.2　APB 总线读取数据时序图[28]

```
task ahb_read_expect8;  //APB 总线读取任务，含与预设数据比较的功能
input[7:0] REG_AD;  //输入量寄存器名，在测试代码开始处所有的寄存器//
名被声明为该寄存器的地址。
input[7:0]  data; //输入量，预设的数据，即将和读出的数据比较
begin
@(posedge PCLK)      //第一个 PCLK 上升沿
begin
PADDR <= REG_AD;    //将目标寄存器地址赋值给 APB 地址总线 PADDR
PWRITE<=1'b0;        //PWRITE=0 代表读取操作
PSEL<=1'b1;          //片选信号有效，选中目标模块
PENABLE<=1'b0;       //PENABLE 置 0，确保本 PCLK 周期为非操作周期
end
@(posedge PCLK)      //第二个 PCLK 上升沿
PENABLE<=1'b1;       //PENABLE 置 1，本 PCLK 周期为操作周期
@(posedge PCLK)      //第三个 PCLK 上升沿
begin
PENABLE<=1'b0;       //操作周期完成，PENABLE 归 0
PSEL<=1'b0;          //片选信号失效
```

```
    end

    if (PRDATA!=data)   //若 APB 总线读取的数据 PRDATA 和预设值 data 不同，
    begin               //则显示错误的寄存器地址，以及预期值和错误值
    $display("wrong,regaddress:%b,---expect:%b,          ---real:%b",REG_AD,
data,PRDATA);
    $Stop;              //读取数据和预期值不符时停止仿真
    end
    end
    endtask
```

5.3 仿真结果分析

顺利通过了功能仿真。选用低成本的 FPGA 验证芯片：Altera CycloneII 系列的 EP2C5F256C6，采用 QuartusII7.2 综合产生了正确的门级网表文件。资源占用较少，共使用 487 个逻辑单元（11%）和 37 个管脚（23%）。时序仿真时，通过尝试，本 SPI 模块的数据传输时钟 CLKOUT 频率可达 10.4MHz, 此时对应的 APB 总线时钟 PCLK 频率为 83.3MHz，分频比率为 1/8。

图 5.3 反映了一次完整的时序仿真过程的 Modelsim 控制台输出结果。可以看出，在 0ps 开始运行仿真，0.45us 时，reset_test 完成；在 test_CPSR 过程中，CPSR 的值共产生 11 个，并于 373us 时完成；之后开始的是数据传输测试 txrx，其中 407us 时完成 u0Master，u1Slave 部分的测试，442us 时完成另一半 u0Slave，u1Master 测试，同时 txrx 测试结束，整个 SPI 模块测试成功完成。

```
# .wave
 VSIM 8> run
# Loading spi_v.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /ajspi_tb/u0SPI
# Loading spi_v.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /ajspi_tb/u1SPI
# Time=              0 ps:==================Test SPI Start!==================
# Time=         450000 ps:==================reset_test done!==================
# ----i=        0--cpsr=11100011------
# ----i=        1--cpsr=11110010------
# ----i=        2--cpsr=00001000------
# ----i=        3--cpsr=01111100------
# ----i=        4--cpsr=11000000------
# ----i=        5--cpsr=10000001------
# ----i=        6--cpsr=11100100------
# ----i=        7--cpsr=00010010------
# ----i=        8--cpsr=11010011------
# ----i=        9--cpsr=11001000------
# ----i=       10--cpsr=00010110------
# Time=      372318000 ps:==================test_CPSR done!==================
# Time=      407166000 ps:==================u0Mu1S_test done!==================
# Time=      442014000 ps:==================u0Su1M_test done!==================
# Time=      442014000 ps:==================bnx done!==================
# ==================Test SPI Finished Successful==================
# Break at E:/My projects/modelsim/ssp_pl022/ajspi_tb.v line 170

VSIM 9>
```

图 5.3 Modelsim 时序仿真的控制台输出

下面依次分析各项子测试任务对应的 Modelsim 时序仿真波形。

1. 图 5.4 对应于重启测试。仿真图的信号分为三个部分，PCLK 直到 PRDATA 是 APB 总线信号，下方 FSSOUT01, CLKOUT01 和 TX0RX1 分别是 u0SPI 模块的桢控制输出信号、SPI 时钟输出信号以及数据发送信号。其下的 TX1RX0、CLKOUT10 和 FSSOUT10 则是 u1SPI 模块的发送数据信号、SPI 时钟输出信号和桢控制输出信号。

