第 42 卷 第 5 期 计 算 机 工 程 2016 年 5 月
Vol. 42 No. 5 Computer Engineering May 2016

·多媒体技术及应用· 文章编号:1000-3428(2016)05-0249-04 文献标识码:A 中图分类号:TP37

基于 CUDA 的 H.264 并行解码器设计与实现

陈 勇,吴晓民,杨 坚,奚宏生

(中国科学技术大学自动化系,合肥 230027)

摘 要: 针对 H.264 视频编解码标准复杂度高、运算量大的缺点,基于统一计算设备架构平台设计并实现 CPU + GPU 异构并行H.264解码器,利用 GPU 的并行计算能力和 CPU 的逻辑控制优势加快运行速度,提高解码性能。实验结果表明,与 FFmpege 中传统的串行解码器相比,利用 GPU 加速的 H.264 并行解码器能获得 2 倍 ~7 倍的性能提升,各并行单独模块也可实现 5 倍 ~11 倍的加速。

关键词: 图形处理器;统一计算设备架构平台;H.264 标准;视频编解码器;并行化

中文引用格式: 陈 勇,吴晓民,杨 坚,等. 基于 CUDA 的 H.264 并行解码器设计与实现[J]. 计算机工程,2016,42(5):249-252,257.

英文引用格式: Chen Yong,Wu Xiaomin,Yang Jian,et al. Design and Implementation of H.264 Parallel Decoder Based on CUDA[J]. Computer Engineering,2016,42(5):249-252,257.

Design and Implementation of H.264 Parallel Decoder Based on CUDA

CHEN Yong,WU Xiaomin,YANG Jian,XI Hongsheng

(Department of Automation,University of Science and Technology of China,Hefei 230027,China)

【Abstract】 Aiming at the defects of H.264 such as high complexity and large amounts of computation,this paper designs and implements the CPU + GPU heterogeneous parallel H.264 decoder based on Compute Unified Device Architecture(CUDA),which makes full use of GPU's parallel computing ability and CPU's logic control advantages,to improve the running speed and decoding performance. Experimental result shows that the performance is improved 2 ~7 times when using GPU acceleration,and the each parallel module can also accelerate 5 to 11 times,compared with that of the traditional serial decoder in FFmpeg.

【Key words】 Graphics Processing Unit(GPU);Compute Unified Device Architecture(CUDA);H.264 standard;video codec;parallelization

DOI: 10.3969/j.issn.1000-3428.2016.05.043

1 概述

H.264 是由 ISO/IEC 动态图像专家组(Moving Picture Experts Group,MPEG)和 ITU-T 视频编码专家组(Video Coding Experts Group,VCEG)联合组成的联合视频组(Joint Video Team,JVT)提出的高度压缩数字视频编解码器标准[1]。为消除大量的时域冗余和空域冗余[2],H.264 对视频进行预测、变换和量化等处理[3]。H.264 编码或解码视频主要分为 5 个步骤:预测,变换,量化,环内去块滤波和熵编/解码[4]。在同等带宽条件下,与现有视频编/解码标准相比,H.264 能够提供质量更好的图像,但这样的优越性是以增加计算复杂度为代价的,其解码的复杂度是 MPEG-4 的 3 倍[5]。

图形处理器(Graphics Processing Unit,GPU)在并行计算方面表现出比 CPU 更优越的性能,NVIDIA 公司的编程并行计算架构——统一计算设备架构(Compute Unified Device Architecture,CUDA)的推广也使得 GPU 编程更规范与高效。在 CUDA 编程模型中,CPU 执行逻辑性强的事务处理和串行计算,GPU

基金项目: 国家自然科学基金资助重点项目"三网融合业务接入系统的分析、建模与调控"(61233003);国家自然科学面上基金资助项目"基于在线测量的网络化媒体服务系统自适应优化与控制"(61174062)。

作者简介: 陈 勇(1989 -),男,硕士研究生,主研方向为视频编/解码、实时视频传输与质量优化;吴晓民,博士研究生;杨 坚,副教授、博士生导师;奚宏生,教授、博士生导师。

收稿日期: 2015-04-10 **修回日期:** 2015-05-14 **E-mail:** cyong@mail.ustc.edu.cn

执行可以高度并行化的数据运算,并且 GPU 拥有自己独立的存储空间,操作它们需要调用 CUDA API 中专用的存储器管理函数,这些函数包括开辟、初始化和释放存储空间、主机与设备数据传输等相关操作。在数值计算、流体模拟、气象预测等数据量大、并发性高的通用计算领域,高并行性的可编程 GPU 已经得到了广泛应用[6]。同时它作为一种流处理器,也非常适合视频编/解码这种典型的流处理应用。

