

基于 FPGA 的卷积神经网络设计与实现

蒋 林¹, 王喜娟¹, 刘镇弢¹, 谢晓燕², 衡 茜¹

(1 西安邮电大学 电子工程学院, 陕西 西安 710121;

2 西安邮电大学 计算机学院, 陕西 西安 710121)

摘 要: 卷积神经网络(Convolutional Neural Network, CNN)在各种计算机视觉应用中取得了巨大成功. 本文研究了卷积神经网络的并行结构, 基于网络计算的多种并行特征, 提出了 CNN 前向传播过程在 FPGA 并行计算的架构. 实验结果表明, 在 110MHz 的工作频率下, 该结构可使 FPGA 的峰值运算速度达到 0.48GOP/s, 相较 ARM Mali-T628 GPU 平台实现 23.5 倍的加速比.

关键词: 卷积神经网络; 现场可编程门阵列; 阵列处理器; 并行性

中图分类号: TP302

文献标识码: A

文章编号: 1000—7180(2018)08—0132—05

Design and Implementation of Convolutional Neural Network Based on FPGA

JIANG Lin¹, WANG Xi-juan¹, LIU Zhen-tao¹, XIE Xiao-yan², HENG Qian¹

(1 College of Electronic Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China;

2 College of Computer Science, Xi'an University of Posts and Telecommunications, Xi'an 710121, China)

Abstract: Convolutional neural networks (CNN) have achieved great success in various computer vision applications. The parallel architecture of convolutional neural networks were studied in this paper. Based on the parallel characteristics of network computing, a parallel CNN forward propagation architecture was proposed. The experimental results showed that under the operating frequency of 110MHz, the architecture could make the FPGA peak operating speed of 0.48GOP/s, compared to the ARM Mali-T628GPU platform to achieve $23.5 \times$ speed.

Key words: convolutional neural network; field-programmable gate array; Array processor; parallelism

1 引言

卷积神经网络算法在机器学习领域越来越受欢迎, 并取得了突破性的成果, 文献[1]提出大规模递归神经网络 GPU 的高效实现, 并证明了该实现在 GPU 上的扩展性. 文献[2]通过一系列的参数配置, 对基于 GPU 的一系列 CNN 开源实现进行全面比较, 分析潜在的性能瓶颈, 并指出进一步的优化方法. 虽然 GPU 具有高度并行计算能力, 能较好的解决计算速度问题, 然而基于 GPU 的视觉系统往往存在功耗较高、体积较大和成本高问题.

为了提升计算能力的同时尽可能降低功耗, 具有大量阵列形式逻辑、运算单元的现场可编程门阵列芯片则在尺寸上、功耗和并行运算方面有突出的优势^[3]. 借助 FPGA 中丰富的逻辑资源和专用乘法器、数字信号处理等资源实现的卷积神经网络, 能高度并行地执行算法中大量重复、独立的乘法和加法运算.

本文基于 FPGA 根据模型特征, 分析网络计算的并行性特点, 针对 CNN 前向传播过程中不同层设计出合适的并行与流水线结构, 并且和已有的 GPU 的实现进行了比较, 得到不错的结果.

收稿日期: 2017—11—16; **修回日期:** 2017—12—22

基金项目: 国家自然科学基金项目(61772417, 61602377, 61634004, 61272120); 陕西省科技统筹创新工程项目(2016KTZDGY02-04-02); 陕西省重点研发计划项目(2017GY-060)

2 CNN 网络

2.1 CNN 网络架构

本文以实现 AlexNet 为例介绍 CNN 网络架构,如图 1.该模型共 8 层,5 个卷积层及 3 个全连接层,每一个卷积层中包含了激励函数 ReLU 以及局部响应归一化(LRN)处理,然后卷积层 C1,C2 和 C5 经过降采样(pool)处理. C1 层有 96 个 $11\times 11\times 3$ 卷积核,C2 层有 256 个 5×5 卷积核,C3,C4 层分别有 384 个 3×3 卷积核,C5 层有 256 个 3×3 卷积核,全连接层 fc6,fc7,fc8 分别采用 4096、4096、1000 个神经元进行全连接,然后通过高斯滤波器得到最终结果.

