

引用格式:杜高明,陈邦溢,王晓蕾,等.基于分时重用行卷积查找表的 BNN 加速器[J].微电子学与计算机,2021,38(9):84-92. [DU G M, CHEN B Y, WANG X L, et al. A BNN accelerator based on time-sharing reuse row convolution LUT[J]. Microelectronics & Computer, 2021, 38(9): 84-92.]

基于分时重用行卷积查找表的 BNN 加速器

杜高明, 陈邦溢, 王晓蕾, 李桢旻

(合肥工业大学 微电子设计研究所, 安徽 合肥 230601)

摘 要: 二值化神经网络(Binary Neural Network, BNN)具有单比特数据位宽的特点,可以很好地解决传统卷积神经网络中存在大量数据量以及计算量的问题.为了进一步加速 BNN 的正向推导并降低所需功耗,提出一种基于 FPGA 的完全二值化卷积神经网络加速器,其中输入图片以及边缘填充都进行了二值化处理,并且通过分时重用行卷积查找表的方式跳过其中的冗余计算.在 Xilinx 的 ZCU102 FPGA 开发板上对所设计的加速器进行评估,结果表明加速器的运算速度可以达到 3.1 TOP/s,并且可以达到 144.2 GOPS/KLUT 的资源效率转换比以及 3 507.8 GOPS/W 的能效转换比.

关键词: BNN;完全二值化;行卷积查找表;分时重用;FPGA

中图分类号: TN47

文献标识码: A

文章编号: 1000-7180(2021)09-0084-09

A BNN accelerator based on time-sharing reuse row convolution LUT

DU Gaoming, CHEN Bangyi, WANG Xiaolei, LI Zhenmin

(Institute of VLSI Design, Hefei University of Technology, Hefei 230601, China)

Abstract: The single-bit data width characteristic of Binary Neural Network (BNN) can tackle large-scale-data and huge-amount-calculation in Convolution Neural Network (CNN). In order to further accelerate the forward inference of BNN and reduce the required power consumption, a fully binarized neural network accelerator based on FPGA is proposed, in which the input image and edge padding are all binarized. And the accelerator skips the redundant calculations by reusing the Row Convolution LUT (RC-LUT) in a time-sharing way. By implementing on Xilinx's ZCU102 FPGA, this accelerator can achieve a Performance of more than 3.1 TOP/s, an Area Efficiency of 144.2 GOPS/KLUT, and a Power Efficiency of 3 507.8 GOPS/W.

Key words: BNN; fully binarized; row convolution LUT; time-sharing reuse; FPGA

1 引言

近年来卷积神经网络(Convolutional Neural Networks, CNNs)在目标识别、语音识别、模式识别、自动驾驶以及医学等领域中的应用变得越来越广泛,并且在自动驾驶和医学诊断这些比较关键的应用领域中,神经网络的识别精度显得尤为

重要.而更高的精度需要更加复杂的网络模型,网络的数据量以及计算量也会跟着上升,这也逐渐成了掣肘神经网络发展的主要因素.例如,在比较经典的神经网络模型中, AlexNet^[1]需要大约 60M 的参数量以及 720 M 的浮点运算来处理一张图片,而 VGG-16^[2]需要 131.9M 的参数量以及 14.4 G 的浮点运算.

研究表明,神经网络存在很大的冗余性.为此,Courbariaux 首先在文献[3]中提出对神经网络中的权重数据进行量化得到二值化权重网络(Binary Weight Network, BWN);随后又在文献[4]中提出同时将特征图和权重数据量化得到二值化神经网络(Binary Neural Networks, BNNs). BNN 中 1 bit 的数据可以大幅减少数据存储量和计算量.

为了提升 BNN 的性能,很多工作对 BNN 的训练过程进行了优化.文献通过对网络中的归一化层和池化层剪枝来跳过部分不必要的冗余计算,但在卷积层中仍存在大量冗余计算.由于直接二值化处理输入图片会使精度损失严重,目前大多数工作都未对输入图片做二值化处理,因此需要单独为第一层设计全精度的计算单元,而全精度计算单元所需的计算资源是二值化计算单元的 9~10 倍.文献[5]将全精度输入图片以数值累加的方式拆分为多通道的单 bit 输入图片,但是这种处理方式使第一层的输入通道变为原来的几百倍.除此之外,文献[5]还提出一种±1 填充的方式来避免 0 值填充带来的数据三值化问题(±1、0),但是这种±1 填充的训练效果没有 0 值填充的训练效果理想.

