基于 ZYNQ 的可重构卷积神经网络加速器

刘 杰[1],葛一凡[1],田 明[2],马力强[1]
(1. 哈尔滨理工大学电气与电子工程学院,黑龙江哈尔滨 150080; 2. 中国电信黑龙江省分公司,黑龙江哈尔滨 150000)

摘 要: 针对卷积神经网络中卷积运算复杂度高、计算量大及算法在 CPU 和 GPU 上计算时存在延时及功耗限制问题,从提高现有硬件平台计算速率、降低功耗角度出发,设计了一种基于 ZYNQ 的具有高吞吐率和低功耗的可重构神经网络加速系统. 为充分利用运算资源,探索了一种卷积运算循环优化电路;为降低带宽访问量,设计了一种数据在内存中的特殊排列方式. 以 VGG16 网络为例,利用 ZYNQ 对系统进行加速,在计算性能上达到 62. 00GPOS 的有效算力,分别是 GPU 和 CPU 的 2. 58 倍和 6. 88 倍,其 MAC 利用率高达 98. 20%,逼近 Roofline 模型理论值. 加速器的计算功耗为 2. 0W,能效比为 31. 00GOPS/W,是 GPU 的 112. 77 倍和 CPU 的 334. 41 倍.

关键词: FPGA; 卷积神经网络; Roofline 模型; 硬件加速
中图分类号: TN47 **文献标识码:** A **文章编号:** 0372-2112 (2021)04-0729-07
电子学报 URL: http://www. ejournal. org. cn **DOI:** 10. 12263/DZXB. 20200409

Reconfigurable Convolutional Network Accelerator Based on ZYNQ

LIU Jie[1], GE Yi-fan[1], TIAN Ming[2], MA Li-qiang[1]
(1. *College of Electrical and Electronic Engineering, Harbin University of Science and Technology, Harbin, Heilongjiang* 150080, *China*;
2. *Heilongjiang Branch of China Telecom, Harbin, Heilongjiang* 150000, *China*)

Abstract: Aiming at the problems of high complexity of convolution operation, large amount of calculation and the limitation of delay and power consumption when the algorithm is calculated on the CPU and GPU in the convolutional neural network, from the perspective of increasing the calculation rate and reducing power consumption of existing hardware platforms, a reconfigurable neural network acceleration system with high throughput and low power consumption based on ZYNQ is presented. In order to make full use of computing resources, a convolution operation loop optimization circuit is explored; in order to reduce the bandwidth access, a special arrangement of the data in memory is designed. Taking the VGG16 network as an example, using ZYNQ to accelerate the system, 62. 00 GPOS effective computing power was reached, which was 2. 58 times and 6. 88 times that of the GPU and CPU respectively. Its MAC utilization rate was as high as 98. 20%, which was close to the theoretical value of the Roofline model. The computing power consumption of the accelerator was 2. 0W, and the energy efficiency ratio was 31. 00GOPS/W, which was 112. 77 times that of the GPU and 334. 41 times that of the CPU.

Key words: FPGA; convolutional neural network; Roofline model; hardware acceleration

1 引言

卷积神经网络(CNN)是由人工神经网络扩展而来的一种深度学习结构,已被广泛应用于计算机视觉、图像识别与处理、自然语言识别等领域中[1,2]. 随着深度学习研究的不断深入,CNN 模型变得越来越大,参数和层数不断增加,需要大量的操作和数据访问,人们关注的重点由传统的 CPU 硬件平台转向 GPU、FPGA 等高性能计算平台. GPU 包含大量的计算单元和高内存带宽,在对数据进行算术运算或者逻辑运算方面有着很大的优势,然而 GPU 功耗大,不适合移动应用. 相比之下,FPGA 效率高、适应性强,更适用于嵌入式设备[3,4].

目前,CNN 加速器的设计方法主要分为定制化和自动化两种,定制化方法根据特定的网络模型及目标器件给出细粒度的优化方案,由此实现较高的性能[5,6];自动化方法提出通用的设计方案,可适用于不

收稿日期:2020-04-29;修回日期:2020-07-14;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 5177090001)

同的网络模型或目标器件. 随着网络模型的快速发展、目标器件的不断更新,自动化方法能较好地满足不同用户的需求. 文献[7]针对 FPGA 数据传输问题,利用大量的寄存器代替 BRAM,但使得 FPGA 片上 BRAM 资源利用率不高,造成资源的浪费,同时占用大量的其他资源;文献[8]针对 FPGA 计算吞吐和提供的内存带宽不匹配问题,对 CNN 卷积层设计了相应加速结构,但并未考虑全连接层,且计算涉及的数据为浮点数,并不是定点数,影响了 FPGA 整体的执行性能. 本文根据 CNN 算法的计算特点探索了一种可并行计算的卷积运算循环优化电路结构,充分利用了硬件平台的运算资源,降低了功耗;同时提出了一种计算数据在内存中的特殊排列方式,降低了带宽访问量.

