基于通用 DSP 的多路视频编码器的优化实现

李 波, 孟庆磊, 姚春莲, 姜 黎
(北京航空航天大学计算机学院数字媒体室, 北京 100083)

摘 要: 本文基于单片 TI DM642 设计实现了四路 MPEG-4 实时编码器, 具有功能强、可靠性高、体积小和功耗低等特点. 在硬件设计中, 充分利用了 DM642 可以同时接受多路视频信号、DSP 数据处理与 EDMA 数据传输并行工作等特点, 构造了处理速度快、控制灵活的视频编码硬件平台. 在软件实现中, 根据 DSP 的硬件和资源特点, 提出了基于最优位置计算的半像素运动估计和高效量化策略, 设计了基于 GMB 的编码方式.

关键词: 编码; 视频; MPEG-4; DSP; 嵌入式
中图分类号: TN919.8 文献标识码: A 文章编号: 0372-2112 (2006) 11-2103-06

Optimized Implementation of Multi-Channel Video Encoder Based on General DSP

LI Bo, MENG Qing-lei, YAO Chun-lian, JIANG Li
(*Digital Media Laboratory, School of Computer Science and Engineering, Beihang University, Beijing* 100083, *China*)

Abstract: This paper presents an embedded multi-channel video encoder using single-chip TI DM642 DSP. Not only could it process 4-way video signals real-time, but also it is full-function, high-reliability, small-size cubage, flexible control and low-power consumption. We construct encoder hardware platform by taking full advantage of DM642 DSP, video ports of DM642 can accept multi-channel video signals at the same time, and DSP handle video data parallel with the operation of data transferring using EDMA. By analyzing the hardware architecture and resource of DM642, we presents GMB coding scheme and efficient algorithm for DSP, namely fast motion estimation algorithm by calculating the best position in sub-pixel level, and DSP-based efficient quantization algorithm etc.

Key words: encoder; video; MPEG-4; digital signal processor (DSP); embedded

1 引言

随着网络和多媒体技术的快速发展, 视频的压缩、分析与传输得到了广泛的应用. 体积小、功耗低和环境适应强是当前远程监控、移动流媒体等应用提出的普遍要求, 基于 DSP 实现嵌入式的视频编解码器[1~4]由于具有开发周期短、可靠性高、处理速度快、便于升级以及体积小、功耗低、适应性强等优点, 目前已成为发展趋势, 因此研究基于通用 DSP 的视频编码器具有重要意义.

当前基于 DSP 实现的编码器通常只能完成一路压缩, 而在许多实际应用中, 常常需要对多路场景进行实时监控和存储, 例如在机载环境监控中需要同时获取发动机、飞行仪表、飞行员和客舱状况的视频信息. 使用一片 DSP 实现多路视频监控, 充分利用 DSP 的运算能力, 可以降低成本、减少设备的功耗和体积. 与单路处理相比, 多路编码器需要处理多路视频的采集和编码等调度问题, 以及由于计算量和存储空间增加而带来的处理难题, 因此如何在硬件设计和编码器软件实现上进行优化, 充分利用 DSP 处理能力强的优点, 是基于 DSP 实现多路视频编码器的关键.

本文给出了基于 TI 的 TMS320 DM642 DSP(以下简称 DM642) 为核心, 实现四路分辨率为 352× 288 的 MPEG-4 视频编码器的实现方案. 在硬件设计上, 充分利用了 DM642 可以接受多路不同格式音视频信号的特点, 实现了 DSP 处理与 EDMA 传输并行工作, 选用双端口 RAM 与其他部件进行高速数据交互, 从而构建了控制灵活、处理速度快的视频编码硬件平台. 在编码器的软件实现中, 根据 DSP 的结构和实时处理需求, 对半像素运动估计、量化等关键算法进行了改进, 使其更适合在 DSP 上运行, 并对资源使用和软件代码进行了全面优化, 从而实现了快速、高效的视频编码.

2 多路视频编码器的系统结构

本文以 DSP 为核心实现的四路 MPEG-4 视频编码器主要由采集、编码和输出三部分构成, 硬件结构如图 1 所示.

