计算机科学与探索
Journal of Frontiers of Computer Science and Technology
1673-9418/2022/16(07)-1570-13
doi: 10.3778/j.issn.1673-9418.2012085

面向OpenVX核心图像处理函数的并行架构设计

潘风蕊[1+],李　涛[1,2],邢立冬[1],张好聪[1],吴冠中[1]
1. 西安邮电大学 电子工程学院,西安 710121
2. 西安邮电大学 计算机学院,西安 710121
+ 通信作者 E-mail: pfr_bang@163.com

摘　要:传统的可编程处理器虽然高度灵活,但其处理速度及性能不及专用集成电路(ASIC),而图像处理往往是多样、密集且重复的操作,因此处理器要兼顾速度、性能及灵活性。OpenVX是图像图形处理、图计算和深度学习等应用的预处理或者辅助处理开源标准,基于最新的OpenVX 1.3标准中的核心图像处理函数库,设计并实现了一种可编程、可扩展的专用指令集处理器(ASIP)——OpenVX并行处理器。首先分析对比了各种互联网络的拓扑特性,选择了性能比较突出的层次交叉互联网络(HCCM+)作为系统主干,在网络节点处设置处理单元(PE)构成支持动态配置的4×4 PE阵列,结合高效的路由通信方式设计了并行处理器,实现可编程的图像处理。其次所提出的架构适合数据并行计算和新兴的图计算,两种计算模式可单独或混合配置使用,分别将核心视觉函数及图计算模型映射到并行处理器上对两种模式进行验证,对比PE数目不同的情况下图像处理的速度。实验结果表明,并行处理器能够完成对基本核心函数和高复杂度的图计算模型的映射,在数据并行计算和流水线处理两种模式下,可以对图像处理线性加速,调用16个PE对各类函数的平均加速比可达15.037 5。验证环境采用20 nm XCVU440平台芯片,综合实现后频率为125 MHz。

关键词: OpenVX核心图像处理函数;专用指令集处理器(ASIP);并行处理器;层次交叉互联网络(HCCM+);图计算模型
文献标志码:A　**中图分类号:**TP302

Parallel Architecture Design for OpenVX Kernel Image Processing Functions

PAN Fengrui[1+], LI Tao[1,2], XING Lidong[1], ZHANG Haocong[1], WU Guanzhong[1]
1. School of Electronic Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, China
2. School of Computer Science & Technology, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

Abstract: Although the traditional programmable processors are highly flexible, their processing speed and performance are inferior to the application specific integrated circuit (ASIC). Image processing is often a diverse, intensive and repetitive operation, so the processor must balance speed, performance and flexibility. OpenVX is an open source standard for preprocessing or auxiliary processing of image processing, graph computing and deep learning applications. Aiming at the kernel visual function library of OpenVX 1.3 standard, this paper designs and implements a programmable and extensible OpenVX parallel processor. The architecture adopts an application specific instruction processor (ASIP). After analyzing and comparing the topological characteristics of various interconnection networks,

基金项目:陕西省科技统筹项目(2015KTCQ013);陕西省教育厅协同创新中心项目(17JF032);陕西省教育厅科研计划项目(20JY058)。
This work was supported by the Science and Technology Overall Planning Project of Shaanxi Province (2015KTCQ013), the Project of Collaborative Innovation Center of Shaanxi Provincial Department of Education (17JF032) and the Scientific Research Project of Shaanxi Provincial Department of Education (20JY058).
收稿日期:2020-12-22　**修回日期:**2021-02-25

the backbone of the ASIP chooses the hierarchically cross-connected Mesh+ (HCCM+) with outstanding performance, and processing element (PE) is set at network nodes. PE array is constructed to support dynamic configuration, and a parallel processor is designed to realize programmable image processing based on efficient routing and communication. The proposed architecture is suitable for data parallel computing and emerging graph computing. The two computing modes can be configured separately or mixed. The kernel visual function and graph computing model are mapped to the parallel processor respectively to verify the two modes and compare the image processing speed under different PE numbers. The results show that OpenVX parallel processor can complete the mapping and linear speedup of kernel functions and high complexity graph calculation model. The average speedup of scheduling 16 PEs to various functions is approximately 15.0375. When implemented on an FPGA board with a 20 nm XCVU440 device, the prototype can run at a frequency of 125 MHz.

Key words: OpenVX kernel image processing functions; application specific instruction processor (ASIP); parallel processor; hierarchically cross-connected mesh+ (HCCM+); graph calculation model

近年来，计算机视觉（computer vision，CV）在深度学习领域的应用迅速发展，图像处理作为CV中比较活跃的一个分支，广泛应用在医疗卫生、安检刑侦、图像检索与分析、增强现实等领域中[1]。而CV领域更新变化飞快，这就要求图像处理具有很高的灵活性、实时性和精确性。传统的硬件很难满足可编程性、高性能和低功耗的要求。可编程技术的出现使得硬件变得可“编译”，能在嵌入式系统上完成多种多样的新任务。并行计算方式的出现使得硬件介质可以提供更加强大的计算能力和密度，大幅提高了芯片系统的总体性能，实现片上超级计算[2]。业界典型的两种并行可编程模型分布式共享内存MPI（multi-point interface）和集中式共享内存OpenMP，与当前GPU多核、众核架构相比[3]，形式过于单一。可编程的专用指令集处理器可以兼顾功耗、性能和灵活性[4]，一种专为图形图像处理而设计的新型多态阵列处理器[5]应运而生，其不但处理性能在一定程度上接近于ASIC（application specific integrated circuit），而且具有灵活的可编程性，它能够将线程并行、数据并行、指令并行和操作并行融合到一个单一的阵列结构中。对于图像计算，ASIP（application specific instruction processor）是一种可行的硬件设计方法，基于ASIP体系结构，本文提出了一种面向计算机视觉底层任务加速的可编程并行处理器。

首先，本文研究了各种拓扑特性对互联网络传输性能的影响，分析了一类基础网络的拓扑特性，选择了一种更加灵活的新型网络结构——层次交叉互联网络（hierarchically cross-connected mesh+，HCCM+），可以根据不同应用的网络流量重新配置为Mesh、HCCM或HCCM−网络，降低整个系统的功耗。其次，以HCCM+的网络拓扑结构为基础，设计实现了一种可编程的OpenVX并行处理器，使用有限的硬件资源，以可编程的方式对OpenVX 1.3标准中核心函数进行映射，实现通用的图像处理。

1 OpenVX的介绍

2019年Khronos发布的OpenVX 1.3标准中，核心图像处理函数包括了基本像素点处理、全局处理、局部处理、特征提取四大类[6]。OpenVX标准是按照新兴的图计算方式指定的，其基本加速原理是根据需求有目的性地对图像矩阵进行一定的操作。如图1所示，OpenVX基本图像处理核函数可以看作整个处理流程中的一个节点（node）。对于图像处理流程往往是数据从源Node流向目标Node，Node与Node之间形成一定的有向无环图（graph）[7]。开发者可以根据需要将这些基本的Node连成Graph，完成对图像的操作。

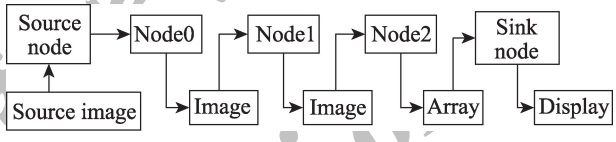


图1 OpenVX图计算模型

Fig.1 OpenVX graph calculation model

OpenVX作为计算机视觉一个标准化的功能框架，接口统一规范，具有良好的移植性，可以直接被应用程序使用；也可以作为高级视觉的加速层、框架、引擎或平台API[8]，被诸多芯片企业（如NVIDIA、AMD、Intel）采用，具有广泛的应用前景。但专门为OpenVX实现的硬件芯片十分匮乏。本文的工作属于比较早涉及此标准的硬件设计之一，所提出的架

