https://doi.org/10.1051/jnwpu/20203810216

面向图像识别的深度学习 VLIW 处理器设计

李林[1,2], 张盛兵[1], 吴鹃[3]

(1.西北工业大学 计算机学院, 陕西 西安 710072; 2.北京微电子技术研究所 设计四部, 北京 100076;
3.西安职业技术学院 动漫软件学院, 陕西 西安 710077)

摘 要:为了适应航空航天领域高分辨率图像识别和本地化高效处理的需求,解决现有研究中计算并行性不足的问题,在对深度卷积神经网络模型各层计算优化的基础上,设计了一款可扩展的多处理器簇的深度学习超长指令字(VLIW)处理器体系结构。设计中采用了特征图和神经元的并行处理,基于 VLIW 的指令级并行,多处理器簇的数据级并行以及流水线技术。FPGA 原型系统测试结果表明,该处理器可有效完成图像分类和目标检测应用;当工作频率为 200 MHz 时,处理器的峰值性能可以达到 128 GOP/s;针对选取的测试基准,该处理器的计算速度至少是 CPU 的 12 倍,是 GPU 的 7 倍;对比软件框架运行结果,处理器的测试精度的平均误差不超过 1%。

关 键 词:图像识别;深度学习;卷积神经网络;超长指令字(VLIW);处理器;可扩展
中图分类号:TP389.1 **文献标志码**:A **文章编号**:1000-2758(2020)01-0216-09

近年来,航拍图像与卫星遥感图像呈现出高分辨率的特点,为了快速地进行图像识别并减少数据传输过程带来的延时,本地化高效处理的需求显得日益迫切。与此同时,深度学习技术在计算机视觉、语音识别、自然语言处理、自动驾驶汽车等人工智能领域取得了巨大突破[1]。特别是在图像识别领域,深度学习技术有效地提高了图像识别的速度和准确率,因此本文将探索将其融入到航空航天领域图像识别的本地化高效处理应用中。

在深度学习图像识别领域,Lecun 等人在 1998 年提出的卷积神经网络 LeNet-5 在手写数字的识别中取得了当时最好的效果[2]。Krizhevsky 等人在 2012 年 ImageNet 大规模视觉识别挑战赛中提出的深度卷积神经网络 AlexNet[3]取得了历史性的突破,不仅夺得了冠军,而且掀起了深度学习研究的热潮。自此之后,更多优秀的深度神经网络算法被提出,使得深度卷积神经网络成为图像识别领域的最佳深度学习算法。

在现阶段,深度卷积神经网络算法的一种实现方式是以软件编程的形式在通用 CPU 或 GPU 上实现。通用 CPU 具有很高的灵活性和较好的并行计算能力,但是深度卷积神经网络算法在其上并不能取得较好的执行效率。GPU 因其特有的多核结构和出众的并行计算能力而被广泛应用于深度神经网络的训练和推理[3-5],但是其功耗却比较大。为了获得较好的性能和较低的功耗,以及应对无 GPU 和嵌入式应用场景,探索面向特定领域体系结构(DSA)的硬件算法加速实现成为当今的一个研究热点[6]。因此,深度卷积神经网络的另一种实现方式是在 FPGA 或 ASIC 上构建专用的硬件算法加速器或协处理器,本文将采用此方法设计面向图像识别的深度学习超长指令字(VLIW)处理器,从而实现航空航天领域图像的本地化高效处理。

在当前的专用硬件算法处理器研究中,众多研究者都是围绕提升处理性能和降低功耗来进行研究。Farabet 等人提出了一种适用于通用视觉算法计算的可扩展数据流硬件结构 NeuFlow 实现了街景分析应用[7]。Peemen 等人提出了一种以层次化存储器为中心的加速器结构来提高卷积运算的性能并降低存储访问的开销[8]。Chen 等人侧重改善片上存储的影响、性能和功耗提出了一种高吞吐的加速器 DianNao[9]。Du 等针对移动视觉处理设计了一

收稿日期:2019-01-08
作者简介:李林(1982—),西北工业大学博士研究生,主要从事微处理器体系结构及集成电路设计研究。

种二维映射结构的加速器 ShiDianNao,有效地提高了卷积运算的性能[10]。Google 公司开发的应用于数据中心的 TPU 可运行多种机器学习算法,性能是通用 CPU 和 GPU 的 15~30 倍,能耗比是其 30~80 倍[11]。以上研究均为各研究机构针对特定应用场景提出的专用深度学习处理器结构,它们各具特色及优点,但是并没有完全挖掘出深度卷积神经网络计算的并行性。本文在对深度卷积网络模型中各层计算的分析和优化基础上,采用多种并行处理技术设计了一种可扩展的多处理器簇的深度学习 VLIW 处理器。通过原型系统验证,该处理器可有效完成图像的分类和目标检测应用,当工作频率为 200 MHz 时,处理能力可以达到 128 GOP/s。

