第 55 卷　第 5 期
2019 年 9 月

南京大学学报(自然科学版)
JOURNAL OF NANJING UNIVERSITY
(NATURAL SCIENCE)

Vol. 55, No. 5
Sept., 2019

DOI:10.13232/j.cnki.jnju.2019.05.004

一种负载均衡的LSTM硬件加速器设计

查　羿,潘红兵*

(南京大学电子科学与工程学院,南京,210093)

摘　要:神经网络在嵌入式端的应用日益广泛,为满足嵌入式端低功耗,低延迟等特点,通常的解决方案是针对长短记忆序列LSTM模型(Long-Short Term Memory)进行压缩,并定制专用的硬件加速器.当LSTM模型经过剪枝等压缩操作后,其网络模型将变得稀疏且不规则,会给PE(Process Element)运算单元带来负载不均衡的问题.通过排序的方法,将权重矩阵按一定的规则重新分发给各个PE单元,并在此基础上针对稀疏化的模型定制专用的硬件单元.在赛灵思zynq系列XCZU9EG-2FFVB1156E开发板上进行实验,实验结果显示,当PE单元多消耗0.314%硬件资源的情况下,其运算速度取得了2%的提升.

关键词:神经网络加速器,模型压缩,负载均衡,嵌入式设计

中图分类号:TN492　　**文献标识码:**A

A load balanced LSTM hardware accelerator design

Zha Yi, Pan Hongbing*

(School of Electronic Science and Engineering, Nanjing University, Nanjing, 210093, China)

Abstract: Neural network is increasingly widely used in embedded end. In order to meet the characteristics of low power consumption and low delay of embedded end, the common solution is to compress LSTM (Long-Short Term Memory) model and customize special hardware accelerator. When LSTM model is compressed by pruning, its network model will become sparse and irregular, which will bring unbalanced load to PE (Process Element) operation unit. In this paper, weight matrix is redistributed to each PE unit according to certain rules by sorting method. On this basis, a special hardware unit is customized for the sparse model. The zynq series xczu9eg-2ffvb1156e development board was used for experiments, and the experimental results showed that the PE unit achieved a 2% improvement in computing speed when it consumed 0.314% more hardware resources.

Key words: neural network accelerator, model compression, load balance, embedded design

RNN(Recurrent Neural Network)的网络模型在语言识别[1-2]、机器翻译等领域应用十分广泛,如南京理工大学针对自然语言处理任务提出一种新型的训练RNN的方法(Light-RNN)[3],中国电子科技大学提出块状张量分解(Block-Term tensor decomposition, BT-RNN)的结构[4],韩国科学技术学院提出CNN-RNN混合加速核[5],并针对动态定点数进行基于查

基金项目:国家自然科学基金(61376075)

收稿日期:2019－05－06

*通讯联系人,E-mail:liningrigs@vip.sina.com

找表的可重构乘法器设计. 而LSTM (Long-Short Term Memory)[6]作为其变体解决了RNN训练中梯度爆炸的问题,因此LSTM得到了广泛的应用,针对LSTM的硬件加速器的设计种类繁多[7−8].

