摘 要

离散余弦变换是广泛应用于信号处理、图像处理领域的重要工具之一，已经被多个国际标准所接受，如 JPEG、MPEG、 H.263 等。DCT 应用到实际系统中的前提是具有能够快速实现的算法，自从 1977 年第一个真正的 DCT 快速算法出现以来，寻求更快、更规则、更简单的 DCT 快速算法一直是信号处理领域的一个热点研究方向。

作为正交变换的 DCT 算法具有计算复杂度适中、快速算法等特点，在数字信号处理等方面有着广泛的应用。由于应用领域的不同，也出现了很多形式的 DCT 算法。

本论文主要研究的是二维离散余弦变换的快速算法设计，其研究的重点是实现二维 DCT 变换的直接分解算法。论文工作分为三个部分：

一. 对 DCT 算法进行总结，概括和描述了近几年出现的 DCT 算法并进行分类。

二. 在研究已有 DCT 算法的基础上提出了一种改进的快速算法－部分和分解算法(PSDA 算法)；在 PSDA 算法中，通过引入部分和的定义，实现了频域输出数据的分集；通过频域输出数据的分集将二维 DCT 变换转换为若干个一维 DCT 变换；PSDA 算法还给出了部分和的计算方法及组成部分和的公共加法项的合并原则。

三. 从 VLSI 实现的角度出发，提出了部分和分解算法的 VLSI 结构，基于该结构的 RTL 代码通过了综合和 FPGA 原型验证，为该算法的后续研究和应用提供了理论基础和实现参考。论文最后给出了 PSDA 算法的 VLSI 实现结果；指出了 PSDA 算法未来的应用前景和发展方向。

相比其它 DCT 算法，本文提出的 PSDA 算法实现了以下几个方面的创新和改进。

1. 提出了部分和相等的分集准则，通过该准则实现了对频域输出数据的分集，并证明了每一个分集内的元素为时域输入数据部分和的一维 DCT 变换输出。

2. 通过对频域数据进行子集划分，将一个二维 DCT 变换转换为若干个一维 DCT 变换，完成 DCT 变换的乘法运算次数减少了一半。

3. 提出了部分和的计算方法和公共加法项的合并原则，通过合并公共加法项减少了加法运算量。

4. 传统的 DCT 算法是针对输入数据长度为 2^n 进行计算的，但是在很多应用领域中都要用到长度非 2^n 的 DCT 算法，而素长度的 DCT 算法是非 2^n 长度 DCT 的核心。因此本文在 2^n 长度 PSDA 算法的基础上提出了改进的二维素长度 PSDA DCT 算法，该算法是基于频域输出数据与部分和的转换和映射关系，将二维素长度 DCT 变换分解为多个一维素长度 DCT 变换。同已有的素长度 DCT 算法相比，减少了一半的乘法计算量。

5. 提出了基于 2^n 长度 PSDA 算法的 VLSI 结构，同间接算法的 VLSI 结构相比，该结构具有不需要转置变换，处理延时低的优点；同其他直接算法的 VLSI 结构相比，该结构具有更规则的结构，和更少的乘法器和加法器开销。

6. 目前的 DCT IP 核都是针对 2^n 长度的，并不能满足完全实际应用中的需要。本文在 PSDA 算法的基础上，分别基于 FPGA 和 ASIC 工艺进行了素数长度 DCT 的 IP 核设计。仿真和综合结果表明，该设计结构简单、层次清晰，具有高度的规则性和模块性。

关键词：离散余弦变换， 快速算法， 部分和分解算法(PSDA)， VLSI 实现

ABSTRACT

The discrete cosine transform is one of the important tools in signal processing and image/video processing. It is now accepted by several international standards, such as JPEG, MPEG, H.263, etc. The precondition for using DCT in practical system is the algorithms for fast implementation of DCT. Since the first true DCT fast algorithm is proposed in 1977, looking for the faster, more structured and simpler algorithm for DCT is one of hot research topics in signal processing fields.

As an Orthogonal transform, DCT has many fast computation algorithms. In the different application fileds, there are different types of DCT algorithms.

The thesis deals with the fast algorithms of 2D－DCT and their VLSI implementation structures suited for implementing in hardware and parallel processing. And the thesis emphasizes on the research of direct－decomposition algorithm for 2D－DCT. It contains three parts:

1. The thesis summarizes the fast algorithms for DCT and classes them into several types.

2. The thesis presents a new fast 2－D DCT algorithm(PSDA). By the definition of partial sum, an allocation algorithm is designed on the frequency output data. By the computation of partial sum, the 2D－DCT is converted to several 1D－DCT, and the partial sum decomposition algorithm utilizes only half multipliers and discards transposition memory comparing to RC algorithm.

3. The thesis presents the corresponding VLSI structure of partial sum decomposition algorithm (PSDA) and points out the applition field of PSDA in the future.

Compared to other DCT algorithms, such innovation is archived in the following:

1. The thesis presents the definition of partial sum, researchs the sub－set allocation algorithm based on partial sum and the computation of partial sum.

2. Via the sub-set allocation based on partial sum, the original Two-Dimensional DCT transform is coverted into several One-Dimensional DCT transforms.

3 The theis researchs the property of partial sum, deduce and present out a new

algorithm for computing $2^n \times 2^n$ type 2D－DCT based on partial sum sharing method. Illustrate how to decrease add operations times during partial sum computation. The algorithm costs fewer multipling and adding times than other known 2D－DCT algorithms .

4 Research a new algorithm for computing prime number sized 2D－DCT based on partial sum decomposition and convert the original two dimensional computation to several 1D prime number size DCT computations, and deduce out the computation complexity of the algorithm. And the algorithm gets the archievement of about half multiplying times comparing to RC algorithm.

5 Research the $2^n \times 2^n$ type 2D－DCT VLSI structure based on PSDA. Comparing to the in-direct VLSI structures, it doesn't use transposition RAM and archieves a lower processing delay. Comparing to the direct VLSI structures, it has a more regular structure and fewer multiplier and adder cost.

6 Research the $q \times q$ type 2D－DCT VLSI structure based on PSDA(q is a prime number). Comparing to the in-direct VLSI structures, it doesn't use transposition RAM and archieves a lower processing delay.

Key Words: discrete cosine transform(DCT), partial sum decomposition algorithm(PSDA), fast algorithm, VLSI Implementation

图目录

表目录

缩略词表

ASIC：	**Application Specific Integrated Circuit**
DA：	**Distributed Arithmetic**
DCT：	**Discrete Cosine Transform**
DFT：	**Discrete Fourier Transform**
FFT：	**Fast Fourier Transform**
PSDA：	**Partial Sum Decomposition Algorithm**
RCM：	**Row－Column Method**
NRCM：	**Non Row－Column Method**
VLSI：	**Very Large Scale Integration**
1D－DCT：	**One－Dimensional Discrete Cosine Transform**
2D－DCT：	**Two－Dimensional Discrete Cosine Transform**

独 创 性 声 明

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

签名：田戎　　日期：2008 年 12 月 16 日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名：田戎　　导师签名：

日期：2008 年 12 月 16 日

第一章 绪论

1.1 技术背景和研究意义

数字图像处理(Digital Image Processing)又称为计算机图像处理，它是指将图像信号转换成数字信号并利用计算机对其进行处理的过程。数字图像处理最早出现于 20 世纪 50 年代，当时的电子计算机已经发展到一定水平，人们开始利用计算机来处理图形和图像信息。数字图像处理作为一门学科大约形成于 20 世纪 60 年代初期。早期的图像处理的目的是改善图像的质量，它以人为对象，以改善人的视觉效果为目的。图像处理中，输入的是质量低的图像，输出的是改善质量后的图像，常用的图像处理方法有图像增强、复原、编码、压缩等。图像处理技术在许多应用领域受到广泛重视并取得了重大的开拓性成就，属于这些领域的有航空航天、生物医学工程、工业检测、机器人视觉、公安司法、军事制导、文化艺术等，使图像处理成为一门引人注目、前景远大的新型学科。 随着图像处理技术的深入发展，从 70 年代中期开始，随着计算机技术和人工智能、思维科学研究的迅速发展，数字图像处理向更高、更深层次发展。人们已开始研究如何用计算机系统解释图像，实现类似人类视觉系统理解外部世界，这被称为图像理解或计算机视觉。70 年代末 MIT 的 Marr 提出的视觉计算理论，这个理论成为计算机视觉领域其后十多年的主导思想。图像理解虽然在理论方法研究上已取得不小的进展，但它本身是一个比较难的研究领域，存在不少困难，因人类本身对自己的视觉过程还了解甚少，因此计算机视觉是一个有待人们进一步探索的新领域。

数字图像处理的信息大多是二维信息，处理信息量很大。如一幅 256×256 低分辨率黑白图像，要求大约 64kbit 的数据量；对高分辨率彩色 512×512 图像，则要求 768kbit 数据量；如果要处理 30 帧/秒的电视图像序列，则每秒要求 1Mbit～22.5Mbit 数据量。因此对计算机的计算速度、存储容量等要求较高。

21 世纪的人类社会已经进入了信息化时代。数字化后的信息，尤其是数字化后的视频和音频信息具有数据海量性，它给信息的存储和传输造成较大的困难，成为阻碍人类有效获取和使用信息的瓶颈问题之一。因此，研究和开发有效的多媒体数据压缩编码方法，以压缩的形式存储和传输这些数据是非常有必要的。

视频数据压缩目前的主要目标是追求较大的压缩率、较快的压缩解压缩速度

以及尽可能好的图像重构质量，同时也在向压缩数据的处理如数据组织、检索、重构等方向发展，力求发展一个比较完整的图像压缩处理解决方案，因此在这方面仍有许多的工作要做，其中最基本的就是要有比较合理且高效的压缩算法。

1.1.1 图像压缩算法基本框架

标准化是产业化活动成功的前提，标准的制定保证了数据流可以在不同的终端和应用间交换。自国际无线电咨询委员会(CCIR: International Radio Consultative Committee)于 1982 年通过了电视演播室数字编码的国际标准(CCIR 601 号建议)以来，视频编码技术日趋成熟，目前的国际视频编码标准已有很多种。如国际标准化组织(ISO: International Standardization Organization)和国际电子学委员会(IEC: International Electronics Committee)下属的活动图像专家组(MPEG: Moving Pictures Experts Group)组织制定的 MEPG－1，MPEG－2， MPEG－4 标准，国际电信联盟(ITU: International Telecommunication Union)制定的 H.261 和 H.263 标准，以及 ISO/IEC 下属的 MPEG 和 ITU 下属的视频编码专家组(VCEG: Video Coding Experts Group)共同成立的联合视频小组 JVT(Joint Video Team)最新完成的 H.264 标准也称为 MPEG－4 AVC(Advanced Video Coding)，以及国内第一个针对音视频产业需求制定的标准 AVS(Audio Video Standard)等都是基于混合编码(Hybrid Coding)框架之上的。这里的所谓混合编码框架是指综合运用预测，变换以及熵编码的编码框架，有以下三个主要特点:

1、利用帧间预测消除图像间的冗余，利用帧内预测消除图像内的冗余。

2、通过对预测残差进行变换和量化来消除图像内的视觉冗余。

3、利用熵编码来消除统计上的冗余。

在实际应用中，编码端首先通过帧内预测或帧间预测，得到相应的残差数据块，对原始图像数据块或残差数据块进行二维变换，然后在变换域中对变换系数进行量化，最后进行熵编码，即采用变长编码或者算术编码等。而在解码端，则对残差的变换量化系数进行反量化反变换，然后和预测值相加即可得到重建图像块。

图 1－1 是图像和视频压缩系统的结构框图。由图可知，变换是系统中的一个核心部分。

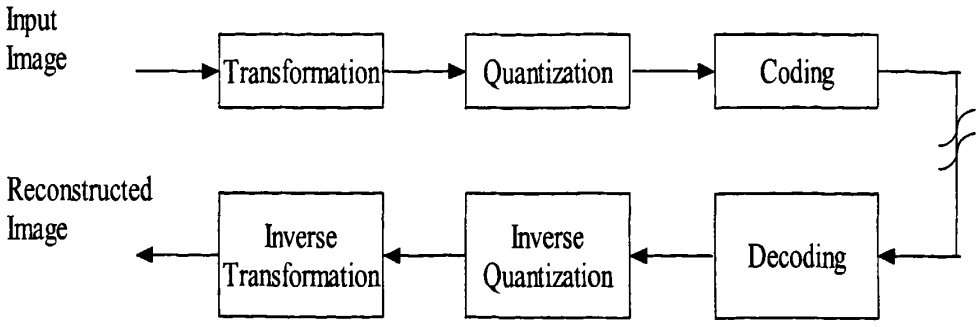


图 1－1 图像和视频压缩系统

1.1.2 离散余弦变换

研究表明，视频数据、图像数据或其残差数据在空间域上仍有着较强的相关性。进行变换的目的就是为了将这些在空间域上有较强相关性的数据变换到以变换矩阵的归一化向量表达的变换域上。在变换域上，这些信号的变换域表示的相关性就很弱，有利于进一步进行压缩处理。因此，变换是视频编码和图像编码中非常重要的部分。当今流行的视频编码标准中大多采用了离散余弦变换作为它们的核心变换。

离散余弦变换是由离散傅立叶变换(DFT)发展过来的，由于离散傅立叶变换在数字信号处理频谱分析等领域中已成为有力的数学工具，但其缺点是复数域运算，运算量太大以至难以实时处理，为克服这些缺点，1974 年，Ahmed 等人构造了一种实数域变换－DCT[1]。由于该变换性能很适用于人类语言及图像信号的特点，从这个意义上，DCT 常常被认为是接近 K－L 变换性能的次佳变换。

DCT 有一个重要性质，就是它的变换矩阵的基向量很近似于 Toeplitz 矩阵(即沿对角线方向的元素都相同)的特征向量。这就是说，对 Toeplitz 矩阵 DCT 将非常接近 K－L 变换。统计表明，人类的语言、图像等信号的自相关矩阵常常表现出具有 Toeplitz 矩阵的特点。从这个意义上来说，DCT 是比较适合对语言、图像等信号作变换处理的一种变换，它的性能接近于 K－L 变换。

DCT 还具有以下性质：

1)对于具有高相关性的数据(信号)，DCT 具有非常好的能量聚焦性，经过变换，信号能量的绝大部分被集中到变换域的少数系数上；

2)DCT 具有可实现的快速算法；

基于以上优点，离散余弦变换被广泛应用于 JPEG 压缩/解压缩、DVD/VCD 播放机、电缆电视、HDTV、图形与图像处理卡、超声波/核磁共振成像系统、数字

录像机、数字照相机、视频电话与会议系统、图像传输系统等。目前 DCT/IDCT 已成为运动图像压缩标准(MPEG)和静止图像压缩标准(JPEG)等的重要组成部分。

实际的应用促进着快速算法的发展，在这之后，针对各个应用领域，不同形式的 DCT 算法也应运而生。DCT 变换也出现了多种扩展形式，如音频压缩中使用的修正离散余弦变换(MDCT)，H264 标准中使用的整型 DCT 变换等。

在音频编码中，编码器通常将 PCM 数据分成若干个等宽的子带等待心理声学模型的判断，做进一步的量化压缩，这种压缩法我们叫做 sub－band coding。这种方式由于是等宽的频带滤波，不符合人耳的听觉特性的，对后续量化阶段的处理不利，相邻子带间也容易发生混叠效应。为了克服以上缺点，音频编码器使用 MDCT 转换，进一步将划分成更细的频带，提高对频率的解析度。

离散余弦变换并不是整数可逆的，因此，在利用有限计算精度的计算机来实现 DCT 变换域编码时，只能是有损的编码。虽然这对于很多应用是可以允许的，但还有一些应用却不允许有任何的图像失真，比如医学图像和遥感图像。此外，一些应用还要求有损和无损编码同时兼顾，即算法要统一。而对于这些应用，就需要实现可逆的整数 DCT 变换。

进行整数 DCT 变换的方法有很多，有代表性的有基于矩阵分解的分裂基 DCT 算法和基于提升结构的算法。但是最根本的原理是利用了提升结构的特点，使逆变换得到的值与原始值没有差别。

提升结构不仅能实现灵活的正交变换，而且还能达到无损压缩效果，因此在变换编码中是一个非常有用的工具。1999 年后发展起来的二进制 DCT 算法利用提升结构消去了乘法，只用移位和加法来实现，得到基于提升结构的无乘法 DCT。整数 DCT 算法的另一个优点是若有足够的字长来表示数据的话，那么中间的误差可以完全消除，实现信息在传输过程的无损。

综上所述，离散余弦变换是音视频编解码的核心变换。因此研究 DCT/IDCT 算法及其 VLSI 实现的地位日趋重要，尤其是高性能低功耗的算法及其 VLSI 实现结构。

1.1.3 ASIC 技术概述

专用集成电路(ASIC：Application Specific Integrated Circuit)是一种为专门目的而设计的集成电路，它与标准集成电路或通用集成电路有着很大的区别。从 1948 年第一只半导体晶体管问世以后，微电子技术就得到了日新月异的发展，50 年代

中期可用于工业上的产品以锗材料制作的结型半导体晶体管为多，由于锗材料比硅材料具有较高的载流子迁移率，因而较易于实现高频性能。但锗材料制作的半导体器件有着固有的缺点。50 年代末出现的硅平面制造工艺技术，不但成为硅半导体晶体管的基本制造工艺，也成为将多个分立器件制作在同一块面积不大的硅片上的集成电路的基本制造工艺。硅器件有效地克服了锗器件所存在的缺点，这种技术也一直沿用到今天。

1.1.3.1 ASIC 电路的分类

ASIC 电路可分为以下三个类别。

1)全定制集成电路

全定制集成电路指设计和制作(所有的逻辑单元和掩膜版)都是按定制的方式进行的集成电路。设计者需要对电路版图进行最精细的制作，以免浪费芯片上任何一个平方微米的空间，由于这些专门设计的 IC 产量高，适用于通用的应用目的，故称它们为全定制通用集成电路(full－custom IC)。

2)半定制集成电路

半定制集成电路指所有的逻辑单元预先进行设计，但其中一些或所有的掩膜版按照定制方式进行制作的集成电路。使用单元库中预先设计好的单元可以大大的简化设计。半定制集成电路可分为标准单元集成电路和门阵列集成电路。

3)可编程 ASIC 电路

可编程 ASIC 指所有的逻辑单元都预先进行设计，但没有一块掩膜版是按定制方式进行制作的集成电路。它主要分为两类：可编程逻辑器件(PLD)和现场可编程门阵列(FPGA)。

1.1.3.2 基于标准单元的 ASIC 设计流程

对于 IP 核的 ASIC 验证，大致的流程如下：

1)设计输入，采用已有的 Verilog HDL 代码即可。

2)逻辑综合，使用逻辑综合工具将设计输入的 Verilog HDL 代码转化成门级网表，通过网表来描述逻辑单元间的连接关系。

3)系统划分，如果有必要，将大系统划分为几个 ASIC。

4)布图前仿真，检查设计的功能是否正确。

5)布图规划，在芯片上排列网表模块。

6)布局，决定模块中单元的位置。

7)布线，单元与模块之间的连接。

8)提取参数，确定真实的互连电阻和电容。

9)布图后仿真，加上互连线负载后检查设计是否能够正常工作。

一般来说，步骤 1－4 为逻辑设计部分，5－9 为物理设计部分。随着工艺的不断提升，现代的设计方法逐渐倾向于前后端设计的融合。

1.1.3.3 ASIC 的单元库

单元库是 ASIC 设计的关键部分，对于可编程 ASIC 而言，FPGA 公司以成套设计工具形式提供逻辑单元库，通常用户没有其他选择。而对于标准单元 ASIC 设计来说，可有 3 种选择：ASIC 供应商(开发 ASIC 的公司)提供单元库；从第三方单元库供应商处购买单元库；建立自己的单元库。

第一种选择采用 ASIC 供应商单元库，要求用一套 ASIC 供应商认可的设计工具输入并进行仿真设计。出于某些原因，在日本，一般的模式是采用 ASIC 供应商提供的工具，而在美国，欧洲以及其它地方的设计者情愿选择自己的工具。

ASIC 供应商的单元库通常是虚库－单元都是空框或者说是虚的，但含有足够的版图信息。完成版图后，将网表传递给 ASIC 供应商，在芯片制造前将其填入空框内。

第 2 种和第 3 种选择是由用户决定购买还是开发。如果购买单元库来完成 ASIC 设计，用户就拥有制造 ASIC 的掩膜板(加工模具)，这也称为客户控制的加工模具。

库供应商利用 ASIC 芯片加工厂提供的工艺信息开发单元库，ASIC 芯片加工厂(不同于 ASIC 供应商)仅是制造，不提供设计帮助。如果单元库满足芯片加工厂的加工规格，则称其为合格单元库。这些单元库一般都很贵(可能几十万美元)，但如果单元库符合几个加工厂的规格，就可以选择最合适的加工厂。这表明，对大批量生产，购买较贵的库最终费用比其他解决方案低。

第 3 种选择是自行开发单元库，很多大的计算机和电子公司会做此选择。尽管单元库开发过程既复杂又花钱，但是如今已设计的单元库多数是自行开发的。

不管哪种方法，ASIC 单元库的每个单元必须包括以下内容：

1)物理版图

2)行为级模型

3)Verilog/VHDL 模型

4)详细时序模型

5)测试策略

6)电路原理图

7)单元图符

8)连线－负载模型

9)布线模型

以上列出的某些方面，像版图，图符等等的库单元内容是显而易见必须有的，但对于行为级模型等内容，这里要做一个简单的介绍。

行为级模型是指对单元电路做的一种高层次的描述，这是因为用户在对一个定态的 ASIC 系统作详细的时序分析时需要花费大量的时间，为了节约时间，在电子系统分析的初期采用行为模型可以大大缩短仿真时间。

ASIC 电路设计者为了掌握电路关键路径的时序性能，就需要对每一个库单元有各自对应的时序模型。一般库设计人员是通过对单元电路所做的参数提取来仿真库单元电路的延迟时间。

为了在实际布线完成前估算出引线的寄生电容，就需要对给定大小的电路模块中线网的电容进行统计估算。这常常采取查表的方式，也称为连线－负载模型。这里还需要每个单元的布线模型。直接用物理设计或版图工具处理大单元太复杂，此时需要比较简单的物理版图表示法－虚库法，但它仍需包含所有必要的信息。虚库可能包含的信息为：告诉自动布线工具能在单元上何处布线，以及连接到单元的位置和类型。

1.1.4 超大规模集成电路(VLSI)

自从 1958 年美国德州仪器公司发明集成电路以来，特别是在最近的 20 年，集成电路技术的发展，已成为当代科技界最引人注目的焦点之一。在当今世界上，无论是科学技术、军事、经济，还是人们的日常生活，都早已和它密切结合，息息相关。

集成电路由于电路复杂程度不同，可根据集成规模分为小规模集成电路、中规模集成电路、大规模集成电路和超大规模集成电路。对于数字集成电路来说，习惯上认为小规模集成电路是集成度小于 10 个门电路或集成元件数少于 100 个元件的集成电路；中规模集成电路是集成度在 0～100 个门电路之间，或集成元件数在 100～1000 个元件之间的集成电路；大规模集成电路是集成度在 100 个门电路

以上或集成元件数在 1000 个元件以上的集成电路。大规模集成电路是在一般中、小规模集成电路的基础上发展起来的，中、小规模集成电路一般是以简单的门电路或单级放大器为集成对象的，而大规模集成电路则以功能部件、整机、子系统为集成对象。从分立元件发展到集成电路是半导体电子技术发展的一次飞跃；从一般中、小规模集成电路发展到大规模集成电路是又一次飞跃，并且还在向超大规模集成电路发展。超大规模集成电路一般指集成度达 1 万个门电路或集成元件数在 10 万个元件以上的大规模集成电路。

随着超大规模集成电路(VLSI)和专用集成电路(ASIC)的广泛应用，集成电路设计已经不再是集成电路行业单独包揽所能胜任的了。由于芯片内部电路的规模越来越大，线路越来越复杂，集成电路专家很难应付各种用户的要求。ASIC 技术的特点是提倡用户自行设计 IC。自行设计 ASIC 是电子、信息业发展的方向。目前许多集成电路工厂向用户开放，接受用户自行设计的芯片版图进行专用芯片的加工。电子专家只要掌握了 VLSI 设计技术，就可以设计自己的专用芯片，将原来安装在一块印刷电路板上的电路，集成在一个芯片内。从而大大缩小整机的体积，降低功耗，提高性能，增加可靠性和保密性，节省装配成本。现代电子产品的更新换代，很重要的一条，就是要依靠 VLSI 技术，要求电子产品设计师自己设计 ASIC。

一个复杂数字系统往往由许多功能模块构成，而设计者的新思想往往只体现于部分单元之中，其它单元的功能则是通用的，如 FFT，FIR，IIR，Viterbi 译码，PCI 总线接口，调制解调，信道均衡等。这些通用单元具有可重用性，适用于不同的系统。如果预先设计好这些通用单元并根据各种工艺对布局和布线进行优化，从而构成具有自主知识产权的功能模块，称之为 IP(Intellectual Property)模块，也可称为 IP 核(IP Core)。那么，相应的，针对不同的数字图像系统，就必须采用不同的处理方法和算法才能加以解决，不可能研制一个图像处理 ASIC 芯片就可以解决所有的信息处理问题。但是图像处理的许多算法都建立在一些基本运算之上，如加法，乘法，离散卷积，矢量内积，矩阵相乘以及 FFT 变换，DCT 变换和图像矩阵计算等。而它们在图像分析与处理以及图像压缩编码中有着广泛的应用。所以，开发人员可以设计出图像处理基本运算和算法的 VLSI 宏单元电路，建立具有自主知识产权的基本运算和算法芯片核(简称基本运算和算法 IP 核)库，可以方便的构建专门应用的图像处理算法芯片。因此，本文的大背景就是开展图像处理核心变换算法研究和 IP 核设计验证技术研究。充分考虑基本运算的内在并行性，研制可复用图像处理基本运算与算法 IP 核，为嵌入式实时图像处理系统芯片的研制

提供关键技术。针对实时图像数据，该算法的 IP 核可以与国内自行设计 ASIC 或 RISC CPU 核结合在一起，组成极具特色的高效图像处理系统芯片，满足实时图像处理的需要。开展基本运算可复用 IP 核的设计工作，对于提高图像处理系统芯片的设计效率具有积极意义。同时这些 IP 核也适用于通信，家电，工业检测，医学诊断等图像处理应用领域，具有巨大的经济价值。

当前世界半导体市场增长最显著的领域是 IP 核。在国外，目前自主开发和经营 IP 核的公司主要有英国的 ARM，Amphion，美国 DeSoc，Rambus 等。以 ARM 公司为例，在 1985 年 ARM 公司设计开发出第一块拥有自主知识产权的 RISC 处理器模块，1990 年首次将其 IP 专利权转让给 Apple 公司。到 2003 年全球已有 IBM，TI，Philips，NEC，Sony 等几十家公司采用其 IP 开发自己的产品。有关 IP 核设计的报道首次出现在 1997 年召开的 CICC(专用 IC 国际年会)的“单元建库”论文分册上。1998 年在美国加州的硅谷召开国际年会“半导体战略论坛”上，以 IP 产业的现状和发展为大会专题，共有 48 个全球著名的微电子公司的主要负责人作了大会发言。同年，“半导体战略论坛 98”组织了 IP 专题研讨会。在 1999 年“ASIC Status 99”的国际年会上论文总量的三分之一是围绕 IP 核的设计开发的文章。这些关于 IP 设计的国际年会从另一个角度展示了 IP 产业迅猛的发展势头。

目前国内总体来说在 IP 的开发和应用方面还处于初级阶段。令人可喜的是，近年来国家在 IP 产业上也有了很大的动作。科技部于 2000 年启动了“十五”国家 863 计划超大规模集成电路 SOC 专项工作。目前，我国已初步建成起具有自主知识产权，品种较为齐全和管理科学的国家级 IP 核库；并掌握国际水平的 SOC 软硬件协同设计，IP 核复用和超深亚微米集成电路设计的关键技术。由摩托罗拉向中国释放 M－Core 而触发的产业界 SOC－IP 讨论实质上是我国 IP 产业的启动。我国 IP 产业正在从概念阶段向实用阶段过渡。最近几年，数字电视和高清电视核心芯片、多功能和智能手机相关芯片、MP3、闪存和视频播放等新概念电子产品 SOC 以及 CPU 和 DSP 等基础 IC 成为中国 IC 公司开发的热点。而图像处理 IP 核既可作为数字电视和高清电视专用芯片的核心，又可与通用 CPU，DSP 结合作为图像处理系统解决方案，有着良好的通用性和广泛的应用。正因为有着上述优点，图像处理 IP 核的研究与设计成为近几年产业研究的热点。

1.1.5 离散余弦变换及其 VLSI 结构的发展现状

1974 年 Ahmel 等首先提出了 DCT 概念[1]，随后 DCT 在数字信号处理的各个

领域迅速得到应用，应用的需要激发了人们对 DCT 快速算法的研究，由于 FFT 的优异性能，最初的 DCT 快速算法基本上都是建立在 FFT 基础上的。但是研究表明，直接设计的 DCT 快速算法将具有更高的效率，1977 年，W.H.Chen，C.H.Smith，S.C.Fralick 利用对 DCT 变换矩阵的直接分解，第一次提出不通过 FFT 实现的 FDCT 算法[2]，至此以后出现了多种 DCT 快速算法。其中，最多也是应用得最普遍的是长度为 2^n 的算法。由于应用中对 DCT 长度的要求，针对非 2^n 长度 DCT 的算法研究受到人们的重视，实时处理应用要求以更快的速度实现 DCT 变换，通过数字滤波器以及硬件实现 DCT 的途径也越来越受到人们的注意。最初的 DCT 算法是基于 FFT 的，Haralick[3]用 N 点 FFT 构造了第一个基于 FFT 的 N 点 DCT 算法，随后 Narashima 和 Peterson[4]、Tsang 和 Miler[5]、Makhoul[6]、Vetterli 和 Nussbaomer[7]也提出基于 FFT 的 DCT 算法。第一个直接设计的 FDCT 算法是由 Chen 等根据矩阵分解方法提出的，计算 8 点 DCT 只需要 16 次乘法，比基于 FFT 的 DCT 算法减少了 1/6，由于矩阵分解的方法不是唯一的，因此，存在类似的其它矩阵分解算法。Lee[8]提出了一种以 cos 函数的倒数作乘法因子的快速算法，但当变换长度较大时将出现很大的乘法因子，使运算过程中产生的计算误差变大。Hou[9]提出了一种与 Lee 算法类似的算法，但使用 cos 函数作为乘法因子，避免了 Lee 算法中的计算误差大的问题。Hou 和 Lee 算法具有相同的计算复杂性，计算 8 点 DCT 都只需要 12 次乘法，达到了最少的乘法次数。Chan 和 Ho[10]、Wu 和 Paoloni[11]将 Hou 算法推广到二维 DCT。Arguello 和 Zapata[12]、Britanak[13][14]，提出了修正的 Hou 算法，减少了算法中的移位操作量，根据应用中并非所有 DCT 系数都需要的特点，Skodras[15]提出了计算 DCT 低频系数的截断算法。应用中往往需要非 2^n 长度的 DCT 快速算法，Yang[16]等提出了第一个素因子 DCT 实现算法并将算法应用到硬件实现中[17]，然而，他们的算法要求比较复杂的下标映射。Wang 和 Yip[18]推广了 Yang 的算法，提出了一组计算离散三角变换(DTT)的素因子快速算法。随后，Lee[19]也修改了 Yang 算法，用查表的方法通过两个映射表来实现下标映射。针对快速变换算法的硬件实现，Chakrabarti 和 Jaja[20]提出了计算素因子 DCT 和离散哈脱莱变换(DHT)的 Systolic 结构，其下标映射类似 Lee 算法。Lee 和 Huang[21]修正了 Lee 算法，提出了另一种素因子 DCT 算法，输入下标映射采用 Ruitanian 映射，而输出下标映射则采用 Lee 算法的映射方法。不同于素因子算法，Heideman[22]提出一种将奇长度 DCT 转化为同长度 DFT 的算法。Chen 和 Siu[23]利用数论方法，提出了一种将 DCT 转化为循环卷积的快速算法，并建议采用分布结构实现卷积[24]。Guo 等[25]修正了 Chen 和 Siu 算法，提出了素长度 DCT 的 Systolic 阵列算法。

在各种一维 DCT 快速算法不断提出的同时，为了适应多维信号处理的需要，二维及多维 DCT 算法的研究也受到人们的重视，Kamangar 和 Rao[26]提出了一种将二维 DCT 转化为一维变换的算法，Haque 从变换矩阵分解的角度出发，修改了 Kamangar 和 Rao 的算法[27]，Nasrabadi 和 King[28]、Vetterli[29]采用类似一维的方法，提出将二维 DCT 转换为二维 DFT 的实现算法。Cho 等[30-32]通过输入输出下标映射的方法，将 N×N 二维 DCT 用 N 个 N 点一维 DCT 实现，乘法复杂性降低到行列法计算的一半；Huang 和 Wu[33]用类似的方法给出另一种将 N×N 二维 DCT 用 N 个 N 点一维 DCT 实现的算法。

除了上面所述的各种 DCT 算法，还有多种将 DCT 转换为其它变换进行计算的方法。Duhamel 和 Guillemot[34]利用快速多项式变换算法提出了另一种计算二维 DCT 的方法。Ta 等[35]提出用快速离散 Radon 变换实现二维 DCT。

在研究各种 DCT 快速算法的同时，为了适应实时信号处理的要求，获得更快的变换速度，人们也在寻找采用硬件或并行处理的 DCT 实现方案[36-44]。

1.2 本文的创新点

本文主要对离散余弦变换的快速算法设计及其 VLSI 结构进行了研究，研究内容包括一维 DCT 快速变换算法，二维以及高维 DCT 快速变换算法，DCT 快速算法实现的 VLSI 结构研究等几个方面。本文的主要创新点简述如下：

1. 提出了一种计算二维 DCT 的快速算法－部分和分解算法(PSDA)。利用积化和差的性质，该算法将变换形式中的两个余弦因子相乘的形式转换为两个余弦因子相加的形式。在时域上将余弦因子相同的数据进行累加求得部分和。根据部分和相同的准则，将频域上的数据进行子集划分，同一集合内的频域数据为其相关部分和的一维 DCT 变换输出。

2. 研究了 PSDA 算法在 $2^n \times 2^n$ 长度二维 DCT 中的应用，提出了变换输出的频域子集划分理论，部分和的计算方法及公共加法项的合并原则。与目前已知的高效直接算法(Cho 算法和森川良孝算法)相比，PSDA 算法有着更少的乘法次数和加法次数。

3. 研究了 PSDA 算法在 $q \times q$(q 为素数)长度二维 DCT－II 变换中的应用。PSDA 算法将二维数据变换转换为若干个 N 点一维素数长度 DCT 奇系数或偶系数的计算；进而将奇系数或偶系数的变换转化为循环卷积或扭循环卷积的形式，解决了当前常用高效算法只适用于长度为 2^n 的情况而不能应用于素数长度的问题。

4. 研究了 PSDA 算法在 $2^n \times 2^n$ 长度二维 DCT－II 变换中的 VLSI 实现结构。一般一个硬件算法结构的优劣主要决定于三个性能参数：面积，速度，精度，在此三个参数基础上，还要求结构规则，具有模块化，这样有利于设计及版图布局布线。PSDA 算法的 VLSI 结构着眼于减少使用乘法器和加法器的数目。该结构使用了 2^{n-1} 个乘法器和 2^{n+2} 个加法器。相对于行列分解法结构(RCM)，该结构不需要转置存储器，而加法器和乘法器的数量与之规模相当。相对于其他直接算法，该结构所用的乘法器的数量是最少的。

5. 研究了 PSDA 算法在 $q \times q$(q 为奇素数)长度二维 DCT－II 变换中的 VLSI 实现结构。该结构的核心为一个部分和计算单元和矩阵变换单元。相对于行列分解法结构(RCM)，该结构不需要转置存储器，加法器和乘法器的数量与之规模相当。

1.3 全文结构安排

本文主要研究二维 DCT 快速算法及其 VLSI 实现结构，文章的内容安排如下：

第一章为绪论，首先介绍了离散余弦变换(DCT)的概念；然后介绍了应用背景，实现原理和未来发展趋势；最后说明了本论文所要解决的问题和取得的成果。

第二章为离散余弦变换(DCT)快速算法及其 VLSI 实现综述。首先指明了 DCT 快速算法发展中面临的主要问题，重点说明了 DCT 快速算法所要解决的关键技术问题，综述了各种一维和二维的 DCT 快速算法的发展概况。一般的 DCT 快速算法可划分为两种：间接算法和直接算法。两种算法都是集中在蝶型结构上，而且目的都是为了减少乘法和加法的计算量。其次介绍并总结了适合 VLSI 实现的各种 DCT 硬件结构的特点。介绍了基于乘法器，分配算法，脉动阵列，CORDIC 算法等当前流行的 DCT 结构。最后介绍了 ASIC 技术和 ASIC 设计流程。对 EDA 开发工具进行了简单的说明。

第三章叙述了数论基本知识，分别叙述了同余，唯一分解定理，剩余类，简化剩余系，同余式，中国剩余定理，原根和指标，为后续章节的展开提供理论基础。数论理论在快速变换领域有着广泛的应用。由于很多正交变换系的核函数是三角函数，而三角函数对 2π 做求模运算具有不变性。因此快速变换的算法研究可以转化为数论中的同余问题。在一维 DCT 变换中，当变换长度为合数时，通过数论中的中国剩余定理可以将一个长序列的一维 DCT 变换转换为一个多维短序列 DCT 变换。从而大大减少了计算复杂度。当变换长度为素数时，通过数论中的原根与指标可以将一维 FFT 变换转换为循环卷积，一维 DCT 变换转换为循环卷积或

扭循环卷积。循环卷积的计算可以通过先求两个序列 Z 变换乘积的反 Z 变换求得。而数论理论中的同余式理论在这个领域得到了广泛的应用。

第四章叙述了一维离散余弦变换(DCT)的快速变换算法设计。一维 DCT 快速变换算法是所有 DCT 快速变换算法的基础。本章详细叙述了 2^n，q^n，素长度以及任意长度 DCT 变换算法，为下一章的部分和分解算法打下基础。第一部分叙述了 2^n 长度 DCT 的常用快速变换算法－基 2 递归分解算法。第二部分为 q^n 长度的基 q 递归分解算法。第三部分为素长度 DCT 快速算法，算法的核心在于去掉变换核中的$(2x+1)$项，将$(2x+1)k$ 形式转换为 yk 形式，再通过数论理论中原根的特性，将原变换转换为循环卷积的形式。最后为任意复合长度的一维 DCT 快速算法的设计思想，通过中国剩余定理，阐述了一维 DCT 与多维 DCT 之间的相互转换关系。通过坐标变换，一维DCT可以分解为多维DCT，而多维DCT也可以转换为一维DCT。

