学术论文撰写之实例介绍

杜伟韬 中国传媒大学 数字化工程中心

本文在一篇学术论文的基础上，介绍如何总结科研成果，编写学术论文。实例论文“一种基 2 DCT 后加运算改进方法”发表于期刊《数据采集与处理》2009S1，该文由数字化中心硕士生于成龙同学的算法创新成果而来，杜伟韬系该生辅导老师，该论文的使用已经得到于成龙同学的授权。本文用于和选修本人承担的“数字信号处理集成电路设计”硕士生课程的同学交流经验，转载请注意出处。下文中红色的字体是关于撰写学术论文的实例介绍。请结合论文原文阅读。

关于创新，创新是发表论文的基本要求，对创新我们可以理解如下，创新不是创万能，评价一个电路或算法会有很多的方面，我们不需要把所有方面都做到完美再拿出去投稿，很多时候，创新就像是在转跷跷板，比如 FFT 算法对于 DFT 算法而言是重大的创新，但是相比 DFT 算法，FFT 算法仍然存在着以下方面的短处，1 标号寻址比较复杂 2 需要把一个序列全收集完才能开始计算 3 不能只计算一个频域样点，必需计算全部频域样点。所以，没有最完美的，只有最合适的。只要你做的东西有某些方面超越了前人的成果那么他就有发表的意义，因为也许你的成果恰好符合某些人的需要。

一种基 2 DCT 后加运算改进方法

于成龙 杜伟韬 曾志斌

(中国传媒大学广播电视数字化教育部工程研究中心，北京，100024)

摘　要：本文在基 2DCT算法基础上，提出一种改进的后加运算策略，该策略能够有效利用循环展开技术生成可灵活调节代码尺寸、缓存开销及读写并发性的算法程序以适应不同的运算平台结构，并且针对该算法设计并实现了参数可配置的快速算法代码生成器。

摘要是对你工作意义的简要陈述，这里主要说明你做出了怎样的新工作，这种工作的意义何在，摘要要简介，不要详细阐述底层技术细节，着重说明有什么好处。吸引审稿人的兴趣，让他看下去。

关键词：后加运算　DCT　蝶型 代码生成器 循环展开

Improved Method to Compute the Post Addition in Radix-2 DCT

ChengLong Yu, WeiTao Du, ZhiBin Zeng

(Digital Engineering Center, Communication University of China, Beijing, 100024, China)

Abstract: This paper proposes an improved post-adding method based on the radix-2 DCT algorithm. By associating the proposed method with loop-unrolling technique, the code size, data buffer and concurrency of the target code can be flexibly balanced to match various processors' architecture. A configurable algorithm code generator is implemented to improve the flexibility and reusability of the algorithm.

Key words: post addition, DCT, butterfly, code generator, loop unrolling

基金项目： CNGI-04-12-2A；教育部重点项目（106041）中国传媒大学 211 工程

引言

引言分为两部分，1 是描述你研究领域内的发展情况和当前的形式，并且指出前人工作的局限性，注意是局限性而不是否定前人的工作，这些局限性正是你研究成果的改进点。2 是相对于摘要而言稍微详细一点的阐述你做出的成果的效果以及意义和价值。

离散余弦变换（DCT）因其具有良好的能量汇聚特性而被诸如JPEG、MPEG、H.263 等多个国际编码标准所采用。自Ahmed 1974 年提出DCT的变换[1]后，针对寻找其高效计算方法开展了一系列研究。1977 年，W. H. Chen等人根据变换矩阵的对称性，首次采用稀疏矩阵分解得到DCT-II的仅需较少的乘加次数的快速算法[2]。B. G. Lee[3]提出使用余割因子的DCT矩阵分解算法，因其具有类Cooley-Tukey式的简单结构而受到广泛关注，但存在余割因子引入的计算精度问题。C. M. Liu[4] 提出一种通用的DCT快速算法，该算法可以在四类DCT之间进行递归分解。C. W. Kok[5]提出一种对偶数点DCT进行递归分解的快速算法，具有公式推导简单，运算复杂度低的特点。以上就是描述研究领域内的来龙去脉，并且要拜一下祖师爷们，这里需要你首先阅读一定的参考文献，这样才能避免重复发明别人的成果。

