# DP-MAP: Towards Resistive Dot-Product Engines with Improved Precision

Necati Uysal[1], Baogang Zhang[1], Sumit Kumar Jha[2], and Rickard Ewetz[1]
[1]Department of Electrical and Computer Engineering, University of Central Florida, Orlando FL, USA
[2]Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA
necati@knights.ucf.edu,rickard.ewetz@ucf.edu

## ABSTRACT

The natural multiply and accumulate feature of memristor crossbar arrays promises unprecedented processing capabilities to resistive dot-product engines (DPEs), which can accelerate approximate matrix-vector multiplication. To overcome the challenges of low-precision devices and voltage drop over non-zero array parasitics, each matrix element can be represented using two memristors. In this paper, we propose differential pair map (DP-MAP) – the first matrix to memristor conductance mapping algorithm specifically designed for crossbars with a differential pair configuration. In contrast, previous works consider the differential pair configuration as an *afterthought*, which limits the achievable precision. The specified conductance values are next programmed to the memristor hardware using accurate closed-loop tuning. Analog computation with high precision is attained by judiciously selecting the conductance range and avoiding to explicitly decompose each matrix into a positive and negative component. Short run-time is achieved using a hierarchical optimization algorithm and two speed-up techniques. Compared with earlier studies, the computational accuracy is improved with 3.36X. This translates into signal and image compression with 61% and 94% higher quality, respectively. The simulation time of complex physical systems modeled using partial differential equations (PDEs) is reduced with 5.87X.

## 1 INTRODUCTION

Memristor crossbar arrays have attracted significant interest due to their natural ability of carrying out matrix-vector multiplication in a single time-step, which is the dominating workload for many important applications [6, 17, 18]. By applying a vector of voltages to the rows of a crossbar array, multiplication with the memristors conductance values is performed using Ohm's law and summation of currents along the columns is performed using Kirchhoff's current law, i.e., matrix-vector multiplication is performed in the analog domain. Recent hardware prototypes have shown that the analog computation is orders of magnitude more efficient than using highly optimized digital ASICs [6].

The main challenge of utilizing memristor crossbars as dot-product engines (DPEs) is that the computational accuracy may be degraded by parasitics, low-precision devices, and various sources of variations. Specifically, the accuracy may be severely impacted by voltage IR-drop over non-zero array parasitics [12], which allows currents to flow from an input to an output using multiple alternative paths within a crossbar. This is related to but not equivalent to the *sneak path problem* for memory applications [20]. Moreover, the issue cannot be easily solved using selector devices or access transistors. Errors within analog computing paradigms are particularly challenging because every error directly impacts the application-level functional correctness [23]. In contrast, variations in digital computing systems mainly introduce timing violations, which can be alleviated by scaling down the clock frequency. Consequently, a technique for mapping matrix-vector multiplication operations to DPEs is needed, which has been the focus of several studies.

Researchers have proposed hardware and software based training schemes for deep learning applications [5, 9, 13, 15]. However, training based approaches results in high power consumption [13, 15] or are limited to 64x64 crossbars [5, 9]. Prominent architectural level studies attempt to circumvent the problem by utilizing bit-slicing and reduction networks [1, 2, 4, 16]. Nevertheless, such architectural level schemes are significantly less efficient because each matrix-vector multiplication operation is decomposed across multiple crossbars and multiple time-steps. To maximize the power-efficiency, many recent algorithm level studies are focused on mapping a matrix $A$ into appropriate memristor conductance values $g$ while accounting for the voltage drop over non-zero parasitics. Many initial studies only captured the output sensing resistance and omitted the array parasitics [7, 19]. More recently, mapping algorithms that account for all non-zero parasitics have been proposed based on steepest gradient descent [12, 24] and Newtons method [8]. Although these algorithms result in improved computational accuracy, the precision is unacceptable for many real-world applications. To boost the computational accuracy, each matrix element can be represented using two (or multiple) devices [7]. When leveraging crossbars with a differential pair configuration, the target matrix $A$ is decomposed into a positive and negative component $A_+$ and $A_-$. Next, the separate components are mapped to a crossbar using one of the developed mapping algorithms. Unfortunately, it can easily be observed that considering the differential pair configuration as an *afterthought* limits the achievable computational accuracy. In this paper, we propose differential pair map (DP-MAP) – the first matrix to memristor conductance conversion algorithm specifically designed for crossbars with a differential pair configuration. The main innovations of DP-MAP are:

- The precision is improved with 3.36X by avoiding to decompose $A$ into $A_+$ and $A_-$ explicitly. Instead, the difference


*ICCAD '20, November 2–5, 2020, Virtual Event, USA*

ACM ISBN 978-1-6654-2324-3/20/11 . . $15 00
https://doi.org/10.1145/3400302.3415683

between $A$ and the matrix realized by the crossbar $A^r$ is directly minimized by iteratively updating the conductance of one memristor in each differential pair. Moreover, the conductance range is judiciously selected by minimizing the gap between a lower and upper bound on $||A - A^r||^2$.
- The overall run-time is reduced with 12X by leveraging a hierarchical algorithm and two speed-up techniques.
- The application level evaluation demonstrates that DP-MAP improves the quality of signal and image compression applications with 61% and 94%, respectively. When resistive DPEs are used to accelerate the simulation of physical systems, the simulation time is on the average reduced with 5.87X.

The remainder of the paper is organized, as follows: preliminaries in Section 2. Motivations in Section 3. The DP-MAP methodology is given in Section 4. Speed-up techniques in Section 5. The experimental evaluation in Section 6 and conclusions in Section 7.

# 2 PRELIMINARIES

## 2.1 DPE friendly Applications

The following applications are candidates for acceleration using resistive DPEs. **Signal compression:** from the time domain into the frequency domain is performed using a matrix-vector multiplication, $c = Dx$, where $D$ is the DCT matrix. $x$ and $c$ are respectively the time and frequency representation of a signal in vector form. **Image compression:** from the spatial domain into the frequency domain is performed using $C = D^T X D$, where $D$ again is the DCT matrix. $X$ and $C$ are the spatial and frequency representation of an image, respectively. **Simulation of physical systems:** is performed by modeling the system using partial differential equations (PDEs). In every time step, a sparse system of linear equations is required to be solved. State-of-the-art solvers for sparse systems of linear equations are based on Krylov subspace methods, where the dominating computation involves computing a new search vector using a matrix-vector multiplication operation.

## 2.2 Matrix-vector multiplication using DPEs

Memristor crossbars have attracted significant attention for their potential to accelerate approximate matrix-vector multiplication ($Ax = y$). A memristor crossbar consisting of wordlines and bitlines with a memristor and an access transistor in each cross-point is illustrated in Figure 1(a). Analog matrix vector multiplication, $v_{out}^T = v_{in}^T G R_s$, is performed using the natural multiply and accumulate feature of the crossbar, where $G$ is the conductance matrix of the resistive network shown in Figure 1(b). $R_s$ is the feedback resistance of the transimpedance amplifiers (TIAs). The inputs and outputs are converted between the analog and digital domain using DACs and ADCs, respectively. When a crossbar with a differential pair configuration is used, the ADCs measure the difference in the output voltage between pairs of adjacent bitlines.