第五章为二维离散余弦变换算法研究。利用数论理论，本章提出了二维 DCT 变换的部分和分解快速算法(PSDA)。PSDA 算法是一种直接分解算法，其核心思想为将二维 DCT 变换直接转换为若干个一维 DCT 变换实现。根据同余理论，将频域输出划分为若干个子集，每一子集内的所有元素构成一个一维 DCT 变换的输出。第 2 节详细讲述了 $2^n \times 2^n$ 型部分和分解算法的推导和实现。在本节中给出了部分和定义，分析了部分和的特性，提出了变换输出的子集划分准则，部分和的计算方法，公共加法项的合并原则。第三节把部分和算法扩展到 $q \times q$ (q 为奇素数)型二维 DCT，将变换输出划分为四个分量之和。第一个分量为一个$(q-1)/2 \times (q-1)/2$ 型二维 DCT，PSDA 算法可以将其直接转换为 $2(q-1)$个$(q-1)/2$ 长度循环卷积的计算。第二个分量和第三个分量为一个一维 DCT 变换，可通过第四章中叙述的算法求解，第四个分量为一个常量。本节最后一节给出了 PSDA 算法在该类型 DCT 中的运算复杂度。

第六章是 PSDA 算法的 VLSI 实现。第二节讲述了 PDSA 算法的 VLSI 结构设计。该结构的核心是部分和的计算。根据 PSDA 算法原理，经过预处理对该类型 DCT 的每一列输入数据，某一特定余弦因子对应的元素不会超过两个。这两个元素可以通过查表法选出累加得到部分和输出。最终的变换输出可以既可以通过乘加运算求和输出，也可以使用其他一维 DCT 变换模块输出。在本节中，分别给出了 $2^n \times 2^n$ 型和 $q \times q$ 型 DCT 的 VLSI 结构。第三节讲述了算法的 ASIC 验证过程，并将验证后的 IP 核与其他公司的产品进行了比较分析。

第七章总结了全文的设计工作，提出了算法未来的发展方向。

第二章 离散余弦变换(DCT)快速算法及其 VLSI 实现综述

本章叙述了一维离散余弦变换，二维离散余弦变换及其主要研究方法；介绍了当前主要的离散余弦变换 VLSI 结构和 ASIC 技术的基本开发流程。

2.1 引言

自从离散余弦变换被提出来以来，DCT 已经被广泛的应用在数字信号处理中，特别是语音及图像数据压缩，自适应滤波以及通信系统等领域。目前基于 8×8 的二维离散余弦变换(2D－DCT)被广泛的应用在各种图像和视频压缩标准中，诸如 JPEG，H.261，MPEG－1，MPEG－2，H.263，H.263+和 MPEG－4。DCT 是图像和视频压缩系统中的一个核心部分，它主要是完成数据变换的功能。类似于离散傅立叶变换，DCT 也是把一个信号或者图像从空域变换到频域中。DCT 根据图像内容的重要性把图像分成不同的部分。对于大多数的图像来说，信号能量主要集中在低频部分。压缩一般都是通过丢弃高频信息来实现的，因为高频信息的损失不容易被人的视觉系统察觉。在选取均方差准则下，KLT 是信号处理的最佳变换，但是 KLT 没有快速算法，且计算困难，没有实用价值。在有快速算法的次佳变换中，DCT 的基向量最接近 KLT，因而发展很快。本章首先对离散余弦变换各种算法的发展作简单综述，然后重点讨论适合于 VLSI 实现的 DCT 算法结构。

2.2 DCT 快速算法

2.2.1 一维离散余弦变换(1－D DCT)的快速算法

DCT 首先由 N.Ahmed 等人于 1974 年提出[1]，N 点 1－D DCT{Y(k)：k=[0，N－1]}对应{X(n)：n=[0，N－1]}有如下四种变换方式：

（1） DCT－I

$$X(k)=\sqrt{\frac{2}{N}}c_k\sum_{n=0}^{N}c_n x(n)\cos\left(\frac{kn\pi}{N}\right),\quad k=0,1,...N \tag{2－1a}$$

$$x(n)=\sqrt{\frac{2}{N}}c_n\sum_{k=0}^{N}c_k X(k)\cos\left(\frac{kn\pi}{N}\right),\quad n=0,1,...N \tag{2－1b}$$

（2） DCT－II

$$X(k)=\sqrt{\frac{2}{N}}c_k\sum_{n=0}^{N-1}x(n)\cos\left(\frac{(2n+1)k\pi}{2N}\right),\quad k=0,1,...N-1 \qquad (2-2a)$$

$$x(n)=\sqrt{\frac{2}{N}}\sum_{k=0}^{N-1}c_kX(k)\cos\left(\frac{(2n+1)k\pi}{2N}\right),\quad n=0,1,...N-1 \qquad (2-2b)$$

（3） DCT－III

$$X(k)=\sum_{n=0}^{N-1}\sqrt{\frac{2}{N}}c_kc_nx(n)\cos\left(\frac{(2k+1)n\pi}{2N}\right),\quad k=0,1,...N-1 \qquad (2-3a)$$

$$x(n)=\sqrt{\frac{2}{N}}c_n\sum_{k=0}^{N-1}c_kX(k)\cos\left(\frac{(2k+1)n\pi}{2N}\right),\quad n=0,1,...N-1 \qquad (2-3b)$$

（4） DCT－IV

$$X(k)=\sqrt{\frac{2}{N}}\sum_{n=0}^{N-1}x(n)\cos\left(\frac{(2k+1)(2n+1)\pi}{4N}\right),\quad k=0,1,...N-1 \qquad (2-4a)$$

$$x(n)=\sqrt{\frac{2}{N}}\sum_{k=0}^{N-1}X(k)\cos\left(\frac{(2k+1)(2n+1)\pi}{4N}\right),\quad n=0,1,...N-1 \qquad (2-4b)$$

其中：

$$c_k=\begin{cases}1/\sqrt{2} & k=0,N\\ 1 & k=1,2,...N-1\end{cases}$$

通常采用的是 DCT－II，说到 DCT 和 IDCT 时大多是指(2－2*a*)和(2－2*b*)。

为了有效的计算 DCT，许多算法被提出，一般 DCT 快速算法分为两种：间接算法和直接算法。两种算法都是集中在蝶型结构上，而且目的都是为了减少乘法和加法的计算量。

间接算法是利用 DCT 和 DFT，DHT 等正交变换之间的关系，用 DFT 或 DHT 快速算法来计算 DCT[2-8]。间接算法过程简单，主要工作是处理算法间的转换，因此往往需要加一些额外的操作步骤，由于将其他变换的快速算法应用在 DCT 中总有其自身的局限性，所以现在算法上已很少有人采用间接算法来计算 DCT。

直接算法包括 DCT 变换矩阵分解，递归算法两种技术，不同之处在于矩阵分解是利用稀疏矩阵分解法将变换矩阵分解，而递归算法是由较低阶 DCT 矩阵递归产生较高阶 DCT 矩阵，可以说递归算法是分解算法的逆算法，但递归算法较矩阵分解算法有良好的数值稳定性。

近年来提出了许多计算离散余弦变换的算法。这些算法大部分都需要 12 个乘法器和 29 个加法器来实现一个 8 点 DCT，如表 2－1 所示。

表 2－1 8 点 DCT 各种算法的计算量

作者	Chen	Wang	Lee	Vetterli	Suehiro	Hou
乘法器	16(13)	13	12	12	12	12
加法器	26(29)	29	29	29	29	29

W.H.Chen 的快速算法[47]，是第一个发表的算法，用的是非常普通的结构，算法中用到的乘法器和加法器数目可以很容易地减小到表 2－1 括号中的数字。Wang 的算法能够很容易从他的 DCT 算法得到离散正弦变换(DST)，离散小波变换(DWT)，离散傅立叶变换(DFT)。1984 年，B.G.Lee 提出一种使用余割因子的 DCT 矩阵分解算法[48]，得到 Cooley－Tukey 式的简单结构，受到广泛重视，它的第一级非常普通，但是在最后一级并不是通常的数据流，需要反余弦值作为系数，这会引起数据溢出的问题。Vetterli 在他的算法中用到了递归公式，而且加法操作需要紧跟在递归计算模块后，这增加了算法中通讯结构的复杂度。1987 年，几乎同时有三篇论文论述了 DCT 的递归算法[49, 50, 51]，其中 H.S.Hou 的快速算法较具有代表性。近几年出现的新算法大多是这几种算法的改进。Feig－Winograd 通过把 DCT 看做是一种循环旋转运算，证明了在有理数域上计算长度为 2^n 的 1－D DCT 所需的最小实数乘法次数为 $2^{n+1}-n-2$[52]，对于 8 点的一维 DCT，最少需要 11 次乘法。Loeffler 的 DCT 快速算法[53]达到了这一极限，它是将 DCT 运算转为旋转运算。

2.2.2 二维离散余弦变换(2－D DCT)快速算法

二维 N×N 点 DCT 定义为：

$$X(k,l)=\frac{2c(k)c(l)}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}x(m,n)\cos\left\{\frac{(2m+1)k\pi}{2M}\right\}\times\cos\left\{\frac{(2n+1)l\pi}{2N}\right\}$$

$$(k=0,1,...,M-1;\ \ l=0,1,...N-1) \qquad (2-5a)$$

$$(m=0,1,...,M-1;\ \ n=0,1,...N-1) \qquad (2-5b)$$

$$c(k)=\begin{cases}1/\sqrt{2} & k=0\\ 1 & k=1,2,...N-1\end{cases}$$

2－DDCT 的快速算法主要有两种：行列分解法(RCM)及非行列分解法(NRCM)。RCM 方法是将 N×N 的数据按行(或列)方向进行 N 个 1－D DCT 计算，产生中间矩阵，然后对中间矩阵再按列(或行)方向进行 N 个 1－D DCT 计算，最后得到 2－D DCT 结果。

NRCM 即直接分解法，典型的直接分解算法是 2－D 矢量基 DCT 算法。2－D 矢量基算法通常为 1－D DCT 的二维扩展，采用矢量基算法乘法数会减少至行列分解算法的 75%。2－D 矢量基 DCT 算法主要有基于 B.G.Lee.的算法和基于 Hou 的算法。对于较长长度的 DCT，基于 Hou 的算法较基于 Lee 的算法性能优越。目前最有效的 2－DDCT 直接分解算法主要有森川良孝等人的算法与 N.I.Cho 等人的算法[30]，这两种算法都将乘法运算量减至传统行列分解法的 50%。森川良孝算法采用切比雪夫多项式同余形式，把 2－D DCT 转换成 N 个 N 点 1－D DCT 和长度为 N 的切比雪夫多项式变换。N.I.Cho 算法采用三角函数法，将长度为 2^n 的 2－D DCT 表示为两个新的二维变换之和，再利用换序移位和附加的实数加法运算，将两个新二维变换转换为 N 个 N 点 1－DDCT。Feig－Winograd 证明了在有理数域上计算长度为 2^n 的 2－D DCT 所需的最小实数乘法次数为 $2^{2n+1}-n2^n-2n+1$，现在还没有达到这个极限的算法。

2.3 DCT 的 VLSI 结构研究现状

对于2－D DCT 的硬件结构，也分为行列分解法(RCM)及非行列分解法(NRCM)两类。对于 2－D DCT RCM 方法框图如图 2－2 所示，此结构采用的算法为各种一维 DCT 的快速算法，行和列的 1－D DCT 结构相同，因此有时为了减少面积只用一个 1－D DCT 处理单元完成两次 1－D DCT 计算，当然这样影响操作时间。对于高吞吐量 RCM 系统，为了快速安排行列转换模块之间的数据流，需要复杂的中间数据转换电路(TRAM)。一般快速转换电路需要大量芯片面积，目前，已有无中间转换电路的 RCM 系统，其中很多是脉动阵列实现的结构。NRCM 系统没有矩阵转换电路，采用的算法是各种非行列分解 2－D DCT 快速算法，它往往要求整个 N×N 输入数据同时参与计算，因此 I/O 处理及数据传递电路复杂，使得它们的 VLSI 实现在性能上不如 RCM 系统。因此，现在行－列分解的方法仍然被广泛地应用在 2－D DCT 芯片设计中。

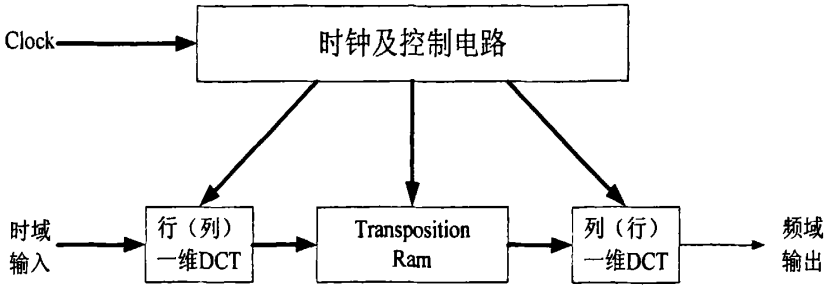


图 2－1 二维 DCT 行列分解法框图

一般一个硬件算法结构的优劣主要决定于三个性能参数：面积，速度，精度，在此三个参数基础上，还要求结构规则，具有模块化，这样有利于设计及版图布局布线。现在已有多种 DCT 算法结构被提出，这些结构都有各自的缺点，因此在各种应用中这些结构并存。另外，因为每种 DCT 算法都有它自己的特殊性和应用领域，不是所有的算法都适合于 VLSI 实现。一个算法 VLSI 实现的有效性主要基于算法中算术单元之间的数据传递的复杂性，而不是算法的计算量。

本文沿用 1998 年陈禾，毛志刚等人的分类[54]，比较各种 DCT 算法结构的优缺点和概述当前研究的状况。

2.3.1 基于乘法器的 DCT 结构

此类结构多采用 W.H.Chen 的算法[4]直接用乘法器去实现 DCT，改善此类 DCT 结构的关键在于乘法器结构的改进，例如 1995 年 A.Madisetti 等人提出的 2－D DCT 结构[13]，采用了乘数再编码乘法器，用冗余的 SD 编码(带符号数编码)来代替普通的乘数数位，使越过乘数中的一串零的平均移位长度增加，从而减少乘法中的加法操作，提高乘法速度。此类 DCT 结构往往因引入乘法器而导致面积增加，且因乘法过程中要有截断舍入，带来的误差将影响精度。另外，此类结构虽具有模块化，但结构不规则，不利于布局布线。

2.3.2 基于分配算法(Distributed Arithmetic－DA)的 DCT 结构

基于 DA 算法的 DCT 结构由于其结构紧凑，高度规则化，精度高，速度快而广泛应用在各个领域中。DA 方法最早是由 Peled 与 Liu 于 1974 年提出的[55]，其基本思想是通过 ROM 查找表，利用 ROM 和累加器(MAC)代替了乘法器。因为 DCT 中的余弦系数为固定系数，因此结果可以预先计算出来存于 ROM 中，根据输入数据对其取址即可。

1987 年，由 M.T.Sun 等人首次将 DA 技术引入 DCT 中，并将其实现[56]。其 2－D DCT 采用 RCM 方法，对于 1－D DCT 采用 Chen 算法，然后用 DA 方法实现。基于 DA 技术的 DCT 有高度规则性结构，非常适合于 VLSI 实现，而且计算是以分配方式进行的，截断被限制，因而能获得较其它结构更高的精度。但是 DA 算法结构采用位串行实现，这样限制了速度，另外 ROM 的取址限制了 DCT 运算制度。很多研究员从这几个方面入手，对基于 DA 方法的 DCT 结构提出了改进，例如 1992 年，S.Uramoto 等人提出的基于 DA 方法的 DCT/IDCT 结构通过采用列交迭存储器的布局结构和双层 ROM 电路两种技术[57]，减少了硬件传播时间，并将 DCT 与 IDCT 的存储内容用一块 ROM 来完成，减少了 ROM 的尺寸；1994 年 M.Matsui 等人提出了一种时钟频率达 200MHz 的 DCT 结构[58]。它将广泛应用在存储器电路中的双轨逻辑(输入/输出为互补双值)应用于 MAC 中，显著的减少了传播时间，加快了 DCT 运算速度。但此结构要求的工艺复杂，要同时用到三种工艺，因此成本很高。

2001 年 Sungwook，Yu 等人提出了一种基于递归 DCT 算法的规则 DCT 结构[59]，该结构较之传统的基于 W.H.Chen 算法的行列分离结构面积又减少了 17%。而且吞吐量保持不变。在不使用部分和技术的情况下，该方法就已经急剧减少了 ROM 的大小。如果再使用部分和技术，就进一步减少了 ROM 的个数且将比特并行加法器换成了比特串行加法器。

2002 年，Ahmed，Shams 等人提出了一种新的分配算法结构[60]，他们称之为 NEDA。该结构较以往 DA 结构的不同之处在于它把固定系数分配到比特域，使得不再需要任何的 ROM 了，并且进一步通过挖掘加法器阵列的冗余性把加法次数降到最低。这样，该结构就不需要任何的 ROM，乘法器，和减法器，所以非常适合低功耗的应用场合。

2.3.3 DCT 的脉动阵列(Systolic Array)实现

脉动阵列概念最早由 H.T.Kung 于 1982 年提出[61]，阵列由一组简单，重复的处理单元(PE)组成，每个 PE 执行固定的，简单的操作，每个 PE 只与其相邻的 PE 有规则的联接，操作时，数据经过连成流水线的 PE，沿途得到连续的有效处理。这种高度的并行处理结构，大大提高 VLSI 的速度，且结构非常规则。DCT 的脉动阵列种类比较多，采用的算法也各不相同，结构实现的主要工作是选取合适的 PE 单元，按照陈禾等人的分类将 DCT 脉动阵列分成以下四类：

①基于 FFT 类型的 DCT 脉动阵列

因为 FFT 技术较成熟，因此最先提出的几种 DCT 结构都是基于 FFT 脉动结构的，在 FFT 脉动阵列的基础上经简单变换实现 DCT，有代表性的是 N.I.Cho 等人于 1990 年提出的结构[62]。

一般在很多实时，高速信号处理中对 DFT，DCT 脉动结构的要求是： 1)每个 PE 要有最少的周边单元；2)数据以数据流形式输入，以数据流形式输出；3)不要求数据预载，即数据参与计算时才载入 PE。因此针对 Cho 中的不足，N.R.Morthy 等人于 1994 年提出了一种改进结构[63]，符合高速 DCT 脉动阵列结构 VLSI 实现的要求。此结构组成的 2－D DCT，虽属 RCM 方法，但无需转换阵列 TRAM。

②基于三角分解法的 DCT 脉动结构

此类结构是由三角公式进行递推得到的一系列递归公式构成的脉动阵列。基于三角分解法的 DCT 脉动结构最早由 L.W.Chang 等人于 1991 年提出[64]，其原理是将 DCT 变换核 cosine 进行三角分解得到一系列递归公式，这些递归公式构成了 1－D DCT 线性脉动阵列。C.L.Wang 等人于 1995 年提出一种脉动阵列[65]，其三角法是基于切比雪夫多项式，此结构有更少的 I/O 端口及更高规则结构。

③直接基于 DCT 定义的脉动结构

此类结构是直接由 DCT 定义出发实现脉动阵列，例如 C.L.Wang 等人于 1995 年提出的一种线性脉动 DCT 结构[66]，对于 N×N 点 2－D DCT，此结构只需 N 个 PE 单元，有较高的规则结构，且不用做电路改动，可直接来设计 DST，DHT 的脉动结构。

④基于各种 1－D DCT 快速算法的脉动阵列

此类脉动阵列是根据各算法的自身特点来寻找 PE 的。例如马维祯等提出的 DCT 脉动结构是通过利用 Horner 规则将 Vetterli－Nussbaumer 的快速算法写成递归形式而实现的[67]。Y.T.Chang 于 1995 年提出的 2－D DCT 结构是无转换 TRAM 的 RCM 方法，其算法是基于 Chen 的快速算法，对行和列的 1－D DCT 采用了不同的 PE 单元。它要求 N 为偶数即可，不必为 2 的指数，与其它 2－D DCT 脉动结构比，它更有简单，更规则的数据传递与控制电路。而 C.M.Wu 与 A.Chiou 于 1992 年提出的一种基于 Lee 算法的 SIMD(单指令流多数据流)脉动结构[68]，只有四个处理单元(PE)，对应 8 点 1－D DCT 算法流图中每一级的四个蝶型运算。

脉动阵列通过并行及流水线处理可获得较高处理速度，且其具有模块化，规则结构，局部内联等特点，适合于 VLSI 实现。但脉动阵列往往要求多个数据同时参与计算，因此通常有较复杂的 I/O 处理。另外，因每个 PE 内有多种算术运算，

截断与舍入误差会影响精度。而且由于一个 PE 内多种运算并存，有时会有多个乘法运算，因而面积往往很大，且脉动阵列结构的时钟控制电路复杂。这些原因使得现在各种应用中采用脉动阵列 DCT 结构的并不多见。

2.3.4 基于 CORDIC 运算技术的 DCT 结构

CORDIC 是坐标旋转数字计算机的同义词。这种计算是 Volder 于 1959 年提出来的[69]。这个概念虽然由来已久，但它的实现和应用还在继续发展，因为此技术简化了体系结构，提高了速度，降低了算术模块的功耗。考虑下列计算：

$$X' = X\cos\theta - Y\sin\theta = (X - Y\tan\theta)\cos\theta$$

$$Y' = Y\cos\theta + X\sin\theta = (Y + X\tan\theta)\cos\theta$$

CORDIC 算法是一个通过以角度 $\alpha_j = \arctan 2^{-j}$ 进行矢量旋转而得到的迭代过程。通过采用 CORDIC 算法，旋转运算中的乘法器可仅由加/减法器与移位寄存器来实现，减少了电路，并通过采用流水线结构，加快了数据传输速度。

1990 年，Duh 与 Wu 首次将 CORDIC 引入 DCT 计算中[70]。E.P.Mariators 等人的旋转结构利用 DCT 中每个 CORDIC 旋转角度是预先确定的事实，找 θ 的表达式，减少运算量，也减少了硬件[71]。而 F.Zou 等[72]是将 DCT 结构进行改进，使得完成一个 DCT 需较少 CORDIC 计算，获得了较高的吞吐量。

但现在 CORDIC 算法结构应用于实际的例子很少，这是因为此结构虽占用面积少，吞吐量高，但速度较其它结构慢，且精度较低。不过将 CORDIC 算法引入 DCT 这一算法刚刚起步不久，以后会有所发展。

2.3.5 其它 2－D DCT 结构

近年来提出了很多用直接法实现的 DCT 结构，通过采用不同的技术提高处理速度和精度，使占用的面积更少。例如 C.T.Chiu 等人于 1992 年提出一种帧递归网络(Lattice)2－D DCT 结构算法[73]。此算法是一种直接 2－D 方法，即 NRCM 方法。系统仅要求 2 个 1－D 网络 DCT 模块阵列。N×N 点的 2－D DCT 总乘法量为 8N，是目前所有 2－D DCT 算法中乘法数量最少的，并且它对 N 的尺寸没有限制。1996 年 V.Srinivasan 等人将此算法实现[74]。此设计不足之处在于，其硬件结构较 RCM 结构复杂，且精度较低。在一些图像压缩应用中，如 JPEG，MPEG，还有采用规格 DCT(scaled DCT)算法的结构。因为在图像压缩编码处理中，图像中的每一像素经过 DCT 后的变换值要除以预先给定的量化矩阵中对应的系数，然后量化取整并

进行熵编码。因此在一些结构中，将量化引入 DCT 或 IDCT 中，这就是规格 DCT 的思想[75]。采用此算法可减少乘法运算量，对 8 点 DCT 只需 5 次乘法。但此算法精度较差，且结构不规则。而 S.C.Hsia 等人于 1995 年提出的一种连续系数 2－D DCT 并行 VLSI 结构[76]，将解压缩系统中的 Z 排序，运动补偿，逆量化等与 IDCT 有机地结合起来，提高了效率，加快了速度，但是电路复杂，缺少通用性。T.S.Chang 等人在 2000 年设计了成本非常低的 DCT 处理核[77]。算法采用直接算法，并且采用基于分配算法的位级(bit－level)加法器和共用子模块等方法来减少硬件实现的代价。整个处理核占用的面积非常小，但是处理速度低，并且精度较差。

2.4 本章小结

本章研究了 DCT 的快速算法及 VLSI 设计结构。近年来提出了很多实用的快速算法，比较有代表性的是 Loeffler 等发表的算法和基于傅立叶变换的缩放 DCT，其中 Loeffler 算法中的乘法器数目已经达到了理论下限。对 2－D DCT 的 VLSI 实现结构进行了分类和比较，总结了各个结构的优缺点。2－D DCT 的 VLSI 实现结构，也分为行列分解法和直接法两类。行列分解法结构采用的算法为各种 1－D DCT 快速算法。行和列的 1－D DCT 结构相同。直接法它往往要求整个 N×N 输入数据同时参与计算，因此 I/O 处理及数据传递电路复杂，使得它们的 VLSI 实现在性能上往往不如行列分解法。

因此，在考虑硬件实现时，主要考虑计算的复杂度，速度，芯片面积，精度等因素，总是在计算的时间和芯片面积上折衷，所以现在有多种 DCT 算法结构被提出，这些结构都有自己的优缺点。基于乘法器的 DCT 结构计算速度快，但是此结构往往因为乘法器的引入而导致面积的增加，而且因乘法过程有截断舍入，带来的误差将影响精度。基于分配(DA)算法的 DCT 结构具有结构紧凑，高度规则化，精度高，速度快的优点，但是 DA 算法结构采用位串实现，这样限制了速度，另外 ROM 取地址操作也限制了 DCT 运算速度。脉动阵列结构具有处理速度快，且其具有模块化，规则结构，局部内联等优点，适于 VLSI 实现。但脉动阵列往往要求多个数据同时参与计算，因此通常要求较复杂的 I/O 处理。另外，因为每个 PE 内有多种算术运算，截断和舍入误差将会影响精度。而且由于一个 PE 内有多种运算并存，有时会有多个乘法运算，因而面积往往很大，且脉动阵列结构的始终控制电路复杂。这些原因使得现在各种应用中，DCT 采用脉动阵列结构的并不多见。

基于 CORDIC 算法的结构具有占用面积少，吞吐量高的优点，但速度较其它结构慢，且精度较低，因而应用于实际的例子很少。

第三章 与 DCT 变换相关的数论理论

本章中叙述了一些数论的基础知识，如同余，同余式，原根，指标等，在后续章节中，将使用这些理论进行 DCT 快速算法的研究。

3.1 引言

数论这门学科最初是从研究整数开始的，所以叫做整数论。后来整数论又进一步发展，就叫做数论了。确切的说，数论就是一门研究整数性质的学科。

自古以来，数学家对于整数性质的研究一直十分重视。我国古代许多著名的数学著作中都有关于数论内容的论述，比如求最大公约数、勾股数组、某些不定方程整数解的问题等等。在国外，古希腊时代的数学家对于数论中一个最基本的问题－整除性问题就有系统的研究，关于素数、合数、约数、倍数等一系列概念也已经被提出来应用了。后来的各个时代的数学家也都对整数性质的研究做出过重大的贡献，使数论的基本理论逐步得到完善。

数论是一门高度抽象的数学学科，长期以来，它的发展处于纯理论的研究状态，它对数学理论的发展起到了积极的作用。但对于大多数人来讲并不清楚它的实际意义。

由于近代计算机科学和应用数学的发展，数论得到了广泛的应用。比如在计算方法、代数编码、组合论等方面都广泛使用了初等数论范围内的许多研究成果；目前，在雷达领域中，数论中的中国剩余定理被用在脉冲多普勒雷达上，解目标的距离模糊和速度模糊，在接收雷达的天线阵列中解目标到达角的模糊；在信号处理理论中，基于中国剩余定理的数论变化是一种重要的快速变换方法；在 IC 设计中，应用中国剩余定理可以获得高效的 IIR 滤波器设计的方法；中国剩余定理还奠定了目前世界上最流行的公钥加密技术(即 RSA)的基础。原根和指标理论，多项式变换理论在快速变换领域中得到了广泛的应用。

数论理论在 DCT 快速算法研究中也有着广泛的应用。第二节叙述了同余的概念及其基本性质，数的唯一分解定理和简化剩余系。在第三节第一小节中叙述了一次同余式，高次同余式的求解；第二小节叙述了中国剩余定理；第三小节叙述了高次同余式的解数及解法。第四节叙述了原根与指标，给出了原根存在的条件。

本章所引用定理的证明请参考[78]。

3.2 同余

3.2.1 同余的概念及其基本性质

给定一个正整数 m，把它叫做模。如果用 m 去除任意两个整数 a 与 b 所得的余数相同，则称 a，b 对模 m 同余，记做 $a \equiv b$。如果余数不同，则称 a，b 对模 m 不同余，记做 $a \not\equiv b$。只有零的模称为零模。下面讨论的对象仅为整数的模，由定义易得：

定理 3.2.1.1

1)任何模中必含有 0

2)若 a，b 在模中，则 $am+bn$ 也在模中，m，n 为任意常数。

定理 3.2.1.2 模中两个元素 a 及 b，则对任意的整数 m，n，$am+bn$ 所组成的整数为一个模。

定理 3.2.1.3 模中任意一个非零元素，必定是模中最小正整数的整数倍数。

定义. 设 a，b 为二整数，将既能被 a 整除又能被 b 整除的最大正整数 d 称为 a，b 的最大公因数，用符号(a, b)表示。

定理 3.2.1.4 (a, b)有如下性质：

有整数 x，y，使 $d=(a, b)=ax+by$

1） 任二整数 x，y，必有$(a, b) \mid ax+by$

2） 若 $e \mid a$，$e \mid b$，则 $e \mid (a, b)$

在 DCT 变换中，变换核函数为余弦函数 $\cos((2x+1)k\pi/2N)$。该余弦函数具有周期性，周期 $4N\pi$。因此，核函数中的余弦因子系数$(2x+1)k$ 与$(2x+1)k \pmod{4N}$等价，因此 DCT 变换可以转换为同余问题来研究。

3.2.2 唯一分解定理

定理 3.2.2.1 若 p 为素数且 $p \mid ab$，　则 $p \mid a$ 或 $p \mid b$

定理 3.2.2.2 若 $c>0$，及$(a, b)=d$，　则$(ac, bc)=dc$

定理 3.2.2.3 n 的标准分解式是唯一的。换言之，若不计次序，则 n 仅能由唯一之方法表示为素数之积。

在一维 DCT 变换中，如果变换长度是一个合数，可以用唯一分解定理将这个

合数分解为若干各素数之乘积，将一个长序列变换转换为若干个短序列变换，从而大大减少运算的复杂度。

3.2.3 剩余类及完全剩余系

在 3.1.1 中引入了同余的概念。由于有了同余的概念，可以把余数相同的数放在一起，这样就产生了剩余类的概念。

定理 3.2.3.1 若 m 是一个给定的正整数，则全部整数可分为 m 个集合，记做 K_0，K_1，…，K_{m-1}。其中 $Kr(r=0, 1, \cdots, m-1)$是由一切形如 $qm+r(q=0, \pm1, \pm2\cdots)$ 的整数所组成的。这些集合具有下列性质：

(I) 每一整数必包含在而且仅在上述的一个集合里面。

(II) 两个整数在同一个集合的充分与必要条件是这两个整数对模 m 同余。

定义 定理 3.2.3.1 中的叫做模 m 的剩余类，一个剩余类中任一数叫做它同余的数的剩余。若 a_0，a_1，…，a_{m-1} 是 m 个整数，并且其中任何两数都不同在一个剩余类里，则 a_0，a_1，…，a_{m-1} 叫做模 m 的一个完全剩余系。

定理 3.2.3.2 设 m 是正整数，$(a, m)=1$，b 是任意整数，若 x 通过模 m 的一个完全剩余系，则 $ax+b$ 也通过模 m 的完全剩余系，也就是说，若 a_0，a_1，…，a_{m-1} 是模 m 的完全剩余系，则 aa_0+b，aa_1+b，…，$aa_{m-1}+b$ 也是模 m 的完全剩余系。

定理 3.2.3.3 若 m_1，m_2 是互质的两个正整数，而 x_1，x_2 分别通过模 m_1，m_2 的完全剩余系，则 $m_2x_1+m_1x_2$ 通过模 m_1m_2 的完全剩余系。

定义 0，1，…，$m-1$ 这 m 个整数叫做模 m 的最小非负完全剩余系；当 m 为偶数时，$-m/2$，…，-1，0，1，…，$m/2-1$ 或 $-m/2+1$，…，-1，0，1，…，$m/2$ 叫做模 m 的**绝对最小完全剩余系**；当 m 为奇数时，$-(m-1)/2$，…，-1，0，1，…，$(m-1)/2$ 叫做模 m 的绝对最小完全剩余系；

如果将模 m 的剩余类看成一个元素，剩余类的相等就可以用同余来刻画，同余的运算性质就可以转化为剩余类的运算性质。这样，模 m 的剩余类的集合对这些运算就作为一个环，称为剩余类环。如果模是合数，那么就有不等于零的剩余类，相乘后为 0，即有零因子。这就为抽象代数提供了一个有零因子的环的具体例子。上述环中所有与 m 互质的剩余类对乘法构成一个群。当模 m 为素数 p 时，上述的环构成一个域，通常记为 Fp。它有 p 个元素，这是有限域的一个重要例子。对于多项式的同余也可以有类似的结论。

在DCT变换中，如果可以将用完全剩余系来表示余弦因子系数$(2x+1)k$，那么就可以把对原余弦因子系数的研究转换为对完全剩余系的研究。

3.2.4 简化剩余系与欧拉函数

上节讨论了完全剩余系的基本性质，本节进一步讨论完全剩余系中与模 m 互质的整数，这就需要引进简化剩余系的概念。在讨论简化剩余系的过程中，需要用到数论上一个很重要的概念，欧拉函数。以下先给出几个定义。

定义 欧拉函数 $\varphi(a)$是定义在正整数上的函数，它在正整数 a 上的值等于序列 0，1，…，$a-1$ 中与 a 互质的数的个数。

定义 如果一个模 m 的剩余类里的数与 m 互质，就把它叫做一个**与模 m 互质的剩余类**。在与模 m 互质的全部剩余类中，从每一类中各任取一数所作成的数的集合，叫做模 m 的一个**简化剩余系**。

定理 3.2.4.1 模 m 的剩余类与模 m 互质的充要条件是此类中有一数与 m 互质。因此与模 m 互质的剩余类的个数是 $\varphi(m)$，模 m 的每一简化剩余系是由与 m 互质的 $\varphi(m)$ 个对模 m 不同余的整数组成的。

定理 3.2.4.2 若 a_1，a_2，…，$a_{\varphi(m)}$是 $\varphi(m)$个与 m 互质的整数，并且两两对模 m 不同余，则 a_1，a_2，…，$a_{\varphi(m)}$是模 m 的一个简化剩余系。

定理 3.2.4.3 若$(a, m)=1$，x 通过模 m 的简化剩余系，则 ax 通过对模 m 的简化剩余系。

定理 3.2.4.4 若 m_1，m_2 是两个互质的正整数，x_1，x_2 分别通过模 m_1，m_2 的简化剩余系，则 $m_2x_1+m_1x_2$ 通过模 m_1m_2 的简化剩余系。

推论 若 m_1，m_2 是两个互质的正整数，则 $\varphi(m_1m_2)=\varphi(m_1)\varphi(m_2)$。

定理 3.2.4.5 设 $a=p_1^{a_1}p_2^{a_2}\cdots p_k^{a_k}$，则

$$\varphi(a)=a(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_k}) \qquad (3-1)$$

3.2.5 欧拉定理与费马定理

欧拉定理 设 m 是大于 1 的整数，$(a, m)=1$，则

$$a^{\varphi(m)}\equiv 1(\mathrm{mod}\, m)$$

推论(Fermat 定理)若 p 是素数，则

$$a^p\equiv a(\mathrm{mod}\, p)$$

3.3 同余式

3.3.1 基本概念及一次同余式

若用$f(x)$表示多项式$f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots a_0$，其中$a_i$是整数，又设$m$是一个正整数，则

$$f(x)\equiv 0 \pmod m \tag{3-2}$$

叫做模 m 的同余式。若$a_n\not\equiv 0(\text{mod m})$，则 n 叫做(3－2)的次数。

定义 若a是使$f(a)\equiv 0(\text{mod } m)$成立的一个整数，则$x\equiv a(\text{mod } m)$叫做(3－2)的一个解。

讨论方程

$$ax+b\equiv 0 \pmod m \tag{3-3}$$

的整数解。

定理 3.3.1.1 若$(a, m)\mid b$，则(3－3)有(a, m)个互不同余的 mod m 解。反之则无解。

定理 3.3.1.2 同余方程

$$a_1x_1+\cdots+a_nx_n+b\equiv 0 \pmod m \tag{3-4}$$

有解$(x_1, \cdots, x_n)$的充分必要条件为$(a_1, \cdots, a_n, m)\mid b$。若此条件适合，则其解的个数(模 m 不同余)为$m^{n-1}(a_1, \cdots, a_n, m)$。

本文第五章研究了部分和分解算法。而部分和的求解需要计算对应某一特定余弦因子的二维坐标。定理 3.3.1.2 可以求得构成某一特定余弦因子系数的坐标的个数。

3.3.2 中国剩余定理

定理 3.3.2.1 中国剩余定理 设 $m_1, m_2, \ldots, m_k$ 是 k 个两两互素的正整数，$m=m_1m_2\ldots m_k$，$m=m_iM_i$，$i=1, 2, \ldots, k$，则同余式组 $x\equiv b_1(\text{mod } m_1)$，$x\equiv b_2(\text{mod } m_2)$，$\ldots x\equiv b_k(\text{mod } m_k)$ 的解是

$$a_1x_1+\cdots+a_nx_n+b\equiv 0 \pmod m \tag{3-5}$$

其中$M_i'M_i\equiv 1(\text{mod } m_i)$，i=1，2，…，k。

定理 3.3.2.2 若$b_1, b_2, \ldots, b_k$为分别过模$m_1, m_2, \ldots, m_k$的完全剩余系，则 过模$m=m_1m_2\ldots m_k$的完全剩余系。

在一维 DCT 变换中，如果变换长度是一个合数，将这个合数分解为若干个素数之乘积后，则可以将原来的一维坐标转换为多维坐标，中国剩余定理可以建立

起这种一维坐标至多维坐标的映射。

3.3.3 高次同余式的解数及解法

本节应用以前的结果，初步地讨论一下高次同余式的解数及解法。通常是先把合数模的同余式化成素数幂模的同余式，然后讨论素数幂模的同余式的解法。

定理 3.3.3.1 若 m_1，m_2，…，m_k 是 k 个两两互质的正整数，$m=m_1m_2\ldots m_k$，则同余式

$$f(x)\equiv 0 \pmod m \tag{3-6}$$

与同余式组

$$f(x)\equiv 0 \pmod m \; i=1,2,\cdots,k \tag{3-7}$$

等价。并且若用 Ti 表示 $f(x)\equiv 0(\text{mod } m_i)$，$i=1$，$2$，…，$k$ 对模 m_i 的解数，T 表示对模 m 的解数，则

$$T=T_1T_2\cdots T_k \tag{3-8}$$

由唯一分解定理可知任一正整数 m 可以写成标准分解式：即

$$m=p_1^{a_1}p_2^{a_2}\cdots p_k^{a_k}$$

由定理 3.2.3.1 知，欲解同余式 $f(x)\equiv 0(\text{mod } m)$，只要解同余式组

$$f(x)\equiv 0 \pmod{p_i^{a_i}} \qquad i=1,2,\cdots,k$$

因此下面就来讨论

$$f(x)\equiv 0 \pmod{p^a} \tag{3-9}$$

但是由模的性质很容易知道适合(3－9)的每一个解都适合同余式

$$f(x)\equiv 0 \pmod p \tag{3-10}$$

因此欲求(3－9)的解，可以从式(3－10)的解出发。

定理 3.3.3.2 设

$$x\equiv x_1 \pmod p$$

即

$$x=x_1+pt_1,\ t_1=0,\pm1,\pm2,\cdots \tag{3-11}$$

是(3－10)的一解并且 $p\nmid f'(x_i)$（$f'(x)$是 $f(x)$ 的导数）。则(3－11)刚好给出(3－9)的一解(对模 p^a 来说)：

$$x=x_a+p^at_a,\ t_a=0,\ \pm1,\ \pm2\cdots$$

3.3.4 素数模的同余式

在 3.3.3 中，解高次同余式的问题归结到了素数模的高次同余式，但是还没有提出一般的方法去解素数模的同余式。

$$f(x)\equiv 0 \pmod p,\quad f(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots a_0 \tag{3-12}$$

