题目：脑机接口载波提取算法的 FPGA 设计与实现

作者：焦晓军

导师：陈亚光 教授

学位: 硕士

学校：中南民族大学

时间：2005

摘　　要

选择适当的信号分析方法从脑电记录中提取用户的信息和命令，是改进脑计算机接口通信速率的方法之一。因此，我们构建了一个基于自回归模型和小波变换多分辨分析的诱发电位单次提取的算法。但该算法计算复杂度较高，BCI 系统的数据量较大，仅靠软件实现往往难以满足实时性的需求，为此我们提出来基于 FPGA 的 BCI 载波提取算法实现方式。

实时信号处理系统中，底层的信号处理算法计算量大，对处理速度的要求高，但运算结构相对比较简单，适于用ＦＰＧＡ进行硬件实现。高层处理算法的特点是所处理的数据量较低层算法少，但算法的控制结构复杂，适于用寻址方式灵活、通信机制强大的ＤＳＰ芯片来实现。

对系统综合分析后，我们提出了选用 DSP 和 FPGA 相结合、构建实现 BCI 接口载波提取算法硬件平台的这一设计方案。

根据上述技术方案对算法进行分解后，提出了具体的硬件体系结构。结构中

充分利用了流水线处理模式，发挥了双端口 RAM 器件的优势，以保证数据处理的高速、高效性。

随后按照自顶向下的设计方法，通过对算法的特点进行分析，把算法划分为不同的功能子模块，包括接口单元、存储单元、运算单元、控制单元等。分别对子模块进行了设计实现。设计过程中，利用了 VHDL 语言的结构严谨、表达清晰的特点，用高层次的描述语言直接去构造最底层的硬件逻辑结构；又充分利用模块化的设计，发挥了可重用技术的优点，使设计层次分明、硬件结构简洁明了。

最后利用集成 EDA 工具 QUARTUS II4.0，结合 ALTERA 公司的 Stratix 系列芯片，完成了 FPGA 开发的全过程。实验结果表明，本文所设计系统能够完成基于自回归模型和小波变换多分辨分析的 BCI 载波的单次提取算法。较好地解决了 BCI 信号处理中实时性的需求。

关键词： 脑机接口 ；小波变换 ；AR 模型 ；FPGA ；VHDL ；EDA

ABSTRACT

A method of improving information transfer rates of BCI is to select an appropriate signal processing algorithm to extract the user's messages and commands from his electroencephalograph (EEG). Then we develop an algorithm of estimation ERP from a record with the statistically independent background EEG noise based on autoregressive model and wavelet multiresolution analysis. But the algorithm have high computational complexity and the amount of data from scalp to be processed is large, it is difficult to implement the algorithm in real-time only by software achievement. So we put forward the method of extraction the BCI carrier of communication based on FPGA.

In our case, the low-level signals processing procedure with the characteristic of high computation load but relatively simple computational complexity, is fit to be implemented by hardware. The high-level processing procedure with low computation load but computational complexity high, is fit to be implemented by DSP. So, we bring forward a design project to construct the hardware platform combined DSP with FPGA for extraction of BCI carrier of communication.

According to this project, we developed a hardware system structure aimed at this idiographic application. In this structure we take full advantage of waterline processing mode and dual-port RAMs to ensure data processing with high speed and high efficiency.

Subsequently we adopt the Top-Down design method, and divide the algorithm into several different function modules, includes interface block, memory block, operation blcok, control block, and so on. Then design and achieve the modules respectively using the high level description language－VHDL.

Finally by means of the integrated EDA tool QUARTUS II4.0 and combine the Stratix device, we complete the design. Experiment results showed that, this system can achieve Single trial ERP extraction for brain-computer interfaces in real time.

Key Words：BCI ；Wavelet Transform ；AR Model ；FPGA ；VHDL ；EDA

第1章　绪　论

1.1 研究背景与现状

1.1.1 脑机接口技术的最新进展

直接用大脑思维活动的信号与外界进行通信，是人类追求的目标。近十年来，国际上许多实验室一直在探索大脑－计算机接口(Brain-Computer Interface, BCI)技术[1-5]。预计，这种全新的通信技术能够用于辅助控制交通工具、武器和其它系统[2]，包括为那些神经肌肉受损，不能使用常规通信手段的患者提供和外界进行交流的方法。为此，目前各国有许多研究小组专门从事这方面的研究工作，美国国立卫生研究所(National Institutes of Health, NIH)和美国国防部高级计划署(Advanced Research Projects Agency, APRA)，也对这个生物医学工程的新项目给予了特殊的支持[2]。

自上世纪 90 年代末以来，在美国“ 人类脑计划 ”的资金支持下， BCI 的研究逐渐成为热点。NATURE 杂志 2000 年刊载了题为“ Real Brains for Real Robots ”的文章,报道了从猴子大脑皮层获取的神经信号，实时控制千里之外的一个机器人，实现了“ Monkey Think, Robot Do ”。近年来，BCI 研究在质和量两个方面都有了迅速增长，1995 年只有不到 6 个研究小组，至 2000 年研究小组的数量已超出了 20 个。2002 年美国国防部高级研究计划署（DARPA）出于军事目的投入了巨资，从而兴起了一个脑－机接口研究的高潮。现在世界各地已有近百个研究小组专门从事该领域的研究工作。1999 年，美国 NIH 在纽约发起了首届关于 BCI 技术的国际会议，50 位来自多个国家代表 22 个小组的研究者参与了会议。此次会议回顾了 BCI 研究历程及现状，定义了 BCI 研究和应用的基本目的，明确和强调了关键的技术问题，考虑了研究规程和评估方法标准，标志着这一热点领域的形成。随着 DARPA 的介入，该领域的研究不断升温，吸引了全世界众多的研究小组参与其中。2002 年 6 月在同一地点又举行了第二次 BCI 的国际会议。92 位来自美国、加拿大、欧洲和中国代表 38 个不同研究小组的研究者参与了会议。这次大会的主题定为“ Moving Beyond Demonstrations（超越演示） ”。

2003 年 3 月，IEEE 生物医学工程学会举行的首届神经工程国际会议，与脑－机接口有关的论文竟达一半以上。各种新技术、新方法层出不穷，朝实用化方向全面推进。在我国，2003 年 10 月在无锡召开的中国生物医学工程学会电子学分会会议上，许多与会者对此领域表示了极大的兴趣，纷纷投入此热点领域的研究之

中。

脑－计算机接口(BCI)，是一个不依赖外周神经和肌肉组织等通常大脑输出通道的通信系统。经过几年的努力，人们已经在实验室构建了一些不同的 BCI 系统[6−13]，这些系统使用来自头皮的 EEG(Electroencephalograph)信号或来自大脑皮质单个神经元的信号。他们尝试利用这些系统去控制光标运动、选择字母或图标、运行某个神经补偿装置、操纵飞行模拟器等。这是一些激动人心的进展，具有很大的理论意义和应用前景。

BCI 系统的通信带宽是相当低的，在最好的情况下只能提供每分钟 5－25 比特的速率[1]，远不能满足正常的需求。速度和准确性的提高取决于信号处理、翻译算法和用户训练方面的改进。这些改进依赖于各学科间合作的增强，如神经科学家、工程师、计算机程序师、心理学家、康复专家，取决于用于评估的客观方法的广泛应用。

1.1.2 本课题的引出

本课题是国家自然科学基金项目“脑－计算机接口的新构思与新技术的研究”的一个组成部分。为了便于分析，首先简介一下项目的主要思想，然后讨论本文打算解决的问题。

1.1.2.1 项目总体介绍

本项目拟构建一个新颖的BCI系统，该系统让被试观察呈现在计算机屏幕上的由36个字符构成的虚拟键盘，采用模拟人类自然阅读的诱发模式，大脑自主地进行选择性输入。系统工作过程中，用户注视虚拟键盘上不同的字符时，从头皮检测到的从事选择活动直接相关的认知信号的时域分布则不同，经计算机自动识别后，确定被试所选定的字符对象，最终使被试的意愿在屏幕上以英文句子的形式显示出来，完成人类大脑和计算机之间的通信。系统结构如图1.1所示：

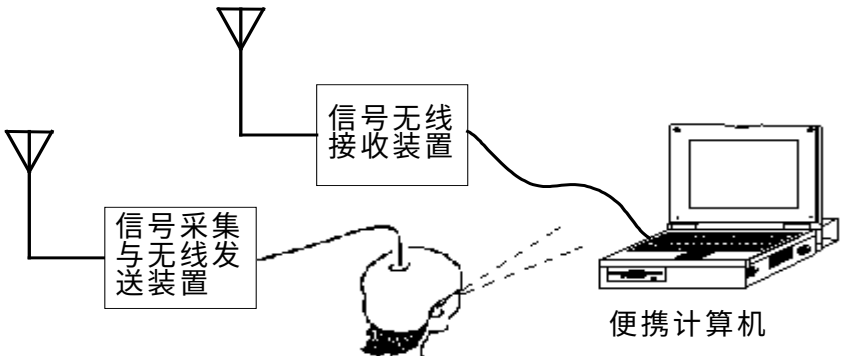


图1.1 脑计算机接口的系统构成

它包括ERP采集和无线发送装置，该装置用于获得来自头皮的ERP电位，经量化编码后以无线的方式传送出去。作为配套设施，我们拟开发一个无线接收装置接收这些数据并将其送到一个便携式计算机中，该计算机的主要任务为：⑴在屏幕上为用户构建一个特殊的虚拟键盘；⑵工作过程中提供模拟人类自然阅读的诱发模式并提取ERP信号；⑶根据所得到的ERP信号和编解码方案进行解码，从而确定被试所选择的字符；⑷由选定的字符组成英文句子，在屏幕上表述被试的意愿。其主要研究内容如下：

①脑－计算机通信载波的选取

选择适当的EEG成分作为脑－计算机通信的载波是至关重要的问题。本研究中，拟使用我们在国家自然科学基金资助下研究成功的一种新技术[14]：采用模拟人类自然阅读诱发模式所得到的ERP成分作为通信的载体。

采用模拟自然阅读的诱发模式有两个优点：第一它能抑制大脑产生外源性成分（诱发噪声），作业过程中从头皮检测到的ERP信号具有较高的信噪比，使得ERP成分的提取变得容易，提高了通信得可靠性；第二，每一次诱发作业中，ERP成分均伴随目标字出现，避免了常规模式下为产生P300而添加的大量冗余作业，提高了通信的速率。

②构建虚拟键盘

为了使 BCI 系统能够完成较复杂的通信任务，还必须解决在多个对象中选定特定对象的问题。为此，我们拟在计算机屏幕上构建了一个虚拟键盘，该键盘和一般虚拟键盘不同，它的每个键粒由键名和一个小视窗组成。如图 1.2 所示：

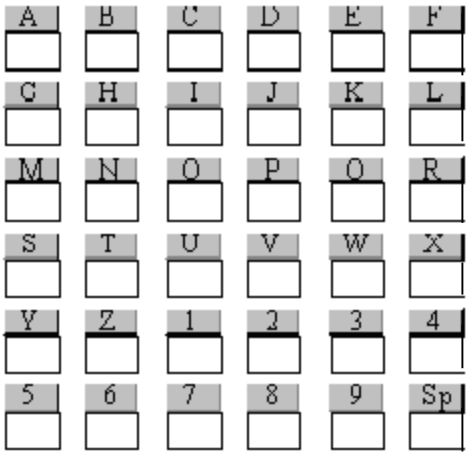


图 1.2 虚拟键盘示意图

该键盘由 36 个键粒构成。键粒的上半部标注有代表该键名称的字符，下半部是一个小视窗。系统运行时，带有指定颜色标记的一组符号串以模拟自然阅读的方式，按一定时间间隔先后平移通过每个小窗视。使用者只需凝视目标字符键下

部的小视窗，并从中搜索特定标记。当特定的标记移过被试注视的小视窗时，将在被试头皮检测到对应的 ERP 成分。

③编码方式

事实上在作业过程中，有 36 个键粒供用户选择，必须能够判定用户当前所选择的是哪一个键粒。为此，我们为每个键粒构建一个特定的诱发符号串，该符号串可以在各自小视窗中匀速移动，不同键粒所对应的诱发符号串的差别在于，目标符号在符号串中所处的位置不同。因此，带有目标符号的诱发符号串在各自的视窗中同时匀速移动时，目标符号进入小视窗的时间也不同。通信过程中，被试注视不同的键粒，该对象所对应的认知信号特征（P200），将在 EEG 记录的不同时段呈现。这样，通信任务就简化成了探测该信号的时域分布，被试打算通信的字符将由该信号在 EEG 中所处的时段不同而确定。

图 1.3 表明了虚拟键盘上 A、B、C 三个键粒所对应的诱发字串。图中深色竖线表示各目标符号中的红色识别标记，由于它们所处的几何位置不同，在其对应键粒的小视窗下同时移动时，红色识别标记出现的时间也不同，依次相差 100 毫秒。

图 1.4 为被试分别注视虚拟键盘中 A、B、C 三个键粒时，在头皮 Fz 处所得到的各自 P200 成分。我们把键粒 A 的目标符号移入其小视窗的时刻定为计时零点，则 B、C 两键粒的对应的目标符号移入 B、C 小视窗的时刻相对于 A 键粒依次延迟 100 毫秒和 200 毫秒。图 1.4.A、图 1.4.B、图 1.4.C 表示了被试注视不同键粒时采集到的 ERP 成分，它们在时段上依次相差 100 毫秒,这样我们就可根据 P200 的时间分布确定被试所选择的字母.

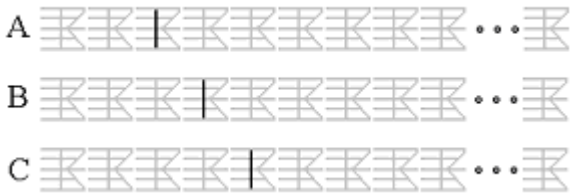


图 1.3　A、B、C 键粒对应诱发字串示例

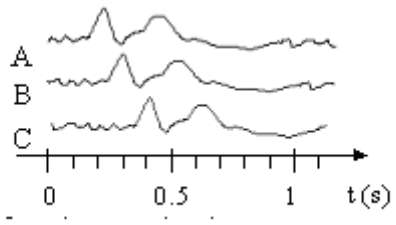


图 1.4 被试分别注视 A、B、C 键粒时所对应的 ERP 成分

1.1.2.2 本文的主要任务

选择适当的信号分析方法[14-25]从脑电记录中提取用户的信息和命令，是改进脑计算机接口通信速率的方法之一。

BCI系统进行信号分析地目的是为了最大限度地提高携带用户信息和命令的EEG的信噪比（SNR），要达到这个目的，必须考虑噪声源的性质[26]，噪声有非神

经源（如眼动、肌电、60Hz工频噪声），和神经源噪声（除了那些对通信有意义的其它EEG信号）当噪声的特征在频率、时段、幅度和所希望得到的信号相似时，噪声的检测和识别是最大的问题。在我们的系统中，来自头皮的ERP实验记录包括自发脑电、肌电及各种环境噪声。对信号影响最大的是自发脑电(EEG)噪声，其幅度远大于诱发响应信号，且其频率和时段与作为通信载波的ERP成分交叠，利用常规的滤波算法难以将其消除。因此，如何从强背景噪声中提取与BCI通信作业有关的ERP信号是首先要解决的问题。

小波变换是ERP提取的另一种常用方法[27-38]。算法从多个尺度和细节对信号进行考察，对信号的时域和频域的局部性质进行分析和处理。它是建立在信号与噪声的小波变换模极大值在不同尺度下的传播行为不同的基础上的。信号的小波变换模极大值随尺度的减小而减小，噪声正好相反。但自发脑电噪声的分布是非高斯性的，不能直接用多尺度小波分析方法确定其局部奇异性，因此，需要首先为EEG噪声建立一个适当的模型，以便对其进行白化预处理。

对于描述EEG的活动，自回归(AR)模型参数估计是一个有用的方法，并可以证明它对于BCI应用的价值[10-12]。为此，本课题组提出了一个基于自回归模型和小波变换多分辨分析的诱发电位单次提取的算法。算法的处理流程是，先在记录中采集一段与作业无关的自发脑电噪声，计算出适合于该噪声的AR模型参数，构成白化滤波器，使得ERP记录通过该滤波器后有色的自发脑电噪声变为白噪声。然后基于信号与噪声有不同的局部奇异性、小波变换模极大值在不同尺度下有不同传播行为的原则，进行消噪，去掉由噪声产生的模极大值点及其模极大值小波域，保留信号的奇异性。最后对信号进行重构和还原处理，得到去噪后的近似诱发电位。

现在的问题是在我们的处理过程中，每次作业记录长度为 3.6 秒，为了判别被试所选择的具体目标，需要将来自头皮的实验数据间隔 100 毫秒进行部分重叠的分段，每段时程为 1 秒，然后对所有数据段分别进行处理。由此可以看出，每次作业后，需要处理的数据量是很大的。为了满足实时性的要求，我们还必须设计一个适当的实现方式。

