ARM7 平台实时 MPEG-4 解码系统的实现与优化*)

郑晓博　陶　品
(清华大学计算机科学与技术系　北京 100084)

摘　要　本文分析了 ARM 处理器的结构特点，介绍了在 Freescale 公司 ARM7 平台 i. 250 处理芯片上实现 MPEG-4 实时解码的策略，针对解码器的优化特点和芯片的硬件结构，采用了算法级、C 语言级、ARM 级联合优化的优化方法，对于标准 MPEG-4 解码过程进行了优化。实验结果表明，在采用该芯片的手机平台中，对于 sub-QCIF 大小的图像系统达到了平均约 15fps 的解码速度，成功实现了 ARM7 平台下的实时 MPEG-4 解码。该项目正与公司合作，具有优化后实时 MPEG-4 解码系统的低成本手机有望实现商业化。

关键词　ARM7，Freescale i. 250，MPEG-4 解码器，优化与实现

Optimization and Realization of a Real-time MPEG-4 Video Decoder Based on ARM7

ZHENG Xiao-Bo　TAO Pin
(Depatment of Computer Science & Technology, Tsinghua University, Beijing 100084)

Abstract　This paper introduces the work of optimizing real-time MPEG-4 decoder based on i. 250 (ARM7 platform) produced by Freescale company. We realize Mpeg-4 decoder with arithmetic, C-language and ARM optimization method, according to the i. 250 architecture. The results indicate that we have successfully realized real-time MPEG-4 decoder base on the mobile phone platform using i. 250. The decoding rate is higher than the dominate products in the market. The low-cost mobile phone products using optimized MPEG-4 decoder will entere the market via corporation with company.

Keywords　ARM7, Freescale i. 250, MPEG-4 decoder, Optimization & realization

1　引言

伴随着手机等移动终端的普及，手机逐渐代替 PC 成为大众化的网络终端和信息处理设备，因此手机中集成的功能也越来越多，其中视频播放功能已经成为人们首先考虑的功能之一。现有的手机视频播放功能大多支持 H. 263/MPEG-4 视频编码标准。MPEG-4 是为新一代的多媒体应用而提出的基于对象的编码标准，其目标是制定一个通用的低比特率的压缩标准，通过基于内容的第二代编码方法的引入实现了低比特率编码。因此，商业化的生产采用 MPEG-4 视频播放平台的手机等移动终端成为消费类电子产品发展的趋势。

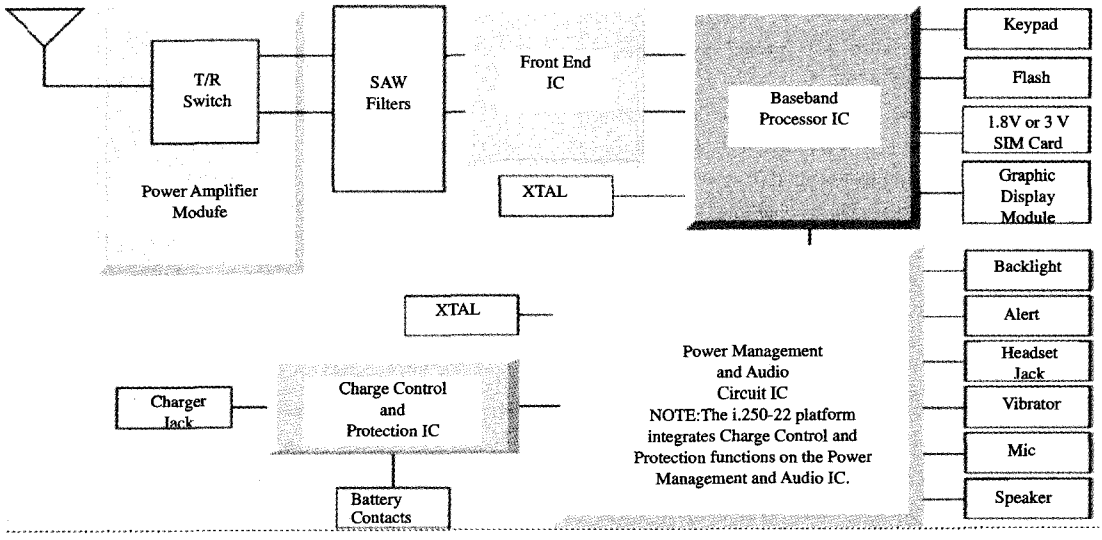


图 1　i. 250 平台主板结构图

另一方面，手机采用的嵌入式处理器是视频播放的关键。随着手机价格的不断下降，如何采用低成本、处理性能相对较低的嵌入式处理器，而又能实现手机上的 MPEG-4 和 H. 263 实时视频播放，成为具有视频播放功能的手机是否具有商业

