北京工业大学

毕业设计（论文）任务书

IEEE 802.11g

题目 无线网卡 Baseband 的FPGA验证和接口设计

专业 自动化 **学号** 01020504 **姓名** 宋 威

主要内容、基本要求、主要参考资料等：

主要内容：

Baseband 为802.11g协议中物理子层（PHY）的实现，最终由硬件电路实现，并集成在一片数字芯片上。数字芯片的设计周期长，而利用FPGA实现其逻辑功能进行功能验证可以缩短设计周期。所以在Baseband的代码生成ASIC电路并流片之前，需要用FPGA对这一部分的逻辑进行充分的验证。整个课题涉及到Verilog和VHDL设计，第三方代码的应用，Baseband和上层（MAC）的接口设计，代码调试和FPGA验证等内容。

基本要求：

1. 编写ARM APB总线和Baseband的接口；
2. 替换ASIC DesignWare为FPGA的IP core以便FPGA实现；
3. 编写Baseband和总体软件测试环境DummyMac的接口实现软件测试；
4. 用FPGA实现Baseband的硬件验证。

主要参考资料：

IEEE std 802.11 Wireless LAN Medium Access Control(MAC) and Physical Layer(PHY) specifications

802.11 Wireless Networks: The Definitive Guide, Mattbew S.Gast, O'Reilly

《Verilog HDL 高级数字设计》，[美]Micheal D.Ciletti,电子工业出版社

《VHDL 设计：表示和综合》，[美]Jams R.Armstrong, F. Gail Gray，机械工业出版社

ModelSim(Advanced Verification and Debugging) SE User's manual, Mentor Graphics Crop.

完成期限： 2005. 6

指导教师签章： 方穗明

专业负责人签章：

2005 年 3 月 日

摘　要

无线局域网（WLAN）是近年来新兴的网络接入方式。其宽带和可无线接入的特性，满足了人们随时随地高速上网的需求，正日益受到用户的青睐。802.11g 是 IEEE 802 标准化委员会于 2003 年最新批准的 WLAN 物理层和媒体访问子层协议。支持该协议的网卡能在 2.4GHz 频段上提供最高 54Mbps 的数据传输率，并完全和上一代 802.11b 网卡兼容，势必成为下一代无线网卡标准的主流。

本文主要内容围绕 802.11g 网卡的基带设计和验证展开。

论文的前半部分，叙述了 802.11g 物理层当中使用 OFDM 调制方式的基带的结构，以及各个模块的原理与功能。详细分析了基带与其它模块之间的接口设计。论文的后半部分则针对基带的验证，分析了 FPGA 验证的流程与其中应当注意的种种问题。

整个论文建立在实际工程的设计与调试的基础上，详细分析了接口设计当中的异步问题、异常处理、状态机构成、APB 从设备的设计、ASIC 设计与 FPGA 设计的转换、硬件代码的综合、FPGA 的布局布线等等实际问题。

希望本文能通过对实际问题的分析，增进读者对 802.11g 基带、接口设计和 FPGA 验证这些实际问题的理解。

关键词: WLAN; 802.11g; FPGA 验证; OFDM; 基带

ABSTRACT

Wireless local area network (WLAN) is a novel method for network connection. Since its capability of providing wider broadband and wireless connection to network, it is been accepted and preferred by more users day by day. The 802.11g protocol is newly published by IEEE 802 protocol group, which specifies the behaviors of MAC sub-layer and PHY layer. The network cards which support this protocol can provide a data transmit speed up to 54Mbps on 2.4GHz ISM band. Furthermore, it is absolutely compatible with the 802.11b network card, which is popular today. It is convinced that, the 802.11g network cards will take place of other cards, becomes the mainstream network cards of the next generation.

My thesis includes two main sections, the design of 802.11g baseband and FPGA verification.

In the first section, the thesis describes the architecture of the whole baseband, with the functions and theories of each block in it, and detailedly analyzes the interfaces between baseband and other modules. After this section, the thesis describes the tool flow and the problems I encountered, specified to FPGA verification process.

Based on design and debugging of an actual project, my thesis puts emphasis on a serial of practical issues, such as the problems of asynchronous interfaces, the process of exceptions, the structure of state machine, the design of APB slavery device, block conversions between ASIC and FPGA, the synthesis flow of RTL code, the place and route of FPGA, etc.

With another expectation, I hope this thesis can improve my readers' understanding of 802.11g baseband, design of interface and FPGA verification.

Keywords: WLAN; 802.11g; FPGA verification; OFDM; Baseband

目 录

1 绪论

随着网络需求的日益增加，人们越来越希望能够不受空间的约束，随时随地上网，并获得一定的网络传输速率。以往的有线局域网虽能提供高速的网络接入服务，但不能摆脱定点接入形式的束缚。而无线上网方式往往面临着低上网速率与高服务费用的困扰。

无线局域网络接入方式，便是在这种形势下越来越受到用户的关注。通过 802.11 系列网卡和相应接入点（AP）的配套使用，用户将能够很容易地获得宽带局域网的功能，既可以在接入点覆盖范围内随时随地上网，同时也能获得较高的网络传输速率。而 802.11g 无线网卡正是当前 802.11 系列当中最新也是功能最强大的一款，正处于产品的研发和推广阶段。

1.1 802.11 无线网卡的背景及其分类

802.11 系列网卡是使用 802.11 媒体访问子层（MAC）和物理层（PHY）协议网卡的简称。802.11 实际上是 IEEE 的一个协议簇的名称。

IEEE 802 标准化委员会于 1990 年成立 802.11 WLAN标准化工作小组，主要负责针对于无线局域网（WLAN）的媒体访问控制子层、物理层以及相关协议的起草工作[1]。可见，802.11 实际上只是MAC层和PHY层的标准，处于OSI分层体系中的最低两层。

802.11 标准化工作小组于 1997 年审定并通过了 802.11 媒体访问控制和物理层协议。其中的物理层协议规定了工作在 2.4~2.485GHz 频段上采用跳频扩频（FHSS）调制方式和直接序列扩频（DSSS）调制方式的射频（RF）传输方法，和工作在 850~950nm 段的红外线传输方法。

其中，FHSS 采用 2~4 电平高斯频移键控（GFSK）调制技术，可在 79 个信道上，提供 22 个跳频序列和 1Mbps 的传输速率。DSSS 方式采用差分二进制相移键控（DBPSK）和差分正交相移键控（DQPSK）调制方式，提供 1Mbps 和 2Mbps 的传输速率。由于少有设备实现使用红外线的传输方式，在此不做讨论。

802.11 协议最高只能提供 2Mbps 的传输速率，显然不能满足宽带上网的要求。所以，802.11 标准化工作小组于 1999 年通过了 802.11b 和 802.11a 物理层协议（802.11a 于 2000 年修订）。并且，802.11 物理层协议很快被 802.11b 物理层协议所取代。

802.11b 物理层协议规定了工作在 2.4~2.4835GHz 频段上采用补偿编码键控（CCK）调制方式的传输方法。从调制原理上来说，802.11b 方式是 802.11 物理层 DSSS 方式的扩展，在直接序列扩频上采用更高的编码率，提供高达 11Mbps 的传输速率，也被称为高速直接序列扩频（HR-DSSS）。

802.11a 物理层协议规定了工作在 5.15~5.825GHz 频段上采用正交频分复用（OFDM）调制方式的传输方法。利用在 5GHz 频段采用 OFDM 方式，802.11a 可提供最高 54Mbps 的传输速率。

802.11a和 802.11b同样在 1999 年由 802.11 标准化工作小组审定通过，但事实上大多数

产品都采用 802.11b协议方式。在过去几年，802.11b无疑已经成为市场的主流。虽然从速度上说，802.11a方式可支持高达 54Mbps的传输速率，但是 802.11a需要工作在 5GHz频段，并且由于 5GHz的射频和 2.4GHz的射频设备不能兼容，导致 802.11a和 802.11b难以兼容。而且，更高的工作频率需要更高的设备制造费用，同时由于高频信号的衰减相应会比低频高，导致 802.11a的网络接入点（AP）分布相应要比 802.11b密集，造成组网费用加大[2]。更进一步，由于 2.4GHz频段属于世界范围内的ISM频段，不需要任何注册费用，而 5GHz频段仅在美国属于ISM频段范围，而在其他国家，该频段并不一定开放，而需要注册使用，致使符合 802.11a标准的网卡缺少在美国以外的兼容性。

鉴于以上的种种原因，从 2000 年至今的所有 802.11 系列无线网卡基本上都是基于 802.11b 协议的。

但是，802.11b 最高只能提供 11Mbps 的传输速率，还是不能满足高速宽带传输的要求。于是，802.11 标准化工作小组于 2003 年 7 月通过了 802.11g 物理层协议。

802.11g和 802.11b一样，都工作在 2.4GHz的ISM频段。但是 802.11g可在 2.4GHz频带上实现OFDM调制方式，因而能提供最大 54Mbps的传输率。从调制方式来说，802.11g支持CCK/CCK和OFDM/OFDM两种必需的调制方式与CCK/PBCC和CCK/OFDM两种可选的调制方式[3]。其中PBCC为包二进制卷积调制方式，为德州仪器公司（TI）提出的标准，被 802.11 标准化小组所接受。802.11g的每种调制方式“/”的前一种表示传输的帧前导（preamble）和信头（PLCP header）的调制方式，“/”的后一种代表数据的调制方式。其中CCK/CCK方式完全和 802.11b兼容，可提供 5.5Mbps和 11Mbps两种传输速率。OFDM/OFDM方式完全和 802.11a的调制方式相同，只是所用的信道频段不一样，能够提供 6Mbps到 54Mbps的多种传输速率。

表 1.1 给出了上文所述的几种协议的参数，从中可以看出 802.11 协议的发展历程。

表 1.1 802.11 协议比较

协议名称	调制方式	速率 (Mbps)	频带 (GHz)	最大传输距离(m)	信道数	可用带宽 (MHz)	通过时间
802.11	FHSS	1	2.4	100	22	83.5	1997
	DSSS	1,2			3		
802.11b	CCK	1,2,5.5,11	2.4	100	3	83.5	1999
802.11a	OFDM	6,9,12,18, 24,36,48,54	5	80	12	300	2000
802.11g	CCK/CCK	5.5,11	2.4	150	3	83.5	2003
	OFDM/OFDM	6,9,12,18, 24,36,48,54					
	CCK/PBCC (可选)	5.5,11,22,33					
	CCK/OFDM (可选)	6,9,12,18, 24,36,48,54					

从表 1.1 中可见，802.11g 标准能够完全兼容 802.11b 的主流产品，并且能够直接在 2.4GHz 频段实现和 5GHz 频段同样的高速率数据传输，从而既减少了重新部署网络所需要的成本，又提高了整个网络的性能。

本论文所对应的 802.11g 网卡为北京市嵌入式系统重点实验室的研发项目，由于以前

的项目已经完全覆盖 802.11b 网卡的设计，所以这一次 802.11g 网卡的开发主要针对于在 2.4GHz 上实现 OFDM 的 802.11a 兼容调制方式。论文所对应的基带（Baseband）也主要对应 OFDM 调制方式所需的基带。

1.2 FPGA 验证概述

FPGA，中文为现场可编程门阵列，由 Xilinx（赛灵思）公司于 1985 年首次推出。

FPGA 是一种可编程，且可以多次修改的大规模集成电路芯片。通过将编写的 Verilog 或 VHDL 代码编译并下载到 FPGA 当中，就可以实现所需要的逻辑功能，具有开发周期短，更改简便等多方面的优点。

FPGA 的可编程性，来源于它的半定制的内部逻辑。FPGA 内部一般含有可配置的连接矩阵、查表单元、控制逻辑和寄存器。通过设置内部元件的参数，可以很容易地实现组合逻辑、查表、时序逻辑和内存的种种功能。并且，随着半导体制造工艺的发展，单片 FPGA 所能支持的逻辑功能已经可以把一个中型规模的 SOC 设计放在一个芯片当中实现。所以，有些 SOC 的芯片已经不再通过 ASIC 的方法去完成，而直接采用 FPGA 的方法。

下面简单介绍 FPGA 在 ASIC 流程中的作用和它的优缺点。

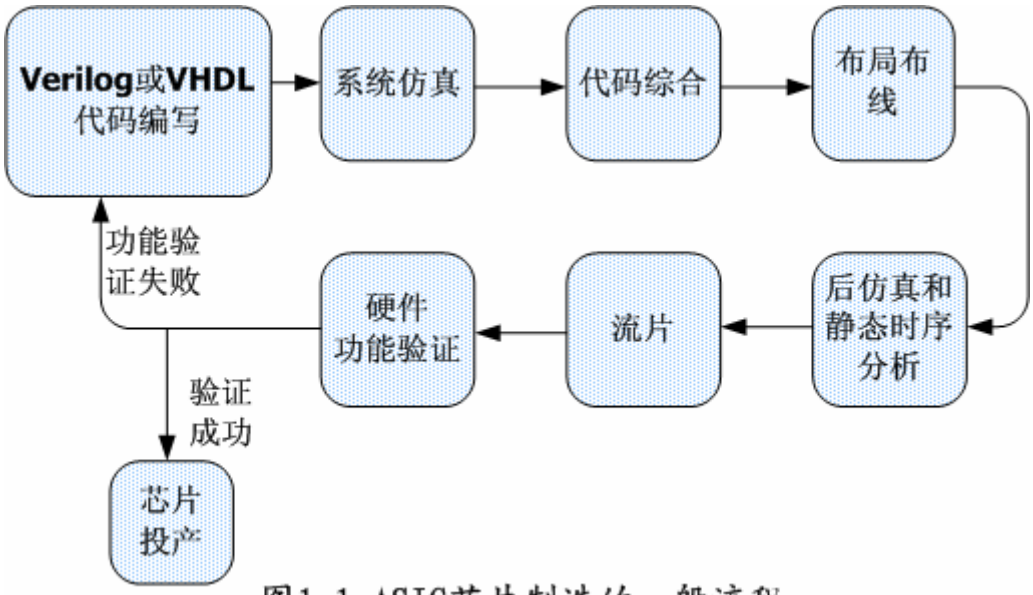


图1.1 ASIC芯片制造的一般流程

ASIC 实际上就是专用集成电路。在电视、电脑当中几乎所有的芯片都属于 ASIC 芯片的范畴。然而，像这样的成品芯片，需要非常复杂的制造流程，一般如图 1.1 所示。

流程当中的每一个步骤，一般都需要以月为时间单位计算，整个流程一般需要 1 年半到 2 年的时间。并且，在系统仿真阶段，如果不借助于 FPGA 的验证，那么只能是完全的软件仿真。软件仿真对跨时钟域、门控时钟、多周期路径、芯片外部接口、IP 接口等等情况并不能做出完全的仿真和验证。那么，经过这样的流程，最终流片时并不能保证硬件功能的正确性。所以很可能在流片完成后发现功能错误，需要重新修改代码重复整个流程。

并且，ASIC 芯片的流片是不可重复的。如果出现错误，整个芯片将会完全报废，并且整个芯片的制造时间会大大增加，将导致芯片成本提高，还有可能导致芯片错过最佳的上市时机。

而 FPGA 的功能实现，只需要将代码通过 FPGA 相应的软件平台转化为 FPGA 的下载文件，下载到 FPGA 芯片上并完成对 FPGA 的配置工作即可。这个周期只需要几天时间。并且 FPGA 能够重复下载，更改代码并不需要付出其他的成本。

更重要的是，此时的验证已经不是简单的软件仿真，而是实际电路的实现。在整个 ASIC 布局布线开始之前就能够实现对芯片功能的物理验证，将大大提高最后流片的成功率。并且，此时出现功能错误的代价仅仅是修改代码，重新进行 FPGA 验证，只需要几天的时间。

所以，现在一般的 ASIC 芯片，都会在系统验证阶段，不仅在软件平台完成基本逻辑的验证，也需要将整个设计在 FPGA 上实现，以求获得最小的重复流片概率。

当然，FPGA 也不是十全十美的。从硬件上说，由于 FPGA 本身属于半定制电路的限制，其运行速度要比 ASIC 慢，且单片成本要比 ASIC 大出许多。如果大批量生产，ASIC 的成本会比 FPGA 要低很多。这也是在现在的半导体器件制造业当中，FPGA 一般只作为验证工具而不直接作为成品出售的原因。

另外，FPGA 的可编程逻辑是由其内部可设定参数的半定制单元提供的，因而其实现方式和 ASIC 的并不完全一致。在一些情况下，需要对 FPGA 验证作专门的代码定制，造成 FPGA 验证的代码版本和实际 ASIC 流片的版本产生区别，也有可能导致流片失败。

更糟糕的是，FPGA 是完全的数字电路，对模拟电路无能为力。如果 ASIC 芯片当中出现模拟电路的成份，是不能用 FPGA 验证的。因此现在数模混合电路的验证还处于完全的软件验证阶段。

1.3 本文结构

从本文的题目也可看出，论文主要讨论两方面的内容，802.11g 无线网卡 Baseband 的接口设计和 Baseband 的 FPGA 验证。

在整个 802.11g Baseband 的设计和验证的项目当中，我主要负责 Baseband 和外部的接口设计，整个 Baseband 的 RTL 代码 ASIC 版本到 FPGA 版本的替换，代码综合和布局布线。因此，对于 Baseband 的结构我只能做一个原理上的介绍，而重点主要放在接口设计和 FPGA 验证上。

第二章将从整体上讲述 Baseband 的结构，每一个功能部分的大体原理。这一章主要是对 802.11g 无线网卡 Baseband 当中 OFDM 方式调制解调的各个功能模块的原理作说明，以便增加对整个系统的理解。

第三章为 Baseband 当中几个接口模块的设计说明。其中和 MAC 的接口主要负责在发送时将 MAC 的控制信息打包封装到信头当中去，而接收时除去信头的信息，仅仅将数据传给 MAC 层，并实现 Baseband 和 MAC 的数据通路连接，和简单的控制功能。APB 的接口主要负责将 Baseband 作为 ARM APB 总线上的一个从设备，提供从 APB 总线读写 Baseband 参数的功能。此外，和 RF 芯片的接口，仅仅是简单的连接，也做一个介绍。

第四章则讲述另外一个重要问题——FPGA 的验证。

该章首先介绍了 FPGA 验证的工具流程，以便对整个 FPGA 验证，从 RTL 代码到最后 FPGA 的实现有一个整体的理解。然后分三节，着重讨论了 IP 替换、FPGA 的代码综合和布局布线这三个主要步骤，并总结了在这几个步骤当中需要注意的种种问题。

2 802.11g Baseband 的结构分析

2.1 802.11g 无线网卡的总体结构

2.1.1 无线网卡在计算机网络结构中的位置

一旦涉及到网络，必然从 OSI 七层参考模型开始。网卡，作为计算机连接互联网的设备，却并不能完成参考模型当中的所有功能，有相当一部分的功能是要通过计算机软件、操作系统和驱动程序一起协调工作，才能完成。

7.应用层
6.表示层
5.会话层
4.传输层
3.网络层
2.数据链路层
1.物理层

图2.1 OSI七层参考模型

一般来说，七层模型中的上四层由计算机系统和驱动程序实现。而网络设备只需要负责实现下三层所对应的逻辑功能。

其中，物理层主要负责对应不同的传输信道的不同调制方式。传输信道可分为很多种，最普通的，以太网可以采用差分双绞线传输，也可以采用同轴电缆；Cable TV 网络则会采用闭路电视系统同轴电缆的高频段；交换局之间会采用光纤作为数据传输媒介；而无线局域网则会采用 2.4GHz 或者 5GHz 的无线频段作为传输信道。

所以，物理层需要对应这些不同的媒介，采用不同的调制方式以适应数据传输的需要。实际上，对于不同的信道和调制方式，必须有不同的物理层协议规定。

数据链路层主要负责节点如何将一帧的数据从信道的一端传到另一端。如果网络出现了冲突，节点如何还能在有冲突的情况下正常地建立通讯，无差错地将帧传递到对方节点。

而网络层要更复杂一些。计算机之间的连接并不总是直连的，在大多数的情况下，数据包会通过其他的节点存储转发之后，才能到达目的站点。那么节点如何才能够知道，甚至保证数据安全无错地到达远方节点呢？这便涉及到路由选择，拥塞控制等等算法问题，需要由网络层来完成[4]。

从功能划分来看，网卡只需要负责实现物理层、数据链路层和一部分网络层的功能。对应这些功能的实现，IEEE802 标准化委员会给出了一整套的协议体系。如图 2.2 所示[5]。

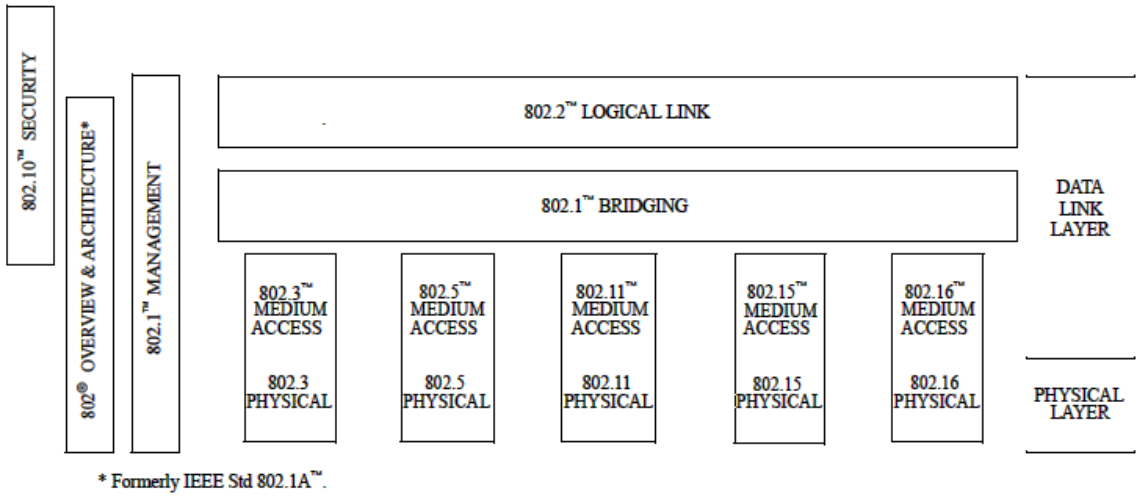


图2.2 IEEE 802 协议体系

从图中可见，802 协议体系将数据链路层进一步划分，分为逻辑链路子层（Logic Link）和媒体访问控制子层（MAC）。同时，为每一个单独的物理层协议定义一个与之配套的 MAC 层，而所有的物理层共享同一个逻辑链路控制（LLC）子层协议，并由一个专门的网桥（Bridge）协议来负责不同 MAC 子层之间的数据交互。

这样做有几点好处：

1、每一种不同的物理层对应一个专门的 MAC 层，可以实现对应不同物理特性信道的不同控制方法。比如在以太网上我们使用带冲突检测的载波侦听多路复用协议（CSMA/CD），而在无线局域网我们使用带冲突避免的载波侦听多路复用协议（CSMA/CA）。

2、所有的 MAC 子层都对应统一的逻辑链路控制协议和网桥协议，有助于不同物理类型的网络之间进行互通。MAC 层的接口只需要和网桥协议的接口对应，就可以实现不同网络间的自由通讯。

