基于 FPGA 的 YOLOv4-tiny 网络的硬件加速与实现

李 雷,黎远松,石 睿
四川轻化工大学 计算机科学与工程学院,四川 宜宾 643000

摘 要:目的 为解决在资源受限的情况下,目标检测算法在边缘硬件平台中提高计算性能和能效比,提出了一种基于现场可编程门阵列(Field Programmable Gate Array, FPGA)的边缘硬件平台实现对 YOLOv4-tiny 网络的加速设计并进行验证。**方法** 采用了高层次综合技术(High level Synthesis)对算法的算子和模块进行了高度并行的设计与优化。为提高设计吞吐量,采用了双缓冲策略增加系统资源利用率,并利用卷积层与 BN(Batch Normalization)层融合和量化模型技术,减少了模型参数量,提高了计算密度。**结果** 在 PYNQ-Z2 平台上进行实验,结果表明:加速器的计算性能为 15.33 GOPS,总功耗为 2.65 W,相较于同类研究的 FPGA 平台计算性能提高了 2.79 倍,相较于 CPU 平台的能效比提高了 29.5 倍。**结论** 对 YOLOv4-tiny 网络在边缘 FPGA 平台加速效果有所提升,为目标检测算法在硬件平台的加速研究提供了参考。

关键词:现场可编程门阵列;高层次综合;YOLO;硬件加速
中图分类号:TP391.4 **文献标识码:**A doi:10.16055/j.issn.1672-058X.2026.0003.006

Hardware Acceleration and Implementation of YOLOv4-Tiny Network Based on FPGA

LI Lei, LI Yuansong, SHI Rui
School of Computer Science and Engineering, Sichuan University of Science and Engineering, Yibin 643000, Sichuan, China

Abstract: Objective To enhance the computational performance and energy efficiency ratio of object detection algorithms on edge hardware platforms under resource-constrained conditions, this paper proposes and verifies an edge hardware platform based on field programmable gate array (FPGA) for accelerating the YOLOv4-tiny network. **Methods** High-level synthesis (HLS) was used to design and optimize the operators and modules of the algorithm in a highly parallel manner. To improve design throughput, a double-buffering strategy was adopted to increase system resource utilization. Additionally, techniques such as fusion of convolutional and batch normalization (BN) layers and model quantization were applied to reduce model parameters and enhance computational density. **Results** Experiments conducted on the PYNQ-Z2 platform demonstrate that the accelerator achieved a computational performance of 15.33 GOPS with a total power consumption of 2.65 W. Compared to existing FPGA platforms, the proposed design improved computational performance by 2.79 times, while achieving a 29.5-fold increase in energy efficiency ratio compared to CPU platforms. **Conclusion** The proposed method effectively enhances the acceleration of the YOLOv4-tiny network on edge FPGA platforms, providing a valuable reference for the acceleration research of object detection algorithms on hardware platforms.

Keywords: field programmable gate array; high-level synthesis; YOLO; hardware acceleration

收稿日期:2024-06-04 **修回日期:**2024-09-24 **文章编号:**1672-058X(2026)03-0045-08
基金项目:国家自然科学基金项目资助(42074218).
作者简介:李雷(2001—),男,四川达州人,硕士研究生,从事目标检测算法研究. Email: 2780678468@qq.com.
通信作者:黎远松(1970—),男,教授,硕士生导师,从事自然语言处理研究. Email: yuansongli@suse.edu.cn.
引用格式:李雷,黎远松,石睿. 基于 FPGA 的 YOLOv4-tiny 网络的硬件加速与实现[J]. 重庆工商大学学报(自然科学版),2026,43(3):45-52.
Li Lei, Li Yuansong, Shi Rui. Hardware acceleration and implementation of YOLOv4-tiny network based on FPGA[J]. Journal of Chongqing Technology and Business University (Natural Science Edition), 2026, 43(3): 45-52.

在当今的科技世界中,目标检测技术在各个领域中扮演着越来越重要的角色,特别是在人脸识别、自动驾驶汽车、工业视觉检测等应用中[1-3]。然而,神经网络的运行需要大量的计算资源,这对现有的计算平台提出了极高的要求。为此,研究者们开展了深入广泛的研究,试图优化神经网络的运行效率,主流 CPU、GPU 的方案[4-5]较为成熟,有较高的计算性能,但是往往能耗比较高。