When the memristor crossbar is ideal, the memristor conductance values $g$ can be obtained by mapping the matrix $A$ linearly into the programmable conductance range $[g_{min}, g_{max}]$. As the conductance values cannot be negative, A is decomposed into a positive and negative component $A_+$ and $A_-$ before the mapping. Next, the conductance values specified in software are programmed to the memristor hardware using closed-loop tuning. The accuracy

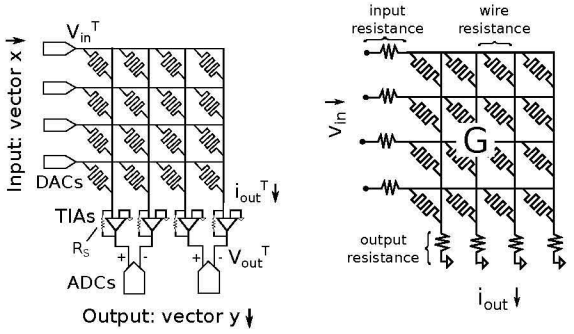


**Figure 1: (a) Memristor crossbar with a differential pair configuration for matrix-vector multiplication. (b) Resistive network and conductance matrix realized by the crossbar.**

of the tuning is called *write* accuracy, which is measured in bits. A write accuracy of 6 to 7 bits has been reported in [6, 8].

Unfortunately, it has been shown that a linear mapping results in unacceptable computational accuracy when non-zero parasitics (input, output, wire resistance) and the write accuracy is considered [8]. Consequently, a conversion algorithm is required to map an arbitrary matrix $A$ into appropriate conductance values $g$ such that the matrix realized by the crossbar $A^r$ is similar to $A$.

## 2.3 Matrix Realized by a Memristor Crossbar

In this section, it is explained how the matrix $A^r$ realized by a crossbar is computed. The relation between the input voltages and the output voltages of the resistive network in Figure 1(b) can be obtained through modified nodal analysis (MNA), as follows:

$$Y(g) \begin{bmatrix} v_{xbar} \\ v_{dac} \\ i_{out} \end{bmatrix} = \begin{bmatrix} 0 \\ v_{in} \\ 0 \end{bmatrix}, \quad (1)$$

where $Y(g)$ is matrix with dimensions $(2NM+M+n)$x$(2NM+M+N)$ that is a function of $g$. The detailed definition of $Y(g)$ is provided in [12]. $v_{xbar}$ and $v_{dac}$ are the node voltages within the crossbar and the DACs, respectively. $i_{out}$ are the output currents. Next, the conductance matrix $G$ is obtained, as follows:

$$G = SY^{-1}(g)B, \quad (2)$$

where $B = [0, I, 0]^T$ is a matrix with dimensions $(2NM+N+M)$x$(M)$ and $I$ is an $M$x$M$ identity matrix. $S = [0, 0, I]$ is a selection matrix with dimensions $(N)$x$(2NM + N + M)$. The matrix is used to select the output currents from $Y(g)^{-1}B$. Next, the matrix realized by the crossbar $A^r$ is obtained, as follows:

$$A^r = (G_+ - G_-)/\alpha, \quad (3)$$

where $\alpha$ is a scaling factor that captures the feedback resistance of the TIAs and the scaling of the DACs and ADCs. $G_+$ ($G_-$) denotes the odd/positive (even/negative) columns of $G$.

## 2.4 Problem formulation

This paper addresses the problem of mapping an arbitrary matrix $A$ into a scaling factor $\alpha$ and memristor conductance values $g$ for crossbar with a differential pair configuration. The objective is to maximize the computational accuracy while accounting for voltage

**Table 1: Comparison of DP-MAP with state-of-the-art mapping algorithms.**

| Work in | Optimization techniques | | | | Overall performance | |
|---|---|---|---|---|---|---|
| | Differential pair decomposition | Optimization of $\alpha$ | Optimization of $g$ | Speed-up of computing $G$ | Precision | Run-time |
| [7, 19] | afterthought | fixed | '-' | no | very low | **short** |
| [12] | afterthought | fixed | gradient decent | no | low | slow |
| [24] | afterthought | heuristic | gradient decent | no | low | very slow |
| [8] | afterthought | '-' | heuristic | no | medium | **short** |
| DP-MAP | **by design** | **gap minimization** | **hierarchical optimization** | **yes** | **high** | **short** |

drop over non-zero array parasitics and the write accuracy. The key is to formulate and solve a mathematical optimization problem.

**The matrix to crossbar mapping problem:** consists of specifying the conductance values $g$ and the scaling factor $\alpha$ that minimize the Frobenius norm between a target matrix $A$ and the matrix realized by the crossbar $A^r$, as follows:

$$\min_{g,\alpha} ||A^r - A||^2, \tag{4}$$

where the realized matrix $A^r$ is obtained from $g$ and $\alpha$ using Eq (2) and Eq (3). The scaling factor $\alpha$ has units siemens and can be set to any real number. The conductance values $g$ are a discrete variable with $2^b$ states between the minimum and maximum programmable conductance $g_{min}$ and $g_{max}$, respectively. $b$ is the memristor write accuracy. The $||.||^2$ operator is the square of the Frobenius norm.

## 2.5 Previous work

A comparison between DP-MAP and previous work is shown in Table 1. The mapping algorithms in [7, 19] are fast but result in low computational accuracy because they do not capture the array parasitics. The methods in [12, 24] are based on minimizing the objective $||A^r - A||^2$. However, the computational accuracy is low because the differential pair decomposition is considered as an afterthought. Moreover, the run-time of the mapping algorithms are slow. In [8], a mapping algorithm with short run-time that results in medium computational accuracy was proposed. However, the precision is only medium because the differential pair configuration was considered as an afterthought and the method is not based on minimizing $||A^r - A||^2$. DP-MAP achieves high computational accuracy while the run-time is relatively short.

# 3 DP-MAP: INSIGHTS AND MOTIVATIONS

In this section, we provide the insights and motivations for the DP-MAP framework. We analyze how to solve two precision bottlenecks and two run-time bottlenecks.

**Precision bottleneck 1:** The explicit (afterthought) decomposition of $A$ into $A_+$ and $A_-$ significantly degrades the achievable computational accuracy. This stems from that it is impossible to realize small elements at certain locations in a crossbar, which is illustrated with an example in Figure 2. The matrix in Figure 2(a) is decomposed into positive and negative component in Figure 2(b). The transpose is mapped to the crossbar in Figure 2(c). Next, consider multiplying the matrix with an input vector $[0, 0.2, 0, 0]$. The ideal output vector is $[0.2, 0]$, i.e., the net output current from bitline three and four should be close to zero. However, due to the matrix data pattern and the non-zero array parasitics, it can be observed that there is significant output current on bitline three. Moreover, the state-of-the-art techniques cannot eliminate the output current by tuning the memristor at row two and column three (2,3) to be less conductive, which is a result of the explicit decomposition. DP-MAP reduces the net output current by tuning the memristor of the negative component at (2,4), which is shown in Figure 2(d).