3、统一的逻辑链路控制子层为网络层屏蔽了各种物理信道的特性区别。因而网络层可以完全不用考虑通讯具体使用何种物理信道，而只需要对网络层的路由算法等等进行设计。

在图 2.2 当中的各种协议列表如下：

IEEE802.2　　逻辑链路控制子层协议

IEEE802.1　　网桥协议

IEEE802.3　　带冲突检测的载波侦听多路访问 MAC 子层和物理层协议（以太网为 802.3 协议中的一种）

IEEE802.5　　令牌环网 MAC 子层和物理层协议

IEEE802.11　　无线局域网 MAC 子层和物理层协议

IEEE802.15　　个人区域网 MAC 子层和物理层协议（蓝牙协议）

IEEE802.16　　无线城域网 MAC 子层和物理层协议

本论文所述的 802.11g 无线网卡实际上是符合 802.11 无线局域网 MAC 子层和物理层协议的。正如论文绪论中所提及，802.11 协议本身也包含不同的子协议，共有 802.11MAC

子层协议、802.11 物理层协议、802.11a 物理层协议、802.11b 物理层协议、802.11g 物理层协议和 802.11i 安全协议。它们的协议体系如图 2.3 所示：

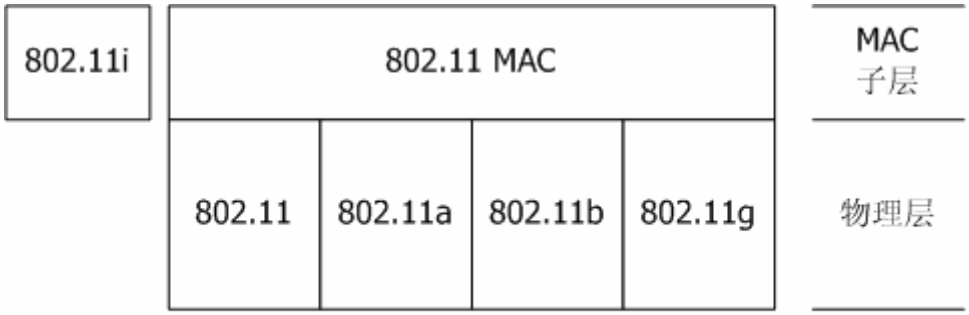


图2.3 802.11协议体系

802.11g 协议是 802.11 协议体系当中对物理层的子规范。对应物理层使用 2.4GHz ISM 频段，采用 OFDM、CCK、PBCC 等调制方式，支持最高 54Mbps 的传输速率。

以上是从计算机网络体系说明 802.11g 协议所处的位置。从计算机结构和功能划分，802.11g 无线网卡是计算机 PCI 总线或者 USB 总线上的子设备。网卡通过计算机总线和系统建立联系。

实际上，只有网卡上的芯片能够支持多种总线接口，才能实现只需要更改网卡的外部电路，就能支持不同的总线结构。所以，论文所述的 Baseband 芯片实际上能够支持 PCI、USB1.1 和 Cardbus 总线结构。

2.1.2 无线网卡的基本结构

正如上一小节所述，无线网卡需要完成物理层和 MAC 层的逻辑功能。但是这些逻辑功能并不是全部由硬件完成的，而是一些由软件完成，一些由硬件电路实现。图 2.4 给出了网卡最基本的结构框架。

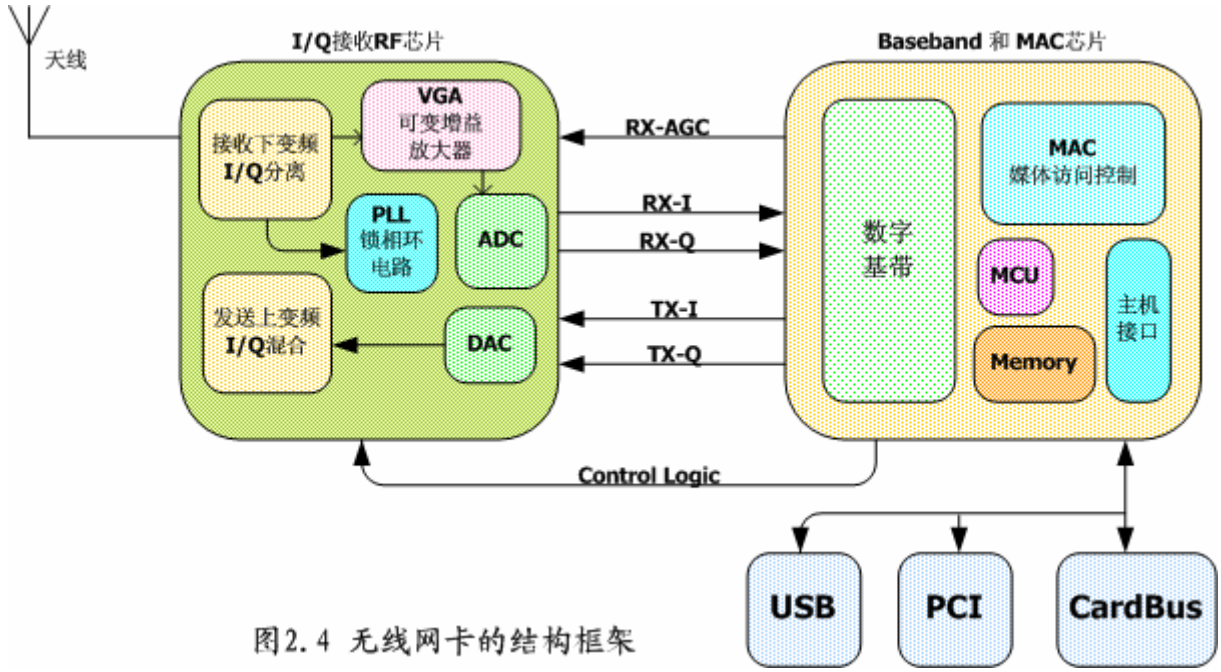


图2.4 无线网卡的结构框架

无线网卡主要由两块芯片构成，RF 射频芯片和基带与 MAC（Baseband&MAC）芯片。

RF 芯片主要工作就是提供数字基带信号和高频的模拟信号之间转换。在发送阶段，芯片内的 DA 单元将基带的数字信号转化为相应的模拟电平，由上变频单元将模拟信号搬移到高频并发送。在接收阶段，首先经过下变频，将高频信号转化为低频信号。通过锁相环电路（PLL）获得恢复后的时钟，并由 AD 单元将模拟信号转化为数字基带信号传给基带芯片。其中的 VGA 为可变增益放大器。它通过检测接收信号的强度，选择不同的天线和控制对低频模拟信号放大系数实现控制数字信号的变化范围的功能。

而基带和 MAC 芯片主要完成数字基带转换和 MAC 层的控制逻辑。可分为基带、MAC 和主机接口等几个模块。

基带（Baseband）为芯片当中最重要的模块。由于这一部分的时钟频率较高，并且逻辑复杂，但功能固定，全部由硬件电路搭建。主要负责对从 RF 芯片接收的数字信号进行解调和纠错，并对 MAC 层发送的信号进行调制的工作。

在 MAC 模块当中，和基带联系紧密的部分由硬件电路实现。而访问控制、冲突响应等等的功能，由于并不需要具有高度的实时性，并且灵活多变，由软件代码实现。所以，芯片当中嵌入了一个微处理器的硬核，并通过将软件代码存储在片上存储单元当中，实现了一个芯片级的嵌入式系统以完成相应 MAC 层的工作。

而主机接口单元，主要负责 MAC 和计算机系统之间的连接。通过主机接口，基带和 MAC 芯片能够成为计算机总线上的子设备，通过相应的数据总线、控制总线和中断系统实现和系统的对接。而主机接口单元就是对 MAC 的数据和计算机总线的数据进行格式转换，并使得 MAC 数据和逻辑符合相应总线的要求，能够自动识别。

2.2 Baseband 的结构分析

论文中对应的基带（Baseband）为对应 OFDM 方式的 802.11a 兼容基带，在绪论当中已经说明。对应使用 CCK 方式的 802.11b 基带已经在实验室以前的项目中完成。此次项目的主要目标是实现工作在 2.4GHz 频带，和 802.11a 兼容使用 OFDM 调制方式的基带设计。

2.2.1 Baseband 的整体结构

图 2.5 为整个基带模块的整体结构图。整个基带可分为 MAC 接口、扰码器、卷积编码器、维特比解码器、交织器、布局器、频域估计和校正单元、快速傅里叶变换单元、时域估计和校正单元、RF 接口、等等。

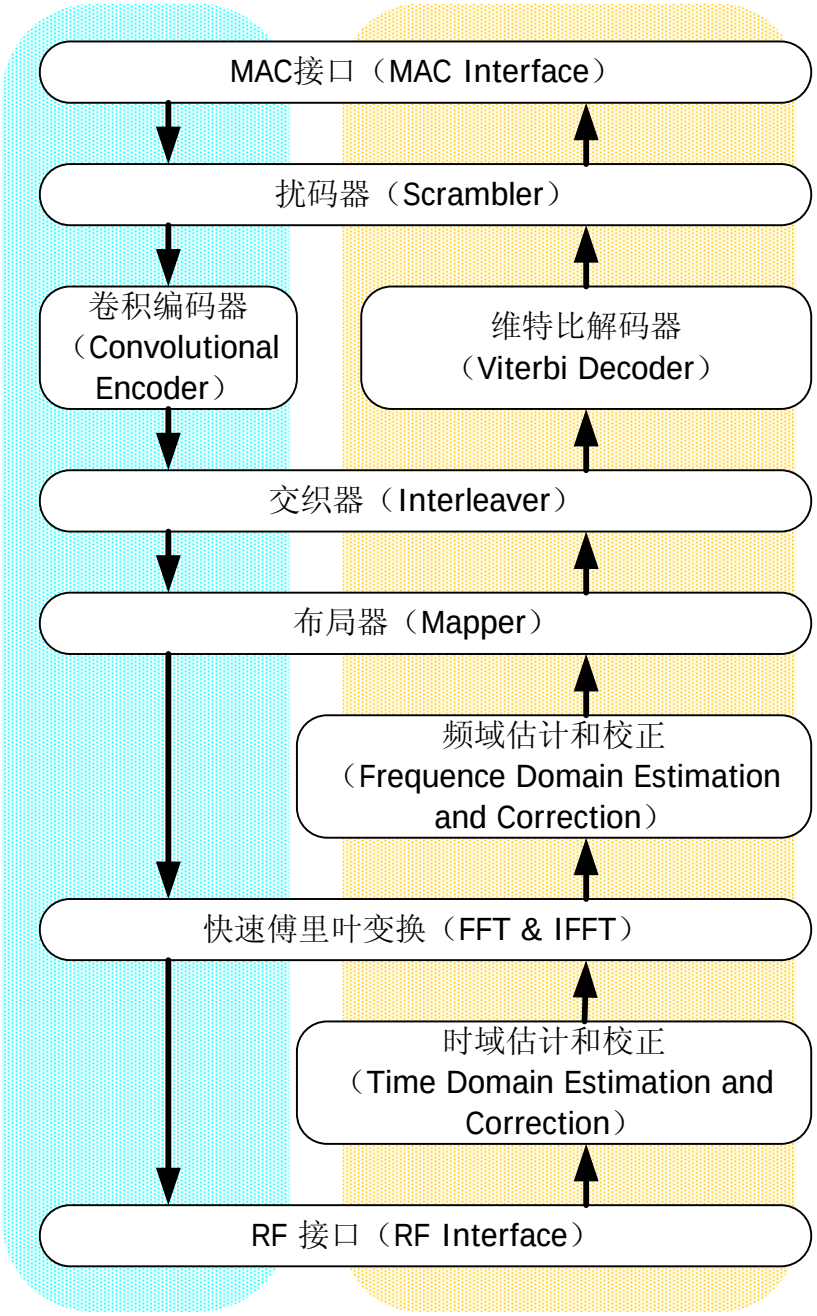


图2.5 Baseband的整体结构

其中，发送部分，根据数据流的方向，由 MAC 接口、扰码器、卷积编码器、交织器、布局器、快速傅里叶变换单元（IFFT）、RF 接口组成。接收方向，依次为 RF 接口、时域估计和校正单元、快速傅里叶变换单元（FFT）、频域估计和校正、布局器、交织器、维特比解码器、扰码器和 MAC 接口单元。

可以看出，除了两个估计和校正单元，从整个结构上，发送方向和接收方向是一一对应的。

整个结构当中最主要的两个单元是维特比解码器和快速傅里叶变换单元，它们所用的电路单元门数也相应最多。

2.2.2 OFDM 的基本原理

OFDM 的全称为正交频分复用。其主要思想是在整个工作频域内将信道划分成多个子信道，利用串并转换将数据流分配到多个子信道上。在每个子信道上使用独立的子载波分别调制，进行并行传输。这样，在每个子信道上采用窄带传输，每个子信道上的单符号传输时间相应增加，从而减轻了由无线信道的多径时延效应引起的码间干扰。

同时，OFDM和传统的频分复用方式有所不同。传统的频分复用方式，子信道之间是不重叠的，并且为了防止信道间的干扰，信道间还要加上保护间隔，进一步导致频谱的利用率下降。而OFDM方式，由于子信道之间能够保持正交性，子信道之间可以重叠而并不影响解调，因而大大提高了频谱的利用率[6,7]。

1. 正交性

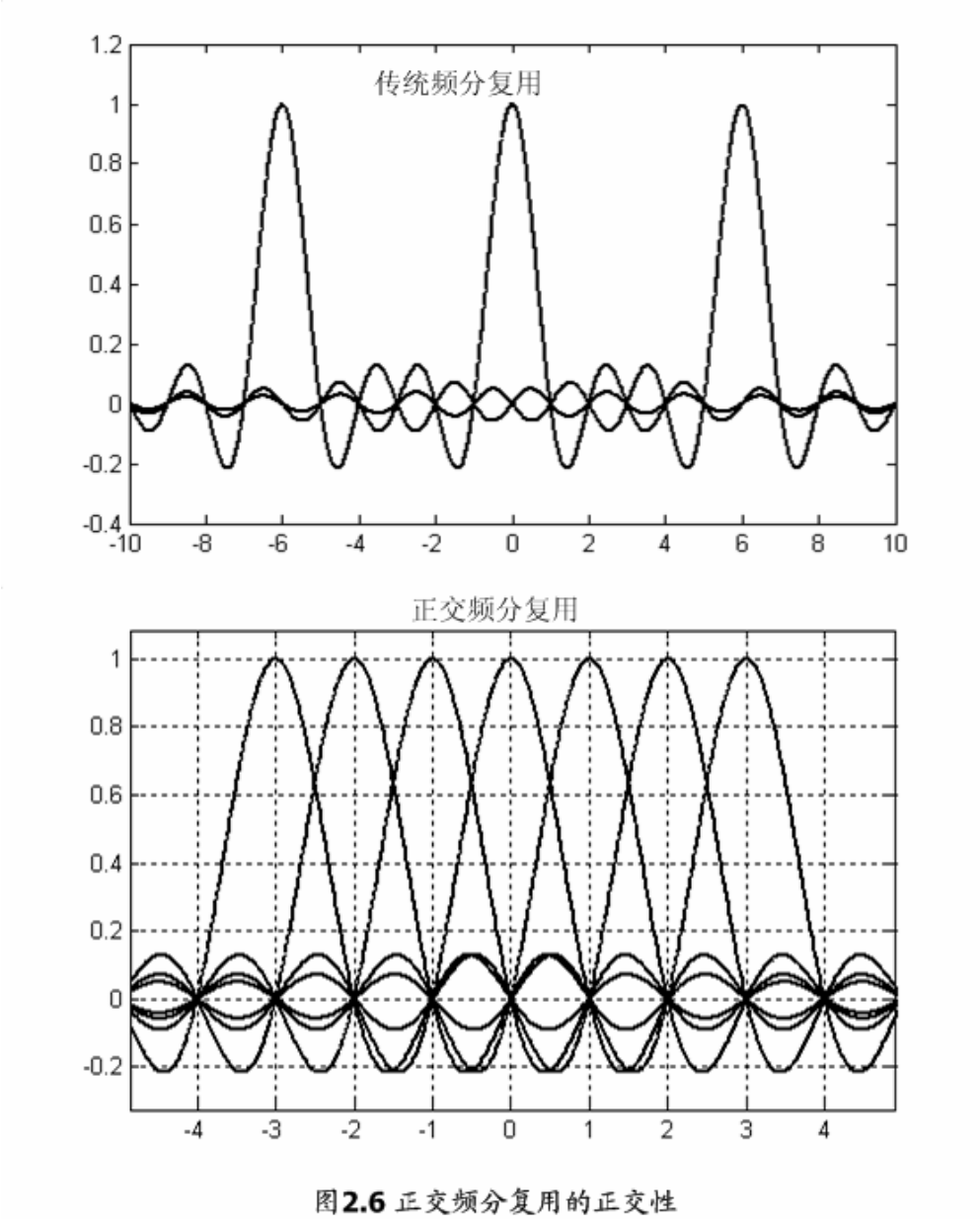


图2.6 正交频分复用的正交性

如果子信道之间的间隔恒定，且子信道中心频率之间的间隔 Δf 满足公式（2.1）

$$\Delta f = \frac{1}{T} = \frac{1}{N\Delta t} \tag{2.1}$$

其中 T 为一个符号周期，在符号周期内传输 N 个数，Δt 为时域的采样时间间隔。那么，子信道之间就能保持完全的正交特性。正如图 2.6 中所示，每个子载波虽然都和相邻的几个子载波的频谱发生交叠，但是由于正交特性，在相邻子载波的采样点都保持了过零特性，也就是对邻载波的中心频点无干扰，因而并不造成邻频干扰。大大提高了频谱的利用率。

2. 保护间隔和循环前缀

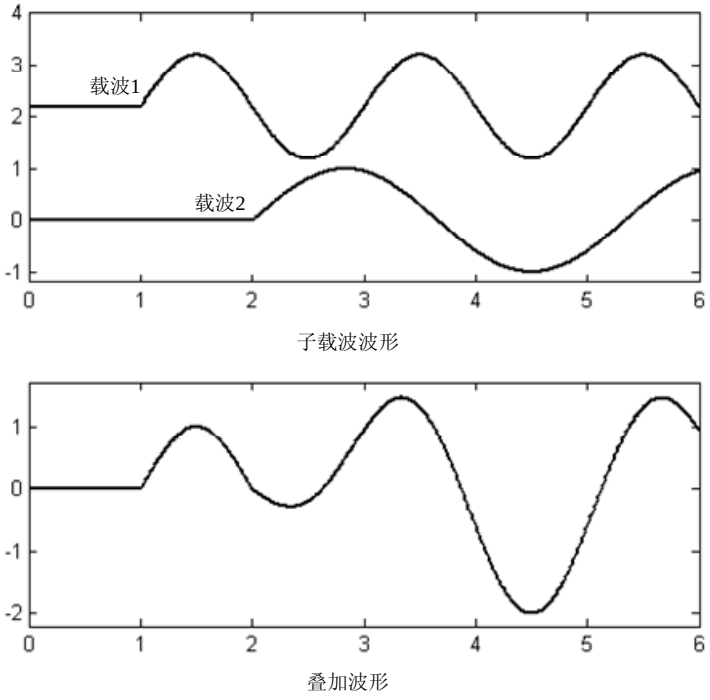
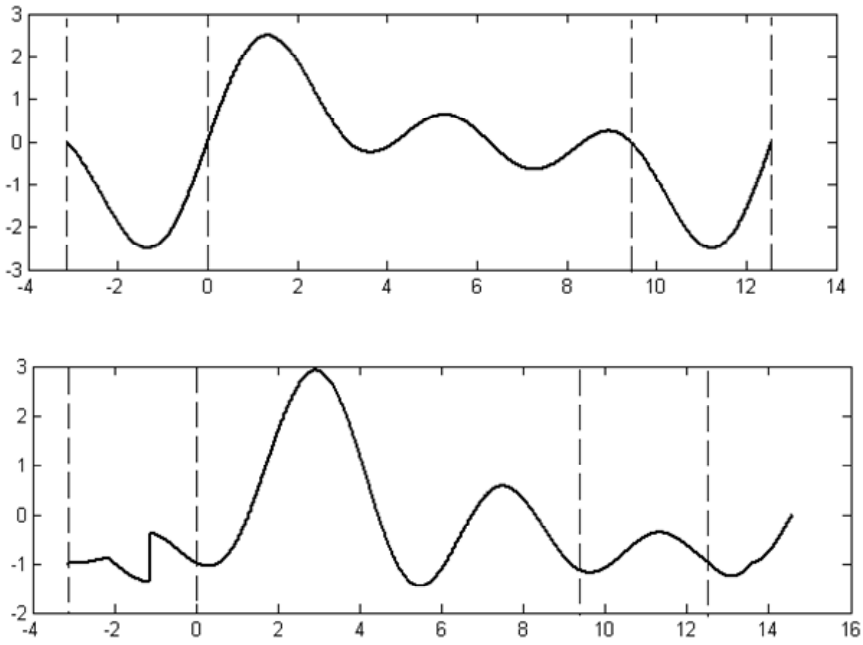


图2.7 延时造成的信道间干扰

为了进一步去除由于多径效应造成的码间干扰（ISI），可以在相邻符号之间加上保护间隔T_g，接收机在保护间隔内并不对信号进行解析。只要多径效应造成的延时时间小于保护间隔长度T_g，则不会对相邻的符号产生影响。

但是，如果在保护间隔不传输任何信号，多径效应会干扰子信道之间的正交性，如图 2.7。由于子载波 2 的信号发生了延时，前面一段的波形就只剩下了上面子载波的信号。同时在 2 微秒时刻，子载波 2 信号突然出现，使得叠加之后的波形相应位置出现了折线，给频带内带来高频谐波信号，而产生邻载波干扰。这样就破坏了子信道之间的正交性，而正交性是 OFDM 调制方式成立的基本条件。

所以在保护间隔内必须采用循环前缀的方式。即在保护间隔内，也继续传递符号的波形。



假设符号时间为 4π，保护间隔时间为 π。当没有发生延时的时候，在 $(-\pi,0)$ 之间的保护间隔，会传输符号传输时间 $(0,4\pi)$ 的最后的 $(3\pi,4\pi)$ 的波形。这样在整个保护间隔和符号传输时间内的波形都保持正交性和波形的连续性。

如果发生了延时，那么在保护间隔内的波形由于发生了延时，而产生了信道间的干扰，无延迟的正交被破坏且波形出现折线点。但是只要延时时间小于保护间隔的时间大小，那么这种干扰不会影响到接收机方向符号传输时间内的波形。当然，由于延时的作用，时域波形还是会产生区别，但是只要保持子信道、子载波之间的正交性，OFDM就能够正确地解调出发射信号[7]。

3. 快速傅里叶变换的使用

OFDM系统的一个重要特点，就是可以利用快速傅里叶变换进行调制和解调，从而大大简化系统实现的复杂度。一个OFDM符号由多个子载波的波形叠加而成，每一个子载波都可以用相移键控或者正交幅度调制的方法调制。如果设定N为子信道的个数，T代表一个OFDM符号的传输时间，$d_i(i=0,1,\cdots N-1)$ 是每一个子信道上的数据，f_c 是第 0 个子载波的中心频率，$rect(t)=1,|t|\le T/2$，那么，从 $t=t_s$ 时刻起，OFDM符号可表示为[7]：

$$s(t)=\begin{cases}\mathrm{Re}\left\{\sum_{i=0}^{N-1}d_i rect(t-t_s-\frac{T}{2})\exp\left[j2\pi(f_c+\frac{i}{T})(t-t_s)\right]\right\},(t_s\le t\le t_s+T)\\ 0,(t<t_s\cup t>t_s+T)\end{cases} \quad (2.2)$$

