doi:10.3772/j.issn.1002-0470.2021.08.001

面向深度可分离卷积的硬件高效加速器设计①

许浩博②*** 王 颖* 王郁杰* 张士长*** 刘博生*** 韩银和③*

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院大学 北京 100190)

摘 要 采用深度可分离(DS)卷积替代标准卷积已成为神经网络轻量化设计的趋势,但是由于深度可分离卷积不规则的数据维度和数据尺寸,现有卷积神经网络加速器在处理这类网络时计算并行度和计算单元(PE)利用率无法得到保证,导致加速器计算性能降低。针对这一问题,本文提出一种通道朝向的计算数据流,该数据流能够将数据维度不同的 Depthwise 卷积、Pointwise 卷积和标准卷积在统一的数据流下展开运算。基于该数据流,设计了一款面向深度可分离卷积的加速器,该加速器采用统一的计算核心处理深度可分离卷积中各类卷积运算,在低面积开销下实现了高计算并行度。实验结果表明,与目前现有的深度可分离卷积加速器相比,该设计获得了 1.32 倍处理速度和 1.76 倍面积效率的提升。

关键词 深度可分离(DS)卷积;加速器;低面积;低延迟;利用率

0 引 言

卷积神经网络(convolutional neural network,CNN)在图像分类、物体检测、动作识别、自然语言处理等众多场景得到广泛应用,推动着人工智能领域的革命性发展[1-3]。但是,卷积神经网络是计算密集型模型,并且随着处理任务的日益复杂、追求的处理精度不断增高,卷积神经网络的发展呈现出层数越来越深、结构越来越复杂、计算量和参数量越来越大的趋势,这对在各类计算平台、尤其是计算资源受限的嵌入式设备和边缘设备中部署卷积神经网络构成了巨大挑战[4-5]。

为了设计轻量化的卷积神经网络,深度可分离(depthwise separate, DS)卷积在近期成为了研究热点。研究学者通过采用深度可分离卷积层替代网络中标准卷积层,大幅降低了网络的规模和计算量,并且能够实现与标准卷积网络相近的计算精度[6-9]。例如,MobileNet[6]作为一种轻量级卷积网络包含大量深度分离卷积结构,其计算量和参数量为 VGG-16[4]的 1/27 和 1/32,但计算精确度仅损失 1%。深度可分离卷积网络的使用为嵌入式设备高效部署各类深度学习任务提供了一种有效途径。

然而,深度可分离卷积实现的网络轻量化是以牺牲网络规整性为代价的,这对日益兴起的神经网络专用加速器设计带来了巨大挑战[10-12]。深度可分离卷积层把标准卷积分解为 Depthwise 卷积层和 Pointwise 卷积层两个阶段,其中 Depthwise 卷积将卷积核分别作用于每个特征图输入通道,因此与标准卷积相比,Depthwise 卷积降低了数据通道方向的维度;Pointwise 卷积与标准卷积类似,但是卷积核尺寸变为 1×1。数据维度和网络尺寸的差异导致神经网络专用加速器无法充分利用这类网络低参数量和

① 国家重点研发计划(2017YFB1301100),国家自然科学基金(61834006, 61874124),北京市科技计划(Z171100000117019, Z181100008918006)和中国科学院战略重点研究计划(XDPB12)资助项目。

② 男,1990 年生,博士生;研究方向:计算机体系结构,神经网络加速器设计;E-mail: xuhaobo@ict.ac.cn

③ 通信作者,E-mail: yinhes@ict.ac.cn

(收稿日期:2020-07-18)

低计算量的特点，其计算并行性和数据重用性无法得到保证，造成计算延迟增加。例如，Dadiannao[10]和张量处理单元（tenser processing unit，TPU）[11]利用输入和输出通道的计算并行性提高计算吞吐量，然而在处理 Depthwise 卷积层时，由于数据通道维度的降低，没有足够的数据能够映射到计算单元阵列中，导致计算单元利用率降低；又如 Eyeriss[12]能够并行处理多个神经元，但是处理 Pointwise 卷积层时，由于片上带宽不足，会出现计算单元等待数据的情况，造成计算效率的降低。

为解决该问题，本文深入研究了深度可分离卷积的数据维度和计算特征，提出了一种通道朝向的计算数据流，该数据流能够将数据维度不同的 Depthwise 卷积、Pointwise 卷积和标准卷积在统一的数据流下展开运算。基于该数据流，设计了一款面向深度可分离卷积的加速器，该加速器采用统一的计算核心处理 Depthwise 卷积和 Pointwise 卷积层等卷积运算，并且能够实现高计算并行度和高计算单元利用率。评估结果表明，与目前最先进的深度可分离 CNN 加速器相比，该设计获得了 1.32 倍加速比和 1.76 倍的面积效率提升。