由于LSTM模型的计算复杂度和空间复杂度较大,模型运行的响应时间长,所需的存储空间大,很难在资源受限的嵌入式设备中应用. 针对此问题,ESE[9],EIE[10]将现有的模型进行剪枝和量化的压缩以减少片上存储的空间,同时缓解了数据从片外DDR预取带来的访存瓶颈和延时,使得片外的参数一次性地搬运到片上缓存,并针对剪枝后的LSTM网络模型定制硬件加速器. 此外,也有一些其他的权重压缩方式,如循环矩阵压缩、分块循环矩阵压缩[11]. 另外,北京大学提出一种C-LSTM的架构,利用结构化的压缩技术降低LSTM模型尺寸,以块循环矩阵取代稀疏矩阵,在加速推断中使用FFT算法[12]. Conti et al[13]提出一种脉动的、可拓展的LSTM硬件加速器(CHIPMUNK),其原型芯片能达到3.08 Gop•s^{-1}的运算速度,在0.75 V的工作电压下功耗只有1.24 mW. 东南大学针对RNN提出一种高能效的可配置架构(E-ERA)[14],采用近似的乘法器. 与一般的架构相比,在语音识别任务中,E-ERA能获得1.78倍的能效比,其能效达到了304 GOPS•W^{-1}. Shin et al[15]针对在光学字符识别任务中的双向LSTM模型,提出专用的硬件架构,此硬件架构同样可处理单向的LSTM模型,具有一定的灵活性. 在光学字符识别任务中,该硬件加速器相较于目前工艺提升了459倍的数据吞吐率,在166 MHz的时钟频率下运算速度能够达到152.16 GOPS.

LSTM算法主要是矩阵向量乘,而模型经过权值剪枝后权重矩阵变得稀疏,稀疏矩阵向量乘则构成了神经网络运算的基本单元. 为了利用权重矩阵的稀疏性,避免0元素的数据预取带来的时钟数的浪费,剪枝后的权重进行0元素的剔除,将非零元素及其对应的索引值共同存储至片外DDR,再由运算单元的控制器进行读取解码运算,以交织的方式分配给各个PE(Process Element)(前32行矩阵依次分配给PE0-PE31,后面的行以此类推). 常见的编码方式有CSC和CSR等以及韩国浦项科技大学提出的CBSR的编码方式[16]. CSC形式存储了非零元素值,以及其索引值、地址指针. 然而,这样的编码方式会导致解码方式的繁琐,并且交织分配PE的方式会使得各个PE的加载不均衡,从而影响整个系统的运算性能.

本文主要解决上述问题,进而最大化地实现LSTM模型的推断加速. 在软件端,通过排序算法根据稀疏矩阵各个行的非零元素个数,对稀疏矩阵的各个行进行重新排序,并均衡地将重排后的矩阵的行分配到各个PE单元内. 针对每个PE单元内部,本文将同一列的权重值拼凑并存储到同一个地址空间,与非零权重元素共同存储的还有它们的行索引. 在硬件架构端,将激励值通过非零检测模块过滤,权重值经过解码模块进行解码,再将二者通过运算单元进行乘加操作. 经过此番优化,对于1024×153大小的权重矩阵,其运算性能上比负载不均衡情况下,提升了几百纳秒. 并且随着矩阵的规模越大,稀疏度在50%左右(0值分布极度不均衡条件下),本文的优化方法取得的效果最好,在消耗少量额外的硬件资源的情况下,运算速度提升了2%.

1 LSTM背景介绍

LSTM隐藏层的核心设计,是一种叫记忆体(cell state)的信息流,它负责把记忆信息从序列的初始位置传递到序列的末端,并通过四个互相交互的“门”单元来控制每一个时间步 t 对记忆信息的修改,包含了输入门、遗忘门、输出门等多个门结构体(如图1所示). 这些门结构体用于对输入信息的筛选,从而使得信息有选择地去影响当前时刻的神经网络状态,而门函数也就是激活函数,一般为sigmoid或relu等函数.

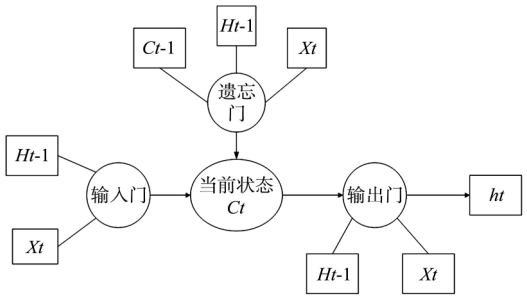

图1　LSTM单元结构示意图

Fig. 1　Schematic diagram of LSTM unit structure

LSTM有庞大的参数量，对于实时性和功耗要求高且没有很多存储空间的嵌入式端，本文需要对LSTM网络进行剪枝和参数的量化，才能将其映射到定制的硬件架构上.

