A Low Power In-DRAM Architecture for Quantized CNNs using Fast Winograd Convolutions

Muhammad Mohsin Ghaffar*
Microelectronic Systems Design
Research Group, TU Kaiserslautern
Kaiserslautern, Germany
ghaffar@eit.uni-kl.de

Chirag Sudarshan*
Microelectronic Systems Design
Research Group, TU Kaiserslautern
Kaiserslautern, Germany
sudarshan@eit.uni-kl.de

Christian Weis
Microelectronic Systems Design
Research Group, TU Kaiserslautern
Kaiserslautern, Germany
weis@eit.uni-kl.de

Matthias Jung
Fraunhofer Institute for Experimental
Software Engineering (IESE)
Kaiserslautern, Germany
matthias.jung@iese.fraunhofer.de

Norbert Wehn
Microelectronic Systems Design
Research Group, TU Kaiserslautern
Kaiserslautern, Germany
wehn@eit.uni-kl.de

ABSTRACT

In recent years, the performance and memory bandwidth bottlenecks associated with memory intensive applications are encouraging researchers to explore Processing in Memory (PIM) architectures. In this paper, we focus on DRAM-based PIM architecture for CNN inference. The close proximity of the computation units and the memory cells in a PIM architecture reduces the data movement costs and improves the overall energy efficiency. In this context, CNN inference requires efficient implementations of the area-intensive arithmetic multipliers near the highly dense DRAM regions. Additionally, the multiplication units increase the overall latency and power consumption. Due to this, most previous works in this domain uses binary or ternary weights, which replaces the complicated multipliers with bitwise logical operations resulting in efficient implementations. However, it is well known that the binary and ternary weight networks considerably affect the accuracy and hence can be used only for limited applications.

In this work, we present a novel DRAM-based PIM architecture for quantized (8-bit weight and input) CNN inference by utilizing the complexity reduction offered by fast convolution algorithms. The Winograd convolution accelerates the widely-used small convolution sizes by reducing the number of multipliers as compared to direct convolution. In order to exploit data parallelism and minimize energy, the proposed architecture integrates the basic computation units at the output of the Primary Sense Amplifiers (PSAs) and the rest of the substantial logic near the Secondary Sense Amplifiers (SSAs) and completely comply with the commodity DRAM technology and process. Commodity DRAMs are temperature sensitive devices, hence integration of the additional logic is challenging due to increase in the overall power consumption. In contrast to previous works, our architecture consumes 0.525 W, which is within the range of commodity DRAM thermal design power (i.e. ≤ 1 W). For VGG16, the proposed architecture achieves 21.69 GOPS per device and an area overhead of 2.04% compared to a commodity 8 Gb DRAM. The architecture delivers a peak performance of 7.552 TOPS per memory channel while maintaining a high energy efficiency of 95.52 GOPS/W. We also demonstrate that our architecture consumes 10.1× less power and is 2.23× energy efficient as compared to prior DRAM-based PIM architectures.

*Both authors contributed equally to this research.

MEMSYS 2020, September 28-October 1, 2020, Washington, DC, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8899-3/20/09…$15.00
https://doi.org/10.1145/3422575.3422790

CCS CONCEPTS

• **Hardware → Memory and dense storage**; **Emerging architectures**; • **Computing methodologies → Artificial intelligence**.

KEYWORDS

DRAM, PIM architecture, Neural Networks, Quantized CNN

ACM Reference Format:
Muhammad Mohsin Ghaffar, Chirag Sudarshan, Christian Weis, Matthias Jung, and Norbert Wehn. 2020. A Low Power In-DRAM Architecture for Quantized CNNs using Fast Winograd Convolutions. In *The International Symposium on Memory Systems (MEMSYS 2020), September 28-October 1, 2020, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3422575.3422790

1 INTRODUCTION

Convolution Neural Networks (CNNs) have shown high accuracies in the fields of image recognition and classification. Over the years, a wide variety of CNN inference accelerators [4, 7, 21, 24, 36] have been proposed, which achieves higher performance and lower power consumption as compared to general purpose computing. These accelerators leverage the hardware aware capabilities like custom data types and tailored memory hierarchy for highly optimized architectures. Most of the aforementioned hardware architectures use Dynamic Random Access Memory (DRAM) as the main memory for storing CNN data and parameters. As a result, due to limited DRAM bandwidth, the performance and energy consumption of these accelerators in most cases is dominated by DRAMs latency and energy consumption. Processing in Memory (PIM) architectures aim to resolve the memory bottlenecks by integrating the CNN computation units in a memory device. This approach eliminates the memory bound (energy and bandwidth) issues that are associated with CNN data and parameter accesses [24]. The tight coupling of the memory and computational logic enables massive data parallelism (performance increase) and minimizes the data movement cost, which results in a much higher energy efficiency. Because of these advantages, PIM is an active research domain in recent years.

Researchers have investigated a wide variety for memories for PIM architectures i.e. Resistive Random Access Memory (ReRAM) [8, 26, 34, 60, 61], DRAM [13, 14, 30–32, 51], and Static Random Access Memory (SRAM) [1, 3, 23, 39, 45, 46, 65] memories. However, ReRAMs are firstly not technologically mature as compared to DRAMs or SRAMs. Second, ReRAMs suffer from a large variability in the resistive state values that can influence the final output of a PIM accelerator [6, 17, 25]. The main disadvantage of using a SRAM for CNN inference is its limited memory capacity. The current trend in CNNs is towards deeper toplogies and hence requires significantly higher memory storage. For this reason, SRAM is not suitable for deep CNNs. Unlike SRAM and ReRAM, DRAM satisfy both technological maturity and high memory capacity requirements.

One of the key challenges with the implementation of DRAM-based PIM architectures, especially for multi precision CNNs is that it requires implementation of the Multiply-Accumulate (MAC) operations near DRAM Sub-Array (SA) [51]. DRAM SA design is highly optimized for area, density and yield. Hence, SAs are highly sensitive towards additional logic and constrained by factors like SA width, fine bit-line pitch and the limited number of available metal layers (three) in the DRAM process. Due to this, most of the prior works are confined to binary or ternary weighted networks, which replace the MAC operation with bitwise logical operations. However, binary and ternary weighted networks reduce the accuracy of the network, which makes them unsuitable for many applications. Alternatively, high accuracy can be achieved by using multi-precision CNN networks but at the cost of increase in latency and high power consumption. A commodity DRAM can withstand thermal design power of ≤ 1 W, which makes integration of the additional logic (i.e. multipliers) inside a DRAM challenging.

An approach to reduce the arithmetic complexity of a convolution layer is fast convolution algorithms such as Winograd convolution. The Winograd convolution has shown good accuracy and performance for small kernel sizes i.e. 3x3 and 5x5, which are extensively used in most of the widely used CNN topologies like ResNet [20], VGG [48], GoogleNet [53]. The Winograd convolution includes input transformation, weight transformation and output transformation matrices to convert the input from spatial to Winograd domain and vice-versa that enables reduced arithmetic complexity. For example, a convolution between a 4x4 input tile and a 3x3 kernel, Winograd convolution algorithm reduces the number of multiplications from 36 to 16, which is 2.25x reduction in the number of multiplications than the direct convolution operation without effecting the accuracy [37]. Furthermore, the Winograd convolution converts the standard MAC operation to an accumulate-multiply-accumulate (due to input transformation) operation. These changes make the implementation of the relatively less expensive adder units, as compared to multiplication units, to be integrated near the Primary Sense Amplifier (PSA) region of the DRAM.The rest of the substantial logic i.e. multipliers are placed near the Secondary Sense Amplifiers (SSAs) at the bank-IO-interface, which has less constraints as compared to the PSA region.