现代处理器的计算性能很大程度上受制于缓存命中率[6]和流水线填充程度。因此，除乘加运算总量之外，在资源受限的嵌入式处理器上实现DCT时，算法的标号/寻址机制和代码尺寸对于实现性能同样具有重要作用。文献[1]-[5]提出的算法能够有效减少计算的乘加数量，但其标号/寻址机制相对复杂，仅适用于在小点数DCT情况下将递归运算完全展开的“硬编码”的实现方式。以上是适当的指出已有成果的局限性，为展示自己的成果做铺垫，写这一段的时候一定要谨慎，避免因浮夸而被砍。

本文在基 2 蝶形DCT基础上[5]，提出一种新的后加运算策略，在使用相同的加法数量下，该策略能以更为简洁有序的标号/寻址机制对DCT蝶型的输出数据进行后加运算。该算法结合循环展开技术可以使不同点数的DCT算法在代码尺寸、缓存开销及读写并发性之间进行灵活调节，从而能够调整目标程序使其在各种多发射结构及不同片上缓存配置的处理器上达到最优性能。在算法实现过程中，为提高易用性及代码重用效率，本文以代码生成器的方式实现了基于该算法的可配置DCT点数及循环展开层数的目标算法生成软件。

以上是对自己成果的陈述，可以稍微详细的陈述一下创新的地方和创新的，这里是你摘要内容的一个加强版。

1 基 2 蝶形运算的 DCT 快速算法

下面要开始描述你自己创新的内容了，这里最好要有一定数量的公式，这样显得论文逻辑性严密，增加命中率。

首先要告诉读者，你站在谁的肩膀上，任何创新都是在已有的基础上来的，所以，你要告诉读者，你的创新是在哪个基础上的，这是你工作的起点或是背景。注意，关于别人的工作，一定要写的越简洁越好，能引用参考文献就引用参考文献，能直接引用结论就直接写结论，不要啰嗦的占版面，编辑最痛恨用别人已有成果占版面充数的论文。

对于 N 点一维序列 $\{x(n);n=0,1\cdots N-1\}$，其 DCT 定义为

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

$$k=0,1\cdots,N-1 \quad (1)$$

式中 $\alpha(k)$ 为正交化因子，为讨论方便可省略。

一个 N 点 DCT 变换可转换两个 $N/2$ 点 DCT，即

$$X(k)=C(i)+D(i)$$

$$k\in[0,N-1],i\in[0,\frac{N}{2}-1] \quad (2)$$

$$C(i)=X(2i) \qquad i\in[0,\frac{N}{2}-1] \quad (3)$$

$$D(i)=X(2i+1) \qquad i\in[0,\frac{N}{2}-1] \quad (4)$$

设

$$p(n)=x(n)+x(N-1-n)$$

$$n\in[0,\frac{N}{2}-1] \quad (5)$$

则

$$C(i)=\sum_{n=0}^{\frac{N}{2}-1}p(n)\cdot\cos[\frac{2\pi}{4N}(2n+1)2i]$$

$$i\in[0,\frac{N}{2}-1] \qquad (6)$$

表达式(4)是一个 $N/2$ 点 DCT 运算。下面对 $D(i)$进行变换

设

$$V(i)=D(i)+D(i-1) \quad i\in[0,\frac{N}{2}-1] \quad (7)$$

$$q(n)=(x(n)-x(N-n-1))2\cos[\frac{2\pi}{4N}(2n+1)]$$

$$n\in[0,\frac{N}{2}-1] \qquad (8)$$

可得

$$V(i)=\sum_{n=0}^{\frac{N}{2}-1}q(n)\cdot\cos[\frac{2\pi}{4N}(2n+1)2i] \quad (9)$$

表达式（9）是一个 N/2 点 DCT 运算。

由 $D(i)$的对称性可得

$$D(0)=D(-1) \qquad (10)$$

以此类推继续分解，可将 N 点 DCT 转换成 $N/2$ 组 2 点 DCT 进行计算。该算法流程如图 1 所示，可理解为由蝶形分解、DCT 运算和后加运算三部分组成。

