An In-DRAM Neural Network Processing Engine

Chirag Sudarshan[1], Jan Lappas[1], Muhammad Mohsin Ghaffar[1], Vladimir Rybalkin[1],
Christian Weis[1], Matthias Jung[2] and Norbert Wehn[1]

[1] *Technische Universität Kaiserslautern, Germany*
{sudarshan, lappas, ghaffar, rybalkin, weis, wehn}@eit.uni-kl.de
[2] *Fraunhofer Institute for Experimental Software Engineering (IESE), Germany*
matthias.jung@iese.fraunhofer.de

***Abstract*—Many advanced neural network inference engines are bounded by the available memory bandwidth. The conventional approach to address this issue is to employ high bandwidth memory devices or to adapt data compression techniques (reduced precision, sparse weight matrices). Alternatively, an emerging approach to bridge the memory-computation gap and to exploit extreme data parallelism is *Processing in Memory* (PIM). The close proximity of the computation units to the memory cells reduces the amount of external data transactions and it increases the overall energy efficiency of the memory system. In this work, we present a novel PIM based *Binary Weighted Network* (BWN) inference accelerator design that is inline with the commodity *Dynamic Random Access Memory* (DRAM) design and process. In order to exploit data parallelism and minimize energy, the proposed architecture integrates the basic BWN computation units at the output of the *Primary Sense Amplifiers* (PSAs) and the rest of the substantial logic near the *Secondary Sense Amplifiers* (SSAs). The power and area values are obtained at *sub-array* (SA) level using exhaustive circuit level simulations and full-custom layout. The proposed architecture results in an area overhead of 25 % compared to a commodity 8 Gb DRAM and delivers a throughput of 63.59 FPS (Frames per Second) for AlexNet. We also demonstrate that our architecture is extremely energy efficient, 7.25× higher FPS/W, as compared to previous works.**

***Index Terms*—Processing-in-Memory (PIM), DRAM, Binary Weighted Network (BWN), Convolution Neural Network (CNN)**

I. Introduction

Convolution Neural Networks (CNNs) have revolutionized the fields of image recognition and classification. Over the years, a wide variety of CNN accelerators [1]–[3] are implemented to achieve higher performance and lower power consumption as compared to general purpose computing. A current emerging concept in this aspect is *Processing-in-Memory* (PIM) [4]. The aim of PIM architecture is to eliminate the memory bound (energy and bandwidth) issues that are associated with CNN data and parameter accesses [1]. It bridges the memory-computation gap by integrating the CNN computation logic in the memory devices. The tight coupling of the memory and computational logic enables massive data parallelism (performance increase) and minimal data movement cost, which results in a much higher energy efficiency. Because of these advantages, PIM is an active research domain in recent years. In this work, we focus on integrating a *Binary Weighted Network* (BWN)-CNN inference engine into a commodity DRAM architecture. This enables the implementation of CNN with a larger memory footprint and provides higher data access bandwidth, low latency and low power consumption. The prior-art that are most related to our work are XNOR-POP [5], DrAcc [6], and DRISA [7]. XNOR-POP is based on the incorrect premise that the number of SSAs are equal to the DRAM page size. In commodity DRAMs, the number of SSAs in a bank are limited (*I/O data width* × *bursts length*). Thus, the proposed solution in XNOR-POP is contrary to standard DRAM architecture. DRISA and DrACC adapts a concept of concurrently activating multiple rows to realize the basic logic operations (AND, OR, NOT), which was first proposed in Ambit [8]. The major drawbacks of this concept are the process variation in the DRAM SA and high energy consumption due to the numerous multi-row activations. The logical operation failure due to process variation in Ambit is as high as 26% for a process variation of 25%. DrAcc extends Ambit by introducing a new special row inside the sub-array (SA) for shift operations. However, DrAcc neither addresses this issue of Ambit that affect CNN inference nor mentions the CNN resilience factor for such failures. DRISA addressed this issue of Ambit by restructuring and redesign sub-array that is different from the the highly optimized commodity DRAM sub-array design. DRISA also proposes an alternative solution, called *1T1C-adder*, that employs commodity DRAM sub-array. Nevertheless, the authors of DRISA highlight that, *1T1C-adder* is not feasible in a DRAM technology from the energy/area perspective due to large adder trees integrated near the PSA in each SA.

