SO(DA)^2: Software Defined Architectures for Data Analytics

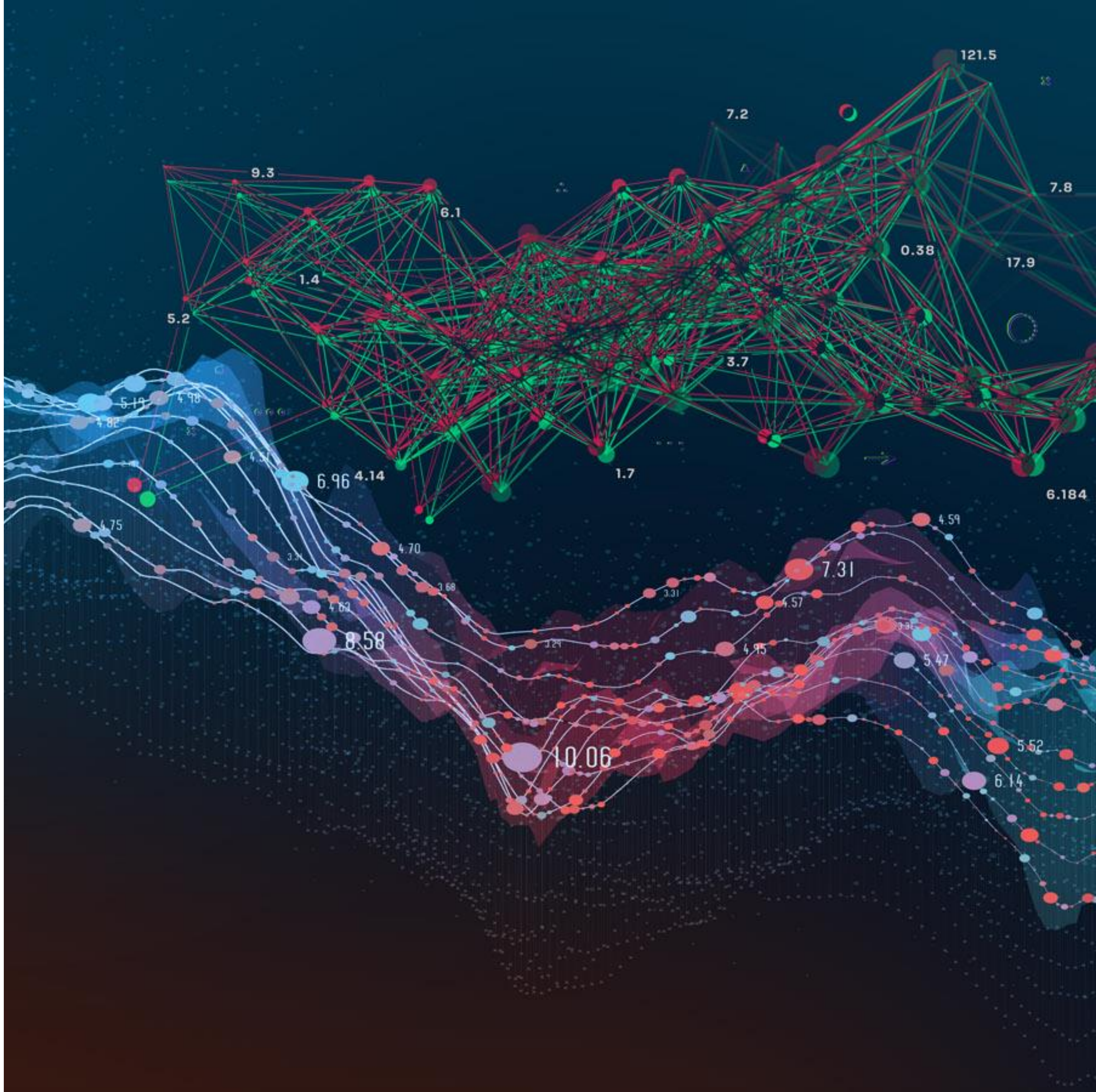
OSCAR 2023 Workshop (co-located with ISCA)
June 20, 2023

Cheng Tan
Microsoft

Nicolas Bohm Agostini, **Ankur Limaye**, Marco Minutoli, Vito Giovanni Castellana, Ang Li, Antonino Tumeo
Pacific Northwest National Laboratory

