计算机科学
COMPUTER SCIENCE

CNN景象匹配算法的加速设计与FPGA实现

王晓峰，李超然，路坤锋，栾天娇，姚娜，周辉，谢宇嘉

引用本文

王晓峰，李超然，路坤锋，栾天娇，姚娜，周辉，谢宇嘉. CNN景象匹配算法的加速设计与FPGA实现[J]. 计算机科学，2023，50(11)：8-14.

WANG Xiaofeng, LI Chaoran, LU Kunfeng, LUAN Tianjiao, YAO Na, ZHOU Hui, XIE Yujia. Acceleration Design and FPGA Implementation of CNN Scene Matching Algorithm [J]. Computer Science, 2023, 50(11): 8-14.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

智能物联网终端自适应模型量化方法

Adaptive Model Quantization Method for Intelligent Internet of Things Terminal

计算机科学, 2023, 50(11): 306-316. https://doi.org/10.11896/jsjkx.230300078

基于多粒度的Transformer目标检测算法

Transformer Object Detection Algorithm Based on Multi-granularity

计算机科学, 2023, 50(11): 143-150. https://doi.org/10.11896/jsjkx.230600028

NeuronSup:基于偏见神经元抑制的深度模型去偏方法

NeuronSup:Deep Model Debiasing Based on Bias Neuron Suppression

计算机科学, 2023, 50(11): 122-131. https://doi.org/10.11896/jsjkx.220900169

基于SVD的深度学习模型对抗鲁棒性研究

Study on Adversarial Robustness of Deep Learning Models Based on SVD

计算机科学, 2023, 50(10): 362-368. https://doi.org/10.11896/jsjkx.220800090

融合跟踪器:融合图像特征和事件特征的单目标跟踪框架

Fusion Tracker:Single-object Tracking Framework Fusing Image Features and Event Features

计算机科学, 2023, 50(10): 96-103. https://doi.org/10.11896/jsjkx.220900075

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221100104

CNN 景象匹配算法的加速设计与 FPGA 实现

王晓峰　李超然　路坤锋　栾天娇　姚　娜　周　辉　谢宇嘉
北京航天自动控制研究所　北京 100854
宇航智能控制技术国家级重点实验室　北京 100854

摘　要　基于卷积神经网络的景象匹配算法较传统方法具有更高的匹配精度、更好的适应性以及更强的抗干扰能力。但是,该算法有海量的计算与存储需求,导致在边缘端部署存在巨大困难。为了提升计算实时性,文中设计并实现了一种高效的边缘端加速计算方案。在分析算法的计算特性与整体架构的基础上,基于 Winograd 快速卷积方法,设计了一种面向特征匹配层的专用加速器,并提出了利用专用加速器与深度学习处理器流水线式计算特征匹配层和特征提取网络的整体加速方案。在 Xilinx 的 ZCU102 开发板上进行实验发现,专用加速器的峰值算力达到 576 GOPS,实际算力达 422.08 GOPS,DSP 的使用效率达 4.5 Operation/clock。加速计算系统的峰值算力达 1600 GOPS,将 CNN 景象匹配算法的吞吐时延降低至 157.89 ms。实验结果表明,该加速计算方案能高效利用 FPGA 的计算资源,实现 CNN 景象匹配算法的实时计算。

关键词:加速计算;景象匹配算法;深度学习;FPGA;Winograd 算法;专用加速器
中图法分类号　TP391

Acceleration Design and FPGA Implementation of CNN Scene Matching Algorithm

WANG Xiaofeng, LI Chaoran, LU Kunfeng, LUAN Tianjiao, YAO Na, ZHOU Hui and XIE Yujia
Beijing Aerospace Automatic Control Institute, Beijing 100854, China
National Aerospace Intelligence Control Technology Laboratory, Beijing 100854, China

Abstract　Compared with traditional methods, the CNN-based scene matching algorithm has higher matching accuracy, better adaptability and stronger anti-interference ability. However, the algorithm has massive computing and storage requirements, which makes it difficult to deploy at the edge. To improve the real-time computing, an efficient edge-side acceleration scheme is designed and implemented. On the basis of analyzing the computation characteristics and overall architecture of the algorithm, correlation specific accelerator(CSA) is designed based on Winograd fast convolution method, and the acceleration scheme using CSA and deep-learning processor unit(DPU) pipelined computing feature correlation layer and feature extraction network is proposed. Experiments on Xilinx's ZCU102 development board finds that the peak perfor-mance of CSA reaches 576 GOPS, the actual performance reaches 422.08 GOPS, and the DSP usage efficiency reaches 4.5 Operation/clock. The peak performance of the acceleration system reaches 1 600 GOPS, and the throughput delay of the algorithm is reduced to 157.89 ms. Experimental results show that the acceleration scheme can efficiently utilize the computing resources of the FPGA, to realize the real-time computing of the CNN-based scene matching algorithm.

Keywords　Acceleration computing, Scene matching algorithm, Deep learning, FPGA, Winograd algorithm, Specific accelerator

1　引言

景象匹配技术通过在基准图中匹配实时图的方式为飞行器提供辅助导航的位置信息,是飞行器辅助定位的重要手段之一,也是计算机视觉技术在飞行器导航任务中的重要应用。卷积神经网络(Convolutional Neural Networks, CNNs)凭借其强大的特征提取能力,在图像分类[1-3]、目标检测[4-7]、图像分割[8-9]等计算机视觉任务中取得了显著成效。研究发现,基于 CNN 的景象匹配算法较传统算法[10-13]具有更高的匹配精度、更好的环境适应性以及更强的抗干扰能力。

