文章编号:1004-9037(2012)06-0724-06

AVS 视频编解码算法在 TMS320C6455 上的实现

王 群 张 涛 张瑞生 全浩军 杨 雪

(天津大学电子信息工程学院,天津,300072)

摘要:在 TMS320C6455 上实现了音视频编码标准(Audio and video coding standard,AVS)视频标准标清视频的实时编码,高清视频码流的实时解码。以 AVS 参考代码 RM52i 为软件基础,综合考虑 TMS320C6455 的性能和 AVS 视频编解码器的实现方式,采用了系统结构优化、线性汇编优化、存储空间优化、EDMA 辅助数据搬移等多种优化措施,最终实现了对 720×576 分辨率视频的实时编码和 1 920×1 080 分辨率高清码流的实时解码。

关键词:编解码算法;音视频编码标准;优化实现

中图分类号:TN919.81 **文献标识码**:A

AVS Video Codec Implementation on TMS320C6455

Wang Qun, *Zhang Tao*, *Zhang Ruisheng*, *Quan Haojun*, *Yang Xue*

(School of Electronic and Information Engineering, Tianjin University, Tianjin, 300072, China)

Abstract: A real-time standard resolution encoder and a real-time high resolution decoder are implemented on TMS320C6455 of Texas Instruments. It is based on the referenced standard code RM52i. considering the hardware performance and the audio and video coding standard (AVS) video standard, a variety of methods are adopted to optimize the codec, including system architecture adjustment, assembly optimization, memory space allocation and EDMA data transit. With the proposed method, a real-time 720×576 video encoder and a real-time 1 920×1 080 video decoder running at 25 frames per second are achieved eventually.

Key words: codec algorithm; audio and video coding standard (AVS); optimized implementation

引 言

视频压缩编解码是数字多媒体处理的关键技术,是各种视频应用如数字电视系统、视频会议等的核心模块。音视频编码标准(Audio and video coding standard,AVS)是中国具备自主知识产权的第二代信源编码标准,由中国音视频编解码标准化工作组制定,已于 2006 年被批准颁布为国家标准[1]。AVS 面向高分辨率视频应用,与相关国际标准 H.264 相比,兼顾了编码效率和实现复杂度,并通过一站式许可政策,解决了 H.264 专利许可问题,便于应用推广。AVS 编解码技术的研究,对于中国相关产业领域的长远发展具有重要意义,研究实现支持 AVS 的编解码器,有助于推动 AVS 的产业化,具有极大的应用价值。

视频编码的实时实现对处理器的计算能力和资源提出了很高的要求。通用处理器不适于做复杂计算高度集中的运算,基于硬件实现的方案开发周期长且缺乏灵活性。本文选用 TI 公司的 TMS320C6455 高性能 DSP 芯片作为 AVS 视频编解码器的实现平台。TMS3206455 是 TI 公司推出的高速定点 DSP,采用了 VelociTI VLIW 结构 C6x+内核,最高时钟频率为 1.2 GHz,32 位定点处理能力可达到 9 600 MIPS。片内采用两级高速缓存结构,其中 L_2 有 2 MB 的 RAM 数据空间可供使用。TMS320C6455 还具有强大的外部存储器接口 EMIF,可以连接 DDR2 等高速外部存储器[2]。TMS320C6455 具有的这些特性可以满足视频处理算法的需求,非常适合作为这类算法的硬件

基金项目:国家自然科学基金(61179045/F01)资助项目;国家重大科技专项(2010ZX03004-003-03)资助项目。

收稿日期:2012-06-16;**修订日期**:2012-10-12

处理平台。

1　算法简介

AVS 属于第二代视频编码标准,采用了混合编码框架(图 1),主要运用了预测、变换、量化、熵编码等技术。AVS 在编码框架的各关键技术模块中都引入了新的技术,从而兼顾了编码效率和编码复杂度。

