DOI: 10.16452/j.cnki.sdkjzk.2025.02.011 文章编号:1672-3767(2025)02-0104-11

应用于 eFPGA 的乘加运算单元设计

李春锋[1],卢丽珍[2,3],余 彬[2,3],舒 毅[3,4],范 迪[2]

(1. 河南天通电力有限公司,河南 平顶山 467000;2. 山东科技大学 电子信息工程学院,山东 青岛 266590;
3. 山东产研集成电路产业研究院有限公司,山东 济南 250102;4. 山东芯慧微电子科技有限公司,山东 济南 250102)

摘 要:针对当前嵌入式可编程逻辑阵列(eFPGA)中实现神经网络模型时资源利用率低的问题,提出一种新型乘加运算单元设计结构,以提升乘加单元资源利用率,充分发挥 eFPGA 高空间并行性。乘加运算单元在保留传统 eFPGA 的数字信号处理单元核心乘加功能基础上,增加了对常用 INT8/16/32 量化位宽数据的单指令多数据 SIMD 运算结构支持,并对位宽扩展后的部分积生成器、压缩树分割方法及并行前缀加法器结构进行了优化,以降低核心乘加单元通路延迟。乘加运算单元采用 UMC 28 nm 工艺实现,仿真与实验结果表明,乘加单元满足功能正确性要求,在神经网络应用测试电路综合结果上的资源利用率提升 1.37~3.05 倍。

关键词:嵌入式可编程逻辑阵列;数字运算单元;乘加器;Booth 算法

中图分类号:TP332.2 **文献标志码**:A

Multiplication and addition operation unit design for eFPGA

LI Chunfeng[1], LU Lizhen[2,3], YU Bin[2,3], SHU Yi[3,4], FAN Di[2]

(1. Henan Tiantong Electric Power Co., Ltd., Pingdingshan 467000, China; 2. College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China;
3. Shandong Industry Institute of Integrated Circuit Industry Co., Ltd., Jinan 250102, China;
4. Shandong Cwise Microelectronics Technology Co., Ltd., Jinan 250102, China)

Abstract: The low resource utilization has become a prominent problem when neural network models are implemented in embedded programmable gate array (eFPGA). To solve this problem, this paper proposes a new design structure of multiplication and addition operation unit to improve the resource utilization rate of the multiplication and addition unit and give full play to the high spatial parallelism of eFPGA. Retaining the core multiplication and addition functions of digital signal processing unit in eFPGA, this design adds single instruction multiple data operation structure support for commonly used INT8/16/32 quantifying bit width data and optimizes the partial product generator, the compressed tree segmentation method and the parallel prefix adder structure of the design after bit width expansion to reduce the path delay of the core multiplication and addition unit. This design is implemented using the UMC 28 nm process. Simulation and experimental results show that the multiplication and addition unit meets the requirements of functional correctness and the resource utilization rate is improved by 1.37~3.05 times in the comprehensive results of the neural network application test circuit.

Key words: eFPGA; digital arithmetic unit; multiplier; Booth algorithm

近年来,嵌入式可编程逻辑阵列(embedded field programmable gate array, eFPGA)因其灵活的可重构

收稿日期:2024-05-05

基金项目:国家语委“十三五”科研规划项目(YB135-125)

作者简介:李春锋(1978—),男,河南周口人,高级工程师,主要研究方向为煤矿供电安全、供配电管理.

范 迪(1974—),女,河南南阳人,教授,博士,主要从事图像处理、机器视觉等研究,本文通信作者.

E-mail:fandi_93@126.com

性、丰富的片上逻辑资源和较低的系统重构成本[1],被逐步集成到不同领域的专用芯片中。同时,随着人工智能技术的蓬勃发展,eFPGA 作为一种高能效的计算加速方案,也被越来越多地应用到专用边缘芯片[2]和大规模人工智能加速芯片[3]等领域,并逐步成为高能效加速器设计的潜在方案之一。传统 eFPGA 结构中的专用数字信号处理单元(digital signal processing, DSP)是面向数字信号处理等领域中乘法累加运算优化设计的,面向当前人工智能领域神经网络推理应用时,传统 eFPGA 结构会导致较低的逻辑资源利用率和能效比[4],无法发挥出 eFPGA 的高度并行性。此外,eFPGA 的资源有限,在有限资源内高效使用神经网络模型具有很大挑战。因此,在保留传统 eFPGA 中 DSP 单元核心功能基础上,针对当前人工智能领域应用需求,改进 DSP 单元设计,提高应用电路的资源利用率,发挥 eFPGA 的高空间并行性,具有重要的实用价值。

由于卷积神经网络模型自身的复杂性和 eFPGA 的硬件特性,面向 eFPGA 实现神经网络模型推理时要考虑硬件资源和数据精度的限制,需要将神经网络模型分成多个平铺块并对数据进行量化处理[5]。神经网络模型推理的大部分计算都在卷积层,但卷积计算中乘法计算的位宽比 eFPGA 中处理器的计算位宽要小,执行乘法计算使用单个 DSP 会造成严重的资源浪费[6],因此在 eFPGA 上实现神经网络模型,需要提升其乘加计算能力来提高资源利用率。Ullah 等[7]认为在神经网络推理中乘法器是影响延迟和资源利用率的主要因素,设计了精确带符号乘法器架构,用较长的延时换取面积的节省,但没有取得大的空间并行性;Zheng 等[8]为把卷积神经网络部署到边缘设备以提高并行计算度,提出改进的 Baugh-Wooly 乘法器,可在一个周期内处理两个 4 位乘法计算,但模型量化为 4 时会造成精度的损失;Reddy 等[9]研究了多种不同数字相乘的技术,提出一种系数矩阵乘法器的全新算法,提高了可编程逻辑阵列(field programmable gate array,FPGA)处理器的效率,但由于矩阵乘法器自身延迟大的特性,该算法会增加模型推理时间。Khalil 等[10]针对 FPGA 实现卷积神经网络推理的低功耗设计,提出基于卷积过程的可用资源流水线设计以及乘法累加运算的共享处理。Tang 等[11]提出面积高效的并行乘法单元设计,减少 FPGA 实现模型推理时的硬件开销。梅其昌[12]为提升 FPGA 中 DSP 的运算能力,在单个 DSP 上实现双 8 bit 乘法运算,并扩展 16 个 DSP 级联操作。综上所述,面向 eFPGA 实现卷积神经网络推理时,可根据目标模型保留算法必要的逻辑,消除冗余信息,获得更高效率,在这个过程中优化乘/加法器设计有着重要的作用。

