基于Rocket-Chip开源处理器的CNN加速模块的设计及实现

杨维科，贺光辉，景乃锋
（上海交通大学 电子信息与电气工程学院，上海 200240）

摘 要：基于RISC-V开源指令集及Rocket-Chip开源处理器，提出了一种基于Eyeriss结构的卷积神经网络加速模块，并与处理器连接形成完整的系统. 该加速器结构通过进行横向（卷积核权值），纵向（输出计算结果）以及斜向（输入图像）的数据重用，大大减少了卷积层的时间消耗. 此外，利用加州大学伯克利分校开发的Chisel3语言，一方面生成该系统的C语言模拟器进行调试，另一方面生成Verilog进行综合和布局布线. 通过对LeNet-5手写数字识别网络及MNIST数据集进行测试，准确度、速度等都达到了令人满意的结果.

关键词：开源处理器；RISC-V；卷积神经网络；Eyeriss
中图分类号：TP39 **文献标识码**：A **文章编号**：1000－7180(2018)04－0017－05
DOI:10.19304/j.cnki.issn1000-7180.2018.04.004

Design and Implementation of CNN Acceleration Module Based on Rocket-Chip Open Source Processor

YANG Wei-ke, HE Guang-hui, JING Nai-feng
(School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: Based on the RISC-V open source instruction set and Rocket-Chip open source processor, we propose a convolution neural network acceleration module based on Eyeriss structure and form a complete system with the processor connection. The accelerator structure greatly reduces the time consumption of the convolution layer by performing data reuse in the lateral (convolution core weight), longitudinal (output calculation result) and oblique (input image). In addition, the use of the UCB developed Chisel3 language, on the one hand to generate the system C language simulator for debugging, on the other hand generate Verilog for integrated and layout and routing. Through the LeNet-5 handwritten digital identification network and MNIST data set, we achieved satisfactory results on speed and accuracy.

Key words: open source processor; RISC-V; convolutional neural network; Eyeriss

1 引言

从2010年开始UCB的Krste等人开始着手研究RISC-V指令集[1]，2015年左右，最终开发出了一套完整的新指令集，还包括相对应的编译器、仿真器和工具链，甚至专门开发了一种名为Chisel的新的硬件构建语言.

RISC-V发展至今短短两年多的时间，已经吸引了学术界和工业界的大量关注. UCB自己已经建立一个名为Rocket-Chip Generator的开源项目[2]，采用基于scalar的Chisel语言进行编写. 项目采用了参数化的配置方法，因此便于调试以及产生不同性能要求的SoC.

卷积神经网络（CNN）现在是许多学科研究的热点之一，被广泛用于多种领域，特别是在模式识别、图像处理、计算机视觉等方面. 已经成功应用的例子有人脸识别、手写数字/字母检测、自动驾驶等. 但是卷积神经网络的主要问题在于计算量太大，特别是其中的卷积层，以Alex-Net为例，占用了90%以上的计算量[3]，因此，卷积神经网络的加速器也是这几年的热门方向. 由于卷积神经网络自身的特点，层与层之间可以理解为是顺序执行，而层内则有着

收稿日期：2017－07－15；修回日期：2017－05－22

?1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

较高的并行性，所以提高层内计算的并行度成为了加速卷积神经网络的重中之重.

本文首先简单介绍 Rocket-Chip，然后以 LeNet-5 手写数字识别网络[4]为实际例子简单地介绍卷积神经网络 CNN 的基本结构，接着根据 CNN 的特点设计了高效的硬件加速方案，并且利用开源硬件 Rocket-Chip 的优点，同时生成对应的 C 代码和 Verilog 代码，且对其性能进行评估测试，并对进一步研究工作进行了展望.

2 背景

本部分简单介绍 Rocket-Chip 开源处理器，CNN 基本结构以及常用 CNN 加速方式.

2.1 Rocket-Chip 介绍

Rocket-Chip 是一种 64 位、5 级流水线、单发射顺序执行的处理器，由 UCB 设计. 它支持 MMU，支持分页虚拟内存；具有分支预测功能，还有兼容 IEEE754-2008 标准的浮点运算单元[2].

