计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20148455
53(4):873-883,2016

多层次细粒度并行 HEVC 帧内模式选择算法

张 峻[1,2] 代 锋[1] 马宜科[1] 张勇东[1]
[1](中国科学院智能信息处理重点实验室(中国科学院计算技术研究所) 北京 100190)
[2](中国科学院大学 北京 100049)
(zhangjun01@ict.ac.cn)

Multi-Level and Fine-Grained Parallel HEVC Intra Mode Decision Method

Zhang Jun[1,2], Dai Feng[1], Ma Yike[1], and Zhang Yongdong[1]
[1](*Key Laboratory of Intelligent Information Processing* (*Institute of Computing Technology, Chinese Academy of Sciences*), *Chinese Academy of Sciences*, *Beijing* 100190)
[2](*University of Chinese Academy of Sciences*, *Beijing* 100049)

Abstract The coding mode space of high efficiency video coding (HEVC) is extremely large so it needs huge amount of computations for HEVC encoders to do mode decision (MD). Parallelizing HEVC encoding on many-core platforms is an efficient and promising approach to fulfill the high computational demands. Traditional coarse-grained parallelizing schemes such as Tiles and wavefront parallel processing (WPP) either cause too much quality loss or can't afford a high parallelism degree. In this paper, the potential parallelism in HEVC intra MD process is exploited, and a multi-level and fine-grained highly parallel intra MD method which works in a coding tree unit (CTU) is proposed. Specifically, the intra MD process in a CTU is divided into six types of sub-tasks, and the data dependencies among adjacent blocks that hinder parallel processing are analyzed and removed, including intra prediction dependency, prediction mode dependency and entropy coding dependency; consequently the MD computation for all fine-grained coding blocks of different levels within the same CTU can be computed concurrently. The proposed parallel MD method is implemented on Tile-Gx36 platform. Experimental results show that the proposed parallel MD method gets an overall speed up of more than 18x with acceptable quality loss (about 3% bit-rate increasing), compared with the non-parallel baseline HM.

Key words high efficiency video coding (HEVC); intra prediction; many-core; parallel mode decision; fine-grained

摘 要 在众核平台上并行加速是解决高效视频编码(high efficiency video coding, HEVC)标准编码复杂度高的有效方法.传统的粗粒度并行方案如 Tiles 和 WPP 未能在并行度和编码质量之间取得较好的平衡,对编码质量影响较大或者并行度不高.充分挖掘 HEVC 帧内模式选择中的并行性,提出了一种在 CTU 内使用的多层次细粒度的帧内模式选择算法.具体说来,对帧内模式选择过程进行了子任务划分,分析并消除了相邻编码块之间多种阻碍并行计算的数据依赖关系,包括帧内预测参考像素依赖、预测模式依赖和熵编码依赖等,实现了同一个 CTU 内所有层次的细粒度编码块的代价计算和模式选择并行进行.将算法在 Tile-Gx36 平台上实现,实验结果表明此并行算法与 HEVC 参考代码 HM 相比能获得 18 倍的整体编码加速比而且编码质量损失较小(码率上升 3%).

收稿日期:2014-12-31;**修回日期**:2015-04-27
基金项目:国家自然科学基金项目(61379084,61402440);中国科学院科研装备研制项目(YZ201321)
This work was supported by the National Natural Science Foundation of China (61379084, 61402440) and the Instrument Developing Project of the Chinese Academy of Sciences (YZ201321).

关键词 高效视频编码;帧内预测;众核;并行模式选择;细粒度

中图法分类号 TP391

高效视频编码(high efficiency video coding, HEVC)[1-2]标准的压缩效率超越以往所有标准,比目前最流行的 H.264/AVC 提高 1 倍.虽然 HEVC 仍然属于基于块的混合编码框架,但是其各个编码阶段都有了增强和改进,其中最重要的变化就是采用了更加灵活的编码结构和高级的编码工具,这也导致 HEVC 的编码模式搜索空间非常大.为了保证编码质量,编码器要进行大量的计算以寻找率失真代价较小的编码模式,即模式选择(mode decision, MD),编码复杂度非常高.随着多核和众核处理器的发展[3],在并行计算平台上并行化 HEVC 编码将是满足其计算能力需求实现实时编码的有效手段[4-12].与以往所有标准不同,HEVC 标准本身就采纳了多种便于并行编解码的工具,如 WPP(wavefront parallel processing)[9], Tiles[10], MER (motion estimation region)[11],可见并行计算对于 HEVC 及今后视频编码领域的重要性.

目前 HEVC 并行相关的研究工作主要集中于计算量较大的帧间 MD 尤其是运动估计(motion estimation, ME)模块[7-8,11],对帧内 MD 并行的研究则相对较少.随着帧间 MD 的并行化,帧内 MD 逐渐成为了速度瓶颈.在文献[13]中,在帧间 MD 被加速了近 15 倍之后,I 帧的平均编码时间大大超过了 P/B 帧的平均编码时间,是其 4~5 倍,限制了编码器的整体效率,因此对帧内 MD 的并行加速同样重要.

现有的对 HEVC 帧内 MD 并行的研究较少,而且算法的并行度不够高[14-16],HEVC 标准支持的 Tiles 和 WPP 也未能在并行度和编码质量取得较好的平衡.针对这些问题,本文提出了一种在编码树单元(coding tree unit, CTU)内使用的并行帧内 MD 算法:本文对帧内 MD 过程进行了子任务划分,深入分析并解除了各个子任务在相邻编码块之间存在的数据依赖性,包括帧内预测参考像素依赖、PU 预测模式依赖、熵编码概率模型依赖和概率建模依赖,实现了每个子任务对整个 CTU 里面多层次细粒度的编码块并行处理,最终实现了 CTU 内并行帧内 MD.

1 背景及研究现状

本节介绍与本文工作相关的一些背景.首先介绍 HEVC 帧内模式选择基本概念,然后对已有的一些并行帧内模式选择算法进行了介绍和分析.

1.1 HEVC 帧内模式选择

在 HEVC 编码标准中,一帧视频图像被均匀地划分成 CTU,CTU 的大小可以为 64×64,32×32 或 16×16,典型且不失一般性,本文以下默认 CTU 大小为 64×64.如图 1 所示,每一个 CTU 四叉树递归地划分为 4 个相同大小的子单元,该四叉树的每一个叶子节点叫作一个编码单元(coding unit, CU);每个 CU 也会采用四叉树递归划分,每一个叶子节点叫作变换单元(transform unit, TU).此外,从图 1 可见,每个 CU 有多种预测单元(prediction unit, PU)划分模式,能更灵活地进行预测编码. HEVC 的帧内预测模式也比 H.264 复杂很多,共 35 种模式.CTU,CU,PU,TU 包含的单一亮度或色度分量信息分别记为 CTB(coding tree block),CB(coding block),PB(prediction block),TB(transform block).

