

基于多核处理器的可视媒体加速 算法研究

Video Media Processing Based Multi-core Processor Study

(申请清华大学工学硕士学位论文)

院(系、所): 计算机科学与技术系
学 科 : 计算机科学与技术
研 究 生 : 庞 一
指 导 教 师 : 杨 士 强 教 授

二零零七年九月

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名： _____

导师签名： _____

日 期： _____

日 期： _____

摘 要

随着人们对于多媒体真实感和交互性要求的不断提升，多媒体技术得到了长足的发展，算法复杂度尤其是视频处理算法的复杂度也随之提高，使得处理器越来越难以支持媒体运算的需求。例如普通的通用处理器无法很好的支持新一代的视频编解码算法（如 H.264、多视点视频编解码算法）和图形处理（如光线跟踪）。另一方面，为使单核处理器性能的提高所需要付出的代价也越发的增加，处理器设计者的思路转向多核处理器。针对可视化媒体处理算法的特点，多核处理器是加速媒体处理的方法之一。

本文基于以上背景，研究基于多核处理器的可视化媒体的加速处理问题，主要贡献有三点：

1. 采用不对称体系结构的多核处理器加速 H.264 解码算法的系统设计与实现

以清华大学 CPU 中心的 thump 处理器为主处理器，自行研制 H.264 媒体加速协处理器。使这种主协结构的不对称多核处理器达到很好的视频编解码算法加速效果。该系统在 thump 的处理器设计模拟器上实现，模拟实验效果达到实时要求。

2. 采用混合多处理器体系结构并行处理的方法，设计加速处理多视点视频编解码算法。

多视点视频编解码算法是新一代视频编解码算法，复杂度远高于 H.264，是 3DTV 等 3D 视频应用的基础技术。本文介绍了以 IBM、Sony、东芝合作研发的 Cell 处理器为多核处理器平台，设计的多视点视频编解码的并行算法，加速比可以达到 8。

3. 采用混合多处理器体系结构并行加速图形算法的核心算法——光线跟踪

以 Cell 处理器为多核处理器平台，分析光线跟踪算法，结合二者的特点，设计并行处理算法，使得可以在较好的光线跟踪图形合成的实验效果的情况下做到处理速度的提升。

关键词：多核处理器，多视点视频编解码，H.264/AVC，光线跟踪，并行处理

Abstract

With the development of internet and computing technology, the next generation of multimedia applications will be interactive and realistic. The complexity of video processing algorithm made processor more and more difficult fit performance requirement of media processing. For example general purpose processor can't process the new generation of video coding algorithm(such as H.264, Multiview Video Coding) and Graphic algorithm(such as Ray Tracing). The other side, Performance goes with square root of single core processor design complexity, and multicore processor is the trend. Multicore processor is a method of accelerating media processing.

Under the background, the thesis research acceleration of visual media processing based multicore processor, the main contribution is:

1. Design and implementation of accelerating H.264 decoding on asymmetric multicore processor

We choose Tsinghua University CPU center's thump as main core, and develop H.264 accelerating core. The system runs on the simulator of thump real time.

2. Design parallel scheme of multiview video coding based hybrid architecture.

Multiview video coding is a new generation of video coding algorithm, and its complexity is much higher than H.264, and the basic technology of 3DTV application. The thesis introduces a parallel method of multiview video coding whose speedup is nearly 8, using Cell processor which is codesigned by IBM, Sony and Toshiba as platform.

3. Porting Ray Tracing algorithm which is classic algorithm in Graphics onto hybrid architecture.

Analyse Ray Tracing algorithm, and design parallel algorithm using Cell processor as platform. We get a big advance without losing much realitice impression.

Key words: Multi-core processor, Multi-view video coding, H.264/AVC, Ray Tracing, Parallel processing

目 录

摘 要	I
Abstract.....	II
目 录	I
图的索引	I
表格索引	I
第 1 章 引言	1
1.1 研究背景	1
1.2 视频编解码算法及其加速系统的发展.....	2
1.2.1 视频编解码标准的发展	3
1.2.2 基本的编码框架中主要的编码技术.....	8
1.2.3 针对于不同视频编解码算法进行的加速系统研究.....	12
1.3 在多核处理器上的可视化媒体加速算法的发展现状.....	13
1.3.1 不对称多核处理器	14
1.3.2 对称多核处理器	15
1.3.3 异构多核处理器	17
1.4 我的论文	18
第 2 章 媒体处理并行算法设计方法.....	19
2.1 预备知识	19
2.2 分解技术	19
2.3 任务的性质	20
2.4 负载平衡的映射技术	20
2.5 减少交互代价	20
2.6 并行算法模型	20
2.7 可视媒体处理算法特点	21
2.8 视频处理算法并行化方法	23
第 3 章 基于主从不对称体系结构的 H. 264 解码加速系统设计及实现.....	25
3.1 H. 264/AVC 算法介绍	25
3.2 总体设计流程	28
3.3 总体设计方案	29
3.4 总体设计目标	31

3.5 H.264 中去块滤波部分的算法流程详解.....	31
3.6 H.264 中去块滤波部分的算法实现.....	37
3.6.1 去块滤波的功能	37
3.6.2 去块滤波部分的设计	37
3.6.3 去块滤波部分的理论分析.....	46
3.6.4 各种综合方案的测试与结果分析.....	51
第4章 基于 Cell 的多视点视频编解码并行算法设计.....	53
4.1 多视点视频编解码算法介绍与分析.....	53
4.2 Cell 处理器	56
4.3 基于多核处理器的并行化多视点视频编解码算法.....	58
4.4 结论	61
第5章 基于 Cell 处理器的光线跟踪设计与实现.....	62
5.1 光线跟踪算法分析	62
5.1.1 光线跟踪算法的特点.....	62
5.2 在 Cell 上的光线跟踪算法设计.....	63
5.2.1 在 Cell 上实现光线跟踪的特殊点.....	63
5.2.2 编程模型	64
5.2.3 空间检索结构	66
5.3 实现	66
5.4 可扩展性	68
5.5 与通用处理器比较	69
5.6 双缓冲区设计	69
5.7 结论与展望	71
第6章 结论	72
参考文献	73
致 谢	78
个人简历、在学期间发表的学术论文与研究成果.....	79

图的索引

图 1 视频编解码标准的发展	3
图 2 WMV9 的用途	7
图 3 WMV9 的工作流程	8
图 4 Max 量化器	11
图 5 AMD 的 Alchemy 体系结构图	15
图 6 MPEG-4 解码系统结构图	15
图 7 Intel 双核处理器的制版图	16
图 8 H.264 编码的线程级并行流程图	17
图 9 IBM Cell 处理器的体系结构图	17
图 10 MPEG-4 AVC/H.264 的编码结构	26
图 11 7 种不同运动补偿块	28
图 12 intra 编码的 8 种不同空域预测模式	28
图 13 设计流程图	29
图 14 体系结构整体图	30
图 15 媒体加速引擎	30
图 16 块边缘强度计算树	32
图 17 去块滤波整体流程图	33
图 18 亮度滤波流程图	34
图 19 8.7.1 核心滤波算法	35
图 20 对每个像素点的滤波处理	35
图 21 8.7.2 得到 $q'_{i,p'_{i}}$	36
图 22 去块滤波与其他设备间的连接关系	40
图 23 亮度色度滤波串行结构图	41
图 24 亮度色度滤波以宏块为单位流水连接结构图	42
图 25 亮度色度滤波以边为单位流水连接结构图	43
图 26 处理一个宏块数据流程图	43
图 27 滤宏块中一条边的结构图	44
图 28 以滤边为单位流水处理一个宏块的结构图	45
图 29 不同种类的宏块位置	46
图 30 各方案理论分析结果	50
图 31 各方案测试结果	51
图 32 多视点视频照相机的摆放方案	53
图 33 八视点视频编码当中一个 GOP 的依赖关系图	54
图 34 多核处理器的目标模型	56
图 35 优化任务调度图	60

图的索引

图 36 光线跟踪原理图	63
图 37 基础算法效果图	68
图 38 多核加速效果图	69
图 39 双缓冲区效果图	70
图 40 双缓冲区多核加速比	71

表格索引

表格 1 视频编解码算法并行加速的主要算法.....	12
表格 2 媒体算法的并行单元分类.....	22
表格 3 视频编解码算法加速方法分类表.....	23
表格 4 FIFO 的处理单元	38
表格 5 MacroBlock_par 参数表.....	38
表格 6 去块滤波可配置参数	46
表格 7 测试方案参数列表	48
表格 8 理论分析结果列	49
表格 9 测试结果列	51
表格 10 最佳方案参数表	52
表格 11 参数表.....	55
表格 12 并行可扩展性	68
表格 13 Cell 与通用处理器性能比较.....	69
表格 14 双缓冲区运行结果	70

第 1 章 引言

1.1 研究背景

近年来，嵌入式应用中媒体处理的数据量和精度的要求飞速提高，性能需求大大增加。随着人们对于多媒体真实感和交互性要求的不断提升，多媒体技术得到了长足的发展，算法复杂度尤其是视频处理算法的复杂度也随之提高，使得处理器越来越难以支持媒体运算的需求。针对于多媒体应用的音视频编解码，主要是基于各种各样的编解码标准来进行，这些标准有 H.261, H.262, H.263, H.264, MPEG-1/2/4, AVS, WMV9 等。随着新的编解码标准的不断推出，人们对于处理器性能的要求也越来越高。例如 H.264 的解码端复杂度比 MPEG-4 增加了 2 到 3 倍，编码端复杂度比 MPEG-4 增加了 4 到 5 倍[1]。市场对低成本、低功耗、高性能针对多媒体内容的处理器需求增加了。

编解码标准的日新月异，使得媒体处理器体系结构的灵活性和适应性显得越发的关键。硬件的运算速度虽然快，但是一旦成型就无法修改，很难适应视频编解码的更新换代；软件的灵活性高，但是基于普通通用处理器和纯软件解码方式的运行速度很难达到日益增加的媒体处理的性能要求。综合软硬件的这些特点，人们提出了软硬件协同设计的方法。随着微电子技术的不断发展，嵌入式芯片进入 SoC (system on chip) 时代，使得把复杂的系统，甚至多个处理器集成在一个芯片上成为可能。近年来，各大厂商科研机构都在积极研制自己针对媒体应用的加速技术，例如 Intel 的 SSE 系列技术；AMD 的 Alchemy；TI 的 TMS320DM642；Philips 的 TriMedia；Equator 的 MAPCA。这个问题随着媒体处理算法的不断复杂化引起了人们越来越多地关注。

这些年，传统结构的 CPU 性能增长的速度放缓了许多，性能提升速度降到了每年 1.2 倍左右。原因有以下几点：一，早期的 CPU 依靠增加更多的运算单元来提升性能，但是随着运算单元的增加，串行的指令流越来越难做到更大程度的指令级并行，使得额外增加的计算单元不能得到充分的利用；二，内存的速度虽然一直在上升，但是没有时钟频率提升得快，而这差距越发的大，致使内存的速度成为瓶颈，随之而来的解决方法就是利用越来越多的空间安放分支预测、乱序执行等电路；三，随着集成度的提高，功耗和散热问题也变得越来越

越重要。以上几点就制约了传统的单纯依靠提高集成度提高时钟频率的方法提高处理器性能的老路。

基于以上情况,处理器的设计者们开始寻找其他的道路以取代简单的依靠提高时钟频率而提高计算能力的办法。设计者们通过增加 SIMD 单元来提高指令级并行性,提出多核处理器体系结构等。事实证明这些增加并行性的方法都十分有效,例如 Nvidia 的 GPU 7800 GTX 可以达到 313Gflops,比 2.2GHz 的 AMD Opteron 处理器快 35 倍[2]。目前 GPU 和 CPU 有逐渐融合的趋势。在已经通过增加流媒体运算模块、SIMD 以及多核技术提高性能之后, GPU 现在的可编程性能越来越高;而 CPU 也加入了越来越多的核以及流处理单元。目前的通用处理器已经可以提供 2~8 个核[3-6]。Intel 公司 2004 年 5 月做出的策略性决定,将调整研发方针,对于核心产品 X86 系列微处理器,终止所有单核产品的开发,转向集中开发多核产品。几乎与此同时,AMD 公司也表明,已停止集成单 CPU 核的 X86 系列微处理器的设计,我们完全有理由相信,多核微处理器是未来处理器的发展方向。在可视化媒体加速的道路上,我们也积极寻找与多核处理器趋势相契合的点。

1.2 视频编解码算法及其加速系统的发展

数字视频压缩编码技术的研究,已历经半个世纪,在理论和工程上都取得了大量的成果。视频信号数字化数据量非常大,但是在统计特性上以及在人眼主观感觉上都存在相当多的冗余信息,因此视频压缩算法的研究主要是基于统计方法和人眼视觉特性。20 世纪 90 年代,在 ISO/IEC 的 MPEG、JPEG 和 ITU-T 的组织下,对已有视频和图像编码技术进行了测试,并综合了这些先进而成熟的编码技术,制定了几个通用的多媒体压缩编码标准,包括适用于二值图像的 JBIG、用于连续灰度和彩色静止图像的 JPEG、用于 Nx64K 视频传输的 H.261、面向 1.5Mbps 数字视频和音频传输与存储的 MPEG-1、面向高品质数字音视频传输和存储的 MPEG-2 和适于低码率视频编码的 H.263 等。这些标准的制定促进了多媒体的广泛应用,尤其是 MPEG-1 和 MPEG-2 标准对多媒体产业的形成具有里程碑的作用,这些标准主要采用以下四类技术:运动补偿、正交变换、量化和熵编码。

1.2.1 视频编解码标准的发展

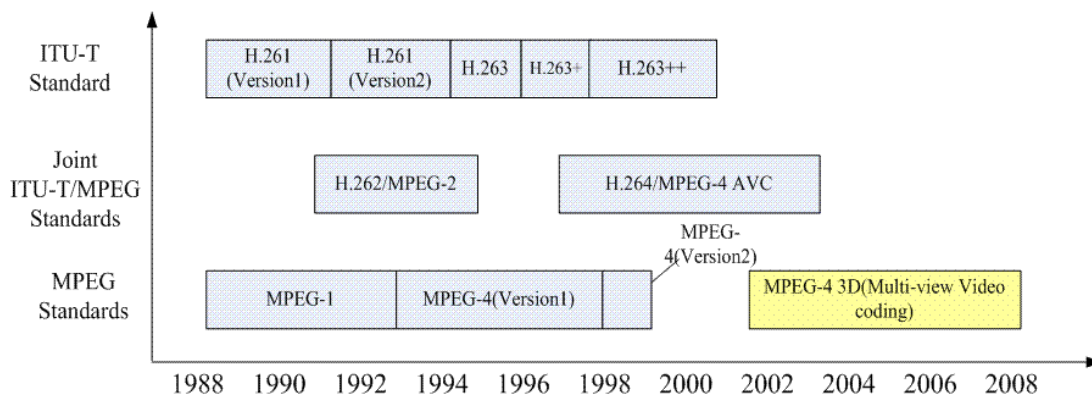


图 1 视频编解码标准的发展

国际标准化组织 ISO MPEG (ISO Moving Picture Experts Group) 和 ITU-T VCEG (ITU-T Video Coding Expert Group) 在视频编码标准的发展过程中发挥了很重要的作用(如图 1 所示)。ISO MPEG 制定的 MPEG 系列(例如 MPEG-1, MPEG-2, MPEG-4 等)标准则主要针对视频的存储媒体(VCD、DVD)和电视广播应用的。ITU-T 制定的 H.26x 系列建议标准(如 H.261, H.262, H.263, H.26L 以及 H.264 等)主要针对实时视频通信应用。在 2001 年 12 月, ISO MPEG 组织 ITU-T VCEG 组织联合成立了一个联合视频组织(Joint Video Team), 双方在 H.26L 基础上形成一个视频编码标准, 这就是后来的 ISO MPEG-4 part 10 AVC (Advanced Video Coding) 和 ITU-T 的 H.264, 这一标准已经在今年年初生成。

● H.261 标准

1984 年 CCITT 的 XV 研究小组成立了一个关于可视电话的特别小组, 目标是建立一个可用于传输率在 $n \times 64kb/s$ ($n=1, 2, \dots, 5$) 的视频编码标准。1990 年 12 月获得通过成为 H.261 标准。H.261 标准总结了 1948 年以来电视图像编码 40 年研究的成果, 这标志着混合编码算法的成熟和实用化。H.261 标准具有帧内编码帧(I 帧)和帧间编码帧(B 帧), I 帧和 B 帧采用不同的编码方法。DCT 变换, 行程编码和 Huffman 变长编码被采纳, H.261 标准也采用了码率控制策略, 保证了输出码率为恒定值。

● MPEG-I 与 MPEG-2 标准

MPEG-1(ISO/IEC 11172)制定于 1993 年,是针对 1.5Mbps 以下数据传输率的数字存储媒质运动图像及其伴音编码的国际标准。MPEG-1 用于在 CD-ROM 上存储同步和彩色运动视频信号。可优化为中等分辨率,并在其优化模式下,采用所谓的标准交换格式(SIF)。MPEG-1 对色差分量采用 4:1:1 的二次采样率。MPEG-1 旨在达到 VRC 质量,其视频压缩率为 26:1。[7]

1995 年又出台了 MPEG-2(ISO/IEC 13818) [8],它追求的是 CCIR601 建议的图像质量 DVB、HDTV 和 DVD 等制定的 3Mbps~10Mbps 的运动图像及其伴音的编码标准。该标准最初的目的是在与 MPEG-1 兼容的基础上,实现低码率和多声道扩展。后为适应演播电视要求,MPEG 于 1994 开始致力于定义一个可以获得更高质量的多声道音频标准。该标准不与 MPEG-1 兼容,定名为 MPEG-2 AAC。AAC 标准完成于 1997 年,经 BBC(U. K.)和 NHK(Japan)使用、测试表明已达到最优化 ITU-R601 推荐的分辨率。AAC(Advanced Audio Coding)对于低比特率的多声道编码能提供相当高的声音质量。由于它不向后兼容,故具有更高的压缩效果。据测试它以 320Kbps 传送的音频信号比 MPEG-2 以 640Kbps 传送的音质还略好些。AAC 标准的发展标志着标准化工作向着模块化方向演变的趋势。这些标准使得基于 CD-ROM 的交互式电视和数字电视成为了可能。

MPEG-1 和 MPEG-2 标准与 H. 261 算法框架基本相同,但是引入了双向预测帧(Bi-directional prediction frame, B 帧),B 帧采用有运动补偿的帧间内插,比 I 帧和 P 帧有更高的编码压缩效率,同时引入了帧内编码量化矩阵。MPEG-2 在 MPEG-1 基础上对混合编码框架又做了一些扩展和改进,针对电视图像隔行扫描的特点,引入两种预测模式:帧预测和场预测。MPEG-2 还支持可选的扫描和可选的帧内变长编码表,可以采用用户自定义的帧内和帧间量化矩阵。MPEG-2 标准还首次引入了档次(Profile)和级别(Level)的概念。

● H.263

1995 年 ITU-T 提出针对低码率视频图像压缩编码标准 H. 263[9]。H. 263 的第一个版本中定义的基本的编码算法与 MPEG-2 类似,同样是使用了 INTER-16x16, INTRA 和 SKIP 等编码模式。但是 H. 263 的基本层(baseline)使用了中值运动向量预测,引入了半像素精度的运动向量,还有针对低码率优化的三维 VLC(run, level, last)。此外, H. 263 的第一个版本包括了 8 个附件

(Annexs A-G), 其中附件 D、E、F 和 G 定义了一些可选的编码工具, 可以进一步提高编码效率, 附件 D 和 F 是比较常用的。在附件 D 中, 运动向量是不受视频帧的边缘限制的, 可以指到帧外。在附件 F 中, 一个 16x16 的宏块可以有四个运动向量, 其中每个 8x8 的子块都可以有自己的运动向量, 这样 INTER-8x8 模式被引入。H. 263 还支持算术编码选项和 BP 帧模式选项, 可以将 B 帧和 P 帧结合在一起编码。这些特点让 H. 263 基本层在低码率的时候编码效率比较高。在 H. 263 的基础上, ITU-T 又提出了两个新的版本 H. 263+[10] 和 H. 263++。H. 263+ 为 H. 263 加入了很多的编码工具, 如环内去块效应滤波器(deblocking filter), 帧内 DCT 变换系数的预测及对应的专门的量化器和 VLC。H. 263++ 加入了另外三个可选的编码工具, 主要为了提高编码的容错 (Error-Resilience) 能力来适应更复杂条件的网络传输。

● MPEG-4

1994 年, MPEG 开始制定一个新的标准, 也就是通常所说的 MPEG-4[11], 不像 MPEG-1 和 MPEG-2, 其重点在于编码的效率上, 而 MPEG-4 的目标是提出一种新的方式来传输、访问和操作数字视听数据 (Audiovisual Data)。MPEG-4 计划为多种通讯规范提供一种普遍的技术解决方案: 电讯, 广播和交互, 最终消除它们之间的差别。MPEG-4 Visual Video Object 部分是在 H. 263、MPEG-1 和 MPEG-2 的基础上发展起来的。与 MPEG-1 和 MPEG-2 不同的是, MPEG-1 和 MPEG-2 是基于帧的规范, 而 MPEG-4 的目标是基于对象的规范。在对图像的描述上, 比传统编码标准增加了对形状、纹理及全景等信息进行描述的算法。同时, MPEG-4 还支持任意形状的运动估计算法, 无限制的运动向量模式 (UMV) 和 4 个 8x8 亮度块预测模式, 运动向量得精度也提高到了 1/4 像素。MPEG-4 沿用了 MPEG-2 得档次和级别的概念, 目的是更广泛的视频应用。MPEG-4 在原来标准的版本 1 和版本 2 之外, 还针对新的应用提出了新的版本 3 (针对视频演播室应用) 和版本 4 (针对网络视频传输应用), JVT 的工作成果也成为 MPEG-4 的第十部分 (MPEG-4 part 10 AVC)。MPEG-4 是全世界成百上千的研究人员和工程师努力的结果, 其正式的 ISO/IEC 命名为 ISO/IEC 14496, 1998 年定稿, 1999 年成为国际标准。

● AVS

AVS (Audio video Coding Standard) 是由数字音视频编解码技术标准工作组 (AVS 工作组) 起草公布的。该工作组是由国家信息产业部科学技术司于 2002

年6月批准成立。工作组的任务是：面向我国的信息产业需求，联合国内企业和科研机构，制（修）订数字音视频的压缩、解压缩、处理和表示等共性技术标准，为数字音视频设备与系统提供高效经济的编解码技术，服务于高分辨率数字广播、高密度激光数字存储媒体、无线宽带多媒体通讯、互联网宽带流媒体等重大信息产业应用。

AVS 标准包括系统、视频、音频、数字版权管理等四个主要技术标准和一致性测试等支撑标准。在2003年12月18-19日举行第7次会议上，工作组完成了 AVS 标准的第一部分（系统）和第二部分（视频）的草案最终稿（FCD），与报批稿配套的验证软件也已完成。2004年12月29日，全国信息技术标准化技术委员会组织评审并通过了 AVS 标准视频草案。2005年1月，AVS 工作组将草案报送信息产业部。3月30日，信产部初审认可，标准草案视频部分进入公示期。

AVS 视频标准从技术上达到国际先进水平，其编码效率是 MPEG-2 的 2-3 倍（根据视频画面尺寸不同有所不同）。AVS 专门针对高清应用进行了优化，方案简洁，实现复杂度明显比 MPEG-4 AVC 低，在高清晰度应用方面处于领先水平。

“第一部分：系统”与目前广泛采用的国际标准 MPEG-2 兼容，并针对数字电视、光盘播放机、网络流媒体、多媒体通信等应用进行了具体的规定和定义，从框架上提供了对国内外各种主流视频、音频编码标准的支持。“第二部分：视频”是 AVS 标准中最复杂的部分。

发布的视频标准草案最终包含了 32 个提案的技术内容，涉及离散余弦变换和量化、隔行预测、熵编码器、对称帧间预测、插值、运动矢量预测、视频编码结构与起始码、滤波器、快速运动估计等技术模块，主要贡献单位包括中国科学院计算技术研究所、上海广电集团、清华大学、浙江大学、华为技术有限公司、华中科技大学、北京工业大学等，是我国科研机构和企业集体创新的成果。