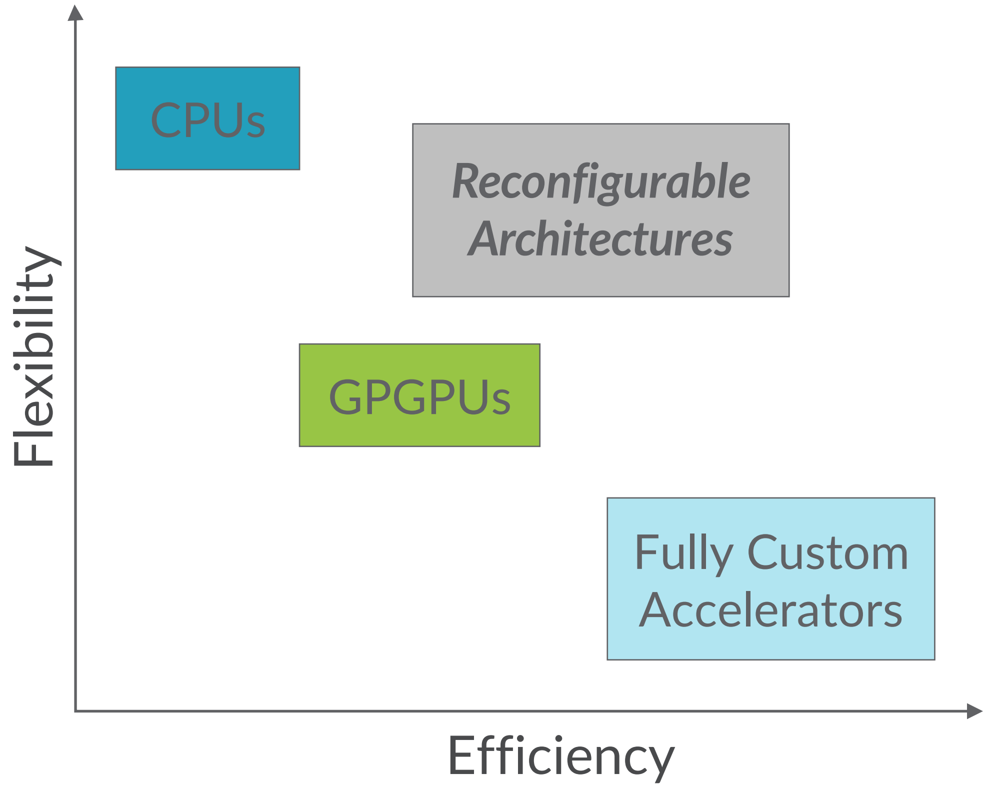
Serena Curzel
Politecnico di Milano

PNNL is operated by Battelle for the U.S. Department of Energy

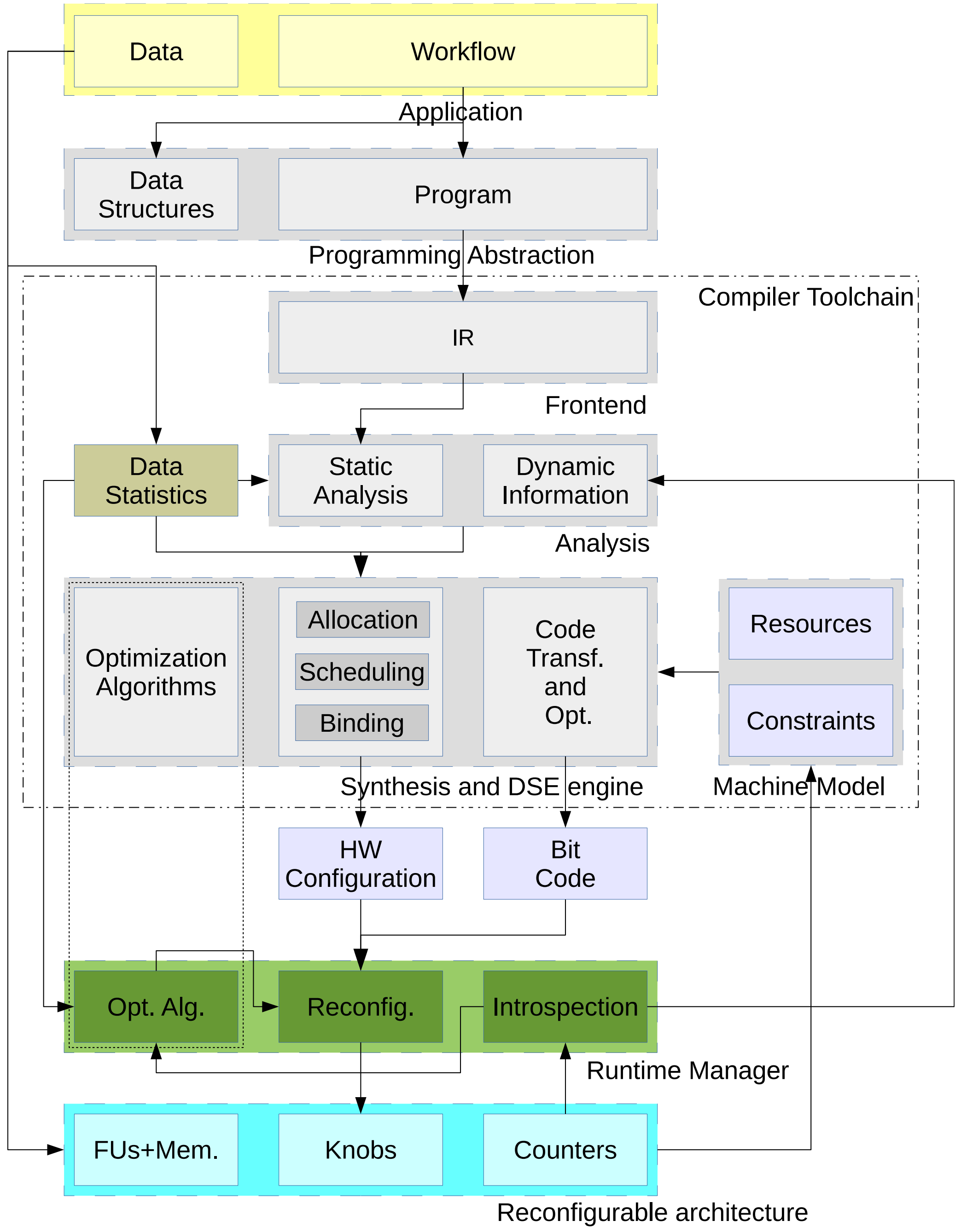


Motivation

- Modern data science applications:
  - Complex mix of algorithms with diverse behaviors
  - Performance highly dependent on the volume, the velocity, and the structure of the data
- Specialized architectures to improve efficiency and low decision latency, often integrate:
  - Domain-specific accelerators
  - Optimized memory interfaces & on-chip networks
- Reconfigurable architectures
  - Provide efficiency through adaptation without trading off flexibility
  - Coarse-grained Reconfigurable Arrays (CGRAs)
  - Functional units and memories interconnected with reconfigurable on-chip networks



