# A Ferroelectric FET based Power-efficient Architecture for Data-intensive Computing

Yun Long, Taesik Na, Prakshi Rastogi, Karthik Rao, Asif Islam Khan, Sudhakar Yalamanchili and Saibal Mukhopadhyay
Georgia Institute of Technology, School of Electrical and Computer Engineering
Atlanta, GA, 30318, USA

## ABSTRACT

In this paper, we present a ferroelectric FET (FeFET) based power-efficient architecture to accelerate data-intensive applications such as deep neural networks (DNNs). We propose a cross-cutting solution combining emerging device technologies, circuit optimizations, and micro-architecture innovations. At device level, FeFET crossbar is utilized to perform vector-matrix multiplication (VMM). As a field effect device, FeFET significantly reduces the read/write energy compared with the resistive random-access memory (ReRAM). At circuit level, we propose an all-digital peripheral design, reducing the large overhead introduced by ADC and DAC in prior works. In terms of micro-architecture innovation, a dedicated hierarchical network-on-chip (H-NoC) is developed for input broadcasting and on-the-fly partial results processing, reducing the data transmission volume and latency. Speed, power, area and computing accuracy are evaluated based on detailed device characterization and system modeling. For DNN computing, our design achieves 254x and 9.7x gain in power efficiency (GOPS/W) compared to GPU and ReRAM based designs, respectively.

## 1 INTRODUCTION

Due to the emergence of deep neural network (DNN), acceleration of data-intensive vector-matrix and matrix-matrix operations have received significant attention in recent past. Direct integration of computation and storage within a memory device can fundamentally eliminate the separation between compute and data, thereby enabling orders of magnitude higher energy-efficiency in data-intensive applications. There have been significant efforts in exploiting emerging non-volatile memory (NVM), in particular, resistive random-access memory (ReRAM), to perform in-memory computation [1-4]. The key idea behind the ReRAM based accelerator is utilizing crossbar array to perform vector-matrix multiplication (VMM), which is the major type of computation for DNN. The pioneering works, PRIME [1] and ISAAC [2], demonstrated that ReRAM based DNN accelerators promise much higher computing efficiency than the CPUs/GPUs.

However, when examined closely from a circuit rather than microarchitecture perspective, we note that designing a scalable architecture with ReRAM based in-memory computation remains challenging. First, a crossbar with many parallel ReRAM devices presents a low-impedance resistive load. This is fundamentally at odds with CMOS gates which are designed to drive high-impedance loads (i.e. the gate of MOSFET). Although many prior works neglected this challenge, we show through circuit simulations that power-hungry analog drivers are necessary to ensure accurate computation in ReRAM crossbar. Second, as ReRAM has relatively low on-state resistance ($1K\Omega$ to $100K\Omega$ for $R_{on}$) [5-7], the energy dissipation during VMM operation can be detrimental as all ReRAM devices in the crossbar simultaneously consume read current. Third, constrained by the crossbar size as well as the system capacity, device re-programming are required to solve large problems. The high programming energy (>1 pJ/cell) [5, 8] in ReRAM as well as the in-efficient data movement can degrade the computing efficiency. Further, the ADC and DAC in prior works introduces large overhead for both power and chip area.

We argue that transforming the promise of in-memory computation to a fully-fledged DNN accelerator requires a cross-cutting solutions connecting emerging device technologies, circuit techniques, and micro-architectural supports. Towards this end, we propose a ferroelectric FET (FeFET) based high efficient architecture for DNN acceleration. Our design is built on three core concepts, namely, (i) leverage unique properties of FeFET for ultra-low read/write energy; (ii) exploit the advantages of FeFET to design an all-digital crossbar peripheral, eliminating the ADC/DAC in prior works; and (iii) enable efficient micro-architecture by connecting multiple VMM engines (crossbar and its peripherals) using a hierarchical network-on-chip (H-NoC) with in-router processing, reducing the data transmission volume and latency.

**FeFET as the basic computing cell**: FeFET has similar structure with a normal MOSFET, except it has a ferroelectric layer inside the gate. The polarization of the ferroelectric layer can be switched and retained, thereby, the transistor threshold voltage can be tuned in a non-volatile fashion. The development of FeFET has made tremendous progress in recent years with demonstrations from commercial foundries [9-11]. As a three-terminal transistor device, FeFET provides a high-impedance gate terminal and very high on/off ratio, thanks to its steep switching slope [11]. The high on/off ratio of FeFET ensures high computing accuracy, while low read current (~ 1 nA/cell) and programming energy (~ 1 fJ/cell) reduces crossbar energy [9].

**FeFET based VMM engine**: We leverage the unique characteristics of FeFET to replace the power/are hungry analog peripherals with lightweight all-digital peripheral design. To be more specific: first, rather than the voltage buffer which are required to drive ReRAM, we observe that low-power digital drivers are sufficient to drive the high-impedance gate of FeFET. Second, we replace the power-hungry ADC with the pre-

ICCAD '18, November 5–8, 2018, San Diego, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3240770

charge/discharge circuit and sense amplifier (SA) to realize the function of time-to-digital conversion (TDC).

**Micro-architecture for efficient data communication**: We develop a communication fabric connecting the VMM engines using a hierarchical router network. We propose routers with embedded logic to process the partial results within the NoC. The proposed H-NoC is coupled with optimized partitioning of matrices and data flow to enables efficient and scalable architecture using many VMM engines.

The chip power, area, and computing speed analyses are driven by experimentally calibrated FeFET models, coupled with detail semi-custom design in 28nm CMOS including full-custom (schematic/layout) design of VMM engines and synthesized designs for digital components such as *activation & pooling* unit, H-NoC with in-router processing, data flow controller, etc. The architectural performance is evaluated using a cycle-level simulator for benchmark convolutional neural networks (CNNs). We also analyze the impact of device variations (modeled with stochastic Gaussian noise) to the computing accuracy.

