2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS) | 978-1-6654-1913-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/AICAS51828.2021.9458576

Trends in Analog and Digital Intensive Compute-in-SRAM Designs

Rishabh Sehgal
Dept. of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78751
Email: sehgal.rish@utexas.edu

Jaydeep P. Kulkarni
Dept. of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78751
Email: jaydeep@austin.utexas.edu

***Abstract*—The unprecedented growth in Deep Neural Networks (DNN) model size has resulted into a massive amount of data movement from off-chip memory to on-chip processing cores in modern Machine Learning (ML) accelerators. Compute-In-Memory (CIM) designs performing DNN computations within memory arrays are being explored to mitigate this 'Memory Wall' bottleneck of latency and energy overheads. Among the incumbent embedded memories, the Static Access Random Memory (SRAM) built using high performance logic transistors and interconnects can enable custom CIM designs while offering low pJ/bit access energy, high-endurance, high-performance, and high- bandwidth. The wordline and bitline voltages and pulse-widths are modulated to realize analog or digital domain multiply-and-accumulate (MAC) computations using multiple SRAM bitcell variants. This paper describes the trends in recent CIM-SRAM designs utilizing such analog and digitally-intensive approaches. In an analog CIM-SRAM design,the inputs/activations are transformed into analog voltage or pulsewidth and applied on wordlines and/or bitlines. Multi-bit MAC computations often involve peripheral data converter circuits which need to be optimized significantly to minimize the area and energy overheads. On the other hand, digitally-intensive CIM-SRAM approaches try to avoid analog circuits by implementing smaller bit-width wordline/bitline computations and utilize sense amplifiers for performing basic logic operations and/or employ small digital logic block next to the SRAM column I/O circuits forming compute-in/near SRAM designs. Key design trends in both the approaches and qualitative comparisons are presented with a perspective on future CIM-SRAM designs.**

I. Introduction and Motivation

Deep learning has become ubiquitous in the modern world and has been hugely successful in performing tasks like computer vision, image recognition, object detection, machine translation and speech recognition[1]-[3]. The share of the machine learning computations is on a steep climb inside the datacenters of the major companies. One major limiter of deep learning in the present scenario is its computation intensity, requiring very high-performance computing resources and long training times. These compute applications increasingly require a large number of external memory accesses which in turn requires a high bandwidth (BW) wireline interface from the compute core to the outside world, which may include several DRAM chips and ASICs[4]-[5].The recent surge in custom-designed HW for AI applications has further exacerbated the BW and memory-access latency problems of the traditional 2D Von-Neumann chips. While computation is a huge bottleneck for the Convolutional (CNN) layers, custom designed inference processors have shown promising results on the Deep Learning computations as compared to CPUs and GPUs. A strategic shift from the traditional resource and power hungry Von-Neumann architecture-based machines is required. Employing "Compute in Memory" (CiM) is a promising step which would effectively save these smart IOTs a large number of memory accesses thereby reducing their energy costs considerably.

A whole new class of smart embedded devices in the mobile form-factor have been in resurgence and these devices form a significant portion of consumer electronics these days (Smartphones, IOTs, smart drones, smart cars, factory automation robots etc). It is very critical to have these computations and memory resources available in these edge devices because most of the times these devices are resource and battery limited and computation on the cloud is deemed unreliable by the fact that it faces key issues such as high latency, limited bandwidth, security and privacy while transferring user's personal data to the cloud. Furthermore, in the conventional Von-Neumann systems, the inference processor regularly polls for the required weight and input data from DRAM or lower level caches causing considerable energy and latency overhead and achieving very less locality.

II. SRAM-Based Compute-In-Memory Systems

The architectures integrating some forms of memory and computation started emerging as early as 1970s. These architectures aimed at reducing the amount of incurred energy and throughput cost based on the unnecessary data movements that was required in a traditional Von-Neumann machine. All of these qualifying architectures use the SRAM memory arrays with a dual purpose, one as a storage bank for weights of the neural network's filters and second as compute engine to calculate MAC or Multiply and average(MAV) of the stored weights with the provided inputs. In the very initial task, these approaches more or less differ in the way of providing input feature vector to the SRAM compute macros and hence a detailed comparative discussion on the same follows in the section IV. Furthermore, the next important task being MAC/MAV operation itself, there are varied memory architectures, bitcells and approaches used by different research groups for