SO(DA)^2 Framework Concept Map

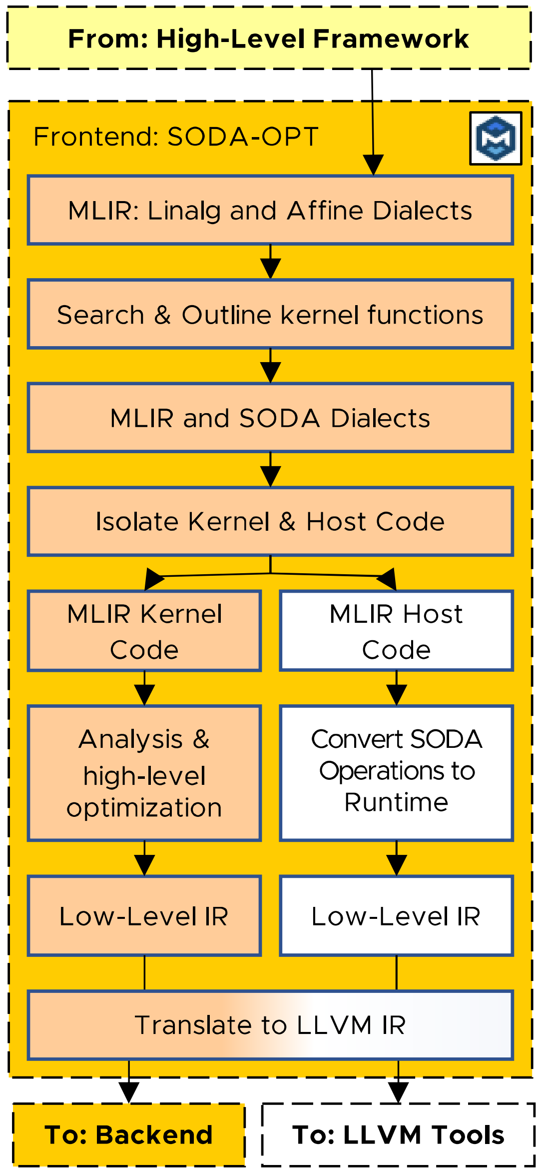


- **High-Level Abstraction and Data-Aware Analysis**
  - Interfaces with high-level programming frameworks
  - Generates high-level intermediate representation (IR)
  - Performs high-level & data-dependent optimizations
- **Design Space Exploration and Synthesis (DSES) Engine**
  - Multi-objective (time, power, area, reuse) optimizations
  - Maps tasks to resources, identifies HW configurations
- **Runtime Manager**
  - Schedules and maps configurations and compiled codes
  - Run-time monitoring & reconfiguration; feedback to DSES
- **Reconfigurable Architecture**
  - Forward looking target: CGRAs
  - Can exploit FPGAs for prototyping
- **Software Components**
  - SODA-OPT: *open-source* MLIR frontend and high-level IR
  - OpenCGRA: *open-source* CGRA generator

[V. G. Castellana, *et al.*, "Software-defined Architectures for Data Analytics," *ASP-DAC 2019*]

SODA-OPT: Frontend and High-Level IR

- SODA-OPT: Search, Outline, Dispatch, Accelerate frontend optimizer generates the SODA High-Level IR
- Employs and embraces the MLIR framework
  - Used in TensorFlow, TFRT, ONNX-MLIR, NPComp, others
  - Several architecture independent dialects (Linalg, Affine, SCF) and optimizations
- Interfaces with high-level ML frameworks leveraging MLIR bridges (e.g., libraries, rewriters)
- Defines the “soda” MLIR dialect and related compiler passes to:
  - Identify dataflow segments for hardware generation
  - Perform high-level optimizations (dataflow transformations, data-level and instruction-level parallelism extraction)
  - Generates interfacing code and runtime calls for the host



SODA-OPT: System Overview

[N. Bohm Agostini, *et al.*, “SODA-OPT: an MLIR-based flow for co-design and high-level synthesis,” *CF 2022* - ***Best Poster Award***]
[N. Bohm Agostini, *et al.*, “An MLIR-based Compiler Flow for System-level Design and Hardware Acceleration,” *ICCAD 2022*]

SODA-OPT: Frontend and High-Level IR

- The SODA-OPT optimization passes:

Structural
- Tiling
- Unrolling

Memory
- Temporary Buffer Allocation
- Alloca Buffer Promotion

Avoid Redundancy and Promote Reuse
- Scalar Replacement of Aggregates
- Early Alias Analysis
- Outlining

Avoid Unnecessary Operations
- Dead Code Elimination
- Common Sub-expression Elimination

Single basic block containing the compute intensive part of the kernel
More freedom to schedule operations

Increased instruction-level parallelism
Schedule independent arithmetic operations on the same cycle when their inputs are available

Increased data-level parallelism
Schedule operations into different memory units on the same cycle

Avoid unnecessary reads from kernel arguments
Reduce expensive accesses to external memory

