高技术通讯 2024 年 第34 卷 第7 期:744-754

doi:10.3772/j.issn.1002-0470.2024.07.008

基于片上系统的可配置卷积神经网络加速器的设计与实现①

张立国② 杨红光③ 金 梅 申 前

(燕山大学电气工程学院 秦皇岛 066004)

摘 要 针对现阶段卷积神经网络(CNN)加速器的设计只能部署在单一现场可编程门阵列(FPGA)平台、不支持硬件平台升级迭代的问题,设计了一种基于片上系统(SoC)的可配置 CNN 加速器。该加速器具备以下2 个特点:(1)在电路设计中将数据位宽、中间缓存空间大小、乘法器阵列(MAC)并行度作为一种可选配置参数,通过调整资源使用量,使得该加速器能够适配不同 FPGA 硬件;(2)提出了动态数据复用的策略,通过对比数据传输过程中不同复用方式下的总参数量差异,动态地选择复用方法,以减少数据传输的等待时间,提高乘法器阵列利用率。该方案在 ZCU104 板卡上进行了实验,实验结果表明,当数据位宽选择 8、乘法器阵列并行度选择 1 024、核心运算模块工作在 180 MHz 时,卷积运算阵列峰值吞吐量为 180 GOPs,功耗为 3.75 W,能效比达到 47.97 GOPs · W^{-1},对于 VGG16 网络,其卷积层的平均乘法器阵列利用率达到 84.37%。

关键词 卷积神经网络(CNN);现场可编程门阵列(FPGA);CNN 加速器;可配置;异构加速

随着人工智能和大数据技术的不断发展,卷积神经网络(convolutional neural network, CNN)在多个应用领域中获得了突破性进展,并被广泛应用于图像、音频、视频[1]等领域的识别和分类任务[2]中。然而,由于卷积神经网络需要大量的计算资源,在移动场景中使用中央处理器(central processing unit, CPU)和图形处理器(graphics processing unit, GPU)等通用计算设备来实现卷积计算能耗太大。为了解决这一问题,CNN 加速器应运而生。CNN 加速器是一种专门用于加速卷积神经网络运算的硬件设备,具有并行处理、低功耗和高能效等特点。因此,CNN 加速器对于加速深度学习模型的训练和推理具有非常重要的意义。

2012 年,AlexNet[3] 在大规模视觉识别批战赛(ImageNet Large Scale Visual Recognition Challenge, ILSVRC)分类任务比赛中以超过第 2 名 10.9% 的优势夺得冠军,拉开了卷积神经网络(CNN)在图像识别领域统治地位的序幕。自此以后,卷积神经网络结构不断演化,2014 年的 VGG(Visual Geometry Group)网络[4]与 AlexNet 的网络结构类似,但是它采用了更深的网络层数。2015 年提出的 ResNet[5] 通过引入残差网络块结构,避免了梯度消失、梯度爆炸和退化问题,同时可以加深网络层数,提高了识别精度。

随着神经网络的不断发展,模型的复杂性和数据参数量不断增加。为了便于在移动场景用应用卷积神经网络,相关人员主要从以下 2 个方面开展研究。一方面是采用轻量化网络模型,例如,MobileNet[6] 模型采用了深度可分离卷积来减小计算量和参数数量;ShuffleNet[7] 模型主要采用了组卷积和通道随机重排的方法来减小模型参数数量和计算量。另一方面是寻找一种计算能力更强、并行度更

① 国家重点研发计划(2020YFB1711001)资助项目。
② 男,1978 生,博士,副教授;研究方向:机器视觉,故障诊断,虚拟现实;E-mail: zlgtime@163.com。
③ 通信作者,E-mail: 1763066586@qq.com。
(收稿日期:2023-03-21)

高、数据吞吐更快、更小型化的高性能硬件。鉴于现场可编程门阵列(field programmable gate array, FPGA)在加速数据处理和并行计算等方面具有突出表现,因此可将神经网络模型移植到 FPGA 中,通过卷积神经网络加速器完成移动场景中的推理工作。

基于 FPGA 的卷积神经网络加速器的研究主要集中在 2 个方向:首先是提升乘累加(multiply accumulate,MAC)阵列并行度,如文献[8]设计的高效卷积计算引擎中通过 3 种并行策略提高了运算单元的利用率;其次是数据传输和复用,如文献[9]提出用于连续片外内存访问的非重叠平铺方法,减少了数据传输的时间。除此之外,还提出了多种新型智能加速器架构[10-11],但普遍存在以下问题:(1)通常设计为固定的计算模式,电路设计不可调整,即只能适用于单种硬件平台,无法实现后期硬件平台的升级;(2)存储系统通常采用固定的设计,对于数据传输过程中出现的数据复用问题,缺少灵活的复用策略。

为改善这种情况,本文设计了一种基于片上系统(system on chip,SoC)的可配置 CNN 加速器,可以适用于多种 FPGA 硬件资源。首先,为减少数据从外部存储器传输到片上缓存空间的总时间,本文提出了一种动态数据复用的策略,通过在每层卷积运算前对比不同复用方式的总参数量,自适应选取最优方式;其次,为便于对 CNN 加速器后期的升级迭代,本文在硬件电路设计中将数据位宽、中间缓存空间大小、乘法器阵列并行度等作为一种可选配置参数,并将相关配置参数封装到配置文件中,通过修改配置方式实现对硬件占用资源的调整。

1 卷积神经网络原理

卷积神经网络主要由卷积层、池化层、全连接层和激活函数等组成,本节将对卷积神经网络的各个基础算子进行简单介绍。

作为卷积神经网络的核心,卷积层通过卷积操作对输入数据进行特征提取,从而实现图像分类、物体识别等任务。在卷积层中,通过设置一组可学习的卷积核,对输入的数据进行卷积操作,得到一组新的特征图,这些特征图可以被看作是对输入数据的不同抽象程度的表示。图 1 所示为单通道时卷积运算过程,首先对输入特征图进行补零操作,然后通过卷积核做卷积运算,其卷积结果为输出特征图。在卷积运算中常用的卷积核大小为 3×3、5×5 和 7×7。

