一种基于频域内推理计算的长短期记忆神经网络硬件加速器设计

靳 松[1),2)] 陈诗琪[1),2)]

[1)](华北电力大学电子与通信工程系 河北 保定 071003)

[2)](华北电力大学河北省电力物联网技术重点实验室 河北 保定 071003)

摘 要 长短期记忆神经网络(Long Short-Term Memory,LSTM)可以捕捉到序列数据间长距离的依赖关系,因此在时间序列预测、自然语言分析和语音识别等领域得到广泛应用。然而,LSTM网络独特的门控机制和状态更新过程导致其推理计算的复杂度较高,参数量较大,对其在资源受限的边缘设备上的部署形成挑战。本文提出一种基于频域内推理计算的长短期记忆神经网络硬件加速器设计。采用循环分块矩阵对网络的权重参数进行压缩存储,结合快速傅里叶变换(Fast Fourier Transform,FFT)和频域激活函数实现频域内网络推理计算,避免在处理不同时间样本时频繁的时域-频域切换开销。采用坐标旋转数字计算机算法(Coordinate Rotation Digital Computer,CORDIC)替换频域内的乘法运算和超函数计算,实现LSTM的低功耗硬件部署。提出的硬件加速器在PYNQ-Z2开发板上进行了原型实现。面向开源时间序列数据集的实验结果表明,加速器实现了63.6 μs的网络平均推理延迟,功耗1.743 W,相比时域LSTM推理计算延迟降低了44.2%,功耗降低6.4%。同时,BRAM和FIFO的资源占用率仅为5%和2%,相比时域LSTM推理计算分别降低了83%和91.2%。

关键词 长短期记忆神经网络;分块循环矩阵;坐标旋转数字计算机;频域推理计算;快速傅里叶变换

中图法分类号 TP391 **DOI号** 10.11897/SP.J.1016.2025.01781

A Hardware Accelerator Design for Long Short-Term Memory Neural Networks Based on Frequency-Domain Inference Computation

JIN Song[1),2)] CHEN Shi-Qi[1),2)]

[1)](Department of Electronic and Communication Engineering, North China Electric Power University, Baoding, Hebei 071003)

[1)](Hebei Key Laboratory of Power Internet of Things Technology, North China Electric Power University, Baoding, Hebei 071003)

Abstract Long Short-Term Memory neural networks, as a type of Recurrent Neural Network (RNN), can effectively handle long-term dependencies in sequential data, thereby avoiding the gradient vanishing or explosion problems that traditional RNNs encounter with long sequences. By introducing mechanisms such as input gates, forget gates, and output gates, LSTM networks can selectively retain and forget information, thereby capturing long-term variations in data, making them widely applicable in fields such as time series prediction, natural language processing, and speech recognition. However, the unique gating mechanism and state update process of LSTMs result in high computational complexity and a large number of parameters. This situationnot only requires substantial memory but also demands significant computational power to support both training and inference processes, creating challenges for deploying these networks on resource-constrained edge devices. Therefore, exploring methods to compress LSTM models to reduce storage and computational demands is crucial for enabling edge

收稿日期:2025-01-09;在线发布日期:2025-05-12。本课题得到河北省省级科技计划资助(平台编号:SZX2020034)、河北省自然科学基金项目(F2021502006)资助。**靳 松**,博士,副教授。中国计算机学会会员,主要研究领域为集成电路设计、智能信息处理。E-mail:jinsong@ncepu.edu.cn. **陈诗琪**,硕士研究生。主要研究领域为集成电路设计、智能信息和语音信号处理。

computing of LSTM networks. Based on this background, this paper proposes a solution that aims to compress network parameters and enhance inference speed while ensuring that accuracy loss remains within an acceptable range. This paper proposes a hardware accelerator design for long short-term memory neural networks based on inference computation in the frequency domain. The method utilizes block-circulant matrix compression to store the network's weight parameters, combined with Fast Fourier Transform (FFT) and frequency-domain activation functions to achieve frequency domain network inference, thereby avoiding the frequent time-domain to frequency-domain switching overhead when processing different time samples. The Coordinate Rotation Digital Computer (CORDIC) algorithm is employed to replace multiplication operations and hyperfunction calculations in the frequency domain, enabling low-power hardware deployment for LSTM. The paper first partitions the input data and performs FFT transformation, followed by element-wise multiplication and accumulation with the frequency-domain weight matrix to obtain the accumulated outputs of the four gates. These outputs are then processed in parallel through frequency-domain activation functions to update the cell state and hidden state. In this way, the forward computation process of the LSTM can be divided into five main modules: the FFT/IFFT module, the multiplication module, the accumulation module, and the activation function module. Among them, the FFT/IFFT module is based on the rotation mode of the CORDIC algorithm in the circular coordinate system, using fixed rotation angles and shift-add operations to replace traditional butterfly calculations. The multiplication module utilizes the rotation mode of the CORDIC algorithm in the linear coordinate system to achieve element-wise multiplication, combined with parallel prediction algorithms to accelerate computation. The accumulation module is responsible for summing the results of each row block. The activation function module adopts frequency-domain linear approximation instead of traditional activation functions, enabling inference computation to be entirely performed in the frequency domain. The proposed hardware accelerator is prototyped on a PYNQ-Z2 development board. Experimental results on an open-source time series dataset demonstrate that the accelerator achieves an average network inference latency of 63.6 μs with a power consumption of 1.743 W. Compared to the time-domain LSTM, the inference latency is reduced by 44.2%, and the power consumption is lowered by 6.4%. Additionally, the resource utilization of BRAM and FIFO is only 5% and 2%, respectively, representing reductions of 83% and 91.2% compared to the time-domain LSTM inference.

Keywords long short-term memory neural networks; block-circulant matrix; the coordinate rotation digital computer algorithm; frequency-domain inference computation; fast fourier transform

1 引 言

长短期记忆网络作为一种循环神经网络(Recurrent Neural Network,RNN),能够有效处理序列数据中的长期依赖关系,避免了传统RNN在长时间序列上的梯度消失或梯度爆炸问题[1]。LSTM通过引入输入门、遗忘门和输出门等机制,能够选择性地保留和遗忘信息,从而捕捉数据中的长期变化趋势。鉴于其强大的序列处理能力,LSTM在语音识别[2-3]、自然语言处理[4-5]、时间序列预测[6]等领域得到了广泛应用,展现了卓越的性能和泛化能力。不同行业对LSTM网络的需求不同,金融行业对LSTM网络的需求主要体现在时间序列的实时处理和预测能力,在股票市场、外汇交易中数据体现出强相关性。该场景下LSTM网络加速器通过并行计算可以显著提升模型的推理速度,减少预测时间,从而非常适用于需要大规模并行计算和高实时性的

金融预测系统。医疗影像分析（如CT、X光等）涉及从大量的图像数据中捕捉时序特征，帮助医生监测病变的发展。LSTM网络加速器的优势在于利用时序分析对影像序列进行精准处理，在几毫秒内处理大量图像数据，帮助医生更早地发现疾病从而降低误诊率。

当前，在手机、可穿戴式计算机和物联网节点等边缘设备上部署人工智能应用已成为一种重要的趋势。这不仅能够降低数据传输到远程服务器的开销，为用户提供更好的使用体验，还可以有效地保护用户隐私[7]。与云计算相比，边缘设备通常只有有限的存储空间和计算资源，难以支持大规模网络的推理计算[8]。虽然LSTM在处理序列数据时表现出色，但其结构复杂，包含大量的权重参数。这不仅需要大量的存储空间，还需要强大的计算能力来支持其训练和推理过程，对其部署在资源受限的边缘设备提出了挑战[9]。

为应对上述挑战，一些研究工作提出结合权重剪枝和参数量化的轻量级LSTM网络[10]，以减少网络在硬件上部署时的参数存储开销。然而，权重剪枝导致的稀疏权重矩阵具有不规则性，需要稀疏压缩存储以及对应的硬件计算模块，经常无法实现最优的推理性能、能效和硬件利用率[11-12]。还有一些工作面向RNN网络提出了专用硬件加速方案[13-14]。文献[13]将RNN的计算拆分为单个时间步的计算，使用乒乓缓冲的结构存取时间步向量，使数据流不间断地输入到处理单元中，是一种用空间换取时间的方法。文献[14]通过多线程管理单元（Thread Management Unit，TMU）控制隐藏层的CE-H和输出层的CE-O两种计算引擎交替使用实现高效的访存。上述研究工作均是在时域内优化网络的推理计算。

