ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.14, No.15, 1–11

E-ERA: An energy-efficient reconfigurable architecture for RNNs using dynamically adaptive approximate computing

Bo Liu[a)], **Wei Dong, Tingting Xu, Yu Gong, Wei Ge, Jinjiang Yang, and Longxing Shi**

National ASIC System Engineering Technology Research Center, Southeast University, Nanjing 210096, P.R.C

a) *liubo_cnasic@seu.edu.cn*

Abstract: This paper proposes an Energy-Efficient Reconfigurable Architecture (E-ERA) for Recurrent Neural Networks (RNNs). In E-ERA, reconfigurable computing arrays with approximate multipliers and dynamically adaptive accuracy controlling mechanism are implemented to achieve high energy efficiency. The E-ERA prototype is implemented on TSMC 45 nm process. Experimental results show that, comparing with traditional designs, the power consumption of E-ERA is reduced by 28.6%~52.3%, with only 5.3%~9.2% loss in accuracy. Compared with state-of-the-art architectures, E-ERA outperforms up to 1.78X in power efficiency and can achieve 304 GOPS/W when processing RNNs for speech recognition.

Keywords: reconfigurable architecture, recurrent neural network, dynamically adaptive accuracy

Classification: Integrated circuits

References

[1] X. Liang, *et al.*: "Semantic object parsing with graph LSTM," ECCV (2016) 125 (DOI: 10.1007/978-3-319-46448-0_8).

[2] Z. Fang, *et al.*: "Comparison of different implementations of MFCC," J. Comput. Sci. Technol. **16** (2001) 582 (DOI: 10.1007/BF02943243).

[3] T. Mikolov, *et al.*: "Extensions of recurrent neural network language model," ICASSP (2011) 5528 (DOI: 10.1109/ICASSP.2011.5947611).

[4] A. Graves, *et al.*: "Hybrid speech recognition with deep bidirectional LSTM," ASRU (2013) 273 (DOI: 10.1109/ASRU.2013.6707742).

[5] A. X. M. Chang, *et al.*: "Recurrent neural networks hardware implementation on FPGA," arXiv preprint (2015) (arXiv:1511.05552).

[6] H. Sharma, *et al.*: "From high-level deep neural models to FPGAs," MICRO (2016) 1 (DOI: 10.1109/MICRO.2016.7783720).

[7] T. Chen, *et al.*: "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," ASPLOS (2014) 269 (DOI: 10.1145/2541940.2541967).

[8] Y. Chen, *et al.*: "Dadiannao: A machine-learning supercomputer," MICRO

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

(2014) 609 (DOI: 10.1109/MICRO.2014.58).
[9] S. Zhang, *et al.*: "Cambricon-X: An accelerator for sparse neural networks," MICRO (2016) 1 (DOI: 10.1109/MICRO.2016.7783723).
[10] Y. H. Chen, *et al.*: "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," ISCA (2016) 367 (DOI: 10.1109/ISCA.2016.40).
[11] M. Tanomoto, *et al.*: "A CGRA-based approach for accelerating convolutional neural networks," MCSoC (2015) 73 (DOI: 10.1109/MCSoC.2015.41).
[12] M. Pietras: "Error analysis in the hardware neural networks applications using reduced floating-point numbers representation," AIP Conf. Proc. **1648** (2015) 660005 (DOI: 10.1063/1.4912881).
[13] M. Horowitz: "1.1 Computing's energy problem (and what we can do about it)," ISSCC Dig. Tech. Papers (2014) 10 (DOI: 10.1109/ISSCC.2014.6757323).
[14] Z. Babić, *et al.*: "An iterative logarithmic multiplier," Microprocess. Microsyst. **35** (2011) 23 (DOI: 10.1016/j.micpro.2010.07.001).
[15] Y. Miao, *et al.*: "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," ASRU (2015) 167 (DOI: 10.1109/ASRU.2015.7404790).
[16] D. Wang and X. Zhang: "THCHS-30: A free Chinese speech corpus," arXiv preprint arXiv (2015) (arXiv:1512.01882).
[17] T. Oliphant: *A Guide to NumPy* (Trelgol Publishing, USA, 2006).
[18] S. Li, *et al.*: "FPGA acceleration of recurrent neural network based language model," FCCM (2015) 111 (DOI: 10.1109/FCCM.2015.50).
[19] S. Han, *et al.*: "EIE: Efficient inference engine on compressed deep neural network," ISCA (2016) 243 (DOI: 10.1109/ISCA.2016.30).

