DOI:10.7544/issn1000-1239.20210142
59(7):1409-1427,2022

FAQ-CNN:面向量化卷积神经网络的嵌入式FPGA可扩展加速框架

谢坤鹏[1,2] 卢 冶[1,2,3] 靳宗明[1,2] 刘义情[1,2] 龚 成[1,2] 陈新伟[4] 李 涛[1,2,3]

[1](南开大学计算机学院 天津 300350)

[2](天津市网络与数据安全技术重点实验室(南开大学) 天津 300350)

[3](计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

[4](福建省信息处理与智能控制重点实验室(闽江学院) 福州 350108)

(xkp@mail.nankai.edu.cn)

FAQ-CNN: A Flexible Acceleration Framework for Quantized Convolutional Neural Networks on Embedded FPGAs

Xie Kunpeng[1,2], Lu Ye[1,2,3], Jin Zongming[1,2], Liu Yiqing[1,2], Gong Cheng[1,2], Chen Xinwei[4], and Li Tao[1,2,3]

[1](*College of Computer Science, Nankai University, Tianjin* 300350)

[2](*Tianjin Key Laboratory of Network and Data Security Technology (Nankai University), Tianjin* 300350)

[3](*State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences), Beijing* 100190)

[4](*Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University), Fuzhou* 350108)

Abstract Quantization can compress convolutional neural network (CNN) model size and improve computing efficiency. However, the existing accelerator designs for CNN quantization are usually faced with the challenges of various algorithms, poor reusability of code modules, low efficiency of data exchange and insufficient utilization of resources, and so on. To meet these challenges, we propose a flexible acceleration framework for the quantized CNNs named FAQ-CNN to optimize accelerator design from three aspects of computing, communication and storage. FAQ-CNN can support rapid deployment of quantized CNN model in the form of software tools. Firstly, a component for quantization algorithms is designed to separate the calculation part from the process of value projection in quantization algorithm; the optimization techniques such as operator fusion, double buffering and pipeline are also utilized to improve the execution efficiency of CNN inference task in parallel. Then, the hierarchical and bitwidth-independent encoding and parallel decoding method are both proposed to efficiently support batch transmission and parallel computing for low bitwidth data.

收稿日期:2021-02-26;**修回日期**:2021-08-24

基金项目:国家重点研发计划项目(2018YFB2100304);国家自然科学基金项目(62002175);计算机体系结构国家重点实验室(中国科学院计算技术研究所)开放课题(CARCHB202016);天津市优秀科技特派员项目(21YDTPJC00380);福建省信息处理与智能控制重点实验室(闽江学院)开放基金项目(MJUKF-IPIC202105);中国高校产学研创新基金项目(2020HYA01003)

This work was supported by the National Key Research and Development Program of China (2018YFB2100304), the National Natural Science Foundation of China (62002175), the Open Project Fund of State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences) (CARCHB202016), the Special Funding for Excellent Enterprise Technology Correspondent of Tianjin (21YDTPJC00380), the Open Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (MJUKF-IPIC202105), and the Innovation Fund of Chinese Universities Industry-University-Research (2020HYA01003).

通信作者:卢冶(luye@nankai.edu.cn)

Finally, the resource allocation optimization model which can be transformed into an integer nonlinear programming problem is established for FAQ-CNN; the heuristic pruning strategy is used to reduce design space size. The extensive experimental results show that FAQ-CNN can support almost all kinds of quantized CNN accelerators efficiently and flexibly. When the activation and weight value are set to 16 b, the computing performance of FAQ-CNN accelerator is 1.4 times that of the Caffeine. When 8 b configuration is applied, FAQ-CNN can achieve the superior performance by 1.23TOPS.

Key words convolutional neural network quantization; decoupling of quantization algorithm; encode and decode in parallel; on-chip resource modeling; accelerator design

摘 要 卷积神经网络(convolutional neural network, CNN)模型量化可有效压缩模型尺寸并提升CNN计算效率.然而,CNN模型量化算法的加速器设计,通常面临算法各异、代码模块复用性差、数据交换效率低、资源利用不充分等问题.对此,提出一种面向量化CNN的嵌入式FPGA加速框架FAQ-CNN,从计算、通信和存储3方面进行联合优化,FAQ-CNN以软件工具的形式支持快速部署量化CNN模型.首先,设计面向量化算法的组件,将量化算法自身的运算操作和数值映射过程进行分离;综合运用算子融合、双缓冲和流水线等优化技术,提升CNN推理任务内部的并行执行效率.然后,提出分级编码与位宽无关编码规则和并行解码方法,支持低位宽数据的高效批量传输和并行计算.最后,建立资源配置优化模型并转为整数非线性规划问题,在求解时采用启发式剪枝策略缩小设计空间规模.实验结果表明,FAQ-CNN能够高效灵活地实现各类量化CNN加速器.在激活值和权值为16 b时,FAQ-CNN的加速器计算性能是Caffeine的1.4倍;在激活值和权值为8 b时,FAQ-CNN可获得高达1.23TOPS的优越性能.

关键词 卷积神经网络量化;量化算法解耦;并行编解码;片上资源建模;加速器设计

中图法分类号 TP391

由于卷积神经网络(convolutional neural network, CNN)往往规模庞大、参数量众多,难以直接部署于边缘计算平台,因此为迁移CNN到资源受限的嵌入式现场可编程门阵列(field programable gate array, FPGA)平台来加速边缘应用计算,研究人员提出了多种CNN量化方法,在符合应用要求的基础上降低CNN的计算规模和存储需求[1-2].CNN的权值和激活值经过量化操作后,只需低位宽的算术运算单元和少量的存储资源就可以部署CNN模型.FPGA凭借其独特的灵活性、可编程性、高层次综合设计和出色的能效比[3],成为设计CNN加速器的首选平台[4-6].因此,利用嵌入式FPGA来实现量化模型的低位宽加速器已成为当前的研究热点[7-10].然而,以往的CNN加速器设计通常存在3个问题:

1) CNN模型量化算法与相应硬件加速器的设计紧耦合,导致模型重构需要相应的硬件修改量大且复杂,无法高效适配不同量化方法,代码复用效率极低.传统量化CNN加速器的设计是根据量化方法的数据格式定义和数据位宽声明来设定计算规则,然后将计算规则映射成为相应的硬件配置[4-5].研究人员往往期望以类似"即插即用"的方式将各种量化CNN快速地部署于FPGA来验证其硬件效率[11-12].然而,不同的量化方案在实现过程中,通常具有自身特殊的数据格式和运算规则,这些差异导致重构时无法高效复用.具体来说,数据格式会直接影响数据传输模式和读写方式,重构时需要针对特定数据格式重新设定编码规则和数据重组织方案;而特定的运算规则又需要重新定制算子才能支持高效计算,另外计算的高度并行需求也会增加硬件重构时的难度.

2) 已有的工作多是利用切片处理方式从不同的角度将大块计算分解为可部署到FPGA上的多个小块计算,并优化数据交换机制来减少片上存储和片外存储之间的通信代价[10,13].但是,低位宽数据在片上片外数据交换过程中,由于缺少对数据的有效组织,导致带宽资源利用不充分、并行读写效率低,从而成为制约高效计算的瓶颈.尽管可将多个数据项打包为单个宽字[14-15],然后在每个周期并行地传输这些数据项,但是未进行编码设计和数据重组织

则会降低并行解码的执行效率.此外,并行解码操作需要存储数据的硬件资源独立分布,才能避免片上数据并行访问冲突.在计算方面,并行计算同样也需要合理的数据存储资源分布,才能保证数据的无冲突并行访问.因此,以往的设计方案需要大量的片上存储资源才能同时满足并行解码和并行计算的需要.

3) 实际部署 CNN 时需要调整计算并行规模、存储资源类型、通信缓冲大小等一系列参数配置以符合 FPGA 片上资源的约束并进行性能优化,但片上资源配置与优化严重依赖人工经验,缺乏建模分析和明确的理论依据来支持决策,导致资源未有效利用、FPGA 性能发挥不充分.具体来讲,仅凭人工经验难以精准决策要分配哪种资源、分配多少资源来用于计算操作、数据存储和数据通信.尽管有一些相关研究尝试对 CNN 加速器的性能和资源进行分析来解决 FPGA 片上资源如何配置的问题,但未能构建全面地覆盖所有资源的分析模型,造成资源配置方法具有局限性.例如,利用 Roofline 搜索最优的 DSP 资源配置[16],但 LUT 等其他片上资源并未涉及和充分利用.因此,对于 CNN 加速器来说,缺乏 FPGA 片上资源(如 LUT,DSP,BRAM)利用率与 CNN 推理延迟的联合分析,会使加速器的设计和优化缺乏理论依据,进而难以进行资源合理配置,无法充分发挥硬件性能.

目前,学术界和工业界缺乏针对量化 CNN 模型提供框架级支撑的灵活通用 FPGA 加速器设计研究工作,现有框架级工作主要集中于适配专用、特定量化方法的 FPGA 加速器设计研究.这是由于 FPGA 设计流程相对复杂且灵活度较高,需要研究人员具备软硬件协同设计能力,既要熟悉硬件研发方法又要对算法执行流程[17]一清二楚,无形中提高了 FPGA 加速框架的设计门槛,也造成了当前面向 CNN 的 FPGA 加速框架相关研究的匮乏.相比而言,GPU 平台上的算法加速研究和应用却很充分,先后出现了多种加速库和框架工具来加速深度神经网络并简化其部署过程,如 cuDNN[18],cuBlas[19],TensorFlow 等功能强大又丰富的类库和框架工具,而 FPGA 平台上的量化 CNN 加速器框架相关内容还未被深入探索.尽管 Xilinx 推出的 VITIS AI 工具可支持 Caffe 和 TensorFlow 的模型直接进行量化并部署到目标平台[20],但是该工具所采用的量化方法固定,难以更改或拓展,而量化 CNN 加速器的研究人员亟需快速集成各种类型量化方法来验证硬件效率[12].

因此,为解决上述问题,本文提出一种面向量化 CNN 模型的嵌入式 FPGA 加速框架 FAQ-CNN,从计算并行、数据通信与数据存储 3 方面来进行相关设计、方案实现和联合优化.FAQ-CNN 把量化方法相关操作抽象成模板化的量化组件,以此支持各种量化算法的灵活接入和量化模型快速部署.CNN 模型量化和方案部署研究人员可利用 FAQ-CNN 框架中的缺省量化组件或自定义新量化组件,以类似“即插即用”的方式来快速定制量化 CNN 加速器,从而减少设计重构带来的代价.为提升定制量化 CNN 加速器的性能,FAQ-CNN 利用搜索工具实现硬件参数快速自动配置,从而发挥各种量化方法的优势并避免框架通用性带来的性能损失.在加速器设计方面,FAQ-CNN 分别实现卷积、池化、ReLU 激活和全连接 4 种不同的 CNN 算子,以此作为建立通用 CNN 结构的基础,并运用算子融合、双缓冲和流水线等优化技术,提升 CNN 推理任务内部并行的执行效率.在数据传输方面,FAQ-CNN 采用高位宽的数据传输模式实现单周期交付多数据,并结合数据预处理、数据并行解码和并行计算等机制进行协同优化.此外,FAQ-CNN 综合分析片上资源利用情况并构建资源配置模型,以 CNN 推理最小延迟为性能目标,以片上资源为约束,来支持搜索工具寻求最佳的片上资源配置方案.本文相关实验验证了 FAQ-CNN 能够高效地实现如 BNN[1],TNN[2],μL2Q[12],DoReFa-Net[21],VecQ[11],PoT[22] 等量化 CNN 加速器.以 8 b 位宽的 DoReFa 量化算法加速器为例,其在 Xilinx ZCU102 平台上 FAQ-CNN 可获得高达 1.23TOPS 的计算性能.