2 基本概念

2.1 CNN 基本原理

卷积神经网络在计算机视觉、人工智能等研究领域受到了越来越多的关注[9],作为一种典型的监督学习算法,CNN 有两个主要阶段:训练阶段和推理(也称为前馈)阶段. 由于许多行业应用程序都是在后台训练 CNN,并且仅在实时情况下执行推理,因此本文中重点关注推理阶段. CNN 推断阶段的目的是为了输入图像获得正确的分类推断. 如图 1 所示,它由多层网络组成,每张图像均送入到第一层. 每一层从上一层接收大量特征图,并通过某些内核过滤后输出一组新的特征图[10~12]. 卷积层、池化层和激活层用于特征图的提取,全连接层用于分类.

2.2 实际的 CNN 分析

近年来在 CNN 网络竞赛中获取冠军的模型包括 AlexNet、ZFNet 及 VGG 等. 其中,2012 年的 AlexNet 模型层数为 8 层,模型大小为 250M;2014 年的 VGG 模型层数为 16 层,模型大小为 500M. 而模型规模的增大和计算量的大幅度提升意味着对硬件计算能力、内存带宽及数据存储的要求也更加苛刻,故需要对模型进行处理,使之对硬件更加友好[5].

表 1 的数据是 16 层 VGG16 模型时间与空间上的分解参数,从中可看出,CONV 和 FCN 层是两个极端特征. CONV 层包含总数据的 19.3%,但需要 99.5% 的计算量;FCN 层需要 0.4% 的计算量,但使用 80.6% 的数据量. 这两层占据了超过 99.9% 的执行时间. 因此,工程上的优化设计需要对 FPGA 上的整个 CNN 进行加速,并最大限度地利用 FPGA 的计算能力和带宽效率. 由于 POOL 和 ReLU 层的简单性足以使其简单加速,因此本文将主要讨论如何加速 CONV 和 FCN 层[13,14].

表 1 VGG16 模型分解

类别	CONV	POOL	ReLU	FCN
计算量 ops (10^7)	3059.6 (99.5%)	0.6 (0%)	1.4 (0%)	12.3 (0.4%)
存储(MB)	113 (19.3%)	0 (0%)	0 (0%)	471.6 (80.6%)
纯电路计算时间比	93.6%	0%	0%	3.7%
整体计算时间比	48.7%	0.0%	0.0%	51.2%

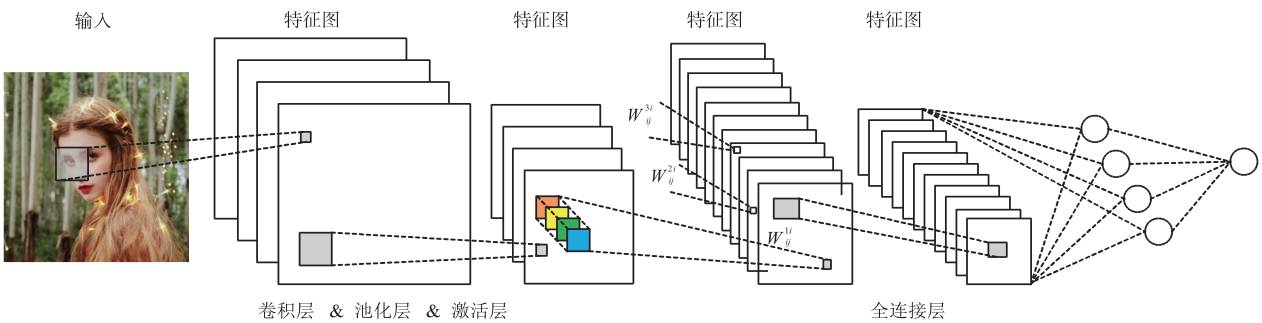

图1 CNN提取特征分类过程图

2.3 Roofline 评估模型

算法在某个硬件平台上的运算时间受两种因素影响:一是硬件平台的算力(计算平台单位时间内能完成的浮点运算次数);二是硬件平台的带宽(计算平台单位时间内能完成的内存交换量). Roofline 模型中使用了计算强度进行定量分析,可更准确地判断运算时间是由算力影响或带宽影响,及它们的影响程度[15]. 因此,本文采用 Roofline 模型评估所设计加速器的性能.

