DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies

Xiaochen Peng, Shanshi Huang, Yandong Luo, Xiaoyu Sun and Shimeng Yu
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA
Email: shimeng.yu@ece.gatech.edu

***Abstract*—**DNN+NeuroSim is an integrated framework to benchmark compute-in-memory (CIM) accelerators for deep neural networks, with hierarchical design options from device-level, to circuit-level and up to algorithm-level. A python wrapper is developed to interface NeuroSim with popular machine learning platforms such as Pytorch and Tensorflow. The framework supports automatic algorithm to hardware mapping, and evaluates both chip-level performance and inference accuracy with hardware constraints. In this work, we analyze the impact of reliability in "analog" synaptic devices, and analog-to-digital converter (ADC) quantization effects on the inference accuracy. Then we benchmark CIM accelerators based on SRAM and versatile emerging devices including RRAM, PCM, FeFET and ECRAM, from VGG to ResNet, and from CIFAR to ImageNet dataset, revealing the benefits of high on-state resistance, e.g. by using three-terminal synapses. The open-source code of DNN+NeuroSim is available at https://github.com/neurosim/DNN_NeuroSim_V1.0.

I. Introduction

To solve the bottleneck of extensive data transfer in the conventional von Neumann architectures, compute-in-memory (CIM) has emerged as a promising paradigm for designing the machine learning hardware accelerator. Recently, the device community has been engineering "analog" synaptic devices for representing the weights in the deep neural network (DNN). These candidates include RRAM [1], PCM [2], FeFET [3] and ECRAM [4], etc. Till today, it still lacks a holistic methodology to evaluate these emerging device properties from the CIM system's perspective. The prior work in IEDM 2017 [5] reported a benchmarking framework named MLP+NeuroSim that could evaluate the impact of device non-ideal properties. However, the prior work was limited to a 2-layer fully connected network for MNIST dataset only, with a focus on the synaptic-array level estimation, while the chip-level peripheries (such as buffers and interconnects) are missing. To enable the machine learning accelerator design to accommodate flexible neural network topologies and support large-scale datasets such as CIFAR and ImageNet, it is crucial to develop a new simulator with comprehensive hierarchical design options from device-level, to circuit-level and up to algorithm-level.

In this work, we propose an end-to-end benchmarking framework for the inference engine with offline training. Our approach is to build a python wrapper to interface NeuroSim, a hardware macro model, with popular machine learning platforms such as Pytorch and Tensorflow. This approach could enable the exploration of CIM accelerator designs for flexible network topologies such as VGG-8 [6] for CIFAR-10, and ResNet-18 [7] for ImageNet, as well as versatile device technologies from CMOS (e.g. SRAM) to beyond-CMOS (two-terminal and three-terminal non-volatile memories).

II. Integrated Framework Principles

A. Framework Structure

Fig. 1 shows the framework structure of DNN+NeuroSim. Flexible DNN topologies are setup in python wrapper (Pytorch and Tensorflow based on low precision training method WAGE [8]), while the weight precision (limited by device multilevel states) and partial sum quantization (limited by ADC precision) are introduced during the software offline training phase, and device retention degradation model [9] is introduced during the hardware inference phase. Fig. 1 (c) shows the simulator taking network topology as input to automatically design the chip floorplan (considering layer by layer computation), while weight-duplication [10] is introduced to maximize memory utilization (defined as percentage of the used memory over the total memory). This is a feature needed for convolutional layer where the unrolled kernel size is smaller than the memory sub-array size, in order to speed up DNN processing. Fig. 1 (d) shows during inference in python wrapper, the traces of synaptic weights and neural activations are unrolled, saved and sent to NeuroSim core, then partitioned and assigned to different locations of the chip according to the automatic floor planning rule. The top-down hierarchy of the CIM system is defined as chip, tile, processing element (PE) and synaptic array. The framework outputs include the hardware-constrained inference accuracy (from python wrapper), and hardware metrics such as chip area, latency, dynamic energy, leakage power, as well as energy efficiency and throughput (from NeuroSim core) for layer-by-layer computation mode. The modular circuit component estimation are all calibrated by SPICE simulations across technology nodes with PTM models.

