计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200600089

基于 FPGA 的 CNN 图像识别加速与优化

齐延荣[1]　周夏冰[2]　李　斌[1]　周清雷[1]
1 郑州大学信息工程学院　郑州 450001
2 苏州大学计算机科学与技术学院　江苏 苏州 215006
(17319793885@163.com)

摘　要　目前,CNN 已广泛应用于许多应用场景中,包括图像分类、语音识别、视频分析、文档分析等。由于 CNN 计算密集,常以 GPU 进行加速,但 GPU 功耗高,不适用于 CNN 推理阶段。基于此,文中研究了基于 FPGA 的 CNN 图像识别加速与优化的应用方法,利用 Intel FPGA 提供的 OpenCL SDK,在 FPGA 板卡上设计并优化了 CNN 前向模型。首先,针对计算量问题,通过功能模块划分,充分发挥 FPGA 的高计算效能优势。其次,优化核心算法,提高运行速度;分析特征图处理操作,利用参数共享策略降低数据存储量;采用通道传输数据,减少访问片外存储次数。最后,对数据缓存、数据流、循环进行优化设计,缓解了 FPGA 片上的资源限制;通过量化参数降低 FPGA 内存资源占用量。实验结果表明,FPGA 具有较低的功耗,CPU 的功耗是其 2.1 倍,而 GPU 的功耗是其 6.5 倍;与近年来相关领域文献中提出的方法相比,所提方法具有较高的吞吐量和计算性能。

关键词:CNN;FPGA;图像识别;OpenCL;模块划分;数据流优化
中图法分类号　TP391

FPGA-based CNN Image Recognition Acceleration and Optimization

QI Yan-rong[1],ZHOU Xia-bing[2],LI Bin[1] and ZHOU Qing-lei[1]
1 School of Information Engineering,Zhengzhou University,Zhengzhou 450001,China
2 School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China

Abstract　Currently,CNN has been widely used in many application scenarios,including image classification,speech recognition,video analysis,document analysis,etc. Because CNN is computationally intensive,it is often accelerated with GPUs. However,GPU has a high power consumption and is not suitable for CNN inference stage. Based on this,this paper studies the application method of FPGA-based CNN image recognition acceleration and optimization. The OpenCL SDK provided by Intel FPGA is used to design and optimize the CNN forward model on the FPGA board. First of all,for the calculation problem,through the division of functional modules,the advantages of FPGA's high computing efficiency are fully utilized. Secondly,this paper optimizes the core algorithm to improve the running speed,analyzes the feature map processing operations,uses the parameter sharing strategy to reduce the amount of data storage,uses the pipeline to transfer data,and reduce the number of accesses to off-chip storage. Finally,it optimizes the design of data cache,data flow and loop to alleviate the on-chip resource constraints of FPGA,quantizes the parameters and reduce the amount of FPGA memory resources occupied. Experimental results show that FPGA has lower power consumption,CPU power consumption is 2.1 times that of FPGA,and GPU power consumption is 6.5 times that of FPGA. Compared with the methods proposed in the literature of related fields in recent years,the proposed method has higher throughput and computational performance.

Keywords　CNN,FPGA,Image recognition,OpenCL,Module division,Data flow optimization

1　引言

近年来,在机器学习研究领域,卷积神经网络(Convolutional Neural Networks,CNN)相比传统算法有了很大的进步[1]。CNN 作为计算机视觉领域的常用技术之一,已被应用于图像识别、图像检测、图像分割等任务,但常用网络模型在计算量上达到了十亿量级,在参数量上达到了上百兆量级,需要消耗很大的算力[2]。为提高 CNN 的计算性能,在 CPU,GPU,ASIC,FPGA 硬件平台上设计了 CNN 加速器。然而,CPU 拥有较少计算单元和大量缓存单元,不适合于密集运

到稿日期:2020-06-15　返修日期:2020-08-13
基金项目:国家重点研发计划"公共安全风险防控与应急技术装配"重点专项(2018XXXXXXX01);国家自然科学基金(61702518)
This work was supported by the National Key R&D Program "Public Safety Risk Prevention and Control and Emergency Technology Assembly" Key Special Project (2018XXXXXXX01) and National Natural Science Foundation of China(61702518).
通信作者:李斌(iebinli@zzu.edu.cn)

算。GPU 拥有大量计算单元，适用于计算量大、并行度高的计算，其运行速度虽较快，但功耗很高。在 CNN 推理阶段，因数据量的限制，GPU 不能充分发挥其高带宽、多计算核心的特点。ASIC 只适用于一类固定算法，当算法发生改变时，原有的 ASIC 芯片不能够再次迭代使用。FPGA 作为一种高性能、低功耗的可编程芯片，可以使用硬件描述语言设计数字电路，以形成对应算法的加速电路结构。因此，FPGA 更适用于 CNN 算法的推理阶段。

与 GPU 相比，FPGA 具有低功耗、低延时等优点，适用于小批量流式应用。与 ASIC 相比，FPGA 可以通过配置重新改变硬件结构，适用于深度学习这种日新月异的领域。由于 FPGA 具有良好的性能、高并行性、高能效以及可重构等特点，学术界越来越多的研究人员开始关注 FPGA 加速器设计。文献[1]指出，以前用于 CNN 的 FPGA 架构未能利用器件的峰值操作，导致性能较低。为解决上述问题，本文引入了基于 OpenCL 的新架构，采用基于 FPGA 的 OpenCL 异构开发环境实现 CNN 加速。该架构具有以下优势：

(1)基于 OpenCL 的 FPGA 加速器具有高效的流水线内核结构，用于实现大规模 CNN。

(2)通过分析模型，为 FPGA 设备和 CNN 找到最佳架构配置，以获得最大吞吐量。

此外，FPGA 与 CPU 之间拥有超高速总线接口，与 GPU 相比可以缩短延时，这对于将算法应用于实际推理环节具有更好的性价比，因此基于 FPGA 的 CNN 图像识别加速与优化具有极大的实用价值。