H.264 解码的时间花费主要在熵解码、反变换/反量化、帧内/帧间预测、去块滤波这 4 个模块上[7]。测试结果表明,熵解码占整个解码时间的比例低于 30%,而后三者时间总和高于总时间的 70%。熵解码的处理过程是从网络抽象层单元(Network Abstract Layer Unit,NALU)中按标准顺序读取码流进行解析,程序分支较多,存在大量的逻辑判断,并且解析码流是顺序串行处理,难以做到在 GPU 上进行并行加速[8],在本文中暂不考虑对熵解码模块进行并行加速;在反变换/反量化模块、帧内/帧间预测模块和去块滤波模块解码时,帧与帧之间、宏块与宏块之间可以做到独立处理,具有数据级并行的特点。为此,本文提出基于 CUDA 实现宏块级别 H.264 并行解码的方法,并以 FFmpeg 中提取出的串行H.264 解码器作为参考进行设计与实现。

2 并行方案设计

2.1 FFmpeg 中 H.264 解码器的工作流程

FFmpeg 在帧级别利用多线程实现了解码器的并行工作,本文的设计思路是实现同一帧不同宏块之间的数据级并行解码,因此,在本文方案中并没有考虑多线程并行加速。下面首先分析 FFmpeg 解码器对一帧图像解码的流程。

本文考虑对编码时 profile 采用 baseline 码流的解码,此时熵编码只使用基于上下文自适应变长编码(Context-based Adaptive Variable Length Coding,CAVLC)[9],整个解码流程也相对简单。FFmpeg 解码一帧的具体流程如图 1 所示。其中,子流程decode_mb_cavlc 主要执行 CAVLC 熵解码(包含反量化),获取当前解码宏块类型、当前解码宏块帧内预测模式或者帧间运动矢量 ***mv***、参考帧、量化参数 *qp* 和滤波强度 *BS* 等信息。子流程 hl_decode_mb 主要执行帧内预测、帧间运动补偿、反变换与重建、去块滤波等操作。由解码流程可以看出,FFmpeg 在解码一帧中的不同宏块时,采用的是先执行一个宏块的所有解码操作,重建出此宏块后再解码下一个宏块的方法。同时,在解码一个宏块之前,先执行 decode_mb_cavlc 子流程获得解码当前宏块所有需要的信息,再在 hl_decode_mb 子流程中进行预测、反变换与去块滤波这 3 个操作,而后者的解码计算量最大。

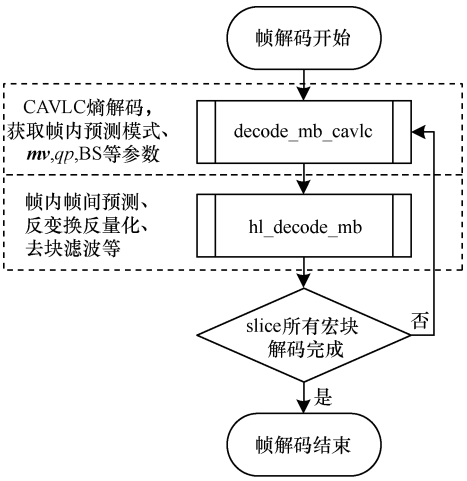

图 1 FFmpeg 中 H.264 解码器 slice 解码流程

2.2 帧内并行解码方案设计

根据上述分析,本文提出一个宏块级别的并行解码方案,具体来说,并行解码器首先执行 decode_mb_cavlc 子流程,对一帧中所有宏块进行处理,获取所有解码所需的信息,例如量化参数 *qp*、运动矢量 ***mv*** 等,保存到中间变量;然后将熵解码之后的残差数据和得到的所有宏块解码信息传送到 GPU 显存,对帧内预测和帧间预测、反量化和反变换、去块滤波模块分别设计并行解码方案,对一帧中所有宏块进行并行处理;最后得到重建好的解码图像传回 CPU,一帧图像的解码过程至此完成。

