Ternary In-Memory Computing with Cryogenic Quantum Anomalous Hall Effect Memories

Arun Govindankutty
North Dakota State University
Fargo, ND, USA
arun.g@ndsu.edu

Shamiul Alam
University of Tennessee
Knoxville, TN, USA
salam10@vols.utk.edu

Sanjay Das
North Dakota State University
Fargo, ND, USA
sanjay.das@ndsu.edu

Nagadastagiri Challapalle*
NVIDIA
CA, USA
nchallapalle@nvidia.com

Ahmedullah Aziz
University of Tennessee
Knoxville, TN, USA
aziz@utk.edu

Sumitha George
North Dakota State University
Fargo, ND, USA
sumitha.george@ndsu.edu

ABSTRACT

With surging interest in quantum computing, space applications, and ultra-fast superconducting processors, the need for compatible cryogenic memory systems is skyrocketing. Among several concurrent candidates for cryogenic data storage solutions, quantum anomalous Hall effect (QAHE) devices have garnered immense interest due to having topologically protected variation-tolerant quantum states. QAHE cells, in addition to being a promising non-volatile storage technology, have several unique properties that make them ideal for in-memory computing operations. In this work, we propose a novel in-memory computing mechanism by harnessing the intrinsic voltage addition property of a QAHE memory array, implemented using twisted bi-layer graphene (tBLG) on hexagonal boron nitride (hBN). In addition, we extensively explore and implement ternary arithmetic operations utilizing the series-connected Hall voltages across devices for the first time. We propose two schemes for in-memory ternary computing namely IMFE and IMSE, and demonstrate balanced scalar multiplication, dot product operations, and ternary half adder with QAHE memory array.

CCS CONCEPTS

• **Hardware → Analysis and design of emerging devices and systems**.

KEYWORDS

Cryogenic memory, dot product, in-memory computing, MAC, parallel processing, scalar multiplication, ternary arithmetic, QAHE.

ACM Reference Format:
Arun Govindankutty, Shamiul Alam, Sanjay Das, Nagadastagiri Challapalle, Ahmedullah Aziz, and Sumitha George. 2023. Ternary In-Memory Computing with Cryogenic Quantum Anomalous Hall Effect Memories. In *Proceedings of the Great Lakes Symposium on VLSI 2023 (GLSVLSI '23), June 5–7, 2023, Knoxville, TN, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3583781.3590236

*Work done while the author was at Penn State, USA

GLSVLSI '23, June 5–7, 2023, Knoxville, TN, USA

ACM ISBN 979-8-4007-0125-2/23/06.
https://doi.org/10.1145/3583781.3590236

1 INTRODUCTION

Cryogenic computing has recently gathered a surging interest with the advent of quantum computing and ultra-fast superconducting processors [1] [2]. Both the quantum computing systems and superconducting computers utilizing Josephson junctions require ultra low operating temperatures [3] [4]. One major obstacle in realizing cryogenic computing is the lack of suitable cryogenic memories [3]. Recent studies show the functionality and retention properties of DRAM modules at lower temperatures and demonstrate that DRAMs remains functional at temperatures as low as $77K$ [5] and high-speed serial link can be used to connect DRAM to $4K$ domain with an additional cost [6]. Considering the future technology requirements, a cryogenic in-memory computing is desirable.

Recently S. Alam et.al[7] demonstrates a cryogenic memory device leveraging quantum anomalous Hall effect (QAHE) phenomenon. Among several concurrent candidates for cryogenic data storage solutions, QAHE devices have garnered immense interest due to having topologically protected variation-tolerant quantum states. The device uses hall resistance for storage and it exhibits hysteritic switching with change in electric current. The ferromagnetic materials used, interact strongly with applied current, thus enabling electrical control over the polarization and thus on hall resistance. These memories are considered ultra-low power, non-volatile and capable of dense scalable cross point array organizations [7]. In addition to being a promising non-volatile storage device, QAHE cells possess some unique characteristics that makes them suitable for in-memory computing. Basic bit-wise binary in-memory operations with QAHE memory array has been demonstrated in [8].