1 Introduction and related works

Recurrent Neural Networks (RNNs) are widely used in Artificial Intelligence, especially tackling the problem in learning sequences of information. RNN is a powerful model for processing sequential data [1]. In speech recognition, the filters used for extracting input features are usually overlapped, meaning feature extraction frames are overlapped between any two adjacent frames [2]. In Mikolov's work, they optimized RNNs for language models and reduced the network size in a certain degree [3]. By Grave's publication [4], the Deep Bidirectional LSTM (DBLSTM) RNN shows superior performance in tasks where acoustic model predominates [4].

However, with the ever increasing requirements on performance of RNNs, the hardware platforms are facing more challenges. Chang's [5] team successfully mapped LSTM RNN to FPGA platform and performs 21x faster than ARM Cortex-A9. But FPGAs are expensive and the programming is time consuming. DNNWEAVER [6], proposed by Sharma, et al., is a remarkable framework that can automatically generate a synthesizable accelerator for a given (DNN, FPGA) pair from a high-level specification in Caffe. But the platform stays in FPGA and the flexibility after generator is limited. Chen, et al. published serial work of DianNao [7], DaDianNao [8] and Cambricon-X [9], concentrating on DNNs, vDNNs and Sparse DNNs, which are specified DSPs with dedicatedly defined

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

instruction sets. With higher demands on area and energy efficiency, new architectures are introduced to accelerating RNNs. Reconfigurable architectures show superior in both performance and flexibility while processing certain neural networks. Eyeriss [10] by Chen, et al. is a stream driven energy efficient architecture which can be reconfigured according to dataflow structures. Tanomoto, et al. [11] proposed a coarse-grained reconfigurable architecture (CGRA) for accelerating convolutional neural networks and achieved 1.93x higher performance per memory bandwidth and 2.92x higher area efficiency than mobile GPUs and multicore CPUs respectively. However, as RNNs are showing greater impacts, there is still a lack of researches on reconfigurable computing architectures for RNNs.

In this paper, we propose a reconfigurable architecture named E-ERA which can be effective in terms of both efficiency and flexibility, including reconfigurable approximate computing units and a dynamically adaptive accuracy controlling mechanism which catered to different precision requirements of RNNs. The rest of this paper is organized as followed. Section 2 presents the architecture of proposed reconfigurable computing array and the dynamically adaptive accuracy controlling. Several case studies and the implementation results are shown in section 3 and this paper is concluded in section 4.

2 Architectures of E-ERA for RNNs

As networks are becoming huge, researches on network compression with proper processing precisions in both the software model side and the hardware architecture side are going deeper. RNNs have been proven to be naturally fault tolerant, and calculation accuracy requirements for various application scenarios are also in large variations [11, 12]. Thus, an Energy-Efficient Reconfigurable Architecture (E-ERA) is proposed, including reconfigurable approximate computing arrays with low energy cost and high processing performance, and self-adaptive approximate computing approach to monitoring and dynamically adjusting the precision of computing.

2.1 Reconfigurable approximate computing unit

In RNNs, the operation numbers of additions and multiplications are almost the equal, however the power consumption of multiplications can account for 96% [13]. Thus, a convincing idea to reduce power consumption for processing RNNs is to improve the energy efficiency of multiplication operations.

Despite of their high accuracy, conventional tree multipliers have problems in reducing area and energy consumption. Thus approximate multipliers are adopted because they can significantly improve energy efficient with little cost in accuracy. In this paper, we propose a reconfigurable computing array integrated with approximate multipliers which can be dynamically configured for different calculation accuracies. Moreover, the trade-offs between computing performance and accuracy for different scenarios are also studied.

Based on the MA algorithm, Z. Babic proposed an iterative logarithmic multiplier [14]. The calculation method is described as follows.

$$N = 2^k + x \tag{1}$$

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

Each multiplicator can be divided into two parts as showed in (1), where k is a characteristic number or the place of the most significant bit with the value of 1. Therefore, the product of two multipliers can be written as follows.

$$P = N_1 * N_2 = 2^{k_1+k_2} + 2^{k_1} * x_2 + 2^{k_2} * x_1 + x_1 * x_2 \quad (2)$$

In (2), the first three terms can be derived through the shift operation, the last term is the error term. If further accuracy is required, we can repeat the proposed multiplication procedure with the new multiplicands x_1 and x_2.