构适合图计算和数据并行计算，实现可编程的加速处理，最大限度地提高硬件功能和性能的可移植性。

2 OpenVX并行处理器的整体架构

OpenVX并行处理器是利用有限的硬件资源实现OpenVX核心函数，图像处理往往是密集且重复的操作，因此除了高速计算之外，提高硬件的通用性，尽可能地使资源共享、降低功耗也成为并行处理器设计的目的。

2.1 互联网络复杂性分析

互连网络[9]是并行处理系统[10-11]的重要组成部分，对于数据信息传递的并行计算[12]，互连网络对系统的整体性能尤为重要，是本文所提出的OpenVX并行处理器的主干。由于简单的拓扑结构易于在超大规模集成电路(very large scale integration，VLSI)中实现和分析，互连网络宜采用简单的拓扑结构，拓扑特性包括[13]：

(1)边数(edge number)：网络链路数，影响互联网络的容量及灵活性。

(2)直径(diameter)：任意两个节点之间的最短路径中的路径长度。直径与通信时间成正比，直径越长，所需通信时间越长。

(3)对分宽度(bisection bandwidth)：网络被分成节点数相等的两部分，切口处最小边数为对分宽度，该参数主要反映了整个网络的最大流量。

互联网络的设计要兼顾上述3种拓扑特性，使得整体结构的性能在一定程度上有所提高。最基本的拓扑结构是Mesh型、XMesh型[14]，本文在基本结构的基础上，选择了HCCM结构，并对HCCM进行扩展得到HCCM−、HCCM+两种拓扑结构，如图2所示。

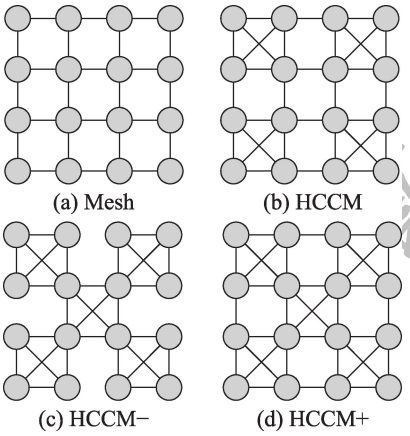


图2 互连网络拓扑结构

Fig.2 Interconnection network topology

根据递归特性、循环和边界条件，推导计算出4种结构的拓扑特性公式，如表1所示。

表1 互联网络拓扑性能比较

Table 1 Comparison of network topology performance

类型	Mesh	HCCM	HCCM−	HCCM+
级数	n	n	n	n
边长	2^n	2^n	2^n	2^n
节点数	2^{2n}	2^{2n}	2^{2n}	2^{2n}
边数	$2^{2n+1}-2^{n+1}$	$(5/2)4^n-2^{n+1}$	$2(4^n-1)$	$(8/3)4^n-2^{n+1}-(2/3)$
直径	$2^{n+1}-2$	$3\times2^{n-1}-2$	2^n-1	2^n-1
等分宽度	2^n	2^n	4	2^n+2

通过对比四种结构的拓扑特性，HCCM+的直径小于Mesh和HCCM，且比Mesh、HCCM和HCCM−具有更宽的对分宽度。在理论上它可以比其他结构承载更大的传输量，性能优于其他结构。因此本文采用HCCM+作为并行处理器的基础网络。

2.2 并行架构设计

OpenVX并行处理器采用了性能较为突出的HCCM+网络，包含了4×4个处理单元(processing element，PE)、路由(RU)及全局控制(global control)，整体结构如图3所示。在开始计算前，全局控制器接收微控制器(micro control unit，MCU)发送的微指令，根据指令中的PE标识号(ID)，将配置及数据信息经配置开关模块(cfg&switch)下发至相应的PE中。PE执行任务时以微指令信息为单位，可以循环执行

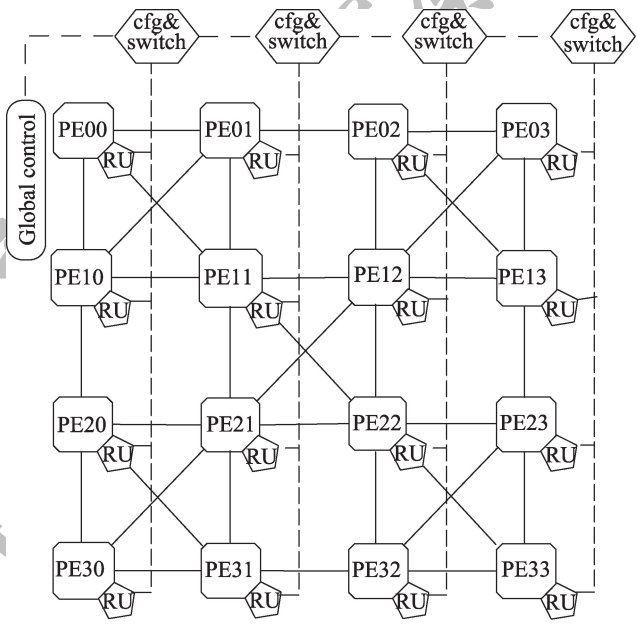


图3 OpenVX并行处理器整体架构

Fig.3 OpenVX parallel processor architecture

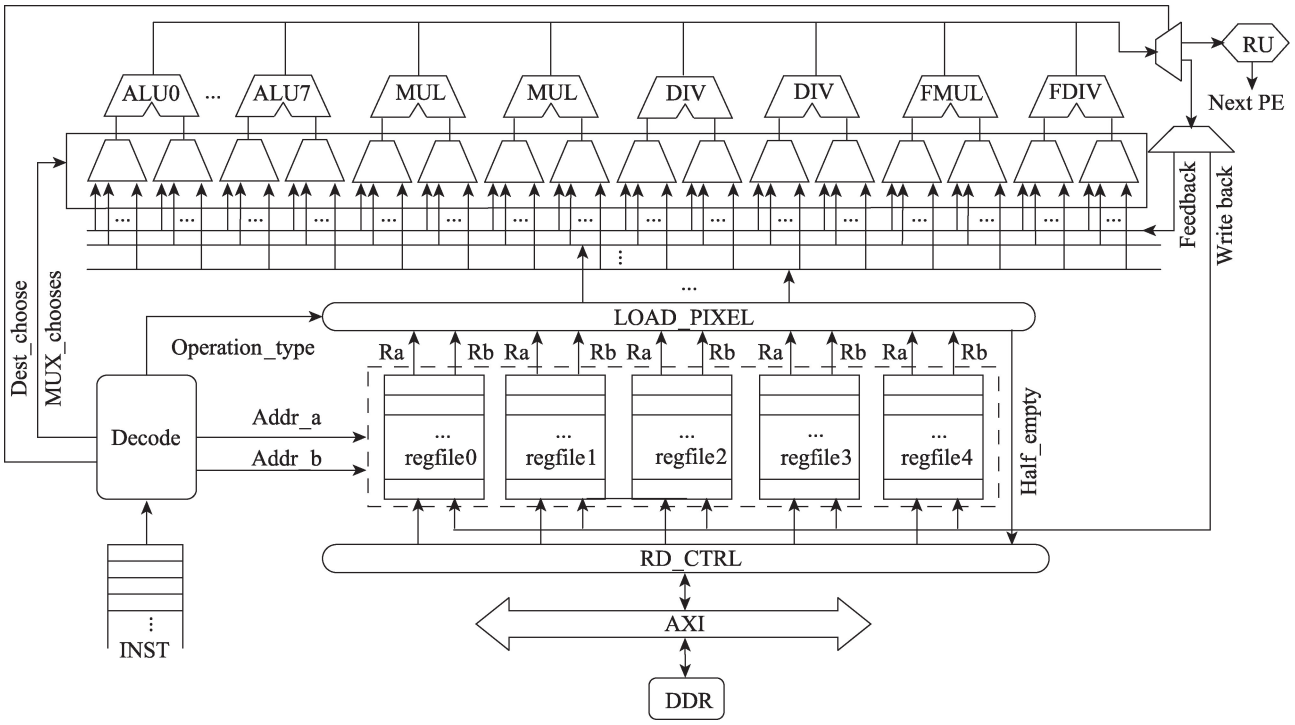


图4 PE整体结构

Fig.4 PE architecture

微指令携带的操作信息，中间结果可暂存到PE内部缓存中，由RU控制选通PE之间的数据链路，完成数据的传输。MCU下发的微指令可实现多个PE的任务切换，实现整体可配置。

