An SRAM-Based Hybrid Computation-in-Memory Macro Using Current-Reused Differential CCO

1st Daniel Giftson E
BTech, Electrical Engineering
IIT Gandhinagar
Gandhinagar, India
daniel.giftson@iitgn.ac.in

2nd Prateek Sharma
MTech, Microelectronics and VLSI Design, Electrical Engineering
IIT Gandhinagar
Gandhinagar, India
sharmaprateek@iitgn.ac.in

I. Introduction

As neural networks (NNs) are becoming widely used in AI applications, the demand for energy-efficient convolutional neural network (CNN) engines has also increased. The traditional CNN engines based on Von Neumann Architecture consume a massive amount of computational power to perform enormous multiply-and-accumulate (MAC) operations and also have relatively low energy efficiency due to the movement of data between the memory and the processor, known as the memory wall problem. This makes them undesirable to be used in edge devices. To overcome this memory wall problem, Computation In Memory (CIM) architecture proves to be a promising solution as it saves energy consumption by processing the computations in the memory itself without moving the data. Moreover, it can achieve high energy efficiency by processing parallel computations. There are two ways to implement this architecture: Static Random Access Memory (SRAM) or nonvolatile memory. Although nonvolatile memory-based CIM (NVM-CIM) can keep the weight data stored without any power and doesn't need to be reloaded when the system is turned on, it comes with a higher cost of write operation. In contrast, although volatile, SRAM-based CIM (SRAM-CIM), although volatile, can provide high energy efficiency and better performance during write operation. Furthermore, since it is compatible with state-of-the-art CMOS logic technology, it is easily scaled to provide better performance and high energy efficiency. Thus, SRAM-CIMs deliver one of the most promising solutions to the problems faced by traditional CNN engines in edge devices. Mostly SRAM-CIMs perform MAC operations in the analog domain as they can achieve high energy efficiency and throughput. However, it gets traded off by robustness due to the low signal-to-noise ratio (SNR) and higher area and energy consumption of the analog-to-digital converter (ADC). Therefore, the authors of this paper have proposed an SRAM-CIM based on hybrid computation using digital IMAC (DIMAC) and phase-domain near-memory-array computing (PNMAC). Bit-wise multiplications are computed using DIMAC, which mitigates the energy-efficiency problem by effective zero-skipping operation. Further, by using PNMAC, a lightweight ADC with a wide dynamic range is implemented that converts the analog data to digital data only once at the end of the MAC operation with ultra-low power consumption. Moreover, the Current-controlled oscillator (CCO), the core of PNMAC, which is an 11-stage ring oscillator, occupies less area, thus optimizing the area utilization.

II. Objective

The main objective of this paper is to implement a 4kb 8T-SRAM (64x64 SRAM array) computation-in-memory (CIM) Macro based on hybrid computation using digital in-memory-array computing (DIMAC) and phase-domain near-memory-array computing (PNMAC). The DIMAC takes care of computing bit-wise multiplication digitally with effective channel-wise zero-skipping with high energy efficiency and performance by employing multiple local dual-column arrays (LDCAs). Afterwards, the PNMAC takes care of the addition and accumulation of the partial MAC operation values in parallel with a high dynamic range by using current steering DAC-based differential current-controlled oscillator (DCCO). Finally, the accumulated phase information is converted to digital data using a phase quantizer. Therefore, the core idea is that by effectively reusing the steered current to accumulate the partial MAC results from the DIMAC, the PNMAC's power consumption can be reduced. In the remainder of the report, we will have a look at the background research done and the important results achieved by this paper.

III. Background

A. Computing Techniques for MAC Operation

Analog domain computing is a very energy-efficient choice. [1] uses current domain computing for the AdaBoost machine learning classifier for the 6T SRAM cell array. MAC values for 4b input and 1b weight are computed energy-efficiently. [2] Current domain computing using an 8T SRAM cell array is proposed. This design prevents write disturbances and provides improved precision. [3] uses charge domain computation using Metal-oxide-metal capacitors, achieving a high computational SNR, higher energy efficiency, higher performance, and robustness of output [4]. Analog domain computing provides energy-efficient MAC operation, but an ADC is needed to convert to a digital domain which consumes a huge amount of power and area. In [5], Phase domain