如图 2 所示,Roofline 模型中,横轴表示计算强度,用 I 表示,单位为 FLOP/Byte;纵轴表示算力,用 P 表示,单位为 FLOP/s;π 表示硬件平台的峰值算力;β 表示硬件平台的最大带宽;I_{max} 表示计算强度的上限,大小为 π/β. 图 2 中,I_{max} 将 Roofline 模型划分出两个瓶颈区域:带宽瓶颈区域和计算瓶颈区域[16].

3 本文加速器的设计

3.1 卷积运算电路的设计

CNN 中需要进行大量的卷积运算,因此卷积层的

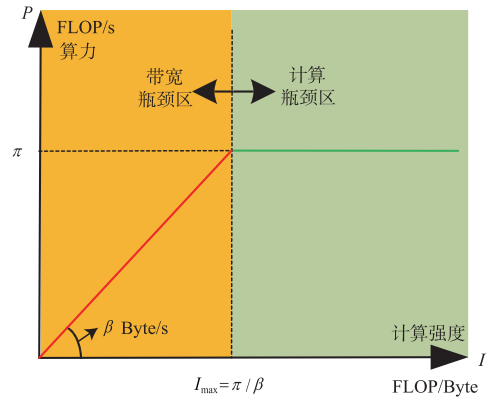

图2　Roofline模型图

电路设计是硬件加速器设计的核心. 为了充分利用运算资源,本文采用循环展开和循环平铺对卷积运算进行优化,具体分块伪代码如图 3 所示.

```
for(k=0;k<ceil(CHout/`K);k=k+1){                          Loop-1
    for(p=0;p<ceil((Hout•Wout)/`K);p=p+1){                Loop-2
        for(c=0;c<ceil(CHin/`K);c=c+1){                   Loop-3
            for(r=0;r<ky;r=r+1){                          Loop-4
                for(s=0;s<kx;s=s+1){
                    for(pp=0;pp<`K;pp=pp+1){              Loop-5
                        for(kk=0;kk<`K;kk=kk+1){          Loop-6
                            for(cc=0;cc<`K;cc=cc++){
sum[pp][kk]+=x[h•Sy-Pad_up+r][w•Sx-Pad_left+s][c•`K+cc]•wt[r][s][c•`K+cc][k•`K+kk];
                    }   }   }
                    for(pp=0;pp<`K;pp=pp+1){              Loop-7
                        for(kk=0;kk<`K;kk=kk+1){          Loop-8
                            out[h][w][k•`K+kk]=sum[pp][kk];
}   }   }   }   }   }   }
```

图3　卷积伪代码

Loop-1 表示多个通道组的循环运算,Loop-2 表示输出通道组的循环运算,Loop-3 表示输入通道组的循环运算. 由于该三个循环不涉及电路的存储与数据的计算,故三个循环可以用计数器电路完成.

Loop-4 表示卷积核的块运算,为提高工作效率,该循环采用流水线“乒乓操作”方式,如图 4 所示. 若无带宽限制,采用流水线后乘加器阵列的工作效率可接近 100%.

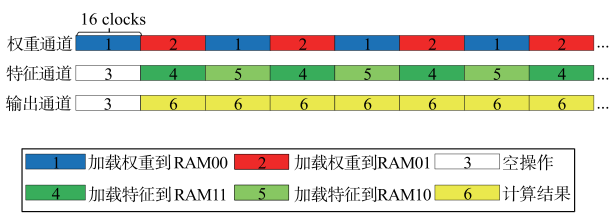

图4　Loop-4时序示意图

Loop-5 为条带运算,如图 5 所示,特征 01 数据加载至计算单元,经一个时钟周期完成计算并输出乘加结果 01,随后重复上述过程,直至输出乘加结果 16,在此期间权重数据保持不变,被 16 组特征数据复用,直至下次循环启动后再重新更换权重数据. Loop-6 为 MAC 运算,该运算为权重数据和特征数据相乘,其中权重数据是 16×16 的矩阵,特征数据是 16×1 的矩阵,完成该矩阵的计算需要 256 次 MAC 运算. Loop-7 用于累加输入通道组的运算结果,由于输入通道数较多,故采用加法树进行多个数据的累加. Loop-8 用于运送累加后结果至输出通道组,故实现 Loop-8 的电路结构可采用计数器将结果逐个送出.

