# 摘　要

离散余弦变换(DCT)及其反变换(IDCT)在图像编解码方面应用十分广泛，至今已被JPEG、MPEG-1、MPEG-2、MPEG-4和H.26x等国际标准所采用。由于其计算量较大，软件实现往往难以满足实时处理的要求，因而在很多实际应用中需要采用硬件设计的DCT/IDCT处理电路来满足我们对处理速度的要求。本文所研究的内容就是针对图像处理应用的8×8二维DCT/IDCT处理核的硬件实现。

本文首先介绍了DCT和IDCT在图像处理中的作用和原理，详细说明了DCT变换实现图像压缩的过程，并与其它变换比较说明了用DCT变换实现图像压缩的优势。接着，分析研究了DCT的各种快速算法，总结了前人对DCT快速算法及其实现所做的研究。本文给出了两种性能、资源上有一定差异的二维DCT/IDCT的FPGA设计方案。两种方案均利用DCT的行列分离特性，采用流水线设计技术，将二维DCT/IDCT实现转化为两个一维DCT/IDCT实现。在一维DCT/IDCT设计中，根据图像处理的特点对Loeffler算法的数据流进行了优化，通过合理安排时钟周期数和简化各周期内的操作，大大缩短了关键路径的执行时间，从而提高了流水线的执行速度。最后，对所设计的DCT/IDCT处理核进行了综合和时序仿真。

结果表明，当使用Altera公司的MERCURY系列FPGA器件时，本文设计的方案一能够在116M时钟频率下正确完成8×8的二维DCT或IDCT的逻辑运算，消耗2827个逻辑单元；方案二能够在74M时钟频率下正常工作，消耗1629个逻辑单元。

该项研究工作得到了国家自然科学基金（60173042）的资助。

**关键字**：图像压缩；离散余弦变换；现场可编程门阵列；寄存器传输级；时序仿真

# Abstract

Discrete Cosine Transform(DCT) and Inverse Discrete Cosine Transform(IDCT) are most widely used image compression techniques and current standards for the compression of still(JPEG) and moving(MPEG-1,2,4 H.26x) images use DCT to remove spatial redundancy in images. It is difficult to make a real-time implementation of it by software method because it takes too many CPU cycles. Therefor we are trend to use hareware implementation to satisfy our requirement. This article is dedicate to the hareware implementation of 2-D DCT/IDCT FPGA core.

This article first introduced the theory and advantages of using Discrete Cosine Transform and Inverse Discrete Cosine Transform in images compression. We portrayed the process of using DCT/IDCT in image compression, compared it with several other transforms such as DST, DFT. Then we made a vivid discussion among several fast DCT algorithms and made a conclusion of such algorithms.

In this article we proposed two different resolutions to fast DCT transform. Both of them enroled the pipelinning technology and use the Row Column decomposition Method (RCM) to decompose the two demensional DCT into one demensional DCTs. When calculating the 1-D DCT/IDCT, we made some improves in Loeffler's fast DCT algorithm according to the characteristics of image compression. After rearrange and simplifying the calculation cycles, we got a shorter critical path and thus speed up the pipelling. We programed the IP core and synthesized it under the MERCURY series FPGA chipset, and at last we proved it by cycle accurate simulation.

The synthesize result shows one of our DCT/IDCT IP core can run as fast as 116 MHz, it takes 2827 Logic Elements(LEs). the other one can run as fast as 74 MHz, but it just takes 1629 Logic Elements.

This work is sponsored by the National Natural Science Foundation of China. Proj. NO. 60173042.

**Key Words:** image compression;Discrete Cosine Transform (DCT);Field Programmable Gate Array (FPGA); Register Transfer Level (RTL); Cycle Accurate Simulation

# 插 图 索 引

# 湖 南 大 学

## 学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：罗天照　　日期：2006年 2月 14日

## 学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本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和汇编本学位论文。

本学位论文属于

1、保密□，在_____年解密后适用本授权书。

2、不保密☑。

（请在以上相应方框内打“√”）

作者签名：罗天照　　日期：2006 年 2月14日

导师签名：　　日期：2006 年 2 月 14日

# 第 1 章　绪 论

## 1.1 研究背景

2005 年 9 月，中央电视台已经在杭州、成都等城市进行了高清电视节目的试播。2006 年元旦，央视《高清影视》频道的节目将正式在全国范围内播放。与此同时，北京 2008 年奥运会将全部用数字高清晰度信号进行电视转播，数字奥运建设工作正在紧张有序的进行之中。

无论央视的高清电视还是 2008 年的数字奥运，都存在着这样一个问题：选择哪个标准或者说按照什么格式对数字电视信号进行存储、传送。我国将于 2008 年全面推广数字高清电视的地面传输，并于 2015 年关闭现有的模拟电视广播。可以说，谁掌握了这个高清标准谁就掌握了我国数字电视的经济命脉。

中国音视频企业在核心技术上一直受制于发达国家跨国公司，虽然中科院计算所早在 2002 年就成立了 AVS 技术标准工作组，专门负责制订数字音视频的压缩、解压缩、处理和表示等共性技术标准。并于 2003 年 11 月 25 日正式公布了数字视频标准最终草案，正式提交信息产业部和国家标准管理委员会进行审批。然而在数字高清电视的标准选择上，央视却放弃了国产标准，而选择了 MPEG-2。并于 2005 年 11 月与松下、日立两家日本厂商签订了合作协议。

目前 MPEG-2 的下一代标准 MPEG-4 已改为收取专利费的形式，即向运营商按每个用户每小时 2 美分收取，以中国 1/3 的家庭使用计算，每年将要缴纳高达 100 亿[1]的专利费。缴纳这些专利费对于央视数字电视产业链的终端厂商无疑将是沉重的打击。

央视选择 MPEG-2 的原因是多方面的，但其中一个重要的原因就是国产 AVS 标准缺乏相关的视频编码/解码芯片、板卡及产品。

在未来十多年的时间内，国内对音视频编码/解码芯片的需求量年均将达 4000 万片以上[2]。因此进行支持国产音频、视频标准的编码/解码芯片的研发将是一件非常有意义的工作。

在所有视频编码/解码芯片中离散余弦变换及其反变换(DCT/IDCT)占到了全部计算量的 20%-30%。二维 DCT/IDCT 处理核是编码/解码芯片中的一个重要模块，其设计的好坏将直接影响整个芯片的性能。

### 1.1.1视频压缩的必要性

数字信号有很多优点，但当模拟信号数字化后其频带大大加宽，一路 6MHz 的普通电

视信号数字化后，其数码率将高达 167Mbps，对储存器容量要求很大，占有的带宽将达 80MHz 左右，这样将使数字信号失去实用价值。数字压缩技术很好地解决了上述困难，压缩后信号所占用的频带大大低于原模拟信号的频带。因此说，数字压缩编码技术是使数字信号走向实用化的关键技术之一，表 1-1 列出了各种应用的码率[3]。

**表 1-1 各种应用的码率**

| 应用种类 | 比特数/象素 | 象素数/行 | 行数/帧 | 帧数/秒 | 亮色比 | 比特/秒(压缩前) | 比特/秒 (压缩后) |
|---|---|---|---|---|---|---|---|
| HDTV | 8 | 1920 | 1080 | 30 | 4:1:1 | 1.18Gbps | 20~25Mbps |
| 普通电视 CCIR601 | 8 | 720 | 480 | 30 | 4:1:1 | 167Mbps | 4~8Mbps |
| 会议电视 CIF | 8 | 352 | 288 | 30 | 4:1:1 | 36.5Mbps | 1.5~2Mbps |
| 桌上电视 QCF | 8 | 176 | 144 | 30 | 4:1:1 | 9.1Mbps | 128kbps |
| 电视电话 | 8 | 128 | 112 | 30 | 4:1:1 | 5.2Mbps | 56kbps |

压缩就是使数字视音频具有较低数据率的一种方式。压缩具有以下优点：

- 对于给定的信源素材，它只需要较少的存储量。
- 在实时工作时，压缩可降低所需带宽。此外，压缩可以使数据在存储介质间的传输速度更快，例如，在磁带和硬盘之间可以实现数据的快速传输。
- 采用压缩记录格式可以减少记录密度，这样就可以降低记录设备对环境因素和设备维护的要求。

### 1.1.2压缩的基本原理

在传递信息内容时，为了减少所需数据量，可以采用两种不同的基本技术。在实用的压缩系统中，常常是这两种技术的组合应用，并且采用了十分复杂的方式。

第一种压缩技术是提高编码效率。对于给定的信息，可以采用许多编码方式。在最简单的视音频数据中，也包含有一定量的冗余度，这就是下面我们要讨论的“熵”的概念。

许多编码技术可以减少或除去这种冗余度。例如游程编码和可变字长编码系统（如霍夫曼编码）。如果应用适当，上述编码技术完全是可逆的，这就是说，解压缩后的数据与编码系统的输入数据是相同的。这种类型的压缩称为无损压缩。存档的计算机程序如 PKZip 就采用了这种无损压缩。很明显，无损压缩虽然十分理想，但它却不能提供视音频应用所需要的数据压缩比。然而，正因为它是无损压缩，所以它可用于系统的任意点，通常我们将无损压缩应用在有损压缩器的数据输出端。

如果除去信息中的冗余度并不能满足所需要的数据压缩量，那就必须要丢弃某些（非冗余的）信息。有损压缩系统就是通过去除不相关的信息或相关性较低的信息来实现所需要的压缩量。不存在对任意数据流均适用的通用有损压缩技术；因为对相关性的评价只能就应用内容本身才能确定，在压缩时应当了解数据代表什么，它又是如何使用的。

在电视情况下，图象和声音的再现是为人的视觉系统和听觉系统而提供的，因此，在设计一个有效的压缩系统时，就必须充分考虑人的主观感受因素。

### 1.1.3数字压缩技术的发展史和现状

1843 年莫尔斯（Morse）的电报码是最原始的变长码数据压缩实例。1938 年里夫斯（Reeves）、1946 年德劳雷恩（E. m. Delorain）以及贝尔公司的卡特勒（C. C. Cutler）分别发明了脉冲编码调制（Pulse Code Modulation， PCM）、增量调制（Delta Modulation，△M）以及差分脉冲编码调制（Differential PCM， DPCM）。

1948 年香农（C. E. Shannon）在其经典论文“通信的数学原理[4]”中首次提到信息率——失真函数概念，1959 年又进一步确立了率失真理论，从而奠定了信源编码的理论基础。1948 年提出电视信号数字化后，就开始了图像压缩编码的研究工作。

1952 年霍夫曼(D. A. Huffman)给出最优变长码的构造方法[5]。同年贝尔实验室的奥利弗(B. M. Oliver)等人开始研究线性预测编码理论；1958 年格雷哈姆(Graham)用计算机模拟法研究图像的 DPCM 编码方法；1966 年奥尼尔（J. B. O’ Neal）对比分析了 DPCM 和 PCM，对电视信号传输进行了理论分析和计算机模拟，并提出了用于电视的实验数据，又于 1969 年进行了线性预测的实验。

20 世纪 60 年代，科学家们也开始探索比预测编码效率更高的编码方法。人们首先讨论了包括 KL 变换、傅立叶变换等正交变换。1968 年安德鲁斯（H. C. Andrews）等人采用二维离散傅立叶变换(2D-DFT)提出了变换编码。此后相继出现了沃尔什-哈达玛（Walsh-Hadamard）变换、斜变换(Slant 变换，由 Enomoto 和 Shibata 引入)、K-L 变换、离散余弦变换（DCT）等。

1976 年美国贝尔系统的克劳切(R. E. Crochjiere)等人引入了语音的子带编码，1985 年奥尼尔（S.D. O’ Neil）将子带编码推广到对图像的编码。

1983 年瑞典的 Forchheimer 和 Fahlander 提出了基于模型图像编码[6]（Model-Based Coding）。在后来的图像编码会议（PCS，Picture Coding Symposium）上（1988~1996 年），以及其他一些国际会议对极低码速率的视频编码进行了深入的研究工作。

1986 年，Meyer 在理论上证明了一维小波函数的存在[7]，创造性地构造出具有一定衰减特性的小波函数。1987 年 Mallat 提出了多尺度分析的思想及多分辨率分析的概念，成功地统一了在此之前各种具体小波的构造方法，提出了相应的快速小波算法——Mallat 算法，并把它有效地应用于图像分解和重构；1989 年，小波变换开始用于多分辨率图像描述。

与小波变换的提出几乎同时，另外一些科学家探讨了使用分数维理论进行数据压缩。1988 年美国 Georgia 理工学院的 M. F. Barnsley 在 BYTE 上发表了分形压缩方法，1992 年 A. Jacquin 实现分块迭代函数系统（PIFS），完善了分形编码压缩方法。

1988 年在图像压缩编码的发展历史中是极为重要的一年。几十年研究的成果集中表

现在确定了 H.261 和 JPEG 两个建议的原理框架，奠定了 20 世纪 90 年初相继提出的 MPEG-1、MPEG-2、H.263 等标准的基础。

1991 年 3 月，“联合图片专家组”（JPEG，Joint Photographic Expert Group）提出 JPEG 标准草案，1994 年正式通过[8]（ISO 10918）。1991 年为二值图像编码制订了 JBIG 标准（ISO 11544）。新的 JPEG 版本是 JPEG-LS（ISO/IEC 14495，1999），和 JPEG 2000（ISO/IEC 15444，等同的 ITU-T 编号 T.800），于 1999 年 3 月形成工作草案，2000 年正式颁布的。JPEG 的这些标准主要应用于静止图像处理。

1992 年，“运动图片专家组”（MPEG，Moving Picture ExpertGroup）提出了“用于数字存储媒体运动图像及其伴音率为 1.5Mbit/s 的压缩编码”，简称为 MPEG-1，作为 ISO CD11172 号建议通过，1993 年正式通过。

1993 年提出 MPEG-2 标准草案[9]，1994 年正式通过（ISO/IEC 13818，视频部分为 ITU-T H.262），处理能力可达广播级水平。MPEG-2 标准兼容 MPEG-1 标准，适应于 1.5Mbit/s～80Mbit/s 编码范围。MPEG-2 标准也是 DVD 和高清晰度电视（HDTV）全数字方案所采用的数据压缩标准。

1995 年 1 月，国际标准化组织 ITU-TLBC 工作组为极低数码率可视电话（VeryLowBitRateVisualTelephony）的工作形成了 H.263 视频压缩编码草案。1997 年 11 月提出了 MPEG-4 用于极低数码率数据压缩，1999 年 MPEG-4 形成国际标准[10]。

H.263 (H.261 P x 64) 涉及低分辨率的视频序列。它可以与为 ISDN 和移动通信开发的音频编码标准一起实现，这些标准已成为 CCITT 标准[11]。

H.264 即 MPEG-4 的第 10 部分[12]，H.264 采用了更小的宏块和更加精细的运动补偿，不仅比 H.263 和 MPEG-4 节约了 50%的码率，而且对网络传输具有更好的支持功能。

AVS 作为中国自主知识产权的音视频压缩标准，视频部分已经于 2003 年 12 月 19 日最终定搞。AVS 以当前国际上最先进的 MPEG-4 AVC/ H.264 框架为起点，采用的核心技术包括：8x8 整数变换、量化、帧内预测、1/4 精度像素插值、特殊的帧间预测运动补偿、二维熵编码、去块效应环内滤波等。其压缩效率明显比现在在数字电视、光存储媒体中常用的 MPEG-2 视频提高一个层次。在压缩效率相当的前提下，又较 MPEG-4 AVC/ H.264 的 main profile 的实现复杂度大为降低。

### 1.1.4压缩带来的问题

在所有的实际节目素材中，存在着两种类型的信号分量：即异常的、不可预见的信号分量和可以预见的信号分量。异常分量称为熵，它是信号中的真正信息。其余部分称为冗余，因为它不是必需的信息。冗余可以是空间性的，如在图象的大片区域中，邻近象素几乎具有相同的数值。冗余也可以是时间性的，例如连续图象之间的相似部分。在所有的压缩系统编码器中都是将熵与冗余相分离，只有熵被编码和传输，而在解码器中再从编码器的发送的信号中计算出冗余[13]。图 1-1 表示了这一过程。

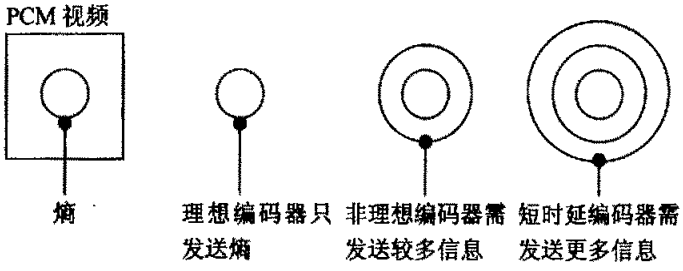



图 1-1 压缩编码原理

一个理想的编码器可以抽取所有的熵并只将熵传输到解码器。理想的解码器能够从熵中恢复原始信号。然而实际上，这种理想的编码解码器是不可能实现的。因为这种理想的编码器在技术上很复杂，而且为了使用时间性冗余而造成很长的延时。在某些应用中，例如节目记录或某些广播传输中，有些延时还是可以接受的，但在视频会议中却是不允许的。还有，一个非常复杂的编码器在价格上也是很昂贵的。这也就是说，不存在一个理想的压缩系统。

## 1.2 问题提出与研究意义

离散余弦变换是目前应用最广泛的图像、视频压缩算法。通过离散余弦变换可以获得很高的数据压缩比，它可以将大量的信息浓缩到少数的变换系数中。对于自然图像，离散余弦变换的压缩性能接近最佳 K-L 变换。并且，离散余弦变换是不依赖于数据的，它的变换矩阵拥有很好的对称性，便于通过软、硬件高效的实现。

离散余弦变换广泛应用于图像压缩、数据压缩、滤波等领域。尤其在图像压缩领域，基于离散余弦变换(DCT)的编码方法是 JPEG 算法的核心内容；MPEG-x、H.26x 等标准均采用离散余弦变换作为图像数据空间域压缩的重要手段之一。

