Analog-to-Digital Converter Design Exploration for Compute-in-Memory Accelerators

Hongwu Jiang, Wantong Li, Shanshi Huang
Georgia Institute of Technology

Stefan Cosemans, Francky Catthoor
IMEC

Shimeng Yu
Georgia Institute of Technology

***Abstract*—Compute-in-memory (CIM) is a promising solution to efficiently implement multiply-and-accumulate (MAC) operations involved in deep neural network (DNN) computations. However, the mixed-signal compute scheme for CIM paradigm faces grand challenges in the necessary analog-to-digital converter (ADC) at the array outputs, which introduces quantization loss for inference accuracy and suffers from large area/energy overhead in peripheral circuitry. In this article, we comprehensively explore conventional ADC designs (Flash vs. SAR) for CIM application. Then we investigate a new data conversion scheme that performs the analog shift-add for multiple weight significance bits, namely analog shift-add ADC. Impact of the ADC precision on inference accuracy performance is thoroughly analyzed and illustrated for the representative CIFAR-10 dataset based on a multi-bit VGG-8 network. The evaluation results show that the analog shift-add ADC can tolerate up to 7-bit quantization loss without accuracy degradation. We benchmark the hardware performance of CIM arrays with various ADC designs at 40nm given similar area constraint, and the results show that the analog shift-add ADC achieves 37× and 4.9× lower energy-delay-product (EDP), compared to state-of-the-art Flash-ADC and SAR-ADC, respectively**

■ **Deep** neural network (DNN) has been enabling various artificial intelligence applications, from image classification to speech recognition. From the hardware's perspective, a variety of application-specific integrated circuit (ASIC) designs have been proposed to accelerate the multiply-and-accumulate (MAC) operations. However, the traditional von Neumann architecture inherently limits the parallelism of DNN algorithms since massive data movement happens between the computing units and the storage units, which results in low energy efficiency. To address this memory bottleneck, researchers have proposed the compute-in-memory (CIM) paradigm whereby the computation is performed inside the memory array [1]. The crossbar-like memory array is usually employed in CIM to store the values of the weight matrix [1], where the weights are mapped as the conductance of the memory cells. The MAC operations can be performed in parallel: the input voltage activates multiple rows and the products between the inputs and the weights are summed up along the columns as current output. In most of today's CIM designs, memory array is equipped with analog-to-digital converter (ADC) to convert analog MAC values to digital outputs, which can be passed to the peripheral circuitry for further processing steps such as activation function/pooling, and then being sent to the next array as the input. This mixed-signal compute scheme offers the scalability towards multiple-array designs via interconnect buses or network-on-chip, while introducing significant power dissipation and area overhead in the necessary data conversion at the array outputs. For example, the ISAAC architecture [2] reports that ADCs contribute up to 58% of the total power and 31% of the total area. From the algorithm's standpoint, despite binary neural network such as XNOR-Net [3] may greatly reduce the required memory capacity and eliminate ADCs, multi-bit precision is a more generic setting for large-scale DNNs for complex datasets to avoid inference accuracy degradation. To reduce the overhead of ADCs, there are two straightforward approaches. One is to limit the number of ADCs employed in one array, which means each ADC is shared by multiple columns. As a penalty, the parallel computing throughput is reduced. The other solution is to lower the ADC precision. However, the quantization loss of partial sum may hamper the inference accuracy performance. Therefore, under the hardware constraints, the choice of ADC topologies and configurations is critical to the design of the CIM architecture.

In this article, we explore the trade-offs involving different types of ADCs and investigate a new ADC design [4] especially suited for the CIM context, which can perform analog shift-add to alleviate the overhead. The impact of quantization loss for different ADC topologies is also analyzed. For the hardware evaluation, we comprehensively compare the ADC-only and array-level performance between the analog shift-add ADC and the conventional ADC designs that are integrated with a CIM array.

BACKGROUND AND RELATED WORK

CIM is a promising solution to perform extensive MAC operations, which is suitable for a weight stationary dataflow in DNN acceleration as it combines memory access and computation. Fig. 1 (a) shows a conceptual resistor-based crossbar array when mapping weight matrix for MAC operations. Memory cell conductance G_{ij} represents the weight W_{ij}. During the computation, the neuron activations X_i, are converted to voltages V_{xi} and applied to the crossbar rows in parallel. The summed current through the column I_i, could be converted to an analog voltage by a resistor divider (or by a diode-connected transistor in practice). Transimpedance amplifier (TIA) is a more advanced approach to realize current-to-voltage conversion. The analog voltage is then quantized by ADC. In the following section, we briefly introduce recent progresses of versatile CIM designs and a generic dataflow in CIM.

