

DOI: 10.11830/ISSN.1000-5013.202409017



# 软硬件协同设计的异构 CNN 加速器

谢志豪<sup>1,2</sup>, 李国刚<sup>1,2</sup>

(1. 华侨大学 信息科学与工程学院, 福建 厦门 361021;

2. 华侨大学 厦门市专用集成电路与功率半导体系统重点实验室, 福建 厦门 361021)

**摘要:** 为解决卷积神经网络(CNN)高效部署的挑战,提出一种基于软硬件协同设计的异构 CNN 加速器,并在 YOLOv4-tiny 模型上进行验证。搭建基于高级精简指令集机器(ARM)处理器与现场可编程门阵列(FPGA)的异构系统。通过高层次综合(HLS)将可并行执行的计算单元映射为 FPGA 端寄存器传输级(RTL)知识产权(IP);ARM 处理器控制系统的协同工作与 IP 核的调度,最终实现前向推理加速。结果表明:该异构 CNN 加速器的工作频率为 130 MHz,功耗为 2.809 W,推理速度达到 511 ms,吞吐率为 13.40 GOPS;相较于桌面端图形处理单元(GPU)、中央处理单元(CPU)及主流嵌入式 AI 加速平台,该设计在推理速度与功耗之间取得了良好平衡,同时关键性能指标均有显著提升;所设计异构 CNN 加速器在边缘计算场景中表现出优异性能,能够满足实际部署需求。

**关键词:** 现场可编程门阵列(FPGA); 硬件加速; 软硬件协同设计; 高层次综合

**中图分类号:** TP 391.4

**文献标志码:** A

**文章编号:** 1000-5013(2025)02-0209-08

## Heterogeneous CNN Accelerator Based on Hardware-Software Co-Design

XIE Zhihao<sup>1,2</sup>, LI Guogang<sup>1,2</sup>

(1. School of Information and Engineering, Huaqiao University, Xiamen 361021, China;

2. Xiamen Key Laboratory of ASIC and Power Semiconductor System, Huaqiao University, Xiamen 361021, China)

**Abstract:** To address the challenges associated with the efficient deployment of convolutional neural network (CNN), a heterogeneous CNN accelerator based on a hardware-software co-design is proposed and validated on the YOLOv4-tiny model. The heterogeneous system is built with an advanced reduced instruction set machine (ARM) processors and a field programmable gate array (FPGA). Through high-level synthesis (HLS), the computational units that can be executed in parallel are mapped to a register transfer level (RTL) intellectual property (IP) on FPGA. The ARM processors manage the collaborative operations of the system and the scheduling of the IP core, ultimately achieving acceleration of forward inference. The results show that the heterogeneous CNN accelerator operates at a frequency of 130 MHz, with a power consumption of 2.809 W and an inference speed of ms, achieving a throughput of 13.40 GOPS. Compared to desktop graphics processing unit (GPU), central processing unit (CPU) and mainstream embedded AI acceleration platforms, the proposed design achieves a favorable balance between inference speed and power consumption, while significantly improving key performance indicators. The designed heterogeneous CNN accelerator demonstrates excellent performance in edge computing scenarios and meets the requirements for practical deployment.

**收稿日期:** 2024-09-25

**通信作者:** 李国刚(1973—),男,副教授,博士,主要从事电路系统和神经网络的研究。E-mail:lgg@hqu.edu.cn。

**基金项目:** 国家自然科学基金资助项目(61370007);福建省高校产学研合作项目(2023H6013)

**Keywords:** field programmable gate array (FPGA); hardware acceleration; hardware-software co-design; high-level synthesis

卷积神经网络(CNN)在特征提取、特征整合等方面的优异性能为目标检测等计算机视觉任务注入了强大动能。近年来,基于 CNN 的目标检测算法已在工业检测、人脸识别、自动驾驶等边缘端场景表现出广泛的应用潜力。随着算法应用场景的拓宽和硬件技术的发展,在高性能、低功耗、灵活及通用的边缘端平台部署 CNN 目标检测算法已成为新的研究热点<sup>[1]</sup>。然而,在实际部署过程中仍然存在瓶颈。

一方面,CNN 模型的计算复杂度和参数量不断膨胀,对硬件平台提出了更高的要求。主流的 CNN 加速平台包括图形处理单元(GPU)和中央处理单元(CPU)为代表的通用性芯片、以专用集成电路(ASIC)为代表的全定制化芯片和以现场可编程门阵列(FPGA)为代表的半定制化芯片等。

