

文章编号 :1004-1478 (2006)04-0085-04

# MPEG 视频解码中离散余弦 逆变换的 FPGA 实现

李绪诚, 杨鲁平, 刘宇红

( 贵州大学 计算机科学与工程学院, 贵州 贵阳 550025 )

**摘要:** 离散余弦逆变换 IDCT 是运动图像专家组 MPEG 视频解码的重要组成部分. 采用行列分解的方法和基于分布算法的乘法累加器, 实现了二维离散余弦逆变换的 FPGA 结构. 设计采用自顶向下的方法, 用硬件描述语言进行电路描述, 并在 ALTERA 公司的 ACEX1KEP1K30 上实现. 仿真实验结果表明, 最大延迟时间为 27ns, 时钟最高频率可以达到 13.35MHz, 数据吞吐能力为 6.515M/s, 完全能满足运动图像压缩标准视频解码的要求.

**关键词:** 运动图像专家组; 视频解码; 分布算法; 离散余弦逆变换; 现场可编程门阵列  
**中图分类号:** TN919.8 **文献标识码:** A

## FPGA Based IDCT Implementation for MPEG Decoding

LIXu - cheng, YANG Lurping, LIUYu - hong

( College of Comp. Sci. and Eng., Guizhou Univ., Guiyang 550025, China )

**Abstract:** Inverse discrete cosine transform ( IDCT ) is an important module in MPEG ( Moving Picture Expert Group ) decoding. The FPGA ( Field Programmable Gate Array ) implementation of two dimensional inverse discrete cosine transform ( 2D - IDCT ) is described. The design uses Multiplier Accumulator ( MAC ) based on the distributed arithmetic ( DA ) technique and row - column decomposition. The design was implemented using top - down design methodology and described with VHDL ( VHSIC Hardware Description Language ). The 2D - IDCT was implemented using the FPGA ACEX1KEP1K30 from ALTERA. Verification shows that the latency is as low as 27ns, clock frequency as high as 13.35MHz, data throughput as high as 6.515M/s. It can implement the MPEG decoding.

**Keywords:** MPEG; video decode; distributed arithmetic; IDCT; FPGA

## 0 引言

基于离散余弦变换 DCT ( Discrete Cosine Transform ) 的变换压缩编码算法是当前应用最为广泛的图像和视频解码算法之一. 研究表明, 对于一阶马可

夫随机信号, DCT 的压缩效果接近于理想的卡洛变换 KLT ( KL Transform ), 并且利用其对称性可以采用有效的快速算法<sup>[1]</sup>. 目前 DCT 已成为运动图像专家组 MPEG ( Moving Picture Expert Group ) 的重要组成部分. 作为 DCT 的逆过程, 离散余弦逆变换 IDCT ( In-

收稿日期: 2006-02-22

基金项目: 贵阳市科学技术计划项目 ( [2005] 筑科技术合同字第 20-3 号 )

作者简介: 李绪诚 ( 1980- ) , 男, 贵州省贞丰县人, 贵州大学硕士研究生, 主要研究方向: 信息传输与处理.

verseDiscreteCosineTransform)是MPEG视频解码中的重要组成部分,并且是计算量最大的部分,采用一种快速有效的IDCT实现方法对MPEG视频解码尤为重要.MPEG视频解码常用的方法有两种:一是基于软件或DSP的实现方法,即采用软件或DSP实现一些较复杂的控制,而用一些硬件来实现大吞吐量的操作;二是采用全硬件实现的方法.前者有一定的灵活性,但控制复杂,数据存取容易发生冲突而且速度较慢.因此目前主要还是使用硬件实现MPEG视频解码.对于该方法的研究,国外有一些介绍<sup>[2]</sup>,但主要问题是硬件非常复杂,实现成本也较高,不适用于小规模的系统实现,如嵌入式系统.本文采用行列分解的方法和基于分布算法的乘法累加器,设计了嵌入式Linux系统的MPEG视频解码系统.

### 1 2D-IDCT 算法及硬件结构

DCT的实现主要有行列分解法、直接实现法和间接变换法.行列分解法将二维DCT变换为两个一维DCT和一个变换矩阵;直接实现法和间接变换法与行列分解法相比主要是省去了变换矩阵的步骤.但是,行列分解法的结构更加规整而且变换矩阵可以用存储器实现,另外,采用基于分布算法(DA)的乘法累加器的结构,可以比常规的乘法累加器方法<sup>[3]</sup>减少至少一个乘法器的使用.同时,可以通过提高乘法器的吞吐率来提高整个IDCT的运算速度.