Even though FeFET based crossbar achieves more than 100x lower energy dissipation than ReRAM crossbar, detailed circuit simulations show that simply replacing ReRAM crossbar with FeFET crossbar without optimization for circuit and architecture will lead to only 1.2x reduction in power. This is because the power is dominated by the peripheral circuits rather than the device/crossbar itself. For our VMM engine design, benefiting from the optimized all-digital peripherals, we observe 6.3x power reduction than a conventional ReRAM design. Further, an efficiency optimization is presented that couples the design of VMM engines with the H-NoC to optimize the data flow, reduce data access latency and maximize computing efficiency (GOPS/W). Overall, for the acceleration of DNN inference, our design demonstrates 254x and 9.7x gain in computing efficiency compared with GPU and ReRAM based design, respectively.

# 2 BACKGROUND

## 2.1 ReRAM based VMM Processing Engines

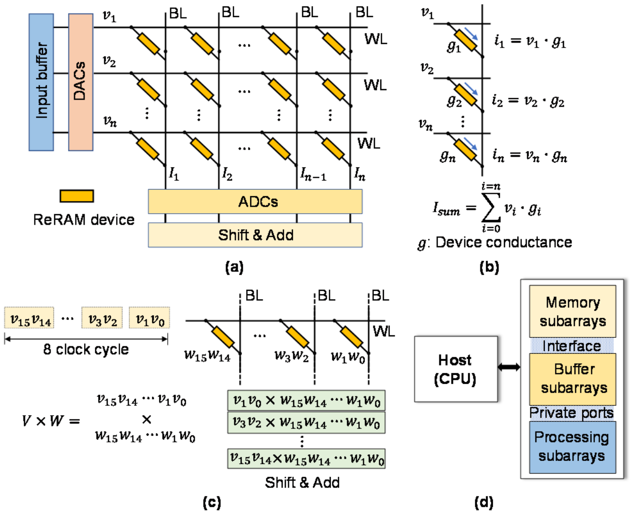


Figure 1: (a) ReRAM based VMM engine. (b) Current summed at bitline based on Kirchhoff's law. (c) 16-bit multiplication. (d) PIM architecture.

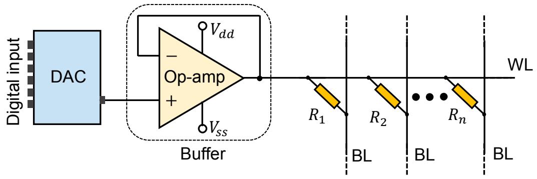


Figure 2: WL buffer is required for ReRAM crossbar.

Figure 1(a) illustrates an ReRAM based VMM engine with the crossbar and peripherals. The DNN's parameters are first programmed into the devices conductance, input vectors are fed as word line (WL) voltage, and the current summed at each bitline (BL) results the multiplication-accumulation (MAC) operation, as shown in Figure 1(b). The digital-to-analog (DAC) and analog-to-digital (ADC) conversions are required in and out of the array.

Note that one-resistor-one-transistor (1T1R) cell is commonly used to increase the selectivity and reduce the leakage current [6]. In practice, rather than storing a whole parameter in a single device, multiple devices connecting to the same WL are used to represent one parameter value [1, 2]. As shown in Figure 1(c), to perform a 16-bit multiplication, 8 devices connecting to the same wordline (WL) are utilized to represent one 16-bit number with each cell stores 2 bits. Similarly, to reduce the overhead of DAC, the input number is divided into several segments and sequentially fed into the crossbar. The final result is summed together with a *shift & add* unit. There are a few recent works explore the ReRAM based processing-in-memory (PIM) architecture where ReRAM array serves for both computation and memory [1, 3], shown in Figure 1(d).

## 2.2 Challenges of ReRAM VMM engine

The first key challenge is **the low on-state resistance ($R_{on}$)** in ReRAM. As illustrated in Figure 2, the WL load of ReRAM crossbar consists of many parallel connected resistors, therefore, a voltage buffer (typically designed with operational amplifier (Op-amp)) with low output-impedance is required to provide large enough current to drive the WL and provide stable WL voltage for inference. This results in increased power dissipation (and chip area overhead) especially for large crossbar arrays. Our SPICE simulation indicates that to ensure a stable WL voltage (i.e. voltage across the ReRAM device), the WL buffer consumes ~10x power over the crossbar itself.

The second key challenge is **the high programming energy**. As DNN becomes more complex and deeper, it is impractical to assume that all DNN parameters can be mapped on chip at once. Therefore, re-programming of crossbars is necessary. As ReRAM programming energy is still very high (~ 1 pJ/cell) [5], the energy-efficiency of ReRAM VMM engine degrades significantly with increasing problem (DNN weights) size.

The third design challenge of ReRAM based VMM engine is **the power and area overheads of DAC/ADC**. For instance, in [2], the ADC occupies almost half of the system power and 45% of chip area.

# 3 FERROELECTRIC FET

Ferroelectric FET is a transistor in which the ferroelectric oxide layer is included in the gate dielectric stack, as shown in Figure 3(a). A ferroelectric oxide is an insulator which exhibits a

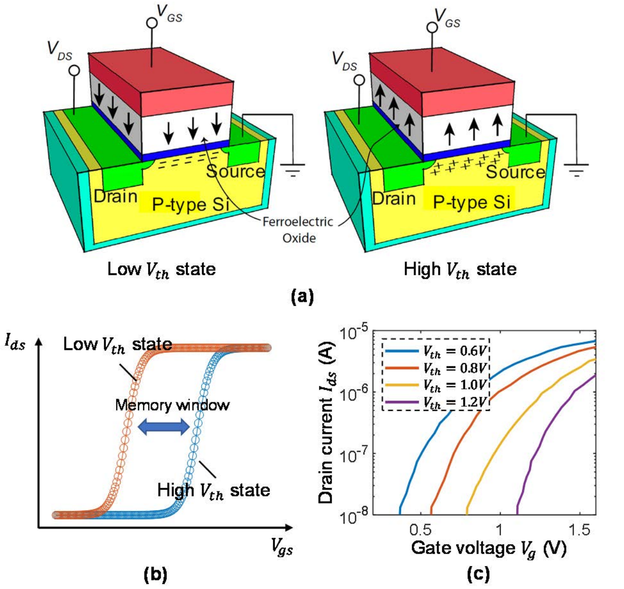


Figure 3: (a) FeFET structure. (b) FeFET hysteresis loop with binary state encoded. (c) Gradual switching of the ferroelectric layer and corresponding I-V characterization [12].

