DOI: 10.13382/j.jemi.B2408032

基于 FPGA 的高效近似乘法器设计*

杨宏浩[1] 隋欣宇[1] 王一然[3] 刘恩华[1] 潘澜澜[2] 李 响[1]

(1. 大连海洋大学信息工程学院 大连 116023;2. 大连海洋大学机械与动力工程学院 大连 116023;
3. 山东大学软件学院 济南 250100)

摘 要:针对现场可编程门阵列(field-programmable gate array, FPGA)在加速卷积神经网络、图像处理算法等近似计算领域中模型不彻底、片上资源消耗较大、性能受限等问题,提出了 5 款近似乘法器设计方法。该方法基于查找表(LUT)的 8 bit×8 bit 无符号无进位链近似乘法器,通过编码 LUT 的 INIT 参数值来优化关键路径简化结构并利用压缩递归调用方法、子积重组计算方法,提出了两款适用于不同现实场景的基于 LUT 的 8 bit×8 bit 无符号近似乘法器。该方法在精度可接受的范围内与同类型乘法器相比最高可节省 60%的面积、约 60.76%的功耗、约 25.4%的关键路径延迟(critical path delay, CPD)。同时,为了满足更加复杂的场景需要,在上述基础上将乘数位数倍增,提出了两款基于 LUT 的 16 bit×16 bit 无符号近似乘法器,与同类型乘法器相比最高可节省约 41.2%的面积、约 77%的功耗、约 35.4%的 CPD 并能弥补精度下降带来的损失。此外,结合所提出的有符号数计算模块提出了一款基于 LUT 的 16 bit×16 bit 有符号近似乘法器来替代 Xilinx(现 ADM)的 Multiplier IP 核,部署至以手写数字识别为功能的卷积神经网络卷积层中并选用 MNIST 数据集中的手写数字图片进行测试,以精度下降 3.4%的代价换取节省约 32.48%的面积、约 41.21%的功耗、约 24.28%的 CPD。实验结果表明,这些乘法器可以较好的满足 FPGA 加速卷积神经网络的需求并在精度与资源开销达成最优平衡。

关键词:乘法器;FPGA;近似容错计算;查找表

中图分类号:TN43;TN791 **文献标识码**:A **国家标准学科分类代码**:510.4030

FPGA-based design of high-efficiency approximate multipliers

Yang Yonghao[1] Sui Xinyu[1] Wang Yiran[3] Liu Enhua[1] Pan Lanlan[2] Li Xiang[1]

(1. School of Information Engineering, Dalian Ocean University, Dalian 116023, China; 2. School of Mechanical and Power Engineering, Dalian Ocean University, Dalian 116023, China; 3. School of Software, Shandong University, Jinan 250100, China)

Abstract: Five approximate multiplier design methods are proposed to address the issues of incomplete models, high on-chip resource consumption, and limited performance of Field Programmable Gate Array (FPGA) in accelerating convolutional neural networks, image processing algorithms, and other approximate computing fields. Based on an 8-bit×8-bit unsigned carry chain approximation multiplier, two LUT based 8-bit×8-bit unsigned approximation multipliers are proposed for different real-world scenarios with a lookup table (LUT) to optimizing the critical path simplification structure by compressed recursive invocation methodology and sub-product recombination computation strategy. This method can save up to 60% of area, about 60.76% of power consumption, and about 25.4% of critical path delay (CPD) compared to similar multipliers within an acceptable range of accuracy. At the same time, in order to meet the needs of more complex scenarios, two 16 bit×16 bit unsigned approximate multipliers with LUT are proposed by doubling the number of multiplies digits. Compared with similar multipliers, the method can save up to about 41.2% of the area, about 77% of the power consumption, about 35.4% of the CPD, which can compensate for the loss caused by the decrease in accuracy. In addition, based on the signed number calculation module, proposed a 16 bit × 16 bit signed approximate multiplier with LUT is proposed to replace Xilinx's (now ADM) Multiplier IP core, which is deployed in the convolutional neural network convolutional layer with handwritten number recognition

收稿日期: 2024-12-10 Received Date: 2024-12-10

* 基金项目:辽宁省应用基础研究计划(2023-179)、辽宁省研究生教育教学改革研究项目(LNYJG2024198)资助

function and tested using handwritten number images in the MNIST dataset. It saves about 32.48% of area, about 41.21% of power consumption, and about 24.28% of CPD, at the cost of a 3.4% decrease in accuracy. It is shown that these multipliers can effectively meet the requirements of FPGA accelerated convolutional neural networks and achieve the optimal balance between accuracy and resource overhead.

Keywords: multiplier; FPGA; approximate fault-tolerant computation; LUT

0　引　言

当前,基于深度学习的图像处理模型(如卷积神经网络)普遍面临参数量庞大、计算复杂度高的问题,这给硬件资源的高效利用带来了巨大挑战。因此,如何在现有硬件条件下实现模型加速已成为研究热点[1]。在该领域中主流的硬件加速器有 3 款,图形处理器(graphics processing unit, GPU),现场可编程门阵列(field-programmable gate array, FPGA)和专用集成电路(application specific integrated circuit, ASIC)[2]。对比上述 3 款加速器,GPU 虽拥有较多计算资源,性能占优,但其需要根据特定的结构进行开发,因此效率低下。ASIC 的运行速度在 GPU、FPGA 三者较快,但 ASIC 的结构固定,其开发和设计需要较强的硬件电路背景,且研发周期较长。相比之下,FPGA 设计灵活、具备并行设计和能源效率高等优势,因此 FPGA 逐渐成为硬件加速的理想选择[3]。

