第 46 卷 第 12 期　　合肥工业大学学报(自然科学版)　　Vol. 46 No. 12
2023 年 12 月　JOURNAL OF HEFEI UNIVERSITY OF TECHNOLOGY (NATURAL SCIENCE)　Dec. 2023

DOI:10.3969/j.issn.1003-5060.2023.12.011

基于 Winograd 算法的高效神经网络加速器及 FPGA 实现

王帅帅，　陈　强，　郭剑博，　肖　昊
(合肥工业大学 微电子学院,安徽 合肥　230601)

摘　要:为了加速卷积神经网络(convolutional neural networks,CNN)的推断过程,文章采用 Winograd 算法,基于现场可编程门阵列(field programmable gate array,FPGA)设计一种高效 CNN 加速器。为解决 Winograd 算法转置后的数据位宽与数字信号处理单元(digital signal processing,DSP)位宽失配问题,文章提出部分积切割方法,充分利用 DSP 实现单周期多输出功能;为降低片上内存占用率,设计一种输入特征图可复用的数据流完成片内外数据交互。所设计的加速器在 XCKU060 板卡上部署,其吞吐率和每个 DSP 运算效率分别达 2.358×10^{12} OPs 和 1.15×10^{9} OPs。结果表明该文提出的加速方法有效提升 CNN 加速器运算单元效率。

关键词:卷积神经网络(CNN);Winograd 算法;现场可编程门阵列(FPGA);处理单元;并行架构
中图分类号:TP183;TN791　　**文献标志码**:A　　**文章编号**:1003-5060(2023)12-1659-07

Design of high-efficiency convolutional neural network accelerator and implementation of FPGA based on Winograd algorithm

WANG Shuaishuai,　CHEN Qiang,　GUO Jianbo,　XIAO Hao
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: In order to improve the inference speed of convolutional neural networks(CNN), this paper utilizes Winograd algorithm to design high-efficiency CNN accelerator based on field programmable gate array(FPGA). In an effort to solve the mismatch problem between the bit width of data after Winograd transpose and the bit width of the digital signal processing(DSP) blocks, a multiplicative partial product cut method is proposed to make full use of DSP to realize the single-cycle multi-output function. In order to reduce the on-chip memory occupancy rate, a data stream with reusable input feature map is designed to complete data interaction between on-chip and off-chip. The designed accelerator is deployed to the XCKU060 board, whose throughput and computing efficiency of each DSP reach 2.358×10^{12} OPs and 1.15×10^{9} OPs, respectively. Experimental results show that the proposed acceleration method effectively improves the efficiency of CNN accelerator computing unit.

Key words: convolutional neural networks(CNN); Winograd algorithm; field programmable gate array(FPGA); process element; parallel architecture

卷积神经网络(convolutional neural networks,CNN)具有良好的图像特征提取功能,广泛应用于图像分类[1]、船舶识别检测[2]和井下视觉识别[3]。传统卷积运算包含大量乘累加计算,使神经网络模型部署至资源有限的现场可编程逻辑门阵列(field programmable gate array,

收稿日期:2022-12-30;**修回日期**:2023-03-16
基金项目:国家自然科学基金资助项目(61974039)
作者简介:王帅帅(1997—),男,山东滨州人,合肥工业大学硕士生;
陈　强(1962—),男,江苏江阴人,博士,合肥工业大学教授,博士生导师;
肖　昊(1982—),男,安徽合肥人,博士,合肥工业大学教授,博士生导师,通信作者,E-mail:xiaohao@hfut.edu.cn.

FPGA)设备存在困难。因此减少卷积计算中乘法次数和降低乘法计算位宽,将CNN高效部署至FPGA成为近年研究热点。

在算法层面,Winograd[4]和快速傅里叶变换(fast Fourier transform,FFT)卷积算法[5]可减少卷积乘法次数,广泛应用于CNN加速器设计。目前网络模型卷积朝卷积核尺寸小、数量多的方向发展,相比FFT算法,Winograd算法在缩减小尺寸卷积乘法次数方面更加高效[6]。Winograd算法通过矩阵变换将卷积数据映射至Winograd域,使用加减法代替乘法,其点乘步骤为2个二维矩阵点乘,可高效映射至FPGA加速器的二维运算阵列,进行并行化处理,提升运算速度。