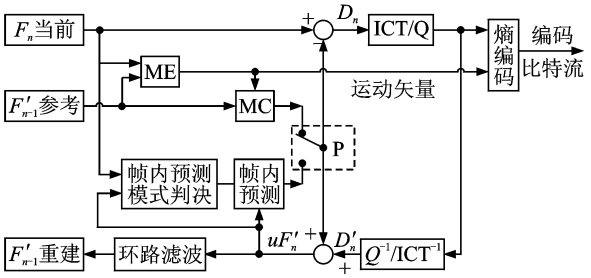

图 1　AVS 编码系统

1.1　帧内预测

帧内预测是利用已知的相邻值来预测当前值,然后对当前实际值和预测值的差进行编码,以去除图像空域冗余。在 AVS 标准中,帧内预测以 8×8 块为单位,以未经环路滤波的当前重建帧同一条带内邻近的左边和上边已编码的像素值来预测当前块的像素值。为了更好地适应图像的纹理特征,AVS 规定了 9 种帧内预测模式,并通过模式判决选择最佳的预测模式进行编码。其中,亮度有 5 种模式,按预测方向分别是直流、水平、右下、垂直和左下,如图 2 所示;色度有 4 种模式,与亮度块类似,分别为直流、水平、垂直和平面模式。

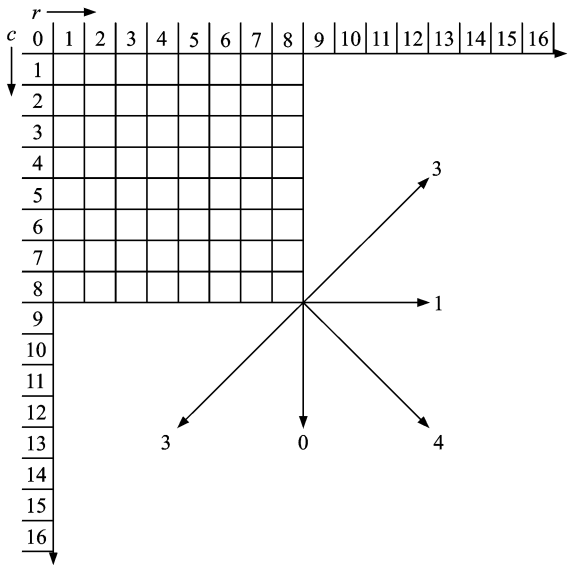

图 2　8×8 亮度块帧内预测模式

1.2　帧间预测

AVS 帧间预测采用了多种技术来提高预测精度,包括多种宏块划分、分像素运动估计和补偿、多个前向参考帧以及运动矢量预测技术。帧内预测模式如表 1 所示。

表 1　8×8 亮度块帧内预测模式

帧内亮度预测模式	名称
0	帧内 8×8 垂直
1	帧内 8×8 水平
2	帧内 8×8 直流
3	帧内 8×8 左下
4	帧内 8×8 右下

(1)多种宏块划分　AVS 每一个宏块(16×16 像素)可以按 4 种方式进行分割:1 个 16×16,或 2 个 16×8,或 2 个 8×16,或 4 个 8×8。这种划分用于运动补偿。

(2)分像素运动估计和补偿　AVS 可对亮度进行 1/4 像素精度的运动估计和补偿,对色度进行 1/8 像素精度的运动补偿,从而提高运动估计的精度,减小预测残差。

(3)运动矢量预测　由于宏块内部各子块的运动矢量具有相关性,AVS 标准中对运动矢量采用了预测编码,只传输运动矢量差值以降低码率。

1.3　整数变换

由于运动估计的最小块是 8×8,AVS 采用了 8×8 变换矩阵以最优化编码效率。为了解决传统的浮点离散余弦变换(Discrete cosine transform,DCT)在编码端和解码端由精度不匹配而导致的误差漂移问题,AVS 采用了整数 DCT 替代浮点 DCT,变换矩阵为

$$\boldsymbol{T}^{\mathrm{T}} = \boldsymbol{T}^{-1} = \begin{Bmatrix} 8 & 10 & 10 & 9 & 8 & 6 & 4 & 2 \\ 8 & 9 & 4 & -2 & -8 & -10 & -10 & -6 \\ 8 & 6 & -4 & -10 & -8 & 2 & 10 & 9 \\ 8 & 2 & -10 & -6 & 8 & 9 & -4 & -10 \\ 8 & -2 & -10 & 6 & 8 & -9 & -4 & 10 \\ 8 & -6 & -4 & 10 & -8 & -2 & 10 & -9 \\ 8 & -9 & 4 & 2 & -8 & 10 & -10 & 6 \\ 8 & -10 & 10 & -9 & 8 & 6 & 4 & -2 \end{Bmatrix}$$

