IEEE802.11 系列标准中的 OFDM 技术的分析与研究

摘 要

正交频分复用(OFDM)技术是当前移动通信领域的一项关键技术，OFDM 技术具有很强的抗符号间干扰、抗多径衰落能力，适合在无线信道中传输高速的数据业务，因而倍受关注。OFDM 采用了正交多载波技术，频谱利用率很高，目前已被应用于无线局域网(WLAN)、数字音频广播(DAB)、数字视频广播(DVB)系统中，并且成为第四代移动通信系统的核心技术。

本文首先研究 OFDM 理论，详细介绍了基于 OFDM 的无线局域网系列标准 IEEE802.11b/a/g 的介质访问控制层和物理层的操作及帧结构。在分析无线局域网的基本原理的基础上，采用仿真软件 SystemView 对 OFDM 在无线局域网中的工作过程进行了系统仿真和分析，通过仿真比较了两种信道编码的性能，并做了相应的误码率分析，得到了有用的结论，建议采用卷积编码。为便于今后硬件实现，利用 M-Link 编程搭建实现了 OFDM 模块功能。然后通过 MATLAB 软件对 OFDM 系统进行了仿真分析，研究了不同调制映射，保护间隔，载波数和信道对 OFDM 系统的影响，并得出一些结论。由于 FFT 算法在 OFDM 中占有重要的地位，所以在本文的最后用 VHDL 语言对 OFDM 系统中的核心部件 FFT 处理器进行了 IP 模块化设计。该模块不仅可以用在无线局域网系统中，同时也可以应用于其他采用 OFDM 技术的传输系统中。

关键词：正交频分复用，无线局域网，IEEE802.11/b/a/g，快速傅立叶变换

ANALYSIS AND RESEARCH OF OFDM TECHNOLOGY IN IEEE802.11 SERIAL STANDARDS

ABSTRACT

Orthogonal Frequency Division Multiplexing is one of key technologies in mobile communication fields. It is suitable for high rate data transmission in wireless channels because it can combat Inter-Symbol interference and multi-path fading efficiently. Now it has become an attractive technology. OFDM is a multiple carries technology, supports high rate data transmission with high spectrum efficiency. At present, OFDM technology has been used in many fields, such as wireless local area network (WLAN), digital audio broadcasting (DAB), digital video broadcasting(DVB) systems and so on. It is very promising to be the core technology of the fourth-generation mobile communications.

This thesis studies the OFDM theory, and thoroughly introduces the operations and the frame structure of the Media Access Control and Physical Layer by the Wireless Local Area Network (WLAN) serial standard IEEE802.11 b/a/g which is based on OFDM technology.

Having explained the WLAN fundamental principles, this research applies the SystemView Simulation software to simulate and analyze the acting process of OFDM in WLAN. By the system simulation, the research makes comparisons of the functions of two types channel coding, and analyzes the corresponding BER, thus it reaches some available conclusions and presents suggestions to use convolutional codes. To be convenient for hardware realization, this research has made programs to realize the OFDM token.

Then, it applies the MATLAB software to simulate and analyze the OFDM system, and studies the influences of mapping, guard interval, sub-carries and channel on the system, in this way makes some useful conclusions.

Finally, considering the importance of FFT in OFDM, this thesis uses VHDL to make IP model design of the FFT processor, which is the key component of the OFDM system. This model can be applied not only to the WLAN system, but also to the other transmission system with OFDM technology.

KEY WORDS: OFDM, WLAN, IEEE802.11b/a/g, FFT

声　　明

本人郑重声明：所呈交的学位论文，是本人在指导教师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写过的科研成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

论文作者签名：何玉　　日期：2006.5.24

关于学位论文使用权的说明

本人完全了解太原理工大学有关保管、使用学位论文的规定，其中包括：①学校有权保管、并向有关部门送交学位论文的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存学位论文；③学校可允许学位论文被查阅或借阅；④学校可以学术交流为目的，复制赠送和交换学位论文；⑤学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

签　　名：何玉　　日期：2006.5.24

导师签名：[illegible]　　日期：2006.5.24

第一章 绪 论

1.1 引言

随着以通信技术和计算机技术为标志的高科技的发展，人们的生活发生了日新月异的巨大变化。人与人之间的信息传递越来越密切，方式也越来越多样化，人类的社会已经进入了“信息社会”。在各种信息技术中，信息的传输起着支撑作用，由于人类社会生活对通信的需求越来越高，世界各国都在致力于现代通信技术的研发。

现代移动通信技术在融汇了当代电子技术、计算机技术、无线通信、有线通信和网络技术的基础上，更是得到了突飞猛进地发展。目前，移动通信已从模拟通信发展到了数字通信阶段，并且朝着个人通信更高阶段迈进。未来的移动通信目标是能在任何时间、任何地点、向任何个人提供可靠的通信服务。同时，计算机网络已经从有线网络向无线网络发展，并且要求能够随时随处自由接入 Internet，享受更多的业务，并确保信息传输安全可靠。在通信技术和计算机技术的驱动下，无线局域网(WLAN, Wireless Local Area Network)得到了迅速的发展，并且成为新一代高速无线接入网络。

1.2 WLAN 技术目前研究现状

无线局域网是计算机网络和无线通信技术相结合的产物，是非常便利的数据传输系统[1]。它利用射频技术取代旧式的有线局域网络，使得各个局域网络的终端通过无线信道互相传输数据。随着通信事业的高速发展，WLAN 技术进入了一个新的天地，由于 WLAN 具有安装便捷、使用灵活、经济节约、易于扩展、应用范围广泛的特点，因而得到了迅速的发展[2]。

目前比较流行的无线接入技术有 IEEE 802.11 标准、Bluetooth 、HomeRF 红外数据传输标准(Infrared Data Association , IrDA)。作为无线局域网协议族中最成熟

的 IEEE802.11b 主要是工作在 2.4GHz 频段，频段范围 2.4GHz-2.483GHz，物理层的调制方式为 CCK(补码键控)[3]。IEEE802.11b 规定的是动态速率，允许数据速率根据噪音状况进行自动调整，这就意味着 IEEE802.11b 设备在噪声的条件下将以比正常速率 11 Mbps 更低的速率，如 1Mbps，2Mbps，5.5Mbps 等多种速率进行传输。多速率机制的 MAC(介质接入控制)确保当工作站之间距离过长、干扰太大或信噪比低于某个门限时，传输速率能够从 11Mbps 自动降到 5. 5Mbps，或者根据直接序列扩频技术调整到 2Mbps 和 1Mbps。

IEEE802.11 a 扩充了标准的物理层，规定该层使用 5GHz 的频带，频段范围为：5.15-5.25，5.25-5.35 及 5.725-5.825 GHz 采用 OFDM(正交频分复用)技术来传输数据，其物理层的吞吐量分别为 6, 12, 18, 24, 36, 48, 54Mbps[4]。高端的 5.8GHz 频段，由于输出功率高，适于建筑物之间或室外环境的无线应用；低端的 5.2GHz 和中部的 5.3GHz 频段特别适合于建筑物内的应用，对于 5.2GHz 的设备必须使用集成天线。

另外 IEEE 在 2003 年 6 月新推出的 IEEE802.11g 标准，采用的调制方式有两种，包括 IEEE802.11a 中采用的 OFDM(正交频分复用)与 IEEE802.11b 中采用的 CCK(补码键控)[5]。IEEE802.11g 其实是一种混合标准，它既能适应传统的 IEEE802.11b 标准，在 2.4GHz 频率下提供每秒 11Mbps 数据传输率，也符合 IEEE802.11a 标准在 5GHz 频率下提供 54Mbit/s 数据传输率。IEEE802.11g 标准不但能够兼容 IEEE802.11b，保护现有的设备投资，而且还可以达到 54Mbit/s 的速率。对于 IEEE802.11g 标准来说，其核心技术还是 IEEE802.11 b/IEEE802.11 a 中的技术。

1.3 OFDM 技术及其发展

正交频分复用是一种多载波数字调制技术，它的英文全称是：Orthogonal Frequency Division Multiplexing，简写为 OFDM。OFDM 最早起源于 20 世纪 50 年代中期，在 60 年代就已经形成了使用并行数据传输和频分复用的概念。OFDM 技术的应用可以追溯到上世纪 60 年代，主要用于军用高频通信系统例如 KINEPLEX，ANDEFT 和 KATHRYN 但是一个 OFDM 系统的结构非常复杂从而限制了其进一步

推广。直到70年代人们才提出了采用离散傅立叶变换来实现多个载波的调制，简化了系统结构，使得OFDM技术更趋于实用化。1970年1月首次公开发表了有关OFDM的专利。1971年，Weinstein和Ebert把离散傅立叶变换(DFT)应用到并行传输系统中，作为调制和解调过程的一部分。这样在完成FDM的过程中，可以完全依靠执行快速傅立叶变换(FFT)的硬件来实施。直到20世纪80年代中期，该方法才开始受到关注并得到了广泛的应用[6]。

80年代人们研究如何将OFDM技术应用于高速MODEM。进入90年代以来，OFDM技术的研究深入到无线调频信道上的宽带数据传输。在高速宽带无线应用环境下OFDM技术的优势很突出，而且可以利用有效的新技术去修正和弥补OFDM的固有缺点，OFDM技术已经被广泛应用于民用通信系统。自20世纪80年代以来，OFDM已经在数字音频广播(DAB)[7]、数字视频广播(DVB)[8]、基于IEEE802.11标准的无线本地局域网(WLAN)以及有线电话网上基于现有铜双绞线的非对称高比特率数字用户线技术(例如ASDL[9])中得到了应用。欧洲的DAB系统使用的就是OFDM调制技术，试验系统已在运行并很快吸引了大量听众，它明显地改善了移动中接收无线广播的效果。用于DAB的成套芯片的开发正在一项欧洲发展项目中进行，它将使OFDM接收机的价格大大降低，市场前景非常看好。其中大都利用了OFDM可以有效地消除信号多径传播所造成的符号间干扰(ISI)这一特征。

20世纪90年代以来，世界各国电信市场相继开放，电信运营主体向多元化方向发展，电信竞争的焦点也由长途骨干网转为本地接入。宽带无线接入凭借其组网快速灵活、运营维护方便以及成本较低等竞争优势成为市场热点。而WLAN作为一种宽带无线数据接入技术，是计算机网络与无线通信技术相结合的产物。它不受电缆束缚，可移动，能解决因有线网布线困难等带来的问题，并且具有组网灵活，扩容方便，与多种网络标准兼容，应用广泛等优点，愈来愈受到重视。OFDM技术作为宽带无线接入系统的基本实现技术之一，必将成为WLAN的核心技术[10]。

OFDM系统存在如下的主要优点：

1. OFDM系统可以有效地减小无线信道的时间弥散所带来的ISI，这样就减小

了接收机内均衡的复杂度，有时甚至不采用均衡器，仅通过采用插入循环前缀的方法消除 ISI 的不利影响。

2. OFDM 系统由于各个子载波之间存在正交性，允许子信道的频谱相互重叠，因此与常规的频分复用系统相比，OFDM 系统可以最大限度地利用频谱资源。

3. 各个子信道中的这种正交调制和解调可以采用 IFFT 和 FFT 方法实现。采用数字信号处理(DSP)技术和 FFT 快速算法，简化电路设计。

4. 无线数据业务一般都存在非对称性，而 OFDM 系统可以很容易地通过使用不同数量的子载波来实现上行和下行链路中不同的传输速率。

5. OFDM 系统可以在某种程度上抵抗窄带干扰，同时可以通过动态比特分配以及动态子信道分配的方法，充分利用信噪比较高的子信道，从而提高系统的性能[11]。

1.4 本课题研究的意义

本论文对无线局域网的 OFDM 系统作了一些探讨和研究。利用 Systemview 和 MATLAB 软件对 IEEE802.11a/g 物理层算法进行了分析和仿真，比较了两种信道编码的性能，并做了相应的误码率分析，得到了有用的结论，建议采用卷积编码，还研究了不同调制映射，保护间隔，载波数和信道对 OFDM 系统的影响，并得出一些结论。通过对 OFDM 的核心部分 FFT 的设计与仿真为软 IP 的开发提供了参考。

1.5 本论文的主要工作

由于无线局域网 IEEE802.11a/g 的协议标准中，物理层的设计主要采用了 OFDM 技术。所以在本论文中，先介绍了 OFDM 技术的基本原理和 IEEE802.11 系列标准的物理层协议。然后根据此标准，用仿真工具软件 Systemview 和 MATLAB 对无线局域网的物理层 OFDM 系统进行了仿真和分析，为得出的相关结论创造了条件。本文对 OFDM 系统的核心 FFT 处理器进行了 FPGA 的硬件设计，并对本文

所设计的 FFT 处理器进行仿真的结果与理论计算的结果相符，证实了该设计的正确性和可行性，从而为软 IP 的开发提供了参考。

本论文的具体结构如下：

第一章介绍了本论文的研究背景和无线局域网技术的发展情况。

第二章介绍了正交频分复用(OFDM)技术的基本原理。

第三章介绍了 WLAN 的 IEEE802.11 系列协议标准及比较。

第四章对 OFDM 系统进行了计算机仿真，为寻找它的最佳工作状态提供了基础。

第五章对 FFT 做了详细的分析并对 FFT 处理器的 FPGA 硬件实现提出了设计方法。由于受设备和实验条件的限制，本论文仅对该设计的正确性和可行性进行了仿真，未能做出成品，但可以确信该设计是有一定的实用价值和理论意义的。

第六章对全文进行了总结并对未来的研究进行展望。

第二章 OFDM 原理

2.1 引言

在传统的并行传输系统中，整个带宽经分割后被送到子信道中，并且频带没有重叠，但是其最大的缺点是频谱利用率很低，造成频谱浪费。所以，人们提出了频谱可以重叠的多载波系统。在 OFDM 系统中各个子信道的载波相互正交，于是它们的频谱可以相互重叠，这样不但减小了子载波间的相互干扰，同时又提高了频谱利用率。

在分析 OFDM 的性能之前，我们有必要从多载波调制（Multi-Carrier Modulation，MCM）的角度去分析这种调制的基本性能。其基本概念是将高速率的信息数据流经串/并变换，分割为若干路低速数据流，每路低速数据采用一个独立的载波调制并迭加在一起构成发送信号。由于速率降低，信息码元周期增大。如果码元周期大于多径时延，那么多径的影响将较少的带到下一个码元，这样就减少了多径干扰对传输系统性能的影响[12]。

2.2 多载波调制

无线信道（特别是陆地移动信道），由于地面情况的复杂性，发射的信号往往是经过多条路径到达接收端，即存在多径传播效应。从而造成接收信号相互重叠，产生信号波形间的相互干扰，造成接收端判断错误，严重影响信号传输质量。这种特性称为信道的时间弥散性。在这样的信道特性下，数字信号在接收端前后码元交迭，产生所谓的码间串扰(ISI)，造成判决错误，严重影响传输质量。特别是在码元速率较高的情况下，更是如此。这是由于此时信号波形的周期很短，在接收端信号波形重叠的程度将进一步加深，信号间的干扰将更加严重，时延扩展将跨越更多的码元，造成严重码间串扰。从另一个角度看，当信号波形的传输速率较

高时，信号带宽较宽。当信号带宽接近和超过信道相干带宽时，信道的时间弥散特性将对接收信号造成频率选择性衰落（信号的衰落与频率有关）。为了保证正确的数据传输，必须对信号的传输速率加以限制。因此，可以说时间弥散是使无线信道传输速率受限的主要原因之一[13]。

对于一个特定的信道特性，设计通信系统时必须考虑如何有效的利用有限的信道带宽，并在收发设备的复杂度折衷条件下可靠的传输信息。对于这种频率响应非理想的滤波信道，一种办法是在单载波调制时，以特定的码元速率进行传输，在不改变码元速率并承认有了较严重的多径扩散的条件下，利用 RAKE 接收机进行均衡和补偿，采用扩频码将传播的多径信号能量分离、校正，并加以收集利用，化害为利，从而设法消除多径干扰的影响[14]。

另外一种方法则是采用多载波调制（MCM），将特定带宽的非理想线性信道划分为 N 个近似线性的子信道，在每个子信道中以 1/N 码元速率的低速码流进行传输。数据传输速率低后，码元周期长，只要时延扩展与码元周期之比小于一定的值，就不会造成码间串扰。因而，从本质上说，多载波调制对信道的时延弥散不敏感，或者说具有抗时延弥散的特性。使用单载波调制进行高速率传输时必须加均衡器，而用多载波调制不加均衡器也能获得较好的性能 。

为了详细说明这一点，我们假设 $C(f)$ 为带宽 W 的带限非理想信道的频率响应，$\Phi_{nn}(f)$ 为加性高斯白噪声的功率谱密度。将 W 划分成 $N=W/\Delta f$ 个带宽为 Δf 的子信道。假设 Δf 足够小，以使 $|C(f)|^2/\Phi(f)$ 近似为常数。同时，还需考虑到，发射机的平均发射功率 P_{av} 一定，假设功率谱密度为 $P(f)$，由此得到以下的约束条件：

$$\int_W P(f)df \le P_{av} \tag{2.1}$$

根据香农公式[15]，理想带限白高斯信道的信道容量为：

$$C = W\log_2\left(1+\frac{P_{av}}{WN_0}\right) \tag{2.2}$$

这里 C 为信道容量，单位为 bit/s，W 为信道的带宽，而 P_{av} 为发射机的平均发射功率。在多载波系统中，若 Δf 足够小，则子信道的容量为：

$$C_i = \Delta f \log_2\left[1+\frac{\Delta f P(f_i)|C(f)|^2}{\Delta f \Phi_{nn}(f_i)}\right] \tag{2.3}$$

因此，信道的总容量为：

$$C = \sum_{i=1}^{N} C_i = \Delta f \sum_{i=1}^{N} \log_2\left[1+\frac{P(f_i)|C(f)|^2}{\Phi_{nn}(f)}\right] \tag{2.4}$$

当$\Delta f \to 0$取极限时，确定整个信道的容量（单位为bit/s）为：

$$C = \int_W \log_2\left[1+\frac{P(f_i)|C(f)|^2}{\Phi_{nn}(f)}\right] df \tag{2.5}$$

代入(2.1)式确定的$P(f)$约束条件，使C最大化的$P(f)$可以通过下列积分式的最大化来确定：

$$\int_W \{\log_2[1+\frac{P(f)|C(f)|^2}{\Phi_{nn}(f)}]+\lambda P(f)\} df \tag{2.6}$$

式中，λ为拉格朗日（Lagrange）乘法因子，可以选择它以满足约束条件。通过变量的微积分进行最大化运算，则发送信号功率的最佳分布是下列方程的解：

$$\frac{|C(f)|^2}{|C(f)|^2 P(f)+\Phi_{nn}(f)}+\lambda = 0 \tag{2.7}$$

因此$P(f)+\Phi_{nn}(f)/|C(f)|^2$必须为常数，且需要满足(2.1)的最大平均功率约束。最后我们得到：

$$P(f) = \begin{cases} K-\Phi_{nn}(f)/|C(f)|^2 & (f \in W) \\ 0 & (f \notin W) \end{cases} \tag{2.8}$$

由上式可以看到，信号最佳功率谱$P(f)$是由噪声功率谱$\Phi_{nn}(f)$和信道传输特性$C(f)$唯一确定的。这样就可以得到非线性信道的最优发射功率分配图，如图2-1所示。$\Phi_{nn}(f)/|C(f)|^2$曲线可以看成单位深度碗的底部，发射机的平均功率P_{av}可看成倒入碗中的水，为了达到最优化的信道容量，倒入碗中的水将自行分配以满足(2.8)式。当最优化达到时，水的上表面为水平且等于常数K，这就是信息论中著名的“注水定理”。

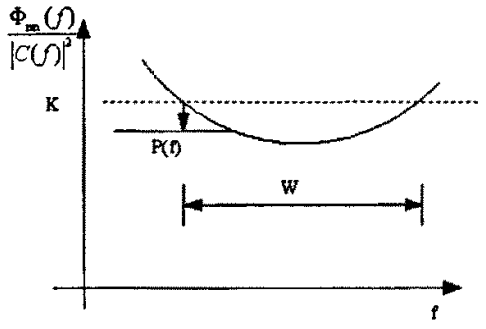


图 2-1 注水定理功率分配图

Fig. 2-1 Power allocated of note the water theorem

我们注意到，当$C(f)$=1 即信道的频率响应为常数时，信道最优化的发射功率谱密度由高斯白噪声的功率谱密度决定。因此，多载波调制将可用的信道带宽划分为多个窄带宽信道。由香农公式的推导可知，当子信道的频率响应为理想线性信道时，可以达到最大的信道容量。在每个子信道中，可以根据信道特性决定的发射功率谱密度。每个信道在独立的进行编码和选用适合在该子信道中传输的映射样式进行传输。在信噪比较好的情况下可以选用 MQAM 映射样式，在信噪比较差时可以选用 BPSK 或 QPSK 的映射样式，甚至在不适合传输的子信道可以关断，以克服单频噪声的影响。另外，我们也可以看到，当Δf足够小的时候，$C(f)$近似为常数，在接收端也就不需要均衡算法进行补偿，因为符号间串扰可以忽略不计了。

但是，多载波调制也存在着一些明显的缺点。例如，当信道随时间快速变化时，会引起频率弥散，造成接收信号的频率偏移和相位跳变。一些多载波调制可用于快速时变信道即频率弥散信道中，并获得好处。这是由于码元周期相对较长，以至于在一次信号衰落期间内码元能量不大可能完全消失。然而，在另一些对于信道载波相互间关系有严格要求的多载波调制中，频率弥散会造成信道间干扰(Inter Channel Interference)。对频率偏移的敏感性常常被认为是多载波调制的主要缺点之一[16]。

多载波调制(MCM)可通过多种技术途径来实现[17]。

多音实现(Multitone Realization)，它使用通常的频分复用技术和带限信道，将整个射频带宽分割成若干个互不交叠的子载波信道来并行传输各个子数据流，在

接收端用一组滤波器来分离各个子信道。这种方法直接、简单。缺点是频谱效率较低，且多个滤波器实现困难。

正交频分复用(OFDM)技术属于多载波调制。MCM 与 OFDM 常用于无线信道中，它们的区别在于：OFDM 技术特指将信道划分成正交的子信道，频谱利用率高；而 MCM 可以是更多种信道划分方法。在有线环境中，该技术通常称为离散多音(DMT)。OFDM 技术的主要思想是：将指配的信道分成许多正交子信道，在每个子信道上进行窄带调制和传输，信号带宽小于信道的相关带宽。它使用相互正交的一组子载波构成子信道来传输各个子数据流。子信道的频谱是可以相互交叠的，这样就提高了频谱效率。通过各个子载波的联合编码，可具有很强的抗衰落能力。另外，采用时间受限的脉冲来进一步降低对时延扩展的敏感性。另一个更重要的优点是 OFDM 能够用 FFT 算法实现，并可以采用非常有效的数字信号处理 (DSP)技术。

MC-CDMA，多载波码分多址或码分复用(Multi-Carrier Code Division Access or Multiplexing)是另外一种将信号扩展到不同子载波上的重要方式。它将直接扩频序列码分多址(DS-CDMA)用于复用，而采用 OFDM 的原理来选择波形，将不同用户的信号线性迭加到一个复用的多载波信号上。研究表明，MC-CDMA 信号能够用结构相当简单的接收机来检测。这种接收机采用 FFT 技术和可变增益分集合并，其每一支路的增益仅由该子载波的信道衰落所控制。MC-CDMA 系统可以在高时间弥散信道中工作并达到令人满意的误比特率。MC-CDMA 的良好特性越来越引起研究者的兴趣。

由上面的分析我们可以看到，OFDM 调制技术与其他的多载波调制技术相比，具有频带利用率高，抗多径效应带来的码间串扰，可利用数字信号处理技术实现的优点；并且可以同当前的许多其他先进技术联合使用，大大提高传输数据的速度。因此，OFDM 技术成了当前多载波调制的研究热点。

2.3 OFDM 基本原理

2.3.1 频分复用（FDM）原理

频分复用(FDM)[18][19]是指将信道划分成 N 个子信道，利用 N 个不同频率的子载波并行的在子信道上传输 N 路数据。频分复用的传输系统发送端的组成框图如图 2-2 所示。接收端则是一个相反的过程。