2　算法及硬件设计的优化

本实验通过软硬件结合的方式对LSTM网络进行加速，在软件层面对LSTM模型在保证精度的前提下进行剪枝量化操作，在硬件层面针对稀疏化后的LSTM模型定制LSTM硬件加速器，使用较少的存储资源，获取较高的运算速度以及较高的能效比.

2.1　权重矩阵的重排　假设LSTM加速器中有四个PE(事实上，PE的数量由片外DDR4的带宽所决定，满足存储带宽恰好能够喂饱PE运算单元)，对于给定的8×8的稀疏矩阵权重矩阵，其重排过程分四步处理.

步骤一，首先统计出每一行的非零权重值的个数，用以分析该行的实际计算量. 每一行的非零元素的个数被统计下来作为“R_len”的变量. 同时，对每一行以“R_id”进行编号，因为后续中，为了PE均匀分配，要将矩阵的行打乱重新排序，所以要保存原来的行次序.

步骤二，该步骤为最关键的一步，正是因为这一小小的改变，满足了PE计算加载的均匀分布. 将矩阵的行按照其包含的非零元素的个数(即R_len)由多到寡重新排序，这使得PE阵列优先处理具有较多非零元素的行数，然后再平衡地将PE映射到各行，以保证每个PE处理具有相同R_len的行数. 然而，矩阵行的重新调整也需要相应地改变原有的激励向量中元素排布的次序. 相应的，ht等输出向量的内部元素次序也会受到影响.

另外，如果不同权重矩阵中的行序号变换的次序不一致，如[$Wx1$, $Wh1$]的(第一行和第三行交换次序)和[Wxc, Whc]的(第一行和第二行交换)，那么输入激励[Xt, Ht-1]的向量元素无法通过调换以匹配所有权重，状态机Ct的值更新则不同步，因而，为了最大化LSTM硬件加速器的吞吐率，应同步调整所有权重矩阵的行，保持它们的行调整后的元素对应与之前的矩阵是一致的. 这里需要一个简单的矩阵结构转换机制，如图2所示.

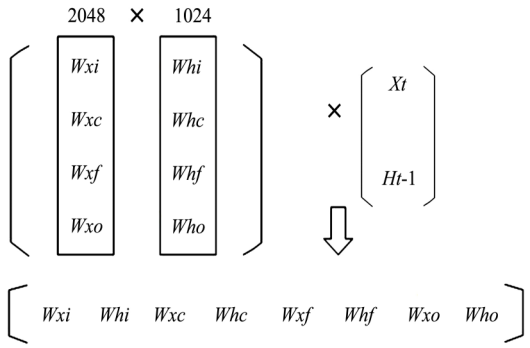

图2　权值矩阵的重组拼凑

Fig. 2　Weight matrix recombination

步骤三，这一步主要决定PE与矩阵的匹配对应情况，即哪一个PE来运算重新排序后的矩阵的哪几行. 每一个运算结果对应Ht的一个状态的更新，而Ht又成为下一层LSTM的输入. 如果矩阵的总行数比PE的数量更多，每个PE则处理多行的矩阵. 如果矩阵的前四行分别由PE0，PE1，PE2，PE3由低到高的次序来运算，那么矩阵的第5～8行分别由PE3，PE2，PE1，PE0来计算. 倘若矩阵行数为12，那么，矩阵的第9行到第12行再由PE0，PE1，PE2，PE3来运算，以此类推. 这样做的目的是为了保证每个PE所加载的非零数据量近似相等. 如图3的step3所示.