离散余弦变换(DCT/IDCT)以及运动向量估计(ME)、运动向量补偿(MC)作为 MPEG-x 以及 H.26x 系列标准的核心部分，消耗着大量的计算时间。Mitsuo 等人的文章[14]指出，在 MPEG-2 编码过程中，以上 4 部分占据了 88%以上的计算量；而离散余弦变换(DCT/IDCT)占据了大约全部计算量的 22%；这意味着对于 MPEG-2 SP@ML 的视频流编码过程，编码器的运算能力必须超过 3.7GOPS/秒。

促使人们对离散余弦变换及其 FPGA 设计感兴趣的是以下四个基本原因：

1. 许多图像、视频压缩标准都采用 DCT 变换作为消除图像数据空间冗余的重要手段，这些标准包括 JPEG、H.26x、MPEG-x 等。DCT 需要的计算量很大，在以上各个标准的编、解码过程， DCT 计算占据了大量的时间，因此，研究 DCT 可以更好的实现现有标准；
2. 用硬件(FPGA、VLSI)实现 DCT 时，面积、速度、精度三个参数相互制约，如果

能找到一个算法结构，在满足应用要求的大前提下，尽量减少所需的芯片面积，提高速度和精度。将极大的降低硬件实现的成本，提高图像的质量，促进数字视频的发展；

3. 研究 DCT 的 FPGA 设计，结构和规则可以作为离散正弦变换(DST)、离散小波变换(DWT)、离散傅立叶变换(DFT)等 FPGA 设计的参考。
4. 现有的图像压缩方法，数据压缩比以及图像压缩质量仍然不能很好的统一，可以研究 DCT 和其他方法相结合，如和基于小波的方法或者基于内容的压缩方法相结合，提高现有图像的压缩比。

## 1.3 研究内容与结构安排

本文研究的主要目的是通过对 DCT 快速算法的研究和 FPGA 设计的研究，了解 DCT 算法 FPGA 实现的一般过程；发现现有 DCT 实现结构的优点与不足；并努力改进现有的 DCT 实现结构，争取在速度、面积、功耗等方面超越现有的设计结构。

本文共分为六章。第一章是绪论部分，阐明了本文的研究背景，详细介绍了数据压缩原理、意义，以及图像、视频压缩的发展史及现状。

第二章研究了基于 DCT 的图像、视频压缩编码标准，就各标准的内容和创新，以及它们相互之间的关系进行了讨论，给出了 MPEG-x 系列视频压缩的基本流程。主要为后续研究提供应用知识背景。

第三章介绍了 DCT 算法的数学基础，研究了各种快速 DCT 算法及各种 2-D DCT 算法，重点介绍了 Loeffler 算法以及二维 DCT/IDCT 的硬件(FPGA、VLSI)实现的一般结构。然后研究了现有的 DCT 实现方案。列举了采用通用 CPU、专用数字信号处理器、FPGA 以及 ASIC 实现的各种方案，并对各自的特点进行分析，指出了其优点与不足。

第四章详细介绍了本文设计的二维 DCT/IDCT 处理核，通过行列分解将二维 DCT/IDCT 分解为两重的一维 DCT/IDCT 变换，并且采用多级流水线结构设计，提高了处理核的性能。对处理核进行了严格的内部字长仿真、功能仿真以及时序仿真，给出了处理核的性能参数以及资源使用情况。

最后一章对全文的工作进行了总结，并指出工作中存在的问题及今后的研究方向。

# 第 2 章　基于 DCT 的图像压缩标准

## 2.1 引言

离散余弦变换(DCT)广泛应用于图像压缩、数据压缩、滤波等领域，已经被证明在是图像和视频压缩编码中非常有效的技术之一。很多图像、视频压缩标准都采用了 DCT 变换。目前，图像压缩方面的国际标准主要有 MPEG 系列(MPEG-1、MPEG-2、MPEG-4)，以及 H.26X 系列(H.261、H.263、H.263+、H.264)和静止图像压缩标准 JPEG、JPEG2000 等；除了 JPEG2000 以外，以上标准全部采用基于 DCT 的图像压缩算法。

为了了解视频压缩的整体架构以及数据压缩流程，下面我们以 MPEG-2 为例进行讲解。

## 2.2 视频压缩的基本原理

MPEG 可以完成对视频和音频的压缩，这里我们当然只谈视频压缩部分。通过压缩去除视频讯息源中的 3 重冗余度：空间冗余度、时间冗余度和结构冗余度。

1. 同一帧讯源图像中相邻像素之间的幅度值相近，即同一行上的相邻像素之间幅值相近，或者相邻行之间同样位置上的像素幅值相近，被称为图像的空间冗余度；
2. 相邻两帧讯源图像同一位置上像素幅度值相近，体现了讯源图像的时间冗余度；
3. 讯源图像上每个像素所用 bit 数的多少表示了比特结构，多用的比特数为冗余量，体现了静态冗余度。

MPEG 主要从两个方面降低冗余度：

1. 利用图像信号的统计特性进行压缩。采用运动补偿（MC）去除时间冗余度；采用离散余弦变换（DCT）和游程长度编码（RLC）去除空间冗余度；采用可变长度编码（VLC）去除静态（比特结构）冗余度。
2. 利用人的视觉生理特性设计压缩。人眼对构成图像的不同频率成分、物体的不同运动程度等具有不同的敏感度，这是由人眼的视觉生理特性所决定的，眼睛对亮度的敏感程度要大于对色彩的敏感程度。据此，可控制图像适合于人眼的视觉特性，从而达到压缩图像数据量的目的[15]。

MPEG-2 标准压缩首先对色差信号进行亚采样，减少数据量，采用运动补偿技术，减少帧间冗余度，利用二维 DCT 变换去除空间相关性，对 DCT 分量进行量化，舍去不重要少帧间冗余度，利用二维 DCT 变换去除空间相关性，对 DCT 分量进行量化，舍去不重要

的信息，将量化后 DCT 分量按照频率重新排序，将 DCT 分量进行变字长编码，最后对每个数据块的直流分量（DC）进行预测差分编码。MPEG 视频的编码框图如图 2-1 所示。

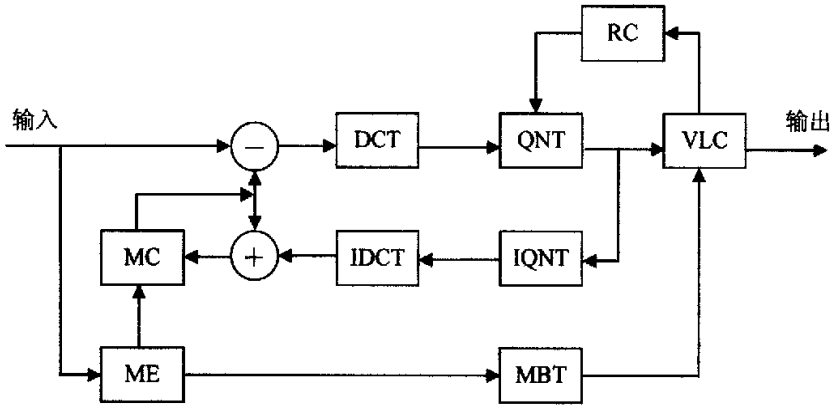



**图 2-1 MPEG-2 编码器的功能模型**

## 2.2.1 离散余弦变换 DCT

MPEG 采用了 Ahmed 等人于 70 年代提出的离散余弦变换（DCT-Discrete Cosine Transform）压缩算法，降低视频信号的空间冗余度。

DCT 将运动补偿误差或原画面信息块转换成代表不同频率分量的系数集，这有两个优点：其一，信号常将其能量的大部分集中于频率域的一个小范围内，这样一来，描述不重要的分量只需要很少的比特数；其二，频率域分解映射了人类视觉系统的处理过程，并允许后继的量化过程满足其灵敏度的要求。

DCT 变换过程本身虽然并不产生码率压缩作用，但是变换后的频率系数却非常有利于码率压缩。

MPEG-2 的编码码流分为六个层次。从上至下依次为：视频序列层，图像组层，图像层，像条层，宏块层和像块层。

宏块层之下是像块层，像块是 MPEG-2 码流的最底层，是 DCT 变换的基本单元。MP@ML 中一个像块由 8×8 个抽样值构成，同一像块内的抽样值必须全部是 Y 信号样值，或全部是 Cb 信号样值，或全部是 Cr 信号样值。另外，像块也用于表示 8×8 个抽样值经 DCT 变换后所生成的 8×8 个 DCT 系数。

DCT 是一种空间变换，在 MPEG-2 中 DCT 以 8×8 的像块为单位进行，生成的是 8×8 的 DCT 系数数据块。DCT 变换的最大特点是对于一般的图像都能够将像块的能量集中于少数低频 DCT 系数上，即生成 8×8DCT 系数块中，仅左上角的少量低频系数数值较大，其余系数的数值很小，这样就可能只编码和传输少数系数而不严重影响图像质量。

DCT 不能直接对图像产生压缩作用，但对图像的能量具有很好的集中效果，使得大多数的信息集中在直流及低频分量上，为紧随其后的压缩工作打下了基础。

### 2.2.2量化

量化是针对 DCT 变换系数进行的，量化过程就是以某个量化步长去除 DCT 系数。量化步长的大小称为量化精度，量化步长越小，量化精度就越细，包含的信息越多，但所需的传输频带越高。不同的 DCT 变换系数对人类视觉感应的重要性是不同的，因此编码器根据视觉感应准则，对一个 8×8 的 DCT 变换块中的 64 个 DCT 变换系数采用不同的量化精度，以保证尽可能多地包含特定的 DCT 空间频率信息，又使量化精度不超过需要。DCT 变换系数中，低频系数对视觉感应的重要性较高，因此分配的量化精度较细；高频系数对视觉感应的重要性较低，分配的量化精度较粗，通常情况下，一个 DCT 变换块中的大多数高频系数量化后都会变为零。

### 2.2.3之型扫描与游程编码

DCT 变换产生的是一 8×8 的二维数组，为进行传输，还须将其转换为一维排列方式。有两种二维到一维的转换方式，或称扫描方式：之型扫描(Zig-Zag)和交替扫描，其中之型扫描是最常用的一种。由于经量化后，大多数非零 DCT 系数集中于 8×8 二维矩阵的左上角，即低频分量区，之型扫描后，这些非零 DCT 系数就集中于一维排列数组的前部，后面跟着长串的量化为零的 DCT 系数，这些就为游程编码创造了条件。

游程编码中，只有非零系数被编码。一个非零系数的编码由两部分组成：前一部分表示非零系数前的连续零系数的数量(称为游程)，后一部分是那个非零系数。这样就把之型扫描的优点体现出来了，因为之型扫描在大多数情况下出现连零的机会比较多，游程编码的效率就比较高。当一维序列中的后部剩余的 DCT 系数都为零时，只要用一个“块结束”标志(EOB)来指示，就可结束这一 8×8 变换块的编码，产生的压缩效果是非常明显的。

### 2.2.4熵编码

量化仅生成了 DCT 系数的一种有效的离散表示，实际传输前，还须对其进行比特流编码，产生用于传输的数字比特流。简单的编码方法是采用定长码，即每个量化值以同样数目的比特表示，但这种方法的效率较低。而采用熵编码可以提高编码效率。熵编码是基于编码信号的统计特性，使得平均比特率下降。游程和非零系数既可独立的，也可联合的作熵编码。熵编码中使用较多的一种是霍夫曼编码，MPEG-2 视频压缩系统中采用的就是霍夫曼编码。霍夫曼编码中，在确定了所有编码信号的概率后生产一个码表，对经常发生的大概率信号分配较少的比特表示，对不常发生的小概率信号分配较多的比特表示，使得整个码流的平均长度趋于最短。

### 2.2.5信道缓存

由于采用了熵编码，产生的比特流的速率是变化的，随着视频图像的统计特性变化。但大多数情况下传输系统分配的频带都是恒定的，因此在编码比特流进入信道前需设置信道缓存。信道缓存是一缓存器，以变比特率从熵编码器向里写入数据，以传输系统标称的恒定比特率向外读出，送入信道。缓存器的大小，或称容量是设定好的，但编码器的瞬时输出比特率常明显高于或低于传输系统的频带，这就有可能造成缓存器的上溢出或下溢出。因此缓存器须带有控制机制，通过反馈控制压缩算法，调整编码器的比特率，使得缓存器的写入数据速率与读出数据速率趋于平衡。缓存器对压缩算法的控制是通过控制量化器的量化步长实现的，当编码器的瞬时输出速率过高，缓存器将要上溢时，就使量化步长增大以降低编码数据速率，当然也相应增大了图像的损失；当编码器的瞬时输出速率过低，缓存器将要下溢出时，就使量化步长减小以提高编码数据速率。

### 2.2.6运动估计/运动补偿

运动估计使用于帧间编码方式时，通过参考帧图像产生对被压缩图像的估计。运动估计的准确程度对帧间编码的压缩效果非常重要。如果估计作的好，那么被压缩图像与估计图像相减后只留下很小的值用于传输。运动估计以宏块为单位进行，计算被压缩图像与参考图像的对应位置上的宏块间的位置偏移。这种位置偏移是以运动矢量来描述的，一个运动矢量代表水平和垂直两个方向上的位移。运动估计时，P 帧和 B 帧图像所使用的参考帧图像是不同的。P 帧图像使用前面最近解码的 I 帧或 P 帧作参考图像，称为前向预测；而 B 帧图像使用两帧图像作为预测参考，称为双向预测，其中一个参考帧在显示顺序上先于编码帧(前向预测)，另一帧在显示顺序上晚于编码帧(后向预测)，B 帧的参考帧在任何情况下都是 I 帧或 P 帧。

利用运动估计算出的运动矢量，将参考帧图像中的宏块移至水平和垂直方向上的相对应位置，即可生成对被压缩图像的预测。在绝大多数的自然场景中运动都是有序的。因此这种运动补偿生成的预测图像与被压缩图像的差分值是很小的。

## 2.3 图像压缩标准

目前视频流传输中最为重要的编解码标准有国际电联的 H.261、H.263 等，联合图像专家组的 JPEG、JPEG2000，国际标准化组织运动图像专家组的 MPEG 系列标准等。

国内为抢占高清领域，也推出了许多媒体压缩、传输标准，其中 EVD 主要面向数字媒体的存储领域，AVS 主要面向数字媒体的传输领域，两者都拥有良好的发展前景。但是由于缺乏相关文档，这里我们不作讨论。

### 2.3.1静止图像压缩标准

国际标准化组织(ISO)和国际电报电话咨询委员会(CCITT)联合成立的专家组JPEG(Joint Photographic Experts Group)经过五年艰苦细致地工作后，于 1991 年 3 月提出了 ISO CDI0918 号建议草案[8]：多灰度静止图像的数字压缩编码(通常简称为 JPEG 标准)。这是一个适用于彩色和单色多灰度或连续色调静止数字图像的压缩标准。它包括基于 DPCM(差分脉冲编码调制)、DCT(离散余弦变换)和 Huffman 编码的有损压缩算法两个部分。前者不会产生失真，但压缩比很小；后一种算法进行图像压缩时信息虽有损失但压缩比可以很大，例如压缩 20 倍左右时，人眼基本上看不出失真。

JPEG 标准实际上有三个范畴：

1. 基本顺序过程 Baseline Sequential processes 实现有损图像压缩，重建图像质量达到人眼难以观察出来的要求。采用的是 8×8 像素自适应 DCT 算法、量化及 Huffman 型的熵编码器。
2. 基于 DCT 的扩展过程(Extended DCT Based Process)使用累进工作方式，采用自适应算术编码过程。
3. 无失真过程(Losslesss Process)采用预测编码及 Huffman 编码(或算术编码)，可保证重建图像数据与原始图像数据完全相同。

其中的基本顺序过程是 JPEG 最基本的压缩过程：符合 JPEG 标准的硬软件编码/解码器都必须支持和实现这个过程。

### 2.3.2视频通信编码标准—H.26x 系列

#### H.261

H.261 又称为 P*64，其中 P 为 64kb/s 的取值范围，是 1 到 30 的可变参数，它最初是针对在 ISDN 上实现电信会议应用特别是面对面的可视电话和视频会议而设计的。实际的编码算法类似于 MPEG 算法，但不能与后者兼容。H.261 在实时编码时比 MPEG 所占用的 CPU 运算量少得多，此算法为了优化带宽占用量，引进了在图像质量与运动幅度之间的平衡折中机制，也就是说，剧烈运动的图像比相对静止的图像质量要差。因此这种方法是属于恒定码流可变质量编码而非恒定质量可变码流编码。

#### H.263

H.263 是国际电联 ITU-T 的一个标准[11]，是为低码流通信而设计的。但实际上这个标准可用在很宽的码流范围，而非只用于低码流应用，它在许多应用中可以认为被用于取代 H.261。H.263 的编码算法与 H.261 一样，但做了一些改善和改变，以提高性能和纠错能力。H.263 标准在低码率下能够提供比 H.261 更好的图像效果，两者的区别有：(1)H.263 的运动补偿使用半象素精度，而 H.261 则用全象素精度和循环滤波；(2)数据

流层次结构的某些部分在 H.263 中是可选的，使得编解码可以配置成更低的数据率或更好的纠错能力；(3)H.263 包含四个可协商的选项以改善性能；(4)H.263 采用无限制的运动向量以及基于语法的算术编码；(5)采用事先预测和与 MPEG 中的 P-B 帧一样的帧预测方法；(6)H.263 支持 5 种分辨率，即除了支持 H.261 中所支持的 QCIF 和 CIF 外，还支持 SQCIF、4CIF 和 16CIF，SQCIF 相当于 QCIF 一半的分辨率，而 4CIF 和 16CIF 分别为 CIF 的 4 倍和 16 倍。

1998 年 IUT-T 推出的 H.263＋是 H.263 建议的第 2 版，它提供了 12 个新的可协商模式和其他特征，进一步提高了压缩编码性能。如 H.263 只有 5 种视频源格式，H.263＋允许使用更多的源格式，图像时钟频率也有多种选择，拓宽应用范围；另一重要的改进是可扩展性，它允许多显示率、多速率及多分辨率，增强了视频信息在易误码、易丢包异构网络环境下的传输。另外，H.263＋对 H.263 中的不受限运动矢量模式进行了改进，加上 12 个新增的可选模式，不仅提高了编码性能，而且增强了应用的灵活性。H.263 已经基本上取代了 H.261。