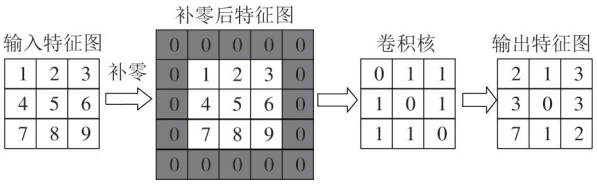


图 1 卷积运算示意图

池化层是一种常见的用于减少卷积层输出维度的操作,通常紧跟在卷积层后面。其池化类型包括最大池化和平均池化 2 种,池化层的主要作用是减少特征图的尺寸,从而减少神经网络的参数数量和计算量。图 2 所示为池化核为 2×2 时的池化运算示意图。

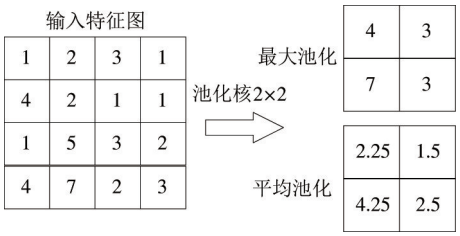


图 2 池化运算方式示意图

全连接层主要作用是将卷积层和池化层的输出映射到指定的输出大小,以生成最终的分类或预测结果。通常,全连接层位于卷积层和池化层的后面,用于对卷积特征进行分类。

激活函数定义了上一层神经元输出与下一层神经元输入之间的函数关系,通常用于卷积层或全连接层后对输入信号进行非线性映射。

2 加速器优化策略

2.1 动态数据复用方式

卷积运算过程是指先将输出特征数据矩阵上各值映射到原特征矩阵中的起始数据位置,然后遍历卷积核范围内的输入特征数据,并与权重数据进行

乘累加运算的过程。当输入特征图为 x，其输入通道为 ch_in，权重卷积核宽度为 kx，高度为 ky，行步进为 Sx，列步进为 Sy，方向补零为 Pad_left，列方向补零为 Pad_up，输出特征图为 out，其宽度为 w_out，高度为 h_out，输出通道为 ch_out，则卷积运算伪代码计算过程如图 3 所示。

```
for(k=0;k<ch_out;k=k+1)
  for(h=0;h<h_out;h=h+1)
    for(w=0;w<w_out;w=w+1)
     {
       sum=0;
      for(c=0;c<ch_in;c=c+1)
        for(r=0;r<ky;r=r+1)
         for(s=0;s<kx;s=s+1)
           {
           data_now=x[h×Sy-Pad_up+r][w×Sx-Pad_left+s][c];
           wt_now=wt[r][s][c][k];
           sum=sum+data_now×wt_now;
           }
      out[h][w][k]=sum;
     }
```

图 3　常规卷积运算伪代码

从伪代码中可以看出，正常卷积运算中乘法运算量 mac_sum 为

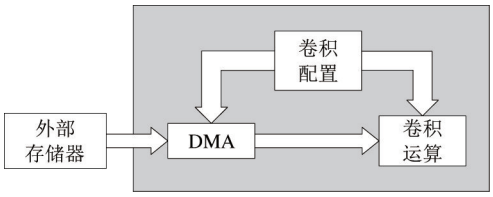
$$mac\_sum = kx \times ky \times ch\_in \times ch\_out \times w\_out \times h\_out \quad (1)$$

由于每次乘法运算需要一个特征数据和一个权重数据，所以传统卷积运算方式中，总数据传输参数量 size_sum 为乘累加运算次数的 2 倍，如式(2)所示。

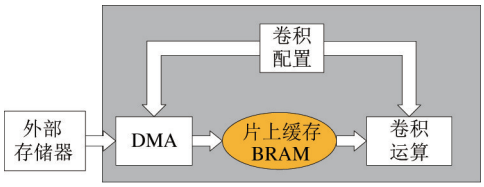
$$size\_sum = mac\_sum \times 2 \quad (2)$$

在 FPGA 中执行卷积层运算时，数据参数量庞大，导致数据从外部存储器到运算模块需要消耗较长时间。当数据带宽无法满足乘累加阵列需求时，会使得乘累加阵列暂时空闲，从而无法获得较好的加速器性能。针对这一问题，本文在硬件设计时优化了电路结构，在外部存储器和运算模块间加入了中间缓存空间。在 FPGA 中执行卷积运算时，先将数据从外部存储器中读取并通过直接内存访问(direct memory access，DMA)搬运到片上缓存空间随机块存储器(block random access memory，BRAM)，再从片上缓存空间直接读取数据进行乘累加运算，如图 4 所示。

相比于直接从外部存储器中读取数据，加入中间缓存空间后，由于数据在片内的传输速度更快，减少了数据传输的等待时间，从而提高了卷积阵列运算效率。然而，FPGA 片上存储资源有限，当卷积层输入参数量较大，无法一次性将所需输入权重数据和输入特征数据存放到片上缓存空间时，需要将权重数据和特征数据进行分块，即将一个大卷积拆分成多个小卷积。对于卷积运算有以下 4 种分块方式，如表 1 所示。



