IEEE Xplore® Digital Library

2020 IEEE International Symposium on Circuits and Systems
Virtual, October 10-21, 2020

In-Memory Multiplication Based on Deterministic Stochastic Computing

Mohsen Riahi Alam[*], M. Hassan Najafi[*] , Nima TaheriNejad[‡]

[*]University of Louisiana at Lafayette, Louisiana, USA
[‡]TU Wien, Vienna, Austria

2020 IEEE International Symposium on Circuits and Systems

Virtual, October 10-21, 2020

Overview

- **An introduction to:**
  - In-Memory Computation (IMC)
  - Stochastic computing(SC)
- **Background**
  - Deterministic Computation with Stochastic Bit-Streams
  - Stochastic Computing and Memristors
  - Memristive In-Memory Computation
- **Proposed Multiplication Method**
  - Binary to Bit-Stream
  - Stochastic Multiplication using MAGIC
  - Bit-Stream to Binary
- **Results and Comparison**
  - Circuit-Level Simulations
  - Comparison with In-Memory Binary Multiplication
  - Comparison with Off-Memory Stochastic Multiplication
- **Conclusion**

Introduction

- **In-Memory Computation (IMC)**
  - **Transferring** data is expensive (Von-Neumann bottleneck).
  - **Memristors** offer to tackle this challenge via computing-in-memory (CIM).
  - The ability to both **store** and **process** within memory.
  - We use Memristor-Aided Logic (**MAGIC**) [1]
  - To **generate** bit-streams, we propose a method which takes advantage of the intrinsic properties of memristors.
  - **NOR** and **NOT** logical operations can be executed within memristors and with a high degree of **parallelism.**
  - Memristive technology is a not a **fully mature technology** yet compared to CMOS

[1] S. Kvatinsky et al, MAGIC—memristor-aided logic. TCASII'2014

Introduction

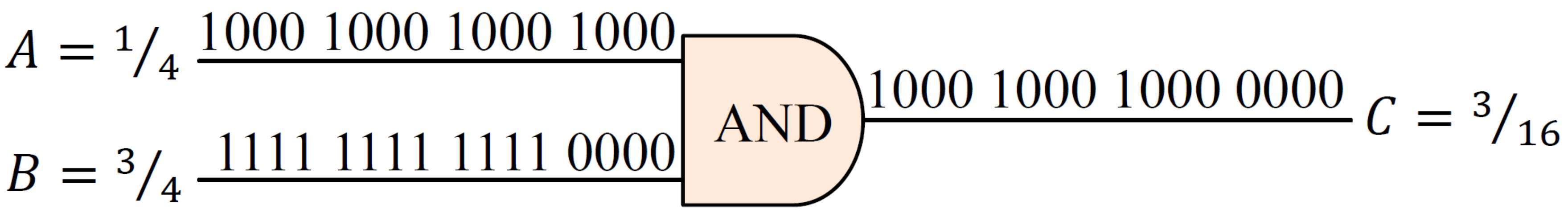
- **Stochastic computing (SC)**
  - A re-emerging computing paradigm, first introduced in **1960s.**
  - Data is represented by **streams** of 0s and 1s, numbers limited to the [0, 1] interval.
  - Logical computation on **random** (or **unary**) bit-streams.
  - All digits have the **same** weight.
  - The **ratio** of the number of 1s to the length of the bit-stream determines the data value.

    e.g., 11100, 10101, 1011011100 → 0.6

  - **Value**: probability of obtaining a one versus a zero.
  - More **noise-tolerant.**
  - An **approximate** computing approach for many years.
  - **Deterministic** approaches have been proposed recently -> **Completely accurate results**

Deterministic Multiplication

- **Deterministic** and **accurate** multiplication using stochastic bit-streams [2]
  - Multiplication is performed by **AND**ing the input bit-streams.
  - **Clock dividing** bit-streams guarantees generating **independent** bit-streams.
  - A=1/4 = 1000 → 1000 1000 1000 1000
  - B=3/4 = 1110 → 1111 1111 1111 0000

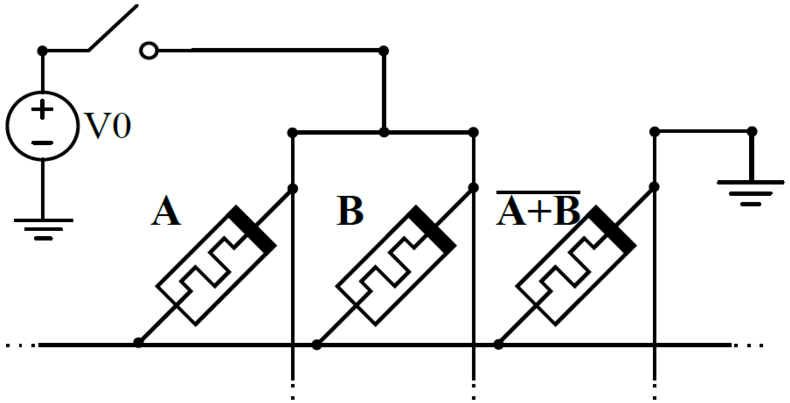


Example of multiplication using clock division method.