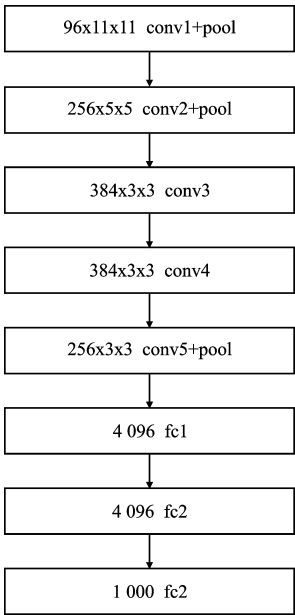


图 1 AlexNet 网络架构

2.2 模型分析

表 1 是 AlexNet 网络每一类型层计算复杂度和内存占用量^[4].由表可知:

- (1)对于单个输入图像进行分类时,CNN 操作过程需要耗费大量计算资源,AlexNet 网络大概需要 1.46 GOP. 在所有的子层中,卷积层计算量是最大的,占整个网络计算量 90%以上.
- (2)网络权值参数约为 230 MB,需要占用大量存储资源. 在所有子层中,全连接层是存储最密集的部分,占总数据量 90%以上. 参数量远超过片上存储容量,所有参数只能存储在外部内存中.
- (3)整个网络中间结果约为 5.39 MB,相比其他参数的数据量非常少.

综上所述,我们可以利用 CNN 中的潜在并行

性来提高执行过程的效率,尤其是卷积层.

表 1 AlexNet 网络总结

层类型	卷积层	全连接层	其他层
运算次数(GOP)	1.33	0.12	0.007
权重(MB)	8.90	223.60	0.000
中间数据(MB)	3.07	0.04	2.290

3 CNN 加速器设计

3.1 并行性分析

(1)卷积窗口内部并行

图像-内核卷积是卷积层的基本计算模式. 在卷积过程中,一个 $k\times k$ 的窗口向右下方扫描图像. 每次扫描执行 $k\times k$ 个乘累加操作. 如图 2(a),神经元所连接到的前级单个卷积窗口(卷积核)内部 $k\times k$ 个乘累加可以并行执行. 图中 $K_{nm}-W_{xy}$ 表示第 nm 个卷积内核中第 xy 个权值.

(2)输入特征图内部卷积窗口间并行

如图 2(b),同一输出特征图上的不同神经元通过相同的卷积窗口连接到同一输入特征图,神经元间不存在数据相关性,得到每个神经元的 $k\times k$ 个乘累加操作可以并行运算.

(3)输入特征图并行

如图 2(c),每个输出神经元所对应的乘加卷积窗口一共有 N 个(N 为输入特征图个数),即每得出一个输出值需要进行 $N\times k\times k$ 个乘加操作,这些不同输入特征图上相同位置的卷积窗口运算可以并行执行.

(4)输出特征图并行 pHam

如图 2(d),不同输出特征图上同一位置的神经元连接到前一层的卷积窗口是相同的,计算 M 个输出特征图上同一位置的神经元需要进行 $N\times M\times k\times k$ 个乘加操作,同样可以并行运算.

3.2 加速器架构

3.2.1 整体架构

加速器整体架构如图 3. 首先,从片外存储器加载输入特征图参数通过共享存储的方式传数到各个 PE(Processing Element, PE)的数据存储器;接着,控制器加载卷积操作对应的像素参数与权值参数传送到乘加模块;然后,乘加模块开始进行乘加操作计算输出特征图像素值存储到对应寄存器中,通过邻接互连方式传输到下个 PE 进行相应操作;最后,一个周期完成后,再次通过邻接互连方式将卷积运算结果从数据存储器中传送到片外存储器.