针对上述提到的几个问题,本文提出一种基于硬件复用的可重构 BNN 硬件加速器,本文主要工作如下:

(1)提出一种完全二值化神经网络(Fully Binarized Neural Network, FBNN)处理和训练方案,将训练集和测试集的输入图片都按 bit 进行拆分得到一个 24 通道的二值化数据集,并利用此数据集训练得到一个完全二值化的卷积神经网络,避免单独为第一层设计全精度的计算单元;

(2)提出一种奇偶通道±w(其中w表示权重)交叉填充的边缘填充方式,训练过程保持 0 值填充,推导过程使用奇偶通道±w交替填充,使正向推导过程可以达到与训练过程中 0 值填充相同的效果,在保证精度的前提下不引入第三个数值 0;

(3)提出一种分时重用的行卷积查找表(Time-sharing Reuse Row Convolution LUT, TRRC-LUT)来跳过卷积层中的冗余计算;预计算制表与查表计算进行乒乓操作,隐藏预处理周期,提升处理速度;并在上下行卷积计算过程中重用行卷积查找表(Row Convolution LUT, RC-LUT),来减少预处理单元的计算量.

2 卷积神经网络基础

2.1 BNN 基础

表 1 给出了本文所用到的参数信息,图 1 是 CNN 和 BNN 的架构示意图,两者都包括 CONV/FC 层,激活(Activation)层和池化(Pool)层, BNN 增加了批量归一化层(Batch Normalization, BN),并且其中的激活层为二值化激活层(Binalize, BIN).

表 1 网络中的参数信息

(N_{in}, N_{out})	层输入输出通道数
(H, W)	特征图长度和宽度
R_{out}	输出特征图尺寸
K, S	卷积核尺寸,卷积核滑动步长
K_p	池化尺寸,本文 $K_p=2$
OP_K, OP_R, OP_C	OP_K 和 OP_R 表示一个卷积窗口和一行卷积所需的操作数, OP_C 表示将一个卷积窗口中所有行卷积结果累加所需的操作数.

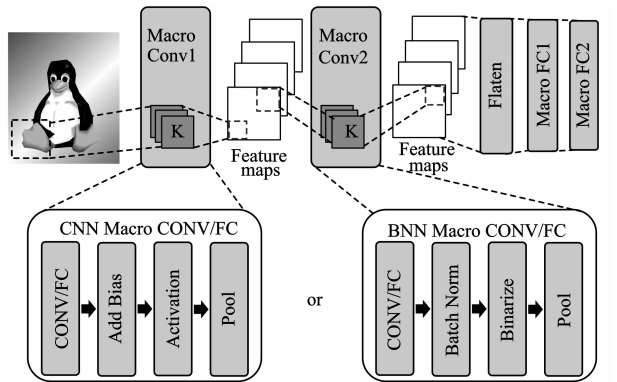


图 1 典型 CNN/BNN 模型架构

CONV/FC 层:卷积层中,通过一组 N_{in} 个 $K \times K$ 的卷积核在 N_{in} 张输入特征图(Input Feature maps, Ifmaps)上以 S 的步长进行滑动,计算得到一张输出特征图(Output Feature maps, Ofmaps).通过 N_{out} 组权重数据计算得到 N_{out} 张输出特征图,公式(1)是第 m 张输出特征图中第 p 行,第 q 列的计算过程.因为硬件设计中,将特征图以及权重中的一 1 转换为 0,因此可以用 XNOR 和 PopCount^[6]来进行乘累加计算. FC 层可以看成输入输出特征图以及卷积核的尺寸都是 1×1 的特殊的卷积层.

$$X_{(p,q)}^m = \sum_{n=1}^{N_{in}} \sum_{i=1}^K \sum_{j=1}^K \text{xnor}(I_{p+i,q+j}^n, W_{(i,j)}^{(n,m)}) \quad (1)$$

BN 层:BN 的计算过程如公式(2)所示,其中的参数 μ, σ, γ 和 β 由训练得到.

$$x^{bn} = \left(\frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \right) \times \gamma + \beta \quad (2)$$

BIN 层: 二值化激活函数如公式(3) 所示:

$$y^b = \text{sign}(x^{bn}) = \begin{cases} +1, x^{bn} \geq 0 \\ -1, x^{bn} < 0 \end{cases} \quad (3)$$