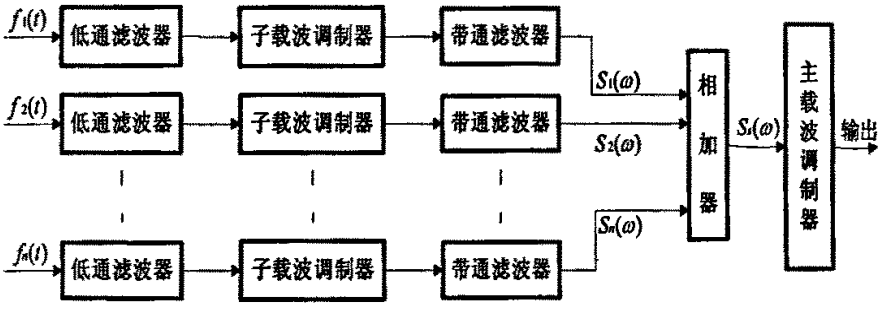


图 2-2 频分多路传输系统组成框图

Fig.2-2 FDM Transmission system

假设待传的 N 个具有相同带宽 ω_f 的信号为 $f_1(t)$，$f_2(t)$，…，$f_n(t)$ 分别通过一个低通滤波器，以保证其带宽不超过 ω_f；因为这些信号占有同一频带，如果直接加于同一信道上，接收端将无法进行区分。所以要对它们的频谱进行搬移，使其在频率轴上互不重叠。因此，各路信号先要用子载波进行调制从而实现频谱搬移。用一组有相同频率间隔的正弦波作为子载波，相应的频率称为子载频。为了限制各路子载波所占频带，在相加器前，每一路设一个带通滤波器。多路信号仍属于基带信号，可以直接用导线传输。信号此时在频带上是互不重叠的，因此可以用相加器将 N 路信号和在一起传输。频分多路信号可表示为：

$$S_s(\omega)=\sum_{n=1}^{N} f_n(t)\cos\omega_n t \tag{2.9}$$

为了实现无线传输，还需将合成的信号对射频载波进行一次调制，称为主载波调制或二次调制。在接受端，解调过程是一个相反的变换。首先，对射频信号

进行主载波解调，恢复出的多路信号加到各个分路带通滤波器上，各个带通滤波器的中心频率分别对应该路带宽和子载波频率，只允许本路信号通过，从而实现了频域的分割。分离后的信号进行子载波解调，就可得到各路信息。FDM 的频谱分析如下图，ω_g是保护间隔。

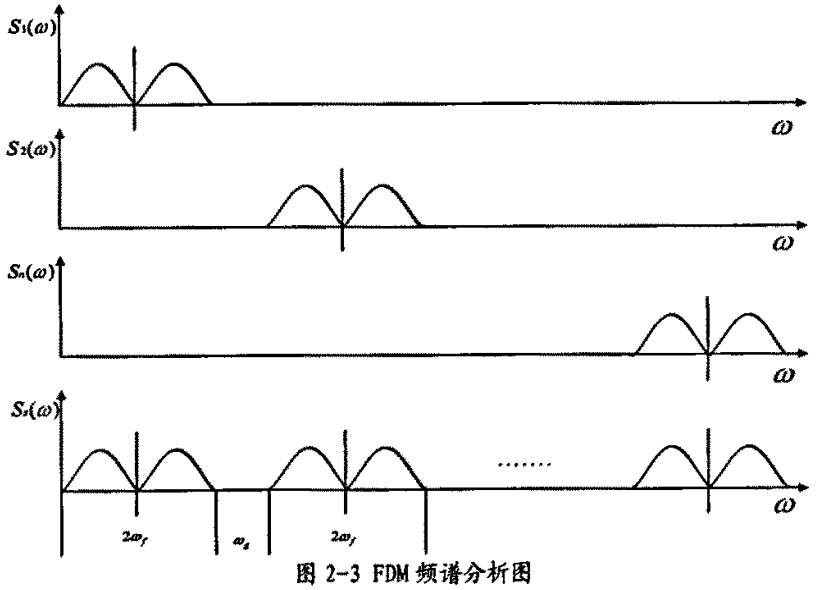


图 2-3 FDM 频谱分析图

Fig.2-3 Analysis figure of FDM frequency spectrum

2.3.2 正交频分复用（OFDM）原理

正交频分复用(Orthogonal Frequency Division Multiplexing，OFDM) [20]是在FDM 的原理的基础上，子载波集采用两两正交的正弦或余弦函数集。OFDM 是一种令人感兴趣的多载波调制方式，单个用户的信息流被串/并转换为多个低速率码流，每个码流都用一个载波发送。

在传统的并行传输系统中，整个系统频带被划分为 N 个互不混叠的子信道，每个子信道被一个独立的信源符号调制，即 N 个子信道被频分复用。这种做法，虽然可以避免不同信道互相干扰但却以牺牲频带利用率为代价，这在频带资源如此紧张的今天尤其不能忍受。所以，人们提出了频谱可以重叠的多载波系统。在 OFDM 系统中各个子信道的载波相互正交，于是它们的频谱是相互重叠的，从而提高了频谱利用率，如图 2-4 所示。

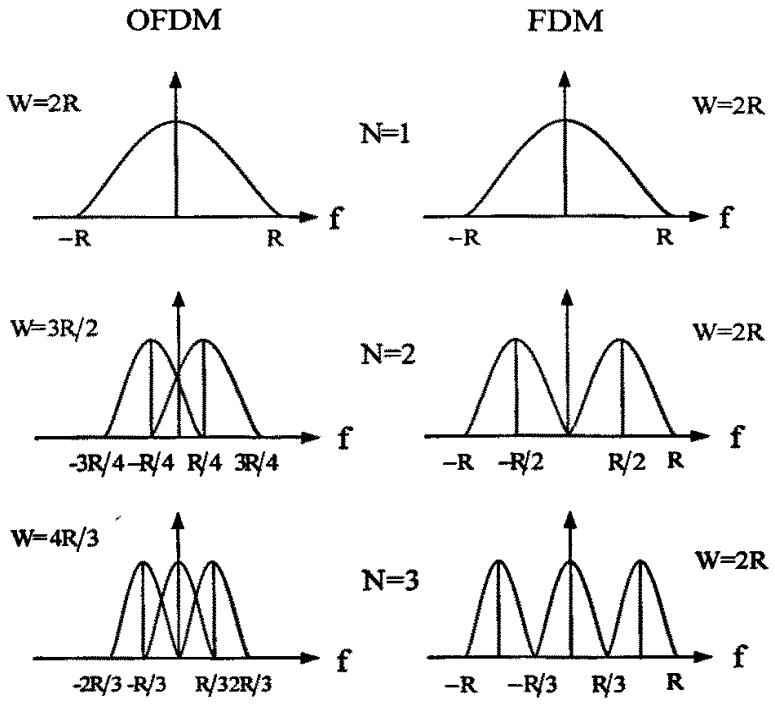


图 2-4 OFDM 和 FDM 频谱利用率比较

Fig.2-4 Frequency spectrum utilization comparing between OFDM&FDM

OFDM 采用了基于载波频率正交的 FFT 调制，由于各个载波的中心频点处没有其他载波的频谱分量，所以能够实现各个载波的正交。尽管还是频分复用，但已与过去的 FDM 有了很大的不同，不再是通过很多带通滤波器来实现，而是直接进行基带处理。这也是 OFDM 有别于其他系统的优点之一。OFDM 的接收机实际上是一组解调器，它将不同载波搬移至零频，然后在一个码元周期内积分，其他载波由于与所积分的信号正交，因此不会对这个积分结果产生影响。OFDM 的数据速率与子载波的数量有关，增加子载波数目就能提高数据的传送速率。

在正交频分复用中由于符号时间大大延长，相对时延不超过符号周期，符号间串扰就会避免。为了更有效的抗符号间串扰，在每一帧信号中插入保护间隔时间，使整个符号持续时间分成有用符号持续时间和保护间隔两部分。保护间隔内不传新的信息，只是加入循环前缀对传输的一小部分信号进行重复。如图 2-5 所示，要传输的符号是长度为 N 的 OFDM 时域符号，将最后的 N_g 个符号添加到待传输

符号的最前面组成长度为 $N+N_g$ 的传输符号，N_g 个采样符号的持续时间即为保护间隔。这样，将每个符号的宽度延长到远大于最大多径扩散的程度，就可以消除信道中绝大部分的符号间串扰，因此也就可以省去了复杂的均衡技术。在接收端，我们将丢弃掉保护间隔内所接收到的采样信号，并且认为在保护间隔后所接收到的采样信号就是 N 个 OFDM 时域采样符号。

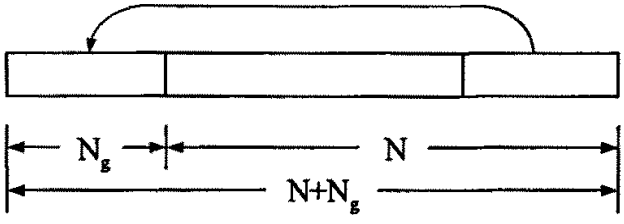


图 2-5 伴有 Ng 个循环扩展的 OFDM 时域信号

Fig.2-5 OFDM time domain signal of N_g extension circulating

同时，从频域上看，多径造成符号间串扰的原因在于多径信道传输函数的频率特性不满足奈奎斯特定理，折叠后的频谱为常数。于是，把信号频谱划分成非常多的子带，就可能在每个子带上分别满足奈奎斯特定理。子载波在信道中的特性稳定，对于频率选择性衰落不敏感，可以通过采用灵活的调制方式(如BPSK,QPSK,16-QAM,64-QAM)来最大限度的提高频谱利用率。

设基带调制信号的带宽为 W，码元调制速率为 R，码元周期为 t_S，且信道的最大迟延扩展 $\Delta_m > t_s$。OFDM 是将原信号分割为 N 个子信号，分割后码元速率为 R/N、周期为 $T_S = Nt_s$，然后用 N 个子信号去分别调制 N 个相互正交的子载波。设第 k 个符号 $d(k)$调制第 k 个载波 $\exp(j2\pi f_k t)$，则 OFDM 调制信号可以表示为：

$$D(t) = \sum_{k=0}^{N-1} d(k)\exp(j2\pi f_k t) \qquad t \in [0, T_S] \tag{2.10}$$

这里 $d(k)$为第 k 个调制码元，T_S为码元周期，各子载波的频率满足下列关系：

$$f_k = f_0 + k/T_S, \qquad k = 0,1,...,N-1 \tag{2.11}$$

载波的基本单元信号为：

$$\begin{cases} f(t,k)=e^{j2\pi f_k t}, 0\le t<T_S, \\ f(t,k)=0, \quad \text{其他}, \end{cases} \tag{2.12}$$

故这些基本的单元信号满足正交性：

$$\begin{cases} \int_0^{T_S} f(t,k)\cdot f^*(t,k')dt=0, & K\neq P \\ \int_0^{T_S} f(t,k)\cdot h^*(t,k')dt=\int_0^{T_S}\left|f(t,k)\right|^2 dt=T_S, & K=P \end{cases} \tag{2.13}$$

在接收端，输入信号分成N个支路，分别用各子载波混频和积分，恢复出子载波上调制的信号，再经过并串变换和常规QAM解调就可以恢复出数据。由于子载波的正交性，混频和积分电路可以有效地分离各个子信道，如下式所示：

$$\begin{aligned} \hat{d}(m)&=\frac{1}{T_S}\int_0^{T_S}\sum_{n=0}^{N-1}d(n)\exp(j\omega_n t)\exp(-j\omega_m t)dt \\ &=\frac{1}{T_S}\sum_{n=0}^{N-1}d(n)\int_0^{T_S}\exp(j(\omega_n-\omega_m)t)dt \\ &=\frac{1}{T_S}\sum_{n=0}^{N-1}d(n)\int_0^{T_S}\exp(\frac{j2\pi(n-m)}{T_S}t)dt \\ &=d(m) \end{aligned} \tag{2.14}$$

OFDM系统的调制解调原理示意图如图2-6所示。

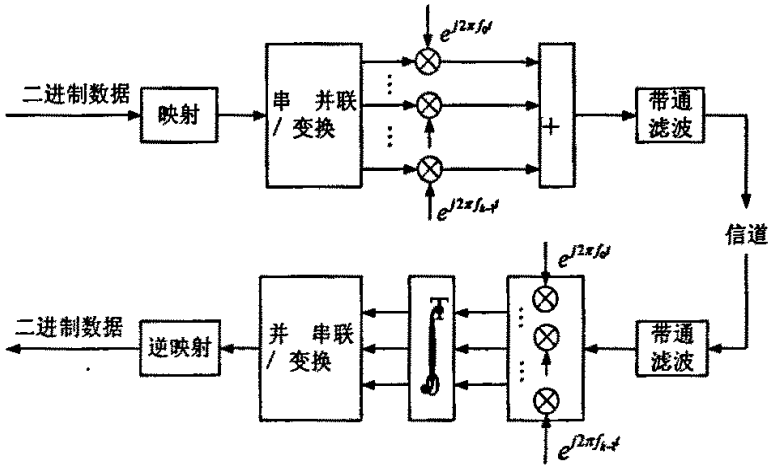


图 2-6 OFDM的调制解调原理图

Fig.2-6 Modulation and demodulation of OFDM

2.4 OFDM 系统性能特点

OFDM 的本质是一种多载波并行传输系统，它将待传输的信息分到多个载波上分别传送。这样一来，每个子带的传输速率就比整个信道带宽小得多。也就是说，经过调制的符号的持续时间变大，可能比反射扩展时间大得多。如果相邻符号之间插入保护间隔，这样多经衰落可能不产生任何符号间干扰[6]。

由于并行系统的特点，OFDM 系统对脉冲干扰的抵抗力比单载波系统大得多，这是因为对 OFDM 信号的解调是在一个很长的符号周期内积分，从而使脉冲噪声的影响得以分散。事实上，对脉冲噪声强大的抑制作用是最初研究 MCM 的动机之一。提交给 CCITT 的测试报告表明，能够引起 MCM 系统发生错误的脉冲噪声的门限电平比单载波系统高 11dB。

由于 OFDM 系统把信息分散到许多个载波上，大大降低了各子载波的信号速率，使符号周期比回波延迟长，从而能够减弱多经传播的影响。

2.4.1 调制方式

OFDM 系统的各个载波可以根据信道的条件来使用不同的调制，比如 BPSK、QPSK、16QAM、64QAM 等等。以频谱利用率和误码率之间的最佳平衡为原则。选择满足一定误码率的最佳调制方式可以获得最大频谱效率。多经信道的频谱选择性衰落会导致接收信号大幅下降，达到 30dB 之多，信噪比也大幅下降。使用与信噪比相匹配的调制方式可以提高频谱利用率。众所周知，可靠性是通信系统运行是否良好的重要考核指标，因此系统通常选择 BPSK 或 QPSK 调制。这样可以确保在信道最坏条件下的信噪比要求，但是这两种调制的频谱效率太低。如果使用自适应调制，那么在信道好的时候终端就可以使用效率较高的调制。同样在终端靠近基站时，调制可以由 BPSK（1bit/s/Hz）转化成 16-64QAM（4-6bit/s/Hz），整个系统的频谱利用率得到大幅度的改善，自适应调制能够使系统容量翻番。但任何事物都有其两面性，自适应调制也不例外。它要求信号包含一定冗余比特，以告知接收端发射信号所采用的调制方式，并且，终端需要定期更新调制信息，

这又势必会增加更多的冗余比特。OFDM 技术使这个矛盾迎刃而解。通过采用功率控制和自适应调制协调工作的技术，信道好的时候，发射功率不变，可以采用高效调制方式（如 64QAM），或者在低效调制（如 QPSK）时降低发射功率。功率控制与自适应调制要取得平衡[21]。

失真与频偏也是在选择调制时必须考虑的因素。信道衰落以及延迟失真的影响使得各个子载波以不同的幅度和相位接收。同时多经传播引起的失真也可能使各子信道把能量扩散到相邻信道，从而产生带间干扰 ICI(Inter-channel Interference)和码间干扰 ISI(Inter-Symbol Interference)，使各载波失去正交性，在接收端不能正确恢复出信号。传输的非线性会造成互调失真（MD），此时信号具有较高的噪声电平，信噪比一般不会太高。失真和多普勒频移造成的频率偏移使信道间失去正交特性。仅仅 1%的频偏就会使成信噪比下降 30dB。自适应调制要求对信道的性能有充分的了解，如果在差的信道上使用效率较高的调制方式，那么就会产生较高的误码率，影响系统的可靠性。OFDM 系统的导频信道或参考码字可以用来测试信道的好坏。发送一个已知数据的码字，在满足通信极限的情况下测量出每条信道的信噪比，根据这个信噪比来确定最合适的调制方式。

2.4.2 信道编码

在有强反射干扰的情况下，不同反射的某种组合可能使得某个载波受到严重衰减，而另一些子带可能被放大。如果信号能量增加是由于那些相互延时超过带宽倒数的不同反射引起的，那么在接收机的输入端的信噪比会提高。由于一部分子带被严重衰减了，为了利用这些能量增加的子带，所以就需要在 OFDM 系统中加入强有力的信道编码措施。

编码将时域交织和频域交织结合在一起，其作用是在那些不同子带上传输的数据之间以及同一子带上前后数据之间建立一种联系。这样，利用接受效果较好的子带上的数据与严重衰落子带上数据之间的联系，可以恢复这种受到破坏的数据。编码和交织应用于 OFDM 可以看作是一个在整个传输带宽和整个交织深度上平均本地衰落的工具。

2.4.3 多天线

OFDM 由于码率低和加入了时间保护间隔而具有极强的抗多径干扰能力。由于多径时延小于保护间隔，所以系统不受码间干扰的困扰。这就允许单频网络(SFN)可以用于宽带 OFDM 系统，依靠多天线来实现，即采用由大量低功率发射机组成的发射机阵列消除阴影效应，来实现完全覆盖[22]。

多天线系统非常适用于无线局域网。一般的局域网由于阴影效应，信号无法完全覆盖，需要使用中继器。对于传统系统来说，中继器可能会带来多径干扰。但 OFDM 不存在这个问题，它的中继器可以加在任何需要的地方，不仅可以完全覆盖网络，并且可以消除多径干扰。

2.5 本章小结

本章在分析多载波调制基础上阐述了OFDM的基本原理，正交频分复用采用并行传送的方式，利用傅立叶变换(FFT)算法实现调制，降低了码速率。数据帧之间插入保护间隔，可以较好克服符号间串扰(ISI)。通过分析可以看到OFDM有以下的特点：

1)、OFDM 技术的最大优点是对抗频率选择性衰落或窄带干扰。在单载波系统中，单个衰落或干扰能够导致整个通信链路失败，但是在多载波系统中，仅仅有很小一部分载波会受到干扰。对这些子信道可以采用纠错码来进行纠错。

2)、有效地对抗信号波形间的干扰，适用于多径环境和衰落信道中的高速数据传输。当信道中因为多径传输而出现频率选择性衰落时，只有落在频带凹陷处的子载波及其携带的信息受到影响，其他的子载波未受损害，因此系统总的误码率性能要好得多。

3)、通过各个子载波的联合编码，具有很强的抗衰落能力。OFDM 技术本身已经利用了信道的频率分集，如果衰落不是特别严重，就没有必要再加时域均衡器。通过将各个信道联合编码，则可以使系统性能得到提高。

4)、可以选用基于 IFFT/FFT 的 OFDM 实现方法，即采用快速傅立叶变换实现 OFDM。

5)、信道利用率很高，这一点在频谱资源有限的无线环境中尤为重要。

当然 OFDM 也有它自身的不足之处，比如峰均值比较大、对频偏和相位噪声敏感等等，随着人们对 OFDM 的进一步研究，相信它的缺点会被克服的。

随着 IEEE802.11a、IEEE802.11g 协议、ETSI BRAN(Broadband Radio Access Network)和多媒体应用的引入，无线通信领域已经为 OFDM 技术的应用做好了准备。

第三章 OFDM 技术在 IEEE802.11 系列标准下的分析

无线局域网（WLAN, Wireless Local Area Network）可定义为，使用射频（RF, Radio Frequency），微波(Microwave)或红外线(Infrared)，在一个有限地域范围内实现互联设备的通信系统。它可以作为有线局域网的扩展来使用，也可以独立作为有线局域网的替代设施，提供了很强的灵活组网性[23]。

1997 年 IEEE802.11 标准的制定是无线局域网发展的里程碑，它是由大量的局域网及计算机专家审定通过的标准。IEEE802.11 标准定义了单一的 MAC 层和多样的物理层，其物理层标准主要有 IEEE802.11b，802.11a，802.11g[24]。

3.1 IEEE802.11b 标准的逻辑结构和调制方式

IEEE802.11b[3]标准是 IEEE802.11 标准的一个扩充，它规定采用 2.4GHz ISM 频带，并在此频段上分配了 14 个工作信道，用来提供一个支持更高传输速率的物理层，调制方法主要采用补偿编码键控（CCK）。IEEE802.11 b 的 MAC 层基本结构、特性和服务与最初的 802.11 标准定义相比，只增加了一项动态速率转换的功能，能保证在 4 种接入速率中实时的切换。同时 IEEE802.11b 标准对 IEEE802.11 标准的物理层进行了完全的扩充，规定了更适合高速数据传输的 PLCP 层帧结构及全新的高速调制方式，能提供更高的数据传输速率和更健全的连接性。

3.1.1 IEEE802.11b 的物理层构成

IEEE802.11b 物理层由三部分组成，分别为物理层汇聚子层（PLCP）、物理介质依赖子层（PMD）和物理层管理实体（PLME）。PLCP 子层的主要作用是为了保证 IEEE 802.11 MAC 可以最低限度地依赖于 PMD 子层，即是简化 PHY 与 802.11 MAC 之间接口。PMD 子层用以提供在两个或更多的基站之间发送数据的一种方法，在 IEEE802.11b 中特指在 2.4G Hz 频段中利用 DSSS 和 CCK 或其它备用调制

方式进行数据调制发射的过程。IEEE802.11b PMD 支持 1，2，5.5，11 Mbps 四种传输速率。PLME 指的是用于对物理层的各种特性进行管理的一些变量、函数及在 MAC 层与 PHY 层之间交换的一些状态信号的集合。可以看出，PLCP 子层在整个物理层中起到一个承上启下的作用，因此它的结构特性为物理层的核心。

3.1.2 PLCP 子层

PSDU（物理层服务数据单元）在 PLCP 子层中转换位 PPDU（PLCP 协议数据单元），在传输过程中，PSDU 将被附加一个 PLCP 前导序列和适配头来创建 PPDU。在 IEEE802.11b 中定义了两种不同的前导序列和适配头：强制性长前导序列和适配头，与 IEEE802.11 标准中 1Mbps 和 2Mbps 速率具有相互操作性；可选的短前导序列和适配头。在接收时，PLCP 前导序列和适配头帮助处理解调和传送 PSDU。

在 IEEE802.11b 中定义短前导序列和适配头的目的是为了获得最大的吞吐量，并且与不具备短前导序列的设备具有可相互操作性，也就是说，它可以与其他设备一样在网络中使用，由用户选择工作模式。下面来介绍这两种 PLCP 帧结构：

1. 长前导序列 PPDU 帧格式

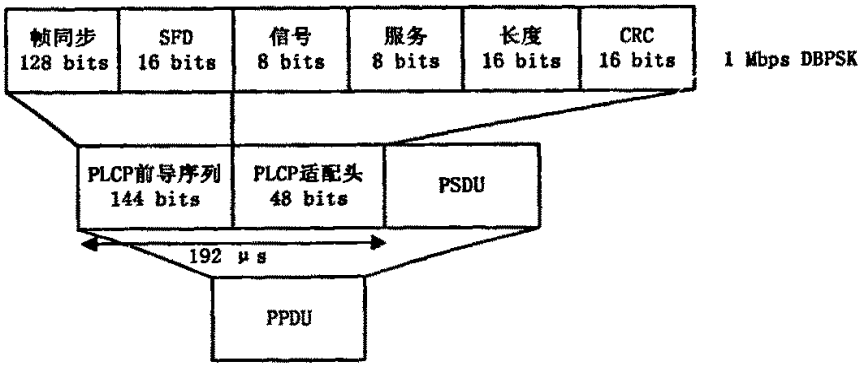


图 3-1 长前导序列 PPDU 帧格式

Fig.3-1 Frame structure of long PLCP preamble PPDU

长前导序列 PPDU 帧格式如图 3-1 所示。其中前导序列由 128 位同步码(SYNC)和 16 位开始帧界定符(SFD)构成。同步码(SYNC)是 128 位经过扰码后的“1”（扰码器的种子码为“1101100”），它被用于唤醒接收设备，使其与接受信号同步。开

