Exploring the Feasibility of Using 3D XPoint as an In-Memory Computing Accelerator

Masoud Zabihi, Salonik Resch, Hüsrev Cılasun, Zamshed I. Chowdhury, Zhengyang Zhao, Ulya R. Karpuzcu, Jian-Ping Wang, and Sachin S. Sapatnekar

***Abstract*—This paper describes how 3D XPoint memory arrays can be used as in-memory computing accelerators. We first show that thresholded matrix-vector multiplication (TMVM), the fundamental computational kernel in many applications including machine learning, can be implemented within a 3D XPoint array without requiring data to leave the array for processing. Using the implementation of TMVM, we then discuss the implementation of a binary neural inference engine. We discuss the application of the core concept to address issues such as system scalability, where we connect multiple 3D XPoint arrays, and power integrity, where we analyze the parasitic effects of metal lines on noise margins. To assure power integrity within the 3D XPoint array during this implementation, we carefully analyze the parasitic effects of metal lines on the accuracy of the implementations. We quantify the impact of parasitics on limiting the size and configuration of a 3D XPoint array, and estimate the maximum acceptable size of a 3D XPoint subarray.**
Keywords: 3D XPoint, Phase-change memory, In-memory computing, Matrix-Vector Multiplication, Neural Network.

I. Introduction

With the rapidly increasing sizes of datasets and challenges in transistor scaling in recent years, the need for new computing paradigms is felt more than ever [1]. In today's computing systems, large portions of computation energy and time are wasted for transferring data back and forth between the processor and the memory [2]. One approach is to bring the processor and memory closer to each other and build a near-memory platform that places the computing engine adjacent to the memory, and hence reduce the energy and time overhead for data transfer. Another approach that even more significantly reduces the time overhead and energy is to use the memory device as the computational unit and built a *true in-memory* computing platform. We follow the latter approach.

The substrate that we work on is 3D XPoint [3], a class of memory technology that fills a unique place within the memory hierarchy between solid state storage drive (SSD) and the system main memory. In comparison with the NAND-based SSD (which is the most ubiquitous storage device available today [4]), it has the advantage of being faster, denser, and more scalable. Its nonvolatility differentiates it from competing technologies such as NAND-based SSDs and dynamic random access memories (DRAMs), although NAND-based SSDs are more cost-effective today and DRAMs are faster. Other emerging nonvolatile technologies face limitations: stand-alone PCM must deal with resistance drift, where the cell resistance increases over time [5]; FeFET is handicapped by its large operating voltage and limited endurance [6]; MRAMs require an access transistor (unlike 3D XPoint), leading to a larger cell size than 3D XPoint; ReRAM is not commercially viable to the level of 3D XPoint and MRAM.

The operation and performance of 3D XPoint as a memory unit are discussed in [7]–[11]. In our work, rather than focusing again on the memory aspects of 3D XPoint, we explore the possibility of exploiting 3D XPoint arrays to perform in-memory computation. This means not only that 3D XPoint can function as a storage unit, but also that it can perform computation inside its array without the need for the data to leave the array. Therefore, unlike conventional computational systems, the information can be processed locally rather than being sent to a processor through the memory hierarchy.

The analysis in this paper considers wire non-idealities and physical design of 3D XPoint subarray. We first show the implementation of thresholded matrix-vector multiplication (TMVM), which is a building block for neural networks (NNs) and deep learning applications. Second, using this core operation, we discuss the implementation of a neural network inference engine. Finally, we discuss how to enable 3D XPoint for more complex versions of these implementations (e.g. multi-bit operations and multi-layer NNs).

For in-memory computing platforms, wire resistances are a substantial source of non-ideality that must be taken into account during the implementations [12]. Some works attempt to analyze the parasitic effects of wires but do not consider all contributing factors with realistic layout considerations [13], [14]. In [15], a framework is presented to incorporate the effects of nonidealities in 2D resistive crossbar performance. In [16], an analytical approach is developed to study the effects of the parasitic of wires for the implementation on spintronics computational RAM.

We discuss the feasibility of using 3D XPoint as an in-memory computing engine for neuromorphic applications, and evaluate its performance for MNIST digit recognition. We present novel methods for the implementation of MVMT and NN on 3D XPoint. We use 3D-stacked PCM memory layers in the 3D XPoint subarray to compute and store the computation results entirely inside the array, without sending the data to the periphery of the array. Our method is scalable by using multiple arrays to handle a large computational workload. In addition, multi-bit operations are supported in our methodology. We evaluate a realistic size of the 3D XPoint subarray and metal features (based on the ASAP7 7nm technology [17], [18]) for accurate electrical correctness. We develop a comprehensive method to analyze the impact of wire parasitics of wires in the 3D XPoint subarray and devise a methodology to determine the maximum size of a 3D XPoint subarray that ensures electrically correct operation.

Next, we discuss the structure of 3D XPoint in Section II. In Section III, we describe the implementation of TMVM, and NN. In Section IV, we explore the methods for more complex

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. This work was supported in part by NSF SPX Award CCF-1725420.

implementations. We develop the models for the effect of wire parasitics in Section V, evaluate the results of our analysis in Section VI, and then conclude the paper in Section VII.

II. Background

Some recent works study the implementation of logic operations using a 3D crossbar array architecture. In [19], a double layer Pt/HfO_{2-x}/TiN ReRAM crossbar array is used and it is experimentally shown that the array can implement MVM and CNN. Additional peripheral circuitry (e.g., AND gates) are required for obtaining the computation results. In [20], it is shown that stateful logic operation can be performed on a memristive TiO_2-based 3D crossbar array. The adverse effects of wire non-idealities are not incorporated in the implementations. Similarly in [21], the authors map logic operations on a memristive 3D crossbar without considering wire parasitic or technology design rules. In [22], the authors use 3D memristive crossbars for neuromorphic computation. For the implementation of a neuron, additional amplifier circuitry is required at the periphery of the crossbar.

Fig. 1 shows the overall structure of a 3D XPoint subarray. A two-level PCM stack is integrated at the top of CMOS peripheral circuitry. The storage device is based on phase-change memory (PCM) technology, which is connected to a compatible ovonic threshold switch (OTS) made of AsTeGeSiN [23]–[25]. Word lines at the top (WLTs), word lines at the bottom (WLBs), and bit lines (BLs) in the middle provide the current path to each individual memory cell [26]. The compatibility of the junction of PCM and OTS devices is a key factor in allowing access to individual cells without facing sneak path problems [27]. The total number of PCM cells in the 3D XPoint subarray with N_{row} rows and N_{column} columns is $(2N_{row} \times N_{column})$, with half in the top PCM level and half in the bottom PCM level, as shown in the figure.

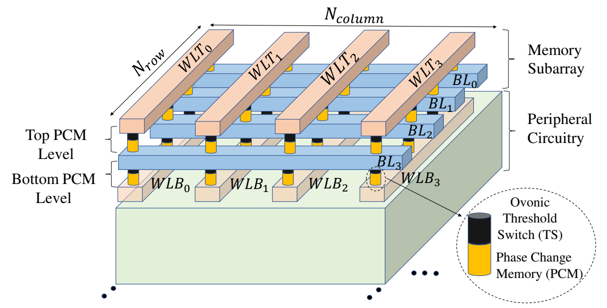

Figure 1: The structure of a 3D XPoint subarray. The CMOS peripheral circuitry is located underneath the memory subarray.

PCM is a non-volatile memory technology exploiting Ge–Sb–Te (GST) alloys (e.g., $Ge_2Sb_2Te_5$) as the storage medium [28]. PCM has two states: a crystalline phase with high conductance (G_C) and an amorphous phase with low conductance (G_A). The GST alloy transition between amorphous and crystalline states is triggered by changing the temperature level [29], [30]. In early explorations of PCM technology (1970s–early 2000s), the temperature level was changed using a laser source [31]. The state-of-the-art research on PCM is focused on using electrical impulses to change the temperature, and hence the state, of the PCM device by applying an electric current (or voltage) pulse across the PCM device [32].