1　深度卷积神经网络模型

在图像识别领域,随着新的算法的不断提出,图像识别的准确率在不断提高,网络的深度和复杂度也在不断增加[3-5]。然而,这些新的算法都是基于深度卷积神经网络模型的,并且它们都是由一些基本的计算层组成。图 1 所示的典型深度卷积神经网络模型中包含了这些基本的计算层,它们分别是:卷积层、池化层、非线性激活函数层、归一化层、全连接层以及 Softmax 输出层。下面将介绍这些基本层的计算,并通过优化计算抽象出本文设计的处理器所支持的基本运算。

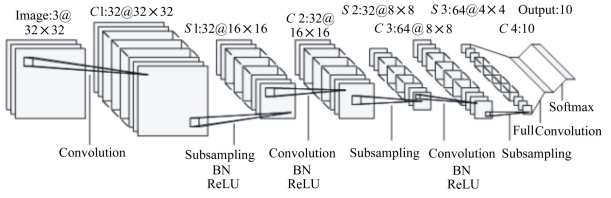

图 1　典型深度卷积神经网络模型

1.1　卷积层

卷积层也称为特征提取层,它由若干个局部滤波器(卷积核)组成,用于从输入特征图中提取各种不同的局部特征。在图像识别应用中进行卷积运算时,假设输入特征图的尺寸为 $W \times H \times C$,M 个作用于特征图上的卷积核可表示为 $K_x \times K_y \times C \times M$,则位于输出特征图 $f_o(f_o \in M)$ 的 (x,y) 位置上的神经元 $N_{x,y}^{f_o}$ 的计算公式如(1) 式所示。

$$N_{x,y}^{f_o} = \sum_{f_i=0}^{C-1} \sum_{j=0}^{K_y-1} \sum_{i=0}^{K_x-1} W_{i,j}^{f_i,f_o} \times N_{x\times S_x+i,y\times S_y+j}^{f_i} + B^{f_o} \quad (1)$$

式中:S_x 和 S_y 分别为 x 方向和 y 方向上进行卷积运算时的滑动步长;W^{f_i,f_o} 为权重参数;B^{f_o} 为偏置参数。从(1) 式可以看出卷积层的基本运算包括乘法和加法,这 2 种运算构成了处理单元的基本运算。

1.2　池化层

池化层也称为降采样层或数据合并层,它基于局部相关性原理降低空间维度,在减少计算量的同时,不仅保留了有用信息,而且有效地控制了深度神经网络过拟合的风险。在深度卷积神经网络中常用到的 2 种池化方式为最大池化和平均池化。当池化窗口为 $K_x \times K_y$ 时,则位于输出特征图 $f_o(f_o \in M)$ 的 (x,y) 位置上的神经元 $N_{x,y}^{f_o}$ 的最大池化和平均池化计算公式分别为(2) 式和(3) 式。

$$N_{x,y}^{f_o} = \max_{0\leqslant i\leqslant K_x-1,0\leqslant j\leqslant K_y-1} N_{x+i,y+j}^{f_i} \quad (2)$$

在进行池化计算时,输入特征图 f_i 和输出特征图 f_o 一一对应。对于最大池化,它是通过对池化窗口内的神经元进行连续比较从而得到最大的值来实现。

$$N_{x,y}^{f_o} = \frac{\sum_{i=0}^{K_x-1} \sum_{j=0}^{K_y-1} N_{x+i,y+j}^{f_i}}{K_x \times K_y} \quad (3)$$

对于平均池化的计算优化,由于神经网络的结构是确定的,所以可获取池化窗口的尺寸,将(3)式中的 $1/(K_x \times K_y)$ 看作系数乘以池化窗口内每个神经元,再做累加计算,这样可以消除除法运算,将其转化为乘法和加法这 2 种处理单元的基本运算,从而节省了硬件资源。

1.3　非线性激活函数层

在神经网络中加入非线性激活函数层的目的是通过分层的非线性映射复合使得整个网络具有非线性学习及表达能力,从而提升表征数据中高层语义的能力。常用到的激活函数包括:Sigmoid、Tanh 和 ReLU。其中 Sigmoid 和 Tanh 函数分别按照(4)式和(5)式进行计算。

$$\mathrm{Sigmoid}(x) = \frac{1}{1+\mathrm{e}^{-x}} \quad (4)$$

$$\mathrm{Tanh}(x) = \frac{\mathrm{e}^{x}-\mathrm{e}^{-x}}{\mathrm{e}^{x}+\mathrm{e}^{-x}} \quad (5)$$

在(4)式和(5)式中的 x 表示神经元值。从 Sigmoid 和 Tanh 的计算公式中可以看出它们都包含指数计

算,这在硬件实现中是非常困难的,所以当前普遍采用分段线性插值的近似计算方法,在本文设计的处理器中采用了一种基于分段线性插值的高效流水线硬件实现方式[1] 来实现这 2 种常用激活函数的计算。

$$\mathrm{ReLU}(x)=\begin{cases}x, x>0\\0, x\leqslant 0\end{cases} \tag{6}$$

ReLU 函数按照(6)式进行计算,其中 x 表示神经元值。由于在处理器设计中采用的是有符号定点数, ReLU 函数的计算仅通过判断符号位即可实现,为了便于计算将其直接设计在处理单元(PE)内,从而可直接输出 ReLU 计算后的神经元,从而节省了存取特征图的操作。