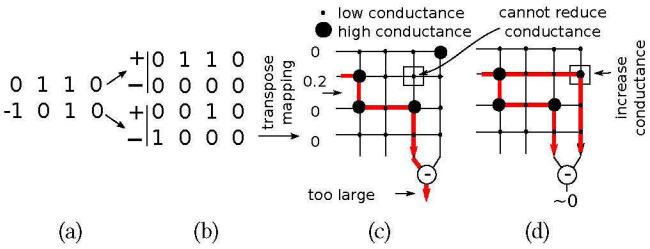


**Figure 2: (a) 2x4 matrix. (b) Differential pair decomposition into a 4x4 matrix. (c) Transpose mapping to crossbar. The net output current from column three and four is too large. The current cannot be reduced by tuning the memristor at (2,3) to be less conductive. (d) DP-MAP reduces the net current by tuning the memristor at (2,4) to be more conductive.**

**Precision bottleneck 2:** The parameter $\alpha$ implicitly regulates the utilization of the conductance range. A large $\alpha$ implies that the entire programmable conductance range is used, which minimizes the *write errors* introduced by the limited write precision. However, a large $\alpha$ introduces significant voltage IR-drop over the non-zero parasitics, which may result in large *value range errors*, i.e., that conductance values are attempted to be specified outside $[g_{min}, g_{max}]$. In [24], it was observed that the errors in Eq (4) were close to minimal when the write errors ($\epsilon_{write}$) were equal to the value range errors ($\epsilon_{value}$). However, the parameter $\alpha$ was optimized using a heuristic. DP-MAP optimizes $\alpha$ judiciously through a binary search that minimizes the gap between a lower and upper bound on the errors in Eq (4).

**Run-time bottleneck 1 and 2:** Intuitively, the objective $||A^r - A||^2$ is required to be optimized in order to achieve high computational accuracy. An overview of the flow of the mapping algorithm in [24] and DP-MAP is shown in Figure 4. The mapping algorithm in [24] decouples the optimization of $\alpha$ and $g$. The left loop is used to minimize $||A^r - A||^2$ by optimizing $\alpha$. Given an $\alpha$ value, the right loop is used to minimize $||A^r - A||^2$ by optimizing $g$ using steepest gradient descent. The figure shows that the right loop is performed *many* times and each iteration takes *long* run-time. The run-time is dominated by the computation of the conductance matrix $G$ using Eq (2). Consequently, the overall run-time is **many*long**.

DP-MAP overcomes the run-time bottleneck using a i) hierarchical algorithm and ii) speed-up techniques for computing $G$. The

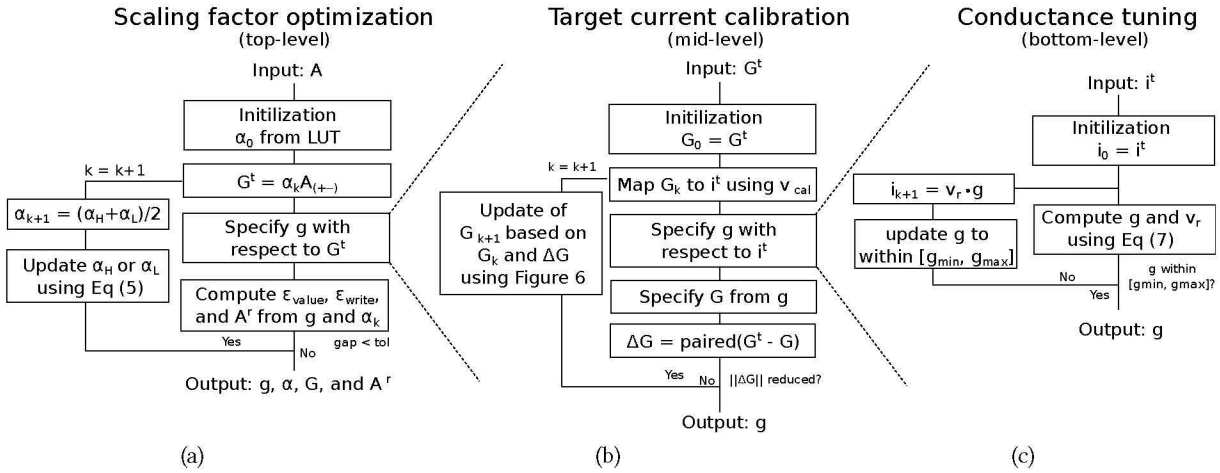


**Figure 3: Flow of DP-MAP.**

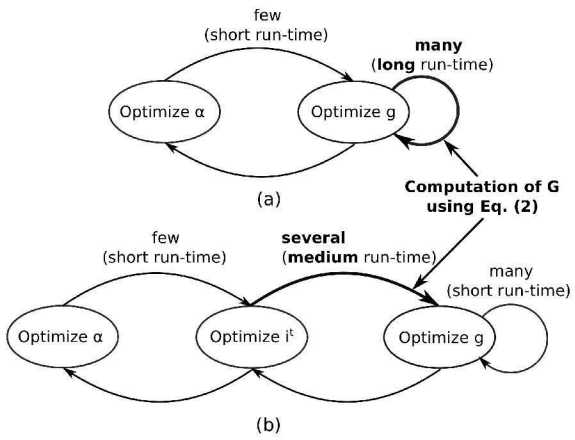


**Figure 4: (a) Flow in [24] and (b) flow of DP-MAP framework.**

hierarchical algorithm reduces the number of times the conductance matrix $G$ is required to be calculated. Instead of optimizing the conductance values $g$ to minimize $||A^r - A||^2$, the conductance values are optimized to deliver a target current $i^t$ through each memristor device. Next, given $\alpha$, the target currents $i^t$ are calibrated with respect to the difference between $A$ and $A^r$. This results in that the number of times $G$ is required to be computed is reduced from **many** to **several** in Figure 4. Moreover, DP-MAP speeds-up the calculation $G$ itself. This is achieved by avoiding a significant amount of redundant computation by reordering linear algebra operations. This explains why the computation of $G$ in Figure 4(b) is labeled with **medium** run-time. Consequently, the overall run-time is reduced from **many*long** to **several*medium**.

## 4 THE DP-MAP METHODOLOGY

An overview of the proposed flow of DP-MAP is shown in Figure 3. The input is a matrix $A$. The output is the conductance values $g$, the scaling factor $\alpha$, the conductance matrix $G$, and the realized matrix $A^r$. The DP-MAP algorithm consists of three hierarchical steps, i.e., a top-level, a mid-level, and a bottom-level step.