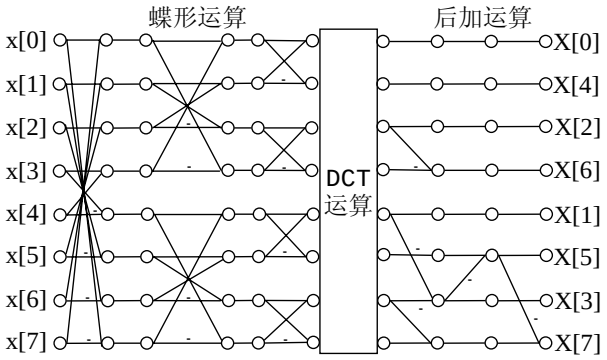

图 1　　8 点DCT算法流程图

从图 1 中可知在后加运算部分中产生一系列折线，随着输入序列点数的增加，折线形式也将越复杂。这些折线虽具有一定规律但其标号系统复杂，很难在代码级对其表述。对于系统的标号系统设计，我们希望它的算法结构具有很强规律性，数据的读写地址能用简单通用的数学表达式表述。为此，本文重点对图 1 所示的算法结构进行优化改进。

2 后加运算的优化算法

2.1 算法提出

这里你是你创新的干货所在，把公式、系统结构图和流程图都堆在这里吧。注意逻辑的严密性，不要有漏洞。

根据上文的算法可将 N 点序列分解的过程用图 2 表示。

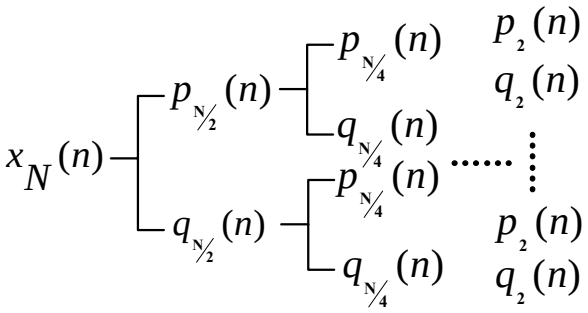

图 2　　N点输入序列分解流程图

对于N点输入序列，共有s级分解，由（6）可知，$p(n)$分解的序列进行DCT运算后得出的结果为它所对应的$C(i)$;但q(n)分解的序列进行DCT运算后得出的结果为它所对应的$V(i)$，需根据（7）和（10）做一组递推运算，可得出它对应的$D(i)$。因此，每做一级蝶形分解，在后面就需做一级$V(i)\rightarrow D(i)$的还原运算，即图 1 中的后加运算部分。

对于第 m 级蝶形分级，共可分解出 j 组 $q(n)$序列，即需要做 j 组后加运算。对于第 m 级的每一组则需做 k 次运算，可推得 s、j、k 关系

$$s=\log_2^N-1 \qquad (11)$$

$$j=2^{m-1} \qquad (12)$$

$$k=2^{\log_2^N-m}-1 \qquad (13)$$

基于以上分析，将s级分解计算得出的$D_s(i)$做后加运算可得出输入序列的$D(i)$。后加运算公式可如下表示

$$D_{s-m}(idxD)=\begin{cases}\frac{1}{2}V_s(idxV)\\ V_{s-1-m}(idxV)-D_{s-m}(idxD)\end{cases} \qquad (14)$$

其中

$$idxV=2^{s-m}+k*2^{s+1-m}+j-1 \quad (15)$$

$$idxD = 2^{s-m} + (k-1) * 2^{s+1-m} + j - 1 \quad (16)$$

从（14）和（15）可得在每一级的后加运算中V_{s-1-m}和D_{s-m}的序列标号之差为2^{i+1-m}常数，因此在后加运算部分的流程图中，将运算节点连接将会形成一条有规律的直线。但因前端碟形运算使后加运算的输入序列按位反序，从而在流程图中产生一系列折线段，如图 1。因此，若将后加运算部分的输入序列进行位反序处理使其正序排列，则后加运算在流程图中将会形成一条直线段，如图 2 以 8 点DCT输入序列为例。

