北京大学高能效计算与应用中心
Center for Energy-efficient Computing and Applications

Parallel Stateful Logic in RRAM: Theoretical Analysis and Arithmetic Design

Feng Wang, Guojie Luo, Guangyu Sun, Jiaxi Zhang, Peng Huang, Jinfeng Kang

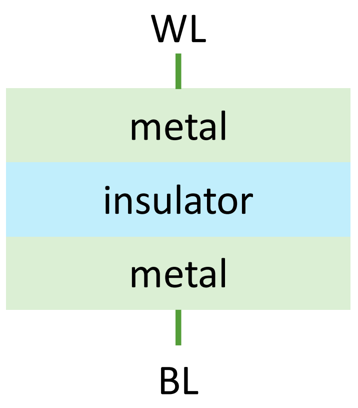
gluo@pku.edu.cn

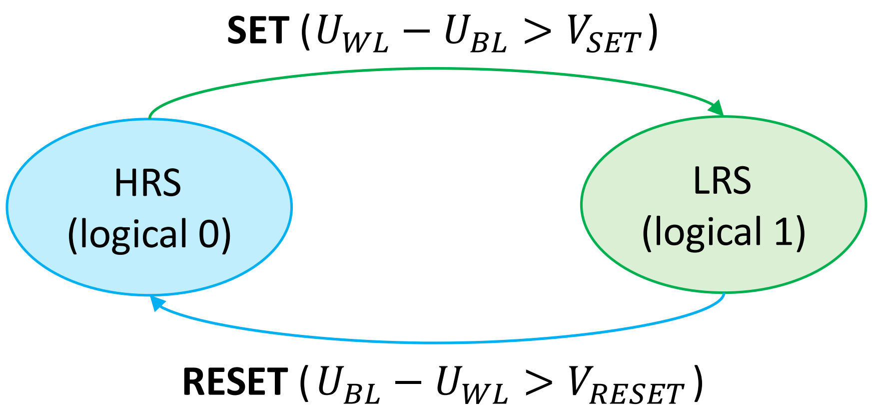
2019/07/16

OUTLINE

- Background
- Theoretical Analysis
- Integer Addition
- Extension
- Experimental Evaluation
- Conclusion