本文的主要贡献包括 3 个方面:

1) 提出面向量化 CNN 的 FPGA 加速框架 FAQ-CNN.软件工具 FAQ-CNN 可提供快速部署量化 CNN 模型的解决方案,将量化方法的相关操作模板化为 FAQ-CNN 框架的量化组件,并提供 2 种不同粒度的 CNN 硬件加速器实现方式;通过算子融合将卷积层的输出结果作为激活层和池化层的输入数据,并采用双缓冲存储张量数据,从而实现 4 段流水线来提升 FAQ-CNN 迭代时的并行执行效率.

2) 提出针对数据特征的分级编码和位宽无关编码的方案,设计低位宽与高位宽数据的编解码规则,高效利用带宽传输数据.在数据传输前,将片外数据重组织为按固定维度展开的连续分布数据;在数据解码时,通过存储资源复用机制来满足并行解码和并行计算的资源需求,降低总体资源消耗.

3) 建立资源配置优化模型并转为整数非线性规划问题,将输入通道与输出通道的切片因子与权值存储方式设定为超参数来探索设计空间.在求解时采用启发式剪枝策略缩小设计空间规模,借助CPU-GPU异构平台来加速参数搜索过程,快速完成相应的FPGA片上资源的最佳配置.

1 相关工作

关于FPGA平台的CNN加速器研究集中于上层CNN量化方法和底层硬件设计2个方面[23-24].上层CNN量化方法侧重降低数据位宽来契合FPGA硬件计算要求[1];底层硬件设计侧重从CNN计算、数据通信、存储等方面来优化加速器性能[13].

在CNN模型量化方法方面,量化可分为超低位宽[1]、线性[21]和非线性[22]这3类方法.对于超低位宽,Courbariaux等人[1]提出了二值化网络BNN,将模型权值量化为1或−1这2个值,仅用1 b即可表示模型的参数.Zhao等人[25]基于FPGA平台实现了BNN,将其模型权值数据全部置于片上存储器,避免片外数据加载高延迟.Xilinx研究实验室提出了基于FPGA的BNN加速框架FINN[26],所实现的加速器能够在CIFAR-10数据集上获得283 μs的推理延迟.Li等人[2]提出了三值化网络TNN,将模型权值量化为1,−1或0,相比于BNN,TNN能够有效提升模型分类精度.Prost-Boucle等人[5]部署TNN量化算法到FPGA平台并获得了16.7TOPS的计算性能.Wang等人[27]提出基于BNN和TNN的混合精度神经网络,并采用多种组合方案来权衡模型的准确性和计算复杂度.对于线性量化,Gong等人[11]提出VecQ量化方法,利用范数形式化表示量化值和全精度值之间的偏差,能够在给定位宽下最小化量化误差.Zhou等人[21]提出了DoReFa-Net,量化反向传播过程中的梯度值为低位宽定点数,从而能够利用FPGA平台加速CNN模型的训练和推理过程.Wang等人[28]提出HAQ量化框架,利用增强学习机制优化量化策略,对模型的各层采取不同的量化策略,能够有效降低推理延迟并减少量化造成的精度损失.对于非线性量化,Miyashita等人[22]提出了PoT量化算法,将模型权值量化为0或2的幂.Jing等人[29]部署PoT量化算法到FPGA平台,利用幂计算的性质将乘法和累加运算转换为利用LUT实现的移位运算.Li等人[30]提出APoT算法,将浮点数量化为多个PoT值(2的幂)的和,从而更好地拟合权值分布.Tambe等人[24]提出AdaptiveFloat量化算法,利用指数和底数来表示量化后的数值,不仅扩大了数值表示范围,而且支持硬件的高效实现.Jain等人[31]提出BiScaled-Fxp定点数表示方式,采用2个不同的缩放因子来拟合长尾分布,按比例分别缩放模型权值中较大数值和较小数值,并成功部署于FPGA平台.

在CNN模型计算方面,主要的计算量集中于卷积层,Cong等人[32]证明了卷积层的计算量占据整个模型计算量的90%.针对卷积层中大量的循环操作,Zhang等人[16]采用多种嵌套循环优化技术来提升卷积层的计算效率,如循环流水线、循环展开和循环平铺.卢冶等人[33]从数据访问独立性的角度提出通道并行计算的方案,引入循环切割因子并在通道维度展开循环,利用数据流优化和缓存优化技术加速CNN的卷积计算.Liang等人[34]利用Winograd算法降低卷积层的计算复杂度,提高卷积层的处理速度.然而,Winograd算法效率依赖卷积步长的选取,当步长为1时计算效率较高,而其他情况下算法效率不够理想.为了部署CNN到配备大规模硬件资源的FPGA平台上,有研究人员从处理器结构方面来加速卷积计算.例如,Wei等人[10]提出利用脉动阵列来进行卷积层计算的方案,能够以280 MHz的工作频率处理CNN推理任务,大幅度降低推理延迟.Cong等人[35]利用多面体模型自动化设计CNN的脉动阵列结构,探索了从计算需求到硬件结构的最佳映射方案.然而,这些基于脉动阵列的计算方案会消耗大量局部存储资源.此外陈桂林等人[36]将卷积计算映射为2维阵列的计算和相应的数据流调度,并通过加速器实例进行了相应验证.

在数据交换效率方面,CNN加速器片上与片外的数据存储和访问方式对加速器的处理速度提升至关重要.Du等人[37]提出ShiDianNao,将所有的模型参数置于片上SRAM,消除对DRAM的访问,避免因片外数据访问引起的高功耗和高延迟.Zhang等人[13]提出将全连接层的计算任务映射到卷积的计算引擎中,通过探究输入特征图和权重数据的映射方式并利用共享机制,来减少对片外存储器的访问时间.Li等人[38]提出矩阵分解的方案,将全连接层的矩阵乘法分解为多个小规模矩阵乘法,并采用批处理的模式来缓解外存访问压力.除了访存优化,数据读取方式优化也能够提高数据传输效率.对片外存储器非连续存放的权值数据进行重新排列,可增大数据读取的猝发长度,进而提高实时带宽[13,39].此外,

Cong 等人[14]提出了利用高位宽模式传输数据的方案,将多个 32 b 的字合并为 1 个 512 b 的宽字来实现单周期传输多数据,以充分利用带宽资源.

在片上资源配置优化和设计空间探索方面,Zhang 等人[16]提出 roofline 模型,利用带宽需求和实时处理性能 2 个指标建模,最小化卷积层处理时间.Motamedi 等人[40]联合各类 CNN 设计方案与硬件平台参数提出基于 FPGA 的 CNN 加速器性能评估模型,并利用设计空间自动化搜索工具获取给定平台的最佳实现方案.Mu 等人[41]提出 LoopTrees 数据结构,对 CNN 加速器进行资源建模,通过 2 阶段先后的粗粒度和细粒度分析来获取精准的资源模型.这些工作在搜索最优的 CNN 硬件实现结构时,主要针对部分片上资源进行建模,如 DSP 和 BRAM.然而,作为片上资源重要组成部分的 LUT 未被充分探索,导致片上资源利用不充分.LUT 的使用会增大建立资源模型的复杂度,而现有的工作缺乏关于 LUT 资源利用方面的探索.

2 FAQ-CNN 框架设计

本节首先介绍 FAQ-CNN 加速框架的总体架构,然后分别从数据并行计算、数据通信和数据存储 3 个方面具体论述 FAQ-CNN 的设计思路和优化方法,并详细阐述 FAQ-CNN 的优势.

2.1 FAQ-CNN 架构

FAQ-CNN 加速框架的 FPGA 设计架构如图 1 所示,分别由量化组件、数据引擎、片上缓存单元、指令单元及计算引擎 5 个部分组成.

1) 量化组件.量化方法往往具有自身独特的数据格式和运算规则.FAQ-CNN 通过量化组件来形式化地描述量化方法的硬件代码实现规则,将量化方法的硬件实现过程分解为运算规则和量化数值映射,以灵活支持各类量化方法的快速集成.具体来讲,量化组件中的模块 op 负责量化算法所涉及的乘加(multiply accumulate, MAC)操作;模块 quantization 负责量化算法的数值映射.这 2 个模块的具体实现可根据量化方法的特点来定制,3.1 节中将详细介绍这 2 个模块的模板化实现方法.

2) 数据引擎.低位宽数据不经重组而直接传输,往往对带宽利用不够充分且效率低下,甚至会导致通信瓶颈,而将低位宽数据组织成高位宽的数据来提升带宽利用率又会增加设计复杂性.FAQ-CNN 设计了支持并行读写的数据引擎,包含编码器和解码器 2 个模块,用来实现单时钟周期内多个数据的并行读写,缓解数据传输与数据计算间速率不匹配的矛盾.

3) 片上缓存.FAQ-CNN 中的片上缓存资源用于存储输入特征图、输出特征图及模型权重.如图 1 所示,数据引擎中的 Load 操作将输入特征图与模型权重加载到片上缓存,并通过 Store 操作将输出特征图写入片外存储器.

4) 指令单元.负责按照预先定义的指令规则解析模型配置参数,这些配置参数规定了数据引擎与

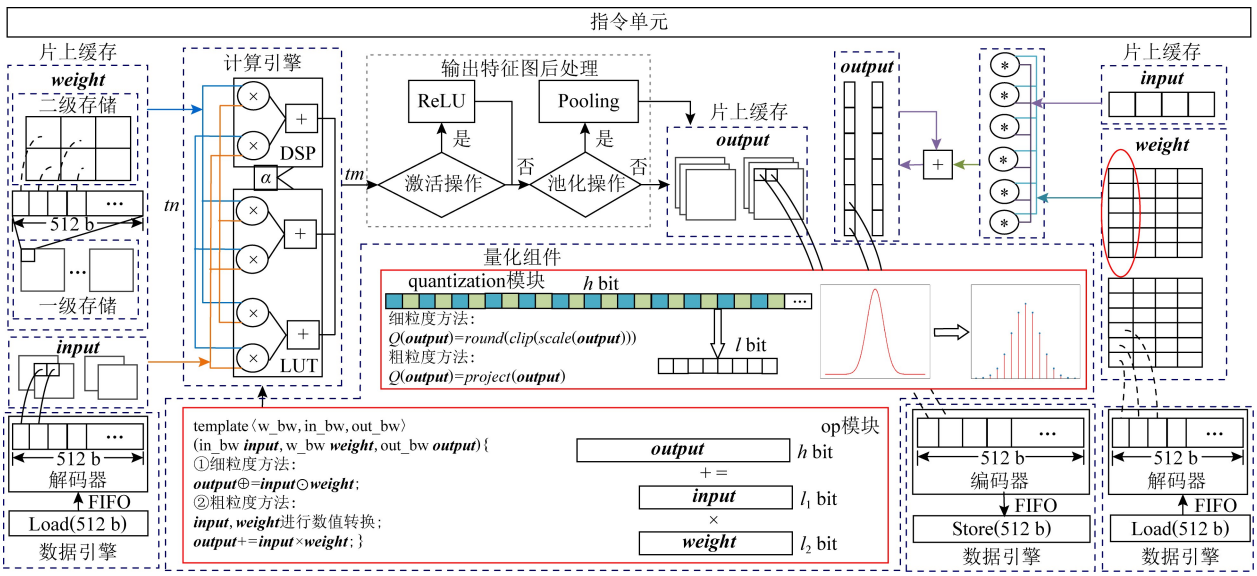
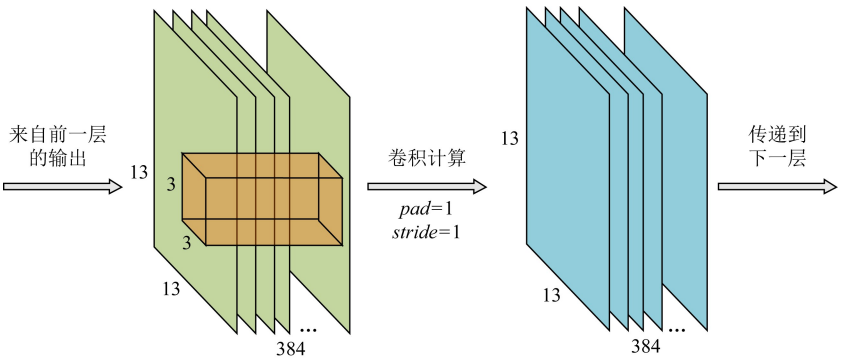


Fig. 1 FAQ-CNN framework components and implementation mechanism

图 1 FAQ-CNN 框架组成和实现机制

计算单元的工作方式.在 FAQ-CNN 中,指令由 8 个元素构成:输入通道、输出通道、输入特征图高度、输入特征图宽度、卷积核大小、卷积步长、卷积填充和计算类型.其中,计算类型用来明确是卷积层的计算还是全连接层的计算以及是否还包含激活层和池化层计算.如图 2 所示,以 AlexNet[42] 模型的第 4 个卷积层为例,按照 FAQ-CNN 所定义的指令格式,则其相应的指令配置参数实例为(384,384,13,13,3,1,1,1).