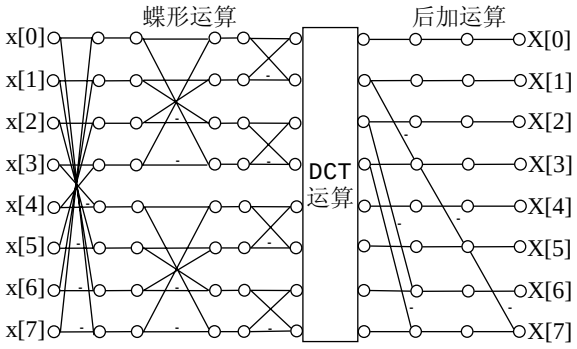

图 2　改进算法后 8 点DCT流程图

通过这种方法可以将复杂的难以用数学公式表述的标号系统转化为具有一定规律、利于代码实现的标号系统。可实现同址设计，降低了算法在代码级实现的复杂度。

2.2 标号系统的设计

对于本算法的标号系统设计可分成蝶形分解和后加运算两部分。本算法的提出是为降低后加运算部分的代码设计复杂度，本文对其做详细介绍。

通过算法分析可知，对于基-2 的 N 点序列，分解出一组奇数组，DCT 变换后即按公式（14）做一组后加运算。蝶形运算中每累加一次即对应图（2）中的一个节点运算。一组该递归运算即对应图（3）中的一条直线。在蝶形分解过程中一级将会同时分解出若干奇数组，在递归运算过程中即形成若干条直线，这些直线构成一个直线族。因此，后加运算即可采用“族运算→组运算→节点运算”的顺序进行设计实现。对于 N 点 DCT 运算需做 *s* 次族运算、*j* 次组运算和 *k* 次节点运算。图（3）为 8 点后加运算流程图。

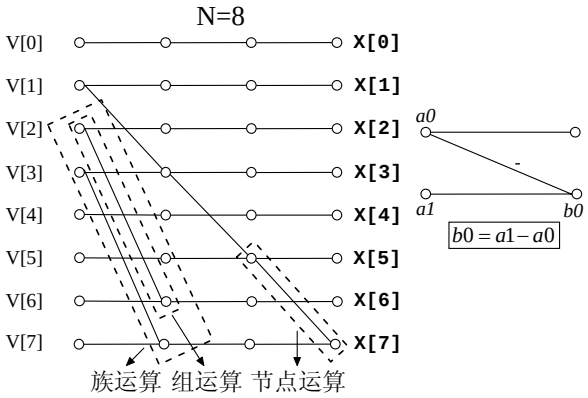

图 3　　8 点后加运算流程图

在本设计中，标号系统采用三维寻址的运算策略，通过三层循环完成标号的寻址。首层循环对应于流程图中的族运算，次层循环对应流程图中的组运算，末层循环对应流程图中的节点运算。

根据图（3）所示可知每次递归运算按照如下规律存取数据：*b0* 写入*a1* 的位置。输入数据*a0*、*a1* 的读取地址*Ra0*、*Ra1*，输出数据*b0* 的写入地址W_{b0}需和标号*s*，*j*，*k*对应。其中*Wb0*== *Ra1*。根据公式(15)(16) 对数据进行读写运算操作，可实现后加运算。

由以上分析可知，对于任意基-2 点的 DCT，后加运算部分都可以通过三层循环运算得出。采用该算法结构设计的代码，只需对输入点数进行配置即可，无须修改代码结构。可提高系统的重用性。

3 代码生成器

结合本文提出的新的算法结构设计基-2DCT快速算法代码生成器，并以TMS320C672xDSP处理器为测试平台，对不同点数DCT运算进行代码效率的测试。根据标号系统的设计可知，后加运算部分需三层循环完成，因此代码生成器的设计中可采用循环展开技术[7]，对代码分别进行部分展开和全部展开处理。用户使用代码生成器时可根据实际对代码尺寸和运行效率的需求对循环展开参数进行设置，获取相应代码。

通过利用 CCStudio 软件提供的性能分析工具来估计本文提出的 DCT 快速算法在 TMS320C672x 上运行时所需要的指令周期，从而评估该代码生成器的运算性能。表 1 给出不同点数输入序列运算时所需要的指令周期。

论文中要有性能评测，最好是表格或是曲线，因为你用了很多的文字来推销了自己的成果，那么怎样能够有说服力的证明你成果的价值呢，表格和

曲线是最有说服力的。表格和曲线的作用在于可以进行量化的对比。这种对比可以是 1、和别人的成果对比，如果别人的文章里面有相关数据的话。2、和自己的成果对比，如果在别人的文章里面没有相关方面的数据。不管用什么来对比，原则只有一个，把你跷跷板上高的那一端拿出来秀。