CNN 以海量的计算为代价,提升了景象匹配算法的精度。在飞行器辅助定位这种具有强实时要求的应用场景中,实现边缘端的高效、实时计算已经成为算法落地的关键挑战。如图 1 所示,该算法由基于 CNN 的特征提取网络和特征匹配层两部分组成。前者用于从基准图和实时图中提取特征,得到基准特征图 $\boldsymbol{B}$ 和实时特征图 $\boldsymbol{R}$,由卷积层、池化层、非线性激活层等 CNN 的典型算子组成,可由深度学习处理器(Deep-learning Processing Unit, DPU)[14-25]完成加速计算。后者用于

到稿日期:2022-11-12　返修日期:2023-05-11
基金项目:国家重点实验室基金(61425010302)
This work was supported by the State Key Laboratory Fund(61425010302).
通信作者:王晓峰(wangxf.casc@foxmail.com)

计算 $\boldsymbol{B}$ 与 $\boldsymbol{R}$ 的相关性进而得到匹配结果 $\boldsymbol{C}$，其计算特性与典型 CNN 算子的差异较大，在大部分 DPU 上的计算效率极低。

为实现算法的高效计算，我们利用 FPGA 硬件可编程的优势，设计了一种特征匹配层专用加速器（Correlation Specific Accelerator，CSA），集成基于 FPGA 的 DPU，流水线式计算特征匹配层和特征提取网络，从而大幅缩减 CNN 景象匹配算法的整体计算时间。

目前，已有大量可用于特征提取网络加速计算的 DPU 被提出[14-25]，但能够高效计算特征匹配层的加速器依然很少。因此，本文将在提出 CNN 景象匹配算法整体加速计算方案的基础上，复用成熟 DPU 的 IP 核，专注于 CSA 的设计与研究。

CSA 的设计过程中需重点考虑并解决以下问题：1）DPU 和 CSA 的性能提升均依赖 FPGA 中 DSP 资源的数量，导致 DSP 数量成为限制系统性能的主要瓶颈；2）特征匹配层的输出通道为 1，在既定的输入特征排序方式下 CSA 可挖掘的并行度有限。针对上述问题，考虑到 Winograd 算法[26]可大幅降低卷积层的计算复杂度，而特征匹配层可等效为超大卷积核的单输出通道卷积层，因此可利用 Winograd 降低 DSP 资源量对 CSA 性能的限制。在 Winograd 计算方式下，每个时钟由 $\boldsymbol{B}$ 分块与 $\boldsymbol{R}$ 分块并行计算得到 $\boldsymbol{C}$ 分块，计算过程中使用的并行计算方式可等效为卷积层的“特征图内并行”和“卷积核内并行”的方式[20]。另外，由于 $\boldsymbol{B}$ 与 $\boldsymbol{C}$ 的通道数一般较大，还可通过并行计算多个输入通道的方式来提高计算并行度。总之，我们利用 Winograd 快速卷积方法，结合“特征图内并行”“卷积核内并行”与“输入通道并行”3 种并行方式，在有限的 DSP 资源下设计实现了一款高并行度的 CSA。

在 CSA 的设计中，为了降低 $\boldsymbol{B}$，$\boldsymbol{R}$ 和 $\boldsymbol{C}$ 的变换过程对 FPGA 资源的消耗量，我们通过变换矩阵化简、同类项合并、等效微调计算次序等一系列方法，将复杂的矩阵乘转化为低代价的移位和加法操作。实现高并行度计算单元的连续满载运行的关键在于片上缓存系统的数据供应能力，通过深入分析特征匹配层与 Winograd 算法的计算特性，我们设计了基于 Line-buffer 结构和滑窗寄存器的片上缓存系统，实现了 $\boldsymbol{B}$ 数据片上复用的最大化以及片上数据带宽与计算单元算力的高度匹配。本文的主要贡献可归纳为以下几点：

1）提出了流水线式计算特征提取网络和特征匹配层的整体加速方案。

2）设计了首个基于 Winograd 快速卷积方法的特征匹配层专用加速器，使得单个 DSP 的使用效率达 4.5 Operation/clock 以上。

3）设计了基于 Line-buffer 结构和滑窗寄存器的片上缓存系统，结合带宽自适应的数据传输单元，实现了数据带宽与计算能力的高效匹配，在典型算例下 CSA 的计算效率达 76%以上。

2 相关工作

目前，CPU，GPU 和 DPU 均是 CNN 的主流计算平台，但适用场景各不相同。在能耗受限的边缘端，CPU 和 GPU 由于能耗比过低而难以满足应用需求。如 Intel Xeon E5（CPU）运行 AlexNet 需要 254.5 ms，功耗为 80 W，能效比仅为 0.07 GOPS/W；同样地，NVIDIA K40（GPU）需要 7.1 ms，功耗高达 235 W，能效比仅为 0.87 GOPS/W。因此，需要针对深度学习算法的计算特性设计专门的 DPU，通过简化控制逻辑、优化片上缓存架构、提高计算并行度等方法来提升能耗比。

随着 CNN 的广泛应用，已有大量 DPU 被提出，根据实现方式的不同可分为基于 ASIC 和基于 FPGA 两种。基于 ASIC 的 DPU 包括世界上第一款深度神经网络加速器 Diannao[14]、基于超大脉动阵列的云端神经网络加速器 TPU[15]、面向低功耗场景的 Eyeriss[16]等，它们具有超高的能效比和充足的算力，但存在灵活性不足、开发周期长、设计成本昂贵等问题，难以适用于部分细分领域的特殊算法。基于 FPGA 的 DPU 得益于 FPGA 硬件可编程的能力具有更高的灵活性，并可根据具体算法的特性进行架构优化、超参调整和功能裁剪，从而达到很高的能效比和算力。基于 FPGA 的 DPU 包括首次利用 Roofline Model[17]提升性能的 CNN 加速器[18]、组合利用 CNN 多种并行方式的 Angel-Eye[19]、利用旋转寄存流水结构提升计算并行度的加速器[20]、专注于可扩展性和高效性的 DPU[21-22]等，但它们的算力均受限于 FPGA 中 DSP 资源的数量。

