一种针对 H.264 的高性能反变换结构

方 健 郑 伟 王 匡 李炳博
(浙江大学 信息与电子工程学系,杭州 310027)

摘 要 H.264 的整数变换存在 8×8 和 4×4 两种尺寸大小,这增加了硬件设计的复杂性。同时,高清视频应用要求解码器具有更强的处理能力。针对这两个问题,文章提出了一种高性能的反变换硬件实现结构。对于 4×4 整数反变换,重构一个 8×8 亚宏块的 4 个 4×4 块,从而使两种尺寸大小的整数反变换具有相同的结构。采用优化的数据存储和流水设计,行列反变换能够同时执行,处理一个 8×8 亚宏块平均只需要 32 个时钟周期。转置存储器采用一个 32×32bits 的双口 SRAM 和 8 组寄存器组实现,和以前的设计相比,可以节省 53.7% 的存储器面积。在 108 MHz 工作频率下,本文提出的硬件结构能够有效执行 H.264 高清实时解码的反变换运算。

关键词 视频解码 H.264 整数反变换
中图法分类号: TN919.81 **文献标识码**: A **文章编号**: 1006-8961(2009)02-0275-06

A High Performance Inverse Integer Transform Architecture for H.264

FANG Jian, ZHENG Wei, WANG Kuang, LI Bing-bo
(*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou* 310027)

Abstract There are two kinds of integer transform in H.264, 8×8 integer transform and 4×4 integer transform. This makes hardware design more complex. At the same time, more powerful decoder is required for high definition video application. High performance hardware architecture is proposed for 2D inverse integer transform in H.264. For 4×4 inverse integer transform, four 4×4 blocks in an 8×8 sub-macroblock was reconstructed. Therefore, the 8×8 2D inverse integer transform and 4×4 2D inverse integer transform could have the same architecture. With a new strategy of data storage and pipeline, inverse transform for column data and inverse transform for row data could perform at the same time. On average, 32 clocks were needed for processing an 8×8 sub-macroblock. The transpose memory was composed of a two-port 32×32bits SRAM and 8 groups of registers. Compared with former design, the new architecture could reduce 53.7% area of transpose memory. When clocked at 108 MHz, the proposed design can perform real-time inverse transform for high definition video decoder of H.264.

Keywords video decode, H.264, inverse integer transform(IIT)

1 引 言

新一代的视频压缩标准 H.264/AVC 相比于以前的标准,压缩率提高了 50% 左右,而且具有良好的网络亲和性[1],在 HDTV、IPTV、视频监控和网络会议等领域具有广泛的应用前景。

H.264 高压缩性能的其中一个原因是采用了自适应块大小的变换技术。H.264 采用 8×8 和 4×4 两种块大小进行整数变换(IT)和整数反变换(IIT),平坦区域采用大尺寸变换,细节多的区域采用小尺寸变换,从而减小了变换域系数的冗余度,提高了压缩率。但是,自适应块大小变换技术带来高压缩率的同时,也增加了解码器的实现复杂度。如果简单采用 8×8 和 4×4 两套变换结构,将大大增加电路规模。参考文献[2]的 DCT/IDCT 快速算法,对 4×

基金项目:国家自然科学基金项目(90307002)
收稿日期:2007-01-23;**改回日期**:2007-08-14
第一作者简介:方健(1980~),男。2008 年于浙江大学获通信信息工程博士学位。研究方向为视频压缩。E-mail:fangjian1980@163.com

4IIT 进行了 8×8 重构,提出了统一结构的整数反变换结构,这有利于硬件解码器的规则化设计。

另一方面,H.264 的高压缩率使它特别适用于高清视频应用领域。高清图像分辨率高,数据吞吐量大,虽然传统的 IDCT 实现结构[3]同样适用于整数反变换,但是它不能满足高清视频解码的处理速度。一种简单的解决办法是采用多个变换核[4],但是这样会大大增加硬件面积。在文献[5]的设计基础上,提出了一种适用于 H.264 反变换的流水结构,它的处理速度能够满足高清解码的需要,同时有效地节省了硬件面积。

