TMS320C80 的快速 DCT 及量化的实现

刘 杰 何佩琨 毛二可
(北京理工大学电子工程系, 北京 100081)

摘 要 简要说明了 TMS320C80 的结构和特点; 分析了快速 DCT 的基本原理; 论述了在 TMS320C80 上实现快速 DCT 及量化的方法, 并给出算法的测试结果.

关键词 TMS320C80 图象压缩 DCT
中图法分类号: TN919.81 **文献标识码**: B **文章编号**: 1006-8961(2000)06-0535-04

The Implementation of Fast DCT and Quantization on TMS320C80

LIU Jie, HE Pei-kun, MAO Er-ke
(*Department of electronics engineering, Beijing institute of technology, Beijing* 100081)

Abstract In this paper, the structure and characteristic of TMS320C80 is presented, the basic principle of fast DCT algorithm is analysed, the implementation of fast DCT algorithm on TMS320C80 is discussed and the result is gived at last.

Keywords TMS320C80, Image compression, DCT

0 引 言

在当今 JPEG、MPEG、H.263 等多项数字图象的压缩编码国际标准中, 都采用 DCT 变换, 因而快速有效的 DCT 算法得到广泛的应用, 而且在用 DSP 进行实时图象压缩中, DCT 也成为关键的处理部分. TMS320C80 DSP 芯片(下简称 C80), 是 TI 公司推出的以 DSP 方式工作的高性能多媒体数字信号处理器, 它包括 4 个先进的并行数字信号处理器(ADSP) 和一个精减指令集计算机(RISC) 处理器. 此外, C80 芯片内还有共享存储器、传输控制器(TC)、视频控制器(VC) 和输入、输出等部分. C80 的主处理器(MP) 是个 32bit 处理器, 可在单周期内完成一个 64bit 的数据存储和一个 32bit 的指令存取动作; 其 4 个并行处理器(PP) 具有 32bit 数据处理能力, 一个周期内可以执行多个并行操作; 而视频控制器(VC) 则包含两个能同时进行图象捕获和显示的帧定时器, 这是 C80 中最具特色的部分, 因而它特别适于视频编解码; 传输控制器(TC) 是一个智能化 DMA, 它负责 C80 与外部数据的交换, 如 VRAM, DRAM, SRAM 直接接口, 还支持一维线性寻址及二维图形处理的 X, Y 寻址等. C80 内部有 50Kbyte 的静态 RAM, 其中 32K 是 5 个处理器公用的. C80 的内部总线采用 crossbar 结构, crossbar 能使各个处理器在一个周期之内读取 5 条指令和访问 10 个数据. C80 所有操作都是流水式的, 运算速度为每秒 2 000M 次. 所有这些使得 TMS320C80 在通信、图象、图形等方面, 具有突出的应用价值.

1 二维离散余弦变换(DCT) 快速算法

JPEG 算法中最有效的部分是以 DCT 为核心的有损压缩技术, 但是该标准并未给出更为详细的技术细节, 特别对核心算法没有具体提供, 这样在采用不同的 DCT 算法下, 系统压缩处理时间有较大差别. 通常二维 DCT 变换采用正交矩阵作二维线性变换来计算 8× 8 DCT 系数, 而且二维 DCT 变换常借助 DCT

收稿日期: 1998-11-06; 改回日期: 1998-12-08

矩阵表示, 其正变换为$[F]=[DCT]\cdot[f]\cdot[DCT]^{T}$; 反变换为$[f]=[DCT]^{T}\cdot[F]\cdot[DCT]$, 其中, 8× 8 DCT 变换矩阵为三角函数矩阵, 但上述矩阵计算很费时间. 为了满足实时性要求, 在 C80 上采用了一种基于 8 点一维 DCT 快速变换来进行二维 8× 8 图象块 DCT 变换系数计算的算法. 即二维 DCT 变换可通过在行、列两个方向上的一维变换来实现. 而一维 8 点 DCT 变换则可使用快速算法, 实验证明采用后者算法速度可提高 1 倍. 其 8 点一维离散余弦正变换(FDCT)和反变换(IDCT)的表达式分别如下:

$$F(k)=\sqrt{\frac{2}{8}}c(k)\sum_{i=0}^{7}f(i)\cos\frac{2i+1}{2\times 8}k\pi \quad k=0,1,\cdots,7 \tag{1}$$

$$f(i)=\sqrt{\frac{2}{8}}\sum_{k=0}^{7}c(k)F(k)\cos\frac{2i+1}{2\times 8}k\pi \quad i=0,1,\cdots,7 \tag{2}$$

$$c(k)=\begin{cases}\dfrac{1}{\sqrt{2}}, & k=0\\ 1, & k=1,\cdots,7\end{cases} \tag{3}$$

由于 FDCT 和 IDCT 是对称的, 从式(2)入手, 参照 FFT 算法对其进行分解:

先作如下定义:

$$\sqrt{8}f(i)=\overline{f(i)} \tag{4}$$

$$\sqrt{2}c(k)F(k)=\overline{F(k)} \tag{5}$$

则式(2)可改写为

$$\overline{f(i)}=\sum_{k=0}^{7}\overline{F(k)}\cos\frac{2i+1}{2\times 8}k\pi,\quad k=0,1,\cdots,7 \tag{6}$$

对于 8 点余弦反变换, 式(6)由奇偶两部分合成

$$\overline{f(i)}=\sum_{k=0}^{3}\overline{F(2k)}\cos\frac{2i+1}{2\times 8}2k\pi+\sum_{k=0}^{3}\overline{F(2k+1)}\cos\frac{2i+1}{2\times 8}(2k+1)\pi$$

令

$$g(i)=\sum_{k=0}^{3}\overline{F(2k)}\cos\frac{2i+1}{2\times 8}2k\pi \tag{7}$$

$$h(i)=\sum_{k=0}^{3}\overline{F(2k+1)}\cos\frac{2i+1}{2\times 8}(2k+1)\pi \tag{8}$$

式(7)显然是 4 点 IDCT 变换, 而式(8)经三角变换可得

$$h(i)=\frac{1}{2\cos\dfrac{2i+1}{2\times 8}\pi}\times\sum_{k=0}^{3}(\overline{F(2k+1)}+\overline{F(2k-1)})\cos\frac{2i+1}{2\times 4}k\pi \tag{9}$$

显然这也是一个 4 点 IDCT 变换, 从而

$$\overline{f(i)}=g(i)+h(i)=\sum_{k=0}^{3}\overline{F(2k)}\cos\frac{2i+1}{2\times 4}k\pi+\frac{1}{2\cos\dfrac{2i+1}{2\times 8}\pi}\sum_{k=0}^{3}(\overline{F(2k+1)}+\overline{F(2k-1)})\times\cos\frac{2i+1}{2\times 4}k\pi \tag{10}$$

由此可见, 8 点一维 IDCT 变换可由两个 4 点变换构成, 其中 i 取 0、1、2 和 3. 实际上上式只求出了 8 点变换中的前 4 点值, 对于后 4 点值, 根据上式易得

$$\overline{f(7-i)}=g(7-i)+h(7-i)=g(i)-h(i)=\sum_{k=0}^{3}\overline{F(2k)}\cos\frac{2i+1}{2\times 4}k\pi-\frac{1}{2\cos\dfrac{2i+1}{2\times 8}\pi}\sum_{k=0}^{3}(\overline{F(2k+1)}+\overline{F(2k-1)})\cos\frac{2i+1}{2\times 4}k\pi \tag{11}$$

其中 $i=0$、1、2 和 3.

通过这样的推导, 将 8 点的反变换分解成了 2 个 4 点反变换 $g(i)$ 与 $h(i)$ 的和差运算. 接着用同样的方法, 还可分别把它们进一步分解成 2 点的 IDCT 变换, 这里不再赘述. 图 1 是该算法的流程图, 从左到右是 IDCT, 反之是 FDCT.