Fig. 2(a) shows that applying a fast high-amplitude current pulse of amplitude I_{RESET} (called the RESET pulse) heats up the GST material to the melting temperature T_{melt} (~600°C or higher [30]), erasing the previous periodic and ordered atomic arrangement. After quenching, the new disordered atomic structure will be frozen, making the transition from high conductance crystalline state to low conductance amorphous state possible. To change the state of the GST from amorphous to crystalline, a slow, relatively low amplitude current pulse of amplitude I_{SET} (called the SET pulse) must pass through the GST material. The SET current pulse causes the GST material to heat up to crystalline temperature T_{cryst} (~400°C [30]). Over a long SET time of several tens of nanoseconds, this is a high enough temperature (still lower than T_{melt}) for the reconfiguration and crystallization of the previous amorphous atomic region to the crystalline state. The desirable PCM characteristics are a lower amplitude of the RESET current and a shorter SET time. A RESET current as low as 10μA and a SET time as low as 25ns for individual PCM devices is already demonstrated with sub-20nm scalability, high endurance 10^{12} cycles, and a projected 10-year retention time at 210°C [5].

Fig. 2(b) shows the electrical model of PCM cell. The resistance across the PCM cell can be modeled by two voltage controlled switches [9]. Depending on the status of switches S_1 and S_2, different currents flow between two lines connected to the terminals of the PCM cell, determined by G_A and G_C. The ON/OFF states of the memory cell are determined by OTS: If the voltage level across the OTS of a cell is larger than a threshold, the cell is considered to be ON, and it is OFF otherwise. In today's technologies, the OTS conductance for the OFF state is up to $10^8\times$ smaller than for the ON state.

The value stored in the PCM device can represent either logic 1 (crystalline phase) or logic 0 (amorphous phase). Three memory operations available in 3D XPoint: write logic 1 (using the fast high-amplitude SET pulse), write logic 0 (using long low-amplitude RESET pulse), and read. For the memory read operation, since it is undesirable to change the state of the PCM cell, a pulse with relatively very small amplitude will be applied, increasing the temperature slightly above the ambient temperature but below T_{cryst} (and of course T_{melt}).

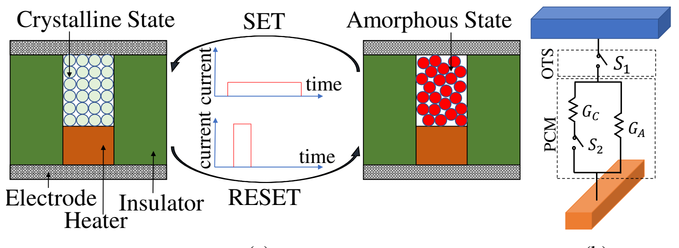

Figure 2: PCM model: (a) the transition between amorphous and crystalline phases by applying SET and RESET pulses across a pillar type PCM device, and (b) PCM cell can be modeled using a resistive circuit with two voltage control switches [9].

III. Realization of In-Memory Computing

A. Implementation of TMVM

TMVM is a fundamental step in the implementation of many applications, and is a fundamental computational kernel in machine learning (ML) applications. Using 3D XPoint as the TMVM computation engine can tremendously decrease the ML computational workload, as the data does not need to leave the 3D XPoint array during the computation.

To show how the first step of a TMVM, let us multiply, without thresholding, matrix $G \in \mathbb{R}^{(N_x+1)\times(N_y+1)}$ and vector $V = [V_0V_1V_2...V_{N_x}]^T \in \mathbb{R}^{(N_x+1)}$, where G is given by:

$$G = \begin{pmatrix} G_{0,0} & G_{0,1} & \dots & G_{0,N_y} \\ G_{1,0} & G_{1,1} & \dots & G_{1,N_y} \\ . & . & \dots & . \\ . & . & \dots & . \\ G_{N_x,0} & G_{N_x,1} & .. & G_{N_x,N_y} \end{pmatrix} \quad (1)$$

This computes $O = \left[O_0O_1O_2,...O_{N_y}\right]^T \in \mathbb{R}^{(N_y+1)})$ where each element of vector O is a dot product. For example,

$$O_0 = G_{0,0}V_0 + G_{1,0}V_1 + ... + G_{N_x,0}V_{N_x} \quad (2)$$

This is computed in the 3D XPoint array by applying voltages across a set of conductances to produce a current O_0.

Today's PCM cells can only store binary values. Hence, we assume that elements of matrix G and vector V are binary. To implement a "neuron-like" operation using TMVM, the O_0 value and computed O_i values are followed by a thresholding operation. In (2), if the sum of products exceeds the current required to flip the output bit, then logic 1 is stored as the conductance, G_{O_0}, of the PCM cell O_0; otherwise, the stored logic value is 0. Similarly, for other O_is, the values after thresholding are stored as the conductance states, G_{O_i}.

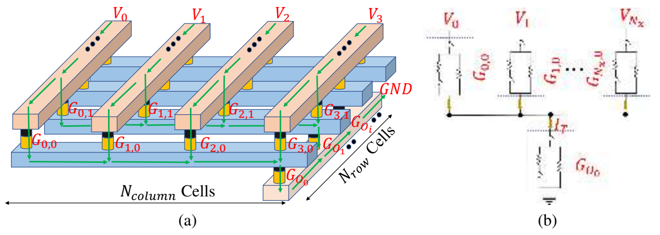

Figure 3: (a) Using 3D XPoint as an in-memory computing engine for TMVM of GV. (b) The equivalent circuit model for the implementation of a dot product (to calculate O_0).

Fig. 3(a) shows the implementation of the TMVM on a 3D XPoint subarray with $(2N_{row} \times N_{column})$ PCM cells ($(N_{row} \times N_{column})$ cells each at top PCM level and bottom PCM level) where $N_{column} = N_x+1$ and $N_{row} = N_y+1$. For clarity, as compared to Fig. 1, only the lines and PCM cells engaged in the computation are shown, and the rest of the lines and the PCM cells (at the bottom) are removed from the figure. All elements of O will be calculated simultaneously and are stored in the same column with N_{row} PCM cells. Considering that today's 3D XPoint cannot store multiple values in a cell, we assume that elements of vector V and G are binary.

The conductances G are first programmed in the top PCM level by memory write operations or by previous computation.

- Before the computation starts, cells that store G_{O_i}s at the bottom are preset to logic 0.
- Then, voltages $V_i, 0 \le i \le N_x$ are applied to the word lines WLTs connected to input cells located at top. If V_i represents logic 1, voltage V_{DD} is applied ($V_i \leftarrow V_{DD}$) to the WLT_i and the current that flows through the corresponding input cell is proportional to $G_{0,i}V_{DD}$.
- If V_i represents logic 0, WLT_i is floated ($V_i \leftarrow$ *float*) and no current passes through the corresponding PCM cell.
- The summation of currents (I_T) from input cells flows through the G_{O_0} in a time interval t_{SET}. Based on the values of V_i and $G_{i,0}$, different currents pass through the input cells that store $G_{i,0}$. If $I_T > I_{SET}$, the state of G_{O_0} changes to logic 1. However, we require $I_T < I_{RESET}$ to avoid erroneous computation.

