深度学习 FPGA 加速器的进展与趋势

吴艳霞[1)] 梁 楷[1),2)] 刘 颖[2)] 崔慧敏[2)]

[1)](哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

[2)](中国科学院计算技术研究所计算机体系结构国家重点实验 北京 100190)

摘 要 随着大数据时代的来临,深度学习技术在从海量数据中提取有价值信息方面发挥着重要作用,已被广泛应用于计算机视觉、语音识别及自然语言处理等领域.本文从深度学习算法的特点和发展趋势出发,分析 FPGA 加速深度学习的优势以及技术挑战;其次,本文从 SoC FPGA 和标准 FPGA 两个方面介绍了 CPU-FPGA 平台,主要对比分析了两种模型在 CPU 和 FPGA 之间数据交互上的区别;接下来,在介绍 FPGA 加速深度学习算法开发环境的基础上,重点从硬件结构、设计思路和优化策略这三个方面详细介绍了采用 FPGA 加速卷积神经网络的设计方案;最后展望了 FPGA 加速深度学习算法相关研究工作的发展.

关键词 深度学习;神经网络;CPU-FPGA;硬件加速;FPGA

中图法分类号 TP302 **DOI 号** 10.11897/SP.J.1016.2019.02461

The Progress and Trends of FPGA-Based Accelerators in Deep Learning

WU Yan-Xia[1)] LIANG Kai[1),2)] LIU Ying[2)] CUI Hui-Min[2)]

[1)](*College of Computer Science and Technology, Harbin Engineering University, Harbin* 150001)

[2)](*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

Abstract With the coming of big data era, the technology of deep learning plays a critical role in extracting the meaningful information from the massive data. Also, it has been widely applied in some domains, such as computer vision, speech recognition and natural language processing. As far as we all know, deep learning algorithms have a large number of parameters and relevant matrix multiplication or multiply-and-add operations with the simple computing model. At the same time, researches and industries need higher and higher accuracy, the complex models which have more and more weights won the image classification and object detection contest. To speed up the inference and training of deep learning becomes much more important. This paper mainly reviews one of the approaches, accelerating deep learning on FPGAs. Firstly, this paper introduces the deep learning algorithms and relative characteristics especially the convolutional neural network and recurrent neural network and why the FPGA approach can fit this problem, what people can be benefit compared with CPU only or GPU and what people should do to enable it. After that, this paper analyzes the challenges of accelerating deep learning on FPGAs. Additionally, the CPU-FPGA is the most welcome architecture at present, but there're different methods to set up a usable platform. And for CPU-FPGA platforms, data communication is one of the important factors for affecting the performance of acceleration. Based on these, this paper introduces different methods from the aspects of SoC FPGA and standard FPGA, and comparatively analyzes the

收稿日期:2017-08-03;录用日期:2018-12-05.本课题得到国家"八六三"高技术研究发展计划项目(2016YFB1000402)、黑龙江省自然科学基金(F2018008)、哈尔滨市杰出青年人才基金(2017RAYXJ016)资助.**吴艳霞**,博士,副教授,中国计算机学会(CCF)会员,主要研究领域为计算机体系结构、编译技术. E-mail: wuyanxia@hrbeu.edu.cn. **梁 楷**,硕士,主要研究领域为计算机体系结构、可重构计算. **刘 颖**,博士,助理研究员,主要研究领域为异构并行编程、编译系统及相关工具. **崔慧敏**(通信作者),博士,副研究员,主要研究领域为并行计算、并行编译、并行编程. E-mail: cuihm@ict.ac.cn.

differences of the data communication between CPU and FPGA of the two methods. For different engineers and developers, the development environments and tools of accelerating deep learning on FPGAs are presented in this paper from the aspects of high-level language, such as C and OpenCL, and hardware description language, such as Verilog. It is not hard to use techniques which require many complex low-level hardware control operations for FPGA implementations to improve performance by using hardware description. But that always requires a working knowledge of digital design and circuits. In contrary to hardware description language, using high-level language allows engineers and developers to have more freedom to explore algorithm instead of designing hardware architectures and it is suitable for researchers who need quickly iterate through design cycles and software developers who have no knowledge of digital design. And on the basis of it, the FPGA-based accelerator for deep learning algorithms is reviewed from the hardware architecture, design methods and optimization strategy in detail. Three typic hardware architectures are presented for convolutional neural network, and hardware architectures of two typic models for recurrent neural network are introduced. Design methods are reviewed from the goal of accelerator, the way to build model of algorithms and how to find the best solution. As for optimization strategy, it is presented from the processer and memory communication, which are critical factors for improving performance of Accelerator. Finally, this paper prospects the research on FPGA-accelerated deep learning algorithms from the aspects of the development environments, process communications between CPU and FPGA, better compression model methods and cloud application with FPGA.

Keywords deep learning; neural network; CPU-FPGA; hardware accelerator; FPGA

1 引 言

随着大数据时代的来临，人们在过去几年里创造的数据呈爆发式增长[1]. 面对如此海量的数据，相比于以前手工提取数据特征的方式，学者更加倾向于采用可以有效避免手工提取繁杂性和提高特征完备性的深度学习技术. 目前，深度学习技术已在众多领域中发挥了重要作用，如图像识别[2-5]、语音识别[6-8]、自动翻译[9-10]及智能管理[11]. 但这其中很多应用场景对硬件设备的性能、功耗等都有严格的要求，所以，越来越多的研究机构对深度学习硬件加速器进行了广泛且深入的研究. 本文在对比分析三种主流的深度学习硬件加速器解决方案的优缺点基础上，详细介绍了采用 FPGA 加速深度学习的硬件结构、设计思路和优化策略，并展望了 FPGA 加速深度学习算法的未来前景.

1.1 深度学习算法模型简介

数据特征[12]的自动提取是通过特征学习实现的. 特征学习是能让机器根据输入的原始数据自动发现分类或者检测所需要的特征的一系列方法. 深度学习[12]就是一种拥有多重表现层的特征学习的方法，它利用一些简单的但是非线性的模型，将原始数据逐步转换成为更高层次的、更抽象的表达. 目前，被广泛使用的深度学习基础模型有两种：一种是卷积神经网络；另一种是递归神经网络[13]. 本文以这两种神经网络为例，对深度学习算法进行说明.

卷积神经网络是一个多层神经网络，而且大部分卷积神经网络“深度”较深，即有很多层. 其中有 3 个主要的层：卷积层，池化层和全连接层.

(1) 卷积层. 该层主要是利用卷积核，通过卷积计算将输入数据转换为输出数据. 以图片的卷积处理为例，其中输入数据为原始图片或者特征图，输出数据为特征图，体现了图片某个方面的特征，卷积核是指特征图计算时图片共享的权值矩阵，其规模较小，一般为 5×5、3×3 或 1×1 等. 特征图中每个像素是将原始图片中与卷积核同样大小的区域与卷积核进行对应点的乘法，然后对所有乘积进行加法生成的，而特征图是通过滑动卷积核，将卷积核与原始图片中不同区域不断进行卷积操作生成的，其中滑动距离的大小记为步长. 图 1 展示了一个简单的二维卷积操作. 由于原始图片以及特征图规模较大，且

在进行卷积操作时一般涉及多个卷积核，因此，卷积层涉及到了大量的卷积操作，即乘加计算，这使得该层对计算能力的要求较高.并且，特征图中的每个像素点之间以及不同的特征图之间都是独立的，即不同特征图以及特征图中不同像素点的计算都是可并行的，因此卷积层中的计算通常是可并行的.

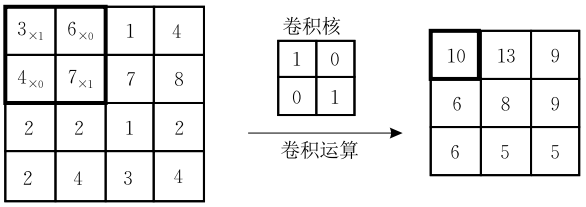

图 1　卷积操作示例

(2)池化层.该层主要作用是将特征图分为多个区域，然后通过对各个区域求像素平均值或者最大值等来减小特征图的尺寸.图 2 简单展示了这一过程.

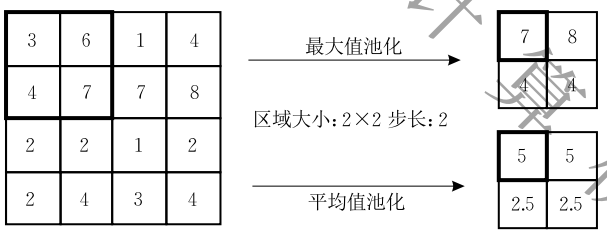

图 2　池化层示例

3)全连接层.全连接层主要是将卷积层等提取的特征表示映射到对对应的样本空间.由于这些特征表示规模一般较大，并且全连接层中基本每个输出数据的计算都与所有的输入数据相关.因此在计算时，涉及到的数据规模较大.图 3 简单展示了全连接层.

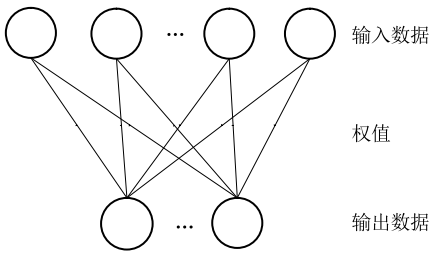

图 3　全连接层示例

卷积神经网络有两个阶段：训练阶段和推理阶段.下面以图片识别为例对这两个阶段进行说明.

在训练阶段，大量已分好类的图片数据作为输入数据输入到卷积神经网络中，此时卷积神经网络模型的参数，如卷积核的权值均为初始值；然后卷积神经网络会逐层处理数据，即下层的输入数据是上层的输出数据，直到输出图片的分类结果；接着根据分类结果和实际图片分类结果的差别进行反向逐层调整模型的参数；重复训练该神经网络，直到训练出合适的模型参数.无论是正向处理数据的过程还是反向调整模型参数的过程，卷积神经网络层与层之间的数据都是相互依赖的，一般下一层数据处理的开始需要等到上层数据处理的结束.

推理阶段是在训练阶段后，利用额外的图片数据测试该卷积神经网络的图片分类能力.其过程主要是卷积神经网络逐层处理数据直到得出图片分类结果，层与层之间同样存在数据依赖.

递归神经网络与卷积神经网络类似，也是一个多层的神经网络，同样有训练和推理阶段，数据的处理也是逐层进行，并且数据的计算主要是乘加计算.递归神经网络与卷积神经网络最大的区别是在递归神经网络中含有循环的连接，如神经网络中隐藏层神经元之间循环连接，这使得在计算输出数据时不仅要考虑当前的输入数据，还需要考虑先前历史输入数据对该输出数据的影响.

综上所述，深度学习对应的神经网络模型一般包含两个阶段：训练阶段和推理阶段、相比于推理阶段，训练阶段包含更多的数据处理，以及反向调整模型参数的过程.这些神经网络模型无论是正向处理数据还是反向调整参数都是逐层进行的，因此，神经网络层与层之间存在数据的依赖，通常很难将这些层作为独立的个体进行并行的处理.神经网络模型主要的层有三个：卷积层、池化层和全连接层.其中卷积层中包含大量可并行的乘加计算，对计算能力要求较高，全连接层涉及的数据较多，通常需要频繁地进行数据交互.