文献[4]首次将Winograd算法应用于CNN模型,经实验得小尺寸Winograd卷积算子的误差更低,但文献[4]的设计仅适用于步长1的卷积类型;文献[7]设计适配Winograd算法和传统卷积算法的数据流,提出统一架构进行FPGA实现,扩展加速器适用范围,但转置步骤需线下运算,转置后数据需进行片上存储,导致FPGA片内存储器资源占用率高达90%;为扩展加速器适用范围的同时减少片上存储资源占用,文献[6,8]使用卷积分解方法,使Winograd算法适用于不同类型卷积,且设计线上转置模块,减少片上卷积核存储量。

文献[9]设计统一计算单元,处理不同类型卷积并使用数据复用策略减少数据搬运次数;文献[10]提出多模式可重构计算单元,提高计算资源利用率;文献[11]将Winograd算法应用于航空目标检测,进一步扩展算法的应用领域。但上述设计存在量化后定点数据位宽与数字信号处理单元(digital signal processing,DSP)位宽失配的问题,使得FPGA的DSP利用率降低。

针对高效Winograd卷积加速器设计,为降低片上存储资源占用率,本文采用线上转置模块,数据转置后输入运算单元处理,实现输入数据复用且减少内存占用。

针对数据位宽失配问题,本文利用部分积切割方法,运算电路由DSP与查找表(look-up table,LUT)构成,缓解数据位宽失配,提高架构运算单元利用率。

1 CNN优化算法

1.1 CNN量化方法

低比特数据量化方法[10-14]将神经网络模型浮点数转为定点数,减少模型参数存储量。本文使用TensorFlow Lite[12]量化方法将神经网络推断过程浮点数运算转为定点整数运算,并保证量化前后两数值的相关性。此方法为嵌入式设备和FPGA高效部署神经网络提供便利。

CNN量化基本思路是将卷积浮点数运算转为定点整数运算。卷积运算中定点整数q与浮点数r的仿射变换关系为:

$$r = S(q - Z) \tag{1}$$

其中,S、Z为统计得出的常量。S的计算公式为:

$$S = (r_{\max} - r_{\min})/(q_{\max} - q_{\min}) \tag{2}$$

正值参数S由模型训练过程浮点数范围与量化后定点数范围比值得出,代表数值量化前后的比例,也称比例因子。Z的计算公式为:

$$Z = q_{\max} - (r_{\max}/S) \tag{3}$$

若Z为定点小数,则四舍五入转为定点整数。Z为浮点数零值对应量化后数值,也称零点。CNN模型每个激活层对应一组量化参数,每个卷积核输出通道对应一组量化参数。

确定浮点数与定点数量化参数后,将卷积运算过程中运算步骤转为定点数运算。浮点数卷积运算为:

$$r_3^{(i,k)} = \sum_{j=1}^{N} r_1^{(i,j)} r_2^{(j,k)} \tag{4}$$

其中:r_1为卷积核元素;r_2为输入特征图元素;r_3为卷积输出特征图。将浮点数利用TensorFlow Lite量化方法,仿射变换为定点整数,即

$$q_3^{(i,k)} = Z_3 + M\sum_{j=1}^{N}(q_1^{(i,j)} - Z_1)(q_2^{(j,k)} - Z_2) \tag{5}$$

$$M = S_1 S_2 / S_3 \tag{6}$$

由式(6)可得,除M外其余为定点整数。将M统一为式(7),使CNN推理过程为整数运算,即

$$M = 2^{-n} M_0 \tag{7}$$

其中,n为量化后定点整数位宽。硬件实现时,式(5)中定点小数由左移n位以及与定点整数M_0相乘2个步骤运算。

CNN模型训练结束后,统计得出卷积激活层和卷积核浮点数值范围。量化参数线下计算,神经网络部署过程中量化参数由存储器存储。CNN推断过程为定点整数运算,简化CNN模型卷积计算过程,利于嵌入式和FPGA设备部署工作。

TensorFlow Lite[12]方法量化后模型识别准确率符合网络模型精度要求,见表1所列,将VGG16、MobileNetV1、MobileNetV2网络模型量化[15]为

8 bit 定点整数后，模型精度仅下降 3%。

表 1　量化模型精度比较　　%

模型	8 bit 定点数	32 bit 浮点数
VGG16	65.0	68.0
MobilenetV1	67.1	70.9
MobilenetV2	68.1	71.9