综上，本文针对先前用于 CNN 的 FPGA 架构的低性能问题，实现了基于 FPGA 的 OpenCL 异构开发环境的 CNN 加速器优化方案。本文采用功能模块划分、多通道并行、参数量化等方法，以 AlexNet 模型为例验证了本方案的吞吐量和计算性能。结果表明，本方案取得了较好的效果。

2 相关工作

2.1 AI 图像识别

CNN 是一种深度学习的神经网络模型，通过前向传播计算输出值，通过反向传播调整连接权重值。稀疏连接和权值共享是其最重要的两大特点，每层神经元与其相邻层神经元之间不再是全连接，而是部分连接，即某个神经元的感知区域仅受上层的部分神经元的影响，这样的网络结构降低了模型的复杂度。图 1 为 CNN 基本结构图。CNN 通常包含卷积层、池化层和全连接层。首先，对输入图像进行预处理，将处理好的图像输入神经网络模型；其次，对输入图像执行多次卷积(conv)操作和池化(pooling)操作，得到越来越复杂的特征图，之后进入全连接层；最后，计算输入图像的卷积值，通过 Softmax 函数计算概率值，再进行归一化操作并排序，将概率最大的节点作为预测目标，并输出分类结果。

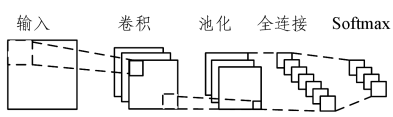


图 1 CNN 基本结构

Fig. 1 Basic structure of CNN

图像识别技术是利用物体之间的共性实现分类，把具有相同共性的图像归为同一类，如按颜色、形状、大小、类别等分类，分类标准不同得到的分类结果自然不同。图 2 为利用深度学习实现图像识别的流程图，其工作原理为：先对输入图像进行预处理操作，再通过特征提取抽取出能够反映图像本质的特征，最后将其输入训练好的网络模型中，获得识别结果。

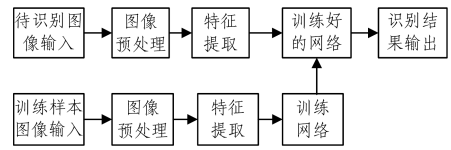


图 2 AI 图像识别流程图

Fig. 2 Flowchart of AI image recognition

2.2 FPGA 进展

FPGA 是一种可自定义编程的硬件电路结构，它提供的并行运算的计算模式契合 CNN 的计算特点，其可重编程的特点也适合于神经网络多变的网络结构。FPGA 具有三大优势。1)更大的并行度：主要通过并发和流水两种技术实现。2)可定制：在资源允许范围内，用户可实现自己的逻辑电路。3)可重构：FPGA 内部逻辑可根据需求改变，减少开发成本。使用 FPGA 复用资源比使用多个固定的 ASIC 模块可为服务器省下更多的空间。

FPGA 用于图像处理的关键点在于：能进行实时流水线运算，达到最高的实时性。因此，对实时性要求非常高的应用领域进行图像处理基本上都采用 FPGA。FPGA 提供的流水结构可以与 CNN 算法很好地匹配，很多研究者利用 OpenCL 针对 CNN 做了较为详尽的工作。Aydonat 等[3]提出一种新的深度学习加速器(Deep Learning Accelerator，DLA)，DLA 以 AlexNet 模型为例，在 Arria10 设备上的实现性能高达 1.38TFOPS[1]。Qiu 等[4]在嵌入式 FPGA 平台上，对图像网络分类中的存储空间和带宽问题进行了深入研究，在设计 CNN 加速器时主要关注两方面——计算引擎和优化内存系统，并提出了一种动态精度数据量化流程，在保证可比精度的前提下，减少内存占用和带宽需求。Wang 等[5]针对 CNN 设计了一种基于 FPGA 的具有可扩展性和灵活性的深度学习加速器 DLAU，使用 tile 技术将输入节点数据划分成更小的集合，并通过分时运算逻辑重复计算。在最先进的 Xilinx FPGA 板卡上的实验结果表明，与 Intel Core2 处理器相比，DLAU 加速器能够实现高达 36.1 倍的加速，功耗为 234mW。Wang 等[6-7]提出一种采用流水线内核的高效硬件架构，并使用不同硬件参数测试 AlexNet 和 VGG 网络模型，以进行设计空间探索，其设计方案在性能密度、资源利用率方面取得了显著改善。Kamel 等[8]在基于 FPGA 的 CNN 加速器中，研究了直接映射(Direct Hardware Map，DHM)的可行性，并表明目前的嵌入式 FPGAs 提供了足够的硬件资源来支撑这种方法。Wei 等[9]提出了一种基于 FPGAs 的高吞吐量 CNN 脉动阵列架构，通过精确的建模技术和有效的设计空间探索策略解决了 PE 映射、形状选择、数据重用策略的挑战。他们在 AlexNet 和 VGG-16 模型上进行了评估，该架构在 Arria 10 器件上可实现高达 1.17TFLOPS 的计算性能。

以上工作虽然都在 CNN 模型架构优化方面做出了很大贡献，但是，目前对于模型复杂程度高、模型参数量大的图像识别的移植优化仍较少。针对这些问题，本文利用 CNN 图像识别的特性，深入分析各网络层的特点，设计了一种基于 FPGA 的高效的 CNN 图像识别加速器。该加速器适用于多个网络模型。本文以 AlexNet 模型为例，量化权重参数，降低了存储规模，并将整个模型移植到 Xilinx FPGA 板卡上，验证了所设计的加速器的性能。