In this work, we enable compute capability into QAHE cryogenic memory array by exploiting the series addition of Hall voltages across each cell for implementing higher level arithmetic operations. We explore ternary in-memory computing using QAHE cells and exploit its benefits. Since QAHE cell, inherently is a binary storage device, we propose to use multi-bit encoding scheme to represent the ternary operands. We choose encoding such that, unique logic operation produces distinct cumulative voltage at the row output eliminating inference conflicts. Our work demonstrates fundamental ternary operations including balanced scalar multiplication, dot product computation , and ternary addition along with

the possibility for parallelization in a QAHE array. To the best of our knowledge this is the first work showcasing ternary in-memory computing with QAHE devices. Ternary Deep Neural Networks using balanced ternary representation for weights(-1,0,-1) are hugely popular due to their faster and less resource intensive computation with minimal trade-off in accuracy [9][10] and benefits directly from the proposed work.

Our contributions in this paper are summarized as follows:

- We propose to utilize the series Hall voltage addition of QAHE cells for complex arithmetic operations.
- We extensively explore and demonstrate ternary in-memory computing functions inside the QAHE memory array for the first time.
- We provide two new data encoding schemes namely In Memory Full Encode Computing (IMFE) and In Memory Select Encode Computing (IMSE) for executing ternary computations.
- We propose and verify ternary balanced scalar multiplication, parallel dot product computations and half adder operations using QAHE array in-memory computing in simulation.

The rest of the paper is organized as follows. Section 2 describes basic QAHE device as memory operation. Section 3 elaborates on the proposed ternary in-memory computing techniques. Section 4 discusses functional verification. Section 5 summarizes the overall work.

2 DEVICE AND MEMORY DESIGN

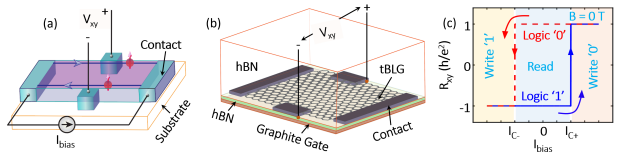

Figure 1: (a)Schematic of single QAHE cell showing bias current I_{bias} and the resulting Hall voltage V_{xy} (b)Structure of QAHE device (c)Ideal Hall resistance switching curve with critical current I_c and corresponding logic states (logic0 and logic1) [7].

Anomalous Hall Effect (AHE) is the phenomenon, in which Hall-voltage is generated across the terminals of a device even in absence of magnetic field. The quantized version of this effect where the hall conductivity is quantized (e^2/h), is called Quantum AHE (QAHE). This effect rises from the topology and strong intrinsic interactions in the material [11] and is oblivious to low sample quality[12].A memory design using twisted bi-layer graphene (tBLG) [7] shows strong interactions with an applied current [13] [11] and enables the switching the hall resistance using bias currents. The schematics of QAHE memory device, structure of the memory device using tBLG and hexagonal Boron Nitride (hBN),and bias current dependencies of hall resistance discussed in [7] are shown in Fig.1(a) (b) and (c) respectively .

A cross point memory array architecture proposed in [7] is shown in Fig.2. The QAHE cells are connected via word line (WL) row wise and bit line (BL) in columnwise. Cell enablement via BL

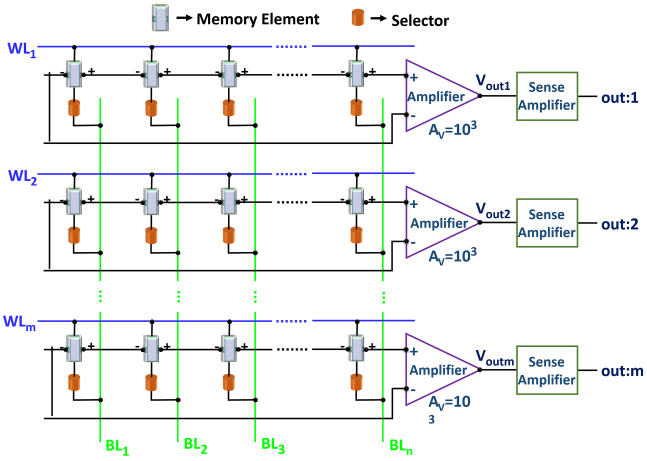

Figure 2: QAHE cross-point memory array organization with horizontal word lines (WL_n) and vertical bit lines (BL_n). V_{out} (mV) appears at the amplifier output and sense amplifier deciphers the logic state[7].

Table 1: Operating voltage and current of QAHE memory array [7]

Operation	V_{bias} (mV)	Bias Current, I_{bias}		
		Accessed	Half-Accessed	Un-Accessed
Write'1'	-500	-6.11nA	-11.4pA	512fA
Write'0'	460	0.922nA	4.38pA	512fA
Read	-480	-2.02nA	-9.73pA	512fA

has a selector device which limits the current flow protecting the data corruption in adjacent cells [14] during write operations.