采集部分由两个 A/D 和 DM642 的 VP(Video Port) 口组成. A/D 将模拟视频信号转换为数字量, DM642 的 VP 口对输入的数字信号进行格式转换, 将采集的 YUV 分离并按照 YUV 分

收稿日期: 2005-12-05; 修回日期: 2006-03-20
基金项目: 国家自然科学基金(No. 60505007); 教育部博士点基金(No. 20030006034); 国防基础科研项目

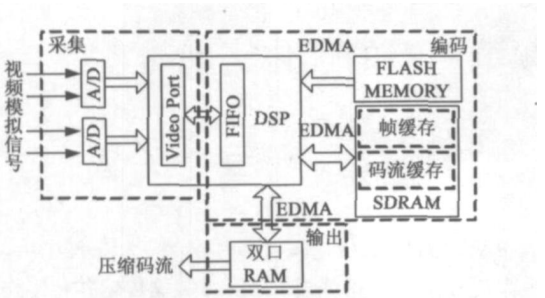

图 1 多路视频编码器硬件系统结构

量各自连续存放的格式存放在SDRAM 中. 同时, DSP 通过 EDMA 完成 4∶2∶2到4∶2∶0 的视频格式转换, 以适应 MPEG 编码器的要求.

编码部分(在不引起混淆时也简称编码器)由 DM642、SDRAM 和 FLASH MEMORY 组成. DM642 作为编码器的主处理芯片, 用于执行程序, 进行数据压缩处理. 采用 Micron 公司的大小为 16MByte 的外部存储器 SDRAM 作为编码模块的数据缓存, 用于存放压缩过程中的参考帧、重建帧、压缩码流等, 以解决编码过程中片内存储资源不足的问题. AMD 公司的 4MByte FLASH MEMORY 芯片作为程序存储器, 实现系统的加电自举.

输出部分主要是双端口 RAM, 大小为 256K× 18Bit. 编码器将压缩码流写入双端口 RAM, 上位机通过周期轮询方式从双端口 RAM 共享空间获取压缩码流, 以中断方式通知编码器从双端口 RAM 的命令共享空间获取指令并执行相应操作.

TI 公司的 TMS320 DM642[5,6] 采用了 VLIW(Very Long Instruction Word)体系结构, 是一种单指令流多操作码多数据的系统结构, 具有相互独立的程序总线、数据总线和一套 DMA 专用总线. 采用 L1/L2 两级存储器结构, 128K-Bit 的 L1P 程序 Cache 和 L1D 数据 Cache, 2M-Bit L2 统一映射的 RAM/Cache, 可灵活配制成 Cache 或片上内存. DM642 有专门为多媒体应用设计的三路视频输入、输出端口(VP 口), 每个 VP 口又分成 A 和 B 两个通道, 可与两路 10bit 或一路 20bit 视频 A/D 转换器直接相连. 通过复用三路 VP 口最多允许 6 路视频输入. 如果将 VP 口配置为视频输出, 则只能在 A 通道输出, 所以最多支持三路视频输出. DM642 目前工作主频是 600MHz, 能通过各种时钟速度进行主频的升级, 有支持不同温度范围的芯片, 有较好的抗震性. 本文的多路视频编码器硬件方案具有以下特点:

(1) 对硬件资源使用充分. DM642 的 VP 口可以和 A/D 方便地实现无缝连接, 通过 VP 口灵活的配置可接受多路视频信号.

(2) 具有高速的数据交互能力. 通过双端口 RAM, 实现了与其他部件的集成协同工作.

(3) 便于升级更新. 采用 DSP 实现, 提供了灵活的平台, 目前软件采用 MPEG-4, 可根据需要进行更新或升级.

3 多路视频编码器的软件实现方案

软件实现方案分为采集部分和压缩部分, 核心是压缩部分的 MPEG-4 视频编码器.

采集是并行实现的, 即通过 DM642 的 VP 口同时处理四路视频源, 将数据存放到对应 VP 口的缓存区. 在 SDRAM 中, 为每一路视频开辟了可存放三帧图像的三个缓存区, 通过中断信号改变缓存区的指针, 将 VP 口中的数据通过 EDMA 搬移到缓存区中周期存放. 在采集部分, 充分使用 DM642 的 VP 口资源和片外的 SDRAM, 无须消耗 DSP 的运算资源, 提高了系统的整体性能.

压缩部分采用的 MPEG-4 是基于分块变换和运动补偿和运动补偿预测的混合编码技术[7], 如图 2 所示, 编码过程主要包括 DCT 变换、量化(Quantization)、熵编码(VLC)、位率控制(RC)、运动估计(ME)和补偿(MC)以及图像重建等部分. 在 MPEG 编码器中, 是以图像组 GOP(Group of Pictures)为单位进行编码, 一个 GOP 中包括一个 I(Intra pictures)图像和若干个 P(Predicted pictures)、B(Bidirectional prediction)图像. 对于帧内编码图像(I 图像), 使用 DCT 变换、量化和熵编码技术, 来消除图像的空域冗余和统计冗余信息. 对于帧间编码图像(P 图像和 B 图像), 利用相邻图像的内容相似性, 首先使用运动估计和运动补偿预测, 去除视频序列的时域冗余信息, 得到残差图像, 然后对残差图像使用帧内压缩算法进行处理. P 图像参考前面的重建图像进行单向预测编, B 图像参考相邻重建图像进行双向预测编. 在编码过程中, 位率控制环节主要完成量化参数的计算.

