micromachines

MDPI

*Review*

# Compute-in-Memory for Numerical Computations

**Dongyan Zhao [1], Yubo Wang [1], Jin Shao [1], Yanning Chen [1,2], Zhiwang Guo [3,*], Cheng Pan [1], Guangzhi Dong [2], Min Zhou [1], Fengxia Wu [2], Wenhe Wang [1], Keji Zhou [3,*] and Xiaoyong Xue [3]**

[1] State Grid Key Laboratory of Power Industrial Chip Design and Analysis Technology, Beijing Smart-Chip Microelectronics Technology Co., Ltd., Beijing 100192, China; dongyan-zhao@sgitg.sgcc.com.cn (D.Z.); wangyubo@sgitg.sgcc.com.cn (Y.W.); shaojin@sgitg.sgcc.com.cn (J.S.); chenyanning@sgitg.sgcc.com.cn (Y.C.); pancheng@sgitg.sgcc.com.cn (C.P.); zhoumin3@sgitg.sgcc.com.cn (M.Z.); wangwenhe@sgitg.sgcc.com.cn (W.W.)
[2] Beijing Chip Identification Technology Co., Ltd., Beijing 100192, China; dongguangzhi@sgitg.sgcc.com.cn (G.D.); wufengxia@sgitg.sgcc.com.cn (F.W.)
[3] State Key Laboratory of ASIC and System, School of Microelectronics, Fudan University, Shanghai 201203, China; xuexiaoyong@fudan.edu.cn
* Correspondence: zwguo20@fudan.edu.cn (Z.G.); kjzhou@fudan.edu.cn (K.Z.)

**Abstract:** In recent years, compute-in-memory (CIM) has been extensively studied to improve the energy efficiency of computing by reducing data movement. At present, CIM is frequently used in data-intensive computing. Data-intensive computing applications, such as all kinds of neural networks (NNs) in machine learning (ML), are regarded as 'soft' computing tasks. The 'soft' computing tasks are computations that can tolerate low computing precision with little accuracy degradation. However, 'hard' tasks aimed at numerical computations require high-precision computing and are also accompanied by energy efficiency problems. Numerical computations exist in lots of applications, including partial differential equations (PDEs) and large-scale matrix multiplication. Therefore, it is necessary to study CIM for numerical computations. This article reviews the recent developments of CIM for numerical computations. The different kinds of numerical methods solving partial differential equations and the transformation of matrixes are deduced in detail. This paper also discusses the iterative computation of a large-scale matrix, which tremendously affects the efficiency of numerical computations. The working procedure of the ReRAM-based partial differential equation solver is emphatically introduced. Moreover, other PDEs solvers, and other research about CIM for numerical computations, are also summarized. Finally, prospects and the future of CIM for numerical computations with high accuracy are discussed.

**Keywords:** compute-in-memory (CIM); numerical computations; resistive random-access memory (ReRAM); partial differential equations (PDEs); crossbar



**Citation:** Zhao, D.; Wang, Y.; Shao, J.; Chen, Y.; Guo, Z.; Pan, C.; Dong, G.; Zhou, M.; Wu, F.; Wang, W.; et al. Compute-in-Memory for Numerical Computations. *Micromachines* **2022**, *13*, 731. https://doi.org/10.3390/mi13050731

Academic Editors: Andrey Sokolov and Haider Abbas

Received: 25 February 2022
Accepted: 18 April 2022
Published: 2 May 2022



## 1. Introduction

Different from data-intensive computing (typically all kinds of neural networks), high-precision computing is aimed at accurate numerical computations like large-scale matrix multiplication and partial differential equations (PDEs). At present, neural networks (NNs) have a lot of applications and are widely used in daily life. However, applications of high-precision computing could not be solved by NNs, whether in scientific research or in the actual scene. The requirements of throughput and energy efficiency for computing are constantly improving; therefore, CIM (computing-in-memory) is proposed as the solution of the Von Neumann bottleneck. The progress of CIM for numerical computations has great value in finance, engineering, computer science and other disciplines. It is ubiquitous in the field of scientific research and engineering. For example, improving the physical authenticity of virtual reality (VR), analyzing SIS infectious diseases with age structure, studying the BSM equations of derivative pricing theory, preprocessing and extracting image information and many other practical problems involve partial differential equations.

In recent years, all kinds of PDEs solvers based on different CIMs, including ReRAM, SRAM, flash memory and PCM, have emerged in numerical computing research. ReRAM-based CIM, as a relatively mature CIM technology, is still used for most of the high-precision CIM research. So, this article reviews the ReRAM technology, the principle of the ReRAM crossbar and the working process of ReRAM in CIM, firstly. Then, it summarizes the numerical methods of PDEs, matrix iterative methods, rearrangement methods and split methods. After that, the working procedure and current developments of all kinds of CIM-based partial differential equation solvers are discussed. Moreover, their performance and characteristics are also compared. Aimed at defects in PDEs solvers, the solutions to get high-precision in large-scale matrix multiplication under environmental effects are proposed. In the future, the developments of CIM-based numerical computations will be improved in the manufacturing process, the write-verify method, the algorithm of sparse matrixes and the software/hardware collaboration.

## 2. ReRAM

### *2.1. The Appearance of ReRAM*

In the early 1960s, various research about ReRAM devices with all kinds of oxide materials, including $Al_2O_3$, NiO, $SiO_2$, $Ta_2O_5$, $ZrO_2$, $TiO_2$ and $Nb_2O_5$, emerged in an endless stream [1–5]. Compared with metal-oxide-semiconductor field-effect transistors (MOSFET), which appeared in 1960 [6,7] for the first time, ReRAM devices are the products of the same period. In the 40 years that followed, the technology of resistive switching has not made significant progress in storage applications.

### *2.2. The Development of ReRAM as NVM*

With the explosive growth of portable electronic devices, the requirement and storage capacity of memory devices have increased rapidly. Higher density, faster speed and lower cost have become the goal of new memory devices. ReRAM, as a kind of non-volatile memory (NVM) [8], was regarded as one of the continuations of NAND flash memory [9], though there were many emerging nonvolatile memory (eNVM) devices, such as phase-change memory (PCM) [10], magnetic random-access memory (MRAM) [11] and ferroelectric random-access memory (FeRAM) [12], over the same period. Table 1 lists the types of NVMs and the category of ReRAM.

**Table 1.** The types of the NVM and the category of ReRAM.

| ReRAM | MRAM | FeRAM | PCM | Flash Memory |
|---|---|---|---|---|
| OxRAM<br>CBRAM | STT-MRAM<br>SOT-MRAM<br>VCMA | FTJ | | NAND Flash<br>Nor Flash<br>AG-AND Flash |

ReRAM is a two-terminal device with a variable resistance based on a physical mechanism of conducting filament formation and rupture [13]. According to the types of filamentary, ReRAM can be divided into oxide ReRAM (OxRAM) and conductive bridge ReRAM (CBRAM) [14]. ReRAM changes between high-resistance states (HRS) and low-resistance states (LRS) under different operating conditions, representing logic 0 and 1, separately. The formation of the conducting filament corresponds to the LRS, and the HRS is the opposite.

Figure 1a is the structure of the OxRAM, and there is a metal oxide material between the two electrodes in the OxRAM. When a positive voltage is applied between the top electrode (TE) and the bottom electrode (BE), a conductive filament is formed between the two electrodes. While in Figure 1b, the electrode of the CBRAM is injected with copper or silver metal (Cu or Ag). Moreover, the CBRAM forms the conductive bridge by diffusing Cu or Ag into the oxide or chalcogenide (like $GeS_2$). When the voltage is sufficiently positive, there will be the oxidation of Cu or Ag at TE, and they will be reduced and deposited at BE. When the voltage transfers to the negative, there will be the reduction of Cu or Ag at

TE, and then the conductive filament connecting the two electrodes, and the state of the CBRAM, changes from HRS to LRS [15].

