An Energy-efficient Multi-core Restricted Boltzmann Machine Processor with On-chip Bio-plausible Learning and Reconfigurable Sparsity

Jiajun Wu[1], Xuan Huang[1], Le Yang[1], Liang Wang[1], Jipeng Wang[1], Zuozhu Liu[3], Kwen-Siong Chong[4], Shaowei Lin[2] and Chao Wang[1]
[1]Huazhong University of Science and Technology, Wuhan, China
[2]Singapore University of Technology and Design, Singapore
[3]National University of Singapore, Singapore
[4]Nanyang Technological University, Singapore
Corresponding email: chao_wang_me@hust.edu.cn

***Abstract*—This paper proposes an energy-efficient multi-core processor design of restricted Boltzmann machine (RBM) with on-chip learning and reconfigurable sparsity. Inspired by bio-plausible variational probability flow (VPF) algorithm, our design significantly reduces the on-chip learning time and associated computation/energy as compared to conventional methods. The multi-core design with reconfigurable sparse weight connections further efficiently and flexibly reduces the required computation time and energy. FPGA implementation shows that the proposed design achieves 63.14 pJ per NW (neural weight) and 9.77 GNWs/s (neural weight update per second) at 128 MHz, which outperforms the baseline design by 44.0% and 24.3%, respectively.**

Keywords—RBM, multi-core processor, on-chip learning.

I. INTRODUCTION

Recently, restricted Boltzmann machine (RBM) has gained a lot of attention as its unsupervised generative learning is very popular to emerging IoT devices with local intelligence. In [1]-[3], researchers propose parallel/multi-core RBM accelerators in FPGA/ASIC which require external SDRAM to extensively buffer weights and bias during learning. Chen *et al.* present an NoC-based spiking neural network processor that can be used to implement RBM efficiently [4]. Most prior RBM designs are based on the conventional or modified contrastive divergence (CD) algorithm that involves multiple computation- and memory- intensive Gibbs sampling (GS) phases to update the parameters in each RBM model during the on-chip learning.

Our multi-core RBM processor design employs the bio-plausible variational probability flow (VPF) learning method in our recent work [5] that takes only one-round GS phase for parameter updates to achieve higher efficiency. The proposed design utilizes novel RBM cores to calculate partial sums for a small group of neurons and each RBM core contains a local transpose memory to buffer weights, which enable parallel partial-accumulation and reconfigurable sparsity for high efficiency. Final-accumulation cores are deployed to exploit the sparsity-induced asynchronous property of the partial sum data flow to calculate the total sum for each neuron, which is more efficient than conventional tree adders. Our RBM design can achieve high energy efficiency for intelligent IoT edge devices.

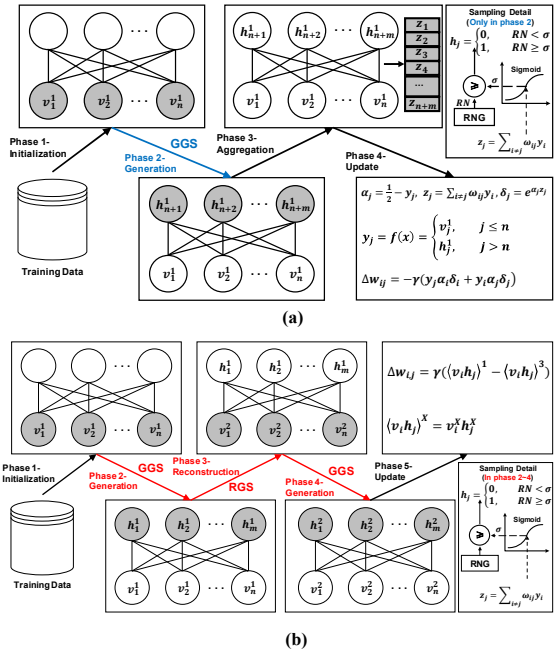


Fig. 1. Schematic diagram of RBM model with n visible neurons and m hidden neurons, and its training & inference process based on (a) VPF learning algorithm and (b) conventional CD learning algorithm, respectively. Note that the details of VPF algorithm are described in [5].