有延

图2.8 彻

如果忽略矩形函数，令 $t_s=0$，且考虑到 Baseband 以 T/N 的速率对接收到的波形进行

采样，那么，可以做出这样的变换：

令 $t=\frac{kT}{N},(k=0,1,\ldots N\text{-}1)$，代入到式（2.2）中得到式（2.3）

$$s_k=s(\frac{kT}{N})=\sum_{i=0}^{N-1}d_i\exp(j\frac{2\pi ik}{N}),(0\le k\le N-1) \quad (2.3)$$

可以看出，在调制时，实际上是在对 d_i 作离散傅里叶逆变换，可以使用 IFFT 算法。

在接收方向：

$$d_i=\sum_{k=0}^{N-1}s_k\exp(-j\frac{2\pi ik}{N}),(0\le i\le N-1) \quad (2.4)$$

实际上在对接收到的时域波形作离散傅里叶变换，可以用 FFT 算法实现。

综上所述，OFDM 有以下几个显著特点：

① 通过串并转换，将高速的数据流分配到并行传输的子信道上，增加单个码元的传输时间，因而能够很好地解决由于无线信道的多径效应造成的码间干扰问题。

② 利用正交技术，采用固定的子载波间隔达到子载波间的正交，增加了频谱的利用率。

③ 正交子载波本身的数学特性，决定了可以用快速傅里叶逆变换和变换来调制和解调信号，减少了实现的复杂度。

④ 采用循环前缀技术，进一步去除了多径效应对解调的影响。

⑤ 由于需要保持子载波间严格的正交关系，系统易受频率偏差的影响。所以当出现多普勒频移或者发送和接收机之间没有完全同步的时候，子载波间的正交性很容易被破坏，需要对信号的频域特性进行估计和补偿。

⑥ 由于频域数据的随机性，导致发出的时域信号具有白噪声的特性，当瞬时各子载波的相位一致的时候，就会出现很高的峰值，出现较大的峰值平均功率比（PAR）。较高的峰值对发射机放大器的线性度提出了很高的要求。如果放大器线性度不好，会造成信号的畸变，产生交调干扰而破坏频谱分布[7]。

2.2.3 MAC 层接口（MAC Interface）

MAC 层接口的主要功能：

① MAC 层的数据单位为字节（Byte），而 Baseband 的操作对象为比特流。MAC 层接口单元为 MAC 和 Baseband 之间提供并串转换。

② 为 Baseband 提供各种控制信号，例如数据传输率、编码速率、发送开始、发送结束、接收开始、接收结束、调制方式等等。

③ 组装和解析 PLCP 信头的信息。

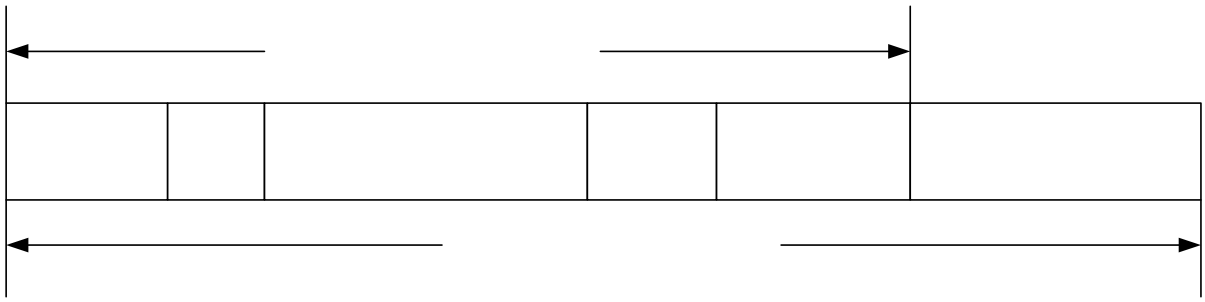
④ 计算发送持续的时间。

总的来说，MAC 层接口是一个控制单元，没有过多的算法内容。但是它提供了很多 Baseband 的控制信号，同时作为 MAC 和 Baseband 的连接桥梁，有很重要的作用。

首先，在 Baseband 当中，我们处理的一般是比特流。然而，正如上文所说，MAC 层的算法由一块嵌入式微处理器 MCU 控制，MCU 的总线为 32 位。MAC 的内部硬件逻辑已经将 32 位的内部总线转换为字节流，同时在 MAC 当中，所有数据长度单位都是字节，所以必须由 MAC 层接口来完成并串转换工作。

同时应当注意到，MAC 层的时钟为 40MHz，而 Baseband 的时钟为 80MHz。并且，在发送时，每 11 个时钟周期左右 Baseband 会从 MAC 层接收一个字节，同样在接收时，每 11 个时钟周期左右 Baseband 会向 MAC 发送一个字节。（如果没有特殊说明，所有模块都是发送和接收共用的）Baseband 的调制和解调过程一旦开始，中间是不能停顿的。因此接口单元必须要保证 MAC 和 Baseband 两边的速率匹配。

在控制信号方面，MAC 层给 Baseband 的只有需要发送的数据和控制位。而 MAC 层接口需要通过这些控制位组装出相应的 PLCP 信头。信头的结构如图 2.9 所示。



信头结构中，头 4 比特表示数据的传输速率。具体的速率分配由表 2.1 说明。保留比特直接置位为 0。长度字段一般由 MAC 层直接给出，用以表示 MAC 层一帧所包含的字节数，最大值为 4095，即长度字段的所有比特位被置为 1。奇偶校验位为偶校验。尾字段为恒定的 6 个 0。服务字段一般为全 0，其中的前 7 个 0 用来同步接收方的解扰器，后 9 个 0 为保留位。在发送的时候，服务字段的前 7 个比特被预先置为加扰器的种子。关于加扰器将会在下一小节论述。

表 2.1 OFDM 速率和调制参数表

传输速率（Mbps）	速率编码	子载波调制方式	编码速率	单符号传输比特数
6	1101	BPSK	1/2	24
9	1111		3/4	36
12	0101	QPSK	1/2	48
18	0111		3/4	72
24	1001	16-QAM	1/2	96
36	1011		3/4	144
48	0001	64-QAM	2/3	192
54	0011		3/4	216

信号符号（Signal Symbol

速率 4bits

保留 1bit

长度 12bits

PLCP 信头

图2.9 PLCP 信头

注：详细细节请参阅*IEEE Std 802.11a-1999: High-speed Physical Layer in the 5 GHz Band*[8]

另外，在每一次发送的开始，MAC 层需要知道此次通讯所持续的时间。而持续时间和发送所采用的数据传输率相关，由 MAC 层接口在发送的开始时计算并返回给 MAC 层。具体计算公式由式（2.5）给出：

$$t_{dur} = \left\lceil \frac{8*length+22}{N_{DBPS}} \right\rceil *4+22,(\mu s) \qquad (2.5)$$

式（2.5）中，t_{dur} 为持续时间，单位为 μs，length 为 12 比特的长度值，N_{DBPS} 为单符号传输比特数。详细的计算过程将在 3.2.4 小节中讨论。

2.2.4 扰码器（Scrambler）

加扰属于信源编码的一种。不加处理的 MAC 层数据流往往具有短周期、长 0 或长 1 的特性。而这些特性会给通讯带来种种的麻烦。

具有短周期的数据，会在它们的频谱上产生峰值较高且频率较高的离散单音峰值信号。这些离散单音可能和载波或者调制信号发生交调，产生非线性频谱，落入相邻信道内，造成邻道干扰。经过加扰处理，可以使周期较短的信号变为周期较长的信号，减小信号产生的单音频率和峰值。这样，即使这些单音信号还是和载波或调制信号发生交调，主要产生的交调干扰还是落在信道内部，而不会对相邻信道产生太大的干扰。

连续长串的 0 或 1，在某些通过数字信号来获取同步的系统当中会造成失同步的问题。同步的获取是通过定位信号的过渡点完成的，而过渡点往往是信号电平的转换点。如果信号电平出现长 0 或者长 1，那么在很长一段时间内将没有过渡点出现，这样就会产生失同步的问题。经过适当的加扰器，可以将长 0 或者长 1 信号转变为过渡点占传输数据近 50% 的信号。不过，在 OFDM 方式当中，加扰器的主要作用不在于获取同步。

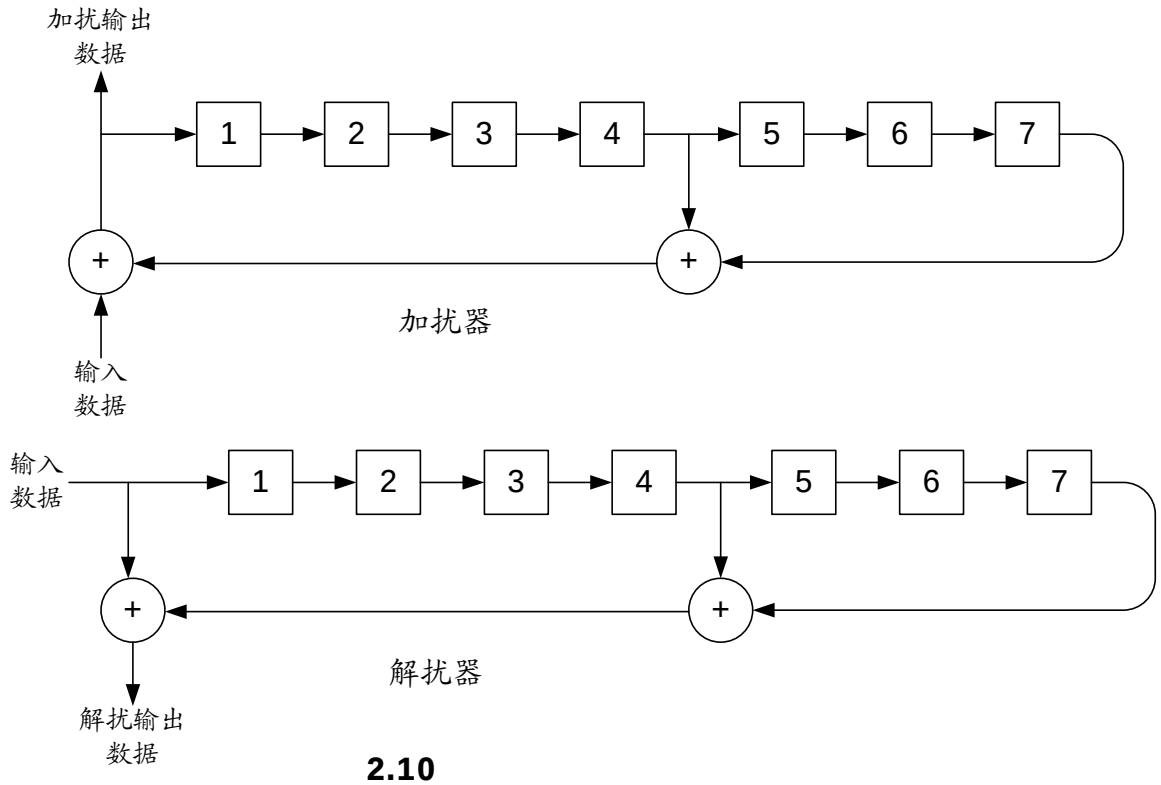
在通讯系统当中，我们不希望系统的数据出现直流分量（DC）。所谓DC，就是信号的频谱在 0 频点出现较高的谱线，即信号流的 0 和 1 的概率不平衡。这样的DC，会给时域均衡造成很大的问题。而加扰器，可以将原信号变为具有多过渡点特性的信号，实际上就是改善了信号 0 和 1 概率不平衡的问题，能防止较大的DC出现[9]。

扰码器的基本原理，其实就是将信号通过特别选择的反馈移位寄存器。反馈移位寄存器的选择应当遵循最大限度扩大信号的周期的原则。我们称这样的反馈移位寄存器为最长线性反馈移位寄存器。

在 802.11g 物理层当中，我们使用移位长度为 7 的最长线性反馈移位寄存器，式（2.6）为它的特征多项式。

$$f(x) = 1 + x^4 + x^7 \qquad (2.6)$$

相应给出加扰和解扰的示意图（图 2.10）。



2.10

需要特殊说明的是，在 MAC 层接口，PLCP 服务字段的前 7 比特在传递给加扰单元时为加扰器初始化的随机种子。在加扰过程中，加扰器直接将这 7 比特置入 7 个移位寄存器当中。而在真正发送的数据流中，这 7 比特为 0。这样，在接收端解扰器工作时，从数据输入端直接移入 7 比特的 0，可以完全初始化解扰器。

2.2.5 卷积编码器（Convolutional Encoder）

卷积编码是于 1955 年由麻省理工学院的伊利亚斯（P. Elias）提出来的，属于非分组的纠错编码中的一种。卷积编码由原数据流通过一个有限状态的卷积编码器产生。一般的卷积编码器由k个m级移位寄存器和几个模二加法器组成。原数据依次进入k个移位寄存器，并根据每一个时刻移位寄存器的状态产生n个结果[10]。也就是说，卷积编码器每输入k个数据，可以得到n个结果，且$n>k$。这样，卷积编码器的输出码流当中，每n个数据当中，就会出现$(n-k)$个冗余数据，这些冗余数据不仅和它所对应的输入数据相关，还和前后传递的码元相关。这样就在发送数据序列内部加入了一定的相关性。

当信道有噪声影响时，在接收端的数据很可能由于噪声的叠加，产生随机错误。如果接收的数据存在这种冗余编码，即数据之间存在一定的相关度，我们便能从冗余编码和相关分析当中纠正随机错误，提高系统的抗干扰能力。所以，数据当中的冗余数据比重越大，码元间的相关距离越远，那么系统的纠错能力就越强。

一般来说，我们定义卷积编码的编码率为$R=k/n$，而相关度由移位寄存器个数 m 决定。卷积编码的编码率越低，说明冗余编码的比重越大，则抗干扰能力更强。同样，m 越大说明码元间的相关距离越长，相关度也就越大。

在 802.11g 当中，使用移位寄存器个数$m=6$的卷积编码器，和符合行业标准的生成多

图 加扰和解

项式 $g_0=(133)_8, g_1=(171)_8, R=1/2$。为了获得更高的编码速率，采用“截短”的方法，在发送时相应去掉一定的比特，而在接收时，在相应位置插入哑元，再进行解码。802.11g 所使用的编码速率有 1/2、2/3 和 3/4 三种，使用情况由表 2.1 规定。

其编码器结构如图 2.11 所示，丢弃相应比特和插入哑元的过程由图 2.12 给出。在发送方向，采用卷积编码，在接收方向，则采用维特比解码器对卷积编码进行解码和纠错。

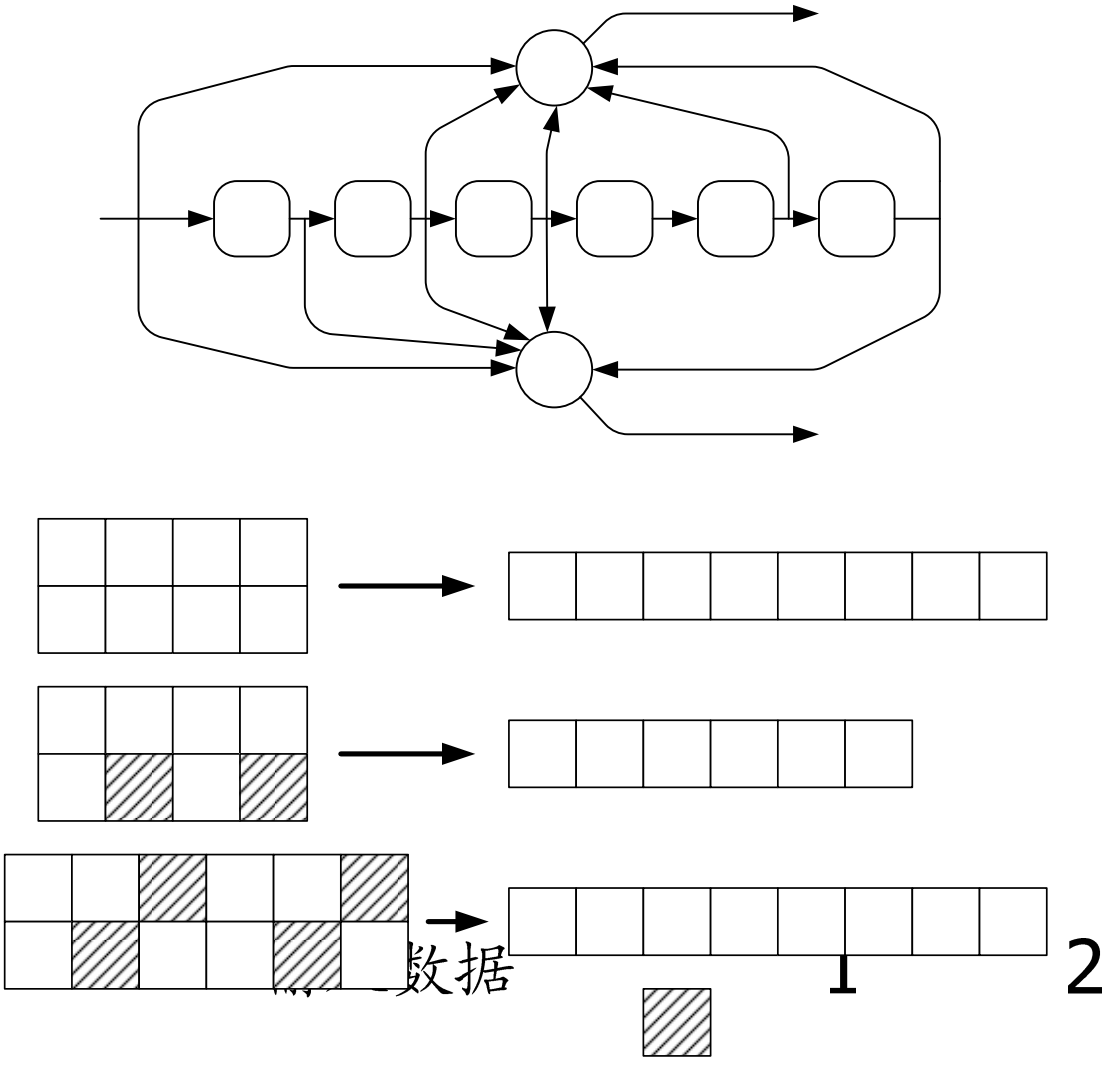


图2.11 卷积

2.2.6 维特比解码器（Viterbi Decoder）

对于卷积编码，我们有两种方法对其进行译码：代数译码和概率译码。其中概率译码将信道的统计特性考虑到译码当中，利用极大似然准则的方法进行估计，译码准确性较高。维特比译码就是概率译码中的一种[9]。

极大似然译码就是将接收序列与所有可能的传输序列相比较，求其汉明距离，从中选择对应最大似然函数的传输序列，然后确定译码器的输出。汉明距离为两个长度为n的编码序列，出现对应比特位不同的个数。但是，如果直接做极大似然比较，对应长度为L比特的编码序列，对应的比较需要进行 2^L 次，与序列长度成指数关系，难以实现[10]。

A0 A1 A2 A3

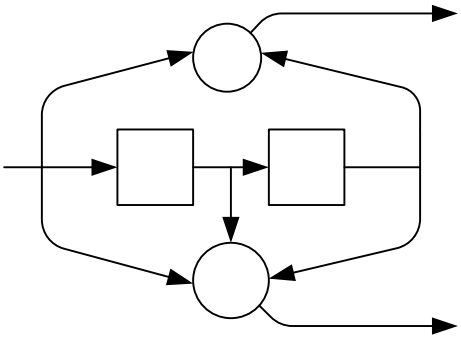
R=1/2

在 802.11g 系统当中，我们采用维特比硬判决的解码方法。将接收到的序列对应卷积编码的网格图，确定最可能的序列。只保留一定的深度，以及在此深度下一定数量的最可能的发送序列。每次解出一个码元就重新计算各保留序列的概率，并抛弃概率最小的情况。保持考虑深度和保留序列的数量不变。

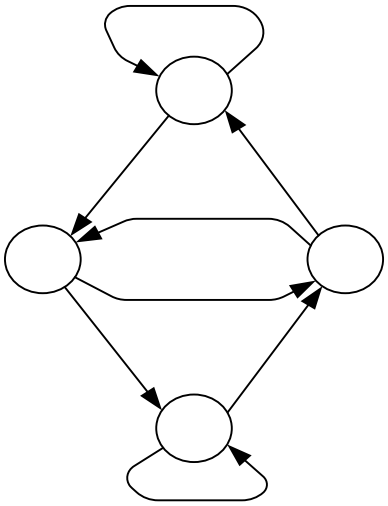
应当说明，判决网格图的大小和卷积编码器移位寄存器个数成指数关系。因而，移位寄存器个数m越大，说明码元的相关度增大，但是也同时意味着解码的复杂度增大。后文仅用m=2 的情况对维特比解码器做出说明[10]。

1．网格图

对应 m=2 的情况，采用生成多项式 $g_1=(101)_2, g_2=(111)_2$。其编码器结构如图 2.13 所示。



对应 m=2，所以整个编码器共有 4 种状态，即移位寄存器里的不同情况。我们可以根据这 4 种不同的状态，得到编码器在不同情况下，状态的转换和输出情况（图 2.14）。



在状态转换图中，我们用圆圈表示不同的状态，即移位寄存器当中的编码。连线方向，和连线上方的 0 或 1 表示输入原码，而连线下方的 00、01、10、11 表示在当前状态下的输

输
入
1

出。

从状态转换图和卷积编码的结构当中，我们可以得到这样的结论，从任何相同状态出发的分支都相同。因而我们可以把同级相同的点合并得到网格图（图 2.15）的表示方式。网格图中直接规定输入 0 对应上分支，输入 1 对应下分支。一般情况下，系统都需要在编码前初始化卷积编码器，使卷积编码器的移位寄存器都为 0，即网格图可以从 00 状态开始考虑。当我们的输入序列为 10110 时，通过网格图和计算我们可以得到输出序列为 1101001010，如图（2.15）所示。

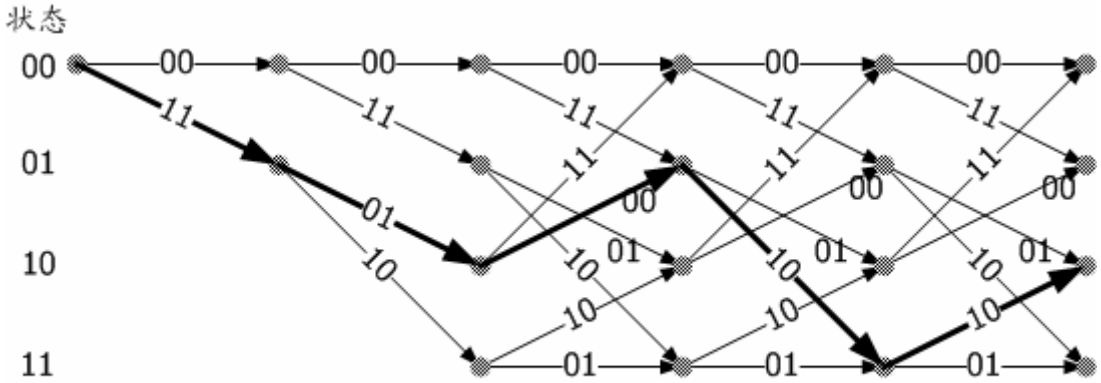


图2.15 卷积编码网格图

从图中可以看出，在任意时刻的任意状态，当 n 等于 2 时，实际上只可能转变为其他两种状态。如果从接收序列上发现转换为其他的状态，也就说明出错。而总的状态数由 m 的大小决定。因此，当 m 越大，状态越多，那么发现错误的可能性也就越大，也越有可能纠错。