In this paper, we propose a novel DRAM based in-memory BWN architecture called *Neural Network DRAM* (NNDRAM) that is compatible with a commodity DRAM architecture and its process. First, our approach does not intend to modify the commodity DRAM sub-array design that is highly optimized for yield and area. Second, the aim is to bring the computations closest to the data to minimize the energy and to exploit maximum data parallelism. Hence, our micro-architecture partitions the BWN computation unit into two entities. Accordingly, the basic units (i.e. first entity) are integrated in close proximity to the PSAs in the SAs and the other substantial logic (i.e. second entity) are implemented near the SSAs at the bank-IO-interface. The challenging aspect is the physical implementation of the basic unit that is constrained by factors like sub-array width, fine bit-line pitch and the limited number of available metal layers (e.g. three) in the DRAM process. Unlike *1T1C-adder* [7], our work does not integrate a big adder-tree near the PSA circuit, which is constrained by the area/energy bounds. We only compute partial sums in the basic units and the final results are computed in the SSA area.

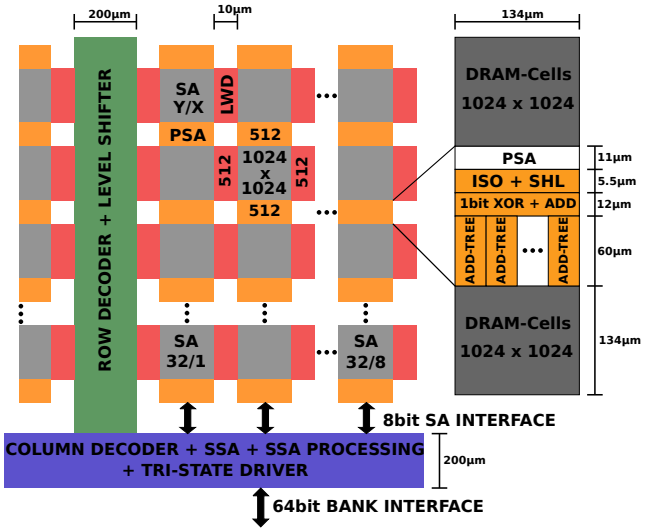


Fig. 1. NNDRAM bank architecture in 65nm

Therefore, a very good trade-off between the performance and energy/area is achieved in this work.

II. Background

1) DRAMs: A DRAM device is organized as a set of banks that include memory arrays. Each bank consists of row and column decoders (see Figure 1), master wordline drivers, and SSAs. The memory array is designed as an hierarchical structure of sub-arrays (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. 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 = 1KB). After row activation, read (*RD*) or write (*WR*) commands are issued to specific columns of this logical row buffer to access the data via 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. 8×8 = 64 bit in DDR3/4) and equal amount of data is fetched (e.g. 8 bits for x8 device) from all the associated SAs (e.g. 8 SAs) of a block.

2) CNNs: A CNN architecture is composed of *convolution*, *pooling*, and *activation* layers. The *convolution* and *pooling* layers are kernel based operations, which perform *Multiply-Accumulate* (MAC) operations and down-sampling respectively between the kernel and corresponding input elements. Most common *pooling* options are maximum *(max)* pooling and average *(avg)* pooling. Finally, the output is squashed with an activation function (i.e. *tanh*, *sigmoid*, or *relu*). The CNN topology is incomplete without an output *Fully-Connected* layer that computes the classification scores. The quantization of CNN parameters is an efficient approach to reduce implementation complexity. The most extreme quantization approach is binarization of weights known as BWN [9]. The authors of [10], [11] present that BWNs (1 bit weights and 2 bit activations) have negligible impacts on the performance as compared to full-precision networks for selected applications.