**H.264**

JVT（Joint Video Team，视频联合工作组）于 2001 年 12 月在泰国 Pattaya 成立。它由 ITU-T 和 ISO 两个国际标准化组织的有关视频编码的专家联合组成。JVT 的工作目标是制定一个新的视频编码标准，以实现视频的高压缩比、高图像质量、良好的网络适应性等目标。目前 JVT 的工作已被 ITU-T 接纳，新的视频压缩编码标准称为 H.264 标准，该标准也被 ISO 接纳，称为 AVC（Advanced Video Coding）标准，是 MPEG-4 的第 10 部分[12]。

H.264 不仅比 H.263 和 MPEG-4 节约了 50%的码率，而且对网络传输具有更好的支持功能。它引入了面向 IP 包的编码机制，有利于网络中的分组传输，支持网络中视频的流媒体传输。H.264 具有较强的抗误码特性，可适应丢包率高、干扰严重的无线信道中的视频传输。H.264 支持不同网络资源下的分级编码传输，从而获得平稳的图像质量。H.264 能适应于不同网络中的视频传输，网络亲和性好。

### 2.3.3运动图像压缩标准—MPEG 系列

该系列标准目前包括了 MPEG-1、MPEG-2、MPEG-4、MPEG7、MPEG-21 等。其中 MPEG7 用于多媒体检索，MPEG-21 用于媒体交换。这里我们仅讨论 MPEG-1、MPEG-2、MPEG-4。

**MPEG-1**

MPEG-1 标准于 1993 年 8 月公布[9]，用于传输 1.5Mbps 数据传输率的数字存储媒体运动图像及其伴音的编码。该标准从颁布的那一刻起，MPEG-1 取得一连串的成功，如 VCD 和 MP3 的大量使用。

**MPEG-2**

MPEG-2 制定于 1994 年，设计目标是高级工业标准的图像质量以及更高的传输率。MPEG-2 所能提供的传输率在 3MB－10MB / s 间,在 NTSC 制式下的分辨率可达 720×486，MPEG-2 能够提供广播级的视像和 CD 级的音质。

**MPEG-4**

MPEG-4 于 1998 年 11 月公布[10]，它不仅是针对一定比特率下的视频、音频编码，更加注重多媒体系统的交互性和灵活性。这个标准主要应用于视像电话、视像电子邮件等，对传输速率要求较低，在 4.8k - 64kbits / s 之间，分辨率为 176×144。MPEG-4 利用很窄的带宽，通过帧重建技术、数据压缩，以求用最少的数据获得最佳的图像质量。

## 2.4 数据率分析

### 2.4.1ITU-R BT.601 彩色电视图像数字化标准

为了在 PAL、NTSC 和 SECAM 彩色电视制之间确定一个共同的数字化参数，早在 1982 年国际无线电咨询委员会就制定了演播室的数字电视编码标准，即 ITU-R BT.601 标准。按照这个标准，使用 4:2:2 的采样格式，亮度信号 Y 的采样频率选择为 13.5MHz/s，而色差信号 Cr 和 Cb 的采样频率选择为 6.75MHz/s，在传输数字电视信号通道上的数据传输率为 270Mb/s(兆比特/秒)，即

亮度（Y):

858 样本/行×525 行/帧×30 帧/秒×10 比特/样本＝ 135 兆比特/秒(NTSC)

864 样本/行×625 行/帧×25 帧/秒×10 比特/样本＝135 兆比特/秒(PAL)

Cr 和 Cb:

429 样本/行×525 行/帧×30 帧/秒×10 比特/样本＝68 兆比特/秒(NTSC)

429 样本/行×625 行/帧×25 帧/秒×10 比特/样本＝68 兆比特/秒(PAL)

总计:

(13.5+6.8+6.8)兆样本/秒×10 比特/样本 = 271 兆比特/秒

### 2.4.2HDTV 高清晰数字电视

HDTV 全称 High Definition Television，中文叫高清晰数字电视。它是采用数字信号来表示电视图像信息，在电视信号的采集、记录、处理、存储、播出、传输和接收过程中使用数字技术的系统。全数字化高画质（8 倍密）广视角(16:9)及高音质(5.1 声道的)数字传输广播和接收(数字电视)。

根据场扫描方式的不同，目前 HDTV 可分为三种模式，分别是：720p（1280×720，

“p”代表逐行扫描）1080i（1920×1080，“i”隔行扫描）1080p（1920×1080，“p”代表逐行扫描）其中 720P 和 1080i 格式的 HDTV 最为常见。

按照上一节的公式，我们可以计算得到未经压缩的 HDTV 的数据率最大可达 2.488Gb/s。

1999 年 10 月 1 日，中央电视台首次采用 HDTV 技术转播的国庆 50 周年庆典实况所用的就是 16：9HDTV 图像格式(1920×1080×59.94Hz/2：1/1125)，即采用美国 ATSC 标准 1920×1080 像素点,场频为 59.94Hz，隔行 1125 像素行扫描，NTSC 制式标准图像格式。

模拟电视信号经过采样和量化之后，得到的数字电视信号的数据量大得惊人，因此要对数字电视信号进行压缩，以便于存储和传输。

# 第 3 章 DCT 的数学基础及其快速算法

## 3.1 引言

图像编码过程中的离散余弦变换编码包括两个阶段：离散余弦变换和量化编码。离散余弦变换属于一种正交变换，就是把信号分解为余弦函数系的线性叠加，藉此可以通过处理系数来代替处理信号。它的作用是使空间域的能量重新分布，降低信号的相关性，它的变换是可逆的。然而离散余弦变换本身并不能达到数据压缩的作用，而要实现数据压缩，就要选择适当的比特分配方案和量化方法，量化过程是不可逆的或者说是有损压缩。

目前广泛使用的压缩算法采用的都是 8×8 的二维 DCT 变换，二维 DCT 变换可通过行列分解的方法，变换成两重的一维 DCT 变换。虽然人们对于一维 DCT 算法已经有了非常多的研究，Loeffler 算法[16]的乘法计算次数已经减少到了理论极限值，研究似乎已经到了尽头。然而对于高效率的二维 DCT 变换算法的研究开始得比较晚，因而这方面仍然有优化的空间。

## 3.2 一维 DCT 算法的发展概况

1974 年由 N. Ahmed 和 K. R. Rao 提出了离散余弦变换 DCT[17]，由于 DCT 变换在假定信源为一阶平稳马尔可夫过程时最接近 K－L 变换且有快速算法，1991 年 Andreas S. Spanias 等人验证了离散余弦变换在 4 种正交变换(DFT、DCT、KLT、WHT)中是最有活力的[18]，因此常被作为变换编码的基本方法。

N 点 DCT/IDCT 的定义如公式(3-1)、公式(3-2)所示：

$$Y(k)=\sqrt{\frac{2}{N}}\cdot a_k\cdot\sum_{n=0}^{N-1}X(n)\cdot\cos(\frac{(2n+1)\cdot k\pi}{2N}) \tag{3-1}$$

$$X(n)=\sqrt{\frac{2}{N}}\cdot\sum_{k=0}^{N-1}a_k\cdot Y(k)\cdot\cos(\frac{(2n+1)k\pi}{2N}) \tag{3-2}$$

其中 a0=$\sqrt{2}$/2，ak=1，k=1, 2, …, N-1，n=0, 1, …, N-1

对于 8 点的 DCT/IDCT，直接按照以上公式计算需要 64 次乘法和 56 次加法才能完成，当然，我们可以采用许多方法来减少 DCT 计算所需的加法和乘法运算，因为乘法运算的代价远高于加法运算，我们着重考虑减少乘法运算的次数。

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[19]。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[20]，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、基于其他变换(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 DCT 被提出来之后的 20 年时间里，提出了许多计算离散余弦变换的算法。这些算法大都需要 12－13 个乘法和 29 个加法来实现一个 8 点的 DCT，如表 3-1 所示。

**表 3-1 8 点 DCT 运算量对比**

| 作者 | Chen | Wang | Lee | Vetterli | Suehiro | Hou |
|---|---|---|---|---|---|---|
| 乘法次数 | 16(13) | 13 | 12 | 12 | 12 | 12 |
| 加法次数 | 26(29) | 29 | 29 | 29 | 29 | 29 |

Chen 的快速算法[20]是第一个发表的算法，这个算法采用了非常规则的结构，算法中用到的乘法器和加法器数目可以很容易地减少到表 3-1 括号中的数字，方法是用后来发表的算法中计算蝶形运算的算法(通过算术变形，可以将蝶形运算所需要的乘法和加法的数量由 4 个 2 个变为 3 个 3 个)。

Wang 的算法[21]不仅适用于 DCT，而且通过简单的变换，我们就能够从 Wang 的算法得到同样适用于离散正弦变换(DST)、离散小波变换(DWT)、离散傅立叶变换(DFT)的算法。

Lee 的算法[22]的第一级非常规则，但在最后一级采用了不规则的数据流，并且 Lee 的算法需要使用反余弦值作为系数，而这有可能引起数据溢出的问题。

Vetterli 在他的算法[23]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之后，这增加了算法中通讯结构的复杂度。

Suehiro 的算法[24]需要的乘法的个数比 Wang 的少。通过适当的变换，我们也可以从 Suehiro 的算法得到同样适用于离散正弦变换(DST)、离散小波变换(DWT)、离散傅立叶变换(DFT)的算法。

Hou 提出了一个递归的算法[25]，基于每个长度为 N 的 DCT 可以看作由两个长度为 N/2 的 DCT 实现。这个算法的数据流很规则，除了最后一级。

Duhamel 已经从理论上证明了 8 点 DCT 计算至少需要 11 个乘法[26]。即在有理数域上，计算长度为 $2^n$ 的一维 DCT 所需的最少实数乘法个数为 $2^{n+1}-n-2$。

Jie Liang 的文章[27]中提到了一种近似 DCT 算法(binDCT)，通过用一个分母为 $2^n$ 的分数近似的表示 DCT 计算中的因子，仅使用加法以及移位操作即可以得到 DCT 的近似结果。该算法适用于对计算精度要求较低的场合。

## 3.3 Loeffler 算法

本文采用的离散余弦变换算法最初是由 Loeffler，Ligtenberg，Moschytz 等提出来的，Loeffler 等人提出了一系列的算法[16]，对于 8 点的 DCT/IDCT 计算，这些算法都只需要 11 次乘法和 29 次加法。

### 3.3.1算法的基本结构

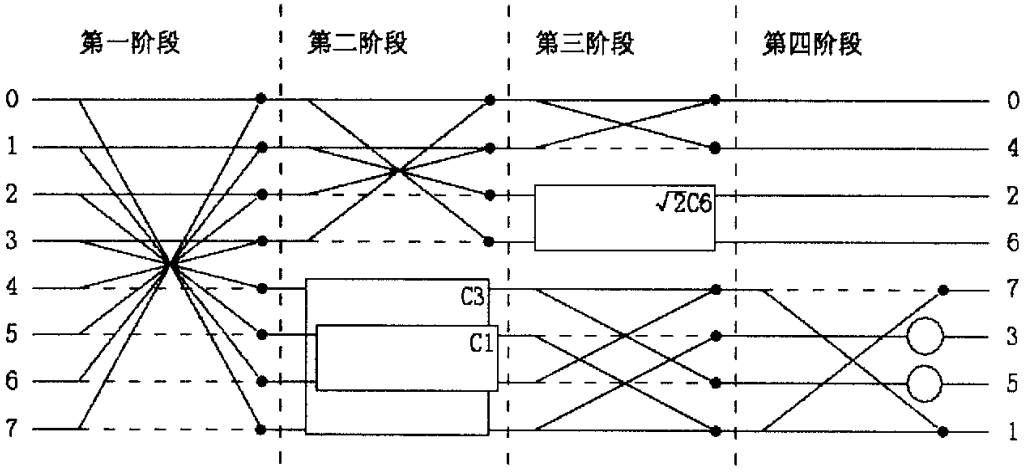



图 3-1 11 乘法的 8 点 DCT，符号解释参见图 3-2

这些算法将计算 8 点 DCT 所需要的乘法次数降低到了理论极限值，同时相对于之前的算法也没有增加加法的次数。其中之一如图 3-1 所示，算法需要分四步进行，并且由于数据相关性，这四个步骤只能顺序执行不能并行执行。而在每一个步骤内部的这些计算是可以并行执行的，从第二个步骤开始，算法分解为两个部分，一部分计算奇数位置的因子，一部分则用于计算偶数位置的因子。而计算偶数位置因子的那一部分恰好等同于一个 4 点的 DCT 计算；并且在第三个步骤 4 点的 DCT 计算再次分解为两个部分。图 3-2 解释了图 3-1 中所用符号的含义。

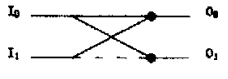
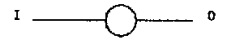
| 符号 | 等式 | 计算量 |
|---|---|---|
|  | $O_0 = I_0 + I_1$<br>$O_1 = I_0 - I_1$ | 2 个加法 |
|  | $O_0 = I_0 \cdot k \cdot \cos\frac{n\pi}{2N} + I_1 \cdot k \cdot \sin\frac{n\pi}{2N}$ | 3 个乘法 |
| | $O_1 = -I_0 \cdot k \cdot \sin\frac{n\pi}{2N} + I_1 \cdot k \cdot \cos\frac{n\pi}{2N}$ | 3 个加法 |
|  | $O = \sqrt{2} \cdot I$ | 1 个乘法 |

图 3-2 图 3-1 中使用的符号及其解释

图中的蝶形运算(第二个符号)原本需要 4 次乘法和 2 次加法，通过适当的数学变换，我们可以得到一个与之等价的式子，并且只需要 3 次乘法和 3 次加法，如公式(3-3)所示。

$$\begin{aligned} y_0 &= a\cdot x_0 + b\cdot x_1 = (b-a)\cdot x_1 + a\cdot(x_0 + x_1) \\ y_1 &= a\cdot x_0 + b\cdot x_1 = -(a+b)\cdot x_0 + a\cdot(x_0 + x_1) \end{aligned} \tag{3-3}$$

反离散余弦变换的算法结构与离散余弦变换相同，只是用相反的顺序计算。输出变成输入，输入变成输出。

### 3.3.2算法最后一阶段的变换

在计算算法中的偶数部分时(图 3-1 中的上面四行)，第二阶段和第三阶段可以交换，并且更换蝶形运算中的因子，而不影响结果(如图 3-3)。

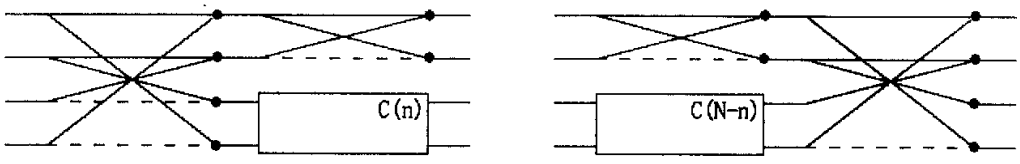



图 3-3 交换偶数部分的第二阶段和第三阶段

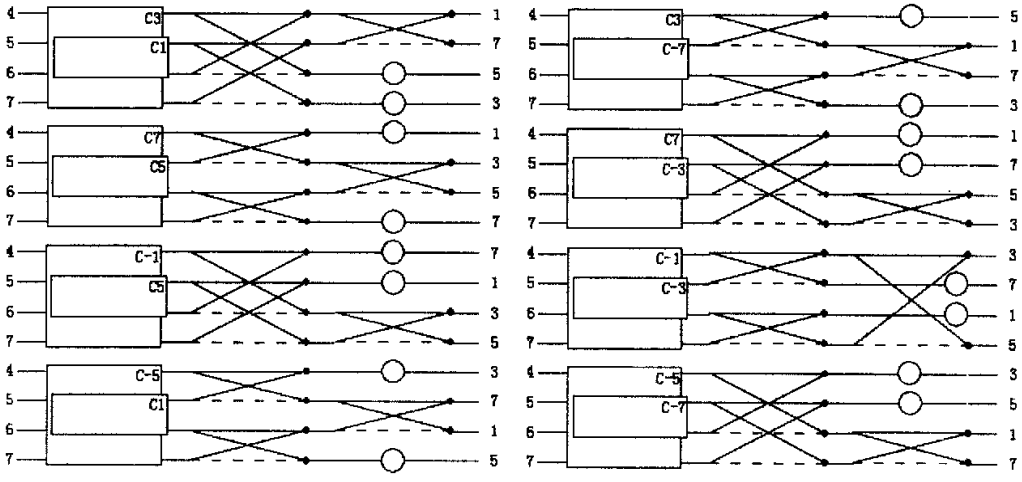



图 3-4 奇数部分第二、三、四阶段的变化

在奇数部分的计算中，有几个可能的变化：在第二阶段，每两个旋转中有 4 个不同的角度可以选择。在这些角度的 16 个组合中，有 8 个具有最小的计算复杂度(如图 3-4)。

我们可以看到，通过选择不同的第二阶段和第三阶段，奇数部分输出的顺序以及符号需要做相应的变化。

在这些变换中，我们也可以改变构成第二、三和第四阶段的模块的计算顺序(类似于偶数部分，如图 3-3 所示)。

### 3.3.3算法第一阶段的变化

在算法的第一阶段，除了前面我们给出的对称结构外，我们还发现了另外四个具有相同计算复杂度的结构，如图 3-5 所示。

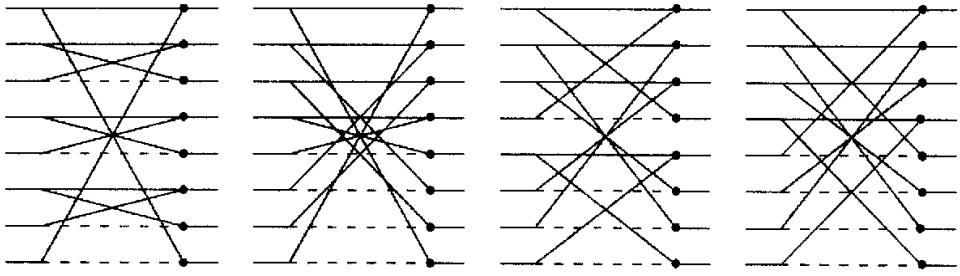


图 3-5 基本算法第一阶段的变化

图 3-6 为基本算法变化的一个例子，其中算法的第一级采用的是上图中所示的第一个结构。从算法结构可以看出，每个算法的最长路径上仍然有 2 个乘法器，4 个加法器，而且奇数和偶数部分直到第二级后才分开。

