H.264中整数DCT变换及量化的DSP实现

刘宝兰 刘贵忠 苏睿

(西安交通大学电子与信息工程学院信息与通信工程系，陕西 西安 710049)

摘 要：在TI的TMS320DM642开发板上实现了H.264视频压缩编码标准中像素压缩模块的优化。重点对H.264协议中的整数DCT变换编码和量化进行了理论分析，并对其进行仿真及优化，有效地降低了整个模块的运行时钟数。实验结果表明该方法的优化取得了良好效果。

关键词：H.264视频编码，整数DCT变换，TMS320DM642，优化

中图法分类号：TP311.56 **文献标识码**：A **文章编号**：1000-7180(2005)06-200-06

Implementation and Optimization of Pixel-Compression Module in H.264 Based on DSP System

LIU Bao-lan, LIU Gui-zhong, SU Rui

(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049 China)

Abstract: As an important content of H.264 compression standard, pixel-compression module is used to reduce the correlation of picture using integer transform, quantization coding. Generally, this module will consume most of the CPU time except for the motion estimation due to its complicated matrix operation. In this paper, 44 integer transform in H.264 is employed, which can reduce computational complexity and avoid occurrence of decoder mismatch. In order to satisfy the real-time requirement, an efficient optimization method of C code and assemble code for pixel-compression module is proposed and evaluated. Experimental results indicate that better performance can be achieved for this work.

Key words: H.264, Integer transform, TMS320DM642, Optimize

1 引言

H.264/AVC是目前算法复杂度最高、性能最好的基于混合编码框架的视频编码标准。开始研究于1997年，最终正式制定于2003年5月，是ITU-T的VCEG和ISO/IEC的MPEG合作成立的JVT(Joint Video Team)开发出来的新一代的视频编码标准。ITU将其命名为H.264/AVC (Advanced Video Coding)；ISO/IEC将其命名为MPEG-4 AVC part 10。

H.264编码效率达到MPEG-2的两倍以上[1]。在网络上传输，不仅可以节省可观的高速传输设备投入，而且能节省一半以上的有线带宽资源、无线频谱资源和存储容量。它这种优于以往视频压缩标准的压缩性能，使得它可应用于因特网、数字摄像、数字视频录像、DVD及数字电视等许多领域。目前，许多大公司(如Nokia和Videolocus等)都已经开始把H.264压缩技术应用到各个领域中。

收稿日期：2005-01-05

基金项目：国家自然科学基金项目(60272072)

国家教育部"跨世纪优秀人才"培养计划项目

国家教育部"十五"、"211"工程重点科研项目

H.264采用一系列新的压缩方法，获得了更好的压缩效果。标准中的整数DCT变换和量化与先前的标准有很大的不同。本文首先简要介绍传统的DCT变换过程，然后详细介绍了H.264采用的整数DCT变换和量化，着重探讨DSP的代码优化方法。

2 传统的DCT变换

传统的二维DCT变换原理如下：

$$F(u,v)=\sqrt{\frac{2}{N}}C(u)\sum_{x=0}^{N-1}[\sqrt{\frac{2}{N}}C(u)\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}]\cos\frac{(2y+1)v\pi}{2N} \quad (1)$$

这里$u,v,x,y=0,1,2,\ldots,N-1$。其中，x和y是象素域的空间坐标；u和v是变换域的坐标；而：

$$C(u),C(v)=\begin{cases}\frac{1}{2^{1/2}}u, & v=0\text{ 时}\\ 1 & \text{其它}\end{cases}$$

DCT的二维变换可以分解成两次一维变换来完成，先对图像的每一行进行一维DCT变换，得到式(2)，再对每一列进行一维DCT变换，得到最终结

果式(3)：

$$f(u,y)=\sqrt{\frac{2}{N}}C(u)\sum_{x=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N} \quad (2)$$

$$F(u,v)=\sqrt{\frac{2}{N}}C(u)\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2y+1)v\pi}{2N} \quad (3)$$

