SCIENCE ▪ PASSION ▪ TECHNOLOGY

Institute of Information Security ■ Inffeldgasse 16a, 8010 Graz, Austria ■

Exploring Large Integer Multiplication for Cryptography Targeting In-Memory Computing

Florian Krieger, Florian Hirner, Sujoy Sinha Roy

{florian.krieger, florian.hirner, sujoy.sinharoy}@tugraz.at

In-Memory Computing (IMC) meets Cryptography

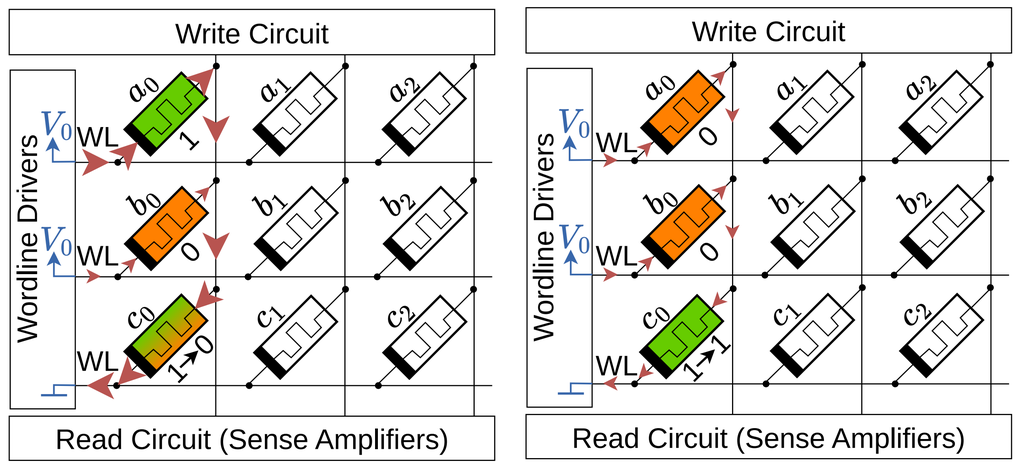
- Von Neumann architectures suffer from CPU ⟷ Memory data streams
  - ➔ Energy consumption ➔ Latency
  - ➔ Especially in data-heavy applications
- Alternative: **In-Memory Computing**
  - ➔ Process data where it is stored
  - ➔ Compute-enabled memory array
  - ➔ Avoids expensive data streams
- Many data-heavy crypto-schemes:
  - ➔ Fully Homomorphic Encryption
  - ➔ Zero-Knowledge Proofs
- IMC + Cryptography: Good fit BUT insufficiently researched!
- **Supporting large integer arithmetic is a key challenge!**

How IMC works

- We use the MAGIC [1] technology
- Uses non-volatile Resistive RAM (ReRAM):
  - High resistance (HR, logic 0)
  - Low resistance (LR, logic 1)
- Performs bit-wise NOR in a SIMD manner

NOR Truth Table

a	b	$c = \text{NOR}(a, b)$
0 (HR)	0 (HR)	1 (LR)
0 (HR)	1 (LR)	0 (HR)
1 (LR)	0 (HR)	0 (HR)
1 (LR)	1 (LR)	0 (HR)



Functionality: a_0 and b_0 are the input operands and c_0 (initialized to LR) should hold the NOR result. First, a voltage V_0 is applied to input word lines (WL) causing a current through c_0 (red arrows). If at least one input resistor is in LR (left figure), the large current in c_0 will switch c_0 to HR. Otherwise, the current in c_0 is small, and c_0 remains in LR (right figure). This applies to all columns in parallel.

Our Proposed IMC Multiplier for Large Integers

Contribution 1: Algorithmic Exploration

➔ Large integer multiplication needs an efficient multiplication algorithm
➔ Runtime, Area, IMC-suitability

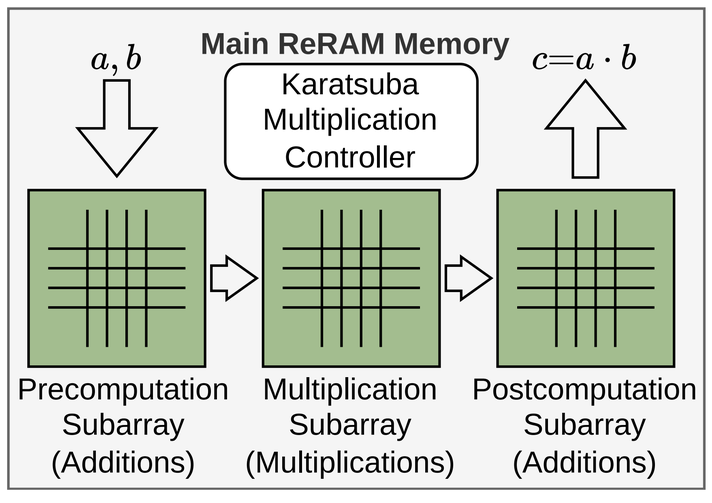
- **Schoolbook method:**
  - ⊖ $O(n^2)$ complexity ➔ Bad for efficiency
- **Toom-Cook method:**
  - ⊕ Good asymptotic complexity
  - ⊖ Concrete efficiency is difficult
  - ⊖ Unfavorable Vandermonde matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/2 & 1/3 & -1 & 1/6 & -2 \\ -1 & 1/2 & 1/2 & 0 & -1 \\ -1/2 & 1/6 & 1/2 & -1/6 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Karatsuba method: Our Choice**
  - ⊖ Slightly slower than Toom-Cook
  - ⊕ Lower complexity than schoolbook
  - ⊕ Suitable for IMC