为提高 eFPGA 在人工智能及相关领域应用电路的资源利用率,本研究提出适用于 eFPGA 实现神经网络模型推理的乘加运算单元设计:

1) 针对卷积计算中的大量乘法计算,优化乘法器,设计适用于单指令多数据(single instruction multiple data, SIMD)处理结构的 Booth 编码器,优化部分积(partial products, PP)生成电路、取反加 1 电路和扩展位符号电路,同时改进共享分割压缩树方法,提高乘法器并行计算能力;

2) 针对卷积计算中的大量加法计算,设计基于并行前缀的进位选择加法器,采用分组的方法减少并行前缀算法逻辑深度,同时实现 SIMD 功能;

3) 整个乘加运算单元支持多个数据通路和不同功能的运算,并支持 SIMD 结构。针对不同量化数据位宽,可实现单路 INT32、双路 INT16、四路 INT8 有符号数或无符号数的乘加计算,适用于实现不同量化大小位宽的卷积神经网络模型推理。

1 乘加运算单元整体结构

乘加单元整体架构如图 1 所示,主要功能模块包括 32 位预加器、32 位乘法器和 64 位加法器,均可实现 SIMD 计算。顶层单元支持四个操作数输入,其中 A、B 操作数可送入预加模块或乘法模块;C 可以作为单独输入加法器的操作数;D 作为同列其他乘加单元计算结果的级联输入,可以与本单元组成级联加法操作;整体乘加单元的输出反馈至本运算单元最后一级加法器的输入,构成累加功能。乘加运算单元各个通路以及实现的功能如表 1 所示。

2 改进的 SIMD Booth 编码算法

子字并行是将一个长的操作数划分为多个短操作数的操作,与 SIMD 结合可提高计算的并行性,实现高

效数据处理。子字并行的操作数如图 2 所示,输入位宽为 32 bit 的操作数,经过子字划分后,可成为两个 16 bit 或四个 8 bit 的子操作数,经过子字并行划分后的操作数使用 SIMD 结构运算器,可同时计算出多个结果,有效提高计算效率。

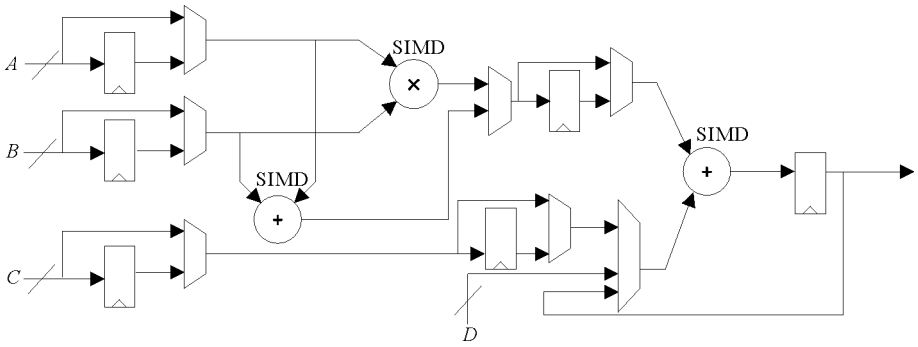


图 1 乘加运算单元整体结构

Fig. 1 Overall structure of the multiplication and addition unit

Radix-4 Booth 算法通过对二进制补码乘数进行编码来减少部分积的数量,每三位乘数生成选择信号,选择信号和被乘数产生部分积。乘数的位宽不同编码后得到的部分积数量不同,假设乘数的位宽为 m bit,当 m 为奇数时生成的部分积数量为 $(m+1)/2$,当 m 为偶数时生成部分积的数量为 $m/2+1$,经过 Booth 编码后生成的部分积数量有效减少。

表 1 乘加运算单元功能描述

Table 1 Description of the functions of the multiplication and addition unit

功能	描述
$Y=A\times B$	实现 A、B 乘法/SIMD 乘法器
$Y=A+B$	实现 A、B 预加计算
$Y=(A\times B)+C$	实现 A、B、C 乘加/SIMD 乘加计算
$Y=(A+B)+C$	实现 A、B、C 三输入加法计算
$Y=(A\times B)+D$	实现 A、B、D 乘加/SIMD 乘加计算
$Y=(A+B)+D$	实现 A、B、D 三输入加法计算

为保证不同计算模式下每个子操作数经过编码后的结果正确,需要消除 Booth 编码连续性对子字划分的影响,子操作数完成符号扩展后在其边界加入 0 进行分隔,可有效解决这个问题。如图 3 所示,16 bit 乘法不划分子字有 9 个部分积,均为有效部分积。划分两个 8 bit 子字计算时,两个子字经过符号扩展的操作数编码可生成 pp0~pp10 共 11 个部分积,其中 pp5 为边界分隔产生的无效部分积。加入分隔位后连续编码不会改变子字操作数编码的正确性,且不会影响其他模式下子操作数的编码。