[2] Jenson and Riedel, A Deterministic Approach to Stochastic Computation. ICCAD'2016

Memristive In-Memory Computation

- ReRAMs (Memristors) are two-terminal electronic devices with a **variable** resistance.
- **High** and **low** resistances are considered as logical **zero** and **one**.
- Two stateful logic families proposed for **IMC**
  - Material Implication (IMPLY) [3]
  - Memristor Aided LoGIC (MAGIC) [4]
- **MAGIC** characteristics:
  - Compatible with the usual **crossbar.**
  - Requires a **lower** number of voltages.
  - Supports **NOR** (can implement any Boolean logic).
  - Operation can be executed by **applying** specific voltages to the input(s).



A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

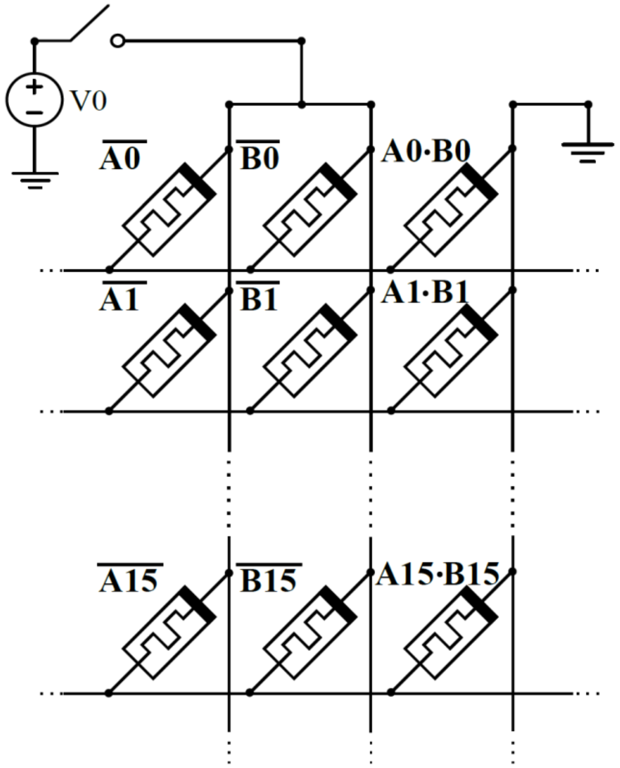
Performing a MAGIC **NOR** operation within a memristive memory

[3] Eero Lehtonen et al. Stateful implication logic with memristors. IEEE/ACM International Symposium on Nanoscale Architectures'09
[4] S. Kvatinsky et al, MAGIC—memristor-aided logic. TCASII'2014

Memristive In-Memory Computation

- Bit-stream **multiplication** is performed by **AND** operation
- MAGIC **AND** is not crossbar compatible.
- MAGIC **NOR** and **NOT** are crossbar compatible.
- **AND** operation performs logical **NOR** on **negated** version of inputs.

$$\overline{\bar{A} + \bar{B}} = A \cdot B$$

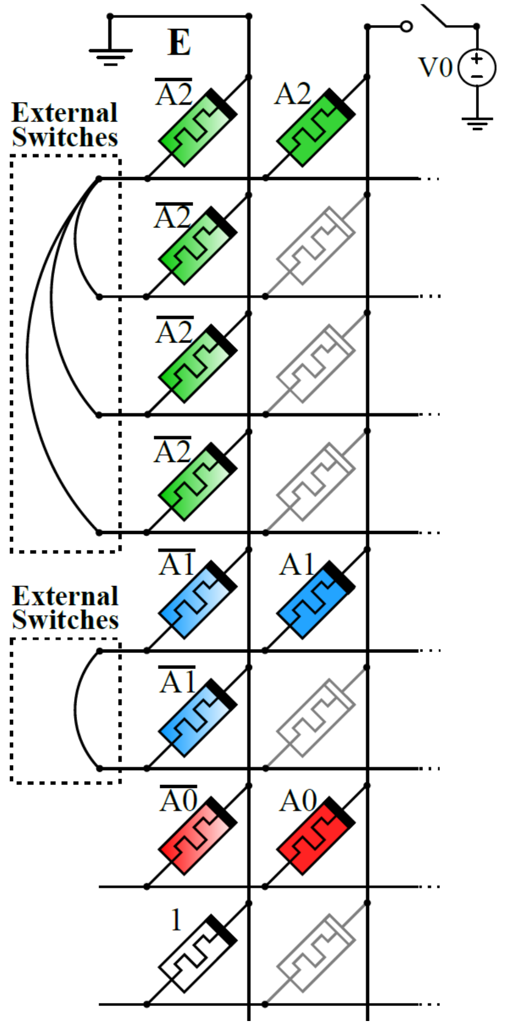


$\bar{A}$	$\bar{B}$	A·B
1	1	0
1	0	0
0	1	0
0	0	1

Performing AND operation using MAGIC NOR.

