DOI:10.3969/j.issn.1003-5060.2024.12.011

高能效低延迟的 BNN 硬件加速器设计

周培培，　杜高明，　李桢旻，　王晓蕾
(合肥工业大学 微电子学院,安徽 合肥　230601)

摘　要:针对二值化神经网络(binary neural network,BNN)硬件设计过程中大量 0 值引发计算量增加以及 BNN 中同一权值数据与同一特征图数据多次重复运算导致计算周期和计算功耗增加的问题,文章分别提出全 0 值跳过方法和预计算结果缓存方法,有效减少网络的计算量、计算周期和计算功耗;并基于现场可编程门阵列(field programmable gate array,FPGA)设计一款 BNN 硬件加速器,即手写数字识别系统。实验结果表明,使用所提出的全 0 值跳过方法和预计算结果缓存方法后,在 100 MHz 的频率下,设计的加速器平均能效可达 1.81 TOPs/W,相较于其他 BNN 加速器,提升了 1.27～4.34 倍。

关键词:二值化神经网络(BNN);权值共享;重复运算;现场可编程门阵列(FPGA);硬件加速器
中图分类号:TN47　　**文献标志码**:A　　**文章编号**:1003-5060(2024)12-1655-07

Design of energy-efficient low-latency BNN hardware accelerator

ZHOU Peipei,　DU Gaoming,　LI Zhenmin,　WANG Xiaolei
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: There are a large number of zero values used in the operation of binary neural network (BNN) applications, which leads to the surge of computations, as well as computing delay and computing power caused by repeated operations of the same weight data and feature graph data in BNN. In this paper, the methods of all-zero skipping and precomputed result caching are proposed. The proposed methods can effectively reduce the computation cost, computing delay and computing power. In addition, a BNN hardware accelerator based on field programmable gate array(FPGA) is designed and applied to handwritten digit recognition system. The experimental results show that after applying the proposed methods, the average power efficiency of the accelerator can reach 1.81 TOPs/W at the frequency of 100 MHz, which is 1.27-4.34 times higher than that of other BNN accelerators.

Key words: binary neural network(BNN); weight sharing; repeated operation; field programmable gate array(FPGA); hardware accelerator

0　引　　言

近年来,卷积神经神经网络(convolutional neural networks,CNN)在智慧医疗[1]、自动驾驶[2]、图像识别[3]等领域起着越来越重要的作用。为了追求更高的识别精度,CNN 模型朝着更深、更复杂的方向发展,导致参数量和计算量越来越大。以 VGG-16[4]为例,大约需要 1.38×10^8 个权重参数量和 1.55×10^{10} 次的计算量。如此高的参数量和计算量对存储和计算提出了更高的要求,难以应用在资源有限的嵌入式设备中,因此目前很多研究主要从减少参数量和计算量角度优化 CNN[5-7]。其中最引人注意的是文献[7]提出的二值化神经网络(binary neural network,BNN)

收稿日期:2023-02-21;**修回日期**:2023-03-10
基金项目:国家重点研发计划资助项目(2018YFB2202604);安徽省高校协同创新资助项目(GXXT-2019-030)
作者简介:周培培(1995—),男,安徽合肥人,合肥工业大学硕士生;
杜高明(1977—),男,湖南隆回人,博士,合肥工业大学教授,硕士生导师,通信作者,E-mail:dugaoming@hfut.edu.cn.

方案,该方案将权重和输入数据全部用“-1”或者“+1”代替,可减少 32 倍的存储量,并且使用同或(XNOR)和位计数运算(Popcount)代替乘累加运算,在降低参数量的同时也简化了计算的复杂度,实现快速推理。