AVS 标准的制定过程吸收了“运动影像专家组（MPEG）”等国际标准化组织制定标准的先进经验，并充分考虑中国数字音视频产业的具体需求。标准制定过程依循三个原则：（1）公开：所有 AVS 会员享有公平的权利和义务，所有的技术都在平等的条件下接受评估；（2）开放：技术方案在工作组范围内完全公开，工作组最终标准草案对全社会公开；（3）竞争：每个技术方案和技术模块的选择都经过公平、充分的比较和评估。[12]

AVS 解码端包括以下几个步骤：宏块解码、块解码、反量化、反变换、帧

内预测、帧间预测、重建、环路滤波。[13]

● WMV9

WMV9 是微软提出来的视频编解码标准，是 Windows Media Video 9 的简称。微软已推出 WMV 一系列视频编解码标准，WMV9 是目前比较流行的视频编解码之一。数据压缩率与 H. 264 大体相同。

WMV9 用于映像内容与移动设备等各种用途。Windows Media 最初的研究重点在针对应用于因特网的个人电脑的音视频流压缩上。后来的目标转变为使数字媒体通过任何网络到任何设备的传输更加有效。

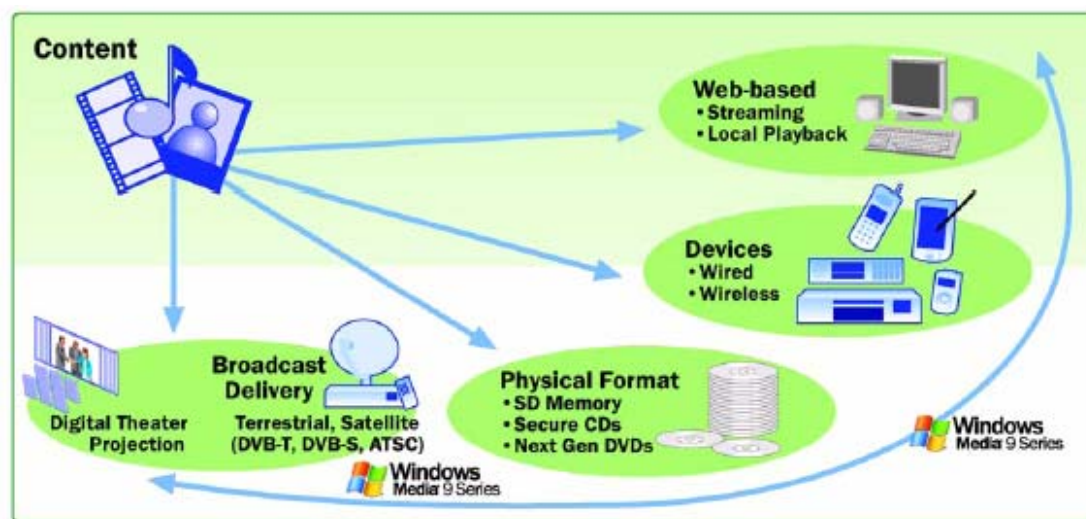


图 2 WMV9 的用途

分为 Authoring, Distribution, Playback 三个阶段。

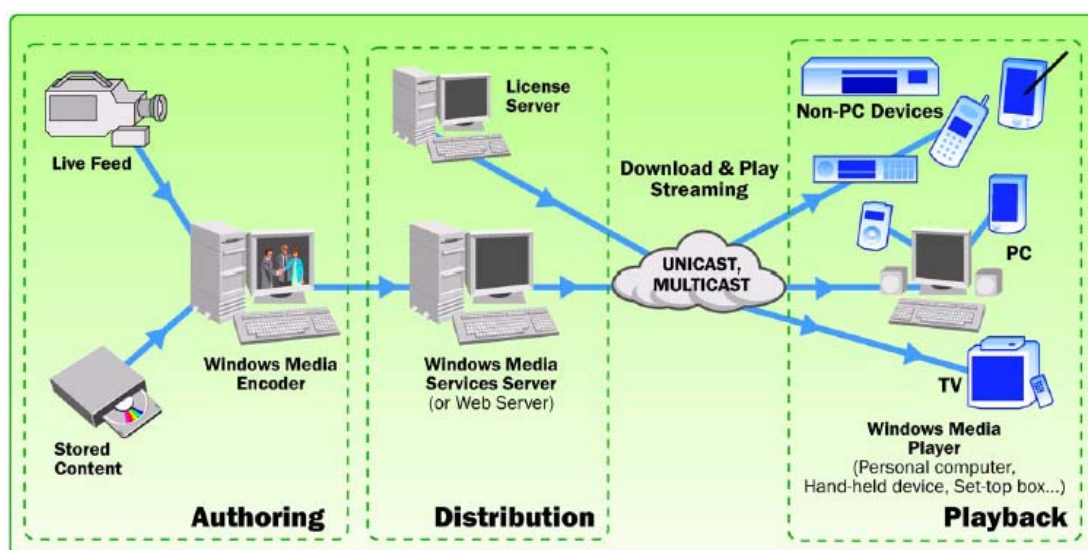


图 3 WMV9 的工作流程

Authoring 是构造数字媒体，对其编码。基于的编码软件是微软的名叫 Windows Media Encoder 9 Series 的软件。它是一种灵活的编码软件，使用者可以通过它实时压缩音视频源文件。

H. 264/AVC 与多视点视频编解码算法会在第五章和第六章进行详细介绍。

1.2.2 基本的编码框架中主要的编码技术

在目前的比较流行的混合编码框架下，主要包含四类编码技术：运动补偿用来去除时域的冗余信息，变换和量化用来去除空域的冗余信息，熵编码用来去除统计上的冗余信息。Cliff Reader 对这四类技术做了很详细的综述。

● 运动补偿

在早期视频研究中，研究人员就认识到视频信号在时域上有很多冗余信息。早在 1929 年，Kell 在一个专利中提出，只需传输帧间变化的部分，可以节省很多传输带宽。上世纪 50 年代初期，随着硬件技术的发展，可以实时的采集编码过程中的统计信息并加以分析。通过一系列统计上的研究，运动补偿这一概念进一步被扩展。Cutlers 在 1952 年首先提出了 DPCM 的概念，之后 Oliver 对图像的线性预测放法进行了详细的理论分析，并首次将其应用于视频编码中，同年 Harris 考虑到利用前一帧视频数据进行线性预测，并且研究了二维预测。

之后的二十年时间里, Bell 实验室的研究人员在空间 DPCM 编码方面做了很多有价值的研究, 比如 Graham 在 1958 年首次用计算机模拟法研究图像的 DPCM 编码。1966 年 O' Neal 根据最小均方差 (MMSE) 准则, 继续用计算机模拟法对图像的 DPCM 编码的线性预测器和量化起做了系统分析。Mounts 等在 1969 年提出了帧间编码的方法。上世纪 80 年代, 运动补偿 DPCM 编码技术首次被 H. 261 标准采纳, 之后的标准一直沿用这一技术。在编码器中, 基于运动补偿 DPCM 编码方法的运动估计部分是计算量最为庞杂的部分, 研究领域内的很多工作都是围绕这方面的软硬件实现进行的。

● 变换

视频信号在空域上是高度相关的, 进行变化编码的目的就是要打破这种相关性, 讲空域的信号变换到频域, 使得信号的能量多集中在少数的低频抽样上。变换编码本身并不实现压缩。Enomoto 和 Shibata 在 1965 年最先把一维 Hadamard 变换应用于视频信号压缩上。Andrew 和 Pratt 在 1968 年首次研究了二维傅立叶变换的方法。由于 Hadamard 变换的运算相对比较简单 (只需要加法和减法), 可以在理论上证明 Hadamard 变换相比傅立叶变换在小尺寸块变换时具有更好的能量聚敛特性, 因而一度成为研究领域的热点。但是 Hadamard 变换基本波形仍旧是方波, 边缘的突变不符合自然视频图像平滑的特性, 会造成主观视觉效果的下降。之后一些变换, 如 Haar 变换和 Slant 变换, 相继被提出, 然而效果也不理想。1974 年, Ahmed、Natarajan 和 Rao 提出的余弦变换取得了突破性的进展。余弦变换可以进行相应的快速软硬件的实现, 并且其性能接近 KL (Karhunen-Loeve) 变换, 这使得余弦变换很快代替上述的一些变化算法成为研究界和工业界的热点。在 H. 261、MPEG-1、MPEG-2、MPEG-4 和 H. 263 等标准中, 基于 8x8 块的 DCT (Discrete Cosine Transform) 变换都是基本的变换算法。由于 DCT 是浮点运算, 编解码之间存在失配 (Mismatch) 问题, 因此在 H. 26L 和 MPEG-4 AVC/H. 264 中采用了类似余弦变换的整系数变换, 这样只需要加法和移位操作就可以实现变化算法, 更易于硬件实现, 而且可以在标准中完整地定义反变换, 避免了编解码之间的失配问题, 还可以将变换与量化步骤结合在一起且将乘法操作集合在量化步骤中。虽然整系数变换的能量聚敛特性比 DCT 变换会差一些, 但是由于 MPEG-4 AVC/H. 264 中采用了许多提高预测精度的工具使残差信号的能量相对比较小, 实际变换的性能稍差带来的视频质量的下

降并不大。

- 量化

经过正交变换、预测处理之后，在熵编码之前，抽样值的能量大多聚敛在低频部分，经过去相关之后，这些抽样值需要进行量化，来减少数据的比特数。量化处理是使数据比特率下降的一个强有力的措施。1960年Max详细分析了这一问题，针对广义的随机变量提出了一个量化器的设计。Max的目标是根据抽样值的概率分布，设计一个统计意义上最优的非线性量化器。它的做法是把概率曲线分割成若干个区域，使得在每两个区域之间抽样值的概率分布是相同的，量化级别就设置在每个区域的中间位置，如图4。尽管Max设计的量化器的性能是接近最优的，但是实际使用非常困难，对于实时的应用场景是不可能的。另一种解决方法是采用均匀量化器，可以通过不断的变换量化系数来控制码率。均匀量化的方法最早由Wintz和Kurtenbach在1968年提出，之后Tasto和Wintz以及Chen在和中分别使用了这种方法。在1983年，Brusewitz在中进一步研究了这种方法，根据各种各样的概率密度函数，在Max的量化器和均匀量化之间寻求一种平衡，经过熵编码之后可以比使用均匀量化器提高1dB到5dB。根据人眼透视原理对变换系数进行加权量化已经被很多研究人员讨论过。1969年Pratt和Andrews提出对于幅值比较大的采样和高频的部分可以进行比较粗略的量化，Tasto和Wintz在1971年对这一现象进行了详细的分析。结论是可以根据人眼透视的原理，给某些量化系数加上统计意义上的权值，得到的结果人眼是无法分辨的。在MPEG-2中使用的就是这种视觉加权量化。H.264中可选51个量化步长，这与H.263中有31个量化步长很相似，但是在H.264中，步长是以12.5%的复合率递进的，而不是一个固定常数。在H.264中，变换系数的读出方式也有两种：之字形(Zigzag)扫描和双扫描。大多数情况下使用简单的之字形扫描；双扫描仅用于使用较小量化级的块内，有助于提高编码效率。

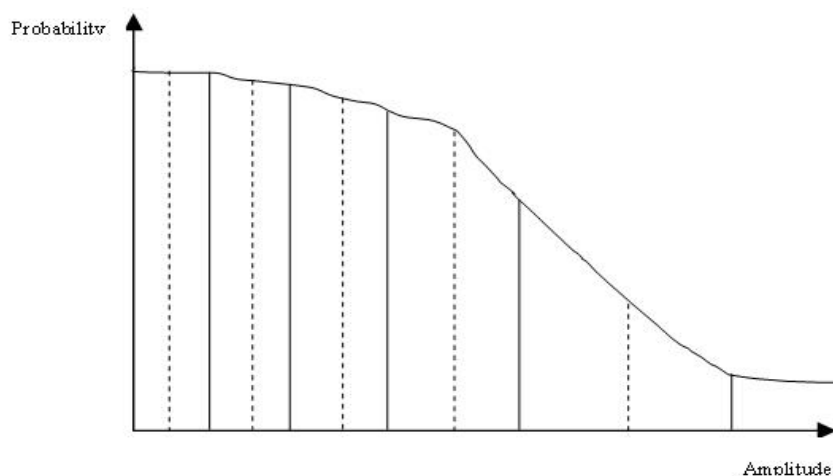


图 4 Max 量化器

● 熵编码

Fano 在 1949 年最早提出熵编码的概念，在 1952 年 Huffman 进一步发展了这一概念。1971 年，Tasto 和 Wintz 第一此把熵编码应用在图像编码之中。其后在 H. 261, MPEG-1 和 MPEG-2 标准中, 二维变长编码 (2D VLC) 编码方法得到广泛应用, 其基本思想是用 (run, level) 表示变换系数中的非零系数 (run 表示非零系数之前 0 的个数, level 表示非零系数的值), 最后一个非零系数用 EOB (End of Block) 标志来表示。但是在低码率情况下, 由于 EOB 的比例过高, 二维变长编码的效率会下降很多, 所以在之后的 H. 263 和 MPEG-4 标准中, 使用三维变长编码 (3D VLC) 来进行变换系数的编码, 其基本思想是在原来二维表示符之上再加一维标志位, 用来表示此非零系数是否为最后一个非零系数, 由此变成三维表示符 (run, level, end), EOB 标志符不再出现。在 MPEG-4 AVC/H. 264 中, 采用了两种熵编码算法, 一中是上下文自适应二进制算术编码 (CABAC-Context-based Adaptive Binary Arithmetic Coding), 另一种是上下文自适应变长编码 (CAVLC-Context-based Variable-Length Coding)。

目前诸多国际视频编码一致采用混合视频编码框架 (HVC-Hybrid Video Coding)。混合编码框架是基于前面提到的四种编码方法的一种混合时间空间视频压缩方案, 首先采用帧间预测编码消除时间域的冗余信息, 然后对残差进行变换编码消除空间相关性, 经过量化之后去除空间的冗余信息, 最后进行熵编码去除统计熵的冗余度。Habibi 在 1974 年首次中提出混合编码的名称, Habibi

的方法首先对使用行变换编码消除空间域的相关性，然后在变换域内纵方向进行预测编码。1975 年，Roese 首先针对活动图像进行二维块变换，然后在时间域上对变换域系数进行 DPCM 编码。1979 年 Jain 首次提出了将变换至于 DPCM 环路之内的编码框架。1981 年 Forchheimer 将变换放在 DPCM 环路内，提出了一种新的混合时间空间压缩方案，首先用帧间预测编码减小时间域的冗余信息，然后对残差进行变换编码和量化消除空间的冗余信息，之后这一混合编码方案得到了广泛的应用，目前已经成为所有国际视频编码标准的编码框架的原型。

1.2.3 针对于不同视频编解码算法进行的加速系统研究

自从视频编解码标准出现的时候（MPEG-1, H.261），针对视频编解码算法进行加速的研究就开始了，加州伯克利分校、普渡大学、香港大学等针对 MPEG-1, H.261 的加速系统都进行过相关研究[14-16]；后续对于 MPEG-2 也有所研究[16, 17]。后续随着视频编解码算法变得越来越复杂，主流的处理器也越来越难支持算法的实时处理，所以对于 MPEG-4[18-21]和 H.264 的加速研究的人也越来越多[22-29]。而 MVC 虽然对于并行加速的设计需求更加迫切，但是由于它的标准尚未成熟，针对于 MVC 的并行加速研究并不多[30, 31]，但是随着 MVC 标准的进一步发展，我们相信它将成为大家研究的焦点。具体的研究情况参见表格。

表格 1 视频编解码算法并行加速的主要算法

	MVC	H.264	MPEG4	MPEG2	H.261	MPEG1
Intel		[23]	★			
IBM		★				
Berkeley Purdue						[15]
NTU		[20, 24, 25]	[20]			
CSELT			[21]			
中科院 北京大学	[30]					
香港大学					[14]	
Hannover Uni.			[19]			
Nokia		[26]				
SFIT				[16]	[16]	[16]

TUD		[27]				
Waseda		[22]				
NCTU		[28]				
Philips		[24, 29]	[18]	[17]		
清华大学	[31]					

1.3 在多核处理器上的可视化媒体加速算法的发展现状

近几十年，CPU 性能随着 CPU 时钟频率的提高而提高。这种提升主要来源于集成电路的发展，按照摩尔定律[32]在单位面积上集成电路元件的数量指数级增加。伴随这种增加，说明电路路径越短，从而支持更高的时钟频率。

然而这些年，传统结构的 CPU 性能指数级增长的速度放缓了许多，甚至停滞了。原因有以下几点：一，早期的 CPU 依靠增加更多的运算单元来提升性能，但是随着运算单元的增加，串行的指令流越来越难做到指令级的并行，使得额外增加的计算单元不能得到充分的利用；二，内存的速度虽然一直在上升，但是没有时钟频率提升得快，而这差距越发的大，致使内存的速度称为瓶颈，随之而来的解决方法就是利用越来越多的空间安放分支预测、乱序执行等电路；三，随着集成度的提高，功耗和散热问题也变得越来越重要。以上几点就制约了传统的单纯依靠提高集成度，提高时钟频率的方法提高 CPU 性能的老路。

基于以上情况，CPU 的设计者们开始寻找其他的道路以取代简单的依靠提高时钟频率而提高计算能力的办法。设计者们通过增加 SIMD 单元来提高指令级并行性，提出多核处理器体系结构等。事实证明这些增加并行性的方法都十分有效,例如 Nvidia 的 GPU 7800 GTX 可以达到 313GFlops。目前 GPU 和 CPU 又逐渐融合的趋势，GPU 的可编程性能越来越高，而 CPU 也加入了越来越多的核以及流处理单元。

目前关于多核处理器的研究是热点问题之一，各大科研院所和企业都在积极研发自己的多核处理器，例如：

科研院所：

- Wisconsin - Multiscalar(1995)
- Stanford - Hydra(1999)
- MIT - MultiALU(1998), RAW(2001)
- CMU - Stampede(1998)

- UT Austin - TRIPS(2003)
- Japan - OSCAR CMP (2003)

企业:

- IBM - Power4/5(1999), Cell(2004)
- SUN - MAJC(1999), Niagara(2005), SPARC64 (2006)
- Compaq & HP - Piranha (2000), MPOC (2002)
- Intel - Yonah (2005)
- AMD - Toledo (2005)

多核处理器从组织结构的角度可划分为：不对称多核处理器、对称多核处理器、异构多核处理器。

1.3.1 不对称多核处理器

不对称多核处理器与对称多核处理器相对，即各个核之间关系不对等，结构不相同。对于视频编解码的加速，软硬件协同设计的主从结构是比较典型的方法。主处理器完成主要工作，运行操作系统、负责调度等核心操作；从处理器负责某些专有任务的处理。二者相辅相成，各司其职，采用硬件加速[19, 33-35]。每个流水段较复杂，可以完成宏块级的运算，延迟时间不固定，采用信号握手协议来控制相邻流水段的数据流。

媒体处理是运算集中的算法，对于运算密集的部分，我们可以采用从处理器加速处理的办法，可以有效地提高处理性能。例如 AMD 的 Alchemy 即为主从结构的媒体加速处理器的代表，其体系结构如图 5 所示。其中“Enhanced MIPS32 CPU”是主处理器，“Media Acceleration Engine”是媒体加速的从处理器。

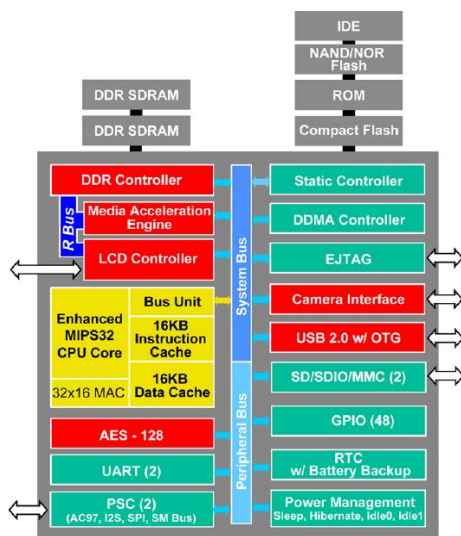


图 5 AMD 的 Alchemy 体系结构图

飞利浦实验室推出的 MPEG-4 解码系统是典型的软硬件协同设计的不对称多核处理器集成的平台[18]。

它首先将应用分解成独立的任务，有清晰的接口和自包含的功能。在分析了每个任务的特点（例如固有的可并行性、控制流程的复杂度、传输带宽需求和对于不同应用不用系统的可重复利用性）后进行软硬件划分。最后确定设计方案，它包括一个超常指令字的媒体处理器，一个或多个 RISC 处理器和一些定制专用处理器，如图 6。

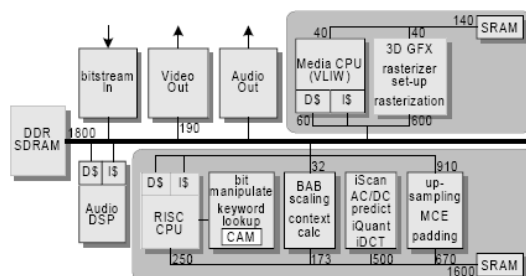


图 6 MPEG-4 解码系统结构图

这种针对应用的设计方案可以使得现有的设计水平和资源发挥更大的作用，达到良好的媒体加速效果，但是，它需要设计者对于媒体处理算法和芯片系统设计都有深入的了解，同时设计方案的转述性较强，可移植性和灵活性较差，不易复用，从而使得面对新的应用，设计周期比较长。

1.3.2 对称多核处理器

所谓对称多核处理器是指处理器内部的各个核结构相同，关系对等，相互对称，以 Intel 推出的双核、四核处理器为代表，他们从双核、四核发展为八核甚至 16 核，每一个核的体系结构都没有很大的变化。图 7 为 Intel 双核处理器的制版图。对称多核处理器的特点是设计成本低，可扩展性好。由于集成工艺的局限，目前的成熟产品中，最多只能将 8 个核集成在一个芯片上（SUN UltraSPARC 8 核处理器[36]）。但未来的发展方向是更多的核集成在一个处理器中。针对这种体系结构，如何把任务分配均匀，以及有良好的可扩展性是关键。参考文献[23]以最新的视频编解码标准 H.264 为例，在 Intel 的四核处理器平

台上优化并行，得到了良好的加速效果。

在优化之前，文献[23]对于 H.264 各个运算步骤的运算量及特点进行测试及分析，有针对性地进行 SIMD 优化[37]。根据 H.264 算法的特点和 Intel 处理提供的 SIMD 指令设计优化算法，加速比分别达到 2~3.5。

利用超线程技术[38]优化，加速比达到 1.18。在选择粗粒数据并行的时候，文献[23]对于 Slice、帧、宏块分别进行性能实验及分析。其中选择帧或者宏块作为并行粒度的话，不会降低压缩效率，如果选择 slice 作为并行粒度，随着每帧 slice 数目的增加会降低压缩效率。如果选择帧一级并行，那么可以获得的并行线程数很有限，选择宏块一级并行可以得到足够数量的并行线程数。所以文献[23]最终选择宏块一级的并行，由此引来了比较多的同步工作与并行算法设计的工作量。在对称四核处理器上进行实验后，加速比接近 4，而且算法的可扩展性良好。利用线程并行技术实现的 H.264 编码算法如图 8。