As shown in Fig. 1, the Reconfigurable Approximate Computing Unit (RACU) is designed with dynamic reconfigurable calculation accuracy. As the iteration numbers of approximate multiplier increase, the computing results will become more accurate. Thus, an iteration controller is implemented in RACU and responsible for dynamically managing the iteration numbers on demand. Therefore, RACU can achieve significant reduction in power consumption compared to these with traditional multipliers, especially when low accuracy calculations can meet the requirements.

As shown in Fig. 1, when two 16-bit inputs enter the Basic Computing Block, two LODs and Encoders extract the characteristic numbers (k_1 and k_2) and rest of inputs (x_1 and x_2). The demultiplexers guide the multiplicands and multipliers to the appropriate block. For example, if the iteration number is 2, the multiplicands and multipliers will be sent to *BasicComputingBlock_A* by demultiplexers. The multiplexers inside each Basic Computing Block configure the data sources either from the former Basic Computing Block or the demultiplexers directly. All the configuration contexts are written in ITC_{sig}, which is the iteration number control signal (characterized by thick lines). The signal controls both the demultiplexers and the multiplexers.

Although the maximum relative errors of calculations with iteration number 0 is as high as 25%, the average relative error (AER) is just around 9% in experiments [14]. When the iteration number is 3, the AER of computing results is only 0.10%, which is far more than acceptable for most cases. In fact that the sequences or frames entering into RNNs are usually forward-backward correlated, the computing accuracy requirements are considered various between frames and continuous inner

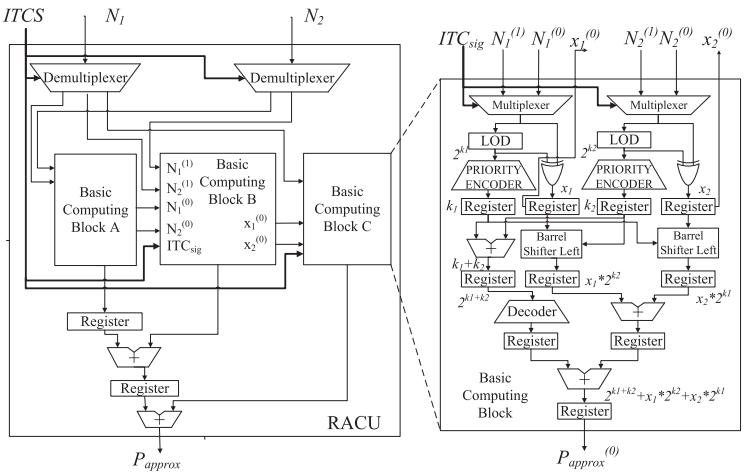

Fig. 1. RACU with dynamic reconfigurable calculation accuracy

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

frame. Therefore, the proposed RACU is implemented with configurable iteration numbers ranging from 0 to 2. In this work, all calculations use 16-bit fixed-point numbers. The details of dynamically adaptive accuracy controlling mechanism is discussed later in next section.

2.2 Dynamically adaptive accuracy controlling based on neuron feedbacks

Referring to the scene of speech recognition where RNNs are usually used, the frequency of speech sampling is much higher than that of changing phonemes in speeches. Therefore, each phoneme sustains for several frames, which leads to the continuity of frames. Due to the correlation inner data, we are able to estimate the calculation accuracy required at time step $t+1$ based on the outputs' quality at time step t. In this work, the valid bits of outputs are regarded as qualities. Meanwhile, by evaluating the qualities of outputs, calculation accuracy requirements of next time step can be predicted. Therefore, the iteration numbers of RACU can be reconfigured dynamically by evaluating results of the former time step.

The trend of RNN maximum output values in speech recognition is evaluated and shown in Fig. 2. It can be seen that in 'phonemes switch period' the maximum output value is low, while in the value is pretty high, which means that the confidence of the RNN network in 'phonemes sustain period' is high. In fact, in these 'phonemes sustain periods', approximate calculations with a much lower accuracy are also acceptable.