方案一是依靠通用计算机及特定的程序完成相应信号处理算法，具有实现简单，参数调整方便等优点，但在我们的系统中运算的速度难以满足实时性的要求。

方案之二是选择大规模或超大规模集成的微处理器，包含数字信号处理器（DSP）来完成信号处理算法[39]，将使运算的速度较通用计算机有较大的提高。它们依靠执行不同的软件程序，几乎可以完成任意种类的系统功能，因此在电子系统中获得了广泛使用。但正因为它们是依靠软件工作的器件，导致其工作处理速度受到限制，在工作速度有较高要求的场合，仍不能获得令人满意的结果。

随着电子设备及 IC 芯片设计的不断复杂化和计算机技术的发展，人们已开始

利用计算机的高速运算特性与逻辑分析能力，在电子设计领域进行仅靠人工将难以完成的例如 VLSI 器件或复杂电子系统的设计、分析工作，由此诞生了一门新兴的 EDA（电子设计自动化）技术。这是一门综合了现代电子与计算机技术的最新研究成果，即以计算机为工作平台对电子线路或系统进行自动化设计与应用的计算机辅助设计技术。EDA 正在逐步取代人工进行复杂电子系统或 VLSI 器件的设计、分析与仿真过程，成为设计、研制、应用现代定制或半定制 ASIC 器件必不可少的技术基础。

目前 EDA 技术已开始进入第四个发展阶段，各种设计软件日益齐全，CPLD 和 FPGA 器件的集成越来越高，FPGA 器件已经达到 300 万门和 200MHz。较复杂的算法可以在单片 FPGA 芯片实现。其运算速度远远超过了 DSP。例如，对于 1024 点的 FFT 运算，采用 TI 54X 系列的 DSP 作为处理器，100MIPS 的处理能力，从输入到输出需要 0.3ms；采用 100MHz 的 FPGA，仅需要几个周期的延迟，即几十 ns 后就可得到输出。软件方案的延迟是硬件方案的 1000 倍以上，显然硬件方案的效率高很多。当数据量增大或算法更加复杂时，硬件方案的效率优势将更加显著。

但是硬件方案效率的提高是以增加资源为代价的，有时甚至会超过器件本身所能提供的资源。对系统综合分析后，我们提出了选用 DSP 和 FPGA 相结合、构建实现 BCI 接口载波提取算法硬件平台的这一设计方案。即实现基于自回归模型和小波变换多分辨率分析的诱发电位单次提取的算法时，把 AR 模型的参数识别等结构复杂的算法用 DSP 去实现，将数据量大、运算结构相对简单的算法用 FPGA 芯片设计的专用硬件实现，以满足实时性的需求。

1.2 本文内容安排

具体结构安排如下：

第二章研究算法的构架设计和方案确定，主要分析 FPGA 设计所要达到的要求指标，经过对算法合理分解，提出了新的硬件架构，及工作流程，确定了设计方案。并介绍了设计所采用的硬件平台及软件平台。

第三章进行可编程逻辑芯片的具体设计实现，主要从各个子模块入手，详细讲述了接口单元、存储单元、运算单元、控制单元的具体实现过程。

第四章是系统的仿真和验证。简单介绍了仿真的目的和方法，分别对设计子模块进行了时序仿真和功能验证，与 Matlab 的仿真结果进行了比较。

最后是结论和展望。

第2章　设计方案

2.1 脑机接口系统算法的分解与划分

实时信号处理系统中，底层的信号预处理算法处理的数据量大，对处理速度的要求高，但运算结构相对比较简单，适于用ＦＰＧＡ进行硬件实现。高层处理算法的特点是所处理的数据量较低层算法少，但算法的控制结构复杂，适于用寻址方式灵活、通信机制强大的ＤＳＰ芯片来实现。

在本文中，诱发脑电信号提取算法的流程如图 2.1 所示：

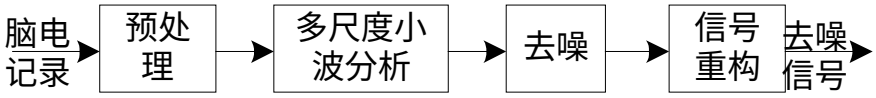


图 2.1 脑电信号处理的流程图

其中，预处理部分主要解决自发脑电噪声的白化处理。首先根据一段与作业无关的自发脑电信号进行 AR 模型参数识别，然后利用得到的模型参数构造白化滤波器，对全部脑电记录进行白化滤波，为随后的小波去噪打下基础。信号重构包括小波合成，逆白化滤波。

经过分析可知，AR 模型参数计算以及去噪部分计算量小，但是控制复杂。因此这部分由 DSP 采用软件实现。而白化滤波、小波分解、小波合成、逆白化滤波部分算法结构简单，计算量大，采用 FPGA 实现。

本文主要讨论相关算法的 FPGA 实现。

2.2 基于 FPGA 实现相关算法的总体设计目标

总体设计目标有以下 7 个方面组成：

1．能够实现预期的算法功能，即实现对数据的白化滤波、小波分解与合成、逆白化滤波等功能。

2．在应用过程中，信号的采样频率较低，而 FPGA 处理数据的时钟频率很高，因此需要进行适当的数据存储，匹配 FPGA 内部与外部工作频率的差异。

3．数据是一定时间段长的信号，所以设计对象除了能对一定长度的数据段进行处理外，还必须提供良好的接口，以便与外部进行数据交换。

4．考虑不同的情况对运算结果的分辨率要求不同，相应的处理过程也会不同，要求系统能够由输入参数进行控制，满足不同变换级数的需要。但级数不超过 6 级。

5．AR 模型参数是随具体样本不同而变化的，系统必须根据这种变化调整滤波器的系数，随时响应参数的改变。

6．AR 模型选择固定的 7 阶。

7．系统以单片或片上单元的形式实现。

2.3 硬件平台

选择可编程逻辑器件 FPGA 作为硬件平台。FPGA 器件可以通过系统内部的重新配置来改变逻辑功能，这个能力赋予系统设计者一个新的自由度，即硬件的功能可以通过软件改变，使得更新或修改设计变得十分容易。

采用动态新配置的方法，使 FPGA 在不同的时间执行不同的功能。利用更新配置逻辑实现系统的自诊断，产生适应不同运行环境的能力。另外，利用可重新配置的 FPGA 器件能够简化硬件的设计和诊断，缩短了产品的上市时间。

本文采用 ALTERA 公司的 Stratix 系列器件，具体型号及资源见附录 B。

该系列器件主要特点包括：

1 高性能体系：Stratix 系列器件的新结构采用了 DitrectDriveTM 技术和快速连续 MultiTrackTM 互联技术。MultiTrackTM 互联技术可以根据走线不同长度进行优化，改善内部模块之间的互联性能。DirectDriveTM 技术保证片内所有的函数可以直接连接使用同一布线资源。这两种技术与 QuartusII 2.0 以上版本软件提供的 LogicLock(tm)功能相结合，便于进行模块化设计，简化了系统集成。Stratix 系统器件片内的全局和本地时钟资源提供了多达 40 个独立的系统时钟，有利于实现最丰富的系统性能；全新的布线结构，分为三种长度的行列布线，在保证延时可预测的同时，增加了布线的灵活性。

2 大容量存储资源：Stratix 器件中的 TriMatrix 存储结构具有高达 10Mbit 的 RAM 和高达 12Tbps 的峰值存储带；有三种不同的嵌入存储模块类型，它们都具有混合宽度和混合时钟模式嵌入移位寄存器功能，可用于多种不同的场合。

3 高带宽 DSP 模块：Stratix DSP 模块包括硬件乘法器、加法器、减法器、累加器和流水线寄存器。各个功能单元之间有专用的走线，具有针对 Stratix 器件内部大量存储器的专用存储器结构接口，因此通过优化设计，DSP 模块可提供高达 2.0GMACS 的 DSP 性能，并且具有尽可能小的布线拥塞。

2.4 软件平台

采用 QUARTUS II4.0 软件平台。这是一个集成的 EDA 工具，可以完成整个集成电路设计过程中的所有工作，包括设计输入、仿真、综合、布线、下载等，其开发流程如图 2.2 所示。最新版本支持 ALTERA 公司的全系列的产品。

为了提高设计效率，优化设计结果，很多厂家还提供了很多专业软件，用以配合芯片厂家提供的工具进行更高效率的设计。最常见的组合是同时使用专业 HDL 逻辑综合软件和集成开发工具。

当前所流行的 HDL 语言中当数 VHDL 和 Verilog，但是考虑到 VHDL 语言设计技术齐全、方法灵活、支持广泛，其系统硬件描述能力强，高层次的行为描述和低层次的 RTL 描述和结构描述的混合使用的便捷，以及其设计编程可与工艺无关和作为 IEEE 的工业标准易于共享和复用等特点，因此采用 VHDL 语言来进行设计。

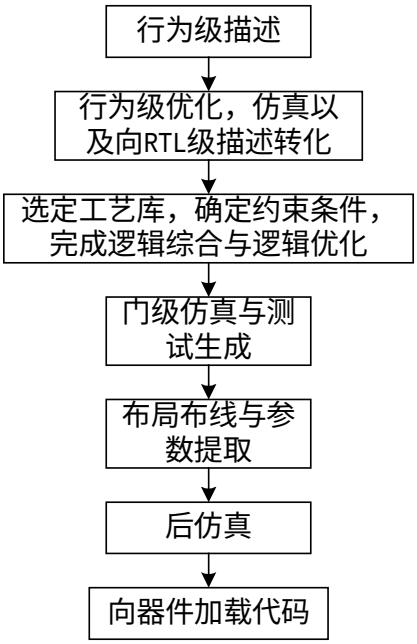


图 2.2　EDA 软件 QUARTUS II4.0 的设计流程

2.5 设计方法

1 采用自顶向下的设计方法。从系统总体要求出发，自上至下的逐步将设计内容细化，然后完成系统硬件的整体设计。在现今电子系统的规模和功能日趋复杂的情况下，这种设计方法已被人们广泛采用。

自顶向下正向设计流程包括如下几个方面：根据功能要求进行系统设计构建整体框图；将系统按功能细分，划分子系统模块；进行逻辑设计；根据逻辑图或功能模块进行电路设计；由电路图设计版图或形成网格表；最后进行工艺设计。

2　IP（Intellectual Property）核重用的方法[40]。IP核模块有行为（Behavior）、结构（Structure）和物理（Physical）三级不同程度的设计。对应描述功能行为的不同分为三类：即软核（Soft IP Core）、完成结构描述的固核（Firm IP Core）和基于物理描述并经过工艺验证的硬核（Hard IP Core）。设计中，采用了IP核重用的方

法，可以有效降低工作量、节省开发时间。

3　利用 VHDL 语言在较高的层次完成设计输入，然后经 EDA 工具进行综合、适配，形成网格表文件，即可下载到具体 FPGA 器件工作。

VHDL 的设计方法的主要优点可归纳为如下几点：

⑴ VHDL具有功能强大的语言结构，可读性强，可用明确的代码描述复杂的控制逻辑设计，并且具有多层次的设计描述功能，支持设计库和可重复使用的元件的生成。

⑵ VHDL允许设计者不依赖于器件，具有相对的独立性。同一设计描述，可以采用多种不同器件结构来实现其功能。若需对设计进行资源利用和性能方面的优化，也并不是要求设计者非常熟悉器件的结构，从而可以集中精力从事设计构思。

⑶ 可移植性强。VHDL的设计描述可以被不同的EDA工具支持，可以在不同的仿真工具、综合工具、工作平台上执行。

⑷　用 VHDL 语言编写的源程序便于文档管理，和设计结果的交流、保存、重用。

2.6 FPGA 内部功能单元的划分

按照设计目标，把 FPGA 内部分为四个子功能模块：即接口单元、存储单元、运算单元、控制单元，如图 2.3 所示：

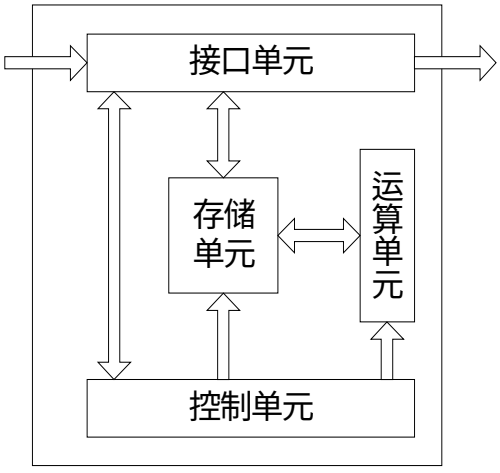


图 2.3　FPGA 内部功能单元的划分框图

接口单元用于完成数据的输入输出以及数据的切换。控制单元用于控制各部分的时序和对数据的读写操作，它们的电路形式不具备规则性，因具体结构而定。存储单元用于存储外部输入的数据、参数和运算单元产生的中间结果。运算单元负责对数据进行运算处理。

一个完整的系统架构均具有以上四个单元，但各种硬件架构的主要区别在于

对存储单元和运算单元的设计和安排。其设计目标是，通过对数据的合理调度和安排，使运算单元能高效地完成计算任务，从而实现对数据的变换与输出。

为提高系统性能常采用的并行处理技术有三类，流水线技术，阵列处理机技术和多处理机技术[41,42]。流水线技术通过时间重叠来提高效率，利用了时间并行性。阵列处理机用多个同步工作的算术逻辑部件来获得空间并行性。多处理机系统则通过共享资源的相互作用处理机来获得异步并行性。

阵列处理机和多处理机系统通过多个处理单元来提高运算能力。在滤波器系统中，本文所涉及地 AR 模型滤波器、小波变换等，其处理单元往往都是乘法累加单元，单元面积都比较大，因此通过阵列处理机和多处理机技术来实现高性能是不利于单片实现的。尤其当滤波器阶数较高的时候更是如此。因此，在进行算法的结构设计时应尽可能利用流水线技术，而阵列处理机和多处理机只能有限制的使用。流水线结构的特点是把整体运算划分为若干部分，各部分在同步时钟控制下依次运算，从而提高数据的吞吐率，提高系统处理能力和硬件利用率。选择流水线级数的一般原则是尽量使流水线内部各部分运算花费时间均匀，并且尽可能避免装配流水线时常遇到的流水线障碍。

一般来讲，流水线级数的增加有利于提高计算吞吐率，但增加流水线级数会导致系统造价的提高，并加大系统编程难度。除此以外，引入流水线气泡的可能性随之增大。尤其是在迭代算法中，计算的中间结果往往是下一步运算所必须的，因而引入流水线的气泡的可能性就更大，使系统资源利用率下降。因此流水线级数往往由算法固定的并行性决定。选择必须折中考虑，级数通常不超过 6。

为了提高系统的性能，应尽可能利用算法的并行性，但同时应努力避免硬件代价的过快增大，因此读取数据时原始输入数据和滤波器系数是并行读入的，而缓存和输出在时间上也应重叠起来以避免因等待而引入流水线气泡。

2.7 运算单元和存储单元硬件架构的安排

首先对运算单元进行详细的分析，比较各种实现结构的特点，找出合理的解决方案。FPGA 内部算法包括 AR 模块（包括 AR 白化滤波器和逆白化滤波器）、小波变换模块（包括小波分解和小波合成）两大部分。这两个部分相对独立，数据所需经过的运算流程如图 2.4 所示：

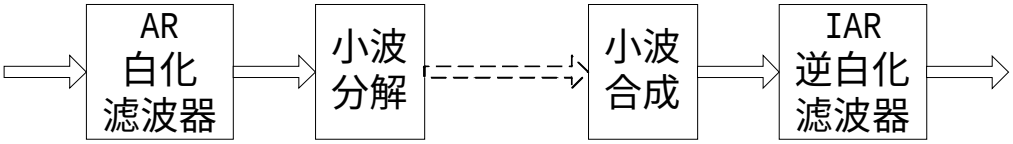


图 2.4　FPGA 运算单元的算法流程图

数据处理算法大致可以分成两个阶段。第一阶段，数据经过 AR 逆滤波器进行

白化处理，然后进行多级小波分解；第二阶段，首先进行多级小波合成，然后由 IAR 滤波器进行逆白化处理，即还原处理得到重建的信号输出。

按待处理数据的特点从整体架构上进行考虑，分为半无限流结构和分帧处理结构两种方式。半无限流结构指数据连续输入，每输入一个数据就处理一个数据，并输出相应结果。在本应用中，电极的采样频率是 1kHz，即每毫秒一个数据，对于 FPGA 来说，处理级别是纳秒级，因此如果采用半无限流处理的方式，绝大多数时间是在等待数据，将会发挥不出器件速度上的优势。而且考虑到 DSP 部分 AR 模型参数提取算法和小波分解系数滤波算法的特点，都需要有足够长的一段数据才能处理，因此分帧处理是更合理的处理方式。