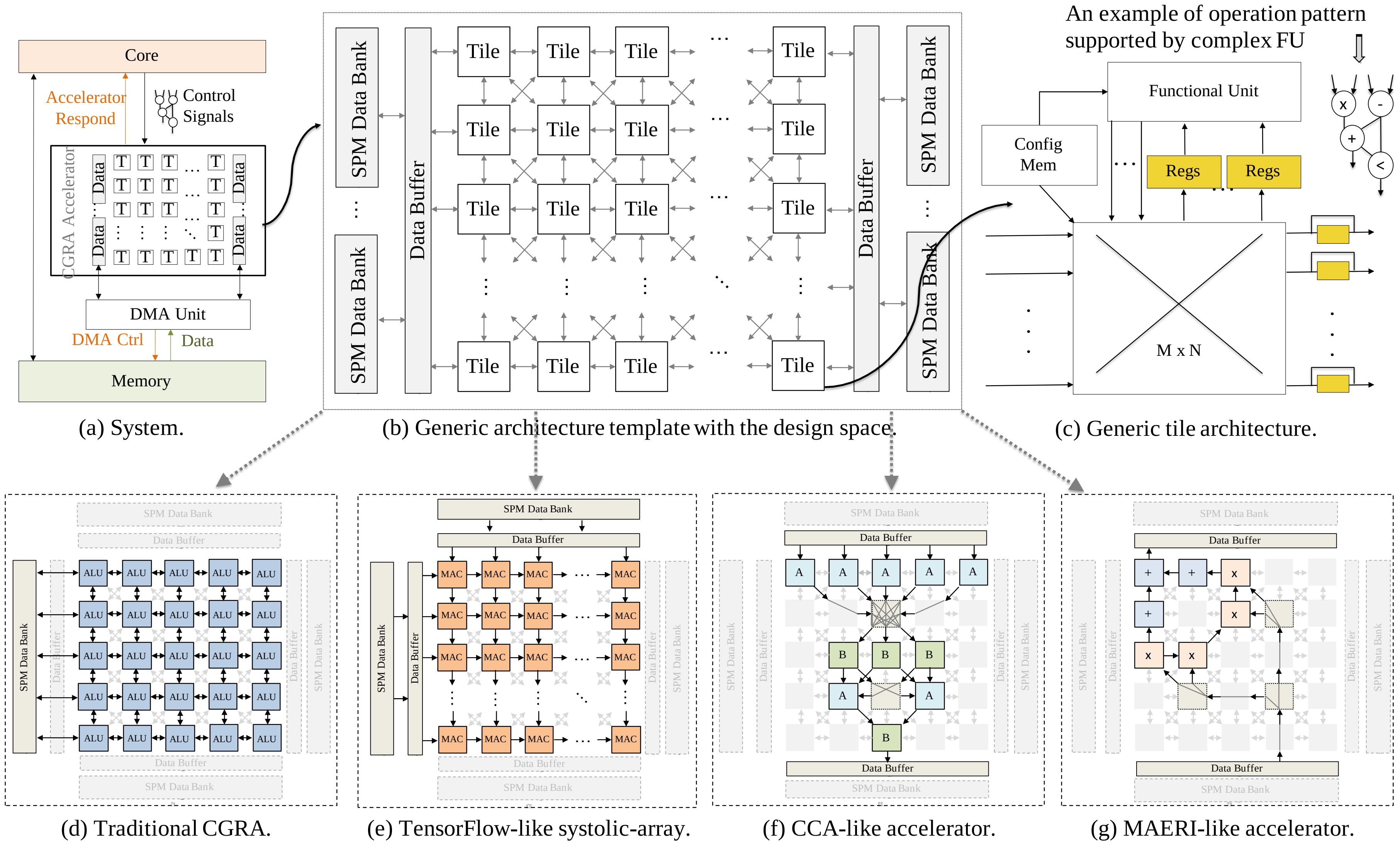
Reuse read results, aggregate on scalars
Save scalar values loaded from memory and intermediate results in registers rather than performing repeated memory accesses

Early alias analysis
Schedule memory operations independently on regions that don't alias

Remove redundant or unnecessary operations
Avoid wasting resources

OpenCGRA: CGRA Generator Overview

- Unified flow for modeling, testing, and evaluating Coarse-Grained Reconfigurable Arrays (CGRAs)
- Flow specializes the functional units of the CGRA tiles given one or more input applications



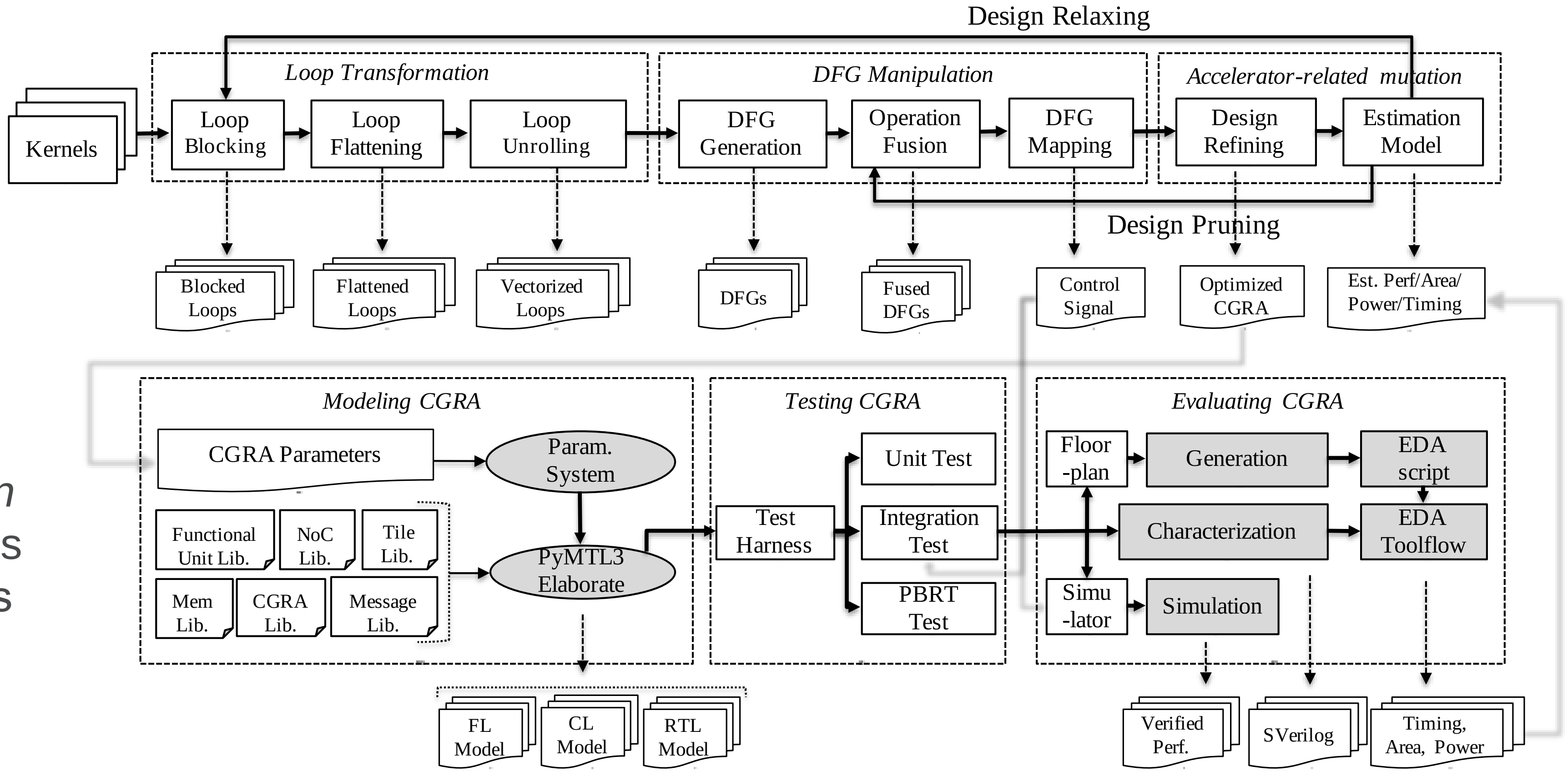
OpenCGRA: Generic architecture template & customizations