1.2　深度学习算法模型的发展趋势

Nurvitadhi 等人在其文献[14]中对深度神经网路的发展趋势进行了总结，其中主要的发展趋势有 3 个：

(1)虽然更深的网络结构能提高分类的准确率，但是这些规模较大的模型正变得难以处理.表 1 展示了近几年在 ImageNet 挑战赛中获取冠军的 4 个经典神经网络模型的相关信息.根据表 1 可以看出，从 2012 年到 2015 年，Top-5 的错误率从 15.3% 降低到了 3.57%，神经网络模型的层数也从 8 层增加到了 152 层.在 2012 年到 2014 年期间，随着神经网络模型层数的增加，模型参数和计算量也极速增加，模型大小从 240 MB 增加到了 500 MB，计算量从 7.2 亿操作提升到了 196 亿，而模型规模的增大和计算量的大幅度提升意味着对硬件计算能力、内存

带宽及数据存储的要求也更加苛刻.为了减小模型规模及让网络模型对硬件更加友好等,学者转向设计更加高效的深度神经网络,如 2014 年的 GoogleNet 利用提出的 Inception 模块等在有效提高神经网络模型层数的同时降低了模型规模的大小,最终模型规模仅为 AlexNet 的十分之一;2015 年的 ResNet 利用"shortcut connection"来跳过某些层,让更多的层能共享权值,使得该模型在网络模型深度上是 VGG-19 模型的 8 倍,但在规模上仅为 VGG-19 模型的一半左右.

表 1 4 个经典神经网络模型的相关信息

网络模型	Top-5 Error /%	层数	模型大小 /MB	乘加计算或操作/亿
AlexNet(ILSVRC'12)	15.30	8	240	7.2
VGG(ILSVRC'14)	7.30	19	500	196.0
GoogLeNet(ILSVRC'14)	6.70	22	24	15.5
ResNet(ILSVRC'15)	3.57	152	240	113.0

(2)通过降低数据精度来提高神经网络模型的效率.深度神经网络模型中的数据在默认情况下通常为 32 位或者 64 位浮点数,但文献[15-16]提出,在不损失准确率或者准确率损失较小的情况下,用位数较低的数据如 4～8 位的数据代替全精度浮点数是可行的.相比于浮点数,使用位数相对较少的数据类型能够有效减少访存的开销及提高计算的吞吐率等.在深度神经网络中使用比 32 位单精度浮点数更加紧凑的数据类型正成为一个新的趋势.在使用紧凑数据类型的深度神经网络模型中,典型的有二值神经网络(Binarized Neural Networks,BNNs)和三元神经网络[17-18](Ternary Neural Networks,TNNs),其中,文献[19-21]对于二值神经网络中的数据使用的是 1 位数据类型,数据值限制为+1 或者-1.该网络模型的分类准确率在小数据集中得到了验证[14];三元神经网络主要是将模型权值限制为 0、+1、-1 这三个值.最近的文献[18]表明,TNNs 分类准确率已经非常接近全精度的 ResNet-152 的分类准确率.

(3)利用稀疏性来提高神经网络模型的效率.在深度神经网络模型中,神经元值或者权值为 0 的计算是没有意义的,其不会影响模型最终的计算结果,因此,人们可以通过去掉网络模型中神经元值或权值为 0 的计算来减少模型的计算量,进而提升神经网络模型的效率.对于网络模型中神经元值为 0 的情况,近几年大部分神经网络模型的激活函数均为 ReLU(Rectified Linear Unit)函数,该函数会将值为负数的输出神经元赋值为 0,这意味着这些网络模型中存在着一定数量的 0 值神经元,根据文献[22]提供的数据,在目前流行的神经网络模型中,有 50%左右的神经元的值为 0.对于网络模型中权值为 0 的情况,文献[15]表明在不损失分类准确率或者准确率损失较小的情况下,将网络模型中一些不重要的权值赋值为 0 是可行的,并且该文献指出,对于 AlexNet 和 VGG-16 中的某些层,在不损失分类准确率的情况下,可赋值为 0 的权值占比超过 90%.除此之外,还有一些神经网络因为本身的特性包含了大量的 0 值,如二值神经网络模型和三元神经网络模型.

除了上述 3 种主要的趋势外,人们还会通过对深度神经网络模型进行数学变换来提高神经网络模型的执行效率,如将卷积操作、全连接操作转化为目前性能较为优异的矩阵乘法,快速傅里叶变换等;以及对深度学习算法进行深度压缩,通过减少模型数据量、计算量的方式来提高整体的执行效率等.

2 深度学习算法的硬件加速

深度学习算法是以数据处理为核心,其包含大量的计算操作,如 GoogleNet 网络模型包含 15.5 亿浮点操作,ResNet-152 网络模型包含了 113 亿浮点操作等.这对 CPU 来说并不友好,从硬件结构上看,CPU 中大部分晶体管都是用于构建高速缓冲存储器以及控制单元等,而负责逻辑运算的部分并不丰富.同时,在深度学习的应用领域中,有很多应用场景对性能、功耗等有一定的要求,通常需要一个高性能或者高能效的解决方案.而传统的 CPU 很难满足这些要求,因此,人们在应用深度学习技术时,通常会利用其它硬件对其进行加速.目前,主流的硬件加速器有 3 类:GPUs(Graphics Processing Units),ASICs(Application-Specific Integrated Circuits)和 FPGAs(Field-Programmable Gate Array).

2.1 GPU 加速的特点

与传统 CPU 结构不同,GPU 的内部结构中包含了大量的逻辑计算单元,其规模远远超过结构中控制单元和寄存器等存储单元的规模,GPU 架构示意图如图 4 所示.这种结构意味着 GPU 拥有较多的计算核心,在对数据进行算术运算或者逻辑运算方面有着很大的优势.

除了拥有较多的计算核心外,GPU 的另一个优势是它的内存结构.首先,在 GPU 的架构中有一些

存储单元，如寄存器等是供一些算术逻辑单元共享的，即 GPU 线程之间的通讯可以依赖这些共享的内存而不依赖全局内存，其中计算单元与共享内存之间的数据传输速度比与全局内存之间的要快. 其次，GPU 拥有相对高速且内存带宽相对较大的全局内存，如 GDDR5. 虽然传统的 CPU 架构中有高速缓冲存储器的存在，但是其容量较小，无法存储深度学习算法模型所需的数据，需要存储在内存中，然而由于架构的原因，CPU 的内存带宽并不如 GPU 的. 因此，GPU 更适合数据吞吐量相对较大的算法，如深度神经网络模型的训练阶段.

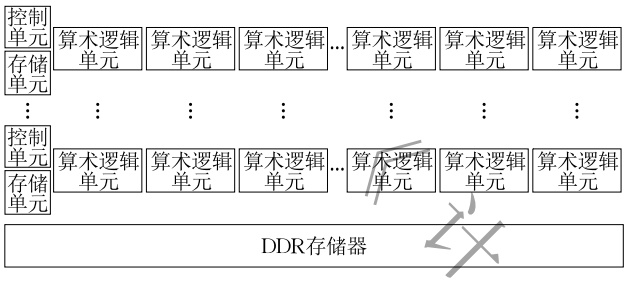

图 4　GPU 架构示意图

目前，一些深度学习框架，如 Caffe、Theano 以及 TensorFlow 等能较好地应用于 GPU 上. 并且 NVIDIA 公司也为 GPU 提供了较好的深度学习开发环境，如深度学习加速库 cuDNN 等，这不仅提高了 GPU 加速的性能，还很大程度上降低了编程的难度.

2.2　ASIC 加速的特点

与通过调整深度学习算法来适应 CPU 或者 GPU 硬件结构进而实现加速的方式不同，利用 ASIC 加速深度学习算法的主要方式通过定制专用的硬件加速算法，如针对卷积神经网络算法的加速设计[23-30]. 由于 ASIC 是针对某一个或者某一类算法进行硬件定制，所以利用 ASIC 加速深度学习算法通常能取得较高的性能，并且功耗较低. 但是这种加速方式也有一些要求和限制：ASIC 加速器的硬件设计与开发周期较长，成本较高，硬件生成后通常无法改变，灵活性不高.

2.3　FPGA 加速的特点

与 GPU 及 ASIC 固定的硬件结构不同，FPGA 具有可编程性，即开发者可以根据自己的需要通过可编程的链接将 FPGA 内部的逻辑块连接起来，实现相应的功能. 因此，FPGA 硬件加速器的设计也需要开发人员有一定的硬件专业知识，这对软件开发人员来说门槛较高. 不过，近几年，一些公司及研究机构不断丰富 FPGA 开发环境，目前已经允许开发人员利用高级编程语言如 C、C++或者 OpenCL 对 FPGA 进行开发，在很大程度上降低了 FPGA 开发的难度，缩短了 FPGA 开发周期，为研究者和开发者提供了便利.

2.4　FPGA 加速与 GPU 和 ASIC 加速的对比

2.4.1　FPGA 加速与 GPU 加速的对比

(1) FPGA 的加速设计是硬件结构适应算法，而 GPU 加速设计是算法适应硬件结构. 由于 GPU 硬件结构是固定的，在加速设计时通常需要调整深度学习算法来适应 GPU 的硬件结构、执行模型等，只有这样才能获取优异的加速性能. 由于 FPGA 具有可重构性和可编程性，因此，在加速设计时通常是根据算法设计硬件结构，这种加速设计方式可以更快速地去加速快速发展的深度学习算法.

(2) FPGA 通常有较好的能效比. 为了实现可重构性，FPGA 内部有很多基于查找表的基本单元，这些基本单元的计算能力在一般情况下并不如 CPU 和 GPU 中的算术逻辑单元，同时，FPGA 片上 DSP 资源通常是有限，其中 DSP 资源主要用于浮点计算，而 GPU 内部有大量的浮点计算单元，因此，FPGA 在浮点计算能力上并不如 GPU. 但是随着 FPGA 技术的快速发展，FPGA 的浮点计算能力也在快速的提升，文献[14]指出 Intel Stratix 10 的 32 位浮点计算吞吐量预计能达到 9.2 TFLOP/s，已经接近最新 Titan X Pascal GPU 所提供的峰值 32 位浮点计算吞吐量 11 TFLOP/s.

同时，与 GPU 相比，其功耗通常要比 GPU 低. 综合计算和功耗这两个方面，FPGA 通常有更加出色的能源效率[31-32]，即在通常情况下，FPGA 能在单位功耗下获取更高的性能. 如 Griffin Lacey 等人在文献[33]中指出，对于卷积神经网络(CNN)的推理阶段，微软团队利用 FPGA(Stratix V D5)实现了高性能的加速，其加速性能为每秒处理 134 张图片，功耗仅为 25 瓦，并且如果使用更高端的 FPGA(Arria 10 GX1150)，这个加速性能预计能达到每秒处理 233 张图片，而功耗基本不变；而对于高性能的 GPU 实现(Caffe+cuDNN)，其加速性能为每秒处理 500～824 张图片，功耗为 235 W. 这意味着 FPGA 的能源效率能达到 GPU 的 2～3 倍. 这对于某些深度学习应用，如超大型数据中心，资源有限的嵌入式应用来说尤为重要. 目前，一些国内外知名公司，如亚马逊、微软、腾讯及阿里巴巴等，也逐步尝试将 FPGA 部署到数据中心中.

(3) GPU 更适合深度学习算法的训练阶段，

FPGA 更适合深度学习算法的推理阶段.在深度学习算法的训练领域,其使用的多为单指令多数据流的计算.以 AlexNet 模型的训练阶段为例,在训练时通常一次输入多张图片,然后以相同的操作统一处理这些图片.这与 GPU 的架构非常契合,能充分发挥 GPU 高带宽、计算能力强的特点.而对于推理阶段,这部分的计算通常为单数据多指令流的特点.以 AlexNet 模型的推理阶段为例,在推理时输入一张图片,然后通过计算对图片进行分类.由于数据量的限制,在推理阶段 GPU 并不能充分发挥高带宽,多计算核心的特点.同时,在未来很多深度学习技术都用于分类、推断,特别是在移动终端上,这要求硬件加速器有低功耗、高性能、低延时等的特性,相比于 GPU,FPGA 拥有较高的能效比,因此,FPGA 更适合加速深度学习算法的推理阶段.