所谓分帧处理结构，即数据按一定的长度进行分段形成一系列的数据帧，然后对每一个数据帧连续处理。帧内数据作为一个整体，帧与帧之间互相独立。为了使系统满足实时性要求，必须对一帧数据进行处理的同时，输出上一帧处理结果和接受下一帧数据。只要完成一帧数据变换所需时间小于一帧数据的周期，即可保证对外部系统的实时关系。这实际上是利用了流水线的工作原理。如图 2.5 所示，一帧数据必须经过输入、变换和输出三个阶段。为满足实时性，要求数据变换所用时间不超过帧周期即可，这里数据的输入输出是同步的。

为了满足实时性的需求，选择合适的数据帧长度非常重要。一般帧长取 2 的整数幂。考虑到我们要处理的信号特征及利用 DSP 估计 AR 模型参数的需要，帧长取 512 个采样，对应的帧周期为 512 毫秒。因此需要保证完成全部帧处理和输出的时间小于此时间，才能使流水线正常工作。

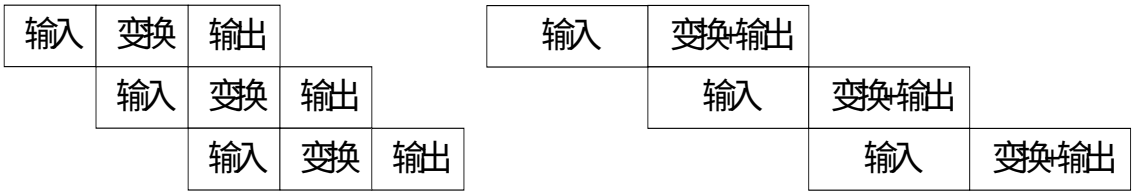


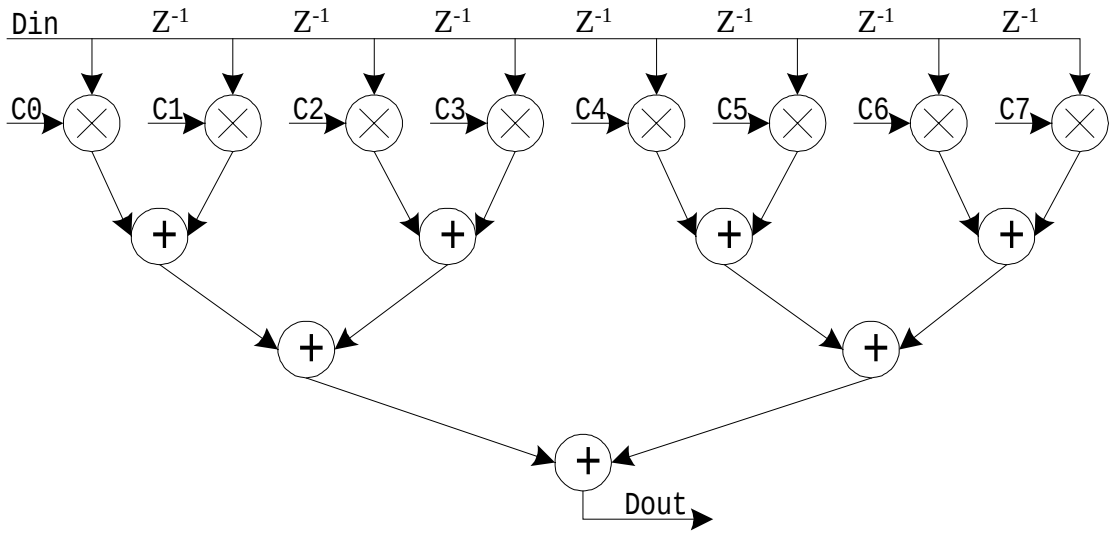
图 2.5　流水线结构示意图

输入输出同步的流水线 3 级结构如图 2.5 中的左图所示。其输入、变换、输出过程中存在一个帧周期的延迟。这种较大的延迟不利于提高系统的实时性。根据应用场合需要，为此我们改进流水线结构，把结构划分为 2 级。如图 2.5 中的右图所示：即输入、变换加输出。如此一来减少了系统的延迟，提高了数据输出的速度。但这种情形下数据输出和输入不再保持同步。

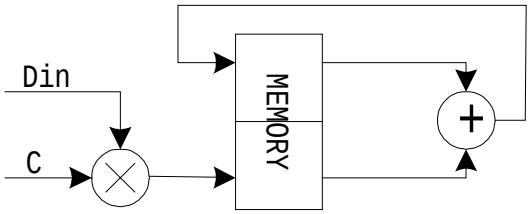
2.7.1　AR 模块的设计

实际上 AR 模块中滤波器的核心是进行固定阶数（7 阶）的卷积运算。数据和

滤波器系数都要求动态输入。卷积主要是乘法和加法运算，因此每输出一个数据都要经过 8 次乘法、7 次加法运算。此时有两种实现方式：并行方式和串行方式，分别如图 2.6 的(a)、(b)所示，



（a）并行方式结构图



（b）串行方式结构图

图 2.6　AR 模块两种运算方式的结构图

并行方式中乘法、加法同步处理的优点是处理速度快，但也存在使用乘法器、加法器较多的缺点。虽然串行方式只需要一个乘法器和一个加法器即可完成相应操作，如图 2.6 中(b)所示。但需要增加存储器的开销。除此以外，由于运算采用串行方式，不利于对实时性要求较高的场合。

考虑到 Stratix 系列器件的特点：即每个器件都包含一定数量的 DSP 模块，每一 Stratix DSP 模块可提供多达 8 个运行在 250MHz 的并行乘法器，各功能单元之间有专用的连线，保证了高速处理数据的能力，为 AR 模块各滤波器选择并行结构提供了极为便利的硬件平台。

综上所述，我们选择并行结构实现 AR 模型滤波器的功能。

接下来确定并行处理的等级。并行性可以分成不同的等级，而且从不同的角度看，等级的分法也不一样。从系统中处理数据的角度来划分，并行性等级从低到高可以分为以下四种，见图 2.7 所示。

· 位串字串——同时只对一个字的一位进行处理。
· 位并字串——同时对一个字的全部位进行处理。
· 位片串字并——同时对许多字的同一位(称位片)进行处理。
· 全并行——同时对许多字的全部或部分位组进行处理。

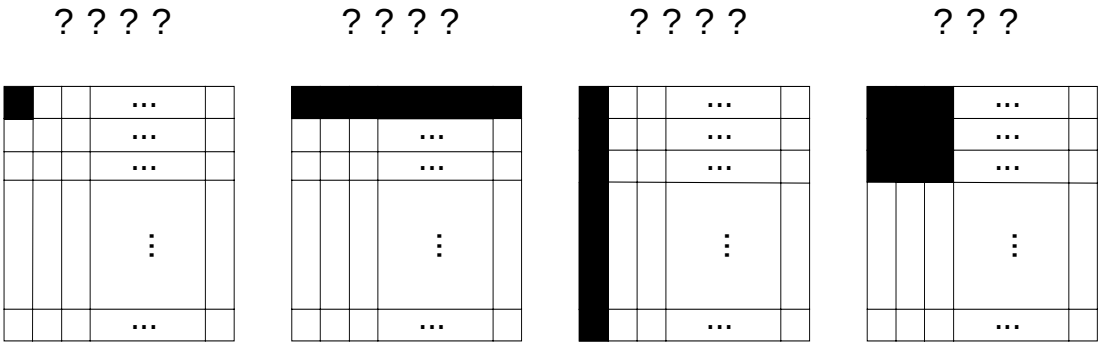


图 2.7 从处理数据的角度划分的并行处理的等级

从处理数据的角度划分的并行处理的等级，从低到高反应了硬件实现的比例在增大，故其实现是一个软硬件功能分配的问题，常需要折中权衡，不同等级的并行处理方案必然导致不同的系统实现方案。本文所用的并行处理方式属于最高等级的全并行方式，设计充分利用了 FPGA 器件丰富的资源，通过占用更多的资源来实现 AR 模块各滤波器的并行结构，即用空间换取较高的处理速度。

2.7.2 小波变换模块的设计

小波变换模块分为小波分解与合成两部分[43,44]，其结构如图 2.8 所示。本系统必须完成小波的多尺度分析，要求对小波变换级数能动态地进行调整。

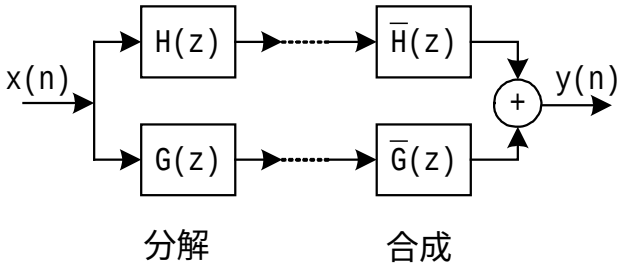


图 2.8 小波变换结构原理图

直接按照小波变换算法的结构进行系统结构设计将需要大量的滤波器，实现每个滤波器都需要大量的逻辑单元，当小波变换级数较高时，耗费的硬件资源将会急剧增长，甚至难以在现有的器件上实现。另一方面，小波变换的级数必须能够动态的进行调整，若采用固定变换级数结构的方式实现小波变换，将无法实现多分辨率分析的设计目的。常用的解决方法是，采用折叠结构来解决滤波器数量

的问题，从而只用一组完整的滤波器来完成多级运算。但是过去由于受器件资源所限，所以这种结构不适于高阶滤波器。同时对小波函数的选择也有一定限制，因为滤波器的复杂度和小波函数直接相关，小波函数的支集越长，则对应的滤波器阶数越高，从而处理单元的复杂度越高。以上原因造成小波变换以单片或片上单元形式实现上的困难。现在，随着 FPGA 器件的快速发展，器件的资源和性能都有了极大的提高，为单片形式实现小波变换算法设计提供了新的选择。

小波变换的折叠结构，是利用资源重复使用来节省系统开销，因为它本质上是串行结构，所以必然会造成处理速度的下降，不过现在器件主频的提高，能够弥补这些速度上的损失。此外，需要增加额外的存储资源的开销，来缓存数据和中间结果。但是 Stratix 系列器件都提供了大量的存储资源，这些存储资源不占用器件的逻辑单元，因此设计中充分利用这些存储资源，节省宝贵的逻辑单元资源。图 2.9 所示为采用折叠结构的多级小波分解与合成的实现框图。

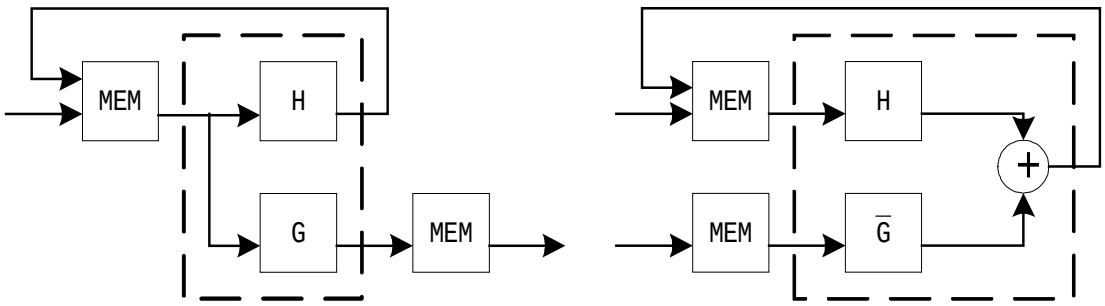


图 2.9　多级小波分解与小波合成的结构框图

图 2.9 中左图为小波分解的框图，单级小波分解包括一个低通滤波器 H，和一个高通滤波器 G。输入信号经过 H 滤波器得到小波分解系数的低频分量，即近似分量。输入信号经过 G 滤波器得到小波分解系数的高频分量，即细节分量。细节分量包含了需要提取的突变信号成分和大量噪声，细节分量是需要进入下一个处理过程进行消噪或提取处理的。多级小波分解，实际上是对得到的近似分量继续进行 H、G 滤波，最终得到多个细节分量和一个近似分量。因此，在小波分解运算单元加入低频分量存储器和高频分量存储器，用来保存输入数据和变换结果。工作流程为：将输入信号存入低频分量存储器，然后开始第一级小波分解，将分解后的低频分量重新写入低频分量存储器，将高频分量写入高频分量存储器；继续下一级分解，同时向外部输出高频分量。为了提高小波分解运算单元的工作速度，并节省存储资源，采用双端口 RAM 来作存储器。双端口 RAM 的特点是，能够同时进行读写操作，读写能以不同的时钟频率工作，非常适合做不同时钟频率区域的接口，起到数据缓冲的作用。用在此处，在低频分量存储器未写满时就开始读取数据进行运算，在高频分量存储器未写满时就向外部发送数据，这种处理方式可以有效的节省等待时间，提高处理速度。

图 2.9 中右图为小波合成的框图，单级小波合成包含一个低通逆滤波器 H*（B 样条小波变换的低通逆滤波器 H*与低通滤波器 H 的滤波器系数是相同的，设计结构也是完全一样的，所以下文中均以 H 滤波器表示），一个高通逆滤波器 G*，以及一个加法器。工作流程为：小波分解后的低频分量送入低频分量存储器，外部输入的处理后的高频分量送入高频分量存储器，分别进行 H 和 G*滤波处理，把变换结果求和后送入低频分量存储器；然后与新输入的高频分量进行下一级的合成，重复进行，直到本帧数据完成相应级数的变换。与小波分解过程类似，此时的高频分量和低频分量存储器也采用双端口 RAM 的结构形式。

2.7.3 存储单元的设计

确定了上述运算单元的结构方案之后，可以进一步确定存储单元的设计方案。本文采用分帧处理的结构，每帧数据 512 个点，输入、输出数据采用 8 位的字长，AR 模型系数也采用 8 位字长。存储器包括数据存储器、AR 模型系数存储器、控制参数存储器。数据存储器主要作用是保存输入数据、中间处理结果、最终输出数据。AR 模型系数存储器是用来存储 AR 模块的系数参数。控制参数存储器是保存系统中需要用到的控制信息，如变换级数。进一步对各子运算单元所需存储器细分可知，AR 模块各滤波器需要一个输入数据存储器，一个系数存储器，一个输出数据存储器；小波分解时需要一个低频分量输入存储器，一个高频分量输出数据存储器；小波合成时需要一个高频分量输入存储器，一个低频分量输入存储器。考虑到，AR 白化滤波器的输出数据存储器可以同时作为小波分解低频分量输入存储器，小波合成低频分量输入数据存储器可以同时作为 IAR 逆白化滤波器时的输入数据存储器。因此共需要 6 个数据存储器，每个数据存储器大小为 512*8 位。因为 AR 模型阶数为 7，所以 AR 模型系数存储器大小为 8*8 位。控制参数存储器，可以保存变换级数和其他选择信息，分配 5*8 位就已足够。因此共需要 6*512*8 + 8*8 + 5*8 ＝ 24 680 位存储空间。对于 Stratix 系列器件，均含有大量 RAM 资源，最少的 EP1S10 器件共有 920 448 位 RAM，这些 RAM 已足够满足本设计需要并提供了扩展空间。

2.8 设计中的一些注意问题和原则

2.8.1 数据类型和表示方法

在用硬件实现算法时，数字的表示方法是一个很关键的问题。一般情况下，定点数的实现方式具有更高的速度和更低的成本，浮点数则具有更高的动态范围，且不需要换算，这对较复杂的算法更有优势。

本文中采用定点数的方式，因为算法结构相对简单和数值范围变化小。所有数据、AR 滤波器系数均以二进制补码的形式表示。输入、输出数据均为 8 位，1 位符号位，7 位整数位，可以表示-128～127 之间的所有整数。AR 滤波器系数一般为纯小数，也为 8 位，1 位符号位，7 位小数位，可以表示-0.9921875～0.9921875 之间的小数，步长 0.0078125。

在 AR 模块各滤波器内部，涉及到 8 位数据与 8 位系数的乘积，因此结果被扩展为 16 位，包括两个符号位，7 个整数位，7 个小数位。在累加求和中，基本不会发生溢出，因此不再添加保护位进行扩展。为了和外部单元接口一致，输出时对变换结果进行处理，仍保持 7 位整数位和一位符号位。

在小波分解和合成模块内部，由于已知 B 样条小波的滤波器系数，需要根据具体的情况决定字长。中间计算过程相应的需要扩展字长才能保证精度，在乘积、累加时一般不发生溢出，取输入数据和滤波器系数字长之和来表示即可。所有的滤波器系数，都用最优 CSD 码表示[45]，即转化为 2 的幂级数的和的形式，在硬件实现的时候，将用移位器来代替乘法器，加快运算时间和节省资源。