3 方案实现

本文采用软硬件相结合的方式，对采集到的图像进行分类处理，所采用的技术路线如图 3 所示。首先，在 PC 端环境下，通过 CPU 搭建 CNN 模型架构，对其进行训练、测试，以得到较好的权重值和偏置值，并将它们提取出来。同时在 FPGA 端搭建 CNN 硬件平台，将在 PC 端提取的权重和偏置运用到 FPGA 中，实现在硬件平台上对图像的分类[10]，取得了较好的加速效果。

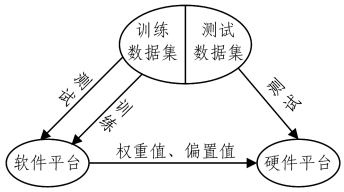


图 3 技术路线

Fig. 3 Technical route

3.1 CNN 加速器的基本架构

目前神经网络的规模越来越大，层数越来越深，由于 FPGA 的存储资源有限，设计基于 FPGA 的 CNN 加速器必须借助外部存储芯片。该设计将外部 DDR 存储芯片与 FPGA 相结合，设计架构如图 4 所示。

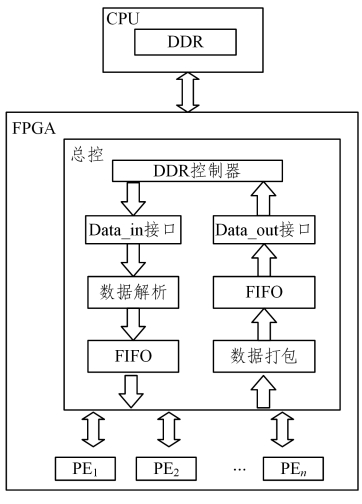


图 4 CNN 加速器的基本架构

Fig. 4 Basic architecture of CNN accelerator

总控模块包括 DDR 控制器和用于存储中间结果的 FIFO，负责管理输入特征图、网络权重、偏置和各种操作产生的中间结果。其中，DDR 控制器负责协调 FPGA 与外部 DDR 存储芯片之间的通信。在突发情况下，它作为 FPGA 与 DDR 之间的数据传输机制，将 DDR 中的数据传输到 FPGA 或者将 FPGA 中的数据传输到 DDR。PE 负责对输入特征值和权重进行计算，得到最终结果。

主机程序首先从硬盘或者网络摄像头加载图像数据，并对图像数据和网络权重进行预处理，将它们转化为 8 位定点数，然后发送到 FPGA 加速器的运算单元以执行 CNN 正向计算，最后将 CNN 正向计算所得结果通过 FPGA 发回给主机，并将结果显示在主机屏幕上。

3.2 软硬件划分

3.2.1 功能模块划分

CNN 应用的数据处理过程主要由 3 部分构成：数据预处理、利用网络进行分类或识别、输出识别结果。其中，图像预处理和识别结果输出部分与应用的前后端联系紧密，对计算能力的需求较低；而数据分类和识别操作的计算量大，并且任务内的数据吞吐量相对较高，因此该部分通过可重构硬件资源来构造专用的硬件加速器，以充分发挥 FPGA 的高计算效能优势。

目标系统的功能模块划分[11]如图 5 所示。为保证经过划分后，系统软件部分可依据应用进行再编程，并保障硬件加速器的可重构性，CNN 的硬件加速器通过高性能片上总线与处理器进行连接，加速器的设计采用数据驱动形式，将输入与输出设置为硬件平台所支持的总线接口形式。

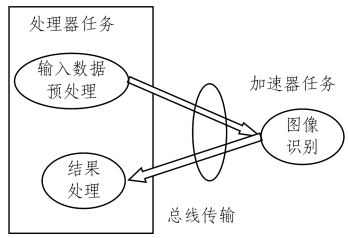


图 5 系统分割

Fig. 5 System partition

3.2.2 核心算法分析

用于图像分类的卷积神经网络由一个或多个卷积层、池化层以及全连接层组成。卷积层的核心部分是三维乘法累加运算，其定义如式(1)所示：

$$D_o(f_o,y,x)=\sum_{f_i=0}^{C_l}\sum_{k_y=0}^{K}\sum_{k_x=0}^{K}W_l(f_o,f_i,k_y,k_x)*D_i(f_i,y+k_y,x+k_x) \tag{1}$$

其中，$D_i(f_i,y,x)$和 $D_o(f_o,y,x)$分别表示输入特征图 f_i 和输出特征图 f_o 中位置(x,y)的神经元。$W_l(f_o,f_i,y,x)$表示第 l 层与 f_i 卷积的相应权重。卷积核的大小为 $K\times K$，输入特征映射总数为 C_l。若使用 HLS 友好的 1-D 卷积结构来实现式(1)，则该结构将 3-D 卷积展平为式(2)：

$$D_o(f_o)=\sum_{f_i=0}^{C_l}W_l(f_o,f_i)*D_i(f_i) \tag{2}$$

式(1)和式(2)可以使卷积循环在内核代码中避免嵌套循环，编译器生成的具有延迟缓冲区的乘法-加法器树构成有效卷积流水线。当选择适当的缓冲区深度时，OpenCL 编译器可以有效地流水线化所提出的结构，初始间隔仅为一个时钟周期。每个卷积流水线构成计算单元(Calculation Unit，CU)，并且内核由多个 CU 组成以执行并行卷积。

池化层主要包含两类：最大池化和平均池化，这两者均对

数据进行下采样操作。由于最大池化进行了特征选择，能够选出分类辨别度更好的模型。因此，本设计中池化部分选择最大池化，其操作如式(3)所示，其中，$p\times p$ 为卷积核大小。池化操作不仅能有效降低下层数据处理量，而且还能避免特征信息丢失。

$$f_{i,j}^{out}=\max_{p\times p}(f_{m,n}^{in},f_{m,n+1}^{in},\cdots,f_{m,n+p-1}^{in},f_{m+1,n}^{in},f_{m+1,n+1}^{in},\cdots,f_{m+1,n+p-1}^{in},\cdots,f_{m+p-1,n}^{in},f_{m+p-1,n+1}^{in},\cdots,f_{m+p-1,n+p-1}^{in}) \tag{3}$$