II. BIO-PLAUSIBLE LEARNING RULE

In our recent work [5], an unsupervised bio-plausible learning rule called VPF is presented for RBM model. Fig. 1 compares the learning process based on the VPF algorithm and conventional CD algorithm. As illustrated, the VPF learning algorithm requires only one generation phase and one aggregation phase (i.e., Phase 2 & 3), while the CD learning algorithm requires at least two generation phases and one reconstruction phase (i.e., Phase 2-4). Furthermore, the VPF method only involves one Gibbs sampling (GS) in the Phase 2, while the CD method requires multiple GS from the Phase 2 to

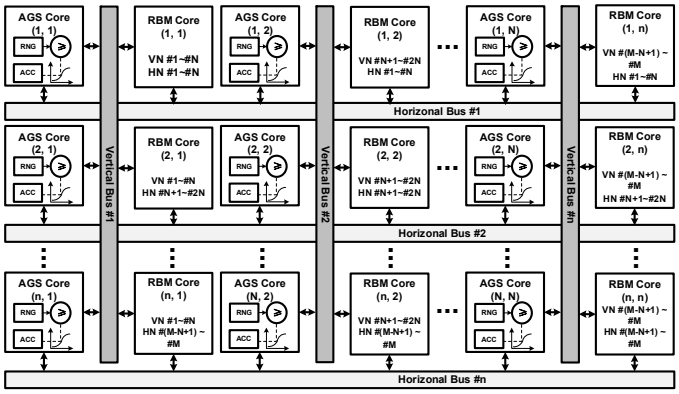


Fig. 2. Proposed multi-core architecture to illustrate how an M × M RBM model is mapped into a mesh of n × n RBM cores with AGS cores to perform the HN generation and VN reconstruction, over horizontal and vertical bus sets, respectively (n = M / N, where N is the sub-network size in each RBM core).

phase 4, as highlighted in Fig. 1 (a) and (b), respectively. As a result, the VPF algorithm significantly reduces the required computation and learning time as compared to the CD algorithm. The weight update rule of VPF learning is described by

$$\alpha_j = \frac{1}{2} - y_j, \qquad z_j = \sum_{i \neq j} w_{ij} y_i, \qquad \delta_j = e^{\alpha_j z_j} \tag{1}$$

$$\Delta w_{ij} = -\gamma\left(y_j \alpha_i \delta_i + y_i \alpha_j \delta_j\right), \tag{2}$$

where y_j (y_i) is the state of neuron j (i), α_j (α_i) is the transition direction of the neuron state, w_{ij} is the weight between the i-th visible neuron and the j-th hidden neuron, z_j is the weighted sum, δ_j is the transition rate of the neuron, γ is the learning rate, and Δw_{ij} is the weight update. The experiment on the VPF weight update rule demonstrates the Spike-Timing-Dependent Plasticity (STDP) and proves that the VPF learning is a bio-plausible method [5]. Due to its locality, the weight update rule makes the implementation of VPF learning very hardware-efficient.

III. Hardware Design and Implementation

A. Proposed Multi-core RBM Architecture

Fig. 2 shows the proposed multi-core RBM architecture with dual bus sets to perform hidden neuron (HN) generation and visible neuron (VN) reconstruction of RBM model, during generation GS (GGS) and reconstruction GS (RGS) phases, respectively. In the GGS phase, each row of RBM cores and AGS (Accumulation with Gibbs Sampling) cores are configured to interact on the horizontal bus to perform HN generation. Similarly, in the RGS phase, the mesh of RBM and AGS cores are reconfigured to perform VN reconstruction with the vertical buses. Each RBM core is for partial weighted summation and local weight update of a small N × N sub-network, while each AGS core is for the accumulation of one particular neuron's partial sums and the subsequent non-linear binarized sampling. The proposed AGS cores also enable the multi-core design to efficiently exploit the sparsity-induced asynchronous property of dataflows from multiple RBM cores, as compared to the conventional tree adder method.

