EPFL

TiC-SAT: Tightly-Coupled Systolic Accelerator for Transformers

Speaker: Alireza Amirshahi

Co-authors: Joshua Klein, Giovanni Ansaloni, David Atienza

Embedded Systems Laboratory (ESL), EPFL, Switzerland

Applications of transformers

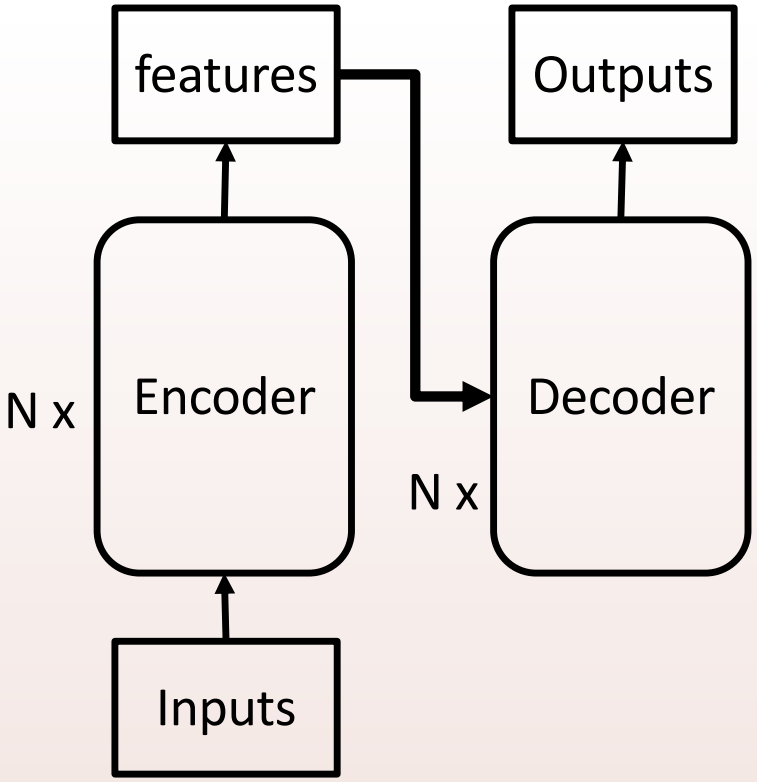
- DALL.E [1]: Text -> Image

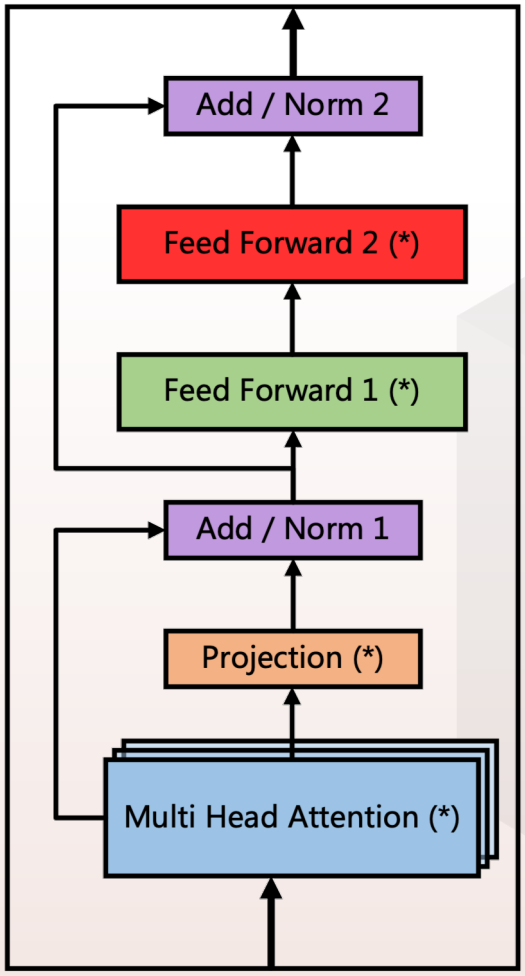
Input: *"A Ph.D. student talking in a conference in the planet of Mars in the year of 2123."*

[1] A. Ramesh et al. ArXiv 2022

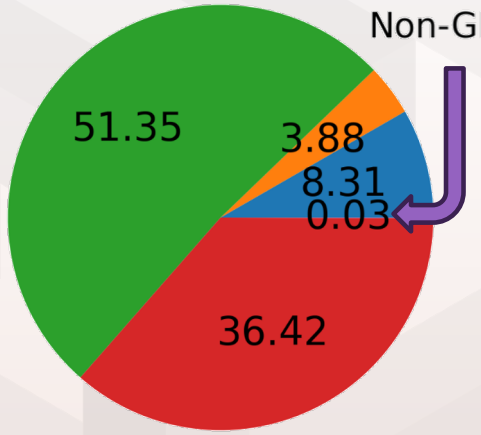
What is a transformer?

- An encoder-decoder structure





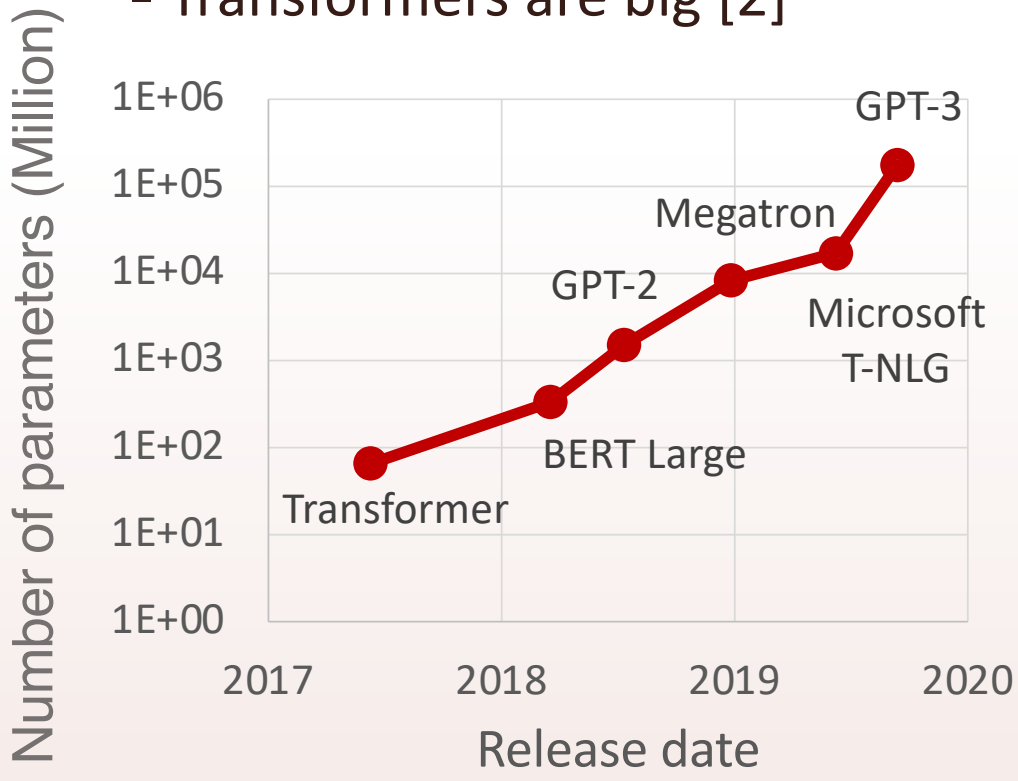
- Mostly General Matrix to Matrix Multiplication (GEMM)