将批量归一化与二值化激活函数结合得到 $y^b = \text{sign}(BN(x))$, 因为训练得到的 $\gamma \geq 0$, 因此将公式进行整合得到公式(4):

$$y = \begin{cases} +1, x \geq \mu - \frac{\beta}{\gamma} \sqrt{\sigma^2 + \epsilon} \\ -1, x < \mu - \frac{\beta}{\gamma} \sqrt{\sigma^2 + \epsilon} \end{cases} \quad (4)$$

Pool(池化)层: 本文所采用的池化方式是最大值池化(Max Pooling), 其函数表达式如公式(5):

$$P^m(p, q) = \text{or}[I_{Kp \times Kp}^m(p, q)] \quad (5)$$

2.2 相关工作

自从 Courbariaux 在文献[4]中提出 BNN 以来, 很多工作基于 FPGA 为 BNN 设计硬件加速器, 并且都取得了非常好的性能. FINN^[6] 首先使用 FPGA 为 BNN 设计硬件加速器, 并且利用数据单比特的特点对各层进行优化, 高效利用 FPGA 中的各种资源; FP-BNN^[7] 提出一种压缩树的方式来对卷积计算中的累加进行优化, 极大的提高了资源利用率; LP-BNN^[8] 采用流水线的设计方案, 大大提高了 BNN 的推理速度. FBNA^[5] 对输入图片进行二值化处理, 并且用交叉 ± 1 填充的方式消除了 0 值边缘填充带来的数值三值化问题.

L-MPC^[9] 提出一种对 BWN 中的冗余计算的优化方案, 但是 L-MPC^[9] 中没有对行卷积查找表进行重用. 并且这种方案没有通过乒乓操作将预计算制表和查表加速计算这两个过程分开, 没有充分利用计算资源以及计算周期.

2.3 研究动因

BNN 正向推导过程中存在的冗余计算, 并采用查找表的方式来跳过这些冗余计算, 以减少计算周期和降低加速器功耗. 如图 2(a)所示, H 和 W 分别表示特征图的长和宽(输入特征图的长和宽分别加 2 是因为添加了 Padding). 对于每一张输入特征图数据, 都会有 N_{out} 个卷积核来与其进行卷积计算, 得到 N_{out} 张输出特征图的部分卷积结果(Part sum, Psum). 其中每一行的输入数据都与 N_{out} 个 1×3 的行权重数据进行卷积计算.

如图 2(b)所示, 所选取的第一行输入 Ifm_L0 与 N_{out} 个尺寸为 1×3 的行权重数据进行卷积计算.

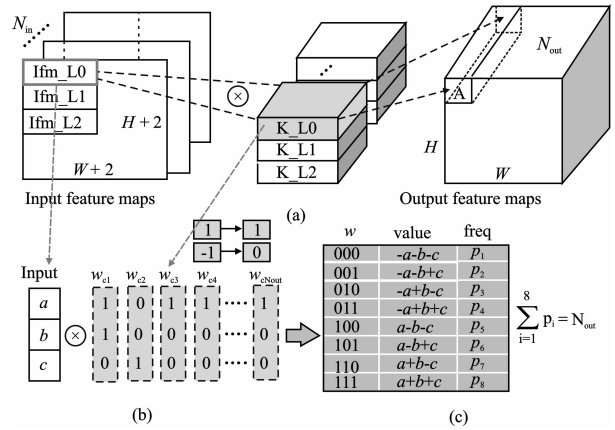


图 2 网络计算冗余性

这 N_{out} 个行权重数据因为二值化的原因, 只有 000 ~ 111 这 8 种可能情况. 卷积神经网络中, N_{out} 数值都很大(如表 2 所示), 因此这 8 个权重都存在很高的重复率, 带来很多的冗余计算, 对此本文提出 TRRC-LUT 来跳过这些重复的冗余计算. 先将每一行的输入都与这 8 种可能的权重进行计算(本文中一行并行取 1×6 的输入, 与 1×3 的权重计算得到 1×4 个 2 bits 的行卷积 Psum), 并将结果制作成 RC-LUT, 再将读取到的权重作为索引来查找计算结果, 实验结果表明此方案可以减少超过 80% 的计算量. 在实现的过程中对制表和查表两个过程进行乒乓操作, 来减少计算周期.