始帧界定符(SFD)用于通知接收机，在 SFD 结束后紧接着就开始传送与物理介质相关的一些参数。

前导序列结束后，就是 PLCP 适配头信息(PLCP header)，这些信息中包含了与数据传输相关的物理参数。这些参数包括：信号(SIGNAL)、服务(SERVICE)、将要传输的数据的长度(LENGTH)和 16 位的 CRC 校验码。接收机将按照这些参数调整接收速率，它有四个值：0Ah、14 h、37 h 和 6Eh，分别指定传输速率为 1Mbps，2 Mbps，5.5 Mbps 和 11 Mbps，接收机将按此调整自己的接受速率。SERVICE 字段长度也是 8 位，它指定使用何种调制编码(CCK 还是 PBCC)。LENGTH 字段长 16 位，用于指示发送后面的 PSDU 需用多长时间(单位为微秒)。16 位 CRC 校验码用于校验收到的信令、业务和长度字段是否正确。

前导序列和 PLCP 适配头信息以固定的 1Mbps 速率发送，而 PSDU 数据部分则可以以 1 Mbps(DBPSK 调制)、2 Mbps(DQPSK 调制)、5.5Mbps(CCK 或 PBCC 调制)和 11 Mbps(CCK 或 PBCC 调制)速率进行传送。

2. 短前导序列 PPDU 帧格式

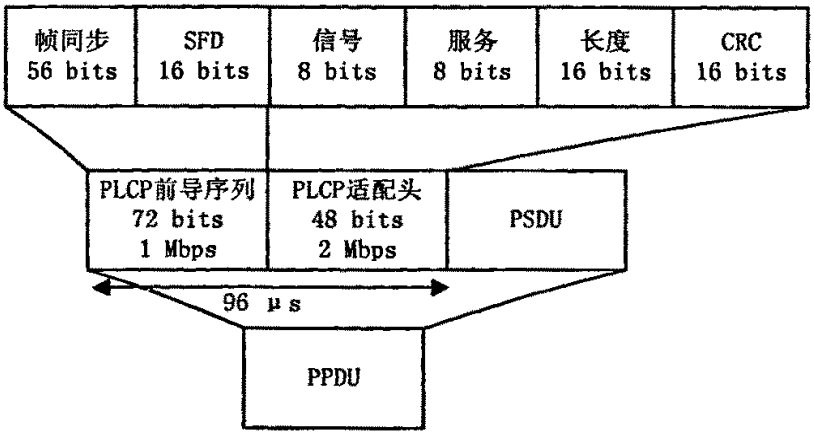


图 3-2 短前导序列 PPDU 帧格式

Fig.3-2 Frame structure of short PLCP preamble PPDU

短前导序列 PPDU 帧格式如图 3-2 所示。其前导序列长度位 72 位，同步码(SYNC)为 56 位经过扰码的“0”(扰码种子码为“0011011”)，开始帧界定符(SFD)长为 16 位，其码值是长 PPDU 格式 SFD 的时间反转码。

SIGNAL 字段长 8 位，只有三个值：14h、37h、6Eh，分别指定传输速率位 2 Mbps、

5.5 Mbps、11 Mbps。SERVICE 字段、LENGTH 字段和 CRC 校验字段与长 PPDU 格式定义相同。

短 PPDU 帧结构的前导序列传输速率为 1 Mbps(DBPSK 调制)，整个 PLCP 适配头信息的传输速率为 2 Mbps，PSDU 数据传输速率为 2 Mbps，5.5 Mbps，11 Mbps。

3.1.3 IEEE802.11b 标准中的 CCK 调制方式

为了提高频率利用率，在 IEEE802.11b 标准中，采用了一种先进的编码技术，这就是 CCK(complementary code keying，补码键控)[25][26]技术。

在 CCK 调制中采用了正交编码技术，在这种技术中，每一个用户有一套正交序列来表述传输的符号集。正交序列集又叫正交编码，一般采用 Hadamard 码或 Walsh 码。信息比特流被划分成很多组，每一组代表一个非二进制信息的符号，与某一特定的传输码序列相关。如果每一组有 N 个比特，则有 2^N 个对应序列，每个符号间隔发送这 2^N 个序列中的一个。接受信号再和 2^N 个匹配代码作相关运算，每一个匹配代码序列与一个符号相关。通过比较相关器的输出，与最大输出相关联的那个符号就被判决为传输的符号。

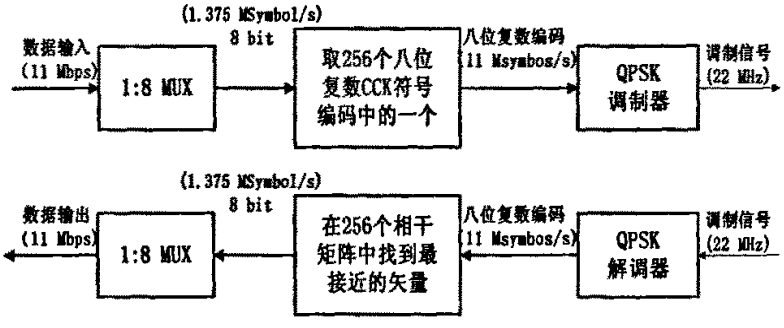


图 3-3 IEEE802.11b 标准中 CCK 调制解调基本原理

Fig.3-3 Modulation and demodulation of CCK in standard IEEE802.11b

IEEE802.11b 标准中规定 5.5 Mbps 和 11 Mbps 速率采用 CCK 调制。CCK 的基本原理方框图如图 3-3。输入的数据被分成 8 个比特一组，共有 256 种符号，速率为 11 Mbps/8=1.375MSymbol/s(符号/秒)。

CCK 码是一种多相位互补的 8 为序列，该序列有明显的自相关响应。设一个

码字集合中第 k 个码字为 $c^k=\{c_1^k,c_2^k,...,c_n^k\}$，其中 n 是码字的长度，这些码字的自相关定义为

$$R_{kk}[j]=\sum_{i=1}^{n-j}c_i^k c_{i+j}^k \tag{3.1}$$

如果是互补的一对码字应该具有如下特性

$$\sum_{k=1}^{2}R_{kk}[j]=\begin{cases}2n, & j=0,\\ 0, & j\neq 0.\end{cases} \tag{3.2}$$

在实际应用中可以将这样的互补码字对扩展到包含 K 个码序列的互补码字集，它们的自相关应该具有如下特性

$$\sum_{k=1}^{K}R_{kk}[j]=\begin{cases}Kn, & j=0,\\ 0, & j\neq 0.\end{cases} \tag{3.3}$$

补码的一个特性是其自相关向量之和处处均为零，只有偏移指数为零指出除外，这个特性提高了在存在多径干扰情况下的信道辨别能力。以上我们仅仅考虑了二进制互补码字，然而对于多相码我们一样可以引入互补的概念。这里的多相码可以用等幅的复数来描述，即如果 c_i^k 是一个多相码元，可以描述为 $c_i^k=e^{j\varphi}$，互补的多相码字的自相关同样满足上式，该式也是复数补码集合所具有的特征之一。

CCK 调制中采用的码字就是一些互补的用复数表示的多相码字，这些互补的复数码字是 Walsh/Hadamard 函数中的一个复数集合，它们与 Walsh 函数有相似之处，那就是这些补码之间也都基本上是正交的，但并不是所有的都完全正交；显然它们与 Walsh 函数也有不同之处，即它们是复数，有两个以上的相位。

CCK 调制的序列都是由复数平面上的 8 个数值组成，共有 4^8=65536 种可能。从这 65536 种可能的组合中选择出 256 种来映射 8 bit 一组的传输数据，选出的 256 种符号是彼此正交的。8 位四相编码块中的每一符号采用 QPSK 四相调制器串行发送。在接收端，每 8 个从 QPSK 解调器解调接收的复数波形再被编组成块送到解码器，根据解调后的 8 位四相位组合信号来找出最接近的 8 位符号。这种 CCK 与 QPSK 调制方式的结合，可对每个符号的 8 位进行编码，使得 IEEE802.11b 可按 11 Mbps 的速率传送信息。编码时采用的映射规则由下面的方程给出

$$c=\left\{e^{j(\varphi_1+\varphi_2+\varphi_3+\varphi_4)},e^{j(\varphi_1+\varphi_3+\varphi_4)},e^{j(\varphi_1+\varphi_2+\varphi_4)},-e^{j(\varphi_1+\varphi_4)},e^{j(\varphi_1+\varphi_2+\varphi_3)},e^{j(\varphi_1+\varphi_3)},-e^{j(\varphi_1+\varphi_2)},e^{j\varphi_1}\right\} \quad (3.4)$$

8 个输入的比特进一步分组成 4 个双比特复数四相符号，它们的对应关系如表 3-1。这样产生的 4 个相位φ_1，φ_2，φ_3，φ_4代入上面的公式就得到 8 个复数编码的波形，再串行送到 QPSK 调制器进行调制。

表 3-1 输入数据与相位的对应关系

Tab.3-1 Correspondence relationship of input data and phase

数据	相位参数
(d_1,d_0)	φ_1
(d_3,d_2)	φ_2
(d_5,d_4)	φ_3
(d_7,d_6)	φ_4

3.2 IEEE802.11a 标准的分层结构分析

3.2.1 IEEE802.11a 标准描述及其物理层

IEEE802.11a 标准[4][27]是作为 IEEE 802.11 的增补协议出现的，主要涉及物理层协议的实现。该标准将 OFDM 作为无线局域网调制解调的核心技术。工作频率定在 5.15~5.25，5.25~5.35 以及 5.725~5.825GHz。通信数据载荷定为 6，9，12，18，24，36，48 和 54Mbit/s。目前的产品已经支持数据载荷为 6，12，24Mbit/s。

该系统采用 52 个子载波，每个子载波上可以映射 BPSK/QPSK、16QAM 或 64QAM 等多种调制样式。主要参数如表 3-2。

表 3-2 IEEE802.11a 标准的主要参数

Tab.3-2 Main parameters of standard IEEE802.11a

数据速率	6，9，12，18，24，36，48，54 Mbps
调制	BPSK，QPSK，16QAM，64QAM
编码速率	1/2，2/3，3/4
子载波数	52
导频数	4
OFDM 符号间隔	4 μs
IFFT/FFT 间隔	3.2 μs
保护间隔	0.8 μs
子载波间隔	312.5 kHz
信号带宽	16.66 MHz
信道间隔	20 MHz

IEEE802.11a 标准的物理层与 3.1.1 节中所描述的 IEEE802.11b 标准的物理层相同，如图 3-4 所示

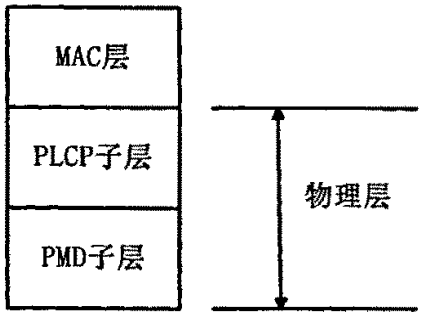


图 3-4 无线局域网的物理层结构

Fig.3-4 Physical layer structure of WLAN

(1) 物理管理层(Physical Layer Management)：物理层管理与 MAC 层管理相连，为物理层提供管理功能。

(2) 物理层汇聚子层(PLCP)：MAC 层和 PLCP 通过物理层服务访问点(SAP)利用原语进行通信。MAC 层发出指示后，PLCP 就开始准备需要传输的介质协议数据单元(MPDU)。 PLCP 也从无线介质向 MAC 层传递引入帧。PLCP 为 MPDU 附加的字段，字段中包含物理层发送器和接收器所需的信息 。PPDU 的帧结构提供了工作站之间 MPDU 的异步传输。PLCP 将 MAC 协议数据单元映射成适合被 PMD 传送的格式，从而降低 MAC 层对 PMD 层的依赖程度。

(3) 物理介质依赖((PMD)子层:在 PLCP 下方，PMD 支持两个工作站之间通过无线介质实现物理实体的发送和接收。为了实现这个功能，PMD 需直接面向无线介质(空气)，并为帧传送提供调制和解调。PLCP 和 PMD 之间通过原语通信，控制发送和接收功能。

3.2.2 IEEE802.11a 的 PLCP 子层

为了降低 MAC 层对物理介质依赖(PMD)子层的依赖程度，物理层收敛过程(PLCP)子层将 MAC 协议数据单元映射成适合被 PMD 子层传送的帧格式。因此研究 PLCP 子层的功能，也就可以研究 OFDM 在物理层的实现过程。

MAC 协议数据单元在 PLCP 层被封装上 PLCP 前导序列和 PLCP 适配头，形成 PLCP 协议数据单元(PPDU)。在接收机端，PLCP 前导序列和 PLCP 适配头被用于协助解调和传输 MAC 协议数据单元。

IEEE802.11a PLCP 子层帧结构如图 3-5 所示[24]，它的 PPDU 格式是 OFDM 物理层所特有的，包括：

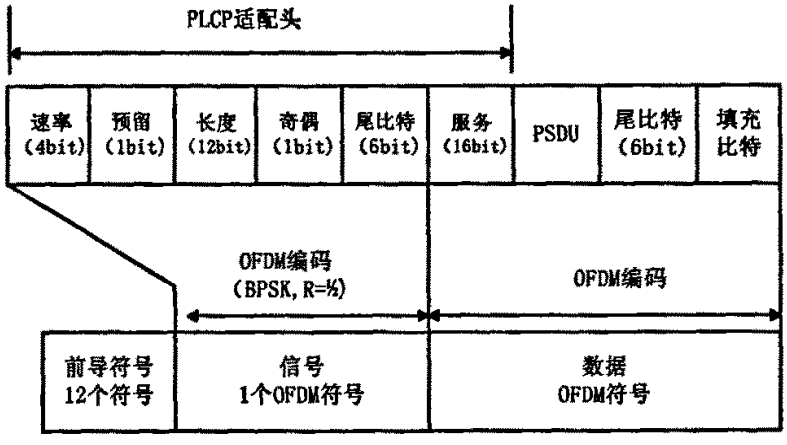


图 3-5 IEEE802.11a PLCP 帧结构

Fig.3-5 Frame structure of PLCP in IEEE802.11a

PLCP 前导序列(PLCP preamble)。这一部分用于获得引入的 OFDM 信号和序列，并使解调同步。PLCP 前导序列由 12 个训练序列组成，包括 10 个短训练序列和 2 个长训练序列。短训练序列用于接受机的自动增益控制，并粗略估计载波的

频率偏移，长训练序列用于精确估计频率偏移。12 个子载波用于短训练序列，53 个子载波用于长训练序列。一个 OFDM 周期位 16 毫秒。PLCP 前导序列采用 BPSK-OFDM 的调制方式，卷积编码率位 1/2，速率可达 6 Mbps。

信号(SIGNAL)。这一部分占 24 比特，包含了下列域：速率 4 bits、预留 1 bits、长度 12 bits、奇偶校验 1bits、尾比特 6 bits。其中速率中的编码信息包括分组中使用的调制方式和编码速率，详见表 3-3；长度域规定了 PSDU 的字节数，取值范围从 1～4095；尾比特用来刷新卷积编码器和中止解码器中的码网络。

数据(DATA)。这一部分包含服务域、PSDU 数据、尾比特和填充比特。服务域的前 7 bit 为 0，使用来初始化解扰码器的，剩余的 9 bit 保留作将来使用。6 个尾比特全是加在 PSDU 后的 0，以保证卷积编码器能回归到零状态。信号域指定了分组中的数据部分的传输速率。

表 3-3 IEEE802.11a 中的速率和调制参数

Tab.3-3 Parameters of rate and modulation in IEEE802.11a

比特	速率/Mbps	调制	编码 R	每子载波编码比特输 N_{BPSCS}	每 OFDM 符号编码比特数 N_{CBP}	每 OFDM 符号速率 N_{DBPS}
1101	6	BPSK	1/2	1	48	24
1111	9	BPSK	3/4	1	48	36
0101	12	QPSK	1/2	2	96	48
0111	18	QPSK	3/4	2	96	72
1001	24	16QAM	1/2	4	192	96
1011	36	16QAM	3/4	4	192	144
0001	48	64QAM	2/3	6	288	192
0011	54	64QAM	3/4	6	288	216

PLCP 适配头如图 3-5 所示

IEEE802.11a 中数据部分的比特数使每 OFDM 符号的编码比特数(48，96，192 或 288 bits)的整数倍，为此，信息的长度必须扩展为每 OFDM 符号的数据比特数的整数倍。因为还要添加尾比特，信息后面至少还要附加 6 个比特位。IEEE802.11a 中的 OFDM 符号数 N_{SYM}，数据部分的比特数 N_{DATA} 和填充比特数 N_{PAD} 可以从 PSDU 的长度计算得到

$$N_{SYM}=\left\lceil \frac{16+8*\mathrm{LENGTH}+6}{\mathrm{N_{DBPS}}} \right\rceil \tag{3.5}$$

$$N_{DATA}=N_{SYM}*N_{DBPS} \tag{3.6}$$

$$N_{PAD}=N_{DATA}-(16+8*\mathrm{LENGTH}+6) \tag{3.7}$$

速率、保留比特、长度、奇偶校验位和 6 个 0 尾比特单独构成了一个 OFDM 符号——标志符号，采用 BPSK 调制方式 1/2 的编码率传输。

PLCP 适配头的服务部分和 PSDU(包括 6 个 0 尾比特和附加比特)以适配头中指定的速率传输，并且使 OFDM 符号的整数倍。尾比特的作用使在接收到尾比特后可以立即对适配头中表示速率和长度的部分译码。速率和长度信息使对包的数据部分解码时所必须的。

IEEE802.11a 对 OFDM 物理层的编码过程给出了详细的规定，编码过程如下：

(1) 生成 PLCP 前导序列。此序列由 10 个重复的短序列和 2 个重复的加保护间隔(GI)的长序列构成。10 个短序列用来进行接收端的自动增益集中控制、分集选择、定时捕获以及完成频率的粗同步。长训练序列作用是在接收端进行信道估计以及进行系统的细同步。

(2) 生成 PLCP 适配头信息。根据发送端的数据位、长度和服务位，再加上适当的比特得到 PLCP 适配头。PLCP 中的 Rate 和 Length 经过 1/2 速率的卷积编码，映射成一个单独的 OFDM 符号，这与 Signal 符号的产生类似。为了能够及时检测 Rate 和 Length，采取在 PLOP 头插入 6 个“0”。由 Signal 得到一个的 OFDM 符号，要经过同样的过程:卷积编码、交织、BPSK 调制、插入导频、傅立叶变换，最后是加保护间隔使速率达到 6Mbit/s。Signal 部分不需要扰码。

(3) 根据发送端的 Rate，计算每个 OFDM 符号所包含的数据比特数(记为 N_{DBPS})。编码速率(R)，每个 OFDM 子载波中的比特数(N_{BPSC})，以及每个 OFDM 符号中已编码比特数(N_{CBPS})。

(4) 将 PSDU 加在服务(SERVICE)域的后面形成比特串，并在该比特串中添加适当的“0"(至少是 6 个)，使得比特流的长度 N_{DBPS} 的整数倍。这个比特串就构成了 PPDU 帧的数据部分。

(5) 用一个非零初值产生的伪随机序列进行扰码，产生一个不规则序列，然后域上述的已扩展数据比特串进行逻辑异或(XOR)处理。

(6) 用一个已经过扰码处理的 6 位“0” bit 替换未经扰码处理的 6 位“0” (这些 bit 叫做尾比特，它们使卷积编码器返回“零状态”)。

(7) 用卷积编码器对已扰码的数据串进行编码。按照特定的删除模式从编码器的输出串中剔掉一部分 bit，以获得需要的码率。

(8) 把以编码的比特串分组，每组含 N_{DBPS} 个 bit。按照所需速率对应的规则，对每组中的 bit 进行交织(Interleaving)处理。

(9) 再把已编码并做过重新排序处理的数据串进行分组，每组含 N_{DBPS} 个 bit。对每一个组，都按 IEEE802.11a 标准文本中给定的编码表转换成一个复数。

(10) 将调制后的复数信号按 48 为单位分成若干组，每一组可以形成一个 OFDM 符号。在每个组中，复数值分别为从 0 到 47，并分别被映射到已用数字标号的 OFDM 的子载波上。这些子载波的号码是： -26 到-22， -20 到-8, -6 到-1，1 到 6，8 到 20 以及 22 到 26。编号为-21，-7，7，21 被跳过，它们将被用作插入导频的子载波。代表中心频率的 0 号子载波可以忽略所以置为零。

(11) 导频插入编号为-21，-7，7 和 21 的 4 个子载波中。这样总的子载波是 52(48+4)。

(12) 每一组编号-26 到 26 的子载波经过傅立叶逆变换转为时域信号。将傅立叶变换后的波形作循环扩展，形成保护间隔(GI)，并采用时间截断的方法对每一个周期性的 OFDM 符号的波形范围进行加窗处理(windowing)。

(13) 以含有 Rate 和 Length 的 Signal 开始的 OFDM 符号流一个接着一个的进入信道传输。

(14) 按照所需信道的中心频率，把得到的“复数基带”波形向上变频到射频并发射。

3.2.2 IEEE802.11a PMD 子层

PMD 子层实现 PPDU 和无线电信号之间的转换，提供调制和解调功能。

IEEE802.11a 中采用的 OFDM 技术，该技术的调制解调原理已经在第二章中描述过，在这里就不再重复。

3.3 IEEE802.11g 标准的帧结构

IEEE802.11g标准[5][28][29]采用了同IEEE802.11a标准相同的OFDM技术应用2.4GHz ISM频段，分别对前导序列、适配头和负载进行调制，这种帧结构称为OFDM-OFDM方式。另外，IEEE802.11g标准规定了可选项和必选项，为了保障与11b兼容，也可采用CCK-OFDM和PBCC（包二进制卷积码)可选调制方式。

OFDM是一种经过验证的高速调制技术，在UNII频段上提供可达54Mbps的速率，它的吞吐量最高，覆盖范围大。然而CCK系统不能识别OFDM网络交换的信号，因此若将IEEE802.11b和基于2.4GHz的OFDM的版本混合安装就回破坏CSMA/CA协议。在此情况下，一种改进的OFDM方案可以解决这种交换问题，这就是CCK-OFDM。CCK-OFDM将CCK调制用于PPDU帧的前导序列和适配头，而将OFDM用于有效信息，这样就可以解决IEEE802.11b和IEEE802.11g混合的兼容问题，但却会降低吞吐速率。这种组合调制的数据速率在所有的通信距离上依然比IEEE802.11b快得多。

IEEE802.11g结合了IEEE802.11a和IEEE802.11b的基本特点，它解决了对于已安装了IEEE802.11b无线局域网设备而又想获得更高数据速率，IEEE802.11a又不兼容现有网络的用户的问题。

最后我们通过列表的方式对IEEE802.11b、IEEE802.11a、IEEE802.11g进行比较[30][31][32]，如表3-4和表3-5。

表3-4 IEEE802.11系列标准传输速率

Tab.3-4 Transmission rate in standard IEEE802.11 series

		IEEE802.11b@2.4GHz		IEEE802.11a@5GHz		IEEE802.11g@2.4GHz	
速率 Mbps	单/多载波	规定	可选	规定	可选	规定	可选
1	单	Barker				Barker	
2	单	Barker				Barker	
5.5	单	CCK	PBCC			CCK	PBCC
6	多			OFDM		OFDM	CCK-OFDM
9	多				OFDM		OFDM, CCK-OFDM
11	单	CCK	PBCC			CCK	PBCC
12	多			OFDM		OFDM	CCK-OFDM
18	多				OFDM		OFDM, CCK-OFDM
22	单						PBCC
24	多			OFDM		OFDM	CCK-OFDM
33	单						PBCC
36	多				OFDM		OFDM, CCK-OFDM
48	多				OFDM		OFDM, CCK-OFDM
54	多				OFDM		OFDM, CCK-OFDM

表 3-5 IEEE802.11 系列标准的比较

Tab.3-5 The comparison of standard IEEE802.11 series

标准	IEEE802.11b	IEEE802.11g	IEEE802.11a
工作频段	2.4GHz ISM 频段		5GHz UNII 频段
通信机制	DSSS(CCK)	PBCC, CCK-OFDM	OFDM
信道带宽	14 个，每个 22MHz	14 个，每个 22MHz	20MHz
数据速率	1, 2, 5.5, 11MHz	1，2，5.5，6，9，11，12，18，22，24，36，48，54MHz	6，9，12，24，36，48，54MHz
发射功率	100mW		40mW(5.15~5.25GHz), 200mW(5.25~5.35GHz), 800mW(5.725~5.825GHz)

3.4 本章小结

本章从物理层的角度分别对无线局域网的三大标准 IEEE802.11b，IEEE802.11a，IEEE802.11g 进行了详细系统的分析和论述，然后以表格的形式给出了三个标准之间的联系和异同，为以后的研究和学习提供一定的理论参考。