[C. Tan, *et al.*, "OpenCGRA: An Open-Source Unified Framework for Modeling, Testing, and Evaluating CGRAs," *ICCD 2020*]

OpenCGRA: CGRA Generator Flow

- Employs PyMTL3 for the generation, synthesis, testing of the Verilog
- Exploits SODA-OPT and LLVM to map the dataflow graph and identify operations (simple/complex)

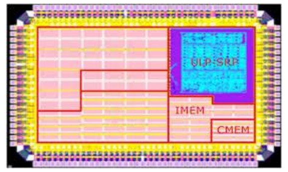
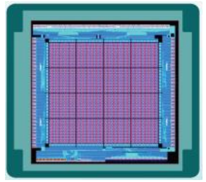
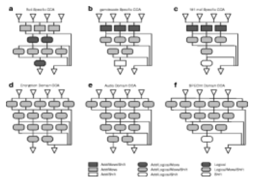
- High level loop transformation
  - Loop blocking/tiling
  - Loop flattening
  - Loop unrolling (depending on the CGRA size)
- Pre-store a set of configurations
  - *For dynamic partial reconfiguration*
  - Limits the number of configurations
  - Only consider the “regular” shapes



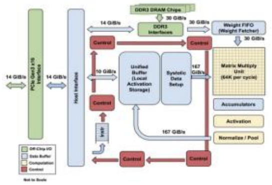
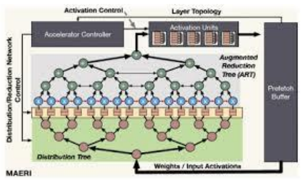
OpenCGRA: CGRA Generation Flow

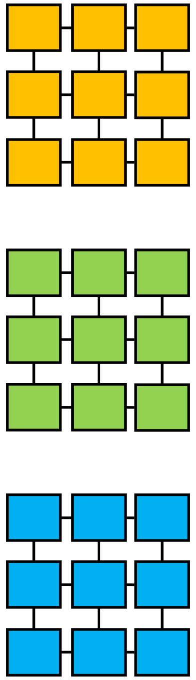
Partial Dynamic Reconfiguration for Streaming Applications in OpenCGRA

- Existing research/industry CGRA products accelerating streaming applications
  - One kernel at a time *vs.* statically partitioned for all kernels at the same time

Multi-kernel/ streaming application

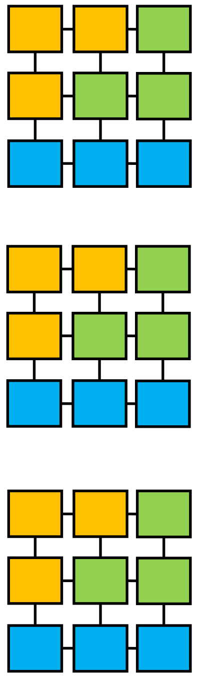





Used in the entirety for a kernel at a time

```
int main (...) {
    stage0();
    stage1();
    stage2();
    return 0;
}
```