近些年来，一些研究工作提出采用分块循环矩阵对卷积神经网络（Convolutional Neural Network，CNN）和RNN的权重参数进行压缩存储，结合快速傅里叶变换及其逆变换（Inverse FFT，IFFT），将时域内的卷积计算转换为频域中的点乘运算，从而实现快速的网络推理计算[15]。但因为仍旧采用时域内激活函数，网络推理计算不得不在时域-频域间反复切换，引入较大的推理时间延迟和计算资源开销。面向CNN，文献[16]提出频域内激活函数，从而在频域完成网络的全部推理计算。上述在频域中神经网络的推理计算方法虽然取得了不错的效果，但只是应用于CNN。目前，尚无面向LSTM网络完全在频域中执行推理计算方法。其次，已有的时域-频域或频域内网络推理计算方法需进行蝶形运算，采用乘法器实现频域内的复数乘法操作，硬件复杂，功耗较高，不利于部署在资源受限的边缘设备上。

本文提出一种基于频域内推理计算的LSTM网络硬件加速器设计。采用循环分块矩阵对网络的权重参数进行压缩存储，结合FFT、IFFT和频域激活函数实现频域内LSTM网络推理计算，避免在处理不同时间样本时频繁的时域-频域切换开销。采用CORDIC算法替换蝶形运算中的复数乘法和超函数计算，实现LSTM的低功耗硬件部署。本文的主要贡献如下：

（1）提出了基于频域内推理计算的LSTM网络硬件加速器架构，使用循环分块矩阵将权重矩阵压缩的同时为LSTM层中的非线性操作引入了频域激活函数，这一设计使得整个网络能够在频域内完成训练，同时保持推理计算全程在频域进行，相比时域LSTM推理计算延迟降低了44.2%；

（2）针对分块循环矩阵的特点，采用CORDIC算法在圆周坐标系完成频域FFT计算，同时在线性坐标系下实现复数乘法。

（3）在PYNQ-Z2 FPGA开发板上进行了加速器的原型实现。面向开源时间序列数据集的实验结果表明，加速器实现了63.6 μs的网络平均推理延迟，功耗1.743 W，相比时域LSTM推理计算延迟降低了44.2%，功耗降低了6.4%。同时，块随机存储器（Block Random Access Memory，BRAM）和先进先出（First In First Out，FIFO）存储器的资源占用率仅为5%和2%，相比时域LSTM推理计算分别降低了83%和91.2%。

本文结构如下：第二部分介绍了相关研究工作；第三部分介绍了基于频域内推理计算的无乘法硬件加速器架构和实现方法；第四部分给出实验结果和分析；第五部分对全文做了总结。

2　相关研究现状

LSTM网络参数量一般较大[17]，导致在模型推理时耗时较多，且占用较多的存储空间。采用小规模网络可加快推理，通常对LSTM进行压缩以实现更快的推理速度。目前比较常见的模型压缩方法主要有模型量化、剪枝、低秩分解和知识蒸馏等。考虑到在资源受限的硬件上实现，浮点数并不是最佳的选择，文献[18]表明在LSTM的推理运算中低位宽乘法比浮点乘法更高效，并且选择适当的低位宽定

点数几乎不会对RNN性能造成损失，且低位宽定点数也更适合在硬件平台上实现。网络剪枝算法按照一定的准则裁剪掉冗余的参数，减少模型的参数量。文献[19]提出了负载平衡的剪枝算法对LSTM网络进行简化，通过设置一个阈值将权重矩阵中小于阈值的权重置0，对精度影响可以忽略不计的同时将参数压缩了90%以上。矩阵低秩分解的方法可避免剪枝引起的内存读取不规则问题。文献[20]将不同的压缩方法相结合，在一个由5层组成的LSTM网络中，底层(1-2层)的循环权重矩阵使用Toeplitz矩阵进行压缩保留更好的特征值，顶层(3-5层)的投影权重矩阵采用共享低秩因子减少全局参数量，但该分层压缩策略未能使用统一的硬件结构，导致计算速度存在层间不均衡的问题。

在对神经网络进行参数压缩后，硬件架构的设计也要针对模型的不同做出可配置的改变。合理的硬件架构对降低计算和内存访问开销以及提高系统速度十分关键。文献[21]以数字信号处理器(Digital Signal Processor，DSP)作为流水线的核心单元，将LSTM中每个神经元的计算映射到一个DSP块，包括乘法、累加和激活函数，留下查找表(Look-Up Table，LUT)用于权重存储。文献[22]针对大型LSTM网络将数据存储在双倍数据速率(Double Data Rate，DDR)内存中，将权重矩阵分为多个块，分批传入计算单元，使计算时间大于等于传输时间，隐藏传输时延，适合大型LSTM网络的推理计算，但因内存带宽不足会产生性能瓶颈。文献[23]允许各层选择不同精度的乘法单元，但是乘法计算需要更多的时钟周期来完成。文献[24]将LSTM矩阵运算中的加法器和乘法器集成在一个统一的计算内核中，计算结果通过直接存储器访问(Direct Memory Access，DMA)传输至主处理器，在主处理器中使用软件计算激活函数(如Sigmoid和tanh)，然而频繁的数据传输会导致额外的时间开销。文献[25]提出在时域内使用CORDIC算法计算LSTM中的乘法和激活函数，每个处理单元中包含一个基于CORDIC算法的乘加单元，执行矩阵-向量乘法，并利用CORDIC算法的线性模式和双曲模式灵活计算出Sigmoid和tanh激活函数。该方法通过将矩阵向量乘法运算转化为外积形式来实现流水线结构。然而，由于CORDIC算法的迭代特性，激活函数的计算需按顺序判断迭代方向，从而引入额外的等待时间。即使部分计算能够并行执行，该问题仍影响整体的推理速度。综上，不同的LSTM网络压缩策略会在一定程度上改变其计算流程。因此，在硬件上部署神经网络除了需要对权重矩阵进行压缩外，还要考虑压缩算法与硬件适配性的问题。

CNN也有很高的计算复杂度，这很大程度上限制了它们在资源受限的边缘设备上的应用。先前的文献[26-28]表明，使用傅里叶变换在频域实现CNN的计算比在时域计算有更快的速度。然而由于缺乏非线性激活函数的频域实现，计算速度受到时域-频域转换的影响。因此，激发了全频域CNN的研究[29-32]。文献[33]提出了一种频域自适应激活函数ReLU的设计方法，目的是在频域模型训练和推理过程中不需要进行时域-频域切换，速度提高了约10倍。目前大部分研究集中于在边缘设备上部署CNN，关于RNN的研究较少，且RNN的资源消耗较大，这使得在边缘设备上使用RNN进行计算更具挑战性。虽然已有关于频域内CNN的研究，但据作者调研，目前尚无基于频域内计算的LSTM网络。本文通过分块循环矩阵对LSTM进行压缩，并利用频域激活函数将计算过程保留在频域内，在频域中完成推理运算。同时，采用CORDIC算法替代LSTM推理过程中的大量乘法操作和蝶形运算，降低硬件实现复杂度。

2.1 LSTM网络

LSTM利用记忆单元和门控机制有效地控制信息存储和遗忘，确保捕捉数据序列间长时间的依赖关系。LSTM网络的核心组件是细胞状态$\boldsymbol{c}_t$和隐藏状态$\boldsymbol{h}_t$。$\boldsymbol{c}_t$负责在序列处理中传递和更新长期的状态，$\boldsymbol{h}_t$融合了当前输入$\boldsymbol{x}_t$和细胞状态$\boldsymbol{c}_t$的信息，用于传递到下一时间步长或作为最终输出。每个LSTM单元包含四个门：输入门$\boldsymbol{i}_t$、遗忘门$\boldsymbol{f}_t$、候选记忆门$\boldsymbol{u}_t$和输出门$\boldsymbol{o}_t$。输入门$\boldsymbol{i}_t$决定输入信息的哪些部分更新到记忆单元中；遗忘门$\boldsymbol{f}_t$决定记忆单元中哪些信息需要丢弃；输出门$\boldsymbol{o}_t$决定哪些信息从记忆单元输出。$\odot$表示逐元素乘法，$+$表示逐元素加法，σ表示使用Sigmoid激活函数。对于给定时间步长t的输入$\boldsymbol{x}_t$，前一时间步的隐藏状态$\boldsymbol{h}_t$和细胞状态$\boldsymbol{c}_t$，LSTM单元的计算过程为

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i)\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f)\\
\boldsymbol{u}_t &= \tanh(\boldsymbol{W}_{ux}\boldsymbol{x}_t + \boldsymbol{W}_{uh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_u)\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{u}_t \odot \boldsymbol{i}_t\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o)\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned} \tag{1}$$

$\boldsymbol{W}_{ix}$、$\boldsymbol{W}_{fx}$、$\boldsymbol{W}_{ux}$、$\boldsymbol{W}_{ox}$和$\boldsymbol{W}_{ih}$、$\boldsymbol{W}_{fh}$、$\boldsymbol{W}_{uh}$、$\boldsymbol{W}_{oh}$分别为输入和隐藏状态的权重矩阵，$\boldsymbol{b}_i$、$\boldsymbol{b}_f$、$\boldsymbol{b}_u$、$\boldsymbol{b}_u$为相应的偏置项。由于LSTM单元中不同门的计算都是将各自的权重矩阵与输入和隐藏状态相乘，因此在实际计算时，可将4个不同门的权重矩阵合并为4个大的矩阵。

