基于FPGA的卷积神经网络和视觉Transformer通用加速器

李天阳[①] 张 帆[*①] 王 松[②] 曹 伟[③] 陈 立[①]

[①](信息工程大学信息技术研究所 郑州 450002)

[②](空军95072部队 南宁 530000)

[③](复旦大学大数据研究院 上海 200433)

摘 要：针对计算机视觉领域中基于现场可编程逻辑门阵列(FPGA)的传统卷积神经网(CNN)络加速器不适配视觉Transformer网络的问题，该文提出一种面向卷积神经网络和Transformer的通用FPGA加速器。首先，根据卷积和注意力机制的计算特征，提出一种面向FPGA的通用计算映射方法；其次，提出一种非线性与归一化加速单元，为计算机视觉神经网络模型中的多种非线性和归一化操作提供加速支持；然后，在Xilinx XCVU37P FPGA上实现了加速器设计。实验结果表明，所提出的非线性与归一化加速单元在提高吞吐量的同时仅造成很小的精度损失，ResNet-50和ViT-B/16在所提FPGA加速器上的性能分别达到了589.94 GOPS和564.76 GOPS。与GPU实现相比，能效比分别提高了5.19倍和7.17倍；与其他基于FPGA的大规模加速器设计相比，能效比有明显提高，同时计算效率较对比FPGA加速器提高了8.02%～177.53%。

关键词：计算机视觉；卷积神经网络；Transformer；FPGA；硬件加速器

中图分类号：TP331; TN47 **文献标识码**：A **文章编号**：1009-5896(2022)00-0001-10

DOI: 10.11999/JEIT230713

FPGA-Based Unified Accelerator for Convolutional Neural Network and Vision Transformer

LI Tianyang[①] ZHANG Fan[①] WANG Song[②] CAO Wei[③] CHEN Li[①]

[①](*Institute of Information Technology*, *Information Engineering University*, *Zhengzhou* 450002, *China*)

[②](*The* 95072 *Unit of PLA Air Force*, *Nanning* 530000, *China*)

[③](*Institute for Big Data*, *Fudan University*, *Shanghai* 200433, *China*)

Abstract: Considering the problem that traditional Field Programmable Gate Array (FPGA)-based Convolutional Neural Network(CNN) accelerators in computer vision are not adapted to Vision Transformer networks, a unified FPGA accelerator for convolutional neural networks and Transformer is proposed. First, a generalized computation mapping method for FPGA is proposed based on the characteristics of convolution and attention mechanisms. Second, a nonlinear acceleration unit is proposed to provide acceleration support for multiple nonlinear operations in computer vision networks. Then, we implemented the accelerator design on Xilinx XCVU37P FPGA. Experimental results show that the proposed nonlinear acceleration unit improves the throughput while causing only a small accuracy loss. ResNet-50 and ViT-B/16 achieved 589.94 GOPS and 564.76 GOPS performance on the proposed FPGA accelerator. Compared to the GPU implementation, energy efficiency is improved by a factor of 5.19 and 7.17, respectively. Compared with other large FPGA-based designs, the energy efficiency is significantly improved, while the computing efficiency is increased by 8.02%～177.53% compared to other FPGA accelerators.

Key words: Computer vision; Convolutional Neural Network (CNN); Transformer; Field Programmable Gate Array (FPGA); Hardware accelerator

收稿日期：2023-07-15；改回日期：2023-09-27；网络出版：2023-10-08

*通信作者： 张帆 17034203@qq.com

基金项目：国家重点研发计划(2022YFB4500900)

Foundation Item: National Key R&D Program of China (2022YFB4500900)

1 引言

在过去的10年中，卷积神经网络(Convolutional Neural Network, CNN)在计算机视觉领域取得了巨大成功，如VGG[1], ResNet[2]和GoogleLeNet[3]。由于最近几年Transformer在自然语言处理领域取得了突破性的成果，基于注意力机制的Transformer[4]也成为计算机视觉领域研究热点。越来越多的视觉Transformer和CNN-Transformer混合模型被提出[6-8]，如用注意力机制代替卷积的视觉转换器(Vision Transformer, ViT)[5]，或者将卷积与注意力机制相结合的检测转换器(DEtection TRansformer, DETR)[6]。这些模型在各类计算机视觉任务中均取得了出色的成绩。

随着基于注意力机制的Transformer模型规模逐渐增大，计算量也变得越来越大，如ViT-H/14的参数量为632 MB[5]，之后发布的ViT-G/14参数量和计算量分别增长到1843 MB和2859.9 GFLOPs[8]。由于Transformer中的注意力机制涉及多个大型矩阵乘法和数据交互，因此在硬件平台上部署Transformer需要高额的计算和存储开销。现有的基于现场可编程逻辑门阵列(Field Programmable Gate Array, FPGA)的视觉Transformer加速器试图从两个方面克服这个问题，包括数据流优化和模型优化。文献[9]将ViT的注意力模块和多层感知机(Multi-Layer Perceptron, MLP)模块分别与层归一化函数进行组合，划分为两部分在FPGA上进行映射，通过平衡每个模块的吞吐量，减少了残差机制带来的路径依赖影响，整个加速器使用全流水线设计。文献[10]使用3个单独的计算阵列并行计算注意力机制中的Qurey($\boldsymbol{Q}$), Key($\boldsymbol{K}$)和Value($\boldsymbol{V}$)，除此之外的矩阵乘法操作与多层感知器时分复用其他计算阵列。文献[11]采用定点和指数两种方式对视觉Transformer进行混合精度量化，通过对FPGA的计算资源总数和目标帧率(Frames Per Second, FPS)建模，训练得到定点量化和指数量化的占比。

然而，这些FPGA加速器不能充分适用于CNN，例如文献[9]采用定制全流水设计，文献[10]为不同的操作分配不同的计算资源，CNN则有着更多的权重共享机制，FPGA加速器通常使用迭代式设计来复用计算资源，而且针对CNN和Transformer的FPGA加速器之间有着不同的并行计算和访存策略。吴瑞东等人[12]面向极致边缘计算场景，提出一种不依赖片外存储的多核FPGA加速器，片上存储采用紧密耦合内存层次结构设计，使用层内映射和层内融合优化方法来减小存储资源消耗。文献[13]则通过为使用残差机制的CNN的Shortcut数据分配可重用的静态存储，最大化复用读入片上的数据，降低片外访存量。因此，可以看出两种神经网络的FPGA加速器设计思路存在差异性，使用某一种专用加速器将不可避免地降低另一方的计算效率，给面向计算机视觉多样化应用的云数据中心加速器部署带来了挑战。