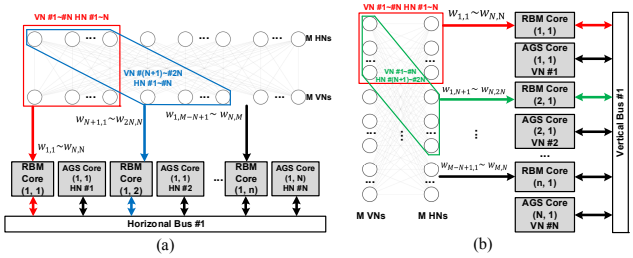


Fig. 3 Exemplary illustration of the mapping of M × M RBM computation into the proposed multi-core architecture.

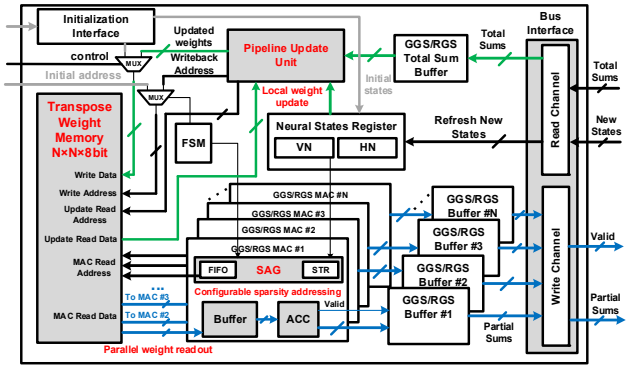


Fig. 4. Architecture and data path of proposed RBM core.

Fig. 3 illustrates the mapping strategy of an M×M RBM computation into the proposed multi-core architecture. For horizontal bus set, it is configured for forward HN generation (e.g. the Horizontal Bus #1 in Fig. 2 (a)). M HNs can be divided and assigned to n (n=M/N) rows of RBM and AGS cores, as shown in Fig. 2. The 1st row can be assigned to compute the M×N sub-network for HNs from #1~#N. The sub-network is further divided and assigned to the n RBM/AGS cores, e.g., RBM core #(1,1) is responsible for computing the N partial sums of HNs #1~#N from VNs #1~#N, while AGS core #(1,1) is in charge of computing partial-sum accumulation and subsequent sampling for HN #1, and so on so forth, as illustrated in Fig. 3 (a). The vertical bus set is configured for backward VN reconstruction (e.g., the Vertical Bus #1 in Fig. 2 (b)) and its mapping strategy is similar to the horizontal bus set. Likewise, RBM core #(1, 1) is responsible for the partial sums of VNs #1~#N from HNs #1~#N, sharing the same weights with forward HN generation, while AGS core #(1,1) is used for VN #1. With the mapping strategy, the dual bus-sets multi-core RBM architecture can achieve efficient reconfigurable computation of forward HN generation and backward VN reconstruction, with only row and column based local data movement, respectively.

B. Proposed RBM Core with Transpose Weight Memory and Reconfigurable Sparsity

Fig. 4 presents the design of proposed RBM core to perform N × N sub-network acceleration with reconfigurable sparsity. During GS phrases, the sparse address generators (SAGs) in the multiplier-accumulator (MAC) array are used to facilitate the efficient implementation of sparsity. The LFSR output, sparsity

threshold register (STR) values and binary VN/HN states are used to determine which weights will be read out from transpose weight memory (TWM). The accumulators (ACCs) calculate weighted sums for N neurons in parallel and send out final partial sums to the AGS cores asynchronously. Sampled new VN/HN states are sent back from the AGS cores for refreshing. Fig. 5 (a) presents the custom TWM design, which enables serial writing in initialization, parallel row- (generation) and column-based (reconstruction) read for summation on the MAC arrays, and weight updating. Fig. 5 (b) shows the storage unit of TWM, in which an 8-bit local weight is stored in flip-flops and bus reuse for R/W operations is enabled by transmission gates. Fig. 5 (c) shows the SAG design for generating sparse addresses. When the LFSR output is larger than STR value and the VN state is 1, the corresponding read address will be generated and stored in the FIFO before sending to the TWM.