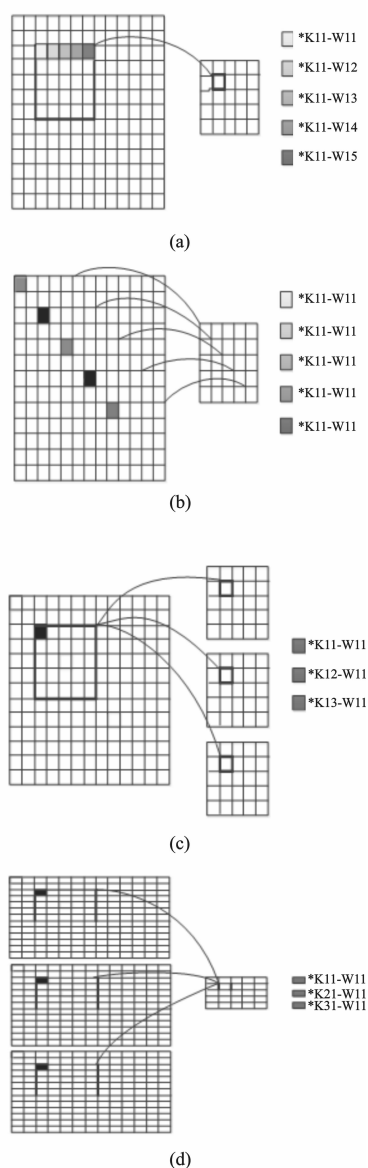


图 2 并行性分析

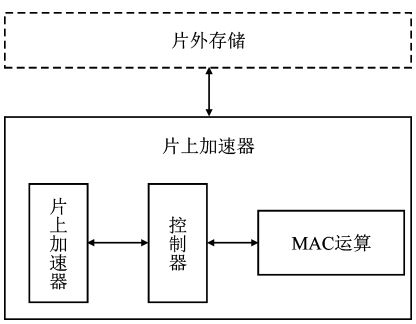


图 3 加速器整体架构

3.2.2 PE 结构

对于复杂的卷积操作,我们通过配置多个卷积器,利用循环并行的方式实现. 本文采用二维卷积器设计,如图 4. 假设卷积窗口大小为 $k \times k$,则卷积器

配置一个 $k \times k$ 乘法阵列和 $k \times k$ 加法运算. 另外,有 $k \times k$ 个权值寄存器和 $k \times k$ 个像素寄存器分别存放权重和像素值. 除此之外,计算结果通过四个共享寄存器以邻接互连方式传送到相邻 PE 实现数据交互.

(1)卷积运算之前,像素值、权重值已经被存放于数据 RAM 中,通过寄存器调用,计算结果存储到数据 RAM 中,以备重新调用;

(2)当获取到相应参数时,卷积开始有效的乘加操作;

(3)在一个循环结束后,寄存器会被更新,重新取数进行相应计算,直到结束;

(4)当卷积窗口移动到右边缘时,窗口首先向下再向右移动,重新从 DIM 中取数. 首先,将每一个块的每一行首地址存放在一个寄存器中;然后,每向右取一个数,每行首地址加一,直到取到块的最右边时,取下一行的块,即每行首地址加块的大小($k \times k$).

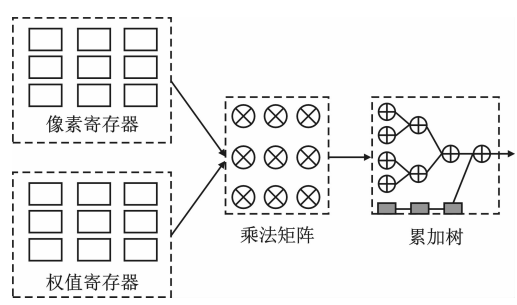


图 4 卷积器结构

3.3 卷积优化

AlexNet 网络共有 11×11 , 5×5 , 3×3 三种卷积核,由于资源受限把 11×11 的卷积操作分配在一个簇的 11 个 PE 进行运算,每个 PE 有一个乘法器,每个乘法器重复 11 次乘加操作,运算结构如图 5. PE33 主要用于加载、存储数据和执行 ReLU 运算, PE00~PE22 执行矩阵乘法和累加操作. 首先,根据相应配置命令信息从信息缓存器中读取首条乘法操作. 当所需数据到达时,乘法操作被触发, PE00~PE22 开始工作. 乘法操作运行结束后, PE 会自动切换至下一条操作, PE00 的计算结果发送给 PE10, PE01 的计算结果发送给 PE11, PE02 的计算结果发送给 PE12, PE03 的计算结果发送给 PE13, PE20 的计算结果发送给 PE21,进行第一轮加法操作. 当第一轮加法操作完成时, PE10 的计算结果发送给 PE11, PE13 的计算结果发送给 PE12, PE21 的计算结果发送给 PE22,相应 PE 开始进行第二轮加法操

作. 当第二轮加法操作完成时, PE11 的计算结果发送给 PE12, PE12 开始第三轮加法操作. 最后, PE12 的计算结果发送给 PE22, PE22 开始第四轮加法操作, 并通过 PE23 将计算结果发送给 PE33.