(4)FPGA 更适合新兴的深度神经网络.文献[14]提出了深度神经网络的趋势,其中包括了利用稀疏性和减小数据精度,其中稀疏性意味着神经网络中可并行的计算并不会像原有的那样整齐有规律,这对以算术逻辑单元为主而控制器相对简单的 GPU 来说并不友好,该文献对 GPU 计算稠密矩阵乘法和以跳过 0 值的方式计算稀疏矩阵乘法的性能进行了测试,实验结果表明,GPU 计算稀疏矩阵乘法的性能要低于计算稠密矩阵乘法的性能;对于减小数据精度,GPU 目前只能较好地支持 32 位浮点型和 8 位整型数据,并不能较好地支持自定义位数的数据类型.而无规律的并行计算和自定义位数的数据类型对 FPGA 来说并不是问题,有的甚至更适合 FPGA,如在 FPGA 上实现二值神经网络时可以用位运算代替原本的乘加计算.该文献对 FPGA 和 GPU 实现新趋势下的深度神经网络的性能进行了测试,实验结果表明,FPGA(Stratix 10)在稀疏深度神经网络、数据类型为 6 位整型的深度神经网络以及二值深度神经网络的矩阵乘法操作中的性能(TOP/sec)比 GPU(Titan X Pascal)提升了 10%、50%和 5.4 倍,在三元 ResNet 网络模型中的能效比 GPU 要高 2.3 倍.

2.4.2 FPGA 加速与 ASIC 加速的对比

(1)ASIC 加速在性能上通常优于 FPGA.与具有可重构性的 FPGA 不同,ASIC 是针对特定算法的专用电路,其硬件结构生成后便无法更改.由于 ASIC 在设计时不需要像 FPGA 那样考虑通用性及可重构性等,ASIC 无论在性能上还是在功耗上通常都会优于 FPGA,以近年来一些经典的应用于深度学习领域的 ASIC 设计为例:IBM 公司设计的 TrueNorth 芯片在加速典型复杂的递归神经网络时,功耗只有 65mW,对应的计算功耗是每瓦 460 亿次神经突触操作[34];寒武纪系列芯片中 DianNao 的平均性能与当时主流的 GPGPU 相当,但功耗和面积仅为主流 GPGPU 的百分之一量级[35],DaDianNao 单芯片性能超过了当时主流 GPU 的 21 倍,而能耗仅为当时主流 GPU 的 1/330[36];Google 公司研发的 TPU(Tensor Processing Unit)在神经网络应用中平均比当前的 GPU(Nvidia K80 GPU)或 CPU(Intel Haswell CPU)快 15~30 倍,能效比(TOPS/Watt)高出约 30~80 倍,此外,如果在 TPU 中采用 GPU 常用的 GDDR5 存储器能使性能 TPOS 指标再高 3 倍,并将能效比指标 TOPS/Watt 提高到 GPU 的 70 倍,CPU 的 200 倍[37].而 FPGA 的功耗通常为瓦这个量级,并且在加速深度学习方面,FPGA 的能效比通常为同代 GPU 的 2~3 倍.

(2)FPGA 加速具有更好的灵活性、更低的开发门槛以及更短的开发周期.虽然 ASIC 在性能和功耗上优于 FPGA,但是其在设计和制造时要经过很多的验证和物理设计,导致开发周期较长,并且,ASIC 的设计通常需要开发人员有硬件专业知识,开发门槛高.同时,ASIC 是针对某一类应用而设计的专用硬件且硬件结构在生成后无法改变,然而深度学习算法正处于快速发展的阶段,对于一些使用广泛但算法并不成熟的应用场景,想要设计一个高性能的通用 ASIC 来适应所有应用场景非常困难.FPGA 更适合加速目前处于快速发展阶段的深度学习算法.

综上所述,由于 FPGA 在加速深度学习方面与 GPU 和 ASIC 相比有着自己独特的优势,利用 FPGA 加速深度学习算法备受学者关注.

3 FPGA 加速深度学习算法

目前,利用 FPGA 加速深度学习同样面临着一些问题和挑战:

(1)有限的 FPGA 资源如何满足神经网络的计算和数据需求.随着深度学习算法和数据的规模不断增大,开发人员已经很难将深度学习算法和数据完全地映射到 FPGA 片上资源上.而且,不同的 FPGA 设计,最终的加速性能以及 FPGA 片上资源的使用会千差万别.因此,对于每一个 FPGA 开发人员来说,如何更高效地利用有限的 FPGA 片上资

源，避免资源的浪费以及如何利用更少的资源获取更高的性能是一个巨大的挑战.

(2) 硬件知识不足的编程人员如何能自如地设计和开发出高效的硬件加速器. 众所周知，高效率、高性能的硬件加速器的设计需要一定的专业硬件知识，这对大多数编程人员来说要求较高. 目前，虽然有很多高层次综合工具为 FPGA 提供了高级语言编程环境，允许编程人员在不具备硬件知识的情况下利用高级语言进行开发，但是想要使用这些工具生成一个高性能的硬件结构并不是一件容易的事情，通常需要编程人员不断地调整和优化代码才有可能达到预期的性能. 如何降低 FPGA 加速设计的门槛，让不具备硬件知识的编程人员能自如快速地开发出高效、高性能的硬件加速器是一个需要重点关注的问题.

基于这些问题和挑战，本小节首先简介常用于加速深度学习的CPU-FPGA 平台，随后说明 FPGA 加速深度学习的开发环境，最后详细地阐述对卷积神经网络和递归神经网络的 FPGA 加速设计方法.

3.1　CPU-FPGA 平台

在加速深度学习算法时通常会采用 CPU-FPGA 平台，即 FPGA 主要负责对计算能力要求较高的并行计算部分，如加速卷积神经网络中的卷积操作等，CPU 则负责系统的剩余工作，通过两者的协同工作，实现对深度学习算法的加速. 从 CPU 和 FPGA 耦合程度的角度分析，常用的 CPU-FPGA 平台有两种：SoC(System on Chip) FPGA 和标准 FPGA.

3.1.1　SoC FPGA

在 SoC FPGA 中 CPU 和 FPGA 是紧耦合的，即 CPU 和 FPGA 是封装在一个芯片中，通过总线连接，其中 CPU 通常分为两种：硬核和软核. 目前 FPGA 供应商如 Xilinx 和 Altera 提供的最新 SoC 开发板使用的 CPU 通常为 ARM 硬核. SoC FPGA 架构如图 5 所示，此架构的主要特点是集成度高、功耗低及通信带宽较高，通常用于嵌入式应用中.

对于利用 CPU-FPGA 平台加速深度学习算法，CPU 和 FPGA 之间的数据交互效率是影响加速性能的关键因素之一. 因此，本文以 Altera 公司的 Arria 10 SoC 为例对 SoC FPGA 中 CPU 和 FPGA 之间的数据交互进行说明. 图 6 展示了 Arria 10 SoC 中硬处理器系统内部及与 FPGA 逻辑之间的峰值带宽. 从图中可以看出 FPGA 逻辑和微处理器之间通讯的峰值带宽为 3.2 Gbps，并且 FPGA 可以

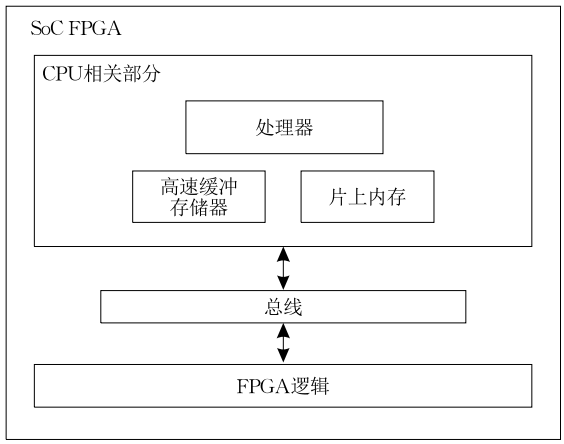

图 5　SoC FPGA 架构图

通过 F2H 桥接器连接高速缓冲存储器、片上内存等，进而实现较小数据量的数据快速交互. 同时，微处理器和 FPGA 逻辑共享一个 SDRAM(Synchronous Dynamic Random Access Memory)，并通过 L3 SDRAM 互联实现与 SDRAM 的数据快速交互，两者带宽峰值分别能达到 4.8 Gbps 和 4 Gbps，而 L3 SDRAM 互联和 SDRAM 控制器之间的峰值带宽高达 17 Gbps.

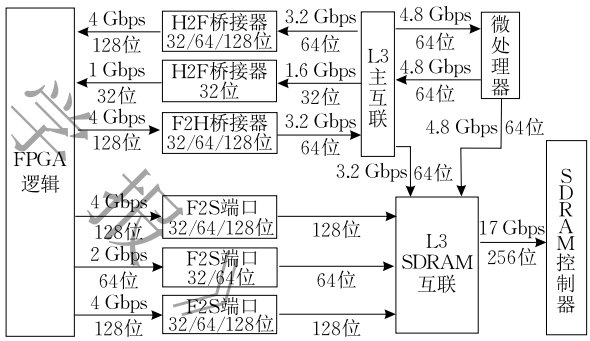

图 6　Arria 10 SoC 中 CPU 和 FPGA 之间的峰值带宽

3.1.2　标准 FPGA

与 SoC FPGA 不同，标准 FPGA 与 CPU 是松耦合结构，即 FPGA 和 CPU 不在一个芯片中，两者一般通过 PCIe(Peripheral Component Interconnect express)接口等进行连接. 由于标准 FPGA 的芯片中未嵌入微处理器，因此在同等芯片面积的情况下，标准 FPGA 相比于 SoC FPGA 中的 FPGA 逻辑部分拥有更丰富的硬件资源，能实现更为复杂的应用.

对于标准 FPGA 中 CPU 和 FPGA 之间的数据交互，目前常用的方式有两种：基于 PCIe 总线的数据交互和基于 QPI(Quick Path Interconnect)总线的数据交互.

(1) 基于 PCIe 总线的数据交互. 在该数据交互模式中，CPU 端和 FPGA 端都有各自的内存，其中

FPGA 端内存是指 FPGA 片外的存储器，如 DRAM (Dynamic Random Access Memory). CPU 与 FPGA 之间数据的交互是通过 PCIe 总线实现的. 图 7 为基于 PCIe 数据交互的结构.

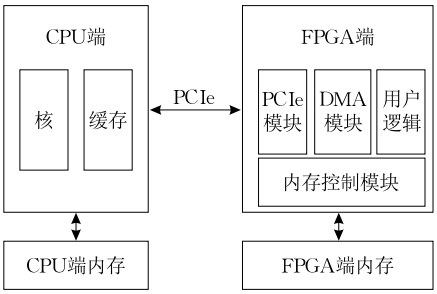

图 7 基于 PCIe 数据交互的结构

(2) 基于 QPI 总线的数据交互. 在该数据交互的模式中，CPU 端有独立的内存，而 FPGA 端没有独立的内存，CPU 端和 FPGA 端通过 QPI 总线进行数据交互. 在 FPGA 端，需要系统协议层模块为用户逻辑提供地址转换等功能. 图 8 表明了基于 QPI 数据交互的结构.