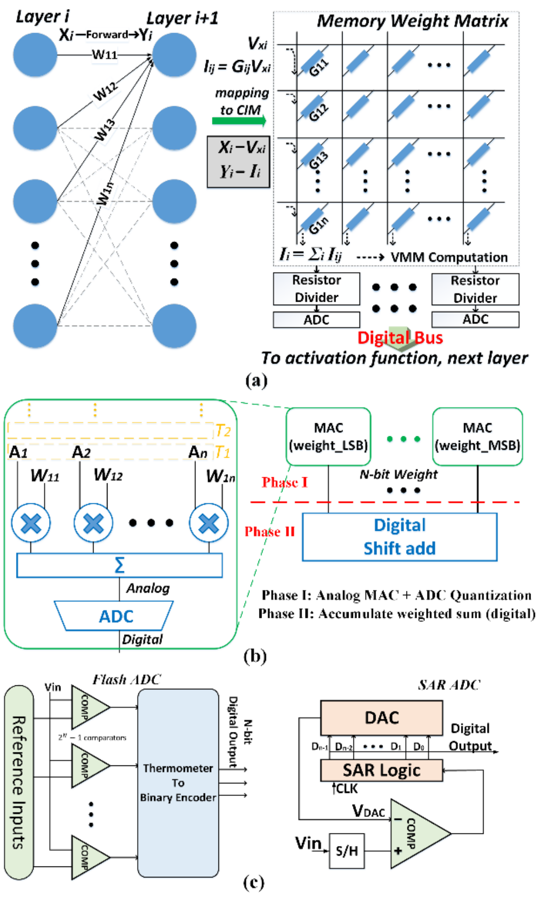

Figure 1. (a) Weight matrix mapping in crossbar array. (b) Generic dataflow of multi-bit MAC operations. (c) Principle of Flash-ADC and SAR-ADC

Memory technologies employed in CIM

Theoretically, the CIM architecture could be implemented by any memory device technologies. SRAM is considered as a mature candidate for CIM from the technology availability point of view. The general idea is to modify the SRAM bit-cell and periphery to enable the parallel access. Sense amplifier (SA) is usually replaced by ADC to produce quantized output. Multi-bit inference [4] and on-device training [5] have been demonstrated in with SRAM based CIM macros. Meanwhile, CIM architectures based on emerging non-volatile memories (eNVMs) have also been proposed. For instance, resistive random-access memory (RRAM) [6] provide attractive solutions due to multilevel states and higher density at the same technology node. Recently, the ferroelectric field effect transistor (FeFET) [7] is also regarded as a promising candidate for low power platforms, with intrinsically low write energy because its switching is field-driven rather than current-driven, and low read energy due to elevated on-state resistance (with proper gate biasing). However, in industrial prototypes [8], binary states are used and multi-level operations are still premature at scaled FeFET. Therefore, in this work, we evaluate the overhead of different ADC designs on the CIM architecture consisting of binary FeFET cells. The evaluation methodology could be extended to multi-level cells in principle.

Generic dataflow in CIM architecture

Fig. 1 (b) shows the conventional dataflow of multi-bit MAC operations in CIM. Input vectors are fed into CIM array cycle by cycle. Each cycle contains one significant bit. Multiple binary memory cells in multiple columns are used to represent a fixed-point weight in a binary format. Therefore, two separate shift-add processes are needed for CIM array: One is to weigh and sum up partial sums according to different significant input bits; the other one is to weigh and sum up partial sums according to different significant weight bits. The shift-add process for input is identical for different ADC topologies, thus it is skipped in our study. As shown in the diagram, the shift-add process for weight contains two phases: analog MAC and ADC in a single column (Phase I) and digital shift-add between columns (Phase II). Phase II is required to accumulate the weighted sum from the Least Significant Bit (LSB) and the Most Significant Bit (MSB). The CIM periphery usually employs digital shift-add module to process such multi-bit MAC operations.

ADC TOPOLOGY DESIGN EXPLORATION

A compact ADC design plays an important role in the area/energy efficiency of the CIM array. Especially when considering the column pitch of eNVM-based array is much less than that of the SRAM, it is imperative to explore possible ADC topology options.