GPU 具备高强度的并行计算性能,尤其适合处理 CNN 中的计算密集型任务。然而,巨大的功耗开销限制了 GPU 在边缘端的应用<sup>[2]</sup>。虽然 CPU 在计算效率等方面不如 GPU,但其擅长处理复杂的逻辑控制等控制密集型任务<sup>[3]</sup>。而高级精简指令集机器(ARM)处理器作为嵌入式 CPU 平台,表现出功耗低、面积小、便携性强及控制流指令精简等优点<sup>[4]</sup>,在边缘端的应用中具有优势。ASIC 是针对特定应用定制化的集成电路,拥有卓越的计算效能。然而,ASIC 一旦制造完成,内部的电路结构将无法进行更改<sup>[5]</sup>,导致设计缺乏灵活性。FPGA 利用可配置的逻辑资源实现半定制化的功能<sup>[6]</sup>,具有可重构和低功耗等特点。虽然 FPGA 加速算法的整体效果不及 ASIC 芯片,但在一些对于开发周期有较高要求的应用场景中,研发人员往往倾向于采用 FPGA 对算法进行迭代设计和验证。此外,FPGA 片上的数字信号处理器(DSP)能够以高效的流水线(Pipeline)方式执行 CNN 中的乘法和加法等运算操作。综上所述,ARM 和 FPGA 在功耗、并行性、可重构性及通用性等方面具有显著优势,吸引了国内外学者的广泛关注。高树静等<sup>[7]</sup>将优化后的 Adaboost 算法部署到硬件平台,利用 ARM 处理器实现人脸检测任务。嵇达龙等<sup>[8]</sup>采用 ZYNQ 平台搭建行人检测系统,FPGA 端的检测速度是 ARM 端的 45 倍。Lu 等<sup>[9]</sup>在 FPGA 上实现了稀疏 CNN 加速器,采用数据复用策略降低片内和片外的访存压力。Bai 等<sup>[10]</sup>对深度可分离卷积进行改进,使其在不同规模的 FPGA 上具有更好的伸缩性。Ma 等<sup>[5]</sup>采用循环展开和循环分块策略设计了基于单指令多数据(SIMD)架构的 CNN 加速器。然而,上述研究单独采用 ARM 处理器或 FPGA 实现 CNN 加速器,并未结合二者处理 CNN 中控制密集型任务和计算密集型任务的独特优势。

另一方面,开发工具的不完善也在一定程度上导致了算法迭代与硬件实现之间的不平衡。尽管基于寄存器传输级(RTL)的 FPGA 建模在细粒度上的把控更加到位,但存在设计复杂、效率低等缺点,因而不利于硬件加速器的快速实现。在这种背景之下,高层次综合(HLS)应运而生。HLS 是一种将高级语言综合成硬件描述语言(HDL),进而生成 RTL 级网表的技术栈。Millón 等<sup>[11]</sup>分别采用 HDL 和 HLS 设计了 Sobel 滤波器,以评估二者在图像处理算法方面的性能差异,结果表明,HDL 设计在资源占用和推理延时方面具有一定的优势,然而其开发周期约为 HLS 设计的 3.17 倍。由此可见,HLS 能够实现 CNN 算法的高级快速原型设计。

针对 CNN 目标检测算法在边缘端场景的应用需求,本文采用软硬件协同设计的方法构建基于 ARM+FPGA 的异构 CNN 加速器。

# 1 相关理论

## 1.1 YOLOv4-tiny 模型

当前 CNN 目标检测算法主要分为两类:一类是以 R-CNN<sup>[12]</sup>、Fast R-CNN<sup>[13]</sup>和 Faster R-CNN<sup>[14]</sup>为代表的两阶段目标检测算法,这类算法计算流程复杂,但检测精度高;另一类是以 YOLO<sup>[15]</sup>系列为代表的一阶段目标检测算法,此类算法在精确度方面做出了一定的妥协,以此换取更精简的工作流程和更快的检测速度。综合考虑模型自身的性能与网络复杂度及硬件平台的算力和存储资源,选择采用 YOLOv4-tiny 算法验证所设计的异构 CNN 加速器。

YOLOv4-tiny 模型结构,如图 1 所示。骨干网络 CSPDark53-tiny 由 3 个 Conv\_BN\_LeakyRelu 模

块和 3 个 Resblock body 模块堆叠而成。其中, Conv\_BN\_LeakyRelu 模块包含二维卷积(Conv2d)、批归一化层(BN)和 LeakyRelu 激活函数; Resblock body 模块则由 4 个 Conv\_BN\_LeakyRelu 模块、1 个 Maxpool 模块和 2 个 Concat 模块构成。在 Resblock body 模块中, 输入  $C_{in}$  经过 Conv\_BN\_LeakyRelu1 后分成两支, 一支为 route1, 另一支经过 Conv\_BN\_LeakyRelu2 得到 route2; route2 与其经过 Conv\_BN\_LeakyRelu3 的输出进行通道拼接(Concat1)后作为 Conv\_BN\_LeakyRelu4 的输入, 并得到 feat; feat 与 route1 经过通道拼接(Concat2)后传入 Maxpool, 得到最终输出。