理想情况下，所有层的乘累加操作同时进行，CNN 模型的最大吞吐量(GOP/s)由式(4)计算得到。其中，*parallel-factor* 表示每层并行的乘累加操作数(OP/s)的最大值，*clock frequency* 为芯片的最大时钟频率。因各层的吞吐量存在差异，故整个网络模型的吞吐量相比理论计算的吞吐量有所下降。

$$MaxThroughput=\sum_{ID=0}^{allIDs}parallel\,factor*clock\,frequency \tag{4}$$

3.3 FPGA 设计与优化

3.3.1 FPGA 整体架构

FPGA 架构由多个通道组成，每个通道包含多个处理单元，多通道并行执行。以其中一个通道为例，所设计的 FPGA 算法加速器如图 6 所示。

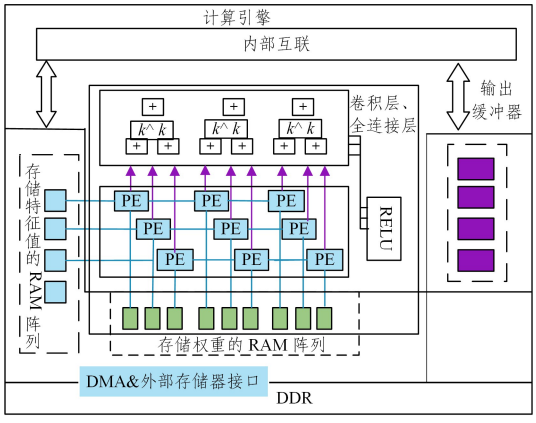


图 6 FPGA 算法加速器

Fig. 6 FPGA algorithm accelerator

该加速器的工作原理如下。输入图像通过 AXI 总线从外部存储器 DDR 输入到 FPGA 片上缓存，并将参与计算的特征存储到 RAM 阵列中。模型权重通过 AXI 总线输入到存储权重的 RAM 阵列中。鉴于模型参数量大，片上本地缓存仅存储正在参与计算的权值，当前计算完成后，再更新卷积层的特征值和权值。输入特征值和权重通过乘法-加法器树实现卷积操作，卷积后得到的结果经过 ReLU，Pooling 等操作写入片上本地内存中。缓存的特征图作为下一卷积层的特征输入，同卷积核再次进行卷积操作，直到得到最终的输出结果。多次卷积操作得到的输出特征值通过 AXI 总线将数据传输到 CPU 上，最后通过执行 Softmax 函数得到图像的最终分类结果。

3.3.2 核心运算优化

卷积层：计算一个卷积像素点需要大量乘加操作，导致卷积计算需消耗大量时间。为此，本文利用了数据矢量化、并行计算单元和 Multipler-Adder Tree 结构。卷积计算操作[12-14]如图 7 所示，卷积原理如下。

(1)在第三维上对输入特征图进行分组，每组 i 个；(2)在第四维度上对卷积核(权重)进行分组，每组 j 个，i，j 可根据实际实验环境设置。本设计中，i 设置为 16，j 设置为 16。矢量化的输入特征和权重由多个通道流式传输，i 个输入特征图卷积窗口同时进行乘法运算，将所得结果相加后存入累加器(accumulator，ACC)。虚线框之间表示输出特征图的并行操作。使用 i 控制输入数据吞吐量，使用 j 控制多通道并行计算以提高计算速度。在流水线填满后，且流水线足够充分的情况下，该计算每周期可以完成 $i\times j$ 个乘累加计算。通道内卷积计算完成后，需要对卷积后的特征图进行激活操作，激活的目的是保证其在一定范围内减少误差。

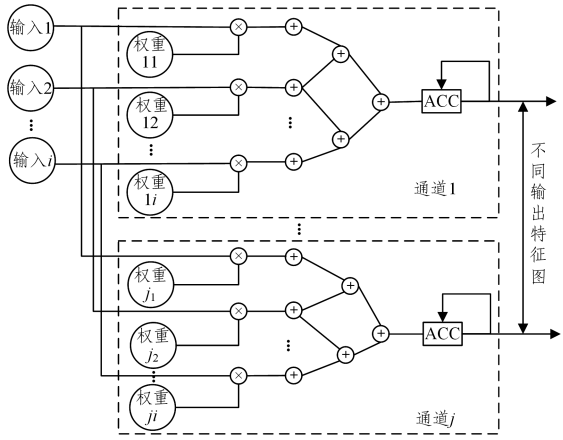


图 7 卷积操作示意图

Fig. 7 Schematic diagram of convolution operation

池化层：为池化内核提出了一种基于行缓冲区的硬件结构。以 3×3 池化核为例，其内部逻辑如图 8 所示，池化计算原理如下。

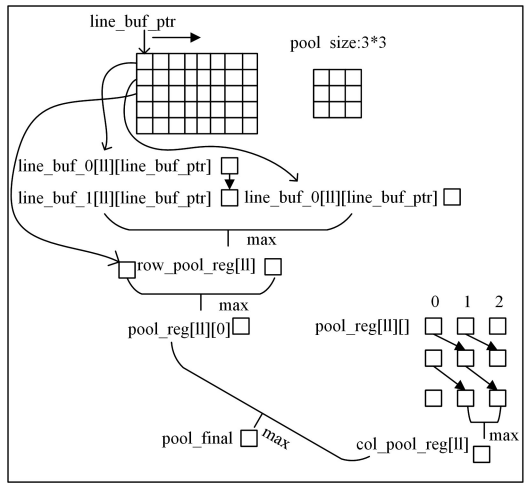


图 8 池化层运算过程

Fig. 8 Operation process of pooling layer

(1)内核先从通道中逐行读取特征图的数据，然后将读取的数据保存在线性缓冲区中。本设计中使用 2 个线性缓冲区(*line_buf_0* 和 *line_buf_1*)。