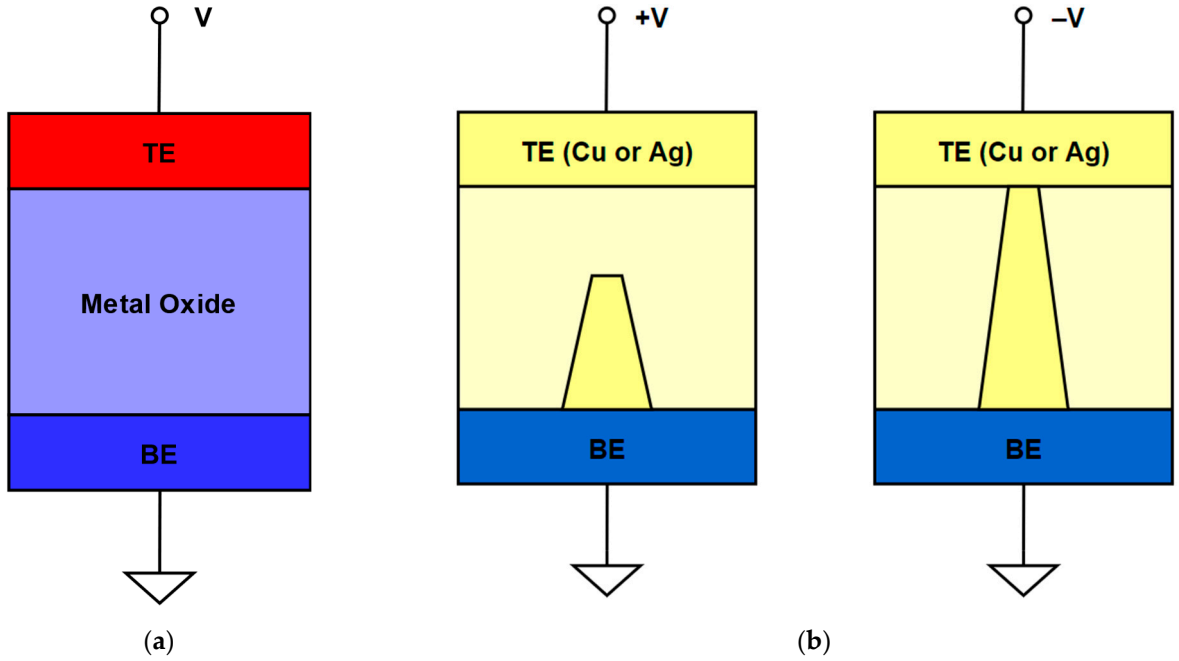



**Figure 1.** (**a**) Structure of OxRAM; (**b**) Structure of CBRAM.

Since 2000, the research on ReRAM have explosively increased. The first $NiO_x$-based ReRAM with promising device characteristics and reliability was proposed by I. Baek in 2005 [16]; the $HfO_2$/Ti device was made with fully conventional fab materials [17]; the 3D vertical ReRAM emerged in 2009 [18]; the 10 × 10 $nm^2$ Hf/$HfO_x$ crossbar resistive RAM was produced in 2011 [19]; the first 16-Gb ReRAM integrated chip with copper oxide material [20], etc. However, because of the 15 nm critical dimension (CD) and the development of 3D NAND flash memory, using the ReRAM in high-density applications became more and more difficult.

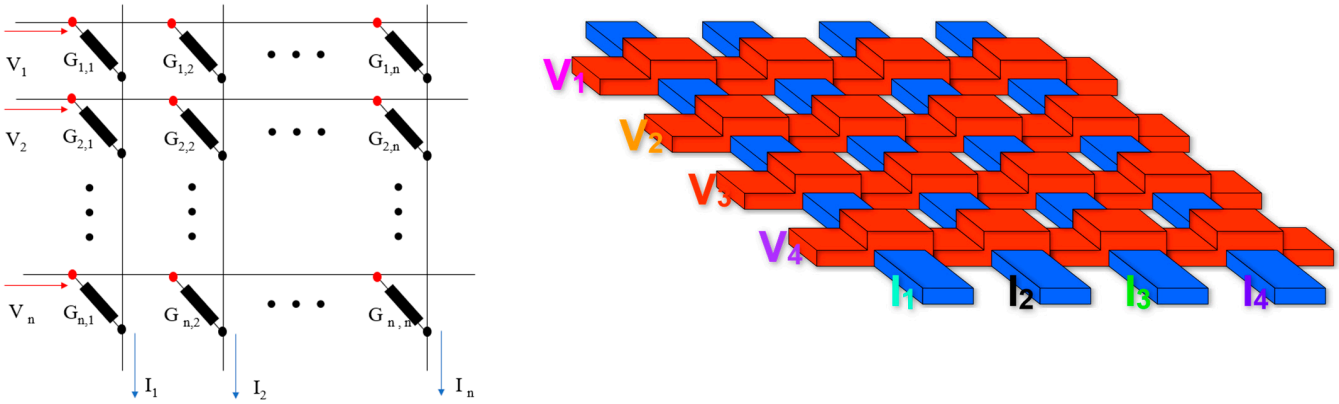
### *2.3. ReRAM in CIM*

In recent years, compute-in-memory widely emerged in machine learning (ML) and data-intensive computing. CIM is an effective method to break the Von Neumann bottleneck when computing large-scale data [21], which can achieve high speed and low-power computing by reducing data handling. Edge AI applications based on deep neural networks (DNNs) are designed to find the solution to achieving portable, fast, accurate and convenient computing. Computing efficiency (defined as terra-operations-per-second-per-millimeter-squared, TOPS/$mm^2$) and energy efficiency (defined as terra-operations-per-second-per-watt, TOPS/W) are the two most significant parameters to measure the performance of computing. For digital neural network accelerators, multiplication and addition are calculated in the processing element (PE). However, the global buffer or cache is urgently needed to store the weights and the inputs/outputs, which increases the data storage and handling. There is quite a lot of research on optimizing data flow at the chip, micro and SOC (system on chip) levels, but computation and memory are all separated, which leads to efficiency degradation. The memory not only stores the weights and the inputs but also achieves analog computation. That is what computing-in-memory means. Compared with traditional digital signal accelerators, CIM as a mix-signal processing tremendously increases throughput, area efficiency and energy efficiency, but with the decline of accuracy. Though the requirement of analog-to-digital converters (ADCs) is inevitable, CIM still has enormous appeal in power consumption, no matter whether now or in the future.

Because of the resistive properties of ReRAM, ReRAM could be a natural electrical multiplier, with the function of storage following Ohm's laws and Kirchhoff's current laws (KCL). Utilizing $I = V \cdot G$ and the sum of the current, multiplication and accumulation are calculated by the ReRAM array in the analog domain, respectively. Obviously, eNVM, including resistive random-access memory, is an excellent memory device for CIM, and other eNVM devices such as PCM, MRAM and FeRAM are also of interest for CIM [22]. ReRAM is a better choice for compute-in-memory because of the 22 nm high reliability and the compatibility with the complementary metal-oxide-semiconductor (CMOS) process at present. In addition, ReRAM could potentially offer multi-bit per cell capability [22].

### *2.4. ReRAM Crossbar*

Figure 2a,b shows the principle of calculation in the ReRAM crossbar array. One terminal of each RRAM is connected to the bit line (BL) collecting the current, and the other is connected to the word line (WL) as the input of the voltage. Additionally, the currents through the BLs are added as the outputs of the ReRAM array to achieve the accumulation. One of the outputs could be given by $I_j = \sum V_i \cdot G_{i,j}$ (where $I_j$ is the current of the $j$th column, $V_i$ is the input voltage of the $i$th row, and $G_{i,j}$ is the conductance of the $i$th row and $j$th column in ReRAM array). Due to the fact that the outputs are the analog currents, ADC is an indispensable part of the ReRAM crossbar peripheral circuit.




(**a**) (**b**)

**Figure 2.** (**a**) Schematic diagram of ReRAM crossbar; (**b**) Structure of ReRAM crossbar.

## 3. Partial Differential Equation

A partial differential equation is any equation with a function of multiple variables and their partial derivatives [23]. The function $u$:

$$u = u\left(t\,,\,x_1\,,\,\cdots\,,\,x_n\right) \tag{1}$$

and the Partial differential equation can be defined as:

$$g\left(t\,,\,x_1\,,\,\cdots\,,\,x_n\,,\,u\,,\,\frac{\partial u}{\partial t}\,,\,\frac{\partial u}{\partial x_1}\,,\,\cdots\,,\,\frac{\partial u}{\partial x_n}\,,\,\frac{\partial^2 u}{\partial t^2}\,,\,\cdots\right) = 0 \tag{2}$$

Typical results of partial differential equations have two forms: the analytical solution and the numerical solution. The analytical solution that can be expressed by an analytical expression is an exact combination of finite common operations. Given any independent variable, its dependent variable could be solved, so the analytical solution is also known as the closed-form solution. The numerical solution needs to be calculated iteratively from the boundary condition step-by-step [24], and it is the emphasis of the method in PDEs solver research. With the decrease of the step size, the numerical solution will be more

accurate. There are multiple numerical methods, including the Euler method, Runge-Kutta, finite-difference [25], finite-element method [26] and finite-volume method [23].