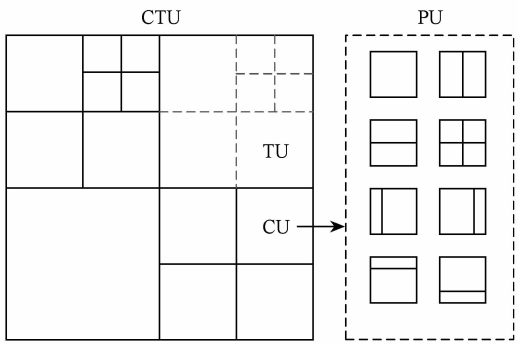

Fig. 1 Flexible coding structures in HEVC.

图 1 HEVC 标准中灵活的编码结构

在 HEVC 帧内 MD 过程中,对于一个 CTU 来说,它可以采用四叉树递归划分的方式划分出更小的 CU,最小为 8×8.对于 8×8 CU 有 $2N\times 2N$ 和 $N\times N$ 两种 PU 划分方式,其他 CU 只有 $2N\times 2N$ 的 PU 划分.$N\times N$ 划分情况下包含 4 个亮度 PB 和 1 个色度 PB,$2N\times 2N$ 划分时包含 1 个亮度 PB 和 1 个色度 PB,每个 PB 有一个预测模式.对一个帧内编码 CU 来说,色度 PB 和亮度 PB 的预测模式可以是不同的,TU 划分模式对亮度和色度是相同的,因此一个帧内编码 CU 的模式可由亮度 PB 预测模式、色度 PB 预测模式和 TU 划分模式来表征,如式(1)所示.帧内 MD 即对 CTU 里面的每个 CU 从式(1)

所表示的模式空间中寻找率失真代价最小的模式组合,并且据此决策出整个 CTU 的最佳 CU 四叉树划分方式.在 HEVC 的参考软件 HM 中,一个 CTU 里面 CU 的 MD 计算是按深度优先遍历顺序进行的,如图 2 所示,为简便起见只画了 3 层,每个节点表示一个 CU.

CU 模式:=(亮度 PB 预测模式,色度 PB 预测模式,TU 划分模式).　(1)

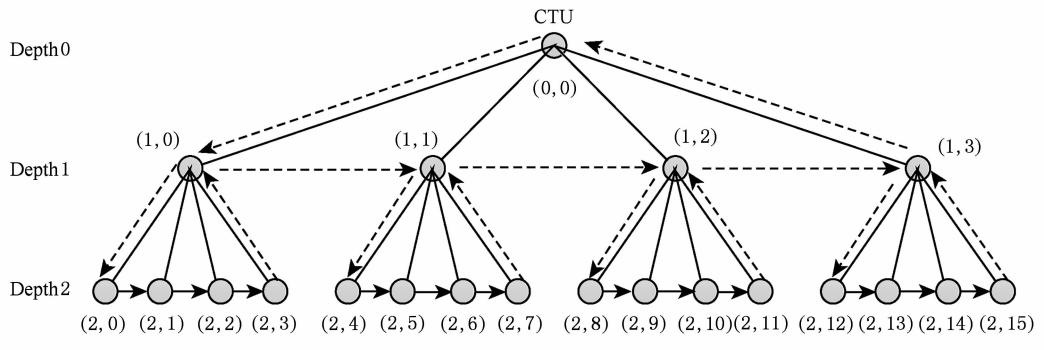

Fig. 2　Depth-first serial processing order of CUs in a CTU of HM.

图 2　HM 中一个 CTU 里 CU 的深度优先串行处理顺序

1.2　相关研究工作

由于采用了灵活的编码结构和复杂的预测模式,帧内 MD 计算量非常大.为了加快帧内 MD,许多研究工作[17-19]提出了快速算法.快速算法能在一定程度上减小计算复杂度,但是获得的加速比有限,距离实时编码仍然有很大的差距.

并行计算是提高 HEVC 编码速度的有效手段,标准本身采纳了几种粗粒度的并行化工具.Tiles 将一帧图像纵横划分成若干可以独立进行编码的子图像,子图像之间无依赖关系,所以可以并行处理.Tiles 划分对并行编码比较容易实现且并行度比较灵活,缺点是对编码效率影响较大,经测试[20],将 1080p 的视频均匀划分成 6×3 的 Tiles 用于帧内 MD,在 intra_main 配置下实际平均加速比为 12,BDBR[21]达到 5%左右.HEVC 采用的另一种并行方案为 WPP,由于其尽可能地保持了数据依赖关系,所以对编码质量影响较小,但是并行度却不高.对一个宽度为 W 个 CTU、高度为 H 个 CTU 的视频帧来说,如果满足条件 $2(H-\alpha-1)<W\leqslant 2(H-\alpha)$,其中 $\alpha\in\mathbb{N}$ 且 $1\leqslant\alpha<H$,其理论并行度(theoretical parallelism degree, TPD)为

$$TPD(W,H,\alpha)=\frac{WH}{2W+2\alpha-2},\qquad(2)$$

对于 720p 和 1080p 的视频序列满足 $\alpha=2$,所以 1080p 的视频 TPD 只能到 8[22].

在另外一些并行帧内 MD 的研究中,文献[14-15]都使用前向无环图描述 CTU 之间的依赖关系,实现了 CTU 之间的并行处理,但本质仍属于 WPP,文献[14]平均获得了 5 倍的加速比,文献[15]则使用分类器决策出最佳 CTU 的大小,通过较小的 CTU 大小能获得较高的加速比,平均达到 10 倍.在文献[16]中,该文作者提出了一种在 TU 四叉树划分时 4 个子节点并行帧内预测的算法,但是由于帧内预测仍依赖于重构像素,理论并行度只能达到 4.

从以上内容可以看出,现有的帧内 MD 并行算法主要是粗粒度并行(Tiles,WPP),文献[16]属于细粒度并行但是并行度不高.针对它们存在的问题,本文提出了一种在 CTU 内使用的多层次细粒度的并行帧内 MD 算法,在保证编码质量的前提下获得了更高的并行度.多层次体现在不同深度的 CU,PU,TU 可以并行处理,细粒度则体现在最小计算单元为一个 TB.

2　多层次细粒度并行帧内模式选择

本节提出了一种在 CTU 内使用的多层次细粒度并行帧内 MD 算法.本文首先对 CU 帧内 MD 过程进行了子任务划分,深入分析并解除了各个子任务在相邻 CU,PU,TU 间存在的数据依赖关系,实现了各个子任务在整个 CTU 范围的并行处理,最终实现了各个层次的所有 CU 并行模式选择.

2.1　子任务划分

对于一个 CU 的模式选择,为了减小计算量,HM 对式(1)表示的模式空间的搜索分阶段进行,先选择亮度 PB 预测模式,然后选择 TU 划分模式,再选择色度 PB 的预测模式,如图 3 所示.