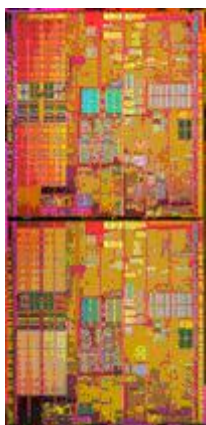


图 7 Intel 双核处理器的制版图

该方法的不足之处是：解码在 CIF 级利用单核可以做到实时；编码即便利用 4 核处理器及所有加速手段，仍无法达到实时要求。

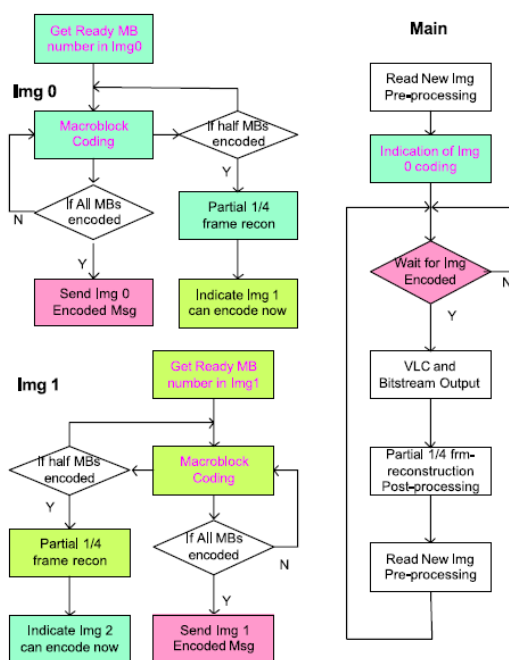


图 8 H. 264 编码的线程级并行流程图

1.3.3 异构多核处理器

异构式多核处理器是将对称与不对称多核处理器的结构相混合，融合双方特点。可能是对称的若干核组成的小组，各组之间关系不对称；也可能是关系对称的各个组，每组内部的核之间不对称。IBM 与 Sony、东芝共同设计开发的 Cell 处理器即是一款混合式多核处理器。一片 Cell 处理器有 9 个微处理器构成的芯片。核心微处理器是标准的 64 位通用处理器，有一级缓存和二级缓存，基于 PowerPC 技术的微处理器，称为 Power 处理元素（Power Processing Element, PPE）。另外 8 个微处理器与此不同，称为协同处理元素（Synergistic Processing Elements, SPE），针对媒体处理而设计。在 Cell 当中，SPE 负责具体的运算，PPE 负责调度和分配。Cell 的体系结构图见图 9[39]。Cell 处理器的特点还会在 6.2 节作详细介绍。

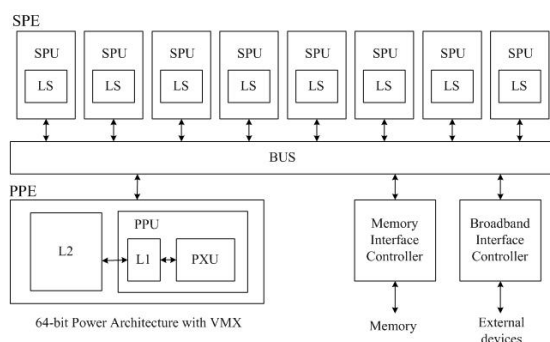


图 9 IBM Cell 处理器的体系结构图

Cell 处理器相对于传统的同时钟频率下的单核处理器加速效果很明显。SPE 向量运算可以比普通的单精度浮点运算快 4 倍。8 个 SPE 同时工作可以提高 8 倍，这样理论上共比典型的单核处理器快 32 倍。传输数据的时间和计算时间可以重叠一部分。当然，由于内存竞争，线程的互相依赖以及负载的不均衡，实际情况可能不可以完全达到峰值性能。

IBM 公司在 Cell 上实现了高清的实时解码。他们的方法是帧级数据并行与任务级并行相混合的并行架构。为与 Sony PS3 一致，让两个 SPE 闲置，剩下的 6 个 SPE 平均分为两组，每一组负责一帧图像的解码，两组并行操作。在一组内

部，将一帧的解码分为三大步，每个 SPE 负责一步，三步流水进行。同时充分利用 SIMD 技术优化性能。由此，可以做到 3 路高清 H.264 视频实时解码。

一个对于媒体算法比较了解的人，只要对于 Cell 的体系结构有所了解，熟悉它编程环境，那么可以比较好的发挥 Cell 的性能。从最终性能的提高，Cell 远优于 Intel 的对称多核处理器；从设计的复杂度来看，在 Cell 上进行媒体处理设计比设计不对称的媒体处理平台要容易得多，可以说类似于 Cell 的混合多核处理器是性能与设计复杂度平衡后的产物。

1.4 我的论文

论文分为八大部分介绍我硕士期间的工作，首先在阐述题目背景之后，介绍各种视频编解码的发展状况，简要分析各种编解码标准的基本技术及特点。在第三章介绍多核处理器的发展状况，从核的对称结构角度将多核处理器分为：对称多核处理器、不对称多核处理器以及异构多核处理器三种，具体介绍三种多核处理器的区别及在其上已有媒体并行加速工作。接下来以并行计算的方法论为桥梁，将多核处理器及媒体处理算法连接起来，分析媒体处理算法的特点，将并行计算的方法论运用到媒体处理算法上。第五、六、七三章分别选用可视化视频处理算法中具有代表意义的三项应用：H.264、光线跟踪、多视点视频编解码为例，在不对称多核系统以及异构多核处理器上进行并行优化，验证了在多核处理器上进行可视化媒体并行加速的可行性及有效性，最后给出结论。

第 2 章 媒体处理并行算法设计方法

2.1 预备知识

所谓并行算法设计，就是把一个应用分解为若干任务，根据各任务之间的依赖关系，画出依赖关系图。根据情况，选定并行的粒度，确定可以同时并行操作的单元与任务，以及各单元任务间的交互方式。把设计好的并行算法映射到多核处理器上，处理好进程和处理器的关系。下面具体介绍一些并行算法设计的技术。

2.2 分解技术

分解技术主要有递归分解、数据分解、探测分解、投机分解和混合分解几种方法。根据需要并行的算法的特点，选择合适的分解方法来设计并行算法。

递归分解适用于递归程序，例如快排序的方法。一个序列被分解为两个子序列，对于每个子序列的操作可以并行处理。

数据分解是比较常用的程序划分方法，分为两步，第一步是把数据的运算进行划分，第二步是把按照这些数据设计的运算进行划分成任务。可以参照输入数据、输出数据和中间数据等进行划分。

探测分解适用于潜在的运算需要搜索一个空间来获得解决方案，然后探测分解是把需要搜索的空间分为更小的部分，并行搜索以致达到所需的解决方案。

程序有很多的分支或者切换的语句，这些分支或切换的选择取决于之前运算的输出。**投机分解**是指这些分支语句的执行语句可以并行计算，当之前的决定性条件也运算完成之后，选择正确的分支继续进行。这会引来一些浪费的运算，我们可以根据一些历史的规律性情况，有选择进行提前的分支语句运算，提高命中率，减少浪费运算的比例。

混合分解就是将上述的各种办法，根据程序本身的特点选择两种或多种混合进行，以互补短长，达到更好的效果。

2.3 任务的性质

任务有静态的和动态的两种，随之而来的任务的大小就有确定的和不确定之分。有一些任务的大小是可以知道的，有些是不可以知道的。除了搞清楚任务是静态还是动态，大小是否可以确定的以外，还要知道任务涉及到的数据多少。

2.4 负载均衡的映射技术

并行计算的资源消耗主要有两个：各进程之间交互的时间损耗和一些处理器空闲浪费的时间损耗。所以，优化并行算法的两个主要出发点就是减少交互的时间和尽量让所有的器件都不空闲。

映射的技术往往和分解的技术相对应，例如根据任务的静态还是动态来选择映射方案，可以按照数据划分映射或者利用混合映射的方法等。

2.5 减少交互代价

本地数据最大化，使得需要的数据尽可能在本地，减少传输数据的次数，减少共享数据的大小和交互数据的频率。注意控制热点的数量和规模。

尽量使得运算和交互的时间或者交互和交互的时间可以重叠起来，以隐藏交互时间。为了减少交互，甚至可以引入冗余，复制数据和运算。

对于交互操作可以进行整合优化，尽量使用整合优化后的交互操作。

2.6 并行算法模型

● 数据并行模型

数据并行模型是最简单的算法模型，在这种模型中任务是固定的，可以直接映射到处理进程上，对于不同的数据进行相同的操作和运算。采用共享空间或者消息传递的方法都可以实现并行的同步。

● 任务图模型

任何一个并行算法都可以绘制出一幅任务依赖图，有些时候依赖图是微不足道的，例如矩阵的乘法，但有的时候任务依赖图可以用来清晰地表达映射方

法。在任务图形模型当中，任务之间的关系可以促进本地化和减少交互。

● 工作池模型

工作池和任务池的模型适合于动态映射的问题，不需要预先把任务映射到处理进程上，任务的指针存储在共享的列表、队列、哈希表或树等结构中。任务可以动态变化，把新增加的任务加到全局工作池里面，所有任务由工作池统一管理。空闲的处理进程探测工作池是否有任务，有的话就取出来执行。以此保证各个处理进程都最大限度的运转，资源有效利用。

● 其他模型

除了上面提到的模型以外，还有 Master-Slave 模型、流水线模型和混合模型等。

Master-Slave 模型是 Master 的程序负责分配调度，把任务分配给 Slave 的程序，Master 可以通过设置一些优先级来制定不同的分配策略，以达到资源最好的利用。

流水线模型是一种把互相依赖的串行程序并行化的方法，一个流的数据依次通过一连串的处理，又称流并行。

混合模型就是将一个或多个模型混合使用，根据程序自身的特点进行配合。

2.7 可视媒体处理算法特点

可视媒体处理算法以编解码算法为代表，已经发展了十多年，有其自己的特质，只有针对这些特质设计的处理器或并行算法才可以更好的加速视频算法。以下对于视频处理的算法特点进行简单总结：

● 实时响应

像视频会议、实况转播这样的媒体应用都要求实时响应，但是，同时视频处理的正确性也不像科学计算要求的那样高，例如在很多应用中哪怕跳过一些错误帧，以保证实时，也比毫无丢失的达不到实时处理速度的要好一些[40]。实时响应的特点要求硬件必须达到一定的处理能力，才可以做到实时响应效果。

● 流媒体

在传统的不连续的数据的处理中，大多数的数据是 32 位的，有的甚至是 64 位的，而人可以感觉到的只是很小的范围，同时感觉又是连续的，所以表示媒体的数据一般都是 8 位或 16 位的连续的数据。如果继续用 32 位或者 64 位来表示媒体数据，那么无疑是一种浪费，如果处理单元再小一点的话，浪费问题会更加突出。数据类型的变化为处理器设计又提出了新的问题。

● 可并行性

如上面介绍，可并行性从粒度大小的角度来说，可以分为粗粒度和细粒度的并行，从并行的元素角度可以分为数据并行和任务（控制）并行[41]，两个分类方法相交叉，得到四种并行单元。现有的视频编解码国际标准方案大多采用分块的编码方式，例如 MPEG-1、MPEG-2 和 H. 263 等视频压缩编码标准都采用 8 乘 8 的 DCT 和块大小为 16 乘 16 或 16 乘 8 的运动估计与运动补偿算法，不同的块在运算过程中没有相关性或者相关性很有限。除了块以外，视频编解码算法中的相对独立的数据组织结构从小到大有像素点、宏块、slice、帧、图片组（Group of Pictures, GOP），每个数据单元都可以作为数据并行粒度。从操作环节来看，最新的视频编解码算法，例如 MPEG-4、H. 264 都包含 DCT、量化、帧间/帧内预测、变长编码等步骤。这些运算的特点是规律性、迭代性和局部性，例如运动估计的基本运算是减法、绝对值和求和等，各操作之间关联较小，适合并行，同时运算密集度高，集中优化，效果显著。以上所有的并行性对应到四种并行单元，总结如表格 2：

表格 2 媒体算法的并行单元分类

	粗粒度	细粒度
数据并行	宏块、slice、帧、GOP	像素点
任务（控制）并行	DCT、运动预测/补偿等 视频编解码中常有的处理环节	指令

● 数据与指令的访问密集性

对于视频处理来讲，一般以宏块为基本处理单元，对于每一宏块的处理都是基本相同的，这就出现对于同一段程序反复访问，多个宏块的数据争相访问同一程序的现象。同时，同一段程序对于一个宏块（亮度宏块 256B，色度宏块 64B）的数据也会反复读取，例如 H. 264 的去块滤波操作当中，对于一个宏块数

据会从上到下、从左到右最多滤波 8 次。

● 高传输带宽

视频处理的数据量很大，为了达到实时处理的要求，每秒钟必须处理 30 帧数据，以标清为例，每秒需要处理近 18MB 数据，如果是多视点视频（以 8 个视点为例），那么处理的数据量提高 8 倍，为 142MB/s 数据。而实际需要的带宽不仅这么多，还有中间变量、输出结果等其他数据，所以对于传输带宽提出了很高要求。

2.8 视频处理算法并行化方法

针对于如上介绍的视频处理算法对于高速的要求，数据位数窄的特点，以及从数据并行、任务并行、粗细粒度并行都可行的特征，视频处理算法加并行化方法可以分为四类（表格 3）：粗粒度数据并行、细粒度数据并行、粗粒度任务并行、细粒度控制并行。

表格 3 视频编解码算法加速方法分类表

	粗粒度	细粒度
数据并行	以宏块、slice、帧、GOP 为粒度设计并行算法。	SIMD, Vector
任务（控制）并行	以 DCT、运动预测、运动补偿等视频处理算法中的处理环节为任务并行粒度设计线程并行算法或专有的硬件加速元件。	VLIW, 超标量

粗粒度数据并行主要指以宏块、slice 或者帧或图片组为数据并行粒度。在针对 MPEG-4 和 H.264 的加速系统中，以宏块为并行粒度的较多[16, 23, 28, 29, 42]。因为可以在保证压缩效率的情况下提供比较多的可并行线程。也有以 slice 为并行粒度的[27]，以此来降化同步策略。以帧和 GOP 作为并行粒度的方案较少，因为对于主流的视频编解码算法，由此得到的可并行线程数量比较少，可扩展性差。

细粒度数据并行主要是 SIMD 技术，采用扩展指令集的方法，把媒体处理指令加至通用处理器的指令集[37, 43-48]。这种方法在细粒度并行方面效果显著，对于视频处理算法的加速比可以达到 2^4 [23]。

对于视频处理算法的粗粒度任务并行主要指的是可以流水并行处理的 DCT、运动预测、运动补偿等。以 H. 264 为例，绝大多数的计算量集中在运动预测运动补偿部分，所以针对这一部分集中优化并行处理，可以得到较好的性能提升 [17, 18, 20, 21, 24, 25, 28, 29, 49, 50]。

细粒度的任务并行主要方法是利用超长指令字 (Very Long Instruction Word, VLIW) 和超标量技术。这两个技术是提高处理器性能的一个重要但通用的手段，并不是针对某一应用领域的特殊手段，对于媒体处理的加速会有一定的加速作用，但比较有限。他们往往作为加速性能的一个方面与其他方法混合使用。[17, 19, 24, 29, 48, 51-53]

表格 2 中所列方法并不是互斥的，一个优秀的视频加速并行系统，往往是融合了多种方法于一身，从多方面提高媒体处理的速度 [14, 31]，例如在不同的层次进行数据并行，在利用 SIMD 优化之后，还会采用宏块并行的方法，最终达到实时处理的效果 [23]。所以，以并行方法分类的思路很多处理器会属于两类甚至多类，从视频信号处理器分类角度做不到“不重”的分类原则。下面本文将从体系结构的角度进行分类。

第3章 基于主从不对称体系结构的 H.264 解码加速系统设计与实现

由于新兴的视频编解码标准 H.264 虽然压缩比高,但是运算量也比较大,需要特殊的处理器支持。本章主要介绍的是利用主从不对称结构的处理器来完成对于 H.264 解码的加速系统的设计与研究。

3.1 H.264/AVC 算法介绍

最新的 MPEG-4 AVC/H.264 视频编码标准是由 ITU-T 视频编码专家小组 (Video Coding Expert Group) 和 ISO/IEC 运动图像专家小组 (Moving Picture Expert Group) 的联合视频小组 (Joint Video Team) 合作推出的[54]。其应用背景主要包括以下几个方面:

- 面向 cable、卫星、cable modem 和 DSL 等网络的视频广播;
- 面向 DVD 和大规模存储设备的视频存储;
- 面向 ISDN、以太网、局域网、DSL、无线局域网和其他一些移动网络的视频会话服务;
- 面向 ISDN、cable、DSL、局域网、无线网的视频点播和流化;
- 面向 ISDN、以太网、局域网、DSL、无线局域网和其他一些移动网络的视频的多媒体讯息服务 (multimedia messaging services)。

MPEG-4 AVC/H.264 视频编码的框架与先前的一些标准类似,都是采用运动补偿、正交变换、量化和熵编码等技术。但是,在运动预测、正交变换、熵编码、容错性和传输的灵活性上融入了最新的编码技术,达到了非常好的视频编码性能。

MPEG-4 AVC/H.264 将 H.26L 的初期目标,从提高编码效率以适应低码率视频传输的需求,扩展到从低码率到高码率(超高清视频图像压缩)的应用,编码效率比 MPEG-4 高 50%,目前这一目的已经基本实现。其实 MPEG-4 AVC/H.264 中用到的许多编码工具在早期就有提出,只不过由于当时硬件设备的限制和复杂度过高无法被采纳,而伴随着硬件的发展,很多算法又重新被采纳。

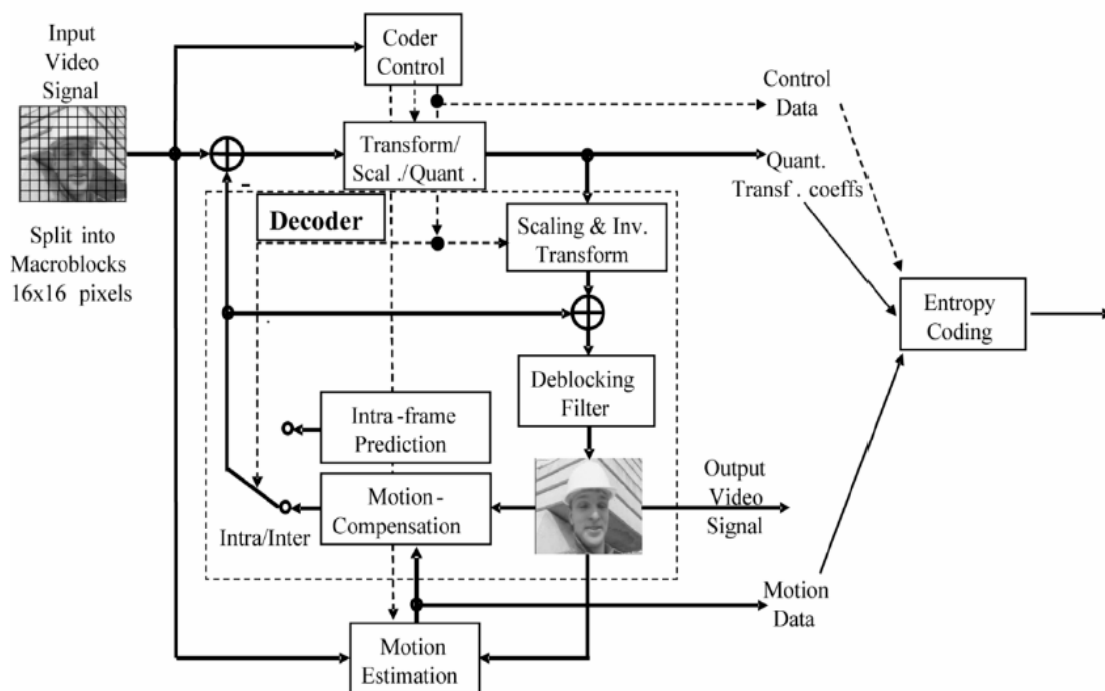


图 10 MPEG-4 AVC/H.264 的编码结构

MPEG-4 AVC/H.264 的编码结构如图 10 所示。从编码结构来看，MPEG-4 AVC/H.264 仍然是基于运动估计补偿的结构，这和 MPEG-1，MPEG-2，MPEG-4 标准是一样的，而不同的是它采用了一些新的编码方法：intra 预测，7 种运动补偿块，4x4 整数变换，UVLC（Universal Variable Length Coding），CABAC（Context-based Adaptive Binary Arithmetic Coding），和环滤波器（Loop Filter）等。

Intra 编码中的空域预测分为 4x4 和 16x16 两种块大小。intra 4x4 包括空间 8 种方向性预测和 DC 直流预测，8 种方向是指 180°、22.5°、45°、67.5°、90°、112.5°、145°、167.5°等，如图 12 所示。而运动估计模块是影响性能最大的部分，H.26L 采用了 7 种自适应块形状，如图 11 所示，其中为了加速编码器处理，它首先从 16x16，16x8，8x16，8x8 四种模式中选择一种预测残差最小的 inter 模式，如果是 8x8 模式，那么还要进行 8x8，8x4，4x8，4x4 等四种模式的选择。Intra 16x16 包括水平、竖直、DC 和平面预测四种模式。

4x4 整数变换可以加速编解码处理速度，同时由于变换都是整数运算，所以在 MPEG-4 AVC/H.264 中变换和量化可以合在一起处理的，并不是和 MPEG-4 那样先进行 DCT 变换，然后是量化，这样做的好处就是既可以减少一次除法运

算，又可以减小舍入误差。

UVLC 是一种特殊 VLC 编码，它属于前缀编码，如果需要编码的码字信息位（二进制）为 3，那么它将被编码成 (0 0 0 1 x₂ x₁ x₀)，其中 1 前面的三个 0 表示信息位有 3 位，这种码字和 MPEG-4 VLC 相比容错能力更强，这主要是 MPEG-4 AVC/H. 264 考虑到无线网络传输中误码率高的特点设计的，另一方面由于 H. 26L 都使用这种码字进行编码，所以在硬件实现时只需要存一个 VLC 码表，而不需要存多个码表。由于残差的 UVLC 码字是专门针对低码率设计的，所以在高码率时编码效率较低，所以 MPEG-4 AVC/H. 264 还设计了一种采用了基于上下文的二进制算术编码 (CABAC)，它可以根据已有的概率模型对各种符号进行编码，并在编码过程中对概率统计进行调整，所以在不同码率下 CABAC 可以取得较好编码效率。

由于 MPEG-4 AVC/H. 264 使用了较小的块变换量化编码，环滤波器 (Loop Filter) 主要是针对 4x4 块边缘的不连续性进行平滑滤波，这可以使参考图像质量更加平滑，而环滤波器是在运动补偿环以内，所以滤波后的参考图像有利于后续帧的运动估计，得到更加平滑的运动向量，减少运动向量编码码率，同时提高运动补偿的效率。它的缺点就是增加了编码的计算量。

MPEG-4 AVC/H. 264 的编码属于运动补偿预测多种编码方法混合的编码系统 (Motion Compensated Prediction Hybrid Coder)，首先输入视频都要经过 Intra 预测模块，并选择预测残差最小的 intra 模式作为候选 intra 编码模式，如果输入帧是 I 帧 (intra 编码)，那么就直接进行变换和量化，然后进行 UVLC 或者 CABAC 编码；如果输入帧是 P 帧 (inter 编码)，那么就要进行运动估计，选择最佳的 inter 编码模式，然后从候选 intra 和 inter 选择该宏块的最佳编码模式 (图 11 中的开关 S₀ 完成)，选择方法是计算率失真函数式 (1-1)，选择代价最小编码模式，最后再对所选预测残差进行变换和量化。

$$J(s, c, MODE | QP, \lambda_{MODE}) = SAD(s, c, MODE | QP) + \lambda_{MODE} \cdot R(s, c, MODE | QP) \quad (1-1)$$

其中 $SAD(s, c, MODE | QP)$ 是预测残差的绝对值和，而 λ_{mode} 是指在码率下失真对码率的斜率绝对值 ($-dD/dR$)， $R(s, c, Mode | QP)$ 是指在模式 $Mode$ 下的编码码率。

