

基于流体系结构的帧内预测算法优化设计

李海燕^{1,2}, 张春元¹, 付 剑¹

(1. 国防科学技术大学计算机学院, 湖南长沙 410073; 2. 海军指挥学院海战实验室, 江苏南京 210016)

摘 要: 为高效实现 H.264 多模式帧内预测, 解决其计算复杂度高造成的计算压力, 本文根据 H.264 帧内预测算法的计算密集与数据并行的特征, 基于流处理执行模型提出适用于 Imagine 流体系结构的分组帧内预测流算法, 并采用长流分段技术进行优化设计. 实验结果表明, H.264 帧内编码器流实现 1280 × 720 高清视频编码帧率达 45.9fps, 满足实时性需求.

关键词: 视频编码; H.264; 帧内预测; 流处理; 长流分段; 循环展开

中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112 (2010) 05-1014-07

Optimized Design of Intra Prediction Algorithm on Stream Architecture

LI Hai-yan^{1,2}, ZHANG Chun-yuan¹, FU Jian¹

(1. School of Computer, National University of Defense Technology, Changsha, Hunan 410073, China;

2. Naval War Gaming Center, Naval Command College, Nanjing, Jiangsu 210016, China)

Abstract: Multi-mode intra prediction in H.264 is computationally expensive. For efficient implementation to reduce computing pressure, according to computation intensity and data parallelism in intra prediction, a multi-group intra prediction stream algorithm suited for imagine stream architecture is proposed based on stream execution model. In addition, strip-mining technology is applied into streaming implementation to design an optimized strip-mined multi-group intra prediction stream algorithm. Experimental results show that H.264 intra coder using optimized stream algorithm processes 1280 × 720 HD video at 45.9fps, satisfying the requirements of real-time video applications.

Key words: video coding; H.264; intra prediction; stream processing; strip-mining; loop unrolling

1 引言

目前流行的 H.264 视频编码标准具有低码率高压缩比高图像质量的编码优势, 它引入了许多性能良好的压缩工具, 帧内预测就是其中一种^[1]. H.264 采用的多模式帧内预测技术, 通过消除视频像素之间的空间冗余可以保证预测编码帧的压缩效果, 但是其计算复杂造成时间开销很大. 对于计算资源受限的视频应用如便携式视频设备来说, 视频应用的实时性需求要求处理芯片在选用包括帧内预测在内的压缩工具时尽可能地采用快速算法或高效实现.

帧内预测的计算密集性与数据并行性使其很适合于流处理实现. 流处理^[2,3]是一种新型的并行处理方式, 它将计算显式划分为多个核心, 将操作数据组织成流的形式, 流中的各元素记录依次通过核心参与计算. 流处理将数据存取与数据计算相互分离, 使得访存与计

算在时间上尽可能重叠, 从而实现延迟隐藏. 流体系结构以其独特的存储带宽层次、大量的运算单元以及高效的执行模式在包括视频在内的媒体与信号处理领域获得优越的性能^[4,5]. 因此, 流体系结构为帧内预测的高效实现提供了一种新的解决方案.

本文结合帧内预测的算法特征与流处理的执行模式, 提出了适用于 Imagine 流体系结构的帧内预测算法, 并分别采用核心级与流级优化策略, 形成分组帧内预测 (Multi-group Intra Prediction, MgIP) 和分段分组帧内预测 (Strip-mined Multi-group Intra Prediction, SmMgIP) 流算法, 提高了流实现的执行效率. 实验结果表明, 基于流处理的帧内预测算法可以使得 H.264 帧内编码器实现对 1280 × 720 HD 视频的实时编码, 帧率可达 45.9fps, 且针对不同视频格式具有扩展性, 应用性很强. 同时, 该算法思想可以移植在其它基于 SIMD 数据并行执行模型的计算平台上, 具有一定的通用性.

2 H.264 帧内预测流特征分析

在 H.264 标准中,帧内预测是直接在空间域对当前块的每个像素做出预测,更能去除真实的空间冗余.它包含三种预测模式^[6]:4×4 亮度预测(Intra_4×4)、16×16 亮度预测(Intra_16×16)和 8×8 色度预测(Intra_Chroma).

在 4×4 亮度预测中,H.264 提供了 9 种 Intra_4×4 预测模式,如图 1 所示,其中 a~p 表示待预测的当前块像素,A~M 表示用于生成预测块的相邻像素.除直流预测模式(Mode2)外,每种预测模式代表着从某一方向对当前 4×4 块相邻像素进行预测计算.通过选择具有最小预测误差的方向作为最佳预测方向,而与该方向对应的模式即为最佳预测模式.