因为 BNN 二值的特点更加有利于硬件设计,所以基于 BNN 的硬件加速器方案相继被提出。文献[8]提出一种输入数据二值化方法和奇偶填充方法以解决第 1 层单独设计全精度计算单元问题和填充 0 值引发的三值问题;文献[9]提出一种 6∶2 压缩器方案以跳过 XNOR 和 Popcount 计算,解决了重复计算问题,提高了计算效率,并针对 0 值填充问题提出一种边缘 0 值跳过方案;文献[10]提出一种分时复用查找表方案,以解决卷积计算过程中重复计算问题;文献[11]提出一种分析模型以获取最优参数,并基于 Popcount 特性提出一种高效的计算阵列,设计出一款灵活可配置的 BNN 加速器,实现不同网络模型的计算。

上述方案都取得了不错的效果,但是 BNN 中还存有以下问题需要解决:

1) BNN 中 0 值会导致计算量增加。BNN 硬件设计中的 0 值实际上代表着软件中的“-1”值,因此在计算时不能被忽略,导致计算量增加,延长了计算周期;而 CNN 中的 0 值没有任何意义,其与任何数的乘法结果仍为 0,因此可以跳过 0 的计算过程。这种 0 的差异使得 CNN 中现有的许多 0 值跳过方案[12-13]无法应用在 BNN 中。

2) BNN 中重复计算会导致功耗增加。文献[9]提出的 6∶2 压缩方案会使查找表的资源随着计算单元的增加而增加;文献[10]提出的分时复用查找表方案需要不断地制表,不仅制表时间长,制表还会导致额外的资源开销。

针对上述问题,本文提出不同的解决办法,相关工作如下:

1) 针对 BNN 中大量 0 值带来的计算量增加问题,本文提出一种全 0 值跳过方法。该方法适用于输入数据中至少有 2 行或者 2 列全 0 值的情况,可减少图片计算的尺寸,提高计算效率,节省计算功耗。根据统计,本文方法最高可减少手写数字图片中 89.8%的计算量,实现网络的快速推理。

2) 针对 BNN 计算过程中重复计算问题,提出一种预计算结果缓存方法。该方法不仅可以跳过 XNOR 和 Popcount 的计算过程,直接得到最终的运算结果,还可以解决计算重复问题,降低计算功耗。

3) 本文对 LeNet 模型进行改进,搭建一个新的 LeNet_BNN 网络模型,通过减小卷积核尺寸和全连接层数减少参数量,最终网络权重参数量仅为 2.589 KiB,相较于 LeNet 减少 65.5%。此外本文设计一款基于现场可编程门阵列(field programmable gate array,FPGA)的 BNN 硬件加速器,由于采用了全 0 值跳过方法和预计算结果缓存方法,该加速器吞吐率最高可达 3.2 TOPs。

1 BNN 理论基础

1.1 BNN 原理

BNN 的基本思想是通过量化函数对浮点型的权重数据和特征图数据都进行量化处理,最终将全精度的浮点数据量化成只有二值的数据。量化函数为:

$$y_{\text{act}} = \text{sign}(x) = \begin{cases} +1, & x \geqslant 0; \\ -1, & x < 0 \end{cases} \tag{1}$$

当特征图数据或权重数据值大于等于 0 时,将该值置为“+1”;当特征图数据或者权重数据值小于 0 时,将该值置为“-1”。

1.2 BNN 编码方式

式(1)的结果是有符号数,硬件设计时至少需要 2 bit 数据表示,这会增加额外的资源开销。由于数据是二值的,因此对式(1)的结果重新编码,编码方式为:

$$A\langle 0,1\rangle = \frac{A\langle -1,1\rangle + 1}{2} \tag{2}$$

其中:$A\langle -1,1\rangle$ 表示编码前数据取值为“-1”和“+1”;$A\langle 0,1\rangle$ 表示编码后数据取值为 0 和 1,即“-1”值可用 0 值表示,“+1”值用 1 值表示。编码后可用单比特数据表示,减少了资源开销。经过编码后,原来的乘累加运算可用 XNOR 和 Popcount 代替,进一步降低了计算的复杂度。计算公式为:

$$y = 2\text{pop}(a \odot w) - L_{\text{en}} \tag{3}$$

其中:a 表示输入数据;w 表示权重数据;L_{en} 表示参与计算的向量长度;pop 表示 Popcount 运算;y 表示最终的卷积计算结果。该结果与乘累加的结果是一致的。

