DOI:10.3979/j.issn.1673-825X.202406270158

基于脉动阵列架构的分布式计算 LSTM 加速器

张红升,成卓立
(重庆邮电大学 光电工程学院,重庆 400065)

摘　要:针对在资源有限的边缘计算端部署长短时记忆(long short-term memory,LSTM)神经网络遇到的计算效率低、功耗高的问题,提出一种基于脉动阵列架构的分布式计算 LSTM 加速器设计方案。通过将输入数据分布式存储,从而以减少数据的流动性并降低功耗;通过脉动的方式传递数据,从而减少计算单元的空置率并提高计算效率。在 VU13P 系列现场可编程门阵列(field programmable gate array,FPGA)的验证结果表明,所设计的 LSTM 加速器在 200 MHz 的工作频率下有效算力 179.2 GOPS,动态功耗 0.343 W,能效比 522.4 GOPS/W,相较于当前典型设计,能效比提升 34%以上。

关键词:长短时记忆(LSTM);现场可编程门阵列(FPGA);硬件加速器;脉动阵列
中图分类号:TN47　　**文献标志码**:A　　**文章编号**:1673-825X(2025)05-0741-07

Distributed computing LSTM accelerator based on pulsating array architecture

ZHANG Hongsheng, CHENG Zhuoli
(School of Optoelectronic Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, P. R. China)

Abstract: A long short-term memory (LSTM) neural network edge computing accelerator based on distributed systolic array architecture was proposed on the resource limited edge computing devices. The design distributes input data storage to reduce data movement and power consumption, while data transmission in a systolic manner minimizes the idle rate of computing units and enhances computational efficiency. Experimental validation on a VU13P field-programmable gate array (FPGA) shows that the proposed LSTM accelerator achieves an effective computing power of 179.2 GOPS at an operating frequency of 200 MHz, with a dynamic power consumption of 0.343 W and an energy efficiency of 522.4 GOPS/W. Compared with typical existing designs, the proposed accelerator improves energy efficiency by more than 34%.

Keywords: long short-term memory (LSTM); field-programmable gate array (FPGA); hardware accelerator; pulsating array

0　引　言

近年来,循环神经网络(recurrent neural network,RNN)因其优异的性能在自然语言处理(natural language processing,NLP)[1]、语音转换为文本[2]和视频分析处理[3]等应用场景中取得了显著成功。

收稿日期:2024-06-27　**修订日期**:2025-09-16　**通讯作者**:张红升　zhanghs@cqupt.edu.cn
基金项目:重庆市技术创新与应用发展专项重大项目(CSTB2023TIAD-STX0023)
Foundation Item: Major Project of Chongqing Technology Innovation and Application Development Special Project (CSTB2023TIAD-STX0023)

但由于 RNN 本身的结构特点,在进行大模型训练中存在长程依赖问题[4],导致难以处理长序列任务。长短时记忆(long short-term memory,LSTM)神经网络作为 RNN 最流行的变体之一,通过引入门控机制来控制信息的积累速度,能处理更加复杂的长序列任务场景[5],但同时也带了更大的计算量,所以如何对 LSTM 进行硬件加速引起了众多学者的研究。

LSTM 在边缘端的应用场景通常对硬件的计算速度和功耗有着较高的要求。中央处理器(central processing unit,CPU)作为指令流处理器,并行计算效率低,而图形处理器(graphics processing unit,GPU)的运行功耗较高,两者均难以满足边缘应用场景的要求。现场可编程门阵列(field programmable gate array,FPGA)由于其可编程、低功耗等特点,在 LSTM 硬件加速领域得到了人们的广泛关注。

