abstractPIM: Bridging the Gap Between Processing-In-Memory Technology and Instruction Set Architecture

Adi Eliahu, Rotem Ben Hur, Ronny Ronen, and Prof. Shahar Kvatinsky

Faculty of Electrical Engineering

Technion – Israel Institute of Technology

VLSI SoC, October 2020

ASIC²
ARCHITECTURES
SYSTEMS
INTELLIGENT COMPUTING
INTEGRATED CIRCUITS

TECHNION
Israel Institute of Technology

Data-Intensive Applications

Neural Networks

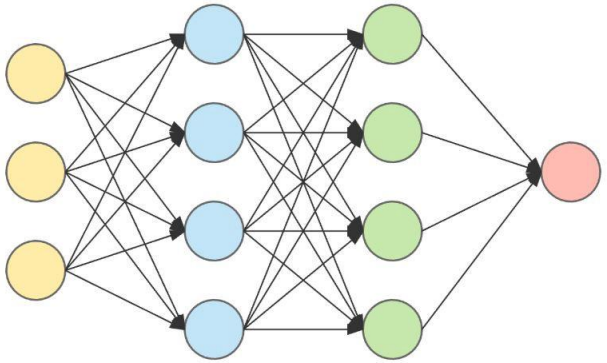
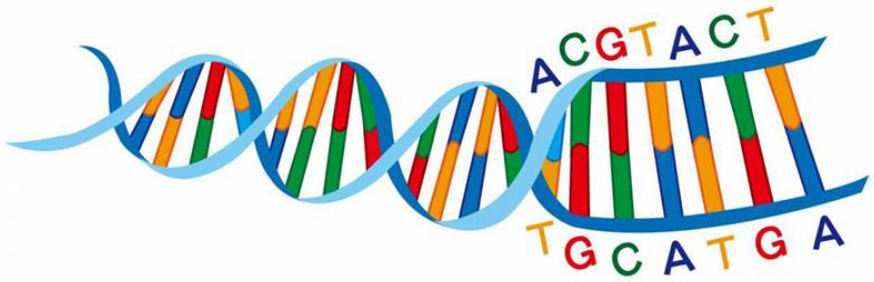


Image Processing

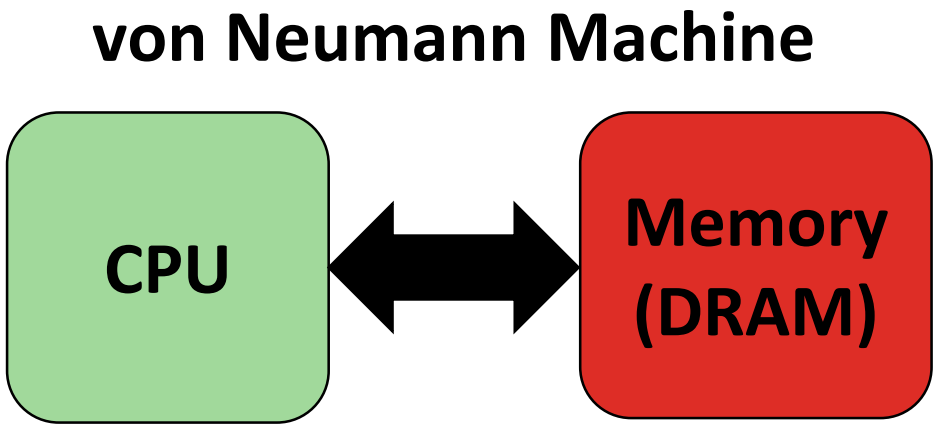
Encryption

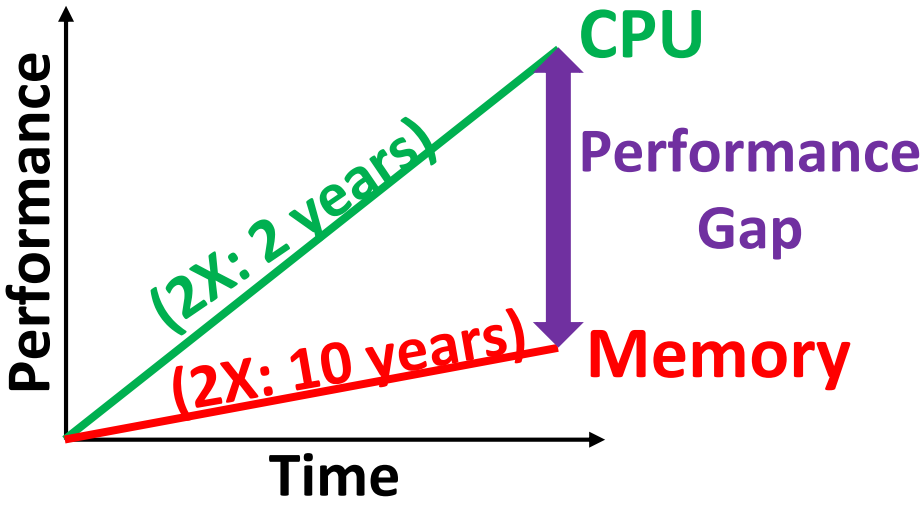
DNA Sequencing



ASIC²

Inefficiencies in the von Neumann Machines



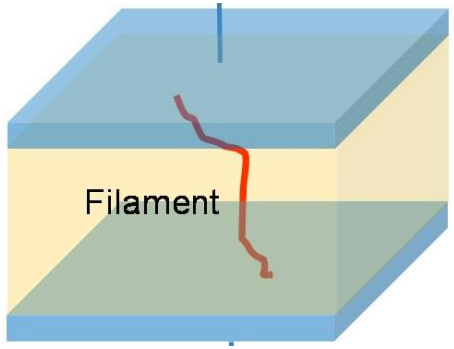


Memory Wall Problem

Memristors

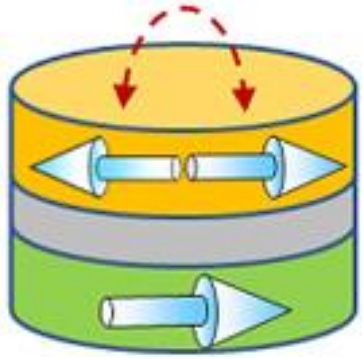
Emerging Nonvolatile Memory Technologies

Resistive RAM (RRAM)



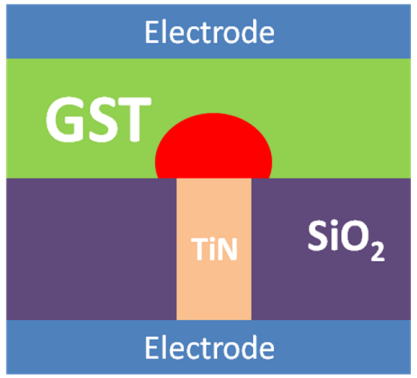
SanDisk SONY hp SAMSUNG Panasonic TOSHIBA SK hynix Crossbar

STT MRAM



EVERSPIN TECHNOLOGIES HITACHI CROCUS Technology Blossoming future TOSHIBA QUALCOMM SAMSUNG

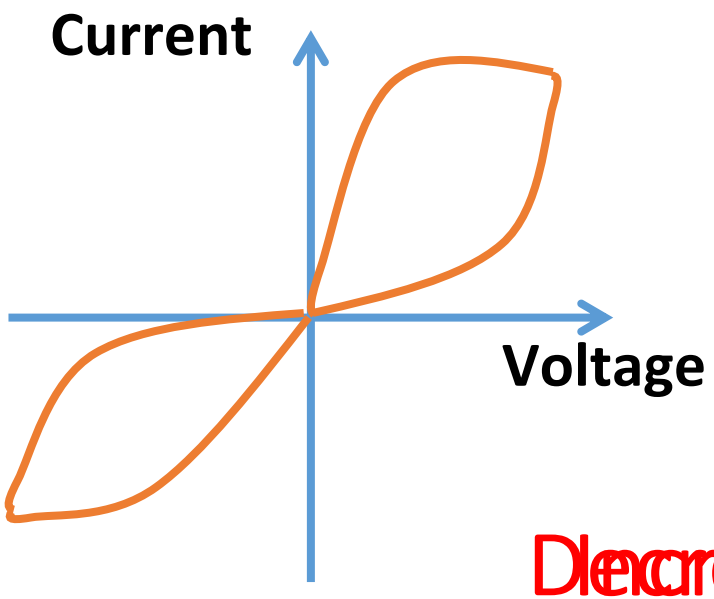
Phase Change Memory (PCM)



ST SAMSUNG IBM SK hynix intel Micron

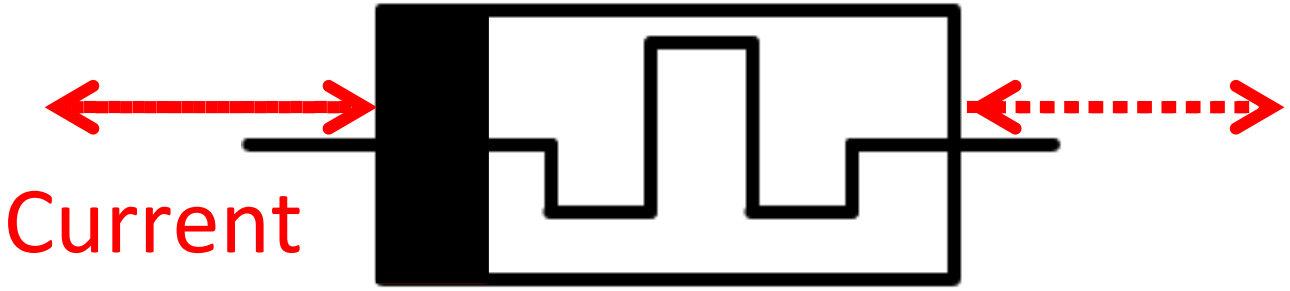
Memristor – Memory Resistor

Resistor with Varying Resistance



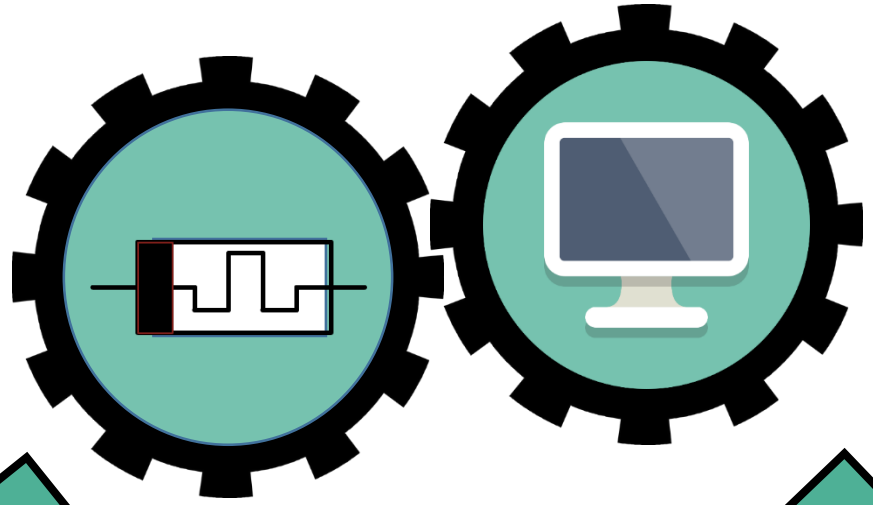
High resistive state
(R_{OFF}, HRS)