整数 DCT 在性能降低有限的情况下克服了匹配误差对视频质量的影响,同时提高了计算效率。

1.4 熵编码

AVS 标准采用的是基于指数哥伦布码(Exponential-golomb codes,E-G)的统计编码技术。其中运动矢量预测残差、参考帧索引、块编码模式等除量化系数外的语法元素均用 0 阶 E-G 码或定长码进行编码,对量化系数则采用基于上下文的二维可变长码(Context-based 2-dimension VLC,C2DVLC)编码。E-G 码相对于经典熵编码,如 Huffman 码的优点在于码表结构简单,易于构造,可以降低编解码的复杂度,且通过选择合适的阶数进行编码同样能接近信源的熵。k 阶的 E-G 码的码字结构如图 3 所示,图中 leadingzeroBits 为前导 0 的个数。

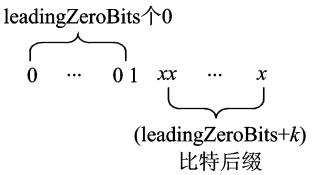

图 3 k 阶 E-G 码的码字结构

AVS 的上下文模型针对 8×8 变换,对游程与幅值对(Run,Level)进行编码。为了适应局部的差异性设计了共 19 个变长码表(帧间亮度块 7 个,帧内亮度块 7 个,色度块 5 个),根据之前幅值增大的趋势来决定当前使用的码表,极大提升了变长编码的效率。

1.5 快速运动搜索

AVS 标准未规定运动搜索的方法,本文采用了非对称十字多六边形搜索(Unsymmetrical-cross multi-hexagon search,UMHexagonS)作为整像素搜索快速算法,这是一种混合搜索算法,提高预测有效性的同时保证了足够的健壮性。

2 算法在 DSP 上的实现与优化

2.1 DSP 平台的选择

AVS 视频编解码算法运算量很大,需要 DSP 有足够强的处理能力。此外,编码器、解码器设计诸多变量,还要为参考帧等开辟空间,这就要求所选 DSP 有足够的存储空间供算法使用。TMS320C6455 是 TMS320C6000 平台中一款高性能定点数字处理器,基于 TI 先进的第三代高性能 VLIW 架构,可应用于视频处理、无线通信等领域。TMS320C6455 可在 1.2 GHz 时钟频率下每秒处理 96 亿指令,这样强大的运算能力,使 TMS320C6455 成为实时多媒体处理和图像处理的理想平台之一。TI 还提供了一套完整的编译开发工具集方便开发调试,为使用 TMS320C6455 进行高效视频处理提供了便利。

2.2 实现与优化方案

AVS 视频编解码算法具有复杂度高、运算密集、对存储空间需求大等特点,且参考代码是为测试标准的性能而设立的,支持标准中定义的所有技术,且包含了大量的测试用信息,功能上存在冗余,结构上不适合 DSP 处理[3]。虽然 TMS320C6455 具有很强的处理性能,但距离实现 AVS 标清视频实时编码或高清实时解码仍有较大距离。为了实现编解码算法的顺利移植,必须综合考虑 AVS 视频编解码算法和 TMS320C6455 的结构特点,有针对性地进行算法改进和代码优化。

优化工作主要分为编解码算法实现的优化和 DSP 平台相关优化两个阶段。

2.2.1 编解码算法实现优化

编解码算法实现的优化是指在不改变 AVS 所用的编解码算法前提下,通过改变算法实现方式,从而达到减少运算量、减少分支判断、提高处理速度的目的。

(1)比特操作的优化

编解码器中涉及大量的比特操作,如熵编码中,绝大部分码字长度非整字节长(8 bit),在写码流时每次写入的起点不一定是字节的边界,需要对要写入的码字进行调整、拼接,以组成连续的码流。通常使用的方法是逐比特提取、逐比特写入码流,这种方式效率低下,不适合在 DSP 上实现。为了提高效率,应一次写入尽可能多的比特。考虑到熵编码中的码字长度必然少于 32 位,可用一个 unsigned int 型变量作为码字缓存,写入码流时先判断码流缓冲区中待写字节的剩余比特数,若剩余比特数大于码字长度,则直接将码字移位后填在码流尾部;否则先从码字中截出部分比特,填满待写字节,此后码流缓冲区即为整字节对齐的,将码字剩余部分利用位移对齐到字节边界,按字节拷贝到码流缓冲区中即可。