目前基于 FPGA 的 LSTM 加速器研究可以分为 2 个方向。一是利用 LSTM 模型的数据稀疏性,通过算法减少其权重参数,以降低模型的计算量和功耗。其中,高琛等[6]基于 Delta 网络算法,对输入序列的稀疏性进行构建,在避免数据不规则加载的前提下,对冗余矩阵向量乘法运算进行过滤,对 LSTM 前向传输运算进行加速;Zheng 等[7]采用排列块对角掩码矩阵来生成稀疏模型,极大地降低了剪枝后索引功耗;Li 等[8]提出了一种按列修剪策略,删除了剩余权重的所有列索引和大部分行索引,降低了权重索引读取功耗,提高了加速器的整体性能。这种方法引入了剪枝等新的操作,势必导致电路的运行时间增加,尤其是在模型规模庞大时,增加的时间甚至可能超过计算本身所需的时间,因此,这种方法主要适用于小规模模型。另一研究方向是针对 LSTM 模型的计算特性,设计特殊计算架构以提高加速器的计算效率。其中,Que 等[9]使用列式矩阵向量乘法代替行式操作,消除了 RNN 推理时的数据依赖,从而提高系统的吞吐量,但是在将行式操作映射到列式矩阵操作的过程中高度依赖于详细的层形状,这使得当需要计算的数据量过大的时候将耗费过多的数据排列时间;T. Joseph 等[10]提出了一种利用收缩阵列结构和并行数据处理单元加速矩阵向量乘法的硬件架构,采用分裂矩阵方法,将较大的矩阵分解为较小的矩阵,从而降低了并行计算的硬件复杂度,但是在对数据进行并行处理时,需要考虑数据缓存占用,当模型数据过大时,这将对硬件资源有更大的要求。

为了避免在利用数据稀疏性减少数据量所带来的索引延迟问题,同时也为了减轻矩阵式计算架构中存在的数据缓存压力,本文根据 LSTM 算法的计算特点,设计并实现了一种基于脉动阵列的分布式计算 LSTM 加速器。首先,本文通过分析传统 LSTM 加速器的向量乘操作,提出一种权重参数切割方式,通过分布式存储输入数据并流动权重参数进行两者的耦合相乘,以减少数据流动性并降低读取功耗;然后,设计并实现了一种基于脉动阵列的分布式计算架构,通过脉动的方式传递数据,从而减少计算单元的空置率并提高计算效率。最后,在 VU13P 系列 FPGA 板上进行验证,实验结果显示,本设计相较于当前其他典型设计均有不同程度的提升。

1 LSTM 基本原理

LSTM 模型最早由 S. Hochreiter 和 J. Schmidhuber[11]于 1997 年提出,通过引入门控机制可以有效地解决 RNN 的梯度爆炸或消失问题。LSTM 基本单元结构如图 1 所示。基于传统 RNN 模型,LSTM 网络改进主要在以下 2 个方面。

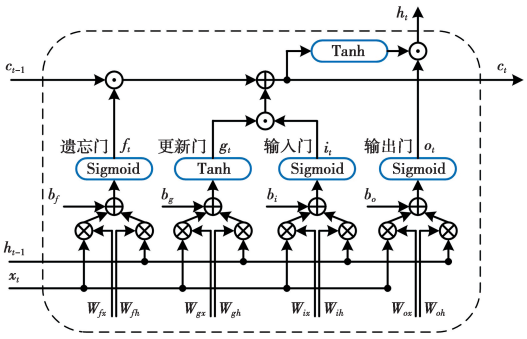

图 1 LSTM 基本单元结构

Fig.1 LSTM basic unit structure diagram

一个是引入 4 个门控机制来控制信息传递的路径,具体计算可用式(1)—(4)表示,分别为输入门 i_t、输出门 o_t、遗忘门 f_t 和更新门 g_t。输入门 i_t 控制当前时刻的更新状态有多少信息需要保存,输出门 o_t 控制当前时刻的更新状态 g_t 有多少信息需要输出给外部状态 h_t,遗忘门 f_t 控制上一个时刻的内部状态 c_{t-1}需要遗忘多少信息,更新门 g_t 使用非线性函数得到当前时刻的更新状态。

$$i_t = \sigma(\boldsymbol{W}_{xi} \otimes x_t + \boldsymbol{W}_{hi} \otimes h_{t-1} + b_i) \quad (1)$$

$$o_t = \sigma(\boldsymbol{W}_{xo} \otimes x_t + \boldsymbol{W}_{ho} \otimes h_{t-1} + b_o) \quad (2)$$

$$f_t = \sigma(\boldsymbol{W}_{xf} \otimes x_t + \boldsymbol{W}_{hf} \otimes h_{t-1} + b_f) \quad (3)$$

$$g_t = \tanh(\boldsymbol{W}_{xg} \otimes x_t + \boldsymbol{W}_{hg} \otimes h_{t-1} + b_g) \quad (4)$$