1.4 归一化层

为了避免深度卷积神经网络随着层级的加深而导致信息的传递呈现逐层衰减的趋势,在网络中加入了归一化层。在深度神经网络中常用的归一化操作包括:局部响应归一化(LRN)、局部对比归一化(LCN)以及批量归一化(BN)。

Krizhevsky 等人在 AlexNet[3] 中首先使用了局部响应归一化,使得识别精度取得了有限的提高。Du 等人在 ShiDiannao[10] 的设计中加入了局部响应归一化和局部对比归一化处理,虽然取得了识别精度上的提高,但是也增加了计算的复杂度。近来批量归一化在深度卷积神经网络中使用十分广泛,有效地加快了训练及收敛的速度,本文将采用此方法进行归一化处理。位于输出特征图 $f_o(f_o\in M)$ 的 (x,y) 位置上的神经元 $N_{x,y}^{f_o}$ 的批量归一化计算如(7)式所示。

$$N_{x,y}^{f_o}=\left(N_{x,y}^{f_i}-\frac{M(f_i)}{S}\right)\Big/\sqrt{\frac{V(f_i)}{S}+0.000\ 01} \tag{7}$$

(7)式其中 $M(f_i)$ 为输入特征图 f_i 的均值,S 为缩放因子,$V(f_i)$ 为输入特征图 f_i 的方差,这 3 个参数均为网络训练时学习到的参数。在采用 Caffe[12] 深度学习框架完成神经网络的训练后,将对获取的参数进行量化及预处理,从而得到更有利于处理器计算的参数。在对批量归一化参数进行处理时,对获取的参数按(8)式和(9)式计算。

$$BN\_a(f_i)=1\Big/\sqrt{\frac{V(f_i)}{S}+0.000\ 01} \tag{8}$$

$$BN\_b(f_i)=-\frac{M(f_i)}{S}\times BN\_a(f_i) \tag{9}$$

则根据(8)式和(9)式,批量归一化计算可转为(10)式所示计算。

$$N_{x,y}^{f_o}=BN\_a(f_i)\times N_{x,y}^{f_i}+BN\_b(f_i) \tag{10}$$

通过对参数的预处理和计算的优化使得原本复杂的批量归一化计算转化为乘法和加法这 2 种处理单元支持的基本运算,从而可以在一个时钟周期内完成单个神经元的批量归一化计算。

1.5 全连接层

在深度卷积神经网络的全连接层中,每个神经元都与其上一层的所有神经元连接,它用来将前边提取到的特征综合起来。则位于输出特征图 $f_o(f_o\in M)$ 的 (x,y) 位置上的神经元 $N_{x,y}^{f_o}$ 的全连接层的计算如(11)式所示。

$$N_{1,1}^{f_o}=\sum_{f_i}^{C-1}W_{1,1}^{f_i,f_o}\times N_{1,1}^{f_i}+B^{f_o} \tag{11}$$

式中,W^{f_i,f_o} 为权重参数,B^{f_o} 为偏置参数,全连接层的计算与卷积层的计算类似,由乘法和加法构成。

1.6 Softmax 输出层

Softmax 通常用于多分类神经网络的输出,可以看作是一种归一化指数函数,它将 N 维向量 V 映射为范围在(0,1)之间且累加和为 1 的 N 维向量 $\boldsymbol{S}$,其计算公式如(12)式所示。

$$\boldsymbol{S}_i=\frac{\mathrm{e}^{\boldsymbol{V}_i}}{\sum_{j=1}^{N}\mathrm{e}^{\boldsymbol{V}_j}}(\text{for } i=1,\cdots,N) \tag{12}$$

式中,$\boldsymbol{V}_i$ 和 $\boldsymbol{S}_i$ 都是 N 维向量。由于 Softmax 计算中也涉及指数运算,所以同样采用分段线性插值的方法进行近似计算,并且可同激活函数的计算采用相同的硬件来实现[1],从而节省硬件资源。

2 VLIW 处理器体系结构设计

2.1 总体结构设计

在当前研究中,源于技术和市场方面的考虑,所提出的深度学习加速器或协处理器[7-11] 仅支持算法推理阶段的加速就足够了[9]。因此,本文采用了线下训练获取参数的方式,在处理器的设计过程中仅考虑推理阶段深度卷积神经网络的算法加速。

本文设计的深度学习 VLIW 处理器体系结构如图 2 所示,包括了 I/O 控制器、指令寄存器、指令译码器、特征图缓存、参数缓存、缓存控制器、神经处理单元、激活函数流水线和输出模块。