3　H.264 的整数 DCT 变换及量化原理

H.264 标准之前的视频编码标准，如 MPEG-1、MPEG-2、MPEG-4、H.261 及 H.263 中都采用 8×8 点的 DCT 作为压缩模块的基本变换。H.264 中，对每一个残差宏块进行变换、量化和编码。在 Baseline Profile 中，根据残差数据的类型不同，分为三种编码方式：① 针对所有残差数据的 4×4 块整数 DCT 变换；② 4×4 亮度 DC 系数变换（仅在 16×16 帧内宏块预测模式下）；③ 2×2 色度 DC 系数变换（所有宏块中）。

宏块中的数据按图 1 所示顺序传输[2]。若宏块以 16×16 的帧内模式编码，则将宏块中所有 DC 系数提取出来，组成一个 4×4 的块，标记为-1。这个包含着每个 4×4 亮度块的 DC 系数块首先被传输。然后，亮度残差块 0-15 按照所示顺序被传输（其中在 16×16 的帧内宏块中的 DC 系数被设置为 0）。块 16、17 包含一个 2×2 的色度 DC 系数矩阵，最后，色度残差块 18-25（其中 DC 系数为 0）被传输。

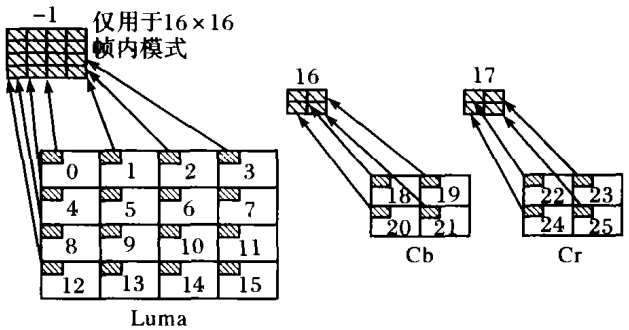

图1　宏块中残差块的扫描传输顺序

3.1　4×4 点整数 DCT 变换和量化（0-15，18-25 块）

3.1.1　4×4 点的整数 DCT 变换

运动估计和运动补偿之后，对残差数据进行 4×4 的整数 DCT 变换（图 1 中标记的 0-15 块和 18-25 块）。整数 DCT 变换基于传统的 DCT 变换，但与其有几个重要的不同点：① 这是一个整数变换（所有的操作都用整数执行，解码端不会有任何失真）；② 在 H.264 标准中详细说明了反变换的过程，若按照标准执行，则在编码器和解码器中不会出现误匹配；③ 变换的核心部分没有使用乘法，而仅使用了加法和移位操作；④ 变换中缩放矩阵的乘法运算被整合到量化过程中，降低了整个运算的乘法数。

4×4 整数 DCT 变换是由传统的 DCT 变换推演而来，其演变过程如下：

$$Y=AXA^T=\begin{bmatrix}a&a&a&a\\b&c&-c&-b\\a&-a&-a&a\\c&-b&b&-c\end{bmatrix}[X]\begin{bmatrix}a&b&a&c\\a&c&-a&-b\\a&-c&-a&b\\a&-b&a&-c\end{bmatrix} \quad (4)$$

上式是一个传统的 4×4 点的 DCT 变换，其中，$a=\frac{1}{2}$，$b=\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right)$，$c=\sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right)$。

对矩阵进行因式分解，则式(4)等价于式(5)：

$$Y=(CXC^T)\otimes E=\left(\begin{bmatrix}1&1&1&1\\1&d&-d&-1\\1&-1&-1&1\\d&-1&1&-d\end{bmatrix}[X]\begin{bmatrix}1&1&1&d\\1&d&-1&-1\\1&-d&-1&1\\1&-1&1&-d\end{bmatrix}\right)\otimes\begin{bmatrix}a^2&ab&a^2&ab\\ab&b^2&ab&b^2\\a^2&ab&a^2&ab\\ab&b^2&ab&b^2\end{bmatrix} \quad (5)$$

