Mixed Size Crossbar based RRAM CNN Accelerator with Overlapped Mapping Method

Zhenhua Zhu*†, Jilan Lin*, Ming Cheng*†, Lixue Xia*†, Hanbo Sun*†
Xiaoming Chen‡, Yu Wang*†, Huazhong Yang*†
*Department of Electronic Engineering, Tsinghua University, Beijing, China
†Beijing National Research Center for Information Science and Technology (BNRist), Beijing, China
‡State Key Laboratory of Computer Architecture, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China

ABSTRACT

Convolutional Neural Networks (CNNs) play a vital role in machine learning. CNNs are typically both computing and memory intensive. Emerging resistive random-access memories (RRAMs) and RRAM crossbars have demonstrated great potentials in boosting the performance and energy efficiency of CNNs. Compared with small crossbars, large crossbars show better energy efficiency with less interface overhead. However, conventional workload mapping methods for small crossbars cannot make full use of the computation ability of large crossbars. In this paper, we propose an Overlapped Mapping Method (OMM) and MIxed Size Crossbar based RRAM CNN Accelerator (MISCA) to solve this problem. MISCA with OMM can reduce the energy consumption caused by the interface circuits, and improve the parallelism of computation by leveraging the idle RRAM cells in crossbars. The simulation results show that MISCA with OMM can achieve 2.7× speedup, 30% utilization rate improvement, and 1.2× energy efficiency improvement on average compared with fixed size crossbars based accelerator using the conventional mapping method. In comparison with GPU platform, MISCA with OMM can perform 490.4× higher on average in energy efficiency and 20× higher on average in speedup. Compared with PRIME, an existing RRAM based accelerator, MISCA has 26.4× speedup and 1.65× energy efficiency improvement.

KEYWORDS

Convolutional Neural Networks, RRAM, Computer Architecture

1 INTRODUCTION

Recently, convolutional neural networks (CNNs) have made many breakthroughs in the computer vision area. Despite the high accuracy that can be achieved by state-of-the-art CNNs, the training and inference of state-of-the-art CNNs typically consume high energy and long computation time. For example, the inference of VGG-16 for one image needs 4.3J energy consumption on GPU platforms [12] [16]. In order to improve the energy efficiency and the performance of CNNs, many accelerators based on different hardware platforms have been proposed, such as FPGA [12] and GPU [14] based accelerators. Though they offer high performance for inference and/or training, the conventional von Neumann architecture causes high energy consumption because of large data movements between processors and memory. Thus, there are urgent needs for developing new computer architectures to satisfy the power efficiency requirements of modern CNNs.

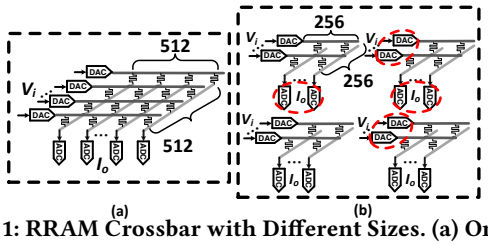

Figure 1: RRAM Crossbar with Different Sizes. (a) One 512 × 512 **RRAM crossbar (b) Four** 256 × 256 **RRAM crossbars**

Emerging resistive random-access memories (RRAMs) and RRAM based Computing Systems (RCSes) provide alternative solutions to improve the energy efficiency of inference or training of CNNs [15]. As shown in Fig. 1, RRAMs can be used to build a crossbar structure to efficiently perform matrix-vector products. Convolution operations in CNNs can also be regarded as dot products (or matrix-vector products). Therefore, recent work has demonstrated that RRAM crossbars can improve the energy efficiency by over 100× compared with CPU- or GPU-based solutions [3] [13] [15].

The current RRAM architectures use small crossbars, e.g., 128 × 128 crossbars in ISAAC [13]. When mapping a CNN to small crossbars, large convolutional (Conv) layers or fully-connected (FC) layers need to be split and mapped to multiple crossbars [3]. Therefore, more interfaces, which mainly include analog-to-digital convertors (ADCs) and digital-to-analog convertors (DACs), are brought in. However, additional interfaces significantly increase the cost because interfaces typically dominate more than 85% energy in RCSes [9]. Therefore, using large crossbars instead of small ones can reduce the interface cost. For example, as shown in Fig. 1, with the same number of RRAM cells, using one 512 × 512 crossbar can eliminate at most 50% ADCs and DACs compared with four 256×256 crossbars. Consequently, using large crossbars is the future trend for RCSes because of the reduction of energy consumption. Besides, with the development of the RRAM fabrication technology, crossbars will be larger. Recent work has demonstrated an energy-efficient RCS based on fabricated 512 × 512 crossbars [6].

However, the conventional mapping method used in [3] [15] cannot make full use of the computation ability of large crossbars. For example, according to Table 1, when mapping ResNet-18 [5] and AlexNet [8] to 512×512 crossbars, the utilization rates are both only 57%. Thus, more than 40% cells in the crossbars are "wasted".

ICCAD '18, November 5–8, 2018, San Diego, CA, USA

ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3240825

Table 1: Energy consumptions of conv layers and utilization rate of crossbars with different crossbar sizes (working at $100MHz$, using conventional mapping method[3][15])

CNN Model	AlexNet			ResNet-18		
Crossbar Size	128	256	512	128	256	512
Energy Consumption of Convolutions (mJ)	1.33	0.82	0.59	2.47	1.78	1.52
Utilization Rate	99%	79%	57%	99%	88%	57%

The low utilization rate of large crossbars is caused by two factors. First, the conventional mapping method maps one kernel to one column of a crossbar, but $60 \sim 90\%$ output channels of CNNs are less than 512, resulting in a waste of unused crossbar columns. Second, in the conventional mapping method, a CNN kernel is unfolded to a column vector, whose length is not an integral multiple of the crossbar size. So the kernels cannot fill one crossbar column, causing a waste of unused crossbar rows. For instance, we need two 512×512 crossbars to store the kernel with size of $3 \times 3 \times 64 = 576$ in ResNet-34 [5]. The number of unused rows is $512 \times 2 - 576 = 448$, which causes more than 87.5% waste of the second crossbar.

