基于阵列处理器的 HEVC 数据流图可重构实现

胡传瞻[1] 蒋 林[2] 朱 筠[3] 谢晓燕[1] 杨 坤[3] 崔馨月[3]

[1](西安邮电大学计算机学院 陕西 西安 710121)

[2](西安科技大学集成电路实验室 陕西 西安 710054)

[3](西安邮电大学电子工程学院 陕西 西安 710121)

摘 要 提出一种基于阵列处理器的 HEVC 算法数据流图可重构实现方法。基于动态重构机制完成不同的划分方式、算法间的灵活切换,采用深度优先贪婪对数据流图划分后子任务时域流水的并行方式对 HEVC 中典型编码算法的数据流图重新划分后设计合理映射方案,以 Sobel 算子值为重构依据在阵列处理器上进行帧内预测算法验证。实验结果表明,与块间流水方案实现相比加速比可达 14.97,各算法资源利用率及计算速度均有提升,与帧内预测模式选择快速算法相比每个时钟周期可多处理 7.1 个像素。

关键词 数据流图 HEVC 深度优先贪婪 可重构阵列处理器

中图分类号 TP301.6 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.03.037

RECONFIGURABLE IMPLEMENTATION OF HEVC DATA FLOW GRAPH BASED ON ARRAY PROCESSOR

Hu Chuanzhan[1] Jiang lin[2] Zhu Yun[3] Xie Xiaoyan[1] Yang Kun[3] Cui Xinyue[3]

[1](*School of Computer, Xi'an University of Posts & Telecommunications, Xi'an 710121, Shaanxi, China*)

[2](*Laboratory of Integrated Circuit, Xi'an University of Science and Technology, Xi'an 710054, Shaanxi, China*)

[3](*School of Electronic Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, Shaanxi, China*)

Abstract This paper presents a reconfigurable implementation method of HEVC algorithm data flow graph based on array processor. Based on the dynamic reconstruction mechanism, the flexible switching between different partition methods and algorithms was completed. The data flow graph of typical coding algorithms in HEVC was redivided by depth first greedy parallel mode of time-domain pipelining of sub tasks after data flow graph partition. After that, a reasonable mapping scheme was designed. The intra prediction algorithm was verified on array processor based on Sobel operator value. The experimental results show that compared with the inter block pipelining scheme, the speedup ratio can reach 14.97, and the resource utilization and computing speed of each algorithm are improved. Compared with the fast intra prediction mode selection algorithm, each clock cycle can process 7.1 more pixels.

Keywords Data flow graph HEVC Depth first greedy search Reconfigurable array processor

0 引 言

随着可编程逻辑器件的快速发展,可重构计算成为一种新的计算方式[1],凭借可重构处理器高并行的运行方式,可灵活高效地执行多媒体算法、图像处理、数据压缩等计算密集型应用。因此,在可重构阵列处理器上实现计算密集型的应用,计算性能可以提高数倍。

但高级编程语言编写的应用程序映射到可重构处理器上是一个复杂的过程。在可重构计算的任务实现过程中,首先需要转换优化应用程序的代码以获得数

收稿日期:2021-01-20。国家自然科学基金项目(61772417,61834005,61802304,61602377)。**胡传瞻**,硕士生,主研领域:任务时域划分,视频编解码算法。**蒋林**,教授。**朱筠**,讲师。**谢晓燕**,教授。**杨坤**,硕士生。**崔馨月**,硕士生。

据流图(Data Flow Graph,DFG),然后将 DFG 映射到可重构处理器。但由于硬件资源有限,DFG 通常需要划分为一系列相互依存的子图[2]。传统的任务 DFG 时域划分算法中,簇划分可获得较小的划分块之间的通信成本,但划分块间的边数仍然偏大。国内外许多学者对传统的任务划分算法进行了改进,基于簇的层次敏感划分[3]算法克服了传统簇划分算法机械选取节点划分的缺点,且能使用较少的可重构资源获得较小的通信代价,但任务的数据流图分割产生的边数很多。深度优先贪婪搜索(Depth First Greedy Search Partitioning,DFGSP)算法[4],DFGSP 的局限性划分之后的每个子任务之间存在数据依赖关系,并行度没有达到最大[4],可重构阵列处理器可以解决其并行度低的问题。