computation is implemented using a Gated Ring Oscillator (GRO) to accumulate the results. It continuously accumulates the MAC values, and at the end of the MAC operation, the phase value is obtained when a read logic arrives. In [6], CNN is implemented using time domain MAC computation. In [7], CCO-based ADC is used, which is less precise but has less latency. A wide-range and high-frequency generator is employed to improve the system performance, which consumes a huge current and needs a feed-forward compensation circuit for its operation. Time and Phase domain computing produces precise results with less area and energy consumption but has low throughput as inputs are fed serially in these MAC circuits. To overcome this issue, In [5], the MAC rate is increased to a high frequency, increasing power consumption. In [8], a Zero-skipping convolution SRAM is proposed for energy-efficient CIM operations. Energy consumption of CIM operations is further reduced by using a charge reuse scheme. Analog computing is more energy efficient as compared to digital domain computing. Digital domain computing is more immune to noise and process variation but is less energy efficient. These computing techniques have trade-offs among computation energy, dynamic range, read accuracy, and area efficiency. The current and charge domain computing techniques are highly energy efficient and have good performance, but the output of these computing techniques needs to be digitized using ADC. Phase domain computing has ultra-low-power computing along with a wide dynamic range and better area efficiency but has low performance because of its sequential operation. In this paper, the authors have used both phase and digital domain computing to curtail the power overhead of ADCs and improve the robustness of IMAC. This is done by: 1) Using the proposed PNMAC that can perform addition and accumulation parallelly in place of phase domain computing having a low performance. 2) Using bit-serial digital computing enhances the simplicity and energy efficiency of in-memory computation.

B. SRAM-Based Computing Structures

Near memory array computing (NMAC) structures use SRAM as memory, and either analog or digital computing can be employed for energy-efficient operations. It uses an SRAM macro along with computing blocks. [9] uses multi-row read access in the memory macro and capacitive-charge-sharing scheme for MAC operation. Its NMAC is robust to process-voltage-temperature variations. Although DAC is not needed because of its multi-row read scheme, a heavy ADC is needed to digitize the results. In Memory array computing (IMAC), structures read multiple data parallelly and operate by applying multi-bit data to the SRAM memory. IMACs use analog computing techniques to achieve high energy efficiency and throughput. In [10], an 8T1C CIM macro is proposed using capacitive-coupling computing (C3) performed in the memory array, giving high energy efficiency and performance by performing fully parallel vector-matrix multiplication but it needs flash ADCs which consume much power and have low resolution. IMACs have limited analog SM for multi-bit MAC operations. In [11], this SM problem is overcome by using a hybrid of in-memory and near-memory computing structures. In this, partial MAC operations are performed in analog, and the rest operations in the digital domain, so the ADC digitizes the partial MAC results with sufficient SM. Using a hybrid structure enables achieving high energy efficiency, flexibility, and programmability, but it still needs an ADC, which consumes huge power. Also, many analog-to-digital conversions are required for each partial MAC operation. To solve this problem, a CIM macro based on hybrid computation is proposed, which continuously accumulates the partial results in the phase domain, giving ultra-low-power and wide-dynamic-range analog to digital conversion.

IV. Proposed SRAM-Based Hybrid Computation Using DIMAC and PNMAC

Fig. 1 shows the entire proposed hybrid CIM architecture. It consists of two main parts: 1) DIMAC which performs in-memory bit-wise multiplication in a very energy-efficient manner and generates Pulse-Width-modulation (PWM) signals to control the PNMAC. 2) The partial MAC results generated by DIMAC are continuously accumulated by the PNMAC. It also converts the accumulated results into digital data only once at the end of MAC operation with ultra-low power consumption.

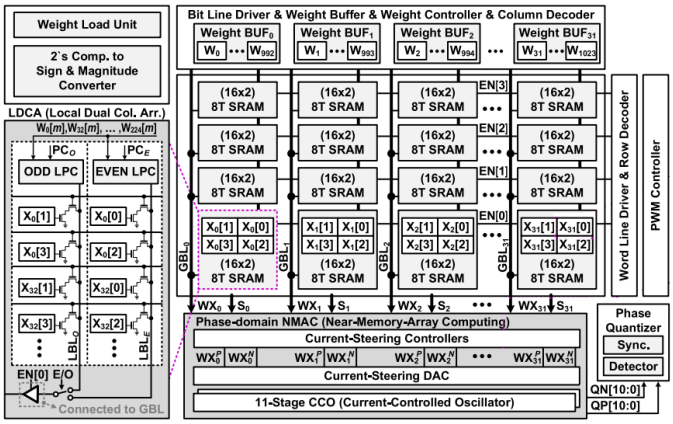

Fig. 1. Block diagram of the proposed hybrid CIM macro with digital in-memory-array computing (DIMAC) and phase-domain near-memory array computing (PNMAC)