B. Architecture of CIM Accelerators

Fig. 2 shows the detailed system architecture from chip level down to synaptic-array level. In different levels, peripheries are introduced, including buffers, interconnects (based on H-tree routing), neural-functional units (such as pooling, accumulation and activation). The synaptic array could be implemented by SRAM, two-terminal devices such as RRAM, PCM and STT-MRAM, or three-terminal devices such as FeFET and ECRAM,

978-1-7281-4032-2/19/$31.00 ©2019 IEEE

while both sequential (row-by-row) and parallel read-out schemes are available for each of these device technologies.

III. Behcmark Results

In CIM inference engine, data retention and ADC quantization are the key factors for inference accuracy degradation. Here we evaluate their impacts and benchmark across technologies on an accelerator design based on VGG-8, for CIFAR10 dataset, with 8-bit weight and 8-bit activation precision.

A. Data Retention

As the analog intermediate state retention still needs more experimental characterization, we consider four representative scenarios of conductance drift [9], which are drifting to maximum, minimum or intermediate states, and random drift. Fig. 3 (b-d) show the inference accuracy as a function of time, while the conductance are assumed to drift towards different final states (from -1 to 1, according to the algorithm weight range), or randomly drift, based on three various drift rates, which are equivalent to conductance drift by 2%, 6% and 10 % over 10 years, respectively. The results show that, in scenarios with fixed drift directions, drifting to maximum or minimal states degrade the accuracy faster than drifting to the middle states, while the random drift is the best scenario for maintaining inference accuracy even for 10 years (assuming the equivalent conductance drift by 2%).

B. ADC Quantization

Typical weight matrix size is larger than the memory sub-array size (if unrolling the 4D kernels into 2D arrays in convolutional layers), therefore, partial sum going through ADC needs to be accumulated from multiple sub-arrays. Here we assume practical array sizes from 64×64 up to 256×256, with three kinds of memory cell precision (1-bit/cell, 2-bit/cell and 4-bit/cell), and sweep the ADC precision from 3-bit to 5-bit (with nonlinear quantization), to find the optimal design option considering the trade-offs between inference accuracy and hardware overhead. Fig. 4 shows with 1-bit cell precision, 4-bit ADC is sufficient to guarantee ~89% accuracy for 64×64 and 128×128 synaptic array, while 5-bit ADC is necessary to avoid significant accuracy loss in multi-bit cell precisions. Fig. 5 shows the trade-offs when increasing the array size. Increasing array size results in smaller chip area but worse throughput and energy efficiency, due to large column currents and parasitic loading capacitance. The radar plot shows that the design based on 128×128 array size with 5-bit ADC achieves relatively balanced trade-offs among accuracy, energy efficiency, throughput, area and memory utilization. Fig. 6 shows the impact of ADC and memory cell precisions on hardware performance. Higher ADC precision is detrimental to the area and energy efficiency, while higher memory cell precision is beneficial because the peripheral circuitry could be saved.

C. Benchmark Across Device Technologies

Table 1 shows the benchmarking results across state-of-the-art device technologies, where the sequential and parallel read-out SRAM-based accelerators are evaluated at both 7nm and 32nm, and the parallel read-out NVM-based accelerators at 32nm. The reason for choosing 32nm for NVMs is because state-of-the-art RRAM is at 22nm [11], PCM is at 40nm [12], and FeFET is at 28nm [3]. Consider the read-noise and on/off ratio, 4-bit/cell is assumed for [1, 2, 3, 4]. The benchmark results show that, large on-state resistance R_{on} is the key factor to achieve better hardware performance. To avoid large voltage drop, the transistors in 1T1R or peripheral mux have to be sized up for small R_{on}, yielding significant area overhead. As a result, it takes longer time to activate the synaptic arrays (due to the increased capacitance loading), adversely increasing latency and lowering throughput. Thus, the conventional RRAM [11] or PCM [2] with a couple kΩ to tens of kΩ is not competitive, even with multi-bit per cell. Overall, the "analog" synaptic device based designs with large R_{on} (>100kΩ) (e.g. interface-engineered analog RRAM [1] or three-terminal FeFET [3] and ECRAM [4]) at 32nm could achieve superior energy efficiency (in TOPS/W) than parallel SRAM-based design at 7nm, plus the benefits of non-volatility for instant-on applications.