为解决上述问题，本文基于FPGA提出了一种CNN和视觉Transformer通用加速器。本文的主要研究工作及贡献如下：(1)根据注意力机制和卷积的计算特征，面向FPGA提出一种通用的计算映射方法，对计算阵列进行设计空间探索；(2)提出一种支持Softmax、高斯误差线性单元(Gaussian Error Linear Unit, GELU)和层归一化的硬件加速单元，通过指令配置复用计算和存储资源，在同等面积条件下实现了更高的处理效率；(3)在Xilinx XCVU37P FPGA上对加速器进行了综合实现，同时支持CNN和视觉Transformer两类神经网络加速。在ImageNet数据集上的实验结果表明，所提出的非线性与归一化加速单元在提高吞吐量的同时仅造成很小的精度损失。ResNet-50和ViT-B/16在所提FPGA加速器上的吞吐量分别达到了589.94 GOPS和564.76 GOPS，能效比远高于GPU实现和其他基于FPGA的大规模加速器设计，且计算效率高于对比FPGA加速器。

2 研究背景

2.1 卷积和注意力机制概述

卷积是传统CNN最核心的组成部件，同时也是CNN模型中计算量最大的操作，卷积计算过程如图1所示。卷积使用卷积核在输入特征图的2维空间上滑动计算每个像素点及其相邻像素点与卷积核的点积，最终得到一张输出特征图。图1中的C_{in}和C_{out}分别表示卷积操作的输入通道数和输出通道数。K表示卷积核大小。H_{in}和W_{in}表示输入特征图的高度和宽度。H_{out}和W_{out}表示输入特征图的高度和宽度。

注意力机制最关键的部分是自注意力操作，自注意力计算流程如图2(a)所示。X_{in}是自注意力操作的输入，分别与权重W_Q, W_K和W_V作线性矩阵

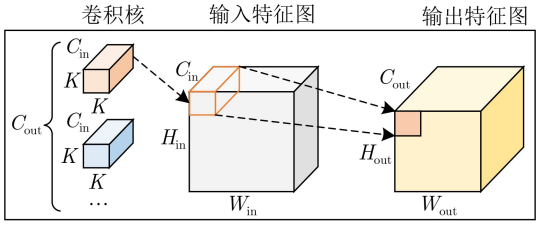
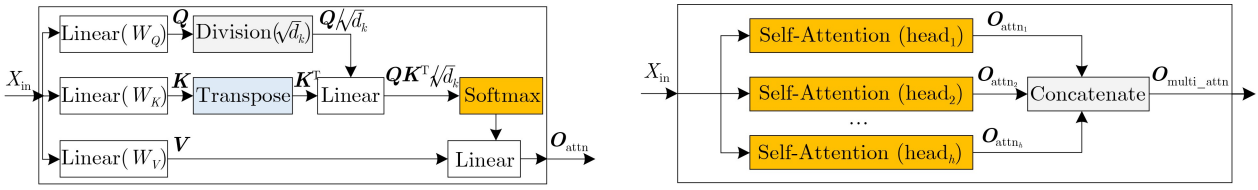


图 1 卷积示意图



(a) 自注意力　　(b) 多头自注意力

图 2 注意力计算过程

乘法得到$\boldsymbol{Q}$, $\boldsymbol{K}$和$\boldsymbol{V}$。为避免Softmax函数的输入绝对值过大，导致函数的梯度过小进而影响到梯度下降。将$\boldsymbol{Q}$除以$\sqrt{d_k}$进行缩放，d_k是输入向量维度。缩放后的$\boldsymbol{Q}$与$\boldsymbol{K}$的转置$\boldsymbol{K}^{\mathrm{T}}$相乘，然后进行Softmax操作，结果与$\boldsymbol{V}$相乘得到自注意力最终结果$\boldsymbol{O}_{\mathrm{attn}}$。现有的基于注意力机制的Transformer模型大多数使用多头自注意力，由多个自注意力操作组成，数量用Head(h)表示。图2(b)展示了Transformer中多头自注意力的计算过程。每个头使用不同的权重W_{Q_i}, W_{K_i}和W_{V_i}来分别得到$\boldsymbol{Q}_i$, $\boldsymbol{K}_i$和$\boldsymbol{V}_i$。最后将h个自注意力输出拼接为$\boldsymbol{O}_{\mathrm{multi_attn}}$作为多头自注意力的最终结果。

2.2 问题分析

2.2.1 计算方式差异

由上述可知，卷积和注意力机制遵循着不同的设计范式。卷积使用卷积核在感受野上计算得到输出值，感受野是指输出特征图上的像素点在原始图像上映射的区域。相比之下，注意力机制则是对输入特征图及其中间结果进行操作，专注于不同的区域，捕捉更多的特征信息。在推理过程中，卷积计算过程中只涉及输入数据和训练后的模型参数，而注意力计算过程还包括对当前输入的中间结果进行操作，如$\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$等。另外，与注意力中的单通道2维矩阵乘法不同，卷积计算通常是多通道的。

2.2.2 非线性与归一化操作差异

CNN和Transformer模型除了计算密集的线性操作，还包括激活和归一化函数。CNN通常采用批归一化，由于批归一化在推理时可以作为固定权重叠加进卷积层，同时CNN频繁使用的激活函数ReLU可以在FPGA上使用简单的查找表实现，其他非线性操作通常只在输出层进行，因此CNN中的非线性与归一化操作基本不会影响其在FPGA上推理的计算效率。Transformer通常采用层归一化，层归一化的对象是单个样本，它对该样本的所有维度的特征进行操作。层归一化使用输入数据的均值和方差作为参数，与输入数据强相关。除了层归一化，Transformer还包括频繁使用的Softmax和GELU激活函数，这些操作在推理时引入了大量时间成本，在CPU上推理一张图像的运行时间占比如图3所示。

2.2.3 问题小结

由于CNN和Transformer在计算方式和非线性与归一化操作上存在着以上差异，给当前的计算机视觉神经网络FPGA加速器设计带来了以下挑战。

(1) 如何在FPGA计算资源上同时映射计算方式不同的卷积和注意力机制；

(2) 如何在保证精度的前提下设计非线性与归一化加速单元以支持Transformer中的层归一化，Softmax和GELU激活函数加速；

(3) 基于上述两个挑战，如何将基于CNN和Transformer的算法模型映射到FPGA上进行加速。

3 加速器设计

3.1 卷积和注意力机制计算映射

为了最大限度地复用FPGA的计算和存储资源，本文将卷积和注意力机制进行统一映射，这使得两种类型的操作可以使用相同的计算阵列完成。这需要对维度大小不同的卷积和注意力机制中的矩阵乘法操作进行划分，将这些大规模计算映射成计算阵列上的有限次计算[14]。