### *3.1. Numerical Methods*

#### 3.1.1. Finite-Difference Method

The principle of the finite-difference method (FDM) is understandable: converting the continuous problem to its corresponding discrete form and getting results within a finite number of calculations. The core mechanism of FDM is to approximate the partial derivatives at each point using its nearby values based on Taylor's theorem. There are three basic steps in the finite-difference method:

(1) Regional discretization. According to the appropriate step size, the domain that needs to be calculated is divided into finite grids and using the function values on discrete grid points to approximate the continuous function values.
(2) Transformation of partial differential equations. Using the difference coefficient to approximate the exact derivatives.
(3) Solution of partial differential equations. Bringing the boundary conditions into the equation and repeating calculations to solve a large number of equations.

The first-order finite-difference of $g(x)$ of variable $x$ can be defined as:

$$\Delta g(x) = g(x + \Delta x) - g(x) \tag{3}$$

where $\Delta x$ is step size, or the so-called spacing between two grid points, and the first-order difference coefficient of $g(x)$ of variable $x$ can be defined as:

$$\frac{dg(x)}{dx} \approx \frac{\Delta g(x)}{\Delta x} = \frac{g(x + \Delta x) - g(x)}{\Delta x} \tag{4}$$

So that the forward difference coefficient, backward difference coefficient and central difference coefficient can be expressed as:

$$\frac{g(x + \Delta x) - g(x)}{\Delta x}, \frac{g(x) - g(x - \Delta x)}{\Delta x}, \frac{g(x + \Delta x) - g(x - \Delta x)}{2\Delta x} \tag{5}$$

The finite-difference method uses the difference coefficient to approximate the exact derivative. Similarly, the second-order difference coefficient can be expressed as:

$$\frac{d^2 g(x)}{dx^2} \approx \frac{g(x + \Delta x) + g(x - \Delta x) - 2g(x)}{\Delta x^2} \tag{6}$$

Taking a simple one-dimensional heat diffusion equation without a heat source as an example, the equation and boundary conditions are as follows:

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) = \frac{\partial^2 u}{\partial t^2}(x,t) + f(x,t) \\ u(x,0) = \varphi(x)\ ,\ f(x,t) = 0 \\ u(a,t) = v1\ , u(b,t) = v2 \\ a \le x \le b\ , 0 \le t \le T \end{cases} \tag{7}$$

where $u(x,t)$ is the temperature at grid point $x$ at time $t$, $a^2$ is the thermal diffusivity, $f(x,t)$ is the heat source and $v1$ and $v2$ are the constant boundary conditions at grid point $a$ and $b$, respectively.

Then, the solution domain is divided into finite grids in Figure 3. The step size in the t-axis direction is taken as $\Delta t$, and the step size in the x-axis direction is taken as $\Delta x$. Therefore,

$$x_i = a + (i + 1)\Delta x\ ,\ \Delta x = \frac{b - a}{N} \tag{8}$$

$$t_i = 0 + (j+1)\Delta t\ ,\ \Delta x = \frac{T}{M} \tag{9}$$

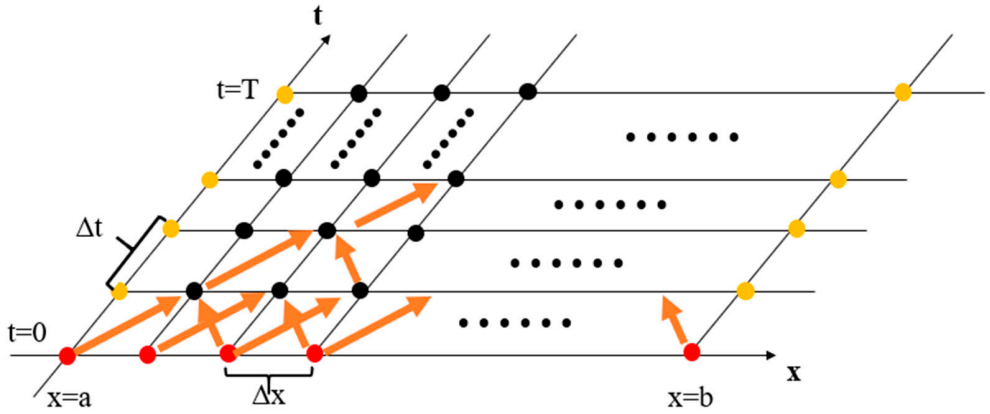



**Figure 3.** Lattice graph of a one-dimensional heat diffusion equation.

According to (4) and (6), the continuous function (7) will change to the function of values on discrete grid points:

$$\frac{u(x_i, t+\Delta t) - u(x_i, t)}{\Delta x} = a^2 \frac{u(x_i + \Delta x, t) + u(x_i - \Delta x, t) - 2u(x_i, t)}{\Delta x^2} \tag{10}$$

$$u(x_i, t+\Delta t) = u(x_i, t) + \frac{\Delta t \cdot a^2}{\Delta x^2}[u(x_i + \Delta x, t) + u(x_i - \Delta x, t) - 2u(x_i, t)] \tag{11}$$

Excluding the boundary values, when $t = \Delta t$, the actual temperature of each point is

$$\begin{array}{c} u(x_1, \Delta t) = \varphi(a) + \frac{\Delta t \cdot a^2}{\Delta x^2}[u(x_2, \Delta t) + u(x_0, 0) - 2u(x_1, 0)] \\ u(x_2, \Delta t) = \varphi(a + \Delta x) + \frac{\Delta t \cdot a^2}{\Delta x^2}[u(x_3, \Delta t) + u(x_1, 0) - 2u(x_2, 0)] \\ \vdots \\ u(x_{N-1}, \Delta t) = \varphi(b) + \frac{\Delta t \cdot a^2}{\Delta x^2}[u(x_N, \Delta t) + u(x_{N-2}, 0) - 2u(x_{N-1}, 0)] \end{array} \tag{12}$$

The above system of linear equations contains $N-1$ equations, and we can rewrite them into a matrix equation:

$$\begin{bmatrix} u_1^{\Delta t} \\ u_2^{\Delta t} \\ \vdots \\ u_{N-1}^{\Delta t} \end{bmatrix} = \begin{bmatrix} u_1^0 \\ u_2^0 \\ \vdots \\ u_{N-1}^0 \end{bmatrix} + \frac{\Delta t \cdot a^2}{\Delta x^2} \cdot \begin{bmatrix} 2 & -1 & \cdots & 0 \\ -1 & 2 & -1 & \vdots \\ \vdots & -1 & 2 & -1 \\ 0 & \cdots & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1^0 \\ u_2^0 \\ \vdots \\ u_{N-1}^0 \end{bmatrix} \tag{13}$$

(13) can also be expressed as:

$$U^{t+\Delta t} = U^t + AU^t \tag{14}$$

where $A$ is the coefficient matrix, $U^t$ is the matrix at time $t$ and $U^{t+\Delta t}$ is the matrix at time $t + \Delta t$.

The one-dimensional equation is the most ideal and simplest physical scenario. However, researchers and engineers need to solve the problems of two-dimensional or even multidimensional space aimed at the actual requirements. At this time, the two-point difference method is not suitable. As an example, there is a two-dimensional equation, a Laplace equation:

$$\begin{cases} \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = 0 \\ u(a,y) = \phi_1(y)\,, u(b,y) = \phi_2(y) \\ u(x,c) = \phi_3(y)\,, u(x,d) = \phi_4(y) \\ a \le x \le b\,, c \le y \le d \end{cases} \tag{15}$$

According to (6), the Laplace equation can be written as:

$$u(x+\Delta x, y) + u(x-\Delta x, y) + u(x, y+\Delta y) + u(x, y-\Delta y) - 4u(x,y) = 0 \tag{16}$$

where the step size in the x-axis direction is taken as $\Delta x$, and the step size in the y-axis direction is taken as $\Delta y$. In Figure 4, the function value of each point can be calculated by the function values of its four neighboring points, and the whole domain is divided into $N \times N$ grid points.