(a) 直接数据传输方式



(b) 优化后数据传输方式

图 4　卷积运算模块电路架构

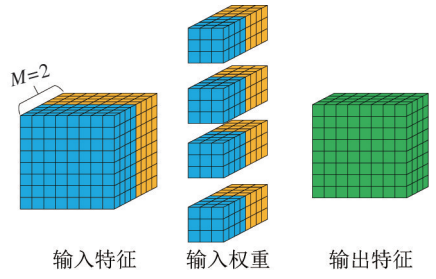
表 1　数据分块方式

分块方式	描述
方式 1	按输入通道方向分块
方式 2	按输出通道方向分块
方式 3	按输入特征高度方向分块
方式 4	按输入特征宽度方向分块

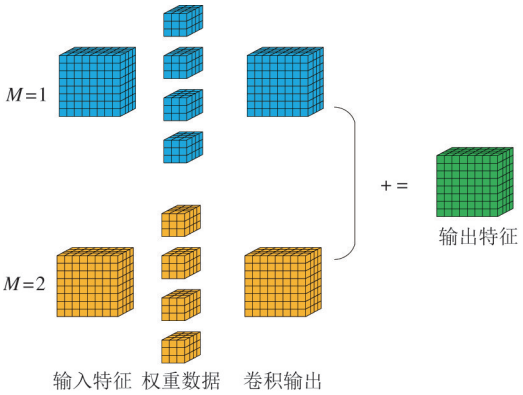
分块方式 1 将输入特征数据按通道方向分为 M 份，如图 5(a)所示。由于在卷积运算中特征数据和权重数据是相互对应的，所以相应的权重数据也随之分割。这种分块方式的优点是输入特征数据、权重数据所需存储空间均减少，但因为在卷积运算中需要将上一次小卷积运算结果与本次小卷积运算结果相加，所以需要缓存上一次计算结果，如图 5(b)所示，这个过程会增加过多的缓存带宽。

分块方式 2 按照输出通道方向将权重数据分为 N 份，如图 6 所示。由于权重和输出通道相互对应，所以权重数据也相应分为 N 份，这种做法的好处是权重所需空间减少，但输入特征数据量不变。

分块方式3将输入特征图按高度方向分为K



(a) 分块方式



(b) 分块运算

图 5　分块方式 1 示意图

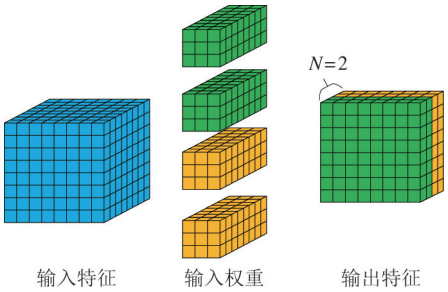
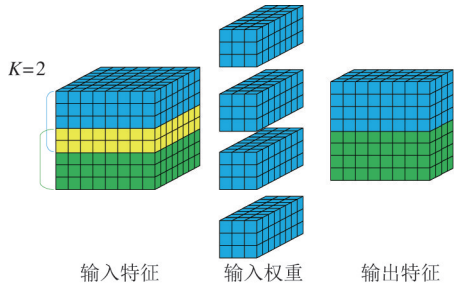


图 6　分块方式 2 示意图

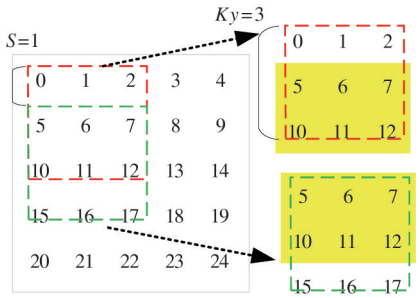
份,如图 7(a)所示。这种划分优点是输入特征所需存储空间会减少,但相邻输入特征块之间会存在重叠区域。这是由于在卷积运算中,当卷积核尺寸为 Ky、步进值为 S 时,在高度方向上会存在 $Ky - S$ 的重叠数据,如图 7(b)所示。

分块方式 4 是将输入特征图按宽度方向分块,如图 8 所示。与方式 3 相似,这种划分优点是输入特征所需存储空间会减少,但相邻输入特征块之间会存在重叠区域。

对比分析 4 种分块方法,由于方式 1 按照输入通道分块以后,产生过多的额外缓存带宽,造成较多的资源占用,而其他分块方式又无法同时将特征数



(a) 分块方式



(b) 数据重复使用

图 7　分块方式 3 示意图

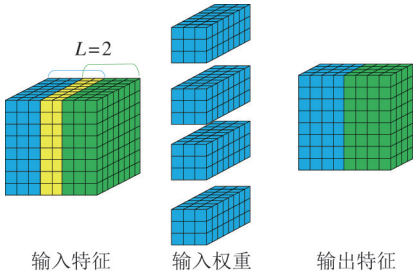


图 8　分块方式 4 示意图

据和权重数据进行分块。针对这一问题,本文采用了分块方式融合的方法,同时在多个方向分割,考虑到方式 3 和方式 4 类似,都是对输入特征数据分块,所以选用一种即可。最终本文的分块方法如图 9 所示,在输入高度方向上分为 K 份、输出通道上分为 N

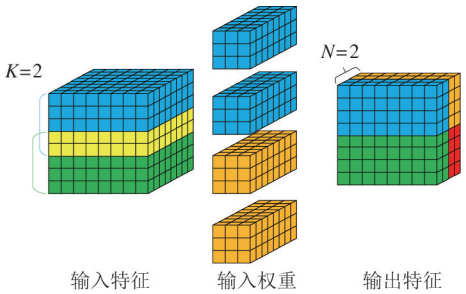


图 9　本文分块示意图

份，这种分块方式的好处是同时对权重数据和特征数据进行分块，从而便于将数据从外部存储器中读出并暂存到片上缓存空间中。

经以上分块方式，可以将一个大卷积分成 $K \times N$ 次小卷积。但是这种分块方式伴随着数据的重复使用，如图 10 所示，从图中可以看出在 4 次小卷积运算中存在特征数据块和权重数据块被重复使用的情况。