本文对卷积操作的划分如图4(a)所示，其中d_{ic}, d_{iw}和d_{ih}分别表示输入特征图划分块的通道数、宽度和高度，d_{oc}, d_{ow}和d_{oh}分别表示输出特征图划分块的通道数、宽度和高度。d_{ow}和d_{oh}可以根据d_{iw}, d_{ih}以及卷积核大小和步长得到，因此划分块

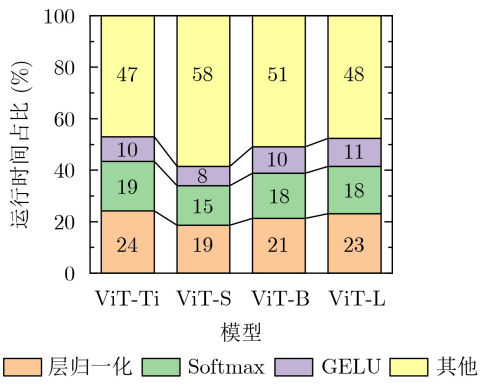
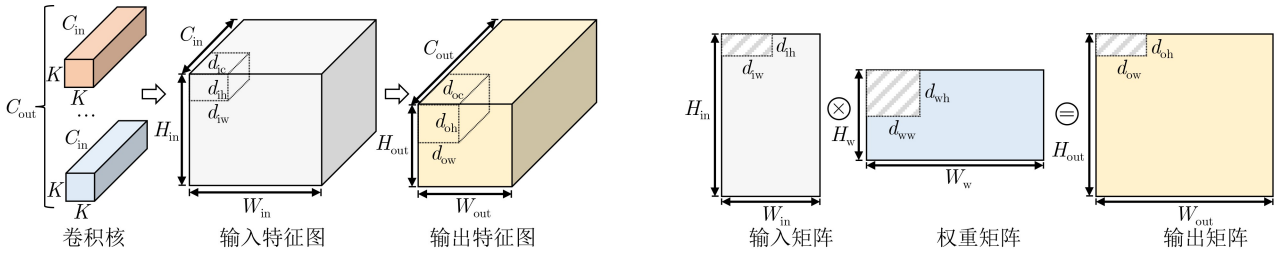


图 3 ViT各种配置下的运行时间占比



(a) 卷积划分 (b) 注意力矩阵划分

图 4 卷积和注意力矩阵乘法划分

可以使用4元组$(d_{\mathrm{ic}}, d_{\mathrm{oc}}, d_{\mathrm{ow}}, d_{\mathrm{oh}})$来表示。本文对注意力机制中矩阵乘法操作的划分如图4(b)所示，由于$d_{\mathrm{oh}} = d_{\mathrm{ih}}$，因此使用3元组$(d_{\mathrm{iw}}, d_{\mathrm{ow}}, d_{\mathrm{oh}})$表示矩阵乘法划分块。这些参数均由CNN和Transformer模型各层特征图尺度以及FPGA计算阵列规模决定。

FPGA计算阵列通常采用由若干个计算单元组成的二叉树结构或脉动阵列结构，本文用(N_i, N_o)来参数化计算阵列配置，N_i表示每个计算单元的输入通道数，N_o表示计算单元的数量，即计算阵列输出通道数。本文将(N_i, N_o)与$(d_{\mathrm{ic}}, d_{\mathrm{oc}}, d_{\mathrm{ow}}, d_{\mathrm{oh}})$和$(d_{\mathrm{iw}}, d_{\mathrm{ow}}, d_{\mathrm{oh}})$前两个维度对应。对于卷积，对划分块逐通道计算，$N_i$对应$d_{\mathrm{ic}}$，$N_o$对应$d_{\mathrm{oc}}$。对于注意力机制中的矩阵乘法，对划分块逐行计算，$N_i$对应$d_{\mathrm{iw}}$，$N_o$对应$d_{\mathrm{ow}}$。

为了探索计算阵列配置(N_i, N_o)，本文以计算量最小化为目标，提出了片上模型计算量穷举法。通过分析基于FPGA的CNN和Transformer模型在计算阵列各种配置下实际的计算量得到(N_i, N_o)最优解，如算法1所示。由于神经网络每层的最终计算结果总是作为下一层的输入，为了减少无效数据填充，避免降低计算效率，本文约束$\mathrm{mod}(N_i, N_o) = 0$。对于每组有效的$(N_i, N_o)$，遍历神经网络模型每层的卷积或注意力操作，通过计算阵列利用率n_{iu}和n_{ou}得到实际的计算量。

由于CNN各层通道数和Transformer输入向量维度通常是2的整数次幂的整数倍，本文将资源受限的FPGA计算阵列中的乘法器数量$\mathrm{Mult}_{\mathrm{num}}$设置为256, 512, 1 024和2048, 得到的(N_i, N_o)有效配置如表1所示。基于这些配置，本文对典型的CNN模型ResNet-50和典型的Transformer模型ViT-B/16应用算法1，得到的各种计算阵列配置下模型实际的计算量如图5所示。当$\mathrm{Mult}_{\mathrm{num}} = 256$时，$(N_i, N_o)$配置为(16, 16)使得ResNet-50和ViT-B/16实际的计算量同时达到最小；当$\mathrm{Mult}_{\mathrm{num}} = 512$时，$(N_i, N_o)$取(16, 32)是最优解；当$\mathrm{Mult}_{\mathrm{num}} = 1024$时，$(N_i, N_o)$可配置为(32, 32)；当$\mathrm{Mult}_{\mathrm{num}} = 2048$时，$(N_i, N_o)$配置为(32, 64)是最优解。

算法1 片上模型计算量穷举法

输入：神经网络模型Model, 乘法器数量$\mathrm{Mult}_{\mathrm{num}}$

输出：(N_i, N_o)设计空间探索结果，片上模型计算量$\mathrm{Comp}_{\mathrm{num}}$

(1) 初始化(N_i, N_o),$\mathrm{Comp}_{\mathrm{num}}$

(2) for $i = 0; i < \lfloor\sqrt{\mathrm{Mult}_{\mathrm{num}}}\rfloor; i{+}{+}$ do:

(3) $\quad n_i = i + 1$

(4) $\quad n_o = \left\lfloor \dfrac{\mathrm{Mult}_{\mathrm{num}}}{n_i} \right\rfloor$

