文章编号:1007-5321(2025)05-0083-08 DOI:10.13190/j.jbupt.2024-185

面向 FPGA 的资源高效型二值神经网络卷积加速方法

陈思源, 周晓波, 张世龙
(北京交通大学 电子信息工程学院, 北京 100044)

摘要:针对现场可编程逻辑门阵列(FPGA)在加速卷积运算时,面临的资源需求过高、难以满足嵌入式领域低功耗要求的问题,提出了一种适用于二值神经网络(BNN)的资源高效型 FPGA 卷积加速方法。首先系统分析了卷积层在前向推理过程中,不同并行方案的计算特性及其资源消耗规律,并利用 BNN 特有的低位宽优势提出了高维数据拼接降维存储方案。进而提出了一种针对 BNN 的通道降维旋转缓存(CDR)结构,旨在适度扩展缓存带宽的条件下,实现 BNN 中卷积核内并行和特征图间并行 2 个并行方案的乘积效应。此外,为发挥 CDR 结构的性能优势,设计了专用的 CDR 处理单元,并对加法树结构进行优化。处理单元支持不同流水级别的灵活调整,并可通过重复调用机制实现更高等级的并行计算能力,适应多样化的系统需要。实验结果表明,基于 CDR 结构的 BNN 加速器在 Xilinx 的 XC7Z020 芯片上,可实现显著优于现有同类解决方案的计算密度和存储密度,具备低功耗特性,且推理速度更快,适用于资源、功耗受限的嵌入式平台。

关 键 词:现场可编程门阵列;资源优化;二值神经网络;并行计算;硬件加速
中图分类号:TN791 **文献标志码**:A

A Resource-Efficient Convolution Acceleration Method of Binary Neural Network for FPGA

CHEN Siyuan, ZHOU Xiaobo, ZHANG Shilong
(School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing 100044, China)

Abstract: To address the challenges of excessive resource demands and difficulty in meeting low-power requirements in the embedded domain when field programmable gate array (FPGA) are utilized to accelerate convolution operations, a resource-efficient FPGA-based convolution acceleration method for binary neural networks (BNN) is proposed. First, the computational characteristics and resource consumption patterns of various parallelization schemes during the forward inference process of the convolution layer are systematically analyzed. Leveraging the low bit-width feature of BNN, a high-dimensional data splicing and dimensionality reduction storage scheme is introduced. Subsequently, a channel dimension reduction rotation cache (CDR) structure tailored for BNN is proposed, aiming to achieve the combined benefits of intra-convolution kernel parallelism and inter-feature map parallelism with moderate cache bandwidth expansion. Furthermore, to fully exploit the performance advantages of the CDR structure, a specialized CDR processing unit is designed, and the adder tree structure is optimized. The processing unit supports flexible adjustment of different pipeline levels and can achieve higher-level parallel computing capabilities through a repeated invocation mechanism, adapting to diverse

收稿日期:2024-09-15
基金项目:新疆维吾尔自治区重点研发专项(2022B01008-3);国家自然科学基金项目(92164203,62334006)
作者简介:陈思源(2000—),男,硕士生。
通信作者:周晓波(1973—),男,副教授,硕士生导师,邮箱:xbzhou@bjtu.edu.cn。

system requirements. Experimental results demonstrate that the BNN accelerator based on the CDR architecture achieves significantly superior computational density and storage density compared to existing state-of-the-art solutions when deployed on the Xilinx XC7Z020 chip. It also exhibits low-power characteristics and enables faster inference speed, making it well-suited for resource-and power-constrained embedded platforms.

Key words: field programmable gate array; resource optimization; binary neural network; parallel computation; hardware acceleration

近年来,卷积神经网络(CNN, convolutional neural network)被广泛运用在图像处理的各个领域内,包括图片分类、目标检测、形状分割等[1],展现出了卓越的性能表现。但随着CNN网络深度和参数量的增加,对计算和内存资源的需求也在急速增长,二值神经网络(BNN, binary neural networks)应运而生。

随着计算方式的多元化发展,图形处理器(GPU, graphics processing unit)、专用集成电路(ASIC, application-specific integrated circuit)、现场可编程门阵列(FPGA, field programmable gate array)等多种不同的计算单元被引入加速计算[2],相较于ASIC电路无法编程、灵活性差,GPU能耗比差、不适合边缘端使用的缺点,FPGA在保证运算处理速度的同时,也具备一定的扩展性能。此外,FPGA的低功耗设计对于嵌入式平台具有重大意义[3]。