To calculate the minimum and maximum allowable applied voltage (V_{DD}) to the lines, we consider a simplified electrical model for the implementation of a dot product (e.g., for O_0) shown in Fig. 3(b). Current I_T can be written as follows

$$I_T = G_{O_0}\frac{\sum_{i=0}^{N_x} V_iG_{i,0}}{\sum_{i=0}^{N_x} G_{i,0} + G_{O_0}} \quad (3)$$

When the computation begins, $G_{O_0} \approx G_A$ since the preset is 0, and $I_T(t=0)$ is small (of the order of few hundred nAs). However, by the passage of time, the amorphous region near the heater in the output PCM starts to turn crystalline, resulting in increasing G_{O_0} (and consequently I_T) and heat (generated by the flow of more electric current). If the applied voltage V_{DD} is large enough to provide a current larger than I_{SET}, crystallization repeats until a threshold point where the whole amorphous region in the output PCM turns into a crystalline region with high conductivity, representing logic 1. On the other hand, the V_{DD} must not be so large that the generated temperature exceeds T_{melt}, causing erroneous computation.

To calculate the V_{DD} range for the accurate implementation of the described dot product, we analyze the two cases corresponding to V_{min} (the minimum acceptable voltage) and V_{max} (the maximum acceptable voltage). For the V_{min} case, we assume that all V_is, and all $G_{i,0}$s represent logic 1, i.e., V_{DD} voltages are applied to all WLTs and the conductances of input cells are in the high conductance state corresponding to G_C. In this case, from (3), $I_T = \left(\frac{N_x+1}{N_x+2}\right)G_CV_{DD}$. Since $I_{SET} \le I_T \le I_{RESET}$, the V_{min} requirement implies a first constraint, requiring that V_{DD} to lie in the range:

$$\mathcal{R}_1 = \left[\left(\frac{N_x+2}{N_x+1}\right)\left(\frac{I_{SET}}{G_C}\right), \left(\frac{N_x+2}{N_x+1}\right)\left(\frac{I_{RESET}}{G_C}\right)\right] \quad (4)$$

For the V_{max} case, all V_is are set to logic 1, while all $G_{i,0}$s are set to logic 0. Since the result of the dot product should be at logic 0, we expect that the preset value of PCM stored O_0 remains intact. At the maximum voltage possible, we hypothetically assume that the applied voltage pulse should be below the level required to change the output state from logic 0 to logic 1, even if conductances of all input cells are G_A. In other words, $I_T = \left(\frac{(N_x+1)G_AG_C}{(N_x+1)G_A+G_C}\right)V_{DD} < I_{SET}$, i.e., the output state cannot be altered. Therefore, the second set of constraints require V_{DD} to lie in the range

$$\mathcal{R}_2 = \left[0, \left(\frac{(N_x+1)G_A+G_C}{(N_x+1)G_AG_C}\right)I_{SET}\right] \quad (5)$$

The acceptable range for V_{DD} is $\mathcal{R}_1 \cap \mathcal{R}_2$. Therefore, the minimum and maximum acceptable voltages are $V_{min} = \min(\mathcal{R}_1)$ and $V_{max} = \min(\max(\mathcal{R}_1), \max(\mathcal{R}_2))$, respectively.

B. Implementation of NN

Using the TMVM implementation, we implement a neuromorphic inference engine. Fig. 4(a) shows a single-layer NN with N inputs and P outputs. Fig. 4(b) shows the data layout for the NN implementation on a 3D XPoint subarray. The top PCM cells are allocated for storing the weights ($W_{i,j}$s), similar to

$G_{i,j}$s in TMVM that were stored in the top PCM cells, and the bottom PCM cells are allocated for storing the outputs (Y_is), similar to $O_i s$ in TMVM. The inputs (X_is) are applied to WLT_is as voltage pulses (similar to V_is in TMVM).If $N \leq N_{column}$ and $P \leq N_{row}$, then all Y_js can be determined simultaneously in one step. The output elements of the NN can be stored in any column at the bottom (here, we choose column 1). In Fig. 4(b), all other cells at the bottom patterned by diagonal stripes are not engaged in the computation of Y_is, i.e., WLBs connected to these cells are floated.

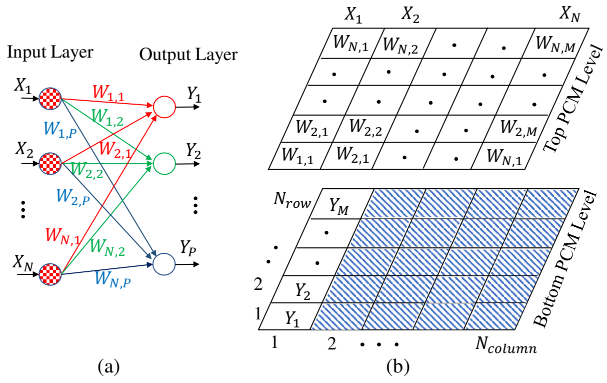

Figure 4: NN implementation: (a) A single-layer neuromorphic inference engine. (b) Data layout for the NN implementation.

An application for the proposed NN implementation is handwritten digit recognition of MNIST dataset with 10K test images [33]. Analyzing each MNIST test image can be performed using a similar NN shown in Fig. 4(a). Here $P = 10$, as in MNIST each image represents a digit (from 0 to 9). In each computational step, $\lfloor \frac{N_{row}}{P} \rfloor$ images can be processed and stored in a column.

IV. Enabling More Complex Implementations

In this section, we discuss three concepts that enable more complex computations. Then, we provide the implementation of a multi-layer NN as an example. Our approach can also be extended to perform multi-bit computation directly using the principles in [34].

A. 3D XPoint with four stacked level of PCM cells

Industry projections show that the next generation of 3D XPoint will have four-level stacked PCM cells [35]. If the number of PCM levels increases, then the volume of stored information per footprint area increases, and more complex implementations are possible. Although a two-level subarray of PCM cells is sufficient to implement any NN (see Section IVD), we will illustrate how we can use a four-level subarray of PCM cells to implement a multi-layer NN by exploiting the extra PCM levels. The NN in Fig. 5 has three layers: an input layer, a hidden layer, and an output layer. At the top PCM level, the first set of weights are stored. In the next PCM level, the hidden layer data is calculated, and by applying the second set of weights as voltage pulses, we obtain the outputs Y_i of the NN at the third PCM level.

B. Scalability of 3D XPoint to large computations

We can connect multiple 3D XPoint subarrays to create a larger array to handle computations with higher matrix

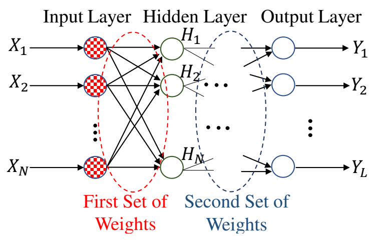

Figure 5: Multi-layer NN with an input, hidden, and output layer.

dimensions. In Fig. 6(a), switches connect BLs of subarray 1 to those of subarray 2, enabling current flow from the BLs of subarray 1 to those of subarray 2. The WLB of subarray 2 that is scheduled to store the computation results will be connected to ground, while all other WLBs not engaging in the computation (in both subarrays 1 and 2) are floated. Hence, the computation results in subarray 1, are being calculated and stored at the bottom PCM level of subarray 2 (BL-to-WLT). In Fig. 6(b), switches connect BLs of the subarray 1 to WLT of subarray 2. In this configuration, the results are being calculated at the top PCM level of subarray 2. The status of lines during the computation for these two configurations are listed in the Supplementary Materials.

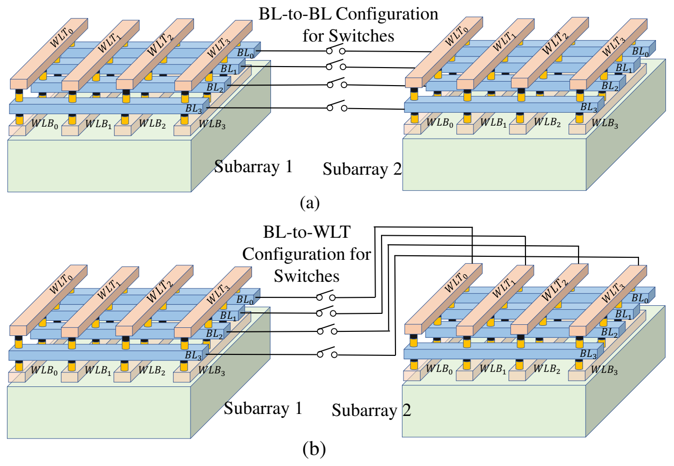

Figure 6: Two configurations for communication between 3D XPoint subarrays: (a) switches connect BLs of subarray 1 to BLs of Subarray 2, and (b) switches connect BLs of the subarray 1 to WLTs of subarray 2.