第四章 OFDM的计算机仿真

4.1 基于System View的仿真分析

4.1.1 System View软件简介

System View[33][34][35]是一个用于现代工程与科学系统设计及仿真的动态系统分析平台。从滤波器设计、信号处理、完整通信系统的设计与仿真，直到一般系统的数学模型建立等各个领域，System View在友好且功能齐全的窗口环境下，为用户提供了一个精密的嵌入式分析工具。它具有以下优点：1)、强大的仿真设计功能。b)、丰富的库资源。System View的基本库中包括多种信号源、接收窗、加法器、乘法器，各种函数（包括多项式、三角函数、对数函数、指数函数、逻辑函数等常用函数）运算器等；另外它还带有各种专业库如通信、逻辑、数字信号处理、射频/模拟等以备选择，特别适合于现代通信系统的设计、仿真和方案论证。c)、开放友好的用户界面。d)、灵活的硬件设计接口。除了一般的方案论证外，System View 还提供了与多种硬件设计工具的接口；与 Xinlinx 公司的软件 CORE Generator配套，可以将System View系统中的部分器件生成下载FPGA芯片所需的数据文件；通过与TI公司DSP设计工具CCS(Code Composer Studio)的接口，可以将其DSP库中的部分器件生成DSP芯片编程的C语言源代码，或在系统仿真中嵌入实际硬件电路；通过与Xpedion公司的射频/微波仿真工具的接口，可以将系统级仿真与电路级仿真结合起来，对分立元器件的射频/微波特性进行仿真。e)、智能化的辅助设计。f)、动态的分析和后台处理。

在Systemview中，具体的电路系统是由单独的模块组合而成的，这些模块称为Token，每一个Token都有自己定义的输入输出，以及相应的参数，以实现特定的功能。Systemview基本属于一个系统级工具平台，并配置了大量模块(Token)库，用户可以构造出所需要的仿真系统，只要调出有关图符块并设置好参数，完成图符块间的连线后运行仿真操作，最终以时域波形、眼图、功率谱、星座图和各类

曲线形式给出系统的仿真分析结果。

4.1.2 OFDM 调制与解调模型

在 Systemview 的通信库里有 OFDM 调制与解调模块，它的参数如表 4-1：

表 4-1 OFDM Token 参数

Tab.4-1 Token parameters of OFDM

Sample	N	每周期内的符号数
Symbol Time	T_S	符号周期
Guard Time	Δ	保护时隙

OFDM 调制模块的输入有两路，分别为同相信号C_{kr}和正交信号C_{ki}，输出分别为Z_{mr}和Z_{mi}，基本数学方程定义如下：

$$Z(t)=\sum_{m=0}^{N-1}C_k e^{2\pi(m-N/2)t/T_S}\quad,\qquad 0\le t\le T_S \tag{4.1}$$

$C_k=C_{kr}+jC_{ki}$为输入复数信号，$T_M=T_S-\Delta$是有用符号周期，输出信号也为复数信号$Z(t)=Z(m/T_M)=Z_m=Z_{kr}+jZ_{ki}$。

Systemview虽然自带了OFDM调制与解调器，但要将其与DSP技术结合起来，还需利用基本库与专业库中大量现有的模块及 Matlab 软件来搭建。下面介绍自己搭建的 OFDM 仿真系统，并将其与用 Systeview 自带的 OFDM 调制与解调器搭建的仿真系统相比较。

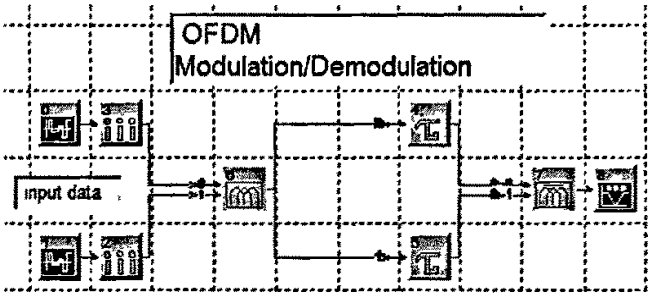


图 4-1 Systemview 中自带的 OFDM 搭建的仿真系统

Fig.4-1 Simulation system of OFDM from Systemview

表 4-2 参数设置表（图 4-1）

Tab.4-2 Parameters of Fig.4-1

Token 序号	Token 名称	Token 参数设置
0，1	信号源库，伪随机 PN 序列发生器	Amplitude=1,Rate=64Hz
2，3	算子库，采样器	Rate=64Hz
4，5	算子库，延迟模块	Delay=1sec
6	通信库，OFDM 调制器	N=64,T_S=1sec, Δ=0.2sec
7	通信库，OFDM 解调器	N=64,T_S=1sec, Δ=0.2sec
8	观察窗库，系统观察窗	无

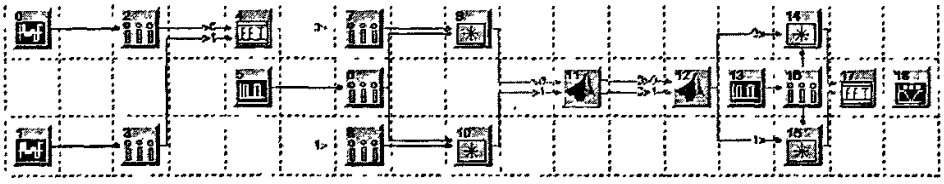


图 4-2 OFDM 仿真系统

Fig.4-2 Simulation system of OFDM

表 4-3 参数设置表(图 4-2)

Tab.4-3 Parameters of Fig.4-2

Token 序号	Token 名称	Token 参数设置
0，1	信号源库，伪随机 PN 序列发生器	Amplitude=1,Rate=64Hz
2，3	算子库，采样器	Rate=64Hz
4	数字信号处理库，IFFT 模块	N=64
5	信号源库，方波	Amplitude=64,frequency=32Hz
6，7，8	算子库，采样器	Rate=64Hz
9，10，14，15	乘法器库，乘法器	无
11，12	Matlab 连接图库，M-link 模块	详见论文中叙述
13	信号源库，方波	Amplitude=1,frequency=32Hz
16	算子库，采样器	Rate=32Hz
17	数字信号处理库，FFT 模块	N=64
18	观察窗库，系统观察窗	无

图 4-1 为 Systemview 中自带的 OFDM 调制与解调器搭建的仿真系统，参数设置如表 4-2；图 4-2 为自己搭建的仿真系统，参数设置如表 4-3。下面将对这两个仿真系统进行比较。因 N 点的 IFFT 与 IDFT 一样，每个频谱点都是由 N 个采样数据加权计算得到的，必须待 N 个数据都一次输入后才能开始计算各个频谱点的值，OFDM 理论上的数据先进行串并变换再并行调制在使用 DSP 的 IFFT 进行调制时已不再需要，因为进行 N 点的 IFFT 和 FFT 时相当于进行了 N 倍的周期扩展，这也是产生延时的一个原因。在自己搭建的仿真系统中(图 4-2)，Token0 和 1 是两路输入信号，为速率 64 的二电平伪随机码序列，信号幅度为 1，分别表示同相和正交信号的 QPSK 方式；Token2，3，6，7，8 和 16 是采样器，采样速率与信源相同；因 OFDM 的可用 IFFT 和 FFT 实现，只是系数不同和相位偏移，所以 Token4 和 17 是 DSP 库的 IFFT 和 FFT 模块，两个输入端的信号分别作为待变换信号的实部与虚部，实现 64 点 IFFT 与 FFT 变换，Token5 是幅度为 64 频率为 32 的方波，其与 IFFT 输出相乘来实现系数的变化和相位的偏移，而 Token13 是幅度为 1，频率为 32 的方波，将其与去保护间隔后的传输数据相乘，恢复成 IFFT 变换后数据以便更好的解调数据。

在 Systemview 的 Tools 菜单里，有一个 Systemview M-Link 选项，这个功能提供了 Systemview 与 Matlab 软件之间的可靠连接，Matlab 函数还可以作为信号源为其提供多种形式的信号，也可以作为功能模块进行信号处理，还可作为接收图符对 Systemview 的仿真结果进行检验和分析。用户可以利用 Matlab 及其工具定义的某些函数，编辑完成相应功能，设置参数等并在 Systemview 中调用，Systemview 的 Matlab 库可以包含 Matlab 中带有的所有成员函数，并可以利用 Matlab 进行矩阵(向量)运算。

利用 Systemview 的 M-Link 功能，可以方便的实现添加循环前缀和去循环前缀，Token11 和 12 便是 M-Link 模块，从 Systemview 的专业库中拖出 M-Link 的图标，双击它，出现名为 Systemview Matlab Library 的对话框，点击 Define 按钮，出现一对话框。左上角的 No.Inputs 表示输入信号的个数，input captions 组合框表示每个输入的名称，右边对应选项分别表示输出个数和名称，输入输出均以向量表示。因为此模块要实现加循环前缀，信号速率为 64，总符号周期 T_S =1 有用信

号周期 $T_U = T_S - \Delta = 0.8s$，循环前缀(保护间隔) $\Delta = 0.2s$ 占有用符号的 1/4，因此要复制每一周期后 16 个样值添加到前面作为循环前缀，所以输入输出分别定义为 In0，Inl 和 Out0，Outl 各分别代表同相和正交的信号，长度分别为 64 和 80。模块属性选择 General，它表示模块既有输入也有输出。再右边是模块参数定义，No.Parameters 设为零，表示无参数。中间的文本框用来写函数定义，它表示模块所要实现的功能。内容如下：

```
%Start SystemView Definition
FunctionSyntax='Out0(1:16,1)=In0(49:64,1);Out1(1:16,1)=In1(49:64,1);OutO(17:80,1)
=In0(1:64,1);Out1(17:80,1)=In 1(1:64,1);';
%End SystemView Definition
```

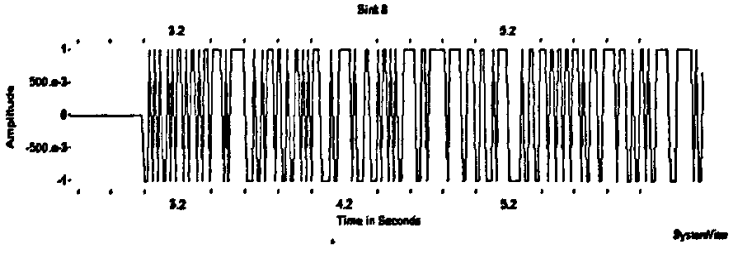
去循环前缀的程序为：

```
%Start SystemView Definition
FunctionSyntax='OutO(1:64,1)=In0(17:80,1);Out1(1:64,1)=In1(17:80,1);';
%End SystemView Definition
```

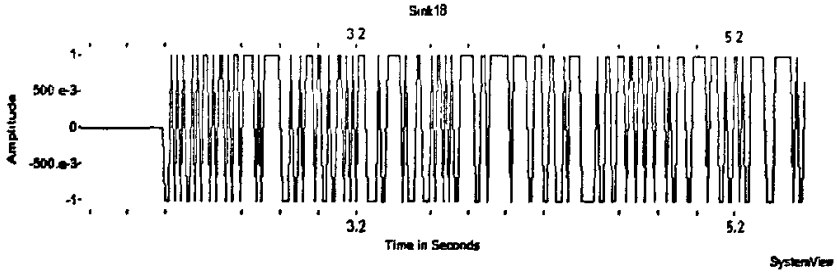
以上函数是完全按照 Matlab 的 M 文件格式编写，可以即时编写，也可以从外部加载进来，可见用 M-Link 来实现方便简单。

Token18 是系统型观察窗，运行结束后，不但在分析窗口可以显示波形在设计窗口内也可显示信号波形，便于观察。

系统参数设置为:采样速率：80Hz；采样点数：512 点；运行时间：6.3875s；循环次数：1 次。



(a)



(b)

图 4-3 波形图

Fig.4-3 Waveform

以同相信号为例，Sink8，18 分别为自带的 OFDM 调制解调解得到的信号波形图 4-3(a)和自搭建的 OFDM 调制解调得到的信号波形图 4-3 (b)。比较图 4-3 的解调信号，可得所搭建的 OFDM 仿真系统能够正确的调制与解调，只是有 4 个符号周期的延时，这种现象在实际电路中也是应该存在的。

4.1.3 无线局域网的仿真分析

如图 4-4 为 OFDM 系统原理框图，二进制数据进入后。经过添加填充位、交织、编码、映射、串并、IFFT 等进行调制发射出去。接收端经过相反的处理恢复原信号。

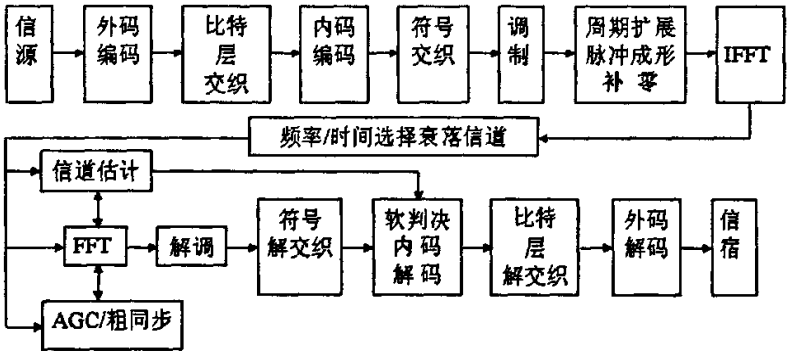


图 4-4 OFDM 系统原理框图

Fig.4-4 Block diagram of OFDM system

为了提高数字通信系统性能，信道编码和交织是通常采用的方法。对于衰落信道中的随机错误，可以采用信道编码；对于衰落信道中的突发错误，可以采用交织。实际应用中，通常同时采用信道编码和交织，进一步改善整个系统的性能。

在 OFDM 系统中，如果信道衰落不是太深，就不需要采用均衡技术了。因为 OFDM 系统自身具有利用信道分集特性的能力，一般的信道特性信息已经被 OFDM 这种调制方式本身所利用了。OFDM 系统的结构为在子载波间进行编码提供了机会，形成 COFDM 方式。编码可以采用各种码，如分组码、卷积码等。卷积码的效果要比分组码好。

参照图 4-4 我们用 SystemView 软件设计了下面的仿真图对无线局域网 OFDM 进行仿真分析。如图 4-5 所示。

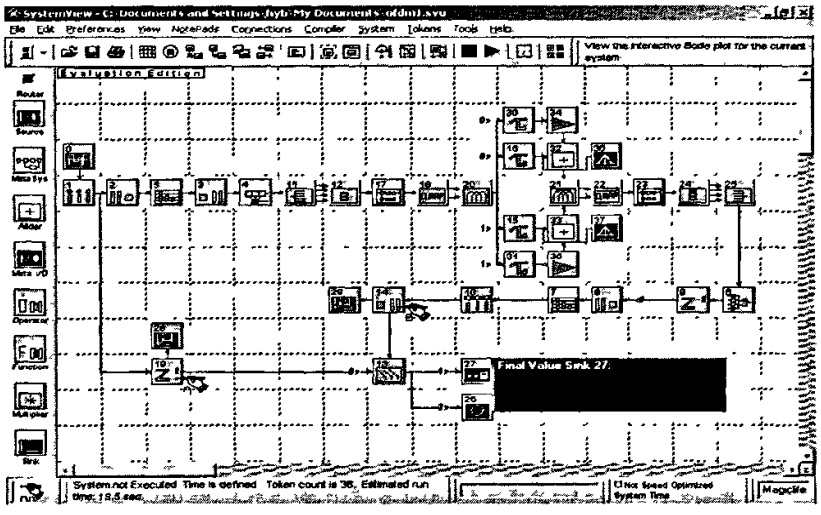


图 4-5 OFDM 仿真分析图

Fig.4-5 Analysis graph of OFDM simulation system

仿真中，系统以 Token0（即 PN 序列发生器）产生的伪随机序列为信号源。该伪随机码序列经 Token1（采样器）将数据速率变为每位有一个采样点，以便进行后续的编码等处理。采样后的比特流经 Token2（比特到符号的转换器）将每 8 位转换为一个符号，即完成比特流到字节流的转换。随后由 Token5（卷积交织器）完成卷积交织。为了进行编码外织后的字节流又经过 Token3 转换为比特流。该比特流经过 Token4 完成编码率为 R=1/2 的卷积编码。经过卷积编码后的比特流首先根据调制阶数及类型进行去复用。该功能由比特去复用 Token11 采用 16QAM 完成。去复用后的 4 个子码流分别输入 Token12（比特交织器）中，即 16QAM 调制的比特交织器，对 4 个子码流分别独立地进行比特交织。比特交织完成后，输出

的比特流由 Token17（交织器）完成数据交织，再由 Token18（符号映射器）完成符号映射。随后由 Token20 完成 OFDM 的调制功能。为了仿真系统的频道传输延迟，在两个通道中分别加入了一定的延迟量，并仿真了多径衰落信道，在这次仿真中没有加入高斯噪声。后续 Token21、22、23、24、25 构成解调模块，分别与调制模块的 Token20、18、17、12、11 相对应。Token21 为 OFDM 的解调器，Token22 为 OFDM 的逆映射器，Token23 为符号解交织器，Token24 为比特解交织，Token25 完成比特复用。最后由卷积解码器 Token6 对解调器输出的数字序列进行解码。系统输出由观察窗 Token29 进行观察,并用 Token33 来分析对比该系统的误码率。

上图中的各个模块即系统中各图标相应的参数设置如表 4-3 所列。

表 4-3 OFDM 仿真分析电路图标参数设置表
Tab.4-3 Parameters of OFDM simulation system in Fig.4-5

Token	Token 名称	Token 参数设置
0	信号源库，伪随机 PN 序列发生器	Amp=0.5v,Offset=0.5v,Rate=14.99294MHz,Levels=2, Phase=0°
1	算子库，采样器	Rate=14.9294MHz
2	通信库，比特到符号转换	MSB is first bit, Bits/Symbol=8,Threshold=0.5
3	通信库，符号转比特	MSB is first bit, Bits/Symbol=8
4	通信库，卷积码编码器	Code Length n=2, Info Bits k=1, Constraint L=7, Polynomial=(133)(171)o Threshold=0.5V
5	通信库，交织器	Mode=Interleave，Registers=12 sampls，Length=17 sampls, Offset=0 s
6	通信库，卷积码解码器	Hard Decision, Code Length n=2，Info Bits k=1, Constraint L=7, Polynomial=(133)(171)o, Path Length =15, Threshold=0.5，Offset=12610 bits
7	通信库，解交织器	Mode= De-Interleave, Registers=12 sampls,Length=17sampls,Offset=2367 bit
8	通信库，比特到符号转换	MSB is first bit, Bits/Symbol=8,Threshold=0.5
9	算子库，采样点延迟	Delay=10 samples
10	算子库，重采样器	Rate=1.866MHz

续表 4-3 OFDM 仿真分析电路图标参数设置表

Token	Token 名称	Token 参数设置
11	比特去复用	Modulation Type=2，Alpha Hierarchy Mode=1,Threshold=0V
12	比特交织器	Max Inputs=4, Alpha Hierarchy Mode=1
13	通信库，误码率计算	No.Trails=1 bits,Threshold=0.5v, Offset=25400 bits
14	通信库，符号转比特	MSB is first bit, Bits/Symbol=8
15,16	算子库，时间延迟器	Delay=4.08μs
17	通信库，数据交织器	Mode=Interleave, Rows=255 samples, Columns=255 samples
18	通信库，QAM 映射器	Constellation=16,ExternalFile=C:\ProgramFiles\SystemView\Examples\16QAM.tx
19	算子库，采样点延迟	Delay=37016 samples
20	通信库，OFDM 调制器	Samples Per Block=1024, Symbol time=1s, Guard Time=0s,
21	通信库，OFDM 解调器	Samples Per Block=1024, Symbol time=1s, Guard Time=0s,
22	通信库，QAM 逆映射器	Constellation=16,ExternalFile=C:\Progrm Files\SystemView\Examples\16QAM.txt
23	通信库，数据解交织器	Mode=De-Interleave, Rows=255 samples, Columns=255 samples
24	比特解交织器	Modulation Type=2, Input delay=4.08μs
25	比特复用	Alpha Hierarchy Mode=1, Input delay=4.08μs
26	观察窗库，控制终止观察窗	Action= Go To Next Loop,Memory=Retain last sample,
27	观察窗库，运行终值观察窗	无
28,29	观察窗库，分析观察窗	无
30,31	算子库，时间延迟器	Delay=4.08ms
32,33	加法器	无
34,35	算子库，增益	Gain=0.5
36,37	信号源库，高斯噪声	Mean=1v
时钟	开始时间 0s， 采样频率 32.4MHz，	

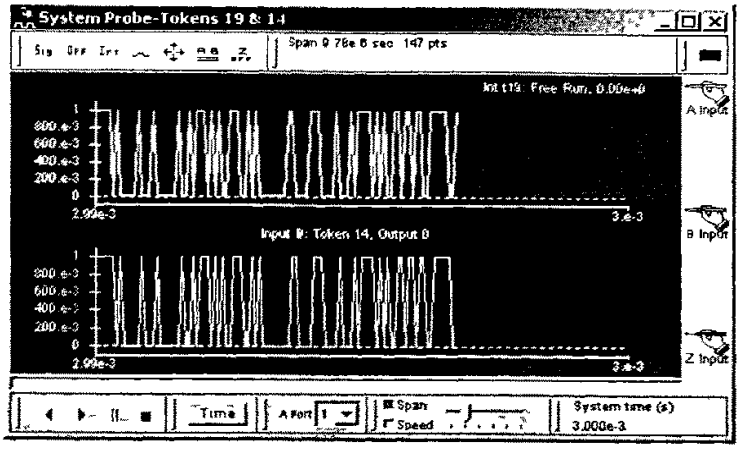


图 4-6 原信息信号与调制解调后的输出波形比较

Fig.4-6 Comparison of waveform between original signal and signal after modulating and demodulating

信源经调制、解调后的输出与原来的信源一同送到误码率 Token7 中。为了保证两路信号的同步，原信息信号经过了适当的延迟。运行系统，误码率计算的输出小于 10^{-5}。也就是说，信号编码、交织以及相应的解码、解交织后，基本上恢复了原来的信息信号。同时，输入和输出两路信号分别通过探头 A、B 由探头示波器输出观察，运行结果如图 4-6 所示(取 SNR=20dB)。对波形进行比较可见，解调输出波形与发送端信息数据波形基本一致。该系统正确完成了信息数据的调制、发送和解调功能。

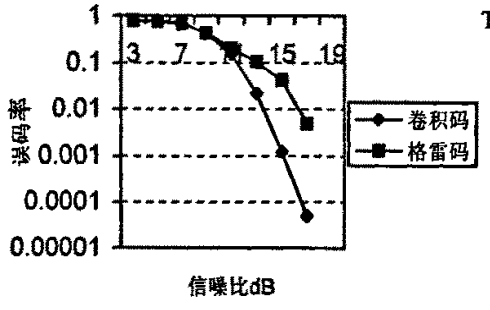


图 4-7 使用卷积编码和格雷码编码误码率比较图

Fig.4-7 Comparison of BER between using convolutional code and Gray code

表 4-4 仿真数据（图 4-8）

Tab4-4 Simulation data of Fig4-8

	卷积码	格雷码
3	0.8012	0.7804
5	0.7802	0.7632
7	0.7031	0.6831
9	0.4222	0.4232
11	0.1524	0.2024
13	0.0213	0.1013
15	0.0012	0.0412
17	0.00005	0.005

由于在 IEEE802.11a/g 中没有给出具体的信道编码方式，为了进行比较，我们

采用了两种常用的编码方式卷积码和格雷码。通过仿真分析，得到的结论是当信噪比较大的时候，采用卷积编码的系统比采用格雷码编码的误码率要低。表 4-4 是不同信噪比下两者的误码率的仿真数据，比较分析图如图 4-7 所示。

4.2 基于 MATLAB 的仿真分析

4.2.1 MATLAB 简介

MATLAB 是一种强大的工程计算软件，是一种功能强、效率高、便于进行科学和工程计算的交互式软件包。其工具箱中包括：数值分析、矩阵运算、通信、数字信号处理、建模和系统控制等应用工具程序，并集应用程序和图形于一体，便于应用到集成环境中。在此环境下所解问题的 MATLAB 语言表述形式和其数学表达形式相同，不需要按传统的方法编程。MATLAB 的特点是编程效率高，用户使用方便，扩充能力强，语句简单，内涵丰富，具有高效方便的矩阵和数组运算和方便的绘图功能。