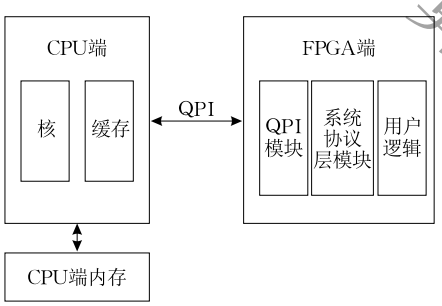

图 8 基于 QPI 数据交互的结构

对比这两种数据交互的模式. 在结构上，基于 PCIe 总线的数据交互结构，其 CPU 端与 FPGA 端数据的交互实质上是两个内存之间数据的交互. 而基于 QPI 总线的数据交互结构，其 CPU 端与 FPGA 端共享一个内存，FPGA 通过一个高效的数据通道来实现数据交互. 在性能方面，Choi 等人在文献[38]对这两种数据交互模式进行了对比分析，其研究结果指出 QPI 总线的带宽最大为 7.0 GB/s，PCIe-DMA 的带宽最大为 1.6 GB/s，PCIe-DMA 带宽较低的一个主要因素为 PCIe-DMA 数据传输不仅包含 PCIe 的传输，还包括 CPU 端缓存区的分配和内存拷贝. 在一般情况下，基于 QPI 总线的数据交互在带宽上是优于基于 PCIe 总线的. 同时，其研究对有效延时也进行了分析，研究结果指出 PCIe Gen3 x8 的有效延时高于基于 QPI 总线的内存的有效延时，两者相差最大能达到两个数量级. 因此，基于 QPI 总线的数据交互模式更加适合对延时较为敏感的应用，特别是当传输数据量小且传输次数较多的时候.

3.2 FPGA 开发环境

目前在 FPGA 上开发应用有两种主要的方式：一种是底层硬件设计，即开发人员利用硬件描述语言或者 IP 核(Intellectual Property core)对硬件结构进行设计；另一种方式是采用高层次综合工具，即开发人员只需要利用高级语言实现算法，而算法程序到 FPGA 硬件结构的映射由编译器自动完成. 下面将对这两种方式进行简要的说明.

(1) 采用高层次综合工具方式. 近几年，一些公司如 Altera、Xilinx 为 FPGA 提供了高级语言开发环境，并开发了相应的集成开发环境：Xilinx 公司的 SDAccel 以及 Altera 公司的面向 OpenCL 的软件开发套件等. 利用这些工具，开发设计人员只需要考虑如何采用 OpenCL 实现应用程序，而不需要考虑后续 FPGA 硬件结构映射的工作，这样在一定程度上降低了 FPGA 的开发门槛. 但是，采用这些工具对 FPGA 进行开发也面临着一些问题：这些工具支持的开发板通常是有限的，开发人员并不能任意地选择板卡；对于大多数开发人员来说，采用这些工具实现高性能的 FPGA 硬件结构并不是件容易的事情，通常需要不断地调整和修改高级语言的程序.

同时，不同的开发工具及开发板，在性能、资源使用等方面也是不一样的. Tapiador 等人在文献[39]中对利用 Xilinx 和 Altera 提供的基于 OpenCL 的开发工具及开发板加速卷积神经网络进行了研究，该研究结果表明，Xilinx 公司的工具能实现更快地综合，有更高的资源利用率等，而 Altera 公司的开发社区更加成熟，程序的执行时间更短.

(2) 采用底层硬件设计方式. 虽然 SDAccel 等开发工具提供了方便的 OpenCL 开发环境，但是这些工具支持的板卡是有限的，并且工具将 OpenCL 程序转换为 FPGA 可部署字节流文件的过程是封闭的，即开发人员很难将其它性能更优的特定硬件模块加入到工程中. 因此，也有部分学者选择利用硬件描述语言和 IP 核来编写底层硬件结构. Xilinx 和 Altera 等公司也为其提供了相应的开发环境，如 Xilinx 公司的 Vivado 工具、Vivado HLS 工具以及相应的 IP 核，如 PCIe 对应的 IP 核，内存管理对应的 IP 核. 其中 Vivado 工具的编程语言为硬件描述语言，其主要功能为对编写的程序进行综合、仿真、布局布线以及生成 FPGA 所需的字节流文件，虽然

Vivado HLS 也是高层次综合工具，但是其主要功能只是将 C/C++程序转换为 IP 核，接下来仍需要对硬件整体的结构进行设计，并不能像 SDAccel 工具那样提供完整的系统设计，因此，这种方式能按照需要灵活地设计硬件结构，但是需要相关的硬件专业知识.

目前，一些公司将 FPGA 和云计算进行了结合，在云端提供了统一的硬件平台和中间件，让用户在云端使用 FPGA 成为可能，如亚马逊公司提供的 FPGA 加速云服务、华为公司提供的 FPGA 加速云服务器等. 这些 FPGA 云服务在很大程度上降低了 FPGA 加速器的开发和部署成本，也使开发人员能够快速、便捷地使用 FPGA 进行开发. 但由于 FPGA 加速云服务处于刚起步的阶段，很多地方还需要进一步的完善.

在加速深度学习算法方面，FPGA 供应商也在不断完善 FPGA 加速深度学习算法的开发环境，为开发者提供了算法相关的框架和库，如 Xilinx 公司已面向终端和云端提供了框架 Caffe 和 mxnet. 这在一定程度上为开发人员开发深度学习相关应用提供了便利.

3.3　卷积神经网络的 FPGA 加速设计

由于卷积神经网络推理阶段的计算特点为单数据多指令流，且在实际应用中通常有低功耗、高性能、低延迟等的要求，因此比较适合 FPGA. 利用 FPGA 加速卷积神经网络性能的瓶颈主要体现在两个方面：计算和数据传输. 因此，在进行 FPGA 加速设计时，通常采用转换模型算法和优化硬件结构的方式，通过充分发挥卷积神经网络中的并行计算能力来提升计算效率；采用减少数据量、减少访存次数等方式来减少数据传输带来的开销.

基于此，本小节将从 FPGA 加速卷积神经网络推理阶段的硬件结构、设计思路以及优化策略这 3 个方面对 FPGA 的加速设计进行阐述.

3.3.1　FPGA 加速卷积神经网络的硬件结构

本小节将根据本文第 1.2 节说明的深度神经网络的发展趋势，总结 FPGA 加速卷积神经网络推理阶段的硬件结构.

(1) 卷积神经网络对应的基础硬件结构. 这里的基础硬件结构指的是在对深度神经网络算法模型未进行减小数据精度、剪枝等优化的情况下的 FPGA 加速硬件结构. 文献[40]针对计算吞吐和内存带宽不匹配的问题对卷积神经网络的卷积层设计了相应的 FPGA 硬件加速结构，具体结构如图 9 所示，FPGA 硬件结构主要有两个部分：计算部分和数据传输部分.

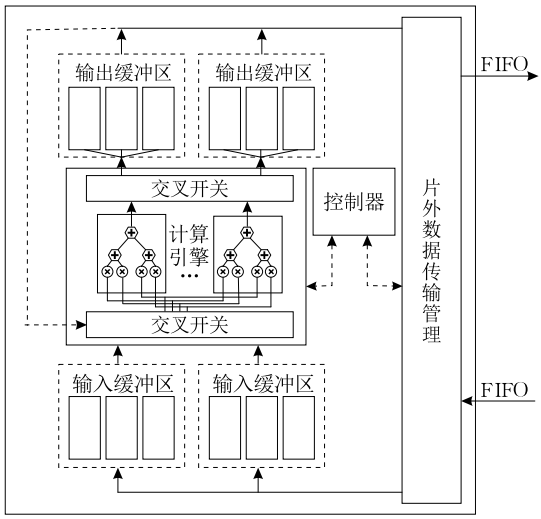

图 9　典型的基础硬件结构[40]

对于计算部分，其利用多个计算引擎并行执行，不同的计算引擎计算不同卷积核的卷积操作，计算引擎中的硬件结构为“树形”结构，即最下层为乘法器，用于并行执行卷积操作中的乘法计算，剩下的为多层加法器，用于并行计算卷积操作中的求和运算，这种硬件结构充分利用了卷积神经网络卷积层中可并行执行的计算：不同卷积核的卷积操作可并行执行、卷积操作中的乘法计算可并行执行以及卷积操作中的加法计算可部分并行执行等. 这样的硬件结构有效提高了计算的效率.

对于数据传输部分，其对输入和输出部分分别使用了两个缓冲区，利用“乒-乓”的数据传输机制有效地用计算时间掩盖了数据传输的时间，有效减少了程序整体的执行时间. 同时，由于每个计算引擎计算的是不同卷积核对同一输入数据的卷积操作，因此输入数据只需要从片外存储器中读取一次便可以在片上多次使用，这有效提高了本地数据的使用率，一定程度上减少了访问片外存储的次数. 这种硬件结构的设计在当时取得了优异的性能：在 100 MHz 的频率下实现了 61.62 GFLOPS 的峰值性能，并显著优于以前的设计方法[40].

虽然这种硬件结构取得了优异的性能，但是依然存在一些问题：

① 该结构主要针对的是卷积神经网络的卷积层，并未考虑全连接层，而全连接层中包含了大量的模型参数，会造成频繁的数据交互，大量数据传输带来的开销可能会成为性能提升的瓶颈.

② 计算引擎中的“树状”结构适合规则的卷积

操作，并不适合通过“剪枝”处理后的不规则卷积计算.因此，该结构不适合对进行稀疏性处理后的卷积神经网络.

③计算涉及的数据为浮点数，并不是定点数.使用定点数不仅能节省 FPGA 片上资源的使用，还能提升 FPGA 整体的执行性能.

(2)针对卷积神经网络稀疏性设计的 FPGA 硬件结构.卷积神经网络模型由于 ReLu 激活函数等原因通常包含一些零值，而这些零值参与的计算并不影响网络模型最终的计算结果，如果在计算时能合理地跳过这些零值，程序整体的执行性能将得到有效地提高.对于该种方式，FPGA 硬件结构设计的难点在于零值数据处理和不规则计算硬件结构的设计.文献[14]对此提出了一种硬件结构的设计，具体结构如图 10 所示.其中处理单元的硬件结构如图 11 所示.

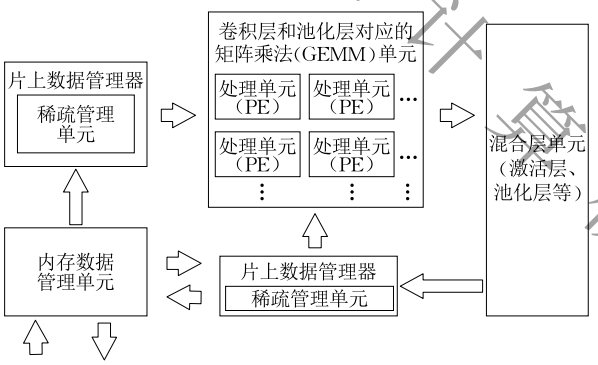

图 10 针对卷积神经网络稀疏性设计的硬件结构

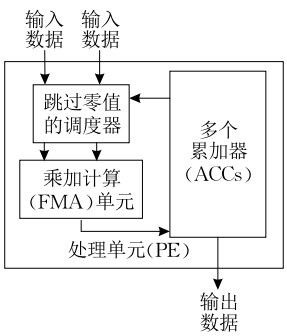

图 11 处理单元的硬件结构[14]

从图 11 可以看出，卷积层和全连接层的硬件结构是针对矩阵乘法设计的，即其将卷积层和全连接层中的计算转换为了矩阵乘法.由于全连接层本身可以看做卷积核为 1 的卷积操作，因此，本文以卷积操作为例说明该转换的常用方式.对于一次卷积操作，其通常为两个规模相同的矩阵进行点对点的乘法，然后累加求和，该计算过程与一维向量乘法的计算过程吻合，因此，一次卷积操作可以转换为两个一维向量相乘，而卷积核的平移操作将一维向量的乘法转变为了矩阵的乘法.图 12 展示了输入特征图规模为 11×11，卷积核规模为 2×2，步长为 1 的二维卷积操作转换为矩阵乘法的过程.由于稀疏卷积神经网络模型中大约有 5%～50%的值为非零值，并不是极致的稀疏(1%或更少的非零值)，其在使用矩阵时是以稠密的格式而非稀疏的格式.