Binary to Bit-Stream

- **Convert** the binary data (N-bit) to deterministic and accurate bit-streams (2^N-bit).
- **Initialize** 2^N memristors in a column to Low Resistance State (LRS) .
- Turn on the **external switches** to make proper connection based on the bit-stream generation method
  - **Clock division**
- Apply **V0** to the positive terminal of the input binary memristors.
- This operation makes complement results similar to MAGIC **NOT** operation.
- We have a representation which is **complementary** to conventional bit-stream representation.
- This **complementing** is to our advantage as it reduces the number of steps necessary to perform multiplication



3-bit precision binary to bit-stream conversion in memory

Stochastic Multiplication using MAGIC

- To perform crossbar compatible **AND** in MAGIC, the input operands need to be **inverted**, followed by a **NOR** operation.
- Our bit-streams are **already** in their inverted form.
- So the multiplication only needs one **MAGIC NOR** operation between the two bit-streams.
- 2^2N and 2^N memristors for bit-stream representation for exact (full) and limited precision multiplication.
- Example of **full-precision** multiplication:
  - Inputs are **A = 1/4** and **B = 3/4** in binary format.
  - **The clock division method** is used to make A and B operand bit-streams.
  - The output bit-stream **S** represents **3/16.**

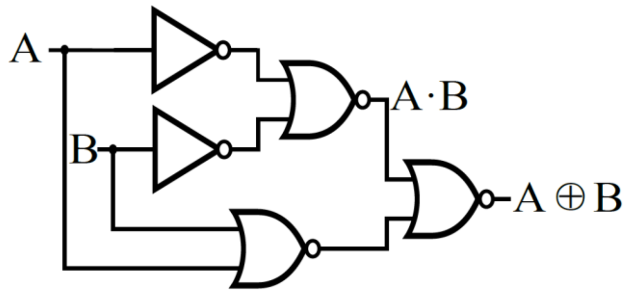
2-input bit-stream-based multiplication using the proposed method

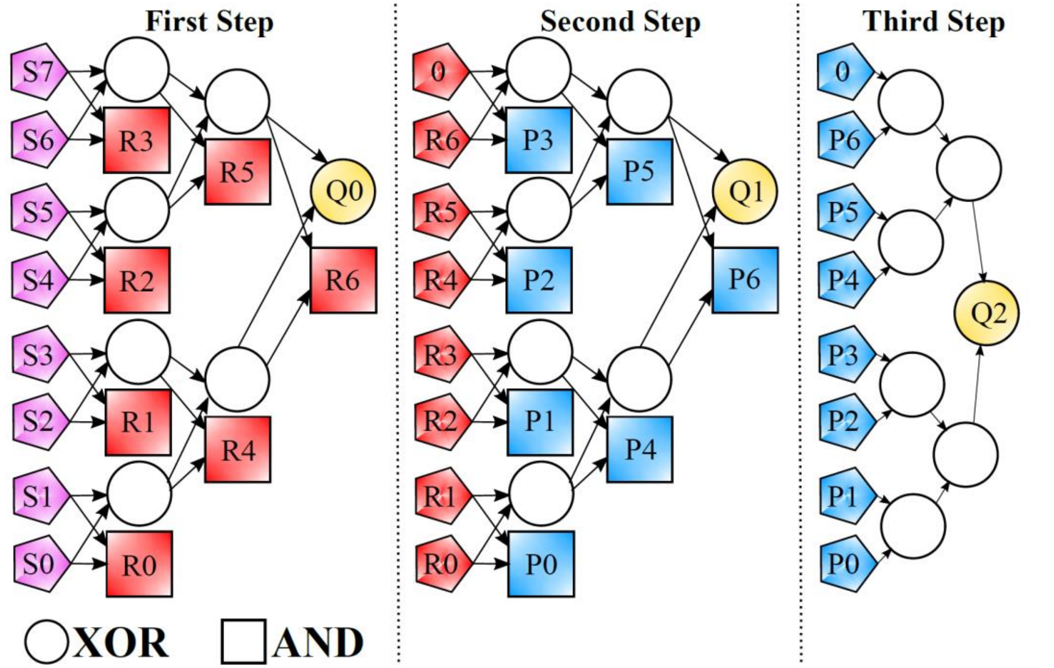
Bit-Stream to Binary

- When the output is desired in **binary** format.
- **Counting** the number of **1s** in the bit-stream.
- We suggest two **conversion** methods:
  - In-memory conversion
  - Off-memory conversion
- **Off-memory conversion**:
  - Output bitstream is **read** from the memory.
  - Off-memory **CMOS** circuit is used for accumulation.
  - We let the synthesis tool implement the **best** design.