2.2.1 运算单元结构设计

每个PE中包含了8个ALU、2个定点乘法器（MUL）、2个浮点乘法器（FMUL）、1个定点除法器（DIV）、1个浮点除法器（FDIV）、1个内部交叉开关（Crossbar）、5个32×16的寄存器堆（regfile）以及寄存器堆的访存模块（RD_CTRL，LOAD_PIXEL），整体结构如图4所示。

（1）ALU负责执行add、sub、and、or、xor、sll、srl等指令。当ALU执行add指令时，a输入端和b输入端有三个数据源：①指定寄存器堆内的数据；②译码单元的立即数；③计算单元的输出数据。Crossbar从三个数据源中选择一个送至ALU的输入端，输出结果由分配器派遣至下一个指定的PE，进行新的处理。其中第三个数据源的选择，是为了处理比较复杂的函数如Canny边缘检测、Harris角点检测等，中间结果写回，进行新的配置，继续计算至最终结果的输出。

（2）PE内部的Crossbar设计为60×32的开关矩阵（switch matrix）、60个输入、32个输出。通过配置译码模块（configer decode）将选通信号发送给分配器和选择器，ALU输入操作数可以有选择性地连接寄存器堆或ALU的计算结果。如图5所示，以ALU0为例，ALU0的两个操作数是多路选择器的输出，ALU0计算结果（Mid_out）通过分配器输出。输入、输出数据流向选择取决于译码模块的配置指令。将多个计算

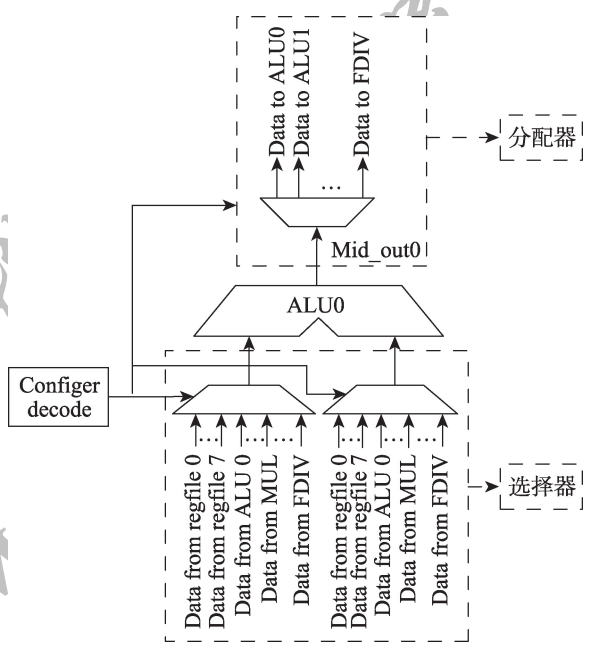


图5 ALU0 I/O数据通路

Fig.5 ALU0 I/O data path

单元的输入、输出数据通路进行互联，组成开关矩阵。

(3)寄存器堆访存模块包括了RD_CTRL和LOAD_PIXEL。RD_CTRL通过AXI总线访问DDR，采用突发读写的模式，将像素数据加载至寄存器堆中，减少了对DDR访问次数。LOAD_PIXEL访问寄存器堆，将像素数据加载至Crossbar的数据端口，对于像素点操作和模板类操作的函数，由译码模块发送的操作类型信号(operation_type)配置不同的加载方式。当LOAD_PIXEL模块访问至寄存器堆的中间或最大地址时，会向RD_CTRL发送Half_empty信号，RD_CTRL继续访问DDR，形成一个动态加载和动态取数的过程。

2.2.2 全局控制

全局控制(global control)负责控制各个模块之间以及模块内部的数据交互、数据流向及数据选择。图6是对一个PE进行控制的电路设计。

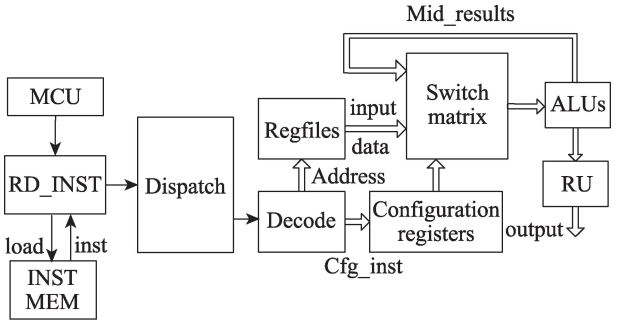


图6 全局控制电路

Fig.6 Global control circuit

MCU负责发送Node类型，开始进行图像处理。RD_INST向INST_MEM发送请求，读取微指令。派遣模块(dispatch)负责将微指令派遣至相应PE的译码单元(decode)进行解析。译码单元将地址信息(address)发送至寄存器堆(regfile)的地址端，将配置信息(cfg_inst)顺序写入到开关矩阵(switch matrix)的配置寄存器(Cfg_inst)中，进行数据通路的选择，像素数据流向对应的ALU进行计算。ALU的输出可作为中间结果(Mid_results)返回，也可经RU直接输出至下一个PE对应的缓存中。

2.2.3 PE外部路由的设计

PE之间数据路由受全局控制模块控制，本文路由以XY路由为基础[14]，并对其进行了一定的改进，增加了新的判断状态，在数据从当前节点流向目标节点的过程中增加一条对角边。路由结构如图7所示，上一级各个方向PE的输出Edout、Sdout、Wdout、Ndout、ESdout、WNdout输入至相应的缓存E_buf、S_buf、W_buf、N_buf、ES_buf、WN_buf。Arbiter模块采用先来先到的仲裁机制决定数据的传输顺序。RU模块根据规定的传输方向优先级决定数据流向，数据经全互联交叉开关输入至目的PE。

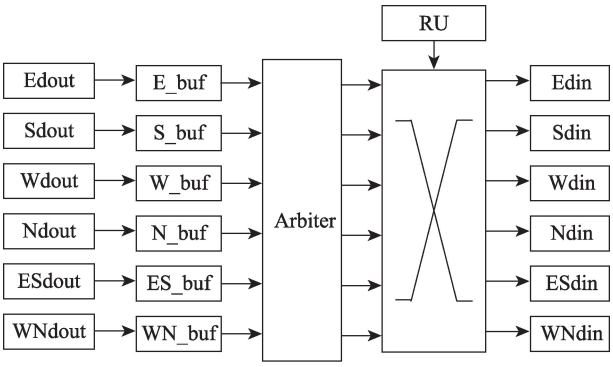


图7 路由结构

Fig.7 Route structure

RU模块支持一对一单目标传输，其中对角快速通道优先级最高，其次为水平方向的数据通路，最后为竖直方向；RU也支持一对多目标的数据扇出，此时目标与目标之间优先级相同，对于某一目标通道优先级与一对一单目标传输时优先级相同。译码模块将路由配置信息写入PE内部路由选择寄存器中，选通本地PE到目标PE的通信路径，将计算结果发送至目标PE，执行新任务。