Statically partitioned to all the kernels at the same time

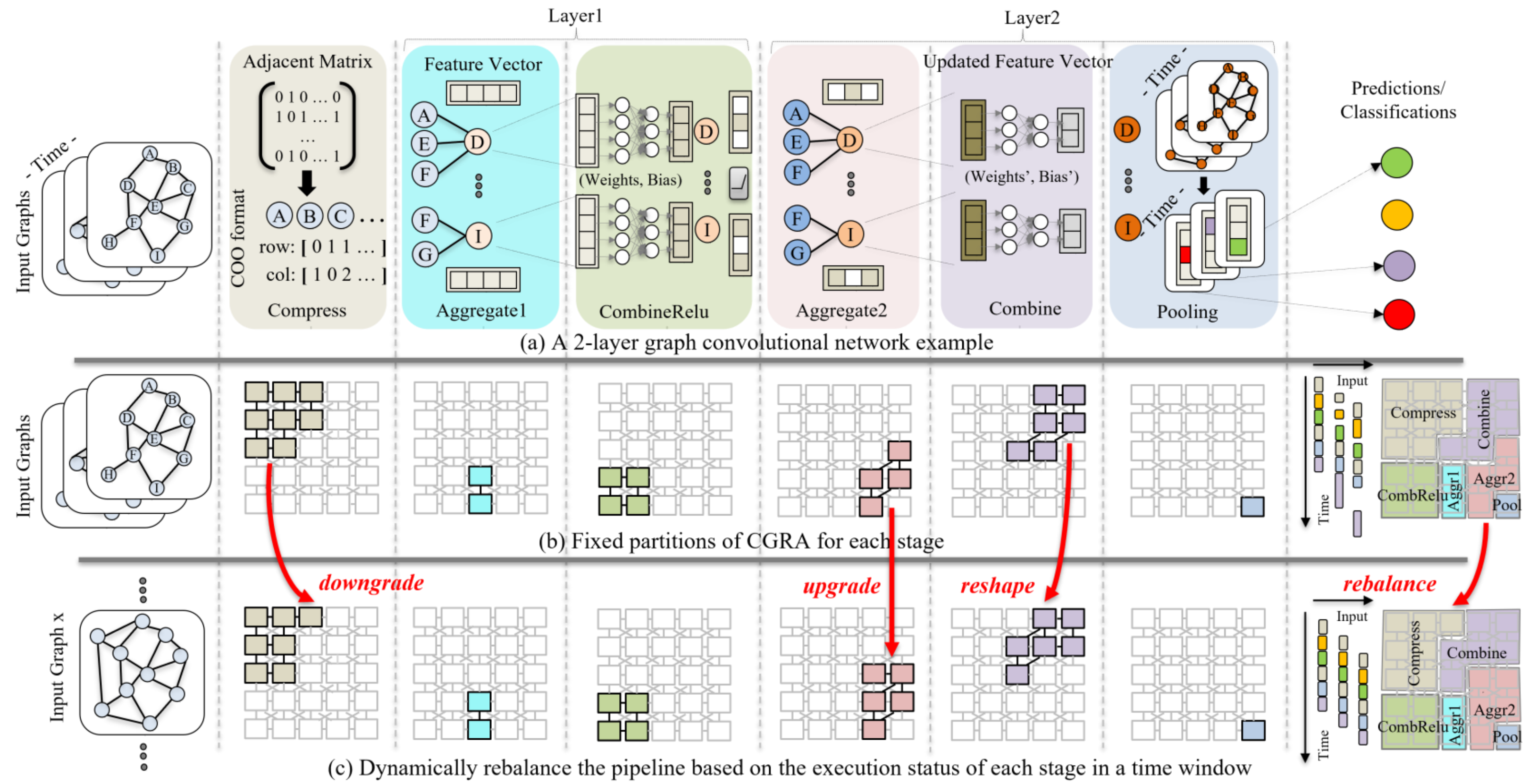


Partial Dynamic Reconfiguration for Streaming Applications in OpenCGRA

- OpenCGRA allows for partial dynamic reconfiguration
- Example case study: 2-layer GCN, with 5 kernels and different input graphs

Traditional – Statically partitioned for all the kernels at the same time

DynPaC – Dynamically reconfigure the CGRA after a time window



[C. Tan, *et al.*, "DynPaC: Coarse-Grained, Dynamic, and Partially Reconfigurable Array for Streaming Applications," *ICCD 2021- Best Paper Award*]
[C. Tan, *et al.*, "DRIPS: Dynamic Rebalancing of Pipelined Streaming Applications on CGRAs," *HPCA 2022*]

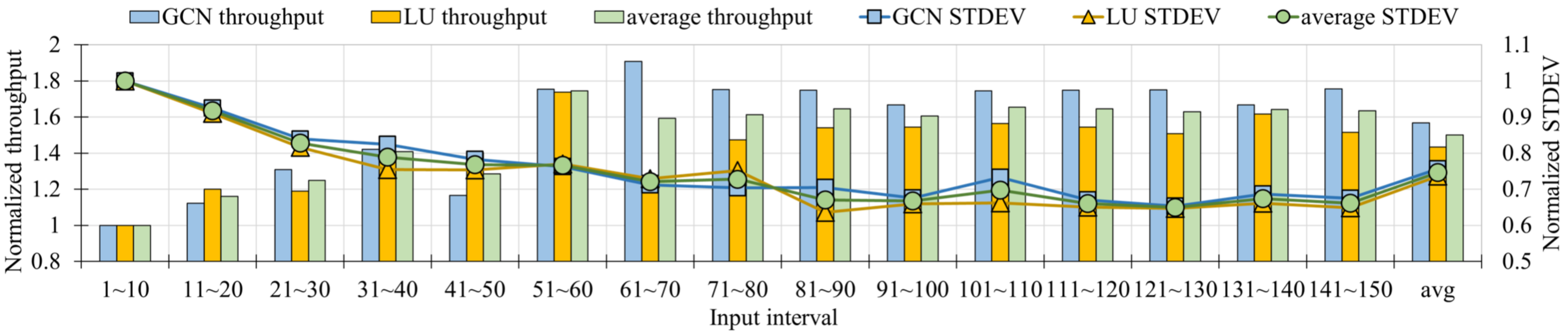
Case study: Experimental Evaluation

- Streaming applications targeting high throughput:
  - GCN: 5 kernels, ENZYME data set (150 for inference)
  - LU decomposition: 150 matrices (100x100) from UF Sparse Matrix Collection
- Design space: each kernel of an application runs on a CGRAs with different numbers of tiles (4x4, 4x8, 6x8) and unrolling factors (1, 2, and 4).
- #opt = number of LLVM instructions; OpSp = Optimal Speed Up; OpPa = Optimal Partition (regularly shaped)

Application	Dataset	Kernel	4x4 CGRA, U. F. = 1			4x8 CGRA, U. F. = 2			6x8 CGRA, U. F. = 4		
			#opt	OpSp	OpPa	#opt	OpSp	OpPa	#opt	OpSp	OpPa
2-layer Graph Convolutional Network (*GCN*)	ENZYME 600 graphs 450 for training 150 for inference	*Aggregate* (×2)	27	6.8	2×4	54	13.5	2×7	99	19.8	5×5
		Combine	26	6.5	2×3	52	13	3×5	95	23.8	5×5
		CombRelu	30	7.5	3×3	60	15	3×6	111	18.5	4×5
		Pooling	16	4	2×2	32	8	2×4	55	13.6	3×5
Synthesized Lower–Upper (*LU*) Decomposition kernels	150 matrices (within the size of 100×100) selected from the University of Florida sparse matrix collection	*Init*	7	1.8	1×2	11	4	1×3	19	4.8	2×3
		Decompose	87	12.4	3×4	167	20.9	5×5	327	23.4	6×6
		Solver0	31	7.8	3×3	63	12.6	4×4	121	17.3	4×5
		Solver1	33	8.3	3×3	67	13.4	4×4	129	18.4	4×5
		Invert	65	13	4×4	127	15.9	5×5	251	19.3	6×6
		Determinant	20	3.3	2×2	39	3.9	2×2	71	3.9	2×2

