Friedrich-Alexander-Universität
Technische Fakultät

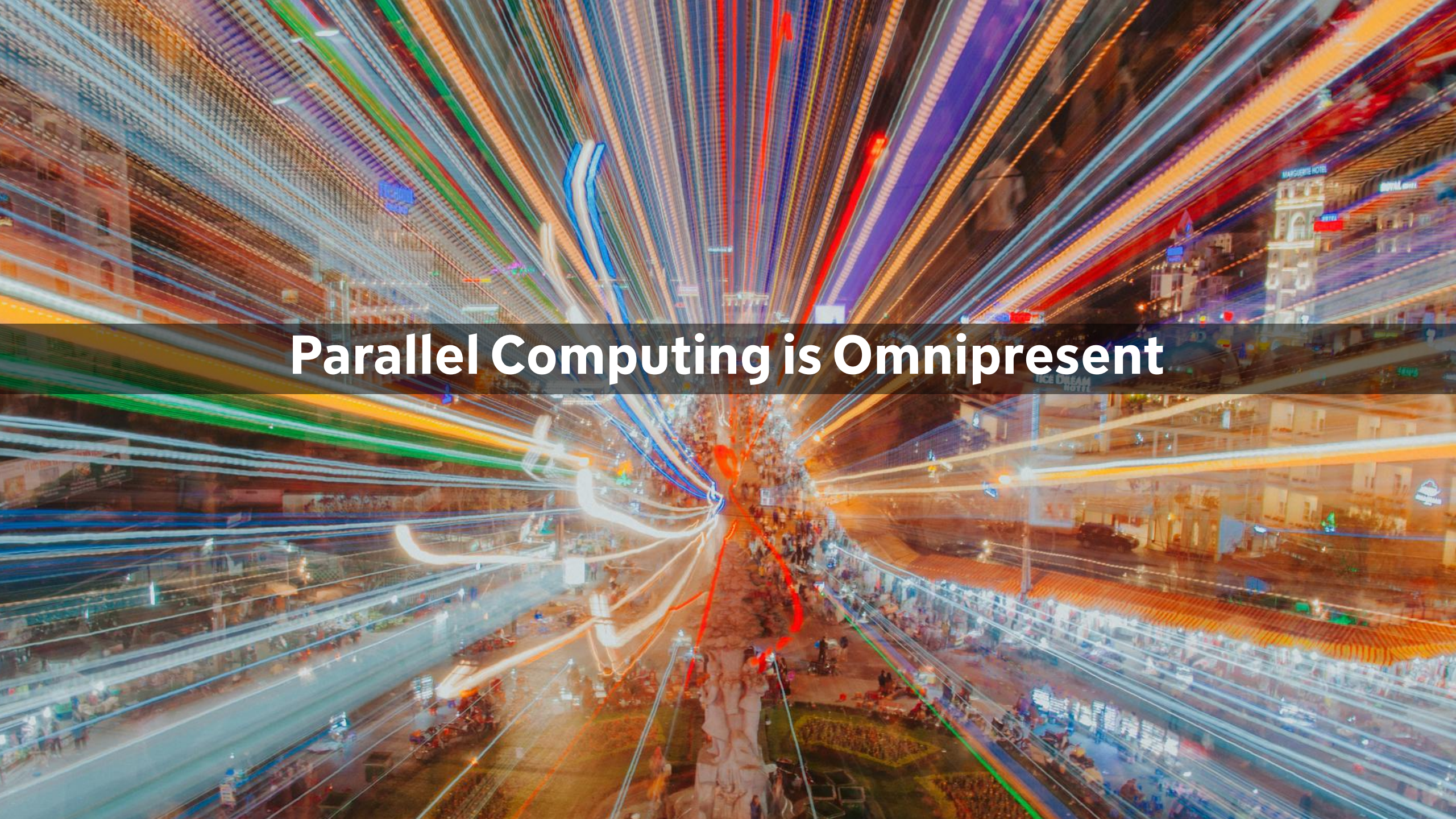
(Re-)Configurable Processor Arrays On-Chip – "Low Energy/High Performance Loop Nest Acceleration"

Talk at NHR PerfLab Seminar

Jürgen Teich

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

July 22, 2025

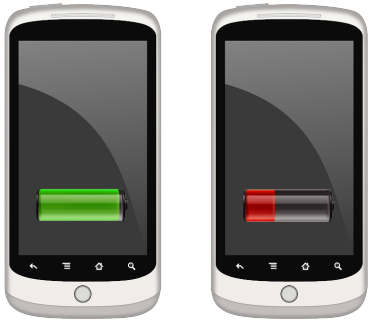

Parallel Computing is Omnipresent

Motivation and introduction

An era in which **computing power** (performance) is **limited by electrical power**

- High-performance computers and data centers
  - Energy costs
  - Costs for infrastructure

- Mobile devices
  - battery operated
  - passively cooled



- Medical devices

Motivation and introduction

Power wall

- **Power wall:** Only a constant power budget is available per device
- At the same time, we expect an increase in performance

  $\Rightarrow$ Energy per operation must be reduced

$$Power = Energy_{Op} \times \frac{Ops}{second}$$

$$= \quad \Downarrow \quad \Uparrow$$

- These days, **energy efficiency** (FLOPS/Watt) is much more important than pure computing power (FLOPS)
- This trend goes across all levels, from portable devices and embedded devices to HPC systems

Motivation and introduction

It's all about energy efficiency

- How can greater energy efficiency be achieved?

  ... through heterogeneous hardware

  - Multiple processor cores
  - + instruction level parallelism (ILP)
  - + vector units, i.e., SIMD processing
  - + instruction set extensions
  - + accelerators (GPUs, FPGAs, ASICs)

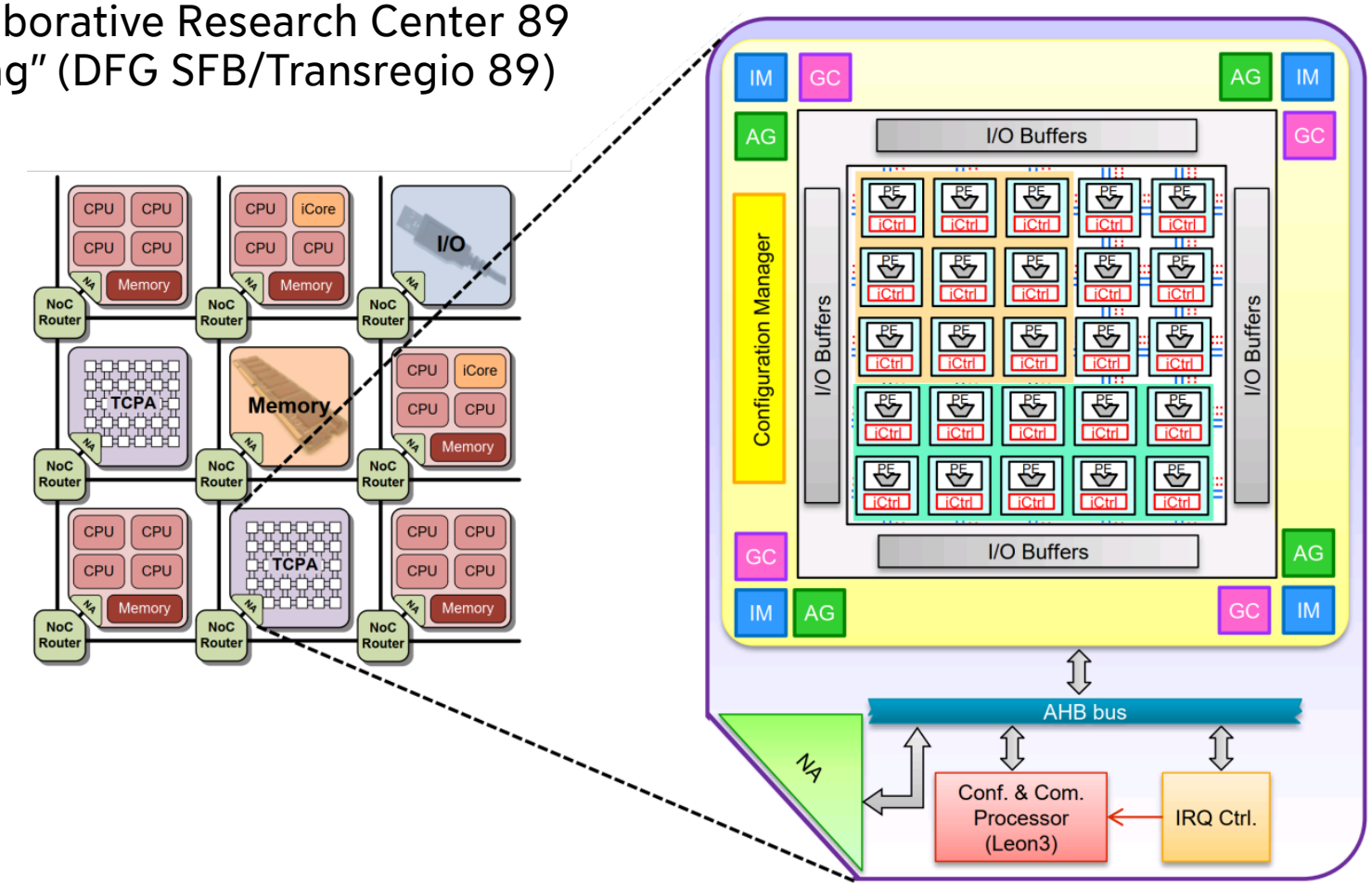
Source: http://www.phys.ncku.edu.tw/~htsu/humor/fry_egg.html

⇒ **Customization and heterogeneity are the keys to success for future performance gains**

An example

Tiled architectures developed in “Invasive Computing”[1,2]

Transregional Collaborative Research Center 89
“Invasive Computing” (DFG SFB/Transregio 89)



[1] J. Teich, J. Henkel, A. Herkersdorf, D. Schmitt-Landsiedel, W. Schröder-Preikschat, and G. Snelting. “Invasive Computing: An Overview”. In: *Multiprocessor System-on-Chip – Hardware Design and Tool Integration*. Springer, 2011, pp. 241–268. DOI: 10.1007/978-1-4419-6460-1_11.

[2] N. Anantharajaiah et al. *Invasive Computing*. Ed. by J. Teich, J. Henkel, and A. Herkersdorf. FAU University Press, Aug. 22, 2022. DOI: 10.25593/978-3-96147-571-1.

What are processor arrays good for?

Application domains

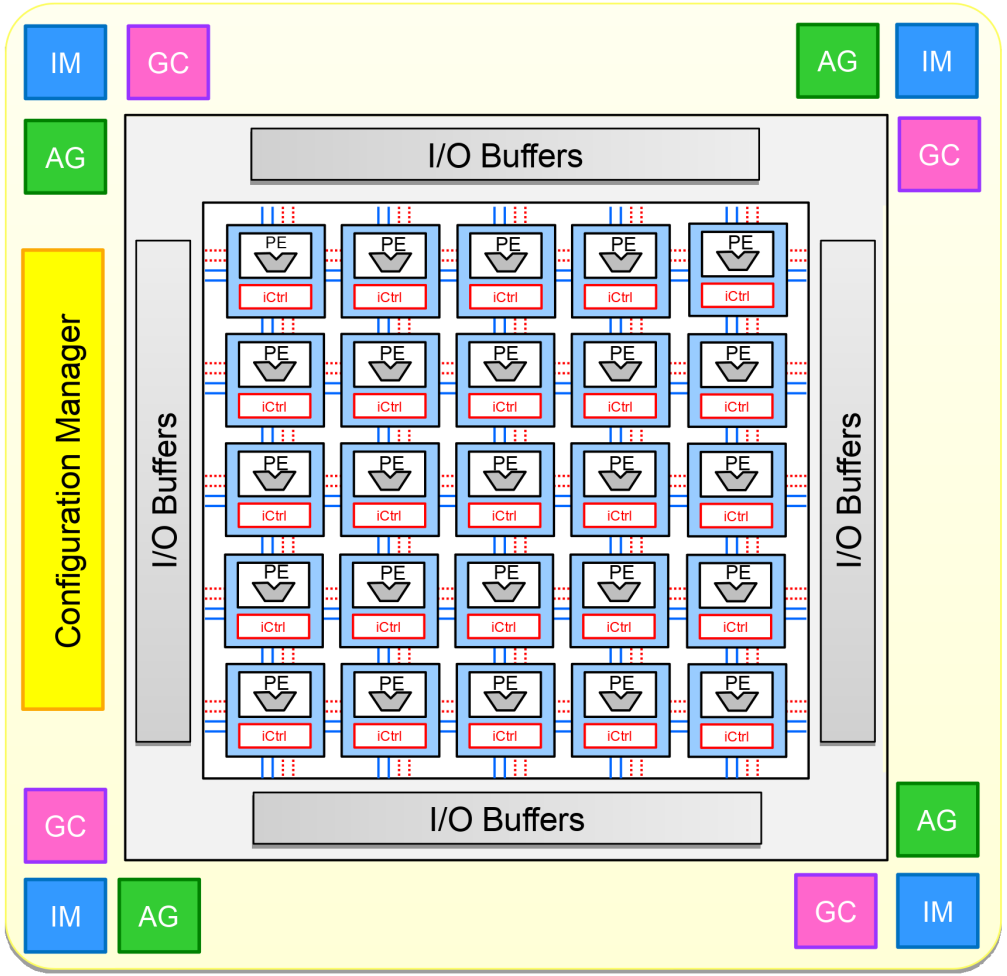
- Most of the execution time in scientific computing and digital signal processing applications is spent on a small fraction of code
- **90/10 law** — 90 % of the execution time comes from 10 % of the code
- Profiling to identify computationally intensive inner loops
- Examples of dataflow-dominant application domains:
  - Linear algebra: GEMM, matrix decomposition (LU, QR, SVD), least-square methods, bilinear and trilinear interpolation, ...
  - Video, audio, and other digital processing: FIR, AR, Kalman filter, ...
  - Image processing: median filter, Hough transformation, 2D convolution
  - Reconstruction: algebraic reconstruction technique (ART), Kaczmarz, filtered back projection (FBP)
  - Edge detection, feature extraction, ...
  - Artificial neural networks, especially convolutional neural networks (CNNs)