目前,神经网络的FPGA加速研究主要集中在并行计算和内存优化2个方面。Motamedi等[4]详细总结了卷积层并行计算优化方法,提出了4种不同的卷积并行计算方法。Zhao等[5]先将输入特征图与权重数据全部存储在随机块存储器(BRAM, block random access memory)中,再从中读取数据进行卷积运算,对权重数据进行了重用。Ma等[6]提出一种从输出特征图维度、高度和宽度展开的架构,并且优化循环操作和数据流操作,从而加速卷积神经网络的办法。Zhang等[7]针对FPGA片上BRAM资源无法存储模型权重与正向推理所得中间结果的挑战,采用循环分块等策略完成神经网络部署。

针对以上方面,笔者在前人的研究基础上,完成了以下研究工作。

1)对卷积层的并行性进行分析,量化研究了各类并行方法的资源消耗情况,并且利用BNN的低位宽特性,提出了高维数据拼接降维的方案,随后针对BNN提出了通道降维旋转缓存(CDR, channel dimensionality-reduction rotating-cache)加速方法,包含CDR结构和CDR计算单元。在CDR计算单元中优化了加法树结构,使处理单元能够支持不同级别的流水线处理,且能够通过重复调用实现更高等级的并行能力。

2)基于CDR加速方法,提出了一种BNN硬件加速器架构,并将其部署于Xilinx公司Zynq-XC7Z020芯片上,准确率与中央处理器(CPU, central processing unit)平台测试结果相同,极大地提高了计算性能,高效利用了硬件资源。

1 二值神经网络及并行分析

1.1 二值神经网络

CNN是一种前馈神经网络,由1个输入层、1个输出层和多个隐藏层组成。网络的隐藏层一般由卷积层、池化层、批归一化层和全连接层构成,其中卷积层、池化层和批归一化层用于提取图像特征,全连接层则依靠多层感知网络对提取的图像特征进行分类识别[8]。

BNN是一种基于CNN的模型压缩方法,在正向推理时,除第1层卷积层外,模型的权重参数与中间结果均量化为单比特,使得模型大小与运算量都大大减少。为了减少在网络传输过程中由二值化运算带来的分布信息丢失问题,BNN一般在网络结构中添加批归一化层[9],在保证原始数据分布信息稳定性的同时,也加速了训练过程。

1.2 卷积层并行分析

神经网络的计算重点在于卷积层,因此研究卷积层的加速技术尤为重要,特别是并行计算的优化。从算法角度分析,卷积层的前向计算过程可以被概括为6重嵌套循环的运算过程,具体实现的伪代码如下:

```
for(F_b = 0; F_b < F_n; F_b ++) //特征图间循环
  for(F_r = 0; F_r < H_f; F_r ++) //特征图内行循环
    for(F_c = 0; F_c < W_f; F_c ++) //特征图内列循环
      for(K_b = 0; K_b < K_n; K_b ++) //卷积核间循环
```

for($K_r=0;K_r<H_k;K_r++$) //卷积核内行循环

for($K_c=0;K_c<W_k;K_c++$) //卷积核内列循环

$$F_{\text{out}}[K_b][F_r][F_c]+=F_{\text{in}}[F_b][F_r+K_r][F_c+K_c]W[K_b][F_b][K_r][K_c]$$

从上述伪代码可以看出,嵌套循环的最内层执行累乘加(MAC, multiply accumulate)运算,并行执行 p 个 MAC 运算,可以定义为并行度 p。同时,这 6 重嵌套循环间没有顺序依赖,因此可以任意调换顺序,以不同的方式调换嵌套顺序,并展开内层循环,即可实现不同的并行结构。笔者将卷积运算的 6 重嵌套循环整理为 4 种不同类型的计算并行性,如图 1 所示。

1)卷积核并行。卷积核并行分为卷积核间并行和卷积核内并行 2 种并行计算方法。卷积核间并行指的是卷积核的计算彼此相互独立,所有卷积核共享输入特征图,如图 1(a)所示,此时并行度 $P_{\text{Kbtw}}=6$,即同时进行 6 个卷积核的卷积运算。卷积核内并行指的是 $K_r\times K_c$ 的卷积核与输入特征图中 $K_r\times K_c$ 的滑窗进行卷积时,需要完成 $K_r\times K_c$ 次乘法运算和 $K_r\times K_c$ 次加法运算,如图 1(b)所示,此时 $K_r=K_c=3$,并行度 $P_{\text{Kin}}=9$,即同时完成 9 次乘法运算和 9 次加法运算。