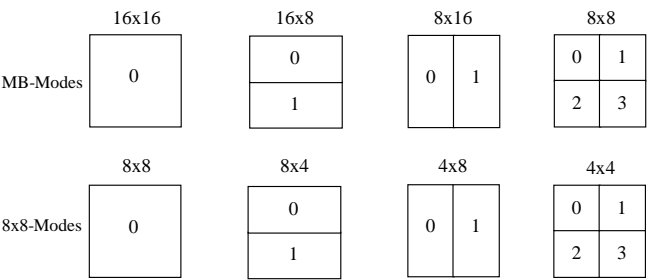


图 11 7 种不同运动补偿块

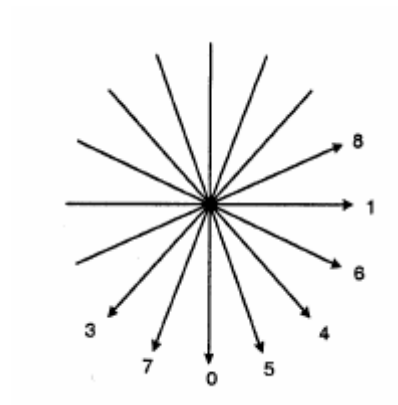


图 12 intra 编码的 8 种不同空域预测模式

3.2 总体设计流程

总体的设计流程如图 13 所示。

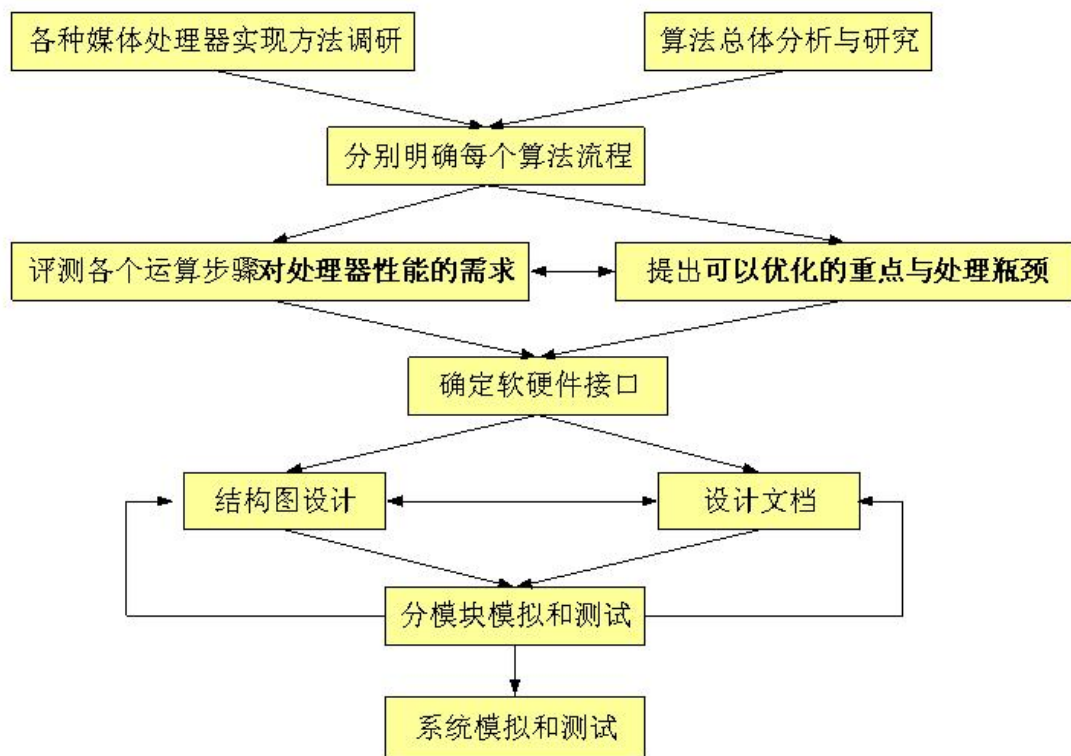


图 13 设计流程图

CPU 的设计流程是不断循环反复验证的过程，上述流程只是一个循环结束而已，确定最终的体系结构，还需要不断按照上述流程反复验证修改。但上述流程不仅适合于 H.264 这种视频编解码的算法，同时也适用于各种媒体处理器的设计，具有普适性。同时，通过我们的设计实现可以证明该方法的可行性。

3.3 总体设计方案

综合前面调研的结果和算法的分析情况，我们选择的最终的设计方案是基于片上多处理器（CMP）的软硬件协同设计的 H.264 媒体加速系统。片上多处理器在降低设计复杂度的同时不降低运算速度，以保证媒体处理的实时性。软硬件协同设计保证了整个系统的灵活性，为今后扩展支持 MPEG-4，AVS，WMV9 打下了基础。

整体的体系结构如图 14 所示，通用 CPU 与媒体加速协处理器构成异构的片上多处理器结构，它们通过总线交流数据。其中通用 CPU 的 MIPS core 是采用

清华大学 CPU 中心研制开发的 thump 芯片。

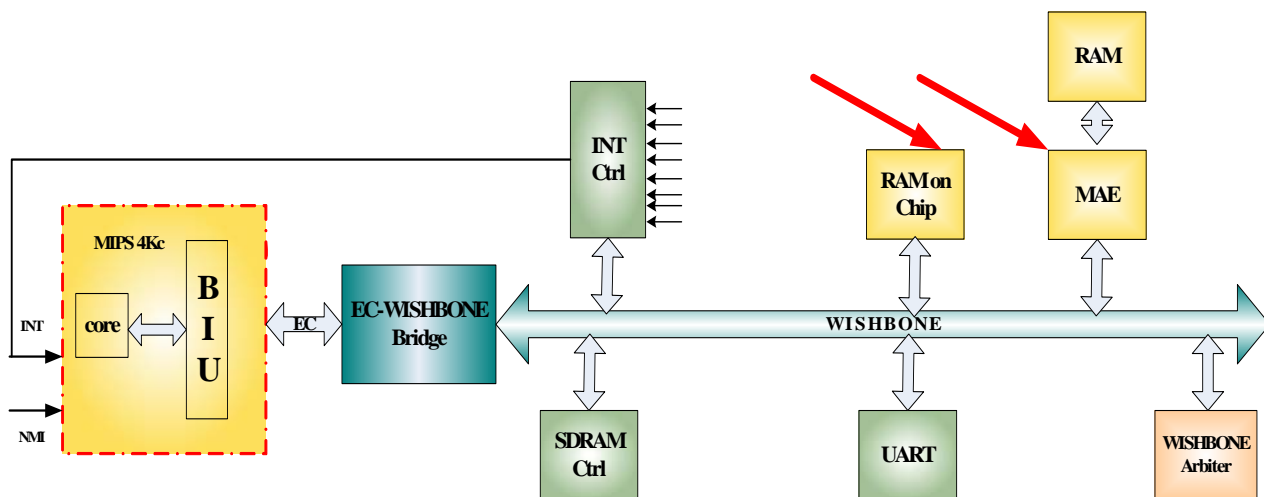


图 14 体系结构整体图

MAE（媒体加速引擎）的具体内部构造如图 15 所示。在文章的稍后部分着重介绍作者的主要工作——去块滤波部分的设计与测试。

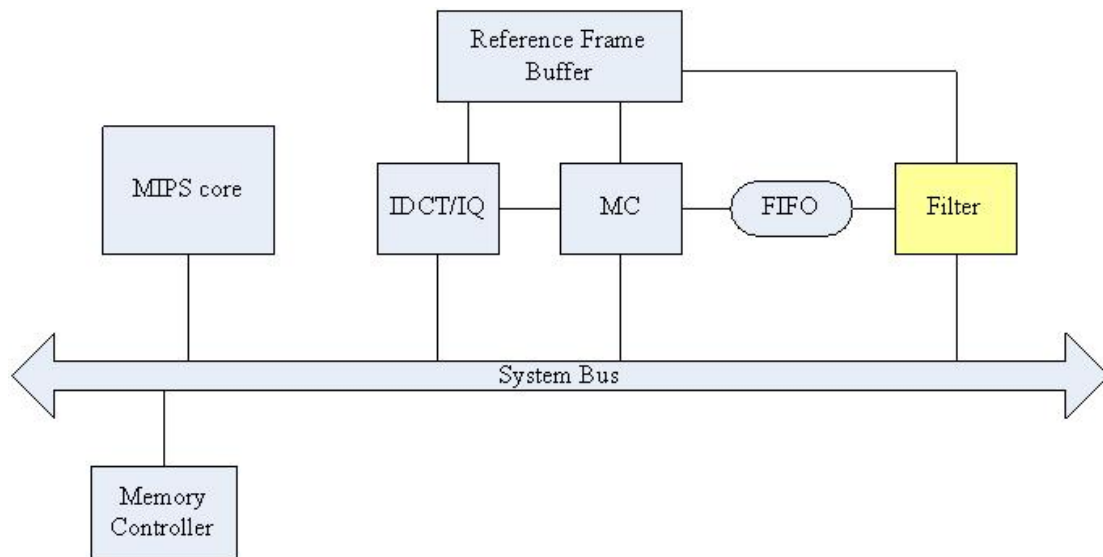


图 15 媒体加速引擎

3.4 总体设计目标

● 验证软件

这一部分是根据 H.264 解码算法实现的与体系结构相对应的解码软件。它追求的不是良好的性能，而是正确性。验证软件的结构与硬件的体系结构相对应，可以从验证软件当中的各个部件之间 trace 下数据，这样，硬件设计时各个模块的输入输出数据都有了参考，可以验证单独一个部件的正确性，这对于体系结构设计非常必要。

● 体系结构设计验证

另一项工作是提出媒体处理器的设计瓶颈，提出优化的体系结构设计方案。步骤是采用无计算测试模式得到测试数据，结合理论分析从而得出结论。在第五章当中会针对去块滤波部分作详细的介绍。

3.5 H.264 中去块滤波部分的算法流程详解

传统基于块的视频编码系统，在相对码率较低的视频编码时总会遇到块效应这个问题。该问题是由于基于块的各种操作造成的。为了解决这个问题，人们提出了各种方法[55]，例如重叠的正交变换、嵌入的零数小波变换等方法。这些方法虽然效果比较好，但是他们改变了基本的编码方式，而基于块编码方法的性能、复杂度、兼容性和市场需求都比较好，所以目前国际标准中主要还是使用基于块的编码方式。而块效应会随着前一个重建帧而积累下去，所以在编码环中引入去块效应的滤波系统作为一种后处理系统在提高视频图像性能方面具有极大的重要性。

去块滤波系统是 H.264 的重要组成部分，它是 H.264 在相对码率较低的情况下依旧能保持较好的主观视觉效果的重要因素之一。它的优点主要有两个：1. 图像中由于运动补偿、变换及量化产生的虚假边界可以被平滑，降低图像块效应，提高了主观视觉效果；2. 滤波后的帧用于后续帧的运动补偿预测，从而避免了虚假边界累计误差导致的图像质量的进一步降低。

由于 H.264 中的整数变换是基于 4×4 点的，所以去块滤波的处理单元也是 4×4 点的块。对于每一个 4×4 亮度块的边界都定义有“块边缘强度”。边缘强

度如图 16，由此可见它的大小主要取决于编码中产生的残差大小。

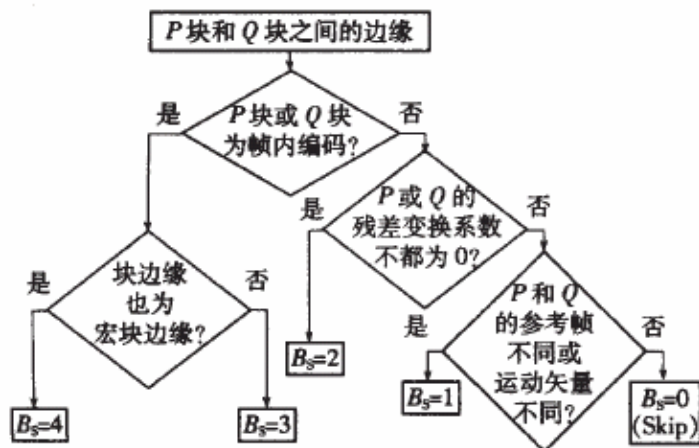


图 16 块边缘强度计算树

块效应的产生是由于运动补偿和量化误差造成的，同时视频图像序列本身还存在着物体的真实边界。一般来讲，真实的边界两侧像素的梯度差要比虚假的大，因此为滤波设定了两个门限值，判断是否要对边界滤波。

具体算法流程如图 17 所示：

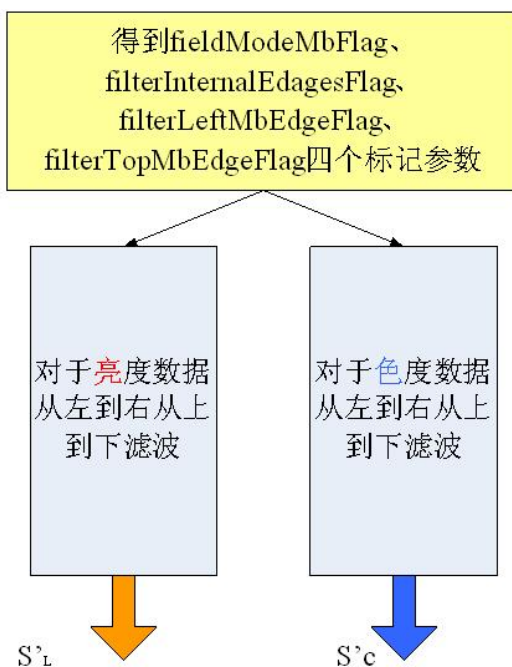


图 17 去块滤波整体流程图

图 17 所示为去块滤波的整体流程图。首先计算出若干参数，然后亮度滤波和色度滤波可以并行处理，按照从左到右从上到下的顺序滤波。亮度滤波和色度滤波的流程类似，只是数据量不同，以亮度滤波为例，如图 18 所示。亮度滤波按照特定顺序执行，其中核心操作是标准中 8.7.1 节，由于后面的操作依赖并可能修改前一操作的结果，所以不可以做并行，只能做串行和流水。

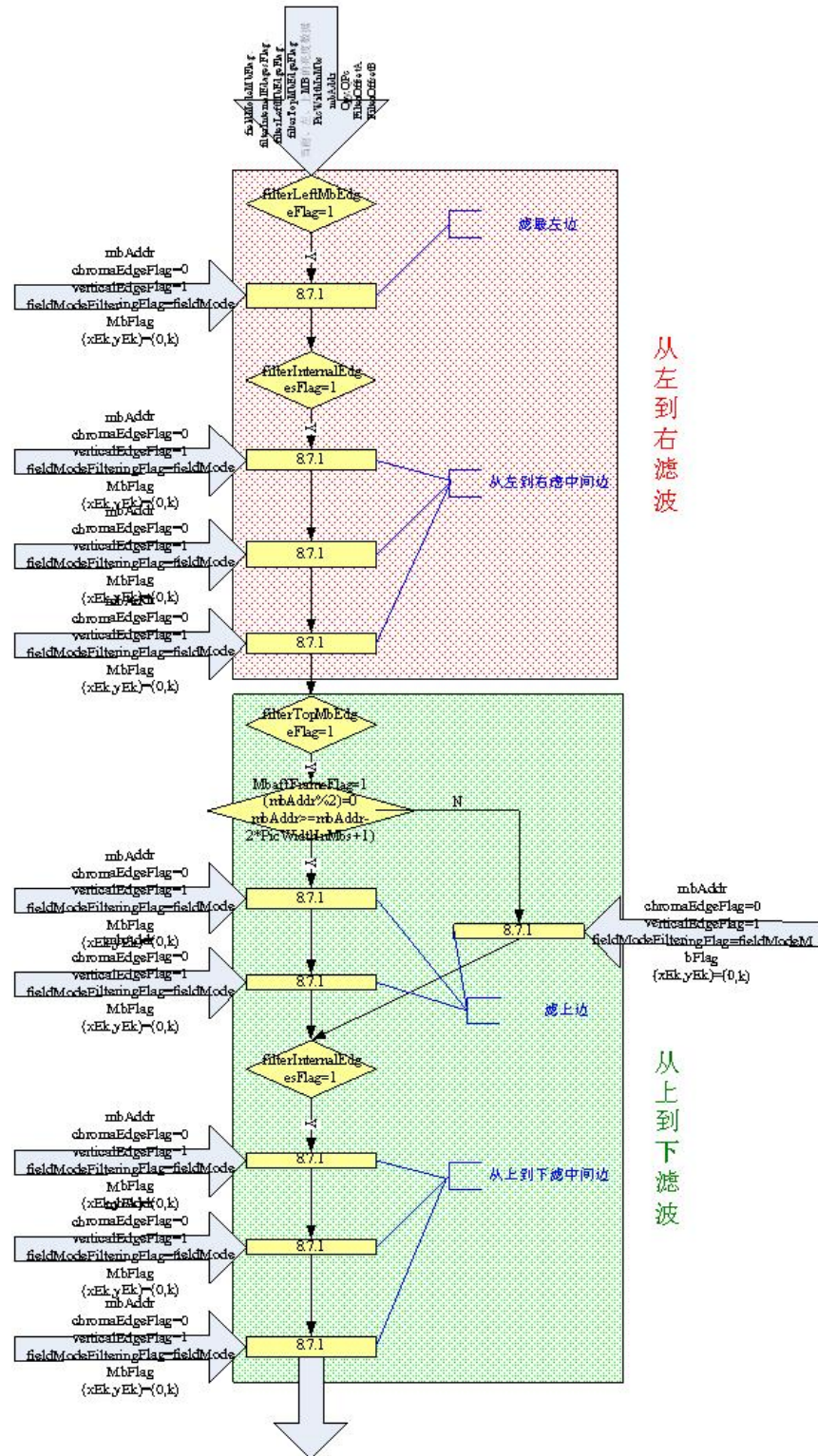


图 18 亮度滤波流程图

8.7.1 核心滤波操作如图 19 所示。

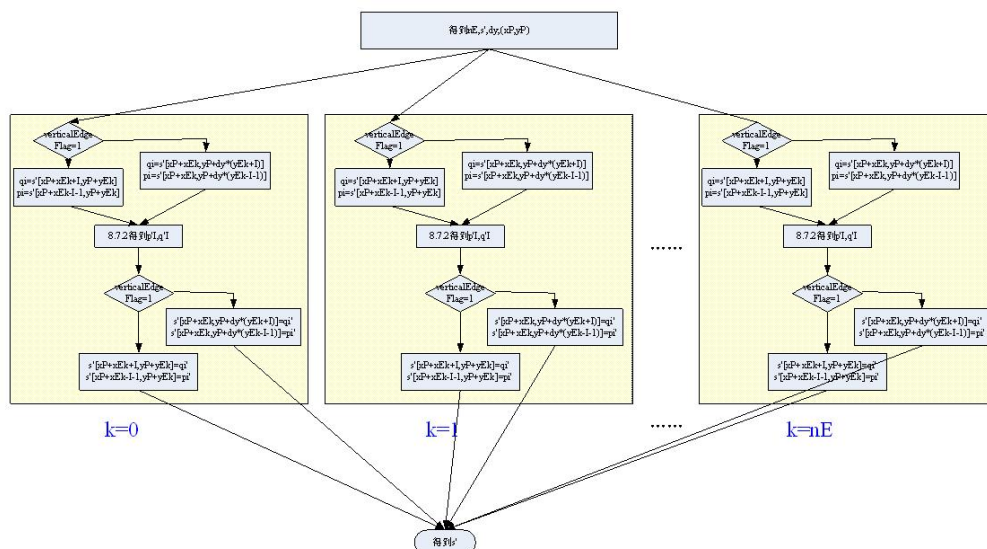


图 19 8.7.1 核心滤波算法

8.7.1 是对于边上的每一个像素点进行相同的处理，并且各像素点的滤波没有关联，可以并行。对于每个象素点的操作流程如图 20 所示。

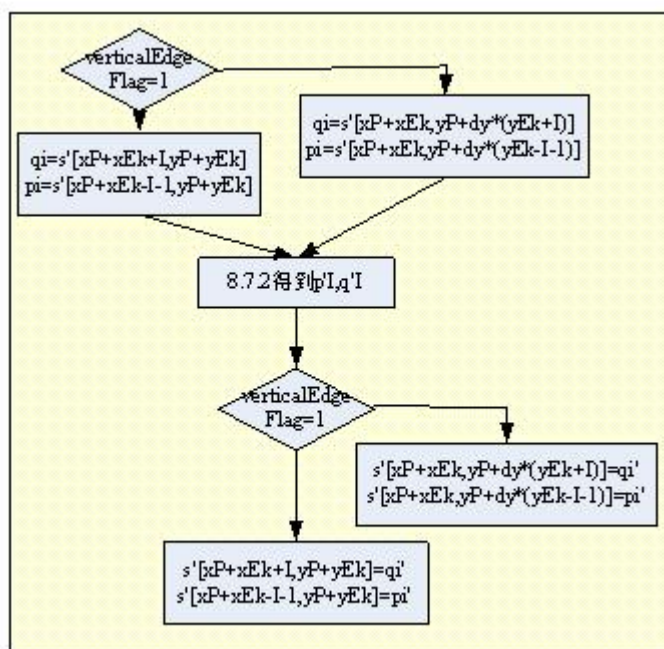


图 20 对每个像素点的滤波处理

在像素点的滤波处理中，8.7.2 得到 q'_i , p'_i 的操作是最关键的，它的算法流程图如图 21 所示。

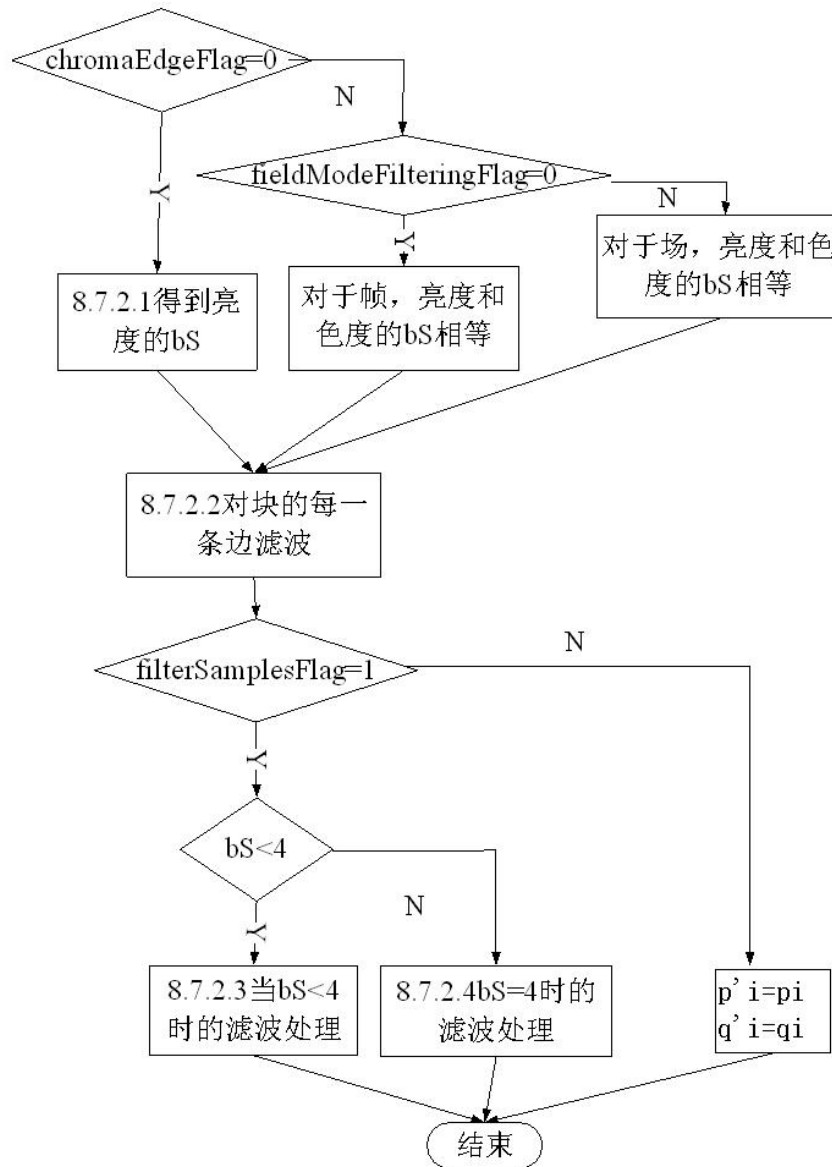


图 21 8.7.2 得到 q'_i , p'_i

综上所述，去块滤波的算法是从粗到细，逐渐深入的。其中亮度色度滤波之间虽然有 bS 参数的依赖，但总体来讲可以并行处理；边的滤波过程由于后面的依赖于前面的结果，不可以并行，但对于同一条边上的像素点的处理彼此间

没有关系，可以并行处理。

3.6 H.264 中去块滤波部分的算法实现

3.6.1 去块滤波的功能

去块滤波是整个解码过程的最后一步，目的是消除块效应。它的处理单元是一整幅图片，处理过程是按照 MB（宏块）来进行的，对于一幅图像的每个宏块的亮（色）度数据均按照从上到下从左到右的顺序依次滤波，算法详见 2.4.3。