1.2　Winograd 算法基本原理

CNN 应用 Winograd 算法中二维卷积算子，二维算子由一维算子嵌套得出[4]。本文采用二维 Winograd 卷积算子，即

$$\boldsymbol{U}=\boldsymbol{G}\boldsymbol{F}\boldsymbol{G}^{\mathrm{T}}$$
$$\boldsymbol{V}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{I}_{\mathrm{in}}\boldsymbol{B}$$
$$\boldsymbol{O}_{\mathrm{out}}=\boldsymbol{A}^{\mathrm{T}}[\boldsymbol{U}\odot\boldsymbol{V}]\boldsymbol{A} \tag{8}$$

其中：$\boldsymbol{F}$、$\boldsymbol{I}_{\mathrm{in}}$ 分别为神经网络卷积核与输入特征图；$\boldsymbol{B}$、$\boldsymbol{A}$、$\boldsymbol{G}$ 分别为 Winograd 算法输入转置矩阵、输出转置矩阵、卷积核转置矩阵。输出特征矩阵由中间结果矩阵 $\boldsymbol{U}$、$\boldsymbol{V}$ 点乘并转置得出。

转置过程会产生误差，本文选择转置矩阵由 0、1、−1、1/2、−1/2 构成的算子，文献[4]实验表明此类算子误差数量级在 10^{-5} 内，符合神经网络识别精度要求。本文选取 3×3 卷积核、4×4 输入块及 2×2 输出规模的 Winograd 卷积算子适配 VGG16 网络模型，记为 $F(2\times2,3\times3)$，其转置矩阵为：

$$\boldsymbol{B}=\begin{pmatrix}1&0&0&0\\0&1&-1&-1\\-1&1&1&0\\0&0&0&1\end{pmatrix},$$
$$\boldsymbol{A}=\begin{pmatrix}1&0\\1&1\\1&-1\\0&1\end{pmatrix},$$
$$\boldsymbol{G}=\begin{pmatrix}1&0&0\\1/2&1/2&1/2\\1/2&-1/2&1/2\\0&0&1\end{pmatrix} \tag{9}$$

$F(2\times2,3\times3)$ 卷积算子可用移位和加法操作实现，简化硬件实现复杂度。CNN 输入特征图是大规模特征矩阵，Winograd 卷积算子无法直接匹配运算。因此采用切片方式[7,16]将输入特征图（input feature map，IFMAP）划分为适配 Winograd 卷积算子的矩阵划块，最终将部分结果组合为输出特征图（output feature map，OFMAP），此结果与传统卷积输出等价。卷积算子数据流如图 1 所示。

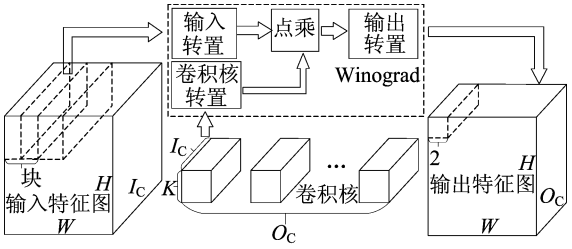

图 1　Winograd 卷积算子数据流

1）本文采用 $F(2\times2,3\times3)$ 卷积算子，输入划块大小为 4×4。VGG16 模型卷积步长为 1 且卷积核尺寸为 3×3，3×3 卷积窗口在 4×4 输入划块进行 2 次水平滑动和 2 次垂直滑动，$F(2\times2,3\times3)$ 卷积算子每次输出 4 个传统卷积等价结果。卷积核滑动步长为 1，2 个输入划块间存在重叠部分，重叠长度为卷积核长度与滑动步长差值，本文设计为 2。按上述规则取输入块，先横向提取首个输入通道 I_C，再纵向提取。划块占满总宽度 W 和高度 H，第 1 个输入通道划块提取完成，其余通道按相同规则划块。

2）单通道卷积核 K 尺寸为 3×3，卷积核通道总量与对应特征图输入通道数相等，卷积核个数与输出通道 O_C 数量相同。卷积核大小可适配本文所选 $F(2\times2,3\times3)$ 卷积算子，无需预处理操作，与输入特征图划块匹配传入 Winograd 卷积算子运算。