Tightly coupled processor arrays[3] (TCPAs)

A class of highly parameterizable processor arrays

- Processor array
  - Weakly programmable processing elements (PE)
  - Reconfigurable interconnect
- Exploiting
  - Loop-level parallelism
  - Instruction-level parallelism
  - Word-level parallelism
- Data streamed through the processor array
- Peripherals
  - I/O buffers
  - Address generators (AG)
  - Global controllers (GC)
  - Configuration manager (CM)

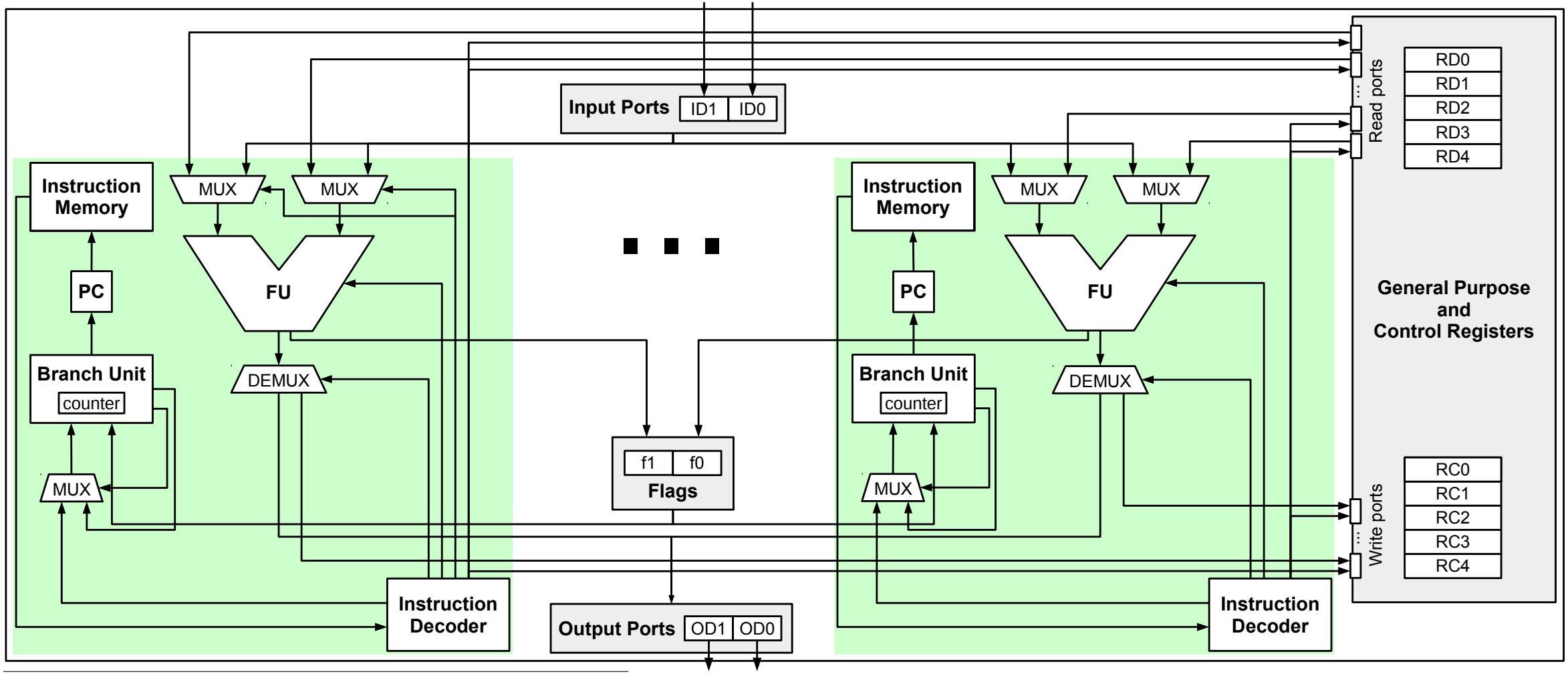
⇒ **Co-design of architecture and compiler**



[3] F. Hannig, V. Lari, S. Boppu, A. Tanase, and O. Reiche. "Invasive Tightly-Coupled Processor Arrays: A Domain-Specific Architecture/Compiler Co-Design Approach". In: *ACM Transactions on Embedded Computing Systems (TECS)* 13.4s (Apr. 2014), 133:1–133:29. DOI: 10.1145/2584660.

Orthogonal instruction processing[4]

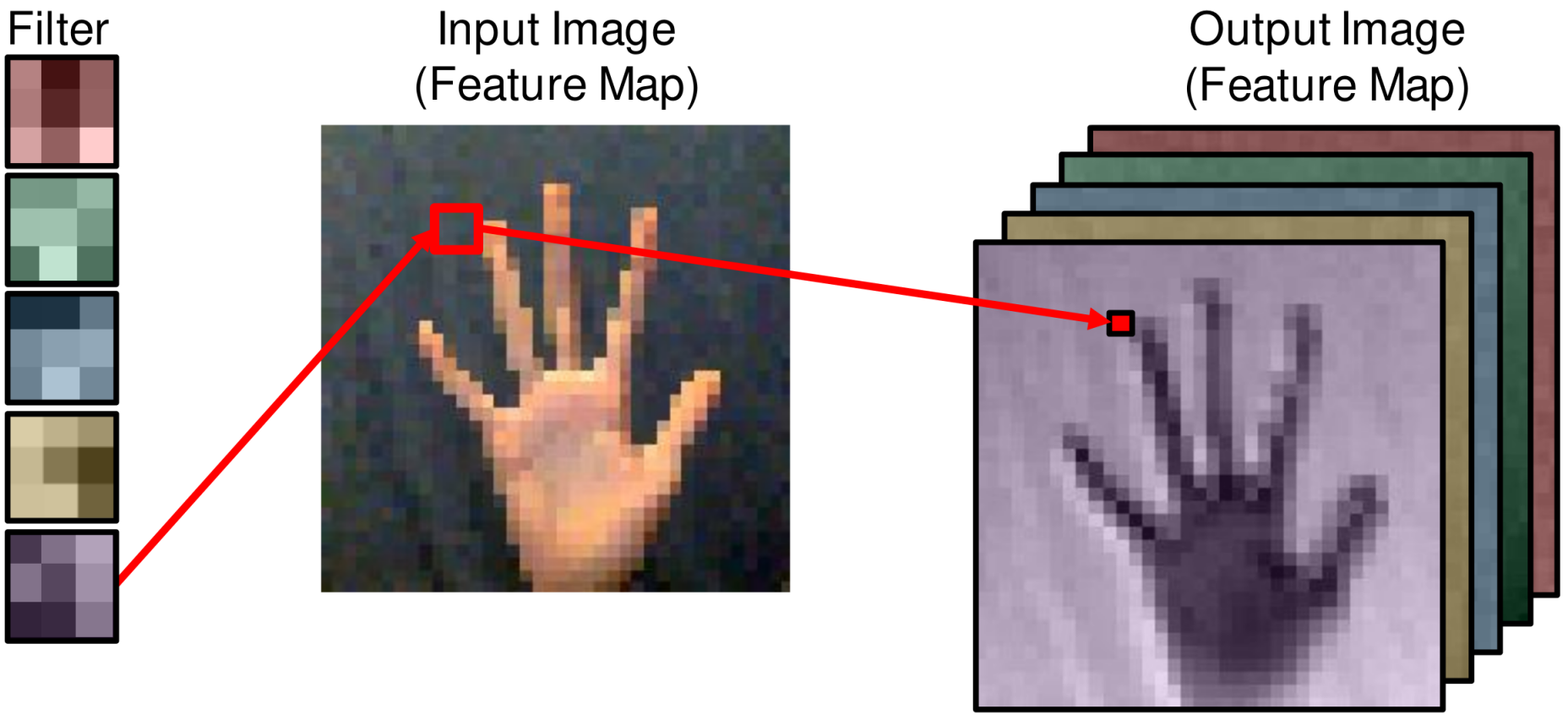
An alternative to VLIW



[4] M. Brand, F. Hannig, A. Tanase, and J. Teich. "Orthogonal Instruction Processing: An Alternative to Lightweight VLIW Processors". In: *Proceedings of the IEEE 11th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip (MCSoC)*. IEEE Computer Society, Sept. 18–20, 2017, pp. 5–12. DOI: 10.1109/MCSoC.2017.17.

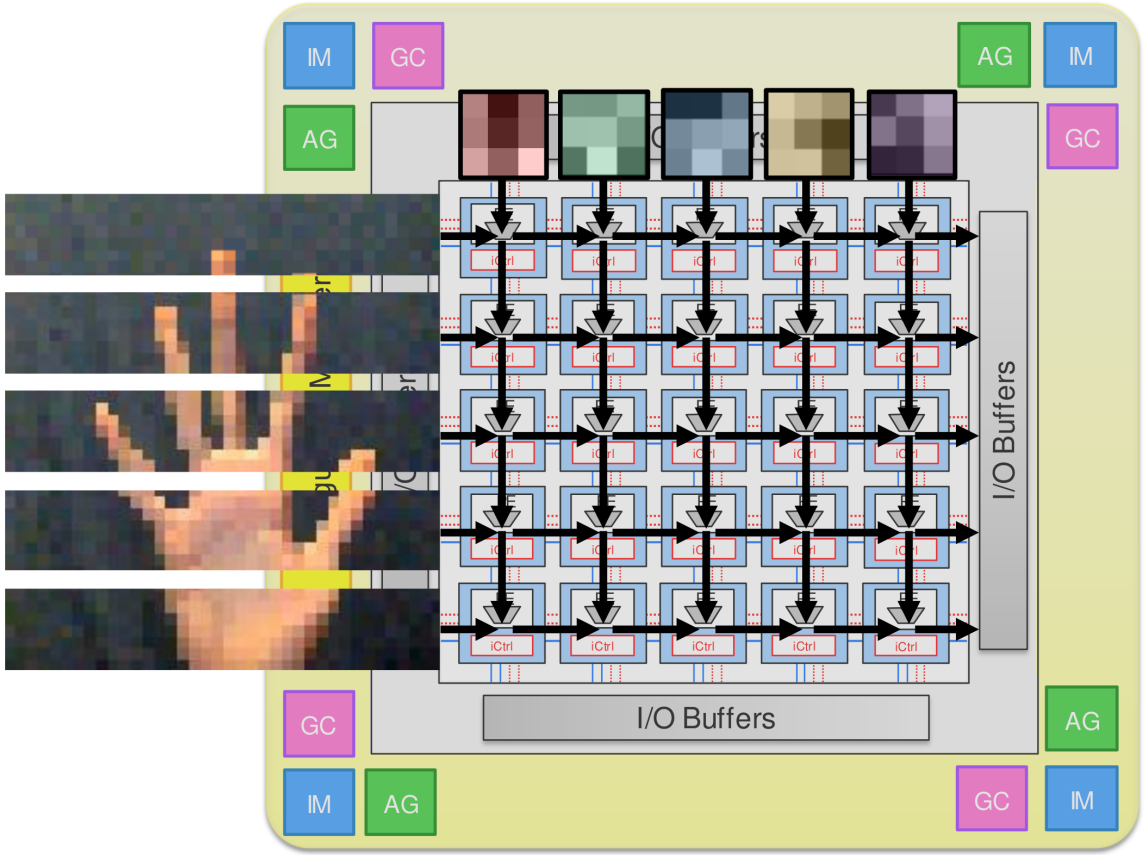
Execution model by example

Convolutions in CNNs



Execution model by example

Mapping of CNNs onto TCPAs[5,6]



[5] C. Heidorn, M. Witterauf, F. Hannig, and J. Teich. "Efficient Mapping of CNNs onto Tightly Coupled Processor Arrays". In: *Journal of Computers* 14.8 (Aug. 2019), pp. 541–556. DOI: 10.17706/jcp.14.8.541-556.