图中: $c_i=\dfrac{1}{2\cos\dfrac{i\pi}{16}}$, $i=1,2,\cdots,7$

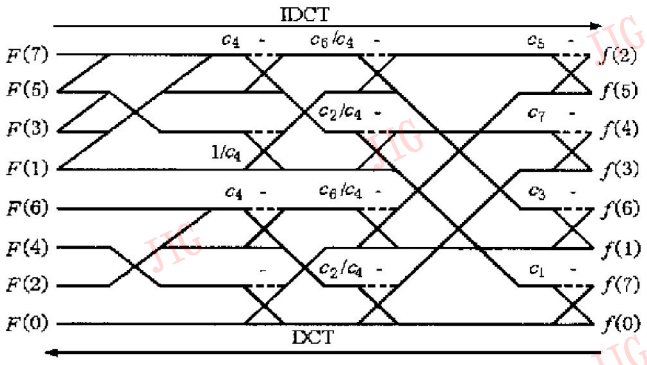

图 1 8 点 1D 快速离散余弦变换算法流程图

2 C80 算法实现的基本考虑

由于 C80 是多处理器系统, 其应用的关键是其

并行处理能力, C80 内部的 MP, PP, TC, VC 几个主要部件可以并行. C80 图象处理的基本过程是: 图象的采集与显示由 VC 负责; 数据在芯片内、外的传输由 TC 完成; MP 作为整个系统的管理者, 负责对 PP 的指挥调动, 将待处理的图象分为 4 块, 由 4 个 PP 并行处理. 由于芯片内处理速度高于芯片外, 所以首先需用包传输将图象数据从芯片外送到芯片内, 待处理完后, 再用包传输送到芯片外 RAM. 在软件的设计中, PP 的编程是核心的问题. 由于 PP 的字长是可以分裂的, 即 32bit 可分成 2 个 16bit, 4 个 8bit 等等, 因此这特别适合图象处理, 而运算后, 还可由 mf 寄存器进行扩展恢复. 由于 PP 的编程中大量使用循环, 且 PP 中有 3 个硬件循环控制器, 控制多层循环, 从而加快了循环的执行过程. 用户可以灵活使用硬件紧循环及 mf 扩展. 每个 PP 的指令字是 64bit 宽, 而每一个指令由多个独立场构成, 每个场指定一个独立的操作, 使每一个 PP 在一个时钟周期中, 又可并行执行 10 个操作, 只要这些指令不发生资源冲突即可. 处理时, 将任务合理地划分到 5 个处理器上, 在每个处理器上将运算与数据传输合理分配, 使二者并行执行; 另外, 在指令中合理分配计算与对芯片内 RAM 的存取, 并充分利用流水线与零等待循环, 使两种计算可在同一指令内或在若干条指令内并行执行; 而在某些情况下, 还可将运算单元进行分解, 使多个单元同时进行操作.

3　C80 上快速 DCT 及其量化的实现

(1) DCT 变换的运算

在图 1 所示的 DCT 快速算法中, 要进行 5 次的迭代运算. 而每一次以两点为一组进行蝶式加乘、减乘运算, 共需进行 20 次运算. C80 实现时, 先对 8 行象素分别进行变换, 然后再按列进行 8 列变换. 中间变换系数 c_1~ c_7 是 0.5~ 3 的小数. 由于 C80 的并行处理器是定点处理器, 必须将变换系数由小数变为整数. 为了保证编码中的 DCT 的精度, 在计算中保留 3bit 小数位, 即 c_1~ c_7 乘 2 的 13 次幂, 然后取整, 存放在 C80 芯片内 RAM 中; 图象的数据也要事先放到芯片内 RAM 中, 然后进行迭代运算. 由于一般视频图象灰度值为 8bit, 加上 3bit 小数位及一个符号位共 12bit. 而 C80 的寄存器是 32bit, 因此一个寄存器可分成两个 16bit, 用其存放两个象素的灰度值足以满足要求. 为了保证计算精度, 计算中, 通过移位以保证始终具有 3bit 小数, 待运算结束时再进行舍入. 由于 C80 的寄存器数目有限, 计算的中间结果暂存于 C80 芯片内 RAM 中. 在 DCT 计算中, 由于左上角第一个象素的值较大, 当其中间结果最高位为 1 时, 如为 0x8134 与 c_0(0x510) 相乘时, 由于相乘后的最高位是 1, 系统可能认为结果是负数而造成错误. 为此可将该值右移 1 位, 相乘后再左移, 以保证结果的正确. 在一般的 DCT 算法中, 对 DCT 中的行, 列元素对换, 使用单独循环处理, 在 C80 实际运算中可在 DCT 处理后, 直接将变换后的值送入对应的单元中, 而勿须再进行行列对调, 以节省时间.