2 基于 8×8 块的整数反变换

离散余弦变换和反变换的浮点运算精度不一致,会造成解码数据失配现象。H.264 对 DCT 变换进行了整型化处理——整数变换,消除了编解码的失配现象,同时有效地减少了计算量。为了提高压缩率,H.264 采用了 4×4 和 8×8 两种块大小的整数变换。在解码端,压缩数据经过熵解码,DC 系数 Hadamard 变换和反量化运算得到整数变换域系数,再通过整数反变换恢复得到图像残差数据。本文就是针对整数反变换运算进行的优化设计。

4×4 整数反变化如式(1);8×8 变换的 IIT 变化如式(2)。其中 $\boldsymbol{Y}_{4\times4}$ 为反变换系数矩阵,$\boldsymbol{X}_{4\times4}$ 为反量化系数矩阵,$\boldsymbol{C}_{4\times4}$ 为 4×4 变换矩阵,$\boldsymbol{C}_{8\times8}$ 为 8×8 变换矩阵,'T'为矩阵转置运算。

$$\boldsymbol{Y}_{4\times4}=\boldsymbol{C}_{4\times4}\boldsymbol{X}_{4\times4}\boldsymbol{C}_{4\times4}^{\mathrm{T}};\boldsymbol{C}_{4\times4}=\begin{bmatrix}1 & 1 & 1 & \frac{1}{2}\\ 1 & \frac{1}{2} & -1 & -1\\ 1 & -\frac{1}{2} & -1 & 1\\ 1 & -1 & 1 & -\frac{1}{2}\end{bmatrix} \tag{1}$$

$$\boldsymbol{Y}_{8\times8}=\boldsymbol{C}_{8\times8}\boldsymbol{X}_{8\times8}\boldsymbol{C}_{8\times8}^{\mathrm{T}};\boldsymbol{C}_{8\times8}=\begin{bmatrix}1 & \frac{3}{2} & 1 & \frac{5}{4} & 1 & \frac{3}{4} & \frac{1}{2} & \frac{3}{8}\\ 1 & \frac{5}{4} & \frac{1}{2} & -\frac{3}{8} & -1 & -\frac{3}{2} & -1 & -\frac{3}{4}\\ 1 & \frac{3}{4} & -\frac{1}{2} & -\frac{3}{2} & -1 & \frac{3}{8} & 1 & \frac{5}{4}\\ 1 & \frac{3}{8} & -1 & -\frac{3}{4} & 1 & \frac{5}{4} & -\frac{1}{2} & -\frac{3}{2}\\ 1 & -\frac{3}{8} & -1 & \frac{3}{4} & 1 & -\frac{5}{4} & -\frac{1}{2} & \frac{3}{2}\\ 1 & -\frac{3}{4} & -\frac{1}{2} & \frac{3}{2} & -1 & -\frac{3}{8} & 1 & -\frac{5}{4}\\ 1 & -\frac{5}{4} & \frac{1}{2} & \frac{3}{8} & -1 & \frac{3}{2} & -1 & \frac{3}{4}\\ 1 & -\frac{3}{2} & 1 & -\frac{5}{4} & 1 & -\frac{3}{4} & \frac{1}{2} & -\frac{3}{8}\end{bmatrix} \tag{2}$$

可以发现,$\boldsymbol{C}_{4\times4}$ 和 $\boldsymbol{C}_{8\times8}$ 具有类似的特性:偶数列(0,2,4,6)偶对称,奇数列(1,3,5,7)奇对称。通过奇偶分解可以简化运算,但是两种不同的运算尺寸不利于硬件实现。为了能够采用相同的反变换结构,需要将一个亚宏块的 4 个 4×4 块重构为一个 8×8块进行整数反变换。