HEVC 算法具有数据量大、算法复杂度高的特点。以非常清晰的电影数字视频格式 1080P 为例,每秒生成的数据量为 $1\ 920 \times 1\ 080 \times 3 \times 8 \times 30 = 1.49 \times 109$ bit,针对如此高复杂的计算,仅仅从算法角度进行优化仍然无法满足实时视频编码的需求,所以很多国内外学者已经开始把视频算法向专用硬件上转移[5]。

因此本文结合视频编解码数据量大、循环多、计算密集的特性,基于 DFGSP 算法,将 HEVC 中典型编码算法转换成的 DFG 图进行重新划分,设计并行映射方案。而帧内预测算法中不同预测模式计算公式差别较大,且使用的参考像素位置不同,增加了预测像素计算的复杂度。并将 Sobel 算子值作为重构依据算法放入帧内预测算法中进行重构验证,在视频阵列处理器(DPR-CODEC)[6]上映射实现,使其兼具软件可编程灵活性和硬件实现的高效性,达到提高计算速度和资源利用率目的。

1　HEVC 算法 DFG

在帧内预测过程中,对应的预测编码单元的尺寸包括 4×4、8×8、16×16、32×32 和 64×64,根据图像不同区域纹理度的不同,对编码单元进行划分后用于进行模式选择。本文通过计算 Sobel 算子的梯度值作为块大小划分的依据。并对 35 种预测模式中复杂度较高的 DC 模式构建 DFG 降低其计算复杂度。判断比较不同预测模式 SAD 值,选取最优预测模式。

1.1　Sobel 算子 DFG

索贝尔算子(Sobeloperator)主要用作边缘检测,用来运算图像亮度函数的灰度之近似值。

(1) Sobel 算子水平方向的图像滤波。Sobel 算子通常需要进行水平方向的图像滤波,其水平方向的滤波系数如式(1)所示。

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ 1 & 0 & 2 \end{pmatrix} \times A_x \begin{pmatrix} x_0 & x_1 & x_2 \\ x_3 & x_4 & x_5 \\ x_6 & x_7 & x_8 \end{pmatrix} \tag{1}$$

式中:A_x表示向量系数。

通过分析式(1),在进行算法映射时,不需要乘法器,而仅仅通过移位操作实现,水平方向的数据流图如图 1 所示。

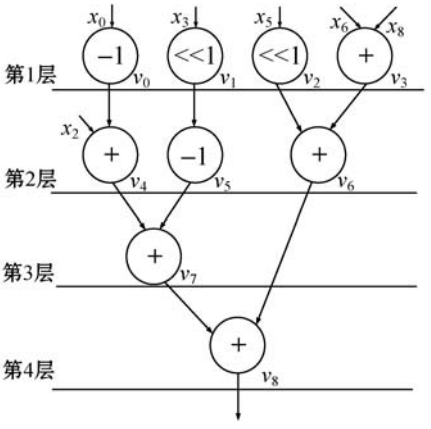


图 1　Sobel 算子水平方向 DFG

(2) Sobel 算子垂直方向的图像滤波。Sobel 算子通常需要垂直方向的图像滤波,其垂直方向的滤波系数如式(2)所示。

$$G_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \times A_x \begin{pmatrix} x_0 & x_1 & x_2 \\ x_3 & x_4 & x_5 \\ x_6 & x_7 & x_8 \end{pmatrix} \tag{2}$$

在映射时可以通过移位操作就可以实现乘法,垂直方向滤波的数据流图如图 2 所示。

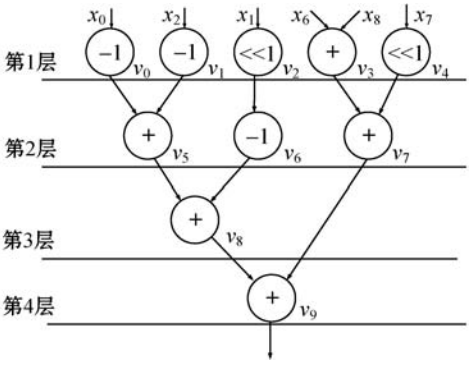


图 2　Sobel 算子垂直方向 DFG

1.2　DC 模式选择算法 DFG

帧内预测中 DC 模式(Direct Current)复杂度较高,需要对边界的参考像素平均值进行预测,其主要计算为对预测值 dcValue 的计算,dcValue 值的计算会直接影响到 DC 模式选择的执行速度,其主要操作为其中循环累加的运算,累加的次数与预测块大小相关。

$$d_{\text{cValue}} = \left(\sum_{x=1}^{N} R_{(x,0)} + \sum_{y=1}^{N} R_{(0,y)} + N\right) >> (\log_2(N) + 1) \tag{3}$$