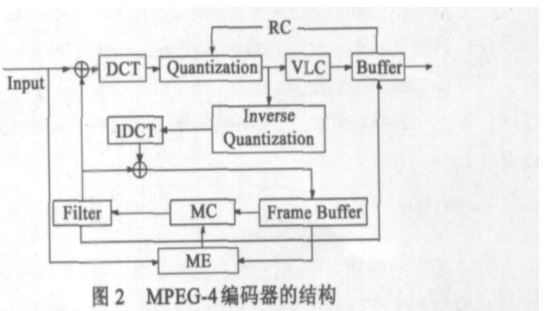

图 2 MPEG-4 编码器的结构

在软件实现中, 考虑到多路视频的处理、实时性的需求和嵌入式环境的限制, 压缩部分在如下三个方面进行了改进:

(1) 在 MPEG 编码器中 B 图像进行双向预测编码, 压缩比很高. 但其缺点是运算复杂度高, 需要对前后两参考帧进行搜索; 内存空间增加了一倍, 要保留前后两个参考帧; 增加了延迟, 需要先对其后的参考帧编码然后才能对当前 B 图像编码, 不满足实时性的要求. 为此在本编码器中改进了 GOP 的结构, 去掉了 B 图像, 而只保留了 I 和 P 图像.

(2) 多路编码调度有两种候选方案: 一是使用 TI 集成的开发环境 CCS(Code Composer Studio)中提供的 DSP/BIOS 开发工具. 它是一个简易的嵌入式操作系统, 提供了按优先级进行任务和中断的调度, 即四路编码程序按时间片轮转方式执行. 二是四路编码串行执行, 即用户程序控制 DSP, 在编码完第一路的当前视频帧后, 再编码下一路的当前视频帧, 依次循环进行. 在第一种方案中, 所有与硬件的操作都必须借助 DSP/BIOS 本身提供的函数完成, DSP/BIOS 本身也要占用一些 CPU 资源和片内存储; 但是可以在工具中静态建立 DSP/BIOS 对像

并在生成应用程序之前对对像属性的合法性进行校验,也提供了图形化显示各任务的时间开销等分析和评估代码工具.在第二种方案中,使用汇编语言和 C 语言编写应用程序,可以人为更合理地安排内存资源的使用,直接控制硬件资源,按用户期望的方式执行程序.

综合考虑实现的复杂性以及对资源分配和硬件操作的必要性,本文采用了第二种方案,即多路视频编码串行执行.在编码每路的当前帧图像时,从相应的缓存区读取图像数据,进行压缩编码和输出.

(3)系统要求具有实时性,即每 10ms 要编码一帧 CIF 大小的视频图像.但是 DSP 的并行体系结构同微机 CPU 有较大区别,其运行频率也远远不及 CPU 高,存储资源也非常有限.而且 MPEG-4 压缩算法的计算复杂度高、编码过程中需要频繁交互数据,在 PC 上开发且运行效率很高的程序在 DSP 上执行时并行程度很差,程序的软件流水不足.为了保证四路视频的实时编码和恢复图像质量,除了在硬件设计和软件总体上进行考虑外,需要充分利用硬件资源、针对 DSP 的特点,在算法、传输、存储器资源分配和代码实现等方面进行优化.

4　MPEG-4 算法的改进

根据硬件资源、延迟限制和实时编码的需求,在对 MPEG-4 标准及其验证模型分析的基础上,本文对在 DSP 上运行时间、空间复杂度高的算法进行了改进,这里主要介绍半像素运动估计和量化的算法改进与优化.

4.1　半像素运动估计算法优化

运动估计由整像素运动估计和半像素运动估计组成.整像素运动估计在整像素级(即采集的像素)寻找当前待编码块(简称"当前块")的最优匹配块,本文编码器采用文献[8]中快速整像素运动估计算法 SMVFAST.