这篇文章的表格属于和自己对比类型的，因为以往的论文只是统计乘加数量，不统计处理器实测周期。本文的算法在乘加数量上没有突破，但是寻址策略的优化使得算法运行对于 Cache 和流水线更为友好，于是就使用处理器周期作为评价指标。

另外，如果你的研究成果不是算法或电路，而是一种方法或流程或软件结构，那么也尽可能用量化的方法来评价自己的成果，因为量化的结果比较有说服力，会提高命中率。

表 1　DCT 快速算法性能比较

输入序列		32 点	64 点	256 点	512 点
指令周期	不展开	6186	14184	72068	159922
	展开 1 层	5640	12310	58002	125032
	展开 2 层	5394	10967	50241	108886
	展开 3 层	4752	10326	49602	108248

表 2　代码尺寸比较

输入序列		32 点	64 点	256 点	512 点
代码尺寸	不展开	1632B	1632B	1632B	1632B
	展开 1 层	2048B	2240B	2592B	2752B
	展开 2 层	2816B	4224B	13120B	24896B
	展开 3 层	2176B	4928B	28160B	66080B

从表 1 可以看出，应用代码生成器运算 DCT，随着输入序列点数的增加，应用循环展开技术后节省的指令周期效果越显著。用户在应用该代码生成器时可根据实际需求，通过输入参数配置循环展开层次，从而达到对代码尺寸和运行效率的兼顾，更加灵活的满足用户需求。表 2 给出了采用循环展开技术后代码尺寸的比较。

4 结束语

结束语就是降龙十八掌的打完收功，在这里再次总结自己的成果，实际上就是引言部分意思的适当翻版，但是注意，意思的翻版不是文字的拷贝，不要直接拷贝引言部分。

代码体积和运行效率是设计一个高效实用快速算法所必需面对的两个问题。特别是在资源有限、运行速度低的开发平台上这个问题显得尤为重要。本文提出的 DCT 快速算法以 C. W. Kok 提出的算法为基础，对其算法结构进行优化改进，使算法更利于代码级的实现。采用本文提出的算法结合实际需求完成了 DCT 代码生成器的设计。并采用循环展开技术，使用户可根据对时间或存储效率的需求，合理的利用系统资源，具有一定实用价值。

总的来看，文章是一个纺锤形的，1 引言，2 描述，3 结尾。3 处的核心理念都是在介绍你的成果，3 处所不同的是介绍的深度和详细程度不同，但是整体而言，你的文章是连贯的。

参考文献：

[1] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform[J]. IEEE Trans. Computers, 1974, 23(1): 90-93.

[2] CHEN W H, SMITH C H, FRALICK S C. A fast computational algorithm for the discrete cosine transforms [J]. IEEE Transactions on Communications, 1997, 25(9): 1004-1009.

[3] LEE B G. A new algorithm to compute the discrete cosine transform [J]. IEEE Transactions on Acoustics, Speech and Signal Processing,1984,32(6):1243-1245

[4] C. M. Liu, Wen-Chieh Lee. A Unified Fast for Cosine Modulated Filter Banks in Current Audio Coding Standards [J]. Journal of Audio Engineering Society, 1999, 47(12): 1061-1075

[5] C. W. Kok, Fast algorithm for computing discrete cosine transform [J]. IEEE Transactions on Signal Processing, 1997,45(3): 757–760.

[6] John L. Hennessy, David A. Patterson. Computer Architecture: A Quantitative Approach, Fourth Edition [M]. Singapore: ELSEVIER, 2007. 288-310

[7] Weiss S, Smith J E. A study of scalar compilation techniques for pipelined supercomputers [J]. ACM Transactions on Mathematical Software, 1987, 22(10):105-109

作者简介：于成龙（1985-），男，硕士研究生，研究方向：数字通信技术、数字信号处理技术等，E-mail：ycl@cuc.edu.cn ；杜伟韬（1980-），男，中国传媒大学讲师，研究方向：数字通信技术、数字信号处理的 VLSI 实现等。