(a) 2维卷积计算

	输入通道数	输出通道数	输入行	输入列	卷积核尺寸	步长	填充大小	计算类型
指令格式	384	384	13	13	3	1	1	1

(b) 卷积参数配置实例

Fig. 2 Definition of instruction format

图 2 指令格式定义

5) 计算引擎.当加载满足计算条件的输入数据之后,FAQ-CNN 的计算引擎按指令配置参数来对这些数据进行计算并输出相应的结果数据.由于 CNN 模型各层的计算任务类型不同,比如卷积层是计算密集型的,而全连接层却是通信密集型的,因此 FAQ-CNN 在计算引擎中采用 2 个计算内核来分别处理这 2 类计算.

此外,由于 FPGA 片上资源受限,所以在 CNN 模型部署时无法将其某一层的全部张量数据都直接加载进来,只能利用数据切片来迭代地完成整个层的计算.对于每次的迭代处理,其过程可分为 4 个阶段,即加载、计算、后处理、存储.其中,后处理主要包括非线性激活操作和池化 2 类操作.此外,FAQ-CNN 为片上缓存相应地设计了双缓冲区.双缓冲区的设计能够有效支持 FAQ-CNN 利用流水线技术对该 4 个阶段数据流进行并行处理.

2.2 数据并行计算

当张量数据加载到片上存储器后,FAQ-CNN 中的卷积层和全连接层进行相应计算,其流程代码如图 3 所示.结合硬件设计特性,FAQ-CNN 采用多种机制加速计算.

1) 循环展开.FAQ-CNN 卷积层的嵌套循环结构如图 3(a)的代码所示.由于张量数据在不同通道

```
for(int kh=0;kh<kernel;kh++){
for(int kw=0;kw<kernel;kw++){
for(int tcc=0;tcc<tc;tcc++){
for(int trr=0;trr<tr;trr++){
#pragma HLS PIPELINE
for(int too=0;too<tm;too++){
for(int tii=0;tii<tn;tii++){
  op(&output_buff[too][trr][tcc],
  input_buff[tii][trr+kh][tcc+kw],
  weight_buff[too][tii][kh][kw]);
}}}}}}
```

(a) 卷积层

```
for(int j=0;j<tm;j++){
#pragma HLS PIPELINE
for(int i=0;i<tm;i++){
    op(&output_buff[i],
    input_buff[j],
    weight_buff[i][j]);
}}
```

(b) 全连接层

Fig. 3 Computing engine of convolutional layer and full connected layer

图 3 卷积层和全连接层的计算引擎

上的计算操作是彼此独立的,因此可将与输入和输出数据通道相关的 2 个循环放置到嵌套循环的最内层并进行循环展开.pragma 指令规定了编译器以复制硬件资源的方式来展开这 2 个内部循环并进行流水线处理.

2) 数据切片.在图 3(a)中,代码的 4 个外层循环用来对单个通道的卷积核和特征图进行处理,tm 和 tn 参数分别代表输出特征图和输入特征图在通道维度上的切片因子.因此,卷积计算的并行度可由 tm 与 tn 的乘积获得.此外,为符合并行计算的要求,权重数据需沿着输入通道和输出通道进行数组分割,输入特征图和输出特征图同样需要在通道维度上进行数组分割.全连接层的循环结构代码如图 3(b)所示.与卷积层类似,全连接层的输入和输出均是 1 维张量,图 3(b)中的 tm 和 tn 分别表示输出和输入张量模长的切片因子.

3) 运算规则.图 3 中 op 模块定义运算规则,可以代替 MAC 操作并依据特定量化方法的数据格式进行定制.FAQ-CNN 中提供了 2 种不同粒度的 op 运算,有关 op 运算操作更多的实现细节将在 3.1 节进行详细讨论.

4) 算子融合.FAQ-CNN 将激活与池化操作直接融合到卷积层或全连接层的后处理阶段.该方式能够充分利用 FPGA 片上资源,减少片上存储器和片外存储器的数据传输次数,从而快速处理此类计算任务.此外,在 FAQ-CNN 流水线处理流程中,计算阶段和数据加载阶段是主要的耗时阶段,而后处理阶段并不耗时,因此将激活层和池化层的计算任务融合到卷积层或者全连接层的后处理阶段,不会影响 FAQ-CNN 流水线处理性能.

2.3 通信带宽优化

CNN 参数量庞大,但 FPGA 片上存储资源有限,难以容纳如此众多的参数,因此张量数据须放置于片外存储器以备访问.FAQ-CNN 为了充分利用数据传输带宽资源,采用将多个数据项打包合并成 1 个宽字(例如 512 b 的字)的方式,在 1 个时钟周期内批量地传输数据,以此来提高数据传输效率.FAQ-CNN 依据 CNN 模型数据特征设计了 2 种便于高效读写的数据编码规则和相应的并行解码方法,并充分利用猝发传输的优势来进一步提高宽字传输效率.

1) 编码规则.FAQ-CNN 采用 2 种编码规则来处理不同场景的张量数据,分别是分级编码和位宽无关编码.考虑到实际应用场景的位宽限制,分级编码采用 3 种不同的位宽来编码数据,如图 4 所示,分别是低位宽 l bit,次高位宽 q bit 和高位宽 h bit.低位宽的 l bit 数据是真正需要访问的数据,高位宽的 h bit 数据是所要传输的对象数据.多个 l bit 的数据首先被编码为 1 个 q bit 的数据,然后多个 q bit 的数据再被编码为 h bit 的数据.在解码时,可根据分级规则来读取实际数据,即按 q bit 直接读取(一般为 32 b),然后再逐个读取 l bit 的数据.位宽无关编码则是将多个 l bit 的数据直接编码为 1 个 h bit 的数据,在解码时可直接读取每一个 l bit 的数据.2 种编码规则在遇到多个数据项的比特位之和不足以填充目标位宽时,将采用无效值 0 作为填充项.

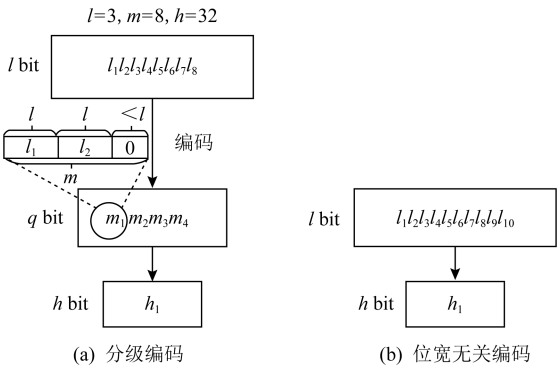


Fig. 4 Encoding scheme

图 4 编码规则示意

FAQ-CNN 采用位宽无关编码方式处理权重数据,而对输入特征图和输出特征图数据则采用分级编码方式.这是因为,对于权值数据来讲,在模型推理过程中只需加载一次并保持不变,采用位宽无关编码方式可以最大限度地利用带宽资源,避免填充大量无效值 0.对于输入特征图和输出特征图数据来讲,相比于权重数据,特征图数据的数据项较少且维度不同,数据在线重组织会引入较高的时间消耗,而分级编码方式适用于来自不同维度的少量数据项,且能有效减少无效数据,支持特征图数据的实时处理.以输出特征图为例,图 5 展示了具体的分级编码流程,输入特征图亦可按类似方式进行处理.

2) 并行解码.片上与片外的数据交换性能除受带宽利用率的影响外,还取决于数据的解码速率.FAQ-CNN 中的编码数据对象是多维张量,对这些多维张量数据在同周期内并行解码将极大提升数据交换性能.为避免并行解码引起片上存储资源访问冲突,FAQ-CNN 将宽字数据的各个数据项存放在 FPGA 上的不同存储区域(bank)来解决此问题.FAQ-CNN 解码时利用资源复用机制同时支持解码和计算引擎对存储资源的访问,从而降低总体资源消耗量.

具体来说,对于模型权重数据,FAQ-CNN 解码时沿着数据的输入通道和输出通道的维度进行访问,

并采用与计算引擎相同的数组分割规则以便并行执行,具体过程如图 6 所示.尽管对输入特征图和输出特征图亦可沿着通道维度并行解码,但是,考虑到输入特征图和输出特征图的数据拷贝方式是每次只传输通道维度的部分数据,如果沿着通道维度进行编解码,在传输过程中将会出现大量的碎片化数据,导致传输性能大幅降低.虽然数据重组织能够缓解数据碎片化的问题,但会引起额外的时间消耗.因此,对于输入特征图和输出特征图数据,FAQ-CNN 解码时沿着特征图数据的列维度处理,既能连续读取数据,又能有效支撑并行执行.在实际应用中,为减少列维度数组分割因子的大小,采用适当的编码位宽来满足数据高效传输的需求又不消耗过多的片上存储资源.

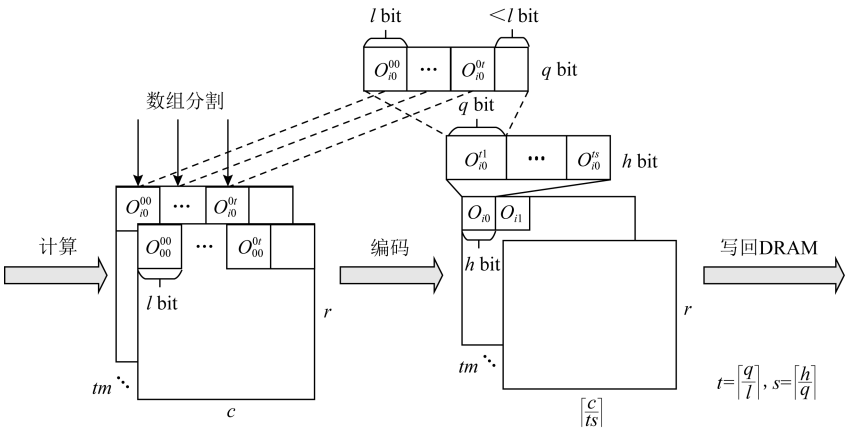


Fig. 5 Encoding detail of output feature map

图 5 输出特征图编码细节

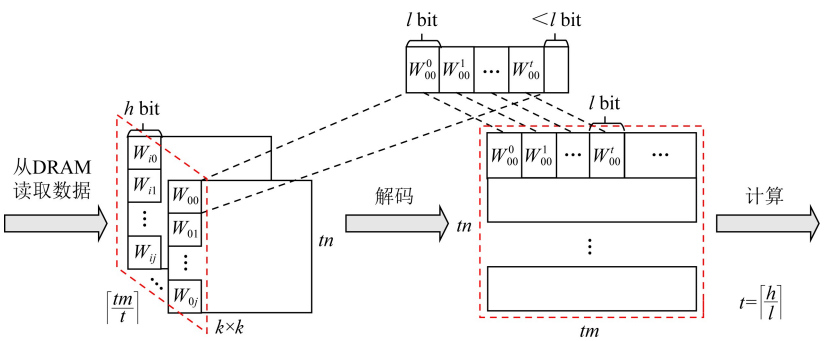


Fig. 6 Decoding detail of model weights

图 6 模型权值解码细节

3) 猝发式传输.利用猝发式传输的方式可提高数据传输性能,尤其对于宽字传输更为有效.猝发式传输高效与否取决于猝发传输长度的具体设置,而猝发传输长度的取值受片外存储和访问方式的制约,2.4 节中将讨论提升猝发传输长度的存储策略.本文前期初步探究了不同传输位宽和猝发传输长度在 200 MHz 频率下的实时带宽情况,如图 7 所示.可以看到,带宽峰值会随着位宽的增加和猝发传输长度的增加而提升.当位宽增加时猝发传输长度对实时带宽的影响更加明显.因此,FAQ-CNN 通过提升猝发传输长度来利用猝发传输的优势进一步提高宽字传输效率.