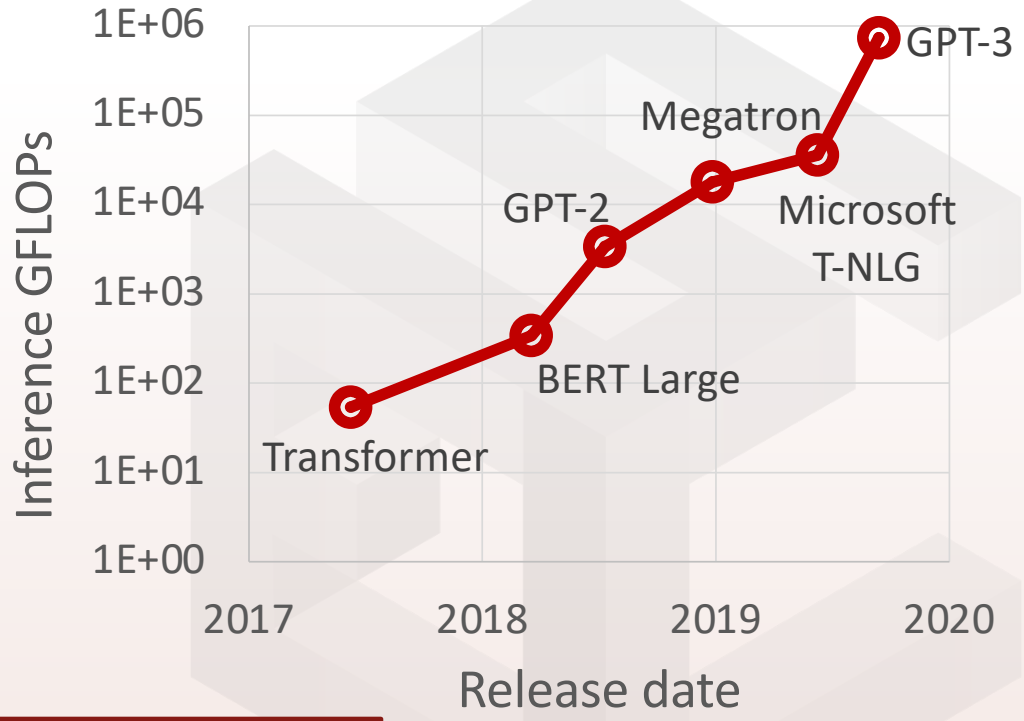


Execution time (%)

Problem definition

- Transformers are big [2]

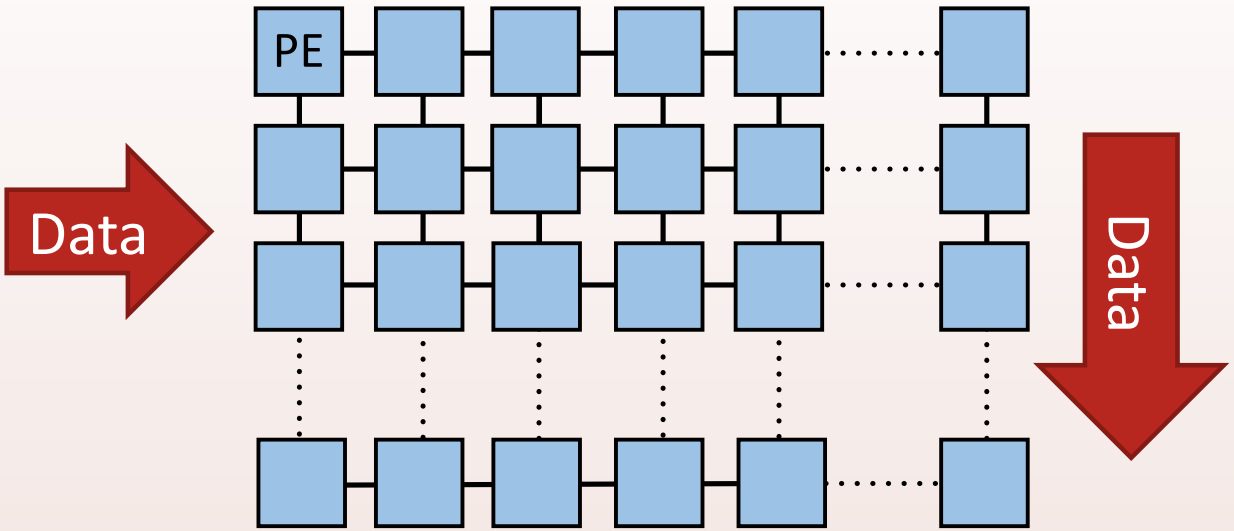




Need for GEMM accelerators for transformers

Our accelerator's core: A systolic array

- A homogeneous network of identical processing elements (PEs)
- Orchestrate flow of data between PEs based on the desired task