In this work, we propose a novel DRAM-based PIM architecture for quantized CNN using Winograd convolution. The presented architecture is fully compatible with a commodity DRAM design and its process. Our approach does not modify the commodity DRAM SA design that is highly optimized for area, density and yield. The goal is to bring the computations closest to the data to minimize the energy and to exploit maximum data parallelism. Hence, our approach integrates the accumulation computations of input transform at the output of the PSA that is the nearest place to the cell, closest to the SA. We only compute partial sums near the PSA circuit and all other substantial computations are performed in the SSA area. Therefore, a very good trade-off between the performance and energy/area is achieved in our work.
The key contributions of this work are:

(1) We present a low power DRAM based PIM architecture for quantized CNN by leveraging the reduced complexity offered by the Winograd convolution algorithm (refer Section 3). To the best of our knowledge, we propose the first PIM architecture for the reduced precision CNN.
(2) Our architecture is fully compatible with the existing commodity DRAM technology, strictly not modifying the DRAM SA design (refer Section 4).
(3) We evaluate area, performance, and energy consumption of our novel architecture for VGG16 benchmark. Furthermore, we compare the presented architecture results with FPGA, GPU, and previous DRAM-based PIM architectures (refer Section 6).

The rest of the paper is organized as follows. Section 2 summarizes related work in terms of PIM architectures and DRAM-based PIM architecture specifically for CNNs inference. Section 3 gives a detailed background on DRAMs and Winograd convolution. Section 4 presents the proposed architecture for quantized-CNN using the Winograd convolution approach. Section 5 and Section 6 provides details about the experimental setup and implementation results and comparisons with the previous works respectively. Finally, Section 7 presents the conclusion of the presented work.

2 RELATED WORK

This section briefly elaborates the prior-work related to PIM architectures. Furthermore, DRAM-based PIM architectures specifically for CNN inference are discussed in detail.

2.1 PIM architectures

Placing the computational logic inside the memory is very appealing for the researchers as it reduces the data movement costs for data intensive applications. In particular, the technique known as PIM offers high performance and low energy consumption as compared to conventional computing systems. The PIM based architectures was first presented by Stones [50], which proposed offloading computation logic inside a memory array. Since than, a number of subsequent works propose different ways to integrate logic inside a memory, which includes FlexRAM [62], Computational RAM [16], DIVA [15]. Most recent works such as active memory cube from IBM [38], Samsung [19], and ARM [64] shows the industry's interest in the presented approach to resolve the problems associated with data intensive applications. Furthermore, PIM architectures for arithmetic and bitwise operations using DRAM [2, 32, 41, 43, 44], PCM [33], SRAM [1, 3, 23, 39, 45, 46, 65] and ReRAM [8, 26, 34, 60, 61] are proposed. Lee [30] proposed a MAC based architecture for matrix-vector multiplication using HBM2-based experimental platform, which is limited to only performing 16 MAC operations per bank. This limits the overall performance of the system, especially for the networks with deeper topologies. In this work, we focus only on DRAM-based PIM, therefore other PIM architectures i.e. SRAM, PCM and ReRAM are declared out of scope for this paper.

2.2 DRAM-based PIM architectures for CNN

In this section, we discuss two types of DRAM- based PIM prior-art architectures for CNN: binary/ternary, and multi-precision. NNDRAM [51], DRISA [32], DrAcc [13], XNOR-POP [22], and NID [47] propose PIM architectures for binary or ternary weighted CNNs. These architectures achieves high energy efficiency and performance by replacing the expensive MAC operations with cheap bitwise operations. Although, this technique is quite efficient for a PIM architecture however, the binary and ternanry weight networks considerably affects the accuracy of deep CNN networks. Furthermore, DRISA [32], and DrAcc [13] architectures are based on Ambit [42]. These architectures rely on the concurrent activation of multiple DRAM rows (by asserting multiple wordlines) and exploit the analog property of the SA bitlines (i.e. charge sharing) to perform bulk bit-wise logic operations. This means a single logic operation output is obtained at the sense amplifiers on each bitline for every multi-row activation. Such outputs can then be combined in order to implement a complex logic function such as multiplication. Nevertheless, there are two main drawbacks of using Ambit-based-approaches. First, they have a high-latency and high-power consumption when performing multi-bit-precision inference. The reason is that they require several iterations of multi-row activations in order to perform a complex logic function such as multiplication (e.g. 8-bit multiplication requires 143 iterations [31]). And second, the process variation in DRAM technology also impacts the precision of the computed result. More specifically, the logical operation failure due to process variation in Ambit is 26% for a process variation of 25%.

Quantization plays a key role in achieving comparative levels of accuracy as compared to full precision (32-bit floating point) networks. The 8-bit quantized CNNs have shown minimal accuracy reduction as compared to full precision networks [67]. Hence, Lee[30], LAcc [14] and SCOPE [31] presented DRAM-based inference architectures for quantized networks. SCOPE uses stochastic computing to preform multi-precision multiplication, which converts integer multiplication into simple bitwise AND operations.

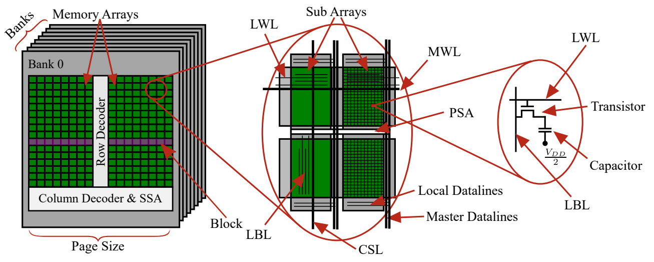

Figure 1: DRAM Device Architecture.

SCOPE proposed to reorganize the DRAM structure with extremely large number of banks i.e. 1024 banks per device. Consequently, the power consumption and area overhead of SCOPE were extremely high. LAcc proposed a lookup tables based approach for fast and accurate vector multiplications. However, LAcc integrated additional logic (to perform XOR operations required for accumulation) in the DRAM SA, which contradicts the DRAM architecture as it is not compatible with the DRAM process.

To date, only a few DRAM-based prior works presented the PIM architectures for multi-precision CNN inference. To the best of our knowledge, this is the first work to exploit the reduced complexity offered by Winograd convolution for an in-DRAM architecture.

3 BACKGROUND

3.1 DRAMs

A DRAM device is organized as a set of banks that include memory arrays, as shown in Figure 1. Each bank consists of row and column decoders, Master Wordline (MWL) drivers, and SSAs. The memory array is designed as an hierarchical structure of SAs (eg. 1024×1024 cells). To access the data from DRAM, an activate command (ACT) is issued to a specific row, which activates the associated Master Wordline Driver (MWLD) and Local Wordline Drivers (LWDs). This initiates the charge sharing between the memory cells and the respective Local Bitlines (LBLs) and the voltage difference is sensed by the PSAs integrated in each SA (E.g. 1024 bit of data). The concurrently sensed data by the PSAs of all the SA in a block (i.e. row of SA) creates an illusion of a large row buffer or a page (eg. Page size = 2KB). After row activation, read (RD) or write (WR) commands are issued to specific columns of this logical row buffer using Column Select Lines (CSL) to access the data via Master Bitline (MBL) and SSAs. In order to activate a new row in the same bank, a precharge (PRE) command has be issued to equalize the bitlines and to prepare the bank for a new activation. Reading or writing the data from/to the PSAs of a block (or page) is performed in a fixed granularity of *I/O data width* $\times$ *bursts length* (e.g. $8 \times 8 = 64$ bit in DDR3).