The control flow chart of proposed dynamically adaptive accuracy controlling mechanism is shown in Fig. 3 and the parameters are explained in Table I. The iteration number of RACU for time step $t+1$ is dynamically predicted and configured depend on the maximum value of outputs at time step t. According to the range of the network outputs, $n-1$ thresholds are preset, and the outputs will be classified into n levels accordingly. Each level corresponds to a certain calculation accuracy level and each indicates a corresponding iteration number of

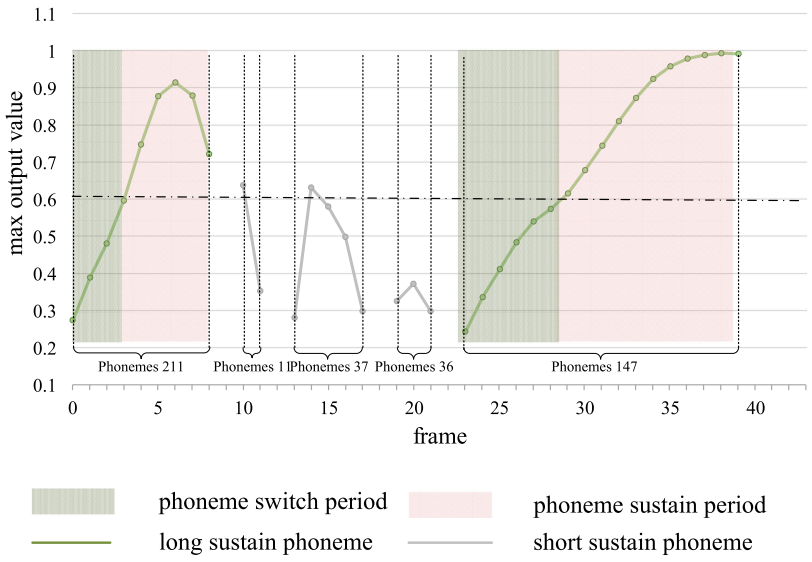

Fig. 2. Trend of RNN maximum output values in speech recognition

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

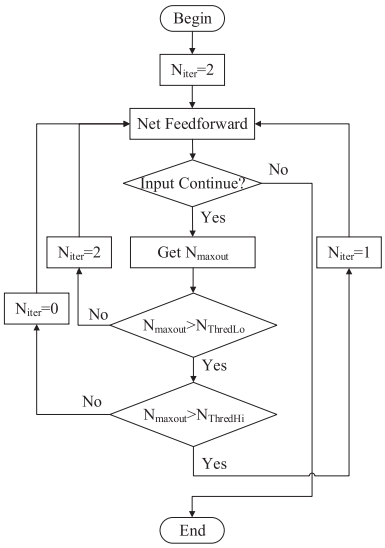

Fig. 3. Control flow chart of self-adaptive RACU

Table I. Parameters explanation in adaptation procedure

N_{Iter}	iteration numbers using in the next frame
N_{maxout}	The max value of the output vector
$N_{ThredLo}$	To decide whether to use 2 iterations, when the N_{maxout} is less than $N_{ThredLo}$, the iteration numbers used in next frame should be 2
$N_{ThredHi}$	To decide whether to use 0 iterations, when the N_{maxout} is higher than $N_{ThredHi}$, the iteration numbers used in next frame should be 0

RACU. In this work, we set n as 3 and thus there are three calculation accuracy levels can be chosen and configured, including the high-accuracy-level (iteration number is 2), the mid-accuracy-level (iteration number is 1) and the low-accuracy-level (non iteration). In addition, the network synapses preprocessing is not required in this work and the control signal of iteration numbers only requires 2 bits.

As shown in Fig. 4, when computing in low-accuracy-level, no extra iteration operation for RACU is required. The multiplicands and multipliers are transferred directly to *BasicComputingBlock_C* by demultiplexers. For mid-accuracy-level, the factors are sent to *BasicComputingBlock_B* and then the outputs are routed to *BasicComputingBlock_C* where the same operation will be iterated one time again. For high-accuracy-level, first, the computing process starts at *BasicComputing-Block_A* and the outputs will be routed to *BasicComputingBlock_B*; then, the operation will be iterated once at *BasicComputingBlock_B* and the outputs will be routed to *BasicComputingBlock_C*; finally, iteration operation will be perform once again at *BasicComputingBlock_B* and the computing results will be obtained. To be specific, all the results are generated from *BasicComputingBlock_C* finally. The Basic Computing Blocks drawn with dotted line in Fig. 4 indicate that there is not any data loaded into these modules, so we can use clock-gating to avoid level