2.8.2 边界效应的处理

无论是 AR 模块还是小波变换模块，其核心都是卷积运算。数据关系复杂程度随着滤波器阶数增加而增加，由于采用分帧处理，在两个边界会产生截断，造成较大的误差，这就是边界效应。一般用软件方式实现算法时，可以对数据进行各种方式的延拓来解决边界效应；但在用硬件方式实现时，为了简化结构，对信号的边界常采用补零方式处理。一般情况下，补零方式误差较大，影响帧的两个边界的处理结果的准确性。

本文中，在运算模块设计时，采用输出滞后输入若干时钟周期的方式，对于数据帧头，采用周期延拓的方式处理边界效应；对于数据帧尾，则采用补零方式处理。

2.8.3 时钟的处理

在现代的集成电路芯片中，随着设计规模的不断扩大，一个系统中往往含有数个时钟。多时钟域带来的一个问题就是，如何设计不同时钟频率区域之间的接口电路。本系统中，充分发挥双口 RAM 的特点，解决不同时钟频率区域的接口匹配问题，是解决这个问题的一种简便、快捷的方案。

2.8.4 设计中的原则

采用单独编写 VHDL 代码文本生成功能模块，和调用系统模块、由模块向导生成功能单元相结合的原则进行设计。既可以保证设计的自主性、灵活性，根据具体的需求量身定做；又可以发挥系统已有的 IP 核的优势，快速生成所需的功能单元，减少工作量。

一般系统中的 IP 核都是通用核[46,47]，考虑了满足多种应用的需要，所以存在较大的冗余度，对特定的应用会造成资源浪费。手工编写的 IP 核，都是针对具体的情形，最大程度减少了这种冗余，但通用性差。所以把两种设计方法结合起来，发挥两者的优势。

系统的性能取决于系统结构的构造和编码风格。首先，充分理解 VHDL 是一种硬件描述语言，不要用写软件的方法去写 FPGA 程序，它是一种并行的程序。其次，不要只靠通过提高器件的速度等级来使设计的系统达到处理速度要求，而应优先考虑改变设计构架和程序代码的编排来提高系统速度。

第3章　设计实现

FPGA 由三部分组成：可编程逻辑单元阵列块（CLB）、可编程输入/输出单元阵列（IOB）、以及互连资源。其良好的可编程性，使得硬件的功能可以象软件一样通过编程来实现。这种称为“软”硬件的全新的系统设计概念，使新一代的电子系统具有极强的灵活性和适应性，它不仅使电子系统的设计和开发以及产品性能的改进和扩充变得十分简易和方便，而且使电子系统具有适应多功能性的能力，为实现许多复杂的信号处理和信息加工提供新的思路和方法。

设计采用自顶向下的方法，整个设计流程是一个输入、实现、验证的递归过程，直到实现的设计正确及完整。在成熟的代码写入器件前，都是脱离硬件的，EDA 环境中集成有各种器件的库，代替实际器件工作。并且由于器件可以多次编程，在电路级调试设计时不会因某种错误而损坏器件。

3.1 顶层模块的设计与实现

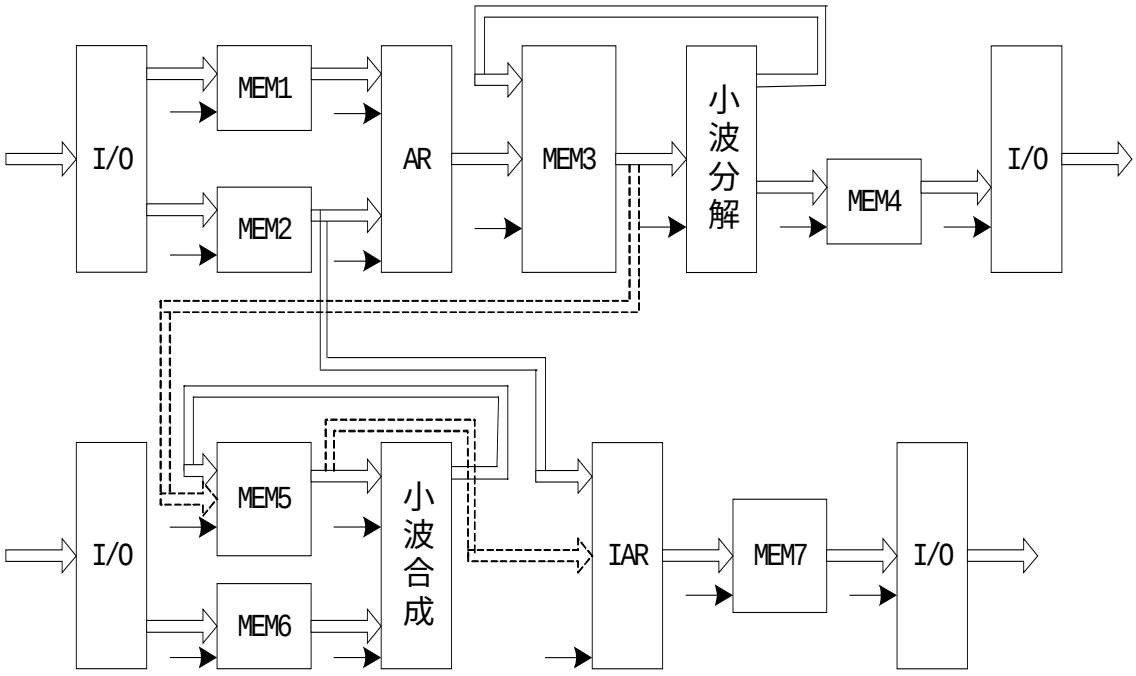


图 3.1　FPGA 顶层模块的系统架构图

根据第二章中进行的方案分析，建立如图 3.1 所示的系统顶层框架。空心箭头表示数据或系数的流向，实箭头表示控制信号的方向。控制信号主要由读写地址产生器、状态机产生。MEM1～7（MEM2 除外）是 512 字节的双端口数据存储器，它可以同时以不同的时钟频率分别进行读写，起到连接两个不同时钟区域的作用。

MEM2 是 AR 模型系数存储器。系统工作流程大致划分为两部分。

1．数据帧的白化及小波分解过程

外部输入的数据及 AR 模型系数通过接口存入相应的存储器 MEM1 和 MEM2，MEM2 存储完毕后开始输出系数到 AR 模块暂存起来（包括白化滤波器和逆白化滤波器），以供计算时调用。由于 MEM1 的容量远远大于 MEM2，所以这个过程在 MEM1 存储数据期间可以完成。

当 MEM1 完成存储后，开始向 AR 白化滤波器模块发送数据，AR 白化滤波器模块开始运算并输出结果到存储器 MEM3。

AR 白化滤波器模块的输入、输出时钟频率都采用内部工作频率。当 MEM3 存储器存储完毕，开始向小波分解模块输出。分解变换之后的高频分量输出到 MEM4 存储器，低频分量输入 MEM3 存储器。此时变换级数减 1，MEM4 中的高频分量开始向外部输出，MEM3 中的数据继续进行小波分解，直至变换级数减至 0。当分解全部结束时，MEM3 中的数据开始向 MEM5 中备份。

2．数据帧的小波合成和逆白化过程

外部处理后的高频分量从最大尺度到最小尺度依次由 I/O 接口存入 MEM6 后，同 MEM5 中的低频分量一起送入小波合成模块，经过变换后得到上一级尺度的低频分量并回存到 MEM5 中。然后 MEM6 通过 I/O 口得到新尺度的高频分量，会同 MEM5 中已更新的数据继续进行下一级的小波合成变换，直至一帧数据的小波合成过程结束。实际过程中，当用于小波合成的各尺度高频分量成分依次送入 MEM6 后，控制逻辑将产生一个输入控制信号表示高频分量输入结束，通知运算模块为下一帧的处理作好准备。

随着控制信号的到来，MEM5 中的数据开始输出到 IAR 逆白化滤波器模块进行逆白化处理，处理后的结果存入 MEM7 中。

3．工作时钟频率的划分

MEM1 有两个不同的时钟频率，它的输入时钟频率由外部时钟提供，输出时钟频率采用系统内部工作频率。最终根据设计的仿真结果进行调整输入、输出时钟之间的关系，在数据输入到一定数量后即开始同时输出处理的结果，利用时间复用提高系统速度。MEM3 的读写时钟频率均采用内部工作频率，MEM4 则不同，其写时钟频率采用内部工作频率，读时钟频率由外部控制。

MEM5、小波合成模块、逆白化滤波器模块的时钟频率采用内部工作频率，MEM6、MEM7 读写时钟频率分别由外部和内部提供。

处理流程的第一部分和第二部分的工作基本上是独立进行的，但当 AR 模型系数存储器输出时，以及 MEM3 存储器向 MEM5 备份时存在时序上的逻辑关系。这带来一个问题，即不但在两部分内部充分利用流水线的方式提高处理速度，还要

兼顾在两部分之间通过流水线的方式提高处理速度。

经过分析我们发现 AR 模型系数存储器只是在第一帧数据处理前输出，然后在 AR 模块内部缓存，使用时在内部调用，因此不会有冲突。但当小波合成过程中向 MEM5 回存数据时同时发生 MEM3 存储器向 MEM5 备份将造成数据丢失的问题。为了解决这个问题，我们为 MEM5 增加一个同样大小的数据存储器（MEM8），双缓冲乒乓式处理结构[47]。工作中 MEM5 和 MEM8 轮换作为 MEM3 中低频分量的存储器，如图 3.2 所示。改进之后，第一部分与第二部分之间也可以采用流水线处理方式，如图 3.3 所示。

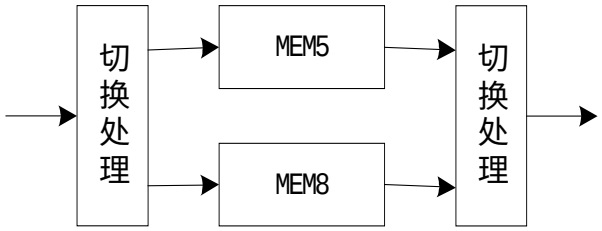


图 3.2 双缓冲乒乓式处理结构

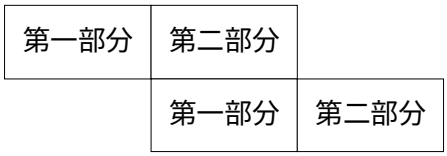


图 3.3 二级流水线结构

3.2 接口单元的设计与实现

接口单元主要负责外部输入和内部存储单元之间的联系，用于管理系统和外部电路之间的数据及控制信号接口关系。除了管理输入输出数据之间的时序关系之外，还用于完成输入数据存储器和滤波器系数存储器间的通道切换。它包括输入处理电路、输出控制电路以及信号切换电路三部分。

接口单元共有三种形式，作为处理过程第一部分的输入接口；作为第二部分的输入接口；作为第一部分、第二部分的输出接口。此处第一部分指的是数据帧的白化及小波分解部分；第二部分指的是数据帧的小波合成和逆白化部分。

作为处理过程第一部分的输入接口的工作流程图如图 3.4 所示。

下列代码显示了作为处理过程第一部分的输入接口的信号定义和设计实现。

```
ENTITY io IS
    PORT(
    din        : IN      STD_LOGIC_VECTOR(7 DOWNTO 0);
    clkin      : IN STD_LOGIC;
    cs         : IN      STD_LOGIC_VECTOR(1 DOWNTO 0);
    clkout     : OUT     STD_LOGIC;
    dout_1,dout_2    : OUT    STD_LOGIC_VECTOR(7 DOWNTO 0));
END io;
```

cs 是 2 个字长的选择信号，当 cs2 为 1 时，使能接口。如果 cs1 为 1 时，表示

输入的是数据，此时 dout_1<=din，即送往数据存储器；如果 cs1 为 0 时，表示输入的是滤波器系数，此时 dout_2<=din，即送往滤波器系数存储器。clkin 是外部输入的同步时钟，clkout 采用 clkin 的时钟频率，与输出数据同步，用作下级存储器的写入时钟。

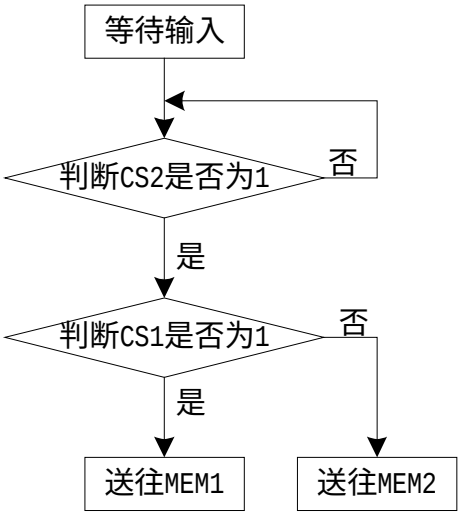


图 3.4　接口单元的工作流程图

对于接口单元的另外两种形式，结构相对更简单，减少了选择判断部分，主要是同步存储器和外部之间的数据输入或输出。

整个接口单元与外部之间的连接信号如表 3.1 所示，

表 3.1　接口单元输入输出信号列表

信号	类型	功能定义
Dain[8]	IN	第一部分的输入数据线
Dbin[8]	IN	第二部分的输入数据线
Daout[8]	OUT	第一部分的输出数据线
Dbout[8]	OUT	第二部分的输出数据线
CS[3]	IN	选择信号线。CS0 为 1 表示输入数据，为 0 表示输入系数。 CS1 为 1 表示使能第一部分输入。 CS2 为 1 表示使能第二部分输入。
CLKA	IN	第一部分输入同步时钟
CLKB	IN	第二部分输入同步时钟
CLKC	IN	第一部分输出数据的同步时钟
CLKD	IN	第二部分输出数据的同步时钟
CLK	IN	系统时钟

S[3]	IN	变换级数选择线。 001：一级变换 010：二级变换 011：三级变换 100：四级变换 101：五级变换 110：六级变换 其它：保留。
Req[3]	OUT	向外部申请信号线。 Req0 为 1 表示第一部分请求输出 Req1 为 1 表示第二部分请求输入 Req2 为 1 表示第二部分请求输出
B	IN	表示一帧数据的高频分量全部输入结束

3.3 存储单元的设计与实现

存储单元主要包括两种形式，一是存储输入、输出及中间结果的存储器，二是存储 AR 模型系数的存储器。它们在容量大小及使用方式上有很大不同。因此需要分别进行设计。存储器主要分为两大类型：随机读写存储器（RAM）和只读存储器（ROM）。两种存储器又各自细分为多种类型的存储器，如图 3.5 所示：

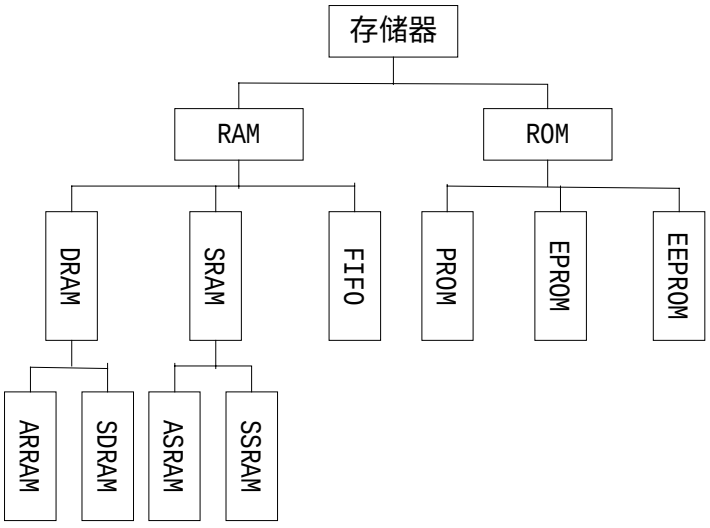


图 3.5 存储器的分类

ROM 一般用来保存永久性的数据，如程序代码、常数表等。RAM 主要用来存储动态数据。本文中设计的存储器都属于 RAM。同步存储器的特点是存取操作用同步时钟控制，因此读写速度快于通常的异步存储；在高速实时信号处理的场合，

异步存储器全部可以采用相应的同步存储器代替。

设计中，可以充分利用 EDA 软件的强大功能和器件丰富的内部资源，即使用 Megawizard Plug-In Manager 功能，由向导生成相应的存储模块，把存储器指定在内部存储区域，而不占用器件的逻辑单元。这对资源宝贵的 FPGA 器件来说是非常重要的。

3.3.1 数据存储器的设计与实现

由上一章的分析可知，数据以帧为单位进行处理，每帧 512 个 8 位数据，所以存储器大小为 512*8 位。

为使系统能够工作于较高的速度，必须尽可能的采用并行处理，为此对于数据的存取采用并行方式[48]，这要求存储器的读操作口和写操作口可独立工作，即为写－读双口存储器（dual-port RAM）。设计代码如下：

```
entity sdram is
generic(    width : integer:=8;
              depth : integer:=512;
              addr : integer:=9);
port( di    :in std_logic_vector(width-1 downto 0);
        do    :out std_logic_vector(width-1 downto 0);
        we,re,wrclk,rdclk    :in std_logic;
        raddr,waddr    : in std_logic_vector(addr-1 downto 0));
end sdram;
```