IV. Framework Performance

To explore the framework's performance to large-scale system, we extend a FeFET [3] based inference engine benchmarking with a deeper DNN, i.e. ResNet-18 for ImageNet, comparing the results and run-time for different simulation methods. The real-traced simulation is the default method in the framework (all the traces are transferred and accessed hierarchically). In pseudo-traced simulation, the traces are only accessed once to generate the activity parameters of weight and activations (percentage of non-zero values in traces) for each layer, which are passed hierarchically as inputs instead of the large traces (to save simulation run-time). Similarly, but without trace accessing, the average simulation only passes user-assumed activity parameters (e.g. 50%). Fig. 7 shows that with reasonable run-time, the real-traced framework achieves most accurate results, while the other two methods underestimate the performance (due to inaccurate column-current estimation).

V. Conclusion

In this work, we develop an end-to-end framework to benchmark CIM-based inference engine, which integrates NeuroSim with Pytorch and Tensorflow. With introduced device retention model and ADC quantization effects, it is efficient to investigate the trade-offs among inference accuracy, energy efficiency, throughput, area and memory utilization. With parallel read-out scheme and large R_{on}, the "analog" synaptic device based accelerators show promises. An improved version of DNN+NeuroSim with on-chip training capability is under development for future release.

Acknowledgment

This work is supported by ASCENT, one of the SRC/DARPA JUMP centers, NSF/SRC E2CDA program, and NSF-CCF-1903951.

Reference

[1] W. Wu et al. VLSI 2018.P. [2] W. Kim et al. VLSI 2019. [3] K. Ni et al. IEDM 2018. [4] J. Tang et al. IEDM 2018. [5] P.-Y. Chen et al. IEDM 2017. [6] K. Simonyan et al. ICLR 2015. [7] K. He et al. CVPR 2015. [8] S. Wu et al. ICLR 2018. [9] P.-Y. Chen et al. IRPS 2018. [10] X. Peng et al. ISCAS 2019. [11] Jain et al. ISSCC 2019. [12] J.Y. Wu et al. IEDM 2018.