式(5)中，CXC^T 是“核心”2D 变换；E 为缩放矩阵；符号⊗表示 (CXC^T) 中的每个元素与矩阵 E 中相同位置元素对应相乘（即为标量相乘而不是矩阵相乘）；常数 a、b 的定义与式(4)中相同，$d=c/b$（约为 0.414）。通常，为了简化运算，取 $d=0.5$。而为了确保变换的正交性，b 值也要做一定的调整，因此，a、b、d 定义如下：

$$a=1/2,\ b=\sqrt{2/5},\ d=1/2$$

变换的过程只使用整数变换，而矩阵 C 的第二行和第四行以及矩阵 C^T 的第二列和第四列出现了 1/2，会使精度降低，故乘以缩放因子 2 使 1/2 整数化，在矩阵 E 中除 2 作为补偿。所以，最终的 4×4 整数 DCT 变换为：

$$Y=C_fXC_f^T\otimes E_f=\left(\begin{bmatrix}1&1&1&1\\2&1&-1&-2\\1&-1&-1&1\\1&-2&2&-1\end{bmatrix}\begin{bmatrix}1&2&1&1\\1&1&-1&-2\\1&-1&-1&2\\1&-2&1&-1\end{bmatrix}\right)\otimes\begin{bmatrix}a^2&ab/2&a^2&ab/2\\ab/2&b^2/4&ab/2&b^2/4\\a^2&ab/2&a^2&ab/2\\ab/2&b^2/4&ab/2&b^2/4\end{bmatrix} \quad (6)$$

该变换近似于一个传统的 4×4 点的 DCT 变换，但是由于 b 和 d 值的改变，它已不等同于以往的 DCT 变换。这种改变克服了传统 DCT 变换的一个缺点：矩阵中的部分系数是无理数，在计算机中使用迭代方法进行变换和反变换后，不能得到一致的初始值。

相应的反变换定义如下：

$$X'=C_i^T(Y\otimes E_i)C_i=\begin{bmatrix}1&1&1&1/2\\1&1/2&-1&-1\\1&-1/2&-1&1\\1&-1&1&-1/2\end{bmatrix}\left([Y]\otimes\begin{bmatrix}a^2&ab&a^2&ab\\ab&b^2&ab&b^2\\a^2&ab&a^2&ab\\ab&b^2&ab&b^2\end{bmatrix}\right)\begin{bmatrix}1&1&1&1\\1&1/2&-1/2&-1\\1&-1&-1&1\\1/2&-1&1&-1/2\end{bmatrix} \quad (7)$$

上式中 Y 为整数变换后的系数，X' 为反变换后的恢复值，$a=1/2$，$b=\sqrt{2/5}$。

3.1.2 4×4 残差块的整数 DCT 量化原理

H.264 采用的是分级量化方式。基本的前向量化公式为：

$$Z_{ij}=\text{round}\,(Y_{ij}/Q\text{step}) \quad (8)$$

上式中，Y_{ij} 是整数 DCT 变换后的系数。Qstep 为量化步长，Z_{ij}是量化后的系数。Round()为下取整操作。

H.264 标准中共有 52 个量化参数 QP，它与量化步长 Qstep 的关系如表 1 所示。

表 1 量化参数QP与量化步长Qstep的关系

QP	0	1	2	3	4	5	6	7	8	9	10	11	…
Qstep	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	2.25	…
QP	…	18	…	24	…	30	…	36	…	42	…	48…	51
Qstep		5		10		20		40		80		160	224

从表 1 中可以看出，QP 值每增加 6，量化步长 Qstep 增加一倍。H.264 中量化步长的范围较广，这使得编码器在控制视频质量的正确性和灵活性之间有个很好的折衷。对于亮度块和色度块来说，QP 值略有不同。色度块的 QP 值 QP_C 的 52 个值是从亮度块的 52 个 QP 值 QP_Y 中抽取出来的，所以当 QP_Y >30 时，$QP_C<QP_Y$，如表 2 所示。

H.264 中，整数变换时的尺度缩放因数 a^2，$ab/2$，$b^2/4$ 被整合到前向量化器中计算。令 $W=CXC^T$，则每个 W_{ij}的操作为：

表2 QPC 与QPY的对应关系

QP_Y	<30	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
QP_C	$=QP_Y$	29	30	31	32	32	33	34	34	35	35	36	36	37	37	37	38	38	38	39	39	39	39

$$Z_{ij}=\text{round}\left(W_{ij}\cdot\frac{PF}{Q\text{step}}\right) \quad (9)$$

式中，PF 的取值取决于像素点的位置(i, j)，关系如表 3 所示。

表 3 PF 的值与像素点的位置关系

像素点在宏块中的位置				PF
(0,0)	(2,0)	(0,2)	(2,2)	a^2
(1,1)	(1,3)	(3,1)	(3,3)	b^2
其它				$ab/2$