现场可编程门阵列(Field Programmable Gate Array, FPGA)拥有高度的重配置性和并行处理能力,能够同时处理多个运算单元和多个数据并行操作。FPGA 与卷积神经网络(CNN)的结合,有助于提升 CNN 的部署效率和性能。由于 FPGA 低功耗特性进一步增强了其吸引力。此外,FPGA 可以根据具体算法需求量身打造硬件加速器。针对动态深度神经网络在边缘计算中的部署,FPGA 展现出了良好的适应性。例如,Li 等[6]提出了一种由多个 FPGA 构成的集群架构,以提高处理吞吐率和计算性能。尽管这样的设计在某些情况下效果显著,但多 FPGA 的协同工作需求并不完全符合边缘计算所要求的轻量化与独立运行特点。Liu[7]将 Winograd 算法[8]引入 FPGA,并提出了结合 Winograd 算法与脉动阵列的架构方案。该架构依赖于一个复杂的内存管理子系统,并且 Winograd 算法只能应用于卷积步长单一的操作,限制了其通用性。Lu 等[9]采用软硬件协同优化的策略,开发了一个 FPGA 加速器。将移位量化算法与 FPGA 结合使用,然而所提出的结构基于传统的 CNN 模型。对于新兴的轻量级 CNN 模型,这一架构可能无法有效支持。尽管 FPGA 在卷积神经网络加速领域具有明显优势,但现有研究依旧存在需求专门设计和适应性局限的问题,这要求未来的研究更加注重通用性和灵活性的提升。

在此背景下,本文选用了轻量化神经网络 YOLOv4-tiny 作为目标模型,YOLOv4-tiny 作为 YOLOv4 的小型化版本,YOLOv4-tiny 网络拥有优良的目标检测能力,但所需的计算力仍然较高,并且由于存储空间、能耗以及带宽等资源限制,在嵌入式领域中的存储与计算依旧是一个很大的挑战。从硬件层面来看,因为神经网络的大部分计算都在卷积上,常规 CPU 难以满足神经网络算法的实用性,并行架构的 FPGA 能更好地适配神经网络,而 GPU 虽然有优秀的并行计算能力和效率,但功耗太高,容易造成资源浪费,所以使用 FPGA 进行加速是较好的选择。

本文采用了定点 16 位量化模型,保留了原模型的高效性,同时大幅减少了计算资源的消耗,适用于边缘计算平台[10]。使用了高层次综合技术(High-level Synthesis)[11]进行设计和优化,并在 FPGA 平台上进行实现,从简化模型、并行化[12]和流水化[13]三个角度对模型进行硬件加速设计,分别设计了整体系统架构、卷积 IP 模块、池化 IP 模型和对应算子,提高系统的性能和降低了系统的能耗。使用 Xilinx Vivado 开发工具,在 Vivado 集成开发环境(IDE)中,成功搭建了系统级芯片(SoC)架构,在 PYNQ-Z2 开发板上对所研究的设计进行验证,并对其设计进行全面的性能评估。本文方法提高了 YOLOv4-tiny 网络在边缘硬件 FPGA 平台的计算性能、资源利用率以及能耗比等关键指标。

1 相关技术

1.1 YOLOv4-tiny

YOLO(You Only Look Once)算法[14]能同时进行目标识别和定位,显著加快了检测速率。此算法通过将检测任务转换为回归问题,有效克服了传统目标检测算法(如 R-CNN)的一些主要问题,包括慢速运行和优化难度等。YOLOv4-tiny 与 YOLOv4 模型各方面性能的比较如表 1 所示,其中 YOLOv4-tiny 的均值平均精度(Mean Average Precision,mAP)略低于 YOLOv4,是因为 YOLOv4-tiny 的网络层数比较浅,一共只有 38 层,但是正因如此 YOLOv4-tiny 模型处理图片速率的 FPS 是 YOLOv4 的数倍。YOLOv4-tiny 训练出来的模型参数量只有 23.1 MB,相较于 YOLOv4 得到了极大的缩小,更加有利于在边缘硬件平台实现加速。

表 1 参数比较

Table 1 Parameter comparison

Model	mAP/%	FPS	GOPS	Params/MB
YOLOv4-tiny	40.2	371	6.9	23.1
YOLOv4	43.5	65	198	245

YOLOv4-tiny 的网络结构主要有三部分:主干网络(CSP-Marknet53-tiny)、颈部特征金字塔网络(Feature Pyramid Networks,FPN)和两个 YOLO 检测头组成,如图 1 所示。输入的特征图尺寸为 416×416×3,最终输出有 26×26×255 和 13×13×255 两个计算结果。