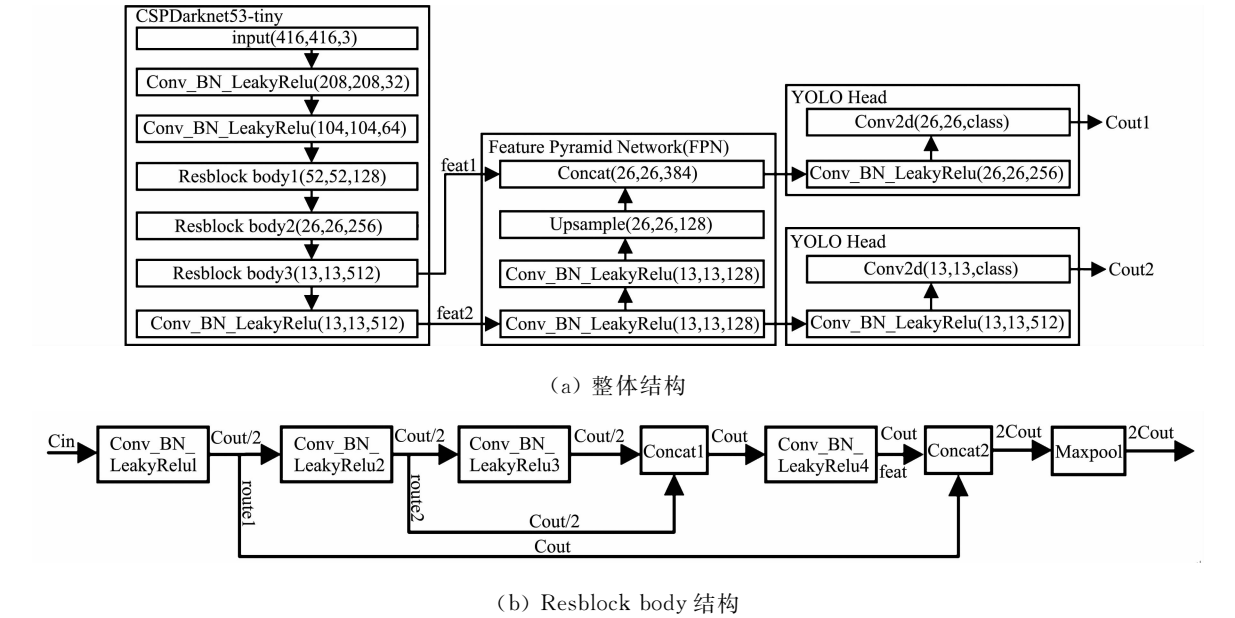


图 1 YOLOv4-tiny 模型结构

Fig. 1 YOLOv4-tiny model structure

特征金字塔网络(FPN)由 2 个 Conv\_BN\_LeakyRelu 模块、1 个 Upsample 模块和 1 个 Concat 模块构成。CSPDarknet53-tiny 的最后一层有效特征  $feat_2$  经过 2 个 Conv\_BN\_LeakyRelu 模块后输入 Upsample 进行上采样, 输出结果再与前一层有效特征即 Resblock body3 的池化前特征  $feat_1$  进行 Concat 操作。检测头(YOLO Head)由 Conv\_BN\_LeakyRelu 模块和 Conv2d 构成。YOLOv4-tiny 模型含有 2 个 YOLO Head, 分别输出  $C_{out1}$  和  $C_{out2}$ , 其维度分别为  $(13, 13, class)$  和  $(26, 26, class)$ , 其中,  $class$  表示数据集的类别数。

### 1.2 高层次综合开发

HLS 将高层次算法映射到 RTL 级网表的过程包括调度阶段、初始绑定阶段和目标绑定阶段, 如图 2 所示。调度阶段将乘法运算和加法运算安排在 2 个时钟周期内完成, 初始绑定阶段则将这些运算绑定到乘法器和加法器。FPGA 片上的 DSP48E1 单元本质上是具有专用硬件乘法器的 CPU, 能够在 1 个时钟周期内完成乘法和加法运算。因此, 在目标绑定阶段由 DSP48E1 单元执行相应的运算操作。

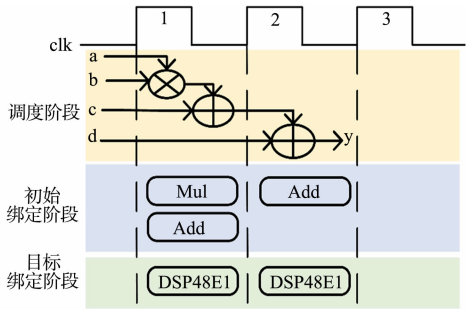


图 2 HLS 调度和绑定

Fig. 2 Scheduling and binding of HLS

## 2 异构 CNN 加速器软硬件协同设计

### 2.1 顶层设计

异构系统的顶层设计架构, 如图 3 所示。ARM 处理器是异构系统的中央控制单元, 通过协调和控制接口映射、数据交互、内存管理及任务调度等核心功能, 确保系统各部分的协同运行。FPGA 端加速 IP 核集成了标准卷积、逐点卷积、最大池化及上采样等通用计算单元。加速 IP 作为协处理器, 由 ARM 处理器挂载与驱动。