为了简化计算，我们对式(8)进一步变形：

$$Z_{ij}=\text{round}\left(W_{ij}\cdot\frac{MF}{2^{q\text{bits}}}\right) \quad (10)$$

这里：

$$\frac{MF}{2^{q\text{bits}}}=\frac{PF}{Q\text{step}} \quad (11)$$

$$q\text{bits}=15+\text{floor}(QP/6) \quad (12)$$

整数运算中，除法可以用右移操作来完成，最终的量化操作为：

$$|Z_{ij}|=(|W_{ij}|\cdot MF+f)\gg q\text{bits} \quad (13)$$

$$\text{sign}(Z_{ij})=\text{sign}(W_{ij})$$

对于帧内块，上式中的 $f=2^{q\text{bits}}/3$；帧间块中，$f=2^{q\text{bits}}/6$。MF 的取值如表 4 所示。

表4 MF 的值与像素点的位置及QP 的关系

位置	QP =0	QP =1	QP =2	QP =3	QP =4	QP =5
(0,0)(2,0)(0,2)(2,2)	13107	11916	10082	9362	8192	7282
(1,1)(1,3)(3,1)(3,3)	5243	4660	4194	3647	3355	2893
其它	8066	7490	6554	5825	5243	4559

反量化的过程与量化过程相反：

$$Y_{ij}=Z_{ij}\cdot Q\text{step} \tag{14}$$

将尺度缩放矩阵 E 和尺度缩放因数 a^2,ab,b^2 整合在一起，为避免量化时的取整误差，反量化乘以因数64：

$$W'_{ij}=Z_{ij}Q\text{step}\cdot PF\cdot 64 \tag{15}$$

$W'_{ij}=C_i^T WC^T$，H.264中并没有直接规定 Qstep 和 PF 的值，而是在 $0\leqslant QP\leqslant 5$ 时，令 $V=Q\text{step}\times PF\times 64$，则针对每个像素点，上式变换为：

$$W_{ij}'=Z_{ij}V_{ij}\cdot 2^{\text{floor}(QP/6)} \tag{16}$$

$0\leqslant QP\leqslant 5$ 时，V 的取值如表5所示。

表5 V的值与像素点的位置及QP的关系

位置	QP=0	QP=1	QP=2	QP=3	QP=4	QP=5
(0,0)(2,0)(0,2)(2,2)	10	11	13	14	16	18
(1,1)(1,3)(3,1)(3,3)	16	18	20	23	25	29
其它	13	14	16	18	20	23

3.2 H.264中的亮度 *DC* 系数变换及量化(仅针对16×16帧内模式下)

3.2.1 H.264的4×4亮度DC系数变换

若被编码的宏块是在16×16的帧内预测模式下，则整个16×16点宏块像素值都是通过相邻像素预测得到的。这种情况下，相应的4×4点的像素块通过式(6)变换后得到的DC系数还有很大的相关性。所以，H.264中将每个4×4点的DC系数抽出，再采用Hadamard变换进行处理，如式(17)：

$$Y_D=\left(\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix}X_D\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix}\right) \tag{17}$$

X_D 是由16个DC系数组成的4×4的块，Y_D 是Hadamard变换后的结果。

3.2.2 H.264的4×4亮度DC系数量化原理

亮度DC系数经Hadamard变换后的结果 $Y_{D(i,j)}$ 采用式(18)进行量化：

$$|Z_{D(i,j)}|=(|Y_{D(i,j)}|MF_{(0,0)}+2f)\gg(q\text{bits}+1) \tag{18}$$
$$\text{sign}(Z_{D(i,j)})=\text{sign}(Y_{D(i,j)})$$