在体系结构中,指令寄存器中用于存储接收到

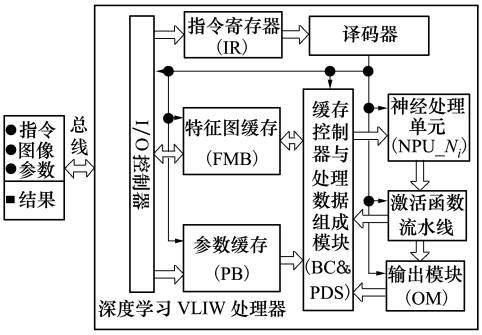

图 2　深度学习 VLIW 处理器体系结构

的 VLIW 指令,经译码器译码后将控制信号分别发送给各执行模块;I/O 控制器、缓存控制器、神经处理单元、激活函数流水线和输出模块构成了 VLIW 的具体执行模块。其中,I/O 控制器通过总线接收 VLIW 指令、图像数据和参数数据,并将处理结果通过总线返回。缓存控制器用于读写特征图缓存和读取参数缓存,并完成待处理神经元数据和参数的数据组织。神经处理单元作为处理器的计算核心,负责完成深度卷积神经网络模型定义的各层计算。激活函数流水线采用了基于分段线性插值计算方法的 5 段流水线设计,可高效完成 Sigmoid、Tanh 和 Softmax 函数的计算[1]。输出单元完成 Softmax 函数的最终计算以及深度卷积神经网络计算结果的输出。

2.2　神经处理单元(NPU)的设计

深度卷积神经网络具有计算密集的特性,神经处理单元作为深度学习处理器的计算核心,它的设计直接影响了处理器的性能。通过对深度卷积神经网络算法的分析,可知算法内在包含了突触并行、神经元并行和特征图并行 3 种粒度的并行计算。突触并行是指一次完成一个输出神经元的计算;神经元并行是指同时计算多个输出神经元;同时计算多个输出特征图则称为特征图并行。在当前研究中,Chen 等人在 DianNao[9] 的设计中采用了突触并行和神经元并行,但是受卷积核尺寸的影响,不适用于神经元较多的情况;Du 等人在 ShiDianNao[10] 的设计中采用了 2-D 处理单元实现神经元并行,适合图像识别应用,但是并行度有待提高;Google 的 TPU[11] 则采用了基于脉动阵列的突触并行和神经元并行,对片上存储器读写带宽需求较大,适用于服务器端数据处理。

本文在神经处理单元的设计中,针对图像识别应用,考虑到图像数据及计算中产生的特征图的三维特性,以及卷积核尺寸的多样化,处理单元的数量和读取缓存的带宽,选择了神经元并行和特征图并行方式。在单个神经处理单元内通过二维排列的处理单元(PE)实现特征图一个通道内的二维并行计算,通过多个神经处理单元组成处理器簇的设计来实现特征图的多个通道的并行计算。所设计的神经处理单元结构及其数据流图如图 3 所示。

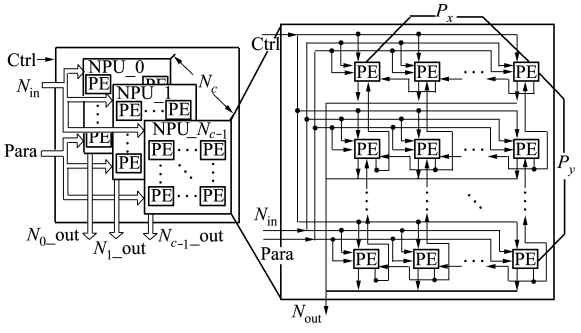

图 3　神经处理单元(NPU)结构及数据流图

在神经处理单元的结构设计中,考虑到卷积层的计算占据了整个网络模型计算的大部分,并且特征图并行时卷积数据的可共享的特点,设计了 N_c 个处理器簇(NPU_0,…NPU_N_{c-1}),可同时完成 N_c 个输出特征图的计算,相对于具有多个处理单元的单一处理阵列在处理较小特征图时具有优势;在每个处理器簇中设计了 $P_x \times P_y$ 个处理单元(PE)可同时计算 $P_x \times P_y$ 个输出神经元。处理单元(PE)可按行或列接收来自缓存控制器输入的神经元,也可接收神经处理单元内相邻 PE 传递的神经元,从而节省了读取特征图缓存的时间。PE 的设计通常包括,基于加法树[9]、基于乘累加器[10] 和基于脉动阵列[11] 等 3 种形式。通过对深度卷积网络模型各层计算的分析,采用了基于乘累加器的分时复用方式设计的处理单元如图 4 所示。

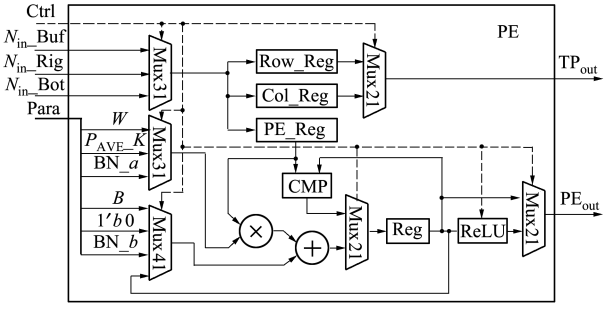

图 4　处理单元(PE)

