一种新的基于 VLIW 的 IDCT 和运动补偿算法

欧阳万里, 肖创柏, 刘 广
(北京工业大学计算机学院, 北京 100022)

摘 要: 本文使用矩阵形式在超长指令字(VLIW)的观点下将几种经典算法与已有的适合于 VLIW 的算法进行了比较. 然后利用 VLIW 结构的特性, 提出了一种快速 IDCT 算法. 与现有算法相比, 新算法进一步减少了所需的指令周期. 并利用 VLIW 结构的寄存器特性, 将视频编解码过程中的运动补偿(预测)和 IDCT(DCT)组合, 使运动补偿所需时间降低为原来的约 50%, 这种思想能应用于 MPEG1/2/4, H. 263 和 H. 264.

关键词: 超长指令字 (VLIW); 离散余弦变换 (DCT); IDCT; 快速算法; 并行算法; 运动补偿; 视频压缩; DSP

中图分类号: TP391. 41 文献标识码: A 文章编号: 0372-2112 (2005) 11-2074-06

A New IDCT and Motion Compensation Algorithm Based on VLIW

OU-YANG Warr-li, XIAO Chuang-Bai, LIU Guang
(*Institute of Computer of Beijing University of technology, Beijing* 100022, *China*)

Abstract: In this paper, using a very long instruction word (VLIW) architecture view, the existing state of the art algorithms were compared with an algorithm that is tailored for VLIW architecture in the form of matrix. The architecture's character is further exploited to design a new fast inverse discrete cosine transform (IDCT) algorithm. Compared with existing algorithms, the new algorithm reduces the number of instruction cycles needed. Further, the register character of VLIW is used for merging motion compensation with IDCT (DCT) in the video codec process. This achieves 50% speed improvement. The concept is applicable for MPEG1/2/4 and H. 264/AVC.

Key words: very long instruction word (VLIW); discrete cosine transform (DCT); IDCT; fast algorithm; parallel algorithm; motion compensation; video compression; DSP

1 引言

离散余弦变换(DCT)及其反变换(IDCT)是一种在信号、图像处理和视频压缩过程中常用的方法. 在图像压缩标准 JPEG、视频压缩标准 MPEG1/2/4 以及 H. 264 中, 都使用或者支持 DCT 来进行变换编码, 对于预测帧(P、B 帧), IDCT 变换后通常紧跟着是运动补偿, 他们都是很耗时的处理, 解码过程中使用频率都很高, 使用快速算法将两者速度提高具有现实意义.

对于 DCT 和 IDCT 的快速算法, 现有研究多从 DCT 出发, 用类似方法推及 IDCT.

文献[1~3]提出了基于多维 DCT 的快速算法.

8* 8 的二维 DCT 快速算法因在 MPEG1/2/4、H. 263 和 JPEG 中的应用, 理论研究和实际应用价值很高, 而利用行列变换分离性的算法(row-column method)由于其简单和快速而成为研究热点. 人们提出了许多 8 点 DCT 变换的方法, 经典的如文献[4,5]提出的算法一次 8 点运算使用 12 次乘法, 29 次加法; 文献[6]提出的算法只需要 11 次乘法和 29 次加法, 将所需乘法次数进一步降低. 由于出现连续乘法和不利于使用多媒体指令, 这些算法对于 VLIW 结构并不是最优的.

在基于 VLIW 结构的 DCT 和 IDCT 算法研究方面, 文献[11]较早地提出了一种利用 VLIW 结构的算法, 文献[9,10]提出了一种性能比文献[11]更好的算法. 文献[9,10]提出的算法更好地利用 VLIW 的特点来并行完成每个步骤内的计算任务, 使用定点运算, 进一步提高了运算速度, Trimedia 上的实际应用表明该算法比文献[6]提出的更有效.

同时还出现了很多应用于视频 IDCT 的研究, 如文献[12~14]. 其中文献[14]使用多媒体指令, 并考虑零系数多的特性, 对 MPEG2 解码中的 IDCT 进行了优化.

运动补偿算法方面, 从不同结构出发的算法也越来越多, 如适合于纯软件的方法文献[15].

本文在超长指令字(VLIW)的观点下对几种经典算法与已有的适合于 VLIW 的算法进行了比较. 然后提出了一种改进的基于 VLIW 的快速 IDCT 算法, 并利用寄存器特性, 将视频编解码过程中的运动补偿(预测)和 IDCT (DCT)组合.