令 8×8 亚宏块反变换系数矩阵为 $\boldsymbol{Y}_{8\times8}=\begin{bmatrix}\boldsymbol{Y}_{00} & \boldsymbol{Y}_{01}\\ \boldsymbol{Y}_{10} & \boldsymbol{Y}_{11}\end{bmatrix}$,对应的反量化系数矩阵为 $\boldsymbol{X}_{8\times8}=\begin{bmatrix}\boldsymbol{X}_{00} & \boldsymbol{X}_{01}\\ \boldsymbol{X}_{10} & \boldsymbol{X}_{11}\end{bmatrix}$,其中 $\boldsymbol{Y}_{00}=\boldsymbol{C}_{4\times4}\boldsymbol{X}_{00}\boldsymbol{C}_{4\times4}^{\mathrm{T}}$,$\boldsymbol{Y}_{01}=\boldsymbol{C}_{4\times4}\boldsymbol{X}_{01}\boldsymbol{C}_{4\times4}^{\mathrm{T}}$,$\boldsymbol{Y}_{10}=\boldsymbol{C}_{4\times4}\boldsymbol{X}_{10}\boldsymbol{C}_{4\times4}^{\mathrm{T}}$,$\boldsymbol{Y}_{11}=\boldsymbol{C}_{4\times4}\boldsymbol{X}_{11}\boldsymbol{C}_{4\times4}^{\mathrm{T}}$。$\boldsymbol{Y}_{ij}(i,j=0,1)$ 分别是亚宏块 4 个 4×4 块的反变换系数矩阵,$\boldsymbol{X}_{ij}(i,j=0,1)$ 分别是 4 个 4×4 块的反量化系数矩阵。由此得到下式

$$\begin{aligned}\boldsymbol{Y}_{8\times8}&=\begin{bmatrix}\boldsymbol{C}_{4\times4}\boldsymbol{X}_{00}\boldsymbol{C}_{4\times4}^{\mathrm{T}} & \boldsymbol{C}_{4\times4}\boldsymbol{X}_{01}\boldsymbol{C}_{4\times4}^{\mathrm{T}}\\ \boldsymbol{C}_{4\times4}\boldsymbol{X}_{10}\boldsymbol{C}_{4\times4}^{\mathrm{T}} & \boldsymbol{C}_{4\times4}\boldsymbol{X}_{11}\boldsymbol{C}_{4\times4}^{\mathrm{T}}\end{bmatrix}\\ &=\begin{bmatrix}\boldsymbol{C}_{4\times4} & [0]_{4\times4}\\ [0]_{4\times4} & \boldsymbol{C}_{4\times4}\end{bmatrix}\times\begin{bmatrix}\boldsymbol{X}_{00} & \boldsymbol{X}_{01}\\ \boldsymbol{X}_{10} & \boldsymbol{X}_{11}\end{bmatrix}\times\\ &\quad\begin{bmatrix}\boldsymbol{C}_{4\times4}^{\mathrm{T}} & [0]_{4\times4}\\ [0]_{4\times4} & \boldsymbol{C}_{4\times4}^{\mathrm{T}}\end{bmatrix}\end{aligned} \tag{3}$$

其中 $[0]_{4\times4}$ 为 4×4 零矩阵。

H.264 的 8×8 反变换记做 8×8 IIT,4×4 重

构成 8×8 的整数反变换记做 4×4 IIT。这样,反变换可以统一成式(4)。其中 $\boldsymbol{H}$ 为反变换矩阵,分别对应式(2)中的 $\boldsymbol{C}_{8\times8}$ 矩阵和式(3)中的 $\boldsymbol{D}_{8\times8}$ 矩阵。

$$\boldsymbol{Y}_{8\times8}=\boldsymbol{D}_{8\times8}\boldsymbol{X}_{8\times8}\boldsymbol{D}_{8\times8}^{\mathrm{T}};\boldsymbol{D}_{8\times8}=$$

$$\begin{bmatrix}1&1&1&\frac{1}{2}&0&0&0&0\\1&\frac{1}{2}&-1&-1&0&0&0&0\\1&-\frac{1}{2}&-1&1&0&0&0&0\\1&-1&1&\frac{1}{2}&0&0&0&0\\0&0&0&0&1&1&1&\frac{1}{2}\\0&0&0&0&1&\frac{1}{2}&-1&-1\\0&0&0&0&1&-\frac{1}{2}&-1&1\\0&0&0&0&1&-1&1&-\frac{1}{2}\end{bmatrix}$$

$$\boldsymbol{Y}_{8\times8}=\boldsymbol{H}\boldsymbol{X}\boldsymbol{H}^{\mathrm{T}}\tag{4}$$

直接计算 2 维 DCT 反变换,运算量大,结构不规则,不利于硬件实现。采用文献[2]的快速算法,将 2 维 8×8 反变换分解为行列两个方向依次进行的 1 维反变换,如式(5)。这样先进行 1 维列变换,将计算结果转置后再进行 1 维行变换,就可以完成 2 维 8×8 IIT。而且行变换和列变换可以复用相同的结构。

$$\boldsymbol{Y}_{8\times8}=\boldsymbol{H}\boldsymbol{X}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{H}(\boldsymbol{H}\boldsymbol{X}^{\mathrm{T}})^{\mathrm{T}}\tag{5}$$