4) 传输频率.数据中心的 FPGA 平台能够支持最大 512 b 的数据传输位宽,Zynq 系列嵌入式 FPGA 的数据传输接口支持位宽有限,通常不超过 256 b 或 128 b[43].因此,为提高数据接口读写效率,FAQ-CNN 利用异步时钟传输方式,并在 AXI 传输总线的路径上添加时钟转换器和位宽转换器来弥合通信接口速率差异,从而提高 FPGA 片外存储器的数据读写速率.例如,片外存储器的工作频率设定为 FPGA 片上存储器 I/O 端口频率的 2 倍甚至更高,以支持快速读写.

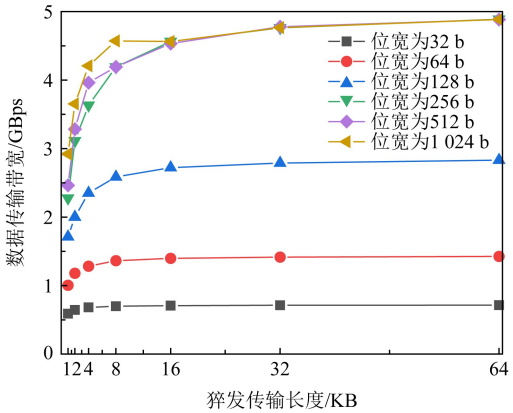


Fig. 7 Bandwidth of different bit width and burst lengths

图 7 不同位宽和猝发传输长度下的带宽

2.4 数据存储优化

数据存储方式是影响加速器性能的另一重要因素.针对片上存储,FAQ-CNN 避免硬件资源访问端口数目限制导致的数据访问冲突,设计满足单周期多数据访问需求的存储方案.对于片外存储,数据重组织机制用于保证数据分布连续性,提升数据通信效率.FAQ-CNN 针对片上存储和片外存储 2 个方面进行联合优化.

1) 片上存储.在 2.2 节中,计算引擎沿数据的通道维度并行访问张量数据.因此,为避免访问冲突,不同通道的数据必须位于不同的存储资源块中.具体优化时则利用 pragma 指令将张量数据沿通道维度进行数组分割.需要注意的是,对卷积层的权重、输入通道和输出通道同时进行数组分割将造成片上存储压力.为减少存储资源消耗,FAQ-CNN 设计了 2 级缓存机制,将从片外存储器拷贝的大块数据用高位宽模式缓冲于片上存储中.2 级存储主要面向计算引擎,依靠 LUT 资源实现.1 级存储既可以利用 LUT 资源也可以利用 BRAM,实际使用时需要根据片上资源消耗的情况来进行配置,同时还需决策是否对权重进行双缓冲.在 3.3 节中,本文将通过设计空间探索来具体阐述资源配置优化的实现细节.

2) 片外存储.由于切片方式的设计,FAQ-CNN 对多维张量的数据访问具有局部性.如图 8 所示,FAQ-CNN 每次所访问的原始数据在片外存储器中是非连续存放的.图 8 中使用地址偏移量表示的数据是全连接层流水线处理过程中的一次迭代中需要传输的数据,可以明显地看到数据是碎片形式分布的.和图 8 全连接层的权值相比,卷积层的权值仅多了卷积核行和列 2 个维度,权值访问同样存在数据碎片化的问题.在 2.3 节中提到,猝发传输长度直接影响实时的数据传输带宽,而猝发传输长度取值受片外数据存储方式和访问方式的制约,碎片化数据访问模式将严重降低数据传输性能.为了最大化数据传输硬件资源的性能,FAQ-CNN 利用重组织操作将片外存储碎片化的数据进行整理,形成连续的数据分布.图 8 显示了全连接层权值数据的重组织过程,FAQ-CNN 根据数据访问方式,将每次需要传输的数据按照地址连续的方式存放于片外存储器中.片外数据经过重新组织后,可有效利用猝发传输长度机制来提高实时带宽.由于 CNN 模型在处理图像分类任务时,其权重数据的访问保持恒定顺序,因此数据重组织的处理只需要离线执行 1 次.一旦经过重组织,权值数据加载到片外存储器后每次需要访问的数据是地址连续的.模型中卷积层权重数据的具体处理过程如算法 1 所示:

算法 1. 卷积层权重数据重组织算法(CWRA).

输入:原始权重张量 $\boldsymbol{W}$、权值位宽 bw_l、编码后的位宽 bw_h、输出通道数 m、输入通道 n、输出通道切片因子 tm、输入通道切片因子 tn;

输出:重组织后的权重张量 $\boldsymbol{W}'$.

① $weight\_num \leftarrow bw_h/bw_l$;

② $word\_num \leftarrow \lceil tm/weight\_num \rceil$;

③ for all $bo < \lceil m/tm \rceil$ do

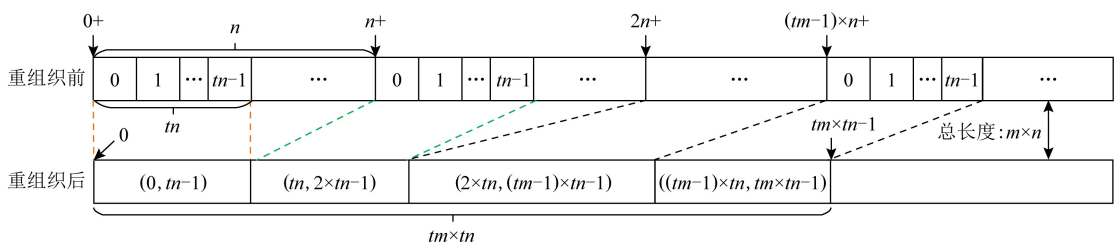


Fig. 8 Weight reorganization of fully connected layer

图 8 全连接层权值重组织过程

```
④  for all bi < ⌈n/tn⌉ do
⑤    for all i<k×k do /* 卷积核大小 */
⑥      for all ti<tn do
⑦        for all to<word_num do
⑧          temp←0;
⑨          for all s<weight_num do
⑩            使用 bo,bi,i,ti,s 计算源 index;
⑪            temp|=W[index]≪(s×bw_l);
⑫          end for
⑬          使用 bo,bi,i,ti 计算目的 index;
⑭          W'[index]←temp;
⑮        end for
⑯      end for
⑰    end for
⑱  end for
⑲ end for
```

3 FAQ-CNN 框架实现

本节分别从加速器的量化方法适配、片上资源建模和设计空间探索 3 个方面来阐明 FAQ-CNN 框架实现过程.为了适配多种量化方法,FAQ-CNN 框架通过模板将量化方法自身的具体操作抽象成量化组件.为优化 FPGA 片上资源的配置,FAQ-CNN 对资源消耗情况进行建模,采用启发式规则来降低搜索规模,并利用 CPU-GPU 异构平台快速寻找最佳配置.

3.1 量化方法适配

为了适配各种量化方法,FAQ-CNN 中的量化组件需要与计算引擎、数据引擎等模块解耦.FAQ-CNN 通过模板化的方式将量化组件分解为数值映射与运算规则,量化算法能够灵活地适配到 FAQ-CNN 框架中,并定制 CNN 加速器.

FAQ-CNN 主要考虑量化后数值的数据位宽和数据格式,因为数据位宽会影响数据拷贝的方式进而影响传输效率,尤其处理宽字数据时更为敏感,借助 2.3 节中的编解码方案可以屏蔽底层数据传输时的位宽差异.每种量化方法都有其独特的数据格式,而数据格式会影响到数值映射方式和相应的运算规则.例如,BNN 中的 1 b 用于表示数值 1 和 −1 而不是 0 和 1,所以运算时需要用 −1 来替代 0[1].与之不同的是,PoT 量化方法的数值位表示 2 的幂,且最高位用作符号位,所以运算时需要进行移位等操作[22].为应对数据位宽和数据格式的变化,FAQ-CNN 分别实现了粗粒度和细粒度的量化方法适配方案.

FAQ-CNN 中的粗粒度适配方案能够快速且简便地定义数值映射和运算规则,即将量化后的目标数值转化为可直接参与 MAC 操作的补码数值,然后在运算时对补码数值直接进行 MAC 操作.对于数值映射,线性量化过程通常包含缩放(scale)、截断(clip)和取整(round)三个步骤.通过这 3 个步骤,FAQ-CNN 可得到补码数值,然后利用逆变换规则,将补码数值转换为目标数值则可得到最终结果.

然而,粗粒度的解决方案无法应对复杂的量化方法,例如基于 K-means 的量化[44].这类量化方法通常需要提供完整的量化规则才能完成相应的数值映射.因此,为支持细粒度量化方法适配方案,FAQ-CNN 将数值映射和运算规则封装成抽象接口函数来支持具体的量化方法实现.对于数值映射,FAQ-CNN 提供全精度输入和量化后的低精度输出,接口函数负责完成全精度数值向低精度输出的投影(project);对于运算规则,FAQ-CNN 提供 2 个低精度的量化数值和 1 个全精度的输出数值,该接口函数负责完成低精度数值的乘法操作并把乘法结果累加到单精度 32 b 浮点数表示的输出数值上.

3.2 片上资源模型构建

嵌入式 FPGA 上的片上资源有限,因此实现方案要充分利用这些资源才能达到最佳性能.在具体方案实现过程中,为了提高资源利用率和模型推理性能,FAQ-CNN 通过对资源消耗分析并构建资源模型来寻求最优的资源配置.FPGA 片上的主要资源为 DSP,BRAM 和 LUT,因此 FAQ-CNN 将它们作为资源模型的主要分析对象.此外,还需要结合 FAQ-CNN 中卷积层与全连接层的未确定的切片因子,才能构建 FAQ-CNN 的资源消耗模型.为方便理解,定义切片因子四元组

$$t_{\text{FAQ-CNN}}=\langle conv\_tm, conv\_tn, fc\_tm, fc\_tn\rangle, \quad (1)$$

$conv\_tm$,$conv\_tn$,fc_tm,fc_tn 分别代表卷积层的输出通道和输入通道的切片因子以及全连接层输出向量和输入向量的切片因子.实际上,全连接层的操作可以复用卷积层的计算引擎来实现,即可将全连接层的操作转换为卷积操作,以减少 FAQ-CNN 中的资源占用.因此,全连接层的切片因子 fc_tm 和 fc_tn 可配置为 0.对于以卷积层为主的 CNN 模型,这种复用机制非常有效.下面着重分析 DSP,

BRAM,LUT 的资源使用场景并构建资源消耗模型.对应 FPGA 模块调用的设计模式,FAQ-CNN 采用树状自顶向下的方式分析总资源消耗.