#### 1.1 2D-IDCT 算法

$N \times N$ 点的二维离散余弦逆变换(2D-IDCT)定义如下:

$$x(i, j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u) C(v) X(u, v) \cos \frac{(2i+1)u}{2N} \cos \frac{(2j+1)v}{2N}$$

其中,  $x(i, j)$  是像素值,  $i, j = 0, 1, 2, \dots, N-1$ ;  $X(u, v)$  是变换后的系数( $u, v = 0, 1, 2, \dots, N-1$ );  $C(0) = 2^{-\frac{1}{2}}$ ,  $C(u) = C(v) = 1 (u, v \neq 0)$ .

#### 1.2 2D-IDCT 系统结构

IDCT的运算量很大,一个大小仅为 $8 \times 8$ 的图像子块如果直接进行二维DCT/IDCT,就要进行8192次乘法和3584次加法操作.因此,各种快速算法应运而生,在此采用的是基于矩阵分解的分布算法.矩阵分解的思想是通过两次1D-IDCT完成2D-IDCT<sup>[4]</sup>. 式的矩阵形式可表示为  $X = [X^T C^T] C = C^T X C$ ,其中  $C$  为  $\cos$  系数矩阵,  $C^T$  是  $C$  的转置矩阵,则  $[X^T C^T]^T$  实现1D-IDCT. 将  $N \times N$  数

据按行(或列)方向进行  $N$  个1D-IDCT计算,产生中间矩阵,然后对中间矩阵再按列(或行)方向进行  $N$  个1D-IDCT计算,最后得到2D-IDCT结果,如图2.其中SIPO(SerialInandParallelOut)实现数据的串-并转换功能,PISO(ParallelInandSerialOut)实现数据的并-串转换功能.

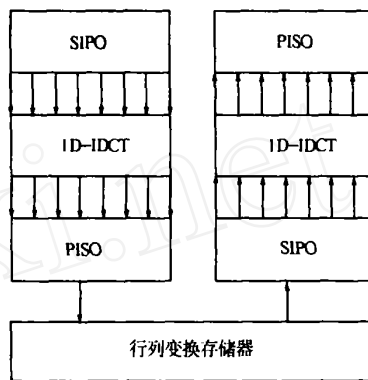


图1 2D-IDCT 结构图

#### 1.3 1D-IDCT 的快速算法

IDCT的计算仅由乘法及累加实现,因此,关键是如何在较短时间用较少硬件实现它.采用1D-IDCT快速算法及基于DA算法的乘法累加器(MAC)来解决此问题.处理的数据块尺寸为8个 $\times$ 8个像素,每个像素为16位.对于每个系数,直接计算公式

需要64次乘法,目前已有很多快速算法被提出,但其中的很多算法在用有限字长实现时,会造成精度的降低.采用Chen提出的算法<sup>[5]</sup>可以在有效的精度内用较少硬件实现.根据该算法,8点1D-IDCT可用下列公式计算:

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & B & A & C \\ A & C & -A & -B \\ A & -C & -A & B \\ A & -B & A & -C \end{bmatrix} \begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix}$$

$$\begin{bmatrix} X(7) \\ X(6) \\ X(5) \\ X(4) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & B & A & C \\ A & C & -A & -B \\ A & -C & -A & B \\ A & -B & A & -C \end{bmatrix} \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \end{bmatrix} - \frac{1}{2} \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} x(1) \\ x(3) \\ x(5) \\ x(7) \end{bmatrix}$$

其中,  $A = \cos \frac{\pi}{4}, B = \cos \frac{\pi}{8}, C = \sin \frac{\pi}{8}, D = \cos \frac{\pi}{16}, E = \cos \frac{3\pi}{16}, F = \sin \frac{3\pi}{16}, G = \sin \frac{\pi}{16}; x(i) (i = 0, 1, 2, \dots, 7)$  是像素值;  $X(u) (u = 0, 1, 2, \dots, 7)$  是变换系数.

此算法通过行列分解,需要的乘法数比式(1)少一半,图 2 表示 1D-IDCT 的结构.其中后处理器的作用是完成式(2)矩阵乘之后两个矩阵的一组加法,输入来自乘法累加器,其结果要按规整顺序输出到存储器.