(2)使用函数指针数组加速分支选择

在 AVS 解码过程中,经常会用到多个分支条件选择的情况。例如分像素插值的计算,需要根据 dx 与 dy 变量选择计算对应像素的插值。而当最后一个分支情况满足(即计算位置 m 及对应位置的 1/4 插值像素数据)时,需要进行至少 12 次判断才

能进行处理。由于像素插值的计算需要多次反复执行,因此过多的条件判断严重影响了解码的速度。而使用函数指针数组则可以简化判断流程。

在函数指针数组中,数组的维数由分支数目决定,而每个元素所对应的函数都是针对某一种情况的数据处理。这样当需要根据不同情况进行处理时,只需要由每种情况对应的数字在数组中检索对应的函数即可。这样,通过检索函数指针数组便可直接进入到对应的条件分支,进行相应的计算。因此大大简化了判断流程,提高了解码速度。

(3)指数哥伦布码编解码优化

指数哥伦布码编解码的关键在于确定码字的长度和码字取值。观察指数哥伦布码的码字结构可以发现,指数哥伦布码取值 GolombValue 和编码值 CodeNum 之间满足关系 GolombValue=CodeNum+2^k。据此,文献[4]中提出了一种新的指数哥伦布码编解码算法。在编码时,首先由 CodeNum 和 k 求出 GolombValue,再由 GolombValue 的二进制表示得到前缀 0 的个数 leadingZeroBits,则最终码字长度为(leadingZeroBits * 2+k+1)位,码字的后几位为 GolombValue 的二进制值。相应的,在解码时,首先从比特流的当前位置开始找到第一个非零比特,记连零个数为 leadingZeroBits,再将第一个非零比特及其后面(leadingZeroBits+k)比特赋值给 GolombValue,最后得到解码后的值 CodeNum。

使用以上方法和逐比特计算的编解码算法相比,编码时间平均可缩短 10%,解码时间平均可缩短 30%。

(4)使用数组指针优化内存访问

在编码器端,编码是以宏块为单位进行的。对于每个 INTER 帧的宏块,编码时需要获取参考帧中以其对应位置为中心,80×80 像素大小的数据作运动估计。若采用从参考帧中逐行拷贝的方式,80×80 大小的数据需要循环 80 次完成,且每次搬运数据量小,不适于用增强型直接内存存取(Enhanced direct memory access,EDMA)实现。

本文采用优化的方法,使用数组指针来访问参考帧数组,边界扩展后的参考帧每行包含 784 个像素,故可用一个 byte(*)[784]型的数组指针指向 80×80 参考块的左上角像素,这样数组指针每增加 1,实际上在参考帧数组中移动了 784 byte 大小,即指向了 80×80 块下一行的起点,再通过数组下标即可访问每行的像素。通过数组指针访问极大简化了对参考帧数据的访问,实验结果表明可获得很好的时间增益。

2.2.2　DSP 平台相关优化

DSP 平台相关的优化主要是针对 TMS320C6455 的结构,优化程序以充分利用 DSP 的处理能力和硬件资源。主要包括存储空间优化、使用内联函数、用线性汇编改写关键函数、利用 EDMA 加速数据搬移等几个方面。

(1)存储空间优化

TMS320C6455 片内为两级缓存结构,其中 L_2 SRAM 大小为 2 048 KB,且能将其中的部分空间配置成 Cache 使用。TMS320C6455 最高可支持 512 MB 的片外 DDR2 SRAM。片上 SRAM 存取速度极快,但容量较小;片外 DDR SRAM 容量大,但存取速度相对慢。由于片内空间有限,应将调用次数多、使用频繁的数据放在片内,而将不经常使用的数据,特别是大块数据放在片外,待需要时搬到片内,从而提高程序/数据存取速度,同时降低 Cache 的 miss 概率,减少片内-片外数据的通信消耗。本文在 AVS 编解码器均做了数据存储空间的优化,主要包括输入输出时的 Ping-Pong 缓冲区使用、片内缓存编码信息等。

(2)利用内联函数进行并行运算

在进行规则的数据处理时,可以使用 C6000 提供的内联函数来对数据并行运算。例如在 AVS 解码器中,将参考图像数据与残差数据叠加得到重建的图像块的算法:

```
for(yy=0;yy<8;yy++)
    for(xx=0;xx<8;xx++)
    {
        curr_val=imgmpr[yy][xx]+globaldata_m78[locationSelect][yy][xx];
        globaldata_buffer_y_frm[b8_y_3+yy][pix_x_b8_x+xx]=Clip3(0,255,curr_val);
    }
```