对于亮度 PB 预测模式选择,先对所有 35 种模式进行代价粗算(rough mode cost computation, RMCC),代价记作 RMC,包括 PB 预测值与原始值

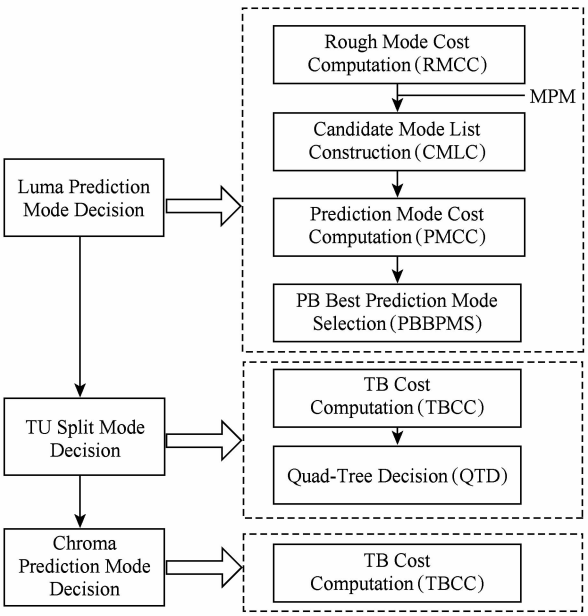

Fig. 3 The flowchart of intra MD for a CU in HM encoder.

图 3 HM 编码器中对一个 CU 的帧内 MD 流程图

之间的残差变换绝对值(sum of absolute transformed differences, SATD)和预测模式编码位数,从中选择一定数量 RMC 最小的预测模式,与当前 PB 的最可能预测模式(most probable mode, MPM)一起构造候选模式列表(candidate mode list construction, CMLC),记作 CML,对 CML 里的每一个模式都去计算率失真代价(prediction mode cost computation, PMCC),代价记作 PMC,选择 PMC 最小的预测模式作为当前 PB 的最佳预测模式(PB best prediction mode selection, PBBPMS),记作 PBBPM. 然后对 TU 划分模式进行决策,主要计算量是亮度 TB 的率失真代价计算(TB cost computation, TBCC),代价记作 TBC,使用 TBC 进行 TU 四叉树决策(quad-tree decision, QTD). 最后选择色度预测模式,主要计算是色度 TB 的 TBCC 计算. 为便于后续引用,在表 1 对上述子任务以及相应的计算结果进行了归纳.

Table 1 Sub-Tasks and Their Output Results

表 1 子任务划分和相应的输出结果

Name of Sub-Tasks	Abbreviation	Result
Rough Mode Cost Computation	RMCC	RMC
Candidate Mode List Construction	CMLC	CML
Prediction Mode Cost Computation	PMCC	PMC
PB Best Prediction Mode Selection	PBBPMS	PBBPM
TB Cost Computation	TBCC	TBC
Quad-Tree Decision	QTD	Complete MD

如图 2 所示,在串行帧内 MD 时,由于 CU 的 MD 计算是按照深度优先遍历顺序去串行进行的,所以对于表 1 中的每个子任务,在串行 MD 时它们在 CTU 范围内的执行路径也是按四叉树深度优先遍历顺序去进行的. 本文提出的并行算法将每个子任务在 CTU 范围内进行多层次细粒度的并行,每个子任务都能并行处理一个 CTU 里面的所有 CU, PU 或 TU,即并行处理四叉树中的所有节点,一个节点对于不同的子任务表示一个 CU,PU 或 TU.

2.2 数据依赖性分析与消除

对于每个子任务,要在一个 CTU 范围内进行并行 MD 计算,而相邻块之间有多种数据依赖关系,这些数据依赖会阻碍子任务在 CTU 范围内的并行计算. 经过本文的分析,有 4 种数据依赖需要解除:

1) 帧内预测时重构像素依赖. 此依赖关系出现在帧内预测时,如图 4 所示,在一个 PB 或 TB 进行代价计算时,需要进行帧内预测,即参考相邻块已经重构出来的像素对自身进行预测. 对于一个 $M\times M$ 的 TB 来说,需要参考周围的 $4M+1$ 个重构像素,分别来自于其左、上、左下、右上和左上方向已经编码重构完成的相邻图像区域.

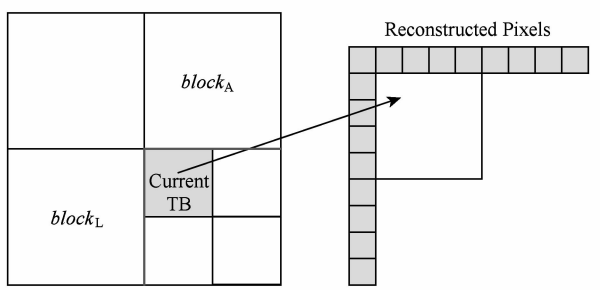

Fig. 4 Reconstructed pixels dependencies during intra prediction.

图 4 帧内预测时重构像素依赖

在本文的并行算法中,RMCC, PMCC, TBCC 在一个 CTU 里面要对所有 PB 或 TB 并行进行代价计算,需要进行帧内预测,如果一个 PB 或 TB 要参考的重构像素跟它位于同一个 CTU,那么重构像素是不可用的,因为相邻块也同时在进行模式选择,并未重构完成. 如图 4 所示, $block_L$, $block_A$ 和当前块位于同一个 CTU 进行并行处理,那么当前块所依赖的 $block_L$ 和 $block_A$ 的重构像素不可用. 要想并行计算,帧内预测参考像素的依赖关系必须要解除.

为了解除这种相关性,本文提出使用原始像素代替重构像素进行帧内预测. 为了更准确地模拟实际的编码过程,在对一个 PB 或 TB 进行参考像素构造

的过程中,虽然所有原始像素都是可用的,但依然按照标准遵循 Z 扫描顺序来决定某一个像素是不是可用,对于不可用的像素则调用替换(substitution)过程去生成.另外,对参考像素的滤波操作也同样按照标准进行.由于原始像素是始终可用的,相邻 PB 或 TB 之间不会再有重构像素的依赖问题,所以同一个 CTU 里面所有 PB 和 TB 都可以并行地进行帧内预测.

2) 编码预测模式时 MPM 计算依赖.为了提高压缩效率,HEVC 在编码帧内预测模式时需要参考相邻(左边和上边)PB 的预测模式,构造出一个长度固定为 3 的 MPM 列表,如图 5(a)所示.如果左 PB(left PB,LPB)和上 PB(above PB, APB)的预测模式不可用或者相同,则还会加入 DC、planar、垂直、水平等模式.令(x_c, y_c)表示当前 PB 的左上角在当前图像帧中的坐标,LPB 定义为覆盖点 $L(x_c-1, y_c)$的 PB,APB 定义为覆盖点 $A(x_c, y_c-1)$的 PB.