(2)借助两个行缓冲区求出列最大值，并将列最大值存储

在池化寄存器 *pool_reg* 中，然后通过池化寄存器移位计算出行最大值，即一个池化窗口最大值（*pool_final*）。

（3）用行指针控制行读取，当行缓冲区填满后，将其发送到池化操作的下一阶段（池窗口纵向移动），继续进行池化操作，直到对特征图执行完所有的池化操作。

该池化操作通过采用行列同时进行的方式，节省了计算时间，提升了池化操作速度。

局部响应归一化（Local Response Normalization，LRN）：在 AlexNet 模型中首次提出，是深度学习模型训练时提高准确度的一种方法。LRN 在激活函数后进行集中处理，且仅出现在 conv1 层和 conv2 层。采用分段线性逼近方案实现 LRN 的核心指数功能，通过查找表的方式来获取相关系数。图 9 为局部响应归一化内核设计的伪代码，其中，K 为 LRN 窗口尺寸。

```
计算前K/2个特征的sum_of_squares
对于每个input_feature i:
 对于特征i中的每个神经元j:
  对N_LRN个神经元并行执行以下操作:
   计算sum_of_squares[j]+=input_feature[i+K/2][j]
   计算output_feature[i][j]=input_feature[i][j]*pwlf(a_K*sum_of_squares[j])
   更新sum_of_squares[j]-=input_feature[i-K/2][j]
```

图 9　归一化内核设计伪代码

Fig. 9　Pseudocode of normalized kernel design

读内存：用于从全局内存中获取数据。

写内存：用于将数据写入全局内存中。

3.3.3　特征图处理

图 10 为特征图处理图，其中，***Map_matrix*** 为输入特征图的映射矩阵，C 为相邻通道数目，K 表示卷积核的尺寸大小，Q 为输入特征映射图数目，M_{out} 为输出特征映射图数目，N_{out} 为输出特征图宽度。对 ***Map_matrix*** 转置，使得卷积窗口内的像素地址是连续的，相邻通道中具有相同位置的卷积窗口像素地址也是连续的，故可以将 $C\times K^2$ 的像素合并到一个事务中。图 10 中，取 $K=2$，$C=2$，一个虚线框中的像素由一个 $WI(y,x)$ 处理，其中 $x=0,1,\cdots,M_{out}-1$，$y=0,1,\cdots,Q/C-1$。

对于输入数据，越界元素用零填充并写入片上缓冲区。对于输出数据，在每个工作项回写其对应的元素之前，它将检查位置是否超出界限，因此越界元素不会被写回。

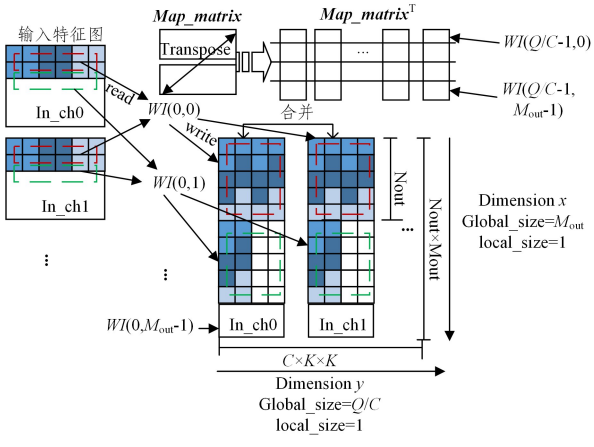


图 10　特征图处理

Fig. 10　Feature map processing

3.3.4　存储设计

本文通过为输入数据、权重、偏置、卷积、池化等设计独立的数据来获取函数，优化了加速器的核心数据流，同时使用了数据流优化指令。该指令不仅可以并行执行数据获取操作，而且还可以并行执行卷积计算单元和缓存单元数据的获取，从而提高并行度。

根据算法中的每个步骤计算并行度和计算数据之间是否相互独立，对核心算法进行内核划分。CNN 计算过程中，仅内存读内核、内存写内核可以访问全局内存，这些内核通过通道实现数据传输。通道采用 FIFO，用于内核之间数据的直接传递，图 11 为通道示意图。其中，RAM 存储了特征值和权重。采用通道传输降低了传输时延，提高了传输效率。

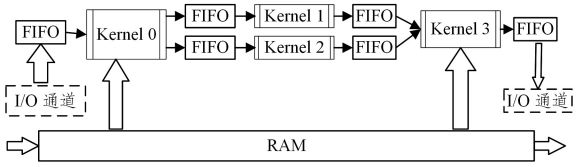


图 11　通道实现框图

Fig. 11　Channel realization block diagram

引入通道后，内核 1、内核 2、内核 3 均无须通过访问全局内存获取数据，而是直接从 FIFO 缓冲区获取数据。在内核执行时，通过移位寄存器读写数据，通过编程提高内核间的并行性，下一内核计算所需的数据在前一内核执行时顺序传递到通道中。

3.4　数据流优化

3.4.1　数据缓存

为提高 FPGA 计算吞吐量，需通过总线将片外内存数据传输到 FPGA 内部片上内存缓存，然后在设备上执行并行内核程序。如果内核程序每次执行都需要访问片外内存中的数据，则会在数据传输上浪费大量时间，从而使性能下降。为避免内核每次都从片外内存读写数据，尽量将数据存储在 FPGA 片上内存。通过这种方式，内核会优先使用片上缓存数据，从而避免一些无谓的数据传输。