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

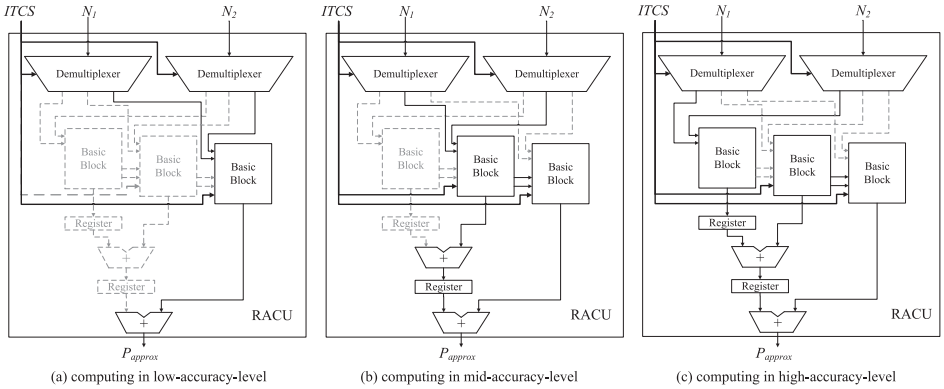

Fig. 4. Control flow chart of dynamically adaptive accuracy controlling mechanism

flip, thereby reducing the dynamic power consumption of RACU. As mentioned above, the iteration number used by RACU is determined by the preset thresholds and the network outputs. When the network output value is higher than the threshold value, the RNN network confidence can be considered as high enough, so RACU can use fewer iterations to computing the results. During the whole computing process, the configuration of RACU computing iteration numbers always follows the network outputs, so the calculation accuracy can be dynamically adaptable to the RNN networks.

2.3 Reconfigurable computing array based on RACU

In this work, a reconfigurable computing array (RCA) with 4*8 RACUs is implemented to process the RNN networks. As shown in Fig. 5, the RACU(mul) and RACU(add) are configured to process multiplication and addition operations respectively. For mapping the RNN networks, the RCA routing structure is configured as shown in Fig. 5. All RACU(add) units are interconnected and the output of each RACU(add) will be finally accumulated and transferred to the output FIFO. The input FIFO loads data from the Data Memory after the previous calculation, and the calculation results are transferred to the Data Memory via output FIFO. As shown in Fig. 5, the Configuration Controller consists of two parts: the Predefined Configuration Context part loads configuration context stored in the Configuration Memory, including the *Ls_Ctx* which controls the memory access address of the FIFOs, the *Rt_Ctx* which controls the RCA routings, and the *Op_Ctx* which controls the RACU operations (e.g. multiplication or addition); the Dynamic Generated Configuration Context part are used to configure multiplication iteration numbers used by RACU dynamically based on the former outputs of RCA. When one frame calculation is completed, the Iterative Determiner will generate the iteration numbers used in the next frame calculation by evaluating current network output and write this information to the *It_Ctx*. In the Iterative Determiner, the network output is transferred to two adders, and the other inputs of the two adders are the negative values of the two preset thresholds (the adder finishes subtracting the threshold). Only the sign bits of the adders' result (*Sgn_a* and *Sgn_b*) is needed to determine the iteration numbers.