III. Architecture

This section presents the NNDRAM's architecture. The primary architectural goal is to maximize the data-computation proximity under the constraint to retain the highly optimized commodity SA unmodified. The foremost aspect that is considered in this architecture design is to conserve the regular and repetitive architecture of the DRAM banks. The newly integrated computation units should not break this modularity. Hence, NNDRAM aims to integrate a basic computation unit into each SA at the output of the PSAs, that can be scaled to form a network of dynamic size. The rest of the substantial logic required to process the output of these basic units are integrated in the SSA region. The computational hierarchy complements the memory hierarchy without affecting the bank architecture. We first elaborate the micro-architecture of a basic unit followed by the details of the overall architecture. The proposed architecture is for a state-of-the-art commodity 8 Gb x8 DRAM device, 1024×1024 SA dimension, 16 SAs per block (8 per Half-Bank see Fig. 1), 2 KB page size, 1024 PSAs per SA, and 64 master datalines and SSAs. The architecture is scalable to other DRAMs and DRAM technologies. It is designed for BWN with 1-bit weight ($0 \rightarrow +1, 1 \rightarrow -1$), 2-bit quantized input data and activation function respectively. Furthermore, the *max-pooling* can be performed using either 2×2 or 4×4 kernel dimensions.

A. Basic Computation Unit

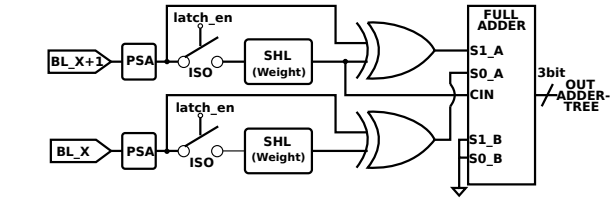
The basic computation unit is presented in Fig. 2.a,b), that implements the partial result of the convolution operation. This architecture considers the kernel dimension for convolution operation as $k \times k \times d$ ($length \times width \times depth$). The first stage of the convolution operation is multiplication or dot product that is realized using XOR gates (as shown Fig. 2.a). It outputs a two's complement number (3-bit) of the input (2-bit) depending on the sign of the weight. The second stage, i.e. adder, has to sum up $k \times k \times d$ digits per kernel. Note that the value d (i.e. depth) varies from layer to layer depending on the topology of the network. Typically such an adder is implemented in a sequential manner. However, such an adder cannot be implemented near PSA due to the lack of clocking signals in a SA. An alternative choice is a combinatorial logic based adder-tree that is feasible only if the number of inputs are fixed. Therefore, we only compute the partial sums corresponding to a single surface ($k \times k$) out of depth d instead of a complete kernel ($k \times k \times d$). These partial sums are then forwarded to a sequential adder integrated in a secondary logic (at the output of the SSAs) where the clock signals are available. The final output of sequential adder after the summation of d partial sums corresponds to a single MAC/convolution operation. One of the key advantages

of computing partial sums is reduced adder-tree depth and reduced area/energy bound issues. The large adder-trees are unfeasible in SA region in DRAM technology. Hence, the maximum dimension of 'k' that has direct effect on the adder tree depth is constrained to maximum of five i.e $max(k) = 5$.