1 背 景

1.1 深度分离卷积神经网络

深度可分离卷积的结构示意图如图 1 所示，深度可分离卷积将常规的卷积运算分为了 Depthwise 卷积运算和 Pointwise 卷积运算两个阶段。Depthwise 卷积运算过程中，特征图中各个通道独立进行卷积运算，各个通道的卷积结果不再像标准卷积那样在通道方向求和。Pointwise 卷积运算本质是标准卷积，通过建立 M 个 $1\times1\times C$ 的卷积完成卷积核大小为1×1的标准卷积。与标准卷积相比，深度可分离卷积减少了所需运算量和参数量。例如，对于具有 N 个输入通道且卷积核尺寸为 $K\times K$ 网络层来说，若采用深度分离卷积结构，其参数量和运算量仅为标准卷积层的 $(1/N+1/K2)$[6]。

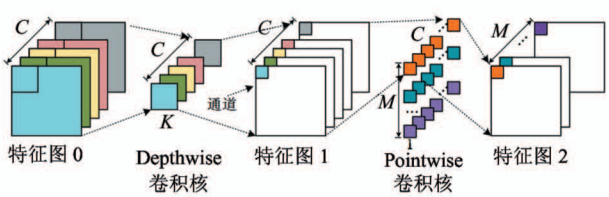

图 1 深度可分离卷积结构示意图

1.2 相关工作及存在的问题

尽管深度可分离卷积与标准卷积相比计算量和参数量大幅降低，但是其数据维度和尺寸的不规则性导致神经网络加速器中计算单元的利用率不高，无法保证高并行度计算。学者们针对这一问题开展了一系列研究。Wu 等人[13]提出了一种能够高效处理 MobileNet 网络（MobileNet 中最主要的结构是深度可分离卷积）的加速器，该加速器针对 Depthwise 卷积层和 Pointwise 卷积层分别部署了两个专用计算引擎，并针对 Pointwise-Deptwise 网络层顺序做了精细化的流水调度设计。这种“分而治之”的设计策略尽管能够增加加速器的吞吐率，但是增加了电路规模和设计复杂度。Bai 等人[14]提出了一种兼容 Depthwise 卷积层和 Pointwise 卷积层的矩阵乘法引擎，并利用分层的存储结构缓解带宽压力。但是所提出的结构仅适用于卷积核大小 3×3 的 Depthwise 卷积，但处理具有其他卷积核尺寸的网络（如 Mobilenet v3[8]中尺寸为 5×5 的 Depthwise 卷积层）时效率较低。

不难看出，现有深度可分离卷积加速器的设计多以牺牲硬件开销和功能灵活性为代价，而对于嵌入式系统和边缘系统来说，神经网络必须在严格的时序约束和资源约束下稳定运行，因此在有限硬件资源下如何实现深度可分离卷积的低延时处理是深度可分离卷积加速器设计亟待解决的难题。针对这一问题，本文通过分析 Depthwise 卷积层和 Pointwise 卷积层的计算特点，设计了一套能够高效处理这两类卷积运算的统一计算架构。该设计与现有技术方案相比，具有占用资源小、计算单元利用率高以及处理延迟低的特点，能够在计算资源受限的设备实现深度可分离卷积神经网络低延时处理。

2 计算数据流设计

深度可分离卷积中数据尺寸和通道维度差异很

大,因此处理这类网络结构需要一种灵活高效的计算数据流。为此,本文设计了一种通道朝向的计算数据流,如图 2 所示。在输入特征图的通道维度上,将深度为 C 的通道划分为不同的通道组,每个通道组包含 Tc 个特征图通道,在每个计算批次处理一个通道组。每个通道组又进一步划分为多个通道向量,每个通道向量加载至同一个计算单元(processing element,PE)中。在 PE 单元内部,PE 以通道朝向串行处理通道向量内的激活值。在输入特征图的平面维度上,将大小为 $W \times H$ 输入特征图划分为大小为 $Tw \times Th$ 特征图,多个 PE 并行处理特征图内每个激活值。这样的数据流有两点好处,首先,输入特征图中不同通道的计算过程在时间维度上相对独立,这样消除了 Depthwise 卷积和 Pointwise 卷积在通道维度上的差异;其次,这种“局部串行,全局并行”的策略保证了系统的吞吐量。