[6] C. Heidorn, D. Walter, Y. Candir, F. Hannig, and J. Teich. "Hand Sign Recognition via Deep Learning on Tightly Coupled Processor Arrays". In: *Proceedings of the 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, Aug. 30–Sept. 3, 2021, p. 388. DOI: 10.1109/FPL53798.2021.00079.

Programming and Compilation

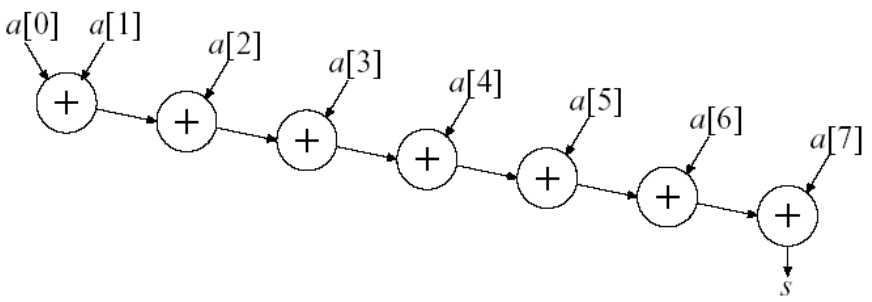
Why not start from C code?

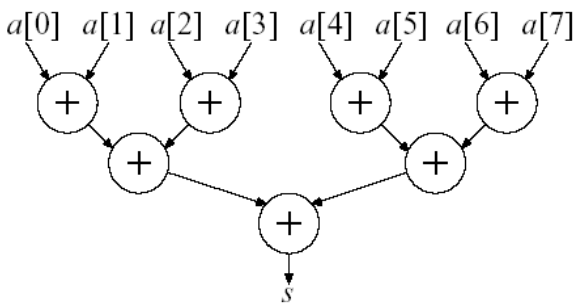
- Most existing high-level synthesis (HLS) tools and CGRA compilers start from C/C++ code
- Limitation:
  Semantics of input language (statement order, loop order) define execution order

  ⇒ **limited parallelism**

  Example:

```
int s = a[0];
for (i=1; i<=7; i++)
  s += a[i];
```





Tools and compilers have only a few high-level transformations in order to parallelize code

Domain-specific language for loops

PAULA

- **PAULA**[7], language features
  - Based on the algorithm class of *dynamic piecewise linear algorithms* (DPLA)[8]
  - Functional programming language
  - Full static single assignment (SSA) form, also for multi-dimensional arrays
  - Expressions for the specification of polyhedral and lattice iteration domains
  - Handling of run-time dependent control flow to a certain degree
  - Reductions such as $\prod$ or $\sum$
  - Architectural modeling capabilities
- **Semantic model** (intermediate representation)
  - *Extended reduced dependence graph* (RDG)[9]
  - Non-perfectly nested loop programs
  - Indexed variables with affine indexing functions

[7] F. Hannig, H. Ruckdeschel, and J. Teich. "The PAULA Language for Designing Multi-Dimensional Dataflow-Intensive Applications". In: *Proceedings of the GI/ITG/GMM-Workshop – Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen* (Freiburg, Germany). Shaker, Mar. 3–5, 2008, pp. 129–138.

[8] F. Hannig and J. Teich. "Resource Constrained and Speculative Scheduling of an Algorithm Class with Run-Time Dependent Conditionals". In: *Proceedings of the 15th IEEE International Conference on Application-specific Systems, Architectures, and Processors (ASAP)*. IEEE Computer Society, Sept. 27–29, 2004, pp. 17–27. DOI: 10.1109/ASAP.2004.1342455.

[9] F. Hannig. "Scheduling Techniques for High-Throughput Loop Accelerators". Dissertation. University of Erlangen-Nuremberg, Germany, Aug. 11, 2009. 307 pp.

Internal representation

Extended RDG

$$\mathcal{I} = \{(i\ j\ k)^\mathsf{T} \in \mathbb{Z}^3 \mid 1 \le i,j,k \le N\}$$

$$S_1: \quad a[i,j,k] = a_{in}[i,k] \qquad \text{if } j = 1$$

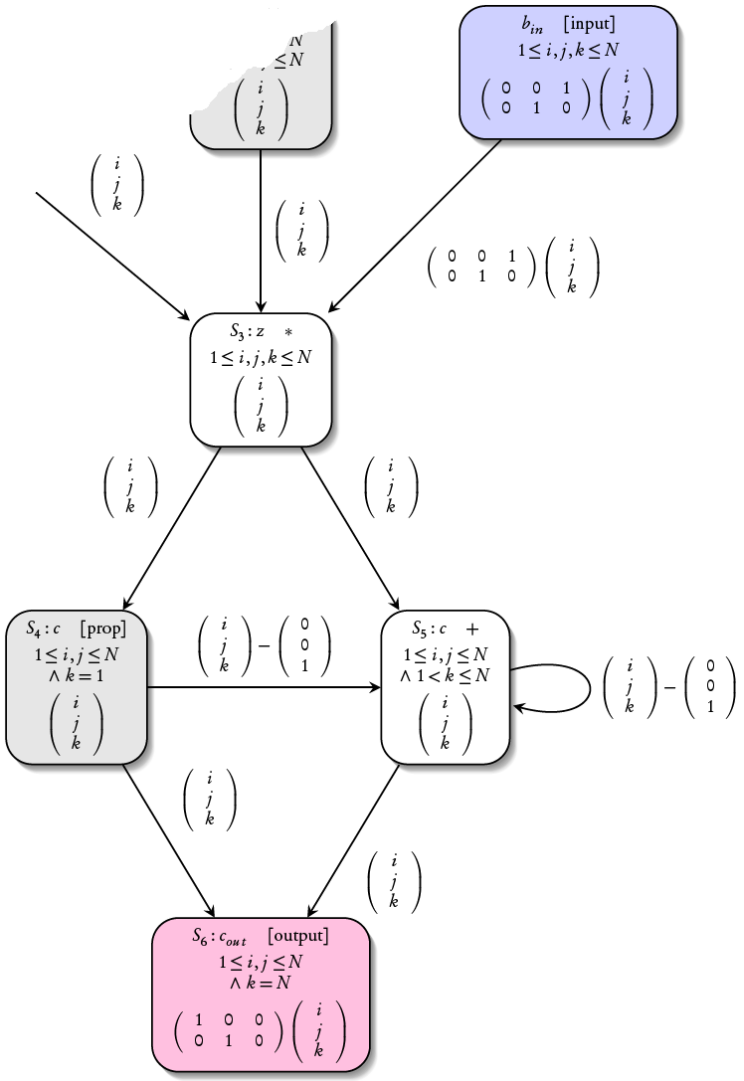
$$S_2: \quad a[i,j,k] = a[i,j-1,k] \qquad \text{if } j > 1$$

$$S_3: \quad z[i,j,k] = a[i,j,k] \cdot b_{in}[k,j]$$

$$S_4: \quad c[i,j,k] = z[i,j,k] \qquad \text{if } k = 1$$

$$S_5: \quad c[i,j,k] = c[i,j,k-1] + z[i,j,k] \qquad \text{if } k > 1$$

$$S_6: \quad c_{out}[i,j] = c[i,j,k] \qquad \text{if } k = N$$



A symbolic approach to loop programs

Mapping[10] and compilation[11,12]

for $0 \le i_0 < N$ and $0 \le i_1 < M$ **do**

$b[i_0, i_1] = B_{in}[i_1]$ **if** $i_0 = 0$

$b[i_0, i_1] = b[i_0 - 1, i_1]$ **if** $i_0 > 0$

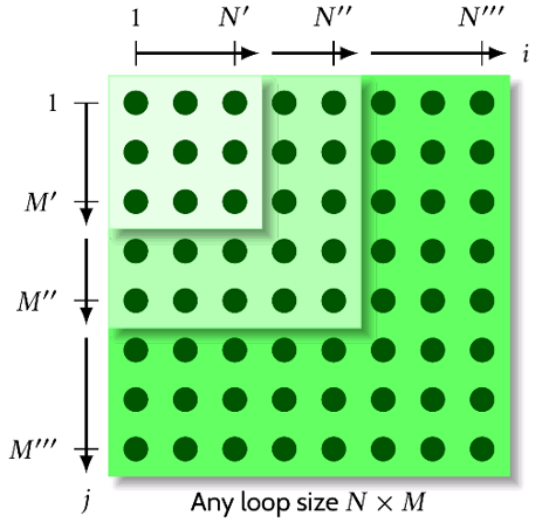
$t[i_0, i_1] = A_{in}[i_0, i_1] * b[i_0, i_1]$

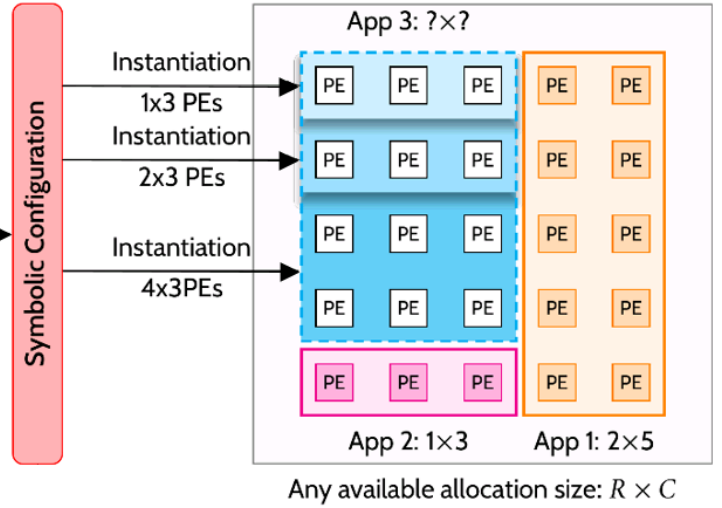
$u[i_0, i_1] = t[i_0, i_1]$ **if** $i_1 = 0$

$u[i_0, i_1] = u[i_0, i_1 - 1] + t[i_0, i_1]$ **if** $i_1 > 0$

$O_{out}[i_0] = u[i_0, i_1]$ **if** $i_1 = M - 1$

end for





Design Time: **Loop Program** Compile Time: **Symbolic Mapping** Run Time: **Instantiation**

[10] J. Teich, A. Tanase, and F. Hannig. "Symbolic Parallelization of Loop Programs for Massively Parallel Processor Arrays". In: *Proceedings of the 24th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. Best Paper Award. IEEE, June 5–7, 2013, pp. 1–9. DOI: 10.1109/ASAP.2013.6567543.

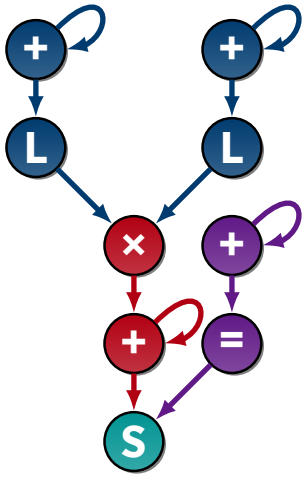
[11] M. Witterauf, A. Tanase, F. Hannig, and J. Teich. "Modulo Scheduling of Symbolically Tiled Loops for Tightly Coupled Processor Arrays". In: *Proceedings of the 27th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, July 6–8, 2016, pp. 58–66. DOI: 10.1109/ASAP.2016.7760773.

[12] M. Witterauf, D. Walter, F. Hannig, and J. Teich. "Symbolic Loop Compilation for Tightly Coupled Processor Arrays". In: *ACM Transactions on Embedded Computing Systems (TECS)* 20.5 (July 2021), 49:1–49:31. DOI: 10.1145/3466897.