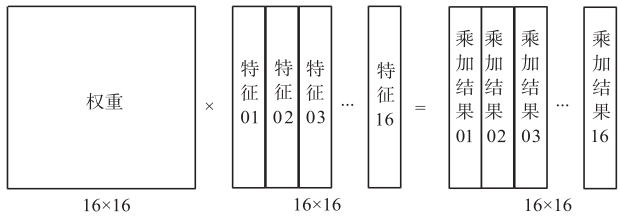

图5　Loop-5的计算过程示意图

本文对卷积运算电路优化后,得到循环 Loop-5 单个处理单元(Processing Engine,PE)的硬件资源结构,如图 6 所示,图中乘加器阵列为实现上述运算的硬件计算结构,加法树用于累加运算数据,随后运算数据与偏置项(Bias)相加后输出最终结果。单个 PE 中有 16 个相同的 MAC 结构,单个 MAC 结构中有 16 个乘加器,故一个 PE 中有 256 个乘加器,因此本文设计的优化电路 Loop-5 通过 PE 电路结构,一次即可完成 256 次乘加运算.

3.2　权重和特征数据的排列设计

CNN 中卷积计算的本质是权重与特征乘累加,本文提出一种权重和特征数据在内存中的特殊排列方式. 如图 7 所示,将四维的权重数据展开成三维的数据,每组包含 16 个三维数据,单个三维数据中以 16 个通道为一个子块,图中数据与外部存储器中相同的颜色表示排列后的对应关系. 权重数据具体的排列方式为:第一组数据中的所有 00 子块按 W0、W1、…、W15 的顺序排列于外部存储器,之后排列 01 子块、…、08 子块、09 子块、…、17 子块,按照上述方式,直至排列完成第 K 组权重数据.

如图 8 所示,将一组三维特征数据分为多个子块,单个子块的通道数为 16,分块后特征数据按照图中的排布方式进行排列. 在设计过程中,不同网络层中的特征通道数可能无法被 16 整除,此时在排列中通道数与分块数不足 16 的位置全部补 0.

3.3　池化电路的设计

由于本文所设计的加速电路是一种可重构的加速器,因此池化电路应支持以上不同类型及不同方式的池化.