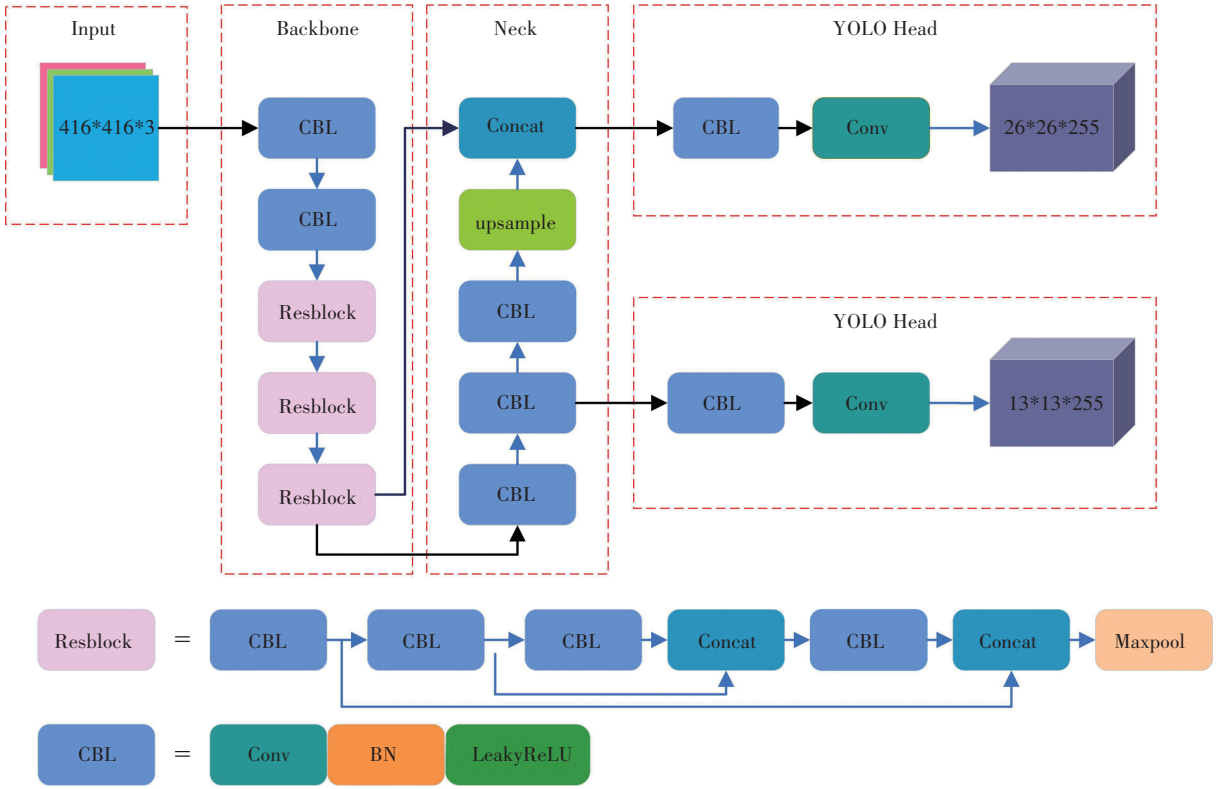

图 1　YOLOv4-tiny 网络结构

Fig. 1　YOLOv4-tiny network structure

主干网络的 CSPdarknet53_tiny,相较于 YOLOv4 的主干网络 CSPdarknet53,在 CBL(Conv + BN + LeakyReLU)层将 Mish 激活函数修改为更迅速的 LeakyReLU 函数,卷积模块包含 3×3 标准卷积和 1×1 点卷积。Resblock 模块采用了双重残差网络架构,减少了网络的层数,同时提高了性能。末尾是 Maxpool2d 的池化层来降低特征图的大小,以增加感受野并减少计算量。颈部的 FPN 添加了一个为 SPP(Spatial Pyramid Pooling)的模块,可以捕获不同尺度的特征,这有助于模型检测不同大小的目标,并且使用了 Upsample 和特征层融合技术提高特征图中信息和细节,有利于模型对目标识别和检测的能力。检测头采用了轻量化的设计,由两个卷积层构成,对一个网格可生成三个预测边界框,通过非极大值抑制(Non-Maximum Suppression,NMS)[15]选出最好的预测框,提升了模型能够检测到物体的数量和精确度。

1.2　双缓冲策略

对于边缘计算平台,板卡上的资源有限并且带宽也受到限制,无法将所有数据都缓存到片上缓冲区,想要实现高性能的设计,双缓冲策略[16]也称乒乓策略对硬件来说尤为重要。本文正是采用了这种设计,双缓冲策略就是利用并行处理的方法,当其中一个缓冲区正在加载数据时,另一个缓冲区可以同时计算已加载的数据。这样,加载和计算操作可以在互相不干扰的情况下几乎同时进行,从而提高系统的效率,减少了硬件资源消耗。双缓冲与单缓冲设计的处理流程对比如图 2 所示。在只有一个缓冲区时,无法同时进行读写数据和计算操作,只能串行执行。必须等待输入特征图读取到可编程逻辑(PL)上的缓冲区后,才能对其中存储的数据进行后续计算。然而,如果有两个输入缓冲区,在第一次计算开始时,一个缓冲区中对已加载的数据进行计算,同时可以将第二份数据读取到另一个缓冲区中。

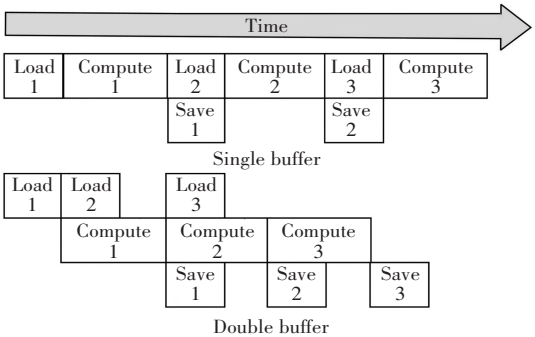

图 2　双缓冲与单缓冲对比图

Fig. 2　Double buffer versus single buffer

一般情况下,计算(Compute)所需的时间大于加载(Load)时间,因此这种方式可以覆盖加载时间,从而实现了并行和连续的数据处理流程。节省的加载时间就越多,对数据的传输和处理速率的提升也就越大。