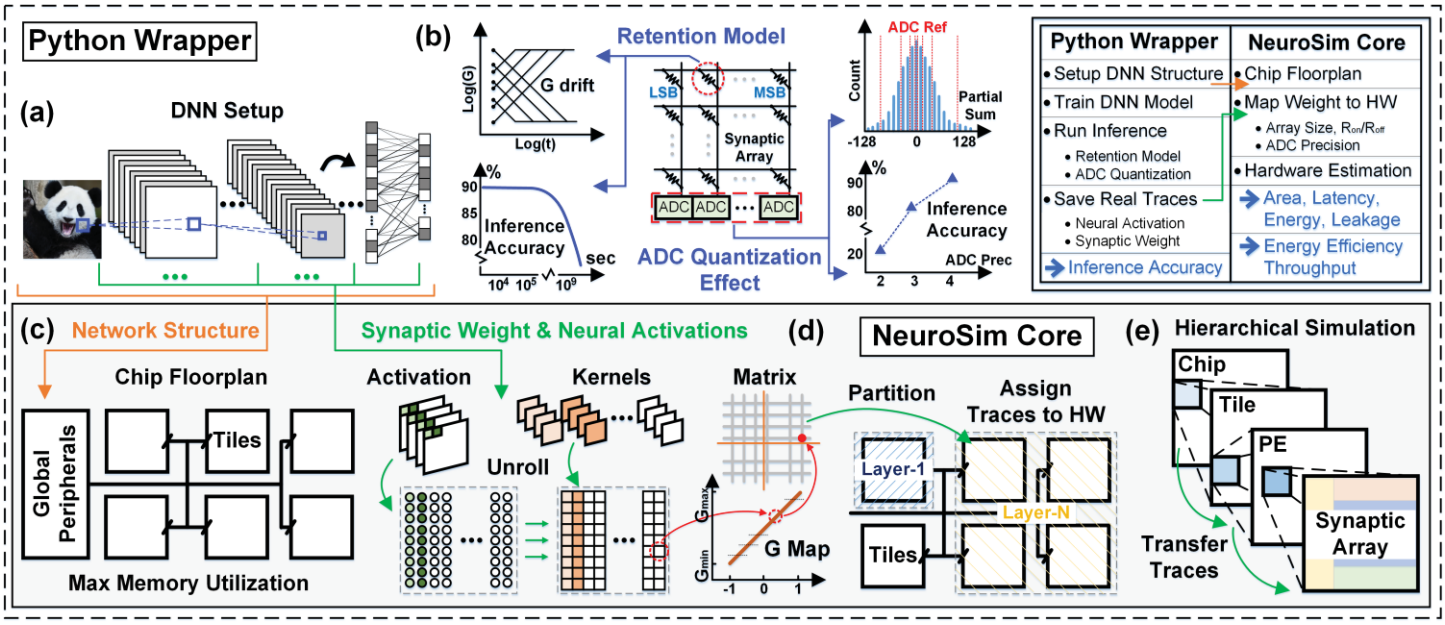


Fig. 1. Framework structure of DNN+NeuroSim. (a) DNN setup in python wrapper, software training with hardware constraints such as weight precision and partial sum quantization; (b) introduction of retention model and ADC quantization effects to inference accuracy; (c) pre-defined network structure is loaded as input to NeuroSim core, for automatic floor planning which weight-duplication to maximize memory utilization; (d) loading real trace (synaptic weights and neural activations) into NeuroSim, mapping data to conductance and digital voltage input cycles, which are to be partitioned and assigned to different locations of the CIM system; (e) hierarchical simulation from chip to tile, and from processing element (PE) to synaptic array.

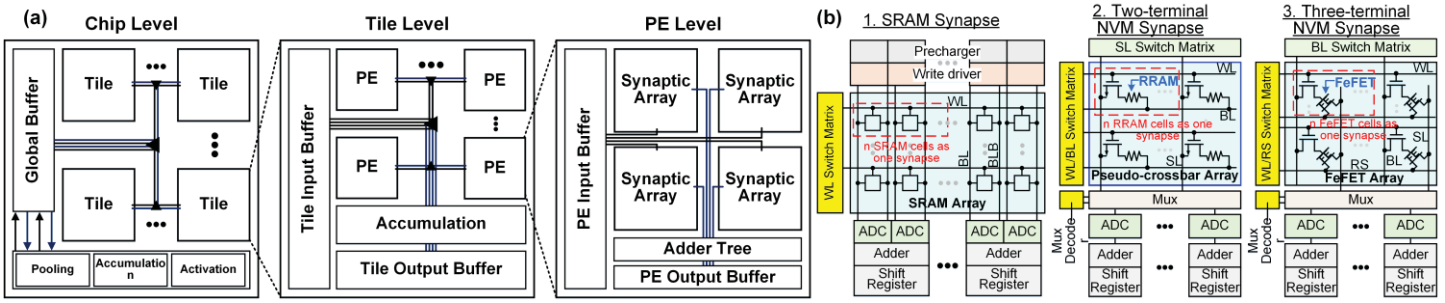


Fig. 2 (a) Architecture structure defined in the simulator, the top level of chip contains tiles, global buffer and neural-functional peripheries (including pooling, accumulation and activations). Inside a tile, it is further portioned into multiple processing elements (PEs), while each PE consists of several synaptic arrays, along with adder trees and local buffers. H-tree routing is used for interconnect. (b) Parallel read-out synaptic arrays based on SRAM, two-terminal NVMs (RRAM, PCM and STT-MRAM), and three-terminal NVMs (FeFET and ECRAM). Sequential (row-by-row) read-out modes also available. The circuit modules are all calibrated by SPICE simulations across technology nodes with PTM model.