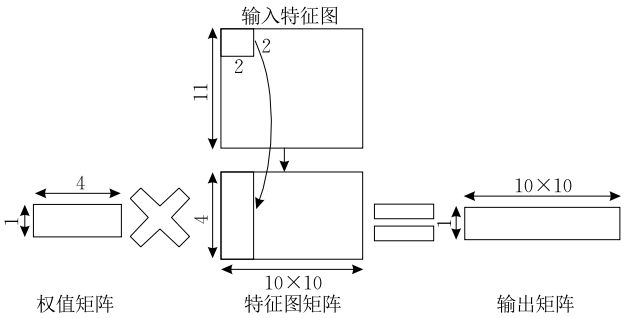

图 12 二维卷积操作转换为矩阵

对于跳过非零值结构的设计，其利用稀疏管理单元确定零值位置并提供给矩阵乘法单元，随后处理单元中的调度器根据相关信息提供非零值给计算单元进行计算.实验结果表明，Stratix 10 FPGA 利用该硬件结构执行矩阵乘法的性能(TOP/sec)比 Titan X Pascal GPU 的性能提升了 10%[14].

虽然这种硬件结构能跳过卷积神经网络模型中的无效计算，有效提高模型的执行效率，但同样有一些限制和提升的空间：

①矩阵乘法的转换是以数据复制为代价的，这对于带宽有限的 FPGA 开发板来说，模型整体的执行性能会降低[41].文献[42]对将 AlexNet 和 VGG16 网络模型中卷积操作转换为矩阵乘法后数据复制量进行了统计，结果表明转换后的每一层输入特征图数据量是原始数据量的 7.6～25 倍，中间特征图和权值数据量是原始数据量的 1.35～4.8 倍.

②稀疏矩阵乘法的计算是不规则的，可能会使计算在多个处理单元中分布不均匀，导致部分处理单元利用不充分，影响模型整体的执行性能.

(3)针对紧凑数据类型卷积神经网络的硬件结构.在对 FPGA 硬件结构进行设计时，数据的位数本身由开发者设定，因此，FPGA 通常能较好地支持自定义位数的数据类型，且在应用低精度数据类型时，通常不需要对原有硬件整体结构进行修改.由于在极端精度的情况下，如数据位数为 1，数据的乘加计算可以转换为简单的位运算，其对应的硬件结构与高精度数据类型对应的硬件结构不同，因此本文重点说明二值神经网络对应的加速硬件结构.文献

[43]对此提出了一种硬件结构的设计，具体硬件结构如图 13 所示.

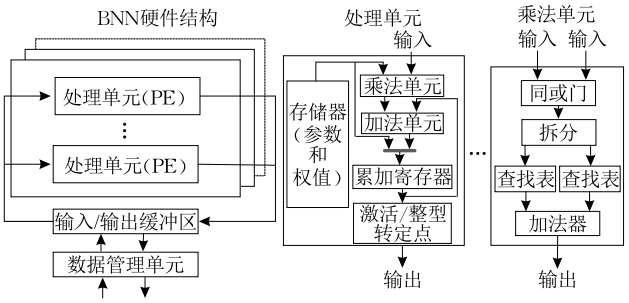

图 13　二值神经网络硬件结构[43]

在该硬件结构中，其用同或门代替了乘法器，这在很大程度上降低了 FPGA 片上资源的使用和计算的难度，提高了计算的效率. 由于其在硬件结构设计中重点关注的是计算结构的设计，忽略了数据交互相关的优化，如使用双缓冲区等，因此该硬件结构加速性能还有一定的提升空间.

文献[43]利用 Aria 10 FPGA 实现了该硬件结构，并将其执行性能与 CPU 和 GPU 进行了对比，结果表明 FPGA 的能效比远高于 CPU 和 GPU.

3.3.2　FPGA 加速卷积神经网络的设计思路

在采用 FPGA 加速卷积神经网络时，一般设计思路是根据提出的设计目的、针对的目标以及卷积神经网络算法本身的特点等进行数学建模，将其转换为利用数学公式求最大值或者最小值的问题. 数学建模完成后会利用穷举、动态规划等方法找到最优的解决方案，然后依据最优解决方案的参数、神经网络拓扑结构等生成最终的 FPGA 硬件结构. 其中设计目的通常为提高 FPGA 片上资源的利用、加速器的计算性能或者数据传输的能力，以及在优化资源使用的情况下找到最优的神经网络拓扑结构等. 针对目标通常为针对卷积神经网络的卷积层或者所有层等. 数学公式中参数是根据具体的目标、神经网络拓扑结构以及优化的策略等来选取. 同时，在数学建模时也会有一些约束条件，这些约束条件大部分是根据 FPGA 片上资源的限制转换而来的，如设计使用 LUT、DSP 资源的总数目要分别小于 FPGA 片上 LUT、DSP 资源的总数目.

总而言之，其核心思想是在 FPGA 片上资源的限制下，利用数学建模的方式寻找最佳加速性能的方案. 图 14 对卷积神经网络的 FPGA 加速设计流程进行了总结.

(1)设计的目标通常包括两个方面：提高 FPGA 资源的利用和 FPGA 的计算性能. 对于 FPGA 资源的利用，文献[40]重点关注的是片上逻辑资源和内存带宽的利用，这些资源没有被合理利用可能会引起计算吞吐和 FPGA 提供的内存带宽不匹配，即数据交换的速率无法满足计算的需求，进而影响 FPGA 加速的性能；文献[44]重点关注 FPGA 资源中 DSP 资源的使用，DSP 在 FPGA 中主要用于卷积神经网络的卷积计算，卷积计算是卷积神经网络中的关键操作，提高 DSP 的利用率能有效提高 FPGA 加速的性能；对于 FPGA 计算性能，大部分研究重点关注的是计算吞吐和整体执行时间，如文献[41]通过充分利用 FPGA 可重构能力来最大化整个系统的计算吞吐；文献[45]通过将 FPGA 片上资源划分成多个小处理器来提高计算的效率和获取更好的整体吞吐；文献[46]则利用多个 FPGA 加速卷积神经网络来提高整体的计算吞吐等. 除此之外，文献[47]针对特定的应用，为了找到最优的神经网络拓扑考虑了神经网络分类的准确率.

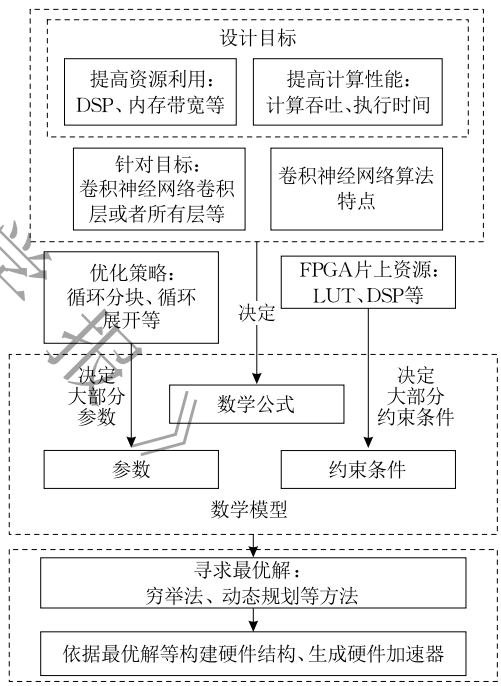

图 14　卷积神经网络的 FPGA 加速设计流程

(2)在设计目标确立后，会根据设计目标、卷积神经网络算法的特点和 FPGA 资源的限制等进行数学建模. 数学模型主要包括两个部分：模型对应的数学公式和 FPGA 片上资源对应的约束条件.

模型对应的数学公式主要是根据设计目标和卷积神经网络算法转换而来. 下面将根据学者们建立模型的方式，从以下几个方面对该模型公式的建立进行详细的说明.

① FPGA 资源的利用. 由于 FPGA 片上资源是

有限的,而且 FPGA 片上资源是否能有效利用在很大程度上决定了硬件加速器性能的好坏.因此,在设计时,通常会对 FPGA 片上的资源进行建模,通过模型来衡量对 FPGA 片上资源的利用,甚至有时会用于约束条件的建立.在对片上资源建模时主要考虑两个方面:计算资源和内存资源.对于计算资源建模,主要针对的是 DSP 资源,如文献[44]重点关注如何最大化计算速率,对应到片上资源的利用即为如何提高 DSP 资源的利用.因为 FPGA 中的 DSP 资源主要用于卷积神经网络中的卷积操作,而对于常见的算术运算,利用 DSP 计算将会更加高效.在 FPGA 加速设计中,最大计算速率通常归结为最大化 DSP 资源的利用.其针对 DSP 资源建立的模型公式为 DSP 的利用率,即在计算时,实际使用的 DSP 资源数目与所有可用 DSP 资源数目的比值,这样建立 DSP 资源模型能直观反映 DSP 资源利用情况.而文献[45]关注的是 DSP 资源具体的使用量,即根据处理器在加速每层时 DSP 资源的具体使用情况,对 DSP 具体使用数据进行建模,其模型对应的数学公式即为 DSP 资源具体使用的数目.这种 DSP 模型的建立不仅能有效体现 DSP 资源具体使用情况,还能用于约束条件的生成,如 DSP 资源使用的总数目小于 FPGA 片上 DSP 资源的总数目.

对于内存资源建模,一般会对 BRAM 和内存等资源进行建模.如在文献[45]中,将 FPGA 资源拆分成多个小处理器来加速卷积神经网络,对于每个小处理器的开销进行建模,其中包括了对内存资源和片上 BRAM 资源的建模.对于片上 BRAM 资源,其主要作用是存储数据,如被用来构造存储输入数据、输出数据的缓存区,在该研究中,为了更好地对 BRAM 资源建模,将由 BRAM 资源构造的缓存区划分为多个 bank,并对这些 bank 进行建模,其建立的模型主要为 bank 的个数,以及为对神经网络单层进行计算时每个 bank 存储的字节数.这样建模主要是为了反映 bank 资源存储数据的情况,方便存储约束条件的生成,如 bank 存储数据量小于 bank 最大存储量等.对于内存资源,重点关注的是内存带宽,其对应的模型公式为对神经网络单层进行计算时数据交换量与时间的比值,这样建模能直观反映数据的传输速率,并可以以此来观察数据传输速率是否与计算吞吐率匹配,同时还能用于相关约束条件的生成,如内存带宽不能超过 FPGA 提供的最大内存带宽等.

② 性能方面.在建模时通常会建立性能模型[48-49],并以此来衡量硬件加速的性能.性能模型主要分为两个方面:计算吞吐率和整体执行时间.

计算吞吐率的建模.如文献[40]重点关注 FPGA 加速设计中计算吞吐率和 FPGA 提供的内存带宽不匹配的问题,并根据此问题提出了 Computation To Communication(CTC)的定义并使用了 roofline 性能模型[49],CTC 主要用来描述每次内存访问对应的计算操作,其计算公式为操作总数与访存数据总量的比值.基于此,该研究对计算吞吐率建立了两个模型,一个是计算资源能达到的最大计算吞吐率,其对应的模型公式为操作总数与执行周期总数的比值;另外一个是根据 CTC 与内存带宽计算出来的最大计算吞吐率,其计算公式为 CTC 与内存带宽的乘积.而整体最大吞吐是这两个模型的较小值.文献[46]考虑到了 FPGA 片上资源的限制等因素,认为单个 FPGA 设计可能很难实现最优的能源效率,想利用多个 FPGA 来加速卷积神经网络.相比于单个 FPGA 计算吞吐率模型的建立,多个 FPGA 不仅要考虑每个 FPGA 的计算吞吐率,还需要考虑 FPGA 之间的通讯对计算吞吐的影响.该研究在建立整体计算吞吐率时利用了动态规划的思想,将多个 FPGA 整体计算吞吐率模型利用递归公式转换为了单个 FPGA 计算吞吐率模型,并且,在每次添加一个 FPGA 时都考虑了 FPGA 之间通信带来的影响.其中单个 FPGA 计算吞吐率模型与研究中类似,FPGA 之间通信模型公式为通信时间即每一层输出数据量与 FPGA 之间通信带宽的比值.