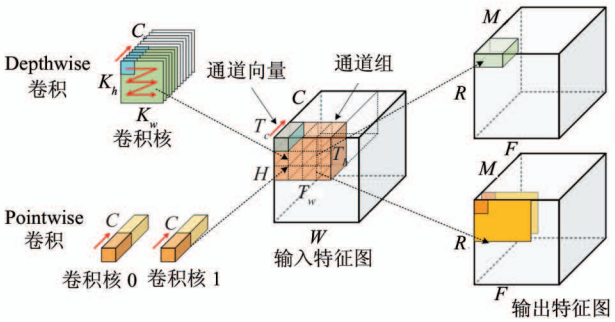

图 2　通道朝向的计算数据流示意图

图 3 是本文设计的通道朝向计算数据流的循环嵌套(loop nests)伪代码。在 PE 内循环中,输出部分和(partial sum)是以通道朝向的方式串行计算的,并且部分和将暂存在 PE 内部;在 PE 间循环中,采用空间平铺策略,多个 PE 并行处理多个激活值。循环套嵌包括 3 个可变循环参数 Kh、Kw 和 C,通过改变这 3 个参数的数值,不同的卷积类型能够在统一的数据流下展开运算。当参数 C 的数值为 1 时,即输入特征图中激活值和权重的乘积不会在通道方向上累加,适用于 Depthwise 卷积运算;当参数 Kh 和 Kw 等于 1 时,卷积核的尺寸是 1 × 1,适用于 Pointwise 卷积运算;当参数没有特别限制时,适用于标准卷积运算。通过这样的方式,不同的卷积类型可以按照相同的循环顺序处理,这为在统一的数据流下设计低控制开销、低硬件开销的硬件高效加速器提供了机会。

```
Parallel_for (r=0; r<R; r++){                                  PE间循环
Parallel_for (f=0; f<F; f++){
    for (kh=0; kh<Kh; k++){                                    PE内循环
        for(kw=0; kw<Kw; k++){
            for(m=0; m<M; m++){
                for(c=0; c<C; c++){                            //Channel-oriented
                    O[r][f][m]+= I[r+s*kh][f+s*kw][c]*W[kh][kw][m][c]
}}}}}}
```

图 3　通道朝向的计算数据流伪代码

然而,面向通道的数据流虽然统一了不同卷积类型的循环套嵌计算顺序,但是该数据流中可变的循环参数使得计算过程中的数据依赖变得复杂,造成加速器中 PE 利用率得不到保证:首先,PE 内循环的最内层循环参数 C 以及卷积核尺寸 Kh 和 Kw 为 1 时,需要频繁存储和加载输入激活值和部分和,破坏了数据局部性;其次,PE 间循环需要并行处理多个激活值,这对数据带宽的需求急剧增加。在第 3 节和第 4 节中,本文将介绍如何通过计算架构和存储架构设计来支持本文提出的通道朝向的计算数据流,并能够在处理不同卷积类型时均具备高 PE 利用率。

3　结构设计

本文设计的深度可分离卷积加速器整体架构如图 4 所示,其中计算部件由 4 个 PE 计算引擎组成,每个 PE 计算引擎独立处理一个通道组,多个 PE 计算引擎并行处理多个通道组。存储架构分为 3 个层次,包括片外动态随机存取存储器(dynamic random access memory, DRAM)存储、片上缓存(用于存储权重和输入/输出神经元)以及每个PE中的本地寄

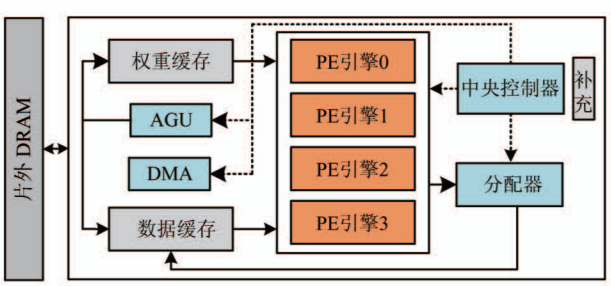

图 4　电路整体结构

存器。中央控制单元(central control unit,CCU)根据加载的控制指令生成控制信号,地址生成单元(address generation unit,AGU)根据控制信号自动地从存储器中读取数据并分发至 PE 中。调度器(dispatcher)从 4 个 PE 计算引擎中收集计算结果,完成计算后处理,并将最终结果加载到片上缓冲中。

3.1　支持通道朝向数据流的 PE 单元架构设计

PE 单元的功能是处理图 3 中的 PE 内循环,并将计算结果输出至调度器进行后处理,其具体结构如图 5 所示,主要包括乘法器、累加器、本地多通道寄存器(LCReg)和本地激活值寄存器(LAReg)等部件。其中,乘法器完成激活值和权重的乘法运算并生成部分积,累加器累加部分积得到部分和,本地多通道寄存器用来缓存部分和,本地激活值寄存器用来缓存输入激活值。本地多通道寄存器的设计同时考虑了 Depthwise 卷积和 Pointwise 卷积的数据局部性,在 Depthwise 卷积阶段,LCReg 将不同通道的部分和分别存储在独立的寄存器中,因此图 3 中 PE 内循环不会因输入通道维度 C 的降低而破坏计算数据流;在 Pointwise 卷积阶段,不同输出通道的部分和缓存到单独的 LCReg 中,充分挖掘了部分和的可重用性。值得注意的是,在每个周期每个 PE 内最多访问一个 LCReg,而其他 LCReg 只消耗静态功耗,因此由 LCRegs 引入的功耗开销是可以接受的。