C. Multi-layer NN implementation in a two-level PCM stack

We now illustrate how a multi-layer NN can be implemented using a three-layer PCM stack. As an example, we discuss the implementation of the three-layer NN (shown in Fig. 5) using two two-level 3D XPoint subarrays. Let us assume that the NN is required to analyze 10K images of the MNIST dataset. The data layout of this implementation is illustrated in Fig. 7 using two subarrays connected with BL-to-WLT configuration (see Fig. 6(b)). The first set of weights is stored at the top PCM cells of subarray 1. The inputs $(X_0, X_1, ..., X_N)$ are applied as the voltages to the WLTs of subarray 1. We assume that at each time step, the hidden layer values $(H_1, H_2, ..., H_N)$ for a specific image from MNIST are being processed. For example, the hidden layer values of the second image $(H_1^{im2}, H_2^{im2}, ..., H_N^{im2})$ is calculated in

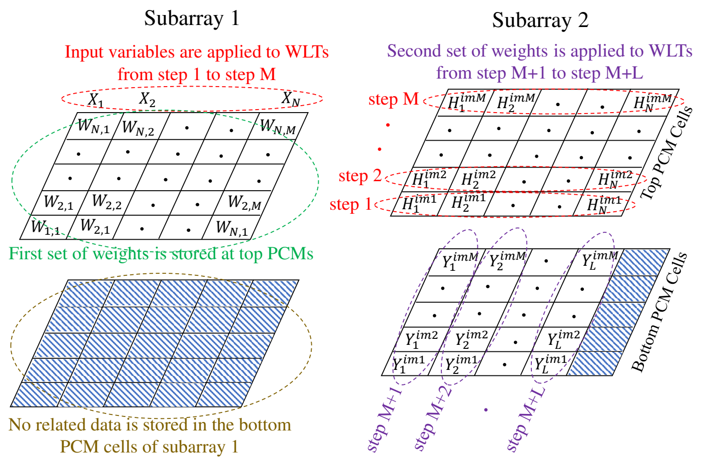
Subarray 1
Input variables are applied to WLTs from step 1 to step M
First set of weights is stored at top PCMs
No related data is stored in the bottom PCM cells of subarray 1
Subarray 2
Second set of weights is applied to WLTs from step M+1 to step M+L
step M
step 2
step 1
Top PCM Cells
Bottom PCM Cells
step M+1
step M+2
step M+L

Figure 7: Data layout for the implementation of 3-layer NN.

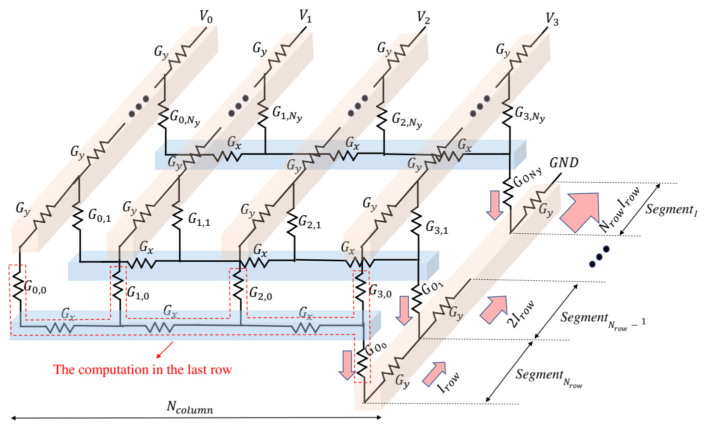

Figure 8: The equivalent circuit model for the TMVM implementation with considering wire parasitics.

step 2. Assuming that we calculate the hidden layer values of $M(= N_{row})$ images in each set of computation, we require M steps to calculate and store the hidden layer values of M images at the top PCM cells of subarray 2. In each of these steps, the corresponding BL in subarray 2 is connected to GND, and the remaining BLs in subarray 2 are floated. After all hidden layer values are stored at the top PCM cells of subarray 2, we apply the second set of the weights (as voltage pulses) to the WLTs of subarray 2. At each column at the bottom PCM cells of subarray 2, the outputs (Y_is) of M images are calculated and stored.

V. Analyzing Interconnect Parasitic Effects

To ensure the electrical correctness of the implementations in in-memory computing platforms, we must consider non-idealities due to wire parasitic effects [12], [16]. As an example, we consider the implementation of a TMVM illustrated in Fig. 3(a). In thhe equivalent circuit model shown in Fig. 8, the WLTs, BLs, and WLBs have nonzero parasitics that cause a voltage drop in the current path across the 3D XPoint subarray, that may potentially lead to errors in the results of TMVM. Let G_x and G_y be the conductances of the segments of BLs and WLs, respectively. The conductances for WLT and WLB are considered equal (both G_y) due to the symmetry and equal allocation of metal resources to WLTs and WLBs. We use $G_{i,j}$ to denote the conductance of PCM cell (i, j) at the top level, and G_{O_j}s to denote conductances of a column of PCM cells at the bottom level. In the worst case, each row performs an identical operation, and carries an equal current I_{row}. The total voltage drop to the last row is

$$\frac{I_{row}}{G_y} + \frac{2I_{row}}{G_y} + ... + \frac{N_{row}I_{row}}{G_y} = \frac{N_{row}(N_{row}+1)I_{row}}{2G_y} \quad (6)$$

where the first, second, and last terms on the left side of the equation are for voltage drops of $Segment_{N_{row}}$, $Segment_{N_{row}-1}$, and $Segment_1$, respectively. The voltage drop of the last row increases quadratically with the number of rows, and this causes a significant limit on the accuracy of the implementations [12], [16]. Hence, it is important to find the maximum allowable subarray size in which the voltage drop does not impair the electrical of implementations.

During the computation, the resistive network shown in Fig. 8 can have different configurations based on the applied voltage to WLTs. For example, if $V_2 \leftarrow float$, then all $G_{2,j}$s and their connected parasitics must be removed from the equivalent circuit model of Fig. 8. To analyze parasitic effects, we consider the corner case for voltage drop, where only $V_0 \leftarrow V_{DD}$ and the rest of the V_is are floated, resulting in minimum equivalent conductance for inputs and wire parasitics. Moreover, for the corner case, we assume that inputs and outputs are located N_{column} columns away from each other (the farthest possible distance). The value of all inputs assumed to be 1, and therefore, the current flows from the inputs of the TMVM computation must be sufficient enough to change the state of the output of the TMVM computation. An excessive voltage drop across the input and output cells causes a failure in the TMVM implementation discussed earlier.

The rows far from the drivers have larger parasitics between them and the driver. In particular, for the last row (farthest from the driver, see Fig. 8), the voltage drop is the worst. If the electrical correctness for the last row does not hold up, the implementations would be unacceptable. We observe the rest of the circuit from the last row and calculate (for the worst case) the Thevenin resistance (R_{th}) and Thevenin voltage (V_{th}) (see Fig. 9(a)). We define the Thevenin coefficient, $\alpha_{th} = \frac{V_{th}}{V_{DD}}$, and its value is between 0 and 1. Both R_{th} and α_{th} can be obtained analytically using a recursive approach explained in Appendix A. Both are functions of parameters such as N_{row}, N_{column}, PCM cell width (W_{cell}) and length (L_{cell}) as well as other parameters of PCM and wire devices. Fig. 9(b) and (c) shows R_{th} and α_{th} for different N_{row} values. For the smaller N_{row}, PCM resistances are the dominant resistances, and the parasitic effect of wires is minimum. When the N_{row} increases the values of the collective parasitic resistances become comparable to those of PCM devices and hence can degrade the electric correctness of the subarray. The configuration of lines are based on configuration 1 that will be discussed in Table I in the next Section.