由于片上缓存空间有限，本文利用“乒-乓”缓冲区读写片外、片上内存数据，图 12 为“乒-乓”缓存操作示意图。“乒-乓”缓存操作的核心思想是利用数据计算的时间来掩盖数据传输的时间。当缓冲区 1 处理数据时，缓冲区 2 传输数据；当缓冲区 2 处理数据时，缓冲区 1 传输数据。这种方式以额外硬件资源为代价，很好地掩盖了数据传输带来的损耗，提高了本方案的整体执行性能。

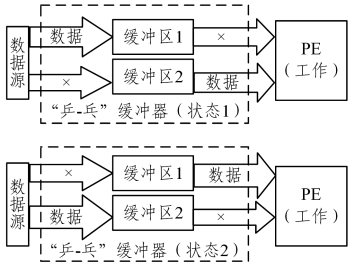


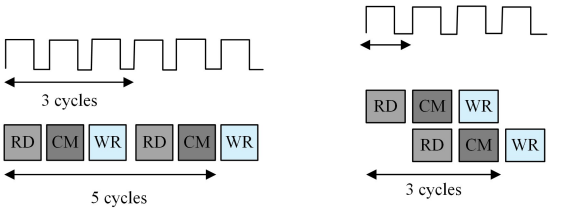
图 12　“乒-乓”缓冲

Fig. 12　“ping-pang” buffer

3.4.2 循环优化

卷积层的实现具有多层循环性，为进一步提高程序的并行执行性能，在主机程序上使用循环展开优化(unroll)指令，该指令用来指定循环展开的内核程序并告诉编译器需循环展开多少次(或由编译器决定需要循环展开多少次)。循环展开利用了卷积核间的并行性，充分利用计算资源，并行执行计算操作。

访问内存数据大致分为 3 个流程：读取数据、计算数据、存储数据。采用流水线方式能够缩短整体执行时间，假设每步操作需要 1 个时钟周期。图 13 分别展示了未使用循环展开优化和使用了循环展开优化指令的单流水模式。由图 13 可知，带循环的函数在未使用循环展开优化指令的情况下，在下次执行时延迟了 5 个时钟周期，而使用循环展开优化指令的循环函数只延迟了 1 个时钟周期。在 FPGA 中运用循环展开优化指令，可以将 for 循环展开，以提高程序在循环迭代时的并行性。



(a)未使用 unroll 指令的单流水线模式 (b)使用 unroll 指令的单流水模式

图 13 循环模式

Fig. 13 Cycle mode

3.4.3 全局内存优化

本设计采用了数据通道存储，使用通道将数据从一个内核传输到 FPGA 内部的另一内核，避免了计算过程中内核对全局内存数据的多次存取，从而极大地提高了数据并行性，降低了数据吞吐量和整个系统的延迟。图 14 对比了是否采用通道进行内核间数据传输。可以看出，使用通道极大地减少了内核访问全局内存的次数，提高了数据传输效率。

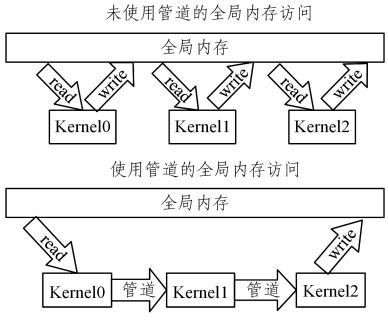


图 14 全局访存优化

Fig. 14 Global memory access optimization

本方案中，内存读内核通过 DDR 控制器从 DDR 中获取数据，通过通道将数据传输到其他内核。内核操作执行完成后，将结果数据通过通道传输到内存写内核，内存写内核再通过 DDR 控制器将数据传输到 DDR 中。整个执行过程中，仅内存读内核和内存写内核访问了外部存储器，降低了访存次数，提高了数据传输率。

3.5 量化设计

CNN 模型涉及的操作数一般默认为 32 位或 64 位浮点数，如果直接存储会占用较多 FPGA 内存资源。与浮点运算相比，在 FPGA 上实现定点运算效率更高。Int8 量化能够减小模型尺寸、减少存储空间、减少内存耗用。式(5)为由浮点到定点的量化公式，式(6)为定点到浮点的反量化公式。

$$Q=\frac{R}{S}+Z \tag{5}$$

$$R=(Q-Z)*S \tag{6}$$

其中，R 表示真实浮点值，Q 表示量化后的定点值，S 为定点量化后能表示的最小刻度，Z 表示浮点值 0 所对应的量化定点值。S 和 Z 的求值公式如下：

$$S=\frac{R_{\max}-R_{\min}}{Q_{\max}-Q_{\min}} \tag{7}$$

$$Z=Q_{\max}-\frac{R_{\max}}{S} \tag{8}$$

其中，$R_{\max}$ 和 $R_{\min}$ 表示最大、最小浮点值，$Q_{\max}$ 和 $Q_{\min}$ 表示最大、最小定点值。

以 AlexNet 模型为例，统计已训练好的模型权重，各层权重值分布区间存在差异，以层为单位进行量化。以第一层权重为例，图 15 为 AlexNet 模型第一层权重值分布图。首先，统计权重值范围，确定该层权重最大浮点值和最小浮点值；然后，采用 Int8 量化模型，根据式(5)将该层所有权重值映射为定点值，定点量化区间为[-128,127]。采用 Int8 量化，AlexNet 模型前向推理的准确率仅降低了 3%左右。