4.2.2 OFDM 通信系统仿真设计说明

设计的实现如果选择用 MATLAB 的仿真工具箱 SIMULINK 来实现，由于 SIMULINK 中高度的集成模块与本章上一节中 System View 软件里的集成模块有相似之处，将使得我们仍然不能对整个系统的实现有较为完整的了解。所以本节的 OFDM 调制解调通信系统决定采用 MATLAB 语言程序来设计。使用通信工具箱里面的通用函数，可以仿真 OFDM 系统中的某些组成部分，然后把形成的每一个部分级联起来就构成了整个系统。但是为了系统的连贯性，有多个函数还是选择自己编写，如 BPSK、QPSK、16QAM 的调制与解调函数，串并及并串转换函数等。具体系统的设计流程如图 4-8 所示，MATLAB 语言程序源代码见附录。

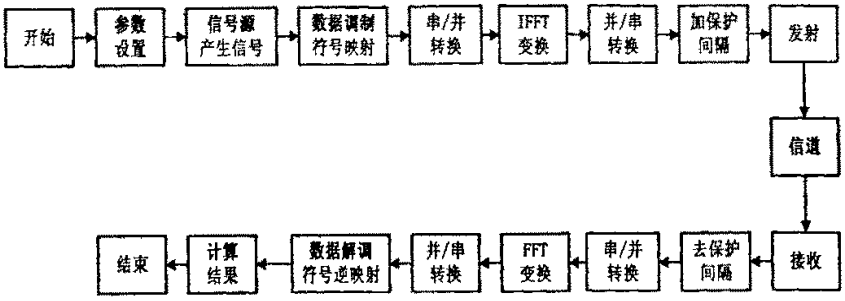


图 4-8 OFDM 通信系统仿真程序流程图

Fig4-8 Simulation flowchart of OFDM system

主程序中的几点说明：1)、考虑到上射频后，射频频率较高，由抽样定理，要恢复信号必须采用更高的采样频率，在实际编程中，大大降低了程序运行效率，使得仿真所需的时间无法忍受。可以证明以射频为中心频率的带通信号可以用与之相对应的基带信号等效。因此不进行射频调制和解调并不影响对系统性能的评估。在信道模拟部分人为的加入了信道白噪声和一路多径时延信号。白噪声的强度由信噪比这个参数来确定。而多径时延信号的时间延时量由时延参数来确定。在通过了加入白噪声后的信道后，接收端应该加入数字滤波器提高接收信号的信噪比，具体见附录程序。2)、系统采用的调制方式可以为 BPSK,QPSK,16QAM，主程序在执行时分别调用其子程序，附录程序中显示调用的是 16QAM。3)、采用的 IFFT 模块和 FFT 模块，IFFT 和 FFT 函数所采用的离散傅立叶计算的点数由采用的子载波的数目决定。

4.2.3 仿真数据分析

1. 调制方式对信噪比和误码率的影响

在仿真程序中保持载波数为 256 和保护间隔为 16 不变，分别用 BPSK、QPSK 和 16QAM 对输入信号进行调制，来测试不同信噪比下 OFDM 系统的误码率，仿真得到的数据如表 4-5, 分析比较见图 4-9。

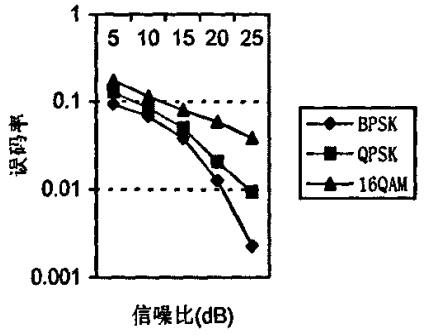


图 4-9 不同调制样式下误码率随信噪比变化图
Fig4-9 The graph of BER and SNR in different modulation style

表 4-5 仿真数据（图 4-9）
Tab4-5 Simulation data of Fig4-9

SNR	BPSK	QPSK	16QAM
5	0.09326	0.12872	0.17317
10	0.06775	0.08394	0.11353
15	0.03806	0.05063	0.08014
20	0.01263	0.02077	0.05973
25	0.00231	0.00946	0.03876

由实验数据可以看到同一信噪比下，BPSK 和 QPSK 的误码率要比 16QAM 的小。使用 16QAM 作为调制方式时，它对信噪比的最低容忍度是 27dB 至 30dB，在信噪比降为 25dB 时，它的误码率就会大大增加了，然而使用 BPSK 或 QPSK 能把信噪比的最低容忍度降到 23dB 至 25dB。所以，若对误码率有严格的要求，并且不需要很大的数据传输，那么只能选择 BPSK 或 QPSK 的 OFDM 调制方式；若需要传输数据量大的数据，16QAM、64QAM 是很好的选择。

2. 载波数，FFT 的长度对信噪比和误码率的影响

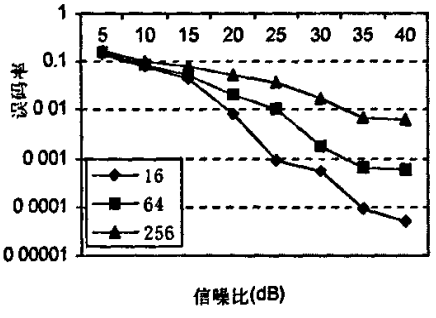


图 4-10 不同载波数下误码率随信噪比变化图
Fig4-10 The graph of BER and SNR in different carrier number

表 4-6 仿真数据（图 4-10）
Tab4-6 Simulation data of Fig4-10

SNR	16	64	256
5	0.14843	0.14972	0.17033
10	0.07796	0.08564	0.10483
15	0.04196	0.05193	0.08044
20	0.00808	0.01967	0.05036
25	0.00087	0.00996	0.03553
30	0.00054	0.00169	0.01694
35	0.00009	0.00062	0.00653
40	0.00005	0.00058	0.00641

在仿真程序中保持信号的调制样式 16QAM 和保护间隔为 16 不变，通过载波数分别取 16、64 和 256 来测试不同信噪比下 OFDM 通信系统的误码率，仿真得到

的数据如表 4-6, 分析比较见图 4-10。

由以上数据可以看到当子载波的数目增加时，在同一个信噪比的条件下，其误码率也会增加。这就决定了不能为了克服多径干扰而盲目地增加子载波数目。

3. 保护间隔δ对信噪比和误码率的影响

在仿真程序中保持信号的调制方式 16QAM 和载波数 256 不变，保护间隔δ分别取 8，16，32 来测试不同信噪比下 OFDM 通信系统的误码率，仿真得到的数据如表 4-7, 分析比较见图 4-11。

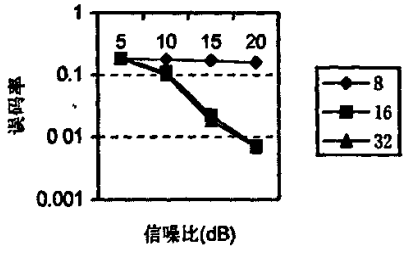


图 4-11 不同保护间隔下误码率随信噪比变化图
Fig4-11 The graph of BER and SNR in different protection partition

表 4-7 仿真数据（图 4-11）
Tab4-7 simulation data of Fig4-11

SNR	8	16	32
5	0.18382	0.18271	0.18262
10	0.17501	0.11073	0.09964
15	0.16673	0.02189	0.01796
20	0.15859	0.00701	0.00691

对一个给定的N，增加δ可以减小N个保存在接收器中的抽样失真。因此，增加δ减小了载波间干扰(ICI)和码间干扰(ISI)。另一方面，增加δ减小了功率效率，其减小因子为$N/(N+\delta)$，由于接收器仅仅保存了$N+\delta$个信号中的N个信号。这表明了保护间隔对功率的损失具有一定的影响。由图 4-11 可以看出保护间隔为 16 和 32 时误码率差别不大，所以建议将保护间隔设置为 16。

4. 多径信道中 OFDM 通信系统的性能分析

在前面的仿真分析中，信道里只是加入了高斯白噪声，没有加入多径干扰，接下来我们分析一下 OFDM 系统在多径干扰下的性能。

根据现实中的信噪比，将系统信噪比 SNR 分别设定为 5、10、20、30、40、50dB，用 MATLAB 软件仿真由于多径信道的影响而产生的误码率情况。仿真中分别用 BPSK、QPSK 和 16QAM 进行调制信号，在不同的信噪比的情况下分别计算它们在多径条件下的误码率，我们用反射信号来作为多径信号进行测试，反射信

号的幅值要比原信号弱 10dB，图 4-12、4-13、4-14、4-15、4-16、4-17 分别为 SNR =5、10、20、30、40、50dB 时多径对误码率影响的曲线。

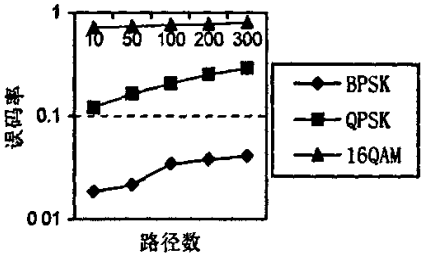


图 4-12 SNR = 5dB 时多径对误码率的影响
Fig4-12 Paths to influence of BER when SNR = 5dB

表 4-8 仿真数据（图 4-12）
Tab4-8 Simulation data of Fig4-12

	BPSK	QPSK	16QAM
10	0.0185	0.1203	0.7217
50	0.0216	0.1639	0.7405
100	0.0347	0.2050	0.7621
200	0.0380	0.2507	0.7805
300	0.0412	0.2894	0.8076

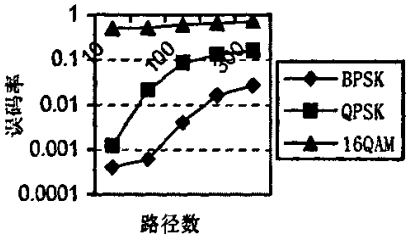


图 4-13 SNR = 10dB 时多径对误码率的影响
Fig4-13 Paths to influence of BER when SNR = 10dB

表 4-9 仿真数据（图 4-13）
Tab4-9 Simulation data of Fig4-13

	BPSK	QPSK	16QAM
10	0.0004	0.0012	0.5021
50	0.0006	0.0209	0.5205
100	0.004	0.0850	0.6062
200	0.016	0.1315	0.6508
300	0.0271	0.1608	0.7076

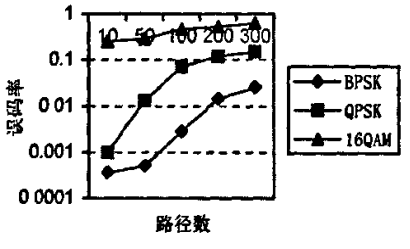


图 4-14 SNR = 20dB 时多径对误码率的影响
Fig4-14 Paths to influence of BER when SNR = 20dB

表 4-10 仿真数据（图 4-14）
Tab4-10 Simulation data of Fig4-14

	BPSK	QPSK	16QAM
10	0.00036	0.0010	0.2531
50	0.00052	0.0132	0.2894
100	0.0029	0.0713	0.4731
200	0.0143	0.1209	0.5270
300	0.0255	0.1517	0.6374

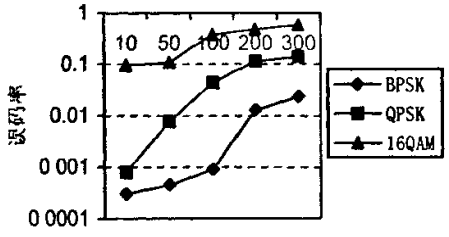


图 4-15 SNR = 30dB 时多径对误码率的影响

Fig4-15 Paths to influence of BER when SNR = 30dB

表 4-11 仿真数据（图 4-15）

Tab4-11 Simulation data of Fig4-15

	BPSK	QPSK	16QAM
10	0.0003	0.0008	0.0962
50	0.00046	0 0076	0.1083
100	0 0009	0.0435	0 3876
200	0.0128	0.1128	0.4837
300	0.0234	0.1386	0.5923

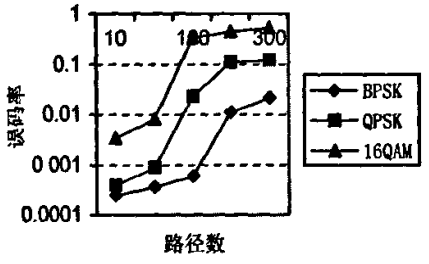


图 4-16 SNR = 40dB 时多径对误码率的影响

Fig4-16 Paths to influence of BER when SNR = 40dB

表 4-12 仿真数据（图 4-16）

Tab4-12 Simulation data of Fig4-16

	BPSK	QPSK	16QAM
10	0.00025	0.0004	0.0034
50	0.00037	0.0009	0.0082
100	0.0006	0.0227	0.3317
200	0.0112	0.1099	0.4492
300	0.0211	0.1202	0.5419

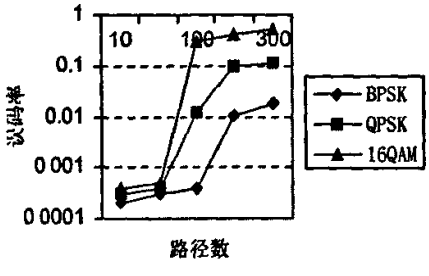


图 4-17 SNR = 50dB 时多径对误码率的影响

Fig4-17 Paths to influence of BER when SNR = 50dB

表 4-13 仿真数据（图 4-17）

Tab4-13 Simulation data of Fig4-17

	BPSK	QPSK	16QAM
10	0.0002	0.0003	0.0004
50	0.0003	0.0004	0.0005
100	0.0004	0.0124	0.3062
200	0.0102	0.1015	0.4108
300	0.0188	0.1168	0.5104

上面这几幅曲线图可以说明，OFDM 系统的调制方式的选择取决于所传输的数据量和系统对误码率的要求。BPSK、QPSK 调制能使系统在噪声比较大的条件下具有较低的误码率，然而它是以有限的数据传输量为代价的。采用 16QAM 则正好相反，如果系统对误码率要求不太严格，而且需要传输大数据量的数据时，建

议系统采用 16QAM 调制。

4.3 本章小结

本章首先采用仿真软件分析了 OFDM 技术在无线局域网中的应用，通过仿真得到的结论是当信噪比较大的时候，采用卷积编码的系统比采用格雷码编码的误码率要低。

本章还研究了输入信号信噪比、载波数目 N 和保护时间δ对 OFDM 系统性能的影响。主要得到以下结论：

1)、在信噪比较好的情况下可以选用 MQAM 映射样式；在信噪比较差时可以选用 BPSK 或 QPSK 的映射样式。而且，OFDM 系统的各个载波可以根据信道的条件来使用不同的调制，比如 BPSK、QPSK、16QAM、64QAM 等等。

2)、载波数目的最佳值随着传播时延的增加而增加，同时随着信噪比的减小而减小。这就说明了不能盲目地增加子载波数目

3)、对一个给定的N，增加δ可以减小N个保存在接收器中的抽样失真。因此，增加δ减小了载波间干扰(ICI)和码间干扰(ISI)。另一方面，增加δ减小了功率效率，其减小因子为$N/(N+\delta)$，由于接收器仅仅保存了$N+\delta$个信号中的N个信号。这表明了保护间隔对功率的损失具有一定的影响。建议将保护间隔设置为 16。

对 OFDM 系统进行仿真的目的就是为了在不同系统需求和传输条件下分析系统的特性，了解各种非理想因素对 OFDM 系统的影响，这对于应用 OFDM 传输技术的标准的制定以及相应系统的实现具有重要参考价值。

第五章 FFT 的 FPGA 实现

由于快速傅立叶变换 FFT 是 OFDM 系统的核心组成部分，所以本章首先介绍实现快速傅立叶变换的按时间抽取基 2 算法，然后详细介绍具体的硬件设计。

5.1 快速傅立叶变换

5.1.1 FFT 算法基本思想

FFT[36][37]算法的基本思想：可以将一个长度为 N 的序列的离散傅里叶变换逐次分解为较短的离散傅里叶变换来计算，这些短序列的 DFT 可重新组合成原序列的 DFT，而总的运算次数却比直接的 DFT 运算少得多，从而达到提高速度的目的。

快速傅立叶变换就是利用W_N^{nk}的特性，逐步地将 N 点序列分解成较短的序列，计算短序列的 DFT，然后组合成原序列的 DFT，使运算量显著减少。这种分解基本上可分为两类，一类是将时间序列 x(n)进行逐次分解，称为按时间抽取算法(Decimation In Time)；另一类将傅立叶变换序列 X(k)进行分解，称为按频率抽取算法(Decimation In Frequency)。本章主要介绍了按时间抽取基-2 FFT 算法。

5.1.2 按时间抽取基 2FFT 算法

我们已经知道 FFT 算法主要是利用W_N^{nk}的性质，通过把序列逐渐分解为短序列实现运算量的减少。W_N^{nk}的以下三种性质在 FFT 运算中得到了应用：

性质 1：W_N^{nk}的周期性　　$W_N^{nk} = W_N^{n(k+N)}$

性质 2: W_N^{nk}的对称性　　$(W_N^{nk})^* = W_N^{-nk}$

性质 3: W_N^{nk}的可约性　　$W_N^{nk} = W_{mN}^{mnk}$，　$W_N^{nk} = W_{N/m}^{nk/m}$

由于目前 FFT 比较普遍使用的算法还是基-2 算法，本节将讨论基-2 的按时间抽取算法。

基-2 算法中，序列 x(n)的长度 N 为 2 的整数次幂，即 $N=2^M$，其中 M 为正整数。最初通过将 x(n)分解为奇数项序列和偶数项序列的形式使 FFT 运算分为两组。

设： $x(2r)=x_1(r)$

$x(2r+1)=x_2(r)$ $\quad r=0,\ 1,\ \ldots,\ N/2-1$

设 $X_1(k)$为 $x_1(r)$的 DFT，$X_2(k)$为 $x_2(r)$的 DFT，利用 W_N^{nk} 的性质可得 x(n)的运算为：

$$X(k)=X_1(k)+W_N^k X_2(k)$$

$$X(k+N/2)=X_1(k)-W_N^k X_2(k) \qquad k=0,\ 1,\ \ldots,\ N/2-1$$

上面式子的运算可用图 5-1 的蝶形信号流图符号表示：

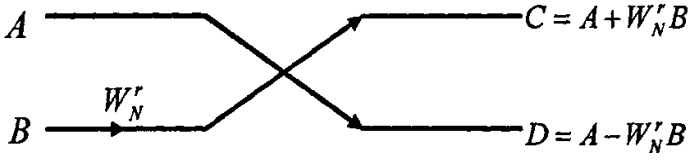


图 5-1 蝶形运算流图符号

Fig.5-1 The symbol of butterfly operation

由此可见，一个 N 点 DFT 分解为两个 N/2 点的 DFT，从而实现了运算量的减少，再经过逐次分解最终分解为 2 点的 DFT，实现了 FFT 运算。

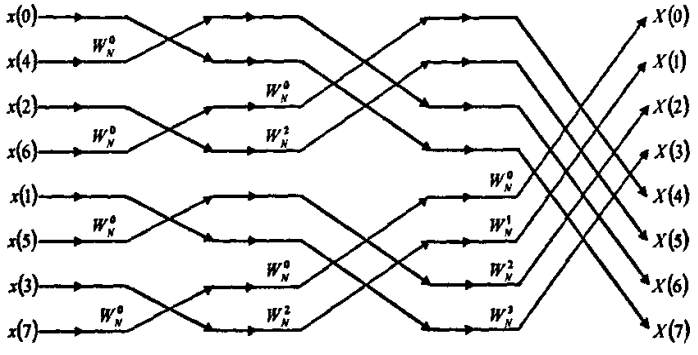


图 5-2 N=8 的时间抽取法 FFT 运算流图

Fig.5-2 DIT FFT operation when N=8

FFT 运算的核心是蝶形运算。通过顺序计算全部蝶形实现 FFT 算法。图 5-2 给出 N=8 时的按时间抽取 FFT 流图。

对于序列的长度 $N=2^M$ 的 FFT，共有 M 级蝶形，每级由 N/2 个蝶形运算组成，每个蝶形包括一次复乘、二次复加，则 M 级运算的运算量为

复数乘法：$\frac{N}{2}\times M=\frac{N}{2}\times\log_2 N$

复数加法：$N\times M=N\times\log_2 N$

由此可见，FFT 算法与直接 DFT 算法相比运算量大为减少，如 N=1024 时，DFT 所需的复数乘法运算次数为：N^2 =1048576 次，而 FFT 所需的复数乘法运算次数仅为 $\frac{N}{2}\times\log_2 N=5120$ 次，可见 N=1024 时 DFT 算法的运算量是 FFT 算法的运算量 $N^2\Big/\frac{N}{2}\log_2 N=1048576/5120=204.8$ 倍。从而可以看出 FFT 算法的优越性，且当点数 N 越大越能突出 FFT 算法的优越性。

5.1.3 按时间抽取 FFT 算法的特点

图 5-2 所示流图描述了计算离散傅里叶变换的一种算法。在图 5-2 中特别重要的是连接节点的支路和每个支路的传输比。只要节点间的连结和连结的传输比维持不变，则无论在流图中的诸节点如何重新排列，它始终表示相同的计算。由图 5-2 可以推知按时间抽取算法在运算方式上的特点，主要有以下两种特点:同址运算，倒位序规律。以下分别介绍这两种特点：

特点 1：同址运算

由图 5-2 可以看出，对于长度为 N 的序列，每一级计算均需要一组 N 个复数，并通过图 5-1 形式的基本蝶形把它们变换为下一组的 N 个复数，这种过程重复 $\log_2 N$ 次，最后得到所要求的离散傅立叶变换。当实现图 5-2 中描述的运算时，我们可以想象使用两列(复数的)存贮寄存器，一列存贮要计算的数，另一列存贮计算中要用到的数据。例如，当计算图 5-2 中的第一列时，第一组存贮寄存器当存放输入数据，而第二组存贮寄存器应当存放第一级计算出的结果。把第 m 级计算得出的复数序列记作 $X_m(n)$，其中 n=0，1，…，N-1，。对于第 m 级计算，可以认为 $X_{m-1}(n)$

是输入列，$X_m(n)$是输出列。这样，对于图 5-2 中所示的 N=8 的情况，有把从第 1 级输入的复数序列记作 $X_0(n)$：

$$\begin{aligned} X_0(0) &= x(0) \\ X_0(1) &= x(4) \\ X_0(2) &= x(2) \\ X_0(3) &= x(6) \\ X_0(4) &= x(1) \\ X_0(5) &= x(5) \\ X_0(6) &= x(3) \\ X_0(7) &= x(7) \end{aligned} \tag{5.2}$$

利用这种表示法，可将图 5-1 中蝶形计算的输入和输出标记为图 5-3 所示的那样，并有相应的方程

$$X_m(p) = X_{m-1}(p) + W_N^r X_{m-1}(q) \tag{5.3 a}$$

$$X_m(q) = X_{m-1}(p) - W_N^r X_{m-1}(q) \tag{5.3 b}$$

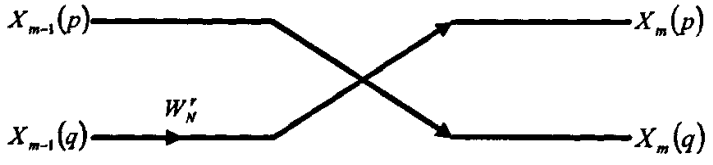


图 5-3 式 5-2 的流图

Fig.5-3 The flowchart of (5.3)

在式(5.3)中，p，q 和 r 从一级到另一级是不同的，从图 5-2 和图 5-3 中可以清楚地看出，要计算第 m 列的 p 和 q 位置上的复数节点值，只需要第(m-1)列在 p 和 q 位置上的复数节点值。因此，若将 $X_m(p)$和 $X_m(q)$ 分别存放在原存放 $X_{m-1}(p)$和

$X_{m-1}(q)$的同一存贮寄存器中，则实现全部计算实际上只需要一列存贮 N 个复数的寄存器。这种计算通常称为同址计算。

由此可见，采用同址运算只需 N 个存储单元，大大节省了存储单元，从而降低了设备成本。

特点 2：倒位序规律

为了实现同址计算，输入序列不能按照原来的先后顺序存贮(或至少不能这样读取)，而应如图 5-2 的流图那样，实际上，这种输入数据存贮和读取的顺序称为倒位序。我们注意到，对于已经讨论过的 8 点流图，只需要用三位二进制码来标注整个数据。如果用二进制形式写出式(5.2)中的标号，就得到如下一组式子：

$$X_0(000)=x(000)$$

$$X_0(001)=x(100)$$

$$X_0(010)=x(010)$$

$$X_0(011)=x(110)$$

$$X_0(100)=x(001)$$

$$X_0(101)=x(101)$$

$$X_0(110)=x(011)$$

$$X_0(111)=x(111)$$

若(n_2,n_1,n_0)为序列$x(n)$中标号的二进制表示，则序列值$x(n_2,n_1,n_0)$存放在数列$X_0(n_0,n_1,n_2)$的位置上.也就是说，要确定$x(n_2,n_1,n_0)$在输入序列中的位置，我们必须将标号 n 的位序颠倒。