1) DSP 模型.DSP 资源的使用分为 2 部分:①与卷积层和全连接层的乘法有关,该部分 DSP 消耗量可由切片因子四元组 $t_{\text{FAQ-CNN}}$ 直接表示;②用于数组索引等计算,与切片因子四元组无关.因此,DSP 的资源模型可表示为

$$R_{\text{total}}^{\text{DSP}}=R_{\text{init}}^{\text{DSP}}+R_{\text{mac}}^{\text{DSP}}\times(conv\_tm\times conv\_tn+fc\_tm),\tag{2}$$

其中,$R_{\text{init}}^{\text{DSP}}$ 是与切片四元组不相关的资源消耗,可以通过高层次综合工具获取具体数值.$R_{\text{mac}}^{\text{DSP}}$ 则表示 1 次乘法运算的 DSP 消耗量,受操作数位宽的影响.当执行乘法运算时,如只使用 LUT 资源(例如 BNN 量化方法),$R_{\text{mac}}^{\text{DSP}}$ 可以为 0.

2) BRAM 模型.在 FAQ-CNN 实现时,BRAM 用于存储大块数组数据.BRAM 有 3 种不同的配置模式,分别是:①单端口(SP);②真双端口(TDP);③简单双端口(SDP)[45].3 种模式所支持的读写方式和存储形式不同,因此 BRAM 的使用量与配置模式密切相关.此外,数组分割因子也会影响所需的 bank 数量,而 bank 数量可直接反映消耗多少 BRAM.BRAM 的总消耗量可定义为

$$R_{\text{total}}^{\text{BRAM}}=R_{\text{init}}^{\text{BRAM}}+R_{\text{conv}}^{\text{BRAM}}+R_{\text{fc}}^{\text{BRAM}},\tag{3}$$

其中,$R_{\text{init}}^{\text{BRAM}}$ 表示与切片四元组不相关的 BRAM 消耗,主要用于 FPGA 数据传输接口.$R_{\text{conv}}^{\text{BRAM}}$ 和 $R_{\text{fc}}^{\text{BRAM}}$ 分别表示卷积层的 BRAM 消耗量和全连接层的 BRAM 消耗量.以卷积层为例,输入特征图张量和输出特征图张量利用双缓冲机制,而模型权重张量可有多种存储模式,既可以使用 LUT 资源存储,也可以使用 BRAM 资源存储.因此,$R_{\text{conv}}^{\text{BRAM}}$ 可定义为

$$R_{\text{conv}}^{\text{BRAM}}=R_{\boldsymbol{input}}^{\text{BRAM}}\times 2+R_{\boldsymbol{output}}^{\text{BRAM}}\times 2+R_{\boldsymbol{weight}}^{\text{BRAM}}\times\alpha,\tag{4}$$

α 的值和模型权重的存放模式相关,可能取 0,1,2,表示权重是使用 LUT 形成的存储资源存放还是 BRAM 资源存放,以及是否采用双缓冲模式.$R_{\boldsymbol{input}}^{\text{BRAM}}$,$R_{\boldsymbol{output}}^{\text{BRAM}}$ 和 $R_{\boldsymbol{weight}}^{\text{BRAM}}$ 能够参考文献[45]进行计算.即

$$R_{\text{buffer}}^{\text{BRAM}}=\left\lceil\frac{data\_bits}{width}\right\rceil\times\left\lceil\frac{buff\_size}{pf\times depth}\right\rceil\times pf,\tag{5}$$

其中,$data\_bits$ 是张量元素的位宽,$buff\_size$ 表示输入张量、输出张量或者权重张量的大小,而 $width$ 和 $depth$ 取决于 BRAM 各种配置模式中块 BRAM 的宽度和深度,pf 表示数组分割因子.由于输入特征图数据和模型权重只进行读操作,所以使用单端口的存储模式即可.输出特征图的数据包含累加操作,所以需要真双端口的模式支持单周期的读写.以此类推,也可为全连接层构造 BRAM 资源消耗模型.全连接层是通信密集的,FAQ-CNN 需要传输大量的权值数据,因此权值使用 BRAM 资源存储并进行双缓存,输入向量和输出向量在寄存器资源中缓存确保并行访问.

3) LUT 模型.FAQ-CNN 在循环控制、循环内部模块设计、BRAM、寄存器和独立的表达式逻辑中都使用到 LUT 资源.事实上,几乎所有的模块设计都会用到 FPGA 中的 LUT 资源.因此,准确构建 LUT 资源模型十分重要.FAQ-CNN 通过理论分析和 Vivado[43] 综合工具来联合构建 LUT 资源模型.首先,借助文献[14,17]的分析模式给出 LUT 资源消耗的参数化表示方式;然后,利用 Vivado 工具获得表示方式中的未知参数,并将这些参数作为常数来形成最终的资源模型.LUT 资源的总消耗量可被定义为

$$R_{\text{total}}^{\text{LUT}}=R_{\text{init}}^{\text{LUT}}+R_{\text{conv}}^{\text{LUT}}+R_{\text{fc}}^{\text{LUT}},\tag{6}$$

其中,$R_{\text{conv}}^{\text{LUT}}$ 是卷积层消耗的 LUT 资源.$R_{\text{conv}}^{\text{LUT}}$ 由 2 部分组成:一部分用于逻辑实现,另一部分用于存储.因此

$$R_{\text{conv}}^{\text{LUT}}=R_{\text{logic}}^{\text{LUT}}+R_{\text{memory}}^{\text{LUT}},\tag{7}$$

$R_{\text{logic}}^{\text{LUT}}$ 用于所有循环控制、循环内部模块、BRAM 多路复用器和独立的表达逻辑,定义为

$$R_{\text{logic}}^{\text{LUT}}=R_{\text{loop}}^{\text{LUT}}+R_{\text{mux}}^{\text{LUT}}+R_{\text{expr}}^{\text{LUT}},\tag{8}$$

其中,$R_{\text{loop}}^{\text{LUT}}$ 描述了迭代循环的 LUT 消耗,它与循环展开因子 lf 相关,lf 和切片四元组直接相关,因此有

$$R_{\text{loop}}^{\text{LUT}}=R_{\text{loop\_ctrl}}^{\text{LUT}}+R_{\text{loop\_module}}^{\text{LUT}}\times lf,\tag{9}$$

其中,$R_{\text{loop\_module}}^{\text{LUT}}$ 是循环内部的模块设计,可以是简单的表达式模块,也可以是复杂的模块设计,复杂的模块设计可以使用递归的方式进行分析.式(8)中的 $R_{\text{mux}}^{\text{LUT}}$ 是连接 BRAM 存储器的数据多路复用器所消耗的 LUT 资源,它可被表示为

$$R_{\text{mux}}^{\text{LUT}}=\sum_{b\in conv.buff}R_{\text{mp}}^{\text{LUT}}\times pf,\tag{10}$$

其中,$R_{\text{mp}}^{\text{LUT}}$ 表示连接单个 BRAM 存储器分区的多路复用器的 LUT 消耗,$conv.buff=\{\boldsymbol{input},\boldsymbol{weight},\boldsymbol{output}\}$,$pf$ 表示数组分割因子.式(7)中的 $R_{\text{memory}}^{\text{LUT}}$ 和使用 LUT 作为存储资源的张量数据相关,可以用相同的方式进行分析.

以此类推,对于全连接层来说,$R_{\text{fc}}^{\text{LUT}}$ 由切片四元组的 fc_tm 和 fc_tn 进行表示.采用式(2)~(10)

的方式,总资源消耗可由与设计相关的切片四元组 $t_{\text{FAQ-CNN}}$、权重的存储模式 α 和与硬件相关的固定常数($R_{\text{init}}^{\text{DSP}}$,$R_{\text{init}}^{\text{BRAM}}$,$R_{\text{init}}^{\text{LUT}}$,$R_{\text{loop\_ctrl}}^{\text{LUT}}$,$R_{\text{expr}}^{\text{LUT}}$,$R_{\text{mp}}^{\text{LUT}}$)来表示,而这些数值可以通过 Vivado 综合工具获得.

3.3 设计空间探索

在构建资源模型后,FAQ-CNN 通过探索设计空间确定卷积层等资源最佳配置方式,实现 CNN 推理延迟最小.根据 2.1 节中的数据流调度规则,FAQ-CNN 利用流水线模型来表示总延迟.CNN 模型中某 i 层计算延迟和 CNN 模型中所有层的处理延迟总和被定义为

$$layer_i=\max(C_{\text{load}},C_{\text{compute}},C_{\text{post\_process}},C_{\text{store}})\times\beta,$$
$$latency_{\text{total}}=\sum_i layer_i, \tag{11}$$

其中,β 是流水线的迭代次数.流水线迭代计算 2 批不同输出数据的间隔时间,取决于流水线 4 个阶段中时间消耗最多的环节.依据 3.2 节中已构建的资源模型,FAQ-CNN 将资源配置设计空间搜索问题转换为整数非线性规划(integer nonlinear programming, INLP)问题,即:

目标函数 $\min: latency_{\text{total}}$,

约束条件

$$R_{\text{total}}^{\text{DSP}}\leqslant A_{\text{total}}^{\text{DSP}},$$
$$R_{\text{total}}^{\text{BRAM}}\leqslant A_{\text{total}}^{\text{BRAM}},$$
$$R_{\text{total}}^{\text{LUT}}\leqslant A_{\text{total}}^{\text{LUT}}, \tag{12}$$

其中,CNN 模型中所有层的处理延迟总和 $latency_{\text{total}}$ 可以使用切片四元组 $t_{\text{FAQ-CNN}}$ 和 CNN 模型参数直接表示.$A_{\text{total}}^{\text{DSP}}$,$A_{\text{total}}^{\text{BRAM}}$,$A_{\text{total}}^{\text{LUT}}$ 表示可获取的总资源量,$R_{\text{total}}^{\text{DSP}}$,$R_{\text{total}}^{\text{BRAM}}$,$R_{\text{total}}^{\text{LUT}}$ 可以使用切片四元组和权重存储模式 α 表示.以 AlexNet[42] 为例,参数四元组会受 CNN 模型参数的约束,如输入特征图通道数和输出特征图通道数的约束.AlexNet[42] 模型的搜索空间数量级为 10^{13},在如此巨大的可行解空间中搜索最优解需要消耗大量时间,而且神经网络模型规模越大,搜索需要的时间越长.为能够在有限时间内寻找到最优解,FAQ-CNN 中采用基于启发式规则的剪枝策略来缩小搜索空间.以卷积层为例,FPGA 片上各种资源消耗都与 $conv\_tm$ 和 $conv\_tn$ 的乘积有关,为 $conv\_tm$ 和 $conv\_tn$ 的乘积设置最大阈值 pf_max 以符合资源的粗粒度制约,考虑到卷积层的计算复杂性,用最小阈值 pf_min 来作为并行程度的下限,具体的约束条件则可表示为

$$pf\_min<conv\_tm\times conv\_tn<pf\_max. \tag{13}$$

对于全连接层,可在每次迭代期间计算出 fc_tm 个元素.为了计算输出向量的所有元素,需要将迭代次数向上取整.如果元素总个数不能被 fc_tm 整除,则向上取整操作必然会引入无效计算.因此,限制 fc_tm 来减少无效操作:

$$fc\_tm\in\left\{x\mid x=\min_{x_j}\left\{\left\lceil\frac{fc\_m}{x_j}\right\rceil=i\right\},\right.$$
$$\left.i=1,2,\cdots,fc\_m\right\}, \tag{14}$$

其中,fc_m 表示全连接层输出向量的最大长度.fc_tn 可以使用相同的方式约束.通过式(14)的约束条件,AlexNet 的搜索空间规模可由 10^{13} 降低到 10^9.此外,本文在 FAQ-CNN 实现时利用 CPU-GPU 异构平台来加快参数搜索过程.在 CPU 端计算可行解并借助约束条件完成设计空间剪枝,然后在 GPU 端来验证可行解并寻找最优解.借此,AlexNet[42],VGG16[46] 等典型模型只需花费数秒即可完成配置参数搜索.