Table I. Comparison between FeFET and ReRAM.

| Device characterization | FeFET [9] | ReRAM [6] |
|---|---|---|
| Write endurance | $10^5$ ($10^9$)* | $10^6$ ($10^{10}$)* |
| Date retention | > 10 years | < 10 years |
| Write speed | 500 ns (10 ns)* | 50 ns (10 ns)* |
| Write energy | ~ 1 fJ | ~ 5 pJ (1 pJ)* |
| On/off ratio | > $10^3$ | < 10 ($10^3$)* |
| Area | 4 $F^2$ | 4 $F^2$ |

* Date in parentheses are the best reported results from literatures.

spontaneous electric polarization in the absence of electric field. The direction of the polarization can be switched by applying a voltage larger than the coercive voltage on the gate terminal of FeFET [10]. As shown in Figure 3(a), when the polarization is pointing downwards, channel is in inversion, bringing the transistor into the 'ON' state (i.e. low $V_{th}$ state). Similarly, if the polarization is pointing upwards, channel is in accumulation which gives the transistor 'OFF' state (i.e. high $V_{th}$ state). Figure 3(b) shows the FeFET hysteresis loop with binary state encoded. Moreover, gradual switching of the ferroelectric layer (i.e. multi-level of threshold voltages and channel conductance) has been demonstrated. Figure 3(c) presents the experimental data showing 4 different levels of transistor threshold voltages [12].

Thanks to the recent discovery of ferroelectricity from silicon doped hafnium oxide (Si:HfO$_2$) [10], the HfO$_2$ thin file based FeFET is transferred to the mainstream CMOS platform with demonstrations from major commercial foundries [9, 12]. It has already been demonstrated that Hafnium oxide FeFET has good temperature stability, writing endurance, data retention and switching speed/energy which make FeFET now comparable or even better than other non-volatile memory candidates such as ReRAM (the comparison is shown in Table 1). The ultra-low writing energy due to the unique electrical field effect switching mechanism is the most prominent feature which distinguish FeFET from other emerging technologies.

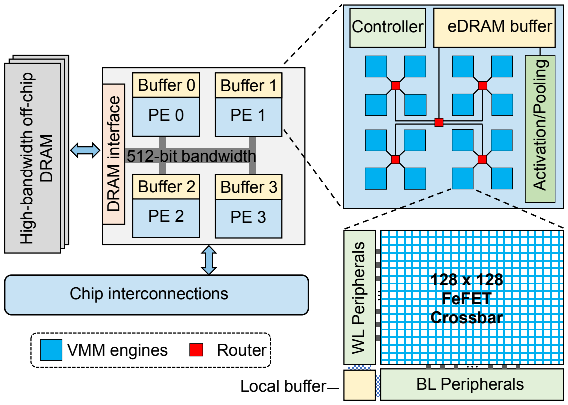


Figure 4: System architecture.

Besides utilizing FeFET as non-volatile memory [9, 10, 12], there have been a few recent works exploring FeFET based logic (AND, OR, etc.) design [13] and binary neural network acceleration (using 4 FeFET cells for XNOR logic) [14]. However, prior works focus on device/crossbar modeling and lack of system/architecture level design. In this work, we propose a fully-fledged system level design combining emerging device technologies, circuit optimization, and architecture innovations.

# 4 SYSTEM DESIGN

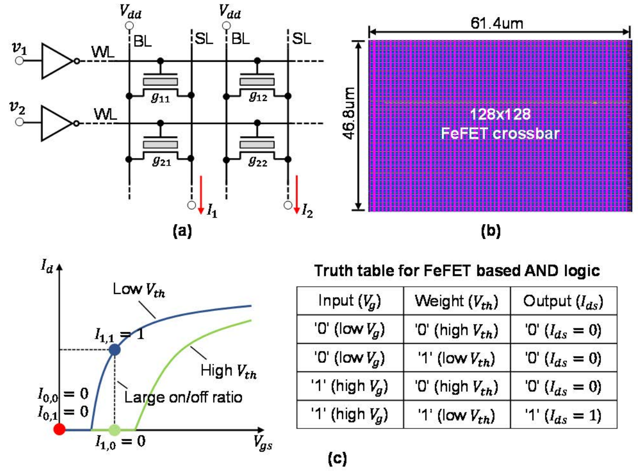
Figure 4 shows the overview of the system architecture consisting of 4 parallel processing engines (PE) connected to an off-chip memory. Inside each PE, there are a set of interconnected VMM engines. Our current design assumes there are 256 VMM engines in each PE. A PE also contains one global buffer to store the temporary input/output data, and an *activation & pooling* unit to handle the activation function and pooling operations. H-NoC is utilized to shuttle data between the buffer and VMM engines. At the bottom level, each VMM engine consists of a FeFET crossbar with 128 × 128 devices, WL/BL peripherals, and a small local buffer.

## 4.1 FeFET based VMM engine

### *4.1.1 FeFET for 1-bit multiplication*

Figure 5(a) shows the configuration of the FeFET crossbar, where gate, drain, source of the transistor are connected to WL, BL and source line (SL), respectively. Figure 5(b) shows the corresponding layout view of a 128 × 128 crossbar under 28nm technology. Weights are stored as transistor channel conductance (i.e. threshold voltage) and input vectors are used to drive WLs (i.e. transistor gate).

Unlike the case of ReRAM where the read current is the direct multiplication of applied voltage (DNN's input) and device conductance (DNN's weight) ($I = V \times G$), FeFET is a field effect device where drain current ($I_{ds}$) depends on the difference between the gate voltage ($V_{gs}$, represents input) and the threshold voltage ( $V_{th}$ , represents weight). Hence, directly performing the multiplication of input and weight is not possible in FeFET. To address this, we employ the FeFET based AND logic [13] to perform the 1-bit multiplication, as shown in Figure 5(c). One-bit



Truth table for FeFET based AND logic

| Input ($V_g$) | Weight ($V_{th}$) | Output ($I_{ds}$) |
|---|---|---|
| '0' (low $V_g$) | '0' (high $V_{th}$) | '0' ($I_{ds} = 0$) |
| '0' (low $V_g$) | '1' (low $V_{th}$) | '0' ($I_{ds} = 0$) |
| '1' (high $V_g$) | '0' (high $V_{th}$) | '0' ($I_{ds} = 0$) |
| '1' (high $V_g$) | '1' (low $V_{th}$) | '1' ($I_{ds} = 1$) |

Figure 5: (a) Configuration of FeFET crossbar. (b) Layout view of a 128 × 128 crossbar under 28nm technology. (c) FeFET based 1-bit multiplication (i.e. AND logic).