而实际上，维特比硬解码过程，就是通过在对应接收序列，在网格图上寻找出一条可能性最大的路径，这条路径的状态转换也就反映了原码序列。

2．算法流程

可以简单假设算法的深度为 3，保留的路径数为 3，接收序列本应该是上文所求得的 1101001010，但是在传输过程中出现了两个误码，使得接收序列变为 1001000010，利用维特比硬解码算法进行解码。

首先在解码前，必须假设维特比解码器已经初始化，可以从 00 状态开始解码。那么，硬解码过程会使用如下算法进行解码。

① 计算所有可能路径与接收序列的汉明距离。

② 如果达到保存深度，从所有可能路径当中选择一条汉明距离最小的路径（如果出现汉明距离相等的路径，随机选择一条）。

③ 在所有剩下的路径当中选择 3 条汉明距离最小的可能路径，作为保留路径（如果出现汉明距离相等的路径，随机选择一条）。

④ 判断是否接收完毕。如果没有，回到①

⑤ 接收完毕，选择汉明距离最小的路径，输出最后 3 比特解得的原码。

具体算法计算过程如表 2.2 所示：

表 2.2 维特比硬解码过程

接收序列	保留路径		状态转换	汉明距离	输出结果
10	1	00	00	1	—
	2	11	01	1	
	3	—	—	—	
01	1	00-00	00-00	2	—
	2	00-11	00-01	2	
	3	11-01	01-10	1	
00	1	00-00-00	00-00-00	2	—
	2	11-01-11	01-10-00	3	
	3	11-01-00	01-10-01	1	
00	1	01-00-10	10-01-11	2	1
	2	01-00-01	10-01-10	2	
	3	01-11-00	10-00-00	3	
10	1	00-10-10	01-11-10	2	0
	2	00-01-00	01-10-01	3	
	3	00-01-11	01-10-00	3	
结束	—	00-10-10	01-11-10	2	110

正如表中所得到的结果，尽管接收码流当中出现了两个误码，误码率达到 20%，但是解码结果 10110 和输入序列一致。下面继续给出在网格图上的计算过程（图 2.16）。在每一条路径上标有该路径的汉明距离。显而易见，最后保留并被选中的路径就是汉明距离最小的路径。则该路径作为极大似然函数的结果，被解码器认定为原序列，并输出。

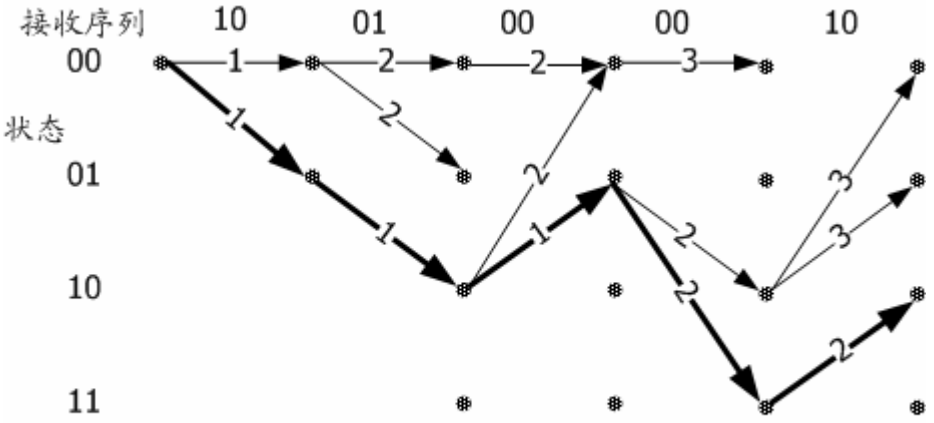


图2.16 接收序列搜索路径

需要进一步说明的是，维特比解码能够正确启动的前提是解码器的初始化。从卷积编码来看，802.11g 使用 m=6 的卷积编码器。所以，如果原码序列为连续 6 个 0，那么就可以肯定，在解码时从 000000 状态开始解码。所以，在传输的数据流中，PLCP 信头的尾字段总是保持 6 个 0，就是确保卷积编码器的初始化。同时在接收方向，解码过程可以确定从 000000 状态开始。另外，在每一帧的末尾，也必须有连续 6 个以上的 0。

2.2.7 交织器（Interleaver）

无线信道的噪声，并不仅仅包括随机噪声。由于发射天线到接收天线的多径效应，接收到的信号也许会是直射波、反射波、折射波甚至衍射波的叠加。如果考虑到发射机或者接收机处于移动当中，还会有多普勒频移效应等等。

这些噪声，有时候会造成数据的突发性错误，或者一个频点上的信号强度极其微弱，直接导致一个子信道所有的数据都发生误码，我们称之为衰落效应。那么，即使在接收方向，接收机采用维特比解码器对数据进行解码，也不能解决误码问题。维特比解码的纠错能力和卷积编码器的移位长度 m 有关，如果突发性错误的误码长度大于 m，那么使用维特比解码的方法无论如何也不可能得到正确的结果。并且，由于维特比解码器的构造，还会造成错误传播，进一步造成误码率的提高。

所以，对应于无线信道的这种突发性误码干扰，802.11g采用交织技术将卷积编码的相关编码输出均匀地分配到各个子信道上，在接收端再通过交织器将分散在各个子信道上的编码重新编排回原来的顺序。这样，即使一个子信道上的信号强度衰落为 0，所有由该信道传输的数据都出现了误码，但是由于交织作用，出现误码的序列由接收端交织器重新编序之后，分散到整个符号的数据序列的区域。这样就将原来突发性的误码分散到整个接收序列，近似于随机误码，维特比解码起就有能力对其进行纠正了[6]。

在 802.11g当中，交织器要对卷积编码器的输出序列做两次变换[8]。

第一次变换：

每一个符号利用 48 个子信道进行传输，交织器将这 48 个子信道分成 16 组，根据式（2.7）将原序列均匀分布到 16 个组中，并确保在原码流中相邻的码元处于不同的分组。

$$i=(N_{CBPS}/16)(k \bmod 16)+floor(k/16), k=1,2,\ldots,N_{CBPS}-1 \quad (2.7)$$

N_{CBPS}：单符号传输的比特数，k：原码流序列下标，i：第一次变换后的下标。

第二次变换：

将每一个子载波上的相邻编码比特交替地映射到星座的高有效位和低有效位上，根据式（2.8）进行变换。

$$\begin{aligned} j &= s*floor(i/s)+(i+N_{CBPS}-floor(16*i/N_{CBPS})) \bmod s \\ s &= \max(N_{BPSC}/2,1) \end{aligned} \quad (2.8)$$

N_{BPSC}：单载波传输比特数

在接收方向，相应对序列做逆变换即可，逆变换公式（2.9）和（2.10）。

$$i=s*floor(j/s)+(j+floor(16*j/N_{CBPS})) \bmod s \quad (2.9)$$

$$k=16*i-(N_{CBPS}-1)floor(16*i/N_{CBPS}) \quad (2.10)$$

2.2.8 布局器（Mapper）

如上一小节所述，在 802.11g 当中，数据传输只需要使用 48 路子载波。然而，一个 802.11g 的 OFDM 信道，在数据传输时，同时有 52 路子载波在同时工作，编号-26 到 26，其中第 0 号为直流载波子信道而不使用。其中第-21、-7、7 和 21 用于导频子载波，可以在发生频率迁移和存在相位噪声的情况下进行稳定的相干检测。

所以，布局器的主要作用，就是将交织器的输出码流，对应不同速率下的调制方式，映射到 IFFT 单元相应的频点。

在BPSK调制时，每个子载波传输 1 比特信息，QPSK调制时为 2 比特，同理 16-QAM 时为 4 比特，64-QAM为 6 比特。布局器根据调制方式，将码流分为 48 组，编号为k（k = 0,1,…,47）。布局完成后对应的子信道为M(k)，分布函数如式（2.11）所示[8]。

$$M(k)=\begin{cases} k-26, & 0\le k\le 4 \\ k-25, & 5\le k\le 17 \\ k-24, & 18\le k\le 23 \\ k-23, & 24\le k\le 29 \\ k-22, & 30\le k\le 42 \\ k-21, & 43\le k\le 47 \end{cases} \tag{2.11}$$

布局器另外的作用，就是根据不同的调制方式，对数据进行归一化处理。由于发射机必须保证发射信号的平均功率恒定。因而对应不同的调制方式，每个频点的数据应乘以归一化因子，使得在不同调制方式下，发射机都能输出相同的平均功率。表 2.3 给出了在不同调制方式下的归一化因子[8]。

表 2.3 归一化因子 K_{MOD}

调制方式	K_{MOD}
BPSK	1
QPSK	$1/\sqrt{2}$
16-QAM	$1/\sqrt{10}$
64-QAM	$1/\sqrt{42}$

3 Baseband 的接口设计

对于规模较大的系统，系统内部模块之间，第三方模块（IP 模块）和本方模块之间，不同时钟域之间，有时会出现信号格式或速度不统一的问题。在这种情况下，一般需要接口单元来统一信号。

不同时钟域之间的数据传输，需要防止时钟频率较高的时钟域向频率较低的时钟域传输突发性的数据导致数据丢失，也需要解决由于时钟频率较高的时钟域的控制信号维持时间过短而导致频率较低的时钟域出现采样遗漏的问题。

而对于 IP 模块，由于模块的接口已经确定，一般不能更改。因此，如果出现端口信号不一致的问题，只能添加接口模块，提供针对于 IP 模块的信号转换功能。

有时在内部模块之间也需要接口单元，这有时是由于代码的重用引起的。尽管模块都是 RTL 代码，但是很有可能另一个模块为他人所写，已经经过很多严格的测试，现在不好更改。甚至接口两边的模块都是重用的模块，只有通过接口来达到统一。

本设计当中的 Baseband 模块也不例外。

3.1 Baseband 的接口概述

Baseband 的接口主要有三个，和 MAC 层的接口，APB 接口和 RF 接口。

这三个接口有一个共同的特点，都是跨时钟域。由于 Baseband 是整个 802.11g 网卡中最复杂，同时计算量也最大的模块，所以 Baseband 使用 80MHz 的时钟。而 MAC 和 RF 都使用 40MHz 的时钟，因而 Baseband 和外部的接口都需要考虑跨时钟域的问题。

和 MAC 的接口主要提供 Baseband 和 MAC 之间的数据通路，另外接口还必须负责 PLCP 信头的组装和解除的工作。其内在原因是由于在 Baseband 代码中，包括 MAC interface 的一部分到 FFT 单元都是第三方 VHDL 代码，已经经过严格验证，除非在特殊情况下，一般不做改动。而 VHDL 代码的 MAC interface 当中没有提供 PLCP 信头组装和解除的功能，需要另外编写接口单元进行补充。

APB 接口的主要目的是将整个 Baseband 作为 ARM APB 总线上的一个从设备，提供 MCU 直接读写 Baseband 参数寄存器的功能。通过更改这些寄存器，能够使 MAC 层更好地控制物理层参数，并且提供系统根据信道状态动态更改物理层特性的可能。

和 RF 的接口比较简单，大多数都是信号直连，唯一需要注意的问题是要根据 RF 芯片当中 AD 和 DA 的要求更改输入和输出数据的格式。

3.2 Baseband 和 MAC 的接口设计

3.2.1 结构功能概述

MAC 接口单元，如 2.2.3 节所述，提供并串转换、组装发送方向 PLCP 信头、解析接收方向 PLCP 信头、生成帧开始和帧结束信号、自动填充尾字段、计算发送持续时间等等

功能。原 VHDL 代码已经提供并串转换、解析接收方向 PLCP 信头、生成帧开始帧结束信号、自动填充尾字段这三个主要功能，因而设计当中只需要着重解决 PLCP 信头的组装、接收时从数据流中去除 PLCP 信头、计算发送持续时间和跨时钟接口等等问题。

3.2.2 PLCP 信头的组装与去除

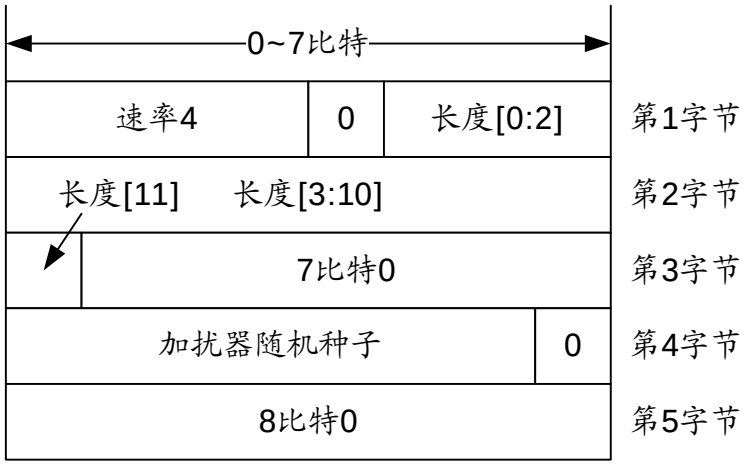
PLCP 信头结构如图 2.9 所示，一共 5 个字节，主要包括长度，传输速率和加扰器的 7 位随机种子。

在发送阶段，接口单元需要组装出符合协议标准的 PLCP 信头。其中长度字段直接从 MAC 层接口 bb_tx_psdu_length 信号获得，传输速率从 MAC 层 bb_tx_data_rate 信号译码得到。bb_tx_data_rate 为 8 比特信号，和 802.11b 速率传输格式兼容，因此需要将 8 比特的数据传输率转变成 4 比特的 802.11a OFDM 速率表述格式（802.11g 的 OFDM 和 802.11a 兼容，直接使用 802.11a 协议规范），其映射由表 3.1 规定。

表 3.1 传输速率转换表

数据传输率	8 比特 MAC 表示	4 比特 Baseband 表示
6	10001100	1101
9	10010010	1111
12	10011000	0101
18	10100100	0111
24	10110000	1001
36	11001000	1011
48	11100000	0001
54	11101100	0011

最后组装得到的 5 字节 PCLP 信头结构如图（3.1）。



3.1 5

其中加扰器随机种子为 7 比特的任意值。在信头之后就是 MAC 发送的数据，直接将数据传给 Baseband 即可。也就是说，在整个传输数据之前，需要将 5 个字节的信头传给 Baseband。

接收方向，Baseband 首先传递给接口模块的 5 字节相应为接收帧的 PLCP 信头。由于 MAC 只需要从 Baseband 获得数据流，因此接口单元需要将 PLCP 信头从接收字节流中去除。

此外，接口单元还需解析信头当中的长度和速率字段，并将信息提供给 MAC 层。其中长度字段由原 MAC 接口单元信号 rx_psdu_length 直接提供，因此直接传递给 MAC 层即可。另外，原 MAC 接口单元能够提供符合 802.11a 规范的调制方式（rx_mod_type）和编码速率（rx_code_rate）信息，可以通过这两个信号直接译码得到接收帧的传输速率（bb_rx_data_rate），如表 3.2。

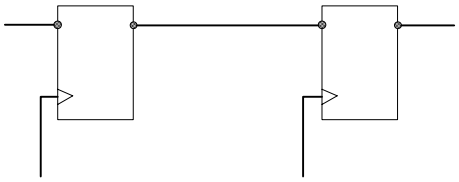
表 3.2 接收数据传输率译码表

传输率	调制方式	编码速率	8 比特传输速率
6	00	00	10001100
9	00	10	10010010
12	01	00	10011000
18	01	10	10100100
24	10	00	10110000
36	10	10	11001000
48	11	01	11100000
54	11	10	11101100

3.2.3 跨时钟域的数据传递

跨时钟域主要有两个问题：控制信号的采样和数据传输的速率匹配。

首先说明第一个问题。在 VLSI 设计当中，控制信号一般由寄存器输出，因此控制信号的最短维持时间为一个时钟周期。在通常的情况下，对信号的采样使用上升沿触发的寄存器。如图 3.2。



如果在同一个时钟域内，控制寄存器和采样寄存器的时钟端使用同一个时钟信号，采样寄存器每一个周期都会对控制信号采样一次，因而总能够探测到控制信号的变化。（图 3.3）

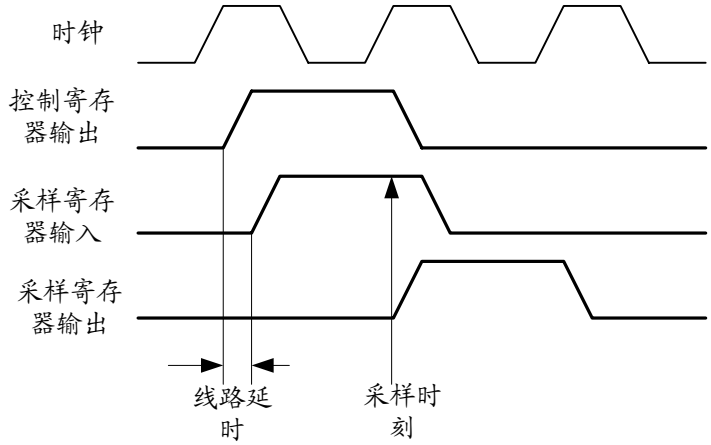
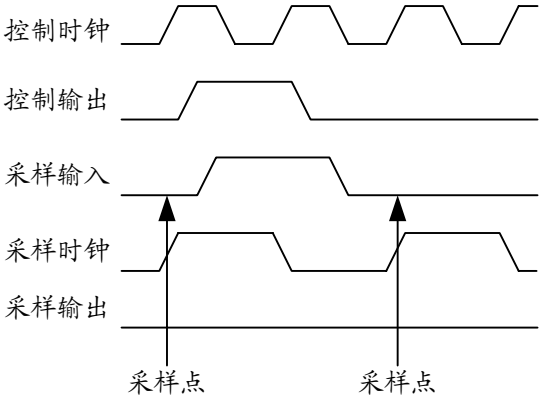


图3.3 同时钟域采样

如果控制寄存器和采样寄存器处在不同的时钟域，而且控制寄存器的时钟周期较短，就很有可能发生失采样的问题，如图 3.4 所示。

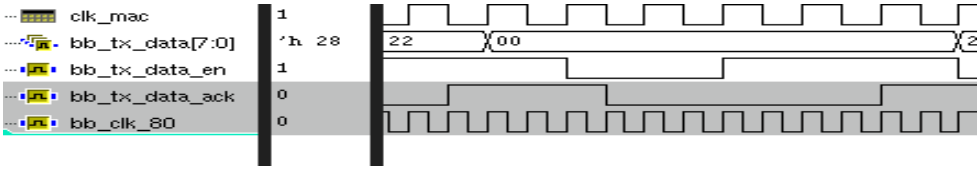


3.4

由于控制时钟较快，因此控制寄存器电平维持时间过短，而导致采样寄存器不能锁住控制信号，出现了失采样。

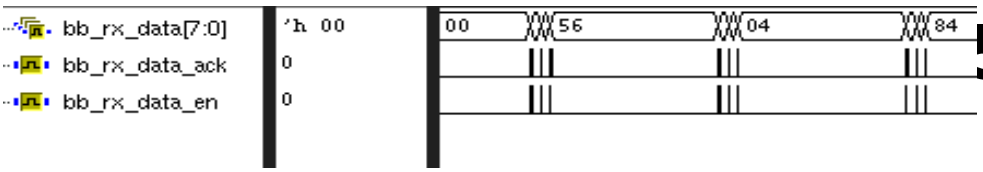
而 Baseband 与 MAC 的接口就是这种情况。Baseband 使用 80MHz 的时钟，而 MAC 的时钟只有 40MHz，因而能够确保 Baseband 采样到 MAC 的数据就绪（bb_tx_data_en）信号。但是，Baseband 给 MAC 的数据接收（bb_tx_data_ack）信号，如果不考虑 MAC 层是否采样，而仅仅维持一个周期，就会发生失采样而丢失数据。并且，即使 MAC 的采样寄存器采到了 Baseband 的信号，有时因为寄存器已满，MAC 层需要延时响应，也会出现数据丢失的问题。

所以，为了确保 MAC 层接收到相应的控制信号，必须采用握手信号的方式。即 Baseband 数据接收完毕（bb_tx_data_ack）信号必须等待 MAC 的数据就绪（bb_tx_data_en）信号下降之后，才能回到低电平，确保 MAC 层的采样寄存器采到了数据接收完毕信号。（图 3.5）



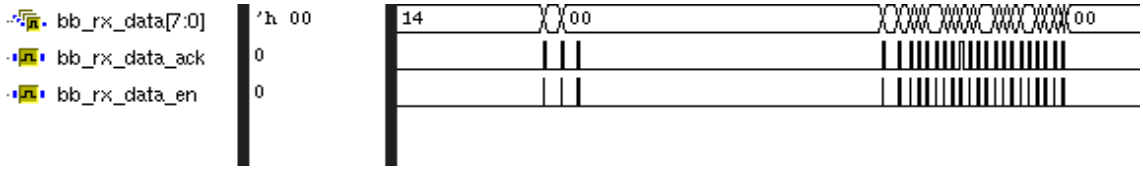
但是，尽管在使用握手信号的情况下，也不能保证数据的正确传递。由于握手信号实际上是利用等待来确保数据传递的正确性，是一种用时间换可靠性的做法。但是，如果数据传输是突发性的，数据与数据之间的等待时间变小，握手信号也不能保证数据传输的正确性。

实际情况当中，突发性的数据传输也是存在的。在正常情况下，由于符号间的间隔，导致接收数据的字节流并不是突发性的，在 3 到 12 个字节之后，会有一段间隔，如图 3.6 所示，因此 MAC 层有足够的时间处理接收字节流。



5 发送端

然而，由于维特比解码器的缘故，当一个完整的帧传输完毕时，维特比解码器会从所有的剩余路径当中选择一条最可能的搜索路经直接输出，导致突发性的数据接收，如图 3.7，MAC 层就有可能出现速度不匹配的问题。



解决该问题的办法是在接口模块内部为输入输出数据通路设置一个合适深度的 FIFO，当 MAC 层的接收速度达不到 Baseband 的要求时，通过 FIFO 的存储深度来暂时保存数据，防止数据丢失，从而达到速率匹配的目的。

所以，在这种情况下，FIFO 的深度成为比较关键的问题。如果 FIFO 的深度不够，那么当出现突发性数据流时，FIFO 不足以存储没有被 MAC 层接收的字节流，依然会出现速度不匹配的问题。而过深的 FIFO 会导致无谓的面积浪费，并提高了复杂度。

FIFO 的深度主要由两个因素确定，突发性数据流的数据传输率与突发数据的最大长度和 MAC 层的最低接收速率。如果在 MAC 的接收速率为最低接收速度，并且突发性字节流的长度为最大长度时，FIFO 的深度都能满足要求，那么该 FIFO 就可以保证在任何情况下都不出现速率不匹配的问题。