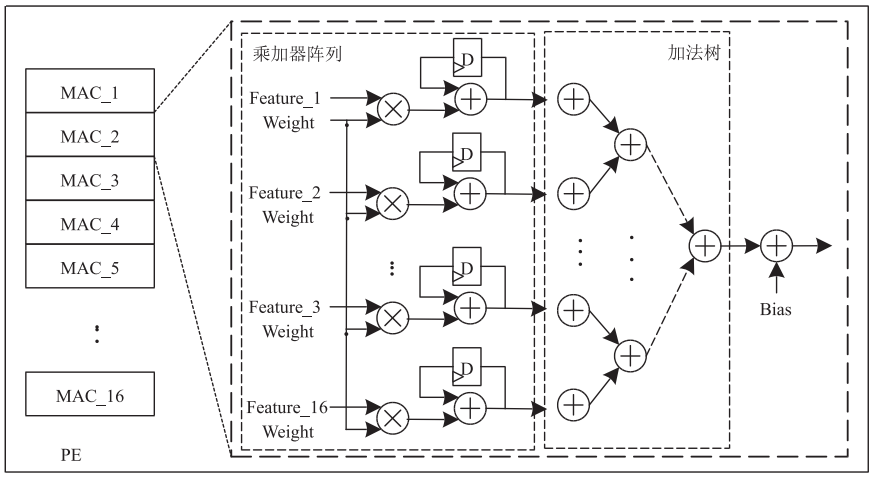

图6 单个PE的硬件资源结构图

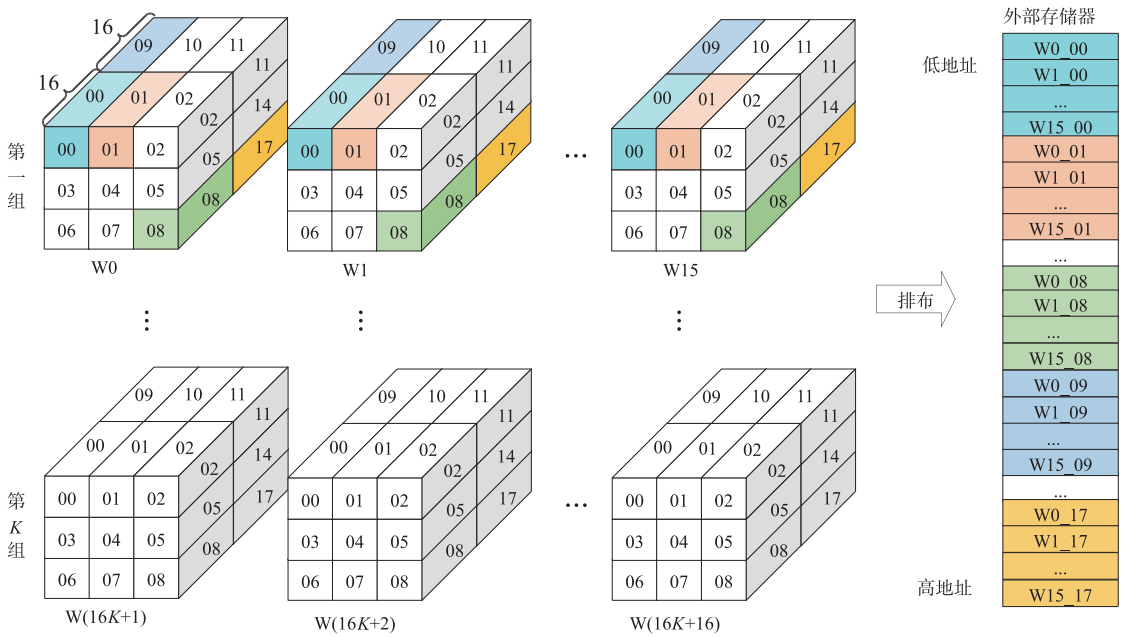

图7 权重数据在内存中的排布方式

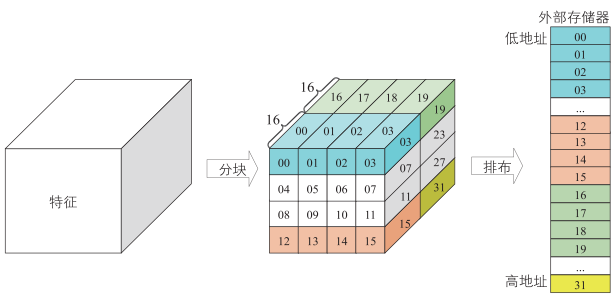

图8 特征数据在内存中的排布方式

本文采用先横向池化,后纵向池化的计算方式,如图9所示,横向相邻的两组数据经过比较运算后存入一块缓存器中,当第二行特征数据加载至片上时,即可完成纵向相邻的两组数据比较运算. 上述电路结构有

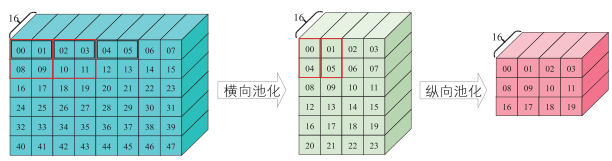

图9 池化方式图

利于降低数据的片上缓存.

4 实验与结果分析

4.1 软硬件实验环境

本文所测试的神经网络模型为 VGG16[17],该网络模型的操作数大约为 30GOPS;使用的目标器件为 Xilinx 评估板 ZCU104;CPU 对照组采用英特尔 Core i9-9900K 处理器,主频为 3.60GHz,处理器数量为 8 核 16

线程,配有两个 16GB 的 DDR4 内存,Windows10 操作系统;GPU 对照组采用 Nvidia GTX 1080Ti 芯片,拥有 3584 个 CUDA 核心,核心频率 1708MHz,配有 11GB 的显存,Linux 操作系统;输入图像模型为 ImageNet 数据集. 为了保证时序稳健的运行,在 ZCU104 开发板上对本设计进行了 210MHz 的实验,加速器的资源利用率如表 2 所示,其中 1 个 DSP 资源可以支持 1 个 16bit 的定点乘法.

表 2　加速器的资源利用率

类别	FF	LUT	DSP48	BRAM
资源总数	460800	230400	1726	312KB
使用数	317645	160032	530	280KB
使用率	68.9%	69.5%	30.7%	89.7%

从表 2 可看出,DSP48 资源使用率较低,其原因是:若使用更多的 DSP48 资源需配合更多的 BRAM,而 BRAM 的使用率已达到 89.7%,因此仅使用了 30.7% 的 DSP48.

本文使用 Vivado 2018.2 对设计完成综合实现. 由于神经网络有更好的鲁棒性,16bit 的定点数可以满足加速器设计的精度[14],为节省硬件资源,本文将训练的 32bit 的浮点数截断为 16bit 的定点数,因此每层网络的计算结果也均采用该精度.