In this paper, we analyze the reasons of the low utilization rate of large crossbars when mapping CNNs, which are shown in Section 3. In order to solve this problem, we propose an Overlapped Mapping Method (OMM) in Section 4, which uses the idle cells in large crossbars to improve the computational parallelism. However, the performance of OMM is limited when using fixed size crossbars, which will be discussed in Section 5.1 in detail. Inspired by this, we present a MIxed Size Crossbar based RRAM CNN Accelerator (MISCA) in Section 5. Then we design an area-constrained crossbars allocation and mapping strategy for mapping CNNs on MISCA. The main contributions of this paper include:

(1) We propose OMM, which maps one kernel to multiple columns of a crossbar, and different portions of the kernel are overlapped for reusing the input data. We demonstrate that OMM can improve the speedup and the energy efficiency of RCSes.
(2) We present MISCA, to leverage the advantages of the OMM algorithm and break the performance limitation caused by the fixed crossbars size.
(3) We propose an area-constrained mixed size crossbar based mapping strategy for mapping an entire CNN model on MISCA. To the best of our knowledge, this is the first work to utilize mixed size crossbars to optimize the mapping of 1×1 convolutional layers.
(4) A set of CNNs are used to evaluate the performance of MISCA with OMM. Compared with using fixed size crossbars, MISCA with OMM can achieve 2.7× speedup and 1.2× energy efficiency improvement, and the utilization rate of RRAM crossbars is increased from 56.75% to 91.26% on average. Compared with GPU, MISCA can perform 490.4× higher in energy efficiency and 20× higher in speedup. Compared with PRIME[3], an existing RRAM-based accelerator, MISCA has 26.4× speedup and 1.65× energy efficiency improvement.

2 PRELIMINARIES AND RELATED WORK

2.1 Convolutional Neural Network

Typical CNNs usually consist of several convolutional (Conv) layers and fully-connected (FC) layers. The convolution operation is the main operation in Conv layers, which can be expressed as Equ. (1):

$$f_o(x,y,z) = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{k=1}^{C_{in}} f_i(x+i, y+j, k)k_z(i,j,k) \quad (1)$$

where 3-dimensional matrices f_o and f_i represent the input and output feature maps, respectively. K is the kernel size and C_{in} is the number of input channels. k_z represents the z^{th} 3-dimensional convolution kernel with size of $K \times K \times C_{in}$.

FC layers can be expressed as Equ. (2):

$$output_i = f(\sum_j (w_{i,j} \times input_j + b_i)) \quad (2)$$

where $w_{i,j}$ represents the (i,j) element of weight matrix W, $f(\cdot)$ is a nonlinear activation function, e.g. ReLU, $\max(0,x)$.

2.2 RRAM and Its Crossbar Structure

An RRAM is a passive two-terminal device and its crossbar structure can implement matrix-vector products efficiently [9], as shown in Fig. 1(a) (b). The input voltage vector $\boldsymbol{V}$ and the output current vector $\boldsymbol{I}$ have the relationship expressed as Equ. (3) [9]:

$$i_{out,k} = \sum_{j=1}^{N} g_{k,j} v_{in,j} \quad (3)$$

where $v_{in,j}$ is the j-th element of the input voltage vector $\boldsymbol{V}$, $g_{k,j}$ represents the conductance of RRAM device that in the k-th column and the j-th row and $i_{out,k}$ is the k-th element of the output current vector $\boldsymbol{I}$. Thus, we can implement Conv and FC layers on crossbars. The input data f_i or $input_j$ is represented by analog voltages on the wordlines of a crossbar, and the kernel weights k_z or $w_{i,j}$ are represented by the RRAM cell conductances in the crossbar.

RRAM is an analog device and the analog output signals are difficult to store. Thus, ADCs and DACs are used as the interfaces between crossbars and digital circuits, as Fig. 1 shows. In RCSes, [9] has demonstrated that the interfaces occupy a significant portion ($> 85\%$) of area and power consumption.

2.3 Existing RRAM based CNN Accelerators

Previous work has demonstrated several RRAM based CNN accelerators [3] [13] [15]. These architectures use fixed size crossbars in their designs, e.g. 128×128 in [13] [15] and 256×256 in [3] .

All of the current RRAM based accelerators use the conventional mapping method [15]. In this mapping method, 3-dimensional kernels with size of $K \times K \times C_{in}$ are unfolded to a $K^2 C_{in} \times 1$ column vector at first. Then the unfolded kernel is mapped to one column of the crossbar. If the length of the column vector exceeds the size of the crossbar, multiple crossbars are needed to store it and the convolution results are accumulated by adding the outputs of these crossbars. Different kernels of one Conv layer are mapped to different columns, such that the convolution results of these kernels can be computed in parallel. Assume that the crossbar's size is $S \times S$ and the number of output channels is C_{out}, the number of required crossbars can be calculated by Equ. (4):

$$N = N_x \times N_y = \left\lceil \frac{C_{out}}{S} \right\rceil \times \left\lceil \frac{K \times K \times C_{in}}{S} \right\rceil \quad (4)$$

where N_x and N_y represent the number of crossbars required in the horizontal direction (denoted as x-direction) and in the vertical direction (denoted as y-direction), respectively.

Similarly, the feature map data used for convolutions are unfolded to a column vector and loaded into the input ports of the crossbar. In the next cycle, the kernel window shifts to the right and another set of the feature map data is loaded, as shown in Fig. 2(b).

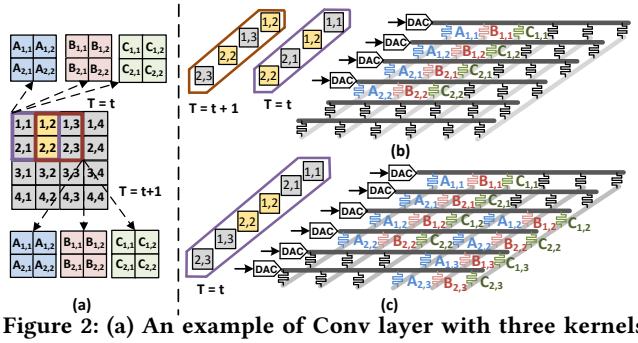

Figure 2: (a) An example of Conv layer with three kernels (kernel A, B, and C); (b) The mapping result of the conventional mapping method; (c) The mapping result of OMM.

3 MOTIVATIONAL EXAMPLE

The current RRAM based accelerators adopt small crossbars. However, when mapping CNNs to small crossbars, more interfaces are brought in because we need to split the networks.

