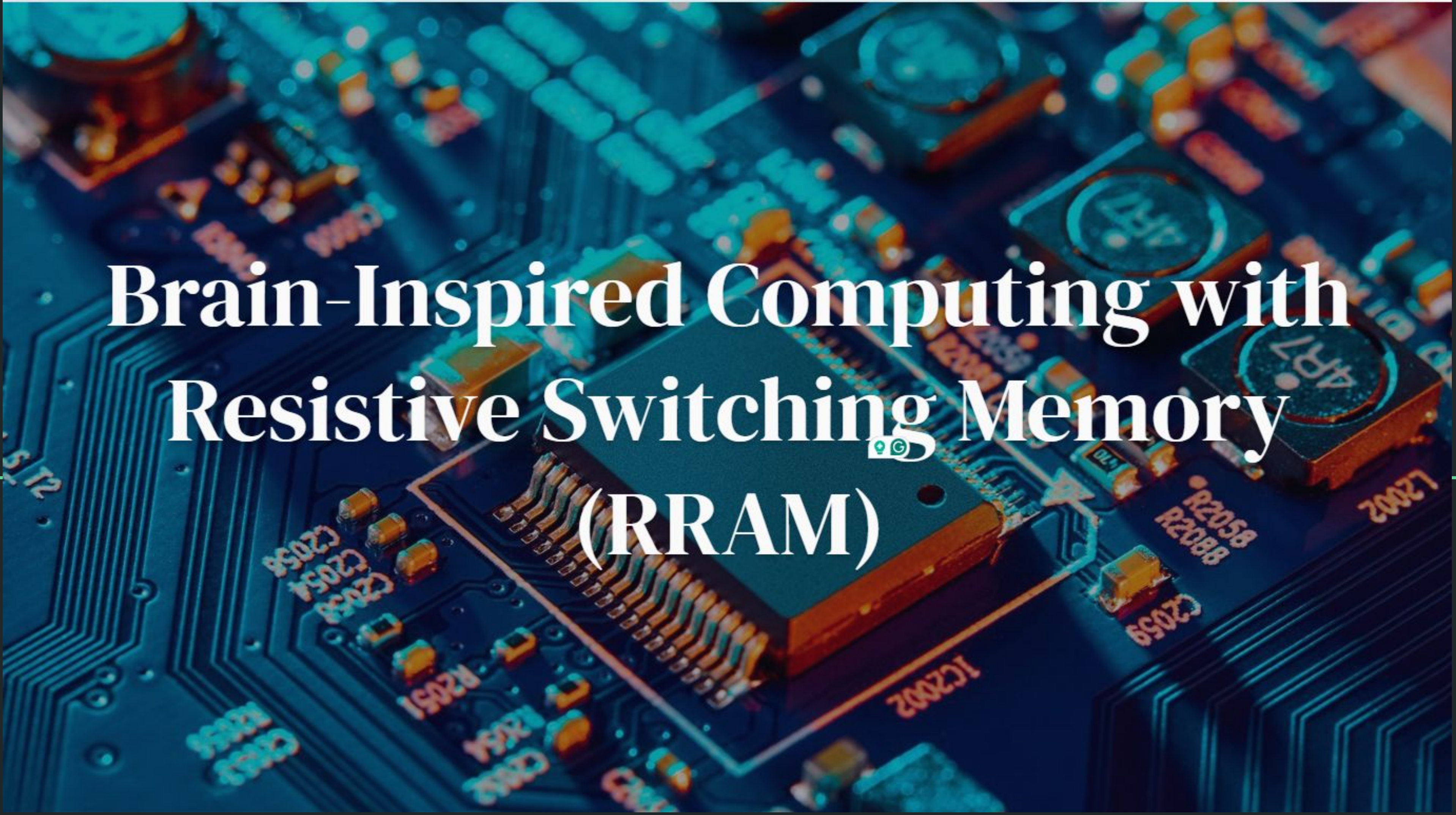


Brain-Inspired Computing with Resistive Switching Memory (RRAM)

Indian Institute of Engineering Science and Technology, Shibpur (IIEST)

Under the guidance of-
Prof. Biplab Kr Sikdar
Dept. of Computer Science & Technology