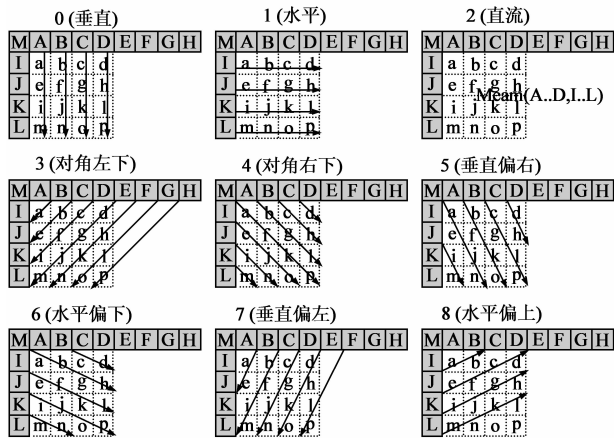


图1 4×4亮度块帧内预测示意图

H.264 规定了每种预测模式的计算公式^[7]:

$$p(x,y) = (a_{-1,0}I + a_{-1,1}J + a_{-1,2}K + a_{-1,3}L + a_{-1,-1}M + a_{0,-1}A + a_{1,-1}B + a_{2,-1}C + a_{3,-1}D + a_{4,-1}E + a_{5,-1}F + a_{6,-1}G + a_{7,-1}H + RoundNum) \gg ShiftNum \quad (1)$$

其中, RoundNum 为四舍五入参数, ShiftNum 为移位参数.从公式(1)可以看出, Intra_4×4 生成的预测像素是通过当前块左边一列像素与上边一行像素进行不同的加权和计算来预测的.

Intra_16×16 和 Intra_Chroma 具有基本一致的预测模式,包括垂直预测、水平预测、直流预测与平面预测 4 种预测模式.同时两种色度分量采用相同的预测模式.

在 H.264 标准中,亮度块帧内预测共有 13 种预测模式,色度块帧内预测共有 4 种预测模式.而对单一块或宏块,只能选择一种亮度或色度最佳预测模式.那么如何选择预测模式,是帧内预测需要解决的关键问题.预测模式的选择,直接关系到整个系统的性能,但在 H.264 标准中并没有具体规定.在 JM86^[8]中,帧内预测采用率失真优化技术并通过全搜索来确定最佳预测

模式,对一个宏块的帧内预测需要执行 592 次不同的计算.因此,H.264 帧内预测计算复杂度高,属于计算密集型应用.

在文献[9]中,流执行模型突出显示了流控制模块、流计算模块和流存储模块之间的控制关系与数据流向,见图 2.流控制模块负责解析并发射指令给相应的功能模块(如核心执行单元、片上流寄存器文件等)执行.流计算模块是实现密集计算的执行部件,多簇核心执行单元的硬件结构利于开发大量数据并行性.在帧内预测计算过程中,宏块亮度分量的预测与色度分量的预测是完全独立的,可以并行地执行.数据并行处理的粒度也可以不同,如每个运算簇处理一个 4×4 块,或者每个运算簇处理一个 16×16 宏块,并行粒度的差异对帧内预测的执行效率也有一定的影响.片上流寄存器文件(Stream Register Files,SRF)是流存储模块中的重要部件,用于存放计算所需的数据流.流处理器将处理的数据组织成流的形式,流中的每个元素生存周期并不长,仅持续一小段时间并经过相同的计算过程.因此,数据流的特征是,流元素具有高度的空间局部性,但时间局部性有限.那么,如何有效地分布片上流寄存器文件,使得最大工作数据集全部容纳在片上存储空间而无需写回片外存储系统,发挥“生产者-消费者”局部性的优势,将是流应用获取最优性能必须考虑的问题.同时,帧内预测的输入是当前帧各像素块的相邻像素集合,输出则是当前帧各像素块对应的预测块集合.这样,在计算之前能够准确获知整个预测程序的输入输出集合,属于规则索引的数据访问模式,好处在于可以人为地设计数据流重组,提前进行数据预取,以便实现访存的延迟隐藏.

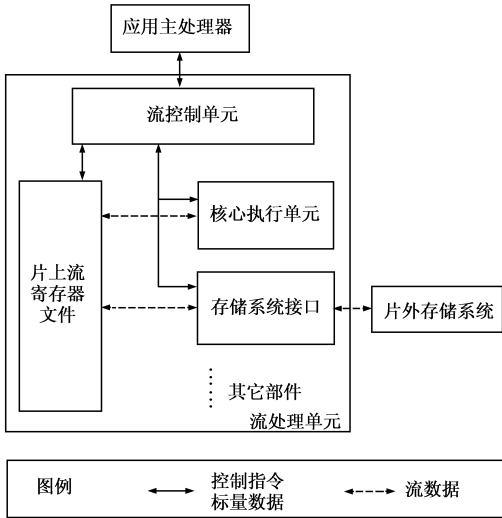


图2 通用流执行模型

接下来,我们以 H.264 亮度块 4×4 帧内预测(Intra_4×4)为例,说明基于 Imagine 流体系结构的帧内

预测流算法.