Conventional ADC choices for CIM application

The popular ADC topologies in prior CIM works are Flash-ADC [9] and successive-approximation-register (SAR)-ADC [6] due to their simplicity and suitability for low to medium precision (i.e. <8 bit). Flash-ADC is made of cascading comparators. For an N-bit converter, the circuit employs $2^N - 1$ comparators. The thermometer code generated by comparators is then encoded to the digital output code. Flash-ADC is the fastest ADC design in principle but consumes exponentially larger power and area when the precision increases. SAR-ADC employs a single comparator but performs one-bit comparison only in one internal clock. Based on binary search algorithm, the SAR logic (implemented with multi-stage shift registers) will adjust the references dynamically and does the comparison in a bit-by-bit fashion. This sequence continues all the way from MSB to LSB. Once all bits are done, the conversion is complete and the N-bit digital output is available in the register. Typically, a capacitive digital-to-analog converter (DAC) array that exploits the charge redistribution is used to generate the analog reference voltage. Generally Flash-ADC has better performance for lower precision (3-bit or below) while SAR-ADC performs better for relatively higher precision. In the recent CIM macros [10], 3-bit Flash-ADC was used for small-scale arrays with 64~128 rows. However, the scalability of Flash-ADC towards large-scale arrays remains unexplored.

Analog Shift-add ADC

As aforementioned, there is a tight area constraint for ADC in CIM array, which means multiple columns have to share one ADC. Consequently, the throughput will be reduced due to the time-multiplexing. Besides, additional circuits such as multiplexer (MUX) and digital shift-add are also needed. To reduce the ADC overhead, an analog shift-add approach was demonstrated in [5]. As shown in Fig. 2 (a), modification of analog shift-add ADC is applied to where the shift-add is performed. Here the shift-add process for weight precision is moved prior to the ADC and being conducted in the analog domain, which will weigh and sum up the analog MAC outputs from each

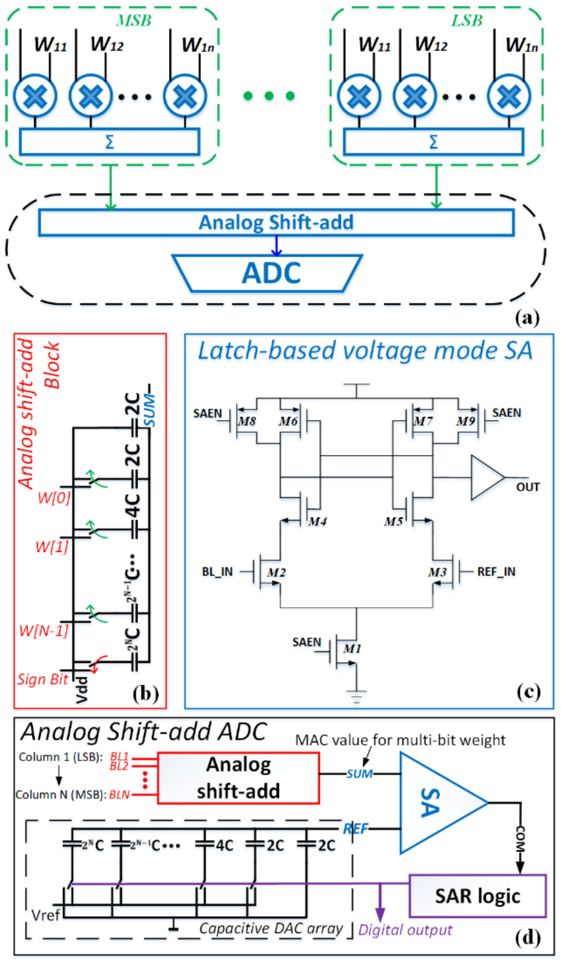

Figure 2. (a) Alternative CIM dataflow which performs weighted sum before ADC. (b) Schematic of analog shift-add block. (c) Schematic of latched-based sense amplifier. (d) Top-level structures of analog shift-add ADC.