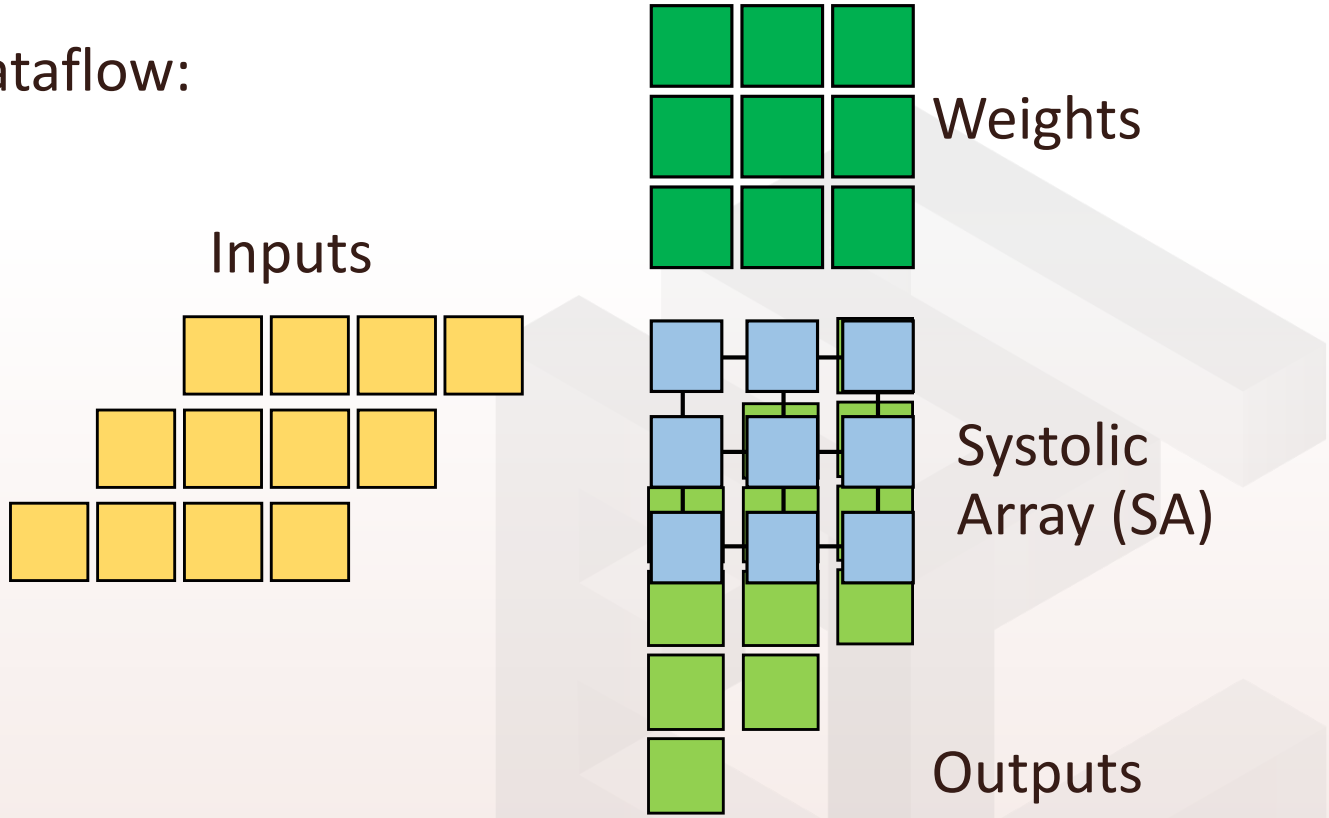


- Now popular, Example: TPU

[Google 2021]

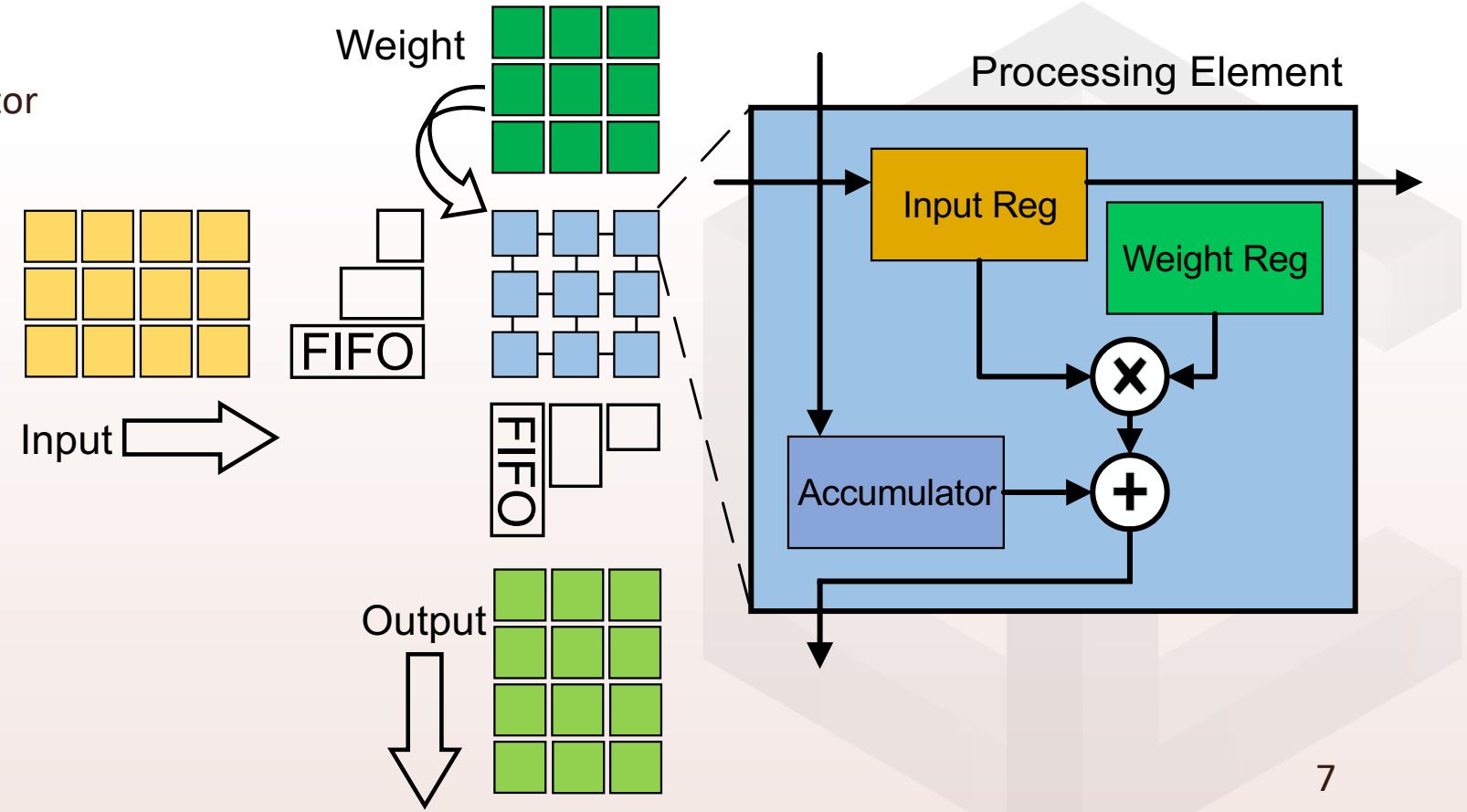
How do systolic arrays perform GEMM?