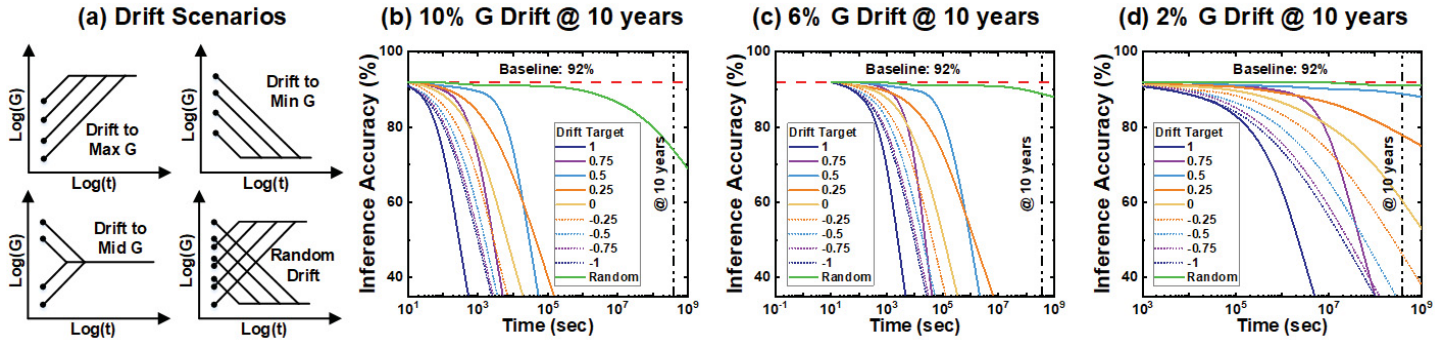


Fig. 3. (a) Different scenarios of conductance drift. Inference accuracy as a function of time for equivalent conductance drift by (b) 10%; (c) 6%; (d) 2% at 10 years, with different conductance uni-directional drifting targets (maximum 1 or minimum -1 or other intermediate states) or random drift of each weights.

VGG-8 (8-bit activation; 8-bit weight) on CIFAR10, with Novel Weight Mapping and Dataflow [9]									
Technology node (LSTP)	7 nm		32 nm						
Device	SRAM		SRAM		RRAM (Intel) [11]	TaOx/HfOx (TsingHua) [1]	GST PCM (IBM) [2]	HZO FeFET (NotreDame) [3]	ECRAM (IBM) [4]
ADC precision	Sequential	4-bit	Sequential	4-bit	5-bit	5-bit	5-bit	5-bit	5-bit
Cell Precision	1-bit		1-bit		2-bit	4-bit	4-bit	4-bit	4-bit
Ron (Ω)	\	\	\	\	6k	100k	40k	500k	500M
On/Off Ratio	\	\	\	\	17	10	12.5	100	40
Inference Accuracy (%)	92%		92%				91%		
Area (mm^2)	4.65	**4.28**	97.83	87.47	86.07	**20.45**	22.63	**19.71**	**19.71**
Memory Utilization (%)	99.29%	99.29%	99.29%	99.29%	98.69%	97.05%	97.05%	97.05%	97.05%
L-by-L Latency (ms)	0.85	0.15	1.61	0.33	19.92	1.16	2.65	0.38	0.28
L-by-L DynamicEnergy (uJ)	13.25	10.63	162.79	76.82	285.96	32.17	38.41	27.69	28.57
L-by-L Leakage power (mW)	104.85	101.69	1.41	1.33	0.22	0.11	0.11	0.11	0.11
Energy Efficiency (TOPS/W)	3.95	**14.95**	3.70	7.92	2.10	**18.97**	15.76	**22.17**	**21.51**
Throughput (FPS)	1171.15	**6875.94**	619.53	3001.13	50.16	**859.75**	378.03	**2617.24**	**3623.67**

Table 1. Benchmark results of DNN accelerators on VGG-8 for CIFAR10, based on SRAM (both sequential and parallel read-out at 7nm and 32nm), and reported "analog" synaptic devices (assumed at 32nm technology). **Green bold** values shows the devices with good performance.

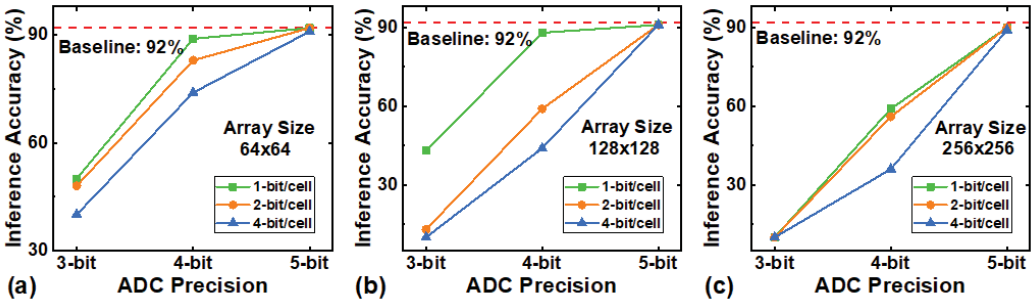


Fig. 4. Inference accuracy for VGG-8 for CIFAR10 as a function of ADC precision with different memory cell precision, at array size of (a) 64×64; (b) 128×128 and (c) 256×256, based on interface-engineered TaO_x/HfO_x RRAM [1]. 5-bit ADC is necessary for multi-bit per cell to maintain accuracy.