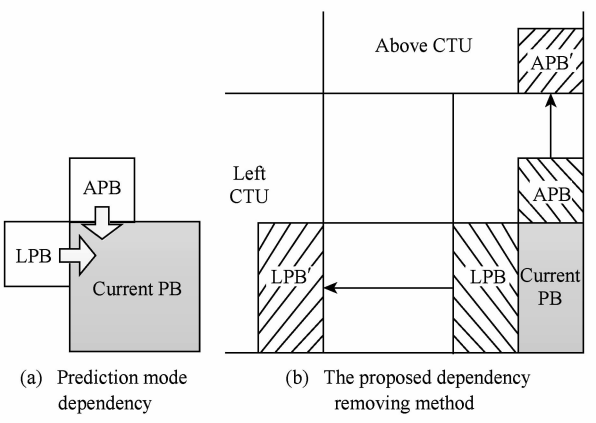

Fig. 5　Prediction mode dependency among adjacent PUs and the proposed dependency removing method.

图 5　相邻 PU 预测模式依赖和本文提出的依赖性消除方法

在本文的并行算法中,RMCC,CMLC,PMCC 要对 CTU 内所有 PB 并行地进行 MPM 计算,如果当前 PB 与参考的 LPB 或 APB 位于同一个 CTU 里面,即如式(3)所示,其中(x_n, y_n)对于 LPB 或 APB 分别为(x_c-1, y_c)或(x_c, y_c-1),那么 LPB 和 APB 的预测模式是不可用的,因为它们也同时在进行模式选择,预测模式还未得到.

为了解除此依赖关系,如果当前 PB 和 LPB 或 APB 位于同一个 CTU,那么本文使用已经编码过的 CTU 里面距离 LPB 和 APB 最近的对应 PB 来代替它,记作 LPB′和 APB′,如图 5(b)所示,用 LPB′和 APB′的预测模式代替 LPB 和 APB 的预测模式去构造 MPM. LPB′定义为覆盖点 $L'(x_c-x_c\%\ 64-1, y_c)$的 PB,APB′定义为覆盖点 $A'(x_c, y_c-y_c\%\ 64-1)$的 PB,其中"%"表示取余运算,64 表示 CTU 大小.由于 CTU 是按照扫描顺序进行编码的,所以左 CTU 和上 CTU 里面的信息一定是可以使用的,这样,同一个 CTU 里面 PU 的预测模式依赖关系就被解除了,可以并行进行预测以及代价计算:

$$\frac{x_n}{64}==\frac{x_c}{64}\ \&\&\ \frac{y_n}{64}==\frac{y_c}{64}. \tag{3}$$

3) 概率模型(context model,CM)继承依赖.HEVC 中使用上下文自适应的二进制算术编码(context adaptive binary arithmetic coding, CABAC)进行语法元素的熵编码.CABAC 的主要过程包括语法元素的二进制化、概率建模、算术编码和 CM 更新.为了提高编码效率,在编码的过程中 CM 会自适应动态更新,以更好地反映图像的局部区域特性,获得更高的压缩比.在 HM 模式选择过程中,熵编码器会使用 CM 去估计编码产生的位数以计算编码代价,CM 是模拟实际编码过程动态更新的,Z 扫描顺序更小的块的 MD 完成之后会将 CM 传递给 Z 扫描顺序大于它的块使用,如图 6(a)所示,TU1 使用的 CM 是 TU0 计算之后的结果,这样就在相邻块之间产生了 CM 的继承依赖.

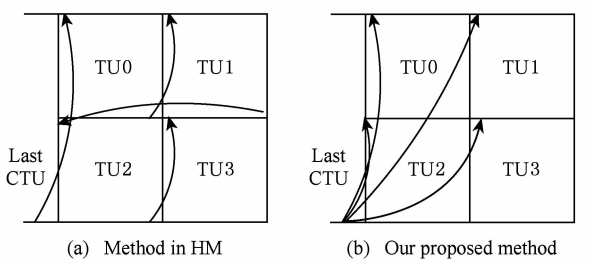

Fig. 6　CMs inheritance in HM and our proposed method.

图 6　HM 中的 CM 继承依赖和本文提出的方法

在本文的并行算法中,RMCC,PMCC,TBCC,QTD 任务要并行地对一个 CTU 内的所有 CU,PU 或 TU 进行代价计算,而由于相邻 CU,PU,TU 之间存在 CM 的继承依赖问题而导致无法并行,要想实现并行代价计算,必须要解决此依赖关系.

为了解除同一个 CTU 内 CM 继承依赖,本文提出以下解决方法:同一个 CTU 内的所有 CU/PU/TU 使用同一套 CM,该 CM 来自于上一个已编码的 CTU 经过训练之后的结果.如图 6(b)所示,为了简洁,一个 CTU 内只画出 4 个 TU.通过这种 CM 继承方式,一个 CTU 里面所有的 CU/PU/TU 都有了自己的 CM,所以可以并行处理.

4) 概率建模依赖. 为了提高编码效率,CABAC编码二进制符号(binary symbol, bin)时需要进行概率模型选择(也叫概率建模),概率模型即当前bin是0/1的概率,概率建模即从可选的概率模型里选择出最能反映当前bin概率估计的模型. 为了能得到较精确的概率估计,概率建模会依赖于当前bin序号、编码深度、色度或亮度、周围编码信息等. HEVC在编码语法元素split_cu_flag时,会根据当前CU的深度和其左CU、上CU的划分深度进行概率模型选择,如图7(a)所示. 假设左CU编码深度为$depth_L$,上CU编码深度为$depth_A$,当前CU的深度为$depth$,那么当前CU的split_cu_flag的概率模型序号如式(4)所示,其中"＞"为逻辑运算符. cu_skip_ flag的概率建模与此类似. 在本文的并行算法中,QTD要对CTU内所有CU并行地进行代价计算和模式选择,由于相邻CU之间在概率建模时有编码模式依赖,所以要并行处理,这种依赖关系必须要解除. 为了解除此类数据依赖性,本文采用和解除MPM计算依赖性类似的方法,即如果参考的CU和当前CU位于同一个CTU,则使用已编码的相邻CTU中的对应CU来代替它,如图7(b)所示. 另外,TB的语法元素cbf在概率建模时会依赖当前TB在CU中的划分深度信息,但是在本文的算法中,整个CTU里面的所有TB会并行计算代价TBCC,并无CU概念,所以TB在CU中的深度无法得到. 对于这个依赖关系,本文的解决方法是:对于4×4的TB,深度设置为1,其他TB的深度设置为0.

$$(depth_L > depth) + (depth_A > depth). \quad (4)$$

以上对阻碍并行计算的多种数据依赖关系进行了分析和消除. 需要注意的是本文提出的数据依赖性消除算法只在模式选择阶段使用,为了保证编码结果符合标准且保证编码质量,当一个CTU的模式选择完成之后,即确定了CU划分方式、PU划分方式、PU预测模式和TU划分方式之后,在进行实际熵编码的过程中需要对量化系数和预测模式编码进行修正,即按照标准规定去参考相邻块的重构像素进行帧内预测,对残差重新进行变换和量化得到量化系数,MPM的构造也按标准规定参考相邻的PB,CABAC的概率模型继承和概率建模也按标准规定进行. 这个修正的过程是需要串行计算的.