RRAM Resistance for Representing Logic Values

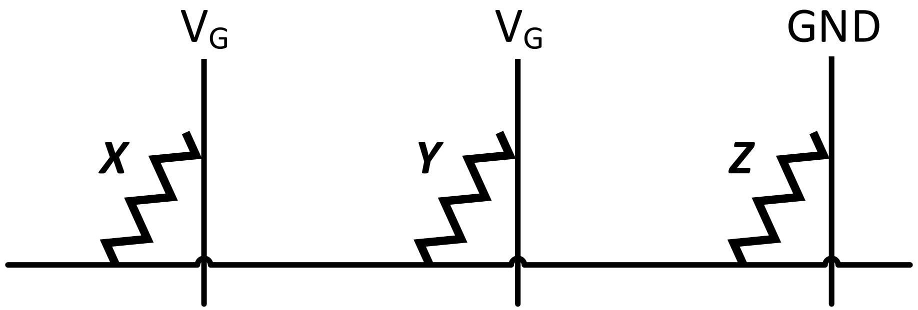


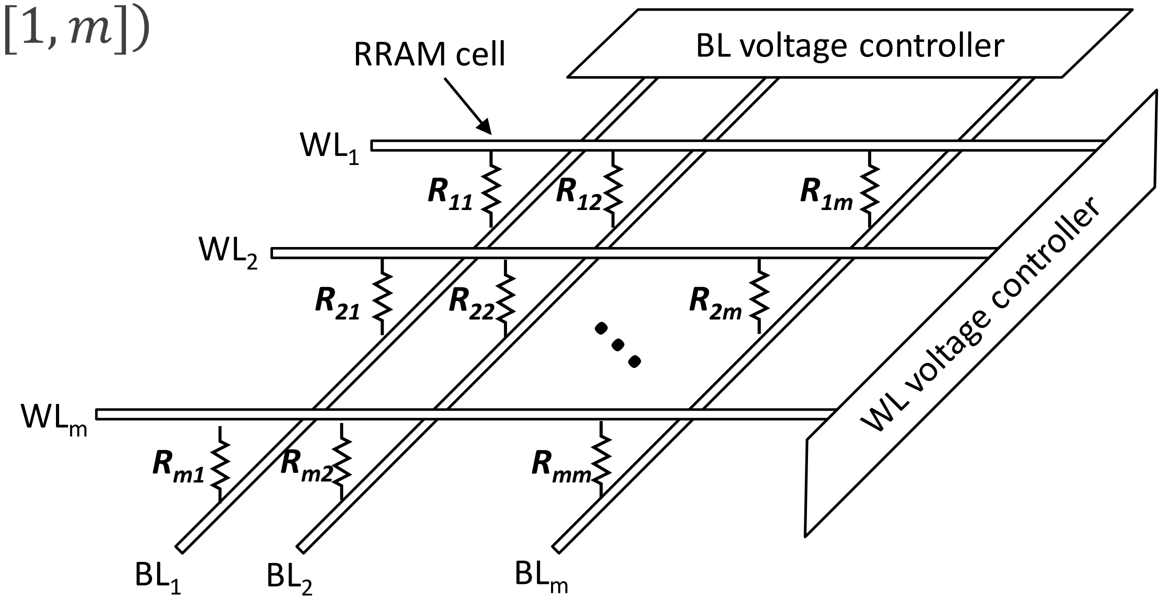


[1] Akinaga, H., & Shima, H. (2010). Resistive random access memory (ReRAM) based on metal oxides. *Proceedings of the IEEE*.

RRAM State Switching as Primitive Logic Operations

- MAGIC NOR implementation $Z = \mathrm{NOR}(X, Y)$
  - $V_G > 2 \cdot V_{RESET}$
- Parallel execution over rows and columns
  - WL parallelism: $R_{im} = \mathrm{NOR}(R_{i1}, R_{i2})(i \in [1, m])$
  - BL parallelism: $R_{mi} = \mathrm{NOR}(R_{1i}, R_{2i})(i \in [1, m])$





[2] Kvatinsky, S., Member, S., Belousov, D., Liman, S., Satat, G., Member, S., … Weiser, U. C. (2014). MAGIC — Memristor-Aided Logic. *TCAS-II*.

RRAM-based Stateful Logic Families

- Support parallel execution
- Functionally complete

Work	Stateful logic operations
[3]	IMP
[2]	NOR, NOT
[4]	NAND, NIMP
[5]	NOR, NAND, Min, OR
[6]	NOR, NOT, NAND, NIMP, XOR

[3] Borghetti, J., Snider, G. S., Kuekes, P. J., Yang, J. J., Stewart, D. R., & Williams, R. S. (2010). "Memristive" switches enable "stateful" logic operations via material implication. *Nature*.
[4] Huang, P., Kang, J., Zhao, Y., Chen, S., Han, R., Zhou, Z., … Liu, X. (2016). Reconfigurable Nonvolatile Logic Operations in Resistance Switching Crossbar Array for Large-Scale Circuits. *Advanced Materials*.
[5] Avati, V., Eggert, K., & Taylor, C. (2018). FELIX: Fast and Energy-Efficient Logic in Memory. In *ICCAD*.
[6] Xu, L., Bao, L., Zhang, T., Yang, K., Cai, Y., Yang, Y., & Huang, R. (2018). Nonvolatile memristor as a new platform for non-von Neumann computing. In *ICSICT*.

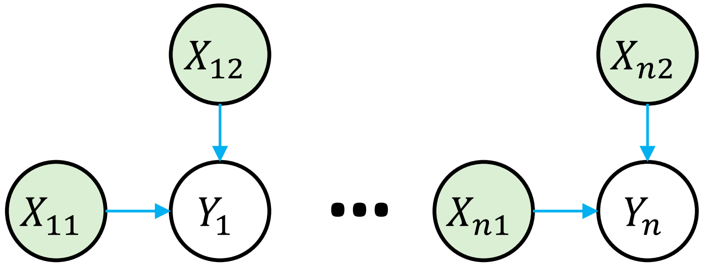
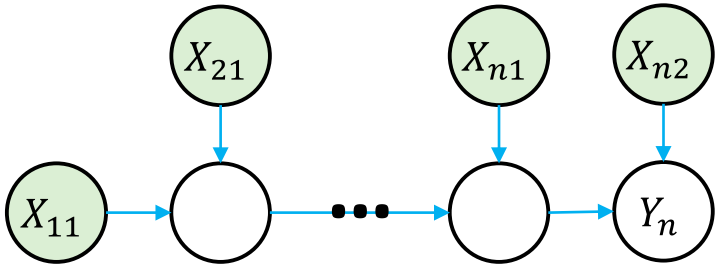
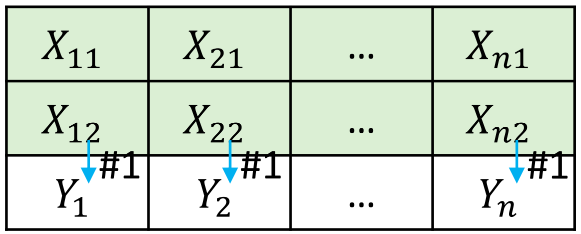
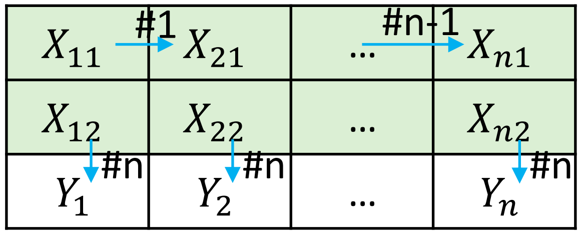
OUTLINE