针对上述问题，北京大学高能效计算与应用中心的研究者基于 Winograd 设计了 CNN 加速器，在 Xilinx ZCU102 开发板上的能效比为 NVIDIA TitanX GPU 的 2.98 倍[23]。后续研究者基于 Winograd 算法进一步设计了面向 3D 卷积和稀疏卷积的 CNN 加速器[24-25]，但均面向常规卷积核、多输出通道的卷积层进行设计，难以高效计算特征匹配层。

本文提出基于 Winograd 快速卷积方法设计 CSA，以突破 DSP 资源对算力的限制；并与高效的 DPU 流水线式计算特征匹配层和特征提取网络，充分发挥 FPGA 硬件可编程的优势，实现对 CNN 景象匹配算法的高效计算。

3 CNN 景象匹配算法与特征匹配层

3.1 CNN 景象匹配算法

景象匹配算法根据基准图与实时图成像方式的异同分为同源和异源景象匹配。同源景象匹配算法适用场景相对受限，而异源景象匹配算法的应用限制则更小，如基准图可使用更容易获取的可见光图像，而实时图则可使用不易受云雾、黑夜等自然因素影响的红外图像。

景象匹配算法的主流技术可分为基于区域的方法[10-11]和基于特征的方法[12-13]。基于区域的方法主要利用灰度信息及其变换进行匹配，一般适用于同源景象匹配。而基于特征的方法则是通过从图像中提取、抽象特征，并利用特征的不变性来进行匹配，是异源景象匹配领域的热门研究方向。这类算法一般分为两步：1）从图像中提取特征；2）计算特征图之间的相似性，测度最大值则为匹配位置。如 Cao 等利用 HOG 算法提取图像特征，采用向量归一化相关系数作为特征相似性度量准则[12]。CNN 景象匹配算法亦属于基于特征的方法。计算过程如图 1 所示，首先利用 CNN 强大的特征提取

能力从基准图和实时图中提取高层次特征，然后通过特征匹配层计算相关度，选择最大相关值作为匹配点。

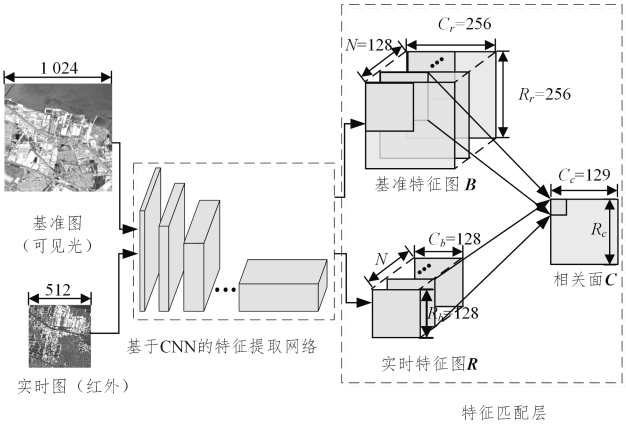

图 1 CNN 景象匹配算法的整体架构

Fig. 1 Architecture of CNN scene matching algorithm

3.2 特征匹配层及其 Winograd 计算过程

特征匹配层的具体计算过程为：

$$C_{i,j}=\sum_{n=0}^{N-1}\sum_{u=0}^{H_r-1}\sum_{v=0}^{W_r-1}B_{i+u,j+v,n}\times R_{u,v,n} \tag{1}$$

其中，$i\in[0,H_c)$和 $j\in[0,W_c)$是 $\boldsymbol{C}$ 的行、列索引；$u\in[0,H_r)$和 $v\in[0,W_r)$是 $\boldsymbol{R}$ 的行、列索引；则 $i+u\in[0,H_b)$和 $j+v\in[0,H_b)$是 $\boldsymbol{B}$ 的行、列索引。其他符号及其含义如表 1 所列。分析式(1)可知，特征匹配层的计算过程本质上是一种通用矩阵乘(General Purpose Matrix Multiplication，GEMM)，其计算复杂度为：

$$O(N\times H_r\times W_r\times H_c\times W_c) \tag{2}$$

式(2)与 $\boldsymbol{B}$，$\boldsymbol{R}$ 的尺寸和通道数成比例。Winograd 可通过矩阵变换把 GEMM 转换为计算复杂度更低的逐点矩阵乘(Element Wise Matrix Multiplication，EWMM)。

表 1 常用符号及其含义

Table 1 Common symbols and their meanings

符号	含义
$\boldsymbol{B},\boldsymbol{R},\boldsymbol{C}$	基准特征图、实时特征图、匹配结果
$\boldsymbol{b},\boldsymbol{r},\boldsymbol{c}$	$\boldsymbol{B},\boldsymbol{R}$ 和 $\boldsymbol{C}$ 的 Winograd 分块
W_b,W_r,W_c	$\boldsymbol{B},\boldsymbol{R}$ 和 $\boldsymbol{C}$ 的宽度，$W_c=W_b-W_r+1$
H_b,H_r,H_c	$\boldsymbol{B},\boldsymbol{R}$ 和 $\boldsymbol{C}$ 的高度，$H_c=H_b-H_r+1$
w_b,w_r,w_c	$\boldsymbol{B},\boldsymbol{R}$ 和 $\boldsymbol{C}$ 在列方向的分块个数
h_b,h_r,h_c	$\boldsymbol{B},\boldsymbol{R}$ 和 $\boldsymbol{C}$ 在行方向的分块个数
m,n,k	$\boldsymbol{b},\boldsymbol{r}$ 和 $\boldsymbol{c}$ 的尺寸
N,n	$\boldsymbol{B}$ 和 $\boldsymbol{R}$ 的通道数及其索引
$\boldsymbol{H},\boldsymbol{G},\boldsymbol{A}$	$\boldsymbol{B},\boldsymbol{R}$ 和 $\boldsymbol{C}$ 的 Winograd 变换矩阵