- Types of systolic array dataflow:
  - **Weight Stationary**
  - Input Stationary
  - Output Stationary



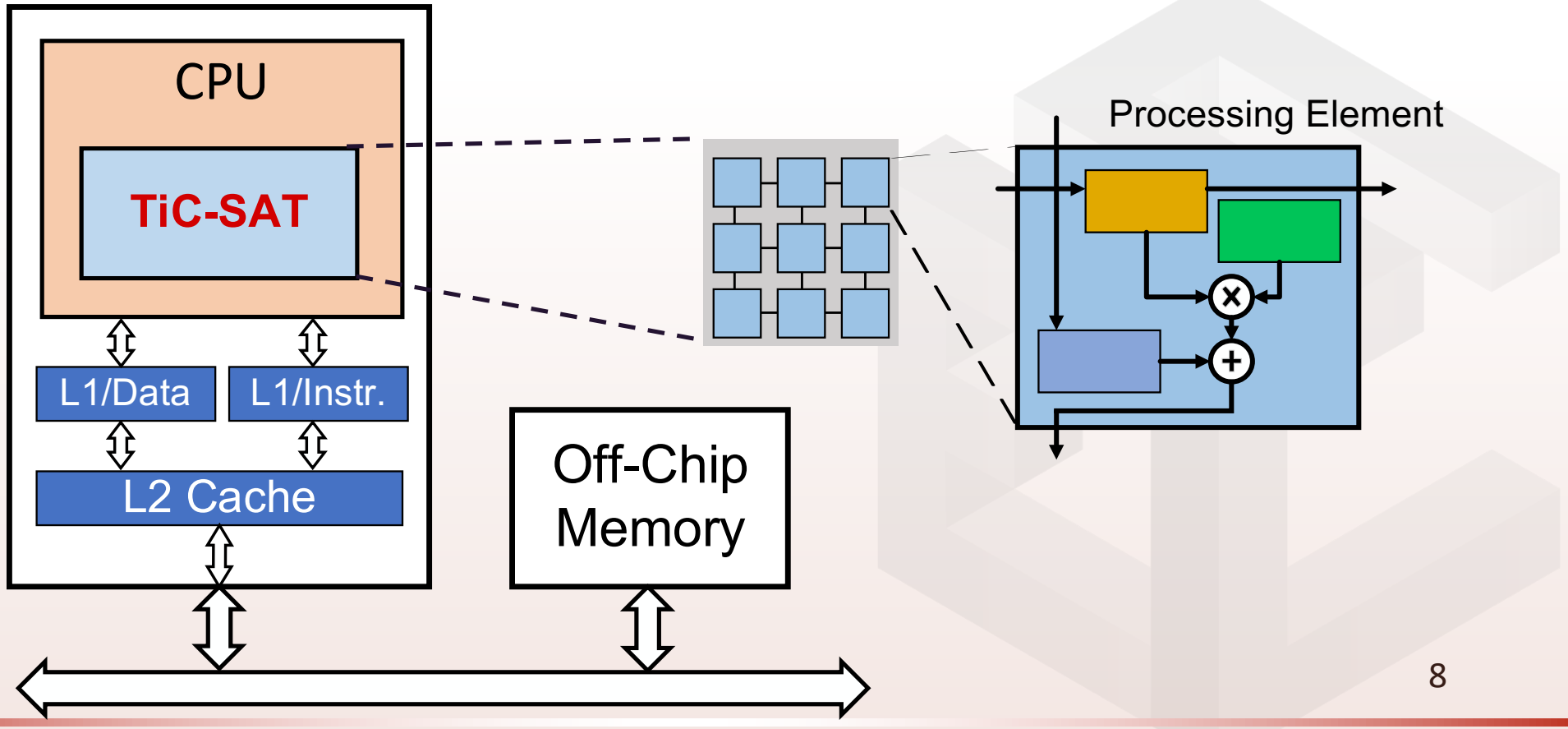
Implementing a systolic array

- Processing elements:
  - Registers
    - Accumulator
    - Input
    - Weight
  - Multiplier
  - Adder
- FIFOs:
  - Input
  - Output