另一个是引入一个新的内部状态 c_t 专门进行线性的循环信息传递,由式(5)计算得到;同时非线

性的输出信息给隐藏层的外部状态 h_t，由式(6)表示。

$$c_t = f_t \times c_{t-1} + i_t \times g_t \tag{5}$$

$$h_t = o_t \times \tanh(c_t) \tag{6}$$

式(1)—(6)中：σ 表示 Sigmoid 函数；tanh 表示 Tanh 函数；⊗表示向量乘；×表示向量点乘；x_t 表示当前的输入数据；h_t 表示外部状态；$\boldsymbol{W}$ 表示权重参数矩阵；b 表示偏置参数。

传统 RNN 只具有短期记忆 h 这一个状态信息，而 LSTM 网络新引入状态信息 c 作为长期记忆，这使得在面对长序列任务时，LSTM 网络的预测精度会更高。

2　LSTM 加速器的设计

2.1　加速器整体系统架构

图 2 展示了本文所设计的 LSTM 加速器整体系统架构。由顶层控制模块(Top Control)、全局寄存器(Global REG)以及 4 个计算核心(PE Core)3 部分组成。本文设计的 LSTM 加速器的输入数据 x 和中间数据的位宽均为 16 bit，权重参数 W_in 的位宽为 8 bit。存储在 DRAM 中的权重参数 W_in 通过 64 bit位宽的数据线每次送入 8 个数据到计算核心中，然后，输入数据 x 和 LSTM 的外部状态 h_t 经全局寄存器切割重排后通过 128 bit 位宽的数据线以 x_t 和 h_{t-1}形式每次送入 8 个数据到计算核心。在计算中，顶层控制模块实时调控 W_in 的流动与 x_t 和 h_{t-1} 耦合相乘，并将计算结果 h_t 送回全局寄存器中，在计算结束后，将最终输出结果 h 送回到 DRAM 中。4 个计算核心分别对应 LSTM 模型中的 4 个门结构，每个计算核心由 2 个 8×7 的计算阵列(PE Array)、加法器(ADD)和激活函数(Sig/Tanh)组成，计算阵列实现的是式(1)—(4)中的向量乘⊗操作。

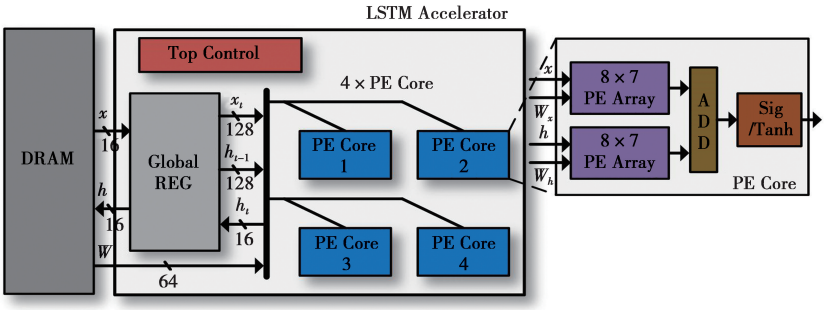

图 2　本文 LSTM 加速器整体系统架构

Fig.2　Overall system architecture of LSTM accelerator in this article

2.2　基于脉动阵列的分布式计算架构

计算阵列(PE Array)架构如图 3 所示，主要包括 2 个输入端口、7 个权重存储(WH0-6)模块、56 个 PE 组成的 PE 阵列和 1 个加法树(ADD Tree)模块，其中，2 个输入端口分别连接输入数据 x 或外部状态 h，及其对应的权重参数矩阵 $\boldsymbol{W}_x$ 或 $\boldsymbol{W}_h$。计算阵列运行过程为①每次输入 8 个权重参数，按顺序将所有参数存储到 WH0-6 中；②每次输入 8 个数据 x 或 h，按顺序存储到每一列的 8 个 PE 当中；③PE 阵列流动每一列的权重参数 W_in 与预先存储的输入数据 x 或外部状态 h 进行耦合相乘，并以脉动的方式将结果传递到下一列；④通过加法树模块将结果相加，以实现向量乘操作。