Table 1 shows read and write bias voltages (V_{bias}) across WL and BL of a cell and corresponding bias current values for the read/write operations. A bias current greater than or equal in magnitude to $-6.11nA$ is passed through the device for writing and is considered $logic1$ in this work. This current results in switching the resistance of QAHE cell to $-h/e^2$(Hall resistance). Similarly a current of $923pA$ results in a resistance value of $+h/e^2$ and is considered $logic0$ in this work. The write operation is one cell at a time for the shown memory array. We obtain a hall voltage of approximate $+50\mu V$ across the device corresponding to $logic1$ and $-50\mu V$ corresponding to $logic0$ with the application of read current, $-2.02nA$. The fully accessed cell will have $I_{bias}\ x\ R_{he}$ developed across the terminals and un-accessed cells will have 0 V[7]. The array output is connected to a voltage amplifier of gain 1000x to raise the voltage from micro-volts to milli-volts range and a voltage comparator (sense amplifier) decodes the voltage output to logic levels. S. Alam et.al [7] demonstrate memory operation capability of the QAHE memory by reading and writing to a single cell at a time.In this work, we show how to enable complex in-memory computation involving multiple cells inside QAHE memory array and is described in the next section.

3 QAHE ARRAY IN-MEMORY COMPUTE

We propose to exploit the series addition of Hall voltages in a QAHE cross-point memory row for efficient in-memory computing. Higher

Table 2: Weight and Input encoding scheme for IMFE and IMSE methods

Trit	IMFE			IMSE		
	Weight	Input	Output	Weight	Input	Output
-1	0111	0111	10	0011	0011	10
0	0000	0000	00	1010	0110	00
+1	1111	1111	01	1100	1100	01

radix compute like ternary compute has more computation density advantages over binary counterpart and can be efficiently implemented with our proposed method. In this work, we show ternary computation and each trit (basic ternary digit) is encoded with multiple QAHE bits (each QAHE cell represents a bit). We utilise the inherent Hall voltage output addition of selected cells in the same row to get a cumulative voltage output and use an inference scheme to interpret the cumulative voltage output as arithmetic result. We propose two different methods for in-memory computing viz; IMFE (In Memory Full Encode) and IMSE (In Memory Select Encode). We demonstrate balanced ternary scalar multiplication, $2D$ and $3D$ dot product computation, and ternary half adder using in-memory compute. The subsequent sections describe each operation in detail.

3.1 Balanced Ternary scalar multiplication

Balanced ternary arithmetic is gaining traction due to its ability to provide acceptable accuracy with higher efficiency[9] especially in Deep Neural Networks. Balanced ternary representation and arithmetic further simplifies the complexity of design and provides acceptable accuracy for computations in Ternary DNNs [9] where weight is limited to one of the three discreet values viz, $-1, 0, +1$. In Ternary DNNs, scalar multiplication is vital due to large number of MAC operations involved in its convolution layers. We achieve scalar multiplication using two new schemes viz; In-Memory Full Encode (IMFE) Computing and In-Memory Select Encode (IMSE) Computing, each with its own benefits. Subsequent sections explain the process of scalar multiplication in detail.

3.1.1 In-Memory Full Encode (IMFE) Computing. In this method, we encode both input operands using multiple cells in the QAHE memory row. ie, QAHE memory array serves as a placeholder for balanced ternary encoded weights and inputs. Hall voltage addition across the row of QAHE memory array is used for inferring the scalar multiplication result. Encoding scheme is employed to ensure different cumulative voltage at the array output for distinct computation outcomes with distinct input combinations, which is explained later. Table 2 captures the input, weight, and output encoding for the corresponding trit values. We use 4 bits (QAHE cells) for weight and input encoding and 2 bits for output encoding.