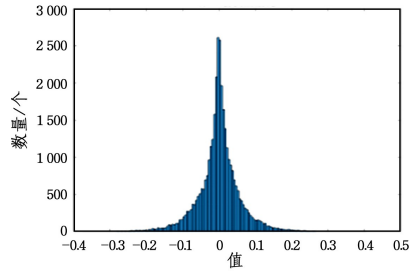


图 15 AlexNet 模型的权重参数分布

Fig. 15 Distribution of weight parameters of AlexNet model

4 实验及结果分析

4.1 实验环境搭建

为验证本设计方法的有效性，对文中所描述的加速方案进行上板测试。选用的 FPGA 开发工具是 Xilinx 公司的 SDx IDE 软件，选用的开发板为华为 FX600，硬件开发环境为 Vivado 2017.4。其中，高性能 FX600 开发板的 FPGA 芯片为 Virtex UltraScale+ VU9P FPGA，具有 4 个通道的 DDR4-2400 SDRAM，扩展部分重配置流程以实现较高的结构资源可用性，该芯片可用于具有 PCIe Gen3 x16 连接性的 PCI Express 的 Xilinx DMA 子系统。

4.2 测试数据集与网络模型

本文实验采用 ILSVRC2012 数据集，AlexNet 模型。其中，ILSVRC2012 数据集包含 1 281 167 张训练图片、50 000 张验证图像和 10 万张测试图片。AlexNet 网络模型包含 8 层，

其中 5 层卷积层(表示为 C)、3 层全连接层(表示为 F)。每个卷积层中都有激活函数(ReLU)和局部响应归一化(LRN)操作，在 C1，C2，C5 中还有池化操作，具体的网络参数如表 1 所列。该模型输入图片尺寸为 227×227×3，全连接层用于分类判别，即利用卷积层从训练集图像提取的特征来判断所识别测试集图像中的特征区域属于哪一类别。

表 1 AlexNet 网络参数

Table 1 AlexNet network parameters

层	输入	卷积核	数量	输出
C1	227×227×3	11×11×3	96	27×27×96
C2	27×27×96	5×5×96	256	13×13×256
C3	13×13×256	3×3×256	384	13×13×384
C4	13×13×384	3×3×384	384	13×13×384
C5	13×13×384	3×3×384	256	6×6×256

在 PC 端完成基于 AlexNet 模型的卷积神经网络训练过程后，保存训练权重值。为增加内存空间利用率，将训练得到的权重值进行定点优化后再移植到 FPGA 中，以供前馈卷积神经网络使用。

4.3 各模块资源占用

本设计经过综合布局布线后，各模块的资源占用情况如表 2 所列，其中卷积操作、读内存操作占用了较多资源。因 CNN 是乘法密集型的，将消耗的 DSP 数量作为评估硬件资源利用率的主要因素。表 3 列出了资源利用率情况。从表 3 可以看出，该板卡资源利用率均比较低，系统内仍有足够资源可以使用，可进一步优化程序以达到提升资源利用率的目的。

表 2 各模块资源占用情况

Table 2 Resources occupied by each module

module	FF	LUT	DSP	BRAM
convolution	37874	48929	48	4
lrn	11967	11262	21	8
pooling	3500	7757	0	32
memRead	23168	53354	22	112
memWrite	4750	7316	17	2

表 3 资源消耗情况

Table 3 Resource consumption

资源种类	$N_{已用}$/个	$N_{可用}$/个	利用率/%
FF	148698	2215298	6.71
LUT	118072	1059293	11.15
DSP	108	6836	1.60
BRAM	101	1926	5.24

4.4 性能分析

首先，在图片加载方面，将 FPGA 与 CPU，GPU 平台进行对比，结果如表 4 所列。可以看出，CPU 加载一张图片需要 67.84 ms(耗时长)，GPU 功耗较高，FPGA 功耗低且能效比高。综合分析，FPGA 更适用于 CNN 推理阶段。文献[9]所使用的 FPGA 平台 AWSF1 与本设计使用的 FPGA 平台 FX600 具有相同的架构，表 5 列出了具有相同架构的 FPGA 平台性能对比的详细信息。从表 5 可以看出，32 位浮点精度不具有优势，本方案采用的 8 位定点数具有较低的延迟(识别速度快)和较高的吞吐量。

表 4 单张图片处理时间的对比

Table 4 Comparison of processing time of single picture

平台	CPU	GPU	FPGA
型号	UHD Graphics 620	GeForce GTX 108	FX600
识别速度/ms	67.84	1.23	3.48
功耗/W	75	225	34.9
能效比/(img/s/W)	0.197	3.61	8.23

表 5 具有相同架构的 FPGA 平台性能对比

Table 5 Performance comparison of FPGA platforms with the same architecture

	频率/MHz	CNN 模型	精度	延迟/ms	吞吐量/GOPS
文献[9]	231.85	VGG	fixed 8-16 bit	26.85	576.28
	221.65	VGG	float 32 bit	54.12	285.9
	239.62	AlexNet	float 32 bit	4.05	356.99
本方案	260.48	AlexNet	fixed 8 bit	3.48	415.47

其次，在吞吐量和性能功耗方面将本文方案与其他方案进行了对比，结果如表 6 所列。由表 6 可以发现，吞吐量与 FPGA 中的 DSP 数目呈正相关趋势。本设计实现所用 FPGA 器件中包含较多的 DSP，在吞吐量指标上占有优势。既然吞吐量与 FPGA 中的 DSP 数目呈正相关趋势，那么对不同 FPGA 之间同一目标的实现，可以利用 DSP 效率(单位 DSP 所贡献的吞吐量)来衡量设计方案的有效性。由表 6 可知，在吞吐量和 DSP 效率指标上本文方案优于其他设计方案，即本方案具有较高的性能。本文通过性能功耗比来衡量实现方案对能量的利用率，可以看出本方案在性能功耗比这个指标上优于其他的实现方案，即单位能量中本文方案可以贡献较多计算量。

表 6 实验数据与其他方案的对比

Table 6 Comparison of experimental data with other schemes

设计	平台	时钟频率/MHz	功耗/W	吞吐量/GOPS	DSP 数目/个	DSP 效率/(GOPS/个)	性能功耗比/(GOPS/W)
文献[15]	SX240t	120	14.00	16.00	1056	0.015	1.14
文献[16]	XC7Z045	150	8.00	23.18	900	0.026	2.90
文献[17]	GSD8	120	—	117.80	3926	0.030	-
本文方案	FX600	260.48	34.9	415.47	6836	0.061	11.90