(5) $\quad$ if $\mathrm{mod}(N_o, N_i) == 0$ then

(6) $\quad\quad (N_i, N_o).\mathrm{append}(n_i, n_o)$

(7) $\quad$ end if

(8) end for

(9) for each(n_i, n_o)in(N_i, N_o)do

(10) $\quad \mathrm{comp}_{\mathrm{total}} = 0$

(11) $\quad$ for each layer in Model do

(12) $\quad\quad$ ic, oc, $\mathrm{madd}_{\mathrm{num}}$= layer.configuration

(13) $\quad\quad n_{\mathrm{iu}} = \mathrm{mod}(\mathrm{ic}, n_i) == 0\ ?\ 1\ :\ \dfrac{\mathrm{mod}(\mathrm{ic}, n_i)}{n_i}$

(14) $\quad\quad n_{\mathrm{ou}} = \mathrm{mod}(\mathrm{oc}, n_o) == 0\ ?\ 1\ :\ \dfrac{\mathrm{mod}(\mathrm{oc}, n_o)}{n_o}$

(15) $\quad\quad \mathrm{comp}_{\mathrm{total}} {+}{=} \mathrm{madd}_{\mathrm{num}}/(n_{\mathrm{iu}} \times n_{\mathrm{ou}})$

(16) $\quad$ end for

(17) $\quad \mathrm{Comp}_{\mathrm{num}}.\mathrm{append}(\mathrm{comp}_{\mathrm{total}})$

(18) end for

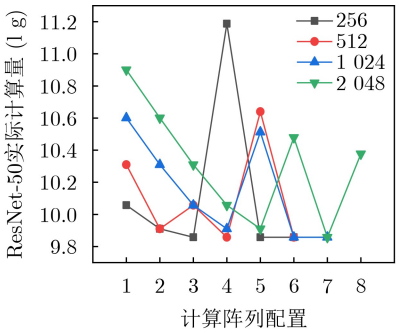
(19) return(N_i, N_o), $\mathrm{Comp}_{\mathrm{num}}$

表 1 计算阵列有效配置

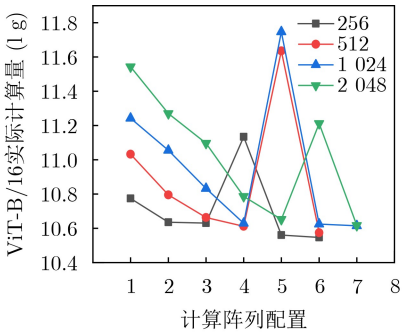
(N_i, N_o)	256	512	1 024	2048
1	(1, 256)	(1, 512)	(1, 1 024)	(1, 2 048)
2	(2, 128)	(2, 256)	(2, 512)	(2, 1 024)
3	(4, 64)	(4, 128)	(4, 256)	(4, 512)
4	(6, 42)	(8, 64)	(8, 128)	(8, 256)
5	(8, 32)	(13, 39)	(13, 78)	(16, 128)
6	(16, 16)	(16, 32)	(16, 64)	(26, 78)
7	/	/	(32, 32)	(32, 64)
8	/	/	/	(45, 45)

3.2 非线性与归一化加速单元

为了避免溢出和可能的精度损失，视觉Transformer中的Softmax函数可以表达为



(a) ResNet-50实际计算量



(b) ViT-B/16实际计算量

图 5 面向ResNet-50和ViT-B/16的计算阵列设计空间探索

$$\mathrm{Softmax}(x_{i,j})=\frac{\mathrm{e}^{x_{i,j}-x_{\max_i}}}{\sum_{j=1}^{n}\mathrm{e}^{x_{i,j}-x_{\max_i}}} \tag{1}$$

其中，$x_{i,j}$表示输入数据，$x_{\max_i}$表示沿列方向n个输入的最大值。

视觉Transformer中的GELU函数使用式(2)近似

$$\mathrm{GELU}(x_{i,j})=0.5x_{i,j}\cdot\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}(x_{i,j}+0.044715x_{i,j}^{3})\right)\right) \tag{2}$$

层归一化函数可表示为

$$\mathrm{LN}(x_{i,j})=\alpha_j\times\frac{x_{i,j}-\mu_i}{\sqrt{{\sigma_i}^2+\varepsilon}}+\beta_j \tag{3}$$

其中，μ_i表示第i行输入的均值，${\sigma_i}^2$表示第i行输入的方差，ε是一个非常小的常量，以避免分母为零，α_j和β_j分别表示第j列的权重和偏置。

由于在FPGA上直接计算e^x会导致巨大的硬件开销，因此本文使用式(4)来近似计算e^x。当n取2的整数次幂时，除法操作和幂运算就可以转换成移位操作以节省计算成本

$$\mathrm{e}^x=\lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n \tag{4}$$

为了确定合理的n值，本文在ViT-B/16模型中Softmax和GELU涉及e^x计算的输入范围内进行测试，将n分别设置为64, 128, 256, 512, 1 024，得到的结果如图6所示。当n=128时，近似函数基本与e^x函数一致，随着n值增大，相似度越来越大。为了在保证模型精度的前提下尽可能减少计算量，本文将n设置为128，在4.2节展示了n=128时详细的精度测试结果。

本文设置$g(x)=(1+x/128)^{128}$，对于Softmax函数，可以将式(1)表示为

$$\mathrm{Softmax}(x_{i,j})=\frac{g(x_{i,j}-x_{\max_i})}{\sum_{j=1}^{n}g(x_{i,j}-x_{\max_i})} \tag{5}$$

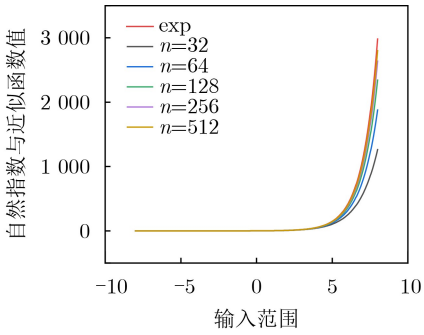
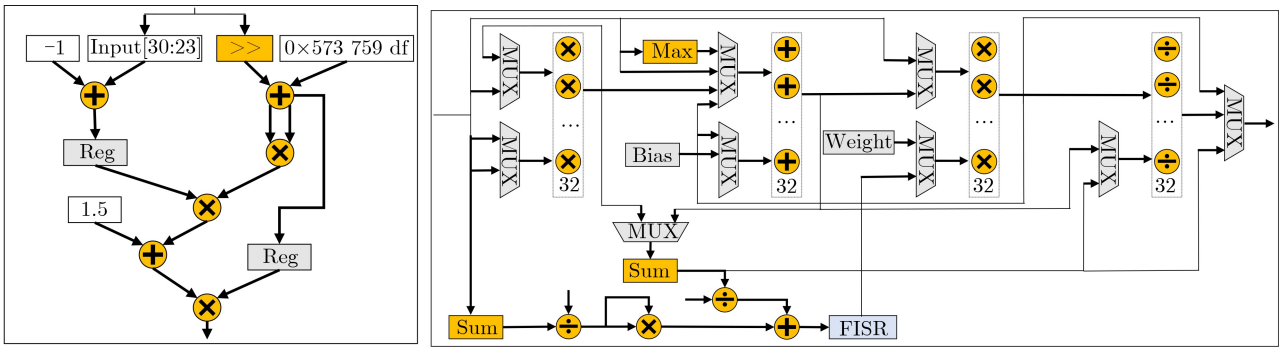