There are negligible parasitics between the first row and the driver, and the voltage range that ensures accuracy of computing in the first row is closer to $[V_{min}, V_{max}]$ (discussed in Section III) than that of the last row. For the last row, values of α_{th} and R_{th} are significantly affected by parasitics. Let us assume that the new voltage range ensures electrical correctness of the last row is $[V'_{min}, V'_{max}]$. The voltage ranges for the first row and last row are shown in Fig. 10(a). We use the voltage ranges of first row and last row as two corner

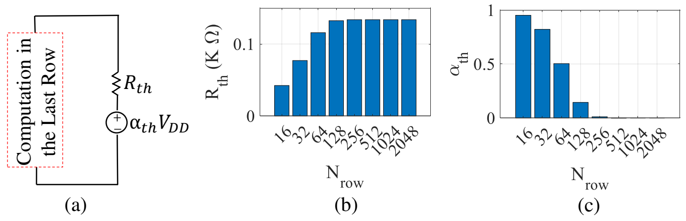

Figure 9: (a) Thevenin equivalents can be observed from the last row, (b) effects of N_{row} on R_{th}, (c) and on α_{th}.

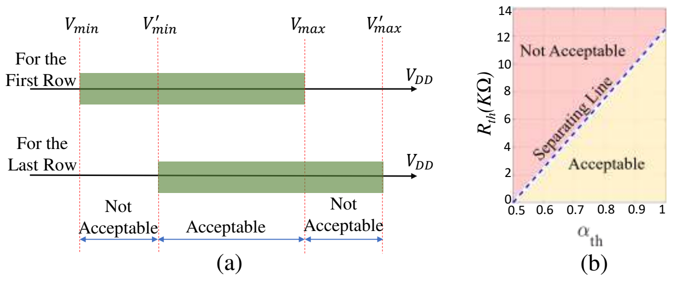

Figure 10: (a) Calculated voltage ranges for the first and last rows. (b) Acceptable and unacceptable regions in the (α_{th}, R_{th}) plane.

cases (with least and most voltage drops, respectively), and we find a voltage range the satisfy both corner cases; the obtained voltage range guarantees the electrical correctness for intermediate rows as well. The final acceptable voltage range is the overlap between two voltage ranges shown in Fig. 10(a), $[V'_{min}, V_{max}]$, ensuring all of the rows from first row to the last row receiving the proper voltage.

The noise margin (NM) in implementations is defined by

$$NM = \frac{V_{max} - V'_{min}}{V_{mid}} \tag{7}$$

where $V_{mid} = (V_{max}+V'_{min})/2$. Clearly, we desire $NM \geq 0$. In Fig. 10(b), the acceptable and unacceptable regions in the (α_{th}, R_{th}) plane is shown. The NM on the separating line is 0, above it NM is negative (unacceptable), and below it NM is positive (acceptable). Our goal is to choose wire configurations so that the corresponding (α_{th}, R_{th}) of the design falls into the acceptable region with maximum NM possible.

VI. Results and Discussion

A. NM evaluation

To realistically analyze the effect of the parasitics, we assumed that metal layers in 3D XPoint are constructed based on

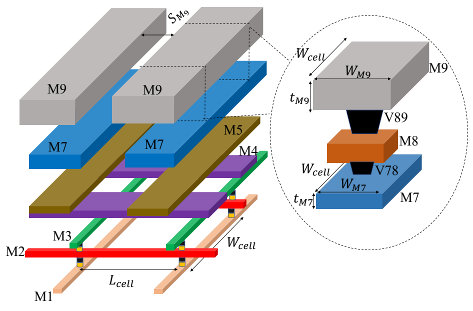

Figure 11: Multi-metal layer configuration can be utilized for the design of WLTs, BLs, and WLBs of 3D XPoint subarray.

Table I: Different Configurations of Metal Lines in the 3D XPoint Subarray Based on ASAP7 Design Rules.

Config	*WLT*	*WLB*	*BL*	$W_{min}\times L_{min}$
1	M3	M1	M2	36nm×36nm
2	M3, M6, M8	M1, M7, M9	M2, M4, M5	48nm×80nm
3	M3, M5, M6, M8	M1, M4, M7, M9	M2	36nm×80nm

ASAP7 design rules [17], [18] (see Fig. 11). We can create different configurations for allocating metal lines to WLTs, WLBs, and BLs. Table I lists three possible configurations. In configuration 1, only M1, M2, and M3 (the first three metal lines) in ASAP7 are exploited for 3D XPoint, and they are allocated to WLB, BL, and WLT, respectively. For configurations 2 and 3, we assume that other than M1, M2, and M3, the other metal layers (M4 to M9) can also be allocated to the 3D XPoint lines. In configuration 2, we allocate M4 and M5 to the BLs, and M6 to M9 are allocated equally between WLTs and WLBs. In configuration 3, we assume that all metals from M4 to M9 are allocated equally between WLTs and WLBs; no extra top metal lines are allocated to BLs. We report the minimum cell width (W_{min}) and length (L_{min}) for each configuration based on the minimum required width of a line and space between adjacent lines in each layer. The values of parameters for metal lines and PCM devices are available in the Supplementary Material.

NM **improves with increasing** N_{row}: Fig. 12(a) shows NMs of different N_{row} values. NM is significantly sensitive to N_{row}. For N_{row} as large as 2048, the implementations are not valid due to excessive voltage drop, and hence negative NM. Configuration 3 provides the best NM, because more metal resources dedicated to WLT and WLB causes smaller parasitics in the current path across rows.

NM **improves with increasing** L_{cell}: Fig. 12(b) shows the NMs for different L_{cell}s (for each configuration, values are normalized to L_{min} listed in Table I). By increasing L_{cell}, the width of the WLTs and WLBs increase, decreasing the parasitic resistances related to WLTs and WLBs.

NM **decreases with increasing** W_{cell}: Fig. 12(c) shows the NMs for different W_{cell} (for each configuration, values are normalized to W_{min} listed in Table I). By increasing W_{cell}, the length of the WLTs and WLBs increase, and consequently, parasitics related to WLTs and WLBs considerably increase. Therefore, for all cases, smaller W_{cell} causes larger NM.

NM **remains unchanged with increasing** N_{column}: Fig. 12(d) shows that the increase in N_{column} does not affect NM significantly. By increasing N_{column}, parasitics of BLs increase. However, since the BL resistances are in series with those of PCM devices with orders of magnitude larger resistance, the increase in BL resistance does not affect NM.