3 硬件架构设计

3.1 整体硬件架构

图 3 所示是硬件加速器整体架构, 中央控制器(Central Controller)先从配置指令缓存(CBuf)中读取层配置链进行译码, 并根据层读写信号从两个特征图缓存(Fm-Buf)中选取一个作为输入 Fm-Buf, 另一个作为输出 Fm-Buf. 之后根据译码信息从片外存储 DDR 中读取输入和权重数据存入输入 Fm-Buf 和权重缓存(WBuf)中, 特征图传输控制器(FmT-CTR)从输入 Fm-Buf 中读取输入特征图并传给输入复合处理单元(Ifm-Macro PE)进行计算. Psum 传输控制器(Psum-CTR)将 Ifm-Macro PE 的计算结果传给输出复合处理单元(Ofm-Macro PE). Ofm-Macro PE 中, 累加单元(Acc)对输入通道的卷积结果进行累加, 并将累加结果经过激活单元(Act)以及池化单元(Pool)处理得到输出特征图. FmT-CTR 将输出特征图存入输出特征图缓存 Fm-Buf 中. 当该层计算完成之后, 将该层输出

要 $N_{out} \times W \times H \times Q_{psum}$ 的存储空间, 导致存储空间的浪费, 其中 Q_{fmb} 和 Q_{psum} 分别为二值化特征图和乘累加结果的数据位宽。

综上, 为了避免存储 Psum 造成存储资源的浪费, 本文采用如图 6 所示的数据流, 采用 n 组预计算单元阵列 (PrPEA) 并行计算 (本文中 $n=8$), 每个 PrPEA 计算 2×6 的输入数据. 将输入通道分为 n 组, 每个周期读取 n 张输入特征图中 2×6 的输入, 由 PrPEA 计算得到 RC-LUT, 经过 N_{in}/n 个周期计算得到所有输入通道的 RC-LUT. 将 n 组 PrPEA

计算得到的 RC-LUT 存入 n 个 RLBA 中, 每个 RLBA 中有 3 个行卷积查找表缓存 (RC-LUT Buffer, RLB), 每个 RLB 存储 N_{in}/n 张输入特征图对应的 RC-LUT. 3 个 RLB 对制作查找表以及使用查找表进行乒乓操作, 其中一个 RLB 用于存储当前阶段 PrPEA 制作的 RC-LUT, 另外两个 RLB 用于查找卷积计算, 这个过程在下一节中详细介绍. 因为本文采用的网络中卷积层的输入通道数最大为 256 通道, 因此每个 RLB 的深度设置为 32.

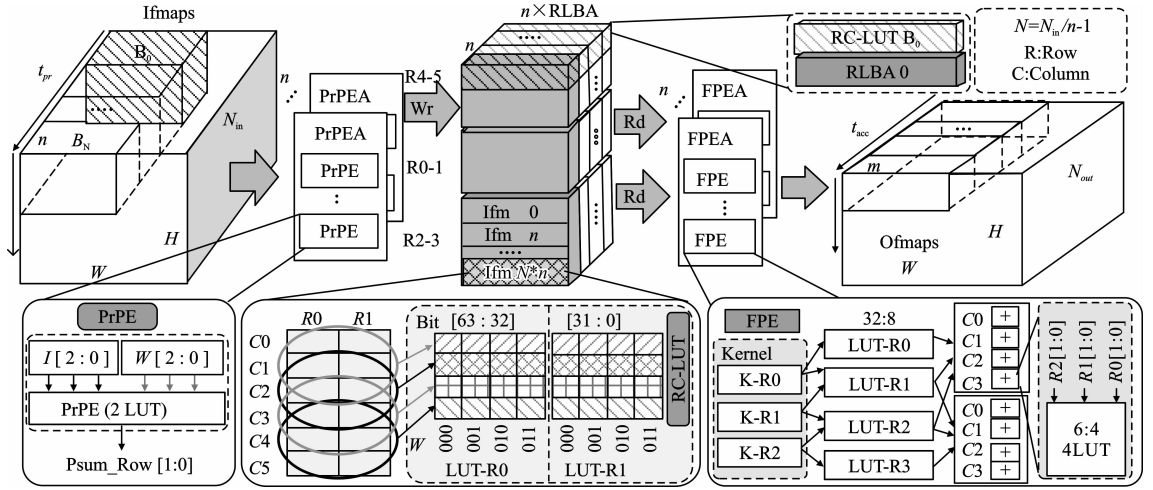


图 6 数据流以及行卷积查找表

如图 6 所示, PrPE 是由 2 个 6 输入 LUT 构成的 $6:2$ 行卷积计算单元, 用于计算 1×3 的输入和权重对应的计算. 快速计算单元 (FPE) 由 4 个深度为 4 宽度为 8 的查找表以及 8 个加法器组成, 以读取到的行权重作为索引进行查表, 加法器由 4 个 6 输入 LUT 构成, 来对三行权重读取到的三行 2 bit 的输出进行相加. 因为 8 个预计算权重中, $3'b000 \sim 3'b011$ 与 $3'b111 \sim 3'b100$ 的计算结果相反, 因此可以只算一组预计算权重, 另一组预计算权重对应的计算结果可以通过取反换算得到, 减少 50% 的 RC-LUT 缓存所需。