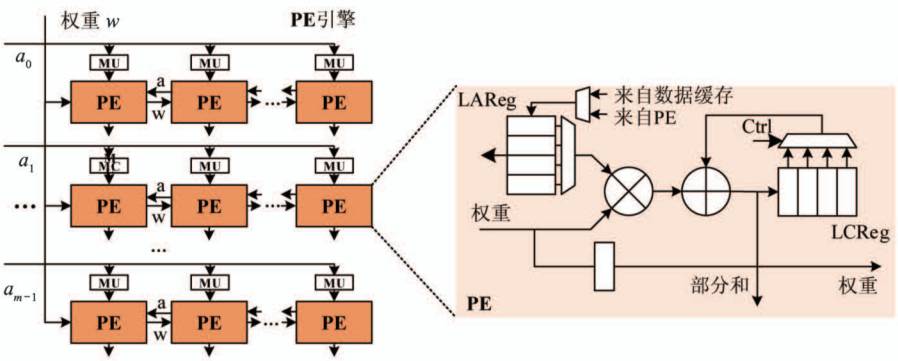

图 5　PE 结构示意图

$m \times n$ 个 PE 单元通过互连结构组成 PE 引擎,并行处理特征图平面方向上的乘加(multiply and accumulate,MAC)运算,如图 5 所示。权重加载至 PE 引擎的最左列,并且能够从左向右在 PE 单元间传播,每一列的 PE 单元共享同一个权重。激活值在 PE 引擎的每一行能够通过广播方式分发至任意一列 PE 中,激活值在分发过程中附带一个列 ID 标签,匹配单元(match unit,MU)能够通过 ID 标签判断所需接收的激活值,通过这样的机制激活值能够准确载入至任意一列计算单元中。激活值的载入在多个 PE 单元行中并行执行,多个激活行能够共享同一个 ID,因此 ID 标签的带宽开销可以忽略不计。

3.2　PE 间通信设计

如 3.1 节所述,权重能够在行方向通过 PE 间互连实现权重在 PE 间的复用。进一步分析卷积运算规则和本文提出的通道朝向的数据流后发现,卷积核在输入特征图平面内滑动时,滑动窗口间存在数据重叠,因此能够进一步挖掘重叠部分中输入激活值的计算复用。如图 5 所示,本文通过片间互连使得输入激活值能够向左传播至相邻的 PE 单元,这种机制带来了两个好处。一方面,输入激活值能够在每一行的 PE 单元间重用,减少了 PE 和片上缓存之间的访存通信量;另一方面,PE 间通信能够快速填充 PE 计算所需的数据,无须请求访问片上存储器,能够避免由带宽不足和复杂访存模式导致的流水线停滞问题。

3.3　片上存储结构设计

3.1 小节介绍了 PE 内本地存储结构,本小节重点介绍片上数据缓存的存储结构和存储拓扑,如图 6所示。片上数据缓存被分为 4 个 Bank,与 4 个 PE 引擎相对应,每个 Bank 为一个 PE 引擎提供计算所需要的数据;每一个 Bank 被分为 Tm 个存储块,每个存储块能够存储 Th 个元素。在读存储器阶段,PE 引擎中每一列 PE 需要处理的数据来自于同

一数据缓存行,其中同一个通道的数据又来自于同一数据缓存行中同一个存储块。在写存储器阶段,Depthwise 卷积和 Pointwise 卷积略有差异。对于 Depthwise 卷积,写操作和读操作是完全的镜像过程,每一列 PE 的计算结果存储至同一数据缓存行中,不同列的计算结果存储在不同数据缓存行中,并能够通过地址索引;Pointwise 卷积由于需要在通道方向求和,因此来自 4 个 PE 引擎的计算结果首先会先载入到调度器中的加法树组,求和后得到同一个输出通道的激活值,存入至一个存储块中。“按块存储”仅需要简单的译码电路即可实现,无须复杂的面积开销;数据的存储和载入由 CCU、AGU 和分配器配合完成。通过精细的存储结构设计,不同卷积类型的存储访问变得规整而统一,降低了访存调度的复杂度并能够充分利用传输带宽,提高了计算效率。