3.2 Winograd Convolution

Winograd convolution is a fast approach for convolution operation, which reduces the number of multiplications as compared to direct convolution. This approach is based on a minimal filtering algorithm proposed by Winograd [59]. A direct convolution performs MAC operations between the input image and kernels in a raster scan manner. However, the Winograd convolution divides the input

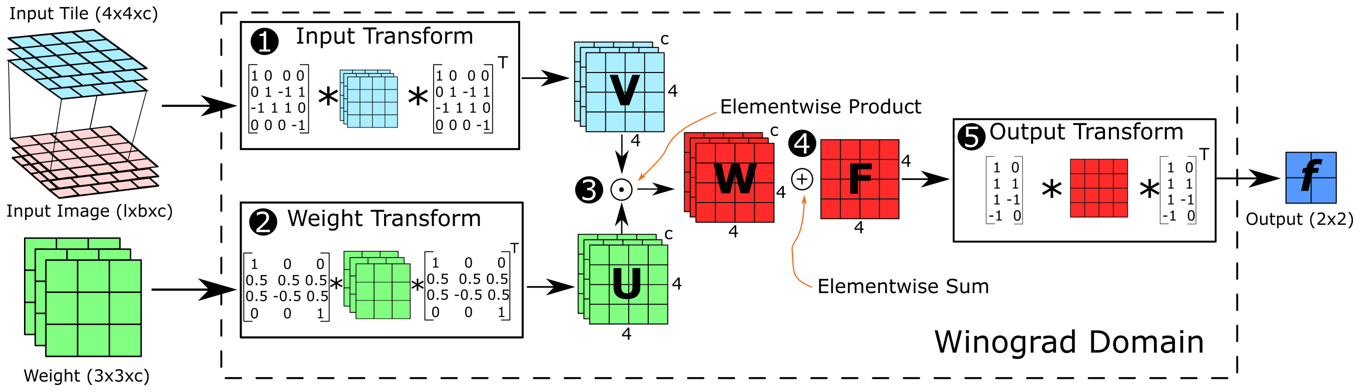

Figure 2: Detailed flow diagram of Winograd convolution *F(2x2,3x3)*

image in tiles and uses hard-coded matrices to transform input and kernel to the Winograd domain. A 2D Winograd operation is represented by *F(m x m,k x k)*, where *m* and *k* are output tile and kernel size respectively. The kernel of size *(kxk)* is applied to an input tile of size *(m+k-1)x(m+k-1)*. The formulas of Winograd convolution for input tile *x* of size *((m+k-1)x(m+k-1))* and kernel *w* of size *(kxk)* are given by:

$$U = (Ww)W^T \tag{1}$$

$$V = X(xX^T) \tag{2}$$

$$f = Z^T[U \odot V]Z \tag{3}$$

Where W (1), X (2), and Z (3) are weight transformation, input transformation and output transformation matrices respectively. Figure 2 depicts the transformation matrices and the Winograd algorithm for an input image of size *(lxbxc)* and a weight matrix of size *(3x3xc)*, where *l*, *b*, and *c* are height, width and channels of input image respectively. For *f(2x2,3x3)* Winograd convolution requires a kernel size of *(3x3)* and input tile of size *(4x4)* extracted by using the aforementioned formula *(3+2-1=4)*. The input transformation (Figure 2-❶) and weight transformation (Figure 2-❷) functions converts the *4x4xc* input tile and the *3x3xc* weights from spatial to Winograd domain respectively. Next, point-wise multiplication (Figure 2-❸) between the resulting (*4x4xc*) U and V matrices is performed. Finally, before output transformation (Figure 2-❺) from Winograd to spatial domain element-wise accumulation (Figure 2-❹) across the channels (*c*) is carried out.

For *f(2x2,3x3)* Winograd convolution, the input and output transformation matrices consists of only 3 values (i.e. 0 , 1 and -1). These matrices can be decomposed and implemented using only the ADD and SUB operations. The weight matrix consists of 0.5 and -0.5 values as well and can be implemented using the SHIFT operation. However, in our implementation the weight transformation is performed offline (directly after the training).

A convolution layer is followed by an activation function, which introduces non-linearity the system. The most common activation functions are *relu*, *tanh*, or *sigmoid* . Furthermore, a *pooling* layer down-samples the input image size. Most common pooling options are maximum *(max)* pooling and average *(avg)* pooling. Finally, The classification scores are computed by an output *Fully-Connected* layer.

4 PROPOSED ARCHITECTURE

In this section, we present our novel DRAM-based PIM architecture for Winograd CNN inference. The architecture exploits the inherent parallelism of the commodity DRAM structure i.e. bank-level and SA level parallelism for enhancing the computation speed. It also employs pipelining techniques to overlap the computations and the data transfers (i.e. output feature map) that minimize write-back-latency. The architecture is designed for the 8-bit input data and 8-bit weights. One of the key aims of this architecture is to retains all the original functionality of the commodity DRAM and to be compatible with the commodity DRAM architecture.

4.1 Top-level Architecture

DRAM architectures offer several levels of parallelism that enable high computation parallelism for PIMs. However, the transfer and storage of output feature maps to/in the memory locations corresponding to the subsequent CNN layers is the major performance bottleneck. Hence, the architecture partitions the number of available banks into two sets referred to as *computation bank-set* and *storage bank-set*. The computation bank-set is responsible for computing the output feature map of an arbitrary CNN layer and forwarding of the resulting output. The storage bank-set concurrently stores these computed results of the output feature map in the appropriate memory locations corresponding to the next CNN layer. This is similar to a pipelining technique and avoids unnecessary write-back latencies. The functionality of the bank-sets is swapped after all the computations of a particular CNN layer are finished. In order to transfer the output feature maps between the bank-sets, the internal data bus and the IO circuitry that manages the transfers of data/addresses between the device IO and the banks are reused.

After each bank computes the required portion of the output feature map, the PIM control circuitry generates a customized read command to the banks in computation bank-set in order to read this output data (SSA region, refer Section 4.4). Note that the customized read command is encoded within the code space of an internal command/address bus in a commodity DRAM and incurs minimal hardware modification to the bank control circuitry. Then, the generated portion of the output feature map from the aforementioned bank is sent into the device IO circuitry using the read path. The PIM control circuitry blocks this read data from being transferred externally and it internally generates ACT and/or WR commands to the appropriate banks of a storage bank-set. The data to be written into the banks of the storage bank-set corresponds to the portion of the output feature map generated by a bank of the computation-bank-set.

In some CNN topologies having residual connections, the output feature maps have to be written to multiple banks. Therefore, the bank select circuitry is slightly modified to select multiple banks. In other words, the internally generated commands and their respective data (i.e output features) are applicable to multiple banks. Hence, the same output features are concurrently written to the addressed memory arrays in multiple banks.

4.2 Bank-Level Architecture

Our micro-architecture partitions the CNN computation into two entities, i.e. the Primary Processing Units (PPUs) and the secondary processing units (SPUs), in order to reduce the circuit complexity near the DRAM SA. This split is required because the surroundings of the DRAM SA is constrained by a limited area, which is not the case near SSA region or the bank-IO-interface that can tolerate larger logic blocks. Hence, to achieve the balance between area and performance, such a split is adapted in our architecture. In order to meet the thermal constraints of a commodity DRAM package, the proposed architecture activates only a single block (i.e. row of SA) per bank, similar to commodity DRAM architecture. Thus, only PPUs corresponding to the activated block are operated concurrently.

The SA level parallelism within a block is exploited to concurrently compute the input feature map channels. Each Column of SAs (COS) is responsible for computing a single channel of a input feature map. Further, each SA is partitioned in two parallel entities (refer Section 4.3). Therefore, a total of 32 channels per bank and 128 channels per computation bank-set are computed in parallel[1]. However, the control flow of all the COS is synchronized.