1) 接口映射。ARM 处理器和 FPGA 之间通过高级可扩展接口 (AXI) 进行高效通信。AXI 可分

为高性能的 AXI HP 及通用的 AXI GP。其中,AXI HP 主要用于传输高速数据流,而 AXI GP 主要负责传递控制流指令。

2) 数据交互。输入特征图、权重、偏置和输出特征图对应的数据流传输端口分别绑定到 AXI\_IFM、AXI\_Wt、AXI\_Bias 及 AXI\_OFM。而前向推理阶段所需的算子超参数(计算核参数  $k$  等)及加速 IP 核的配置状态等,则通过 AXI GP 传输。

3) 内存管理。异构系统采用“加载-计算-写回”的三级计算通信机制,从而优化数据传输路径,并补偿传输延时。ARM 处理器将外部存储(SD 卡)存放的数据流加载到主存(DDR)中,并通过数据缓存(Dcache)存储。这些数据流通过 AXI HP 到达 FPGA 片上缓存区( $*\_buff$ ),进而以多维数组的形式参与计算任务。输出特征图同样经过 Dcache-DDR 环路写回 SD 卡。

4) 任务调度。YOLOv4-tiny 模型包含不同类型的计算单元。因而,ARM 处理器通过传送计算核参数  $k$ ,驱动标准卷积( $k=3$ )、逐点卷积( $k=1$ )、最大池化( $k=2$ )或上采样单元( $k=0$ ),并执行加速计算。最终,通过多次调用加速 IP 核实现前向推理加速。

2.2 FPGA 端卷积单元设计

2.2.1 卷积并行计算 卷积单元的乘累加计算占据了 CNN 前向推理过程中 90% 以上的计算量<sup>[16]</sup>。因此,重点讨论卷积单元的设计方案。假设输入特征图的维度为  $N_{if} \times N_{ix} \times N_{iy}$ ,卷积核的维度为  $N_{of} \times N_{if} \times N_{kx} \times N_{ky}$ ,卷积步长为  $s_x \times s_y$ ,则输出特征图的维度为  $N_{of} \times N_{ax} \times N_{oy}$ ,且存在如下关系:

$$N_{ix} = (N_{ax} - 1) \times s_x + N_{kx},$$

(1)

$$N_{iy} = (N_{oy} - 1) \times s_y + N_{ky}。$$

(2)

HLS 设计采用多重 for 循环表示卷积计算,如图 4 所示。CNN 中存在 4 种并行计算架构,即卷积核内并行、输入通道级并行、输出通道级并行及卷积窗口并行。图 4 中:第 5~6 行、第 4 行及第 1 行代码分别对应前 3 种并行计算架构。第 1、4、5、6 行代码分别遍历卷积核的输出通道、输入通道、宽度和高度,数据块尺度为  $n_o \times n_i \times k_x \times k_y$ ;而第 1~3 行代码则遍历输出特征图的各维度,数据块尺度为  $n_o \times x \times y$ 。因此,该伪代码的每次循环迭代读取输入特征图上  $n_i \times [(x-1) \times s_x + k_x] \times [(y-1) \times s_y + k_y]$  个像素,以及卷积核中  $n_o \times n_i \times k_x \times k_y$  个权重值,并执行乘累加运算。

多维数组通常被映射为 BRAM 结构,进而构成 FPGA 片上的缓存区。为适应 BRAM 的容量限制,对数据进行分块处理(输入通道分块为  $T_{if}$ 、输出通道分块为  $T_{of}$ 、输出特征图数据块的高度为  $T_{oy}$ 、宽度为  $T_{ax}$ ,分别对应  $n_i$ 、 $n_o$ 、 $y$  及  $x$  的批处理强度)。同时,通过高层次指令 HLS Array Partition 对多维数组的指定维度进行切分,从而数据块的局部并行处理。