The weight and input values are stored in consecutive cells as illustrated in Fig. 3a for IMFE based scalar multiplication. Bit lines of all the columns where the weights and inputs are stored are provided with bias voltages so that read current is generated to corresponding cells resulting in cell read operation. Shaded cells indicates that the cells are enabled (or accessed) for read, partaking in scalar multiplication. Hall voltage corresponding to the stored bit is generated across each enabled cell and the cumulative sum of the

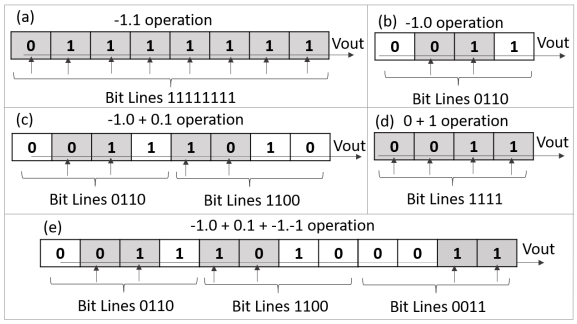

Figure 3: Memory array organization structure for various ternary in-memory compute operations. Bit lines of shaded cells are enabled for read operation. Enabled bit lines are represented as 1 and others as 0. (a) IMFE based scalar multiplication. -1 stored as 0111 and 1 stored as 1111, gives V_{out}=300mV (after 1000x amplification). (b) IMSE based scalar multiplication. -1 stored in cells as 0011 and 0 encoded using bitlines as 0110, giving $V_{out} = 0V$. (c) 2D dot-product computation using IMSE. $(-1.0)+(0.1)$ shown giving $V_{out} = 0V$. (d) Ternary half adder operation using IMFE. 0 stored as 00 and 2 stored as 11. All bit lines enabled for addition giving $V_{out} = 0mV$. (e) 3D dot-product computation using IMSE, $(-1.0)+(0.1)+(-1.-1)$ shown giving $V_{out} = 100mV$.

voltages is obtained at the end of the row. This accumulated voltage is amplified using a high gain voltage amplifier to obtain V_{out} before passing to decode circuit shown in Fig. 4 (a) for interpretation. The sense amplifiers (SA) and an XOR gate in decode circuit gives the encoded output bits $Z1$ and $Z0$. For example, Fig3(a) shows operation $[-1.1]$. The encoded value for -1 is 0111 and for +1 is 1111. When all the bit lines are biased for read, the cumulative output generated is +300μV which is amplified at the array output to +300mV and provided to the decode circuitry shown in Fig 4 (a). Table.3 demonstrates weight and input, array output voltage (V_{out}), encoded output bits ($Z1Z0$) and corresponding trit value for IMFE scheme.

The decode circuit consists of sense amplifiers (SA) which operates based on the reference voltages provided and XOR gate. The output of first SA will be logic high only when the V_{out} value is greater than 50mV. Second and third SA and the XOR gate together forms the logic to obtain the MSB of the output bit, $Z1$. The output $Z1$ XOR-ed with the output of first SA gives the LSB of the output Z0. 300mV is greater than 250mV but less than 350mV making Z1 $logic1$. Output of first SA is $logic1$ as 300mV is greater than 50mV, making Z0 as $logic0$ (1 XOR 1). Thus $Z1Z0 = 10$ and corresponds to -1 as per the output encoding scheme. This method enables parallel scalar multiplications in each row making it amenable for neural networks.

3.1.2 In-Memory Select Encode (IMSE) Computing. This method utilizes both memory elements and bit lines to represent encoded weights and inputs respectively. Weight is encoded using four QAHE memory cells and input is encoded using four bit lines. Based on bit line values(voltages), particular columns get enabled/disabled and hence the name select encode.