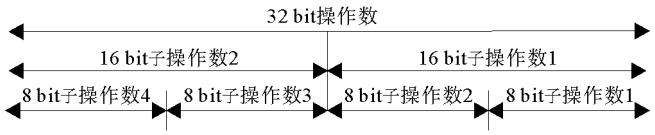


图 2 子字并行的操作数

Fig. 2 Number of operands of sub-word parallelism

2.1 新型部分积生成电路设计

为实现同时计算有/无符号数和共用编码器,需要对乘数进行符号扩展,乘数最高位扩展相同位宽,有符号数高位扩展符号位,无符号数高位补 0,以此保证编码后结果的正确性。

乘数重新编码生成三个选择信号 n、z、u 与被乘数生成部分积,如表 2 所示,生成的部分积有 0、A、$2A$、$-A$ 和 $-2A$ 五种情况。

将以上情况归类,优化后的部分积生成电路可以使用三个选择信号和被乘数通过译码电路得到部

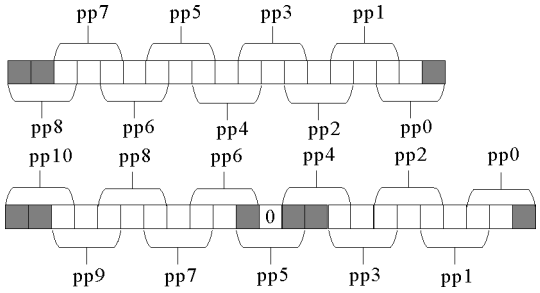


图 3 不同位宽子字的编码格式

Fig. 3 Encoding format for sub-word with different bit widths

分积。选择信号的生成逻辑表达式为:

$$n=M_{(i+1)}, \tag{1}$$

$$u=\overline{M_i \oplus M_{(i-1)}}, \tag{2}$$

$$z=(M_{(i+1)}+M_i+M_{(i-1)})\&\overline{(M_{(i+1)}\&M_i\&M_{(i-1)})}。 \tag{3}$$

式中,M_i 表示第 i 位乘数,"—"表示取反,其生成电路如图 4 所示。三位乘数中的最高位为 n 信号,有效时部分积取反;乘数低两位相同时,倍数信号 u 有效,此时部分积左移一位生成倍数;三位乘数相同时,清零信号 z 有效,此时生成的部分积为 0。三个控制信号的顺序优先级依次为 n、z、u。

选择信号与被乘数 D 生成部分积,其电路表达式为:

$$P=u\&(D\&z)+\overline{(u\&(M\&z)_{\text{shift}})}。 \tag{4}$$

部分积生成电路如图 5 所示,只需要输入被乘数与控制信号就能涵盖所有部分积生成情况。与文献[14]方法相比,本研究降低了输入信号的复杂性。首先 n 信号选择部分积为负的情况;n 为 1 时,被乘数与其异或取反,否则不变。然后,z 信号为 0 时部分积与之相与为 0,经过 n 和 z 筛选后的部分积通过 u 选择是否左移扩大倍数。最后,u 为 1 时部分积左移取倍数。

表 2 Radix-4 Booth 编码表

Table 2 Radix-4 Booth encoding

$B_{i+1}B_iB_{i-1}$	n	z	u	部分积
000	0	0	1	0
001	0	1	0	$1\times A$
010	0	1	0	$1\times A$
011	0	1	1	$2\times A$
100	1	1	1	$-2\times A$
101	1	1	0	$-1\times A$
110	1	1	0	$-1\times A$
111	1	0	1	0

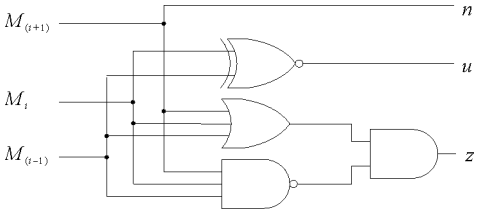


图 4 Booth 编码选择信号生成电路

Fig. 4 Generation circuit of Booth encoding select signals

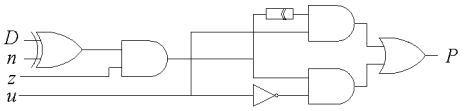


图 5 部分积生成电路

Fig. 5 Generation circuit of partial product

2.2 新型部分积阵列

为保证部分积累加结果正确,需要扩展部分积的符号位到与乘积结果相同长度再相加,8 bit Booth 乘法部分积的符号扩展如图 6 所示,其中 S 表示部分积的符号位。

图 6 中符号扩展方法占据较大的电路面积。改变部分积的符号扩展方式[12],可有效解决符号位过多的问题。图 7(a)中假设生成的部分积全为负的情况,相应符号扩展位全为 1;图 7(b)中将符号扩展的 1 保留进位按列相加。

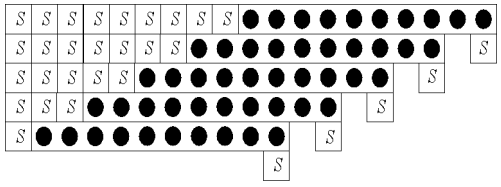
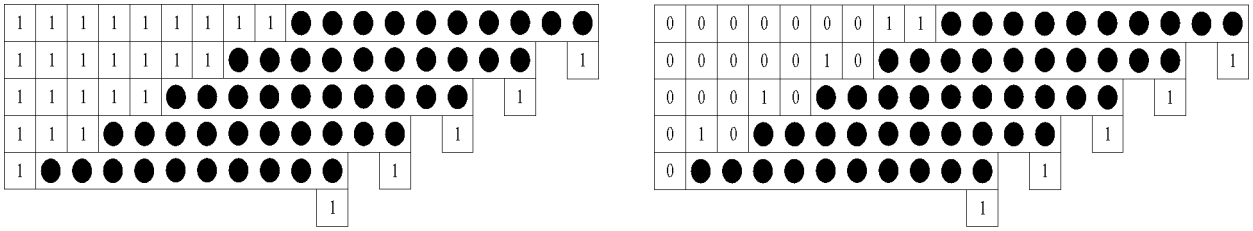


图 6 扩展符号位的 Booth 编码部分积阵列

Fig. 6 Partial product array of Booth encoding extended sign bit

图 8(a)中每个相加后符号扩展的末尾加 1,再次按列保留进位相加使部分积的扩展位全部变为 0,此时是部分积为正的符号扩展形式;图 8(b)中编码为负时,部分积取反末尾也有加 1 的操作。是否加 1 由部分积的符号 S 决定:部分积为正时 S 为 0,扩展位末尾加 1、部分积末尾加 0;部分积为负时 S 为 1,扩展位末尾加 0、部分积末尾加 1。