式中:R 表示对应位置的像素点值。

本文以 $N=8$ 作为 dcValue 计算进行 DC 模式选择,通过分析将其中累加计算操作展开,得到 DC 模式选择的数据流图,如图 3 所示。

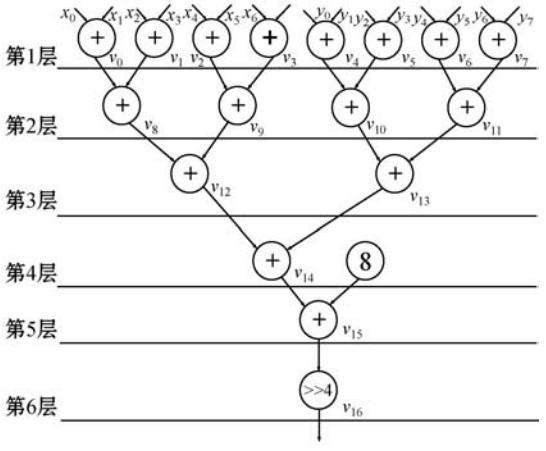


图 3 DC 模式选择 DFG

1.3 SAD 算法 DFG

HEVC 视频编解码标准中绝对差值和(Sum of Absolute Differences,SAD)值用来表示在预定搜索区域内当前帧与参考帧的绝对差值和,如式(4)所示。

$$S_{AD_{(i,j)}} = \sum_{i=1}^{m}\sum_{j=1}^{n} |f_k(m,n) - f_{k-1}(m+i,n+j)| \quad (4)$$

式中:$f_k(m,n)$ 为当前帧中亮度块的像素值,$f_{k-1}(m+i,n+j)$ 为参考帧中亮度块的像素值。在进行 SAD 值比较过程中,可以将 64×64 的编码单元拆分为 256 个 4×4 预测块,4×4 预测块之间可以相互组合独立计算不同编码单元的值,本文以最小划分块 4×4 为例实现 SAD 算法,将循环计算操作展开,得到 SAD 数据流图,如图 4 所示。

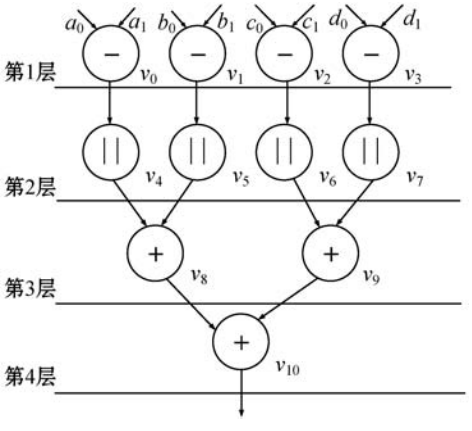


图 4 SAD 算法 DFG

2 DFGSP 的帧内预测算法 DFG 划分

本文采用 DFGSP 划分方法,基于算法的计算 DFG(其中每个节点表示为一个操作符),分别对当前待处理的各节点沿数据流向依次进行深度搜索。具体实现为:先从待处理的数据流图节点的就绪队列中取出首先要处理的节点,在给定的约束条件下,逐个划入同时满足下述两个条件的节点。

(1) 待处理节点的前驱节点已经处理完输入数据并至该节点输入端口。

(2) 新节点加入后,当前划分块的输出边数不在原来的基础上增加。

若任一条件不符合时,则将该节点跳过,并将之前跳过流程后的后驱节点划为就绪节点,以贪婪搜索的思路继续搜索未处理的就绪节点,直至没有待处理节点加入,对数据流图的搜索结束。

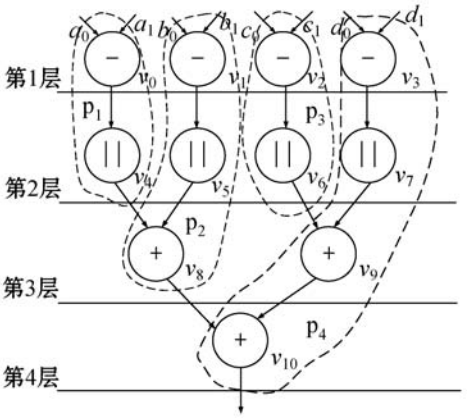
本文以 SAD 算法 DFG 划分为例详细描述划分过程。图 4 中 a_0、b_0、c_0、d_0 表示原始像素直接从数据输入存储(Data Input Memory,DIM)中读取,a_1、b_1、c_1、d_1 表示参考像素直接从数据输出存储(Data output memory,DOM)中读取。将所有待处理的节点分别划分为 v_0-v_{10},初始待处理节点为第一层中的 v_0、v_1、v_2、v_3 节点,具体划分步骤如下:

(1) 第一层的 v_0 节点按深度优先流向第二层的 v_4 节点。数据流到第三层的 v_8 节点时,需等待第二层 v_5 的计算结果。

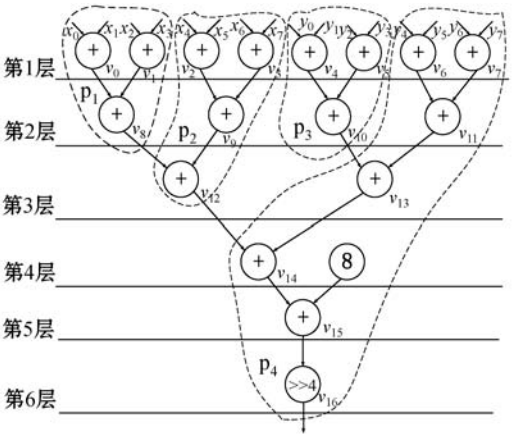
(2) v_8 节点加入后的块间边数小于当前块的边数,满足 DFGSP 算法条件,故将 v_8 节点加入 $v_1\rightarrow v_5$ 流程中。因此 $v_0\rightarrow v_4$ 为任务 p_1,$v_1\rightarrow v_5\rightarrow v_8$ 为任务 p_2。

(3) 第一层的 v_2 节点按深度优先流向第二层的 v_6 节点。数据流到第三层的 v_9 节点时,需等待第二层 v_7 的计算结果。

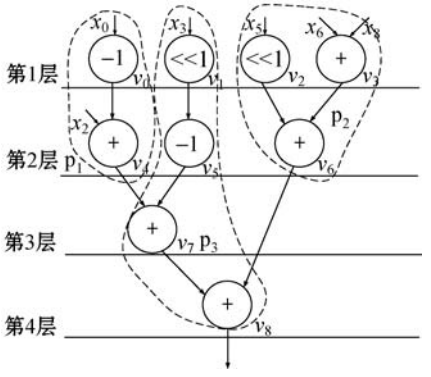
(4) v_9 节点加入后的块间边数小于当前块的边数,满足 DFGSP 算法条件,故将 v_9 节点加入 $v_3\rightarrow v_7$ 流程中,数据流到第四层的 v_{10} 节点时,v_8 节点已计算完毕。因此 $v_2\rightarrow v_6$ 为任务 p_3,$v_3\rightarrow v_7\rightarrow v_9\rightarrow v_{10}$ 为任务 p_4。可将 SAD 数据流图划分为如图 5(a)所示。



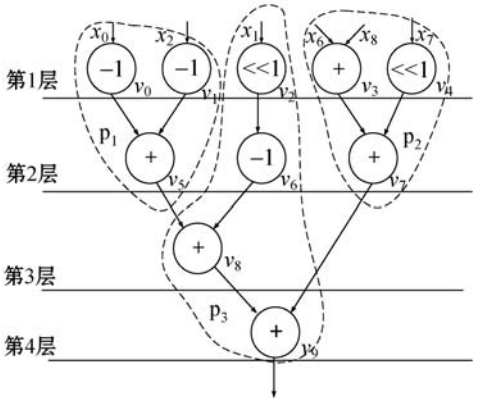
(a) SAD 算法 DFG 划分



(b) DC 模式选择 DFG 划分



(c) Sobel 算子水平方向 DFG 划分



(d) Sobel 算子垂直方向 DFG 划分

图 5　帧内预测算法 DFG 划分

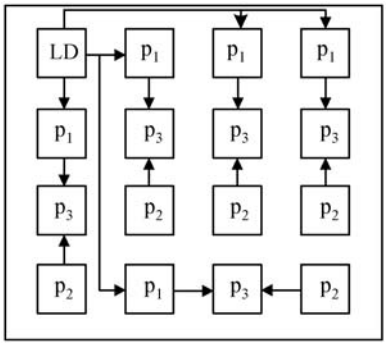
同理可得,DC 模式选择算法的数据流图划分为如图 5(b)所示,Sobel 水平方向滤波划分为如图 5(c)所示,垂直方向滤波的数据流图划分为如图 5(d)所示。