4.2　性能评估

本次测试使用 ImageNet 数据集中的 20 张图片,分别送入 ZCU104、GPU、CPU 计算平台,对每张的图片在不同计算平台上的处理时间、VGG16 网络每层的运算时间和运算量及不同平台的计算功耗进行测试,其结果分别如图 10、图 11 和表 3 所示.

从图 10 可看出,VGG16 网络在该加速器上识别一张图片的平均时间约为 0.25s,计算速率分别是 GPU 与 CPU 的 5.16 倍、13.76 倍,三组数据对应的均方差分别是 4.3×10^{-5}、1.0×10^{-2}、4.9×10^{-2},说明本文设计的加速器在 ZCU104 上处理时间更加稳定.

从图 11 可看出,卷积层和全连接层的运算时间占整体的 99.11%;池化层的运算主要是逻辑运算或加法运算,并不使用 DSP 运算,故池化层的运算量为 0;卷积层的运算量较大,运算时间较长,但全连接层的运算量相比卷积层较小,运算时间却与卷积层相当,比如 CONV10 与 FC1,其主要原因是数据的传输延时决定了全连接层的实际计算时间.

表 3　不同计算平台的功耗

类别	ZCU104	GPU	CPU
待机功耗/W	14.3	57.2	61.6
峰值功耗/W	16.3	144.6	158.8
实际功耗/W	2.0	87.4	97.2

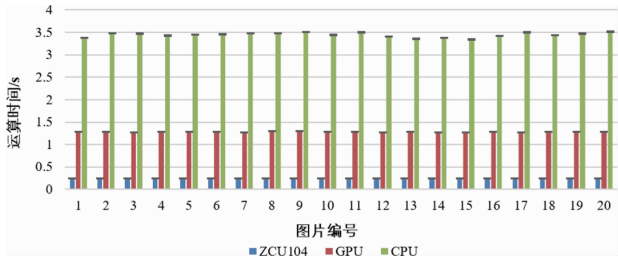

图10　单张图片的处理时间

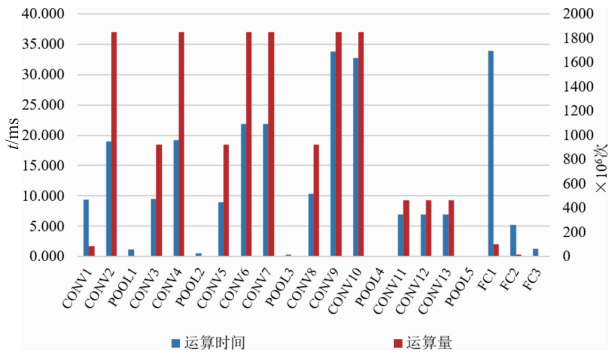

图11　各层的运算时间与运算量

从表 3 中的数据可知,该加速器实际功耗仅为 2W,远低于 GPU 与 CPU,适合使用于嵌入式移动设备.

VGG16 网络的 16 位定点数运算量为 15.5G,故该加速器每秒的有效算力为 15.5/0.25 = 62GOPS,能效比为 62/2 = 31GOPS/W. 用相同的标准对 CPU 和 GPU 分析得到表 4 中的数据.

表 4　平台间数据对比

类别	ZCU104	GPU	CPU
单张图片计算时间/s	0.25	1.29	3.44
有效算力/GOPS	62.00	24.03	9.01
实际功耗/W	2.0	87.4	97.2
能效比/GOPS/W	31.0000	0.2749	0.0927

从表 4 可看出,该加速器的有效算力是 GPU 的 2.58 倍,CPU 的 6.88 倍;能效比是 GPU 的 112.77 倍,CPU 的 334.41 倍. 因此,本文设计的加速器有着较好的性能收益.

4.3　Roofline 模型分析

如图 12 所示,图中的折线表示加速器在 Roofline 模型下的最大理论峰值算力,纵坐标表示加速器的峰值算力,横坐标表示计算强度,斜率表示加速器的存储器访问带宽速率. 从图 12 中可以看出,仅有第 11、12、13 层卷积层相差理论值较大,其原因是大规模的卷积切割后使得最后一块的规模较小,不能使 MAC 层计算满载;虽然第 1 层卷积层和 3 层全连接层的峰值算力较低,但仍然逼近 Roofline 模型的理论值;其他网络层均逼近 Roofline 模型的理论值,最高的 MAC 利用率高达 98.12%,因此该加速电路性能优良.