*)本课题研究得到国家 863 科学基金(No. 2006AA01Z321)的资助。郑晓博　硕士生，主要研究方向为视频编解码技术；陶　品　副教授，硕导，主要研究方向为嵌入式媒体应用。

化竞争力的重要因素。

本解码器采用了低成本，处理性能较低的 freescale 公司的 i.250 芯片，对 MPEG-4 解码器的实现进行了优化处理，并在嵌入式平台中实现，达到了预定的播放性能。

2 系统硬件平台

本项目手机中的嵌入式处理器采用 Freescale 公司的 i.250 芯片。i.250 芯片平台是为 2.5G 设计，支持 GSM 和 GPRS。该芯片集成了 ARM7 内核，具有 37 个通用寄存器，工作频率为 52MHz，有 256k 片内 RAM。手机的工作电压为 4V。图 1 是基于 i.250 平台的系统结构图。

严格来说，ARM7 处理能力有限，更适合用于控制类型的应用，而不太适合于视频解码等数据处理类应用。i.250 芯片中的 ARM 核采用的是通用处理器架构，并没有针对视频解码数据处理而优化设计。但是由于该芯片具有明显的成本优势，所以经过优化，在充分利用其性能的前提下，还是可以作为手机等嵌入式系统的视频解码应用。

3 解码器优化

3.1 效率更高的 IDCT 变换

3.1.1 DC-AC 预判断

通常，MPEG-4 编码过程中 8×8 块在 DCT 变换后，AC 系数大都接近于零，经过量化后直接变成了零。同时根据帧间预测的相关性：在运动不是非常剧烈的情况下，量化后大部分 DCT 相关性是零。表 1 显示了在快速运动和慢速运动序列所有全零块的百分比。

表 1 64kbits/sec 的全零块百分比

序列(QCIF)	Fast motion	Slow motion
	Foreman	Mother-Daughter
全零块	24.8%	47.0%

对于快速运动序列，大约 25%的 DCT 块是全零的；对于慢速运动序列，全零块的百分比是大约 47%。这样，我们可以把 DCT 块分为下面不同的三类：一类是全零块(DC 系数和 AC 系数都是零)，一类是只含有 DC 系数(AC 系数是零)，一类是含有 DC 系数和 AC 系数。如图 2 所示(这里用 4×4 的块举例，D 代表 DC 系数，A 表示 AC 系数)。

然后对于不同的 IDCT 进行不同的处理：对于第一类情况，全零块，跳过反变换；对于第二类情况，只进行反 DC 变换。通常除以 8，即移 3 位即可；对于非零 AC 系数块，按照快速的 IDCT 处理。

这样就可以针对不同的情况采用不同的处理办法，提高了解码效率。

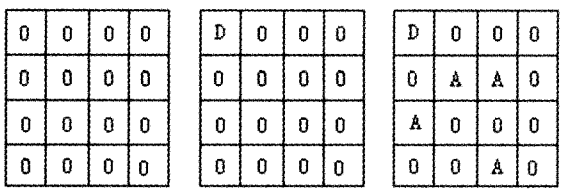


图 2 DC 和 AC 系数的分类

3.2 效率更高的运动补偿

3.2.1 运动补偿的扩边

MPEG-4 在进行运动补偿的时候，使用运动向量在参考图像中寻找预测块。如果运动向量变化比较快，运动向量很可能指向参考图像以外。MPEG-4 标准框架中，采用了很多和分支判断的语句来处理如果运动向量指向参考图像以外的情况。一方面 IF 语句的判断会降低程序的效率，造成解码过程速度的下降，另一方面如果运动向量没有指向参考图像以外，IF 判断就显得有些多余。为了提高解码效率，我们采用参考帧扩边的方式来解决。为了简便，提高效率，将参考图像的边界扩大的部分全部置零。这样就可以减少很多判断的语句，提高了解码效率。在实际中，运动向量的有效范围很大，但考虑到当运动向量使计算一个预测块所需的像素完全处于参考图像以外，那么不论运动向量的水平分量或者垂直分量延伸多远，所得到的预测块都是相同的。而运动补偿既可以基于块(8×8)的，也可以基于宏块(16×16)的，因此将扩展的字节数取为 16 就可以了，同时将运动向量的两个分量分别裁减到不超过参考图像左边和上边的边界 8 个字节以及下边和右边的边界 2 个字节。扩展后的参考图像见图 3。