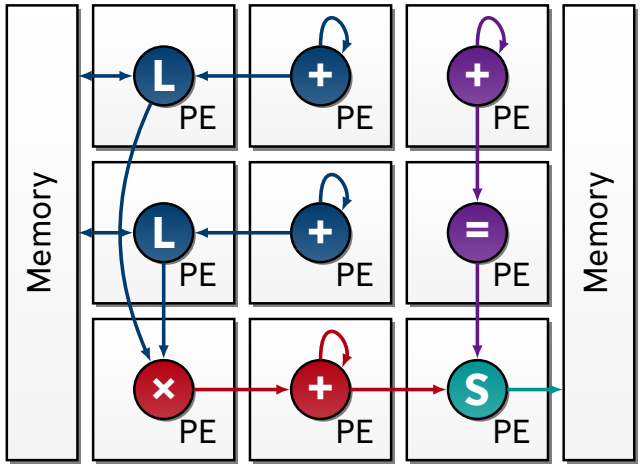
Different mapping strategies

CGRA vs. TCPA

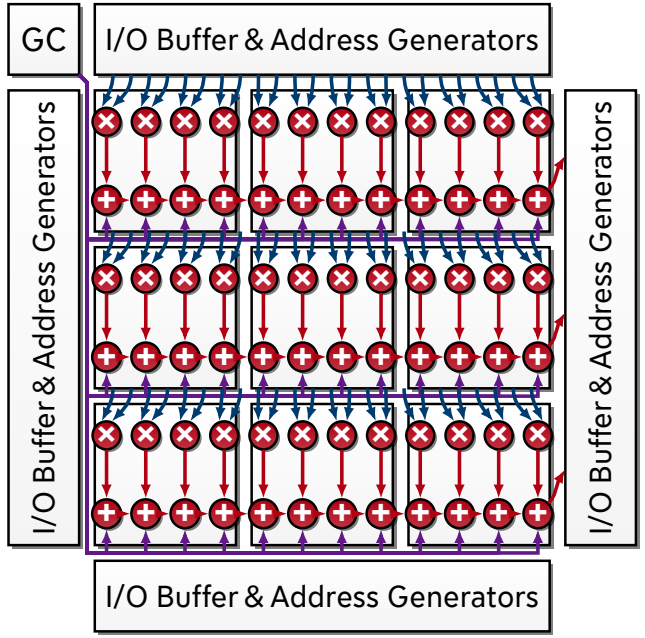
Scalar Product



CGRA Mapping[13,14]



TCPA Mapping[15]



[13] C. Tan, C. Xie, A. Li, K. J. Barker, and A. Tumeo. "OpenCGRA: An Open-Source Unified Framework for Modeling, Testing, and Evaluating CGRAs". In: *Proceedings of the IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 381–388. DOI: 10.1109/ICCD50377.2020.00070.

[14] D. Wijerathne, Z. Li, M. Karunaratne, L.-S. Peh, and T. Mitra. "Morpher: An Open-Source Integrated Compilation and Simulation Framework for CGRA". In: *Proceedings of the Fifth Workshop on Open-Source EDA Technology (WOSET)*. 2022.

[15] D. Walter et al. "Mapping and Execution of Nested Loops on Processor Arrays: CGRAs vs. TCPAs". In: *The Computing Research Repository (CoRR)* (Feb. 17, 2025). DOI: 10.48550/arXiv.2502.12062.

Control overhead in loops

Polybench kernels

- Collection of benchmarks containing static control parts
- Various kernels and solvers from linear algebra, data mining algorithms, stencil codes, and others

Benchmark
Dataflow Operations
Executed Instructions
Control Overhead

trmm	trisolv	syrk	syr2k	symm	seidel-2d	mvt
9 648 000	160 400	17 308 800	69 177 600	24 288 000	142 563 600	640 000
43 465 540	643 217	69 795 026	196 476 513	77 425 348	262 561 052	2 724 024
77.80 %	75.06 %	75.20 %	64.79 %	68.63 %	45.70 %	76.51 %

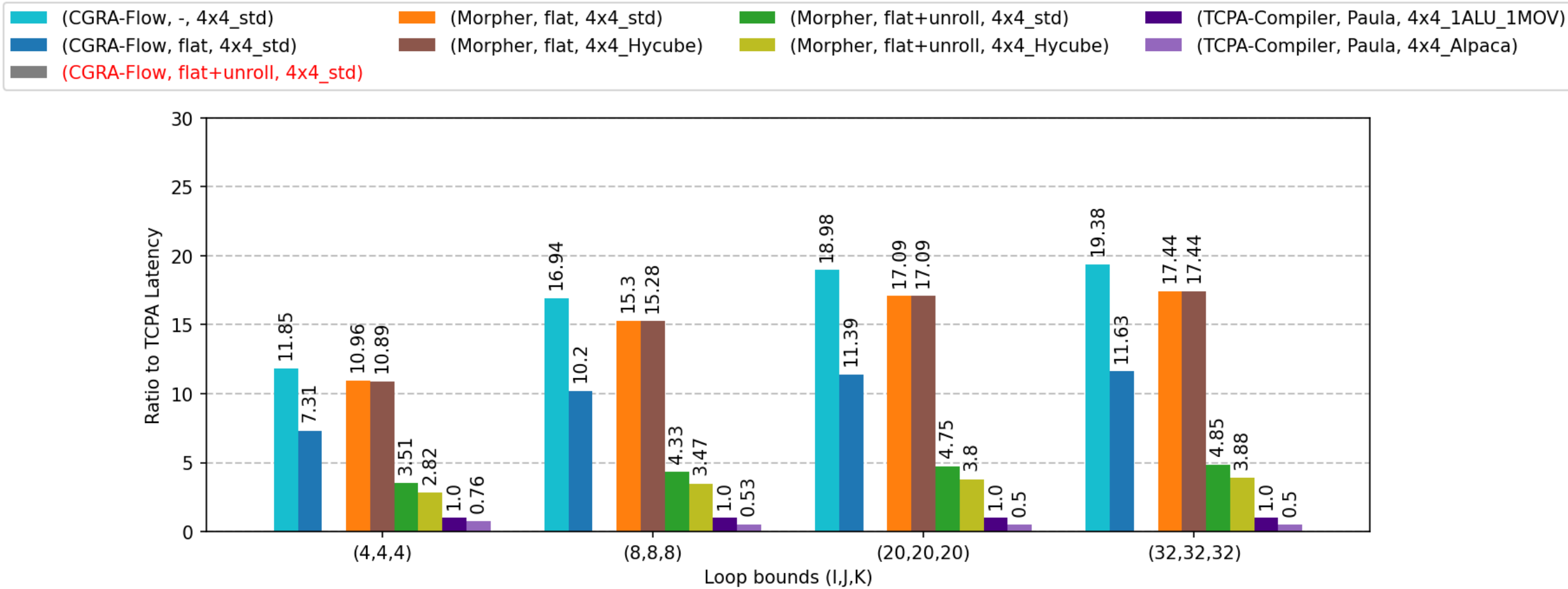
ludcmp	lu	jacobi-2d	jacobi-1d	gramschmidt	gesummv	gemver
64 400 400	32 080 000	37 200 600	159 600	46 248 000	250 750	1 600 400
109 239 230	213 417 020	102 052 753	419 420	109 437 634	1 003 269	5 289 238
41.05 %	84.97 %	63.55 %	61.95 %	57.74 %	75.01 %	69.74 %

gemm	floyd-warshall	fdtd-2d	dynprog	durbin	doitgen	covariance
31 724 000	500 000 000	52 824 000	4 967 600	562 000	14 620 000	30 130 102
95 393 625	1 251 503 521	182 096 857	23 257 050	1 452 037	59 078 389	68 659 335
66.74 %	60.05 %	70.99 %	78.64 %	61.30 %	75.25 %	56.12 %

correlation	cholesky	bigc	atax	adi	3mm	2mm
21 736 080	32 241 200	640 421	640 400	72 040 000	45 711 900	36 667 800
61 389 731	85 656 422	2 562 114	2 721 501	227 240 349	183 299 647	132 684 516
64.59 %	62.36 %	75.00 %	76.47 %	68.30 %	75.06 %	72.36 %

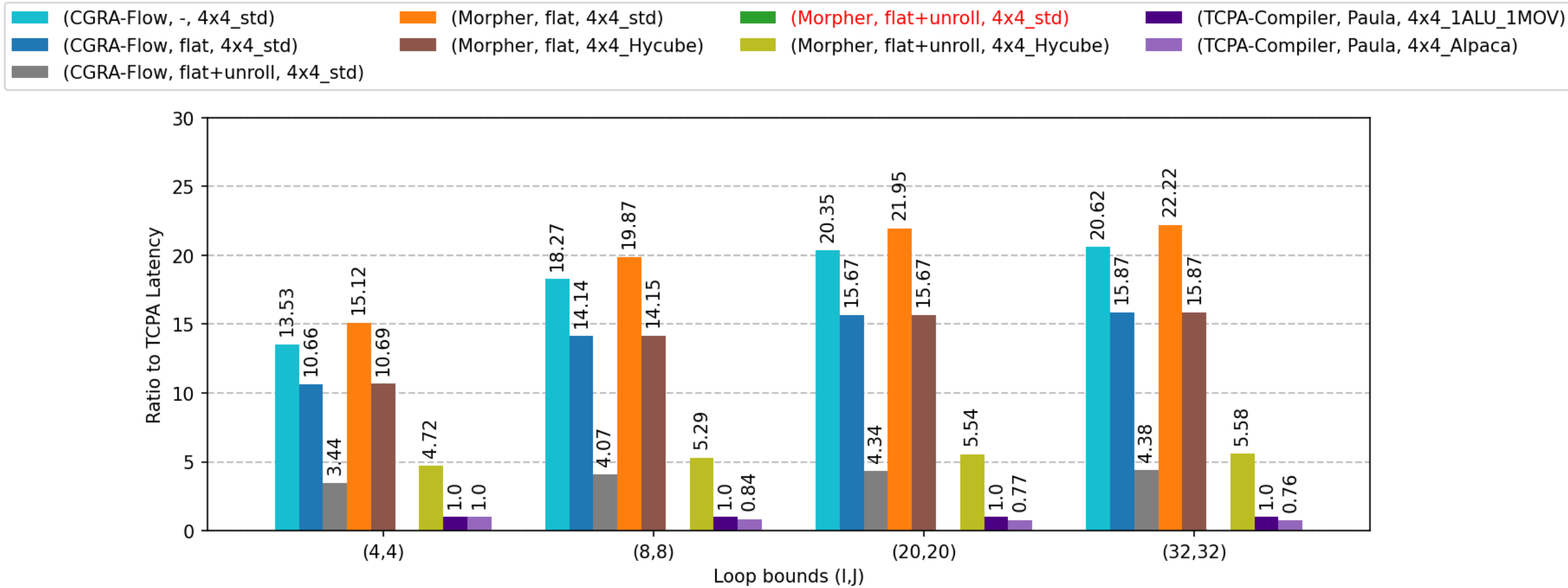
Comparing latency of GEMM

$C = C + A \cdot B$



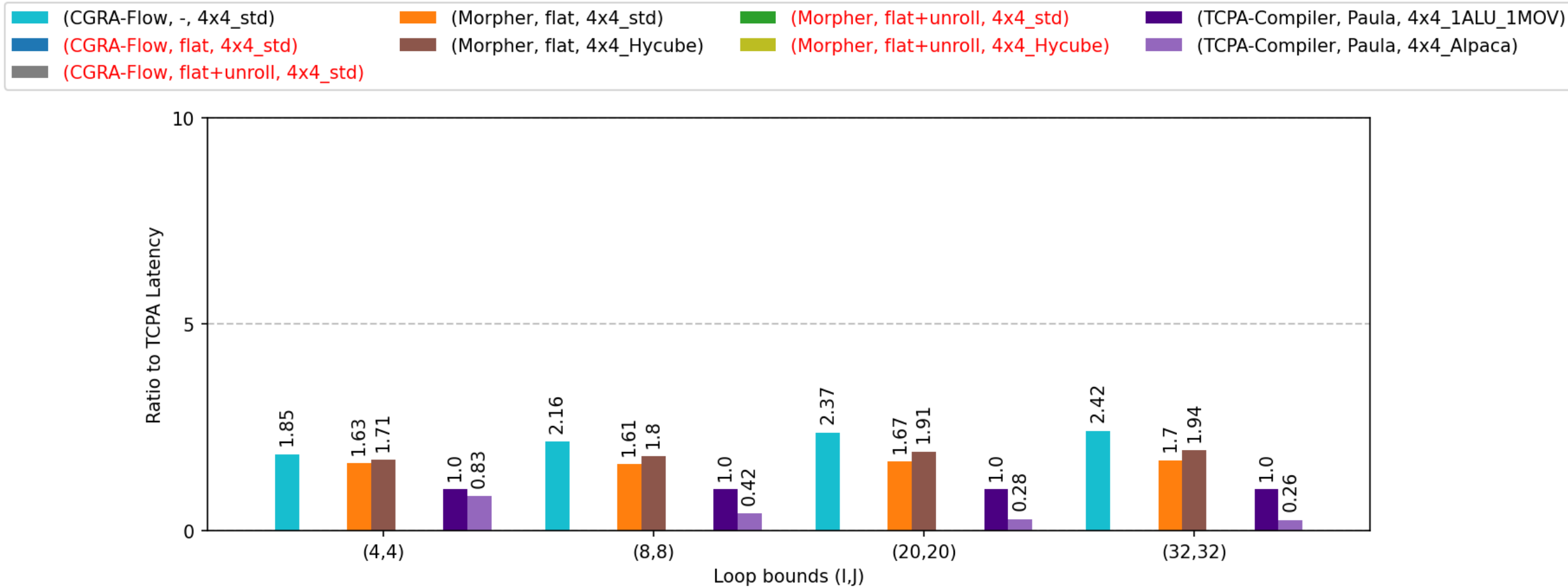
Comparing latency of ATAX

$y = A^T \cdot (A \cdot x)$



Comparing latency of TRISOLV

$$x_i = \frac{b_i - \sum_{j=0}^{i-1} L_{i,j} \cdot x_j}{L_{i,i}}$$



How to Solve Big Problems?