The top-level optimization is called "*Scaling factor optimization*" and is described in Section 4.1. The objective is to determine the scaling factor $\alpha$ that minimizes $||A^r - A||$. The top-level step uses the mid-level step to determine the conductance values $g$ with respect to an $\alpha$ value. The conductance values $g$ directly define $A^r$ using $\alpha$, Eq (2), and Eq (3). The mid-level step is called "*Target current calibration*" and it is invoked from the top-level. The objective is to determine the target current $i^t$ delivered through each memristor device with respect to the target conductance matrix $G^t = \alpha A_{(+-)}$ and a calibration vector $v_{cal}$, where $A_{(+-)}$ is a matrix with the rows in $A_+$ and $A_-$ interleaved as shown in (a) and (b) of Figure 2. The specifics of the target current calibration are given in Section 4.2. The bottom-level step is called "*Conductance tuning*" and it is invoked from the mid-level. The objective is to determine the conductance values $g$ that deliver the target currents $i^t$ through the memristors with respect $v_{cal}$, which is detailed in Section 4.3.

### 4.1 Scaling factor optimization

The scaling factor optimization is shown in Figure 3(a). The objective is to determine the scaling factor $\alpha$ that minimizes Eq (4) given the matrix $A$. The trend of the value range, write, and total errors with respect to the scaling factor $\alpha$ is shown in Figure 5(a). The value range errors ($\epsilon_{value}$) are defined to be $||A^r - A||$ while treating $g$ as a continuous variable between $[g_{min}, g_{max}]$. The total errors ($\epsilon_{total}$) are defined to be $||A^r - A||$ after $g$ has been quantized (to $2^b$ states) to capture the write errors. The write errors ($\epsilon_{write}$) are equal to $\epsilon_{total} - \epsilon_{value}$. Based on the figure, it can be observed that the total errors are minimized when $\epsilon_{write} \approx \epsilon_{value}$. Consequently, the optimal value of $\alpha$ can be determined using a binary search. First, we guess an $\alpha_L$ and $\alpha_H$ that satisfy $\epsilon_{write} < \epsilon_{value}$ and $\epsilon_{write} > \epsilon_{value}$, respectively. Next, the mid-level target current calibration is invoked to determine $\epsilon_{write}$ and $\epsilon_{value}$ with respect to $\alpha_k = (\alpha_L + \alpha_H)/2$. Based on the ratio of $\epsilon_{value}$ to $\epsilon_{write}$, $\alpha_L$ or $\alpha_H$ is updated, follows:

$$\begin{aligned} \alpha_L &= \alpha_k, \quad \epsilon_{write} < \epsilon_{value}, \\ \alpha_H &= \alpha_k, \quad \epsilon_{write} > \epsilon_{value}. \end{aligned} \tag{5}$$

The currently best observed solution is naturally an upper bound on the total errors. Similarly, a lower bound on the total errors is the value range errors for $\alpha_L$ plus the write errors for $\alpha_H$. In

Figure 5(b), we show that the gap between the lower and upper bound is reduced in each iteration of the binary search. The top-level flow is terminated when the gap between the lower and the upper bound is smaller than $\epsilon_{tol}$.

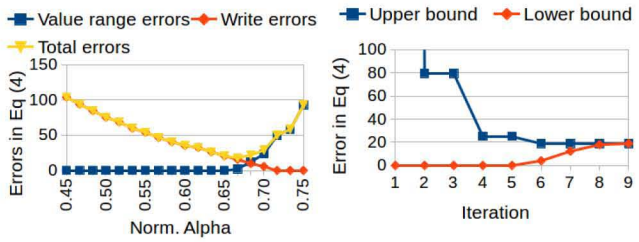


**Figure 5: (a) Write, value, and total errors. (b) Narrowing of gap between the lower and upper bound on the total errors.**

## 4.2 Target current calibration

In this section, the target currents $i^t$ delivered through the memristor devices are determined with respect to a target conductance matrix $G^t$ and a calibration vector $v_{cal}$, which is shown in Figure 3(b). Let an ideal crossbar have zero input, output, and wire parasitics, i.e., the voltages applied to the wordlines are applied across each memristor device. Let the conductance matrix of the ideal crossbar be denoted $G_k$. $G_0$ is initialized to $G^t$. The target currents $i^t$ are obtained by applying a calibration vector $v_{cal}$ to the ideal crossbar programmed with $G_k$. An arbitrary uniform vector is used as $v_{cal}$, i.e., all the entries in the vector have the same magnitude. Next, the bottom-level conductance tuning step is used to determine the conductance values $g$ that deliver the target currents $i^t$ through the memristor devices in a crossbar with parasitics. Using the determined conductance values $g$, the realized conductance matrix $G$ is computed using Eq (2) and the error $\triangle G$ is computed as $\triangle G = paired(G^t - G) = (G^t_+ - G^t_-) - (G_+ - G_-)$, i.e., the positive minus the negative components of both $G^t$ and $G$. If the error $||\triangle G||^2$ is reduced from the previous iteration, $G_k$ is updated to $G_{k+1}$. Otherwise, the mid-level step is terminated and $g$ is returned to the top-level. The update of $G_k$ to $G_{k+1}$ is performed by first setting $G_{k+1}$ equal to $G_k$. Next, half of the elements in $G_{k+1}$ are updated based on $\triangle G$ using the flow in Figure 6. The algorithm ensures that i) *one memristor in each pair has the conductance* $g_{min}$ and ii) *the conductance of only one memristor in each pair is updated in each iteration.* Next, the flow is repeated with the new ideal conductance matrix $G_{k+1}$. Experimentally, we have observed that the flow quickly converges to a solution with small errors. Despite that it is an ideal conductance matrix that is updated, the step is called target current calibration because $G_k$ and $v_{cal}$ uniquely define $i^t$.

## 4.3 Conductance tuning

In this section, the conductance values $g$ are specified to deliver the target currents $i^t$ through the memristor devices, which is shown in Figure 3(c). The conductance values are determined using an iterative approach because it may be impossible to deliver the target currents $i^t$ through the memristors while the conductance values $g$ are between $[g_{min}, g_{max}]$. Let $i_k$ be the target currents in iteration $k$ and $i_0$ is initialized to $i^t$. The conductance values $g$ are specified by reformulating the system of linear equation in Eq (1) to capture

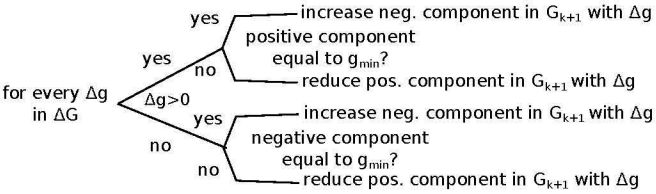


**Figure 6: Flow for updating $G_k$ to $G_{k+1}$ based on $\triangle G$. $\triangle g$ is an element in $\triangle G$. The update ensures that one memristor in each pair has the minimum conductance and that only one memristor in each pair is updated in each iteration.**

the currents flowing through the memristors using the right hand side, as follows:

$$\overline{Y}\begin{bmatrix} v_{xbar} \\ v_{dac} \\ i_{out} \end{bmatrix} = \begin{bmatrix} -i_k \\ i_k \\ v_{cal} \\ 0 \end{bmatrix}, \tag{6}$$