文章第二部分先介绍了 VLIW 架构的特性, 给出了文献

收稿日期: 2004-11-10; 修回日期: 2005-05-08
基金项目: 北京市教育委员会科技发展计划项目(No. KM200510005012); 国家自然科学基金(No. C070515-02)

[6, 9, 10] 中算法的矩阵分解形式, 并用矩阵形式在 VLIW 结构的观点下对文献[4~6, 9, 10, 17] 提出的算法进行了分析比较, 然后提出了一种在文献[9, 10] 基础上更充分利用 VLIW 特性的 IDCT 快速算法. 第三部分介绍了如何利用 DSP 的寄存器特性和编程环境提供的支持, 通过寄存器传递中间结果, 将 IDCT 变换结合在一起的方法. 第四部分给出实验结果. 第五部分是结论.

2　一种改进的 IDCT 方法

2.1　支持 VLIW 的 DSP 具有的特性

支持 VLIW 的 DSP 具有的特性是一条指令同时执行多个操作, 每一个操作可以看成是实现某个功能(如整数加、乘, 读写内存等) 的 RISC 指令. 同一条指令内同时执行的有效操作数是不同的, 例如 TM1300 把指令中每一个操作认为占据一个操作位(slot) , 一条指令共有五个 slot, 所以一条指令最多可以同时执行 5 个有效操作. 一般认为平均一条指令可以执行的有效操作越多, 并行度越高. 每一个 slot 可以执行的操作的功能不同, 如表 1 所示.

表 1　TM1300 功能单元分配[10]

功能单元	Slot 分配				
	1	2	3	4	5
Constant	Y	Y	Y	Y	Y
Integer ALU	Y	Y	Y	Y	Y
Load/ Store				Y	Y
DSP ALU	Y		Y		
DSP MUL		Y	Y		
Shifter	Y	Y			
Branch		Y	Y	Y	
Int/ Float MUL		Y	Y		
Float ALU	Y			Y	
Float Compare			Y		
Float sqrt/ div		Y			

其中 i 行 j 列有 Y 代表在 j 指令位有功能单元 i, 空白代表 j 指令位没有功能单元 i. 如果一条指令中排不满 5 个操作, 就会以空操作(nop) 形式代表此操作位不进行任何有效操作. 由于内存操作容易产生 nop, 并可能造成机器停转(stall) , 内存读写操作对速度是有影响的. VLIW 结构一般寄存器数目多, 例如 TM1300 有 128 个寄存器.

根据 VLIW 特性来设计算法, 对并行度是有要求的. 同时, 由于最大并行度低, 所需操作数目的减少带来的效率提高也是需要考虑的因素.

2.2　一维算法的矩阵形式和分析

一维 DCT 的公式为:

$$y_n = \sum_{k=0}^{N-1} C_n x_k \cos\left[\frac{(2k+1)\pi n}{2N}\right]$$

一维 IDCT 的公式为:

$$x_k = \sum_{n=0}^{N-1} C_n y_n \cos\left[\frac{(2k+1)\pi n}{2N}\right]$$

其中 $C_0 = 1/\sqrt{N}$; $C_n = \sqrt{2/N}$, $n \neq 0$.

8 点 DCT 变换可用 $Y = \boldsymbol{C}_8 * X$ 的矩阵形式来表示, 设 $r(k) = \cos(k\pi/16)$, 其中 $\boldsymbol{C}_8$ 为:

$$\boldsymbol{C}_8 = \frac{1}{2}\begin{bmatrix} r(4) & r(4) & r(4) & r(4) & r(4) & r(4) & r(4) & r(4) \\ r(1) & r(3) & r(5) & r(7) & -r(7) & -r(5) & -r(3) & -r(1) \\ r(2) & r(6) & -r(6) & -r(2) & -r(2) & -r(6) & r(6) & r(2) \\ r(3) & -r(7) & -r(1) & -r(5) & r(5) & r(1) & r(7) & -r(3) \\ r(4) & -r(4) & -r(4) & r(4) & r(4) & -r(4) & -r(4) & r(4) \\ r(5) & -r(1) & r(7) & r(3) & -r(3) & -r(7) & r(1) & -r(5) \\ r(6) & -r(2) & r(2) & -r(6) & -r(6) & r(2) & -r(2) & r(6) \\ r(7) & -r(5) & r(3) & -r(1) & r(1) & -r(3) & r(5) & -r(7) \end{bmatrix}$$

8 点 IDCT 的矩阵形式为 $X = \boldsymbol{C}_8^T Y$. (T 表示转置)

通常 DCT 快速算法(如文献[4~6, 9, 10, 17] 中提出的算法) 第一步将矩阵 $\boldsymbol{C}_8$ 分解为:

$\boldsymbol{C}_8 = 1/2PMQ$, 其中:

$$\boldsymbol{P} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} 1&0&0&0&0&0&0&+1 \\ 0&1&0&0&0&0&+1&0 \\ 0&0&1&0&0&+1&0&0 \\ 0&0&0&1&+1&0&0&0 \\ 1&0&0&0&0&0&0&-1 \\ 0&1&0&0&0&0&-1&0 \\ 0&0&1&0&0&-1&0&0 \\ 0&0&0&1&-1&0&0&0 \end{bmatrix}$$

$$\boldsymbol{M} = \begin{bmatrix} r(4) & r(4) & r(4) & r(4) & & & & \\ r(4) & -r(4) & -r(4) & r(4) & & & & \\ r(2) & r(6) & -r(6) & -r(2) & & & & \\ r(6) & -r(2) & r(2) & -r(6) & & & & \\ & & & & r(1) & r(3) & r(5) & r(7) \\ & & & & r(3) & -r(7) & -r(1) & r(5) \\ & & & & r(5) & -r(1) & r(7) & r(3) \\ & & & & r(7) & -r(5) & r(3) & -r(1) \end{bmatrix}$$

其中空白表示 0. 变换后的矩阵 $\boldsymbol{M}$ 左上角是对偶系数的求解, 右下角是对奇系数的求解.

对于矩阵 $\boldsymbol{M}$ 的左上角的分解, 文献[6, 9, 10] 提出的算法都采用以下形式:

$$\begin{bmatrix} r(4) & r(4) & r(4) & r(4) \\ r(4) & -r(4) & -r(4) & r(4) \\ r(2) & r(6) & -r(6) & -r(2) \\ r(6) & -r(2) & r(2) & -r(6) \end{bmatrix} = r(4) * \begin{bmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & \sqrt{2}r(6) & \sqrt{2}r(2) \\ & & -\sqrt{2}r(2) & \sqrt{2}r(6) \end{bmatrix} \begin{bmatrix} 1 & & & 1 \\ & 1 & 1 & \\ & 1 & -1 & \\ 1 & & & -1 \end{bmatrix}$$

文献[17] 提出的算法对上式进一步分解, 但如果使用定点运算, 对于有多媒体指令 IFIR(指令含义见图 4 中 IFIR16) 的 VLIW 架构已经没必要了. 由于文献[6, 9, 10] 对 $\boldsymbol{M}$ 左上角采取的方法相同, 我们不作进一步分析.

对于矩阵 $\boldsymbol{M}$ 的右下角, 文献[9, 10] 直接使用 $\boldsymbol{M}$ 右下角来对奇系数求解. 而文献[6] 中标准形式采用以下形式来求解:

$$\begin{bmatrix} r(1) & r(3) & r(5) & r(7) \\ r(3) & -r(7) & -r(1) & -r(5) \\ r(5) & -r(1) & r(7) & r(3) \\ r(7) & -r(5) & r(3) & -r(1) \end{bmatrix} =$$

$$r(4)^{*}\begin{bmatrix} 1 & & & 1 \\ & \sqrt{2} & & \\ & & \sqrt{2} & \\ -1 & & & 1 \end{bmatrix}\begin{bmatrix} 1 & & 1 & \\ & -1 & & 1 \\ 1 & & -1 & \\ & 1 & & 1 \end{bmatrix}\begin{bmatrix} r(3) & & & r(5) \\ & r(1) & r(7) & \\ & -r(7) & r(1) & \\ -r(5) & & & r(3) \end{bmatrix}$$

文献[17]提出的Scaled DCT算法则使用如下形式:

$$\begin{bmatrix} r(1) & r(3) & r(5) & r(7) \\ r(3) & -r(7) & -r(1) & -r(5) \\ r(5) & -r(1) & r(7) & r(3) \\ r(7) & -r(5) & r(3) & -r(1) \end{bmatrix} =$$

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & & -1 \\ & & 1 & \end{bmatrix} G4\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & & 1 \\ & & -1 & \end{bmatrix}\begin{bmatrix} & & & 1 \\ & & 1 & \\ & 1 & & \\ 1 & & & \end{bmatrix}$$

$$G4 = \frac{1}{2}D4^{-1}$$

$$\begin{bmatrix} 1 & 1 & -1 & 0 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 \\ 1 & -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & & & \\ & r(4) & & \\ & & r(6) & r(2) \\ & & -r(2) & r(6) \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

$$D4 = \begin{bmatrix} r(5) & & & \\ & r(1) & & \\ & & r(3) & \\ & & & r(7) \end{bmatrix}$$

从VLIW结构的观点看,如果直接使用矩阵M的右下角来求奇系数,会获得很好的并行性,不存在连续的乘法,而且便于使用多媒体操作IFIR来完成乘加操作;文献[6]中的算法虽然需要的VLIW操作少,但是有连续的乘法(nested multiplication),效率反而不如文献[9,10]提出的高,文献[4,5]提出的算法连续乘法数目更多,也不利于在VLIW上的应用.文献[17]中介绍的一维方法,如果将$D4^{-1}$放入量化,虽然不存在连续的乘法,对于一般指令需要操作数少,但由于它不规则,不利于充分利用VLIW架构中的特有指令,定点运算时实际所需VLIW指令更多,而且读写相关的步骤增多,如果使用行列分离的方法来解决二维问题,效率不如文献[9,10]提出的高.