上式中，$MF_{(0,0)}$值如表4所示，$f=2^{q\text{bits}}/3$，$q$bits=15+floor(QP/6)。

在解码端，先进行反Hadamard变换，再进行反量化操作。这个步骤与通常我们所采用的步骤不同，具体反变换过程为：

$$W_{QD}=\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix}[Z_D]\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&-1&1\\1&-1&1&-1\end{bmatrix} \tag{19}$$

反量化为：

$$W'_{D(i,j)}=W_{QD(i,j)}V_{(0,0)}2^{\text{floor}(QP/6)-2} \qquad QP\geqslant 12 \tag{20}$$
$$W'_{D(i,j)}=[W_{QD(i,j)}V_{(0,0)}+2^{1-\text{floor}(QP/6)}]\gg(2-\text{floor}(QP/6)) \qquad QP<12$$

其中，$V_{(0,0)}$是尺度缩放因数 V 在表4中(0,0)位置上的相应值。H.264采用这种特定的反变换顺序可以扩大反变换的动态范围。一般来说，DCT变换后的能量主要集中在DC系数中，而H.264采用针对DC系数的特殊的变换，使能量进一步集中在少数几个重要系数中。

3.3 H.264的2×2色度DC系数变换及量化

在H.264宏块中，每个色度分量由4个4×4的块组成。先对这4个块分别作4×4的整数DCT变换，将变换后的直流分量组成一个2×2的DC系数块，记作 W_D。对这个2×2的DC系数块进行离散Hadamard变换，然后进行量化。

正向Hadamard变换为：

$$Y_D=\begin{bmatrix}1&1\\1&-1\end{bmatrix}[W_D]\begin{bmatrix}1&1\\1&-1\end{bmatrix} \tag{21}$$

对变换后的输出结果 Y_D 进行量化：

$$|Z_{D(i,j)}|=(|Y_{D(i,j)}|\cdot MF_{(0,0)}+2f)\gg(q\text{bits}+1) \tag{22}$$
$$\text{sign}(Z_{D(i,j)})=\text{sign}(Y_{D(i,j)})$$

$MF_{(0,0)}$值如表4所示；对于帧内块，上式中的 $f=2^{q\text{bits}}/3$；帧间块中 $f=2^{q\text{bits}}/6$，$q$bits=15+floor(QP/6)。

在解码端，先进行反Hadamard变换，再进行反量化操作。这个步骤与4×4亮度DC系数变换量化步骤相同，具体反变换过程为：

$$W_{QD}=\begin{bmatrix}1&1\\1&-1\end{bmatrix}[Z_D]\begin{bmatrix}1&1\\1&-1\end{bmatrix} \tag{23}$$

上式中，W_{QD} 为2×2的色度DC系数反变换后的结果。再对其结果进行反量化：

$$W'_{D(i,j)}=W_{QD(i,j)}\cdot V_{(0,0)}\cdot 2^{\text{floor}(QP/6)-1} \qquad QP\geqslant 6$$
$$W'_{D(i,j)}=[W_{QD(i,j)}\cdot V_{(0,0)}]\gg 1 \qquad QP<6 \tag{24}$$

对于亮度和色度DC系数的变换和量化可以进一步提高压缩效率。

3.4 H.264中的像素压缩模块的实现流程

H.264中对输入的残差块 X 进行整数DCT变换、量化、反量化和反变换的完成过程如图2所示。

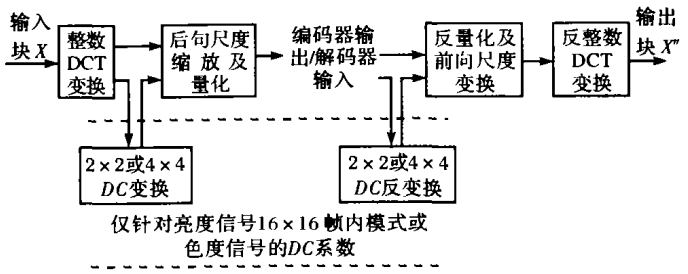

图2 变换、量化、反量化及反变换流程图

输入为4x4的残差系数,前项核心变换为:$W=C_fXC_f^T$,量化为:$Z=W\cdot\text{round}(PF/Q\text{step})$。解码端,反量化:$W'=Z\cdot Q\text{step}\cdot PF\cdot 64$,反核心变换 $X'=C_i^TW'C_i$,修正系数:$X''=\text{round}(X'/64)$,即为最终输出 X''。