使用 FPGA 来加速卷积神经网络等算法,其运行速度与 FPGA 的片上资源有较大关联。当前大多的 FPGA 加速方案须调用大量的 Multiplier IP 核进行计算,存在着 FPGA 资源尤其是有限的 DSP 数字信号处理器资源和查找表(LUT)资源消耗量较大甚至被耗尽而导致应用程序性能下降等问题[4]。蒋康宁等[5]对数据量化为位 16 bit 的定点数后,采用输入输出通道并行展开的流水线处理方式在宇航级的 FPGA 上部署了 YOLOv5s 算法,其片上 DSP 消耗量达到了 85%。陈昌川等[6]采用动态定点量化的方式将 32 bit 的权重和特征图数据定点量化为 8 bit,并采用通道剪枝技术将岩渣分类卷积神经网络部署在 FPGA 上进行加速,其片上 DSP 消耗量达到了 99%。黄沛昱等[7]将卷积神经网络模型中的权重、偏置项、和输入输出的特征值量化为 16 bit 定点数,采用循环展开和分块的方法将卷积神经网络部署在 FPGA 上进行加速,其片上 LUT 的消耗量达 73.2%,片上 DSP 资源消耗量达到 66.8%。Zhang 等[8]对数据进行 8 bit 量化,采用双符号乘法校正电路在 FPGA 上对 YOLOv2-tiny 网络模型进行加速,其片上 LUT 消耗量达到约 40%,片上 DSP 资源消耗量达到 95.2%。戴伟杰等[9]将 32 bit 浮点数量化为 16 bit 定点数,通过并行计算后重组的技术将铝片识别功能的卷积神经网络部署在 FPGA 上,其片上 LUT 的消耗量近 70%,片上 DSP 资源消耗量达到 68.64%。

为了解决上述问题的制约,诸多学者提出了基于 FPGA 的近似计算方法。虽然该方法只是精确结果的近似,但是由于卷积神经网络模型的鲁棒性,即便计算结果略有偏差也可以确保模型的准确率在可接受范围内。同时,采用近似计算代替精确计算带来的性能提升、功耗增益也非常明显。Ullah 等[10]针对上述问题,提出了基于 LUT 的 8×8 bit 的无符号近似乘法器来降低资源消耗,但其面积、功耗和 CPD 仍然较高。文献[11-12]通过去除进位链的方式,提出了基于 LUT 的 8×8 bit 无符号无进位链近似乘法器,其资源消耗较低但精度亦较低。Waris 等[13]通过输入操作数的概率分布,提出了基于混合部分积的 8×8 bit 无符号近似乘法器,其性能依然有改进的空间。Guo 等[14]提出了 3 款基于 LUT 的 16×16 bit 无符号近似乘法器,其精度较高但资源节省量较少。

在保证精度和资源开销方面,诸多学者在卷积神经网络特征值和计算量优化方面做出了贡献,但利用 8 或 16 bit 有符号定点数设计近似乘法器时,大多无法同时权衡性能、精度和位数的需求。

针对上述问题,本文所作如下改进。

1) Proposed8×8_1:基于 LUT 的 8×8 bit 无符号近似乘法器。通过编码 INIT 参数值、简化输入参数结构的方式对无进位链近似乘法器中的近似加法器关键路径做出改进。在构建高阶乘法器时,本文提出压缩调用的方法替代传统递归调用低阶乘法器,以此较大幅度减少资源消耗。

2) Proposed8×8_2:基于 LUT 的 8×8 bit 无符号近似乘法器。为了权衡精度和资源消耗以适应不同应用,本文提出了子积重组计算模块代替 Proposed8×8_1 方法中的近似加法器。

3) Proposed16×16_1:基于 LUT 的 16×16 bit 无符号近似乘法器。为了应对卷积参数为 16 bit 无符号定点数的情况,在 Proposed8×8_1 乘法器基础上,本文提出了基于压缩调用技术的 Proposed16×16_1 近似乘法器,该乘法器在保证计算精度的同时消耗较少资源。

4) Proposed16×16_2:基于 LUT 的 16×16 bit 无符号近似乘法器。为了保证乘法器资源消耗与精度的均衡化以适应要求为复杂的应用场景,在 Proposed8×8_2 的基础上提出了 Proposed16×16_2 近似乘法器。

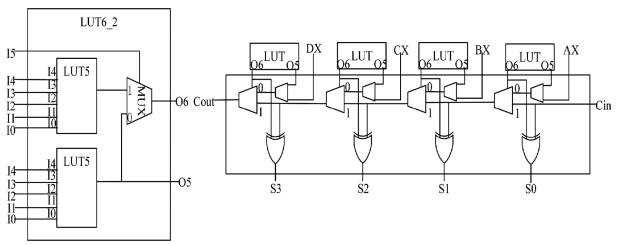
5) Proposed16×16_3:基于 LUT 的 16×16 bit 有符号

近似乘法器。为了适应有符号运算的特殊情况,在Proposed16×16_2乘法器基础上提出了增加补码计算模块的Proposed16×16_3近似乘法器。

1　相关工作

目前主流的FPGA由可编程逻辑功能块(configurable logic blocks, CLB)作为基本逻辑单元组成各种功能电路。每个CLB包含两个SLICE,每个SLICE由4个6输入的LUT、8个触发器和1个进位链组成[15]。

LUT可以配置成带有一个6输入的LUT6,6输出的O6端口作为其输出位。也可以配置成LUT6_2,两个5输入的端口I0~I4作为其输入,O5端口和O6端口作为其输出,如图1(a)所示。用户使用FPGA LUT原语来定义INIT参数从而修改LUT的逻辑功能。INIT参数由十六进制数组成,其实际上为输入端口逻辑表达式真值表的结果。此外LUT还被用来控制进位链以实现更为强大的逻辑功能。例如使用O5端口作为进位生成信号,O6端口作为进位传播信号,以此来实现超前进位加法器(carry-lookahead adder),如图1(b)所示。



