文章编号：2096-1618(2022)04-0374-05

基于 FPGA 的 RNN 硬件加速架构

相博锏， 凌味未， 李 鑫， 邹金成
(成都信息工程大学通信工程学院，四川 成都 610225)

摘要：针对边缘计算场景下，循环神经网络消耗计算资源过多，且计算流程相对复杂所导致的计算效率较低的问题，提出一种 RNN 模型的硬件加速方法，并在 FPGA 平台对该方法进行验证。为在计算资源可复用的前提下尽可能提高计算速度，该加速器利用一种 SIMD 指令集，通过软件编程的形式来配置运算流程，适配不同层数和维度的 RNN 及其相关模型。还根据 RNN 模型数据流的特点，对加速器设计进行优化，并设置合理的片内缓存方式和并行逻辑以充分利用存储器带宽，降低资源开销。实验结果表明，加速器在100 MHz的工作频率下运算性能达到6.7 GOPS，需要的功耗为2.15 W。基于指令集和软硬件协同的方式对两种网络模型进行实现，速度是微控制器的230倍。

关 键 词：微电子学与固体电子学；硬件加速器；可编程逻辑器件；循环神经网络；指令集架构
中图分类号：TP183 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2022.04.002

0 引言

随着近年深度学习技术的发展，神经网络模型高昂的计算成本对硬件平台提出了更高的要求。在神经网络模型中，向量和矩阵之间的乘法运算在整个模型运算量中占了最大的比重。由于不同元素间的乘法不具有相关性，因此非常适合利用并行处理来实现硬件运算加速。文献[1]证明了基于 FPGA 的 RNN 加速器能获得更高的能源效率；文献[2]利用 HLS 工具能够在 FPGA 平台对 LSTM 快速实现；文献[3]提出基于 FPGA 部署压缩后的 LSTM 模型的方法，该方法所实现的语音识别模型推理速度比 Core i7 5930CPU 和 Pascal Titan X GPU 分别快43倍和3倍；文献[4]提出针对 CNN 的硬件加速指令集架构，在100 MHz的频率下运算性能达到41.73 GOPS；文献[5]着重关注硬件加速器的计算性能和通信要求，提出的架构在实验中峰值性能达到7.26 GFLOP/s。

以往基于 FPGA 的 RNN 硬件加速器主要聚焦在软件层面的模型压缩、硬件层面的数据存储模式、稀疏矩阵实现、流水线设计和并行层面的计算架构，针对的主要是单一类型的神经网络模型硬件实现，这类加速方法往往局限性较强，如利用 Winograd 优化算法来加速 CNN 的方法[7]，仅适用于卷积计算中步长为1的情况。文献[8]根据特定网络模型，直接将计算流程硬件化，后期却难以进行优化或是兼容其他相关模型。本文在已有研究的基础上，提出一种基于 SIMD 指令集的通用 RNN 加速器架构。该架构通过软硬件协同方式，能够适配 RNN 及其变体 GRU、LSTM 等模型，并且基于多级流水线设计来优化数据流在模块间的传输，尽可能地提高运算速度。最后对两种 RNN 模型进行实现，并与相关的神经网络模型计算平台进行对比分析。

收稿日期：2021-11-03
基金项目：国家自然科学基金资助项目(61201094)

1 RNN 结构分析

1.1 RNN 模型

RNN 及其变体 LSTM 和 GRU 的运算流程如图1所示。

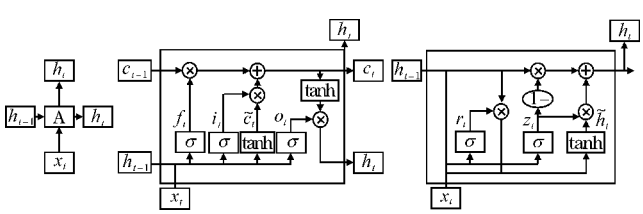

图1 RNN 运算流程

在 RNN 模型中，当前时刻的输出不仅取决于当前时刻输入，也和先前的输入序列有关。给出输入序列 $\boldsymbol{X}=(\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_t)$，RNN 计算得到输出序列 $\boldsymbol{Y}=(\boldsymbol{y}_1,\boldsymbol{y}_2,\cdots,\boldsymbol{y}_t)$，计算过程可以表示为

$$\boldsymbol{h}_t=\sigma(\boldsymbol{W}_{xh}\boldsymbol{x}_t+\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1}+\boldsymbol{b}_h) \tag{1}$$

$$\boldsymbol{y}_t=\boldsymbol{W}_{hy}\boldsymbol{h}_t+\boldsymbol{b}_y \tag{2}$$