2.2　分块循环压缩矩阵

可以采用分块循环矩阵对LSTM网络中的权重矩阵进行压缩处理，从而节省权重参数的存储空间。一个分块循环矩阵是由多个较小的子矩阵块组成的，每个子矩阵块$\boldsymbol{W}_{ij}$本身是一个循环矩阵[34]。假设权重矩阵$\boldsymbol{W}$是一个$a\times b$的矩阵，每个子矩阵块的块大小为$n\times n$，则$\boldsymbol{W}$可以分为$p\times q$个子矩阵块$p=a/n$，$q=b/n$，那么整个矩阵可以表示为

$$\boldsymbol{W}=\begin{bmatrix}\boldsymbol{W}_{11} & \boldsymbol{W}_{12} & \cdots & \cdots & \boldsymbol{W}_{1q}\\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} & \cdots & \cdots & \boldsymbol{W}_{2q}\\ \vdots & \vdots & \boldsymbol{W}_{ij} & \ddots & \vdots\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ \boldsymbol{W}_{p1} & \boldsymbol{W}_{p2} & \cdots & \cdots & \boldsymbol{W}_{pq}\end{bmatrix}$$

采用分块循环矩阵表示LSTM的权重矩阵，结合FFT可以实现快速的网络推理计算，具体原因如下：LSTM模型在时域推理阶段的前向传播过程为

$$\boldsymbol{a}=\boldsymbol{W}\boldsymbol{x}\Leftrightarrow\begin{bmatrix}\sum_{j=1}^{q}\boldsymbol{W}_{1j}\boldsymbol{x}_j\\ \sum_{j=1}^{q}\boldsymbol{W}_{2j}\boldsymbol{x}_j\\ \cdots\\ \sum_{j=1}^{q}\boldsymbol{W}_{pj}\boldsymbol{x}_j\end{bmatrix}=\begin{bmatrix}\boldsymbol{a}_1\\ \boldsymbol{a}_2\\ \cdots\\ \boldsymbol{a}_p\end{bmatrix}\tag{2}$$

其中$\boldsymbol{a}_i$为列向量。在时域内的计算本质是循环卷积计算。由于每个循环矩阵$\boldsymbol{W}_{ij}$都可以简化为一个向量$\boldsymbol{w}_{ij}$，即$\boldsymbol{w}_{ij}$是$\boldsymbol{W}_{ij}$的第一行向量，分块循环矩阵的结构使得可以采用FFT将时域内的循环卷积变换为频域内的点乘运算。因此，式(2)在频域内的计算可表示为

$$\boldsymbol{a}_i=\sum_{j=1}^{q}F^{-1}\left[F\left(\boldsymbol{w}_{ij}\right)\odot F\left(\boldsymbol{x}_j\right)\right]\tag{3}$$

其中$F(\cdot)$是DFT算子，$F^{-1}(\cdot)$是IDFT算子，$\odot$表示权重与输入在频域上的点乘。

3　LSTM网络硬件加速器设计

3.1　算法级LSTM网络实现

本文模型的训练方法基于文献[34]，采用2层LSTM，每层有512个隐藏节点，并针对62个音素标签进行分类。采用分块循环矩阵和FFT对LSTM层进行压缩和计算，网络层的计算流程变为：FFT→元素乘法→IFFT。该过程涉及较为频繁的时域/频域切换，由于激活函数是非线性的，需要在时域中进行计算，每次在频域中计算后需转换到时域中进行激活，如果将推理过程保留在频域，则需在分块循环矩阵的基础上引入频域激活函数。根据傅里叶变换的线性性质，对于输入信号$x(t)$，其频域表示为

$$X(f)=F\{x(t)\}=\int_{-\infty}^{\infty}x(t)e^{-j2\pi ft}dt\tag{4}$$

传统LSTM网络采用公式(1)的激活函数Sigmoid和tanh，激活函数的时域表达式为

$$\mathrm{Sigmoid}(x)=\frac{1}{1+e^{x}}\tag{5}$$

$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\tag{6}$$

显然Sigmoid和Tanh是非线性运算，傅里叶变换不能直接用于计算其频域表示，为解决这一问题本文采用线性近似方法。经典LSTM网络使用Sigmoid和tanh作为激活函数，ReLU激活函数在LSTM网络中应用较少。这是因为ReLU在输入负值时输出为0，导致在训练过程中权重无法通过梯度下降更新[35]，其次ReLU在正值区域内的无界性可能引起梯度爆炸，与LSTM网络通过门控机制捕捉序列中长期依赖关系的设计目标存在冲突。鉴于以上原因，本文选择线性函数作为频域激活函数近似Sigmoid和tanh。在送入激活函数之前，先通过批归一化层对数据进行处理，将激活函数的输入特征归一化为零均值和单位方差，确保输入频域激活函数的值保持在-1到1的范围内，这样可以最大程度地减小近似引入的误差。训练期间为了保持反向传播算法不变，从LSTM层输出前需使用IFFT转换回时域，用于最后分类的全连接层的计算在时域中执行。

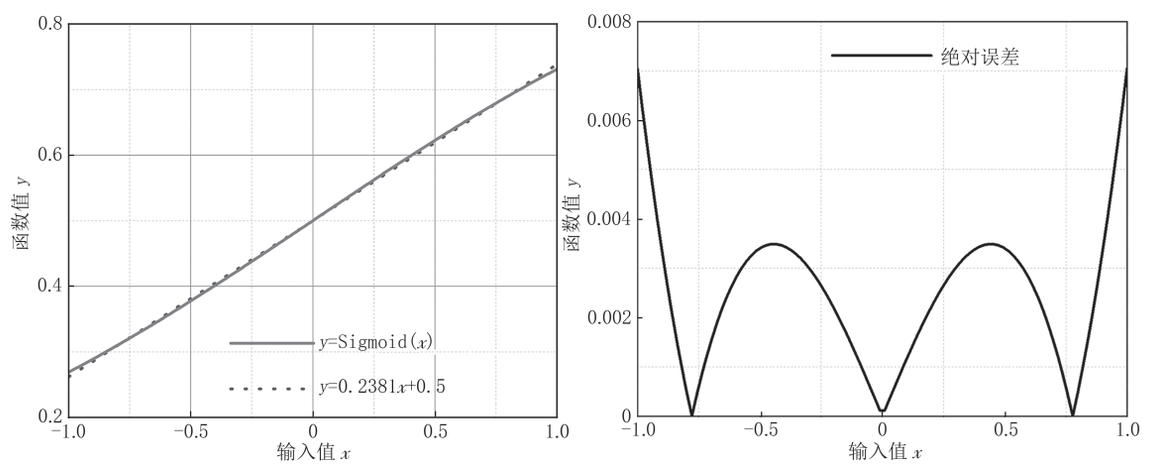
LSTM模型的前向传播过程为

$$\boldsymbol{a}_i=\sum_{j=1}^{q}F\left(\boldsymbol{w}_{ij}\right)\odot F\left(\boldsymbol{x}_j\right)\tag{7}$$

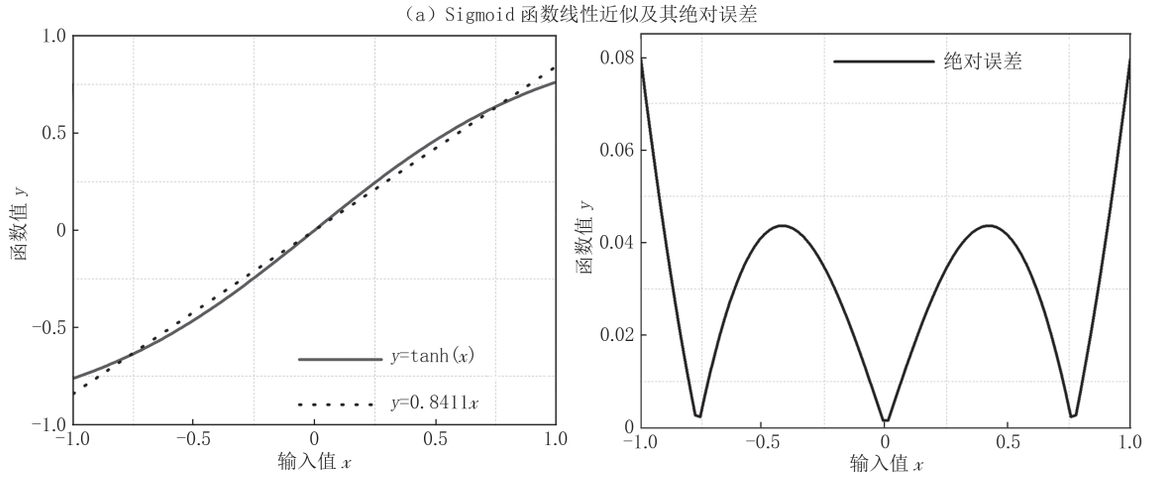
反向传播过程也可以使用分块循环矩阵来实现，用a_{il}表示$\boldsymbol{a}_i$中的第l个输出元素，用L表示损失函数。使用链式法则可以推导出反向传播过程如下：

$$F\left(\frac{\partial L}{\partial \boldsymbol{w}_{ij}}\right)=F\left(\frac{\partial L}{\partial \boldsymbol{a}_{il}}\right)F\left(\frac{\partial \boldsymbol{a}_{il}}{\partial \boldsymbol{w}_{ij}}\right)$$
$$F\left(\frac{\partial L}{\partial \boldsymbol{x}_{j}}\right)=\sum_{i=1}^{p}F\left(\frac{\partial L}{\partial \boldsymbol{a}_{i}}\right)F\left(\frac{\partial \boldsymbol{a}_{il}}{\partial \boldsymbol{x}_{j}}\right) \tag{8}$$