在处理单元中,分为数据传输路径和计算路径。数据传输路径通过行寄存器(Row_Reg)和列寄存器(Col_Reg)来暂存读入的神经元,当滑动卷积窗口时通过局部传输端口 TPout 将暂存的神经元传入相邻的神经元,从而减少对片上缓存的访问。计算路径分为用于卷积计算、全连接计算、平均池化、归一化、ReLU 计算的乘累加运算路径,和用于最大池化的比较运算路径。在乘累加运算路径中,为了表明可实现的分层计算,设计了 Mux31 和 Mux41 选择器,实际中为 2 个端口,每层参数提前存入参数缓存中,在计算时通过缓存控制器分别读入相应端口完成计算,达到分时复用的效果。

在处理器的计算中,神经元和参数均采用了 16 位定点数,在保证误差可忽略的前提下相对于浮点数节省了存储和计算单元的面积。同时,为了避免处理单元中连续的乘累加计算带来的数据"膨胀",处理单元设计中采用了截断乘法器。

2.3 片上多级存储及缓存控制器设计

在图像识别应用中,神经处理单元的高效计算离不开数据的及时送达和有效组织。同时,深度卷积神经网络具有存储密集的特性,为了获得高效的读取和有效的数据组织,特征图和参数的存储方式也是非常关键的。本文采用了如图 5 所示的片上多级层次存储结构设计的缓存来存储特征图和参数。

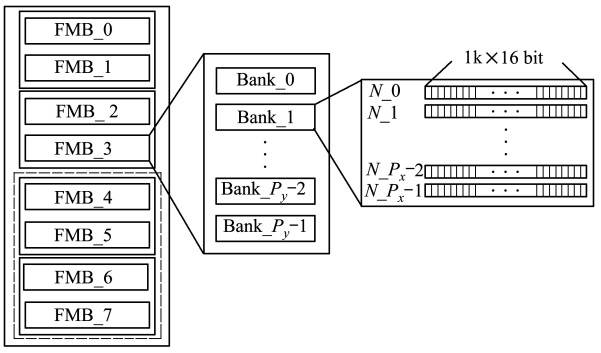

图 5 片上多级层次存储结构

基于 NPU 中 $P_x \times P_y$ 处理单元阵列的设计,在特征图缓存(FMB)中包含 8 个特征图缓存块(FMB_i, i=0,1,..7)。每 2 个特征图缓存块组成一组同向缓存,例如 FMB_0 和 FMB_1 构成输入特征图缓存,交替存储输入特征图,从而形成乒乓存储器,FMB_2 和 FMB_3 此时构成输出特征图缓存,需要说明的是特征图的输入与输出特性不是固定的,它们交替使用,采用状态位表征输入/输出特性。为了适应不断增加的深度神经网络算法规模,设计了 4 个特征图缓存块(FMB_4~FMB_7)用于在大规模深度卷积神经网络中与另外 4 个缓存块(FMB_0~FMB_3)交替存储输入/输出特征图。每个缓存块中包含 P_y 个 Bank,每个 Bank 采用 P_x 个 1 k×16 bit 的 SRAM 设计,每个存储单元对应一个神经元。参数缓存(PB)的组织结构与特征图缓存相似,在此不再赘述。

缓存控制器用于在神经处理单元计算时读取神经元和参数值,经过数据组织后送入神经处理单元(NPU);在 NPU 处理完成后,将输出神经元写入特征图缓存中。为了有效读出数据并进行组织,缓存控制器对特征图缓存和参数缓存的读操作包括 4 种方式:①读取 P_y 个 Bank,每个 Bank 读取 P_x 个神经元;② 读取一个 Bank 的 P_x 个神经元;③ 从 P_y 个 Bank 中以固定步长 Stride 读取神经元;④ 读取 1 个神经元。为了有效进行数据组织,在缓存控制器中分别设计了一个 $P_x \times P_y \times 16$ bit 的神经元寄存器阵列和一个 $P_x \times P_y \times 32$ bit 的参数寄存器阵列,对应于 $P_x \times P_y$ 的处理单元阵列,用于处理数据的建立组织。在神经处理单元完成计算后,缓存控制器将输出神经元交替存储到一组输出特性的特征图缓存中,用作下一层计算的输入。

2.4 控制与 VLIW 指令

由于深度卷积神经网络的各层计算之间具有相关性,并且每一层的计算中又包含多个计算步骤,同时每个计算步骤都需要多个功能模块协同工作。因此,本文采用了基于超长指令字(VLIW)的指令级并行方式进行设计。

在进行处理器控制及指令系统的设计时,采用了由主处理器发送控制指令,并在深度学习处理器内部设置指令寄存器的方式[11],相对于将指令事先存储在片上指令存储器中的方式[9-10],不仅省去了取指令的逻辑,而且不需要大容量的片上指令存储器,仅增加了传输指令时与主处理器交互的带宽。

深度学习处理器的 VLIW 指令格式的设计,存在 2 种方式:①如图 6 所示,直接从网络模型中抽取出指令的抽象方式。这种方式可有效减少指令的数量,但是使得译码和控制逻辑变得复杂;②如图 7 所示,将网络模型映射到处理器各相关单元的具体方式。这种方式实现容易,但是实现整个深度学习网络算法的指令数较多。