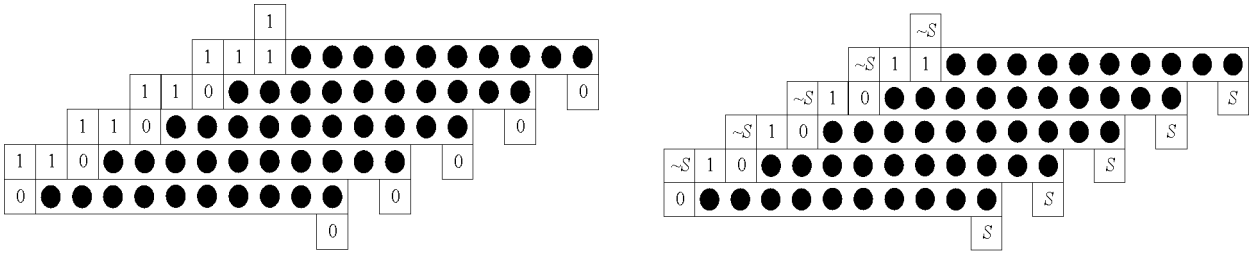
有限位符号扩展的最终格式如图 9 所示。原本 25 位的符号扩展位使用有限扩展后只有 10 位。



(a) 扩展位为1 Booth编码阵列　　(b) 扩展位按列相加Booth编码阵列

图 7　部分积为负的符号扩展阵列

Fig. 7　Sign-extended array with negative partial product



(a) 扩展位末尾加1 Booth编码阵列　　(b) 加入部分积符号的Booth编码阵列

图 8　部分积为正的符号扩展阵列

Fig. 8　Sign-extended array with positive partial product

由以上推导可得,有限符号位扩展的符号由部分积符号决定,本研究提出一种新型求解符号扩展的方法,其表达式为:

$$E=\overline{D_{\mathrm{msb}}\oplus n}+z。\tag{5}$$

式中:E 表示符号扩展位,D_{msb} 表示被乘数的最高位,有限符号位扩展电路如图 10 所示。该方法只需要用到被乘数最高位和 n、z 控制信号,在得到部分积时能同时获得扩展符号,有效减少部分积生成的复杂度和延时。

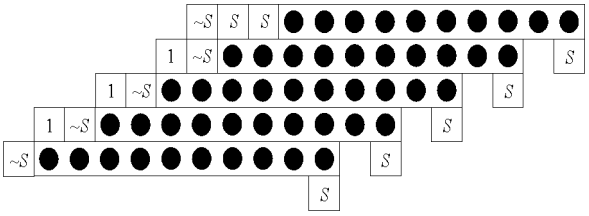


图 9　有限符号位扩展

Fig. 9　Limited sign bit extension

Booth 编码对被乘数生成$-1\times A$ 部分积时,需要对被乘数所有位数取反再加 1,生成$-2\times A$ 部分积时,有两次加 1 操作。对于加 1 操作的处理可将其单独作为一个部分积处理,或者将其置于下一部分积的低两位,可以减少部分积压缩的面积。

文献[14]优化编码器设计但未指出取反加 1 的处理,文献[15]中的求补加 1 电路未考虑到部分积为$-2\times A$ 的情况。本设计使用 2 bit 位宽表示取反加 1,分别使用 C_i 和 $C_{(i-1)}$ 表示,适用于部分积为$-1\times A$ 和$-2\times A$ 的情况。取反加 1 位与控制信号有关,具体电路实现如图 11 所示。

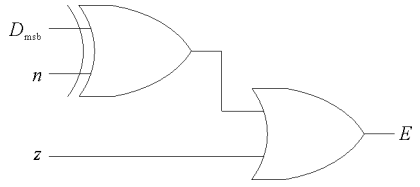


图 10　符号扩展位生成电路

Fig. 10　Generation circuit of sign-extended bit

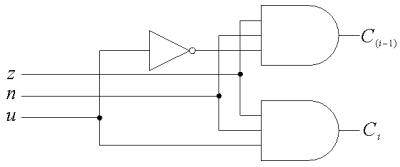


图 11　部分积取反加 1 位生成电路

Fig. 11　Generation circuit of partial product inverted by adding 1 bit

2.3　改进共享分割的压缩树结构

部分积压缩是乘法器设计的主要组成部分之一,由压缩器组成的华莱士树是最常见的压缩结构。在

SIMD 结构下,32 位操作数在划分子字位宽为 8 bit 时有 28 个部分积需要压缩。不同 SIMD 模式下子乘法器的部分积数量和位宽不同,若使用固定位宽和级数的压缩树实现所有计算模式共用,需要采用共享分割压缩树方式对子乘加器进行计算。基于共享分割的华莱士树结构如图 12 所示,P_0 表示子操作数 A_0 与 B_0 相乘得到的部分积。共享分割方法把子乘法器生成的部分积根据不同模式分配在不同位置,保证在每一级的压缩计算中各个子乘加器运算的结果互不干扰。在计算共享分割的子字并行乘加时,把不属于子字并行有效部分积的区域设置为 0[16]。

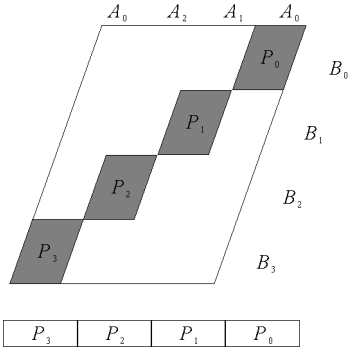


图 12 基于共享分割的华莱士树

Fig. 12 Wallace tree based on shared partitioning

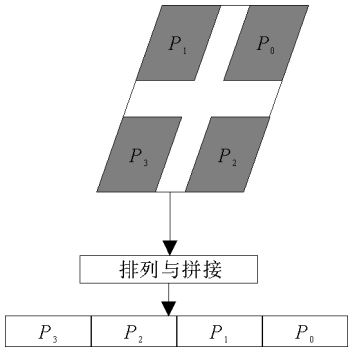


图 13 基于改进共享分割的华莱士树

Fig. 13 Wallace tree based on improved shared partitioning

然而,操作数位宽越长划分子字越多,压缩树中设置为 0 的无效区域越大。使用 Booth 编码得到部分积时,划分子字后得到的部分积总和大于不划分子字时的部分积数量,划分不同位宽的子字得到总部分积的数量不同,因此共享分割并不是 SIMD 结构 Booth 乘法器的最优划分方式。