3.6.2 去块滤波部分的设计

针对去块滤波部分的算法特点进行设计。去块滤波部分的算法并不复杂，但是运算量很大，数据量也很大，所以设计的难点在于满足实时处理的前提下如何降低频率与功耗和减少存储量与 I/O 带宽，但设计中发现很多时候两者是相互制约的，所以设计的最终目的是在二者之间找好平衡点。

整个去块滤波的实现设计分为三大部分：输入部分、滤波运算部分和输出部分。作者在软件模拟器上实现了这一部分的体系结构，并且实现了可配置性，通过调节参数，可改变体系结构，十分便于测试。这些参数统一写在 MAEGlobal.h 的头文件中。根据得到的大量测试数据，得出优化方案。下面对去块滤波的模拟器实现与测试进行详细的介绍。

● 输入输出接口

去块滤波之前的设备（MC）与去块滤波之间依靠 FIFO 传递数据，如图 23。由于 MC 的处理单元是宏块，而且处理顺序不定，所以 FIFO 缓存了一整幅图的相关数据。MC 每处理好一个宏块，就把相关数据放入 FIFO，去块滤波在处理数据前把整幅图的数据取进自己的缓存中。下面对 FIFO 和去块滤波的输入输出接口作详细说明。

FIFO:

FIFO 的输入输出与存储单元是同一类型个，包含着一个宏块的亮（色）度

数据和相关参数如表格 4。

表格 4 FIFO 的处理单元

类型	名称	功能描述	比特位
int	PicNum	记录宏块所在图像的编号	8
int	field_pic_flag	标记是否为场编码	1
int	MbaffFrameFlag	自适应标记	1
int	disable_deblocking_filter_idc	是否滤波	2
int	FilterOffsetA	滤波计算 indexA 的偏移量	5
int	FilterOffsetB	滤波计算 indexB 的偏移量	5
int	PicHeightInMbs	以宏块计算图像的高	8
int	PicWidthInMbs	以宏块计算图像的宽	8
int**	LData	一个宏块的亮度数据	$16 \times 16 \times 8$
int**	CrData	一个宏块的色度数据	$8 \times 8 \times 4$
int**	CbData	一个宏块的色度数据	$8 \times 8 \times 4$
Macro Block _par	mb	有关该宏块的参数	

其中 MacroBlock_par 的参数表如表格 5。

表格 5 MacroBlock_par 参数表

类型	名称	功能描述	比特位
int	avail	使能位	1
int**	blkAvail	每一个子块的可用性	$4 \times 4 \times 1$
int	mbAddr	宏块在图像中的位置	16
int	sliceNum	宏块所在 slice 的编号	16
int	mbIsFrame	宏块是否是帧编码	1
int	intra	宏块是否是帧内编码	1
int	IPCM	宏块是否是 IPCM	1
int	QpY	量化参数	6
int	QpC	量化参数	6
int	cbp_blk	每一个 bit 指示一个 4×4 的 blk 中是否有 non-zero level.	32
int	mbPartIdc	mbPart 以及 subMbPart 的划分方式	2
int	intra4x4	标识这个宏块如果是 intra 的话是否 4×4	1

int*	refIdx	每一个 8x8(for luma, and for chroma is 4x4) 的 partition 对应的参考图象	8×4
int**	mv	每一个 4x4(for luma, and for chroma is 2x2) 的 blk 对应的运动向量, 以 1/4 像素精度为单位	16×16×2

综上所述, 一个 FIFO 处理单元的大小是:

$$\text{Size(FIFO)} = 8 + 1 + 1 + 2 + 5 + 5 + 8 + 8 + 16 \times 16 \times 8 + 8 \times 8 \times 4 + 8 \times 8 \times 4 + 1 + 4 \times 4 + 16 + 16 + 1 + 1 + 1 + 6 + 6 + 32 + 2 + 1 + 8 \times 4 + 16 \times 16 \times 2 = 2598 \text{ (bits)}$$

这是相关一个宏块的数据量。FIFO 的输入端口宽度是一个 FIFO 的处理单元。FIFO 的输出端口宽度是 FIFO 的处理单元的 n 倍 ($n \geq 1$, n 为整数, n 是一幅图像最大宏块数的约数), n 取决于结构设计。

去块滤波

去块滤波的输入端口与 FIFO 的输出端口一致。去块滤波的输出分两部分: 写总线。写参考帧的缓存, 以供运动估计之用。这两部分的输出方式有多种, 详见 5.2.4, 但无论怎样的结构, 输出的最小单位都是宏块级数据。

输入数据量: 处理一幅图片数据前总的输入数据量是 $\text{size}(\text{FIFO}) \times \text{MAXMBNUM}$, 我们在实验中实现的是标清 (SD), 其中 $\text{MAXMBNUM}=1620$, 此时一幅图总的输入数据量是 $\text{size}(\text{FIFO}) \times \text{MAXMBNUM} = 2598 \times 1620 = 4208760 \text{ (bits)}$
=514 kB

输出数据量: 为了满足解码图像实时播放的要求, 输出数据的速率要达到 30 帧/秒。输出只需要数据, 不需要再传递参数。所以输出速率为:

$$(16 \times 16 \times 8 + 8 \times 8 \times 4 + 8 \times 8 \times 4) \times \text{MAXMBNUM} \times 30 = 124416000 \text{ bits/s} = 14.8 \text{ MB/s}$$

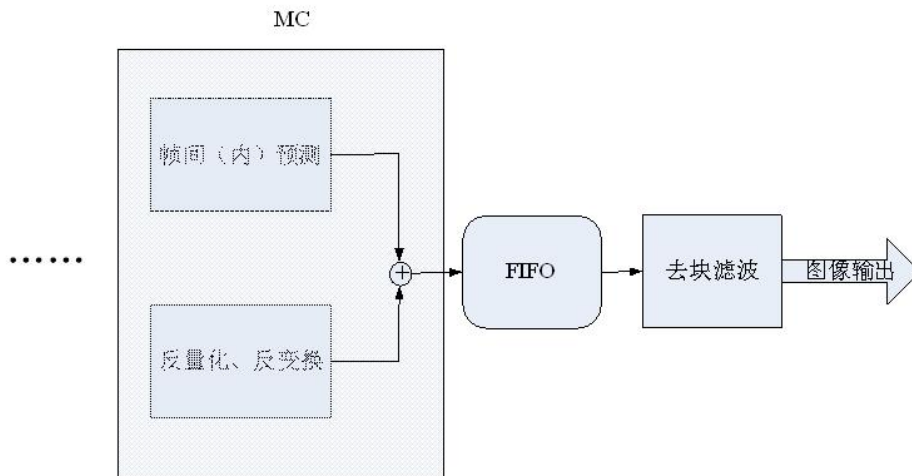


图 22 去块滤波与其他设备间的连接关系

● 输入部分设计

输入部分的调节参数是：

```
#define INPUTWIDTH 1//一个周期输入几个宏块的数据
#define TWOPICBUFFER 1//标记是否有两个存储 pic 数据的 buffer，1 表示有，0 表示没有
```

当 filter 只有一个存储整幅数据的 buffer 时（TWOPICBUFFER=0），INPUTWIDTH 这个参数可以调节输入端口的宽度。参数代表每个周期最多读取的 FIFO 处理单元的个数。该参数的取值范围是 1~MAXMBNUM（一幅图像中所容的最多宏块数）。

当 filter 有两个存储整幅数据的 buffer 时（TWOPICBUFFER=1），INPUTWIDTH 只能为 1。设计两个 buffer 是为了一个用于滤波计算，一个用于缓存从 FIFO 读出的下一个宏块的数据，这样可以运算与读数据并行，提高效率，同时不需要太大的输入端口带宽。实现的方法是设一个标记为 bufferflag，两个 buffer 我们称之为 buffer0 和 buffer1，如果 bufferflag=1，那么 buffer1 用来滤波计算，buffer0 用来缓存读取的数据；如果 bufferflag=0，则相反。

● 滤波计算部分设计

滤波计算部分是主要部分，有两个参数调节。

```
#define LC 0//表示亮度和色度滤波间的连接关系，0表示串行，1表示以宏块为单位流水，2表示以边为单位流水
```

#define MBCYCLE 1//表示滤一个宏块的周期，除 1 以外还可以是 8

1. 亮度和色度滤波间的连接关系

(1) 串行连接

这种体系结构忠实于原算法标准，真正按照先滤波亮度数据，再滤波色度数据的顺序执行。结构图如图 23 所示。

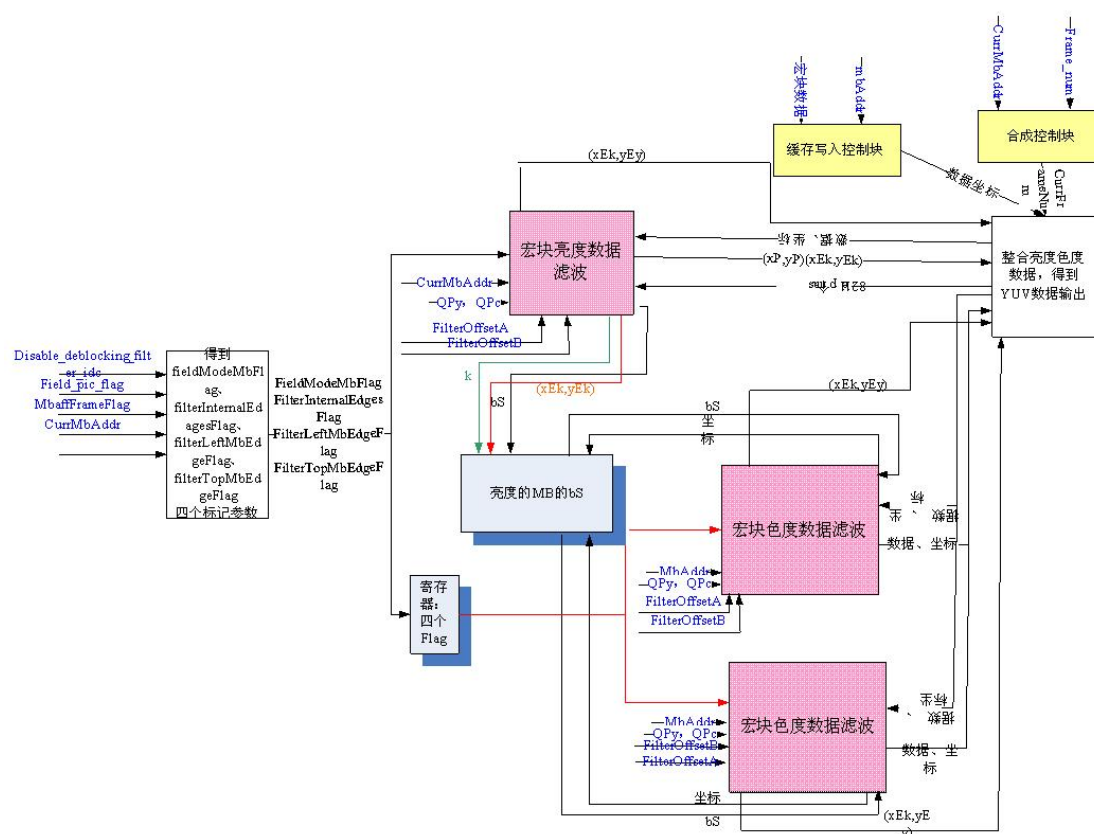


图 23 亮度色度滤波串行结构图

(2) 以宏块为单位流水处理

这种结构使得同一周期当中亮度滤波比色度滤波提前一个周期。但滤一个宏块的总时间缩短了。同时，由于色度滤波需要同一宏块的亮度滤波计算出来的参数 bS ，所以与前一种方案相比，增加了一个存储 bS 的缓存。

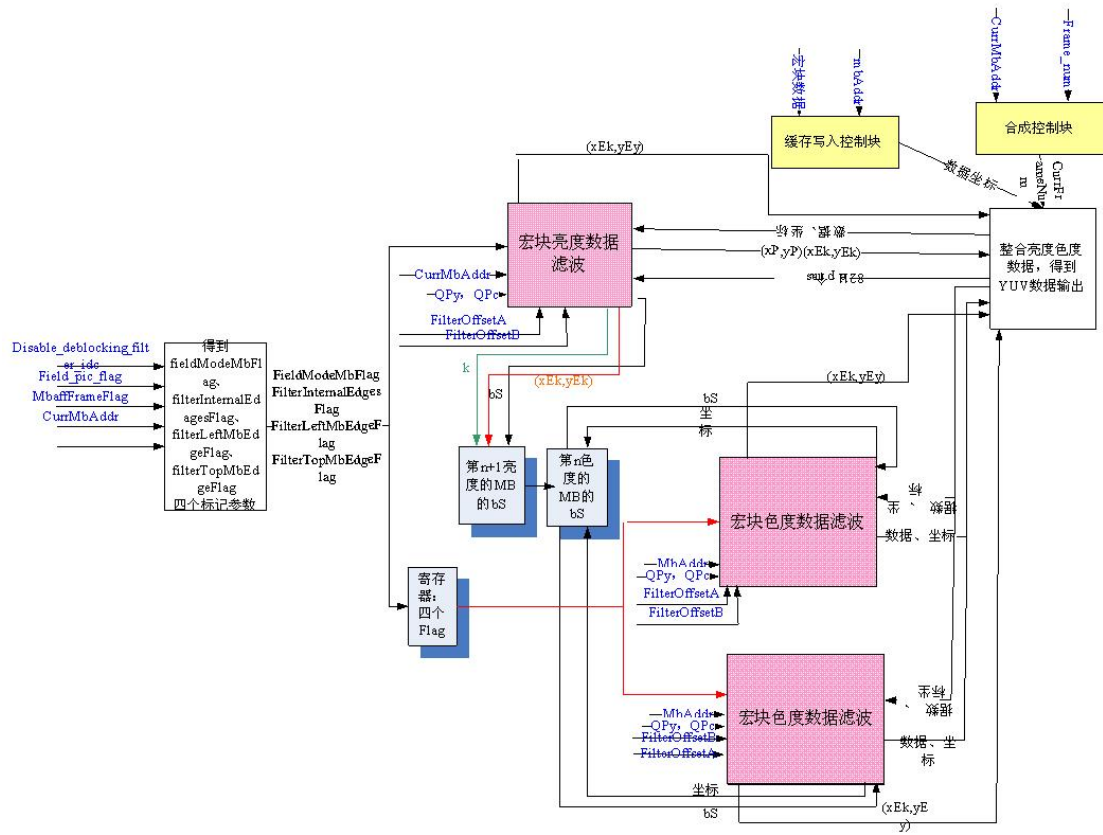


图 24 亮度色度滤波以宏块为单位流水连接结构图

(3) 以边为单位流水处理

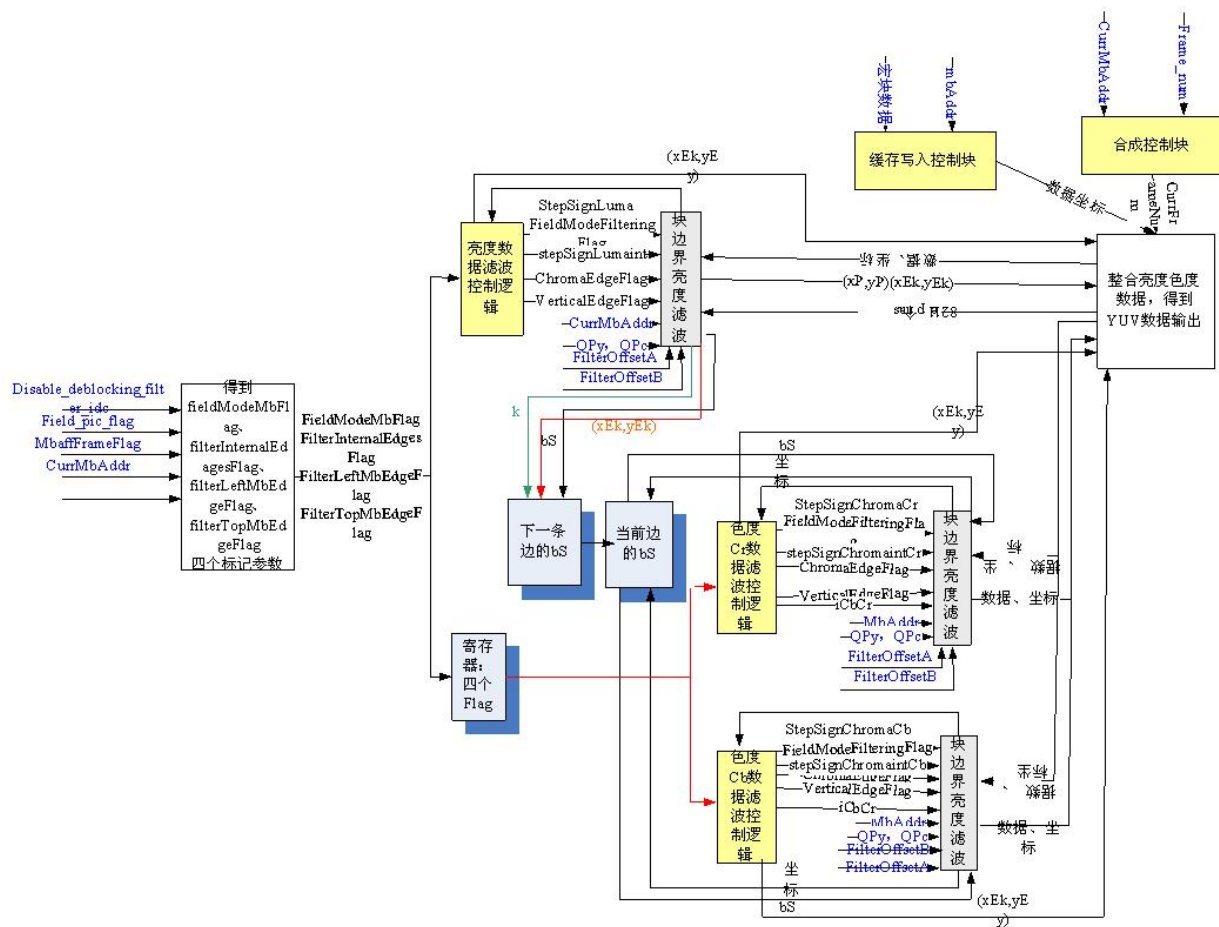


图 25 亮度色度滤波以边为单位流水连接结构图

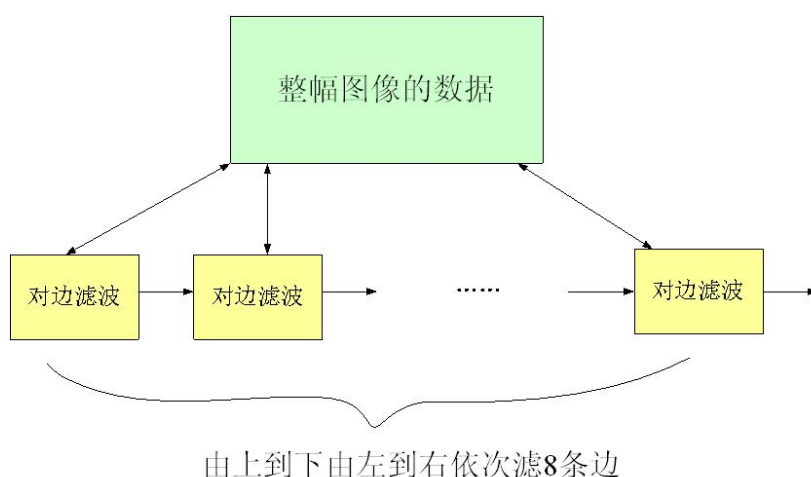


图 26 处理一个宏块数据流程图

2. 处理一个宏块的方式

(1) 处理一个宏块需要 8 个周期，算和写一条边需要一个周期。

(2) 处理一个宏块需要 16 个周期，即算和写一条边分别需要一个周期。

例如处理一个宏块需要 8 个周期, 即滤一条边需要一个周期, 图 27 所示是滤一条边的内部结构, 除此之外还有控制器控制滤边的顺序等逻辑, 如图 28。

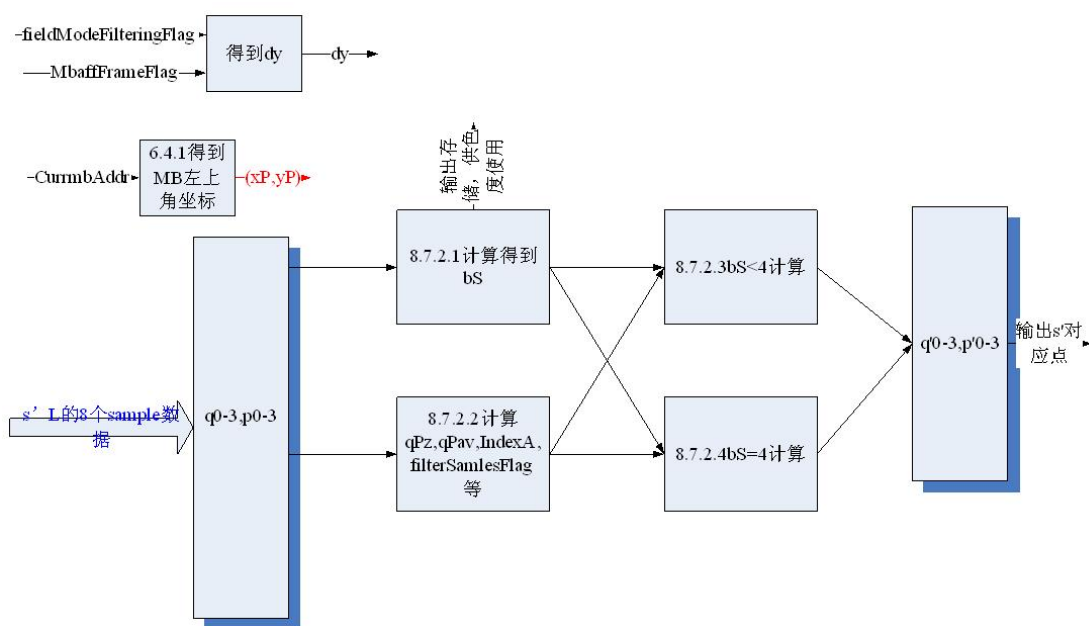


图 27 滤宏块中一条边的结构图

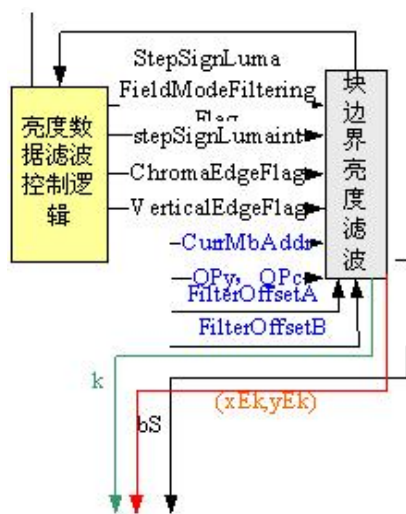
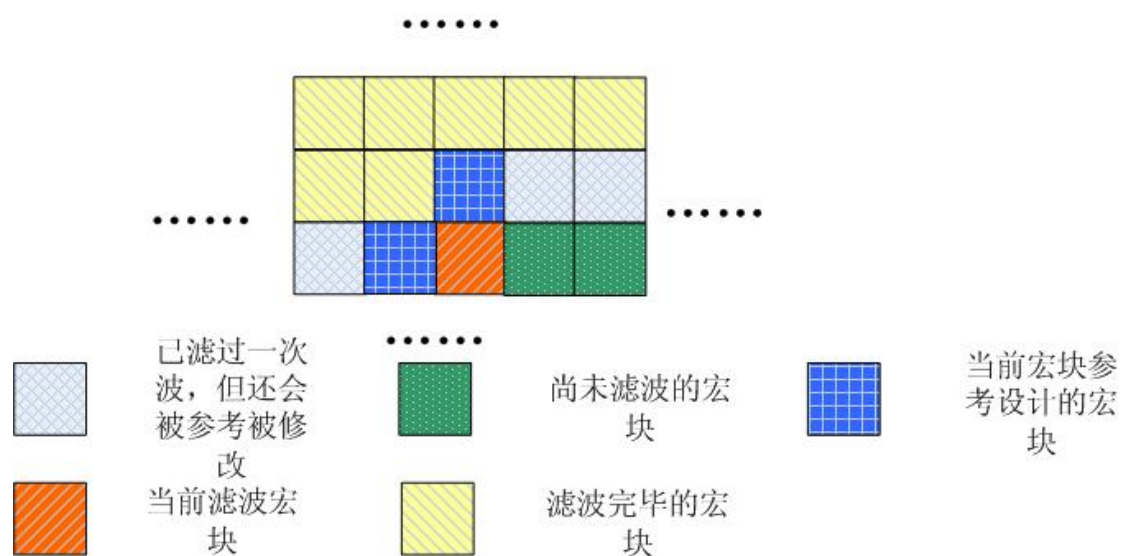