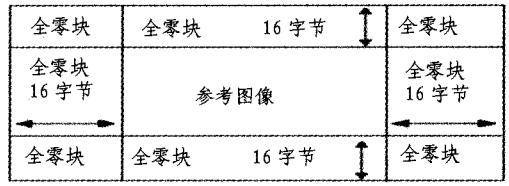
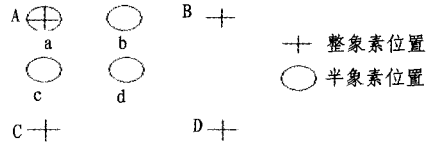


图 3 扩展后的参考图像

3.2.2 双线性插值的改进

MEPG-4 解码算法中，运动补偿是基于宏块为单位进行的。最初的做法是基于参考图像采用双线性插值，见图 4。因为要涉及很多的数据读取和计算，该模块比较耗时。



$$a = A$$

$$b = (A + B + 1 - rounding\_control)/2$$

$$c = (A + C + 1 - rounding\_control)/2$$

$$d = (A + B + C + D + 2 - rounding\_control)/4$$

图 4 半像素插值示意图

我们对这一做法进行了如下改进：对宏块的运动补偿是根据获得运动矢量进行不同的判断，而不是固定采用双线性插值的算法。运动补偿根据从解码数据中获得的水平运动矢量 MV_x 和垂直运动矢量 MV_y 来进行，即根据 MV_x 和 MV_y 最低位为 0 或 1 的情况，分为：只进行直接复制相应数据；或者只进行垂直方向插值；或者只进行水平方向插值；或者进行双线性插值。具体做法如下：

当 MV_x 和 MV_y 的最低位都为零时，运动矢量指向的 16×16 的块本身与缓冲区中的像素重合，这是不需要进行任何插值处理，直接复制相应数据。

当 MV_x 最低位为零而 MV_y 的最低位不为零时，运动矢量指向的 8×8 的块的点落在某列两相邻像素的中间，这时只需要进行垂直方向的插值。

当 MV_x 最低位不为零而 MV_y 的最低位为零时，运动矢量指向的 8×8 的块的点落在某行两相邻像素的中间，这时

只需要进行水平方向的插值。

当 MV_x 与 MV_y 的最低位均不为零时，运动矢量指向的 8×8 的块的点落在相邻四个像素的中心，这时必须同时进行两个方向的插值。

由于相邻帧之间具有很大的时间相关性，所以本帧和上一帧大部分数据是相同的。我们假设上面 4 中运动补偿情形各占 1/4，当进行水平或垂直插值的时候，运动补偿所占的运算量仅为原来的双线性插值的 1/2，我们比双线性插值约节省一半的计算量，从而大量节省了运动补偿的时间。

3.2.3 像素的并行处理

同时，解码过程中处理的像素是 8 位。如果运动补偿是在字节或像素的基础上执行，那么字节加载和存储将被使用，它是存储器访问中代价最高的操作。因为 ARM7 是 32 位微处理器，存储器可以按字读取数据，因此设计出一种有效的运动补偿方法，即在字数据的基础上进行操作。利用这种方法，便可以用一种非常有效的方式同时对 4 像素进行运动补偿。下面以水平方向的半像素补偿为例，讲述补偿的过程。

首先读入一个字到寄存器中，从低到高的数据依次对应的是像素 0、像素 1、像素 2 和像素 3；然后将读码流指针增加 1 字节，再读取下一个字到另一寄存器中，从低到高的数据依次对应的为像素 1、像素 2、像素 3 和像素 4。示意图如图 5 所示。然后按照补偿公式对上述两个寄存器进行相加移位操作。

图 5 像素并行处理原理示意图

对于垂直方向和水平垂直方向的半像素补偿，其原理与水平方向相同。

在具体函数实现过程中，有解码数据获得的当前数据块的运动矢量，根据获得的运动矢量得到当前数据块在参考帧的具体位置，从而得到运动补偿所需要的参考数据块。参考数据拷贝到片内。运动补偿在片内实行，按照字读取数据并根据情况采用不同的半像素插值，提高了程序的执行效率。

3.3 VLC 优化

由于 MPEG-4 变长编码中的码字长度是不定的，而解码器的输入是连续的比特流，码字之间没有间隔符，所以首先 VLC 码表必须判断码字的长度。在通常情况下，VLC 解码是通过不断搜索和判断，得到的码字和码长，故解码的时间因码长而异。对于实时处理来说，若该部分计算量过大，将影响整个系统的处理速度。