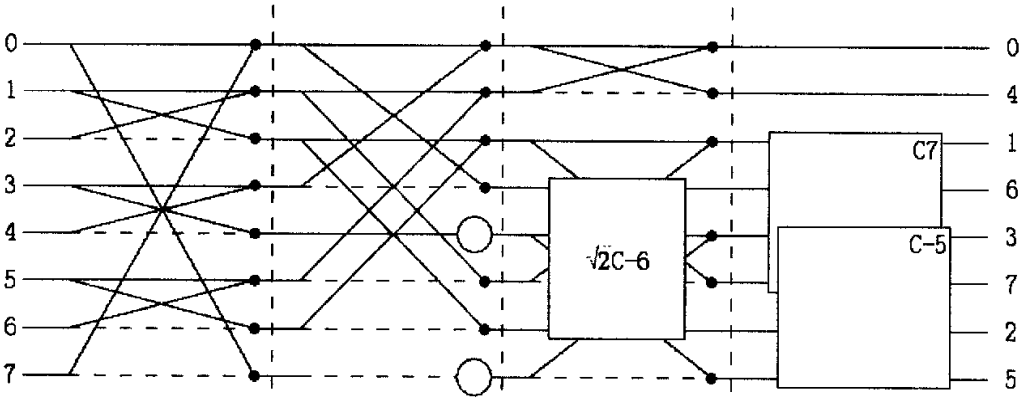



图 3-6 基本算法变化的例子

## 3.4 二维 DCT 变换

二维 DCT 将运动补偿误差或原画面信息块转换成代表不同频率分量的系数集[15]，这有两个优点：其一，视频信号常将其能量的大部分集中于频率域的 1 个小范围内，这样一来，描述不重要的分量只需要很少的比特数；其二，频率域分解映射了人类视觉系统的处理过程，并允许后继的量化过程满足其灵敏度的要求(对于肉眼敏感的信息，我们用较多的 bit 来记录，对于肉眼不敏感的信息，我们用较少的 bit 来记录)。

### 3.4.1原始计算公式

N×N 的二维 DCT 和二维 IDCT 的数学定义分别如公式(3-4)和公式(3-5)所示。

$$Z(k,l)=\frac{2}{N}a(k)a(l)\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}x(m,n)\cos\frac{(2m+1)\pi k}{2N}\cos\frac{(2n+1)\pi l}{2N} \tag{3-4}$$

$$x(m,n)=\frac{2}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}a(k)a(l)Z(k,l)\cos\frac{(2m+1)\pi k}{2N}\cos\frac{(2n+1)\pi l}{2N} \tag{3-5}$$

其中，a(0)=1/$\sqrt{2}$，a(k)=a(1)=1（当 k、1≠0 时），x(m, n)是原始图像，Z(k, 1)是各频率分量的大小（m, n, k, 1 =0, 1, 2, …, N-1）。

二维 DCT/IDCT 的矩阵形式如公式(3-6)所示：

$$\begin{aligned} Z &= AXA^T \\ X &= A^T ZA \end{aligned} \tag{3-6}$$

$$=\begin{bmatrix} & & & & - & - & - & - \\ & & - & - & - & - & f & c \\ d & -g & -b & -e & e & b & g & -d \\ a & -a & -a & a & a & -a & -a & a \\ e & -b & g & d & -d & -g & b & -e \\ f & -c & c & -f & -f & c & -c & f \\ g & -e & d & -b & b & -d & e & -g \end{bmatrix} \quad \text{其中} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} = \sqrt{\frac{2}{N}} \begin{bmatrix} \cos\frac{\pi}{4} \\ \cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} \\ \cos\frac{5\pi}{16} \\ \cos\frac{3\pi}{8} \\ \cos\frac{7\pi}{16} \end{bmatrix}$$

**图 3-7 矩阵 A 的值**

其中的矩阵 A 是一个正交矩阵，即 $AA^T = I_N$，对于 8 点的二维 DCT，矩阵 A 的值图 3-7 所示。

公式(6)可以分解成 2 个 4×4 的矩阵乘的形式，从而使运算量减半，如公式(3-7)、(3-8)、(3-9)、(3-10)所示。

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \begin{bmatrix} X(0)+X(7) \\ X(1)+X(6) \\ X(2)+X(5) \\ X(3)+X(4) \end{bmatrix} \tag{3-7}$$

$$\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} = \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X(0)-X(7) \\ X(1)-X(6) \\ X(2)-X(5) \\ X(3)-X(4) \end{bmatrix} \tag{3-8}$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} + \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} \quad (3\text{-}9)$$

$$\begin{bmatrix} X(7) \\ X(6) \\ X(5) \\ X(4) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} - \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} \quad (3\text{-}10)$$

### 3.4.2物理意义

其中 N 是像块的水平、垂直像素数，一般取 N=8。N 大于 8 时效率增加不多而复杂性大为增加。8×8 的二维数据块经 DCT 后变成 8×8 个变换系数，这些系数都有明确的物理意义。譬如当 U=0，V=0 时 F(0，0)是原 64 个样值的平均，相当于直流分量，随着 U，V 值增加，相应系数分别代表逐步增加的水平空间频率和垂直空间频率分量的大小。

可见图像信号被分解成为直流成分；以及从低频到高频的各种余弦成分；而 DCT 系数只是表示了该种成分所占原图像信号的份额大小；对应于 8×8 的像素块；其空间基底如图 3-8 所示：它是由 64 个像素值所组成的图像，通常也称之为基本图像。把它们称为基本图像是因为在离散余弦变换的反变换式中，任何像块都可以表示成 64 个系数的不同大小的组合。既然基本图像相当于变换域中的单一的系数，那么任何像元也可以看成由 64 个不同幅度的基本图像的组合[28]。这与任何信号可以分解成基波和不同幅度的谐波的组合具有相同的物理意义。

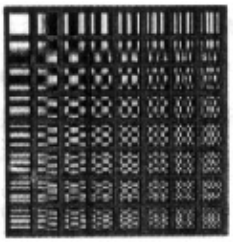


**图 3-8 基本图像**

经过二维 DCT 变换计算后，64 个样值仍然得到 64 个系数，本身码率并没有压缩；但是，经 DCT 变换后，比特数却增加了。原样值是 8 比特，数据从 0～255；得到的直流分量的最大值是原来 256 的 64/8 份，即 0～2047，交流分量的范围是-1024～1023；

但经过量化之后，大多数高频分量的系数变为 0，一般说来，人眼对低频分量比较敏感，对高频分量则不太敏感；因而量化的结果是去掉了不太重要的高频分量，降低了码率。

### 3.4.3二维 DCT/IDCT 快速算法

直接按照公式 3-4、公式 3-5 计算 8×8 的二维 DCT 变换，求得每个 DCT 系数需要 64 次乘法和 63 次加法，那么整个二维 DCT 变换总共需要 4096 次乘法和 4032 次加法。因为计算量巨大，人们研究了许多二维 DCT 的快速算法，下面做一个简要的介绍：

**行列分解法(RCM)**

二维 DCT 变换是可以被分解的[29]。对于 N×N 的二维 DCT，可以先计算 N 行的一维 DCT，然后对行 DCT 变换的结果计算 N 列的一维 DCT。二维 IDCT 变换也具有类似的特性。

行列分解法算法结构简单，数据流也很规则，非常适合于用硬件实现。

**AAN 及 FEIG 算法**

AAN 算法[30]计算 8×8 的二维 DCT 变换只需要 80 次乘法和 464 次加法。但是 AAN 算法计算得到的是放大了的 DCT 变换结果，而且在固定精度的定点运算中由于尺度转换合并到量化中导致计算结果不精确。FEIG 算法[31]在 AAN 算法的基础上发展而来，FEIG 算法计算 8×8 的二维 DCT 总共只需要 462 次加法、54 次乘法和 6 次左移操作。FEIG 较 AAN 算法速度更快，但是具有与 AAN 算法类似的缺点，并且结构更加复杂。

由于 AAN、FEIG 系列算法计算结果精度不高，同时算法结构较为复杂，多应用于基于通用 CPU 的软件方案。

**多项式变换法**

多项式变换法基于多项式环的特性(Polynomial transform based DCT implementation.)，通过蝶形加法运算进行列 DCT 计算，从而减少乘法数量。

基于多项式变换的算法[32],[33],[34]仍然利用二维 DCT 的行列分解特性，并且采用常见的一维 DCT 算法计算行 DCT，然后通过蝶形加法运算计算列 DCT。对于 8×8 的二维 DCT 变换，Duhamel 的多项式变换法[35]的乘法次数等于 8 个一维 8 点 DCT 变换所需的乘法次数。如果采用 Loeffler 算法来计算一维 DCT，则 Duhamel 的算法需要进行 88 次乘法。

## 3.5 已有的软、硬件实现方案

### 3.5.1基于通用 CPU 的方案

基于通用 CPU 的方案结构灵活，对算法的修改也可以快速实现；同时，互联网上拥有大量的源码资源供设计参考；因此，当推出一个新的视频压缩标准后，往往很快就会推出相应的软件编/解码方案。但是由于通用 CPU 的计算能力有限，通常无法完成实时的音、视频编码工作，而只能采取存储一编码工作的方式。即，将摄像机采集的原始数

据存储在硬盘中，然后再从影片中读取视频流进行编码。由于高分辨率的视频图像数据近乎海量，采用硬盘存储一部影片的原始视频数据是不现实的，这限制了通用 CPU 方案的应用，使得这种方案仅适用于验证算法的有效性或者将视频流从一种压缩格式转换到另外一种压缩格式。

FFmpeg 是一个开放源码的 MPEG 解决方案[35]，包含了对音、视频流的编码、转换、解码等功能模块。FFmpeg 的二维 DCT 编码部分采用的是浮点 AAN 算法[31]，

DivX 以及开放源码的 XviD[36]是最近在网络上极为流行的流媒体压缩格式，同时也是一套完整的流媒体编码解码方案。他们采用与 MPEG-4 类似的编码策略，能够在一张 650M 的 CD-ROM 上存放一部影片，其图像质量略低于 DVD。DivX 及 XviD 的二维 DCT 编码部分使用行列分解法，并使用 Loeffler 快速算法[16]进行一维整数 DCT 计算。

### 3.5.2基于数字信号处理器(DSP)的方案

数字信号处理器(DSP)是一类专用 CPU，通过采用哈佛结构，并且专门针对数字信号处理如 FFT、FIR 等进行了优化设计，使得 DSP 的处理能力远远高于通用 CPU。采用 DSP 进行音频、视频的压缩编码工作具有开发成本低，开发周期短等优点。

魏忠义等在 ADSP 开发板上实现了 JPEG 图像压缩编码算法[37]，其中的二维 DCT 部分采用行列分解法，一维 DCT 部分使用 DFT 系方法实现。

刘宝兰等采用美国德州仪器(TI)公司的 TMS320－DM642 开发板作为实现和开发的平台，并在其上进行算法的实现和优化[38]。文章中的二维 DCT 变换算法采用的是 H.264/AVC 参考设计 JM 的算法，并使用汇编语言进行了优化。

金燕波等的文章[39]采用以 TI 公司的高速 DSP 芯片 TMS320C6201 为核心的数字信号处理板作为图像压缩器的硬件平台，通过自行开发的压缩程序，实现了图像的实时压缩。文章中的二维 DCT 计算采用行列分解法，一维 DCT 计算部分采用了 TI 公司提供的库函数 fdct_8x8。

### 3.5.3基于 FPGA 的方案

FPGA(Field Programmable Gate Array)是一种设计灵活，开发速度很快的现场可编程逻辑器件。与 DSP 相比，FPGA 的主要优点有：①速度快，FPGA 有内置的高速乘法器和加法器。②DSP 内部一般没有大量的存储器，而目前高档的 FPGA 中有巨量的高速存储器，其速度更快，电路也更简单。③FPGA 是硬件可编程的，比 DSP 更加灵活，DSP 往往需要外部的接口和控制芯片配合工作，FPGA 则不需要，这样使得硬件更简单和小型化。

陈玲晶等的文章[40]简述了自行设计的基于 PCI 总线的 IP 测试验证平台，利用该平台的资源，采用 Sparn2EXC2S600E FPGA 硬件，实现并且验证了所设计的二维离散(逆)余弦变换 IP 核。文章中依然采用行列分解法将二维 DCT 计算分解为两重一维 DCT 计算，并采用 DA(Distribute Arithmetic)结构进行一维 DCT 计算。

钟文荣等的文章[41]介绍了一种基于行列变换快速算法的高速 DCT 处理芯片的设计，并详细阐述了实现这一算法的电路结构。

### 3.5.4基于 ASIC 方案

ASIC 是 Application Specific Integrated Circuit（专用集成电路）的缩写，它是按照特定用户的要求或特定电子系统的需要而设计、制造的集成电路。与 FPGA 相比，ASIC 具有较低的功耗、更小的芯片面积以及更快的运行速度。但是由于 ASIC 缺乏灵活性，研发成本较高，通常只在产品已定型并且需要大批量生产时才会采用 ASIC 技术。

单一功能的二维 DCT/IDCT 芯片或 FPGA 设计已经是上个世纪的事情了，随着集成电路工艺的不断改进，芯片集成度不断提高，目前国内已经有能力生产 1000 万门级别的芯片，而 Intel 公司的 Pentium 840EE 更是达到了 2.3 亿万个晶体管，最新的显卡芯片 G80 竟然集成 3.5 亿只晶体管。目前的 ASIC 方案已经可以将整个音、视频编码系统集成到一片芯片中 (SOC)，而将二维 DCT/IDCT 变换作为其中的一个功能模块。

MPEG-4 方面，国外已经有很多公司开发出了 ASIC 芯片，如 WIS 公司推出的 wisGO7007，飞利浦公司的 Trimedia 等，而国产芯片有汉唐科技的 Osoon、同济大学超大规模集成电路研究的神芯一号等。

H.264 方面，国外已经量产的 H.264 编码芯片有 Broadcom 公司的 BCM7411、Conexant 公司的 CX2418X、ST 公司的 STB7100 等，这些芯片都具有播放高清分辨率视频的能力，而且大部分都将支持标准中的 High Profile.；国内的产品有上海富瀚微电子的标清解码器芯片 FH8601 等

AVS 方面[42]，在 2005 年 11 月 30 日举行的国际计算机创新大会 CI6016 上，中科院计算所对外正式发布“凤芯二号”。凤芯 2 号完全兼容 AVS1.0、H.264/AVC main profile 标准，最高支持 HD(1920 x 1080i)格式的实时解码，内嵌龙芯 32bitCPU 主控解码全过程。同时为降低系统成本，凤芯 2 号集成了 32bit PCI2.1 系统总线接口，DDR SDRAM 控制器，FLASH 接口等，并实现了高效率的外部存储管理。未来 10 年内，AVS 产业化将为信息技术企业提供 3 亿～5 亿枚解码芯片，同时为企业节省 10 亿美元的专利费。

# 第 4 章 2-D DCT/IDCT 的 FPGA 设计

## 4.1 引言

DCT 是一种空间变换，在 MPEG-2 等图像压缩算法中 DCT 以 8×8 的像块为单位进行，生成的是 8×8 的 DCT 系数数据块。DCT 运算量极大，一个大小仅为 8×8 的图像子快，如果直接进行二维 DCT 需进行 4096 次乘法和 4032 次加法操作。在 JPEG 压缩过程中，DCT 变换所占用的时间大概占到总时间的 70%左右。在 MPEG 系列解码过程中，计算 DCT/IDCT 的时间约占总时间的 20%-30%。因而能否快速实现 DCT/IDCT 成为视频、图像压缩编码和解码过程中的一个关键性问题。

本文提出了两种实现二维 DCT/IDCT 的方案，两种方案都采用行列分解法将二维 DCT/IDCT 变换分解为两重的一维 DCT/IDCT 变换。对于一维 DCT/IDCT 变换，本文通过延长周期的方法对 Loeffler 算法进行了一定的改进，以缩短硬件实现 Loeffler 算法时流水线的执行时间，从而提高处理核的性能。通过采用两种不同的乘法器结构，本文提出的两种方案，方案一采用通用乘法器，硬件开销上比参考系统节约了大约 50%的资源；方案二针对 DCT 变换中乘法的特点采用优化了的专用乘法器，最高工作频率比参考系统提高了大约 10%，而硬件开销上仍然比参考系统节约了大约 10%。

## 4.2 DCT 的实现结构

大多数的二维 DCT/IDCT 处理核采用行列分解的结构实现。这是因为大多数的快速直接二维 DCT 算法往往结构很不规整，而且大量使用全局互连，因而不适合采用 FPGA 或者 ASIC 电路来实现。另外一个重要的原因在于基于现有的一维 DCT/IDCT 结构来实现二维 DCT/IDCT 比重新设计一个新的并且不兼容一维 DCT/IDCT 的二维 DCT/IDCT 结构更加有吸引力，因为这样可以节约大量的设计时间[43]。K.J.R Liu 等人的论文[44]中指出，通过时间递归实现的一维 DCT 结构具有很规则的数据流，模块化的结构，并且不需要全局互连。在此基础上采用行列分解的方式实现的二维 DCT/IDCT 计算核同样具有数据流规则，结构简单等特点，适合采用 FPGA、VLSI 实现。

图 5-1 是二维 DCT/IDCT 处理核的模块图。该处理核既可以进行二维 DCT 变换也可以进行二维 IDCT 变换。当该处理核进行二维 DCT 变换时，输入数据为 9bit 带符号数，输出数据为 12bit 带符号数；当该处理核进行二维 IDCT 变换时，输入数据为 12bit 带符号数，输出数据为 9bit 带符号数。这些数值是由图像压缩标准的输入输出数据决定的。