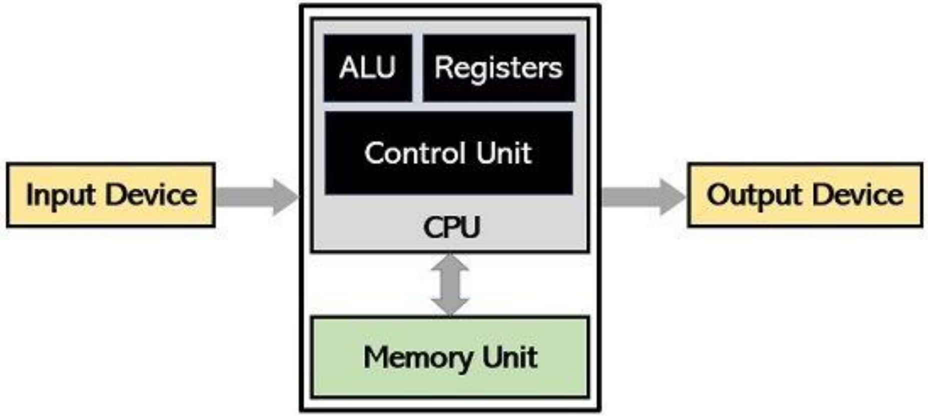
By-

Dipmay Biswas (2021CSB043)

Suman Kayal (2021CSB120)

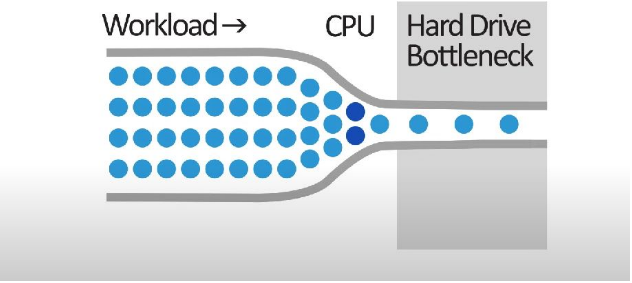
Dt-12/03/2024

Von Neumann Bottleneck



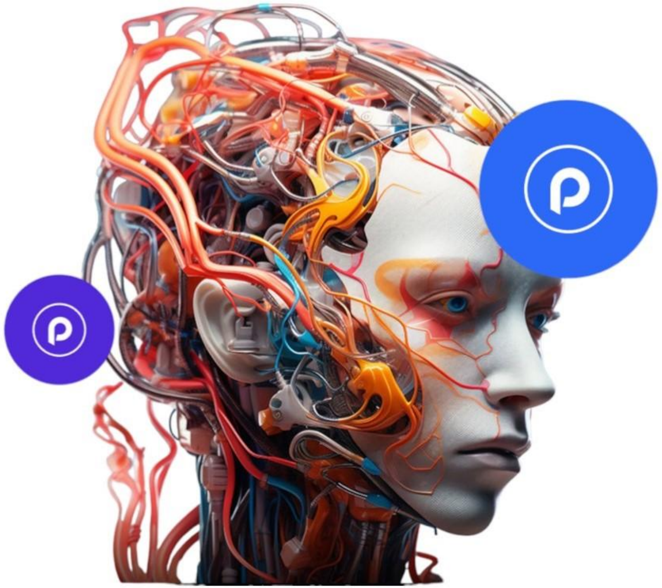
Von Neumann Architecture

VON NEUMANN BOTTLENECK



why do we need new type architecture today. most of the hardware is based on Von neumann architectures which typically separates computer memory and Computing this occurs because Von neumann chips have to settle information back and forth between the memory and the CPU leading to the waste of both memory and energy a problem known as Von neumann Bottleneck problem.

What is Brain Inspired Computing? Or Neuromorphic Computing



The human brain can perform advanced computing tasks, such as learning, recognition, and cognition, with extremely low power consumption and low frequency of neuronal spiking.

This combination of high computing capability and density scalability can be obtained with the nanodevice technology, notably by resistive-switching memory (RRAM) devices.

RRAM devices with improved window and reliability thanks to SiOx dielectric layer are discussed.

Why Neuromorphic Computing ??

Parallel Processing: The brain processes information in parallel across billions of neurons, enabling it to perform complex tasks efficiently.

Low Power Consumption: Despite its immense computational capabilities, the brain operates on very low power consumption compared to traditional computing systems.