图 6 自然指数近似函数对比

对于GELU函数，设置$h(x)=\sqrt{\frac{2}{\pi}}(x_i+0.044715{x_i}^3)$，式(2)可表示为

$$\begin{aligned}\mathrm{GELU}(x_{i,j})=&0.5x_{i,j}\left(1+\frac{\mathrm{e}^{h(x_{i,j})}-\mathrm{e}^{h(x_{i,j})}}{\mathrm{e}^{h(x_{i,j})}+\mathrm{e}^{h(x_{i,j})}}\right)\\=&0.5x_{i,j}\left[1+\left(1-\frac{2}{\mathrm{e}^{2h(x_{i,j})}+1}\right)\right]\\=&x_{i,j}\left(1-\frac{1}{\mathrm{e}^{2h(x_{i,j})}+1}\right)\\=&x_{i,j}\left(1-\frac{1}{g(2h(x_{i,j}))+1}\right)\end{aligned} \tag{6}$$

对于层归一化函数，在FPGA上计算方差${\sigma_i}^2$的倒数平方根会导致较大的硬件开销，因此本文使用快速倒数平方根算法(Fast Inverse Square Root, FISR)[15]，分解其计算流程进行硬件适配，硬件结构如图7(a)所示。本文对上述变换后的Softmax, GELU和层归一化进行操作分解，尽可能地复用FPGA计算资源。非线性与归一化加速单元结构如图7(b)所示，主要使用了2组32路输入的乘法器，1组32路输入的加法器和1组32路输入的除法器，2组32路输入的加法树，在运行中通过指令配置实现Transformer中Softmax, GELU和层归一化3种函数的计算。为了在保持精度的前提下进一步降低资源消耗，除了使用32位单精度浮点数计算方差的倒数平方根，非线性与归一化加速单元使用16位半精度浮点数计算。



(a) FISR硬件结构 (b) 总体结构

图 7 非线性与归一化加速单元

3.3 加速器架构

本文提出的CNN和Transformer通用FPGA加速器架构如图8所示。为了简洁明了，省略了底层逻辑和存储资源的互连。本文通过主机解析网络结构描述文件，对神经网络模型进行划分，确定调度顺序，并生成映射指令。预编译的映射指令、预处理的划分块在运行前通过基于高速串行计算机扩展总线标准(Peripheral Component Interface experss, PCIe)的直接内存访问(Direct Memory Access, DMA)存储在片外动态随机存取存储器(Dynamic Random Access Memory, DRAM)。全局控制器接收预编译的映射指令来驱动加速器，将指令发送到片上存储和计算阵列等模块控制数据处理，或将数据请求发送到DRAM读写数据。

在FPGA设计中，使用一个大规模的设计等效替代多个小规模的设计可以降低设计复杂度和控制难度。因此，在本文的加速器设计中，尽可能选择使用大规模计算阵列，将(N_i, N_o)配置为(32, 64)。基于DSP时钟倍频和INT8乘法并行计算方法[16,17]，

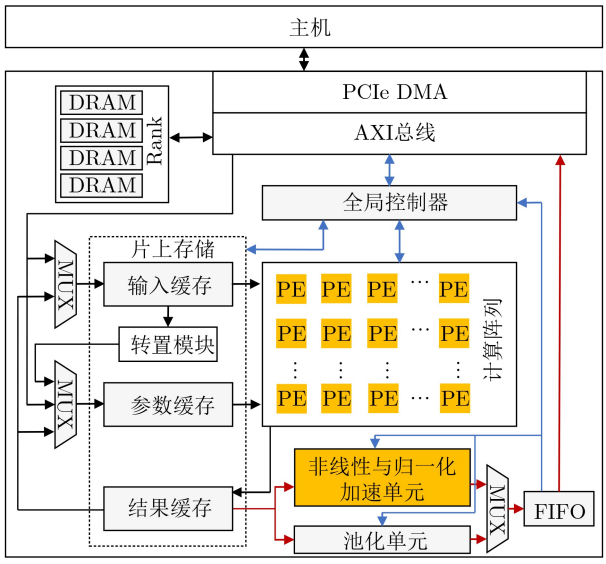


图 8 加速器架构

1个DSP可以在每个系统时钟周期内完成4次乘法操作。因此当(N_i, N_o)配置为(32, 64)时，计算阵列的乘加器只需消耗512个DSP，每个DSP负责4个输出通道方向上的计算。在计算阵列中嵌入了分布式存储(Look-Up-Table Random Access Memory, LUTRAM)来尽可能地复用权重，减少片外访存量。

预处理的划分块在运行时从DRAM中提取到输入缓存或参数缓存。计算阵列从这些缓存读取数据和参数进行计算。结果缓存负责存储中间或最终结果。当该层包括非线性或归一化或池化操作时，计算结果从结果缓存读入非线性与归一化加速单元或池化单元进行处理，完成后写入DRAM或写回输入缓存或参数缓存进行下一步计算。本文通过映射指令配置了加速器的数据路径和计算并行性。

4 实验与分析

4.1 实验环境

本文所设计的加速器目标器件为Xilinx Virtex Ultrascale+ XCVU37P FPGA，主机是一台配置多核Intel Xeon Gold 5 218 CPU的服务器，CPU主频为2.3 GHz。我们使用Vivado v2021.1综合实现加速器，系统运行频率为200 MHz，计算阵列的运行频率为400 MHz。

4.2 精度评估

很多量化研究工作[18,19]已证明量化后的CNN模型在提高性能的同时几乎不会带来明显的精度损失，但视觉Transformer中存在大量与CNN不同的非线性与归一化操作，直接对其低比特量化可能会带来严重的精度下降。为了证明本文设计的有效性，使用ImageNet2012验证集对加入本文设计的典型视觉Transformer模型进行精度消融实验。非线性与归一化函数替换模式分别设置为：仅替换Softmax(OS)、仅替换GELU(OG)、仅替换层归一化(OL)和全部替换(ALL)。测试模型选用DeiT-S, DeiT-B, ViT-B/16, ViT-L/16, Swin-T和Swin-S，