表 5-1 给出了 N=8 时的自然顺序二进制数以及相应的倒位序二进制数。

表 5-1 码位的倒位序 (N=8)

Tab.5-1 The converse sequence of code position

自然顺序(n)	二进制码	倒位序二进制码	倒位序数
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

实际运算中，总是先按自然顺序将输入序列存入存储单元，要实现 FFT 算法，首先将按自然顺序存放的序列经变址运算变换得到倒位序的排列。

5.2 8 点 32 位快速傅立叶变换的 FPGA[38][39][40][41]实现

FFT 处理器的操作过程主要分为三个过程：数据输入、FFT 计算、和数据输出过程。其划分如下图 5-4 表示[42][43][44]：

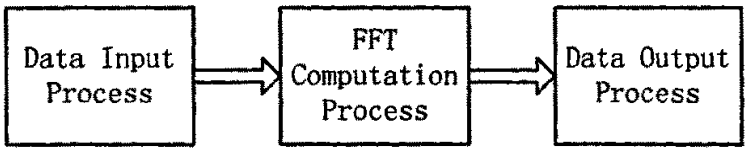


图 5-4 FFT 处理器的操作过程

Fig5-4 The operation of FFT processor

在数据输入阶段，数据被读进和存储到存储器中，在输入阶段数据需要进行倒序处理；在 FFT 计算阶段，FFT 计算内存中的数据；在数据输出阶段，FFT 计算结果被输出。

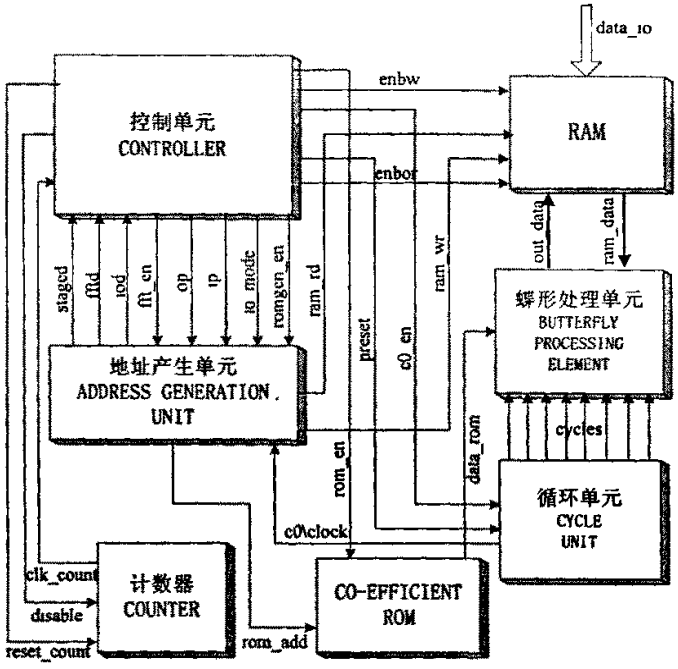


图 5-5 FFT 处理器的结构图

Fig5-5 Block diagraph of FFT processor

FFT 处理器具体结构图 5-5 所示，主要包括基 2 的蝶形运算单元、双口 FIFO RAM、存储旋转因子系数的 ROM、控制单元、地址产生单元和循环单元（用来生成 c0, c1, c2 和 c3 等控制信号）等模块。从读第一个数据开始需要 5 个时钟的时间才有输入写入 RAM，还需要记数器来记录周期的循环数。

以下为图 5-5 中 FFT 处理器的重要信号说明。

staged：高电平有效，表示 FFT 运算一个阶段的完成，

fftd：高电平有效，表示 FFT 运算完成。

iod：高电平有效，表示输入、输出结束。

fft_en：高电平有效，地址产生器的使能信号。

io_mode：高电平有效，表示输入/输出过程进行中。

op：高电平有效，表示输出。

ip：高电平有效，表示输入。

romgen_en：高电平有效，ROM 地址产生器的使能信号。

ram_rd:：RAM 读地址。

ram_wr:：RAM 写地址。

enbw：RAM 的可写使能。

enbor：RAM 的可读使能。

out_data：写入 RAM 的数据。

data_ram：从 RAM 输出到蝶形运算单元的数据。

cycles：包括时钟信号 c0,c1,c2,c3,c0_c1,c2_c3,c0_c2,c1_c3。

CLOCK 主时钟、INITIAL 初始信号、RESET 复位信号、ENBLE 使能信号等系统时钟和控制信号图中没有表示。

下面对 FFT 处理器的重要组成部分进行硬件实现。

5.2.1 蝶形运算单元

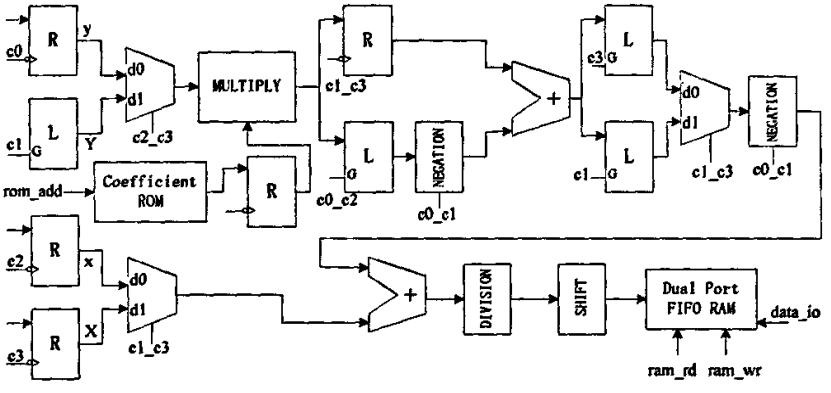


图 5-6 蝶形运算单元的结构图

Fig5-6 Block diagraph of butterfly processing unit

蝶形运算单元是 FFT 处理器的基本组成部分，其主要功能是从内存取两个数进行蝶形运算，结果再写回内存。结构采用深度为 4 的流水线结构，每 4 个时钟周期进行一次蝶形运算，包括一个乘法和两个加法运算。图 5-6 为蝶形运算单元的结构图。主要包括 9 个触发器模块(R 和 L)、3 个多路复用器模块、两个反向器模块(NEGATION)、两个加法器模块、一个乘法器模块、一个分配器模块(DIVISION)。乘法器模块进行复数运算的部分积运算，输出 32 位的结果。第一个加法器模块对相关的部分积进行和运算，第二个加法器模块的输出为蝶形单元的输出。R 模块是由 32 个下降沿 D 触发器构成，L 模块则是由 32 个上升沿 D 触发器构成，每个代

表 1bit。在进行 FFT 计算时 c0， c1，c2 和 c3 分别是一个周期内的四个控制信号。c0_c1 是信号 c0，c1 的或运算，同理，c0_c2 是 c0，c2 的或运算，如图 5-7 所示。

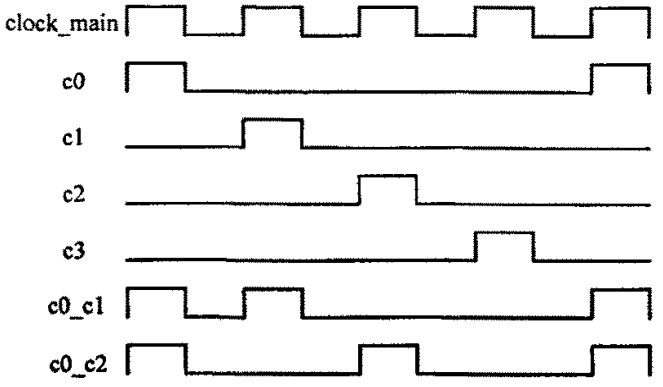


图 5-7 蝶形运算单元控制信号的波形图

Fig5-7 Waveform of cycles used in the butterfly processing unit

表 5-2 周期中不同时序蝶形运算单元的输出数据

Tab5-2 Output data of butterfly processing unit in different cycles

Cycle	RAM Read	ROM Read	Multip Output	First Adder Output	2nd Adder Output	Ram Write
c0	y	$\cos\phi$	prev ysinϕ	settling	prev y'	prev X'
c1	Y	$\sin\phi$	ycosϕ	prev Ycosϕ -ysinϕ	prev Y'	prev y'
c2	x	$\cos\phi$	Ysinϕ	settling	prev x'	prev Y'
c3	X	$\sin\phi$	Ycosϕ	ycosϕ +Ysinϕ	prev X'	prev x'
c0	next y	$\cos\phi$	ysinϕ	settling	y'	prev X'
c1	next Y	$\sin\phi$	next ycosϕ	Ycosϕ -ysinϕ	Y'	y'
c2	next x	$\cos\phi$	next Ysinϕ	settling	x'	Y'
c3	next X	$\sin\phi$	next Ycosϕ	next ycosϕ +Ysinϕ	X'	x'
c0	next y	$\cos\phi$	next ysinϕ	settling	next y'	X'

蝶形运算单元单个运算需要 5 个时钟周期，但并行流水线操作时，每四个周期产生一个结果。表 5-2 说明了在不同的时钟周期内乘法器、加法器和 ROM、RAM 的输入、输出情况。

5.2.2 地址产生单元(AGU, Address Generation Unit)

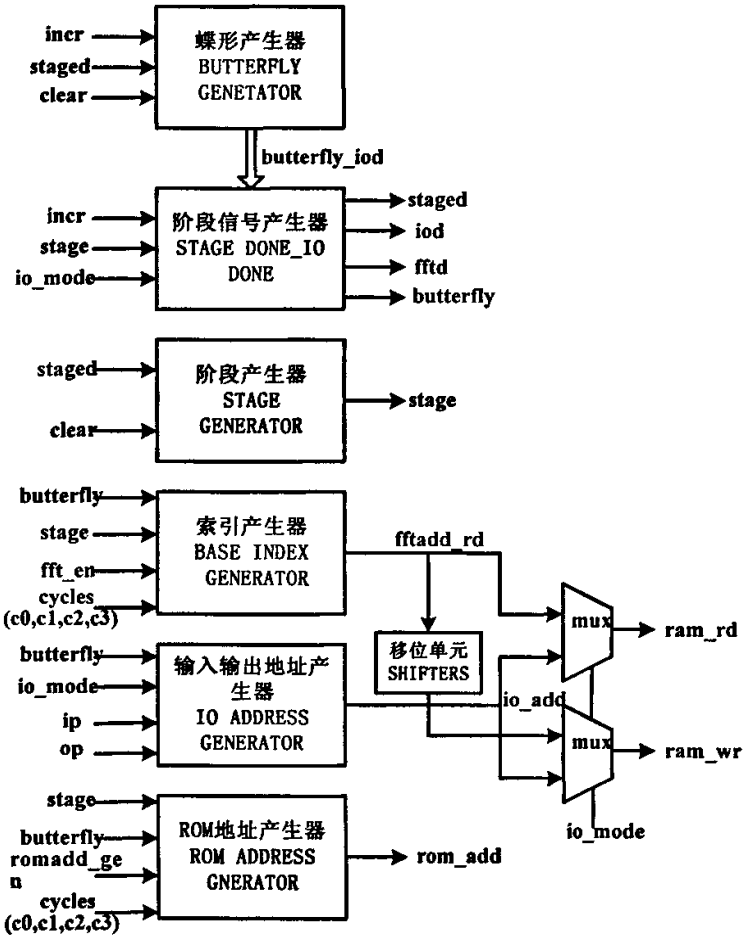


图 5-8 地址产生单元(AGU)的结构图

Fig5-8 Block diagraph of address generation unit

地址产生单元产生供 RAM 和系数 ROM 使用的正确地址，同时跟踪蝶形运算单元所处的阶段。对 8 点复数 FFT 来说，共有 3 个阶段，每个阶段包含 4 个蝶形运算。除此之外由于产生的地址在输入、输出和 FFT 计算过程中是不一样的，所以地址产生单元需要跟踪系统的运行模式从而产生相关的地址。系统运行模式的

信息由控制器提供。AGU 部件的结构图如图 5-8 所示。

下面分别介绍地址产生单元中各个模块的设计及功能。

1)、蝶形产生器(Butterfly Generator)

蝶形产生器的主要作用是跟踪在某一个阶段正在进行第几个蝶形运算。对于 8 点的复数 FFT 而言，每一个阶段有 4 个蝶形运算，每个蝶形运算有 4 个数据字节(两个实数、两个虚数)，因此采用 16 位的记数器来实现。

这里需要说明的是在数据输入、输出阶段中蝶形产生器单元是通过主时钟进行记数的，而在 FFT 计算阶段中，蝶形产生器单元通过 c0 来计数的。由于一个蝶形运算需要 4 个时钟周期，因此在 FFT 运算阶段，蝶形产生器每 4 个时钟周期计数一次。主时钟和 c0 之间的选择通过多路复用器(mux)来实现，信号“io_mode”实现控制选择。当信号“clear”或者“stage done”置为高电平时，蝶形产生器单元复位。下面分别给出了蝶形产生器综合后的的顶层封装模块、根据综合结果提炼出来的原理图以及行为级仿真波形。

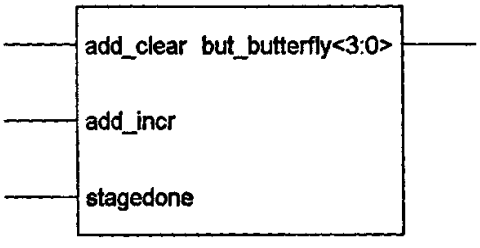


图 5-9 顶层 Butterfly Generator 的模块符号

Fig5-9 Module symbol of Butterfly Generator

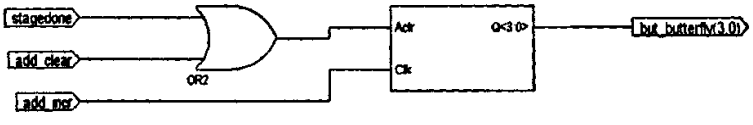


图 5-10 综合后 Butterfly Generator 的原理图

Fig5-10 Schematic of Butterfly Generator after synthesizing

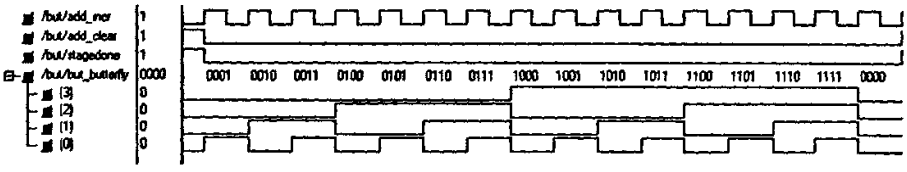


图 5-11 蝶形产生器的行为级仿真

Fig5-11 Behavioral simulation of Butterfly Generator

2)、阶段信号产生器(Stage Done_io Done)

该模块主要产生四个信号，“iod”，“staged”，“fftd”，和“butterfly”。当信号“butterfly”读数为 15 时，信号“iod”产生。此信号告知控制器数据输入或数据输出结束。当信号“butterfly”为 4 时，“staged”信号产生，使阶段产生器(Stage Generator)加 1。当阶段信号“stage”为 3 时，“fftd”信号产生，此信号告知控制器 FFT 计算阶段已经完成，可以开始输出数据阶段。下面分别给出了阶段信号产生器综合后的的顶层封装模块、根据综合结果提炼出来的原理图以及行为级仿真波形。

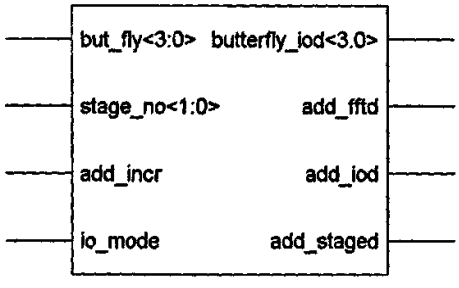


图 5-12 顶层的阶段信号产生器模块符号

Fig5-12 Module symbol of Stage Done_io Done

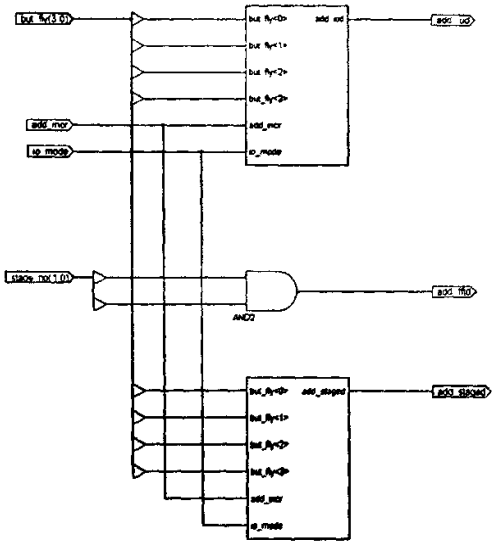


图 5-13 综合后的阶段信号产生器原理图

Fig5-13 Schematic of Stage Done_io Done after synthesizing

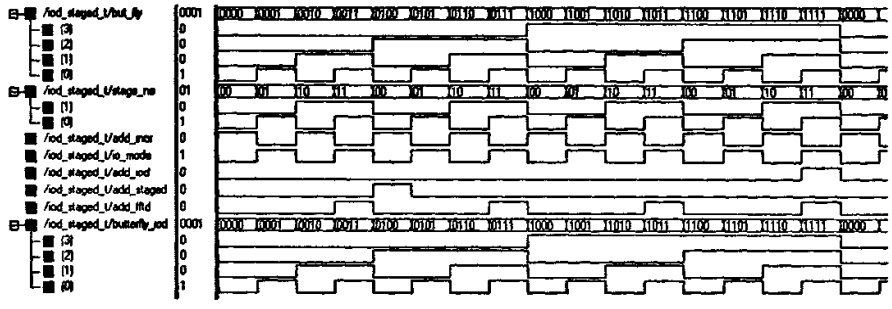


图 5-14 阶段信号产生器的行为级仿真

Fig5-14 Behavioral simulation of Stage Done_io Done

3)、阶段产生器(Stage Generator)

阶段产生器用来跟踪在 FFT 计算中所处的阶段，从而产生当前阶段所处的编号，并将这些编号提供给索引产生器(Base Index Generator)。8 点的 FFT 共有 3 个阶段，因此需要一个 2 位的计数器在每 4 个蝶形计算后计数一次。下面分别给出了阶段产生器综合后的的顶层封装模块、根据综合结果提炼出来的原理图以及行为级仿真波形。

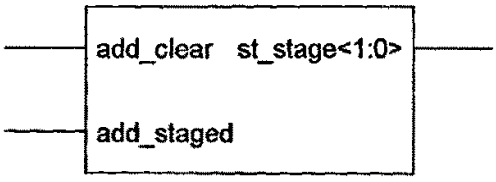


图 5-15 顶层的阶段产生器模块符号

Fig5-15 Module symbol of Stage Generator

图 5-16 综合后的阶段产生器原理图

Fig5-16 Schematic of Stage Generator after synthesizing

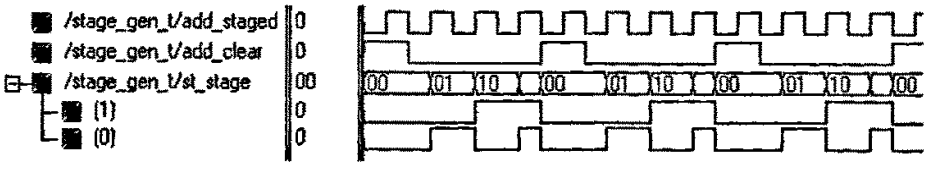


图 5-17 阶段产生器的行为级仿真

Fig5-17 Behavioral simulation of Stage Generator

4)、输入输出地址产生器(IO Address Generator)

在 RAM 数据输入、输出过程中，输入输出地址产生器产生地址。在数据输入阶段信号“butterfly”用来定位 RAM 中的 16 个地址。而在数据输出阶段，数据需要进行位倒序，倒序地址由 AGU 中的多路复用器来实现。控制器通过“io_mode”信号告知是否处于 IO 模式。下面分别给出了输入输出地址产生器综合后的的顶层封装模块、根据综合结果提炼出来的原理图以及行为级仿真波形。

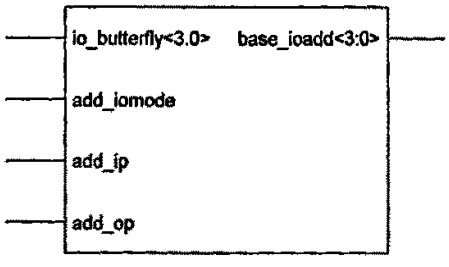


图 5-18 顶层的输入输出地址产生器模块符号

Fig5-18 Module symbol of IO Address Generator

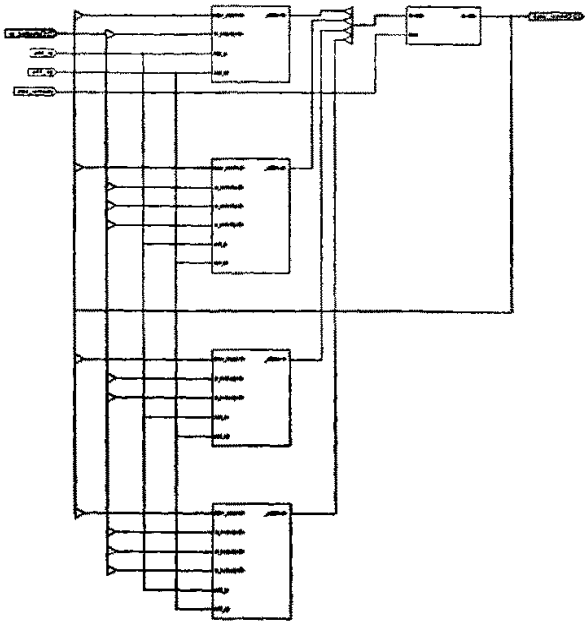


图 5-19 综合后的输入输出地址产生器原理图

Fig5-19 Schematic of IO Address Generator after synthesizing

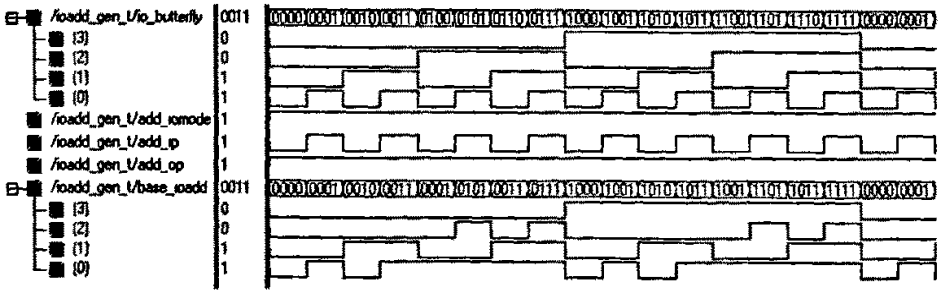


图 5-20 输入输出地址产生器的行为级仿真

Fig5-20 Behavioral simulation of IO Address Generator

5)、索引地址产生器(Base Index Generator)

该单元产生蝶形运算时输入、输出的 RAM 地址。通过控制蝶形单元数和阶段数及时钟循环来控制地址，如表 5-3 所示：

表 5-3 蝶形运算时数据输入地址产生规则

Tab5-3 Address of input data in FFT

butterfly	cycles	stage	value read	address
00	c0	00	y	0100
00	c1	00	Y	1100
00	c2	00	x	0000
00	c3	00	X	1000
01	c0	00	y	0101
01	c1	00	Y	1101
...	...	...	...	...
11	c0	00	y	0111
11	c1	00	Y	1111
11	c2	00	x	0011
11	c3	00	X	1011
00	c0	01	y	0010
01	c1	01	Y	1010
...	...	...	...	...
11	c0	01	y	0111
11	c1	01	Y	1111
...	...	...	...	...
00	c0	10	y	0001
...	...	...	...	...
11	c3	10	X	1110

蝶形运算需要两个复数的输入数据 A 和 B，这两个数据经过处理以后生成四个输出的数据 x、X、y 和 Y，其中 X 和 Y 也为复数。地址的产生是对蝶形产生器(Butterfly Generator)，阶段产生器(Stage Generator)的输出以及信号 c0,c1,c2,c3 进行处理实现的。如果把 4 位的信号“butterfly”设为“b3,b2,b1,b0”，则 x、X、y 和 Y 的产生规律如表 5-4 所示。其中 X、Y 是将 x、y 的最高位置为 1。以上这些描述都会在 VHDL 的编程中有所体现。