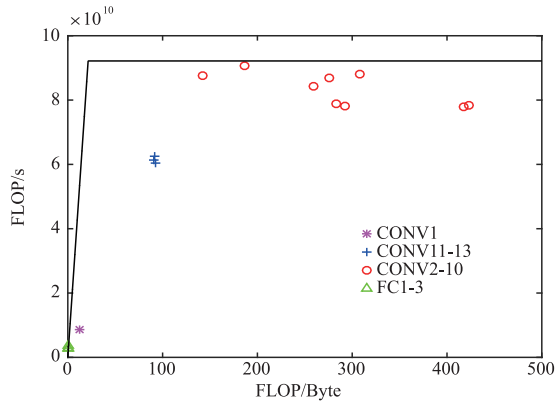

图12 Roofline模型评估

表5 相关工作比较

类别	文献[18]	文献[19]	文献[20]	文献[21]	本文
设备	ZC706	XC7Z020	Kintex-7	Arria 10	ZCU104
DSP使用数	900	205	284	2852	530
数据精度	8-16Fix	16Fix	16Fix	16Fix	16Fix
功耗/W	/	2.61	/	/	2.0
吞吐量/GOPS	58.3	17.78	20.49	166.67	62.00
能效比/GOPS/W	/	6.81	/	/	31.00
计算密度/GOPS/DSP	6.4×10^{-2}	8.1×10^{-2}	7.2×10^{-2}	5.8×10^{-2}	11.7×10^{-2}

为了进一步评估加速电路的计算性能,将本研究与近年的CNN加速器进行对比,由于各个文献中使用的评估板和运算资源都有区别,数据精度也有差异,因此本文使用计算密度(单个DSP下的有效算力)来评估性能.对比文献[18~21],从表5中可以看出,本文设计加速器的能效比高,计算密度依次提高了82.8%、44.4%、62.5%、101.7%.由于文献[21]中的DSP的使用量远高于本文,故其吞吐量优于本文,但其计算密度要低于本文.

综上所述,本文提出的方案在资源和功耗受限的情况下可以提供更高的计算性能.

5 总结

本文从提高计算速度和提高能效比的角度出发,设计了一种基于ZYNQ的可重构神经网络加速系统,通过软硬件的协同设计,优化了卷积层电路和池化电路及降低了带宽访问量.加速器在计算性能上达到62.00GPOS的有效算力,分别是GPU和CPU的2.58倍和6.88倍,其MAC利用率高达98.20%,逼近Roofline模型理论值.加速器的计算功耗为2.0W,能效比为31.00GOPS/W,是GPU的112.77倍和CPU的334.41倍.实验结果表明,本文提出的CNN加速器适合应用于嵌入式设备.与近年来相关领域文献对比,本文提出的方案在资源和功耗受限的情况下可以提供更高的性能.同时该加速器适用于其它应用神经网络架构的系统,具有较高的应用推广价值面.

参考文献

[1] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognition[A]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition[C]. Los Alamitos, USA: IEEE, 2016. 770-778.

[2] SZEGEDY C, LIU W, JIA Y Q, et al. Going deeper with convolutions[A]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition[C]. Los Alamitos, USA: IEEE, 2015. 1-9.

[3] NURVITADHI E, VENKATESH G, SIM J, et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks[A]. Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. New York, USA: ACM, 2017. 5-14.

[4] 余乐,李任伟,王瑶,等. 综述:面向SoC-FPGA的开源处理器[J]. 电子学报,2018,46(4):992-1004.
YU Le, LI Ren-wei, WANG Yao, et al. Open source processors for SoC-FPGA: A survey[J]. Acta Electronica Sinica, 2018, 46(4): 992-1004. (in Chinese)

[5] ZHANG J L, LI J. Improving the performance of Open CL-based FPGA accelerator for convolutional neural network[A]. Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. New York, USA: ACM, 2017. 25-34.

[6] MA Y F, Cao Y, VRUDHULA S, et al. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks[A]. Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. New York, USA: ACM, 2017. 45-54.

[7] ZHOU Y, JIANG J. An FPGA-based accelerator implementation for deep convolutional neural networks[A]. Proceedings of the 4th International Conference on Computer Science and Network Technology[C]. Harbin, China: IEEE, 2015. 829-832.

[8] ZHANG C, LI P, SUN G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks[A]. Proceedings of the 2015 ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. Monterey, USA: ACM, 2015. 161-170.