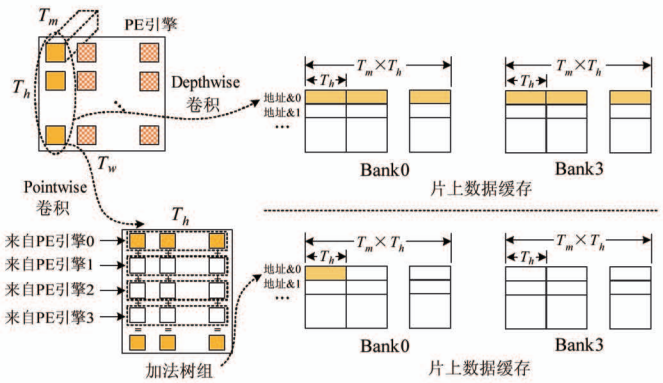

图 6　片上存储结构

4　工作流程和数据映射方式

本文将通过 2×2 的 PE 计算引擎阐述所提加速器如何利用通道朝向的计算数据流处理 Depthwise 卷积和 Pointwise 卷积。

4.1　Depthwise 卷积运算的数据映射方式与计算过程

本文构造了一个小尺寸的 Depthwise 卷积层来阐述数据映射方式和工作流程,待处理的输入特征图大小为 3×3×3,卷积核大小为 2×2×3,3 个通道的卷积过程彼此独立,卷积步进(stride)为 1。详细的数据映射和工作流程如图 7 所示。

周期#0　输入特征图的输入值 $Ax(x=0,1,2)$、$Dx(x=0,1,2)$ 和权重 $wa0$ 加载至 PE 引擎中最左侧的一列计算单元 PE00 和 PE10 中,同一列的 PE 共享相同的权重 $wa0$。此时,PE00 和 PE10 计算得到输出特征图通道 0 中 O00 和 O10 的部分和(为方便描述,将此周期定义为“通道 0 执行阶段”),得到的部分和存储在通道 0 的本地寄存器 LCReg0 中,以供后续计算使用。输入值 $Ax(x=0,1,2)$ 和 $Dx(x=0,1,2)$ 暂存在计算单元中的 LAReg 中。

周期#1　输入特征图的激活值 $Bx(x=0,1,2)$ 和 $Ex(x=0,1,2)$ 加载至第 2 列计算单元 PE01 和 PE11 中,PE00 和 PE10 中的权重 $wa0$ 向右传播到第 2 列 PE 中,第 2 列中的 PE 进入“通道 0 执行阶段”,输入神经元 B0、E0 与权重 $wa0$ 相乘,乘积在累加器中累加后得到输出激活值 O01 和 O11 的部分和并存储在 LCReg0 中。与此同时,PE00 和 PE10 进入“通道 1 执行阶段”,权重通道 1 的元素 $wa1$ 输入至第 1 列 PE00 和 PE10 中,输入神经元 A1、D1 和权重 $wa1$ 相乘,乘积在累加器中累加后存储在通道 1 的本地寄存器 LCReg1 中。通过这样的方式,“通道 1 执行阶段”的计算结果不会冲刷掉“通道 0 执行阶段”存储在 LAReg0 的计算结果,不同通道的计算能够以互不干扰的方式展开,同时不破坏图 2 和图 3 所描述的数据流。输入值 $Bx(x=0,1,2)$ 和 $Ex(x=0,1,2)$ 暂存在计算单元 PE01 和 PE11 中的 LAReg。

周期#2　与周期#1 类似,第 1 列和第 2 列中的 PE 分别对输入特征图中通道 3 和通道 2 中的激活值开展运算。权重通道向量 $wax(x=0,1,2)$ 通过 3 个周期已完全加载至 PE 引擎中。

周期#3　中央控制单元 CCU 产生控制信号能使 PE 间通信，并将 PE01（PE11）中的激活值 $Bx(Ex)$ 传递到左侧相邻 PE 中，从而相邻两个卷积滑动窗口中重叠部分（激活值 Bx 和 Ex）在 PE 单元间实现了数据重用，减少了对片上存储器的访问。同时，一个全新的权重通道向量开始载入至 PE 引擎，$wb0$ 首先载入至最左一列 PE 单元中，PE00 和 PE10 的回滚到“通道 0 执行阶段”，权重 $wb0$ 和激活值 $B1$ 相乘，处理图 3 中第 4 行所描述的卷积核内循环。

每个 PE 运行 12 个周期后即可计算得到输出特征图中的所有元素。期间，处理的 MAC 总数为 48，因此计算加速比为 48（MAC 总数）/12（周期数）=4，与 PE 个数一致。因此，在此例中基于通道朝向的计算数据流和硬件结构设计，Depthwise 卷积运算的 PE 利用率能够达到 100%。