2)特征图并行。特征图并行分为特征图间并行和特征图内并行 2 种并行计算方法。特征图间并行是指对多通道输入特征图进行同时处理,对应通道的卷积核与输入特征图分别进行乘加计算得到中间结果,再将中间结果相加得到单通道输出特征图,如图 1(c)所示,此时并行度 $P_{\text{Fbtw}}=6$,即同时对 6 幅特征图进行卷积运算。特征图内并行是指输入特征图在不同位置共享卷积核,如图 1(d)所示,此时并行度 $P_{\text{Fin}}=3$,即 3 个相同的卷积核同时在 1 幅特征图上进行卷积滑窗运算。

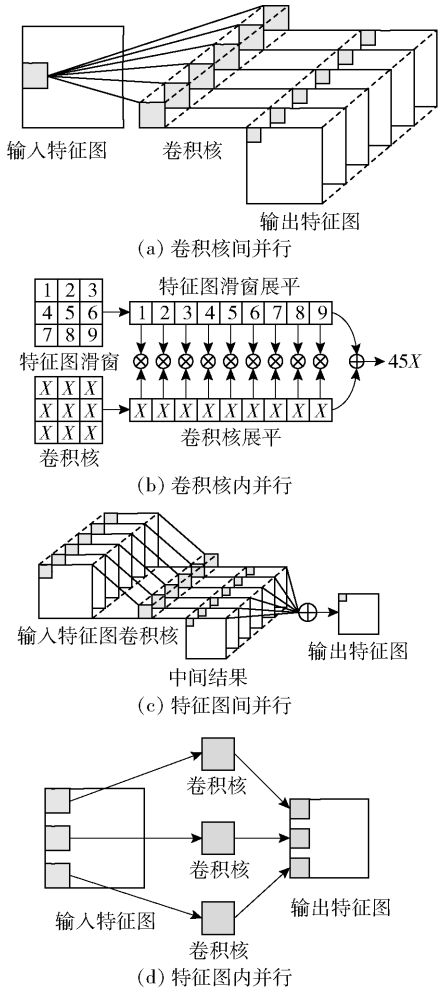

图 1　卷积层并行性分析

2　通道降维旋转缓存加速方法

2.1　通道降维旋转缓存结构设计

在 FPGA 有限的硬件和存储条件下,在 FPGA 上部署的神经网络加速器大多工作在 100 ~ 300 MHz[10],因此一般通过提高并行度的方式提高算力。为了说明卷积层不同类型计算并行性的特点,笔者首先建立一个 FPGA 上一般性的卷积加速器模型,此模型由 3 个缓存单元和处理单元(PE, process element)组成,每次计算时 PE 从特征缓存单元获取输入特征,从权重缓存单元获取权重数据,完成 MAC 运算后,计算结果写入计算缓存单元中。由 1.2 小节分析可知,4 种不同类型的计算并行性会产生不同的计算需求和片上带宽缓存需求,假定并行度均为 p,MAC 计算结果位宽为 n bit,表 1 对比了此情况下一般 BNN 加速器在不同计算并行性下的缓存需求。

表 1　BNN 不同类型计算并行性的缓存带宽需求比较

缓存带宽需求	卷积核内并行 P_{Kin}	卷积核间并行 P_{Kbtw}	特征图内并行 P_{Fin}	特征图间并行 P_{Fbtw}
特征缓存	p	1	p	p
权重缓存	p	p	p	p
结果缓存	n	np	np	n
总需求	$2p+n$	$np+p+1$	$(n+2)p$	$2p+n$

从表1可以看出,在实现相同并行度的条件下,卷积核内并行及特征图间并行的所需缓存带宽需求最小,从该角度出发,卷积核内并行及特征图间并行是进行卷积优化加速的最佳选择。

在BNN中,特征和权重数据位宽仅为1位,因此相较于CNN,这种特性能够轻松实现数据的"降维"操作:将多通道数据的某1维,如通道,按位拼接成1个数据,即可利用存储器的位宽作为数据的通道维数据,而使得多通道数据"降维"成单通道数据,单通道数据的位宽为多通道数据的通道数,且单通道数据的位宽从低到高每1位分别对应多通道数据的第1个通道直至最后1个通道,如图2所示。