4 实验评估

本节通过实验验证 FAQ-CNN 性能收益,并通过设计空间探索实验来论证 FAQ-CNN 的资源配置方法.首先,介绍实验所需软硬件配置,并选取典型量化方法作为 FAQ-CNN 加速器设计的实现对象.其次,通过 7 组不同传输位宽配置的实验来观测 FAQ-CNN 编解码方案对硬件加速器的数据传输效率的性能增益.再次,呈现各种量化方法在不同位宽下完成单个 MAC 操作所需的资源情况,并依据资源模型探究资源的整体消耗情况.最后,通过寻找片上资源最佳配置,并与其他最新方法进行性能对比,明确 FAQ-CNN 的性能优势.

4.1 实验设置

1) 软件环境.实验采用集成开发环境 Vitis 2020.1[47] 进行 FPGA 加速器设计.通过编程语言 C++设计各模块,并利用 Vitis HLS 工具来进行高层次综合.

2) 硬件环境.实验采用 Xilinx ZCU102 SoC FPGA 开发板作为硬件加速器验证平台,FPGA 的运行频率设定为 200 MHz.此外,配备工作站运行 Vitis 软件,搭载 Intel Xeon Silver 4210 CPU@2.20 GHz 和 64 GB DDR4 内存.配备 1 块 GeForce RTX2080Ti GPU 显卡,用于搜索参数配置.

3) CNN 模型与量化方法.选取 AlexNet 模型和当前业界经典的 CNN 量化方法,这些量化方法的激活值和权值的位宽配置如表 1 所示:

Table 1 Bit Width and LUT Consumption of Different Quantization Methods' MAC Operation

表 1 不同量化方法乘累加操作的位宽配置和 LUT 消耗

量化方法	位宽/b		LUT 消耗量	
	激活值	权值	细粒度	粗粒度
BNN[1]	4	1	6	7
TNN[2]	4	2	8	8
PoT[22]	4	4	22	36
DoReFa[21]	8	8	89	89
VecQ[11]	4	4	26	26
μL2Q[12]	8	4	44	44

4) 评估指标.FAQ-CNN 的实验评估选取 3 个定量指标来观测和分析 FAQ-CNN 的性能收益.

① 数据传输效率.为了验证 FAQ-CNN 的编解码方案能够有效提升数据传输效率,实验以量化方法 DoReFa 的加速器为例,来探究各种编解码位宽对加速器的数据传输速度的影响.

② 片上资源利用率.该指标能够反映 FAQ-CNN 所实现的低位宽加速器在完成单个 MAC 操作后的资源消耗情况及该加速器的整体资源消耗情况,资源利用率数据可由 Vivado 综合工具得到,能够准确反映 FAQ-CNN 是否高效利用 FPGA 片上资源.

③ 每秒运算次数.该指标能够反映 FAQ-CNN 所实现的低位宽加速器的峰值计算性能和 CNN 模型中各卷积层的实时计算性能.

4.2 编解码效率增益

依据 2.3 节给出的编解码规则,高位宽字的位个数会直接影响数据传输的效率.本节实验通过评测数据传输的执行时间,来探究编解码位宽对数据传输效率的影响.在 CNN 卷积层流水执行过程中,主要包含 3 个阶段,分别是 computing,loading input,loading weight,其中 loading input 和 loading weight 阶段负责加载输入特征图和权值数据到片上存储,占据数据传输的主要部分.首先,根据搜索工具获得的最佳配置来设定卷积层的输入通道和输出通道的切片因子,分别设置为 24 和 128.在此基础上,computing 阶段的时间会确定下来.然后,对不同位宽下的 loading input 和 loading weight 的时间进行评测.

以 DoReFa 量化算法为例,实验结果如表 2 所示,可以看到施加编解码之前,即数据的实际传输位宽和数据的位宽相同时,数据传输的时间远远高于计算所需时间,成为 CNN 计算耗时的主要原因.随着位宽的提升,数据传输耗时不断降低.从表 2 可知,当权重使用 256 b 的传输模式时,对于 AlexNet 的 5 层卷积层,数据传输的能力都可满足计算引擎的需求,此时编解码技术理论上带来的增益是未使用编解码时的 32(32=256/8)倍.对于输入特征图,当数据编码位宽设置为 32 b 即可满足数据传输需求,此时编解码技术提升 4 倍的传输速度.

Table 2 Clock Cycle of Computing, LI(loading input) and LW(loading weight) Phases

表 2 计算、LI(loading input)和 LW(loading weight)三个阶段的时钟周期数

模型层	计算阶段	位宽为 8 b		位宽为 16 b		位宽为 32 b		位宽为 64 b		位宽为 128 b		位宽为 256 b	
		LI	LW	LI	LW	LI	LW	LI	LW	LI	LW	LI	LW
Conv1	53 250	26 704	34 858	13 366	17 434	6 697	8 722	3 421	4 366	1 783	2 188	964	1 099
Conv2	4 735	8 476	76 851	4 252	38 410	2 140	19 210	1 084	9 610	566	4 810	292	2 410
Conv3	829	3 484	27 658	1 756	13 834	892	6 922	460	3 466	244	1 738	244	874
Conv4	829	3 484	27 658	1 756	13 834	892	6 922	460	3 466	244	1 738	244	874
Conv5	829	3 484	27 658	1 756	13 834	892	6 922	460	3 466	244	1 738	244	874

注:Conv 表示卷积层.

2.3 节中明确指出权重数据的编解码是沿着通道维度进行的.当编码位宽采用 256 b 时,FAQ-CNN 要求片上权值的存储资源在输入或者输出通道上的数组分割因子是 32.因为计算引擎的并行能力是 24×128,即要求权重数组在输入通道的数组分割因子是 24,在输出通道维度的数组分割因子是 128,所以通过复用的机制对权重数据编解码时沿着输出通道进行对数组的分割程度最小,即会使用最少的存储资源.对于输入特征图,数据张量的存储方式在列维度数组分割因子是 4,尽管无法满足计算引擎所要求的数组分割规则,即无法在通道维度上进行资源复用.但是,仍可通过使用较低的输入切片

因子降低对输入特征图的数组分割因子,从而降低存储资源的需求.在输入切片因子较低时,使用较高的输出切片因子保证较强的并行能力.

4.3 MAC 操作资源消耗

本节实验测量多种经典量化方法中单个 MAC 操作的 LUT 资源消耗量,用来指导 LUT 资源配置方式和构建资源模型.给定的操作数位宽下,分别使用细粒度和粗粒度 2 种方式完成各种量化方法的 MAC 操作.实验采用 Vivado 来获取准确的资源消耗量.

实验结果如表 1 所示,对于 BNN 和 PoT 算法,粗粒度方式比细粒度方式完成 MAC 操作需要更多的 LUT 资源,这是因为它们的数值不能直接参与 MAC 操作.粗粒度计算模式中需要中间转换步骤,例如 BNN 算法需要将 1 b 的 0 转化为 2 b 的−1,而细粒度计算模式中通过定制的计算方式完成 MAC 操作.BNN 只需要判断 1 b 的操作数为 0 或者 1,然后选通另一操作数原值或者取反后的值.

此外,可看出操作数位宽最低的 BNN 量化算法需要最少的 LUT 数量,在细粒度模式下只需要 6 个 LUTs 就可以完成 MAC 操作.操作数位宽相同的 PoT 量化算法和 VecQ 相比,PoT 算法的 MAC 操作需要更少的 LUT 资源.PoT 的数值表示的是 2 的幂次方,乘法操作可通过高效的移位操作来代替.对于激活值和权重配置为 8 b 的 DoReFa 量化方法,MAC 操作需要 89 个 LUTs 来完成.由于 LUT 资源总量有限,优先选择使用 DSP 资源来实现 8 b 的 MAC 操作.

4.4 卷积层整体开销与性能对比分析

本节实验部署多种量化算法到 FAQ-CNN 框架中,得到不同量化算法的加速器并统计相应的资源消耗量和计算性能.片上资源面向 AlexNet 模型进行配置且该模型用于评估各种量化算法的分类准确率.为了探究同时使用 DSP 和 LUT 这 2 种资源完成计算带来的收益,利用 3 组实验对比 8 b 的 DoReFa 量化方法加速器的资源消耗和性能,分别是只使用 DSP、只使用 LUT、联合使用 DSP 和 LUT 完成 MAC 操作.对于其他低位宽的量化算法,仅使用 LUT 资源完成 MAC 操作.最后,选择 FAQ-CNN 框架实现的 3 种加速器与 Caffeine 等量化加速器的性能进行对比.

表 3 中显示了各种量化算法加速器的资源利用率、峰值计算性能以及分类准确率.位宽配置最低的 BNN 加速器可以实现 3.28 TOPS 的峰值计算性能,其中 LUT 资源使用率最高达到了 66.40%,仅使用少量的 DSP 资源.但是,BNN 量化方法会严重降低分类精度导致无法保证应用需求.当部署 8 b 的 DoReFa 量化算法时,可获得 1.23 TOPS 的峰值计算性能,且 CNN 推理精度损失极小.此外,从 DoReFa 量化算法的 3 组实验可以看到,当只使用 DSP 或者 LUT 的时候都不能完全发挥片上资源的计算能力.当所有 MAC 操作使用 DSP 资源执行时,加速器设计仅使用了 39.98% 的 LUT 资源.同样,当所有 MAC 操作使用 LUT 资源执行时,加速器设计仅使用了 4.52% 的 DSP 资源.由此可见,这 2 种设计方式对片上逻辑资源使用都极不充分.在联合使用 DSP 和 LUT 完成 MAC 操作时,LUT 资源和 DSP 资源的使用率分别达到 82.36% 和 99.17%,该设计能够显著提升计算引擎的计算能力.

Table 3 Resource Consumption Utilization, Hardware Configuration and Computing Performance of Different Quantization Methods Accelerator

表 3 不同量化方法加速器的资源消耗率、硬件配置和计算性能

量化方法	利用率/%				*conv_tm*	*conv_tn*	α	计算性能/TOPS	分类准确率/%	
	LUT	FF	DSP	BRAM					Top-1	Top-5
BNN	66.40	17.13	4.84	65.34	128	64	0	3.28	38.38	63.53
TNN	66.86	17.80	4.80	43.88	128	48	0	2.46	52.65	76.41
PoT	78.91	34.06	7.70	75.28	128	32	1	1.64	58.81	81.22
VecQ	85.60	32.28	5.12	75.28	128	24	1	1.64	58.11	80.64
μL2Q	79.21	29.36	99.17	93.99	128	24	1	1.23	59.87	81.75
DoReFa_DSP+LUT	82.36	32.57	99.17	81.57	128	24	1	1.23	62.85	84.53
DoReFa_LUT	73.57	18.92	4.52	47.42	96	24	1	0.61	62.85	84.53
DoReFa_DSP	39.98	19.11	96.23	54.61	96	16	1	0.92	62.85	84.53

注:FPGA 总资源量中 LUT 为 274080,FF 为 548160,DSP 为 2520,BRAM 为 1780;*conv_tm* 表示卷积层输出通道并行因子;*conv_tn* 表示卷积层输入通道并行因子;α 表示权值存储模式(0 表示 LUT 单缓冲,1 表示 LUT 双缓冲).