column in multiple internal clock cycles. Then the pre-shifted and added MAC value will go through the regular SAR-ADC to generate the final output that already contains the weight significance. This design effectively eliminates the digital shift-add module for multi-bit weight computation and reduce the MUXs of multiple columns. So it can improve throughput and energy efficiency under the same area constraint but it has a potential precision impact. Both aspects will be studied in more details in the next section. Fig. 2 (b) shows the circuit schematic of the analog shift-add block. We exploit the charge redistribution nature of the capacitor array to perform the weighted accumulation before ADC quantization. This block is connected to multiple columns that represent the

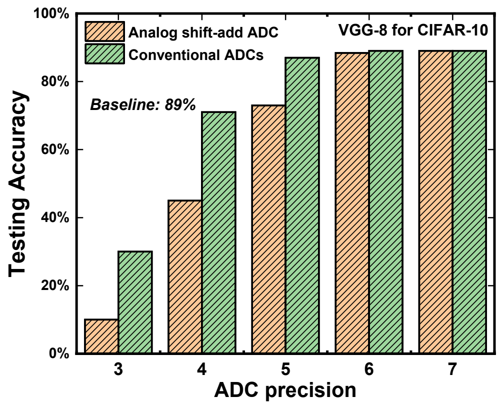

Figure 3. Accuracy performance vs. precision for different ADC designs.

different weight significance. As the weight value could be either positive or negative in DNN, 2's complement code is used to store the weight where the first bit is the sign bit. The capacitors involved in this block from top to bottom exponentially increase in capacitance, which represent from LSB to MSB. The switches enable the corresponding capacitor to be coupled to the power supply or the analog output from the corresponding column. All the switches are synchronously switched. When the voltage level of the sign-bit switch equals to zero, the voltage level of all other switches equals to one. Once the charge redistribution is stabilized, the analog shifted and added MAC value can be directly read out from the SUM port for the regular SAR-ADC process. The comparator is based on a simple latch-based sense amplifier, as shown in Fig. 2 (c). It comprises M4-M7 transistor as strong positive feedback amplifier and M8-M9 are used to pre-charge the output node and M2-M3 can be treated as common source differential amplifier. Tail transistor M1 is used to enable the sensing process. Such a latch-based sense amplifier is commonly used in embedded memory design due to its advantages of low power and high speed. The top-level diagram of the analog shift-add ADC is shown in Fig. 2. (d). To our best knowledge, such analog shift-add ADC design was not thoroughly evaluated in prior works, especially lacking of exploring the impact on software accuracy performance and hardware overhead compared to conventional ADCs at array-level. Array-level hardware performance will be quite different from the ADC-only result where the rest of array is ignored since different ADC topologies may also cause diversities on other periphery circuits. In this article, we comprehensively explore the ADC designs above for CIM application.

ADC PRECISION IMPACT ON TESTING ACCURACY

To determine the minimally required ADC precision, we investigate the impact of ADC quantization loss on inference accuracy performance. As a case study, we build a VGG-8 model for CIFAR-10 dataset. There are two algorithmic techniques to quantize the partial sum from the CIM array: One is to reduce full range of the partial sum. Since the partial sum distribution is typically concentrated near the center [10], appropriately reducing range in the two tails will not introduce significant accuracy loss. The other is the nonlinear quantization to statistically gather similar counts of partial sums in each quantization step [10]. Fig. 3 shows the accuracy performance vs. ADC precision for the conventional ADC designs (representing both Flash and SAR) and the proposed analog shift-add ADC. Here we group Flash-ADC and SAR-ADC together since they have the same dataflow in which ADCs first quantize partial sums and then digital shift-add modules are employed to accumulate digitized partial sums. Oppositely, the analog shift-add ADCs first weigh and sum up the analog multi-bit MAC outputs and then quantize the final output that already contains the weight significance. We embedded these two different quantization approaches in software simulation and investigated the impact of ADC quantization loss on inference accuracy performance. The data precision setting in the algorithm is 4-bit weight, 8-bit activation, 8-bit gradient and 8-bit error, following the WAGE method [11]. The software baseline accuracy is 89%.