在 MPEG-4 的半像素精度运动估计中,先插值参考图像,然后搜索整像素最优匹配块周围的 8 个 1/2 像素块,得到最佳 1/2 像素块.其主要缺点是:插值参考图像将增大为原始输入图像的 4 倍,在 DSP 受限的存储资源下,必然导致大量的数据交换,程序执行效率急剧下降,无法满足实时处理要求;此外,若每个块都进行 8 次 1/2 像素匹配,则增加的计算量很大,对于在一片 DM642 上完成四路编码来说,增加的计算量也是无法容忍的.因此,MPEG-4 标准的半像素运动估计在本系统中是不可行的.为解决这一问题,本文提出了基于最优位置计算的快速半像素运动估计算法 FSME*,它利用整像素运动估计的中间结果计算最优半像素位置,不仅将搜索块数降低到 2 个以下,而且避免了 MPEG-4 算法中因插值半像素点而占用的大量存储开销.

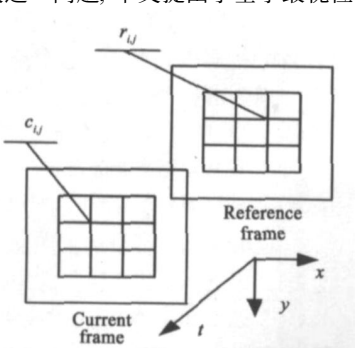

图 3　整像素运动估计示意图

如图 3 所示,设当前块上的点为 $c_{i,j}$, $i \in (0, M)$, $j \in (0, N)$,其中 M, N 分别表示当前块包含的水平、垂直方向像素数;整像素运动估计得到的参考帧中最优匹配块上的点为 $r_{i,j}$; $r_{i,j}$, $r_{i+1,j}$ 之间的半像素值为 $h_{i,j}$,见图 4.推导过程中取 MSE(均方误差)作为匹配准则,记图像中任意两点 $d_{x,y}$ 与 $d'_{x,y}$ 所在块的 MSE 值为 $D_{d_{x,y},d'_{x,y}}$.

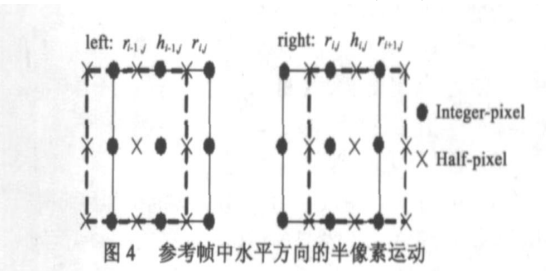

图 4　参考帧中水平方向的半像素运动

由双线性插值知:

$$h_{i,j} = \frac{1}{2}(r_{i,j} + r_{i+1,j}) \tag{1}$$

由整像素运动估计可得:

$$D_{c_{i,j}r_{i,j}} = \frac{1}{MN}\sum_{i=0}^{M}\sum_{j=0}^{N}(c_{i,j} - r_{i,j})^2 \tag{2}$$

$$D_{c_{i,j}r_{i+1,j}} = \frac{1}{MN}\sum_{i=0}^{M}\sum_{j=0}^{N}(c_{i,j} - r_{i+1,j})^2 \tag{3}$$

半像素运动估计 MSE 值:

$$D_{c_{i,j}h_{i,j}} = \frac{1}{MN}\sum_{i=0}^{M}\sum_{j=0}^{N}(c_{i,j} - h_{i,j})^2$$

由式(1)、(2)、(3)得:

$$D_{c_{i,j}h_{i,j}} = \frac{1}{2}D_{c_{i,j}r_{i,j}} + \frac{1}{2}D_{c_{i,j}r_{i+1,j}} - \frac{1}{4}D_{r_{i,j}r_{i+1,j}} \tag{4}$$

同理可推导出

$$D_{c_{i,j}h_{i-1,j}} = \frac{1}{2}D_{c_{i,j}r_{i-1,j}} + \frac{1}{2}D_{c_{i,j}r_{i,j}} - \frac{1}{4}D_{r_{i-1,j}r_{i,j}} \tag{5}$$

由式(4)、(5)可计算出

$$\begin{aligned} &D_{c_{i,j}h_{i,j}} - D_{c_{i,j}h_{i-1,j}} \\ &= \frac{1}{2}D_{c_{i,j}r_{i+1,j}} - \frac{1}{2}D_{c_{i,j}r_{i-1,j}} \\ &\quad - \frac{1}{4MN}\left(\sum_{i=M}^{M}\sum_{j=0}^{N}(r_{i,j} - r_{i+1,j})^2 - \sum_{i=0}^{0}\sum_{j=0}^{N}(r_{i,j} - r_{i-1,j})^2\right) \end{aligned} \tag{6}$$