where $i_k$ are the target currents; $v_{cal}$ is the input calibration vector used to determine the target currents. $\overline{Y}$ is a matrix with the same dimensions as $Y(g)$ in Eq (1) but it is not a function of $g$. Based on Eq (6), the conductance values $g$ and the voltage across each resistive device $v_r$ are determined, as follows:

$$\begin{bmatrix} g \\ v_r \end{bmatrix} = D\overline{Y}^{-1}\begin{bmatrix} -i_k \\ i_k \\ v_{cal} \\ 0 \end{bmatrix} \tag{7}$$

where $D$ is a matrix with dimensions $(2MN)$x$(2MN + N + M)$. The matrix $D$ is a function of $i_k$. Note that $\overline{Y}^{-1}[i_k, -i_k, v_{cal}, 0]^T$ can be computed extremely efficiently using LU factorization. This stems from that the target currents decouple each of the wordlines and bitlines in $\overline{Y}$. Consequently, $\overline{Y}$ is a block-wise matrix with $(N + M)$ subblocks, which the LU factorization can be solved for independently. If the obtained conductance values $g$ satisfy the lower and upper bounds, the flow converges and the $g$ is returned. In this paper, we propose to update the target currents $i_k$ to $i_{k+1}$ such that it is expected the conductance values obtained using Eq (7) are within the programmable conductance range. The update is performed by setting conductance values in $g$ that are outside $g_{min}$ and $g_{max}$ to $g_{min}$ and $g_{max}$, respectively. Next, the voltage across each memristor in the crossbar $(v_r)$ is assumed to be fixed and the new target currents $i_{k+1}$ are obtained as $i_{k+1} = v_r \cdot g$. Here, '$\cdot$' denotes an element-wise multiplication of the vectors $g$ and $v_r$.

# 5 SPEED-UP OF DP-MAP

In this section, we propose a technique to further speed-up the run-time of DP-MAP. We focus on the computation of $G$ because it stands for between 90% to 99% of the total run-time. The conductance matrix $G$ is computed by solving Eq (2) using sparse LU factorization with pivoting, as follows:

$$G = SQU^{-1}L^{-1}PB \tag{8}$$

where $P$ and $Q$ are permutation matrices. $L$ and $U$ are respectively a lower and upper triangular matrices. The expression is evaluated from right-to-left. First, $T_1 = L^{-1}(PB)$ is computed using forward

substitution. Next, $T_2 = U^{-1}T_1$ is solved using backward substitution. Lastly, $G$ is computed using $G = SQT_2$.

We propose to reduce the run-time by (i) permuting the order linear algebra operations and (ii) avoiding redundant computation. The first speed-up is obtained by computing $SQT_2$ from left-to-right instead of from right-to-left. This reduces the run-time significantly because $S$ only has $N$ non-zero entries. The matrix dimensions are also smaller than of $T_2$. The reordering results in that $Q$ permutes the elements in $S$ to form a new selection matrix $\widetilde{S} = SQ$. $\widetilde{S}$ is used to select $N$ rows in $T_2$. Next, we turn our attention to avoiding to perform redundant computation by exploiting the structure of $\widetilde{S}$. $\widetilde{S}$ is an $(N)$x$(2NM + N + M)$ matrix with only $N$ non-zero elements. Let $r$ be the number of columns starting from the first in $\widetilde{S}$ that only contain zeros. It can be observed that the top $r$ rows of $T_2$ are not used, which is illustrated in Figure 7(a). Consequently, the top $r$ rows in $T_2$ are not required to be computed when solving $T_2 = U^{-1}T_1$, which is illustrated in Figure 7(b). Therefore, the matrices $\widetilde{S}$, $U$, $T_1$, and $T_2$ can be reduced to $\widetilde{S}_r$ $U_r$, $T_{1r}$, and $T_{2r}$ by removing rows and columns as illustrated in (a) and (b) of Figure 7. Next, $G$ can be computed as $G = \widetilde{S}_r(U_r^{-1}T_{1r})$.

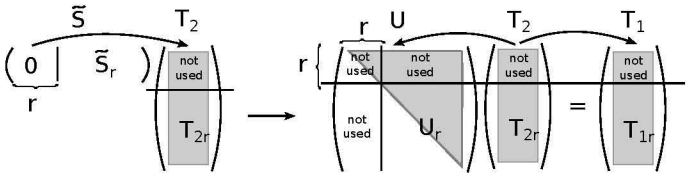


**Figure 7: (a) Exploiting the structure of $S$ to reduce $T_2$ into $T_{2r}$. (b) Exploiting the structure of $T_{2r}$ to reduce $U$ and $T_1$ into $U_r$ and $T_{1r}$, respectively.**

In Figure 8(a), we show the breakdown of the run-time with respect to the different steps of $G$ for a crossbar with dimensions 256x256. The figure shows that the run-time of LU decomposition and $T_1 = L^{-1}PB$ are identical w/o the speed-up technique. The run-time of computing $G = SQT_2$ (or $\widetilde{S}_rQT_2$) is reduced close to zero. The run-time of $T_2 = U^{-1}T_1$ (or $T_{2r} = U^{-1}T_{1r}$) is slightly reduced. The impact of the techniques on the overall run-time of computing $G$ is shown in Figure 8(b). It can be observed that the run time is reduced with 30% to 35% based on the crossbar size.

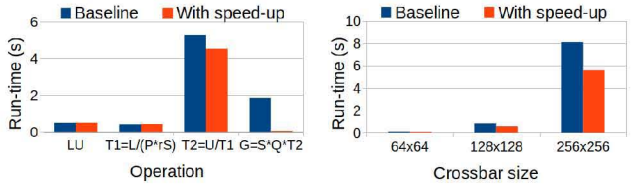


**Figure 8: (a) Run-time for computing $G$ w/o speed-up for a 256x256 crossbar. (b) Speed-up of Eq (2) w.r.t the crossbar size.**

## 6 EXPERIMENTAL EVALUATION

The experimental results are obtained using a quad core 3.4 GHz Linux machine with 32GB of memory. DP-MAP is implemented in MATLAB. Memristor crossbars with dimensions 64x64 to 256x256 are used in the evaluation. The crossbars have a wire resistance $r_w$=1Ω, and an input and output resistance are 100Ω [6, 8]. The programmable conductance range of the memristors is set to $2k\Omega$ to $3M\Omega$ [8]. The maximum input voltage is $0.2V$. The bit-accuracy $b$ of the closed-loop programming of the memristors is set to 6-bits, which is modeled by quantizing the conductance values $g$ to $2^b$ states between the lower and upper bound, respectively. The DAC/ADC domain interfaces are set to 8-bits. The computational accuracy of the analog matrix-vector multiplication performed by a DPE is evaluated using the conductance matrix obtained from Eq (2), where $g$ and $\alpha$ are obtained using DP-MAP.

Using the outlined setup, DP-MAP is compared with the previous techniques in terms of matrix-vector multiplication in Section 6.1. The capability of DP-MAP to accelerate signal/image compression and the simulation of physical systems is evaluated in Section 6.2.

