Power Optimization Through Peripheral Circuit Reusing Integrated with Loop Tiling for RRAM Crossbar-based CNN

Yuanhui Ni*, Weiwen Chen*, Wenjuan Cui†, Yuanchun Zhou†, Keni Qiu*§
*Capital Normal University, Beijing, China
†Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
§Corresponding author: Keni Qiu (qiukn@cnu.edu.cn)

***Abstract*—Convolutional neural networks (CNNs) have been proposed to be widely adopted to make predictions on a large amount of data in modern embedded systems. Prior studies have shown that convolutional computations which consist of numbers of multiply and accumulate (MAC) operations, serve as the most computationally expensive portion in CNN. Compared to the manner of executing MAC operations in GPU and FPGA, CNN implementation in the RRAM crossbar-based computing system (RCS) demonstrates the outstanding advantages of high performance and low power. However, the current design is energy-unbalanced among the three parts of RRAM crossbar computation, peripheral circuits and memory accesses, the latter two factors can significantly limit the potential gains of RCS.**

Addressing the problem of high power overhead of peripheral circuits in RCS, this paper adopts the Peripheral Circuit Unit (PeriCU)-Reuse scheme to meet a certain power budget. The underlying idea is to put the expensive AD/DAs onto spotlight and arrange multiple convolution layers to be sequentially served by the same PeriCU. Furthermore, it is observed that memory accesses can be bypassed if two adjacent layers are assigned in the different PeriCU. Then a loop tiling technique is proposed to further improve the energy and throughput of RCS. The experiments of two convolutional applications validate that the PeriCU-Reuse scheme integrated with the loop tiling techniques can efficiently meet power requirement, and further reduce energy consumption by 61.7%.

I. INTRODUCTION

CNN-based methods are characteristics of computational intensive and resource-consuming, and thus are challenging to be integrated into embedded systems. The prior studies show that convolutional computations which consist of numbers of multiply and accumulate (MAC) operations, serve as the most computationally expensive portion of computations in CNN. Therefore, how to speed up MAC operations is a key issue of CNN acceleration. The recent research has proposed that the emerging metal oxide resistive switching random-access memory (RRAM) crossbar is able to do MAC operations through its natural analog computing in-situ in the cells where data are stored with extremely low energy [1] [2]. In this paper, the proposed RCS offers a solution to meet power budget of a system and further optimize the energy.

The RRAM crossbar network of a RCS accomplish MAC in analog, and thus analog-to-digital and digital-to-analog converters (AD/DAs) are required in the mixed-signal system to bridge the digital part and the RRAM crossbar-based analog data processing unit. However, compared with the high density and efficiency of RRAM crossbar, AD/DAs not only take up most of the chip area, but also consume much more power than RRAM devices and other analog peripheral circuits. Consequently, the overhead of conversions highly limits the potential efficiency gains of RCS. Unfortunately, the current design is energy-unbalanced among computation part, peripheral circuits and memory accesses. It is observed that the peripheral circuits cost the most significant portion of power consumption for the computation part of RCS. The memory access also costs a big portion for the whole RCS consisting of both computations and memory accesses. Therefore, the potential efficiency gains of RCS are significantly limited by the peripheral circuits and memory accesses.

Addressing the problem of high power overhead of Peripheral Circuit Unit (PeriCU) and memory accesses in RCS, this paper proposes a PeriCU-Reuse scheme to meet power budget. Different from the conventional designs of arranging peripheral circuits to serve the key computation unit, the underlying idea of this study is to put the expensive peripheral circuits onto spotlight and arrange multiple convolution layers to be alternatively served by the same PeriCU. In the PeriCU-Reuse solution, the first step is to determine a hybrid structure which is organized by multiple PeriCUs. In the second step, a loop tiling technique considering layer dependencies is conducted to further improve the energy efficiency as well as the throughput because it can save memory accesses by a smart layer schedule. The experiments validate that the PeriCU-Reuse scheme can efficiently meet the power budget and the tiling technique can further optimize energy consumption.

In summary, this work makes efforts to offer a solution of efficient convolutional computing in a novel hybrid structure integrated with loop tiling technique. We make the following main contributions.

- Given a power budget, a hybrid structure is proposed to organize convolutional layers to operate with multiple PeriCUs. The layers are computed in parallel inter-PeriCUs while alternatively intra-PeriCU.
- An energy-driven loop tiling technique considering layer dependencies is conducted to further improve the energy efficiency through smartly saving memory accesses. The safe layer starting time is discussed.