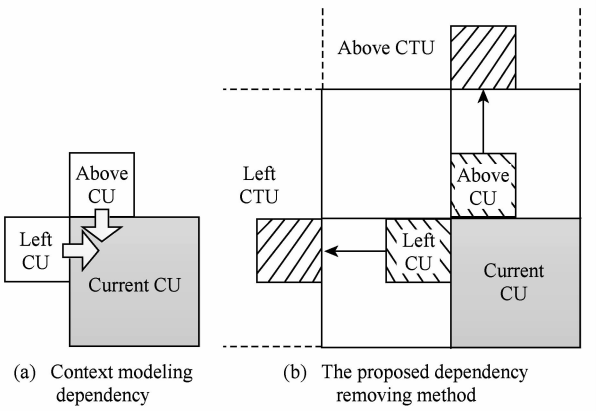

Fig. 7 Coding depth dependency among adjacent CUs and our proposed dependency removing method.

图7 CU之间依赖和提出的依赖性消除方法

2.3 并行策略

在解除了以上多种数据依赖关系之后,下面给出本文提出的多层次细粒度并行算法的并行策略,为了便于表达,进行如下符号定义:

CU,PU,TU分别表示在一个CTU里面的编码单元、预测单元、变换单元;CB,PB,TB分别表示一个CU,PU,TU包含的亮度或色度分量块;$depth_i$表示当前单元(CU,PU,TU)或分量块(CB,PB,TB)相对于CTU的四叉树划分深度,其中$0 \leqslant i \leqslant 4$;$zorder_j$表示第$depth_i$层深度的某一个单元的Z扫描顺序序号,其中$0 \leqslant j \leqslant 4^i - 1$;$mode_k$表示一个帧内预测模式,其中$0 \leqslant k \leqslant 34$.

对表1中的各子任务,分别在CTU内进行如下数据级划分和并行:

1) RMCC. 对于亮度PB的RMCC任务,主要工作是计算PB的帧内预测和SATD的计算. 本文提出的并行策略:一个CTU里面所有的亮度PB的所有预测模式都并行进行RMCC,表示为$RMC_{i,j,k} = RMCC(PB(depth_i, zorder_j), mode_k)$,其中$0 \leqslant i \leqslant 4$,$0 \leqslant j \leqslant 4^i - 1$,$0 \leqslant k \leqslant 34$,$i,j,k$并行.

2) CMLC. 在一个亮度PB的RMCC完成之后,需要从中选出一定数量RMC最小的模式,然后与当前PB的MPM一起组成该PB的候选模式列表CML. 本文提出的并行策略:CTU内所有亮度PB并行进行CMLC操作,表示为$CML_{i,j} = CMLC(PB(depth_i, zorder_j))$,其中$0 \leqslant i \leqslant 4$,$0 \leqslant j \leqslant 4^i - 1$,$i,j$并行.

3) PMCC. 当一个亮度PB的CML构造之后,要从中选出一个最好的模式作为当前PB的预测模式. 当前PB在其CML里面的每一个模式都计算出一个率失真代价PMC,为了减少计算量,只有64×64的PB进行一层TB划分,其他的PB都直接作为一个TB计算. 本文提出的并行策略:CTU里面所有的PB在其CML里面的所有模式并行,表示为

$PMC_{i,j,k}=PMCC(PB(depth_i,zorder_j),mode_k)$, $0\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1, k\in CML_{i,j}$, i,j,k 并行,其中对于 $PB(depth_0,zorder_0)$,划分为 4 个子块并行计算.

4) PBBPMS. 当一个亮度 PB 的 PMC 计算完成之后,需要从中选择最小 PMC 对应的预测模式作为 PB 的最终预测模式,记为 PBBPM. 本文提出的并行策略:CTU 内所有亮度 PB 并行进行 PBBPMS 操作,记为 $PBBPM_{i,j}=PBBPMS(PB(depth_i,zorder_j))$, $0\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1$, i,j 并行.

5) TBCC. 选出亮度 PB 最佳预测模式之后,需要决策每个 CU 的最佳 TU 划分,因此要计算一个 CU 所包含的所有 TB 的率失真代价. 本文提出的并行策略:整个 CTU 里面所有 TB 并行地进行率失真代价 TBC 计算,表示为 $TBC_{i,j,k}=TBCC(TB(depth_i,zorder_j),mode_k)$,由于亮度 TB 最大为 32,所以 $1\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1, k\in TBPM_{i,j}$, i,j,k 并行,其中 $TBPM_{i,j}$ 表示 $TB(depth_i,zorder_j)$ 所需计算的预测模式. 由于本文使用原始像素取代重构像素来进行帧内预测,所以任何一个 TB 对于同一个预测模式其预测值是完全相同的. 因为同一个 TB 可能会被多个 PB 同时覆盖,所以同一个 TB 在一个模式下只需要计算一次即可,减少了计算量. 假如最大 TU 划分层次为 3 层,那么 $TB(depth_i,zorder_j)$ 会被 $PB(depth_i,zorder_j)$, $PB(depth_{i-1},zorder_{j/4})$ 和 $PB(depth_{i-2},zorder_{j/16})$ 同时覆盖,故 $TBPM_{i,j}$ 是它们最佳模式的并集,即:

$$TBPM_{i,j}=\{PBBPM_{i,j},PBBPM_{i-1,j/4},PBBPM_{i-2,j/16}\}. \tag{5}$$

对于色度 PB 的预测模式选择,色度 PB 在亮度 PB 预测模式一定的情况下有 5 种预测模式需要计算,如表 2 所示. 本文直接将这 5 种模式全部并行计算:$TBC_{i,j,l}=TBCC(TB(depth_i,zorder_j),mode_l)$,其中 $1\leqslant i\leqslant 3, 0\leqslant j\leqslant 4^i-1, l=0,1,2,3,4$, $mode_l$ 表示需要计算的预测模式,i,j,l 并行.