进一步考虑,1 维 8×8 反变换还可以利用变换矩阵的奇偶性分解为两个 4×4 矩阵乘法,加快运算速度,减小硬件面积。8×8 IIT 的分解结构如式(6)。

$$\begin{bmatrix}\boldsymbol{Y}_0\\\boldsymbol{Y}_1\\\boldsymbol{Y}_2\\\boldsymbol{Y}_3\end{bmatrix}=\begin{bmatrix}a&c&a&f\\a&f&-a&-c\\a&-f&-a&c\\a&-c&a&-f\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_0\\\boldsymbol{X}_2\\\boldsymbol{X}_4\\\boldsymbol{X}_6\end{bmatrix}+\begin{bmatrix}b&d&e&g\\d&-g&-b&-e\\e&-b&g&d\\g&-e&d&-b\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_1\\\boldsymbol{X}_3\\\boldsymbol{X}_5\\\boldsymbol{X}_7\end{bmatrix}$$

$$\begin{bmatrix}\boldsymbol{Y}_7\\\boldsymbol{Y}_6\\\boldsymbol{Y}_5\\\boldsymbol{Y}_4\end{bmatrix}=\begin{bmatrix}a&c&a&f\\a&f&-a&-c\\a&-f&-a&c\\a&-c&a&-f\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_0\\\boldsymbol{X}_2\\\boldsymbol{X}_4\\\boldsymbol{X}_6\end{bmatrix}-\begin{bmatrix}b&d&e&g\\d&-g&-b&-e\\e&-b&g&d\\g&-e&d&-b\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_1\\\boldsymbol{X}_3\\\boldsymbol{X}_5\\\boldsymbol{X}_7\end{bmatrix}\tag{6}$$

式中,$[a\ b\ c\ d\ e\ f\ g]=\left[1\ \frac{3}{2}\ 1\ \frac{5}{4}\ \frac{3}{4}\ \frac{1}{2}\ \frac{3}{8}\right]$

4×4 IIT 变换矩阵的奇偶特性略有不同,分解结构如式(7)。反量化系数进行相同的奇偶分解,只是变换系数和结果的位置顺序不同。但是在选择乘数和存储结果时做出相应的变化就可以采用相同的运算结构。

$$\begin{bmatrix}\boldsymbol{Z}_0\\\boldsymbol{Z}_1\\\boldsymbol{Z}_4\\\boldsymbol{Z}_5\end{bmatrix}=\begin{bmatrix}a&a&0&0\\a&-a&0&0\\0&0&a&a\\0&0&a&-a\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_0\\\boldsymbol{X}_2\\\boldsymbol{X}_4\\\boldsymbol{X}_6\end{bmatrix}+\begin{bmatrix}b&d&0&0\\d&-b&0&0\\0&0&b&d\\0&0&d&-b\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_1\\\boldsymbol{X}_3\\\boldsymbol{X}_5\\\boldsymbol{X}_7\end{bmatrix}$$

$$\begin{bmatrix}\boldsymbol{Z}_3\\\boldsymbol{Z}_2\\\boldsymbol{Z}_7\\\boldsymbol{Z}_6\end{bmatrix}=\begin{bmatrix}a&a&0&0\\a&-a&0&0\\0&0&a&a\\0&0&a&-a\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_0\\\boldsymbol{X}_2\\\boldsymbol{X}_4\\\boldsymbol{X}_6\end{bmatrix}-\begin{bmatrix}b&d&0&0\\d&-b&0&0\\0&0&b&d\\0&0&d&-b\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_1\\\boldsymbol{X}_3\\\boldsymbol{X}_5\\\boldsymbol{X}_7\end{bmatrix}\tag{7}$$