3 帧内预测流算法设计与优化

3.1 初始流算法 (OrigIP)

算法 3.1 Intra_4×4 预测 OriginalIntra×4, OrigIP

输入: 行排列参考像素流 $S_refRowPixels$
列排列参考像素流 $S_refColumnPixels$
输出: 最佳亮度预测块流 $S_bestPredBlocks$
最佳亮度预测模式流 $S_bestPredModes$

- 1 对 $S_refRowPixels$ 中每 4 个像素
- 2 派生加载 $S_refRowPixels$ 该组像素的后续 4 个像素 (即右上像素组);
- 3 索引加载 $S_refRowPixels$ 该组像素的前 1 个像素 (即左上角像素);
- 4 索引加载 $S_refColumnPixels$ 对应 4 个列像素 (即左像素组);
- 5 执行 Intra_4×4 模式 0~8 预测计算;
- 6 生成各模式下的预测块 $predBlock_i (i = 0.8)$
- 7 循环 1~6 步直至所有当前块的参考像素均完成预测, 生成各模式下的预测块流 $S_predBlock_i (i = 0.8)$
- 8 对 Intra_4×4 的模式 0~8 预测生成的预测块流 $S_predBlock_i (i = 0.8)$
- 9 执行预测残差的 SAD 计算;
- 10 对当前 SAD 值流中每个流记录
- 11 比较当前 SAD 值与以前最小 SAD 值 $MinSAD$. 如果当前 SAD 小于 $MinSAD$, 则当前 SAD 值为新的 $MinSAD$, 当前模式为最佳预测模式, 当前块为最佳预测块;
- 12 循环 8~11 步直至所有的预测模式都完成, 更新生成完整的 $S_bestPredBlocks, S_bestPredModes$.

图 3 Intra_4×4 初始流算法

根据核心划分的解耦原则, 初步将 Intra_4×4 预测划分为 2 个核心, 如图 3 所示. 第一个核心是算法 3.1 中第 1~7 行描述的预测块计算部分 (简称为 $K_predComp$), 第二个核心是算法 3.1 中第 8~12 行描述的预测模式选择部分 (简称为 $K_predMode$). 算法 3.1 中第 5 行是核心 $K_predComp$ 的计算集中点, 包含 9 种预测模式的全部计算. 核心 $K_predMode$ 的输入流是 $K_predComp$ 的输出流, 即各预测模式下生成预测块流 $S_predBlock_i (i = 0.8)$, 这就是核心之间“生产者-消费者”局部性, 作为中间结果流仍然驻留在 SRF 中, 无需传回片外存储器. 流算法 3.1 中第 8~11 行是采用以绝对差和 (Sum of Absolute Differences, SAD) 评估函数作为预测模式选择策略. 核心 $K_predMode$ 的输出流将作为整个 Intra_4×4 程序的结果.

3.2 分组预测流算法 (MgIP)

由于流处理器的 SIMD 执行模式要求核心对每个像素块执行相同的操作, 那么 OrigIP 核心对边界像素块 (视频帧第一行与第一列) 的计算过程与对非边界的普通像素块完全相同, 这就意味着边界像素块的部分预测计算是无效的, 带来了多余的计算时间, 影响预测的效果.

为解决这个问题, 根据 Intra_4×4 预测所需相邻像素的可用性, 将 OrigIP 中的核心 $K_predComp$ 拆分为 3 个不同的核心 $K_predByLeft, K_predByUp$ 和 $K_predByUpLeft$ (见图 4), 生成 Intra_4×4 分组预测算法 (图 5 中的算法 3.2). 同时, 将参考像素流 $S_refPixels$ 按照索引流方式拆分为若干子流 ($S_leftPixels, S_upPixels, S_upRightPixels, S_upLeftPixels$) 作为三个不同预测核心的输入流. 其中, 核心 $K_predByLeft$ 用于处理只存在左边相邻像素的 4×4 块, 即帧图像的第一行; 核心 $K_predByUp$ 用于处理只存在上边相邻像素的 4×4 块, 即帧图像的第一列; 核心 $K_predByUpLeft$ 用于处理左边与上边像素同时存在的 4×4 块, 即帧图像除去边界外的其余部分. 输出流按照 9 种预测模式分别生成各自的预测像素流, 再通过流整合重组预测像素流形成按照自然行排列的预测块流 $S_predBlock_i (i = 0.8)$. 而预测模式选择核心 $K_predMode$ 按照程序设计的需要拆分为 2 个连续的核心 $K_compSAD$ 与 $K_compMinSAD$, 分别计算各预测模式对应的 SAD 值以及计算最小 SAD, 从而选择最佳预测模式以及确定对应的最佳预测块.

表 1 描述了 H.264 标准规定的 Intra_4x4 各模式计算过程涉及到的基本指令操作, 除输入输出外, 使用加法和移位操作便可实现整个预测过程. 其中, 赋值是指不同变量之间的值传递.