Case study: Experimental Evaluation

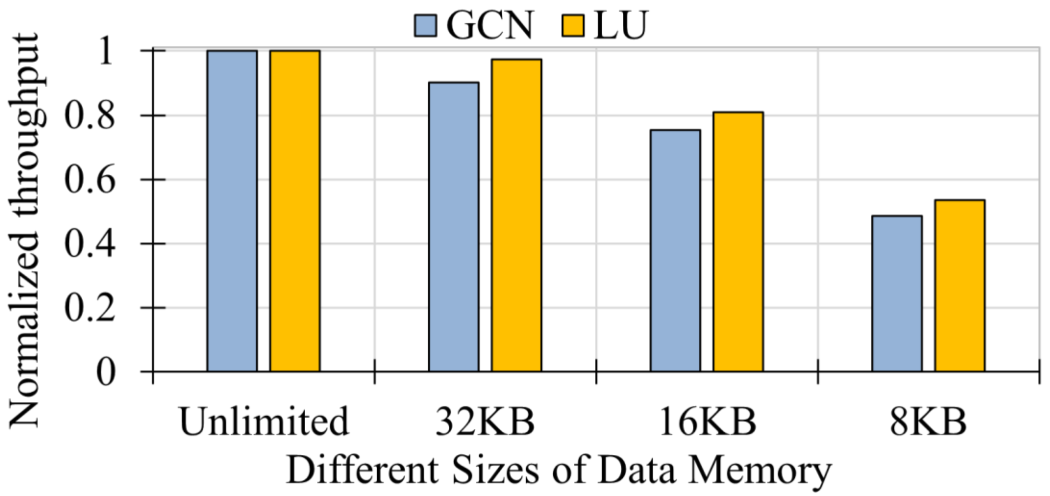
- Effects of dynamic rebalancing:
  - Triggered after time-window of 10 executions of the whole pipeline
  - Results include the dynamic reconfiguration overheads (<1k cycles, ~few nanoseconds)
  - Rebalancing overhead is negligible with respect to the execution time of the entire pipeline of kernels (e.g., 30k to 50k cycles for the GCN)



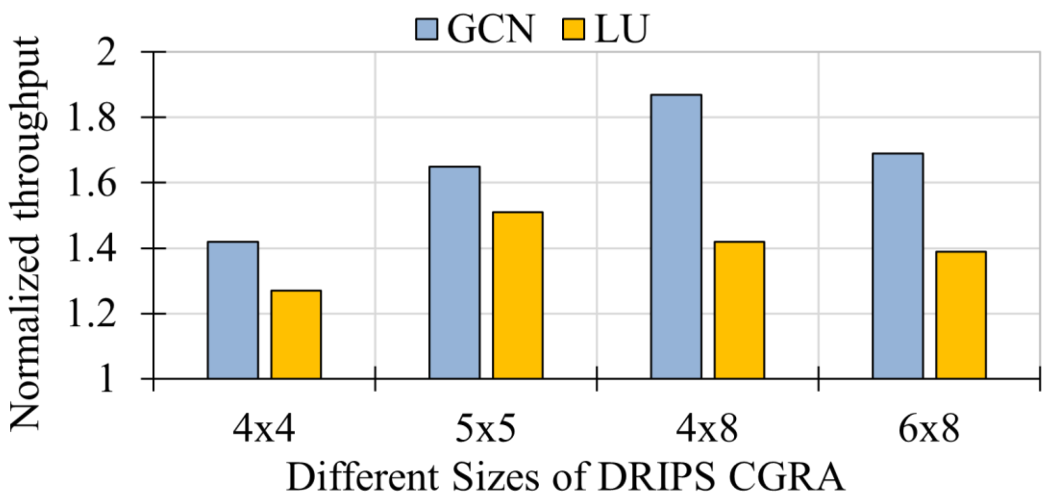
Throughput and standard deviation averaged per time window (i.e., 10 input samples)

Case study: Experimental Evaluation

- Architectural exploration:



Throughput of different SPM sizes

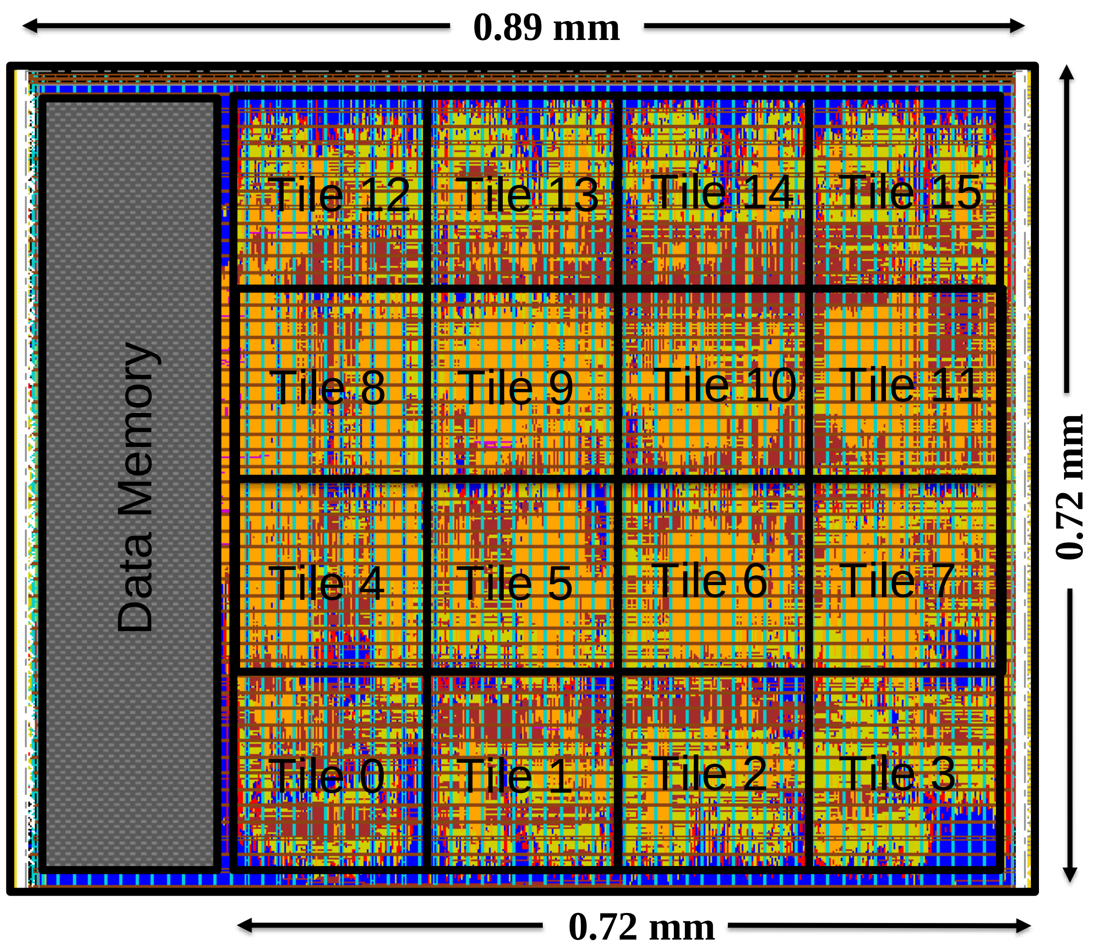


Throughput with DRIPS partial dynamic reconfiguration over statically partitioned design