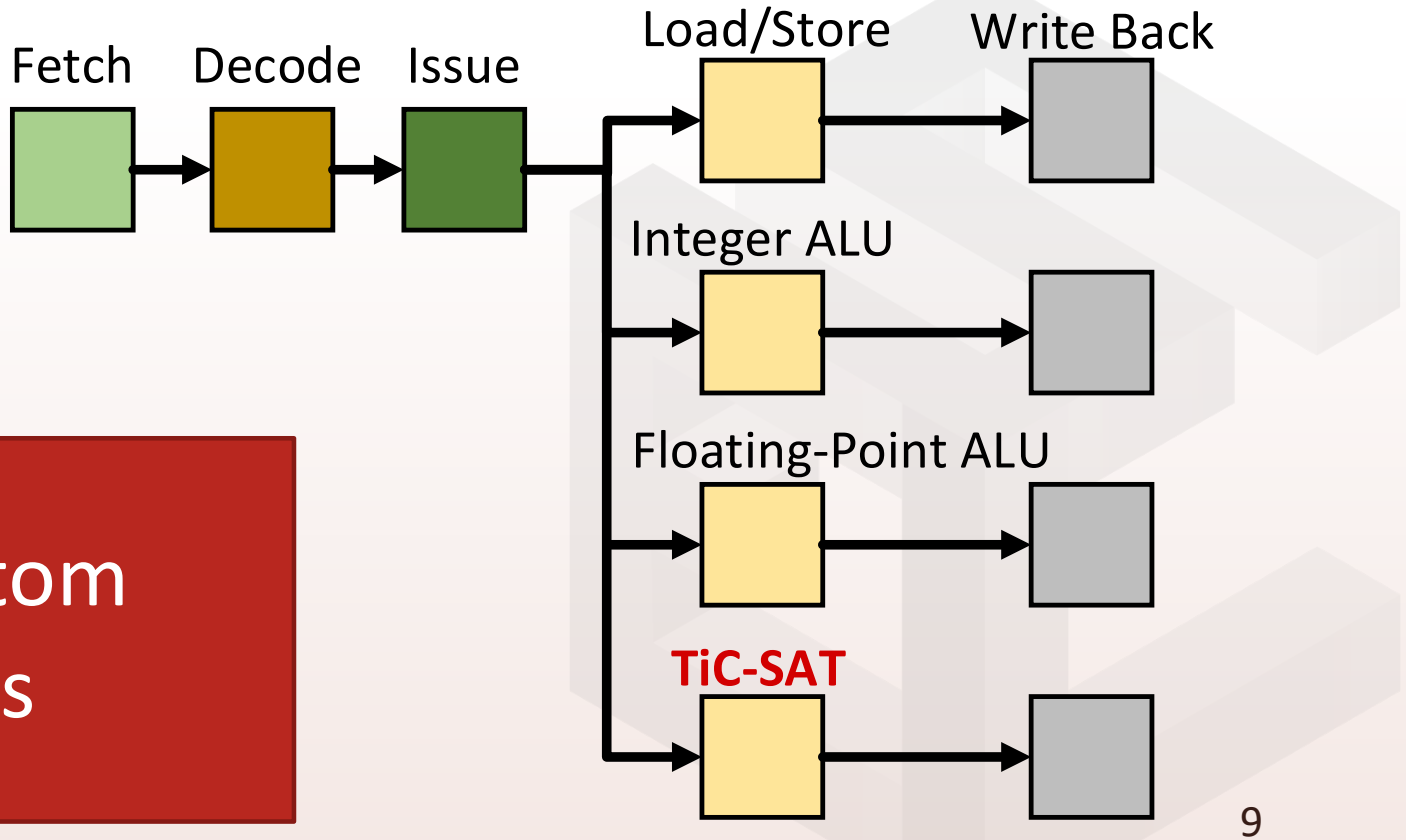
Our accelerator, TiC-SAT

- TiC-SAT: Tightly-Coupled Systolic Array for Transformers



Custom functional unit for TiC-SAT

- Custom functional unit in a CPU pipeline

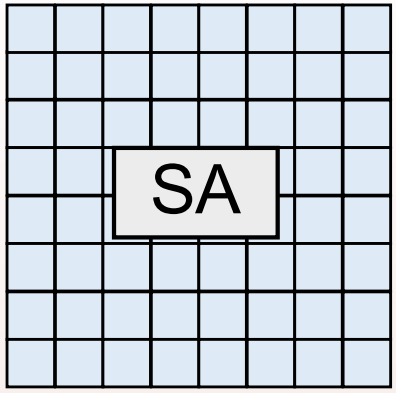


We need custom instructions

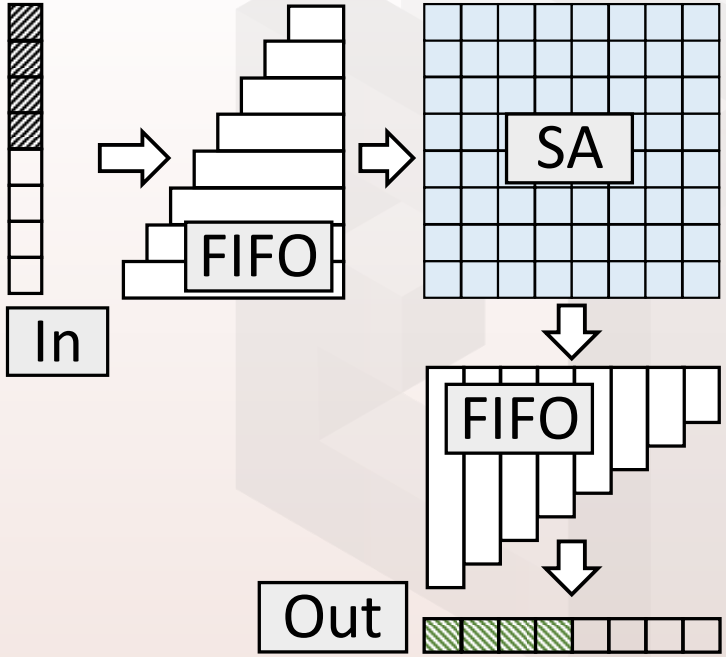
Custom instructions

- ①SA_LW:

Load weights



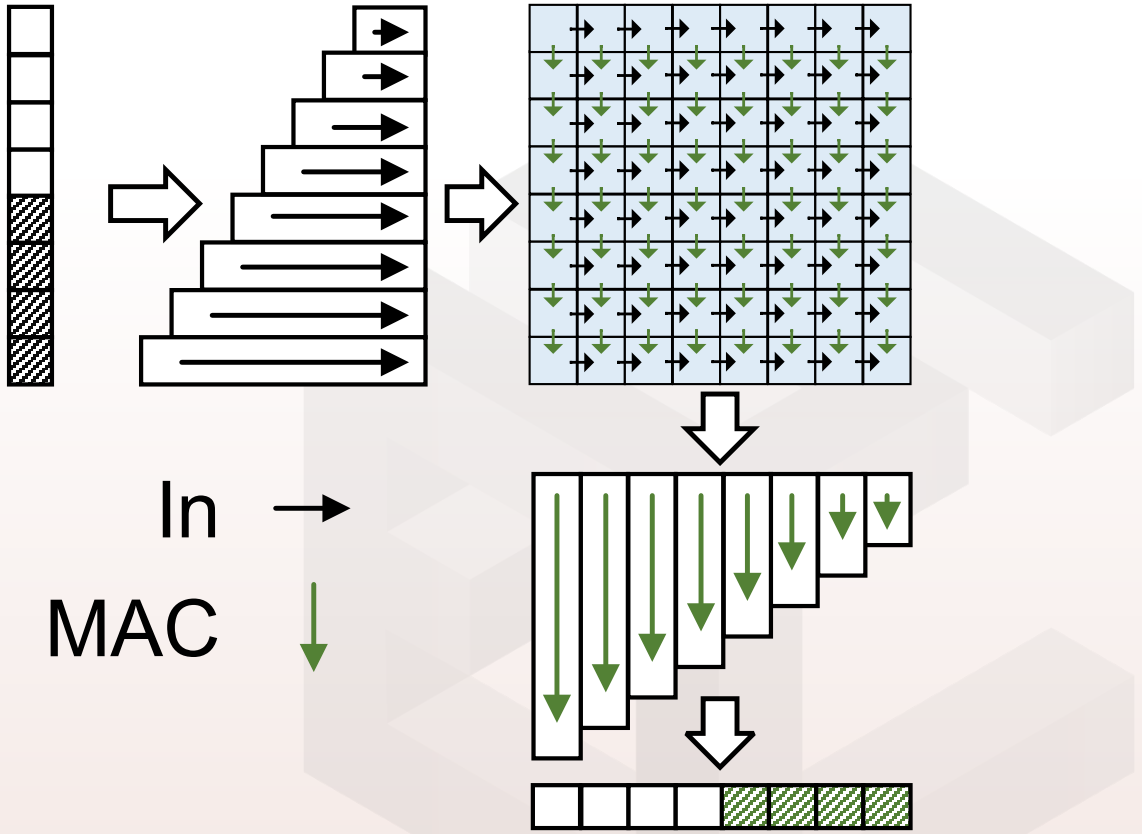
- ②SA_IO: Input, output
  - Send input, read output in the same time
  - Bus width constraint
    - (ex. 32-bit bus, 8-bit value -> Only 4 values)