在 H.26X、MPEG 等标准中，变换前输入的象素值的范围为 0～255，因此，变换后直流(DC)系数的动态范围为 0～2040，任一交流(AC)系数的动态范围为－1000～1000。但是，对于 P 帧和 B 帧画面，象素表示的是差值，因此对于 P 帧和 B 帧象素值的范围为－255～255。因此，任何系数的最大动态范围为－2000～2000。由此可见，编码器用 12bit 即可表示任一系数，其值的范围为－2048～2047，相应的象素值用 9bit。

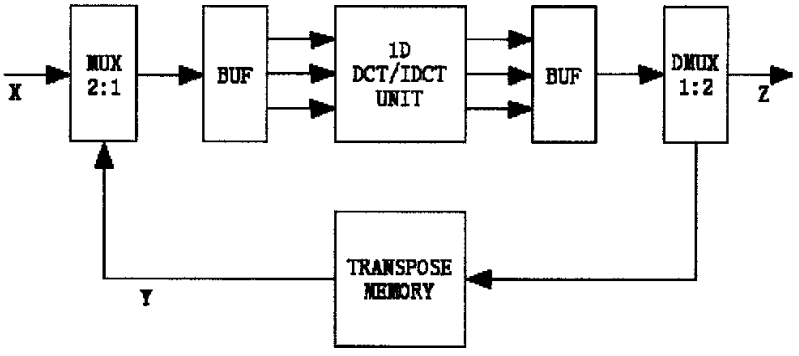



图 4-1 二维 DCT/IDCT 模块图

图 5-1 中的主要部分是一个一维的 DCT/IDCT 处理单元、一个转置内存模块，另外还有一个 2 选 1 多路选择器，一个 1：2 线数据分配器，2 个 8 字节的串－并转换缓冲区。

二维 DCT 被行列分解为两个一维 DCT。首先进行的是行变换，此时输入数据首先经过多路选择器进入第一个缓冲区，在这里等待第一行的 8 个数据全部到齐之后，并行的将数据发送到一维 DCT/IDCT 处理单元，经过 8 个时钟周期后，一维 DCT/IDCT 处理单元再将处理结果并行地发送到第二个缓冲区，然后经过数据分配器，逐个按行将数据写入转置内存。与此同时第一个缓冲区将第二行 8 个数据发送到一维 DCT/IDCT 处理单元，并开始第二行的一维 DCT 计算。

在全部 8 个行 DCT 变换计算完毕之后，开始列 DCT 变换。这时候转置内存模块中的数据按照列的顺序逐个读取出来，经过多路选择器进入第一个缓冲区，计算过程同上。在第一列的 DCT 计算完成之后，DCT/IDCT 处理核将在连续的 64 个时钟周期内将二维 DCT/IDCT 变换的结果(即图像块经过二维离散余弦变换后得到的系数)逐个的输出到模块外部。并在第 57 个系数输出的同时开始从外部输入端读取第二个图像块的数据。

## 4.3 转置 RAM

转置内存模块可以形象地看成是 8×8 的阵列。由一个 64 字节的双端口同步内存以及相关的控制电路组成。字节的宽度根据需要而定，本文采用字节宽度为 15bit。双端口内存的大小为 64×15＝960bit。内存中的前 8 个字节对应着阵列的第一行，第 9 到 16 个字节对应着阵列的第二行，依此类推。假设模式一是按照{1, 2, 3, 4, 5, 6, 7, 8, 9, …, 64}的顺序来存取数据，这对应着对阵列从左上角开始按行存

取，模式二按照{1, 9, 17, 25, 33, 41, 49, 57, 2, 10, 18, …, 64}的顺序存取数据，这对应着对阵列从左上角开始按列来存取。数据具体在转置内存中的写入和读取顺序如图 5-2 所示。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |

(a)模式一

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 9 | 17 | 25 | 33 | 41 | 49 | 57 |
| 2 | 10 | 18 | 26 | 34 | 42 | 50 | 58 |
| 3 | 11 | 19 | 27 | 35 | 43 | 51 | 59 |
| 4 | 12 | 20 | 28 | 36 | 44 | 52 | 60 |
| 5 | 13 | 21 | 29 | 37 | 45 | 53 | 61 |
| 6 | 14 | 22 | 30 | 38 | 46 | 54 | 62 |
| 7 | 15 | 23 | 31 | 39 | 47 | 55 | 63 |
| 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 |

(b)模式二

图 4-2 转置内存中数据的读/写顺序

数据按照如下的方式实现转置：首先，在第一个 64 个时钟周期内数据按照模式一存储，然后在第二个 64 个时钟周期内数据按照模式二读取，从而实现了对第一次存储的数据的转置操作。

## 4.4 一维 DCT/IDCT 单元

现在已经提出了很多的一维 DCT/IDCT 快速算法，以减少 DCT/IDCT 计算中的冗余部分。这些算法可以分为两类：近似算法和精确算法，Loeffler 算法在所有精确一维 DCT/IDCT 算法中是乘法次数最少的。本文的一维 DCT/IDCT 处理模块采用的是重新安排了时钟周期的改进 Loeffler 算法。

图 5-3 是一维 DCT/IDCT 单元的模块框图，该模块包括一个加法、乘法网络，10 字节的寄存器，一个 3bit 的状态机，以及相关的状态控制电路。每 8 个时钟周期计算 1 次的 8 点 DCT/IDCT。

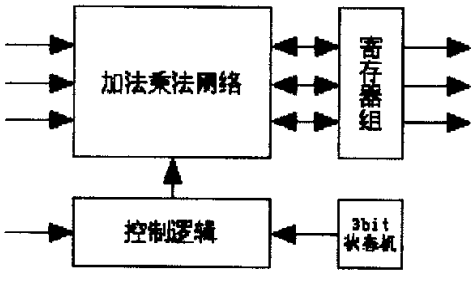



图 4-3 一维 DCT/IDCT 处理单元

根据方案的不同，加法乘法网络包含的硬件加法器和硬件乘法器的数量也会不同，

对于采用通用乘法器的方案一，加法乘法网络包含了 10 个加法器和 2 个通用乘法器；采用专用乘法器的方案二，加法乘法网络包含了 10 个加法器和 19 个专用乘法器。虽然两种方案的加法乘法网络内部结构完全不同，但是由于采用了良好的模块化设计，使得两种方案对外界表现出完全相同的功能和行为，因而外围电路完全相同。

该模块的输入数据宽度为 12bit，输出数据宽度也是 12bit，内部计算精度为 15bit，状态机可以沿两个方向进行状态转换：当模块进行 DCT 变换时，状态机沿{0,1,2,…,7,0}的方向旋转；当模块进行 IDCT 变换时，状态机沿{7,6,5,…,0,7}的方向旋转。寄存器组拥有 10 个 15bit 的寄存器，用于保存中间计算结果。

无论是二维 DCT/IDCT 变换还是一维 DCT/IDCT 变换，其变换因子都是浮点数。然而使用硬件实现浮点乘法的开销是很大的，速度也难以保证。为解决乘法中的小数问题，我们将每一个小数都乘以一个整数倍数(2 的 n 次方)，将所有的小数转换成整数进行处理，对输出结果再除以这个整数(采用移位操作完成或硬件直接丢弃多余的位)，以得到正确的结果。整数乘法运算的优点是运算速度快，硬件开销较小，但是精度略低于浮点乘法运算。

由 Loeffler 算法计算一维 DCT/IDCT 的输出结果比真实输出结果大$\sqrt{N}$倍，因此，如果不做特殊处理，最终的二维 DCT 输出结果就比真实的输出结果大了一个倍数 N。本文采取的解决办法是在行变换的数据进入一维 DCT/IDCT 模块之前，先将数据放大 N 倍。这样在行、列变换全部完成之后，得到的结果将比真实结果大 N×N 倍，由于 N=8，即结果放大了 64 倍，比真实结果多了 6bit。因此可以认为本文用于存储中间结果的寄存器的 15bit 数据中，最低 3bit 存储的是结果的小数部分，于是在一维 DCT/IDCT 模块输出结果之前，只需要简单的丢弃最后 3 个 bit 的结果即可。

该模块具有四种工作模式：

行 DCT 变换。此时输入的数据是图像或者图像差值，数据的范围是－255～255，因而输入数据只有 9bit 有效数据；输出数据的范围是－721～721，输出数据的有效数据为 11bit。

列 DCT 变换。此时输入的数据是行 DCT 变换后得到的 8 行共 64 个系数，数据的范围是－721～721；输出数据的范围是－2040～2040，输出数据的有效数据位为 12bit。

行 IDCT 变换。此时输入的数据是二维 DCT 变换后的系数，输出的结果是一维行 DCT 变换后的系数，输入数据的范围是－2040～2040，12bit 有效数据位，输出数据的范围是－721～721，11bit 有效数据位。

列 IDCT 变换。此时输入的数据是行 DCT 变换后的 8 行 64 个系数，输出的结果是 IDCT 变换后得到的图像数据或者图像差值数据。输入数据的范围是－721～721，11bit 有效数据位，输出数据的范围是－255～255，9 比特有效数据位。

## 4.5 时序分析与优化

为提高FPGA芯片的数据处理速度及降低芯片功耗，逻辑电路设计应重点采用以下措施[45]：

1. 采用流水线，降低芯片功耗，提高系统时钟。流水线是一种设计技巧，它在很长的组合逻辑路径中插入寄存器，寄存器虽增加了运算周期数，却能大大减少组合逻辑延时，提高整个系统工作频率。
2. 按面积优化组合逻辑，减小组合逻辑的复杂性，从而减少组合电路需要的逻辑门数量，逻辑门数的减少，意味着芯片功耗的降低。
3. 以原理图描述功能模块的数据流，以VHDL/Verilog语言的行为语句描述控制流。这种逻辑电路设计思想，充分利用原理图设计直观、形象和VHDL/Verilog输入法简单明了的优势，既可以获得具有高效率流水线结构的同步电路，又能够大大缩短设计时间。
4. 在电路设计过程中，应使用“自底向上”与“自顶向下”设计相结合、“逻辑设计”与“功能仿真”交替进行的设计技巧，以保证逻辑电路的层次化、模块化以及功能的正确性。

### 4.5.1流水线设计技术

流水线处理是高速设计中的一个常用设计手段。如果某个设计的处理流程分为若干步骤，而且整个数据处理是“单流向”的，即没有反馈或者迭代运算，前一个步骤的输出是下一个步骤的输入，则可以考虑采用流水线设计方法来提高系统的工作频率[46]。

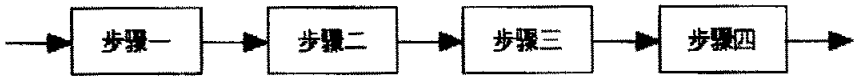



图 4-4 流水线的结构

流水线设计的结构示意图如图5-4所示。其基本结构为：将适当划分的 n 个操作步骤单流向串联起来。流水线操作的最大特点和要求是，数据流在各个步骤的处理从时间上看是连续的，如果将每个操作步骤简化假设为通过一个 D 触发器（就是用寄存器打一个节拍），那么流水线操作就类似一个移位寄存器组，数据流依次流经 D 触发器，完成每个步骤的操作。流水线设计时序如图 5-5 所示。

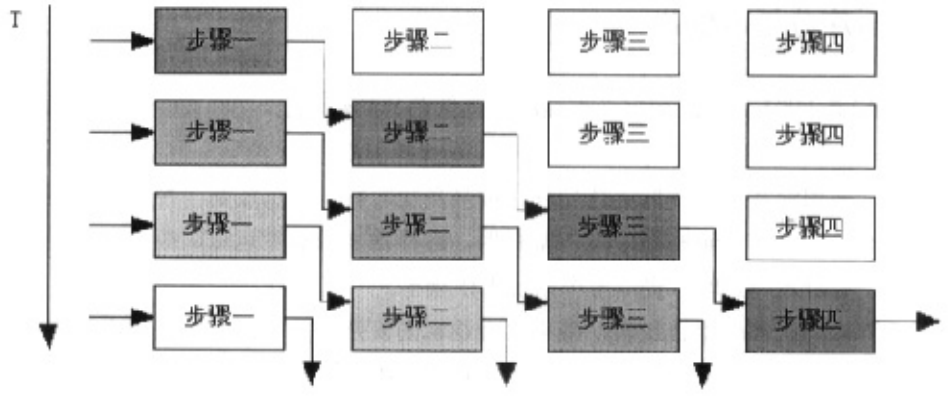
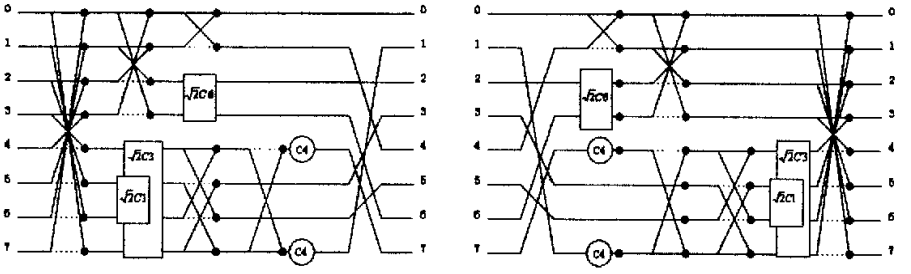



图 4-5 流水线设计时序

流水线设计的一个关键在于整个设计时序的合理安排，要求每个操作步骤的划分合理。如果前级操作时间恰好等于后级的操作时间，设计最为简单，前级的输出直接汇入后级的输入即可；如果前级操作时间大于后级的操作时间，则需要对前级的输出数据适当缓存才能汇入到后级输入端；如果前级操作时间恰好小于后级的操作时间，则必须通过复制逻辑，将数据流分流，或者在前级对数据采用存储、后处理方式，否则会造成后级数据溢出[47]。

## 4.5.2Loeffler 算法的时序

原始 Loeffler 算法如图 5-6 所示。




(a)DCT　　(b)IDCT

图 4-6 原始 Loeffler 算法

由于受到数据相关性的影响，用现有方法实现 Loeffler 算法时，一般需要用 8 个寄存器，并按顺序分 5 个时钟周期执行，算法的数据流如表 2 所示。IDCT 变换可以看作是将 DCT 变换的 5 个步骤按照反方向执行，只是在做蝶形计算时略有不同。

表 5-1 为 Loeffler 算法 DCT 变换数据流，表 5-2 为 Loeffler 算法 IDCT 变换的数据流。表中每一行表示一个寄存器，从 0 到 7 总共 8 个寄存器，每一列表示一个时钟周期，从 1 到 5 总共 5 个时钟周期，DCT 变换数据流的最后一列以及 IDCT 变换数据流的第一列

代表的是输出数据的位置或者输入数据的位置，因而不需要计算，也没有标注时钟周期。图标中的数字代表寄存器的位置编号，字母 a…g 代表 7 个常量因子，具体的数值见图 3-7。

例如第 1 行第 1 列的 0+7 表示将第 0 号寄存器加上第 7 号寄存器，然后将结果存入第 0 号寄存器；第 2 行第 5 列的(4+7)*d +(e-d)*7 表示将第 4 号寄存器加上第 7 号寄存器，结果乘以 $\cos(3\pi/16)$，第 7 号寄存器乘以 $\cos(5\pi/16)-\cos(3\pi/16)$，两者相加的结果存入第 5 号寄存器。

**表 4-1 Loeffler 算法 DCT 变换的数据流**

| step / pos | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 0 | 0+7 | 0+3 | 0+1 | | | 0 |
| 1 | 1+6 | 1+2 | 0-1 | | | 7 |
| 2 | 2+5 | 1-2 | (2+3)*f +(c-f)*3 | | | 2 |
| 3 | 3+4 | 0-3 | (2+3)*f -(c+f)*2 | | | 5 |
| 4 | 3-4 | (4+7)*d +(e-d)*7 | 4+6 | 7-4 | 4*c4 | 1 |
| 5 | 2-5 | (5+6)*b +(g-b)*6 | 7-5 | - | - | 6 |
| 6 | 1-6 | (5+6)*b -(g+b)*5 | 4-6 | - | - | 3 |
| 7 | 0-7 | (4+7)*d -(e+d)*4 | 7+5 | 7+4 | 7*c4 | 4 |

**表 4-2 Loeffler 算法 IDCT 变换的数据流**

| step / pos | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | - | 0+1 | | 0+3 | | 0+7 |
| 1 | 4 | 0-1 | | 1+2 | | 1+6 |
| 2 | - | | f*(2+3)-(c+f)*3 | 1-2 | | 2+5 |
| 3 | 6 | | f*(2+3)+(c-f)*2 | 0-3 | | 3+4 |
| 4 | 7 | 4*C4 | 7-4 | 4+6 | d*(4+7)-(e+d)*7 | 3-4 |
| 5 | 3 | | - | -7-5 | b*(5+6)-(g+b)*6 | 2-5 |
| 6 | 5 | | - | -4-6 | b*(5+6)+(g-b)*5 | 1-6 |
| 7 | 1 | 7*C4 | 7+4 | 7+5 | d*(4+7)+(e-d)*4 | 0-7 |

### 4.5.3优化的基础

原始的 Loeffler 算法各个时钟周期的计算量分布很不平衡。以 DCT 为例，第 1 和第 4 个时钟周期进行的是加法运算，第 2 和第 3 个时钟周期却分别要进行 2 次加法和 2 次乘法(注意，由于因子是常熟，所以因子之间的加减可以预先计算)。而且由于数据相关性，这些加法、乘法必须顺序执行。仍然以第 2 行第 5 列为例，第一步：计算 reg4+reg7；第二步：用 reg4+reg7 的结果乘以 d，同时计算出(e-d)*reg7；第三步：将第二步的两个结果相加，并存入 reg4。

我们设计流水线的时候应当合理划分各个步骤，使得各个步骤可以独立运行，并且尽量将运算均匀分布于各个步骤。因为流水线运行的速度取决于流水线中运行时间最长的那个步骤所需要的时间。由于原始的 Loeffler 算法各个步骤执行的时间差异很大，显然这种数据流是不利于流水线的快速工作的。

考虑到实际应用中的二维 DCT/IDCT 变换电路采用的是逐字节方式输入数据，输入一个 8×8 的图像子块需要 64 个时钟周期。对于电路内部的 1-D DCT/IDCT 模块来说，每 8 个时钟周期才能获得一行数据。因此，如果该模块在 5 个时钟周期内完成一行数据的 DCT/IDCT 变换，那么该模块在 8 个时钟周期内，每工作 5 个周期就需要等待 3 个周期。显然这种等待在一定程度上是一种资源的浪费。