3 并行解码器实现

3.1 基于 CUDA 的并行解码器流程

如上节分析,本文设计基于 CUDA 的宏块级并行解码器,具体流程如图 2 所示。

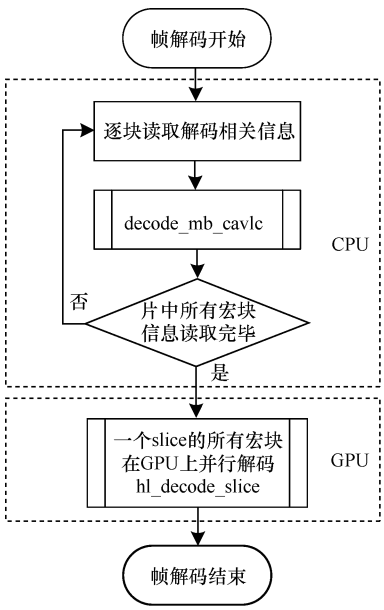

图 2 基于 CUDA 的并行解码器 slice 解码流程

对比图 1 可知,2 种解码方案最大的不同即是:在 FFmpeg 解码过程中,每个宏块从最初的读取码流进行熵解码到最后的图像重建整个过程全部完成,再循环进行下一个宏块的解码,而并行化方案则是先将一帧的所有宏块解码需要的信息从码流中全部读出,再在 GPU 上进行宏块间的并行处理。读取码流信息逻辑判断多,计算量相对较小,适合在 CPU 上循环串行处理。读取到所有码流信息后,就可以同时对一帧中的所有宏块进行解码,同时解码后续流程反量化和反变换、预测、环路滤波恰好是计算量大的部分,可以利用 GPU 的高计算性能进行处理。并行化方案充分利用 CPU 和 GPU 各自的特点,将整个解码流程划分为相互独立的 2 个步骤分别进行处理。

3.2　并行解码器各模块分析

3.2.1　帧内预测与帧间预测

预测是找出当前块的相似块的过程。当前块和与当前块差别很小的预测块相减后,其编码所需码流就可大幅减小,从而节省了码率,更利于传输。寻找预测块的过程则是在某种评判标准下在同一帧或者其他帧找到与当前块最相似的宏块,这里面涉及复杂的运算。H.264 帧内预测宏块有 2 种大小:4×4和 16×16。4×4 宏块帧内预测分为 9 种模式,16×16 的则分为 4 种。因为编码当前块进行帧内预测时需要参考左边宏块和上边宏块,所以一帧中的所有宏块不能同时并行预测,因此,设计并行方案为 WPP 方式[10]。为了存储帧内预测模式,本文将 16×16 宏块当作 16 个 4×4 子宏块对待,设计一个二维数组 *predmode*[2][17],前面表示一个 16×16 宏块的变换方式是 16×16 或者 4×4,后面表示 16 个 4×4 子宏块实际的预测模式(*predmode*[0][0]用来标识宏块大小)。

帧间预测模块的运算量集中在像素插值、运动估计和重建 3 个过程[3]。其中像素插值是为了获取插值图像;而运动估计是根据解析码流得到 ***mv***,并结合参考帧求出当前块预测值的过程;把运动估计预测块与残差相加得到解码图像即重建[11]。H.264 的像素插值分为半像素插值、1/4 像素插值和 1/8 像素插值 3 种,前两者针对整帧一次性并行计算完,后者在需要时选择性地完成。***mv*** 则是根据码流中解析出来的 *MVD* 和根据周围宏块预测得到的 *MVP* 计算得到,以 4×4 为基本块大小。设计二维数组 ***mv***[16][2]保存每个 16×16 宏块水平和垂直方向的 2 个分量。

3.2.2　变换与量化及反变换与反量化

变换去除了图像的相关性,将图像信息集中到一个小的区域;量化则减小了图像编码的动态范围,对图像信号进行了压缩[12]。H.264 采用整数离散余弦变换,避免了变换和逆变换之间的失配问题[13]。变换和量化原理与算法都较为简单,需要注意的是一帧图像中分为不同大小的变换块,由于宏块大小和扫描顺序的原因,其位置并不确定,因此需要在前期根据大小记录下宏块位置。具体的变换算法采用蝶形算法[14]。反量化则更简单,4×4 宏块和 16×16 宏块的量化表尽管不一样,但是 *qp* 范围一定,量化表总数量一定,因此,先将所有量化表存放于 GPU 的常数存储器中,并行计算时查表即可。