因为进行的是 XNOR 运算,所以存在着一些特性,即

$$A \text{ XNOR } B = B \text{ XNOR } A \tag{4}$$

$$A \text{ XNOR } \overline{B} = \overline{(A \text{ XNOR } B)} \tag{5}$$

$$\overline{A}\ \text{XNOR}\ \overline{B} = A\ \text{XNOR}\ B \tag{6}$$

其中：A、B 分别表示 2 个变量；“—”表示取反操作。式(4)表示同或运算的交换性质；式(5)表示 A 与 B 的相反数进行同或运算，其结果等同于对 A 与 B 同或结果的取反；式(6)表示 A 的相反数与 B 的相反数进行同或，结果等价于 A 与 B 同或结果。

为了加快网络训练和收敛速度，避免过拟合，通常在卷积运算结束后进行批量归一化(batch normalization，BN)操作[14]，BN 的计算公式为：

$$y_{\text{bn}} = \gamma \frac{y-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta \tag{7}$$

其中：y 为卷积核的计算结果；μ、σ 分别为输入数据的均值和标准差；ε 为避免分母为 0 而设置的极小的正常数；γ、β 分别为缩放因子和偏移因子，且 γ 为正数。但是上述计算复杂度较高，不利于硬件实现，因此将式(7)与其后的激活函数层进行融合，可得：

$$y_{\text{bn}} = \begin{cases} +1, & y \geqslant \mu - \dfrac{\beta}{\gamma}\sqrt{\sigma^2+\varepsilon}; \\ -1, & y < \mu - \dfrac{\beta}{\gamma}\sqrt{\sigma^2+\varepsilon} \end{cases} \tag{8}$$

融合后降低了 BN 层的计算复杂度，非常有利于硬件的实现。

2　低延时高能效手写数字识别系统

2.1　系统整体架构

手写数字识别系统即 BNN 加速器的整体架构如图 1 所示。该 BNN 加速器主要由全局控制单元、输入/输出缓存单元、权重和偏置缓存单元、计算核(computing core，CC)模块组成。

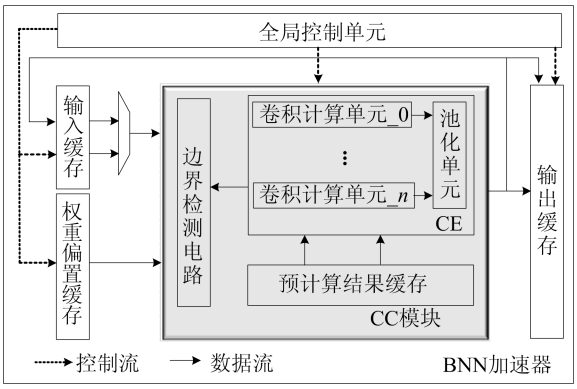


图 1　BNN 加速器整体架构

CC 模块包括边界检测电路、计算单元(computing element，CE)和预计算结果缓存单元。边界检测电路主要是检测非全 0 区域，此区域需要进行计算，非 0 区域外的是全 0 区域，不需要计算，可直接跳过；CE 由 n 组(本文中 $n=16$)卷积计算单元和池化单元组成，卷积计算单元用于实现卷积运算，池化单元用于实现池化操作；预计算缓存单元是一个 10×3 的寄存器堆，存储着所有的输入数据和权重数据的运算结果。当 CE 单元计算时，根据权重和输入数据组成的索引，访问该预计算结果缓存单元，直接得到相对应的计算结果，从而跳过了整个计算过程，加快了网络推理速度。

1) 系统启动时，全局控制单元首先向片外存储器发出读数据请求，读取输入特征图数据和权重数据分别存储在输入缓存和权重偏置缓存单元中。

2) 在全局控制单元的作用下，将此次需要运算的输入特征图数据和权重偏置数据加载至 CC 模块，通过边界检测电路检测出需要计算的非全 0 区域后，将该区域的输入特征图数据和相应的权重偏置数据传送至 CE 模块中进行运算；对于全 0 区域可根据检测电路检测到的地址将预计算结果直接输出至输出缓存中即可。