(a) LUT6_2结构　(b) 四位charry chain结构
(a) LUT6_2structure　(b) Four-digit charry chain structure

图1　查找表与进位链结构

Fig. 1　LUT and carry chain structure

为节省LUT的消耗量,文献[16]提出了两种方法。第1种是删除LUT,则输出值只能是0或者1,如式(1)所示。第2种是当两个LUT的输入参数相同时,可以采用合并的方法将其中一个的输入信号设为1,由此来产生LUT6_2,如式(2)所示。

$$O_6 = f(I_5, I_4, I_3, I_2, I_1, I_0) \rightarrow O_6 = 0 or 1 \tag{1}$$

$$\left.\begin{aligned} O_{6_1} &= f_1(I_5, I_4, I_3, I_2, I_1, I_0) \\ O_{6_2} &= f_2(I_5, I_4, I_3, I_2, I_1, I_0) \end{aligned}\right\} \rightarrow \begin{cases} O_6 = f_1'(I_4, I_3, I_2, I_1, I_0) \\ O_5 = f_2'(I_4, I_3, I_2, I_1, I_0) \end{cases} \tag{2}$$

Intel等FPGA供应商以DSP的形式提供高性能乘法器。由于DSP资源较少并且使用DSP模块可能会导致应用程序性能下降。因此,FPGA供应商还提供了基于LUT的Multiplier IP核用来进行乘法计算。

2　基于LUT查找表的高效近似乘法器

2.1　基于LUT的4×4 bit无符号近似乘法器设计

传统二进制乘法在计算原理上和十进制的乘法相似,即进行移位操作后产生部分乘积然后相加得到最终结果。例如,4 bit二进制数$A_4A_3A_2A_1$和$B_4B_3B_2B_1$进行相乘,如图2所示。第1步是4 bit二进制数A和4 bit二进制数的每个位都通过运算产生部分乘积,第2步是对生成的部分乘积进行累加操作得到最终结果。在基于LUT的近似乘法器中并不是以此法计算二进制乘法,而是直接把A和B的每一位送进LUT中进行计算生成部分结果,最后在使用进位链的情况下生成最终乘法结果。

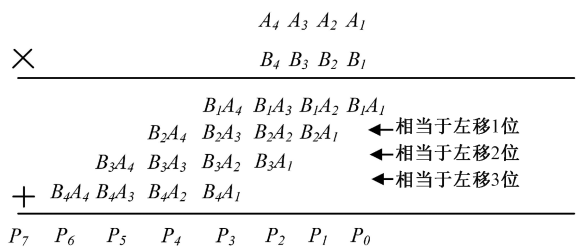


图2　二进制乘法图示

Fig. 2　Binary multiplication diagram

进位链是CLB中的一种基本结构,它被用来进行进位加法,但是使用进位链会带来额外的CPD。进位链在基于LUT的近似乘法器中会造成33.6%~55.2%的CPD[11]。在基于ASIC的近似乘法器中,通过OR-based压缩技术可以有效减少乘积的CPD,这种方法也同样适用于FPGA上[17]。

基于这种方法,Yao等[11]提出了基于LUT的无进位链无符号4×4 bit近似乘法器,该近似乘法器由6个LUT组成。将乘数和被乘数进行分割,每个LUT完成不同的运算分工。LUT1~LUT5使用OR运算来储存部分乘积,在LUT6中确保最终结果的精确性,每个LUT对应的功能如表1所示。

表1　基于LUT的4×4 bit无符号近似乘法器结构

Table 1　4×4-bit unsigned approximate multiplier structure based on LUT

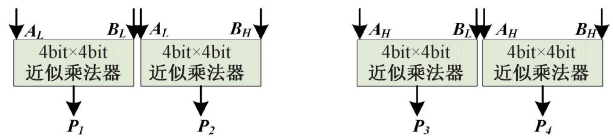
LUT编号	对应功能
1	$P_0 = A_0B_0$　$P_1 = A_1B_0 \mid A_0B_1$
2	$S = A_2B_1 \mid A_1B_2 \mid A_2B_0A_1B_1$; $P_2 = A_2B_0 \mid A_1B_1$
3	$P_3 = S \mid A_3B_0 \mid A_0B_3$
4	$P_4 = A_3B_1 \mid A_2B_2 \mid A_1B_3$
5	$P_5 = A_3B_2 \mid A_3B_2 \mid A_1B_3A_3B_1$
6	$P_6 = A_3B_3 \oplus A_3B_2A_2B_3$; $P_7 = A_3B_3 \& A_3B_2A_2B_3$

2.2　基于 LUT 的 8×8 bit 无符号近似乘法器整体设计

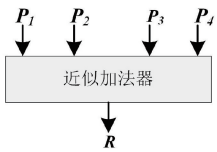
传统的基于 LUT 的 8×8 bit 无符号近似乘法器,其原理是递归调用基于 LUT 的 4×4 bit 无符号近似乘法器,如图 3 所示。

首先进行分块计算来产生部分乘积结果,然后在基于 LUT 的近似加法器中进行处理产生最终的结果。例如,8 bit 二进制数 A 与 8 bit 二进制数 B 相乘,首先将 A 的低四位分割为 A_L,高四位分割为 A_H,将 B 的低四位分割为 B_L,高四位分割为 B_H,如图 3(a)所示。其次,对基于 LUT 的 4 bit×4 bit 无符号近似乘法器进行递归调用,对 A_L、A_H 和 B_L、B_H 分别两两相乘得到部分乘积结果 P_1 ~ P_4,如图 3(b)所示。最后,如图 3(c)所示,调用基于 LUT 的近似加法器模块对上述部分乘积做处理得到最终的乘法结果 R。