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

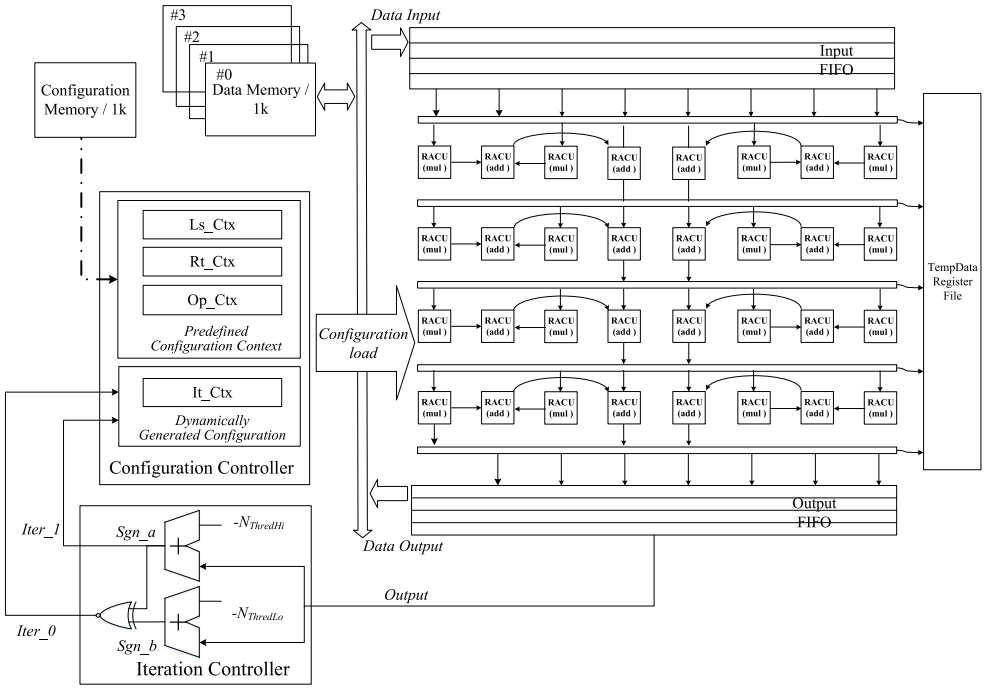

Fig. 5. Architecture of Reconfigurable Computing Array

The top level architecture of an E-ERA prototype system is as shown in Fig. 6. It consists of a system controller implemented with ARM7TDMI, a scratch-pad memory (SPM) with a size of 32 KBytes, two RCAs for accelerating RNN networks, and several assistant modules, including an Interrupt Controller (IntCtl), a Direct Memory Access Controller (DMAC), and an External Memory Interface (EMI) with a data I/O width of 64 bits. All of the modules are AMBA2.0-AHB-compatible and connected to the 32-bit AHB Bus module, used as the system bus.

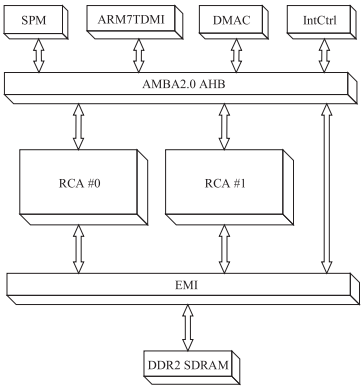

Fig. 6. Prototype system of E-ERA

3 Sample case study and simulation results

To verify the proposed work, a RNN network is trained using EESEN [15]. We use the THCHS-30 Speech Corpus [16], including 750 sentences, as the training set. The RNN used for experimentation has 4 LSTM layers. Each layer contains 640 memory cells. The trained RNN achieves the train accuracy of 82.93% and the valid accuracy of 81.83% (token accuracy).

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

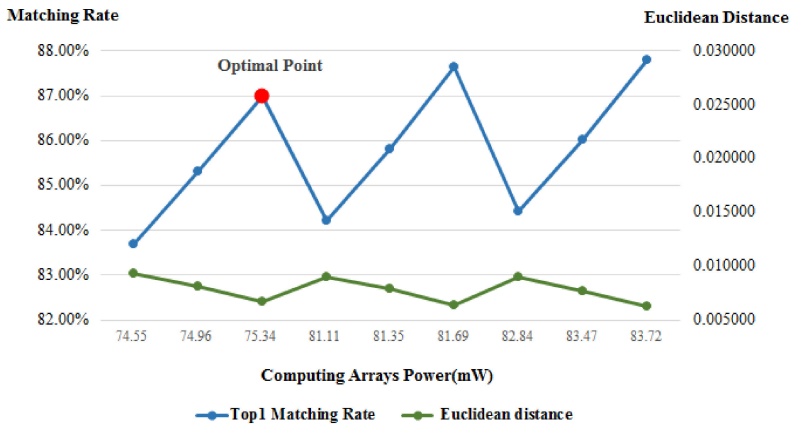

Fig. 7. Performance with different thresholds

The trained RNN is implemented using numpy library [17] as the reference network, and the RNN with proposed dynamically adaptive accuracy computing is implemented using python. The Top-1 Matching Rate for different thresholds is shown in Fig. 7. Because RNN is usually used in acoustical model, the results of the network are combined with the language model to acquire the final results. The probability distribution of all phonemes is also important, which means the total output vector should maintain a certain degree of accuracy. To be specific, the deviation is measured by the Euclidean Distance between network outputs. According to Fig. 7, some thresholds achieve lower accuracy but their power consumption are quite high. This is because that the values of $N_{ThredHi}$ are too low and the values of $N_{ThredLo}$ are too high. The improper thresholds result in an excessive number of 0 iterations and 2 iterations. Too much 0-iteration computing leads to plenty of inference failures, and the extra 2-iteration computing wastes much energy. It can also be seen in the figure that although some thresholds can achieve higher accuracy and lower deviation, while the optimal point (corresponding to thresholds 0.1 and 0.8) can perform almost the same. Therefore, the thresholds are chosen as 0.1 and 0.8. Under this threshold group settings, compared with the reference network, the power consumption of proposed work is reduced by 52.3% with just 9.2% Mismatching Rate and 0.0067 deviation.