The rest of the paper is organized as follows. Section II introduces the background information of RRAM crossbar-based CNN and loop tiling technique. In Section III, the problem and motivation are presented. In Section IV, the hybrid structure integrated with loop tiling is presented. The experimental results are discussed in Section V. Finally, Sec-

978-3-9819263-0-9/DATE18/©2018 EDAA

tion VI concludes the paper.

II. Background and Related Work

A. RRAM crossbar-based CNN

The RRAM device is exploited to build cross-point structure, known as *RRAM crossbar array*. To implement CNNs on RRAM crossbar, the weights matrix is represented by conductance of RRAM cells and vector by the input voltage values [3] [4]. In this way, RRAM crossbar network can accomplish matrix-vector multiplication or vector-vector inner product intrinsically [1] [2].

The overhead of the current RCS design exhibits the following features.

- The power ratio of ADs/DAs to computation part of RCS is larger than 85% [5], presenting a very high overhead on peripheral circuits.
- The power ratio of memory accesses to the entire RCS is 37% [5], also presenting a big overhead.

Therefore, an energy efficient method to solve the high overhead of ADs/DAs and memory accesses is highly demanded.

B. Loop tiling

In recent work, loop tiling is used for the novel purpose of tailoring the CNN layers into smaller tiles on FPGA hardware platform to improve the efficiency of data reuse. The work of [6] employ loop tiling to reorder computations and memory accesses, increasing throughput and reducing data transfer of CNN. And the paper [7] demonstrated that a memory-centric design method to maximize data reuse for a buffer size restriction by loop tiling on the nested-loop description of a CNN layer. Different from the above papers, in this paper, a loop tiling technique considering layer dependencies is proposed to improve the energy efficiency as well as the throughput because it can save memory accesses by a smart layer schedule among different PeriCU of our proposed hybrid structure.

III. Problem and Motivation

CNN implementation in the RRAM crossbar-based computing system (RCS) demonstrates the advantages of high performance and low power [8] [9]. However, the peripheral circuits and memory accesses can significantly limit the potential gains of RCS.

Addressing the problem of high power overhead of peripheral circuits in RCS, this work is motivated to adopt a Peripheral Circuit Unit (PeriCU)-Reuse scheme to meet some given power budget. The underlying idea is to put the expensive ADs/DAs onto spotlight and arrange multiple convolution layers to be alternatively served by the same PeriCU. Based on this idea, a hybrid structure [10] is established. Furthermore, a loop tiling technique [7] [6] is proposed to further reduce memory accesses of RCS. With tiling the loop implementation inside feature maps [11], we can arrange two adjacent convolutional layers of one work item operate in different PeriCUs so that memory accesses can be smartly saved. Putting the hybrid structure and the loop tiling technique together, an energy efficient RRAM crossbar-based CNN design can be achieved to meet given power budget.

IV. The Hybrid Structure Integrated with Loop Tiling

In this section, we present a hybrid structure of reusing peripheral circuits and a further loop tiling technique to optimize memory accesses.

A. Peripheral circuit reusing and hybrid structure

Fig. 1(a) depicts a novel idea of design with which the MAC computation of two different layers share the same peripheral circuits, and they are implemented alternately. In other words, multiple layers' crossbar are served by one set of peripheral circuits. In this way, the power consumption can be significantly saved. In this structure, the shared Peripheral Circuit Unit (PeriCU) should provide sufficient resource to support the largest scales of kernels and feature maps for the layers that share the same PeriCU.

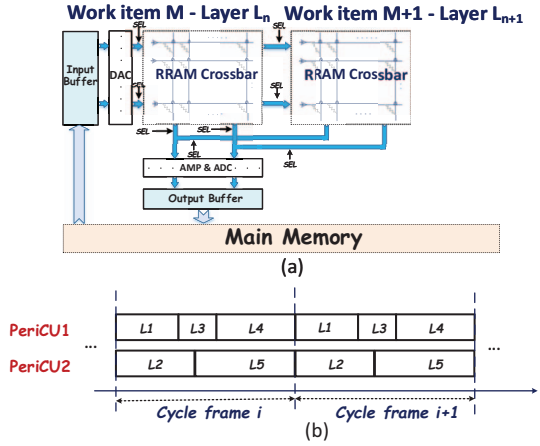

Fig. 1. (a) Design with shared PeriCU for two layers; (b) the hybrid structure.