在 Imagine 流处理器的 ISIM 模拟器^[10]中, 加法操作 ADD 与移位操作 SHIFT 都是使用加法器完成的, 其中移位操作延迟是 1 个时钟周期, 加法操作延迟是 2 个时钟周期. 输入操作 INPUT 与输出操作 STORE 的时钟延迟均为 1 个时钟周期. 由此可得对 MgIP 算法进行详细

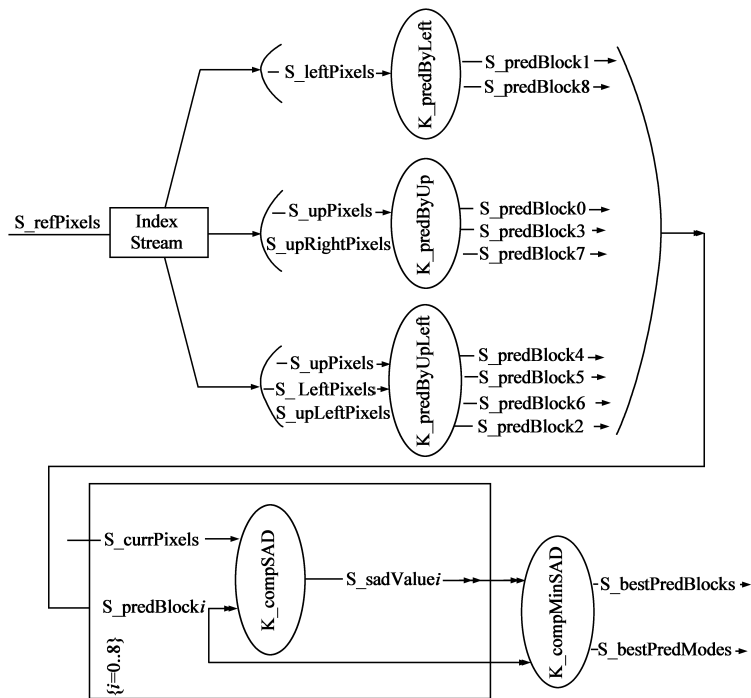


图 4 Intra_4x4 预测核心划分

算法 3.2 Intra_4×4 分组预测 MultiGroupIntra4×4, MgIP

```
输入:参考亮度像素流 S_refPixels
输出:最佳亮度预测块流 S_bestPredBlocks
最佳亮度预测模式流 S_bestPredModes
1 对片外存储器中参考亮度像素流 S_refPixels 进行索引值的计算;
2 加载索引子流 S_leftPixels, S_upPixels, S_upRightPixels, S_upLeftPixels 到 SRF 中;
3 对 S_leftPixels 中每 4 个像素
4 执行 Intra_4×4 模式 1,8 预测计算;
5 对 S_upPixels 中每 4 个像素, S_upRightPixels 中每 4 个像素
6 执行 Intra_4×4 模式 0,3,7 预测计算;
7 对 S_leftPixels 中每 4 个像素, S_upPixels 中每 4 个像素, S_upLeftPixels 中每 4 个像素
8 执行 Intra_4×4 模式 2,4,5,6 预测计算;
9 对 Intra_4×4 的模式 0 至 8 预测生成的预块流 S_predBlocki (i = 0..8)
10 执行预测残差的 SAD 计算;
11 对当前 SAD 值流中每个流记录
12 比较当前 SAD 值与以前最小 SAD 值 MinSAD. 如果当前 SAD 小于 MinSAD, 则当前 SAD 值为新的 MinSAD, 当前模式为最佳预测模式, 当前块为最佳预测块;
13 循环 9~12 步直至所有的预测模式都完成, 更新生成完整的 S_bestPredBlocks, S_bestPredModes.
```

图 5 Intra_4×4 分组预测流算法

表 1 Intra_4×4 各预测模式的基本指令操作

	加法 ADD	移位 SHIFT	输入 INPUT	输出 STORE	赋值
垂直预测 (Mode 0)	0	0	4	16	16
水平预测 (Mode 1)	0	0	4	16	16
直流预测 (Mode 2)	8	1	8	16	16
对角左下预测 (Mode 3)	48	32	8	16	0
对角右下预测 (Mode 4)	48	32	9	16	0
垂直偏右预测 (Mode 5)	41	25	9	16	0
水平偏下预测 (Mode 6)	41	25	9	16	0
垂直偏左预测 (Mode 7)	40	24	8	16	0
水平偏上预测 (Mode 8)	25	15	4	16	6