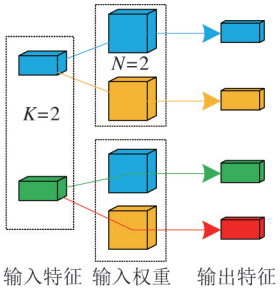


图 10　分块卷积示意图

针对该情况，可采用数据复用的优化方式，包括权重复用和数据复用。权重数据复用是在卷积运算中先保持权重数据不变，遍历输入特征数据；特征数据复用是在卷积运算中保持输入特征数据不变，遍历权重数据。

当选用权重复用的方式时，数据从外部存储器到片上缓存的总传输量 $Size_1$ 可用式(3)表示，其中权重数据不变，特征数据被遍历了 N 次。

$$Size_1 = [A + overlap] \times N + B \tag{3}$$

式中，A 为输入特征总大小，B 为输入权重总大小，$overlap$ 为一次小卷积运算中特征数据重叠部分，即图 7 中重叠部分。则在小卷积运算中，被重复使用的特征数据总参数量 $overlap$ 可用式(4)来表示。

$$overlap = \frac{A}{Hin} \times (Ky - S) \times (K - 1) \tag{4}$$

式中，Hin 为输入特征高度，Ky 为卷积运算中卷积核尺寸，S 为步进值。

当选用特征复用的方式时，从外部存储器到片上缓存空间的总传输参数量 $Size_2$ 可用式(5)表示。

$$Size_2 = K \times B + A + overlap \tag{5}$$

为分析不同复用方式的差异，计算当卷积核尺寸为 3、步进为 1 时，不同复用方式下总传输数据量，结果如表 2 所示。

表 2　不同复用方式下总传输数据量

	卷积 1	卷积 2	卷积 3
输入尺寸	(224,224)	(112,112)	(56,56)
输入通道	3	16	128
输出通道	64	128	256
(K, N)	(2,2)	(4,1)	(3,1)
传统方式	86 704 128	924 844 032	924 844 032
权重复用	305 472	919 552	724 992
特征复用	155 328	1 140 736	1 314 816
本文	155 328	919 552	724 992

从表中可以看出权重复用方式和特征复用方式下总传输数据量不同，且不同分块方式下最优复用方式不同。因此本文提出了一种数据动态复用的方式，在每层卷积运算前对比 2 种复用方式，自适应选择最优方式，以减少数据传输过程中乘累加阵列的等待时间，提高乘法器阵列(MAC)利用率。

2.2　可配置硬件电路

常规卷积运算伪代码如图 1 所示，其运算过程不涉及并行运算，无法发挥 FPGA 并行运算的优势，对此本文对卷积计算过程进行优化，设计了一种并行度可配置的卷积运算阵列，其卷积伪代码如图 11 所示。

```
for(k=0;k<ch_out;k=k+Tk)
  for(pp=0; pp<h_out*w_out; pp=pp+Tk)
    {
      sum=0;
      for(c=0;c<ch_in;c=c+Tk)
        for(r=0;r<ky;r=r+1)
          for(s=0;s<kx;s=s+1)
            {
              if (pp<h_out*w_out)
                {
                  h=pp/w_out;
                  w=pp %w_out;
                  for(kk=k,kk<k+Tk,kk=kk+1)
                    for(cc=c,cc<c+Tk,cc=cc+1)    并行展开计算
                      {
                      data_now=x[h*Sy-Pad_up+r][w*Sx-Pad_left+s][cc];
                      wt_now=wt[r][s][c][kk];
                       sum=sum+data_now*wt_now;
                      }
                }
            }
      out[h][w][kk]=sum;
    }
```

图 11　优化后卷积运算伪代码

改进后的卷积运算基于数据分组的思想，首先在卷积运算中将输入通道、输出通道进行分组，每组内各包含 Tk 个数据，然后将一组特征数据分别与 Tk 组权重数据相乘，从而将乘法并行度变为原来的 $Tk \times Tk$ 倍。

为便于乘累加阵列并行运算,需要将输入特征数据和权重数据进行重组排序。图 12 所示为重组排序后,输入特征数据在存储空间的排列顺序。从图中可以看出,在输入通道方向上将 *Tk* 个特征数据分为一组,这正好与乘累加阵列并行运算方式相匹配。

同样地,输入权重数据也按照特定的排序方式存储在外部存储器指定空间中。权重数据的存储方式如图 13 所示,在输入通道上将 *Tk* 个权重数据分为一组,在输出通道方向上将相邻 *Tk* 组输出通道的权重数据组合成一小块空间。