它是采用 UCB 基于 Scala 语言设计的 Chisel (Construsting Hardware in an Scala Embedded Language)语言编写的，弥补了传统硬件编程语言的缺点，引入了面向对象、函数式编程、类型推断等概念. Rocket-Chip 的 Chisel 代码全部开源，而且更为重要的是，使用 Chisel 编写的硬件电路，一方面可以通过编译生成 verilog 文件，进而生成 vivado 工程，另一方面可以生成对应的 C++模拟器，大大方便了硬件的开发. 同时，由于其面向对象的特性，Rocket-Chip 实现了高度的可定制化，通过调整参数就能生成具有不同参数的处理器.

为了与 Rocket-Chip 统一，本文的 CNN 加速器也是由 Chisel 语言编写，并且在软件模拟和硬件实现上都进行了验证.

2.2 CNN 基本结构

对于 LeNet-5 卷积神经网络来说，总共有 8 层，包括：

Input 输入层，32×32

C_1 卷积层 6 kernels, each with size 5×5, stride = 1

S_2 降采样层 each with size 2×2, stride = 2

C_3 卷积层 6×16 = 96 kernels, each with size 5×5, stride = 1

S_4 降采样层 each with size 2×2, stride = 2

C_5 卷积层 16×120 = 1 920 kernels, each with size 5×5, stride = 1

F_6 全连接层

Output 输出层

其中，kernel 是卷积核，在 LeNet-5 中都采用大小为 5×5 的卷积核对特征图像进行卷积操作；stride 表示卷积核在特征图像上进行卷积时的平移间隔. 卷积层占用了 90%以上的计算量[3]，因此接下来介绍一下具体卷积层的计算方式.

如图 1 所示，输入图像为 P 张 $M\times N$ 的特征图，输出图像为 Z 张 $X\times Y$ 的特征图，卷积核有 Z 组，每组 P 个 $R\times S$ 卷积核. 在实际计算的过程中，每一组 P 个卷积核分别与 P 张输入图像进行卷积，然后把结果求和，生成一张输出特征图. 把这个过程用伪代码来表示的话如图 2 所示.

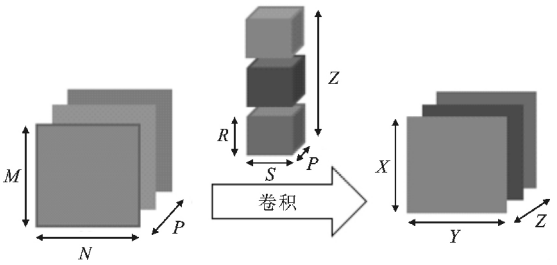


图 1 一个简单卷积层的示意图

```
For z=1:Z
 For x=1:X
  For y=1:Y
   For r=1:R
    For s=1:S
     For p=1:P
      Out[x][y][z]+=Input[r+xxt][s+yxt][p]xkernel[r][s][p][z];
```

图 2 原始的卷积层计算的伪代码

2.3 常用 CNN 加速方式简介

由图 2 可知，CNN 网络中的卷积计算是由一连串互相嵌套的 for 循环完成的，因此宏观上主要的加速思路有两种：1，改变循环嵌套的先后顺序，提高数据的重用率，减少对外部存储的访问；2，将一些低层循环进行 Unroll 操作展开，主要利用的是低层循环中数据互相不依赖，有进行并行计算的可能[5].

如图 3，就是一种进行循环展开的硬件加速方式. 图 3 的例子中，输入图像为 4 张，输出为 2 张，即 $P=4$，$Z=2$，因此总共有 $P\times Z=8$ 个卷积核. 如果按照图 2 的方法顺序执行的话效率不高，而且这 8 次卷积互相没有依赖关系，因此下图的设计中采用了 8 个 buffer 存储 8 个卷积核，使用 8 个 Multiplier 同时进行卷积操作，得出来的结果利用加法树的形式相加.

而进一步的加速，需要从卷积器，即上图的 Multiplier 上进行. 对于每一次实际的卷积，通俗地

?1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

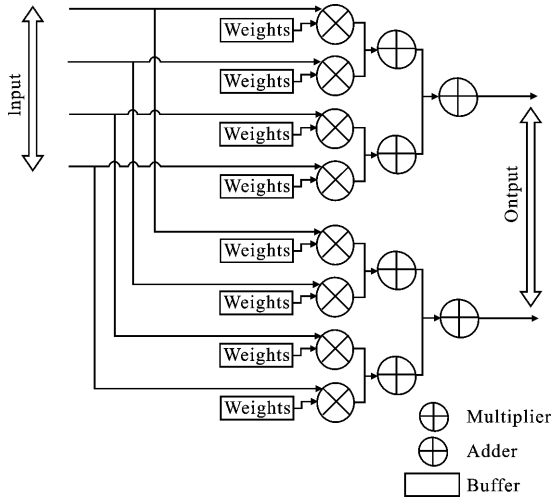


图 3　一种 CNN 卷积加速器结构

讲就是一个框在输入图像上进行移动，每次移动都算出卷积核和图像的点积做为结果. 如果不加优化的话会造成频繁的访问外部存储，降低计算速度. 本文据此，基于 Eyeriss 结构提出了一种 CNN 的卷积加速结构.