- Background
- Theoretical Analysis
- Integer Addition
- Extension
- Experimental Evaluation
- Conclusion

SIML Computation Model

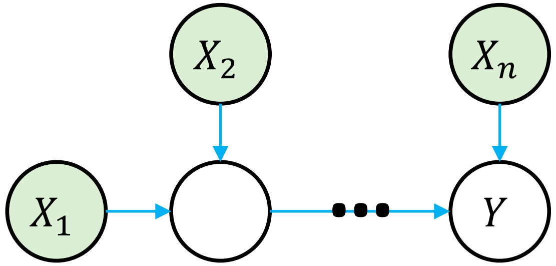
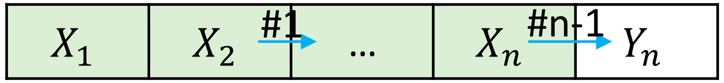
- All of the stateful logic families satisfy the four assumptions
  - The latency of a single stateful logic operation is identical.
  - The input number of a single operation cannot exceed a constant.
  - The latency of WL and BL operations is identical.
  - The degree of parallelism can reach the crossbar size. The crossbar size scales with the problem size.

Lower Bounds of the Time Complexity (1)

<table>
<tr><th></th><th>(a) Theorem 1</th><th>(b) Theorem 2</th></tr>
<tr><td>Condition</td><td>bitwise functions</td><td>most arithmetic functions</td></tr>
<tr><td>Parallelism upper bound</td><td>$\max(w,h)$</td><td>$O\left(\frac{T}{w+h}\right)$</td></tr>
<tr><td>Time complexity lower bound</td><td>trivial bound: $O\left(\frac{T}{\max(w,h)}\right)$</td><td>shape bound: $O(w+h)$</td></tr>
<tr><td>Example function</td><td>$Y_i = X_{i1}\ \mathrm{NOR}\ X_{i2}\ (i = 1, 2, \dots, n)$</td><td>$Y_i = \mathrm{NOR}_{k=1}^{i} X_{k1}\ \mathrm{NOR}\ X_{i2}\ (i = 1, 2, \dots, n)$</td></tr>
<tr><td>Example algorithm (netlist)</td><td></td><td></td></tr>
<tr><td>Example layout in RRAM</td><td></td><td></td></tr>
</table>

$O(T)$: total cycles in the series implementation, w: width of layout (# of BLs), h: height of layout (# of WLs), len_{max}: length of the critical path.

Lower Bounds of the Time Complexity (2)

	(c) Corollary 1	(d) Theorem 3
Condition	square layout	a given algorithm
Parallelism upper bound	$O(\frac{T}{\sqrt{wh}})$	$O\left(\frac{T}{len_{max}}\right)$
Time complexity lower bound	***function bound***: $O(\sqrt{wh})$	***algorithm bound***: len_{max}
Example function	—	$Y = \mathrm{NOR}_{k=1}^{n} X_{k1}$
Example algorithm (netlist)	—	
Example layout in RRAM	—	

$O(T)$: total cycles in the series implementation, w: width of layout (# of BLs), h: height of layout (# of WLs), len_{max}: length of the critical path.

OUTLINE

- Background
- Theoretical Analysis
- Integer Addition
- Extension
- Experimental Evaluation
- Conclusion

Ripple Carry Adder

time complexity: $O(n)$
shape / algorithm lower bound

One-bit full adder

Pulse	Logic operation
1	$T_1 = \mathrm{NOR}(A, B)$
2	$T_2 = \mathrm{NOR}(A, T_1)$
3	$T_3 = \mathrm{NOR}(B, T_1)$
4	$T_4 = \mathrm{NOR}(T_1, T_2)$
5	$T_5 = \mathrm{NOR}(T_4, C_i)$
6	$C_o = \mathrm{NOR}(T_1, T_5)$
7	$T_6 = \mathrm{NOR}(T_4, T_5)$
8	$T_7 = \mathrm{NOR}(T_5, C_i)$
9	$S = \mathrm{NOR}(T_5, T_7)$