(a) 将8bit二进制乘数A和B进行位数分割
(a) The 8-bit binary multiplier A and B aresplit by bits



(b) 对基于LUT的4×4 bit近似乘法器进行递归调用
(b) Recursive invocation of a LUT-based 4-by-4-bit approximate multiplier



(c) 对部分乘积结果进行最终处理
(c) The results of the partial productarefinally processed

图 3　基于 LUT 的 8×8 bit 无符号近似乘法器原理

Fig. 3　Overall design of 8×8-bit unsigned approximate multiplier based on LUT

传统的递归调用法是将一次 8×8 bit 的运算调用 4 次 4 个功能相同的 4×4 bit 的算子,如电路计算复杂将增加额外的 CPD 和资源消耗。为了改善上述问题,本文将 4 个 4×4 bit 的算子功能压缩进一个算子中提出了改进型的压缩递归调用方法。通过扩大算子的输入输出数量,将乘数与被乘数分割完成后只需调用一个算子即可同时完成计算并输出部分乘积结果 P_1~P_4。

近似加法器对所有乘法器的最后收尾工作,其面积、功耗、CPD 等性能直接影响到近似乘法器整体的性能。Ullah 等[18]提出了带有 3 个进位链的加法器,其功耗较高且 CPD 较长。文献[10]提出了无进位链的近似加法器,但其结构复杂仍然有改进的空间。首先,通过对该近似加法器进行剪枝并修改 8 个 LUT 的逻辑功能使其输出端口 O6=I0|I1|I2。通过 OR 运算只对输入的前 3 个参数进行处理,由此可以简化结构并提升近似加法器性能。其次,为了规避由 B[3]到 R[6]关键路径带来的较高 CPD,本文使用原语将第 2 个 LUT 的 INIT 参数编码为 64′hFEFEFEFEFEFEFEFE 用来处理 P_1[5]、P_2[1]、P_3[1]共 3 个部分乘积。最后,该近似加法器对部分乘积 P_1 ~ P_4 进行处理。将生成的结果 Y_1 ~ Y_5 与 P_1 的低四位和 P_4 的高四位通过拼接操作得到最终的乘法结果 R。改进基于 LUT 的无进位链近似加法器的其功能结构如图 4 所示。

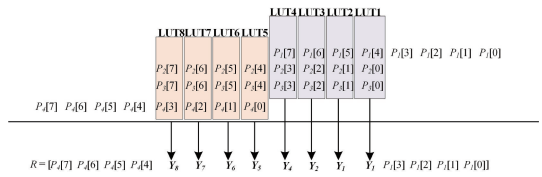


图 4　基于 LUT 近似加法器原理

Fig. 4　Based on the LUT approximate adder principle

结合上述压缩递归调用方法及改进的基于 LUT 的无进位链近似加法器,本文提出了改进的基于 LUT 的 8 bit×8 bit 无符号近似乘法器 Proposed8×8_1,结构如图 5(a)所示。其结构更为简单,在精度略微下降的情况下,与同类型乘法器相比其面积、功耗和 CPD 均有较大改善,适用于资源较为紧张的实际情况。

将 8 bit 二进制乘数 A 与 8 bit 二进制乘数 B 分割为 A_L、A_H、B_L、B_H 后根据二进制数的原理可知 $A = A_H \times 4 + A_L$,$B = B_H \times 4 + B_L$。在 FPGA 中直接使用乘法运算符(“*”)进行乘法会造成产生较大的 CPD 和功耗,因此采用 FPGA 中擅长的位移操作进行运算,如式(3)和(4)所示。

$$A = (A_H \ll 2) + A_L \tag{3}$$

$$B = (B_H \ll 2) + B_L \tag{4}$$

结合上述生成的部分乘积结果 P_1 ~ P_4,由此可推出 $A \times B$ 的数学关系在 FPGA 中如式(5)所示。

$$A \times B = (P_4 \ll 8) + P_1 + (P_3 + P_2) \ll 4 \tag{5}$$

根据该关系,本文提出了一种新的子积重组计算模块代替了乘积近似加法器来对生成的部分乘积结果进行处理,并名为 Proposed8×8_2,如图 5(b)所示。相较于同类型乘法器,该乘法器可以较好的取得资源开销与精度之间取得平衡,增加了方法的普适性,更加适应于复杂的实际场景。

2.3　16×16 bit 近似乘法器整体设计

本文在上述基础上提出了两款基于 LUT 的 16×16 bit 无符号近似乘法器和一款基于 LUT 的 16×16 bit 有符号近似乘法器。这不仅增强了乘法器的功能和现实