3) 在 CE 模块中，需要对输入特征图数据和权重数据进行重新排序，然后将排序的结果作为索引地址，从预计算结果缓存中得出其计算结果，经卷积计算单元内部的加法树累加得到卷积的结果。

4) 经池化单元的处理后，将池化结果输出至输出特征图缓存中。当该层所有计算完成后，将该层的输出特征图数据作为下一层的输入特征图数据，开始下一层的运算，重复上述过程，直至完成所有层的运算为止。

2.2　全 0 值跳过方法

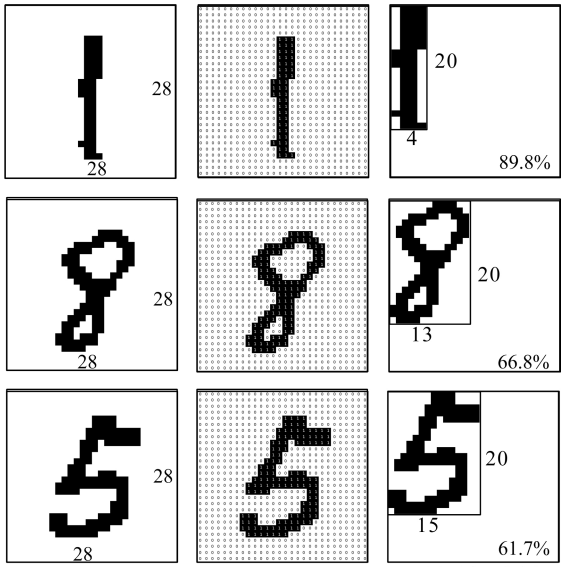
2.2.1　设计思路

在一个卷积核中，如果输入特征图的数据都是 0 值，那么其与权重数据的计算结果是一个定值。只要提前计算出这个值，即可跳过这些全 0 值的计算，减少计算量。MNIST 手写数字图片中全 0 值所占比重如图 2 所示。