传统的 LSTM 模型中向量乘操作如图 4 所示。以输入数据维度和 LSTM 模型隐藏维度均是 56 为例，每个时间步输入一维的 1×56 的输入数据，与二维的 56×56 的权重参数矩阵进行向量乘，即输入数据遍历权重参数矩阵中的每一行进行乘累加操作，每一行输出 1 个数据，最后共输出一维的 56×1 的数据。具体计算流程如算法 1 所示。

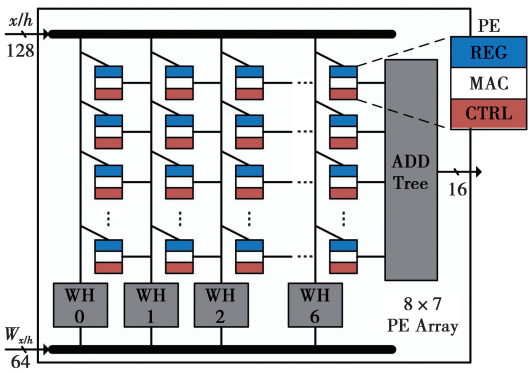

图 3　计算阵列(PE Array)架构

Fig.3　Computational array (PE Array) architecture

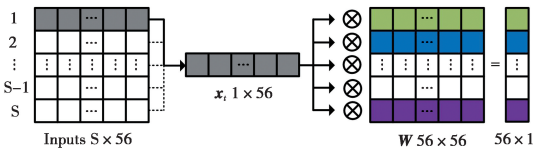

图 4 向量乘操作示意图

Fig.4 Schematic diagram of vector multiplication operation

算法 1 传统 LSTM 模型中的向量乘计算流程伪代码

输入:输入数据维度、LSTM 模型隐藏维度:$N=56$、时间步:S、权重参数矩阵:$\boldsymbol{W}$、输入数据向量:$\boldsymbol{x}_t$;

输出:y_t。

```
for i=1 to S loop1{
    for j=1 to N loop2{
        D=x_t×W[j];
        k=N;
        while (k>1) loop3{
            D[1:k/2]=D[1:k/2]+D[k/2+1:k];
            k=k/2;
        end loop3}
        y_t[j]=D[1];
    end loop2}
endloop1}
```

在计算每一个时间步的 loop1 中,需要中间缓存空间 56×16 bit,消耗 56 个乘法器和 55 个加法器。生成计算结果所消耗的循环周期数可以用式(7)计算,N 表示输入数据维度和 LSTM 模型隐藏维度 56,1 表示 56 个乘法器并行进行需要 1 个周期,lb(N)表示计算 N 个数据需要的加法树级数。由式(7)可得,传统 LSTM 模型中向量乘计算共需要消耗 392 个循环周期数。

$$C_{\text{Cycles-before}} = N \times (1 + \text{lb}(N)) \tag{7}$$

权重参数切割方式如图 5 所示。分析向量乘操作的计算过程可知,输入数据中每个数据对应相乘的总是权重参数矩阵中的对应位置的一列,由于输入数据远少于权重参数矩阵,则可以通过固定输入数据,将权重参数进行流动遍历输入数据的方式进行计算,以减少数据流动性,降低数据读取功耗;权重参数在模型进行推理过程中是固定不变的,则可以通过将其切割分布存储的方式,以减轻流动时对硬件的带宽压力。在本文设计中将 56×56 的权重参数矩阵 $\boldsymbol{W}$ 按照列维度,以 8 列为一组进行切割,分别存储在 WH0-6 中。

图 6 展示了 8×7 脉动计算阵列的详细架构,主体由 8 行 7 列共 56 个 PE 单元组成。计算阵列提前将输入数据按顺序依次存入 0-55 序号 PE 单元中,计算开始后,第 1 个周期从 WH0 中读出第 1 行,分割输入到 PE0-7 中,分别与输入数据 X0-7 进行乘操作;第 2 个周期,PE0-7 输出相乘结果并输送给 PE8-15,WH1 读出第 2 行,分割输入到 PE8-15,与输入数据 X8-15 和 PE0-7 输出的相乘结果进行乘加操作,同时 WH0 中读出第 2 行,分割输入到 PE0-7 中进行乘操作;以此类推;直到第 8 个周期开始,PE48-55 输出数据到加法数模块中。加法数模块有 3 级加法树,共 7 个加法器组成,对 PE 阵列中的 8 行输出的 8 个数据进行相加操作。具体计算流程如算法 2 所示。