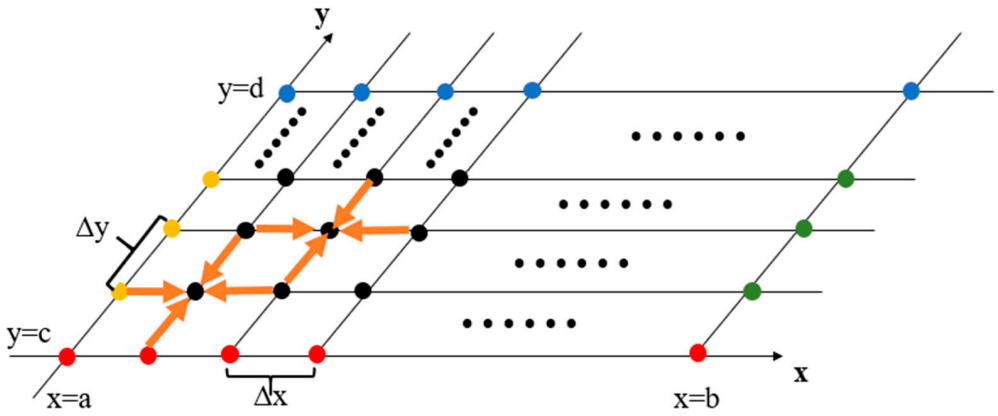



**Figure 4.** Lattice graph of a Laplace equation.

The equation in the grid point of $(i, j)$ can be expressed as:

$$u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1} = 0 \tag{17}$$

Thus, (15) can be transformed into a matrix form as:

$$\begin{bmatrix} u_{-1,0} + u_{0,-1} \\ u_{-1,1} \\ u_{-1,2} + u_{0,3} \\ \vdots \\ u_{N-1,N} + u_{N,N-1} \end{bmatrix}_{N^2 \times 1} = \begin{bmatrix} B & -I & \cdots & 0 \\ -I & B & -I & \vdots \\ \vdots & -I & B & -I \\ 0 & \cdots & -I & B \end{bmatrix}_{N^2 \times N^2} \begin{bmatrix} u_{0,0} \\ u_{0,1} \\ u_{0,2} \\ \vdots \\ u_{N-1,N-1} \end{bmatrix}_{N^2 \times 1} \tag{18}$$

where

$$B = \begin{bmatrix} 4 & -1 & \cdots & 0 \\ -1 & 4 & -1 & \vdots \\ \vdots & -1 & 4 & -1 \\ 0 & \cdots & -1 & 4 \end{bmatrix}_{N \times N}, \; I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{N \times N} \tag{19}$$

Consequently, (18) can also be expressed as:

$$U^b = AU \tag{20}$$

where $A$ is the coefficient matrix, $U$ is the matrix in grid point of $(i, j)$ and $U^b$ is the matrix composed of the boundary conditions.

3.1.2. Runge-Kutta Method

The Runge-Kutta method includes two kinds of methods: second-order Runge-Kutta and fourth-order Runge-Kutta.

Second-Order Runge-Kutta: $k_1$ and $k_2$ can be calculated by the following equations,

$$k_1 = h \times f(x_n, y_n) \tag{21}$$

$$k_2 = h \times f(x_n + h, y_n + k_1) \tag{22}$$

The solution of the PDE can be expressed as:

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2) \tag{23}$$

Fourth-Order Runge-Kutta: $k_1$, $k_2$, $k_3$ and $k_4$ can be calculated by the following equations,

$$k_1 = h \times f(x_n, y_n) \tag{24}$$

$$k_2 = h \times f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \tag{25}$$

$$k_3 = h \times f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \tag{26}$$

$$k_4 = h \times f(x_n + h, y_n + k_3) \tag{27}$$

The solution of the PDE can be expressed as:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{28}$$

*3.2. Matrix Iterative Methods*

After getting the matrix equations, the next main task is the iterative computation of a large-scale matrix, and common iterative methods include the Jacobi method, the Guass Seidel method and the SOR method.

3.2.1. Jacobi Method

The principle of the Jacobi method is to disassemble the coefficient matrix $A$ into a diagonal matrix $D$, a negative upper triangular matrix $U$ and a negative lower triangular matrix $L$. Consequently, $A$ can be written as:

$$A = D - L - U \tag{29}$$

$$D = \begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{bmatrix}, \; L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -a_{2,1} & 0 & 0 & \vdots \\ \vdots & \ddots & \ddots & 0 \\ -a_{n,1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix}, \; U = \begin{bmatrix} 0 & -a_{1,2} & \cdots & -a_{1,n} \\ 0 & 0 & 0 & \vdots \\ \vdots & 0 & \ddots & -a_{n-1,n} \\ 0 & \cdots & 0 & 0 \end{bmatrix} \tag{30}$$

For an equation of a matrix like $B = A \cdot X$, replacing the coefficient matrix $A$, it will change to:

$$B = (D - L - U) \cdot X \tag{31}$$

$$X = D^{-1}(L + U)X + D^{-1}B \tag{32}$$

After the iterative calculation, the calculation result of the $(K+1)$th iteration is

$$X^{(k+1)} = D^{-1}(L + U)X^{(k)} + D^{-1}B \tag{33}$$

3.2.2. Guass Seidel Method

The principle of the Guass Seidel method is similar to the Jacobi method, where the difference is the derivation process, the Guass Seidel method rewrites $B = A \cdot X$ into:

$$X = (D - L)^{-1}UX + (D - L)^{-1}B \tag{34}$$

After the iterative calculation, the calculation result of the $(K + 1)$th iteration is

$$X^{(k+1)} = (D - L)^{-1}UX^{(k)} + (D - L)^{-1}B \tag{35}$$

In most cases, the Guass Seidel method converges faster than the Jacobi method, and only one set of storage units is needed to store $(D - L)^{-1}$, but $D^{-1}(L + U)$ and $D^{-1}$ are required to store in the Jacobi method.

3.2.3. SOR Method

Based on the Guass Seidel method, a convergence factor $\omega$ is added to the SOR method in order to improve the convergence speed. The calculation result of the $(K + 1)$th iteration is

$$X^{(k+1)} = (D - \omega L)^{-1}[(1 - \omega)D + \omega U]X^{(k)} + \omega(D - \omega L)^{-1}B \tag{36}$$

3.2.4. Krylov Subspace Method

For an equation of a matrix like $B = A \cdot X$, the result $X$ can be expressed as $A^{-1} \cdot B$ directly. However, if the matrix $A$ has a large size or is a sparse matrix, $A^{-1}$ will be very hard to solve. The principle of the Krylov subspace method is to approximate $A^{-1} \cdot B$.

$$A^{-1} \cdot B \approx \sum_{i=0}^{m-1} \beta_i A^i B = \beta_0 B + \beta_1 AB + \beta_2 A^2 B +, \cdots, +\beta_{m-1} A^{m-1} B \tag{37}$$

where $\beta_0$ , $\beta_1$ , $\beta_2$ , $\cdots$ , $\beta_{m-1}$ are unknown coefficients, the step size $m$ is related to the accuracy of the approximation and is less than the dimension of matrix $A$.

After the discussion above, the number of iterations used by the Jacobi method, the Guass Seidel method and the SOR method is decreasing, which also means the improving of the calculation accuracy. Additionally, in terms of the hardware consumption and hardware implementation, the Guass Seidel method is better than the others. Moreover, the Krylov subspace method sacrifices accuracy to improve speed and is generally used large-scale matrixes. Relatively speaking, the SOR method is the most accurate and efficient method. However, the matrixes could not be inputted into the ReRAM arrays iteratively as weights. Calculating the matrixes directly will waste computing power in the multiplication of zero elements, because the matrixes used in numerical computations are generally sparse matrixes which include more than 60% of zero elements.