如3.1.2小节所示，LSTM网络推理计算时的矩阵向量乘法可以在频域内通过点乘实现。本文采用频域激活函数来计算神经元的值，从而保证网络的全部推理计算除全连接层外均在频域内完成。对于LSTM层，只需进行输入$\boldsymbol{x}_t$和输出$\boldsymbol{y}_t$时进行两次傅里叶变换，分别位于LSTM层的边界。对于频域激活函数，我们利用FFT的线性性质引入近似的Sigmoid和Tanh函数。激活函数的输入被归一化为具有零均值和单位方差，函数的中心区域可以用线性函数近似表示。如图1所示，可以看出激活函数在[−1，1]范围内表现出较好的线性近似特性。Sigmoid函数的线性拟合公式为Sigmoid($\boldsymbol{x}$)=0.2181$\boldsymbol{x}$+0.5，Tanh函数的线性拟合公式为Tanh($\boldsymbol{x}$)=0.8411$\boldsymbol{x}$。为了证明该方法的有效性，将本文中使用的线性拟合函数与Sigmoid和Tanh函数进行比较。我们计算了激活函数与其近似值之间的平均绝对误差(Mean Absolute Error，MAE)，在[−1，1]之间生成100个等距点。Sigmoid拟合的平均绝对误差为0.025，Tanh拟合的平均绝对误差为0.0308，所产生的误差处于可接受的范围内。为了更直观地说明频域激活函数对Sigmoid和tanh的拟合程度，引入决定系数(R^2)[36]进行评估，R^2的取值范围为0到1之间，越接近1表示拟合效果越好，其计算公式如下：



(a) Sigmoid函数线性近似及其绝对误差



(b) Tanh函数线性近似及其绝对误差

图1 频域激活函数

$$R^2=1-\frac{\sum\left(y_{true}-y_{fit}\right)^2}{\sum\left(y_{true}-\bar{y}_{true}\right)^2} \tag{9}$$

式中，y_{true}表示真实值，y_{fit}表示预测值，$\bar{y}_{true}$表示所有真实值的平均值，计算公式为$\bar{y}_{true}=\frac{1}{n}\sum_{i=1}^{n}y_{true,i}$。经计算，在区间[−1，1]内线性函数对Sigmoid和tanh函数拟合的R^2分别为0.9996和0.995，均超过

了0.99，表明本文提出的线性近似方法能很好地拟合目标激活函数。使用频域激活函数的频域内推理过程仅比在分块循环矩阵压缩后采用时域表达式[34]的推理过程准确率低了0.73%。在这项工作中，训练过程在离线完成。训练后的权重矩阵经FFT后用来配置LSTM的推理模型。在推理过程中，循环层的计算都保持在频域内。我们提出的频域内LSTM网络设计，只需一次从时域到频域的转换（即输入序列的FFT）和一次从频域到时域的转换（即第一个全连接层前的IFFT）。

3.2　LSTM网络硬件加速器设计

LSTM网络的计算过程中主要涉及两个关键部分：矩阵乘法和激活函数。本文为了优化硬件资源并加速计算过程，采用了分块循环矩阵对权重矩阵进行压缩，矩阵乘法被转移到频域中进行处理。时域内矩阵与向量的乘法转化为了两个复数序列的乘法运算，于是矩阵乘法被拆分为三个模块：FFT模块、复数乘法模块和累加模块。FFT模块对输入序列做快速傅里叶变换，将时域数据转换为频域数据。复数乘法模块将压缩后的权重矩阵与经过FFT处理后的输入序列进行复数乘法计算，这两个模块均采用了CORDIC算法进行优化。累加模块将每一行中的q个块对应值相加。激活函数采用了频域内线性近似方法，将传统的Sigmoid和tanh激活函数转化为适合频域计算的线性形式，从而在频域内完成LSTM网络的推理，避免了频繁的时域-频域切换，只需在最终的输出时通过IFFT模块将结果转换回时域。图2展示了这一计算过程的系统架构，其中包括FFT/IFFT模块、复数乘法模块、累加模块和激活函数模块共5个模块。

利用该系统架构实现LSTM网络，首先定义LSTM各门控单元的频域权重表示分别为：输入门的频域权重$\boldsymbol{FW}_{\text{inp}}$，遗忘门的频域权重$\boldsymbol{FW}_{\text{fg}}$，输出门的频域权重$\boldsymbol{FW}_{\text{out}}$，候选值门的频域权重$\boldsymbol{FW}_{\text{u}}$。在频域内输入序列的频域向量序列$\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_q$与频域权重矩阵$\boldsymbol{FW}_{\text{inp}}$、$\boldsymbol{FW}_{\text{fg}}$、$\boldsymbol{FW}_{\text{out}}$、$\boldsymbol{FW}_{\text{u}}$进行逐元素点乘运算，并按顺序累加得到四个门的累加结果$\boldsymbol{I}$、$\boldsymbol{F}$、$\boldsymbol{O}$、$\boldsymbol{U}$。累加后将频域数据保存，送到频域激活函数中进行运算。对不同门控单元的输出应用不同的激活函数。输入门、遗忘门和输出门的激活函数采用频域近似的Sigmoid函数，候选值门的激活函数采用频域近似的Tanh函数。输入为$\boldsymbol{I}$、$\boldsymbol{F}$、$\boldsymbol{O}$和$\boldsymbol{U}$，输出为输入门激活结果$\boldsymbol{SIG_I}=\sigma(\boldsymbol{I})$，遗忘门激活结果$\boldsymbol{SIG_F}=\sigma(\boldsymbol{F})$，输出门激活结果$\boldsymbol{SIG_O}=\sigma(\boldsymbol{O})$和候选值门激活结果$\boldsymbol{TANH_U}=\tanh(\boldsymbol{U})$。其中，$\sigma$表示Sigmoid函数的频域近似$\sigma(x)=0.2381x+0.5$，Tanh函数的频域近似$\text{Tanh}(x)=0.8411x$。结合当前时刻激活后的频域输入门$\boldsymbol{SIG_I}_t$、遗忘门$\boldsymbol{SIG_F}_t$和候选值门$\boldsymbol{TANH_U}_t$以及前一时刻的细胞状态$\boldsymbol{C}_{t-1}$，更新当前细胞状态$\boldsymbol{C}_t=\boldsymbol{SIG_F}_t\odot\boldsymbol{C}_{t-1}+\boldsymbol{SIG_I}_t\odot\boldsymbol{TANH_U}_t$。根据当前时刻输出门的激活输出$\boldsymbol{SIG_O}_t$以及当前时刻的细胞状态$\boldsymbol{C}_t$，计算当前时刻的隐藏状态$\boldsymbol{H}_t=\boldsymbol{SIG_O}_t\odot\tanh(\boldsymbol{C}_t)$，完成LSTM网络单个时间步上的推理过程。

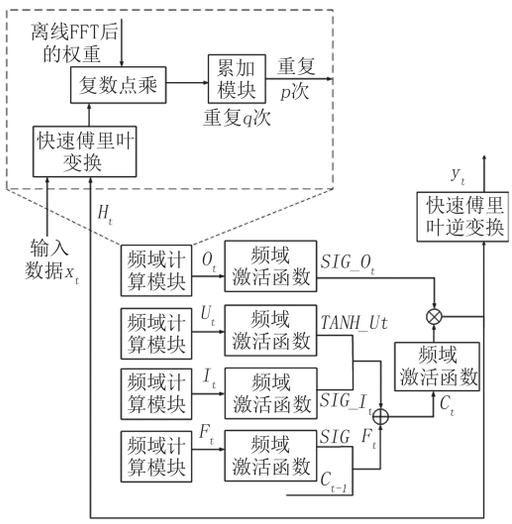


图2　系统架构图

3.2.1　硬件加速器架构

提出的硬件加速器架构如图3所示，输入向量$\boldsymbol{x}_t$按每n个值为一组送入基于CORDIC的FFT模块，输出复序列$\boldsymbol{X}_q$并行的乘以每个处理单元（Processing Element，PE）中相应的权值，复数乘法的结果传送到下一个PE的累加器寄存器中，每个门函数的累加结果按顺序送入激活函数模块，最后将激活函数$\boldsymbol{C}_{t-1}$，$\boldsymbol{H}_{t-1}$的输出存储在内部存储器中，以便在下一个时间步长计算中使用或者经IFFT后输出。硬件加速器的主要模块由一个快速傅里叶变换（FFT）模块，一个快速傅里叶逆变换（IFFT）模块，一组独立计算的处理单元PE和一个频域激活函数模块组成。每个处理单元主要由三个部分组成，分别是BRAM，基于CORDIC的复数乘法单元和累加器寄存器，BRAM保存离线计算好的频域权重，基于CORDIC的复数乘法单元执行对应的元素相乘，累加器寄存器则用来累加这些复数乘法的结果。与文献[24]和[25]的时域LSTM网络相比，本文首先对输入序列进行快速傅里叶变换，利用输入