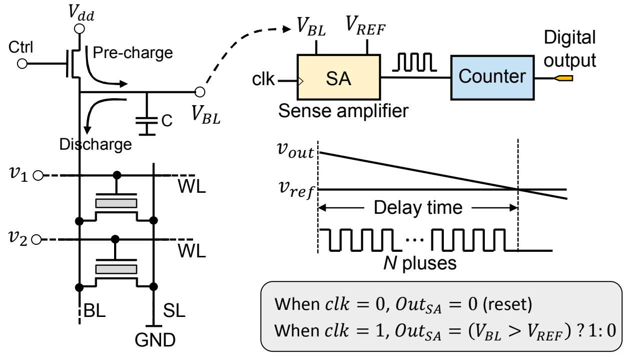
of weight is encoded as high $V_{th}$ or low $V_{th}$, representing either 0 or 1, respectively; similarly, 1 bit of input vector can be encoded as high or low WL voltage ($V_{gs}$). When the input bit is 0 (i.e. low $V_{gs}$), the current is always 0 with either high $V_{th}$ or low $V_{th}$ since the transistor is turned off (red dot in Figure 5(c)). On the other hand, if the input bit is 1 (i.e. high $V_{gs}$), the transistor is still off when $V_{th}$ is high (green dot in Figure 5(c)), but turns on when $V_{th}$ is low (blue dot in Figure 5(c)). The large on/off ratio of FeFET, thanks to its steep subthreshold slope (<60 mV/Dec) [11], creates large difference between the output '1' current and output '0' current.

Another advantage of the proposed FeFET crossbar configuration is that it has a similar architecture with the FeFET memory array [9]. Therefore, the well-developed and chip verified programming scheme can be seamlessly employed in our design.

### *4.1.2 VMM engine peripherals*

One advantage of our design is that now the WL connects to transistor's gate, which is a capacitive load. Therefore, there is no word line voltage drop issue as in the ReRAM scenario. Moreover, since we are performing 1-bit multiplication, there is no need for DAC. This allows us to use digital CMOS for WL peripherals, significantly reducing power dissipation without sacrificing accuracy.

To eliminate the large overhead of ADC, distinguished from prior works and inspired by the reading scheme of the SRAM, we propose a pre-charge/discharge approach as shown in Figure 6. First, the BL is pre-charged to the supply voltage $V_{dd}$. Then, during computing, depending on how many transistors (FeFETs) in the same column are turned on, the BL voltage ($V_{BL}$) drops with different speed. We utilize a sense-amplifier (SA, similar with the one in SRAM) to sample the difference between the reference voltage ($V_{REF}$) and $V_{BL}$ periodically by a clock signal *clk*. When *clk* is low, the output is 0 (reset). When *clk* is high, the output of SA is 1 if $V_{BL} > V_{REF}$, or 0 if $V_{BL} < V_{REF}$. Therefore, within 1 clock cycle, if $V_{BL}$ is larger than the reference voltage, the SA generates a pulse; if not, the output of SA remains 0. We then utilize a counter to count the number of pulses from SA. Basically, with a simple SA and counter, we realize the time to digital converting (TDC). Our SPICE simulation indicates that, for a 128 × 128 FeFET crossbar, our design consumes 2.7x less power than the ADC based approach in ISAAC [2], while achieving the same speed.



When $clk = 0$, $Out_{SA} = 0$ (reset)
When $clk = 1$, $Out_{SA} = (V_{BL} > V_{REF})\ ?\ 1 : 0$

Figure 6: Pre-charge/discharge reading scheme and the SA+counter based TDC design.

### *4.1.3 On-chip memory*

We employ a mix of SRAM and eDRAM as the on-chip memory [2, 15]. The small (128 Byte) local buffer in each VMM engine to store temporary input/output data is implemented using SRAM. Each PE also contains a global buffer implemented using eDRAM. The global buffer receives input data from off-chip DRAM and collect computing results from FeFET arrays. The size of global buffer is 16 KB. In total, there are 192 KB on-chip memory (*local buffer* + *global buffer*) in our system.

On the other side, our system contains 4 PEs with each PE has 256 VMM engines (one 128 × 128 FeFET crossbar inside). In total, the maximum size of DNN parameter can be mapped on our system is 2 MB. Compared with recent ReRAM based work such as ISAAC (30 MB storage capacity consuming 66 W power), our design is very compact in terms of chip area and power, enabling the integration on the mobile and edge devices. With limited storage capacity, we emphasize the necessity of considering device re-programming during computing.

## 4.2 Micro-architecture support

In this subsection we discuss the micro-architectural support including data partitioning and mapping, the communication architecture, and how they are integrated to design a scalable system.

### *4.2.1 Data partitioning and mapping*

Figure 7 illustrates a common approach to partition a large matrix-matrix multiplication operation across multiple VMM engines. Assuming the crossbar inside the VMM engine can hold parameters of size s × s, the weight matrix is then partitioned into several small segments with the granularity of s × s. Each partition is assigned (programmed) to a VMM engine, in total, n × m VMM engine will be used (the definition for *n* and *m* are shown in Figure 7). Similarly, the input matrix is first transposed, partitioned and sequentially fed into the corresponding VMM engines. Note that in Figure 7, different color and shade are used to help tracking the input and weight mapping.