### *3.3. Rearrangement and Split*

Rearrangement and split are proposed to solve the multiplication of sparse matrixes whose majority of elements are zero elements. There are a number of studies focusing on cutting or splitting matrixes to improve computational efficiency, while many matrixes cannot be cut or split directly all the time. Therefore, rearrangement of sparse matrixes is needed to change the layout of matrix elements. In sparse matrix-vector multiplication (SpMV), the rearrangement matrix and the splitting matrix can replace sparse matrixes with dense matrix operations in many cases, which can greatly save memory and reduce computational overhead [27]. Moreover, the methods reducing the bandwidth of sparse matrixes in SpMV are quite useful for matrixes got from the PDEs. Taking a $1024 \times 1024$ sparse matrix as an example in Figure 5, the gray part of the matrix is composed of zero elements, and the blue part is composed of nonzero elements. Firstly, the $1024 \times 1024$ sparse matrix is rearranged to a diagonal aggregation matrix (not a diagonal matrix). In addition, it is

rewritten as a combination of $B$ and $I$. At last, the matrix is divided into four types of $16 \times 16$ slices (a, b, c, d).

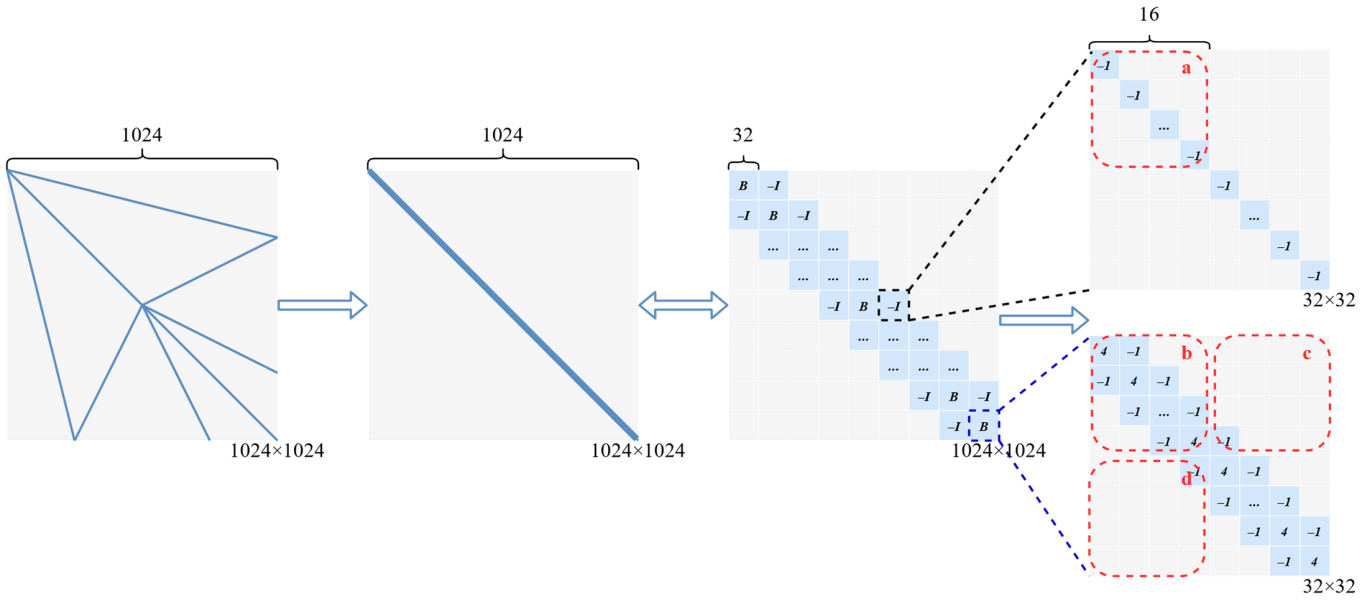



**Figure 5.** The process that a $1024 \times 1024$ sparse matrix is divided into slices.

According to the discussion of the matrix iterative methods in Section 3.2, the number of iterations used by the Jacobi method, the Guass Seidel method and the SOR method is decreasing. Though the Guass Seidel method has a lower hardware consumption and the SOR method has the highest computational efficiency, utilizing the Jacobi method with the splitting matrix can not only improve the accuracy but also cut down the number of iterations efficiently. Because the principal concern is which method has fewer zero elements, the Jacobi method is still the best choice for the CIM-based PDEs solver at present.

## 4. CIM-Based Partial Differential Equation Solver

With the discussion of matrix iterative methods, the problem of the PDEs will be changed to the multiplication of the large-scale matrix. Therefore, the essence of the PDEs Solver is to achieve the high-performance multiplication. The CIM-based PDEs Solver could be composed with input drivers, shifters, adders, a computing array and DAC/ADCs. Matrix $A$ will be stored in the array as weight, and matrix $X$ will be entered into the array as input. Because of the limited size and precision of the array, the array maps a single column of the matrix to serval columns of an array. After the shifters and adders, the results of the serval columns will be collected and then be quantified in ADCs. The ADCs in the CIM PDEs solvers are usually SAR ADCs to balance the area, power and speed. Recently, CIM-based partial differential equation solvers can be based on different CIMs, including ReRAM, SRAM, flash memory and PCM. Each of them has advantages and their own suitable calculating methods and circuits.

### *4.1. ReRAM-Based Partial Differential Equation Solver*

The coefficient matrix has been divided into several slices, and next they will be written as resistance values into the ReRAM array. Matrix $X$ will be entered into the ReRAM array as input, and the slices can be used several times without replacement. That means the ReRAM array rarely needs to be written. The larger the size of the slices, the smaller the number of slices with the decrease of write times. On the contrary, the times of the iterative calculation will increase, and the computational efficiency will be lower. After the multiplication in the simulation domain, ADCs are required to convert the analog signals into digital signals. Then, after the digital signal processing, the result will be iterative, as

will the input of the ReRAM array, and the final result can be got from the ReRAM-based partial differential equation solver.

In the work of Mohammed A. Zidan, a general memristor-based partial differential equation solver is proposed with the finite-difference method. To solve the general matrix Equation (20), they use the Jacobi method to decrease the calculation of zero elements. Their memristor is composed of a Ta top electrode, a Pd bottom electrode and a thin $Ta_2O_{5-x}$ metal oxide. Additionally, the memristor crossbar has extremely high energy efficiency and area utilization, but a lower accuracy because of the variation. With the write-verify approach, they decrease the conductance variation from 5.3% to 0.85%, which overcomes the accuracy defects of the ReRAM immensely. They divide the matrix into equally sized slices, and practical crossbar sizes can be mapped onto the active slices exactly. Cutting the matrix not only minimizes the effects of the series resistance, sneak currents and virtual grounds, but also reduces the operation of zero elements [23]. With the memristor-based hardware and matching software system, they get high-precision computing results.

Shichao Li and Wenchao Chen simulated fully coupled multiphysics based on bipolar resistive random-access memory in 2017 [28]. They utilized the finite-difference method and the Scharfetter-Gummel method to solve the PDEs, solving three fully coupled partial differential equations by the crossbar of the $HfO_x$–based ReRAM [29]. Like the work of Mohammed A. Zidan, the accuracy has not been effectively improved, while more PDEs are discussed in this work.

In recent work by S. S. Ensan and S. Ghosh, a ReRAM-based linear first-order PDE solver (ReLOPE) is proposed to solve PDEs of the following form:

$$f\,(x,\,y) = y' = ay + bx + c \tag{38}$$

Unlike the general memristor-based partial differential equation solver [23], ReLOPE is the first PDEs solver purely based on hardware and used only for linear first-order PDEs [20]. Moreover, the principle of the ReLOPE is based on the second-order Runge-Kutta method. Though theoretically fourth-order Runge-Kutta offers higher accuracy, the value of the resistance tremendously interferes with the iteration under the limitation of the accurate programming of the ReRAM. Substituting (3) and (4) into (37), (37) will change to:

$$y_{n+2} - y_{n+1} = \frac{2 + 2h{\cdot}a + h^2{\cdot}a}{2}(y_{n+1} - y_n) + \frac{2h{\cdot}b + h^2{\cdot}a{\cdot}b}{2}(x_{n+1} - x_n) \tag{39}$$

Similarly, substituting Equations (3)–(6) into (37), (37) will change to:

$$\begin{aligned} y_{n+2} - y_{n+1} = &\left(1 + \tfrac{2}{3}h{\cdot}a + \tfrac{1}{3}h^2{\cdot}a^2 + \tfrac{h^3{\cdot}a^3 + 2h^3{\cdot}a^2 + h^4{\cdot}a^4}{24}\right)(y_{n+1} - y_n) \\ &+\left(\tfrac{1}{6}b + \tfrac{1}{4}h{\cdot}b + \tfrac{1}{8}h^2{\cdot}a{\cdot}b + \tfrac{h^3{\cdot}a^2{\cdot}b + 2h^3{\cdot}a^2{\cdot}b + h^4{\cdot}a^3{\cdot}b}{24}\right)(x_{n+1} - x_n) \end{aligned} \tag{40}$$

Figure 6 is the overview of ReLOPE. ReLOPE includes a fully ReRAM crossbar-based CIM, shifters, adders and DAC/ADCs. It expands the operating range of the solution by exploiting shifters to shift input data and output data. ReLOPE improves its power consumption, solving a PDE by 31.4×. The above methods with ReLOPE used have two limitations: (1) it is unachievable to program RRAMs with this accuracy (six decimal points for fourth-order Runge-Kutta in the paper of ReLOPE) at the current technical level; (2) the accuracy has loss due to the nonlinear variation of resistance with voltage and the iterative use of ADC. Therefore, ReLOPE cannot further improve its accuracy with the method of Runge-Kutta.