二维 Winograd 算法 $F(m^2,k^2)$的计算过程可表示为：

$$\boldsymbol{c}=\boldsymbol{W}(\boldsymbol{r},\boldsymbol{b})=\boldsymbol{A}^{\mathrm{T}}[[\boldsymbol{G}r\boldsymbol{G}^{\mathrm{T}}]\odot[\boldsymbol{H}^{\mathrm{T}}b\boldsymbol{H}]]A \tag{3}$$

即 $\boldsymbol{r}$ 和 $\boldsymbol{b}$ 分别经过 $\boldsymbol{G}$ 变换和 $\boldsymbol{H}$ 变换后进行 EWMM 运算，计算结果经 $\boldsymbol{A}$ 变换后得到 $\boldsymbol{c}$。$F(m^2,k^2)$的加速比可表示为：

$$\eta_w=\frac{m^2\times k^2}{n^2}=\frac{m^2\times k^2}{(m+k-1)^2} \tag{4}$$

其中，m^2k^2和 n^2 分别为使用 Winograd 算法前后 GEMM 和 EWMM 方式下的乘法数。

$F(2^2,3^2)$的转换矩阵通常取值为：

$$\boldsymbol{A}=\begin{bmatrix}1, & 0\\ 1, & 1\\ 1, & -1\\ 0, & 1\end{bmatrix},\boldsymbol{H}=\begin{bmatrix}1, & 0, & 0, & 0\\ 0, & 1, & -1, & -1\\ -1, & 1, & 1, & 0\\ 0, & 0, & 0, & 1\end{bmatrix},$$

$$\boldsymbol{G}=\begin{bmatrix}1, & 0, & 0\\ \frac{1}{2}, & \frac{1}{2}, & \frac{1}{2}\\ \frac{1}{2}, & -\frac{1}{2}, & \frac{1}{2}\\ 0, & 0, & 1\end{bmatrix} \tag{5}$$

其中转换矩阵只包含 0，±1 和 ±1/2，可用加法、移位等低代价逻辑实现。另外，由式(4)可得 $F(2^2,3^2)$的加速比达 2.25，故选用 $F(2^2,3^2)$作为 CSA 的加速方案。

利用 $F(2^2,3^2)$计算特征匹配层的具体过程为：

$$c_{i',j'}=\sum_{n=0}^{N-1}\sum_{u'=0}^{h_r-1}\sum_{v'=0}^{w_r-1}\boldsymbol{W}(b_{i'+u',j'+v',n},r_{u',v',n}) \tag{6}$$

其中，$u'\in[0,h_r)$和 $v'\in[0,w_r)$是 $\boldsymbol{R}$ 的行、列方向块索引；$i'\in[0,R_C)$和 $j'\in[0,C_C)$是 $\boldsymbol{C}$ 的行、列方向块索引；则 $i'+u'\in[0,R_B)$和 $j'+v'\in[0,C_B)$是 $\boldsymbol{B}$ 的行、列方向块索引。即 $\boldsymbol{B}$ 和 $\boldsymbol{R}$ 按照如图 2 所示的方式分解，各分块经 Winograd 计算后在相关窗口内累加后得到匹配结果分块 $\boldsymbol{c}$。

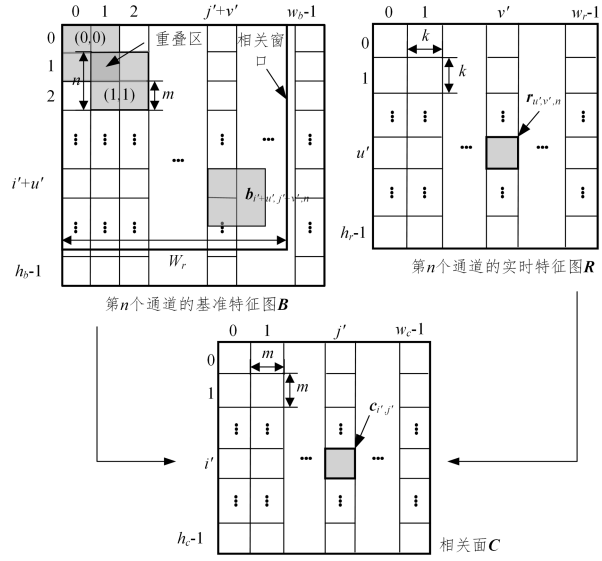

图 2 特征图 $\boldsymbol{B}$，$\boldsymbol{R}$ 与匹配结果 $\boldsymbol{C}$ 的分解方式

Fig. 2 Decomposition of $\boldsymbol{B}$，$\boldsymbol{R}$ and matching result $\boldsymbol{C}$

4 加速方案设计

4.1 流水线式加速计算方案设计

CNN 景象匹配算法的计算过程可分为基准图的特征提取过程、实时图的特征提取过程和特征匹配过程，它们之间的耦合度较低，可采用如图 3 所示的流程加速计算。基准图可在景象匹配任务前确定并在任务中保持不变，因此可在初始化过程中完成基准特征图 $\boldsymbol{B}$ 的计算，并一直保存于片外存储 DDR 中供实时推理时调用。$\boldsymbol{R}$ 和 $\boldsymbol{C}$ 则在实时推理时分别由 DPU 和 COR 流水线式计算得到，系统的吞吐速率满足：

$$S=1/\max(T_{\mathrm{R}},T_{\mathrm{C}}) \tag{7}$$

其中，T_{R}，T_{C} 分别表示计算 $\boldsymbol{R}$，$\boldsymbol{C}$ 所需的时间。通过分析发现，有两种途径可提升系统整体的吞吐速率：1)同时提升 DPU