2.3 一种改进的IDCT算法

文献[9,10]提出的IDCT蝶形算法如图1所示,图2是关于图1的计算符号及含义说明,其DCT算法参见文献[9,10].

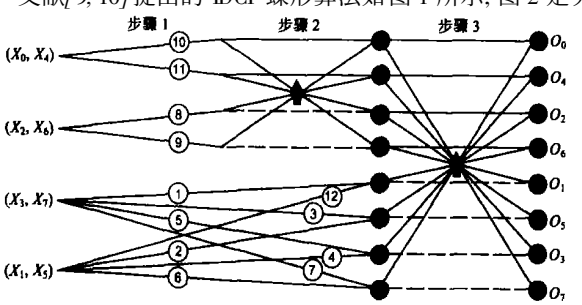

图1 文献[9,10]中的IDCT算法

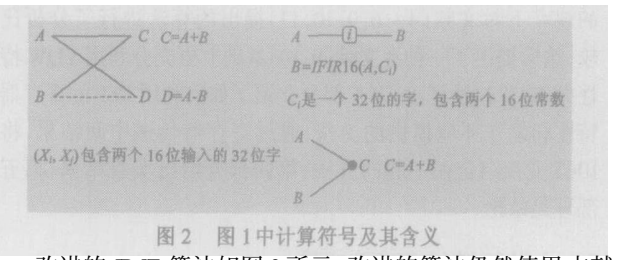

图2 图1中计算符号及其含义

改进的IDCT算法如图3所示,改进的算法仍然使用文献[9]中提到的定点运算,只是文献[9]中是将所有常数C_i放大$2^{14}\sqrt{2}$倍,改进算法中将所有常数C_i放大2^{15}倍以提高精度(与文献[10]相同).放大2^{15}倍的原因在于被放大前的常数C_i都小于1并有部分常数大于1/2.基于此原因,文献[9,10]算法中DCT的算法实际上也可以将所有常数C_i放大2^{15}倍以提高精度.使用CCITT(ITU-T)建议的方法在10,000次8×8IDCT实验中发现放大2^{15}倍与放大$2^{14}\sqrt{2}$倍相比,平均MSE降低约0.005.详细的实验方法参见文献[18]第3节,在文献[18]中提出的算法平均MSE比文献[4]降低0.003.

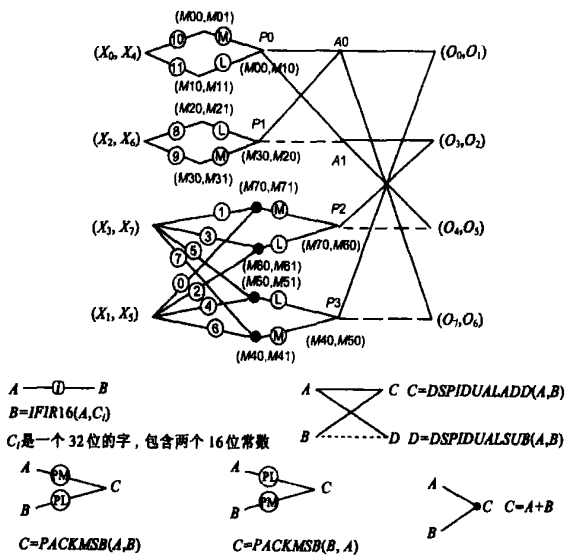

图3 一种改进的IDCT算法

下面以输出(O_0, O_1)中的O_0为例来说明改进算法的过程,其他值可以类推.由IDCT的定义:

$$F_i = \sqrt{\frac{2}{N}}\sum_{v=0}^{N-1} C_i x_v \cos\left[\frac{(2i+1)\pi v}{2N}\right] \quad (1)$$

其中$C_0 = 1/\sqrt{N}$; $C_n = \sqrt{2/N}$, $n \neq 0$.