图 2a 所示为从 MNIST 数据集中随机选取的 3 张手写数字图片；其完全二值化后的像素值如图 2b 所示；图 2c 所示为计算出的全 0 值所占比重。

从图 2 可以看出，手写数字 1 中全 0 值所占比例最高，为 89.8%，即使用全 0 值跳过方法可

减少 89.8%的计算量。



(a) 原始图片 (b) 原始图片像素值 (c) 行列全0值占比

图 2 MNIST 图片中行列全 0 值所占比重

2.2.2 硬件设计

因为手写数字中全 0 值分布不均匀,相较而言,非全 0 值的分布更加均匀,所以本文通过检测非全 0 值的计算区域来确定全 0 值的范围,检测电路如图 3 所示。

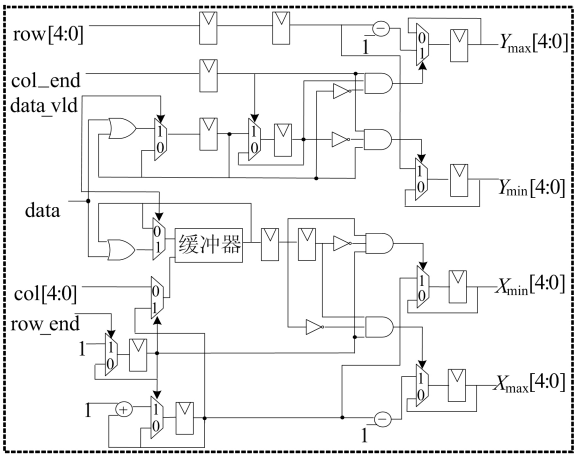


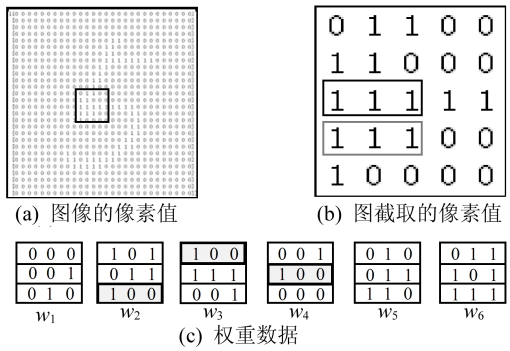
图 3 边界检测电路

首先,从左向右按列读取图像数据,当读取完某一行所有列时,从下一行开始读取,直至读完整个图像。在读取某一行的过程中,对数据不断地进行位或运算,当一行计算完成后开始进行检测;若检测到相邻两行的结果从 0 变为 1,则此时检测到了最小的行值 Y_{min};若检测到相邻两行的结果从 1 变为 0,则检测到最大的行值 Y_{max}。对于列的检测,需要等图像全部读取完之后才能进行。因此在行检测时对列像素进行位或计算并将结果暂存在缓冲器中,等读取完图片后,再进行行列的检测,检测方法与行检测一致,最终可以确定最小的列值 X_{min} 和最大的列值 X_{max}。当确定完列的最大值后,检测结束,由此可以确定需要计算的区域及全 0 的区域。

2.3 预计算结果缓存方法

2.3.1 设计思路

BNN 中,将一个 3 * 3 的卷积核按行拆分,可得到 3 个 1 * 3 的行卷积核,每个行卷积核的取值只有 8 种情况,即 000,001,010,…,110,111。网络中的重复计算如图 4 所示。



(a) 图像的像素值 (b) 图截取的像素值 (c) 权重数据

图 4 网络中的重复计算

图 4c 中每个卷积核均不相同,但是卷积核在滑动过程中,卷积核 w_2、w_3、w_4 中行卷积核“1 0 0”分别与图 4b 特征图中的“1 1 1”进行运算,导致重复计算,这种重复计算是冗余且非必要的,不仅会消耗计算周期,还会带来功耗损失。

针对上述问题,本文提出预计算结果缓存方案。通过观察分析得到权重数据与特征图数据间潜在的关系:1 * 3 的行卷积核中,权重数据的取值只有 8 种,与之对应的特征图数据也应该只有 8 种取值情况;两两组合共有 64 种情况。64 种组合方式如图 5a 所示。

但在这些组合中又存在着许多相似的运算,如 010 XNOR 011 与 011 XNOR 010 的数据是相同的。根据式(4),可将这 2 个组合方式归为 1 种。同理,根据式(5)和式(6)可以进一步压缩 64 种中的组合方式,最终压缩的情况如图 5b 所示。从图 5b 可以看出,简化后的组合方式只有 10 种,相较于图 5a 中 64 种组合方式,减少了 84.4%。

由于图 5b 中组合方式是固定的,可以根据式(3)提前计算出最终结果并保存在存储器中,之

后权重和输入数据的结果可通过查找此存储器的方式得到。

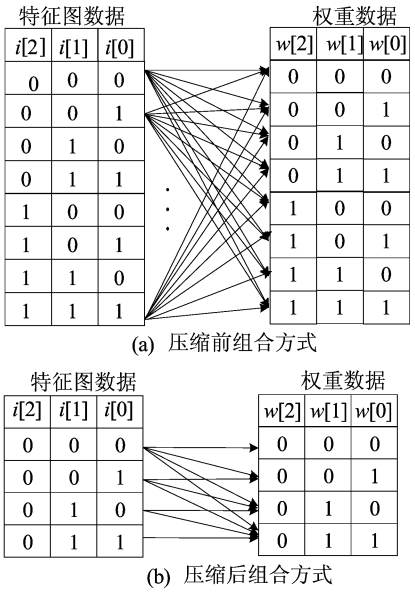


图 5　压缩前后数据的组合方式

2.3.2　硬件设计

预计算结果缓存和计算单元示意图如图 6 所示。其中，预计算结果缓存是由 10×3 的寄存器堆组成，存储着 10 种权重和输入的组合结果；卷积计算单元由 3 组处理单元（processing element，PE）阵列和 1 个加法树组成，用以实现一个 3 * 3 的卷积核计算；每个 PE 阵列由 8×14 个 PE 组成，一次可实现 8×14 的行卷积核计算。