和 CSA 的性能，降低 T_R 和 T_C；2）通过超参数配置，使得 DPU 和 CSA 的实际算力之比与实时图特征提取过程和特征匹配过程的计算量之比相匹配，即最小化空闲时间 $|T_R - T_C|$。

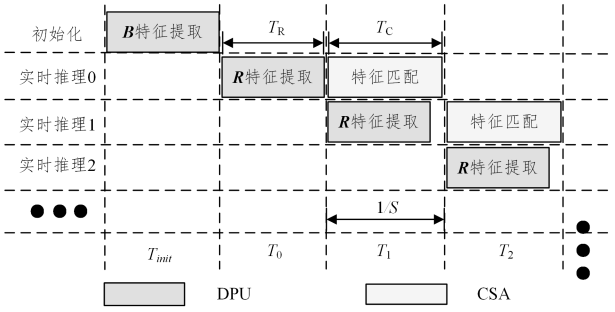

图 3 CNN 景象匹配算法的计算流程图

Fig. 3 Computation flowchart of CNN scene matching algorithm

4.2 特征匹配层专用加速器设计

特征匹配层专用加速器（CSA）的整体架构如图 4 所示，由控制单元、Winograd 加速器核心（Winograd Accelerator Core，WAC）、数据传输单元和 DDR 组成。控制单元具有轻量级控制和指令分发功能，可通过 AXI-Lite 总线与主控端交互，实现 CSA 的运行时功能配置和状态查询，从而提高 CSA 的通用性和灵活性，使 CSA 成为一个易用高效的协处理器。

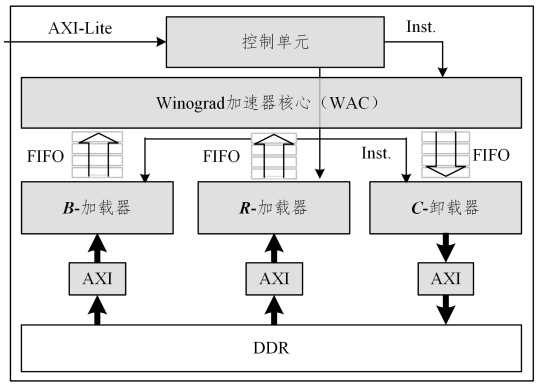

图 4 特征匹配层专用加速器的整体架构

Fig. 4 Architecture of correlation specific accelerator

数据传输单元由 ***B***-加载器、***R***-加载器和 ***C***-卸载器组成，可根据控制单元的指令自主完成各类数据的传输。它们通过 AXI 总线与 DDR 进行数据交互，通过流式结构 FIFO 与 WAC 连接，具备数据缓冲功能的同时实现了数据传输与计算的解耦。它们可以通过感知 AXI 总线中 VALID 信号的占空比以及 FIFO 中数据的变化速率，实现数据的自适应传输，最大化地利用有限 DDR 带宽。

WAC 的主要功能是根据控制单元的指令，从上级 FIFO 获取数据，基于 Winograd 算法完成特征匹配层的加速计算，并将结果打入下级 FIFO。其内部结构如图 5 所示，由片上缓存（On-Chip Memory，OCM）和 p 通道处理单元（p-channel Processing Element，p-PE）组成。p-PE 是 WAC 的计算引擎，其设计难点在于用尽可能简单的硬件逻辑实现式（6）所示的计算过程。p-PE 采用 4 级流水线设计：第一级流水并行完成 ***b*** 和 ***r*** 的变换，后面三级流水线分别完成 EWMM、p 通道数据累加和 ***y*** 的反变换。

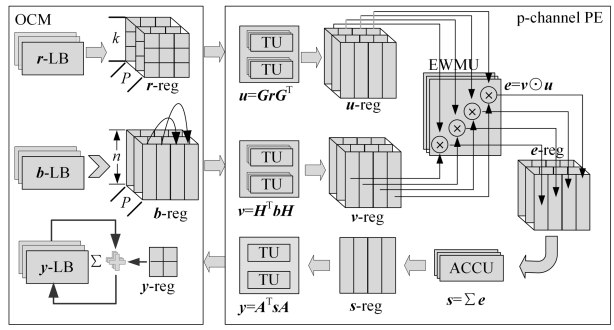

图 5 Winograd 加速器核心的内部结构

Fig. 5 Structure of Winograd accelerator core

数据转换单元（Transformation Units，TUs）包含 $\boldsymbol{u}=\boldsymbol{GrG}^{\mathrm{T}}$，$\boldsymbol{v}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{bB}$ 和 $\boldsymbol{y}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{sA}$ 共 3 类 Winograd 数据转换。该转换过程本质上是线性变换，每个输出元素均为输入元素的线性组合，系数由转换矩阵决定。如式（5）所示，转换矩阵具有一定的稀疏性，使得输出元素仅与少量输入元素相关，如每个 ***v*** 元素仅与 4 个 ***b*** 元素相关。转换矩阵的简洁性可进一步降低 TUs 的复杂度。若矩阵 ***H*** 和 ***A*** 中仅包含 0 和 ±1，则可用加法实现 ***v*** 和 ***y*** 的变换；***G*** 中增加了 $\pm1/2$，则可用加法和移位组合实现 ***r*** 的变换。此外，在 p-PE 的实现中，等效微调了 Winograd 算法的计算过程，即将 p 通道累加过程前置、***y*** 反变换过程后置。上述两个过程的加法总数在微调前后的比值为：

$$\eta_p=\frac{m^2px+m^2p}{m^2x+n^2p} \tag{8}$$

其中，x 为 ***y*** 反变换每个元素所需的加法数，对于式（5）所示的矩阵 ***A***，$x=8$；m^2px 和 m^2x 分别为微调前后 ***y*** 反变换所需的加法数；m^2p 和 n^2p 分别为微调前后 p 通道累加所需的加法数。将 $x=8$ 带入式（8）可得：

$$\eta_p=\frac{36p}{32+16p} \tag{9}$$

通过分析发现，当输入通道并行度 $p\geqslant2$ 时，上述微调方法便可降低 p-PE 的复杂度；当 $p=32$ 时，复杂度降低到微调前的 1/2 以下。