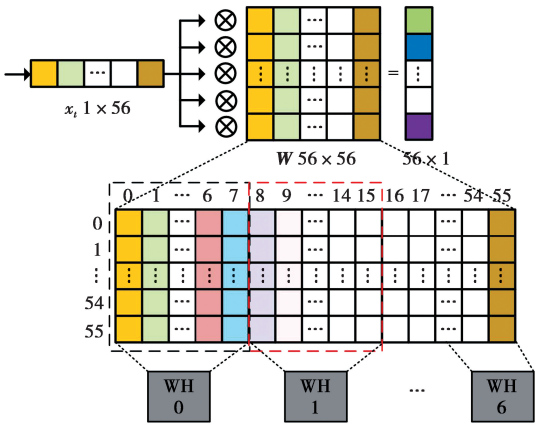

图 5 权重参数切割方式

Fig.5 Weight parameter cutting method

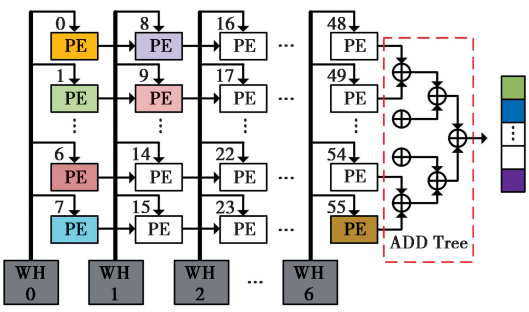

图 6 基于脉动阵列的分布式计算架构图

Fig.6 Distributed computing architecture diagram based on pulsating array

算法 2 基于脉动阵列架构的分布式计算流程伪代码

输入:输入数据维度和 LSTM 模型隐藏维度:$N=56$、PE 阵列维度:$H\times C=8\times 7$、时间步:S、权重参数存储模块:WH0-6、输入数据向量:$\boldsymbol{x}_t$;

输出:y_t。

```
for i=1 to S loop1{      //遍历时间步
    k=list[0:C];
    for j=1 to N loop2{
        D1<=x_t[H×0+1:H]×WH0[H×k[0]+1:8×k
```

[1]];

5:　　$D2<=D1+x_t[H\times1+1:H\times2]\times WH1[H\times k[1]+1:H\times k[2]]$;

6:　　$D3<=D2+x_t[H\times2+1:H\times3]\times WH2[H\times k[2]+1:H\times k[3]]$;

7:　　$D4<=D3+x_t[H\times3+1:H\times4]\times WH3[H\times k[3]+1:H\times k[4]]$;

8:　　$D5<=D4+x_t[H\times4+1:H\times5]\times WH4[H\times k[4]+1:H\times k[5]]$;

9:　　$D6<=D5+x_t[H\times5+1:H\times6]\times WH5[H\times k[5]+1:H\times k[6]]$;

10:　　$D7<=D6+x_t[H\times6+1:H\times7]\times WH6[H\times k[6]+1:H\times k[7]]$;

11:　　$D8<=D7[1:4]+D7[5:8]$;

12:　　$D9<=D8[1:2]+D8[3:4]$;

13:　　$y_t[j]<=D9[1]+D9[2]$;

14:　　$k=\{k[1:C],k[0]\}$;

15:　end loop2}

16: end loop1}

在计算每一个时间步的 loop1 中，由于是并行流水线计算，不需要中间缓存空间，共消耗 56 个乘加器和 7 个加法器。生成计算结果所消耗的循环周期数可以用式(8)计算，N 表示输入数据维度和 LSTM 模型隐藏维度 56，C 表示 PE 阵列的列维度，lb(H)表示计算 PE 阵列的行维度 H 个数据需要的加法树级数。由式(8)可得，基于脉动阵列架构的分布式计算共需要消耗 66 个循环周期数。

$$C_{\text{Cycles-later}} = N + C + \text{lb}(H) \tag{8}$$

本文设计的基于脉动阵列的分布式计算架构，66 个循环周期可以完成 1×56 的输入数据与 56×56 的权重参数矩阵进行的向量乘操作，对比传统 LSTM 模型中向量乘需要的 392 个周期，在计算速度上有 5.9 倍的提升。而且，由于中间数据是通过脉动方式传递到下一级，不需要中间缓存数据，减轻了部署模型的边缘计算端设备的内存压力。