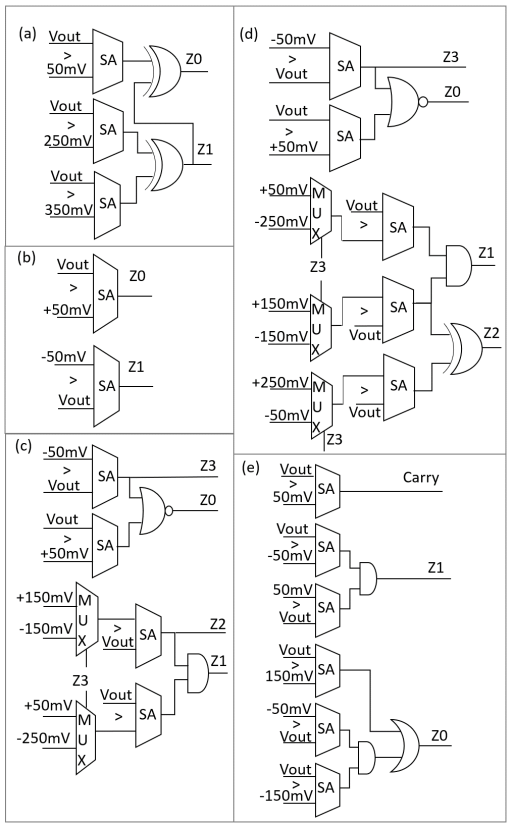

Figure 4: Decode circuits for various in-memory compute operations. Sense amplifier (SA) gives $logic1$ at output when voltage at upper terminal is greater than the voltage at lower terminal. (a) IMFE based scalar multiplication. SA produces $logic1$ or $logic0$ depending on the V_{out} and the reference voltage values, and XOR gates decodes them to corresponding Z1 and Z0 output bits. (b) IMSE based scalar multiplication. (c) 2D dot-product computation decoder. Z3 acts as select input for the MUX ($Z3 = 1$ selects lower reference voltage and $z3 = 0$ selects upper reference voltage). (d) 3D dot-product computation. (e) Ternary half adder decoder

Table 3: Scalar Multiplication using IMFE and IMSE methods

Weight Input		V_{out}(mV)		Output Z1Z0	Output Trit value
		IMFE	IMSE		
-1	-1	200	100	01	1
-1	0	-100	0	00	0
-1	1	300	-100	10	-1
0	-1	-100	0	00	0
0	0	-400	0	00	0
0	1	0	0	00	0
1	-1	300	-100	10	-1
1	0	0	0	00	0
1	1	400	100	01	1

Table.3 shows the encoding scheme for weight, input and output for the IMSE computing scheme. The memory array structure where operands are stored and the output decode circuit is shown in Fig.3b and Fig.4b respectively. The bit lines encoded as 1 (shaded cells in Fig.3b) implies that corresponding column or cell is selected for read operation and the bit line 0 (un-shaded cell) remains unselected. The unselected cells do not contribute in the voltage addition and thus in the operation being carried out. The cumulative voltage is measured at the end of the array and passed to the decode logic after amplification for interpretation. First SA (Fig.4b) gives $logic1$ as output when the voltage V_{out} is greater than $+50mV$ which is the LSB of the output $Z0$ and second SA gives $logic1$ at output only when V_{out} is less than $-50mV$, the MSB ($Z1$). For example, for the operation -1.0, the weight -1 (0011) is stored in the QAHE memory and the input 0 (0110) is provided using the bit line biasing. The output voltage at the end of the array is $0\mu V$ and after amplification, $0mV$ is provided to the decode circuit. Both sense amplifiers in the decode circuit generates zero output making $Z1Z0$ to be 00 which corresponds to the 0 trit value. The weight, input, V_{out}, Output $Z1Z0$, and corresponding trit value are shown in Table 3. This method is suitable for parallel operations when input is the same and weights are different like in dense neural networks. Compared to IMFE, IMSE takes less storage, as in IMSE one operand is coded using bit lines instead of using storage cells. We now extend the method to compute dot product.

3.2 Dot Product computation

Dot product is one of the significant operations in scientific world and forms the core of Multiply and Accumulate (MAC) operation in DNN applications. A dot product computation consists of scalar multiplications and additions, $2D$ dot product is defined as $a1.b1 + a2.b2$ and $3D$ dot product as $a1.b1 + a2.b2 + a3.b3$ where '$a1, a2$, and $a3$' and '$b1, b2$, and $b3$' are components of input vectors 'a' and 'b' respectively. Thus a $2D$ dot product involves two scalar multiplication operations and one addition and $3D$ dot product computation involves 3 scalar multiplications and two additions to obtain the final result. Sections 3.2.1 explains $2D$ dot product and 3.2.2, the $3D$ dot product computation using the IMSE compute scheme proposed in section 3.1.2. Both IMFE and IMSE can be used for dot-product computation, but we choose IMSE in this work due to the compact array size of the method.

Table 4: 2D Dot product output interpretation,

Input Combinations ($a1.b1 + a2.b2$)	V_{out}(mV)	$Z3Z2Z1Z0$	Output Value
-1.1 + 1.-1	-200	1110	-2
-1.1 + 1.0	-100	1000	-1
1.1 + 1.-1	0	0101	0
-1.-1 + -1.0	100	0110	1
1.1 + 1.1	200	0000	2

3.2.1 Dot product of 2D vectors: a1.b1+a2.b2. In this work, we employ IMSE compute scheme to achieve dot product computation. The encoding scheme for weights and inputs remain the same as explained in section 3.1.2. The output encoding uses 4 bits and can have five different values depending on the operands. The five distinct output values $(-2, -1, 0, 1, 2)$ and their bit encoding is