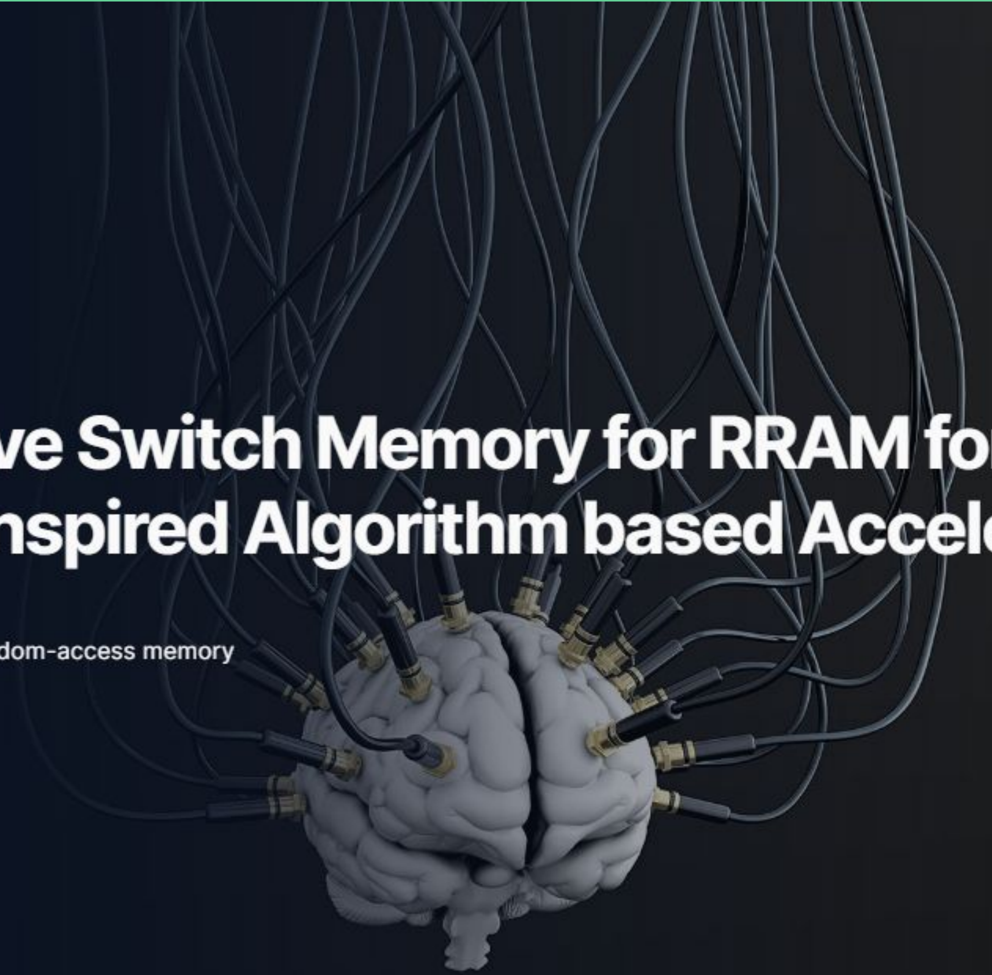
Adaptability and Learning: The brain is capable of learning, adaptation, and self-organization. Neuromorphic computing seeks to incorporate these capabilities into artificial systems.

Fault Tolerance: The brain is remarkably robust and can continue functioning even if individual neurons or connections fail. Neuromorphic computing aims to develop systems with similar fault-tolerant properties.

TrueNorth Chip by IBM

Resistive Switch Memory for RRAM for Brain-Inspired Algorithm based Accelerator

RRAM- Resistive random-access memory



INTRODUCTION

Resistive memory crossbars can drastically reduce the energy required to perform computations in neural algorithms by at least six orders of magnitude when compared to a conventional CPU.

New approaches based on memristor or resistive memory crossbars can enable the processing of large amounts of data by significantly reducing data movement, taking advantage of analog operations, and fitting more memory on a single chip.

Resistive memories are essentially programmable two terminal resistors. If a write voltage is applied to the device, the resistance will increase or decrease based on the sign of the voltage, allowing the resistance to be programmed. At lower voltages, the state does not change.

Consequently, these devices can be used to model a neural synapse wherein the resistance acts like a weight that modulates the voltage applied to it. This has resulted in a large interest in developing neuromorphic systems based on such devices. Ideally, the resistive memories would have a perfectly linear and controllable response allowing them to be programmed to any arbitrary analog value.