Fig. 1(b) shows a hybrid structure of operating five convolutional layers to meet a $1/2$ power budget. There are two PeriCUs to serve the five layers. That is, *L1, L3* and *L4* are served by PeriCU1 and *L2* and *L5* are served by PeriCU2. In this way, layers belonging to the same PeriCU can be only executed in sequence while layers belonging to different PeriCUs can be executed in parallel. It exhibits a segmenting execution manner for the hybrid structure.

B. Memory access bypassing and layer scheduling

In the hybrid structure, it can be observed that if two adjacent convolutional layers in the same or different PeriCUs are from the same work item, these two convolutional layers can exchange data between input and output buffers directly and thus bypassing the expensive memory accesses. In order to obtain a highly efficient memory access inside a PeriCU, we can schedule the layers to save memory access between two adjacent layers of one work item. It is supposed scheduling layers in same PeriCUs has been done as a baseline scheme in

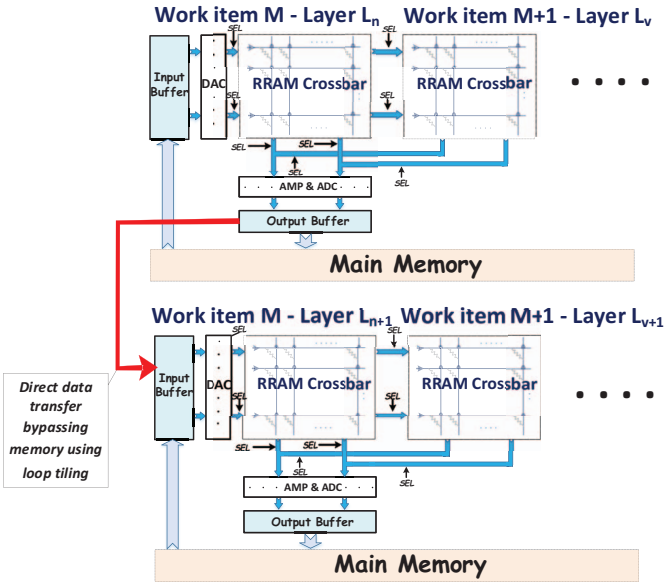

Fig. 2. A PeriCU-Reuse design with data transfer bypassing memory.

the hybrid structure. And this work mainly focuses on scheduling layers in different PeriCUs using loop tiling technique to further save memory access.

It is assumed that *Layer* L_n and *Layer* L_{n+1} are two adjacent layers of work item *M* in Fig. 2. Similarly, *Layer* L_v and *Layer* L_{v+1} are assumed to be two adjacent layers of work item *M+1*. Then the output of *Layer* L_n can be directly fed to *Layer* L_{n+1} such that data transferring from *Layer* L_n-*output* buffer to memory and from memory to *Layer* L_{n+1}-*input* buffer can be bypassed. There are the same with *Layer* L_v and *Layer* L_{v+1} to the course and the result. And the interleaved operation of *Layer* L_n and *Layer* L_{n+1}, *Layer* L_v and *Layer* L_{v+1} can be performed in a finer granularity. That is, instead of performing *Layer* L_n/L_v first and then *Layer* L_{n+1}/L_{v+1}, in the fine-grained mode, *Layer* L_{n+1}/L_{v+1} can start immediately when the output data of *Layer* L_n/L_v are ready for the first operation in *Layer* L_{n+1}/L_{v+1}. In this way, the resource requirements of input and output buffers can be greatly reduced.

In summary, layer scheduling in the hybrid structure can make the opportunities of bypassing memory accesses and also mitigating input and output buffer space requirements.

C. Energy-driven loop tiling

Due to the intrinsic layer dependence of CNNs, each point in a hidden layer of the network only depends on a well-defined region of the previous layer [12]. Constrained to this rule, a loop tiling technique is proposed based on the hybrid structure to achieve a further optimized scheduled cycle frame. Loop tiling can enable two adjacent layers of one work item while belonging to different PeriCUs to execute in parallel, thus saving memory accesses between them [13].

Fig. 3 shows the tiling size derivation with a assumption that the two adjacent layers (*Layer* L_n and *Layer* L_{n+1}) of the same work item are assigned to different PeriCUs. The inputs of *Layer* L_n comprise N different feature maps with size of $(C - S_1 + K_1) \times (R - S_1 + K_1)$. The kernel weights of *Layer* L_n and *Layer* L_{n+1} are $K_1 \times K_1 \times N$ and $K_2 \times K_2 \times M$ respectively. The corresponding strides are S_1 and S_2 respectively. As shown in Fig. 3, we apply tiling to the convolutional layer *Layer* L_n with tile size of $T_H \times T_L \times N$. *Layer* L_n then convolves all *M* of its kernels ($K_1 \times K_1 \times N$) across this tile ($T_H \times T_L \times N$), producing the $T_R \times T_C \times M$ region illustrated with red outline in feature maps of *Layer* L_{n+1}.