PE之间主要数据通路如图8所示，假设当前节点为PE00，目标节点为PE32，则数据流向为：PE00→ES_dout→PE11→ES_dout→PE22→S_dout→PE32。

2.3 整体架构的优化

2.3.1 高性能并行指令集

高性能OpenVX并行处理器共有五种指令类型：定点运算、浮点运算、逻辑运算、加载/存储、特殊指令(控制、存取及停止)。以定点运算为例，常规的指令格式为：

opcode	Rd	Ra	Rb

本文指令格式为：

opcode1	Rd1	Ra1	Rb1	…	opcode4	Rd4	Ra4	Rb4

其中，Rd为目的寄存器，提供计算结果地址；Ra为源寄存器，提供源操作数A地址；Rb为源寄存器，提供源操作数B地址。opcode为操作类型，相比于传统的指令，本文每条指令中有4个opcode，一次可控制多个操作，使得更多运算单元能够同时工作，增大了操作的并行度，提高了像素的吞吐率。

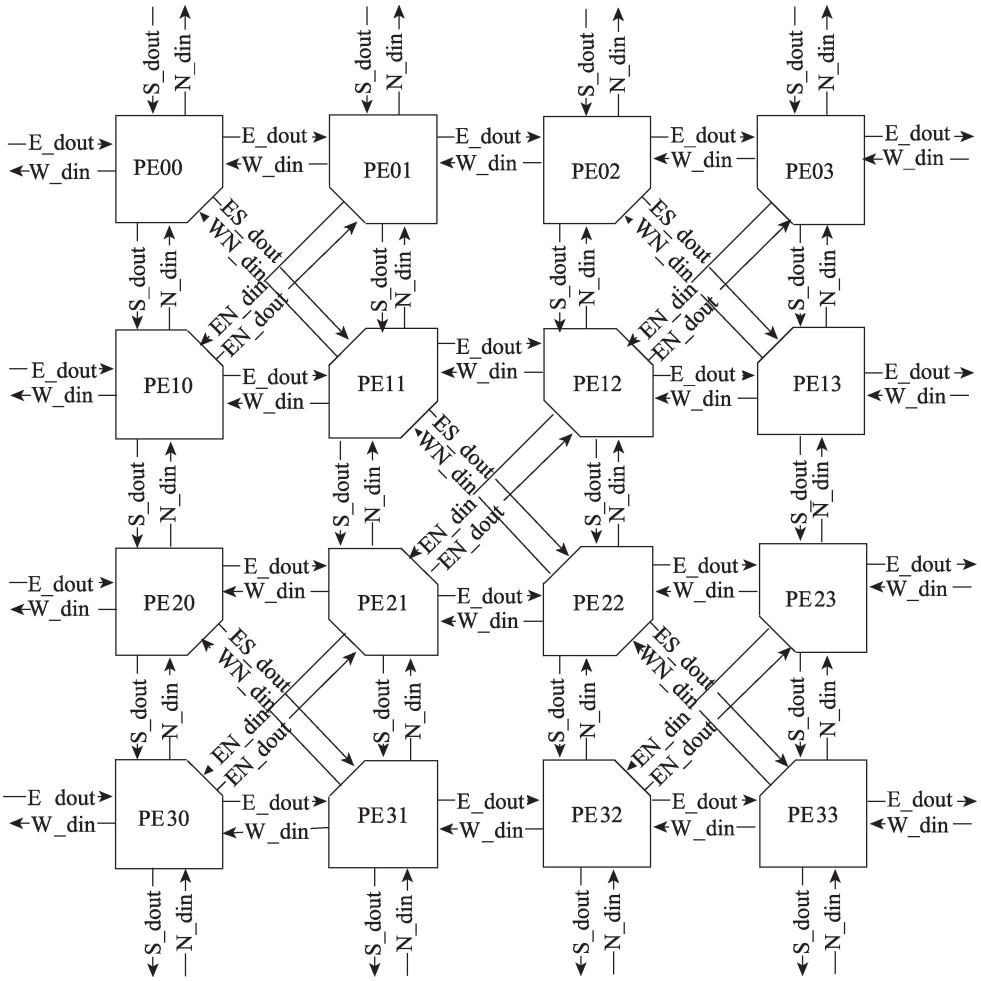


图8 PE之间数据路由

Fig.8 Data routing between PEs

为了减少数据通路，ALU中将加法和移位合并，同时包含了与、或、非以及异或等布尔运算。例如局部图像处理，模板总是固定的常系数，目标中心像素由常系数与窗口内的像素进行乘加得到。而OpenVX并行处理器中只需要配置ALU执行ADD操作即可，耗费一个时钟周期就可完成4组像素的乘加操作。

同时在用汇编指令实现核心函数时，可以合理地、自由地调整配置指令的顺序，尽量避免相邻指令间数据传递依赖性，减小整个程序的执行时间。

2.3.2 PE及访存电路的优化

传统运算单元中只有一个ALU，合理地增加PE中ALU数目，可以减少流水停滞时间，增大像素吞吐率，计算速度快。

为减少PE通过AXI总线访问DDR的次数，在PE内部设置了5个深度为16，宽度为32 bit寄存器堆，用于快速存取原像素数据及中间计算结果。PE之间传递数据直接通过访问邻接共享存储，实现数据的传递复用，只有最后一个PE的计算结果才写回DDR，用于最终的显示。

2.3.3 灵活的网络结构及高效的路由方式

本文所采用的HCCM+中，嵌入了Mesh、HCCM和HCCM−，可以根据不同应用的网络流量重新配置为Mesh型、HCCM型或HCCM−型结构。对于流量较大的应用，可以使用HCCM+所有边缘。对于中等级别的流量，可以关闭一些边缘以形成HCCM网络，这有助于节省功耗。对于大多数短距离轻型通信，可以关闭更多的边缘来形成HCCM−网络，进一步降低功耗且更加灵活。

为了适应系统网络结构，本文路由在XY路由算法的基础上，增加了对角传输路径。这种改进不仅为数据到达目标节点提供了最佳传输路径，而且增大了单节点的数据扇出，可以适应复杂度更高的Graph，使更多的PE可以交互，延长PE数据传输链路，提高PE单元的复用率。同时路由中采用结构简单的全互

联交叉开关,传输延迟低,速度快,可实现数据并行传输。

3 Graph映射

并行处理器的流处理类似于FPGA(field programmable gate array)或ASIC的流水线[15],每个运算单元处理的颗粒粗细程度由MCU进行配置调整,宏观上实现流水处理。

3.1 数据并行计算模式

数据并行计算模式是对图像分块处理,将图像分配至不同的处理单元,配置相同的指令,对不同的图像数据进行相同的操作,实现数据级并行操作。针对该模式,选取基本的核心函数在并行处理器上进行映射,多个处理单元同时执行同一种操作。如图9所示,将Sobel函数在并行处理器上以数据并行处理的模式进行映射。图10为数据并行处理模式下图像分块方式,PE调度的个数与图像分块的个数相关。在每个PE内部,针对核心函数的算法,对基本算数逻辑操作流水细分,按需分配,微观上实现细粒度并行运算。