3.2.3　去块滤波

H.264 标准采用环路滤波去除反变换量化后图像出现的方块效应,它自适应地对宏块边界和样点量化值进行条件判断和处理,从而提高图像的主客观质量。实际判断时采用滤波强度来选择去块滤波的滤波参数,控制去除方块效应的程度[11]。对亮度帧,一帧图像中每个 16×16 宏块的 16 个 4×4 宏块需要对上边和左边的边进行滤波处理,分为水平滤波和垂直滤波 2 种,共设置 32 个滤波强度值。在并行方案中,一条水平(垂直)滤波边界一次处理。因为相邻 2 条边的滤波操作的数据有重叠,所以将所有边分为奇偶分为 2 次并行处理,即共进行 4 次并行处理[15]。对色度帧 *U*,*V* 分量的并行滤波方案与亮度帧类似。

4　实验结果与分析

为了对本文设计并实现的并行化 H.264 解码器性能进行评价,本文对各个并行化模块以及整个并行解码器在台式机上进行了实验测试。所有的开发以及测试都是基于 Ubuntu 12.04 操作系统,开发语言为 C 语言(CPU 部分)以及 CUDA-C 语言(GPU 部分)。实验环境配置包括 CPU 和 GPU 两方面,具体如下:

(1)CPU:Intel Core-TM i7-4770 CPU@3.40 GHz,8 GB 内存;

(2)GPU:GeForce GTX 680,2 GB 显存,48 KB shared memory,CUDA Toolkit 5.5。

x264 编码器使用版本为 x264-snapshot-20130224-2245-stable,指定 profile 为 baseline,FFmpeg 版本为 ffmpeg-0.5.13。视频测试序列以标准 YUV 测试序列用 x264 编码为 264 文件,具体如下:

(1)akyio_qcif.264,分辨率 176×144,300 帧;

(2)akyio_cif.264,分辨率 352×288,300 帧;

(3)ducks_take_off_720p.264,分辨率 1 280×720,500 帧;

(4)ducks_take_off_1080p.264,分辨率 1 920×1 080,500 帧。

4.1 性能测试结果

使用 ducks_take_off_720p 测试视频序列,实验分别测试了各个并行加速模块的性能,并与 FFmpeg 的 H.264 解码进行对比,具体结果如表 1 所示。

表 1 各并行化模块的性能测试结果对比

解码模块	CPU 帧率/(f·s^{-1})	GPU 帧率/(f·s^{-1})	加速比
帧内预测	15.67	81.46	5.20
帧间预测	12.43	103.63	8.34
变换与量化	11.68	208.44	17.85
去块滤波	10.42	124.37	11.93

如上文所述,H.264 视频编解码计算量最大的部分即预测模块。因为并行化解码方案以宏块为基本单位,而预测过程对宏块彼此间的相互依赖较为严重,而这对帧内预测尤甚,所以 GPU 并行方案相对 CPU 串行方案加速比只有 5 倍~9 倍,没有达到数量级上的性能提升。作为模块之间的对比,变换量化以及去块滤波波 2 个模块的加速比就有显著的数量级上的提升,这是因为它们都是基于单独的宏块,并且宏块之间无任何关联,所以特别适合并行处理。这也与这 2 个模块的原理所展现出来的特点相符合。

本文同时也使用不同分辨率的测试视频序列对解码器整体性能做了实验测试,并与 FFmpeg 的 H.264解码进行了对比,结果如表 2 所示。

表 2 不同分辨率下的解码性能测试结果对比

测试序列	FFmpeg 帧率/(f·s^{-1})	GPU 帧率/(f·s^{-1})	加速比
akyio_qcif	2 127.66	4 150.64	2.12
akyio _cif	649.35	1 707.79	2.63
ducks_take_off_720p	17.95	113.98	6.35
ducks_take_off_1080p	8.24	61.72	7.49

从表 2 可以看出,对于整个解码器的性能比较,GPU 并行解码器比 CPU 串行解码器有明显的提升,但是在视频分辨率较低的情况下(如 176×144(QCIF)和 352×288(CIF)等),GPU 并行解码器相对 CPU 串行解码器性能提升只有较小的 2 倍~3 倍,并没有表现出巨大的优势,因为能够进行并行运算的数据和只能顺序执行的逻辑判断的处理量相差不大。而在视频分辨率提高(达到 1 280×720(720p)、1 920×1 080(1 080p))时,GPU 并行解码器的优势就显露出来了,此时它比串行解码器的速度可以加快 7 倍左右。这是因为视频分辨率越高,一帧中能够并行化处理的数据量就越大,能并行处理的数据量也就越多,在整个解码过程中所占的比例也就越大,更能体现出并行方案的优越性,这与 GPU 的运算特点也是相符的。