V. Digital In-Memory-Array Computing

The proposed Digital In-Memory Array Computing (DIMAC) Architecture consists of a 64x64 SRAM Macro divided into 32 dual-columns of 4 Local dual-column arrays (LDCA) each. The weights are selected as an external source and represented in S&M format by S, W[2:0]. Each column has a global BL (GBL) which reads out the multiplication result from the local bit lines (LBLs) through a tri-state buffer enabled by EN[k]. LDCA which is of 32-bit capacity (16x2) generates the PWM signals of the multiplication values W[m]X[3:0]. Here, we have implemented the schematic of one Local dual-column

array and carried out the simulations on Cadence Virtuoso. Additionally, we have implemented all of the circuits proposed in this paper in 28nm CMOS technology with a supply voltage (VDD) of 900mV. We have kept the operation frequency of the CIM as 300MHz as proposed in the paper.

A. Schematics

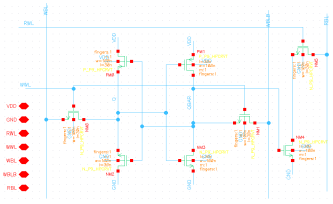

Fig. 2. 8T SRAM Schematic

Fig. 2 shows the schematic of an 8T SRAM. It carries out bit-wise multiplication by performing an AND operation, W[m]X[n], with the help of a local precharger (LPC). Here, we are using an active low precharge clock (PC) signal to precharge the LBL as we are using a PMOS transistor to pass the W[m] to LBL. Before the computation happens, the LBL is precharged to the weight bit, W[m]. Once the read wordline (RWL) is turned high, the AND operation takes place on the LBL. When W[m]=0, the LBL will always remain 0 irrespective of the value, X[n], stored in the 8T SRAM cell. When W[m]=1, the LBL will remain 1 if X[n]=1 as there is no discharging path for the LBL. But if X[n]=0, there is a discharging path formed for the LBL to discharge thus eventually going to 0.

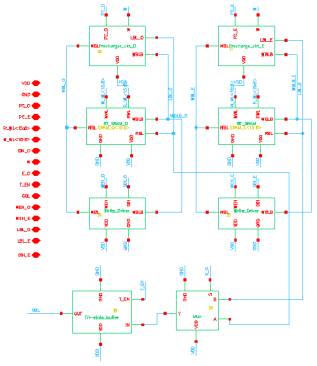

Fig. 3. Local Dual-Column Array Schematic

Fig. 3 shows the schematic of a 16x2 LDCA. This employs an overlapping precharging technique to improve the throughput of the DIMAC. For example, if LBL_E is driving the GBL, then the precharging and AND operation can be carried out on LBL_O just before the flipping of the E/O signal. This increases the throughput by hiding the precharging time in the PWM pulse.

B. Waveforms

Fig. 4 represents the timing diagram of the SRAM write operation for the input value (X[3:0]) 0101. In the first cycle,

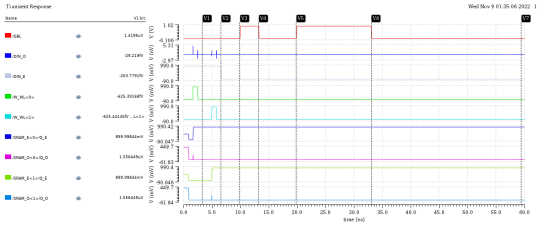

Fig. 4. Timing diagram of the SRAM Write operation

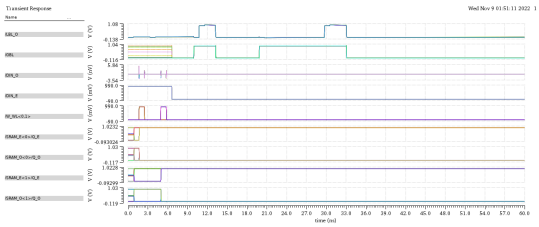

Fig. 5. Timing diagram of the SRAM Write operation after Monte-Carlo Simulation

since the write wordline for address 0 (W_WL[0]) is activated after precharging, the values, X[0] and X[1] will be written in their respective even and odd SRAM cells with the help of Write drivers. Similarly, X[2] and X[3] get written in their respective SRAM cells in the next cycle due to the activation of W_WL[1]. Fig. 5 represents the timing diagram of the SRAM Write operation for the same after the Monte-Carlo Simulation. This confirms no SRAM write errors or disturbances.