在时间接近 0ns 处，PRESETn 出现一个持续 1 个 PCLK 周期的低电平，对模块进行重启操作。之后若干个 PSEL 和 PENABLE 的有效脉冲对应于读取各寄存器的操作。此时 SPI 通讯尚未开始，波形图下方和 SPI 通讯相关的信号电平全部为 0。

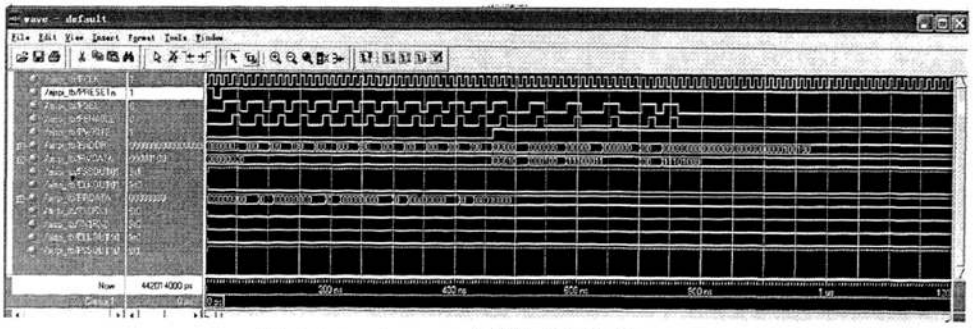

图 5.4 Modelsim 时序仿真图局部-reset_test

2. 图 5.5 对应于 CPSR 测试波形的局部。图中可见 u0SPI 的 SPI 时钟输出信号 CLKOUT01 的频率改变了两次，这说明 u0 的寄存器 CPSR 的值被成功设定为不同的数值。同时可看出在每种时钟下数据信号 TX0RX1 以及 TX1RX0 的

数据波形正常。

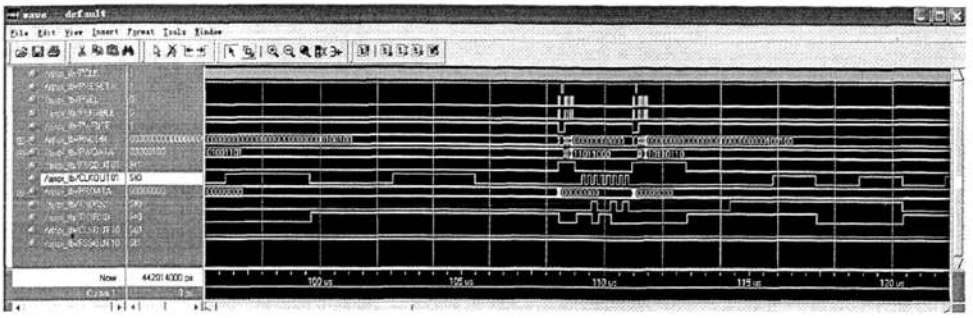

图 5.5 Modelsim 时序仿真图局部-test_CPSR

3. 图 5.6 至图 5.9 对应于 u0Master, u1Slave 时的四种时钟模式。图 5.6 对应于 CPOL=0, CPHA=0 的局部，图中可见空闲时钟为低电平；在桢信号 FSSOUT01 有效后的第一个 CLKOUT01 时钟沿开始数据传输。