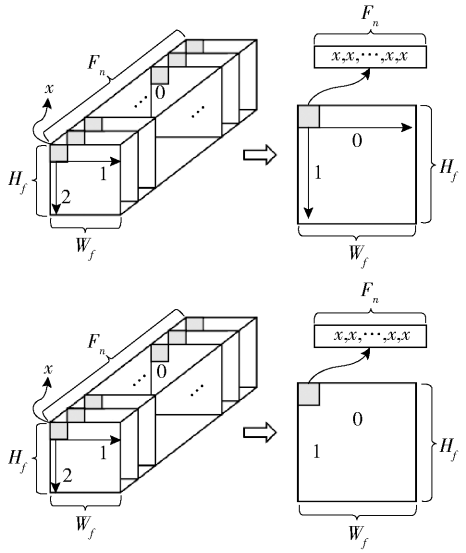

图2 通道二值数据降维

综合以上2点,笔者提出了一种针对BNN的CDR结构,在适度扩展缓存带宽的条件下,实现了卷积核内并行及特征图间并行2个并行方案的并行度乘积效应,即 $P_{\mathrm{CDR}} = P_{\mathrm{Fbtw}}\ P_{\mathrm{Kin}}$。CDR结构如图3所示,主要由Fin_buf、Rotating_reg和Feature_reg组成。

1)Fin_buf。用于缓存上1级PE计算完毕的输出特征图,且以如图2所示方法"降维"排序,排序后的数据位宽为 F_n,并输出至Rotating_reg进行缓存。

2)Rotating_reg。由 H_k+1 个Rotating_buffer组成,每个Rotating_buffer的位宽为 F_n,深度为 W_f,即每个Rotating_buffer可存储全部输入通道中的1行特征数据。本级PE开始计算前,需确认前1级PE已经将特征图全通道的前 H_k 行计算完毕,并经Fin_buf"降维排序"后写入对应的前 H_k 个Rotating_buffer,随后每个时钟周期内 H_k 个Rotating_buffer向Feature_reg提供 $H_k \times F_n$ 个特征数据;与此同时,第 H_k+1 行数据开始写入第 H_k+1 个Rotating_buffer中。当前 H_k 个Rotating_buffer的数据完成计算后,第 H_k+1 个Rotating_buffer中的数据也写入完毕,此时第1个Rotating_buffer中开始写入第 H_k+2 行数据,第2至第 H_k+1 个Rotating_buffer向Feature_reg提供特征数据。以此类推,计算开始后总有 H_k 个Rotating_buffer在向Feature_reg提供数据,而另一个Rotating_buffe缓存从Fin_buf内输出的数据,整个过程好似Rotating_reg在旋转。

3)Feature_reg。由 W_k+1 个Feature_buffer组成,每个Feature_buffer能够缓存 $H_k \times F_n$ 个特征数据,即每个时钟周期内Rotating_reg中读取的数据量。前 W_k 个Feature_buffer依次写满后,PE单元即可开始工作,Feature_reg在每个时钟周期内向PE单元提供 $W_k \times H_k \times F_n$ 个特征数据;下个时钟周期来临后,第 W_k+1 个Feature_buffer也已写满,此时第2至第 W_k+1 个Feature_buffer向PE单元提供 $W_k \times H_k \times F_n$ 个特征数据,第1个Feature_buffer准备写入下个周期Rotating_reg输出的数据,整个过程好似Feature_reg在旋转。

在整个计算过程中,通道降维旋转缓存结构通过2级缓存旋转,利用Rotating_reg实现了特征数据的行复用,利用Feature_reg实现了特征数据的列复用,以此实现卷积核内并行;并结合Fin_buf提供的二值"降维"数据实现了特征图间并行计算。

2.2 通道降维旋转缓存计算单元设计

在BNN中,卷积核形状一般为正方形,且一般大小不超过7×7,假设卷积核大小 $H_k = W_k = K$。对于 K^2 个单bit卷积运算,可使用异或运算逻辑全并行完成;对于 K^2 个数的求和运算,硬件设计中一般采用加法树实现[11],即通过补零的方式将数量从 K^2 个扩充至 $2^{\mathrm{lb}K^2}$ 个,再两两逐级累加直至得到总和,显然可计算出经典加法树,此情况下所需加法器为 $2^{[\mathrm{lb}K^2]}-1$,寄存器个数为 $2^{[\mathrm{lb}K^2]+1}-1$,时钟周期个数为 $[\mathrm{lb}K^2]$,其中[ ]为向上取整运算符。但该方法会造成较高的硬件资源消耗和较大的带宽需求,例如取 $K=6$ 和取 $K=8$ 时,加法树所需寄存器个数均为127,加法器个数均为63,时钟周期均为4。显然,后者虽然求和个数约为前者的2倍,但消耗的计算、存储和时间资源却是相同的,即出现了近乎一半的资源浪费。因此针对这个问题,笔者改进的加法