Table 2　The Relationship of Prediction Mode Between Luma and Chroma Components in a CU

表 2　色度分量预测模式与亮度分量预测模式的对应关系

Chroma Mode Index	Luma Prediction Mode				
	0	26	10	1	X(Others)
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

6) QTD. 此任务的主要工作是根据前面所述任务的计算结果,决策出 CU 的最终模式. 因为在 PBBPMS 中已经得到了亮度 PB 的预测模式,在 TBCC 中已经得到了亮度和色度的 TBC,所以 QTD 的主要工作就是决策出当前 CU 最优的 TU 划分和最优色度预测模式,并且计算出当前 CU 的率失真代价,完成模式选择过程. 本文提出的并行策略:一个 CTU 里面的所有 CU 并行进行 QTD:$QTD(CU(depth_i,zorder_j))$,其中 $0\leqslant i\leqslant 3, 0\leqslant j\leqslant 4^i-1$, i,j 并行.

2.4　理论并行度分析

为了对所提并行帧内 MD 算法的理论并行度进行近似分析,进行 2 点假设:1) 对于 RMCC, PMCC, TBCC 子任务,主要计算量是帧内预测和变换量化,它们的计算时间与计算单元的面积成正比,且与预测模式无关; CMLC, PBBPMS, QTD 的主要操作是结果的比较和选择,计算量较小且与计算单元面积无关. 2)对于同一类子任务由于解除了相邻数据单元之间的数据相关性,所有的数据单元都可以同时被计算,因此理论并行度由整个CTU 串行计算时间和计算量最大的数据单元的计算时间决定. 基于以上 2 点假设,各子任务分析如下:

1) RMCC 阶段. 令 $CT_{i,j,k}$ 表示 $RMCC(PB(depth_i,zorder_j),mode_k)$ 的计算时间,令 $T=CT_{0,0,0}$,那么整个 CTU 里所有 PB 的 RMCC 串行计算时间近似为

$$\sum_{i=0}^{4}\sum_{j=0}^{4^i-1}\sum_{k=0}^{34}CT_{i,j,k}=35\sum_{i=0}^{4}\sum_{j=0}^{4^i-1}CT_{i,j,0}=35\sum_{i=0}^{4}4^i\times CT_{i,0,0}=175T,$$

最大计算量任务的计算时间为 T,所以 $TPD=175$.

2) PMCC 阶段. 令 $CT_{i,j,k}$ 表示 $PMCC(PB(depth_i,zorder_j),mode_k)$ 的计算时间,令 $T=CT_{1,0,0}$,经统计亮度 PB 的 CML 里面平均含有 5 个模式,所以整个 CTU 里所有 PB 的 PMCC 的串行计算时间近似为

$$5\left(4T+\sum_{i=1}^{4}\sum_{j=0}^{4^i-1}CT_{i,j,0}\right)=5\left(4T+\sum_{i=1}^{4}CT_{i,0,0}\right)=100T,$$

最大计算量任务的计算时间为 T,所以 $TPD=100$.

3) TBCC 阶段. 令 $CT_{i,j,k}$ 表示 $TBCC(TB(depth_i, zorder_j), mode_k)$ 的计算时间,令 $T=CT_{1,0,0}$,假设 TU 最多划分 3 层,在串行 MD 过程中整个 CTU 里所有 TB 的 TBCC 串行计算时间为

$$2\times 4T+3\left(\sum_{i=1}^{3}\sum_{j=0}^{4^i-1}CT_{i,j,0}\right)+4T=44T,$$

最大计算量任务的计算时间为 T,所以 $TPD=44$.

4) CMLC 任务和 PBBPMS 的计算量较小且与计算单元大小无关,TPD 为整个 CTU 里面所有的亮度 PB 数量,即 $\sum_{i=0}^{4}4^i=341$,同理 QTD 的 TPD 为 CTU 中 CU 数量,即 $\sum_{i=0}^{3}4^i=85$.

从以上分析可以看出,本文提出的并行帧内 MD 算法的理论并行度很高,适用于众核平台.

3 实现和实验结果

基于 HEVC 参考代码 HM13.0[23],本文使用多线程实现本文所提出的并行模式选择算法. 本文修改了代码中模式选择模块(TEncCu::xCompressCU 及相关函数)使之并行化,其他的模块保持不变. 本文重点是提出一种适用于众核平台的高并行度算法,主要关注算法的并行度和对编码质量的影响,所以并未对单核性能进行指令、内存等任何优化. 实验采用的计算平台为 TILE-Gx36 众核平台,共有 36 个处理核心,单核主频 1.2 GHz.

3.1 实现方法

为了减少频繁创建和销毁线程带来的开销,本文使用线程池实现本文算法. 在编码器刚启动时创建一定数量所需线程,线程池中所有的线程在编码器整个生命周期内一直存在. 为了减小线程切换带来的性能影响和增加实验的可重现性,所有的线程都绑定在一个固定的核上且每个核只绑定一个线程. 0 号核不予使用,1 号核绑定主线程,其余 34 个核用于线程池里的工作线程. 线程池中维护一个任务链表,所有需要处理的任务都插入链表,空闲的工作线程自动从链表首取走任务进行处理. 主线程的主要工作就是往链表里面放任务,实际去完成任务计算的是工作线程.

在本文的实现中,定义了 6 种类型的任务:RMCC, CMLC, PMCC, PBBPMS, TBCC, QTD,由于这些任务之间对于同一个数据单元存在先后处理顺序的依赖关系,所以本文定义了任务的优先级,如表 3 所示. 优先级越大的任务表示越应该优先被计算,因为其他类型的任务可能会使用它的计算结果,在放入任务链表时会根据优先级大小顺序存放,优先级越大的任务在任务链表中排序越靠前. 另外,为了保证块越大的任务先被计算,在每一种任务类型里面,块越大的任务即深度越小的任务优先级越大. 对于不同数据单元,计算任务之间则没有依赖关系,不同种类的子任务之间可以并行.

Table 3 Sub-Tasks with Their Parallel Hierarchy and Priority

表 3 子任务并行层次和优先级定义

Sub-task	Parallel Hierarchy	Priority
RMCC	$0\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1, 0\leqslant k\leqslant 34$	6
CMLC	$0\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1$	5
PMCC	$0\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1, k\in CML_{i,j}$	4
PBBPMS	$0\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1$	3
TBCC	Luma: $1\leqslant i\leqslant 4, 0\leqslant j\leqslant 4^i-1, k\in TBPM_{i,j}$ Chroma: $1\leqslant i\leqslant 3, 0\leqslant j\leqslant 4^i-1, l=0,1,2,3,4$	2
QTD	$0\leqslant i\leqslant 3, 0\leqslant j\leqslant 4^i-1$	1