3）输入特征图划块和卷积核进行输入转置、卷积核转置、点乘和输出转置步骤。

4）输入特征图与对应卷积核运算结果累加得到首个输出通道特征图。每个输出通道特征图由 $F(2\times2,3\times3)$ 卷积算子输出的 2×2 特征矩阵构成，数量为 $(W/2)^2$。输出通道数量与卷积核个数相等。卷积核与对应输入特征图运算结果聚集为输出特征图矩阵（如图 1）。VGG16 模型卷积步长为 1，卷积核为 3×3，单个传统卷积计算需 9 次乘法，$F(2\times2,3\times3)$ 卷积算子运算包含 4 次传统卷积，乘法总量为 36。使用 Winograd 算法计算 $F(2\times2,3\times3)$ 卷积算子时，4×4 输入和 3×3 卷积核转置后都为尺寸 4×4 的矩阵，然后两矩阵对应点乘，需 16 次乘法计算，最后经输出转置过程得到与传统卷积等价的 2×2 输出。本文使用 Winograd 算法运算 $F(2\times2,3\times3)$ 卷积算子需 16 次乘法，相比传统卷积的 36 次乘法，节约率达 55.56%。

2　CNN 加速器设计

2.1　CNN 加速器系统架构

基于 Winograd 算法的神经网络加速器系统

架构如图 2 所示,CNN 模型处理部分由 FPGA 片上完成,数据存储至片外动态随机存取内存(dynamic random access memory,DRAM),两者通过高级可扩展接口协议(advanced extensible interface,AXI)通信。加速器由系统控制模块、输入缓存模块、权重缓存模块、维度转换模块、输入转置模块、权重转置模块、运算处理单元(process element,PE)、输出转置模块、最大池化模块、量化模块和输出缓存模块构成。开发板上电启动时,模型参数和图片数据从安全数码卡加载至 DRAM 中,数据准备完成后,CNN 加速器开始工作。每轮计算将部分输入特征图和权重数据通过 AXI4 总线接口传入片上缓存单元,加速器循环处理剩余数据。

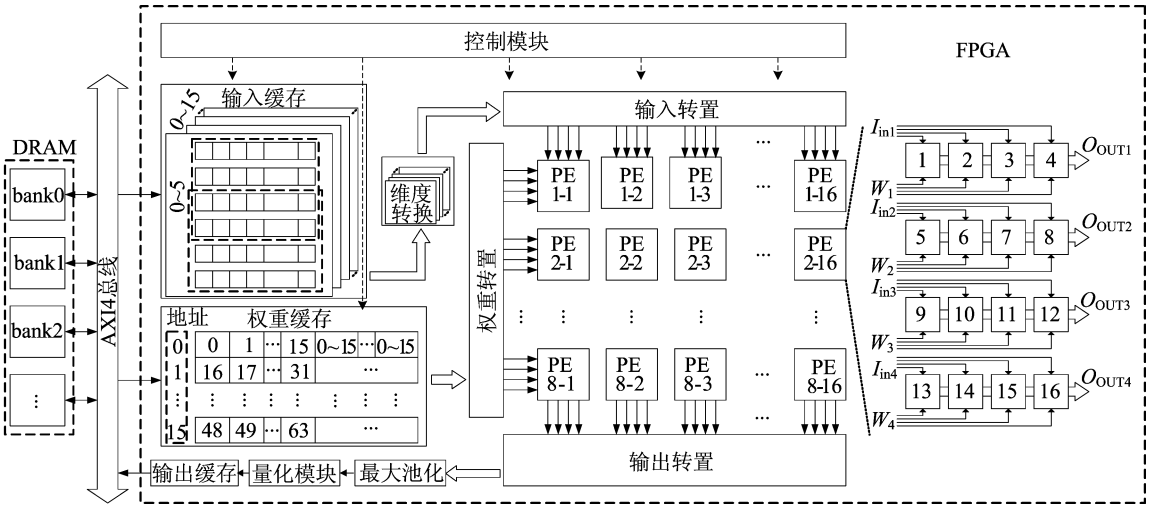

图 2 CNN 加速器整体架构

硬件加速器工作步骤如下:

1) 输入特征图缓存用双缓冲区策略[17],部分行数据预存至输入缓存模块,经维度转换模块后,二维输入数据块传入转置模块线上转置,输入转置模块由加法器和移位寄存器构成,所用系数与输入转置矩阵中元素一致。

2) 权重缓存由 256 个片上随机存储器块(block random access memory,BRAM)构成。每周期提供 256 组输入通道权重,每个运行周期可处理 16 个卷积输入通道和 16 个输出通道,与处理单元阵列规模对应,存储块可提供处理单元每个周期计算所需权重参数。

3) 输入转置模块和权重转置模块将数据转置,广播至处理单元进行点乘运算。每个处理单元由 16 个 DSP 与 LUTs 组成的乘法器组成,与本文 Winograd 卷积算子所需 16 个乘法操作匹配。本文二维处理单元每个运算周期执行一组点乘运算,相比一维计算单元缩短了等待周期,提高了加速器运行吞吐率。

4) 处理单元运算结果以二维形式广播至输出转置模块,进行输出转置,该模块转置矩阵如式(9)所示。转置后结果经通道累加后缓存,直至所有输入通道卷积循环运算完成,得出最终累加结果。输出结果即为传统卷积等价结果。后经最大池化、量化模块处理,通过 AXI4 总线接口传至外部 DRAM 缓存,作为后续卷积层输入特征图数据。

2.2 输入缓存与线上维度转换模块

卷积核参数经 Winograd 算法转置过程,存储容量增加。为减少片上存储资源开销,本文采取线上转置模块[8]。因为转置过程需二维计算,所以将缓存一维数据转为二维数据运算。

本文设计维度转换模块和双缓冲区缓存模块实现上述功能,维度转换模块由移位寄存器构成,如图 3a 所示。$F(2\times2,3\times3)$ 卷积算子输入划块为 4×4 且两划块间重叠长度为 2,输入缓存模块用行缓存且 2 组相邻行缓存间共用 2 个缓存单元,此模块可同时输出 4 行数据对应划块纵向维度 4,同时实现 Winograd 输入划块的数据复用,如图 3b、图 3c 所示。

移位寄存器完成输入划块横向维度 4,以2 个相邻输入块为例,移位寄存器每周期接收2 个 输入缓存的特征像素,并左移 2 位。按此运行规则,2 个 等待周期后移位寄存器高 4 位,每周期输出一组有效特征块,与后续模块组合完成全流水线运算。一组移位寄存器输出 4 列数据,4 组相同移

位寄存器输出 4 行划块，共同输出二维 4×4 特征划块。根据 $F(2\times2,3\times3)$ 卷积算子输入特征块重叠规律，设计输入缓存模块对数据进行垂直维度复用，维度转换模块实现水平维度复用数据。片上内存占用率减少至 58%。

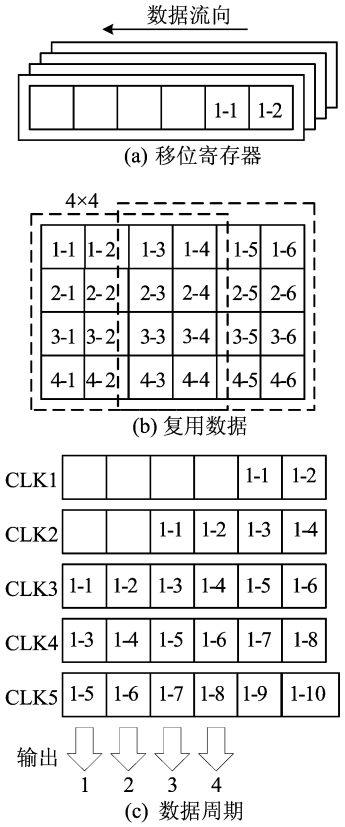

图 3　移位寄存器数据流

2.3　部分积切割方法

经 $F(2\times2,3\times3)$ 卷积算子转置后，输入特征图和权重 8 bit 数据增加至 10 bit，即乘数为10 bit 定点整数。DSP 乘法器位宽为 27×18，若直接用此乘法器，则造成乘法器与数据位宽失配，导致硬件资源乘法器利用率降低。