指令码 4 bit	输入特征图 31 bit				卷积核 20 bit					输出特征图 13 bit	
Coonv (0010)	FMB_in 起始地址 3 bit	宽(W) 9 bit	高(H) 9 bit	通道(C) 10 bit	K_x 3 bit	K_y 3 bit	K_c 10 bit	Stride 2 bit	Padding 2 bit	输出通道 10 bit	FMB_out 起始地址 3 bit

图 6　网络模型中抽取的卷积操作指令格式

在本文的设计中,考虑到处理器硬件实现的便利性及复杂度,将网络模型映射到深度学习处理器 VLIW 指令的过程以软件编译器的方式实现,通过编译器可得到处理器执行 VLIW 指令序列,从而简化了片上译码和控制逻辑。所设计的 VLIW 指令宽度为 75 位,以使用频率最高的卷积运算指令为例,它所对应的 VLIW 指令格式如图 7 所示。

指令码 4 bit	特征图缓存 (FMB) 30 bit					参数缓存 (PB) 23 bit			
Conv (0010)	OP 1 bit	起始地址 20 bit	读模式 2 bit	步长 1 bit	长度 6 bit	OP 1 bit	起始地址 20 bit	读模式 2 bit	
	-0读 -1:写		-00:读多个 Bank -01:读一个 Bank -10:按步长读多个 Bank -11:读一个神经元	-0:步长=1 -1:步长=2		-0读 -1:写		-00:读多个 Bank -01:读一个 Bank -10:按步长读多个 Bank -11:读一个神经元	

缓存控制器 (BC) 2 bit		神经处理单元(NPU_N_i) 16 bit					
数据组织模式 1 bit	参数组织模式 1 bit	PE使能 9 bit	N_{in}选择 2 bit	写Reg选择 2 bit	$Pool_{Max}$使能 1 bit	累加使能 1 bit	ReLU使能 1 bit
-0:NPU 内广播 -1:NPU 间广播			-00:读外部 -01:行传递 -10:列传递	-00:写行 Reg -01:写列 Reg -10:行列全写	0:无效 1:无效		

图 7　卷积操作的 VLIW 指令格式

3　原型系统测试与分析

3.1　原型测试系统的设计与实现

为了验证深度学习 VLIW 处理器的系统方案和实现算法,采用 Verilog HDL 按照体系结构设计分模块进行了硬件实现,并采用 SystemVerilog 基于 UVM 在 Mentor 公司的 QuestaSim 软件上搭建了硬件验证平台对其进行了仿真验证。对经过充分仿真验证的深度学习 VLIW 处理器,选择 Xilinx 公司的 XC6VLX240T 进行原型测试平台开发,总体结构如图 8 所示。在原型系统中,Micro Blaze 在键盘输入的命令控制下将 VLIW 指令序列送入深度学习 VLIW 处理器中。深度学习 VLIW 处理器根据接收到的指令,首先读取存储在 FLASH 中的参数数据和测试集图像数据;然后进行算法加速计算;最后将结果存入缓冲区。输出显示控制器从缓冲区中读出数据后发给显示器进行显示输出。

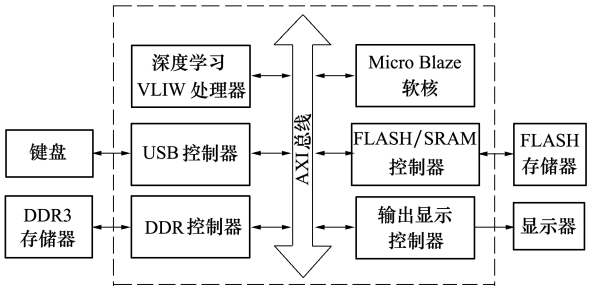

图 8　深度学习 VLIW 处理器的原型系统框图

在采用 ISE13.4 开发软件进行深度学习 VLIW 处理器硬件实现过程中,根据选择 FPGA 资源情况,设计了 8 个 NPU 处理器簇,每个处理器簇包含 8×8 个 PE,共 512 个 PE。特征图缓存包含 8 个缓存块,每个缓存块包含 8 个 Bank,每个 Bank 为 8 个 1 k×16 bit 的 SRAM,特征图缓存容量为 512 k×16 bit,参数缓存容量为 128 k×16 bit。在逻辑实现中,乘法器采用了 FPGA 中的 DSP48E1s 资源;特征图缓存、参数缓存以及指令缓存等采用了片上 BRAM 实现;深度学习 VLIW 处理器的 FPGA 主要资源利用率如表 1 所示,经过布局布线后,工作频率达到 200 MHz。

表 1　深度学习 VLIW 处理器的 FPGA 资源利用率

Slice Logic Utilization	Used/Available	Utilization/%
Slice Registers	33,152/301,440	11
Slice LUTs	105,578/150,720	70
RAMB18E1/FIFO18E1	709/832	85
DSP48E1s	596/768	77