$\boldsymbol{A_{11}}$	$\boldsymbol{A_{12}}$	…	$\boldsymbol{A_{1n}}$
$\boldsymbol{A_{21}}$	$\boldsymbol{A_{22}}$	…	$\boldsymbol{A_{2n}}$
T_{11}	T_{12}	① …	T_{1n}
…	…	…	…
T_{41}	T_{42}	…	T_{4n}
T_{51}	T_{52}	…	T_{5n}
C_{i1}	C_{o2} ②	…	C_{on}
C_{o1}	$C_{o1}(C_{i2})$	…	$C_{o(n-1)}(C_{in})$
T_{61}	T_{62}	…	T_{6n}
…	…	③ …	…
$\boldsymbol{S_1}$	$\boldsymbol{S_2}$	…	$\boldsymbol{S_n}$

①③ add n BLs in parallel
② generate and propagate carries in series

Carry Select Adder

time complexity: $O(\sqrt{n})$
function / algorithm lower bound

$\boldsymbol{A_{11}}$	…	$\boldsymbol{A_{1\sqrt{n}}}$	$\boldsymbol{A_{21}}$	…	$\boldsymbol{A_{2\sqrt{n}}}$	0	S_1	…	$S_{\sqrt{n}}$
$\boldsymbol{A_{1(\sqrt{n}+1)}}$	…	$\boldsymbol{A_{1(2\sqrt{n})}}$	$\boldsymbol{A_{2(\sqrt{n}+1)}}$	…	$\boldsymbol{A_{2(2\sqrt{n})}}$	0	$S_{\sqrt{n}+1}$	…	$S_{2\sqrt{n}}$
…	…	…	…	…	…	…	…	…	…
$\boldsymbol{A_{1(n-\sqrt{n}+1)}}$	…	$\boldsymbol{A_{1n}}$	$\boldsymbol{A_{2(n-\sqrt{n}+1)}}$	…	$\boldsymbol{A_{2n}}$	0	$S_{n-\sqrt{n}+1}$	…	S_n'
$A_{1(\sqrt{n}+1)}'$	…	$A_{1(2\sqrt{n})}'$	$A_{2(\sqrt{n}+1)}'$	…	$A_{2(2\sqrt{n})}'$	1	$S_{\sqrt{n}+1}'$	…	$S_{2\sqrt{n}}'$
…	…	…	…	…	…	…	…	…	…
$A_{1(n-\sqrt{n}+1)}'$	…	A_{1n}'	$A_{2(n-\sqrt{n}+1)}'$	…	A_{2n}'	1	$S_{n-\sqrt{n}+1}'$	…	S_n'
…	…	…	…	…	…		$\boldsymbol{S_1}$	…	$\boldsymbol{S_{\sqrt{n}}}$
…	…	…	…	…	…		$\boldsymbol{S_{\sqrt{n}+1}}$	…	$\boldsymbol{S_{2\sqrt{n}}}$
…	…	…	…	…	…		…	…	…
…	…	…	…	…	…		$\boldsymbol{S_{n-\sqrt{n}+1}}$	…	$\boldsymbol{S_n}$

① copy the input
② add $2\sqrt{n}-1$ WLs in parallel
③ select the correct result

Carry Save Adder

time complexity: $O(\sqrt{M})$
function lower bound

a_1	…	$a_{\sqrt{M}}$	A_1
$a_{\sqrt{M}+1}$	…	$a_{2\sqrt{M}}$	A_2
…	…	…	
$a_{M-\sqrt{M}+1}$	…	a_M	$A_{\sqrt{M}}$
…	…	…	$\boldsymbol{\sum a}$

① accumulate each WL in parallel
② accumulate the sums in the same BLs

$\boldsymbol{A_{11}}$	$\boldsymbol{A_{12}}$	…	$\boldsymbol{A_{1n}}$
$\boldsymbol{A_{21}}$	$\boldsymbol{A_{22}}$	…	$\boldsymbol{A_{2n}}$
$\boldsymbol{A_{31}}$	$\boldsymbol{A_{32}}$	…	$\boldsymbol{A_{3n}}$
…	…	…	…
C_{o1}	C_{o2}	…	C_{on}
PS_1	PS_2	…	PS_n

② shift C_{o1}, …, C_{on} right

0	C_{o1}	…	$C_{o(n-1)}$
PS_1	PS_2	…	PS_n
…	…	…	…
$\boldsymbol{S_1}$	$\boldsymbol{S_2}$	…	$\boldsymbol{S_n}$

① add n BLs in parallel
③ add the last two addends C_o, PS

Adder Summary

Function	two n-bit integer addition		M integer addition
Algorithm	ripple carry adder	carry select adder	carry select adder
Layout	$O(n) \times O(1)$	$O(\sqrt{n}) \times O(\sqrt{n})$	$O(\sqrt{M}) \times O(\sqrt{M})$
Time complexity	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{M})$
Lower bound type	shape / algorithm bound	function / algorithm bound	function bound