### 4.5.4本文优化后的时序

因此本文考虑对 Loeffler 算法的时序进行调整，将 DCT 运算中的第 2 步、第 3 步以及 IDCT 运算中的第 2 步和第 4 步进行重新分配。通过重新分配减少单个步骤内部计算的相关性，缩短这些步骤所需要的运行时间，从而提高系统的工作频率，提高处理核的性能。

为此，我们对处理单元的结构进行了调整，增加了两个字节的寄存器，并将流水线的长度由 5 增加到了 8，从而使得系统有了调整的空间。计算量重新分配后的 Loeffler 算法的数据流如表 5-3 和表 5-4 所示。

表 4-3 改进的 Loeffler 算法 DCT 数据流

| step / pos | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0+7 | 0+3 | 0+1 | | | | | | 0 |
| 1 | 1+6 | 1+2 | 0-1 | | | | | | 7 |
| 2 | 2+5 | 1-2 | | | | | (c-f)*3 | 8+2 | 2 |
| 3 | 3+4 | 0-3 | | | | | (c+f)*2 | 8-3 | 5 |
| 4 | 3-4 | (e-d)*7 | | | 8+4 | 4+6 | 7-4 | 4*c4 | 1 |
| 5 | 2-5 | | | (g-b)*6 | 9+5 | 7-5 | | | 6 |
| 6 | 1-6 | | | (g+b)*5 | 9-6 | 4-6 | | | 3 |
| 7 | 0-7 | (e+d)*4 | | | 8-7 | 7+5 | 7+4 | 7*c4 | 4 |
| 8 | | 4+7 | 8*d | | 2+3 | 8*f | | | |
| 9 | | 5+6 | 9*b | | | | | | |

表 4-4 改进的 Loeffler 算法 IDCT 数据流

| step / pos | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | - | | | | | | 0+1 | 0+3 | 0+7 |
| 1 | 4 | | | | | | 0-1 | 1+2 | 1+6 |
| 2 | - | | (c+f)*3 | | 8-2 | | | 1-2 | 2+5 |
| 3 | 6 | | (c-f)*2 | | 8+3 | | | 0-3 | 3+4 |
| 4 | 7 | 4*C4 | 7-4 | 4+6 | (e+d)*7 | | | 8-4 | 3-4 |
| 5 | 3 | | | 7-5 | | (g+b)*6 | | 9-5 | 2-5 |
| 6 | 5 | | | 4-6 | | (g-b)*5 | | 9+6 | 1-6 |
| 7 | 1 | 7*C4 | 7+4 | 7+5 | (e-d)*4 | | | 8+7 | 0-7 |
| 8 | | | 2+3 | f*8 | 4+7 | | d*8 | | |
| 9 | | | | | | 5+6 | b*9 | | |

改进后的数据流每个时钟周期只需要进行一个加法或者一次乘法运算，因而流水线的运行频率可以大大提高，从而提高处理核的性能。同时由于流水线在每一个执行阶段最多只有两个乘法操作，因此可以考虑使用通用乘法器以节约硬件资源。为此本文提出

两种方案：方案一采用专用的乘法器件，以提高运行速度；方案二采用通用乘法器，以节约硬件资源。

## 4.6 有限字长仿真

对于给定一组数，由于采用了不同的 IDCT 结构而使得编码核解码系统中的输出出现差异，这种差异称之失配误差(Missmatch Error)。对于帧内编码来说，失配误差的影响不大，但是在帧间编码模式下，由于编码和解码过程构成了一个重构环，如果产生失配误差，会在这个重构环中累加，并作为附加噪声出现在重构环中。这样，在下一个帧内编码的帧(I 帧)到来之前，重构图像质量的下降将随着编码过程的持续而日益严重[47]。为了解决编/解码系统中由于 IDCT 的失配而造成图像质量下降，CCITT/ISO 联合委员会制定了关于 IDCT 有穷字长标准[48]。如果设计的 IDCT 运算精度符合这个标准，IDCT 的失配误差就不会导致图像质量的明显下降。

产生误差的主要原因有两个，即量化误差和计算误差。量化误差是在对系数矩阵进行量化的时候产生的，量化误差的大小跟随预期的图像质量不同而变化，而且是不可避免的(因为基于 DCT 的图像压缩算法都属于有损压缩)。计算误差则是因为我们在对数据进行计算的时候，采用的是有限字长的定点运算，因而必然会丢失部分计算精度，从而产生一定的误差。我们应当尽量减少计算误差。

### 4.6.1精度测量的过程

测量一个给定的 IDCT 的精度的过程描述如下：

1. 按照标准[48]附录中给出的随机数发生器(C 语言)产生范围在-L 和+H 之间的随机整数。将每 8 个数为一行构成一个 8×8 的数据块。产生 10000 个这样的数据块，每个块的范围为(L=256, H=255)，(L=H=5)和(L=H=300)；
2. 对每一个 8×8 的数据块进行 DCT，计算 DCT 时至少用 64 位的浮点精度；
3. 对每一个 8×8 的数据块变换后的系数取整为最相近的整数值。然后将这些值限定在-2048 到 2047 的范围内。将这些值作为 IDCT 变换的 12bit 输入值。
4. 对于第三步产生的 12 位的 8×8 的数据块进行 IDCT，至少采用 64 位的浮点计算精度。将结果取整为最相近的整数，并限定在-256 到 255 的范围内。这个 8×8 的数据值就是 IDCT 的参考输出。
5. 对第三步产生的 12 位的 8×8 数据块用被测试的 IDCT 芯片或者精确仿真的软件进行 IDCT 变换。将结果限定在-256 到 255 的范围内。这个 8×8 的数据块就是 IDCT 的测试输出；
6. 对以上产生的 10000 个数据块，计算每个 IDCT 数据块中的 64 个象素位置的参考值和测试值之间的峰值，均值和均方差；
7. 重新测试，使用的数据与第一步使用的数据相同，但是改变每一个象素值的符

号。

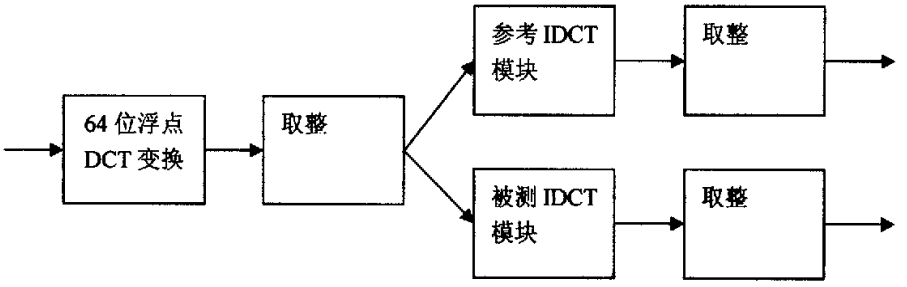



**图 4-7 精度测试的流程框图**

图 5-7 为精度测试的流程框图。以上的测试误差应该满足下一节说明的精度要求。

## 4.6.2精度要求

以上的测试误差应该满足下列精度要求：

1. 对任何一个象素位置，峰值误差(PPE)不能超过数量 1；
2. 对任何一个象素位置，均方差(PMSE)不能超过 0.06；
3. 全部的均方差(OMSE)不能超过 0.02；
4. 对任何一个象素位置，平均误差(PME)不能超过 0.015；
5. 全部的平均误差(OME)不能超过 0.0015；
6. 如果输入全为零，则要测试的 IDCT 的输出也应该全部为零。

上面提到的误差项的定义如下：设 $x_k(i,j)$ 为 IDCT 的参考输出，表示第 k 个数据块，所在的象素位置为(i, j)，$\hat{x}_k(i,j)$ 是 IDCT 的测试输出，其中 i, j=0, 1, …, 7，k=1, 2, …, 10000。

误差 $e_k(i,j)$ 是 IDCT 的测试输出和参考输出之间的差值，定义如下：

$$e_k(i,j) = \hat{x}_k(i,j) - x_k(i,j) \tag{5-1}$$

在象素位置(i, j)的峰值误差 PPE(i, j)定义为 $e_k(i,j)$ 的峰值，k=1, 2, …, 10000。

在位置(i, j)的均方差 PMSE(i, j)的定义为：

$$\text{PMSE(i, j)} = \frac{\sum_{k=1}^{10000} e_k^2(i,j)}{10000} \tag{5-2}$$

全部均方差 OMSE 的定义为：

$$\text{OMSE} = \frac{\sum_{i=0}^{7}\sum_{j=0}^{7}\sum_{k=1}^{10000} e_k^2(i,j)}{64\times 10000} \tag{5-3}$$

在位置(i, j)的平均误差 PME(i, j)定义为：

$$PME(i,j)=\frac{\sum_{k=1}^{10000}e_k(i,j)}{10000} \tag{5-4}$$

全局平均误差 OME 的定义为：

$$OME=\frac{\sum_{i=0}^{7}\sum_{j=0}^{7}\sum_{k=1}^{10000}e_k(i,j)}{64\times10000} \tag{5-5}$$

### 4.6.3高级语言仿真

在本文的设计中，有两个地方会产生误差：①系数量化，②有限字长。系数量化的误差是在系数因子(例如$\cos(3\pi/16)$)放大 2n 倍时产生的误差。由于在进行 IDCT 变换时，所有的系数都要与这些系数因子进行一次或者多次的乘法，因此这些误差会传递到最后的计算结果中，从而使结果产生误差。可以通过采用更多的位数表示来改善。内部计算的有限字长也会导致计算结果产生误差，改善这个误差的方法是内部计算时采用更长的字长。

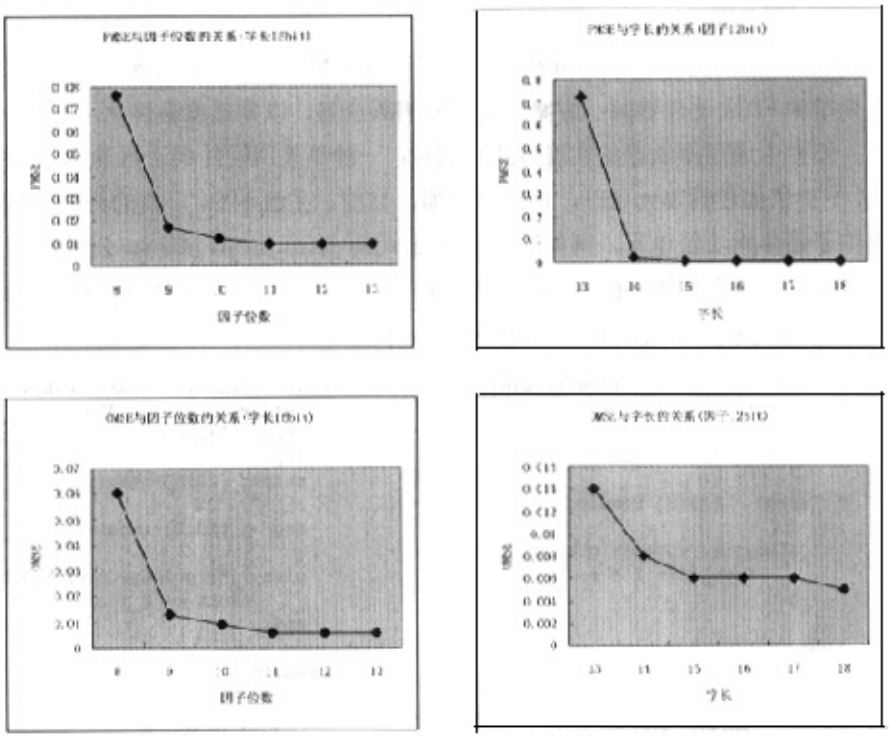


图 4-8 有限字长仿真实验结果

本文采用 C 语言进行了有限字长仿真，以确保实现的 IDCT 满足这个标准的要求。仿真实验的过程按照标准要求应当输入 10000 个 8×8 的数据块，由于仿真试验时存在溢出问题，本文采用 500 个 8×8 的数据块进行仿真实验。数据是由标准附录中的随机数发生器产生的。仿真实验结果如图 5-8 所示。

由于存在系数因子位数和计算字长两个因素影响计算精度，本文采取的方法是先固定其中一个因素(字长)，改变另一个因素(因子位数)获得一批实验结果，然后用同样的方法获得另外一批实验结果。CCITT 的 IDCT 误差标准要求有 5 项，这里仅用图例显示了其中的两项：最大峰值误差均方差(PMSE)以及全部误差均方差(OMSE)。

通过上面的实验可以看出，当字长 15 位、系数因子 12 位时，整数计算的精度能很好的满足标准的要求。因此本文的设计采用了 15 位的内部字长。实现的二维 DCT/IDCT 处理核的仿真实验结果如表 5-5 所示。

**表 4-5 仿真实验结果**

| | PPE<br>< 1 | PMSE<br>< 0.06 | OMSE<br>< 0.02 | PME<br>< 0.015 | OME<br>< 0.0015 |
|---|---|---|---|---|---|
| 因子 12bit，字长 15bit | 0.250 | 0.007063 | 0.006386 | 0.006500 | 0.000188 |

## 4.7 乘法器设计技术

本文提出的两种方案的区别就在于采用了不同类型的乘法器，方案一采用通用乘法器，以期节约 FPGA 硬件资源；方案二采用专用乘法器，以期速度最优化。

FPGA 器件中通用乘法器的实现方式有两种：一种是采用硬件乘法器单元，一种是使用 FPGA 中提供的逻辑单元(LE)，构建乘法器。目前，主流 FPGA 厂商的许多 FPGA 芯片中都提供了硬件乘法器单元，例如：Xilinx 公司的 Spartan-3，Atera 公司的 Stratix 系列以及 Cyclone 系列等。Spartan-3 具有高达 1 百万的系统门，包括 24 个硬件乘法器，而 Cyclone II则最多可包含 150 个硬件乘法器单元。

```
module dct2d( l, r, result, clk);
    input  [11:0] l, r;
    input  clk;
    output [23:0] result;

    reg    [23:0] result;

    always @( posedge clk) begin
        result <= l * r;
    end

endmodule
```

(a)通用乘法器

```
module dct2d( l, result, clk);
    parameter r = 12'd3456;
    input  [11:0] l;
    input  clk;
    output [23:0] result;

    reg    [23:0] result;

    always @( posedge clk) begin
        result <= l * r;
    end

endmodule
```

(b)专用乘法器

**图 4-9 乘法器实验的源码**

因此，当采用方案一时，如果选择包含硬件乘法器单元的FPGA芯片，可以大大节约LE资源。当然，如果选择的芯片不包含硬件乘法器单元(如MERCURY系列)，我们仍然可以通过逻辑单元构成乘法器，但是这种方案在资源利用和速度方面都不是很理想。

对于方案二，我们进行了如下实验：我们选择Altera公司的经典FPGA产品FLEX10K系列中的epf10k10lc84-3作为实验对象(因为它不包含硬件乘法器单元)。首先我们采用Verilog设计一个12×12位的通用乘法器，用Synplify综合后的报告显示该乘法器消耗了294个逻辑单元；然后我们将乘法器的乘数与被乘数之一改为一个常量(比如12'd3456)，然后再次用Synplify进行综合，综合后的报告显示现在这个乘法器仅仅消耗了28个逻辑单元。对比前面的通用乘法器，后者消耗的逻辑单元数仅为前者的1/10。实验中所用的源代码见图5-9。

可见，当乘数、被乘数之一为一个常数时，采用逻辑优化的方案设计专用乘法器比采用通用乘法器可以大大节约逻辑单元，同时由于电路复杂度降低，乘法器的性能也会得到提高。

由于在进行DCT、IDCT计算时，所有的乘法都具有上述特点，因此我们提出了第二个方案，即采用逻辑优化的专用乘法器代替通用乘法器，实现二维DCT/IDCT变换。

## 4.8 具体的实现过程

完整的FPGA设计过程包括设计输入、功能仿真、综合、时序仿真、器件编程、制板、验证等过程。因条件限制(时间、设备、成本)，本文完成了包括时序仿真在内的大部分工作，而没有进行最后的器件编程、制板、验证等工作。

### 4.8.1功能仿真

功能仿真也称为前仿真，是最基本的仿真验证。功能仿真是在设计布线和配置之前进行的，它只能仿真设计中的逻辑功能。通过功能仿真，可以验证整个系统的逻辑功能是否正确。用户可以通过观看仿真的波形来对系统的逻辑功能进行分析，并可以以此为依据，对设计进行必要的修改和完善。

对于系统级设计，我们通常首先使用高级语言比如C、SystemC对系统建模，进行功能仿真；然后使用VHDL/Verilog等语言进行模块设计，并使用ModelSIM等工具再次进行功能仿真。

本文所设计的处理核规模相对较小(约9-10万门)，一般情况下，对这种中小规模的设计可以直接使用ModelSIM进行功能仿真。但是由于笔者在进行有限字长实验时已经使用C语言编写了一部分功能仿真的代码(IDCT部分)，因此这里的功能仿真同样采用C语言完成。本文使用的编译环境为TurboC，并使用MPEG软件模拟工作组提供的浮点二维DCT/IDCT代码作为参照，对算法的正确性进行了验证。

## 4.8.2RTL 级设计

Verilog 有四种不同的层次用来描述一个模块的内部，以下简要概叙这四个层次：

1. 行为级或算法级描述( Behavioral or algorithmic level )，这是 Verilog HDL 中的最高层次，在这个层次中我们只需要考虑模块的功能或是函数，不需要考虑硬体方面详细的电路是如何实现。在这个层次的设计工作就好像写 C 语言一样。
2. 寄存器传输级描述(Register Transfer Level)，在这个层次中我们必须要指明数据处理的方式。设计的人要说明资料如何在寄存器中储存与传送，在设计里数据处理的方式。
3. 门级层次(Gate level )，在这层次中模块是由逻辑门连接而成，在这层次的设计工作就好像以前用门级原理图来设计线路一样。
4. 开关级层次( Switch level )，这是 Verilog 最低级的层次，线路是由开关（即晶体管）组成。在这层次设计的人必须知道晶体管的元件特性。