Example Block LU decomposition

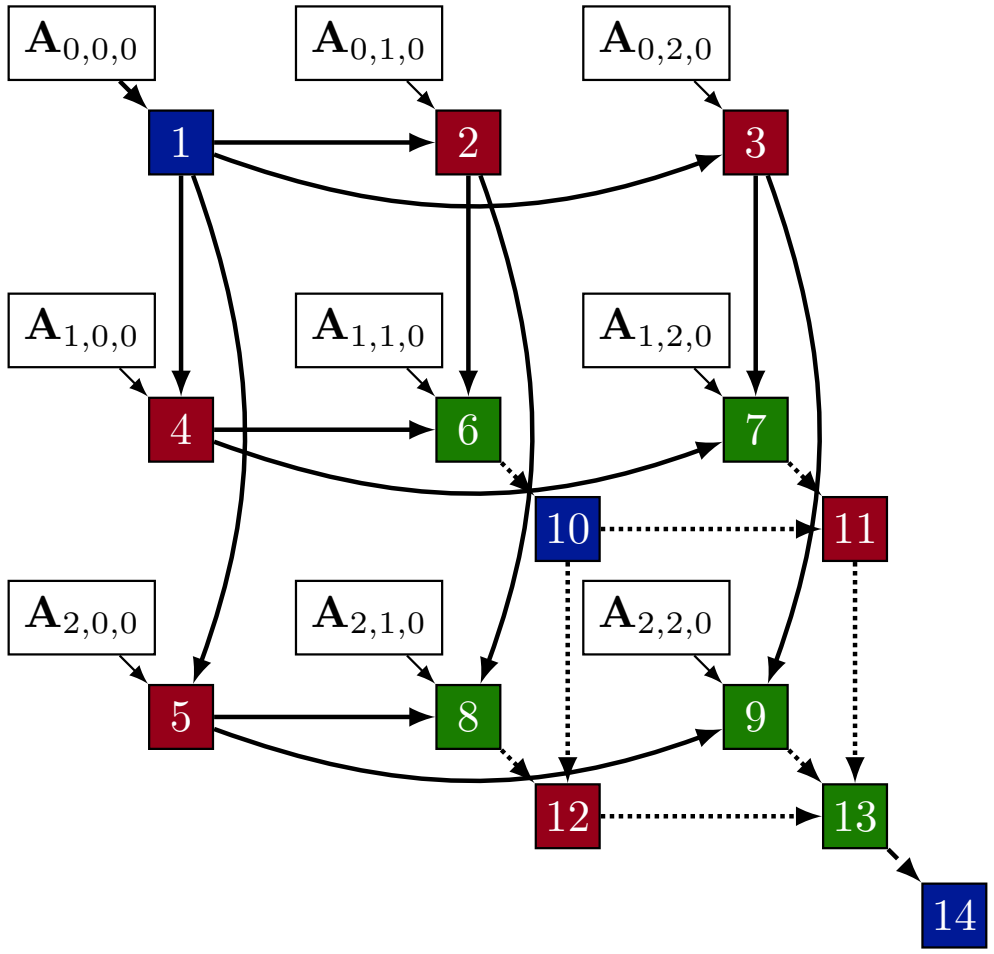
$$\begin{pmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} & \mathbf{A}_{0,2} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,0} & \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{0,0} & 0 & 0 \\ \mathbf{L}_{1,0} & \mathbf{L}_{1,1} & 0 \\ \mathbf{L}_{2,0} & \mathbf{L}_{2,1} & \mathbf{L}_{2,2} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{0,0} & \mathbf{U}_{0,1} & \mathbf{U}_{0,2} \\ 0 & \mathbf{U}_{1,1} & \mathbf{U}_{1,2} \\ 0 & 0 & \mathbf{U}_{2,2} \end{pmatrix}$$

An $(nN \times nN)$ matrix $\mathbf{A}$ can be decomposed into $N \times N$ blocks of size $n \times n$ each and the rhs result matrices solved iteratively for $i = 0, \cdots, N-1$:

1. LU: Obtain $\mathbf{L}_{i,i}, \mathbf{U}_{i,i}$ by decomposing
$$\mathbf{A}_{i,i,i} = \mathbf{L}_{i,i} \cdot \mathbf{U}_{i,i}$$
2. TRSM: Obtain $\forall j > i : \mathbf{L}_{j,i}, \mathbf{U}_{i,j}$, by solving
$$\mathbf{A}_{j,i,i} = \mathbf{L}_{j,i} \cdot \mathbf{U}_{i,i}, \quad \mathbf{A}_{i,j,i} = \mathbf{L}_{i,i} \cdot \mathbf{U}_{i,j}$$
3. GEMM: Obtain $\forall j, l > i : \mathbf{A}_{j,l,i+1}$ by computing
$$\mathbf{A}_{j,l,i+1} = \mathbf{A}_{j,l,i} - \mathbf{L}_{j,i} \cdot \mathbf{U}_{i,l}$$

Background

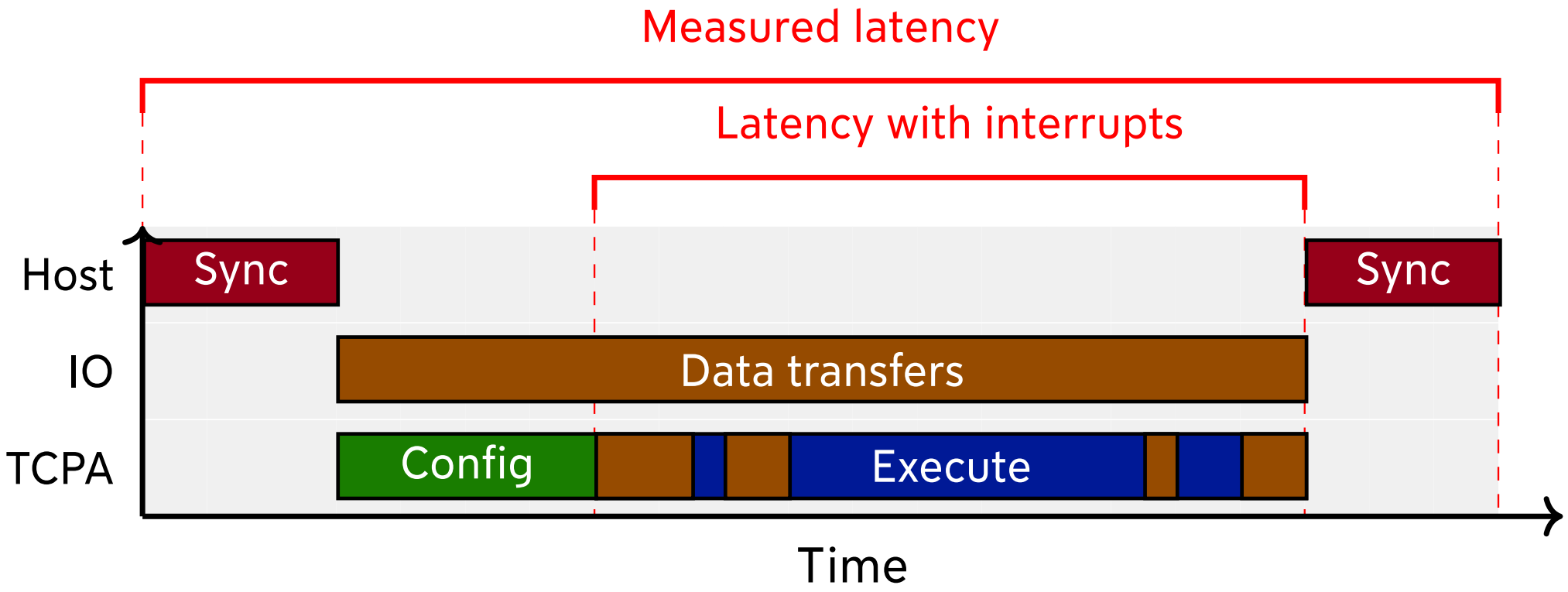
LU kernel dependencies and a block schedule



1	LU
2	TRSM
3	TRSM
4	TRSM
5	TRSM
6	GEMM
7	GEMM
8	GEMM
9	GEMM
10	LU
11	TRSM
12	TRSM
13	GEMM
14	LU

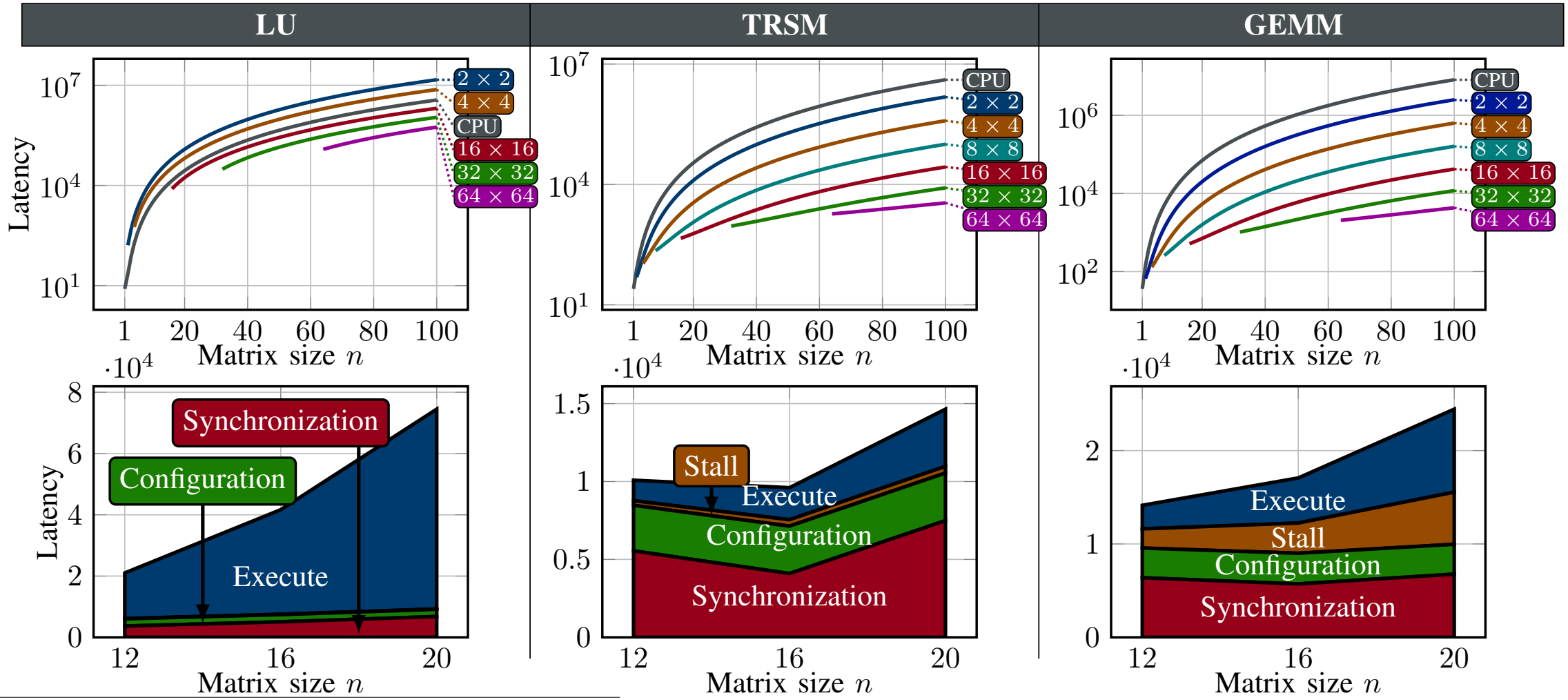
Kernel Execution

Host offloads a loop to a TCPA (prototyped on FPGA)



Kernel Evaluation

Anal. and measd. latencies[16] on FPGA xc7z100ffg900-1



[16] D. Walter, T. Adamtschuk, F. Hannig, and J. Teich. "Analysis and Optimization of Block LU Decomposition for Execution on Tightly Coupled Processor Arrays". In: *Proceedings of the 35th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)* (Hong Kong). IEEE, July 24–26, 2024, pp. 97–106. DOI: 10.1109/ASAP61560.2024.00029.