式中,$a=1$,$b=1$,$d=\frac{1}{2}$。

提出的反变换结构就是基于式(5)~式(7)的硬件实行方案。

3　硬件结构

H.264 视频解码器反变换硬件结构如图 1。该结构包括偶序数据处理(EODP)单元,奇序数据处理(OODP)单元,转置存储器 TM 和数据重排(DRO)模块。

其中 EODP0、OODP0 和两个后端加法器实现列变换运算,EODP1、OODP1 和两个后端加法器实现

行变换运算。X_{ec}和X_{oc}分别表示列数据的偶序输入(X_{0n},X_{2n},X_{4n},X_{6n})和奇序输入(X_{1n},X_{3n},X_{5n},X_{7n}),其中$0 \leqslant n \leqslant 7$,代表8×8块的列坐标;$Z_{ac}$和$Z_{sc}$分别表示列变换后偶序与奇序数据的求和运算与求差运算结果,存储到转置矩阵 ***TM***;X_{er}和X_{or}分别表示行数据的偶序输入(X_{m0},X_{m2},X_{m4},X_{m6})和奇序输入(X_{m1},X_{m3},X_{m5},X_{m7}),其中$0 \leqslant m \leqslant 7$,代表8×8块的行坐标;$Z_{ar}$和$Z_{sr}$分别表示行变换后偶序与奇序数据的求和运算与求差运算结果。Y_e和Y_o分别表示2维反变换运算结果的列数据的偶序(Y_{0n},Y_{2n},Y_{4n},Y_{6n})和奇序(Y_{1n},Y_{3n},Y_{5n},Y_{7n})。这样,两点反量化列数据输入,两点反变换列数据输出,行列变换流水执行。

图1中的EODP和OODP分别实现偶序数据和奇序数据的4×1矩阵运算。硬件结构如图2和图3,该结构已经根据式(6)和式(7)进行了优化。其中'×'表示乘法器;'>>'表示移位器,实际通过位选实现;'MUX'表示选择器组;'D'表示寄存器组;累加器由加法器和寄存器构成。X_e和X_o分别表示偶序输入和奇序输出,Z_e和Z_o分别表示对应4×1矩阵运算输出。可以发现,通过运算器复用,8×8 IIT和4×4 IIT结构实现了统一,有效减小了电路规模。