4.3 Primary Processing Unit

The Primary Processing Unit (PPU) is responsible for performing input transformations of the input tiles corresponding to a single channel. The 2D input transformation are decomposed as follows:

$$\begin{aligned} v_{00} &= (x_{00} - x_{20}) - (x_{02} - x_{22}) & v_{10} &= (x_{10} + x_{20}) - (x_{12} + x_{22}) \\ v_{01} &= (x_{01} - x_{21}) + (x_{02} - x_{22}) & v_{11} &= (x_{11} - x_{21}) + (x_{12} + x_{22}) \\ v_{02} &= -(x_{01} - x_{21}) + (x_{02} - x_{22}) & v_{12} &= -(x_{11} - x_{21}) + (x_{12} + x_{22}) \\ v_{03} &= (x_{01} - x_{21}) - (x_{03} + x_{23}) & v_{13} &= (x_{11} - x_{21}) - (x_{13} + x_{23}) \end{aligned}$$

where, v_{ij} is an element of matrix V (see Figure 2) and x_{ij} is an element of input tile X. For the sake of simplicity only two rows of matrix V is presented in the above equations. It is evident from the formulated input transformation that the partial sums (e.g. $x_{12}+x_{22}$) can be reused. In order to exploit this re-usability, the PPU only computes partial sums. The partial sums are then forwarded to SPU where the input transformation is completed. The main advantage of this approach is that the area of PPU is reduced and it maintains an equal number of words that has to be transferred

[1]Assuming, 8 banks per device and 16 COS per bank

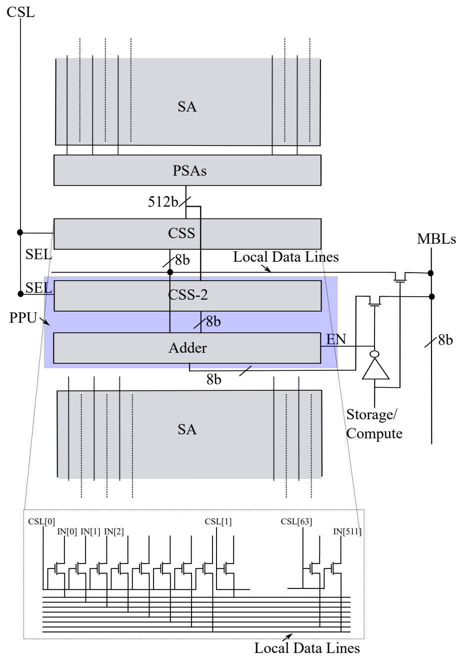

Figure 3: PIM PPU Architecture (PSAs and SAs are also depicted for the sake of clarity).

between PPU and SPU for a given 4×4 tile in comparison to the PPU that performs whole input transformation.

Figure 3 shows the architecture of the PPU. Each SA consists of two PPUs, one on either side of the SA. Similar to PSAs the PPUs are also shared with its neighboring SAs. For the sake of the image clarity, the reference LBL connection to the respective PSAs are not shown in Figure 3. The SA operates in two modes i.e. storage mode and computation mode. In the storage mode the PPU is disabled and the SA operates similar to a commodity DRAM SA. In the computation mode, the PPU is enabled, the local data lines are disconnected from MBLs and connected to PPU, the output of the PPU is then connected to MBLs. The PPU consists of an eight bit adder with a sign input and an additional CSS referred as CCS-2. The operation of the SA in computation mode is as follows:

(1) The PIM bank-level control unit initially sends an ACT command to a row that senses the input data and buffers it in PSAs.
(2) After the sensing is done (i.e. t_{RCD}), the bank-level control unit sends a custom command that enables the appropriate CSLs to select the required inputs to the adder. Note, only two inputs are selected per clock cycle and the sign bit for the adder is set by the control unit.
(3) That adder output is transferred to SPU via MBLs. The output of the adder is 9-bits, but only 8-bits are transferred neglecting the LSB bit. This has a very low impact on the prediction accuracy of the CNN.
(4) Go to step 2, bank-level control unit repeats this cycle until all the data in a given row is processed.
(5) Upon processing all the data in a row, bank-level control unit precharges the row and activates a new row.

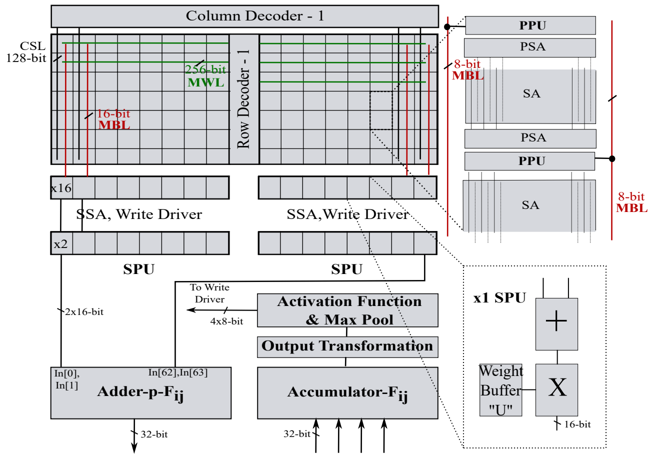

Figure 4: Secondary region architecture

In this architecture a single DRAM row/page (i.e. 1024-bits) is used as two pseudo half pages (i.e 512-bits) in the computation mode. The PSAs on each side of the SA are used as one pseudo half page. The memory controller has to make sure that all the elements of any given input tile (i.e. 4×4) is available in the same pseudo half page. To support this feature the DRAM architecture is slightly modified to split the incoming input data as two individual words and is written to the respective pseudo half page but to the same columns. Note that, this extra feature will not change the DRAM protocol, the memory controller has to reorganize the data that will be written as two individual words. Hence the PPU on each side of the SA are used as an individual entity and operates concurrently. However, the control flow of both the PPU (i.e. activating, selecting operands, transferring PPU output to SPU, precharging) are synchronized.

4.4 Secondary Processing unit

Figure 4 shows the secondary region architecture including the SPU, SSAs, column decoder, row decoder etc. Secondary Processing Units (SPU) are complementary to the PPU. More specifically, the SPU completes the input transformation and performs element wise multiplication of transformed input matrix V with the weight matrix U. SPUs are integrated near the SSAs and has two SPUs per COS in a bank i.e. one SPU per PPU of an active SA. The sequentially accessed partial results from the PPU are accumulated to compute the final result of input transformation i.e. v_{ij} of the matrix V. The obtained v_{ij} of matrix V is multiplied with the corresponding element of matrix U. The weight matrix transformation is performed offline (i.e. directly after training), hence the transformed weight matrices are stored in the DRAM for CNN inference. Before activation of the data row for input transformation, the weight matrix is accessed and stored in a $16 \times 8 - bit$ buffer, referred as weight buffer, in the SPU. The weight buffer is not reloaded until a single channel of the output feature map is fully computed.

The resulting W_{ij} from all 32 channels (i.e. SPU) are added to obtain the partial sum of an element F_{ij} by an adder referred as adder-p-F_{ij}. The partial sum of an element F_{ij} computed in all the banks of computation bank-set are transferred to the appropriate banks of the storage bank-set, where they are accumulated to obtain the final result F_{ij} using the accumulator-F_{ij}. In order to concurrently transfer the partial sums (i.e. 32-bit) from all four banks of computational bank-set, the data width of the internal data bus is set to 128-bits.

The output of accumulator-F_{ij} (i.e. 32-bit) is forwarded to output transformation unit. The output transformation unit is composed of four 32-bit adders i.e. one adder per f_{ij}. The transformation is done sequentially, as and when the elements F_{ij} are available. The final outputs of output transformation unit is sent to activation function and the resulting $4 - bit$ f_{ij} are stored in the appropriate locations for the computation of the subsequent layers. If a max-pooling is required between any two CNN layers, then the output of the activation function is forwarded to max-pooling unit before storing the results in the appropriate locations.