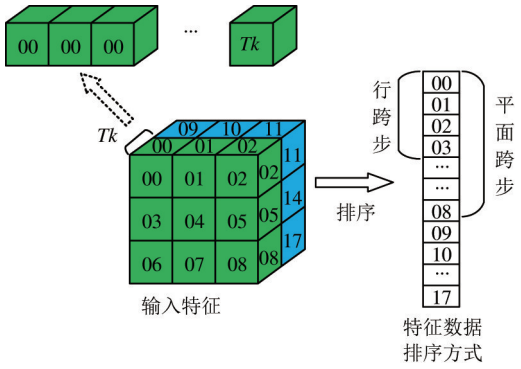


图 12　特征数据在外部存储器的排序顺序

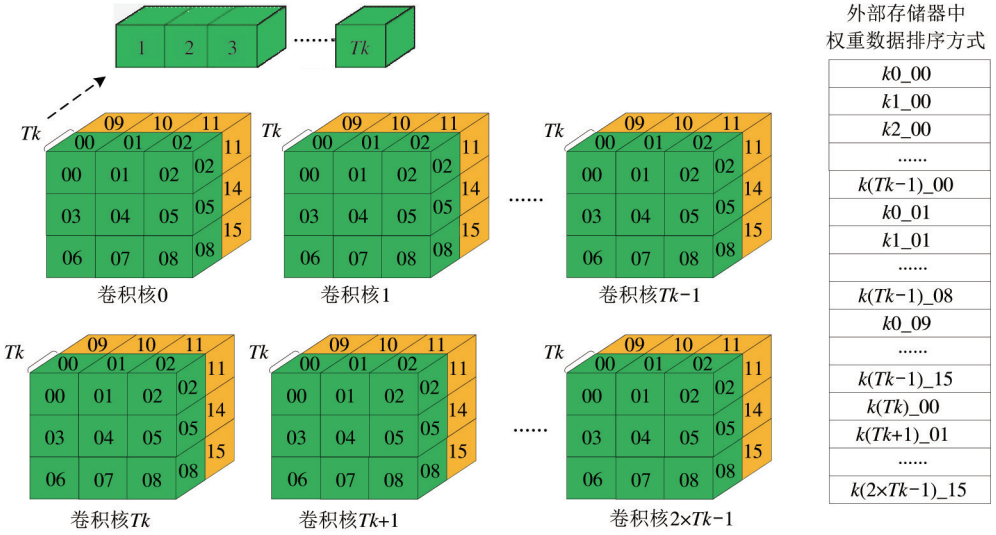


图 13　权重数据在外部存储器的排序顺序

经数据重组后,即可在卷积运算阵列中将一组特征数据与 *Tk* 组权重数据做乘累加运算,得到一组输出特征数据,其运算过程如图 14 所示。

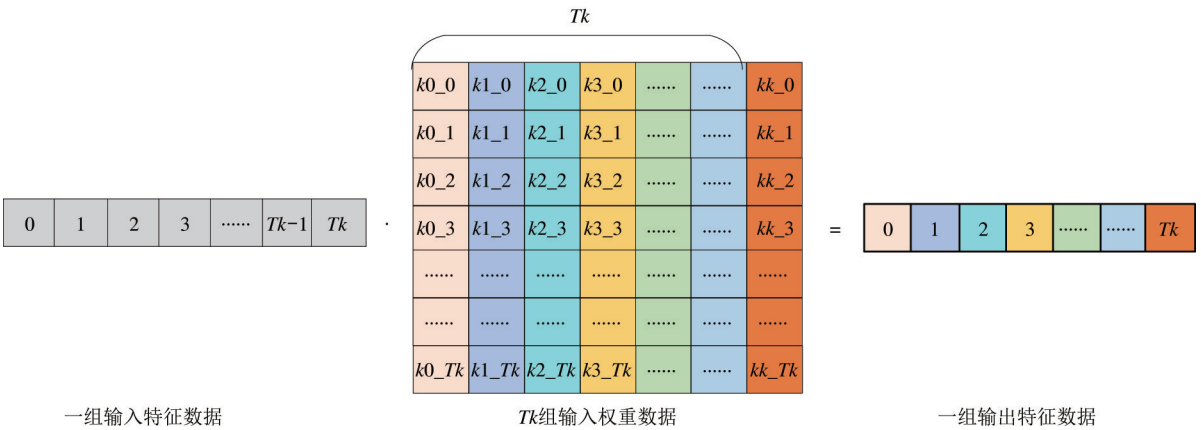


图 14　卷积并行运算方式

最后,将分组数(*Tk*),乘法并行度(*Tk* × *Tk*)作为可选项添加到配置文件中,即可实现并行度可配置的卷积运算阵列。除此之外,本文还将输入数据位宽、片上缓存空间大小等参数添加到配置文件

中,通过选用不同配置方式即可调整 CNN 加速器的资源使用量,从而适配不同 FPGA 平台。表 3 是本文设计的 CNN 加速器的主要配置参数。

表 3　硬件电路中可配置参数表

配置参数	含义
Tk	数据排序时每组内包含数据的数目($Tk = 2^n$,n 为正数)
$Tk \times Tk$	乘法阵列并行度($Tk \times Tk$)
DN	数据位宽(16 或 8)
BRAM _ DATA	特征数据所在的 BRAM 空间地址深度
BRAM _ WT	权重数据所在的 BRAM 空间地址深度

3　加速器架构设计与实现

本文 CNN 加速器分为硬件层(programmable logic,PL)和软件层(processing system,PS),整体设计架构如图 15 所示。PS 层包括指令集、数据复用方式选择和硬件驱动 3 部分;PL 层包括数据数据通路模块、卷积 + 激活模块、池化模块 3 部分。首先在软件层构建单层卷积硬件驱动,并配置专用 CNN 指令集,当单层卷积数据参数量超过板载缓存空间大小时,将该层卷积数据进行分块,并自动选择最佳

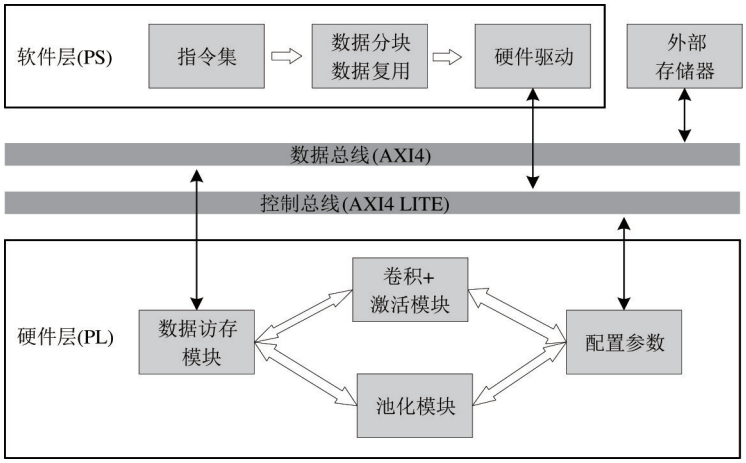


图 15　CNN 加速器设计架构

数据复用方式;然后通过控制总线将相关配置参数发送给硬件层运算模块,驱动运算模块开始工作。在硬件层,数据和权重存放在外部存储器中,数据通路模块负责通过数据总线从外部存储器中取出相关数据传输到卷积 + 激活模块或池化模块参与运算,并将最终运算结果写入到外部存储器指定空间。