序列的频域值与分块循环矩阵的频域值做复数乘法运算，在最终输出时通过快速傅里叶逆变换模块将频域值转换回时域。此外，与文献[24]中通过软件计算激活函数值和文献[25]中使用CORDIC-激活函数模块不同，本文通过用线性函数近似频域激活函数，从而显著减少了计算时间。

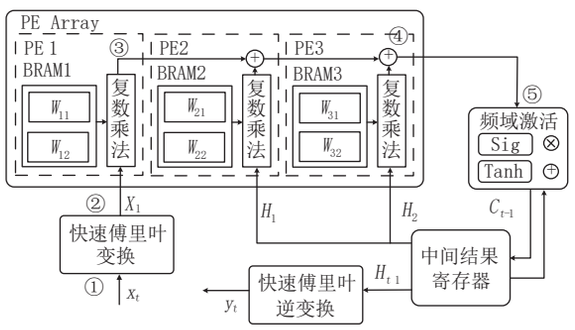


图3 基于频域的LSTM网络无乘法器硬件加速器架构

3.2.2 数据流

图4展示了当子矩阵块大小为$n=8$，每列包含$p=2$个块，每行包含$q=3$个块时，频域内LSTM数据流图，具体步骤如下：

(1)输入向量分块：将输入序列$\boldsymbol{x}_t$和隐藏状态$\boldsymbol{h}_t$每8个一组划分为若干输入向量块(如$x_1,x_2,\cdots,x_8$，记作第1个输入向量块)。

(2)快速傅里叶变换：对输入向量块进行快速傅里叶变换，得到频域表示$F(x_1),F(x_2),\cdots,F(x_8)$。

(3)频域矩阵乘法：输入向量块的频域值$\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_q$与预先计算好的权重矩阵的频域值$\boldsymbol{W}_{11},\boldsymbol{W}_{21},\cdots,\boldsymbol{W}_{q1}$进行点乘，得到点乘结果$\boldsymbol{X}_1\odot\boldsymbol{W}_{11},\boldsymbol{X}_2\odot\boldsymbol{W}_{21},\cdots,\boldsymbol{X}_q\odot\boldsymbol{W}_{q1}$。

(4)累加运算：将这q个点乘结果进行累加，得到累加后的频域结果$\boldsymbol{F}_1$。不同于传统时域方法需要将频域结果转换回时域再进行累加，本文直接在频域中完成累加。

(5)激活函数计算：累加后将频域数据保存，送到频域激活函数中进行运算。将步骤1-4重复p次，即可获得输入向量与权重矩阵乘积的全部结果。

文献[25]中提出的在时域中采用CORDIC算法实现LSTM网络推理，仅存在一个激活函数模块，4个门函数分为6个流水线阶段按顺序计算，且需耗费13个周期才能获取一个激活函数值。与之相比，本文提出的数据流架构一方面避免了激活函数按序计算的局限性，另一方面具备在单个周期内完成激活函数计算的能力。即便四个门函数以流水线形式展开计算也仅仅需4个周期。为了进一步减

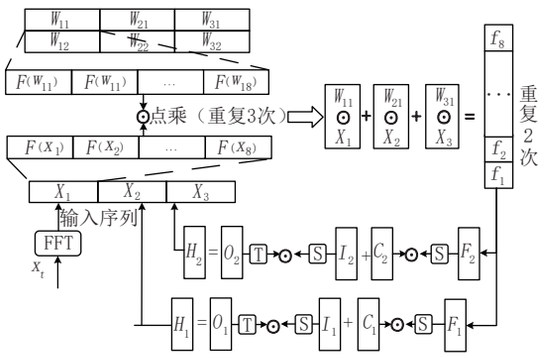


图4 基于频域的LSTM网络计算数据流

少数据等待时间，本文在基于频域的LSTM网络中采用了数据流水线策略。具体而言，每当计算出一行权重矩阵与输入向量的累加和时(如图4中的f_1)，便立即将该结果送入频域激活函数模块计算，四个权重矩阵$\boldsymbol{i}_t$、$\boldsymbol{c}_t$、$\boldsymbol{u}_t$、$\boldsymbol{f}_t$与输入向量的乘积同时进行，实现全并行运算。八个元素$f_1,f_2,\cdots,f_8$按顺序通过激活函数模块并与相应的门函数相乘，一旦完成H_t的第一个元素的计算，即可送到下一个时间步长，参与新一轮的迭代。

3.3 基于CORDIC算法的无乘法运算

CORDIC算法只需要使用移位、加法和减法操作即能完成三角函数、乘法和除法等数学运算，非常适合低功耗应用场景。CORDIC算法的广义表达式为

$$\begin{aligned} x_{i+1}&=x_i+md_i2^{-i}y_i\\ y_{i+1}&=y_i-d_i2^{-i}x_i\\ z_{i+1}&=z_i+d_i\theta_i \end{aligned} \tag{10}$$

在式(10)中，i表示迭代次数，d_i为旋转方向，由不同模式下y或z的正负决定。m决定了旋转所处的坐标系，$m=-1,0,+1$分别代表了双曲坐标系，线性坐标系和圆周坐标系，θ_i分别代表了三种不同坐标系下的增量，对于双曲坐标系$\theta_i=\mathrm{atanh}(2^{-i})$，线性坐标系$\theta_i=2^{-i}$，圆周坐标系$\theta_i=\mathrm{atan}(2^{-i})$。在CORDIC算法中，每次迭代所涉及的旋转操作实际上是围绕坐标系的原点展开的。这些旋转操作可被视为矢量的伸缩与旋转这两种基本变化的组合，不仅适用于计算FFT/IFFT中向量与旋转因子的乘积，还可以完成元素乘法。

3.3.1 CORDIC实现FFT

快速傅里叶变换是一种将时域信号转换为频域信号的高效算法，其中涉及大量复数乘法和加法运算。在传统硬件实现中，通常使用乘法器直接相乘的方式，以点数N为8的FFT为例，其计算过程需

要三个层级的迭代，共需要完成12次复数乘法和24次复数加法，并且还需配置一块只读存储器(Read-Only Memory，ROM)存放旋转因子，从而消耗大量的硬件资源，其对应的蝶形图如图5所示。

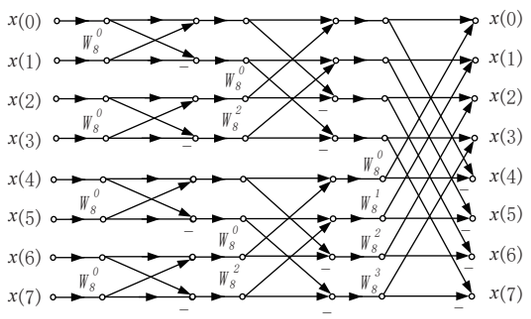


图5　蝶形运算

设有向量$\boldsymbol{A}=x_a+jy_a$，旋转因子$W_N^k=e^{-j2\pi k/N}$，二者的复数乘法表达式为

$$\boldsymbol{A}\cdot W_N^k=(x_a+jy_a)\cdot\left(\cos\left(-\frac{2\pi k}{N}\right)+j\sin\left(-\frac{2\pi k}{N}\right)\right) \tag{11}$$

假设相乘后的结果为$x_a'+jy_a'$，则有式(12)：

$$\begin{bmatrix}x_a'\\y_a'\end{bmatrix}=\cos\left(-\frac{2\pi k}{N}\right)\cdot\begin{bmatrix}1 & -\tan\left(-\frac{2\pi k}{N}\right)\\ \tan\left(-\frac{2\pi k}{N}\right) & 1\end{bmatrix}\begin{bmatrix}x_a\\y_a\end{bmatrix} \tag{12}$$

在CORDIC算法中，假设有一向量$\boldsymbol{A}(x_a,y_a)$处于直角坐标系内，逆时针旋转θ角度后得到一个新向量$B(x_b,y_b)$，这个过程可用式(13)表示：

$$\begin{bmatrix}x_b\\y_b\end{bmatrix}=\cos\theta\begin{bmatrix}1 & -\tan\theta\\ \tan\theta & 1\end{bmatrix}\begin{bmatrix}x_a\\y_a\end{bmatrix} \tag{13}$$