We assume 512×512 memory array is used which means the full precision of each column's partial sum is 9-bit. For conventional ADCs, we can see that 6-bit precision could obtain the same accuracy as the baseline which means 3-bit quantization loss is tolerable. For analog shift-add ADC, the input full precision for partial sum is 13-bit (9-bit from column partial sum and 4-bit from shift-add for weight precision). From the plot, we can find that 6-bit analog shift-add ADC could also maintain baseline accuracy, tolerating 7-bit quantization loss. Analog shift-add ADC could tolerate more quantization loss because quantization loss only happens at the final stage for analog shift-add ADC. On the contrary, conventional ADCs have quantization loss on each partial sum and then accumulate these quantized partial sums by shift-add, which means errors will be accumulated. In sum, the analog shift-add before the ADC could preserve more information, while the digital shift-add after the ADC may lose some residual information, resulting in more errors on the final output. In our evaluation environment, using ADCs with more than 6-bit resolution is an overkill, which will introduce more hardware penalties with no accuracy improvement, as shown in Fig. 3. On the other hand, the accuracy loss for 3-5 bits is non-trivial under our configuration. Other quantized low-precision networks (e.g. XNOR-Net [3]) or using smaller sub-array size (e.g. 128×128 instead of 512×512) can maintain accuracy performance even under lower ADC precision 3-5 bits [12]. Thus, we made the ADC comparison from 3-bit to 6-bit in this work.

EVALUATION

In this section, we evaluate the hardware performance of the three ADC topologies based on SPICE simulation. As a case study, we use FeFET as memory device for the CIM array, as already motivated in the introduction section. The reason to select FeFET is because its elevated channel resistance (~500kΩ [7]) could potentially support a large array size as 512×512. The FeFET bit cell size (4F by 4F) is assumed considering a relaxed channel length as high voltage is required for programming. The technology node used is 40 nm low power due to our foundry PDK availability. The ADC designs and evaluation results presented in this work generally are applicable to other charge-based CIM scheme in other technology nodes and other memory devices.

ADC-only Performance Benchmarking

Firstly, we compare the hardware performance of a single ADC only. To have a fair comparison, ADC performance should be evaluated under the same number of operations. The analog shift-add ADC is designed to convert multi-bit MAC results during one conversion process. As described in the previous section, traditional dataflow in CIM could only perform non-weighted MAC at one time, corresponding to 1-bit input × 1-bit weight. As shown in Fig. 4. (a), a MUX is needed to switch from different significant weight bits. Therefore, the hardware performance for Flash-ADC and SAR-ADC should be evaluated during 4× single ADC conversion time (as 4-bit weight is used in our case study). Fig. 4 (b) shows the hardware performance comparison including power, latency, energy and EDP among the three ADC designs. We can find that Flash-

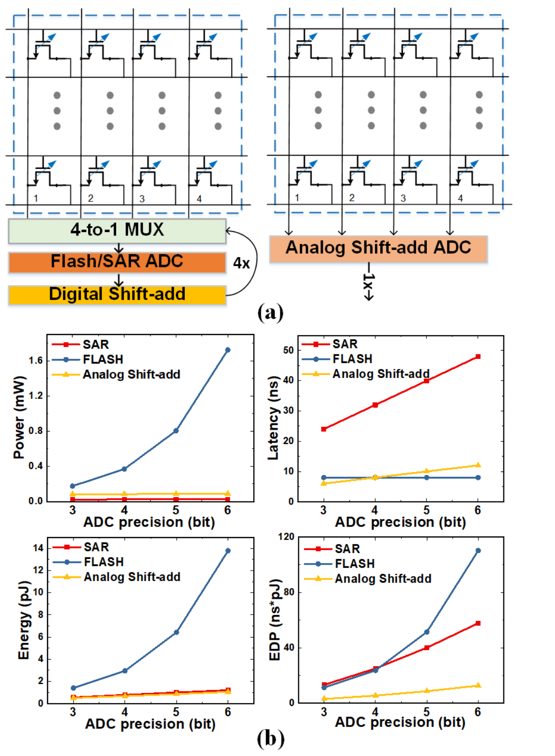

Figure 4. (a) Periphery configuration for different ADC design. (b) ADC performance comparison.

ADC consumes the most power which exponentially increases with ADC precision, while SAR-ADC and analog shift-add ADC have power dissipation nearly irrespective to ADC precision. Obviously, Flash- ADC is faster than SAR-ADC. One interesting observation is that analog shift-add ADC could complete multi-bit MAC in one conversion, thus its processing speed is even comparable to Flash-ADC. Since the conversion mechanism of this analog shift-add ADC is the same as SAR-ADC, the latency will linearly increase with ADC precision but with much smaller slope as shown in the latency plot. For the energy consumption, we can see an exponential growth in Flash-ADC. Finally, we evaluate energy-delay product (EDP), a well-defined metric that reflects the balance between energy consumption and throughput performance. We can see that analog shift-add ADC has the best performance across 3-6 bits for completing the same number of MAC operations. However, this is only a module-level benchmarking for ADC comparison regardless of CIM array-level constraints.