Decrease in resistance



Memristor

Memristor Integration in Computer Architectures



accelerators

analog computation

general-purpose computation

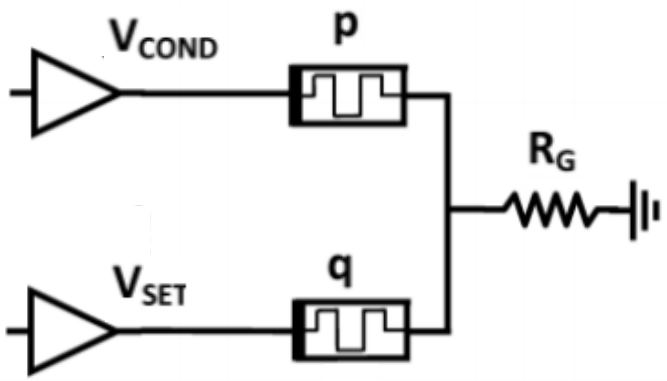
stateful logic

ASIC²

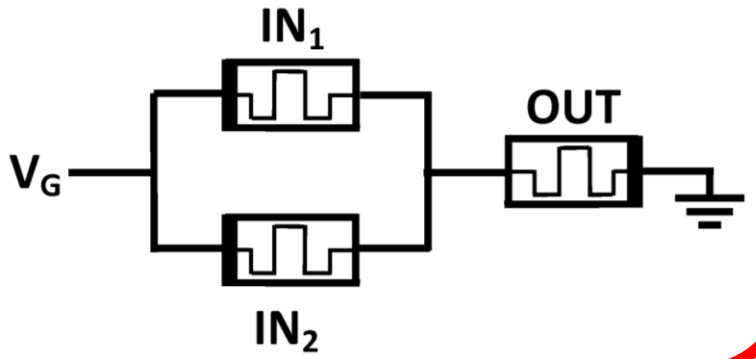
Stateful Logic [1]

The logic gate inputs and outputs are represented by memristor resistance

IMPLY [2]

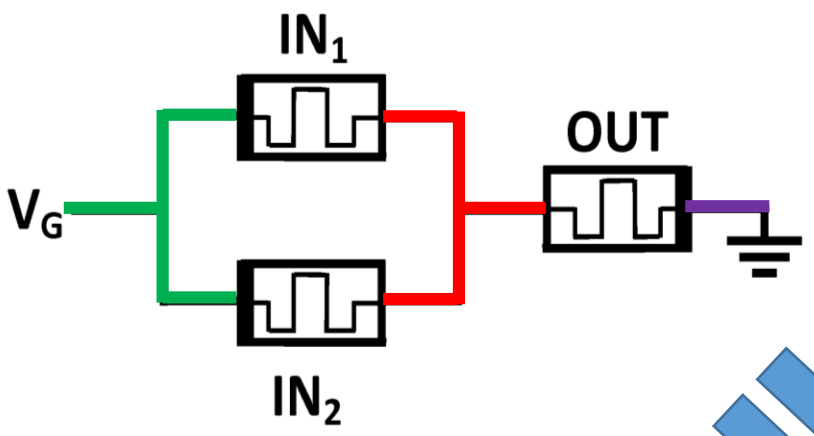


MAGIC [3]

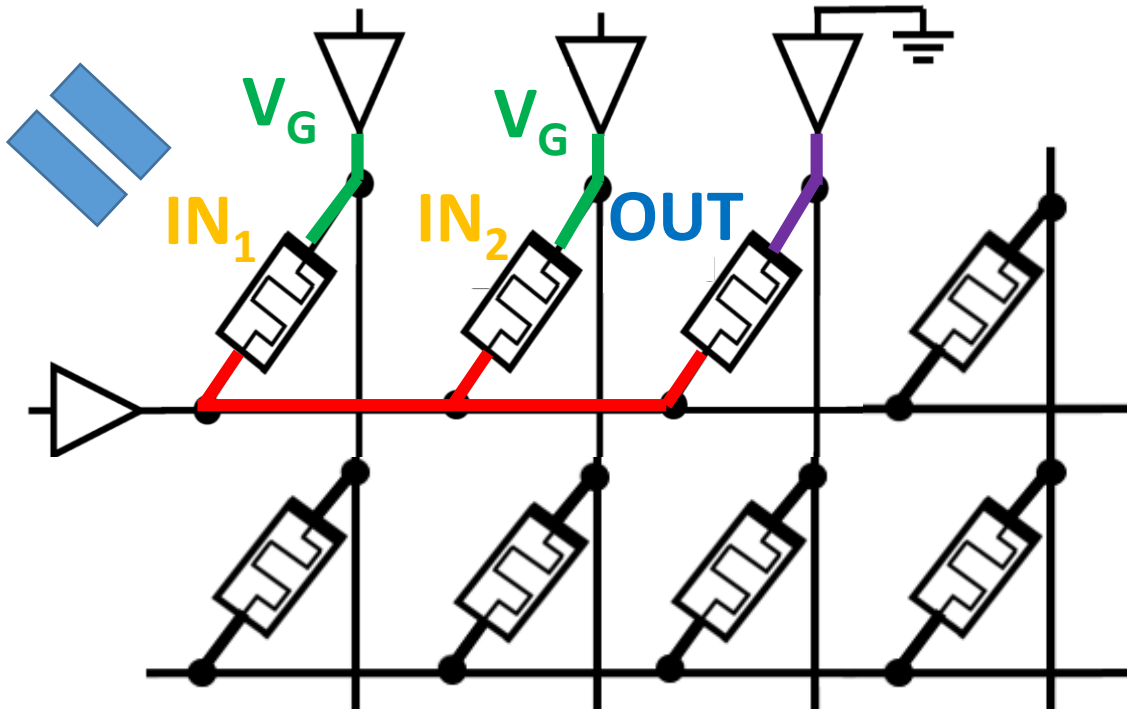


[1] J. Reuben *et al.*, "Memristive logic: A framework for evaluation and comparison," PATMOS, Sep. 2017
[2] S. Kvatinsky *et al.*, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies", TVLSI, Oct. 2014
[3] S. Kvatinsky *et al.*, "MAGIC – Memristor Aided LoGIC," TCAS II, Nov. 2014

MAGIC NOR in Memristive Crossbar



Functionally complete

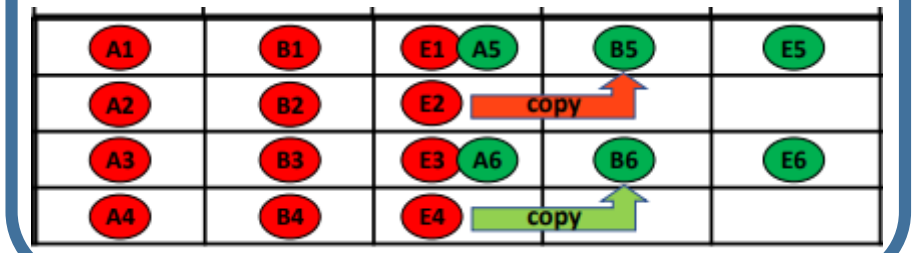


Crossbar Compatible

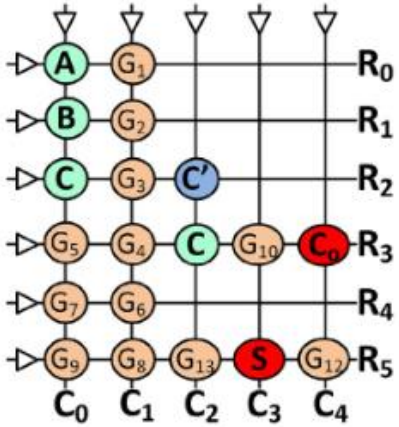
ASIC²

Logic Execution within a Memristive Crossbar

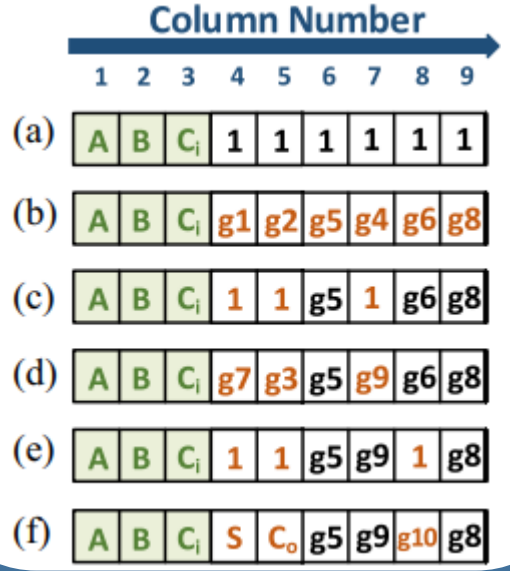
SIMPLE [1]



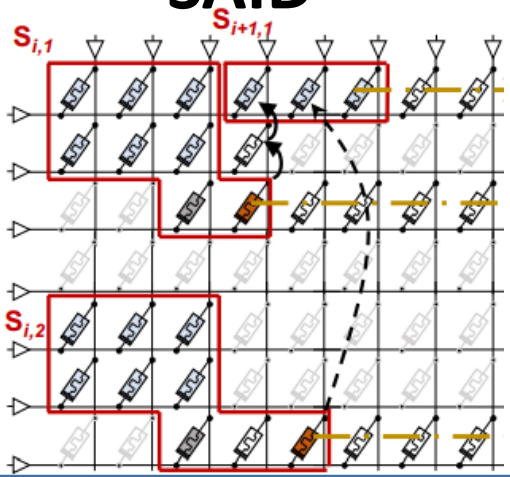
Yadav et al. [2]



SIMPLER [3]

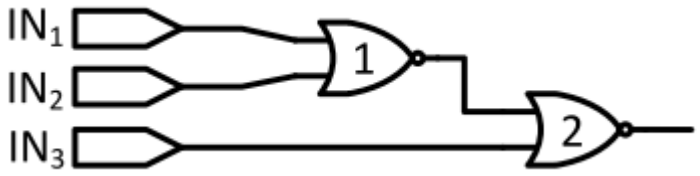


SAID [4]



All use two steps:

1. Logic synthesis

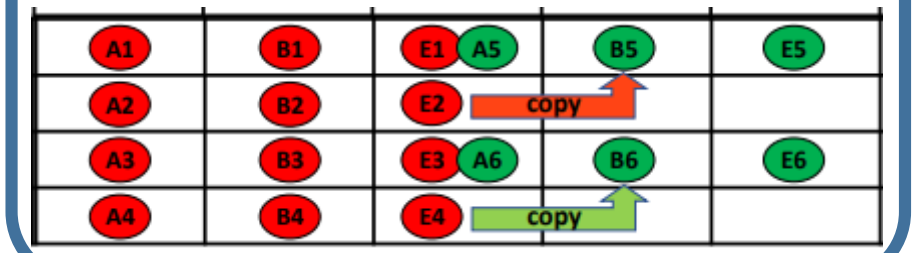


2. Each gate is assigned:
   - Clock cycle
   - Cell mapping

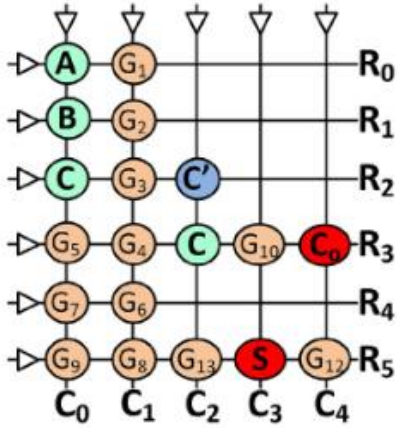
[1] R. Ben Hur *et al.,* "SIMPLE MAGIC: Synthesis and In-memory MaPping of Logic Execution for Memristor-Aided loGIC", IEEE ICCAD, Nov. 2017
[2] D. N. Yadav *et al.,* "Look-ahead mapping of Boolean functions in memristive crossbar array", Integration, Jan. 2019
[3] R. Ben Hur *et al.,* "SIMPLER MAGIC: Synthesis and mapping of in-memory logic executed in a single row to improve throughput," IEEE TCAD, Jul. 2019
[4] V. Tenace *et al.,* "SAID: A Supergate-Aided Logic Synthesis Flow for Memristive Crossbars", DATE, Mar. 2019

Logic Execution within a Memristive Crossbar

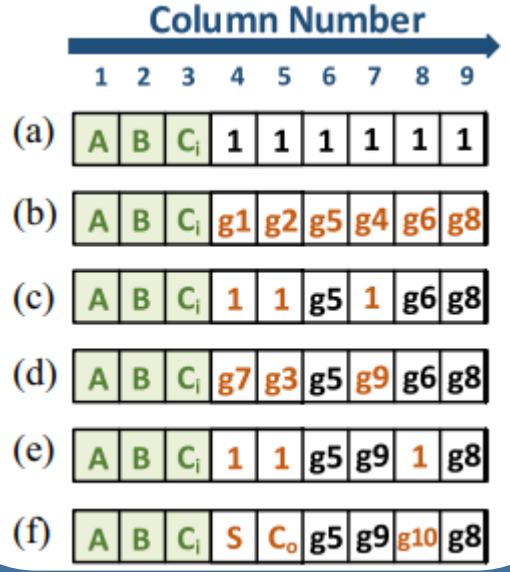
SIMPLE [1]



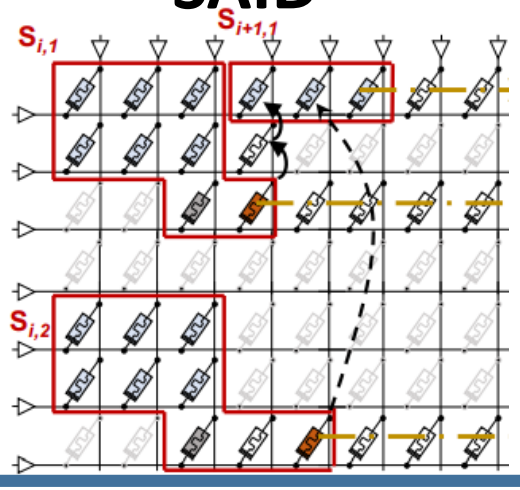
Yadav et al. [2]



SIMPLER [3]



SAID [4]



Challenges with current solutions:

1. Gap between target machine constraints and instruction set is not addressed
2. No backward compatibility with other machines

[1] R. Ben Hur *et al.,* "SIMPLE MAGIC: Synthesis and In-memory MaPping of Logic Execution for Memristor-Aided loGIC", IEEE ICCAD, Nov. 2017
[2] D. N. Yadav *et al.,* "Look-ahead mapping of Boolean functions in memristive crossbar array", Integration, Jan. 2019
[3] R. Ben Hur *et al.,* "SIMPLER MAGIC: Synthesis and mapping of in-memory logic executed in a single row to improve throughput," IEEE TCAD, Jul. 2019
[4] V. Tenace *et al.,* "SAID: A Supergate-Aided Logic Synthesis Flow for Memristive Crossbars", DATE, Mar. 2019

abstractPIM: Hierarchical Flow

Previous Work:

SIMPLE, SIMPLER, SAID, YADAV

This Work:

abstractPIM

Instruction Implementation

PIM Technology

MAGIC NOR

Hierarchical: separation of compilation and in-memory computation

PIM technology independent

Instruction Set

Instruction Implementation

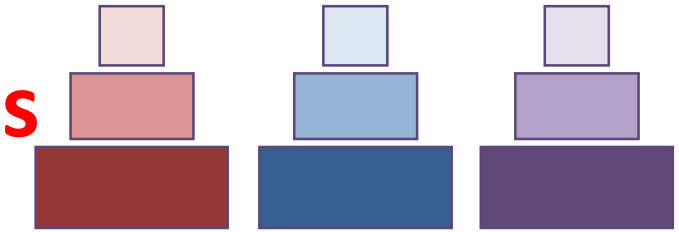
PIM **Technologies**

MAGIC NOR

MAGIC NAND

...

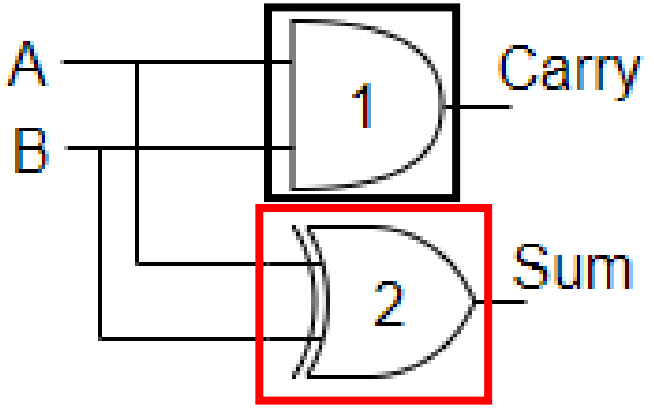
Comparison between different setups

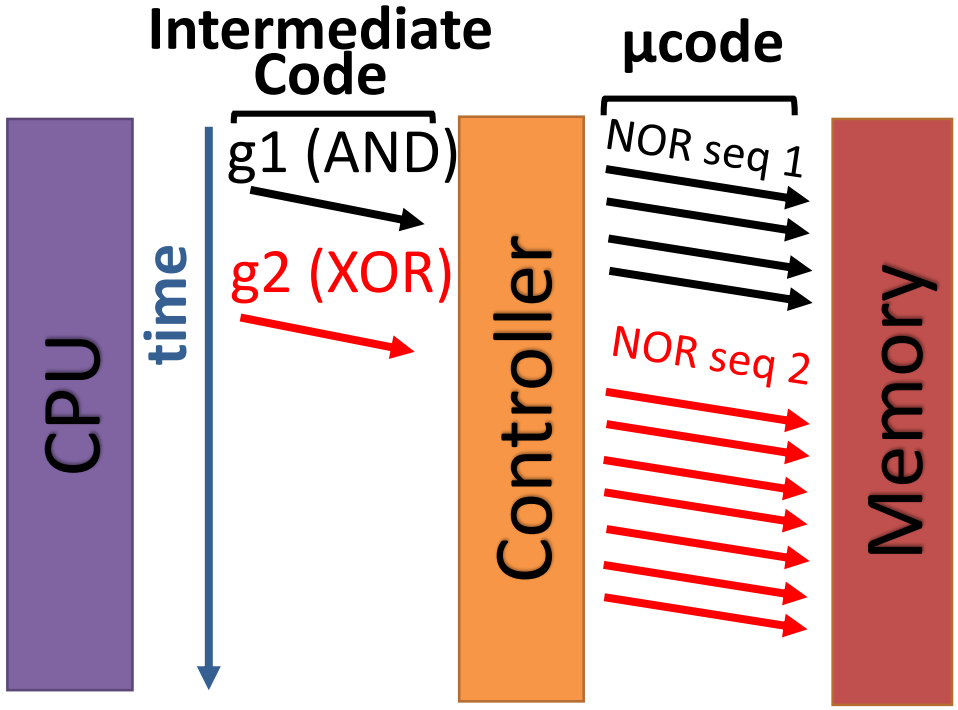


abstractPIM: Hierarchical Flow

Advantages:

1. **Hierarchical**

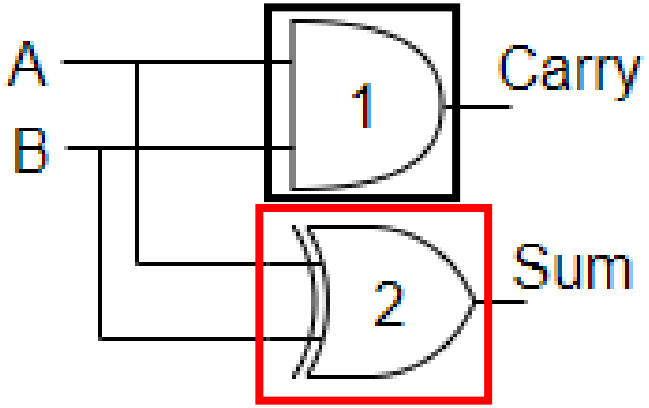


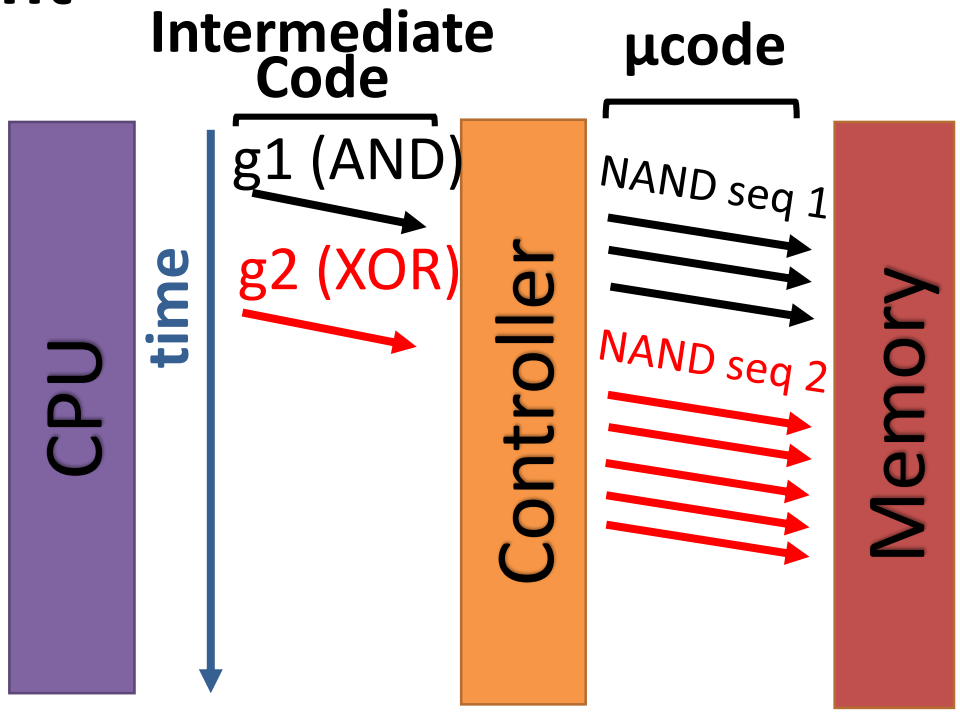


abstractPIM: Hierarchical Flow

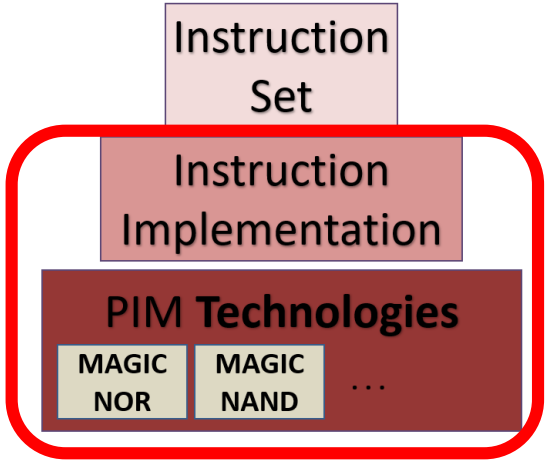
Advantages:

1. **Hierarchical**
2. **PIM technology independent**

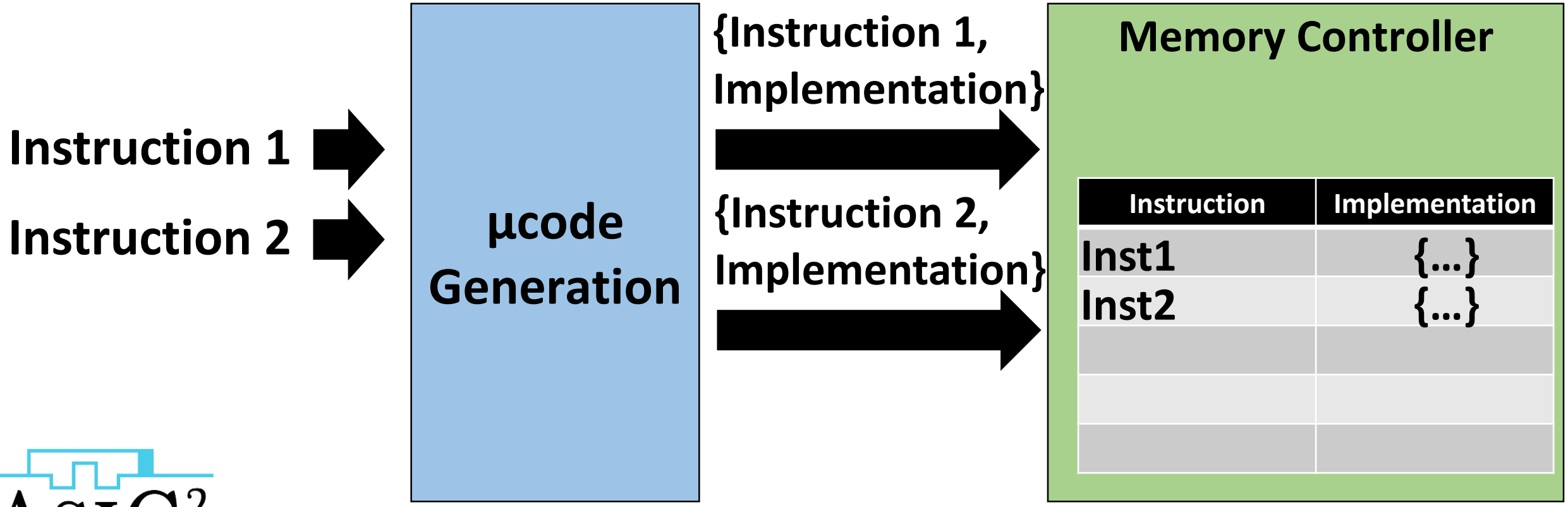




abstractPIM Flow: Microcode Generation

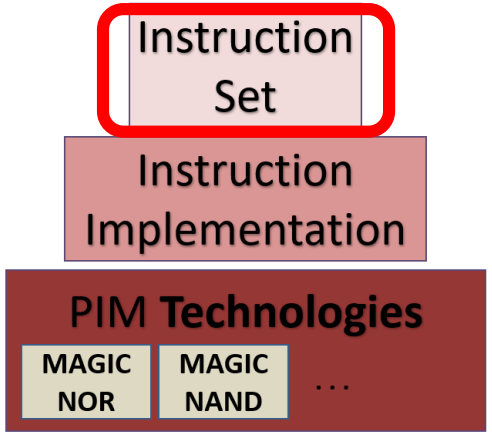


The controller is designed one time, offline

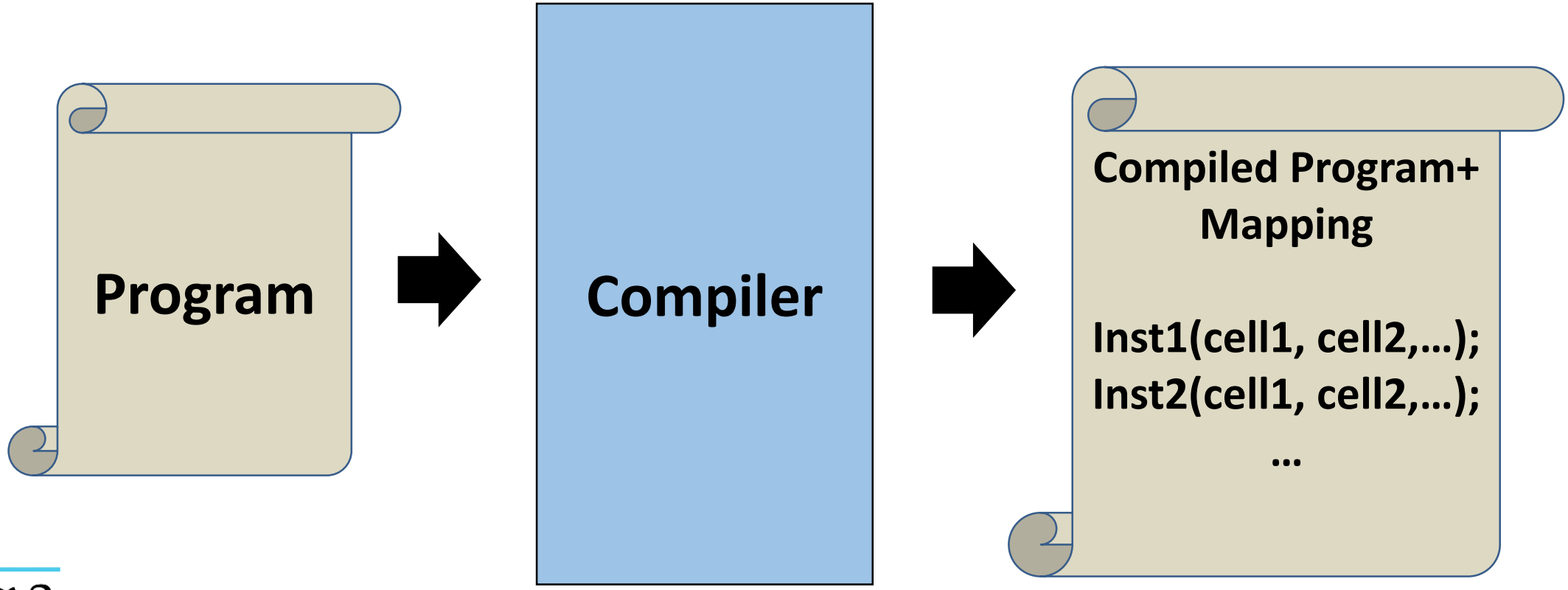


ASIC²

abstractPIM Flow: Intermediate Representation



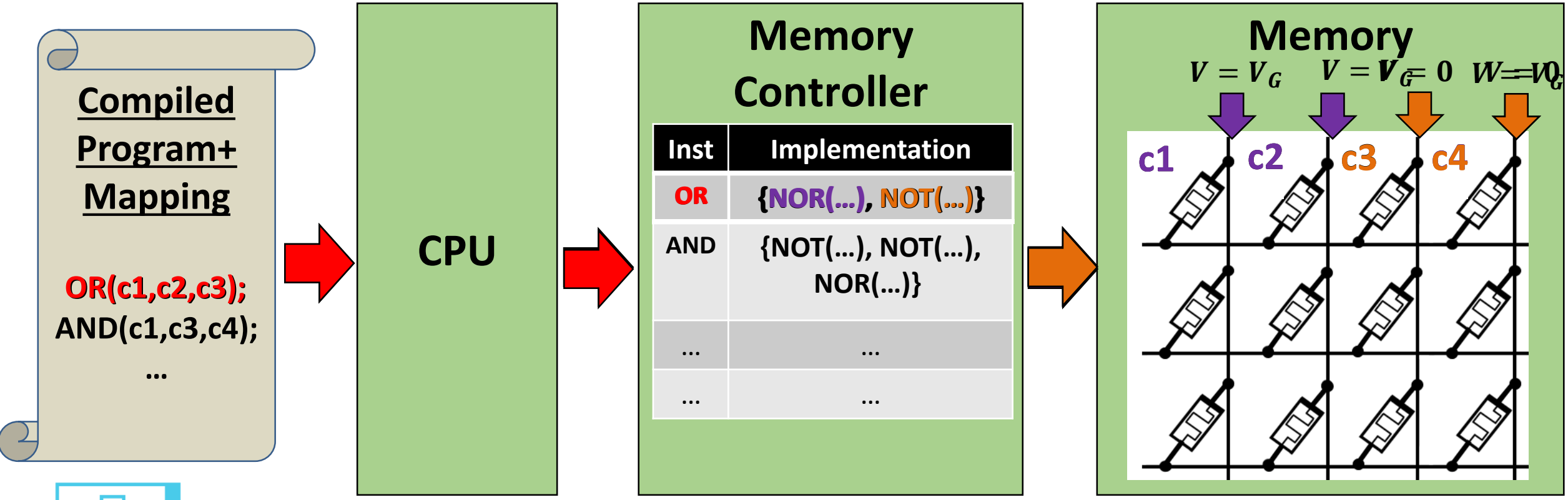
The program is compiled



ASIC²

abstractPIM Flow: Runtime Execution

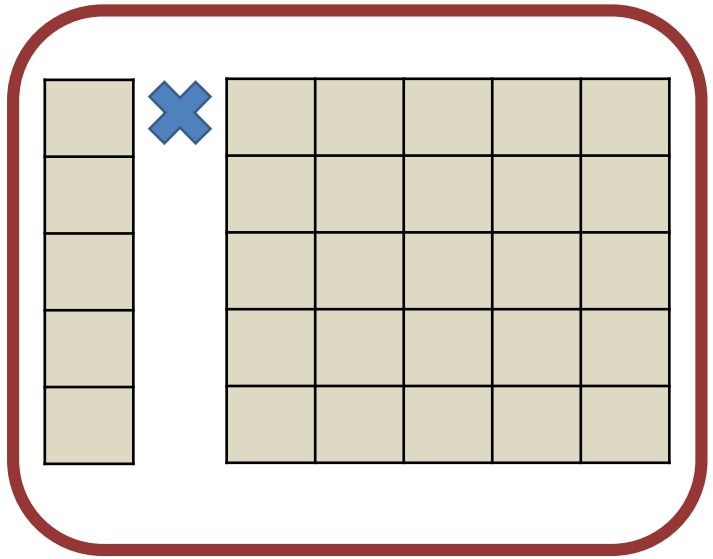
The program runs on the CPU



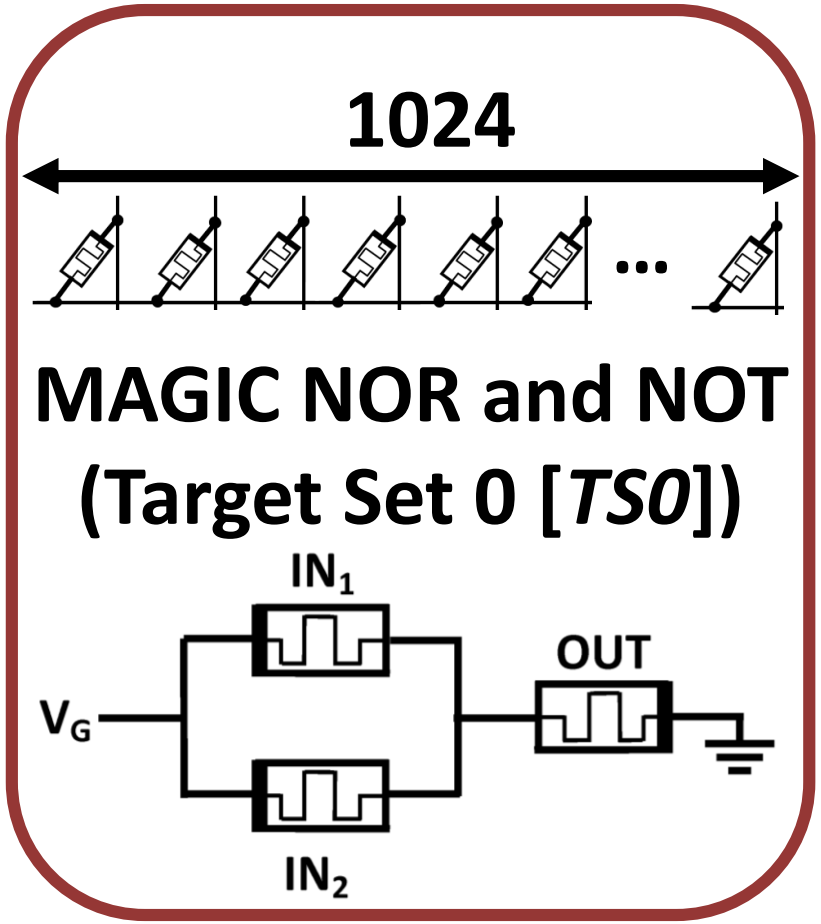
ASIC²

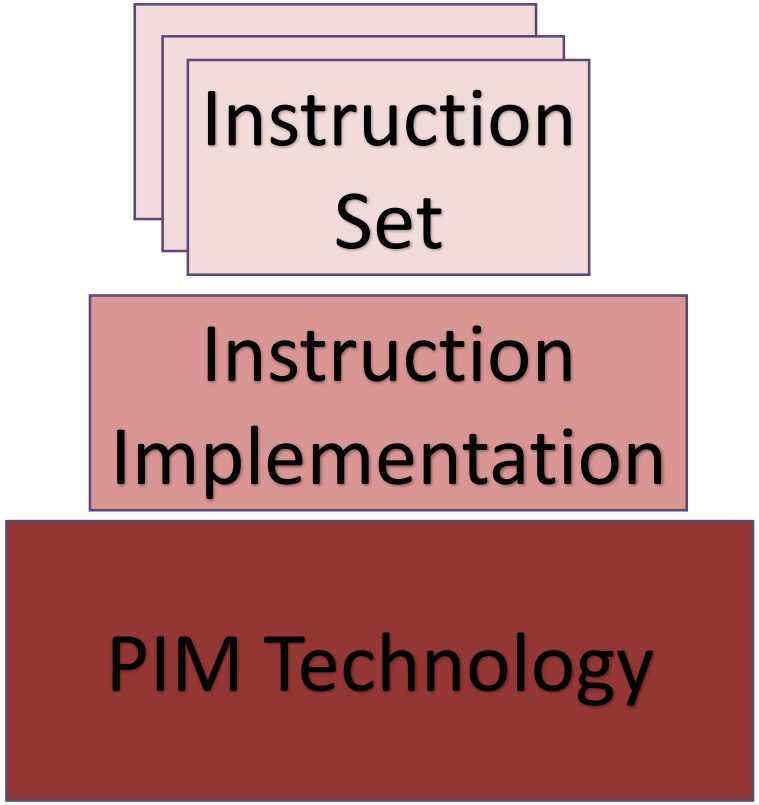
Case Study: Vector-Matrix Multiplication

Benchmark



Target Machine





ASIC²

Different Instruction Set Architectures

- Inclusive sets:
  - $TS0 \subset TS1 \subset IS2 \subset IS3$
- TS0: basic set (NOT, NOR)

Instruction	I	O	TS0	TS1	IS2	IS3
NOT	1	1	✓	✓	✓	✓
NOR2	2	1	✓	✓	✓	✓
NOR3	3	1	-	-	-	✓
NOR4	4	1	-	-	-	✓
OR2	2	1	-	✓	✓	✓
OR3	3	1	-	-	-	✓
OR4	4	1	-	-	-	✓
AND2	2	1	-	✓	✓	✓
AND3	3	1	-	-	-	✓
AND4	4	1	-	-	-	✓
NAND2	2	1	-	-	✓	✓
NAND3	3	1	-	-	-	✓
NAND4	4	1	-	-	-	✓
XOR2	2	1	-	-	✓	✓
XOR3	3	1	-	-	-	✓
XOR4	4	1	-	-	-	✓
XNOR2	2	1	-	-	✓	✓
XNOR3	3	1	-	-	-	✓
XNOR4	4	1	-	-	-	✓
IMPLIES	2	1	-	-	✓	✓
!IMPLIES	2	1	-	-	✓	✓
MUX	3	1	-	-	✓	✓
HA	2	2	-	-	✓	✓
HS	2	2	-	-	✓	✓

ASIC²

Different Instruction Set Architectures

- Inclusive sets:
  - $TS0 \subset TS1 \subset IS2 \subset IS3$
- **TS0**: basic set (NOT, NOR)
- **IS2**: all functions with 1 or 2 inputs, common combinational functions

Instruction	I	O	TS0	TS1	IS2	IS3
NOT	1	1	✓	✓	✓	✓
NOR2	2	1	✓	✓	✓	✓
NOR3	3	1	-	-	-	✓
NOR4	4	1	-	-	-	✓
OR2	2	1	-	✓	✓	✓
OR3	3	1	-	-	-	✓
OR4	4	1	-	-	-	✓
AND2	2	1	-	✓	✓	✓
AND3	3	1	-	-	-	✓
AND4	4	1	-	-	-	✓
NAND2	2	1	-	-	✓	✓
NAND3	3	1	-	-	-	✓
NAND4	4	1	-	-	-	✓
XOR2	2	1	-	-	✓	✓
XOR3	3	1	-	-	-	✓
XOR4	4	1	-	-	-	✓
XNOR2	2	1	-	-	✓	✓
XNOR3	3	1	-	-	-	✓
XNOR4	4	1	-	-	-	✓
IMPLIES	2	1	-	-	✓	✓
!IMPLIES	2	1	-	-	✓	✓
MUX	3	1	-	-	✓	✓
HA	2	2	-	-	✓	✓
HS	2	2	-	-	✓	✓

ASIC²

Different Instruction Set Architectures

- Inclusive sets:
  - $TS0 \subset TS1 \subset IS2 \subset IS3$
- **TS0**: basic set (NOT, NOR)
- **IS2**: all functions with 1 or 2 inputs, common combinational functions
- **IS3**: IS2 extended to 3,4 inputs

Instruction	I	O	TS0	TS1	IS2	IS3
NOT	1	1	✓	✓	✓	✓
NOR2	2	1	✓	✓	✓	✓
NOR3	3	1	-	-	-	✓
NOR4	4	1	-	-	-	✓
OR2	2	1	-	✓	✓	✓
OR3	3	1	-	-	-	✓
OR4	4	1	-	-	-	✓
AND2	2	1	-	✓	✓	✓
AND3	3	1	-	-	-	✓
AND4	4	1	-	-	-	✓
NAND2	2	1	-	-	✓	✓
NAND3	3	1	-	-	-	✓
NAND4	4	1	-	-	-	✓
XOR2	2	1	-	-	✓	✓
XOR3	3	1	-	-	-	✓
XOR4	4	1	-	-	-	✓
XNOR2	2	1	-	-	✓	✓
XNOR3	3	1	-	-	-	✓
XNOR4	4	1	-	-	-	✓
IMPLIES	2	1	-	-	✓	✓
!IMPLIES	2	1	-	-	✓	✓
MUX	3	1	-	-	✓	✓
HA	2	2	-	-	✓	✓
HS	2	2	-	-	✓	✓