Contribution 2: Karatsuba Design for IMC

- Unrolled Karatsuba multiplication
- 3-stage pipeline
  - ➔ Independent memory subarrays
  - ➔ Parallel processing



Optimizations:

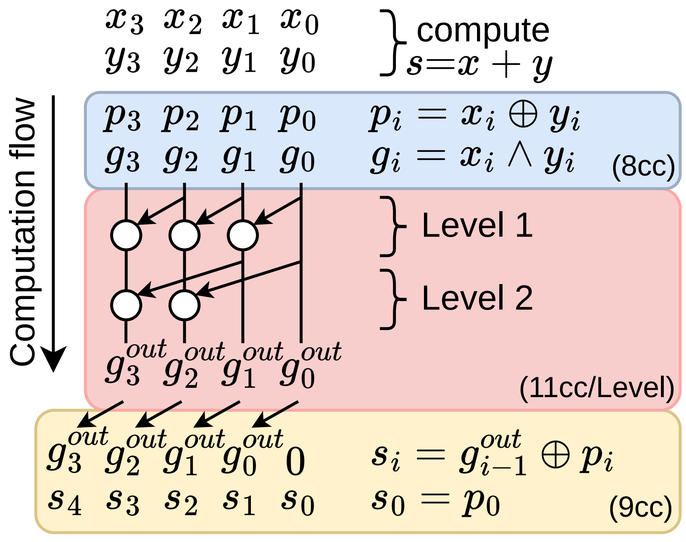
- Resource sharing: Reduces area
- Load balancing: Prolongs cell lifetime

Contribution 3: Kogge-Stone Adder

- Pre- and postcomputation arrays perform repeated additions

➔ Requires efficient IMC addition

- Kogge-Stone adder:
  - ⊕ Logarithmic depth
  - ⊕ Regular structure
  - ⊕ Good choice for IMC



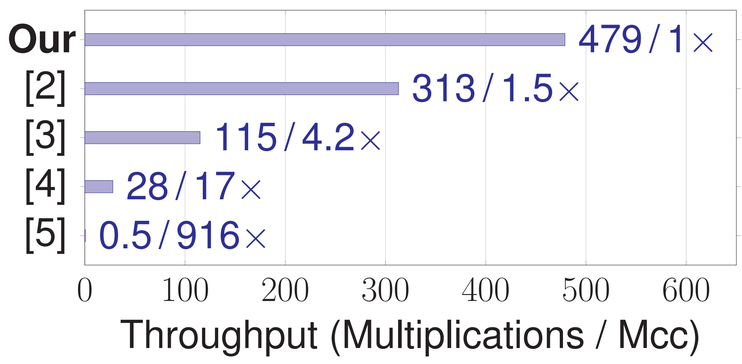
Results & Comparison

This work presents the **first Karatsuba multiplier for IMC**. The figures below present & compare our results for **384-bit** multiplication.

Throughput:

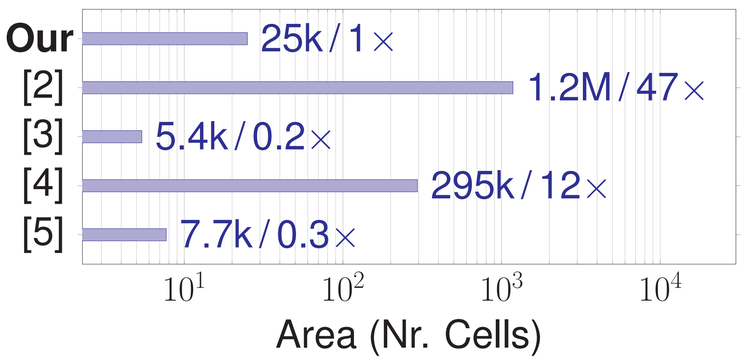
- Between 1.5× and 916× improvement

➔ Highest throughput reported in literature



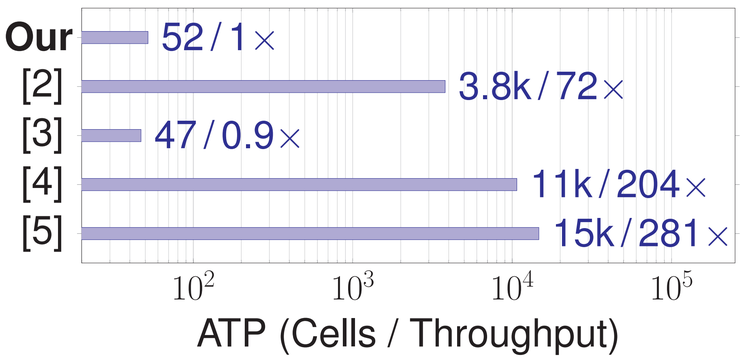
Area:

- We reach a balanced area consumption
- 2nd fastest work [2] is 47× larger



Area-Time Product (ATP):

- [3] has lower ATP but very long WLs
- Our work benefits from shorter WLs



References:

[1] S. Kvatinsky *et. al*, “Magic - Memristor-Aided Logic,” *IEEE Transactions on Circuits and Systems II*, 2014.
[2] V. Lakshmi *et al.*, "A novel in-memory wallace tree multiplier architecture using majority logic", *IEEE Transactions on Circuits and Systems I*, 2022.
[3] O. Leitersdorf *et al.*, "Multpim: Fast stateful multiplication for processing-in-memory", *IEEE Transactions on Circuits and Systems II*, 2022.
[4] D. Radakovits *et al.*, "A memristive multiplier using semi-serial imply-based adder", *IEEE Transactions on Circuits and Systems I*, 2020.
[5] A. Haj-Ali *et al.*, "Imaging: In-memory algorithms for image processing", *IEEE Transactions on Circuits and Systems I*, 2018.