其中 p 是素数，而 $a_n\not\equiv 0 \pmod p$。

定理 3.3.4.1 同余式(3－12)与一个次数不超过 $p-1$ 的素数模等价。

定理 3.3.4.2 设 $k\le n$，而 $x\equiv a_i \pmod p$ ($i=1, 2, \ldots, k$)是(3－12)的 k 个不同解，则对任何整数 x 来说

$$f(x)\equiv (x-a_1)(x-a_1)\cdots(x-a_k)f_k(x) \pmod p \tag{3-13}$$

其中 $f_k(x)$ 是 $n-k$ 次多项式，首项系数是 a_n。

定理 3.3.4.3

(i)对任何整数 x 来说，

$$x^{p-1}-1\equiv (x-1)(x-2)\cdots(x-(p-1)) \pmod p$$

(ii)$(p-1)!+1\equiv 0 \pmod p$，

定理 3.3.4.4 若 $n\le p$，则同余式

$$f(x)\equiv 0 \pmod p,\ f(x)=x^n+a_{n-1}x^{n-1}+\cdots a_0 \tag{3-14}$$

有 n 个解的充分与必要条件是以 $f(x)$ 除 x^p-x 所得余式的一切系数都是 p 的倍数。

在计算循环卷积时，通常不直接按定义式计算。而是先计算原序列的 z 变换，然后将 z 变换相乘，最后通过逆 z 变换求得循环卷积结果。而 z 变换可以转换为模一个高阶项后的高次同余式。由于该高阶项一般是可分解的，可以通过中国剩余定理把高次同余式转换为若干个低次同余式，从而大大减少计算复杂度。

3.4 原根与指标

本节讨论同余式

$$x^n\equiv a \pmod m \tag{3-15}$$

在什么条件下有解。在讨论过程中要引进原根与指标这两个概念。这两个概念在数论中是很有用的。本节通过对原根与指标的研究，最后要把(3－15)对某些特殊的 m 有解的条件利用指标表达出来。

3.4.1 指数及其基本性质

由欧拉定理知道：若$(a, m)=1$，$m>1$，则 $a^{\varphi(m)}\equiv 1 \pmod m$。这就是说，若$(a, m)=1$，$m>1$，则存在一个正整数 γ 满足 $a^{\gamma}\equiv 1 \pmod m$，因此也存在满足上述要求的最小正整数。故有

定义 若 $m>1$，$(a, m)=1$，则使得同余式

$$a^{\gamma} \equiv 1 (\mathrm{mod}\ m)$$

成立的最小正整数 γ 叫做对模 m 的指数。

若 a 对模 m 的指数是 $\varphi(m)$，则 a 叫做模 m 的一个原根。

定理 3.4.1.1 若 a 对模 m 的指数是 δ，则 $1=a^0$，…，$a^{\delta-1}$ 对模 m 两两不同余。

定理 3.4.1.2 若 a 对模 m 的指数是 δ，则 $a^{\gamma} \equiv a^{\gamma'} \pmod{m}$ 成立的充分必要条件是 $\gamma \equiv \gamma' (\mathrm{mod}\ \delta)$，特别地，$a^{\gamma} \equiv 1 (\mathrm{mod}\ m)$成立的充分必要条件是 $\delta | \gamma$。

由定理 2 及欧拉定理立刻得出

推论 若 a 对模 m 的指数是 δ，则 $\delta | \varphi(m)$。

定理 3.4.1.3 若 x 对模 m 的指数是 ab，则 x^a 对模 m 的指数是 b。

定理 3.4.1.4 若 x 对模 m 的指数是 a，y 对模 m 的指数是 b，并且$(a, b)=1$，则 xy 对模 m 的指数是 ab。

3.4.2 原根存在的条件

任给一模 m，原根是不一定存在的。实际上，只有在 m 是 2，4，p^a，$2p^a$(p 是奇素数)四者之一时，原根才存在。

定理 3.4.2.1 若 p 是单素数，则模 p 的原根是存在的。

定理 3.4.2.2 设 $a \geq 1$，g 是模 p^a 的一个原根，则 g 与 $g+p^a$ 中的单数是模 $2p^a$ 的一个原根。

定理 3.4.2.3 模 m 的原根存在的充要条件是 m 等于 2，4，p^a，$2p^a$，其中 p 是单素数。

定理 3.4.2.4 设 $m>1$，$\varphi(m)$的所有不同质因数是 q_1，q_2，…，q_k，$(g, m)=1$，则 g 是模 m 的一个原根的充要条件是

$$g^{\varphi(m)/q_i} \not\equiv 1 (\mathrm{mod}\ m), \quad i=1,2,\cdots,k \tag{3-16}$$

3.4.3 指标及 n 次剩余

在某些特定情况下，模 m 的原根是存在的，本节就在这两种情况下引进指标的概念，并推出它的基本性质。进一步应用指标的性质来研究下列同余式

$$x^n \equiv a (\mathrm{mod}\ m), (a, m)=1 \tag{3-17}$$

有解的条件及解数，并且求出模 m 的原根的个数。

在本节里假定 m 是 p^a 或 $2p^a$，$c=\varphi(m)$，g 是模 m 的一个原根。

定理 3.4.3.1 若 γ 通过模 c 的最小非负完全剩余系，则 g^{γ} 通过模 m 的一个简化剩余系。

利用定理 3.4.3.1 可以对每一个与模 m 互质的数引进指标的概念。指标的概念与对数的概念很相像，而原根相当于对数的底。

定义 设 a 是一整数，若对模 m 的一个原根 g，有一整数 γ 存在使得下式

$$a \equiv g^{\gamma} \pmod{m}, \gamma \geq 0$$

成立，则 γ 叫做以 g 为底的 a 对模 m 的一个指标。

由定义可以看出，一般来说，a 的指标不仅与模 m 有关，而且与原根也有关。由定理 3.4.3.1 可知任一与模 m 互质的整数 a，对于模 m 的任一原根 g 来说，a 的指标是存在的。若$(a, m) \neq 1$，则对模 m 的任一原根 g 来说，a 的指标是不存在的。

定理 3.4.3.2 若 a 是一个与 m 互质的整数，g 是模 m 的一个原根，则对模 m 来说，a 有一个以 g 为底的指标， 并且以 g 为底的 a 对模 m 的一切指标是满足下列条件的一切整数：

$$\gamma \equiv \gamma' \pmod{c}, \gamma \geq 0$$

a 的以 g 为底的指标的 mod c 最小非负剩余记做 $\mathrm{ind}_g a$(或 inda)

定理 3.4.3.3 设 g 是模 m 的一个原根，是一个非负整数，则以 g 为底，对 m 有同一指标 γ 的一切整数是模 m 的一个与模互质的剩余类。

定理 3.4.3.4 若 a_1，a_2，…，a_n 是与 m 互质的 n 个整数，则

$$ind(a_1 a_2 \cdots a_n) \equiv ind(a_1) + inda_2 + \cdots + inda_n \pmod{c}$$

特别地，

$$\mathrm{ind}(a^n) \equiv n\mathrm{ind}a \pmod{c}$$

定理 3.4.3.5 若(n, c)=d，(a, m)=1，则

(i)同余式

$$x^n \equiv a \pmod{m}$$

有解(即 a 是对模 m 的 n 次剩余)的充分必要条件是：$d \mid \mathrm{ind}\ a$；并且在有解的情况下，解数是 d。

(ii)在模 m 的一个简化剩余系中，n 次剩余的个数是 c/d。

推论 a 是对模 m 的 n 次剩余的充分与必要条件是

$$a^{\frac{c}{d}} \equiv 1 \pmod{m},\ d = (n, c)$$

定理 3.4.3.6 若(a, m)=1，则 a 是模 m 的一个原根的充分与必要条件是$(\mathrm{ind}\ a, c)$=1。

在一维素长度 DCT 变换中，可以利用原根的特性把变换核函数内的$(2x+1)k$转换为两个幂函数 $a^{-s}a^{r}$ 相乘的形式，进而把 DCT 变换转换为循环卷积来减少运算复杂度。

3.5 本章小结

本章叙述了一些数论的基础知识，如同余，同余式，原根，指标等，为第四章一维 DCT 快速算法研究和第五章二维 DCT 快速算法研究提供理论基础。在后续章节中，将使用这些理论进行 DCT 快速算法的推导。

第四章 一维 DCT 快速算法研究

本章归纳了两种基－2 DCT 递归分解算法；提出了一种基－q DCT 算法，利用循环卷积，给出了一种十分高效的实现素数长度 DCT 快速算法，根据数论关于数的分解定理，提出了任意复合长度 DCT 算法的设计结构。

4.1 引言

众所周知，1965 年 Cooley－Tukey 提出的快速傅里叶变换(FFT)算法开创了数字信号处理(DSP)的迅速发展，基本 Cooley－TukeyDFT 算法的各种改进也导致了数字器件技术的发展。同样地，1974 年离散余弦变换(DCT)的提出也在 DSP 领域产生了重要推动作用。最初 DCT 算法是建立在 FFT 基础上的，通过把 DCT 转换为 DFT 加以实现。1977 年 Chen、Smith 和 Fralick 利用 DCT 变换矩阵的分解提出了第一个真正的 DCT 快速算法。随后的二十多年里，各种 DCT 算法不断提出，如时域抽取(DIT)算法、频域抽取(DIF)算法、分裂基算法、基于其它变换(DHT、DWT 等)的算法、素因子算法等，这些算法主要从算法的计算复杂性减少和算法结构的简化着手，来提高算法的实现效率。在各种已经存在的 DCT 算法中，大多数都是针对长度为 2 的幂的情况。而在实际应用中，对 DCT 长度的要求往往是多样的。当出现要求的变换长度非 2 的幂时，通常要采用数据加长的方法，先将被变换数据长度增加到某个 2 的幂，然后再利用已有算法进行变换。这样处理虽然可以利用现有算法实现变换，但是，由于数据长度的加长，无疑增加了计算量，从而降低了实现效率。另一方面，由于数据的加长，必然使得变换产生边沿失真，虽然提出了一些克服(减小)失真的方法，但这种失真是固有的。如果直接使用数据长度的快速 DCT 算法，那么，将可以从根本上消除边缘失真。随着 DSP 技术的发展，对处理速度的要求也越来越高，硬件实现和并行处理是实现高速处理的一个有效途径，同时，计算机技术和数字器件的发展也为 DSP 基本部件的硬件实现和并行处理提供了条件。

本章将研究各种可能长度的一维 DCT 快速算法，为后面的二维 DCT 快速算法设计做好准备。本章首先讨论长度为 2 的幂情况，归纳了 DCT－II 的递归分解算法并给出了一种 DCT－III 的递归分解算法。(也作为本章其它长度算法以及下章

二维 DCT 算法的一个组成部分)；第 3 节研究了长度为奇素数幂的一维 DCT 快速算法，论述一种基－q DCT 算法；第 4 节研究了奇素数长度一维 DCT 快速算法，将素长度 DCT 转化为循环卷积计算；第 5 节将研究任意复合长度的情况，阐述了任意长度快速 DCT 算法的设计思想。

4.2 一维 DCT 的基 2 递归分解算法

4.2.1 DCT－II 的基 2 递归分解法

长度为 N 的一维序列$\{x(n)$：$n=0$，1，…，$N-1\}$的 DCT－II 定义为：

$$X(k)=\alpha(k)\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{2N}k(2n+1);\qquad k=0,1,\cdots N-1 \tag{4－1}$$

其中：

$$\alpha(k)=\begin{cases}1/\sqrt{2} & k=0\\ 1 & k\neq 0\end{cases}$$

为正交化因子，为了讨论方便，下面将省略此因子。如果长度 N 为偶数，我们将 DCT 系数按下标的奇偶分两部分处理：

$$\begin{aligned}X(2k)&=\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{2N}(2n+1)2k \qquad k=0,1,\cdots\frac{N}{2}-1\\ X(2k+1)&=\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{2N}(2n+1)(2k+1)\end{aligned} \tag{4－2}$$

对于偶下标部分，设

$$u(n)=x(n)+x(N-1-n)\qquad n=0,1,\cdots\frac{N}{2}-1 \tag{4－3}$$

则：

$$U(k)=X(2k)=\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{2N}(2n+1)2k \qquad k=0,1,\cdots\frac{N}{2}-1 \tag{4－4}$$

参照定义式(4－1)，上式正好为长度 N /2 的 DCT，如果长度 N 是 2 的幂，那么，对上面缩短了一半的 DCT 可继续进行类似的递归处理，直至长度降到 4 或 2。

对于奇下标部分，由于

$$\cos\big((2l+1)\pi\pm\alpha\big)=-\cos\alpha$$

设

$$v'(n)=x(n)-x(N-1-n)\qquad n=0,1,\cdots,\frac{N}{2}-1 \tag{4－5}$$

则(4－2)中的奇下标部分可改写为

$$X(2k+1)=\sum_{n=0}^{\frac{N}{2}-1}v'(n)\cos\frac{\pi}{2N}(2n+1)(2k+1)\quad k=0,1,\cdots,\frac{N}{2}-1 \tag{4－6}$$

这是一个长度为 $N/2$ 的第 4 类 DCT，下面来讨论其算法。

4.2.2 DCT－IV 的基 2 递归分解法

根据三角函数的积化和差性质

$$2\cos\alpha\cos\beta=\cos(\alpha+\beta)+\cos(\alpha-\beta)$$

有下面两个转换式：

$$\cos\frac{\pi}{2N}(2k+1)(2n+1)+\cos\frac{\pi}{2N}(2k-1)(2n+1)$$

$$=2\cos\frac{\pi}{2N}(2n+1)*\cos\frac{\pi}{N}k(2n+1)$$

$$2\cos\frac{\pi}{2N}(2n+1)*\cos\frac{\pi}{2N}(2k+1)(2n+1)$$

$$=\cos\frac{\pi}{N}k(2n+1)+\cos\frac{\pi}{N}(k+1)(2n+1)$$

把这两个转换式分别应用到公式(4－6)中，将构造出相应的算法。

4.2.2.1 算法 1

将

$$\cos\frac{\pi}{2N}(2k+1)(2n+1)+\cos\frac{\pi}{2N}(2k-1)(2n+1)$$

$$=2\cos\frac{\pi}{2N}(2n+1)\cdot\cos\frac{\pi}{N}k(2n+1) \tag{4－7}$$

应用到(4－6)式中，有

$$X(2k+1)+X(2k-1)\quad k=1,\cdots,\frac{N}{2}-1$$

$$=\sum_{n=0}^{\frac{N}{2}-1}\left[2v'(n)\cos\frac{\pi}{2N}(2n+1)\right]\cos\frac{\pi}{N}k(2n+1) \tag{4－8a}$$

$$X(1)=\frac{1}{2}\sum_{n=0}^{\frac{N}{2}-1}\left[2v'(n)\cdot\cos\frac{\pi}{2N}(2n+1)\right] \tag{4－8b}$$

令

$$v(n)=v'(n)\cdot 2\cos\frac{\pi}{2N}(2n+1)\qquad n=0,1,\cdots,\frac{N}{2}-1$$

计算上面这个新的长度为 N/2 的序列的 DCT 系数：

$$V(k)=\sum_{n=0}^{\frac{N}{2}-1}v(n)\cdot\cos\frac{\pi}{N}k(2n+1)\qquad k=0,1,\cdots,\frac{N}{2}-1 \tag{4-9}$$

可以从这些系数构造原序列 DCT 系数的奇下标部分：

$$\begin{aligned}X(1)&=\frac{1}{2}V(0)\\X(3)&=V(1)-X(1)\\X(5)&=V(2)-X(3)\\&\cdots\\X(N-1)&=V(\frac{N}{2}-1)-X(N-3)\end{aligned}$$

用矩阵表示为

$$\begin{bmatrix}X(1)\\X(3)\\X(5)\\\vdots\\X(N-1)\end{bmatrix}=\begin{bmatrix}\frac{1}{2}&&&&\\-\frac{1}{2}&1&&&\\\frac{1}{2}&-1&1&&\\&&&\ddots&\\-\frac{1}{2}&1&-1&\cdots&1\end{bmatrix}\cdot\begin{bmatrix}V(0)\\V(1)\\V(2)\\\vdots\\V(\frac{N}{2}-1)\end{bmatrix} \tag{4-10}$$

现在把奇偶两部分结合起来，用矩阵分解的形式给出算法一的完整表达：

$$\begin{aligned}C(N)=S(N,\frac{N}{2})\cdot&\begin{bmatrix}\mathrm{I}(\frac{N}{2})&\\&\mathrm{T}(\frac{N}{2})\end{bmatrix}\cdot\begin{bmatrix}\mathrm{C}(\frac{N}{2})&\\&\mathrm{C}(\frac{N}{2})\end{bmatrix}\cdot\\&\begin{bmatrix}\mathrm{I}(\frac{N}{2})&\\&\mathrm{M}(\frac{N}{2})\end{bmatrix}\cdot\begin{bmatrix}\mathrm{I}(\frac{N}{2})&\hat{\mathrm{I}}(\frac{N}{2})\\\mathrm{I}(\frac{N}{2})&-\hat{\mathrm{I}}(\frac{N}{2})\end{bmatrix}\end{aligned} \tag{4-11}$$

其中，

C(N)为修正的 N 点 DCT 变换矩阵(省略正交化因子)；

I(N/2)为 N/2 阶单位矩阵；

$\hat{\mathrm{I}}$(N/2)为 N/2 阶反单位矩阵；

S(N，N/2)为 N 阶 2 分换序矩阵，将偶下标数据顺序排于前半部分，奇下标数据顺序排于后半部分。

$$T(\frac{N}{2})=\begin{bmatrix} \frac{1}{2} & & & & \\ -\frac{1}{2} & 1 & & & \\ \frac{1}{2} & -1 & 1 & & \\ & & & \ddots & \\ -\frac{1}{2} & 1 & -1 & \cdots & 1 \end{bmatrix}_{\frac{N}{2}\times\frac{N}{2}}$$

$$M(\frac{N}{2})=\begin{bmatrix} 2\cos\frac{\pi}{2N} & & & & \\ & 2\cos\frac{3\pi}{2N} & & & \\ & & 2\cos\frac{5\pi}{2N} & & \\ & & & \ddots & \\ & & & & 2\cos\frac{(N-1)\pi}{2N} \end{bmatrix}_{\frac{N}{2}\times\frac{N}{2}}$$

现在分析算法一的计算复杂性。根据递归分解的关系，算法所需的计算量为乘法：

$$\mu_m(N)=2\mu_m(\frac{N}{2})+\frac{N}{2}$$

加法：

$$\mu_a(N)=2\mu_a(\frac{N}{2})+N+(\frac{N}{2}-1)$$

如果递归分解可一直进行到2 点 DCT，由于 $\mu_m(2)=1$， $\mu_a(2)=2$，那么，DCT－II 算法的计算复杂性可表示为：

$$\mu_m(N)=\frac{N}{2}\cdot\log_2 N \qquad (4-12a)$$

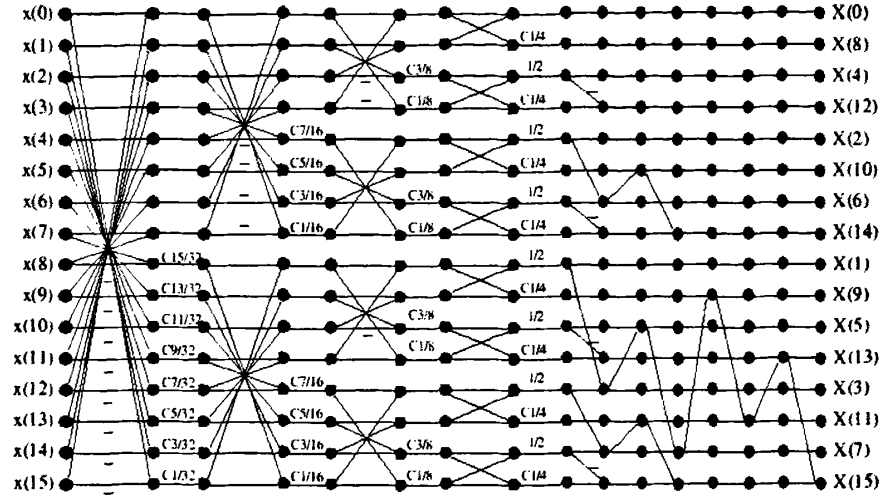
$$\mu_a(N)=\frac{N}{2}\cdot(3\log_2 N-2)+1 \qquad (4-12b)$$

观察图中 DCT－IV 的实现形式，可以发现 N 点 DCT－IV 经过第一级旋转因子相乘处理后可以转换为 N 点 DCT－II 的形式，再加上最后一级迭代运算增加了 N－1 次加法运算，故 DCT－IV 算法的计算复杂性：

$$v_m(N)=N+\frac{N}{2}\cdot\log_2 N \qquad (4-13a)$$

$$v_a(N)=\frac{3N}{2}\cdot\log_2 N \tag{4-13b}$$

图 4-1 给出了 16 点 DCT-II 的算法流程。



$$Cn/M=2\cos\frac{n\pi}{M}$$

图 4-1 算法 1 的 16 点 DCT 信号流程

4.2.2.2 算法 2

利用

$$\begin{aligned}&2\cos\frac{\pi}{2N}(2n+1)*\cos\frac{\pi}{2N}(2k+1)(2n+1)\\&=\cos\frac{\pi}{N}k(2n+1)+\cos\frac{\pi}{N}(k+1)(2n+1)\end{aligned} \tag{4-14}$$

式(4-7)可改写成

$$\begin{aligned}X(2k+1)&=\sum_{n=0}^{\frac{N}{2}-1}v'(n)\cos\frac{\pi}{2N}(2n+1)(2k+1)\quad k=0,1,\cdots,\frac{N}{2}-1\\&=\sum_{n=0}^{\frac{N}{2}-1}\frac{v'(n)}{2\cos\frac{\pi}{2N}(2n+1)}2\cos\frac{\pi}{2N}(2n+1)\cos\frac{\pi}{2N}(2n+1)(2k+1)\end{aligned} \tag{4-15}$$

定义

$$v(n)=\frac{v'(n)}{2\cos\frac{\pi}{2N}(2n+1)} \qquad n=0,1,\cdots,\frac{N}{2}-1 \tag{4-16}$$

利用其 DCT 系数

$$V(k)=\sum_{n=0}^{\frac{N}{2}-1}v(n)\cdot\cos\frac{\pi}{N}k(2n+1) \qquad k=0,1,\cdots,\frac{N}{2}-1 \tag{4-17}$$

可以获得原 DCT 奇下标分量

$$X(2k+1)=V(k)+V(k+1) \qquad k=0,1,\cdots,\frac{N}{2}-1 \tag{4-18}$$

这一关系可写成如下矩阵形式：

$$\begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ \vdots \\ X(N-1) \end{bmatrix}=\begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \\ & & & \cdots & 1 \end{bmatrix}\cdot\begin{bmatrix} V(0) \\ V(1) \\ V(2) \\ \vdots \\ V(\frac{N}{2}-1) \end{bmatrix}$$

类似算法 1，将算法 2 的完整分解写成下列矩阵式：

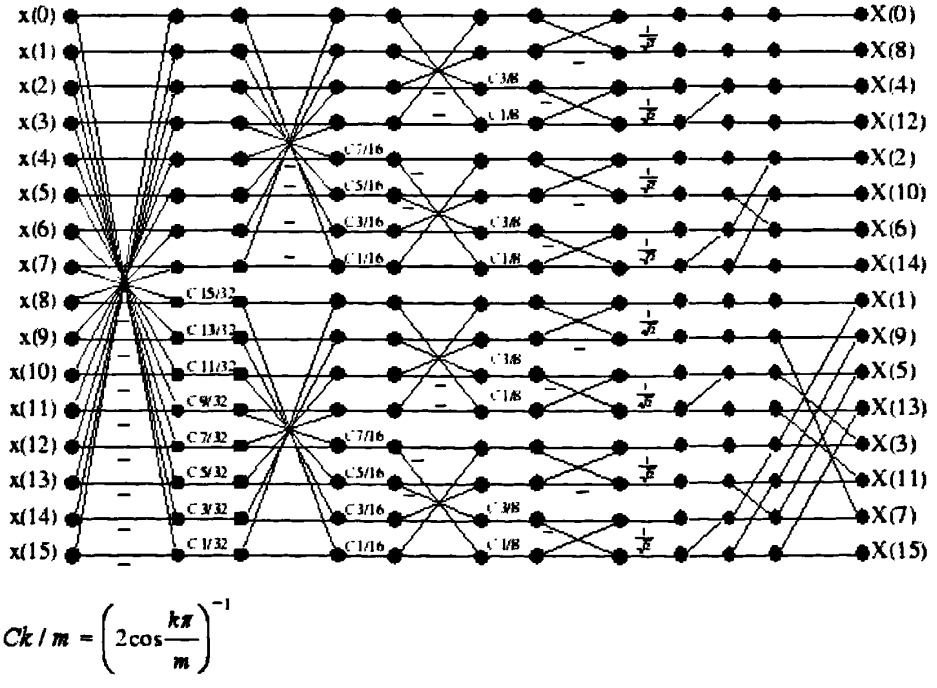
$$\begin{aligned} C(N)=S(N,\frac{N}{2})\cdot&\begin{bmatrix} \mathrm{I}(\frac{N}{2}) & \\ & \mathrm{A}(\frac{N}{2}) \end{bmatrix}\cdot\begin{bmatrix} \mathrm{C}(\frac{N}{2}) & \\ & \mathrm{C}(\frac{N}{2}) \end{bmatrix}\cdot \\ &\begin{bmatrix} \mathrm{I}(\frac{N}{2}) & \\ & R(\frac{N}{2}) \end{bmatrix}\cdot\begin{bmatrix} \mathrm{I}(\frac{N}{2}) & \hat{\mathrm{I}}(\frac{N}{2}) \\ \mathrm{I}(\frac{N}{2}) & -\hat{\mathrm{I}}(\frac{N}{2}) \end{bmatrix} \end{aligned} \tag{4-19}$$

其中，

$$A\left(\frac{N}{2}\right)=\begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \\ & & & \cdots & 1 \end{bmatrix}$$

$$R(\frac{N}{2})=\begin{bmatrix} \frac{1}{2\cos\frac{\pi}{2N}} & & & & \\ & \frac{1}{2\cos\frac{3\pi}{2N}} & & & \\ & & \frac{1}{2\cos\frac{5\pi}{2N}} & & \\ & & & \ddots & \\ & & & & \frac{1}{2\cos\frac{(N-1)\pi}{2N}} \end{bmatrix}\frac{N}{2}\times\frac{N}{2}$$

观察算法 1 和算法 2 的分解，可以看出，两种算法的结构基本相同，具有相同的计算复杂性。算法二中所使用的乘法因子恰为算法一中所使用乘法因子的倒数，当 N 较大时，算法二中出现较大的乘法因子，所以可能引起较大的计算误差。图 4－2 给出了算法 2 的 16 点 DCT 信号流程。



$$Ck/m=\left(2\cos\frac{k\pi}{m}\right)^{-1}$$

图 4－2 算法 2 的 16 点 DCT 信号流程

4.2.3 DCT－III 的基 2 递归分解法

长度为 N 的一维序列$\{x(n): n=0, 1, \cdots, N-1\}$的 DCT－III 变换定义为：

$$X(k)=\alpha(k)\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{2N}n(2k+1); \qquad k=0,1,\cdots N-1 \tag{4－20}$$

其中：

$$\alpha(k)=\begin{cases}1/\sqrt{2} & k=0\\ 1 & k\neq 0\end{cases}$$

为正交化因子，为了讨论方便，下面将省略此因子。如果长度 N 为偶数，我们将序列 $x(n)$按下标的奇偶特性分两部分处理。令：

$$\begin{aligned}A(k)&=\sum_{n=0}^{\frac{N}{2}-1}x(2n)\cos\frac{\pi}{2\frac{N}{2}}n(2k+1); \qquad n=0,1,\cdots\frac{N}{2}-1\\ B(k)&=\sum_{n=0}^{\frac{N}{2}-1}x(2n+1)\cos\frac{\pi}{2N}(2n+1)(2k+1); \quad n=0,1,\cdots\frac{N}{2}-1\end{aligned} \tag{4－21}$$

则：

$$X(k)=A(k)+B(k) \qquad k=0,1,\cdots N-1 \tag{4－22}$$

B(k)是一个长度为 N/2 的第 4 类 DCT，其算法在 4.2.1 中已有阐述，这里不再重复。参照定义式，A(k)恰为长度 N /2 的 DCT－III，如果长度 N 是 2 的幂，那么，对上面缩短了一半的 DCT 可继续进行类似的递归处理，直至长度降到 2。

$$\begin{aligned}A(N-1-k)&=\sum_{n=0}^{\frac{N}{2}-1}x(2n)\cos\frac{\pi}{2\frac{N}{2}}n(2N-2k-1)\\ &=\sum_{n=0}^{\frac{N}{2}-1}x(2n)\cos\frac{\pi}{2\frac{N}{2}}n(2k+1)\\ &=A(k); \qquad k=0,1,\cdots\frac{N}{2}-1\end{aligned} \tag{4－23}$$

$$
\begin{aligned}
B(N-1-k) &= \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\cos\frac{\pi}{2N}(2n+1)(2N-2k-1) \\
&= -\sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\cos\frac{\pi}{2N}(2n+1)(2k+1) \qquad (4-24) \\
&= -B(k); \qquad k = 0,1,\cdots\frac{N}{2}-1
\end{aligned}
$$

将序列 $x(n)=\{x(0)，x(1)，x(n)\}$重新排序，得到新的序列 $y(n)=\{$
$x(0)，x(N/2)，x(N/4)，x(3N/4)，...，x(1)，x(3)，...，x(N-1)\}$。

现在把 A(k)和 B(k)两部分结合起来，用矩阵分解的形式给出算法的完整表达：

$$
C(N) = S(N,\frac{N}{2})\cdot\begin{bmatrix} M(2) & & & \\ & \cdots & & \\ & & M(\frac{N}{4}) & \\ & & & M(\frac{N}{2}) \end{bmatrix}\cdot\begin{bmatrix} C(2) & & & \\ & \cdots & & \\ & & C(\frac{N}{4}) & \\ & & & C(\frac{N}{2}) \end{bmatrix}\cdot \\
\begin{bmatrix} T(2) & & & \\ & \cdots & & \\ & & T(\frac{N}{4}) & \\ & & & T(\frac{N}{2}) \end{bmatrix}\cdot G \qquad (4-25)
$$

其中，

C(N)为修正的 N 点 DCT－II 变换矩阵(省略正交化因子)；

I(N/2)为 N/2 阶单位矩阵；

$\hat{I}$(N/2)为 N/2 阶反单位矩阵；

S(N，N/2)为 N 阶换序矩阵，将偶下标数据顺序排于前半部分，奇下标数据顺序排于后半部分。

$$T(\frac{N}{2})=\begin{bmatrix} \frac{1}{2} & & & & \\ -\frac{1}{2} & 1 & & & \\ \frac{1}{2} & -1 & 1 & & \\ & & & \ddots & \\ -\frac{1}{2} & 1 & -1 & \cdots & 1 \end{bmatrix}_{\frac{N}{2}\times\frac{N}{2}}$$

$$M(\frac{N}{2})=\begin{bmatrix} 2\cos\frac{\pi}{2N} & & & & \\ & 2\cos\frac{3\pi}{2N} & & & \\ & & 2\cos\frac{5\pi}{2N} & & \\ & & & \ddots & \\ & & & & 2\cos\frac{(N-1)\pi}{2N} \end{bmatrix}_{\frac{N}{2}\times\frac{N}{2}}$$

$$G=\begin{bmatrix} I(1) & I(1) & & \\ \hat{I}(1) & -\hat{I}(1) & & \\ & & I(N-2) & \end{bmatrix}\times\cdots\begin{bmatrix} I(\frac{N}{4}) & I(\frac{N}{4}) & & \\ \hat{I}(\frac{N}{4}) & -\hat{I}(\frac{N}{4}) & & \\ & & I(\frac{N}{2}) & \end{bmatrix}\times\begin{bmatrix} I(\frac{N}{2}) & I(\frac{N}{2}) \\ \hat{I}(\frac{N}{2}) & -\hat{I}(\frac{N}{2}) \end{bmatrix}$$

现在分析算法一的计算复杂性。假设N×N点DCT－IV所用的乘法次数$u_m(n)$，加法次数为$u_a(n)$。根据递归分解的关系，基2的DCT－III算法所需的计算量为

乘法：

$$\mu_m(N)=u_m(\frac{N}{2})+u_m(\frac{N}{4})+\cdots u_m(2)+1 \tag{4－26}$$

加法：

$$\mu_a(N)=\left(v_a(\frac{N}{2})+N\right)+\left(v_a(\frac{N}{4})+\frac{N}{2}\right)+\cdots+u_a(2)+2 \tag{4－27}$$

又：

$$u_m(N) = N\left(1+\frac{1}{2}\log_2 N\right)$$

$$u_a(N) = \frac{3N}{2}\cdot\log_2 N$$

如果递归分解可一直进行到 2 点 DCT，由于 $u_m(2)=1$，$u_a(2)=2$。令 $N=2^n$，算法的计算复杂性可表示为：

$$\begin{aligned}\mu_m(N) &= 1+\sum_{k=1}^{n-1}2^k+\sum_{k=1}^{n-1}k\cdot 2^{k-1}\\ &= 2^n-1+\sum_{k=1}^{n-1}\left(k\cdot 2^{k-1}\right)\\ &= 2^n-1+\left.\frac{d\sum_{k=1}^{n-1}x^k}{dx}\right|_{x=2}\\ &= n\cdot 2^{n-1} \qquad n>0\end{aligned} \tag{4-28}$$

$$\begin{aligned}u_a(N) &= \left(v_a(\frac{N}{2})+N\right)+\left(v_a(\frac{N}{4})+\frac{N}{2}\right)+\cdots v_a(2)+2\\ &= 2+\sum_{k=1}^{n-1}\left(v_a(2^{k-1})\right)+\sum_{k=1}^{n-1}2^{k+1}\\ &= 2^{n+1}-2+3\cdot\sum_{k=1}^{n-1}\left(k\cdot 2^{k-1}\right)\\ &= 2^{n+1}-2+3\cdot\left.\frac{d\sum_{k=1}^{n-1}x^k}{dx}\right|_{x=2}\\ &= 2^n+n+\sum_{k=2}^{n-1}\left(3k\cdot 2^{k-1}\right)\\ &= (3n-2)\cdot 2^{n-1}+1 \qquad n>0\end{aligned} \tag{4-29}$$

图 4－3 给出了 16 点 DCT－III 的算法流程。

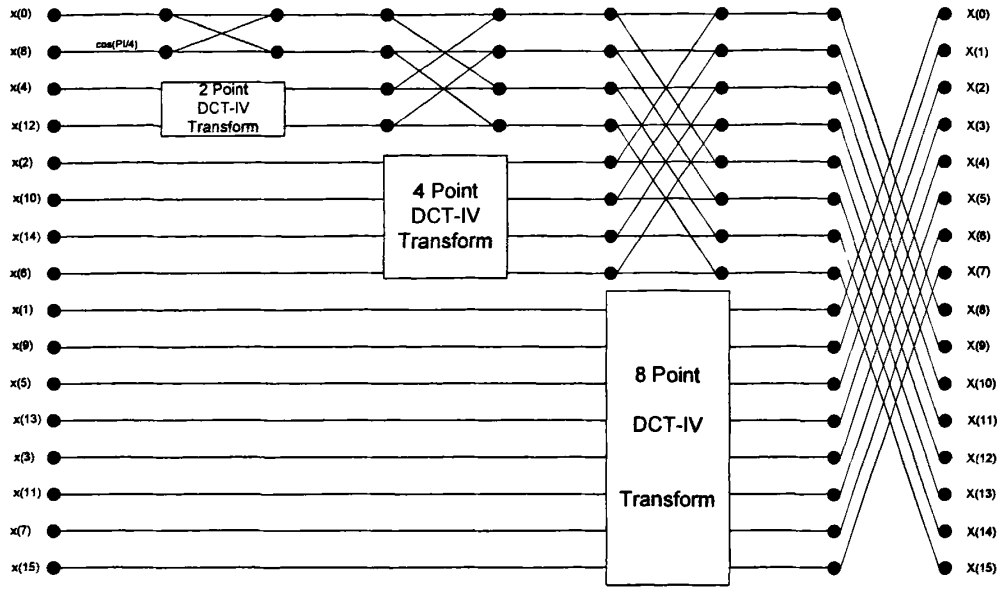


图 4－3 16 点 DCT－III 信号流程

4.3 一维 DCT 的基－q 递归分解算法

当 DCT 长度 $N=q^p$ 时，将 DCT 改写成

$$\begin{aligned} X(k)=&\sum_{n=0}^{\frac{N}{q}-1}x(qn+\frac{q-1}{2})\cdot\cos\frac{\pi(2n+1)}{2(N/q)}k \\ &+\sum_{m=0}^{\frac{q-3}{2}}\sum_{n=0}^{\frac{N}{q}-1}x(qn+m)\cdot\cos\frac{\pi\left(q(2n+1)-(q-1-2m)\right)}{2N}k \\ &+\sum_{m=0}^{\frac{q-3}{2}}\sum_{n=0}^{\frac{N}{q}-1}x(qn+q-1-m)\cdot\cos\frac{\pi\left(q(2n+1)+(q-1-2m)\right)}{2N}k \end{aligned} \qquad (4-30)$$

$$k=0,1,\cdots,N-1$$

为讨论方便，将上式中各项分别记作

$$A(k)=\sum_{n=0}^{\frac{N}{q}-1}x(qn+\frac{q-1}{2})\cdot\cos\frac{\pi(2n+1)}{2(N/q)}k \qquad (4-31a)$$

这是一个 N/q 点 DCT。

$$C_m(k)=\sum_{n=0}^{\frac{N}{q}-1}\left[x(qn+m)+x(qn+q-1-m)\right]\cdot\cos\frac{\pi\left((2n+1)k\right)}{2N/q} \qquad (4-31b)$$

$$S_m(k)=\sum_{n=0}^{\frac{N}{q}-1}\left[x(qn+m)-x(qn+q-1-m)\right]\cdot\sin\frac{\pi\left((2n+1)k\right)}{2N/q} \tag{4-31c}$$