表 5-4 x、X、y、Y 产生规律表

Tab5-4 Generation regulation of x、X、y、Y

Stage	Address for x	Address for X	Address for y	Address for Y
00	0 0 b1 b0	1 0 b1 b0	0 1 b1 b0	1 1 b1 b0
01	0 b1 0 b0	1 b1 0 b0	0 b1 1 b0	1 b1 1 b0
10	0 b1 b0 0	1 b1 b0 0	0 b1 b0 1	1 b1 b0 1

下面分别给出了索引地址产生器综合后的的顶层封装模块、根据综合结果提炼出来的原理图以及行为级仿真波形。

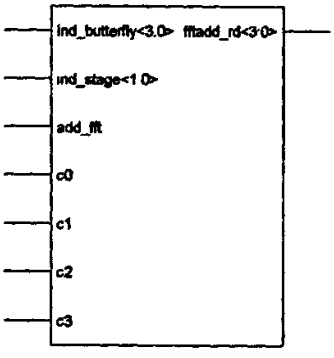


图 5-21 顶层的索引地址产生器模块符号

Fig5-21 Module symbol of Base Index Generator

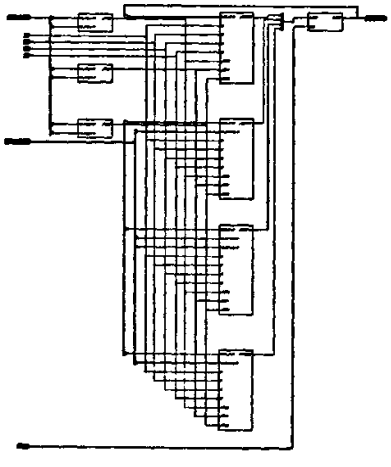


图 5-22 综合后的索引地址产生器原理图

Fig5-22 Schematic of Base Index Generator after synthesizing

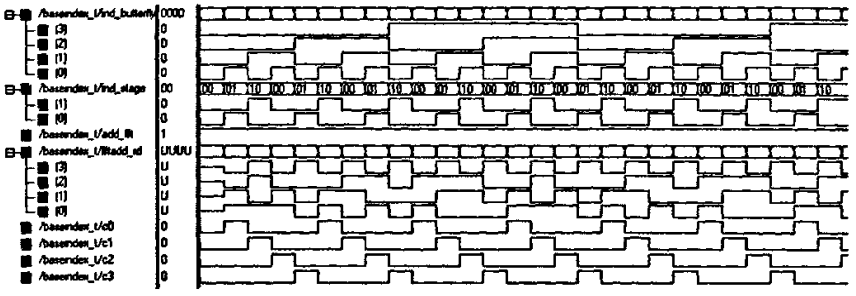


图 5-23 索引地址产生器的行为级仿真

Fig5-23 Behavioral simulation of Base Index Generator

6)、移位单元(Shifters)

FFT 运算的输出要写到输入时相同的 RAM 位置。在计算过程中有 5 个时钟的延迟。例如：如果“y”在“c0”为高电平时被读入，则在 5 个时钟之后“y'”在“c1”高电平时被写入同样的地址作为输入。因此输入地址在每 5 个时钟从移位单元移出，最后一个移位器的输出给出写地址。下面分别给出了移位单元综合后的的顶层封装模块以及行为级仿真波形。

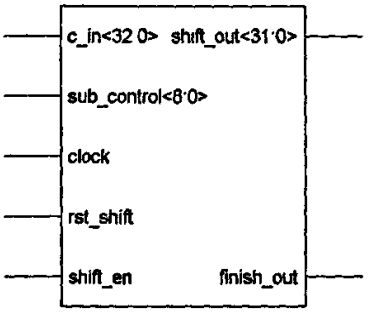


图 5-24 顶层的移位单元模块符号

Fig5-24 Module symbol of Shifters

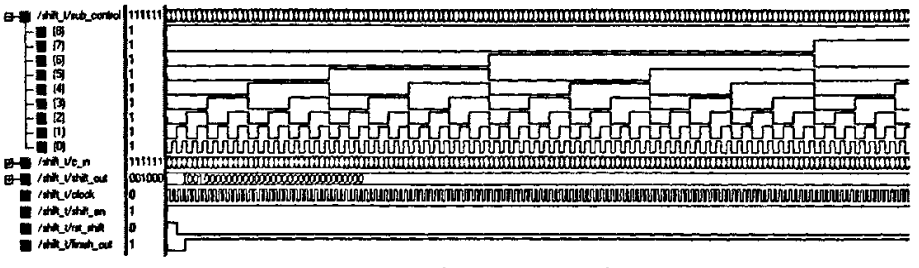


图 5-25 移位单元的行为级仿真

Fig5-25 Behavioral simulation of Shifters

7)、ROM 地址产生器(ROM Address Generator)

ROM 地址产生器是为在 ROM 中存放的蝶形运算系数提供正确地址，而蝶形运算系数的值根据给定的数学模型产生。地址的选取根据下面表 5-5 的规则：

表 5-5 ROM 地址产生规则

Tab5-5 Generation regulation of ROM address

butterfly	stage	coefficient
0000	00	W_8^0
0001	00	W_8^0
…	…	…
0000	01	W_8^0
0001	01	W_8^2
0010	01	W_8^0
…	…	…
0000	10	W_8^0
0001	10	W_8^1
0010	10	W_8^2
0011	10	W_8^3
0100	10	W_8^0
…	…	…

下面分别给出了移位单元综合后的的顶层封装模块以及行为级仿真波形。

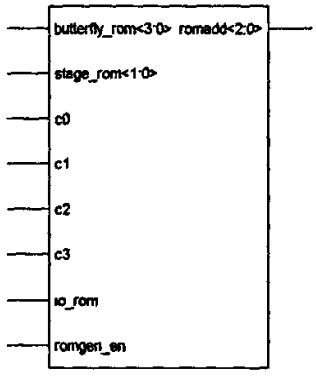


图 5-26 顶层的 ROM 地址产生器模块符号

Fig5-26 Module symbol of ROM Address Generator

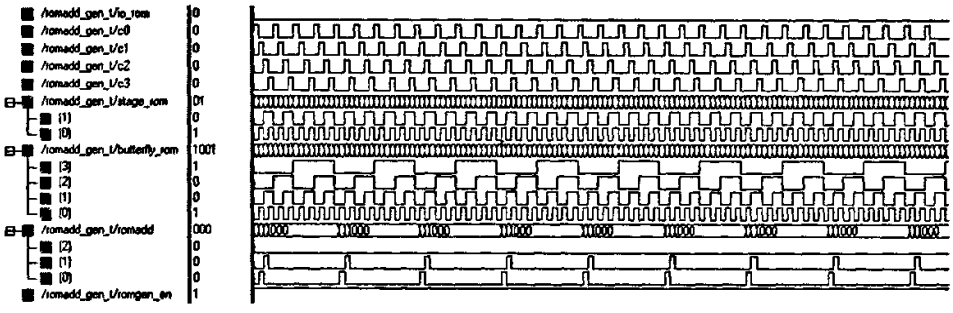


图 5-27 ROM 地址产生器的行为级仿真

Fig5-27 Behavioral simulation of ROM Address Generator

5.2.3 控制器模块(Controller)

控制器采用有限状态机方式建模，从 rst1 到 rst7。下面分别给出了控制器综合后的的顶层封装模块以及行为级仿真波形。

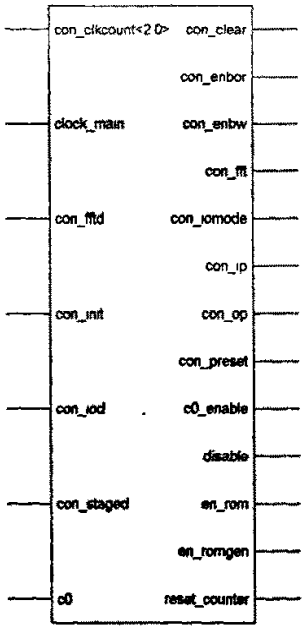


图 5-28 顶层的控制器模块符号

Fig5-28 Module symbol of Controller

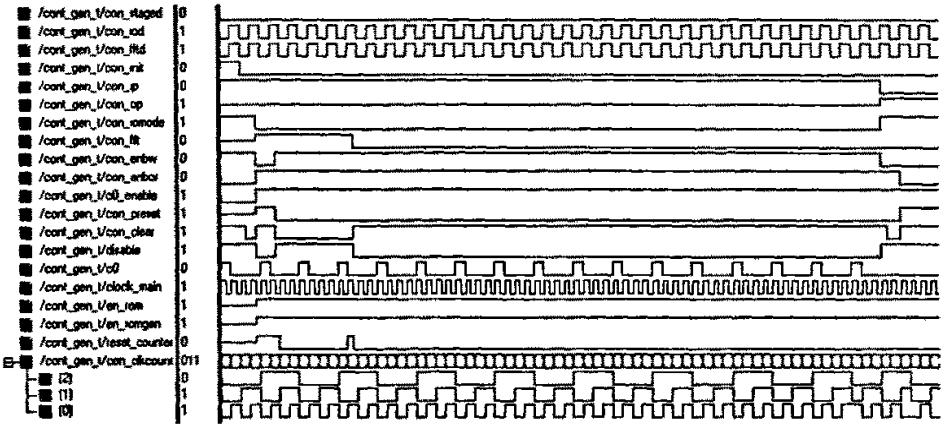


图 5- 29 控制器的行为级仿真

Fig5-29 Behavioral simulation of Controller

5.2.4 RAM 和 ROM

数据首先输入到 RAM。在 FFT 计算过程中，FFT 的两个数据进行计算和写回到 RAM 的相同位置，在输出过程中，根据位倒序地址给出相应的输出值。ROM 用来存储 FFT 计算过程中需要的正弦和余弦函数，根据给定的地址输出相应的值。

下面首先分别给出了 RAM 综合后的的顶层封装模块以及行为级仿真波形。

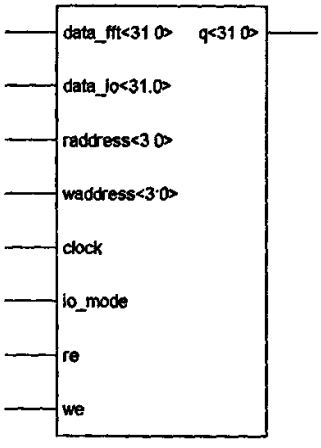


图 5-30 顶层的 RAM 模块符号

Fig5-30 Module symbol of RAM

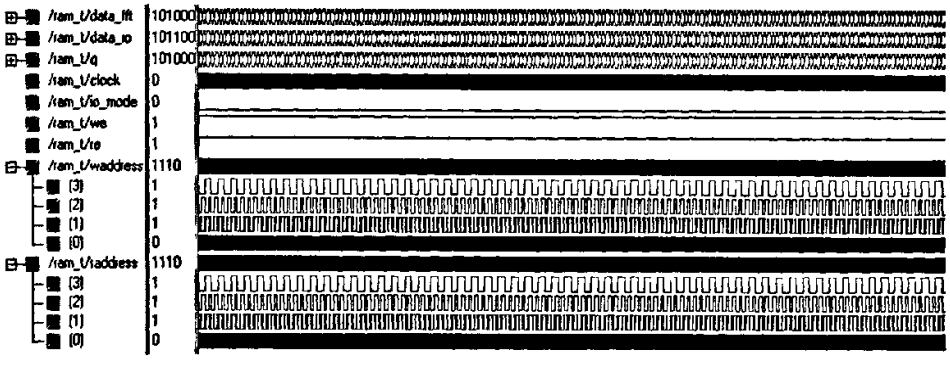


图 5-31 RAM 的行为级仿真

Fig5-31 Behavioral simulation of RAM

接着分别给出了 ROM 综合后的的顶层封装模块以及行为级仿真波形。

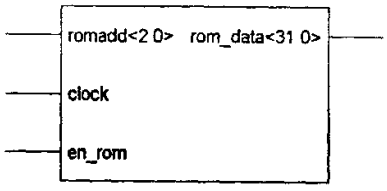


图 5-32 顶层的 ROM 模块符号

Fig5-32 Module symbol of ROM

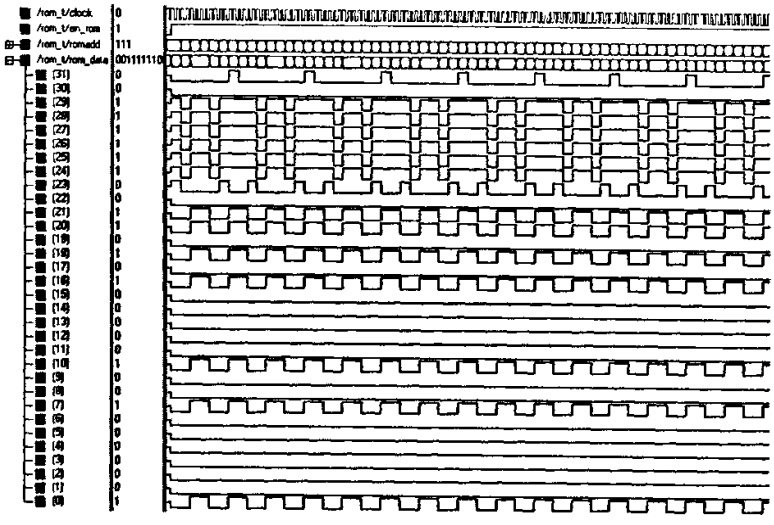


图 5-33 ROM 的行为级仿真

Fig5-33 Behavioral simulation of ROM

5.2.5 FFT 整体模块仿真结果

本模块采用XILINX公司的Virtex2系列芯片进行设计验证，工程属性如图5-34所示。软件开发平台为ISE6.1i[45][46]，综合工具为ISE中集成的XST，仿真工具为ModelSim XE II 5.7c。

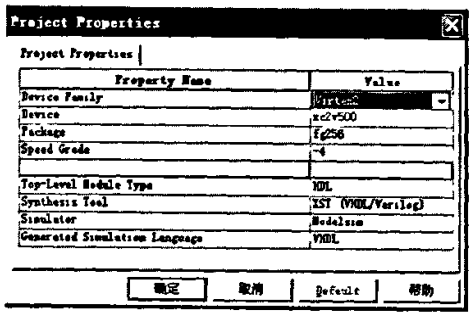


图 5-34 工程属性

Fig5-34 Project properties

下面给出了 FFT 模块的综合后的的顶层封装模块以及行为级仿真波形。

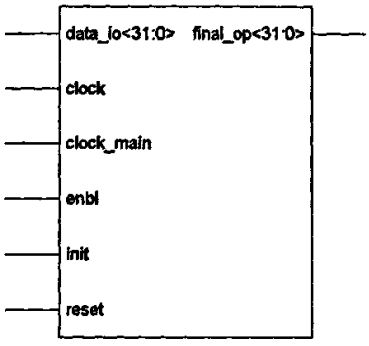


图 5-35 顶层的 FFT 模块符号

Fig5-35 Module symbol of FFT

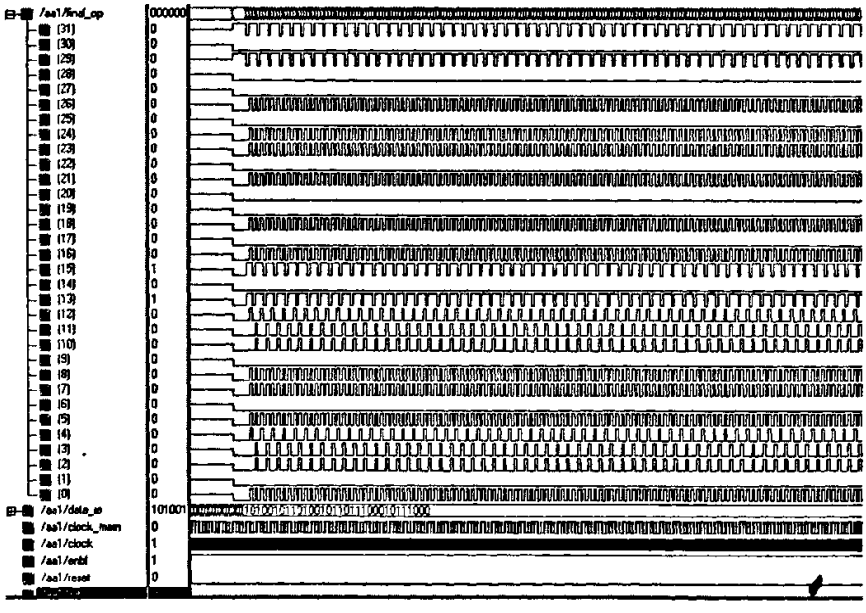


图 5-36 FFT 的行为级仿真

Fig5-36 Behavioral simulation of FFT

为验证设计的正确性，现给出一组输入数据为[-1 1 2 -0.5 -3 -1 2 0]，虚部都为 0，经过 C 程序转换为下列形式作为 FFT 模块的输入。

10111111100000000000000000000000

00111111100000000000000000000000

01000000000000000000000000000000

10111111000000000000000000000000

11000000010000000000000000000000

10111111100000000000000000000000

01000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

通过 VHDL 编程，把仿真的输出部分写入一个名为“result.txt”的文本文件中，具体结果如下

10111111000000000000000000000000

01000000011100010010001011010000

11000001000000000000000000000000

00111110011011011101001100000000

00111111000000000000000000000000

00111110011011011101001100000000

11000001000000000000000000000000

01000000011100010010001011010000

00000000000000000000000000000000

10111111100001111100001101100000

10111111000000000000000000000000

10111111100001111100001101100000

00000000000000000000000000000000

00111111100001111100001101100000

00111111000000000000000000000000

00111111100001111100001101100000

经过 C 程序的转换得到以下数据：

实部为[-0.5 3.76775 -8 0.23225 0.5 0.23225]

虚部为[0 -1.06065 -0.5 -1.06065 0 1.06065 0.5 1.06065]

运用 MATLAB 计算结果如下：

```
p=[-1 1 2 -0.5 -3 -1 2 0]
y=fft(p)
disp(y)
y =
  Columns 1 through 4
  -0.5000    3.7678 - 1.0607i   -8.0000 - 0.5000i    0.2322 - 1.0607i
  Columns 5 through 8
   0.5000    0.2322 + 1.0607i   -8.0000 + 0.5000i    3.7678 + 1.0607i
```

以上可以看到硬件系统进行运算得出结果和 MALAB6.5 软件计算结果误差在可容忍范围之内。采用时间抽取基二算法实现 FFT 变换简单、实用，设计出的 FFT 部件符合设计要求，同时采用简单的硬件结构，占用硬资源较少，使在一个 FPGA 芯片上实现整个系统成为可能。

5.3 本章小结

本章首先对 FFT 算法做了详细的分析，其后重点设计实现了 8 点 32 位 FFT 处理器，关于总体结构这一点，虽然有一些文献可以参考，但由于种种原因，对于具体怎么实现，各类文献介绍的较少。作者在参考前人的工作上，设计了该 FFT 处理器的总体结构并且分别对几个主要模块进行了仿真，最后通过比较可以看出整体仿真结果与理论值吻合的很好。

随着芯片集成度的不断提高，用这种结构实现的 FFT 运算其优越性将越来越明显，而且用这种结构实现的 FFT 很容易扩展，只需要增加蝶形的个数和循环次

数即可。基于 FPGA 的 FFT/IFFT 处理器由于其硬件上的并行性，速度远远快于一般的通用 DSP。FPGA 具有成千上万的查找表和触发器，因此，FPGA 平台可以利用更低的成本达到比通用 DSP 更快的速度。采用 FPGA 技术，还可以获得高性能，满足成本要求，并享有快速有效地对新设计进行优化的灵活性。

第六章 全文总结

6.1 总结

本文是通过对无线局域网技术、OFDM 技术、可编程器件设计三大技术的研究，得到了一些有用的结论。

现将本文的工作概括总结如下：

首先，在查阅、分析有关技术资料的基础上，对无线网络技术、OFDM 调制解调技术、可编程逻辑器件应用技术进行了研究，重点介绍了 OFDM 技术在无线局域网中的应用的基本原理和 OFDM 系统的设计思想。该方法可以有效地克服多径干扰和码间干扰对数据传输的影响，从而大大降低了误码率。数据帧之间插入保护间隔，可以较好克服符号间串扰(ISI)。

接着，从物理层的角度分别对无线局域网的三个标准 IEEE802.11b，IEEE802.11a，IEEE802.11g 进行了详细的介绍，以表格的形式给出了三个标准之间的联系和异同，为本课题研究提供一定的理论参考。

然后采用仿真软件 SystemView 对 OFDM 的传输性能进行了系统仿真。为便于硬件实现可能，利用 M-Link 编程搭建并实现 OFDM 模块功能。通过仿真比较了两种信道编码的性能，并做了相应的误码率分析，得到了有用的结论，建议采用卷积编码。然后通过 MATLAB 软件对 OFDM 系统进行了仿真和分析，研究了不同调制映射，保护间隔，载波数和信道对 OFDM 系统的影响，得出一定的结论。

本文最后给出了 FFT 总体实现框图，使用 VHDL 语言编写了基于 FPGA 的无线局域网系统的核心 FFT IP 模块，实现了 OFDM 系统的核心部分 FFT 处理器，并对本文所设计的 FFT 处理器进行仿真结果与理论计算的结果相比较，结果相符，证实了该设计的正确性和可行性。

6.2 不足与展望

由于受设备和实验条件的限制，本论文只是对无线局域网的 OFDM 系统以及 FFT 处理器的 FPGA 硬件设计等进行了计算机仿真，未能做出成品，但可以确信设计是有一定的实用参考价值和理论意义的。就本课题来说，还有很多需要进一步改善的地方，如对一些重要参数进行设置调整，从而进行最终的硬件调试，以达到实际应用的目的。通过对可编程逻辑器件及 IP 核技术和 SOC(片上系统)技术做了有益的探索，为组成完整的硬件系统打下了坚实的基础。

由于目前我国无线局域网技术比国外还有一定的差距，难以满足用户的需求，网络的管理目前尚未完善，网络的安全问题仍倍受质疑，这些都为无线局域网的开发商提出了新的课题，如何更好的解决这些矛盾将是我们值得关注的问题。因为 IEEE802.11a 标准和 IEEE802.11b/g 标准的不兼容，导致 IEEE802.11a 无线局域网在市场推广上面临困境，如何在一个设备中同时实现这三个不同的协议标准就成了研究的重点。一些国外公司已开发出了同时支持这三种协议的基带芯片，下阶段问题将集中在射频部分电路的融合上。因此可以采用软件无线电技术，通过软件修改传输参数来实现 IEEE802.11a 和 IEEE802.11b/g 两个系统的切换。

参考文献

[1] 彭林等，第三代移动通信技术，北京，电子工业出版社，2003，p331-335

[2] 王伟，无线局域网技术现状及发展趋势，2004，http://www.cnii.com.cn/2004021

[3] IEEE Std.802.11b, "Wireless LAN Medium Access Control (MAC) and Physical Layer(PHY)specifications, High-speed Physical Layer Extension in the 2.4GHz Band", September, 1999

[4] IEEE Std.802.11a, "Wireless LAN Medium Access Control (MAC) and Physical Layer(PHY)specifications, High-speed Physical Layer Extension in the 5GHz Band", December, 1999

[5] IEEE Std.802.11g, "Wireless LAN Medium Access Control (MAC) and Physical Layer(PHY)specifications, High-speed Physical Layer Extension in the 2.4GHz Band", June, 2003

[6] 宋铁成，下一代移动通信系统中的 OFDM 技术，移动通信，2001，11 期，p15-18

[7] D.Castelain, B.LeFloch, and R. Halbert-Lassalle, Digital sound broadcasting to mobile receivers , IEEE Trans. On Consumer Electron., 1989, Vol.73, p30-34

[8] Ulrich Reimers. Digital Video Broadcasting. IEEE Communications Magazine. June 1998, p104-110

[9] J.S. Chow, L.C. Tu and J.M. Cioffi, "Performance evaluation of a multichannel transceiver system for ADSL and VHDSL services", IEEE J. Selected Area Commune., Aug.1991, Vol.SAC-9, No.6, p909-919

[10] 梅辉，罗文茂，第四代移动通信中的 OFDM 技术，电信快报，2003，8 期，p28-30

[11] 金旗，裴昌幸，张振生，OFDM 技术的基本原理，通信技术，2004，6 期，p44-46

[12] Bingham J AC. Muticarrier modulation for Data Transmission: An Idea Whose

time has Come, IEEE Commune, May 1999, Vol.28, p5-14

[13] Kalet, The multitone channel, IEEE Trans. Commune, 1999, Vol 37, No 2, p119-124