其中，$\boldsymbol{W}_{xh}$、$\boldsymbol{W}_{hh}$和 $\boldsymbol{W}_{hy}$分别表示输入层与隐藏层之间的权重矩阵、隐藏层间传播的权重矩阵及输出层和隐藏层之间的权重矩阵。

LSTM 模型通常用来解决 RNN 梯度消失、梯度爆炸及长时间依赖问题。式(3)~(6)中遗忘门 $\boldsymbol{f}_t$、输入门 $\boldsymbol{i}_t$、候选值 $\tilde{\boldsymbol{c}}_t$ 和输出门 $\boldsymbol{o}_t$ 分别由输入向量与权重作矩阵乘法得到,这些中间结果在式(7)、(8)中作 Hadamard 积,得到最终结果单元状态 $\boldsymbol{c}_t$ 和隐藏层输出 $\boldsymbol{h}_t$。其运算过程为

$$\boldsymbol{f}_t=\sigma(\boldsymbol{W}_{fx}\boldsymbol{x}_t+\boldsymbol{W}_{fh}\boldsymbol{h}_{t-1}+\boldsymbol{b}_f) \tag{3}$$

$$\boldsymbol{i}_t=\sigma(\boldsymbol{W}_{ix}\boldsymbol{x}_t+\boldsymbol{W}_{ih}\boldsymbol{h}_{t-1}+\boldsymbol{b}_i) \tag{4}$$

$$\tilde{\boldsymbol{c}}_t=\tanh(\boldsymbol{W}_{cx}\boldsymbol{x}_t+\boldsymbol{W}_{ch}\boldsymbol{h}_{t-1}+\boldsymbol{b}_c) \tag{5}$$

$$\boldsymbol{o}_t=\sigma(\boldsymbol{W}_{ox}\boldsymbol{x}_t+\boldsymbol{W}_{oh}\boldsymbol{h}_{t-1}+\boldsymbol{b}_o) \tag{6}$$

$$\boldsymbol{c}_t=\boldsymbol{f}_t\circ\boldsymbol{c}_{t-1}+\boldsymbol{i}_t\circ\tilde{\boldsymbol{c}}_t \tag{7}$$

$$\boldsymbol{h}_t=\boldsymbol{o}_t\circ\tanh(\boldsymbol{c}_t) \tag{8}$$

GRU 模型相对 LSTM 有更精简的结构,因此参数规模更小,但实验证明二者精度相差并不高。在 GRU 模型中,遗忘门和输入门被融合为一个更新门 $\boldsymbol{z}_t$,而重置门 $\boldsymbol{r}_t$ 用来控制前一状态 $\boldsymbol{h}_{t-1}$ 的保留比例,候选值 $\tilde{\boldsymbol{h}}_t$ 和隐藏层结果 $\boldsymbol{h}_t$ 为

$$\boldsymbol{r}_t=\sigma(\boldsymbol{W}_{rx}\boldsymbol{x}_t+\boldsymbol{W}_{rh}\boldsymbol{h}_{t-1}+\boldsymbol{b}_r) \tag{9}$$

$$\boldsymbol{z}_t=\sigma(\boldsymbol{W}_{zx}\boldsymbol{x}_t+\boldsymbol{W}_{zh}\boldsymbol{h}_{t-1}+\boldsymbol{b}_z) \tag{10}$$

$$\tilde{\boldsymbol{h}}_t=\tanh[\boldsymbol{W}_{\tilde{h}x}\boldsymbol{x}_t+\boldsymbol{W}_{\tilde{h}h}(\boldsymbol{r}_t\circ\boldsymbol{h}_{t-1})+\boldsymbol{b}_{\tilde{h}}] \tag{11}$$

$$\boldsymbol{h}_t=(1-\boldsymbol{z}_t)\circ\boldsymbol{h}_{t-1}+\boldsymbol{z}_t\circ\tilde{\boldsymbol{h}}_t \tag{12}$$