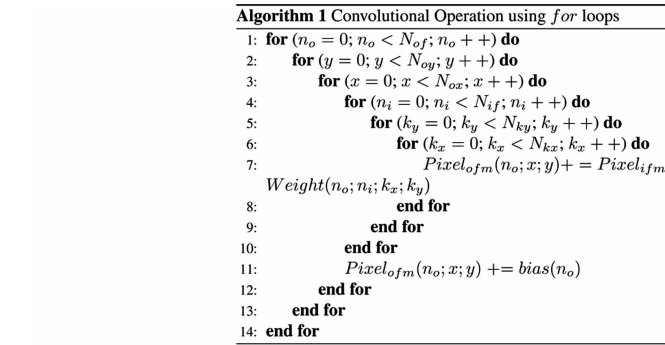


图 4 多重 for 循环表示卷积计算

Fig. 4 Representation of convolutional computation using multiple for-loops

2.2.2 卷积单元硬件结构 卷积单元计算架构,如图 5 所示。采用循环分块策略分批次加载尺度为

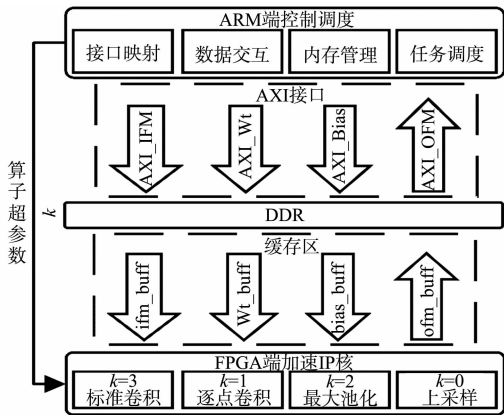


图 3 顶层设计架构

Fig. 3 Top-level design architecture

$T_{if} \times T_{ix} \times T_{iy}$  的输入特征图分块及尺度为  $T_{of} \times T_{i,f} \times N_{kx} \times N_{ky}$  的卷积核分块。输入特征图在内存中存储的基地址(baseaddr)为  $c \times N_{ix} \times N_{iy} + \text{row} \times N_{iy} + \text{col}$ , 其中,  $c$ , row 和 col 分别表示输入特征图的通道索引, 像素的行索引与列索引。通过设定行偏移量(rowoffset)为  $T_{iy} \times N_{iy}$ , 得到行缓存区的偏移地址为  $\text{in} + \text{baseaddr} + \text{rowoffset} + T_{ix}$ , 其中, in 为指向输入缓存区(ifm\_buff)的指针。在电路设计中, 使用先入先出(FIFO)及寄存器(Regs)将来自 ARM 端的数据流推入 FPGA 片上缓存区。随后, 依次对输入缓存区和权重缓存区(wt\_buff)中的数据进行乘累加运算, 计算结果进一步送入 LeakyRelu 激活函数单元, 从而完成非线性映射。然后, 对该部分的输出叠加偏置(bias)。最终生成的输出特征图被写回到输出缓存区(ofm\_buff)。

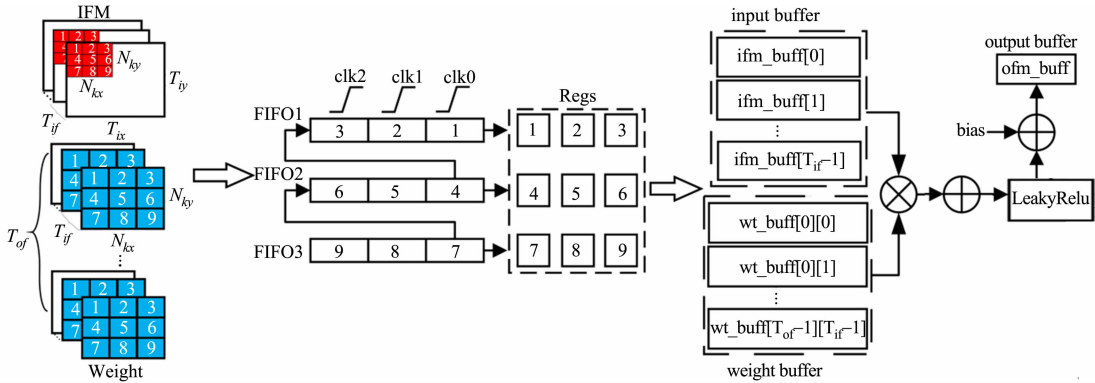


图 5 卷积单元计算架构

Fig. 5 Computing architecture of convolutional units

### 2.3 ARM 端控制调度设计

ARM 端控制调度程序流程图, 如图 6 所示。

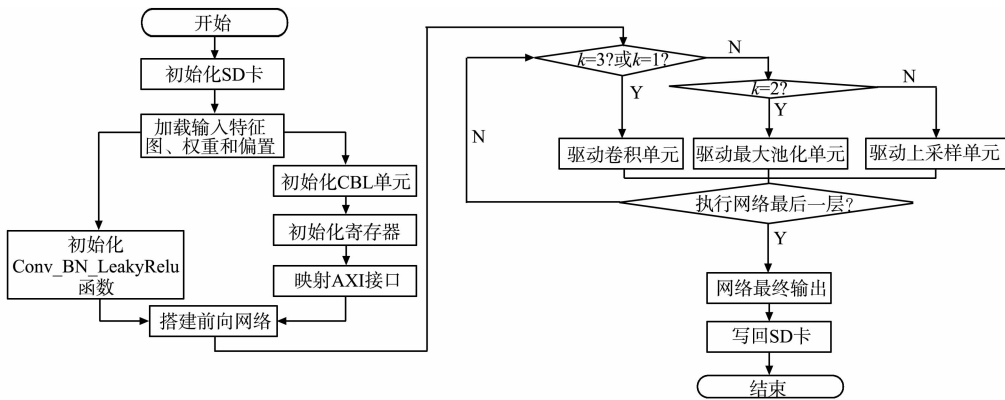


图 6 ARM 端控制调度程序流程图

Fig. 6 Program flow chart of controlling and scheduling on ARM

1) 初始化 SD 卡读写函数。首先, 读取浮点数格式的输入数据。接着, 对输入数据进行缩放处理, 并将其存放到目标数组中。在此过程中, 将浮点数转换为定点数, 从而进一步降低 FPGA 的计算负载。