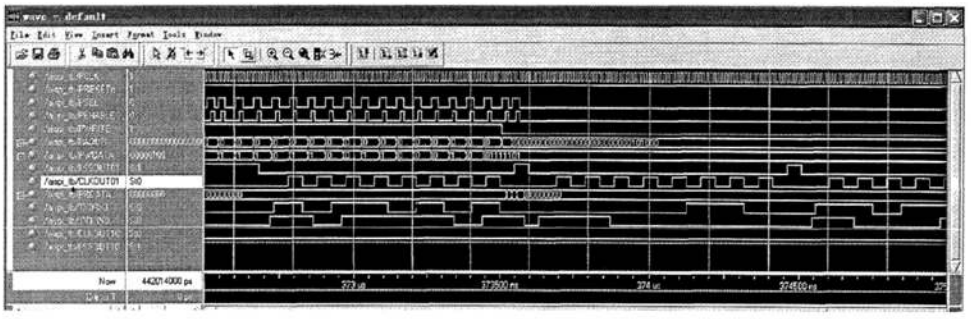

图 5.6 时钟模式 1 CPOL=0，CPHA= 0

图 5.7 对应于 CPOL=1, CPHA=0 的局部，可见空闲时钟为高电平；在桢信号有效后的第一个 CLKOUT01 时钟沿开始数据传输。

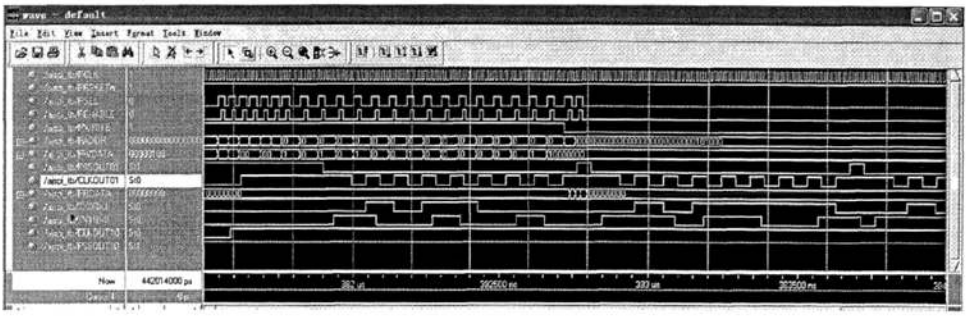

图 5.7 时钟模式 2 CPOL=1，CPHA= 0

图 5.8 对应于 CPOL=0, CPHA=1 的局部，图中可见空闲时钟为低电平；在桢信号 FSSOUT 有效后的第二个 CLKOUT01 时钟沿开始数据传输。

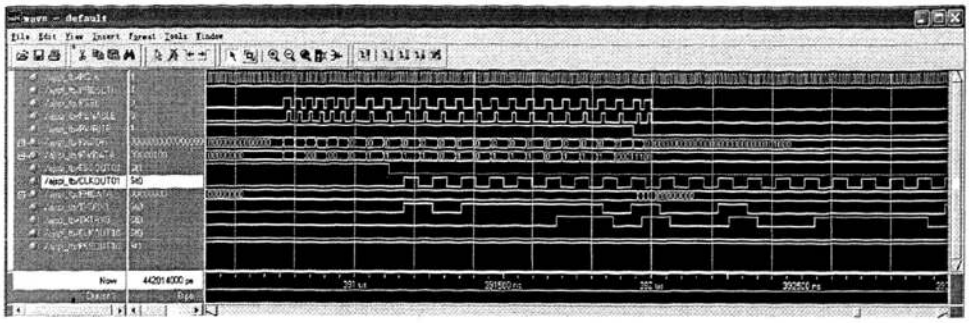

图 5.8 时钟模式 3 CPOL=0，CPHA= 1

图 5.9 对应于 CPOL=1, CPHA=1 的局部，图中可见空闲时钟为高电平；在桢信号 FSSOUT 有效后的第二个 CLKOUT01 时钟沿开始数据传输。