In-memory conversion

- An in-memory algorithm is proposed to count the number of 1s in the bit-stream.
- The algorithm consists of **XOR** and **AND** operations
  - Implemented by MAGIC **NOR** and **NOT** operations.
- The output of AND operation is used for XOR operation
- **Re-using** memristors to minimize the number of required memristors.
- **Scalable:** to convert longer bit-stream.
- Calculate one bit (Qi) in each step.
- AND operations outputs are used as input of the next step
- Final Step consists of XOR operations.
- Example:
  - Proposed algorithm for 8-bit bit-stream (S7-S0) as input.
  - Output is a 3-bit binary (Q2-Q1-Q0).





Counting 1s in memory using XOR and AND operations.

Circuit-Level Simulations

- A 16×16 crossbar and necessary control signals in **LTSpice.**
- Voltage-controlled Threshold Adaptive Memristor (**VTEAM**) model[4]

R_{on}	R_{off}	VT_{on}	VT_{off}	x_{on}	x_{off}	k_{on}	k_{off}	α_{on}	α_{off}
$1k\Omega$	$300k\Omega$	$-40mV$	$300mV$	$0nm$	$3nm$	-100 m/sec	0.091 m/sec	4	4

- We evaluated **all** cases of 2-bit precision multiplication and verified the functionality of the design.

[4] S. Kvatinsky et al, Vteam: A general model for voltage-controlled memristors. IEEE TCAS II, Aug 2015.

Comparison with In-Memory Binary Multiplication

- Comparing **latency** and **area** with prior in-memory fixed-point multiplication methods [5][6]
- Our proposed multiplier is **significantly faster** than the prior methods.
- Our method is more **area efficient** for N < 5 for the limited precision.

Methods		Latency (Cycles)	Area (# of memristors)
Full Precision	Haj-Ali *et al.* [6]	$13N^2 - 14N + 6$	$20N - 5$
	Imani *et al.* [5]	$15N^2 - 11N - 1$	$15N^2 - 9N - 1$
	This work	3	3×2^{2N}
Limited Precision	Haj-Ali *et al.* [6]	$6.5N^2 - 7.5N - 2$	$19N - 19$
	This work	3	3×2^N

[5] Mohsen Imani et al, Ultra-Efficient Processing In-Memory for Data Intensive Applications. DAC'2017
[6] Haj-Ali et al, Efficient algorithms for in-memory fixed point multiplication using MAGIC. ISCAS'2018

Comparison with Off-Memory Stochastic Multiplication

- We compared the **energy consumption** of **3** versions of our **in-memory** design with the **off-memory** bit-stream-based multiplier
- Our in-memory design +
  - In-memory bit-stream-to-binary conversion.
  - Off-memory bit-stream-to-binary conversion.
  - Without bit-stream-to-binary conversion.
- We implemented the off-memory **CMOS** design in **45nm technology**
  - Data is read/written from/to a memristive memory
  - Per bit energy consumption was extracted from [7]

ENERGY CONSUMPTION COMPARISON (pJ)

Design Method	Limited Precision							Full Precision						
	N=2	3	4	5	6	7	8	N=2	3	4	5	6	7	8
This work (no bit-stream-to-binary conversion)	0.04	0.07	0.14	0.28	0.56	1.13	2.26	0.14	0.56	2.3	9.0	36	145	578
This work (+ in-memory bit-stream-to-binary)	0.05	0.11	0.23	0.49	1.03	2.20	4.64	0.23	1.03	3.7	21	90	393	1702
This work (+ off-memory bit-stream-to-binary)	9	17	31	58	110	212	416	31	110	416	1628	6462	25561	102184
Off-Memory Exact SC-based Multiplication	38	40	44	53	76	124	234	54	76	133	694	3092	13919	62541

[7] J Joshua Yang et al,Memristive devices for computing. Nature nanotechnology 2013.

Discussion and Conclusions

- Here, we proposed the first **in-memory** architecture to execute **exact** SC-based multiplication
- Proposed a crossbar compatible method for bit-stream-to-binary conversion.
- The proposed method significantly **reduces** the **energy** consumption compared to the SoA exact off-memory SC-based multiplier.
- Compared to prior in-memory fixed-point multiplication methods, the proposed design provides **faster** results.
  - For smaller Ns, the **area** is lower or comparable too.
  - For larger Ns, the **area** is the price for the gained speed.

Thank you

Questions?

mohsen.riahi-alam@louisiana.edu,
najafi@louisiana.edu,
nima.taherinejad@tuwien.ac.at