4 算法的DSP实现及优化

我们采用美国德州仪器(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。TMS320DM642是TI公司在2003年推出的一款新型数字媒体处理器,它是在TMS320C64x的基础上增加了很多外设和接口开发出来的。所以,它能与TI的其它C64x数字信号处理器进行代码兼容,它具有目前最先进的定点指令集。同时还能提供片上集成的高精度(HD)视频端口、无缝以太网、多通道音频及66 MHz的PCI连接性。这种高集成度的处理器,不仅使软件可编程性更加方便,还超越了许多传统的硬连线固定功能芯片组。从而大大降低了系统成本,进一步简化了开发进程。

本文所用的软件开发平台是TI公司的集成开发环境CCS(Code Composer Studio),该软件平台提供了基于C语言的系统调试工具、汇编优化器和链接器,给程序的开发提供了极大的方便。

代码开发流程分为3个阶段:第一阶段产生C代码,第二阶段优化C代码,第三阶段编写线性汇编。每个阶段完成的任务如下:

第一阶段:在不考虑任何与C6000有关知识的情况下,根据任务编写C代码。然后使用profiling工具确定代码中的低效率段。通常这个阶段的代码性能很低,为进一步改进代码性能,需进入第二阶段。

第二阶段:利用内联函数、编译选项和其他优化方法改进C代码。使用性能测试工具(profiling)检查其性能,如果代码仍不能达到所期望的效率,则进入第三阶段。

第三阶段:从C代码中抽出对性能影响很大的代码段,用线性汇编重新改写这段代码,使用汇编优化器优化该代码[4,5]。

以下结合H.264编码器具体实现过程,分别对这三个部分作较详细的介绍。

H.264编码器在CCS上的优化,主要解决的是编码效率问题。如何提高编码速度,减少算法实现的时钟数,是对算法进行优化的主要目标。为了提高执行速度和效率,在进行软件开发时,经常采用C语言和汇编语言混合编程。混合编程一般采用C语言为主体框架,然后嵌入汇编代码的方式。具体的优化方法有:

(1) 将乘除运算均改为移位运算 在运算过程中,所有的乘法和除法运算均可以转化成左移和右移操作。

(2) 采用内联函数 CCS编译器提供的intrinsics是直接与C6000汇编指令映射的在线函数。一般来说,不易用C/C++语言实现其功能的汇编指令都有相对应的intrinsics函数。

(3) 数据打包处理 CCS访问存储器是很费时的,若程序中所使用的Load/Store指令比较多,则可以采用数据打包处理方式,即以32位存取指令对16位数据进行存取,可以节省一半的存取周期。

(4) 使用编译器编译优化选项,打开软件流水,展开循环,以减小时钟周期数。

当函数的逻辑关系复杂,判断、跳转、函数调用情况特别多时,优化整个复杂函数的方法是不划算的。这时可以使用线性汇编将其中的循环部分改写成一个函数,以优化后的函数调用代替循环部分,可以提高算法的运行速度。但是,算法的速度和代码的长度始终是一对矛盾。提高了速度,代码长度必然增加,反之,缩小代码长度,时钟周期数也必然加大。

在整个算法实现的过程中,汇编算法的优化一定要与相对应的芯片的结构相联系。如寄存器的分配,乘法加法运算的溢出都要详细地考虑。

5 实验结果

本文在TMS320DM642DSP处理器中实现了H.264编码中像素压缩模块算法,并对其进行了仿真和优化,调试成功。主要重点和难点是编码算法的优化。表6是部分函数优化前后的性能对比,以及将本文所作的H.264中的整数DCT/IDCT变换与传统的8x8点DCT/IDCT变换的比较。