### 6.1 Evaluation of matrix-vector multiplication

The performance of DP-MAP is compared with previous works in terms of matrix-vector multiplication in Figure 9. In particular, we compare DP-MAP with a linear mapping, the mapping in [24], and the mapping in [8]. The mapping algorithms are evaluated in terms of normalized errors in Eq (4), maximum output error, and run-time. The reported results are the averages with respect to multiple matrices and several input vectors.

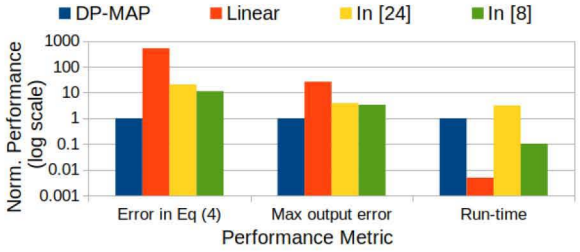


**Figure 9: Comparison of different mapping techniques in terms of normalized errors in Eq (4), maximum output error, and mapping run-time on a 128x128 crossbar.**

The figure shows that DP-MAP archives 521X, 21X, 11X smaller errors in Eq (4). Therefore, it is not surprising that the maximum output errors are 26X, 3.87X, and 3.36X smaller. The improvement in computational accuracy stems from that the use of two memristors in each differential pair is considered in the design of the algorithm (see pairwise mapping in Figure 6) and judiciously selecting the conductance range. The run-time is 200X and 10X slower than linear mapping and the mapping in [8], respectively. However, we deem the run-time penalty very acceptable for a 26X and 3.36X improvement in computational accuracy, respectively.

Next, we further evaluate the performance of DP-MAP in Figure 10. In Figure 10(a), we show the equivalent digital accuracy for crossbars of different sizes while using both a single and two memristors per matrix element. The equivalent digital accuracy is obtained through regression with respect to the maximum error of a fixed-point multiplier (where the matrix is represented using one to eights bits). The figure shows that the equivalent digital accuracy is gracefully degraded from 6.18 to 4.91 bits when the dimensions of the crossbar is increased from 64x64 to 256x256. The

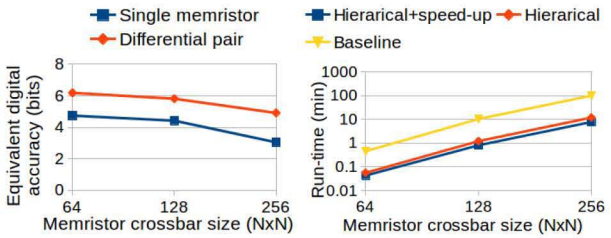


**Figure 10: (a) Equivalent digital bit-accuracy using differential pair and a single memristor per matrix element. (b) Run-time with different configurations.**

upper bound is seven bits, i.e., two memristors with six bits each. It is very promising that the differential pair configuration improves the digital bit-accuracy with more than one bit on the average, which validates the effectiveness of the mapping algorithm. The improvement is expected due to the occurrence and compensation of non-desired currents as illustrated in (c) and (d) of Figure 2. The run-time of DP-MAP for different crossbar sizes, w/o the hierarchical algorithm, and the speed-up technique is shown in Figure 10(b). Compared with a baseline version where the conductance values are tuned using steepest gradient descent, the hierarchical approach reduces the run-time with about an order of magnitude. The speed-up techniques further reduces the run-time with 32% on the average. The run-time of DP-MAP is 0.04 min, 0.84 min, and 7.84 min for crossbars of dimension 64x64, 128x128, and 256x256, respectively.

## 6.2 Application level evaluation

In this section, we evaluate the impact of using DP-MAP to program the resistive DPEs used to accelerate signal compression, image compression, and simulation of physical systems. We only compare DP-MAP with the mapping algorithm in [8] because it is superior to linear mapping and the mapping in [24].

*6.2.1 Signal compression.* In this section, we evaluate the application level performance when a resistive DPE is used to perform signal compression using 1D-DCT. The compression is performed by first programming a 128x256 memristor crossbar with the DCT matrix $D$ [11]. Next, samples of the signals in Table 2 are converted from the time domain to the frequency domain using a matrix-vector multiplication operation. To improve the degree of compression, frequency components are discarded such that only 99% of the signal energy is preserved [11]. Next, decompression is performed using digital hardware. The compression is evaluated by comparing the original signal with the decompressed signal. The evaluation is performed in terms of mean square error (MSE) and average bits per sample (BPS), respectively.

**Table 2: Properties of input signals and input images [14].**

| Id | Equation | Samples | BPS (num) |
|---|---|---|---|
| 1 | $\sin(x)$ | 128 | 8 |
| 2 | $\sin(2x)$ | 128 | 8 |
| 3 | $\sin(2x) + \cos(x)$ | 128 | 8 |
| 4 | $\sin(x) + \sin(2x) + \sin(3x)$ | 128 | 8 |
| 5 | $\sin(2x) + 0.2\sin(x)$ | 128 | 8 |

| Image (name) | Rows (num) | Columns (num) | BPP (num) |
|---|---|---|---|
| bird | 320 | 480 | 24 |
| lizard | 320 | 480 | 24 |
| boat | 320 | 480 | 24 |
| dolphine | 320 | 480 | 24 |
| geese | 320 | 480 | 24 |

We compare the state-of-the-art method in [8] with DP-MAP and DP-MAP+Q in Table 3. DP-MAP+Q is DP-MAP extended with quantization of the frequency components, where the quantization level is configured such that the signal quality in terms of MSE is similar to in [8]. Quantization is a standard compression technique within signal and image processing [11]. The mapping performance is evaluated using errors in Eq (4) and run-time.

**Table 3: Evaluation of signal compression in terms of mapping and application-level performance.**

| Signal (id) | Work | Mapping performance | | Application performance | |
|---|---|---|---|---|---|
| | | Error in Eq (4) | Run-time (s) | Quality (MSE) | Compression (BPS) |
| 1 | In [8] | 1206.0 | 10.2 | 199.7 | 1.4 |
| | DP-MAP | 19.9 | 105.3 | 92.4 | 1.4 |
| | DP-MAP+Q | 19.9 | 105.3 | 144.9 | 0.9 |
| 2 | In [8] | 1206.0 | 10.2 | 242.3 | 1.6 |
| | DP-MAP | 19.9 | 105.3 | 84.5 | 1.6 |
| | DP-MAP+Q | 19.9 | 105.3 | 150.2 | 1.1 |
| 3 | In [8] | 1206.0 | 10.2 | 211.8 | 1.4 |
| | DP-MAP | 19.9 | 105.3 | 100.3 | 1.4 |
| | DP-MAP+Q | 1206.0 | 10.2 | 143.7 | 0.9 |
| 4 | In [8] | 1206.0 | 10.2 | 122.6 | 2.7 |
| | DP-MAP | 19.9 | 105.3 | 39.5 | 2.8 |
| | DP-MAP+Q | 1206.0 | 10.2 | 219.8 | 1.2 |
| 5 | In [8] | 1206.0 | 10.2 | 165.9 | 1.5 |
| | DP-MAP | 19.9 | 105.3 | 60.5 | 1.6 |
| | DP-MAP+Q | 0.0 | 0 | 119.8 | 1.1 |
| **Norm.** | In [8] | 1.00 | **1.00** | 1.00 | 1.00 |
| | DP-MAP | **0.02** | 10.32 | **0.39** | 1.04 |
| | DP-MAP+Q | **0.02** | 10.32 | 0.91 | **0.63** |

Compared within [8], it can be observed that DP-MAP reduces the errors in Eq (4) with 98% at the expense of increasing the mapping time from a few seconds to just under two minutes. Note that the improvement in errors in Eq (4) is highly dependent on the data pattern of the matrix. On the application level, DP-MAP reduces the MSE with 61% while achieving a 4% worse compression. When quantization is applied to improve the degree of compression, i.e., DP-MAP+Q, the image quality is still 9% higher while reducing BPS with 37% on the average. Compared with a digital ASIC [6], the speed-efficiency product is improved with 500X.