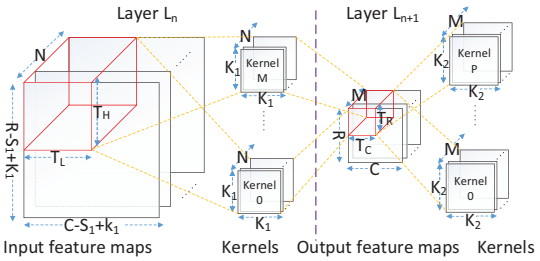

Fig. 3. Tiling size analysis.

To ensure that *Layer* L_{n+1} can be started, the produced output tile $< T_R, T_C, M >$ must be larger than or equal to the kernel size $< K_2, K_2, M >$. And T_R, T_C should meet the constraints of $T_R \geq K_2$ and $T_C \geq K_2$. In this case, we start from the data in the output tile $< T_R, T_C, M >$ and trace backwards to find the tile region of the input feature maps $< T_H, T_L, N >$ that it depends on. To achieve a minimal T_R and T_C, the depended input feature map tiles' minimal size can be obtained: $T_H = K_1 + S_1(K_2 - 1)$, $T_L = K_1 + S_1(K_2 - 1)$. If the output map tile T_R, T_C are both equal to the minimum K_2, it means that the input feature map produces K_2 times output map due to the intrinsic layer dependence. For the input feature map, each kernel is applied on the input maps with a shifting stride of *S*, which generates output map. As a result, the first input map data convolved by weights is K_1 while the other input depended data are $S_1(K_2 - 1)$. Therefore, the total minimal depended input map tile are: $T_H = K_1 + S_1(K_2 - 1)$, $T_L = K_1 + S_1(K_2 - 1)$. It is already known that the tiling factors should not be larger than the bounds of the feature map. Therefore, the tile solution for input *Layer* L_n can be illustrated by Equation 1.

$$\begin{cases} K_1 + S_1(K_2 - 1) \leq T_H \leq C - S_1 + K_1 \\ K_1 + S_1(K_2 - 1) \leq T_L \leq R - S_1 + K_1 \end{cases} \quad (1)$$

To guarantee correct data dependence between layers, the safe starting time of a layer should be considered if its previous layer is tiled in a different PeriCU. The safe starting time should guarantee that the previous layer need to generate enough output data which are fed to the next layer to forward convolutional computations as soon as possible in terms of one work item.

V. Results on Energy Efficiency

Detailed parameters of circuits and RRAM crossbar are summarized in Table I [5]. An off-chip DDR3 DRAM with access energy EDRAM of 70pJ/bit and bandwidth of

12.8GB/s [14] are adopted in this work. The simulation results of the computation amount of each kernel are achieved with PSPICE. The results of the complete convolution computations are derived from MatLab. We apply our design to two practical CNN algorithms where the convolutional layers and the concerned kernels are listed in Table II.

TABLE I
EXPERIMENTAL SETUP [5] [14].

Technology Node	65nm
RRAM Resistance Range	500Ω - 500kΩ
R_s	2kΩ
AMP	4.8mW
ADC	8bit, 35mW
DAC	8bit, 40mW
Frequency	800MHz
E_{DRAM}	70nJ/bit
BW_{DRAM}	12.8GB/s

TABLE II
TWO PRACTICAL CNN ALGORITHMS.

Workloads	Layers	Kernels	Layer Size
PV	Input		1@50×50
	C1	8@6×6×1	8@45×45
	C3	12@3×3×8	12@20×20
	C5	16@3×3×12	16@8×8
	C6	10@3×3×16	10@6×6
	C7	6@3×3×10	6@4×4
AlexNet	Input		3@224×224
	C1	48@11×11×3	48@55×55
	C3	128@5×5×48	128@27×27
	C5	192@3×3×128	192@13×13
	C6	192@3×3×192	192@13×13
	C7	128@3×3×192	128@13×13

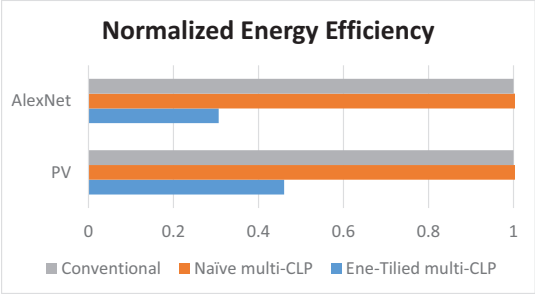

Fig. 4. Energy efficiency evaluation.