目前对行为级描述的综合还不是很成熟，而对于寄存器传输级(RTL)描述的综合目前已有很多成熟使用的商业软件，如 Synplicity 公司的 Synplify/Synplify Pro，Cadence 公司的 NC-Verilog 以及 Altera 公司专为自己公司产品设计的 QuartusII 等等。因此本文的处理核选择在寄存器传输级(RTL)设计、完成。

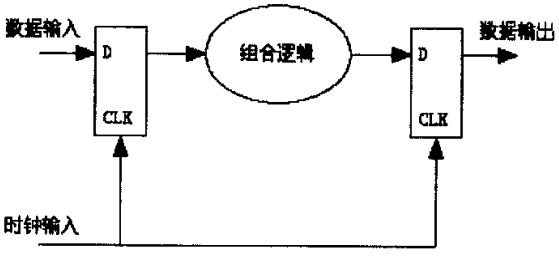



图 4-10 寄存器传输级设计模型

所谓寄存器传输级(RTL)设计是指在数字系统设计时，可以将数字系统简单的看作是由寄存器和寄存器之间的组合逻辑(Combinational Logic)所组成。对 RTL 的描述是以规定设计中采用各种寄存器形式为特征，然后在寄存器间插入组合逻辑(如图 5-10 所示)，这类寄存器或者公开地通过元件具体装配，或者作隐含的描述；组合逻辑或者由逻辑方程、顺序控制语句(CASE、IF THEN EHSE 等)和子程序描述，或者通过并行语句由寄存器之间的逻辑关系来表示。

例如图 5-11 为转置内存的 RTL 级 Verilog 描述，module 和 endmodule 语句定义了一个模块，模块的名字为 dctram；模块有 6 个端口，分别是 dout、din、waddr、raddr、clk 和 wr。这是一个双端口内存，可以同时对其进行读和写操作，waddr 是往转置内存中写入数据时使用的地址，din 是写入的数据；raddr 是从转置内存中读取数据时使用的地址，dout 是读取的数据，clk 是时钟信号，wr 是写使能信号。

```
module dctram(dout, din, waddr, raddr, clk, wr);
    parameter   WIDTH = 15;          // bit width
    parameter   ADDR_WIDTH = 6;      // bytes

    input       [ADDR_WIDTH-1:0]     waddr, raddr;
    input       [WIDTH-1:0]          din;
    input                            clk, wr;
    output      [WIDTH-1:0]          dout;
    reg         [WIDTH-1:0]          ram[(1<<ADDR_WIDTH)-1:0];

    always @( posedge clk) begin
        if( wr)
            ram[waddr] <= din;
    end

    assign dout = ram[raddr];
endmodule
```

图 4-11 转置内存的 RTL 级描述

always @( posedge clk) begin…end 表示总是在时钟的上升沿执行 begin…end 之间的语句。if( wr) ram[waddr] <= din;表示此时如果写使能信号为高电平，就在上升沿过后将 din 的数据写入到地址为 waddr 的内存中。

assign dout = ram[raddr];表示输出端口 dout 总是显示着地址为 raddr 的内存单元的内容。

经过综合之后，Synplify 会将这一段代码编译成一个预编译的内存单元，其 RTL 级视图如图 5-12 所示。

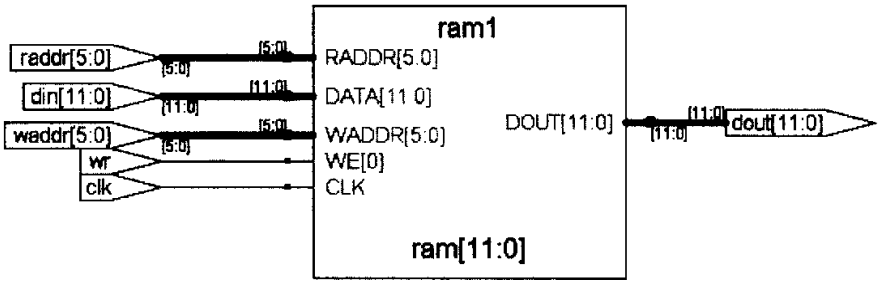



图 4-12 转置内存的 RTL 级视图

整个处理核由四个模块组成，他们分别是：顶层的处理核、转置内存、二维 DCT 计算的状态及逻辑控制和一维 DCT/IDCT 单元。处理核的顶层视图如图 5-13 所示。

处理核的输入信号包括：使能信号 en、时钟信号 clk、DCT/IDCT 选择信号 idct、以及 12 位的数据输入；输出信号包括输出数据开始信号 rdy、12 位的计算结果输出、以及用于监控目的的 8 位内部状态。

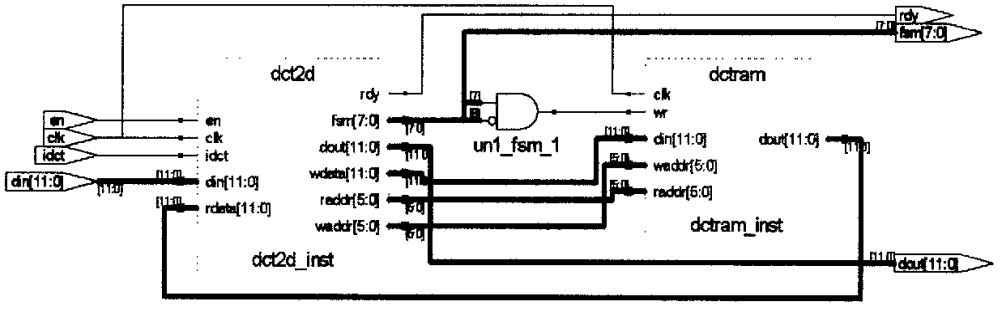



图 4-13 处理核的顶层视图

## 4.8.3综合

综合，就是针对给定的电路实现功能和实现此电路的约束条件，如速度、功耗、成本及电路类型等，通过计算机进行优化处理后，获得一个能满足上述要求的电路设计方案[49]。也就是是说，被综合的文件是 HDL 文件（或相应文件等），综合的依据是逻辑设计的描述和各种约束条件，综合的结果则是一个硬件电路的实现方案，该方案必须同时满足预期的功能和约束条件。对于综合来说，满足要求的方案可能有多个，综合器将产生一个最优的或接近最优的结果。因此，综合的过程也就是设计目标的优化过程，最后获得的结构与综合器的工作性能有关。

Synplicity 公司的 Synplify 综合器是针对高层次设计综合的需要而设计的，它主要特点是：代码的解释能力强，具有非常快的综合速度。Synplify 综合器的 PC 版本只需运行在普通的 PC 平台上，就能迅速完成数万门规模设计的综合；生成的门级模型合理，并支持先进的基本电路实现方案。在编写 HDL 语言代码时，对一些基本电路无须细致描述、综合器就会自动综合成相应电路；支持多种可编程逻辑器件，包括 Actel、Altera、Lattice、Lucent、Philips、Vantis、Quicklogic、Xilinx 等厂商生产的逻辑器件；能提供综合指令、属性和宏来控制综合的过程，以达到某些特定要求。Synplify 综合器主要是为 FPGA 和 CPLD 等可编程逻辑器件而设计的，但也可用于 ASIC 的研发工作。

本文的综合工作在 Synplify Pro 8.1 下完成，主要包括以下步骤：

1. 创建一个工程
2. 创建并编辑 Verilog 源代码文件，定义各个模块，设计模块的内部结构以及相互之间的连接关系。
3. 选择目标器件，为了和参考系统做对比，这里我们选择了 Altera 公司的 MERCURY 系列器件中的 EP1M120F484C5。该器件共有 4800 个逻辑单元（LE）、12 个嵌入式系统模块（ESB）、303 个用户可使用的 IO 引脚。
4. 设置各项参数，例如 Verilog 语言的版本，目标工作频率，是否生成.vqm 文件等。

5. 编译并综合，首先 Synplify 会采用 HDL 的语法规则对源文件进行分析并提示用户纠正语法错误；接着以选定的 FPGA 结构和器件为目标，对 HDL 和 FPGA 网表文件进行逻辑综合；最后根据用户的设计约束对速度和面积进行逻辑优化，产生一个优化的 FPGA 网表文件，以供 FPGA 布局和布线工具使用。
6. 查看综合结果日志以及 RTL 视图、技术视图，我们可以通过查看日志了解处理核的性能、时钟歪斜的程度、内部各个信号的延迟特性以及目标器件的资源使用情况。通过查看 RTL 视图我们可以知道 Synplify 生成的电路是否是我们想要的。

通过综合我们得到了网表文件，接下来我们将验证网表文件的正确性。

### 4.8.4时序仿真

**时序仿真的意义**

时序仿真是将设计适配到芯片后的仿真验证方式。时序仿真在严格的仿真时间模型下，模拟芯片的实际运作。仿真时间模型将最基本的门级时延计算在内，从而可有效地分析出设计中的竞争和冒险。经过时序仿真验证后的设计基本上与实际电路是一致的。

**仿真工具**

仿真使用的是 Altera 公司的 Quartus II 4.2 版。Quartus II是 Altera 新一代 FPGA/PLD 开发软件，针对 Altera 公司的新器件和大规模 FPGA 开发进行了特别优化。

**仿真结果**

首先我们针对双端口内存模块(转置内存)进行了时序仿真。将 Synplify 编译、综合后生成的 Verilog Quartus Mapping File 文件(文件后缀为.vqm)加入到 Quartus II的项目中，选择对应的器件型号之后进行编译、综合。在波形输入文件中加入测试输入，进行仿真。仿真结果如图 5-14 所示。

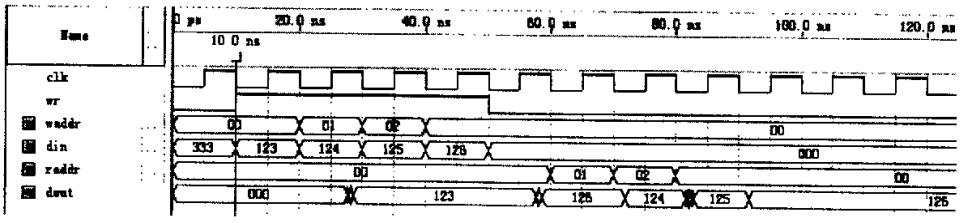



图 4-14 双端口内存仿真波形图

从波形图中可以看出：写数据在当前时钟的上升沿过后即写入到内存中，读数据则会在下一个时钟上升沿到来之前读取到当前的数据。例如：在第 1 个时钟上升沿写入一个数据，这个数据会在第 1 个时钟上升沿过后写入内存，在第 2 个时钟上升沿过后读到这个数据，在第 3 个时钟上升沿到来的时候，这个数据肯定已经从 dout 稳定输出了。

通过时序仿真实验，证明了双端口内存(转置内存)的正确性，同时也发现：①双端口内存的输出存在一定的毛刺；②使用不同的 FPGA 芯片，双端口内存会表现出不同的特性。例如使用 FLEX10 可能出现更多的毛刺；而使用 Stratix II 系列器件时，Quartus II 会将其编译为一个寄存器数组，并且时序关系也与 MERCURY 系列器件不一样。

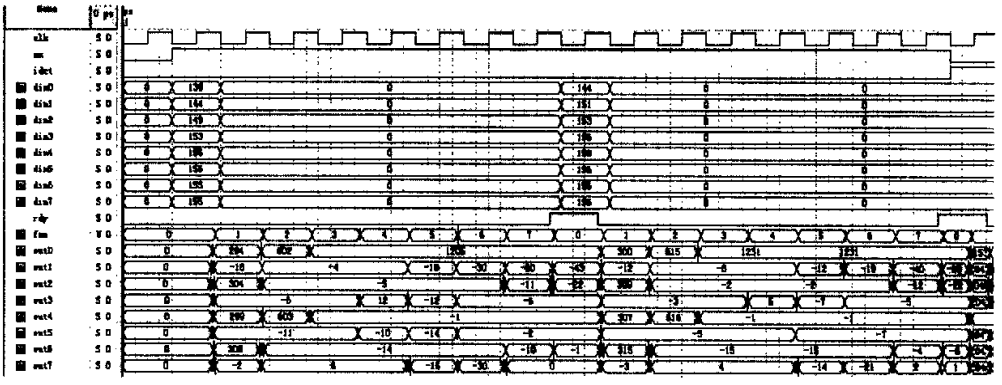


图 4-15 一维 DCT/IDCT 模块 DCT 仿真波形图

图 5-15 是一维 DCT/IDCT 模块时序仿真的结果。该模块在第 1 个时钟周期并行地输入 8 个字节的数据，然后进行计算并在第 9 个时钟周期并行地输出 8 个计算结果，与此同时并行地输入第 2 行 8 个字节的数据，并在第 17 个时钟周期并行地输出第 2 行的 8 个计算结果，以此类推。

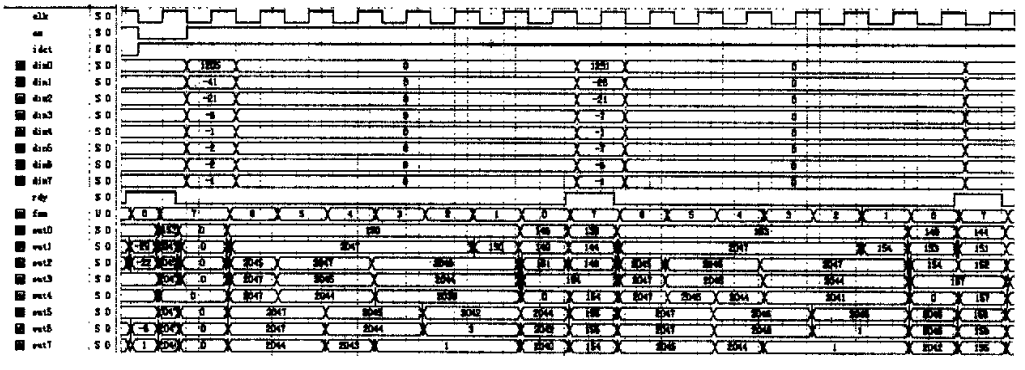


图 4-16 IDCT 时序仿真波形图

该模块进行 IDCT 计算的情况与 DCT 计算类似，所不同的是模块的输入信号 IDCT 应当为高电平，时序仿真结果如图 5-16 所示。

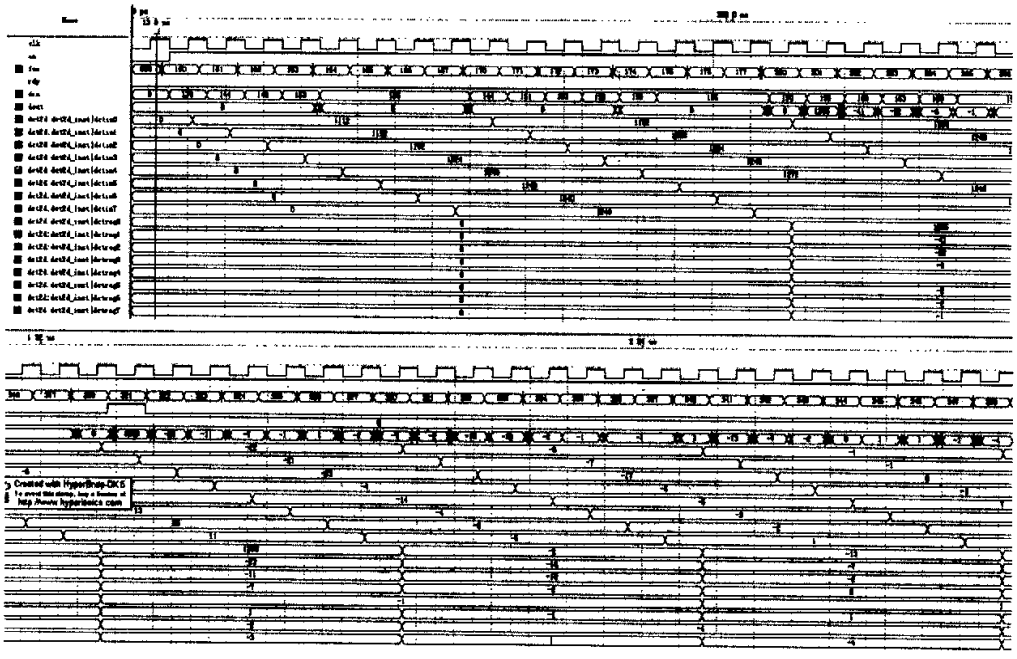


**图 4-17 处理核顶层时序仿真图(部分)**

图 5-17 为处理核进行二维 8×8DCT 变换的时序仿真结果的一部分，其中 fsm 表示处理核的内部状态，din 为输入数据，dctin0…dctin7 为内部一维 DCT/IDCT 模块的输入，dctreg0…dctreg7 为内部一维 DCT/IDCT 模块的输出，rdy 为输出有效指示，在 rdy 有效之前输出的是模块计算过程中的中间数据。从 en 变为 1 开始，97 个时钟周期后 rdy 输出 1，同时模块开始顺序输出第 1 到第 64 个结果。图中的 fsm 采用 8 进制表示，其他采用 10 进制表示。

处理核进行 8×8 二维 DCT 变换和 IDCT 变换所需的时间是相同的，都需要 161 个时钟周期。整个过程可以分为以下几个阶段：

1. 从模块外部读入第 1 行的 8 个数据(8 个时钟周期)。此时一维 DCT/IDCT 模块由于没有有效的输入数据，而对垃圾数据进行计算。
2. 从模块外部读入第 2--8 行数据，一维 DCT/IDCT 模块逐行进行 DCT/IDCT 计算，并将结果存入转置内存(64 个时钟周期)。
3. 将第 8 行 DCT/IDCT 计算结果逐个写入转置内存。此时一维 DCT/IDCT 模块再次因为没有有效的输入数据，而对垃圾数据进行计算(8 个时钟周期)。
4. 从转置内存中读取第 1 列的数据(8 个时钟周期)。
5. 从转置内存中读取第 2—8 列数据，同时一维 DCT/IDCT 模块逐列进行 DCT/IDCT 计算，并将计算结果输出到模块外部(64 个时钟周期)。
6. 输出最后一行计算结果(9 个时钟周期)。

```
44      always @( posedge clk ) begin
45          case ( fsm )
46              8'b00010000: begin
47                  fsm <= 8'b01110000;
48              end
49              default : begin
50                  fsm <= (en)? fsm + 1: 8'b01110000;
51              end
52          endcase
53      end
```