3　基于 Eyeriss 结构的 CNN 加速器

Eyeriss 是在 2016 年发表在 ISCA 上的一种卷积神经网络加速结构[6]，通过横向，竖向以及斜向的数据重用来减少访存次数. 在本部分，我们首先介绍 Eyeriss 结构的基础：一维卷积器，然后介绍是如何构建 PE 阵列进行数据重用和加速，最后介绍在提高硬件通用性上做的一些工作.

3.1　一维的卷积器

如图 4 所示，举一个 3×3 卷积核的例子，一般来说，CNN 的卷积核是二维的，那么当卷积核移动到下面的框的位置的时候，两框之间重叠部分的数据需要重新载入造成延迟，那么为了提高数据的重用性，可以采用以行做为单位的一维卷积器.

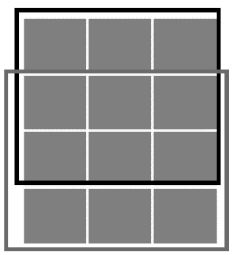


图 4　二维卷积器示例

一维卷积器如图 5 所示，随着时间推移卷积核在输入图像上单方向地向右移动.

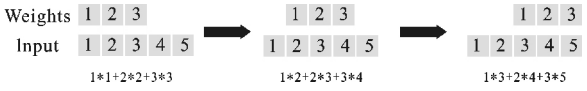
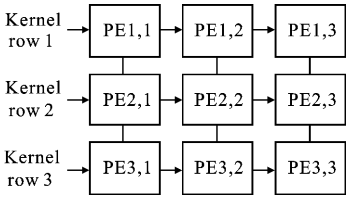


图 5　一维卷积器示例

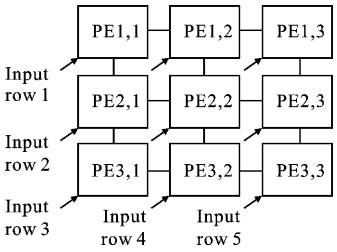
将二维的卷积操作降维后，通过建立 PE 阵列进行数据重用.

3.2　PE 阵列以及数据重用

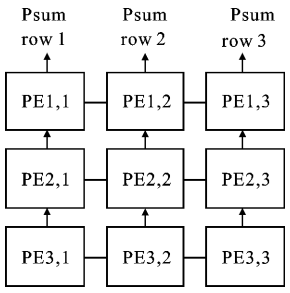
如图 6 所示，是一个 3×3 的 PE 阵列，卷积核权值的行通过横向进行复用(a)，输入特征图像的行通过斜向进行复用(b)，计算出来的部分和通过竖向进行复用(c). 通过这些复用可以减少访问 DRAM 的次数. 在图 6 的例子中，卷积核的尺寸是 3×3，输入图像的行数为 5，输出图像的行数为 3. 所有复用同一个值的 PE 单元都在同一个周期得到该值；部分和在计算出来的时候立即送往相邻的 PE 单元.



(a)



(b)



(c)

图 6　PE 阵列三种数据重用方式示例

我们假设输入图像尺寸为 5×5，那么输出图像尺寸是 3×3. 同时，乘法和加法计算都认为各消耗一个周期. 不进行加速的情况下，输出结果每个点都需要 9 个乘法和 8 个加法，因此计算出结果共需要

?1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

周期数为：9×(9+8)=153

而加速后每 3 个周期可以算出 3 个结果，总共需要 9 个周期，加速比为 17. 同理，如果卷积核大小为 5×5，那么加速比为 49. 在理想情况下，要得到一个 $N\times N$ 的输出图像，需要时间为 $N\times N$，即每个周期可以算出一个结果[7].

对于本文实际使用的 LeNet-5 网络来说，最大的情况下输入图像为 32×32，卷积核大小为 5×5，因此理论上最大需要 PE 阵列尺寸为 5×28.