为解决定点数位宽与 DSP 乘法器失配问题，本文设计部分积切割方法，硬件由 DSP 和 LUTs 组合电路实现，最大化利用乘法器位宽，提升乘法器资源利用率。

数字信号处理器支持单周期操作，计算 2 个 8 bit 定点整数乘法[18]，但 8 bit 输入和权重数据经转置过程后，位宽增加至 10 bit，单个 DSP 无法计算 2 次乘法。本文利用部分积切割方法，电路由 LUTs 扩展模块与 DSP 构成，运算单元单周期可同时计算 2 次乘法，实现单周期多输出功能。DSP 部分积划分如图 4a 所示，LUTs 部分积划分如图 4b 所示，完成的乘法运算为：

$$M_{\text{Mult0}}=IW_0=M_{\text{Mult0\_P}_1}+M_{\text{Mult0\_P}_2}+M_{\text{Mult0\_P}_3},$$
$$M_{\text{Mult1}}=IW_1=M_{\text{Mult1\_P}_1}+M_{\text{Mult1\_P}_2}+M_{\text{Mult1\_P}_3} \tag{10}$$

其中：I、W 两乘数为 10 bit；式(10) 结果为 20 bit。本设计将 10 bit 权重划分为 8、2 bit 2 个部分，2 个 8 bit 权重部分与 10 bit 输入乘法由 DSP 完成，2 个 2 bit 权重部分与 10 bit 输入乘法由 LUTs 组成扩展部分完成，乘法结果由部分积相加得出。

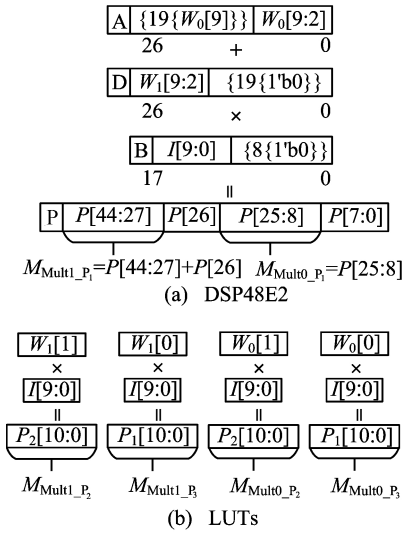

图 4　部分积切割映射关系

DSP 与 LUTs 组合电路结构如图 5 所示。2 bit 权重与 10 bit 输入乘法由 LUTs 组成扩展单元计算，扩展单元的部分积运算由与门和寄存器实现。

从图 5a 可以看出，第 1 个扩展单元输入卷积核 0 低 2 bit，输出相乘的 2 个部分积 $W_{\text{W0\_E0}}$ 和 $W_{\text{W0\_E1}}$。从图 5b 可以看出，第 2 个扩展单元输出卷积核 1 低 2 bit，输出相乘的 2 个部分积 $W_{\text{W1\_E0}}$ 和 $W_{\text{W1\_E1}}$。卷积核 0 乘法结果为 W_0 对应的3 个部分积之和，卷积核 1 乘法结果为 W_1 对应的 3 个 部分积之和。

从图 5c 可以看出，DSP 中乘法器和预加器组合计算 2 个乘法，2 个乘数分别为 8 bit 权重与 10 bit 输入。将 2 个 8 bit 权重通过预加器拼接至乘法器 27 bit 输入端，10 bit 输入低位补 0 后传至乘法器 18 bit 输入端，结果按输出端口 P 所示规则截取，DSP 输出端口第 8～25 位为卷积核 0 的高 8 位与输入相乘结果，DSP 输出端口第 27～44 位与第 26 位之和为卷积核 1 的高 8 位与输入相乘结果。