OUTLINE

- Background
- Theoretical Analysis
- Integer Addition
- Extension
- Experimental Evaluation
- Conclusion

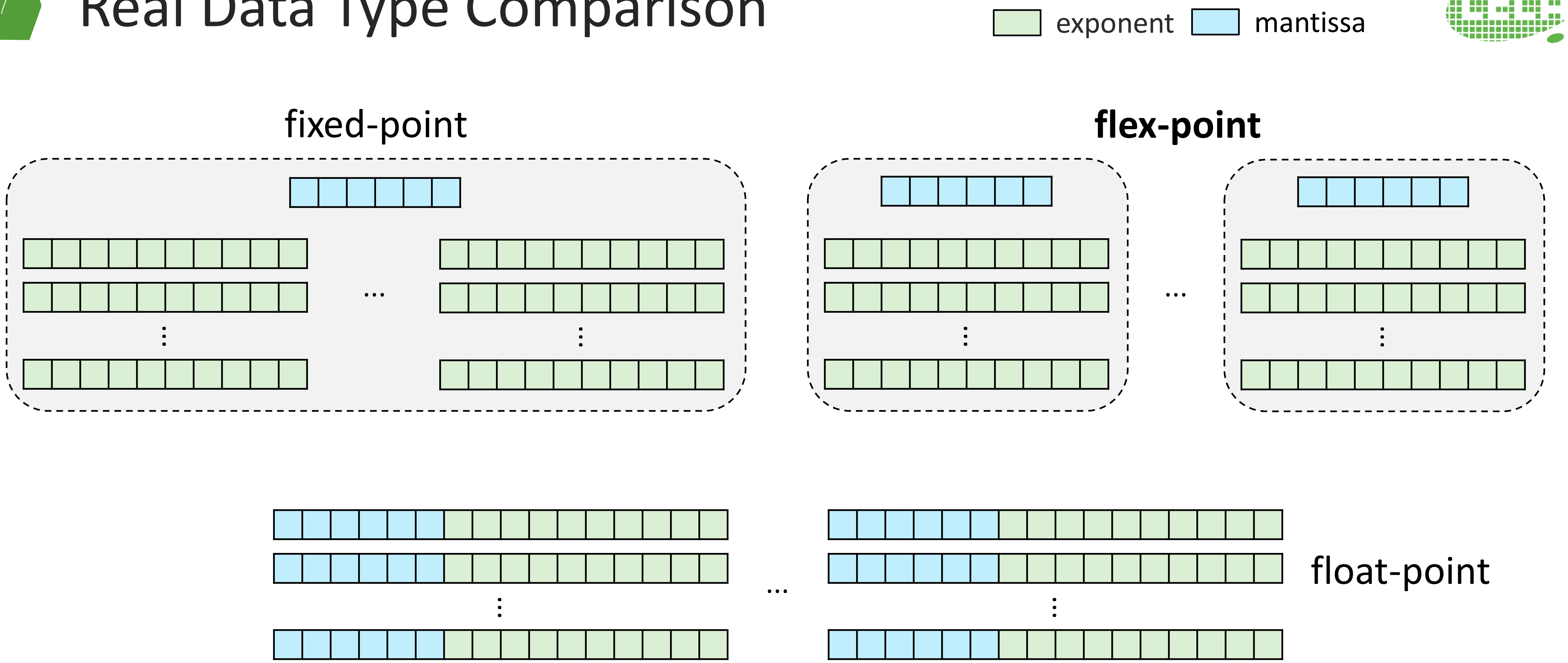
Dot Product

time complexity: $O(\sqrt{M})$
function lower bound

a_{11}	…	$a_{1\sqrt{M}}$	a_{21}	…	$a_{2\sqrt{M}}$	P_1	…	$P_{\sqrt{M}}$	S_1
$a_{1(\sqrt{M}+1)}$	…	$a_{1(2\sqrt{M})}$	$a_{2(\sqrt{M}+1)}$	…	$a_{2(2\sqrt{M})}$	$P_{\sqrt{M}+1}$	…	$P_{2\sqrt{M}}$	S_2
…	…	…	… ①	…	…	…	…	② …	…
$a_{1(M-\sqrt{M}+1)}$	…	a_{1M}	$a_{2(M-\sqrt{M}+1)}$	…	a_{2M}	$P_{M-\sqrt{M}+1}$	…	P_M	$S_{\sqrt{M}}$
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	③ C_o
…	…	…	…	…	…	…	…	…	PS
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	$A_1 \cdot A_2$