上述代码显示了数据存储器的接口定义，数据字长为 8 比特，地址字长为 9 比特，数据深度为 512。读写操作完全独立，有各自的使能、时钟、地址信号。操作时序如图 3.6 所示：

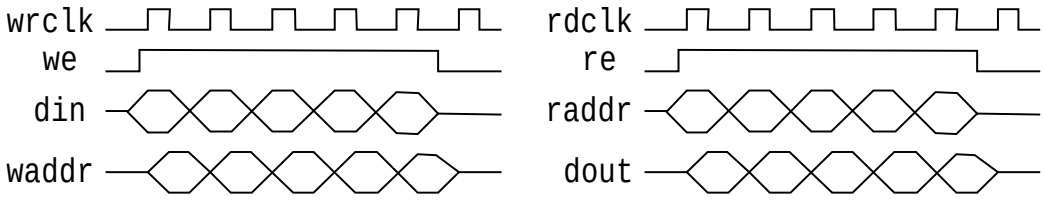


图 3.6　数据存储器的写和读操作时序图

3.3.2 AR 模型系数存储器的设计与实现

系数存储器大小为 8*8 位，可以采用 FIFO 结构来实现。FIFO 中数据的存放

结构与一般的 RAM 一致，主要区别是读取方式不同。它主要作为堆栈缓冲区，先进入的数据先输出。

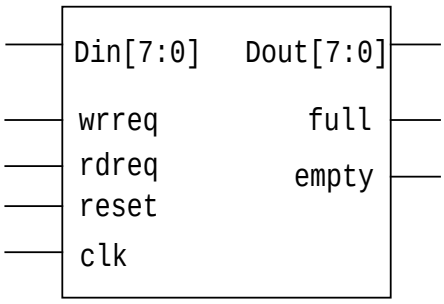


图 3.7　AR 模型系数存储器的封装图

AR 模型系数存储器的封装如图 3.7 所示，Din 是 8 位数据输入线，Dout 是 8 位数据输出线，wrreq 表示写请求信号，rdreq 表示读请求信号，reset 是复位信号，clk 是同步时钟，full 是存储器满标志，empty 是存储器空标志。它省去了读写地址线，可以灵活的控制输出方式。

工作方式描述如下：当写请求信号 wrreq 为 1 时，如果 full 信号为 0 表示存储器尚有空单元，在同步时钟 clk 的作用下，Din 上的数据开始向 FIFO 中写入。每写入一个数据，内部写地址自动加 1，当存储器写满时 full 信号输出 1，此时不能写入。当读请求信号 rdreq 为 1 时，如果 empty 信号为 0 表示存储器中尚有数据等待读取。在同步时钟的作用下，FIFO 中的数据开始逐个输出到 Dout 上，每读取一个数据，内部读地址自动加 1，直至全部读取完毕，empty 信号输出 1，此时不能继续读取。由内部写地址和读地址的偏差来控制 full 和 empty 信号的状态，当偏差为 0 时，empty 置为 1；当偏差为 7 时（即最大存储容量），full 为 1；其它值时，保持不变。当 reset 为 1 时，复位内部写地址和读地址的值，均清为 “ 0 ”。

3.4 运算单元的设计与实现

运算单元在设计方案中划分为三个功能子模块：AR 模块、小波分解模块和小波合成模块。其中 AR 模块包括 AR 白化滤波器和 IAR 逆白化滤波器。AR 模型固定为 7 阶，它的系数是要求能够动态变化的，因此采用可参数化的设计。小波分解与合成变换，采用特定的三次 B 样条小波，因此滤波器系数是固定的，如表 3.2 所示：

表 3.2　小波变换各滤波器系数表

i	h(i)	g(i)	$h^{\wedge}(i)$	$g^{\wedge}(i)$
-4	0.0000	0.0000	0.0000	0.001953125
-3	0.0000	0.0000	0.0000	0.017578125

-2	0.0625	0.0000	0.0625	0.072265625
-1	0.2500	0.0000	0.2500	0.181640625
0	0.3750	-2	0.3750	-0.181640625
1	0.2500	2	0.2500	-0.072265625
2	0.0625	0.0000	0.0625	-0.017578125
3	0.0000	0.0000	0.0000	-0.001953125
4	0.0000	0.0000	0.0000	0.0000

小波变换每一级，都用到四个滤波器，小波分解高通滤波器 G、低通滤波器 H^*，小波合成高通滤波器 G^*、低通滤波器 H^*。由上表中的系数可以看出，H^*和 H 是相同的，它们的滤波器结构也是相同的，都用 H 滤波器代替。

3.4.1 AR 模块的设计与实现

自发 EEG 是非平稳的随机信号，但在一个短的时间段内，通常可被认为是平稳的。平稳的随机信号可由白噪声激励某一确定系统产生，如图 3.8 所示，

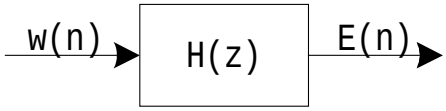


图 3.8　随机信号产生框图

图 3.8 中 w(n)是白噪声激励，E(n)是响应信号。对于自回归（AR）模型，对应的差分方程为：

$$E(n) = w(n) - \sum_{k=1}^{p} a_k E(n-k) \qquad (3\text{-}1)$$

则系统的传递函数为：

$$H(z) = \frac{1}{1+\sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)} \qquad (3\text{-}2)$$

显然系统是可逆的，那么，根据 AR 模型可以构造一个白化滤波器，其系统函数为：

$$A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k} \qquad (3\text{-}3)$$

式中 a_k 是 AR 模型参数，p 是 AR 模型的阶数。自发脑电信号通过系统 A(z) 的输出信号是白噪声 w(n)，如图 3.9 所示：

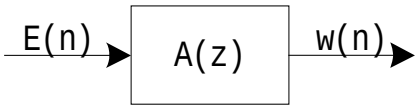


图 3.9　白化滤波器框图

假设 $a_0=1$，

$$w(n)=\sum_{k=0}^{p}a_k E(n-k) \tag{3-4}$$

通过对自发脑电进行白化，把对自发脑电信号的奇异性检测转化为对白噪声的奇异性检测，为小波除噪打下了基础。当系统完成除噪后，需要对经过白化处理的信号进行还原，即逆白化处理。

由以上推导，我们可以写出白化滤波和逆白化滤波的差分方程，即：

$$y(n)=\sum_{k=0}^{p}a_k x(n-k) \tag{3-5}$$

$$y(n)=x(n)-\sum_{k=1}^{p}a_k y(n-k) \tag{3-6}$$

AR 模块的核心设计采用并行处理的方式。在设计时，采用系统模板向导进行设计，以便充分利用芯片内部的 DSP 模块的资源。

在硬件设计中都存在着一个速度与开销的问题，如果想提高速度就要花费更多的资源，而如果想节约资源则要牺牲电路的并行性。利用芯片内部的 DSP 模块可以极大的提高系统的速度，使得系统对于运算单元速度的依赖性减少。根据 Stratix 系列 FPGA 的结构特点和资源情况，以及系统的要求和应用背景，采用改进结构。调用系统内部 IP 核，既充分利用了已有资源，包括软件代码和硬件结构两方面，又可以大大节省开发时间。

图 3.10 所示是采用系统内部的功能单元组成的 AR 模块并行运算核，包括 8 个乘法器、7 个加法器。乘法器、加法器单元的分别定义为 mulcell、addcell，在使用时直接作为器件进行调用。接口定义如下，

```
component mulcell
    PORT(dataa : IN STD_LOGIC_VECTOR(7 downto 0);
         datab : IN STD_LOGIC_VECTOR(7 downto 0);
         result : OUT STD_LOGIC_VECTOR(15 downto 0)
```

```
    );
end component;
component addcell
    PORT(dataa : IN STD_LOGIC_VECTOR(15 downto 0);
         datab : IN STD_LOGIC_VECTOR(15 downto 0);
         result : OUT STD_LOGIC_VECTOR(15 downto 0)
    );
end component;
```

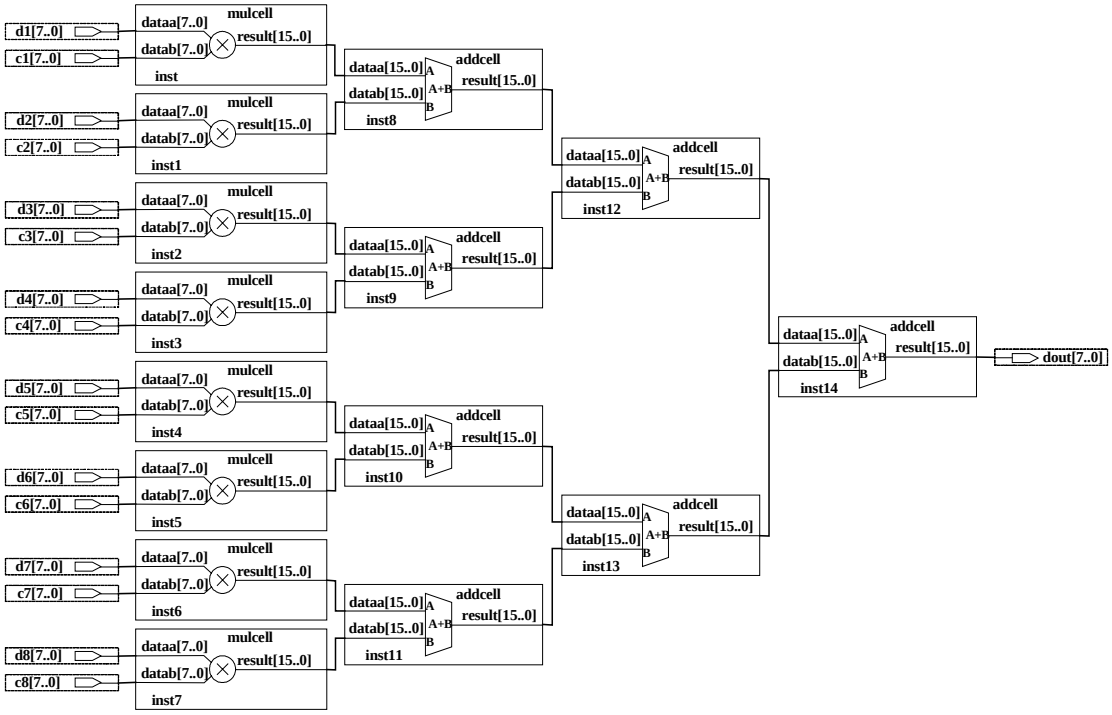


图 3.10　AR 模块并行处理核的结构图

乘法器输入为两个 8 位有符号数，输出是 16 位有符号数。加法器输入、输出均为 16 位有符号数。在进行乘法运算时不会出现溢出，但在加法运算时，有可能溢出。但是考虑到实际情况，系数一般为纯小数，数据均为有符号数，累加后基本不会溢出，因此没有扩展输出字宽。

白化滤波器和逆白化滤波器的设计都直接调用并行处理核进行运算。但是在实际运行时，由于输出和输入信号关系之间的差异，需要根据具体情况对输入信号进行预处理，使其满足数学模型的需要。

对于白化滤波器，从式 3-5 可以看出，当前的输出与当前的输入和在此之此前的 7 个输入有关。为了实现这一点，我们对输出和输入的时序进行了调整，具体做法是输出相对输入延迟 8 个时钟，即第 9 个输入信号来到时，开始运算并输出。

设计代码如下：

```
ENTITY ar IS
    PORT(
            datain,coefin: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
            clk0: IN STD_LOGIC;
            encoef,endata    : IN STD_LOGIC;
            enable   : OUT   STD_LOGIC;
            result       : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
    END ar;
ARCHITECTURE aaa OF ar IS
…
BEGIN
…
U1:arblock   PORT   MAP(co1 ,co2 ,co3 ,co4 ,co5 ,co6 ,co7 ,co8 ,t1 ,t2 ,t3 ,t4 ,t5 ,
t6 ,t7 ,t8 ,do );
    enable<=en;
    temp<=CONV_INTEGER(do);
    result <= CONV_STD_LOGIC_VECTOR((temp/128),8);
END aaa;
```

上述程序段中定义了几个使能控制信号：encoef 是输入 AR 模型参数使能信号，高电平有效，维持 8 个时钟周期；endata 是输入数据使能信号，在整个输入数据期间保持高电平状态；enable 是输出使能信号，高电平有效。在程序结构体中，直接调用所设计的 arblock 核进行运算处理，输出时其运算结果调整为 8 比特字长。

对于逆白化滤波器，从式 3-6 可以看出，当前的输出不但与当前的输入有关，还与过去的 7 个输出有关，为此我们在内部设计了 7 个寄存器用来保存连续的 7 个处理结果，使之能再次被内部运算调用。程序的其余结构类似白化滤波器。

3.4.2 小波分解模块的设计与实现

小波变换的核心算法如式 3-7，

$$y(n)=\sum_{k} g_k x(n-k) \qquad (3\text{-}7)$$

式中 g_k 表示滤波器系数，x(n-k)表示 n-k 时刻的输入数据。

在进行小波分解模块设计时，首先把滤波器系数 g_k 转化为 2 的幂级数和的形

式，表 3.3 所示是 H 滤波器系数的转化结果，

表 3.3　H 滤波器系数转化为 2 的幂级数和的形式

系数	转换为 2 的幂级数表示后的系数
0.0625	2^{-4}
0.2500	2^{-2}
0.3750	$2^{-3}+2^{-2}$

由于输入数据是 8 比特，H 滤波器系数可以用 4 位二进制字长表示，G 滤波器系数只需要用 1 位二进制字长表示即可。为了保证中间计算结果的精度，我们对运算过程中数据字段长度进行了拓展。拓展后的字长除了保证大于数据字长加上滤波器系数字长要求外，还需要考虑累加溢出效应。实际上当进行多级小波分解时，会造成数据范围放大，但每一级分解最多不大于 1 位。综上考虑，统一采用 16 位字宽表示中间结果，既保证了足够的精度，又节约了资源。其设计代码如下：

```
PACKAGE n_bit_int IS
   SUBTYPE BITS8 IS INTEGER RANGE 0 TO 255;
   SUBTYPE BITS16 IS INTEGER RANGE 0 TO 2**16-1;
END n_bit_int;
LIBRARY work;
USE work.n_bit_int.ALL;
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_signed.ALL;
ENTITY dec IS
          PORT(
          din        : IN      BITS8;
          clk  : IN STD_LOGIC;
          dout_1,dout_2    : OUT    BITS8);
END dec;
ARCHITECTURE h OF dec IS
      TYPE STATE_TYPE IS (one,two,three,four,five,six,seven,eight,nine);
      SIGNAL state : STATE_TYPE;
     SIGNAL sum1,sum2 : BITS16;
```

```
    SIGNAL d1,d2,d3,d4,d5        :     BITS8;
BEGIN
    delay:PROCESS
    BEGIN
    WAIT UNTIL clk='1';
    CASE state IS
        WHEN one =>
                d1<=din;
                state<=two;

        WHEN two =>
                d2<=din;
                state<=three;
        WHEN three =>
                d3<=din;
                state<=four;
                sum1<=din+d2*4+d2+d1*8+d1*2;
                sum2<=din*0;

        WHEN four =>
                d4<=din;
                state<=five;
                sum1<=d1+d1*4+d2*4+d2*2+d3*4+din;
                sum2<=d1*2-d2*2;

        WHEN five =>
                d5<=din;
                state<=six;
                sum1<=d1+d2*4+d3*4+d3*2+d4*4+din;
                sum2<=d2*2-d3*2;

        WHEN six =>
                d1<=din;
                state<=seven;
```

```
            sum1<=d2+d3*4+d4*4+d4*2+d5*4+din;
            sum2<=d3*2-d4*2;

        WHEN seven =>
            d2<=din;
            state<=eight;
            sum1<=d3+d4*4+d5*4+d5*2+d1*4+din;
            sum2<=d4*2-d5*2;
        WHEN eight =>
            d3<=din;
            state<=nine;
            sum1<=d4+d5*4+d1*4+d1*2+d2*4+din;
            sum2<=d5*2-d1*2;
        WHEN nine =>
            d4<=din;
            state<=five;
            sum1<=d5+d1*4+d2*4+d2*2+d3*4+din;
            sum2<=d1*2-d2*2;
    END CASE;
END PROCESS delay;
    dout_1<=sum1/16;
    dout_2<=sum2;
END h;
```