表 2 分组预测算法复杂度分析

		串行理论值 (cycles)	并行理论值 (cycles)	实际值 (cycles)
K_predByLeft	Mode 1	4 * 1 + 16 * 1 = 20	(25 * 2 + 15 * 1) / 3 = 22	32
	Mode 8	25 * 2 + 15 * 1		
		+ 4 * 1 + 16 * 1 = 85		
K_predByUp	Mode 0	4 * 1 + 16 * 1 = 20	(48 * 2 + 32 * 1 + 40 * 2 + 24 * 1 + 8 * 1) / 3 = 78	64
	Mode 3	48 * 2 + 32 * 1 + 8 * 1 + 16 * 1 = 152		
	Mode 7	40 * 2 + 24 * 1 + 8 * 1 + 16 * 1 = 128		
	Mode 2	8 * 2 + 1 * 1 + 8 * 1 + 16 * 1 = 41		
	Mode 4	48 * 2 + 32 * 1 + 9 * 1 + 16 * 1 = 153		
K_predByUpLeft	Mode 5	41 * 2 + 25 * 1 + 9 * 1 + 16 * 1 = 132	(8 * 2 + 1 * 1 + 32 * 1 + 41 * 2 + 25 * 1 + 41 * 2 + 25 * 1) / 3 = 120	96
	Mode 6	41 * 2 + 25 * 1 + 9 * 1 + 16 * 1 = 132		
	Mode 8	25 * 2 + 15 * 1 = 65		

的分析,见表 2.

Imagine 运算簇包含 3 个加法器,8 个输入输出端口.各功能单元在满足指令相关性的前提下可以并行

执行.对于核心 K_predByLeft,实际值比并行理论值大,这是因为该核心体积小,计算简单,主要集中于 Mode 8 预测部分,计算前后的访存开销无法隐藏在计算过程之中.对于核心 K_predByUp 与 K_predByUpLeft,这两个核心体积大,计算密集,且存在相同临时变量的运算,因此在寄存器分配时冗余的运算无需重复执行,从而减少了实际的操作数;另外,计算前后的访存时间可以分销在计算过程之中,几乎忽略不计.这种将几种预测模式的计算封装在一个核心的方法,实质上是一种核心合并优化技术的体现,它有效地开发了指令级并行性,增强了计算密集性,提高了核心的执行效率.

3.3 分段分组预测流算法 (SmMgIP)

SRF 是流处理器片上存放流数据的主要存储部件,因此数据流的最大长度不能超出 SRF 的容量.目前,Imagine 流处理器流级编译提供双缓冲机制^[11],这样可以支持任意长度的流.随着视频帧图像格式分辨率的增大,用以描述同一帧内像素的数组规模往往很大,映射到流程序时将产生很长的数据流.我们期望将长流切分成为若干个短流,再分批处理,从而使得每一批次的短流计算涉及到的流集合包括输入输出流以及生成的中间流可以全部容纳在 SRF 中,以提高 SRF 的带宽利用率,这就是长流分段技术.

算法 3.3 Intra_分段分组预测 StripminedMultiGroupIntra4×4 SmMgIP

```
输入:参考亮度像素流 S_refPixels
输出:最佳亮度预测块流 S_bestPredBlocks
最佳亮度预测模式流 S_bestPredModes
1 对片外存储器中参考亮度像素流 S_refPixels 进行索引值的计算;
2 按事先设定的分段大小,处理每个批次像素块的预测过程;
3 加载索引子流 S_leftPixels, S_upPixels, S_upRightPixels, S_upLeftPixels 到 SRF 中;
4 对 S_leftPixels 中每 4 个像素
5 执行 Intra_4×4 模式 1,8 预测计算;
6 对 S_upPixels 中每 4 个像素, S_upRightPixels 中每 4 个像素
7 执行 Intra_4×4 模式 0,3,7 预测计算;
8 对 S_leftPixels 中每 4 个像素, S_upPixels 中每 4 个像素, S_upLeftPixels 中每 4 个像素
9 执行 Intra_4×4 模式 2,4,5,6 预测计算;
10 执行 Intra_4×4 的模式 0 至 8 预测生成的预测块流 S_predBlocki (i = 0..8)
11 执行预测残差的 SAD 计算;
12 对当前 SAD 值流中每个流记录
13 比较当前 SAD 值与以前最小 SAD 值 MinSAD. 如果当前 SAD 小于 MinSAD, 则当前 SAD 值为新的 MinSAD, 当前模式为最佳预测模式, 当前块为最佳预测块;
14 循环 10~13 步直至所有的预测模式都完成, 更新生成完整的 S_bestPredBlocks(j = 0...stripCount - 1), S_bestPredModes(j = 0...stripCount - 1);
15 循环 2~14 步直至所有批次的像素块预测完毕, 更新生成完整的 S_bestPredBlocks 和 S_bestPredModes.
```

图 6 Intra_4×4 分段分组预测流算法

对于 H.264 视频编码器的程序设计,我们把整帧像素独自作为流记录组成一条输入流.那么当对 QCIF 图像格式编码时,输入流的长度为:176 * 144 * 4B

=99kB(单位 Bytes 简称为 B);而 1080p HD 图像格式对应的输入流长度则为: $1920 \times 1080 \times 4B \approx 7.91MB$. 在 Imagine 流处理器中,片上 SRF 的大小为 128kB,即可以同时容纳 32k 个 4 字节的字.那么,当图像分辨率大小增加时,输入流的长度随之逐渐增加,远超过 SRF 的容量.因此,我们在编码过程的第一个模块即帧内预测部分便对整帧像素流进行长流分段,使得每一小段像素流独立完成完整的编码过程.图 6 中的算法 3.3 是在 MgIP 算法的基础上应用长流分段优化技术形成的分段分组流算法,与 MgIP 算法的主要区别在于算法 3.3 第 2 行指示的最外层循环.