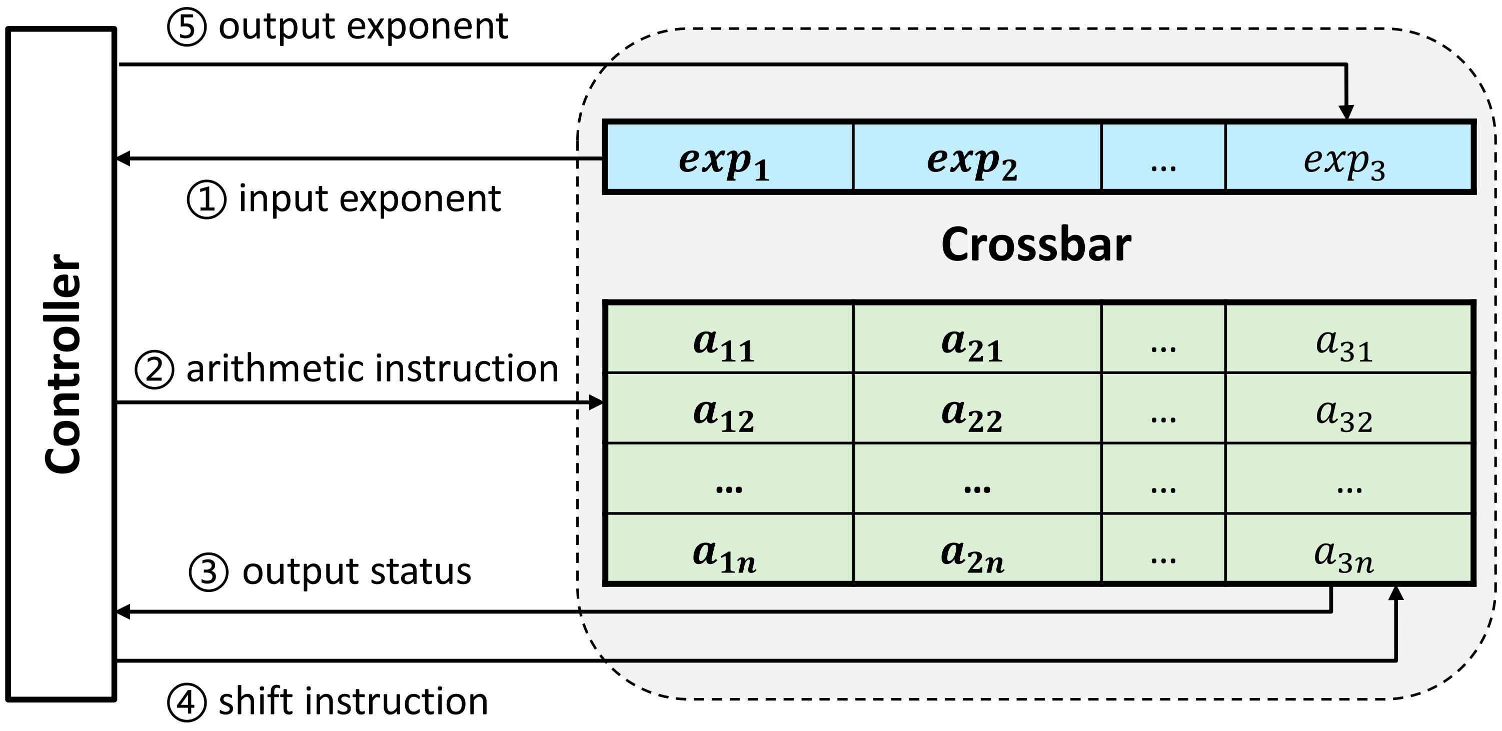
① element-wise multiplication
② accumulate $\sqrt{M}$ WLs in parallel
③ accumulate S_ks using the carry save adder

Real Data Type Comparison



[7] Köster, U., Webb, T. J., Wang, X., Nassar, M., Bansal, A. K., Constable, W. H., … Rao, N. (2017). *Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks. arXiv.*