Paper Reference (SRAM type)	Analog based CIM	Bitwidth Inputs/Weights	Inputs applied to:	DAC used:	ADC used and type:	SRAM data storage format	All Digital
[7] In-Mem Compute (6T)	Yes	5b/1b (+1,-1)	WL voltage	Yes	Comparator SA	Column Major	No
[8] DIMA (6T)	Yes	8b/8b	WL (PWM)	No	Yes (8b)	Column Major	No
[9] Algorithm-dependent CIM (6T)	Yes	1b/1b (+1,-1 wts and activations)	WL (digital); WLL, WLR	No	Multi-Level Sense-Amp (3b)	Column Major	No
[10] Twin-8T SRAM (8T)	Yes	2b/5b (4b I/P possible)	WL (Even Odd Dual-Channel)	No	Yes; named C2PU	Column Major	No
[11] BNN CIM (6T)	Yes	1b/1b	WL (digital)	No	Sense-Amp Based	Column Major	No
[12] Two-Way-Transpose SRAM (6T)	Yes	2b,4b,8b/4,8b	WL (digital)	No	SOGE-SA	Column Major	No
[13] CONV-SRAM (10T)	Yes	6b/1b	Bitline (V_a)	Yes	Serial-Integrating ADC;	Row Major	No
[14] 7nm TSMC CIM (8T)	Yes	4b/4b	Wordline (Bin. weighted comp. caps)	No	4b Flash ADC;	Column Major	No
[15] Thinker-IM RNN-CIM (6T)	No	1b/1b (XNOR)	Wordline (digital)	No	3b VSA (Var. Reference);	Column Major	Yes
[16] CSRAM bit-serial (8T)	No	(Arbitrary upto 32b/32b)	Wordline (digital)	No	No (Digital peripheral ckts);	Column Major	Yes
[17] CASH-RAM (8T)	Yes	5b/2b; ternary wts(+1/0/-1)	Bitline (V_a)	Yes	Yes;	Row Major	No
[18] 1-to-8bit configurable CIM (6T)	Yes	Both Configurable from 1b to 8b;	Serial-Inps WL (digital)	No	Yes (Self-Reference Multi-level Reader);	Column Major	No
[19] Charge-domain CIM MOM caps (6T)	Yes	1b to 1b;	WL (digital)	No	Yes (SAR-like);	Column Major	No

TABLE I: Survey of Compute In Memory Architectures

optimizing the energy and throughput requirements of the heavy CNN computations involved and also keeping the feasibility and technology scaling of the macros in mind.

Difference also lies in the number of weight bits accommodated by the CIM macro. A detailed discussion and comparative analysis are presented in the section V for the same. Lastly, the partial sums calculated in charge domain and/or voltage domain are resolved back to digital values using innovative ADC architectures which use the locality and types of the computations involved to their advantage. There are also efforts to directly calculate post activation function processed outputs which is usually done by innovative circuit techniques applied in conjunction with sense-amplifiers and similar column-based circuitry. There is also a class of CIM macros which performs the functions related to input application and output partial sums resolution in an all-digital fashion by using an assortment of bit-serial computation techniques. The review of this third part is provided in detail in the section VI of this paper.

III. Feeding in the Input F-Maps/Activations

Most of the SRAM CiM systems are weight-stationary i.e. the kernel/filter values are stored in the SRAM arrays. Subsequently to perform convolutions the quantized input pixel values (Input Feature Maps or IF-Maps) are fed into the SRAM array using different techniques based on the underlying macro architecture. These techniques are discussed in details as follows:

A. Wordline based IF-Maps

One of the earliest approaches was taken by Zhang et al[7] was to drive the WLs of the 6T SRAM bitcell array with analog voltages representing the feature values xi, leading to the corresponding bit-cell currents IBC,I. These currents representing the IF-Maps values were thought of as multiplying with the weight stored (+1 or -1). The resulting aggregated discharge from all bitcells in a column was resolved by a comparator providing sign thresholding. In another notable approach called deep in-memory architecture (DIMA)[8], multiple rows of the standard 6T SRAM array were accessed using pulsewidth modulated (PWM) WL signals and which processed the resulting bitline (BL) voltage drops ΔV_{BL} via column pitch-matched low-swing analog circuitry in the mid-logic area. Multi-bit precision (8-b) for both weights and inputs for the above-mentioned algorithms and upto 53x (Energy Delay product) EDP reduction in measured results was demonstrated. In both of the above techniques multi-row WLs were asserted per bitline precharge as the weights were stored in a column major format. Another notable paper using the WL based IF-Maps inputting is [9] where binary DNN with 0/1-neuron and +/-1weight are implemented. Since both weights and inputs are quantized and constrained to 1-bit values, the authors do not employ WL DACs or WL PWM circuitry. Si et al in Twin-8T SRAM CIM paper[10] also apply the input (IN) to the read-WL (RWL) which supports binary (0V and Vwll3) or 2b-(4 levels: 0V, Vwll1, Vwll2 and Vwll3). This is used to generate the weighted cell current Imc. Kim et al [11] also use the WLs to provide the 1bit (2levels: Vddl and GND) input value and also trigger multiple WLs per precharge to compute the product in terms of VBL. Su et al [12] present a paper called Two-Way Transpose 6T SRAM which provides the digital 2 bit input to the forward versions of the WLs namely FWLM=IN[1] and FWLL=IN[0]. While [17] again uses a DAC to convert the inputs to a V_a to be applied on WLs, another approach [18] sends inputs in a bit serial fashion

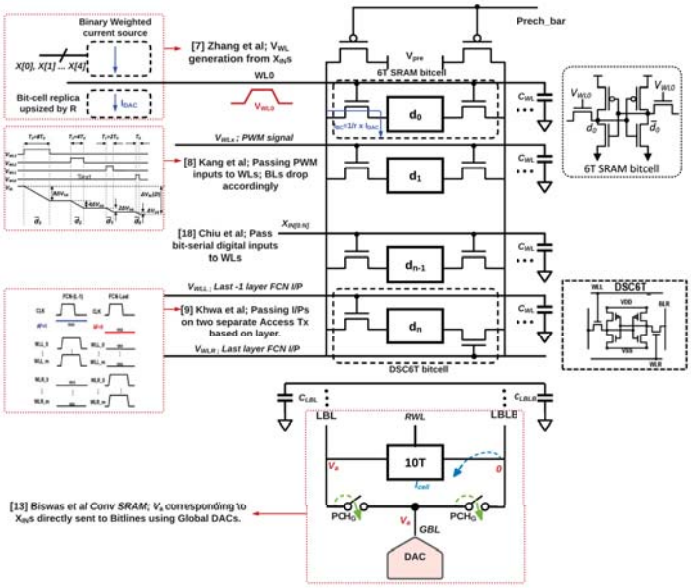

Fig. 1: Input Feeding techniques (to Wordlines or Bitlines)

to a block of multiple WLs containing the corresponding weights.

B. Bitline based IF-Maps

Another technique to provide input activations to the SRAM macros emerged in the subsequent papers such as CONV-SRAM where Biswas et al [13] feed the IFMP values (Xins) into columnwise DACs, which convert the digital XN codes to analog input voltages on the global read bit lines (GBRLs). These GBRLs are shared by all the local arrays and they provided the analog input voltage to the local bitlines of the subarrays. This implements the fact that in CNN's each input is shared/processed in parallel by multiple filters. The authors introduce the terms multiply and average(MAV) where-in they compute the MAV of the dot products and a scaling factor of M_k is applied after computing the entire dot product. Their system allows 8bit signed inputs (1b sign) to be converted into analog voltages which are then multiplied by the corresponding 1bit filter weights. This approach is more robust to the SRAM bit-cell Vt variations arising from their minimum sized geometry. Unlike this technique the WL based IFMPs relied on discharge current through SRAM cells Icell which has a significant spread from its mean values (σ = 30%μ). Now if I_{cell} is used to modulate the analog voltage V_a on the bitline, there is a wide variation in the V_a and it cannot be controlled very well. But in this approach Biswas et al send the Va directly to the bitlines using global DACs and since the global DACs can be upsized, the variation due to them is significantly less than that compared to the min sized bitcell.