*6.2.2 Image compression.* In this section, we evaluate the application level performance when a resistive DPE is used to perform image compression using 2D-DCT. A 64x128 crossbar arrays is first programmed with a reconstructed DCT matrix as described in [21, 22]. Next, the DPE is used to convert the images in Table 2 from the spatial domain to the frequency domain. Decompression of the image is performed using digital hardware. We evaluate the compression by comparing the original images with the decompressed images. The evaluation is performed using mean square errors (MSE) and bits per pixel (BPP). We again compare programming the crossbar using the state-of-the-art technique in [8] with DP-MAP and DP-MAP+Q. The mapping performance is evaluated in terms of errors in Eq (4) and run-time.

Compared with the mapping in [8], DP-MAP and DP-MAP+Q reduces the errors in Eq (4) with 99% at the expense of increasing the mapping time from a few seconds to just under a minute. The smaller errors translate into that the MSE is reduced with 94% while the BPP is 10% higher. We speculate that the higher BPP stems from

**Table 4: Evaluation of image compression in terms of mapping and application level performance.**

| Image (name) | Work | Mapping Performance | | Application Performance | |
|---|---|---|---|---|---|
| | | Error in Eq (4) | Run-time (s) | Quality (MSE) | Compression (BPP) |
| bird | In [8] | 49.62 | 0.05 | 10.1 | 12.1 |
| | DP-MAP | 0.27 | 0.57 | 0.9 | 14.0 |
| | DP-MAP+Q | 0.27 | 0.57 | 10.1 | 5.9 |
| lizard | In [8] | 49.62 | 0.05 | 11.1 | 12.6 |
| | DP-MAP | 0.27 | 0.57 | 0.8 | 14.3 |
| | DP-MAP+Q | 0.27 | 0.57 | 10.8 | 6.4 |
| boat | In [8] | 49.62 | 0.05 | 15.8 | 13.8 |
| | DP-MAP | 0.27 | 0.57 | 0.7 | 15.1 |
| | DP-MAP+Q | 0.27 | 0.57 | 12.1 | 7.5 |
| dolphine | In [8] | 49.62 | 0.05 | 8.7 | 11.5 |
| | DP-MAP | 0.27 | 0.57 | 0.4 | 12.2 |
| | DP-MAP+Q | 0.27 | 0.57 | 11.2 | 6.6 |
| geese | In [8] | 49.62 | 0.05 | 9.3 | 9.3 |
| | DP-MAP | 0.27 | 0.57 | 0.4 | 10.0 |
| | DP-MAP+Q | 0.27 | 0.57 | 7.8 | 5.1 |
| **Norm.** | In [8] | 1.00 | **1.00** | 1.00 | 1.00 |
| | DP-MAP | **0.01** | 12.59 | **0.06** | 1.10 |
| | DP-MAP+Q | **0.01** | 12.59 | 0.97 | **0.53** |

that DP-MAP more accurately captures frequency components with low magnitude. When quantization is applied, it can be observed that DP-MAP+Q is capable of simultaneously improving image quality with 3% and compression with 47%. Compared with performing the compression using a digital ASIC the speed-power efficiency is improved with 300X [6].

*6.2.3 Simulation of physical systems.* When simulating a physical systems modeled using PDEs, a sparse system of linear equations is required to be solved in each time-step. We follow the approach of solving linear systems of equations using mixed-precision in-memory computing paradigm detailed in [10]. The method is based on utilizing a flow with an inner and outer loop. In the inner loop, an approximate solution to $Ax = r$ is determined using the conjugate gradient method while leveraging low-precision resistive DPEs to compute new search vectors using matrix-vector multiplication operations. $r$ is initialized to $b$ in the first iteration of the outer loop. Next, in the outer loop, the right hand side $r$ refined based on the residual $r = Ax - b$ using high precision digital hardware. Interestingly, the final precision is only limited by the precision of the digital hardware and not the resisitve DPEs. The advantage over digital implementations is that the number of high precision digital matrix-vector multiplications is greatly reduced. We compare DP-MAP with the state-of-the-art mapping in [8] and a digital approach in Table 6 using the five systems in Table 5. We utilize the kernel to crossbar mapping in [4] to decompose the sparse matrices to crossbars of arrays of different dimensions. The tolerance on the relative residual is set to $10^{-15}$.

The table shows that DP-MAP reduces the errors in Eq (4) with 2.13X compared with in [8]. The reductions in errors are smaller than for the other applications because the matrices are sparse and easier to map. However, the run-time penalty of the mapping is also smaller. On the application level, it can be observed that only four of the five systems converge to a solution with an error bellow the tolerance using the mapping in [8]. Moreover, the convergence of the second system of linear equations is extremely slow. Consequently, it is advantageous to utilize DP-MAP to perform the mapping. Compared with a digital approach, the number of iterations is reduced with 5.87$X$, i.e., the number of digital high precision matrix-vector multiplication operations are reduced with 5.87$X$. This translates into that the latency and energy is improved with a similar factor because the computation in the inner-loop using the DPEs is almost negligible compared with the outer-loop [10].