Based on Equ. (4), we can calculate the number of DACs and ADCs required in an RCS accroding to Equ. (5):

$$N_{DAC} = \left\lceil \frac{C_{out}}{S} \right\rceil \times K \times K \times C_{in};\ N_{ADC} = \left\lceil \frac{K \times K \times C_{in}}{S} \right\rceil \times C_{out} \quad (5)$$

Equ. (5) shows that the number of ADCs and DACs increases rapidly with the decreasing of S. Previous work has demonstrated that DACs and ADCs occupy more than 85% energy consumption of an RCS [9]. Therefore, small crossbars are less energy efficient because of the peripheral circuits. As shown in Table 1, compared with 512×512 crossbars, 128×128 and 256×256 crossbars consume 1.94× and 1.29× higher energy, respectively. The data for energy estimation are referred to [1] [2] [11]. The parameters of the CNN models are determined by [5] [8]. All CNNs are mapped to crossbars using the conventional mapping method illustrated in [15].

Furthermore, FC layers have the same problem as Conv layer, which is caused by additional interfaces when using small crossbars. For example, in AlexNet[8] and VGG[7], there exist several FC layers with 4096 inputs and 4096 outputs. When utilizing 128×128 crossbars, we need 4× ADCs and DACs compared with 512×512 crossbars, which means nearly 3.4× energy consumptions.

In order to improve the energy efficiency, large crossbars, e.g., 512×512 in [6], are used in RCSes. However, the conventional mapping method does not make full use of the computation resources of RRAM crossbars. We define the utilization rate in Equ. (6). It represents the ratio of the number of computing cells to the number of cells in the entire crossbar.

$$U = \frac{K \times K \times C_{in} \times C_{out}}{\left\lceil \frac{K \times K \times C_{in}}{S} \right\rceil \times \left\lceil \frac{C_{out}}{S} \right\rceil \times S^2} \quad (6)$$

Table 1 shows that the utilization rate of a 512×512 crossbar when mapping different CNN models on it. The crossbar's utilization rates on AlexNet and ResNet-18 are both 57%. Thus, more than 40% cells are "wasted" in the conventional mapping method.

The low utilization rate of large size crossbars comes from the fact that either the number of output channels (C_{out}) or the size of a kernel ($K \times K \times C_{in}$) is not an integral multiple of the crossbar's size. For example, in Fig. 2(b), the kernel size is $2 \times 2 \times 1$ with three output channels and the size of the crossbar is 6×6. There are three idle columns and two idle rows in the conventional mapping method, which are the black RRAM cells in Fig. 2(b).

4 OVERLAPPED MAPPING METHOD

To overcome the drawbacks of the conventional mapping method, we propose *Overlapped Mapping Method (OMM)* which takes full advantage of idle cells of crossbars to improve the computational energy efficiency.

4.1 Description of OMM

Considering the sliding window in Conv layers, when the window shifts to the right, some feature map data are reused in the next convolution. Consequently, the input data used in different cycles can be merged to achieve equivalent effect of sliding window. As the example shown in Fig. 2(a) (c), the yellow block data $[(1,2),(2,2)]^T$ is reused in $T = t$ and $T = t+1$ for convolution. So we merge the input data used in $T = t, t+1$ to a long vector, i.e. the purple frame in Fig. 2(c). Thereby, the corresponding kernel vectors are staggered and overlapped, as shown in Fig. 2(c). Then, one kernel is mapped multiple times to the different columns on one crossbar.

The stride of the Conv layer is s_t and the length of overlapped part $l_{overlapped}$ can be expressed in Equ. (7):

$$l_{overlapped} = \begin{cases} (K - s_t) \times K \times C_{in} & K > s_t \\ 0 & K \le s_t \end{cases} \quad (7)$$

In the example, we can complete six convolutions in one cycle, while we need two cycles in original mapping method.

4.2 Analysis of OMM

Speedup Analysis. As mentioned in Section 4.1, OMM boosts the computation parallelism so that it improves the throughput of the convolutional layer. Under the constraints that we do not add additional crossbars, there are two factors limiting the parallelism:

The quantity of crossbars in x-direction limits the number of columns which are used to represent the same kernel. The maximum speedup derived from x-direction is shown in Equ. (8) :

$$x_{speedup} = \left\lfloor \frac{\left\lceil \frac{C_{out}}{S} \right\rceil \times S}{C_{out}} \right\rfloor \quad (8)$$

The quantity of crossbars in y-direction limits the length of the long column vector merged by multiple input vectors, which is expressed in Equ. (9):

$$y_{speedup} = \left\lfloor \frac{\left\lceil \frac{K^2 \times C_{in}}{S} \right\rceil \times S - K^2 C_{in}}{s_t \times K \times C_{in}} + 1 \right\rfloor \quad (9)$$

According to Equ. (8) and Equ. (9), the speedup (s_p) of the Conv layer is calculated as follows:

$$s_p = \min(x_{speedup}, y_{speedup}) \quad (10)$$

Energy Consumption Analysis. Depending on OMM, we reduce the times of loading feature map data into the crossbars by overlapping and reusing the input elements. Therefore, the number of DACs decreases, reducing the energy consumption. Besides, the number of Conv layer's output data stays unchanged, leading to the same energy consumption caused by ADCs as before.

According to Fig. 2(b) (c), in the conventional mapping method, two cycles are needed for getting six results and four DACs are used. In OMM, these results only need one cycle with six DACs. Therefore, OMM can reduce the energy consumption caused by DACs by 25%. In common CNNs, the overlapped part is much longer than this example because of the large input channel (C_{in}). Thus, OMM can achieve better the energy efficiency of RCSes for CNN.

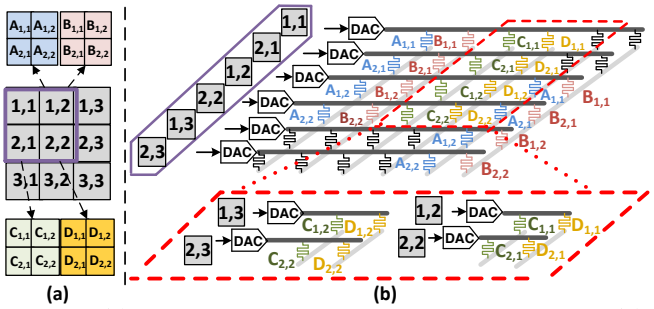

Figure 3: (a) An example of unbalanced mapping result; (b) Balance the mapping with mixed size crossbars

Utilization Rate Analysis OMM makes use of the idle columns of crossbars, which increases the utilization rate of large size crossbars, the new utilization rate can be calculated by Equ. (11):

$$U_{OMM} = \frac{[(s_p - 1) \times s_t \times K \times C_{in} + K^2 C_{in}] \times s_p \times C_{out}}{\left\lceil \frac{K \times K \times C_{in}}{S} \right\rceil \times \left\lceil \frac{C_{out}}{S} \right\rceil \times S^2} \quad (11)$$

5 MISCA

The proposed OMM can improve the energy efficiency, speedup and, utilization rate, but the performance is limited by the size of crossbars. In order to break this limitation, we design an architecture based on RRAM, MISCA. In Section 5.1, we analyze the limitation of OMM on existing RRAM based architecture. In Section 5.2, we propose MISCA and its details. Then, the detailed mapping method and crossbars allocation strategy on MISCA for Conv layers and FC layers are introduced in Section 5.3 and Section 5.4, respectively.