Table 1: Area comparison of 8 Gb DRAMs (estimated vs commodity DRAMs)

Type	Area, [mm^2]
2y nm Estimated	74.4442
20 nm Samsung-DDR4 [9]	59.00
21 nm Gen-1 Hynix-DDR4 [10]	76.0
21 nm Gen-2 Hynix-DDR4 [10]	53.6
18 nm Micron-DDR4 [11]	58.48

4.5 Control Logic

CNN is a data flow rich computation and the operations are highly repetitive, regular and deterministic. Hence, a small state machine with a configurable address range and control sequence is sufficient for the PIM architectures focusing on CNNs. The configuration is done similar to mode register configuration of a commodity DRAM.

5 EXPERIMENTAL SETUP

Our PIM uses a standard commodity state-of-the-art DRAM architecture: 8 Gb DRAM, 8 banks, 64 blocks per bank (a block is a row of SAs), 16 SAs per block, a 1024×1024 standard SA dimension, and a 2KB page. The area calculation of the commodity DRAM SA is performed using the reverse engineered unit cell size and the LBL and LWL pitches (e.g. 66 nm) of 2y nm 8Gb DRAM dies [9, 10]. The area of the other standard DRAM circuits (e.g. decoders, sense amplifiers, etc.) is evaluated using the DRAM architecture exploration tool DRAMSpec [58]. The required 2y nm DRAM technology input parameters by DRAMSpec are extracted from multiple sources and articles [9, 10, 12, 28, 28, 29, 54–57]. To provide an indication of the correctness of our area evaluation methodology, we compare the estimated 8 Gb DRAM die area to 8 Gb commodity DRAM dies from various vendors (Table 1). Note that all the values lie in the similar range.

In [52] it was shown that the peripheral transistor performance in a 2y nm DRAM technology is approximately equivalent to a 65 nm logic technology. For the sake of worst case area, performance and energy, the estimations of the PIM computation units are performed using an adapted (i.e. metal pitch, number of metal layers, etc.) UMC 65 nm Low-Leakage CMOS technology. The number of metal layers for the computation units are restricted to three, reserving the fourth for power lines and MBL. All the computation unit results are extracted post place and route (using Synopsis' IC Compiler).

The latency and energy of the DRAM circuits are estimated using an in-house enhanced circuit-level model that can be equipped with different kind of SPICE transistor models and PDKs, such as FreePDK [49] with PTMs [66], Cadence PDK models, and as well proprietary DRAM transistor models. For the SPICE simulations, we employ BSIM4 model cards built on Low Power PTM transistor models that had to be adapted to 2y nm DRAM technology appropriately, to ensure functional and timing correctness. Additionally, Monte Carlo simulations validate the functionality of the circuits to account for manufacturing process variations [5].

To evaluate the performance of the proposed architecture, we used 8-bit quantized input and 8-bit quantized weight VGG16 [48] for ImageNet dataset [40]. The reason of choosing this model for detailed analysis is based on the fact that VGG is widely used for power and performance analysis of Winograd convolution due to its uniform structure. VGG16 topology consists of 13 convolution layers and 3 fully connected layers. Furthermore, the convolution layers uses *Relu* activation function and *max-pooling* for image size reduction. All the convolution layers use (3x3) kernels with stride 1. In the Winograd domain this translates to *f(2x2,3x3)* with stride 2 (refer Section 3.2).

6 RESULTS

Although, Winograd convolution is numerically different from direct convolution however, it does not effect the accuracy of the convolution operation [18, 27]. Furthermore, 8-bit quantized Winograd shows negligible ($< 0.5\%$) impact on the accuracy as compared to its 16-bit fixed point implementation [63].

In this section, we present detailed evaluations of our architecture in terms of area, power and performance. Furthermore, we present a comparison of the proposed architecture against state-of-the-art DRAM-based PIM architectures.

6.1 Area Evaluation

Based on the methodology described in Section 5, the area evaluations of all the relevant circuits of the presented PIM architecture are shown in Table 2. The presented area results for the PIM enhanced DRAM are evaluated for three architecture variants, (1) all DRAM SAs are enhanced with basic computation units (referred as All-MAC-rows) (2) a limited number of SAs (eight in this case) in each COS are enhanced with basic computation units (referred as 8-MAC-rows) (3) 16-MAC-rows. The preferred variant is 8-MAC-rows as it achieves the highest memory density (bits/mm^2) and the lowest area overhead of 2.04% (measured as additional area in comparison with the baseline standard DRAM die). This enables our architecture to be employed as high memory density host memory with accelerator feature. The 8-MAC-rows architecture variant offer a storage space of 512 kB for storing a single channel of input feature map. The available data space is overwritten after finishing the computation of a particular CNN layer and is sufficient for most of the CNN topology. The weight transformation is performed offline and the transformed weights are stored in non-PIM enhanced SAs.

Table 2: Area evaluation results of all the building blocks of the PIM enhanced DRAM. The area of the PIM dies is measured in terms of an overhead computed on top of the standard DRAM die area.

Block	Area $Height \times Width$ [μm^2]
Sub-array (SA)[a]	73.68×74.036
PSAs (512×), per side of SA	10×74.036
SA with PSAs	93.68×74.036
Bank without computation units	6329.2×1444.58
Column decoder	125×1444.58
Row decoder	6329.2×250
CSS-2	0.5×74.036
Adder (PPU)	5×74.036
SPU	1292
Adder-p-F_{ij}	7521.83
Accumulator-F_{ij}	739.8
Output Transformation Adders (4×)	4×401.4
Activation Function (ReLU)	23
Max-Pooling	548.3
Bank - PIM (8-MAC-rows)	6460.72×1444.58
Bank - PIM (16-MAC-rows)	6548.72×1444.58
Bank - PIM (All-MAC-rows)	7076.72×1444.58
8 Gb Standard DRAM Die	**74.4442 (mm^2)**
8 Gb PIM Die (8-MAC-rows)	(2.04% overhead)
8 Gb PIM Die (16-MAC-rows)	(3.4% overhead)
8 Gb PIM Die (All-MAC-rows)	(9.7% overhead)

[a] including redundancy

Table 3: Latency and Energy of various operations for the core clock frequency of 200 MHz.

Operation	Latency [ns]	Energy [pJ]
ACT (2KB page)	12	614
PRE (2KB page)	10	314
PPU	1.60	0.14
RD/WR (PSA-SSA)	3ns	418/438
SPU	4.50	1.2
Adder-p-F_{ij}	4.78	24.93
Accumulator-F_{ij}	4.63	3.3
Output Transf. Adders (4×)	4.56	6.4
Bank to Bank communication	3 clks	1pJ/bit
Device Background Power	-	34mW

6.2 Latency and Energy Cost of Basic Operations

Using the tools and methodology described in Section 5, we compute the cost in terms of latency and energy of basic operations in the proposed PIM-device. The results are summarized in Table 3. For instance, activating a 2KB bank page (row) takes approx. 12 ns and consumes around 614 pJ.

Table 4: Comparison of DRAM based PIM architectures.