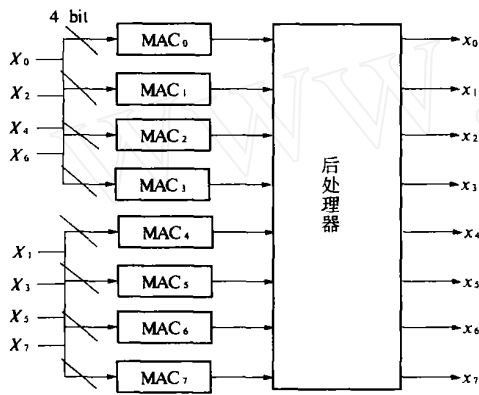


图 2 1D-IDCT 结构图

### 1.4 基于 DA 算法的乘法累加器

乘法累加器 (MAC) 用一个加法和乘法器实现,图 3 是基于传统方法的乘法累加器.

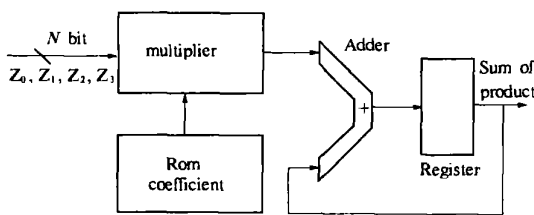


图 3 基于传统方法的乘法累加器

乘法累加器用 DA 算法实现<sup>[6]</sup>,如图 4.在 DA 算法中,设输入数据用  $N$  位二进制补码表示:

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n}$$

乘法累加可表示如下:

$$y = \sum_{k=1}^K a_k x_k = \sum_{k=1}^K a_k (-b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n})$$

其中  $a_k$  为固定系数.式可写成

$$y = \sum_{n=1}^{N-1} [\sum_{k=1}^K a_k b_{kn}] \cdot 2^{-n} + \sum_{k=1}^K a_k \cdot (-b_{k0})$$

由式(3)可看出,部分积  $\sum_{k=1}^K a_k b_{kn}$  可预先计算并存放在 ROM 中,这样可以充分利用 IDCT 的乘法运算中一个乘数为定常数的特点,将乘法器改造成位串式查表运算,从而有效降低了乘法的开销.DA 的主要部件有内存单元、加法器和寄存器.每一个变量串行地进入 ROM.每个变量有 0 或 1 两个取值,4 个变量就有 24 种取值,这些取值可以全部被预先计算出来并存入 ROM.假定每个变量长度是  $N$  bits,  $N$  个周期后,乘法器的最终结果就可以得到.

ROM 的设计尺寸是  $16 \times 12$ ,用 ALTERA 公司的 ACEX1K 系列 FPGA(Field Programmable Gate Array) 中嵌入式存储块 (EAB) 来实现,每个 EAB 容量为 4KB.ACEX1K 器件将查找表 (LUT) 和 EAB 相结合.基于 LUT 逻辑为数据路径管理、寄存器、数字信号处理的设计提供优化的性能和效率,而 EAB 可实现 RAM,ROM 等功能.

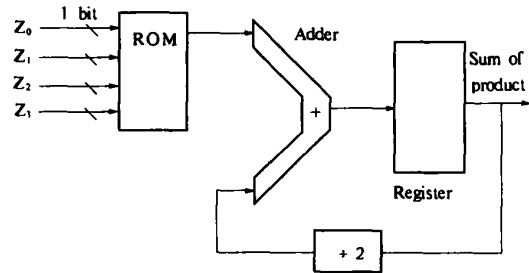


图 4 基于 DA 的乘法累加器

### 1.5 行列变换存储器

行列变换存储器一般用 RAM 或寄存器实现.在此采用寄存器实现,寄存器实现的好处是没有地址计算、控制简单.其基本结构由一个行变换缓冲器和列变换缓冲器构成.每个缓冲器由 64 个独立寄存器连接而成.行列变换器的作用是存储第一个 IDCT 产生的数据,并将其排序后输出.一个数据首先按列的方式读入行列变换缓冲器,一旦行变换缓冲器的 64 个寄存器被填满,它们就同时传送到列变换缓冲器.这种传输方法能迅速为新到来数据提供空间.

## 2 仿真实验

根据上述思想,给出基于 DA 算法的 2D-IDCT 的 FPGA 设计,采用自顶向下设计方法,用 VHDL 进行电路描述.当 VHDL 仿真通过后,综合成门级电路.采用的 EDA 工具为 ALTERA 公司的 Quartus II 5.0 和 ModelSim 6.0,最后在 ALTERA 公司的 ACEX1K

EPIK30上实现.