3.4.2 行卷积查找表的分时重用

计算两行输出需要预先将对应的 4 行输入特征图制成 RC-LUT, 如图 7 (a) 所示, Cycle0-1 选取的两组 2×6 的输入数据对应计算第 0~1 行的输出数据, Cycle1-2 选取的 4 行输入数据对应计算第 2~3 行输出数据. 计算这两组输出都需要 Cycle1 选取的输入数据, 可以重用这一部分输入对应的 RC-LUT 来减少计算量. 在 Cycle2 中, 快速计算单元阵列 (FPEA) 使用 Cycle0-1 得到的 RC-LUT 来计算第 0

~1 行输出, 同时 PrPEA 计算第 4~5 行输入对应的 RC-LUT; 在 Cycle3 中, 继续使用 Cycle1 得到的 RC-LUT, 与 Cycle2 计算得到的 RC-LUT 组合来计算第 2~3 行的输出, 同时 PrPEA 计算第 6~7 行输入对应的 RC-LUT.

如图 7 (b)(c) 所示, 每个 LUT_BufA 中有 3 个 RC-LUT Buf 子缓存模块, 其中两个 RC-LUT Buf 分别存储前两个 Cycle 计算由 PrPEA 计算得到的 RC-LUT, 这两个 RC-LUT Buf 组成 4×6 的输入对应的 RC-LUT, 并将这两个 RC-LUT Buf 用于 FPEA 的查表计算, 每个周期读取一个地址上的数据作为查找表来查找 m (本文中 $m=16$) 个通道的输出进行累加计算. 经过 N_{in}/n 个周期将查找表缓存中的所有输入特征图查找表遍历完, 并得到完整的 m 个通道的输出并将累加结果传入激活单元中, 并且重新遍历查找表缓存并计算下一组 m 个通道的输出. 同时, n 组 PrPEA 计算下一组 2×6 的输入特征图对应的 RC-LUT 并存入另一个 RC-LUT Buf 中. 在此过程中, 通过一个 3 bit 的循环移位信号控制 3 个 RC-LUT Buf 的读写。

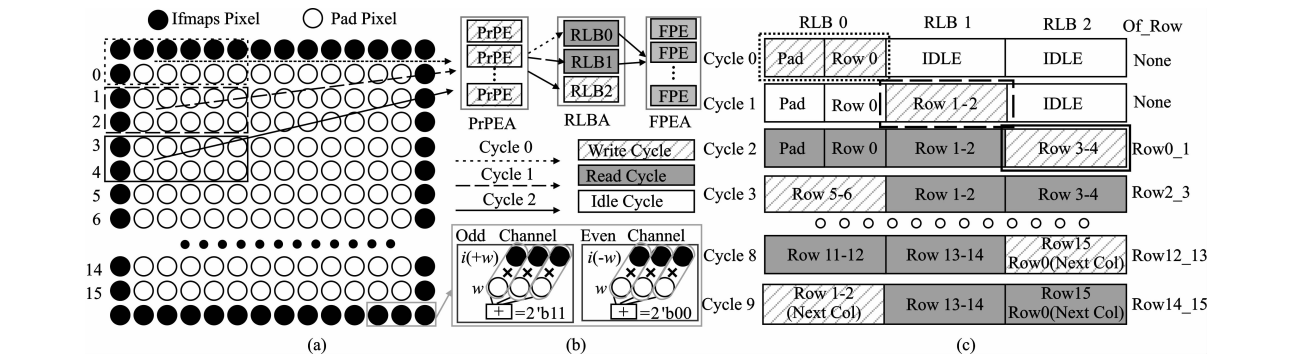


图 7 行查找表的分时重用

4 实验结果对比分析

对本文创新点进行性能评估,对输入图片进行二值化处理使网络精度从86.57%降低到85.26%,但是相比于处理之后得到的完全二值化神经网络的优势,这个误差是可以接受的。