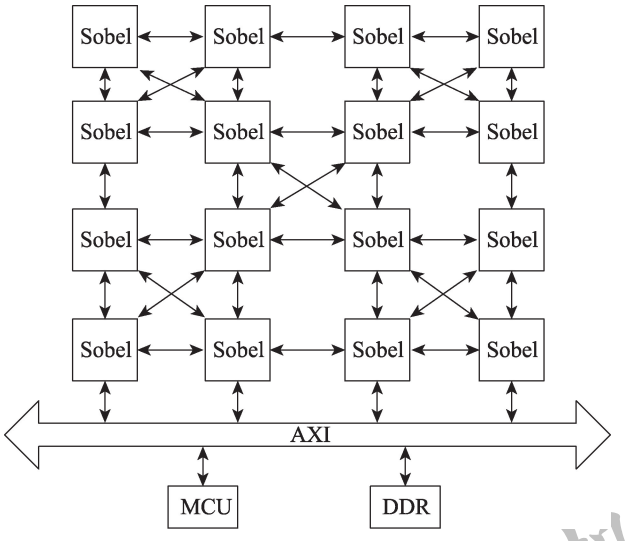


图9 数据并行计算模式映射

Fig.9 Data parallel computing pattern mapping

3.2 流水线处理

流水线处理是将复杂的程序分解为数个简单的操作,分配到不同的处理单元上,处理单元之间可通过相邻的数据通路进行数据传递,整体实现流水处理。相邻处理单元之间指令独立,使本结构非常适合流水线处理的运行方式。针对该模式,选取均值滤波(box filter)、通道合并(channel combine)、色系转换(color convert)、图像膨胀(image dilate)和图像腐蚀(image erode),构造形态学滤波的Graph计算模型,执行流程如图11所示。每个基本函数为一个Node,将该执行流程映射到并行处理器上,由于通道合并和通道提取比较简单,将其与均值滤波合并到一起,在结构映射中并未体现,映射结果如图12所示。

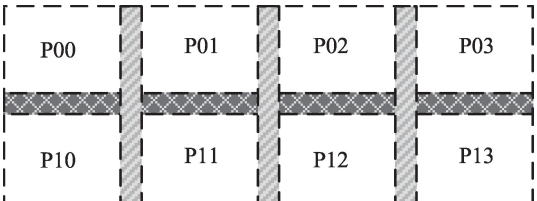


图10 数据并行计算模式下图像分块

Fig.10 Image segmentation in data parallel computing mode

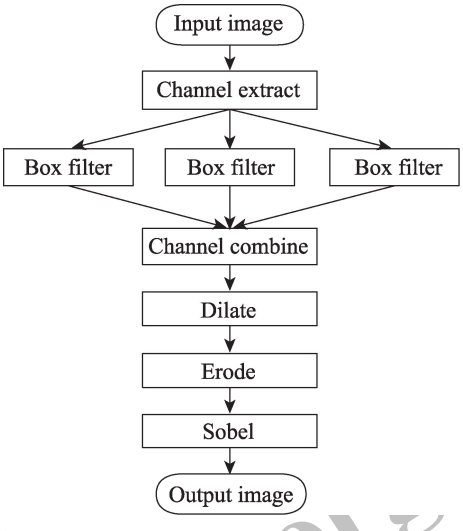


图11 形态学滤波执行流程

Fig.11 Morphological filtering execution flow

如图13所示是形态学滤波数据分块分配方式,将图像数据分成四块,其中阴影部分为图像分块后的边界。常见的边界处理方法是边界复制和边界填零[16]。MCU控制加载图像像素数据时,发送两次边缘像素的读取指令,实现对边界像素进行复制。第一块图像数据经过P00、P01、P02、P03所构成的运算单元链(流水线)分任务进行处理。实际上,形态学滤波是由多流水线并行处理实现的,执行不同任务

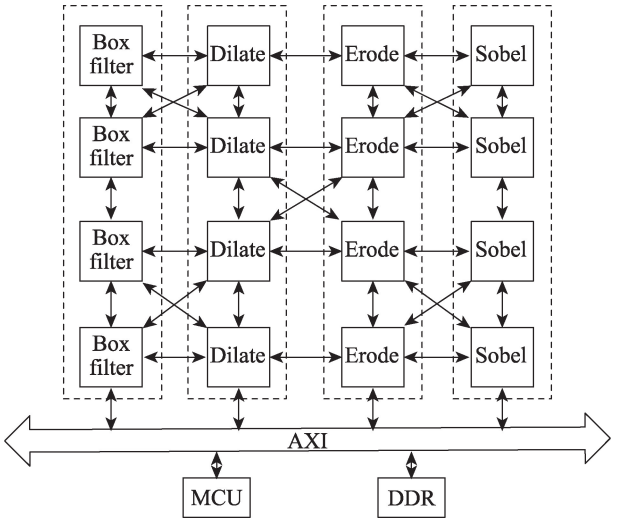


图12 流水线处理模式映射

Fig.12 Pipeline processing pattern mapping

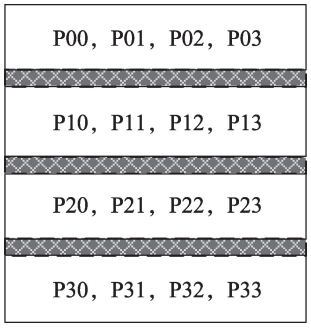


图13 形态学滤波数据分块

Fig.13 Morphological filtering data segmentation

的PE实现流水线处理，执行同一任务的PE实现数据并行处理，宏观上实现粗粒度并行运算。

4 测试验证及结果分析

4.1 验证平台

对图像处理硬件系统设计一个完整稳定的验证系统也是至关重要的，本设计基于VS2015的MFC仿真测试平台，如图14所示。VS平台和FPGA平台共享输入源图像数据，经过相同的算法，将最终图像处理的结果打印至文本中。可以直接打开文本查看结果，也可以对软件算法处理结果与FPGA的处理结果进行逐像素对比验证，并输出两者的比对结果。

4.2 实验结果分析

除了要关注图像处理的正确性外，还要考虑并行处理器的性能，性能可以通过阿姆达尔定律模型展开分析[17]，用加速比来衡量并行处理的效果，式(1)在高度理想情况下等号成立，因此加速比往往是小

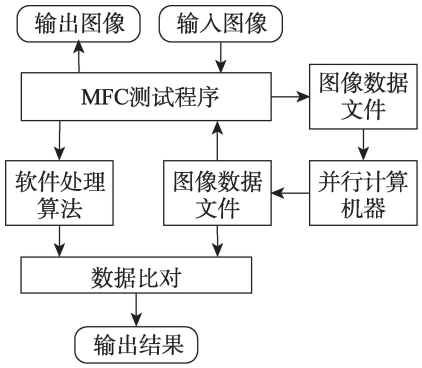


图14 仿真测试平台

Fig.14 Simulation test platform

于处理器的个数。

$$Speedup \leqslant \frac{s+p}{s+\frac{p}{n}} = \frac{T_{p+s}}{T_{np+s}} \tag{1}$$

式中，s 为串行处理部分，p 为可并行处理部分。在数据并行计算模式下，n 为正在处理的像素数据个数，即图像分块的个数。在流水线处理模式下，n 为执行不同指令的处理器个数，同时也是Graph计算模型流水线的阶段个数。

4.2.1 数据并行计算结果分析

在数据并行计算模式下，从OpenVX视觉函数库中四类函数(Ⅰ基本像素点处理函数、Ⅱ全局处理函数、Ⅲ局部处理函数、Ⅳ特征提取函数)中选取部分典型函数进行映射。用Modelsim仿真工具进行仿真，调用不同数目PE对分辨率为640×480像素的图像进行处理，对每个函数的处理时间进行了统计。如表2所示，对图像处理时间为 T，结合阿达姆定律计算了相应的加速比 S_{up}。经统计分析后，对同一函数，调用更多数目PE的情况下，图像处理时间骤减，所对应的加速比成线性增长。但对于不同类别的函数加速效果存在一定的差异，Ⅱ、Ⅳ类函数的加速比略低于Ⅰ、Ⅲ类函数。