2) 搭建前向传播网络。Conv\_BN\_LeakyRelu 模块是 YOLOv4-tiny 模型的基础组成单元。因此, 定义 Conv\_BN\_LeakyRelu 函数, 并初始化前向推理子函数及加载参数子函数。通过堆叠 Conv\_BN\_LeakyRelu 模块, 进而构建前向传播网络。此外, CBL 模块的作用体现在两方面: 一方面, 初始化数据寄存器, 从而获取内存数据的基地址、偏移地址及加速 IP 核的配置状态等; 另一方面, 映射 AXI, 以实现 ARM 处理器与 FPGA 双端的计算通信。通过在前向推理子函数中多次调用 CBL 模块, 生成中间计算结果, 从而驱动上层计算任务。

3) 执行前向推理。顺序执行前向推理网络, 根据  $k$  值挂载对应的计算单元, 从而实现前向推理加速。在写回阶段, 首先, 将输出特征图反量化为浮点数格式, 以便后续进行解码工作。接着, 通过输出特征图与 16 进制数 0xfffffe0 的按位与操作, 确保数据对齐。最后, 通过 Xil\_DcacheFlushRange 函数和

Xil\_DcacheInvalidateRange 函数将数据从 Dcache 刷新到 DDR 中,并流回 SD 卡。此过程实现了 ARM 处理器与主存之间数据传输的一致性。

### 3 实验结果与分析

#### 3.1 实验环境

使用 ZYNQ 7020 异构计算平台,主控芯片为 Xilinx XC7Z020clg400-2。采用 6 核心 12 线程 Intel i5-12490F CPU 和 Nvidia RTX 3060(12 GB)GPU。使用 Dashcam<sup>[17]</sup>数据集,其中包括训练集 16 200 张、验证集 1 800 张、测试集 2 000 张。

#### 3.2 资源评估

异构加速器的资源消耗情况,如表 1 所示。表 1 中:DSP 主要用于构建卷积单元中的乘法器和加法器,使用率为 96.82%;BRAM 主要用于构建缓存区,使用率为 70.36%;FIFO 主要用于缓存数据,使用率为 46.53%。

表 1 异构加速器的资源消耗情况  
Tab.1 Resource consumption of heterogeneous accelerator

| 逻辑资源 | 资源总量    | 资源使用量     | 资源使用率/% | 逻辑资源   | 资源总量   | 资源使用量    | 资源使用率/% |
|------|---------|-----------|---------|--------|--------|----------|---------|
| LUT  | 53 200  | 37 815.00 | 71.08   | LUTRAM | 17 400 | 6 984.00 | 40.14   |
| FIFO | 106 400 | 49 511.00 | 46.53   | BRAM   | 140    | 98.50    | 70.36   |
| DSP  | 220     | 213.00    | 96.82   | BUFG   | 32     | 1.00     | 3.13    |

#### 3.3 对比实验

不同硬件平台上 CNN 加速器的性能对比,如表 2 所示。首先,通过与桌面端 GPU(RTX 3060)和 CPU(Intel I5-12490F)的对比可以看出,异构 CNN 加速器的推理速度约是 Intel I5-12490F CPU 的 28 倍。尽管异构 CNN 加速器在推理速度方面不及 RTX 3060,但功耗仅为 7.2%。另外,桌面端平台在体积及便携性等方面均难以满足边缘端的应用需求。其次,将异构 CNN 加速器与同样基于 ARM 处理器的嵌入式 AI 加速平台进行对比。文献[18]分析了在 ASUS Tinker Edge R 与 Google Coral 上部署超轻量级 MobileNet-v2 的性能指标。尽管 ASUS Tinker Edge R 与 Google Coral 在推理速度方面略优于异构 CNN 加速器,但这种优势源于其定制化的 AI 芯片(NPU/TPU)。此外,ASUS Tinker Edge R 与 Google Coral 配备了主频更高的 ARM 处理器及更高端的主存。类似地,文献[19]在嵌入式端 GPU(Nvidia Jetson nano)上部署 YOLOv3-tiny 模型,其推理速度的相对优势同样源自高度优化的多核 GPU。由此可见,上述 AI 加速平台执行 CNN 算法时所取得的性能表现依赖于内部高端硬件的支持。然而,这不可避免地带来了更高的功耗开销。