上述是小波分解模块的 VHDL 实现代码。它通过对 H、G 滤波器卷积算法的仔细分析，把计算过程分为几个阶段，如图 3.11 所示，

3.11*a* 是卷积算法的分析示意图。首先 H、G 滤波器系数翻转，然后依次与输入数据序列 x(n)相乘求和。运算过程中滤波器系数固定不动，按照粗箭头方向依次移动 x(n)序列。卷积算法得到的输出序列长度是滤波器阶数和原序列长度之和，对于超出长度的部分数据采取截尾方法处理。

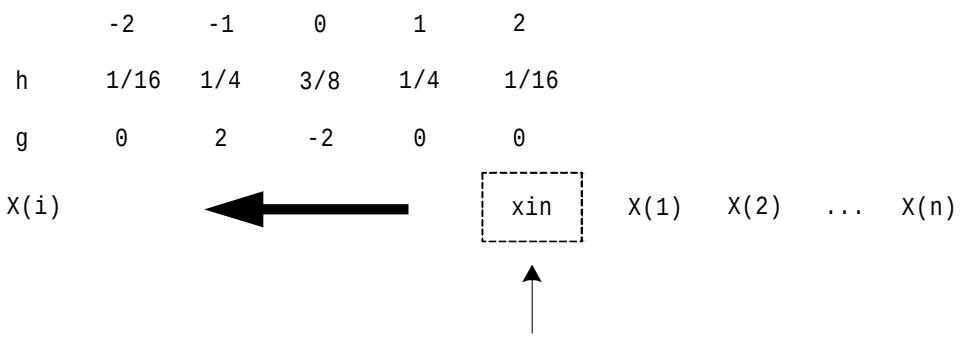
3.11*b* 表示了运算过程的第一阶段。其作用是延迟，仅把输入数据进行缓存，而不执行运算和输出，共需要进行两个时钟周期。

3.11*c* 是运算过程的第二阶段，其作用是处理帧边界。首先对序列进行对称延拓，然后计算卷积结果并输出，共需要两个时钟周期。这使帧头由于截断产生的误差得到解决，只是输出比输入值滞后了两个时钟周期。而帧尾部分依旧采用补

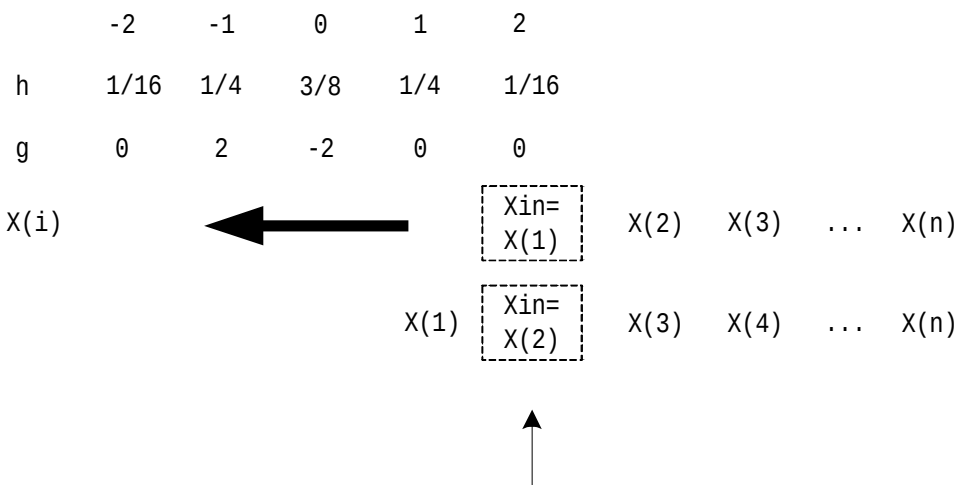
零方法。

3.11*d* 是运算的第三阶段。即对后续的数据进行卷积计算，在程序中是一个不断循环的过程，直至本帧计算完毕。

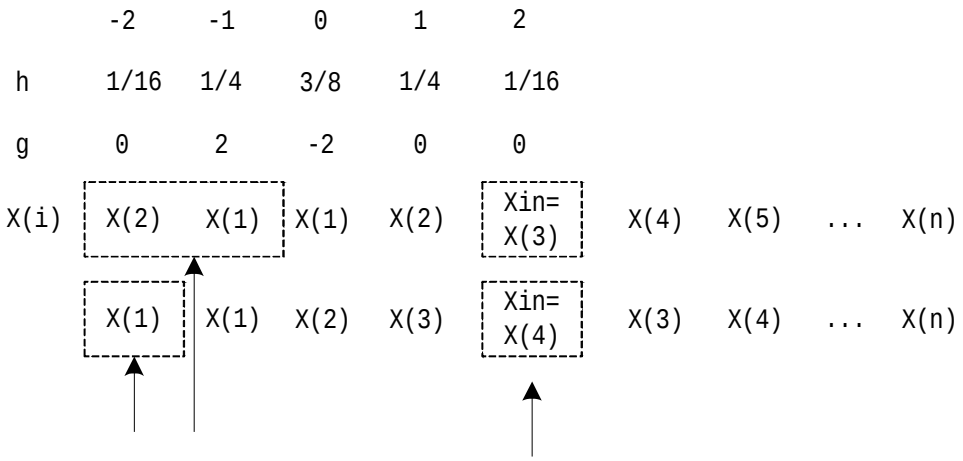
在程序中定义了 9 种状态，第 1、2 个状态是第一阶段，第 3、4 个状态是第二阶段，5～9 这五个状态是第三阶段。



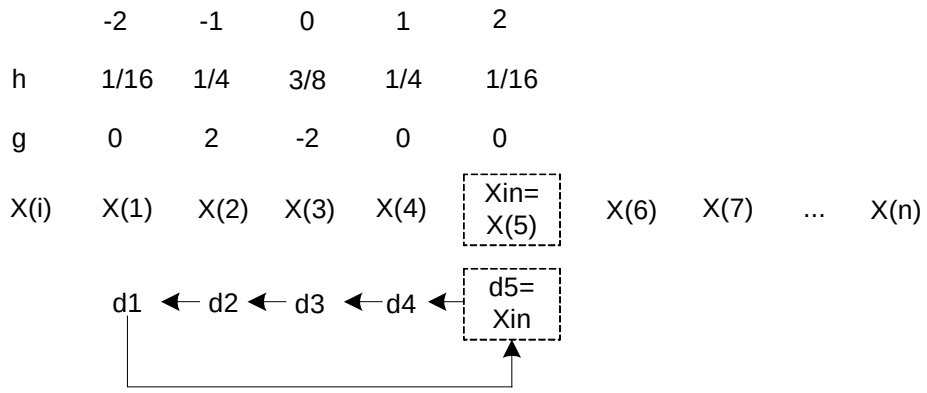
（a）卷积算法的分析示意图



（b）输入数据的延迟



（c） 帧头的对称延拓处理



（d） 数据帧的循环运算过程

图 3.11 小波分解模块的运算过程分析

3.4.3 小波合成模块的设计与实现

小波合成模块的设计和小波分解模块方法完全一致，但是滤波器系数有较大不同，表 3.4 给出了 G^*滤波器系数转化为 2 的幂级数表示后的结果，

表 3.4 G^*滤波器系数转化为 2 的幂级数和的形式

系数	转换为 2 的幂级数表示后的系数
0.001953125	2^{-9}
0.017578125	$2^{-9}+2^{-6}$
0.072265625	$2^{-9}+2^{-7}+2^{-4}$
0.181640625	$2^{-9}+2^{-7}+2^{-6}+2^{-5}+2^{-3}$
-0.181640625	$-(2^{-9}+2^{-7}+2^{-6}+2^{-5}+2^{-3})$
-0.072265625	$-(2^{-9}+2^{-7}+2^{-4})$
-0.017578125	$-(2^{-9}+2^{-6})$
-0.001953125	$-(2^{-9})$

G^*滤波器系数需要用 9 位二进制数表示，所以中间运算过程的数据字长需要进行调整。设计中长度调整为 8+9+1+6＝24 比特。即在滤波器系数与数据长度之和的基础上再增加 1 位累加溢出保护位、6 位多级变换溢出保护位，以便保证中间运算结果的精确性。在输出时，进行适当的舍取，保留 8 位整数位作为输出结果。设计代码如下：

```
ENTITY red IS
        PORT(
        dain,dbin        : IN    BITS8;
```

```
        clk  : IN STD_LOGIC;
        dout      : OUT    BITS8);
END red;
ARCHITECTURE h OF red IS
    TYPE STATE_TYPE IS (s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15);
    SIGNAL state : STATE_TYPE;
   SIGNAL sum1 : BITS24;
    SIGNAL sum2 : BITS16;
    SIGNAL y1,y2 : BITS8;
   SIGNAL da1,da2,da3,da4,da5,da6,da7,da8       :     BITS8;
   SIGNAL db1,db2,db3,db4,db5,db6,db7,db8           :     BITS8;
BEGIN
…
 y1<=sum1/512;
 y2<=sum2/16;
 dout<=y1+y2;
 END h;
```

上述是小波合成模块的程序代码，内部具体流程与小波分解模块的代码类似，故略去。主要不同是定义了 15 个状态，也分为三个阶段：延迟阶段、边界处理阶段和循环运算阶段。系统的输出只有一个，即高通滤波器 G^*与低通滤波器 H^*变换结果的求和。

3.5 控制单元的设计与实现

控制单元的作用是控制整个系统的运行时序、流程，以及产生各个模块所需要的控制信息。包括存储器读写地址、使能信号等。它可以分为两个子模块：地址产生器模块和状态机模块。

实现一个控制功能，可以用有限状态机实现，也可以用 CPU 实现。二者相比，前者性能远高于后者。因为 CPU 需要许多操作（例如取数和执行）和部件（例如数据通路和 ALU 寄存器）。而状态机中，状态存储在多个触发器中，表示行为的代码存储在门级网络中。对于通常的一条选择判断语句，如果用 CPU 实现，一般需要 10～20 条机器指令，其执行时间与 CPU 的速度有关。如果由门和触发器实现，则执行时间为一个时钟周期。因此用 VHDL 实现的状态机的控制性能要优于 CPU 实现的方案。

3.5.1 地址产生器的设计与实现

地址产成器主要用来生成写/读存储器所需要的地址。由存储器的设计可知，只有数据存储器的读写才需要输入地址，因此只需设计一种规格的地址产生器。因为数据存储器的大小为 512，对应的地址空间也是 512，只需要 9 位地址即可满足要求。接口定义如下，

```
ENTITY addrgen IS
    PORT(
    clkin     : IN STD_LOGIC;
    addrout   : OUT     STD_LOGIC_VECTOR(8 DOWNTO 0);
    enable    : OUT    STD_LOGIC);
END addrgen;
```

工作流程为，当使能信号 enable 为 1 时，在输入同步时钟的作用下，开始依次输出地址信号，范围从 0 到 511，逐次加 1。

3.5.2 状态机的设计与实现

状态机定义为用来表现系统内部详细动作的方式。其主要功能是控制系统运行的时序和流程。通常将状态机分为三类，

1. Moore 状态机：次态＝f（现状），输出＝f（现状）。
2. Mealy 状态机：次态＝f（现状，输入），输出＝f（现状，输入）。
3. 混合型状态机。

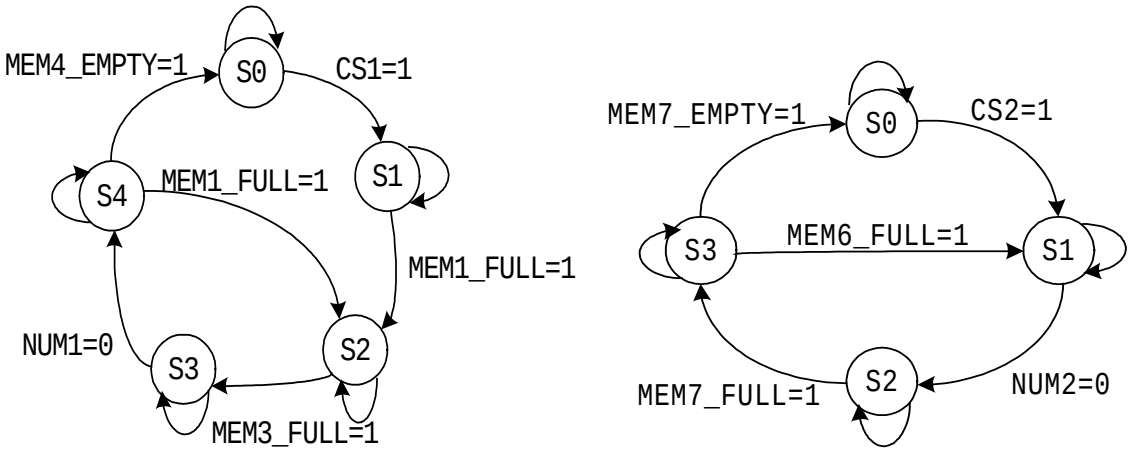
Moore 与 Mealy 两种状态机的不同点在于：Moore 状态机的输出信号随着现状同步变化，一种状态必定对应着某一特定输出。Mealy 状态机的下一状态和输出信号与当前状态和现行输入两者有关。

从实现电路功能的角度来讲，这两种状态机都可以实现同样的功能。但它们的输出时序不同，在选择使用哪种状态机时要根据实际情况进行具体分析。

本文在设计状态机时，按照系统工作在二级流水线的处理方式基础上进行分析。即系统处理流程分为：输入、变换处理（包括 AR 滤波和小波变换）加输出。由于状态的变换需要不断根据中间过程控制信号的变化进行调整，所以采用 Mealy 状态机来实现。

确定了采用 Mealy 状态机后，接下来是构造状态表，确定各个工作状态及其转换条件。进行状态设计时，把处理流程划分为双进程并行状态，分别用 sa、sb 表示处理过程第一阶段和第二阶段的状态，sa 有 5 个状态，sb 有 4 个状态，状态

图如图 3.12 所示：



（a）sa 状态机流程图　　（b）sb 状态机流程图

图 3.12 双进程状态机流程图

图 3.12*a* 描述如下，对于 sa：

S0 表示空闲状态。此时处理过程的第一部分没有动作，始终等待输入。如果输入片选信号 CS1 变高，则进入下一个状态 S1。

S1 表示写存储器 MEM1。此状态过程中，控制写使能信号及地址产生器输出写入地址。首先是向 MEM2 中输入 AR 滤波器系数，然后是向 AR 模块输出系数进行缓存，接下来是向 MEM1 中写数据。当写满标志 MEM1_FULL 变高后，切换状态到 S2。

S2 表示 AR 模块运算状态。包括读存储器 MEM1、变换处理、输出结果到 MEM3，当变换结束标志 MEM3_FULL 变高后，进入状态 S3。

S3 表示小波分解变换状态。包括读 MEM3，分解变换，写 MEM3、MEM4，MEM4 的数据输出等具体动作。直到分解级数全部完成，标志信号 NUM1 为 0 后，进入状态 S4。

S4 表示备份输出状态。包括 MEM3 中的低频分量向 MEM5 中备份，MEM4 中的最后一级高频分量向外部输出，以及外部输入向 MEM1 中写入，这里体现了流水线处理的思想。当输出、备份完成，而没有数据输入时，转到 S0 状态；当 MEM1_FULL 为高后，表示数据输入完毕，此时输出、备份也已完成，转到 S2 状态。

图 3.12*b* 描述如下，对于 sb:

S0 表示空闲状态。此时处理过程的第二部分没有动作，始终等待输入。如果输入片选信号 CS2 变高，则进入下一个状态 S1。

S1 表示小波合成变换状态。包括写存储器 MEM6，读 MEM6 和 MEM5，小波合成，写 MEM5 等动作。当完成全部级数的合成后，标志信号 NUM1 为 0，进

入状态 S2。

S2 表示 AR 模块运算状态。包括读存储器 MEM5、变换处理、输出结果到 MEM7，当变换结束标志 MEM7_FULL 变高后，进入状态 S3。

S3 表示输出状态。包括读 MEM7，写 MEM5、MEM6，等动作。当输出完成，而没有数据输入时，转到 S0 状态；当 MEM6_FULL 为高后，表示数据输入完毕，此时输出、MEM5 输入也已完成，转到 S1 状态。

当外部复位信号 rst 置 1 时，系统复位，sa、sb 均进入 S0 状态。

第4章　仿真验证

4.1 仿真综述

仿真的目的就是在软件环境下，验证电路的行为和设想中的是否一致，即在一个尽可能真实的系统环境中检验内核设计的正确性。这是硬件设计过程中最重要的一个环节，它可以在进行最终硬件固化以前发现设计中的错误及缺陷。尤其对于 FPGA 设计来说，仿真所能达到的效果和代码下载进入具体器件以后运行的效果基本是一致的，所以仿真的正确性代表了设计的正确性。

仿真的分类：

a) 功能仿真：在 RTL 层进行的仿真，其特点是不考虑构成电路的逻辑和门的时间延迟，着重考虑电路在理想环境下的行为和设计构想的一致性；

b) 时序仿真：又称为后仿真，是在电路已经映射到特定的工艺环境后，综合电路的路径延迟和门延迟对电路行为的影响，来比较电路的行为是否还能够在一定条件下满足设计构想。