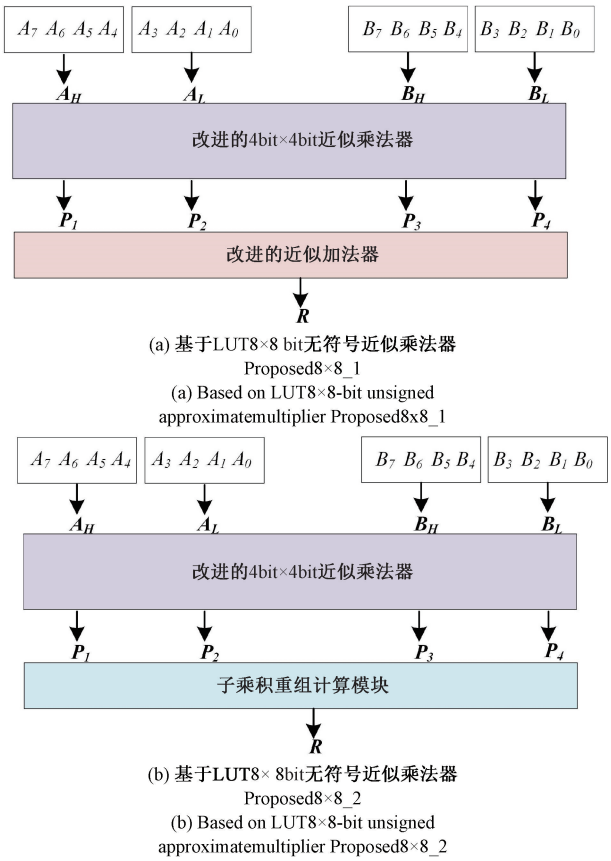


图 5　改进型基于 LUT 的 8×8 bit 无符号近似乘法器原理

Fig. 5　An improved LUT-based 8×8-bit unsigned approximate multiplier principle

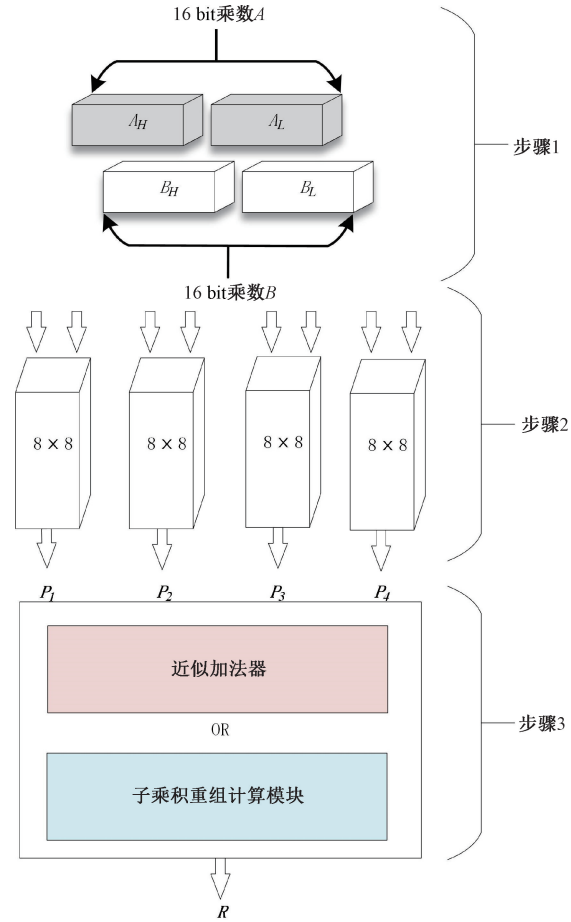


图 6　基于 LUT 的 16×16 bit 无符号近似乘法器原理

Fig. 6　Overall design of 16×16-bit unsigned approximate multiplier based on LUT

意义而且更适用于 FPGA 加速卷积神经网络算法。

其工作结构如图 6 所示。在步骤 1) 中采用 2.2 节提出的分块计算的方法对 16 bit 乘数 A 和 B 进行分块，产生 8 位分块乘数 A_L、A_H 和 B_L、B_H。在步骤 2) 中，调用本文提出的基于 LUT 的 8×8 bit 无符号近似乘法器对分割后的乘数进行计算，最终生成部分乘积结果 P_1、P_2、P_3、P_4。在传统的递归调用方法中须累计调用 16 次 4×4 bit 近似乘法器，而采用本文提出的压缩递归调用方法只需调用 4 次 4×4 bit 近似乘法器，可减少重复调用时额外的 CPD 和资源消耗。在步骤 3) 中调用改进的近似加法器或者子积重组计算模块对部分乘积进行组合得到最终结果。两者原理已在上文给出，此处不再赘述。

根据上述步骤，在 Proposed8×8_1 的基础上，结合改进的近似加法器提出了基于 LUT 的 16×16 bit 无符号近似乘法器 Proposed16×16_1。在 Proposed8×8_2 的基础上，结合子积重组计算模块提出了基于 LUT 的 16×16 位无符号近似乘法器 Proposed16×16_2。由于负数在 FPGA 中以补码的形式表示，而有符号数的最高位为符号位，正数的原码、补码、反码一致[19]。基于上述补码原理，提出了基于 LUT 的 16×16 bit 有符号近似乘法器 Proposed16×16_3。

3　实验结果与分析

3.1　资源消耗实验方法

首先，采用文献[10-13]的方法将本文提出的 5 款不同类别的近似乘法器在 Vivado19.2 中进行布局布线。对面积、功耗、CPD 等参数进行静态测量后计算出 PDAP 参数值用来综合衡量乘法器的资源消耗并与对比文献中的同类型近似乘法器进行对比。PDAP 等于面积、功耗、CPD 三者乘积，其值越小则证明综合资源消耗越少。

其次，在动态测试中，测试平台选用以手写数字识别为功能的卷积神经网络，该卷积神经网络在上位机中进行训练，将偏置和参数量化为 16 bit 定点数部署在 FPGA 上并通过 HDMI 进行视频输出。在该平台上选用 MNIST 数据集[20]中数字 0~9 的手写数字图片各 50 张进行识别，数字图片以 1 s 为时间间隔进行滚动播放。在 1 min

内 FPGA 识别出图片上的数字即为识别成功，以此方法来计算识别成功次数、识别成功率和平均识别成功率。

最后，因卷积层在整个卷积神经网络中计算量占比较大，提高卷积层中乘法的运算效率可以直接提升加速器的整体性能[21]。因此将本文提出的基于 LUT 的 16×16 bit 的有符号近似乘法器封装为 IP 核后，在相同实验环境下部署在该卷积神经网络中乘法计算较为密集的卷积层中以代替 Multiplier IP 核。采用上述测试方法计算出识别成功次数、识别成功率和平均识别成功率并与之前结果进行比较。本文的开发平台为 Vivado19.2，FPGA 型号为 ZYNQ7000 系列的 xc7z020，摄像头型号为 OV5640。