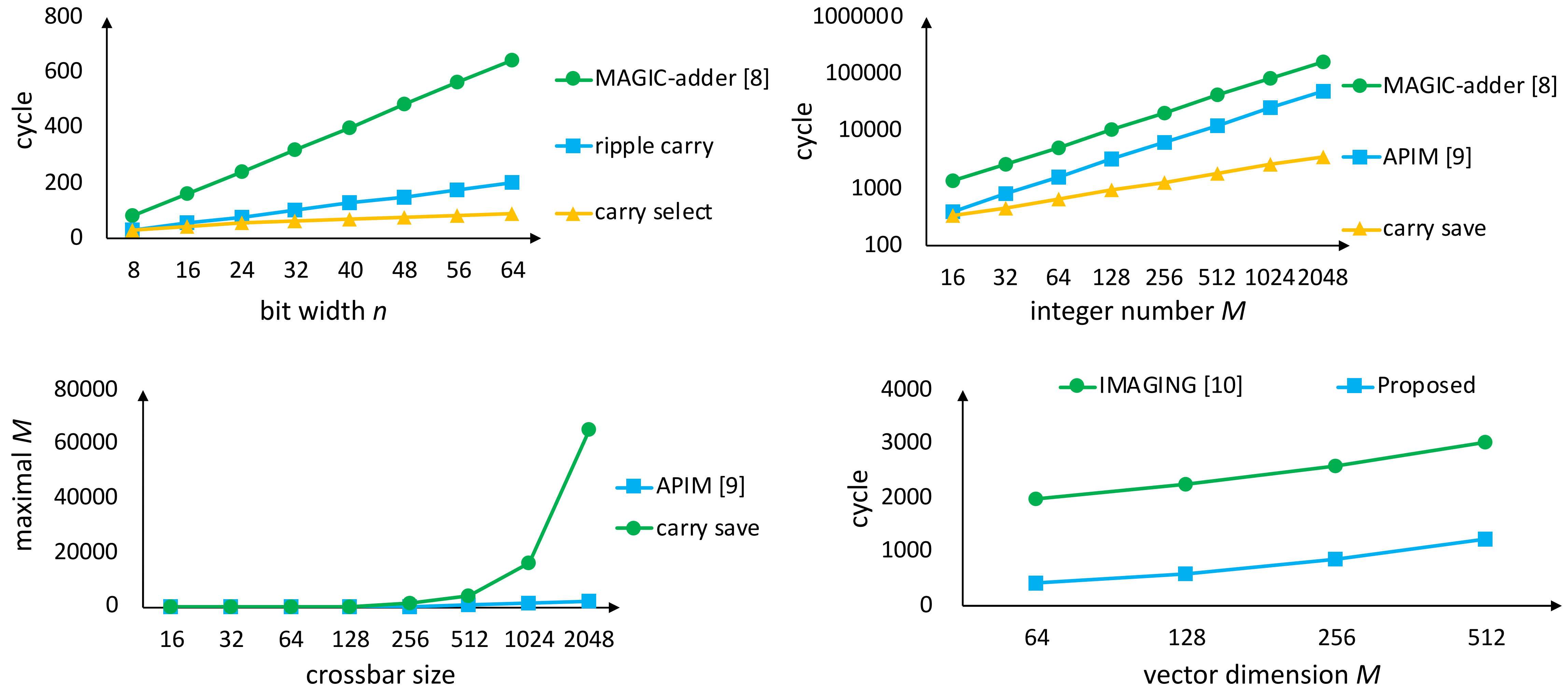
Flex-point Support



OUTLINE

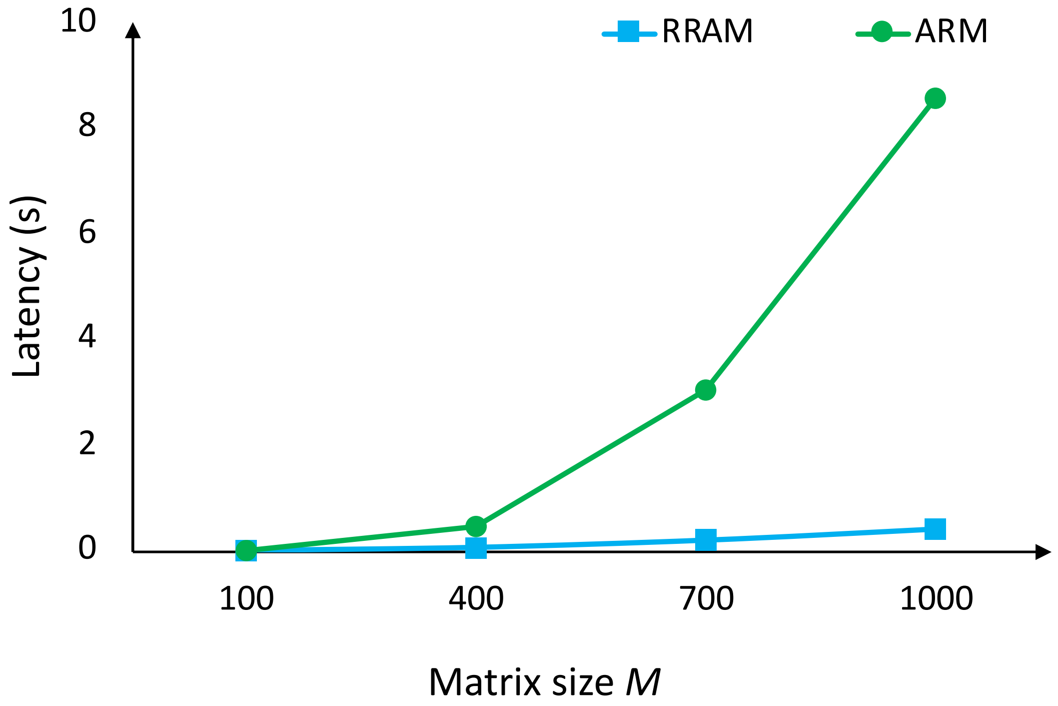
- Background
- Theoretical Analysis
- Integer Addition
- Extension
- **Experimental Evaluation**
- Conclusion