步骤四，PE任务的主要分配已经在步骤三完成，it，ft，Ot等门的权重对(即[Wxi, Whi]，

	0	1	2	3	4	5	6	7
0		A				B	C	
1				D	E			
2			F					
3	G	H		I				
4		J				K		L
5							M	
6					N			O
7							P	

Step 1

R_id	R_len
0	3
1	2
2	1
3	3
4	3
5	1
6	2
7	1

Step 2

R_id	R_len
0	3
3	3
4	3
1	2
6	2
2	1
5	1
7	1

Step 3

PE	R_id	R_len
0	0	3
1	3	3
2	4	3
3	1	2
3	6	2
0	2	1
1	5	1
2	7	1

图 3 权值矩阵重新排序以及 PE 的分配过程(以上步骤 1 到 3 的总概括图)

Fig. 3 Weight matrix reordering and PE allocation (general overview of step1 through 3 above)

$[Wxf, Whf]$,$[Wxo, Who]$,$[Wxg, Whg]$)事先预取到片上,针对每一个 PE 内所处理的权重矩阵进行编码,取出所需要的非零元素值,将同一列的权重值打包成数据块(低四位为行索引)存到一片的地址区,以 0 元素为该区的结束符.

2.2 LSTM 加速器硬件架构

2.2.1 总体架构 LSTM 加速器的总体架构如图 4 所示,主要包含以下几个部分:稀疏矩阵向量乘单元、累加单元、激活函数单元、存储单元和按元素相乘单元. 选定 DDR4 的型号时,它的吞吐率也随之确定. 为了最大化地提升整个 LSTM 加速器的吞吐率,应排布尽可能多的 PE 进行运算. 假设 PE 的个数为 n,CLKPE 为 PE 的时钟周期,则有下式:

$$\frac{\text{单个PE一个时钟的数据量} \times n\text{PE}}{clk\text{PE}} \leqslant DDR\text{吞吐率}$$

根据计算,选择 32 个 PE 组成 PE 运算的阵列.

LSTM 有四个门:输入门 it、遗忘门 ft、输出门 Ot、候选门 gt,可以看到每一个门的运算过程大致分为如下:稀疏矩阵向量乘运算、向量与向量加法运算、非线性函数. 而存储状态单元 Ct 则是向量与向量乘法运算,故可将 LSTM 硬件加速器分为四个运算模块:

(1)稀疏矩阵向量乘(SpMV):用于计算稀疏矩阵向量乘运算.

(2)累加器(Accumulator):用于计算向量与向量的加法运算.

(3)元素乘(ElemMul):用于计算向量与向量元素乘.

(4)非线性函数(Sigmoid/Tanh):用于计算对应的非线性函数,采用分段线性函数和查找表结合的方式.