替换后的模型与各基线模型的Top-1和Top-5准确率比较结果如表2所示。

与基线模型相比，DeiT-S/16和DeiT-B/16在各种替换模式下的精度几乎没有损失，甚至在很多种模式下精度略有提升。ViT-B/16, ViT-L/16, Swin-T和Swin-S的Top-1与Top-5准确率基本不受影响，在各种模式下精度损失均小于0.1%。

4.3 非线性与归一化加速单元性能评估

为了进一步评估非线性与归一化加速单元的计算性能，本文将所提出的加速单元部署在资源规模较小的Xilinx Kintex Ultrascale+ XCKU15P FPGA与现有的其他FPGA加速设计进行对比，结果如表3所示。

表 2　模型精度消融实验(%)

模型	模式	数据类型	Top-1 Accuracy	Top-5 Accuracy	Top-1 Diff	Top-5 Diff
DeiT-S/16	基线	FP32	79.834	94.950	/	/
	OS	FP16	79.814	94.968	–0.020	+0.018
	OG	FP16	79.812	94.952	–0.022	+0.002
	OL	FP16	79.816	94.948	–0.018	–0.002
	ALL	FP16	79.814	94.984	–0.020	+0.034
DeiT-B/16	基线	FP32	81.798	95.594	/	/
	OS	FP16	81.842	95.620	+0.044	+0.026
	OG	FP16	81.832	95.610	+0.034	+0.016
	OL	FP16	81.828	95.600	+0.030	+0.006
	ALL	FP16	81.824	95.634	+0.026	+0.040
ViT-B/16	基线	FP32	84.528	97.294	/	/
	OS	FP16	84.522	97.262	–0.006	–0.032
	OG	FP16	84.520	97.306	–0.008	+0.012
	OL	FP16	84.526	97.292	–0.002	–0.002
	ALL	FP16	84.524	97.262	–0.004	–0.032
ViT-L/16	基线	FP32	85.840	97.818	/	/
	OS	FP16	85.800	97.816	–0.040	–0.002
	OG	FP16	85.818	97.818	–0.022	0.000
	OL	FP16	85.820	97.818	–0.020	0.000
	ALL	FP16	85.784	97.81	–0.056	–0.008
Swin-T	基线	FP32	81.172	95.320	/	/
	OS	FP16	81.152	95.304	–0.020	–0.016
	OG	FP16	81.156	95.320	–0.016	0.000
	OL	FP16	81.164	95.322	–0.008	0.002
	ALL	FP16	81.148	95.300	–0.024	–0.02
Swin-S	基线	FP32	83.648	97.05	/	/
	OS	FP16	83.642	97.02	–0.006	–0.030
	OG	FP16	83.646	97.08	–0.002	0.030
	OL	FP16	83.638	97.04	–0.010	–0.010
	ALL	FP16	83.636	96.966	–0.012	–0.084

在加速支持灵活性方面，文献[20-22]仅支持Softmax函数，文献[23]可加速Softmax和GELU函数，而本文提出的非线性与归一化加速单元可以同时支持Softmax, GELU和层归一化函数。在吞吐量方面，与基于传统方法的大规模实现[20]相比，本文的非线性与归一化加速单元吞吐量提高了14.23倍。与面向嵌入式的小规模实现[21-23]相比，吞吐量分别提高了9.81倍、28.44倍和83.24倍。

由于各文献使用的FPGA器件、计算资源和运行频率不同，仅采用吞吐量作为对比指标可能不足以证明本文设计的优越性。因此本文将所有操作实现为LUT，使用各加速器在100 MHz下的面积效率(Throughput Per LUT, TPL)来全面地对比性能，TPL是指运行时平均每个LUT的吞吐量。

与基于传统方法的大规模实现[20]相比，本文的非线性与归一化加速单元TPL提高了3.64倍。与面向嵌入式的小规模实现[21-23]相比，本文的大规模实现在TPL方面上基本达到了同等水平，这证明了本文的加速设计具有良好的面积效率，可以高效地部署在FPGA上，以满足大型视觉Transformer的处理需求。

4.4 加速器性能评估

本文在所提出的加速器上实现了ResNet-50和ViT-B/16，并与两种神经网络的GPU实现以及现有的神经网络FPGA加速器进行对比。GPU计算平台采用Nvidia V100，基于Tensorflow框架实现了ResNet-50，基于Pytorch框架实现了ViT-B/16。计算阵列上的卷积和注意力机制中的矩阵乘法等线性操作的数据类型使用INT8。本文使用GOP表示模型总操作数，GOPS表示加速器平均每秒完成的操作数，计算阵列(N_i, N_o)配置为(32, 64)，单个时钟周期完成2 048个乘法操作，实验结果如表4所示。

部署在本文所提出的加速器上的CNN模型ResNet-50帧率FPS达到了76.22；视觉Transformer模型ViT-B/16的帧率FPS则为32.16。由于各文献使用的FPGA器件、资源规模、运行频率和部署的神经网络模型不同，仅仅使用FPS作为对比指标无法评估各加速器的真实性能。

为了进一步评估加速性能，本文使用能效比和计算效率两种指标来对比分析各种实现。能效比即为吞吐量除以功耗，计算效率则由加速器运行时的平均吞吐量除以加速器的理论峰值吞吐量得到。由于映射到FPGA加速器上的ResNet-50和ViT-B/16的计算量集中在计算阵列上，并且本文加速设计与对比文献的加速设计的计算阵列均使用DSP实现，因此可以根据计算阵列的DSP效率来衡量FPGA加

速器的计算效率。对于未使用DSP的GPU实现，计算效率使用GOPS除以GPU峰值吞吐量得到。DSP效率由式(9)得到，其中N_{dsp}表示DSP数量，每个DSP同时处理两个INT8并行操作，∂取2，freq表示运行频率。能效比和计算效率的对比结果分别如图9所示和图10所示。

$$\mathrm{DSP\_Efficiency}=\frac{\mathrm{GOPS}}{2\times\partial\times N_{\mathrm{dsp}}\times\mathrm{freq}}\times 100\% \quad (7)$$