3.1　数据通路模块

本设计中的数据通路模块由输入仲裁和输出接口(advanced extensible interface 4,AXI4)组成,如图 16所示,其中仲裁部分对来自卷积运算模块的读权重信号、读特征信号、写特征信号和池化运算模块的读数据信号、写数据信号等 5 种请求类型进行仲裁,最终仲裁目标通过数据总线实现与外部存储器的数据传输。

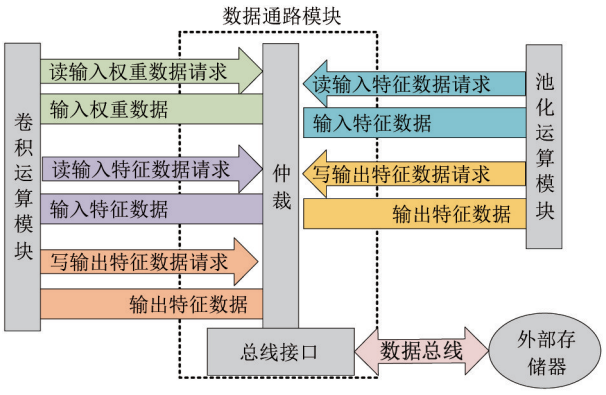


图 16　数据通路模块控制框图

3.2　卷积模块

卷积运算电路如图 17 所示,包括参数配置、DMA1 数据传输、中间缓存空间、卷积运算阵列、激活函数和DMA2数据传输。通过DMA1将权重和

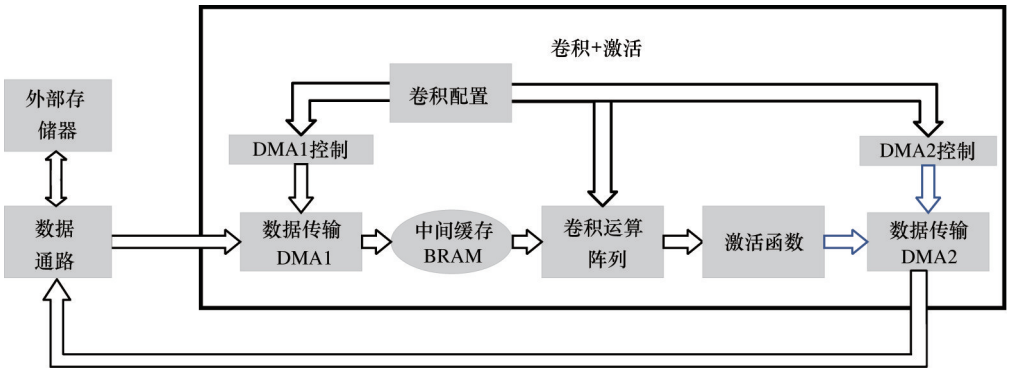


图 17　卷积模块设计图

特征数据从外部存储器中读取存放到中间缓存空间,当中间缓存空间中数据满足一次乘法运算所,开始将数据传输到可配置卷积运算阵列,然后参与卷积、激活运算。

3.3　池化模块

池化运算具体实现过程如图 18 所示,中间缓存部分采用先入先出存储方式(first in first out,FIFO),按数据流可分为 4 部分:行池化、行数据缓存 FIFO1、列池化和输出数据缓存 FIFO2。先对每行进行池化运算,并将结果暂存在 FIFO1 中;当 FIFO1 中存满一行时,再将当前行池化输出结果与 FIFO1 中的数据进行列池化运算;最终池化输出结果存入 FIFO2。

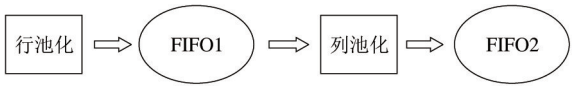


图 18　池化运算流程图

图 19 所示为池化核尺寸为 2×2、特征尺寸为 4×4、池化类型为最大池化时计算示意图。

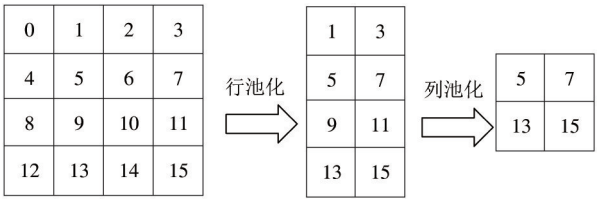


图 19　池化模块运算示意图

4　实验结果分析

4.1　实验环境

神经网络训练过程在个人电脑(PC)端运行,采用 Pytorch 框架,选用 Python 3.7.5 与 Pytorch 1.4 为训练环境;硬件设计过程基于 Vivado 2021.1 系列工具完成硬件描述语言(hardware description language,HDL)代码开发、仿真、综合、布局布线和上板调试;最终板级验证采用 Xilinx 公司的 ZCU104 评估套件。