1.2　硬件加速分析

由式(1)~(12)知,RNN 模型的计算主要以乘法、加法及三角函数运算组成。在 RNN 的实际应用中,通常是矩阵和向量乘法部分运算量最大并且耗时最长[9]。因此,首先需要考虑如何充分利用并行特点来设计加速器,以减轻 CPU 的计算压力。

对于中型和大型的 RNN 模型,通常需要较多的权重参数,而常规 FPGA 器件内部资源常常无法满足存储需求,因此需要借助外部存储器,如利用 DDR3 SDRAM 作为权重的存储介质。

更深的层数和更宽的维度常常能够给神经网络带来更强的抽象能力,但也存在着大量冗余[10]。文献[4]进行了基于负载平衡的剪枝,使硬件运算单元能够更高效地实现稀疏矩阵运算。

对模型权重进行低位宽定点数量化处理,能使加速器进一步提高计算速度、降低对计算能耗和存储带宽的需求,同时适当的量化方式在推理运算中几乎不会对 RNN 性能造成损失[11]。

每层模型通常会利用激活函数来保证网络的非线性特性,如 GRU 和 LSTM 会使用 Sigmoid 和 Tanh 来处理控制门的运算结果。对于其中三角函数类运算进行硬件实现时,有 CORDIC、查表法、分段拟合法等方式,不同方法所带来的延时、精度和资源消耗也会不同。

2　系统实现

2.1　加速器模块架构

为保证模型的通用性,本文提出一种针对 RNN 的硬件加速架构,并在存储和计算模块上加以优化。对模型运算进行加速时,需要根据模型运算特点对加速器进行编程,使其工作时能够以软硬件协同的方式对整个运算流程实现控制;通过以太网或 PCIe 可以实现外部设备和本系统的高速数据通信,利用 DDR3 SDRAM 实现片外高速存储,其中利用以太网可以向加速器交换输入值和推理结果,DDR3 可以向加速器输入神经网络模型权重或暂存中间数据。

加速器架构如图 2 所示。在推理计算过程中,Controller 模块由总线上的 Register 模块获取所编程的运算控制参数和指令。推理过程中,Controller 模块将待运算的向量由片内 SRAM 移至向量 Buffer,再并行送入计算单元 MAC Array 作并行乘-累加运算,计算结果由 MUX 逻辑写回向量 Buffer 作为中间结果暂存,或作为最终推理结果由 SRAM 缓冲并读回外设。

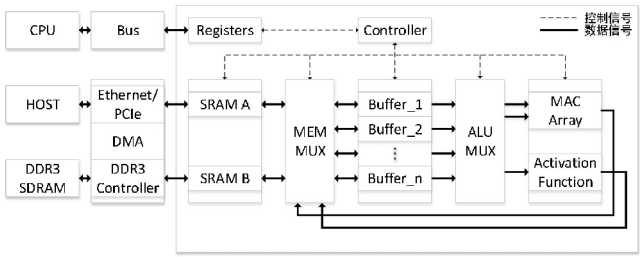

图 2　加速器架构

在加速器的结构设计中,存储单元(SRAM、Buffer)和计算单元(MAC Array、Activation)并没有强耦合在一起,而是分成两组模块,并利用 Controller 模块来控制其数据流向。这种设计方式尽管在细节上难达到其他针对某一模型专用加速器高效,但基于这样的 SIMD 指令集,能够在软件层面更容易通过编程来支持多种神经网络模型,在硬件层面方便未来对计算或存储模块进行迭代。在数据流的设计上,使用多缓冲器和深度流水线能够进一步提高加速器的吞吐率,更灵活地利用片上存储空间。

相较于训练时的浮点数运算,在硬件电路中使用基于定点数的运算模块速度更快并且面积更小,但量化过程会带来一定程度的精度损失。本文提出的加速器运算单元使用 16-bit 定点数的量化方式,其中最高位为符号位,其余分别为 5 位整数位和 10 位小数位。