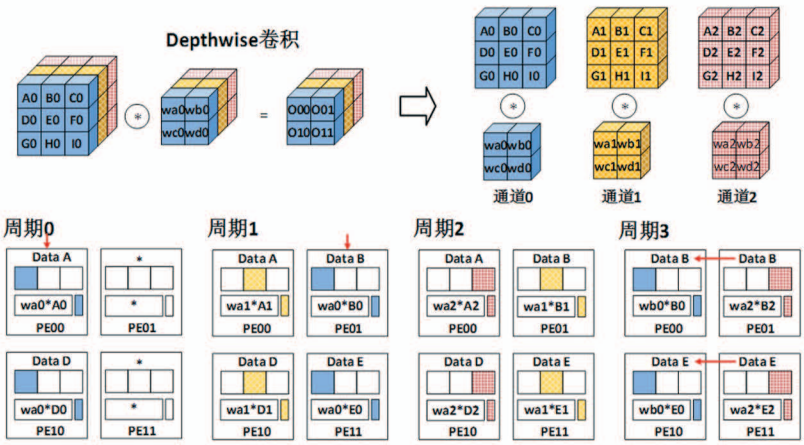

图 7　Depthwise 卷积数据映射示意图

4.2　Pointwise 卷积运算的数据映射方式与计算过程

本文采用 3×3×3 大小的输入特征图和 1×1×3 大小的卷积核介绍 Pointwise 卷积的计算过程。本文描述具有代表性的前 4 个计算周期。

周期#0 到#2　激活值和权重值按照与 Depthwise 卷积过程相同的顺序载入至计算单元中，而与 Depthwise 卷积不同的是，Pointwise 卷积需要将不同输入通道计算得到的部分和进一步求和得到卷积结果。因此在周期#0 到#2，每个 PE 工作在“通道 0 执行阶段”，处理图 3 中最内层循环，生成 O00、O10、O30 和 O40 的部分和存储在 LCReg0 中，如图 8 所示。

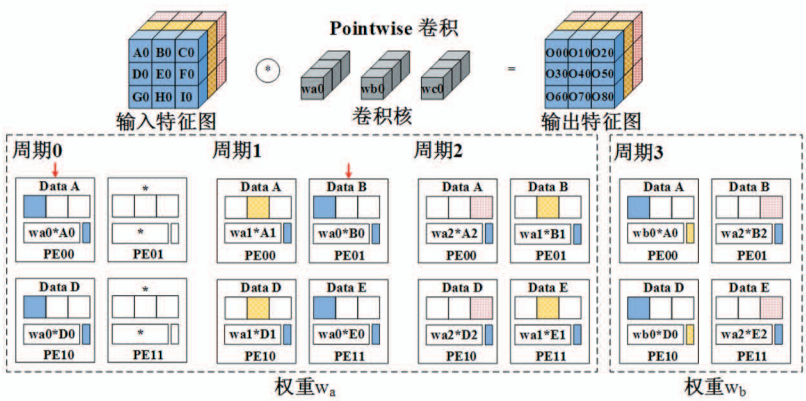

图 8　Pointwise 卷积数据映射示意图

周期#3　第 2 个卷积核中的 $wb0$ 加载到 PE 中，PE00 和 PE10 进入“通道 1 执行阶段”，PE00 和 PE10 计算得到的部分和存储在 LCReg1 中。PE01 和 PE11 依旧工作在“通道 0 执行阶段”，计算得到

O10 和 O40 的部分和存储在通道 0 本地寄存器 LCReg0 中。

通过这样的计算方式,Pointwise 卷积过程中 PE 也能够得到充分利用,并与 Depthwise 卷积运算保持了相同的输入数据流,因此本文提出的加速器能够在统一的计算架构中几乎无差异地处理这两类卷积,实现了低控制开销与低硬件开销下深度可分离卷积的高效运算。此外在 Depthwise 卷积和 Pointwise 卷积计算过程中,本文提出的电路结构能够利用不同的数据重用模式,如输出数据重用、权重重用(权重被多个激活值重复使用)和输入激活值重用(输入激活值在滑动窗口间重复使用)。通过高效的数据重用,本文提出的设计能够以较低的片上存储带宽为 PE 提供计算所需要的数据,避免 PE 由数据等待而产生的计算停顿。

5 实验和结果

5.1 实现方法

本文将所提出的加速器结构用 Verilog 实现,采用 Synopsys Design Compiler 综合工具在台积电 65 nm工艺下进行综合,使用 Synopsys VCS 仿真工具对所提出的加速器进行了仿真与验证,使用 Synopsys PrimeTime 对功耗进行了评估。加速器共包含 196 个 PE,单个 PE 的通道处理能力 Tc 为 8。