C. Digital to Analog Converters to provide inputs values

Most of the papers discussed in the previous two sections utilize Digital to Analog converter blocks (DACs) to convert the multibit digital input values to corresponding analog voltages V_a. Zhang et al[7] used a DAC formed from binary-weighted pMOS current sources. The resulting current I_{DAC} is converted into an output WL voltage by passing it through an upsized replica of a bit-cell which contains a diode connected access transistor device and this current is also mirrored by the bit-cell. This V_{WL} leads to the resulting bitcell currents which discharge the BLs through the pull down transistors of bit cells. Kang et al[8] rely on generating pulse width modulation values on the wordlines. An on-chip pulse generator is used to generate the binary weighted pulses which in turn discharges the BL capacitance C_{BL} depending on the stored bit. Khwa et al [9] do not require explicit DACs because the binary neuron activation values 0/1 are passed directly as WL 0/1 respectively for 6T SRAM design and in 8T design complementary WLs with digital input activations are used. Bit-Serial inputs in [18] also means no DACs are required and [10] also employed digital WL input values. Conv-SRAM [13] uses a DAC to convert the digital input code to analog voltage which is based on time to digital and digital to analog voltage generation. Cascode pMOS stack biased to act as a constant current source and is based on duration of the ON pulse (modulated based on digital input code) that is provided to this circuitry. [17] also use a DAC to convert inputs to a V_a value to be applied at the bitline.

IV. Parallel Multiply and Accumulate/Average Operations in CiM Macros

Most SRAM arrays use either standard 6T bitcells or more robust larger-footprint bitcells such as 8T and 10T cells. In some CIM SRAM designs, 8T/10T cell designs are tuned to make the bitcells bi-directional and also very robust to read disturbs. Zhang[7] and Kang[8] employed standard 6T SRAM cell to realize a CIM-SRAM macro; Biswas[13] used a 10T bitcell with decoupled read and write ports to mitigate any read/write disturbs. Khwa et al[9] demonstrated a CIM-SRAM scheme with both 6T (Hybrid BNN) and 8T bitcells (XNOR-BNN). Su et al[12] utilized foundry provided compact 6T-SRAM bitcells which are reconfigured to be used as two-way transpose macro with multibit weight dot products capability. All of the above-mentioned architectures store the weights in a column major format except the Conv-RAM which stores them in a row major format. Si et al[10] adopted a Twin-8T SRAM macro using 2 decoupled-read 8T bitcells avoids the read disturb issue. The twin-cell are divided into a MSB bitcell and a LSB bitcell. In [16] Wang implements an all-digital bit-serial approach and used a special type of transposable 8T bitcell which can be accessed by both vertical Compute wordline and horizontal bitline. The computation is performed in both the horizontal bitlines and pitch-matched compute logic at the end of bitlines. This architecture also stores the data vectors in a column major format. In a different approach, Valavi[19] implements charge-domain CIM using backend capacitors. In this case, along with a 6T bitcell, a backend MOM capacitor is included which stores the output of XNOR computation between the weight stored on the bitcells and the inputs applied through a pair of PMOS transistors.

V. Analog to Digital Converters and Custom Sense Amplifiers for Analog CIM-SRAMs

As discussed in the previous section, many CIM-SRAM approaches use voltage or charge domain analog computing which necessitate special Analog to Digital Converter (ADC) architectures and specialized sense amplifiers based comparators to resolve the analog MAC or MAV output into a digital code. Earlier designs such as Zhang et al[7] relied on sense-amplifier comparators accommodating rail-to-rail inputs to sense and classify decisions based on bitline differential. On the other hand, DIMA[8] uses a separate 8-bit ADC with V_{res} = 1mV. This output is followed by a digital logic to realize thresholding operation.Conv-RAM[13] use a serial charge-sharing and integrating ADC which has Sense Amplifier based on a StrongARM latch architecture. 7nm CIM[14] from TSMC uses a 4 bit flash ADC with area efficient sense amplifier instead of analog comparator to save on area and reduce energy consumption. Sense Amplifiers were implemented in [11] to resolving the Vsum to its digital codes bit by bit while [15] uses a Serial-phase triple sensing controller to select the reference voltages for 3bit voltage sense amplifier (VSA) and provide control signals to it. [9] employs ADC-like 3 bit multi-level sense amplifiers whereas [12] use a specially designed block called small-offset gain-enhancement sense amplifier (SOGE-SA) to tolerate small read margin. SAR-like ADC is used in the paper implementing Charge-domain CIM using MOM caps[19] but a specialized Self-Reference Multi-level Reader block is used in [18] to resolve the psums into digital code.