功能仿真的目的包括一下几个方面：

a）设计出能工作的电路。功能仿真不是一个孤立的过程，其和综合、时序分析等形成一个反馈工作过程，只有这个过程收敛，各个环节才有意义。而孤立的

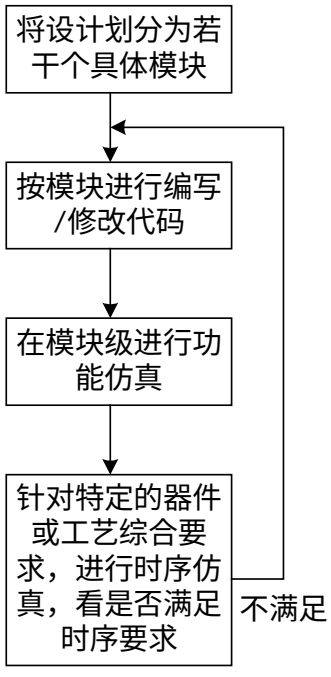


图 4.1 仿真的工作流程图

功能仿真通过是没有意义的。如果在时序分析过程中发现时序不满足需要更改代码，则功能仿真必须从新进行。工作流程如图 4.1 所示。

b）代码排错，功能仿真是代码排错的重要的手段之一。可以说，设计的质量是依靠精心的仿真来保证的。对于复杂系统的仿真，没有一种系统化的方法是不行的。仿真还应该与描述设计同时开始，即设计好一个模块，就仿真一个模块。这样做可以降低整个系统的仿真量，及早发现问题，避免模块的错误向更大的模块系统传播。仿真的覆盖率是个重要的问题，原则上应该将所有的功能都仿真到。

仿真输入的手段有很多，对于时序关系简单、数据量少的仿真，可以通过直接波形输入的方式，方便直观。对于较复杂的数据量大的模块仿真，最好通过仿真测试文本文件进行输入。后者可以得到更大的代码覆盖率，这是仿真过程中一个重要指标。

利用仿真结果查找错误、判断设计是否正确也存在一定问题，一是输入数据的好坏决定了所能查出错误的多少。二是由于输入数据难以穷举，不能保证百分之百的代码覆盖率。尤其是输入激励波形的方式，它本身就是一个复杂的课题。

4.2 各模块的仿真结果

看仿真波形无疑是代码排错的主要手段，它的优点就是直观清楚，便于多个信号的同时观察和比较。下面分别列出各主要功能模块的仿真波形图。

图 4.2 所示是处理过程第一部分的 I/O 输入模块的时序仿真图，可以看到选择信号 sel 为 01 时，输入信号 din 切换到输出信号 dout_1，表示输入的是数据，送往数据存储器；当 sel 为 10 时，输入信号 din 切换到输出信号 dout_2，表示输入的是滤波器系数，送往系数存储器；当 sel 为其它值时，输出均为高阻状态。Clkout 是输出信号的同步时钟。可以看出 I/O 接口的切换功能得到了实现，并且与输入仅有几个纳秒延时，近乎同步。同时可以看到，在切换到输出信号的时候，出现了一个短暂的不是所需要的数值，这是代码的延时造成的，但是它是发生在时钟的低电平时期，所以不会影响正确的输出结果。

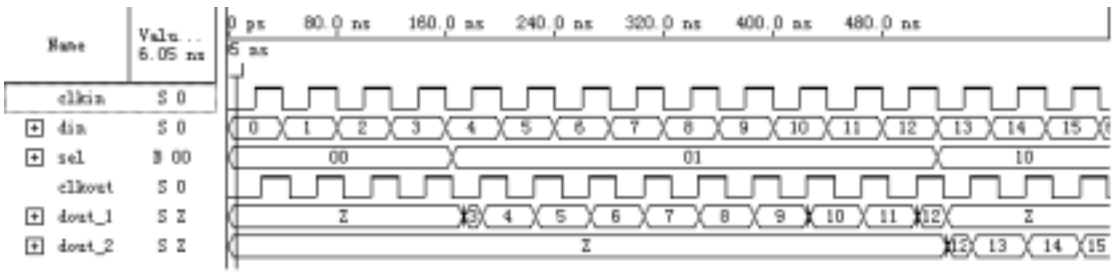


图 4.2 第一部分的 I/O 输入模块时序仿真图

图 4.3 所示是地址产生器的仿真结果，可以看到，当使能信号 enable 为 1 时，

开始顺序输出地址信号，输出与输入时钟是同步的，仅有几个纳秒的延迟。较好的实现了预期的地址产生器的功能。

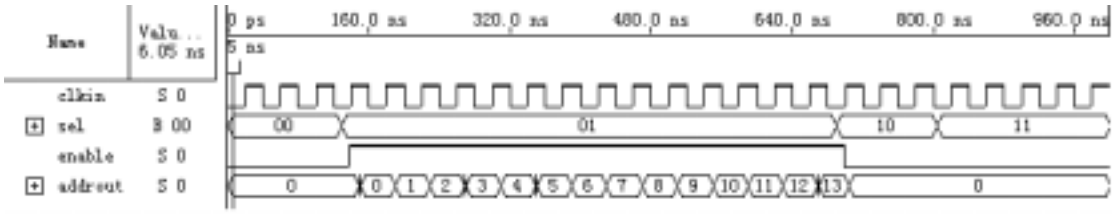


图 4.3　地址产生器的时序仿真图

图 4.4 所示是 AR 模块并行运算核的功能仿真波形图，c1~c8 是滤波器系数，在一组数据计算中是相对固定的，d1~d8 是输入数据，计算过程是有符号数的乘加运算，输出 dout 和理论计算分析完全吻合。此模块共占用 112 个逻辑单元，8 个 DSP 块。代码覆盖率 56.95%。

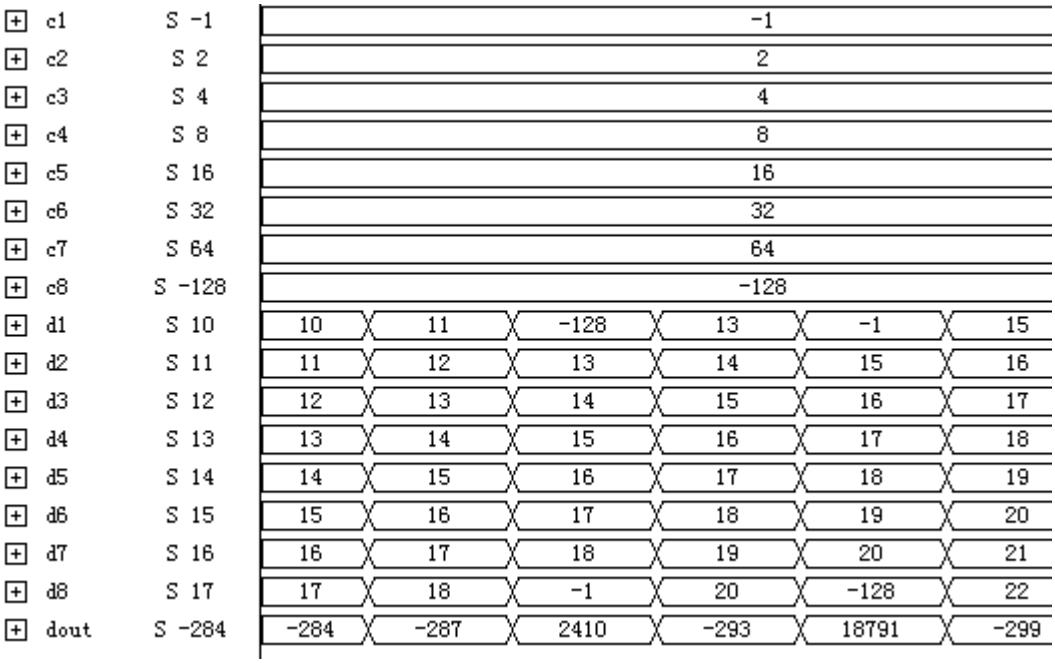


图 4.4　AR 模块并行运算核的功能仿真图

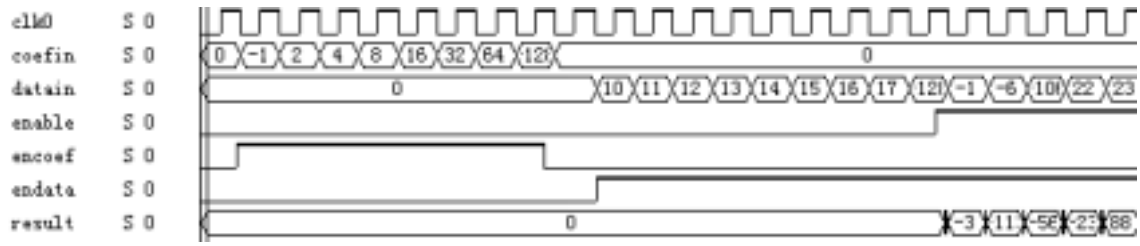


图 4.5　白化滤波器时序仿真图

图 4.5 是白化滤波器的时序仿真图，输出比输入滞后 8 个时钟周期，共占用 537 个逻辑单元，8 个 DSP 块，最高频率可达 239.69MHz。代码覆盖率 56.69%。

图 4.6 是逆白化滤波器的时序仿真波形图，输出比输入滞后不到 1 个时钟周期，共占用 386 个逻辑单元，最高频率可达 65.59MHz。代码覆盖率 77.64%。

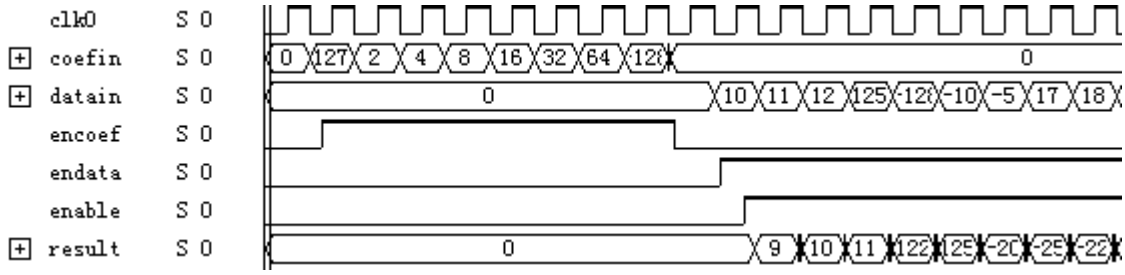


图 4.6　逆白化滤波器时序仿真图

图 4.7 所示是小波分解模块的时序仿真波形图，可以看出，输出比输入滞后 2 个时钟周期，高频分量和低频分量的输出与理论分析完全吻合。共占用 509 个逻辑单元，最高频率可达 89.67MHz。代码覆盖率 99.05%。

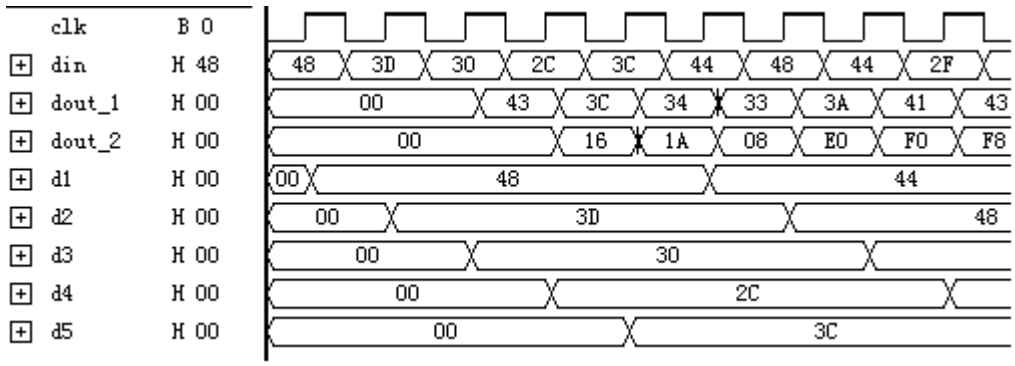


图 4.7　小波分解模块的时序仿真图

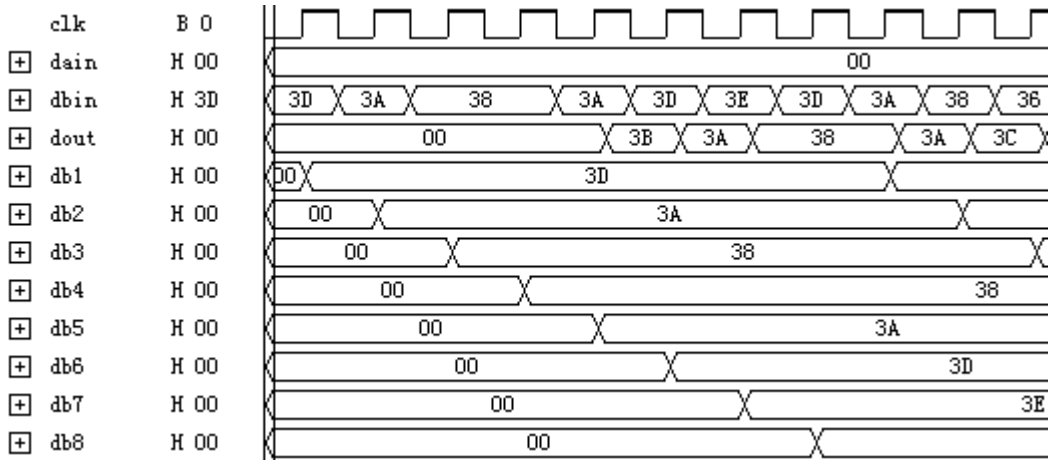


图 4.8　小波合成模块的时序仿真图

图 4.8 是小波合成模块的时序仿真波形图，输出比输入滞后 4 个时钟周期，共占用 3153 个逻辑单元，最高频率可达 28.28MHz。代码覆盖率 70.05%。

4.3 仿真结果的比较与讨论

通过与 Matlab 仿真结果的比对，观察本文设计的处理效果。如图 4.9 所示，(a)是待处理数据，由试验中得到的一条 EEG 信号得到，然后加入一定的白噪声，(b)模拟被噪声污染的情形。（c)、(d）两图分别为调用系统小波函数和利用手工编写函数进行三级滤波消噪的效果，对高频细节分量均采用清零的方式处理。(e）是 FPGA 滤波之后得到的波形。

由各种方式进行滤波处理之后的波形可以观察到，FPGA 滤波效果是比较理想的，和原始信号波形保持着较好的一致性。仅是在帧的尾部有较大误差，这是由于在滤波器的设计时，对帧头部进行了对称延拓，而对于帧尾部仅是进行了补 0 拓展，因此导致帧尾部的变换产生较大误差。

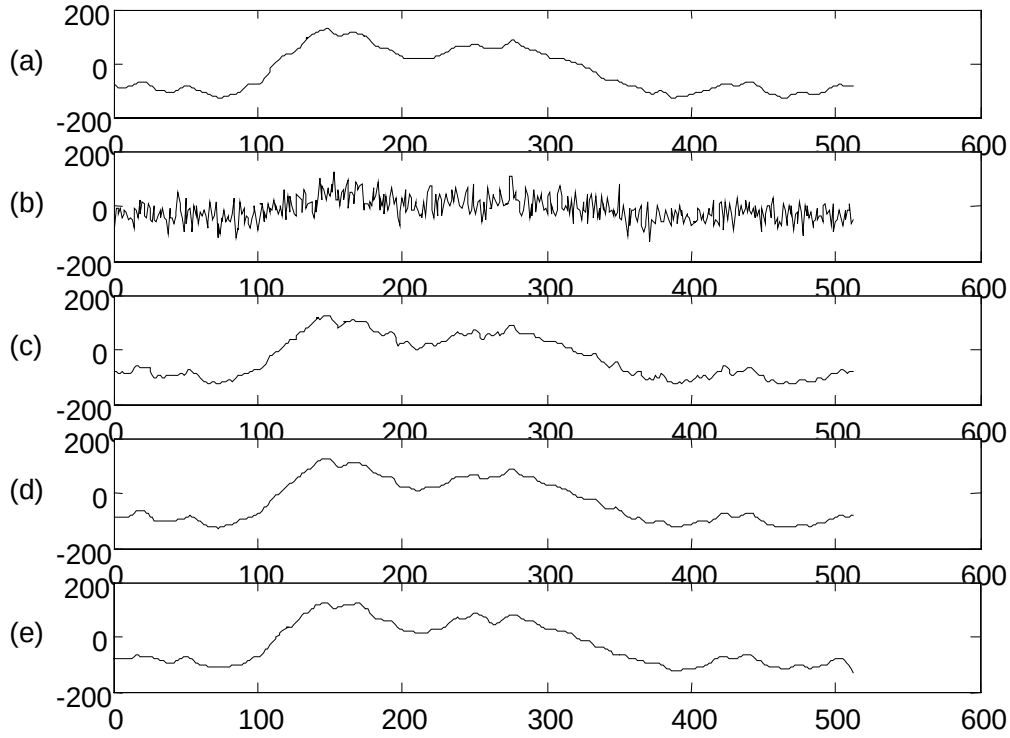


图 4.9　一条 EEG 记录经过不同方式处理后的波形比较

把通过 FPGA 算法进行处理的一帧数据结果，与 Matlab 处理结果进行了比较，如表 4.1 所示。可以看到，FPGA 处理的效果和 Matlab 处理的效果相似，都在原始数据附近有一定的波动，各自在局部数据点上有更优的效果。显然，通过 FPGA 处理所达到的效果验证了本文方案的可行性。