Three versions of results are evaluated: the traditional one-to-one mapping design (referred as *Conventional version*), the PeriCU-Reuse design with naive layer assignment (referred as *Naive PeriCU-Reuse version*) and the PeriCU-Reuse designs with loop tiling optimization (referred as *Ene-tiled PeriCU-Reuse version*). Fig. 4 depicts the energy consumptions with the three evaluation versions. On average, the loop tiling technique can bring 61.7% reduction on energy consumption compared to the conventional scheme. It shows that the PeriCU reusing integrated with loop tiling scheme offers a feasible design to meet power requirements while optimizing energy efficiency of RCS.

VI. CONCLUSION

Addressing the problem of high overhead on peripheral circuits in RCS, this paper proposes a PeriCU-Reuse scheme to achieve low power by sequentially reusing the expensive peripheral circuits such as AD/DAs. A hybrid structure organized by the concept of cycle frame is proposed to arrange multiple convolutional layers on PeriCUs under power budget. A loop tiling technique considering layer dependencies is proposed to further reduce energy. The experimental results show that the PeriCU-Reuse scheme offers a feasible design to meet power requirements while optimizing energy efficiency of RCS.

ACKNOWLEDGMENT

This work is supported by the grants from Beijing Advanced Innovation Center for Imaging Technology, Beijing Innovation Center for Future Chip, National Natural Science Foundation of China [Project No. 61502321] and the Project of Beijing Municipal Education Commission [Project No. KM201710028016].

REFERENCES

[1] B. Li, P. Gu, Y. Shan, Y. Wang, Y. Chen, and H. Yang, "RRAM-based analog approximate computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 12, pp. 1905–1917, 2015.

[2] Y. Wang, H. Li, and X. Li, "Re-architecting the on-chip memory sub-system of machine-learning accelerator for embedded devices," in *International Conference on Computer-Aided Design (ICCAD'16)*, 2016, pp. 1–6.

[3] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *International Symposium on Computer Architecture (ISCA'16)*, 2016, pp. 27–39.

[4] T. Tang, L. Xia, B. Li, R. Luo, Y. Chen, Y. Wang, and H. Yang, "Spiking neural network with RRAM: Can we use it for Real-World application?" in *Design, Automation & Test in Europe Conference (DATE'15)*, 2015, pp. 860–865.

[5] L. Xia, T. Tang, W. Huangfu, M. Cheng, X. Yin, B. Li, Y. Wang, and H. Yang, "Switched by input: Power efficient structure for RRAM-based convolutional neural network," in *Design Automation Conference (DAC'16)*, 2016, pp. 1–6.

[6] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'15)*, 2015, pp. 161–170.

[7] M. Peemen, A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in *International Conference on Computer Design (ICCD'13)*, 2013, pp. 13–19.

[8] B. Li, X. Xia, P. Gu, Y. Wang, and H. Yang, "Merging the interface: Power, area and accuracy co-optimization for RRAM crossbar-based mixed-signal computing system," in *Design Automation Conference(DAC'15)*, 2015, pp. 13:1–13:6.

[9] H. Li, Y. Chen, C. Liu, J. P. Strachan, and N. Davila, "Looking ahead for resistive memory technology: A broad perspective on ReRAM technology for future storage and computing," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 94–103, 2017.

[10] Y. Shen, M. Ferdman, and P. Milder, "Maximizing CNN accelerator efficiency through resource partitioning," in *International Symposium on Computer Architecture (ISCA '17)*, 2017, pp. 535–547.

[11] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Architectural Support for Programming Languages and Operating Systems (ASPLOS'14)*, 2014, pp. 269–284.

[12] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators," in *IEEE/ACM International Symposium on Microarchitecture (MICRO'16)*, 2016, pp. 1–12.

[13] P. Panda, A. Sengupta, S. S. Sarwar, G. Srinivasan, S. Venkataramani, A. Raghunathan, and K. Roy, "Cross-layer approximations for neuromorphic computing: From devices to circuits and systems," in *Design Automation Conference (DAC'16)*, 2016, pp. 1–6.

[14] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," vol. 25, no. 8, pp. 2220 – 2233, 2017.