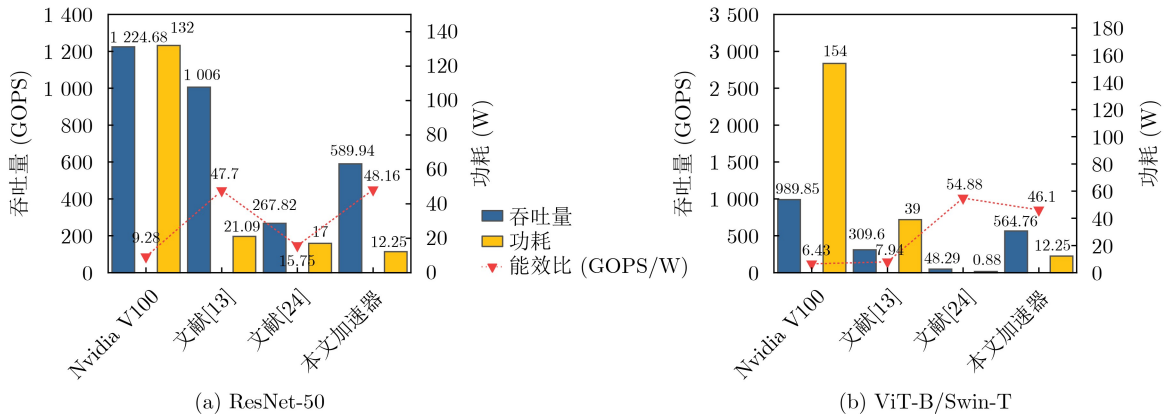
对于CNN模型ResNet-50，本文所提出的加速器的吞吐量达到了589.94GOPS，能效比为48.16 GOPS/W，计算效率为60.64%。与GPU实现相比，能效比提高了5.19倍；与文献[13]相比，能效比稍

表 3 与相关FPGA加速器的比较结果

类别		文献[20]	文献[21]	文献[22]	文献[23]	本文
灵活性	Softmax	√	√	√	√	√
	GELU	×	×	×	√	√
	层归一化	×	×	×	×	√
资源占用	LUT	17870	2564	2229	324	52639
	Slice Register	16400	2794	224	318	24403
	DSP	0	0	8	1	0
FPGA设备		ZYNQ-7000	–	Kintex-7 KC705	Zynq-7000 ZC706	Kintex XCKU15P
频率 (MHz)		150	436	154	410	200
数据精度(bit)		32	16	16	9+12	16+32
吞吐量 (Gbit/s)		2.4	3.49	1.2	0.41	34.13
100 MHz下TPL(Mbit/s)		0.089	0.312	0.349	0.309	0.324

表 4 与GPU实现和其他文献FPGA加速器的比较结果

	GPU		文献[13]	文献[24]	文献[9]	文献[10]	本文	
模型	ResNet-50	ViT-B/16	ResNet-50	ResNet-50	Swin-T	ViT-B/16	ResNet-50	ViT-B/16
计算平台	Nvidia V100		Xilinx KCU1500	Xilinx XCVU9P	Xilinx Alveo U50	Xilinx ZC7020	Xilinx XCVU37P	
制程(nm)	12		20	16	16	28	16	
频率(GHz)	1.46		200	125	300	150	200/400	
数据类型	FP32		INT8	INT8	FP16	INT8	INT8	INT8+FP16
输入尺寸	224×224		256×256	224×224	224×224	224×224	224×224	
GOP	7.74	17.56	11.76	7.74	–	17.56	7.74	17.56
DSP	–		2240	6005	2420	220	608	
计算延迟(ms)	6.32	17.74	11.69	28.90	–	363.64	13.12	31.09
帧率(FPS)	158.19	56.37	85.54	34.60	–	2.75	76.22	32.16



(a) ResNet-50 (b) ViT-B/Swin-T

图 9 能效比对比

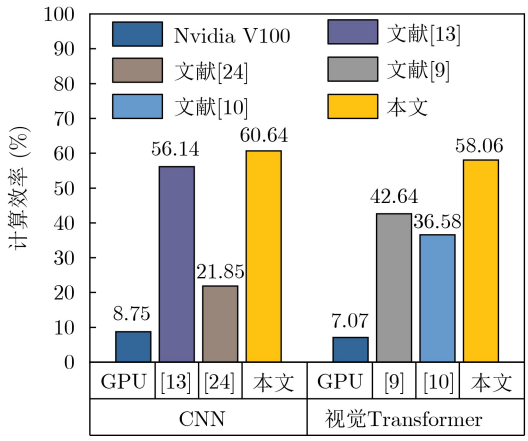


图 10 计算效率对比

有提高，计算效率提升了8.02%；与文献[24]相比，能效比提高了3.06倍，计算效率提升了177.53%。

对于视觉Transformer模型ViT-B/16，加速器的吞吐量达到了564.76 GOPS，能效比为46.1 GOPS/W，计算效率为58.06%。与GPU实现相比，本文所提出的加速器能效比提高了7.17倍；与文献[9]的Swin-T FPGA加速器相比，能效比提高了5.81倍，计算效率提升了36.16%；与文献[10]相比，虽然计算效率提升了58.72%，但能效比稍有下降，其原因在于文献[10]是面向边缘计算场景，FPGA器件资源规模很小，更容易控制加速器的功耗。上述实验结果可以证明本文实现的加速器具有良好的能效比和计算效率。

5　结束语

本文提出一种面向卷积神经网络和视觉Transformer的通用FPGA加速器架构。本文通过分析卷积和注意力机制的计算特征实现了面向FPGA的统一映射，对计算阵列规模进行设计空间探索最大化计算效率。通过分解视觉Transformer的各类非线性与归一化操作，所设计的非线性与归一化加速单元可最大限度地复用FPGA资源。实验结果表明，本文设计在提高吞吐量的同时仅造成很小的精度损失，所提出的加速器架构在实现ResNet-50和ViT-B/16推理中，帧率FPS分别为76.22和32.16，吞吐量达到了589.94 GOPS和564.76 GOPS，能效比远高于GPU实现和其他基于FPGA的大规模加速器设计，且计算效率优于对比加速器。后续工作将研究如何优化硬件资源占用，同时寻求支持更多种神经网络模型。本文研究可以拓展为计算机视觉领域专用计算架构，以完成计算机视觉任务高性能和高能效处理，具有较高的应用推广价值。

参 考 文 献

[1] SIMONYAN K and ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[C]. 3rd International Conference on Learning Representations, San Diego, USA, 2015.

[2] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, *et al*. Deep residual learning for image recognition[C]. 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016: 770–778. doi: 10.1109/CVPR.2016.90.