表 2 不同硬件平台 CNN 加速器性能对比  
Tab.2 Performance comparison of CNN accelerators on different hardware platforms

| 设计来源   | 硬件平台               | 加速引擎               | 主频/GHz | DDR 配置                      | 模型           | 推理速度/ms   | 功耗/W        |
|--------|--------------------|--------------------|--------|-----------------------------|--------------|-----------|-------------|
| 文献[18] | ASUS Tinker Edge R | ARM Cortex A72+NPU | 1.400  | 4 GB LPDDR4/<br>2 GB LPDDR3 | MobileNet-v2 | 153.00    | 7.500~7.800 |
| 文献[18] | Google Coral       | ARM Cortex A53+TPU | 1.500  | 1 GB LPDDR4<br>@1 600 MHz   | MobileNet-v2 | 277.00    | —           |
| 文献[19] | Nvidia Jetson nano | ARM Cortex A57+GPU | 1.430  | 4 GB LPDDR4<br>@1 600 MHz   | YOLOv3-tiny  | 125.00    | 10.200      |
| 文中     | RTX 3060           | GPU                | 1.780  | 12 GB GDDR6<br>@12 000 MHz  | YOLOv4-tiny  | 3.06      | 39.000      |
| 文中     | Intel I5-12490F    | CPU                | 3.000  | 32 GB DDR4<br>@4 800 MHz    | YOLOv4-tiny  | 14 255.70 | —           |
| 文中     | ZYNQ 7020          | ARM Cortex-A9+FPGA | 0.766  | 1 GB DDR3<br>@533 MHz       | YOLOv4-tiny  | 511.00    | 2.809       |

综上所述,异构 CNN 加速器并不依赖于高规格的硬件引擎,而是通过 HLS 将算法映射、绑定到 FPGA 端有限且通用的 DSP 单元上,从而加速并行计算。此外,异构 CNN 加速器利用 ARM 处理器出色的控制调度性能构建了功能完善、协同工作的异构计算系统,从而在固有硬件配置存在局限性的条件

下实现了功耗与性能的良好权衡。

异构 CNN 加速器与其他文献的对比,如表 3 所示。表 3 中:所列工作均基于 ARM+FPGA 异构架构;吞吐率为模型的计算量与在硬件平台上推理速度的比值,反映了硬件加速器的算力;能效比为吞吐率与功耗的比值,反映了硬件平台的计算效率;DSP 效率为吞吐率与消耗 DSP 数量的比值,反映了加速器的优化程度。由表 3 可知:相较于文献[20]~[24],异构 CNN 加速器在推理速度、能效比、吞吐率及 DSP 效率等方面均取得了显著的提升;与文献[23]相比,异构 CNN 加速器的推理速度提高约 3427%;与文献[21]、[24]相比,异构 CNN 加速器的能效比分别提高了约 440%、48%;与文献[20]、[21]、[22]相比,异构 CNN 加速器的吞吐率分别提高约 47%、483%、130%;与文献[22]相比,异构 CNN 加速器的 DSP 效率提升了约 113%。

表 3 异构 CNN 加速器性能对比

Tab. 3 Performance comparison of heterogeneous CNN accelerators

| 设计来源   | 硬件平台      | 模型          | 工作频率/<br>MHz | 量化方式          | 功耗/W  | 推理速<br>度/ms | 吞吐率/<br>GOPS | 能效比/<br>GOPS·<br>W <sup>-1</sup> | DSP 效率/<br>GOPS·<br>DSP <sup>-1</sup> |
|--------|-----------|-------------|--------------|---------------|-------|-------------|--------------|----------------------------------|---------------------------------------|
| 文献[20] | ZYNQ 7020 | YOLOv3-tiny | 100          | fixed-16      | 2.141 | 613         | 9.12         | 4.260                            | —                                     |
| 文献[21] | ZYNQ 7020 | YOLOv5s     | 150          | dynamic fixed | 2.600 | 1 300       | 2.30         | 0.884                            | —                                     |
| 文献[22] | ZC702     | YOLOv5s     | 100          | fp32          | 2.508 | —           | 5.83         | 2.320                            | 0.029 6                               |
| 文献[23] | Zedboard  | YOLOv4-tiny | 100          | dynamic-16    | 2.384 | 18 025      | —            | —                                | —                                     |
| 文献[24] | ZYNQ 7020 | YOLOv4-tiny | —            | fixed-16      | 2.860 | 376         | 9.24         | 3.230                            | —                                     |
| 文中     | ZYNQ 7020 | YOLOv4-tiny | 130          | fixed-16      | 2.809 | 511         | 13.40        | 4.770                            | 0.063 0                               |

## 4 结束语

基于软硬件协同设计的方法,实现了异构 CNN 加速器,并在轻量级模型 YOLOv4-tiny 上进行验证。实验结果表明,所设计异构 CNN 加速器在功耗与性能方面取得了良好的平衡,在边缘计算场景中具有重要的研究意义与应用价值。

## 参考文献:

[1] 李全. 面向 ARM 嵌入式平台的卷积神经网络前向加速研究[D]. 武汉: 华中科技大学, 2019.

[2] 陈朋, 陈庆清, 王海霞, 等. 基于改进动态配置的 FPGA 卷积神经网络加速器的优化方法[J]. 高技术通讯, 2020, 30(3): 240-247. DOI:10.3772/j.issn.1002-0470.2020.03.004.

[3] NIKOLIC G S, DIMITRIJEVIC B R, NIKOLIC T R, *et al.* A survey of three types of processing units: CPU, GPU and TPU[C]//57th International Scientific Conference on Information, Communication and Energy Systems and Technologies. Ohrid: IEEE Press, 2022: 1-6. DOI:10.1109/ICEST55168.2022.9828625.