C80 提供的扩展 ALU, 即 EALU 对 DCT 的实现也是很有效的, EALU 在进行乘法运算时, 可以自动进行舍入运算, 又不溢出. 并且利用乘加、移位功能, 还可并行地进行数据存储. 但 EALU 运算, 由于其对乘法结果每次都进行舍入, 而不保留其余小数位, 故使其计算精确度降低. 此外, 其计算结果保存在 32bit 寄存器的高低 16bit 上, 由于在进行下一步运算时要先取出到另一个寄存器, 因而要占用两个时钟周期.

(2) 量化处理

即在 DCT 处理后进行量化. 量化实际是将计算值除以量化表中的值. 由于 C80 没有除法器, 而是通过迭代减法完成除法, 因而速度很慢. 可采用乘法代替除法, 其方法是先计算好量化值的倒数, 然后将其乘 2 的 15 次幂取整. 量化时可直接相乘, 最后结果再右移 15bit. 在量化中, 当符号数右移后, 由于 C80 对小于 0 的数均认为是 0xffffffff, 为此可先取其绝对值, 判断该值是否小于 0, 若小于 0, 则量化值为 0. 对正值量化后加 0.5, 而对负值量化后则要减 0.5. 实际处理时, 还要判断量化后的第一位小数的值是否为 1, 若为 1, 则舍位进 1, 若为 0, 则舍去, 以保证量化的精度. 实践证明, 采用本方法比使用除法, 量化速度提高 1 倍.

(3) C80 算法编程及调试

C80 程序编制是很复杂的过程, 为了提高实时性, 要充分利用 C80 的并行指令, 只要相互不发生冲突即可, 如 $d_1 = d_2 * d_3 \| d_2 = d_7 - d_5 \| d_3 = * a_0$, 可并行执行 3 条指令. 再如乘法和加法的并行, 可执行 3 输入的 ALU 以及 2 次数据存取功能等并行处理, 可能出现冲突, 因此需合理使用寄存器, 使中间运算结果不必放到 RAM 中, 从而使在芯片内的内存中保持所需的常数. 在 C80 程序调试时, 经常遇到单步与连

续运行结果不一致,主要是由于流水操作所致,此时可在中间加入 nop 空指令等待即可. 此外笔者在编程时发现 C80 中的右移操作,如 $d_1 \gg u- d_2$,必须先对 d_2 取补码~ d_2,再对 d_2 加 1 后,才能实现右移功能,与 C80 资料介绍的不一致.

4 实验结果及结论

通过 MP 的定时器寄存器 tcount,对 C80 上用 C 语言及汇编语言进行的 8× 8 块的图象 DCT 变换的耗时实施了测试,结果如下:

C 语言算法二维 DCT: 150μs.

C 语言算法一维快速 DCT: 80μs.

汇编语言二维 DCT: 26μs

汇编语言快速 DCT: 15μs

可见,若能优化 JPEG 中的其它部份,如 HUFFMAN 编码,完全可实现实时图象压缩处理.

参 考 文 献

1 TMS320(MVP) User's Guide, Texas Instrument, 1995.

2 TMS320C8X SOFTWARE Development Board Technical Reference

刘 杰 北京理工大学电子系博士生,主要从事图象处理、图象压缩等.

何佩琨 北京理工大学电子工程系教授,博士生导师. 主要研究方向为高速实时数字信号处理、雷达信号处理.

毛二可 北京理工大学电子工程系教授,博士生导师,中国工程院院士. 主要研究方向为信号处理、雷达、电子系统设计.