B. Implementing NNs on the 3D XPoint substrate

We list the performance of various 3D XPoint subarrays of various sizes for the digit recognition of MNIST dataset in Table II. Each MNIST test image is scaled to 11×11 as in [36], a transformation that maintains 91% recognition accuracy and reduces computation. We use configuration 3 in all cases, as it provides the best NM among all alternatives. For the smallest subarray with size 64×128, NM is the maximum among all cases. For the largest subarray with size 1024×1024, we increase L_{cell} by 2.6× (compare to that of 64×128 subarray)

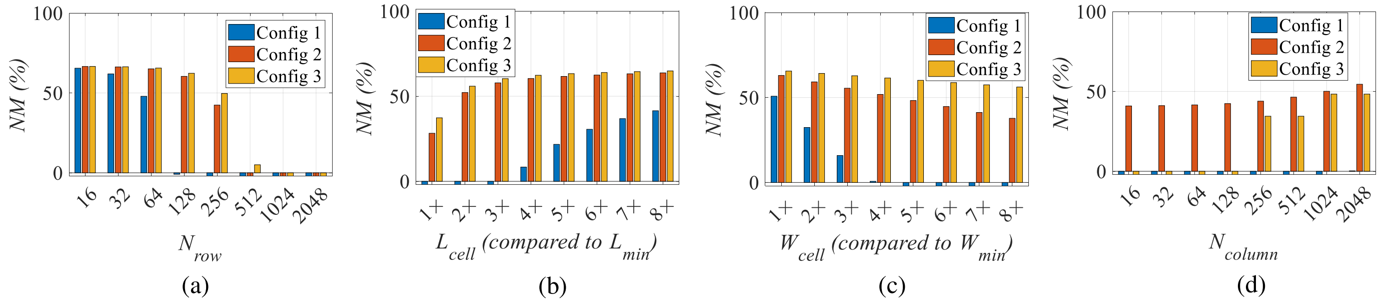

Figure 12: NMs of the three metal line configurations: (a) changing N_{row} (while $N_{column} = 128$, $L_{cell} = 4L_{min}$, and $W_{cell} = W_{min}$), (b) changing L_{cell} (while $N_{column} = 128$, $N_{row} = 128$, and $W_{cell} = W_{min}$), (c) changing W_{cell} (while $N_{column} = 128$, $N_{row} = 64$, and $L_{cell} = 4L_{min}$), and (d) changing N_{column} (while $N_{row} = 256$, $L_{cell} = 4L_{min}$, and $W_{cell} = W_{min}$).

Table II: Evaluation of Different Subarray Sizes for Digit Recognition Application.

Subarray Size	Cell Size (nm×nm)	#Image per Step	Energy per Image	Subarray Area (μm^2)	Execution Time (μs)	V_{max} (V)	V'_{min} (mV)	NM	NM w/ 10% Var.
64×128	36×240	6	21.5pJ	62.9	133.3	1.25	636.2	65.1%	64.9%
128×256	36×320	12	21.5pJ	335.5	66.7	1.25	650.6	63.1%	62.7%
256×512	36×400	25	20.7pJ	1677.7	32.0	1.25	681.0	58.9%	58.1%
512×1024	36×480	51	20.3pJ	8053.0	15.7	1.25	732.5	52.2%	50.8%
1024×2048	36×640	102	20.3pJ	42949.6	7.8	1.25	882.2	34.5%	31.5%

to decrease the parasitics of lines. Consequently, we achieve acceptable NM of 34.5%. With this relatively large subarray, we have more parallelism that allow to process a larger number of MNIST images in each computational step, reducing the total execution time (17× faster than that of 64×128 subarray). The energy per image is similar for all cases because the subarray sizes listed in Table II are large enough to allow fully processing an 11×11 MNIST image locally without extra data movement between subarrays or peripheral circuitry.

The table also shows the impact of 10% process variation on NM. In the second to the last column, interconnect resistances are changed by 10%. For small arrays, the change in NM is negligible because the resistance of the PCM element and OTS dominate wire resistance. For larger arrays, this effect becomes more noticeable (but is still not significant). In the last column, PCM device parameters as well as interconnect resistances are simultaneously varied by 10% to find the worst NM due to variations of all parameters. NMs for all arrays are still positive and acceptable (between 12.4% for the largest subarray and 46.6% for the smallest subarray).

VII. Conclusion

We have presented methods for the implementations of TMVM, NN, and 2D convolution on 3D XPoint. To ensure the accuracy of the implementations, we considered wire parasitics in our implementations. We have demonstrated that interconnect parasitics have a significant effect on the implementations performance and have developed a comprehensive model for analyzing this impact. Using this methodology, we have developed guidelines for the 3D XPoint Subarray size and configurations based on ASAP7 technology design rules. We used different size 3D XPoint subarrays for digit recognition of MNIST dataset. Using the our methodology methodology, we design a relatively large subarray of 2 Mb with acceptable NM of 34.5%, providing the opportunity for processing more images per step without any energy overhead.

References

[1] A. McAfee, *et al.*, "Big data: The management revolution," *Harvard Business Review*, Oct. 2012.
[2] S. W. Keckler, *et al.*, "GPUs and the future of parallel computing," *IEEE Micro*, vol. 31, pp. 7–17, Nov. 2011.
[3] "3D XPoint technology." https://www.intel.com/content/www/us/en/architecture-and-technology/intel-micron-3d-xpoint-webcast.html.
[4] N. Agrawal, *et al.*, "Design tradeoffs for SSD performance," in *USENIX 2008 Annual Technical Conference*, p. 57–70, June 2008.
[5] M. Le Gallo and A. Sebastian, "An overview of phase-change memory device physics," *Journal of Physics D Applied Physics*, vol. 53, p. 213002, May 2020.
[6] S. Beyer, *et al.*, "FeFET: A versatile CMOS compatible device with game-changing potential," in *Proceedings of the IEEE International Memory Workshop (IMW)*, 2020.
[7] K. Son, *et al.*, "Signal integrity design and analysis of 3-D X-Point memory considering crosstalk and IR drop for higher performance computing," *IEEE Transactions on Components and Packaging Technologies*, vol. 10, pp. 858–869, May 2020.
[8] K. Son, *et al.*, "Modeling and verification of 3-dimensional resistive storage class memory with high speed circuits for core operation," in *IEEE Asia-Pacific Microwave Conference*, pp. 694–696, Mar. 2019.
[9] K. Son, *et al.*, "Modeling and signal integrity analysis of 3D XPoint memory cells and interconnections with memory size variations during read operation," in *IEEE Symposium on Electromagnetic Compatibility, Signal Integrity and Power Integrity*, pp. 223–227, July 2018.
[10] Q. Lou, *et al.*, "3DICT: A reliable and QoS capable mobile process-in-memory architecture for lookup-based CNNs in 3D XPoint ReRAMs," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, Jan. 2018.
[11] J. Yang, *et al.*, "Exploring performance characteristics of the optane 3D XPoint storage technology," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 5, Feb. 2020.
[12] D. Ielmini and G. Pedretti, "Device and circuit architectures for in-memory computing," *Advanced Intelligent Systems*, vol. 2, pp. 2000049–1–2000040–19, May 2020.
[13] Y. Jeong, *et al.*, "Parasitic effect analysis in memristor-array-based neuromorphic systems," *IEEE Transactions on Nanotechnology*, vol. 17, pp. 184–193, Jan 2018.
[14] Naveen Murali G., *et al.*, "Modelling and simulation of non-ideal MAGIC NOR gates on memristor crossbar," in *Proceedings of the International Symposium on Embedded Computing and System Design*, pp. 124–128, Dec 2018.
[15] S. Jain, *et al.*, "RxNN: A framework for evaluating deep neural networks on resistive crossbars," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 326–338, 2021.
[16] M. Zabihi, *et al.*, "Analyzing the effects of interconnect parasitics in the STT CRAM in-memory computational platform," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, pp. 71–79, June 2020.
[17] L. T. Clark, *et al.*, "Design flows and collateral for the ASAP7 7nm FinFET predictive process design kit," in *Proceedings of the IEEE International Conference on Microelectronic Systems Education*, pp. 1–4, June 2017.
[18] L. T. Clark, *et al.*, "ASAP7: A 7-nm FinFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, July 2016.