为进一步分析每类函数加速比存在差异的影响因素，在每一类中选取一个具有代表性的函数，分别对处理过程串并比例进行统计。如图15所示，分别为通道提取、直方图、Sobel、高斯金字塔中串行并行比重统计，s 表示串行处理部分，p 表示并行处理部分。对于Ⅰ类函数，串行处理时间主要为预加载像素的时间；Ⅲ类函数相较于Ⅰ类，串行处理额外地增加了读取窗口内多个像素及多像素之间相关性操作时间。Ⅱ类函数串行处理时间主要来自遍历整幅图像、统计或累加预处理的时间，串行预处理完后才开

表2 基本核心函数加速比对比

Table 2 Comparison of speedups of basic kernel functions

函数类别		$T\&S_{up}$	1PE	2PE	4PE	8PE	16PE
Ⅰ	通道提取	T/clk	307 364	167 045	90 667	44 740	20 382
		S_{up}	1.00	1.81	3.45	6.94	15.29
	颜色转换	T/clk	307 375	169 820	89 094	44 290	20 103
		S_{up}	1.00	1.84	3.39	6.87	15.08
	位深转换	T/clk	307 372	171 716	90 403	44 871	20 301
		S_{up}	1.00	1.79	3.40	6.85	15.14
Ⅱ	直方图	T/clk	614 426	372 379	187 897	94 818	41 375
		S_{up}	1.00	1.65	3.27	6.48	14.85
	图像积分	T/clk	438 723	270 816	141 068	68 981	29 643
		S_{up}	1.00	1.62	3.11	6.36	14.80
Ⅲ	中值滤波	T/clk	307 283	167 914	85 594	44 858	20 162
		S_{up}	1.00	1.83	3.57	6.85	15.21
	图像膨胀	T/clk	307 282	167 001	85 832	44 990	20 162
		S_{up}	1.00	1.79	3.52	7.34	15.24
	图像腐蚀	T/clk	307 282	167 001	85 832	44 924	20 162
		S_{up}	1.00	1.79	3.52	7.34	15.24
	Sobel	T/clk	307 285	168 837	86 074	44 924	20 256
		S_{up}	1.00	1.82	3.59	7.12	15.17
Ⅳ	Canny	T/clk	307 314	187 386	91 735	46 775	20 750
		S_{up}	1.00	1.64	3.35	6.57	14.81
	高斯金字塔	T/clk	409 615	245 278	121 547	62 824	27 695
		S_{up}	1.00	1.67	3.37	6.52	14.79
	Harris Corner	T/clk	307 309	182 922	89 856	46 917	20 722
		S_{up}	1.00	1.68	3.42	6.55	14.83

始进行并行处理。Ⅳ类函数，复杂度明显高于其他三类，存在中间结果写回，导致流水细分程度略低于其他三类函数。

在调用不同数目PE下，四个代表函数处理时间对比如图16所示，由于每类函数的可并行处理部分所占的比重不同，处理时间存在较大的差异，可并行处理度越高的函数，实现所用时间越短。

4.2.2 流水线处理结果分析

在流水线处理模式下，将形态学滤波Graph执行模型映射到本结构上，启用多个流水线进行同一Graph的运算。本文将PE阵列分为4个流水线进行加速比的计算。如果需要增加线程数，则需要将Graph计算模型中的操作重新划分，增加每个PE的任务负载，在总任务不变的情况下，对空闲的PE重新分配任务。或是将本并行处理器作为基本簇，以此扩展，实现多簇并行执行任务，进一步提高整体的并行性。如图17所示是原图与本文硬件处理后结果图的对比，其中(a)是源图像，(b)是膨胀腐蚀后的开操作结果，(c)是Sobel边缘检测结果图，(d)是软硬件处理结果比对图，验证了硬件电路功能的正确性。

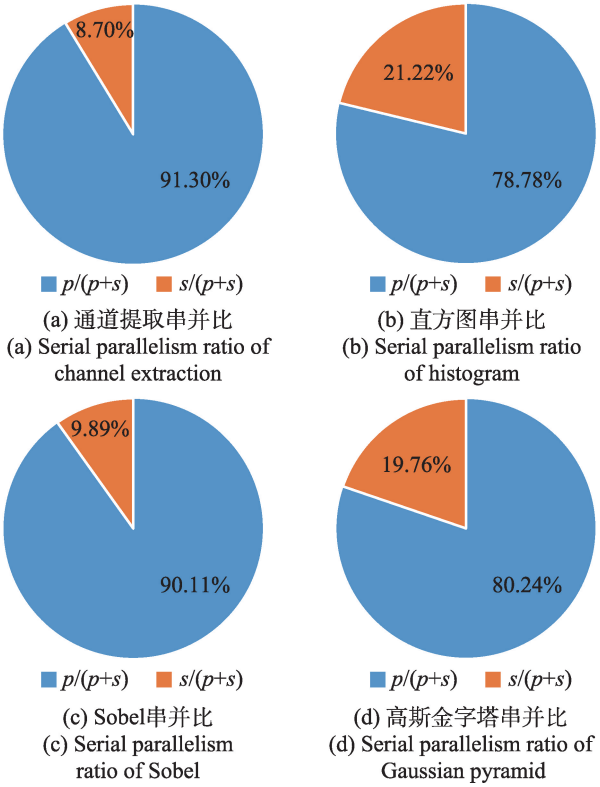


图15 不同函数串并比处理权重对比

Fig.15 Weight comparison of different functions serial parallelism ratio

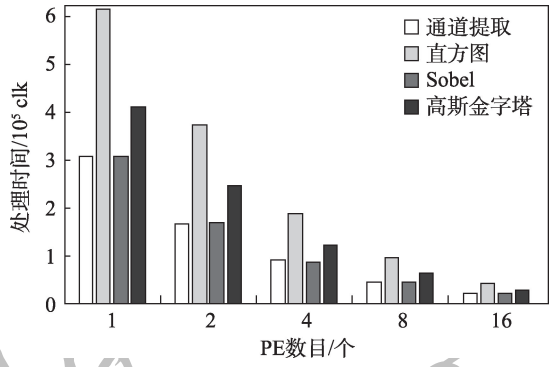
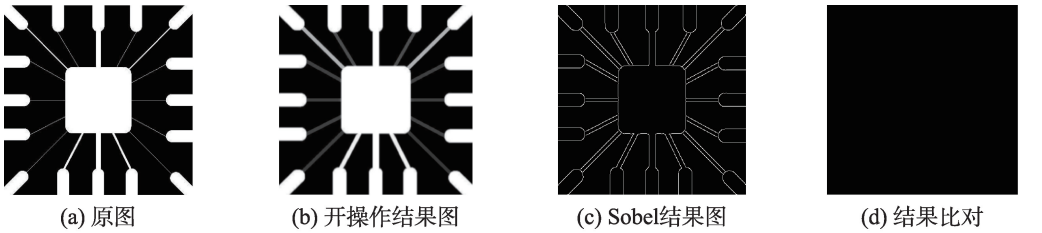


图16 不同函数处理时间对比

Fig.16 Comparison of processing time of different functions

图18为Graph单流水线处理图，T_{ld_reg}是加载PE内部寄存器堆时间(预处理)，T_{ld_ram}为加载PE间缓存时间，图像数据从Graph中的第一个节点流向最后一个节点并输出，整体流水实现耗时T_{all}为313 045 clk，相比逐函数串行处理(T_{all} = 1 843 860 clk)速度提升了4.89倍。



(a) 原图 (a) Source image　(b) 开操作结果图 (b) Open operation result image　(c) Sobel结果图 (c) Sobel result image　(d) 结果比对 (d) Comparison of results