在 SmMgIP 算法中,输入流为一条参考亮度像素点流,两条输出流分别是最佳亮度预测块流和最佳亮度预测模式流.假设每个 32 位字的流记录为一个像素点,对于 176×144 的 QCIF 图像格式,最优分段大小为每批次 36 行像素点,即将整幅视频帧划分为 4 个短流依次参与帧内预测计算.因此,可得如图 7 所示的 SmMgIP 算法伪代码.

```
1  int stripCount = 4; //for QCIF
2  int stripSize ≈ picWidth * picHeight / stripCount;
3  for (int i = 0; i < stripCount; i++) {
4      im_stream < im_int > currRefPixels(stripSize);
5      streamCopy(refPixels(i * stripSize, (i + 1) * stripSize, currRefPixels);
6      processIntra4 × 4(currRefPixels, ...);
7      ...
8  }
```

图 7 Intra_4 × 4 分段分组预测流算法伪代码

图 7 的第 3~8 行为 SmMgIP 算法的主循环,其中第 4 行为最优分段条件下当前处理像素流的定义,流长度为 stripSize,每个流记录是 32 位整型数据;第 5 行负责准备当前需要处理的像素流,streamCopy 是流复制语句,它将原始流 refPixels 中规定起止位置(从 $i \times \text{stripSize}$ 到 $(i + 1) \times \text{stripSize}$)的部分流记录复制到当前处理子流 currRefPixels 中;第 6 行函数调用,processIntra4 × 4 是表征 SmMgIP 算法计算主体的函数.

4 实验结果及讨论

结合流处理执行模型,分析得出帧内预测算法是以规则索引流访问与规则流计算为主要流特征,按其逐步优化方法相继设计了初始流算法(OrigIP)、分组预测流算法(MgIP)、分段分组预测流算法(SmMgIP),见表 3.我们将三组 Intra_4 × 4 流算法分别映射在 Imagine 时钟精确模拟器 ISIM(500MHz)上.

将 Intra_4 × 4 长流分段的分组帧内预测流算法策略同样应用于 Intra_16 × 16 与 Intra_Chroma,两者的预测模式相对简单,垂直预测与水平预测可以直接通过流赋值来实现,而直流预测与平面预测合并在一个核心内实现.在 ISIM 上测得执行一次 Intra_4 × 4 主循环

的时间为 302cycles,而执行一次 Intra_16 × 16 与 Intra_Chroma 主循环仅需 79cycles 与 52cycles.流化的 H.264 帧内编码器处理一个宏块的时钟周期为 3025cycles,那么 Imagine 流处理器一秒内可处理 1.65×10^5 个宏块.对于 1280×720 HD 视频格式,帧内编码器达到 45.9fps 的编码效率,满足实时性需求.这得益于流程序在核心级与流级两方面的性能优化.

表 3 帧内预测 Intra_4 × 4 流算法列表

算法名称	算法描述
初始流算法(OrigIP)	正常输入流
分组预测流算法(MgIP)	在 OrigIP 基础上,按程序特征对输入流分组
分段分组预测流算法(SmMgIP)	在 MgIP 基础上,根据 SRF 容量对输入流分段

核心级优化的目的是为了提高指令级并行.而指令级并行的优劣与功能单元的占用率存在直接关系,所以衡量核心级程序性能的重要指标是其对功能单元的占用率.预测程序本身的密集计算与良好的数据级并行性,已经能够使得未经优化的核心有着较好的功能单元占用率,以三种预测中计算相对较多的 K_predByUpLeft 为例,表 4 给出循环展开 2 遍优化前后加法器占用情况的比较.总体来说,核心 K_predByUpLeft 的加法器占用率都在 85% 以上,经过循环展开之后占用率提高 2% 左右.但在 Intra_4 × 4 中,由于核心体积过大存在循环展开无法编译成功的情况.因此,我们得出结论:循环展开技术对于那些原本功能单元占用率不是很高且循环体较小的核心有着很好的加速效果.