3　基于可重构阵列处理器的 HEVC 算法映射

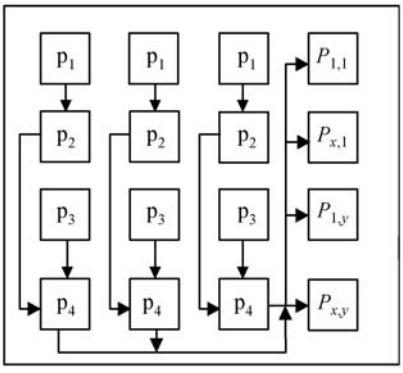
本文基于项目组自主研发的可重构视频阵列处理器(DPR-CODEC)[6]对算法进行仿真验证。前期研究的帧内预测算法[7]将帧内预测算法中不同预测模式分别在不同的处理元上并行实现,编码块之间采用流水线的方式在可重构视频阵列处理器上进行映射。

3.1　HEVC 算法并行化设计

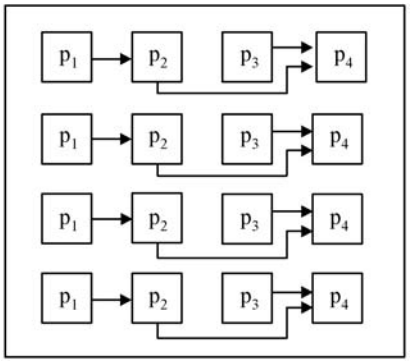
本文首先将已经划分后的 Sobel 算子数据流图映射在 DPR-CODEC 上,如图 6(a)所示,其中:任务 p_1 在 PE01、PE02、PE03 和 PE10 上分别实现加法和“-1、+”操作;任务 p_2 在 PE21、PE22、PE23 和 PE30 上分别实现加法和“<<、+”操作;任务 p_3 在 PE20、PE11、PE12 和 PE13 上分别实现加法和“<<、+、-1”操作;$P_{1,1}$、$P_{x,1}$、$P_{1,y}$、$P_{x,y}$表示经过计算后的对应位置的像素点值。



(a) Sobel 算法



(b) DC 模式选择算法



(c) SAD 算法

图 6　帧内预测算法映射图

对不同预测模式进行计算,其中 DC 模式实现如图 6(b)所示,任务 p_1 从 DIM 中读取原始像素值,在 PE00、PE10、PE20 上执行“+”操作,在 PE30 上执行“+”操作和“>>”操作。通过任务 p_1、p_2、p_3、p_4 计算预测块的 dcValue,通过 PE30、PE31、PE32 将 dcValue 的值计算完毕后,通过 PE 间的传输机制将数值传送

至 PE03、PE13、PE23 中分别计算对应的左上角、第一行像素、第一列像素的预测值。图 7 所示为 DC 模式选择 DFG 时域流水线,通过数据流水线方式并行执行完所有的像素点,完成 DC 模式选择算法。SAD 算法实现如图 6(c)所示。任务 p_1 在 PE00、PE10、PE20、PE20 上分别实现加法和“-、||”操作;任务 p_2 在 PE01、PE11、PE21、PE21 上分别实现加法和“-、||、+”操作;任务 p_3 和 p_4 与上述类似。

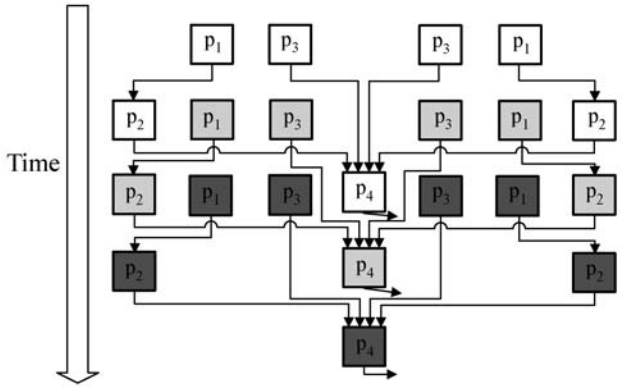


图 7 DC 模式选择 DFG 时域流水线

3.2 帧内预测算法重构实现

HEVC 帧内预测算法中,编码块可被划分为从 8×8 至 64×64 大小,对图像中的 64×64 编码块的像素点进行梯度计算,以流水的方式并行执行完所有的像素值,算出水平方向的滤波系数 G_x 和垂直方向的滤波系数 G_y。通过对当前预测单元内所有像素点的梯度值累加判断图像复杂度,如果大于阈值,说明图像复杂度较高,应该采用更小的块进行处理,通过这种算法得到最终图像的划分方式。经过大量的测试[8],得到不同预测单元的经验阈值,最终采取的经验阈值如表 1 所示。通过 HM 平台对不同测试序列的划分结果进行比较所得,8×8 大小的编码块被选择的平均概率为 66%,16×16 大小的编码块被选择的平均概率为 25%,因此本文以 8×8 和 16×16 的重构为例对不同编码块大小的算法进行切换实现。