整体执行时间的建模.如文献[41]重点关注充分利用 FPGA 可重构能力来最大化硬件加速性能.在该研究中,其对整体的执行时间进行建模,也对其中各层的执行时间分别建模,其中卷积层时间模型公式为卷积层操作总数与实际每周期操作数的比值,其他层时间模型也是根据同样的方式建立,均为操作总数与实际每周期操作数的比值.而整体执行时间模型公式为这些层执行时间的总和,并且把整体执行时间最小值作为最佳的加速性能.文献[45]重点关注将 FPGA 拆分为多个小处理器来加速卷积神经网络.并且对每个小处理器计算每一层所需的时间进行建模,作为该处理器的性能模型.该研究在建立时间模型时,假设频率是一个固定值,而执行时间对应的模型公式为每层的计算规模.同时由于学者在设计加速器时会采取优化策略来加速计算效率,因此在执行时间的模型公式中需要考虑优化策略带来的影响.

③其它方面.主要是根据具体的应用或者需求建立其他的模型.如文献[47]考虑 DSP 资源优化和卷积神经网络分类准确率的问题,以分类准确率与 DSP 使用数目的比值作为性能模型;而有的研究则考虑了能源消耗的问题,并结合计算吞吐率,建立了吞吐/功耗的性能模型等.

在进行数学建模时,不仅要考虑模型公式的建立,还要考虑模型公式中关键参数的选择,以及约束条件的建立.模型公式中关键参数的取值决定了公式最后的结果.这些关键参数主要是根据采用的优化策略选取的,关于如何选取这些关键参数会在说明加速优化策略时具体说明.也有一些关键参数的选取依赖其它因素或者具有特殊性,如卷积神经网络具体的层数、FPGA 的个数等,本文不进行具体的说明.这些关键参数的取值并不是没有限制,这些限制主要与 FPGA 片上资源有关.而 FPGA 片上资源限制在数学模型是以这些参数的约束条件的形式体现的.这些约束条件一般分为两个方面:计算方面和内存方面.在计算方面,学者一般考虑的是计算资源的使用不能超过 FPGA 片上计算资源的上限.其中计算资源的使用情况在建立的数学模型中主要是由公式中的关键参数体现的,因此这些参数的值需要小于对应的 FPGA 片上计算资源的最大数目,如 LUT、DSP 计算资源等.在内存方面,一般考虑的是 FPGA 片上存储资源所能存储数据的最大值,以及内存带宽的限制,即数据传输速率不能超过内存带宽的上限.

(3)在数学建模完成之后,需要采取一些方法来寻求加速性能的最优解.有的研究利用穷举法,即尝试所有可能取值组合,然后在所有得出的加速性能中选取最优解.这种方式比较简单,但需要耗费大量的时间,特别当取值组合较多时,尝试所有的可能组合将变得特别艰难.也有的研究使用了动态规划的方法,在研究中,将多个 FPGA 求解最优加速性能的问题转换到单个 FPGA 求解最优加速性能的子问题,并给出了由单个 FPGA 到多个 FPGA 加速性能转换的递推公式,这样就可以利用动态规划的方法来解决求解最优解的问题.

(4)在最优解确定之后,根据关键参数的取值设计具体硬件实现的细节,然后利用开发工具验证硬件设计的正确性,验证完成后生成最终的硬件加速器.

3.3.3　FPGA 加速卷积神经网络的优化策略

优化在 FPGA 加速设计中是必不可少的,采用合理的优化策略能有效地提高 FPGA 计算性能,数据交换的能力以及片上资源的利用.同时,数学建模中的关键参数大部分根据优化策略进行选取.学者们在进行 FPGA 加速设计时利用的优化策略主要分为两个方面:计算优化[50-51]和内存优化[52-55].

(1)面向计算的优化.对于卷积神经网络,尤其在卷积阶段,涉及到大量的乘加计算,采用合理的计算优化策略能有效提高系统整体的性能.在计算方面,学者们进行优化的方式有很多,本文主要介绍其中重要的 3 种方式:

①计算并行优化.主要是指卷积神经网络中计算的并行,提高计算的并行度能有效提高 FPGA 加速的性能[56-59].而卷积神经网络计算的并行有多种情况,如层与层之间的并行与层内的并行等.文献[59]对卷积神经网络中的并行方式进行了详细的说明,主要分为 4 种:不同层之间的并行,由于卷积神经网络相邻层之间有数据依赖,所以不同层之间并行较难;不同输出特征图之间的并行,对于同层的不同输出特征图,它们之间是没有数据依赖的,可以独立地计算,理论上是可以完全并行的;像素点间的并行,输出特征图中的每个像素点的计算是并行的;像素点计算内的并行,在计算每个输出特征图的像素点时会进行大量的乘加操作,而这些乘法之间是相互独立的,可以进行并行的计算.在实际并行优化时,大部分学者都会考虑不同输出特征图之间的,像素点间的和像素点计算内的这 3 种并行.然而,在进行这些优化时,也需要考虑一些资源、硬件等方面的问题,如在进行像素点间的并行优化时,需要考虑片上计算资源和存储资源的大小,可能计算资源无法支持所有输出像素同时计算或者存储资源无法存储所有计算所需的操作数.大部分学者面对此类问题的解决方案是将输出特征图等划分成更小的部分进行分批计算,这样能较好地解决资源限制的问题;在进行像素点计算内的并行优化时,需要考虑 FPGA 片上存储资源是否能一次传输并行乘法计算所需的所有数据,在文献[60]中就指出,利用 FPGA 片上 BRAM 资源存储计算数据时,由于 BRAM 是双端口的,无法在一个周期内将 BRAM 中存储的计算所需数据传输给计算单元,导致计算只是部分并行.在该文献中,学者的解决方案主要是用大量的寄存器来代替 BRAM,这样很好地解决了数据传输的问题,但是这样会使 FPGA 片上 BRAM 资源利用率不高,造成资源的浪费,同时可能会占用大量的其他资源,如 LUT,这样会给其它的设计造成更多资源

上的限制.

② 循环分块和循环展开. 卷积神经网络算法大部分是利用循环来实现的. 图 15 中的代码片段展示了一个典型的利用输入特征图和卷积核计算输出特征图的卷积操作过程.

```
for( row=0; row <R; row++) {
  for ( co l=0; col <C; col++) {
    for ( to=0; to <M; to++) {
      for ( ti=0; ti <N; ti++) {
        for ( i=0; i <K; i++) {
          for ( j=0; i <K; j++) {
            output_fm[to][row][col]+=
              weights[to][ti][i][j] *
              input_fm[ti][S * row+i][S * col+j];
} } } } } }
```

图 15 一个卷积层的伪代码[40]

针对图 15 中的循环,一般采用循环分块和循环展开进行优化. 循环分块可以将循环拆分成更小的模块进行计算,这样可以使片上存储数据更多次的进行重用,减少访问内存的次数,提高计算的效率,具体分块伪代码如图 16 所示;循环展开可以让更多的计算并行进行,使计算资源得到更加充分的利用. 在对循环进行分块和展开时,需要考虑循环中数据之间的关系,不同数据关系对应的硬件设计也是不同的,文献[40]对此进行了详细的说明,例如,当卷积核较小时不需要进行分块处理. 由于循环分块大小影响了计算的并行度,部分决定了单位时间内进行的计算操作,所以在针对计算性能进行数学建模时,学者们一般会将分块的大小,如输出特征图行分块大小、输出特征图列分块大小等,作为模型中的关键参数,并进行自身约束,如行分块大小必须小于总行数等,以及片上资源的限制给出相应的约束条件.

```
for (row=0; row <R; row+=tr ) {
 for (co l=0; col <C; col+=tc ) {
  for (to=0; to <M; to+=tm ) {
   for (ti=0; ti <N; ti+=tn ) {
    for (tile_row=row; tile_row<min(row+tr, R); tile_row++){
     for (tile_col=col; tile_col <min(col+tc, C ); tile_col++) {
      for (tile_to=to; tile_to <min(to+tm, M ); tile_to++) {
       for (tile_ti=ti; tile_ti <min(ti+tn, N ); tile_ti++) {
        for (i=0; i <K; i++) {
         for (j=0; i <K; j++) {
          output_fm[to][row][col]+=
           weights[to][ti][i][j] *
           input_fm[ti][S * row+i][S * col+j];
} } } } } } } } } }
```

图 16 循环分块的伪代码[40]

③ 循环流水技术. 循环流水技术是加速神经网络的一种关键技术. 虽然设计人员可以在高层次综合工具中进行相应设置实现循环流水优化,但是,神经网络算法中的循环可能存在数据依赖的情况,所以,循环流水技术的应用不只是简单设置参数可以实现的. 在文献[40]中,学者采用一种基于多面体的优化架构[61]来解决循环体依赖的问题,有效地提高了循环流水的性能.

(2) 面向内存的优化. 因为 FPGA 片上存储资源有限,无法存储卷积神经网络中所有计算所需的数据,需要频繁与 FPGA 片下存储进行数据的交换. 而这些数据交换的开销,对于系统整体的加速性能来说,是不可忽视的,而且相比于计算,数据传输更加消耗资源[62-64]. 因此,学者们在进行加速设计时,会采取一些优化策略来减少数据交换的次数,进而提高系统整体的性能. 本文主要介绍以下几种重要的优化方式:

① 减小数据精度. 卷积神经网络中计算涉及的操作数一般默认是 32 位或者 64 位的浮点数,但是有的文献[65-70]提出,在不损失准确率或者准确率损失较小的情况下,对于计算的操作数,用定点数的形式代替浮点数的形式表示是可行的,而且定点数的位数可以大幅度下降到 16 位或者 16 位以下. 相比于浮点数,使用位数相对较少的定点数,如 8 位和 16 位等,能够有效减少位宽,使 FPGA 片上存储资源存储更多的数据,减少访问片下资源的次数,同时还能增加吞吐量和计算的性能等. 学者们在选择定点数位数时,一般会进行准确率的分析,确定选择的定点数位数对应的准确率与浮点数对应的准确率一致或者有相对可以忽略的损失. 文献[71]针对如何综合考察不同的数据量化策略和如何权衡定点数位数和准确率的问题提出了一种动态精度数据量化的方法. 其中动态指的是在确定定点数位数后,神经网络不同层和不同的特征图集合对应的定点数整数和小数的位数是不同的. 该方法有两个阶段:权重量化阶段和数据量化阶段. 其中权重量化阶段针对的是每一层的权重;数据量化阶段针对的是两层之间一些特征图. 然后每个阶段都在给定定点数位数的情况下找出最佳整数和小数位数. 利用这种方法能找到更短的定点数位数而且能维持与浮点数相近的准确率. 但是这种方法比起静态量化的方式更加复杂,而且需要设计特殊的硬件来支持这种方法.

② 合理利用本地存储. 本地存储指的是 FPGA 片上存储结构,其与 FPGA 片上计算单元之间数据交互的开销要远小于 FPGA 与片外存储之间数据交互的开销,因此,合理地利用本地存储能有效减少 FPGA 访问片外存储的次数,有效提升整体的执行效率. 如在文献[55]中,学者针对卷积神经网络提出

了一种基于数据流的硬件设计，在该硬件结构中，卷积神经网络每一层的处理单元之间是通过 FPGA 片上存储单元组成的 FIFO 进行通信，而不是通过 FPGA 片外存储，这样在很大程度上减少了 FPGA 访问片外存储的开销.

③基于双缓冲区的“乒-乓”数据传输机制. 该机制的核心思想为利用数据计算的时间掩盖数据传输的时间，即当计算单元在使用缓冲区 A 进行数据处理时，缓冲区 B 进行数据传输，当计算单元在使用缓冲区 B 进行数据处理时，缓冲区 A 进行数据传输. 这种方式虽然会使用额外的硬件资源，但是能有效掩盖数据传输带来的开销，提高整体的执行性能.