5.1 Limitation of OMM on Existing RRAM based Architecture

There exist three factors that limit the performance of OMM on existing RRAM based architecture:

Limited improvement of utilization rate. Even if OMM can improve the utilization rate, there still exists a waste of RRAM cells in crossbars. For example, as shown in Fig. 4, the second Conv layer of ResNet-18 contains $C_{out} = 64$ kernels, each with $3 \times 3 \times 64 = 576$ elements [5]. The blue bar represents the 64 kernels and the orange border boxes are 512×512 crossbars. According to Equ. (7), the length of overlapped part in the column direction is $2 \times 3 \times 64 = 384$. Therefore, there are $576 - 384 = 192$ elements for each kernel that cannot be overlapped. In other words, this un-overlapped part takes up $192 \times C_{out}$ cells out of 192×512 array in the crossbar. The unused cells in these 192 rows are shown in gray. Under this circumstance, if we use one 256×256 crossbar to cover the un-overlapped part, a higher utilization rate and smaller area can be obtained. Besides, the small crossbars do not increase the energy consumption without additional interfaces. As shown in Fig. 4(b), the number of ADCs, which are circled by red, remains the same.

Speedup improvement is limited by the unbalanced mapping result. When the size of crossbars is not an integral multiple of C_{out}, e.g., AlexNet contains layers with 96 and 384 output channels, we get an unbalanced mapping result, which is shown in Fig. 3. In this case, limited by the crossbar size, only a part of the kernels can be overlapped. As the example shown in Fig. 3(b), there are four kernels A, B, C, D while only the former two kernels A, B can be overlapped mapping. In the first cycle, we load $[(1,1),(2,1),(1,2),(2,2),(1,3),(2,3)]^T$, and only kernel A and B convolve with $[(1,2),(2,2),(1,3),(2,3)]^T$. Thus, we need to reload

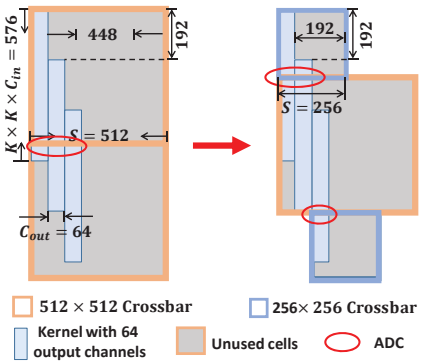

Figure 4: Improve the utilization rate with mixed size crossbars

$[(1,2),(2,2),(1,3),(2,3)]^T$ for convolving with kernel C and D in the second cycle. It increases the energy consumption. An effective solution is proposed to use additional crossbars storing C, D to "balance" the mapping, as shown in Fig. 3(b). Thus, if the additional columns are less than 128, adopting smaller crossbars can solve the unbalanced problem better compared with 512×512 crossbars because of its higher utilization rate.

No speedup gains for 1×1 convolutions and FC layers. The analysis of OMM shows that the speedup comes from overlapping the reused input data, then the computational parallelism is improved. On the one hand, FC layers do not have reused input data according to Equ. (2), so there is no speedup for FC layers from OMM. On the other hand, for 1×1 convolutions in some CNNs, e.g. ResNet, the kernel size is 1 and the stride is 1 or 2. The length of the overlapped part of input data is 0 according to equ. (7), which means they can not be accelerated by OMM. Because of the branch structure of ResNet, 1×1 convolutions may be the bottleneck of the computation time. To tackle these problems, weight matrix duplications, which use multiple crossbars to store the same weight, are adopted for higher parallelism. Therefore, how to improve the parallelism with the minimum duplication overhead is critical. Small crossbar has advantages in area and flexibility. So it is suitable for 1×1 kernels' duplications (e.g. more than 50% 1×1 Conv layers of ResNet have input channels and output channels less than 128). But FC layers have much more weight data than Conv layers. So large crossbars are better for minimum interfaces overhead.

Inspired by these factors, a generalized architecture with mixed size crossbars, MISCA, is designed to break the limitation caused by the fixed crossbars.

5.2 Architecture of MISCA

The overall architecture of MISCA is shown in Fig. 5(a). The MISCA consists of eight RRAM banks and a global buffer. All the banks and the global buffer have a shared bus for loading and storing the intermediate data between different layers. Each RRAM bank is composed of *Input Data Rearrangement Circuits (IDRC)*, three *Process Element Arrays (PEA)* with crossbars of different sizes, *Crossbar Selection Circuits (Sel)*, *SUM Circuits*, *Pooling and ReLU Circuits* and a *Controller (Ctr)*, as Fig. 5 shows.

Input data rearrangement circuits are shown in Fig. 5(b), which contains a input data block queue and a counter. As shown in Section 4.1, the feature map data is merged to a long column vector, which can be divided into $\lceil K/s_t \rceil$ blocks. Each block contains $K \times s_t \times C_{in}$ input data. At the end of each clock cycle, the first