4.1 实验环境

实验基于 Zynq UltraScale+ MPSoC ZCU102 开发板进行试验,使用 Verilog 语言以及 Vivado2017.4 和 Modelsim10.6d 进行硬件开发和测试;基于 Tensorflow1.14 和 Jupyter notebook 搭建软件平台进行神经网络训练. 本文所采用的数据集是 Cifar-10,通过 3.2 节中的图片量化方式得到二值化的 Cifar-10 数据集,其中训练集和测试集中的图片都由 RGB 三通道变为 24 通道,并用这个二值化的数据集来训练网络。

表 2 VGG-Like 神经网络模型

Layer	Structure ^a	(N_{in}, N_{out})	Weights/K	MACs/M
1	C-B-A	(24,128)	27	28.3
2	C-B-A-P	(128,128)	144	151.0
3	C-B-A	(128,128)	144	37.7
4	C-B-A-P	(128,256)	288	75.5
5	C-B-A	(256,256)	576	37.7
6	C-B-A	(256,256)	576	37.7
7	C-B-A-P	(256,256)	576	37.7
8	C-B-A	(256,256)	576	9.4
9	C-B-A	(256,256)	576	9.4
10	C-B-A-P	(256,512)	1 152	18.9
11	F-B-A	(2 048,1 024)	2 048	2
12	F-B-A	(1 024,1 024)	1 024	1
13	F-B-A	(1 024,10)	10	0.01
Total			7 717	446.5

注:F = FC, C = CONV, P = POOL, B = Batch Normalization, A = Activation

如表 2 所示为本文所采用的网络模型中各层的详细参数信息,表中 Weights 和 MACs 分别表示各层权重数据量以及乘法的总数量. 由于输入图片由 RGB 三通道变为 24 通道,因此在训练阶段设置网络模型的超参数时,将第一层卷积层的输入通道数设置为 24 通道,并对此网络进行训练. 而且由表 2 中 Weights 一栏可知,网络模型中第一层的参数量相比于网络整体参数量是可以忽略不计的. 除此之外,卷积层中卷积核的尺寸和滑动步长分别为 3 和 1,池化层的尺寸为 2,步长为 1.

训练过程中数据的值为+1 和-1,原数值按照补码形式需要 2 bits 的数据来表示. 本文所设计的加速器采用表 3 所示的对应关系进行数据转换,将其中的-1 转换为 0,使硬件实现过程中可以使用 1 bit 的数据来进行计算以及存储。

表 3 软硬件数据转换对应关系

Software			Hardware		
x^b	w^b	$\times$	x^b	w^b	XNOR
-1	-1	+1	0	0	1
-1	+1	-1	0	1	0
+1	-1	-1	1	0	0
+1	+1	+1	1	1	1

4.2 奇偶通道±w 交叉填充性能评估

表 4 是奇偶通道±w 交叉填充与现有工作的效果对比,其中 LUT 一栏表示单个计算单元所需的 LUT 开销. 在训练过程中使用 0 值边缘填充,得到的网络中存在-1,+1,0 三种数值,导致硬件实现正向推导的过程中需要 2 bits 的数据来表示. 数据位宽的增加,使硬件设计所需的计算资源以及存储资源也随之增加。

与传统的 0 值填充相比,另外三种数据填充方

式得到的网络中只有-1和+1两种数值,硬件中可以用1 bit数据来表示数值,单个PE所需的LUT相比于0值填充减少了50%,并且累加模块所需的计算资源也随着数据位宽的降低而减少.但是文献[5]和文献[10]中的两种边缘填充方式需要在训练的过程中改变数据填充方式,会使得网络训练精度有所损失.本文所采用的方案在训练过程中仍使用0值填充,只在推导过程中使用奇偶通道±w交叉填充,来达到和训练过程中0值填充相同的效果.

表 4 边缘填充方案效果比较

Pad method	Values	Bit	LUT	Accuracy/%
0	±1,0	2	6	86.57
+1 ^[10]	±1	1	3	85.93
±1 ^[5]	±1	1	3	86.26
±w	±1	1	3	86.57

4.3 分时重用行卷积查找表优化效果

将本文所采用的分时重用行卷积查找表与L-MPC^[9]基于VGG-Like网络进行对比评估.两种方案都分为预计算制作查找表和查表计算两个阶段,各阶段操作数的计算公式如表5所示.