④层间数据复用. 卷积神经网络是流式的，计算时通常是逐层计算，上层的计算输出会作为下层计算的输入，而这些两层之间的中间计算结果需要存储在片外存储中. 因此，在每层计算的开始和结束，都需要片下存储的访问，来进行读或者写的操作. 这占据了大量的片外存储的访问. 而大部分学者在进行数据复用时主要考虑的是单层内的数据复用，没有考虑层与层之间的数据复用. 文献[72]提出了层与层之间融合的方式来进行数据复用，在该方法中，学者利用一个金字塔状的多层滑动窗口对输入数据进行处理，从而直接得到几层之后的数据结果. 同时，还提出了对应的优化框架，探究如何划分融合层，层内划分等. 根据该研究的实验结果，这种方法减小了高达 95%的片外存储访问.

3.4　递归神经网络的 FPGA 加速设计

由于利用 FPGA 加速递归神经网络是目前新兴的研究，相关研究并不丰富，同时递归神经网络和卷积神经网络有很多相似的地方，利用 FPGA 加速递归神经网络的设计思想在本质上与加速卷积神经网络的基本是一致的，因此，本文重点举例说明目前常用的基于递归神经网络的 LSTM(Long Short-Term Memory)模型[73]和 GRU(Gated Recurrent Unit)模型[74]对应的硬件结构.

对于 GRU 模型，文献[32]指出 GRU 模型中稠密的矩阵向量乘法操作占据了模型大部分执行时间，是 GRU 模型中的关键瓶颈，并针对该操作设计了相应的硬件结构，硬件结构如图 17 所示. 同时，其还根据 GRU 模型推理阶段的特点提出了记忆化的优化方式，通过牺牲 3 倍的内存空间换取一半稠密矩阵向量乘法不需要计算的效果.

在该硬件结构设计中，矩阵被拆分为了多个列块，每个浮点乘法器计算每个列块中的一行，通过浮点乘法器的并行执行让矩阵行和列的计算并行，进而提高计算的效率. 但由于该硬件结构重点关注的是计算的并行，并没有对数据交互进行优化，系统整体的执行性能还有提升的空间.

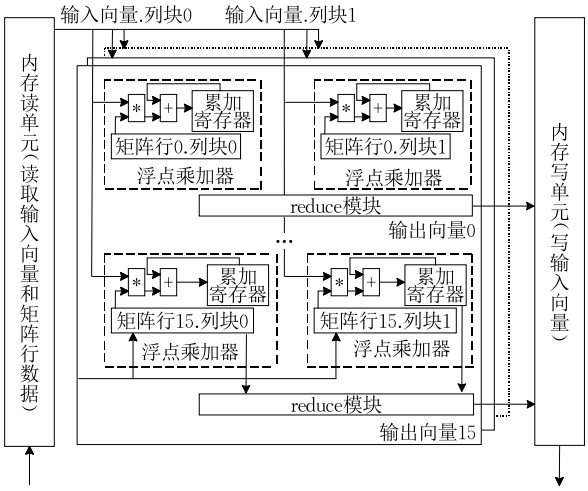

图 17　稠密矩阵向量乘法操作[32]

在文献[32]中，学者利用 FPGA 实现了该硬件结构，并将其执行性能与 CPU 和 GPU 进行了对比，结果表明 FPGA 的能效比要优于 CPU 和 GPU.

对于 LSTM 模型，韩松等人在文献[75]中对稀疏 LSTM 模型进行了深入的研究，利用深度压缩的算法对稀疏 LSTM 模型进行了压缩，针对压缩后多核并行负载不均衡的问题提出了负载平衡的剪枝算法，保证剪枝后分配到每个核的计算量相当，并根据处理后的 LSTM 模型设计出了专用处理器架构. 结合新的算法、专用编译器以及提出的专用处理器架构，对于一个实际使用的 LSTM 模型，其在 FPGA 上实现的性能比 Titan X Pascal GPU 实现的性能高出 3 倍，而功耗要低 3.5 倍. 其加速的 FPGA 硬件结构如图 18 所示，通道的硬件结构如图 19 所示.

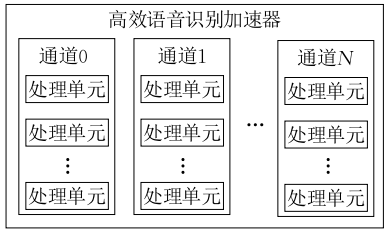

图 18　高效语音识别加速器结构[75]

该 FPGA 硬件结构由多个通道单元组成，每个通道单元中包含多个处理单元、激活向量队列单元、加法树单元等. 其中激活向量队列单元包含多个 FIFO，其 FIFO 个数与处理单元(PE)个数对应，该

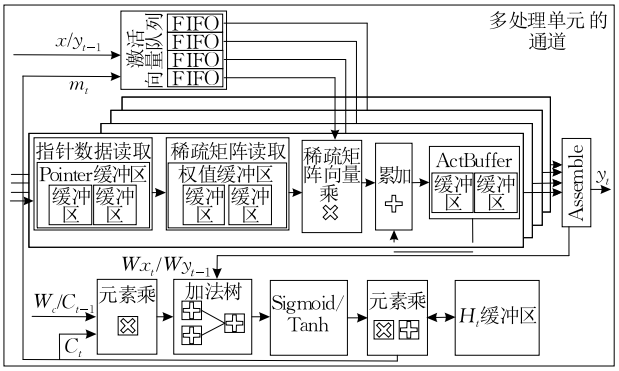

图 19 通道的硬件结构[75]

单元的主要作用是解决由于不同处理单元工作负载不平衡所带来的问题,即运行较快的处理单元在完成工作后可以从对应 FIFO 中获取数据继续执行而不需要等待其它处理单元均执行完成后才能继续执行;指针缓冲区和权值缓冲区主要存储和输出权值矩阵以及表明矩阵中非零值的位置;ActBuffer 主要用于存储稀疏矩阵向量乘单元计算的局部结果;累加器则负责对存储稀疏矩阵向量乘单元新计算结果与存储在 ActBuffer 中的局部结果进行求和计算.处理单元中的缓冲区均为双缓冲区,并以"乒-乓"的方式进行数据传输,这样能利用计算时间来掩盖数据传输的时间,有效提高系统整体的执行效率.

4 研究展望

近几年,深度学习技术已经应用于很多重要的领域,但由于深度学习本身的特性以及应用对功耗、实时性等的要求,利用硬件加速器来加速深度学习是必然的趋势.目前,面向深度学习的 FPGA 硬件加速器的设计方兴未艾,今后一段时间内的研究工作,主要集中在以下几个方面:

(1)更完善的 FPGA 的开发环境.目前,虽然有高层次综合工具支持采用高级语言开发 FPGA,并且各公司也提供了深度学习相关的库和实现了对一些深度学习框架的支持等,但是这些库和支持的框架依然存在一些限制,如支持的板卡有限,只支持部分深度学习框架等,同时对于大部分软件开发人员来说,利用高层次综合工具编写高效的实现代码依然不是一件容易的事情.未来 FPGA 加速深度学习算法的库以及支持的深度学习框架应该更加完善,高层次综合工具应该为软件开发人员提供更好的开发环境,让开发人员更专注于深度学习算法的探究.

(2)更优化的 FPGA 通信机制.目前,大部分 FPGA 加速深度学习算法的主要瓶颈是带宽,即 FPGA 与其它硬件之间的通信效率较低.在未来,除了使用传统堆带宽和减少深度学习算法模型带宽需求的方式解决带宽问题外,可以从体系结构的角度对 FPGA 通信机制进一步优化,通过合理利用计算时间掩盖通信延迟,进而解决该问题.

(3)更完善的 FPGA 云服务.FPGA 与云计算的结合给 FPGA 加速深度学习的发展带来了新的契机.目前,FPGA 云服务处于刚起步的阶段,还有很多不完善的地方以及很多值得研究的内容,如 FPGA 硬件资源的虚拟化、虚拟化 FPGA 的任务迁移和负载均衡、高效并行的多机多 FPGA 异构加速体系结构等.

(4)进一步优化采用 FPGA 加速压缩后的深度学习算法.目前,压缩深度学习算法模型是深度学习算法主要发展趋势之一,对算法进行压缩能有效地减少数据传输和数据存储的开销.然而,目前面向压缩后的深度学习算法进行 FPGA 加速设计的研究较少.未来,采用 FPGA 加速压缩后的深度学习算法也是研究热点之一.

5 总 结

由于深度学习算法及其应用领域的特点,采用硬件加速深度学习算法已经成为如今的发展趋势.而 FPGA 与 GPU 相比有较高的能效比,与 ASIC 相比有较好的灵活性、较短的开发周期等优势,更符合当前深度学习算法的发展趋势.相信随着 FPGA 加速深度学习技术研究的成熟,FPGA 硬件加速器将被应用到更多的领域中.

参 考 文 献

[1] Sze V, Chen Y H, Einer J, et al. Hardware for machine learning: Challenges and opportunities//Proceedings of the Custom Integrated Circuits Conference. Austin, USA, 2017: 1-8

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

[3] Farabet C, Couprie C, Najman L, et al. Learning hierarchical features for scene labeling. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(8): 1915-1929

[4] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 2015: 1-9

[5] Tompson J, Jain A, Lecun Y, et al. Joint training of a convolutional network and a graphical model for human pose estimation//Proceedings of the Neural Information Processing Systems. Montreal, Canada, 2014: 1799-1807

[6] Mikolov T, Deoras A, Povey D, et al. Strategies for training large scale neural network language models//Proceedings of the Automatic Speech Recognition and Understanding (ASRU). Waikoloa, USA, 2011: 196-201

[7] Hinton G, Deng L, Yu D, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 2012, 29(6): 82-97

[8] Sainath T N, Mohamed A-r, Kingsbury B, et al. Deep convolutional neural networks for LVCSR//Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. Vancouver, Canada, 2013: 8614-8618

[9] Jean S, Cho K, Memisevic R, et al. On using very large target vocabulary for neural machine translation//Proceedings of the Meeting of the Association for Computational Linguistics. Baltimore, USA, 2014: 1-10

[10] Sutskever I, Vinyals O, Le Q. Sequence to sequence learning with neural networks//Proceedings of the Neural Information Processing Systems. Montreal, Canada, 2014: 3104-3112

[11] Putnam A, Caulfield A M, Chung E S, et al. A reconfigurable fabric for accelerating large-scale datacenter services//Proceedings of the ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). Minneapolis, USA, 2014: 13-24

[12] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444

[13] Marhon S A, Cameron C J F, Kremer S C. Recurrent neural networks//Processing of the Handbook on Neural Information Processing. Berlin, Germany, 2013: 29-65

[14] Nurvitadhi E V G, Sim J, et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks ?//Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2017: 5-14

[15] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman Coding. Fiber, 2015, 56(4): 3-7

[16] Gysel P, Motamedi M, Ghiasi S. Hardware-oriented approximation of convolutional neural networks. arXiv preprint arXiv:1604.03168, 2016

[17] Li F, Liu B. Ternary weight networks. arXiv preprint arXiv: 1605.04711, 2016

[18] Venkatesh G, Nurvitadhi E, Marr D. Accelerating deep convolutional networks using low-precision and sparsity//Proceedings of the International Conference on Acoustics, Speech, and Signal Processing. New Orleans, USA, 2017: 2861-2865

[19] Courbariaux M, Bengio Y, David J-P. Binaryconnect: Training deep neural networks with binary weights during propagations//Proceedings of the Advances in Neural Information Processing Systems. Montreal, Canada, 2015: 3123-3131