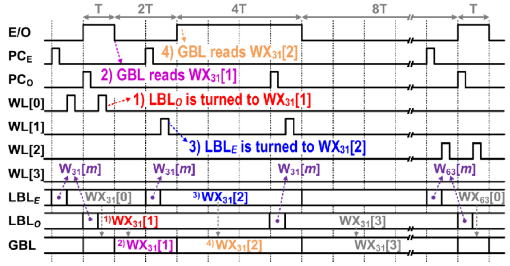

Fig. 6. Proposed timing diagram of the PWM pulse generation

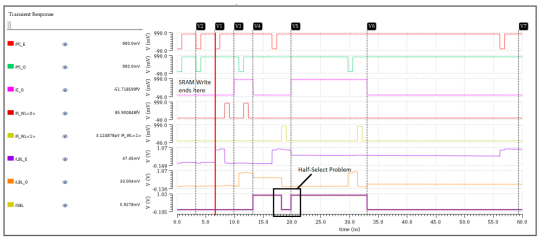

Fig. 7. Proposed PWM pulse generation waveform on Virtuoso

Fig.6 shows the proposed timing diagram of the PWM pulse generation of WX[3:0]. The E/O signal switches in a ratio of 1:2:4:8 cycles to get the PWM voltage signals of W[m]X[3:0] in the time domain. But, as we can see in Fig. 7, when the PWM pulse was generated for W[m]=1 and X[3:0]=0110, the

PWM signal went to a 0 just before the switching of E/O signal, but in theory, it should have retained 1 for the PWM signal to represent 0110 in the time domain. This problem is known as the ”Half-selection problem”. Here, when LBL_O, which was turned to W[m]X[1] (=1), was driving the GBL, the overlapping precharge and computation due to activation of R_WL[1] not only turned LBL_E to W[m]X[2] (=1) but also turned LBL_O to W[m]X[3] (=0) thus changing the value being driven to GBL as R_WL[1] is common for all the SRAM cells in address 1.

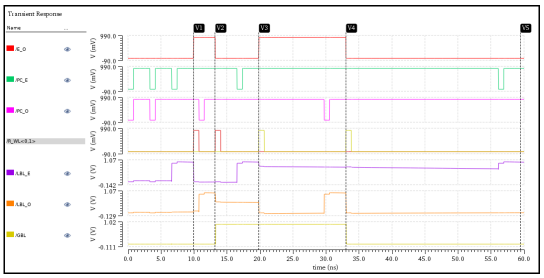

Fig. 8. PWM pulse generation after eliminating the Half-selection problem

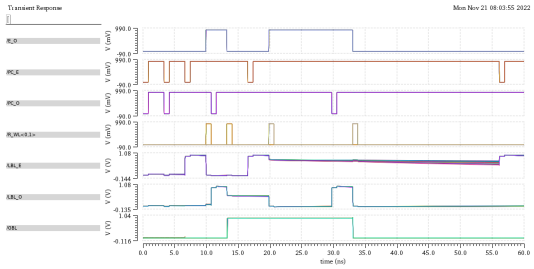

Fig. 9. Timing diagram of the PWM pulse generation after Monte-Carlo Simulation

Therefore, the approach to getting accurate PWM voltage signals without making any circuital changes is to make timing-related changes. As we can see in Fig. 8, We have assumed to keep the overlapping precharging timing as it is but decided to drive the read wordline at the time of the E/O signal flipping to avoid any PWM data being changed due to the half-selection problem. Fig. 9 shows the PWM pulse generation for the same after the Monte-Carlo Simulation. This confirms no read or computation errors.

C. Results

We have carried out total energy analysis for a 16x2 LDCA for various inputs and weight bits. The table in Fig. 10 shows the total energy (in fJ) generated for the PWM pulse generation for various inputs (X[3:0]) and weight bits. Fig. 11 shows the comparison between W[m] = 0 and W[m] = 1. We can observe a huge variation in the energy values when PWM data is computed through W[m] = 1 but we see that LDCA consumes exactly the same energy to generate PWM signals when W[m] = 0. Primarily, this is because of the fact that the LBLs will always stay at 0 throughout the 15 cycles without any charging or discharging. Secondly, the tri-state buffer mostly remains idle in the case of W[m] = 0 as the tri-state buffer consumes energy to drive GBL only when the in-memory AND operation values change from 0 to 1 or 1 to 0.