3.2　精度实验方法

本文引入文献[10-13]用来量化评估精度的参数 ED、RED 和 MRED 在相同测试环境下进行仿真乘法实验。

误差距离 ED 是基本的误差度量方式，其值等于精确乘法结果 M 与近似乘法结果 M' 的距离，如式(6)所示。

$$ED = |M - M'| \tag{6}$$

RED 表示为相比于精确结果的相对距离，如式(7)所示。

$$RED = \sum_{i=1}^{2^{2n}} \frac{ED_i}{M_i} \tag{7}$$

最终以参数 MRED 来较为客观全面的评估近似乘法器的精准度，其值越小则表明误差越小[22]，如式(8)所示。

$$MRED = \frac{RED}{2^{2n}} \tag{8}$$

3.3　基于 LUT 的 8×8 bit 无符号近似乘法器测试

根据 3.1 与 3.2 节中的实验方法测得本文提出的两款基于 LUT 的 8×8 bit 无符号近似乘法器与前人同类型乘法器的相关数据，如表 2 所示。

为了更加客观的表示资源消耗量与精度之间的关系，采用文献[10-13]的方法绘制了 Pareto Optimal 图由图 7 所示。图 7 中在 Pareto Front 上的点证明其精度和资源消耗取得了较好的平衡，并且最靠近原点的点即为取得了最好的平衡[23]。

由图 7 可知，本文提出的 Proposed8×8_1 与 Proposed8×8_2 均在 Pareto Front 上。Proposed8×8_2 最靠近原点，证明其精度和资源消耗相较之下达到了最优平衡，Proposed8×8_1 资源消耗量相较之下最少，可弥补精度下降带来的损失。与文献[12]的 Ax8_1 相比其面积、功耗、CPD 可节省 60%、约 60.76%、约 25.4%。

表 2　基于 LUT 的无进位链 8×8 bit 无符号近似乘法器测试结果

Table 2　Test results of 8×8-bit unsigned approximate multiplier of uncarried chain based on LUT

方法	面积/LUTs	功耗/W	CPD/ns	MRED	PDAP
文献[10]	32	0.253	11.893	0.143	96.29
NCCA444[11]	44	0.574	11.897	0.108	302.75
MODA444[11]	45	0.696	14.588	0.065	456.90
ACCA444[11]	45	0.809	14.591	0.059	531.19
Ax8_1[12]	80	0.581	15.162	0.079	704.73
Ax8_2[12]	81	0.318	14.305	0.127	368.47
Ref[13]	44	0.574	11.987	0.108	302.75
Proposed8×8_1	**32**	**0.228**	**11.309**	**0.146**	**82.51**
Proposed8×8_2	**41**	**0.441**	**13.866**	**0.091**	**252.41**

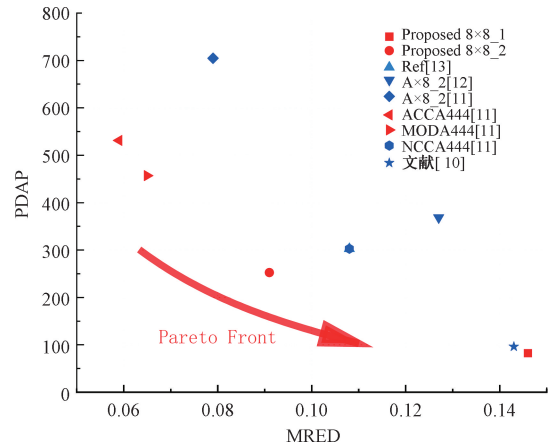


图 7　基于 LUT 的 8×8 bit 无符号近似乘法器 Pareto Optimal 图

Fig. 7　LUT-based 8×8-bit unsigned approximate multiplier Pareto Optimal diagram

3.4　基于 LUT 的 16×16 bit 近似乘法器整体测试

按照上述方法对 Proposed16×16_1、Proposed16×16_2、Proposed16×16_3 以及前人同类型的乘法器进行测试，如表 3 所示。

表 3　基于 LUT 的无进位链 16×16 bit 无符号近似乘法器测试结果

Table 3　Test results of 16×16-bit unsigned approximate multiplier of uncarried chain based on LUT

	面积/LUTs	功耗/W	CPD/ns	MRED	PDAP
HSLP_1134_16[13]	225	2.571	17.479	0.121	10 105.98
MODA_1334_16[13]	217	3.35	18.788	0.115	13 647.09
ACCA_1111_16[13]	245	3.994	20.359	0.001 3	19 912.11
NCCA_1134_16[13]	220	2.424	15.192	0.121	8 090.56
Ca_16x16[21]	245	3.574	10.312	0.112	9 029.49
Proposed16×16_1	**144**	**0.919**	**13.160**	**0.213**	**1 742.33**
Proposed16×16_2	**197**	**2.061**	**0.101**	**0.091**	**7 421.34**
Proposed16×16_3	**199**	**1.715**	**15.382**	**0.213**	**5 255.71**

根据表3中数据绘制了Pareto Optimal图,如图8所示。本文提出的Proposed16×16_1与Proposed16×16_2均在Pareto Front上,而Proposed16×16_3由于增加了有符号数计算模块因此与其他近似乘法器类型不同。结合表3与图8可知,Proposed16×16_2相较于其他同类型乘法器,其在资源消耗和精度上取得了最优平衡。相较之下,Proposed16×16_1实现了资源节约的最大化可弥补精度下降带来的损失。其最高可节省约41.2%的面积、约77%的功耗、约35.4%的CPD。