[19] Y. Kim, *et al.*, "Kernel application of the stacked crossbar array composed of self-rectifying resistive switching memory for convolutional neural networks," *Advanced Intelligent Systems*, vol. 2, no. 2, p. 1900116, 2020.
[20] N. Xu, *et al.*, "A stateful logic family based on a new logic primitive circuit composed of two antiparallel bipolar memristors," *Advanced Intelligent Systems*, vol. 2, no. 1, p. 1900082, 2020.
[21] N. G. Murali, *et al.*, "Mapping of boolean logic functions onto 3d memristor crossbar," in *Proceedings of the International Conference on VLSI Design*, pp. 500–501, 2019.
[22] B. R. Fernando, *et al.*, "3d memristor crossbar architecture for a multicore neuromorphic system," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8, 2020.
[23] M. Lee, *et al.*, "Highly-scalable threshold switching select device based on chaclogenide glasses for 3D nanoscaled memory arrays," in *IEEE International Electronic Devices Meeting*, pp. 2.6.1–2.6.3, Dec. 2012.
[24] G. W. Burr, *et al.*, "Access devices for 3D crosspoint memory," *Journal of Vacuum Science & Technology B*, vol. 32, pp. 040802–1–040802–23, July 2014.
[25] DerChang Kau, *et al.*, "A stackable cross point phase change memory," in *IEEE International Electronic Devices Meeting*, pp. 1–4, Mar. 2009.
[26] W. Chien, *et al.*, "A study on OTS-PCM pillar cell for 3-D stackable memory," *IEEE Transactions on Electron Devices*, vol. 65, pp. 5172–5179, Nov. 2018.
[27] L. Shi, *et al.*, "Research progress on solutions to the sneak path issue in memristor crossbar arrays," *Nanoscale Advances*, vol. 2, pp. 1811–1827, Mar. 2020.
[28] J. Y. Wu, *et al.*, "A low power phase change memory using thermally confined TaN/TiN bottom electrode," in *IEEE International Electronic Devices Meeting*, pp. 3.2.1–3.2.4, Jan. 2011.
[29] H. . P. Wong, *et al.*, "Phase change memory," *Proceedings of the IEEE*, vol. 98, pp. 2201–2227, Dec. 2010.
[30] G. W. Burr, *et al.*, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, pp. 146–162, June 2016.
[31] M. Wuttig and N. Yamada, "Phase-change materials for rewriteable data storage," *Nature Materials*, vol. 6, p. 824—832, Nov. 2007.
[32] Q. Zheng, *et al.*, "Nanoscale phase-change materials and devices," *Journal of Physics D: Applied Physics*, vol. 50, p. 243002, May 2017.
[33] Y. LeCun, "The MNIST database of handwritten digits." http://yann.lecun.com/exdb/mnist/.
[34] P. Chi, *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *isca*, pp. 27–39, Aug. 2016.
[35] "3D XPoint with four-level PCM cells." https://www.anandtech.com/show/15972/intel-previews-4layer-3d-xpoint-memory-for-secondgeneration-optane-ssds.
[36] M. Liu, *et al.*, "A scalable time-based integrate-and-fire neuromorphic core with brain-inspired leak and local lateral inhibition capabilities," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 1–4, July 2017.

Appendix

We derive recursive expressions for calculating R_{th} and V_{th} of a $(N_{row} \times N_{column})$ subarray of 3D XPoint.

Within the footprint area of a cell $(W_{cell} \times L_{cell})$, we define G_y (representing the conductance for WLT and WLB segments) and G_x (representing the conductance of BL segment). Fig. 13 shows the equivalent simplified circuit model for the implementation TMVM in the corner case. Row i is separated from its predecessor by conductance G_y at each end. The input cell is connected to the output cell, N_{column} columns away. The conductances in the last row are rearranged to create a two-port structure consisting of the PCM conductances so that the rest of the network can be modeled using Thevenin Equivalents (R_{th} and V_{th}).

For configuration 1 (listed in Table I), $G_y = G_{M_1} = G_{M_3}$ (assuming similar wire conductance for WLTs and WLBs) and $G_x = G_{M_2}$ in which the conductance, G_{M_k}, is given by: $G^{-1}_{M_k} = \frac{\rho_{M_k} L_{M_k}}{t_{M_k} W_{M_k}}$, where ρ_{M_k}, L_{M_k}, t_{M_k}, and W_{M_k} are, respectively, the resistivity, length, thickness, and width in metal layer k (see the Supplementary Material). For configurations 2 and 3, the equivalent conductance of the wire segment must be calculated based on the multi-metal layer configuration

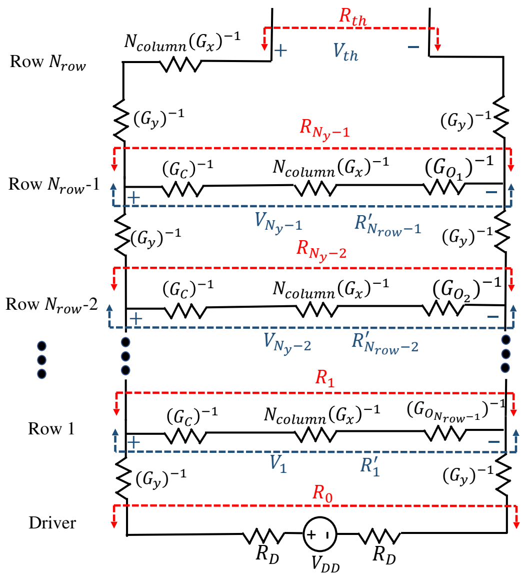

Figure 13: Notations used for calculating Thevenin resistance (R_{th}) and Thevenin voltage (V_{th}) shown on the circuit model for the implementation of TMVM in the worst case scenario.

of a given segment. For example, in configuration 2, G_y (representing a segment conductance of WLT) is obtained by $G_y = G_{M_3} + G_{M_6} + G_{M_8}$.

To calculate R_{th} and V_{th}, we derive recursive expressions. For conciseness, we define the resistance, R_{row_i}, of row i as:

$$R_{row_i} = N_{column}\left(G_x\right)^{-1} + \left(G_C\right)^{-1} + \left(G_{O_{N_{row}-i}}\right)^{-1} \quad (8)$$

We can obtain R_{th}, using the notations in Fig. 13, as:

$$R_{th} = 2\left(G_y\right)^{-1} + N_{column}\left(G_x\right)^{-1} + R_{N_{row}-1} \quad (9)$$

where $R_{N_{row}-1}$ is calculated using the recursive expression:

$$R_i = \left(R_{row_i}\right) || \left(R_{i-1} + 2\left(G_y\right)^{-1}\right) \quad (10)$$

The base case corresponds to the driver row that precedes the first row, and is $R_0 = 2R_D$, as seen in Fig. 13.

To compute V_{th}, as illustrated in Fig. 13, we first compute the intermediate variable R'_j, which corresponds to the effective downstream resistance (away from the source) seen from node j. The computation proceeds in a recursive fashion from the last row towards the first as:

$$R'_{j-1} = \left(R_{row_{j-1}}\right) || \left(R'_j + 2\left(G_y\right)^{-1}\right) \quad (11)$$

with the base case $R'_{N_{row}-1} = R_{row_{N_{row}-1}}$. Having computed R'_j, we may now compute $V_{th} = V_{N_{row}}$, using a recursive computation on V_i:

$$V_j = \frac{R'_j}{2\left(G_y\right)^{-1} + R'_j} V_{j-1} \quad (12)$$

in which $2 \leq j \leq N_{row} - 1$ and the base case is:

$$V_1 = \frac{R'_1}{R'_1 + 2\left(G_y\right)^{-1} + 2R_D} V_b \quad (13)$$

Supplementary Material

A. PCM parameters

The conductance values of parameters in a PCM cell listed in Table III. In this work, we adopt the RESET current (I_{RESET}) of 100μA with RESET Time (t_{RESET}) of 15ns, and SET time (t_{SET}) of 80ns with the assumption of SET current (I_{SET}) of 50μA ($= \frac{I_{RESET}}{2}$) [7], [8].