X[3:0]	Total Energy (in fJ)	
	W[m] = 1	W[m] = 0
0	8.838	3.535
1	16.68	3.541
2	25.11	3.538
3	16.41	3.547
4	60.85	3.555
5	67.42	3.563
6	24.71	3.56
7	15.67	3.573
8	87.36	3.571
9	94.89	3.583
10	80.36	3.579
11	62.15	3.569
12	58.33	3.578
13	64.98	3.58
14	22.95	3.55
15	13.37	3.842

Fig. 10. Total energy (in fJ) generated for the PWM pulse generation for various inputs and weight bits

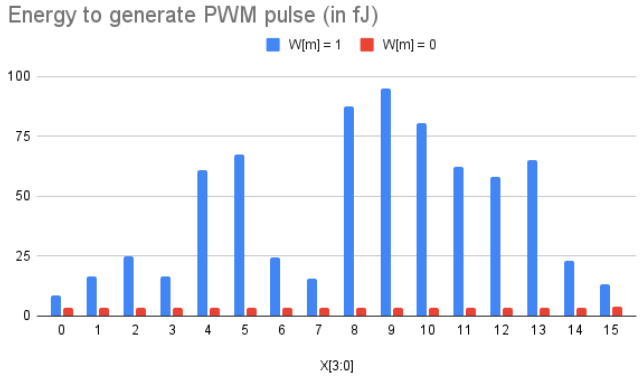

Fig. 11. Comparison of the total Energy in PWM pulse generation for both W[m] = 0 and 1

Data Stored in the SRAM	Worst Case Computation Delay
X[0]	10.98 ps
X[1]	15.8 ps
X[2]	19.1 ps
X[3]	20.3 ps

Fig. 12. Worst case Computational Delays

We have also carried out the computational delay analysis of the LDCA which is a governing factor for determining the throughput of the CIM. Fig. 12 shows the worst-case computational delays computed from different SRAM cells considering a 4-bit input, X[3:0] and W[m]=1 since LBLs will always be 0 in the case of W[m]=0.

VI. Phase Near-Memory-Array Computing

A. Schematics

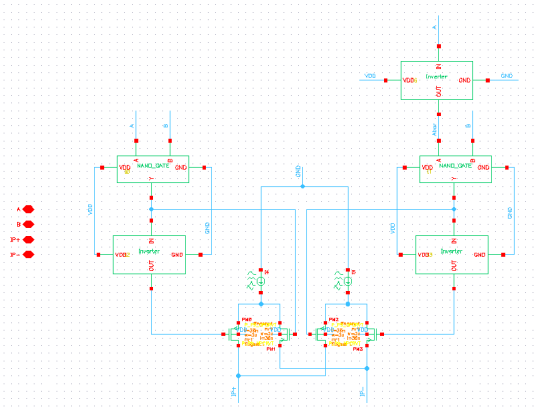

Fig. 13. Current Steering Controller Schematic

Fig. 13 represents the schematic of a Current Steering controller (CSC). This controls the Differential Current Source Pairs (DCSPs) which we have taken as 1 micro-A each, based on the 4b multiplication values from the GBLs of the DIMAC and the sign bits of weights. The following three cases arise based on the values of WX and S: 1) When The WX=0 the unit current (Iu) is added to both Ip and In irrespective of the sign bit value. 2) When the value of WX=1 and S=0, Double the unit current is added to Ip, and no current is added to the In. 3) When the value of WX=1 and S=1, Double the unit current is added to the In, and no current is added to the Ip

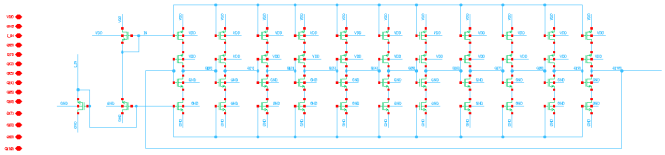

Fig. 14. Differential Current Controlled Oscillator Schematic

Fig. 14 represents the schematic of a Differential Current Controlled Oscillator, an 11-stage ring oscillator, which accumulates the phases based on the current translated from the CSCs. Further, the signal margin of PNMAC is improved by doubling the frequency difference by reusing the steering current in the CCO on the opposite side.