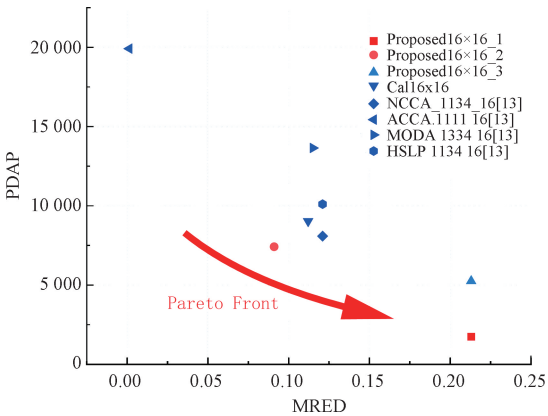


图8　基于LUT的16×16 bit近似乘法器Pareto Optimal图

Fig. 8　LUT-based 16×16-bit approximate multiplier Pareto Optimal diagram

3.5　动态测试

动态测试平台由FPGA开发板、上位机、HDMI显示屏、数字显示屏组成,如图9所示。

图9　实机测试平台

Fig. 9　Real machine testing platform

按照3.1节的实机测试方法分别在卷积层部署Proposed16×16_3、Multiplier IP核后进行数据测算。与部署了Multiplier IP核(Speed Optimized)的卷积层相比,面积减少了约23.7%,功耗下降了46.88%,卷积层中CPD增加了约9.3%。本文与部署了Multiplier IP核(Area Optimized)的卷积层相比,面积减少了约32.48%,功耗下降了41.21%,卷积层中最大CPD下降了约24.28%如表4所示。

表4　积层测试结果

Table 4　Convolutional layer test results

	卷积层面积	卷积层功耗	卷积层CPD
Multiplier IP (Speed Optimized)	4 013	0.239	12.383
Multiplier IP (Area Optimized)	4 538	0.216	17.881
Proposed16×16_3	**3 064**	**0.127**	**13.54**

对在卷积层部署Multiplier IP核的卷积神经网络识别率进行测试。经过测试得到原卷积神经网络各数字的识别率,并计算出平均识别率为90.2%,如表5所示。

表5　原卷积神经网络测试数据

Table 5　Original convolutional neural network test data

数字	测试次数	识别成功次数	识别成功率/%	平均成功识别率/%
0	50	48	96	90.2
1	50	49	98	
2	50	48	96	
3	50	35	70	
4	50	47	94	
5	50	46	92	
6	50	45	90	
7	50	40	80	
8	50	48	96	
9	50	45	90	

对部署了本文提出的Proposed16×16_3的卷积神经网络进行测试。测得部署后卷积神经网络各数字的识别率并计算出平均识别率为86.8%,仅下降3.4%,如表6所示。

表6　部署后的卷积神经网络测试数据

Table 6　Deployed convolutional neural network test data

数字	测试次数	识别成功次数	识别成功率/%	平均成功识别率/%
0	50	45	90	86.8
1	50	47	94	
2	50	47	94	
3	50	33	66	
4	50	46	92	
5	50	43	86	
6	50	45	90	
7	50	39	78	
8	50	45	90	
9	50	44	88	

由以上实验证明,本文提出的Proposed16×16_3在精度损失较小的情况下,带来的资源节省量明显。

4 结　论

本文针对 FPGA 加速卷积神经网络时资源消耗量大、加速不彻底的核心问题,结合压缩递归调用、优化关键路径、子积重组计算、有符号数判断模块等技术方法,提出了 5 款适用于不同场景、各有优势的基于 LUT 的近似乘法器。这些乘法器可以较好的满足 FPGA 加速卷积神经网络的需求并在精度与资源开销达成最优平衡的条件下最高可节省 60% 的面积、约 60.76% 的功耗、约 25.4%的 CPD。本研究不仅对硬件加速领域具有一定的参考价值,还为近似计算领域做出了一定的贡献。未来的工作将进一步拓展近似乘法器的适用性,在能耗与精度的动态平衡方面进行更为深入的研究。

参考文献

[1] 杨永杰, 郑君泰, 马立, 等. 一种改进型 LeNet 的交通标识多分类异构加速器的实现[J]. 北京大学学报(自然科学版), 2024,60(6):1001-1008.
YANG Y J, ZHENG J T, MA L, et al. Implementation of an improved LeNet traffic sign multi-classification heterogeneous accelerator [J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2024, 60(6): 1001-1008.

[2] 李放, 曹健, 李普, 等. 基于 ARM+FPGA 异构平台的目标检测加速模块设计与实现[J]. 北京大学学报(自然科学版), 2022,58(6):1035-1041.
LI F, CAO J, LI P, et al. Design and implementation of object detection acceleration module based on an ARM+FPGA heterogeneous platform [J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2022, 58(6): 1035-1041.

[3] 缪丹丹, 张鹏, 张鑫宇, 等. 基于 ZYNQ 平台的通用卷积加速器设计[J]. 国外电子测量技术, 2022, 41(11):72-77.
MIAO D D, ZHANG P, ZHANG X Y, et al. Generalized convolutional accelerator design based on ZYNQ platform[J]. Foreign Electronic Measurement Technology, 2022, 41(11):72-77.

[4] LOU W, GONG L, WANG C, et al. OctCNN: A high throughput FPGA accelerator for CNNs using octave convolution algorithm [J]. IEEE Transactions on Computers, 2022,71(8):1847-1859.

[5] 蒋康宁, 周海, 卞春江, 等. 基于宇航级 FPGA 的 YOLOv5s 网络模型硬件加速[J]. 空间科学学报, 2023,43(5):950-962.
JIANG K N, ZHOU H, BIAN CH J, et al. Hardware acceleration of YOLOv5s network model based on aerospace-grade FPGA [J]. Chinese Journal of Space Science, 2023, 43(5): 950-962.