4.2 结果分析

从上述各模块的性能测试对比和解码整体性能测试对比可以看出,对于计算量大、适合做并行处理的解码模块,设计一个好的并行化方案并实现,会取得非常好的加速性能,从而进一步促进解码整体性能的提升。并行处理的数据量越大,GPU 的并行计算能力越能够得到充分利用,因此,视频分辨率越大,加速效果就越好[10]。同时,并行处理的数据之间相关性越小则越适合做并行处理。而由表 1 和表 2 数据对比分析可以看出,并行解码器的整体性能提升比分模块测试得到的结果略差,原因在于由各模块结合而成的解码器整体存在数据与流程方面的一些相互依赖和制约。但是总体上,并行解码器比 CPU 串行解码器的性能已有显著的提升。

5 结束语

本文参考 FFmpeg 设计并实现了基于 CUDA 平台的并行 H.264 解码器。依据视频解码运算量大、适合并行计算的特点,针对解码过程的每一个步骤,分模块设计并行化方案,并且基于 CUDA 实现。从实验结果可以看出,利用 GPU 加速的解码器性能能够得到明显提升,并且对于单独的某个模块来说,性能提升更多。

下一步将针对尚未并行化的模块做进一步分析,讨论其并行化的可行性,并结合 CPU 的特点进行优化加速。同时,基于此并行解码器进行功能扩展,使之支持 main 和 high profile,从而进一步提升解码器的性能。

参考文献

[1] 管 辉. 基于同构多核处理器的 H.264 并行解码算法研究[D]. 哈尔滨:哈尔滨工业大学,2009.

[2] Jie C, Liu K J, Koc U V. Design of Digital Video Coding Systems: A Complete Compressed Domain Approach[M]. [S.l.]:CRC Press,2001.

[3] Wiegand T, Sullivan G J, Bjontegaard G, et al. Overview of the H.264/AVC Video Coding Standard[J]. IEEE Transactions on Circuits and Systems for Video Technology,2003,13(7):560-576.

[4] Tamhankar A, Rao K R. An Overview of H.264/MPEG-4 Part 10[C]//Proceedings of the 4th EURASIP Conference on Video/Image Processing and Multimedia Communications. Washington D.C., USA: IEEE Press, 2003:1-51.

[5] Ephraim Y, Malah D. Speech Enhancement Using a Minimum-mean Square Error Short-time Spectral Amplitude Estimator[J]. IEEE Transactions on Acoustics, Speech and Signal Processing,1984,32(6):1109-1121.

(下转第 257 页)

5　结束语

本文采用 MSRCR 增强操作对低光照视频进行了质量增强,而在其后的编码中利用图像纹理变化的相关性,提出一种快速的模式选择判断算法。通过对原始模式的分析利用,减少编码过程中模式判断的计算复杂度。对比实验结果显示该算法能够有效提高转码速度,而且 PSNR 质量几乎不变。下一步将对算法中运动矢量的变化进行研究,通过对运动信息的重用,加快其运动搜索和运动估计速度。

参考文献

[1] 余圣发,陈曾平,庄钊文. 针对网络视频应用的视频转码技术综述[J]. 通信学报,2007,28(1):111-118.

[2] Wang Zhihang,Gao Wen,Zhao Debin. et al. A Fast Intra Mode Decision Algorithm for AVS to H. 264 Transcoding[C]//Proceedings of 2006 IEEE International Conference on Multimedia and Expo. Washington D. C., USA:Press,2006:61-64.

[3] Zhou Zhi, Sun Shijun, Shawmin L, et al. Motion Information and Coding Mode Reuse for MPEG-2 to H. 264 Transcoding [C]//Proceedings of 2005 IEEE International Symposium on Circuits and Systems. Washington D. C. ,USA:2005:1230-1233.

[4] Lu Xiaoan,Tourapis A M,Peng Y. Fast Mode Decision and Motion Estimation for H. 264 with a Focus on MPEG-2/H. 264 Transcoding[C]//Proceedings of 2005 IEEE International Symposium on Circuits and Systems. Washington D. C.,USA:IEEE Press,2005:1246-1249.