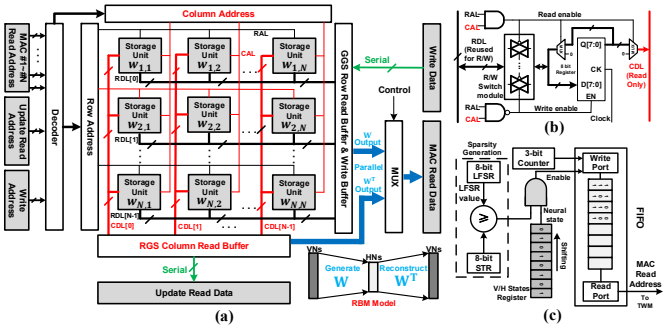


Fig. 5. (a) Transpose weights memory (TWM), (b) Storage unit of TWM and (c) Sparse address generator (SAG). In the TWM, data lines and address lines are divided into Row Data Line (RDL), Column Data Line (CDL) and Row Address Line (CAL) and Column Address Line (CAL).

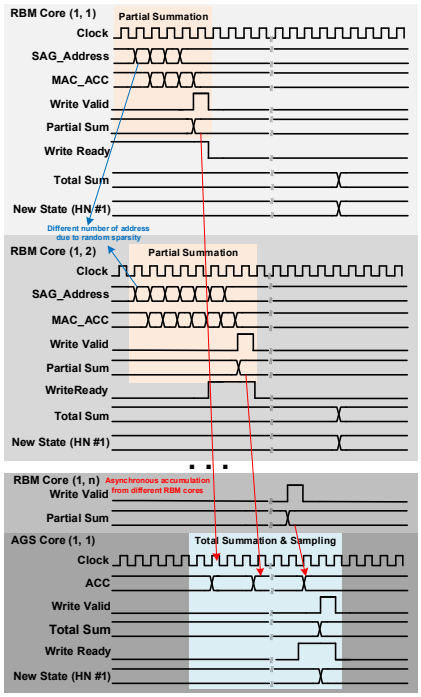


Fig. 6. Waveform diagram to illustrate the asynchronous data streams of partial sums from RBM cores to an AGS core to perform total summation and afterward sampling for a hidden neuron in generation phase.

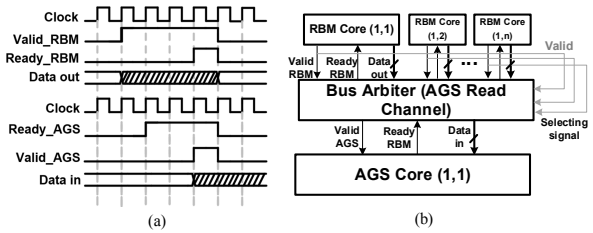


Fig. 7. (a) Timing diagram of handshake protocol and (b) schematic of bus arbiter.

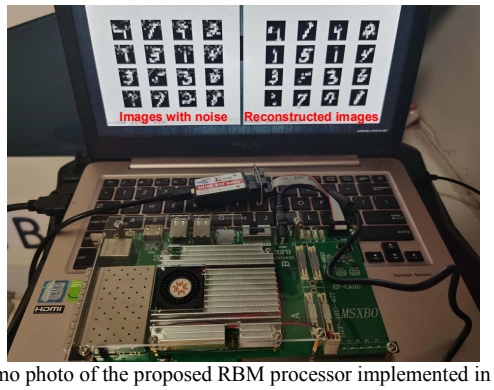


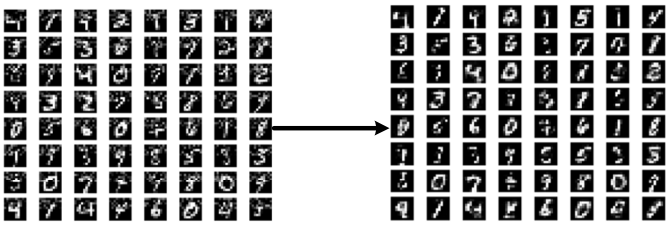
Fig. 8. Demo photo of the proposed RBM processor implemented in FPGA.