Table 5: 3D Dot Product output interpretation

Input Combinations ($a1b1+a2b2+a3b3$)	V_{out}(mV)	$Z3Z2Z1Z0$	Output Value
-1.1 + 1.-1 + 1.-1	-300	1000	-3
-1.1 + 1.0 + 1.-1	-200	1010	-2
-1.1 + -1.-1 + 1.-1	-100	1100	-1
-1.-1 + 1.-1 + 1.0	0	0001	0
-1.0 + 1.1 + 1.0	100	0010	1
-1.-1 + 1.0 + 1.1	200	0100	2
-1.-1 + 1.1 + 1.1	300	0000	3

captured in Table 4. Table 4 also describes different input vector combinations, distinct output voltage values (V_{out}), corresponding to each output values and its bit combination ($Z3Z2Z1Z0$). The decimal value of the output is also included in Table 4. Memory array compute element configuration for $2D$ dot product computation is shown in Fig.3c. The output decode circuit which interprets the V_{out} value into respective output bit is captured in Fig.4c. The final output is encoded using 4 bits, $Z3, Z2, Z1$ and $Z0$ as per Table 4. $Z3$, MSB of the output can be inferred as sign bit (as it is one only for negative numbers) which can be used by subsequent stages for comparison and provide easy way for minimization and truncation functions in neural network applications if needed. To explain the dot product operation, we consider two vectors $[-1, 1]$ and $[0, 0]$. The dot product of these vectors is calculated as $-1.0 + 1.0$ and the result is 0. The weights -1 (0011) and 0 (1010) are stored in the memory array and bit line values 0110 and 1100 represents 0 and 1 respectively(refer Table 2). The cumulative addition provides a V_{out} of $0mV$ after amplification(refer Fig.3(c)). V_{out} value of $0mV$ drives the output bits $Z3$, and $Z1$ to $logic0$ and drives $Z2$, and $Z0$ to $logic1$ making the output 0101, corresponding to $logic0$ as per output encoding shown in Table 4.

3.2.2 Dot product of 3D vectors: a1.b1+a2.b2+a3.b3. The method mentioned in 3.2.1 is extended further to compute the dot product of two 3-dimensional vectors A and B ($A.B = a1.b1 + a2.b2 + a3.b3$). The memory array compute element employing IMSE method for $3D$ dot product computation is captured in Fig.3d and the output voltage decoding circuit to interpret the V_{out} is shown in Fig.4d. Table 5 summarizes the input combinations, output voltages, corresponding output bits values for $Z3, Z2, Z1$ and $Z0$ and their equivalent value in decimal for better comprehension. Repeated occurrences of input combinations are not explicitly shown in the table for brevity. The output voltage decode circuit is slightly different and consist of slightly more number of components compared to the $2D$ dot product decode circuit. This is because the number of distinct output states are more (seven instead of five) and needs to be decoded accordingly. Consider the dot product of vectors with components $[-1, 0, -1]$ and $[0, 1, -1]$. The expected answer is $(-1.0) + (0.1) + (-1. - 1)$, equals to 1. The V_{out} value obtained at the end of the array after amplification is $100mV$ for which output bits $Z3, Z2, Z0$ are driven to $logic0$ and $Z1$ is driven to $logic1$, making the output 0010 which corresponds to value 1, the expected output. The MSB of the output $Z3$ serves as sign bit in $3D$ dot product computation as well and can be used by subsequent stages as needed.

3.3 Ternary half adder

In this section we show the in-memory ternary half adder. Table 6 shows ternary half adder operations with the input combinations, the output voltage V_{out} generated, the carry and sum outputs ($Z1Z0$) along with the corresponding trit value for ternary half addition. The trit values $0, 1$ and 2 are encoded as $00, 01$ and 11 respectively. Ternary addition proposed in this work utilizes IMFE compute method explained in section 3.1.1. Here, for ternary half addition, the number of QAHE cells used to encode each input is only half (two) compared to scalar multiplication (four) (Section 3.1.1) (For adder, output voltage values are incremental with increasing input digit value. This leads to reduced the complexity of decode and hence requires only reduced number of bits for encoding).