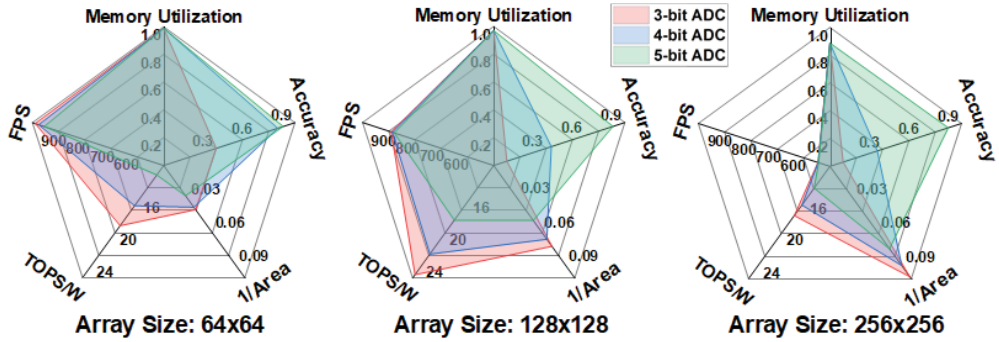


Fig. 5. Comparison of inference accuracy, memory utilization, area, energy efficiency and throughput, across different synaptic array sizes with 4-bit cell precision for VGG-8 for CIFAR10, based on interface-engineered TaO_x/HfO_x RRAM [1]. 128×128 array size with 5-bit ADC is chosen as a balanced design option.

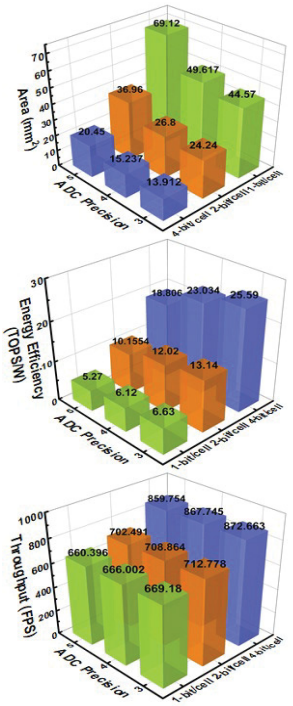
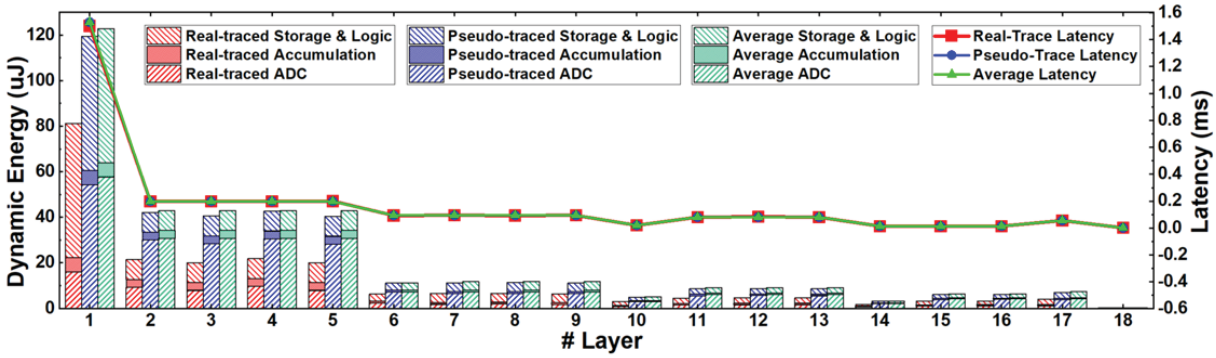


Fig. 6. Impact of ADC precision and cell precision on area and hardware performance, with 128×128 array size based on interface-engineered TaO_x/HfO_x RRAM [1].



	Real-traced	Pseudo-traced	Average
Area (mm^2)	20.08		
TOPS/W	**11.6**	6.14	5.93
FPS	**330**	326	326
Run Time (s)	**460**	86	2

* Storage & Logic: buffers, digital logic modules (e.g. decoder, switch matrix, mux), interconnect
* Accumulation: adders, shift+adders, adder trees, accumulation units

Fig. 7. Benchmark results of FeFET-based [3] DNN accelerators on ResNet-18 for ImageNet, with three different estimation methods (real trace, pseudo-trace and average). **Green bold** values shows real-traced method achieves more accurate estimation as other two methods underestimate the CIM performance, though real-traced method runs slower in a workstation (Intel Xeon Gold-6136 24-core 3.0GHz with 256GB DDR4).