如图 13 所示,在共享分割的基础上改进子乘法器的有效部分积分配方式,在固定位宽和固定压缩器的情况下灵活分配,多个子乘法器的部分积可以并行计算。这种分配方式可以减少压缩器使用个数和无效区域面积,提高计算的并行性。

华莱士树压缩计算部分压缩级数取决于部分积的数量,分配过程需要考虑不同乘法器之间部分积的相互影响。综合子乘法器划分个数以及不同子字生成的部分积数量,压缩树最大高度为 22 个部分积,使用 3-2 压缩器与改进的 4-2 压缩器[17]混合树型结构[15]压缩,压缩级数为 4 级。

2.4 基于并行前缀的进位选择加法器

并行前缀算法在提高加法运算速度上有优异表现,常见并行前缀算法结构有 Sklansy、Brent-Kung、Kogge-Stone 等,其中 Kogge-Stone 加法器(Kogge-Stone adder,KSA)的传播延迟最低,所有位宽的延迟基本恒定[18],并且 Kogge-Stone 结构有较小的逻辑深度和有界扇出,输入数据生成进位传播信号 P 和进位信号 G,P 和 G 按照 Kogge-Stone 递归方式排列成进位树,m bit 数经过 $\log_2 m$ 级计算就能得到所有进位。

在设计过程中,Kogge-Stone 加法器作为加法器主体,使用进位选择的思想提高运算速度。如图 14 所示,整体结构是基于并行前缀的进位选择加法器,输入数据 $X[31:0]$和 $Y[31:0]$经过 Kogge-Stone 网络生成进位选择网络,Kogge-Stone 加法器每 8 位一组,计算出进位信号分别为 0 和 1 的结果。进位网络生成的进位信号选择每组加法计算正确结果,最后得到进位 cout 和结果 S。在此过程中,进位选择网络与加法器可实现 Kogge-Stone 网络的复用,进位计算和结果计算同时进行,极大提高运算并行性。

由 Kogge-Stone 算法可知,加法器位宽过大时,使用的电路资源和延时会增加。如图 15 所示,采用分组的方法把 8 bit 加法器作为一组加法单元,可有效减少逻辑深度,降低电路复杂性。

基于并行前缀的进位选择加法器性能更快,面积效率更高[19]。文献[18]中提出进位选择与并行前缀结合,把行波进位加法器换为 Kogge-Stone 加法器,虽提高部分运算速度,但选择信号依旧是低位传到高位,存在一定延迟。如图 16 所示,将 64 位加法分为 8 组,把每组加法单元最高位的 PG 信号使用 Kogge-Stone 并行前缀网络计算,与初始进位信号生成每组加法单元的进位信号,此时的进位信号可以作为选择信号去选择下一组加法单元的正确结果。采用进位选择计算方式可以节省进位选择信号从低位向高位的传播时间。