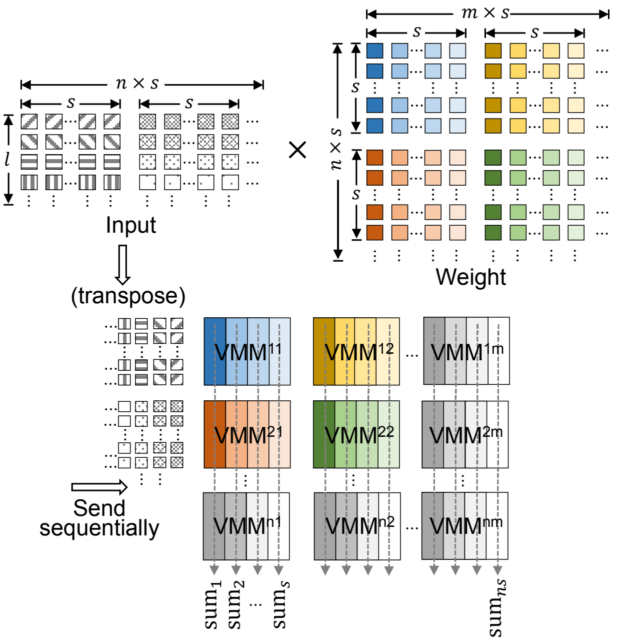


Figure 7: Matrix partition and mapping to multiple VMM engines. Different color and shade are used to help tracking the input and weight mapping.

From Figure 7, we observe that each input segment is shared across multiple VMM engines horizontally (e.g. VMM$^{11}$, VMM$^{12}$, till VMM$^{1m}$). We call it as ***row-wise input sharing***. On the other side, partial results generated from the same column of multiple VMM engines should be summed together vertically (e.g. VMM$^{11}$, VMM$^{21}$, till VMM$^{n1}$ in Figure 7) since they belong to the same column in the original weight matrix. We call it as ***column-wise output summation***.

### *4.2.2 VMM organization and H-NOC design*

As shown in Figure 8(a), VMM engines are organized in a hierarchy fashion with H-NoC for the interconnection. Even though the hierarchical NoC topology is not a new concept [4], we show that our H-NoC is specifically designed to address the discrepancy between *row-wise input sharing* and *column-wise output summation*, reduce data transmission volume and latency.

At the bottom level, 4 VMM engines share a router. Then, 4 such routers are connected to a router in the higher level. Considering 256 VMM engines in a PE, there are 64, 16, 4, and 1 routers exist in different levels (Figure 8(a) only shows 3 levels). Figure 8(b) shows the router design, containing five input/output ports and corresponding I/O buffers. A 5 × 5 switching matrix is equipped to route input/output ports and the routing is based on store-and-forward (SAF) approach. Distinguished from conventional router designs, we insert a computing block (i.e. accumulator) inside the router to enable on-the-fly partial results summation. The benefits of the proposed H-NoC design are in two-fold:

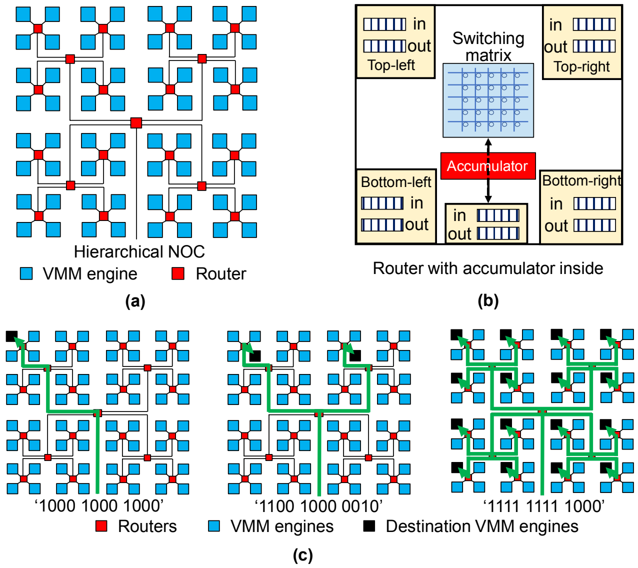
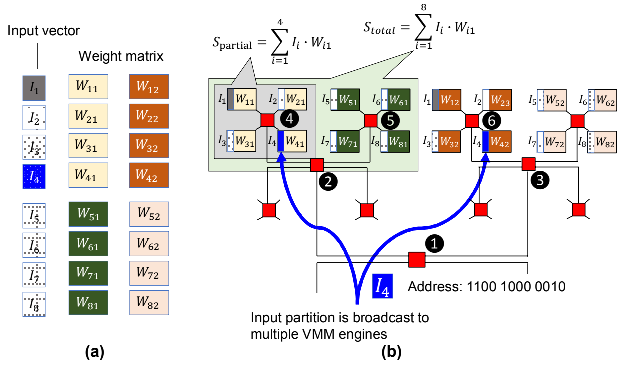


Figure 8: (a) Hierarchical network-on-chip. (b) Router design with accumulator integrated. (c) Three different data forwarding patterns and corresponding addresses, including one-to-one forwarding and broadcasting.

**First, H-NoC is dedicated to realizing efficient row-wise input sharing**. Figure 8(c) illustrates three different data forwarding patterns. The first example shows the one-to-one forwarding. The top-level router decodes the first 4-bit address (each bit represents the on/off of *top-left*, *top-right*, *bottom-right*, *bottom-left* output ports, e.g. '1000' means the packet goes to its top-left branch) and sent the packet to its sub-level router. Then the sub-level router decodes the next 4-bits and repeats until the packet arrives the designated VMM engine at the top-left corner. Besides one-to-one forwarding, the packet can be broadcast. As shown in the last example of Figure 8(c), since the first 4-bit address is '1111', the top-level router broadcasts the packet to its sub-level routers in four directions. This process repeats and finally a single packet is assigned to 16 distributed VMM engines simultaneously.