**Table 5: Properties of five physical systems modeled using PDEs from four different application domains [3].**

| Id | Domain | Matrix Properties | | | Crossbar dimensions | | |
|---|---|---|---|---|---|---|---|
| | | Rows (num) | Cols (num) | Non-zero (num) | 32x32 (num) | 64x64 (num) | 128x128 (num) |
| 1 | economic | 2048 | 2048 | 11760 | 35 | 30 | 33 |
| 2 | acoustics | 2048 | 2048 | 35012 | 84 | 35 | 64 |
| 3 | materials | 2048 | 2048 | 40722 | 123 | 21 | 122 |
| 4 | structural | 2048 | 2048 | 12074 | 102 | 15 | 58 |
| 5 | structural | 2048 | 2048 | 12142 | 89 | 15 | 39 |

**Table 6: Evaluation of simulation of physical systems in terms of mapping and application performance.**

| Id | Work | Mapping performance | | Application performance | | |
|---|---|---|---|---|---|---|
| | | Error in Eq (4) | Run-time (min) | Converg (yes/no) | Relres ($10^{-15}$) | Iter (num) |
| 1 | Digital | '-' | '-' | yes | 0.21 | 205 |
| | In [8] | 3.4 | 3.3 | yes | 0.65 | 51 |
| | DP-MAP | 2.6 | 6.9 | yes | 0.11 | 28 |
| 2 | Digital | '-' | '-' | yes | 0.14 | 113 |
| | In [8] | 78.4 | 8.7 | yes | 0.85 | 201 |
| | DP-MAP | 30.5 | 23.7 | yes | 0.78 | 28 |
| 3 | Digital | '-' | '-' | yes | 0.72 | 79 |
| | In [8] | 97.5 | 19.0 | no | $10^{13}$ | '-' |
| | DP-MAP | 48.3 | 42.3 | yes | 0.92 | 35 |
| 4 | Digital | '-' | '-' | yes | 0.94 | 61 |
| | In [8] | 37.6 | 9.5 | yes | 0.07 | 6 |
| | DP-MAP | 16.3 | 20.7 | yes | 0.05 | 6 |
| 5 | Digital | '-' | '-' | yes | 0.68 | 245 |
| | In [8] | 39.1 | 9.5 | yes | 0.50 | 44 |
| | DP-MAP | 16.1 | 21.2 | yes | 0.89 | 44 |
| **Norm.** | Digital | '-' | '-' | **5/5** | | 5.87 |
| | In [8] | 2.13 | **0.44** | 4/5 | | 2.23 |
| | DP-MAP | **1.00** | 1.00 | **5/5** | | **1.00** |

## 7 SUMMARY AND FUTURE WORK

In this paper, we proposed DP-MAP for mapping arbitrary matrices to memristor crossbars with a differential pair configuration. Compared with state-of-the-art, the computational accuracy is improved with up to 3.36X. This translates into significantly improved performance for applications as signal compression, image compression, and simulation of physical systems. In the future, we plan to further improve the run-time and robustness to variations.

## ACKNOWLEDGMENTS

This work was in part supported by ONR award N00014-20-1-2251, DARPA co-operative agreement HR00112020002, NSF awards 1822976, 1908471, 1755825 and Cyber-Florida award 3910-1011-00.

## REFERENCES


[1] M. N. Bojnordi and E. Ipek. 2016. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning *(HPCA'16)*. 1–13.

[2] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. 2016. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory *(ISCA'16)*. 27–39. https://doi.org/10.1109/ISCA.2016.13

[3] Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* 38, 1, Article 1 (2011), 25 pages. https://doi.org/10.1145/2049662.2049663

[4] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek. 2018. Enabling Scientific Computing on Memristive Accelerators *(ISCA'18)*. 367–382.

[5] Zhezhi He, Jie Lin, Rickard Ewetz, Jiann-Shiun Yuan, and Deliang Fan. 2019. Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping *(DAC'19)*. 57:1–57:6.

[6] Miao Hu et al. 2018. Memristor-Based Analog Computation and Neural Network Classification with a DPE. *Adv. Materials* 30 (2018).

[7] Miao Hu, Hai Li, Yiran Chen, Qing wu, Garrett Rose, and Richard W Linderman. 2014. Memristor Crossbar-Based Neuromorphic Computing System: A Case Study. *NN. and Learning Sys., IEEE Tran. on* 25 (2014), 1864–1878.

[8] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams. 2016. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication *(DAC'16)*. 1–6.

[9] Shubham Jain, Abhronil Sengupta, Kaushik Roy, and Anand Raghunathan. 2018. RxNN: Framework for evaluating and training Neural Networks on Resistive Crossbars. *CoRR* abs/1809.00072 (2018). arXiv:1809.00072 http://arxiv.org/abs/1809.00072

[10] Manuel Le Gallo, Abu Sebastian, Roland Mathis, Matteo Manica, Heiner Giefers, Tomas Tuma, Costas Bekas, Alessandro Curioni, and Evangelos Eleftheriou. 2018. Mixed-precision in-memory computing. *Nature Electronics* 1 (2018), 246–253. https://doi.org/10.1038/s41928-018-0054-8

[11] Can Li et al. 2017. Analogue signal and image processing with large memristor crossbars. *Nature Electronics* (2017).

[12] B. Liu, H. Li, Y. Chen, X. Li, T. Huang, Q. Wu, and M. Barnell. 2014. Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems *(ICCAD'14)*. 63–70.

[13] B. Liu, Hai Li, Yiran Chen, Xin Li, Qing Wu, and Tingwen Huang. 2015. Vortex: Variation-aware training for memristor X-bar *(DAC'15)*. 1–6. https://doi.org/10.1145/2744769.2744930

[14] D. Martin, C. Fowlkes, D. Tal, and J. Malik. [n. d.]. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *ICCV'01*.

[15] Mirko Prezioso, Farnood Merrikh-Bayat, Brian Hoskins, Gina C. Adam, Konstantin K. Likharev, and Dmitri B. Strukov. 2014. Training and Operation of an Integrated Neuromorphic Network Based on Metal-Oxide Memristors. *CoRR* abs/1412.0611 (2014). arXiv:1412.0611 http://arxiv.org/abs/1412.0611

[16] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars *(ISCA'16)*. 14–26.

[17] R. S. Williams. 2017. What's Next? [The end of Moore's law]. *Computing in Science Engineering* 19, 2 (2017), 7–13. https://doi.org/10.1109/MCSE.2017.31

[18] Wm. A. Wulf and Sally A. McKee. 1995. Hitting the Memory Wall: Implications of the Obvious. *SIGARCH Comput. Archit. News* 23, 1 (1995), 20–24. https://doi.org/10.1145/216585.216588

[19] Lixue Xia, Peng Gu, Boxun Li, Tianqi Tang, Xiling Yin, Wenqin Huangfu, Shimeng Yu, Yu Cao, Yu Wang, and Huazhong Yang. 2016. Technological Exploration of RRAM Crossbar Array for Matrix-Vector Multiplication. *Journal of Computer Science and Technology* 31, 1 (2016), 3–19.

[20] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie. 2015. Overcoming the challenges of crossbar resistive memory architectures *(HPCA'15)*. 476–488.

[21] B. Zhang, N. Uysal, and R. Ewetz. 2020. Computational Restructuring: Rethinking Image Compression using Resistive Crossbar Arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2020), 1–1.

[22] Baogang Zhang, Necati Uysal, and Rickard Ewetz. 2020. Computational Restructuring: Rethinking Image Processing using Memristor Crossbar Arrays *(DATE'20)*.

[23] B. Zhang, N. Uysal, D. Fan, and R. Ewetz. 2019. Handling Stuck-at-fault Defects using Matrix Transformation for Robust Inference of DNNs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2019), 1–1.

[24] Baogang Zhang, Necati Uysal, Deliang Fan, and Rickard Ewetz. 2020. Representable Matrices: Enabling High Accuracy Analog Computation for Inference of DNNs using Memristors *(ASP-DAC'20)*. 538–543.