3.2 实验结果

为了验证本文提出的并行帧内 MD 算法的有效性,包括编码加速比和对编码质量的影响,本文用基准编码器 HM 和本文并行化的编码器分别对 6 个 1080p 和 2 个 1600p 的高分辨率视频使用 HM 提供的配置文件 intra_main 进行了编码,每个序列使用了前 100 帧,根据文献[20]对每个序列分别使用 4 个不同的量化参数(quantization parameter, QP):22, 27, 32, 37. 加速比按照式(6)计算,其中 Enc_Time_r 和 Enc_Time_p 分别表示基准编码器 HM 和并行编码器的编码时间,编码效率的下降使用 BDBR[21]来衡量. 为了验证并行算法的可伸缩性即加速比与核数的关系,对每个序列都使用了不同数量的工作线程进行编码. 实验结果如表 4 所示,其中的核数是用于工作线程的核数,加速比是对 4 个 QP 取平均的结果. 从表 4 可以看出,本文提出的并行帧内模式选择算法在使用 34 个核时对所有序列平均能获得 18 以上的加速比,平均 BDBR 为 3.04%. 部分序列的率失真曲线如图 8 所示.

$$Speedup=\frac{Enc\_Time_r}{Enc\_Time_p}. \tag{6}$$

Table 4　Detailed Experimental Results, *Speedup* and BDBR Included

表 4　加速比和编码质量实验结果

Class	Sequence	*Speedup* with Different Number of Cores							BDBR/%
		2	4	8	16	24	32	34	
1080p	BasketballDrive	1.77	3.51	6.95	12.79	16.77	18.63	18.63	+3.62
	BQTerrace	1.88	3.76	7.20	13.25	17.42	19.15	19.84	+3.05
	Cactus	1.86	3.71	7.19	13.17	17.25	19.04	19.59	+2.96
	Kimono	1.80	3.53	6.96	12.79	16.96	18.66	19.00	+3.09
	ParkScene	1.85	3.66	7.08	13.00	17.16	18.85	19.12	+2.34
	Tennis	1.80	3.56	7.02	12.88	17.19	18.67	19.05	+2.93
1600p	PeopleOnStreet	1.75	3.40	6.89	12.71	16.73	18.36	18.42	+3.12
	Traffic	1.73	3.42	6.87	12.67	16.59	18.11	18.21	+3.21
Average		1.80	3.58	7.02	12.91	17.01	18.68	18.98	+3.04

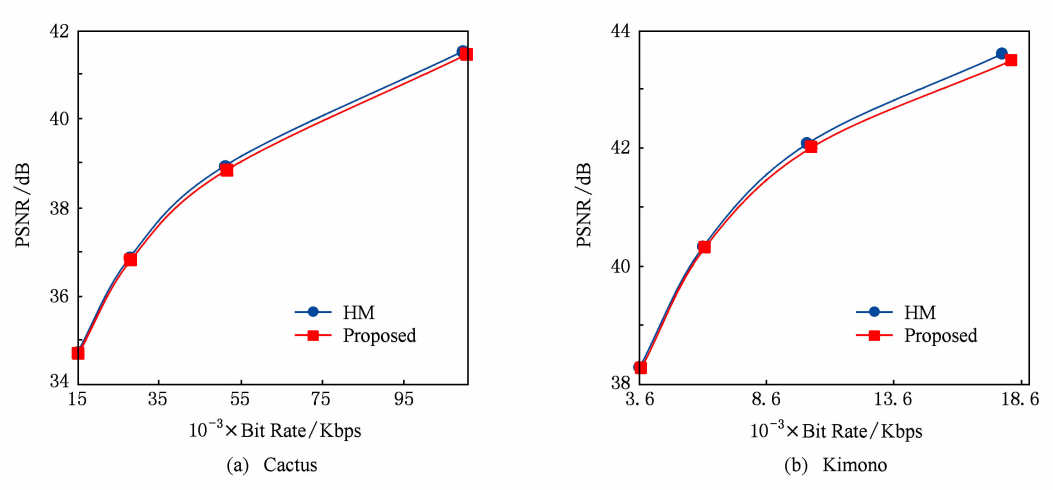

Fig. 8　Rate distortion curves for some test sequences.

图 8　部分序列的率失真曲线图

图 9 给出使用不同数量的核数(2,4,8,16,24,32,34)得到的所有序列的平均加速比与核数之间的关系曲线图,并且与 1080p 和 1600p 的 WPP 的理论值进行了对比. 从图 9 可以看出,本文所提出的算法在核数增多时有更高的加速比,适用于众核计算平台;当核数较少时加速效果比较明显,但是随后曲线逐渐趋于平坦,且加速比与核数之比远小于 1,离算法的 *TPD* 有较大的差距. 造成这种现象的原因可能有 3 点:

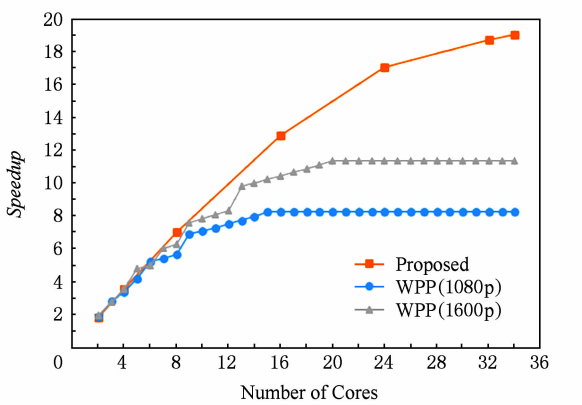

Fig. 9　*Speedup* vs number of cores for our proposed method and WPP (*TPD*).

图 9　算法加速比和 WPP(*TPD*)与核数关系

1) 并行计算数据单元划分不均衡. 在本文的算法中,不同大小的 PU 和 TU 计算量相差较大,如果假设计算量与块面积成正比,那么第 0 层 TU 的 TBCC 计算量近似是第 1 层 TU 的 TBCC 的 4 倍. 任务划分不均衡会导致处理器核的计算负载不均衡,造成 CPU 核的饥饿等待,影响并行度.

2) 由于所有的并行任务都放在一个公共链表里面,所有线程都会竞争此资源以取得任务. 随着核数的增加,线程之间在取任务和放任务的竞争会越来越激烈,势必会造成很大的线程同步开销.

3) 算法只进行了模式选择计算的并行化,其他的部分仍然串行执行,虽然占的比例很小,但是会对整体的并行度造成上限制约.另外,编码阶段的修正过程也是串行执行,也会影响实际的加速比.

对于编码质量的下降,主要原因是代价计算的准确性不高,这是本文所采取的多种数据依赖性消除算法的综合作用结果.

由于本文提出的并行算法是在局部区域 CTU 内进行,所以可以和其他全局粗粒度并行方法如 WPP 和 Tiles 结合使用,能进一步提高并行度.

4 总 结

本文提出了一种在 CTU 内使用的多层次细粒度的并行 HEVC 帧内模式选择算法,一个 CTU 内所有不同层次的 CU,PU,TU 以及所有预测模式可以并行地进行代价计算和模式选择.本文分析了存在的多种数据依赖性并提出了有效的依赖性解除方法.实验结果表明本文提出的方法能达到 18 倍以上的编码加速比并且编码质量损失较小,比传统的粗粒度并行方案更能适用于众核平台.