原始查表方法涉及到多次读取和判断，计算量较大。另外，信源符号内容不同，对应码长也不同，造成查表判断耗费的时间差别很大。我们采用基于分组的办法，根据码字编码位的不同划分为多个码表，将码字按照不同的区域进行了划分。这样，不断的搜索判断可以简化为三个步骤。第一步是读入定长码字，第二步是通过对读入数据大小的判断确定读入的符号应属于哪一个查找表，最后利用得到的码字在查找表中直接获得其对应的信息。因每个分组包含的符号较少，所以可在取出分组信息后，从剩下的信息位中直接得到符号在表中对应的位置。

经分组后，解码过程简化为：

（按最大码字长度读入数据，以 8 位数据为例，设分成码长小于 3 的小码表和码长大于 3 的大码表）

1）对读入数据进行大小判断。因分组时考虑到数据大小判断的简便性，再次可用移位代替。

2）数据大小的比较，右移 5 位，判断是否为 0，如果为 0，则符号落在码长小码表中；否则符号落在大码表中。

3）若符号落在小码表中，以右移 5 位的读入数据作为相对地址，直接在小码表中找到对应非零系数个数和+/−1 个数及码长。若符号落在大码表中，则直接以读入的数据为相对地址，在大码表中找到相应的信息。

无法预见的反复读取和判断经过基于分组的解码优化，简化成上述三个可预见的步骤，减少了判断次数，加快了处理时间。

3.4 数据缓冲区管理

本系统中，ARM 有自己的 256k 的存储空间 RAM。由于 ARM 访问片内 RAM 的速度比访问片外存储器快很多，为了提高效率，应该尽量将数据分配到片内 RAM，较少对片外内存访问。

为了提高执行解码的效率，我们为解码函数中的关键函数在片内 RAM 上划分了数据缓冲区给其调用。片上的数据缓冲区具体分配如下：

1）压缩码流输入缓冲区

由于片内存储量小，输入的压缩码流和解码后的视频输出都存储在片外。输入的压缩码流在解码启动的时候先放到片外保存，解码时候分批将数据复制到片内，片内设置一个压缩码流缓冲区，大小为 15kByte。

2）重建宏块数据缓冲区

重建宏块数据缓冲区，该缓冲区大小为 384（8×8×6）字节，用于在解码过程中保存当前解码宏块的重建数据。为了防止过多的片内与片外数据传输的请求，可以先将当前重建宏块数据复制到块组缓冲区中当前宏块位置，待整个块组解码完毕后，再将重建的块组数据一次性复制到重建帧缓冲区中。

3）运动补偿参考块缓冲区

片内还设置了一个运动补偿参考块缓冲区，由于片内存储容量小，不能缓存一帧参考图像，运动补偿时候，参考图像在片外，根据运动矢量在片外找到参考块（最大为 17×17），用 DMA 方式复制到片内参考块缓冲区缓存。因为 MPEG-4 中的运动补偿既可以是基于块（8×8）的也可以是基于宏块（16×16）的，所以缓冲区应取大些，既能用于存宏块，也能存块。选 17×17，比宏块还大一行一列是因为半像素插值精度需要进行水平或垂直插补运算，而插补运算需要用到相邻参考块中的一行或一列。

4）IDCT 变换缓冲区

IDCT 变换是以像素块为单位的。IDCT 变换要求输入的数据为 16 位，还需要额外一个块大小的缓冲区做中间处理，因此需要在片内数据 RAM 开一个大小为 896（128×（6+1））字节的 IDCT 缓冲区 idctbuffer。每个像素块需要 128（8×8×2）字节，并且各块在相应缓冲区中按其在宏块的顺序排列。

5）纹理解码输出缓冲区

设置了一个二维数组 Block［6］［64］，用来存放一个宏块纹理解码结果，把解码后的 DCT 系数存在此数组中，然后经

过 Zigzag 扫描，IQ-IDCT 处理，最后得到一个宏块结果。对于帧内宏块存放的就是解码后的结果，对于帧间存放的是纹理残差值。

同时将变长解码的码表，逆 zigzag 扫描表存放到片内，在片内还设置了全局变量，用来保存 VOP、VOL 的头信息。表 2 给出了片内 RAM 主要分配。

表 2 片内 RAM 缓冲区主要分配

全局变量	所占空间/Byte
变长解码表(VLD)	4906
Zigzag 扫描表	192
VOL、VOP 头信息	108
解码输出缓存区(一个宏块行)	8448
重建宏块数据缓存区	384
运动补偿参考块缓存区	289
IDCT 反变换缓存区	896
宏块纹理解码缓存区	384
输入压缩码流缓存区	15K
合计	45.3K