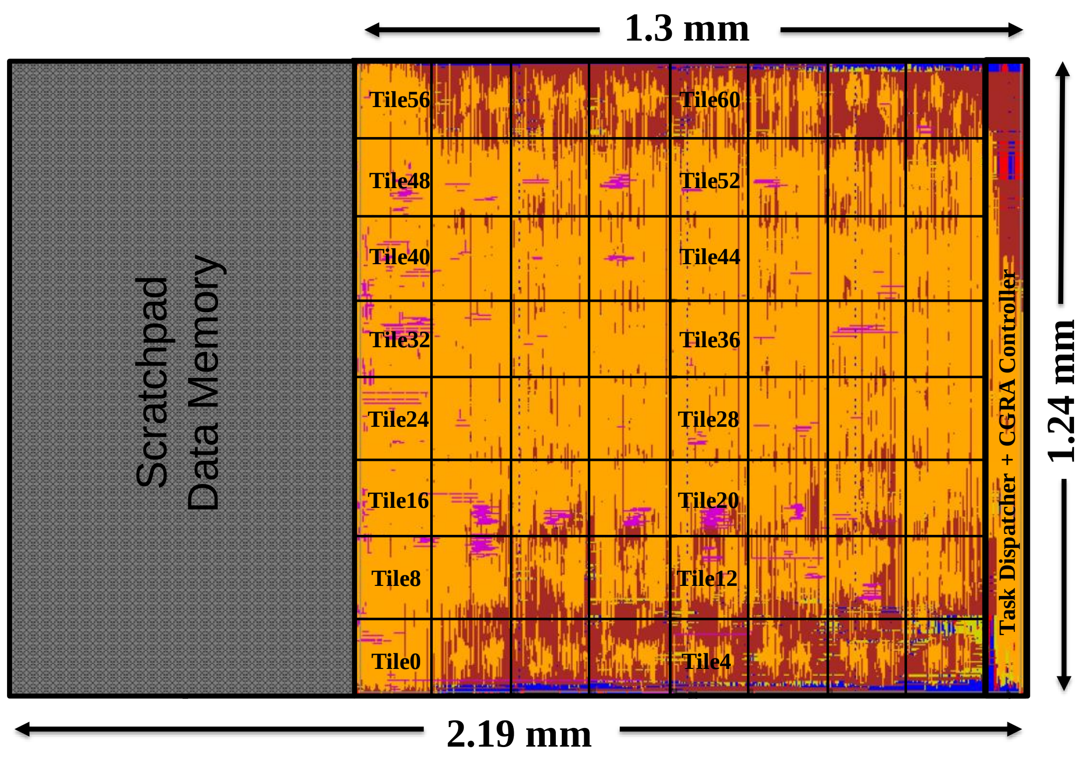
	SPM	Tile	Controller	Config NoC
Actual area (mm2)	1.1	0.63	0.248	0.09
Area distribution (%)	53.19%	30.46%	11.99%	4.35%

Area of a 5x5 CGRA design with 32KB SPM

SO(DA)^2 Example Designs



4x4 heterogeneous design with specialized tiles



8x8 homogeneous design with general-purpose tiles

Research Opportunities

- Evaluation of the impact of more advanced technology nodes and larger designs
  - Chiplet-based designs
  - 3D-Stacked Memory
- Integration of new functional unit tiles
  - New numeric formats
  - Highly specialized tiles generated with High-Level Synthesis
- Design Space Exploration with more complex heuristics than Simulated Annealing
  - Bioinspired search heuristics
  - Reinforcement learning
- Dataflow computing model vs. Static scheduling
- Additional metrics and monitors for partial dynamic reconfiguration

Acknowledgements

- This research was supported by PNNL Laboratory Directed Research & Development (LDRD) programs:
  - The "*Software Defined Accelerators for Ultra-Low Latency Reasoning*" (SODA-ULTRA) project in the Adaptive Tunability for Synthesis and Control via Autonomous Learning on Edge (AT SCALE) Initiative
  - The "*Software Defined Architectures for Portability and Performance*" (SODAPOP) project in the Data-Model Convergence (DMC) Initiative

Key Takeaways

- Open-source agile hardware design and prototyping:
  - SODA-OPT: https://gitlab.pnnl.gov/sodalite/soda-opt
  - OpenCGRA: https://github.com/pnnl/opencgra
- References:
  - SODA-OPT
    - ✓ N. Bohm Agostini, *et al.*, "An MLIR-based Compiler Flow for System-level Design and Hardware Acceleration," *ICCAD 2022.*
  - OpenCGRA
    - ✓ C. Tan, *et al.*, "OpenCGRA: An Open-Source Unified Framework for Modeling, Testing, and Evaluating CGRAs," *ICCD 2020.*
    - ✓ C. Tan, *et al.*, "AURORA: Automated Refinement of Coarse-Grained Reconfigurable Accelerators," *DATE 2021.*
    - ✓ C. Tan, *et al.*, "DynPaC: Coarse-Grained, Dynamic, and Partially Reconfigurable Array for Streaming Applications," *ICCD 2021.*
    - ✓ C. Tan, *et al.*, "DRIPS: Dynamic Rebalancing of Pipelined Streaming Applications on CGRAs," *HPCA 2022.*

Pacific Northwest
NATIONAL LABORATORY

Thank you!