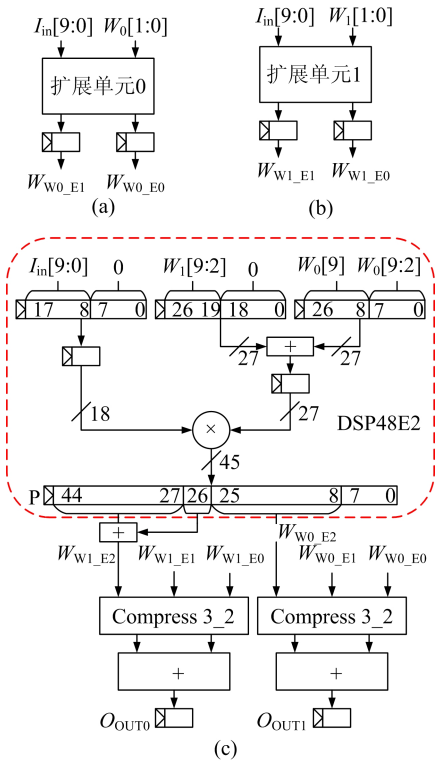

图 5 DSP 与 LUTs 组合电路结构

此外,本文设计部分和加法器采用华莱士树加法器,Compress 3_2 使用全加器将 3 个部分积压缩为 2 个部分进行加法,有助于缩短关键路径延迟,减小电路建立时间违例概率,使系统运行稳定。

3 性能评估

3.1 软硬件平台速度对比

本文神经网络加速器与中央处理器(central processing unit,CPU) 和图形处理器(graphics processing unit,GPU) 平台速度对比见表 2 所列。

表 2 不同平台速度对比

实现平台	器件型号	推理时间 /ms
FPGA	Xilinx XCKU060	19.12
CPU	Intel i5-4590	640.11
GPU	Nvidia GTX1070 Ti	65.97

CPU 和 GPU 平台的软件语言使用 Python 3.7,机器学习库为 Pytorch 1.12,指令集架构为 CUDA 11.7。本文所用中央处理器和图形处理器运行 VGG16 模型推理速度分别为 640.11、65.87 ms。本设计使用 XCKU060 FPGA 部署 VGG16 模型,工作频率达 180 MHz,推理阶段卷积计算花费时间为19.12 ms。相比 Intel i5-4590 CPU 和 Nvidia GTX 1070 Ti GPU,本文硬件加速器卷积运算速度分别提升 33.47、3.43 倍。

3.2 硬件平台性能评估

本文将 VGG16 网络模型部署至 XCKU060 开发板,并评估加速器性能。仿真验证和硬件实现 EDA 软件为 Vivado 2021.1。构建开发板评估环境,测试数据和网络模型权重数据预存至 SD 卡,开发板上电后控制器初始化系统,将测试数据和模型参数搬运至DDR。其输入数据与权重参数为量化后8 bit 定点整数。开发板片上资源利用率见表 3 所列。

表 3 FPGA 片上资源利用率

片上资源	已使用	利用率 /%
LUT 数量	235 492	71
BRAMs/KiB	22 824	58
DSP 数量	2 048	74

本文设计与近年来相关工作性能的对比见表 4 所列。

文献[6] 使用大规模处理单元阵列提升硬件加速器吞吐量,使用 DSP 数量超出本文 1 倍,但 Winograd 卷积加速器[6] 的性能仅比本文设计高 0.11 倍。文献[7] 将 Winograd 算法数据流调整与传统卷积数据流匹配,扩展加速器应用范围,但文献[7] 计算数据位宽与硬件匹配度低,造成乘法器资源的效率仅为 0.28。

文献[9] 利用卷积核尺寸为 3×3 的算子,使用统一转置矩阵节约片上硬件资源。但 CNN 加速器[9] 选取8 bit 定点数据,而设计中 DSP 无法高效利用低比特数据优势。

表 4 VGG16 模型加速器性能对比

参数	文献[6]	文献[7]	文献[9]	文献[11]	本文
卷积算法	Winograd				
FPGA	Xilinx XCVU9P	Virtex-7 Vx690T	XC7J325T	Xilinx VCU118	Xilinx XCKU060
频率 /MHz	330	200	150	200	180
DSP 数量	4 096	1 436	750	4 608	2 048
吞吐率 /(10^{12} OPs)	5.288	0.407	0.440	3.812	2.358
每个 DSP 运算效率 /(10^9 OPs)	1.28	0.28	0.52	0.82	1.15

文献[11]使用乒乓操作提升DSP利用率，但是运算单元乘法器[11]输入端的位宽为 27 bit，其位宽占用率仅为59%。本文通过输入缓存模块和维度转换模块对数据复用，减少数据搬移，使加速器处理卷积层的平均吞吐率达 2.358×10^{12} OPs。此外，采用部分积切割方法，充分利用 DSP 输入端的位宽，使 CNN 加速器每个 DSP 运算效率达 10^9 OPs，相比文献[9]、文献[11]分别提升了54%、29%。