4.3 片上缓存系统设计

p-PE 的峰值算力仅取决于时钟频率和计算并行度，而实际算力则很大程度取决于片上缓存系统（OCM）能否在指定时间内提供 p-PE 所需的数据。其设计难点在于用尽可能少的片上存储资源满足 p-PE 的带宽需求，并最大化片上数据的复用率。如图 6 所示，OCM 由 3 个独立的局部缓存（Local Buffer，LB）组成，***b***-LB，***r***-LB 和 ***y***-LB 分别用于缓存基准特征图、实时特征图和中间结果数据。

p-PE 的 ***r*** 数据带宽需求为：

$$B_r=\frac{k^2p}{w_c} \tag{10}$$

其中，k^2p 表示每次更新的数据量，w_c 表示 ***r***-reg 数据的重用次数，与 ***C*** 的列方向分块个数相等。***r***-LB 由 k 个独立真双口 RAM 组成，每个 RAM 可同时读写 p 个数据。则 k 个时钟周期便可完成 1 次 ***r***-reg 更新，一般情况下 w_c 远大于 k，更新时间对整体的计算效率影响较小。

为满足 p-PE 的 ***r*** 数据带宽需求 m^2/clock，***r***-LB 由 m^2 个

独立真双口 RAM 组成，每个 RAM 可同时读写 1 个数据。

p-PE 的 ***b*** 数据带宽需求最大，为 n^2p/clock。直接设计相同带宽的片上缓存会使得片上存储资源和布线资源紧张。我们利用 ***b*** 分块之间在行、列方向均具有 $n-m$ 大小重叠区(见图 2)的特性，设计了由 Line-Buffer 和滑窗寄存器组成的基准特征图片上缓存系统，具体如图 6 所示。***b***-LB 由 m 个 Line-Buffer 组成，每个 Line-Buffer 包含 $m+n$ 个带宽为 p 的单口 RAM。***b***-block 的第 w 列数据存储于 Line-Buffer[$w\%m$]，第 i 个 ***b***-block 的第 h 行存储于 LB[0:$m-1$][$(i\times m+h)\%(n+m)$]。Line-Buffer 可利用 ***b*** 分块行方向的重叠区，降低$(n-m)/n$ 的 ***B*** 数据片外带宽需求。m 个 Line-Buffer 可向 ***b***-reg 提供 mnp/clock 的带宽，剩余$(n-m)np$/clock 的带宽由滑窗寄存器内部实现。即利用 ***b*** 分块列方向的重叠区，将前一时刻 ***b***-reg[m:$n-1$]的数据直接转存至当前时刻的 ***b***-reg[0:$n-m-1$]。

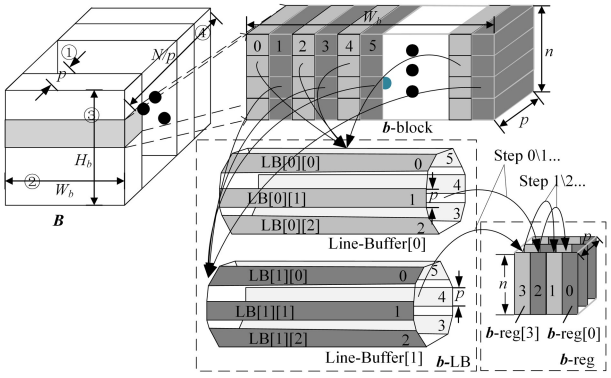

图 6 基准特征图的片上缓存系统

Fig. 6 On-chip memory of basic feature map

5 实验及结果分析

5.1 实验环境与设置

为验证加速计算方案的有效性以及 CSA 的高效性，本文基于 Xilinx ZCU102 开发板实现了整套实验验证系统。使用 Vivado HLS 2019.2 完成 CSA 的设计、仿真和综合，得到由 Verilog 实现的 IP。扩展设计成熟 DPU[21-22]，在 Vivado 2019.2 中按照 2.1 节所述方案将 CSA 和 DPU 集成为一种可流水线式工作的加速计算系统。在 SDK 2019.2 中完成使用程序开发，实现加速计算系统的正确性验证和性能测试。

DPU 与 CSA 的超参数设置分别如表 2 和表 3 所列。

表 2 CSA 的超参数

Table 2 Hyper-parameters of CSA

分类	名称	取值
数据类型	基准特征图 ***B***	INT8
	实时特征图 ***R***	INT8
	匹配结果 ***C***	INT32
并行度	输入通道的并行度 p	32
	特征图内的并行度 m^2	4
	卷积核内的并行度 k^2	9
片上缓存	局部缓存 ***b***-LB 的深度	512
	局部缓存 ***r***-LB 的深度	512
	局部缓存 ***y***-LB 的深度	4508
WAC 与数据传输单元间的 FIFO 深度		64

表 3 DPU 的超参数

Table 3 Hyper-parameters of DPU

分类	名称	取值
数据类型	输入/输出特征图	INT8
	权重和偏置	INT8
并行度	输入通道的并行度 m	32
	输出通道的并行度 n	32
	加速器核心的数量 s	2
片上缓存	输入特征缓存 Fbuf 的深度	5120
	权重缓存 Wbuf 的深度	5120
	中间结果缓存 Cbuf 的深度	5120
FIFO	Big-FIFO 深度	1024
	DF-FIFO 深度	16