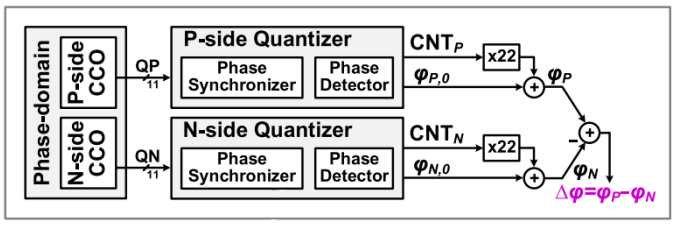

Fig. 15. MAC result readout circuit

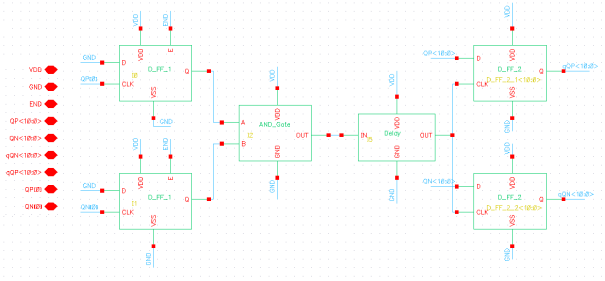

Fig. 16. Phase Synchronizer Schematic

Fig. 15 represents the MAC result readout circuit which consists of 2 phase quantizers, 2 constant multipliers, and 3 adders. The counter outputs and the latched DCCO outputs are sampled only once at the end of every MAC operation as the MSBs and LSBs respectively. We have implemented only the schematic of the phase synchronizer as shown in Fig. 16. The rising-edge detection D flip-flops are enabled by the END signal triggered at the end of the MAC Operation to detect the rising edges of QP[0] and QN[0]. Since the SAMPLE signal is synchronized with QP[0] and QN[0], it will switch from low to high after both QP[0] and QN[0] are 1 after which the DCCO outputs are sampled(latched) at the rising edge of the SAMPLE signal.