MAC Array 的基本结构见图 3。由缓冲器读出待

计算的 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 后,先由并行乘法器实现对应元素相乘,再将乘积通过加法树求和,最后进行累加。MAC Array 为了适配更多模型类型的计算需求,还支持对输入向量进行线性预处理操作,并且乘法、加法树、累加功能也都可以分别被旁路。

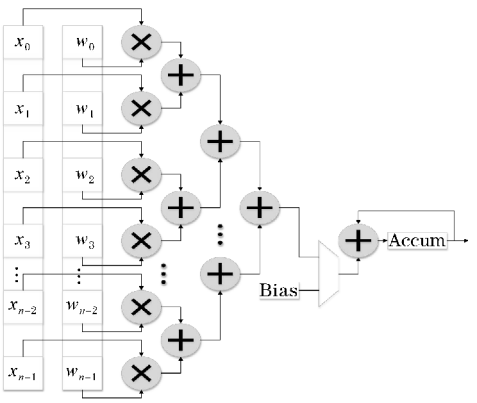

图 3 MAC 运算阵列结构

除了上述的乘法和加法运算单元以外,还对常用的 tanh 和 sigmoid 激活函数运算进行硬件设计。为兼顾计算过程中低延时、高精度的特性,做到电路资源的节约,基于分段二次函数拟合,通过流水线方式分别对两种激活函数进行实现。对于式(13),在 3-bit 整数,12-bit 小数的量化方式下,使用二次函数分段拟合,其硬件实现用式(14)表示,仿真得到平均误差为1.03×10^{-4}。而对于 Softmax 函数中的指数运算部分,则是通过先对输入分区间再利用查找表的方式直接求结果。

$$\mathrm{Sigmoid}(x)=\frac{1}{1+e^{-x}} \tag{13}$$

$$f(x)=\begin{cases}0.245x+0.5 & 0<x\leqslant 0.5\\ -0.045x^2+0.285x+0.491 & 0.5<x\leqslant 2.45\\ -0.017x^2+0.15x+0.658 & 2.45<x\leqslant 4\\ -0.004x^2+0.048x+0.854 & 4<x\leqslant 5.6\\ 0.0015x+0.989 & 5.6<x\leqslant 7.5\\ 1 & x\geqslant 7.5\\ 1-f(-x) & x\leqslant 0\end{cases} \tag{14}$$

2.2 软硬件协同设计

为支持多种类型的模型运算,本文基于一种 SIMD 指令集对加速器结构进行设计,使其可以利用软件编程的方式,通过 CPU 向总线上的寄存器写入控制参数或指令。如运算过程中的乘法、累加、激活函数等运算操作及各缓冲器和运算单元之间的数据,可以由程序指令配置。而对于逻辑复杂但运算量不大的操作,则仍由 CPU 独立进行。根据操作类型的不同,指令集主要可分为存储指令、运算指令和配置指令。

表 1 展示了加速器的核心指令,LDR 和 STR 指令需要和 DMA 模块配合来实现加速器和外部存储器之间的数据交换;MUL 和 MLA 指令操作乘法运算阵列,实现向量间的运算;ACT 指令用来实现激活函数;OUT_CH 和 IN_CH 为 MLA 运算指令配置模型特征通道个数。本文软硬件协同设计的主要目的是根据实际神经网络模型特点,将传统的细粒度、同质化的通用矩阵乘法、激活函数算法的控制方式进行优化。

表 1 加速器核心指令

Name	Byte3	Byte2	Byte1	Byte0
LDR_EXT	0x20	Buf sel	length[15:8]	length[7:0]
STR_EXT	0x21	Buf sel	length[15:8]	length[7:0]
MUL	0x60	Res buf	buf_0|buf_1	length[7:0]
MLA	0x61	y buf sel	b buf sel	x buf sel
ACT_SGD	0x70	o_buf sel	i_buf_sel	—
ACT_TAH	0x71	o_buf sel	i_buf_sel	—
OUT_CH	0xa1	—	length[15:8]	length[7:0]
IN_CH	0xa2	—	length[15:8]	length[7:0]