Table III: PCM cell parameters and values [7], [8]

Parameters	Description	Value
G_A	PCM conductance in the amorphous state	660 nΩ^{-1}
G_C	PCM conductance in the crystalline state	160$\mu\Omega^{-1}$
S_1	Voltage control switch for OTS	100nΩ^{-1}(<0V) and 10Ω^{-1}(>0.3V)
S_2	Voltage control switch for PCM in crystalline state	10Ω^{-1}(<0.8V) and 100nΩ^{-1}(>1V)

B. Interconnect specifications for ASAP7

The interconnect specifications are listed in Table IV, which shows the metal thickness (t_M) and resistivity (ρ_M), the minimum line spacing (S_{min}), minimum line width (W_{min}), and Table V, which shows the via parameters [17], [18].

Table IV: Specification of Metal Layers in ASAP7 [17], [18]

Metal	t_M	S_{min}	W_{min}	ρ_M
M1(V)	36nm	18nm	18nm	43.2Ω.nm
M2(H)	36nm	18nm	18nm	43.2Ω.nm
M3(V)	36nm	18nm	18nm	43.2Ω.nm
M4(H)	48nm	24nm	24nm	36.9Ω.nm
M5(V)	48nm	24nm	24nm	36.9Ω.nm
M6(H)	64nm	32nm	32nm	32.0Ω.nm
M7(V)	64nm	32nm	32nm	32.0Ω.nm
M8(H)	80nm	40nm	40nm	28.8Ω.nm
M9(V)	80nm	40nm	40nm	28.8Ω.nm

Table V: Specification of Vias in ASAP7 [17], [18]

Via	R_V	Via Size	Minimum Spacing
V12 (M1 and M2)	17Ω	18nm×18nm	18nm
V23 (M2 and M3)	17Ω	18nm×18nm	18nm
V34 (M3 and M4)	17Ω	18nm×18nm	18nm
V45 (M4 and M5)	12Ω	24nm×24nm	33nm
V56 (M5 and M6)	12Ω	24nm×24nm	33nm
V67 (M6 and M7)	8Ω	32nm×32nm	45nm
V78 (M7 and M8)	8Ω	32nm×32nm	45nm
V89 (M8 and M9)	6Ω	40nm×40nm	57nm

C. Status of lines during communications between subarrays

The status of 3D XPoint lines during the communications with each other is listed in Table VI.

D. Corner case circuit

The corner case circuit is shown in Fig. 14. In the Appendix A, we simplified the circuit even further and calculate the Thevenine equivalents observed from the last row. In Fig. 15, we show the reconfiguration and simplification of circuit shown in Fig. 14.

Table VI: Status of 3D XPoint Lines for Two Different Configurations

Line	Subarray	Configuration	
		BL-to-BL	BL-to-WLT
WLTs	1	V_is are applied	V_i are applied
	2	all float	all active
BLs	1	all active	all active
	2	all active	all float -{output row connect to the ground}
WLBs	1	all float	all float
	2	all float-{output column connect to the ground}	all float

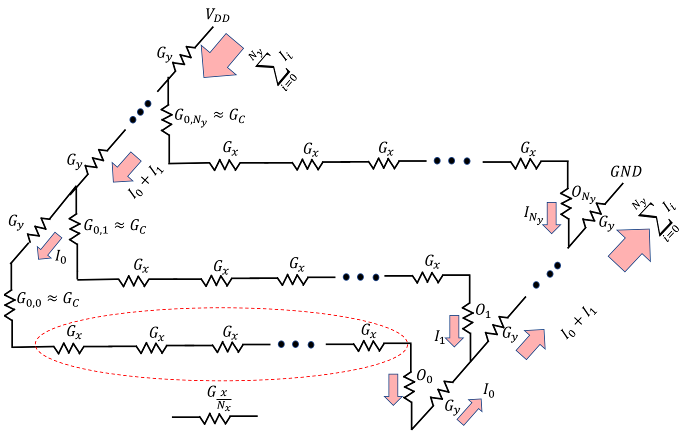

Figure 14: The equivalent circuit model for the worst case

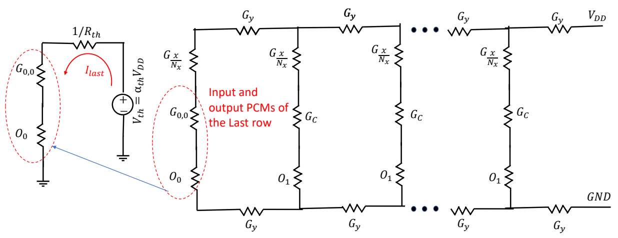

Figure 15: Reconfiguration and simplification of equivalent circuit model.

E. Multi-bit operations

Thus far, we have discussed the implementations of operations with binary digits. We introduce two methods that enable us to implement operations with multi-bit digits. For brevity, we explain the principle using two-bit digits, where each digit consists an MSB (most significant bit) and an LSB (least significant bit). Let us assume that we want to perform TMVM of $G_{2bit}V$ where G_{2bit} is a $(N_x+1)\times(N_y+1)$ matrix with two-bit elements, meaning the element located in row i and column j of the matrix G_{2bit} is $G_{i,j}^{MSB}G_{i,j}^{LSB}$.

Fig. 16 illustrates two ways to implement 2-bit operations on a 3D XPoint subarray. Fig. 16(a) shows an area-efficient approach. For example, to calculate O_0, we need to calculate $(G_{0,0}^{LSB}+2G_{0,0}^{MSB})V_0+(G_{1,0}^{LSB}+2G_{1,0}^{MSB})V_1+...+(G_{N_x,0}^{LSB}+2G_{N_x,0}^{MSB})V_{N_x}$. To do so, we can apply V_0 to the WLT_0 (connected to the PCM storing $G_{0,0}^{LSB}$ bit), and we can apply $2V_0$ to WLT_1 (connected to the PCM storing $G_{0,0}^{MSB}$ bit). Therefore, the current flowing through the MSB cell is two times larger than that of the LSB cell. Another more area-

Table VII: Evaluation of Implementation Energy and Area for Multi-Bit TMVM using Area Efficient and Low Power Schemes.

Parameters	Implementation Scheme	Number of Bits for each G element					
		1	2	3	4	5	6
TMVM Energy (pJ)	Area Efficient	2.0	5.0	13.1	▨	▨	▨
	Low Power	2.0	2.2	2.4	2.5	2.6	2.6
TMVM Area (μm²)	Area Efficient	0.2	0.4	0.6	▨	▨	▨
	Low Power	0.2	0.6	1.3	2.8	5.7	11.6

intensive approach, which does not require multiple voltage levels, is shown in Fig. 16(b) where we copy the MSB in pair of adjacent cells, and we apply the same voltage to their corresponding WLTs. The current through the MSB cell is weighted to be twice that of the current through the LSB cell.

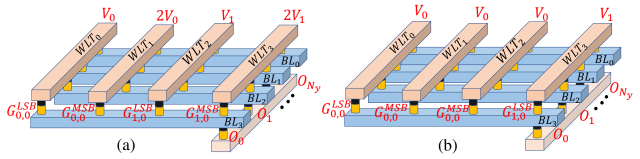

Figure 16: Two implementations with multi-bit operations: (a) area-efficient implementation, (b) low-power implementation.

We analyze the energy and area for the implementation a multi-bit TMVM using two schemes that we introduced in Section IV. We listed the results in Table VII. As we increase the number of bits for the $G_{i,j}$s, the allocated area for both implementations increases. However, while for area-efficient scheme, the area increases linearly, for the low-power scheme, the area increases exponentially. The implementation energy in the low-power scheme slightly increases with increasing the number of bits, while for the area-efficient schemes, energy increases rapidly. For the area-efficient scheme, we do not list the energy and area values beyond 3 bits, because it requires applying a large voltage level ($>$5V) within the subarray, making the implementation infeasible and unrealistic.