3.3 对于非理想情况的考虑

在实际电路中，可能并不能提供足够的物理 PE，或者 PE 阵列尺寸与实际需要的尺寸不符，这时候就需要进行物理 PE 与虚拟 PE 的映射，以及将完整的计算拆分成多个部分.

例如，对于一个 12×14 的阵列，实际需要的是 5×28 的阵列，那么一种解决方法是将 5×28 的阵列从中间截开，两段就都能放进 12×14 的阵列中，尽可能充分地利用了电路资源. 需要注意的是对于交界处斜向的输入数据的重用.

3.4 完整的系统结构

如图 7 所示，是本文搭建的 Rocket-Chip 与加速器交互的示意图(忽略外部存储). 首先 CPU 向 buffer 中写入数据，然后发指令让加速器从 buffer 中读取所需的数据进行计算. 加速器计算完成后将结果写回 buffer，CPU 再从 buffer 中把计算结果读回.

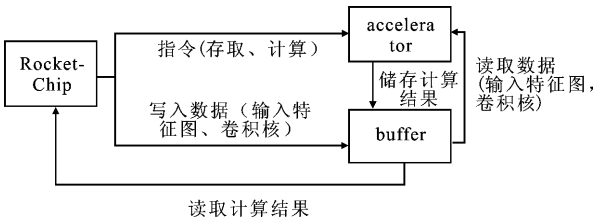


图 7 CNN 加速器与 CPU 的结构

4 仿真及验证结果

本部分，先是对 LeNet-5 的参数进行定点化，然后通过 Chisel 产生的 C++模拟器测试该 Rocket-Chip+加速器的功能以及性能，最后通过 Vivado 综合 Chisel 产生的 Verilog 文件，评估 Rocket-Chip+加速器的时序及资源开销.

4.1 LeNet-5 定点化

在 Visual Studio 2015 软件环境下，对原始的 LeNet-5 进行测试，针对 MNIST 数据集正确率为 96.66%. 而经过定点后，如图 8 所示，横坐标为小数点后二进制位的个数，也就是说位数越高，精度越高. 可以看出，位数在 6 以上时几乎没有精度损失，在 5 及以下则正确率迅速降低. 因此采用小数点后 6 位表示. 在这种情况下，正确率位 96.43%.

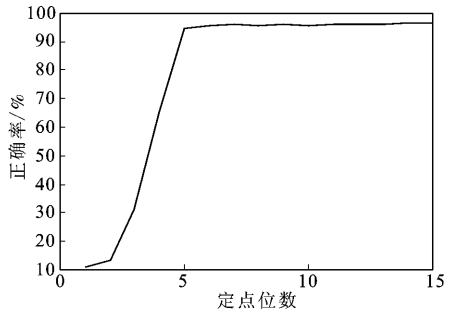


图 8 定点位数与正确率的关系

4.2 C++模拟器仿真

在对 LeNet-5 网络进行定点化之后，搭建好 Rocket-Chip+加速器的系统，通过 Chisel 生成的 C++模拟器进行功能验证和性能测试. 表 1 列出了未加速和加速后的一幅图像 Forward 过程中各个卷积层的周期数.

表 1 卷积层加速前后计算周期数对比

	卷积层 1	卷积层 2	卷积层 3
未加速	136 152	267 094	64 671
加速后	16 983	24 786	71 826
改进	12 356	12 077	38 624

对于卷积层 1，加速比达到 8.01，对于卷积层 2，加速比达到 10.77，而对于卷积层 3，反而消耗了更多的周期.

据前文所述，卷积层 1 输入 1 幅 32×32 的图像，有 6 个 5×5 的卷积核，输出 6 幅 28×28 的图像；卷积层 2 输入 6 幅 14×14 的图像，经过 96 个 5×5 的卷积核，生成 16 幅 10×10 的图像；卷积层 3 输入 16 幅 5×5 的图像，经过 1 920 个 5×5 的卷积核，生成 120 幅 1×1 的图像. 由于卷积层 3 的图像输入已经很小，所以需要进行访存的次数过多，也就是说主要时间消耗在搬运数据的过程中.

为此，为了提高输入图像的重复利用率，将图 2 中循环的次序进行调整，将 P 也就是输入图像放在最外层，尽量长时间地使用输入图像. 改进后的结果也如表 1 所示.

最终，经过测试处理一张图像总共需要的周期数平均为 49 万周期左右，而加速改进后需要 7.5 万周期左右，加速比为 6.53.