Custom instructions (continued)

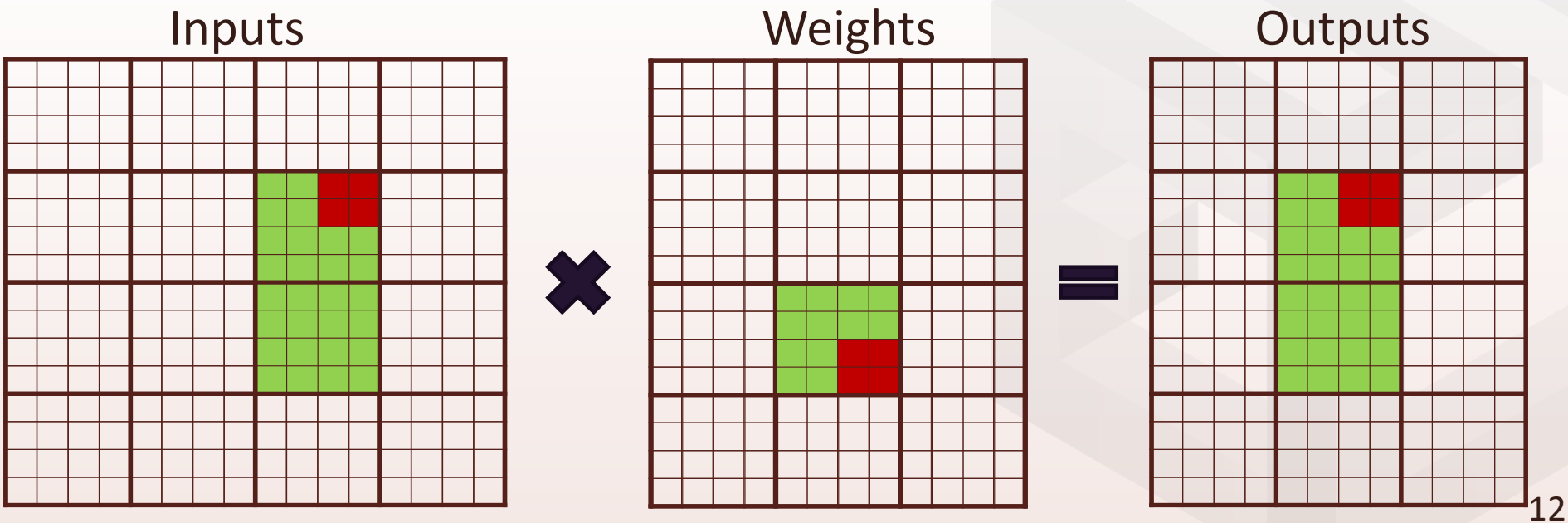
- ③ SA_IOC:

Input, output, and compute



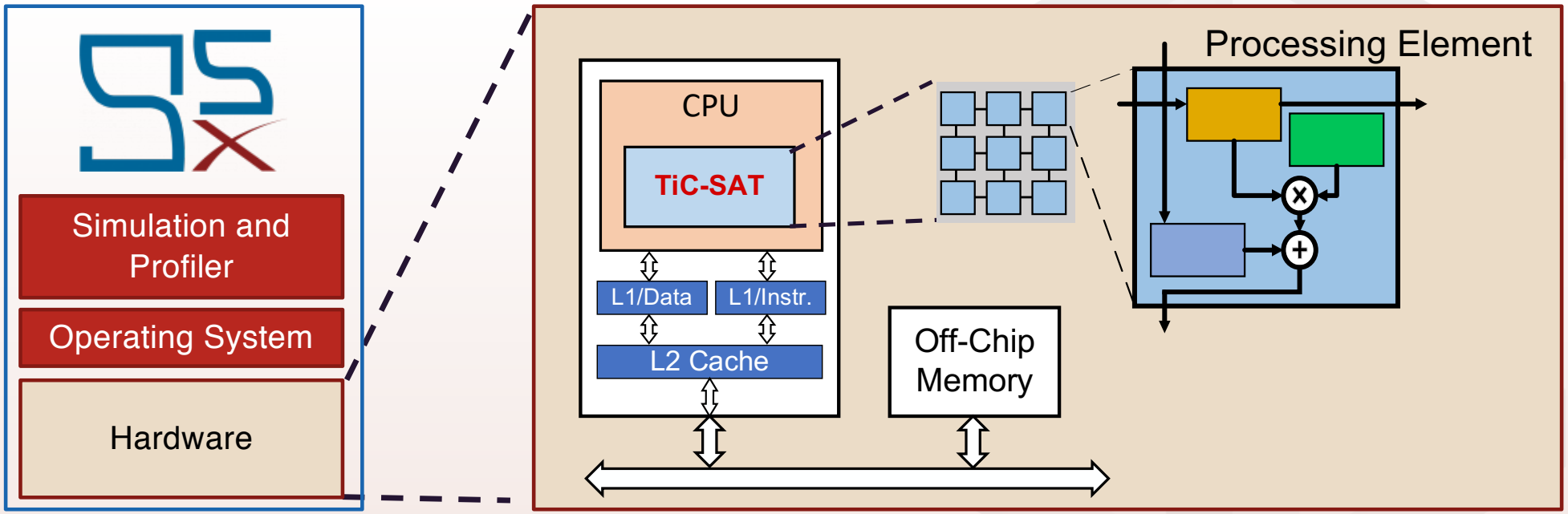
Maximize data reuse

- A transformer matrix is bigger than SA size
- It is even bigger than L1 cache size!
- TiC-SAT enables us to use cache hierarchy with tiling



Full-system simulation

- Simulate the TiC-SAT integration in a full-system simulation.
  - gem5-x is a full-system simulation tool [3].