5.2 PE 利用率和性能评估

MobileNet 网络和 MobileNet v2 网络内包含大量深度可分离卷积结构,本文评估了加速器在处理 MobileNet 网络和 MobileNet v2 网络时的硬件利用率。评估结果表明,加速器在处理这两个网络时整体利用率分别能够达到 96.39% 和 92.1%,对于 MobileNet 网络,本文提出的加速器在处理卷积核大小为 3×3、步进为 1 的网络层时,能够实现接近 100% 的 PE 利用率,PE 的空闲主要来自于 PE 进入流水线进程所消耗的等待周期。对于 MobileNet v2 中卷积核大小为 3×3、步进为 1 的网络层,本文提出的加速器同样能够实现接近 100% 的 PE 利用率,但是在处理通道维度较低的网络层时 PE 利用率略有降低。原因是多个 PE 引擎具有并行处理多个通道组的能力,一个通道组内又包含多个通道,MobileNet v2 包含通道数较低的网络层,当网络层通道数低于通道处理能力时,PE 利用率就会降低。在保证高 PE 利用率的情况下,本文提出的加速器在处理 MobileNet[6] 和 MobileNet v2[7] 时分别能够达到 136.3 fps 和 205.3 fps 的处理性能,这表明本文提出的设计能够在极低延迟下处理深度可分离卷积,满足了实时处理需求。

5.3 面积和功耗评估

本文设计实现的加速器整体电路面积为 3.12 mm^2,各个模块的面积占比如图 9(a)所示。由于本文提出的加速器能够将 Depthwise 卷积、Pointwise 卷积和常规卷积按照统一的访问形式载入至计算单元中,因此控制开销较低,CCU 和 AGU 仅占总面积的 1.54% 和 1.85%。片上 SRAM 用来缓存激活值、权重值以及中间计算结果,占电路总面积的 65.12%。片上 SRAM 容量的选取是电路面积和性能的折中,在设计阶段考虑减小 SRAM 容量以进一步降低面积开销,但是这种策略会加剧片外访存,甚至会产生严重的数据等待,导致计算延迟降低。

在 500 MHz 工作频率下,本文提出的加速器处理 Mobilenet v2 网络的平均功耗为 190.31 mW,各个模块的功耗占比如图 9(b)所示。PE 占据了电路绝大部分的功耗,主要原因有两点,首先,PE 频繁执行 MAC 运算,PE 部件的动态功耗很高;其次,本文提出的通道朝向数据流和相对应的计算架构能够有效挖掘数据局部性和 PE 引擎内的数据重用,降低了片上存储带宽,减少了 SRAM 的动态功耗。

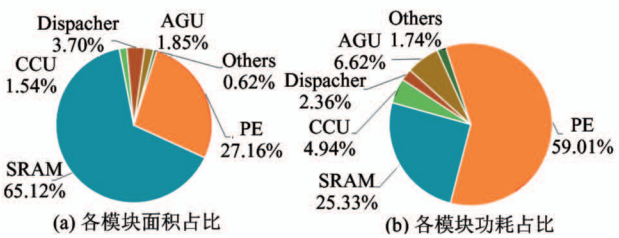

图 9 模块面积与功耗占比

5.4 性能比较

本文将提出的加速器与现有的面向深度可分离卷积神经网络的加速器[13,14] 设计进行了比较。本

文选择 MobileNet[6]、MobileNet v2[7]、MobileNet v3[8]和 Xception[9]等典型深度可分离卷积模型作为测试基准。为了公平比较，本文在同等工艺条件下实现了文献[13]和文献[14]的设计，并在评估过程中调整了文献[13]和文献[14]的工作频率，使其与本文提出的设计保持一致。文献[13]中实现了小规模设计(文献[13]-S)和大规模设计(文献[13]-L)两个版本。为充分比较，本文也实现了具有 1568 个 PE 的高性能设计版本(Proposed-L)。归一化后的计算性能比较结果如图 10 所示，Proposed-L 具有最高的计算性能，与文献[13]-L 相比计算速度提高了 1.32 倍。文献[13]提出的设计在处理 Pointwise-Depthwise 的网络组合时能够实现高计算性能，但是 Xception 中深度可分离卷积层的顺序是 Depthwise-Pointwise，因此文献[13]中定制化设计的流水线不能发挥最大性能。文献[14]不能高效处理尺寸是 5×5 的卷积核，因此在处理 MobileNet v3 网络时性能损失较大。

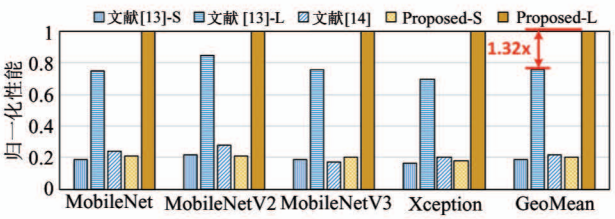

图 10　性能比较

为了进一步分析在有限计算资源下各个设计的性能，本文进一步评估了面积效率(性能/面积)，各个设计的归一化面积效率比较如图 11 所示。本文设计的加速器能够采用统一的计算架构处理 Depthwise 和 Pointwise 等多种卷积类型，面积开销下降且在计算过程中计算单元利用率处于较高水平，因此本文提出的设计与文献[13]-S相比面积效率平均提高1.76倍，本文的高性能设计 Proposed-L 也实现了比文献[13]-L 更高的面积效率。实验结果表明，本文提出的加速器能够在资源受限和面积受限的条件下实现出色的处理性能，为 IoT 系统和边缘计算设备处理深度可分离卷积提供了一种有效解决方案。