将$N=8$, $i=0$代入式(1),可得:

$$2^{*}\ F_0 = \sum_{v=0}^{7} C_i x_v \cos\left(\frac{\pi v}{16}\right)$$
$$= 1/\sqrt{2}x_0\cos(0) + x_1\cos(\pi/16) + x_2\cos(2/16) + x_3\cos(3/16) + x_4\cos(4/16) + x_5\cos(5/16) + x_6\cos(6/16) + x_7\cos(7/16)$$

设

$$m0 = 1/\sqrt{2}x_0\cos(0) + x_4\cos(4/16) = 1/\sqrt{2}(x_0 + x_4)$$

$$m3 = x_2\cos(2/16) + x_6\cos(6/16)$$

$$m7 = x_1\cos(2\pi/16) + x_5\cos(5/16) + x_3\cos(3/16) + x_7\cos(7/16)$$

则

$$F_0 = (m0 + m3 + m7)/2 \quad (2)$$

表 2　与输出 O_0 有关的 C_i 的含义

C_i	10	9	6	7
高 16 位	$(1/\sqrt{2}) * D(0)$	$D(2)$	$D(1)$	$D(3)$
低 16 位	$(1/\sqrt{2}) * D(0)$	$D(6)$	$D(5)$	$D(7)$

设 $D(v) = 2^{15} * \cos(v/16)$, 图 3 中 C_i 的含义如表 2, 例如 C_9 中高 16 位是 $D(2)$, 用于与 x_2 相乘, 低 16 位是 $D(6)$, 用于与 x_6 相乘. 由图 3 的改进算法可得:

$$M0 = 1/\sqrt{2}[x_0 * D(0) + x_4 * D(0)]$$

$$M3 = x_2 * D(2) + x_6 * D(6)$$

$$M7 = (x_1 * D(1) + x_5 * D(5)) + (x_3 * D(3) + x_7 * D(7))$$

$P0$ 的高 16 位为 $M0$ 取高 16 位 $\approx m0/2$ (3)

$P1$ 的高 16 位为 $M3$ 取高 16 位 $\approx m3/2$ (4)

$P3$ 的高 16 位为 $M7$ 取高 16 位 $\approx m7/2$ (5)

$A0$ 的高 16 位 $\approx m0 + m3$

(O_0, O_1) 的高 16 位(即 O_0) $\approx (m0 + m3 + m7)/2$

对于式(3), 由于余弦系数 $D(i)$ 被放大 2^{15} 倍, $M0$ 的高 16 位是 $m0/2$ 的整数部分, 低 16 位是小数部分, 可以认为高 16 位与 $m0/2$ 近似相等, 同理可得式(4)、(5). 所以算法输出的 (O_0, O_1) 的高 16 位(即 O_0) 与由 IDCT 公式所得的式(2) 是近似相等的. 如果需要增加精度, 可以使用缩放. 即通过位移操作将输入数据放大 n(n 为 2 的幂) 倍, 然后将计算后的输出数据通过位移操作缩小 n 倍, 只是这样对输入数据 x_i 大小就会增加限制: $-2^{16}/n < x_i < (2^{16} - 1)/n$.

表 3　算法所需 VLIW 操作数比较

(算法 1 为文献[9, 10]中提到的, 2 为改进

算法 iadd/sub: 普通 32 位 SISD 加减法指令)

算法	iadd&isub	packmsb	ifir	dspidual add/sub	总计
1	16	0	12	0	28
2	4	4	12	6	26

$C = (C0, C1)$　$A = (A0, A1)$　$B = (B0, B1)$

$A0$, $B0$, $C0$: 分别为 A, B, C 的低 16 位

$A1$, $B1$, $C1$: 分别为 A, B, C 的高 16 位

$C = DSPIDUALSUB(A, B)$ $C0 = A0 - B0$ $C1 = A1 - B1$	$C = DSPIDUALADD(A, B)$ $C0 = A0 + B0$ $C1 = A1 + B1$
$C = PACK16MSB(A, B)$ $C0 = A0$ $C1 = B0$	$C = IFIR16(A, B)$ $C = A0 \times B0 + A1 \times B1$

图 4　图 3 中所使用的 VLIW 指令说明

与文献[9]中的算法的比较如表 3. 各指令的含义可以参考图 4 和文献[9, 10], 它更多地利用了特有的 DSP 指令以减少加减法次数. 值得指出的是, 如果输出是以 16 位的方式存放的话, 那么(O_0, O_1) 和(O_4, O_5) 已经是所需要的形式, 而其他两个只要经过循环移位操作即可, 所以如果需要存储的话, 和原算法相比, 每个 8 点运算可以减少 2 次为存入内存所需的 packmsb 操作. 与文献[9]相比, 不需要进行缩放, 可以节省操作.