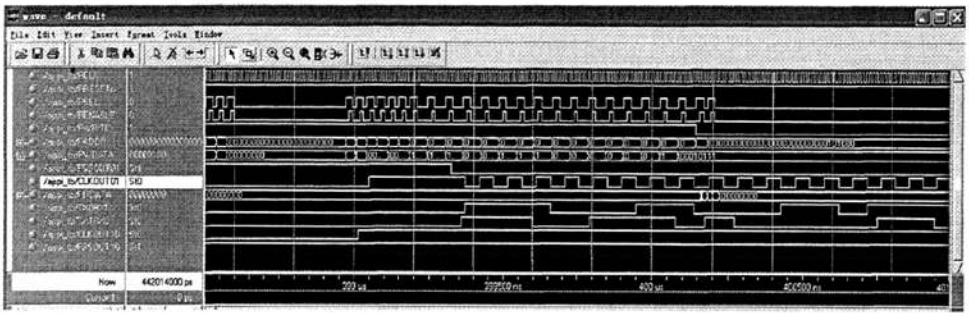

图 5.9 时钟模式 4 CPOL=1，CPHA= 1

4. 图 5.10 对应于 Master/Slave 切换过程。

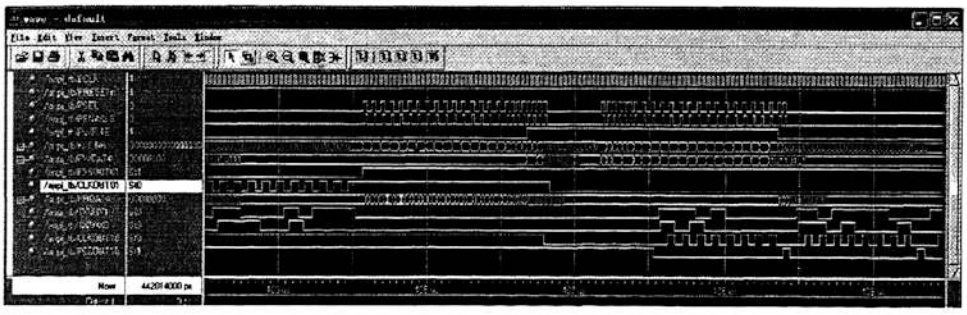

图 5.10 Modelsim 时序仿真图局部-Master/Slave 切换

波形左侧 CLKOUT01 有输出，而 CLKOUT10 电平不变。SPI 协议规定主模块负责输出时钟信号，说明此时的配置是 u0Master, u1Slave。可看出在 CLKOUT01 的控制下，数据信号 TX0RX1 和 TX1RX0 波形正常，每个周期传输一位数据。此外从 404.5us 到 405.5us 之间 CLKOUT01 经历了 10 个周期，验证了本 SPI 模块的数据传输时钟频率为 10Mhz 左右。经过一系列 APB 总线读写操作后，波形右侧 CLKOUT01 电平不变，而 CLKOUT10 产生规律的时钟波形，

说明此时配置已经成功设定为u0Slave, u1Master。此时同样可以正确地收发数据。此外，图中可以看出，Master-Slave 功能切换发生在约 407us 处，和前述 Modelsim 控制台的输出结果相符。

5.4 本章小结

本章使用业界通用的仿真软件 Modelsim 和 QuartusII 对本 IP 核的 Verilog HDL 代码设计进行了功能仿真和 FPGA 时序仿真。测试了模块重启、更换不同时钟分频频率、在所有四种传输时序下以及 Master 和 Slave 模式下的数据传输，结果表明模块的寄存器值和读写数据值均符合预期，仿真顺利通过。此外，通过采用随机数法覆盖了更多测试情况，充分验证了 IP 核的设计功能。

结论

SPI 接口技术是一种高速、全双工、同步的通信总线，并且连线简单，有利于节省 PCB 空间。现在越来越多的芯片集成了这种通信协议。本项目设计可复用的实现 SPI 协议的 IP 核，这符合 SOC 设计技术发展的方向，具有重要的实用价值。选择 AMBA APB 总线作为本 IP 核接口，因其具有技术简单，功耗低，实用性强等优点，广泛应用于嵌入式系统中。

设计实现了基于 AMBA 总线的 SPI 协议 IP 核，根据设计功能划分子模块，分析内部互连信号，给出了完整的模块结构图及详细端口连接和寄存器设置。阐述了 IP 核整体的设计思路，进一步以时钟分频模块、APB 接口模块、收发逻辑控制模块等关键子模块为例介绍了 Verilog HDL 代码设计过程中的几种典型问题与其解决过程。

使用 Modelsim 和 Quartus 仿真，成功进行 FPGA 时序仿真，验证了设计功能。本 IP 核使用 APB 总线和 APB Host 通信，可通过内部寄存器设定不同的传输速率，能适应所有四种 SPI 时序模式，且可配置为主模式或从模式。其功能完备，使用方便灵活，具备作为 IP 核进行 SOC 设计或 FPGA 开发的工程价值。

参考文献

[1] 李兵 骆丽，SOC 技术现状及其挑战[j],今日电子，2005.8, 40-41

[2] 赖祥宁,SOC 技术及国内发展现状[j],世界电子元器件,2002.8, 40-41

[3] 王莹 中国 IC / SOC 设计业的现状及发展思路[j], 电子设计自动化，2002.1.A 35-39

[4] 张繁，刘笃仁. SOC 设计面对的技术挑战[j], 今日电子 2004.12 85-87

[5] 汪健，刘小淮. 嵌入式 SOC 片上总线技术的研究[j]. 集成电路通讯， 2008.9， 26(3)

[6] 周砚江，顾焕峰，冯佳良.基于 SPI 的快速多通道数据采集和数字滤波方法及应用研究[J].电子测量与仪器学报,2008,22(3):100-104.

[7] 陈肖华，任德志，徐丽萍，等.基于增强型 SPI 接口的大容量 Flash 扩展实现[J].国外电子测量技术,2006,25(6):15-18.

[8] 何最红，张辉.TMS320C5402 与单片机 SPI 串口通信的实现方法[J].国外电子测量技术,2005,24(2):4-6.

[9] 方承志,李元,李明栋.用于嵌入式系统多路 SPI Master 接口设计[J].电子测量技术,2004(2):3-4.

[10] 李跃峰.基于 Verilog+HDL 的 SPI 可复用 IP 核的设计与实现[D].成都,西南交通大学,2008.