	DRISA [32] [a]	LAcc [14]	SCOPE [31]	This Work 8-MAC-Rows (per device)
DRAM process technology	22nm	25nm	22nm	2ynm
Capacity [Gb]	8	8	8	8
# of Banks	1024	32	1024	8
Page size	4KB [b]	4KB	4KB [b]	2KB
Peak Performance [TOPS]	1.65	0.267	**7.08**	0.059
Power [W]	-	5.3	176.4	**0.525**
Area [mm^2]	258.2	**54.8**	273.38	75.916
Area Overhead [%]	318.27	**<1**	342.86	2.04
Power Density [W/cm^2]	-	9.67	64.53	**0.69**
Energy Efficiency [GOPS/W]	-	50.37	40.13	**112.38**
Normalized Performance/Bank [GOPS] [c]	0.805	4.25	3.45	**7.3**

[a] re-evaluated to 8Gb capacity [31]

[b] single bank has 16 SAs (256 × 2048) and all SAs operate in parallel, equivalent to 4KB page size

[c] normalized to 2KB page size - linear scaling

6.3 Deployment in Servers

Recently, cloud-oriented neural network accelerator designs are emerging. Dedicated hardware-based or FPGA-based accelerator cards are deployed in servers to perform neural network inference [24, 35]. Integrating the proposed DRAM devices, instead of conventional DRAM devices into server memory racks, enables accelerator functionality without requiring any extra accelerator cards for neural network inference. Note that the presented architecture maintains high memory density and retains the host memory functionality of the DRAM devices. A typical DDR4 memory channel configuration in servers is 16 devices per rank, four rank per Dual In-line Memory Module (DIMM) called, Quad Rank Load Reduced DIMM (LRDIMM), and two DIMMs per memory channel. This memory configuration with the proposed 8 Gb DRAM device achieves a peak performance of 7.552 TOPS per channel. Energy efficiency per memory channel is similar to a single PIM device with 15% margin. Deploying the PIM-enhanced DRAM dies in a DDR3 memory channel (i.e. 3 LRDIMMs per channel), then the peak performance increases to 11.564 TOPS per channel.

6.4 Comparison with Other DRAM-based PIMs

Table 4 presents the detailed comparison of our architecture with other DRAM based PIM architectures i.e DRISA [32], LAcc [14] and SCOPE [31]. As mentioned in Section 1, the prior-art PIM architectures re-organize the DRAM with larger page size (i.e 4KB) and a higher number of banks that has a direct impact on the overall throughput. Contrary to this approach, our architecture consists of only 8 banks and has a standard page size of 2KB. For the sake of a fair comparison between these works, Table 4 also presents normalized performance per bank. Our architecture achieves the highest performance (7.3 GOPS) per bank.

The major concerns of LAcc and SCOPE architectures are the high power consumption and power density. LAcc and SCOPE consume 10.1× and 336× high power as compared to our architecture respectively. The high power consumption and power density of LAcc and SCOPE make them infeasible for commodity DRAM technology and packaging. Additionally, such a high power density accelerates the leakage in the DRAM cells. These architectures require special packaging i.e. ceramics and extra cooling to withstand high power consumption. Contrarily, the power consumption (0.525 W) and power density (0.69 W/cm^2) of our architecture are within the accepted range of a commodity DRAM device and its package properties.

The proposed architecture has the highest energy efficiency among all previous architectures. The energy efficiency of our work is 2.23× and 2.80× better compared to LAcc and SCOPE respectively. Hence, fulfilling one of the key aims of this architecture i.e. not only emphasize on the throughput (like DRISA or SCOPE) but rather to be compatible with commodity DRAMs with acceptable power consumption and higher energy efficiency. Additionally, the presented PIM-DRAM chip also satisfies the norms of the high-density main memory (host memory) on top of the CNN accelerator features.

6.5 Comparison with GPU and FPGA

In this section, we compare the performance and power consumption of our proposed PIM architecture with Huang [21] and Lavin[27], which are FPGA and GPU implementations of Winograd convolution respectively. We explicitly consider Huang [21] FPGA implementation because this work includes the influence of external memory on the overall performance and power consumption estimations. Furthermore, both the implementations use reduced precision weights for their implementations.

A bit true model of our architecture is implemented and fed with the data presented in Table 3 to estimate the total performance and energy values. The performance results are depicted in

Table 5: Comparison with FPGA and GPU implementations of Winograd convolution

	GPU [27]	FPGA [21]	This Work [a]
Platform	NVIDIA Titan X	Xilinx VUS440	-
Precision [bits]	16 float	16 fixed	8 fixed
Frequency [MHz]	1126	200	200
CNN topology	VGGE[b]	VGG16	VGG16
Fast convolution	yes	yes	yes
Avg. Performance [GOPS]	**≈7060**	943	21.69
Power [W]	250	25	**0.525**
Energy Effic. [GOPS/W]	28.24	37.72	**41.3**

[a] 8-MAC-Rows, (per device)
[b] VGGE has 16 convolution layers as compared to VGG16, which has 13 convolution layers

Table 5. The GPU has the highest performance among the architectures due to high amount of parallelism offered by the CUDA cores. It can be seen from the results that the proposed architecture consumes 47.6× and 476× less power than FPGA and GPU implementations, respectively. Furthermore, our architecture is 1.46× and 1.09× energy efficient than the corresponding GPU and FPGA implementations. From the presented results, it is evident that the proposed PIM architecture is extremely favourable for low power embedded devices.

7 CONCLUSION

In this paper, a novel DRAM-based PIM architecture for multi-precision CNN inference is presented that is compatible with existing commodity DRAM architecture and process. We used Winograd convolution algorithm to reduce the computation complexity of the convolution layers and to avoid the multiplication first approach of MAC, which is a huge area, power, and performance bottleneck for multi precision CNN in PIM implementations. The area overhead of the proposed architecture is 2.04% compared to a commodity 8 Gb DRAM. Furthermore, the proposed architecture achieves a peak performance of 59 GOPS per device and energy efficiency of 112.38 GOPS/W, which is 2.23× improvement over the prior DRAM-based PIM architectures. Additionally, the power consumption (i.e. 0.525 W) of the presented architecture is within the accepted range of a commodity DRAM device and its package properties. The proposed architecture deployed in server domain achieves a peak performance of 7.552 TOPS per memory channel while still functioning as a high-density host memory. In the future, we aim to extend this work to perform in-memory neural network training.

ACKNOWLEDGMENTS

The project OPRECOMP acknowledges the financial support of the EU FET programme under grant agreement No.732631. This work is also supported by German Academic Exchange Service (DAAD). Furthermore, we thank the anonymous reviewers for their valuable suggestions.

REFERENCES

[1] A. Agrawal, A. Jaiswal, D. Roy, B. Han, G. Srinivasan, A. Ankit, and K. Roy. 2019. Xcel-RAM: Accelerating Binary Neural Networks in High-Throughput SRAM Compute Arrays. *IEEE Transactions on Circuits and Systems I: Regular Papers* PP (04 2019), 1–13. https://doi.org/10.1109/TCSI.2019.2907488

[2] S. Angizi and D. Fan. 2019. Accelerating Bulk Bit-Wise X(N)OR Operation in Processing-in-DRAM Platform. *CoRR* abs/1904.05782 (2019). arXiv:1904.05782 http://arxiv.org/abs/1904.05782

[3] A. Biswas and A. P. Chandrakasan. 2018. Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 488–490.

[4] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'Brien, and Y. Umuroglu. 2018. FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks. *CoRR* abs/1809.04570 (2018). arXiv:1809.04570 http://arxiv.org/abs/1809.04570

[5] K. Chandrasekar, C. Weis, B. Akesson, N. Wehn, and K. Goossens. 2013. Towards variation-aware system-level power estimation of DRAMs: An empirical approach. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–8.

[6] A. Chen and M. Lin. 2011. Variability of resistive switching memories and its impact on crossbar array performance. In *2011 International Reliability Physics Symposium*. MY.7.1–MY.7.4. https://doi.org/10.1109/IRPS.2011.5784590

[7] Y. Chen, T. Krishna, J. S. Emer, and V. Sze. 2017. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits* 52, 1 (Jan 2017), 127–138. https://doi.org/10.1109/JSSC.2016.2616357

[8] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. 2016. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 27–39.