图 4 是基于本文提出的指令集,对一层神经网络模型运算进行编程的伪代码。在各模块配置完毕后,通过操作 DMA 模块使加速器预先装载 Bias 和 $\boldsymbol{X}$ 向量。接下来执行 MLA 指令,加速器 Controller 模块按照先前初始化好的操作参数,分多次将缓冲器中的权重矩阵与输入向量相乘。乘积会自动进入加法树与累加器,运算结果向量被保存到缓冲器中,最后由选择的 Sigmoid 激活函数模块执行非线性运算。

```
Initialization();
DMA_Run(b_addr, length);
LDR_EXT(buf_b_sel, length);
DMA_Run(x_addr, length);
LDR_EXT(buf_x_sel, length);
MLA(y_buf, b_buf, x_buf);
DMA_Run(w_addr, length);
LDR_EXT(buf_w_sel, length);
ACT_SGD(buf_o_sel, buf_i_sel, length);
```

图 4 程序指令实现软硬件协同

3 仿真与测试

3.1 实验方案

本文以 Xilinx KC705 板卡作为硬件平台。为更好地对时序和控制逻辑进行优化,加速器的设计未使用 HLS 工具,而是在 RTL 层面进行实现。系统使用 ARM Cortex-M0 软核 IP 作为 CPU,频率为50 MHz;加速器模

块工作在100 MHz频率下,模型权重由以太网传入板卡,并储存于 DDR3 SDRAM,由程序为模型提供初始输入向量。加速器使用 16-bit 量化,FPGA 板卡使用 DDR3 SDRAM 配置在6.4 GB/s的带宽,加速器选择以32 路并行乘法器组成 MAC 阵列,在100 MHz的频率下以流水线方式运行。由此,乘法运算理论峰值达到3.2 G/s来匹配6.4 GB/s的存储器带宽。由于计算单元开始工作时,需要先对数据进行预读取,所带来的开销使实际性能无法达到理论性能。此外,在加速器工作过程中,由于模型自身的特点,向量长度可能不是2的整数次幂,并行乘法阵列存在空闲,从而会造成一定的性能损失。

利用表 1 提出的指令集,将基于 LSTM 的一种字符推理模型 Char-RNN 和基于 GRU 的一种音符推理模型 TonicNet 作为实验的模型原型,分别利用两种编程方式,将两种模型在加速器上进行实现。不包括激活函数及预处理和后处理中的编码,两种 RNN 模型需要的运算量主要集中在向量运算,其中一次前向运算所需要的乘法和加法次数见表 2。

表 2　模型运算量统计

运算类型	Char-RNN	TonicNet
乘法次数	239K	1235K
加法次数	240K	1236K
总运算量	479K	2471K

3.2　实验结果

实验将两种神经网络模型分别部署到通用 CPU Core i7-4720HQ、微控制器内核 Cortex-M0 及加速器上进行运算,并统计所需时间。其中通用 CPU 使用 PyTorch 框架搭建模型;微控制器内核运行程序进行串行计算;加速器模块与微控制器内核利用软硬件协同的方式,按译码后的指令实现并行加速运算。通用 CPU 使用 Python 内置的 timer 库计算程序消耗时间,微控制器和加速器使用额外定时器模块统计所需时间。

实验结果如表 3 所示,装载了加速器的 FPGA 平台利用软硬件协同的计算方式,速度约为仅使用微控制器内核的 230 倍,约是通用 CPU 的 7 倍以上。

表 3　不同硬件平台消耗时间对比

模型	Core i7-4720HQ/ms	Cortex-M0 50 MHz/ms	FPGA Accelerator/μs
Char-RNN	1.2	24.5	102.7
TonicNet	3.9	142.3	512.1

同软件模型相比,由于加速器需要对原先模型中的浮点数据进行 16-bit 定点量化,因此推理过程会带来新的误差。在以 TonicNet 为原型进行实验时,硬件加速器的隐藏层结果 $\boldsymbol{h}_t$ 对 PyTorch 的软件运算结果误差单层平均在3.2%。

表 4 是加速器中各类资源的使用情况。相比其他加速器设计,本文提出的硬件加速架构对 FPGA 片上各类资源要求较少,在拥有强可复用性和较低功率的条件下计算性能达到6.7 GOPS。