根据三角函数的性质，有

$$S_m(\frac{N}{q}-k)=\sum_{n=0}^{\frac{N}{q}-1}(-1)^n\left[x(qn+m)-x(qn+q-1-m)\right]\cdot\cos\frac{\pi\left((2n+1)k\right)}{2N/q} \tag{4-32}$$

因此，(4－31*b*)和(4－31*c*)实际上包含了 $q-1$ 个 N / q 点 DCT。利用(4－31)可将(4－30)式表示的 N 点 DCT 写成：

$$X(k)=A(k)+\sum_{m=0}^{\frac{q-3}{2}}\left[C_m(k)\cos\frac{\pi(q-1-2m)k}{2N}+S_m(k)\sin\frac{\pi(q-1-2m)k}{2N}\right] \tag{4-33}$$

对(4－31*b*)和(4－31*c*)仔细观察，发现

$$C_m(\frac{jN}{q}\pm k)=\begin{cases}(-1)^{j/2}C_m(k)\\ \pm(-1)^{(j+1)/2}C_m(\frac{N}{q}-k)\end{cases} \tag{4-34}$$

$$S_m(\frac{jN}{q}\pm k)=\begin{cases}\pm(-1)^{j/2}C_m(k)\\ (-1)^{(j+1)/2}C_m(\frac{N}{q}-k)\end{cases} \tag{4-35}$$

利用这一关系，(4－33)式中 cos 因子和 sin 因子可进一步减少。令

$$F_{2j}(k)=\frac{X(2jN/q+k)+X(2jN/q-k)}{2}$$

$$=(-1)^j\left\{A(k)+\sum_{m=0}^{\frac{q-3}{2}}\left[C_m(k)+S_m(k)\tan\frac{\pi(q-1-2m)k}{2N}\right]\cdot\cos\frac{\pi(q-1-2m)k}{2N}\cos\frac{2\pi(q-1-2m)j}{q}\right\}$$

$$j=0,1,\cdots,\frac{q}{2}-1;\ k=0,1,\cdots,N/q-1$$

(4－36a)

$$F_{2j+1}(k)=\frac{X(2jN/q+k)+X(2jN/q-k)}{2}$$

$$=(-1)^j\left\{A(k)+\sum_{m=0}^{\frac{q-3}{2}}\left[C_m(k)+S_m(k)\tan\frac{\pi(q-1-2m)k}{2N}\right]\cdot\cos\frac{2\pi(q-1-2m)k}{2N}\cos\frac{\pi(q-1-2m)(2j+1)}{q}\right\}$$

$$j=0,1,\cdots,\frac{q}{2}-1;\ k=0,1,\cdots,N/q-1$$

(4－36b)

那么，通过 F_j 可计算出 DCT 系数：

$$\begin{aligned} X(k) &= F_0(k) && k = 0,1,2\cdots\frac{N}{q}-1 \\ X(\frac{jN}{q}) &= \frac{F_j(0)}{2} && j = 1,2\cdots\frac{N}{q}-1 \\ X(\frac{jN}{q}+k) &= F_j(k)-F_j(\frac{jN}{q}-k) && j = 1,2\cdots\frac{N}{q}-1 \end{aligned} \tag{4—37}$$

下面分析算法的计算复杂性。

1. 对每个$k\left(1\sim\frac{N}{q}-1\right)$，(4—36a)中方括号内需要$\frac{q-1}{2}$次乘法；(4—36b)方括号内也需要$\frac{q-1}{2}$次乘法；而 k=0 时方括号内无需乘法；因此(4—36)方括号内共需$(q-1)(\frac{N}{q}-1)$次乘法。

2. 对每个$k\left(1\sim\frac{N}{q}-1\right)$，(4—36)方括号外，(4—36a)需要$\frac{q+1}{2}\cdot\frac{q-1}{2}$次乘法，(4—36b)需要$\frac{q-1}{2}\cdot\frac{q-1}{2}$次乘法；共需$N\cdot\frac{q-1}{2}$次乘法。

3. 对每个$k\left(1\sim\frac{N}{q}-1\right)$，(4—36a)需要加法$\frac{q-1}{2}+\frac{q^2-1}{4}$次，(4—36b)需要加法$\frac{q-1}{2}+\frac{(q-1)(q-3)}{4}$次；k=0 时，(4—36a)需要加法$\frac{q^2-1}{4}$次，(4—36b)需要加法$\frac{(q-1)(q-3)}{4}$次；共需要$\frac{N}{q}\cdot\frac{q^2-1}{2}-(q-1)$次加法。

4.(4—31)中输入数据的蝶形运算共需要$\frac{N}{q}\cdot(q-1)$次加法。

5. (4—37)中需要$q-1$次移位(除以 2)和$(\frac{N}{q}-1)\cdot(q-1)$ 次加法。

根据上面的分析，如果以μ_m(N) 表示 N 点 DCT 的乘法复杂性，以μ_a(N)表示 N 点 DCT 的加法复杂性，那么，可得到下面的计算复杂性递归方程：

$$\begin{aligned} \mu_m(N) &= q\cdot\mu_m(\frac{N}{q})+\frac{q^2+q-2}{2}\cdot\frac{N}{q}-(q-1) \\ \mu_a(N) &= q\cdot\mu_a(\frac{N}{q})+\frac{q^2+4q-5}{2}\cdot\frac{N}{q}-2(q-1) \end{aligned} \tag{4—38}$$

解此差分方程，可以得到本算法分解到长度 q 时的计算复杂性：

$$\mu_m(N)=\frac{N}{q}\cdot\mu_m(q)+\frac{(q+2)(q-1)}{2}N\log_q N-\frac{(q+1)}{2}N+1$$
$$\mu_a(N)=\frac{N}{q}\cdot\mu_a(q)+\frac{(q+5)(q-1)}{2}N\log_q N-\frac{q^2+4q-1}{2q}N+2 \qquad (4-39)$$

如果确定了素长度 DCT 的算法(将在下节讲述),则可以给出相应的基$-q$ DCT 算法的计算复杂性。

4.4 素长度 DCT 快速算法

上一节讨论了长度为素数的幂时一维 DCT 快速算法，在算法中通过递归分解最终将 DCT 长度降低到素数长度，本节将讨论素数长度 DCT 的快速算法。由于长度为偶素数 2 的 DCT 只涉及简单的蝶形运算,所以下面主要讨论奇素数的情况。

4.4.1 DCT 系数的分离

设 DCT 长度 N 为奇素数，{x(n)；n = 0，1，…，N−1}为待变换数据序列；{X(k)；k = 0，1，…，N−1}为相应的 DCT 系数序列。长度为 N 的 DCT 定义为：

$$X(k)=\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{2N}k(2n+1);\qquad k=0,1,\cdots N-1 \qquad (4-40)$$

其中，为了讨论方便省略掉正交化因子和归一化因子。记 N_1 = (N−1)/2，下面将把 DCT 系数分成三个部分来讨论。

1. DC 分量

对应于 DCT 的 DC 分量，下标 k=0，根据 DCT 定义(4−1)有

$$X(0)=\sum_{n=0}^{N-1}x(n)=\sum_{n=0}^{N_1-1}\left[x(n)+x(N-1-n)\right]+x(N_1)\qquad k=0,1,\cdots N-1 \qquad (4-41)$$

由(4-41)可知，直流分量只涉及加法运算。

2.偶下标系数

对于除 DC 分量外的偶下标系数，输出下标为 $2k$，k=1，2，…，N_1

$$\begin{aligned}X(2k)&=\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{N}k(2n+1)\\&=(-1)^k x(N_1)+\sum_{n=0}^{N_1-1}\left[x(n)+x(N-1-n)\right]\cdot\cos\frac{\pi}{N}k(2n+1)\end{aligned} \qquad (4-42)$$

定义一个长度为 N_1 的新序列

$$u(n)=x(N_1-n)+x(N_1+n) \qquad n=1,2,\cdots N_1 \tag{4-43}$$

把公式(4-41)、(4-43)和(4-46)代入公式(4-42)中，可得到偶下标 DCT 系数的计算公式

$$\begin{aligned} X(2k)&=\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{N}k(2n+1)\\ &=(-1)^k x(N_1)+\sum_{n=1}^{N_1-1}\left[x(n)+x(N-1-n)\right]\cdot\cos\frac{\pi}{N}k(2n+1)\\ &=(-1)^k x(N_1)+\sum_{n=1}^{N_1-1}\left(x(N_1-n)+x(N_1+n)\right)\cdot\cos\frac{\pi}{N}k(2N_1+1-2n)\\ &=(-1)^k\left\{x(N_1)+\sum_{n=1}^{N_1}u(n)\cdot\cos\frac{2\pi}{N}\,k\cdot n\right\}\\ &k=1,2,\cdots N_1 \end{aligned} \tag{4-44}$$

定义序列 $u(n)$的素长度截断余弦傅里叶变换(PCDFT)为

$$U(k)=\sum_{n=1}^{N_1}u(n)\cdot\cos\frac{2\pi}{N}\,k\cdot n, \qquad k=1,2,\cdots N_1 \tag{4-45}$$

则偶下标 DCT 系数被转化为 PCDFT 和 $x(N_1)$之和：

$$U(k)=\sum_{n=1}^{N_1}u(n)\cdot\cos\frac{2\pi}{N}\,k\cdot n, \qquad k=1,2,\cdots N_1 \tag{4-46}$$

根据序列 $u(n)$的定义，DC 分量的计算公式(4-41)可改写为

$$X(0)=x(N_1)+\sum_{n=1}^{N_1}u(n) \tag{4-47}$$

与公式(4-41)相比，上式计算将节省 N_1 次加法运算。

3.奇下标系数

对于 DCT 的奇下标系数，输出下标为 $2k-1$，$k=1$，2，…，N_1

$$\begin{aligned} X(2k-1)&=\sum_{n=0}^{N-1}x(n)\cos\frac{\pi}{2N}(2k-1)(2n+1)\\ &=\sum_{n=0}^{N_1-1}\left[x(n)-x(N-1-n)\right]\cdot\cos\frac{\pi}{2N}(2k-1)(2n+1) \end{aligned} \tag{4-48}$$

将输出序列的顺序反转，上式变为

$$\begin{aligned} X(N-2k) &= X\left(2\cdot\frac{N+1-2k}{2}-1\right) \\ &= \sum_{n=0}^{N_1-1}\left[x(n)-x(N-1-n)\right]\cdot\cos\left[\frac{2n+1}{2N}(N-2k)\pi\right] \\ &= \sum_{n=0}^{N_1-1}(-1)^n\left[x(n)-x(N-1-n)\right]\cdot\sin\frac{\pi}{N}(2n+1)\cdot k \\ k &= 1,2,\cdots,N_1 \end{aligned} \tag{4-49}$$

定义另一个长度为 N_1 的序列 v

$$v(n)=(-1)^{N_1-n}\left\{x(N_1+n)-x(N_1-n)\right\} \qquad n=1,2,\cdots N_1 \tag{4-50}$$

序列 $v(n)$的素长度截断正弦傅里叶变换(PSDFT)定义为

$$V(k)=\sum_{n=1}^{N_1}v(n)\cdot\sin\frac{2\pi}{N}\,k\cdot n, \qquad k=1,2,\cdots N_1 \tag{4-51}$$

$$\begin{aligned} X(N-2k) &= \sum_{n=0}^{N_1-1}(-1)^n\left[x(n)-x(N-1-n)\right]\cdot\sin\frac{\pi}{N}(2n+1)\cdot k \\ &= \sum_{n=1}^{N_1}(-1)^{N_1-n}\left[x(N_1-n)-x(N_1+n)\right]\cdot\sin\left(\frac{\pi}{N}(2N_1+1-2n)\cdot k\right) \\ &= (-1)^{k+1}\sum_{n=0}^{N_1-1}(-1)^{N_1-n}\left[x(N_1-n)-x(N_1+n)\right]\cdot\sin\left(\frac{2n\pi}{N}\cdot k\right) \\ &= (-1)^{k}\sum_{n=0}^{N_1-1}(-1)^{N_1-n}\left[x(N_1+n)-x(N_1-n)\right]\cdot\sin\left(\frac{2n\pi}{N}\cdot k\right) \end{aligned}$$

因此，DCT 系数的奇下标分量可表示为 PSDFT：

$$\begin{aligned} X(N-2k) &= (-1)^k\sum_{n=1}^{N_1}v(n)\cdot\sin\frac{2\pi}{N}\,k\cdot n=(-1)^kV(k) \\ k &= 1,2,\cdots N_1 \end{aligned} \tag{4-52}$$

上面分析表明，通过 N－1 次蝶形预加法(用于生成序列 u 和 v)，除了 DC 分量，借助公式(4－44)、(4－49)，N 点 DCT 的计算可转化为 PCDFT 和 PSDFT 的计算。

4.4.2 PCDFT 和 PSDFT 的卷积实现

本节讨论用循环卷积实现 PCDFT 和 PSDFT 的方法。

定义 4－1. 设 N 为奇素数，N_1=(N－1)/2。对于任意整数 n，如果整数 $k\leqslant N_1$ 满足

$$k=\begin{cases} k \bmod N; & k \bmod N\leq N_1 \\ N-(k \bmod N); & k \bmod N > N_1 \end{cases} \tag{4-53}$$

称 k 为 n 在 N 下的绝对值模，记为 $k=amod(n,\ N)$。其中 $mod(n,\ N)$为通常意

义下的模 N 运算。

显然，如果 $k_1 = amod(n_1, N)$，$k_2 = amod(n_2, N)$，那么有

$$a\,mod\,(k_1 \cdot k_2, N) = a\,mod\,(n_1 \cdot n_2, N) \tag{4-54}$$

定理 4-1. 设 N 为奇素数，$\Re = \{1, 2, \cdots, N_1\}$为一个整数集合。则在绝对值模乘法运算下 $\Re$ 构成一个群，并存在一个生成元 a，使得对任意 $n \in R$，在 $\Re$ 中存在唯一对应的元素 $k = amod(a_n, N)$，即 $amod(a_n, N)$，$n=1, 2, \cdots, N_1$ 构成 $\Re \to \Re$ 的一一映射。

$$\Re = \left\{ a\,mod\left(a^n, N\right), N \in \Re \right\} \tag{4-55a}$$

显然，$\Re$ 的生成元是不唯一的，事实上，如果 α 是一个生成元，则 $amod(a^{-1}, N)$也是 $\Re$ 的生成元：

$$\Re = \left\{ a\,mod\left(a^{-n}, N\right), N \in \Re \right\} \tag{4-55b}$$

现在利用定理 4-1 将 PCDFT 转化为循环卷积，在 PCDFT 定义中作下标变换：

$$\begin{aligned} k &= a\,mod\left(a^j, N\right) \qquad j = 1, 2, \cdots N_1 \\ n &= a\,mod\left(a^{-i}, N\right) \qquad i = 1, 2, \cdots N_1 \end{aligned} \tag{4-56}$$

由于

$$\cos\frac{2\pi}{N} n \equiv \cos\left\{ \frac{2\pi}{N} amod(n, N) \right\}$$

因此，PCDFT 变为

$$\begin{aligned} U\left(a\,mod(a^j, N)\right) &= \sum_{i=1}^{N_1} u\left(a\,mod(a^{-i}, N)\right) \cdot \cos\left\{ \frac{2\pi}{N} \cdot a\,mod(a^{j-i}, N) \right\} \\ j &= 1, 2, \cdots, N_1 \end{aligned} \tag{4-57}$$

为简单起见，记

$$\begin{aligned} \hat{u}(i) &= u\left(a\,mod(a^{-i}, N)\right) \\ \widehat{U(j)} &= U\left(a\,mod(a^j, N)\right) \\ C_m &= \cos\left\{ \frac{2\pi}{N} \cdot a\,mod(a^m, N) \right\} \end{aligned}$$

则(4-48)式可重写为

$$\hat{U}(j) = \sum_{i=1}^{N_1} \hat{u}(i) \cdot C(j-i), \qquad j = 1, 2, \cdots, N_1 \tag{4-58}$$

上式说明，经过适当排序(下标变换)，截断余弦傅里叶变换 PCDFT 可转化为一个同长度的循环卷积。下面讨论截断正弦傅里叶变换 PSDFT。

定义 4-2 设 N 为奇素数，$N_1 = (N-1)/2$。对于任意整数 n，如果整数 $k \leqslant N_1$ 满足

$$k=\begin{cases} k \bmod N; & k \bmod N \le N_1 \\ (k \bmod N)-N; & k \bmod N > N_1 \end{cases} \tag{4-59}$$

称 k 为 n 在 N 下的带符号模，记为 $k=smod(n,\ N)$。其中 mod(n，N)为通常意义下的模 N 运算。

显然，如果 $k_1=smod(n_1,\ N)$，$k_2=smod(n_2,\ N)$，那么有

$$smod(k_1\cdot k_2, N)=smod(n_1\cdot n_2, N) \tag{4-60}$$

对于截断正弦傅里叶变换，注意到

$$V(-k)=-\sum_{n=1}^{N_1} v(n)\cdot \sin\frac{2\pi}{N}\, n\cdot k, \qquad k=1,2,\cdots N_1$$

假设 $v(-n)=-v(n)$和 V($-k$)= $-$V(k)，如果 a 是 $\Re$ 的一个生成元，令

$$k=smod\left(a^j,N\right) \qquad j=1,2,\cdots N_1$$

$$n=smod\left(a^{-i},N\right) \qquad i=1,2,\cdots N_1$$

那么，PSDFT 变成

$$V\left(smod(a^j,N)\right)=\sum_{i=1}^{N_1} v\left(smod(a^{-i},N)\right)\cdot \sin\left\{\frac{2\pi}{N}\cdot smod(a^{j-i},N)\right\} \tag{4-61}$$

$$j=1,2,\cdots,N_1$$

定义 4－3 设 a 是 $\Re$ 的一个生成元，定义模符号函数如下：

$$s_i=\begin{cases} 1 & mod\,(a^i,N)\le N_1 \\ -1 & mod\,(a^i,N)>N_1 \end{cases}$$

根据定义 4－1~3，有

$$smod\left(a^i,N\right)=s_i\cdot amod\left(a^i,N\right) \tag{4-62}$$

把这一关系代入公式(4－47)，有

$$s_j\cdot V\left(amod(a^j,N)\right)=\sum_{i=1}^{N_1} s_{-i}\cdot v\left(amod(a^{-i},N)\right)\cdot s_{j-i}\sin\left\{\frac{2\pi}{N}\cdot amod(a^{j-i},N)\right\}$$

$$j=1,2,\cdots,N_1$$

对截断傅里叶变换的输入输出序列作下标变换符号校正：

$$\hat{v}(i)=s_{-i}\cdot v\left(amod(a^{-i},N)\right)$$

$$\hat{V}(j)=V\left(s_j\cdot amod(a^j,N)\right)$$

并记

$$S(m)=\sin\left\{\frac{2\pi}{N}\cdot amod(a^m,N)\right\}$$

则有

$$\hat{V}(j)=\sum_{i=1}^{N_1} s_{j-i}\cdot\hat{v}(i)\cdot S(j-i), \qquad j=1,2,\cdots,N_1 \tag{4-63}$$

定理 2－2 设 N 为奇素数，$N_1=(N-1)/2$。$\Re=\{1，2，\cdots，N1\}$在绝对值模乘法运算下的群，生成元为 a。则在带符号模运算下，下式成立：

$$smod\left(a^{\pm N_1},N\right)=\begin{cases}1 & N_1\text{为奇素数}\\ -1 & \text{其他}\end{cases} \tag{4－64}$$

根据定理 4－2，公式(2－49)中的符号函数 s_{j-i} 具有下面性质

$$S_{N_1-n}=\begin{cases}S_{-n} & N_1\text{为奇素数}\\ -S_{-n} & \text{其他}\end{cases} \tag{4－65}$$

因此，对于奇素数的 N_1，公式(4－63)构成一个循环卷积；而对于非奇素数的 N_1，公式(4－63)构成一个扭循环卷积。

4.5 复合长度 DCT 的快速算法

当 DCT 长度为任意合数 N 时，根据定理 3.2.2，该合数可以分解成如下形式：

$$N=2^{r_0}\times q_1^{r_1}\times\cdots q_n^{r_n}$$

其中，q_i 为奇素数，r_i 为正整数。

将 DCT 作如下分解：

1、如果 $r_0>0$，则按照本章第二节的方法进行奇偶分离处理，直至 $r_0=0$。

2、对每个奇素数 q_i，按照本章第三节的方法进行基 q_i 分解，直至 $r_i=1$。

通过上述分解，最后得到长度为素数乘积的 DCT，采用本节设计素因子算法对素因子长度 DCT 进行计算。

不失一般性，下面的讨论中设 $N=N_1\times N_2$(N_1、N_2 为互不相等的奇素数)，这时 DCT 定义为

$$X(k)=\sum_{n=0}^{N_1N_2-1}x(n)\cdot\cos\frac{\pi(2n+1)k}{2N_1N_2} \tag{4－66}$$

由于 N_1 和 N_2 互素，因此有

$$\begin{aligned}X(k_1N_2+\mu k_2N_1)&=\sum_{n=0}^{N-1}x(n)\cdot\cos\frac{\pi(2n+1)(k_1N_2+\mu k_2N_1)}{2N}\\&=A(k_1,k_2)-B(k_1,\mu k_2)\end{aligned} \tag{4－67}$$

$$k_1=0,1,\cdots,N_1-1;\quad k_2=0,1,\cdots,N_2-1$$

这里定义了两个二维新序列

$$A(k_1,k_2)=\sum_{n=0}^{N-1}x(n)\cdot\cos\frac{\pi(2n+1)k_1}{2N_1}\cos\frac{\pi(2n+1)k_2}{2N_2}$$
$$B(k_1,\mu k_2)=\sum_{n=0}^{N-1}x(n)\cdot\sin\frac{\pi(2n+1)k_1}{2N_1}\sin\frac{\pi(2n+1)k_2}{2N_2} \qquad (4-68)$$
$$k_1=0,1,\cdots,N_1-1;\quad k_2=0,1,\cdots,N_2-1$$

分析上面两式可知

$$\begin{aligned}&B(k_1,k_2)=\mu A(N_1-k_1,N_2-k_2)\\&B(0,l)=0\end{aligned} \qquad (4-69)$$

因此，只需计算

设

$$y(n_1,n_2)=\begin{cases}x(2n_1N_2+n_2) & n_1\geq 0\\ x(-2n_1N_2-n_2-1) & n_1<0\end{cases} \qquad (4-70)$$

那么，将得到：

$$A(k_1,k_2)=\sum_{n2=0}^{N2-1}\sum_{n=0}^{N-1}y(n_1,n_2)\cos\frac{\pi(4n_1N_2+2n_2+1)k_1}{2N_1}\cos\frac{\pi(2n_2+1)k_2}{2N_2} \qquad (4-71)$$
$$k_1=0,1,\cdots,N_1-1;\quad k_2=0,1,\cdots,N_2-1$$

上式中对于每个 n_2，定义一个下标映射 $n_1\rightarrow s$

$$2s+1=\begin{cases}\left|\left|4n_1N_2+2n_2+1\right|-4N_1\right| & \left|4n_1N_2+2n_2+1\right|>2N_1\\ \left|4n_1N_2+2n_2+1\right| & \left|4n_1N_2+2n_2+1\right|<2N_1\end{cases} \qquad (4-72)$$

这样，将(4－73)转变成

$$A(k_1,k_2)=\sum_{n2=0}^{N2-1}\left(\sum_{n=0}^{N1-1}y'(l,n_2)\cos\frac{\pi(2l+1)k_1}{2N_1}\right)\cos\frac{\pi(2n_2+1)k_2}{2N_2} \qquad (4-73)$$
$$k_1=0,1,\cdots,N_1-1;\quad k_2=0,1,\cdots,N_2-1$$

至此，一维 DCT 已经被转换成二维 DCT，括号中为 N_1 点 DCT(共需进行 N_2 次)，而括号外则为 N_2 点 DCT(共需进行 N_1+N_2 次)。

现在分析算法的计算复杂性。根据上面算法的推导，乘法运算只包含在二维 DCT 中，因此，算法的乘法复杂性可表示为

$$\mu_m(N)=N_1\cdot\mu_m(N_2)+N_2\cdot\mu_m(N_1) \qquad (4-74)$$

对于加法复杂性，在算法中除了二维 DCT 外，在(4－73)中还包含有$(N_1-1)(N_2-1)$次加法。因此，算法的加法复杂性可表示为

$$\mu_a(N)=N_1\cdot\mu_a(N_2)+N_2\cdot\mu_a(N_1)+(N_1-1)(N_2-1) \qquad (4-75)$$

4.6 本章小结

本章研究和归纳了长度为 2 的幂时一维 DCT 的递归分解算法，并针对应用中对 DCT 长度的多样性要求，研究了各种长度情况下 DCT 的快速算法。首先讨论了基$-q$ DCT 算法，算法具有简单结构和较低的计算复杂性。在基$-q$ DCT 算法中需要使用素长度 DCT 的快速算法作为其核心。考虑到循环卷积在实现上的高效率，叙述了一种素长度 DCT 算法，在将奇偶下标系数分离后，经过蝶形数据预处理，DCT 转化为截断离散余弦傅里叶变换(PCDFT)和截断离散正弦傅里叶变换(PSDFT)，通过自定义的下标映射，PCDFT 被转化循环卷积，而 PSDFT 则被转化为循环卷积或扭循环卷积。本章最后对素因子长度 DCT 的算法进行了研究，将一维数据映射到二维实现。利用基-2 算法、基$-q$ 算法、素因子算法和素长度 DCT 算法，提出了任意复合长度 DCT 算法的实现结构。

第五章 基于部分和分解的二维 DCT 快速算法研究

本章提出了一种直接分解的二维 DCT 算法－部分和分解算法。利用部分和相同的原则，设计了一种高效的 $2^n \times 2^n$ 型二维 DCT 算法和 $q \times q$(q 为奇素数)型二维 DCT 算法，该算法的计算复杂性很低，可作为各种矩形 DCT 法递归分解的核心模块。

5.1 引言

上一章研究了各种长度的一维 DCT(1D－DCT)快速算法，而图像和视频信号处理研究和应用要求设计高效的二维 DCT(2D－DCT)快速算法，本章将研究二维情况下 DCT 快速算法的设计和研究。

由前面章节可知，2D－DCT 是一种可分离变换，因此，利用 1D－DCT 快速算法，各维分别变换即可实现 2D－DCT，这种方法称为行列法，由于其简单的结构而被广泛采用。但是，行列法完全没有考虑二维情况两个维之间的关系，因此所需要的计算量相对比较大。

通常采用降维的办法使二维 DCT 转化为一维 DCT。利用三角函数的积化和差变换，能将原有的变换直接转变到一维平面上来实现。在将 DCT 变换定义式转换到一维平面后，计算工作量主要转移到了加法的运算次数上来。

5.2 节首先提出了部分和的概念，针对 $2^n \times 2^n$ 型二维 DCT 算法，设计了一种计算复杂性很低的部分和分解快速算法。将二维 DCT 变换输出转换为若干个部分和和余弦因子乘加的形式，根据部分和相同的原则对二维 DCT 变换输出的二维序列进行分集。分集内的元素都可以通过若干个部分和的一维 DCT－III 或 DCT－IV 变换求得。然后出了 DCT 变换输出的频域分集准则，研究了部分和的计算方法，部分和中公共加法表达式的合并准则。本节最后给出了部分和分解算法的计算复杂度并和其他几种高效算法进行了对比。

本章第 3 节设计了一种素数尺寸 2D－DCT 的部分和分解快速算法；将 2D－DCT 变换转换为若干个交流分量之和。针对其中最主要的一个交流分量，给出了素数尺寸下的部分和分集准则及分解形式，根据同余方程根理论给出了部分和的求解方法；最后讨论了该种情况下部分和分解算法的计算复杂度。

5.2 $2^n \times 2^n$ 型 DCT 变换的部分和分解算法

5.2.1 部分和定义

二维 DCT 的定义为：

$$X(k,l)=\frac{2c(k)c(l)}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}x(m,n)\cos\left(\frac{(2m+1)k\pi}{2M}\right)\cos\left(\frac{(2n+1)l\pi}{2N}\right)$$
$$(k=0,1,...,M-1;\ \ l=0,1,...N-1) \tag{5-1}$$
$$c(k)=\begin{cases}1/\sqrt{2} & k=0\\ 1 & k=1,2,...N-1\end{cases}$$

设 $M=N=2^n$，且忽略常数因子 $c(k)$。

$$\begin{aligned}X(k,l)&=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cos\left(\frac{(2m+1)k}{4N}2\pi\right)\cos\left(\frac{(2n+1)l}{4N}2\pi\right)\\&=\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(i,j)*\cos\left(\frac{\left((2m+1)k-(2n+1)l\right)}{4N}2\pi\right)\\&\quad+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(i,j)*\cos\left(\frac{\left((2m+1)k+(2n+1)l\right)}{4N}2\pi\right)\end{aligned} \tag{5-2}$$

令：

$$\Omega(x)_N=amod(x,2N)=\begin{cases}x \bmod 2N; & x \bmod 2N<N\\ 2N-(x \bmod 2N); & N\le x \bmod 2N<2N\end{cases} \tag{5-3}$$

$$\begin{aligned}X(k,l)=&\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{amod\left((2m+1)k-(2n+1)l,2N\right)}{4N}2\pi\right)\\&+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{amod\left((2m+1)k+(2n+1)l,2N\right)}{4N}2\pi\right)\end{aligned} \tag{5-4}$$

合并(5－4)中余弦系数相同的项，(5－2)可以最终转化为

$$X(k,l)=\begin{cases}\sum_{t=0}^{N-1}S(k,l,t)*\cos\left(\frac{2t}{4N}\cdot 2\pi\right); & k\equiv l(\bmod 2)\\ \sum_{t=0}^{N-1}S(k,l,t)*\cos\left(\frac{2t+1}{4N}\cdot 2\pi\right); & k\not\equiv l(\bmod 2)\end{cases} \tag{5-5}$$

在(5－4)中

$$S(k,l,t)=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}a(m,n,k,l,t)x(m,n);\ \ a(m,n,k,l,t)=0,1 \tag{5-6}$$

以下称 $S(k,\ n,\ t)$为部分和。

5.2.2 基于部分和的(k，l)集合划分

设 $k=1$，由式(5－5) 可知：

$$X(1,l)=\begin{cases}\sum_{t=0}^{N-1}S(1,l,t)*\cos\left(\frac{2t}{4N}2\pi\right); & l\equiv 1(\bmod 2)\\ \sum_{t=0}^{N-1}S(1,l,t)*\cos\left(\frac{(2t+1)}{4N}2\pi\right); & l\equiv 0(\bmod 2)\end{cases} \tag{5－7}$$

$$X(l,1)=\begin{cases}\sum_{t=0}^{N-1}S(l,1,t)*\cos\left(\frac{2t}{4N}2\pi\right); & l\equiv 1(\bmod 2)\\ \sum_{t=0}^{N-1}S(l,1,t)*\cos\left(\frac{(2t+1)}{4N}2\pi\right); & l\equiv 0(\bmod 2)\end{cases} \tag{5－8}$$

$$\begin{aligned}X(2r+1,(2r+1)l)&=\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega\left((2m+1)(2r+1)-(2n+1)(2r+1)l\right)_N}{4N}2\pi\right)\\&\quad+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega\left((2m+1)(2r+1)+(2n+1)(2r+1)l\right)_N}{4N}2\pi\right)\\&=\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega\left((2r+1)\left((2m+1)-(2n+1)l\right)\right)_N}{4N}2\pi\right)\\&\quad+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega\left((2r+1)\left((2m+1)+(2n+1)l\right)\right)_N}{4N}2\pi\right)\\&=\begin{cases}\sum_{t=0}^{N-1}S(1,l,t)*\cos\left(\frac{2t(2r+1)}{4N}\cdot 2\pi\right); & k\equiv l(\bmod 2)\\ \sum_{t=0}^{N-1}S(1,l,t)*\cos\left(\frac{(2t+1)(2r+1)}{4N}\cdot 2\pi\right); & k\not\equiv l(\bmod 2)\end{cases}\end{aligned}$$

$$S(1,l,t)=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}a(m,n,1,l,t)x(m,n);\quad a(m,n,1,l,t)=0,1 \tag{5－9}$$

$$
\begin{aligned}
X((2r+1)l,2r+1)=&\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega_{2N}\left((2m+1)(2r+1)l-(2n+1)(2r+1)\right)}{4N}2\pi\right)\\
&+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega_{2N}\left((2m+1)(2r+1)l+(2n+1)(2r+1)\right)}{4N}2\pi\right)\\
=&\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega_{2N}\left((2r+1)\left((2m+1)l-(2n+1)\right)\right)}{4N}2\pi\right)\\
&+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left(\frac{\Omega_{2N}\left((2r+1)\left((2m+1)l+(2n+1)\right)\right)}{4N}2\pi\right)\\
=&\begin{cases}\sum_{t=0}^{N-1}S(l,1,t)*\cos\left(\frac{2t(2r+1)}{4N}\cdot 2\pi\right); & k\equiv l(\mathrm{mod}2)\\ \sum_{t=0}^{N-1}S(l,1,t)*\cos\left(\frac{(2t+1)(2r+1)}{4N}\cdot 2\pi\right); & k\not\equiv l(\mathrm{mod}2)\end{cases}\\
S(1,l,t)=&\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}a(m,n,l,1,t)x(m,n);\ \ a(m,n,l,1,t)=0,1
\end{aligned}
\tag{5-10}
$$

由式(5－9)和(5－10)知，$X(2r+1,(2r+1)l)$有相同的部分和，$X((2r+1)l,(2r+1))$有相同的部分和。

$$
\begin{aligned}
X(2r+1,\Omega_{2N}((2r+1)l)_N)=&\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}x(m,n)\cos\left\{\frac{(2m+1)(2r+1)\pi}{2M}\right\}*\cos\left\{\frac{\left((2n+1)(2r+1)\cdot l-k\cdot 2N\right)\pi}{2N}\right\}\\
=&(-1)^k\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)*\cos\left\{\frac{(2m+1)(2r+1)\pi}{2M}\right\}*\cos\left\{\frac{(2n+1)(2r+1)\cdot l\pi}{2N}\right\}\\
=&\begin{cases}(-1)^k\sum_{t=0}^{N-1}S(1,l,t)*\cos\left(\frac{2t\cdot(2r+1)}{4N}\cdot 2\pi\right); & k\equiv l(\mathrm{mod}2)\\ (-1)^k\sum_{t=0}^{N-1}S(k,l,t)*\cos\left(\frac{(2t+1)\cdot(2r+1)}{4N}\cdot 2\pi\right); & k\not\equiv l(\mathrm{mod}2)\end{cases}
\end{aligned}
\tag{5-11}
$$

由(5－11)可知，$X((2r+1),\Omega((2r+1)l)_N)$可表示为 N 个部分和的一维 DCT－III 或 DCT－IV 变换输出。

令

$$k=2^u(2p+1),l=2^v(2q+1)$$

由(5－8)和(5－10)的推导可知 $X(2^u(2p+1),2^{u+v}(2p+1)(2q+1))$ 具有相同的部分和，如果把具有相同部分和的(k,l)划分为一个子集，根据 k，l 的奇偶性质，可以得到 N×N 点 DCT 的子集划分。

$$\left\{X(0,0)\right\},\left\{X\left(0,\frac{N}{2}\right)\right\},\left\{X\left(\frac{N}{2},0\right)\right\},\left\{X\left(\frac{N}{2},\frac{N}{2}\right)\right\}$$

$$\left\{X\left(0,\frac{N}{4}\right),X\left(0,\frac{3N}{4}\right)\right\},\left\{X\left(\frac{N}{2},\frac{N}{4}\right),X\left(\frac{N}{2},\frac{3N}{4}\right)\right\}$$

$$\left\{X\left(\frac{N}{4},0\right),X\left(\frac{3N}{4},0\right)\right\},\left\{X\left(\frac{N}{4},\frac{N}{2}\right),X\left(\frac{3N}{4},\frac{N}{2}\right)\right\}$$

$$\left\{X\left(\frac{N}{4},\frac{N}{4}\right),X\left(\frac{3N}{4},\frac{3N}{4}\right)\right\},\left\{X\left(\frac{N}{4},\frac{3N}{4}\right),X\left(\frac{3N}{4},\frac{N}{4}\right)\right\}$$

$$\cdots$$

$$\left\{X\left(\Omega(2(2l+1))_N,\Omega(4k)_N\right),X\left(\Omega(6(2l+1))_N,\Omega(12k)\right),\cdots X\left(\Omega\left((\frac{N}{2}-1)2(2l+1)\right)_N,\Omega\left((\frac{N}{2}-1)4k\right)_N\right)\right\}$$

$$\left\{X\left(\Omega(4l)_N,\Omega(2(2k+1))_N\right),X\left(\Omega(12l)_N,\Omega(6(2k+1))_N\right),\cdots X\left(\Omega\left((\frac{N}{2}-1)4l\right)_N,\Omega\left((\frac{N}{2}-1)2(2k+1)\right)_N\right)\right\}$$

$$\left\{X\left(\Omega(2(2l+1))_N,\Omega(2(2k+1))_N\right),X\left(\Omega(6(2l+1))_N,\Omega(6(2k+1))_N\right),\right.$$
$$\left.\cdots X\left(\Omega\left((\frac{N}{2}-1)2(2l+1)\right)_N,\Omega\left((\frac{N}{2}-1)2(2k+1)\right)_N\right)\right\}$$

$$\left\{X\left(2l+1,\Omega(2k)_N\right),X\left(\Omega(3(2l+1))_N,\Omega(6k)_N\right),\cdots X\left(\Omega((N-1)(2l+1))_N,\Omega((N-1)(2k))_N\right)\right\}$$

$$\left\{X\left(\Omega(2l)_N,2k+1\right),X\left(\Omega(6l)_N,\Omega(3(2k+1))_N\right),\cdots X\left(\Omega((N-1)(2k))_N,\Omega((N-1)(2k+1))_N\right)\right\}$$

$$\left\{X\left(2l+1,2k+1\right),X\left(\Omega(3(2l+1))_N,\Omega(3(2k+1))_N\right),\cdots X\left(\Omega((N-1)(2l+1))_N,\Omega((N-1)(2k+1))_N\right)\right\}$$

以 8×8 点 DCT 为例，其子集划分为：

$\{X(0,0)\},\{X(0,4),\{X(4,0)\},\{X(4,4)\}$, $\{X(2,0),X(6,0)\},\{X(0,2),X(0,6)\}$,
$\{X(2,4),X(6,4)\}$, $\{X(4,2),X(4,6)\}$, $\{X(2,2),X(6,6)\}$, $\{X(2,6),X(6,2)\}$,
$\{X(1,0),X(3,0),X(5,0),X(7,0)\}$, $\{X(0,1),X(0,3),X(0,5),X(0,7)\}$,
$\{X(1,2),X(3,6),X(5,6),X(7,2)\}$, $\{X(2,1),X(6,3),X(6,5),X(2,7)\}$,
$\{X(1,4),X(3,4),X(5,4),X(7,4)\}$, $\{X(4,1),X(4,3),X(4,5),X(4,7)\}$,
$\{X(1,6),X(3,2),X(5,2),X(7,6)\}$, $\{X(6,1),X(2,3),X(2,5),X(6,7)\}$,
$\{X(1,1),X(3,3),X(5,5),X(7,7)\}$, $\{X(1,3),X(3,7),X(5,1),X(7,5)\}$,
$\{X(1,5),X(3,1),X(5,7),X(7,3)\}$, $\{X(1,7),X(3,5),X(5,3),X(7,1)\}$