B. Waveforms

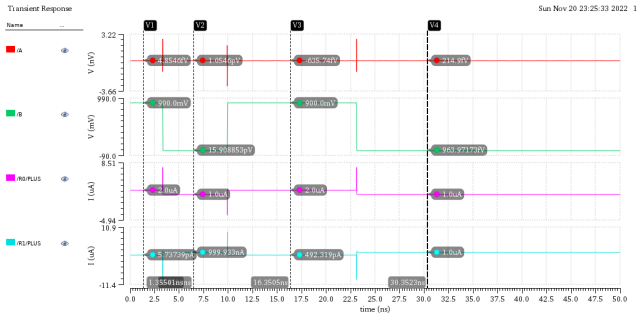

Fig. 17. Current Steering for 0101 PWM signal and sign bit of 0

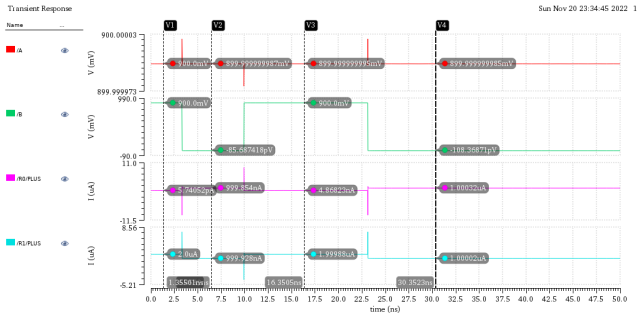

Fig. 18. Current Steering for 0101 PWM signal and sign bit of 1

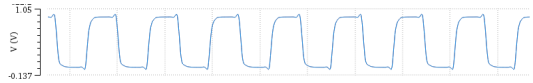

Fig. 19. Zoomed-in DCCO Outputs for PWM Data 0101 and sign bit 0

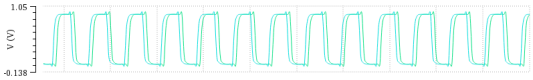

Fig. 20. Zoomed-in DCCO Outputs for PWM Data 0101 and sign bit 0

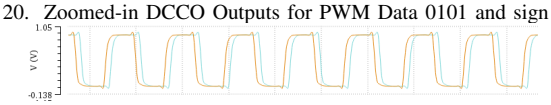

Fig. 21. Zoomed-in DCCO Outputs for PWM Data 0101 and sign bit 0

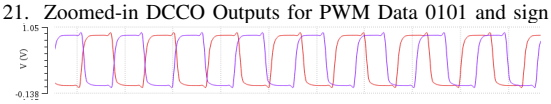

Fig. 22. Zoomed-in DCCO Outputs for PWM Data 0101 and sign bit 0

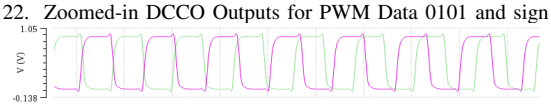

Fig. 23. Zoomed-in DCCO Outputs for PWM Data 0101 and sign bit 0

VII. Novelty

As we have seen before in Fig. 6 and Fig. 7, Half-selection problem is seen as a major problem in the PWM pulse generarion of the DIMAC. The proposed architecture or the timing of the signals doesn't take into account this issue. Therefore, to tackle this issue and not disrupte the PWM pulse generation, we have come up with two ways:

- Circuitally, we can have two different read wordlines, one for odd and another for even so that the computation can be done on the respective LBLs, and thus not interferring with the value already being driven to GBL by the other set of LBLs. But the tradeoff of this method comes in the extra area required for the extra wordline driver.
- Another way is to make timing-related changes as we implemented in Fig. 8. We have assumed to keep the overlapping precharging timing as it is but decided to drive the read wordline at the time of the E/O signal flipping to avoid any PWM data being changed due to the half-selection problem. Thus, the MAC Rate will still remain the same, and since there is no circuit-level changes, area is also remain the same.

References

[1] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," IEEE J. Solid-State Circuits, vol. 52, no. 4, pp. 915–924, Apr. 2017.

[2] X. Si et al., "A twin-8T SRAM computation-in-memory unit-macro for multibit CNN-based AI edge processors," IEEE J. Solid-State Circuits, vol. 55, no. 1, pp. 189–202, Jan. 2020.

[3] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," IEEE J. Solid-State Circuits, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.

[4] Z. Chen et al., "CAP-RAM: A charge-domain in-memory computing 6T-SRAM for accurate and precision-programmable CNN inference," IEEE J. Solid-State Circuits, vol. 56, no. 6, pp. 1924–1935, Jun. 2021

[5] Y. Toyama, K. Yoshioka, K. Ban, S. Maya, A. Sai, and K. Onizuka, "An 8 bit 12.4 TOPS/W phase-domain MAC circuit for energy-constrained deep learning accelerators," IEEE J. Solid-State Circuits, vol. 54, no. 10, pp. 2730–2742, Oct. 2019.

[6] A. Sayal, S. S. T. Nibhanupudi, S. Fathima, and J. P. Kulkarni, "A 12.08-TOPS/W all-digital time-domain CNN engine using bi-directional memory delay lines for energy efficient edge computing," IEEE J. Solid-State Circuits, vol. 55, no. 1, pp. 60–75, Jan. 2020.

[7] R. Khaddam-Aljameh et al., "HERMES core—A 14 nm CMOS and PCM-based in-memory compute core using an array of 300ps/LSB linearized CCO-based ADCs and local digital processing," in Proc. Symp. VLSI Circuits, Jun. 2021, pp. 1–2.

[8] J.-H. Kim, J. Lee, J. Lee, J. Heo, and J.-Y. Kim, "Z-PIM: A sparsity-aware processing-in-memory architecture with fully variable weight bit-precision for energy-efficient deep neural networks," IEEE J. Solid-State Circuits, vol. 56, no. 4, pp. 1093–1104, Apr. 2021.

[9] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A Variation-tolerant in-memory machine learning classifier via on-chip training," IEEE J.Solid-State Circuits, vol. 53, no. 11, pp. 3163–3173, Nov. 2018.

[10] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: An in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism," IEEE J. Solid-State Circuits, vol. 55, no. 7, pp. 1888–1897, Jul. 2020.

[11] X. Si et al., "A local computing cell and 6T SRAM-based computing-in-memory macro with 8-b MAC operation for edge AI chips," IEEE J. Solid-State Circuits, vol. 56, no. 9, pp. 2817–2831, Sep. 2021.