The E-ERA prototype system was described with Verilog HDL language and simulated with Synopsys Verilog Compiler Simulator (VCS) to evaluate the system performance. This design was implemented under TSMC 45 nm LP process. The area and timing results were generated by Synopsys Design Compiler (DC) using the worst case conditions, and the dynamic power was estimated by Synopsys PrimeTime-PX (PTPX). The implementation details are outlined in Table II. The area of E-ERA prototype system is about 3.64 mm^2 and the dynamic power is estimated to 82.01 ~ 109.77 mW for processing RNN speech recognition with different SNRs.

In this work, the RNN network is used for speech recognition and additional noise have been added to the original sound to identify the flexibility of proposed dynamically adaptive accuracy computing approach. The accuracy and power of the computing arrays (the two RCAs) with different SNRs are shown in Table III. Comparison between computing arrays using usual Wallace-Tree multipliers and

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

Table II. The implementation details of proposed E-ERA

Module	Energy consumption		Hardware overhead	
	Power (mW)	Percecnt (%)	Area (um^2)	Percent (%)
Array	56.03~83.80	68.33~76.18	1132433	31.11
Memory	24.17	22.02~29.13	2105924	57.85
Others	1.81	1.65~2.21	401715	11.04
Total	82.01~109.77	100	3640072	100

Table III. Accuracy and power of RCAs in different SNRs

Speech Noise (*SNR*)	Iteration Distribution (0, 1, 2 *iterations*)	Token Accuracy	Power (mW)
clean	11.99%, 79.32%, 8.69%	70.68%	56.03
10 dB	6.79%, 64.38%, 28.83%	67.79%	64.72
20 dB	0%, 22.37%, 67.63%	55.98%	83.80

Table IV. Power and accuracy comparison with usual Wallace-Tree-Multiplier based design

Speech Noise (*SNR*)	Token Accuracy		Power (mW)	
	Wallace-Tree	Proposed	Wallace-Tree	Proposed
clean	79.83%	70.68%	117.42	56.03
10 dB	74.18%	67.79%	117.42	64.72
20 dB	61.28%	55.98%	117.42	83.80

proposed RACU are shown in Table IV. The proposed architecture can reduce power consumption with acceptable loss in accuracy due to the dynamically adaptive accuracy computing approach.

Table V gives the comparisons for processing RNN on different platforms. The result shows that, CGRAs can achieve much higher power efficiency than FPGAs for accelerating RNNs; And comparing with state-of-the-art architecture EIE, E-ERA can achieve up to 1.78 times better in power efficiency by using the the dynamically adaptive accuracy computing approach.

Table V. Comparison with other accelerators

	RNNLM [18]	EIE [19]	proposed E-ERA
Architecture	FPGA	Reconfigurable Architecture	
Technology (nm)	-	45	45
Frequency (MHz)	150	800	400
Power Consumption (mW)	25000	590	82.01~109.77
Peak Throughput (GOPS)	9.6	102	25
Power Efficiency (GOPS/W)	0.384	173	227~304

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017

4 Conclusions

This paper proposed a reconfigurable architecture named E-ERA for RNNs with dynamically adaptive accuracy computing approach. To achieve high energy efficiency, the reconfigurable approximate computing unit (RACU) with scalable precision and the reconfigurable computing array (RCA) with dynamically adaptive accuracy controlling mechanism are implemented. Comparing with other RNN accelerators, E-ERA performs 1.78 times better in power efficiency than SoA architectures and can achieve 304 GOPS/W when processing RNNs for speech recognition.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61404028 and 61574033) and the National High Technology Research and Development Program of China (863 Program) (No. 2012AA012703).

DOI: 10.1587/elex.14.20170637
Received June 15, 2017
Accepted June 28, 2017
Publicized July 14, 2017
Copyedited August 10, 2017