Kernel Scheduling

Reuse schemes

- Synchronization reuse
  - Intermediate synchronizations can be totally avoided if all kernels are scheduled on the TCPA
- Configuration reuse
  - Reconfiguration is not necessary if the same kernel is executed multiple times
  - Configure once and reuse the configuration for a sequence of equal kernel type executions
- Data reuse
  - Keep output data in the TCPA's I/O buffer and reuse it for the next data-dependent kernel invocation
  - Load input data only once and reuse it for successive kernels sharing the same data dependency

$\rightarrow$Can a clever execution order of kernels exploit these reuse schemes to reduce the overhead?

Kernel Scheduling

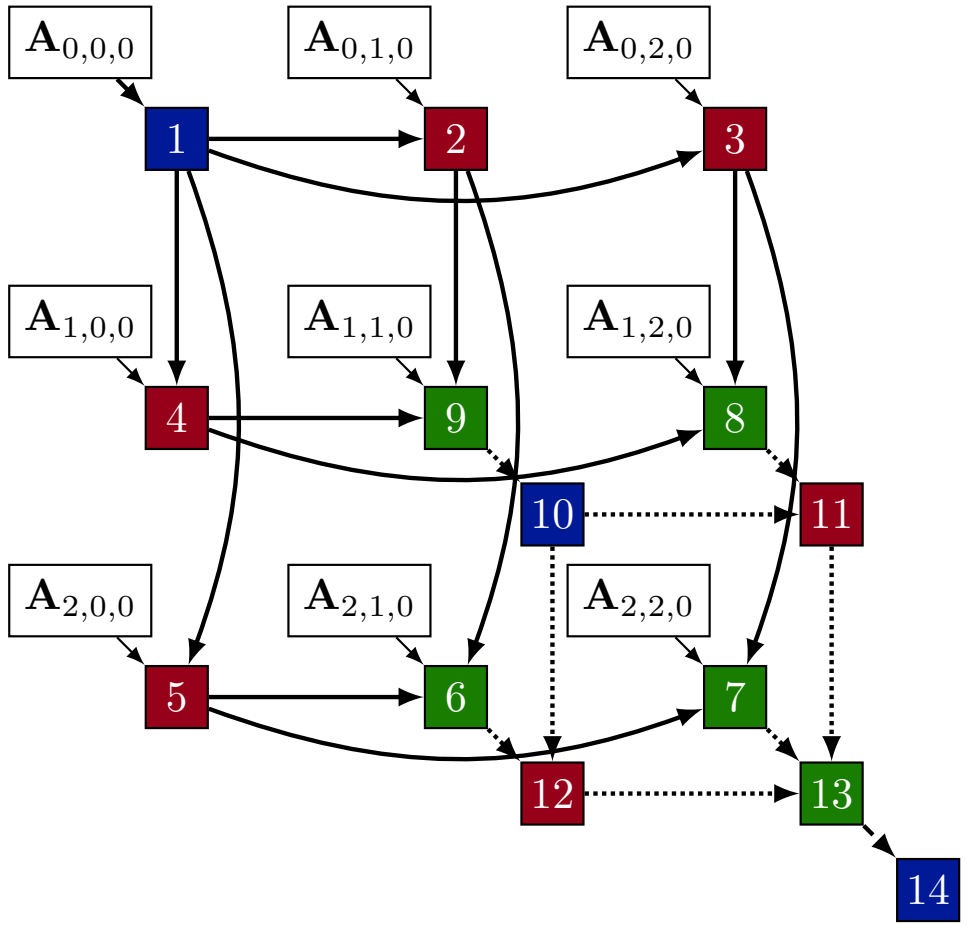
Right-looking schedule

- Exploit long chains of TRSM and GEMMs to reuse configurations
- Zic-zac kernel execution reduces I/O
- #Configurations:

$$3N - 2$$

- #I/O Operations:

$$(N^3 + \frac{1}{2}N^2 - \frac{1}{2}N + 1) \cdot n^2$$



1	LU
2	TRSM
3	TRSM
4	TRSM
5	TRSM
6	GEMM
7	GEMM
8	GEMM
9	GEMM
10	LU
11	TRSM
12	TRSM
13	GEMM
14	LU

Kernel Scheduling

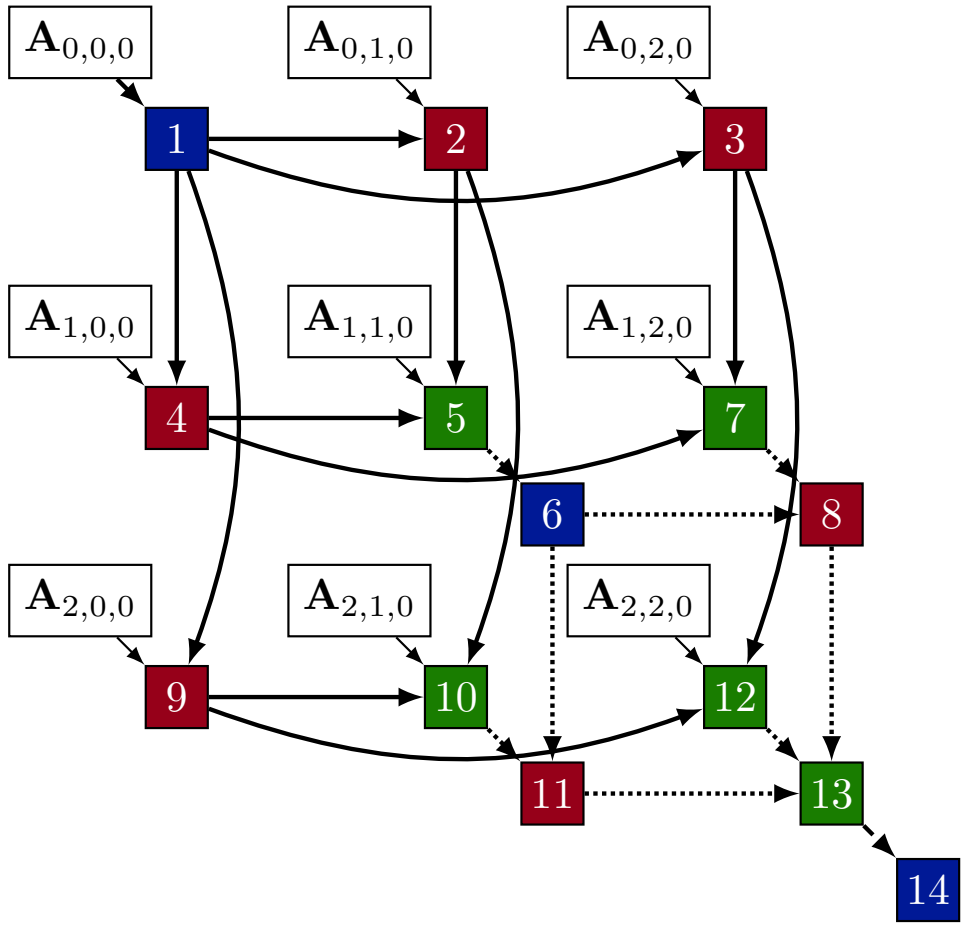
Left-looking schedule

- Frequent change of kernel requires more reconfigurations
- GEMM chains on the same block reduce I/O
- #Configurations:

$$2N^2 - 2N + 1$$

- #I/O Operations:

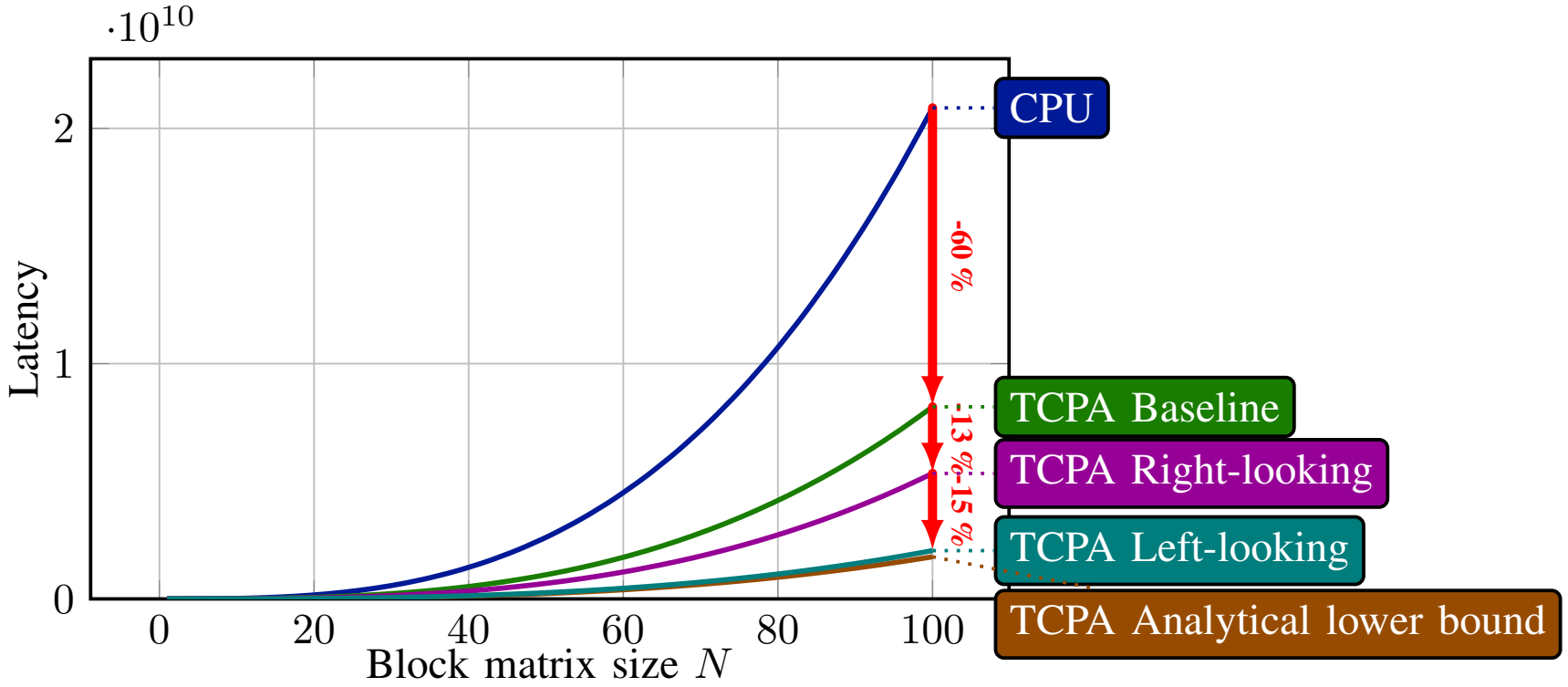
$$\left(\frac{2}{3}N^3 + 2N^2 - \frac{2}{3}N\right) \cdot n^2$$



1	LU
2	TRSM
3	TRSM
4	TRSM
5	TRSM
6	GEMM
7	GEMM
8	GEMM
9	GEMM
10	LU
11	TRSM
12	TRSM
13	GEMM
14	LU

Schedule Evaluation

Latencies for $n = 20$ over N on 4×4 TCPA on FPGA [17]



[17] D. Walter, T. Adamtschuk, F. Hannig, and J. Teich. "Analysis and Optimization of Block LU Decomposition for Execution on Tightly Coupled Processor Arrays". In: *Proceedings of the 35th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)* (Hong Kong). IEEE, July 24–26, 2024, pp. 97–106. DOI: 10.1109/ASAP61560.2024.00029.

Summary

LU decomposition on Tightly Coupled Processor Arrays

- Successful mapping of a block LU decomposition including three loop kernels to TCPAs:
  - Flat LU Decomposition
  - Triangular Solve (TRSM)
  - General Matrix Multiplication (GEMM)
- Identified significant overhead in individual kernel calls
  - GEMM: Over 50 % overhead observed
- Presented different kernel schedules for reuse of: Configuration, data and synchronization
  - Right-looking approach: minimal reconfigurations, 25 % I/O savings
  - Left-looking approach: reduced reconfigurations, 50 % I/O savings
- Left-looking implementation approaches analytical optimum
  - Performance gain: 10x speedup in latency over CPU baseline
- Future work:
  - Methods to determine optimal block sizes
  - Schedules utilizing clusters of smaller TCPAs
  - Symbolic energy analysis of loop schedules

Let's do a Chip!