[4] 王江波. 基于 ZYNQ 嵌入式平台的 CNN 图像识别加速器研究与实现[D]. 沈阳: 中国科学院大学(中国科学院沈阳计算技术研究所), 2022.

[5] MA Yufei, CAO Yu, VRUDHULA S, *et al.* Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks[C]//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. New York: Association for Computing Machinery, 2017: 45-54. DOI:10.1145/3020078.3021736.

[6] 帅禄玮, 张柳欣, 叶蕾, 等. 基于低误差并行计算加速的 OFDR 实时处理技术[J]. 中国激光, 2024, 51(14): 233-242. DOI:10.3788/CJL231526.

[7] 高树静, 王程龙, 董廷坤. 基于 ZYNQ 的优化 Adaboost 人脸检测[J]. 计算机工程与应用, 2020, 56(6): 201-206. DOI:10.3778/j.issn.1002-8331.1812-0228.

[8] 嵇达龙, 张尤赛, 王亚军. 基于 ZYNQ 的行人检测系统的设计与实现[J]. 计算机工程与设计, 2020, 41(1): 238-245. DOI:10.16208/j.issn1000-7024.2020.01.039.

[9] LU Liqiang, XIE Jiaming, HUANG Ruirui, *et al.* An efficient hardware accelerator for sparse convolutional neural networks on FPGAs[C]//27th Annual International Symposium on Field-Programmable Custom Computing Machines. San Diego: IEEE Press, 2019: 17-25. DOI:10.1109/FCCM.2019.00013.



[10] BAI Lin,ZHAO Yiming,HUANG Xinming. A CNN accelerator on FPGA using depthwise separable convolution [J]. IEEE Transactions on Circuits and Systems II : Express Briefs,2018,65(10):1415-1419. DOI:10. 1109/TC-SII. 2018. 2865896.

[11] MILLÓN R,FRATI E,RUCCI E. A comparative study between HLS and HDL on SoC for image processing applications[EB/OL]. (2020-12-15)[2024-08-20]. <https://doi.org/10.48550/arxiv.2012.08320>.

[12] GIRSHICK R,DONAHUE J,DARRELL T,*et al.* Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Columbus; IEEE Press,2014:580-587. DOI:10. 1109/CVPR. 2014. 81.

[13] GIRSHICK R. Fast R-CNN[C]//Proceedings of the IEEE International Conference on Computer Vision. Santiago; IEEE Press,2015:1440-1448. DOI:10. 1109/ICCV. 2015. 169.

[14] REN Shaoqing,HE Kaiming,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. DOI:10. 1109/TPAMI. 2016. 2577031.

[15] 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. Las Vegas; IEEE Press,2016:779-788. DOI: 10. 1109/CVPR. 2016. 91.

[16] CONG J,XIAO Bingjun. Minimizing computation in convolutional neural networks[C]//International Conference on Artificial Neural Networks. Cham; Springer International Publishing,2014:281-290. DOI:10. 1007/978-3-319-11179-7\_36.

[17] CIJOV A. Self-driving cars[EB/OL]. (2021-12-08)[2024-08-20]. <https://www.kaggle.com/datasets/alincijov/self-driving-cars>.

[18] CHOI K,SOBELMAN G E. An efficient CNN accelerator for low-cost edge systems[J]. ACM Transactions on Embedded Computing Systems,2022,21(4):1-20. DOI:10. 1145/3539224.

[19] MAZZIA V,KHALIQ A,SALVETTI F,*et al.* Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application[J]. IEEE Access,2020,8:9102-9114. DOI:10. 1109/ACCESS. 2020. 2964608.

[20] 戴振宇. 基于 ZYNQ 的卷积神经网络加速设计与实现[D]. 呼和浩特: 内蒙古大学,2021.

[21] 李景阳. 基于 Zynq 的热成像人体目标识别算法研究及硬件加速[D]. 成都: 电子科技大学,2023.

[22] YU Hao,LI Sizhao. A higher performance accelerator for resource-limited FPGA to deploy deeper object detection networks[C]//16th International Conference on Anti-Counterfeiting, Security, and Identification. Xiamen; IEEE Press,2022:1-5. DOI:10. 1109/ASID56930. 2022. 9995953.

[23] LI Peng,CHE Cheng. Mapping YOLOv4-tiny on FPGA-based DNN accelerator by using dynamic fixed-point method[C]//12th International Symposium on Parallel Architectures, Algorithms and Programming. Xi'an; IEEE Press,2021:125-129. DOI:10. 1109/PAAP54281. 2021. 9720468.

[24] XU Shanyong,ZHOU Yujie,HUANG Yourui,*et al.* YOLOv4-tiny-based coalgangue image recognition and FPGA implementation[J]. Micromachines,2022,13(11):1983. DOI:10. 3390/mi13111983.

(责任编辑: 黄晓楠      英文审校: 陈婧)