2.3　PE 结构

图 7 展示了 PE 的结构。PE Control 模块调控 PE 有 3 个工作模式：待机模式、存储模式和计算模式。待机模式，PE 计算单元处于待机状态，无输出；存储模式，受 x_en 控制，接收输入数据 X 并存储在 Input_x/h REG 寄存器中；计算模式，受 MAC_en 控制，接收累加数据 D 和权重参数 W_in，一个周期进行乘累加操作 $X\times W$_in$+D$，并在下一周期输出乘累加结果。其中，X、D 均是 16 bit 数据，在计算过程是补码表示的定点数，1 bit 符号位，4 bit 整数位以及 11 bit 的小数位；而 W_in 是 8 bit 数据，1 bit 符号位和 7 bit 小数位，所以在送入 MAC 前需要对其末位补 0 至 12 bit 数；而接收乘累加结果需要 32 bit 位宽，为了控制运算过程的位宽统一，使用 Cropping 模块裁切至 16 bit 数。

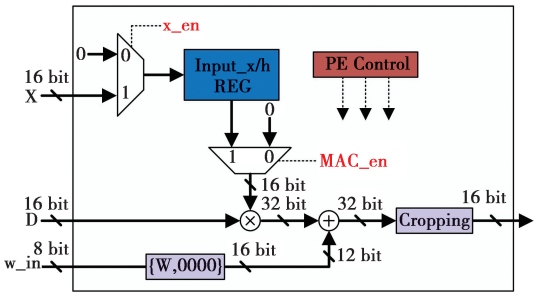

图 7　PE 结构

Fig.7　PE structure

3　设计实验与结果分析

3.1　软硬件实验环境

本文基于 Xilinx 公司的 Vivado2022.2 开发环境进行功能仿真和综合验证，选择 Xilinx 公司的 VirtexUltraScale+VU13P 系列 xcvu13p-fhga2104-2-e 芯片 FPGA 开发板作为测试载体，完全使用 Verilog HDL 语言对设计的加速器进行编码。硬件资源消耗如表 1 所示。其中，VU13P 系列 FPGA 板载 DSP 支持定点 16 bit 的乘加运算，BRAM 资源用于模拟外部存储 DRAM 存储输入数据以及权重参数。

表 1　硬件资源消耗

Tab.1　Hardware resource utilization

资源类型	消耗量
LUT	15 257
Register	16 836
DSP	507
BRAM	256

同时，本文在 PC 端基于 Pytorch 库搭建了标准 LSTM 模型作为本设计的 CPU 实验平台对比，PC 端使用 Intel 酷睿 i5-12400F 处理器，主频为 2.5 GHz，DDR3 内存大小为 48 GByte，操作系统为 Win11。

3.2　性能评估

本文使用 MNIST 数据集[13]对所设计的 LSTM 加速器进行验证，由于数据集中的图片维度是 28×28，而本文设计的加速器的输入数据维度是 56，故对 MNIST 数据集图片扩展至 56×56 后再放入标准

LSTM 模型中进行训练与推理验证。

衡量硬件加速器性能的一个重要指标是其峰值计算能力。在理想情况下,可以通过每个时钟周期可执行的计算次数(包括乘法和加法)来评估硬件加速器的峰值计算性能,即算力。具体的计算公式[12]为

$$P_{\text{pere-perk}} = 2 \times P \times f \tag{9}$$

式(9)中:P 表示加速器中 PE 的数量;系数 2 表示每个 PE 在一个周期内执行一次乘法和一次加法操作;f 为时钟频率。

本文设计的加速器,共有 4 个 PE 核心,每个 PE 核心有 2 个 MAC 阵列,每个 MAC 阵列有 56 个 PE,合计 4×2×56 = 448 个 PE。当时钟频率为 200 MHz 时,由式(7)可得,加速器算力为 179.2 GOPS,处理 1 张 56×56 图片需要 4 852 个周期,即 4 852/200 MHz = 24.26 μs。