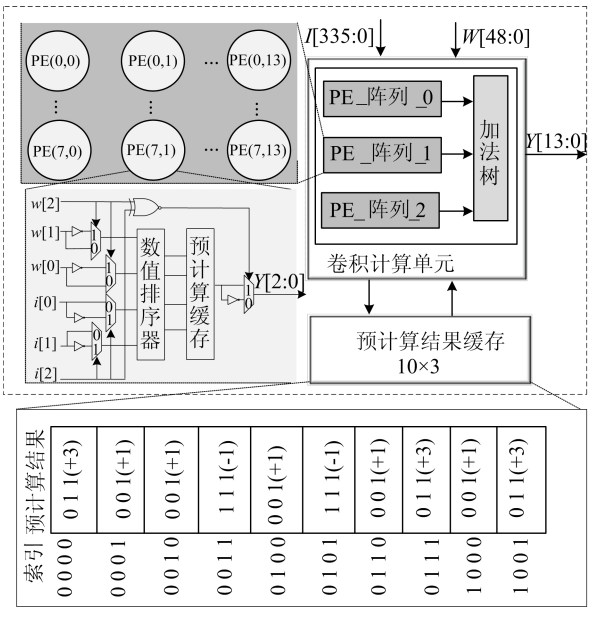


图 6　预计算结果缓存及计算单元示意图

每个 8×14 的 PE 阵列中：8 表示 8 个输入通道；14 表示每个输入通道的 14 个行卷积数据；每个 PE 实现 1 个行卷积核的运算。

当输入和权重数据进入卷积计算单元后，会将数据进行拆分，分配到每个 PE 单元中参与运算；在 PE 中需要对输入和权重数据按大小排序，将排序的结果作为预计算结果缓存的索引查找对应的结果；根据输入和权重数据最高位同或的结果，判断是否需要对读出的结果进行取反操作；最后将 3 个 PE 阵列的计算结果通过加法树累加后输出至池化单元中。

3　实验结果分析

3.1　实验环境与 L_BNN 模型

实验使用 Verilog 语言以及 Vivado2019.1 进行硬件开发和测试；基于 Pytorch 和 Jupyter 进行神经网络的软件训练。

本文对 LeNet 进行改进，搭建一个新的 L_BNN 模型，该网络模型只有 4 层，具体结构见表 1所列。

表 1　L_BNN 模型结构

层数	结构	输入量	输出量	权重量/bit	计算量
1	C-B-A-P	1×28^2	6×14^2	54	42 336
2	C-B-A-P	6×14^2	16×7^2	864	169 344
3	C-B-A	16×7^2	32×7^2	4 608	225 792
4	F-B-A	32×7^2	10	15 680	15 680
总计				21 206	453 142

注：C 表示卷积层（Conv）；B 表示批归一化层（BN）；A 表示激活层（Activation）；P 表示池化层（Pooling）；F 表示全连接层（FC）。

表 1 中，计算量表示乘法的数量。

搭建的 L-BNN 网络模型中：卷积核尺寸为 3×3，移动步长为 1，每个卷积层都进行了填充；池化的尺寸为 2×2，步长为 2，采用最大池化方法。

L-BNN 网络模型最终对 MNIST 的训练精度达到 95.8%，权重量为 21 206 bit，相较于 LeNet[15] 精度下降 3.0%，权重量减少了 65.5%，即在精度损失较小的情况下，极大地减少了权重参数量。