文献[9, 10]中 DCT 的加减法运算使用的是 SIMD 多媒体指令 dspidualadd/sub, 而其 IDCT 的加减法运算使用的是 SISD 指令 iadd/sub. 改进算法与文献[9, 10]提到的 DCT 算法使用 SIMD 指令的算法思想对应性更强. 而从数据流图的拓扑结构上看, 改进算法更像是文献[9, 10]中 DCT 算法数据流图的倒序排列, 对应性更强.

2.4　改进的算法在 8* 8 的二维 IDCT 中的实现

利用二维 IDCT 变换的分离性和改进的 8 点 IDCT 算法, 我们可以采取行列分离的方法(row-column method) 来实现 8* 8 二维 IDCT 的计算. 在行变换前, 通过 packmsb 以及 packlsb 操作来组织数据. 因为由两个 16 位的输入组成的输入对下标相隔都为 4, 可以将第 i 行和第 $i+4$ 行同时进行 IDCT 变换, 使用 packmsb 以及 packlsb 合并第 i 行和第 $i+4$ 行($i = 0, 1, 2, 3$) 数据经过行变换后的结果. 其中, 第 i 行的第 j 个输出与 $i+4$ 行的第 j 个输出进行合并.

在数据保存时, 若需要保存的是 16 位数据, 那么, 在列变换后的结果中, (O_3, O_2) 和(O_7, O_6) 需要经过交换, 使之成为 (O_2, O_3)、(O_7, O_6). 而(O_0, O_1)、(O_4, O_5) 已经是所需要的形式, 从而节省了附加操作.

3　利用寄存器特性将运动补偿与 IDCT(DCT) 结合的方法

在常用的混合编码(Hybrid Coding) 的解码过程(如 MPEG1/2/4 和 H.264) 中, 对于预测帧(P, B 帧), IDCT 变换以后紧跟着就是运动补偿; 编码过程中刚好相反, 先是运动补偿, 紧接着是 DCT 变换. 运动补偿中使用的数据一部分来自被预测的帧, 另一部分来自 IDCT 变换后的输出. 由于读写内存占操作数, 并且容易产生 nop, 如果使用通常的方法, IDCT 变换后的数据存入内存, 然后运动补偿时把数据从内存读出, 不仅会增加操作数, 增加内存读写时间, 还可能会增加更多的 nop.

实际上, 在寄存器数目足够多的情况下(如 TM1300 有 128 个寄存器), 可以选用其中合适数目的寄存器来进行参数传递, 以减少内存操作, 从而达到提高效率的目的. 下面以第二部分的改进算法与运动补偿组合为例, 介绍怎样实现通过寄存器传递中间结果, 将运动补偿与 DCT 组合可以使用相同思想.

3.1　将运动补偿与第二部分的改进算法组合

由于第二部分输出是以 16 位表示一个数据, 对于 8* 8 的变换, 共有 64 个数据, 以下假设寄存器数据字长为 32 位, 那么共需要 32 个寄存器.

其方法简述如下:

IDCT_and_MC

输入:"需要被 *IDCT* 变换的数据","被预测帧数据".

(1) 将“需要被 IDCT 变换的数据”IDCT 变换,将 IDCT 变换结果存于 32 个“暂存 IDCT 输出的寄存器”中.

(2) 利用上述 32 个“暂存 IDCT 输出的寄存器”,结合“被预测帧数据”,进行运动补偿.

输出:运动补偿后数据

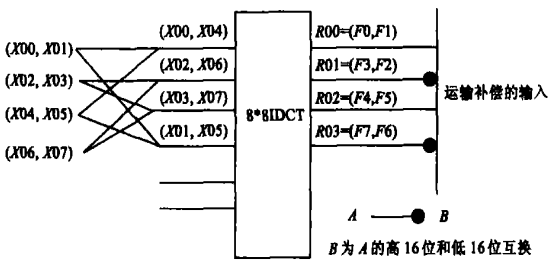

图 5 运动补偿与 IDCT 结合的数据流图

图 5 中以输入的第 0 行数据为例说明了整个过程. 其中, X_{ij}是第 i 行 j 列的输入数据, F_j 是第 0 行第 j 列的输出数据. R00, R01, R02, R03 可以认为是暂存于寄存器中.

4 实验结果及分析

使用两种方式在 Phillips 公司的 TM1300 开发工具 SDE2.2 的仿真环境上检验实验结果:

第一种是使用 Phillips 公司提供的编译器,用编译器编译和优化 C 语言写的程序,察看汇编代码(.s 文件),计算该算法对应的函数所用的指令周期(cycle)数. 由于函数中使用循环打开(loop unroll),程序顺序执行,所以计算是很方便的.