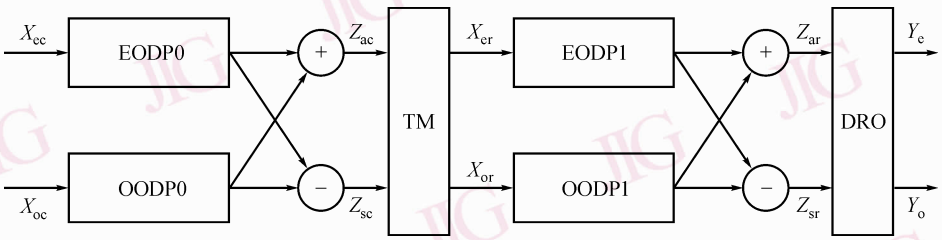

图1 反变换硬件结构

Fig. 1 Structure of inverse transform

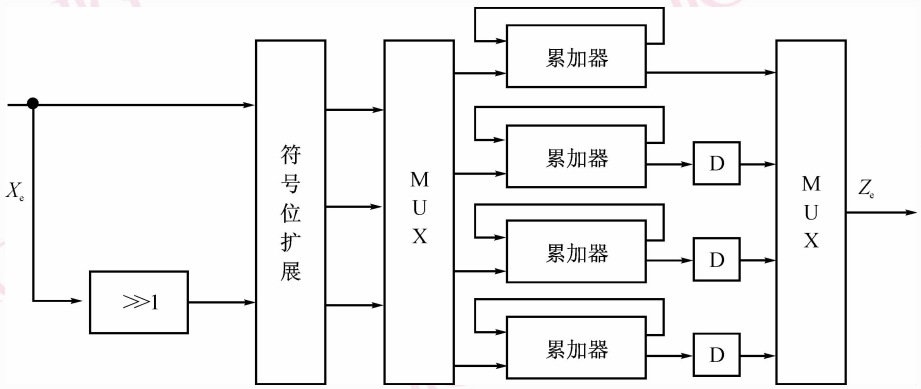

图2 EODP结构

Fig. 2 Structure of EODP

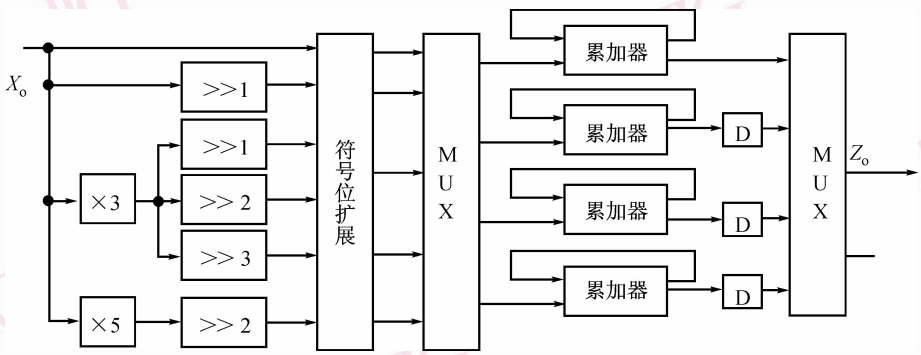

图3 OODP结构

Fig. 3 Structure of OODP

列变换	$A_{00}\ A_{20}\ A_{40}\ A_{60}$ $A_{10}\ A_{30}\ A_{50}\ A_{70}$	$A_{01}\ A_{21}\ A_{41}\ A_{61}$ $A_{11}\ A_{31}\ A_{51}\ A_{71}$	$A_{02}\ A_{22}\ A_{42}\ A_{62}$ $A_{12}\ A_{32}\ A_{52}\ A_{72}$	$A_{03}\ A_{23}\ A_{43}\ A_{63}$ $A_{13}\ A_{33}\ A_{53}\ A_{73}$	……	$A_{00}\ A_{20}\ A_{40}\ A_{60}$ $A_{10}\ A_{30}\ A_{50}\ A_{70}$	$A_{01}\ A_{21}\ A_{41}\ A_{61}$ $A_{11}\ A_{31}\ A_{51}\ A_{71}$	$A_{02}\ A_{22}\ A_{42}\ A_{62}$ $A_{12}\ A_{32}\ A_{52}\ A_{72}$
列变换重排寄存		$B_{00}\ B_{10}\ B_{20}\ B_{30}$ $B_{70}\ B_{60}\ B_{50}\ B_{40}$	$B_{71}\ B_{61}\ B_{51}\ B_{41}$ $B_{70}\ B_{60}\ B_{50}\ B_{40}$	$B_{00}\ B_{20}\ B_{40}\ B_{60}$ $B_{10}\ B_{30}\ B_{50}\ B_{70}$	……	$B_{77}\ B_{67}\ B_{57}\ B_{47}$ $B_{76}\ B_{66}\ B_{56}\ B_{46}$	$B_{00}\ B_{10}\ B_{20}\ B_{30}$ $B_{70}\ B_{60}\ B_{50}\ B_{40}$	$B_{71}\ B_{61}\ B_{51}\ B_{41}$ $B_{70}\ B_{60}\ B_{50}\ B_{40}$
转置存储			$B_{00}\ B_{10}\ B_{20}\ B_{30}$ $B_{01}\ B_{11}\ B_{21}\ B_{31}$	$B_{70}\ B_{60}\ B_{50}\ B_{40}$ $B_{71}\ B_{61}\ B_{51}\ B_{41}$	……	$B_{06}\ B_{16}\ B_{26}\ B_{36}$ $B_{07}\ B_{17}\ B_{27}\ B_{37}$	$B_{76}\ B_{66}\ B_{56}\ B_{46}$ $B_{77}\ B_{67}\ B_{57}\ B_{47}$	$B_{00}\ B_{10}\ B_{20}\ B_{30}$ $B_{01}\ B_{11}\ B_{21}\ B_{31}$
行变换						$B_{00}\ B_{02}\ B_{04}\ B_{06}$ $B_{01}\ B_{03}\ B_{05}\ B_{07}$	$B_{10}\ B_{12}\ B_{14}\ B_{16}$ $B_{11}\ B_{13}\ B_{15}\ B_{17}$	$B_{20}\ B_{22}\ B_{24}\ B_{26}$ $B_{21}\ B_{23}\ B_{25}\ B_{27}$
行变换重排寄存							$C_{00}\ C_{01}\ C_{02}\ C_{03}$ $C_{07}\ C_{06}\ C_{05}\ C_{04}$	$C_{17}\ C_{16}\ C_{15}\ C_{14}$ $C_{07}\ C_{06}\ C_{05}\ C_{04}$
行变换输出存储								$C_{00}\ C_{01}\ C_{02}\ C_{03}$ $C_{10}\ C_{11}\ C_{12}\ C_{13}$