Unfortunately, realistic devices have three key non-idealities:

❑ read noise which causes the value read from the resistive memory to be different than the true value,

❑ write noise which causes the value written to be different from the intended value, and

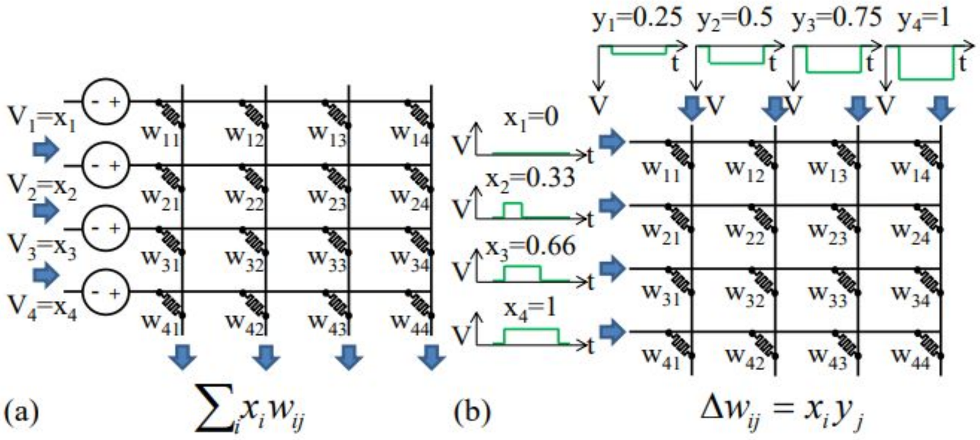
❑ write nonlinearities which means that the change in conductance due to a write pulse will be different depending on the device’s current state.

We use a numerical simulation, written in Python, to model how backpropagation performance is impacted by the three hardware non-idealities for different data sets, and determine device properties required to maintain high learning accuracy.

The key design considerations for an effective neural algorithm accelerator is that it should both reduce the computation energy by orders of magnitude, and it should be flexible enough that it can run many different neural algorithms. In it is shown that a resistive memory crossbar can accelerate two key operations:

1) a parallel read, or vector matrix multiply, and

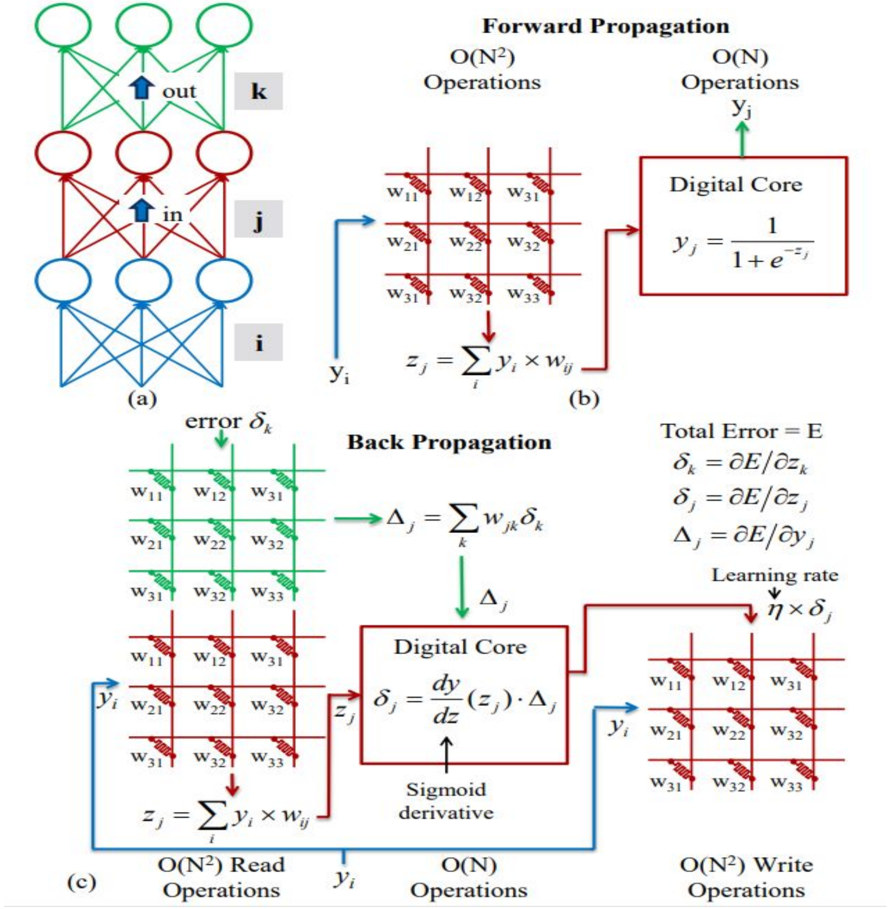
2) a parallel write.