表 4 展示了采用 DoReFa 量化算法的 FAQ-CNN 与相关量化加速器的性能对比,Caffeine[13] 和 AccELB[27] 加速器的时钟频率同样为 200 MHz. Caffeine[13] 采用高位宽 16 b 的数据配置,AccELB[27] 采用低位宽 8 b 和 2 b 的数据配置.在 2 b 数据配置下,FAQ-CNN 和 AccELB[27] 相比,计算引擎完全采用 LUT 资源完成计算,其他模块仅使用少量的 DSP 资源.在 16 b 数据配置下,FAQ-CNN 受限于存储资源限制实现和 Caffeine 接近的性能.在低位宽 8 b 数据配置下,FAQ-CNN 充分利用 DSP 资源和 LUT 逻辑资源实现 1 229 GOPS 的计算性能,和采用 16 b 的 Caffeine 相比,峰值性能提升至 3.6 倍.

Table 4 Computing Performance of FAQ-CNN and Other Quantization Accelerators

表 4 FAQ-CNN 与相关量化加速器计算性能对比

加速器	位宽/b	LUT	FF	BRAM	DSP	计算性能/GOPS
Caffeine[13]	16	1E5	8E4	782	1038	338
FAQ-CNN	16	1.5E5	8E4	1 740	1 272	461
FAQ-CNN	8	2.1E5	1.6E5	1 446	2 483	1 229
AccELB[27]	8	8.6E3	5E4	606	808	493
AccELB[27]	2	1E5	9.4E3	926	880	1 990
FAQ-CNN	2	1.8E5	9.8E4	781	121	2 458

由于参数的设置面向特定的 CNN 模型,虽然增大并行参数能够增强峰值计算性能,但是并不能增强实时处理性能.这是因为采用切片的方法,当通道切片因子无法整除总体通道数目时,必须向上取整计算完所有通道数据,不可避免地引发大量的无效计算.本节的实验是根据具体的 CNN 模型选取通道切片因子,在保证最佳的实时处理性能时选取最小化资源消耗量,所以片上的资源仍有余量.

4.5 资源配置优化与性能对比

本节对低位宽加速器的设计资源消耗量进行建模,并利用自动化搜索工具搜索片上资源最佳配置.采用的量化算法是在 AlexNet 模型上分类准确率最高的 INT8 数据类型的 DoReFa 量化算法.根据搜索到的参数对 FPGA 进行资源配置,实验结果与采用相同计算引擎的相关研究工作 Caffeine 进行对比.值得注意的是,Caffeine 仅实现了 16 b 的 AlexNet 网络模型.本节利用 FAQ-CNN 分别实现 16 b 和 8 b 的 AlexNet 模型与 Caffeine 进行细粒度性能对比.此外,将搜索算法得到的配置代入硬件设计中,分析 AlexNet 所有层的实时计算性能.

FAQ-CNN 依据搜索工具得到的参数值对 FPGA 片上资源进行配置,图 9 显示了卷积层和全连接层的资源配置.值得注意的是,卷积核较大的第 1 层卷积层单独实现.可以看到计算并行度较高的卷积层使用 60%左右的 LUT 资源和 95%的 DSP 资源,通信密集的全连接层仅使用了 10%左右的片上资源.其中,卷积层输出通道切片因子和输入通道切片因子分别是 128 和 24,模型权值使用 LUT 资源进行存储同时采用双缓冲机制.图 10 反映了 AlexNet 模型 5 层卷积层单层的实时处理性能,表 5 显示了卷积层的理论峰值性能和 5 层卷积层的平均计算性能.其中,FAQ-CNN 和 Caffeine 加速器设计的资源使用量显示在表 4 中.在 16 b 数据下,FAQ-CNN 受限于存储资源限制,实现和 Caffeine 加速器相近的处理性能.在 8 b 数据配置下,FAQ-CNN 得到卷积层 1.23 TOPS 的峰值计算性能和 490GOPS 的平均处理性能.与 Caffeine 相比,FAQ-CNN 使用了 2.3 倍的 DSP 计算资源和 1.67 倍的 LUT 资源.值得注意的是,8 b 数据的乘法和 16 b 数据的乘法都仅利用 1 个 DSP 完成计算.FAQ-CNN 的峰值处理性能是采用 16 b 数据配置 Caffeine 加速器的 3.6 倍,其中 DSP 资源提供了 2 倍左右的性能增益,LUT 资源提供额外的性能增益.因为第 1 层的输入通道较少,并不适合这种通道并行的设计,得到的实时处理性能只有 96 GOPS.除此之外,其他层的实际计算性能和峰值性能差距较小.对于卷积层,由于设计方案中采用流水线的机制,除了指令参数外,其他所有的数据都进行双缓存.通过这种缓存机制,在整个流水线计算过程中,计算引擎在 70%的时间内处于活跃状态.在计算方面,FAQ-CNN 联合

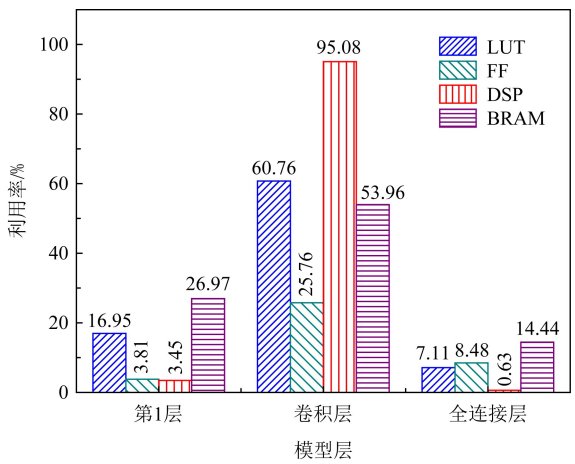


Fig. 9 Resource configuration of convolutional layer and full connected layer

图 9 卷积层和全连接层的资源配置

使用 LUT 资源和 DSP 资源完成 MAC 操作的方式可以最大化片上资源带来的计算性能,从而能够实现 24×128 的计算阵列.在 DSP 和 LUT 联合使用时,有多种配置方式,即按照输入通道进行配置和按照输出通道进行配置.由于 DSP 能够同时完成乘法操作和累加操作,所以在设计中采用输出通道配置的方式会更加有效.

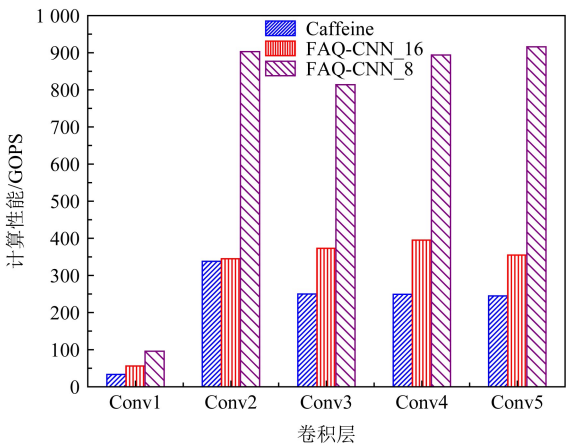


Fig. 10 Computing performance of AlexNet convolutional layers of FAQ-CNN and Caffeine

图 10 FAQ-CNN 和 Caffeine 处理 AlexNet 卷积层的计算性能

Table 5 Hardware Platform and Computing Performance of FAQ-CNN and Caffeine

表 5 FAQ-CNN 和 Caffeine 的硬件平台和计算性能

设计方案	硬件平台	峰值性能/GOPS	平均性能/GOPS
Caffeine	KU060	338	163
FAQ-CNN_8	ZCU102	1 229	490
FAQ-CNN_16	ZCU102	461	235

在数据通信方面,对于权重数据来说,通过重组织操作进行预处理,可确保每次所需传输的数据在外部存储器中是连续分布的.表 6 显示了数据重组织策略对卷积层和全连接层权值通信性能的影响.由于 AlexNet 第 3,4,5 层的权值数据传输方式相同,仅以第 3 层为例进行说明.

从表 6 中可以看到,重组织后的权值数据在进行数据传输时,猝发传输长度能够有效地提升到 27 KB 甚至更高,从而能够得到数据位宽为 256 b 模式下的最高传输带宽 4.77 GBps.对于输入特征图,尽管未进行重组织,但是采用的是 32 b 的数据位宽,依然能够得到该传输模式下的最高带宽 715 MBps.表 6 的数据传输速度足够保证计算引擎对数据的需求.此外,表 7 显示了 AlexNet 中的 3 个全连接层的参数配置和相应的处理时间.可以看到,全连接层的实时平均带宽速度最高达到了 4.54 GBps,接近硬件资源能够提供的最大带宽速度 4.77 GBps.FAQ-CNN 充分利用带宽资源,能够实现全连接层的最低处理延迟.

Table 6 Impact of Data Reorganization on Communication Performance

表 6 数据重组织对通信性能的影响

模型层	猝发传输长度/KB		数据传输时间/μs	
	重组织前	重组织后	重组织前	重组织后
Conv1	34	34	6.92	6.92
Conv2	0.59	75	43.86	1.50
Conv3	0.20	27	33.41	5.58
FC	0.50	64	41.98	12.79

注:Conv 表示卷积层,FC 表示全连接层.

Table 7 Realtime Performance of Fully Connected Layers

表 7 全连接层的实时处理性能

模型层	参数量/MB	延迟/ms	带宽/GBps
FC1	36	7.74	4.54
FC2	16	3.56	4.38
FC3	3.9	0.89	4.27

注:FC 表示全连接层.

5 总 结

本文提出一种灵活的嵌入式 FPGA 加速器设计框架 FAQ-CNN,从量化方法到硬件加速器设计进行联合优化,支持嵌入式 FPGA 平台上快速且高效地部署量化 CNN 模型.FAQ-CNN 将量化方法分解为量化数值映射和数值运算规则,并通过模板来统一描述量化方法的实现过程;通过分级和位宽无关 2 种编码方案与并行解码机制,来提升片上与片外存储之间的数据交换效率;建立 FPGA 资源和性能的联合优化模型,针对带约束条件的设计空间,采用启发式策略进行裁剪,并利用异构平台搜索最优参数,实现 FPGA 片上资源的快速配置.实验结果表明,FAQ-CNN 能够高效适配多种量化方法,并能实现 8 b 下高达 1.23TOPS 的优越性能.FAQ-CNN 能够支持相关研究人员快速构建量化 CNN 加速器,对深度学习及异构计算等领域具有很好的指导意义和研究价值.

作者贡献声明:谢坤鹏提出算法思路,负责完成实验并撰写论文;卢冶提出系统思路、实验方案和指导意见,并修改与审核论文;靳宗明协助完成部分实验;刘义情参与论文校对和图表修正;龚成负责调研和数据分析;陈新伟参与论文校对和实验数据审查;李涛提出方法的指导意见.