由于突发性数据流由维特比解码器在结束时，直接将最优路径输出导致，突发性字节流的最高速率为每 8 个 80MHz 时钟周期传输一字节，即每个 80MHz 时钟周期都输出 1 比特的接收数据，8 个时钟周期一个字节。而突发性数据长度为维特比解码器的路径长度深

图3.6 存在

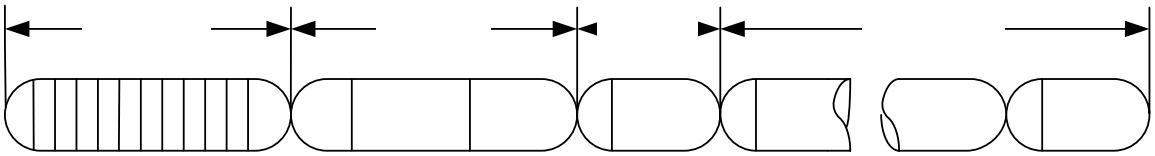
度，在这里为 150 比特，约 20 字节。而 MAC 层方向，由于接收到的数据需要保存到 MCU 的内存当中，以便 MCU 读取。其中需要经过异步接口、接收控制等等模块。因而 MAC 层的最低接收速度约为 6 个 40MHz 时钟周期接收一个字节。因此，接收 20 字节的时间为 120 个 40MHz 的时钟周期，相应为 240 个 80MHz 的时钟周期。而维特比解码器在最快速度下，每 8 个时钟周期输出一个字节，因而保留路径的 20 字节所需的输出时间为 160 个 80MHz 时钟周期。

MAC 的接收时间和维特比的输出时间相差 80 个 80MHz 时钟周期（240 和 160 的差值）。在没有 FIFO 的情况下，由于 MAC 层每 6 个 40MHz 时钟周期才会接收一个字节，将会有大约 7 个字节发生丢失。因此，FIFO 的深度应当为 7。在实际情况下，由于 MAC 的接收速度不会总是处于最低的接收速率，并且最后的保留路径当中包含尾字段，不需要 MAC 层接收，直接在解扰单元就已从接收数据流中去除，FIFO 的深度一般大于 4 即可满足要求。

3.2.4 发送持续时间计算

根据 MAC 层的相应规范，需要在发送开始时计算发送持续的时间，用以计算网络分配向量（NAV）。所以，在每一帧的开始时，MAC 层可以向 Baseband 发出计算帧发送时间的请求，具体就是使 bb_tx_computedur 信号从低电平变为高电平。Baseband 检测到 bb_tx_computedur 信号为高，则开始计算帧发送所需时间。经过大概 20 个时钟周期之后，计算完成，bb_tx_durack 信号从低电平变为高电平，MAC 则可以从 bb_tx_duration 信号获得具体的传输时间，单位为 μs 。

OFDM 一帧的时间分布如图 3.8 所示。



在每帧传输之前，会首先传递 10 个短训练序列和两个长训练序列，也称为短前导和长前导，以供接收机进行同步和校正。所有的训练序列共需要 16 μs ，如图 3.7 所示。紧跟前导为信号符号，传输 PLCP 信头部分，然后是数据符号，最后是数据、填充码和尾字段。尾字段共 6 比特 0，如果最后所剩数据和尾字段不够一个完整符号，则填入填充码构成一个完整的符号进行传输。并且，数据符号个数和数据传输率相关，数据传输率大，则符号数小。

所以，计算发送持续时间实际上只需要计算数据符号的个数 N，则总符号个数 $N_{symbol} = N + 1$，然后根据式 3.1 即可得到传输时间。

$$t_{dur} = 4 * N_{symbol} + 16 \qquad (3.1)$$

根据 PLCP 的帧结构，服务字段是跟随数据符号一同传输的。因此，数据符号的总比特数为 MAC 的帧长度乘 8（从字节换算为比特数），加上 PLCP 信头服务字段 16 比特，再

加上尾字段的 6 个 0 和不定个数的填充比特。因而，可以得到数据符号个数 N 的表达式：

$$N = \left\lceil \frac{length*8+16+6}{N_{DBPS}} \right\rceil \tag{3.2}$$

其中 N_{DBPS} 为每符号所传输的数据比特数。根据不同的调制方式，每个子载波传输数据比特数为 N_{BPSC}，每符号使用 48 路子载波进行传输，因此每符号传输比特数 N_{CBPS} 为：

$$N_{CBPS} = N_{BPSC} * 48 \tag{3.3}$$

根据不同的编码速率 R，可得到

$$N_{DBPS} = N_{CBPS} * R \tag{3.4}$$

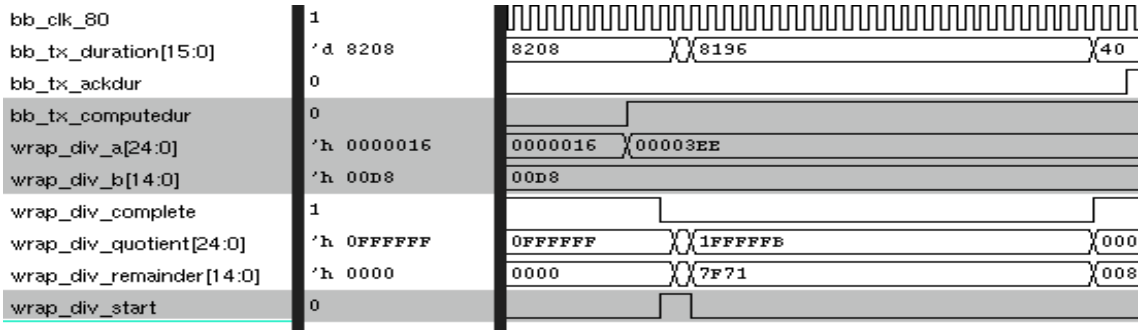
从式 3.3 和 3.4 可以推得不同传输速率下，每符号数据比特数 N_{DBPS} 的映射表（表 3.4）。

表 3.3 不同传输速率下的 N_{DBPS}

传输速率	调制方式	编码速率	N_{BPSC}	N_{DBPS}
6	BPSK	1/2	1	24
9		3/4		36
12	QPSK	1/2	2	48
18		3/4		72
24	16-QAM	1/2	4	96
36		3/4		144
48	64-QAM	2/3	6	192
54		3/4		216

得到 N_{DBPS}，代入式 3.2 和式 3.1 即可最终算出发送持续时间。综合各式即可得到 2.2.3 小节所述公式 2.5。

计算过程中需要除法运算，但是除法器在硬件电路当中，实现比较困难，并且占用资源较多。同时发送持续时间仅仅在每帧传输的开始计算一次。所以最终采用复用除法器的方法，和其他的模块共用一个 25 除 15 的串行除法器[11]。



3.2.5 异常处理

由于 802.11g 的 MAC 层要对应 DSSS 和 OFDM 两种方式的基带传输，因此针对于 OFDM 的异常处理需要由 Baseband 自己完成。对应 Baseband 与 MAC 接口模块的具体工作，即控制整个 Baseband 的接收使能信号（recv_enable）。

当然，发送部分的控制信号也由接口单元给出，主要是发送使能信号。当 MAC 的发送使能信号出现上升沿，说明一个发送帧的到达，接口单元相应给出发送使能信号。当整个发送完毕时，由发送频偏估计单元给出信道清理信号，确认发送完毕，从而 MAC 接口单元给出发送完毕信号返回 MAC 层，由 MAC 层复位发送使能信号，从而完成整个发送过程。由于其逻辑相对简单，这里不再赘述。

接收的过程相对复杂。由于信道当中的噪声干扰，会造成接收数据的错误。特别是 PLCP 信头部分的错误，会造成多个模块的状态机死锁。如果出现错误，说明整个帧的接收失败，需要对整个接收系统进行复位，最直接的办法就是复位接收使能信号。

图3.9 持

但是，MAC 层并没有处理 Baseband 异常的责任，因此不可以将异常交给 MAC 层以复位接收使能信号。所以，接口单元需要为整个 Baseband 构造接收使能信号。

1. 异常信号说明

Baseband 一共提供给 MAC 接口单元三个异常信号：ofdm_err、parity_err 和 data_rate_err。

这三个信号在正常无错的接收状态时都保持低电平，当出现错误，则出现高电平以提示有错误发生。

其中，ofdm_err 表示接收单元已经锁住了信号强度，并且等待了一段时间，但是并没有发现数据。该错误可能发生在信噪比过低的情况下，尽管接收机能够锁住信号，但是不能正确地确定帧开始时间，或者出现了误锁定。

而 parity_err 和 data_rate_err 发生在 PLCP 信头解析错误的时候。PLCP 信头当中包含速率和帧长度字段等关键信息。如果出错，整个帧的解析必然错误，因此需要重新初始化接收机，迫使当前帧丢弃。Data_rate_err 说明 PLCP 信头当中的速率字段不可识别，而 parity_err 说明 PLCP 信头当中的奇偶校验出错。需要注意的是，如果没有任何错误信号的出现，也不能保证接收正确。PLCP 信头部分采用 BPSK 以 6M 的速率传送，因此可靠性较高，而数据符号会以高传输率发送。并且，Baseband 只处理 PLCP 信头部分出错的情况，如果数据解析出错，只能通过 MAC 层处理。

2．接收使能信号的控制

MAC 接口单元需要为 Baseband 的接收部分构造接收使能信号（recv_enable）。该信号主要受到两方面控制。

一方面，如果 MAC 层的接收使能信号为低，说明系统当前处于发送状态或者收发转换的过渡状态，当然需要复位 recv_enable。

另一方面，如果出现了异常情况，也需要复位 recv_enable 信号。但是，这个复位时间应当是一定的。复位的作用，是使得接收方向的所有模块的状态机恢复到空闲状态，从而初始化状态机。如果复位时间过短，某些模块的状态机来不及复位，就可能造成状态死锁情况的发生；而复位时间过长，导致系统长时间处于不收也不发的状态，会导致丢帧现象的发生。

所以，综合考虑，如果出现异常情况，MAC 接口单元需要将 recv_enable 信号复位 8 个 80MHz 的时钟周期，以使所有接收模块的状态机复位。

3．抓沿电路

需要注意的是，异常信号由接收方向的各个模块给出，这些模块有可能使用 80MHz 的时钟，也有可能使用 40MHz 的时钟。错误信号可能为电平信号，持续多个时钟周期，也可能是脉冲信号，只维持一个时钟周期。然而，MAC 接口单元不可能要求其他模块给出特定形式的异常信号，因此只能自身处理。

最普通的办法就是使用抓沿电路。

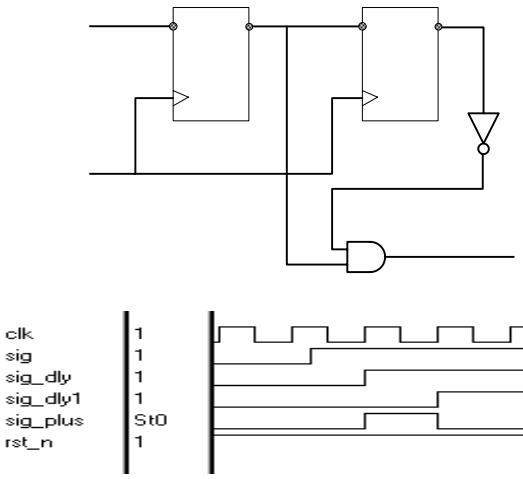
抓沿电路包含两个寄存器。一个寄存器直接寄存输入信号，其作用是获得输入信号的模块内部副本，并使得信号的上升沿和模块内其他的信号的上升沿一致，起到一定跨时钟域的作用。另一个寄存器继续将输入信号的模块内部副本寄存，以获得一个周期的延迟。

前一寄存器的输出和后一寄存器的反信号作与运算，便生成了一个脉冲信号，并且该脉冲和输入信号是沿信号还是电平信号无关，达到统一信号的目的。

假设原信号为 sig，经过抓沿电路之后的信号为 sig_puls，其 Verilog 代码的一般表达形式为：

```
reg sig_dly;       //内部信号副本
reg sig_dly1;      //内部信号副本的延迟
wire sig_plus;     //生成的延信号

assign sig_plus = sig_dly && (!sig_dly1);    //从两次寄存信号，获得脉冲信号

always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)
    begin
        sig_dly <= 1'b0;
        sig_dly1 <= 1'b0;
    end
    else
    begin
        sig_dly <= sig;          //寄存输入信号，获得内部信号副本
        sig_dly1 <= sig_dly;     //寄存内部信号副本
    end
end
```

相应的电路和波形如图 3.10 所示。



3.2.6 状态机的构成

状态机是控制模块当中的常见单元。状态机本身能够存储模块当前的状态信息。因此，模块能够根据状态机的状态，判断模块当前的主要任务。同时，状态机将模块内部大部分的控制任务，转化为确定状态转化的条件上，从一定程度上明确了模块的设计思路。

在 MAC 接口单元当中一共包含三个状态机，发送状态机、接收状态机和除法器的状态机。其中接收状态机最为复杂。论文当中也仅就接收状态机的功能、信号与编码展开讨论。

1．状态机的编码形式

一般来说，状态机的编码形式是固定的。

状态机首先应当定义状态机的不同状态，一般以 parameter 的形式定义。例如一个 4 种状态的状态机，可以定义成如下方式：

```
parameter IDLE = 2'b00;
parameter STA1 = 2'b01;
parameter STA2 = 2'b10;
parameter STA3 = 2'b11;
```

在 FPGA 当中，为了获得更稳定的状态转换，有时采用热点（one-hot）的状态机结构

```
parameter IDLE = 4'b0001;
parameter STA1 = 4'b0010;
parameter STA2 = 4'b0100;
parameter STA3 = 4'b1000;
```

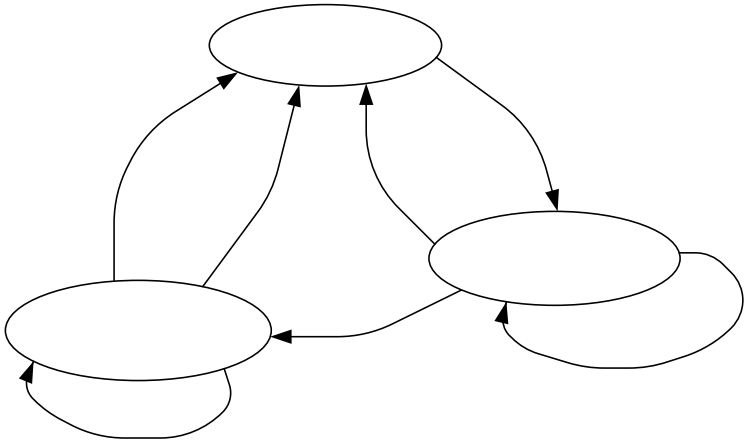
整个状态机一般由两部分组成，一个组合逻辑的状态转换判断模块，和一个状态存储模块。

```
reg [1:0] STA_CURRENT;   //当前状态
reg [1:0] STA_NEXT;      //下一周期状态

always @(STA_CURRENT or ... )   //状态转换判断
```

```
case(STA_CURRENT)
    IDLE:    //初始状态
        if(...)     //当满足初始条件，跳转状态 1
            STA_NEXT = STA1;
        else        //否则维持当前状态
            STA_NEXT = IDLE;
    STA1:    //状态 1
        if(...)     //满足转换条件，发生状态跳转
            STA_NEXT = STA2;
        else if(...)     //出现异常，回到初始状态
            STA_NEXT = STA1;
        else        //否则维持当前状态
            STA_NEXT = IDLE;
    STA2:
        ...
    STA3:
        ...
    default:  //在未知情况下，恢复状态机为初始化状态
        STA_NEXT = IDLE;
endcase

always @(posedge clk or negedge rst_n)    //状态存储
if(!rst_n)
    STA_CURRENT <= 0;
else
    STA_CURRENT <= STA_NEXT;
```

状态转换判断模块通过对各种信息的处理，提前获得下一周期的状态信息。而状态存储模块，仅仅将状态转换判断模块的结果，直接寄存到状态寄存器当中，完成状态的跳转。

2．接收状态机分析



接收状态机分为三个不同的状态：初始状态（IDLE）、PLCP 信头接收状态（STA_SIGNAL）和数据接收状态（STA_DATA）。

在上电之后或者处于发送状态时，状态机处于 IDLE 状态。当出现接收开始信号，也就是时域单元锁住了信号强度，并且已经监测到前导序列之后，状态机进入 STA_SIGNAL，准备从 Baseband 接收字节流。否则，状态机保持初始状态。

一旦进入 STA_SIGNAL，MAC 接口单元立即给出 recv_enable 信号，整个 Baseband 开

始工作。正如上文所述，PLCP 信头包含 5 个字节，需要从接收字节流当中去除。因此信头计数器开始工作，如果连续从 Baseband 收到 5 个字节，即计数器计到 4，状态跳转为 STA_DATA。在这个过程中，可能出现各种异常情况。如果发生异常，通过上文所述的抓沿电路，接口单元将复位 recv_enable 信号 8 个时钟周期，并自动跳回初始状态。

在实际验证的过程中，还出现帧长度为 0，但是不出现异常信号的错误情况。在该情况下，状态机会从 STA_SIGNAL 状态直接跳回 IDLE 状态，引起某些单元的状态死锁。为了让 MAC 层知晓有一帧到来，但是出现了错误，从而复位 MAC 层的 bb_rx_en 信号，必须提前返回 MAC 层接收就绪信号，但并不向 MAC 层发送数据。在信头计数器为 2 的时候，说明 PLCP 信头当中的编码速率和帧长度已经正确解析，则应当返回 MAC 层接收就绪信号，MAC 层则可从帧长度为 0 的信息中判断出接收出现错误，从而复位 bb_rx_en 信号。

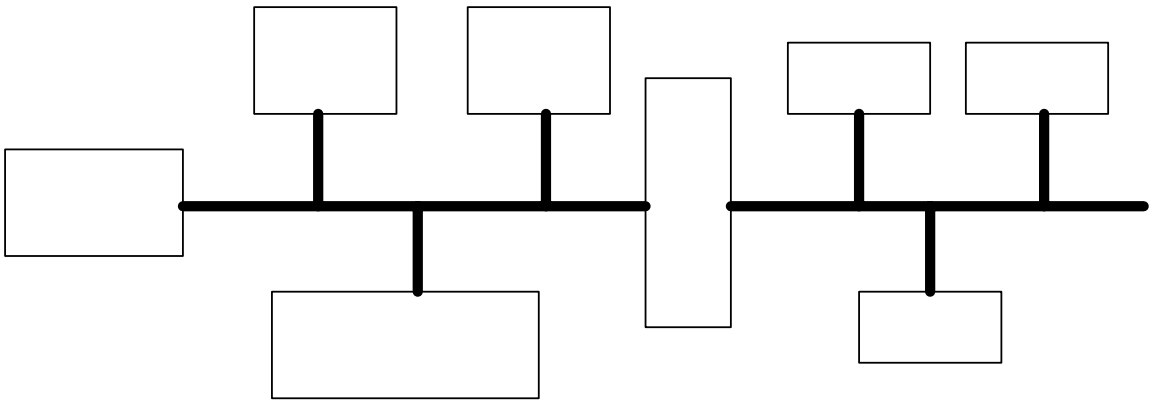
进入 STA_DATA 状态之后，MAC 接口单元将直接把 Baseband 的字节流传递给 MAC 层，直到一帧的数据传送完毕。当然，在这个过程中也可能出现异常情况，可能从数据接收状态直接返回到初始状态。

3.3 Baseband 和 APB 总线的接口设计

3.3.1 APB 总线标准

APB 总线为 ARM 公司 AMBA 总线上 AHB 总线的二级总线。提供了低速和低功耗设备与 AHB 总线的连接。关于 AHB 总线的协议规范并不属于本论文的讨论范围，详细内容请查阅参考文献 12：*AMBA™ Specification (Rev 2.0)*。

图 3.12 给出了AMBA总线的基本构架[12]。



从图中可以看出，APB 总线设备，通过 AHB 和 APB 总线桥和 AHB 总线相连。在整个 APB 总线当中，只有一个主设备，就是 AHB 和 APB 总线桥，而其他单元都为总线的从设备。

因而，整个总线由总线桥控制，而总线桥为 AHB 总线上的从设备，因此 MCU 可以很容易地通过 AHB 和 APB 总线桥访问到 APB 上的所有设备。

此外，APB 总线是一种无三态的总线设计，当设备不占用总线时，并不需要输出高阻。实际上每一个 APB 从设备都有自己的数据总线输出，这些设备的输出总线连接到总线选择

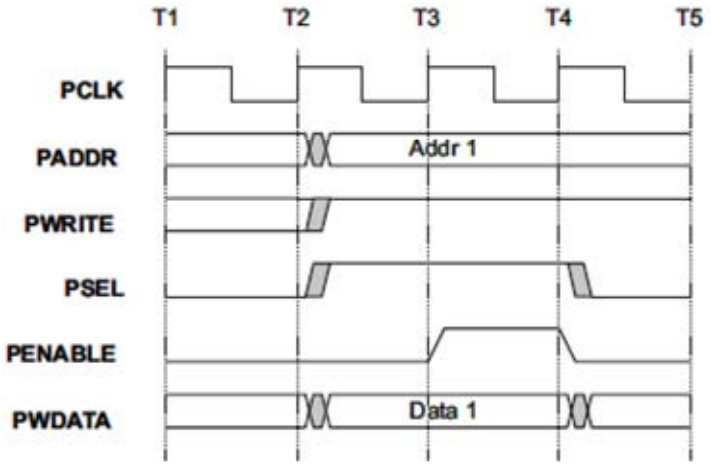
单元。总线桥单元具有对 APB 设备的选择权，因而总线桥能够很明确地知道当前时刻需要从哪一个从设备的输出总线上取得数据，因此能够通过对总线选择单元的控制，选择一个从设备的数据输出总线与总线桥相连，达到选择设备并建立连接的功能。

通过这样的设计，APB 总线能够避免三态门的使用，因而能很容易地在 VLSI 设计中实现。为了符合 APB 总线标准，APB 从设备必须提供以下接口信号：

表 3.4 APB 总线接口信号

信号名称	方向	位宽	描述
PCLK	I	1	APB 总线时钟
PADDR	I	32	APB 总线地址
PWRITE	I	1	APB 写信号，高有效
PSEL	I	1	APB 片选信号，高有效
PENABLE	I	1	APB 使能信号，高有效
PWDATA	I	32	APB 写数据总线
PRDATA	O	32	APB 读数据总线

在这些信号当中，所有的输入信号都直接由AHB和APB总线桥或者总线选择器给出，输出信号PRDATA在APB总线上只有一个从设备时可以直接给总线桥，否则应当连接总线选择器。协议规定的写时序如图 3.13 所示[12]：