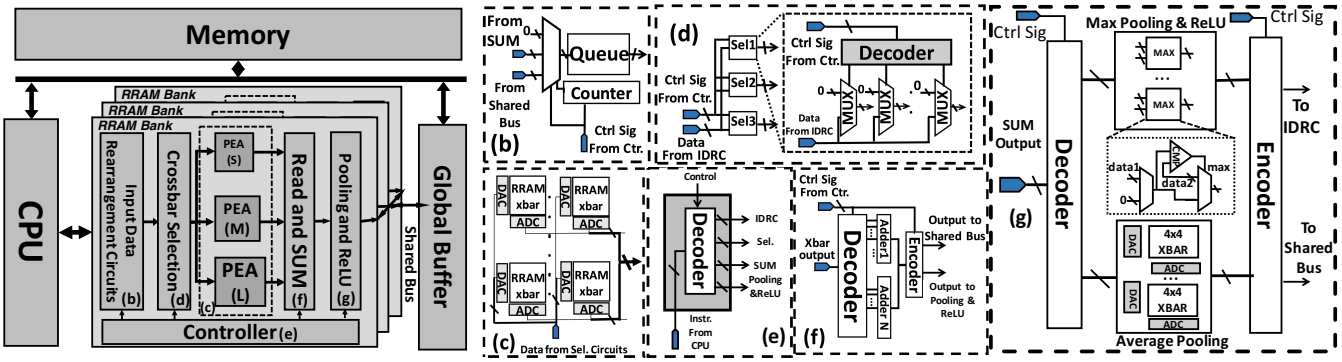

Figure 5: (a) The overview of MISCA architecture; (b) Input Data Rearrangement Circuits (IDRC); (c) Process Elements Array (PEA); (d) Crossbar Selection Circuits (Sel); (e) MISCA Controller (Ctr); (f) SUM Circuits (SUM); (g) Pooling and ReLU Circuits

block is pulled from the queue and another block is read from the global buffer or the *Pooling and ReLU Circuits*. The counter indicates whether the sliding window shifts to the end of the row. If the window reaches the end, the queue is flushed and loads a new data block as the window slides to the beginning of the next row.

Process Element Array (PEA) is shown in Fig. 5(c). There exist three *PEAs* in one bank: large, medium, and small RRAM crossbars array, which contains 512×512 crossbars, 256×256 crossbars, and 128×128 crossbars. Each *PEA* has $32 \times 2 = 64$ crossbars, and the whole system contains $64 \times 8 = 512$ large, medium, and small crossbars, which are enough for mapping a large scale CNN (e.g. ResNet-50). Input data is transfered from *Sel* to each crossbar. At the falling edge of each cycle, convolution results are read from crossbars and then sent to *SUM circuits*.

Crossbar Selection Circuits are aimed at selecting the exact number of crossbars in *PEAs* and transferring the feature map data to them. As demonstrated in Fig. 5(d), this part is composed of three submodules connecting to three *PEAs*. The submodule contains a decoder and a group of MUXs. Each MUX connects to a crossbar in the *PEA* for selection. It sends zero input to the unused crossbars.

Controller receives instructions and signals from CPU and provides signals to peripheral circuits in RRAM bank as shown in Fig. 5(e). Its functions contain setting the threshold of counter in *IDRC*, configuring the settings of *Sel*, and controlling the data flow.

SUM Circuits are shown in Fig. 5(f) for adding the outputs of different crossbars to get the convolution results, which are composed of ADD decoders, adders, and an encoder. ADD decoders receive control signal from *controller*, and determine which crossbars' output should be sent to the adders. The outputs of adders are transferred to the encoder. The encoder determines whether send the results to *Pooling and ReLU Circuits* in this bank or transfer the data to the shared bus.

Pooling and ReLU Circuits are shown in Fig.5(g). It implements the pooling and the nonlinear activation function after Conv layers. There exist three parts of this circuit: *ReLU part*, *max pooling part*, and *average pooling part*. *ReLU part* performs the nonlinear activation function y = max$(x, 0)$, which can be composed by a comparator and a transfer gate. *Max pooling part* implements the function $y = \max(x_1, x_2, ...x_N)$. In MISCA, in order to reduce the area overhead, we reuse the circuits of ReLU part to perform max pooling, and use a tree structure to reduce the operation time. *Average pooling part* computes the function $y = \text{avg}(x_1, x_2, ...x_N) = \sum_{i=1}^{N} 1/N \times x_i$. According to Equ. (3), we can use crossbars with the same weight in each cell, i.e. $1/N$, to perform average operations. In MISCA, we use 64 4×4 crossbars to compute the average function.

5.3 Mapping Convolutional Layers on MISCA

In this section, we propose the crossbars allocation and mapping strategy for mapping Conv layers on MISCA. As demonstrated in Section 5.1, OMM cannot accelerate the 1×1 convolutions. So we discuss the detailed mapping strategy for convolutions with kernel size larger than 1 (we call them regular convolutions) and 1×1 convolutions separately in this section. Then we show the area-constrained mapping strategy for the entire Conv layers, which aims to achieve the highest speedup with little extra area overhead, compared with fixed size crossbars based accelerator.

Mixed size crossbar based mapping strategy for regular convolutions. In OMM, different convolutions need different number of crossbars to map. We introduce a square coverage model to describe the mapping problem. CNN kernels are abstracted into a rectangle with the size of $[length, width] = [K \times K \times C_{in}, C_{out}]$. Given a specific speedup s_p, the rectangle is duplicated for s_p copies. Then we arrange these copies in line, but the later one is staggered with the former one. An example is shown in Fig. 4, the blue bars are the Conv kernels with $s_p = 3$. After that, the mapping process has a problem that uses three kinds of squares with the size of 128, 256, and 512, to cover these rectangles. Considering the trade-off between energy consumption and utilization rate of crossbars of different sizes, we use the greedy method to improve utilization rate for reducing the energy consumption. The coverage strategy is composed of four steps:

(1) Use a grid to cover the duplicated rectangles. The grid consist of $I \times J$ squares with the size of 128×128, (I, J) are calculated from Equ. (12).

$$I = \left\lceil \frac{(s_p - 1) \cdot s_t \cdot K \cdot C_{in} + K^2 \cdot C_{in}}{512} \right\rceil \times 4;\ J = \left\lceil \frac{s_p \cdot C_{out}}{512} \right\rceil \times 4 \quad (12)$$

Where s_t is the stride of the Conv layer. If the (i, j) (the ith row, the jth column) square has covered a part of the kernels, then we mark the (i, j) square with 1.

(2) Find a set of 16 (4×4) grid squares, which meet the following conditions: *a)* Any square in the set does not be marked as 3; *b)* The sum of the marked number of these squares is larger than eight, i.e. half of the squares have covered the proportion of kernels, and the sum of marked number of other sets. The number "8" means that the utilization rate is nearly or more than 50% when using large crossbars, or we need more than three middle crossbars. Compared with

one large crossbars, three middle crossbars have nearly 50% extra energy consumption overhead. Then we mark these squares with 3 and repeat this step until no more sets meet these conditions.