表 5 卷积层操作数计算公式

	阶段名称	操作数(Operations)表达式
L-MPC ^[14]	未优化	全过程 $N_{in} \times N_{out} \times R_{out} \times OP_K$
	预计算	$N_{in} \times (R_{out} \times 3 - 2) \times R_{out} \times OP_R \times 8$
	查表计算	$N_{in} \times N_{out} \times R_{out} \times OP_C$
本文方案	预计算	$N_{in} \times R_{out} \times R_{out} \times OP_R \times 8$
	查表计算	$N_{in} \times N_{out} \times R_{out} \times OP_C$

图8是两种方案的效果对比,图中Operations表示每一层卷积层计算所需的乘累加操作数,Reduce Ratio是两种方案与未优化情况下每一层操作数的比值,本文主要以Reduce Ratio代表两种方案的优化效果.根据图中Reduce Ratio的波动曲线可以看出,输出通道数越大,两种方案的优化效果也越明显.统计整个网络,L-MPC^[9]的Reduce Ratio为21.57%,而TRRC-LUT仅为19.04%,并且因为本文在LUT的制作和使用上使用了乒乓操作,因此在加速器所需计算周期上的优化效果更显著.

4.4 加速器整体性能对比分析

本文所提出的三种优化方案,可以大大提升

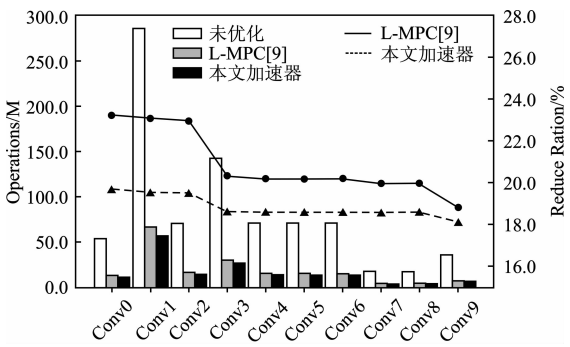


图 8 RC-LUT 方案效果对比

BNN硬件加速器的整体性能.首先,基于位拆分的输入图片二值化处理方案避免了单独为第一层卷积层设计全精度PE;而实验证明,单个二值化PE所需硬件资源开销以及功耗分别为全精度PE的0.104倍以及0.168倍.其次,本文所提出的奇偶通道±w交替填充方案在保持训练精度的前提下,解决了数据三值化问题,避免了BNN硬件加速器中数据位宽的增加,从而减少加速器整体所需的硬件资源开销,并降低系统功耗.最后,本文通过使用RC-LUT跳过卷积层中的冗余计算,并对RC-LUT进行传递重用,实验证明此方案可以减少80.96%的计算量,从而大大降低系统整体延时以及计算功耗.得益于以上三种方案带来的综合优化效果,本文所设计的BNN硬件加速器在使用较少硬件资源的前提下,达到3157GOPs的计算能力,并且系统所需功耗仅为0.9W.

目前很多工作基于FPGA设计了各种BNN硬件加速器,并且取得了良好的整体性能.表6中将本文所设计的加速器与其中几项现有BNN硬件加速器进行了对比.主要采用运算速度Performance、资源效率转换比Area Efficiency以及功耗效率转换比Power Efficiency作为对比指标.其中Area Efficiency和Power Efficiency可以更加公平地对比相同资源或者功耗下,加速器所能达到的运算速度.此外也给出了Frequency、FPGA资源消耗以及Latency这些对比指标来更加全面地进行对比.从表5中可以看出,本加速器的运算速度是文献[11]的接近1.5倍;RC-LUT大大提高了相同并行度下,加速器的计算能力,因此资源效率转换比可以达到144.2GOPS/KLUT,相比于文献[12]提升了2倍多.除此之外,计算量的减少带来功耗上的降低,本加速器所需的功率仅为0.9W,功耗效率转换比能

达到 3507 GOPS/W,相比于文献[12]提升了超过 7 倍.