[3] SZEGEDY C, LIU Wei, JIA Yangqing, *et al*. Going deeper with convolutions[C]. 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, USA, 2015: 1–9. doi: 10.1109/CVPR.2015.7298594.

[4] VASWANI A, SHAZEER N, PARMAR N, *et al*. Attention is all you need[C]. The 31st International Conference on Neural Information Processing Systems, Long Beach, USA, 2017: 6000–6010.

[5] DOSOVITSKIY A, BEYER L, KOLESNIKOV A, *et al*. An image is worth 16x16 words: transformers for image recognition at scale[C]. 9th International Conference on Learning Representations, 2021.

[6] CARION N, MASSA F, SYNNAEVE G, *et al*. End-to-end object detection with transformers[C]. The 16th European Conference on Computer Vision, Glasgow, UK, 2020: 213–229. doi: 10.1007/978-3-030-58452-8_13.

[7] 陈莹, 匡澄. 基于CNN和TransFormer多尺度学习行人重识别方法[J]. 电子与信息学报, 2023, 45(6): 2256–2263. doi: 10.11999/JEIT220601.
CHEN Ying and KUANG Cheng. Pedestrian re-identification based on CNN and Transformer multi-scale learning[J]. *Journal of Electronics & Information Technology*, 2023, 45(6): 2256–2263. doi: 10.11999/JEIT220601.

[8] ZHAI Xiaohua, KOLESNIKOV A, HOULSBY N, *et al*. Scaling vision transformers[C]. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, USA, 2022: 1204–1213. doi: 10.1109/CVPR52688.2022.01179.

[9] WANG Teng, GONG Lei, WANG Chao, *et al*. ViA: A novel vision-transformer accelerator based on FPGA[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022, 41(11): 4088–4099. doi: 10.1109/TCAD.2022.3197489.

[10] NAG S, DATTA G, KUNDU S, *et al*. ViTA: A vision transformer inference accelerator for edge applications[C]. 2023 IEEE International Symposium on Circuits and Systems, Monterey, USA, 2023: 1–5. doi: 10.1109/ISCAS46773.2023.10181988.

[11] LI Zhengang, SUN Mengshu, LU A, *et al*. Auto-ViT-Acc: an FPGA-aware automatic acceleration framework for vision transformer with mixed-scheme quantization[C]. 2022 32nd

International Conference on Field-Programmable Logic and Applications, Belfast, UK, 2022: 109–116. doi: 10.1109/FPL57034.2022.00027.

[12] 吴瑞东, 刘冰, 付平, 等. 应用于极致边缘计算场景的卷积神经网络加速器架构设计[J]. 电子与信息学报, 2023, 45(6): 1933–1943. doi: 10.11999/JEIT220130.
WU Ruidong, LIU Bing, FU Ping, *et al*. Convolutional neural network accelerator architecture design for ultimate edge computing scenario[J]. *Journal of Electronics & Information Technology*, 2023, 45(6): 1933–1943. doi: 10.11999/JEIT220130.

[13] NGUYEN D T, JE H, NGUYEN T N, *et al*. ShortcutFusion: from tensorflow to FPGA-based accelerator with a reuse-aware memory allocation for shortcut data[J]. *IEEE Transactions on Circuits and Systems I:Regular Papers*, 2022, 69(6): 2477–2489. doi: 10.1109/TCSI.2022.3153288.

[14] LI Tianyang, ZHANG Fan, FAN Xitian, *et al*. Unified accelerator for attention and convolution in inference based on FPGA[C]. 2023 IEEE International Symposium on Circuits and Systems, Monterey, USA, 2023: 1–5. doi: 10.1109/ISCAS46773.2023.10182145.

[15] LOMONT C. Fast inverse square root[EB/OL]. http://lomont.org/papers/2003/InvSqrt.pdf, 2023.

[16] WU E, ZHANG Xiaoqian, BERMAN D, *et al*. A high-throughput reconfigurable processing array for neural networks[C]. 27th International Conference on Field Programmable Logic and Applications, Ghent, Belgium, 2017: 1–4. doi: 10.23919/FPL.2017.8056794.

[17] FU Yao, WU E, SIRASAO A, *et al*. Deep learning with INT8 optimization on Xilinx devices[EB/OL]. Xilinx. https://www.origin.xilinx.com/content/dam/xilinx/support/documents/white_papers/wp486-deep-learning-int8.pdf, 2017.

[18] ZHU Feng, GONG Ruihao, YU Fengwei, *et al*. Towards unified INT8 training for convolutional neural network[C]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, USA, 2020: 1966–1976. doi: 10.1109/CVPR42600.2020.00204.

[19] JACOB B, KLIGYS S, CHEN Bo, *et al*. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 2704–2713. doi: 10.1109/CVPR.2018.00286.

[20] SUN Qiwei, DI Zhixiong, LV Zhengyang, *et al*. A high speed SoftMax VLSI architecture based on basic-split[C]. 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology, Qingdao, China, 2018: 1–3. doi: 10.1109/ICSICT.2018.8565706.

[21] WANG Meiqi, LU Siyuan, ZHU Danyang, *et al*. A high-speed and low-complexity architecture for softmax function in deep learning[C]. 2018 IEEE Asia Pacific Conference on Circuits and Systems, Chengdu, China, 2018: 223–226. doi: 10.1109/APCCAS.2018.8605654.

[22] GAO Yue, LIU Weiqiang, and LOMBARDI F. Design and implementation of an approximate softmax layer for deep neural networks[C]. 2020 IEEE International Symposium on Circuits and Systems, Seville, Spain, 2020: 1–5. doi: 10.1109/ISCAS45731.2020.9180870.

[23] LI Yue, CAO Wei, ZHOU Xuegong, *et al*. A low-cost reconfigurable nonlinear core for embedded DNN applications[C]. 2020 International Conference on Field-Programmable Technology, Maui, USA, 2020: 35–38. doi: 10.1109/ICFPT51103.2020.00014.

[24] HADJIS S and OLUKOTUN K. TensorFlow to cloud FPGAs: Tradeoffs for accelerating deep neural networks[C]. 29th International Conference on Field Programmable Logic and Applications, Barcelona, Spain, 2019: 360–366. doi: 10.1109/FPL.2019.00064.

李天阳：男，博士生，研究方向为高性能计算、可重构计算技术.
张 帆：男，副研究员，研究方向为高性能计算、大数据、片上系统芯片设计.
王 松：男，工程师，研究方向为信息安全.
曹 伟：男，讲师，研究方向为可重构计算技术、片上系统芯片设计.
陈 立：男，博士生，研究方向为先进计算与类脑智能、计算机视觉.

责任编辑：余 蓉