在 250MHz 的时钟频率下 DPU 与 CSA 的峰值算力分别为1024GOPS 和 576GOPS，特征提取网络和特征匹配层的计算量分别为 73×10^9 和 70×10^9。

作为对照实验，我们部署 Xilinx 官方的深度学习处理器 DPUCZDX8G[27]，串行计算 CNN 景象匹配算法。DPUCZDX8G 的超参数如表 4 所列，包含 3 个架构为 B4096 的 DPU 核心，整体的计算并行度为 12288 Operation/clock。在 333.3MHz 的时钟频率下峰值算力为 4095.96GOPS。

表 4 DPUCZDX8G 的超参数设置

Table 4 Hyper-parameters settings of DPUCZDX8G

分类	名称	取值
数据类型	输入/输出特征图	INT8
	权重和偏置	INT8
并行度	特征图内的并行度(PP)	8
	输入通道的并行度(ICP)	16
	输出通道的并行度(OCP)	16
	加速器核心的数量	3
约束	卷积核的尺寸	小于等于 16
	输入通道数	小于等于 256×ICP
	输出通道数	小于等于 256×OCP

5.2 实验结果与对比分析

加速计算系统(DPU+CSA)在 ZCU102 上的 FPGA 资源消耗量如表 5 所列，各类消耗量均小于 DPUCZDX8G。通过观察发现，在 CSA 中 Winograd 加速器核心(WAC)作为主体单元消耗了大部分资源，控制单元、数据传输单元以及各单元之间的 FIFO 等配套部分的各类资源消耗量均在 3%以内。这说明将控制、通信与计算解耦的设计方法，能够以很小的 FPGA 资源消耗为代价，实现简化系统设计复杂性、提高系统灵活性和通用性以及提高片外存储带宽利用率的目的。

表 5 FPGA 资源消耗量

Table 5 FPGA resource consumption

系统	DSP	FF	BRAM	LUT
WAC/%	20.75	3.26	22.92	12.00
CSA/%	**20.99**	**4.66**	**25.88**	**13.49**
DPU/%	42.66	16.70	42.98	37.27
CSA+DPU/%	**63.65**	**21.35**	**68.86**	**50.76**
DPUCZDX8G/%	84.52	54.08	83.88	56.21
资源总量	2520	548160	912	274080

DPUCZDX8G 串行计算特征提取网络和特征匹配层的测试结果如表 6 所列。基于卷积核的最大尺寸和输出通道并行度约束(见表 4)，在计算过程中需将特征匹配层转换成 64 个卷积核大小为 16×16、输出通道数为 16 的卷积层，并对卷积

结果累加得到最终的匹配结果。

表 6 DPUCZDX8G 的计算性能

Table 6 Computation performance of DPUCZDX8G

计算任务	计算量	吞吐时延/毫秒每帧	实际算力/GOPS	计算效率/%
特征提取	73.41×109	**62.38**	1176.77	28.73
特征匹配	69.80×109	**632.69**	110.32	2.69
提取＋匹配	143.21×109	**695.08**	206.04	5.03

DPU 与 CSA 的实际性能测试结果如表 7 所列。DPU 和 CSA 的吞吐延时分别为 149.56 毫秒每帧和 157.89 毫秒每帧，系统的吞吐速率达 6.33 帧每秒，流水线上的空泡时间仅有 8.33 毫秒每帧，不足系统吞吐时间的 5.28%。这说明 DPU 和 CSA 的实际算力之比与实时图特征提取过程和特征匹配过程的计算量之比非常匹配。与 DPUCZDX8G 的串行计算方案相比，CSA 的特征匹配层的计算速度增长了 4 倍以上，DPU＋CSA 将景象匹配算法的整体计算性能提高了 4.40 倍以上。

表 7 CSA 与 DPU 的计算性能

Table 7 Computation performance of CSA and DPU

	DPU	CSA	DPU＋CSA
计算任务	特征提取	特征匹配	提取＋匹配
计算量	73.41×10^9	69.80×10^9	143.21×10^9
吞吐时延/毫秒每帧	**149.56**	**157.89**	**157.89**
峰值算力/GOPS	1024	576	1600
实际算力/GOPS	490.80	442.08	907.02
计算效率/%	47.93	76.75	56.69

5.3 计算效率分析

由表 3 可知，DPU 的计算并行度为 4096 Operation/clock，共有 1024 个 DSP48E 用于计算，DSP 的使用效率达 4 Operation/clock。DPU[21-22]在设计中使用了“多输出通道并行计算”的方式，在该并行方式下多组卷积核共用 1 组输入特征，因此可基于加法的结合律利用 1 个 DSP48E 完成 2 个输出通道的计算，即 1 个 DSP48E 并行计算 2 组乘加操作。在 CSA 设计中，由于特征匹配层的输出通道为 1，因此上述提高 DSP48E 使用效率的方法难以适用。在使用 Winograd 算法降低特征匹配层计算复杂度的情况下，CSA 的等效并行度达 2304 Operation/clock，共有 512 个 DSP48E 用于计算，DSP 的使用效率达 4.5 Operation/clock。这说明 Winograd 算法是提升 DSP 使用效率最有效的手段之一，可降低 FPGA 计算资源对加速器性能的限制。

如表 7 所列，经扩展设计和优化裁剪后的 DPU 计算特征提取网络的效率依然不足 48%，主要是因为特征提取网络中存在大量特殊算子，包括全局平均池化、深度分离卷积、shortcut、一维卷积、Sigmoid 等，使得 DPU 的实际计算效率较低。相对而言，特征匹配层的计算过程非常规整，但 CSA 的计算效率只达到 76.75%。通过分析发现，*r*-reg 的数据更新过程及其导致的 *p*-PE 流水线断裂是 CSA 效率损失的主要原因，我们将在后续研究中继续完善。