C. *Data Stream of the Proposed RBM Architecture*

Fig. 6 presents the waveform of asynchronous data streams between a row of RBM cores and an AGS core. Due to the randomness-generated sparsity in SAG blocks, different RBM cores (e.g. RBM core #(1, 1), #(1, 2), …, #(1, n) in Fig. 6) finish partial-sum accumulation for hidden neuron #1 asynchronously in time. Therefore, asynchronous transfer of partial sum and final accumulation in AGS cores is proposed in our architecture. Partial sums from those RBM cores with fewer SAG addresses (generated by LSFRs randomly) are ready earlier for an AGS core for perform final accumulation, which is exploited by the proposed asynchronous data transfer for final accumulation in the AGS core to achieve higher hardware utilization and better energy efficiency. To avoid bus contention among two or more RBM cores that happen to transfer partial sums in the same clock cycle, we implement a pre-defined priority protocol based on bus arbiter, as illustrated in Fig. 7. The arbiter decides the paring and connection of ports between the AGS and RBM cores according to the valid signals from different RBM cores.

IV. Measurement Results

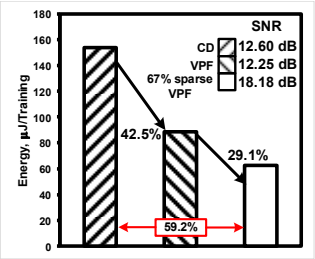
The proposed RBM processor with 10×10 cores for a 100×100 RBM model is implemented on Xilinx XC7Z100 FPGA platform to perform image denoising tasks on MNIST dataset. Due to the limited hardware resources, the handwritten images are subsampled to 10×10 pixels. Fig. 8 presents the FPGA demo of the processor. The initial weights, binarized MNIST data and control commands are sent from PC to the FPGA where the multi-core RBM processor is implemented. After the RBM processor completes the RBM computation, the calculated VN states are transferred back to the PC via UART. Software in PC renders the data to images and displays the evaluation results.



(a)

RBM Hardware Scale	100 VNs×100 HNs	
Original SNR	10.85 dB	
Reconstruction SNR	Sparsity 1	12.25 dB
	Sparsity 0.67	18.18 dB
	Sparsity 0.5	12.70 dB
Energy, µJ/Training	Sparsity 1	88.49
	Sparsity 0.67	61.27
	Sparsity 0.5	46.15

(b)



(c)

Fig. 9. Performance evaluation of the proposed RBM processor: (a) Reconstruction of subsampled MNIST images originally with random noise, (b) design summary including comparison of SNR and energy consumption, and (c) energy saving contributed by the VPF algorithm and sparse connection.

Table I. Comparison with State-of-the-art RBM Processor Designs

	C. Tsai [3] A-SSCC 2016	G. K. Chen [4] JSSC 2019	This Research	
Platform	65-nm CMOS ASIC	10-nm FinFET ASIC	28-nm XC7Z100 FPGA	
Learning Rule	CD	Spiking CD	Baseline CD	Proposed VPF
Data Dimension	4096 neurons per layer	1024 neurons per layer	100 neurons per layer	100 neurons per layer
LUTs	-	-	175,027	184,737
FF	-	-	169,518	194,987
Weight Memory	DDR3 SDRAM	6T SRAM	FF-based Transpose-RAM	FF-based Transpose-RAM
Weight Bit-width	16 bits	7 bits	8 bits	8 bits
On-chip SRAM	128 KB	1MB	10KB	10KB
Clock frequency	210Hz @1.2V	506MHz @0.9V	128MHz @1.8V	128MHz @1.8V
SNR	-	-	#12.60 dB	#12.25 dB
Learning Energy efficiency	*41.31pJ/NW	16.8pJ/NW	112.76pJ/NW	63.14pJ/NW
Inference Energy Efficiency	26.74pJ/NW	8.3pJ/NW	27.43pJ/NW	27.43pJ/NW
Learning Throughput	*7.53GNWs/s	10.3GNWs/s	7.86GNWs/s	9.77GNWs/s
Inference Throughput	26.74pJ/NW	25.2GNWs/s	25.12GNWs/s	25.12GNWs/s