ALPACA: Asic LooP ACcelerAtor

a TCPA chip design

ALPACA: The chip design

Motivation

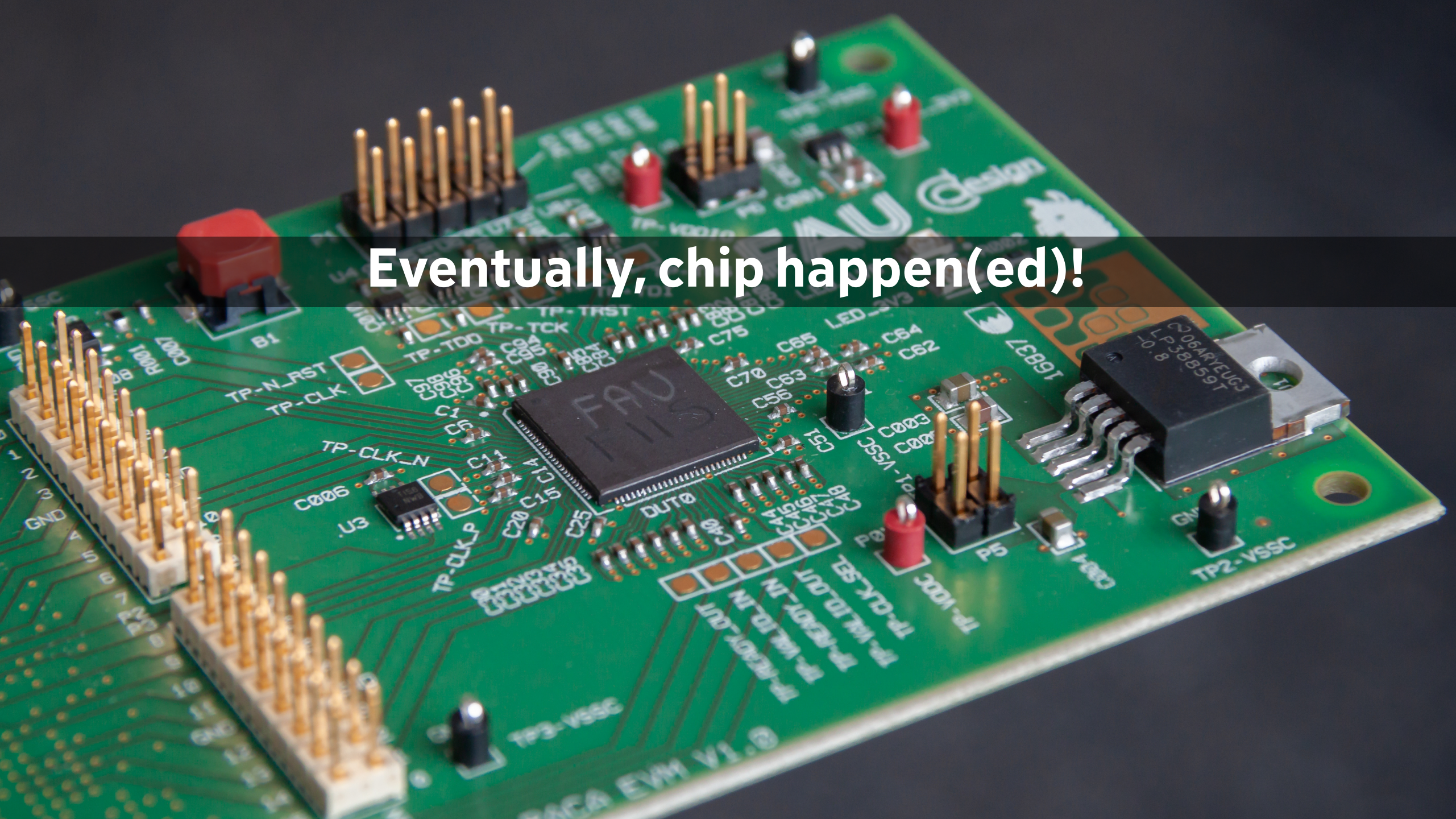
- Build a large TCPA instance in hardware, i.e., as ASIC
- Achieve orders of magnitude higher performance compared against simulation and FPGA-based prototyping
- Possibility of integrating chip as accelerator
- Assess power and energy efficiency
- Advance skills in chip design

Sounds like valuable goals and fun ;)

Success probability of a first-time-right chip

- 100 % correct RTL description
- 100 % correct netlist
- 100 % correct implemented design (P & R)
- 100 % correct manufacturing
- 100 % correct packaging (i.e., bonding)
- 100 % correct design of printed circuit board (PCB)

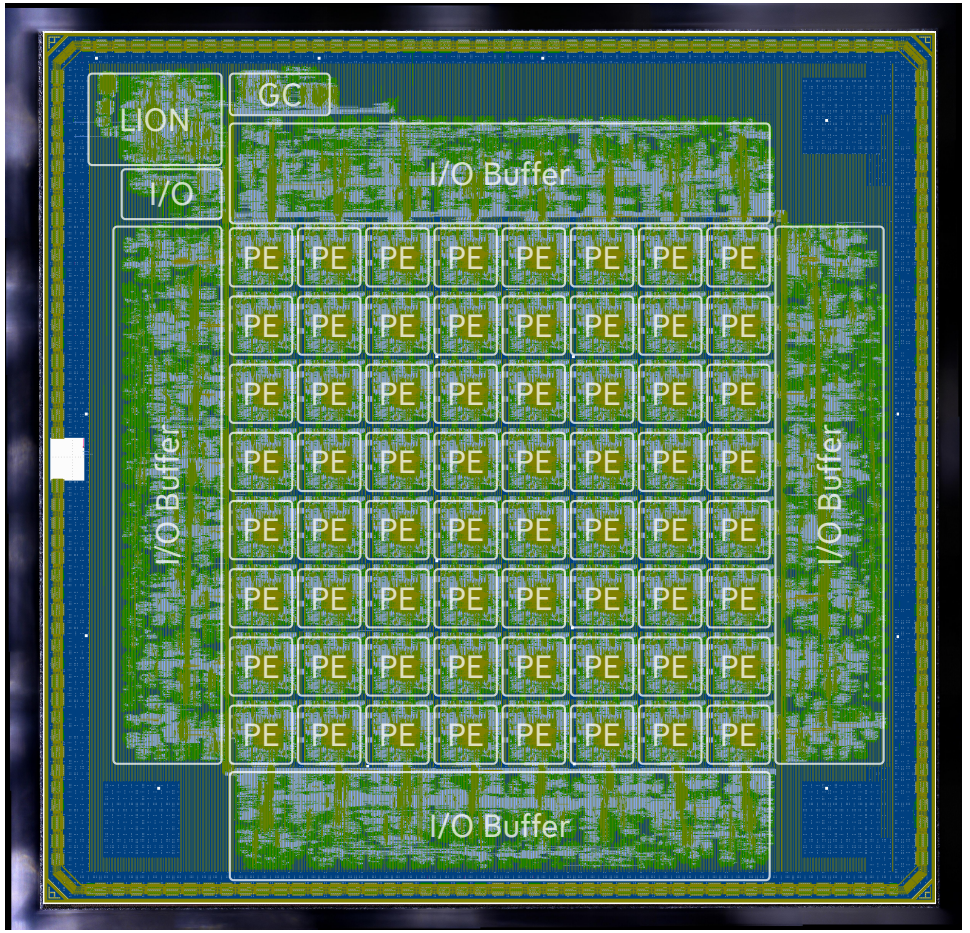
If each step has a probability of only 10 % of producing an error, assuming independence, the probability of overall success drops already to 50 %


Eventually, chip happen(ed)!

Specification

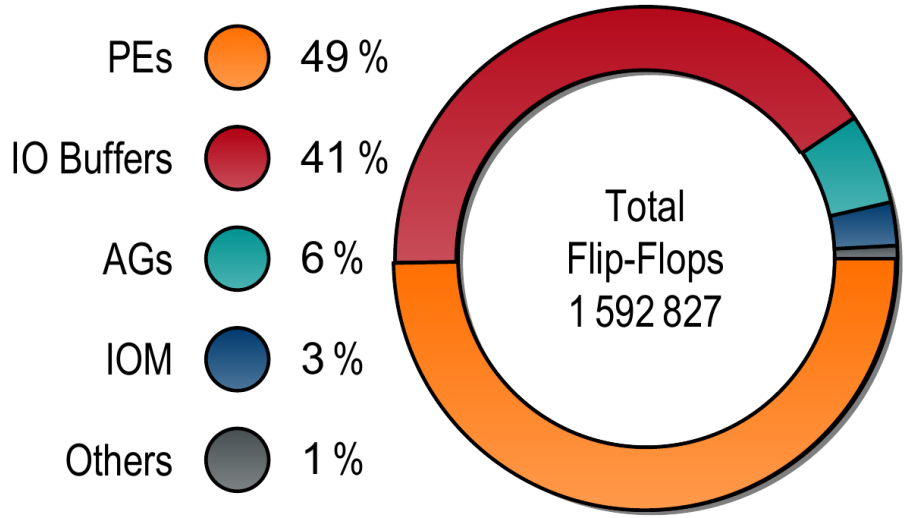
10 mm^2 chip in 22 nm technology[10]

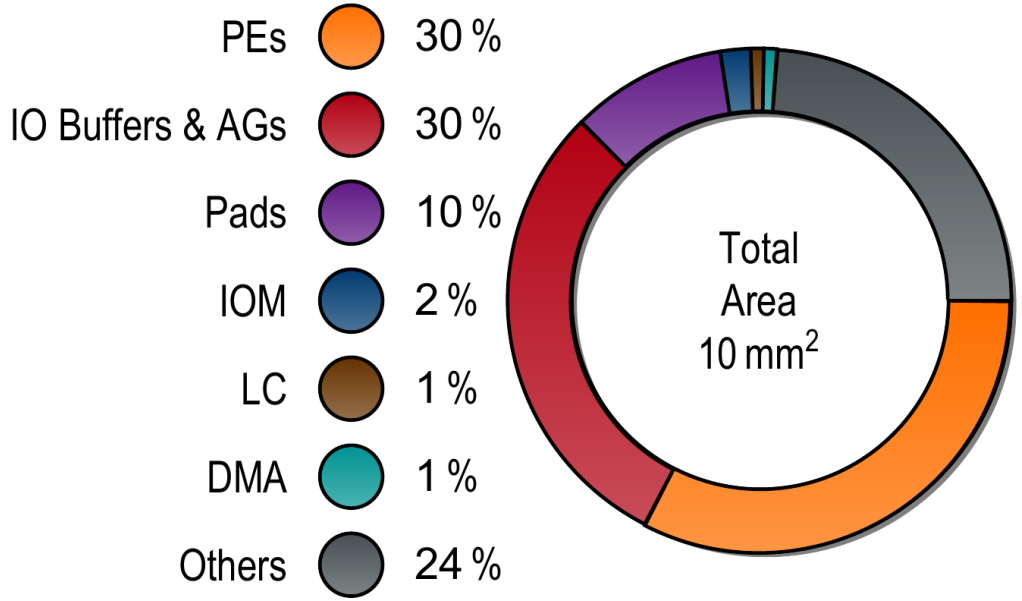
- Technology
  - 22 nm FDSOI process
  - 3.200×3.125 mm chip area
  - 6.157 million standard cells
  - >90 million transistors
  - 500 MHz core frequency
  - 178 I/O pads
- 8×8 TCPA
  - 64 processing elements
    - 192 floating-point vector units
    - 384 GFLOPS
  - 4 I/O Buffers
    - 128 memory banks (à 512 bytes)
    - 64 address generators
  - 16-bit parallel data interface
  - JTAG interface for testing and debugging



[10] D. Walter, M. Brand, C. Heidorn, M. Witterauf, F. Hannig, and J. Teich. "ALPACA: An Accelerator Chip for Nested Loop Programs". In: *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)* (Singapore). IEEE, May 19–22, 2024.