参 考 文 献

[1] Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1 [J]. arXiv preprint, arXiv:1602.02830, 2016

[2] Li Fengfu, Zhang Bo, Liu Bin. Ternary weight networks [J]. arXiv preprint, arXiv:1605.04711, 2016

[3] Cong J, Liu Bin, Neuendorffer S, et al. High-level synthesis for FPGAs: From prototyping to deployment [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30(4): 473-491

[4] Li Guang, Wang Peisong, Liu Zejian, et al. Hardware acceleration of CNN with one-hot quantization of weights and activations [C] //Proc of Design, Automation & Test in Europe Conf & Exhibition(DATE). Piscataway, NJ: IEEE, 2020: 971-974

[5] Prost-Boucle A, Bourge A, Pétrot F, et al. Scalable high-performance architecture for convolutional ternary neural networks on FPGA [C/OL] //Proc of the 27th Int Conf on Field Programmable Logic and Applications(FPL). Piscataway, NJ: IEEE, 2017 [2020-10-12]. https://ieeexplore.ieee.org/abstract/document/8056850

[6] Mousouliotis P G, Petrou L P. CNN-Grinder: From algorithmic to high-level synthesis descriptions of CNNs for low-end-low-cost FPGA SoCs [J/OL]. Microprocessors and Microsystems, 2020, 73: 102990 [2021-01-05]. https://www.sciencedirect.com/science/article/pii/S0141933119300146

[7] Guyen D T, Nguyen T N, Kim H, et al. A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection [J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems, 2019, 27(8): 1861-1873

[8] Han Dong, Zhou Shengyuan, Zhi Tian, et al. A survey of artificial intelligence chip [J]. Journal of Computer Research and Development, 2019, 56(1): 7-22 (in Chinese)
(韩栋, 周圣元, 支天, 等. 智能芯片的评述和展望[J]. 计算机研究与发展, 2019, 56(1): 7-22)

[9] Jiao Li, Luo Cheng, Cao Wei, et al. Accelerating low bit-width convolutional neural networks with embedded FPGA [C/OL] //Proc of the 27th Int Conf on Field Programmable Logic and Applications(FPL). Piscataway, NJ: IEEE, 2017 [2020-11-01]. https://ieeexplore.ieee.org/abstract/document/8056820

[10] Wei Xuechao, Yu C H, Zhang Peng, et al. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs [C/OL] //Proc of the 54th Annual Design Automation Conf. New York: ACM, 2017 [2020-07-25]. https://dl.acm.org/doi/abs/10.1145/3061639.3062207

[11] Gong Cheng, Chen Yao, Lu Ye, et al. VecQ: Minimal loss DNN model compression with vectorized weight quantization [J/OL]. IEEE Transactions on Computers, 2020 [2020-09-04]. https://arxiv.org/pdf/2005.08501.pdf

[12] Gong Cheng, Li Tao, Lu Ye, et al. μL2Q: An ultra-low loss quantization method for DNN compression [C/OL] //Proc of Int Joint Conf on Neural Networks(IJCNN). Piscataway, NJ: IEEE, 2019 [2020-08-27]. https://ieeexplore.ieee.org/abstract/document/8851699

[13] Zhang Chen, Sun Guangyu, Fang Zhenman, et al. Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 38(11): 2072-2085

[14] Cong J, Wei Peng, Yu C H, et al. Automated accelerator generation and optimization with composable, parallel and pipeline architecture [C/OL] //Proc of the 55th ACM/ESDA/IEEE Design Automation Conf(DAC). Piscataway, NJ: IEEE, 2018 [2019-10-12]. https://ieeexplore.ieee.org/abstract/document/8465940

[15] Véstias M, Duarte R P, de Sousa J T, et al. A fast and scalable architecture to run convolutional neural networks in low density FPGAs [J/OL]. Microprocessors and Microsystems, 2020: 103136 [2020-12-04]. https://www.sciencedirect.com/science/article/pii/S0141933120303033

[16] Zhang Chen, Li Peng, Sun Guangyu, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks [C] //Proc of the 23rd ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2015: 161-170

[17] Cong J, Wei Peng, Yu C H, et al. Automated accelerator generation and optimization with composable, parallel and pipeline architecture [C/OL] //Proc of the 55th ACM/ESDA/IEEE Design Automation Conf(DAC). Piscataway, NJ: IEEE, 2018 [2020-04-26]. https://ieeexplore.ieee.org/abstract/document/8465940

[18] Nvidia. Nvidia CUDNN document [EB/OL]. [2020-12-12]. https://docs.nvidia.com/deeplearning/cudnn/index.html

[19] Nvidia. CUDA toolkit document [EB/OL]. [2020-12-12]. https://docs.nvidia.com/cuda/cublas/index.html

[20] Xilinx. Vitis AI User Guide [EB/OL]. [2020-12-12]. https://www.xilinx.com/support/documentation/sw_manuals/vitis_ai/1_3/ug1414-vitis-ai.pdf

[21] Zhou Shuchang, Wu Yuxin, Ni Zekun, et al. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients [J]. arXiv preprint, arXiv: 1606.06160, 2016

[22] Miyashita D, Lee E H, Murmann B. Convolutional neural networks using logarithmic data representation [J]. arXiv preprint, arXiv:1603.01025, 2016

[23] Nogami W, Ikegami T, O'uchi S, et al, Optimizing weight value quantization for CNN inference [C/OL] //Proc of Int Joint Conf on Neural Networks(IJCNN). Piscataway, NJ: IEEE, 2019 [2020-05-04]. https://ieeexplore.ieee.org/abstract/document/8852331

[24] Tambe T, Yang Enyu, Wan Zishen, et al. Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference [C/OL] //Proc of the 57th ACM/IEEE Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2020 [2020-05-22]. https://ieeexplore.ieee.org/abstract/document/9218516

[25] Zhao R, Song Weinan, Zhang Wentao, et al. Accelerating binarized convolutional neural networks with software-programmable FPGAs [C] //Proc of the 25th ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2017: 15-24

[26] Umuroglu Y, Fraser N J, Gambardella G, et al. FINN: A framework for fast, scalable binarized neural network inference [C] //Proc of the 25th ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2017: 65-74

[27] Wang Junsong, Lou Qiuwen, Zhang Xiaofan, et al. Design flow of accelerating hybrid extremely low bit-width neural network in embedded FPGA [C] //Proc of the 28th Int Conf on Field Programmable Logic and Applications (FPL). Piscataway, NJ: IEEE, 2018: 163-169

[28] Wang Kuan, Liu Zhijian, Lin Yujun, et al. HAQ: Hardware-aware automated quantization with mixed precision [C] // Proc of IEEE/CVF Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE 2019: 8612-8620

[29] Jing Lu, Liu Jun, Yu Fuhai. A deep learning inference accelerator based on model compression on FPGA [C/OL] // Proc of the 27th ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2019 [2020-03-12]. https://dl.acm.org/doi/abs/10.1145/3289602.3293938

[30] Li Yuhang, Dong Xin, Wang Wei. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks [J]. arXiv preprint, arXiv: 1909.13144, 2019

[31] Jain S, Venkataramani S, Srinivasan V, et al. BiScaled-DNN: Quantizing long-tailed datastructures with two scale factors for deep neural networks [C/OL] //Proc of the 56th ACM/IEEE Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2019 [2020-06-15]. https://ieeexplore.ieee.org/abstract/document/8806964

[32] Cong J, Xiao Bingjun. Minimizing computation in convolutional neural networks [C] //Proc of the 24th Int Conf Artificial Neural Networks. Berlin: Springer, 2014: 281-290

[33] Lu Ye, Chen Yao, Li Tao, et al. Convolutional neural network construction method for embedded FPGAs oriented edge computing [J]. Journal of Computer Research and Development, 2018, 55(3): 551-562 (in Chinese)
(卢冶, 陈瑶, 李涛, 等. 面向边缘计算的嵌入式 FPGA 卷积神经网络构建方法[J]. 计算机研究与发展, 2018, 55(3): 551-562)

[34] Lu Liqiang, Liang Yun, Xiao Qingcheng, et al. Evaluating fast algorithms for convolutional neural networks on FPGAs [C] //Proc of the 25th Annual Int Symp on Field-Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2017: 101-108

[35] Cong J, Wang Jie. PolySA: Polyhedral-based systolic array auto-compilation [C/OL] //Proc of the 31st IEEE/ACM Int Conf on Computer-Aided Design(ICCAD). Piscataway, NJ: IEEE, 2018 [2020-04-08]. https://ieeexplore.ieee.org/abstract/document/8587678

[36] Chen Guilin, Ma Sheng, Guo Yang. Survey on accelerating neural network with hardware [J]. Journal of Computer Research and Development, 2019, 56(2): 240-253 (in Chinese)
(陈桂林, 马胜, 郭阳. 硬件加速神经网络综述[J]. 计算机研究与发展, 2019, 56(2): 240-253)

[37] Du Zidong, Fasthuber R, Chen Tianshi, et al. ShiDianNao: Shifting vision processing closer to the sensor [C] //Proc of the 42nd Annual Int Symp on Computer Architecture. New York: ACM, 2015: 92-104

[38] Li Huimin, Fan Xitian, Jiao Li, et al. A high performance FPGA-based accelerator for large-scale convolutional neural networks [C/OL] //Proc of the 26th Int Conf on Field Programmable Logic and Applications (FPL). Piscataway, NJ: IEEE, 2016 [2021-02-12]. https://ieeexplore.ieee.org/abstract/document/7577308

[39] Qiu Jiantao, Wang Jie, Yao Song, et al. Going deeper with embedded FPGA platform for convolutional neural network [C] //Proc of the 24th ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2016: 26-35

[40] Motamedi M, Gysel P, Akella V, et al. Design space exploration of FPGA-based deep convolutional neural networks [C] //Proc of the 21st Asia and South Pacific Design Automation Conf (ASP-DAC). Piscataway, NJ: IEEE, 2016: 575-580

[41] Mu Jiandong, Zhang Wei, Liang Hao, et al. Optimizing OpenCL-based CNN design on FPGA with comprehensive design space exploration and collaborative performance modeling [J]. ACM Transactions on Reconfigurable Technology and Systems, 2020, 13(3): 1-28

[42] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [J]. Advances in Neural Information Processing Systems, 2012, 25: 1097-1105

[43] Xilinx. Vivado design suite: AXI reference guide [EB/OL]. [2020-09-12]. https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf

[44] Han Song, Mao Huizi, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding [J]. arXiv preprint, arXiv: 1510.00149, 2015

[45] Zhao Jieru, Feng Liang, Sinha S, et al. Performance modeling and directives optimization for high level synthesis on FPGA [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(7): 1428-1441

[46] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition [J]. arXiv preprint, arXiv: 1409.1556, 2014

[47] Xilinx. Xilinx Vitis: Unified software platform for all developers [EB/OL]. [2020-12-05]. https://www.xilinx.com/products/design-tools/vitis.html

Xie Kunpeng, born in 1996. PhD candidate. Member of CCF. His main research interests include heterogeneous computing, machine learning and embedded system.

谢坤鹏,1996 年生.博士研究生,CCF 会员.主要研究方向为异构计算、机器学习和嵌入式系统.

Lu Ye, born in 1986. PhD, associate professor. Senior member of CCF. His main research interests include embedded system, heterogeneous computing and artificial intelligence.

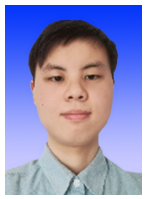
卢　冶,1986 年生.博士,副教授,CCF 高级会员.主要研究方向为嵌入式系统、异构计算和人工智能.

Jin Zongming, born in 1997. Master candidate. His main research interests include cloud computing, IoT security and heterogeneous computing.

靳宗明,1997 年生.硕士研究生.主要研究方向为云计算、物联网安全和异构计算.

Liu Yiqing, born in 1994. Master candidate. His main research interests include DNN model pruning, model transformation tool and FPGA deployment implementation.

刘义情,1994 年生.硕士研究生.主要研究方向为 DNN 模型剪枝、模型转换工具与 FPGA 部署实现.



Gong Cheng, born in 1993. PhD candidate. His main research interests include heterogeneous computing, machine learning and IoT.

龚　成,1993 年生.博士研究生.主要研究方向为异构计算、机器学习和物联网.

Chen Xinwei, born in 1984. PhD, associate professor. Member of CCF. His main research interests include robot control technology, industrial vision system, mobile robot system.

陈新伟,1984 年生.博士,副教授,CCF 会员.主要研究方向为机器人控制技术、工业视觉系统和移动机器人系统.

Li Tao, born in 1977. PhD, professor. Distinguished member of CCF. His main research interests include heterogeneous computing, machine learning and IoT.

李　涛,1977 年生.博士,教授,CCF 杰出会员.主要研究方向为异构计算、机器学习和物联网.