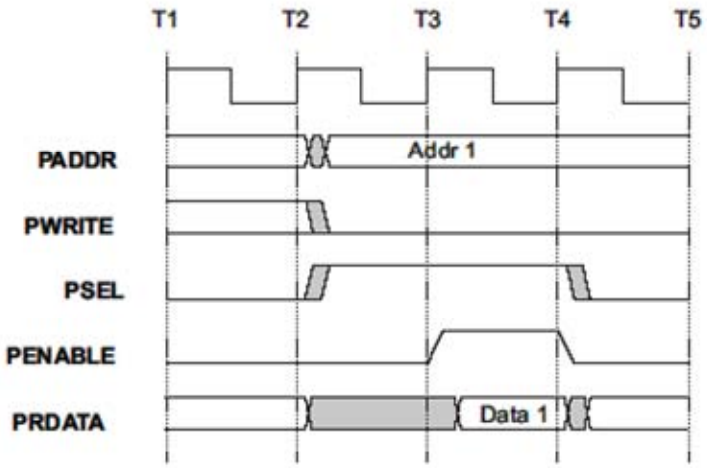
3.13 APB

写操作在 2 个 APB 时钟周期内完成。在第一个时钟周期的上升沿（T2），总线桥给出地址信号（PADDR），写信号（PWRITE），片选信号（PSEL）和写数据（PWDATA）。但是，此时从设备还不能判断有写操作。APB 总线为了支持低功耗设计，因此在不需要使用 APB 总线的时候，写信号保持不变。如果上一个周期是写操作，并且不继续使用 APB 总线，写信号并不改变状态。

在第二个时钟周期，总线桥会给出使能信号（PENABLE）确认写操作，此时从设备就能够确定有写操作发生。在第三个周期（T4），如果不需继续发生写操作或者读操作，片选信号（PSEL）和使能信号（PENABLE）会被总线桥复位，而地址信号（PADDR）和

写信号（PWRITE）会保持不变以减小功耗。如果继续发生总线操作，PSEL 会维持为高，第三个时钟上升沿（T4）直接作为下一个操作的第一时钟周期的上升沿，重复整个过程。

如果发生读操作，其时序如图 3.14 所示[12]：



读操作也分为两个时钟周期，在第一个时钟周期（T2）总线桥给出地址（PADDR）、写信号（PWRITE）和片选信号（PSEL），其中写信号为低，说明为读操作。

在第二个时钟周期（T3），给出使能信号（PENABLE），但是 APB 从设备必须在这个时钟周期内给出读数据，并放到总线上。所以，读操作可以不检测使能信号，直接根据片选和读写信号判断有读操作发生。当然，如果 APB 从设备本身的时钟高出 APB 总线时钟一倍以上，那么还是有机会判断使能信号，并在第二个时钟周期内完成读操作。

如果没有后续总线操作，和写操作一样，总线桥会在第三个时钟周期（T4）复位片选和使能信号，否则，片选信号继续为高，开始下一个总线操作。

3.3.2 接口单元设计

正如“3.1 Baseband 的接口概述”一节中所述，Baseband 和 APB 接口单元的主要目的是使 Baseband 成为 APB 总线上的一个从设备，从而提供 MCU 直接读写 Baseband 参数寄存器的功能。

由于 Baseband 本身工作频率较高，如果将各参数寄存器都集中到接口单元，将增加综合和布局布线的难度。所以，Baseband 的参数寄存器分布在 Baseband 的各个模块当中。因此，我们需要在 Baseband 内部形成一条地址总线，一条数据写总线，写信号和多条数据读总线。各个含有参数寄存器的模块共用内部地址总线，数据写总线和写信号，并拥有独享的数据读总线。

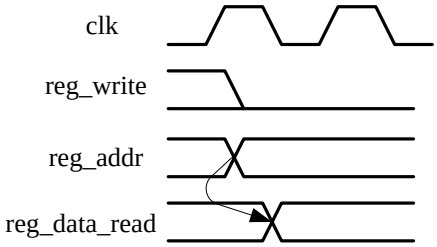
图3.14 APB

表 3.5 Baseband 内部寄存器读写接口信号

信号名称	方向（对应内部模块）	位宽 (比特)	描述
reg_addr	I	6	内部地址总线

reg_data_write	I	16	内部数据写总线
reg_write	I	1	写寄存器信号，高有效
reg_data_read	O	16	内部数据读总线，每模块一条

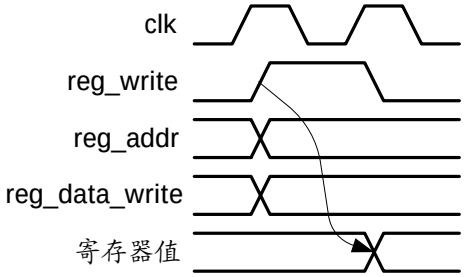
当发生读操作时，APB 接口单元必须给出相应的内部地址。相关模块根据地址信息将相应寄存器数据放到模块专有的数据读总线上。APB 接口单元再根据地址判断应从哪一条数据读总线上获取数据，从而完成读操作。



3.15

在写操作时，APB 单元需要给出相应的地址，将需要写入数据放到数据写总线上，并同时给出一个周期高电平的写信号。相应单元通过判断地址总线的地址信息和写信号，将数据写总线上的数据写入相应的寄存器单元。

因此，APB 接口单元的一个主要任务就是对 APB 总线上的地址进行译码，转变为内部参数寄存器的地址。当需要读操作时，通过地址译码，从正确的数据读总线上获取寄存器值；在需要写操作的时候，将写数据放到内部数据写总线上，并产生写信号。

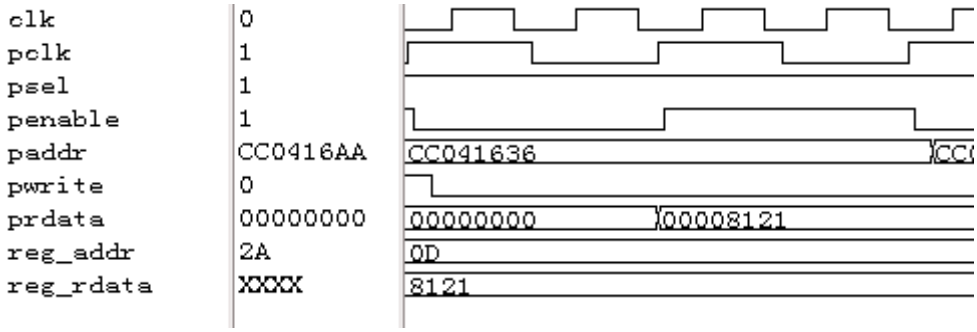


3.16

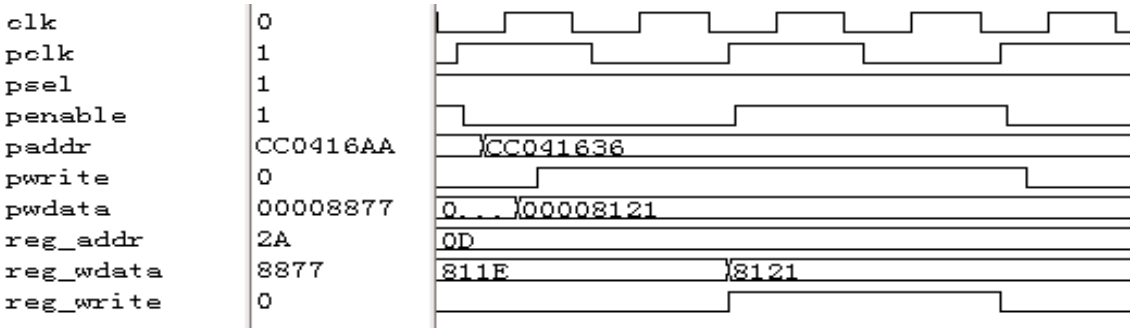
需要注意的是跨时钟域问题。虽然 APB 和 Baseband 的时钟为同源时钟，但是两个时钟的频率不同，因此在综合和布局布线当中还是会出现问题。

尤其在读数据操作上。由于 Baseband 内部为 80MHz 时钟，并且参数寄存器分布在 Baseband 的各个模块。因此，从各个模块到接口单元的连线很长，会造成很大的线路延时。如果不作处理，接口单元直接将读到的数据放到 APB 的 PRDATA 数据总线上，会产生延时过大的综合问题。因此，接口单元要对内部数据写总线、内部数据读总线进行锁存处理，使得总线桥到接口为 40MHz 时钟的时钟域，接口单元到 Baseband 的各个模块为 80MHz 的时钟域。

下面分别给出读操作（图 3.17）和写操作（图 3.18）时 APB 接口单元的时序图：



3.17



3.4 Baseband 和 RF 的接口概述

RF 模块为单独的一块芯片，因此和 RF 的接口实际上就是 Baseband 和电路的接口，为芯片的输入和输出接口。这一部分的接口，主要包括给 RF 的 10 比特位宽的 I 路和 Q 路数据信号，对 RF 芯片自动增益放大器（VGA）的控制逻辑，以及其他的控制逻辑。

这些接口没有任何逻辑设计问题，只有两个细节问题需要说明。

1．RF 接口为芯片的顶层接口，因此需要注意锁存输出

ASIC 芯片和 FPGA 一样，在输入输出端口位置需要用 PAD 进行电平转换和提供电流驱动能力。因此，在 PAD 内部有很大的信号延迟。以 FPGA 为例，标准 TTL 电平的输出端口的延迟为 4ns。

因此，如果芯片内部逻辑时序紧张，而顶层端口上的输出没有寄存，那么输出端口以及 RF 芯片输入端口，电路板上的延时，这三部分的时间都需要算入内部逻辑的延迟时间。这会给综合和布局布线带来很大的困难。并且，电路延迟和 RF 芯片的输入延时为未知参数，如果不加以注意，将会引起芯片工作失常的问题。

另外，FPGA 内部输出 PAD 模块内部带有寄存器。如果输出信号除驱动输出 PAD 之外无其他负载，FPGA 布局布线工具会自动使用 PAD 模块内部的寄存器。这样将进一步减少输出延时。而如果顶层输出没有寄存，并且还被其他模块使用，那么该输出将不能使用 PAD 的内部寄存器，因而增加输出延时的不确定性。

2．数据接口的码制问题

RF 和 Baseband 的接口在 RF 芯片内部使用 ADC 和 DAC 完成。一般 ADC 的输出为补码或者原码，而 DAC 的输入采用补码或者偏移二进制。

在 ASIC 芯片内部，计算往往采用补码的形式，因此和 RF 的接口就需要进行码制转换。实际验证过程中就出现了这样的问题：由于 RF 输入 DAC 的输入为偏移二进制，而 Baseband 输出为原码，造成了调试波形的错误。

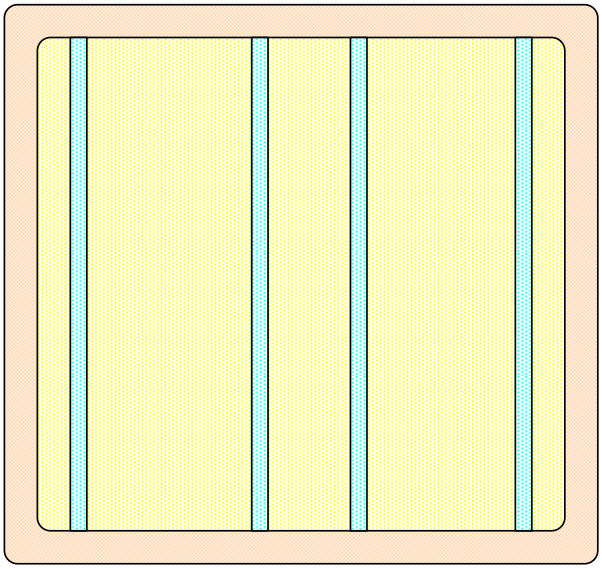
4 FPGA 验证

4.1 FPGA 的结构概述

由于本文所涉及的内容主要为 802.11g 的 FPGA 验证，而不着重讨论 FPGA 的特性，因此本文只针对于验证环境中所使用的 Xilinx Virtex II 系列 FPGA 进行分析。其相关细节可查阅参考文献 18：*Virtex-II Platform FPGA User Guide* 的相关章节。

Virtex II FPGA 的内部结构就像一个棋盘，分布有大量的水平方向和垂直方向的内部总线。这些内部总线将 FPGA 内部的可编程模块连接起来。FPGA 通过设定内部的可编程模块的特性和端口使用来完成不同的逻辑功能。

FPGA（后文所指 FPGA 都为 Xilinx Virtex II 结构）的总体结构如图 4.1 所示。



最外围为 FPGA 的输入输出接口单元（IOB），内部的大部分空间为可编程逻辑块（CLB）。CLB 为 FPGA 内部可编程单元的基本单位，分布在 FPGA 内部的整个区间。其中还包含 4 条特殊资源带：乘法器单元（Mult）和块内存（BRAM）。

其中，IOB 可以提供 FPGA 和外部电路的连接，并且可以提供多种电平和输入输出模式。

Mult 单元可以提供 18 乘 18 的并行乘法运算能力，在最高输出 36 位情况下，工作频率最高可达 75MHz 左右。

BRAM 单元提供了大量的片内 RAM 单元，可以作为单口、双口的 SRAM 使用，也可以作为 ROM 单元加速查表操作。

剩下就是 CLB 了。CLB 是 FPGA 最基本的单元，它提供了基本的组合逻辑、查表、寄存器时序逻辑、分布式内存等等功能。

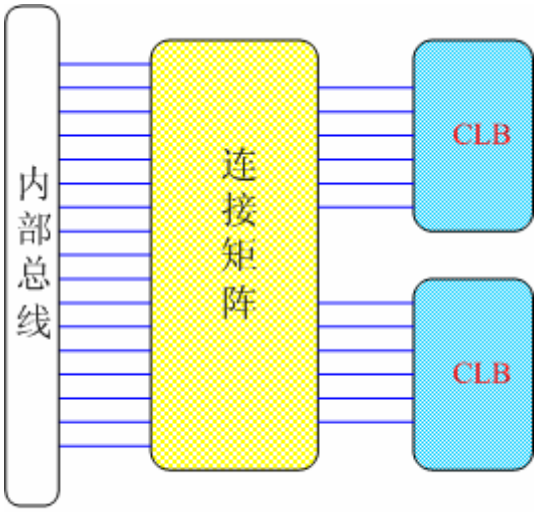


图4.2 CLB和内部总线

从图 4.2 可以看出，CLB 通过连接矩阵（GRM）和内部总线相连，而内部总线贯穿于整个 FPGA 内部，使得单个 CLB 之间可以通过内部总线建立连接。但是，利用内部总线连接毕竟需要大量的线延迟时间，因此，CLB 之间也有快速进位链进行连接，可以很容易地实现串行加法器的功能。

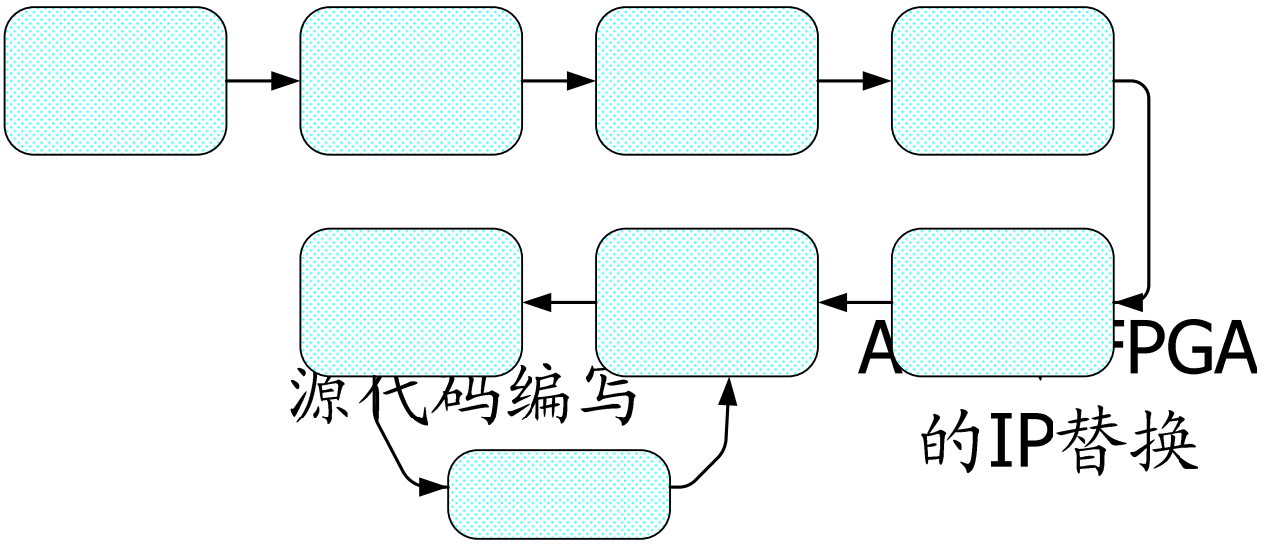
每一个 CLB 由两个功能块（Slice）组成，每个 Slice 内部提供 2 个 4 选 1 的查表单元（LUT），2 个可以配置的触发器，可作为寄存器（Register）或者锁存器（Latch），两个异或门（XOR），和预定制的控制逻辑。

通过 LUT 的使用，可以实现与非逻辑、查表、ROM 等逻辑功能。触发器则可实现时序逻辑，比如状态机、计数器、流水线等等。控制逻辑则控制这些单元的连接，将多个 LUT 并行使用可以构成 16 位、32 位的分布 ROM，利用 CLB 之间的进位链可以构成高速串行加法器，利用控制逻辑和寄存器构成状态机等等。并且，所有这些都是可配置的，通过工具可以选择内部连线的使用，LUT 的工作方式，触发器的类型等等。

综上所述，FPGA是一种半定制的ASIC电路，通过内部可配置的CLB、BRAM等等单元可以构成不同类型的组合逻辑、时序逻辑，从而达到可编程的目的[18]。

4.2 FPGA 的验证流程

FPGA 的验证流程（图 4.3）：



1．源代码编写

验证代码的逻辑功能，首先一步当然是代码的编写。这里所指的代码，可以是 Verilog 或者 VHDL，有时甚至可以是 SystemC。并且，在 FPGA 验证之前，要做好软件仿真的工作。由于 FPGA 实际上为硬件验证，只能通过端口来判断逻辑功能，内部信号对于测量工具来说是不可见的，如果不进行软件仿真，而直接将代码下载到 FPGA 进行验证，将很难确定出现问题的确切位置。软件仿真能提供内部逻辑的波形图，因而能更好地进行前期查错。

这里将软件仿真也放入源代码编写阶段，相对 FPGA 的验证来说，它们都属于代码编写。同时，这里的代码应当是针对于 ASIC 设计的。毕竟整个系统最终要在 ASIC 上实现，因此从一开始便应当用 ASIC 的设计思想编写整个系统。

2．ASIC 和 FPGA 的 IP 替换

正如代码编写阶段所述，源代码应当针对于 ASIC 设计编写。ASIC 设计的很多功能模块，比如乘法单元、加法器、只读存储单元一般使用 Synopsys 的 DesignWare 编写。而 DesignWare 为 ASIC 设计的 IP 模块，难以在 FPGA 上实现。工程所用 FPGA 可以实现 Xilinx 公司的 FPGA 专用 IP 模块，因此需要将 DesignWare 的 IP 进行替换。

后文 4.3 节将详细说明 IP 替换的过程。

3．对应 FPGA 的综合

无论 FPGA 还是 ASIC 代码，都需要经过综合过程。综合将原寄存器传输级（RTL）代码转化为对应不同实际器件的门级网表。但是 FPGA 的实现逻辑决定了它与 ASIC 的不同，很多在 ASIC 综合当中十分有效的方法，在 FPGA 当中可能不适用。比如 FPGA 中的串行加法器要比并行加法器快。

因此，FPGA 需要专门的综合工具，根据 FPGA 的特点对代码进行综合以达到最佳的效果。相关内容在 4.4 节叙述。

4．设定布局布线约束

FPGA 的布局布线约束主要包括 3 个部分：时间约束、位置约束和端口的指定。其中位置约束和端口指定在工具中都叫做面积约束（Area Constraint），但是它们的目的有所不

同，所以把它们从概念上分开。

时间约束一般由综合工具生成，来源于最原始的综合脚本。当然，根据不同情况可以在布局布线阶段加上新的约束。端口约束主要是设定输入输出管脚的位置。FPGA 一般有上百个 I/O 脚，根据 FPGA 测试板的电路，我们必须指定输入输出信号使用的 I/O 脚以便和外部电路连接。而位置约束主要是为了优化布局布线的效果。布局布线软件的随机性往往会导致最终结果也具有随机性。通过制定特殊资源和模块的位置，限制软件的优化能力，能在一定程度上增加工程的可控性。

5．FPGA 布局布线

在布局布线的相关脚本准备完毕之后，就可以利用软件开始布局布线。布局布线的主要工作，就是把综合生成的门级网表内的器件单元定位到 FPGA 内部具体的 Slice 上。而这个过程也是分几步的。

首先是由布局工具确定各个模块在 FPGA 内的大体位置，然后将每个模块内部的基本门级电路确定到 Slice 上，最后布线工具将各个 Slice 连接起来。关于 FPGA 的约束和布局布线部分，详见论文的 4.5 节。

6．生成二进制文件并下载

此过程为工具流程的最后一步，只需要使用工具生成即可。生成的最后终文件为 FPGA 上 FLASH 的映射二进制文件。FPGA 和 CPLD 在这一点上有所不同：CPLD 生成的文件，会在下载的同时将 CPLD 的内部逻辑直接进行指定，从而实现逻辑功能。而 FPGA 仅仅将文件下载到 FLASH Memory 内。在上电时，FPGA 从 FLASH Memory 中取得配置文件，动态地对内部逻辑进行配置，因而可以减少内部电路的复杂度并且提高 FPGA 的使用寿命。

7．FPGA 的实际测量

到了这一环节才能真正对 FPGA 的内部逻辑进行测试。可能需要为测试搭建不同的平台和制作电路板。一般的做法是使用通用的试验板和逻辑分析仪。逻辑分析仪和仿真的波形显示软件类似，但一个是软件仿真波形，一个是抓取的实际波形。

可以通过在 FPGA 软件当中，对已经布局布线的最终网表进行抓取信号，将内部逻辑的波形反映到 FPGA 的管脚上，再通过逻辑分析仪抓取波形进行分析。这样就可以在不用重新布局布线的情况下，对内部信号进行分析。

4.3 FPGA 的 IP 替换

4.3.1 IP 替换的意义

在源代码编写过程当中，一般都倾向使用 Synopsys DesignWare 的 IP 模块。它是 Synopsys 公司针对 ASIC 设计开发的一个功能模块库，在 Synopsys 的 ASIC 设计流程中隐含包括。Synopsys 的综合工具，比如 DC、MC 和 PC，能够直接识别 DesignWare 的模块。

在设计中直接使用DesignWare，归结起来有以下几点好处[19]：

1．增加设计效率

DesignWare 模块一般针对算术运算，标准确定的功能模块，特定资源模块等等。这些模块的编程往往比较复杂，和硬件特性紧密相关但和设计思路没有太多的联系。因此，利

用 DesignWare 的模块，能够节省编程人员的工作，将主要精力放到功能模块的设计上。

2．灵活配置

DesignWare 能够根据具体的配置要求和器件库特性，在综合时自动生成相关结构以达到设计者的要求。而手动编写则不能提供如此高的灵活性，往往需要对应不同的元件库或不同的时序要求进行重写，复杂度提高。

3．加快仿真时间

DesignWare 已经提供了相应的仿真模块。由于硬件电路已经由 Synopsys 多次验证，确保不会出现问题。因而可以在仿真时使用仿真模型，而不使用实际电路的代码，提高仿真的速度，从另一方面节省了设计时间。