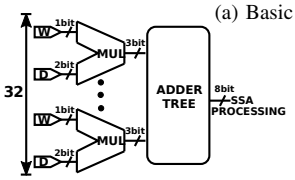
The basic computation unit is constructed out of the 32x multiplication units and an adder-tree (32x, 3-bit inputs) that outputs 8 bit partial sums (as shown Fig. 2b). The main reasons for not confining the basic unit to 25x (i.e. $k \times k$) are balancing the adder tree, and dealing with the manufacturing errors using redundancy. The DRAM SA has spare cells at the end of each row to address manufacturing errors. In case of a manufacturing error, the faulty cells are overtaken by the spare cell in a process called *fixature*. This *fixature* will affect the input wiring of our basic units. Therefore, with such redundant inputs, the input from the column with faulty cell is neglected (forced to zero) and its associated data are mapped to a neighboring column, which is similar to fixature. Hence, the basic computation units are implemented with redundancy so the yield of NNDRAM is the same as commodity DRAM. In any CNN, a particular weight matrix is shared by several convolution computations. The architecture stores weights and the corresponding input data in different rows, but aligned by column (i.e. 1-bit weights are duplicated so as to align with 2-bit input). Thus, a single weight row corresponds to several data rows. Once the weights are latched in shadow-latches (similar to activation) it could be used for several computations by changing only the input data in PSAs (activation of new row). This is also the premise for wiring the inputs of the basic computational units. A single row in a SA stores input data or weight corresponding to sixteen $k \times k$ matrices including the redundancy (assuming a SA width is 1024 bits). Hence, each SA is integrated with sixteen basic units, eight on each side of the SA as shown in Fig 1.

B. Overall Architecture

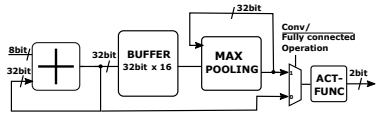
The computation of output feature-maps in each layer of the CNN are independent and parallel. This architecture exploits different levels of parallelism available in DRAM to achieve higher throughput. The associated computation of a single output feature-map is mapped to a single SA in a block. Thus, every block in the DRAM bank computes a maximum of 16 output feature-maps concurrently as there are 16 SAs in a DRAM block. In case the data and the weight associated with the computation of single output feature-map can not be stored in a single SA, they are extended to a neighboring block of the same bank and is aligned by column. If the number of output feature-maps in a CNN layer is greater than 16, the rest of the computations are mapped to a different block either in the same bank or a different bank. The partial sums computed by each basic computation unit in SA are accessed using SSA to be accumulated by a sequential adder, integrated in a secondary logic (as shown Fig. 2c). The access of partial sums are similar to conventional DRAM read methodology except the data is not transferred to IOs. A single bank consists of 16 sequential adders along with max-pooling and activation circuits to align with the number of parallel output feature-



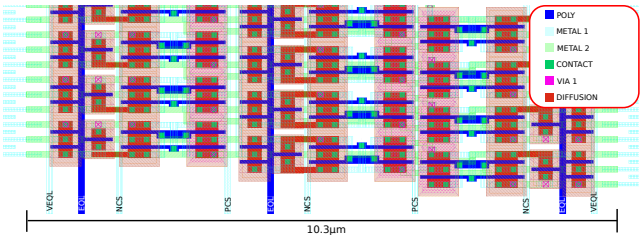
(a) Basic unit at the PSAs



(b) Adder tree



(c) Processing Units at the SSAs



(d) Layout of the PSA

Fig. 2. NNDRAM Computational Units and Layout

map computations. For the max-pooling kernel size of 4×4, the intermediate result of 16 neighbouring MAC operations are buffered in $16 \times 32bit$ buffers respectively. Once the convolution result of all 16 neighbouring MAC operations are completed, the max-pooling and activation functions are triggered to generate a 2-bit output. The 2-bit output is then replicated 8 times in order to distribute (write-back) it to eight SAs of the block associated with the computations of the next CNN layer. The write-backs are in the granularity of 64-bit (equal to the number of MBLs per bank) and a single write-back has the output data of four activation functions (16-bit each). Write-backs are performed using conventional DRAM write methodology that inherently distribute 2-bit output of 4 activation functions to 8 SAs in a block.