1.3 高层次综合简介

高层次综合(High-level Synthesis,HLS)是一种通过 Xilinx 的 Vivado HLS 工具实现的技术,它可以将高抽象级语言(如 C、C++)转化为较低抽象级的硬件描述语言(如 Verilog、VHDL)。这种将语言逻辑转换寄存器传输级硬件电路模型的技术,打破了传统上仅使用硬件描述语言进行 FPGA 设计的局限。并且,HLS 的代码更加清晰易懂、便于优化调整,同时简化了各种接口协议的实现,并提高了对计算单元的复用性。此外,HLS 还能对于复杂的数字信号处理(DSP)应用和定制硬件开发,开发效率可以得到提升。并且,使用 HLS 进行设计可以更容易地移植到不同的硬件平台上,而不必重新实现底层硬件描述。

HLS 无法像手动优化的硬件描述语言那样有效地利用硬件资源,因此生成的硬件描述可能导致资源利用率不高。通常,HLS 使用的基本 FPGA 的查找表(Look Up Table, LUT)资源比传统的 RTL 多 40%左右,所以对于某些类型的设计,特别是对于具有特定时序要求或需要极低功耗的设计,HLS 可能不是最佳选择。

2 硬件加速器设计

2.1 加速器框架分析

基于对网络结构的分析,有必要将 YOLOv4-tiny 网络模型的计算密集型的算子和资源占用较大的模块放在 FPGA 开发板中,以实现集中加速。YOLOv4-tiny 需要进行设计的算子主要包括有两种步长的 3×3 标准卷积、1×1 点级卷积、上采样和 2×2 最大池化层,其中 BN(Batch Normalization)层可利用层融合的方法加入卷积层当中,Concat 通过指针操作在 PS 端实现更加方便。其中,3×3 标准卷积和 1×1 点级卷积占网络模型中整体网络计算量的 90%以上,因此卷积计算非常需要着重加速。首先,在 FPGA 开发板中,PL 端调用所设计的动态可配置的卷积 IP 模块。其次,在 PS 端构建了 YOLOv4-tiny 网络结构,通过调用本文设计的卷积 IP 模块和最大池化 IP 模块得到计算结果。

整体的加速器架构如图 3 所示,FPGA 平台主要有特征数据输入模块、权重输入模块和输出模块,封装设计的卷积 IP 模块和最大池化 IP 模块,配置 AXI4-Lite 和 AXI4-Master 接口。其中,BRAM 是 FPGA 的一种存储块,用于存储数据或指令。DDR 控制器的任务是管理外部存储器与 AXI 内存总线接口之间的数据传输,并且存储全部的交互数据以及神经网络计算出的最终结果。鉴于训练后的算法模型和待检测图像的权重所占用的空间较大,因此需要首先将它们存储在 SD 卡中,然后再将其加载到 DDR 中进行后续处理。