A case study is used to illustrate how the *row-wise input sharing* benefits from the input broadcasting. As shown in Figure 9(a), a large weight matrix is first partitioned into several segments (we show $2 \times 8 = 16$ segments). Based on our analyses, in Figure 9(a), the inputs are shared horizontally and the outputs are summed vertically. Then, we map $W_{11}$, $W_{21}$, $W_{31}$, $W_{41}$ to 4 VMM engines sharing the same router node ❹ (Again, we use different color and shade to help tracking the input and weight mapping). Then, input



Input vector
Weight matrix
$I_1$ $I_2$ $I_3$ $I_4$ $I_5$ $I_6$ $I_7$ $I_8$
$W_{11}$ $W_{12}$ $W_{21}$ $W_{22}$ $W_{31}$ $W_{32}$ $W_{41}$ $W_{42}$
$W_{51}$ $W_{52}$ $W_{61}$ $W_{62}$ $W_{71}$ $W_{72}$ $W_{81}$ $W_{82}$
$S_{partial} = \sum_{i=1}^{4} I_i \cdot W_{i1}$
$S_{total} = \sum_{i=1}^{8} I_i \cdot W_{i1}$
❶ ❷ ❸ ❹ ❺ ❻
$I_4$
Address: 1100 1000 0010
Input partition is broadcast to multiple VMM engines
(a)
(b)

Figure 9: A case study to illustrate how data are mapped via H-NoC.

vectors are sent to corresponding VMM engines (horizontally sharing). For example: *I$_4$* (in blue) should go to two VMM engines which store *W$_{41}$* and *W$_{42}$*. Conventionally, this requires two packets and two cycles since there are two destination VMM engines. With H-NoC, this can be done with a single packet and one cycle. As shown in Figure 9(b), router ❶ decode the first 4-bit address (1100) and then broadcasts *I$_4$* to its sub-level router at top-left and top-right directions (i.e. sends packet to routers ❷ and ❸). These two routers then decode the next 4-bits (1000) and sent the packet to their top-left router ❹ and ❻. Finally, the packet goes to the bottom-right leaf VMM engines of router ❹ and ❻.

Note that, the broadcasting has a uniform spatial pattern (as in Figure 9(b), I4 is broadcast to bottom-right VMM engines of different regions) which is, coincidentally, in accordance with the regulated weight matrix mapping.

**Second, H-NoC is dedicated for efficient column-wise partial results summation**. Enabled by the in-router accumulator, the results summation is performed on-the-fly, i.e. output summation happens during data transmitting. Again, we use the case in Figure 9 as an example. It takes two steps to get the summation ($S_{total} = \sum_{i=1}^{8} I_i \cdot W_{i1}$). First, router ❹ and ❺ works independently and parallelly, each receiving four partial results from the connected VMM engines and summing the partial data utilizing the built-in accumulator (i.e. $S_{partial} = \sum_{i=1}^{4} I_i \cdot W_{i1}$ and $S_{partial} = \sum_{i=5}^{8} I_i \cdot W_{i1}$). Router ❷ then accumulates the partial results from ❹ and ❺ and sends the final summation to global buffer. Therefore, rather than sending each partial result to the global buffer as separate packets, only 1 packet is sent to the global buffer leveraging the on-the-fly/parallel processing enabled by H-NOC.

In a general case, depending on how many VMM engines are involved for one matrix computing, this process repeats until all the partial results are summed together. As routers in the same level are working in parallel, the worst-case latency is limited to $4 \times$ number of router levels, since it takes 4 clock cycles for a router to accumulate partial results from its 4 branches.

Note that the VMM operation and input transfer are in pipeline, ensuring high transmission rate and clock frequency. Also, the proposed H-NoC is naturally deadlock free since the routing only happens in the up-down directions.

As a conclusion, we argue that the proposed H-NoC design best exploits the large weight-matrix partitioning, input vector broadcasting/share, and output summation. These benefits combine to provide a harmonic, fully-fledged micro-architecture design. Moreover, it can also be employed in other non-volatile memory (such as ReRAM) based DNN accelerator architecture.

## 4.3 Execution model

Figure 10 illustrates the chip-level execution model which contains 6 steps. (1): The controller inside PE asks the memory interface to load data from off-chip memory and stores in the global buffer. (2): The data are then dispatched to VMM engines via H-NoC for computation. (3): After the computing is done, partial results are first summed on-the-fly and then collected back to the global buffer. (4) and (5): The output from VMM engine arrays is fed into the *activation & pooling* unit. The *activation & pooling* unit supports the computation of rectified linear unit (ReLU) and max pooling. (6): The result is sent back to off-chip memory.

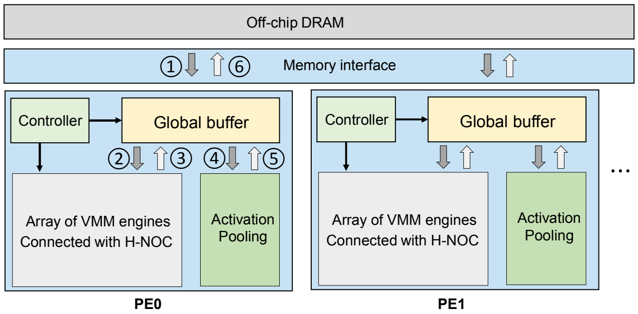


Figure 10. Chip-level execution model.

Table II: Power and area of our system.

| Component | Power (mW) | Area (um$^2$) | Number |
|---|---|---|---|
| WL peripherals | 0.0001 | 0.38 | 128 |
| BL peripherals | 0.011 | 14.8 | |
| Local buffer (128 Bytes) | 0.04 | 972.6 | 1 |
| FeFET crossbar | 0.00098* | 2873.5 | |
| Array total | 1.46 | 5789.1 | 256 |
| Activation/pooling | 19.2 | 165376 | 1 |
| H-NOC | 170.0 | 1616700 | |
| Global buffer (16 KB) | 5.2 | 21000 | |
| controller | 0.48 | 940.3 | |
| PE total | 0.568 W | 3.29 mm$^2$ | 4 |
| Chip total | 2.274 W | 13.14 mm$^2$ | 1 |

* The power number for FeFET crossbar is for reading/inference.

Currently, our system only supports the inference stage of CNN computing. Including training capability is our future work.

# 5 RESULTS

## 5.1 System power and area

We performed SPICE simulation with 28nm CMOS technology using extracted netlist of the crossbar together with the proposed WL/BL peripherals to estimate power and latency of the VMM engine. The SPICE simulation of the VMM engine is then coupled with synthesized digital blocks (such as *shift & add* unit, *activation & pooling* unit, H-NoC, and controller) to form a completed chip-level modeling. Synopsys Design Compiler and PrimeTime are used to model the power and area of the synthesized components. The on-chip memory is modeled with CACTI [16]. To best reduce the off-chip data transmission latency, DRAM with high bandwidth, such as High Bandwidth Memory (HBM) and Hybrid Memory Cube (HMC) is desired. We model the off-chip data access latency with the specification of HMC [17]. Table II summarizes the power and area of each block of our system. The total chip power is 2.27 W, and the chip area is 13.14 mm$^2$.