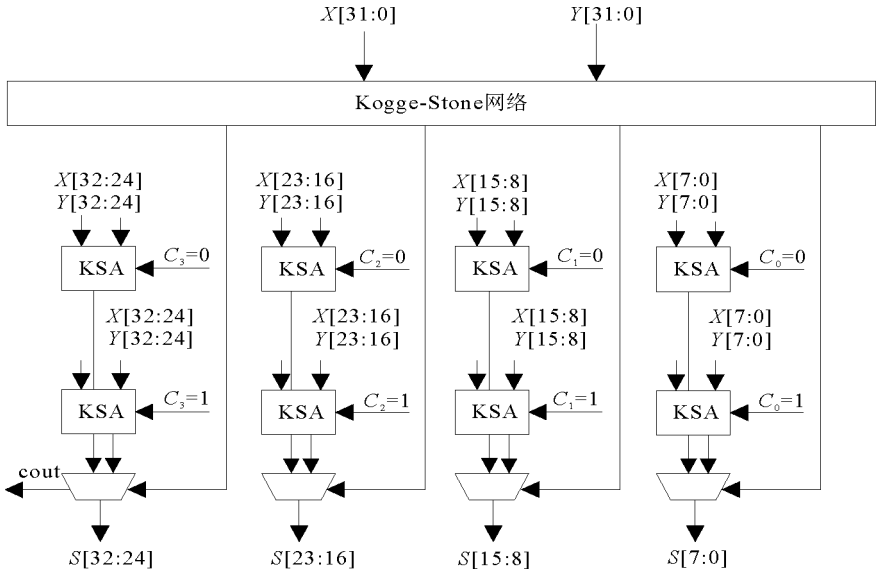


图 14 基于并行前缀的进位选择加法器

Fig. 14 Carry select adder based on parallel prefix

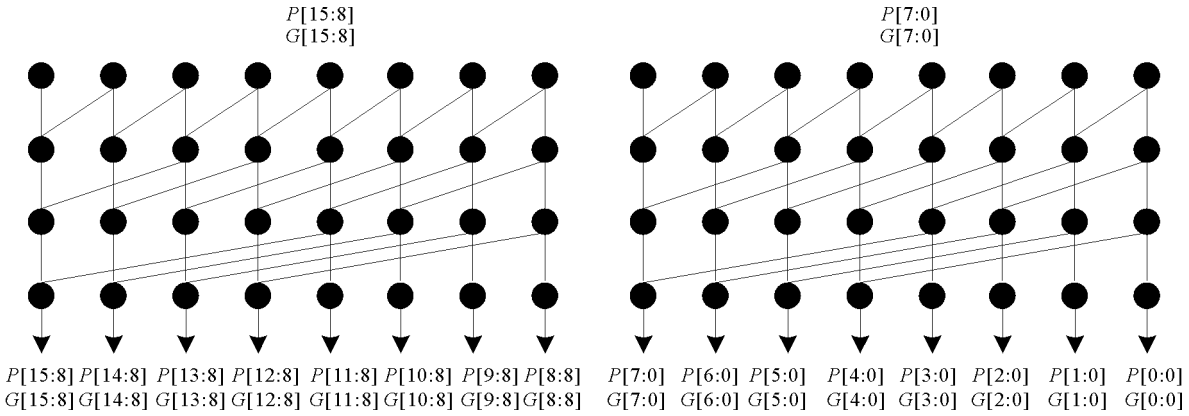


图 15 分组的 16 bit Kogge-Stone 加法器 PG 网络

Fig. 15 PG network of grouped 16 bit Kogge-Stone adder

为满足子字并行乘法器的输出格式,加法器同样采用 SIMD 结构,具体实现方式为:Kogge-Stone 加法器划分为加法单元后按照 SIMD 要求分组,划分子字为 8 bit 时,每组加法单元看作一个独立的子加法器;划分子字为 16 bit 时,2 组加法单元作为一个子加法器;划分子字为 32 bit 时,4 组加法单元为一个子加法器;不划分子字时,8 组加法单元为一个加法器。如图 17 所示,在进位选择网络上加入进位消除机制,根据计算模式把来自低位的进位传播信号设置为 0,在生成进位选择信号时把每个子字加法器的初始输入进位设置为本身信号,这样可以消除低位信号的影响,图 17 中的虚线部分表示消除的进位传播信号。

3 验证与分析

3.1 功能验证

乘加器使用 Verilog HDL 硬件描述语言设计,功能验证基于 Modelsim 软件,编写测试文件生成 32 位随机数模拟卷积运算数据,SIMD-16 模式仿真结果如图 18 所示,实现的功能是计算 2 路 16 bit 的 $A\times B+C$ 计算,左侧的信号栏为输入信号,其中 sub_a、sub_b、sub_c 分别为 16 bit 的子操作数,sub_res 为本研究中乘加器计算结果,mac_res 为调用乘加知识产权核(intellectual property,IP)计算结果,计算结果相同时 correct 信号输出为 1。后缀数字为 0 是低 16 位子操作数,后缀数字为 1 是高 16 位子操作数,右侧数据栏为仿真计算结果,通过 correct 信号拉高为 1 可知功能验证结果正确。

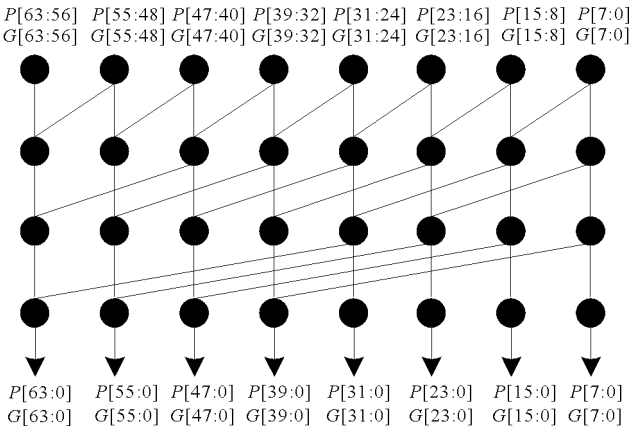


图 16 进位选择信号 PG 网络

Fig. 16 PG network of carry select signals

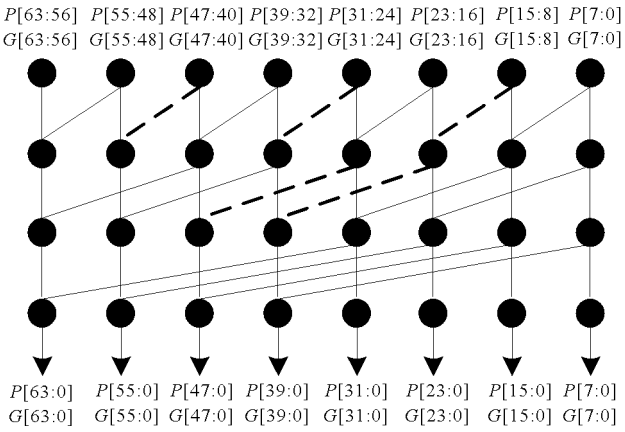


图 17 加入进位消除机制的 PG 网络

Fig. 17 PG network with carry elimination mechanism

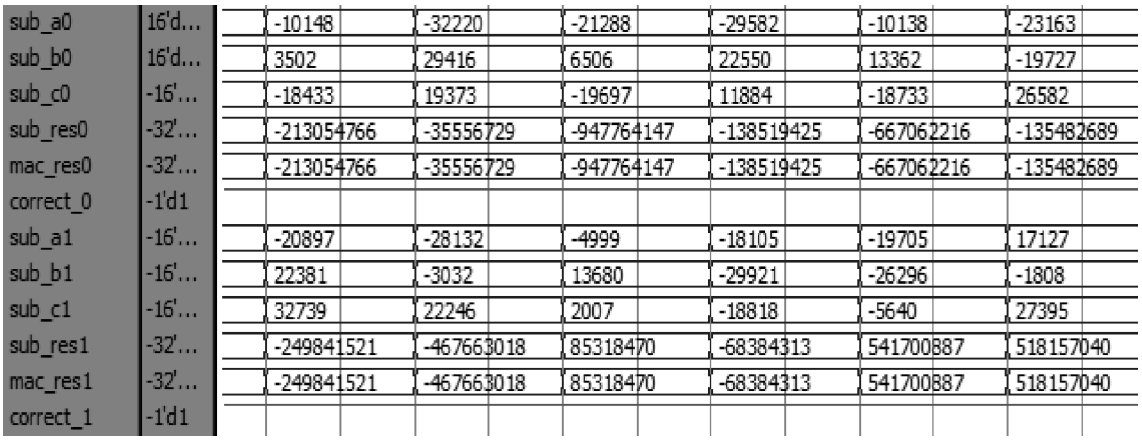


图 18 乘加器功能仿真

Fig. 18 Function simulation of the multiplier

3.2 资源利用率分析对比

性能仿真基于 UMC 28 nm 工艺库,使用 Synopsys 综合工具 Design Compiler 对乘加运算单元和 Xilinx Artix 系列的 DSP48E1 模块进行综合。DSP48E1 的乘法位宽为 25×18 bit,本研究中乘加运算单元将乘法位宽升级为 32 位。经过综合,DSP48E1 的面积为 5 655.89 μm^2,本设计面积为 6 191.36 μm^2,较 DSP48E1 增大 9.46%。