表 1 不同预测单元经验阈值

预测单元大小	经验阈值
64×64	13 000
32×32	5 000
16×16	4 000
8×8	3 000

本文基于上下文切换的规模重构,将指令转化为可重构阵列处理器可识别的二进制代码存储在指令存储器,通过检测数据存储中的标志位信息,对不同的处理单元下发重构配置信息。以 Sobel 算子的经验阈值为编码块划分的判断依据,通过分层编程网络(Hierarchical Programming Network,HPN)向 PE00 中的第 500 号地址下发 Sobel 算子计算完毕后判断编码块大小的状态,并将该值指向对应代码的标志位,标志位 1 为执行 8×8 编码块的帧内预测算法,标志位 2 为执行 16×16 编码块的帧内预测算法。当执行 call 指令时,根据指令中的立即数地址,从存储块中读取出任务的配置信息,同时修改程序计数器,将配置信息的首地址写入其中。PE 根据配置信息,执行相应操作,实现 8×8 与 16×16 的块切换。

4 实验结果分析

4.1 测试方法及结果

在 DPR-CODEC 平台上进行基于数据流图的帧内预测算法的实现。首先将一帧原始像素和补充像素值存入 DPR-CODEC 的 DIM 中,并将帧内预测算法的指令初始化到指令存储器,用 Modelsim 仿真软件分别对文献[7]中项目组前期研究的帧内预测算法和基于数据流图的帧内预测算法进行仿真验证。实验选取了文献[7]中(Class A、B、C、D、E)不同分辨率下的一种序列来验证不同帧内预测并行算法性能,如表 2 所示。

表 2 不同测试序列算法执行时间对比

类型	测试序列	本文/ms	文献[7]/ms	加速比 S_n
Class A	Traffic. yuv (2 560×1 600)	483.30	7 237.06	14.97
Class B	BasketballDrive. yuv (1 920×1 080)	292.40	3 980.05	13.61
Class C	BasketballDrill. yuv (832×480)	101.97	747.25	7.33
Class D	RaceHorses. yuv (416×240)	29.39	171.44	5.83
Class E	FourPeople. yuv (1 280×720)	160.72	1 559.00	9.70

以加速比衡量两种方案的程序性能,加速比定义如下:

$$S_n = \frac{T_1}{T_n} \tag{5}$$

式中:T_1 是文献[7]算法串行执行计算时间;T_n 是本文提出的算法流水计算时间;S_n 是加速比。相比文献[7],针对不同测试序列,Class A 类测试序列 Traffic. yuv (2 560×1 600)加速比可达 14.97,Class D 类测试序列 RaceHorses. yuv(416×240)加速比为 5.83,需要处理的图像越大,可取得的加速比越高,算法执行时间平均减少了 92.2%。

资源利用率 U 的定义如式(6)所示。

$$U = \frac{r}{R} \tag{6}$$

式中:r 表示已经使用的 PE 资源数量,R 表示总 PE 数量。

文献[7]进行实现时,资源利用率为 25%,而采用时空流水线进行实现时,资源利用率为 83.3%,资源利用率提高了 69.9%。因此对帧内预测中循环计算多的 Soble 算子、SAD 算法和复杂度较高的预测模式采用上述方法进行实现,资源利用率提高如表 3 所示。各个算法的资源利用率均有提升。

表 3　资源利用率分析(%)

算法	串行实现 U	数据流图实现 U	ΔT
Sobel 算子	62.50	93.75	33.33
SAD 算法	68.75	83.30	17.47
Planar 模式	56.25	87.50	35.71
角度模式 2	68.75	87.50	17.47
角度模式 22	50.00	66.70	25.04
角度模式 26	31.25	75.00	58.33