Table 6: Output interpretation for ternary Half adder

Inputs	Encoding	V_{out}(mV)	Carry	Sum (Z1Z0)	Output Trit
0 0	00 00	-200	0	00	00
0 1	00 01	-100	0	01	01
0 2	00 11	0	0	10	02
1 0	01 00	-100	0	01	01
1 1	01 01	0	0	10	02
1 2	01 11	+100	1	00	10
2 0	11 00	0	0	10	02
2 1	11 01	+100	1	00	10
2 2	11 11	+200	1	01	11

An example of ternary half adder operation with inputs 0 and 1 is shown Fig 3d. The addition of inputs 0 and 1 is carried out by storing the input operands in the memory array compute element and then enabling all the cells for read. The resulting V_{out} after amplification provided to the decode circuit to obtain carry bit and sum bits. Ternary half adder output decode circuit for the amplified V_{out} from memory array is shown in Fig.4e. The V_{out} value of $-100mV$ produces carry 0 and sum 01 corresponding to output trit vaule 01 as per the Table 6. In this work we consider only two input operands and thus referred to as ternary half addition. Full addition (including carry input) can be achieved by replicating the half-adder units but is not covered in this work.

4 RESULTS AND DISCUSSIONS

We verify the design functionality by simulation using HSPICE and use phenomenological Verilog-A model[7] for the QAHE devices. This model was calibrated with the experiments reported in [8]. In the waveform,operands are represented as 'Value', BLE is the bit line enable signal (=$-0.48V$), and a voltage of $0.7V$ represents $logic1$ and $0V$ represents $logic0$. Fig. 5a shows simulation output for the IMFE based balanced scalar ternary multiplication operation(-1.1) producing output of $(1, 0)$, corresponding to value -1. Other weight input combinations are not shown in the waveform to keep the figure concise.Fig 5(b) shows simulation waveform for IMSE scalar multiplication. We show all the possible input combinations. For example the first cycle shows operation $[-1.-1]$ producing the output 01 (corresponding to trit value 1), second cycle corresponding to operation $[-1.1]$ yielding output 10 (trit value -1) and so on.

$2D$, $3D$ and parallel dot product computation operations are captured in Fig.5(c)(d) and 6(a) respectively. Fig.6(a) shows the

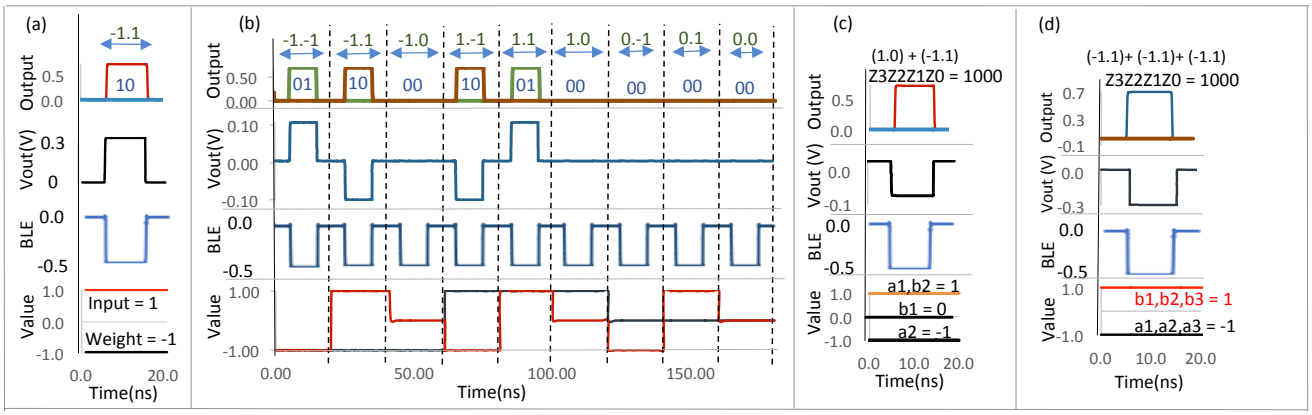

Figure 5: Simulation outputs for various in-memory computing operations. Operands are represented by the 'Value' in the graph, 'BLE' represents bit-line-enable voltages for read operation, 'Vout' represents the amplified output at the end of row, and 'Output' the final decoded logic outputs. (a) IMFE based scalar multiplication of operands -1 and 1, yielding output 10.The two output lines Z1Z0 are shown in red and blue.(b) IMSE based scalar multiplication (shows all possible input operand combinations). (c) 2D dot-product computation ((1.0)+(−1.1), yielding output (1000). (d) 3D dot-product computation ((−1.1)+(−1.1)+(−1.1), yielding output (1000).