5.2.3 部分和特性研究

令：

$$k=2^u(2p+1), l=2^v(2q+1),$$
$$0\le u,v<\log_2 N, 0\le p<\frac{N}{2^{u+1}},\ 0\le q<\frac{N}{2^{v+1}} \tag{5-12}$$

由(5-2),

$$\begin{aligned}X((2p+1)2^u,(2q+1)2^v)&=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cdot\cos\left(\frac{(2m+1)\cdot(2p+1)\cdot 2^u\cdot\pi}{2N}\right)\cdot\cos\left(\frac{(2n+1)\cdot(2q+1)\cdot 2^v\cdot\pi}{2N}\right)\\&=\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cdot\cos\left(\frac{(2m+1)\cdot(2p+1)\cdot 2^u\cdot\pi+(2n+1)\cdot(2q+1)\cdot 2^v\cdot\pi}{2N}\right)+\\&\quad\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cdot\cos\left(\frac{(2m+1)\cdot(2p+1)\cdot 2^u\cdot\pi-(2n+1)\cdot(2q+1)\cdot 2^v\cdot\pi}{2N}\right)\end{aligned} \tag{5-13}$$

令:

$$\begin{aligned}f(m,n)&=\cos\left(\frac{(2m+1)(2p+1)2^u\pi}{2N}\right)\cos\left(\frac{(2n+1)(2q+1)2^v\pi}{2N}\right)\\g(m,n)&=\cos\left(\frac{(2m+1)(2p+1)2^u\pi-(2n+1)(2q+1)2^v\pi}{2N}\right)\\h(m,n)&=\cos\left(\frac{(2m+1)(2p+1)2^u\pi+(2n+1)(2q+1)2^v\pi}{2N}\right)\end{aligned} \tag{5-14}$$

则:

$$\begin{aligned}f(m,n)&=\frac{1}{2}g(m,n)+\frac{1}{2}h(m,n)\\X((2p+1)2^u,(2q+1)2^v)&=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cdot f(m,n)\\&=\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cdot g(m,n)+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cdot h(m,n)\end{aligned} \tag{5-15}$$

$$\begin{aligned}f\left(m+r\cdot\frac{N}{2^u},n+s\cdot\frac{N}{2^v}\right)&=\cos\left(\frac{(2m+1)(2p+1)\cdot 2^u\pi}{2N}\right)\cdot\cos\left(\frac{(2p+1)\cdot r\cdot 2^{u+1}\cdot\frac{N}{2^u}\cdot\pi}{2N}\right)\cdot\\&\quad\cos\left(\frac{(2n+1)(2q+1)\cdot 2^v\cdot\pi}{2N}\right)\cdot\cos\left(\frac{(2q+1)\cdot s\cdot 2^{v+1}\cdot\frac{N}{2^v}\cdot\pi}{2N}\right)\\&=(-1)^{r+s}f(m,n)\qquad 0\le r\le 2^{u-1},0\le s\le 2^{v-1}\end{aligned} \tag{5-16}$$

$$f\left(m+r\cdot\frac{N}{2^u},s\cdot\frac{N}{2^v}-1-n\right)=\cos\left(\frac{(2m+1)(2p+1)\cdot 2^u\pi}{2N}\right)\cdot\cos\left(\frac{(2p+1)\cdot r\cdot 2^{u+1}\cdot\frac{N}{2^u}\cdot\pi}{2N}\right)\cdot$$

$$\cos\left(\frac{(2n+1)(2q+1)\cdot 2^v\cdot\pi}{2N}\right)\cdot\cos\left(\frac{(2q+1)\cdot s\cdot 2^{v+1}\cdot\frac{N}{2^v}\cdot\pi}{2N}\right)$$

$$=(-1)^{r+s}f(m,n)\qquad 0\le r\le 2^{u-1},0\le s\le 2^{v-1}\qquad(5-17)$$

$$f\left(r\cdot\frac{N}{2^u}-1-m,n+s\cdot\frac{N}{2^v}\right)=\cos\left(\frac{(2m+1)(2p+1)\cdot 2^u\pi}{2N}\right)\cdot\cos\left(\frac{(2p+1)\cdot r\cdot 2^{u+1}\cdot\frac{N}{2^u}\cdot\pi}{2N}\right)\cdot$$

$$\cos\left(\frac{(2n+1)(2q+1)\cdot 2^v\cdot\pi}{2N}\right)\cdot\cos\left(\frac{(2q+1)\cdot s\cdot 2^{v+1}\cdot\frac{N}{2^v}\cdot\pi}{2N}\right)$$

$$=(-1)^{r+s}f(m,n)\qquad 0\le r\le 2^{u-1},0\le s\le 2^{v-1}\qquad(5-18)$$

$$f\left(r\cdot\frac{N}{2^u}-1-m,s\cdot\frac{N}{2^v}-1-n\right)=\cos\left(\frac{(2m+1)(2p+1)\cdot 2^u\pi}{2N}\right)\cdot\cos\left(\frac{(2p+1)\cdot r\cdot 2^{u+1}\cdot\frac{N}{2^u}\cdot\pi}{2N}\right)\cdot$$

$$\cos\left(\frac{(2n+1)(2q+1)\cdot 2^v\cdot\pi}{2N}\right)\cdot\cos\left(\frac{(2q+1)\cdot s\cdot 2^{v+1}\cdot\frac{N}{2^v}\cdot\pi}{2N}\right)$$

$$=(-1)^{r+s}f(m,n)\qquad 0\le r\le 2^u,0\le s\le 2^v$$

$$(5-19)$$

$$f\left(m+(2r+1)\cdot\frac{N}{2^{u+1}},n+(2s+1)\cdot\frac{N}{2^{v+1}}\right)=\cos\left((2r+1)\cdot(2p+1)\cdot\frac{\pi}{2}+\frac{(2m+1)(2p+1)\cdot 2^u\pi}{2N}\right)\cdot$$

$$\cos\left((2s+1)\cdot(2q+1)\cdot\frac{\pi}{2}+\frac{(2n+1)(2q+1)\cdot 2^v\pi}{2N}\right)$$

$$=(-1)^{r+p+s+q}\cdot\frac{1}{2}\left(g(m,n)-h(m,n)\right)$$

$$0\le r\le 2^u,0\le s\le 2^v$$

$$(5-20)$$

$$f((2r+1)\cdot\frac{N}{2^{u+1}}-1-m,(2s+1)\cdot\frac{N}{2^{v+1}}-1-n)=\cos\left((2r+1)\cdot(2p+1)\cdot\frac{\pi}{2}-\frac{(2m+1)(2p+1)\cdot 2^u\pi}{2N}\right)\cdot$$
$$\cos\left((2s+1)\cdot(2q+1)\cdot\frac{\pi}{2}-\frac{(2n+1)(2q+1)\cdot 2^v\pi}{2N}\right)$$
$$=(-1)^{r+p+s+q}\cdot\frac{1}{2}\left(g(m,n)-h(m,n)\right)\quad 0\le r\le 2^{u+1},0\le s\le 2^{v+1} \tag{5-21}$$

将(5－16)~(5－21)代入(5－15)得，

$$X((2p+1)2^u,(2q+1)2^v)=\sum_{m=0}^{\frac{N}{2^u}-1}\sum_{n=0}^{\frac{N}{2^v}-1}\left(\sum_{r=0}^{2^u}\sum_{s=0}^{2^v}\left((-1)^{r+s}[x(m+r\cdot 2^u,n+s\cdot 2^v)+x(r\cdot\frac{N}{2^u}-1-m,n+s\cdot 2^v)+\right.\right.$$
$$\left.\left.x(m+r\cdot 2^u,s\cdot\frac{N}{2^v}-1-n)+x(r\cdot\frac{N}{2^u}-1-m,s\cdot\frac{N}{2^v}-1-n)]\right)\cdot f(m,n)\right) \tag{5-22}$$

令:

$$y(m,n)=\sum_{r=0}^{2^u}\sum_{s=0}^{2^v}\left((-1)^{r+s}[x(m+r\cdot 2^u,n+s\cdot 2^v)+x(r\cdot\frac{N}{2^u}-1-m,n+s\cdot 2^v)+\right.$$
$$\left.x(m+r\cdot 2^u,s\cdot\frac{N}{2^v}-1-n)+x(r\cdot\frac{N}{2^u}-1-m,s\cdot\frac{N}{2^v}-1-n)]\right)$$

(5－22)可进一步表示为：

$$X((2p+1)2^u,(2q+1)2^v)=\sum_{m=0}^{\frac{N}{2^u}-1}\sum_{n=0}^{\frac{N}{2^v}-1}y(m,n)\cdot f(m,n)$$
$$=\frac{1}{2}\sum_{m=0}^{\frac{N}{2^u}-1}\sum_{n=0}^{\frac{N}{2^{v+1}}-1}(y(m,n)+y(\frac{N}{2^{u+1}}-1-m,\frac{N}{2^{v+1}}-1-n))\cdot g(m,n)+$$
$$\frac{1}{2}\sum_{m=0}^{\frac{N}{2^u}-1}\sum_{n=0}^{\frac{N}{2^{v+1}}-1}(y(m,n)-y(\frac{N}{2^{u+1}}-1-m,\frac{N}{2^{v+1}}-1-n))\cdot h(m,n) \tag{5-23}$$

由(5－23)可以看出，构成输出 $X(k,l)$集的所有部分和有着大量重复的加法运算，通过合并具有相同余弦因子的元素，可以极大地减少加法运算的数量。

表 5－1*a*，*b*，*c* 给出了 8×8 点 DCT 中 X(2*l*+1，2*k*)，X(2*l*，2*k*+1)，X(2*l*+1，2*k*+1)的部分和项，而 X(2*l*，2*k*)可以看作一个 4×4 点 DCT 求解。附录 1 为 4×4 点 PSDA_DCT 的 C 程序。该程序计算一次 4×4 DCT 总共需要 70 次加法，14 次乘法和 12 次移位运算。

表 5－1*a* X (1，2*k*)的部分和表项

X(*k*, *l*)	余弦因子	余弦因子对应的部分和
X(1,0)	$\frac{1}{16}\pi$	[*y*(0,0) + *y*(0,3)] +[*y*(0,1) +*y*(0,2)]
	$\frac{3}{16}\pi$	[y(1,0) + y(1,3)] +[y(1,1) +y(1,2)]
	$\frac{5}{16}\pi$	[y(2,0) + y(2,3)] +[y(2,1) +y(2,2)]
	$\frac{7}{16}\pi$	[y(3,0) + y(3,3)] +[y(3,1) +y(3,2)]
X(1,2)	$\frac{1}{16}\pi$	{[*y*(0,0)－*y*(0,3)]+[*y*(3,1)－*y*(3,2)]}+ {[*y*(1,0)－*y*(1,3)]+[*y*(2,1)－*y*(2,2)]}
	$\frac{3}{16}\pi$	{[*y*(0,0)－*y*(0,3)]－[*y*(3,1)－*y*(3,2)]}+ {[*y*(1,1)－*y*(1,2)]+[*y*(2,0)－*y*(2,3)]}
	$\frac{5}{16}\pi$	{[*y*(0,1)－*y*(0,2)]+[*y*(3,0)－*y*(3,3)]}+ {[*y*(1,0)－*y*(1,3)]－[*y*(2,1)－*y*(2,2)]}
	$\frac{7}{16}\pi$	{[*y*(0,1)－*y*(0,2)]－[*y*(3,0)－*y*(3,3)]}－ {[*y*(1,1)－*y*(1,2)]－[*y*(2,0)－*y*(2,3)]}
X(1,4)	$\frac{1}{16}\pi$	{[y(1,0)+y(1,3)]－[y(1,1)+y(1,2)]}+ {[y(2,0)+y(2,3)]－[y(2,1)+y(2,2)]}
	$\frac{3}{16}\pi$	{[y(0,0)+y(0,3)]－[y(0,1)+y(0,2)]}+ {[y(3,0)+y(3,3)]－[y(3,1)+y(3,2)]}
	$\frac{5}{16}\pi$	{[y(0,0)+y(0,3)]－[y(0,1)+y(0,2)]}－ {[y(3,0)+y(3,3)]－[y(3,1)+y(3,2)]}
	$\frac{7}{16}\pi$	{[y(1,0)+y(1,3)]－[y(2,1)+y(2,2)]}－ {[y(2,0)+y(2,3)]－[y(2,1)+y(2,2)]}
X(1,6)	$\frac{1}{16}\pi$	－{[y(0,1)－y(0,2)]－[y(3,0)－y(3,3)]}－ {[y(1,1)－y(1,2)]－[y(2,0)－y(2,3)]}
	$\frac{3}{16}\pi$	{[y(0,1)－y(0,2)]+[y(3,0)－y(3,3)]}－ {[y(1,0)－y(1,3)]－[y(2,1)－y(2,2)]}
	$\frac{5}{16}\pi$	{[y(0,0)－y(0,3)]－[y(3,1)－y(3,2)]}－ {[y(1,1)－y(1,2)]+[y(2,0)－y(2,3)]}
	$\frac{7}{16}\pi$	{[y(0,0)－y(0,3)]+[y(3,1)－y(3,2)]}－ {[y(1,0)－y(1,3)]+[y(2,1)－y(2,2)]}

表 5－1*b*　X(2*k*,1)的部分和表项

X(*k*, *l*)	余弦因子	余弦因子对应的部分和
X(0,1)	$\frac{1}{16}\pi$	$[y(0,0)+y(3,0)]+[y(1,0)+y(2,0)]$
	$\frac{3}{16}\pi$	$[y(0,1)+y(3,1)]+[y(1,1)+y(2,1)]$
	$\frac{5}{16}\pi$	$[y(0,2)+y(3,2)]+[y(1,2)+y(2,2)]$
	$\frac{7}{16}\pi$	$[y(0,3)+y(3,3)]+[y(1,3)+y(2,3)]$
X(2,1)	$\frac{1}{16}\pi$	$\{[y(0,0)-y(3,0)]+[y(1,3)-y(2,3)]\}+$ $\{[y(0,1)-y(3,1)]+[y(1,2)-y(2,2)]\}$
	$\frac{3}{16}\pi$	$\{[y(0,0)-y(3,0)]-[y(1,3)-y(2,3)]\}+$ $\{[y(1,1)-y(2,1)]+[y(0,2)-y(3,2)]\}$
	$\frac{5}{16}\pi$	$\{[y(1,0)-y(2,0)]+[y(0,3)-y(3,3)]\}+$ $\{[y(0,1)-y(3,1)]-[y(1,2)-y(2,2)]\}$
	$\frac{7}{16}\pi$	$\{[y(1,0)-y(2,0)]-[y(0,3)-y(3,3)]\}-$ $\{[y(1,1)-y(2,1)]-[y(0,2)-y(3,2)]\}$
X(4,1)	$\frac{1}{16}\pi$	$\{[y(0,1)+y(3,1)]-[y(1,1)+y(2,1)]\}+$ $\{[y(0,2)+y(3,2)]-[y(1,2)+y(2,2)]\}$
	$\frac{3}{16}\pi$	$\{[y(0,0)+y(3,0)]-[y(1,0)+y(2,0)]\}+$ $\{[y(0,3)+y(3,3)]-[y(1,3)+y(2,3)]\}$
	$\frac{5}{16}\pi$	$\{[y(0,0)+y(3,3)]-[y(1,0)+y(2,0)]\}-$ $\{[y(3,0)+y(3,3)]-[y(1,3)+y(2,3)]\}$
	$\frac{7}{16}\pi$	$\{[y(0,1)+y(3,1)]-[y(1,2)+y(2,2)]\}-$ $\{[y(0,2)+y(3,2)]-[y(1,2)+y(2,2)]\}$
X(6,1)	$\frac{1}{16}\pi$	$-\{[y(1,0)-y(2,0)]-[y(0,3)-y(3,3)]\}-$ $\{[y(1,1)-y(2,1)]-[y(0,2)-y(3,2)]\}$
	$\frac{3}{16}\pi$	$\{[y(1,0)-y(2,0)]+[y(0,3)-y(3,3)]\}-$ $\{[y(0,1)-y(3,1)]-[y(1,2)-y(2,2)]\}$
	$\frac{5}{16}\pi$	$\{[y(0,0)-y(3,0)]-[y(1,3)-y(2,3)]\}-$ $\{[y(1,1)-y(2,1)]+[y(0,2)-y(3,2)]\}$
	$\frac{7}{16}\pi$	$\{[y(0,0)-y(3,0)]+[y(1,3)-y(2,3)]\}-$ $\{[y(0,1)-y(3,1)]+[y(1,2)-y(2,2)]\}$

表 5－1c $X(1,2k+1)$的部分和表项

X(k,l)	余弦因子	余弦因子对应的部分和
X(1,1)	0	[y(0,0) + y(3,3)] +[y(1,1) +y(2,2)]
	$\frac{1}{8}\pi$	{[y(0,0)－y(3,3)]+[y(1,2)+y(2,1)]}+ {[y(0,1)+y(3,2)]+[y(1,0)+y(2,3)]}
	$\frac{1}{4}\pi$	{[y(0,1)－y(3,2)]+[y(1,0)－y(2,3)]}+ {[y(0,2)+y(3,1)]+[y(1,3)+y(2,0)]}
	$\frac{3}{8}\pi$	{[y(0,2)－y(3,1)]－[y(1,3)－y(2,0)]}+ [y(0,3)+y(3,0)]+[y(1,1)－y(2,2)]}
X(1,3)	0	－[y(0,2)+y(3,1)]+[y(1,0)－y(2,3)]
	$\frac{1}{8}\pi$	{[y(0,0)－y(3,3)]－[y(1,2)+y(2,1)]}－ {[y(0,2)－y(3,1)]+[y(1,3)－y(2,0)]}
	$\frac{1}{4}\pi$	{[y(0,0)+y(3,3)]－[y(1,2)－y(2,1)]}－ {[y(0,3)－y(3,0)]+[y(1,1)+y(2,2)]}
	$\frac{3}{8}\pi$	－{[y(0,1)+y(3,2)]－[y(1,0)+y(2,3)]}－ [y(0,3)+y(3,0)]－[y(1,1)－y(2,2)]}
X(1,5)	0	－[y(0,1)+y(3,2)]+[y(1,3)+y(2,0)]
	$\frac{1}{8}\pi$	－{[y(0,1)+y(3,2)]－[y(1,0)+y(2,3)]}+ [y(0,3)+y(3,0)]－[y(1,1)－y(2,2)]}
	$\frac{1}{4}\pi$	{[y(0,0)+y(3,3)]+[y(1,2)－y(2,1)]}+ {[y(0,3)－y(3,0)]－[y(1,1)+y(2,2)]}
	$\frac{3}{8}\pi$	{[y(0,0)－y(3,3)]－[y(1,2)+y(2,1)]}+ {[y(0,2)－y(3,1)]+[y(1,3)－y(2,0)]}
X(1,7)	0	－{[y(0,1)－y(0,2)]－[y(3,0)－y(3,3)]}－ {[y(1,1)－y(1,2)]－[y(2,0)－y(2,3)]}
	$\frac{1}{8}\pi$	{[y(0,2)－y(3,1)]－[y(1,3)－y(2,0)]}－ [y(0,3)+y(3,0)]+[y(1,1)－y(2,2)]}
	$\frac{1}{4}\pi$	－{[y(0,1)－y(3,2)]－[y(1,0)－y(2,3)]}+ {[y(0,2)+y(3,1)]－[y(1,3)+y(2,0)]}
	$\frac{3}{8}\pi$	{[y(0,0)－y(3,3)]+[y(1,2)+y(2,1)]}－ {[y(0,1)+y(3,2)]+[y(1,0)+y(2,3)]}

5.2.4 部分和分解算法的软件实现

以 8×8 点 DCT 为例：先对 $x(m, n)$进行以下预处理。

Step1：对每列数据进行图 5－1 的预处理。

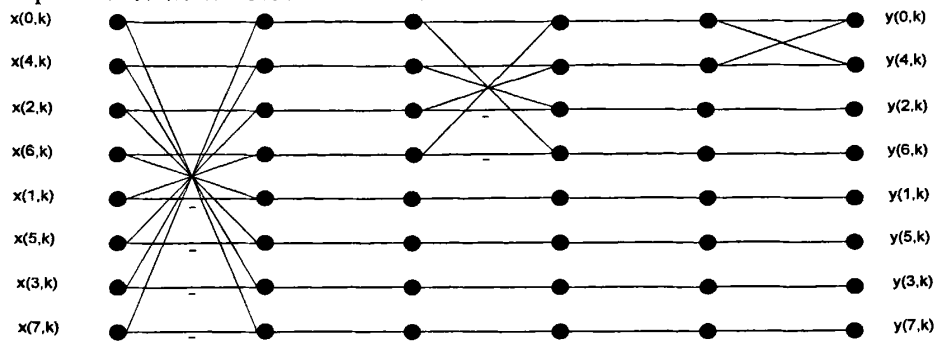


图 5－1 列数据预处理

Step2：对每行数据进行图 5－2 的预处理

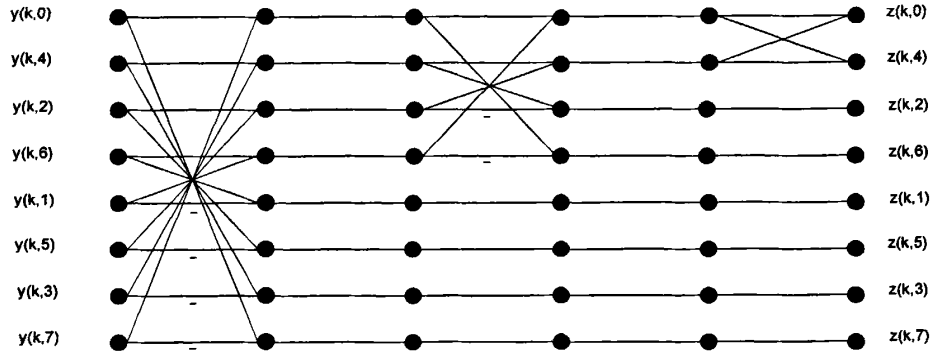


图 5－2 行数据预处理

Step3：计算部分和

计算 X(l，k)的部分和等价于求解某一特定余弦因子 cos($k\pi$/N)所对应的 $x(m, n)$。也就是求解同余方程：

$$\left|2m+1\pm(2n+1)k\right| \equiv t \pmod{2N} \tag{5－24}$$

由定理 3.3.1.2，求得(5－10)的解满足：

$$m \equiv \left|\frac{t-1\mp(2n+1)k}{2}\right| \pmod{N} \tag{5－25}$$

由(5－20)，可以求解出满足要求的(m，n)。将这些具有相同余弦因子的 $x(m, n)$累加，即可求得对应的 X(l ， k)的部分和。

Step4：将输出部分和按 5.2.3 的划分准则分组。计算输出部分和的 DCT－IV 或 DCT－III 变换输出。

附录 2 为 8×8 点 PSDA_DCT 的 C 程序。

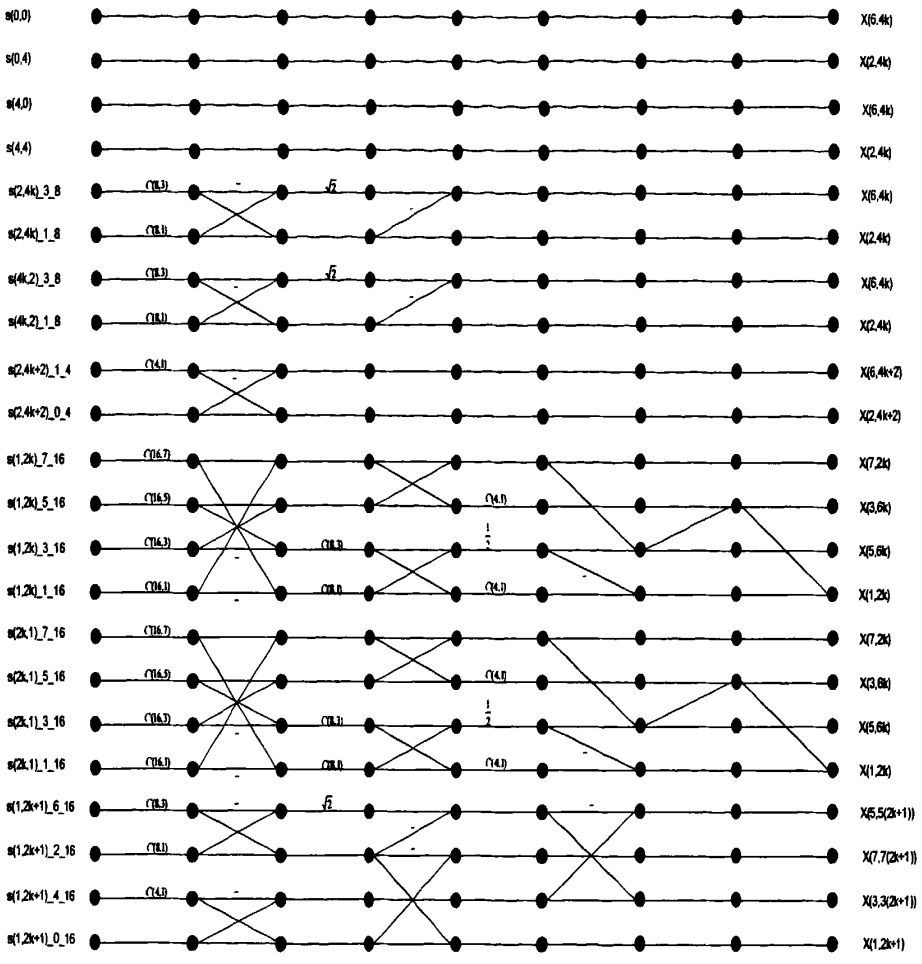


图 5－3 二维 DCT 变换输出

5.2.5 $2^n \times 2^n$ 型 DCT 部分和分解算法复杂度

令 $N=2^n$，记 $2^n \times 2^n$ 型 DCT 的乘法复杂度为 $\mu_m(2^n,2^n)$，加法复杂度为 $\mu_a(2^n,2^n)$，2^n 型 DCT－II 的乘法复杂度为 $\mu_m(2^n)_{DCT\_II}$，加法复杂度为 $\mu_a(2^n)_{DCT\_II}$，2^n 型 DCT－IV 的乘法复杂度为 $\mu_m(2^n)_{DCT\_IV}$，加法复杂度为 $\mu_a(2^n)_{DCT\_IV}$，

由前面的推导可得：

$$\begin{aligned}\mu_m(2^n,2^n)&=\mu_m(2^{n-1},2^{n-1})+2\cdot 2^{n-1}\mu_m(2^{n-1})_{DCT\_IV}+2^{n-1}\cdot\mu_m(2^{n-1})_{DCT\_II}\\&=\mu_m(2^n,2^n)+\frac{(n+1)\cdot 2^n\cdot 2^n}{4}+\frac{(n-1)\cdot 2^n\cdot 2^n}{8}\\&=\mu_m(2^n,2^n)+\frac{3}{8}n\cdot 4^n+\frac{1}{8}4^n\\&\vdots\\&=\frac{1}{8}\sum_{k=2}^{n}(3k+1)\cdot 4^k\end{aligned}\tag{5-26}$$

$$\begin{aligned}\mu_a(2^n,2^n)&=\mu_a(2^{n-1},2^{n-1})+2\cdot 2^{n-1}\mu_a(2^{n-1})_{DCT\_IV}\\&\quad+2^{n-1}\cdot\mu_a(2^{n-1})_{DCT\_II}+8\cdot 4^{n-1}+3n\cdot 4^{n-1}\\&=\mu_a(2^{n-1},2^{n-1})+3\cdot 2^{n-1}\cdot\left(\frac{2^n}{4}\cdot(3n-5)+1\right)\\&\quad+3n\cdot 4^{n-1}+8\cdot 4^{n-1}\\&\vdots\\&=\mu_a(4,4)+\frac{15}{8}\sum_{k=3}^{n}k\cdot 4^k+\frac{1}{8}\sum_{k=3}^{n}4^k+3\cdot\sum_{k=3}^{n}2^{k-1}\end{aligned}\tag{5-27}$$

$\mu_m(2,2)=0$，故：

$$\begin{aligned}\mu_m(2^n,2^n)&=\frac{1}{8}\sum_{k=2}^{n}(3k+1)\cdot 4^k\\&=\frac{3}{8}\sum_{k=2}^{n}(k+1)\cdot 4^k-\frac{1}{4}\sum_{k=2}^{n}4^k\\&=\frac{3}{8}\sum_{k=2}^{n}(k+1)\cdot x^k\Big|_{x=4}-\frac{1}{4}\sum_{k=2}^{n}4^k\\&=\frac{3}{8}\frac{d\sum_{k=2}^{n}x^{k+1}}{dx}\Big|_{x=4}-\frac{1}{4}\sum_{k=2}^{n}4^n\\&=n\cdot 2^{2n-1}-2\end{aligned}\tag{5-28}$$

$\mu_a(4,4)=70$，故：

$$
\begin{aligned}
u_4(2^n,2^n) &= \mu_4(4,4)+\frac{15}{8}\sum_{k=3}^{n}k\cdot 4^k+\frac{1}{8}\sum_{k=3}^{n}4^k+3\cdot\sum_{k=3}^{n}2^{k-1} \\
&= 70+\frac{15}{8}\sum_{k=3}^{n}(k+1)4^k+3\cdot\sum_{k=3}^{n}2^{k-1}-\frac{7}{4}\sum_{k=3}^{n}4^k \\
&= 70+3\cdot(2^n-4)-\frac{7}{4}\cdot\frac{4^{n+1}-64}{3}+\frac{15}{8}\frac{d\sum_{k=3}^{n}x^{k+1}}{dx}\Bigg|_{x=4} \\
&= 58+3\cdot 2^n-\frac{7}{4}\cdot\frac{4^{n+1}-64}{3}+\frac{15}{8}\cdot\frac{(x-1)\left((n+2)x^{n+1}-4x^3\right)-(x^{n+2}-x^4)}{(x-1)^2}\Bigg|_{x=4} \\
&= 58+3\cdot 2^n-\frac{7}{4}\cdot\frac{4^{n+1}-64}{3}+\frac{5}{8}\cdot\frac{(3n+2)4^{n+1}-512}{3} \\
&= \frac{5}{2}\cdot n\cdot 4^n+3\cdot 2^n-\frac{2\cdot 4^n+34}{3}
\end{aligned}
\tag{5-29}
$$

表 5－2 2^nx2^n 型 DCT 计算复杂性比较

运算复杂度	加法		乘法	
	行列法	PSDA 算法	行列法	PSDA 算法
	$n\times 4^n$	$\frac{n}{2}\cdot 4^n-2$	$(3n-2)\times 4^n+2^{n+1}$	$\frac{5}{2}\cdot n\cdot 4^n+3\cdot 2^n-\frac{2\cdot 4^n+34}{3}$
4×4	32	14	72	70
8×8	192	94	464	440
16×16	1024	510	2592	2426
32×32	5120	2558	13376	12202

5.3 $q\times q$ (q=奇素数)型 DCT 的部分和分解算法

设 M =N = q ，q 为奇素数，下面讨论该条件下的二维 DCT 算法。

$$
\begin{aligned}
X(k,l) &= \sum_{m=0}^{q-1}\sum_{n=0}^{q-1} x(m,n)\cdot\cos\left\{\frac{(2m+1)k\pi}{2q}\right\}\cdot\cos\left\{\frac{(2n+1)l\pi}{2q}\right\} \\
&= \sum_{\substack{m=0 \\ m\neq\frac{(q-1)}{2}}}^{q-1}\sum_{\substack{n=0 \\ n\neq\frac{(q-1)}{2}}}^{q-1} x(m,n)\cdot\cos\left\{\frac{(2m+1)k\pi}{2q}\right\}\cdot\cos\left\{\frac{(2n+1)l\pi}{2q}\right\} + \\
&\quad \sum_{m=0}^{q-1} x(m,\frac{q-1}{2})\cos\left\{\frac{(2m+1)k\pi}{2q}\right\}\cdot\cos\left\{\frac{l\pi}{2}\right\} + \\
&\quad \sum_{n=0}^{q-1} x(\frac{q-1}{2},n)\cos\left\{\frac{(2n+1)l\pi}{2q}\right\}\cdot\cos\left\{\frac{m\pi}{2}\right\} - \\
&\quad x(\frac{q-1}{2},\frac{q-1}{2})\cdot\cos\left\{\frac{m\pi}{2}\right\}\cdot\cos\left\{\frac{l\pi}{2}\right\}
\end{aligned}
\tag{5-30}
$$

令

$$
A(k,l) = \sum_{\substack{m=0 \\ m\neq\frac{(q-1)}{2}}}^{q-1}\sum_{\substack{n=0 \\ n\neq\frac{(q-1)}{2}}}^{q-1} x(m,n)\cdot\cos\left\{\frac{(2m+1)k\pi}{2q}\right\}\cdot\cos\left\{\frac{(2n+1)l\pi}{2q}\right\} \tag{5-31}
$$

$$
B(k,l) = \sum_{m=0}^{q-1} x(m,\frac{q-1}{2})\cos\left\{\frac{l\pi}{2}\right\}\cos\left\{\frac{(2m+1)k\pi}{2q}\right\} \tag{5-32}
$$

$$
C(k,l) = \sum_{n=0}^{p-1} x(\frac{q-1}{2},n)\cos\left\{\frac{k\pi}{2}\right\}\cos\left\{\frac{(2n+1)l\pi}{2q}\right\} \tag{5-33}
$$

则

$$
X(k,l) = A(k,l)+B(k,l)+C(k,l)-x(\frac{q-1}{2},\frac{q-1}{2})\cos\frac{k\pi}{2}\cos\frac{l\pi}{2} \tag{5-34}
$$

由式(5－34)可知，$X(k，l)$可分为四个子项之和，又因为$-x(\frac{q-1}{2},\frac{q-1}{2})\cos\frac{k\pi}{2}\cos\frac{l\pi}{2}$为常量，因此只需要计算分别A($k$，$l$)，B($k$，$l$)，C($k$，$l$)即可。

5.3.1 交流分量 A(k，l)

由

$$A(k,l)=\sum_{\substack{m=0\\ m\neq\frac{(q-1)}{2}}}^{q-1}\sum_{\substack{n=0\\ n\neq\frac{(q-1)}{2}}}^{q-1}x(m,n)\cdot\cos\left\{\frac{(2m+1)k\pi}{2q}\right\}\cdot\cos\left\{\frac{(2n+1)l\pi}{2q}\right\}$$

$$=\frac{1}{2}\sum_{\substack{m=0\\ m\neq\frac{(q-1)}{2}}}^{q-1}\sum_{\substack{n=0\\ n\neq\frac{(q-1)}{2}}}^{q-1}x(m,n)\cdot\left[\cos\left(\frac{(2m+1)k+(2n+1)l}{2q}\pi\right)+\right.\tag{5-35}$$

$$\left.\cos\left(\frac{(2m+1)k-(2n+1)l}{2q}\pi\right)\right]$$

若 $k=0$，则

$$A(0,l)=\sum_{\substack{n=0\\ n\neq\frac{(q-1)}{2}}}^{q-1}\left[\sum_{\substack{m=0\\ m\neq\frac{(q-1)}{2}}}^{q-1}x(m,n)\right]\cdot\cos\left\{\frac{(2n+1)l\pi}{2q}\right\}\tag{5-36}$$

若 $l=0$，则

$$A(0,l)=\sum_{\substack{m=0\\ m\neq\frac{(q-1)}{2}}}^{q-1}\left[\sum_{\substack{n=0\\ n\neq\frac{(q-1)}{2}}}^{q-1}x(m,n)\right]\cdot\cos\left\{\frac{(2m+1)k\pi}{2q}\right\}\tag{5-37}$$

式(5－36)和式(5－37)为一个一维素长度 DCT，可以用 4.4 中的方法来计算。

若 $k\neq0$，$l\neq0$，设

$$amod\left((2m+1)k+(2n+1)l,2q\right)=t;\qquad 0\le t<q\tag{5-38}$$

$$amod\left((2m+1)k-(2n+1)l,2q\right)=t;\qquad 0\le t<q\tag{5-39}$$

因

$$(m,n,q)=1$$

由定理 3.3.1.2，可知，若

$$mod(t-(k\pm l),2)=0$$

又 $m\neq(q-1)/2$，$n\neq(q-1)/2$，同余方程(5－38)有$(q-1)$个解；同余方程(5－34)有$(q-1)$个解。故同余方程共有$(2q-2)$个解。设这$(2q-2)$个解为：

设这 $q-1$ 个解为

$$\left\{(m_0,n_0,t),(m_1,n_1,t),\cdots(m_{\frac{q-3}{2}},n_{\frac{q-3}{2}},t),(m_{\frac{q+1}{2}},n_{\frac{q+1}{2}},t),(m_{q-2},n_{q-2},t),(m_{q-1},n_{q-1},t)\right\}$$

令：

$$s(m_i,n_i,t)=\begin{cases}1 & mod\left((2m_i+1)k+(2n_i+1)l\right),2q)<q\\ -1 & mod\left((2m_i+1)k+(2n_i+1)l\right),2q)>q\end{cases}$$

余弦因子 $\cos(t\pi/2q)$所对应的部分和为：

$$Sum(t,k,l)=\sum_{\substack{i=0\\ i\neq\frac{q-1}{2}}}^{q-1}\left(s(m_i,n_i,t)\cdot x(m_i,n_i,t)\right) \tag{5-40}$$

若

$$mod(k\pm l,2)=0$$

则

$$A(k,l)=\sum_{t=0}^{\frac{q-3}{2}}Sum(t,k,l)\cos\left\{\frac{t\pi}{q}\right\} \tag{5-41}$$

若

$$mod(k\pm l,2)=1$$

则

$$A(k,l)=\sum_{t=0}^{\frac{q-3}{2}}Sum(t,k,l)\cos\left\{\frac{(2t+1)\pi}{2q}\right\} \tag{5-42}$$

由式(5－41)和(5－42)得

$$A(1,2v+1)=\sum_{t=0}^{\frac{q-1}{2}}Sum(t,1,2v+1)\cos\left\{\frac{t\pi}{q}\right\},\quad 0\le v<\frac{q-1}{2} \tag{5-43a}$$

$$A(2u,2v)=\sum_{t=0}^{\frac{q-1}{2}}Sum(t,2u,2v)\cos\left\{\frac{t\pi}{q}\right\},\ 0\le u<\frac{q-1}{2},\ 0\le v<\frac{q-1}{2} \tag{5-43b}$$

$$A(1,2v)=\sum_{t=0}^{\frac{q-3}{2}}Sum(t,1,2v)\cos\left\{\frac{(2t+1)\pi}{2q}\right\},\quad 0\le v<\frac{q-1}{2} \tag{5-43c}$$

$$A(2u,1)=\sum_{t=0}^{\frac{q-3}{2}}Sum(t,2u,1)\cos\left\{\frac{(2t+1)\pi}{2q}\right\},\quad 0<u<\frac{q-1}{2} \tag{5-43d}$$