FPGA实现之前,用 QuartusII 的时间和功能仿真工具对设计的处理核加以验证,结果如图 5.

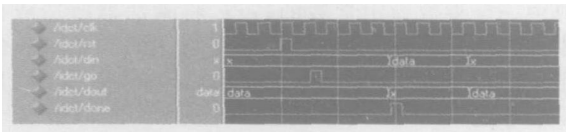


图 5 IDCT的仿真波形图

图 5 中 din 为输入, dout 为输出, go 表示一次完整的 IDCT. 两组输入数据间隔 13 个时钟周期, 从 din 送入. 高电平脉冲信号启动 IDCT 变换, 经过 26 个时钟周期, 当一个 8 点 × 8 点输入数据块运算结束时, done 输出高电平, 表示 dout 上的数据有效. rst 为复位信号, rst=1 时, 输出信号 dout 清零. 仿真表明, 2D-IDCT 关键路径的最大延迟时间为 27ns, 时钟最高频率可以达到 13.35MHz, 数据吞吐能力为 6.515M/s.

在 FPGA 实现阶段, 对加法器、ROM 和寄存器进行仔细选择以保证最小的计算错误和预期的精确度. 分析 IDCT 的设计发现, 加法器和 ROM 产生最大的传输延时. 因此, 为了进一步提高系统的数据吞吐能力, 采用了快速加法器和快速读取 ROM 技术<sup>[7]</sup>.

最后的主要实验数据如表 1, 2D-IDCT 只占用了 ACEX1KEP1K30 可用逻辑单元门的 41%, 占用可用存储器资源的 19%. 表 1 的数据表明, 设计具有高数据吞吐率和低延时的优点, 完全能满足 MPEG 视频解码的要求, 具有一定应用价值.

### 3 结语

给出一种基于行列分解的二维离散余弦逆变换的 FPGA 实现方法. 采用 DA 算法以降低硬件复杂度. 系统在 ALTERA 公司的 ACEX1K 系列 FPGA 实现, 仿真结果表明, 最大延迟时间为 27ns, 时钟最高频率可达 13.35MHz, 数据吞吐能力为 6.515M/s,

表明数据吞吐率高, 延时较小, 整个系统符合预期要求, 可实现完整的 MPEG 视频播放功能.

表 1 2D-IDCT 的主要实验数据

项目	参数
块大小	8 点 × 8 点
数据格式	输入 : 16 -bits, 输出 : 8-bits
时钟频率	13.35MHz
逻辑单元	3121 个
存储器	1840 位
数据吞吐能力	6.515M/s
FPGA 利用率	41 %
系统延时	46 个时钟周期, 240ns

### 参考文献:

- [1] AhmadN, NatarajanT, RoaKR. DiscreteCosineTransform [J]. IEEETransComputer, 1974, (23) :90 —93.
- [2] GuoJiun - In, JuRei - Chin, ChenJia - Wei. Anefficient2 - D DCT/IDCTcoredesignusingcyclicconvolutionandadder - basedrealization[J]. IEEETransCASforVideoTechnology, 2004, 14 (4) :416 —428.
- [3] YuSungwook, EarIESwartzlanderJr. DCTimplementation withdistributedArithmetic[J]. IEEETransonComputers, 2001, 50 (9) :985 —991.
- [4] ShirichiUramoto, YoshitsuguInoue, AkihikoTakabatake, et al. A100 -MHz2 - Ddiscretecosinetransformcoreprocessor [J]. IEEEJournalofSolid - stateCircuit, 1992, 27 (4) :58 —61.
- [5] CHENWH, SMITHCH, FRALICKSC. Afastcomputationalalgorithmforthediscetecosinetransform[J]. IEEE TransCommun, 1977, 25:1004 —1009.
- [6] 陈禾. 离散余弦变换集成技术研究[D]. 哈尔滨: 哈尔滨工业大学, 1998.
- [7] ChaudharyK, VermaH, NagS. AnInverseDiscreteCosine Transform(IDCT) ImplementationinVirtexforMPEGVideo Applications, Xilinx, XAPP204[DB/OL]. <http://www.xilinx.com/bvdocs/appnotes/xapp208.pdf>, 2002-05-02.