则一维水平最优位置 x 为:

$$x = \begin{cases} -\frac{1}{2}, & \text{if } (D_{c_{i,j}h_{i,j}} \geq D_{c_{i,j}h_{i-1,j}}) \\ \frac{1}{2}, & f\ (D_{c_{i,j}h_{i,j}} < D_{c_{i,j}h_{i-1,j}}) \end{cases} \tag{7}$$

由式(6)、(7)可直接计算半像素精度下,使当前块与水平半像素块的 MSE 达到最小值的半像素点的位置,即水平最优半像素点的位置.在式(6)中, $D_{c_{i,j}r_{i+1,j}}$、$D_{c_{i,j}r_{i-1,j}}$ 均为整像素运动估计的中间结果,因此相对与 MPEG-4 的半像素搜索算法中的插值及搜索匹配代价,额外计算量相对很小,仅仅需要计算 $2N$ 次减法和平方运算.

考虑到物体的二维运动可看作水平、垂直两个一维方向上的分运动的合成.如图 5 所示,FSME* 算法通过式(7)直接推导出水平、垂直方向的半像素最优位置 x 和 y,再根据二维运动的分解假设,将位置的对应块作为二维最优候选块.此

外, 考虑到该候选块的位置是水平、垂直方向合成的结果, 可能是不准确的, 为提高估计的精度, FSME* 将水平最优位置块、垂直最优位置块中残差最小者作为另一候选块. 将候选块集合中的各块与当前待搜索块匹配求取 MSE, 并与整像素搜索结果相比较, 取最优块作为最终结果.

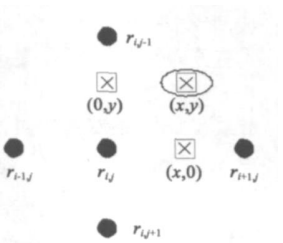

图 5 二维近似最优位置的确定

表 1 给出了采用半像素全搜索算法(SME)和本文算法(FSME*)的 PSNR 值和编码时间, 测试序列为 MPEG 组织提供的标准测试序列 Basket(运动剧烈), Flower(图像纹理丰富, 运动中等, 背景运动)和 Coastguard (纹理简单, 运动缓慢), 分辨率为 352× 288(CIF), 编码总帧数为 100 帧, 帧率 25 帧/秒, 500Kbps 带宽, 本文以后的测试条件与此一致.

表 1 测试结果比较

序列	SME PSNR(dB)	FSME* PSNR(dB)	减少时间(%)
Basket	27.65	27.62	50.48
Flower	25.88	25.84	59.06
Coastguard	30.21	30.20	43.04

从表 1 中可以看出, FSME* 与 SME 的 PSNR 值非常接近, 压缩性能基本保持不变, 但运行时间却减少了 40~ 60%, 而且避免了因插值半像素而占用的大量存储开销. FSME* 中采用的对二维运动的近似分解导致计算出的最优位置与实际最优位置略有一点偏差, 这是压缩性能略有下降的原因.

4.2 MPEG-4 量化优化策略

MPEG-4 采用了两步量化: 首先使用视觉量化矩阵对 DCT 系数矩阵进行归一化处理, 然后用量化参数进行量化. 以非帧内编码块系数为例:

第一步用帧间视觉加权量化矩阵 $\boldsymbol{Q}_{MI}$ 归一化所有的块中 DCT 系数:

$$C_r = (16\times C_{ij}) \parallel \boldsymbol{Q}_{MI} \tag{8}$$

其中 C_{ij} 表示编码块中待量化的 DCT 系数, $\parallel$ 表示四舍五入除法.

第二步使用位率控制算法得到的量化参数 Q_P 对归一化的结果进行量化:

$$L_r = C_r/(2\times Q_P) \tag{9}$$

最后对式(9)的结果 L_r 进行饱和处理, 即剪裁到$[-127, 127]$范围内.

这种两步量化考虑了人眼的视觉特性, 因此恢复图像主观质量较高, 其缺点是增加了编解码的复杂度. 在嵌入式环境下, 运行效率很低, 其原因是:

(1) MPEG-4 量化过程中进行了大量的除法运算, 而 DSP 芯片没有提供硬件除法部件, 另外量化过程中的除法操作需要进行浮点运算, 而定点 DSP 在进行浮点运算时处理过程复杂, 处理效率很低.

(2) 在标准量化过程中, 对量化结果的饱和处理时存在大量条件判断和分支处理, 导致 DSP 不能有效进行并行处理.