表 4.1　一帧数据经过 Matlab 仿真和 FPGA 仿真后的结果比对（部分）

序号	原始数据量化	Matlab 结果	FPGA 结果	序号	原始数据量化	Matlab 结果	FPGA 结果
441	-68	-72	-63	477	-109	-114	-102
442	-69	-73	-67	478	-110	-114	-105
443	-72	-76	-71	479	-110	-115	-107
444	-75	-80	-76	480	-110	-117	-109
445	-79	-85	-80	481	-111	-118	-112
446	-84	-89	-84	482	-112	-120	-113
447	-88	-93	-85	483	-113	-120	-114
448	-92	-97	-87	484	-114	-120	-113
449	-96	-101	-89	485	-115	-118	-112
450	-101	-104	-91	486	-115	-116	-109
451	-105	-108	-96	487	-114	-113	-108
452	-108	-111	-99	488	-113	-111	-107
453	-111	-114	-104	489	-112	-109	-106
454	-112	-116	-107	490	-111	-109	-105
455	-113	-117	-108	491	-111	-108	-105
456	-114	-118	-108	492	-110	-108	-104
457	-115	-119	-109	493	-107	-107	-104
458	-117	-120	-109	494	-103	-105	-101
459	-118	-123	-109	495	-98	-102	-98
460	-120	-125	-111	496	-95	-98	-93
461	-122	-127	-112	497	-91	-96	-90
462	-125	-128	-113	498	-88	-93	-86
463	-126	-127	-115	499	-86	-91	-84
464	-127	-125	-116	500	-84	-89	-82
465	-128	-124	-118	501	-83	-87	-80
466	-128	-123	-118	502	-81	-85	-79
467	-127	-123	-115	503	-80	-83	-79
468	-124	-124	-113	504	-80	-82	-79
469	-121	-124	-111	505	-82	-82	-82
470	-118	-124	-108	506	-84	-83	-86
471	-115	-124	-106	507	-86	-84	-90
472	-112	-122	-105	508	-86	-85	-93
473	-110	-121	-102	509	-85	-85	-98
474	-108	-118	-101	510	-85	-84	-104
475	-108	-116	-100	511	-85	-83	-114
476	-109	-115	-101	512	-85	-83	-128

进一步对 Matlab 和 FPGA 的仿真结果进行分析，如表 4.2，

表 4.2　仿真结果的统计分析

	原始数据	Matlab 处理结果	FPGA 处理结果
期望	-30.496	-31.264	-26.002
方差	6176.5	6103.9	6015.1

可以看到，整体结果上，Matlab 处理效果更好，FPGA 处理后的期望值偏离原始数据相对更大一点。经计算，Matlab 处理结果的最大绝对误差为 17，FPGA 处理结果的最大绝对误差为 43，发生在帧数据的尾部，是因为没有对数据进行延拓处理，边界效应导致的较大误差。

结　论

本文进行了脑电信号载波提取算法的硬件设计，主要进行的工作有，

1. 在 BCI 接口的系统架构层次，对提取算法进行了具体分析，按照软硬件协同设计的思想进行了提取算法的划分。充分利用 DSP+FPGA 系统架构的资源优势，实现了诱发脑电信号的载波提取算法的处理过程。
2. 在 FPGA 设计中，按照从上到下的设计思想，对算法功能进行了模块划分，并逐渐细化，确定了最终设计方案，使设计思路清晰，难度降低，工作量减少。
3. 按照设计方案，进行了 FPGA 的具体设计实现。在 EDA 环境中，以 VHDL 为开发语言，分模块进行了代码开发，并辅以多种输入形式，利用器件的各种资源，完成了所有模块的设计实现。
4. 对设计的代码，分模块进行了仿真验证，对仿真结果进行了分析，指出了各模块的性能、所占资源等状况。并与 Matlab 的仿真结果进行了比较。

从仿真结果分析，本文的设计实现了算法的基本功能，即信号提取及多尺度分析，达到了预期目的。创新在于用单片 FPGA 芯片实现了 AR 滤波器和多尺度小波分解与合成变换。所设计的 AR 滤波器是固定阶数但系数是可变的，具有一定的灵活性；小波分解与合成采用特定小波，滤波器系数是固定的，但变换级数是可以变化的，即可以实现多尺度分析，克服了目前普遍采用的固定系数、固定级数小波变换的设计方式的不足，增强实用灵活性。

但是本文的工作属于阶段性的，仍有许多有待完善改进的地方，还需要继续进行后续的研究。对下一步工作的设想是，

1. 在系统级完成仿真验证，即把所有子模块集成起来进行全局的仿真，进一步发现设计中的不足，完善设计。
2. 对功能进行改进、拓展，如增加数据接口带宽，从 8 位提高到 16 位，进一步提高处理数据的能力和性能；增加一定的存储模块和控制信号，使系统能够用作多通道信号的处理。
3. 和 DSP 结合进行联合测试，进一步发现问题，改进系统，使系统更趋实用化。

信号分析算法是 BCI 系统中的一个核心问题，如何高效、准确的实现载波的单次提取是脑科学研究的一个方向。新的算法不断出现，及时把新的算法转为硬件实现是重要而实用的工作，在科研和实际应用中都有很大的价值。

FPGA 技术的发展也是日新月异，速度更快、性能更好、资源更丰富的器件的

出现，为复杂的算法向硬件转化提供了物质平台。在生物医学信号处理、数字通信等领域，FPGA 作为一种重要的实现方式，必将发挥越来越大的作用。

参 考 文 献

[1] J. R. Wolpaw, et al, “Brain–Computer Interface for Communication and Contral,” Clinical Neurophysiology, 2002, 113: 767–791

[2] J. R. Wolpaw, et al, “Brain–Computer Interface Technology: A Review of the First International Meeting,” IEEE Trans. Rehab. Eng., 2000, 6(8): 166–173.

[3] Schalk, G, McFarland, D.J, Hinterberger, T, Birbaumer, N, Wolpaw, J.R. BCI2000: a general-purpose brain-computer interface (BCI) system. Biomedical Engineering, IEEE Transactions on, 2004, 51(6): 1034 – 1043

[4] 华小梅，林家瑞，官金安. “脑－机接口”的研究进展. 国外医学生物医学工程分册，2004，27(2)：94－97

[5] 杨坤德，田梦君，张海南等. 脑－计算机技术的研究进展. 生物医学工程杂志， 2004，21(6)：1024－1027

[6] M. Middendorf et al., “Brain–computer interfaces based on steady-state visual-evoked response,” IEEE Trans. Rehab. Eng., 2000,6(8): 211–214.

[7] N. Birbaumer et al., “The thought translation device (TTD) for completely paralyzed patients,” IEEE Trans. Rehab. Eng., 2000, 6(8): 190–193

[8] E. Donchin et al., “The mental prosthesis: Assessing the speed of a P300- based brain–computer interface,” IEEE Trans. Rehab. Eng., 2000, 6(8): 174–179

[9] P. R. Kennedy et al., “Direct control of a computer from the human central nervous system,” IEEE Trans. Rehab. Eng., 2000, 6(8): 198–202

[10] G. Pfurtscheller et al., “Current trends in Graz brain–computer interface (BCI) research,” IEEE Trans. Rehab. Eng., 2000, 6(8): 216–219

[11] J. R. Wolpaw et al., “Brain–computer interface research at the Wadsworth Center,” IEEE Trans. Rehab. Eng., 2000, 6(8): 222–226

[12] W. D. Penny et al., “EEG-based communication: A pattern recognition approach,” IEEE Trans. Rehab. Eng., 2000, 6(8): 214–215

[13] Fabiani, G.E, McFarland, D.J, Wolpaw, J.R., Pfurtscheller, G. Conversion of EEG activity into cursor movement by a brain-computer interface (BCI). Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 2004, 12(3): 331 – 338

[14] Xie Ying, Yang Zhongle. “Event-related Potentials during Imitated Natural Reading”. International Journal of Psychophysiology. 2002, 45 (1-2): 422

[15] Townsend, G, Graimann, B, Pfurtscheller, G. Continuous EEG classification

during motor imagery-simulation of an asynchronous BCI. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 2004, 12(2): 258 – 265

[16] Burke, D.P., Kelly, S.P, de Chazal, P., Reilly, R.B., Finucane, C. A parametric feature extraction and classification strategy for brain-computer interfacing. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 2005, 13(1): 12 – 17

[17] 何庆华，彭承琳，吴宝明等. 基于视觉诱发电位的脑机接口实验研究. 生物医学工程学杂志，2004，21(1)：93－96

[18] 何庆华，彭承琳，吴宝明. 脑机接口技术的研究方法. 重庆大学学报，2002，25(12)：106－109

[19] Amari S,Cichocki A. A Adaptive Blind Signal Processing- Neural Network Approaches[J], Proceedings of the IEEE, 1998,86(10): 2026－2049

[20] Fung KSM,Chan FHY,Lan FL,Poon PWF. A Tracing Evoked Potential Estimator[J], Signal Processing, 1994,36: 287－314

[21] 郭晓静，吴小培，张道信等. 基于独立分量分析的脑电消噪与特征提取. 系统仿真学报，2003，15(2)：287－289

[22] 张继武, 郑崇勋, 耿中行. 用奇异性检测技术提取诱发电位[J], 中国生物医学工程学报, 1999,18(3): 295－303

[23] 洪波, 唐庆玉, 杨福生, 潘应辐等. ICA 在视觉诱发电位的少次提取与波性分析中的应用[J], 中国生物医学工程学报, 2000,19(3): 334－341

[24] 杨福生, 洪波, 唐庆玉. 独立分量分析及其在生物医学工程中的应用[J], 国外生物医学工程分册, 2000,23(3): 129－134

[25] 朱常芳, 胡广书. 诱发电位快速提取算法的新进展[J], 国外医学生物医学工程分册, 2000,23(4): 211－216

[26] D. J. McFarland et al., “Design and operation of an EEG-based braincomputerinterface with digital signal processing technology,” Behav. Res. Meth. Inst. Comp., 1997, 29: 337–345

[27] Vladimir Bostanov, BCI Competiotion 2003----Data Sets Ib and IIb: Feature Extraction From Event-Related Brain Potentials with the Continuous Wavelet transform and the t-Value Scalogram. IEEE Trans.on Biomed. Eng. 2004,6(51)

[28] 李勇, 张圣训, 华蕴博. 小波分析理论在脑电分析中的应用[J], 中国生物医学工程学报, 1998,17(4): 320－324

[29] 杨毓英, 史习智, 韦蕾. 小波变换用于豚鼠脑干听觉诱发电位信号的特征提取[J], 中国生物医学工程学报, 1998,17(3): 226－231

[30] Agante, Marques de Sa. ECG noise filtering using wavelets with soft-thresholding methods. Computers in Cardiology 1999, IEEE. 1999,9(26): 535－538

[31] Zhang, J.-H., Janschek, K, Bohme, J.F., Zeng, Y.-J. Multi-resolution dyadic wavelet denoising approach for extraction of visual evoked potentials in the brain Vision Image and Signal Processing, IEE Proceedings, 2004,3(151): Volume: 180-186

[32] J.Britton, B.W.Jervis, R.A.Grunewald. Extracting single trial event related potentials. Measurement and Technology, IEE Proceedings, 2000, 147(6): 382-388

[33] Cuiwei L,Chongxun Z,Changfeng T. Detection of ECG Characteristic Points Using Wavelet Transforms[J], IEEE Transactions on Biomedical Engineering,1995, 42(1): 19-28

[34] Mallat S., Hwang M.W. Singularity Detection and Processing with Wavelets[J], IEEE Transactions on Information Theory, 1992,38(2): 617-642

[35] Mallat S,Zhong S. Characterization of Signals from Multiscale Edges[J], IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992,14(7): 710－732

[36] Cetin A,Ansari R. Signal Revovery from Wavelet Transform Maxima[J], IEEE Trans. Signal Processing, 1994,42(1): 27－37

[37] 陈德智, 唐磊, 盛剑霓, 王恒利. 由小波变换的模极大值快速重构信号[J], 电子学报, 1998,26(9): 82－85

[38] 王玉平, 蔡云龙. 多尺度 B 样条小波边缘检测算子[J], 中国科学（A 辑）, 1995,25(4): 426－437

[39] 徐丁峰，高小榕，高上凯等. 数字信号处理器在脑－机接口系统中的应用. 北京生物医学工程，2002，21(4)：256－259

[40] 李丽. 集成电路设计方法及 IP 设计技术的研究：合肥工业大学博士学位论文. 万方数据库，2002.11，10-12

[41] Parhi, K.K, Nishitani, T. Folded VLSI architectures for discrete wavelet transforms . Circuits and Systems, IEEE International Symposium, 1993, 3：1734 – 1737

[42] Hongyu Liao, Mandal, M.Kr, Cockburn, B.F. Efficient architectures for 1-D and 2-D lifting-based wavelet transforms. Signal Processing, IEEE Transactions on, 2004, 52(5)：1315 – 1326

[43] Cox, de Carvalho. Common architecture for discrete wavelet transform analysis and synthesis with sequential and constant processing elements. Instrumentation and Measurement Technology Conference, 2004. IMTC 04. Proceedings of the 21st IEEE ,2004,5(1): 45 – 50

[44] Walker, S.L.; Foo, S.Y.; Petrone, J. On the performance of a hardware implementation of the wavelet transform. System Theory, 2003. Proceedings of the 35th Southeastern Symposium, 2003, 3：397 - 399

[45] Uwe Meyer-Baese. 数字信号处理的 FPGA 实现. 刘凌，胡永生译. 第 1 版. 北

京：清华大学出版社，2003，24－36

[46] 马伍新，崔占忠，代方震. 用 FPGA 实现（5，3）小波变换. 计算机工程与设计，2003，24(1)：47－49

[47] 庞峰. 通用小波包变换处理器芯片的设计研究：上海大学硕士学位论文. 万方数据库，2003.2，32－35

[48] 杨咸春. 离散小波/小波包变换的 VLSI 结构及其在 FPGA 上的实现：天津大学硕士学位论文. 万方数据库，2001.1，17－19

致 谢

本文是在导师陈亚光教授的指导下完成的。在我攻读硕士学位的三年时间里，我在思想上、学业上和科研工作等诸多方面得到了陈老师的悉心指导和关怀，使我受益匪浅，为今后的工作打下了良好的基础。在此，我要向陈老师表示深深的感谢和敬意。

本文课题是国家自然科学基金资助的科研项目——脑机接口技术研究中的一个组成部分。感谢课题组中各位老师、同学给予我的指导和帮助，为我工作的顺利展开、论文的撰写提供了诸多便利。

感谢研究生学习期间给过我帮助的各位同学。

感谢我的家人对我精神和物质上的鼓励和支持。

附录 A　攻读学位期间所发表的学术论文目录

[1] 熊新兵，焦晓军，陈亚光. 用提升小波变换提取诱发脑电. 中南民族大学学报，2004，23（3）：34-37

[2] 焦晓军，陈亚光. 手术体位固定气囊的自动压控系统设计. 中南民族大学学报，2004，23（4）：37-40

[3] 焦晓军，陈亚光. 诱发脑电信号的基于 RLS 算法的自适应滤波处理及仿真. 科技创业月刊，2004，11：154-155

附录 B　Stratix 器件内部资源表

内部资源	EP1S10	EP1S20	EP1S30	EP1S40	EP1S60	EP1S80	EP1S120
逻辑单元	10,570	18,460	32,470	41,250	57,120	79,040	114,140
M512 RAM 模块	94	194	295	384	574	767	1,118
M4K RAM 模块	60	82	171	183	292	364	520
Mega RAM 模块	1	2	4	4	6	9	12
RAM 总量 (bit)	920,448	1,269,248	3,317,184	3,423,744	5,215,104	7,942,520	10,118,016
DSP 模块	6	10	12	14	18	22	28
嵌入式乘法器	48	80	96	112	144	176	224
锁相环	6	6	10	12	12	12	12
最大可用引脚	422	582	726	818	1,018	1,234	1,310