第二种方法是使用文献[9]中介绍的仿真模拟环境. 原算法和改进算法都是使用了相同的基于 VLIW 特性的其他优化后(如循环展开,考虑 cache 等)的结果.

4.1 使用第一种方法的实验结果和分析

为了方便重现,使用的编程环境中的指令为:

tmcc - S IDCT. c

只考虑第二部分提出的 IDCT 算法,文献[10]中提到,使用缩放的 IDCT 最少需要 160 - 170cycle,根据可以查到的文献实现的算法,最低能达到 163Cycle,其中有 52 个 nop. 使用缩放,对行和列变换都使用本文改进后的算法的话,能达到 157 Cycle,其中有 74nop. 不使用缩放为 154Cycle,其中 88nop. 使用缩放的改进算法需要的 Cycle 数为原算法的 96.31%;不使用缩放为原算法的 94.48%.

仔细分析使用缩放的 IDCT 算法,每 8 点行或列运算减少 2 条指令(见表 3),8* 8 的二维 IDCT 可以减少 32 条,同时还可以减少为写内存而对输出数据处理进行的 packmsb 操作 2 * 8= 16 个(见 2.4),共有 48 个操作被减少. 编译后的汇编级代码减少 6Cycle,增加 22nop,所以减少 6* 5+ 22= 52 个操作,数目基本是相同的.

有趣的是,如果只对行变换使用本文改进的算法,使用缩放能达到 157Cycle,38nop;不使用缩放能达到 152Cycle,46nop,与行列都使用改进算法相比,需要相同甚至更少的 Cycle 数. 其原因在于列变换后,使数据保持在[0 - 255]的操作与使图 4 中 R01, R03 等的高低 16 位互换的操作在指令操作位上有一定的互斥关系. 在 TM1500 上这个问题已经解决,因为高低 16 位互换操作从原来分配到 2 个处理单元(表 1 中的 shifter)改为分配到 5 个处理单元.

运动补偿和 IDCT 结合,IDCT 行列变换都使用改进的算法后,使用缩放需要 185Cycle,其中 57nop,nop 数比只使用第二种方法少,其原因在于运动补偿中的操作插入到了 IDCT 的 nop 中. 不使用缩放需要 179Cycle,其中 47nop.

4.2 使用第二种方法的实验结果和分析

为了方便重现,使用的编程环境中的指令为:

tmcc -G -O3 IDCT_and_MC.c

(IDCT_and_MC.c 为包含该函数和主程序的文件)

tmsimr ns statfile a.stat a.out

tmprof scale -f func a.stat a.out

表 4 中是关于 IDCT 算法(使用缩放)的比较. 表 5 中是运动补偿和 IDCT(使用缩放)组合的数据. 数据与文献[9]中不同的原因之一可能是本文的统计只看单个函数,文献[9]中实验可能对包括一些不必要的其他函数的时间进行了统计;另一种可能是计量时使用的度量单位不一样. 第二部分改进算法所需时间为原算法的 97.88%,第三部分改进算法约为原算法的 80%,单从运动补偿来看,所需为:2019 - 1576= 443,为原来(913)的 48.52%.

表 4 IDCT 算法的比较

文献[9,10]中的算法		改进后第二部分的算法	
Function	Total Cycles	Function	Total Cycles
IDCT	1610	IDCT	1576

表 5 第三部分算法的比较

运动补偿未与 IDCT 结合		结合后	
Function	Total Cycles	Function	Total Cycles
IDCT	1610	IDCT_and_MC	2019
MC	913		

5 结论

文章给出了文献[6,9,10]提出的算法的矩阵分解形式,在 VLIW 的观点下将经典算法文献[4~6,17]与文献[9,10]提出的算法进行了比较,提出了对文献[9,10]中 IDCT 的改进算法,并将改进算法与运动补偿结合达到提高效率的目的. 由实验结果可知,由于合理使用 DSP 指令,使得总操作数减少,适合于 VLIW 需要并行度和操作数综合考虑的特点,改进的 IDCT 算法效率得到提高. 第三部分中利用 DSP 寄存器特性,使运动补偿所需时间减少为原来的约 50%. 这种方法有广泛适用性,可以用于编码过程中将运动预测与 DCT 结合,以及其他一些上下功能中参数传递紧密,参数数目合适于 DSP 寄存器数目的过程. 该算法能用于 MPEG1/2/4,虽然在传递的参数数目不同,第三部分思想同样可以用于 H.264,第二部分对于 H.264 的 4* 4DCT 和 IDCT 同样有借鉴作用.