表 6 BNN 加速器性能对比分析

	文献[11]	文献[5]	文献[12]	文献[10]	本文
Platform	Virtex-7 VXT458t	Xilinx ZC702	Xilinx KC705	Zynq 7Z020	Xilinx ZCU102
Datasets	CIFAR-10/MNIST	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-10
Precision	1-bit fixed	1-bit	1-bit	1-2bit	1-bit
Frequency (MHz)	200	143	100	143	200
LUT(k)	232	29.6	17.068	46.9	21.9
DSP	2 352	/	532	3	0
BRAM	832	103	130	94	182
FF(k)	121	/	28.374	46.1	6.862
Power(W)	28	3.3	2.1	4.7	0.9
Latency(us)	4 600	1 920	833.3	5 940	247.1
Performance (GOPS)	2 100(CONV)	722	1 006	318.9(Conv)/207.8(all)	3 157
Area Efficiency (GOPS/KLUT)	9.05	24	58.94	4.43	144.2
Power Efficiency (GOPS/W)	75	219	479.05	44.2	3 507.8

5 结束语

本文针对 BNN 正向推导过程中存在的大量冗余计算问题,提出利用分时重用行卷积查找表的方式来跳过这些冗余计算,以此来减少 BNN 推理过程中所需的计算量,并且对此过程采用乒乓操作的方式来充分计算周期. 本文还提出一种输入图像二值化处理方式以及奇偶通道± ω 交叉填充方案,并基于这种二值化方案训练得到一个完全二值化卷积神经网络,最后基于分时重用行卷积查找表来为此网络设计硬件加速器,实现结果达到很高的面积效率转换比 (Area Efficiency) 和功耗效率转换比 (Power Efficiency).

参考文献:

[1] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Advances in Neural Information Processing Systems, 2012, 25(2): 1097-1105.

[2] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2015.

[3] COURBARIAUX M, BENGIO Y, DAVID J P. BinaryConnect: training deep neural networks with binary weights during propagations[C]//Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal, Canada: MIT Press,

2015; 3123-3131.

[4] 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.

[5] GUO P, MA H, CHEN R Z, et al. FBNA: a fully binarized neural network accelerator[C]//Proceedings of the 28th International Conference on Field Programmable Logic and Applications. Dublin, Ireland: IEEE, 2018; 51-513. DOI: 10. 1109/FPL. 2018.00016.

[6] UMUROGLU Y, FRASER N J, GAMBARDELLA G, et al. FINN: a framework for fast, scalable binarized neural network inference[C]//Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, California, USA: ACM, 2017; 65-74. DOI: 10.1145/3020078.3021744.

[7] LIANG S, YIN S Y, LIU L B, et al. FP-BNN: binarized neural network on FPGA[J]. Neurocomputing, 2018, 275: 1072-1086. DOI: 10. 1016/j. neucom. 2017. 09. 046.

[8] GENG T, WANG T Q, WU C S, et al. LP-BNN: ultra-low-latency BNN inference with layer parallelism [C]//Proceedings of the 30th International Conference on Application-Specific Systems, Architectures and Processors. New York, USA: IEEE, 2019; 9-16. DOI: 10.1109/ASAP.2019.00-43.

- [9] LIU H, LIU L B, ZHU W P, et al. L-MPC: a LUT based multi-level prediction-correction architecture for accelerating binary-weight hourglass network [C]//Proceedings of the 56th Annual Design Automation Conference 2019. Las Vegas, USA: ACM, 2019: 1-6. DOI: 10.1145/3316781.3317854.
- [10] ZHAO R, SONG W N, ZHANG W T, et al. Accelerating binarized convolutional neural networks with software-programmable FPGAs [C]//Proceedings of 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, California, USA: ACM, 2017: 15-24. DOI: 10.1145/3020078.3021741.
- [11] DUAN Y Z, LI S, ZHANG R P, et al. Energy-efficient architecture for FPGA-based deep convolutional neural networks with binary weights [C]//Proceedings of the 23rd International Conference on Digital Signal Processing. Shanghai, China: IEEE, 2018: 1-5. DOI: 10.1109/ICDSP.2018.8631596.
- [12] WANG J, JIN X, WU W. TB-DNN: a thin binarized deep neural network with high accuracy [C]//Proceedings of the 22nd International Conference on Advanced Communication Technology. Phoenix Park, Korea: IEEE, 2020: 419-424. DOI: 10.23919/ICACT48636.2020.9061291.

作者简介:

杜高明 男, (1977-), 博士, 教授. 研究方向为多核体系结构、片上网络体系结构、多核、NoC 性能分析.

E-mail: dugaoming@hfut.edu.cn

陈邦溢 男, (1996-), 硕士研究生. 研究方向为卷积神经网络硬件加速器和集成电路设计.

王晓蕾 女, (1978-), 硕士, 副教授. 研究方向为集成电路设计理论研究.

李桢旻 男, (1982-), 博士, 讲师. 研究方向为片上网络体系结构、机器学习、加密算法硬件实现.