图 28 以滤边为单位流水处理一个宏块的结构图

● 输出部分设计

输出部分需要写总线和参考帧缓存，并且分为按宏块写和按图片写两种情况，综合起来有四种组合：按宏块写总线和参考帧缓存，按图片写总线和参考帧缓存，按宏块写总线按图片写参考帧缓存，按图片写总线按宏块写参考帧缓存。

过滤一个宏块的数据，会涉及并有可能修改到它上面和左面的两个宏块的数据。所以只有当前宏块上面的宏块的左面那个宏块之前的所有宏块是滤波完毕的，如图 29。当滤波进行到第二行第二个宏块的时候，可以输出第一行第一个宏块了，从此开始每滤波完毕一个宏块，就可以输出它左上角的宏块。这就是按宏块输出。



如果是按图片输出，即整幅图片滤波完毕在整体输出。

3.6.3 去块滤波部分的理论分析

● 去块滤波可配置参数

表格 6 去块滤波可配置参数

参数名称	参数含义
INPUTWIDTH	输入带宽
TWOPICBUFFER	存储整幅图片数据的缓存数量
OUTPUTWIDTH	输出带宽
OUTARCH	输出模式
LC	亮度和色度滤波的连接方式
MBCYCLE	滤波一个宏块的周期数

表格 6 显示的是去块滤波中所有的可配置参数。其中 INPUTWIDTH、TWOPICBUFFER 是输入部分的调节参数；OUTPUTWIDTH、OUTARCH 是输出部分的调节参数；LC、MBCYCLE 是滤波计算部分的调节参数。

INPUTWIDTH 表示输入带宽是几个 size(FIFO) 大小。

TWOPICBUFFER 为 0 表示有一个存储整幅图片数据的缓存，1 表示有两个这样的缓存。

OUTPUTWIDTH 表示以图片为输出单位的输出方输出带宽是几个 size (FIFO) 大小。

OUTARCH 为 0 表示总线 and 参考帧缓存以宏块为单位输出；为 1 表示总线 and 参考帧缓存以整幅图象为单位输出；为 2 表示以宏块写总线以整幅图象写参考帧缓存；为 3 表示以整幅图象写总线以宏块写参考帧缓存。

LC 为 0 表示串行连接两度滤波和色度滤波；为 1 表示以宏块为单位流水；2 表示以边为单位流水。

MBCYCLE 表示滤 1 个宏块需要的周期数，至少是 8，并且是 8 的倍数。

任何确定的一组参数，都确定了一个体系结构。这些参数在程序中都有所体现，所以整个模型是可配置的。

● 周期计算

由于总线上挂了比较多的设备，所以等待总线的周期难于预知。可以计算出来的周期数是去掉等待总线周期数的周期。

总周期数（无等总线）= 输入周期 + 滤波周期 + 输出周期

$$\text{输入周期} = \frac{\text{MAXMBNUM}}{\text{INPUTWIDTH}} \times (1 - \text{TWOPICBUFFER}) + \text{TWOPICBUFFER} \quad \text{公式 5.5-0-1}$$

$$\text{滤波周期} = \begin{cases} \text{MBCYCLE} \times 2 \times \text{MAXMBNUM} & \text{LC}=0 \\ \text{MBCYCLE} \times (\text{MAXMBNUM} + 1) & \text{LC}=1 \\ \text{MBCYCLE} \times \text{MAXMBNUM} + \frac{\text{MBCYCLE}}{8}, \text{LC} = 2 \end{cases} \quad \text{公式 5.5-0-2}$$

$$\text{输出周期} = \begin{cases} \text{size(MB)} \times \text{MAXMBNUM} / \text{FILTER_CYCLE_TO_CORE} & \text{OUTARCH}=0 \\ \text{size(MB)} \times \text{MAXMBNUM} / \text{FILTER_CYCLE_TO_CORE} - \text{滤波周期} - \text{输入周期} & \text{OUTARCH}=1 \\ \frac{\text{MAXMBNUM}}{\text{OUTPUTWIDTH}} + \text{size(MB)} / \text{FILTER_CYCLE_TO_CORE} \times (\text{MAXMBNUM} - \text{PicWidthInMbs} - 2) & \text{OUTARCH} = 2 \\ (\text{MAXMBNUM} - \text{PicWidthInMbs} - 2) + \text{size(MB)} \times \text{MAXMBNUM} / \text{FILTER_CYCLE_TO_CORE} - \text{滤波周期} - \text{输入周期} & \text{OUTARCH}=3 \end{cases}$$

公式 5.5-0-3

$$\text{size(MB)} = 16 \times 16 \times 8 + 8 \times 8 \times 4 \times 2 = 2560 \text{ (bits)} = 80 \text{ (words)}$$

写总线一个周期写一个字的数据。

● 缓存计算

$$\text{缓存大小} = \text{size(FIFO)} \times \text{MAXMBNUM} \times (1 + \text{TWOPICBUFFER}) + \begin{cases} \text{size(bS)} \times (\text{LC} + 1) & \text{LC} = 0, 1 \\ \text{size(bS)} \times \frac{2}{8}, & \text{LC} = 2 \end{cases}$$

公式 0.5-0-4

$$\text{size(FIFO)} = 2598 \text{ (bits)} \approx 325 \text{ Byte}$$

$$\text{size(bS)} = 3 \times 8 \times 8 = 192 \text{ (bits)} = 24 \text{ Byte}$$

● 带宽计算

$$\text{输入带宽} = \text{INPUTWIDTH} \times \text{size(FIFO)}$$

公式 05-0-5

$$\text{输出带宽} = \begin{cases} \text{size(MB)} + 1 & \text{OUTARCH} = 0, 3 \\ \text{OUTPUTWIDTH} \times \text{size(MB)} + 1 & \text{OUTARCH} = 1, 2 \end{cases} \quad \text{公式 05-0-6}$$

单位是字。

● 测试方案

表格 7 测试方案参数列表

方案编号	INPUTWIDTH	TWOPICBUFFER	OUTPUTWIDTH	OUTARCH	LC	MBCYCLE
1	1	0	1	1	1	8
2	1	1	1	1	1	8
3	2	0	1	1	1	8

4	8	0	1	1	1	8
5	2	0	8	1	1	8
6	2	0	1	0	1	8
7	2	0	1	2	1	8
8	2	0	2	2	1	8
9	2	0	8	2	1	8
10	2	0	1	3	1	8
11	2	0	2	3	1	8
12	2	0	8	3	1	8
13	2	0	1	0	0	8
14	2	0	1	0	0	16
15	2	0	1	0	1	16
16	2	0	1	0	2	8
17	2	0	1	0	2	16

其中方案 1-4 是针对于 INPUTWIDTH & TWOPICBUFFER。

方案 5-12 是针对于 OUTPUTWIDTH & OUTARCH。

方案 13-17 是针对于 LC& MBCYCLE。

● 理论分析最佳方案

利用公式 5.5-1~5.5-6 计算各个参数如表格 8

表格 8 理论分析结果列

方案编号	周期（无等总线）	输入带宽（字）	输出带宽（字）	总带宽（字）	缓存（B）
1	25920	81	81	162	526143
2	25920	81	81	162	1052238
3	25920	162	81	243	526143
4	25920	648	81	729	526143
5	25920	162	641	803	526143
6	39698	162	81	243	526143
7	40582	162	81	243	526143
8	39772	162	161	323	526143
9	39165	162	641	803	526143
10	27494	162	81	243	526143
11	27494	162	81	243	526143
12	27494	162	81	243	526143

13	52650	162	81	243	526119
14	78570	162	81	243	526119
15	52666	162	81	243	526143
16	39691	162	81	243	526098
17	52652	162	81	243	526098

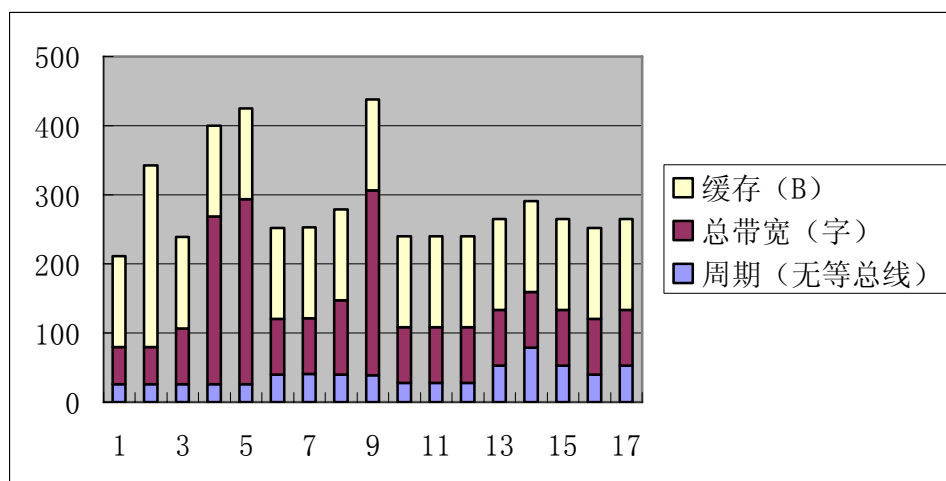


图 30 各方案理论分析结果

由方案 1-4 的理论计算结果可以看出，输入部分的周期数和带宽不是整体部件的瓶颈，输入部分结构的调整对整体基本没有作用。

由方案 5-12 的理论计算结果可以看出，输出部分的带宽对于性能影响是关键，输出部分的结构也有一些作用关键，降低频率的办法是扩大输出带宽。

由方案 13-17 的理论计算结果可以看出，滤波运算部分的周期数是影响总周期的关键因素。由于写总线可以与计算并行，所以两者的周期匹配也很关键。应该在可以接受的前提下尽量减少滤波周期。

我们希望在满足实时显示的前提下，器件频率尽量低、带宽尽量小、缓存尽量小、设计复杂度不高等。综合这些条件，在无法判断等待总线周期的前提下，理论分析的最佳方案是方案 1。

3.6.4 各种综合方案的测试与结果分析

表格 9 测试结果列

方案编号	周期	频率 ^①	总带宽(字)	缓存(B)
1	116326	3.33	162	526143
2	116326	3.33	162	1052238
3	116326	3.33	243	526143
4	116326	3.33	729	526143
5	25857	0.74	803	526143
6	119711	3.42	243	526143
7	117900	3.37	243	526143
8	117090	3.35	323	526143
9	116482	3.33	803	526143
10	118017	3.38	243	526143
11	66339	1.90	243	526143
12	27549	0.79	243	526143
13	133770	3.83	243	526119
14	158582	4.54	243	526119
15	132951	3.80	243	526143
16	119703	3.42	243	526098
17	132665	3.80	243	526098

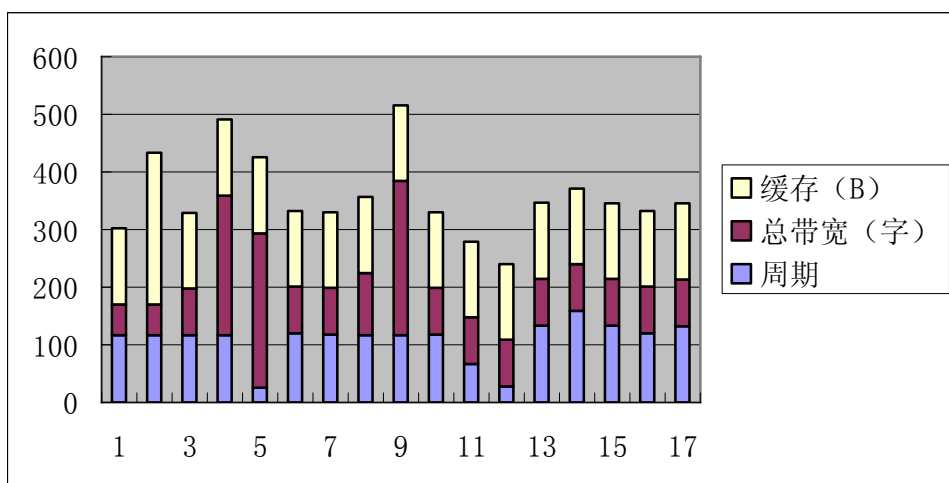


图 31 各方案测试结果

^① H.264 标准要求达到 30 帧/秒的数据输出速度，该频率是根据这个输出速率结合周期数推算得出，下同。

由上述测试结果可以看出，一部分与理论分析是相符的，但是由于等待总线周期的引入，有一些部分发生了改变。

方案 1-4 的测试结果与理论分析一致，但由于增加了等待总线的周期，整体运行周期提高了。由于输入部分的结构设计不是影响性能的瓶颈，所以等待周期数也是相同的。

方案 5-12 的测试结果可以看出，盲目的增加输出带宽不一定会有好的效果。扩大写缓存的带宽没有明显效果。当以整幅图为单位写总线的时候，调节总线的带宽或者每次 burst 写总线的数量可以比较大的影响频率。

方案 13-17 的测试结果与理论分析也是一致的，去块滤波运算与输出部分的设计要相互协调。

具体来讲，第 5 种方案频率最低，但是需要的带宽比较多；第 12 种方案整体比较均衡，但是每次写总线占用的时间过长，使其它器件阻塞时间过长；多种方面相平衡仍然是方案 1 比较好。方案 1 的具体设计如表格 10。

表格 10 最佳方案参数表

参数名称	数值	参数含义
INPUTWIDTH	1	输入带宽为 size(FIFO)
TWOPICBUFFER	0	存储整幅图片数据的缓存数量为 1 个
OUTPUTWIDTH	1	输出带宽为 size(FIFO)
OUTARCH	1	输出模式为以整幅图象写总线和参考帧缓存
LC	1	亮度和色度滤波的连接方式是串行
MBCYCLE	8	滤波一个宏块的周期数为 8

第 4 章 基于 Cell 的多视点视频编解码并行算法设计

4.1 多视点视频编解码算法介绍与分析

随着网络和计算技术的发展,下一代多媒体应用会更加具有交互性和真实感。多视点视频技术是达到新的媒体要求的关键技术之一。多视点视频技术是从不同的角度用不同的照相机拍摄图像,然后对这些图像编码、传输、解码、综合成 3 D 影像的技术。它解决了表示、交互、编解码和传输 3 D 交互视频影像的问题。多视点视频为自由视点视频[56]、三维显示[55, 57]、高性能图像[58]等应用的进一步扩展提供了可能。图 32 展示的是三种主要的照相机摆放位置。

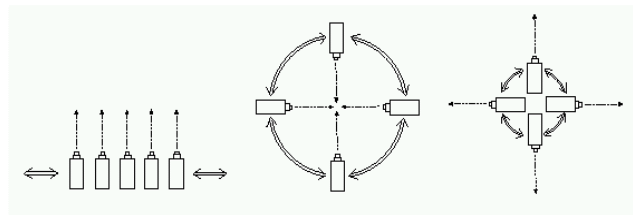


图 32 多视点视频照相机的摆放方案

多视点视频编码标准是正在建立的一个标准。MPEG 组织最近发布了一个有关多视点视频编解码的提案征集 [59],并且说明最终的多视点视频编解码标准会是 H. 264 的一个扩展,在 2008 年正式推出。

对于 3D 视频,高性能的多视点视频编码结构是非常重要的。在编码端,需要利用空间和时间上的冗余有效的降低比特率,以利于传输和存储。在解码端,多视点视频编解码支持时间、位置和图片上空间区域的随机访问。[60]

传统的编码技术,例如运动补偿和视差补偿对于两眼立体视频和多视点视频并不是很有效。在多视点视频编码算法中,我们使用全局运动补偿的方法来改进运动补偿,这种全局运动补偿把运动补偿和视差补偿结合在一起。运动补偿是利用的时间冗余性,视差补偿是利用空间冗余性。多视点视频编码继承了 H. 264/AVC 当中对于一帧的处理,只是重新设计了个帧间预测帧之间的关系。[61, 62]

根据不同的运动估计方法,所有的帧被分为三种类型:I 帧、P 帧和 B 帧[55],其含义与 H. 264 当中的相同。我们以 8 视点视频为例,8 个视点形成 8 个视频流。8 个视点的视频编码算法的依赖图在多视点视频编解码的草案 2.0 (Joint Draft 2.0 on Multi-view Video Coding [63]) 和多视点视频模型 3.0 (Joint Multi-view Video Model (JMVM) 3.0[64]) 当中有描述,如图 33

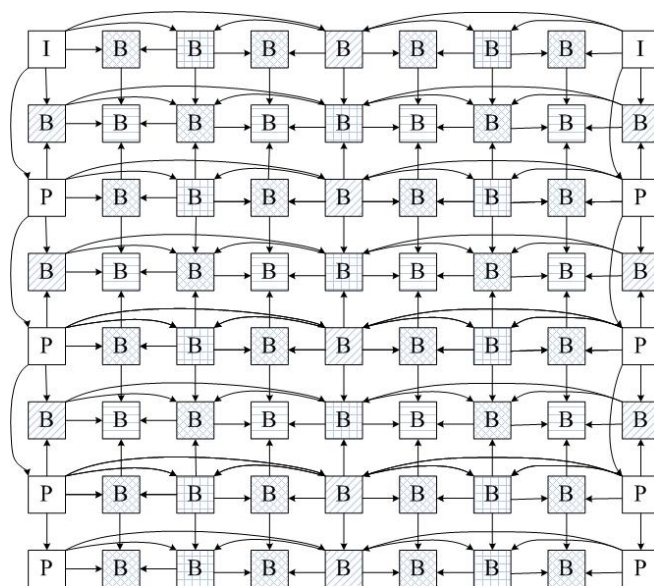






图 33 八视点视频编码当中一个 GOP 的依赖关系图

当涉及到这些标准的应用不断增加的时候,实时处理变成了显而易见的关键点。多视点视频编解码的运算复杂度随着数据的增加而线性增加。单核处理器很难支持实时多视点视频编解码的处理。

根据图 34 我们可以了解多视点视频编解码的核心算法。图 34 的每一行代表从一个角度获得的视频流,每一列代表一个时间点,每一个方块代表对于一帧的处理。方块上的字母标示该帧为 I 帧、P 帧还是 B 帧。其中 B 帧分为三类,一类是参考两帧的 B2,一类是参考三帧的 B3,还有一类是参考四帧的 B4,这些参考关系由图 34 中的箭头表示。对于每一帧的处理与 H. 264/AVC 对于 I、P、B 帧的处理相同,整个多视点视频编解码标准的草案是基于 H. 264 的标准基础上的补充。

所有的帧可以分为六个不同的优先级,不同方块之间的箭头方向表示了数据之间的依赖关系。箭头指向的方块依赖于箭头发出的方块的数据。图 34 展示

的是一整个 GOP 和下一个 GOP 的第一列的数据，其中两个 I 帧包含着最高的优先级，需要在 GOP 当中被最先处理，其他所有的帧都依赖于这两个帧的数据。8 个 P 帧拥有第二高的优先级，剩下的 B 帧的优先级按照下面的图案顺序依次降低，，，，和。只有等到优先级较高的帧处理完毕之后优先级较低的帧才可以处理，相同优先级的帧数据尽量并行操作。对于每一帧的操作继承 H.264 标准的操作。一个 GOP 共有六个优先级，从 P1 到 P6 是从高到低，各个优先级所包含的帧的类型如表所示。如表，每一种帧的数量用 N_I 、 N_P 、 N_{B2} 、 N_{B3} 、 N_{B4} 表示，每一种帧处理的时间用 T_I 、 T_P 、 T_{B2} 、 T_{B3} 、 T_{B4} 表示。

表格 11 参数表

帧 类 型	每个优先级包含的帧的数 量						数 量	处 理 一 帧 的 时 间
	P1	P2	P3	P4	P5	P6		
I	2	0	0	0	0	0	$N_I=2$	T_I
P	0	8	0	0	0	0	$N_P=8$	T_P
B2	0	0	10	8	16	0	$N_{B2}=34$	T_{B2}
B3	0	0	1	2	4	0	$N_{B3}=7$	T_{B3}
B4	0	0	0	3	6	12	$N_{B4}=21$	T_{B4}
总量 ($T_i, i=1,2,\dots,6$)	2	8	11	13	26	12	72	

8 视点视频的数据量比单视点视频数据量的八倍要大，如果让 8 视点视频的编解码达到实时的要求，那么每秒钟要处理的帧数将会从单视点视频的 30 帧提高到 240 帧[65]。多视点视频编解码亟待解决的最重要的问题是单核处理器很难完全支持实时处理的很高的运算复杂度，这是由于猛增的数据量和运算复杂度的需求造成的。因此，将多视点视频编解码的算法移植到片上多处理器上，

使之并行化，是解决这个问题的途径之一。

4.2 Cell 处理器

图 34 表示的是片上多处理器的一种模型，我们的解决方案说明，将多视点视频编解码算法映射到这种多核处理器模型的方法是将多视点视频编解码算法并行化的一种可行方案。所有的协处理器（SPE）是一样的，负责基础的运算和任务。每一个协处理器单元都包含一个本地存储（LS），所有在 SPU 上运行的代码和数据都要存储在 LS 当中。CPE 是一个中心控制微处理器，控制着数据流并且调度任务的顺序。两个 SPE 之间独立的传输数据交换信息是不可以的，所有的这些交互必须由 CPE 控制。所有的数据和代码通过总线传递，总线必须足够宽，传输足够快。

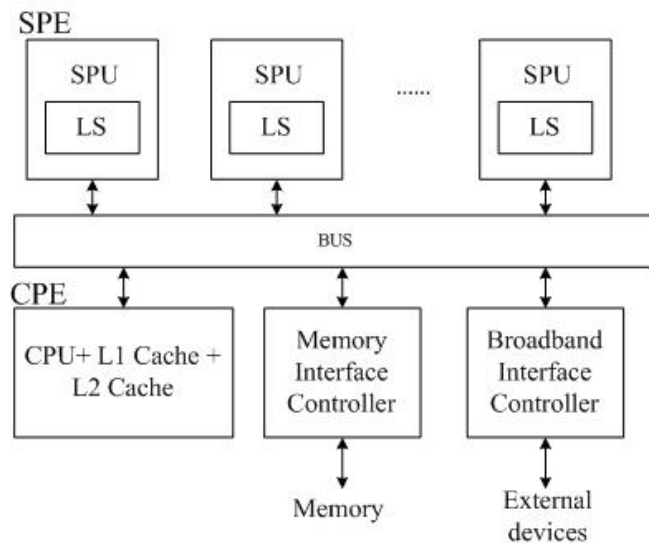


图 34 多核处理器的目标模型

Cell 处理器是 IBM、Sony 和东芝共同研发的，十分适合于上面这个模型的多核处理器实例。IBM、索尼和东芝三家公司于 2005 年 8 月正式推出了 Cell 宽带引擎（Cell Broadband Engine，简称 Cell BE）。Cell BE 处理器[66]是由 9 个微处理器构成的芯片，是非对称体系结构。核心微处理器是标准的 64 位通用

处理器，有一级缓存和二级缓存，基于 PowerPC 技术的微处理器，称为 Power 处理元素（Power Processing Element, PPE）。另外 8 个微处理器与此不同，称为协同处理元素（Synergistic Processing Elements, SPE），它比普通的 CPU 强很多，针对媒体处理而设计，有自己的更加精简的类 RISC 指令集，一条 32 位的指令有 2~7 周期的延迟（未考虑双精度浮点运算指令）。在 Cell 当中，SPE 负责具体的运算，PPE 负责调度和分配。每个 SPE 都包含基于 128 位矢量的高性能计算协同处理单元（Synergistic Processing Unit, SPU）和本地存储单元（Local Storage, LS）。