Array-level Performance Comparison

To systematically compare ADC performance for CIM application in a fair way, we need to consider the area constraint at the array-level. Under a similar area constraint (e.g., 2× of memory array area, ~13,422 μm^2 in our case study), we need to consider how many columns should share one ADC and employ appropriate MUX to switch columns for the time multiplexing. For Flash-ADC and SAR-ADC, the cost of digital shift-add should also be included for multi-bit MAC operations. Table 1 shows array-level ADC configuration under area constraint. For SAR-ADC and analog shift-add ADC, since the area of single ADC is relatively small and only increases slightly as ADC precision increases. The column sharing factor maintains 16 from 3-bit to 6-bit ADC precision. For Flash-ADC, as the area is exponential to ADC precision, we have to reduce the number of ADCs to satisfy area constraint, which means more column sharing when the precision increases. We can find that even with larger column sharing factor, the total area for Flash-ADC is still much larger than that of SAR- and analog shift-add ADC. Array-level performance of ADC-related periphery is shown in Fig. 5. Fig. 5. (a)

Table 1. ADC configuration under area constraint

SAR ADC(bit)	ADC unit area(μm^2)	No. of columns share one ADC	ADC total area(μm^2)
3	99.52	16	4207.04
4	127.48	16	5357.12
5	163.60	16	5235.20
6	214.76	16	6872.32
FLASH ADC(bit)	**ADC unit area(μm^2)**	**No. of columns share one ADC**	**ADC total area(μm^2)**
3	144.12	8	9223.68
4	318.40	16	10188.80
5	668.96	32	10703.36
6	1375.19	64	11001.51
Analog Shift-add ADC(bit)	**ADC unit area(μm^2)**	**No. of columns share one ADC**	**ADC total area(μm^2)**
3	107.52	16	3440.64
4	143.48	16	4591.36
5	195.60	16	6259.20
6	278.76	16	8920.32

Area constraint: < 13421 μm^2

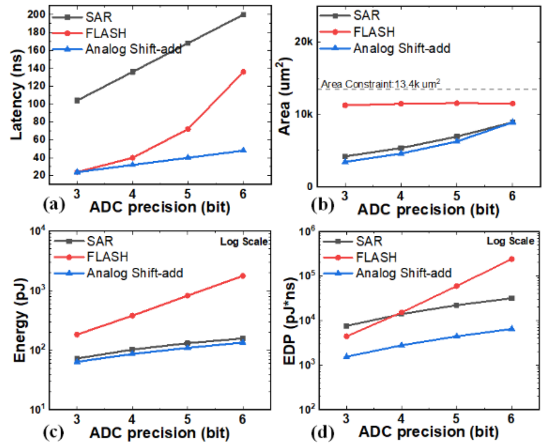

Figure 5. Array-level performance comparisons with possible ADC designs. (a) Latency vs precision. (b) Area vs precision (c) Energy vs precision (d) EDP vs precision.

represents total latency of ADC processing time including digital shift-add processing time. For example, for 6-bit Flash-ADC, 64 columns share one ADC thereby sacrificing the throughput by 8 times compared to 3-bit case (8 columns share one) if considering ADC only. However, the latency of digital shift-add for weight significance will remain the same as ADC precision increases, which slows down the increase of total latency, because the number of shift-add cycles is only related to weight precision. This is the reason why total latency (including shift-add latency) increases 7 times from 3-bit to 6-bit, instead of 8 times. The plot shows that analog shift-add ADC needs the smallest latency to finish all the operations for the entire array since it can convert a multi-bit MAC operation in one cycle and less total number of cycles are caused by a relatively small column sharing factor.

Fig. 5 (b) shows the substantial area overhead of Flash-ADC but it is flattened with higher precision due to the increasing column sharing factor. With the same column sharing, the area of the other two ADCs slightly increases, but still it is much smaller than that of Flash-ADC, even at 6-bit. Fig. 5 (c) and (d) show that the analog shift-add ADC has the lowest energy and EDP across the targeted precision range from 3-bit to 6-bit. From the previous results on ADC precision impact, we need 6-bit ADC to maintain the inference accuracy for a 512×512 array size. Thanks to the improved dataflow that performs multi-bit MAC operations before ADC, analog shift-add ADC is a promising solution for CIM application.