4　结　　论

本文面向 FPGA 硬件部署神经网络模型，提出一种基于Winograd算法高效部署CNN模型的加速器架构。利用 Winograd 卷积算法代替VGG16 模型中传统卷积算法，减少 55.5% 的乘法运算量。通过将FPGA片上DSP与LUTs资源结合构成处理单元运算电路，实现部分积切割方法，具备单周期多输出特点，解决数据位宽与乘法器失配问题，提高乘法器利用率和处理单元运行效率。实验结果表明，加速器运行 VGG16 模型卷积层的平均吞吐率和每个 DSP 运算效率分别为 2.358×10^{12} OPs 和 1.15×10^{9} OPs。

[参　考　文　献]

[1] KAREN S, ANDREW Z. Very deep convolutional networks for large-scale image recognition[C]//International Conference on Learning Representations. [S. l. : s. n.], 2015: 1-14.

[2] 崔巍，杨亮亮，夏荣，等. 基于 Faster R-CNN 算法的船舶识别检测[J]. 合肥工业大学学报(自然科学版)，2020，43(2)：182-187.

[3] 韩江洪，沈露露，卫星，等. 基于轻量级 CNN 的井下视觉识别策略[J]. 合肥工业大学学报(自然科学版)，2020，43(11)：1469-1475.

[4] ANDREW L, SCOTT G. Fast algorithms for convolutional neural networks[C]//IEEE Conference on Computer Visionand Pattern Recognition. [S. l.]: IEEE, 2016: 4013-4021.

[5] LIANG Y, LU L, XIAO Q, et al. Evaluating fast algorithms for convolutional neural networks on FPGAs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(4): 857-870.

[6] YANG C, WANG Y Z, WANG X L, et al. WRA: a 2.2-to-6.3 TOPS highly unified dynamically reconfigurable accelerator using a novel Winograd decomposition algorithm for convolutional neural networks[J]. IEEE Transactions on Circuits and Systems-I: Regular Papers, 2019, 66(9): 3480-3493.

[7] KALA S, JOSE B R, MATHEW J, et al. High-performance CNN accelerator on FPGA using unified Winograd-GEMM architecture[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 27(12): 2816-2828.

[8] YANG C, WANG Y Z, WANG X L, et al. A stridebased convolution decomposition method to stretch CNN accelerat-ion algorithms for efficient and flexible hardware implementation[J]. IEEE Transactions on Circuits and Systems-I: Regular Papers, 2020, 67(9): 3007-3020.

[9] HUANG C, DONG X, LI Z, et al. Efficient stride 2 Winograd convolution method using unified transformation matrices on FPGA[C]//International Conference on Field-Programmable Technology (ICFPT). [S. l. : s. n.], 2021: 1-9.

[10] YUAN Z, NI W, RAN J. Reconfigurable convolutional neural network accelerator based on Winograd algorithm[J]. Electronic Science and Technology, 2022, 35(12): 1-9.

[11] SHI L, WANG S, XIAO H. Design of neural network accelerator for aeronautical target detection[J]. Aeronautical Science & Technology, 2022, 33(5): 89-96.

[12] JACOB B, KLIGYS S, CHEN B, et al. Quantization and training of neural networks for efficient integerarithmeticonly inference[C]//IEEE Conference on Computer Vision and Pattern Recognition. [S. l.]: IEEE, 2018: 2704-2713.

[13] WU J, LENG C, WANG Y, et al. Quantized convolutional neural networks for mobile devices[C]//IEEE Conference on Computer Vision and Pattern Recognition. [S. l.]: IEEE, 2016: 4820-4828.

[14] HU Y, LI M. Structured compression and acceleration of network based on tiny-YOLOv3[J]. Electronic Science and Technology, 2022, 36(8): 1-8.

[15] XU Y, WANG S, LI N, et al. Design and implementation of an efficient CNN accelerator for low-cost FPGAs[J]. IEICE Electronics Express, 2022, 19(19): 1-6.

[16] SHI F, LI H. Sparse Winograd convolutional neural networks on small-scale systolic arrays[C]//FPGA'19, California USA. [S. l. : s. n.], 2019: 1-7.

[17] 赵博雅. 基于卷积神经网络的硬件加速器设计及实现研究[D]. 哈尔滨：哈尔滨工业大学，2018.

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

（责任编辑　张　镅）