在 PC 端使用 CPU 作为推理设备,基于 Pytorch 库调用 time.time( ) 函数监控运行时间,得到推理 10 000 张图片,用时 26.3 s,平均处理 1 张图片用时 2.63 ms。本文使用的标准 LSTM 模型运算量为 2 819 040,2 819 040/2.63 ms = 1.07 GOPS[14],CPU 有效算力为 1.07 GOPS。表 2 给出了基于 FPGA(VU13P)与 CPU 在图像计算任务中的性能与能效对比,该加速器的计算速度是 CPU 平台的 108 倍,功耗为 CPU 的 1/79,算力是 CPU 的 167 倍。

表 2 FPGA 与 CPU 数据对比

Tab.2 Performance comparison between FPGA and CPU

类别	VU13P	CPU
单张图片计算时间/μs	24.26	2 630
静态功耗/W	2.958	14.2
峰值功耗/W	3.301	41.5
实际功耗/W	0.343	27.3
有效算力/GOPS	179.2	1.07
能效比/(GOPS/W)	522.4	0.039

图 8 展示了加速器各模块的功耗以及计算模块中各资源的功耗占比。功耗报告是通过 Vivado 2022.2 软件对 LSTM 加速器源代码进行实现(IMPLEMENTATION)后生成的。在图 8 中,左侧柱状图显示了加速器 3 个模块的功耗分布,其中,由 4 个计算核心组成的计算模块占总功耗的 84.55%。右侧饼状图则展示了计算模块中各资源的功耗占比,Logic 的功耗远小于 DSP 的功耗,这是因为加速器设计中所有的乘加运算均通过 DSP 资源直接完成,而没有使用 LUT 查找表进行替代。此外,由于 BRAM 的功耗低于 0.001 W,因此未单独列出。表 3 展示了本文设计的 LSTM 加速器与其他 4 种文献中加速器的性能对比,本文设计的加速器能效比相较于文献[7],文献[9]、文献[15]、文献[16]分别提高了 34%、72%、193%、731%,但有效算力均低于其他参考文献。这一差距源自其他文献所采用的 LSTM 模型规模较大,因此耗费的资源也更为庞大,这不利于在边缘计算端设备部署,所以本文选取验证的 LSTM 模型相对较小。文献[7]消耗的 DSP 为 0,是因为它采用 LUT 查找表替代 DSP 计算乘加,因此,LUT 消耗的资源较多,而文献[15]中 LUT 消耗较少的原因是其主要利用 DSP 进行计算,所以 DSP 消耗的资源较多。

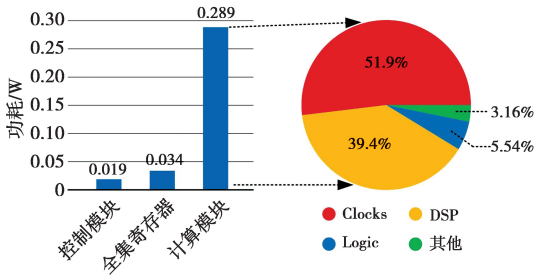

图 8 加速器功耗及计算模块中各个资源功耗占比图

Fig.8 Accelerator power consumption diagram and the proportion of power consumption of each resource in the calculation module

表 3 与其他文献 LSTM 加速器性能对比

Tab.3 Performance comparison with prior LSTM accelerators

类别	文献[7]	文献[9]	文献[15]	文献[16]	本文
FPGA 型号	Arria10	Stratix10 GX2800	XCKU9P	AlevoU50	VU13P
工作频率/MHz	150	250	200	280	200
DSP/个	0	5 245	1 600	4 224	507
LUT/个	257 871	487 232	5 600	122 935	15 257
有效算力/GOPS	2 220	29 574	200	2 036	179.2
实际功耗/W	5.7	98	9.0	32.3	0.343
能效比/GOPS/W	398.5	302	177.8	62.84	522.4

4　结束语

本文设计了一种基于脉动阵列架构的分布式计算 LSTM 加速器,加速器的分布式计算阵列通过脉动的方式进行数据传输与计算,在减少中间数据缓存占用的同时还提高了计算速度。在 VU13P 系列 FPGA 进行实验的结果表明,本文设计的加速器在 200 MHz 的频率下有效算力 179.2 GOPS,动态功耗 0.343 W,能效比 522.4 GOPS/W,计算速度和有效算力分别是 CPU 的 108 倍和 167 倍,功耗是其 1/79。相较于当前典型设计,能效比分别有 731%,193%,72%,34%的提升。实验表明,本文设计的 LSTM 加速器适合应用于资源受限和对功耗要求严格的边缘计算端设备,同时在后续研究工作中,可以基于本文设计的脉动阵列架构,加入可重构的思想,实时调控 PE 的计算组合,以适配不同规模及种类的神经网络并对其加速。

参考文献:

[1] GOLDBERG Y. A primer on neural network models for natural language processing[J]. Journal of Artificial Intelligence Research, 2016(57): 345-420.