为了解决 MPEG-4 的两步量化运行效率低的问题, 本文研究实现了基于定点 DSP 的高效量化, 其策略如下:

结合式(8)和(9)并进行适当的取舍, 可得到如下等价变换:

$$L_r = \frac{16}{2\cdot Q_P\cdot \boldsymbol{Q}_{MI}}\times C_{ij} \tag{10}$$

式(10)中存在除法运算和浮点数乘法运算, 不适合在定点 DSP 上计算, 为了提高运行效率, 本文采用如下方法, 令:

$$\boldsymbol{Q}_{MIS} = \frac{16\cdot 2^N}{2\cdot Q_P\cdot \boldsymbol{Q}_{MI}}$$

式(10)可以表示为:

$$L_r = (\boldsymbol{Q}_{MIS}\times C_{ij}) \gg N \tag{11}$$

其中$\gg N$ 表示向右移 N 位. $\boldsymbol{Q}_{MIS}$ 是新构造的移位量化矩阵, 对于每一个固定的 Q_P, 可以事先计算出一个相应的移位量化矩阵.

这样, 通过构造新的移位量化矩阵和移位操作, 将 MPEG 的两步量化合并成一步完成, 将量化中的浮点运算变为定点运算, 去除了除法运算. 充分利用了 DSP 中提供的乘法器及其并行结构, 大大提高了程序的运行效率.

5 片内 RAM 的分配策略和代码级优化

编码器程序经过算法优化后还不能满足实时处理要求, 我们发现主要有三方面的原因: 第一, 需要下载到 DSP 中的程序超过 300KB, 但是 DM642 的 L2RAM 只有 256 KB, 程序均存放在 DSP 外的 SDRAM 中, 运行时存在大量对程序和数据的反复读写. 第二, 在编码过程中, 需要频繁交互数据, 如输入采集的图像数据和参考帧数据到片内、输出重建图像数据到 SDRAM 等, 以及程序运行过程中分配的临时存储空间都放在片外 SDRAM; 而 DSP 读写 SDRAM 时, 不但速度慢需要插入等待周期, 而且打断了 DSP 的软件流水. 第三, 由于 DSP 的 VLIW 体系结构与 PC 机结构不同, 部分程序代码不适合 VLIW 结构, 程序运行的并行度低. 针对这些问题, 本文研究了片内 RAM 的分配策略和程序代码的优化.

5.1 基于 GMB 的编码

MPEG-4 标准采取基于帧的编码, 即每次对整帧图像数据进行处理. 若将一路视频图像的当前帧和参考帧全部载入 DSP 内, 则需要约 360KB 片内存储空间, 此外存储源程序代码需要超过 300 KB, 程序中的其他变量需要约 300KB, 远远大于 DM642 所能提供的 256KB 的片内存储空间. 因此, 如果仍然按照以帧为单位进行编码, 则必须将所有的图像数据放置在 SDRAM 中, 导致在编码过程中需要进行频繁的片内外数据交互. 访问 SDRAM 的速度比访问片内 RAM 慢很多[6], 这就进一步激化了处理器内核的处理速度与数据存取速度之间的矛盾, 使得 DSP 大部分时间都用于对外部存储器的访问, 不能充分发挥 DSP 在数据处理上的优势.

在 MPEG-4 编码器的工作过程中, 首先以 16× 16 的宏块为单位, 再进一步将宏块分成 8× 8 的块进行编码, 编码器中的算法(如 DCT 变换)都是以宏块/块为单位进行. 为此, 本文提出了基于 GMB(Group of MacroBlock, 即由相邻的若干宏块组成的图像组)的编码方案. 为了尽量减少编码 GMB 过程中与

SDRAM 的交互操作,片内编码数据缓冲区必须足够存储每个 GMB 对应的原始图像数据,参考图像数据,以及运行相关的变量空间. GMB 的具体大小由片内存储区所能分配给图像帧存的大小和图像的分辨率决定.

同时考虑到视频数据量巨大,传输时间比较长,而 DM642 的 EDMA 能提供超过 2Gb/s 的数据传输带宽. 如果在 DSP 完成初始配置后,EDMA 自动进行数据传输,而无须 DSP 介入,则可以有效减少 DSP 用于数据传输的时间.