3.2　训练与测试

1) 训练:本文设计的深度学习 VLIW 处理器采用线下训练方式获取参数。训练过程中采用当前业界流行的深度学习框架 Caffe[12],硬件环境包括了 CPU(Core i7,6700HQ)和 GPU(GTX960M)。测试基准采用了网络结构修改过的 LeNet-5[1-2] 和 AlexNet[1,3],MobileNet[5] 和 SSD300+MobileNet[4-5] 等深度卷积神经网络模型,训练及测试数据集分别采用了 MNIST[2], CIFAR-10[13], Stanford Dogs[14], PASCAL VOC2007[15] 和 VOC2012[15] 等。训练完成后从得到的 Caffemodel 模型中提取神经网络的参数,经过预处理后用于处理器的计算。将要部署的深度神经网络模型 Prototxt 文件通过软件编译器映射到处理器,产生处理器运行的 VLIW 指令序列。

2）测试：为了充分测试处理器的功能及性能，分别设计了针对不同网络结构和测试数据集的图像分类和目标检测试验，并在测试前，采用软件方式对测试数据集分别进行了归一化处理。

在图像分类测试中，采用了 LeNet-5[1-2]、AlexNet[1,3] 和 MobileNet[5] 作为测试基准，分别在 MNIST[2]、CIFAR-10[13] 和 Stanford Dogs[14] 数据集上进行测试，取得的测试精度与软件框架 Caffe[12] 测试的精度对比如表 2 所示。

表 2 图像分类试验测试精度对比

网络模型	测试集	Caffe 测试精度	处理器测试精度
LeNet-5	MNIST	0.990 8	0.989 1
AlexNet	CIFAR10	0.730 2	0.728 9
MobileNet-224	Stanford Dogs	0.811 3	0.808 7
0.5MobileNet-224	Stanford Dogs	0.752 9	0.750 2

在图像分类试验过程中，通过 Caffe[12] 的计时功能测得了测试基准在相应数据集上处理一副图像分别采用硬件环境中 CPU 和 GPU 所占用的时间，并通过仿真获得了深度学习 VLIW 处理器的运行时间，其对比如表 3 所示。

表 3 图像分类试验每幅图片运行时间对比

测试基准	CPU/ms	GPU/ms	深度学习 VLIW 处理器/μs
LeNet-5 MNIST	41.015	1.521	39.58
AlexNet CIFAR-10	249.077	6.499	126.58
MobileNet-224 Stanford Dogs	88.137	48.835	6.896×10^3
0.5MobileNet-224 Stanford Dogs	42.964	27.356	1.811×10^3

在目标检测试验中，采用了 SSD300[4] 网络作为框架，MobileNet[5] 作为主干网络进行测试。通过在 PASCAL VOC2007[15] 和 VOC2012[15] 训练验证数据集上训练得到的平均精度（mAP）为 0.682，深度学习 VLIW 处理器实际测试取得的 mAP 为 0.676。与此同时，为了检验图像目标检测试验的效果，采用 Python 语言基于 OpenCV 库编写了输出显示程序，将处理器运行输出的图像类别、类别得分、坐标信息标注在处理的图像上，图像目标检测的效果如图 9 所示。

图 9 基于 MobileNet+SSD 的图像目标检测试验效果

采用 MobileNet[5] +SSD300[4] 算法在本文设计的深度学习 VLIW 处理器上进行图像目标检测时，输入图像尺寸为 300×300 像素，处理速度可以达到 45 帧/秒。对于当前先进的 WorldView-3 卫星[16]，它的分辨率为 0.31 m，每天可覆盖 680 000 km^2，仅需要 22 个本文设计的深度学习 VLIW 处理器即可实现卫星遥感图像的实时目标检测。

在图像分类及目标检测试验中，原型系统中所实现的深度学习 VLIW 处理器每个时钟周期可完成 512 次 16 位定点乘加运算，同时加上缓存控制器读取 8×8 的输入特征图以及将 8×8 的输出特征图写出的操作，在工作频率为 200 MHz 时，峰值性能可以达到 128 GOP/s。与当前研究中的不同深度学习处理器的性能对比如表 4 所示。

表 4 深度学习处理器性能对比

深度学习处理器	频率/MHz	性能/（GOP·s^{-1}）	实现方式
文献[17]	100	61.62	FPGA Virtex-7 VX485T
NeuFlow[7]	200	120	FPGA Virtex-6 ML605
本文	200	128	FPGA Virtex-6 XC6VLX240T
ShiDianNao[10]	1 000	194	ASIC 65nm CMOS

3.3 测试结果分析

由表 2 所示的图像分类试验数据和图 8 所示的目标检测试验效果,可以看出部署在本文设计的深度学习 VLIW 处理器上的多种深度学习网络模型在相应的测试数据集上可有效完成图像分类和目标检测任务。处理器设计中采用 16 位定点数得到的测试精度接近软件框架中采用浮点数得到的测试精度,并且平均误差不超过 1%。在处理性能方面,由表 3 可以看出基于图像分类试验的测试基准,本文所设计的深度学习 VLIW 处理器的计算速度至少是 CPU 的 12 倍,是 GPU 的 7 倍。通过表 4 对当前研究中的不同深度学习处理器性能进行比较可以看出,以 FPGA 方式实现时,在片上计算资源相近的情况下,本文设计的深度学习 VLIW 处理器性能优于其他设计,但是相对于以 ASIC 方式实现的寒武纪 ShiDianNao[10] 在工作频率及处理性能上都不及对方,这不仅反映了以 ASIC 实现深度学习处理器的优势,也将是本文的下一步工作方向。