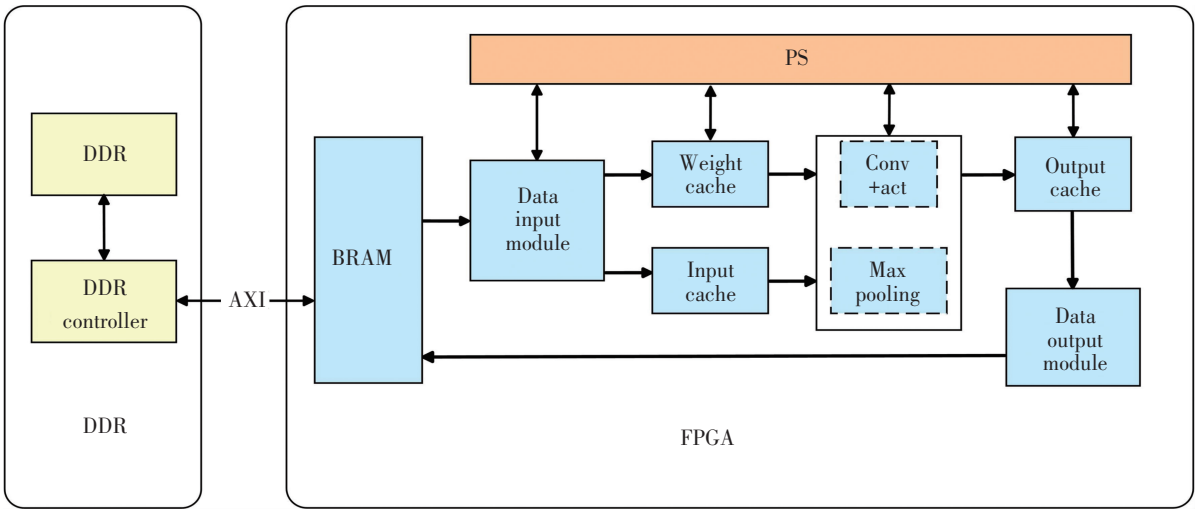

图 3 整体的加速器架构

Fig. 3 Overall accelerator architecture

卷积 IP 核心主要完成算法中的卷积和 LeakyReLU 激活函数的计算。在计算过程中,特征输入数据、权重数据、偏差数据等是相互独立的,数据可以同时收集。因此,可以在加速器中设置输入模块来连接两种 AXI 接口的读取通道,从而提高数据的并行性。而池化计算单元主要完成最大池化操作。BRAM 通过 AXI 内存

总线接口与外部 DDR 进行通信,从外部存储器读取所需的计算输入数据,并将结果保存到输出缓冲区中。然后,通过数据输出模块和 AXI 内存总线接口,将结果写回外部 DDR。卷积 IP 核心和池化 IP 核心通过双缓冲策略使用输入和权重缓冲区,有效节省了硬件存储资源。FPGA 中的一些模块由 PS 进行操作配置和调度。

2.2　卷积层与 BN 层融合

深度神经网络在非线性变换前输入值的分布随着网络深度的增加或在训练过程中逐渐偏移或变化,一般接近非线性函数值区间的上限或下限。这就导致了浅层神经网络的梯度在反向传播的情况下消失,以及诸如训练困难和收敛速度慢等问题。BN(Batch Normalization)层则可以用于解决深度卷积神经网络在网络深化过程中的这类问题。BN 层通过计算网络每一层的均值和方差,对输入数据进行标准化处理,然后再经过缩放和平移操作,最终得到归一化的输出。这个处理过程有助于解决神经网络训练中出现的梯度消失、梯度爆炸等问题,从而加速网络的收敛,提高模型的泛化能力,实现训练的加速。在 YOLOv4-tiny 网络中,几乎所有 BN 层都设置在每个卷积层的后面,这有利于模型训练,但另一方面,它增加了模型的计算量。因此,需要将卷积层和 BN 层的计算进行优化,以减少计算量。卷积层与 BN 层的融合过程[17]如下:

对于卷积层,卷积的输入和输出都由多个二维特征图组成,卷积层的计算输出的公式可以表示为

$$Y_C = wx + b \tag{1}$$

其中,Y_c 表示目标位置卷积后的输出特征图,w 是该位置卷积核的权重,x 为输入特征图的参数,b 为偏执量。

对于 BN 层,BN 层是在每一层的前向传播过程中对输入数据进行规范化,应用两个可学习的转换参数即缩放因子和偏置项,来调整归一化的范围,其公式为

$$Y_{\mathrm{BN}} = \gamma \frac{Y_c - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta \tag{2}$$

其中,Y_{BN} 表示经过 BN 层后的输出特征图,γ 表示比例因子,μ 为输入数据的平均值,σ 表示输入数据的方差,β 表示偏移系数,ε 是一个非常小的固定值,以防止分母为零。

为了将卷积层与 BN 层融合,将卷积计算式(1)引入 BN 层计算式(2)中,并展开得到融合式(3):

$$Y' = \gamma \frac{wx}{\sqrt{\sigma^2 + \varepsilon}} + \gamma \frac{b - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta \tag{3}$$

融合后新的卷积层如式(4)、式(5)所示:

$$w' = \gamma \frac{w}{\sqrt{\sigma^2 + \varepsilon}} \tag{4}$$

$$\beta' = \gamma \frac{b - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta \tag{5}$$

融合后的卷积计算如下:

$$Y' = w'x + \beta' \tag{6}$$

在此过程中,BN 层和卷积层融合后没有精度损失,这种方法减少模型的参数量,提高计算密度。从而可以降低卷积的复杂度,提高模型的计算效率。

2.3　卷积 IP 模块设计

通常,卷积的过程是利用卷积核作为滑动窗口,对相应的输入特征图进行计算。因此,该过程的输出公式为

$$\begin{aligned}\mathrm{out\_fm}[f][r][c] = {} & w_t[f][c_h][k_r][k_c] \times \\ & \mathrm{in\_m}[c_h][S \times r + k_r][S \times c + k_c]\end{aligned} \tag{7}$$

其中,out_fm 表示输出特征图,f 表示对第几个卷积核进行计算,r 和 c 分别表示第几行和第几列,w_t 表示对应位置的权重,c_h 表示通道,k_r 和 k_c 分别表示卷积核内的第几行和第几列,in_m 表示输入特征图,S 表示卷积步长。

设计卷积 IP 模块的关键在于设计一个卷积算子。通常的设计思路是按照不同的特定算子在 PL 端实现固定尺寸的 PE 阵列,所以需要的工作量较大。本文先确定卷积后输出特征图的尺寸大小 OFM,利用卷积尺寸计算公式倒推出对应的输入特征图的尺寸大小和坐标 IFM,最后进行填充,从而简化了算子设计方法,如图 4 所示。