[11] 高谷刚,罗春.可复用 SPI 模块 IP 核的设计与验证[J].单片机与嵌入式系统应用，2004(11):5-8.

[12] 蒋彭龙，宋征宇，刘志华，等.AMBA 总线技术在片上系统(SOC)集成设计中的应用[J].航天控制，2008,24(5):37-39.

[13] 杨震，文爱萍 SOC 的关键技术和设计方法[j]. 微电子技术，2001.10, 25(5)

[14] 魏少军 SOC 设计方法学[j]. 电子产品世界，2001.5A:37-38

[15] 王卉，王小军 集成电路设计中的 IP 设计与集成方法[j]. 2006.3, 29(1):223-226

[16] 廖志国,面向 SOC 的 IP 核设计及软硬件协同验证研究[D]，2002,，南京大学

[17] JanM.Rabaey，数字集成电路设计透视(影印版)[M]。清华大学出版社，1995:18-31

[18] 汪健，刘小淮. 嵌入式 SoC 片上总线技术的研究[j]. 集成电路通讯， 2008.9， 26(3)：6-13.

[19] 马秦生，魏翠，孙力军，秦鸣，曹阳. 嵌入式 SoC 息线分析与研究[j]. 中国集成电路，2007.3，94：45-49.

[20] 唐杉等，数字 IC 设计:方法、技巧与实践[M].机械工业出版社，2006:123~129

[21] 李争，李范鸣，陈捷，庄良. 基于 Wishbone SoC 总线接口的高性能 SDRAM 控制器[J]. 科学技术与工程，2008，8 (12)：3342-3345.

[22] 黄荣志. AMBA 总线的在集成电路中的研究[j].微计算机信息，2008，Vol.23 No.5-2：

71-75.

[23] 李春红，高建华. AMBA 总线及其应用[j].计算机工程与设计，2008，Vol.28 No.4：766-769.

[24] 章彪，姜良华，肖剑，张沫. ARM 体系以及 AMBA 总线分析[j]. 计算机工程，2007，Vol.23 No.11-3：268-269.

[25] 邢翠芳，赵广利，吕海燕. SPI 串行总线接口及其实现[j]. 自动化与仪器仪表，2008，Vol.33：175-177.

[26] 杨晓江，王继成.用于嵌入式系统多路 SPI Master 接口设计[j]. 计算机工程与应用，2003，26：62-66.

[27] Diana M. Selfa，Maya Carrillo，Ma.The Jupiter-Io interaction as a model for star-planet-interaction (SPI)[j]. Mobile Technology，Applications and Systems，2007.

[28] AMBA™ Specification(Rev 2.0), ARM, 1999

[29] 牛风举. 刘元成. 朱明程基于口复用的数字 IC 设计技术[M]. 电子工业出版社，2003

[30] 徐光辉. 程东旭黄如. 基于 FGPA 的嵌入式开发与应用[M]. 电子工业出版社，2006.

[31] Sierra Circuit Design,Inc. Datasheet: Serial Peripheral Interface(SPI) User's Manual, 2000

[32] 王二萍，高速可复用 SPI 总线的设计与 Verilog+HDL 实现[D]，河南大学，2004.

[33] User's Manual S3C2410A–200MHz & 266MHz 32-Bit RISC Microprocessor Revision 1.0, Samsung Electronics, 2004,3.

[34] 夏宇闻, Verilog 数字系统设计教程[M].北京航空航天大学出版社，2003.

致 谢

在尊敬的导师易波教授的悉心指导下，本文终于完成了。从选题、方案选择到实现验证的整个过程中，易老师以他渊博的知识和和丰富的经验对我进行了精心的指导，常给我很大的启发。在研究生的学习期间，易老师不仅在学业上给我精心指导，还对我自己的设想和计划给于了最大的包容和支持。在此论文完成之际，谨向易老师表达我最诚挚的谢意。

在微电子实验室的学习和生活中，所有的老师和同学们都给了我很大的帮助，在此对他们表示衷心的感谢。实验室就像是一个大家庭，让每个身处其中的人感到温暖。衷心祝愿实验室的每一位老师和同学，在以后的工作和生活中都能幸福美满。

感谢我的父母。你们养育了我，这些年来一直默默的支持我。你们辛苦了。

在读期间发表的学术论文与取得的研究成果

[1] 《基于 AMBA 总线的 SPI 协议 IP 核的设计与实现》.赵杰，曹凡，江殿亮.《电子测量技术》已录用.

[2] 《微弱电流测量电路的设计和仿真》. 赵杰，曹凡，李翔宇.《电子技术》已录用.