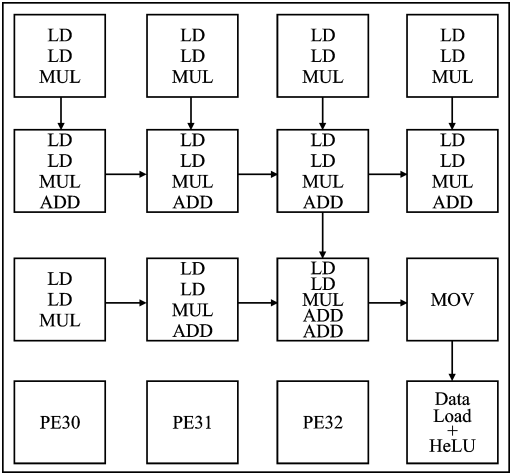


图 5 11×11 卷积运算结构图

5×5 卷积操作运算结构如图 6, 由图可知 PE33 用于加载、存储数据和执行 ReLU 运算, 其他 15 个 PE 执行乘法运算和累加操作, 每个 PE 执行一个乘法, 重复计算 5 次. 当所有的乘法操作完成时, PE 会自动切换至下一条操作, PE00 的计算结果发送给 PE10, PE01 的计算结果发送给 PE11, PE02 的计算结果发送给 PE12, PE03 的计算结果发送给 PE13, PE20 的计算结果发送给 PE21, PE30 的计算结果发送给 PE31, 进行第一轮加法操作. 当第一轮加法操作完成时, PE10 的计算结果发送给 PE11, PE13 的计算结果发送给 PE23, PE21 的计算结果发送给 PE22, PE31 的计算结果发送给 PE32, 相应 PE 开始第二轮加法操作. 当第二轮加法操作完成时, PE21 的计算结果发送给 PE22, PE32 的计算结果发送给 PE22, 相应 PE 开始第三轮加法操作. 当第三轮加法操作完成时, PE22 的计算结果发送给 PE32, PE32 进行第四轮加法操作. 最后, PE22 的计算结果发送给 PE23, PE23 进行第五轮加法操作, 并将计算结果发送给 PE33.

3×3 的卷积运算结构如图 7, PE33 用于加载、存储数据和执行 ReLU 运算, PE10~PE23 和 PE32 执行矩阵乘加操作. 当所需数据到达时, 乘法操作被触发, PE10~PE23 和 PE32 开始工作. 当乘法操作完成时, PE 会自动切换至下一条操作, PE10 的计算结果发送给 PE20, PE11 的计算结果发送给 PE21, PE12 的计算结果发送给 PE22, PE13 的计算

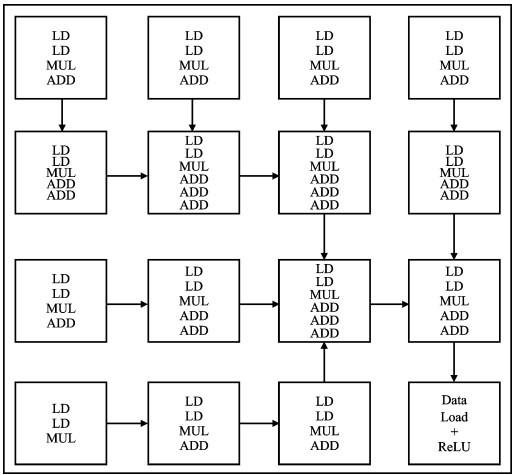


图 6 5×5 卷积运算结构图

结果发送给 PE23, 进行第一轮加法操作. 当第一轮加法操作完成时, PE20 的计算结果发送给 PE21, PE23 的计算结果发送给 PE22, 相应 PE 进行第二轮加法操作. 当第二轮加法操作完成时, PE21 的计算结果发送给 PE22, PE22 进行第三轮的加法操作. 当第三轮的加法操作完成时, PE22 的计算结果发送给 PE32, PE32 进行第四轮加法操作, 并将计算结果发送给 PE33.

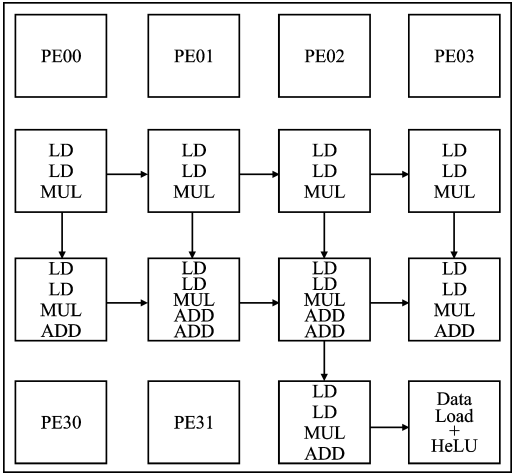


图 7 3×3 卷积运算结构图

4 实验结果和分析

本实验使用 Xilinx 公司的 ISE 进行硬件开发, 使用 QuestaSim 10.1d 进行仿真. FPGA 使用 Ver-tex—6 系列 XC6VLX760 器件, 工作频率可以达到 110 MHz.