基于上述考虑本文采用图 6 所示的双缓冲结构,将每行宏块划分成一个 GMB,则处理一个 GMB 所需的数据交换区大小为 10.125KB(原始 GMB) + 49.125KB(参考 GMB) + 27.5KB(结构变量),约为 86.5KB. 这样,编码器的数据存储方式可以分为帧级和 GMB 级两种,帧级缓存置于 SDRAM,GMB 级缓存置于片内 RAM. 图中的 Buf1、Buf2 组成乒乓结构,其数据的输入输出都使用 EDMA. 当 Buf1 用于存储当前编码 GMB 的数据时,Buf2 进行下一个 GMB 数据的输入;反之亦然,则数据交互区共需要 173KB. 这样使 DSP 与 EDMA 控制器在任何时刻都分别对不同的缓冲区进行操作,避免了访问冲突,实现了 DSP 处理和 EDMA 传输的并行操作.

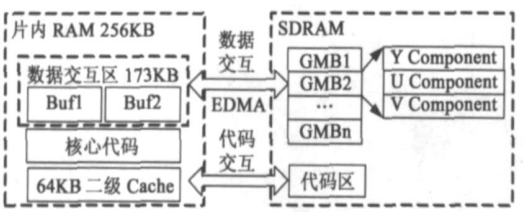

图 6　基于 GMB 的编码方案

5.2　代码调度和 CACHE 优化

基于 GMB 的编码解决了数据调度频繁的问题,但是如果相应程序的调度不能及时完成,则仍然无法提高编码器的性能. 由于 DSP 片内存储空间的限制,不可能将所有的程序代码装载进片内 RAM,所以只能关注调度频繁的代码. 因此,在片内 RAM 中可以分配一段固定空间,用于存放频繁使用代码,如 DCT、IDCT、量化、反量化、sad 计算等函数.

去除数据交互区占用的 173KB 后,还剩余 83KB 片内空间可用于存储程序代码和全局数据(如 4.2 节的移位量化矩阵),研究中分析了三种候选方案:

方案一: 83KB 存储空间全部配置成片上内存,形成 ALL SRAM 的 L2 模式.

方案二: 将 83KB 存储空间配置成 64KB 的 Cache,19KB 用作存储常用代码(记为核心代码),其余程序代码和数据放置在 SDRAM 中;

方案三: 将 83KB 存储空间配置成 32KB 的 Cache,其余用作存储常用代码和数据.

在方案一可以设置代码和数据的存储位置,一旦发生访问外存则将占用 DSP 大量的处理时间. 方案二和三中对外存的访问可能在 Cache 中高速完成,缺点是代码和数据的映射不可见,因此发生 CPU 读缺失的几率高于方案一,会占用 DSP 一定的处理时间. 对三种方案的性能测试表明,第二种方案的运行速度明显高于其余两种. 因此,本文采第二种方案,如图 6 所示.

根据 Cache 访问的局部性原理,为了提高 Cache 的命中率,在实现中对代码结构进行合理的模块化调整. 将较大的算法分割成较小的模块,将过小的模块合并成大小适当的模块,将经常协同工作的代码段在物理位置上连续放置;同时对数据存放位置也进行合理分配,尽量将相关联的数据放置在一起.

表 2　基于 GMB 编码和 Cache 优化的运行速度比较

序列(352×288)	运行时钟周期	
	无优化	有优化
Basket	1573760883	79081115
Flower	1278872434	75421781
Coastguard	1978098817	83970092

表 2 为采用基于 GMB 结构的编码和方案二的 Cache 优化后的测试结果,在编码 Basket、Flower 和 Coastguard 序列时,执行速度比不进行优化平均提高了 20.14 倍.

5.3　代码级优化

一般情况下,在程序代码中 20% 的代码耗费了 80% 的时间[9],优化要抓住耗时多的关键代码段. CCS 编译环境提供了性能剖析工具(Profiler),可以帮助寻找代码的执行效率瓶颈,也可以启动 DSP 内部的 TIMER 获得代码段的运行时钟周期数,进而分析代码的运行效率. 本文主要从以下几方面来进行代码优化:

(1)展开循环: 循环体通常是系统运行最耗时的部分,应尽可能将循环展开,使用确定的循环次数性,在循环体内部不使用跳转指令数,可以获得更好的软件流水.

(2)使用逻辑移位: 乘除运算指令的执行时间要远远超过逻辑移位指令,所以在程序中要尽量使用逻辑移位操作来代替乘除法运算.

(3)使用高效的库函数和内联函数: DSPLIB 和 IMGLIB 为 TI 提供的基础库,其中有很多优化 TI DSP 的高效库函数,如二维 FDCT 和 IDCT 变换等库函数. 内联函数是 C6000 编译器提供的专门函数,它们与嵌入式的汇编指令是一一对应的,可以提高 C 源程序的执行速度.