(3) Similarly, find a set of 4 (2×2) grid squares, which meet the following conditions: *a)* Any square in the set does not be marked as 3 and 2; *b)* The sum of the marked number of these squares is larger than four and the sum of marked number of other sets. Then mark these squares with 2. Repeat this step until no more sets meet these conditions.

(4) Use 512×512, 256×256, and 128×128 crossbars to map the grid square marked with 3, 2, and 1, respectively.

Mixed size crossbar based mapping strategy for 1×1 convolutions. OMM does not work for 1×1 convolutions, so other optimization methods should be considered for higher parallelism, such as the weight matrix duplications. Table 2 shows the simulation results of the area of different Conv layers with different sizes of crossbars. From the results, we know that 1×1 convolutions occupy small chip area. Therefore, duplicating the weight matrix to multiple crossbars is feasible, and the smaller area means the higher parallelism when using the same chip area. Furthermore, the last row of Table 2 shows that the smaller crossbar does not mean the smaller area cost in total because smaller crossbars leads to more interfaces, which damage the area advantage of smaller crossbars. Consequently, for achieving the smallest area for 1×1 convolutions, we should not only use small crossbars for their area advantage, but also consider large crossbars for the less interface overhead. We modified the mapping strategy for regular convolutions to map the 1×1 convolutions on MISCA with the smallest area:

(1) Abstract the 1×1 kernel into a rectangle with the size of $[length, width] = [C_{in}, C_{out}]$. Use a grid to cover the kernel rectangle. The grid consist of $I \times J$ squares with the size of 128×128, (I, J) are calculated in Equ. (13).

$$I = \left\lceil \frac{C_{in}}{512} \right\rceil \times 4; \quad J = \left\lceil \frac{C_{out}}{512} \right\rceil \times 4 \tag{13}$$

If the (i, j) square has covered part of the kernels, then we mark the (i, j) square with 1.

(2) Find a set of 16 grid squares, which meet the following conditions: *a)* Any square in the set does not be marked as 3; *b)* The sum of the marked number of these squares is larger than 6.8 and the sum of marked number of other sets. We choose the number "6.8" because the area of one 512×512 crossbar is 6.8× of one 128×128 crossbar. We mark these squares with 3 and repeat this step until no more sets meet these conditions.

(3) Find a set of 4 grid squares, which meet the following conditions: *a)* Any square in the set does not be marked as 3 and 2; *b)* The sum of the marked number of these squares is larger than 2.5 and the sum of other sets' marked number. "2.5" means the area of one 256×256 crossbar is 2.5× of one 128×128 crossbar. We mark these squares with 2 and repeat this step until no more sets meet these conditions.

(4) Use 512×512, 256×256, and 128×128 crossbars to map the grid square marked with 3, 2, and 1, respectively.

Area-constrained mixed size crossbar based mapping strategy for the entire Conv layers. The former parts present the mapping strategy for one Conv layer. Based on these, we design an area-constrained mixed size crossbar based mapping strategy for the entire Conv layers, as shown in Algorithm 1.

Algorithm 1 Area-constrained mixed size crossbar based mapping strategy for the entire convolutional layers

Input: The CNN model.
Output: Speedup of the entire convolutional layers; number of different size crossbars for each layer.

1: Get run time T_f, area of regular conv layers A_{f1}, and area of 1×1 conv layers A_{f2} @ fixed size crossbar;
2: Set $\{Speedup_i\} = 1$, map on MISCA;
3: Calculate the number of large, middle, and small crossbars:$\{N_1(i), N_2(i), N_3(i)\}$;
4: Get runtime of each layer $\{T_l(i)\}$, area of regular conv layers A_{m1}, and area of 1×1 conv layers A_{m2};
5: **while** $A_{m1} < A_{f1}$ **or** $A_{m2} < A_{f2}$ **do**
6: Calculate runtime of each level $\{T_L(l)\}$, find the most time consuming level: L;
7: Find the bottleneck layer l for level L;
8: **if** l is regular conv **and** $A_{m1} < A_{f1}$ **then**
9: $Speedup(l) = Speedup(l) + 1$;
10: Speedup with OMM;
11: Update $T_l(l)$ and A_{m1};
12: Calculate $N_1(l), N_2(l), N_3(l)$;
13: **end if**
14: **if** l is 1×1 conv **and** $A_{m2} < A_{f2}$ **then**
15: $Speedup(l) = Speedup(l) + 1$;
16: Speedup with weight matrix duplication;
17: Update $T_l(l)$ and A_{m2};
18: Calculate $N_1(l), N_2(l), N_3(l)$;
19: **end if**
20: **end while**
21: Calculate the entire speedup: $Speedup = T_f / \sum_i (T_l(i))$;
22: **return** $Speedup, \{N_1(i), N_2(i), N_3(i)\}$;

In this strategy, we first get results with fixed size crossbars, as Line 1 shows. After that, we set the speedup of each layer to 1, map each layer on MISCA, and get results. This initial procedure is shown in Line 2 to Line 4. If the area is smaller than that of the fixed size crossbars, we can do further optimization. Different from one-layer-acceleration, mapping the entire CNN model needs to find the bottleneck of the critical path, then allocate more computing resources for it. We call all the branches with the same input and output node in CNNs as a level, the runtime of each level is determined by the critical path of it. In the mapping strategy, we use the greedy method. In each iteration, we find the most time-consuming level and the critical layer of this level (Line 6 and Line 7). According to the type of this layer, we use different methods to accelerate and map on MISCA, as shown in Line 8 to Line 18. we repeat this procedure until the mapping reaches the area limit. After that, we calculate the speedup of entire Conv layers and the crossbars allocation scheme (Line 21 and Line 22).

5.4 Mapping FC Layers on MISCA

FC layers have much more data than Conv layers, in VGG, the parameter's number of FC layers is 8× than that of Conv layers.

Table 2: Comparisons between the area and crossbars number of 1×1 convolutions (the area unit is mm^2, and the number in parentheses is the normalized area compared with the regular convolutions of CNNs)

	Only Large Crossbars		Only Middle Crossbars		Only Small Crossbars		MISCA	
	Area	# of Crossbars	Area	# of Crossbars	Area	# of Crossbars	Area	[# of Large, Middle, Small Crossbars]
ResNet-18	0.117(5.70%)	8	0.053(2.59%)	10	0.051(2.49%)	24	0.038(1.85%)	[0,4,8]
ResNet-34	0.117(2.77%)	8	0.053(1.26%)	10	0.051(1.21%)	24	0.038(0.90%)	[0,4,8]
ResNet-50	2.079(93.4%)	142	2.100(94.3%)	396	3.208(144%)	1496	1.682(75.6%)	[64,96,86]

Moreover, FC layers can be represented as one matrix-vector multiplication, so RRAM crossbar can compute them in one cycle, according to the Equ. (2). Therefore, even if OMM does not work for FC layers, FC layers is not the bottleneck of the whole system.