最后，以 AlexNet 模型为例，实现了所提 CNN 模型算法，数据精度采用 8 位整数量化，该模型在不同平台上的计算性能对比如表 7 所列。

表 7 不同平台实现 AlexNet 的性能对比

Table 7 Comparison of performance of different platforms to achieve AlexNet

相关工作	文献[3]	文献[9]	文献[18]	文献[19]	本文方案
FPGA 平台	GX1150	AWSF1	ZCU102	V7xc7v2000	FX600
数据精度	32 bit float	32 bit float	16 bit fixed	32 bit float	8 bit int
频率/MHz	303	230	200	189	260.48
计算性能/(TFlop/s)	1.38	1.88	1.28	1.24	3.56

由于优化 CNN 的工作使用不同的并行策略，采用不同

的FPGA平台很难对它们进行直接比较。为了实现较为公平的比较，表7中使用了“计算性能”这个指标来进行比较，计算性能(TFlops/s)表示每秒万亿次浮点运算，通常用于具有大量浮点运算的CNN前向推理过程中，以评估能效。从表7中可以看出，本文设计方案具有较高的计算性能。

结束语 本文提出了一种基于OpenCL的FPGA加速器，该加速器具有高效的流水线内核，可扩展性强，适应于多种网络模型。通过对FPGA工作原理、特点和结构的深入了解，挖掘OpenCL优势，利用OpenCL技术对网络模型结构进行简化，综合利用软硬件划分、循环优化、流水线优化、数据流优化及数据缓存等方法，优化了基于卷积神经网络的图像识别的硬件实现。与CPU相比，基于OpenCL的FPGA加速器具有较快的加载速度；与其他方案相比，本文方案具有较高的吞吐量和计算性能。

该方案资源利用率较低，下一步可从提高资源利用率方面进一步优化程序。权重采用Int8量化，在识别准确率损失较小的情况下，可考虑能否进一步量化，以压缩模型大小，减小存储空间。

参 考 文 献

[1] ZHOU F Y, JIN L F, DONG J. A review of convolutional neural network research[J]. Journal of Computer Science, 2017, 40(6): 1229-1251.

[2] WU Y X, LIANG K, LIU Y, et al. Progress and Trend of Deep Learning FPGA Accelerator[J]. Chinese Journal of Computers, 2019, 42(11): 2461-2480.

[3] AYDONAT U, O'CONNELL S, CAPALIJA D, et al. An opencl deep learning accelerator on arria 10[J]. arXiv: 1701.03534v1, 2017.

[4] QIU J, WANG J, YAO S, et al. Going deeper with embedded FPGA platform for convolutional neural network[C]// Acm/Sigda International Symposium on Field-programmable Gate Arrays. 2016: 26-35.

[5] WANG C, GONG L, YU Q, et al. DLAU: A Scalable Deep Learning Accelerator Unit on FPGA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, 36(3): 513-517.

[6] WANG D, XU K, JIANG D. PipeCNN: An OpenCL-based open-source FPGA accelerator for convolution neural networks[C]// 2017 International Conference on Field Programmable Technology(ICFPT). Melbourne, VIC, 2017: 279-282.

[7] WANG D, AN J J, XU K. PipeCNN: An OpenCL-Based FPGA Accelerator for Large-Scale Convolution Neuron Networks[J]. arXiv: 1611.02450v1, 2016.

[8] ABDELOUAHAB K, PELCAT M, SÉROT J, et al. Tactics to Directly Map CNN Graphs on Embedded FPGAs[J]. IEEE Embedded Systems Letters, 2017, 9(4): 113-116.

[9] WEI X C. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs[C]// 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). Austin, TX, 2017: 1-6.

[10] WANG Y, ZHOU H Y, FENG H, et al. Network traffic classification method based on deep convolutional neural network[J]. Journal of Communications, 2018, 39(1): 14-23.

[11] LU Y, CHEN Y, LI T, et al. Construction method of embedded FPGA convolutional neural network for edge computing[J]. Computer Research and Development, 2018, 55(3): 551-562.

[12] ZHOU Y M, JIANG J F. An FPGA-based accelerator implementation for deep convolutional neural networks[C]// 2015 4th International Conference on Computer Science and Network Technology(ICCSNT). Harbin, 2015: 829-832.

[13] ZHANG C, LI P, SUN J, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks[C]// Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays. 2015: 161-170.

[14] JIAN Q, ZHANG P Y, WANG X J. A configurable CNN co-accelerator FPGA implementation method[J]. Acta Electronica Sinica, 2019, 47(7): 1525-1531.

[15] CHAKRADHAR S, SANKARADAS M, JAKKULA V, et al. A dynamically configurable coprocessor for convolutional neural networks[C]// Proc. ACM SIGARCH Comput. 2010: 247-257.

[16] GOKHALE V, JIN J, DUNDAR A, et al. A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks[C]// 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops. Columbus, OH, 2014: 696-701.

[17] SUDA N. Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks[C]// Proc. ACM/SIGDA Int. Symp. Field Program. 2016: 16-25.

[18] LU L, LIANG Y, XIAO Q, et al. Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs[C]// 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). Napa, CA, 2017: 101-108.

[19] HAN X, ZHOU D, WANG S, et al. CNN-MERP: An FPGA-based memory-efficient reconfigurable processor for forward and backward propagation of convolutional neural networks[C]// 2016 IEEE 34th International Conference on Computer Design (ICCD). Scottsdale, AZ, 2016: 320-327.

QI Yan-rong, born in 1995, postgraduate. Her main research interests include image processing and high-performance computing.

LI Bin, born in 1986, Ph. D, associate professor. His main research interests include information security and high performance computing.