4.2　实验过程

经硬件电路设计、仿真验证、综合、布局布线后上板实验,下图为当数据位宽为 8、乘法并行度为 1 024 时的实验图。图20为功率使用情况,图21为

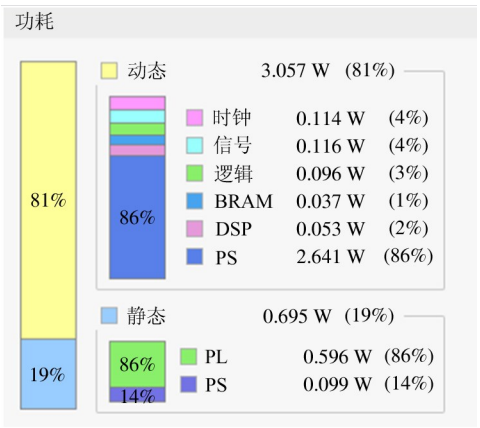


图 20　功率使用情况

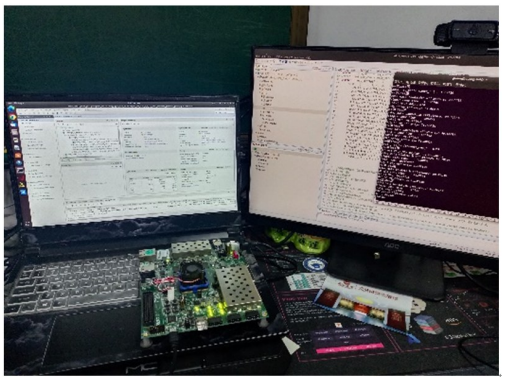


图 21　上板实验

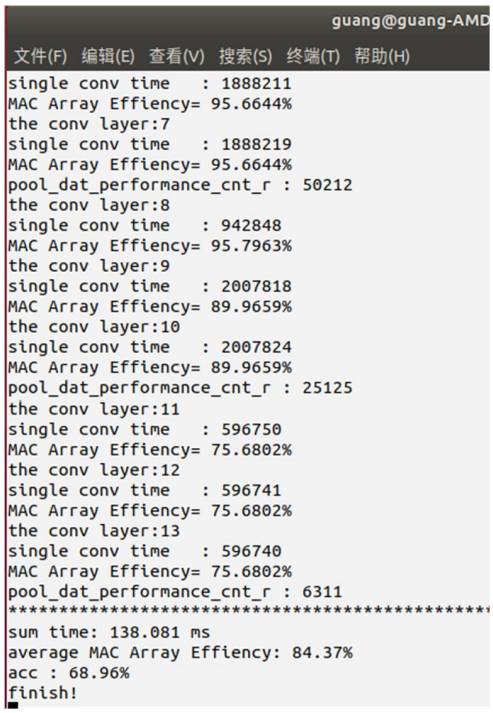


图 22　最终打印信息

最终上板实验,图 22 为最终打印信息。从打印信息中可以看出一张图片推理时间为 138.081 ms,平均 MAC 利用率为 84.37%。

4.3　资源占用

文本设计了一种可配置 CNN 加速器,表 4 选出了其中 3 种硬件配置方式。

表 4　3 种硬件配置方式

配置方式	*Tk*	乘法阵列并行度	数据位宽
1	32	1 024	8
2	32	1 024	16
3	16	256	16

经综合,布局布线后 3 种配置方式下硬件资源使用情况如表 5 所示。

表 5　不同配置方式下资源使用情况

资源类型	配置 1	配置 2	配置 3	总资源
查找表	88 291	46 150	22 075	230 400
查找表型存储器	3 114	5 528	3 019	101 760
触发器	48 130	75 075	28 244	460 800
BRAM	114	228	114	312
乘法器	411	1 179	331	1 728

4.4　网络推理精度

PC 端和 CNN 加速器推理结果如表 6 所示,表中 CNN 加速器推理结果与 PC 端推理结果的误差主要是由于定点量化引起的,数据位宽为 8 时其精度损失为 2.08%,数据位宽为 16 时其精度损失为 1.18%,其中 16 位量化方式可获得更高推理精度。

表 6　VGG 16 网络模型推理结果

网络	数据集	个人电脑端	INT16	INT8
VGG16	ImageNet	70.98%	69.80%	68.90%

4.5　配置方式的选择

CNN 加速器硬件设计的性能指标主要为能效比,其计算表达式如式(6)所示。

$$EER = \frac{G}{power} \tag{6}$$

式中,*EER* 为加速器能效比,*G* 为吞吐量,*power* 为运行功耗。由计算式可知,能效比与吞吐量成正比,与功耗成反比。其中吞吐量也被称做算力,在本文中可以理解为每秒完成的乘累加运算量,其计算式可表示为

$$CAP = f \times C_{\_TK} \tag{7}$$

式中,*CAP* 是加速器吞吐量,*f* 是时钟运行频率,$C_{\_TK}$ 是乘累加阵列的并行度。

乘法阵列并行度越大吞吐量越高,表 7 中列出了不同配置方式下,本文的 CNN 加速器在功耗、吞吐量和能效比等方面的性能。

表 7　不同配置方式下 CNN 加速器的性能

配置方式	方式 1	方式 2	方式 3
频率/MHz	180	180	180
并行度	1 024	1 024	256
数据位宽	8	16	16
吞吐量/GOPs	180	180	45
功耗/W	3.752	4.022	3.633
能效比/(GOPs · W^{-1})	47.97	44.75	12.39

从表中可知,乘法器并行度越大,吞吐量和能效比越高,但往往配置方式的选择还需要综合考虑多方面因素,如当所使用的硬件平台资源充足时,则可

以选择较高的乘法阵列并行度。然而当硬件板卡资源受限时,则可以降低乘法阵列并行度,牺牲吞吐量获取较低的功耗。同理数据位宽的选择也需要考虑多方面因素,当数据位宽为 8 时不满足精度要求,则调整数据位宽为 16。

4.6 性能对比

本文设计的 CNN 加速器在 ZCU104 平台上工作时钟频率为 180 MHz,吞吐量为 180 GOPs,功耗为 3.75 W,能效比为 47.97 GOPs · W^{-1}。将本文设计的 CNN 加速器与其他基于 FPGA 的神经网络加速器进行分析对比,结果如表 8 所示。