VI. All-Digital and BitSerial CiM Architectures

Apart from the charge/current domain analog computation-based architectures, there are also a few all-digital and/or bit-serial architectures that have come to the forefront. One of the forerunners in this domain was Eckert et al's Neural Cache[20] paper which used bit-serial architecture to re-purpose the cache usually available in the cores to an efficient engine executing convolutions, fully connected layer operations and pooling in-cache. The paper relied heavily on bit-serial arithmetic and the final computation (and and nor) performed on the data stored in the activated WLs is performed in analog domain but sensed immediately via SAs and converted to digital bits. Some digital computation is performed by innovative peripheral circuits near the SRAM array. Aga et al[21] in-Compute Caches presented a precursor of the Neural Cache idea. Wang et al[16] presented a compute SRAM (CRAM) architecture which combined 8T transposable bit cell with vector based, bit-serial in-memory arithmetic to accommodate a wide range of bit-widths from single to 32 to 64 bits. This paper also implemented a complete set of operations types such as integer and FP addition, multiplication and division. It achieved 30GFLOPs on 32-bit operands and energy efficiency of 0.56TOPS/W for 8 bit multiplication at 0.6V and 114MHz.

VII. Perspectives and Conclusion

The CIM-SRAM architectures are evolving at a very fast rate since they are being actively explored for performing ML centric computations with higher throughput and improved energy efficiency. The CIM approach is promising for resource constrained applications to minimize the energy overhead due to data movements. The edge computing applications such as IOTs, smart-drones, smart-phones, healthcare sensors etc. have very stringent energy efficiency requirements. Going forward, analog based CIM-SRAMs would need to address the issues of non-ideality and energy and area overheads due to data converters and the inherent V_t variations in the bitcells which could affect the neural network accuracy. Bit-serial and all-digital architectures can be useful in resource constrained edge application where the on-board SRAM can be repurposed as a CIM engine and hence the computations could be performed using a small logic block in/near SRAM array. All-digital CIM approaches can avoid the power and area overheads introduced by the data converters in the analog CIM-SRAM designs. They can also be amenable to process technology scaling and low-voltage operation due to digital nature of computation. Digital-intensive bit-serial CIM designs also require little tweaking of the existing memory arrays and caches which could enable easy integration into memory compilers. However, bit serial approach can degrade the throughput of digital-intensive CIM-SRAMs due to sequential nature of the computation which would necessitate careful dataflow optimization. Furthermore, limited I/O bitwidth can affect the parallelism in digital CIM-SRAMs. Overall, an optimal design approach analog or digital or a combination of thereof can lead to impressive gains in energy efficient CIM-SRAMs for future ML accelerators.

References

[1] G. Hinton et al., *in IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[2] A. Krizhevsky et al., *in Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.
[3] Y. H. Chen et al., *in IEEE JSSC*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
[4] T.-J. Yang et al., "Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning," *IEEE CVPR*, July 2017.
[5] M. Horowitz, *2014 IEEE (ISSCC)*, pp. 10-14.
[6] J. Sim, et al., *in Proceeding of IEEE ISSCC*, pp. 264-265, 2016.
[7] J. Zhang et al., *in IEEE JSSC*, vol. 52, no. 4, pp. 915-924, April 2017.
[8] M. Kang, et al., *in IEEE JSSC*, vol. 53, no. 2, pp. 642-655, Feb. 2018,
[9] W. Khwa et al., *2018 IEEE ISSC)*, pp. 496-498
[10] X. Si et al., *2019 IEEE ISSCC*, 2019, pp. 396-398,
[11] J. Kim et al., *2019 Symposium on VLSI Circuits*, 2019, pp. C118-C119
[12] J. Su et al., *2020 IEEE ISSCC*, 2020, pp. 240-242
[13] A. Biswas et al., *in Proc. of ISSCC*, pp. 488-490, 2018.
[14] Q. Dong et al., *2020 IEEE ISSCC*, 2020, pp. 242-244.
[15] R. Guo et al., *2019 Symposium on VLSI Circuits*, 2019, pp. C120-C121.
[16] J. Wang et al., *in IEEE JSSC*, 2020, pp. 240-242.
[17] A. Agrawal et al., *in IEEE Journal on Emerging and Selected Topics in Circuits and Systems*,vol. 10, no. 3, pp. 295-305, Sept. 2020.
[18] Y. -C. Chiu et al., *IEEE JSSC*, vol. 55, no. 10, pp. 2790-2801, Oct. 2020
[19] H. Valavi et al., *in IEEE JSSC*, vol. 54, no. 6, pp. 1789-1799, June 2019,
[20] C. Eckert et al., *2018 ACM/IEEE 45th Annual ISCA*, 2018, pp. 383-396.
[21] S. Aga et al., *2017 IEEE Intl. Symposium on HPCA*, 2017, pp. 481-492.