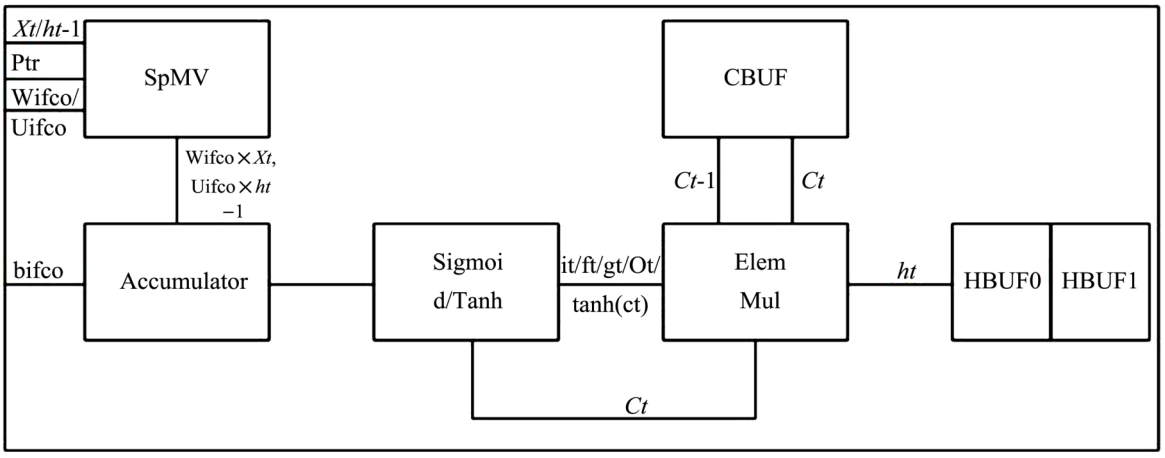

图 4 LSTM 算法的运算模块示意图

Fig. 4 Schematic diagram of operation module of LSTM algorithm

2.2.2 数据读取运算流程 下面介绍 LSTM 控制器的数据流和各个状态,如图 5 所示.

由 LSTM 主控制器发出开始信号,读控制模块向 DDR4 发送读请求信号,从 DDR4 中读取需要运算的参数和激励数据,根据预先的仲裁协议,分别存放至片上的权重缓存 buffer 和

激励数据缓存buffer中,控制器调度PE将缓存器中的数据传输至PE单元内的bram中进行运算.运算后的结果送入累加器进行累加.

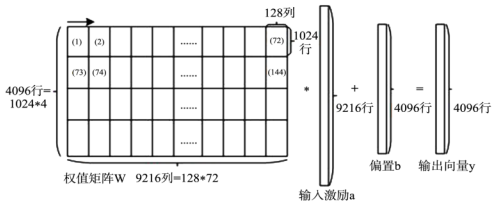

图5　矩阵向量乘分块运算

Fig. 5　Matrix vector multiplication and partitioning

为了提高运算流程的并行效率,引入流水线操作.由于片上存储资源的有限,不可能把所有的权重和激励都存放至片上,故庞大的矩阵与向量相乘不可能一次完成.因而,需要对原有的大矩阵进行划分,划分成多个行和列,如棋盘一样.以4096×9216的大矩阵为例,当状态机处于initial状态时,读控制模块进行第一块小矩阵的数据读取,读完之后,跳入状态2,在该状态下进行下一块(2)的权重参数读取和当前块(1)片上bram中数据的运算,从而形成两级流水.直到运算第一行的最后一个矩阵块时,进入状态3的换行状态:第一行所有矩阵块运算结束,输出最终结果向量的一部分,同时读取下一行的第一块矩阵.当运算结果和数据读取同时完成,重新跳回到状态2.运算矩阵第二行的数据.当全部矩阵运算完成后,进入结束状态.具体的运算流程参见图6.

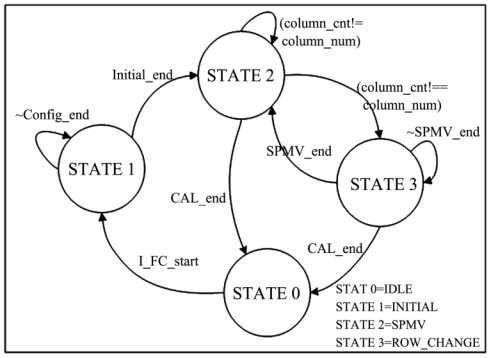

图6　稀疏矩阵向量乘运算的状态图

Fig. 6　Sparse matrix vector multiplication state diagram

2.2.3　稀疏矩阵向量乘单元结构　稀疏矩阵向量乘单元运算量最大,所消耗的运算资源和存储资源最多,运算的时长也最多.为了提高运算的速度,应尽可能利用权重和激励的稀疏度.首先,将权重及其相对的行地址索引值存储到PE内部的bram中,然后依次送入激励值,每个时钟送入一个激励,并对激励通过非零检验模块,跳过0元素值,将非零激励值(和它在激励向量中的索引值)同时分发到32个PE的激励FIFO缓冲区内,每个PE接收到一个激励.同一时刻内,32个PE接收到的激励值是同一个值.在每一个PE内,根据激励的行索引值,在weight_ctrl模块内解锁出与激励相对应的所有权重值(事先约定好,同一列的权重值存储在bram的同一个地址中,bram的地址序号即为权重所在列的列号),然后在arithmetic模块内,将权重分别与激励相乘,其结果保存在arithmetic模块内的累加器中(参见图7).