图17　图像处理前后对比

Fig.17　Contrast before and after image processing

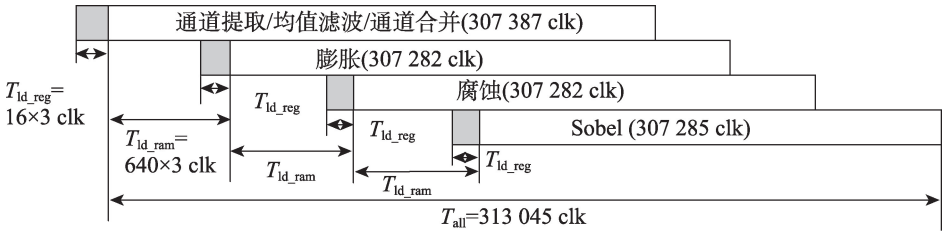


图18　Graph单流水线处理

Fig.18　Graph pipeline processing

启用不同数目的流水线对同一Graph进行处理，测试结果如表3所示，根据阿达姆定律计算了相应的加速比，如图19所示。当执行流水线数增加时，Graph的处理加速比成线性增长。

表3　不同数目流水线处理时间

Table 3　Processing time of different number of pipelines

流水线数目/个	处理时间/clk
1	313 045
2	220 454
3	132 086
4	92 072

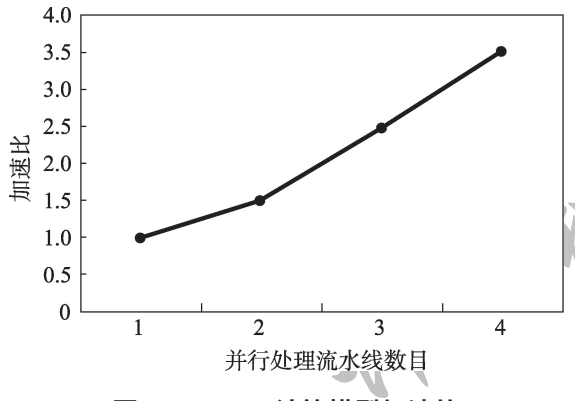


图19　Graph计算模型加速比

Fig.19　Graph execution model acceleration ratio

4.3　性能分析

4.3.1　与通用图像处理电路对比

文献[4,8]针对OpenVX 1.0标准中的核心库函数，提出了基于Mesh型结构的阵列处理器。通过改变源节点的数据注入率，对本文系统与文献[4,8]系统的平均延时和吞吐量进行了统计分析。测试中每个节点作为源节点注入数据的概率是相同的，通过一定约束，对目标节点的选择符合随机均匀分布。如图20所示，平均延时均随着数据注入率的增大而增大，由于HCCM+相比于Mesh型网络多了对角快速传输通道，本文平均延时整体小于Mesh型系统。

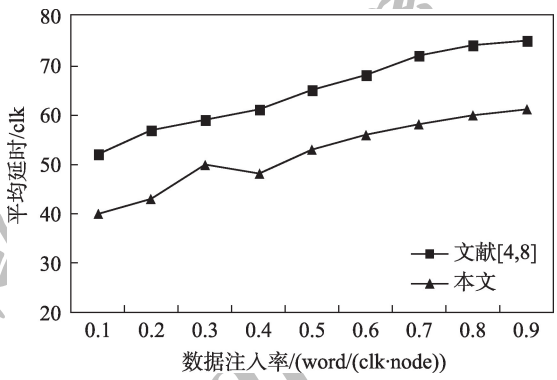


图20　平均延时对比

Fig.20　Average delay comparison

如图21所示，随着数据注入率的增大，网络负载增大，吞吐量也随之增大。当数据注入量增大到一定程度时，网络负载达到饱和，吞吐量处于稳定状态。由于本文系统中网络的对分宽度大于文献[4,8]，本文系统的吞吐量整体上更大，可以承载更大的数据流量。

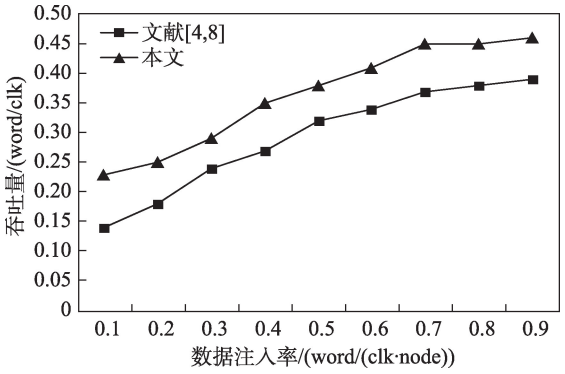


图21 吞吐量对比

Fig.21 Throughput comparison

针对相同Kernel函数,本文与文献[8]的加速比对比如图22所示,(a)、(b)和(c)分别是中值滤波、颜色转换及图像腐蚀的加速比对比。结果表明,在处理性能上,本文相比同类的图像阵列处理器,具有更大的加速比,更有优势。

4.3.2 与专用图像处理电路对比

文献[18-20]是基于FPGA对Sobel算法的并行化设计,实现了专用的图像处理。如表4所示,当本文调用的PE数目比较少时,处理速度小于专用硬件电路,但是随着调用PE数目的增加,处理速度明显提高,当采用16个PE进行映射时,处理速度大于专用硬件电路。本文与文献[18-20]资源占用对比如表5所示,本文资源占用高于其他三个设计,但系统工作频率更高,可支持更多的图像处理类型,更加通用。

表4 处理时间对比

Table 4 Comparison of processing time

比较项	FPGA型号	PE数目	处理时间/ms
本文	XCVU440-flga2892-2-e	1	2.46
		2	1.35
		4	0.69
		8	0.36
		16	0.16
文献[18]	XC3S2000	—	0.66
文献[19]	EP1K30TC144-1	—	6.30
文献[20]	CYCLONEEP1C20	—	1.30

表5 性能对比

Table 5 Performance comparison

比较项	LUT (look up table)	FF (filp flop)	Slice	频率/MHz	支持处理类型
本文	28 768	8 704	1 596	125	38
文献[18]	1 289	1 458	1 130	105	1
文献[19]	—	—	—	80	1
文献[20]	4 096	—	—	50	1

5 结束语

针对传统的处理器灵活性与处理速度不能兼顾的问题,本文设计实现了一种OpenVX并行处理器,不但在性能上接近于ASIC,而且具有灵活的可编程性,结构简单易扩展。并行处理器支持数据并行和管线处理两种计算方式,使用有限的硬件资源完成对OpenVX核心函数和复杂Graph执行模型的映射并且线性加速。对Ⅰ类函数的最大平均加速比为15.170,对Ⅱ类函数的最大平均加速比为14.825,对Ⅲ类函数的最大平均加速比为15.215,对Ⅳ类函数的最大平均加速比为14.810,能够有效地提高图像的处理速度,实现数据级并行和任务级并行。与同类阵列处理器相比,加速效果更加明显。今后研究工作重点是继续分析各个函数的处理瓶颈,找到更优的映射方式,优化最长路径。其次以本并行处理器为基本簇,结合更加有效的通信管理机制,对其进一步扩展,实现多并行处理器簇,进一步提高整个系统的性能。