结束语 针对 CNN 景象匹配算法的结构特点，本文提出利用 DPU 和 CSA 流水线式计算特征提取网络和特征匹配层的整体加速计算方案。针对特征匹配层计算密集的特性，提出了基于 Winograd 算法设计特征匹配层专用加速器（CSA），以解决 FPGA 的计算资源紧张的问题。基于 Xilinx ZCU102 开发板构建实验系统，用于验证本文方案。结果表明，流水线式加速方案的性能是 DPUCZDX8G 串行计算方案的 4.40 倍以上，CSA 的计算效率达 76.75% 以上，单个 DSP48E 的使用效率达 4.5 Operation/clock，充分验证了本文加速计算方案的有效性和 CSA 的高效性。

参 考 文 献

[1] SIMONYAN K，ZISSERMAN A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. arXiv：1409.1556，2014.

[2] 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. 2016：770-778.

[3] TAN M，LE Q. Efficientnet：Rethinking Model Scaling for Convolutional Neural Networks[C]// International Conference on Machine Learning. PMLR，2019：6105-6114.

[4] REN S，HE K，GIRSHICK R，et al. Faster R-CNN：Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence，2016，39(6)：1137-1149.

[5] BOCHKOVSKIY A，WANG C Y，LIAO H Y M. Yolov4：Optimal Speed and Accuracy of Object Detection[J]. arXiv：2004.10934，2020.

[6] TAN M，PANG R，LE Q V. Efficientdet：Scalable and Efficient Object Detection[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020：10781-10790.

[7] LIU W，ANGUELOV D，ERHAN D，et al. SSD：Single Shot Multibox Detector[C]// European Confe-rence on Computer Vision. Cham：Springer，2016：21-37.

[8] HE K，GKIOXARI G，DOLLÁR P，et al. Mask R-CNN[C]// Proceedings of the IEEE International Conference on Computer Vision. 2017：2961-2969.

[9] SUN P，ZHANG R，JIANG Y，et al. Sparse R-CNN：End-to-End Object Detection with Learnable Proposals[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021：14454-14463.

[10] REN S H，CHANG W G，LIU X J. A Scene Matching Algorithm based on Wavelet Transform and Variable Scale Circle Template Fusion[J]. Acta Electronica Sinica，2011，39(9)：2200-2203.

[11] BO L F，HAN J，ZHANG Y，et al. Infrared and Visible Image Registration Algorithm using Improved Gradient Mutual Information and Particle Swarm Optimization Algorithm[J]. Infrared and Laser Engineering，2012，41(1)：248-254.

[12] CAO Z G，WU B. The Down-View Scene Matching Algorithm using HOG Features[J]. Infrared and Laser Engineering，2012，41(2)：513-516.

[13] ALEKSANDRA S，SIMON B. Optimizing SIFT for Matching of Short Wave Infrared and Visible Wavelength Images[J]. Re-

mote Sensing, 2013, 5(5): 2037-2056.
[14] CHEN T, DU Z, SUN N, et al. Diannao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine Learning [J]. ACM SIGARCH Computer Architecture News, 2014, 42(1): 269-284.
[15] JOUPPI N P, YOUNG C, PATIL N, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit [C]// Proceedings of the 44th Annual International Symposium on Computer Architecture. 2017: 1-12.
[16] CHEN Y H, KRISHNA T, EMER J S, et al. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks [J]. IEEE Journal of Solid-State Circuits, 2016, 52(1): 127-138.
[17] WILLIAMS S, WATERMAN A, PATTER-SON D A. Roofline: An Insightful Visual Performance Model for Multicore Architectures [J]. Communications of the ACM, 2009, 52(4): 65-76.
[18] ZHANG C, LI P, SUN G, et al. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks [C]// Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2015: 161-170.
[19] GUO K, SUI L, QIU J, et al. Angel-eye: A Complete Design Flow for Mapping CNN onto Customized Hardware [C]// 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2016: 24-29.
[20] WANG X F, JIANG P L, ZHOU H, et al. High Parallelism FPGA Accelerator Design for Convolutional Neural Networks [J]. Journal of Computer Applications, 2021, 41(3): 812-819.
[21] WANG X, GE Y, GAO Y, et al. A More Scalable Deep-Learning Processing Unit for Depthwise Separable Convolution [C]// 2021 6th International Conference on Integrated Circuits and Microsystems (ICICM). IEEE, 2021: 285-290.
[22] WANG X, LIU G, GE Y, et al. A More Efficient Deep-Learning Processing Unit Architecture with Runtime Configurable Parallelism [C]// 2021 China Automation Congress (CAC). IEEE, 2021: 5941-5945.
[23] LU L, LIANG Y, XIAO Q, et al. Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs [C]// 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2017: 101-108.
[24] SHEN J, HUANG Y, WANG Z, et al. Towards a Uniform Template-Based Architecture for Accelerating 2D and 3D CNNs on FPGA [C]// Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2018: 97-106.
[25] LU L, LIANG Y. SpWA: An Efficient Sparse Winograd Convolutional Neural Networks Accelerator on FPGAs [C]// Proceedings s of the 55th Annual Design Automation Conference. 2018: 1-6.
[26] LAVIN A, GRAY S. Fast Algorithms for Convolutional Neural Networks [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 4013-4021.
[27] XILINX. DPUCZDX8G for Zynq UltraScale+ MPSoCs Product Guide (PG338) [EB/OL]. (2022-06-24) [2022-12-07]. https://docs. xilinx. com/r/en-US/pg338-dpu.

WANG Xiaofeng, born in 1995, master, engineer. His main research interest is intelligent computing.

(责任编辑:喻藜)