将式(12)与式(13)相比较可以看出，向量与旋转因子的复数乘法可以看作将向量旋转了$\theta=-2\pi k/N$度。所以对于FFT每一级的蝶形运算，其结果与旋转因子相乘的这一步可以通过CORDIC算法来实现，同理，IFFT也遵循相同的计算逻辑。

通过CORDIC算法实现FFT的结构如图6所示，在该实现过程中，旋转因子的不同取值会引发不同的处理策略。当旋转因子$W_8^0=1$时，根据式(11)输入值不进行旋转操作。当旋转因子$W_8^2=e^{-j\pi/2}$时代入到公式(13)，可得到表达式(14)如下：

$$\begin{aligned}x_b&=x_a\cos(-90^\circ)-y_a\sin(-90^\circ)\\ y_b&=x_a\sin(-90^\circ)+y_a\cos(-90^\circ)\end{aligned} \tag{14}$$

其中$\cos(-90^\circ)=0$，$\sin(-90^\circ)=-1$，此时只需将向量顺时针旋转90°，通过对原向量的虚实部交换与取反操作可得到结果$\begin{cases}x_b=y_a\\ y_b=-x_a\end{cases}$。当旋转因子$W_8^1=e^{-j\pi/4}$时，在CORDIC算法中需顺时针旋转45°。为了简化和加速计算，仅执行首次迭代(顺时针旋转45°)，同时附加模校正环节，将X_n，Y_n乘以伸缩因子K。对于迭代次数为1的情况，其伸缩因子$K=0.707\,106\,78$。采用与CORDIC迭代类似的方法，通过有限次迭代逼近伸缩因子。以二进制小数表示$K\approx0.101\,101\,01_2$，将x乘以伸缩因子的运算$x\cdot0.707\,106\,78$近似为$x\gg1+x\gg3+x\gg4+x\gg6+x\gg8$。这样分解后使$X_n$，$Y_n$与$K$的乘法转化为简单的移位加减运算且能在一个周期内完成，降低了硬件复杂度，而且与流水线CORDIC结构相似，使整体结构更加规则，彻底地消除了FFT中乘法器的使用。当旋转因子$W_8^3=e^{-j3\pi/8}$时，表示顺时针旋转135°，这种情况下只需要先旋转90°再旋转45°即可。

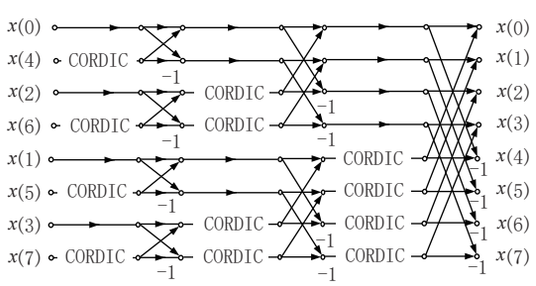


图6　CORDIC实现FFT

3.3.2　CORDIC实现元素乘法

本文采用CORDIC算法线性坐标系下的旋转模式实现元素乘法，其迭代公式(15)如下：

$$\begin{aligned}x_N&=x_{in}\\ y_N&\approx y_{in}+x_{in}z_{in}\\ |z_{in}|&\leqslant1\end{aligned} \tag{15}$$

初始化向量$(x_{in},y_{in},z_{in})=(x_{in},0,z_{in})$，旋转$N$次后得到$y_N=x_{in}\times z_{in}$。引入全并行的预测算法，借助二进制补码预测所有迭代的旋转方向，摆脱了传统算法需依据前一次迭代结果判断当前旋转方向的限制。z_{in}可以展开为

$$z_{in}=-b_0+\sum_{j=1}^{N-1}b_j2^{-j} \tag{16}$$

旋转方向d_i由b_j确定，取值为1和−1，其表达式为

$$\begin{cases}d_1=2b_0-1\\ d_i=1-2b_{i-1},\ i=2,3,\cdots,N\end{cases} \tag{17}$$

上述方法可通过z_{in}确定所有的旋转方向。本文采用16位宽的小数，y_{i+1}可以展开为

$$y_{16}=x_0\cdot d_0\cdot 2^{-0}+x_1\cdot d_1\cdot 2^{-1}+x_2\cdot d_2\cdot 2^{-2}+\cdots+x_{15}\cdot d_{15}\cdot 2^{-15} \tag{18}$$

根据式(18)中的2^{-i}对每个x_id_i进行相应的移位操作，再将移位结果通过加法树逐步累加，得到最终的y_{16}。

在复数乘法模块中，标准的计算公式如式(19)所示：

$$(a+bi)(c+di)=(ac-bd)+(ad+bc)i \tag{19}$$

式(19)可重新表示为式(20)：

$$(a+bi)(c+di)=R+Ii \tag{20}$$

其中实部$R=ac-bd$，虚部$I=(a+b)(c+d)-ac-bd$，经过优化后，原需进行4次实数乘法和2次实数加法的复数乘法运算被简化为3次实数乘法和5次实数加法。由于加法的计算复杂度远低于乘法，同时复数乘法模块利用CORDIC算法计算出ac，bd和$(a+b)(c+d)$，进而减少硬件资源的消耗。

在快速傅里叶变换中，可利用共轭对称性减少计算量和参数量。对于长为N的实数序列x经过FFT变换后，得到的复数序列X满足共轭对称性。因此在对频域的权重进行存储时，可以只存储其一部分值。若N为偶数只需存储$X_0,X_1,\cdots,X_{N/2}$共$N/2+1$个复数，考虑到X_0和$X_{N/2}$均为实数，因此可以将X_0和$X_{N/2}$看作一个复数，从而将存储空间减半。此外，在频域进行复数向量的元素乘法时，只需计算前$N/2+1$个频域点，其余点可通过共轭对称性推导得出，从而减少约一半的计算量。

4 实验结果与分析

4.1 实验设置

本文使用与文献[25]同类型的PYNQ-Z2 FPGA开发板，采用Xilinx Vivado HLS 2018.3开发工具以C++编写加速器模块，将加速器模块封装成IP核后送入Vivado 2018.3生成完整的overlay文件，最终把overlay文件部署到PYNQ-Z2开发板上运行。

为了验证本文提出的LSTM硬件架构的优势，将本文与文献[25]进行对比实验。实验中都采用文献[25]的网络参数，LSTM层的隐藏单元数为40，输入维度为30，由于权重矩阵在推理时是固定的，我们可以预先计算$F(\boldsymbol{w}_{ij})$的值并将其存储在BRAM中。在文献[25]的时域实现方法中，乘法操作和激活函数基于CORDIC算法实现，避免了对DSP单元的依赖。相比之下，我们的方法在此基础上不仅使用分块循环矩阵对LSTM网络层进行压缩，还引入了频域激活函数将计算保留在频域。

4.2 资源利用率与速度

与文献[24-25]相比，虽然本文方法经FFT后的频域值是复数，其实部和虚部均需存储，但由于FFT后复序列的共轭对称性，可消除近一半的共轭复数，进而减轻了BRAM的开销。同时，权重矩阵的频域值$F(\boldsymbol{w}_{ij})$和输入向量的频域值$F(\boldsymbol{x}_j)$相乘的结果也满足共轭对称性，大约可以消除一半的乘法和加法运算。参考文献[24]在PL端通过DSP以全并行的方式完成矩阵乘法计算，而激活函数的处理交由主处理器完成。文献[25]中的激活函数模块采用了基于CORDIC算法的实现方式，与之不同，本文引入了频域激活函数来替代CORDIC算法实现激活函数。采用频域激活函数不但消除了时域与频域切换的时间开销，还消除了原来使用CORDIC算法计算激活函数时需要先计算双曲函数再进行除法运算的复杂过程。CORDIC算法需要13个时钟周期才能完成一次激活函数的计算，而本文所采用的频域激活函数仅需1个时钟周期即可完成。此外，频域激活函数能够支持四个门函数的激活值同时计算，相比文献[25]中四个激活函数共用一个模块并依次完成，本文所提出的频域激活函数在计算效率上明显高于文献[25]的顺序执行方式。

表1和表2分别对单层LSTM在PYNQ-Z2开发板上的资源利用率和速度表现进行比较。在LSTM层后添加了一个形状为(40,20)的全连接层，用于将LSTM层输出$\boldsymbol{h}_t$转换回时域，从而确保在相同的条件下对不同方法进行评估。实验结果表明，本文在一层LSTM的实现中BRAM资源使用量略高于文献[25]，主要原因在于频域计算涉及复数乘法操作，尤其是在$\boldsymbol{c}_t$和$\boldsymbol{h}_t$的计算过程中，两个复数的乘法被分解为3次元素乘法和5次加法，导致存储需求增加。尽管BRAM资源占用较文献[25]略高，但是仅增加了2个BRAM_18K块，占总BRAM资源的0.7%，本文方法在FIFO和LUT资源消耗方面表现出明显优势，相较于文献[25]FIFO资源减少了7422个，LUT资源减少了10 976个。在系统时钟频率为100 MHz的情况下，本文提出的方法响应时间比文献[25]缩短了28.7%。