表 4 预测核心采用循环展开加速比较

K_predByUpLeft 所属模块	加法器占用率		
	循环展开前	循环展开后	加速比
Intra_4 × 4	95.14%	95.14%	1
Intra_16 × 16	89.39%	90.98%	1.018
Intra_Chroma	85.58%	87.54%	1.023

流级设计的初衷在于使得访存时间与计算时间尽可能地重叠在一起,从而缩短流程序的总运行时间,提高系统性能.良好的流级算法能够增强核心计算与流访问传递的并行度,高效地利用片上 SRF 存储空间以及其它关键的硬件资源.相对而言,开销昂贵的片外存储带宽仅在必要的时候使用,避免了巨大的片外访存延迟.长流分段的有效性表明,相对长流而言,较小的可控工作数据集可以更好地开发“生

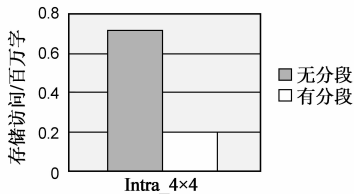


图 8 长流分段优化前后 Intra_4x4 存储访问量对比图

产者-消费者”局部性以及数据并行性,同时减少了访存冲突的发生.图8证实了长流分段技术可以显著降低访存冲突.在帧内预测程序中,使用长流分段优化的存储访问约二十万字,比优化前访存量减少了72%.

图9给出帧内预测程序的部分调度示意图,其中(a)和(b)的左侧列表示核心计算的执行情况,右侧列表示流访存的耗时情况.图9(a)是未采用流级优化措施的程序调度情况,核心调用过程中存在显著的访存停顿,这是因为第二阶段一系列核心所需要的流数据准备无法隐藏在前面核心计算过程中,造成访存单独占用总运行时间,影响了整体性能.由图9(b)与图9(a)比较可得,在同等时钟周期数内,图9(b)调用了更多的核心进行计算.换言之,在执行相同计算过程时,图9(b)比图9(a)节省运行时间.图9(b)通过包括长流分段等流级优化措施很好地改善了计算与访存重叠执行的情况,使得有效的核心计算可以连续执行,无需受限于访存,提高了整体性能.

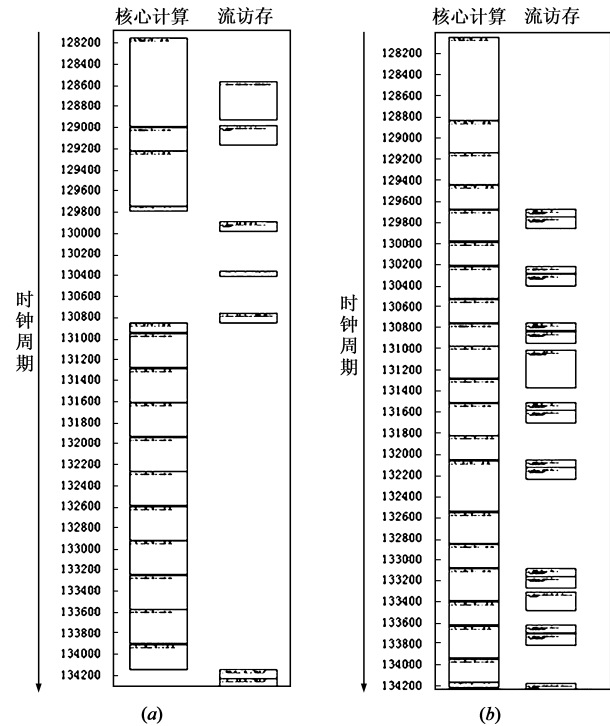


图9 帧内预测程序部分调度图

SRF存储空间的分配是程序获取高性能的重要因素.除了通过长流分段对大规模输入数据进行批次划分之外,也要合理安排流的读写时间,使得每一批次流数据启动计算时所需的输入数据全部准备就绪,且使得流计算涉及到的工作数据集能够很好地分布在SRF中,充分体现出流应用特有的“生产者-消费者”局部性.

图10从存储带宽方面体现了帧内预测程序在各存储层次上的局部性特征,其中LRF表示运算簇内的

本地寄存器文件,SRF表示片上流寄存器文件,MEM表示片外存储系统.帧内预测程序在LRF、SRF和MEM三级使用的存储带宽分别是203.44GB/s、0.91GB/s和0.07GB/s,占Imagine流处理器存储层次峰值带宽^[12]的比例分别为93%、7%和4%.大部分的数据访问都集中在LRF,这是因为计算密集的帧内预测存在大量的核心内部局部性.而SRF与MEM带宽占用情况稍有差异,表明SRF除了作为MEM在片上的替代存储空间外,还有一部分带宽用于捕获核心间的“生产者-消费者”局部性.

