31.1 Conv-RAM: An Energy-Efficient SRAM with Embedded Convolution Computation for Low-Power CNN-Based Machine Learning Applications

Avishek Biswas, Anantha P. Chandrakasan

Massachusetts Institute of Technology, Cambridge, MA

Convolutional neural networks (CNN) provide state-of-the-art results in a wide variety of machine learning (ML) applications, ranging from image classification to speech recognition. However, they are very computationally intensive and require huge amounts of storage. Recent work strived towards reducing the size of the CNNs: [1] proposes a binary-weight-network (BWN), where the filter weights (w_i's) are ±1 (with a common scaling factor per filter: α). This leads to a significant reduction in the amount of storage required for the w_i's, making it possible to store them entirely on-chip. However, in a conventional all-digital implementation [2, 3], reading the w_i's and the partial sums from the embedded SRAMs require a lot of data movement per computation, which is energy-hungry. To reduce data-movement, and associated energy, we present an SRAM-embedded convolution architecture (Fig. 31.1.1), which does not require reading the w_i's explicitly from the memory. Prior work on embedded ML classifiers have focused on 1b outputs [4] or a small number of output classes [5], both of which are not sufficient for CNNs. This work uses 7b inputs/outputs, which is sufficient to maintain good accuracy for most of the popular CNNs [1]. The convolution operation is implemented as voltage averaging (Fig. 31.1.1), since the w_i's are binary, while the averaging factor (1/N) implements the weight-coefficient α (with a new scaling factor, M, implemented off-chip).

Figure 31.1.2 shows the overall architecture of the 256×64 conv-SRAM (CSRAM) array. It is divided into 16 local arrays, each with 16 rows to reduce the area overhead of the ADCs and the local analog multiply-and-average (MAV_a) circuits. Each local array stores the binary weights (w_i's) in the 10T bit-cells (logic-0 for +1 and logic-1 for -1) for each individual 3D filter in a conv-layer. Hence, each local array has a dedicated ADC to compute its partial convolution output (Y_{OUT}). The input-feature-map values (X_{IN}) are fed into column-wise DACs (GBL_DAC), which pre-charge the global read bit-lines (GRBL) and the local bit-lines (LBL) to an analog voltage (V_a) that is proportional to the digital X_{IN} code. The GRBLs are shared by all of the local arrays, since in CNNs each input is shared/processed in parallel by multiple filters. Figure 31.1.3 shows the schematic of the proposed GBL_DAC circuit. It consists of a cascoded PMOS constant current source. The GRBL is charged with this current for a duration t_{ON}, which is directly proportional to the X_{IN} code. For better t_{ON} vs X_{IN} linearity there should only be one ON pulse for every code to avoid multiple charging phases. This is impossible to generate using signals with binary-weighted pulse-widths. Hence, we propose an implementation where the 3 MSBs of X_{IN} are used to select (using TD_{56}) the ON pulse-width for the first-half of charging (TD_{56} is high) and the 3 LSBs for the second-half (TD_{56} is low). An 8:1 mux with 8 timing signals is shared during both phases to reduce the area overhead and the signal routing. As such, it is possible to generate a single ON pulse for each X_{IN} code, as shown for codes 63 and 24 in Fig. 31.1.3. This DAC architecture has better mismatch and linearity than the binary-weighted PMOS charging DACs [4], since the same PMOS stack is used to charge GRBL for all input codes. Furthermore, the pulse-widths of the timing signals typically have less variation compared to those arising from PMOS V_t mismatch.

After the DAC pre-charge phase, the w_i's in a local array are evaluated locally by turning on a RWL, as shown in Fig. 31.1.4. One of the local bit-lines (LBLF or LBLT) will be discharged to ground depending on the stored w_i (0 or 1). This is done in parallel for all 16 local arrays. Next, the RWL's are turned off and the appropriate local bit-lines are shorted together horizontally to evaluate the average via the local MAV_a circuit. MAV_a passes the voltages of the LBLT and LBLF to the positive ($V_{p\text{-}AVG}$) and negative ($V_{n\text{-}AVG}$) voltage rails, depending on the sign of the input X_{IN} (EN_P is ON for $X_{IN}>0$, EN_N is ON for $X_{IN}<0$). The difference between $V_{p\text{-}AVG}$ and $V_{n\text{-}AVG}$ is fed to a charge-sharing based ADC (CSH_ADC) to get the digital value of the computation (Y_{OUT}). Algorithm simulations (Fig. 31.1.1) show that Y_{OUT} has a peak distribution around 0 and is typically limited to ±7, for a full-scale input of ±31. Hence, a serial integrating ADC architecture is more applicable than other area-intensive (e.g. SAR) or more power-hungry (e.g. flash) ADCs. A PMOS-input sense-amplifier (SA) is used to compare $V_{p\text{-}AVG}$ and $V_{n\text{-}AVG}$, and its output is fed to the ADC logic. The first comparison determines the sign of Y_{OUT}, then capacitive charge-sharing is used to integrate the lower of the 2 voltage rails with a reference local column that replicates the local bit-line capacitance. This process continues until the voltage of the rail being integrated exceeds the other one, at which point the SA output flips. This signals conversion completion and no further SA_EN pulses are generated for the SA. Figure 31.1.4 shows the waveforms for a typical operation cycle. To reduce the effect of SA offset on Y_{OUT} value, a multiplexer is used at the input of the SA to flip the inputs on alternate cycles.