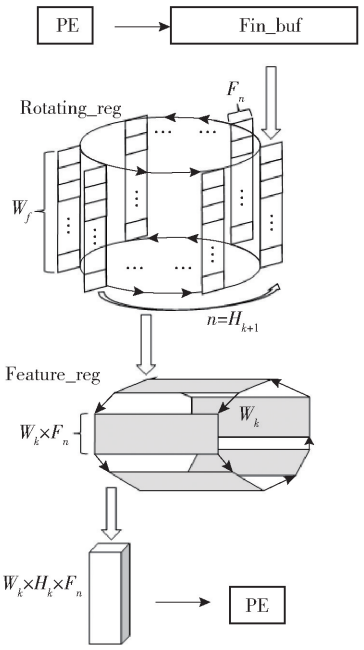

图 3　通道降维旋转缓存加速结构

树如下。

1)若数据个数为偶数,则两两相加;若数据个数为奇数,则除最后 1 个数外,进行两两相加,直至最后 1 级。

2)针对 BNN 运算资源消耗少、逻辑简单、关键路径延时短的特点,利用多级寄存的方式以优化时钟周期数和寄存器使用。假定寄存级别为 N,相同输入下经典加法树时钟周期数为 T,此时消耗时钟周期为 N,若 N 为 T 的任意约数,即每经过 T/N 级加法器进行 1 次数据寄存;若 N 不为 T 的任意约数,前 $N-1$ 次寄存前经过的加法器级数为 $[T/N]-1$,最后 1 次寄存前完成所有加法器运行。

以卷积核大小 $H_k=W_k=K$,寄存级别 N 为例,通过数学归纳法可得,笔者改进加法树所需加法器为 K^2-1,寄存器数量为 $\sum_{i=1}^{N-1}\left[\frac{K^2}{2^{i\left(\left[\frac{T}{N}\right]-1\right)}}\right]+1$,时钟周期数为 N。表 2 以 $K=3,7$,$N=2,3$ 为例,对比了经典加法树和笔者改进加法树之间的资源消耗。

表 2　加法树对比

结构	加法器	寄存器	时钟数
经典树	15($K=3,N=2$)	31($K=3,N=2$)	4($K=3,N=2$)
	63($K=7,N=3$)	127($K=7,N=3$)	6($K=7,N=3$)
改进树	8($K=3,N=2$)	4($K=3,N=2$)	2($K=3,N=2$)
	48($K=7,N=3$)	16($K=7,N=3$)	3($K=7,N=3$)

将笔者改进加法树应用至 PE 中,PE 结构如图 4 所示,单个 PE 中有 F_n 个相同的异或累加运算(XAC, xor accumulate)结构,XAC 结构的异或(XOR, exclusive or)累加结果经过加法树得到最终计算结果。XAC 结构如图 4 中灰色方格所示,XOR 阵列用于实现并行度为 $W_k\times F_n$ 的卷积运算,加法树用于实现 XOR 结果的累加操作。

包含 F_n 个 XAC 结构的 PE 结构可实现并行度为 $H_k\times W_k\times F_n$ 的卷积运算,且仍可根据计算需要,重复调用该 PE 结构,完成更高的并行度运算。为提高工作效率,PE 结构和 XAC 结构中的加法树均采取流水线操作,2 者寄存级别相加为 W_k。具体的运算流程为:经过 W_k 个时钟周期的特征图寄存操作后,每个时钟周期内,PE 可加载 $H_k\times W_k\times F_n$ 个权重-特征对并完成 XAC 运算,运算结果通过 W_k 级寄存的加法树输出至下 1 级 Fin_reg 结构中进行缓存。每当 1 行遍历完成,随后的 W_k 个时钟周期将完成下 1 行的特征图寄存并继续完成并行度为 $H_k\times W_k\times F_n$ 的 XAC。之后不断重复上述过程,直至计算结束。