[14] Andreas F. Molisch 编著，许希斌，赵明等译，宽带无线数字通信，北京，电子工业出版社，2002，p231-283

[15] 萧宝瑾，信息论与编码，兵器工业出版社，2000，p25-28

[16] Jean Armstrong, Analysis of New and Existing Methods of Reducing Intermarries Interference Due to Carrier Frequency Offset in OFDM, IEEE Trans. Comm., March 1999, p365-369

[17] 李建东，杨家玮，个人通信，人民邮电出版社，1998，5 期，p217-229

[18] 樊昌信，张甫翊，徐炳祥等，通信原理(第五版)，国防工业出版社，2001，p83-85

[19] 宋祖顺等，现代通信原理，北京，电子工业出版社，

[20] Yiyan Wu, William Y Zou, Orthogonal frequency division multiplexing, a multi-carrier modulation asheme, IEEE Transaction on Consumer Electronics, 1999.4, Vol.41, No.3, p391-399

[21] 傅延增，张海林，王育民，正交频分复用中的符号同步技术，西安电子科技大学学报，2000，3 期，p335-339

[22] 任立刚等，MIMO+OFDM：新一代移动通信核心技术，中国数据通信，2003，10 期，p102-105

[23] 钟章队等，无线局域网，科学出版社，2004，p3-5

[24] Juha Heiskala，John Terry 著，杨晓春等译，OFDM 无线局域网，电子工业出版社，2003，p146-178

[25] Bob Pearson, Complementary Code Keying Made Simple httn://www.intersil.com/data/an/an9/an9R50

[26] 杨震，崔丙锋，王亚丹，一种改进的 CCK 调制解调方案与 FPGA 实现，北京邮电大学学报，2004，2 期，p33-37

[27] 俞鹤伟，赖声礼，基于 OFDM 的 IEEE802.11a 标准的同步方法，电视技术，2004，2 期，p50-55

[28] 韩旭东，张春业，曹建海，IEEE802.11g 研究综述，世界电信，2004，1 期，p47-50

[29] 石雷，熊建设，徐洪梅，OFDM 物理层帧结构的仿真研究，信息安全与通信保密，2005，5 期，p42-44

[30] 韩旭东，张春业，曹建海，IEEE802.11 无线局域网标准的网络性能比较与分析，今日电子，2004，2 期，p41-44

[31] 王征，江汉红，双频多模无线局域网技术分析，武汉理工大学学报，2005，2 期，p61-64

[32] 雷震洲，无线局域网标准802.11b与802.11a物理层比较，现代电信科技，2002，11期，p35-39

[33] 青松，程岱松，武建华，数字通信系统的 System View 仿真与分析，北京航空航天大学出版社，2001

[34] 李东生，雍爱霞，左洪浩等，System View 系统设计及仿真入门与应用，电子工业出版社，2002，p136-196

[35] 翁剑枫，叶志前，MATLAB LabVIEW SystemView仿真分析基础，机械工业出版社，2005

[36] A.V.奥本海姆，R.W.谢弗著，黄建国，刘树棠译，离散时间信号处理，科学技术出版社，1998，p480-530

[37] Hirosaki B., An orthogonally multiplexed QAM system using the discrete Fourier transform , IEEE Trans. Commune，July,1991, p384-389

[38] 路而红，专用集成电路设计与电子设计自动化，北京，清华大学出版社，2004，p71-138

[39] 谭会生，张昌凡，EDA 技术及应用，西安电子科技大学出版社，2002，p16-75

[40] 求是科技，CPLD/FPGA 应用开发技术与工程实践，北京，人民邮电出版社，2005，p23-75

[41] 曾繁泰，陈美金，VHDL 程序设计（第二版），北京，清华大学出版社，2001，p313-320

[42] S.Bertazzoni, G. C. Cardanilli, 16-point High Speed (I)FFT for OFDM Modulation,

IEEE Transactions on Computers, March,1998, p210-213

[43] M.Huang, R.Kwok, S.P.Chan. Simplified and accurate power-analysis method for deep-submicron ASIC designs. IEEE Proceedings Circuits,Devices and Systems. 2000, 147, p175-182

[44] Yun-Nan Chang, Keshab, Parhi, An Efficient Pipelined FFT Architecture, IEEE Trans. on Circuits and Systems, 2003, 50(6), p322-325

[45] 潘松，王国栋，VHDL 实用教程，成都，电子科技大学出版社，2000

[46] EDA 先锋工作室，FPGA/CPLD 设计工具 ISE 使用详解，北京，人民邮电出版社，2005

附 录 1 VHDL 源程序

1．蝶形产生器(Butterfly Generator)

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use work.butter_lib.all ;
use ieee.std_logic_unsigned.all ;

entity but_gen is
port (
        add_incr , add_clear , stagedone : in std_logic ;
        but_butterfly : out std_logic_vector(3 downto 0) ) ;
end but_gen ;

architecture rtl of but_gen is
begin
process(add_clear , add_incr , stagedone)
variable cnt : integer ;
variable count : std_logic_vector(3 downto 0) ;
begin
if(add_clear = '1' or stagedone = '1') then
count := "0000" ;
but_butterfly <= "0000" ;
elsif (add_incr'event and add_incr = '1') then
but_butterfly <= (count + 1) ;
count := count + 1 ;
end if ;
end process ;
end rtl ;
```

2．阶段信号产生器(Stage Done_io Done)

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use work.butter_lib.all ;
use ieee.std_logic_unsigned.all ;
```

```
entity iod_staged is
port (
        but_fly : in std_logic_vector(3 downto 0) ;
        stage_no : in std_logic_vector(1 downto 0) ;
        add_incr , io_mode    : in std_logic ;
        add_iod , add_staged , add_fftd : out std_logic ;
        butterfly_iod : out std_logic_vector(3 downto 0) ) ;
end iod_staged ;
architecture rtl of iod_staged is
begin
process(but_fly,add_incr,io_mode)
begin
if(but_fly = 15 and io_mode = '1' and add_incr='0') then
add_iod <= '1' ; -- io done signal
butterfly_iod <= but_fly ;
add_staged<='0';
elsif(but_fly = 4 and io_mode = '0' and add_incr='1') then
butterfly_iod <=but_fly ;
add_iod <= '0' ;
add_staged <= '1' ; -- stage done signal
else
butterfly_iod <= but_fly ;
add_staged <= '0' ;
add_iod <= '0' ;
end if ;
end process ;

process(stage_no)
begin
if (stage_no=3) then
add_fftd <= '1' ; -- fft done signal
else     add_fftd <= '0' ;
end if ;
end process ;
end rtl;
```

3．阶段产生器(Stage Generator)

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use work.butter_lib.all ;
```

```
use ieee.std_logic_unsigned.all ;

entity stage_gen is
port (
        add_staged , add_clear : in std_logic ;
        st_stage : out std_logic_vector(1 downto 0) ) ;
end stage_gen ;

architecture rtl of stage_gen is

begin
process(add_staged , add_clear)
variable s_count : std_logic_vector(1 downto 0) ;
begin
if (add_clear = '1') then
st_stage <= "00" ;
s_count := "00" ;
elsif(add_staged'event and add_staged= '1' ) then
st_stage <= s_count + 1 ;
s_count := s_count + 1 ;
end if ;
end process ;
end rtl ;
```

4．输入输出地址产生器(IO Address Generator)

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use work.butter_lib.all ;
use ieee.std_logic_unsigned.all ;

entity ioadd_gen is
port (
        io_butterfly : in std_logic_vector(3 downto 0) ;
        add_iomode , add_ip , add_op : in std_logic ;
        base_ioadd : out std_logic_vector(3 downto 0) ) ;
end ioadd_gen ;

architecture rtl of ioadd_gen is
begin
process(io_butterfly , add_iomode , add_ip , add_op)
```

```
variable out_data : std_logic_vector(3 downto 0) ;
begin
if(add_iomode = '1') then
 if (add_ip = '1') then
  out_data := io_butterfly(3 downto 0) ;
 elsif(add_op = '1') then
  if(io_butterfly(3) = '0') then -- ie, real part
  out_data := '0' & io_butterfly(0) & io_butterfly(1) & io_butterfly(2) ;
  elsif(io_butterfly(3)='1') then -- ie, complex part
  out_data := '1' & io_butterfly(0) & io_butterfly(1) & io_butterfly(2) ;
  end if ;
 end if ;
end if ;
base_ioadd <= out_data(3 downto 0) ;
end process ;
end rtl ;
```

5. 索引地址产生器(Base Index Generator)

```
library ieee;
use ieee.std_logic_1164.all;
use work.butter_lib.all ;

entity baseindex is
 port(
    ind_butterfly: in std_logic_vector(3 downto 0);
    ind_stage: in std_logic_vector(1 downto 0);
    add_fft: in std_logic;
    fftadd_rd: out std_logic_vector(3 downto 0);
    c0,c1,c2,c3: in std_logic);
end baseindex;

architecture rtl of baseindex is
begin
process(ind_butterfly,ind_stage,add_fft,c0,c1,c2,c3)
variable out_sig : std_logic_vector(3 downto 0);
begin
if (add_fft='1') then
if(c2='1') then -- address for 'x'. Since this is the real part,
case ind_stage is -- M.S.B is '0'.
    when "00" => out_sig := "00" & ind_butterfly(1 downto 0);
    when "01" => out_sig := '0' & ind_butterfly(1) & '0' & ind_butterfly(0);
```

```
-- when "10" => out_sig := '0' & '1' & '1' & ind_butterfly(3);
    when "10" => out_sig := '0' & ind_butterfly(1 downto 0) & '0';
    when others => out_sig := "0000";
end case;

elsif(c0='1') then -- address for 'y'.
case ind_stage is
  when "00" => out_sig := "01" & ind_butterfly(1 downto 0);
  when "01" => out_sig := '0' & ind_butterfly(1) & '1' & ind_butterfly(0);
  when "10" => out_sig := '0' & ind_butterfly(1 downto 0) & '1';
  when others => out_sig := "0000";
end case;

elsif(c1='1') then -- addresss for 'Y'
case ind_stage is
  when "00" => out_sig := "11" & ind_butterfly(1 downto 0);
  when "01" => out_sig := '1' & ind_butterfly(1) & '1' & ind_butterfly(0);
  when "10" => out_sig := '1' & ind_butterfly(1 downto 0) & '1';
  when others => out_sig := "0000";
end case;

elsif(c3='1') then -- address for 'X'
case ind_stage is
 when "00" => out_sig := "10" & ind_butterfly(1 downto 0);
  when "01" => out_sig := '1' & ind_butterfly(1) & '0' & ind_butterfly(0);
  when "10" => out_sig := '1' & ind_butterfly(1 downto 0) & '0';
  when others => out_sig := "0000";
--else
--out_sig := "ZZZZ";
end case;
end if;
end if;
fftadd_rd <= out_sig (3 downto 0) ;
end process;
end rtl;
```

6．移位单元(Shifters)

```
library ieee ;
use ieee.std_logic_1164.all ;
use work.butter_lib.all ;
use ieee.std_logic_arith.all ;
```

```
use ieee.std_logic_unsigned.all ;

entity shift2 is
port (
        sub_control : in std_logic_vector (8 downto 0) ;
        c_in    : in std_logic_vector (32 downto 0) ;
        shift_out : out std_logic_vector (31 downto 0) ;
        clock , shift_en , rst_shift : in std_logic ;
        finish_out : out std_logic ) ;
end shift2 ;
architecture rtl of shift2 is
begin
process(clock)
variable sub_temp : std_logic_vector(7 downto 0) ;
variable   temp2 , temp4 : std_logic_vector(31 downto 0) ;
variable temp3 , t : std_logic ;
begin
if(rst_shift='0') then
if(shift_en = '1') then
if(temp3 = '1') then
if(sub_control(8) = '1') then
sub_temp := sub_control (7 downto 0) ;
temp2 := '1' & c_in (31 downto 1) ; --'1' for implicit one
temp3 := '0' ;
end if ;
end if ;
end if ;
end if ;

if(rst_shift='0') then
if(shift_en = '1') then
if(t = '1') then
if (sub_control(8) = '1') then
if (conv_integer(sub_temp(7 downto 0)) = 0) then
shift_out <= temp2 ;
finish_out <= '1' ;
t := '0' ;
elsif ( clock = '1') then
temp2 := '0' & temp2 (31 downto 1) ;
sub_temp := sub_temp - "00000001" ;
end if ;
```

```
end if;
end if;
end if;
elsif(rst_shift='1') then
temp3 := '1';
finish_out <= '0';
t := '1';
end if;

end process;
end rtl;
```

7. ROM 地址产生器(ROM Address Generator)

```
library ieee;
use ieee.std_logic_1164.all;
use work.butter_lib.all;
use ieee.std_logic_unsigned.all;

entity romadd_gen is
port (
        io_rom,c0,c1,c2,c3 : in std_logic;
        stage_rom : in std_logic_vector(1 downto 0);
        butterfly_rom : in std_logic_vector(3 downto 0);
        romadd : out std_logic_vector(2 downto 0);
        romgen_en : in std_logic );
end romadd_gen;

architecture rtl of romadd_gen is
begin
process(io_rom,c0,c1,c2,c3,stage_rom,butterfly_rom)
begin
if(romgen_en = '1') then
if(io_rom = '0') then
  case stage_rom is

  when "00" =>
  if(c0='1' or c2='1') then
  romadd <= "000";
  elsif(c1='1' or c3='1') then
  romadd <= "001";
  end if;
```

```
when "01" =>
if(butterfly_rom=0 or butterfly_rom=1) then
 if(c0='1' or c2='1') then
 romadd <= "000" ;
 elsif(c1='1' or c3='1') then
 romadd <= "001" ;
 end if ;
elsif(butterfly_rom=2 or butterfly_rom=3) then
 if(c0='1' or c2='1') then
 romadd <= "100" ;
 elsif(c1='1' or c3='1') then
 romadd <= "101" ;
 end if ;
end if ;

when "10" =>
 if(butterfly_rom=0) then
  if(c0='1' or c2='1') then
  romadd <= "000" ;
  elsif(c1='1' or c3='1') then
  romadd <= "001" ;
  end if ;
 elsif(butterfly_rom=1) then
  if(c0='1' or c2='1') then
  romadd <= "100" ;
  elsif(c1='1' or c3='1') then
  romadd <= "101" ;
  end if ;
 elsif(butterfly_rom=2) then
  if(c0='1' or c2='1') then
  romadd <= "010" ;
  elsif(c1='1' or c3='1') then
  romadd <= "011" ;
  end if ;
 elsif (butterfly_rom=3) then
  if(c0='1' or c2='1') then
  romadd <= "110" ;
  elsif(c1='1' or c3='1') then
  romadd <= "111" ;
  end if ;
```

```
  end if;

 when others =>
  romadd <= "000";

 end case;
end if;
end if;
end process;
end rtl;
```

8. 控制器模块(Controller)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.butter_lib.all;
use ieee.std_logic_unsigned.all;

entity cont_gen is
port (
        con_staged , con_iod , con_fftd , con_init : in std_logic;
        con_ip , con_op , con_iomode , con_fft : out std_logic;
        con_enbw , con_enbor , c0_enable , con_preset : out std_logic;
        con_clear , disable : out std_logic;
        c0 , clock_main : in std_logic;
        en_rom , en_romgen , reset_counter : out std_logic;
        con_clkcount : in std_logic_vector(2 downto 0) );
end cont_gen;

architecture rtl of cont_gen is
type state is (rst1,rst2,rst3,rst4,rst5,rst6,rst7);
signal current_state , next_state : state;
shared variable counter , temp2 : std_logic_vector(1 downto 0) := "00";
begin
process (current_state ,con_staged , con_iod , con_fftd , con_clkcount , c0)

begin
case current_state is
 when rst1 =>
 con_iomode <= '1'; -- set mode to io.
 con_ip <= '1'; -- input mode
```

```
con_clear <= '1' ; -- clear all blocks
con_enbw <= '1' ; -- enable write to RAM
con_enbor <= '0' ; -- disable read
c0_enable <= '0' ; -- disable cycles unit
disable <= '1' ; -- disable counter
next_state <= rst2 ;

when rst2 =>
 con_clear <= '0' ; -- bring clear signal back to zero
 next_state <=rst3 ;

when rst3 =>
 if(con_iod = '1') then
  con_preset <= '1' ; -- reset cycles
  reset_counter <= '1' ; -- reset counter
  c0_enable <= '1' ; -- enable cycles
  con_iomode <= '0' ; -- set io mode to '0'
  con_fft <= '1' ; -- fft mode
  en_rom <= '1' ; -- enable ROM
  en_romgen <= '1' ; -- enable ROM address generator
  con_clear <= '1' ; -- clear all blocks
  con_enbw <= '0' ; -- disable write to RAM
  con_enbor <= '1' ; -- enable read from ROM
  disable <= '0' ; -- enable   counter unit.
  next_state <= rst4 ;
 else
 next_state <= rst3 ;
 end if ;

when rst4 =>
 con_preset <= '0' ; -- reset for cycles
 reset_counter <= '0' ; -- reset for counter
 con_clear <= '0' ; -- clear all signals
 if (con_clkcount = 5) then -- check whether 4 or not
  con_enbw <= '1' ; -- enable write to ROM
  disable <= '1' ; -- disable counter
  reset_counter <= '1' ; -- reset counter
  next_state <= rst5 ;
 else
  next_state <= rst4 ;
 end if ;
```

```
when rst5 =>

 if (con_fftd = '1') then
 disable <= '0' ; -- enable counter
 reset_counter <= '0' ;
 con_clear <= '1' ; -- clear butterfly generator
 con_fft <= '0' ; -- disable fft address generator
 if (con_clkcount = 4) then
    disable <= '1';
    con_enbw <= '0' ;
    con_iomode <= '1' ;
    con_op <= '1' ;
    con_ip <= '0' ;
    next_state <= rst6 ;
    else
   next_state <= rst5 ;
   end if ;
 else
 next_state <= rst5 ;
end if ;

when rst6 =>
 con_clear <= '0' ;
 next_state <= rst7 ;

when rst7 =>
 if(con_iod = '1') then
  con_clear <= '1' ;
  con_preset <= '1' ;
  con_enbor <= '0';
 else
 next_state <= rst7 ;
 end if ;


when others =>
 next_state <= rst1 ;

end case ;
end process ;
```

```
process(clock_main , con_init)
begin
if(con_init = '1') then
current_state <= rst1 ;
elsif (clock_main'event and clock_main = '0') then
current_state <= next_state ;
end if ;
end process ;
end rtl ;
```

9. RAM 模块

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
use work.butter_lib.all ;
entity reg_dpram is
port (
        data_fft , data_io : in std_logic_vector (31 downto 0);
        q : out std_logic_vector (31 downto 0);
        clock , io_mode : in std_logic;
        we , re : in std_logic;
        waddress: in std_logic_vector (3 downto 0);
        raddress: in std_logic_vector (3 downto 0));
end reg_dpram;
architecture behav of reg_dpram is
type MEM is array (0 to 15) of std_logic_vector(31 downto 0);
signal ramTmp : MEM;

begin

-- Write Functional Section
process (clock,waddress,we)
begin
if (clock='0') then
if (we = '1') then
if (io_mode = '0') then
ramTmp (conv_integer (waddress)) <= data_fft ;
elsif (io_mode = '1') then
ramTmp (conv_integer (waddress)) <= data_io ;
```

```
end if;
end if;
end if;
end process;

-- Read Functional Section
process (clock,raddress,re)
begin
if (clock='1') then
if (re = '1') then
q <= ramTmp(conv_integer (raddress)) ;
end if;
end if;
end process;
end behav;
```

10. ROM 模块

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use work.butter_lib.all ;
use ieee.std_logic_unsigned.all ;

entity rom is
port (
        clock , en_rom : in std_logic ;
        romadd : in std_logic_vector(2 downto 0) ;
        rom_data : out std_logic_vector(31 downto 0) ) ;
end rom ;

architecture rtl of rom is
begin
process(clock,en_rom)
begin
if(en_rom = '1') then
if(clock = '1') then
case romadd is
  when "000" =>
  rom_data <= "00111111100000000000000000000000" ;
  when "001" =>
  rom_data <= "00000000000000000000000000000000" ;
```

```
 when "010" =>
 rom_data <= "00111111001101010000010010000001" ;
 when "011" =>
 rom_data <= "00111111001101010000010010000001" ;
 when "100" =>
 rom_data <= "00000000000000000000000000000000" ;
 when "101" =>
 rom_data <=   "00111111100000000000000000000000" ;
 when "110" =>
 rom_data <= "10111111001101010000010010000001"   ;
 when "111" =>
 rom_data <= "00111111001101010000010010000001" ;
 when others =>
 rom_data <= "01000000000000000000000000000000" ;
end case ;
end if ;
end if ;
end process ;
end rtl ;
```

附 录 2 MATLAB 主要源程序

```
%参数的初始化设置
t0=9.215
ts=0.001
t=[0:ts:t0];
N=length(t);
d=1;
nqam=N/4;
SubchaNum=16;
qammode=16;
rownum=N/SubchaNum;
fc=10^6;
c=cos(2*pi*fc.*t);                    %qam 调制载波
c_t=sin(2*pi*fc.*t);
fs=1/ts;
df=0.2;
snr_in_db=17              %信噪比
snr=10^(snr_in_db/10);
dt=0.001                          %空间延时
df1=fs/N;
f=[0:df1:df1*(length(t)-1)]-fs/2;

[b,a]=butter(1,0.9999);           %采用巴特沃斯数字滤波器幅度和相位都满足
                                    要求其实相位的要求是：在通带中没有畸变
                                    即没有衰减
%freqz(b,a,256);
[hlf,f]=freqz(b,a,nqam,fs);
hlf=hlf';

%产生信号源
for i=1:N,
        temp=rand;
        if(temp<0.5)
             dsource(i)=0;
        else
            dsource(i)=1;
        end;
```

```
end;

%qam 调制
qamdata=qammodu(dsource,qammode);      %qam 调制函数，参数为输入的信
                                       号源很采用的进制数，此处默认为
                                       16am

%串并转换
subdsource=cbtrans(qamdata,SubchaNum);

%iff 变换
[numrow,columnum]=size(subdsource);
for i=1:numrow,
    for j=1:columnum,
        ifftdsource(j)=subdsource(i,j);
    end;
    ifftdata(i,:)=ifft(ifftdsource,columnum);
end;


%并串转换
data=bctrans(ifftdata)


%加保护时间间隔
j=1;
for i=1:2:2*nqam-1,
 senddata(i)=data(j);
 j=j+1;
end;
for i=2:2:2*nqam,
    senddata(i)=0;
end;



Eav=0.5;
 sgma=sqrt(Eav/snr);
 %[noise1,noise2]=gngauss(sgma);
 noise1=sgma*rand(1,nqam);
```

```
 noise2=sgma*rand(1,nqam);

%去掉保护时间
 j=1;
 for i=1:nqam,
    recedata(i)=senddata(j);
    j=j+2;
end;


%接收信号解调
    recereal=real(recedata)+noise1;
    receimag=imag(recedata)+noise2;

            v=sqrt(-1);
      recedata=recereal+v*receimag;
    % 经过低通数字滤波器后的数据
     %recedata=recedata.*hlf;

%串并转换
fftsource=cbtrans(recedata,SubchaNum);

%fft 变换
for i=1:numrow,
       for j=1:columnum,
       ffttemp(j)=fftsource(i,j);
       end;
       fftdata(i,:)=fft(ffttemp,columnum);
end;


%并串转换
receqamdata=bctrans(fftdata);


resultdata=qamdemu(receqamdata,qammode);

errbitnum=0;
for i=1:N,
     if(resultdata(i)~=dsource(i)),
```

```
            errbitnum=errbitnum+1;
        end;
    end;
end;
pe=errbitnum/N;
errnumber=errbitnum;
%显示部分
pe
errnumber
```

致 谢

我首先要衷心感谢我的导师萧宝瑾教授。在将近一年的时间里，从论文的选题直到论文最后的审阅，萧老师始终给予了细心的指导和大量的帮助。萧老师雄厚的理论基础和丰富的实践经验给我的学习和工作莫大的启发，同时他严谨的治学态度、一丝不苟的工作作风深深的影响着我，将使我终生受益。

在这里要特别感谢信息工程学院副院长张雪英教授给予的指导和帮助。在撰写论文的过程中，张老师对我提出了许多宝贵的意见和建议，使我得以顺利完成论文工作。

另外，我还要感谢所有关心和帮助我的朋友，他们给予了我诸多的鼓励、启发和帮助，使我感受到集体的温暖和互相协作的愉快。

最后我还要把深深的谢意献给我的家人，是他们在生活和学习上给予了我无私的关爱、照顾和支持，使我顺利完成硕士阶段的学业。