图 4-18 8bit 状态机的控制代码

整个计算过程由 1 个 8bit 的状态机控制，图 5-18 为状态机的控制代码，当使能信号 en 为低电平时，整个模块进入复位状态，此时状态机的计数为 0x70。从 en 信号变为 1 的那个时钟周期开始，模块连续读入 64 个输入数据，状态机的计数为 0xf0。从第一个输入数据开始，到第一行数据计算完毕准备写入转置内存，需要 16 个时钟周期，此时状态机的计数为 0x00，此时状态机的最低 6 个 bit 刚好构成转置内存的地址。当一维 DCT/IDCT 模块对 8 行数据计算完毕，即将从转置内存中按列读取中间数据进行列变换时，状态机计数为 0xc0，此时状态机的最低 6 个 bit 刚好又可以作为转置内存的地址。

## 4.9 DCT/IDCT 处理核的性能分析

### 4.9.1参照系统

本文选择了 Altera 公司网站公布的一个二维 DCT/IDCT 处理核[50]作为参照系统。该处理核由 Barco Silex 公司提供，具有以下特点：

1. 针对特定芯片(APEX 20KC)进行了优化设计。
2. 可进行定点的二维 DCT/IDCT 处理。
3. 采用行列分解法。
4. 兼容 JPEG、MPEG、H.263 等标准。
5. IDCT 变换符合 IEEE 1180-1990 标准。
6. 采用 9bit 图像数据，12bit 系数因子。
7. 深度优化，只消耗很少的逻辑单元(LE)。
8. 每个时钟周期处理一个数据
9. 能够持续地对图像块进行计算
10. 单时钟同步设计
11. 集成 64×22 的双端口同步 SRAM

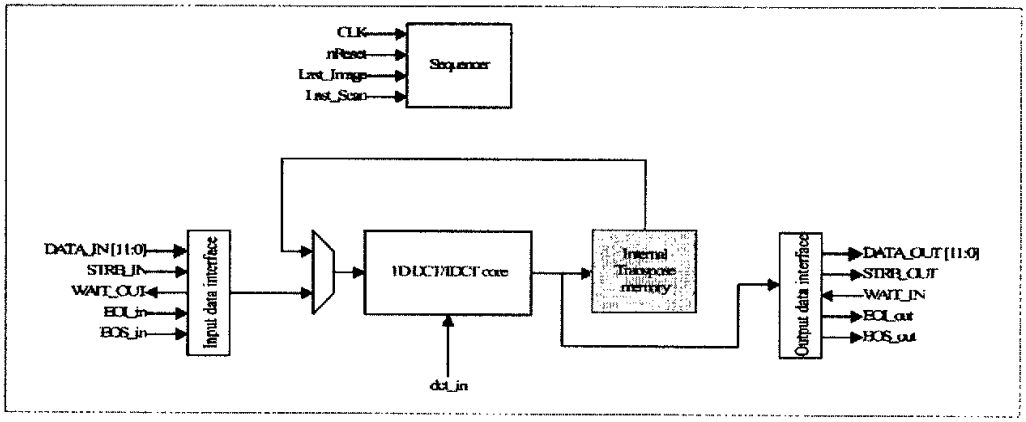



**图 4-19 参照系统的框图**

图 5-19 为参照系统的框图。不难发现，本文设计的系统与参照系统整体结构基本相同。根据 Altera 网站提供的资料，当使用 MERCURY 系列的 EP1M120F484C 芯片时，该 IP 核的最高运行频率为 103MHz，消耗 3203 个逻辑单元(LE)，2 个嵌入式系统块(ESB)。

MERCURY 系列 FPGA 器件并不是 Altera 公司目前主推的产品，无论集成度、速度，功耗还是性能价格比都不是最理想的。但是在本文选择的参照系统(Bidimensional DCT/IDCT)能够运行的 3 种目标 FPGA 器件系列中，参照系统在 MERCURY 上面的表现是最好的(运行频率最高并且使用的逻辑单元数也最少)。因此本文提出的两个方案同样针对 MERCURY 系列器件进行综合以及时序仿真。

## 4.9.2方案一

对于使用专用乘法器的方案一，Synplify 的综合结果显示：方案一的处理核最高运行频率为 116.5MHz，消耗 2827 个逻辑单元以及 2 个嵌入式系统块。最大时钟歪斜 0.725，最大扇出 48。

方案一的时钟频率为参考系统的 113%，逻辑单元消耗占参考系统的 88%。

## 4.9.3方案二

**表 4-6 性能对比**

| Target Device | Speed Grade | Utilization | | | Performance ($f_{MAX}$) | Parameter Setting |
|---|---|---|---|---|---|---|
| | | Logic Cells | ESBs | I/O Pins | | |
| EP1M120F484C | -5 | 1629 | 2 | 36 | 70.4 MHz | 方案一 |
| EP1M120F484C | -5 | 2827 | 2 | 36 | 116.5 MHz | 方案二 |
| EP1M120F484C | -5 | 3203 | 2 | 37 | 103 MHz | Barco Silex |

对于使用通用乘法器的方案二，Synplify 的综合结果显示：方案二的处理核最高运

行频率为 70.4MHz，消耗 1629 个逻辑单元以及 2 个嵌入式系统块。最大时钟歪斜 -4.197ns，最大扇出 282。

方案二的时钟频率为参考系统的 68%，逻辑单元消耗占参考系统的 50%。

本文的方案与参考系统性能对比见表 5-6。

## 4.10 结论

本章详细阐述了本文实现的二维 DCT/IDCT 处理核，采用高级语言(C 语言)进行了功能仿真，并使用 Quartus II 进行了时序仿真。二维 DCT/IDCT 采用行列分解法实现，减少了硬件实现的复杂度。在进行一维 DCT/IDCT 计算时，本文通过对 Loeffler 算法数据流的改进，降低了一维 DCT/IDCT 计算的复杂度，提高了流水线的运行效率。Synplify Pro 综合结果表明，对比参照系统，本文提出的方案一节约了一半的逻辑单元，工作频率则大于参照系统的一半。本文提出的方案二仅使用了参照系统 88%的资源，工作频率却比参照系统高 13%。

# 工 作 总 结 及 展 望

## 本文总结

在有关图像压缩的众多正交变换中，离散余弦变换（DCT）是十分有效的一种，其性能极为接近最佳的 K-L 变换。通过采用行列分解方法，人们只需使用少量的乘法就能完成二维离散余弦变换/反离散余弦变换（2-D DCT/IDCT）。正因为如此，DCT 变换被 JPEG、H.261、H.263、H.264、MPEG-1、MPEG-2、MPEG-4 等众多标准广泛采用。

本文通过对 DCT 快速算法及其 FPGA 设计的研究，对 Loeffler 算法的数据流提出了一种改进方案，并且根据该方案设计实现了两种不同的二维 DCT 处理核，从而证明了这种数据流改进方案的可行性。本文的工作包括以下几个方面：

1. 研究了基于 DCT 的图像压缩编码标准，就各个标准的特点以及他们相互之间的关系进行讨论，并着重研究了 MPEG-2 编码的流程。
2. 研究了近年来提出的各种余弦变换的快速算法，并着重讨论了 Loeffler 算法的数据流，并提出了一种对其数据流进行改进的方案。研究了二维 DCT 变换的物理意义及快速算法。
3. 对比分析了各种二维 DCT 实现方案的优、缺点。并对采用 FPGA 设计中的一些关键问题进行了研究。
4. 采用本文提出的数据流优化方案，并使用不同种类的乘法器，设计实现了两个二维 DCT/IDCT 处理核。进行了严格的内部字长仿真，使精度满足 CCITT 标准要求。

## 进一步的研究方向

虽然本文研究了 DCT 快速算法及其 FPGA 实现结构，采用一种新的方法设计了二维 DCT/IDCT 处理核，并取得了一些成果，但是由于时间及条件的限制，本文的工作还只是初步的，在以下几个方面还值得进一步的研究：

本文虽然通过数据流优化一定程度上提高了流水线的运行速度，但是处理核中的一维 DCT/IDCT 部分的乘加网络比较复杂，延时较大，还有一定的优化空间。

本文提出的方案一采用的 DCT 实现方法是基于通用乘法器的，而本文是通过逻辑单元生成的通用乘法器，速度和资源使用上都不是十分理想，针对目前许多 FPGA 芯片都提供了硬件乘法器单元的情况，可以考虑采用 FPGA 芯片内部的硬件乘法器单元，从而提高处理核的运行速度，并可节约逻辑单元。

本文实现的二维 DCT/IDCT 处理核处理一个 8×8 的数据块需要 161 个时钟周期，因此上层电路将数据发送给处理核的时候，需要设置暂停机制，不利于数据流的连续处理。可以考虑通过增加 1 个一维 DCT/IDCT 处理单元，使处理核达到 1 象素/秒的处理速度，以利于整个视频编、解码器的流畅工作。

# 参 考 文 献

[1] 顾卫民. 央视弃用国产标准AVS痛失市场先机. 中国高新技术产业导报, 2005-11-11

[2] 张伟民. AVS产业化策略与进展. http://www.avsa.org.cn/NewsCenter/upLoalPic/20050722095709970.pdf, 2005-7-1

[3] 高宗敏. 数字电视与有线电视宽带综合网. http://www.btc.sh.cn/wsxy/digi/d4z.htm, 1998-3-1

[4] Shannon CE. A Mathematical theory of communications. Bell Syst. Tech., 1948, 27:398-403

[5] Huffman DA. A method for the construction of minimum redundancy codes. In Proceedings IRE, 1962, 1098-1101

[6] R. Forchheimer, O. Fahlander and T. Kronander. A semantic approach to the transmission of Face Images. Proceedings of the Picture Coding Symposium. 1984

[7] Hortsch M, Avossa D and Meyer DI. The Journal of Cell Biology, 1985, 241-253

[8] G.K. Wallace. The JPEG Still Picture Compression Standard. Commun. ACM, 1991, 30-44

[9] D.L. Gall, MPEG: A Video Compression Standard for Multimedia Applications, Comm. of the ACM, 1991, 34(4):46-58

[10] R. Koenen. Overview of the MPEG-4 standard. 1997

[11] Rijkse K. H.263: video coding for low-bit-rate communication. Communications Magazine, IEEE, 1997, 34(12):42-45

[12] T. Wiegand, GJ Sullivan and G Bjøntegaard et al. Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Tech., 2003, 13(7):560-576

[13] 佚名. MPEG 基础和协议分析指南. http://down.upsdn.net/mpeg/MPEG.pdf 2004-1-1

[14] Mitsuo Ikedat, Tsuneo Okubot, Tetsuya Abet, et al. A Hardware/Software Concurrent Design for a Real-Time SP@ML MPEG-2 Video-Encoder Chip Set. European Design and Test Conference (ED&TC). 1996, 320-326

[15] 郭斌. MPEG-2压缩编码技术原理应用. http://info.broadcast.hc360.com/html/001/002/008/52120.htm 2004-4-1

[16] C. Loeffler, A. Ligtenberg, and G. Moschytz. Practical fast 1-D DCT algorithms with 11 multiplications. Proceedings of IEEE International

Conference Acoustics, Speech and Signal Processing, 1989, 988-991

[17] N. Ahmed, T. Natarajan, and K. R. Rao, Discrete cosine transform. IEEE Transaction on Computer, 1974, 90-93

[18] A.S. Spanias, S.B. Jonsson, S.D. Stearns. Transform methods for seismic data compression. IEEE Transaction on Geoscience and Remote Sensing, 1991, 29(3), 407-415

[19] Markus Püschel. Cooley-Tukey FFT like Algorithms for the DCT. Proc. ICASSP, 2003, 2:501-504

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

[21] Zhongde Wang. Fast Algorithms for the Discrete W-Transform and for the Discrete Fourier Transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984, 32(4):803-816

[22] Byeong Lee. A New Algorithm to Compute the Discrete Cosine Transform. IEEE Transactions on Acoustics. Speech and Signal Processing, 1984, 32(6):1243-1245

[23] M. Vetterli, H. Nussbaumer. Simple FFT and DCT Algorithms with Reduced Number of Operations. Signal Processing (North Holland), 1984, Vol 6: pp 267-278.

[24] N. Suehiro and M. Aatori. Fast Algorithms for the DFT and other Sinusoidal Transforms. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1986, 34(3):642-644

[25] H.S. Hou, A Fast Recursive Algorithm for Computing the Discrete Cosine Transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1987, 35(10):1455-1461

[26] P. Duhamel and H. H. Mida. New 2 DCT Algorithms suitable for VLSI Implementation. Proceedings IEEE International conference on Acoustics, Speech and Signal Processing, 1987, 12:1805-1808

[27] Liang Jie. Fast multiplierless approximations of the DCT with the lifting scheme. IEEE Transactions on Signal Processing, 2001, 49(12):3032-3044

[28] 张海亮. 浅议电视信号的数字化与码率压缩. http://www.lunw.com/thesis/39/7043_1.html, 1999-2-1

[29] Hassan EL-Banna, Alaa A. EL-Fattah. An Efficient Implementation of the 1D-DCT using FPGA Technology. ECBS 04, 2004, 356-360

[30] Y Arai, T Agui, M Nakajima. A fast DCT-SQ scheme for images. The Transaction of The IEIGE, 1988, 71(11):1095-1097

[31] Ephraim Feig. A fast scaled DCT algorithm. Proceedings of SPIE, 1990,

1244:2-13
[32] Lizhi Cheng, Yonghong Zeng. New polynomial transform algorithm for multidimensional type-IV DCTs. IEEE Trans Signal Process, 2000, 48(10): 2814-2821
[33] J. Prado, P. Duhamel. A polynomial transform based computation of the 2-D DCT with minimum multiplicative complexity. ICASSP, 1996, Vol 3: pp 1347-1350
[34] P. Duhamel, C. Guillemot. Polynomial transform computation of the 2-D DCT. Acoustics, Speech, and Signal Processing, 1990, Vol 3: pp 1515-1518
[35] http://ffmpeg.sourceforge.net/index.php, 2005-10-1
[36] http://www.xvid.org/, 2005-11-1
[37] 魏忠义, 朱磊. 基于DSP的JPEG图像解码算法的实现. 现代电子技术. 2005, Vol 2: pp 66-68
[38] 刘宝兰, 刘贵忠, 苏睿. H.264中整数DCT变换及量化的DSP实现. 微电子学与计算机, 2005, Vol 6: pp 200-205
[39] 金燕波, 朗锐, 罗发根等. 利用TMS320C6201 DSP芯片进行图像压缩. 电子技术应用, 2004, Vol 1: pp 63-66
[40] 陈玲晶, 郑学仁, 范健民等. 二维DCT/IDCT的FPGA实现及验证方法. 集成电路应用, 2005, Vol 1: pp 49-53
[41] 钟文荣, 陈建发. 二维DCT算法的高速芯片设计. 厦门大学学报:自然科学版, 2005, Vol 2: pp 198-201
[42] http://www.avs.org.cn/, 2005-08-01
[43] C.T. Chiu, K.J.R.Liu. Real-Time Parallel and Fully-Pipelined Two-Dimen sional DCT Lattice Structures with Application to HDTV Systems. IEEE Transa ctions On Signal Processing, 1992, Vol 2: pp 25-37
[44] K.J.R. Liu, C.T. Chiu. Unified Parallel Lattice Structures for Time-Re cursive Discrete Cosine/Sine/Hartley Transforms. IEEE Transactions On Signa l Processing, 1993, Vol 41: pp 1357-1377
[45] 程子敬, 姜宏旭. 高速低功耗FPGA的应用设计. 电子产品世界, 2001, Vol 8: pp 37-38
[46] 王诚. FPGA设计的四种常用思想与技巧. http://www.ieechina.com/Upload/Tec h/200501141109440359.pdf, 2005-11-1
[47] 傅宇卓, 王嘉芳, 胡铭曾. 基于行列变换结构的2-DCT/IDCT的误差分析. 小型微型计算机系统, 2004, Vol 7: pp 1286-1289
[48] IEEE Std 1180-1990, IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform, 1990.
[49] 孙富明, 李笑盈. 基于多种EDA工具的FPGA设计. 电子技术应用, 2002, Vol 1:

pp 70-73

[50] Barco Silex. Bidimensional DCT/IDCT. http://www.altera.com.cn/products/ip/dsp/transforms/m-bar-bidimensional.html, 2005-11-1

# 致 谢

导师邝继顺治学严谨，学识渊博，品德高尚，平易近人，在我学习期间不仅传授了做学问的秘诀，还传授了做人的准则。这些都将使我终生受益。无论是在理论学习阶段，还是在论文的选题、资料查询、开题、研究和撰写的每一个环节，无不得到导师的悉心指导和帮助。我愿借此机会向导师表示衷心的感谢！

回顾三年学习期间的近一千个日日夜夜，自己为能重返学校，静心钻研，潜心研究，并取得初步研究成果而感到欣慰。欣慰之余，我要向关心和支持我学习的所有老师和朋友们表示真挚的谢意！感谢他们对我的关心、关注和支持！

在即将毕业离校之际，我要感谢舍友傅红普、杨帆、唐步尧、刘善平、蒋沛航、王慧文、陈宇等在生活上给予我的关心和帮助以及学业上的切磋和指点，感谢计算机通信学院2003级全体同学的帮助和勉励。同窗之谊和手足之情，我将终生难忘！

路漫漫其修远兮，吾将上下而求索。我愿在未来的学习和工作过程中，以更加丰厚的成果来答谢曾经关心、帮助和支持过我的所有领导、老师、同学和朋友。此外，我要感谢我们实验室的所有成员，感谢我们寝室的全体室友，感谢你们在平时的学习工作中给予的关心和帮助。

最后感谢我们的父母和姐姐、姐夫。感谢父母对我养育和栽培之情，感谢姐姐、姐夫对我的不断鼓励，感谢姐姐对我的无私关心和疼爱，感谢我的女友雷莉给予我的关怀与鼓励。谢谢你们。

感谢各位专家百忙之中对本文的审阅和提出的宝贵意见！

2005年12月