表 8 性能分析对比

	文献[12]	文献[13]	文献[14]	本文
平台	VC707	ZCU102	XC7Z020	ZCU104
频率/MHz	100	200	214	180
位宽	16	16	8	8
MAC 利用率/%	72.78	65.40	—	84.37
吞吐量/GOPs	188.4	495.4	84.3	180.0
功耗/W	8.15	15.40	3.50	3.75
能效比/(GOPs · W^{-1})	23.10	32.16	24.10	47.97

首先,从 MAC 利用率分析,与文献[12]和[13]相比,本文通过动态数据复用的方法使得 MAC 利用率分别提升了 11.59% 和 18.97%。其次,从功耗分析,与文献[12]相比,在吞吐量和资源占用相近的情况下,本文通过优化卷积电路使得功耗降低了 2.15倍。最后,从能效比分析,与文献[12]、[13]和[14]相比,由于能效比为吞吐量与功耗的比值,所以其能效比分别提升 2.07 倍、1.49 倍和 1.99 倍。

综上分析可得,本文设计的 CNN 加速器在保持通用能力的基础上,还具有高能效比、低功耗的优势,相比同产品具有较为明显的优势。

5 结 论

针对现阶段神经网络加速器主要应用在单一 FPGA 硬件,不支持硬件平台后期升级迭代的问题,本文提出了一种基于 SOC 的可配置 CNN 加速器,既能够满足多种卷积神经网络模型、实现边缘加速的效果,又能够根据不同硬件平台资源差异,调整卷积运算中乘法阵列并行度,实现对当前硬件平台资源的较好利用。

该加速器虽然具备较高的通用能力,能够实现对当前硬件平台资源的较好利用,但功能模块比较单一,目前只支持目标分类网络。而想要应用于目标检测网络,后续还需设计其他功能模块,例如残差块、上采样等模块。

参考文献

[1] 胡硕, 赵银妹, 孙翔. 基于卷积神经网络的目标跟踪算法综述[J]. 高技术通讯, 2018,28(3):207-213.

[2] 付秀丽, 黎玲萍, 毛克彪, 等. 基于卷积神经网络模型的遥感图像分类[J]. 高技术通讯, 2017,27(3):203-212.

[3] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017,60(6):84-90.

[4] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition [EB/OL]. (2014-09-04)[2023-03-13]. https://arxiv.org/pdf/1409.1556.pdf.

[5] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA: IEEE, 2016:770-778.

[6] HOWARD A G, ZHU M, CHEN B, et al. MobileNets: efficient convolutional neural networks for mobile vision applications[EB/OL]. (2017-04-17)[2023-03-13]. https://arxiv.org/pdf/1704.04861.pdf.

[7] ZHANG X, ZHOU X, LIN M, et al. ShuffleNet: an extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City, USA: IEEE, 2018:6848-6856.

[8] LI X, HUANG H, CHEN T, et al. A hardware-efficient computing engine for FPGA-based deep convolutional neural network accelerator[J]. Microelectronics Journal, 2022,128:105547.

[9] TIAN T, JIN X, ZHAO L, et al. Exploration of memory access optimization for FPGA-based 3D CNN accelerator

[C]//2020 Design, Automation & Test in Europe Conference & Exhibition. Grenoble, France: IEEE, 2020: 1650-1655.

[10] DESOLI G, CHAWLA N, BOESCH T, et al. 14.1 a 2.9 TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems[C]//2017 IEEE International Solid-State Circuits Conference. San Francisco, USA: IEEE, 2017:238-239.

[11] 张志超, 王剑, 章隆兵,等. 面向目标检测的卷积神经网络优化方法[J]. 高技术通讯, 2022,32(3):227-238.

[12] 吴成路. 基于分组剪枝的 CNN 加速器设计与 FPGA 验证[D]. 南京: 东南大学, 2020.

[13] LU L, XIE J, HUANG R, et al. An efficient hardware accelerator for sparse convolutional neural networks on FPGAs[C]//2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines. San Diego, USA: IEEE, 2019:17-25.

[14] GUO K, SUI L, QIU J, et al. Angel-eye: a complete design flow for mapping CNN onto embedded FPGA [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, 37(1): 35-47.

Design and implementation of configurable CNN accelerator based on SoC

ZHANG Liguo, YANG Hongguang, JIN Mei, SHEN Qian

(School of Electrical Engineering, Yanshan University, Qinhuangdao 066004)

Abstract

A configurable convolutional neural network(CNN) accelerator based on system of chip (SoC) is designed to address the issue that the current design of CNN accelerators can only be deployed within a single field programmable gate array(FPGA) and cannot be used across platforms. The accelerator has two characteristics. First, in the circuit design, data bit width, intermediate buffer space size, and multiply accumulate (MAC) array parallelism are optional configuration parameters. By adjusting the resource utilization, the accelerator can adapt to different FPGA hardware. Second, a dynamic data reuse strategy is proposed to reduce the waiting time for data transmission and improve the utilization of the MAC array by dynamically selecting the reuse method based on the difference in total parameter amounts between different reuse methods during data transmission. The scheme is tested on the ZCU104 board, and the experimental results show that when the data bit width is 8, the multiplier array parallelism is 1 024, and the core operation module works at 180 MHz, the peak throughput of the convolution operation array is 180 GOPs, with a power consumption of 3.75 W, and an energy efficiency ratio of 47.97 GOPs · W^{-1}. For the VGG16 network, the average MAC utilization rate of its convolutional layers reaches 84.37%.

Key words: convolutional neural network(CNN), field programmable gate array(FPGA), CNN accelerator, configurable, isomerization acceleration