令：

$$Sgn(x)=\begin{cases}1 & mod\left((x,2q\right)<q\\ -1 & mod\left(x,2q\right)\ge q\end{cases}$$

由式(5.2.2)，可得：

$$A(2u+1,amod\left((2u+1)(2v+1),2q\right))=Sgn\left((2u+1)\cdot(2v+1)\right)\cdot \sum_{t=0}^{\frac{q-3}{2}}Sum(t,1,2v+1)\cos\left\{\frac{(2u+1)t\pi}{q}\right\} \tag{5-44a}$$

$$0\le u<\frac{q-1}{2},0\le v<\frac{q-1}{2}$$

$$A\left(amod\left((2r+1)\cdot 2u,2q\right),amod\left((2r+1)\cdot 2v,2q\right)\right)=Sgn(2(2r+1)u)\cdot$$

$$Sgn(2(2r+1)u)\cdot\sum_{t=0}^{\frac{q-1}{2}}Sum(t,2,2v)\cos\left\{\frac{(2r+1)t\pi}{q}\right\} \quad (5-44b)$$

$$0\le u\le\frac{q-1}{2},0\le v\le\frac{q-1}{2},0\le r\le\frac{q-3}{2}$$

$$A(2u+1,amod\left(2v(2u+1),2q\right))=Sgn(2v(2u+1))\cdot$$

$$\sum_{t=0}^{\frac{q-3}{2}}Sum(t,1,2v)\cos\left\{\frac{(2t+1)(2u+1)\pi}{2q}\right\} \quad (5-44c)$$

$$0\le u<\frac{q-1}{2},0\le v\le\frac{q-1}{2}$$

$$A(amod\left(2u(2v+1),2q\right),2v+1)=Sgn(2u(2v+1))\cdot$$

$$\sum_{t=0}^{\frac{q-3}{2}}Sum(t,2u,1)\cos\left\{\frac{(2t+1)(2v+1)\pi}{2q}\right\} \quad (5-44d)$$

$$0\le u\le\frac{q-1}{2},0\le v<\frac{q-1}{2}$$

观察式(5－44a)，(5－44*b*)，发现它们与式(4－42)同型，使用前一章 4.4 节中素长度 DCT 快速算法中的偶下标系数的计算方法即可，这里不再详细推导。

观察式(5－44*c*)，(5－44*d*)，发现它们与式(4－51)同型，使用前一章 4.4 节中素长度 DCT 快速算法中的奇下标系数的计算方法即可，这里不再详细推导。

5.3.2 交流分量 B(*k*，*l*)

若 l=2u +1，则 B(k，2u+1)=0。

若 l=2u，则由(5－27)

$$B(k,2u)=(-1)^u\sum_{m=0}^{q-1}x(m,\frac{q-1}{2})\cos\left\{\frac{(2m+1)k\pi}{2q}\right\} \quad (5-45)$$

这是一个一维 DCT 变换的形式，可以用第四章中的一维 DCT 快速算法来计算。

5.3.3 交流分量 C(*k*，*l*)

若 k=2v + 1，则 B(2v+1，　l)=0。

若 k=2v，则由(5－26)

$$C(2v,l)=(-1)^v\sum_{n=0}^{q-1}x(\frac{q-1}{2},n)\cos\left\{\frac{(2n+1)l\pi}{2q}\right\} \quad (5-46)$$

这是一个一维DCT变换的形式，可以用第四章中的一维DCT快速算法来计算。

5.3.4 基于部分和的频域(k，l)集合划分

记二维频域(k，l)平面上的点集为 A。令 B(k，l) = {($amod(2r+1)k$，$2q$)，$amod(2r+1)l$，$2q$))，$0 \le r < (q-1)/2$，$0 \le k < q$，$0 \le l < q$}。下面讨论 A 与 A(r，k，l)的关系。

设 $r_1 \neq r_2$，$0 < k < q$，$0 < l < q$，因 q 为奇素数，可知

$$amod\left((2r_1+1)\cdot k, 2q\right) - amod\left((2r_2+1)\cdot k, 2q\right) = amod\left(2(r_1-r_2)\cdot k, 2q\right) \neq 0$$

$$amod\left((2r_1+1)\cdot l, 2q\right) - amod\left((2r_2+1)\cdot l, 2q\right) = amod\left(2(r_1-r_2)\cdot l, 2q\right) \neq 0$$

故：

{($amod(2r+1)k$，2q)，$amod(2r+1)l$，2q}}，$0 \le r < (q-1)/2$，$0 < k$，$l <$ q} = {(k，l)，$0 < k$，$l <$ q}

$$A = \bigcup_{k=0}^{\frac{q-1}{2}-1} \bigcup_{l=0}^{\frac{q-1}{2}-1} B(k,l) \tag{5-47}$$

$$B(k_1,l_1) \bigcap B(k_2,l_2) = \phi, \qquad k_1 \neq k_2 \text{ or } l_1 \neq l_2 \tag{5-48}$$

由式(5－38)和式(5－39)可知，每个 $B(k, l)$中有$(q-1)/2$ 个元素，$B(k, l)$子集的个数为

$$R(B(k,l)) = \frac{q^2-2q+1}{(q-1)/2} = 2(q-1)$$

以 7×7 点 DCT 为例，根据部分和相同的元素放在同一子集的原则，划分出的子集如下。

$$\{X(0,0)\},$$

$$\{X(0,1), X(0,3), X(0,5)\}, \{X(0,2), X(0,4), X(0,6)\}$$

$$\{X(1,0), X(3,0), X(5,0)\}, \{X(2,0), X(4,0), X(6,0)\}$$

$$\{X(1,1), X(3,3), X(5,5)\}, \{X(1,2), X(3,6), X(5,4)\},$$

$$\{X(1,3), X(3,5), X(5,1)\}, \{X(1,4), X(3,2), X(5,6)\},$$

$$\{X(1,5), X(3,1), X(5,3)\}, \{X(1,6), X(3,4), X(5,2)\},$$

$$\{X(2,1), X(6,3), X(4,5)\}, \{X(2,2), X(6,6), X(4,4)\},$$

$$\{X(4,1), X(2,3), X(6,5)\}, \{X(4,2), X(2,6), X(6,4)\},$$

$$\{X(6,1), X(4,3), X(2,5)\}, \{X(6,2), X(4,6), X(2,4)\}$$

5.3.5 部分和求解

(5.2.4)中推导了部分和分解算法的频域子集划分方法。本节推导部分和的计算方法。

若 $k\neq 0$，$l\neq 0$，设

$$amod\left((2m+1)k+(2n+1)l,2q\right)=t;\ 0\le t<q,0\le m,n<q,m,n\neq\frac{q-1}{2} \tag{5-49}$$

$$amod\left((2m+1)k-(2n+1)l,2q\right)=t;\ 0\le t<q,0\le m,n<q,m,n\neq\frac{q-1}{2} \tag{5-50}$$

同余方程(5－38)有 $q-1$ 个解；同余方程(5－39)有 $q-1$ 个解。设这 $q-1$ 个解为$\{(m_{i,t},\ n_{i,t})|0\le i<q,\ 0<t<q\}$，以下根据 k，l 的奇偶情况讨论这 $2q-2$ 个解的推导：

I：$(k,l,2)=0$

令

$$amod\left((2m+1)k+(2n+1)l,2q\right)=1 \tag{5-51a}$$

$$amod\left((2m+1)k-(2n+1)l,2q\right)=1 \tag{5-51b}$$

的解为$\{(m_{i,1},\ n_{i,1})|0\le i<q\}$

令

$$m_{i,t}=\left(mod((2m_{i,1}+1)\cdot(2t+1),2q)-1\right)/2 \tag{5-52a}$$

$$n_{i,t}=\left(mod((2n_{i,1}+1)\cdot(2t+1),2q)-1\right)/2 \tag{5-52b}$$

由同余特性可证明

$$\begin{aligned}&amod\left(2m_{i,t}+1)\cdot k+(2n_{i,t}+1)\cdot l\right)\\&=amod\left(mod((2m_{i,1}+1)\cdot(2t+1),2q)\cdot k+mod((2n_{i,1}+1)\cdot(2t+1),2q)\cdot l,2q\right)\\&=2t+1\end{aligned} \tag{5-53a}$$

$$\begin{aligned}&amod\left(2m_{i,t}+1)\cdot k-(2n_{i,t}+1)\cdot l\right)\\&=amod\left(mod((2m_{i,1}+1)\cdot(2t+1),2q)\cdot k-mod((2n_{i,1}+1)\cdot(2t+1),2q)\cdot l,2q\right)\\&=2t+1\end{aligned} \tag{5-53b}$$

若 $t_1\neq t_2$，则

$$m_{i,t_1}-m_{i,t_2}=mod((2m_{i,1}+1)\cdot(t_1-t_2),2q)\neq 0 \tag{5-54a}$$

$$n_{i,t_1}-n_{i,t_2}=mod((2n_{i,1}+1)\cdot(t_1-t_2),2q)\neq 0 \tag{5-54b}$$

由式(5－54*a*)和(5－54*b*)可看出，如果我们可求得满足(5－51*a*)和(5－51*b*)的余弦因子 $\cos(\pi/2q)$所对应的时域解$(m_{i,1}, n_{i,1})$，那么其他余弦因子 $\cos((2t+1)\pi/2q)$所对应的时域解为$(m_{i,t}, n_{i,t})$。

II: $(k,l,2)=2$

令

$$amod\left((2m+1)k+(2n+1)l,2q\right)=2 \tag{5－55a}$$

$$amod\left((2m+1)k-(2n+1)l,2q\right)=2 \tag{5－55b}$$

的解为$\{(m_{i,2}, n_{i,2})|0\le i<q\}$

令

$$t'=\left(mod\left((2m_{i,2}+1)\cdot(2t+1),2q\right)-1\right)/2 \tag{5－56a}$$

由同余特性可证明

$$\begin{aligned} &amod\left((2t'+1)\cdot k+(2t'+1)\cdot l\right)\\ &=amod\left(mod((2m_{i1}+1)\cdot(2t+1),2q)\cdot k+mod((2n_{i1}+1)\cdot(2t+1),2q)\cdot l,2q\right)\\ &=2t+1 \end{aligned} \tag{5－57a}$$

$$\begin{aligned} &amod\left((2t'+1)\cdot k-(2t'+1)\cdot l\right)\\ &=amod\left(mod((2m_{i1}+1)\cdot(2t+1),2q)\cdot k-mod((2n_{i1}+1)\cdot(2t+1),2q)\cdot l,2q\right)\\ &=2t+1 \end{aligned} \tag{5－57b}$$

若 $t_1\neq t_2$，则

$$m_{i,t1}-m_{i,t2}=mod((2m_{i,1}+1)\cdot(t_1-t_2),2q)\neq 0 \tag{5－58a}$$

$$n_{i,t1}-n_{i,t2}=mod((2n_{i,1}+1)\cdot(t_1-t_2),2q)\neq 0 \tag{5－58b}$$

由式(5－58*a*)和(5－58*b*)可看出，如果我们可求得满足(5－55*a*)和(5－55*b*)的余弦因子 $\cos(\pi/2q)$所对应的时域解$(m_{i,2}, n_{i,2})$，那么其他余弦因子 $\cos(2t\pi/2q)$ $(t\neq 0)$所对应的时域解为$(m_{i,t}, n_{i,t})$。

由 I，II，我们可得到以下结论：

部分和的时域解集可由集合$\{(m_{i,0}, n_{i,0})\}$，$\{(m_{i,1}, n_{i,1})\}$，$\{(m_{i,2}, n_{i,2})\}$全部求出。

5.3.6 部分和分解算法的计算复杂性

如果以 $\mu_m(N)$ 表示 N 点 DCT 的乘法复杂性，以 μ_a(N)表示 N 点 DCT 的加法复杂性；以 μ_m(A) 表示计算 A(k，l)的乘法复杂性，以 μ_a(A)表示计算 A(k，l)的加法复杂性；以 μ_m(B)表示计算 B(k，l)的乘法复杂性，以 μ_a(B)表示计算 B(k，l)的加法复杂性；以 μ_m(C)表示计算 C(k，l)的乘法复杂性，以 μ_a(C)表示计算 C(k，l)的加法复杂性；以 $\mu_m(q\times q)$表示计算 $q\times q$ DCT 的乘法复杂

性，以 $\mu_a(q\times q)$表示计算 $q\times q$ DCT 的加法复杂性；那么，可以得到 $q\times q$ DCT 的计算复杂度为：

$$\mu_m(q\times q)=\mu_m(A)+\mu_m(B)+\mu_m(C) \tag{5－59a}$$

$$\mu_a(q\times q)=\mu_a(A)+\mu_a(B)+\mu_a(C)+\frac{(q-1)^2}{4}+2q^2 \tag{5－59b}$$

下面分别求解 A(k，l)，B(k，l)，C(k，l)的计算复杂性。

1) A(k，l)的计算复杂性

$$\begin{aligned}\mu_m(A)&=\mu_m(A(0,l))+\mu_m(A(k,0))+\sum_{k=1}^{p-1}\sum_{l=1}^{p-1}\mu_m(A(k,l))\\&=\mu_m(A(0,l))+\mu_m(A(k,0))+\sum_{k=1}^{p-1}\sum_{l=1}^{p-1}\mu_m(A(k,l))\\&=\mu_m(q)+\mu_m(q)+(q-1)\mu_m(q)\\&=(q+1)\mu_m(q)\end{aligned} \tag{5－60a}$$

$$\begin{aligned}\mu_a(A)&=\mu_a(A(0,l))+\mu_a(A(k,0))+\sum_{k=1}^{p-1}\sum_{l=1}^{p-1}\mu_a(A(k,l))\\&=\mu_a(A(0,l))+\mu_a(A(k,0))+\sum_{k=1}^{p-1}\sum_{l=1}^{p-1}\mu_a(B(k,l))+2q\cdot\frac{q-1}{2}\cdot 2(q-1)\\&=\mu_a(q)+\mu_a(q)+(q-1)\mu_a(q)+2q\cdot\frac{q-1}{2}\cdot 2(q-1)\\&=(q+1)\mu_a(q)+2q\cdot(q-1)^2\end{aligned} \tag{5－60b}$$

2) B(k，l)的计算复杂性

$$\mu_m(B)=\mu_m(q) \tag{5－61a}$$

$$\mu_a(B)=\mu_a(q) \tag{5－61b}$$

3) C(k，l)的计算复杂性

$$\mu_m(C)=\mu_m(q) \tag{5－62a}$$

$$\mu_a(C)=\mu_a(C) \tag{5－62b}$$

结合式(5－40)，(5－41)，(5－42)，(5－43)，可求得 $q\times q$ DCT 的计算复杂度

$$\mu_m(q\times q)=(q+1)\mu_m(q)+\mu_m(q)+\mu_m(q)=(q+3)\mu_m(q) \tag{5－63a}$$

$$\mu_a(q\times q)=\mu_a(A)+\mu_a(B)+\mu_a(C)+\frac{(q-1)^2}{4}+2q^2$$

$$=(q+1)\mu_a(q)+2q\cdot(q-1)^2+\mu_a(q)+\mu_a(q)+\frac{(q-1)^2}{4}+2q^2 \qquad (5-63b)$$

$$=(q+3)\mu_a(q)+(2q+\frac{1}{4})\cdot(q-1)^2+2q^2$$

表 5－3 为部分和分解算法与行列法的计算复杂性比较，从表中可以看出，相对于常规的行列法，PSDA 算法的乘法次数减少了约一半，而加法次数也基本持平。

表 5－3 qxq 型 DCT 加法运算复杂性比较

DCT 长度	行列法	PSDA 算法
$q\times q$	$2qu_a(q)$	$(q+3)\mu_a(q)+(2q+\frac{1}{4})\cdot(q-1)^2+2$

表 5－4 qxq 型 DCT 乘法运算复杂性比较

DCT 长度	行列法	PSDA 算法
$q\times q$	$2qu_m(q)$	$(q+3)u_m(q)$

5.4 本章小结

本章第二节首先提出了部分和的概念，根据部分和相同的原则对 X(k, l)进行子集划分，同一子集内的所有元素由这些部分和所组成的 DCT－III 或 DCT－IV 变换对构成。然后针对一种特殊尺寸(N×N，N = 2^n)2D－DCT，设计了一种基于部分和分解的快速算法，通过适当的下标映射将 N×N 2D－DCT 直接转化为 2N 个长度从 2 到 N/2 的 1D－DCT。相对 N.I.Cho 等人的高效算法[30]，该算法实现了更少的乘法次数和加法次数。

本章第三节提出了素数尺寸($q\times q$， q=奇素数)型 DCT 的部分和分解算法，这是一种算法复杂性低而且结构规则的算法；根据同余方程理论，提出了求解部分和的算法；按照相同部分和的分集准则，将一个二维 DCT 变换分解为(q－1)个尺寸为(q－1)的一维素长度偶下标系数变换和奇下标系数变换。相对于行列法，本章提出的部分和分解算法乘法次数减少了约一半，而加法次数维持同等规模。

本章工作已写成阶段论文，分别发表在《the Chinese Journal of Electronics》2007 年第 2 期上[83]，录用在 IEEE International Workshop on VLSI Design and Video Technology 2005 国际会议 [84]和第六届国际智能交通会议上[85]，并已被电子与信息学报录用。

第六章 部分和分解算法的 VLSI 实现及验证

本章研究部分和分解算法的 VLSI 结构，给出了 $2^n \times 2^n$ 型 DCT 和素数型二维 DCT 的 VLSI 结构。

6.1 引言

第五章研究了部分和的计算方法和部分和分解算法的运算复杂度，相对于别的算法，该算法虽然实现了更少的加法次数和乘法次数，但是 VLSI 实现更关心的是乘法器个数和数据传递复杂性。本章讨论部分和分解算法的 VLSI 实现和。

在 DCT 的硬件实现算法中，直接实现的计算量太大，加上乘法器在 IC 实现中要占用较大的面积，因此，许多实现 DCT 的方案都强调减少乘法器的数量。为了减少乘法数量，N×N DCT 可以通过采用能够降低乘法数量的各种各样的蝶型结构来实现。但是，为了能达到要求的数据吞吐量，许多乘法器仍然是必不可少的，此外，蝶型结构的方案经常产生不规则的结构和复杂的布线，由此会增大硅晶片面积，延长设计时间。而且，由于在有限精度的算法中，乘法的多个步骤都有四舍五入的操作，从而导致结果的准确度严重下降。

根据前述对当前 DCT 算法的研究现状的分析来看，不难发现当前 DCT 算法还是以基于 DA 算法的行列分离的结构来计算 2D－DCT 的方法居多。因为这样的结构易于设计，比较规则，没有很复杂的蝶型网络。虽然可能乘法的次数和加法的次数不是最少的，但是硬件实现最重要的是算法算术单元之间的数据传递的复杂性。采用行列分离(RCM)结构的硬件实现，模块比较清晰，适合自顶向下的设计方法。当然，这种结构对于高吞吐量系统来说，为了快速安排行列转换模块之间的数据流，需要复杂的中间数据转换电路(TRAM)。一般快速转换电路都需要很大的芯片面积。所以，近年已经有很多无中间转换电路的 RCM 系统，其中很多是用脉动阵列实现的。但是，采用这种算法往往要求整个 N×N 的输入数据同时参与计算，因此 I/O 处理及数据传递电路复杂，使得它们的 VLSI 实现在性能上反而不如 RCM 系统。

在第二节中，根据前一章提出的部分和分解算法，设计了 $2^n \times 2^n$ 型 DCT 结构和 $q \times q$ (q=奇素数)型 DCT 的 VLSI 结构。这种结构具有规则性，相对其他常

用 2D－DCT 算法，使用了更少的乘法器和加法器。本节最后给出了与其他常用 2D－DCT 算法的资源开销对比。

第三节详细叙述了 PSDA 算法的 VLSI 结构的设计和验证过程，给出了综合结果。

第四节给出了 PSDA 算法 IP 核的性能参数，并与其他几家公司的 IP 核进行了对比。验证结果证明了算法的正确性和高效性。

6.2 部分和分解算法的 IP 核实现

6.2.1 $2^n \times 2^n$ 型 DCT 部分和分解算法的 VLSI 结构

图 6－1 是 8×8 DCT 算法的 VLSI 实现原理框图。$2^n \times 2^n$ 型 DCT 实现方法类似于 8×8 DCT。图 1 中， 数据按时钟节拍输入；Step1－3 实现 5.2.4 中 step1 的功能。Step 4 为部分和计算单元；Step 5 储存部分和输出；Step 6 为乘加运算单元。

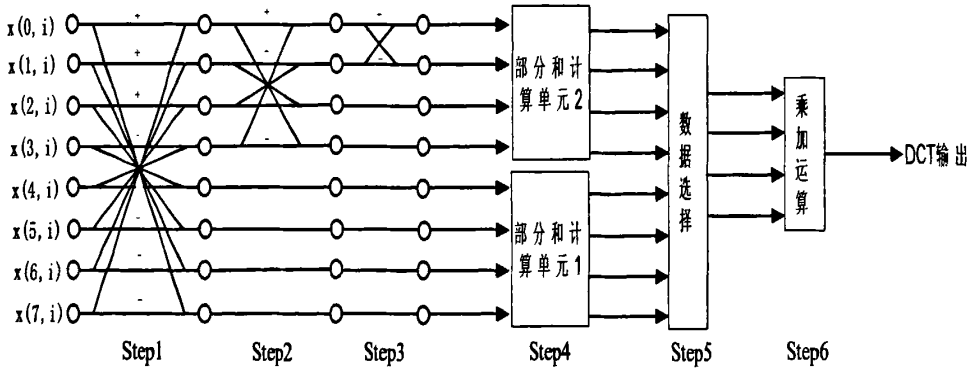


图 6－1 8×8 DCT 算法的 VLSI 实现原理框图

图 6－2 是部分和计算单元原理框图。由(5－13)可知，对于每一列输入数据，某一特定余弦因子对应的元素不会超过两个。在图 6－2 中，这两个元素可以通过两个数据选择器选出，再根据预先存储好的表格决定这两个元素的运算关系，将运算结果累加即可得到部分和输出。

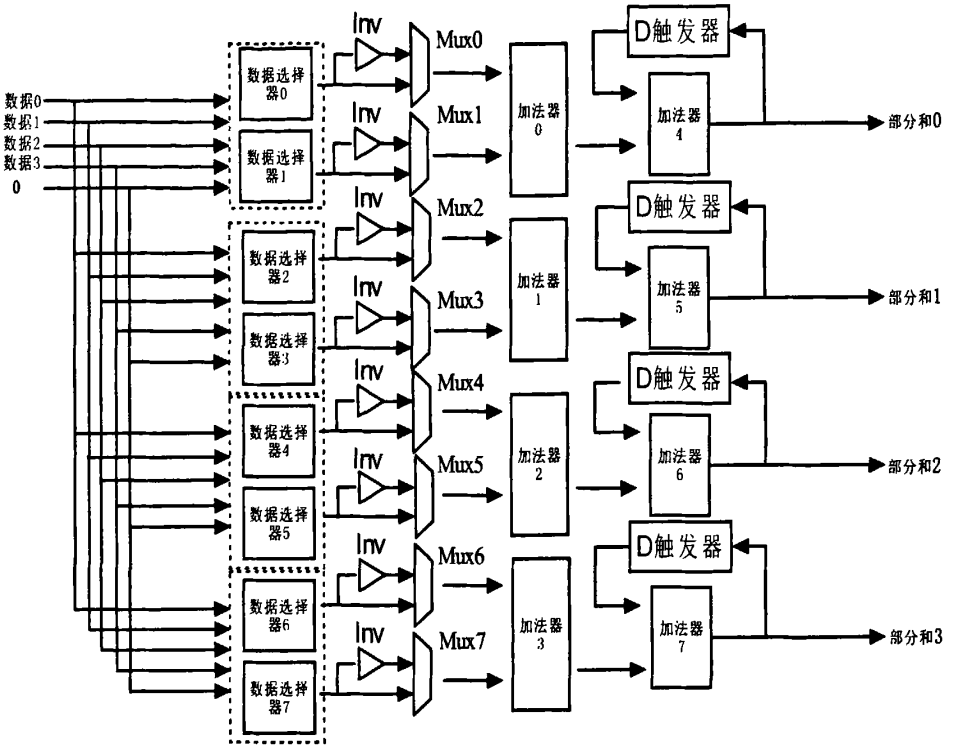


图 6—2 部分和计算单元原理框图

图 6—3 是数据选择单元原理框图。由于每 8 拍时钟输出 8 个部分和，正好可通过乘加运算计算出 8 个 DCT 输出。数据选择单元所起的功能为在 step4 中部分和计算单元 1 和部分和计算单元 2 的输出进行选择切换。再根据预先存储好的表格决定余弦因子，部分和输出到乘加单元。

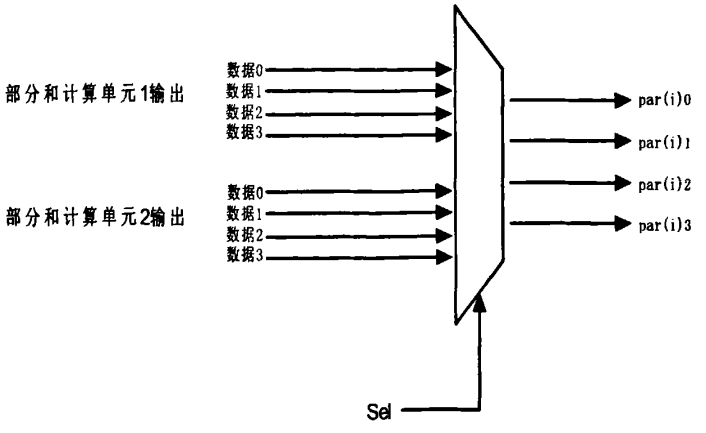


图 6—3 数据选择单元原理框图

图 6—4 是乘加单元原理框图，考虑到蝶型单元的不规则性，可直接采用乘积叠加的形式，每时钟输出一个有效数据。

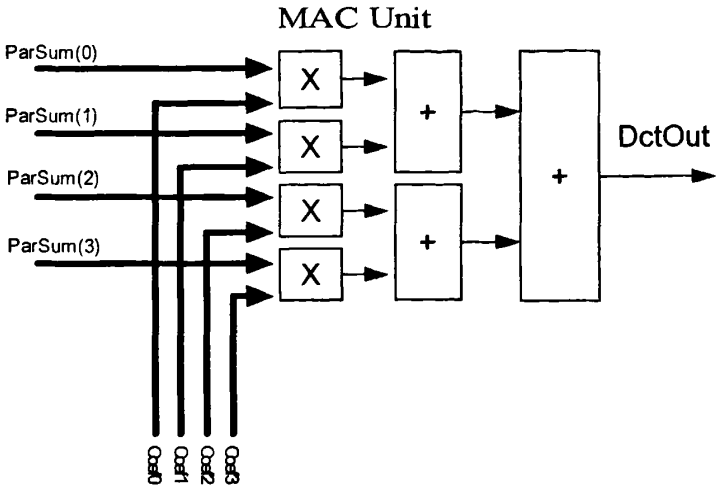


图 6—4 乘加单元原理框图

表 6—1 显示了 PSDA 算法和其他 2 维 DCT 算法的比较。Cho 的结构[37] 达到了最高的吞吐率但是对于实时系统计算复杂性太高。 Yang 的结构[40]计算复杂性很低，但是内部的连线逻辑随 DCT 的尺寸复杂呈非线性增加。该 VLSI 结构规则吞吐率高，而且乘法器和加法器的数目也很少。因此该算法具有较高的推广价值。

表 6－1 不同 2D－DCT 算法的性能比较

	Cho [30]	Lee [79]	Uramoto [80]	Yang [81]	Gong [82]	PSDA 算法 [83]
Algorithmic Approach	Direct 2－D algorithm	Direct 2－D algorithm	R－C decompose	Direct 2－D algorithm	R－C Decompose	Direct 2－D algorithm
No.of Multipliers	$\frac{N^2 * \log_2 N}{2}$	$\frac{N}{2}+1$	N	$\frac{N}{2}$	N	$\frac{N}{2}$
No.of Adders	$\frac{I}{2}(5N^2 * \log_2 N - 2N) + 2$	$\left(\frac{N}{2}\right)^2$	$2N$	$N^2+\left(\frac{N}{2}\right)^2$	$2N$	$4N+1$
Transposition	No	No	Yes	No	No	No
Total cycles per N×N 2－D DCT	1	$\frac{N}{2}\times \log_2 N$	N^2+N	$\log_2 N$	N^2	$N+\log_2 N$
I/O Ports	Parallel In Parallel Out	Serial In Parallel Out	Serial In Serial Out	Serial In Parallel Out	Parallel In Parallel Out	Parallel In Parallel/ Serial Out

6.2.2 $q\times q$ (q=奇素数)型 DCT 部分和分解算法的 VLSI 结构

图 6－5 为素数型二维 DCT 的部分和分解算法原理框图。图 6－6 给出了采用部分和分解算法计算 A(k, l)的原理框图。由式(5－22)可知，X(k, l)由 A(k, l)，B(k, l)，C(k, l)和常量之和组成。

在图 6－5 中，第(q－1)/2 行和第(q－1)/2 列数据分别经过两个素长度一维 DCT

计算单元得到 B(k，l)和 C(k，l)；其余的时域数据通过图 6－6 中 A(k，l)计算单元得到 A(k，l)。A(k，l)，B(k，l)，$C(k, l)$和常量 $x((q-1)/2, (q-1)/2)$或 0 之和即为相应的 DCT 变换输出 X(k，l)。

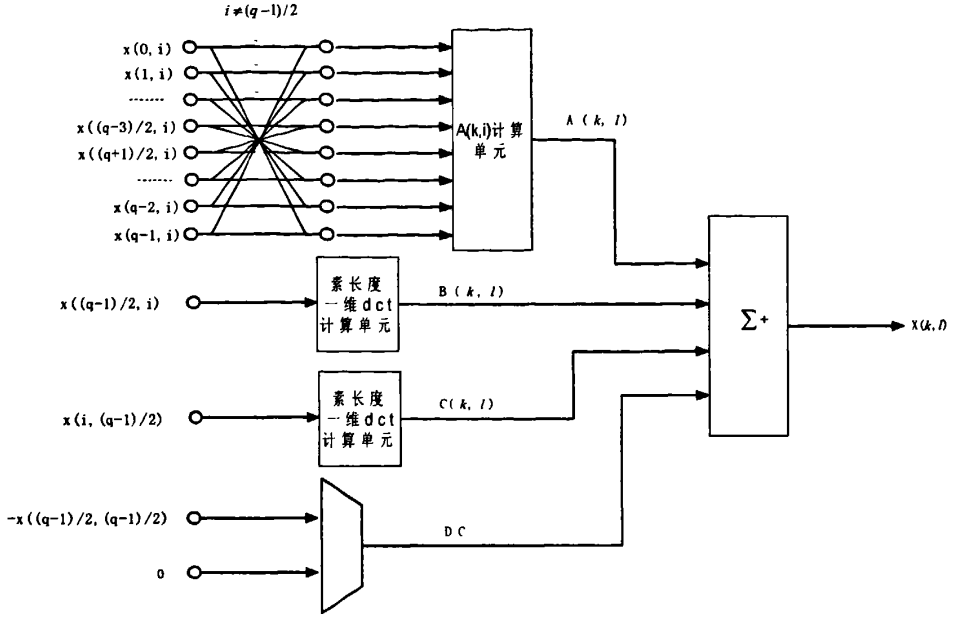


图 6－5 素数型二维 DCT 部分和分解算法原理框图

图 6－6 为 A(k，l)计算单元原理框图。A(k，l)计算单元由两个部分和计算单元，一维一个素长度 DCT 偶系数计算单元和奇系数计算单元组成。部分和计算单元结构与图 6－2 相同，也是从每列输入数据中选出两个进行部分和累加计算。

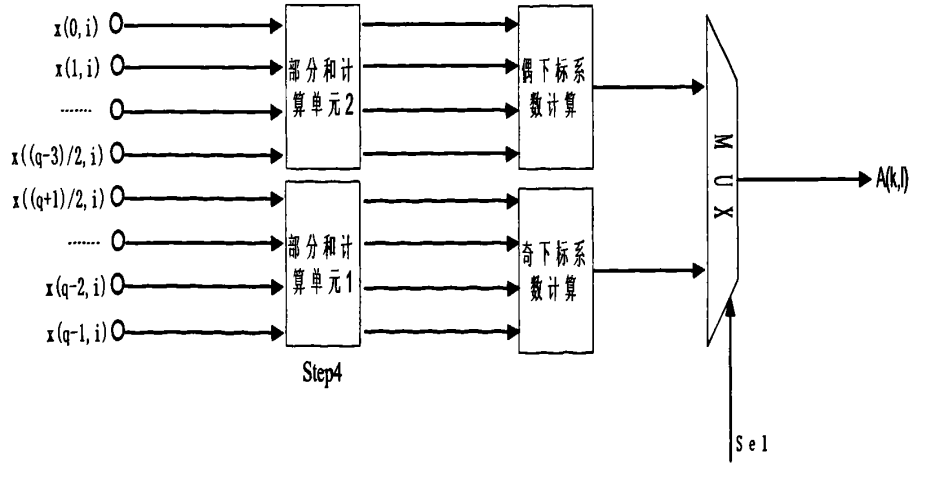


图 6－6 A(k，l)计算原理框图

6.3 部分和分解算法 IP 核实现及验证

6.3.1 IP 核设计

对于 IP 硬核的设计来说，整个设计一定要流片并且进行 demo 板测试才能交付。由于本文以研究为主，只开发了 DCT 的 IP 软核，所以只需要映射到工艺库上，进行门级仿真即可。下面为将设计综合到 SMIC 的 0.18um 工艺库上的全过程。

1)建立设计环境

建立设计环境的工作就是编写 Design Compile 的名称为.synopsys_dc.setup 的设置文件。该文件的内容如下：

```
search_path = search_path + {"." "f:/Synopsys/libraries/syn/smic_35"}
link_library = {"*", "smic18a_tt_33_25.db"}
target_library = {"smic18a_tt_33_25.db"}
symbol_library = {"SN18a.sdb"}
company = "UESTC 111 lab"
designer = "Tian Mao"
view_background = "black"
define_design_lib work －path work
```

2)读入 HDL 描述

按照设计的层次，从底到上，使用 analyze 和 elaborate 命令读入设计。读入设计可以用 DC 的图形界面 design_analyzer 或者它的文字界面 dc_shell。用 design_analyzer 读入所有设计后的 fdct_da 顶图如图 6－7a，b 所示。

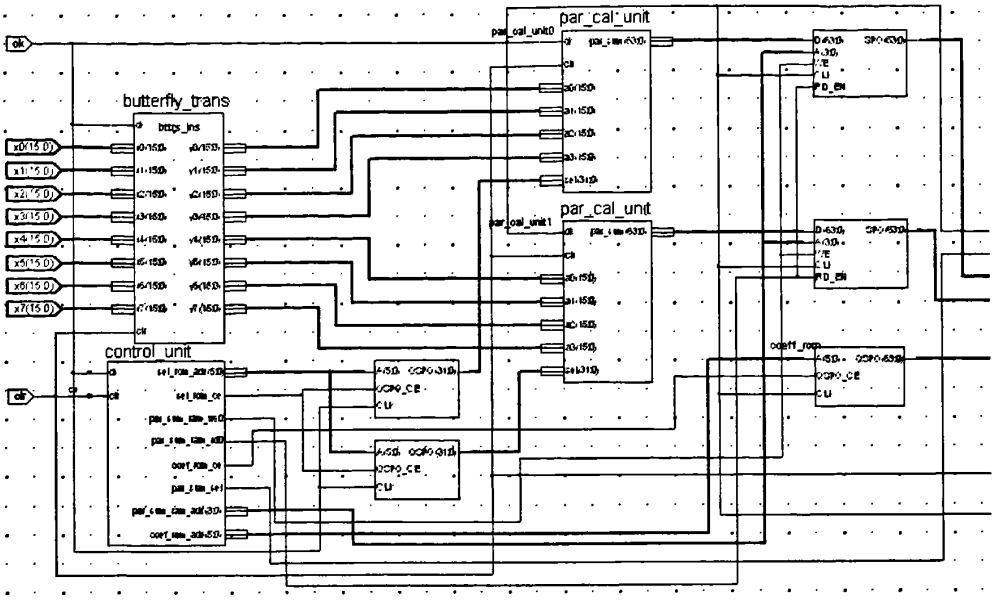


图 6－7a 部分和分解算法顶图(1)

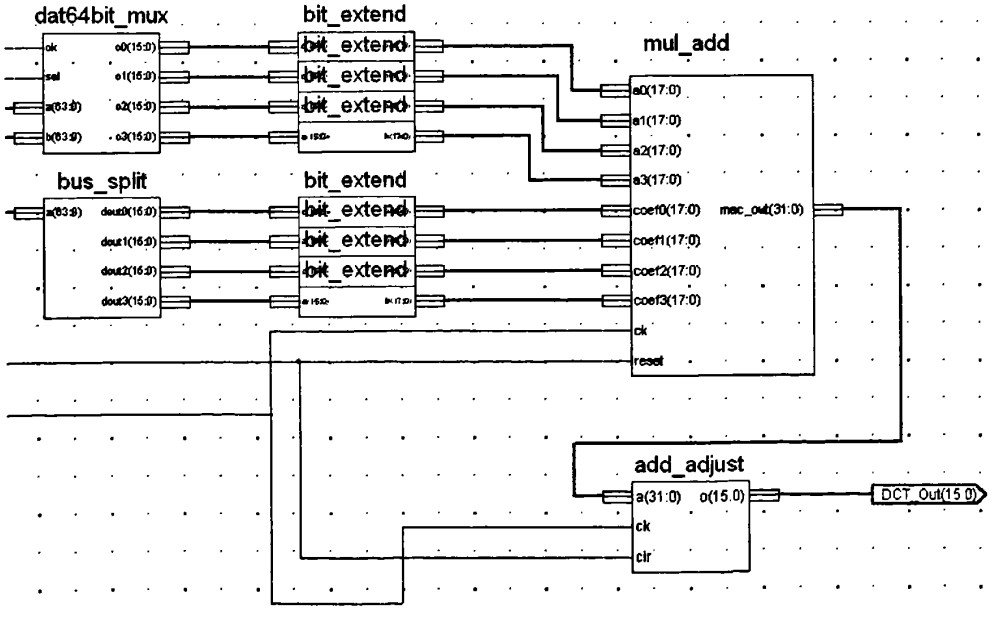


图 6－7b 部分和分解算法顶图(2)