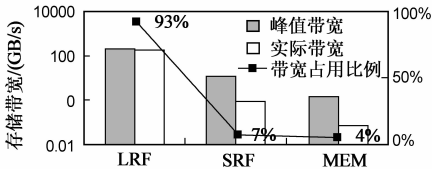


图10 帧内预测程序存储带宽占用图

流处理器的性能优势在于它独特的带宽层次与充足的计算资源,能够开发帧内预测这类典型流应用中固有的数据级并行性与指令级并行性,核心内数据局部性与核心间的“生产者-消费者”局部性可以使功能单元尽早获取所需数据执行计算,有效地提高了程序的计算效率.同时,长流分段优化根据片上SRF容量合理地划分流批次的大小,避免大规模数据在SRF与MEM之间来回传递产生的巨大存储延迟.但是,目前如Imagine流体系结构的硬件组成,尚不能满足1080i HD的实时视频,主要原因在于Imagine流处理器与主处理器之间的存储开销所占比例无法很好地分销在计算之中,致使整体性能不够.解决方案可以考虑改善体系结构,增强流处理器与主处理器之间数据交换的能力,并增加运算簇组数量以提供更多的计算资源加大并行粒度.

5 结论

本文以H.264帧内预测中Intra_4×4为例,针对其规则索引流访问与规则流计算的程序特征,提出基于流处理机制的最优长流分段的分组帧内预测流算法,并在Imagine流处理器上映射实现.该算法按照帧内预测所需参考像素位置的不同采用分组加载执行预测的方法,同时采用流处理器长流分段优化技术,提高帧内预测流实现的效率.实验结果表明,流化的H.264帧内编码器处理1280×720 HD视频格式的帧率可达45.9fps,能够满足视频应用的实时性需求.

参考文献:

[1] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, Ajay Luthra. Overview of the H.264/AVC video coding standard

- [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(7): 560 – 576.
- [2] Brucek Khailany, William J Dally, Scott Rixner, et al. Imagine: media processing with streams[J]. IEEE Micro, 2001, 21(2): 35 – 46.
- [3] 吴伟, 文梅, 等. 多维可扩展流体系结构研究与评测[J]. 电子学报, 2008, 36(5): 899 – 905.
Wu Wei, Wen Mei, et al. Research and evaluating of a multiple-dimension scalable stream architecture[J]. Acta Electronica Sinica, 2008, 36(5): 899 – 905. (in Chinese)
- [4] John D Owens, Scott Rixner, Ujval J Kapasi, et al. Media processing applications on the Imagine stream processor[A]. In Proceedings of the IEEE International Conference on Computer Design[C]. Freiburg: IEEE Computer Society, 2002. 295 – 302.
- [5] Jathin S Rai, Yu-Kuen Lai, Gregory T Byrd. Packet processing on a SIMD stream processor[A]. In Proceedings of the Workshop on Network Processors and Applications in conjunction with HPCA10[C]. Madrid: IEEE Computer Society, 2004. 146 – 157.
- [6] Iain E G Richardson. H. 264 and MPEG – 4 Video Compression-video Coding for Next Generation Multimedia[M]. Chichester: John Wiley & Sons Ltd, 2003.
- [7] 田应洪. 基于 H. 264 基线规范的算法研究与实现[D]. 上海: 复旦大学, 2007. 4.
Tian Yinghong. Study on Algorithms and Implementations Based on H. 264 Baseline Profile[D]. Shanghai: Fudan University, 2007, 4. (in Chinese)
- [8] JVT. JM8. 6, reference software of H. 264 [OL]. <http://iphome.hhi.de/suehring/tml/index.htm>, August 2007.
- [9] 李海燕, 张春元, 李礼, 刘东. 流执行模型及其验证[A]. 2007 年全国高性能计算学术年会论文集[C]. 深圳: 中国科学院深圳先进技术研究院, 2007. 65 – 73.
Li Haiyan, Zhang Chunyuan, Li Li, Liu Dong. Stream execution model and its verification [A]. In Proceedings of the National Annual Conference on High Performance Computing, HPC China 2007[C]. Shenzhen: Shenzhen Institute of Advanced Tech-

nology, Chinese Academy of Sciences, 2007. 65 – 73. (in Chinese)

- [10] Abhishek Das, Peter Mattson, Ujval Kapasi, et al. Imagine programming system user's guide 2.0 [OL]. <http://cva.stanford.edu/projects/imagine>. 2004-06-26.
- [11] Ujval J Kapasi, Peter Mattson, William J Dally, et al. Stream scheduling[R]. California: Stanford University, 2002.
- [12] Jung Ho Ahn, William J Dally, Brucek Khailany, et al. Evaluating the Imagine stream architecture[A]. In Proceedings of the 31st Annual International Symposium on Computer Architecture[C]. Munich: IEEE Computer Society, 2004. 14 – 25.

作者简介:



李海燕 女, 1981 年生于辽宁. 国防科学技术大学计算机学院博士研究生. 主要研究方向为流体系结构应用研究、视频编解码技术.

E-mail: hy_lee@163.com



张春元 男, 1964 年生于安徽. 国防科学技术大学计算机学院教授, 博士生导师. 主要研究方向为高性能微处理器体系结构、并行处理技术、嵌入式系统.



付 剑 男, 1985 年生于浙江. 国防科学技术大学计算机学院硕士研究生. 主要研究方向为嵌入式系统.