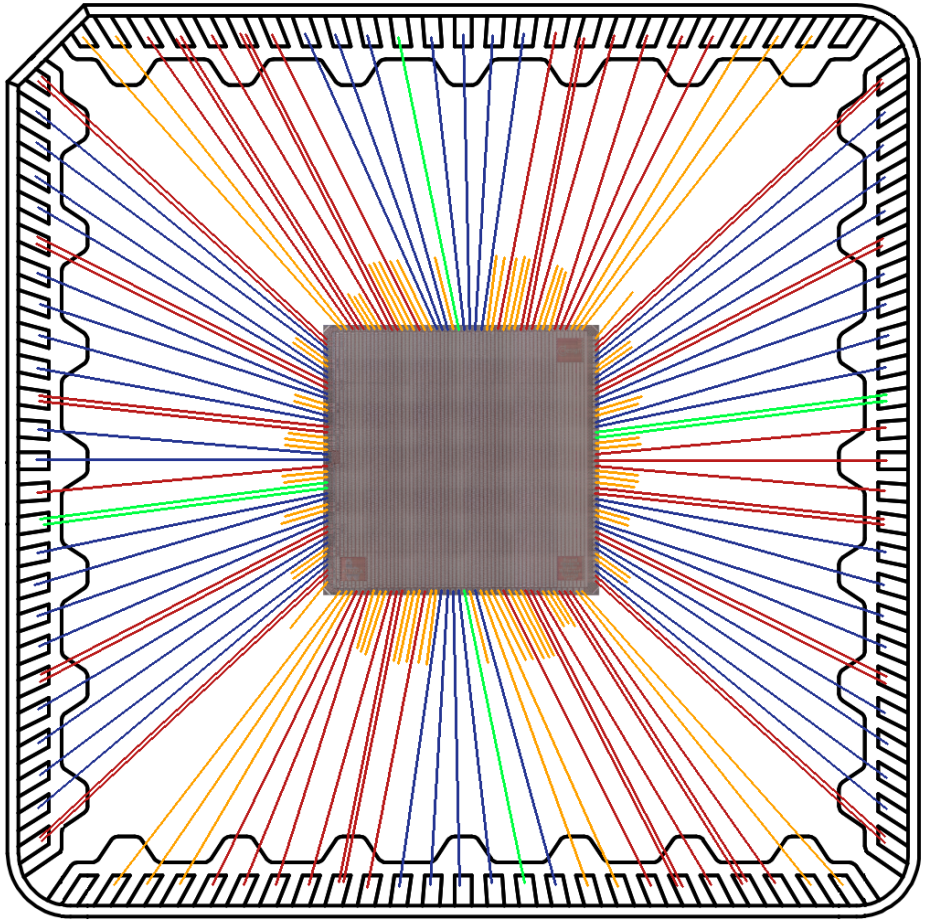
Breakdown of flip-flops and chip area





Chip package

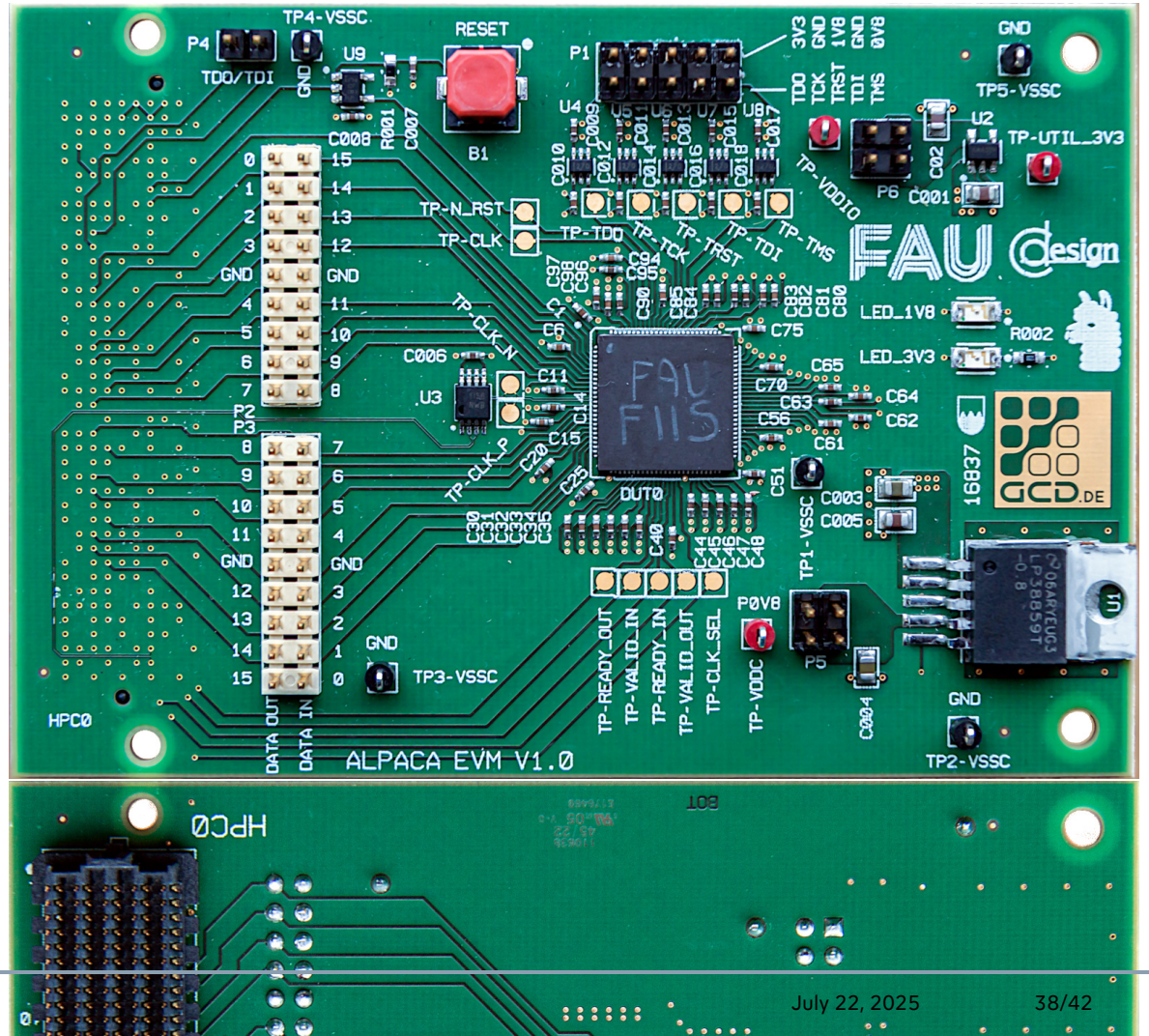
- 12×12 QFN with 100 pins
- 98 pins in use
  - 36 core power (0.8 V)
  - 12 core ground
  - 4 I/O power (1.8 V)
  - 2 differential I/O (500 MHz, 0.8V)
  - 44 standard I/O (100 MHz, 1.8V)
- Common ground plate
- Length of bond wires: 3–4 mm



FMC test board

(FMC = FPGA Mezzanine Card)

- Power supply
  - 3.3 V to 0.8 V
  - 3.3 V to 1.8 V
- Parallel data interface
  - 16-bit input and 16-bit output pins
  - Supports backpressure in both directions via valid and ready signals
  - Single-ended I/O clock
- JTAG interface
  - 4 input and 1 output pins
  - Dedicated debug functionality
  - Allows direct access to all I/O buffers and PEs
- Clock converter (LVDS to CML)
- Six-layer PCB



Flexibility/performance tradeoffs

w.r.t. recent chips with comparable technology and area

Flexibility ⟵ ⟶ Performance

Parameters	Conti et al.[11]	Wang et al.[12]	Feng et al.[13]	Our Work[14]	Du et al.[15]	Moon et al.[16]	Huang et al.[17]
Technology	22 nm	40 nm	16 nm	22 nm	28 nm	28 nm	22 nm
Die Area	18.7 mm^2	4.7 mm^2	20.1 mm^2	10 mm^2	9.08 mm^2	9.61 mm^2	30.6 mm^2
Applications	General Software	DFGs	DFGs	Loop Nests	Deep Learning	Deep Learning	CNN
Cores	16 RISC-V	16 PEs	384 PEs	64 PEs	1024 PEs	32 PEs	N/A
INT Support	2–32	N/A	16	32	8	1–8	1–8
FXP Support	N/A	32	N/A	N/A	N/A	N/A	N/A
FP Support	16, 32	N/A	16	8, 32	N/A	N/A	N/A
Supply Voltage	0.5–0.8 V	0.8–1.1 V	0.84–1.29 V	0.6–1.2 V	0.66–1.3 V	0.46–1.13 V	0.7–0.8 V
Frequency	420 MHz	753 MHz	955 MHz	700 MHz	500 MHz	400 MHz	200 MHz
Performance	6.9 GFLOPS	5.38 GOPS	367 GOPS	537.60 GFLOPS	512 GOPS	18.9 TOPS	24.88 TOPS
Energy Efficiency	207 GFLOPS/W	26.4 GOPS/W	538 GOPS/W	270 GFLOPS/W	11.2 TOPS/W	127.8 TOPS/W	251 TOPS/W

[11] F. Conti et al. "A 12.4TOPS/W @ 136GOPS AI-IoT System-on-Chip with 16 RISC-V, 2-to-8b Precision-Scalable DNN Acceleration and 30%-Boost Adaptive Body Biasing". In: *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, Feb. 19–23, 2023, pp. 326–327. DOI: 10.1109/ISSCC42615.2023.10067643.

[12] B. Wang, M. Karunarathne, A. K. Mohite, T. Mitra, and L. Peh. "HyCUBE: A 0.9V 26.4 MOPS/mW, 290 pJ/op, Power Efficient Accelerator for IoT Applications". In: *Proceedings of the IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, Nov. 4–6, 2019, pp. 133–136. DOI: 10.1109/A-SSCC47793.2019.9056954.

[13] K. Feng et al. "Amber: Coarse-Grained Reconfigurable Array-Based SoC for Dense Linear Algebra Acceleration". In: *Proceedings of the IEEE Hot Chips 34 Symposium (HCS)*. IEEE, Aug. 21–23, 2022, pp. 1–30. DOI: 10.1109/HCS55958.2022.9895616.

[14] D. Walter, M. Brand, C. Heidorn, M. Witterauf, F. Hannig, and J. Teich. "ALPACA: An Accelerator Chip for Nested Loop Programs". In: *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)* (Singapore). IEEE, May 19–22, 2024.

[15] C. Du et al. "A 28nm 11.2TOPS/W Hardware-Utilization-Aware Neural-Network Accelerator with Dynamic Dataflow". In: *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, Feb. 19–23, 2023, pp. 332–333. DOI: 10.1109/ISSCC42615.2023.10067774.

[16] S. Moon, H. Mun, H. Son, and J. Sim. "A 127.8TOPS/W Arbitrarily Quantized 1-to-8b Scalable-Precision Accelerator for General-Purpose Deep Learning with Reduction of Storage, Logic and Latency Waste". In: *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, Feb. 19–23, 2023, pp. 330–331. DOI: 10.1109/ISSCC42615.2023.10067615.

[17] W. Huang et al. "A Nonvolatile AI-Edge Processor with 4MB SLC-MLC Hybrid-Mode ReRAM Compute-in-Memory Macro and 51.4-251TOPS/W". In: *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, Feb. 19–23, 2023, pp. 258–259. DOI: 10.1109/ISSCC42615.2023.10067610.

(Some) lessons learned

- Took twice as long as originally planned
- Having a great team of (test) application developers, compiler experts, front-end and back-end designers
- Synthesizing and testing each macro after place & route
- Keep all software, tools, and the PDK up-to-date!

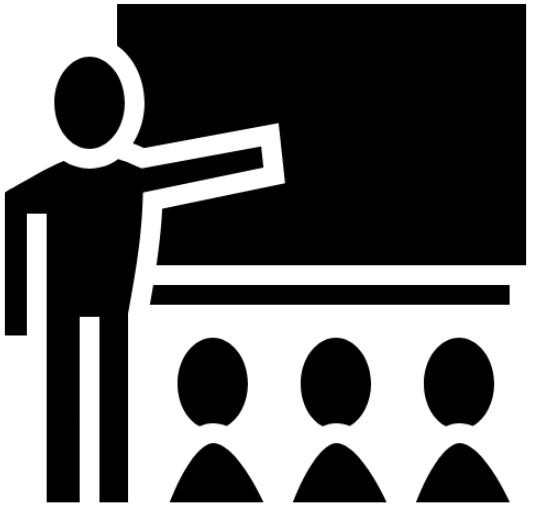


Image by Delapouite under CC BY 3.0 (https://game-icons.net/1x1/delapouite/teacher.html)

Acknowledgments and contact

Acknowledgments

Contributors (in alphabetical order):

Thomas Adamtschuk, Marcel Brand, Marita Halm, Frank Hannig, Christian Heidorn, Jürgen Teich, Dominik Walter, Michael Witterauf

Funding:

This work was partly supported by the German Research Foundation (DFG) – Project number 146371743 – as part of the Transregional Collaborative Research Centre “Invasive Computing” (SFB/TR 89).

Contact

Jürgen Teich

Email: juergen.teich@fau.de
Web: www.cs12.tf.fau.de
Phone: +49 9131 85-25150

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl für Informatik 12
(Hardware-Software-Co-Design)
Cauerstr. 11
91058 Erlangen


Alpaca_in_Action Video Demo on FAU TV:
https://www.youtube.com/watch?v=ys8muYt7drI
ALPACA Output
ROHDE&SCHWARZ
HMP4040
12.000 V
1.068 A
0.800 V
12.82 W
1.693 A
0.000 V
1.35 W
1.000 A
0.000 V
0.0 mW
1.000 A
0.0 mW
Ch1
Ch2
Ch3
Ch4