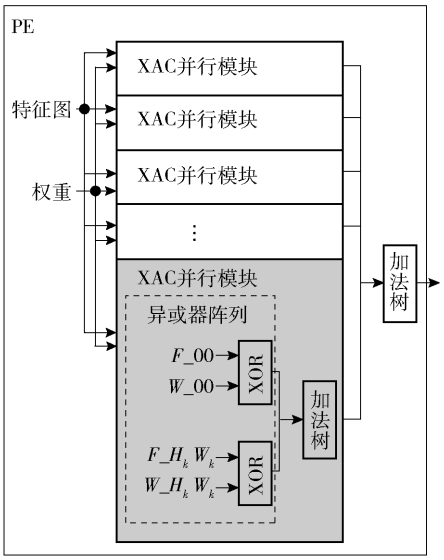

图 4　通道降维旋转缓存计算单元

笔者在 2.1 和 2.2 小节中探讨了 CDR 加速方法的结构设计及其计算单元的具体实现,为进一步阐明该加速方法的运作机制,笔者给出 CDR 加速方法的伪代码如下:

```
for(K_b=0;K_b<K_n;K_b++)                    //Loop1
  for(F_r=0;F_r<H_f;F_r++)                  //Loop2
    for(F_c=0;F_c<W_f;F_c++)                //Loop3
      parallel{
        for(F_b=0;F_b<F_n;F_b++)
```

for($K_r=0;K_r<H_k;K_r++$) //Loop4
for($K_c=0;K_c<W_k;K_c++$)
sum$[F_b][K_r][K_c]=F_{in}[F_b][F_r+K_r][F_c+K_c]W[K_b][F_b][K_r][K_c];$}
for($F_b=0;F_b<F_n;F_b++$) //Loop5
parallel{
for($K_r=0;K_r<H_k;K_r++$) //Loop6
for($K_c=0;K_c<W_k;K_c++$)
$F_{out}[K_b][F_r][F_c]+=$sum$[F_b][K_r][K_c];$}

其中:Loop1 表示卷积核间的循环运算,Loop2 表示特征图内行循环运算,Loop3 表示特征图内列循环运算,由于该3个循环只涉及输出位置的改变,因此无须计算、存储资源,通过计数即可完成;Loop4 循环完成全通道并行卷积运算,Loop5 用于累加多通道间卷积核的运算结果,Loop6 用于累加单通道内卷积核的运算结果。Loop4 循环和实现运算结果累加的 Loop5 和 Loop6 循环构成 PE,权重和特征数据均以如图2所示通道二值数据降维操作排序存储。

3 仿真实验结果与分析

3.1 实验环境

本实验在 4.7 GHz 主频,8 核 16 线程的锐龙 R7-6800H 处理器的计算机硬件平台环境下,使用 Python 3.8 和 Tensorflow 2.3,利用改良国家标准与技术研究院(MNIST, modified national institute of standards and technology)数据集对模型进行训练,并在 Zynq 异构 FPGA 平台上实现前向推理加速。异构 FPGA 板载芯片为 XC7Z020-2CLG400I,使用 Vivado 2018.3 工具进行开发和仿真。Zynq 是 Xilinx 公司于2010年4月推出的一款搭载异构高性能嵌入式处理器的边缘端计算平台。FPGA 在 Zynq 中作为双重核心进阶精简指令集机器(ARM, advanced risc machine) Cortex A9 处理器的硬件协处理器,主要职责是为计算机视觉、神经网络等复杂算法提供硬件加速计算服务,并通过丰富的高级可扩展接口(AXI, advanced extensible interface)总线与双核 ARM Cortex-A9 处理器进行实时的数据交互和控制,其中 Zynq 平台中的 FPGA 部分即可编程逻辑端(PL, programmable logic),ARM 部分即处理器系统端(PS, processor system)。

3.2 实验模型

笔者在经典的 Lenet-5 卷积神经网络上进行修改,提出了 Lenet-B5 二值神经网络,卷积层特征数据不填充,池化层采取最大池化采样器,详细参数如表3所示。

表3 Lenet-B5 神经网络模型

层名称	层结构
卷积层1	卷积核大小5×5,个数30,步长1
池化层1	池化大小2×2,步长2
批归一化层1	输出二值化
卷积层2	卷积核大小5×5,个数20,步长1
池化层2	池化大小2×2,步长2
批归一化层2	输出二值化
全连接层1	输出神经元个数100
批归一化层3	输出二值化
全连接层2	输出神经元个数10