表 4　资源使用和功耗情况

参数	文献[15]	文献[5]	本文
FPGA 平台	ZCU102	Pynq-Z2	KC705
频率/MHz	100	100	100
LUT	28657	24663	25161
BRAM	571	112	81
DSP	356	44	70
计算精度	fixed-16	fixed-32	fixed-16
性能/GOPS	13.17	41.73	6.7
功耗/W	2.089	2.1	2.15

4　结束语

提出一种基于 FPGA 的 RNN 硬件加速架构,该架构能够基于一套 SIMD 指令集,通过软硬件协同的方式对不同类型的 RNN 模型实现运算加速,不需要重新修改硬件架构,从而简化神经网络于硬件加速器部署的流程。提出的方案与同级别性能的 FPGA 硬件加速器相比,使用较少的片内资源,达到主流的能耗比。加速器在100 MHz的工作频率下,计算性能约为 Cortex-M0 的 230 倍,是通用 CPU 的 7 倍以上。

参考文献:

[1] Nurvitadhi E, Sim J, Sheffield D, et al. Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC[C]. International Conference on Field Programmable Logic & Applications. IEEE, 2016.

[2] Rybalkin V, Wehn N, Yousefi M R, et al. Hardware architecture of Bidirectional Long Short-Term Memory Neural Network for Optical Character Recognition[C]. 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2017.

[3] Han S, Kang J, Mao H, et al. Ese: Efficient speech

recognition engine with sparse lstm on fpga[C]. Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017:75-84.

[4] 邓良,陈章进,乔栋,等. 基于 FPGA 的指令集架构神经网络协处理器的设计与验证[J]. 小型微型计算机系统,2021,42(6):1129-1135.

[5] Guan Y, Yuan Z, Sun G, et al. FPGA-based accelerator for long short-term memory recurrent neural networks[C]. 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE,2017.

[6] 高琛,张帆. 基于 FPGA 的递归神经网络加速器的研究进展[J]. 网络与信息安全学报,2019,5(4):1-13.

[7] Yu J, Hu Y, Ning X, et al. Instruction driven cross-layer CNN accelerator with winograd transformation on FPGA[C]. 2017 International Conference on Field Programmable Technology(ICFPT),2018.

[8] Sun Z, Zhu Y, Zheng Y, et al. FPGA acceleration of LSTM based on data for test flight[C]. 2018 IEEE International Conference on Smart Cloud (SmartCloud). IEEE,2018:1-6.

[9] 范军,巩杰,吴茜凤,等. 基于 FPGA 的 RNN 加速 SoC 设计与实现[J]. 微电子学与计算机,2020,37(11):1-5.

[10] Han S, Mao H, Dally W J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding[J]. Fiber,2015,56(4):3-7.

[11] Lee M, Hwang K, Park J, et al. FPGA-Based Low-Power Speech Recognition with Recurrent Neural Networks[C]. 2016 IEEE International Workshop on Signal Processing Systems(SiPS). IEEE,2016.

[12] Song H, Mao H, Dally W J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding[C]. ICLR,2016.

[13] 彭井桐,祝永新,汪辉,等. 基于 FPGA 的 GRU 神经网络飞行数据异常检测[J]. 微电子学与计算机,2021,38(11).

FPGA-based Hardware Accelerator for RNN

XIANG Boqiang, LING Weiwei, LI Li, ZOU Jincheng

(College of Communication Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: Aiming at the problem of low computing efficiency caused by the excessive consumption of computing resources and the complex computing process of Recurrent Neural Network in edge computing scenarios, the authors proposed a hardware acceleration method of RNN models and verified it on FPGA platform. To improve the processing speed under the premise that computing resources can be reused, the accelerator uses an SIMD instruction set, which can configure the operation flow in the form of software programming, and can adapt to different layers and dimensions of RNN and its related models. Moreover, according to the properties of RNN model's data flow, the authors optimized the accelerator architecture with providing on-chip caches and parallel logic circuits to make full use of memory bandwidth and simultaneously reduce overhead. Experiments show that the performance of the accelerator reaches 6.7 GOPS at 100 MHz and the power consumption is 2.15 W. The two network models are implemented based on instruction set with the cooperation of software and hardware, and the speed is 230 times faster than that of the microcontroller.

Keywords: microelectronics and solid state electronics; hardware accelerator; FPGA; RNN; ISA