4．加快综合时间

使用 DesignWare 的设计可以利用 Module Compiler 在综合阶段加快综合的速度。

5．增加模块的重用性

由于 DesignWare 仅仅对模块的功能做出限定，因此可以在使用模块时通过更改模块的参数而使端口不同但功能相同的模块使用同一个 IP。这样就统一了功能模块的接口，也使得模块重用的困难度降低。

但是，正如 4.2 节所述，DesignWare 是不能够在 FPGA 上直接使用的，FPGA 的验证流程，或者更具体地说，Xilinx 公司的工具并不能直接支持 DesignWare 的使用。作为另一种选择，我们可以在 FPGA 验证阶段使用相应 FPGA 的 IP 模块，比如 Xilinx Core。这样，既能够在源代码当中使用 DesignWare 的模块，继续保持它的优势，也可以在 FPGA 中用 Xilinx Core 替换掉原有 DesignWare 的模块，完成功能验证。

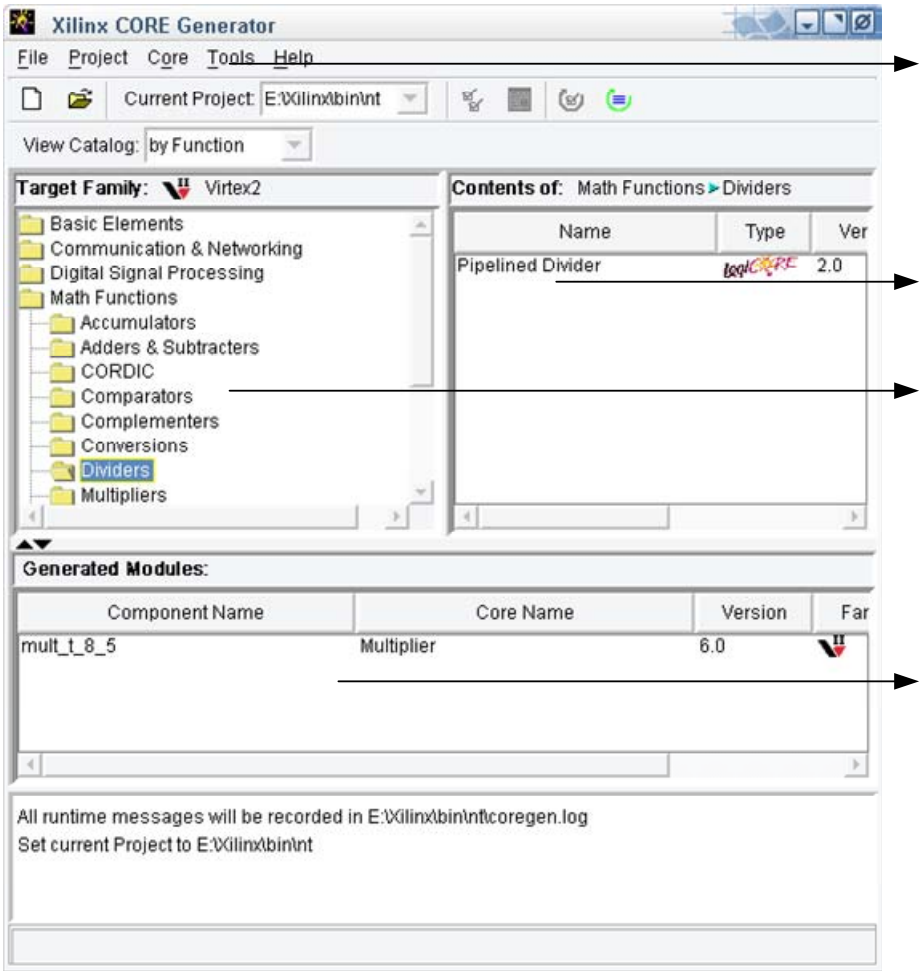
当然，Xilinx Core 和 DesignWare 还是有一些区别的。Xilinx Core 针对 FPGA 进行优化，因此能够更好地利用 FPGA 的特殊资源。但是，Xilinx Core 生成的结果为网表文件，因此不同配置的模块需要生成不同的网表，而不能像 DesignWare 一样通过用参数进行灵活配置。

4.3.2 CoreGen 工具的使用

为了方便 FPGA IP 的生成，Xilinx 集成开发环境 ISE 提供了其专门的 Xilinx Core 生成工具——CoreGen。

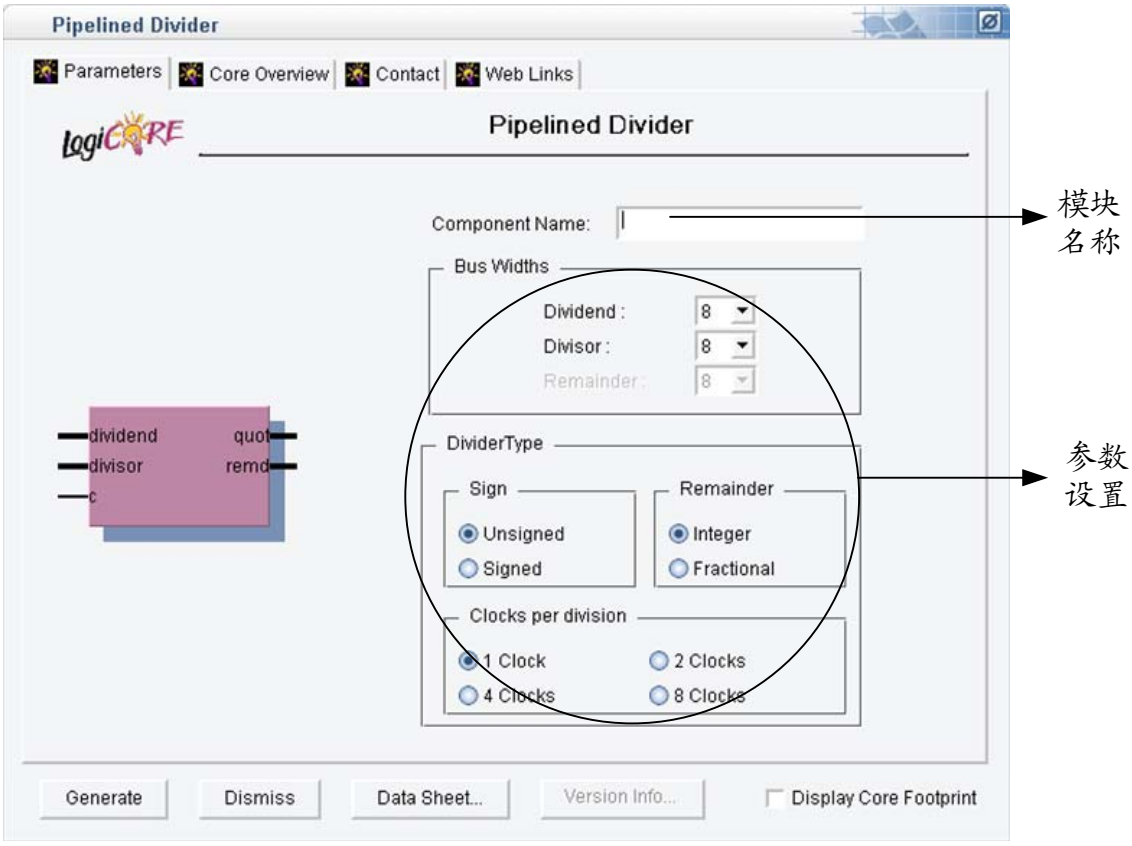
在 UNIX 或者 Linux 下，可以通过直接在 Terminal 中使用 coregen 命令打开 CoreGen，也可以在 ISE 的 Process 窗口下，使用 Create New Source 当中的 IP（CoreGen & Achitecture Wizard）打开 CoreGen 窗口。在 Windows 下，CoreGen 可以在开始菜单中，ISE 的 Accessories 文件夹下找到。

如果使用 Terminal 方式打开，需要首先设置工程参数。对应工程所用 FPGA 选择目标架构（Target Architecture）为 Virtex2，设计入口（Design Entry）为 Verilog，网表总线格式（Netlist Bus Format）为 ISE 方式。如果在 ISE 中打开就不用设置这些参数，CoreGen 将沿用 ISE 工程的参数。



在 CoreGen 的主窗口当中（图 4.4），左边为 IP 目录，一般情况下将会根据功能划分。在每个功能子目录下，会在右边的 IP 模块窗口当中显示出可供选择的 IP 模块。双击其中的模块，就可以新建一个 IP。

下面以流水线的除法器作实例说明：



4.5

双击主窗口中流水线除法器（Pipelined Divider）之后，便会出现图 4.5 所示界面，在器件名称（Component Name）中输入合适的模块名称。窗口的主要部分是模块参数的设置。当参数很多，一页显示不完时，会有 next 按钮，提示有多页。对应所有的参数设置完毕，可以点击生成（Generate）按钮生成 IP 单元。

对于每个模块，CoreGen 都提供相应 PDF 文档对模块的使用和功能进行说明。察看相应文档，可点击 Data Sheet 按钮。模块生成完毕后，在图 4.4 的已完成模块（Generated Module）窗口中会有显示，如果需要更改，用右键点击相应模块，选择重新定义（Recustomize）即可。

针对不同类型的模块，最后 CoreGen 生成的文件也有所不同。表 4.1 给出 CoreGen 常用的输入和输出文件类型及其描述。

表 4.1 CoreGen 常用输入输出文件

文件后缀	输入输出	描述
.coe	输入	FIR 滤波器和内存的参数初始化文件
.asy	输出	提供给 ISE 的图形化信息
.edn	输出	给布局布线工具的网表文件
.mif	输出	FIR 滤波器和内存单元的参数仿真文件
.ngc	输出	二进制的 Xilinx 实现网表
.v	输出	Verilog 形式的模块 wrapper 文件

图　模块生

.veo	输出	Verilog 形式的模块实例化示例文件
.vhd	输出	VHDL 形式的模块 wrapper 文件
.vho	输出	VHDL 形式的模块实例化示例文件
.xco	输出	模块的参数纪录，用于 ISE 综合工具

4.3.3 IP 替换需要注意的问题

在 IP 替换的过程中，我遇到了一些实际的问题，现在把它们总结如下：

1．器件选择应和实际器件一致

在所有的工程代码当中，有一部分是原 802.11b 方式 FPGA 验证的遗留代码，其中也有一些 Xilinx 的 Core，包括乘法器和内存单元。但是，这些 IP 都是对应于 Virtex 结构的。尽管 Virtex 和 VirtexII 的结构和特殊资源相差并不大，但是还是有区别。

例如在 Virtex 结构下内存单元的基本结构为 4kx1 的，而在 VirtexII 结构下基本单元为 16kx1。可见，尽管大部分结构一致，但是 VirtexII 结构内存要比 Virtex 结构密集，容量更大一些。如果直接沿用 Virtex 的 IP，可能造成内存单元的浪费。在此例中，便会引致布局布线软件的一个警告，提示对应器件不一致。

2．注意 Xilinx Core 和 DesignWare 的区别

DesignWare 是一种软的 IP，在 IP 生成阶段，仅仅需要设定 IP 的特性，而并不生成真正的门级网表。在综合阶段，综合软件根据实际时序要求的需要，再生成特定的结构以满足时序要求。但是 Xilinx Core 在生成 IP 的时候已经生成了门级网表文件。而相应的 Verilog 或者 VHDL 代码的作用仅仅是提供仿真功能或者在综合软件当中将 IP 体现为一个黑盒子。因而，Xilinx 的 IP 不可以像 DesignWare 一样通过手动设置参数的方式复用单元。如果两个功能相同的单元有任何一点不一样，比如输入输出宽度不同，就必须为其重新建立 IP 模块。

关于文件的使用。如果在 ISE 中使用，直接加入 xco 文件，ISE 的综合工具 XST 能够直接识别 IP 模块。如果使用 Synopsys 的 FPGA CompilerII 或者 Synplicity 的 Synplify，只能在工程中加入 Verilog 的 wrapper 文件，把 IP 当成黑盒子处理。

在布局布线阶段，直接将生成的网表（edn）文件拷贝到 ISE 工程目录下即可。

3．版本一致性问题

使用 Xilinx Core 应当尽可能保证在逻辑功能上和 DesignWare 完全一致。逻辑功能一致包括以下几个方面的要求：

端口一致。软件并不能保证 Xilinx 生成的 IP 和 DesignWare 的 IP 连端口信号都一样，这几乎是不可能的。但是 FPGA 验证的目的实际上是检测 ASIC 版本的逻辑功能，因此应当保证 Xilinx 的 IP 模块和 DesignWare 的信号名、方向和位宽都一模一样。因此，可以在 Xilinx 的 IP 顶层加上一层 wrapper，更改信号的名称，甚至对 Xilinx IP 没有提供的控制信号编写相应的控制逻辑，对多余的信号要手动作处理去除，确保端口一致。

版本共存。应当在代码中既保存 DesignWare 的版本，又保留 Xilinx Core 的 IP。可以在顶层加上 FPGA 的宏定义。当顶层定义 FPGA 宏时，使用 Xilinx Core 的 IP 版本，当没

有定义的时候，说明为 ASIC 版本，则使用 DesignWare 的 IP。如下所示：

```
`ifdef FPGA
    …… //Xilinx Core IP module
`else
    ……//DesignWare IP module
`endif
```

4．功能一致

由于必须保证 Xilinx Core 和 DesignWare 的端口完全一致，有些控制信号需要手动编写。因此，我们需要对 Xilinx Core 生成的模块和手动编写的控制逻辑进行仿真。利用 Xilinx 提供的仿真环境，和 Synopsys 提供的仿真环境，在相同的输入激励下，比较两个 IP 模块的输出波形，确保功能的一致性。

5．时序要求

DesignWare 会在综合时根据时序要求生成不同的模块来满足，但是 Xilinx Core 一旦生成，IP 内部结构已经确定。换句话说，它能够工作的最高频率也已确定。所以，替换过程不仅要保证功能一样，还要使得 IP 模块在实际工程的时序要求下能够正常工作。

在 802.11g 的验证中便出现了这样的问题。在 FPGA 当中，并行乘法器由特殊资源当中的 18 乘 18 硬乘法器实现，能够运行在 90MHz 左右，但是串行乘法器由普通的 Slice 结构搭建而成，只能运行在 75MHz 到 85MHz 之间，而系统 Baseband 的频率为 80MHz，将会造成串行乘法器不能正常工作的情况。因此，只能使用并行乘法器代替串行乘法器，并使用多周期（multi-cycle）的综合手段使其通过综合要求。

6．特殊模块

对于有些特殊的模块，Xilinx Core 可能没有提供，或者提供的模块达不到要求，只能手动编写。

在 802.11g 验证当中，一个串行除法器需要在 5 个周期内算出 18 位除 10 位的除法运算，而 Xilinx Core 并没有相应的串行除法器可供使用，只有流水线结构的除法器。因而只能更改源代码，或者手动实现一个专用的串行除法器完成验证。

最后的选择还是保持原 ASIC 版本不变。根据具体实现的要求，手动编写一个 11 位除 11 位获得 8 位有效结果的专用除法器，可运行在 100MHz，完成验证任务。

4.4 FPGA 的综合

4.4.1 综合的基本概念

Verilog 或 VHDL 源代码一般都在 RTL 级，即用 always 或者 process 关键字表示寄存器组，用运算符号表示逻辑运算。而真正实现在 ASIC 或者 FPGA 内部的电路由寄存器、与门、非门、选择器等等的基本部件搭建而成。它们需要使用门级代码来描述。

因此，我们需要通过综合这个步骤，将 RTL 级的源代码转化为门级代码，也称为门级网表（netlist），才能真正地在实际器件上实现。

然而，在综合的过程中，我们需要对代码进行优化。从 RTL 级代码转化为网表的方式

是多种多样的，每一种方式都有它们的优缺点。就拿 8 位加法器来说，我们可以用 8 个 1 比特的串行加法器实现，这样只需要很少的几个串行加法器，占用资源很少。但是，这样的加法器延时很长，速度较慢，在高速的场合难以达到速度要求。所以，我们可以用查表的方法实现并行加法器，这样只需要简单的查表，马上可以得到结果。但是，加法器速度虽然快了，表的大小为 64K，所需要使用的元件数量增多了几千倍，大大增加了资源使用。

综合软件便是在这样的极端条件之间做折中的选择，以求既能满足速度要求，又不会占用太大的面积的效果。一般来说，我们通过对综合过程下约束来告诉综合软件如何在资源使用与速度之间进行折中。

约束是综合的基本概念。从本质上来说，约束是对 RTL 级代码所需要达到的速度、面积甚至功耗的一种描述，反映了设计者对实际电路性能的一种要求。其中，速度是最关键的问题，只有满足设计的速度要求，才能保证设计的逻辑功能。

对速度的约束一般包含以下几个方面：

① 对时钟的约束。

时钟频率直接决定了电路的速度。在约束当中，我们需要设定时钟的周期、高电平持续时间、上升时间等等指标。而综合工具，会根据实际器件的延时信息，选择不同的逻辑结构，对 RTL 代码进行转换，以满足约束的要求。

② 对路径的约束。

综合当中的路径分为四种，寄存器的输出端到寄存器的输入端、寄存器的输出端到输出端口、输入端口到寄存器的输入端和输入端口直接到输出端口。

从寄存器的输出端到寄存器的输入端的器件延时，一般可以从对时钟的约束推导出，正常情况下为一个时钟周期。然而，其他的路径并没有直接指定。同时有一些路径的延时需要为多个周期或者少于一个周期，就需要我们单独对这些路径设定约束，以帮助综合工具对这些路径进行正确的转换。另外，还有一些路径为逻辑上不可能的路径。对于这些路径，我们必须告诉综合工具，以防止在这些路径上浪费过多的器件。

③ 对端口的约束。

端口也是设计的重点。芯片与芯片的延时由实际的板级电路决定，而综合工具对这些延时并不能做出正确的估计。因此需要设计者来估计芯片间的延时，并将这些信息交给综合工具，才能保证最后的芯片在端口上也能满足时需要求。

4.4.2 FPGA 综合工具介绍

工程当中主要使用两种综合工具，Xilinx 公司的 XST 和 Synopsys 公司的 FPGA CompilerII。

在初期阶段，对单个单元的综合测试使用 XST 综合工具，但是在测试过程中发现很多比较棘手的问题，虽然经过很长时间测试之后得到解决，最终还是选择使用 Synopsys 的 FPGA CompilerII 完成整个工程的综合工作。

FPGA CompilerII（以后简称 FC2）提供图形界面和脚本方式两种操作方式。但是图形界面不太稳定，并且综合过程中不能离开人机交互的过程，因此实际综合时主要采用脚本方式。和其他使用脚本工作的软件一样，FC2 提供了一个命令集和 shell。FC2 的 shell 为

fc2_shell。

一般 FC2 的综合流程如下图所示：

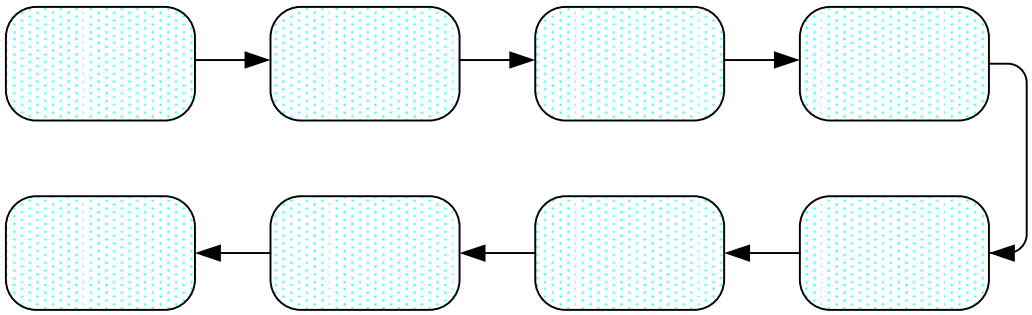


图4.6 FC2

1．建立工程（create_project）

和所有的软件一样，FC2 也用工程来管理项目。因此，在不同综合过程的开始都需要为项目建立工程。

2．添加文件（add_file）

这里的文件就是 RTL 级的原文件，包括 Verilog 或者 VHDL 文件。如果包含 Xilinx Core 的 IP，还应该包括 IP 的 Verilog wrapper 文件。FC2 也可以直接读入网表文件，比如 edn 文件，但是 Xilinx Core 生成的 edn 文件在建立芯片时会出现不能正确连接的错误，一直没有解决，只能将其指定为黑盒处理。

3．文件分析（analyze_file）

在 FC2 里的文件分析，主要是将 RTL 文件转变成软件可以操作的数据。对于 Verilog 或者 VHDL 源代码，文件分析对文件进行语法查错，综合检查并将代码转变为 FC2 的内部数据格式。而对于网表文件，分析阶段仅仅将网表读入软件。

3．建立芯片（create_chip）

芯片是 FC2 里综合和优化的单位。在建立芯片时，我们必须指出芯片所对应的 FPGA 种类、类型和速度。比如 VirtexII6000，相应种类为 VIRTEX2，器件为 2V6000FF1517，速度-4。

芯片建立的主要任务，是将芯片上的各个模块连接起来，但不做任何的优化，形成模块层次结构，为后面的优化作准备。

4．添加约束

添加约束的命令有很多，归结起来分为时钟、最大延迟、优化方法以及资源使用等等几个方面。通过这些约束条件，优化步骤能够尽量地满足工程的时序和功能要求。

5．优化芯片

芯片优化的过程，实际上就是综合工具对应约束条件，对工程进行编译的过程。其优化能力的好坏，直接反映了软件的能力。FC2 的优化能力在大的项目上比 XST 要好一些，至少优化结果比 XST 稳定，但是在局部模块的优化上，XST 还是有它的优势。

6．分析结果

和所有的综合工具一样，在优化完毕之后，需要对优化的结果进行分析，以判断是否

达到约束的要求。不过，FC2 脚本方式的结果分析并不好，而且其分析的结果和最后布局布线的结果有些差别。比如在关键路径上，FC2 的结果有时会和布局布线完成后的关键路径不一致，并且关键路径的时间也偏高。

所以，在实际的综合过程中，FC2 的综合报告仅仅作为结果的参考，一般还是直接到 ISE 里布局布线之后，分析整个流程的优化程度。究其原因，现在的理解是 ISE 的布局布线能力较高，因此对于线路延时过大的关键路径有时有很强的优化能力，使得关键路径的分析结果产生了区别。

7．导出工程

导出工程主要是为后一步布局布线做准备。工程导出有两部分，一部分为综合生成的网表文件，另一部分为综合的约束文件，约束文件使综合工具和布局布线工具在同一套约束下工作，有效地保证了最后布线时序结果满足要求。

4.4.3 综合过程所需要注意的问题

在实际综合过程中，我总结了一些注意事项，现把它们叙述如下：

1．慎用打平

打平（flatten）是综合的一个优化手段。通过打平，可以将所有的内部模块间的端口信号直接变成内部信号，去除了工程模块层次化的结构，有利于综合软件对内部模块的端口进行优化。

一般来说，当时序要求已经十分接近要求，并且很难更改代码和约束条件时，打平是一个比较好的选择。因为优化过程一般只能对模块内部的逻辑进行优化，如果将整个层次化结构变成一个大的模块，优化是针对全局进行的。并且，端口变成了内部信号，优化效果也较好。但是，时序提升是有限度的，如果是时序离最终要求还有距离，打平的作用并不大，而且会有一些不良的后果。