4.3 Vivado 验证及分析

本文采用 Vivado2016.2 软件，选用 Xilinx Kintex-7 开发板为目标环境.

?1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

由于 Rocket-Chip 本身是面向 ASIC 电路的，因此生成 Vivado 的 FPGA 工程的时候时钟频率并不高. 根据 UCB 的流片结果，ASIC 的 Rocket-Chip 的频率是大于 200 MHz 的，所以我们在 vivado 上分析的更多是针对加速器进行.

通过综合及实现，将提供的 PLL 时钟频率设为 100 MHz 时，查看时序报告可得，仍有 2.404 ns 的余量，因此加速器实际最高频率为 131.65 MHz.

相应的，电路的资源消耗见表 2

表 2　FPGA 上资源消耗表

LUTs	109 875
FFs	84364
DSPs	429
BRAMs	23

由于 FPGA 上的 Verilog 代码是由 Chisel 直接编译生成，因此消耗了大量的 LUTs. 接下来改进的展望是使用乘法器 IP，通过使用 DSP 来降低 LUT 的占用.

5　结束语

本文中，我们基于 Rocket-Chip 开源处理器提出了一种卷积神经网络的加速结构. 该结构通过构建一维卷积器的 PE 阵列，从而在横向，竖向和斜向上分别对卷积核，计算结果以及输入特征图像进行复用，从而减少访存的次数. 更进一步，优化了卷积层的计算顺序，尽可能使得输入特征图的利用率达到最高. 并且，通过 Chisel 语言编写的加速器在与 Rocket-Chip 结合后，生成的 C++模拟器上的仿真也证明了该结构的加速作用. 同时，该加速器的 FPGA 也证明了在资源消耗在合理的范围内，能获得良好的工作频率.

参考文献：

[1] Waterman A, Lee Y, Patterson D A, et al. The risc-v instruction set manual, volume i: Base user-level isa [J]. Eecs Department, 2011, 7(9):475

[2] Asanovic K, Avizienis R, Bachrach J, et al. The rocket chip generator[R]. EECS Department, University of California, Berkeley, 2016.

[3] Chen Y H, Krishna T, Emer J S, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.

[4] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[5] 陆志坚. 基于 FPGA 的卷积神经网络并行结构研究[D]. 哈尔滨：哈尔滨工程大学, 2013.

[6] Chen Y H, Emer J, Sze V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks[C]//Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on. Seoul, South Korea, IEEE, 2016: 367-379.

[7] 方睿，刘加贺，薛志辉，等. 卷积神经网络的 FPGA 并行加速方案设计[J]. 计算机工程与应用, 2015, 51(8):32-36.

作者简介：

杨维科　男，(1993-)，硕士研究生. 研究方向为可重构处理器的系统架构. E-mail: wakefieldyoung@sjtu.edu.cn.

贺光辉　男，(1980-)，博士，副研究员. 研究方向为通信系统、VLSI 等.

景乃锋　男，(1982-)，博士，副研究员. 研究方向为计算机体系结构.

(上接第 16 页)

[5] Salama M, Kandil M. An Improved DV-hop localization algorithm based on modified hop-size[C]// Computer Applications & Research. Qingdao, China, IEEE, 2016:83-86.

[6] Mehrabi M, Taheri H, Taghdiri P. An improved DV-Hop localization algorithm based on evolutionary algorithms [J]. Telecommunication Systems, 2017, 64(4) :1-9.

[7] 肖晓丽，李旦江，谭柳斌. 基于布谷鸟搜索算法的无线传感器网络节点定位[J]. 计算机工程与应用, 2017, 53(2):141-145.

[8] 邱奉美，李怀忠. 无线传感器网络 DV-Hop 定位算法的改进[J]. 计算机工程, 2014, 40(8):15-20.

[9] 李长庚，孙克辉. 基于加权最小平方法的 DV-Hop 改进算法[J]. 微电子学与计算机, 2016, 33(1):24-27.

[10] 李云飞，江明，葛愿，等. 基于曲线拟合的改进 DV-Hop 定位算法[J]. 计算机系统应用, 2015, 24(5): 118-123.

作者简介：

马肖旭　女，(1990-)，硕士研究生. 研究方向为计算机网络与安全. E-mail: 1656871571@qq.com.

刘文菊　女，(1963-)，教授. 研究方向为计算机网络与安全、企业信息化.

王　赜　男，(1976-)，教授. 研究方向为计算机网络与安全、企业信息化.

?1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net