4.2　硬件性能分析

本文统计了高清测试序列 highway_qcif.yuv (1 920×1 080)帧内预测算法完成一帧所用的时钟周期数为 283 500,表 4 为 DC 模式选择算法性能分析对比。文献[9]提出了两种硬件架构,并行流水线结构和并行数据路径结构,用于 DC 和 Planar 模式选择,其中 DC 模式使用并行流水线结构每个时钟计算 8×8 编码块所需时钟周期为 8,与之相比本文方案 1 个时钟周期可多处理 4 个像素。文献[10]设计了一个可以用于帧内所有预测模式的硬件结构,其中 DC 模式选择在 14 个时钟周期可处理 64 个像素值,平均每个时钟周期可处理 4.57 个像素,本方案在 1 个时钟周期内可多处理 7.43 个像素值。表 5 为 Sobel 算子、帧内预测算法性能分析。文献[11]提出了一个用于模糊逻辑边缘检测系统,使用 Xilinx Artix7 和 Kintex 设备进行实现和测试,系统时钟大约 100 MHz,与其他不同的实现平台相比,本文方法每个时钟周期均可多处理 1 个像素,工作频率也远高于文献[11]。文献[12]在编码单元简化了最有可能模式的建立规模,但使用原始像素替换其参考像素,对图像的预测并不准确,同时导致硬件的处理周期过长,本文虽然最大工作频率低于文献[12],但是采用了 Sobel 算子实现对图像的划分,不需要遍历所有预测单元,极大程度地减少了计算量,因此在保证编码质量的同时,本文每个时钟周期可比文献[12]多处理 7.1 个像素。

表 4　DC 模式选择性能分析

方法	文献[9]	文献[10]	本文
实现平台	Artix-7	Virtex-7	Virtex-6
最大工作频率/MHz	150.0	119.0	156.6
像素个数	64	64	96
计算时间/cycles	8	14	8
吞吐量/(pixel·cycle^{-1})	8.00	4.57	12.00

表 5　算法性能分析

方法	文献[11]		本文	文献[12]	本文
算法	Sobel			帧内预测	
实现平台	Artix-7	Kintex7	Virtex-6	Virtex-7	Virtex-6
最大工作频率/MHz	150.0	125.0	156.6	196.0	156.6
像素个数	64	20 736	64	1 024	256
计算时间/cycles	8	22 810	6	5 787	35
吞吐量/(pixel·cycle^{-1})	8	9.09	10.60	0.18	7.30
是否重构	否	否	是	否	是

5　结　语

本文提出一种基于数据流图的 HEVC 算法任务划分重构实现方法,结合 HEVC 算法高复杂度、计算量大、编码块之间切换灵活的特点,对 HEVC 中典型编解码算法进行了实现,并通过 Sobel 算子的梯度值划分图像的不同,实现了帧内预测算法的动态重构切换,并对其中复杂度高的 Sobel 算子、DC 模式选择算法和 SAD 算法通过数据流图划分的方式进行实现,采用数据流图子任务并行的方法,将算法在可重构视频阵列处理器加速实现。实验表明,Sobel 算子与文献[11]相比,在不同平台下处理速度均高于文献[11];与项目组前期实现的帧内预测算法相比,加速比可达 14.97,各个算法的资源利用率均有提升。DC 模式选择算法每个时钟周期内可比文献[10]多处理 7.43 个像素值,与文献[12]实现的帧内预测算法相比较,每个时钟周期可多处理 7.1 个像素,有效提高了计算速度。因此,将 HEVC 算法使用数据流图划分的方式在可重构视频阵列处理器上实现可以有效降低计算复杂度和减少执行时间。

参　考　文　献

[1] Dehon A. Fundamental underpinnings of reconfigurable computing architectures[J]. Proceedings of the IEEE,2015,103(3):355-378.

(下转第 257 页)

4 结　语

针对目前高性能计算任务调度中资源利用不足以及负载不均衡问题,提出一种遗传-蚁群算法的高性能计算任务调度算法,利用遗传算法的搜索空间解能力强,获得更好的解决方案,然后将它转换成蚁群初始信息素,最终得到更好的全局搜索性和反馈性。实验结果表明,GA-ACO 拥有更快的收敛速度,任务的执行时间更短,负载均衡性更好,从而说明 GA-ACO 有利于在高性能计算中解决任务调度问题。

参 考 文 献

[1] Borghesi A, Bartolini A, Lombardi M, et al. Scheduling-based power capping in high performance computing systems [J]. Sustainable Computing: Informatics and Systems, 2018,19:1 – 13.

[2] Sun H Y, Stolf P, Pierson J M. Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters[J]. Future Generation Computer Systems, 2017,71:157 – 170.