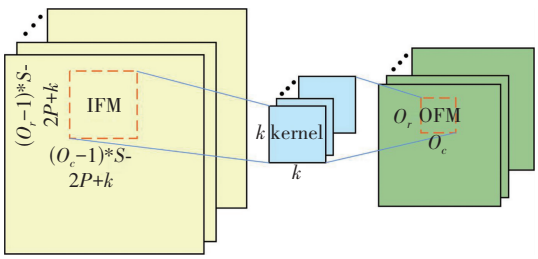

图 4　卷积模块原理图

Fig. 4　Convolutional module schematic diagram

卷积尺寸计算的公式为

$$H'=\frac{H+2\times\text{Padding}[0]-\text{dilation}[0]\times\text{kernel}[0]-1}{\text{Stride}[0]}+1 \tag{8}$$

$$W'=\frac{W+2\times\text{Padding}[1]-\text{dilation}[1]\times\text{kernel}[1]-1}{\text{Stride}[1]}+1 \tag{9}$$

其中,H 表示输入张量的行长,H' 表示输出张量的行长,W 表示输入张量的列长,W' 表示输出张量的列长,dilation 默认值为 1。设 H 为 I_r,H' 为 O_r,Padding 为 P,kernel 为 k,Stride 为 S,则通过公式倒推出输出特征图行的尺寸大小为

$$I_r=O_r\times S-(2\times P+S-K)=(O_r-1)\times S-2P+k \tag{10}$$

同理,输出特征图列的尺寸大小为

$$I_c=O_c\times S-(2\times P+S-K)=(O_r-1)\times S-2P+k \tag{11}$$

其中,O_r、P、k 这 3 个参数都是可以直接配置的,但步长 S 是变化的,所以当 IFM 不能确定时,S 可直接选取最大值 2 计算出尺寸作为 IFM 对应的最大长度。超出图形边界的部分可直接用 0 进行填充,最终的计算结果不会受影响。当得到任意尺寸的 IFM,就可以读取数据计算出 OFM,最后缓存所得结果就实现了卷积计算。

2.4 池化 IP 模块设计

池化计算在卷积神经网络中是很重要的一步,池化层需要对单个输入特征图进行采样,然后进行平均池化和最大池化,其主要功能是降低维数和提取主要特征。与卷积相比,卷积的运算单元是加法器和乘法器,池化运算更多地使用了比较器。与卷积类似,池化计算也是以滑动窗口的形式进行,池化核心是从单个的输入缓存中读取所有数据,并将它们与当前最大值进行比较,在此比较之后,将生成新的最大值写入输出缓存。

在本文中每个时钟周期,池化 IP 模块从输入特征缓存中读取所有数据,并将它们与当前的最大值进行比较,在此比较之后,将生成的最大值写入输出缓存。并且池化 IP 模块采用多个并行比较器,可同时对缓存的多个输入通道中的特征数据进行比较操作。因为池化层不需要权值参数,占整个网络计算量较小,所以池化 IP 模块采用了简单的流水化策略。如图 5 所示,在本文所采用的 YOLOv4-tiny 网络中,池化核的尺寸为 2×2,步长为 2,这可以将输入特征图的尺寸缩小到原始特征图的一半。

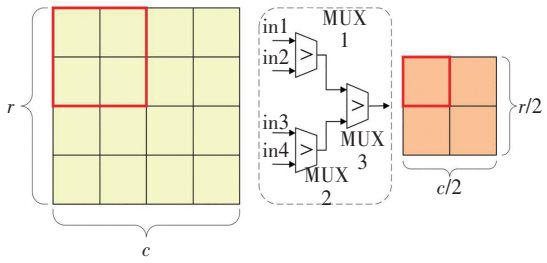

图 5 池化模块原理图

Fig. 5 Pooling module schematic diagram

2.5 量化模型

在相同的硬件资源约束下,利用定点 16 位量化模型,将浮点运算数表示的输入和输出特征图中的权值、参数和偏差转换为整数,实现更简单的整数运算。这可以降低计算复杂度和内存需求,实现高度的计算并行性,也可以减少访问的数据量以及数据带宽,从而提高计算效率和功耗效率。此外,动态定点 16 位量化技术可以缓解由于硬件条件限制而无法使用高精度浮点数的问题,从而促进了卷积神经网络模型在边缘设备上的部署。虽然定点量化在一定程度上降低了模型的检测精度,这是因为量化会导致信息损失和舍入误差。但是,卷积神经网络对输出精度具有较强的鲁棒性,所以对于一些精度要求相对较低的轻量级模型或任务,模型检测精度的下降仍在合理范围内。