However, because of the huge number of parameters, we need more crossbars to store the weights. If the entire FC layers can be stored in crossbars, we store all the weights and complete the FC layer in one cycle. Otherwise, we need to split the FC layers and only store a part of them in crossbars. For instance, we need 948 large crossbars to map the FC layers of VGG, which exceed the total crossbar number of MISCA. So we need to write new weights into crossbars as soon as the computation finished. But many researchers have demonstrated that write crossbars is slower and consumes more energy than read operations [1][18]. Therefore, there exists a trade-off between area and energy or runtime for mapping FC layers. Fig. 6 shows the simulation results of this trade-off, and the data are normalized to Conv layers for simplifying the contrast, which comes from simulation experiment which is discussed in Section 6. The x-axis represents the number of crossbars we used. According to Fig. 6, we choose 237 large crossbars to map the entire FC layers, for the below considerations: *a)* In MISCA, the number of large crossbars is $64 \times 8 = 512$, besides the 142 crossbars used for Conv layers, at most we have 370 large crossbars; *b)* Compared with using 316 crossbars, we can gain 25% area reduction with 10% energy consumption overhead and 12% runtime increment.

In addition, according to [3], the endurance of RRAM is 10^{12} at most. In one inference, our mapping strategy only need three times of write operation for FC layers in VGG, we think the endurance is enough for 10^{11} times of testing. If we need more inference or the endurance of RRAM is limited, we decide to use more crossbars in our architecture to avoid writing.

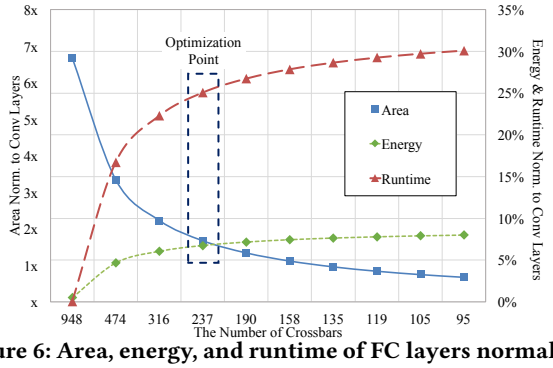

Figure 6: Area, energy, and runtime of FC layers normalized to Conv layers in VGG

6 SIMULATION RESULTS

In this section, we evaluate the performance of MISCA with OMM. In Section 6.1, we describe the experiment setup. Section 6.2 analyses the runtime of MISCA for different networks. The energy efficiency of MISCA is analyzed in Section 6.3. Utilization rate of MISCA is shown in Section 6.4. The area analysis is shown in Section 6.5. Finally, we compare MISCA with PRIME, an existing RRAM-based accelerator, in Section 6.6.

6.1 Simulation Setup

We use five CNN models to evaluate MISCA, i.e., AlexNet [8], VGG-D [7], ResNet-18, ResNet-34, and ResNet-50 [5]. One dataset, ImageNet, is used in these CNNs. The RRAM model we used refers to [1], the cell area is $\sim 0.03\mu m^2$, the read power of the cell is $65\mu W$. The ADC, DAC, SA, and comparator we refer to are shown in [11] [2] [4] [10]. We use MICRON DDR4 SDRAM as global buffer for simulation, whose read bandwidth is $3200MT/s$ and the read power is about $70mW$. The work frequency of MISCA is set to $100MHz$, and the whole system is simulated by MNSIM[17].

All the results are compared with GPU NVIDIA TITAN X, and RRAM based accelerators.

6.2 Runtime Analysis of MISCA

We compare the runtime of MISCA, RRAM based accelerator with fixed size crossbars, and GPU, the results are shown in Table 3. Compared with GPU, MISCA can achieve 4.7 $\sim$ 30.94× speedup, the unsatisfactory result of VGG comes from the slow speed of writing RRAM when mapping FC layers. Compared with fixed size crossbars, MISCA can achieve 1.7 $\sim$ 2.7× speedup for the whole system and 3.1 $\sim$ 6.7× speedup for Conv layers. The speedup of the whole system is limited by the bandwidth of the buffer. Fig. 7 shows the runtime analysis for each CNN model. From the comparison, the time of Conv layers is reduced and the time of data movements becomes the bottleneck. Furthermore, in VGG, the time of FC layers is significantly higher than other CNN models, for the reason that we only need write operations in VGG.

Table 3: The speedup compared with GPU

	ResNet18	ResNet34	ResNet50	AlexNet	VGG
512x512	6.24x	7.33x	7.35x	18.88x	2.33x
MISCA	17.18x	18.7x	13.10x	30.94x	4.7x

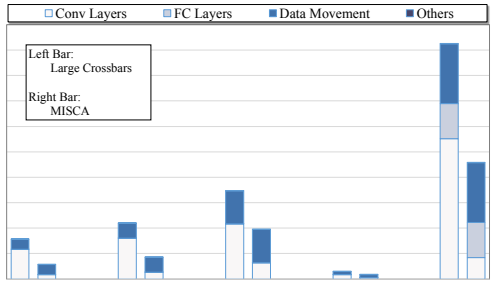

Figure 7: The runtime analysis for each CNN model

6.3 Energy Efficiency Analysis of MISCA

The energy efficiency of RRAM based accelerator compared with GPU is shown in Figure 8. The results show that the energy efficiency of MISCA and large crossbars is higher than that of small

Table 4: The utilization rate of each CNN model.

	ResNet-18		ResNet-34		ResNet-50		AlexNet	VGG
	1 × 1 Conv	Regular Conv	1 × 1 Conv	Regular Conv	1 × 1 Conv	Regular Conv		
512x512	16.80%	59.92%	16.80%	55.88%	88.02%	58.45%	57.16%	79.04%
MISCA	75.00%	92.23%	75.00%	93.65%	99.41%	91.91%	83.62%	94.91%

crossbars, as demonstrated in Section 3. Compared with GPU, MISCA can achieve 139.26 ~ 1041.12× energy efficiency improvement. Compared large size crossbars, MISCA can achieve 2% ~ 22% energy efficiency improvement. The improvement is limited because OMM can only reduce the energy consumption of DACs of the Conv layers. The energy consumption distribution of MISCA is shown in Figure 9(a), from which we know that ADCs/DACs occupy a large part of the energy consumption of RCSes. Thus, the energy efficiency improvement is limited if only reduce the DACs part.