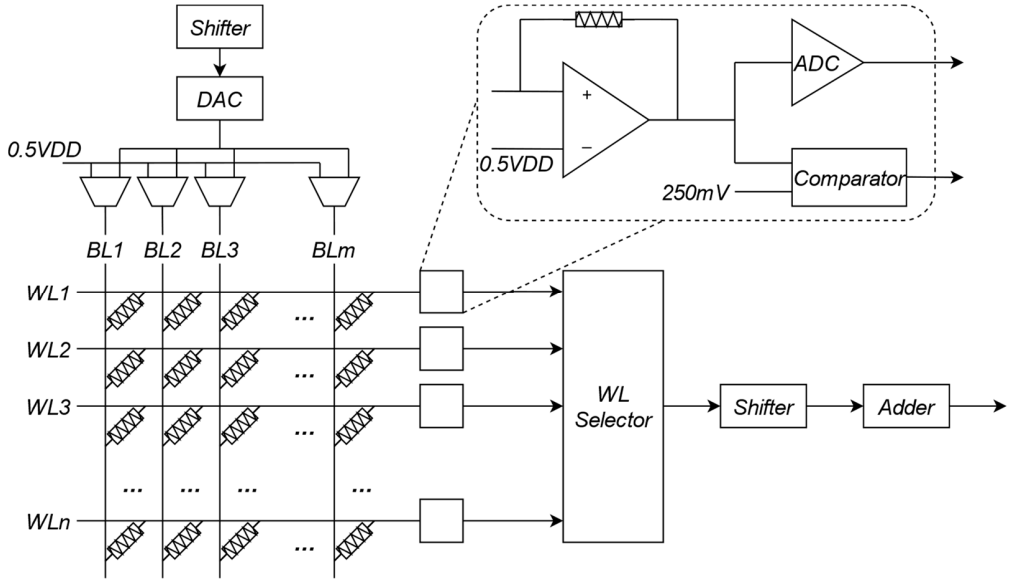



**Figure 6.** Overview of ReLOPE.

### *4.2. SRAM-Based Partial Differential Equation Solver*

Yannis Tsividis et al. proposed a programmable, clockless, continuous-time 8-bit hybrid (mixed analog/digital) architecture (ADC + SRAM + DAC) for solving ordinary and partial differential equations. The architecture, shown in Figure 7, is used to achieve nonlinear functions. The system consisted of an analog multiplier and analog adder/subtractor. The hybrid nonlinear function generator achieves 16× lower power dissipation and the computational accuracy of about 0.5% to 5%.

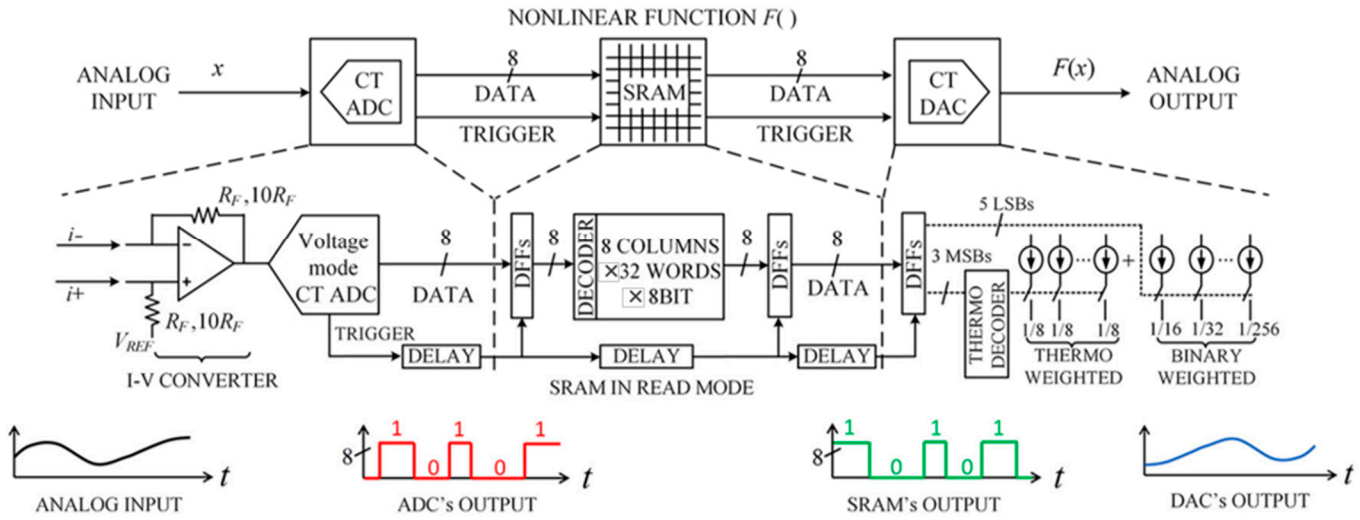
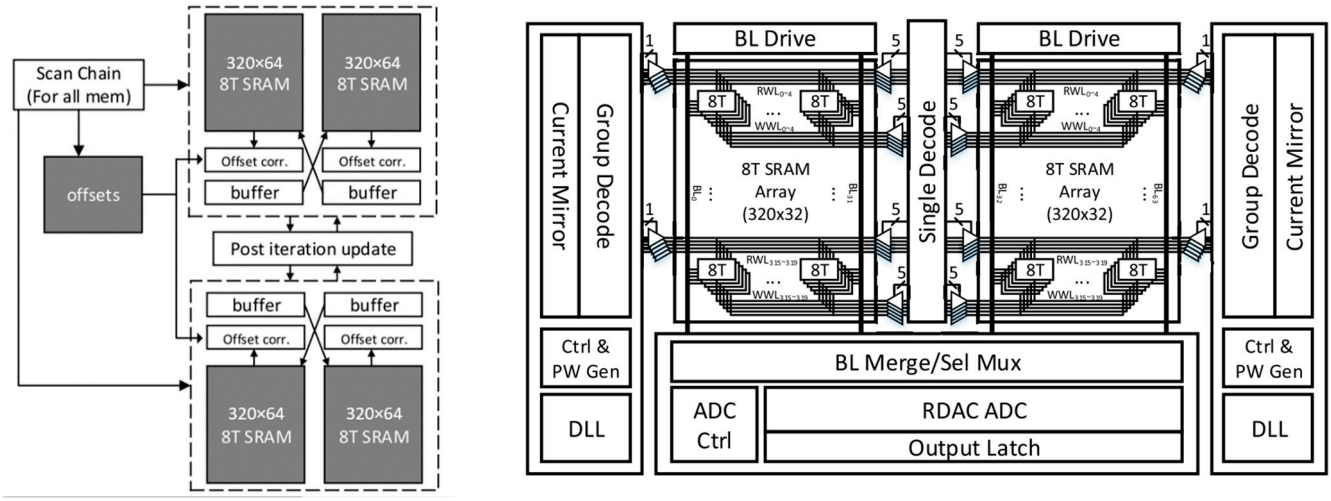



**Figure 7.** The continuous–time programmable nonlinear function generator.

In 2019, Thomas Chen and Jacob Botimer proposed a SRAM-based accelerator for solving PDEs [30]. They reformulated the multigrid Jacobi method in a residual form. By interleaving coarse-grid iterations with fine-grid iterations, their system reduced low-frequency errors to accelerate convergence. Their system contains 4 MAC–SRAMs, and each is a 320 × 64 8T SRAM array. The architecture is shown in Figure 8a,b. A year later, they updated the mapping of the MAC-SRAM [31] on the basis of their previous research. Finally, the SRAM-based accelerator achieved 56.9–GOPS, consuming 16.6 mW at 200–MHz. However, the SRAM-based CIM has the same accuracy problems due to limited multiplicand precision and limited ADC resolution.