The 256×64 CSRAM array is implemented in a 65nm LP-CMOS process. Figure 31.1.5 shows the measured GBL_DAC results, which is used in its 5b mode by setting the LSB of X_{IN} to 0. To estimate the DAC analog output voltage (V_a), V_{GRBL} for the 64 columns are compared to an external V_{ref} by column-wise SA's, used in the SRAM's global read circuit. For each X_{IN}, the V_{ref} at which more than 50% of the SA outputs flip is chosen as an average estimate of V_a. An initial one-time calibration is needed to set V_a = 1V for X_{IN} = 31 (max. input code). As seen in Fig. 31.1.5, there is good linearity in the DAC transfer function with DNL < 1LSB. Figure 31.1.5 also shows the overall system transfer function, consisting of the GBL_DAC, MAV_a and CSH_ADC circuits. For this experiment, same code is provided to all X_{IN}'s, all w_i's have the same value, and the Y_{OUT} outputs are observed. The measurement results show good linearity in the overall transfer function and low variation in the Y_{OUT} values: mainly because variation in BL capacitance (used for averaging and CSH_ADC) is much lower than transistor V_t variation. SA offset cancelation further helps to reduce Y_{OUT} variation. It can be also seen from Fig. 31.1.5 that the energy/ADC scales linearly with the output code, which is expected for an integrating ADC topology.

To demonstrate the functionality for a real CNN architecture, the MNIST hand-written digit recognition dataset is used with the LeNet-5 CNN. 100 test images are run through the 2 convolutional and 2 fully-connected layers (implemented by the CSRAM array). We achieve a classification error rate of 1% after the first 2 convolutional layers and 4% after all the 4 layers, which demonstrates the ability of the CSRAM architecture to compute convolutions. The distribution of Y_{OUT} in Fig. 31.1.6 for the first 2 computation-intensive convolutional layers (C1, C3) show that both layers have a mean of ~1LSB, justifying the use of a serial ADC topology. Figure 31.1.6 also shows the overall computational energy annotated with the different components. Layers C1 and C3 consume 4.23pJ and 3.56pJ per convolution, computing 25 and 50 MAV operations in each cycle respectively. Layer C3 achieves the best energy efficiency of 28.1TOPS/W compared to 11.8 for layer C1, since C1 uses only 6 of the 16 local arrays. Compared to prior digital accelerator implementations for MNIST, we achieve a >16× improvement in energy-efficiency, and a >60× higher FOM (energy-efficiency × throughput/SRAM size) due to the massively parallel in-memory analog computations. This demonstrates that the proposed SRAM-embedded architecture is capable of highly energy-efficient convolution computations that could enable low-power ubiquitous ML applications for a smart Internet-of-Everything.

Acknowledgements:
This project was funded by Intel Corporation. The authors thank Vivienne Sze and Hae-Seung Lee for their helpful technical discussions.

References:
[1] M. Rastegari, et al., "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks", *arXiv:1603.05279*, 2016, https://arxiv.org/abs/1603.05279.
[2] J. Sim, et al., "A 1.42TOPS/W Deep Convolutional Neural Network Recognition Processor for Intelligent IoE Systems", *ISSCC*, pp. 264-265, 2016.
[3] B. Moons, et al., "A 0.3–2.6 TOPS/W Precision-Scalable Processor for Real-Time Large-Scale ConvNets", *IEEE Symp. VLSI Circuits*, 2016.
[4] J. Zhang, et al., "A Machine-Learning Classifier Implemented in a Standard 6T SRAM Array", *IEEE Symp. VLSI Circuits*, 2016.
[5] M. Kang, et al, "A 481pJ/decision 3.4M decision/s Multifunctional Deep In-memory Inference Processor using Standard 6T SRAM Array", *arXiv:1610.07501*, 2016, https://arxiv.org/abs/1610.07501.