$V_1=x_1$
$V_2=x_2$
$V_3=x_3$
$V_4=x_4$
w_{11} w_{12} w_{13} w_{14}
w_{21} w_{22} w_{23} w_{24}
w_{31} w_{32} w_{33} w_{34}
w_{41} w_{42} w_{43} w_{44}
(a)
$\sum_i x_i w_{ij}$
(b)
$y_1=0.25$ $y_2=0.5$ $y_3=0.75$ $y_4=1$
$x_1=0$
$x_2=0.33$
$x_3=0.66$
$x_4=1$
V
t
$\Delta w_{ij} = x_i y_j$

(a) Analog resistive memories can be used to reduce the energy of a vector-matrix multiply. The conductance of each resistive memory device represents a matrix element or weight. Analog input vector values are represented by the input voltages or input pulse lengths, and output vector values are represented by currents. This allows all the read operations, multiplication operations and sum operations to occur in a single step.

(b) A parallel write is illustrated. Weight Wij is updated by xi×yj. In order to achieve a multiplicative effect the xi are encoded in time while the yj are encoded in the height of a voltage pulse. The resistive memory will only train when xi is nonzero. The height of yj determines the strength of training when xi is nonzero. The column inputs yj can also be encoded in time.



out
k
in
j
i
(a)
Forward Propagation
O(N^2) Operations
O(N) Operations
y_j
w_{11} w_{12} w_{31}
w_{21} w_{22} w_{32}
w_{31} w_{32} w_{33}
y_i
$z_j = \sum_i y_i \times w_{ij}$
Digital Core
$y_j = \frac{1}{1+e^{-z_j}}$
(b)
error δ_k
Back Propagation
Total Error = E
$\delta_k = \partial E / \partial z_k$
$\delta_j = \partial E / \partial z_j$
$\Delta_j = \partial E / \partial y_j$
$\Delta_j = \sum_k w_{jk} \delta_k$
Δ_j
Learning rate
$\eta \times \delta_j$
Digital Core
$\delta_j = \frac{dy}{dz}(z_j) \cdot \Delta_j$
Sigmoid derivative
y_i
z_j
$z_j = \sum_i y_i \times w_{ij}$
(c)
O(N^2) Read Operations
O(N) Operations
O(N^2) Write Operations

A mapping of backpropagation to the neural architecture is shown.

(a) A simple network

(b) In a forward evaluation of a neural network, the vector matrix multiply is done in a neural core and the digital core is used to compute the sigmoid neuron function.

(c) The steps required to update the middle layer weights, wij are shown. The red wij crossbar is shown twice to illustrate the two operations that need to be performed.

Many neural algorithms such as sparse coding, restricted Boltzmann machines, and backpropagation rely heavily on these two operations. The difference in the implementation of these algorithms is how the inputs and outputs of a crossbar are processed.

An NxN crossbar accelerates O(N^2) operations, while it has O(N) inputs or outputs. This means that the energy to process an input or output can cost O(N) times more than the energy to read or write a single resistive memory element without significantly increasing the system energy.

This key insight allows us to optimize the tradeoff between energy efficiency and system flexibility. A crossbar based neural core should be used to perform the parallel vector matrix multiply, while a more general purpose digital core can be used to process the inputs and outputs of the crossbar. To represent both positive and negative matrix values with a resistive device, a reference weight is subtracted.

The inputs and outputs to the neural cores are digital and so analog to digital (A/D) and digital to analog (D/A) converters will be needed. This is energetically expensive, but they are O(N) operations and can therefore be the same order of magnitude as the energy needed to drive the crossbar.

The implementation of backpropagation on the general purpose neural architecture is illustrated in Fig. The crossbars process O(N^2) operations while the digital cores handle O(N) operations.

The brain has an extremely dense connectivity where individual neurons in the cerebral cortex can receive roughly 10,000 input synapses from other neurons. To achieve this, the brain takes full advantage of its 3D structure to minimize connection lengths. Currently, high performance CMOS is 2D or at most 2.5D and so it is impossible to hardwire the same number of connections. Consequently, a shared bus is needed to emulate the same connection density.

Overall, to obtain maximum energy efficiency, this type of a system assumes that a neural network has dense local connections that can be mapped to a crossbar. Computation is localized to the maximum extent possible, which minimizes the amount of high energy cost, longer range communications that are required. A dense local and sparse global connectivity is similar to how the brain is organized. If this is not the case for a given algorithm, a single column of a crossbar can be used in a specific read or write step to allow for maximum flexibility at higher energy cost.

Questions ??

THANK YOU !!