图 4　2 维反变换流水

Fig. 4　Pipeline of 2D inverse integer transform

矩阵运算结果 Z_e 和 Z_o 通过和差运算得到 1 维列变换结果,存储到 TM 进行行列转置。TM 由 8 个像素点的寄存器组和一片 32×32bits 双口 RAM 构成。整个 2 维反变换由 6 级流水构成,分别是列变换、列变换重排寄存、转置存储器存储、行变换、行变换重排寄存和行变换输出存储。图 4 以存储器件为界描述反变换流水处理的数据转移情况,同时也揭示了 TM 使用一片 RAM 和 8 组重排寄存器组实现行列转置的过程。其中,A_{mn} 代表列反变换输入,m(0~7)代表在 8×8 块中的行位置,n(0~7) 代表在 8×8 块中的列位置;B_{mn} 代表列变换结果,同时是行变换输入;C_{mn} 代表行变换结果即 2 维反变换结果。第 1 列反变换 4 拍结果(B_{00}, B_{70})、(B_{10},B_{60})、(B_{20}, B_{50})和(B_{30},B_{40})寄存到 8 组寄存器组,第 2 列结果(B_{01},B_{11},B_{21},B_{31})和寄存数据(B_{00},B_{10},B_{20},B_{31})经过 4 拍写到转置存储器 TM, 同时(B_{71},B_{61},B_{51},B_{41})寄存到(B_{00},B_{10},B_{20},B_{30})对应寄存器。第 3 列反变换时,(B_{70}, B_{71})、(B_{60},B_{61})、(B_{50}, B_{51})和(B_{40},B_{41})经过 4 拍写到 TM,同时第 3 列反变换结果寄存。TM 以此交替缓存,实现 2 点列数据输入和 2 点行数据输出,保证行列反变换流水执行。行变换结果利用 DRO 的重排寄存器组,采用相同的缓存策略,实现顶场和底场数据同时输出。

SRAM 位宽 32bits,一个地址存储同一行的两个列变换中间数据。由于行列交替,因此存在两种存储结构如表 1 和表 2。这两种结构交替使用,保证行列变换的同时读写。4×4 IIT 的存储和读写策略一致,只需要调整数据存储的先后顺序。

表 1　转置存储器中的数据(1)

Tab. 1　Data in transpose memory (1)

B_{00}	B_{01}	B_{02}	B_{03}	B_{04}	B_{05}	B_{06}	B_{07}
B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	B_{15}	B_{16}	B_{17}
B_{20}	B_{21}	B_{22}	B_{23}	B_{24}	B_{25}	B_{26}	B_{27}
B_{30}	B_{31}	B_{32}	B_{33}	B_{34}	B_{35}	B_{36}	B_{37}
B_{40}	B_{41}	B_{42}	B_{43}	B_{44}	B_{45}	B_{46}	B_{47}
B_{50}	B_{51}	B_{52}	B_{53}	B_{54}	B_{55}	B_{56}	B_{57}
B_{60}	B_{61}	B_{62}	B_{63}	B_{64}	B_{65}	B_{66}	B_{67}
B_{70}	B_{71}	B_{72}	B_{73}	B_{74}	B_{75}	B_{76}	B_{77}

表 2　转置存储器中的数据(2)

Tab. 2　Data in transpose memory (2)

B_{00}	B_{01}	B_{20}	B_{21}	B_{40}	B_{41}	B_{60}	B_{61}
B_{10}	B_{11}	B_{30}	B_{31}	B_{50}	B_{51}	B_{70}	B_{71}
B_{02}	B_{03}	B_{22}	B_{23}	B_{42}	B_{43}	B_{62}	B_{63}
B_{12}	B_{13}	B_{32}	B_{33}	B_{52}	B_{53}	B_{72}	B_{73}
B_{04}	B_{05}	B_{24}	B_{25}	B_{44}	B_{45}	B_{64}	B_{65}
B_{14}	B_{15}	B_{34}	B_{35}	B_{54}	B_{55}	B_{74}	B_{75}
B_{06}	B_{07}	B_{26}	B_{27}	B_{46}	B_{47}	B_{66}	B_{67}
B_{16}	B_{17}	B_{36}	B_{37}	B_{56}	B_{57}	B_{76}	B_{77}