其中 imgmpr 数组是由参考帧获得的图像参考块数据,globaldata_m78 是反变换得到的残差数据。代码的核心循环需要进行 64 次累加和截断操作。而使用内联函数进行并行运算时,使用_amemd8 来读取数据,之后分别取出高位和低位进行对应的相加运算。该代码的核心循环部分重复 16 次(参考块像素与残差值相加 32 次)。实验结果表明,使用内联函数进行并行计算的计算速度明显

提升。

(3)使用线性汇编优化

通过前面对 AVS 编解码器中各函数的分析发现,用于计算 IDCT 的函数 inv_transform_B8 在编解码器中均被频繁调用,累计执行时间占编解码器各自总时间比重较大。IDCT 是编解码器中必不可少的环节,每个 8×8 块的运动残差数据都要通过 IDCT 获得,对其进行优化能极大减少运算时间。AVS 中的二维 IDCT 运算可通过两次一维变换实现,其蝶形快速算法如图 4 所示。

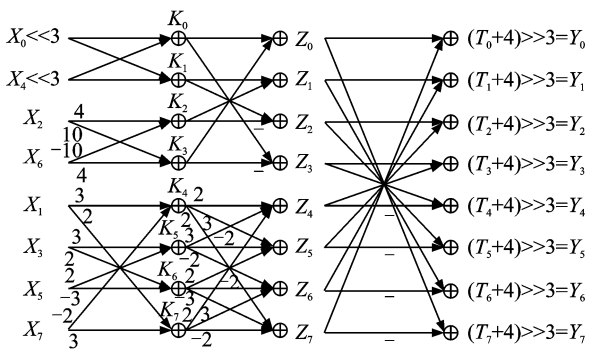

图 4 AVS 一维 IDCT 蝶形图

IDCT 中含有大量的循环和较规则的乘加运算,使用线性汇编对其进行改写,能充分利用 TMS320C6455 的多个功能单元和打包数据处理指令,提高指令并行度,从而减少函数执行时间[5]。在改写线性汇编的过程中,通过以下几个方面提高了指令的并行程度[6-7]:

①打包数据处理指令的使用;

②优化数据读取,减少数据依赖;

③根据功能单元占用情况优化指令选择;

④多块 IDCT 同时计算。

通过以上方法,用线性汇编改写后的 inv_transform_B8 函数,耗时由原来的 429 cycle/block 减少到 726 cycle/(6 block)。

3 性能与结果

经过本文所述的优化步骤,最终在 TMS320C6455 上实现了 AVS 视频编解码算法。编译环境为 CCS v4.0.1.01001,硬件平台为 DSK 6455。

(1)编解码算法实现性能分析

统计某种算法在 DSP 上运行的耗时通常有两种方式:①利用 CCS 的性能剖析工具 Profile 统计程序运行消耗的时钟周期数;②在程序中调用 CSL 库函数统计运行耗时。前一种方法相对精确,但统计过程花费时间较长,后一种方法使用简单,和前一种方法相比结果误差极小。因此,本文采用第二种方法来统计编解码算法在 TMS320C6455 上运行耗时,结果如表 2,3 所示。表 2 所列为编码 D_1 分辨率视频序列的结果,表 3 所列为解码 1 080 分辨率码流的耗时统计。从表中每帧耗时可见,编码器和解码器均已达到实时,但部分序列编码后码率较大。

(2)片内存储空间占用情况

经过优化后,编码器和解码器各自的片内空间占用情况为:

(a)编码器 L_1D:32 KB Cache;L_1P:32 KB Cache;L_2: 256 KB Cache,1 792 KB SRAM;L_2 SRAM 中,代码空间:95 KB,数据空间:1 338 KB,剩余:359 KB。

(b)解码器 L_1D:32 KB Cache;L_1P:32 KB Cache;L_2: 256 KB Cache,1 792 KB SRAM;L_2 SRAM 中,代码空间:89 KB,数据空间:1 246 KB,剩余:457 KB 片内空间占用情况示意图如图 5 所示,片内空间仍有一定剩余,为进一步改进提供了可能。