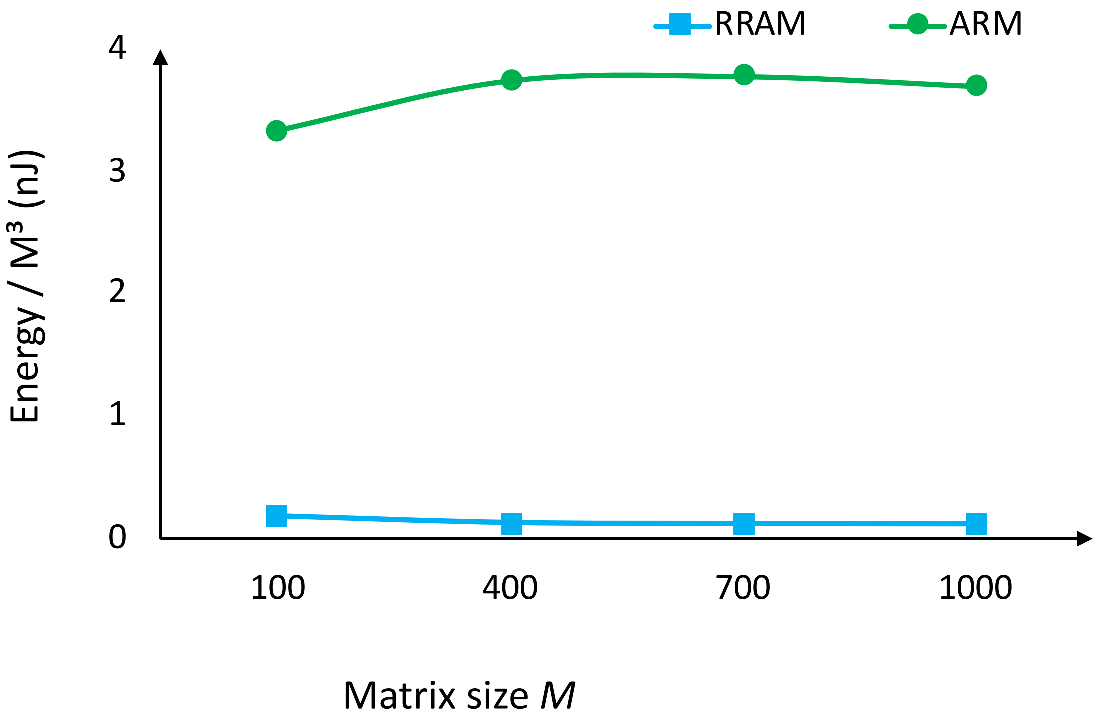
Integer Algorithm Evaluation



[8] Talati, N., Gupta, S., Mane, P., & Kvatinsky, S. (2016). Logic design within memristive memories using memristor-aided loGIC (MAGIC). *TNANO.*
[9] Imani, M., Gupta, S., & Rosing, T. (2017). Ultra-Efficient Processing In-Memory for Data Intensive Applications. In *DAC.*

Flex-support Evaluation





OUTLINE

- Background
- Theoretical Analysis
- Integer Addition
- Extension
- Experimental Evaluation
- Conclusion

Conclusion

- Theoretical analysis
  - SIML model
  - time complexity lower bound
- Integer addition
  - ripple carry adder
  - carry select adder
  - carry save adder
- Extension
  - multiplication support
  - flex-point support

北京大学高能效计算与应用中心
Center for Energy-efficient Computing and Applications

Q & A