若对权重矩阵采取分块循环矩阵的压缩方式，但是将矩阵乘法运算的结果转换回时域，并使

表1　PYNQ-Z2资源利用率对比(一层LSTM)

芯片:Xc7z020	总共	文献[25]	本文
BRAM	280	3/1%	5/1%
FIFO	10 6400	9804/9%	2382/2%
LUT	53 200	16 783/31%	5807/10%
DSP	220	0	0

表2　PYNQ-Z2速度对比(一层LSTM)

	文献[25]	本文
时钟频率	100 MHz	100 MHz
总延迟	46.28 μs	32.99 μs

用CORDIC算法实现激活函数，这种方法被称为时域激活，用来与本文的方法比较。基于与参考文献[25]相同的完整网络参数配置，表3展示了LSTM网络的四种加速器设计在资源消耗方面的对比情况。由表3可知，本文提出的方法相较于文献[24]在消除了DSP模块的同时，显著降低了LUT、FIFO和BRAM的资源利用率。对比参考文献[25]，本文的LUT利用率仅为其1/4，BRAM的利用率降至参考文献[25]的1/6，FIFO的利用率更是锐减至参考文献[25]的1/11，这种资源的减少在保证性能的同时还提高了系统可扩展性。与本文设计相比，采用时域激活方法的BRAM资源消耗略低于本文，因为本文在频域内实现时$\boldsymbol{c}_t$和$\boldsymbol{h}_t$的计算都涉及复数乘法，完成1次复数运算的3次元素乘法需要调用3次CORDIC元素乘法模块，而在时域激活的计算中，$\boldsymbol{c}_t$和$\boldsymbol{h}_t$只需要调用1次CORDIC元素乘法模块。与单层LSTM情况类似，尽管对比时域激活的方法BRAM的使用量略有增加(2个BARM_18K块)，但是FIFO和LUT资源消耗大幅度减少，其中FIFO减少了12 502个，LUT减少了18 685个。表4对不同设计的推理速度进行了对比，在资源消耗减少的情况下时域激活的推理速度相较于文献[25]提高了12.5%，本文LSTM网络的推理速度更是提升了近1倍。这恰恰说明，使用频域激活函数避免了时域/频域切换，大大加快推理速度。

4.3　功　耗

基于PYNQ-Z2开发板对本文设计的LSTM加速器进行了功耗评估并与参考文献[25]对比，如表5所示。在总功耗方面，本文的设计较文献[25]降低了0.119 W，主要因为系统总功耗中PS端功耗占比较大，而本文的优化主要集中在加速器IP核，所以整体功耗的改善幅度相对有限。但在加速器IP核的功耗方面，本文方法相比文献[25]表现出显著优势，将功耗从0.438 W降低至0.079 W，降幅达82%，这一改善主要得益于在LSTM推理过程中硬件资源占用的减少，从而有效减少了IP核的功耗。综上所述，本文设计的LSTM加速器在动态功耗、静态功耗、总功耗和加速器IP核功耗方面均优于参考文献[25]的设计，尤其在加速器IP核的功耗方面得到了明显的降低。

表3　PYNQ-Z2资源利用率对比

	文献[24]	文献[25]	时域激活	本文
BRAM	179.2/64%	82.5/29.46%	10/3%	16/5%
FIFO	69 160/65%	24 042/22.6%	15 581/14%	3079/2%
LUT	51 604/97%	36 285/68.2%	27 731/52%	9046/17%
DSP	180.4/82%	0	0	0

表4　PYNQ-Z2速度对比

	文献[25]	时域激活	本文
时钟频率	100 MHz	100 MHz	100 MHz
总延迟	114 μs	99.7 μs	63.6 μs

表5　PYNQ-Z2功耗对比

	文献[25]	本文
动态功耗(W)	1.705	1.605
静态功耗(W)	0.157	0.138
总功耗(W)	1.862	1.743
加速器IP核功耗(W)	0.438	0.079

4.4　能　效

表6详细列出了在与参考文献[24-25]、[37]中的LSTM加速器在芯片型号、计算精度和操作数上基本保持一致的情况下，本文设计在矩阵乘法延迟、总延迟、吞吐量和功耗方面表现出了显著的优化效果。根据作者调研，目前尚无完全在频域内实现LSTM网络推理的相关实验，所以选取时域LSTM的相关文献作为对比实验。文献[24]的架构加速了LSTM的矩阵乘法部分，将结果卸载到PS端进行激活。文献[25]在PL端执行所有的矩阵乘法和激活函数，从而减少了参数加载和最终结果卸载的I/O通信。文献[37]同样使用CORDIC算法实现LSTM网络的激活函数，与文献[25]不同的是采用了On-The-Fly CORDIC优化方法，通过比较CORDIC算法中当前角度z_i和最近微旋转角度θ_j之间的距离，来跳过不必要的旋转操作。如果当前角

度z_i小于中点$m_j=(\theta_j+\theta_{j+1})/2$则跳过该旋转，否则继续旋转。On-The-Fly CORDIC优化方法能将CORDIC实现激活函数的周期降低50%以上。

在矩阵乘法延迟优化方面，对比文献[25]本文将矩阵乘法的延迟从的30.46 μs大幅缩短至6.96 μs，提升了约4.4倍的计算速度，对比文献[24]矩阵乘法的计算速度更是提高了约6.7倍，在文献[37]的矩阵乘法的参数量是本文一半的情况下，本文矩阵乘法耗时仅比文献[37]慢0.16 μs。这种提升归功于分块循环矩阵对参数的压缩。吞吐量的计算公式定义为：吞吐量=操作数/总延迟，其中操作数计算方法参考文献[24]，文献[24]的操作数为0.1984 M(512×193×2+128×3×2)，由此得到本文的操作数为0.2 M。实验数据表明本文的吞吐量为3.1 GOPS，相比文献[25]的1.8 GOPS提高了1.72倍，较文献[37]的2.46 GOPS提高了1.26倍。功耗优化方面，本文设计的LSTM加速器IP核实现了0.079 W的低功耗，相比参考文献[24]的0.25 W和文献[37]的0.274 W，分别降低了约68.4%和71.2%。在能效方面，本文提出的设计能效为1.78 GOPS/W，参考文献[25]的能效为0.96 GOPS/W。可见本文的能效为文献[25]的约两倍。上述实验数据验证了本文设计在物联网设备及边缘计算平台等场景中的巨大潜力。

表6 与文献[24]、[25]、[37]的性能与功耗对比

	本文	文献[25]	文献[24]	文献[37]
芯片	Xc7z020	Xc7z020	Xc7z020	Xc7z020
频率	100 MHz	100 MHz	150 MHz	200 MHz
精度	Fixed-16	Fixed-16	Fixed-16	Fixed-12
操作数	0.2 M	0.2 M	0.1984 M	-
矩阵乘法延迟	6.96 μs	30.46 μs	46.7 μs	6.8 μs
总延迟	63.6 μs	114 μs	-	-
吞吐量	3.1 GOPS	1.8 GOPS	-	2.46 GOPS
功耗	1.743 W	1.862 W	2.29 W	-
加速器IP核功耗	0.079 W	0.438 W	0.25 W	0.274 W

5 结 论

本文提出了基于频域内推理计算的无乘法器LSTM网络模型，首先采用基于分块循环矩阵的压缩方法，并进一步利用FFT加速计算。为了消除由于FFT带来的频繁的时域/频域切换开销，本文引入了频域激活函数，通过将所有的参数和操作映射到频域。同时利用CORDIC算法代替DSP和FFT运算。实验结果显示，本文设计的LSTM加速器在资源利用率和计算速度方面显著优于参考文献[24-25]、[37]中的方法。具体来说，本文的方法相比文献[25]将矩阵乘法延迟从30.46 μs减少到6.96 μs，总延迟从114 μs减少到63.6 μs，LUT的利用率从68.20%降至17%，FIFO的利用率从22.60%降至2%，而BRAM的利用率则从29.46%降至5%。同时将功耗从1.862 W降低到1.743 W。综上，本研究方法以较少硬件资源代价获取优异性能，这在硬件资源与功耗受限的嵌入式平台上具有重要价值。

参 考 文 献

[1] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780

[2] Billa J. Dropout approaches for LSTM based speech recognition systems//Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing. Calgary, Canada, 2018: 5879-5883

[3] Zhang Yu, Zhang Peng-Yuan, Yan Yong-Hong. Long short-term memory with attention and multitask learning for distant speech recognition. Journal of Tsinghua University (Science and Technology), 2018, 58(3): 249-253 (in Chinese)
(张宇，张鹏远，颜永红. 基于注意力LSTM和多任务学习的远场语音识别. 清华大学学报：自然科学版，2018, 58(3): 249-253)