在上表中,dct_luma()、Quant()/Dequant()和Idct_luma()分别表示本文所采用的整数dct函数,量

表6 部分函数优化前后性能对比

函数名	优化前 clock	优化后 clock	倍数
本文算法的 dct_luma ()	1383	108	12.81
本文算法的 Quant( )/ Dequant()	2473	558	4.432
8×8 点的 dct_int32()	47386	1477	32.08
8×8 点的 Idct_int32()	91164	3256	28.00
本文算法的 Idct_luma()	2317	83	27.92

化/反量化函数,反变换函数;8×8 点的 Fdct_int32() /Idct_int32()是在相同的实验条件下,所作的传统的 8×8 点的 DCT/IDCT 变换;表中第二列表示算法优化前的时钟数,第三列表示算法优化后的时钟数。从上表中可以看出,在算法优化前,H.264 所采用的整数 DCT 算法的函数 dct_luma()和 Idct_luma()运行时所用的时钟数远少于传统的 DCT 算法的时钟数。对于 4 点的一维 DCT 变换,其核心部分的算法没有使用乘法,只用到加法和移位操作,便于硬件实现。优化后,整数 DCT 变换时钟数少于传统 DCT 算法。表中第三列中,我们将优化前后时钟数进行一个对比,说明本文的优化结果确实达到了良好的效果。对于模块中的其它算法,我们也同样取得了良好的优化结果。

6 结束语

本文在 TMS320DM642 上实现了 H.264 Baseline Profile 中像素压缩模块算法。并对算法中的各个函数包括整数 DCT/IDCT 函数,量化/反量化函数,针对 *DC* 系数的变换和量化函数进行了有效的优化,达到了良好的效果。尽管本算法是在 TMS320DM642 上开发并实现的,但是可以嵌入到其他的 DSP 芯片中运行,也能方便地集成到其它应用程序中。

参考文献

[1] Iain E G Richardson. H.264 and MPEG-4 Video Compression. Great British by Antony Rowe, Chippenham, Wiltshire. 2003: 4~6.

[2] H.264/MPEG-4, Part 10, White Paper, www.vcodex.com

[3] ITU & ISO/IEC, Draft ITU-T Recommendation and Final Draft International Standard IF Joint Video Specification. ITU-T Rec. H.264 ISO/IEC 14496-10 AVC, the 7th Meeting:Pattaya,Thailand,7-14 March, 2003: 72.

[4] 李方慧,王飞,何佩琨. TMS320C6000 系列 DSPs 原理与应用 第二版. 北京:电子工业出版社 2003:197.

[5] Texas Instruments Incorporated. TMS320C6x Programmer's Guide. Literature Number: SPRU187E, Feb. 1999.

刘宝兰 女,(1980-),硕士研究生。研究方向为图像、视频数据压缩算法和 DSP 实现。

刘贵忠 男,(1962-),1989 年 6 月获荷兰 Eindhoven 大学博士学位,教授,博士生导师。研究方向为非平稳信号的分析与处理、音视频数据压缩、模式识别、反演等方面的理论、方法与应用研究。

苏 睿 男,(1974-),博士研究生。研究方向为视频压缩编码算法研究、视频压缩系统的 VLSI 实现,以及专用脉动阵列处理器结构设计等。

(上接第 199 页)

Transactions on Computers, 1980, series C29(12): 1104~1113.

[5] Schillo M, Kray C, Fischer K, The cager bidder problem: a fundamental problem of DAI and selected solutions, In: Maria G, Tour I, eds, proceedings of the Interational Joint Conference on Autonomous Agents and Mulitagent Systems, Bologna, Net York; ACM press, 2002, 599~606

[6] Sarit Kraus, Negotiation and Cooperation in Mulitagent Environments, Artificial Intelligence, 1997, 94: 79~97.

[7] Finin T., Fritzsom R., McKay D., et al. KQML an in Agent Communication Language [EB/ OL]. http://www.cs.umbc.edu/kqml/papers/kqml - acl. ps.

[8] Sycara K P. Multi-Agent Systems[C]. AAAI,1998, 79~92.

[9] Maes P. Sofeware Agents. CHI97 Sofeware Agents Tutorial, 1997.

蔡大鹏 男,(1967-),博士研究生。研究方向为分布式决策支持系统。

张书杰 男,(1942-),教授,博士生导师。研究方向为决策支持系统和网络安全的研究。