[3] Y. Qureshi et al., HPC 2019

Configurations

- System:

CPU Core	Single-core in-order CPU @ 1 GHz
L1/L2 Cache Size	L1-Instruction: 32 KB, L1-Data: 32 KB, L2: 1 MB
Instruction Set Architecture (ISA)	ARMv8 (AArch64)
Main Memory	4 GB DDR4 @ 2400 MHz
Operating System	Ubuntu 16.04 LTS

- Application: BERT-Large [4]

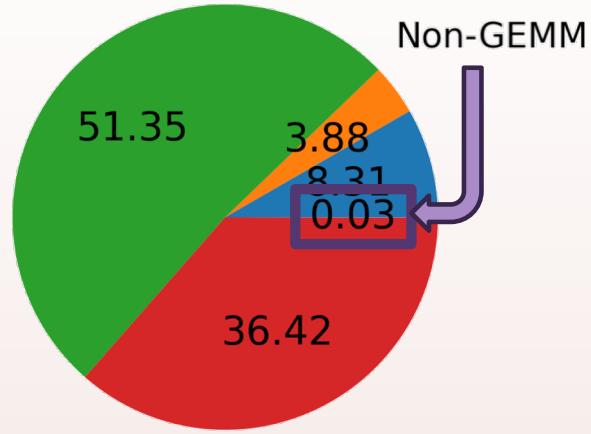
Sequence Length	512
Model Dimension	1024
Feed-forward Dimension	4096
Number of Heads	16
Total Number of Parameters	340 M

- TiC-SAT systolic-array size:
  - 16*16
  - 8*8
  - 4*4

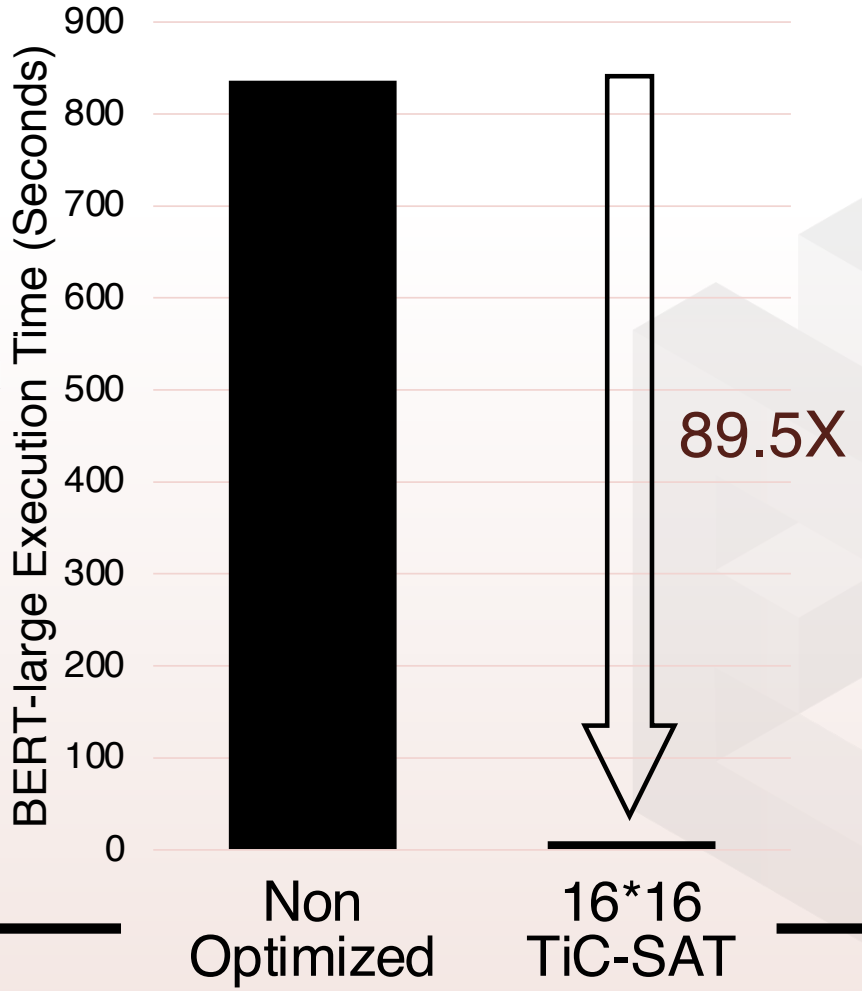
[4] J. Devlin et al., ArXiv 2019

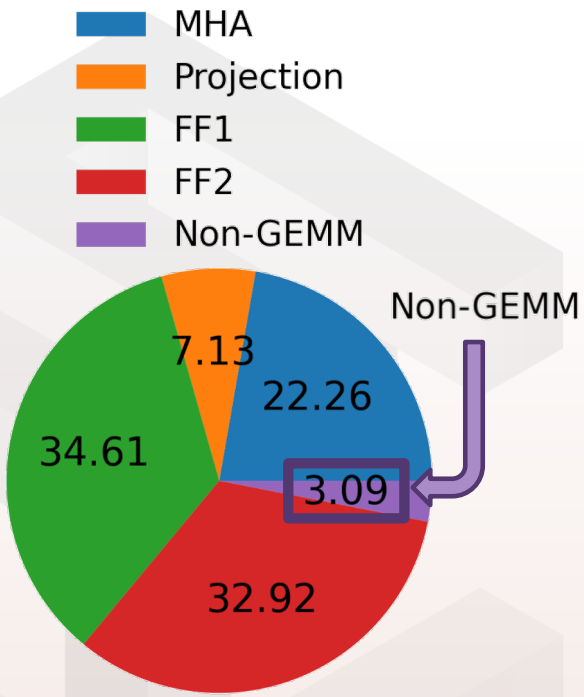
Compare timing results with a baseline

The GEMM operations are still dominant in the execution time.