[4] Yao L, Guan Y. An improved LSTM structure for natural language processing//Proceedings of the 2018 IEEE International Conference of Safety Produce Informatization. Chongqing, China, 2018: 565-569

[5] Liu Jian-Wei, Wang Yuan-Fang, Luo Xiong-Lin. Research and development on deep memory network. Chinese Journal of Computers, 2021, 44(8): 1549-1589 (in Chinese)
(刘建伟，王园方，罗雄麟. 深度记忆网络研究进展. 计算机学报，2021, 44(8): 1549-1589)

[6] Siami-Namini S, Tavakoli N, Namin A S. The performance of LSTM and Bi-LSTM in forecasting time series//Proceedings of the 2019 IEEE International Conference on Big Data. Los Angeles, USA, 2019: 3285-3292

[7] Merenda M, Porcaro C, Iero D. Edge machine learning for ai-enabled iot devices: A review. Sensors, 2020, 20(9): 2533

[8] Ren Jie, Gao Ling, Yu Jia-Long, et al. Energy-Efficient deep learning task scheduling strategy for edge device. Chinese Journal of Computers, 2020, 43(3): 440-452 (in Chinese)
(任杰，高岭，于佳龙 等. 面向边缘设备的高能效深度学习任务调度策略. 计算机学报，2020, 43(3): 440-452)

[9] Wu D, Xu H, Jiang Z, et al. EdgeLSTM: Towards deep and sequential edge computing for IoT applications. IEEE/ACM Transactions on Networking, 2021, 29(4): 1895-1908

[10] Dai X, Yin H, Jha N K. Grow and prune compact, fast, and

accurate LSTMs. IEEE Transactions on Computers, 2019, 69(3): 441-452

[11] Rybakov O, Kononenko N, Subrahmanya N, et al. Streaming keyword spotting on mobile devices. arXiv preprint arXiv: 2005.06720,2020

[12] Jiao Li-Cheng, Sun Qi-Gong, Yang Yu-Ting, et al. Development, implementation and prospect of FPGA-basedDeep neural networks. Chinese Journal of Computers, 2022, 45(3):441-471(in Chinese)
(焦李成，孙其功，杨育婷 等. 深度神经网络 FPGA 设计进展，实现与展望. 计算机学报，2022, 45(3):441-471)

[13] Chang A X M, Martini B, Culurciello E. Recurrent neural networks hardware implementation on FPGA. arXiv preprint arXiv:1511.05552, 2015

[14] Li S, Wu C, Li H, et al. FPGA acceleration of recurrent neural network based language model//Proceedings of the 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines. Vancouver, Canada, 2015: 111-118

[15] Ding C, Liao S, Wang Y, et al. Circnn: Accelerating and compressing deep neural networks using block-circulant weight matrices//Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. Boston, USA, 2017: 395-408

[16] Ko J H, Mudassar B, Na T, et al. Design of an energy-efficient accelerator for training of convolutional neural networks using frequency-domain computation//Proceedings of the 54th ACM/EDAC/IEEE Design Automation Conference. Austin, USA, 2017: 1-6

[17] Ho N M, Wong W F. Exploiting half precision arithmetic in Nvidia GPUs//Proceedings of the2017 IEEE High Performance Extreme Computing Conference. Waltham, USA, 2017: 1-7

[18] Lee M, Hwang K, Park J, et al. FPGA-based low-power speech recognition with recurrent neural networks//Proceedings of the2016 IEEE International Workshop on Signal Processing Systems. Dallas, USA, 2016: 230-235

[19] Han S, Kang J, Mao H, et al. ESE: Efficient speech recognition engine with sparse LSTM on FPGA//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.New York, USA, 2017: 75-84

[20] Lu Z, Sindhwani V, Sainath T N. Learning compact recurrent neural networks//Proceedings of the2016 IEEE International Conference on Acoustics, Speech and Signal Processing. Shanghai, China, 2016: 5960-5964

[21] Ioannou L, Fahmy S A. Streaming overlay architecture for lightweight LSTM computation on FPGA SoCs. ACM Transactions on Reconfigurable Technology and Systems, 2022, 16(1): 1-26

[22] Que Z, Zhu Y, Fan H, et al. Mapping large LSTMs to FPGAs with weight reuse. Journal of Signal Processing Systems, 2020, 92: 965-979

[23] Judd P, Albericio J, Hetherington T, et al. Stripes: Bit-serial deep neural network computing//Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture. Taipei, China, 2016: 1-12

[24] Zhang X, Jiang W, Hu J. Achieving full parallelism in LSTM via a unified accelerator design//Proceedings of the 2020 IEEE 38th International Conference on Computer Design. Hartford, USA, 2020: 469-477

[25] Mohamed N A, Cavallaro J R. A unified parallel CORDIC-based hardware architecture for LSTM network acceleration. IEEE Transactions on Computers, 2023, 72(10): 2752-2766

[26] Mathieu M, Henaff M, LeCun Y. Fast training of convolutional networks through ffts. arXiv preprint arXiv:1312.5851, 2013

[27] Zhang C, Prasanna V. Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. New York, USA,2017: 35-44

[28] Zeng H, Chen R, Zhang C, et al. A framework for generating high throughput CNN implementations on FPGAs//Proceedings of the 2018 ACM/SIGDA international symposium on field-programmable gate arrays. New York, USA, 2018: 117-126

[29] Rippel O, Snoek J, Adams R P. Spectral representations for convolutional neural networks//Proceedings of the 29th International Conference on Neural Information Processing Systems.Cambridge, USA, 2015: 2448-2457

[30] Francesca M, Hughes A, Gregg D. Spectral convolution networks. arXiv preprint arXiv:1611.05378, 2016

[31] Lin J, Ma L, Cui J. A frequency-domain convolutional neural network architecture based on the frequency-domain randomized offset rectified linear unit and frequency-domain chunk max pooling method. IEEE Access, 2020, 8: 98126-98155

[32] Ayat S O, Khalil-Hani M, Ab Rahman A A H, et al. Spectral-based convolutional neural network without multiple spatial-frequency domain switchings. Neurocomputing, 2019, 364: 152-167

[33] Liu S, Fan H, Luk W. Design of fully spectral CNNS for efficient FPGA-based acceleration. IEEE Transactions on Neural Networks and Learning Systems, 2024, 35(6), 8111-8123

[34] Wang S, Li Z, Ding C, et al. C-LSTM: Enabling efficient LSTM using structured compression techniques on FPGAs//Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.New York, USA, 2018: 11-20

[35] Wang Jian. Chinese text sentiment analysis based on deep learning[Master's dissertation]. National University of Defense Technology, Changsha,Hunan,2017(in Chinses)
(汪健. 基于深度学习的中文文本情感分析[硕士学位论文]. 国防科技大学,湖南,长沙,2017)

[36] Devore J L. Probability and statistics for engineering and the sciences. 8th ed.Boston,USA: Cengage Learning, 2011

[37] Luo Z, Mai J, Yao E. OTFC-LSTM: An efficient design of LSTM accelerator based on on-the-fly CORDIC//Proceedings of the 2024 IEEE 6th International Conference on AI Circuits and Systems. Abu Dhabi, United Arab Emirates, 2024: 372-376

JIN Song, Ph. D., associate professor. His research interests include VLSI integrated circuits design and intelligent information processing

CHEN Shi-Qi, M. S. candidate. Her research interests include integrated circuit design, intelligent information and speech signal processing.

Background

The problem addressed in this paper belongs to the field of deep learning acceleration, specifically focusing on the acceleration of inference computations for Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks. RNNs excel at processing sequential data, such as speech recognition and natural language processing, but their high computational and memory demands often lead to challenges related to latency and excessive resource consumption. Currently, cloud computing and edge computing are widely used to address these issues, but cloud computing often faces latency and privacy risks when handling real-time applications. While edge computing can reduce latency, it requires more advanced hardware, and accelerating RNNs remains difficult.

Currently, there have been some international efforts to accelerate CNN inference using specialized hardware, such as FPGA and ASIC, but these specialized architectures cannot be directly applied to accelerate RNN inference. This is because RNNs involve complex neuron connections, large memory footprints, and the need to frequently update internal states, making acceleration more challenging. Existing solutions are mainly focused on CNN acceleration, with relatively fewer studies on RNN acceleration. The main challenges for RNN acceleration are optimizing memory management and improving computational efficiency.

This paper presents a hardware accelerator design for LSTM networks based on frequency domain and multiplication-free operations. It utilizes cyclic block matrices to compress and store the network's weight parameters, and combines FFT, IFFT, and frequency-domain activation functions to implement full-frequency domain network inference, avoiding the frequent overhead of switching between time-domain and frequency-domain when processing different time samples. The CORDIC algorithm is used to replace complex multiplication and hyperbolic function calculations in butterfly operations, achieving low-power hardware deployment for LSTM.