3.2　全 0 值跳过方法效果验证

不同全 0 值尺寸对 3 个卷积层计算量跳过比例见表 2 所列。由于行与列的分析方法相同，在此只分析不同行全 0 值尺寸对 3 个卷积层计算量的影响。

从表2可以看出:随着全0值尺寸的增大,3个卷积层可跳过的计算量比例也会增大;但是受池化的影响,同一全0行尺寸在3个卷积层的计算量比例呈下降趋势。

表2 不同行全0值尺寸对3个卷积层计算量跳过比例 %

全0行尺寸	卷积层1	卷积层2	卷积层3
0×0	0	0	0
1×28	0	0	0
2×28	7.14	0	0
4×28	14.29	0	0
5×28	17.86	14.29	0
10×28	35.71	28.57	0
11×28	39.29	35.71	28.57
20×28	71.43	64.29	57.14
24×28	85.71	78.57	71.43

从表2还可以看出:输入层全0值不多于5行且不少于2行时,仅在卷积层1中有可跳过的计算;当输入层全0值不多于11行且不少于5行时,仅在卷积层1和卷积层2中有可跳过的计算;当输入层全0值大于11行时,3个卷积层均有可跳过的计算。

3.3 预计算结果缓存方法效果验证

为了更好地预计算缓存,评估本文方法的效果,将本文方法同文献[7]、文献[9]方法进行对比,除计算方式不同外,其他条件均保持相同,3种方法的资源消耗见表3所列。

表3 3种方法的计算方式资源消耗对比

方法	LUT	FF
文献[7]	168	155
文献[9]	155	142
本文方法	156	63

从表3可以看出,本文方法使用的触发器(flip-flop,FF)资源要明显低于其他2种方法,本文方法使用的查找表(look-up table,LUT)资源比文献[9]高的原因是由于本文方法需要对最终的结果进行逻辑判断,而这个操作需要消耗LUT资源。

3.4 手写数字识别系统整体性能测试

进行Vivado综合,手写数字识别系统资源消耗情况见表4所列。

从表4可以看出,FF的资源消耗占总体的7.61%,BRAM资源消耗占总体的7.19%,资源消耗较少。由于整个系统需要进行较多的逻辑控制,因此LUT的资源使用量较高,占总体的43.71%。

表4 手写数字识别系统资源消耗情况

资源	总使用量	资源可利用量	利用率/%
LUT	89 080	203 800	43.71
FF	31 004	407 600	7.61
BRAM	32	445	7.19

为了进一步分析本文设计的BNN加速器性能,与现有的几款BNN硬件加速器性能进行对比,结果见表5所列。

表5中给出了峰值吞吐量和平均吞吐量,造成这种差异的原因是受网络模型的影响,不同卷积层对计算单元的利用率不同,使得计算单元整体性能没有全部发挥出来。

从表5可以看出,本文设计的加速器大大降低了推理时间,相较于文献[11]加速器提升近5倍,并且也取得了不错的能效比,最高可达4 088.2 GOPs/W。

表5 加速器性能对比

加速器	文献[8]	文献[11]	文献[16]	文献[10]	文献[9]	本文
Dataset	SVHN		SVHN	CIFAR-10	MNIST	MNIST
Platform	Xilinx ZC702	Xilinx XC7Z100	Kintex XCKU115	Xilinx ZCU102	Artix-7 XC7A100T	Kintex-7 XC7K325T
Clock/MHz	143	166	150	200	100	100
Precision	1-bit		1-bit	1-bit	1-bit	1-bit
Power/W	3.200	5.700	26.400	0.900	0.399	0.789
Latency/μs	1 550	3		247.1	34	0.6
Performance^{O}/(GOPs)	2 236.0	1 932.0	21 042.0			1 428.6
Performance^{P}/(GOPs)				3 157.0	1 184.5	3 225.6
Efficiency^{O}/(GOPs/W)	699.0	339.0	797.1			1 810.5
Efficiency^{P}/(GOPs/W)				3 507.8	2 968.6	4 088.2

注:上标O表示Overall(均值);上标P表示Peak(峰值);Dataset表示数据集;Platform表示平台;Clock表示频率;Precision表示精度;Power表示功耗;Latency表示延时;Performance表示性能;Efficiency表示能效。