由于 Xilinx 开发软件 Vivado 自带的综合工具无法统计本研究中乘加单元在实现神经网络模型时的使用量,为证明本研究的有效性,使用开源综合工具 yosys 分别调用 Xilinx 的 DSP 和本研究中乘加运算单元综合神经网络模型。具体实现方法为:修改底层单元库文件将乘加运算单元加入其中,并修改相关映射库和脚本文件,测试 eFPGA 实现不同量化大小和不同卷积核尺寸的卷积神经网络(convolutional neural network,CNN)模型时所使用 DSP 和乘加运算单元的数量和资源利用率。把 DSP48E1 和本研究中乘加单元的面积比率分别定义为 1 和 1.09,定义乘加单元资源利用率为 r(r 等于 DSP 的面积比率与资源使用数之积除以乘加器的面积比率与资源使用数之积)。测试模型电路的参数以及综合后的结果如表 3 所示。

结果表明,在实现不同量化位宽和不同大小模型时乘加运算单元的资源利用率均大于 DSP 的。使用 DSP 实现神经网络模型,需要空间并行架构组成大的 MAC 阵列,由于本研究中乘加运算单元采用 SIMD 结构,可进一步提升空间并行性,因此使用的资源数量明显下降。推理量化大小为 INT8 的模型时,资源使用数分别减少11和28。CNN模型2的规模大于CNN模型1,资源使用量减少更加明显,资源利用率是使用

DSP 计算的 3.05 倍。CNN 模型 3 的量化位宽为 INT16,由于规模较小未体现较大优越性,但资源利用率仍是 DSP 的 1.37 倍。

为进一步说明乘加运算单元在提升空间并行性的有效性,使用规模较大的视觉几何组(visual geometry group,VGG)模型进行应用模拟测试,对本研究进行评估。VGG16 是卷积神经网络常用的网络结构,几乎包含神经网络模型最基本的操作,由 13 个卷积层和 3 个全连接层组成,每层卷积后经过批量归一化层和激活函数。卷积层共分为 5 组,每组卷积层后执行池化操作。根据本研究中乘加运算单元的位宽设计,将所有数据和参数量化为 8 位二进制补码定点数,为充分利用电路资源,对模型进行拆分,拆分后得到网络的数据、权重和计算次数如表 4 所示。

经过拆分后的模型同样使用 yosys 调用 DSP 资源和本研究中乘加运算单元,并模拟识别 CIRFAR-10 数据集图片的过程,经过综合后的资源使用与其他相关设计的对比分析如表 5 所示。

本研究使用乘加运算单元代替 DSP 实现卷积神经网络模型,可充分发挥 eFPGA 空间并行能力。卷积神经网络模型的量化位宽为 INT8 时,节省资源效果明显。本研究方法与文献[8]方法相比有更高的工作频率,付出的面积代价更小并获得更高的能效比。本研究方法与文献[6]和文献[20]方法相比,在模拟识别同样大小数据集图片时,使用乘加运算单元的数量远小于使用 DSP 的数量,面积是 DSP 的 1.09 倍,能效比为 2.32 倍。在相同实现场景下乘加运算单元的资源利用率均大于 DSP,证明本研究应用于 eFPGA 可有效提高卷积神经网络的能力。

表 3 不同模型使用 DSP 与乘加单元资源对比

Table 3 Comparison of DSP and multiplication and addition unit resources used by different models

神经网络模型	CNN 模型 1	CNN 模型 2	CNN 模型 3
量化位宽	8	8	16
通道数	3	3	1
卷积矩阵尺寸	5×5	28×28	10×10
卷积核大小	3×3	5×5	3×3
DSP 使用数量	16	40	3
乘加单元使用数量	5	12	2
资源利用率	2.93	3.05	1.37

表 4 经过拆分后的 VGG16 模型结构

Table 4 Structure of VGG16 model after splitting

层数	数据大小	权重大小	计算次数
conv1_1	224×224×3	3×3×3	1×16
conv1_2	224×224×4	3×3×3	16×16
pool1	224×224×4	0	16
conv2_1	112×112×16	3×3×16	4×8
conv2_2	112×112×16	3×3×16	8×8
pool2	112×112×16	0	8
conv3_1	56×56×64	3×3×64	2×4
conv3_2-3	56×56×64	3×3×64	4×4
pool3	56×56×64	0	4
conv4_1	28×28×256	3×3×256	1×8
conv4_2-3	28×28×256	3×3×256	2×8
pool4	28×28×256	0	2
conv4_1-3	14×14×512	3×3×512	1×8
pool5	14×14×512	0	1
fc1	7×7×512	16×7×7	256
fc2	4 096	128×4 096	32
fc3	4 096	125×4 096	8

表 5 VGG-16 模型推理资源使用对比

Table 5 Utilization comparison of inference resources in VGG-16 model

设计	工作频率/MHz	面积比率	资源使用数量	资源利用率
文献[8]	333	1.57	—	2.00
文献[20]	100	—	471(DSP)	1.63
文献[6]	200	—	451(DSP)	1.73
本研究	400	1.09	309(乘加单元)	2.32

3.3 同类型乘加器设计性能对比

本研究在乘加器优化方面提出新型部分积生成器,采用改进共享分割的方法实现华莱士树压缩,最终部分积相加采用基于并行前缀的进位选择加法器。为证明乘加器优化的充分性,分别对 SIMD 模式乘加器与不同计算位宽的非 SIMD 乘法器进行建模,在 UMC 28 nm 工艺下使用 Design Compiler 对其进行性能仿真,工作频率设为 1 GHz。经过综合验证,本研究设计中所有关键路径均满足时序要求,未出现时序违规情况,本研究与其他同类型设计的性能参数对比如表 6 所示。

表 6 同类型设计性能参数

Table 6 Performance parameters for designs of the same type

部件类型	延时/ns	面积/μm^2	功耗/μW
文献[17]乘加器	1.27	—	2 443.00
文献[14]乘加器	0.98	1 858	—
文献[21]乘加器	0.49	—	108.98
本研究 SIMD 乘加器	0.98	6 191	643.93
本研究 8 位乘加器	0.45	819	111.14
本研究 16 位乘加器	0.62	1 748	186.15
本研究 32 位乘加器	0.97	5 226	561.05