参 考 文 献

[1] Bross B, Han W J, Ohm J R, et al. JCTVC-L1003: High efficiency video coding (HEVC) text specification draft 10 (for fdis & last call)[C/OL]. 2013 [2014-11-01]. http://phenix.int-evry.fr/jct/doc_end_user/documents/12_Geneva/wg11/JCTVC-L1003-v34.zip

[2] Sullivan G J, Ohm J R, Han W J, et al. Overview of the high efficiency video coding (HEVC) standard [J]. IEEE Trans on Circuits and Systems for Video Technology, 2012, 22(12): 649-1668

[3] Peng Xiaoming, Guo Haoran, Pang Jianmin. Multi-core processor-technology, tendency and challenge [J]. Computer Science, 2012, 39(11A): 320-326 (in Chinese)
(彭晓明, 郭浩然, 庞建民. 多核处理器——技术、趋势和挑战[J]. 计算机科学, 2012, 39(11A):320-326)

[4] Yan Chenggang. Research on key techniques for parallel video coding on many-core processor [D]. Beijing: Institute of Computing Technology, Chinese Academy of Sciences, 2014 (in Chinese)
(颜成钢. 面向众核处理器的并行视频编码关键技术研究[D]. 北京: 中国科学院计算技术研究所, 2014)

[5] Choi K, Jang E S. Leveraging parallel computing in modern video coding standards [J]. IEEE Multimedia, 2012, 19(3): 7-11

[6] Zhang Yongdong, Yan Chenggang, Dai Feng, et al. Efficient parallel framework for H.264/AVC deblocking filter on many-core platform [J]. IEEE Trans on Multimedia, 2012, 14(1): 510-524

[7] Yan Chenggang, Zhang Yongdong, Dai Feng, et al. Highly parallel framework for HEVC motion estimation on many-core platform [C] //Proc of the 23rd Date Compression Conf (DCC 2013). Piscataway, NJ: IEEE, 2013: 63-72

[8] Yan Chenggang, Zhang Yongdong, Xu Jizheng, et al. Efficient parallel framework for HEVC motion estimation on many-core processors [J]. IEEE Trans on Circuits and Systems for Video Technology, 2014, 24(12): 2077-2089

[9] Clare G, Henry F, Pateux S. JCTVC-F274: Wavefront parallel processing for HEVC encoding and decoding [C/OL]. 2011 [2014-11-01]. http://phenix.int-evry.fr/jct/doc_end_user/documents/6_Torino/wg11/JCTVC-F274-v2.zip

[10] Fuldseth A, Horowitz M, Xu S, et al. JCTVC-F335: Tiles [C/OL]. 2011 [2014-11-01]. http://phenix.int-evry.fr/jct/doc_end_user/documents/6_Torino/wg11/JCTVC-F335-v2.zip

[11] Zhou Minhua. JCTVC-H0082: Configurable and CU-group level parallel merge/skip [C/OL]. 2012[2014-11-01]. http://phenix.int-evry.fr/jct/doc_end_user/documents/8_San%20Jose/wg11/JCTVC-H0082-v2.zip

[12] Chi C C, Mauricio A M, Juurlink B, et al. Parallel scalability and efficiency of HEVC parallelization approaches [J]. IEEE Trans on Circuits and Systems for Video Technology, 2012, 22(12): 1827-1838

[13] Zhang Jun, Dai Feng, Ma Yike, et al. Highly parallel mode decision mehtod for HEVC [C] //Proc of the 30th Picture Coding Symp (PCS2013). Piscataway, NJ: IEEE, 2013: 281-284

[14] Zhao Yanan, Song Li, Wang Xiangwen, et al. Efficient realization of parallel HEVC intra encoding [C]//Proc of 2013 IEEE Int Conf on Multimedia and Expo Workshops (ICMEW2013). Piscataway, NJ: IEEE, 2013: 1-6

[15] Yan Chenggang, Zhang Yongdong, Dai Feng, et al. Efficient parallel HEVC intra-prediction on many-core processor [J]. Electronics Letters, 2014, 50(11): 805-806

[16] Jiang Jie, Guo Baolong, Mo Wei, et al. Block-based parallel intra prediction scheme for HEVC [J]. Journal of Multimedia, 2012, 7(4): 289-294

[17] Choi K, Park S H, Jang E S. JCTVC-F092: Coding tree pruning based CU early termination [C/OL]. 2011[2014-11-01]. http://phenix.int-evry.fr/jct/doc_end_user/documents/6_Torino/wg11/JCTVC-F092-v3.zip

[18] Cho S, Kim M. Fast CU splitting and pruning for suboptimal cu partitioning in HEVC intra coding [J]. IEEE Trans on Circuits and Systems for Video Technology, 2013, 23(9): 1555-1564

[19] Leal da Silva T, da Silva Cruz L, Agostini L V. HEVC intra mode decision acceleration based on tree depth levels relationship [C]//Proc of the 30th Picture Coding Symp (PCS2013). Piscataway, NJ: IEEE, 2013: 277-280

[20] Bossen F. JCTVC-L1100: Common test conditions and software reference configurations [C/OL]. 2013 [2014-11-01]. http://phenix.int-evry.fr/jct/doc_end_user/documents/12_Geneva/wg11/JCTVC-L1100-v1.zip

[21] Bjontegaarda G. VCEG-M33: Calculation of average psnr differences between rd-curves [C/OL]. 2001 [2014-11-01]. https://github.com/gabrieldiego/tg/blob/master/ref/VCEG-M33.doc

[22] Zhang Shaobo, Zhang Xiaoyun, Gao Zhiyong. Implementation and improvement of wavefront parallel processing for HEVC encoding on many-core platform [C] //Proc of 2014 IEEE Int Conf on Multimedia and Expo Workshops (ICMEW2014). Piscataway, NJ: IEEE, 2014: 1-6

[23] JCTVC. HEVC test model: HM13.0 [CP]. [2014-11-01] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-13.0-dev

Zhang Jun, born in 1987. Received his BE degree from Xidian University in 2009 and received his PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences in 2015. His research interests include video coding and image processing.

Dai Feng, born in 1979. Received his MS and PhD degrees from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2008. Associate professor with the Multimedia Computing Group, Advanced Research Laboratory, Institute of Computing Technology, Chinese Academy of Sciences, Beijing. Member of China Computer Federation. His research interests include video coding, video processing and computational photography (fdai@ict.ac.cn).

Ma Yike, born in 1980. Received his MS and PhD degrees in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2011. Member of China Computer Federation. His research interests include computer architecture, parallel algorithm and computational photography (ykma@ict.ac.cn).

Zhang Yongdong, born in 1973. Received his PhD degree in electronic engineering from Tianjin University, Tianjin, China, in 2002. Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. Senior member of China Computer Federation. His research interests include multimedia content analysis and understanding, multimedia content security, video coding, and streaming media technology (zhyd@ict.ac.cn).