Note: The data of this work is based on fully-connected models, i.e., without sparsity; *It does not include power consumption of external memory access. #The Adam optimizer in our previous work [5] isn't implemented in the RBM processor due to the limited hardware resources in FPGA. Therefore, the denoising performance of VPF is slightly worse than CD.

Fig. 9 (a) shows the proposed RBM processor can perform good denoising on corrupted images. To directly reflect the denoising performance of RBM processor, Signal Noise Ratio (SNR) is used for evaluation. Original images are added random noise and fed into well-trained RBM processor, then the SNR of original and reconstructed images are compared. The average SNR of reconstructed images and energy efficiency are summarized in Fig. 10 (b). It can be seen that with an appropriate sparsity, i.e., 50% in this case, the proposed RBM processor can achieve optimal results in terms of reconstruction SNR and energy consumption, which is 30.8% energy reduction with an increase of 5.93 dB in SNR against the design without a sparsity. When performing image reconstruction without added noises, the proposed RBM processor with configured sparsity of 67% can achieve 42.5% and 29.1% energy efficient better than the baseline CD-based design and VPF-based fully-connected design, respectively, as shown in Fig. 10 (c).

Table I compares the proposed multi-core RBM design against the state-of-the-art designs. As compared to the 65-nm ASIC design in [3], our FPGA-based RBM design achieves comparable energy efficiency and higher throughput during both on-chip learning and inference modes. The proposed RBM design also achieves 44.0% energy-efficient better and 24.3% faster than the baseline CD-based design. Our RBM design reduces the energy per image processing to 46.15 µJ with a SNR of 12.25 dB. The proposed RBM design with bio-plausible VPF learning and reconfigurable sparsity has shown great potential to achieve high energy efficiency, and therefore it will be suitable for intelligent IoT edge computing applications. The demo link of the proposed RBM processor is https://vimeo.com/431228758 (No password required).

V. Conclusion

In this paper, a multi-core RBM processor design with on-chip learning and reconfigurable sparsity is proposed to reduce energy consumption and improve processing throughput. The FPGA implementation results show that the proposed RBM design achieves 44.0% energy saving and 24.3% speed improvement in RBM training operation against the baseline CD-based RBM design. In the future, we will focus on ASIC implementation of our proposed RBM processor to further improve the energy efficiency and minimize the hardware cost.

Acknowledgment

This work is partially supported by National Natural Science Foundation of China (61974053), National Research Foundation Singapore under its AI Singapore Program (RP-2018-002) and Fundamental Research Funds of the Central Universities of China (2019KFYXJJS049).

References

[1] J. Su *et al.*, "Increasing network size and training throughput of FPGA restricted Boltzmann machines using dropout," *Proc. IEEE 24th Annual Intl. Symp. on FCCM*, 2016.

[2] K. Ueyoshi *et al.*, "FPGA implementation of a scalable and highly parallel architecture for restricted Boltzmann machines," *Circuits and Systems*, vol. 7, no. 9, 2016.

[3] C. Tsai *et al.*, "A 41.3pJ/26.7pJ per neuron weight RBM processor for on-chip learning/inference applications," *Proc. IEEE A-SSCC*, pp. 265-268, Nov. 2016.

[4] G. K. Chen *et al.*, "A 4096-neuron 1M-synapse 3.8-pJ/SOP spiking neural network with on-chip STDP learning and sparse weights in 10-nm FinFET CMOS," *IEEE JSSCC,* vol. 54, no. 4, pp. 992-1002, Apr. 2019.

[5] Z. Liu *et al.*, "Variational probability flow for biologically plausible training of deep neural networks," *Proc. AAAI 2018*.