[5] Wang Zhihang,Ji Xiangyang,Gao Wen,et al. Effective Algorithms for Fast Transcoding of AVS to H. 264/AVC in the Spatial Domain [J]. Multimedia Tools and Applications,2007,35(2):175-202.

[6] Moiron S,Faria S,Navarro A,et al. Video Transcoding from H. 264/AVC to MPEG-2 with Reduced Computational Complexity [J]. Signal Processing: Image Communication,2009,24(8):637-650.

[7] Chia-Tien L,Lin Yinyi. Motion Re-estimation for H. 264/AVC Video Downscaling Transcoding Using EPZS Algorithm [C]//Proceedings of the 18th IEEE International Conference on Image Processing. Washington D. C., USA:IEEE Press,2011:2109-2112.

[8] Wang Jiao,Yang Enhui,Yu Xiang. An Efficient Motion Estimation Method for H. 264-Based Video Transcoding with Spatial Resolution Conversion[C]//Proceedings of 2007 IEEE International Conference on Multimedia and Expo. Washington D. C., USA: IEEE Press, 2007: 444-447.

[9] Sangkwon N, Chong-Min K. A Multi-layer Motion Estimation Scheme for Spatial Scalability in H. 264/AVC Scalable Extension [C]//Proceedings of 2009 IEEE International Conference on Multimedia and Expo. Washington D. C.,USA:IEEE Press,2009:69-72.

[10] von dem Knesebeck M, Nasiopoulos P. A Fast Mode Decision Algorithm for Downscaled Transcoding of H. 264 Preencoded Video [C]//Proceedings of IEEE International Conference on Consumer Electronics. Washington D. C. ,USA:IEEE Press,2010:87-88.

[11] Ahmad I,Wei Xiaohui,Sun Yu,et al. Video Transcoding: An Overview of Various Techniques and Research Issues[J]. IEEE Transactions on Multimedia,2005,7(5): 793-804.

[12] 褚晶辉,俞斯乐,鲁照华. 视频转换编码及其实现技术的研究[J]. 电子学报,2004,32(10):1678-1683.

[13] Bertalmío M, Caselles V, Provenzi E. Issues About Retinex Theory and Contrast Enhancement [J]. International Journal of Computer Vision, 2009, 83 (1): 101-119.

[14] Saponara S,Fanucci L,Marsi S,et al. Algorithmic and Architectural Design for Real-time and Power-efficient Retinex Image/video Processing [J]. Journal of Real-time Image Processing,2007,1(4):267-283.

[15] Lee J Y,Park H. A Fast Mode Decision Method Based on Motion Cost and Intra Prediction Cost for H. 264/AVC[J]. IEEE Transactions on Circuits and Systems for Video Technology,2012,22(3):393-402.

编辑　金胡考

(上接第 252 页)

[6] 张　舒,褚艳丽,赵开勇,等. GPU 高性能运算之 CUDA[M]. 北京:中国水利水电出版社,2009.

[7] 郭　信,陈耀武. 基于功能模块的 H. 264 并行解码算法[J]. 计算机工程,2010,36(23):231-233.

[8] Draft I. Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H. 264|ISO/IEC14496-10 AVC)[Z]. 2003.

[9] Puri A, Chen X, Luthra A. Video Coding Using the H. 264/MPEG-4 AVC Compression Standard[J]. Signal Processing: Image Communication, 2004, 19 (9): 793-849.

[10] Sullivan G J,Ohm J,Han W J,et al. Overview of the High Efficiency Video Coding (HEVC) Standard[J]. IEEE Transactions on Circuits and Systems for Video Technology,2012,22(12):1649-1668.

[11] 陆　达. 基于 CUDA 的 H. 264 视频并行编解码器研究与实现[D]. 湘潭:湘潭大学,2012.

[12] 毕厚杰. 新一代视频压缩编码标准[M]. 北京:人民邮电出版社,2005.

[13] Richardson I E. The H. 264 Advanced Video Compression Standard[M]. [S. l.]:John Wiley & Sons,2011.

[14] 孙少林,汤　伟,任小青,等. H. 264 中的整数 DCT 及其蝶形算法[J]. 信息技术,2013,37(10):165-166.

[15] 刘　虎,孙召敏,陈启美. CUDA 架构下 H. 264 快速去块滤波算法[J]. 计算机应用,2010,30(12):3252-3254.

编辑　金胡考