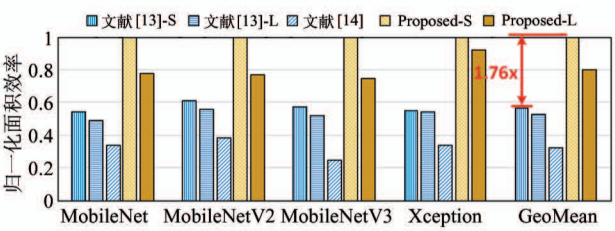

图 11　面积效率

6　结　论

采用深度可分离卷积代替标准卷积已成为构建轻量化卷积神经网络的主流方法，但是现有的神经网络加速器没有充分挖掘深度可分离卷积低参数量和低计算量的特点，无法高效处理这类卷积类型。针对这一问题，首先通过设计通道朝向的计算数据流，统一了 Depthwise 卷积和 Pointwise 卷积的计算过程；接着基于该数据流，设计了兼容 Depthwise 卷积和 Pointwise 卷积的加速器，该加速器能够使用统一的计算架构实现不同卷积类型的高并行度计算，不再需要为不同的卷积类型设计独立的计算引擎，从而避免了额外的面积开销。特别地，本设计通过 PE 间通信和合理存储架构设计，充分挖掘了数据重用，在低传输带宽的情况下保证了计算单元的高利用率。实验结果表明，本文提出的设计在较小的面积内实现了深度可分离卷积的低延迟处理，为在计算资源有限的嵌入式系统和边缘计算设备上部署深度可分离卷积提供了有效的手段。

参考文献

[1] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]// European Conference on Computer Vision, Zurich, Switzerland, 2014: 818-833

[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]// Advances in Neural Information Processing Systems, Lake Tahoe, USA, 2012: 1097-1105

[3] Tran D, Wang H, Torresani L, et al. A closer look at spatiotemporal convolutions for action recognition[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 6450-6459

[4] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. *arXiv*:1409.1556, 2014

[5] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, USA, 2015: 3431-3440

[6] Howard A G, Zhu M, Chen B, et al. Mobilenets: efficient convolutional neural networks for mobile vision applications[J]. *arXiv*:1704.04861, 2017

[7] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 4510-4520

[8] Howard A, Sandler M, Chu G, et al. Searching for mobilenetv3[C]//Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 2019: 1314-1324

[9] Chollet F. Xception: Deep learning with depthwise separable convolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 2017: 1251-1258

[10] Chen Y, Luo T, Liu S, et al. Dadiannao: a machine-learning supercomputer[C]//The 47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014: 609-622

[11] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit[C]//Proceedings of the 44th Annual International Symposium on Computer Architecture, Toronto, Canada, 2017: 1-12

[12] Chen Y H, Krishna T, Emer J S, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. *IEEE Journal of Solid-State Circuits*, 2016, 52(1): 127-138

[13] Wu D, Zhang Y, Jia X, et al. A high-performance CNN processor based on FPGA for MobileNets[C]//The 29th International Conference on Field Programmable Logic and Applications, Barcelona, Spain, 2019: 136-143

[14] Bai L, Zhao Y, Huang X. A CNN accelerator on FPGA using depthwise separable convolution[J]. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2018, 65(10): 1415-1419

Hardware efficient accelerator for depthwise separate convolution

Xu Haobo[*,**], Wang Ying[*], Wang Yujie[*], Zhang Shichang[*,**], Liu Bosheng[*,**], Han Yinhe[*]

([*]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]University of Chinese Academy of Sciences, Beijing 100190)

Abstract

Recent advances in convolutional neural networks (CNNs) reveal the trend towards designing compact structures such as depthwise separable(DS) convolution. However, the diverse data dimensions of depthwise and pointwise in DS convolution increases the difficulty of data-parallel computing, which incurs performance loss due to the decline in the processing element(PE) utilization. To overcome this problem, a novel channel-oriented dataflow is proposed to unify the computing dataflow of Depthwise convolution and Pointwise convolution. Based on the proposed dataflow, a compact CNN accelerator is developed that can process the Depthwise and Pointwise convolution in a unified processing core with high PE utilization. The experimental results show that the proposed accelerator achieves 1.32× speedup and 1.76× area efficiency compared with the state-of-the-art depthwise separable CNN accelerator for the evaluated workloads.

Key words: depthwise separate (DS) convolution, accelerator, low area, low latency, utilization