The benchmark comprises 4 different well-known CNNs, namely, AlexNet, GoogleNet, VGG-16, and VGG-19. We evaluate the benchmarks performance with a large and sophisticated dataset, ImageNet. We evaluate our benchmarks with Caffe deep learning framework running on a state-of-the-art NVIDIA GTX 1080Ti GPU.

## 5.2 Performance analyses

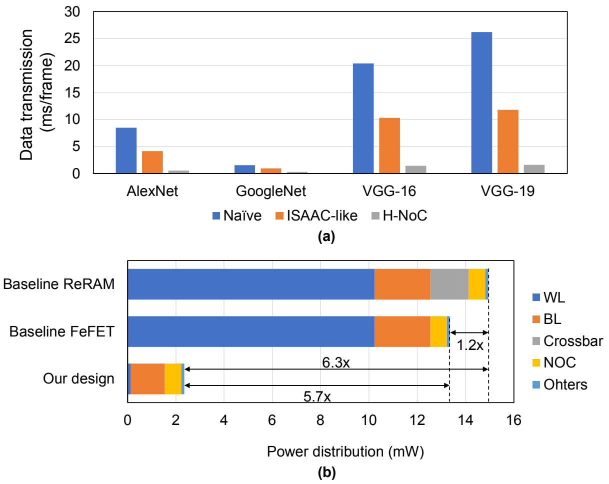


Figure 11: System performance improvements for (a) H-NoC for data transmission and (b) VMM engine for computation.

We first evaluate the performance of our system from two independent aspects: *H-NoC for data transmission* and *VMM engine for computation*.

First, for data transmission efficiency, we compare our H-NoC design with the naive approach (no input broadcasting/reuse or output on-the-fly processing) and ISAAC-like design (using two stage hierarchical buffer for output accumulation) [2]. Figure 11(a) shows the data (input, weights, and internal temporary data) transmission latency for processing one image using 4 different benchmark CNNs. On average, our design reduces the latency by 14.5x and 6.7x over the naive approach and ISAAC-like design across the benchmark CNNs, respectively.

Second, we analyze the power efficiency of FeFET VMM engines and compare with ReRAM based design, as illustrated in Figure 11(b). For the baseline ReRAM design, we consider using ADC in the BL peripherals and insert buffer to drive the WL. With a simple technology replacement from ReRAM to FeFET (still using the same peripherals), we observe that the baseline FeFET based design achieves only 1.2x power reduction because the power consumption on the peripherals (WL buffer and ADC at BL) dominated. Therefore, we argue that only technology replacement (ReRAM->FeFET) does not provide significant advantage at chip and system level. On the other hand, with the optimized digital-like peripherals (i.e. replace the power-hungry ADC with SA based TDC design and also eliminate the WL buffer), significant power efficiency improvement is observed (another 5.7x). In total, with the cross-cutting solutions combining emerging device technologies and circuit innovations, FeFET based VMM engine demonstrates 6.3x power efficiency over the baseline ReRAM design.

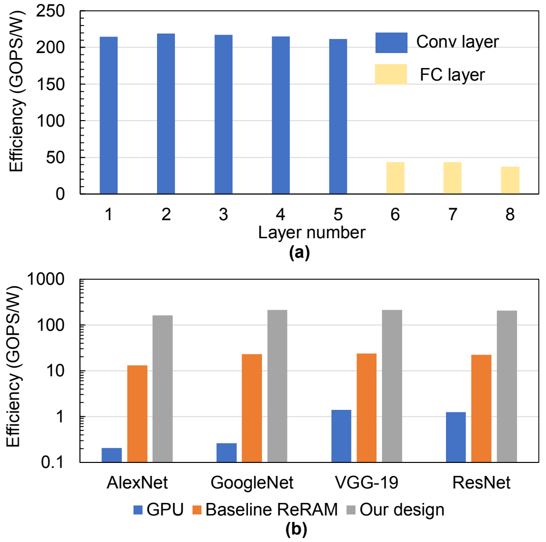


Figure 12: (a) Computing efficiency (GOPS/W) for the layer-by-layer analysis of AlexNet. (b) Computing efficiency of benchmark DNNs and comparison with GPU/baseline ReRAM.

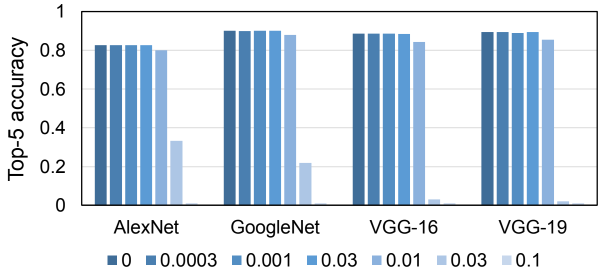


Figure 13: The top-5 ImageNet classification accuracy considering device variation.

We then evaluate the overall efficiency (GOPS/W) on the system level which combine both data transmission as well as computation (including device re-programming). Figure 12(a) shows the layer-by-layer efficiency of AlexNet. We observe that FC layers shows lower efficiency mostly due to large weight matrix requires more crossbar re-programming. We also compare with GPU and ReRAM based design across the benchmark, shown in Figure 12(b). Thanks to high efficient H-NoC and low power FeFET VMM engine, the average computing efficiency of our design across the benchmarks are 254x and 9.7x higher over GPU and ReRAM designs, respectively.

We argue that the device technologies, circuit optimization, together with the micro-architecture innovation, make our work a very computing efficient solution for DNN accelerator when compared with recent NVM based designs.

## 5.3 Computing accuracy

The device variation of FeFET can potentially impact the computing accuracy. Similar with prior ReRAM based design, we use Gaussian noise to represent the stochastic device variation [18]. We calibrate our device variation model with experimental data in recent published works [9, 12]. The typical variation (the standard deviation: σ) varies from 1% to 20%.

Figure 13 shows the classification accuracy deterioration under device variation. The computation shows good robustness when the device variation is low (σ < 3%). However, the accuracy quickly drops to zero when the variation is high, indicating that device with high uniformity is high desired. Should aware that the observation for FeFET here can be seamlessly applied to ReRAM which has similar range of device variation [18].