Multiple write-backs are performed in order to distribute the 2-bit output of all 16 activation circuit to all SAs of the blocks associated with computation of the next CNN layer. The only extra wiring necessary in our architecture is the global control signals (per block) that enable the CNN computations and indicate the latching of weights in the shadow-latches. These signals are in parallel to the *Master Wordline* (MWL). The control logic required for latching weights, input data, transferring partial sums is performed externally in a memory controller using conventional DRAM commands like ACT, PRE, RD and WR (if the device is configured in accelerator mode). This architecture also enables pipelining i.e when the partial sums of one block is transferred to secondary region, other block of same bank computes the next set of partial sums. Note that the architecture is also applicable for the fully connected layers.

IV. Experimental Setup

This work investigated all the relevant DRAM circuits (e.g. PSAs) and the aforementioned BWN circuits using an

TABLE I
AREA COMPARISON FOR 22NM TECHNOLOGY

Solution	Area	Overhead
DRAM 8Gb 65nm (for scaling)	192.23 mm^2	-
DRAM 8Gb	60.44 mm^2	0.00%
NNDRAM 8Gb with 8 MAC ROWS	75.43 mm^2	24.83%
NNDRAM 8Gb ALL MAC ROWS	114.84 mm^2	89.98%
DRISA 4Gb 3T1C	64.58 mm^2	213.68%
DRISA 4Gb 1T1C-NOR	54.90 mm^2	181.68%
DRISA 4Gb 1T1C-MIXED	65.22 mm^2	215.80%
DRISA 4Gb 1T1C-ADDER	90.91 mm^2	300.83%
DrAcc 8Gb	61.65 mm^2	2.00%

adapted UMC 65nm Low-Leakage CMOS bulk technology. This process technology was modified to emulate a 65nm DRAM process by changing the transistor parameters, restricting the number of metal layers (three metal-layers only), and layout rules, such as the minimum metal pitch. The results of all the extra circuitry are based on the exhaustive circuit level simulations and the full custom layout implementations that are done using Cadence Spectre and Virtuoso, respectively. The rest of the DRAM architecture, including SAs, SSAs, peripherals circuitry, row and column decoders, is evaluated with the DRAM architecture exploration tool DRAMSpec [12]. The clock period of the secondary logic is assumed to be 3ns. Based on the results obtained from the DRAMSpec and the circuit simulation, we calculate the area, power and performance of the NNDRAM device in 65nm that is conservatively scaled to a 22nm node. The CNN benchmarks used in our experiments are AlexNet [13] (ImageNet) and LeNet [14] (MNIST).

V. RESULTS

In this section, we compare the NNDRAM results against the state-of-art. The comparison does not include XNOR-POP due to its infeasible assumptions. All the results of NNDRAM are for a 8 Gb DDR4 DRAM device with sixteen banks.

1) Area: We designed NNDRAM in 65 nm process and scaled the area results to 22nm in order to compare NNDRAM to a 8 Gb commodity DRAM and other state-of-art results (Table I). The presented area results are for the two configurations, (1) all DRAM SAs are enhanced with computation units (referred as all MAC rows) (2) limited number of blocks (eight in this case) per bank are enhanced with computation units (referred as 8 MAC rows). The latter configuration is preferable since the rest of array can be used for conventional memory accesses. NNDRAM's area overhead is 25% for the preferred 8 MAC rows configuration. The area overhead of NNDRAM is evidently better as compared to DRISA even if both architectures are configured for all MAC rows. Although, DrAcc area overhead presents to be smaller as compared to NNDRAM, however DrAcc did not address process variation issues and logic operation failures. Addressing the variation issues like DRISA, the area overhead of DrAcc will be similar to DRISA *1T1C-NOR*.