Cell 处理器相对于传统的同时钟频率下的单核处理器加速效果很明显。SPE 向量运算可以比普通的单精度浮点运算快 4 倍。8 个 SPE 同时工作可以提高 8 倍，这样理论上总共比典型的单核处理器快 32 倍。传输数据的时间和计算时间可以重叠一部分。当然，由于内存竞争，线程的互相依赖以及负载的不均衡，实际情况可能不可以完全达到峰值性能。

PPE 和 SPE 都是顺序（in-order）处理器。LS 的延迟是固定的 7 个周期，顺序的指令执行比较简单，编译器可以预测内存的访问，由此可以比较好的调度指令的执行顺序。顺序的 PPE 两级缓存的结构使得它的性能比乱序（out-order）的标准处理器逊色些，为了减少这种性能损失，PPE 引入了双线程[39]，类似于 Intel 的超线程技术[38]。

每个 SPE 都有的 LS 是一个高速的 256KB 缓存，取代了高速缓存，所有 SPU 要执行的代码和数据都存放在 LS 当中，SPE 不可以直接访问内存。DMA 是用来在内存和 LS 之间，两个 SPE 间或者 PPE 和 SPE 间传递数据的，它可以支持每个 SPE 16 个并发请求。

与普通的 CPU 相比，每个 SPU 都有 128 个 128 位 SIMD 寄存器。绝大多数的 SPU 指令是 SIMD 指令，比 SSE/SSE2/SSE3 当中的指令更加灵活[37]，例如三操作数乘加指令在 3.2GHz SPE 上理论峰值可以达到 25.6GFlops（单精度）。每个

SPE 有两条流水线,理论上可以达到每秒 64 亿次 SIMD 指令。SPE 没有分支预测,取而代之的是一条分支提示指令。

元素连接总线 (Element Interconnect Bus, EIB) 负责 PPE, SPE 和内存控制器之间的交流,速度可达每周期 96 字节。系统内存带宽总计 25.6GB/s, SPE 之间的带宽在 3.2GHz 的 Cell 上超过 300GB/s。Cell 的体系结构图见图 9。

Cell BE 处理器有一些重要的特征需要编程者注意。SPU 很适合单精度浮点运算,但是双精度的运算相对比较慢。SPU 缺少分支预测的硬件支持,所以编程的时候要尽量避免或者隐藏分支语句。SPU 支持向量运算,所以编程的时候要尽量多地使用。SPE 的 DMA 可以独立于运算单元 SPU,所以数据传输和计算的时间可以重叠。信号 (Signal) 机制和 mailbox 寄存器可以在 SPE 和 PPE 之间传递简短的信息,在同步的时候十分有用。

Cell 处理器相对于传统的同时钟频率下的处理器加速效果很明显。SPE 向量运算可以比普通的单精度浮点运算快 4 倍。8 个 SPE 同时工作可以提高 8 倍,这样理论上总共比典型的单核处理器快 32 倍。传输数据的时间和计算时间可以重叠一部分。当然,由于内存竞争,线程的互相依赖以及负载的不均衡,实际情况可能不可以完全达到峰值性能。

SPE 简单的设计 (没有高速缓存,没有分支预测,顺序执行) 使得他可以达到很高的频率, Sony 的 PlayStation3 的 Cell 时钟频率是 3.2GHz, 其实 Cell 可以达到 4GHz 以上。[67]

4.3 基于多核处理器的并行化多视点视频编解码算法

采用多视点视频并行处理的方法,片上多处理器可以解决运算量大的问题。因为设计复杂度和功耗问题,但和处理器的性能发展放缓。下一代体系结构是片上多处理器的时代。应用软件映射到这种系统上去,可以很容易的实现大量的并行编程。也只有在程序的并行性被很好的开发出来之后,多核处理器的优势才

能显现出来，获得成功。通过上面对于多视点视频编解码算法的分析，多视点视频编解码算法是非常适合于基于多核处理器的并行化的。无论是数据并行还是任务并行都是可行的。如果开发多核处理器上的并行性就是我们接下来要关注的问题。

在视频编码当中，帧是处理的基本单元之一，也是并行粒度的一种选择，对于每一帧的处理过程可以分配到一个 SPU 上完成。PPE 根据优先级控制任务分配。这样九个核的处理器并行运行起来。我们根据最优化理论及方法[65]得到了一个任务分配策略。根据最优化理论我们可以到了一种最优的解，即在满足一些条件的前提下，找到最大值（或最小值）。

假设参考每一帧的运算是相近的，那么参考两帧的 B 帧的运算量是参考一帧的 P 帧的运算量的两倍还多一些，参考三帧的 B 帧是 P 帧的三倍还多，以此类推，我们得到 $T_1 < T_P < T_{B2} < T_{B3} < T_{B4}$ 的时间排序关系。

假设 T 是完成整个 GOP 编码工作的总时间，那么我们的目标就是让 T 尽量的小。给八个 SPE 编号为 1st, 2nd, ..., 到 8th。五种帧分别为 I 帧，P 帧，B2 帧，B3 帧，和 B4 帧。所有的这些帧被分到 6 个不同的优先级当中。 N_{iCP_k} ($i=1, 2, \dots, 8$; $C=I, P, B2, B3, B4$; $P=1, 2, \dots, 6$) 代表帧的数目，右下角的字母标示帧的种类，C 表示五种帧的某一类，P 代表所在的优先级，i 表示分配到的 SPE 的编号。例如，具有第三级优先级的 B2 帧被分配到第 3rdSPE 的数量是 N_{3B2P3} 。根据最优化理论及方法，我们可以得到下面的式子：

我们要在下面的条件的限制下求得最小的 T：

$$T = \sum_{k=1}^6 \max \left\{ \sum_{C=I, P, B2, B3, B4} N_{iCP_k} \times T_C, i=1, 2, \dots, 8 \right\} \quad (1)$$

$$T \geq \sum_{k=1}^6 \sum_{C=I, P, B2, B3, B4} N_{iCP_k} \times T_C, i=1, 2, \dots, 8 \quad (2)$$

$$\sum_{i=1}^8 \sum_{k=1}^6 N_{iCP_k} = N_C, C=I, P, B2, B3, B4 \quad (3)$$

$$\sum_{i=1}^8 \sum_{C=I, P, B2, B3, B4} N_{iCP_k} = T_k, k=1, 2, \dots, 6 \quad (4)$$

$$0 < T_I < T_P < T_{B2} < T_{B3} < T_{B4}, \text{ 所有参数大于等于 } 0 \quad (5)$$

整合分析上述限制条件方程式之后，我们可以得出 $T = T_I + T_P + 4T_{B2} + 2T_{B3} + 4T_{B4}$ 。然后事实上一个 GOP 平均是 64 帧，最右边的一列会被前后两个 GOP 重复利用，所以平均下来，第一优先级有一个 I 帧，第二优先级有 4 个 P 帧 in P2，第三优先级有 7 个 B2 帧和一个 B3 帧。因此一个 GOP 被并行编码的执行时间 T_{Parallel} 是 $T_I + T_P + 3T_{B2} + 2T_{B3} + 4T_{B4}$ 。对于一个 GOP 的 64 帧的编码工作在 8 个 SPE 的工作安排见图 35。

图 35 中 y 轴方向是 1st 到 8th 这 8 个 SPE，x 轴方向是时间。每一行表示给每个 SPE 分配的任务，每条长竖线将两个优先级内的操作分开。整幅图 37 表示的是将一个 GOP 的 $8 \times 8 = 64$ 帧图像分配到了 8 个 SPE 上编码，所有的任务就按照这个顺序从优先级高到优先级低的分配执行。从图 35 中所有的 N_{iCP_k} ($i=1, 2, \dots, 8$; $C=I, P, B2, B3, B4$; $k=1, 2, \dots, 6$) 都可以查出来，例如 $N_{3B2P5}=2$, $N_{5B3P5}=0$ 。每个阴影条代表处理一帧的时间长短，整个图 35 中连接在一起的最长的阴影条的和就是 T 。

如果是按照传统的串行算法执行，那么处理一个 GOP 的时间是 T_{serial} 等于 $T_I + 4T_P + 31T_{B2} + 7T_{B3} + 21T_{B4}$ 这是 T_{Parallel} 8 倍还多，所以相对于串行操作的加速比为 8。

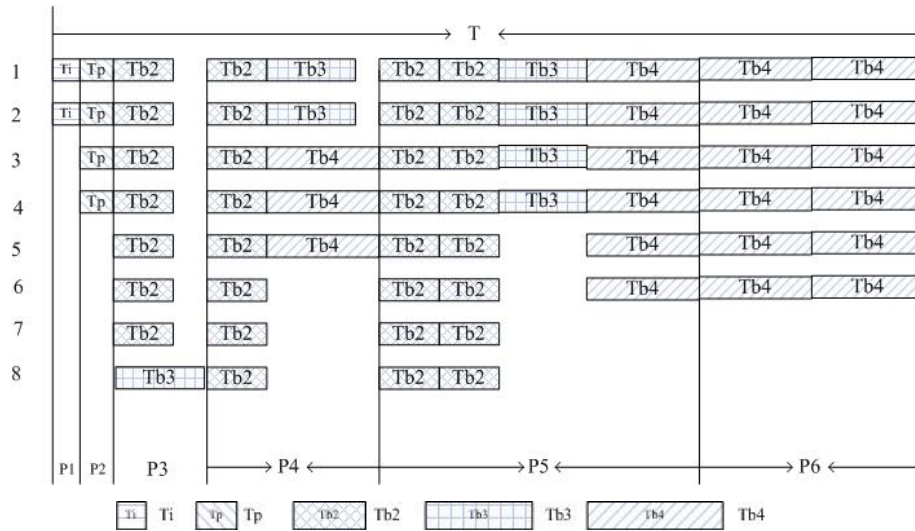


图 35 优化任务调度图

4.4 结论

由于多视点视频编解码算法良好的可并行性，利用片上多处理器可以很好的支持多视点视频编解码算法的高性能需求。根据最优化理论和方法得到的并行化任务分配办法比传统串行方法的提高了 8 倍的速度。所以如果一种算法也具有多视点视频编解码算法的这种数据或者任务的可并行性，那么采用类似的方法片上多处理器的方法也可以加速整个过程。所以多视点视频编解码算法与片上多处理器的结合是很好的一种解决方案。

当然，结果还可以进一步改进。如果考虑到数据的 I/O 和存储，更多的开发数据的可并行性，都以使得性能进一步提升。例如，把要并行处理的数据分别放在不同存储单元当中，以避免读取冲突。我们还可以改变帧的处理速度，以使得 SPE 单元被利用的更加充分。

第 5 章 基于 Cell 处理器的光线跟踪设计与实现

5.1 光线跟踪算法分析

计算机图形学是当今计算机学科中最热门的话题之一。它改变了人们对计算机显示器终端的认识,使得人们在图像处理,工程设计,生物医药,游戏娱乐等方面取得巨大的变革和创新,同时也推动了计算机产业进入千家万户。光线跟踪算法则是计算机图形学中渲染 3D 空间所使用的各种方法中,最具有真实感的一种。它的优势在于完全符合粒子光学的原理,在不涉及衍射等复杂光学现象的时候可以几乎达到 100%的真实。但是它的劣势则在于运算量大处理速度慢。使用一个较为真实的光线跟踪算法渲染一幅单个模型的 1024x768 的图片时,在当今的主流计算机上要花去 10 几分钟,当模型变得很多时,则需要的时间成倍增长;而如果使用比较快速的光线跟踪算法,得出的结果有不尽人意。所以设计精细的光线跟踪算法的运行结果,通常是一幅精美的图片,而不会是一段实时交互的展示。

针对这个问题,本节的主要内容是将光线跟踪算法,在 IBM Cell 多核处理器上实现,利用光线跟踪算法的可并行性,提高处理速度。这无论对于提高光线跟踪性能,还是在测试和实践 IBM Cell 处理器的性能方面,都具有一定的价值。

5.1.1 光线跟踪算法的特点

光线跟踪算法的原理是光路的可逆性。从光源射出的光线,在遇到物体的时候,会发生漫反射,镜面反射和折射三种现象。光源向不同的方向射出的光线,最后汇聚到视角的那些,就是观察者能够看到的光线。所以将从眼睛出发的任意向量,看作一束光线,然后跟踪这束光线的路径,在遇到物体时作出相应的漫反射,镜面反射和折射,即模拟了物体成像的过程,如图 36。对于漫反射的这部分光线,有很多经典的模型可以利用,对于镜面反射和折射的这部分光线,需要使用递归算法,从射线和物体的交点出发继续进行跟踪操作。所以

说，对于一束光线的计算，是一个递归的过程。而递归的最大深度有多深，决定了画面的真实感。

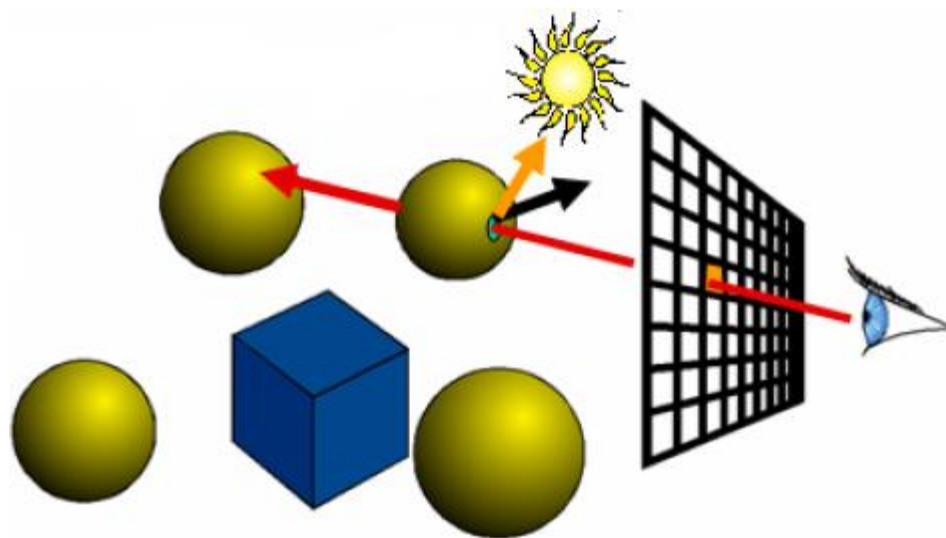


图 36 光线跟踪原理图

上面的这个递归算法，运行完一次之后，我们可以得到屏幕上一个像素的颜色值。要将整个屏幕渲染出来，则需要对该算法进行多次调用。虽然相邻两个像素的颜色值感觉上很相似，光的路径也不会相差很多，但是它们计算的中间步骤相互独立。因为除非你重新跟踪一条射线，否则你无法预测它和它相邻的射线相比光线路径的不同。所以说，对于从视角出发的所有射线的光线跟踪运算之间，是完全独立的。这说明光线跟踪有很强的可并行性。

5.2 在 Cell 上的光线跟踪算法设计

5.2.1 在 Cell 上实现光线跟踪的特殊点

在 Cell 处理器上实现光线跟踪算法有其特别之处，这些不同点需要在设计编程模型之前注意到：

顺序执行：

SPE 是顺序执行指令的，这就意味着代码的依赖性或者错误的分支预测都会导致流水线暂停，这相比于乱序执行的处理器影响性能。SPU 缺少分支预测的硬件支持，所以编程的时候要尽量避免或者隐藏分支语句。虽然编译器可以

自动的进行调度，减小指令间的依赖网络，但适时地调整算法有助于避免流水线暂停的发生。

- **SIMD 指令：**

SPE 的指令集中绝大部分都是 SIMD 指令，一条指令同时作用到多个数据上（根据数据位宽的不同可以同时处理 2 到 16 个数据不等）。这就要求数据之间足够高的独立性，否则依赖关系会导致并行性下降。这些都需要在算法设计上多加考虑。

- **浮点运算**

SPE 很适合单精度浮点运算，有比较强的浮点运算单元作支持，但是双精度的运算相对比较慢，应该尽量避免。

- **内存访问：**

SPE 有非常清晰的三层存储结构：128 乘 128 的寄存器，256KB 的 LS 和主存。LS 与高速缓存不同，所有对于主存的访问都需要由 DMA 控制。内存带宽 25.6GB/s，但是每次内存访问的延迟还是需要几百个时钟周期才能完成。因了隐藏该延迟，DMA 支持异步传输。这种设计对于大块数据处理的流媒体应用有很好的效果，但是对于内存访问不规律的光线跟踪算法还是一种挑战。

信号（Signal）机制和 mailbox 寄存器可以在 SPE 和 PPE 之间传递简短的信息，在同步的时候十分有用。

- **多个 SPE 上的并行执行：**

每个 Cell 有 8 个 SPE，一个双核 Cell Blade 系统有 16 个 SPE。有不同的映射到并行体系结构的方法，可以使得性能大幅提高。由于 Cell 体系结构很好的可扩展性，所以需要并行算法也具有很好的可扩展性来支持。

5.2.2 编程模型

将光线跟踪算法映射到 Cell 处理器上主要有两种思路。异构的方法是任务并行，每个 SPE 运行不同的操作步骤，流水进行，运算结果从一个 SPE 传递到另一个；同构的方法是数据并行，每个 SPE 运行相同的程序，但是操作的数据是不同的数据。传统的多核处理器常用后一种[68]，但是传统的流媒体体系结构倾向于前一种[68-71]。

光线跟踪可以分为如下几步：生成初始光线，光线穿越空间索引结构，光线与几何原型相交，渲染相应的相交点，这其中还包括生成第二级光线以及一些递归操作。

一种将光线跟踪算法映射到 Cell 处理器上的方法是采用异构的方法，将上面说的每一步骤放在一个或多个 SPE 上运行，处理结果在各个 SPE 之间传递。例如，一个 SPE 生成了初始的光线，然后把它传递给一个或多个 SPE 进行光线穿越空间检索结构的运算，然后再依次到达与几何原型相交和渲染等步骤。根据 Cell 流媒体性质的体系结构，以及 SPE 间较高的数据传输带宽（可达 300GB/s），这种方法是可行的。可以异构传输的 DMA 也可以将传输数据与计算的时间重合。类似的设计在之前已有验证[69, 72]。

但是流处理方法只有在数据流相对固定，任务独立性强的时候才可以发挥出较好的效果[73]。而对于光线跟踪算法，它的数据流比较复杂，规律性不强。光线穿越空间搜索结构之后，传递给计算相交的模块，但与几何原型相交的方向不唯一；阴影计算不仅计算阴影，同时还可能触发下一级光线跟踪。由此可知，光线跟踪的数据依赖关系复杂，等待数据很有可能引起流水线暂停，同时，这种方法比较难作到各 SPE 之间的负载平衡，同时一旦一个 SPE 成为瓶颈，其他的 SPE 都需要等待。

当然，上面讲到的 SPE 空等的问题可以用建立缓冲区的方法来解决，设置合适的缓冲区大小，保证缓冲区不空，确保流水线不暂停。SPE 上运行的程序也可以根据程序的运行情况作切换，使得最需要运算的模块分配到较多的 SPE。但这些都意味着实现的复杂度提高很多，需要考虑更多的同步以及负载均衡的问题。同时访问内存的带宽比 SPE 间的带宽小很多，只有 25GB/s，虽然看起来也足够，但是限制了性能进一步提升的余地。

综上考虑，我们选择了同构的方法，利用共享内存的方法，令每个 SPE 独立完成光线跟踪的运算，处理不同像素点的数据。这样增加了各个 SPE 之间的独立性，减少数据交互，避免了中间数据的产生，提高 SPE 的利用率。

这种方法也带来了一些不好的地方，例如 LS 大小的限制使得复杂的场景不能一次性存入，需要采用适当的方法访问缓存场景数据。

5.2.3 空间检索结构

在确定了变成模型之后，我们要选择合适的空间检索结构（Spatial Index Structure, SIS）。高效的光线跟踪需要高效的数据结构，典型的 SIS 有包围盒算法（Bounding Volume Hierarchies, BVH）[74]，网格法（Grids）[75]，KD 树（kd-tree）[76]。

Kd 树是最初产生的方法，它的一些相交处理办法还可以应用到网格法和 BVH 上。由于网格法和 BVH 可以处理动态的场景，我们关注于后两个。这两种方法中，网格法更加通用，它的更加规律的结构适合于流处理结构。而对于复杂的场景 BVH 显得比网格法更加快速，对于二级光线更加鲁棒。因此我们选择了 BVH 的方法。

与 KD 树相比，针对 Cell 体系结构 BVH 有更多优点。BVH 的内存访问比较少，可以得到较高的运算与内存访问比，因为它比 Kd 树的节点少，每个节点的运算量较大。

5.3 实现

我们选择处理 1024 乘 768 的图，要渲染出一整张图，须要对 1024 乘 768 束光线分别作光线跟踪。由于上文的分析，任意两束光线的计算过程，都没有依赖关系，而两次计算所需要的数据完全相同。可以将这 1024x768 束光线分给 8 个 SPE 完成。由于每次递归算法所需要的数据相同，这大大减少了将模型导入到 LS 的数据量，充分发挥 Cell 的结构优势。

● PPE 与 SPE 的分工

核心处理器 PPE 的主要工作是初始化数据。由于这些模型数据是大家共用，所以由 PPE 进行简单整合再交给 SPE，避免每个 SPE 重复处理。首先，向 PPE 中读入相应的数据，进行初始化处理，在内存中存放：视点，屏幕的四个点，物体的数目，每个物体的三角形数，每个三角形类，每个物体的包围盒，灯的数量和灯。然后将相关参数传递给 SPE 的主函数。

当 SPE 接收到这些模型数据后，开始计算不同的像素点值。在 SPE 上运行程序，需要先创建 SPE 上下文，将程序加载到这个上下文中，然后再运行这个程序。

● 本地存储的管理方式

由于每个 SPE 只有 256KB 的 LS，容量有限，无论是代码段，数据段，还是堆栈段，都必须要塞进这 256KB 的空间内。为把整个模型放在 LS 中，所以不得不简化模型，这影响了真实感，同时代码段也相应简化。

在完成 SPE 上的程序时，编译出的二进制代码约为 180KB，而使用 Local Store 剩余的近 80KB 空间来存储 3D 模型，对于不复杂的模型来说是够的。计算完的像素值，每满 2048 字节后使用 DMA 传输至 Cell 的内存，只需要占用 2KB 的空间。所以，LS 的空间被有效的利用起来。

● 包围盒算法 (BVH)

所谓“包围盒算法”处理光线跟踪是指，求交的光线首先跟包围盒进行求交测试，若相交，则光线再与景物求交，否则光线与景物必无交。利用形状简单的包围盒与光线求交的速度较快来提高算法的效率的。它对于复杂的、动态的场景效果显著，有一定的灵活性。

● Phong 模型，

我们采用的像素点光线颜色模型是 Phong 模型。最终的模型可以表示为：

$$C = C_i \times I + K_d \times C_d$$

$$I = I_a K_a + I_p (L \times N) + \frac{I_p (R \times V)}{n - n(R \times V) + R \times V}$$

I_a ：环境光强度； K_a ：环境光反射系数

I_p ：入射光强度； L ：入射光向量

N ：物体法向； R ：反射光向量

V ：视点向量； n ：镜面反射指数

C ：最终颜色； C_i ：物体固有颜色

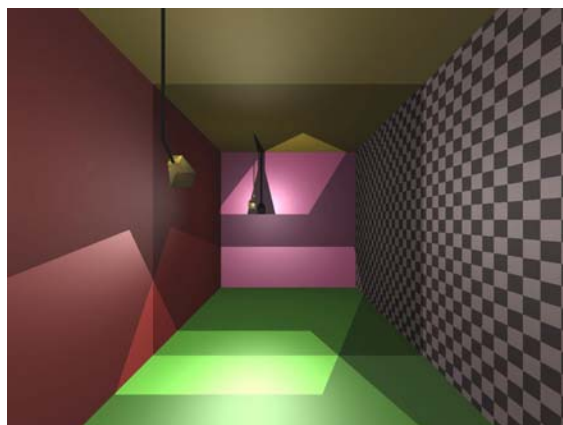


图 37 基础算法效果图

依据模型的公式计算出各个像素点的光强度和颜色。画出的效果图如图 37，包含了 5 面墙，一个镜子，一个灯。

5.4 可扩展性

如第一节中所讲，处理器的发展方向是多核处理器，并且绝大多数的可编程处理器朝着对称更多核的方向发展，这就要求程序的可扩展性好，随着核数的增加，程序可以提供足够多的并行线程。我们的方法无疑具有很好的可扩展性，理论上讲渲染 1024 乘 768 的图，每个像素点的计算都可以作为一个线程在一个核上运行，即产生近 79 万个线程，这个数字在短时间内处理器的核数是无法达到的。