# 6 COMPARISONS WITH PRIOR WORKS

Table III: Performance comparison with other DNN accelerators.

| | Technology | Hardware platforms | Training support | Parameter storage | Power (W) | Area ($mm^2$) | Computing efficiency (GOPS/W) |
|---|---|---|---|---|---|---|---|
| DaDianNao [15] | 28 nm | ASIC | No | eDRAM (on-chip) | 20.1 | 67.7 | 286.4 |
| ESE [18] | 22 nm | FPGA | No | DRAM (off-chip) | 41 | - | 6.88 |
| ISAAC [2] | 28 nm | ReRAM | No | ReRAM | 65.8 | 85.4 | 380.7 |
| PipeLayer [3] | - | ReRAM | Yes | ReRAM | - | 82.6 | 142.9 |
| Our work | 28 nm | FeFET | No | FeFET | 2.27 | 13.1 | 443.5 |

The high demand for energy efficient execution of deep neural networks have motivated the fast development of DNN accelerators across various platforms including GPU, ASIC [15], FPGA [19], and NVM [1-4]. Among these solutions, NVM based architecture best exploits the in-memory computing and data-level parallelism, largely eliminating the memory wall bottleneck in von-Neumann architecture and providing the unprecedented performance over the conventional approaches.

Beyond the use of a new technology, our design fundamentally differs from prior ReRAM based NVM solution. First, we demonstrate that orders of magnitude increase in the efficiency of FeFET VMM crossbar may not lead to similar performance enhancement at the system level as peripherals dominate system power. Therefore, we present lightweight digital peripherals to increase chip's efficiency. Second, we present a communication fabric, realizing input vector sharing and partial results on-the-fly processing. Third, we propose a compact system design with the emphasis of device re-programming, making the system suitable for power-constrained platforms.

We perform a detailed comparison between our design and accelerators implemented with ASIC, FPGA and ReRAM. The key design features are summarized in Table 3. Should note that the ReRAM efficiency reported in Table 3 is higher than the number in our simulation (Figure 12). This is because prior works did not consider overheads of WL drivers. Also, to have an apple-to-apple comparison, 443.5 GOPS/W for our system is the peak performance without considering the device re-programming.

## I. CONCLUSIONS

We present a FeFET based accelerator design for data-intensive applications. With a cross-cutting solution combining emerging device technologies, circuit optimization, and micro-architectural innovations, state-of-the-art performance is achieved. Our simulation indicates the proposed design improves the computing efficiency by 254x and 9.7x over GPU and ReRAM designs, respectively. As FeFET continues to mature towards a commercial technology, we show the pathway to a high-efficient architecture that successfully leverages unique properties of this technology to accelerate challenging data-intensive computing applications.

## ACKNOWLEDGMENT

This material is supported by the National Science Foundation.

## REFERENCES

[1] P. Chi *et al.*, "PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *ACM SIGARCH Computer Architecture News*, 2016, vol. 44, no. 3, pp. 27-39: IEEE Press.

[2] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News,* vol. 44, no. 3, pp. 14-26, 2016.

[3] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*, 2017, pp. 541-552: IEEE.

[4] D. Fujiki, S. Mahlke, and R. Das, "In-Memory Data Parallel Processor," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 1-14: ACM.

[5] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *Electron Devices Meeting (IEDM), 2012 IEEE International*, 2012, pp. 10.4. 1-10.4. 4: IEEE.

[6] S. Yu *et al.*, "Binary neural network with 16 Mb RRAM macro chip for classification and online training," in *Electron Devices Meeting (IEDM), 2016 IEEE International*, 2016, pp. 16.2. 1-16.2. 4: IEEE.

[7] C. Nail *et al.*, "Understanding RRAM endurance, retention and window margin trade-off using experimental results and simulations," in *Electron Devices Meeting (IEDM), 2016 IEEE International*, 2016, pp. 4.5. 1-4.5. 4: IEEE.

[8] S. Park *et al.*, "Neuromorphic speech systems using advanced ReRAM-based synapse," in *Electron Devices Meeting (IEDM), 2013 IEEE International*, 2013, pp. 25.6. 1-25.6. 4: IEEE.

[9] M. Trentzsch *et al.*, "A 28nm HKMG super low power embedded NVM technology based on ferroelectric FETs," in *Electron Devices Meeting (IEDM), 2016 IEEE International*, 2016, pp. 11.5. 1-11.5. 4: IEEE.

[10] J. Muller, T. S. Boscke, U. Schroder, R. Hoffmann, T. Mikolajick, and L. Frey, "Nanosecond Polarization Switching and Long Retention in a Novel MFIS-FET Based on Ferroelectric $\hbox {HfO} _ {2} $," *IEEE Electron Device Letters,* vol. 33, no. 2, pp. 185-187, 2012.

[11] M. Lee *et al.*, "Physical thickness 1. x nm ferroelectric HfZrOx negative capacitance FETs," in *Electron Devices Meeting (IEDM), 2016 IEEE International*, 2016, pp. 12.1. 1-12.1. 4: IEEE.

[12] H. Mulaosmanovic *et al.*, "Novel ferroelectric FET based synapse for neuromorphic systems," in *VLSI Technology, 2017 Symposium on*, 2017, pp. T176-T177: IEEE.

[13] A. Aziz, "Computing with Ferroelectric FETs: Devices, Models, Systems, and Applications," presented at the DATE, 2018.

[14] X. Y. Xiaoming Chen, Michael Niemier, Xiaobo Sharon Hu, "Design and Optimization of FeFET-based Crossbars for Binary Convolution Neural Networks," presented at the DATE, 2018.

[15] Y. Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014, pp. 609-622: IEEE Computer Society.

[16] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches," *HP laboratories,* pp. 22-31, 2009.

[17] H. M. C. Consortium, "Hybrid memory cube specification 1.0," *Last Revision Jan,* 2013.

[18] B. Gao *et al.*, "Ultra-low-energy three-dimensional oxide-based electronic synapses for implementation of robust high-accuracy neuromorphic computation systems," *ACS nano,* vol. 8, no. 7, pp. 6998-7004, 2014.

[19] S. Han *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 75-84: ACM.