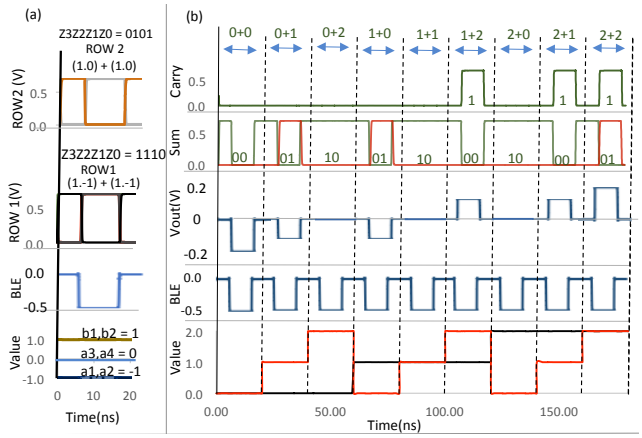

Figure 6: (a) Parallel 2D dot-product compute, ROW1 computes (1.−1)+(1.−1) and ROW2 computes (1.0)+(1.0) (b) Ternary half adder operation with all input combinations.

simulation output of first two rows of an array in parallel for 2D dot product computation. Row 1 computes (1. − 1)+(1. − 1) and Row 2 computes (1.0)+(1.0). Ternary half adder functionality is captured in the Fig. 6(b) and shows the output of addition for all input combinations. For example the first cycle shows operation [0+0] producing the output 000 (corresponding to carry 0 and sum 00), second cycle corresponding to operation [0+1] yielding output 001 (carry 0 and sum 01) and so on.

5 CONCLUSION AND FUTURE WORK

This paper explores the capability of ternary cryogenic in-memory computing and demonstrates the use of binary QAHE memory array for ternary arithmetic. Two methods viz, IMFE and IMSE compute scheme are explained and operations including balanced ternary scalar multiplication, dot product computation, ternary half addition are elucidated. The parallel and in-memory compute capability open up a plethora of opportunities for further research and exploration into cryogenic in-memory computing and motivates further research into QAHE devices.

REFERENCES

[1] P. Wang et. al. Cryogenic performance for compute-in-memory based deep neural network accelerator. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2021. doi: 10.1109/ISCAS51556.2021.9401756.

[2] D.Min. et. al. Cryocache: A fast, large, and cost-effective cache architecture for cryogenic computing. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 449–464, New York, NY, USA, 2020. Association for Computing Machinery.

[3] D.S. Holmes et.al. Energy-efficient superconducting computing—power budgets and requirements. *IEEE Transactions on Applied Superconductivity*, 23(3):1701610–1701610, 2013.

[4] C. L Ayala et. al. Mana: A monolithic adiabatic integration architecture microprocessor using 1.4-zj/op unshunted superconductor josephson junction devices. *IEEE Journal of Solid-State Circuits*, 56(4):1152–1165, 2021.

[5] F. Wang et. al. Dram retention at cryogenic temperatures. In *2018 IEEE International Memory Workshop (IMW)*, pages 1–4, 2018.

[6] F. Ware et. al. Do superconducting processors really need cryogenic memories? the case for cold dram. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '17, page 183–188, New York, NY, USA, 2017.

[7] S.Alam et. al. A non-volatile cryogenic random-access memory based on the quantum anomalous hall effect. *Scientific Reports*, 11(7892), 2021. doi: 10.1038/s41598-021-87056-7.

[8] S. Alam et. al. Cryocim: Cryogenic compute-in-memory based on the quantum anomalous hall effect. *Applied Physics Letters*, 120(14):144102, 2022.

[9] S. Jain et. al. Tim-dnn: Ternary in-memory accelerator for deep neural networks, 2020.

[10] B. Cambou et. al. Can ternary computing improve information assurance? *Cryptography*, 2(1), 2018. ISSN 2410-387X.

[11] M. Serlin et. al. Intrinsic quantized anomalous hall effect in a moiré heterostructure. *Science*, 367(6480):900–903, 2020.

[12] S. Alam et. al. A cryogenic memory array based on superconducting memristors. *Applied Physics Letters*, 119(8):082602, 2021.

[13] A L. Sharpe et. al. Emergent ferromagnetism near three-quarters filling in twisted bilayer graphene. *Science*, 365(6453):605–608, 2019.

[14] G. Burr et. al. Access devices for 3d crosspoint memory. *Journal of Vacuum Science & Technology B*, 32(4):040802, 2014.