时序紧张的模块，往往是由于线上延时过大。因此可以在布局布线阶段对其施加面积约束，迫使模块分布于一个小范围内，就减少了连接线上的延时。如果在综合时将整个工程打平了，内部就不存在子模块，面积约束没有对象而不能施行。同时由于布局布线软件具有随机性，如果针对于一个大的模块布局，一般会布局较松散，不利于减少连线延时。层次化的模块结构，布局步骤可以根据模块进行位置优化。

因此，在使用打平时，应当全局考虑。

2．约束合理

当综合结果不能满足要求时，有可能的解决办法是采用过约束。简单举例，当时钟周期为 12ns 时，结果不满足要求，定义小于 12ns 的时钟进行综合。

但是，过约束是有它的负面影响的，有时过约束甚至会造成最终结果的性能下降。综合软件根据约束的条件全面优化工程，对于时序约束较松的模块，最终使用的器件延时会相应较长，但是器件数量较少；而对于时序约束紧的模块，会使用大量快速的器件和结构，相应器件数量较多。如果过约束，实际上导致对所有的模块提高时序要求，而往往综合工具在原约束条件下已经使用了最快的器件和结构，过约束也不能提高性能，而对其他约束本身不高的模块提高了时序要求，增加了器件的数量和面积。器件数量和面积的增加，会

给布局布线带来压力，致使结果有可能更糟。

3．脚本的使用

灵活使用脚本是使用命令行方式运行的软件的重要能力。

脚本就好像 DOS 下的批命令，完成一个工程的脚本编写，软件就能完全根据脚本运行，而不用在运行过程中不断输入命令，实现自动化运行。

并且，可以根据不同的目的编写不同的脚本。比如添加文件可以单独一个脚本，约束文件也可以单独成为脚本。这样，如果文件列表发生变化，只需要更改添加文件的脚本即可，不用重新编写或更改其他已经编写的脚本。

并且，现在大多数的综合软件都使用脚本方式。综合阶段并不需要了解电路的结构，只关心几种标准路径上的延时。所以，学会使用脚本是快速掌握综合工具的前提条件。

4.5 FPGA 的布局布线

4.5.1 布局布线的基本概念

布局布线的主要工作是将综合生成的网表当中的基本元件放置到 FPGA 或者芯片的特定位置，以满足相应的时序要求。具体可以分成以下三个步骤：

1．布局（map）

在 ASIC 的流程中，布局也被称为 floorplan。其主要工作包括以下几点：

确定 IO，即 IO PAD 的位置和类型。

确定各种 IP 单元的位置，其中包括模拟模块（PLL、AD/DA）、内存单元（SRAM、DRAM）、IP 核（ARM、DSP）、等等。

确定供电的电源环路（Power Ring）和 Strape（纵向的特殊供电结构）的位置，宽度和密度。

规定各种布局约束，包括保护区间（Guard Ring 和 Fence），在分层结构的设计中还包括子模块的位置、大小等等。

针对于 FPGA 的布局就没有这么复杂，所有的工作可以由布局布线软件自动完成。其中包括 IO 的指定，根据 IO 指定的结果确定各个子模块的位置，并做优化。确定特殊资源，时钟资源的位置等等。

2．放置元件（place）

放置元件的目的就是按照布局的结果，将综合生成网表当中所有的器件指定到 FPGA 内部具体的 Slice 上。

由于在上一步已经将各个子模块的位置和面积确定，因此在放置元件阶段，只需要在子模块内部对各个器件的位置进行优化，当然也包括之模块间的优化，但已经不需要在全局考虑器件放置的问题。

3．连线（route）

在各个器件的位置确定之后，剩下的工作只需要将各个 Slice 使用的端口连接起来。连线也需要分两步进行。

首先应当是时钟的连线。正如综合时所说，布局布线工具需要尽量保证时钟的延时和

clock skew 尽量的小。因此，时钟线路应当在其他信号线之前连接，以确保时钟路径首先能够达到综合要求。在 FPGA 当中，当时钟使用时钟专用资源（VirtexII 器件中的 clock buffer 和铜线布线层）时，时钟和普通布线资源是没有冲突的。如果时钟使用普通布线资源，时钟会自动使用长线资源以减小 clock skew。

在时钟之后，才是其他信号线的连接。当然，针对于时序要求较高的模块，连线延时相对较小，而时序约束较低的模块，连线延时相对较大，以节省连线资源。

4.5.2 ISE 布局布线软件介绍

ISE为Xilinx公司为其FPGA特别编写的开发工具环境，包括从综合到布局布线，直到最终生成二进制文件等等一系列的软件。工程所用VirtexII系列FPGA的布局布线工作，都是使用ISE提供的布局布线工具完成[20]。

1．工程管理器（Project Navigator）

工程管理器是 ISE 所有基层工具的连接纽带。通过使用工程管理器，可以很容易的创建、组织和管理工程。

2．约束编辑器（Constraint Editor）

ISE 中使用 UCF 文件保存对布局布线的约束。UCF 文件有它自己专门的语法和格式。为了方便用户编辑 UCF 文件，ISE 提供了约束编辑器以降低 UCF 文件编写的难度。脚本编辑器使用交互式的图形界面，利用按键和图表的方式。用户只需选择相应的约束参数和信号，而最后的 UCF 文件由软件自动生成。

3．引脚与区域约束编辑器（PACE）

在 ISE 当中，对 IO 的指定和子模块位置与面积的约束也是保存在 UCF 文件当中。约束编辑器主要针对时序约束的编写，而 PACE 的主要作用是帮助用户编写对引脚和子模块位置与面积的约束。

在 PACE 当中，ISE 也提供了图形界面。用户只需要将对应的特殊资源，IO 接口和子模块拖放到 FPGA 相应的位置，软件便会完成对 UCF 文件的编写。

4．时序分析器（Timing Analyzer）

时序分析器是 ISE 当中分析时序的主要器件。它能够根据最终布局布线工具生成的网表分析每一条路径的延时，并根据约束条件判断是否存在路径的延时没有达到约束要求，也就是出现错误。

它的功能十分强大，能够根据网表动态分析时序。因此，对网表做细微修改而没有重新布局布线时，只需要重新运行时序分析器就能够得到更改后网表的时序分析结果。

同时，它能够根据约束对错误进行分类，列出错误最严重的路径，即针对每个约束的关键路径。并且能够动态根据用户要求分析局部电路的时序。

5．FPGA 底层编辑器（FPGA Editor）

FPGA 底层编辑器是 ISE 提供的手工布局布线设计工具。通过这个工具，用户可以编辑或查看可配置逻辑功能块（CLB）、输入输出模块（IOB）等 FPGA 内部所有的功能单元。该工具是 ISE 所有工具当中最底层，也是提供最大的优化设计密度和性能的工具。

如果布局布线的最终结果出现几个错误，并且错误不是非常严重的时候，往往可以通

过 FPGA 底层编辑器手动修改，不需要重新布局布线，大大节省了布局布线的时间。同时，当需要临时抓取内部信号时，也可以通过该工具，直接找到相应内部信号，抓取出来即可。

4.5.3 FPGA 布局布线的注意事项

和前几步一样，布局布线阶段也有很多值得注意的问题。

1．脚本检查

尽管 FPGA Compiler 和 ISE 都是比较成熟的软件，但是在实际工程中也不能对它们产生过多的依赖。在每次 ISE 布局布线开始之前，都必须花一点时间检查 FPGA Compiler 生成的约束脚本文件，有时脚本文件会出错或者存在不合适的约束，需要手动修改。

主要可能存在几个可能需要修改的地方：

信号名出错。在特殊情况下，FPGA Compiler 会把时钟信号的信号名弄错，多加了一个空格，如果不仔细检查，会在布局布线时导致出错，出现信号名找不到。

约束冗余。在多时钟域情况下，FPGA Compiler 有时会把约束重复两遍，但是其中一个约束的时钟周期值可能是其他时钟域的周期值，而出现错误。应当在这种情况下手动去除该条约束。

约束不合理。一般情况下，FPGA Compiler 都会自动推断出几个 OFFSET 类型的约束，但是该类型的约束有时过严，导致布局布线工具过多地考虑这些约束，并做优化，使得总体优化结果不好。因此，对这类约束应当适当分析，以判断其是否合理。对于不合理的约束，要手动更改。

2．慎用高级优化选项

无论在哪一阶段，一定要注意约束和选项的合理性，高级选项有时会带来好的优化效果，但有时也会有负面影响。

布局布线的高级选项，往往都是在已有的优化步骤上，增加优化的分析范围或者增加分析步骤。然而这些增加的分析范围或者分析步骤并不一定能解决实际问题，而且往往会浪费大量的时间。

比如在布局选项（Map Properties）中，有一项为“按时序要求进行封装和放置元件（Perform Timing-Driven Packing and Placement）”。也就是根据时序的具体要求，优先对关键路径进行布局。实际经验说明，使用该优化选项，并不能明显提高最后的优化结果，而增加的分析时间往往达到一个小时以上。经过仔细分析，增加对模块位置的面积约束，往往要比让软件自己去分析关键路径效果要好，至少在本项目的实例中，该高级选项是失效的。

同时，还有一些选项，比如优化力度（Effort Level），寄存器重定时等等，都应当根据实际情况，经过分析之后适当使用。

3．仔细检查时钟

实际经验说明，如果单个模块综合都已达到时序要求，但是整体综合、布局布线之后就不能达到时序要求，原因往往在时钟上。很可能出现的问题有以下两点：

时钟没有使用时钟资源：

内部时钟是需要软件自动推断的，如果存在约束被忽略或者代码编写不当的问题，很

可能引发软件时钟推断错误，导致时钟没有放上时钟资源。在这样的情况下，为了满足时序要求，ISE 会使用长线资源来作为时钟。尽管这样的做法，在小的设计时不会对最终时序结果造成很大的影响，但是对于大一些设计，比如资源使用达到 40%以上时，就会大大增加对连线的压力，从而影响最终的时序。

时钟资源为铜线布线层，和普通的布线资源分开，相互间没有影响。如果时钟没有使用时钟资源，而使用长线，就有可能影响普通信号的布线。长线资源为普通布线资源当中的高速布线资源。尽管对时钟来说，长线也可以做到相对较小的 clock skew。但是，长线使用了普通布线资源，减少了普通信号布线资源的可用范围，从而影响了时序。

出现这种错误的原因往往是时钟 BUFG 被优化掉或者没有插入 BUFG，可以通过手动实例化 BUFG 解决。

时钟的 clock skew 过大：

如果时钟没有放上时钟资源，往往 clock skew 会比较大，这时通过实例化 BUFG 能够解决。但是，即使在时钟放上时钟资源的情况下，也可能出现 clock skew 较大的情况。这可能有几种原因：时钟域不平衡，需要通过更改设计纠正。时钟生成单元分散，导致各时钟间延时不一样，可以通过对时钟生成单元添加位置约束纠正。时钟域过多，导致主从时钟分配不当，FPGA 内部只有 8 个主时钟域，需要对时钟域进行分析之后仔细调整时钟域分布。

4．仔细分析报告

这里的报告主要是布局布线完成后静态时序报告（Post Place and Route Static Timing Report，简称 Post-PAR rpt.）。分析报告是综合、布局布线最主要的能力。流程过程中大部分的有用信息，都可以从报告中获得。

Post-PAR rpt.为布局布线完成后的时序报告，也是最接近真实情况的时序报告。对它进行分析，主要注意以下几个关键字的信息：

Slack 分析：

着重分析 Slack 为负，也就是出现错误的路径。如果错误较少，查看这些路径是否存在共同点。

如果所有的路径都处在同一个模块当中，并且它们的连线线延时都大于器件延时，那么就说明这些路径过于分散，可以对这个子模块施加面积约束来改进时序。

如果这些路径都是从有限的几个寄存器开始，或者终结于有限的几个寄存器。那么，在一般情况下，继续分析结束于这几个寄存器或者从这几个寄存器出发的连线延时，一般较长。这样，就可以使用 FPGA 底层编辑器手动修改寄存器位置，减少连线距离，改善时序。

如果错误最严重的路径为 false path，或者逻辑不可能路径，那么说明布局布线工具在该路径上耗费过多优化能力，这是不必要的。可以通过修改约束条件，增加 false path 约束改进时序结果。

Clock Path Delay 分析：

如果出现 Clock Path Delay，说明时钟路径上的延时较大，就要关心 clock skew 对延时的影响了。Clock skew 的出现，往往带来 hold 类型的错误。Hold 错误比 setup 错误更加严

重，setup 错误出现时，可以通过降低整个系统的时钟频率来实现，而出现 hold 错误时，整个电路的逻辑功能将不能实现。

而 clock skew 经常出现在跨时钟域之间。其原因一般有两点：

时钟没有放上时钟资源。此时最明显的特征就是报告中会出现时钟树的结构，同一个时钟会出现时钟子路径标号，说明 ISE 使用长线资源打造了时钟树。解决办法是实例化 BUFG，迫使时钟使用时钟资源。

时钟生成单元分散。这是由于时钟生成单元没有集中分布，而是分散到 FPGA 的各个角落。其最明显的特征就是 skew 出现在两个时钟之间。对时钟生成单元添加面积约束，并仔细确定时钟 BUFG 位置，而不让 ISE 自己去指定，在一定程度上能消除这种影响。

结　论

本文在介绍了 802.11g 网卡、802.11g 针对于 OFDM 调制方式的 Baseband 结构和功能的基础上，着重讨论了 Baseband 和网卡其他模块间的接口，整个 Baseband 从 RTL 代码到最后 FPGA 实现的过程及问题分析。

论文的主要成果可总结如下：

① 论文完整地介绍了 Baseband 和 MAC 层、APB 总线和 RF 芯片的接口设计。

针对于 Baseband 和 MAC 层接口，论文仔细分析了 PLCP 信头的组装与去除、跨时钟域信号的处理、FIFO 深度的估计、异常处理及抓沿电路、状态机的构成等等实际问题。并就这些问题，提供了有效的解决方案。

在 Baseband 与 APB 总线接口部分，论文着重讨论了 Baseband 作为 APB 总线从设备的意义，APB 总线时序要求与接口实现等等问题。

② 完整描述了针对于 FPGA，从 RTL 级代码到最后的器件实现的整个流程。

详细讨论了 IP 替换的意义、方法、需要注意的问题，以及在 802.11g 的 Baseband IP 替换过程中遇到的实际问题，及其解决方案。

对于 FPGA 的综合与布局布线，论文详细讨论了它们的目的、概念和流程。并结合实际使用的经验，提出了一些需要注意的问题，并给与解释。

但是，对于整个 802.11g 网卡设计的总体项目来说，论文所作的工作还是很有限的。在论文完成之后，还有以下几个问题需要进一步研究：

① 在满足系统逻辑功能的前提下，进一步复用系统的功能模块，减少系统的总体器件数量，降低成本。

② 进一步研究门控时钟的用法，减少门控时钟对系统时序的影响，并从系统设计上达到低功耗的要求。

③ 优化布局布线流程，进一步探讨布局布线过程中遇到的问题，减少不合理约束、软件的随机性、器件结构的特殊性对工程实现的影响。

致　谢

本论文从定题到最后论文的完成，自始至终都得到了多位老师和同学的无私指导与帮助，如果没有他们，很难有这篇论文的完成。

首先当然应当感谢我的指导老师方穗明副教授。在论文的方向选择、论文的结构和重点等等问题上，方老师都对我进行了有效的指导。并且在我遇到困难和挫折时给与了我鼓励与支持，在此对她表示感谢！

同时也要感谢实验室的聂红儿老师、周春良老师和邹阳同学，帮助我解决了在 APB 总线、时钟树平衡等等难点上的困难。感谢赵德林老师对我的指导，使我更准确地理解了综合的概念。感谢雄晨荣老师在 802.11g 协议和验证过程中对我的帮助。以及孟宪春同学，他在 FPGA 方面的经验，帮助我解决了很多在布局布线和综合方面的困难。

最后，也要感谢我的父母，宋才发教授和夏桂霞副教授。没有他们这二十几年来对我的培养，也就没有我今天的成就。

在此，我真心地向他们对我的帮助表示感谢！

参考文献

1 郭仕刚, 狄强. WLAN 的标准与发展. 世界电信, 2003, 7: 40~42

2 Matthew S.Gast. *802.11 Wireless Networks: The Definitive Guide*. O'REILLY, 2002, 4: 203~218

3 韩旭东, 张春业, 曹建海. IEEE 802.11g 研究综述. 标准与技术追踪, 2004, 1: 24~29

4 Andrew S. Tanenbaum. 计算机网络（第三版）. 北京: 清华大学出版社, 1998, 7: 259~263

5 *IEEE std 802.11g Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band.* The Institute of Electrical and Electronics Engineers, Inc., 2003, 6

6 刘乃安. 无线局域网（WLAN）——原理、技术与应用. 西安: 西安电子科技大学出版社, 2004, 4: 29~225

7 佟学俭, 罗涛. OFDM 移动通讯技术原理与应用. 北京: 人民邮电出版社, 2003, 6: 15~47

8 *IEEE Std 802.11a-1999: High-speed Physical Layer in the 5 GHz Band*. The Institute of Electrical and Electronics Engineers, Inc., 1999/AMD(1): 2000

9 桑林, 郝建军, 刘丹阳. 数字通信. 北京: 北京邮电大学出版社, 2002, 2: 123~132, 277~290

10 王秉钧, 现代通信系统原理. 天津: 天津大学出版社, 1999, 8: 314~344

11 *DesignWare Building Block IP: Sequential Divider*. Synopsys, Inc., 2003, 6

12 *AMBA™ Specification (Rev 2.0)*. ARM limited , 1999, 5: 167~188

13 中国人民共和国国家标准 信息技术 系统间远程通讯和信息交换 局域网和城域网 特定要求 第 11 部分：无线局域网媒体访问控制和物理层规范. 北京: 中国标准出版社, 2003, 8

14 *IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. The Institute of Electrical and Electronics Engineers, Inc., 1999, 8

15 James R. Armstrong, F. Gail Gray. VHDL 设计：表示和综合（第二版）. 北京: 机械工业出版社, 2002, 4

16 Mark Zwolinski. *Digital system design with VHDL*. Prentice Hall, 2003, 11

17 Michael D. Ciletti. *Advanced Digital Design with the Verilog HDL*. 北京: 电子工业出版社, 2004, 2

18 *Virtex-II Platform FPGA User Guide (v1.9)*. Xilinx, Inc., 2004, 8

19 *DesignWare Building Block IP Introduction*. Synopsys, Inc., 2003, 6

20 EDA 先锋工作室. FPGA/CPLD 设计工具：Xilinx ISE 5.x 使用详解. 北京: 人民邮电出版社, 2003, 6: 19~84, 121~262

21 R. Jacob Baker. *CMOS circuit design, layout, and simulation.* 北京: 机械工业出版社, 2003, 6: 373~422

22 陆平, 郑增钰, 任俊彦. 延迟锁定环(DLL)及其应用. 固体电子学研究与进展, 2005,

2(1): 81~88

23 Li Xia, Sun Hui, Zhang Qianling. *A Novel Divider Based on Dual Bit Algorithm.* 半导体学报, 2004, 6: 645~649

附录 缩写名词表

ADC	模数转换器 Analog Digital Converter
AHB	高级高性能总线（ARM 的总线标准） Advanced High-performance Bus
AMBA	高级微控制器总线结构（ARM 的总线标准）Advanced Microcontroller Bus Architecture
AP	接入点 Access Point
APB	高级外围设备总线（ARM 的总线标准）Advanced Peripheral Bus
ARM	ARM 公司，英国的一家开发 MCU 的公司
ASIC	专用集成电路 Application Specific Integrated Circuit
BRAM	块内存 Block Random Access Memory
CCK	补偿编码键控 Complementary Code Keying
CLB	可编程逻辑块 Configurable Logic Block
CPLD	复杂可编程逻辑器件 Complex Programmable Logic Device
DAC	数模转换器 Digital Analog Converter
DBPSK	差分二进制相移键控 Differential Binary Phase Shift Keying
DC	①直流分量 Direct Current ②Synopsys 的综合工具 Design Compiler
DCM	数字时钟管理单元 Digital Clock Manager
DLL	延迟锁定电路 Delay-Locked Loop
DQPSK	差分正交相移键控 Differential Quadrature Phase Shift Keying
DSSS	直接序列扩频 Direct Sequence Spread Spectrum
DSP	数字信号处理器 Digital Signal Processor
DW	DesignWare
FC2	Synopsys 的 FPGA 综合工具 FPGA CompilerII
FFT	快速傅里叶变换 Fast Fourier Transform Algorithm
FHSS	跳频扩频 Frequency-hopping Spread Spectrum
FIR	有限冲激响应 Finite Impulse Response
FPGA	现场可编程门阵列 Field Programmable Gate Array
GFSK	高斯频移键控 Gaussian Frequency Shift Keying
GI	保护间隔 Guard Interval
GRM	通用布线矩阵阵列 General Route Matrix
HR-DSSS	高速直接序列扩频 High-Rate Direct Sequence Spread Spectrum
IEEE	电气和电子工程师学会 The Institute of Electrical and Electronics Engineers
IFFT	快速傅里叶逆变换 Inverse Fast Fourier Transform
IOB	输入输出接口单元 Input/Output Block

IP	知识产权 Intellectual Property
ISE	Xilinx 公司 FPGA 集成开发环境 Integrated Software Environment
ISI	码间干扰 Inter-Symbol Interference
ISM	工业、科学及医药设备频段 Industrial, Scientific, and Medical
LLC	逻辑链路控制 Logical Link Control
LUT	查表单元 Look-Up Table
MAC	媒体访问控制 Medium Access Control
MC	Synopsys 的综合工具 Module Compiler
MCU	微控制器 Microprocessor Control Unit
Mbps	兆比特每秒 Mega bit per second
OFDM	正交频分复用 Orthogonal Frequency Division Multiplexing
OSI	开放式系统互联参考模型 Open System Interconnect Reference Model
PACE	引脚与区域约束编辑器 Pinout and Area Constraints Editor
PAR	峰值平均功率比 Peak Average Ratio
PC	Synopsys 的综合工具 Physical Compiler
PBCC	包二进制卷积编码 Packet Binary Convolutional Code
PCI	外围设备互连标准 Peripheral Component Interconnection
PDF	Adobe 便携式文档格式 Adobe Portable Document Format
PHY	物理层 Physical (layer)
PLCP	物理层汇聚协议 Physical Layer Convergence Protocol
PLL	锁相环 Phase Locked Loop
QAM	正交幅度调制 Quadrature Amplitude Modulation
RAM	随机存储器 Random Access Memory
RF	射频 Radio Frequency
ROM	只读存储器 Read Only Memory
RTL	寄存器传输级 Register Transfer Level
SOC	片上系统 System On a Chip
USB	通用串行总线 Universal Serial Bus
VGA	可变增益放大器 Variable-Gain Amplifier
VHDL	硬件描述语言中的一种，IEEE 标准硬件语言 Very-High-Speed Integrated Circuit HardwareDescription Language
VLSI	超大规模集成电路 Very Large Scale Integrated Circuit
XST	Xilinx 的 FPGA 综合工具 Xilinx Synthesis Technology