ASIC²

Vector-Matrix Multiplication Results

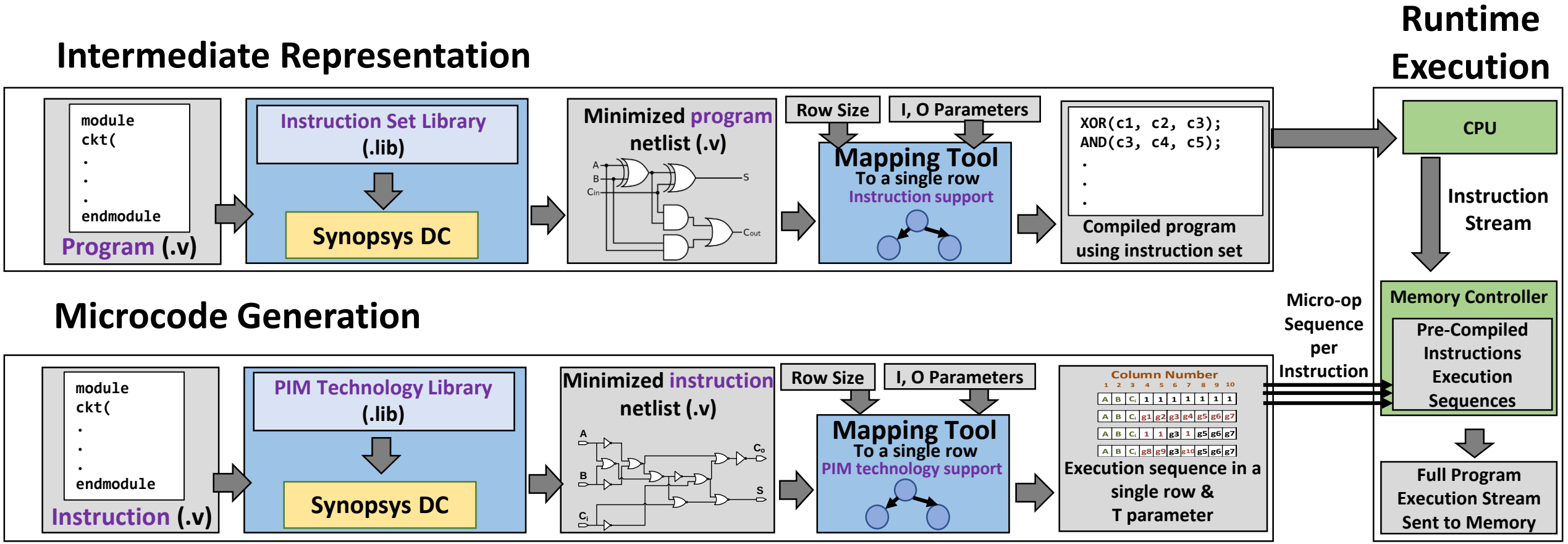
Instruction Set	T [Cycles]	Code Size
TS0	25470	12735
IS2	29515	6140
IS3	27436	5475

+16%
+8%

-52%
-57%

ASIC²

abstractPIM Evaluation Methodology



Rotem Ben Hur *et al.,* "SIMPLER MAGIC: Synthesis and mapping of in-memory logic executed in a single row to improve throughput," IEEE TCAD, Jul. 2019

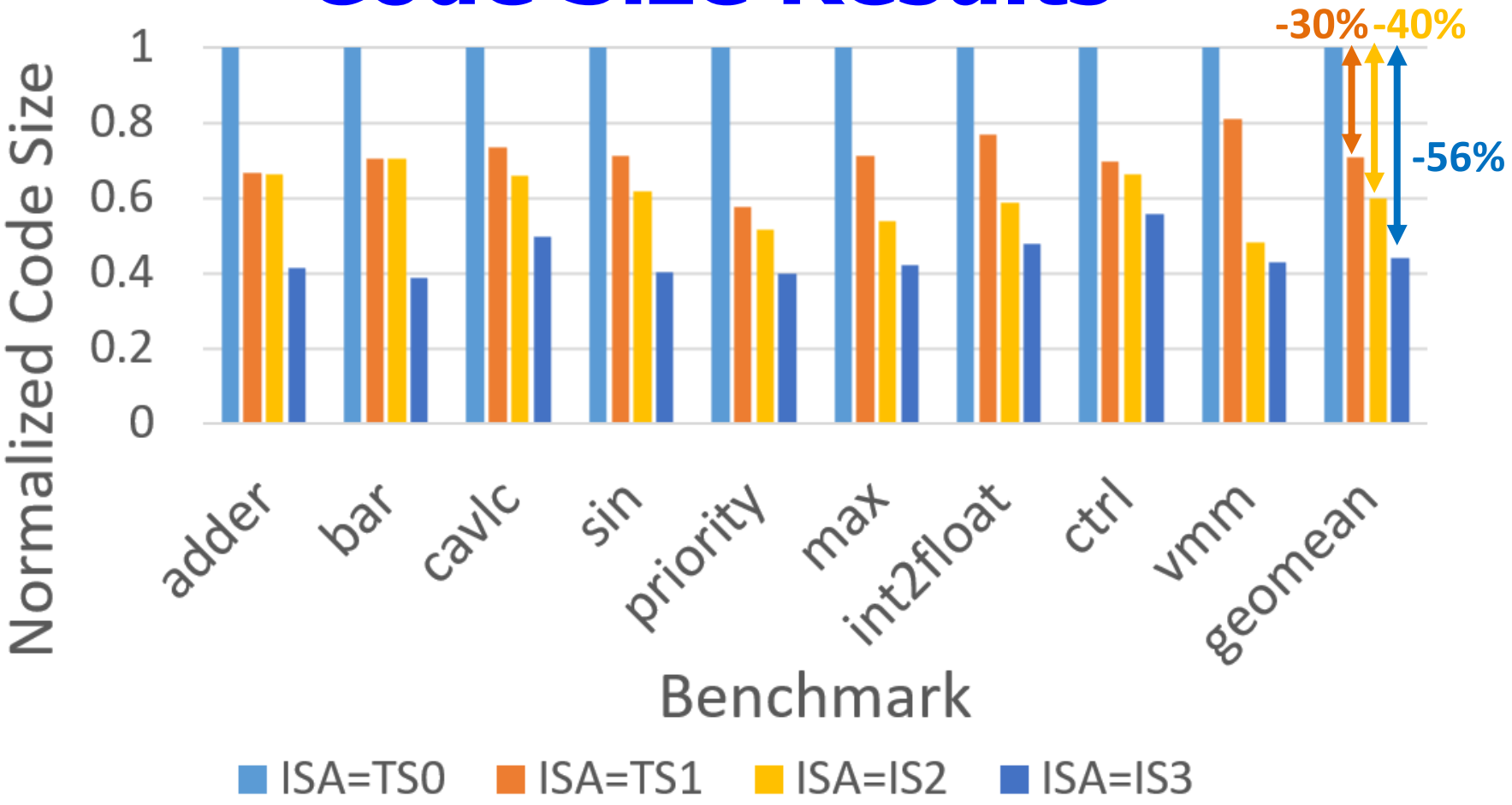
abstractPIM Evaluation Methodology

- Measured metrics:
  - Code size
  - Execution time
- EPFL Benchmarks [1]
- 512-sized or 1024-sized row
- Tested with different *instruction sets* and *technology sets*

ASIC²

[1] L. Amaru *et al.*, "The EPFL combinational benchmark suite," IWLS, 2015.