将神经网络中 32 位浮点算术数的参数量化为 16 位浮点算术数,可以有效地减少模型所占用的计算和存储空间,提高计算速度,使其更容易在 FPGA 等嵌入式设备上实现。本文将 YOLOv4-tiny 的 32 位输入和输出特征图中的权值、参数和偏差量化为 16 位定点数据。量化公式为

$$Y_{\text{fixed}}=\text{round}(B_{\text{float}}\times 2^i) \tag{12}$$

其中,Y_{fixed} 表示后量化后的定点整数,round 是一个舍入函数,B_{float} 为待量化的浮点数据,i 为量化因子,根据各层数据而变化。

3 实验与结果分析

3.1 实验环境

本实验采用 PYNQ-Z2 平台,PYNQ 是将 ZYNQ 部分功能的 Python 化,可直接调用 Python 库和 FPGA 硬件库进行功能的开发。PYNQ 芯片采用 Xilinx 的 ZYNQ7000 系列芯片,ZYNQ-7020 核心板上的芯片类型为 XC7Z020CLG400-2。PYNQ-Z2 平台的处理器是

双核 Cortex-A9 架构芯片,平台资源有 53 200 个 LUT、17 400 个 LUTRAM、140 个 BRAM、106 400 个 FF 和 220 个 DSP,搭载了 DDR3 芯片可以处理大内存和高带宽的情况,DDR3 内存还可以作为 PS 处理器的运行内存。

数据集为 VOC2007 数据集,首先使用 Vivado HLS 2020.1 工具上设计 YOLOv4-tiny 的卷积 IP 模块和池化 IP 模块,构建出网络结构。然后,在 Vivado2020.1 上设计 Block Design 并配置其参数,进行布局布线和综合。最后得到资源使用报告、时序分析报告和功耗报告。

3.2　结果分析

实验所采用的时钟频率为 100 MHz,资源使用报告如表 2 所示。从表 2 中可知,LUT 和 DSP 的资源利用率最高,分别为 82%和 99%,在 YOLOv4-tiny 网络中主要用于乘法器和加法器的计算资源,在 FPGA 中资源利用率越高,计算的并行性就越好,说明设计的效果也越好。

表 2　FPGA 资源利用率情况

Tabel 2　FPGA resource usage

Item	Utilization	Available	Utilization/%
LUT	43 639	53 200	82.03
LUTRAM	9 004	17 400	51.75
FF	45 857	106 400	43.10
BRAM	74.50	140	53.21
DSP	218	220	99.09

在嵌入式系统中,功耗是保证系统稳定可靠运行的重要指标,根据 Vivado 的功耗报告,片上 SoC 的动态功耗为 2.47 W,总功耗为 2.65 W。从验证平台获得本文处理单张图片加速后的延迟为 450 ms,较大地提高了单张图片的计算效率。从计算性能和效率来看,计算性能达到 15.33 GOPS,计算能源效率达到 5.79 GOPS/W。

CPU 性能与文献[18]相比,本文设计和优化的 YOLOv4-tiny 网络的硬件加速器在性能、功耗和能效比较,如表 3 所示。由于本文使用卷积层与 BN 层融合,总计算负载减小,导致功耗略低。在经过算子和模块进行设计和优化后,与未量化的 32 位 CPU 平台相比,性能提高了 0.26 倍,能效提高了 29.5 倍。与文献[18]中设计的方案相比,在更低的时钟频率下,其整体性能提高了 2.79 倍,系统的能效提高了 3.02 倍。

表 3　性能与能效比较

Table 3　Comparison of performance and energy efficiency

Item	CPU(6-Cores)	ZNYQ-7020[18]	This paper
Power/W	65	2.8	2.65
Run time/Hz	3.6 G	180 M	100 M
Performance/(GOPS)	12.12	4.04	15.33
Efficiency/(GOPS · W^{-1})	0.19	1.44	5.79

4　结　论

本文提出了一种轻量化网络模型 YOLOv4-tiny 的硬件加速方法,通过设计和优化将其应用于低功耗 FPGA 硬件平台 PYNQ-Z2 上。对模型进行了卷积层与 BN 层的融合和 16 位定点量化,降低了计算复杂度和资源消耗率;采用双缓冲策略提高了数据吞吐量,加速了计算速度;对卷积模块和池化模型进行了高效并行的设计,提升了整体系统的计算性能。实验得出数据,该方法加速后处理单张图片的计算延迟为 450 ms,计算性能为 15.33 GOPS,功耗仅为 2.65 W,与 CPU 和同类 FPGA 平台相比,在性能、功耗和资源利用等方面具有更高的综合性能。

本文研究充分地利用了硬件 FPGA 的并行性与逻辑资源和 PS 易搭建操作系统,使整个系统架构计算性能高、系统功耗低、资源利用率高,对卷积神经网络在边缘 FPGA 平台进行加速的研究具有一定的价值,但目前由于硬件条件限制,检测速率不够快。后续研究中,可以考虑对神经网络设计更加合理的并行算子和优化整体的系统架构来进一步提高计算性能和检测速率。

参考文献(References):

[1]　梁振明, 黄影平, 宋卓恒, 等. 自动驾驶中基于深度学习的 3D 目标检测方法综述[J]. 上海理工大学学报, 2024, 46(2): 103-119.
LIANG Zhen-ming, HUANG Ying-ping, SONG Zhuo-heng, et al. Survey on deep learning-based 3D object detection methods in autonomous driving[J]. Journal of University of Shanghai for Science and Technology, 2024, 46(2): 103-119.