可以发现,利用 8 个像素点寄存器组,使用一个双口 SRAM 就可以实现行列转置和行列数据同时读写的功能。利用两套 1 维反变换运算单元和 TM

就可以实现行列变换流水执行。而且 8×8 块内和 8×8 块间反变换完全流水,因此平均处理速度为每个8×8块 32 个时钟周期。

4 反变换性能分析

本文结构使用 Verilog HDL 描述,VCS 仿真,采用 Faraday 0.18 μm 工艺库综合,工作频率在 108 MHz,主要性能如表 3。其中处理能力指每秒钟反变换模块能够运算的帧数,以帧每秒(fps)为单位。可以看出,提出的结构只增加一套反变换运算单元,使得反变化处理速度比文献[3]中的传统结构提高了 1 倍,能够有效地支持高清实时解码。

表 3 反变换主要性能

Tab. 3 Main performance of inverse transform

SRAM 容量 (bits)	总面积 (μm^2)	时延 (ns)	处理速度 (cycles)	720pHD 处理能力 (fps)	1 080HD 处理能力 (fps)
32×32	131 835	9.03	32	183.1	68.9

SRAM 是芯片面积的重要方面,而 SRAM 的个数、形状和大小决定存储器面积。文献[5]采用 4 片 8×32bits 的双口 SRAM 实现行列转置和行列反变换流水功能,本文采用一片 32×32 的双口 SRAM 和 8 组寄存器实现了同样的功能,表 4 给出了它们的大小比较。可以看出本文的设计可以节省约 53.7% 的存储器面积,再加上每片 SRAM 的外围电路,实际可以节省更大的面积。

表 4 传统结构和本文结构转置存储器比较

Tab. 4 Former and proposed TM structure Comparison

	SRAM 大小/bits (块数×深度×位宽)	SRAM 单元面积 (μm^2)	寄存器面积 (μm^2)	转置模块总面积 (μm^2)
文献[5]结构	4×16×16	4×28 332	0	113 328
本文结构	1×32×32	1×50 513	1 920	52 433

5 结 论

利用 8×8 IIT 和 4×4 IIT 的对称性,通过数据重构,统一以 8×8 块进行整数反变换运算,从而简化了 H.264 IIT 结构。在传统 IDCT 设计的基础上,提出了一种支持行列流水的反变换硬件结构。通过增加流水深度,利用一个 SRAM 实现了行列转置,相比于采用四个 16×16bits 的 SRAM,节省了 53.7% 的存储器面积。6 级流水设计实现了整数反变换的完全流水,使反变换执行速度提高了 1 倍,能够有效地支持 H.264 高清图像的实时解码。同时,由于采用统一的反变换尺寸,硬件结构规则,电路规模大大减小。提出的重构方法和硬件结构同样适用于 AVS 和 VC1 视频标准的反变换加速设计。

参考文献(References)

1 Wiegand T, Sullivan G J, Luthra A, *et al.* Overview of the H.264/AVC video coding stand[J]. IEEE Transactions on Circuit and System for Video Technology, 2003, **13**(7): 560-576.

2 Lee Y P, Chen T H, Chen L G, *et al.* A cost-effective architecture for 8×8 two-dimensional DCT/IDCT using direct method[J]. IEEE Transactions on Circuit System Video Technology, 1997, **7**(3): 459-467.

3 Fu Yu-zhou, Wang Jia-fang, Hu Ming-zeng. The design and implementation of a novel 2-DCT/IDCT Architecture[J]. Acta Electronica Sinica, 2002, **30**(12): 2126-2129. [傅宇卓,王嘉芳,胡铭曾. 一种新型 2-DCT/IDCT 结构的设计与实行[J]. 电子学报, 2002, **30**(12): 2126-2129.]

4 Wang Nien-tsu. A novel dual-path architecture for HDTV video decoding[A]. In: Proceedings of the Conference on Data Compression [C], Snowbird, Utah, USA: Computer Society, 1999: 557.

5 Zhang Ding, Zhang Ming, Zheng Wei, *et al.* A high-performance 2-D inverse integer transform architecture for AVS[J]. Journal of Circuits and Systems. 2006, **11**(5): 93-95. [张丁,张明,郑伟等. 一种高性能的适用于 AVS 的二维整数逆变换实现结构[J]. 电路与系统学报, 2006, **11**(5): 93-95.]