[20] Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+1 or −1. arXiv: Learning, 2016

[21] Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: ImageNet classification using binary cnvolutional nural networks//Proceedings of the European Conference on Computer Vision. Amsterdam, The Netherlands, 2016: 525-542

[22] Albericio J, Judd P, Hetherington T, et al. Cnvlutin: Ineffectual-Neuron-Free deep neural network computing//Proceedings of the ACM/IEEE International Symposium on Computer Architecture. Seoul, Korea,2016: 1-13

[23] Cavigelli L, Gschwend D, Mayer C, et al. Origami: A convolutional network accelerator//Proceedings of the Great Lakes Symposium on VLSI. Pittsburgh, USA, 2015: 199-204

[24] Qadeer W, Hameed R, Shacham O, et al. Convolution engine: Balancing Efficiency & Flexibility in Specialized Computing//Proceedings of the International Symposium on Computer Architecture. Tel-Aviv, Israel, 2013: 24-35

[25] Chen Y-H, Krishna T, Emer J, et al. 14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks //Proceedings of the 2016 IEEE International Solid-State Circuits Conference (ISSCC). San Francisco, USA, 2016: 262-263

[26] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A convolutional neural network accelerator with In-situ analog arithmetic in crossbars//Proceedings of the ISCA. Seoul, Korea, 2016: 14-26

[27] Andri R, Cavigelli L, Rossi D, et al. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights//Proceedings of the IEEE Computer Society Annual Symposium on VLSI. Pittsburgh, USA, 2016: 236-241

[28] Gokmen T, Vlasov Y. Acceleration of deep neural network training with resistive cross-point devices: design considerations. Front Neurosci, 2016, 10(51): 333

[29] Zhu J, Qian Z, Tsui C-Y. LRADNN: High-throughput and energy-efficient deep neural network accelerator using low rank approximation//Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC). Macao, China, 2016: 581-586

[30] Hong I, Park S, Park J, et al. A 1.9 nJ/pixel embedded deep neural network processor for high speed visual attention in a mobile vision recognition SoC//Proceedings of the Solid-State Circuits Conference (A-SSCC), 2015 IEEE Asian. Xiamen, China, 2015: 1-4

[31] Chen D, Singh D. Fractal video compression in OpenCL: An evaluation of CPUs, GPUs, and FPGAs as acceleration platforms//Proceedings of the Design Automation Conference

(ASP-DAC), 2013 18th Asia and South Pacific. Yokohama, Japan, 2013: 297-304

[32] Nurvitadhi E, Sim J, Sheffield D, et al. Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC//Proceedings of the International Conference on Field Programmable Logic and Applications. Lausanne, Switzerland, 2016: 1-4

[33] Lacey G, Taylor G W, Areibi S. Deep learning on FPGAs: Past, present, and future. arXiv preprint arXiv: 1602.04283, 2016

[34] Akopyan F, Sawada J, Cassidy A S, et al. TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34(10): 1537-1557

[35] Chen T, Du Z, Sun N, et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM Sigplan Notices, 2014, 49(4): 269-284

[36] Chen Y, Luo T, Liu S, et al. Dadiannao: A machine-learning supercomputer//Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture. Cambridge, UK, 2014: 609-622

[37] Jouppi N P, Young C, Patil N, et al. In-dtacenter performance aalysis of a tnsor pocessing uit. SIGARCH Computer Architecture News, 2017, 45(2): 1-12

[38] Choi Y-k, Cong J, Fang Z, et al. A quantitative analysis on microarchitectures of modern CPU-FPGA platforms//Proceedings of the 53rd Annual Design Automation Conference. Austin, USA, 2016: 109

[39] Tapiador R, Riosnavarro A, et al. Comprehensive evaluation of OpenCL-based convolutional neural network accelerators in xilinx and altera FPGAs. arXiv preprint arXiv: 1609.09296, 2016

[40] Zhang C, Li P, Sun G, et al. Optimizing fpga-based accelerator design for deep convolutional neural networks//Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2015: 161-170

[41] Suda N, Chandra V, Dasika G, et al. Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2016: 16-25

[42] Zhang C, Fang Z, Zhou P, et al. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks//Proceedings of the 35th International Conference on Computer-Aided Design. Austin, USA, 2016: 12

[43] Nurvitadhi E, Sheffield D, Sim J, et al. Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC//Proceedings of the International Conference on Field-Programmable Technology. Xi'an, China, 2016: 77-84

[44] Rahman A, Lee J, Choi K. Efficient FPGA acceleration of convolutional neural networks using logical－3D compute array//Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE). Dresden, Germany, 2016: 1393-1398

[45] Shen Y, Ferdman M, Milder P. Maximizing CNN accelerator efficiency through resource partitioning//Proceedings of the 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA). Toronto, Canada, 2017: 535-547

[46] Zhang C, Wu D, Sun J, et al. Energy-efficient CNN implementation on a deeply pipelined FPGA cluster//Proceedings of the 2016 International Symposium on Low Power Electronics and Design. San Francisco Airport, USA, 2016: 326-331

[47] Abdelouahab K, Bourrasset C, Pelcat M, et al. A holistic approach for optimizing DSP block utilization of a CNN implementation on FPGA//Proceedings of the 10th International Conference on Distributed Smart Camera. Paris, France, 2016: 69-75

[48] Wang Z, He B, Zhang W, et al. A performance analysis framework for optimizing OpenCL applications on FPGAs//Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA). Barcelona, Spain, 2016: 114-125

[49] Williams S, Waterman A, Patterson D. Roofline: An insightful visual performance model for multicore architectures. Communications of the ACM, 2009, 52(4): 65-76

[50] Sarkar S, Majumder T, Kalyanaraman A, et al. Hardware accelerators for biocomputing: A survey//Proceedings of the 2010 IEEE International Symposium on Circuits and Systems. Paris, France, 2010: 3789-3792

[51] Herbordt M C, VanCourt T, Gu Y, et al. Achieving high performance with FPGA-based computing. Computer, 2007, 40(3): 50

[52] Peemen M, Setio A A, Mesman B, et al. Memory-centric accelerator design for convolutional neural networks//Proceedings of the 2013 IEEE 31st International Conference on Computer Design (ICCD). Asheville, USA, 2013: 13-19

[53] Peemen M, Mesman B, Corporaal H. Inter-tile reuse optimization applied to bandwidth constrained embedded accelerators//Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). Grenoble, France, 2015: 169-174

[54] 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//Proceedings of the, 2016 IEEE 34th International Conference on Computer Design (ICCD). Scottsdale, USA, 2016: 320-327

[55] Wang D, An J, Xu K. PipeCNN: An OpenCL-Based FPGA accelerator for large-scale convolution neuron networks. arXiv preprint arXiv:161102450, 2016

[56] Sankaradas M, Jakkula V, Cadambi S, et al. A massively parallel coprocessor for convolutional neural networks//Proceedings of the 2009 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors. Boston, USA, 2009: 53-60

[57] Cadambi S, Majumdar A, Becchi M, et al. A programmable parallel accelerator for learning and classification//Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques. Vienna, Austria, 2010: 273-284

[58] Ovtcharov K, Ruwase O, Kim J-Y, et al. Accelerating deep convolutional neural networks using specialized hardware. Microsoft Research Whitepaper, 2015, 2(11): 11-14

[59] Motamedi M, Gysel P, Akella V, et al. Design space exploration of fpga-based deep convolutional neural networks//Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC). Macao, China, 2016: 575-580

[60] Zhou Y, Jiang J. An FPGA-based accelerator implementation for deep convolutional neural networks//Proceedings of the 2015 4th International Conference on Computer Science and Network Technology (ICCSNT). Harbin, China, 2015: 829-832

[61] Zuo W, Liang Y, Li P, et al. Improving high level synthesis optimization opportunity through polyhedral transformations//Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, USA, 2013: 9-18

[62] Dally B. Power, programmability, and granularity: The challenges of exascale computing//Proceedings of the 2011 IEEE International Test Conference. Anaheim, USA, 2011: 12

[63] Horowitz M. 1.1 Computing's energy problem (and what we can do about it)//Proceedings of the 2014 IEEE International Solid-State Circuits Conference on Digest of Technical Papers (ISSCC). San Francisco, USA, 2014: 10-14

[64] Hameed R, Qadeer W, Wachs M, et al. Understanding sources of inefficiency in general-purpose chips. ACM SIGARCH Computer Architecture News, 2010, 38(3): 37-47

[65] Courbariaux M, David J-P, Bengio Y. Low precision storage for deep learning. arXiv preprint arXiv:14127024, 2014

[66] Gupta S, Agrawal A, Gopalakrishnan K, et al. Deep learning with limited numerical precision. CoRR, abs/150202551, 2015, 392

[67] Trinh H-P, Duranton M, Paindavoine M. Efficient data encoding for convolutional neural network application. ACM Transactions on Architecture and Code Optimization (TACO), 2015, 11(4): 49

[68] Anwar S, Hwang K, Sung W. Fixed point optimization of deep convolutional neural networks for object recognition//Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). South Brisbane, Australia, 2015: 1131-1135

[69] Jiang J, Hu R, Mikel L, et al. Accuracy evaluation of deep belief networks with fixed-point arithmetic. Computer Modelling & New Technologies, 2014, 18(6): 7-14

[70] Lian R L. A Framework for FPGA-Based Acceleration of Neural Network Inference with Limited Numerical Precision via High-Level Synthesis with Streaming Functionality. Toronto, Ontario, Canada: University of Toronto, 2016

[71] Qiu J, Wang J, Yao S, et al. Going deeper with embedded fpga platform for convolutional neural network//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, USA, 2016: 26-35

[72] Alwani M, Chen H, Ferdman M, et al. Fused-layer CNN accelerators//Proceedings of the 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Taipei, China, 2016: 22: 12-21

[73] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780

[74] Cho K, Van Merrienboer B, Bahdanau D, et al. On the properties of neural machine translation: encoder-decoder approaches// Proceedings of the Syntax, Semantics and Structure in Statistical Translation. Doha, Qatar, 2014: 103-111

[75] Han S, Kang J, Mao H, et al. ESE: Efficient speech recognition engine with sparse LSTM on FPGA//Proceedings of the Symposium on Field Programmable Gate Arrays (FPGA). Monterey, USA, 2017: 75-84

WU Yan-Xia, Ph. D., associate professor. Her current research interests include compiler technology and computer architecture.

LIANG Kai, master. His current research interests include computer architecture.

LIU Ying, Ph. D., assistant professor. Her current research interests include heterogeneous parallel program, compilation system and related tools.

CUI Hui-Min, Ph. D., associate professor. Her current research interests include parallel computing, parallel compiler and parallel programming.

Background

With the coming of big data era, deep learning plays a critical role in extracting the meaningful information from massive data, and has also been widely applied in the domains of artificial intelligence and multimedia. In order to fulfill requirements such as high throughput, low energy consumption, accelerators are used to accelerate the application of deep learning. And the FPGA accelerators attract more and more attention of scholars.

Recently, there has been a lot of exploration work on FPGA accelerators for deep learning. On the basis of previous research work, this paper firstly reviews the characteristics and the trend of deep learning algorithms and analyzes the advantages and technical challenges of accelerating deep learning on FPGAs. Then, we present the CPU-FPGA platform from the aspects of SoC FPGA and normal FPGA and comparatively analyze the differences of the data communication between CPU and FPGA of the two platforms. Additionally, on the basis of introducing the development environment of accelerating deep learning on FPGAs, the FPGA-based accelerator design for convolutional neural network is reviewed from the hardware architecture and design ideas in detail. Finally, we prospect the research on FPGA-accelerated deep learning algorithms.

This work is supported by the National High Technology Research and Development Program (863 Program) (2016YFB1000402), the Natural Science Foundation of Heilongjiang Province (F2018008), and the foundation for distinguished young scholars of Harbin (2017RAYXJ016).