与 32 位乘加器相比,本研究中加入 SIMD 功能的 32 位乘加器面积仅增加 18%。与文献[17]和文献[14]相比,本研究中 16 位乘法器速度分别提升 51%和 36%,面积降低 5%。本研究设计的 8 位乘法器与文献[21]中 8 位乘法器相比,速度提升 8%。本研究设计的乘加器在延时与面积方面有明显的优化与提升。

4 结论

本研究面向 eFPGA 设计高性能乘加运算单元代替 DSP,提高推理卷积神经网络模型的能力。为提高数据计算能力,乘法器和加法器均加入 SIMD 功能,可同时执行多个计算,根据子字划分结构设计通用 Booth 编码器,并优化部分积生成器设计,部分积、取反加 1 与扩展符号可同时生成,极大减少部分积生成的延时。改进共享分割的压缩树结构,加法器设计采用分组的并行前缀算法提高计算并行性、减小逻辑深度,极大提高计算并行性和计算能力。实现卷积神经网络模型时,本研究设计的乘加运算单元与 DSP 相比,资源利用率最大可达 DSP 的 3.05 倍、实际应用中可达 2.32 倍,提高了 eFPGA 计算卷积神经网络模型的能力。与同类型设计相比,面积与延时有明显优化,乘加器性能得到有效提升。在本研究基础上,将定点乘加运算的设计方法推广到浮点乘加计算,优化浮点乘加的尾数处理方法,可进一步提高计算精度和准确率。

参考文献:

[1] BOUTROS A,BETZ V. FPGA architecture:Principles and progression[J]. IEEE Circuits and Systems Magazine,2021,21(2):4-29.

[2] VÉSTIAS M. Processing systems for deep learning inference on edge devices[J]. Convergence of Artificial Intelligence and the Internet of Things,2020,5(7):213-240.

[3] MEDINAL,CARRION S,ANDREU P,et al. The SELENE deep learning acceleration framework for safety-related applications[C]//2022 Design,Automation & Test in Europe Conference & Exhibition(DATE). Antwerp,2022:636-639.

[4] RASOULINEZHAD S R,ZHOU H,WANG L,et al. PIR-DSP:An FPGA DSP block architecture for multi-precision deep neural networks[C]//2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines(FCCM). San Diego,2019:35-44.

[5] LI X M,HUANG H M,CHEN T S,et al. A hardware-efficient computing engine for FPGA-based deep convolutional neural network accelerator[J]. Microelectronics Journal,2022,128:1-7.

[6] MA T,LI Z W,LI Q J,et al. FPGA optimized accelerator of DCNN with fast data readout and multiplier sharing strategy[J]. Computers,Materials & Continua,2023,77(3):3237-3263.

[7] ULLAH S,NGUYEN T D A,KUMAR A. Energy-efficient low-latency signed multiplier for FPGA-based hardware accelerators[J]. IEEE Embedded Systems Letters,2020,13(2):41-44.

[8] ZHENG Y H,LI Z F,SUN K H,et al. A 40nm area-efficient Effective-bit-combination-based DNN accelerator with the

reconfigurable multiplier[C]//2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS). Hangzhou,2023:1-5.

[9] REDDY N N,SUBASH G,HEMADITYA P,et al. Low power and efficient re-configurable multiplier for accelerator[J]International Journal of Computer Communication and Informatics. 2022,4(2):1-11.

[10] KHALIL K,KUMAR A,BAYOUMI M. Low-power convolutional neural network accelerator on FPGA[C]//2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems(AICAS). Hangzhou,2023:1-5.

[11] TANG S N. Area-efficient parallel multiplication units for CNN accelerators with output channel parallelization[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems,2023,31(3):406-410.

[12] 梅其昌. 基于 FPGA 卷积神经网络加速器研究及应用[D]. 南京:南京邮电大学,2023.
MEI Qichang. Design and application of convolutional neural network hardware accelerator based on FPGA[D]. Nanjing: Nanjing University of Posts and Telecommunications,2023.

[13] ANGEL E D,SWARTZLANDER E E. Low power parallel multipliers[C]//VLSI Signal Processing. San Francisco,1996: 199-208.

[14] 王佳乐,胡越黎. 基于新型 booth 选择器和压缩器的乘法器设计[J]. 微电子学与计算机,2020,37(3):5-8.
WANG Jiale,HU Yueli. Multiplier design based on the new booth selector and compressor[J]. Microelectronics & Computer,2020,37(3):5-8.

[15] 石敏,王耿,易清明. 基于改进的 Booth 编码和 Wallace 树的乘法器优化设计[J]. 计算机应用与软件,2016,33(5):13-16.
SHI Min,WANG Geng,YI Qingming. An optimised design of multiplier based on improved Booth encoding and Wallace tree[J]. Computer Applications and Software,2016,33(5):13-16.

[16] KRITHIVASAN S,SCHULTE M J. Multiplier architectures for media processing[C]//The Thirty-Seventh Asilomar Conference on Signals,Systems & Computers,2003. Pacific Grove,2003:2193-2197.

[17] 仲亚,叶瑶瑶. 基于新型压缩器的乘法器设计[J]. 微电子学与计算机,2019,36(3):28-31.
ZHONG Ya,YE Yaoyao. Design of multiplier based on new compressor[J]. Microelectronics & Computer,2019,36(3): 28-31.

[18] CHAKALI P,PATNALA M K. Design of high speed Kogge-stone based carry select adder[J]. International Journal of Emerging Science and Engineering(IJESE),2013,1(4):34-37.

[19] XIANG L M,ZABIDI M M A,AWAB A H,et al. VLSI implementation of a fast Kogge-stone parallel-prefix adder[C]// Journal of Physics:Conference Series. Johor,2018:1-10.

[20] PANG W,WU C G,LU S L. An energy-efficient implementation of group pruned CNNs on FPGA[J]. IEEE Access, 2020,8:217033-217044.

[21] HAIDER M H,KO S B. Booth encoding based energy efficient multipliers for deep learning systems[J]. IEEE Transactions on Circuits and Systems II:Express Briefs,2023,70(6):2241-2245.

(责任编辑:齐敏华)