[9] KOEPLINGER D, PRABHAKAR R, ZHANG Y Q, et al. Automatic generation of efficient accelerators for reconfigurable hardware[A]. Proceedings of the 43rd ACM/IEEE Annual International Symposium on Computer Architecture[C]. Los Alamitos, USA: IEEE, 2016. 115 – 127.

[10] MA Y F, CAO Y, VRUDHULA S, et al. An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks[A]. Proceedings of the 27th International Conference on Field Programmable Logic and Applications[C]. Los Alamitos, USA: IEEE, 2017. 1 – 8.

[11] GUAN Y J, LIANG H, XU N Y, et al. FP-DNN: an automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates[A]. Proceedings of the 25th IEEE Annual International Symposium on Field Programmable Custom Computing Machines[C]. Los Alamitos, USA: IEEE, 2017. 152 – 159.

[12] VENIERIS S I, BOUGANIS C S. FPGA Conv Net: a framework for mapping convolutional neural networks on FPGAs[A]. Proceedings of the 24th IEEE Annual International Symposium on Field Programmable Custom Computing Machines[C]. Los Alamitos, USA: IEEE, 2016. 40 – 47.

[13] MA Y F, CAO Y. Optimizing the convolution operation to accelerate deep neural networks on FPGA[A]. Proceedings of the IEEE Transactions on Very Large Scale Integration (VLSI) Systems[C]. Los Alamitos, USA: IEEE, 2018. 1354 – 1367.

[14] ZHANG C, PENG L, SUN G Y. Optimizing FPGA-based accelerator design for deep convolutional neural networks[A]. Proceedings of the 2015 ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. Monterey, USA: ACM, 2015. 161 – 170.

[15] ZHANG J L, LI J. Unleashing the power of soft logic for convolutional neural network acceleration via product quantization[A]. Proceedings of the 2019 ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. CA, USA: ACM, 2019. 120 – 120.

[16] LU L Q, LIANG Y, HUANG R, et al. Speedy: an accelerator for sparse convolutional neural networks on FPGAs[A]. Proceedings of the 2019 ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. CA, USA: ACM, 2019. 187 – 187.

[17] ZHANG C, PRASANNA V. Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system[A]. Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. New York, USA: ACM, 2017. 35 – 44.

[18] 蹇强,张培勇,王雪洁. 一种可配置的 CNN 协加速器的 FPGA 实现方法[J]. 电子学报, 2019, 47(7): 1525 – 1531.
JIAN Qiang, ZHANG Pei-yong, WANG Xue-jie. An FPGA implementation method of configurable CNN co-accelerator[J]. Acta Electronica Sinica, 2019, 47(7): 1512 – 1531. (in Chinese)

[19] 张榜,来金梅. 一种基于 FPGA 的卷积神经网络加速器的设计与实现[J]. 复旦学报(自然科学版), 2018, 57(02): 236 – 242.
ZHANG B, LAI J M. Design and Implementation of a FPGA-based accelerator for convolutional neural networks[J]. Journal of Fudan University (Natural Science), 2018, 57(02): 236 – 242. (in Chinese)

[20] 王巍,周凯利,王伊昌,等. 基于快速滤波算法的卷积神经网络加速器设计[J]. 电子与信息学报, 2019, 41(11): 2578 – 2584.
WANG W, ZHOU K L, WANG Y C, et al. Design of convolutional neural networks accelerator based on fast filter algorithm[J]. Journal of Electronics & Information Technology, 2019, 41(11): 2578 – 2584. (in Chinese)

[21] 陆维娜,胡瑜,叶靖,等. 面向卷积神经网络加速器吞吐量优化的 FPGA 自动化设计方法[J]. 计算机辅助设计与图形学学报, 2018, 30(11): 189 – 198.
LU W N, HU Y, YE J, et al. Throughput-oriented automatic design of FPGA accelerator for convolutional neural networks[J]. Journal of Computer-Aided Design & Computer Graphics, 2018, 30(11): 189 – 198. (in Chinese)

作者简介

刘　杰　女,1980 年出生,黑龙江齐齐哈尔人. 2002 年、2005 年和 2013 年分别在哈尔滨理工大学、哈尔滨工业大学和哈尔滨理工大学获工学学士、工学硕士和工学博士学位. 现为哈尔滨理工大学电气与电子工程学院副教授,主要从事 FPGA 设计和深度学习算法优化.
E-mail: liujie@ hrbust. edu. cn

葛一凡　男,1997 年出生,山东聊城人. 哈尔滨理工大学电气与电子工程学院硕士研究生,主要研究方向为深度学习算法优化及应用.