4 实验结果

我们的实验平台是使用 Freescale 公司 i.250 芯片的 GSM 手机，经过上述优化，对 Sub-QCIF(128×96)大小的 MPEG-4 视频文件进行了播放速度测试，结果如表 3。

表 3 实验结果

视频序列	运动强度	播放帧数(f)	播放时间(ms)	解码速度(f/s)
序列 1	中等	974	63169	15.42
序列 2	低	1018	62970	16.17
序列 3	高	487	35222	13.83
序列 4	高	149	12922	11.45
序列 5	中等	1207	67767	17.81
序列 6	高	487	35309	13.79

实验结果表明，经过优化的视频解码解码器在主频为 52MHz 的 ARM7 嵌入式处理器核上，解码速率基本达到了实时效果。在基于低端嵌入式处理器(freescale 公司的 i.250 芯片)平台下所设计的最终手机产品中，实现了低分辨率的嵌入式 MPEG-4 视频实时解码。

通过与 2006 年 Philips 公司所公布的嵌入式播放器 Obigo Media Player 相比(注：Philips 公司所采用的硬件是 ARM7TDMI，52 MHz，256k RAM，Sub-QCIF MEPG-4 decoding in>10fps)，在处理能力基本相近的处理器平台上，本文所设计的系统其解码速度已经达到甚至超过了 Obigo Media Player 的处理水平。

参 考 文 献

1 ISO/IEC. MPEG-4 Video Verfication Model version 18.0. MPEG N3908, 2001

2 钟玉琢，王琪，贺玉文. 基于对象的多媒体数据压缩编码国际标准—MPEG-4 及其校验模型. 科学出版社，2000

3 Talluri R. Error Resilient Video Coding in the ISO MPEG-c4 Standard. IEEE Communications Magazine, 1998, 36(6): 112～119

4 He Yuwen. A Platform-based MPEG-4 Advanced Video Coding (AVC) Decoder with Block Level Pipelining. PCM2003: 15-18, Singapore. December 2003

5 Sloss A N, Symes D, Wright C. ARM System Develoer's Guide: Designing and Optimizing System Software. 北京航空航天大学出版社，2002

6 田纲，胡瑞敏，王中元，等. Trimedia 平台 MPEG4 编码器优化策略. 计算机工程与应用，2006(36): 78～81

7 顾梅花，张太镒. 基于 ARM 的 MPEG4 视频解码器. ARM 应用技术论文大奖赛

(上接第 232 页)

6 Salzman T, Smith G, Stuerzlinger W. Constraint Based 3D Scene Construction: [Master dissertation, York University]. 2000

7 Stuerzlinger W, Smith G. Efficient Manipulation of Object Groups in Virtual Environments. IEEE Virtual Reality, 2002

8 Xu Ken, Stewart J, Fiume E. Constraint-based Automatic Placement for Scene Composition. In: Graphics Interface, 2002. 25～34

9 Ehret B D, Microsystems S. Learning Where to Look: Location Learning in Graphical User Interface. In: Spatial Cognition Minneapolis, Minnesota, USA, 2002. 20～25

10 Adorni G, Di Manzo M, Giunchiglia F. Natural Language Driven Image Generation. COLING 84, 1984. 495～500

11 Clay S R, Wilhelms J. Put: Language-based Interactive Manipulation of Objects. In: IEEE Computer Graphics and Applications, March 1996. 31～39

12 Kingston. Interactive Manipulation of Articulated Objects with Geometry Awareness. In: Proceedings of the 1999 Conference on Graphics Interface, May 1999. 592～598

13 Coyne B, Sproat R. Wordseye: An Automatic Text-to-Scene Conversion System. In: ACM SIGGRAPH, 2001. 487～496

14 Dupuy S, Egges A, Legendre V, et al. Generating a 3D Simulation of a Car Accident from a Written Description in Natural Language: the CarSim System. In: Proc. ACL Workshop on Temoral and Spatial Information Processing, 2001. 1～8

15 Tappan D. Knowledge-based Spatial Reasoning for Automated Scene Generation from Text Descriptions: [Ph D dissertation, New]. Mexico State University, 2004

16 Bernier F, Boivin E, Laurendeau D, et al. Conceptual Models for Describing Virtual Worlds. In: WSCG (Posters), 2004. 25～28

17 陆汝钤，张松懋. 从故事到动画片. 自动化学报，2002，28: 321～348