3)设置约束

◆设置导线负载模型和模式：

set_wire_load "100K" －library "SN35A_TT_33_25"

set_wire_load_mode enclosed

◆设置时钟和复位信号

create_clock －period 15 －waveform {0 5} clk

set_clock_skew －delay 1.0 －minus_uncertainty 1.0 －plus_uncertainty 0.5 clk

set_dont_touch_network {clk rst}

◆设置驱动，由于时钟和复位网络一般在布局时候才加入，所以这里设置的驱动能力为理想驱动，也就是驱动能力无穷大。对于其他的输入信号，驱动强度设置为单元库内一个缓冲单元的输出端口驱动强度，原因是 IP 核最后要集成到 SOC 片上系统中去，与它连接的其他 IP 模块一般都是寄存器输出，或者通过一级缓冲后再联接到输入端口，距离很近，不需要非常强的驱动强度。

set_driving_cell －cell RS_BUF_B －pin O all_inputs()

set_drive 0 {clk，rst}

◆设置输出负载，同样道理，下一级 IP 核也不需要 DCT IP 核有非常大的驱动强度，所以负载设成 0.5pf 足够了。

set_load 0.5 all_outputs()

◆设置输入输出延时，这是为了保持良好的独立性，预估了前后级电路的延时，也增强了本设计的鲁棒性。

set_input_delay 5.0 －clock clk all_inputs()

set_output_delay 5.0 －clock clk all_outputs()

4)编译并写出门级网表和时延文件

使用 compile 命令进行编译，然后使用 write 命令将综合的结果以 db 的格式保存起来，同时还要输出 Vhdl 格式的网表和.sdf 文件为门级仿真做准备。命令使用如下：

write －hierarchy －output fdct_da.db

write －format verilog －hierarchy －output fdct_da.vhd

write_timing －format sdf－v2.1 －output active_design+".sdf"

5)给出报告

使用 report 系列命令可以产生设计的详细报告，包括面积与时序信息，当然也可以使用 PrimeTime 进行更为详细的时序分析。

6)进行门级仿真

在 ModelSim 中编译输出网表.vhd 文件，工艺库对应的.vhd 文件，testbench 文件和反标门级延时就可以进行门级仿真，实验显示结果正确。图 6－8 显示了由于加入了门级延时以后信号相对于时钟的延时。

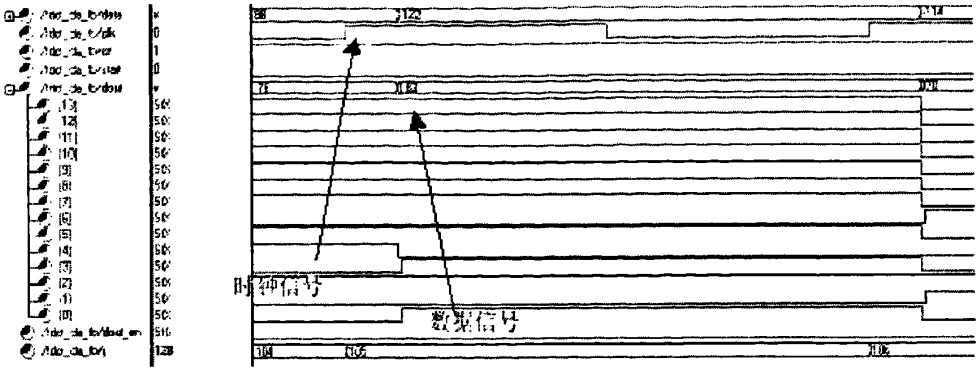


图 6—8 加入了门级延时以后信号相对于时钟的延时

7)综合和仿真结果

表 6—2 给出了 PSDA 算法的综合和仿真结果。附录 3 为 PSDA 算法 IP 核的交付件；附录 4 为 PSDA 算法 IP 核的面积测试结果；附录 5 为 PSDA 算法 IP 核的时序测试结果；附录 6 为 PSDA 算法 IP 核的功耗测试结果。附录 7 为输入数据和 DCT 变换输出仿真波形。

表 6—2 IP 核综合和仿真结果

名称	速度	面积	功耗
PSDA DCT IP	150.83MHz	997363 平方微米	94mW

以上综合和仿真结果表明，PSDA 算法有着良好的性能和推广价值。

6.3.2 FPGA 原型验证

图 6—9 为 FPGA 原型验证原理测试原理图。通过对比软件和硬件变换的输出数据及生成的 JPEG 图像，我们可以测试算法 IP 核的功能正确性和处理性能参数。

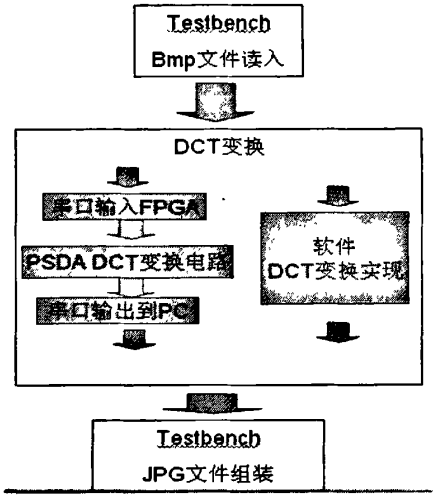


图 6－9 FPGA 原型验证系统测试原理图

图 6－10 为 FPGA 原型验证系统原理框图。由图 6－10 我们可以看到，DCT 验证系统可以分为以下 4 个模块，UART，控制模块，存储器，用户算法验证模块。各模块的功能如下：

UART： 实现异步串行通信功能，完成 PC 与原型验证系统间的数据通信。在 PC 端，通过软件对硬件的 DCT 变换输出进行 Huffman 编码，生成 JPEG 文件来验证 PSDA DCT 算法 IP 核的正确性。

存储器：存储用户算法的输入/输出数据

PSDA DCT：PSDA DCT 算法 IP 核，完成 DCT 变换

控制单元：控制其他模块的启动/停止；数据读取/写入

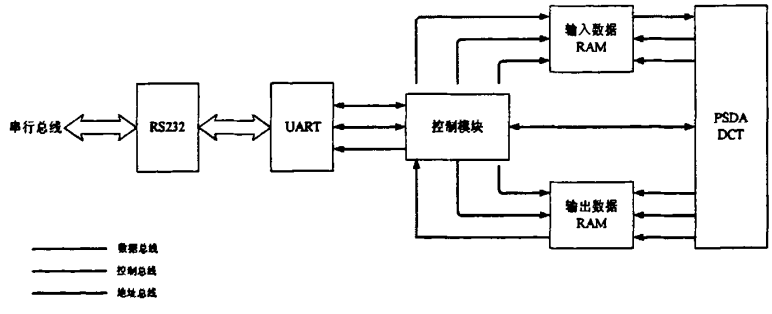


图 6－10 FPGA 原型验证系统原理框图

图 6－11 为 FPGA 原型验证系统所使用的 Altera 开发板。

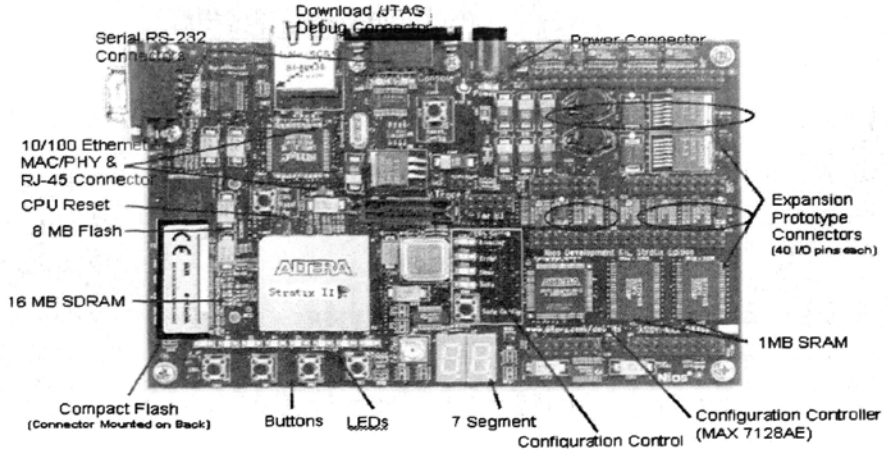


图 6－11 FPGA 原型验证系统板

6.4 本设计与业界销售产品的性能比较

与 Amphion 公司和 Barco Silex 公司的 DCT 硬核比较如表 6－3 所示。

表 6－3 本设计与业界销售产品的性能比较

公司	工艺	逻辑门数	性能	所需额外资源
Amphion	TSMC 0.18um	34k	217 MHz	1kbitram
Barco Silex	UMC 0.18um	28k	165 MHz	2 个 ram
本设计	SMIC 0.18um	16k	150 MHz (门级)	1kbitrom

由表 6－3 可知，在这几个产品中，本设计所用的面积是最小的。系统的瓶颈主要在乘法器的运算速度上。由于使用了组合乘法器，导致了系统的性能上要稍低于以上两家公司的产品。如果采用 DA 算法来实现乘法器的话，系统的性能还会有更大的提升。该 DCT 软核的性能可以达到 JPEG，H.263，MPEG 等图像视频编码标准的要求。

6.5 本章小结

本章讲述了二维 DCT 部分和分解算法的 VLSI 实现。本章主要研究各种长度情况下二维 DCT 的 VLSI 实现算法。主要讨论了 $2^n \times 2^n$ 类型和 $q \times q$(q 为奇素数)

类型的 2D－DCT 算法的 VLSI 实现。该硬件实现算法具有结构简单，计算复杂性低，加法器和乘法器开销小等优点。

本章还叙述了 ASIC 的基本开发流程，在基于 Synopsys 公司的 Design Compiler 软件上进行了部分和分解算法的 ASIC 实现和验证。

部分和分解算法通过了 ASIC 验证。8×8 DCT/IDCT 核的综合结果证明了算法的正确性。系统的处理延时为 12 个时钟，吞吐率能达到时钟频率倍符号/秒。系统的主要瓶颈在于最后一级乘加器的运算速度上。最后一级的乘加累积运算也可以采用 1D－DCT 的 IP 核来替代，系统的吞吐率会增加到原来的 N/2 倍，但是系统的结构也会变得不规则，且面积开销将会大大增加。

第七章 结论与展望

本文主要研究二维 DCT 快速算法及其 VLSI 实现结构。本文首先介绍了离散余弦变换(DCT)的概念；介绍了应用背景，实现原理和未来发展趋势；指明了 DCT 快速算法发展中面临的主要问题，重点说明了 DCT 快速算法所要解决的关键技术问题，综述了各种一维和二维的 DCT 快速算法的发展概况。然后叙述了一维离散余弦变换(DCT)的快速变换算法设计。一维 DCT 快速变换算法是所有 DCT 快速变换算法的基础。本文详细介绍了 2^n，q^n，素长度以及任意长度 DCT 变换算法。在研究了一维 DCT 快速变换算法后，本文研究了二维离散余弦变换的快速算法。利用数论理论，本文提出了二维 DCT 变换的部分和分解算法快速算法－PSDA。PSDA 算法是一种直接分解算法，其核心思想为将二维 DCT 变换直接转换为若干个一维 DCT 变换实现。根据同余理论，将频域输出划为若干个子集，每一子集内的所有元素构成一个一维 DCT 变换的输出。本文提出了变换输出的子集划分准则，部分和的计算方法，公共加法项的合并原则并把部分和算法从 $2^n \times 2^n$ 扩展到 $q \times q$ (q 为奇素数)型二维 DCT。

在研究了 PSDA 算法后，本文研究了该算法的 VLSI 实现。SOC 开发的主要工作就是建立具有自主知识产权的 IP 核库。鉴于 DCT 变换在图像和视频信号压缩的变换编码中，被广泛认为是最高效的一种方法。所以，研究和开发具有自主知识产权的 DCT 模块，对于未来图像处理系统的 SOC 开发具有非常重大的意义，同时也拥有一定的商业价值。基于以上的思路，本文在第二章概述了国内外各种离散余弦变化(DCT)的快速算法及其 VLSI 实现的硬件结构。在此基础上，提出了基于部分和分解算法的全流水线 DCT 硬件结构。该结构的核心是部分和的计算。最终的变换输出既可以通过乘加运算求和输出，也可以使用其他一维 DCT 变换模块输出。由 PSDA 算法可知，对于二维 DCT 的每一列输入数据，某一特定余弦因子对应的时域元素不会超过两个。这两个元素可以通过两个数据选择器选出，再根据预先存储好的表格决定这两个元素的运算关系，将运算结果累加得到部分和输出。DCT 变换输出既可以通过部分和的乘加运算求和输出，也可以使用其他一维 DCT 变换模块输出。本文分别给出了 $2^n \times 2^n$ 型和 $q \times q$ 型 DCT 的 VLSI 结构并将验证后的 IP 核与其他公司的产品进行了比较分析。比较结果证明 PSDA 算法有着良好的性能，可作为图像处理器芯片的核心算法在视频信号处理中得到广泛的

应用。

本文在这些方面的研究和主要贡献总结如下：

1 提出了部分和相等的分集准则，通过该准则实现了对频域输出数据的分集，并证明了每一个分集内的元素为时域输入数据部分和的一维 DCT 变换输出。

2. 通过频域数据的分集，将一个二维 DCT 变换转换为若干个一维 DCT 变换，使完成 DCT 变换的乘法运算次数减少了一半。

3. 提出了部分和的计算方法和公共加法项的合并原则，通过合并公共加法项得到了更少的加法运算量。

4. 传统的 DCT 算法是针对输入数据长度为 2^n 进行计算的，但是在很多应用领域中都要用到长度非 2^n 的 DCT 算法，而素长度的 DCT 算法是非 2^n 长度 DCT 的核心。因此本文在 2^n 长度 PSDA 算法的基础上提出了改进的二维素长度 PSDA DCT 算法，该算法是基于频域输出数据与部分和的转换和映射关系，将二维素长度 DCT 变换分解为多个一维素长度 DCT 变换。同已有的素长度 DCT 算法相比，减少了一半的乘法计算量。

5. 提出了基于 2^n 长度 PSDA 算法的 VLSI 结构，同间接算法的 VLSI 结构相比，该结构具有不需要转置变换，处理延时低的优点；同其他直接算法的 VLSI 结构相比，该结构具有更规则的结构，和更少的乘法器和加法器开销。

6. 目前的 DCT IP 核都是针对 2^n 长度的，并不能满足完全实际应用中的需要。因此，本文在 PSDA 算法的基础上，分别基于 FPGA 和 ASIC 工艺进行了素数长度 DCT 的 IP 核设计。仿真和综合结果表明，该设计结构简单、层次清晰，具有高度的规则性和模块性。

7. 比较分析了基于 PSDA 算法的 DCT IP 核与业界同类产品的优劣，为以后进一步开发 IP 核库打下坚实的基础。

本文所做的研究工作还可以在以下方面深入进行：

1. 部分和分解算法还可以在任意尺寸二维 DCT 快速变换算法设计中得到应用。

2. 部分和分解算法还可以在高维 DCT 快速变换算法设计中得到使用。

3. 随着 CMOS 工艺特征尺寸的不断缩小，和对功耗提出的新要求，PSDA 算法的 IP 核可以利用门控时钟等低功耗技术进一步加强性能，使之能适应更广阔的应用场合。

4. 当条件成熟时，本设计可以从软核转化成硬核，直接作为图像处理核心模块集成到 SOC 片上系统中去。

攻博期间取得的研究成果

[1] Mao Tian, Guangjun Li, Qicong Peng. Fast Algorithm for 2－D Discrete Cosine Transform. The Chinese Journal of Electronics, 2007, 4(16): 337－341 (SCI: 160TL, EI: 071810582422)

[2] 田茂,李广军,彭启琮.基于部分和分解的素长度二维DCT快速算法.电子与信息学报 (已录用)

[3] Mao Tian,Guang－Jun Li,Qi－Cong Peng.A New Fast Algorithm for 8x8 2－D DCT and Its VLSI Implementation. IEEE IWVDVT2005: 179－182 (EI: 05349315602, ISTP:BDH63)

[4] Mao Tian,Guangjun Li, Qicong Peng. Partial Sum Based Algorithm for the Discrete Cosine Transform. IEEE ITSTP2006:1094－1097 (EI:082311298623, ISTP:BFG07)

[5] Mao Tian,Bo Yan,Guangjun Li,Jun He.High Performance Static Image Compression on DSP.电子科技大学学报,2004,33(4): 387－390 (EI: 04448438808)

[6] 田茂,夏刊.TS流复用的软件实现.第十三届国际有线电视技术研讨会,2005:469－476

[7] 田茂,李广军,林水生,包孔林. M68HC912D60 MCU在税务监控系统设计中的应用半导体技术,2003,28(6): 60－62

[8] 田茂,李广军,林水生.一种MSK信号数字中频解调算法设计与实现.四川省通信学会2003年学术年会

[9] 闫波,田茂,李广军.一种基于层次模型的USB2.0接口芯片IP核固件的设计与实现.半导体技术, 2004,29(6): 76－79

[10] 田茂,李广军,彭启综.一种信号盲分离中的变步长算法.已投《信号处理》

[11] 田茂,李广军,彭启综.脉冲噪声随机信号的产生方法.已投《电子信息与技术》

致 谢

本论文是在我的导师李广军教授和彭启琮教授的悉心指导下完成的。李老师和彭老师学识渊博，治学态度严谨，为人谦和，诲人不倦，渊博谦逊。四年中始终给予我宽松的学习环境和悉心的教诲。四年后，在此论文定稿之际，特向恩师致以崇高的敬意和衷心的感谢。

同时向林水生教授，闫波副教授表示深深的感谢与敬意。林老师，闫老师学识渊博，治学严谨，学术思想活跃，为人谦和，待人热情，是我终身的良师益友。

还要感谢本教研室的郭志勇老师，周亮老师、王玉林老师、周英乙老师、杨海芬博士等同学，同他们的交流和讨论使我在很多方面获益匪浅。

感谢我的父母和妻子，在多年的求学生活中，他们任劳任怨，为我的成长付出了难以估计的血汗。他们殷切的目光一直激励着我不断向前，我取得的所有成绩都离不开他们的支持和鼓励。

再次衷心感谢所有关心、爱护、支持和帮助过我的师长、亲人和朋友们。

田 茂

2008 年 8 月

参考文献

[1] Ahmed.N, Tnatarajan and Rao.K.R. Discrete cosine transform. IEEE Trans Computer,1974, 23(1):1974,23(1):90-93

[2] Chen.W.H, Smith.C.H, Fralick.S.C.A fast computational algorithm for the discrete cosine transform. IEEE Transactions on Communications,1977,25(9):1004-1009

[3] Harrlick.M.T.A storage efficient way to implement discrete cosine transform. IEEE Transa -ctions on Computer,1976,25(6):764-765

[4] Narashima.M.J,Peterson.A.M.On computation of the discrete cosine transform. IEEE Trans actions on Communications,1978,26(6):934-946

[5] Tsang.B.D, Miler.W.C.On computing the discrete cosine transform. IEEE Transactions on Computer, 27(10):966-968

[6] Markhou.L.J.A fast cosine transform in one and two dimensions. IEEE Transactions onA coustics, Speech and Signal Processing,1980,28(1):27-34

[7] Vetterli.M, Nussbaomer.H.Simple FFT and DCT algorithm with reduced number of opera- tions. Signal Processing,1984,6(4):267-278

[8] Lee.B.G. A new algorithm to compute the discrete cosine transform. IEEE Transactions on Acoustics, Speech and Signal Processing,1984,32(12):1243-1245

[9] H.S.Hou. A fast recursive algorithm for computing the discrete cosine transform. IEEE T ransactions on Acoustics, Speech and Signal Processing,1987,35(10):1445-1461

[10] Chan.S.C, Ho.K.L.A new two-dimensional fast cosine transform algorithm. IEEE Transac tions on Signal Processing,1991,39(2):481-485

[11] Wu.H.R, Paoloni.F.J.A two-dimensional fast cosine transform algorithm based on Hou's approach. IEEE Transactions on Signal Processing,1991,39(2):544-546

[12] Arguello.F, Zapata.E.L.Fast cosine transform based on the successive doubling method. Electronics Letters,1990,26(20):1616-1618

[13] Britanak.V. On the discrete cosine transform computation. Signal Processing,1994, 40(2): 184-194

[14] Britanak.V. A unified discrete cosine and discrete sine transform computation. Signal Processing, 1995, 43(5):333-339

[15] Skodras.A.N. Fast discrete cosine transform pruning. IEEE Transactions on Signal Processing, 1994, 42(7):1833-1837

[16] Yang.P.N, Narasimha.M.J, Lee.B.G. A prime factor decomposition algorithm for the discrete cosine transform. Proceedings of ICCSSP, 1984, India:132-135

[17] Yang.P.N, Narasimha.M.J. Prime factor decomposition of discrete cosine transform and its hardware realization.Proceedings of IEEE International Conference onASSP,1984,USA: 772-775

[18] Wang.F.M, Yip.P. Fast prime factor decomposition algorithms for a family of discrete trigonometric transform. Circuits, System and Signal Processing,1989,8(4):401-419

[19] Lee.B.G. Input and output index mapping for a prime-factor-decomposed computation of discrete cosine transform. IEEE Transactions on Acoustics, Speech and Signal Processing,1989, 37(2):237-244

[20] Chakrabarti.C, Jaja.J. Systolic architectures for the computation of the discrete Hartley and the discrete cosine transforms based on prime factor decomposition. IEEE Transactions on Computer,1990,39(11):1359-1368

[21] Lee.P.Z, Huang.F.Y. An efficient prime-factor algorithm for the discrete cosine transform and its hardware implementations. IEEE Transactions on Signal Processing,1994,42(8):1996-2005

[22] Heideman.M.T. Computation of an odd-length DCT from a real-valued DFT of the same length. IEEE Transactions on Signal Processing,1992,40 (1):54-61

[23] Chen.Y.H, Siu.W.C. Algorithm for prime length discrete cosine transforms. Electronics Letters, 1990,26(3):206-208

[24] Chen.Y.H, Siu.W.C. On the realization of DCT using the distributed arithmetic. IEEE Transactions on Circuits and Systems-I:Fundamental Theory and Applications,1992, 39(9):705-712

[25] Guo.J.I, Liu.C.M, Jen.C.W. A new array architecture for prime length discrete cosine transform. IEEE Transactions on Signal Processing, 1993, 41(1):436-442

[26] Kamangar.F.A, Rao.K.R. Fast algorithm for the 2-D discrete cosine transform. IEEE transactions on computer, 1982, 31(9): 899-906

[27] Haque.M.L. A two-dimensional fast cosine transform. IEEE Transactions on Acoustics, Speech and Signal Processing,1985,33(6):764-765

[28] Nasrabadi.N, King.R.Computationally efficient discrete cosine transform algorithm. Electronics Letters,1983,19(1):24-25

[29] Vetterli.M. Fast 2-D discrete cosine transform. Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing, 1985, USA, 1538-1541

[30] Cho.N.I, Lee.S.U. Fast algorithm and implementation of 2-D discrete cosine transform. IEEE Tansactions on Circuits and Systems,1991,38(3):297-305

[31] Cho.N.I., Lee.S.U. A fast 4×4 algorithm for recursive 2-D DCT, IEEE.Transactions on Signal Processing,1992,40(9):2166-2172

[32] Cho.N.I, Yun.I.D, Lee.S.U. On the regular structure for the fast 2-D DCT algorithm. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing,1993,40(4):259-266

[33] Huang.Y.M., Wu.J.L. A refined fast 2-D discrete cosine transform algorithm. IEEE Transactions on Signal Processing,1999, 47(3):904–907

[34] Duhamel.P, Guiliemot.C. Polynomial transform computation of the 2-D DCT, Proceedings IEEE International Conference on Acoustics,Speech and Signal Processing,1990:1515-1518

[35] Tan, Attikiouzel.Y, Grebbbin.G. Fast computation of two-dimensional discrete cosine transforms using discrete radon transform. Electronics Letters,1991,27(1):82-84

[36] Goertzel.G. An algorithm for the evaluation of finite trigonometric series. American Mathematics Monthly, 1958, 56(1): 34-35

[37] Curtis.T.E, Wickenden.J.T. Hardware-based Fourier transform: algorithms and architectures. IEE Proceedings-Part F, 1983, 130(5): 423-432

[38] Chau.L.P, Siu.W.C. Direct formulation for the realization of discrete cosine transform with general length. IEEE Transactions on Circuits and System-II: Analog and Digital Signal Processing, 1995, 42(1):50-52

[39] Jain.A.K. A fast Karhumen-Loeve transform for a class of stochastic process. IEEE Transactions on Communications, 1976, 24(6): 1023-1029

[40] Chiu.C.T, Liu.K.J.R. Real-time parallel and fully piplined teo dimensional DCT lattice structure with application to HDTV system. IEEE Transactions on Circuits and System for Video Technology,1992,2(1):25-37

[41] Liu.K.J.R, Chiu.C.T. Unified parallel lattice structures for time-recursive discrete cosine/sine/ Hartley transform. IEEE Transactions on Signal Processing, 41(3): 1357-1377

[42] Liu.K.J.R, Chiu.C.T, Kolagotla.R.K and Jaja.J.F. Optimal unified architectures for the real-time computation of time recursive discrete sinusoidal transform. IEEE Transactions on Circuits and Systems for Video Technology,1994,4(2):168-180

[43] Padmanabhan.M, Martin.K. Filter banks for time-recursive implementation of transforms. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, 1993, 40(1):41-50

[44] Peceli.G. A common structure for recursive discrete transforms.IEEE Transactions on Circuits and Systems, 1986, 33(10):1035-1038

[45] Vetterli.M, Nussbaumer.H.J. Simple FFT and DCT Algorithms with Reduced Number of Operations. Signal Processing,1984, 6: 267-278

[46] 朱跃生, 马维祯. 利用快速 W 变换计算离散 sine 变换及离散 cosine 变换. 数据采集与处理,1990,5(3): 9-12

[47]. W.H.Chen, C.H.Smith and S.C.Frlick. A fast computational algorithm for the discrete cosine transforms. IEEE Trans.Commun.1997, Sep, Vol.COM-25: 1004-1009

[48] Byeong Gi Lee. A New Algorithm to Compute the Discrete Cosine Transform.IEEE Trans. Signal proc.1984, Dec, Vol.ASSP-32, No.6. pp: 1243-1245

[49] Hou.H.S. A fast recursive algorithm for computing the discrete cosine transform. IEEE Trans, Acoust, Speech, Signal Processing,1987, ASSP-35(10):1455-1461

[50] 马维祯, 殷瑞祥. DFT(2^m)和 DCT(2^m)的快速递归新算法.中国电子学会电路与系统年会论文集, 深圳, 1987, 26:20-23

[51] 王中德. 计算离散 cosine 变换的一种算法. 中国电子学会电路与系统年会论文集, 深圳, 1987, 26:9-12

[52] Feig.E, Winograd.S. On the Multiplicative Complexity of Discrete Cosine Transforms. IEEE Trans,Inform,Theory, 1992,38 (4):1387-1391

[53] Loeffler.C, Ligtenberg.A. G.S.Moschytz, Practical fast 1-D DCT algorithms with 11 multiplications. Proc.IEEE ICASSP 89,1989:988-991

[54] 陈禾, 毛志刚, 叶以正. DCT 快速算法及其 VLSI 实现. 信号处理, 1998,14: 62-70

[55] Madisetti.A, Willson.A.N. A 100MHZ 2-D 8×8 DCT/IDCT processor for HDTV applications. IEEE Trans, Circuits Syst, Video Technol,1995,5(2):158-164

[56] Peled.A, Liu.B. A new hardware realization of digital filters. IEEE Trans, Accoust. Speech, Signal Process, 1974, ASSP-22(6):456-462

[57] Sun.M.T, Chen.T.C, Gottlieb.A.M. VLSI implementation of a 16×16 discrete cosine transform. IEEE Trans,Circuits Syst,1989,CAS-36(4):610-617

[58] Uramoto.S, Inoue.Y, Takabatake.A, Takeda.J, Yamashits.Y, Terane. H and Yoshimoto.M. A 100 MHZ 2-D discrete cosine transform core processor. IEEE J.Solid-State Circuits,1992,27(4): 492 -499

[59] Matsui.M, et al, A 200MHZ 13mm^2 2-D DCT macro-cell using sense-amplifying pipeline flip-flop scheme. IEEE J.Solid-State Circuits,1994,29(12)

[60] Sungwook.Yu, Swartzlander.Earl E. DCT Implementation with Distributed Arithmetic. IEEE Trans Comput, 2001,50(9):985-991

[61] Ahmed Shams et al. A low power high performance distributed DCT architecture. Proc. IEEE ISVLSI'02, 2002

[62] Kung.H.T. Why systolic architecture. Computer,1982,15(1):37-46

[63] Cho.N.L and Lee.S.U. DCT algorithms for VLSI parallel implementations. IEEE Trans. Acoust. Speech, Signal Processing,1990,38:121-127

[64] Murthy.N.R, Swamy.M.N.S. On the real-time conputation of DFT and DCT through systolic architectures. IEEE Trans.SP,1994,42(4)

[65] Chang.L.W, Wu.M.C. A unified systolic array for discrete cosine and sine transforms. IEEE Trans.SP,1991,39:192-194

[66] Wang.C.L, Chen.C.Y. High-throughput VLSI architecture for the 1-D and 2-D discrete cosine transforms. IEEE Trans, Circuits Syst. Video Technol,1995,5(1):31-40

[67] Wang.C.L and Chen.C.Y. A Linear systolic array for the 2-D discrete cosine transform. Proc. IEEE Asia-Pacific Conf. Circuit Syst, 1994:73-78

[68] Chang.Y.T, Wang.C.L. New systolic array implementation of the 2-D discrete cosine transform and its inverse. IEEE Trans, Circuit Syst. Video Technol,1995,5(2):150-157

[69] Wu.C.M, Chiou.A. A SIMD-systolic architecture and VLSI chip for the two-dimensional DCT and IDCT. IEEE Trans,CE,1993,39(4):859-869

[70] Volder.J.E. The CORDIC trigonometric computing technique. IRE Trans on electronic computers,1959,EC-8(3):330-334

[71] Duh.W.J, Wu.J.L. Constant-rotation DCT architecture based on CORDIC techniques. INT.J. ELECTRONIC,1990,69(5):583-593

[72] Mariatos.E.P, Metafas.D.E, Hallas.J.A, Goutis.C.E. A fast DCT processor based on special purpose CORDIC rotators. 1994 IEEE International Symposium on Circuit and Systems,1994, vol(6):271-274

[73] Zhou.F, Kornerup.P. High speed DCT/IDCT using a pipeline CORDIC algorithm(1063-6889/ 1995 IEEE),1995,vol(7):180-187

[74] Chiu.C.T, Liu.K.J.R. Real-time parallel and fully pipeline two-dimensional DCT lattice structures with application to HDTV systems. IEEE Trans,Circuit Syst. Video Technol,1992,2(1): 25-36

[75] Srinivasan.V and Liu.K.J.R. VLSI design of high-speed time-recursive 2-D DCT/IDCT proce-

ssor for video applications. IEEE Trans,Circuit syst. Video Technol,1996,6(1)

[76] Feig.E. A fast scaled DCT algorithm. Proc. SPEC Int. Soc. Opt. Eng,1990,vol(1244):2-13

[77] Hsia.S.C, Lu.B.D, Yang.J.F, Bai.B.L. VLSI implementation of parallel coefficient-by-coefficient two-dimensional IDCT processor. IEEE Trans, Circuits Syst. Video Technol,1995,5(5): 396-406

[78] 华罗庚，数论导引. 高等教育出版社, 1977

[79] Lee. J, Vijaykrishnan.N, Irwin.M.J, Radhakrishnan.R.Inverse Discrete Cosine Transform Architecture Exploiting Sparseness and Symmetry Properties.IEEE Transactions on Circuits and Systems for Video Technology,2004,Vol.14:361-366.

[80] Uramoto.S, et al. A 100-MHz 2-D Discrete Cosine Transform Processor. IEEE Journal of Solid-State Circuits,1992,Vol.27:492-499.

[81] Yang.J, Bai.B, Hsia.S.An Efficient Two-Dimensional Inverse Discrete CosineTransform Algorithm for HDTV Receivers.IEEE Transactions on Circuits and Systems for Video Technology, 1995,Vol.5:25-30

[82] Gong.D, He.Y, and Cao.Z. New Cost-Effective VLSI Implementation of a 2-D Discrete Cosine Transform and Its Inverse. IEEE Transactions on Circuits and Systems for Video Technology, Vol.14:405-415

[83] Mao Tian, Guangjun Li, Qicong Peng. Fast Algorithm for 2-D Discrete Cosine Transform. the Chinese Journal of Electronics,2007,Vol.56(2):337-341

[84] Mao Tian, Guangjun Li, Qicong Peng. A New Fast Algorithm for 8x8 2-D DCT and Its VLSI Implementation. Proceedings of the 2005 IEEE International Workshop on VLSI Design and Video Technology (IWVDVT2005):175-179.

[85] Mao Tian,Guangjun Li,Qicong Peng. Partial Sum Based Algorithm for the Discrete Cosine Transform. 2006 6th International Conference on ITS Telecommunication Proceedings: 1094-1097

附 录

附录 1：4×4 点 PSDA DCT C 语言代码

```
#define SQRT2 1.4142
#define PI      3.14159
void f2dct_4x4(int * in_dat, double *out_dat)
{
// total cost
// add operation times:
//   pre-processing 32
//   processing     38
// multiply times: 14
// shift times:       12
     int pre_dat[16];
      int sum_ab,sub_ab,sum_cd,sub_cd;
     double adder_out1,adder_out2;
     double mul_out1,mul_out2;
     //pre-processing
     pre_processing(in_dat,pre_dat,4);
      sum_ab=pre_dat[0]+pre_dat[1];
     sub_ab=pre_dat[0]-pre_dat[1];
      sum_cd=pre_dat[4]+pre_dat[5];
     sub_cd=pre_dat[4]-pre_dat[5];
// compute X(0,0), X(0,2), X(2,0), X(2,2)
     *out_dat=(sum_ab+sum_cd)/2.0;
     *(out_dat+2)=(sub_ab+sub_cd)/4.0;
     *(out_dat+8)=(sum_ab-sum_cd)/4.0;
     *(out_dat+10)=(sub_ab-sub_cd)/8.0;
// compute X(1,1), X(1,3), X(3,1), X(3,3)
```

```
    sum_ab=pre_dat[15]+pre_dat[10];
    sub_ab=pre_dat[15]-pre_dat[10];
    sum_cd=pre_dat[14]+pre_dat[11];
    sub_cd=pre_dat[14]-pre_dat[11];
    mul_out1=(sub_ab+sum_cd)*SQRT2/16.0;
    mul_out2=(sub_ab-sum_cd)*SQRT2/16.0;
    *(out_dat+5)=sum_ab/8.0+mul_out1;
    *(out_dat+15)=sum_ab/8.0-mul_out1;
    *(out_dat+7)=mul_out2-sub_cd/8.0;
    *(out_dat+13)=mul_out2+sub_cd/8.0;
// compute X(0,1), X(2,1), X(0,3), X(2,3)
    sum_ab=pre_dat[3]+pre_dat[7];
    sub_ab=pre_dat[3]-pre_dat[7];
    sum_cd=pre_dat[2]+pre_dat[6];
    sub_cd=pre_dat[2]-pre_dat[6];
    mul_out1=sum_ab*cos(PI/8);
    mul_out2=sum_cd*cos(3*PI/8);
    adder_out1=mul_out1+mul_out2;
    adder_out2=mul_out1-mul_out2;
    *(out_dat+1)=(adder_out1/2.0)*SQRT2/2;
    *(out_dat+3)=adder_out2/2.0-*(out_dat+1);
    mul_out1=sub_ab*cos(PI/8);
    mul_out2=sub_cd*cos(3*PI/8);
    adder_out1=mul_out1+mul_out2;
    adder_out2=mul_out1-mul_out2;
    *(out_dat+9)=(adder_out1/2.0)*SQRT2/4.0;
    *(out_dat+11)=adder_out2/4.0-*(out_dat+9);
// compute X(1,0), X(1,2), X(3,0), X(3,2)
    sum_ab=pre_dat[12]+pre_dat[13];
    sub_ab=pre_dat[12]-pre_dat[13];
    sum_cd=pre_dat[8]+pre_dat[9];
    sub_cd=pre_dat[8]-pre_dat[9];
```

```
    mul_out1=sum_ab*cos(PI/8);
    mul_out2=sum_cd*cos(3*PI/8);
    adder_out1=mul_out1+mul_out2;
    adder_out2=mul_out1-mul_out2;
    *(out_dat+4)=(adder_out1/2.0)*SQRT2/2;
    *(out_dat+12)=adder_out2/2.0-*(out_dat+4);
    mul_out1=sub_ab*cos(PI/8);
    mul_out2=sub_cd*cos(3*PI/8);
    adder_out1=mul_out1+mul_out2;
    adder_out2=mul_out1-mul_out2;
    *(out_dat+6)=(adder_out1/2.0)*SQRT2/4;
    *(out_dat+14)=(adder_out2*cos(PI/4)-adder_out1/2.0)*SQRT2/4;
}
void pre_processing(int *in_dat, int *out_dat, UINT size)
{
    int i,j;
    int sum_ab,sum_cd,sub_ab,sub_cd;
    for(i=0;i<size/2;i++)
        for(j=0;j<size/2;j++)
        {
            sum_ab=*(in_dat+i*size+j)+*(in_dat+i*size+size-1-j);
             sub_ab=*(in_dat+i*size+j)-*(in_dat+i*size+size-1-j);
            sum_cd=*(in_dat+(size-1-i)*size+j)+*(in_dat+(size-1-i)*size+size-1-j);
             sub_cd=*(in_dat+(size-1-i)*size+j)-*(in_dat+(size-1-i)*size+size-1-j);
            *(out_dat+i*size+j)=sum_ab+sum_cd;
            *(out_dat+(size-1-i)*size+size-1-j)=sub_ab-sub_cd;
              *(out_dat+i*size+size-1-j)=sub_ab+sub_cd;
            *(out_dat+(size-1-i)*size+j)=sum_ab-sum_cd;
        }
}
```

附录 2：8×8 点 PSDA DCT C 语言代码

附 2.1：头文件定义

```
#include "stdio.h"
#include "math.h"
#define PI 3.14159
typedef unsigned char uchar;
void pre_processing(short *in_dat, short *out_dat, uchar size);
void direct_dct8x8(short test_data[8][8],short gold_ref[8][8]);
void direct_dct8x8_float(short test_data[8][8],double gold_ref_float[8][8]);
void par_dct8x8(short data[8][8],short dct_out[8][8]);
void par_dct4x4(short data[4][4],short dct_out[4][4]);
void partial_sum_cal(short opt_dat[4][4], short par_sum0[4][4],
                     short par_sum1[4][4],short par_sum2[4][4],
                     short par_sum3[4][4],uchar dat_sel);
void partial_sum_cal_case_odd_even(short opt_data[4][4],short par_sum[4][4]);
void partial_sum_cal_case_odd_odd(short opt_data[4][4],short par_sum[4][4]);
void display(short* dat,uchar row,uchar column);
void display_float(double* par_sum, uchar row, uchar column);
void dct8_odd_part(short* p_par_sum,short* dct_out);
void dct8_even_part(short* p_par_sum,short* dct_out);
```