 978-1-5090-4940-0/18/$31.00

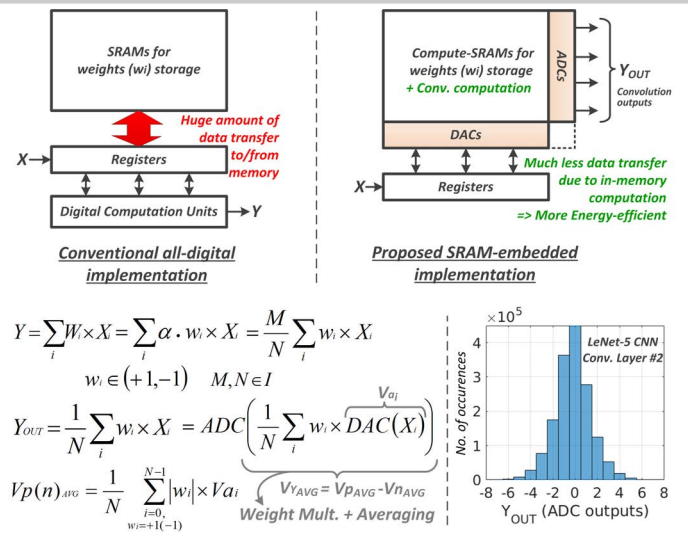


Figure 31.1.1: Concept of embedded convolution computation, performed by averaging in SRAM, for binary-weight convolutional neural networks.

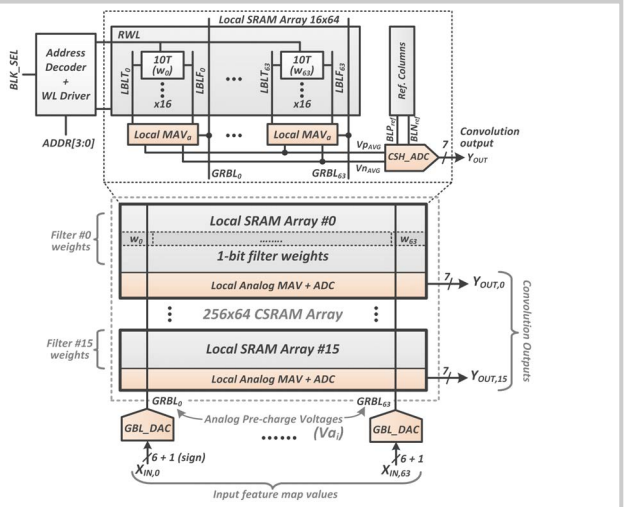


Figure 31.1.2: Overall architecture of the Convolution-SRAM (CSRAM) showing local arrays, column-wise DACs and row-wise ADCs to implement convolution as weighted averaging.

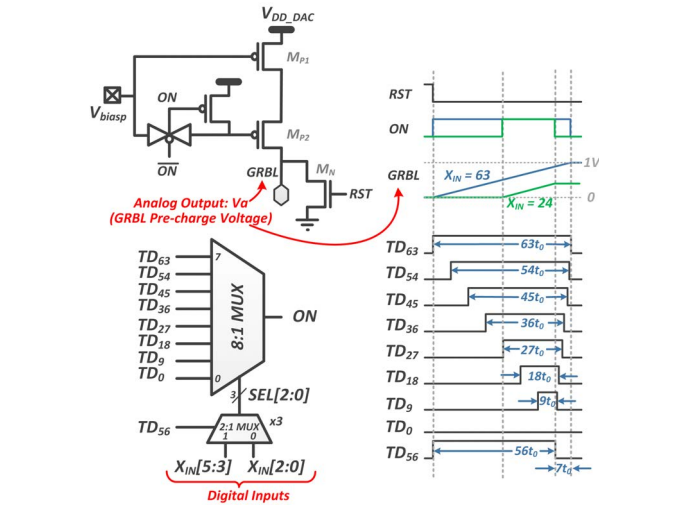


Figure 31.1.3: Schematic and timing diagram for the column-wise GBL_DAC, which converts the convolution digital input to an analog pre-charge voltage for the SRAM.

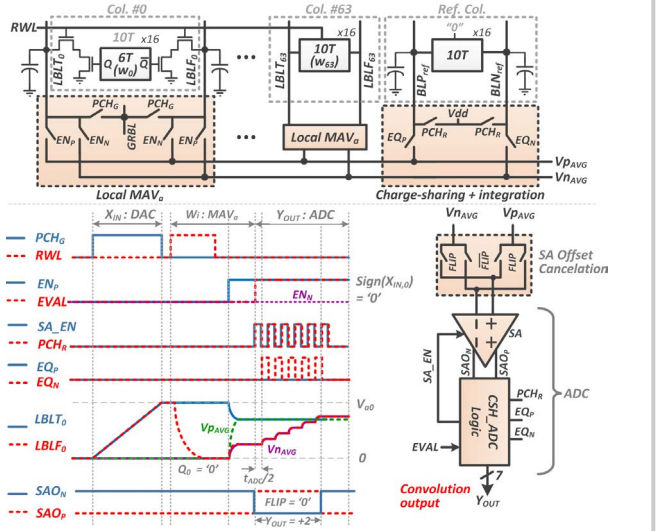


Figure 31.1.4: Architecture for the row-wise multiply-and-average (MAV_a) and CSH_ADC. Operational waveforms for convolution computation.