[9] J. Choe. 2017. *Samsung 18 nm DRAM cell integration: QPT and higher uniformed capacitor high-k dielectrics.* https://www.techinsights.com/blog/samsung-18-nm-dram-cell-integration-qpt-and-higher-uniformed-capacitor-high-k-dielectrics

[10] J. Choe. 2017. *SK hynix' 21 nm DRAM Cell Technology: Comparison of 1st and 2nd generation.* https://www.techinsights.com/blog/sk-hynix-21-nm-dram-cell-technology-comparison-1st-and-2nd-generation

[11] J. Choe. 2018. *Micron's 1x DRAMs Examined.* https://www.eetimes.com/author.asp?section_id=36&doc_id=1333289

[12] A. Das. 2012. *Hynix DRAM layout, process integration adapt to change.* https://www.eetimes.com/hynix-dram-layout-process-integration-adapt-to-change/#

[13] Q. Deng, L. Jiang, Y. Zhang, M. Zhang, and J. Yang. 2018. DrAcc: A DRAM Based Accelerator for Accurate CNN Inference. In *Proceedings of the 55th Annual Design Automation Conference* (San Francisco, California) *(DAC '18)*. ACM, New York, NY, USA, Article 168, 6 pages. https://doi.org/10.1145/3195970.3196029

[14] Q. Deng, Y. Zhang, M. Zhang, and J. Yang. 2019. LAcc: Exploiting Lookup Table-based Fast and Accurate Vector Multiplication in DRAM-based CNN Accelerator. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. 1–6.

[15] J. Draper, J. Chame, M. Hall, C. Steele, T. Barrett, J. LaCoss, J. Granacki, J. Shin, C. Chen, C. W. Kang, I. Kim, and G. Daglikoca. 2002. The Architecture of the DIVA Processing-in-Memory Chip. In *Proceedings of the 16th International Conference on Supercomputing* (New York, New York, USA) *(ICS '02)*. Association for Computing Machinery, New York, NY, USA, 14–25. https://doi.org/10.1145/514191.514197

[16] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R. Mckenzie. 1999. Computational RAM: implementing processors in memory. *IEEE Design Test of Computers* 16, 1 (1999), 32–41.

[17] A. Fantini, L. Goux, R. Degraeve, D. J. Wouters, N. Raghavan, G. Kar, A. Belmonte, Y. . Chen, B. Govoreanu, and M. Jurczak. 2013. Intrinsic switching variability in HfO2RRAM. In *2013 5th IEEE International Memory Workshop*. 30–33. https://doi.org/10.1109/IMW.2013.6582090

[18] J. Fernandez-Marques, P. N. Whatmough, A. Mundy, and M. Mattina. 2020. Searching for Winograd-aware Quantized Networks. arXiv:2002.10711 [cs.LG]

[19] Z. Guz. 2014. Real-Time Analytics as the Killer Application for Processing-In-Memory.

[20] K. He, X. Zhang, S. Ren, and J. Sun. [n.d.]. Deep Residual Learning for Image Recognition. *CVPR '15* ([n. d.]).

[21] Y. Huang, J. Shen, Z. Wang, M. Wen, and C. Zhang. 2018. A High-efficiency FPGA-based Accelerator for Convolutional Neural Networks using Winograd Algorithm. *Journal of Physics: Conference Series* 1026 (may 2018), 012019. https://doi.org/10.1088/1742-6596/1026/1/012019

[22] L. Jiang, M. Kim, W. Wen, and D. Wang. 2017. XNOR-POP: A processing-in-memory architecture for binary Convolutional Neural Networks in Wide-IO2 DRAMs. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 1–6. https://doi.org/10.1109/ISLPED.2017.8009163

[23] Z. Jiang, S. Yin, M. Seok, and J. Seo. 2018. XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks. In *2018 IEEE Symposium on VLSI Technology*. 173–174.

[24] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture* (Toronto, ON, Canada) *(ISCA '17)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246

[25] Y.-F. Kao, W. C. Zhuang, C.-J. Lin, and Y.-C. King. 2018. A Study of the Variability in Contact Resistive Random Access Memory by Stochastic Vacancy Model. *Nanoscale Research Letters* 13, 1 (16 Jul 2018), 213. https://doi.org/10.1186/s11671-018-2619-x

[26] S. Ko and S. Yu. 2020. SMART Paths for Latency Reduction in ReRAM Processing-In-Memory Architecture for CNN Inference. arXiv:2004.04865 [cs.AR]

[27] A. Lavin. 2015. Fast Algorithms for Convolutional Neural Networks. *CoRR* abs/1509.09308 (2015). arXiv:1509.09308 http://arxiv.org/abs/1509.09308

[28] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, and S. Hong. 2014. 25.2 A 1.2V 8Gb 8-channel 128GB/s high-bandwidth memory (HBM) stacked DRAM with effective microbump I/O test methods using 29nm process and TSV. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 432–433.

[29] J. C. Lee, J. Kim, K. W. Kim, Y. J. Ku, D. S. Kim, C. Jeong, T. S. Yun, H. Kim, H. S. Cho, Y. O. Kim, J. H. Kim, J. H. Kim, S. Oh, H. S. Lee, K. H. Kwon, D. B. Lee, Y. J. Choi, J. Lee, H. G. Kim, J. H. Chun, J. Oh, and S. H. Lee. 2016. 18.3 A 1.2V 64Gb 8-channel 256GB/s HBM DRAM with peripheral-base-die architecture and small-swing technique on heavy load interface. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. 318–319.

[30] W. J. Lee, C. H. Kim, Y. Paik, J. Park, I. Park, and S. W. Kim. 2019. Design of Processing-"Inside"-Memory Optimized for DRAM Behaviors. *IEEE Access* 7 (2019), 82633–82648.

[31] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie. 2018. SCOPE: A Stochastic Computing Engine for DRAM-Based In-Situ Accelerator. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 696–709.

[32] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie. 2017. DRISA: A DRAM-based Reconfigurable In-Situ Accelerator. In *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 288–301.

[33] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie. 2016. Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.

[34] H. Liu, J. Han, and Y. Zhang. 2019. A Unified Framework for Training, Mapping and Simulation of ReRAM-Based Convolutional Neural Network Acceleration. *IEEE Computer Architecture Letters* 18, 1 (2019), 63–66.

[35] J. Liu, J. Wang, Y. Zhou, and F. Liu. 2019. A Cloud Server Oriented FPGA Accelerator for LSTM Recurrent Neural Network. *IEEE Access* 7 (2019), 122408–122418.

[36] L. Lu, Y. Liang, Q. Xiao, and S. Yan. 2017. Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 101–108.

[37] L. Meng and J. Brothers. 2019. Efficient Winograd Convolution via Integer Arithmetic. *CoRR* abs/1901.01965 (2019). arXiv:1901.01965 http://arxiv.org/abs/1901.01965

[38] R. Nair, S. F. Antao, C. Bertolli, P. Bose, J. R. Brunheroto, T. Chen, C. . Cher, C. H. A. Costa, J. Doi, C. Evangelinos, B. M. Fleischer, T. W. Fox, D. S. Gallo, L. Grinberg, J. A. Gunnels, A. C. Jacob, P. Jacob, H. M. Jacobson, T. Karkhanis, C. Kim, J. H. Moreno, J. K. O'Brien, M. Ohmacht, Y. Park, D. A. Prener, B. S. Rosenburg, K. D. Ryu, O. Sallenave, M. J. Serrano, P. D. M. Siegl, K. Sugavanam, and Z. Sura. 2015. Active Memory Cube: A processing-in-memory architecture for exascale systems. *IBM Journal of Research and Development* 59, 2/3 (2015), 17:1–17:14.