通过矩阵分解,可以使用矩阵理论进一步提高 DCT 算法的性能. 在进一步的研究中,本文给出的矩阵分解提供了很好的基础.

参考文献:

[1] A Elnaggar, H M Alnuweiri. A new multidimensional recursive architecture for computing the discrete cosine transform[J]. IEEE Trans on Circuits and Systems for Video Technology, 2000, 10(1): 113- 119.

[2] Wang Z S, He Z Y, et al. A generalized fast algorithm for 2-D discrete cosine transform and its application to motion picture coding[J]. IEEE Trans on Circuits and Systems II, 1999, 46(5): 617- 627.

[3] Chen Xinjian, Dai Qionghai, Li Chunwen. A fast algorithm for computing multidimensional DCT on certain small sizes[J]. IEEE Trans on Signal Processing, 2003, 51(1): 213- 220.

[4] Byeong Lee. A new algorithm to compute the discrete cosine transform[J]. IEEE Trans on Acoustics, Speech and Signal Processing, 1984, ASSP232(6): 1243- 1245.

[5] H S Hou. A fast recursive algorithm for computing the discrete cosine transform[J]. IEEE Trans on Acoustics, Speech and Signal Processing, 1987, ASSP235(10): 1455 - 1461.

[6] Christoph Loeffler, et al. Practical fast 1-D DCT algorithms with 11 multiplications[J]. Acoustics, Speech, and Signal Processing, 1989, ASSP289(12): 988- 991.

[7] Takala J, Nikara J, et al. Pipeline architecture for 8×8 discrete cosine transform[A]. Proceedings IEEE Inc on Acoustics, Speech, and Signal Processing[C]. Istanbul: IEEE, 2000, (6): 3303- 3306.

[8] Nam Ik Cho, Sang Uk Lee. DCT algorithms for VLSI parallel implementations[J]. IEEE Trans on Acoustics, Speech, and Signal Processing, 1990, 38(1): 121- 127.

[9] 李学明, 李继. 用超长指令实现 DCT 的新算法[J]. 电子学报, 2003, 31(7): 1074- 1077.

[10] Trimedia Technology Inc. Philips TriMedia™ SDE Documentation 2.2, Book 2, Book4 & TriMedia Databooks[M/CD]. 2000.

[11] Sohm O P, Bull D R, Canagarajah C N. Fast 2D-DCT implementations for VLIW processors[A]. IEEE 3rd Workshop on Multimedia Signal Processing[C]. Copenhagen: IEEE, 1999. 655- 660.

[12] Tian Sheuan Chang, Jiun In Guo, Cheir Wei Jen. A compact IDCT processor for HDTV applications[A]. IEEE Workshop on Signal Processing Systems[C]. Taipei: IEEE, 1999. 151- 158.

[13] Henning R, Chakrabarti C. A quality/energy tradeoff approach for IDCT computation in MPEG-2 video decoding[A]. IEEE Workshop on Signal Processing Systems[C]. Lafayette, LA: IEEE, 2000. 90- 99.

[14] Murata E, Ikekawa M, Kuroda C. Fast 2D IDCT implementation with multimedia instructions for a software MPEG2 decoder[A]. Proceedings IEEE Inc on Acoustics, Speech, and Signal Processing[C]. Seattle, WA: IEEE, 1998, (5): 3105- 3108.

[15] 王业奎, 涂国防. 基于块截断编码和运动补偿的纯软件视频编解码算法[J]. 电子学报, 2001, (2): 275- 278.

[16] Schutten R J, De Haan G. Real-time 2-3 pull-down elimination applying motion estimation/compensation in a programmable device[J]. IEEE Trans on Consumer Electronics, 1998, 44(3): 930- 938.

[17] Feig E, Winograd S. Fast algorithms for the discrete cosine transform[J]. IEEE Trans on Signal Processing, 1992, 40(9): 2174- 2193.

[18] Yeonsik J, Imgeun L, Taekhyun Y, Gooman P, Kyu T P. A fast algorithm suitable for DCT implementation with integer multiplication[A]. Proceedings of Digital Processing Applications (TENCON '96)[C]. Perth, WA, Australia: IEEE, 1996, (2): 784- 787.

作者简介:

欧阳万里　男, 1980 年出生于湖南省, 北京工业大学硕士研究生, 主要研究方向为视频和图像处理, 以及并行计算.
E-mail: wanli210@emails.bjut.edu.cn.

肖创柏　男, 1962 年出生于湖南省, 博士, 北京工业大学教授, 主要从事数字信号处理, 模式识别方向的研究.

刘　广　男, 1980 年出生于河北省, 北京工业大学硕士研究生, 如要研究方向为视频, 图像处理, 音视频通信.