[3] Thoman P, Dichev K, Heller T, et al. A taxonomy of task-based parallel programming technologies for high-performance computing[J]. The Journal of Supercomputing, 2018, 74(4):1422 – 1434.

[4] Zhou A F. Genetic ant colony algorithm improves resource scheduling in cloud computing[C]//3rd International Conference on Information Science and System, 2020:106 – 109.

[5] Kumar A M, Venkatesan M. Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment[J]. Wireless Personal Communications, 2019,107(4):1835 – 1848.

[6] Jiang J, Zhang J, Zhang L J, et al. Passive location resource scheduling based on an improved genetic algorithm [J]. Sensors, 2018,18(7):2093.

[7] Wang D Y, An X S, Zhou X W, et al. Data cache optimization model based on cyclic genetic ant colony algorithm in edge computing environment [J]. International Journal of Distributed Sensor Networks, 2019,15(8):155 – 164.

[8] Xue H, Kim K T, Youn H Y. Dynamic load balancing of software-defined networking based on genetic-ant colony optimization[J]. Sensors, 2019,19(2):311.

[9] Rousset A, Herrmann B, Lang C, et al. A survey on parallel and distributed multi-agent systems for high performance computing simulations[J]. Computer Science Review, 2016, 22:27 – 46.

[10] Xu Y M, Li K L, Hu J T, et al. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues[J]. Information Sciences, 2014,270:255 – 287

[11] 张爱科,谢翠兰. 基于公平性和负载均衡的云计算任务调度算法[J]. 计算机应用与软件,2015,32(2):268 – 271.

[12] Xu M H, Zi Y. ACO-based network selection algorithm[J]. Computer Engineering and Applications, 2012,48(5):84 – 88.

[13] Keerthika P, Suresh P. A multi-constrained grid scheduling algorithm with load balancing and fault tolerance [J]. The Scientific World Journal, 2015,2015:349576.

[14] 孟令玺,孟令威. 云计算下的资源负载均衡性调度仿真[J]. 计算机仿真,2018,35(4):386 – 389.

[15] 饶健. 云计算平台网络应急信息优先调度仿真研究[J]. 计算机仿真,2018,35(6):353 – 356.

(上接第 245 页)

[2] Lu Y, Liu L, Zhu J, et al. Architecture, challenges and applications of dynamic reconfigurable computing[J]. Journal of Semiconductors, 2020,41(2):021401.

[3] Bo Z. A level sensitive cluster based partitioning algorithms for reconfigurable systems[J]. Journal of Computer Aided Design & Computer Graphics, 2006,18(5):667 – 673.

[4] 陈乃金. 基于深度优先贪婪搜索的可重构硬件任务划分算法[J]. 计算机应用,2012,32(1):158 – 162.

[5] Miyazawa K, Sakate H, Sekiguchi S I, et al. Real-time hardware implementation of HEVC video encoder for 1080p HD video[C]//Picture Coding Symposium, 2013.

[6] Zhu Y, Jiang L, Shi P, et al. Parallelization of intra prediction algorithm based on array processor[J]. High Technology Letters, 2019,25(1):74 – 80.

[7] 王飞龙,刘新闯,刘鹏,等. HEVC 帧内预测算法加速设计与实现[J]. 计算机应用与软件,2020,37(1):151 – 156.

[8] Bross B, Han W J, Sullivan G J, et al. High efficiency video coding (HEVC) text specification draft 10 (JCTVC-L1003)[C]//JCT-VC Meeting (Joint Collaborative Team of ISO/IEC MPEG & ITU-T VCEG), 2013.

[9] Lakshmi, Aparna P. Efficient architectures for planar and DC modes of intra prediction in HEVC[C]//2020 7th International Conference on Signal Processing and Integrated Networks (SPIN), 2020.

[10] Choudhury R, Rangababu P. Design and implementation of mixed parallel and dataflow architecture for intra-prediction hardware in HEVC decoder[C]//International Symposium on VLSI Design and Test, 2017.

[11] Kurdi A H, Grantner J L, Abdel-Qader I. Hardware accelerator for edge detection[C]//2020 IEEE 24th International Conference on Intelligent Engineering Systems, 2020.

[12] Xiuzhi Y, Min Z, Longzhao S, et al. Research and implementation of fast algorithm for intra prediction mode selection oriented to hardware[J]. Journal of Computer-Aided Design & Computer Graphics, 2019,31(1):158 – 164.