[39] S. Okumura, M. Yabuuchi, K. Hijioka, and K. Nose. 2019. A Ternary Based Bit Scalable, 8.80 TOPS/W CNN accelerator with Many-core Processing-in-memory Architecture with 896K synapses/mm2.. In *2019 Symposium on VLSI Circuits*. C248–C249.

[40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, and et. al. 2015. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision* 115, 3 (Dec. 2015).

[41] V. Seshadri, K. Hsieh, A. Boroum, D. Lee, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry. 2015. Fast Bulk Bitwise AND and OR in DRAM. *IEEE Computer Architecture Letters* 14, 2 (2015), 127–131.

[42] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M.A. Kozuch, O. Mutlu, P. B. Gibbons, and T.C. Mowry. 2017. Ambit: In-memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, Massachusetts) *(MICRO-50 '17)*. ACM, New York, NY, USA, 273–287. https://doi.org/10.1145/3123939.3124544

[43] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry. 2016. Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM. *CoRR* abs/1611.09988 (2016). arXiv:1611.09988 http://arxiv.org/abs/1611.09988

[44] V. Seshadri and O. Mutlu. 2019. In-DRAM Bulk Bitwise Execution Engine. *CoRR* abs/1905.09822 (2019). arXiv:1905.09822 http://arxiv.org/abs/1905.09822

[45] X. Si, J. Chen, Y. Tu, W. Huang, J. Wang, Y. Chiu, W. Wei, S. Wu, X. Sun, R. Liu, S. Yu, R. Liu, C. Hsieh, K. Tang, Q. Li, and M. Chang. 2019. 24.5 A Twin-8T SRAM Computation-In-Memory Macro for Multiple-Bit CNN-Based Machine Learning. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 396–398.

[46] X. Si, Y. Tu, W. Huanq, J. Su, P. Lu, J. Wang, T. Liu, S. Wu, R. Liu, Y. Chou, Z. Zhang, S. Sie, W. Wei, Y. Lo, T. Wen, T. Hsu, Y. Chen, W. Shih, C. Lo, R. Liu, C. Hsieh, K. Tang, N. Lien, W. Shih, Y. He, Q. Li, and M. Chang. 2020. 15.5 A 28nm 64Kb 6T SRAM Computing-in-Memory Macro with 8b MAC Operation for AI Edge Chips. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 246–248.

[47] J. Sim, H. Seol, and L. Kim. 2018. NID: Processing Binary Convolutional Neural Network in Commodity DRAM. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.

[48] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[49] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal. 2007. FreePDK: An Open-Source Variation-Aware Design Kit. In *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*. 173–174.

[50] H.S. Stone. 1970. A Logic-in-Memory Computer. *IEEE Trans. Comput.* 19, 1 (Jan. 1970), 73–78. https://doi.org/10.1109/TC.1970.5008902

[51] C. Sudarshan, J. Lappas, M. M. Ghaffar, V. Rybalkin, C. Weis, M. Jung, and N. Wehn. 2019. An In-DRAM Neural Network Processing Engine. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5.

[52] M. Sung, S. Jang, H. Lee, Y. Ji, J. Kang, T. Jung, T. Ahn, Y. Son, H. Kim, S. Lee, S. Lee, J. Lee, S. Baek, E. Doh, H. Cho, T. Jang, I. Jang, J. Han, K. Ko, Y. Lee, S. Shin, J. Yu, S. Cho, J. Han, D. Kang, J. Kim, J. Lee, K. Ban, S. Yeom, H. Nam, D. Lee, M. Jeong, B. Kwak, J. Park, K. Choi, S. Park, N. Kwak, and S. Hong. 2015. Gate-first high-k/metal gate DRAM technology for low power and high performance products. In *2015 IEEE International Electron Devices Meeting (IEDM)*. 26.6.1–26.6.4.

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. Going Deeper with Convolutions. In *Computer Vision and Pattern Recognition (CVPR)*. http://arxiv.org/abs/1409.4842

[54] TechInsights. 2014. *TECHNOLOGY ROADMAP of DRAM for Three Major manufacturers: Samsung, SK-Hynix and Micron.* https://vdocuments.site/technology-roadmap-of-dram-for-three-major-manufacturers-samsung-sk-hynix.html

[55] TechInsights. 2017. *Samsung 18 nm DRAMAnalysis.* https://www.techinsights.com/blog/samsung-18-nm-dram-analysis

[56] TechInsights. 2018. Micron Technology MT43A4G40200NFA-S15 ES:A HMC Gen2 - Memory Functional Analysis. https://w2.techinsights.com/l/4202/2019-08-28/2hbr19/4202/248106/Sample_Report_MFR_1810_802_Memory_Floorplan_Analysis.pdf.

[57] C. Weis, I. Loi, L. Benini, and N. Wehn. 2013. Exploration and Optimization of 3-D Integrated DRAM Subsystems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32 (2013), 597–610.

[58] C. Weis, A. Mutaal, O. Naji, M. Jung, A. Hansson, and N. Wehn. 2016. DRAMSpec: A High-Level DRAM Timing, Power and Area Exploration Tool. *International Journal of Parallel Programming* 45 (11 2016). https://doi.org/10.1007/s10766-016-0473-y

[59] S. Winograd. 1980. *Arithmetic Complexity of Computations.* Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611970364 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611970364

[60] C. Xue, W. Chen, J. Liu, J. Li, W. Lin, W. Lin, J. Wang, W. Wei, T. Chang, T. Chang, T. Huang, H. Kao, S. Wei, Y. Chiu, C. Lee, C. Lo, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang. 2019. 24.1 A 1Mb Multibit ReRAM Computing-In-Memory Macro with 14.6ns Parallel MAC Computing Time for CNN Based AI Edge Processors. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 388–390.

[61] C. Xue, W. Chen, J. Liu, J. Li, W. Lin, W. Lin, J. Wang, W. Wei, T. Huang, T. Chang, T. Chang, H. Kao, Y. Chiu, C. Lee, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang. 2020. Embedded 1-Mb ReRAM-Based Computing-in- Memory Macro With Multibit Input and Weight for CNN-Based AI Edge Processors. *IEEE Journal of Solid-State Circuits* 55, 1 (2020), 203–215.

[62] Y. Kang, W. Huang, S.-M. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik, and J. Torrellas. 1999. FlexRAM: toward an advanced intelligent memory system. In *Proceedings*

1999 IEEE International Conference on Computer Design: VLSI in Computers and Processors (Cat. No.99CB37040). 192–201.

[63] C. Yang, Y. Wang, X. Wang, and L. Geng. 2019. WRA: A 2.2-to-6.3 TOPS Highly Unified Dynamically Reconfigurable Accelerator Using a Novel Winograd Decomposition Algorithm for Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems I: Regular Papers* 66, 9 (2019), 3480–3493.

[64] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski. 2014. TOP-PIM: Throughput-Oriented Programmable Processing in Memory. In *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing* (Vancouver, BC, Canada) *(HPDC '14)*. Association for Computing Machinery, New York, NY, USA, 85–98. https://doi.org/10.1145/2600212.2600213

[65] Z. Zhang, J. Chen, X. Si, Y. Tu, J. Su, W. Huang, J. Wang, W. Wei, Y. Chiu, J. Hong, S. Sheu, S. Li, R. Liu, C. Hsieh, K. Tang, and M. Chang. 2019. A 55nm 1-to-8 bit Configurable 6T SRAM based Computing-in-Memory Unit-Macro for CNN-based AI Edge Processors. In *2019 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. 217–218.

[66] W. Zhao and Y. Cao. 2007. Predictive Technology Model for Nano-CMOS Design Exploration. *J. Emerg. Technol. Comput. Syst.* 3, 1 (April 2007), 1–es. https://doi.org/10.1145/1229175.1229176

[67] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. 2016. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. arXiv:1606.06160 [cs.NE]