3.3 实验模型

在神经网络前向推理时,主要的运算在卷积部分,主要的数据搬运、排序在全连接部分,因此为了发挥异构 FPGA 平台的优势,将 CDR 结构放在 PS 端完成,PE 部署在 PL 端。总体单元设计如图5所示,主要包含 PS 端的控制单元、CDR 结构和 PL 端的加速模块,PS 和 PL 之间由控制单元获取中断状态,通过 AXI 总线协议读写双倍数据率(DDR, double data rate)存储控制器实现数据的交互。

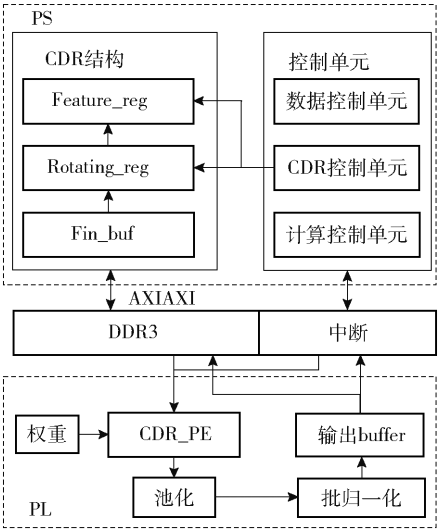

图5 加速系统架构图

3.4 实验结果

笔者测试使用 MNIST 测试集,PL 端和 PS 端的总运算量分别为2 957 和21万次的乘加运算,Zynq 异构 FPGA 加速器运行在 143 MHz 时钟条件下。如

表 4 所示，FPGA 加速系统计算准确率与 CPU 平台相比并无降低，FPGA 在加速层 1 和加速层 2 的处理时间分别为 0.20 ms 和 0.03 ms，CPU 的处理时间为 0.95 ms 和 4.49 ms，加速比分别为 3.8 和 148.7。

表 4　不同平台性能对比

平台	准确率/%	加速层 1 时间/ms	加速层 2 时间/ms
CPU	96.99	0.95	4.49
FPGA	96.99	0.20	0.03

Xilinx FPGA 的主要资源包括查找表（LUT，look-up table）、触发器（FF，flip-flop）、BRAM 和数字信号处理器（DSP，digital signal processor）等。经综合和实现后得到的加速层 1 和加速层 2 资源使用情况如表 5 所示，括号内为资源占用百分比。

表 5　FPGA 资源使用情况

加速层	LUT	BRAM	FF
开发板	53 200	140	106 400
加速层 1	355（0.7%）	3（2.1%）	507（0.5%）
加速层 2	1 890（3.6%）	6（4.3%）	3 154（3.0%）

LUT 用于实现逻辑电路的布尔函数，而 FF 用于存储和同步时序电路中的状态信息，通过将卷积的乘加运算转换为 XOR 逻辑运算和流水加法树操作，加速模块的 LUT 和 FF 的资源占用率均保持在 4% 以下，仍有增加并行处理以提升计算速度的潜力，BRAM 的资源使用率在 5%，仍有增加卷积核数量以提升模型对多种特征进行识别的能力。尤其值得注意的是，本研究针对 BNN 进行加速，因此无须使用 DSP 资源。

如表 6 所示为笔者与其他 FPGA 硬件卷积加速方法的比较，各个系统均使用相同的 FPGA 开发板，且都运行在 143 MHz 时钟频率下，括号内为资源占用百分比。由于各文献用于加速的网络结构不同，因此引入每秒千兆操作数（GOPS，giga operations per second）作为有效算力的衡量标准，即对加法和乘法进行计数：每个二进制的异或运算或加法计数为 1 次操作，并且应用计算密度（每 1 k 查找表下的有效算力）、存储密度（每个 BRAM 下的有效算力）和能耗比（每 W 下的有效算力）来评估性能。从表 6 可以看出，笔者提出的卷积加速方法计算密度为 55.9 GOPS/kLUTs，显著高于其他 3 种方案；存储密度为 14.3 GOPS/BRAM，比现有解决方案提升了 29.1%。