Execution time (%)

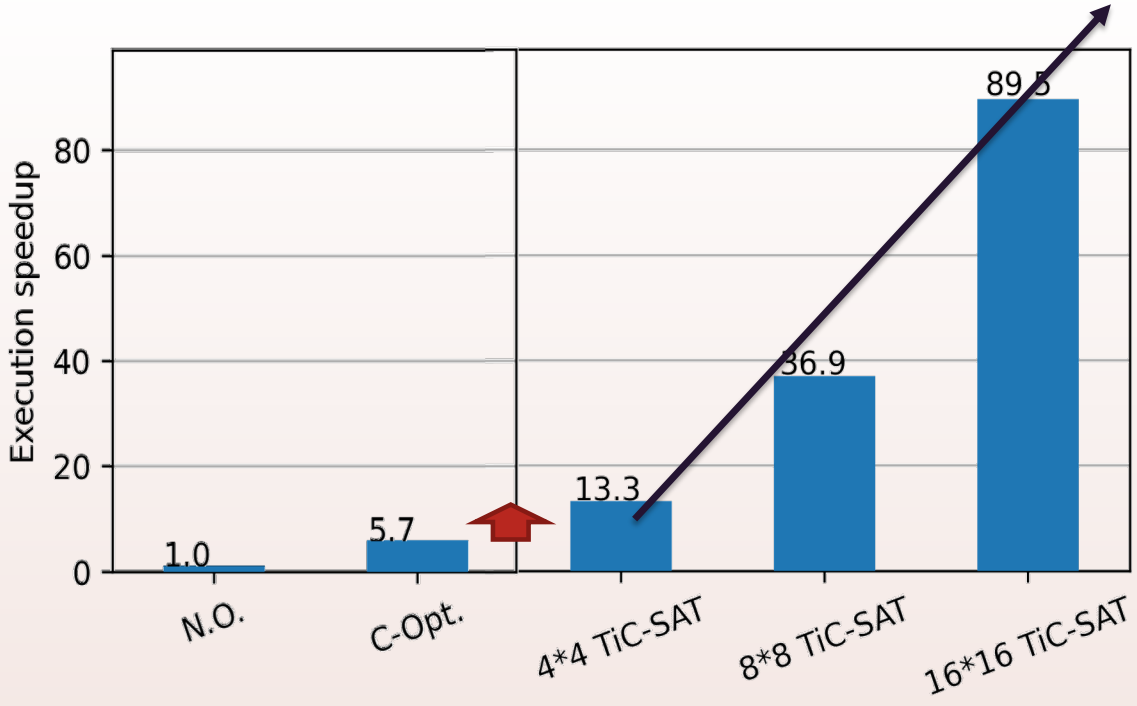




Execution time (%)

Compare with another baseline

- Compare with the baselines:
  - Baseline1: Non-optimized (N.O.)
  - Baseline2: Optimized cache utilization (C-Opt.)



Even a 4*4 TiC-SAT speeds up the execution.

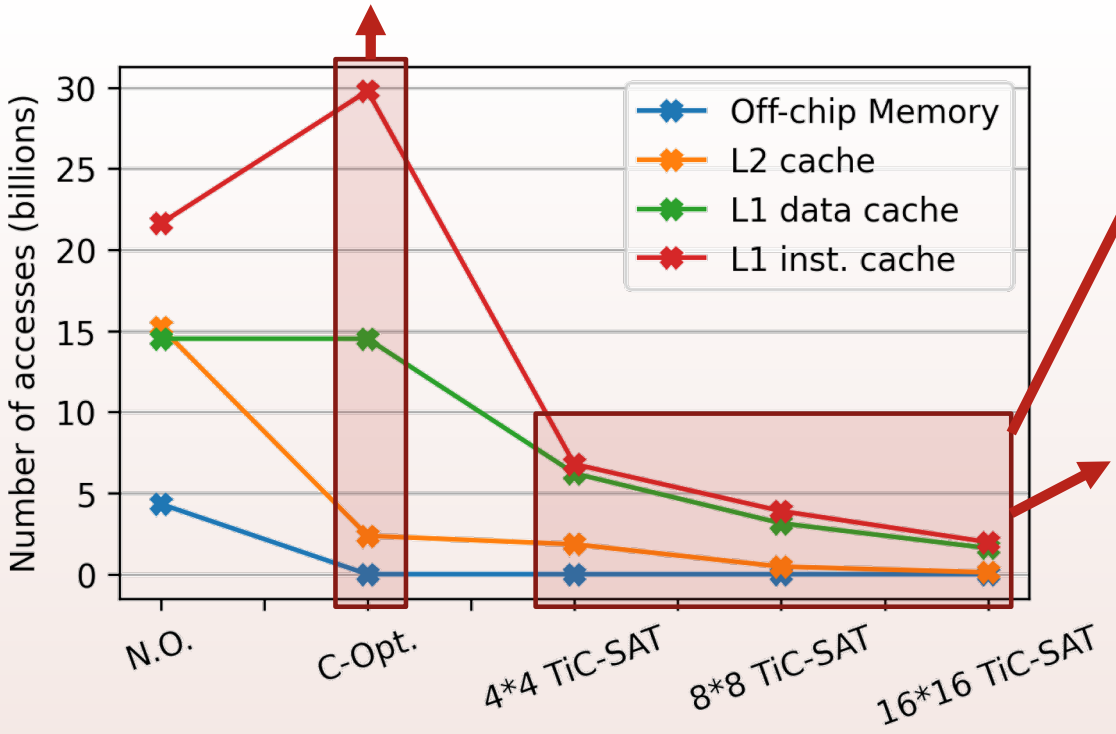
The Speed-up grows almost linearly with the TiC-SAT size.

Data locality impact

More complex loop structure =>

40% More instructions ☹

5X More data reutilization ☺



MAC operations in on TiC-SAT instruction =>

70% Less instructions ☺

Values resident in TiC-SAT =>

16X More data reutilization ☺

Comparison with a loosely-coupled

- Based on SMAUG[5]
- As a loosely-coupled structure

- Structure modification to have a fair comparison w.r.t. area:
  - SMAUG scratchpad size: 3x8 KB
  - TiC-SAT systolic array size: 64*64

System	TiC-SAT	SMAUG
CPU Frequency	1 GHz	1 GHz
L1 Cache Size	32 KB	32 KB
L2 Cache Size	1 MB	1 MB
CPU Type	Out-of-order	Out-of-order
Simulation Tool	gem5-x	gem5-aladdin
Technology	28nm	16nm
Area (mm^2)	0.70	0.61
Speed-up	**234x**	**88x**

2.6 X

[5] S. Xi et al., TACO 2020

Conclusion

- Contributions:
  - A tightly-coupled strategy with systolic arrays and new custom instructions
  - Integration in a full system
- Speed-up:
  - 89.5X using a 16*16 TiC-SAT over the baseline
  - 2.6X over a loosely-coupled implementation

Thank you!

https://github.com/gem5-X/TiC-SAT