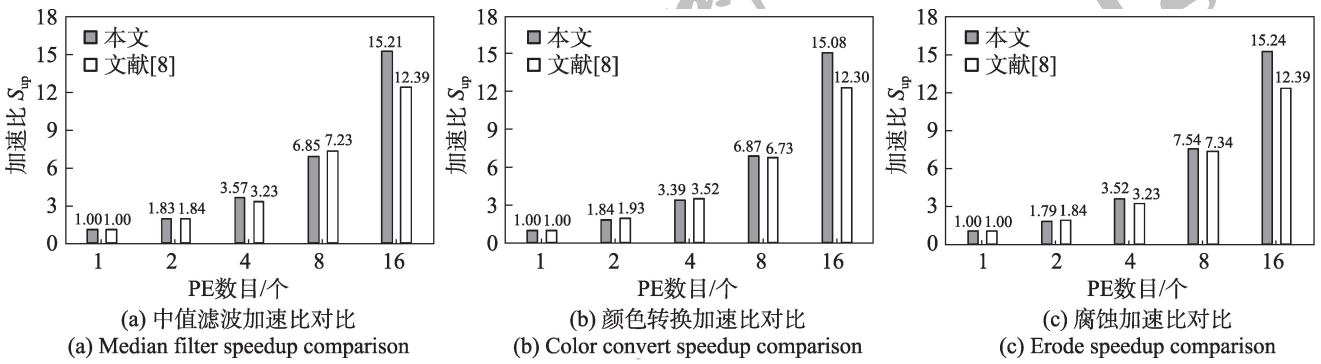


图22 核心函数加速比对比

Fig.22 Speedup comparison of kernel functions

参考文献：

[1] 李雅琪, 冯晓辉, 王哲. 计算机视觉技术的应用进展[J]. 人工智能, 2019(2): 18-27.
LI Y Q, FENG X H, WANG Z. Application progress of computer vision technology[J]. Artificial Intelligence View, 2019(2): 18-27.

[2] 山蕊, 李涛, 蒋林, 等. 视觉阵列处理器超越函数加速单元设计[J]. 西安电子科技大学学报(自然科学版), 2018, 45(4): 166-173.
SHAN R, LI T, JIANG L, et al. Design of the transcendental function computing unit of the computer vision array processor[J]. Journal of Xidian University (Natural Science), 2018, 45(4): 166-173.

[3] GOOSSENS G. 专用指令集处理器设计的架构性研究[J]. 中国集成电路, 2013, 22(10): 41-43.
GOOSSENS G. Research on architecture of special instruction set processor[J]. China Integrated Circuits, 2013, 22(10): 41-43.

[4] 李涛, 杨婷, 易学渊, 等. 萤火虫2: 一种多态并行机的硬件体系结构[J]. 计算机工程与科学, 2014, 36(2): 191-200.
LI T, YANG T, YI X Y, et al. Architecture of a polymorphous parallel computer[J]. Computer Engineering and Science, 2014, 36(2): 191-200.

[5] 孙建, 李涛, 李雪丹. 基于PAAG的图形图像算法的并行实现[J]. 计算机技术与发展, 2015, 25(11): 61-66.
SUN J, LI T, LI X D. Parallel implementation of graphics rendering and image processing algorithm based on PAAG[J]. Computer Technology and Development, 2015, 25(11): 61-66.

[6] The Khronos OpenVX Working Group. The OpenVX specification[EB/OL]. (2020-09-10)[2020-10-05]. https://www.khronos.org/registry/OpenVX/specs/1.3/html/.OpenVX_Specification_1_3.html.

[7] 王鹏博. 多态并行机上的OpenVX系统实现[D]. 西安: 西安邮电大学, 2015.
WANG P B. Implementation of OpenVX system on polymorphic parallel computer[D]. Xi'an: Xi'an University of Posts and Telecommunications, 2015.

[8] 李涛, 孙建, 王鹏博. 基于PAAG的OpenVX核心库函数并行化实现[J]. 西安邮电大学学报, 2015, 20(2): 7-10.
LI T, SUN J, WANG P B. Parallel implementation of kernels of OpenVX based on PAAG[J]. Journal of Xi'an University of Posts and Telecommunications, 2015, 20(2): 7-10.

[9] ZAHN F, LAMMEL S, FRÖNING H. On link width scaling for energy-proportional direct interconnection networks[J]. Concurrency and Computation: Practice and Experience, 2019, 31(2): 1-16.

[10] AKL S G. Parallel computation: models and methods[J]. IEEE Concurrency, 1997, 6(4): 79-80.

[11] SHANG J, SHENG D, LIU R, et al. Research on parallel task optimization of high performance computing cluster[C]// Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Information Systems, Dalian, Mar 20-22, 2020. Piscataway: IEEE, 2020: 777-780.

[12] RAO P S, YEDUKONDALU K. Hardware implementation of digital image skeletonization algorithm using FPGA for computer vision applications[J]. Journal of Visual Communication and Image Representation, 2019, 59: 140-149.

[13] 佟倩. 互联网络拓扑结构与鲁棒适变能力研究[D]. 沈阳: 沈阳理工大学, 2018.
TONG Q. Research on topology and robust adaptability of interconnected networks[D]. Shenyang: Shenyang University of Technology, 2018.

[14] PUNHANI A, KUMAR P, NITIN N. E-XY: an entropy based XY routing algorithm[J]. International Journal of Grid and Utility Computing, 2019, 10(2): 179-186.

[15] 付涛. 高速图像处理算法研究与实现[D]. 绵阳: 西南科技大学, 2016.
FU T. Research and implementation of high speed image processing algorithm[D]. Mianyang: Southwest University of Science and Technology, 2016.

[16] 李海玲, 张昊. 卷积边界扩展研究与实现[J]. 微型电脑应用, 2018, 34(10): 47-49.
LI H L, ZHANG H. Research and implementation of convolution boundary extension[J]. Microcomputer Applications, 2018, 34(10): 47-49.

[17] AL-HAYANNI M A N, XIA F, RAFIEV A, et al. Amdahl's law in the context of heterogeneous many-core systems—a survey[J]. IET Computers & Digital Techniques, 2020, 14(4): 133-148.

[18] 林源晟. 基于FPGA的图像边缘检测系统设计[D]. 西安: 西安电子科技大学, 2014.
LIN Y S. Design of image edge detection system based on FPGA[D]. Xi'an: Xi'an University of Electronic Science and Technology, 2014.

[19] 艾扬利, 杨兵. 基于FPGA的Sobel算子并行计算研究[J]. 现代电子技术, 2005, 28(9): 42-43.
AI Y L, YANG B. Study of FPGA-based parallel processing of Sobel operator[J]. Modern Electronic Technology, 2005, 28(9): 42-43.

[20] 官鑫, 王黎, 高晓蓉, 等. 图像边缘检测Sobel算法的FPGA仿真与实现[J]. 现代电子技术, 2009, 32(8): 109-111.
GUAN X, WANG L, GAO X R, et al. Emulation and realization of Sobel edge detection algorithm based on FPGA [J]. Modern Electronic Technology, 2009, 32(8): 109-111.

潘风蕊(1996—), 女, 陕西渭南人, 硕士研究生, 主要研究方向为集成电路系统设计。
PAN Fengrui, born in 1996, M.S. candidate. Her research interest is integrated circuit system design.

李涛(1954—),男,北京人,博士,教授,CCF会员,主要研究方向为计算机体系结构、计算机图形学、大规模集成电路等。

LI Tao, born in 1954, Ph.D., professor, member of CCF. His research interests include computer architecture, computer graphics, large-scale integrated circuit, etc.

邢立冬(1980—),男,山东人,博士,高级工程师,CCF会员,主要研究方向为集成电路系统设计。

XING Lidong, born in 1980, Ph.D., senior engineer, member of CCF. His research interest is integrated circuit system design.

张好聪(1996—),女,陕西渭南人,硕士研究生,主要研究方向为集成电路系统设计。

ZHANG Haocong, born in 1996, M.S. candidate. Her research interest is integrated circuit system design.

吴冠中(1995—),男,陕西西安人,硕士研究生,主要研究方向为集成电路系统设计。

WU Guanzhong, born in 1995, M.S. candidate. His research interest is integrated circuit system design.