表 6　与其他文献 FPGA 硬件卷积加速方法对比

加速方法	量化精度/bit	LUT 消耗/kLUTs	LUT 占用率/%	BRAM 消耗	BRAM 占用率/%	功耗/W	性能/GOPS	计算密度/（GOPS·kLUTs^{-1}）	存储密度/（GOPS·BRAM^{-1}）	能耗比/（GOPS·W^{-1}）
文献[5]	1～2	46.9	88.2	94.0	67.1	4.7	318.9	6.8	3.4	67.9
文献[12]	1	14.5	27.3	32.0	22.9	2.3	329.5	22.7	10.3	143.3
文献[13]	1	29.6	55.6	103.0	73.6	3.3	722.0	24.4	7.0	218.8
文献[14]	1	38.9	73.1	123.0	87.9	–	378.0	9.7	3.1	–
文献[15]	1	37.3	70.1	130.0	92.9	4.4	193.8	5.2	1.5	44.0
笔者方法	1	8.7	16.4	17.5	12.5	1.8	128.6	55.9	14.3	71.4

4　结束语

笔者通过系统分析前向推理过程中，卷积层不同维度并行计算的原理及资源消耗情况，利用 BNN 的低位宽优势，设计了高维数据拼接降维存储方案，随后结合卷积核间并行和特征图间并行 2 种并行方案提出了一种针对 BNN 的 CDR 结构，并针对 CDR 结构设计了专用 PE 并优化了加法树结构，最后在 FPGA 上构建了一个完整 BNN 加速系统。通过实验分析，笔者提出的卷积加速方法在不损失计算精度的情况下，计算密度为 55.9 GOPS/kLUTs，显著高于现有解决方案，存储密度为 14.3 GOPS/BRAM，比现有解决方案提高了 29.1%，且功耗仅为 1.8 W。实验表明，笔者提出的加速方法具有较高的资源利用密度，适用于嵌入式平台，具有较高的推广价值。

参考文献：

[1] XU F, LUO Y, SUN C, et al. Improved convolutional neural network for traffic scene segmentation[J]. Computer Modeling in Engineering and Sciences, 2024, 138(3): 2691-2708.

[2] BOBDA C, MBONGUE J M, CHOW P, et al. The future of FPGA acceleration in datacenters and the cloud[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2022, 15(3): 1-42.

[3] HONG H, CHOI D, KIM N, et al. Survey of convolutional neural network accelerators on field-programmable gate array platforms: Architectures and optimization techniques[J]. Journal of Real-Time Image Processing, 2024, 21(3): 64-84.

[4] MOTAMEDI M, GYSEL P, AKELLA V, et al. Design space exploration of FPGA-based deep convolutional neural networks[C]//2016 21st Asia and South Pacific Design Automation Conference, Macao, China, 2016: 575-580.

[5] ZHAO R, SONG W, ZHANG W, et al. Accelerating binarized convolutional neural networks with software-programmable FPGAs[C]//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017: 15-24.

[6] MA Y, CAO Y, VRUDHULA S, et al. Optimizing the convolution operation to accelerate deep neural networks on FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018, 26(7): 1354-1367.

[7] ZHANG C, SUN G, FANG Z, et al. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks[C]//Proceedings of the ACM Turing Award Celebration Conference-China 2023, Wuhan, China, 2023: 47-48.

[8] LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, USA, 2015: 3431-3440.

[9] IOFFE S, SZEGEDY C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International Conference on Machine Learning, Lille, France, 2015: 448-456.

[10] SPAGNOLO F, CORSONELLO P, FRUSTACI F, et al. Efficient implementation of signed multipliers on FPGAs[J]. Computers and Electrical Engineering, 2024, 116: 109217.

[11] GUO K, ZENG S, YU J, et al. A survey of FPGA-based neural network inference accelerators[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2019, 12(1): 1-26.

[12] NAKAHARA H, FUJII T, SATO S. A fully connected layer elimination for a binarized convolutional neural network on an FPGA[C]//2017 27th International Conference on Field Programmable Logic and Applications (FPL), IEEE, 2017: 1-4.

[13] GUO P, MA H, CHEN R, et al. FBNA: A fully binarized neural network accelerator[C]//2018 28th International Conference on Field Programmable Logic and Applications (FPL), IEEE, 2018: 510-513.

[14] DE SOUSA A L, VÉSTIAS M P, NETO H C. Multi-model inference accelerator for binary convolutional neural networks[J]. Electronics, 2022, 11(23): 3966-3986.

[15] XIANG M, TEO T H. Implementation of binarized neural networks in all-programmable system-on-chip platforms[J]. Electronics, 2022, 11(4): 663-673.