2) Power Consumption: Table II shows the power consumption comparison for 8 MAC rows. The power overhead

TABLE II
POWER AND PERFORMANCE PER WATT RESULTS FOR ALEXNET

Solution	Weight / Activation	Power (W)	Performance per Watt (FPS/W)
DrAcc [6]	2bit ternary / -	2.1	40.19
DRISA-3T1C [7]	1bit / 8bit	30	19.36
DRISA-NOR [7]	1bit / 8bit	80	13.73
DRISA-MIX [7]	1bit / 8bit	90	14.48
DRISA-ADDER [7]	1bit / 8bit	100	27.27
NNDRAM [this work]	1bit / 2bit	0.218	291.73

of BWN circuits compared to total power consumption is 4%. The remaining power consumption is due to conventional DRAM operations required for the computation. The prime benefit of this architecture is the low power consumption. NNDRAM consumes $9.7\times$ and $137.8\times$ less power as compared to DrAcc and DRISA, respectively. DrAcc and DRISA's high power budget is due to numerous multi-row activations to be performed to realize the basic logic operations. However, NNDRAM computations are based on standard logic gates and the computation units are not integrated in all sub-arrays of a bank. Unlike DRISA that requires active server grade heat sinks and special packaging, NNDRAM is feasible with a typical low cost DRAM packaging.

3) Performance: The throughput of the proposed architecture is 66234 FPS for LeNet and 63.5 FPS for AlexNet and the latency of the basic computation unit is about 2ns. The goal of our new architecture was mainly to optimize energy efficiency, i.e. performance per Watt. The performance (normalized by watts) comparison are shown in table II for 8 MAC rows. Our new architecture outperforms all the other solutions in this aspect. Additionally, DrAcc has long initial latency ($386.4\,s$) due to batch processing. Without batch processing (i.e high-speed mode, as referred in [6]) DrAcc performance reduces further by $23\times$ to 3.63 FPS with a latency of 275ms.

VI. CONCLUSION

We presented a novel PIM accelerator for BWN-CNN inference. The new architecture integrates the basic computation unit at close proximity to the data storage (i.e. output of PSAs) and the rest of the substantial logic near to the SSAs. The evaluation results show that NNDRAM is extremely energy efficient and achieves a very high performance per Watt value (291.73 FPS/W) while executing AlexNet. The small area overhead (25 %) compared to an 8 Gb DDR4 DRAM qualifies this new accelerator architecture to be embedded in upcoming DRAM architectures and devices. In the future, we will extend this work to other input data quantizations and different neural network types.

ACKNOWLEDGMENT

The authors thank Yannick Robin and Chetan Dobariya for their support. This work was initiated in the context of a cooperation with Huawei. The project OPRECOMP acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the European Unions Horizon 2020 research and innovation programme, under grant agreement No.732631 (http://www.oprecomp.eu).

References

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17, Toronto, ON, Canada: ACM, 2017, pp. 1–12.

[2] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[3] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. H. W. Leong, M. Jahre, *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," *CoRR*, vol. abs/1612.07119, 2016. arXiv: 1612.07119.

[4] H. S. Stone, "A logic-in-memory computer," *IEEE Transactions on Computers*, vol. C-19, no. 1, pp. 73–78, Jan. 1970.

[5] L. Jiang, M. Kim, W. Wen, and D. Wang, "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)*, Jul. 2017, pp. 1–6.

[6] Q. Deng, L. Jiang, Y. Zhang, M. Zhang, and J. Yang, "Dracc: A dram based accelerator for accurate cnn inference," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18, San Francisco, California: ACM, 2018, 168:1–168:6.

[7] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17, Cambridge, Massachusetts: ACM, 2017, pp. 288–301.

[8] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17, Cambridge, Massachusetts: ACM, 2017, pp. 273–287.

[9] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *arXiv preprint arXiv:1609.07061*, 2016.

[11] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[12] O. Naji, C. Weis, M. Jung, N. Wehn, and A. Hansson, "A high-level dram timing, power and area exploration tool," in *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, Jul. 2015, pp. 149–156.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.