针对 Cell 拥有的 8 个 SPE，我们在代码中设置了 NUM_SPE 变量做接口，设置参与运算的 SPE 的数量为 1、2、4、8，得到不同 SPE 数目与运行时间的关系如表格 12：

表格 12 并行可扩展性

SPE 数目	运行时间	加速比（相对单核）
1	7.401min	1
2	4.120min	1.80
4	2.029min	3.65
8	0.976min	7.58

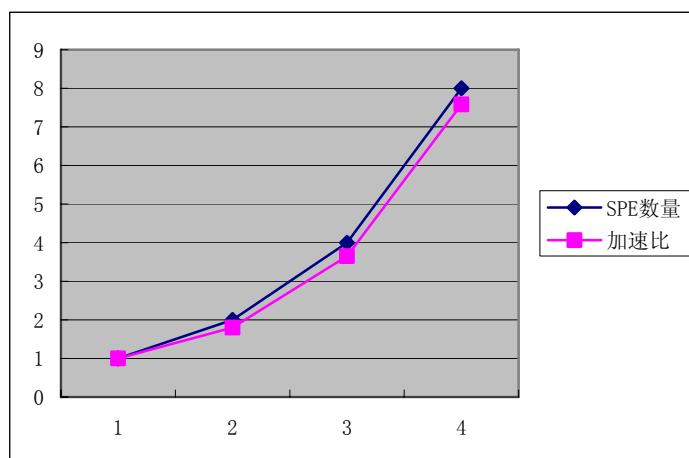


图 38 多核加速效果图

由图 38 可见，我们的并行算法得到了较好的加速效果，同时随着核数的指数级增加，加速比也随着指数级增长。由于 PPE 有一些共用操作不能并行，所以加速比不能做到严格的指数级增加，但随着核数增多，这部分的影响在逐渐减弱。

5.5 与通用处理器比较

通过第 6 节我们知道，Cell 多核的加速效果显著，但是每个核的处理性能又如何呢？为了回答这个问题，我们将同样的程序在 Intel x84 系列单核与多核的处理器上也进行了测试，注意选择了与 Cell 时钟频率同为 2.4GHz 的 Intel 奔腾 4 为实验平台。得到数据如表格 13。

表格 13 Cell 与通用处理器性能比较

处理器	运行时间	加速比
1 个 SPE	7.401min	1
2.4GHz 奔腾 4	7.881min	1.06

有表格 13 的实验数据说明单个 SPE 的处理能力与奔腾 4 处理器的处理能力相当，一个 Cell 相当于 8 个奔腾四的处理能力，可见运算能力之强大。

5.6 双缓冲区设计

由于 LS 有限的空间，使得按照 5.2 节介绍将整个模型存入 LS 的方法限制

了模型的大小。为了解决这一问题，使得 Cell 可以处理复杂场景的光线跟踪问题，我们采用双缓冲的办法来解决这个问题，在原来 LS 已经很满的情况下，再加入一个球物体作为测试例子。具体的办法是将球物体文件的数据分两次导入 SPE 的 LS 中，读入以后分别参与运算，再将两部分叠加的结果送到 PPE，并在屏幕上显示出来。效果图如图 39：

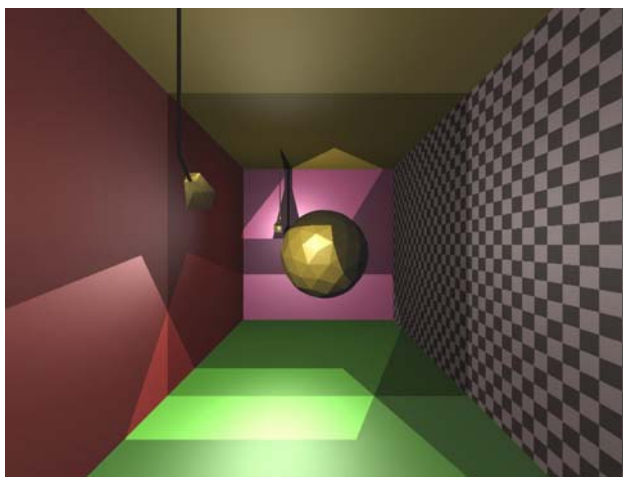


图 39 双缓冲区效果图

我们同样在 1、2、4、8 个 SPE 上运行双缓冲区程序，以检验其对于可扩展性的影响。得到不同 SPE 数目与运行时间的关系如表格 14。

表格 14 双缓冲区运行结果

SPE 数目	运行时间	加速比（相对单核）
1	14.070min	1
2	7.142min	1.97
4	3.663min	3.84
8	1.878min	7.49

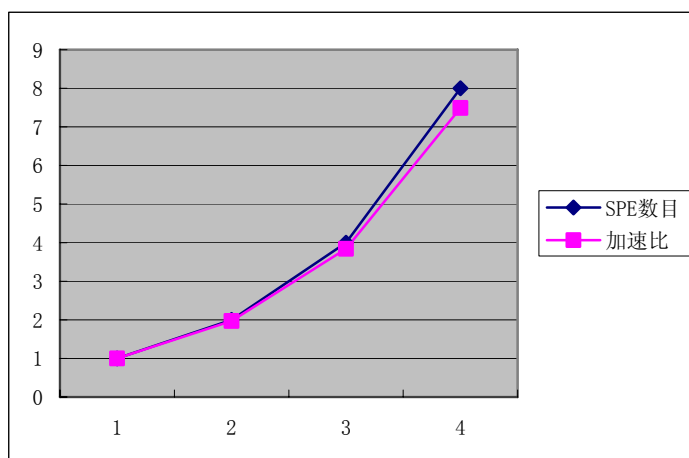


图 40 双缓冲区多核加速比

由统计数据可知，双缓冲区的设计对于程序的可扩展性没有太大影响，反而因为单个 SPE 上运算及数据传输变得复杂了而使得加速比有所提高。

5.7 结论与展望

我们之前的工作是在线程级并行上，算法的可扩展性好，随着参与运算的处理器核数的增加，加速比也随之增长。同时还将 Cell 与通用处理器作对比，得出单个 SPE 的性能与奔腾 4 处理器的性能相当的结论。利用双缓冲区的方法解决 LS 受限的问题，使得算法可以处理场景复杂的问题。

本节的工作还有很多可以改进的地方。在 Cell 上的并行优化分粗粒度和细粒度两个层次，一个是线程级并行，一个是利用 SIMD 技术。所以还可以在细粒度数据并行的角度进行再次优化。

目前的方法运行速度离实时处理还有很大距离。从空间索引数据结构的角度还可以优化性能。目前的光线跟踪程序使用的最简单的算法，在求交上没有做很好的优化，只用了简单的包围盒，没有使用包围树或者立方体阵列之类的包围方式，所以代码运行的时间开销比较大。这部分可以进一步提高运算性能。本节没有涉及任务流水并行的内容，只是完成了像素点之间的数据并行，可以在任务并行，或者任务并行与数据并行混合的方师上进一步尝试。

第6章 结论

随着人们对于多媒体应用的需求不断提高，多媒体应用的功能不断强大，不可避免的使得其运算复杂度与数据吞吐量提高，这样，如何达到对于处理器运算能力与数据传输越来越高的要求是本文主要关注的问题。随着单核处理器性能提高遇到瓶颈，多核处理器不断发展。另一方面媒体处理往往都蕴含着比较好的数据可并行性，所以二者相结合的方法是解决方案。本文以主从多核处理器及 IBM Cell 的异构多核处理器为例，并行化 H.264、多视点视频编解码及光线跟踪等媒体处理应用，得到了很好的加速效果，证明了基于多核处理器的媒体加速方法的可行性及有效性。

参考文献

- [1] Xu, J. *A Case Study in Network-on-Chip Design for Embedded Video*. in *Design, Automation and Test in Europe (DATE 04)*. 2004.
- [2] Moreton, H. 3d graphics hardware: Revolution or evolution. 2005 [cited; Available from: <http://www.graphicshardware.org/previous/www2005/presentations/moretonpresentationgh05.pdf>.
- [3] Devices., A.M. AMD Opteron Processor Model 8 DataSheet. 2003 [cited; Available from: <http://www.amd.com/us-en/Processors>.
- [4] Corp., I. Intel Next Generation Micro Architecture. 2005 [cited; Available from: <http://www.intel.com/technology/computing/ngma/>.
- [5] Machines., I.B. IBM Power5. . 2005 [cited; Available from: <http://www-03.ibm.com/systems/power/>.
- [6] Microsystems., S. UltraSparc-T1. 2006 [cited; Available from: <http://www.sun.com/processors/UltraSPARC-T1/index.xml>.
- [7] JTC1, I.I., Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s-- Part 2: Video. ISO/IEC 11172 (MPEG-1), 1993.
- [8] JTC1, I.-T.a.I.I., Generic coding of moving pictures and associated audio information – Part 2: Video. ITU-T Rec. H.262 – ISO/IEC 13818-2 (MPEG-2), 1994.
- [9] ITU-T, Draft ITU-T Recommendation H.263 (Video Coding for Low Bit RateCommunication). 1995.
- [10] ITU-T, Recommendation H.263+, Draft 12. 1997.
- [11] 钟玉琢、王琪、贺玉文, ed. 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. 2000, 科学出版社.
- [12] W. Wolf, B.O., and T. Lv, Smart Cameras as Embedded Systems. *Computer Graphics*, 2002. 35(9): p. 48-53.
- [13] W. Wolf, T.L., and I.B. Ozer, Design Study for a High-Performance Smart Camera in the MSP-3 Workshop. 2001.
- [14] Yung, N.H.C.K.-K.L., Spatial and temporal data parallelization of the H.261 video coding algorithm. *Circuits and Systems for Video Technology*, IEEE Transactions on, 2001. 11(1): p. 91-104.
- [15] Delp., K.S.a.E. A parallel implementation of an MPEG1 encoder: Faster than real-time! in the SPIE Conference on Digital Video Compression: Algorithms and Technologies. 1995. San Jose, California.
- [16] M. Mattavelli, S.B., D. Mlynek. A parallel multimedia processor for macroblock based compression standards in 1997 International Conference on Image Processing (ICIP'97) 1997.
- [17] Rutten, M.J.v.E., J.T.J. Jaspers, E.G.T. van der Wolf, P. Gangwal, O.P. Timmer, A.

- Pol, E.-J.D. , A heterogeneous multiprocessor architecture for flexible media processing. *Design & Test of Computers*, IEEE, 2002. 19(4): p. 39-50.
- [18] Erik B. van der Tol, E.G.J. Mapping of MPEG-4 decoding on a flexible architecture platform in SPIE Media Processor. 2001.
- [19] Stolberg H.-J., B.M., Friebe, L. , Moch S., Flugel S., Xun Mao, Kulaczewski M.B. , Klussmann H. , Pirsch, P. . HiBRID-SoC: a multi-core system-on-chip architecture for multimedia signal processing applications. in *Design, Automation and Test in Europe Conference and Exhibition*. 2003.
- [20] Shao-Yi Chien, Y.-W.H., Ching-Yeh Chen, Homer H. Chen, and Liang-Gee Chen, *Hardware Architecture Design of Video Compression for Multimedia Communication Systems*. IEEE Communications Magazine, 2005: p. 123-131.
- [21] Franco Casalino, G.D.C., Ronco Luca. MPEG-4 Video Decoder Optimization in 1999 IEEE International Conference on Multimedia Computing and Systems (ICMCS'99. 1999.
- [22] Yang Song, Z.L., Satoshi Goto, Takeshi Ikenaga. Motion Estimation Algorithm Modification and Implementation in H.264/AVC. in *The 18th Workshop on Circuits and Systems*. 2005.
- [23] Yen-Kuang Chen, E.Q.L., Xiaosong Zhou, Steven Ge, Implementation of H.264 encoder and decoder on personal computers. *Journal of Visual Communications and Image Representation*, 2006. 17: p. 509-532.
- [24] Sung-Wen Wang, Y.-T.Y., Chia-Ying Li, Yi-Shin Tung, Ja-Ling Wu The optimization of H.264/AVC baseline decoder on low-cost TriMedia DSP processor. in *Proceedings of SPIE Applications of Digital Image Processing XXVII*. 2004.
- [25] Tung-Chien Chen National Taiwan University, T., Taiwan and C.-J. Lian. Hardware architecture design of an H.264/AVC video codec in the 2006 conference on Asia South Pacific design automation table of contents. 2006.
- [26] Lappalainen V., H.A., Hamalainen T.D. , Complexity of optimized H.26L video decoder implementation. *Circuits and Systems for Video Technology*, IEEE Transactions on, 2003. 13(7): p. 717-725.
- [27] Roitzsch, M., Slice-Balancing H.264 Video Encoding for Improved Scalability of Multicore, in *27th IEEE International Real-Time Systems Symposium RTSS'06*. 2004.
- [28] Shih-hao Wang, W.-h.P., A Software-Hardware Co-Implementation of MPEG-4 Advanced Video Coding(AVC) Decoder with Block Level Pipelining. *VLSI Signal Processing*, 2005. 41: p. 93-110.
- [29] Erik B. van der Tol, E.G.J., Rob H. Gelderblom. Mapping of H.264 decoding on a multiprocessor architecture in *Image and Video Communications and Processing SPIE/IS&T 2003*. 2003. Santa Clara.
- [30] You Yang, G.J., Mei Yu, Dingju Zhu Parallel Process of Hyper-Space-Based Multiview Video Compression. in *Image Processing, 2006 IEEE International Conference on*. 2006.
- [31] PANG Yi, G.S., SUN Lifeng, YANG Shiqiang, Spatial and Temporal Data Parallelization of Multi-view Video Encoding Algorithm, in *MMSP*. 2007.
- [32] Moore., G.E., [Http://www.intel.com/technology/silicon/mooreslaw/](http://www.intel.com/technology/silicon/mooreslaw/).

-
- [33] G.J. Hekstra, G.D.L.H., P.Bingley, F.W.Sijstermans. Trimedia CPU 64 design space exploration. in Int. Conf. Computer Design. 1999. Austin, TX.
 - [34] F.Sijstermans, E.J.P., B.Riemens, K.Vissers, S.Rathnam, Slavenburg, G.Acoustics. Design space exploration for future Trimedia CPU's. in IEEE Int. Conf. Acoustics, Speech, and Signal Processing. 1998.
 - [35] M.Berekovic, P.P., T. Selinger, K.-I. Wels, C. Miro, A. Lafage, C.Heer, G.Ghigo. Architecture of an image redering co-processor for MPEG-4 systems. in IEEE Int. Conf. Application-Specific Systems, Architectures and Processors. 2000.
 - [36] SUN. 2006 [cited; Available from: <http://www.sun.com/processors/>].
 - [37] Corp., I. Intel Pentium III Streaming SIMD Extensions. 2002 [cited; Available from: <http://developer.intel.com/vtune/cbts/simd.htm>].
 - [38] Corp., I. Introduction to Hyper-Threading Technology. 2002 [cited; Available from: <http://developer.intel.com/technology/hyperthread>].
 - [39] Machines., I.B. The Cell Project at IBM Research. 2005 [cited; Available from: <http://www.research.ibm.com/cell/>].
 - [40] Keith Diefendorff, P.K.D., How multimedia workloads will change processor design. Computer Journal, 1997. 30(9): p. 43-45.
 - [41] Ananth Grama, G.K., Vipin Kumar, Anshul Gupta, ed. Introduction to Parallel Computing, 2/E. 2003.
 - [42] Zhenyu Liu, Y.S., Takeshi Ikenaga, Satoshi Goto. Low-Pass Filter Based VLSI Oriented Variable Block Size Motion Estimation Algorithm For H.264. in ICASSP 06'. 2006.
 - [43] A.D.Carlson, R.W.C., R.O.Mueller, Multimedia extensions for a 550-MHz RISC microprocessor. Solid-State Circuits, 1997. 32(11): p. 1618-1624.
 - [44] Corporation, I. Intel Architecture Optimization Reference Manual. 2000 [cited; Available from: <Http://developer.intel.com/design/pentiumii/manuals>].
 - [45] Corporation, I., Intel Releases MMX Technology Details to Software Community to Drive New Multimedia, Game, and Internet Applications. 1996.
 - [46] Nathan T. Slingerland, A.J.S., Multimedia instruction sets for general purpose microprocessors: a survey. University of California at Berkeley Technical Report CSD-00-1124, 2000.
 - [47] Nathan T. Slingerland, A.J.S., Measuring the performance of multimedia instruction sets. IEEE Transactions on Computers, 2002. 51(11): p. 1317-1332.
 - [48] Mombers F., M.D. A multithreaded multimedia processor merging on-chip multiprocessors and distributed vector pipelines. 1999.
 - [49] Lopez, J.F.T., F. Lopez, S. Cortes, P. Lalchand, S. Sarmiento, R. . VLSI video processing elements for real time applications. in IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]. 2002.
 - [50] Yingwei Chen, Z.Z., Tse-Hua Lan, Sharon Peng, and Kees van Zon, Regulated Complexity Scalable MPEG-2 Video Decoding for Media Processors. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY,, 2002. 12(8): p. 678-687.
 - [51] P.Foley. The Mpact media processor redefines the multimedia PC. in IEEE Proceedings of

- COMPCON. 1996.
- [52] Christoforos Kozyrakis, D.P. Vector Vs. Superscalar and VLIW Architectures for Embedded Multimedia Benchmarks. in the 35th International Symposium on Microarchitecture. 2002. Instabul, Turkey.
 - [53] S.Rathnam, G.S. An architectural overview of the programmable multimedia processor,TM-1. in IEEE Proceedings of COMPCON. 1996.
 - [54] JTC1, J.V.T.o.I.-T.a.I.I., Draft ITU-T Recommendation and Final Draft International Standardof Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC). JVT-G050r1, 2003.
 - [55] Tanimoto, M. FTV (Free Viewpoint Television) Creating Ray-Based Image Engineering. in International Conference on Image Processing. 2005.
 - [56] A. Smolic, P.K. Interactive 3-D video representation and coding technologies. in Proceedings of the IEEE. 2005.
 - [57] Dodgson, N.A., Autostereoscopic 3D Displays. IEEE Computer, 2005. 38(8): p. 31-36.
 - [58] B. Wilburn, e.a., High Performance Imaging Using Large Camera Arrays. ACM Transactions on Graphics, 2005. 24(3): p. 765-776.
 - [59] JTC1/SC29/WG11, I.I. Updated Call for Proposals on Multi-view Video Coding. in MPEG Doc. N756. 2005. Nice, Franc.
 - [60] N8218, I.I.J.S.W., Requirements on Multi-view Video Coding v.7. 2006.
 - [61] N8459, I.I.J.S.W., Joint Multi-view Video Model (JMVM) 2.0. 2006.
 - [62] Emin Martinian, A.B., Jun Xin,Anthony Vetro, Huifang Sun, Extensions of H.264/AVC For Multi-view Video Compression, in MITSUBISHI ELECTRIC RESEARCH LABORATORIES. 2006.
 - [63] JVT-V209, J.V.T.J.o.I.I.M.I.-T.V., Joint Draft 2.0 on Multi-view Video Coding. 2007.
 - [64] N8244, I.I.J.S.W., Joint Multi-view Video Model (JMVM) 3.0. 2007.
 - [65] Fletcher, R., Practical Methods of Optimization, . Constrained Optimization, John Wiley and Sons, Chichester, 1981. 2.
 - [66] J.A.Kahle, M.N.D., H.P.Hofstee,C.R.Jogns,T.R.Maeurer,D.Shippy, Introduction to the cell multiprocessor. IBM Systems Journal, 2005. 49(4).
 - [67] B. Flachs, S.A., S.H. Dhong, P. Hofstee, G. Gervais, R. Kim,T. Le, P. Liu, J. Leenstra, J. Liberty, B. Michael, H. Oh, S. M. Mueller, O. Takahashi, A. Hatakeyama, Y. Watanabe, and N. Yano. A Streaming Processing Unit for a CELL Processor. in International Solid-State Circuits Conference. 2005.
 - [68] Carsten Benthin, I.W., Michael Scherbaum, Heiko Friedrich. Ray Tracing on the Cell Processor. in Proceedings of the IEEE Symposium on Interactive Ray Tracing. 2006.
 - [69] Alan Chalmers, T.D., and Erik Reinhard,, Practical Parallel Rendering. . 2002.
 - [70] B. Khailany, W.J.D., U.J. Kapasi, P. Mattson, J. Namkoong, J.D. Owens, B. Towles, A. Chang, and S. Rixner. , Imagine: media processing with streams. IEEE Micro, 2001. 21(2): p. 35-46.
 - [71] Matt Pharr, C.K., Reid Gershbein, and Pat Hanrahan. , Rendering Complex Scenes with Memory-Coherent Ray Tracing. Computer Graphics, 1997. 31: p. 101-108.

- [72] T.J. Purcell, I.B., W.R. Mark, and P. Hanrahan. , Ray Tracing on Programmable Graphics Hardware. *ACM Transactions on Graphics*,, 2002. 21(3): p. 703-712.
- [73] SamuelWilliams, J.S., Leonid Oliker, Shoal Kamil, Parry Husbands, and Katherine Yelick. The Potencial of the Cell Processor for Scientific Computing. in *In CF '06: Proceedings of the 3rd conference on Computing frontiers*. 2006.
- [74] Whitted., S.M.R.a.T., A three-dimensional representation for fast rendering of complex scenes. *Computer Graphics*,, 1980. 14(3): p. 110-116.
- [75] Woo., J.A.a.A. A Fast Voxel Traversal Algorithm for Ray Tracing. in *Proceedings of Eurographics*,. 1987.
- [76] Jansen., F.W. Data structures for ray tracing. in *Proceedings of the workshop on Data structures for Raster Graphics*. 1986

致 谢

在写完硕士论文后,我即将进入攻读博士学位的学习阶段,回想两年多来的学习和生活,我的导师、教研组其他老师以及我的师兄弟都给了我莫大的帮助。

我的导师杨士强教授不仅在学业上悉心教导了我,他更是我人生道路的向导。杨老师在多媒体领域高屋建瓴的远见和把握大局的能力,使我受益匪浅;他严谨认真的科学作风将指导我一生。同时,杨老师勤勉、向上的人生态度使我学到了做人的道理。我非常幸运能在未来的三年内继续接受杨老师的指导。孙立峰老师宽广的知识面和精湛的学术水平也使我学到不少知识。我的师兄弟在学习和工作中给了我很多有益的建议和帮助,他们是黄峰、张建宁、汤筠、李彬、崔鹏、王静远等。在此对他们表示衷心的感谢!

=====

声 明

本人郑重声明:所呈交的学位论文,是本人在导师指导下,独立进行研究工作所取得的成果。尽我所知,除文中已经表明引用的内容外,本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体,均已在文中以明确方式表明。

签 名: _____ 日期: _____

个人简历、在学期间发表的学术论文与研究成果

个人简历

1983 年 8 月 12 日出生于河北省秦皇岛市

2001 年 9 月考入清华大学计算机科学与技术系，2005 年 7 月本科毕业并获得工学学士学位。

2005 年 9 月免试进入清华大学计算机科学与技术系攻读计算机科学与技术系博士至今。

发表的学术论文

- [1] **PANG Yi**, SUN Lifeng, GUO Songliu, YANG Shiqiang. Spatial and Temporal Data Parallelization of Multi-view Video Encoding Algorithm, the ninth international workshop on multimedia signal processing (IEEE MMSP 2007, Short-listed for Best Paper Awarded), Greece, Oct.2007
- [2] **PANG Yi**, YANG Shiqiang. Design Method of Media Accelerating Engine Based RISC, HHME 2005, Kun Ming Oct.2005

研究成果

参加项目：2005.9-2006.9 参加国际科技合作项目“软硬件混合的多媒体处理器开发”

2006 至今 负责 IBM 共享高校研究项目（SUR）