4 结　论

本文针对完全二值化手写数字图片中存在大量的全 0 值导致计算量增加和计算过程中重复计算导致推理时间与功耗增加的问题，分别提出全 0 值跳过方法和预计算结果缓存方法；并基于上述 2 种方法设计了一款基于 FPGA 的 BNN 硬件加速器，即手写数字识别系统。实验结果表明，该系统平均能效可达 1.81 TOPs/W，具有延时低、能效高等特点，非常适合应用在容量和功耗有限的移动嵌入式设备中。

[参　考　文　献]

[1] KHAN P, KADER M, ISLAM S, et al. Machine learning and deep learning approaches for brain disease diagnosis: principles and recent advances [J]. IEEE Access, 2021, 9(1): 37622-37655.

[2] LI P, CHEN X, SHEN S. Stereo R-CNN based 3D object detection for autonomous driving [C]//2019 CVF Conference on Computer Vision and Pattern Recognition. Long Beach, CA: IEEE, 2019: 7636-7644.

[3] RAJYALAKSHMI P, SATHAPPAN S, JYOTHI RANI M, et al. A novel image processing system to recognize and classify images using integrated advanced CNN and FPGA for IoT based applications [C]//2022 4th International Conference on Inventive Research in Computing Applications. Coimbatore: IEEE, 2022: 365-368.

[4] LIANG S, YIN S, LIU L, et al. FP-BNN: binarized neural network on FPGA [J]. Elsevier Neurocomputing, 2018, 275(1): 1072-1086.

[5] HAN S, MAO H Z, DALLY W. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding [J]. Fiber, 2015, 56(4): 3-7.

[6] WESS M, DINAKARRAO S, JANTSCH A. Weighted quantization-regularization in DNNs for weight memory minimization toward HW implementation [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(11): 2929-2939.

[7] COURBARIAUX M, HUBARA I, SOUDRY D, et al. Binarized neural networks: training deep neural networks with weights and activations constrained to + 1 or − 1 [EB/OL]. (2016-02-09) [2016-03-17]. https://doi.org/10.48550/arXiv.1602.02830.

[8] GUO P, MA H, CHEN R Z, et al. FBNA: a fully binarized neural network accelerator [C]//2018 28th International Conference on Field Programmable Logic and Applications. Dublin: IEEE, 2018: 51-54.

[9] DU G, CHEN B, LI Z, et al. A BNN accelerator based on edge-skip-calculation strategy and consolidation compressed tree [J]. ACM Transactions on Reconfigurable Technology and Systems, 2022, 15(3): 1-20.

[10] 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.

[11] GAO J, YAO Y, LI Z, et al. FCA-BNN: flexible and configurable accelerator for binarized neural networks on FPGA [J]. IEICE Transactions on Information and Systems, 2021, 104(8): 1367-1377.

[12] KIM Y, AN J, SUNWOO M. CASA: a convolution accelerator using skip algorithm for deep neural network [C]//2019 IEEE International Symposium on Circuits and Systems. Sapporo: IEEE, 2019: 1-5.

[13] LIU M, HE Y, JIAO H. Efficient zero-activation-skipping for on-chip low-energy CNN acceleration [C]//2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems. Washington, D. C.: IEEE, 2021: 1-4.

[14] TING Y, TENG Y, CHIUEH T. Batch normalization processor design for convolution neural network training and inference [C]//2021 IEEE International Symposium on Circuits and Systems. Daegu: IEEE, 2021: 1-4.

[15] ZHANG L, BU X, LI B. Xnorconv: CNNs accelerator implemented on FPGA using a hybrid CNNs structure and an inter-layer pipeline method [J]. IET Image Processing, 2020, 14(1): 105-113.

[16] HAN Z, JIANG J, XU J, et al. A high-throughput scalable BNN accelerator with fully pipelined architecture [J]. CCF Transaction on High Performance Computing, 2021, 3(1): 17-30.

（责任编辑　胡亚敏）