[2]　杨晓艳, 邓淼磊, 张德贤, 等. 基于判别模型的年龄不变人脸识别方法综述[J]. 计算机工程与应用, 2023, 59(24): 16-25.

YANG Xiao-yan, DENG Miao-lei, ZHANG De-xian, et al. Review of age-invariant face recognition methods based on discriminant models[J]. Computer Engineering and Applications, 2023, 59(24): 16-25.

[3] LIN Q, WANG R, WANG Y, et al. Target detection algorithm incorporating visual expansion mechanism and path syndication[J]. IEEE Access, 2023(11): 56973-56982.

[4] WANG B, ZHANG F, ZHAO Y. LCH: fast RGB-D salient object detection on CPU via lightweight convolutional network with hybrid knowledge distillation[J]. The Visual Computer, 2024, 40(3): 1997-2014.

[5] ARTAMONOV N S, YAKIMOV P Y. Towards real-time traffic sign recognition via YOLO on a mobile GPU[J]. Journal of Physics: Conference Series, 2018, 1096: 012086.

[6] LI R, LIU K, ZHAO M, et al. Maximizing CNN throughput on FPGA clusters[C]//Proceedings ofthe 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. New York: ACM, 2020: 319-330.

[7] LIU X, CHEN Y, HAO C, et al. WinoCNN: Kernel sharing winograd systolic array for efficient convolutional neural network acceleration on FPGAs[C]//Proceedings of the IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors. Piscataway: IEEE Press, 2021: 258-265.

[8] 徐睿, 马胜, 郭阳, 等. 基于Winograd稀疏算法的卷积神经网络加速器设计与研究[J]. 计算机工程与科学, 2019, 41(9): 1557-1566.
XU Rui, MA Sheng, GUO Yang, et al. A convolutional neural network accelerator based on Winograd-Sparse algorithm[J]. Computer Engineering & Science, 2019, 41(9): 1557-1566.

[9] LU L, LIANG Y, XIAO Q, et al. Evaluating fast algorithms for convolutional neural networks on FPGAs[C]//Proceedings of the IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines. Piscataway: IEEE Press, 2017: 101-108.

[10] 施巍松, 张星洲, 王一帆, 等. 边缘计算: 现状与展望[J]. 计算机研究与发展, 2019, 56(1): 69-89.
SHI Wei-song, ZHANG Xing-zhou, WANG Yi-fan, et al. Edge computing: state-of-the-art and future directions[J]. Journal of Computer Research and Development, 2019, 56(1): 69-89.

[11] JIANG H, WANG Z, ZHOU Z, et al. A testing program and pragma combination selection based framework for high-level synthesis tool pragma-related bug detection[J]. IEEE Transactions on Software Engineering, 2024, 50(4): 937-955.

[12] WU D, ZHANG Y, JIA X, et al. A high-performance CNN processor based on FPGA for MobileNets[C]//Proceedings of the 29th International Conference on Field Programmable Logic and Applications. Piscataway: IEEE Press, 2019: 136-143.

[13] 李大琳. 基于FPGA的高性能算法实现的设计模式及其应用研究[D]. 长春: 吉林大学, 2020: 11-17.
LI Da-lin. Research on Design Pattern and Application of High Performance Algorithm Based on FPGA[D]. Changchun: Jilin University, 2020: 11-17.

[14] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real-time object detection[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE Press, 2016: 779-788.

[15] JIN M, LI H, XIA Z. Hybrid attention network and center-guided non-maximum suppression for occluded face detection[J]. Multimedia Tools and Applications, 2023, 82(10): 15143-15170.

[16] 陈辰, 柴志雷, 夏珺. 基于Zynq7000 FPGA异构平台的YOLOv2加速器设计与实现[J]. 计算机科学与探索, 2019, 13(10): 1677-1693.
CHEN Chen, CHAI Zhi-lei, XIA Jun. Design and implementation of YOLOv2 accelerator based on Zynq7000 FPGA heterogeneous platform[J]. Journal of Frontiers of Computer Science and Technology, 2019, 13(10): 1677-1693.

[17] TIAN Y, MAO W, YUAN S, et al. A decision support system for power components based on improved YOLOv4-tiny[J]. Scientific Programming, 2021, 20(21): 4447271.

[18] 曹远杰, 高瑜翔, 杜鑫昌, 等. 基于改进YOLOv4-Tiny的FPGA加速方法[J]. 无线电工程, 2022, 52(4): 604-611.
CAO Yuan-jie, GAO Yu-xiang, DU Xin-chang, et al. FPGA acceleration method based on improved YOLOv4-tiny[J]. Radio Engineering, 2022, 52(4): 604-611.

责任编辑:代小红