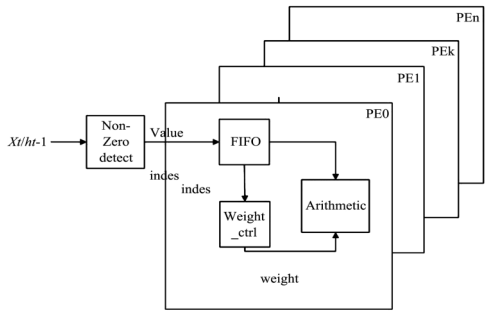

图7　单个PE的SpMV单元内部模块

Fig. 7　Internal module of single PE SpMV unit

3　实验内容及结果分析

3.1　验证工具　本文的验证方案采用vivado工具综合并在Modelsim中仿真测试得到该硬件加速器具体的速度、面积与功耗等.

3.2　硬件加速器的测试比较方案　在matlab模型中生成若干个不同稀疏度的权重矩阵作为测试数据,通过排序算法将其均匀地分配到各个PE中,将同一列的非零数据(以及它们的行相对索引)拼凑成一个大的数据块,彼此不同行的数据之间以特定的分隔符分隔开来,然后将

测试数据导入 vivado 内的稀疏矩阵向量乘 PE 硬件单元中运行. 经过综合工具综合出其硬件电路图,并得出资源报告分析图. 再将原权重矩阵数据直接以 EIE 中的 PE 交织方式分配给 PE,再以 CSC 编码方式进行编码,然后导入到硬件模型中,与本文之前软硬件协同优化后的设计方案在运算速度方面作对比.

3.3 性能评估 在 LSTM 的算法中,分为稀疏矩阵向量乘、向量累加、向量与向量元素乘、非线性激活函数等操作步骤,其中复杂度最高、运算量最大、消耗的运算存储资源最多的就是稀疏矩阵向量乘模块,LSTM 运算过程中,该模块占据了大量的运算时间,故将稀疏矩阵向量乘模块单独进行讨论. 表 1 中的数据来自文献[1],分别将稀疏矩阵向量乘在 CPU,GPU 等硬件平台上实现,记录其性能数据,并与本设计进行对比. 比较的对象是 1024×153 的矩阵,其稀疏度为 88.3%,系统时钟为 100 MHz.

表 1 不同平台进行稀疏矩阵向量乘运算的性能比较

Table 1 Performance comparison of sparse matrix vector multiplication operations on different platforms

	本设计	ESE	GPU	CPU
运算时间(μs)	6.68	5.36	14.5	167.6
加速比	25.09×	31.27×	11.56×	1×
功耗(W)	7.10	41	202	38
功耗比	0.18×	1.07×	5.32×	1×
能效比(加速比/功耗比)	139.3×	29.22×	2.17×	1×

1024×153 的稀疏矩阵,其对应的非零权值为 1024×153×11.7%=18331 个. 当系统时钟为 100 MHz 时,周期为 10 ns,且 32 个 PE 并行处理,则计算该稀疏矩阵的理论消耗时间(极限时间)为 18331/32×10 ns=2864 ns=5.72 μs. 而本论文中所设计的稀疏矩阵向量乘单元其运算时间为 6.68 μs,接近理论消耗时间. 具体结果参见表 2.

可以看到,当权值矩阵的稀疏度为 50% 左右时,本设计的效果最好,但稀疏度过小或过大,则负载均衡处理后的矩阵的运算优势不是十分明显. 这是因为稀疏度过大(零值较多)时,PE 所处理的运算数据本来就十分少,运算的时长也很少,或者说,每个 PE 更多地分到了 0 数据,这也是一种“相对负载均衡”(多为 0 元素)的体现. 而当稀疏度过小(零值较少,只有 10% 左右)时,则各个 PE 单元处理的非零数据很多,每个 PE 中非零元素的个数也相对均衡饱和,所以本文的均衡化设计处理的优势也不那么明显.