实验结果如表 2, 一个 11×11×3 卷积运算花费 1 665 个时钟周期, C1 层生成 55×55×96 个卷积结果, 总共需要 483 516 000 个时钟周期数; 一个

5×5 卷积运算花费 428 个时钟周期,C2 层生成 27×27×96×256 个卷积结果,共需要 7 668 006 912 个时钟周期数;一个 3×3 卷积运算花费 250 个时钟周期,C3 层生成 13×13×256×384 个卷积结果,C4 层生成 13×13×384×384 个卷积结果,C5 层生成 13×13×384×256 个卷积结果,所以 C3 层共消耗 4 153 344 000 个时钟周期数,C4 层共消耗 6 230 016 000 个时钟周期数,C5 层总共消耗 4 153 344 000个时钟周期数.

文献[5][6]都是基于 FPGA 实现,它们的处理速度比我们的方法快很多,但是它们处理的数据少很多.我们第一层输入 3 张特征图,输出 96 张特征图,第二层输入 96 张特征图,输出 256 张特征图,而文献[5][6]第一层输入 7 张特征图,输出 64 张特征图,第二层输入 8 张特征图,输出 19 张特征图等等.

表 2 AlexNet 网络计算时间

层类型	文献[5] (cycles)	文献[6] (cycles)	本文 (cycles)
C1	36 000	366 025	483 516 000
C2	255 000	237 185	2 556 002 304
C3	169 000	160 263	1 384 448 000
C4	128 000	120 198	2 076 672 000
C5	85 000	80 132	1 384 448 000
总和	2 006 000	963 804	7 885 086 304

表 3 是 FPGA 芯片资源使用情况.表 4 是不同实现方式的性能对比.文献[7]是通过 ARM Mali-T628GPU 优化 CNN 代码得到的结果.比较可知,我们的方法是文献[7]的 23.5 倍.

表 3 FPGA 逻辑资源利用统计表

逻辑器件	资源利用率
Slice Registers	15287/948480(1%)
Slice LUTs	52150/474240(10%)
LUT-FF pairs	14354/53083(27%)

表 4 性能对比

设备	OP/S[GOP/S]
文献[7](GPU)	0.02
本文(FPGA)	0.47
性能提升	23.5×

5 结束语

针对计算密集型与存储复杂型的卷积神经网络算法,提出了一种在资源有限的 FPGA 中并行实现

前向传播过程的方法.首先对 CNN 每个类型层的计算复杂度和存储占用进行分析,然后利用卷积层的并行性提出了一种基于 FPGA 的并行结构,从而提高卷积的计算速度.实验结果表明,在 110 MHz 的工作频率下,该结构可使 FPGA 的峰值运算速度达到 0.48 GOP/s,与 ARM Mali-T628 GPU 平台相比实现了 23.5 倍的加速比.

参考文献:

[1] Li B, Zhou E, Huang B, et al. Large scale recurrent neural network on GPU[C]// International Joint Conference on Neural Networks. IEEE, 2014:4062-4069.

[2] Li X, Zhang G, Huang H H, et al. Performance Analysis of GPU-Based Convolutional Neural Networks [C]// International Conference on Parallel Processing. IEEE, 2016:67-76.

[3] 李施豪,应三丛.基于 FPGA 的卷积神经网络浮点激励函数实现[J]. 微电子学与计算机, 2017, 34(10): 105-109.

[4] Liu Z, Dou Y, Jiang J, et al. Throughput-Optimized FPGA Accelerator for Deep Convolutional Neural Networks[J]. Acm Transactions on Reconfigurable Technology & Systems, 2017, 10(3):17.

[5] Shen Y, Ferdman M, Milder P. Overcoming resource underutilization in spatial CNN accelerators[C]// International Conference on Field Programmable Logic and Applications. IEEE, 2016:1-4.

[6] Zhang C, Li P, Sun G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks[C]// Acm/sigda International Symposium on Field-Programmable Gate Arrays. ACM, 2015: 161-170.

[7] Fursin G, Fursin G. Optimizing convolutional neural networks on embedded platforms with OpenCL[C]// International Workshop on Opencl. ACM, 2016:10.

作者简介:

蒋林 男,(1970-),博士,教授.研究方向为集成电路设计. E-mail:jl@xupt.edu.cn.

王喜娟 女,(1992-),硕士研究生.研究方向为电路与系统.

刘镇弢 男,(1971-),博士,讲师.研究方向为集成电路设计.

谢晓燕 女,(1972-),硕士,教授.研究方向为计算机系统结构.

衡茜 女,(1994-),硕士研究生.研究方向为集成电路工程.