(**a**) (**b**)

**Figure 8.** (**a**) Architectural sketch of PDE iteration module; (**b**) Block diagram of MAC SRAM.

### *4.3. Flash Memory-Based Partial Differential Equation Solver*

Jiezhi Chen et al. proposed a flash memory-based CIM hardware system to improve the computation efficiency of the time-dependent partial differential equations [32]. Based on the FDM and Jacobi algorithm, they got the matrix equation, then the coefficient matrix was mapped into the flash memory array as threshold voltages. Matrix $X$ is transformed into pulse time as input. Compared with ReRAM, flash memory enables the realization of vector-matrix multiplication with high accuracy and a good tolerance for device error. Moreover, it also has the advantages of ultra-high density and low cost.

### *4.4. PCM-Based Partial Differential Equation Solver*

In 2018, Manuel Le Gallo and Abu Sebastian et al. used mixed-precision in-memory computing which combined a von Neumann machine with a computational memory unit to solve PDEs [33]. The mixed-precision CIM PDEs solver uses a low-precision computational memory unit to obtain the approximate solution of the first part and high-precision processing iteratively to improve accuracy in the second part. It achieves the low-precision matrix-vector multiplication by using a PCM crossbar array based on the iterative Krylov-subspace method. The PCM-based PDEs solver could offer up to 80 times lower energy consumption than the FPGA solution because of the architecture of PCM-based CIM and the mixed-precision system.

### *4.5. Discussion of Partial Differential Equation Solver*

The memristive crossbar CIM PDEs solver based on the PCM chip could already offer up to 80 times lower energy consumption than the FPGA solution. The energy efficiency and speed of the memristive crossbar CIM PDEs solver is several hundred times than the PDEs solvers based on IBM POWER8 central processing unit (CPU) and NVIDIA Titan RTX graphics processing unit (GPU) [26,34]. However, the CIM-based PDEs usually solves the 4-bit to 8-bit PDEs computation in satisfaction of accuracy requirements.

Table 2 summaries the representative CIM-based PDEs solvers recently and their performance are compared.

**Table 2.** Comparation of several representative CIM-based PDEs solvers and GPU.

| Type of CIM | Reference | Technology Node | Energy Efficiency | Accuracy | Latency |
|---|---|---|---|---|---|
| ReRAM | Sina Sayyah Ensan 2021 VLSI | 65 nm | 31.4× | 11-bit (97%) | 25 ns |
| | Mohammed A. Zdan 2018 NE | NA | NA | 64-bit | 1 us |
| SRAM | Thomas Chen 2020 JSSC | 180 nm | 0.875 TOPS/W | 32-bit | 90 ns |
| | Ning Guo 2016 JSSC | 65 nm | 16× | 18-bit (95%) 8-bit (99.5%) | NA |
| NOR Flash Memory | Yang Feng 2020 SNW | 65 nm | NA | 64-bit | NA |
| PCM | Manuel Le Gallo 2018 NE | 90 nm | 24× | mixed-precision | <100 ns |
| GPU | NVidia Titan RTX | 12 nm FFN | 0.06 TOPS/W | 64-bit | NA |

The deficiencies of ReRAM, including low inherent accuracy, nonlinearity and susceptibility to environmental changes, directly determine the use of ReRAM for high-precision computing. With the constant development of process levels, ReRAM devices will be manufactured accurately and get close to high-precision computing to a certain extent. At the same time, the write-verify method or multiread/write method can also solve accuracy problems, but with an increase in latency. Both of them have a lot of worth in research in ReRAM-based PDEs solvers in the future. To reduce the memristor device variability and nonlinearity, in the work of Mohammed A. Zidan, a general memristor-based partial differential equation solver used a write–verify method to write and update the coefficient values in the crossbar. The write-verify operation is based on a sequence of write-read pulse pairs, each pair including a programming (set or reset) pulse and a subsequent read pulse. When the conductance reaches within a predetermined range of the target value, the write operation is considered complete. The write-verify feedback method could decrease the cell-to-cell variation of <1% from 5.3%.

Though flash memory and SRAM have higher accuracy and density compared with ReRAM for CIM in numerical computations, ReRAM with nonvolatile and multivalued characteristics is still the most extensively studied for CIM-based PDEs solvers.

The sparse coefficient matrix, with only a small number of nonzero elements, usually has a large matrix size. The coefficient matrix must be divided into a certain number of slices, whose sizes are typically $16 \times 16$ or $32 \times 32$. When the ReRAM array, having a huge size and multiple rows or columns, are unopened, the result of the ReRAM array may have a large error because of the unopened leakage currents. Hence, the size of the ReRAM array, and the size of the slices above-discussed, not only depends on the system working, but also is determined by the hardware operation of the ReRAM array. Moreover, the collected currents need ADC for the next computation, and there are losses of accuracy in the process. Other CIM-based PDEs solvers also have similar problems.

How to cut apart the matrix into slices, how small the size of crossbar arrays is and how the single array is to work are urgent problems to be solved at the system level. At the same time, rearrangement of sparse matrixes and SpMV will play an important role in the future of CIMs for high-precision computing tasks.

## 5. Summary and Outlook

Different from neural network computations, numerical computations are used in early, basic disciplines, receiving general attention from researchers all the time. In the past few years, the research has become more and more popular to solve numerical computations using CIM. CIM for numerical computations improves energy efficiency and computational efficiency. All kinds of CIM PDEs based ReRAM, SRAM, flash memory and PCM have

been proposed with various characteristics. The recent developments of CIM for numerical computations were compared. This article described the ReRAM-based CIM technology in detail. Then, it reviewed the numerical methods of PDEs and matrix iterative methods. Finally, the future of CIM for numerical computations can be summarized as follows:

1. Regardless of which array of CIM, the accuracy is still the biggest challenge;
2. More research of CIM-based numerical computations should focus on the computational methods of sparse matrixes;
3. As for matrix iterative methods, the principal concern is which method has fewer zero elements, so the Jacobi method is still the best choice for CIM-based PDEs solvers at present. In addition, the Krylov subspace method is better when solving very large-scale matrixes;
4. The future of CIM for high-precision computing tasks really needs a software/hardware codesign to collaborate the algorithm and the CIM array.