表 2 稀疏度不同的矩阵效果对比

Table 2 Comparison of matrix effects with different sparsity

	稀疏度(零元素值所占比例)	运算时间(μs)	差值
负载不均衡	0.883	6.84	0.16
本设计		6.68	
负载不均衡	0.483	25.95	0.665
本设计		25.285	
负载不均衡	0.083	45.02	0.06
本设计		44.96	

4 结 论

为了最大限度实现运算系统的吞吐率,减少神经网络剪枝带来的 PE 单元负载不均衡,将 LSTM 所有权重矩阵合并在一个大的矩阵中,统计出每行的非零权重值,根据非零权重值的个数将矩阵的行重排,再均匀分配到各个 PE 中以保证 PE 的负载计算均衡. 再将各个 PE 处理

的权重数据及其行索引值保存至片上bram. 根据激励的地址索引,引出对应的权重,进行卷积操作. 当权值的稀疏度在50%左右时,本文的PE处理效果最佳,速度提升了2%左右.

参考文献

[1] Graves A, Mohamed A R, Hinton G. Speech recognition with deep recurrent neural networks// Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. Vancouver, Canada: IEEE, 2013: 6645－6649.

[2] Rybalkin V, Wehn N, Yousefi M R, et al. Hardware architecture of bidirectional long short-term memory neural network for optical character recognition//Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017. Lausanne, Switzerland: IEEE, 2017.

[3] Cho K, Van Merrienboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv: 1406.1078, 2014.

[4] Cho K, Van Merrienboerning B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv: 1406.1078, 2014.

[5] Ye J M, Wang L N, Li G X, et al. Learning compact recurrent neural networks with block-term tensor decomposition. arXiv: 1712.05134, 2018.

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

[7] Guan Y J, Yuan Z H, Sun G Y, et al. FPGA-based accelerator for long short-term memory recurrent neural networks//2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC). Chiba, Japan: IEEE, 2017: 629－634.

[8] Ouyang P, Yin S Y, Wei S J. A fast and power efficient architecture to parallelize LSTM based RNN for cognitive intelligence applications//2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). Austin, TX, USA: IEEE, 2017: 1－63.

[9] Han S, Kang J L, Mao H Z, 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. Monterey, CA, USA: ACM, 2017.

[10] Han S, Liu X Y, Mao H Z, et al. EIE: Efficient inference engine on compressed deep neural network//2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). Seoul, South Korea: IEEE, 2016: 243－254.

[11] Wang Z S, Lin J, Wang Z F. Accelerating recurrent neural networks: A memory-efficient approach. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 25(10): 2763－2775.

[12] Wang S, Li Z, Ding C W, 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. Monterey, CA, USA: ACM, 2017.

[13] Conti F, Cavigelli L, Paulin G, et al. CHIPMUNK: A systolically scalable 0.9 mm^2, 3.08 Gop/s/mW @1.2mW accelerator for near-sensor recurrent neural network inference. arXiv: 1711.05734, 2018.

[14] Liu B, Dong W, Xu T T, et al. E-ERA: An energy-efficient reconfigurable architecture for RNNs using dynamically adaptive approximate computing. IEICE Electronics Express, 2017, 14(15): 20170637.

[15] Shin D, Lee J, Lee J, et al. 14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks//2017 IEEE International Solid-State Circuits Conference (ISSCC). San Francisco, CA, USA: IEEE, 2017.

[16] Park J, Kung J, Yi W, et al. Maximizing system performance by balancing computation loads in LSTM accelerators//2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). Dresden, Germany: IEEE, 2018.

(责任编辑　章　强)