CONCLUSION

In this article, we comprehensively investigated analog shift-add ADC design for CIM array. The quantization loss on inference accuracy for different ADC designs is also explored. The array-level performance with possible ADC schemes are systematically evaluated, providing understandings of the tradeoff between hardware performance and area overhead. In this work, 6-bit ADC precision is sufficient for no accuracy degradation for a large array (512×512). At 6-bit, the analog shift-add ADC scheme achieves 37× and 4.9× higher EDP, compared to Flash-ADC and SAR-ADC, respectively. The area footprint of analog shift-add ADC is comparable to that of SAR-ADC and only 0.77× that of Flash-ADC.

ACKNOWLEDGMENT

This work was in part supported by NSF-CCF-1903951, ASCENT, one of the SRC/DARPA JUMP centers.

■ REFERENCES

1. S. Yu, “Neuro-inspired computing with emerging nonvolatile memory,” *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260-285, 2018.
2. A. Shafiee, et al., “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” *IEEE International Symposium on Computer Architecture (ISCA),* pp. 14-26, 2016.
3. M. Rastegari, et al., “XNOR-Net: ImageNet classification using binary convolutional neural networks,” *European Conference on Computer Vision (ECCV)*, pp. 525-542, 2016.
4. X. Si, et al., “A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning,” *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 396-398, 2019.
5. J.-W. Su, et al., “A 28nm 64Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips,” *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 240-242, 2020.
6. C.X. Xue, et al., “A 1Mb multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for CNN based AI edge processors,” *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 388-390, 2019.
7. K. Ni, et al., "In-memory computing primitive for sensor data fusion in 28 nm HKMG FeFET technology," *IEEE International Electron Devices Meeting (IEDM)*, pp. 16-1, 2018.
8. S. Dunkel et al., “A FeFET based super-low-power ultra-fast embedded NVM technology for 22 nm FDSOI and beyond,” *IEEE International Electron Devices Meeting (IEDM)*, pp. 19-7, 2017.
9. W.H. Chen, et al., "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," *IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 494-496, 2018.
10. R. Liu, et al., "Parallelizing SRAM arrays with customized bit-cell for binary neural networks," *ACM/ /IEEE Design Automation Conference (DAC)*, pp. 1-6, 2018.
11. S. Wu, et al., "Training and inference with integers in deep neural networks," *International Conference on Learning Representations (ICLR)*, 2018.
12. X. Peng, et al., " DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," *IEEE International Electron Devices Meeting (IEDM)*, pp. 32-5, 2019.

Hongwu Jiang is currently working towards the PhD degree in electrical and computer engineering at the Georgia Institute of Technology in Atlanta, Georgia. His research interests include SRAM-/eNVM- based hardware architecture and accelerator design of deep learning. He received the M.S. degree in electrical engineering from Arizona State University in 2014. He is a Student Member of the IEEE.

Wantong Li is currently pursuing the Ph.D. degree in electrical and computer engineering at Georgia Institute of Technology. He received the M.S. degree in electrical engineering from Columbia University in 2016. His research interests include low-power, secure, and fault-tolerant in-memory computing systems.

Shanshi Huang is currently pursuing the Ph.D. degree in electrical and computer engineering at the Georgia Institute of Technology in Atlanta, Georgia. Her current research interests include Deep learning algorithm & hardware co-design and deep learning security. She received the M.S. degree in electrical engineering from Arizona State University in 2014.

Stefan Cosesman (IEEE SM'18) is a Principal Member of Technical Staff at imec, Belgium. His current focus is on circuits and memory devices for AI accelerators, with emphasis on analog in-memory computing. He received his Ph.D. degree in 2009 from KU Leuven for his work on low power SRAM circuits.

Francky Catthoor has been active in research on architectural methodologies; co-exploration of application, computer architecture and deep-submicron technology aspects; wireless, IoT and biomedical systems; and renewable energy systems, all at IMEC. He is IMEC senior fellow, part-time full professor at KULeuven and IEEE Fellow.

Shimeng Yu is currently an associate professor of electrical and computer engineering at Georgia Institute of Technology. He received the Ph.D. degree from Stanford University in 2013. His research interests are emerging non-volatile memories for in-memory computing. He is a senior member of the IEEE