附 2.2：8x8 点 PSDA DCT 实现代码

```
#include "dct_def.h"
void direct_dct8x8(short test_data[8][8],short gold_ref[8][8])
{
    double temp;
    uchar i,j,k,l;
    for(k=0;k<8;k++)
    {
        for(l=0;l<8;l++)
        {
            temp=0;
            for(i=0;i<8;i++)
            {
                for(j=0;j<8;j++)
                {
                    temp=temp+cos((2*i+1)*k*PI/16)*cos((2*j+1)*l*PI/16)*
                        (double)(test_data[i][j]);
                }
            }
            gold_ref[k][l]=(short)temp;
        }
    }
}

void direct_dct8x8_float(short test_data[8][8],double gold_ref[8][8])
{
    double temp;
    uchar i,j,k,l;
    for(k=0;k<8;k++)
    {
        for(l=0;l<8;l++)
        {
            temp=0;
            for(i=0;i<8;i++)
            {
                for(j=0;j<8;j++)
                {
                    temp=temp+cos((2*i+1)*k*PI/16)*cos((2*j+1)*l*PI/16)*
                        (double)(test_data[i][j]);
                }
            }
            gold_ref[k][l]=temp;
        }
    }
}

void par_dct8x8(short data[8][8],short dct_out[8][8])
{
    short par_sum0[4][4],par_sum1[4][4],par_sum2[4][4],par_sum3[4][4];
    short dct_out_part0[4][4],dct_out_part1[4][4],
```

```
        dct_out_part2[4][4],dct_out_part3[4][4];
short temp_data1[8][8],temp_data2[8][8],opt_data[4][4];
short sum_temp,sub_temp;
uchar i,j;
uchar k;
uchar dat_sel;
//128 add operations
pre_processing(&data[0][0],&temp_data2[0][0],8);
printf("\n");
display(&temp_data2[0][0],8,8);
for(k=0;k<4;k++)
{
    dat_sel=k;
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            switch(dat_sel)
            {
                case 0: opt_data[i][j]=temp_data2[i][j];
                        break;
                //rotate data
                case 1: opt_data[j][i]=temp_data2[i][j+4];
                        break;
                case 2: opt_data[i][j]=temp_data2[i+4][j];
                        break;
                case 3: opt_data[i][j]=temp_data2[i+4][j+4];
                    break;
                default:opt_data[i][j]=0;
                    break;
            }
        }
    }
    if(k==0)
    {
        par_dct4x4(opt_data,dct_out_part0);
    }
    else
    {
        printf("\n");
        display(&opt_data[0][0],4,4);
    }
    partial_sum_cal(opt_data,par_sum0,par_sum1,par_sum2,par_sum3,dat_sel);
}

for(i=0;i<4;i++)
{
    dct8_odd_part(&par_sum1[i][0],&dct_out_part1[i][0]);
    dct8_odd_part(&par_sum2[i][0],&dct_out_part2[i][0]);
    dct8_even_part(&par_sum3[i][0],&dct_out_part3[i][0]);
}
printf("\n");
display(&dct_out_part2[0][0],4,4);
```

```
    for(i=0;i<4;i++)
        for(j=0;j<4;j++)
        {
            dct_out[2*i][2*j]=dct_out_part0[i][j];
            dct_out[2*i][2*j+1]=dct_out_part1[i][j];
            dct_out[2*i+1][2*j]=dct_out_part2[j][i];
            dct_out[2*i+1][2*j+1]=dct_out_part3[i][j];
        }
}

void partial_sum_cal(short opt_data[4][4],short par_sum0[4][4],
                     short par_sum1[4][4],short par_sum2[4][4],
                     short par_sum3[4][4],
                     uchar dat_sel)
{
    switch(dat_sel)
    {
        case 0: break;
        case 1: partial_sum_cal_case_odd_even(opt_data,par_sum1);break;
        case 2: partial_sum_cal_case_odd_even(opt_data,par_sum2);break;
        case 3: partial_sum_cal_case_odd_odd(opt_data,par_sum3);break;
        default:break;
    }
}

void partial_sum_cal_case_odd_even(short opt_data[4][4],short par_sum[4][4])
{
    //44 add operations
    short temp_data1[4][2];
    short temp_data2[4][2];
    short temp_data3[4];
    short temp_data4[4];
    short temp_data5[4];
    short temp_data6[4];
    uchar i,j;
    //16 add operations
    for(i=0;i<4;i++)
        for(j=0;j<2;j++)
        {
            temp_data1[i][j]=opt_data[i][j]+opt_data[i][3-j];
            temp_data2[i][j]=opt_data[i][j]-opt_data[i][3-j];
        }

    printf("\n");
    display(&temp_data1[0][0],4,2);

    printf("\n");
    display(&temp_data2[0][0],4,2);
    //8 add operations
    for(i=0;i<4;i++)
        for(j=0;j<1;j++)
        {
            temp_data3[i]=temp_data1[i][j]+temp_data1[i][1-j];
```

```
            temp_data4[i]=temp_data1[i][j]-temp_data1[i][1-j];
        }
    //4 add operations
    //F(1,0) rotate factor pi/16
    par_sum[0][0]=temp_data3[0];
    //F(1,0) rotate factor 3*pi/16
    par_sum[0][1]=temp_data3[1];
    //F(1,0) rotate factor 5*pi/16
    par_sum[0][2]=temp_data3[2];
    //F(1,0) rotate factor 7*pi/16
    par_sum[0][3]=temp_data3[3];
    //F(1,4) rotate factor pi/16
    par_sum[2][0]=(temp_data4[1]+temp_data4[2])/2;
    //F(1,4) rotate factor 3*pi/16
    par_sum[2][1]=(temp_data4[0]+temp_data4[3])/2;
    //F(1,4) rotate factor 5*pi/16
    par_sum[2][2]=(temp_data4[0]-temp_data4[3])/2;
    //F(1,4) rotate factor 7*pi/16
    par_sum[2][3]=(temp_data4[1]-temp_data4[2])/2;

    //8 add operations
    for(i=0;i<4;i++)
        for(j=0;j<1;j++)
        {
            temp_data5[i]=temp_data2[i][j]+temp_data2[3-i][1-j];
            temp_data6[i]=temp_data2[i][j]-temp_data2[3-i][1-j];
        }
    //8 add operations
    //F(1,2) rotate factor pi/16
    par_sum[1][0]=(temp_data5[0]+temp_data5[1])/2;
    //F(1,2) rotate factor 3*pi/16
    par_sum[1][1]=(temp_data6[0]+temp_data5[2])/2;
    //F(1,2) rotate factor 5*pi/16
    par_sum[1][2]=(temp_data6[1]+temp_data5[3])/2;
    //F(1,2) rotate factor 7*pi/16
    par_sum[1][3]=(temp_data5[2]-temp_data5[3])/2;
    //F(1,6) rotate factor pi/16
    par_sum[3][0]=(temp_data6[2]+temp_data6[3])/2;
    //F(1,6) rotate factor 3*pi/16
    par_sum[3][1]=(temp_data6[1]-temp_data5[3])/2;
    //F(1,6) rotate factor 5*pi/16
    par_sum[3][2]=(temp_data6[0]-temp_data5[2])/2;
    //F(1,6) rotate factor 7*pi/16
    par_sum[3][3]=(temp_data5[0]-temp_data5[1])/2;
}

void partial_sum_cal_case_odd_odd(short opt_data[4][4],short par_sum[4][4])
{
    short temp_data1[4][2];
    short temp_data2[4][2];
    uchar i,j;
    //16 add operations
    for(i=0;i<4;i++)
```

```
        for(j=0;j<2;j++)
        {
            temp_data1[i][j]=opt_data[i][j]+opt_data[3-i][3-j];
            temp_data2[i][j]=opt_data[i][j]-opt_data[3-i][3-j];
        }
    printf("\n");
    display(&temp_data1[0][0],4,2);
    printf("\n");
    display(&temp_data2[0][0],4,2);
    //10 add operations
    //F(1,1) rotate factor 0*pi/16
    par_sum[0][0]=(temp_data1[0][0]+temp_data1[1][1]);
    //F(1,1) rotate factor 2*pi/16
    par_sum[0][1]=(temp_data2[0][0]+temp_data1[1][0]+temp_data1[0][1]+temp_data1[2][1]);
    //F(1,1) rotate factor 4*pi/16
    par_sum[0][2]=(temp_data2[1][0]+temp_data1[2][0]+temp_data2[0][1]+temp_data1[3][1]);
    //F(1,1) rotate factor 6*pi/16
    par_sum[0][3]=(temp_data2[2][0]+temp_data1[3][0]+temp_data2[1][1]-temp_data2[3][1]);

    //F(1,3) rotate factor 0*pi/16
    par_sum[1][0]=(temp_data2[1][0]-temp_data1[3][1]);
    //F(1,3) rotate factor 2*pi/16
    par_sum[1][1]=(temp_data2[0][0]+temp_data2[2][0]-temp_data1[2][1]+temp_data2[3][1]);
    //F(1,3) rotate factor 4*pi/16
    par_sum[1][2]=(temp_data1[0][0]+temp_data2[3][0]-temp_data1[1][1]+temp_data2[2][1]);
    //F(1,3) rotate factor 6*pi/16
    par_sum[1][3]=(temp_data1[1][0]-temp_data1[3][0]-temp_data1[0][1]+temp_data2[1][1]);

    //F(1,5) rotate factor 0*pi/16
    par_sum[2][0]=(temp_data1[2][0]-temp_data2[0][1]);
    //F(1,5) rotate factor 2*pi/16
    par_sum[2][1]=(temp_data1[1][0]+temp_data1[3][0]-temp_data1[0][1]-temp_data2[1][1]);
    //F(1,5) rotate factor 4*pi/16
    par_sum[2][2]=(temp_data1[0][0]-temp_data2[3][0]-temp_data1[1][1]-temp_data2[2][1]);
    //F(1,5) rotate factor 6*pi/16
    par_sum[2][3]=(temp_data2[0][0]-temp_data2[2][0]-temp_data1[2][1]-temp_data2[3][1]);

    //F(1,7) rotate factor 0*pi/16
    par_sum[3][0]=(temp_data2[3][0]-temp_data2[2][1]);
    //F(1,7) rotate factor 2*pi/16
    par_sum[3][1]=(temp_data2[2][0]-temp_data1[3][0]-temp_data2[1][1]-temp_data2[3][1]);
    //F(1,7) rotate factor 4*pi/16
    par_sum[3][2]=(temp_data2[1][0]-temp_data1[2][0]-temp_data2[0][1]+temp_data1[3][1]);
    //F(1,7) rotate factor 6*pi/16
    par_sum[3][3]=(temp_data2[0][0]-temp_data1[1][0]-temp_data1[0][1]+temp_data1[2][1]);
}

void dct8_even_part(short* p_par_sum,short* dct_out)
{
    double temp_dat1[4],temp_dat2[4],temp_dat3[4];
    double dct_float_out[4];
    //9 add operations, 4 multiply operations
    temp_dat1[0]=(double)(*p_par_sum);
```

```
    temp_dat1[1]=(double)(*(p_par_sum+1))*2*cos(PI/8);
    temp_dat1[2]=(double)(*(p_par_sum+2))*cos(PI/4);
    temp_dat1[3]=(double)(*(p_par_sum+3))*2*cos(PI*3/8);

    temp_dat2[0]=temp_dat1[0]+temp_dat1[2];
    temp_dat2[2]=temp_dat1[0]-temp_dat1[2];
    temp_dat2[1]=temp_dat1[1]+temp_dat1[3];
    temp_dat2[3]=(temp_dat1[1]-temp_dat1[3])*cos(PI/4);

    temp_dat3[0]=temp_dat2[0];
    temp_dat3[2]=temp_dat2[2];
    temp_dat3[1]=temp_dat2[1]/2;
    temp_dat3[3]=temp_dat2[3]-temp_dat2[1]/2;

    dct_float_out[0]=(temp_dat3[0]+temp_dat3[1])/2;
    dct_float_out[1]=(temp_dat3[2]+temp_dat3[3])/2;
    dct_float_out[2]=(temp_dat3[2]-temp_dat3[3])/2;
    dct_float_out[3]=(temp_dat3[0]-temp_dat3[1])/2;

    *dct_out=short(dct_float_out[0]);
    *(dct_out+1)=short(dct_float_out[1]);
    *(dct_out+2)=short(dct_float_out[2]);
    *(dct_out+3)=short(dct_float_out[3]);
}

void dct8_odd_part(short* p_par_sum,short* dct_out)
{
    //12 add operations and 8 multiply operations
    uchar i;
    double temp_dat1[4],temp_dat2[4],temp_dat3[4],temp_dat4[4],temp_dat5[4];
    double temp_dat_float[4];
    double dct_float_out[4];
    //4 multiply operations
    for(i=0;i<4;i++)
    {
        temp_dat_float[i]=(double)(*(p_par_sum+i));
        temp_dat1[i]=temp_dat_float[i]*2*cos((2*i+1)*PI/16);
    }
    //4 add operations
    temp_dat2[0]=temp_dat1[0]+temp_dat1[3];
    temp_dat2[1]=temp_dat1[1]+temp_dat1[2];
    temp_dat2[2]=temp_dat1[1]-temp_dat1[2];
    temp_dat2[3]=temp_dat1[0]-temp_dat1[3];
    //2 multiply operations
    temp_dat3[0]=temp_dat2[0];
    temp_dat3[1]=temp_dat2[1];
    temp_dat3[2]=temp_dat2[2]*2*cos(3*PI/8);
    temp_dat3[3]=temp_dat2[3]*2*cos(PI/8);
    //4 add operations
    temp_dat4[0]=temp_dat3[0]+temp_dat3[1];
    temp_dat4[1]=temp_dat3[0]-temp_dat3[1];
    temp_dat4[2]=temp_dat3[3]+temp_dat3[2];
    temp_dat4[3]=temp_dat3[3]-temp_dat3[2];
```

```
    //2 mul,2 shift,1 add
    temp_dat5[0]=temp_dat4[0]/2;
    temp_dat5[1]=temp_dat4[1]*cos(PI/4);
    temp_dat5[2]=temp_dat4[2]/2;
    temp_dat5[3]=temp_dat4[3]*cos(PI/4)-temp_dat5[2];
    // 3 add
    dct_float_out[0]=temp_dat5[0];
    dct_float_out[1]=temp_dat5[2]-dct_float_out[0];
    dct_float_out[2]=temp_dat5[1]-dct_float_out[1];
    dct_float_out[3]=temp_dat5[3]-dct_float_out[2];
    *dct_out=(short)(dct_float_out[0]);
    *(dct_out+1)=(short)(dct_float_out[1]);
    *(dct_out+2)=(short)(dct_float_out[2]);
    *(dct_out+3)=(short)(dct_float_out[3]);
}


void display(short* par_sum, uchar row, uchar column)
{
    uchar i,j;
    for(i=0;i<row;i++)
    {
        for(j=0;j<column;j++)
        {
            printf("%8d",*(par_sum+i*column+j));
            printf("%c",0x20);
        }
        printf("\n");
    }
}

void display_float(double* par_sum, uchar row, uchar column)
{
    uchar i,j;
    printf("\n");
    printf("\n");
    for(i=0;i<row;i++)
    {
        for(j=0;j<column;j++)
        {
            printf("%8.4f",*(par_sum+i*column+j));
            printf("%c",0x20);
        }
        printf("\n");
    }
}

void pre_processing(short *in_dat, short *out_dat, uchar size)
{
    int i,j;
    int sum_ab,sum_cd,sub_ab,sub_cd;
    for(i=0;i<size/2;i++)
        for(j=0;j<size/2;j++)
```

```
        {
            sum_ab=*(in_dat+i*size+j)+*(in_dat+i*size+size-1-j);
            sub_ab=*(in_dat+i*size+j)-*(in_dat+i*size+size-1-j);
            sum_cd=*(in_dat+(size-1-i)*size+j)+*(in_dat+(size-1-i)*size+size-1-j);
            sub_cd=*(in_dat+(size-1-i)*size+j)-*(in_dat+(size-1-i)*size+size-1-j);
            *(out_dat+i*size+j)=sum_ab+sum_cd;
            *(out_dat+(i+size/2)*size+j+size/2)=sub_ab-sub_cd;
            *(out_dat+i*size+j+size/2)=sub_ab+sub_cd;
            *(out_dat+(i+size/2)*size+j)=sum_ab-sum_cd;
        }
}

void par_dct4x4(short data[4][4],short dct_out[4][4])
{
    short temp_data1[4][4],par_sum0[2][2],par_sum1[2][2],par_sum2[2][2],par_sum3[2][2],
          dct_out_part0[2][2],dct_out_part1[2][2],dct_out_part2[2][2],dct_out_part3[2][2];
    double dct_out_float[4];
    uchar i,j;
    double temp_dat2[4],temp_dat3[4],temp_dat4[4];
    display(&data[0][0],4,4);
    //32 add operations
    pre_processing(&data[0][0],&temp_data1[0][0],4);
    for(i=0;i<2;i++)
        for(j=0;j<2;j++)
        {
            par_sum0[i][j]=temp_data1[i][j];
            par_sum1[j][i]=temp_data1[i][j+2];
            par_sum2[i][j]=temp_data1[i+2][j];
            par_sum3[i][j]=temp_data1[i+2][j+2];
        }

    display(&par_sum0[0][0],2,2);
    display(&par_sum1[0][0],2,2);
    display(&par_sum2[0][0],2,2);
    display(&par_sum3[0][0],2,2);
    //8 add operations
    pre_processing(&par_sum0[0][0],&dct_out_part0[0][0],2);

    //calculate F(0,1),F(2,1),F(0,3),F(2,3)
    //10 add operations
    temp_dat2[0]=par_sum1[0][0]+par_sum1[0][1];
    temp_dat2[1]=par_sum1[1][0]+par_sum1[1][1];
    temp_dat2[2]=par_sum1[1][0]-par_sum1[1][1];
    temp_dat2[3]=par_sum1[0][0]-par_sum1[0][1];

    temp_dat3[0]=temp_dat2[0]*2*cos(PI/8);
    temp_dat3[1]=temp_dat2[1]*2*cos(3*PI/8);
    temp_dat3[2]=temp_dat2[2]*2*cos(PI/4)*cos(3*PI/8);
    temp_dat3[3]=temp_dat2[3]*2*cos(PI/4)*cos(PI/8);

    temp_dat4[0]=temp_dat3[0]+temp_dat3[1];
    temp_dat4[1]=temp_dat3[0]-temp_dat3[1];
    temp_dat4[2]=temp_dat3[3]+temp_dat3[2];
```

```
temp_dat4[3]=temp_dat3[3]-temp_dat3[2];
dct_out_float[0]=temp_dat4[0]/2;
dct_out_float[1]=temp_dat4[1]*cos(PI/4)-temp_dat4[0]/2;
dct_out_float[2]=temp_dat4[2]/2;
dct_out_float[3]=temp_dat4[3]*cos(PI/4)-temp_dat4[2]/2;

dct_out_part1[0][0]=(short)dct_out_float[0];
dct_out_part1[0][1]=(short)dct_out_float[1];
dct_out_part1[1][0]=(short)dct_out_float[2];
dct_out_part1[1][1]=(short)dct_out_float[3];

//10 add operations
temp_dat2[0]=par_sum2[0][0]+par_sum2[0][1];
temp_dat2[1]=par_sum2[1][0]+par_sum2[1][1];
temp_dat2[2]=par_sum2[1][0]-par_sum2[1][1];
temp_dat2[3]=par_sum2[0][0]-par_sum2[0][1];

temp_dat3[0]=temp_dat2[0]*2*cos(PI/8);
temp_dat3[1]=temp_dat2[1]*2*cos(3*PI/8);
temp_dat3[2]=temp_dat2[2]*2*cos(PI/4)*cos(3*PI/8);
temp_dat3[3]=temp_dat2[3]*2*cos(PI/4)*cos(PI/8);

temp_dat4[0]=temp_dat3[0]+temp_dat3[1];
temp_dat4[1]=temp_dat3[0]-temp_dat3[1];
temp_dat4[2]=temp_dat3[3]+temp_dat3[2];
temp_dat4[3]=temp_dat3[3]-temp_dat3[2];
dct_out_float[0]=temp_dat4[0]/2;
dct_out_float[1]=temp_dat4[1]*cos(PI/4)-temp_dat4[0]/2;
dct_out_float[2]=temp_dat4[2]/2;
dct_out_float[3]=temp_dat4[3]*cos(PI/4)-temp_dat4[2]/2;

dct_out_part2[0][0]=(short)dct_out_float[0];
dct_out_part2[0][1]=(short)dct_out_float[1];
dct_out_part2[1][0]=(short)dct_out_float[2];
dct_out_part2[1][1]=(short)dct_out_float[3];

//calculate F(1,1),F(3,3),F(1,3),F(3,1)
//10 add operations
temp_dat2[0]=par_sum3[0][0]+par_sum3[1][1];
temp_dat2[1]=par_sum3[1][0]+par_sum3[0][1];
temp_dat2[2]=par_sum3[1][0]-par_sum3[0][1];
temp_dat2[3]=par_sum3[0][0]-par_sum3[1][1];

dct_out_float[0]=temp_dat2[0]+(temp_dat2[1]+temp_dat2[3])*cos(PI/4);
dct_out_float[1]=temp_dat2[2]-(temp_dat2[1]-temp_dat2[3])*cos(PI/4);
dct_out_float[2]=-temp_dat2[2]-(temp_dat2[1]-temp_dat2[3])*cos(PI/4);
dct_out_float[3]=temp_dat2[0]-(temp_dat2[1]+temp_dat2[3])*cos(PI/4);

dct_out_part3[0][0]=(short)dct_out_float[0];
dct_out_part3[0][1]=(short)dct_out_float[1];
dct_out_part3[1][0]=(short)dct_out_float[2];
dct_out_part3[1][1]=(short)dct_out_float[3];
```

```
//    dct_out_part1[0][0]=short(dct_out_float[0]);
//    dct_out_part1[0][1]=short(dct_out_float[1]);

      for(i=0;i<2;i++)
          for(j=0;j<2;j++)
          {
              dct_out[2*i][2*j]=dct_out_part0[i][j];
              dct_out[2*i][2*j+1]=dct_out_part1[i][j];
              dct_out[2*i+1][2*j]=dct_out_part2[j][i];
              dct_out[2*i+1][2*j+1]=dct_out_part3[i][j];
          }

}
```

附 2.3：8x8 点 PSDA DCT 测试代码

```
#include "dct_def.h"
void main()
{
    short test_data[8][8];
    short gold_dct_out[8][8];
    double gold_dct_out_float[8][8];
    short par_dct_out[8][8];
    uchar i,j;
    double t[4];
    for(i=0;i<8;i++)
        for(j=0;j<8;j++)
        {
            par_dct_out[i][j]=0;
            if(i<4)
                if(j<4)
                    test_data[i][j]=8*i+j;
                else
                    test_data[i][j]=8*i+7-j;
            else
                test_data[i][j]=8*(7-i)+j;
        }
    display(&test_data[0][0],8,8);
    printf("direct dct out");
    direct_dct8x8(test_data,gold_dct_out);
    printf("\n");
    display(&gold_dct_out[0][0],8,8);
    direct_dct8x8_float(test_data,gold_dct_out_float);
    printf("\n");
    display_float(&gold_dct_out_float[0][0],8,8);

    par_dct8x8(test_data,par_dct_out);
    printf("\n");
    display(&par_dct_out[0][0],8,8);
    i=0;
```

附录 3：PSDA 算法 IP 核交付件(环境设置文件和综合脚本)

Design Compile 的.synopsys_dc.setup 文件

```
search_path = {D:/Synopsys/libraries/syn/smic_18} + search_path
link_library = {"*","smic18a_tt_33_25.db"};
target_library = {"smic18a_tt_33_25.db"};
symbol_library = {"SNsmic.sdb"};
company = "UESTC 111 lab" ;
designer = "Tian Mao";
view_background = "black";
define_design_lib work -path work;
```

fdct_partial_sum_implementation 的综合脚本

```
/*********************************************/
/*Design entry*/
analyze -format vhdl dct_top_connection.vhd
elaborate dct_top_connection
analyze -format vhdl add_adjust.vhd
elaborate add_adjust
analyze -format vhdl butterfly_trans.vhd
elaborate butterfly_trans
analyze -format vhdl control_unit.vhd
elaborate control_unit
analyze -format vhdl dat64bit_mux.vhd
elaborate dat64bit_mux
analyze -format vhdl mul_add_module.vhd
elaborate mul_add_module
analyze -format vhdl par_cal_unit.vhd
elaborate vhdl par_cal_unit
analyze -format vhdl mul_add_module.vhd
elaborate mul_add_module
analyze -format vhdl par_cal_unit.vhd
```

```
elaborate vhdl par_cal_unit
analyze -format vhdl coef_rom.vhd
elaborate vhdl coef_rom
analyze -format vhdl dat_sel_rom.vhd
elaborate vhdl dat_sel_rom
current_design dct_top_connection
uniquify
check_design
/*********************************************/
/*Setup operating conditions, wire_load, clock, reset*/
set_wire_load "100K" -library "SN35A_TT_33_25"
set_wire_load_mode enclosed
create_clock -period 20 -waveform {0 20} clk
set_clock_skew -delay 2.0 -minus_uncertainty 1.0 -plus_uncertainty 0.5 clk
set_dont_touch_network {clk rst}
/*********************************************/
/*Input drives*/
set_driving_cell -cell RS_BUF_B -pin O all_inputs()
set_drive 0 {clk,rst}
/*********************************************/
/*Output loads*/
set_load 0.5 all_outputs()
/*********************************************/
```

附录 4：PSDA 算法 IP 核面积测试结果

Information: Updating design information... (UID-85)

Warning: The trip points for the library named USERLIB differ from those in the library named slow. (TIM-164)

Warning: The trip points for the library named USERLIB differ from those in the library named slow. (TIM-164)

Warning: The trip points for the library named USERLIB differ from those in the library named slow. (TIM-164)

****

Report : area

Design : dct_top_connection

Version: W-2004.12

Date : Tue Mar 27 17:41:43 2007

****

Library(s) Used:

slow (File: /home/careful/tm/lib/slow.db)

USERLIB (File: /home/careful/tm/lib/parsum_ram_slow.db)

USERLIB (File: /home/careful/tm/lib/dat_sel_rom_slow.db)

USERLIB (File: /home/careful/tm/lib/coef_rom_slow.db)

Number of ports:	146
Number of nets:	851
Number of cells:	44
Number of references:	12

Combinational area: 117222.828125

Noncombinational area: 880140.187500

Net Interconnect area: 1742868.125000

Total cell area: 997363.500000

Total area: 2740231.250000

Information: This design contains black box (unknown) components. (RPT-8)

1

附录 5：PSDA 算法 IP 核时序测试结果

Warning: Design 'top_connection' has '3' unresolved references. For more detailed information, use the "link" command. (UID-341)

****

Report : timing

-path full

-delay max

-max_paths 1

Design : dct_top_connection

Version: W-2004.12

Date : Tue Mar 27 17:41:44 2007

****

Operating Conditions: slow Library: slow

Wire Load Model Mode: top

Startpoint: uut12/mux_out_reg_36_

(rising edge-triggered flip-flop clocked by g_ck)

Endpoint: uut13/mul1_reg_25_

(rising edge-triggered flip-flop clocked by g_ck)

Path Group: g_ck

Path Type: max

Des/Clust/Port	Wire Load Model	Library
dct_top_connection	smic18_wl30	slow

Point	Incr	Path
clock g_ck (rise edge)	0.00	0.00
clock network delay (ideal)	0.30	0.30
uut12/mux_out_reg_36_/CK (DFFHQX1)	0.00	0.30 r
uut12/mux_out_reg_36_/Q (DFFHQX1)	0.68	0.98 r
uut12/o1[4] (dat64bit_mux)	0.00	0.98 r

uut13/a1[4] (mul_add_module)	0.00	0.98 r
uut13/U254/Y (INVX8)	0.46	1.44 f
uut13/U1147/Y (NOR2X1)	0.53	1.97 r
uut13/U1149/ICO (CMPR42X1)	0.60	2.57 r
uut13/U1160/ICO (CMPR42X1)	0.52	3.08 r
uut13/U1120/S (CMPR42X1)	0.72	3.81 r
uut13/U1204/CO (CMPR42X1)	0.96	4.76 r
uut13/U1224/S (CMPR42X1)	1.26	5.03 r
uut13/U359/Y (OR2X2)	0.56	6.59 r
uut13/U346/Y (CLKINVX3)	0.25	5.84 f
uut13/U1585/Y (NOR2X4)	0.27	6.11 r
uut13/U1324/Y (NAND2X2)	0.37	6.48 f
uut13/U1325/Y (CLKINVX3)	0.59	6.07 r
uut13/U317/Y (NAND2X1)	0.35	6.42 f
uut13/U307/Y (OAI21XL)	0.68	6.10 r
uut13/U1292/Y (XNOR2X1)	0.54	6.63 f
uut13/mul1_reg_25_/D (DFFHQX2)	0.00	6.63 f
data arrival time		6.63
clock g_ck (rise edge)	7.00	7.00
clock network delay (ideal)	0.30	7.30
clock uncertainty	-0.30	7.00
uut13/mul1_reg_25_/CK (DFFHQX2)	0.00	7.00 r
library setup time	-0.36	6.64
data required time		6.64
---		
data required time		6.64
data arrival time		-6.63
---		
slack (MET)		0.00

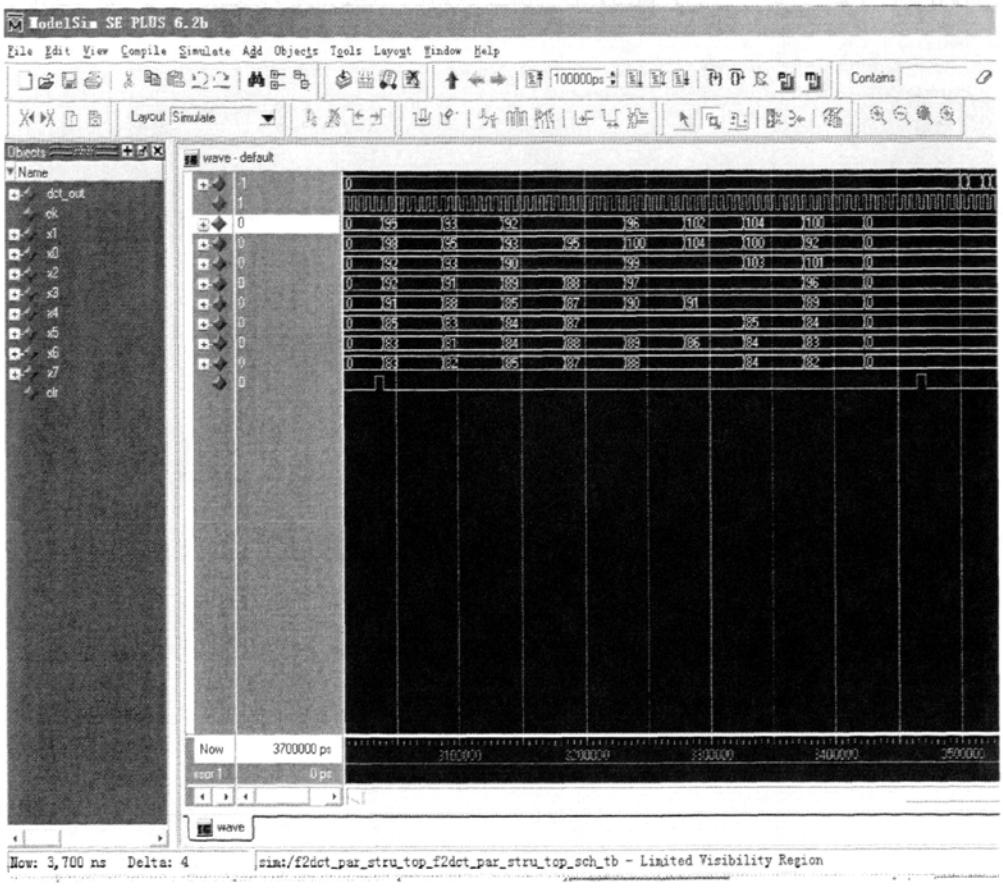
附录 6：PSDA 算法 IP 核功耗测试结果

```
****************************************
Report : power
        -cell
        -analysis_effort low
        -sort_mode cell_internal_power
Design : dct_top_connection
Version: W-2004.12
Date     : Tue Mar 27 17:41:51 2007
****************************************
Library(s) Used:
        slow (File: /home/careful/tm/lib/slow.db)
        USERLIB (File: /home/careful/tm/lib/parsum_ram_slow.db)
        USERLIB (File: /home/careful/tm/lib/dat_sel_rom_slow.db)
        USERLIB (File: /home/careful/tm/lib/coef_rom_slow.db)
Operating Conditions: slow     Library: slow
Wire Load Model Mode: top
Design              Wire Load Model                 Library
------------------------------------------------
dct_top_connection              smic18_wl30           slow
Global Operating Voltage = 1.62
Power-specific unit information :
        Voltage Units = 1V
        Capacitance Units = 1.000000pf
        Time Units = 1ns
        Dynamic Power Units = 1mW      (derived from V,C,T units)
        Leakage Power Units = 1pW
  Attributes ---------h  -  Hierarchical cell
                              Cell         Driven Net   Tot Dynamic         Cell
```

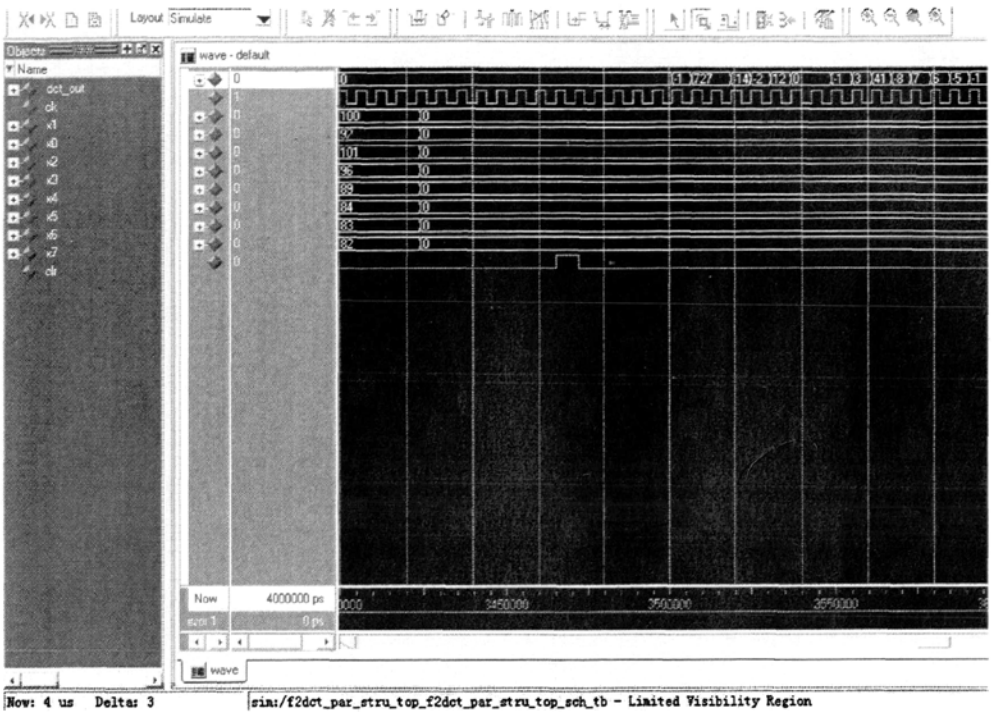
Cell	Internal Power	Switching Power	Power (% Cell/Tot)	Leakage Power	Attrs
uut5	37.2282	0.1438	37.372 (100%)	14000000.0000	
uut6	37.2282	0.1438	37.372 (100%)	14000000.0000	
uut13	11.8907	N/A	N/A (N/A)	5189301.0000	h
uut1	6.4254	N/A	N/A (N/A)	2780534.0000	h
uut12	0.6534	N/A	N/A (N/A)	301113.7188	h
uut2	0.3106	N/A	N/A (N/A)	167174.8281	h
uut7	0.2085	0.0630	0.271 (77%)	6000000.0000	
uut8	0.2085	0.0630	0.271 (77%)	6000000.0000	
uut9	0.1181	0.1260	0.244 (48%)	8000000.5000	
U39	1.644e-03	4.633e-03	6.28e-03 (26%)	540.9948	
U41	1.644e-03	4.633e-03	6.28e-03 (26%)	540.9948	
U43	1.644e-03	4.633e-03	6.28e-03 (26%)	540.9948	
U45	1.644e-03	4.633e-03	6.28e-03 (26%)	540.9948	
U48	1.644e-03	4.647e-03	6.29e-03 (26%)	540.9948	
U52	1.644e-03	4.647e-03	6.29e-03 (26%)	540.9948	
U56	1.644e-03	4.647e-03	6.29e-03 (26%)	540.9948	
U60	1.644e-03	4.647e-03	6.29e-03 (26%)	540.9948	
U47	1.643e-03	4.843e-03	6.49e-03 (25%)	540.9948	
U51	1.643e-03	4.843e-03	6.49e-03 (25%)	540.9948	
U55	1.643e-03	4.843e-03	6.49e-03 (25%)	540.9948	

U59	1.643e-03	4.843e-03	6.49e-03 (25%)	540.9948	
U49	1.643e-03	4.894e-03	6.54e-03 (25%)	540.9948	
U53	1.643e-03	4.894e-03	6.54e-03 (25%)	540.9948	
U57	1.643e-03	4.894e-03	6.54e-03 (25%)	540.9948	
U61	1.643e-03	4.894e-03	6.54e-03 (25%)	540.9948	
U46	1.641e-03	5.125e-03	6.77e-03 (24%)	540.9948	
U50	1.641e-03	5.125e-03	6.77e-03 (24%)	540.9948	
U54	1.641e-03	5.125e-03	6.77e-03 (24%)	540.9948	
U58	1.641e-03	5.125e-03	6.77e-03 (24%)	540.9948	
U38	1.641e-03	5.149e-03	6.79e-03 (24%)	540.9948	
U40	1.641e-03	5.149e-03	6.79e-03 (24%)	540.9948	
U42	1.641e-03	5.149e-03	6.79e-03 (24%)	540.9948	
U44	1.641e-03	5.149e-03	6.79e-03 (24%)	540.9948	
U62	5.686e-04	4.647e-03	5.22e-03 (11%)	490.0853	
U64	5.686e-04	4.647e-03	5.22e-03 (11%)	490.0853	
U66	5.686e-04	4.647e-03	5.22e-03 (11%)	490.0853	
U68	5.686e-04	4.647e-03	5.22e-03 (11%)	490.0853	
U63	5.662e-04	4.894e-03	5.46e-03 (10%)	490.0853	
U65	5.662e-04	4.894e-03	5.46e-03 (10%)	490.0853	
U67	5.662e-04	4.894e-03	5.46e-03 (10%)	490.0853	
U69	5.662e-04	4.894e-03	5.46e-03 (10%)	490.0853	
uut3	0.0000	N/A	N/A (N/A)	0.0000	h
uut4	0.0000	N/A	N/A (N/A)	0.0000	h
uut11	0.0000	0.2760	0.276 (0%)	0.0000	
Totals (44 cells)	94.316mW	N/A	N/A (N/A)	56.455uW	

附录 7：DCT 变换仿真波形



附图 1：DCT 变换输入数据



附图 2：DCT 变换输出数据