4 结　论

本文通过对深度卷积神经网络模型各计算层的优化,设计了一款可扩展的多处理器簇的深度学习 VLIW 处理器,并基于 Xilinx 公司的 FPGA 搭建了原型验证系统。经过测试,该处理器可有效完成图像分类和目标检测等图像识别应用,在多个测试基准环境下,性能优于 CPU 和 GPU,与软件框架相比的测试误差可忽略不计。本文设计的可扩展多处理器簇的深度学习 VLIW 处理器体系结构为实现航空航天领域图像采集端的本地化高效处理提供了硬件基础,实际应用中可将选定的深度学习算法部署在处理器上实现高效处理。

参考文献:

[1] LI L, ZHANG S, WU J. An Efficient Hardware Architecture for Activation Function in Deep Learning Processor[C]//IEEE International Conference on Image, Vision and Computing, 2018: 911-918

[2] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-Based Learning Applied to Document Recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324

[3] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Image Net Classification with Deep Convolutional Neural Networks[C]//International Conference on Neural Information Processing Systems, 2012: 1097-1105

[4] LIU W, ANGUELOV D, ERHAN D, et al. SSD: Single Shot MultiBox Detector[C]//European Conference on Computer Vision, 2016: 21-37

[5] HOWARD A G, ZHU M, CHEN B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [EB/OL].(2017-04-17)[2019-01-02]. https://arxiv.org/abs/1704.04861

[6] HENNESSY J L, PATTERSON D A. Computer Architecture: a Quantitative Approach[M]. 6th Edition. Cambridge: Morgan Kaufmann Publishers Inc, 2018

[7] FARABET C, MARTINI B, CORDA B, et al. NeuFlow: a Runtime Reconfigurable Dataflow Processor for Vision[C]//Computer Vision and Pattern Recognition Workshops, 2011: 109-116

[8] PEEMEN M, SETIO A A A, MESMAN B, et al. Memory-Centric Accelerator Design for Convolutional Neural Networks[C]//IEEE International Conference on Computer Design, 2013: 13-19

[9] CHEN T, DU Z, SUN N, et al. DianNao: a Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning[J]. Acm Sigplan Notices, 2014, 49(4): 269-284

[10] DU Z, FASTHUBER R, CHEN T, et al. Shidiannao: Shifting Vision Processing Closer to the Sensor[C]//International Symposium on Computer Architecture, 2015: 92-104

[11] JOUPPI N, YOUNG C, PATIL N, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit[C]//International Symposium on Computer Architecture, 2017: 1-12

[12] JIA Y, SHELHAMER E, DONAHUE J, et al. Caffe: Convolutional Architecture for Fast Feature Embedding[C]//ACM International Conference on Multimedia, 2014: 675-678

[13] KRIZHEVSKY A, HINTON G. Learning Multiple Layers of Features from Tiny Images[R]. Technical Report TR-2009

[14] KHOSLA A, JAYADEVAPRAKASH N, YAO B, et al. Novel Dataset for Fine-Grained Image Categorization[C]//First Work-

shop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition Colorado Springs, 2011

[15] EVERINGHAM M, ESLAMI S, VAN G, et al. The Pascal Visual Object Classes Challenge: A Retrospective[J]. International Journal of Computer Vision, 2015, 111(1): 98-136

[16] Satellite Imaging Corporation. WorldView-3 Satellite Sensor [EB/OL]. (2018-02-06) [2019-01-02]. https://www.satimagingcorp.com/satellite-sensors/worldview-3/

[17] ZHANG C, LI P, SUN G, et al. Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks[C]// ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2015: 161-170

Design of Deep Learning VLIW Processor for Image Recognition

LI Lin[1], ZHANG Shengbing[1], WU Juan[2]

(1.School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China;
2.Fourth Design Department, Beijing Institute of Micoelectronics Technology, Beijing 100076, China;
3.School of Animation and Software, Xi'an Vocational and Technical College, Xi'an 710077, China)

Abstract: In order to adapt the application demands of high resolution images recognition and efficient processing of localization in aviation and aerospace fields, and to solve the problem of insufficient parallelism in existing researches, an extensible multiprocessor cluster deep learning processor architecture based on VLIW is designed by optimizing the computation of each layer of deep convolutional neural network model. Parallel processing of feature maps and neurons, instruction level parallelism based on very long instruction word (VLIW), data level parallelism of multiprocessor clusters and pipeline technologies are adopted in the design. The test results based on FPGA prototype system show that the processor can effectively complete the image classification and object detection applications. The peak performance of processor is up to 128 GOP/s when it operates at 200 MHz. For selecting benchmarks, the processor speed is about 12X faster than CPU and 7X faster than GPU at least. Comparing with the results of the software framework, the average error of the test accuracy of the processor is less than 1%.

Keywords: image recognition; deep learning; convolutional neural networks; very long instruction word(VLIW); processor; extensible