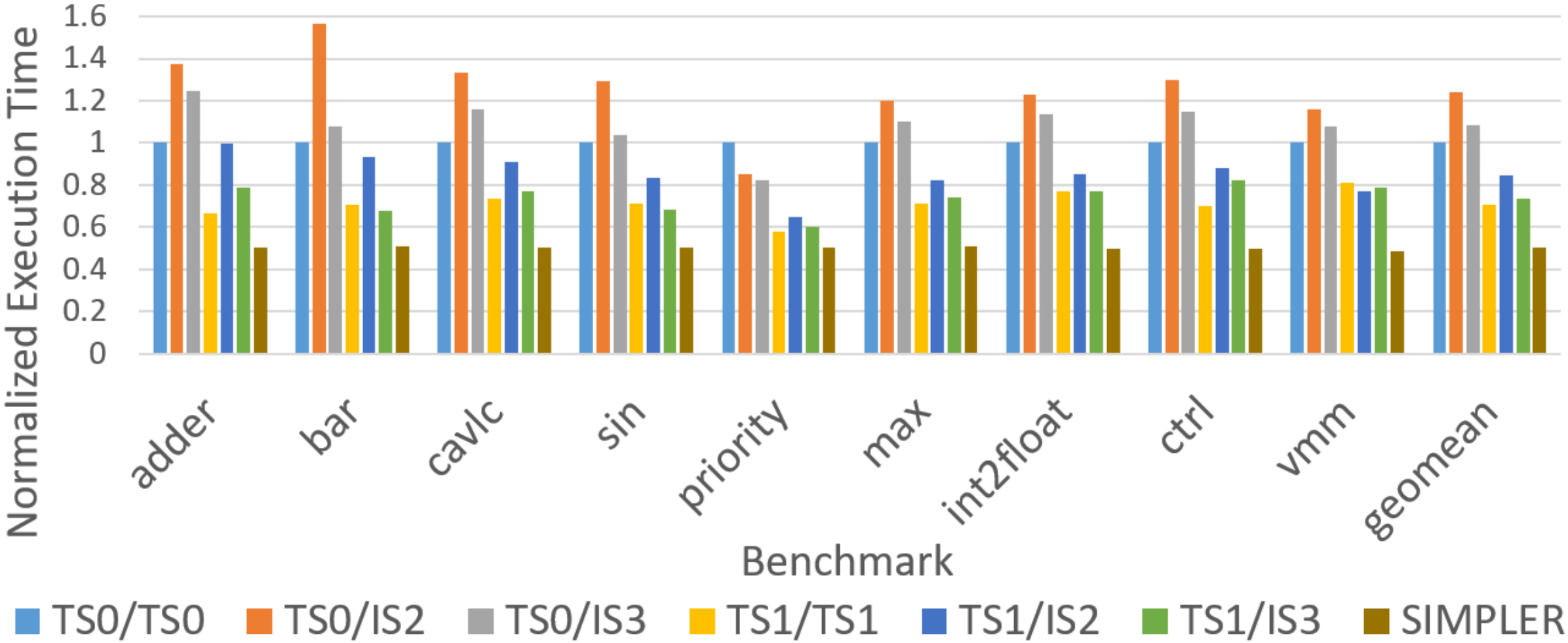
Code Size Results



$$CS_{TS0} > CS_{TS1} > CS_{IS2} > CS_{IS3}$$

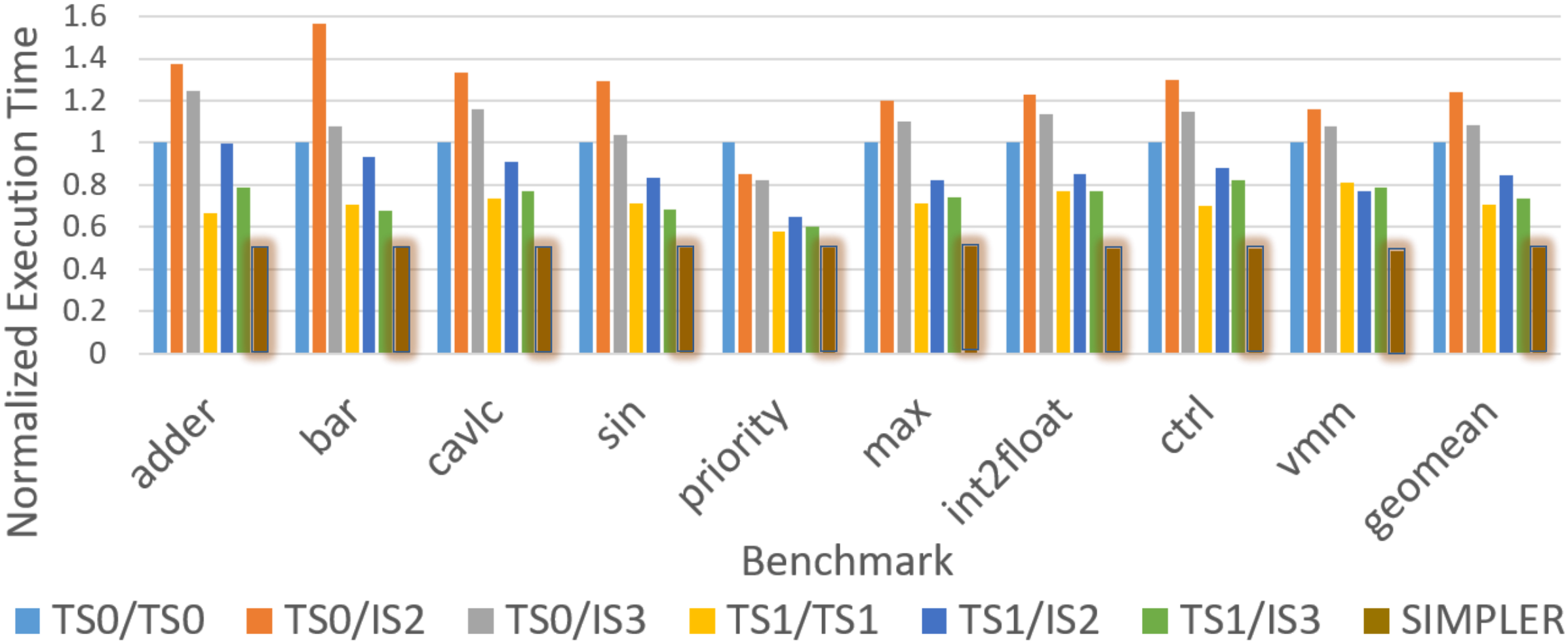
ASIC²

Execution Time Results



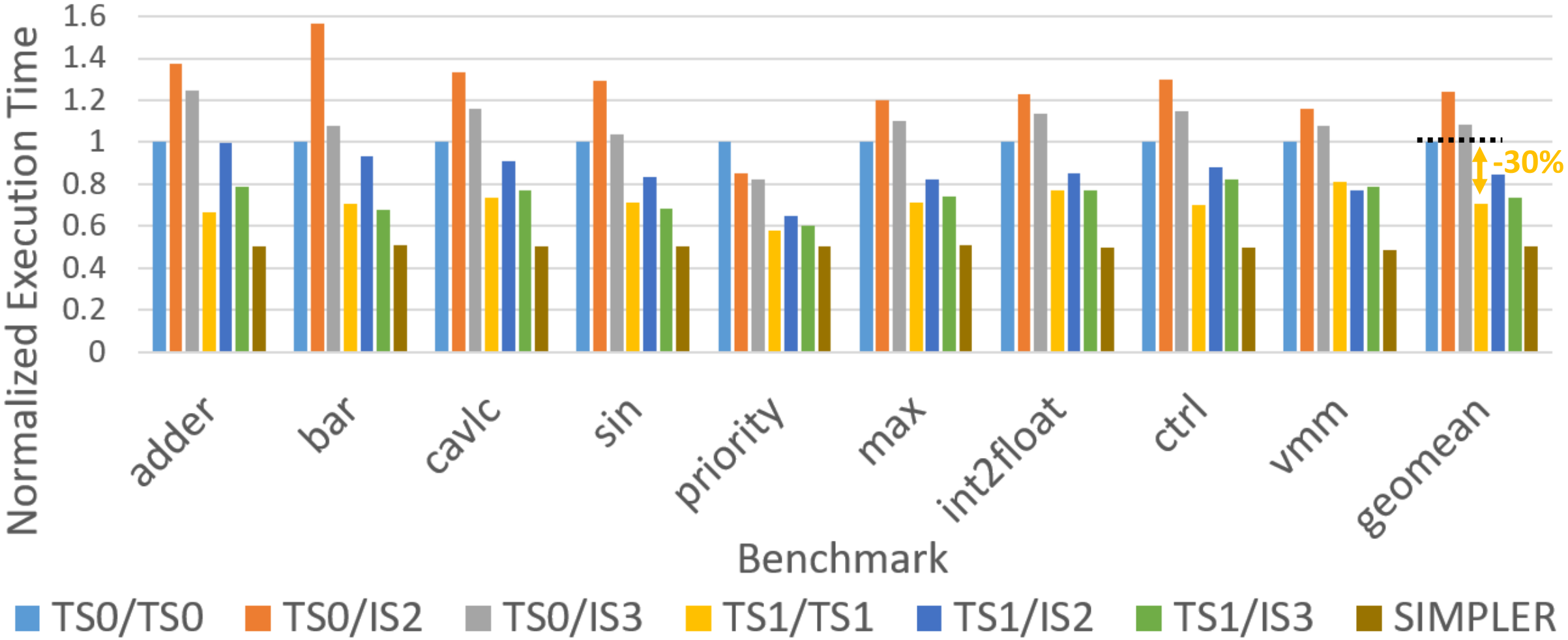
ASIC²

Execution Time Results



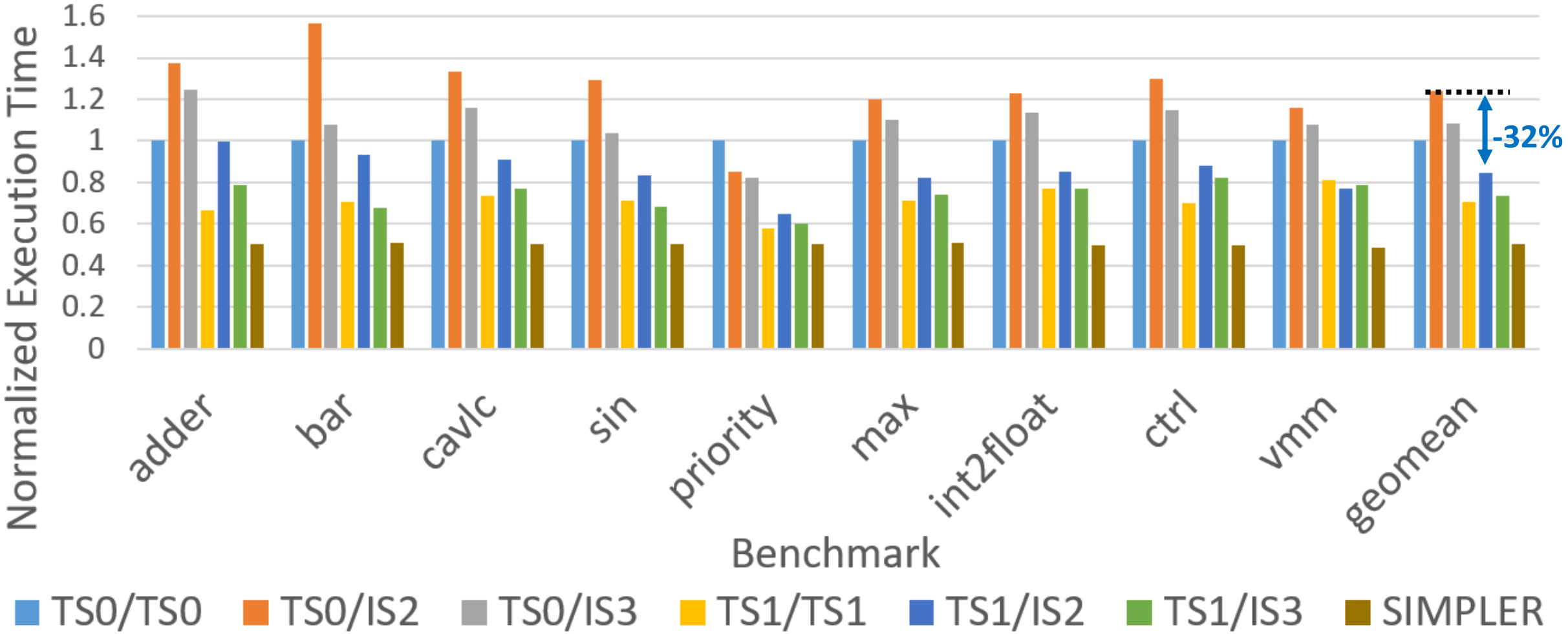
ASIC²

Execution Time Results



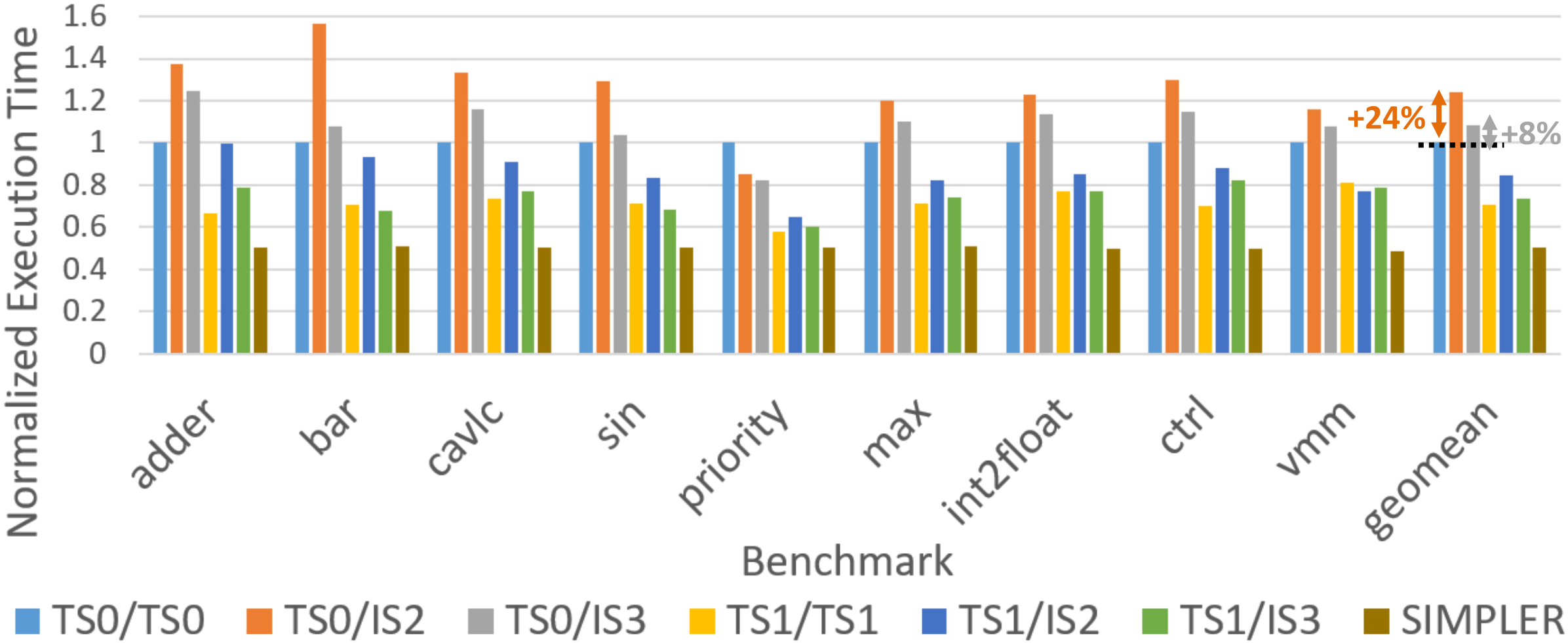
ASIC²

Execution Time Results



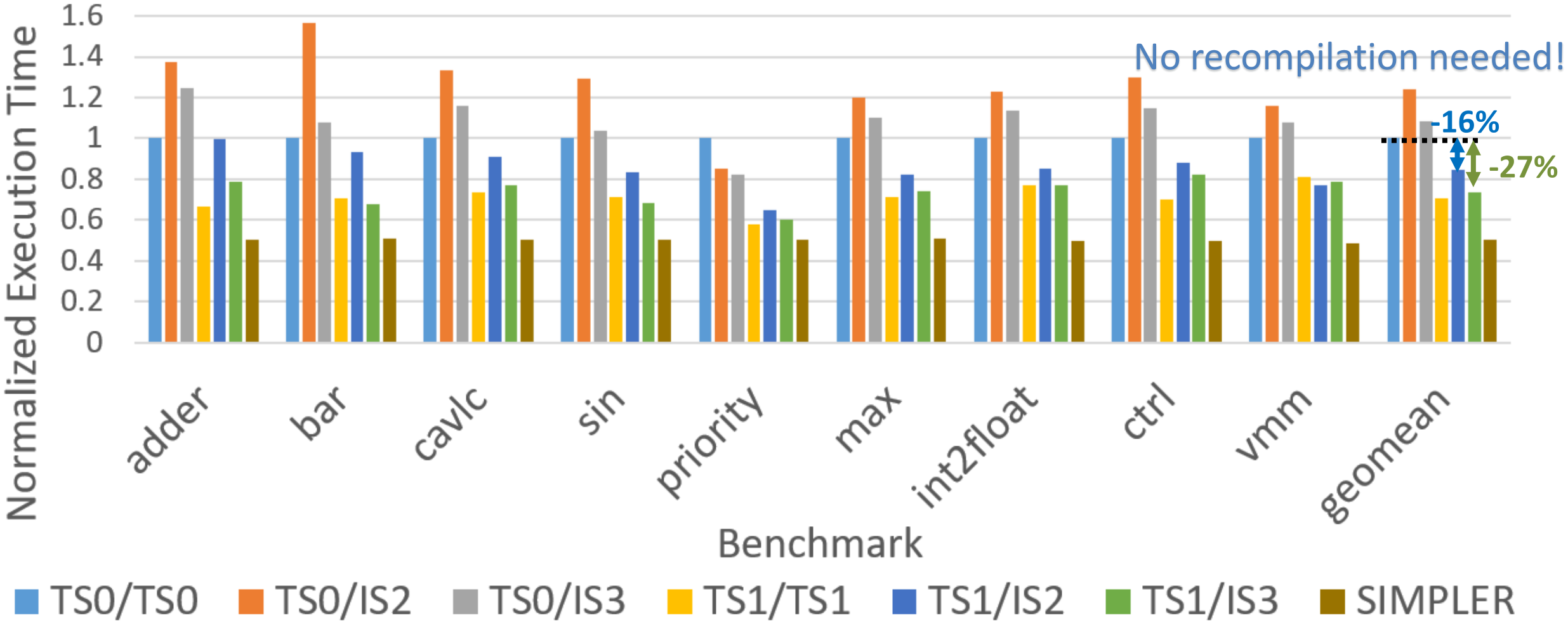
ASIC²

Execution Time Results



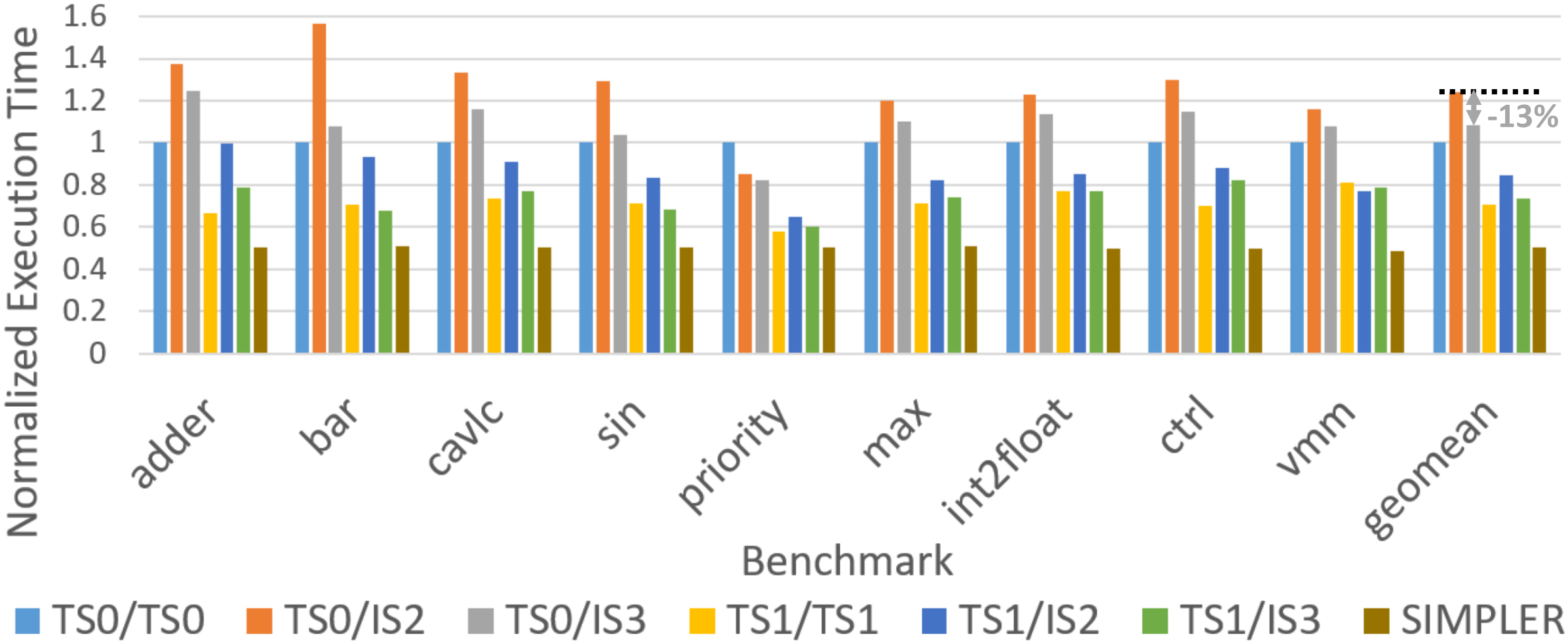
ASIC²

Execution Time Results



ASIC²

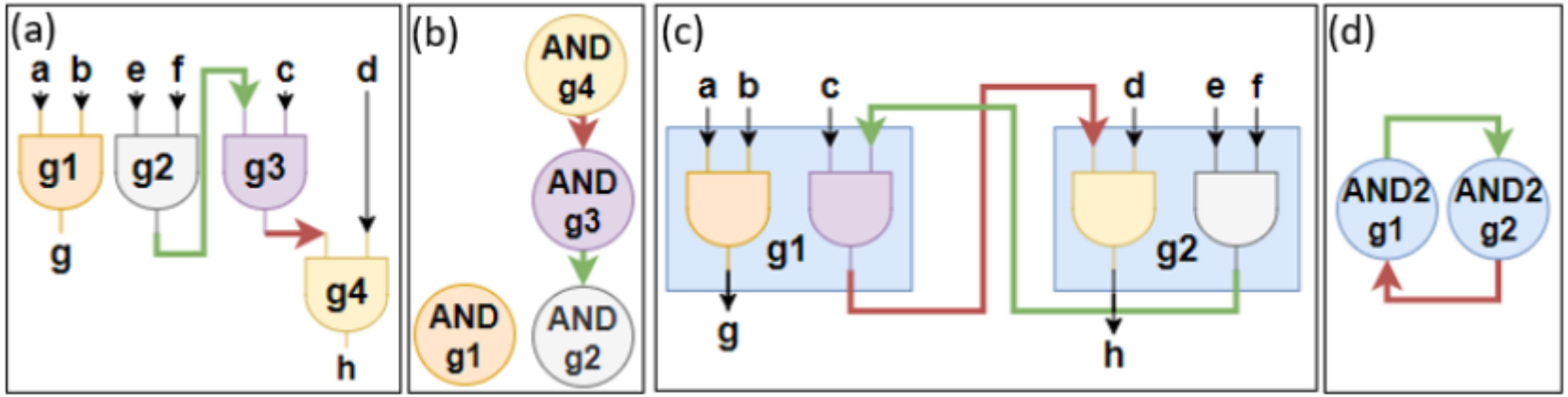
Execution Time Results



ASIC²

Future Work

- Solving the cyclic dependency problem



- Build more complex instruction sets and compare them

ASIC²

Conclusion

- abstractPIM: a hierarchical compilation concept and method for logic execution which provides:
  - Flexibility
  - Abstraction
  - Code size reduction (56%)
  - Backward compatibility with other machines
- Lays a solid foundation for a compiler for a memristor-based architecture

ASIC²

Thanks!

ASIC² ARCHITECTURES SYSTEMS INTELLIGENT COMPUTING INTEGRATED CIRCUITS

GENPRO
The Israeli RISC-V Consortium

Hiroshi Fujiwara Cyber Security Research Center

HiPer

Israel Science Foundation

TCE
technion computer engineering center

WD

Prime Minister's Office
National Cyber Bureau

משרד המדע והטכנולוגיה
Israel Ministry of Science and Technology

RBNI

ICRI-CI
Intel Collaborative Research Institute Computational Intelligence
intel

erc European Research Council

HUAWEI

רשות החדשנות
Israel Innovation Authority

Advanced Circuit Research Center ACRC

CISCO

BSF
United States – Israel Binational Science Foundation