表 2 编码器耗时统计

序列	平均每帧耗时/us	SNR Y/dB	码率/(Kbit·s^{-1})	序列特性
Door	30 604	38.26	1 132.42	简单场景
Flower garden	30 640	33.51	2 663.72	旋转&平移
Basketball	31 940	32.85	2 897.62	人物运动剧烈
Ping pong	29 697	31.51	2 398.97	镜头缩放
Horse riding	39 237	33.49	3 096.01	大面积快速运动

表 3 解码器耗时统计 μs

序列	平均每帧耗时
Kimono	40 114
Running people	40 535
Park scene	40 218
Sunflower	39 191

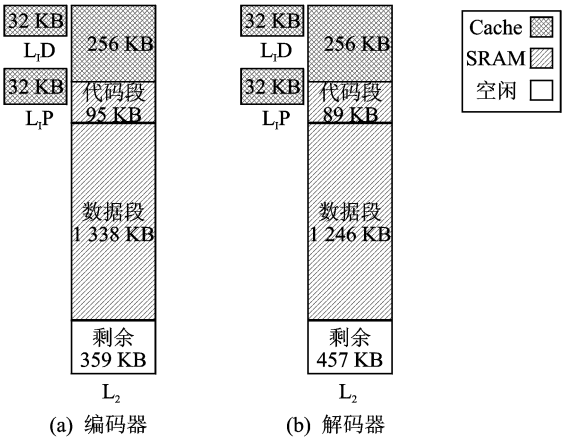

图 5　编解码器内存占用情况示意图

4　结束语

本文将 AVS 参考代码 RM 移植到 DSP 平台上,充分利用 TMS320C6455 的运算性能和硬件资源,通过算法优化和针对 DSP 平台的优化,在单片 TMS320C6455 上实现了 720×576 分辨率标清视频的编码器或 1 920×1 080 分辨率高清视频的解码器。实验数据表明,对于常见的视频序列,编码器和解码器均能达到实时处理,且性能有进一步提升的空间。

参考文献:

[1] AVS工作组. GB/T 20090.2-2006,信息技术先进音视频编码第二部分:视频[S]. 北京:中国标准出版社,2006.
AVS Workgroup. GB/T 20090.2-2006, information technology-advanced audio and video coding, Part 2: video[S]. Beijing: China Standard Press, 2006.

[2] Zhang Xiongkui, Liu Guoman, Gao Meiguo. A high-performance scalable computing system for realtime signal processing applications[C]//Congress-son Image and Signal Processing. Sanya, China: IEEE Press, 2008:556-560.

[3] 牛旭宁,张远,马思伟,等. AVS 解码器复杂度分析[J]. 计算机应用与软件,2010,27(5):67-70.
Niu Xuning, Zhang Yuan, Ma Siwei, et al. AVS decoder complexity analysis[J]. Computer Application and Software, 2010,27(5):67-70.

[4] 文斌,何明华,黄敏琪. 一种新的 AVS 指数哥伦布码算法[J]. 计算机工程,2011,37(15):218-220.
Wen Bin, He Minghua, Huang Minqi. A new exporent golomb coding algorithm for AVS[J]. Computer Engineering, 2011,37(15):218-220.

[5] Texas Instruments Incorporated. TMS320C64x/C64x+DSP CPU and instruction set reference guide [M]. Dallas: Texas Instruments Incorporated, 2010.

[6] 欧阳万里,肖创柏,刘广. 一种新的基于 VLIW 的 IDCT 和运动补偿算法[J]. 电子学报,2005,33(11):2074-2079.
Ouyang Wanli, Xiao Chuangbo, Liu Guang. A new IDCT and motion compensation algorithm based on VLIW[J]. Journal of Electronics, 2005,33(11):2074-2079.

[7] 李学明,李继. 用超长指令实现 DCT 的新算法[J]. 电子学报,2003,31(7):1074-1077.
Li Xueming, Li Ji. New algorithm to implement DCT with VLIW[J]. Journal of Electronics, 2003,31(7):1074-1077.

作者简介:王群(1987-),女,硕士研究生,研究方向:视频编解码技术,E-mail:wq_051@163.com;张涛(1975-),男,副教授,研究方向:数字音视频处理与 DSP 应用;张瑞生(1987-),男,硕士研究生,研究方向:音视频编解码、DSP 优化技术;全浩军(1984-),男,博士研究生,研究方向:数字音视频编解码、软硬件协同设计及 DSP、FPGA 应用;杨雪(1986-),女,硕士研究生,研究方向:视频编解码和 DSP 实现。