**Author Contributions:** Conceptualization, C.P. and X.X.; investigation, D.Z., Y.W., J.S., Y.C., Z.G., G.D., M.Z., F.W. and W.W.; writing—original draft preparation, Z.G.; writing—review and editing, K.Z.; supervision, K.Z.; project administration, C.P. and X.X. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported in part by National Key R&D Program of China under grant 2018YFB0407500. At the same time, this work is also supported by The Laboratory Open Fund of Beijing Smart-chip Microelectronics Technology Co., Ltd., (Beijing, China) under grant SGITZX00XSJS2108594.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Hickmott, T.W. Low-frequency negative resistance in thin anodic oxide films. *J. Appl. Phys.* **1962**, *33*, 2669–2682. [CrossRef]
2. Gibbons, J.; Beadle, W. Switching properties of thin Nio films. *Solid-State Electr.* **1964**, *7*, 785–790. [CrossRef]
3. Nielsen, P.; Bashara, N. The reversible voltage-induced initial resistance in the negative resistance sandwich structure. *IEEE Trans. Electron. Devices* **1964**, *11*, 243–244. [CrossRef]
4. Hiatt, W.R.; Hickmott, T.W. Bistable switching in niobium oxide diodes. *Appl. Phys. Lett.* **1965**, *6*, 106–108. [CrossRef]
5. Chen, Y. ReRAM: History, Status, and Future. *IEEE Trans. Electron. Devices* **2020**, *67*, 1420–1433. [CrossRef]
6. Atalla, M.M.; Kahng, D. *1960—Metal Oxide Semiconductor (MOS) Transistor Demonstrated Silicon Engine*; Tech. Rep.; Computer History Museum: Mountain View, CA, USA, 1960.
7. Kahng, D. Electric Field Controlled Semiconductor Device. U.S. Patent 3 102 230 A, 27 August 1963.
8. Xue, X.; Jian, W.; Yang, J.; Xiao, F.; Chen, G.; Xu, S.; Xie, Y.; Lin, Y.; Huang, R.; Zou, Q.; et al. A 0.13 µm 8 Mb Logic-Based $Cu_xO_y$ ReRAM With Self-Adaptive Operation for Yield Enhancement and Power Reduction. *IEEE J. Solid-State Circuits* **2013**, *48*, 1315–1322. [CrossRef]
9. Ishii, T.; Johguchi, K.; Takeuchi, K. Vertical and horizontal location design of program voltage generator for 3D-integrated ReRAM/NAND flash hybrid SSD. In Proceedings of the 2014 International Conference on Electronics Packaging (ICEP), Toyama, Japan, 23–25 April 2014.
10. Joshi, V.; le Gallo, M.; Haefeli, S.; Boybat, I.; Nandakumar, S.R.; Piveteau, C.; Dazzi, M.; Rajendran, B.; Sebastian, A.; Eleftheriou, E. Accurate deep neural network inference using computational phase-change memory. *Nat. Commun.* **2020**, *11*, 2473. [CrossRef] [PubMed]
11. Jain, S.; Ranjan, A.; Roy, K.; Raghunathan, A. Computing in memory with spin-transfer torque magnetic RAM. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 470–483. [CrossRef]
12. Takashima, D. Overview of FeRAMs: Trends and perspectives. In Proceedings of the 2011 11th Annual Non-Volatile Memory Technology Symposium Proceeding, Shanghai, China, 7–9 November 2011; pp. 1–6.
13. Wong, H.-S.P.; Lee, H.; Yu, S.; Chen, Y.-S.; Wu, Y.; Chen, P.-S.; Lee, B.; Chen, F.T.; Tsai, M.J. Metal-oxide RRAM. *Proc. IEEE* **2012**, *100*, 1951–1970. [CrossRef]
14. Jameson, J.R.; Blanchard, P.; Cheng, C.; Dinh, J.; Gallo, A.; Gopalakrishnan, V.; Gopalan, C.; Guichet, B.; Hsu, S.; Kamalanathan, D.; et al. Conductive-bridge memory (CBRAM) with excellent high-temperature retention. In Proceedings of the 2013 IEEE International Electron Devices Meeting, Washington, DC, USA, 9–11 December 2013; pp. 738–741.

15. Yu, S.; Wong, H.-P. Compact Modeling of Conducting-Bridge Random-Access Memory (CBRAM). *IEEE Trans. Electron. Devices* **2011**, *58*, 1352–1360.
16. Baek, I.G.; Lee, M.S.; Seo, S.; Lee, M.J.; Seo, D.H.; Suh, D.-S.; Park, J.C.; Park, S.O.; Kim, H.S.; Yoo, I.K.; et al. Highly scalable non-volatile resistive memory using simple binary oxide driven by asymmetric unipolar voltage pulses. In Proceedings of the IEEE International Electron Devices Meeting, San Francisco, CA, USA, 13–15 December 2004; pp. 587–590.
17. Lee, H.Y.; Chen, P.S.; Wu, T.Y.; Chen, Y.S.; Wang, C.C.; Tzeng, P.J.; Lin, C.H.; Chen, F.; Lien, C.H.; Tsai, M.-J. Low power and high speed bipolar switching with a thin reactive Ti buffer layer in robust HfO2 based RRAM. In Proceedings of the 2008 IEEE International Electron Devices Meeting, San Francisco, CA, USA, 15–17 December 2008; pp. 297–300.
18. Yoon, H.S.; Baek, I.-G.; Zhao, J.; Sim, H.; Park, M.Y.; Lee, H.; Oh, G.-H.; Shin, J.C.; Yeo, I.-S.; Chung, U.-I. Vertical cross-point resistance change memory for ultra-high density non-volatile memory applications. In Proceedings of the 2009 Symposium on VLSI Technology, Kyoto, Japan, 15–17 June 2009; pp. 26–27.
19. Govoreanu, B.; Kar, G.; Chen, Y.-Y.; Paraschiv, V.; Kubicek, S.; Fantini, A.; Radu, I.; Goux, L.; Clima, S.; Degraeve, R.; et al. $10\times10$ nm$^2$ Hf/HfOx crossbar resistive RAM with excellent performance, reliability and low-energy operation. In Proceedings of the 2011 International Electron Devices Meeting, Washington, DC, USA, 5–7 December 2011; pp. 729–732.
20. Sills, S.; Yasuda, S.; Strand, J.; Calderoni, A.; Aratani, K.; Johnson, A.; Ramaswamy, N. A copper ReRAM cell for storage class memory applications. In Proceedings of the 2014 Symposium on VLSI Technology (VLSI-Technology), Honolulu, HI, USA, 9–12 June 2014; pp. 80–81.
21. Hayakawa, Y.; Himeno, A.; Yasuhara, R.; Boullart, W.; Vecchio, E.; Vandeweyer, T.; Witters, T.; Crotti, D.; Jurczak, M.; Fujii, S.; et al. Highly reliable TaOx ReRAM with centralized filament for 28-nm embedded application. In Proceedings of the 2015 Symposium on VLSI Technology (VLSI Technology), Kyoto, Japan, 17–19 June 2015; pp. 14–15.
22. Yu, S. Compute-in-Memory for AI: From Inference to Training. In Proceedings of the 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 10–13 August 2020.
23. Ensan, S.S.; Ghosh, S. ReLOPE: Resistive RAM-Based Linear First-Order Partial Differential Equation Solver. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 237–241. [CrossRef]
24. Ames, W.F. *Numerical Methods for Partial Differential Equations*; Academic: New York, NY, USA, 2014.
25. Eymard, R.; Gallouët, T.; Herbin, R. *Handbook of Numerical Analysis*; Ciarlet, G.P., Lions, L.J., Eds.; Elsevier: Amsterdam, The Netherlands, 2000; pp. 713–1018.
26. Zidan, M.A.; Jeong, Y.; Lee, J.; Chen, B.; Huang, S.; Kushner, M.J.; Lu, W.D. A general memristor-based partial differential equation solver. *Nat. Electron.* **2018**, *1*, 411–420. [CrossRef]
27. Kabir, H.; Booth, J.D.; Raghavan, P. A multilevel compressed sparse row format for efficient sparse computations on multicore processors. In Proceedings of the 2014 21st International Conference on High Performance Computing (HiPC), Goa, India, 17–20 December 2014; pp. 1–10.
28. Li, S.; Chen, W.; Luo, Y.; Hu, J.; Gao, P.; Ye, J.; Kang, K.; Chen, H.; Li, E.; Yin, W.-Y. Fully Coupled Multiphysics Simulation of Crosstalk Effect in Bipolar Resistive Random Access Memory. *IEEE Trans. Electron. Devices* **2017**, *64*, 3647–3653. [CrossRef]
29. Yu, S. *Resistive Random Access Memory (RRAM): From Devices to Array Architectures*; Iniewski, K., Ed.; Morgan & Claypool: Saanichton, BC, Canada, 2016.
30. Chen, T.; Botimer, J.; Chou, T.; Zhang, Z. An Sram-Based Accelerator for Solving Partial Differential Equations. In Proceedings of the 2019 IEEE Custom Integrated Circuits Conference (CICC), Austin, TX, USA, 14–17 April 2019; pp. 1–4.
31. Chen, T.; Botimer, J.; Chou, T.; Zhang, Z. A 1.87-mm$^2$ 56.9-GOPS Accelerator for Solving Partial Differential Equations. *IEEE J. Solid-State Circuits* **2020**, *55*, 1709–1718. [CrossRef]
32. Feng, Y.; Zhan, X.; Chen, J. Flash Memory based Computing-In-Memory to Solve Time-dependent Partial Differential Equations. In Proceedings of the 2020 IEEE Silicon Nanoelectronics Workshop (SNW), Honolulu, HI, USA, 13–14 June 2020; pp. 27–28.
33. Le Gallo, M.; Sebastian, A.; Mathis, R.; Manica, M.; Giefers, H.; Tuma, T.; Bekas, C.; Curioni, A.; Eleftheriou, E. Mixed-precision in-memory computing. *Nat. Electron.* **2018**, *1*, 246–253. [CrossRef]
34. Jiang, H.; Huang, S.; Peng, X.; Yu, S. MINT: Mixed-Precision RRAM-Based IN-Memory Training Architecture. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5. [CrossRef]