[6] 陈昌川, 王新立, 朱嘉琪, 等. 基于卷积神经网络的岩渣分类算法及其 FPGA 加速[J]. 传感技术学报, 2024,37(1):80-88.
CHEN CH CH, WANG L X, ZHU J Q, et al. Rock slag classification algorithm based on convolutional neural network and its FPGA acceleration [J]. Chinese Journal of Sensors and Actuators, 2024, 37(1): 80-88.

[7] 黄沛昱, 赵强, 李煜龙. 基于 FPGA 的卷积神经网络硬件加速器设计[J]. 计算机应用与软件, 2023, 40(3):38-44.
HUANG P Y, ZHAO Q, LI Y L, et al. Design of FPGA-based convolutional neural network hardware accelerator [J]. Computer Applications and Software, 2023, 40(3): 38-44.

[8] ZHANG J, CHENG L, LI C, et al. A low-Latency FPGA implementation for real-time object detection[C]. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021:1-5.

[9] 戴伟杰, 王衍学, 李昕鸣, 等. 面向 FPGA 部署的改进 YOLO 铝片表面缺陷检测系统[J]. 电子测量与仪器学报, 2023,37(9):160-167.
DAI W J, W Y X, LI X M, et al. YOLO aluminum profile surface defect detection system for FPGA deployment [J]. Journal of Electronic Measurement and Instrumentation 2023,37(9):160-167.

[10] ULLAH S, MURTHY S S, KUMAR A, et al. SMApproxLib: Library of FPGA-based approximate multipliers[C]. 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), 2018:1-6.

[11] YAO S, ZHANG L. Hardware-efficient FPGA-based approximate multipliers for error-tolerant computing[C]. 2022 International Conference on Field-Programmable Technology (ICFPT), 2022:1-8.

[12] GUO Y, ZHOU Q, CHEN X, et al. High-efficiency FPGA-Based approximate multipliers with LUT sharing and carry switching[C]. 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2024: 1-2.

[13] WARIS H, WANG C H, LIU W Q, et al. Hybrid partial product-based high-performance approximate recursive multipliers[J]. IEEE Transactions on Emerging Topics in Computing, 2022,10(1):507-513.

[14] GUO Y, ZHOU Q L, CHEN X, et al. Hardware-efficient multipliers with FPGA-based approximation for error-resilient applications[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2024, 71(12):

5919-5930.

[15] 王壮. 一款 FPGA 可编程逻辑块的设计[D]. 西安:西安电子科技大学, 2023.
WANG ZH. Design of FPGA programmable logic block [D]. Xi′an: Xidian University, 2023.

[16] YAO S, ZHANG L. FHAM: FPGA-based high-efficiency approximate multipliers via LUT encoding [C]. 2022 IEEE 40th International Conference on Computer Design (ICCD), 2022:487-490.

[17] HADDADI I, QIQIEH I, SHAFIK R, et al. Run-time configurable approximate multiplier using significance-driven logic compression [C]. 2021 IEEE 39th International Conference on Computer Design (ICCD), 2021:117-124.

[18] ULLAH S, REHMAN S, PRABAKARAN S B, et al. Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators [C]. 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), 2018:1-6.

[19] 冯桂义. 基于 FPGA 的自适应滤波器研究[D]. 桂林:桂林电子科技大学, 2023.
FENG G Y. Research on adaptive filter based on field programmable gate array [D]. Guilin: Guilin University Of Electronic Technology, 2023.

[20] 邓诗卓, 滕达, 李晓红, 等. 基于球面正则化的支持向量描述视觉异常检测[J]. 仪器仪表学报, 2024, 45(3):315-325.
DENG SH ZH, TENG D, LI X H, et al. Spherical regularized support vector description for visual anomaly detection [J]. Chinese Journal of Scientific Instrument, 2024,45(3):315-325.

[21] 刘谦,王林林,周文勃. 基于 FPGA 的 YOLOv5s 网络高效卷积加速器设计[J]. 电讯技术, 2024, 64(3):366-375.
LIU Q, WANG L L, ZHOU W B. Design of a YOLOv5s network efficient convolution accelerator powered by FPGA [J]. Telecommunication Engineering, 2024, 64(3):366-375.

[22] ULLAH S, REHMAN S, SHAFIQUE M, et al. High-performance accurate and approximate multipliers for FPGA-based hardware accelerators [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022,41(2):211-224.

[23] STROLLO A G M, NAPOLI E, DE CARO C, et al. Approximate multipliers using static segmentation: Error analysis and improvements [J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022,69(6): 2449-2462.

作者简介

杨宏浩,2023 年于枣庄学院获得学士学位,现为大连海洋大学硕士研究生,主要研究方向为 FPGA 与近似容错计算。
E-mail: yhhinsmooth@163.com

Yang Honghao received his B. Sc. degree from Zaozhuang University in 2023. Now he is a M. Sc. candidate in Dalian Ocean University. His main research interests include FPGA and Approximate fault-tolerant computation.

李响(通信作者),分别在 2006 年和 2009 年于沈阳工业大学获得学士学位和硕士学位,2020 年于大连理工大学获得博士学位,现为大连海洋大学副教授,主要研究方向为图像处理。
E-mail: lixiang@dlou.edu.cn

Li Xiang (Corresponding author) received his B. Sc. degree and M. Sc. degree from Shenyang University of Technology in 2006 and 2009, and Ph. D. degree from Dalian University of Technology in 2020, respectively. Now he is an associate professor at Dalian Ocean University. His main research interest includes Image processing.