[2] HAN S, KANG J, MAO H, et al. Ese: Efficient speech recognition engine with sparselstm on fpga[C]//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. New York, NY, USA: ACM, 2017: 75-84.

[3] DONAHUE J, ANNE H L, GUADARRAMA S, et al. Long-term recurrent convolutional networks for visual recognition and description[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ, USA: IEEE, 2015: 2625-2634.

[4] HOCHREITER S, BENGIO Y, FRASCONI P, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies[C]//A Field Guide to Dynamical Recurrent Neural Networks. Piscataway. NJ, USA: IEEE, 2001.

[5] WANG M, WANG Z, LU J, et al. E-LSTM: An efficient hardware architecture for long short-term memory[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems. USA: IEEE, 2019, 9(2): 280-291.

[6] 高琛,张帆,高彦钊.利用数据稀疏性的 LSTM 加速器设计[J].电子学报, 2021, 49(2): 209-215.
GAO C, ZHANG F, GAO Y Z. Design of LSTM accelerator using data sparsity[J]. Acta Electronica Sinica, 2021, 49(2): 209-215.

[7] ZHENG Y, YANG H, JIA Y, et al.PermLSTM: A high energy-efficiency LSTM accelerator architecture[J]. Electronics, 2021, 10(8): 882.

[8] LI S, ZHU S, LUO X, et al. An efficient sparselstm accelerator on embedded fpgas with bandwidth-oriented pruning[C]//2023 33rd International Conference on Field-Programmable Logic and Applications (FPL). USA: IEEE, 2023: 42-48.

[9] QUE Z, NAKAHARA H, NURVITADHI E, et al. Recurrent neural networks with column-wise matrix-vector multiplication on FPGAs[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 30(2): 227-237.

[10] JOSEPH T, BINDIYA T S. Performance-driven LSTM accelerator hardware using split-matrix-based MVM[J]. Circuits, Systems, and Signal Processing, 2023, 42(11): 6660-6683.

[11] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J].Neural Computation,1997,9(8):1735-1780.

[12] 田翔,周凡,陈耀武,等.基于 FPGA 的实时双精度浮点矩阵乘法器设计[J].浙江大学学报:工学版, 2008, 42(9): 1611-1615.
TIAN X, ZHOU F, CHEN Y W, et al. Design of real-time double-precision floating-point matrix multiplier based on FPGA[J]. Journal of Zhejiang University: Engineering Edition,2008, 42(9):1611-1615.

[13] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 2002, 86(11): 2278-2324.

[14] 刘杰,葛一凡,田明,等.基于 ZYnQ 的可重构卷积神经网络加速器[J].电子学报, 2021, 49(4): 729.
LIU J, GE Y F, TIAN M, et al. Reconfigurable convolutional neural network accelerator based on ZYNQ[J]. Acta Electronica Sinica, 2021, 49(4):729.

[15] GHASEMZADEH S A, TAVAKOLI E B, KAMAL M, et al. BRDS: An FPGA-based LSTM accelerator with row-balanced dual-ratiosparsification[EB/OL]. (2021-01-07)[2025-09-18]. https://arxiv.org/abs/2101.02667.

[16] MAO N, YANG H, HUANG Z. An Instruction-Driven Batch-Based High-Performance Resource-Efficient LSTM Accelerator on FPGA[J].Electronics,2023,12(7):1731.

作者简介:

张红升,教授,博士生导师,博士,主要研究方向为超大规模集成电路与 Soc 设计、神经网络加速器设计等。E-mail: zhanghs@cqupt.edu.cn。

成卓立,硕士研究生,主要研究方向为神经网络加速器设计,可重构设计。E-mail:1390764213@qq.com。

(**编辑**:刘　勇)