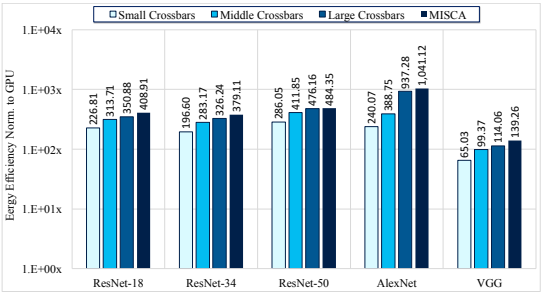

Figure 8: The energy efficiency compared with GPU

6.4 Utilization Rate Analysis

The utilization rate of each CNN model is shown in Table 4. MISCA with OMM can increase the utilization rate about 25% ~ 35% for regular convolutions and 60% for 1 × 1 convolutions in ResNet-18 and ResNet-34. The 512 × 512 crossbars' high utilization rate of ResNet-50 comes from the large number of 1 × 1 kernels which have input channel or output channel larger than 512.

6.5 Area Analysis

The area distribution is shown in Fig. 9(b). The *PEAs* occupy over 95% area, which contain crossbars and peripheral circuits. *IDRC* and *Pooling and ReLU Circuits* take up 2% area. *SUM Circuits* contain an array of 8-bit adders, which takes up 3% area. Besides, the last two columns of Table 2 show that MISCA can map 1 × 1 convolutions with smaller area.

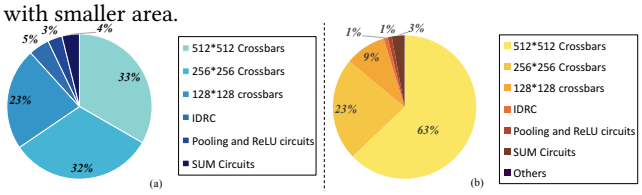

Figure 9: The energy (a) and area (b) distribution of MISCA

6.6 Comparison to PRIME

We compare MISCA with an existing RRAM based accelerator, PRIME[3]. PRIME uses 256 × 256 crossbars for both computation and storage, and it works at 533MHz. We use VGG-D as benchmark to evaluate their performances. Simulation results show that compared with PRIME, MISCA can achieve 26.4× speedup and 1.65× energy efficiency. Note that the buffer and peripheral circuits designs of them are quite different, the results show that MISCA has the potential in improving the speedup and energy efficiency. The ideas of using mixed size crossbars, OMM, and the proposed mapping strategy are also suitable for other architectures like PRIME.

7 CONCLUSION

This paper proposes an Overlapped Mapping Method (OMM) to make full use of the RRAM cells in crossbars when performing the computation of CNNs. MIxed Size Crossbar based on RRAM CNN Architecture, MISCA, is proposed to leverage the advantage of OMM algorithm. An area-constrained mixed size crossbars allocation strategy is proposed to optimize the performance. Simulation results show that MISCA with OMM can achieve 2.7× speedup, 30% utilization rate improvement, and 1.2× energy efficiency improvement on average compared with the fixed size crossbars based accelerator using the conventional mapping method. In comparison with GPU, MISCA can perform 92.6× higher on average in energy efficiency and 20× higher on average in speedup. Compared with an existing RRAM-based accelerator, PRIME, MISCA has 26.4× speedup and 1.65× energy efficiency improvement.

8 ACKNOWLEDGEMENTS

This work was supported by National Key R&D Program of China 2017YFA0207600, National Natural Science Foundation of China (No. 61622403, 61621091), Joint fund of Equipment pre-Research and Ministry of Education (No. 6141A02022608), and Beijing National Research Center for Information Science and Technology (BNRist).

REFERENCES

[1] M. Chang et al. 2014. 19.4 embedded 1Mb ReRAM in 28nm CMOS with 0.27-to-1V read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme. In *IEEE ISSCC, 2014.* 332–333.
[2] S. Y. S. Chen et al. 2011. A 10b 600MS/s multi-mode CMOS DAC for multiple Nyquist zone operation. In *VLSIC, 2011.* 66–67.
[3] P. Chi et al. 2016. PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory. In *ISCA, 2016.* 27–39.
[4] S. Gupta et al. 2014. Simulation and analysis of sense amplifier in submicron technology. *International Journal of Engineering and Technical Research (IJETR).*
[5] K. He et al. 2016. Deep Residual Learning for Image Recognition. In *CVPR, 2016.* 770–778.
[6] M. Hu et al. 2016. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. In *DAC, 2016.* 19.
[7] S. Karen et al. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). arXiv:1409.1556
[8] Alex Krizhevsky et al. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS, 2012.* 1097–1105.
[9] B. Li et al. 2015. MErging the Interface: Power, area and accuracy co-optimization for RRAM crossbar-based mixed-signal computing system. In *DAC, 2015.* 13.
[10] S. Pancholi et al. 2015. A New Design of a CMOS Comparator using 45nm Technology. (2015).
[11] J. Proesel et al. 2010. An 8-bit 1.5GS/s flash ADC using post-manufacturing statistical selection. In *IEEE CICC 2010.* 1–4.
[12] J. Qiu et al. 2016. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In *FPGA, 2016.* ACM, 26–35.
[13] A. Shafiee et al. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-situ Analog Arithmetic in Crossbars. In *ISCA, 2016.* 14–26.
[14] C. Sharan et al. 2014. cuDNN: Efficient Primitives for Deep Learning. *CoRR* abs/1410.0759 (2014). arXiv:1410.0759
[15] L. Song et al. 2017. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In *HPCA, 2017.* 541–552.
[16] Y. Wang et al. 2016. Low power Convolutional Neural Networks on a chip. In *ISCAS, 2016.* 129–132.
[17] L. Xia et al. 2018. MNSIM: Simulation Platform for Memristor-Based Neuromorphic Computing System. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018), 1009–1022.
[18] P. Yao et al. 2017. Face classification using electronic synapses. *Nature Communications* 8 (May 2017), 15199.