(4)应用编译器选项: 在 TI 的 C6000 编译器 CCS2.0 环境下,还可以通过设置编译器的优化选项,让编译工具自动完成对 C 代码的优化. 在使用优化选项时,需要注意代码性能与代码尺寸的权衡.

(5)编写线性汇编: 经过 C 代码的优化之后,还不能满足性能上的要求,可以使用线性汇编重新改写使用率高的核心代码.

表 3　程序优化时钟周期对照表

函数	时钟周期数		提高倍数
	优化前	优化后	
Fdct	9342	220	42.46
Sad16	8760	132	66.36
interpolateH	16488	131	125.86
transfer_16to8copy	4540	37	122.70

表 3 中列出了几个函数优化前后的平均时钟周期数,以

MPEG-4 的插值函数 interpolateH 为例, 经过优化后, 该函数的循环核心代码的并行度大大提高, 优化后速度提高了 125.86 倍.

6 系统性能测试

硬件测试平台为基于本文第二章方案研制的 DM642 编码器板, 主频 600MHz. 优化前后的编码性能如表 4 所示. 从结果可以看到, 在优化前系统不能完成一路视频的实时编码工作, 优化后单路视频每秒的编码帧数达到 100 帧以上. 在实时采集和编码四路视频时, 即使当视频内容比较复杂(纹理丰富、运动剧烈) 的情况下, 总编码帧率仍可到 105~ 110 帧/秒, 证明本文设计实现的 MPEG-4 编码器完全能胜任四路 CIF 大小视频的实时编码.

表 4 优化前后编码器的测试结果(单路)

序列	优化前帧率(帧/秒)	优化后帧率(帧/秒)
Basket	0.064	112.4
Flower	0.089	114.8
Coastguard	0.056	105.3

需要指出的是, 本文提出了半像素运动估计、量化策略等改进技术, 以及资源利用和程序代码的优化方法, 具有较强的通用性. 算法改进技术适用于 JVT、AVS 等其他视频压缩标准, 优化方法可用于其他类型 DSP.

参考文献:

[1] Ville Lappalainen, et al. Performance of H.26L video encoder on general-purpose processor[J]. Journal of VLSI Signal Processing Systems, 2003, 34(3): 239- 249.

[2] Jeff Mc Veigh, et al. A software-based real-time MPEG-2 video encoder[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2000, 10(7): 1178- 1184.

[3] K Benkrid, et al. High level programming for FPGA based image and video processing using hardware Skeletons[A]. IEEE International Conference on Acoustics, Speech and Signal Processing[C]. Istanbul: IEEE Press, 2000. 3227- 3231.

[4] 赵保军, 等. 基于 FPGA 和 DSP 实现的实时图像压缩[J]. 电子学报, 2003, 31(9): 1317- 1319.
ZHAO Bao-jun, et al. Implementation of real-time 2D-DCT with FPGA and DSP[J]. Acta Electronica Sinica, 2003, 31(9): 1317-1319. (in Chinese)

[5] SPRU198. TMS320C6000 Programmer's Guide[DB/OL]. http://www.ti.com. 1999.

[6] 李方慧, 王飞, 何佩琨, 等. TMS320C6000 系列 DSPs 的原理与应用[M]. 北京: 电子工业出版社, 2003.

[7] ISO/IEC 14496-2, Committee Draft, Information Technology Coding of Audio-Visual Objects: Visual[S]. 1998.

[8] Li Bo, et al. A fast block matching algorithm using smooth motion vector filed adaptive search technique[J]. Journal of Computer Science and Technology, 2003, 18(1): 14- 21.

[9] 许小东, 徐佩霞. 基于 TMS320 DM642 的视频解码系统优化[J]. 数据采集与处理, 2005, 20(1): 91- 95.
XU Xiao-dong, XU Pei-xia. Optimization of video decoder system based on TMS320 DM642[J]. Journal of Data Acquisition & Processing, 2005, 20(1): 91- 95. (in Chinese)

作者简介:

李 波 男, 1966 年 8 月出生于四川省, 教授, 博士生导师, 主要研究方向为视频/图像压缩、视频理解、图像分析、智能信息处理和嵌入式图像处理系统. E-mail: boli@buaa.edu.cn

孟庆磊 男, 1980 年 2 月出生于黑龙江佳木斯, 博士研究生, 主要研究方向为视频压缩技术及嵌入式实现.

姚春莲 女, 1973 年 12 月出生于河北唐山, 博士研究生, 主要研究方向为视频采集、视频编码、运动检测.

姜 黎 女, 1982 年 7 月出生于湖南娄底, 硕士研究生, 主要研究方向为视频编解码技术.