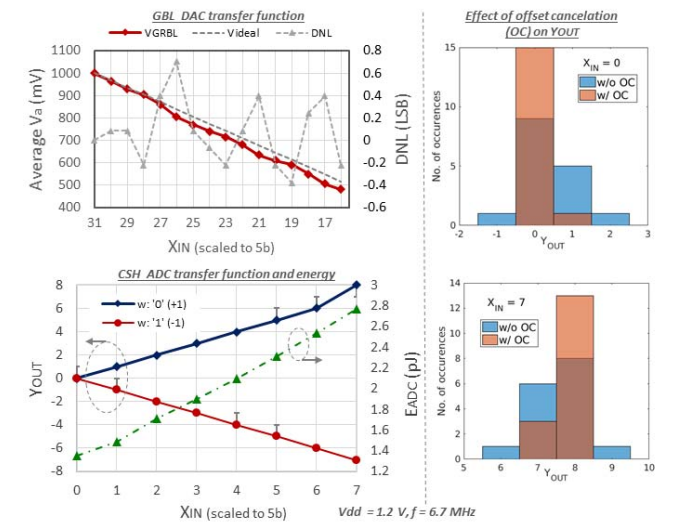
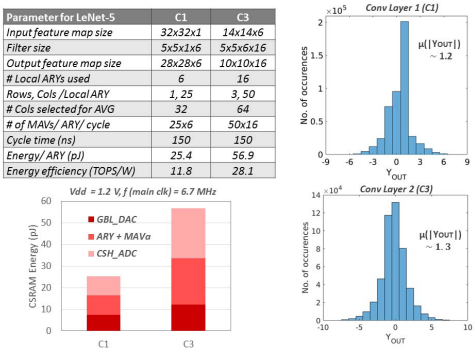


Figure 31.1.5: Measured performance of GBL_DAC and CSH_ADC in the CSRAM array. Also shown is the effect of the offset cancelation technique.

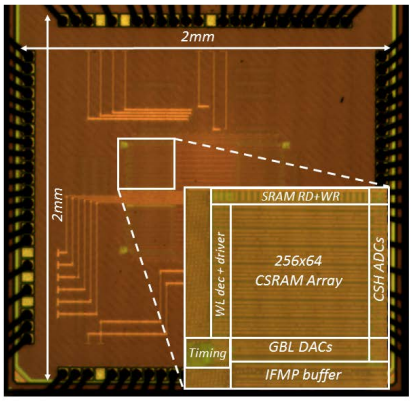
Parameter for LeNet-5	C1	C3
Input feature map size	32x32x1	14x14x6
Filter size	5x5x1x6	5x5x6x16
Output feature map size	28x28x6	10x10x16
# Local ARYs used	6	16
Rows, Cols /Local ARY	1, 25	3, 50
# Cols selected for AVG	32	64
# of MAVs/ ARY/ cycle	25x6	50x16
Cycle time (ns)	150	150
Energy/ ARY (pJ)	25.4	56.9
Energy efficiency (TOPS/W)	11.8	28.1



	Tech. (nm)	ML Algo., # classes	Comp. mode	Input/ Filter	Voltage (V)	SRAM size (KB)	Throughput (GOPS)	Energy eff. (TOPS/W)	FOM
This work	65	CNN, 10	Analog	7b/1b	1.2	2	10.7	28.1	150.3
ISSCC '16 [2]	65		Digital	16b	1.2	36	64	1.42	2.5
VLSI '16 [3]	40		Digital	6b/4b	0.8	144	102	1.75	1.2
Arxiv '16 [5]	65	kNN, 4	Analog	8b	1.0	16	10.2	0.98	0.6

1 MAV, MAC, SAD = 2 OPs. $FOM = \frac{Throughput \times Energy\ eff.}{SRAM\ size}$

Figure 31.1.6: Energy and output distribution measured results for the first two convolutional-layers (C1, C3) of the LeNet-5 CNN for the MNIST dataset. Table showing comparison to prior work.



Technology	65nm
CSRAM size	16 Kb
CSRAM Area	0.067 mm^2
Array Organization	256x64 (10T bitcell)
# of column DACs	64
# of row ADCs	16
Supply Voltage	1.2V/ 0.9V
Main clock frequency	6.7 MHz
ADC clock frequency	364 MHz
Max # of mult+avg per convolution	64
Energy/ convolution	3.6 pJ

Figure 31.1.7: Die micrograph and test-chip summary table.

978-1-5090-4940-0/18/$31.00 ©2018 IEEE