Pacific Northwest
NATIONAL LABORATORY

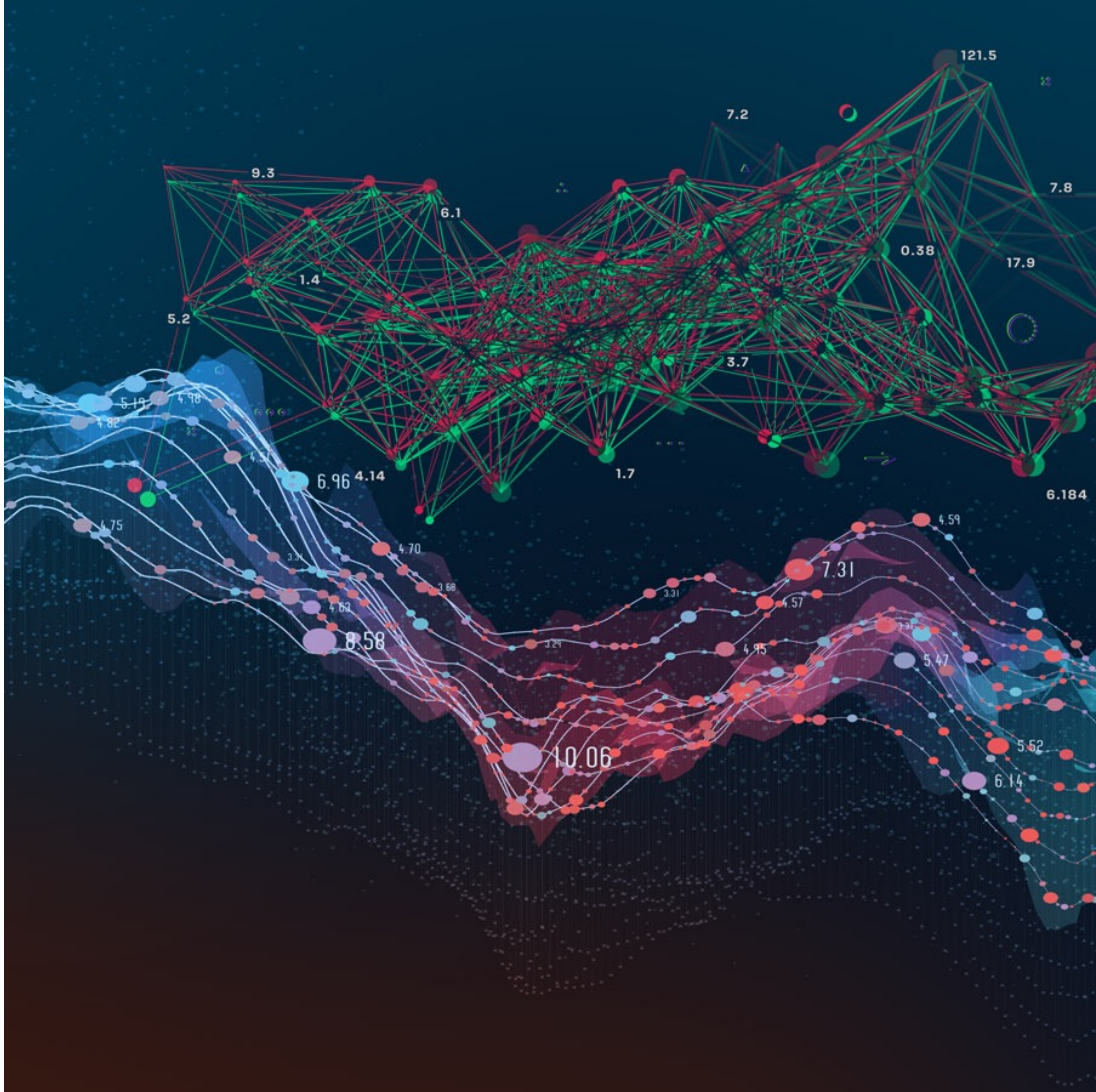
SODA: An End-To-End Open-Source Hardware Compiler for Machine Learning Accelerators

October 12, 2022

Nicolas Bohm Agostini, Serena Curzel, Ankur Limaye
Vinay Amatya, Marco Minutoli, Vito Giovanni Castellana
Joseph Manzano, Fabrizio Ferrandi, Antonino Tumeo

Antonino Tumeo

Chief Scientist, High Performance Computing Group

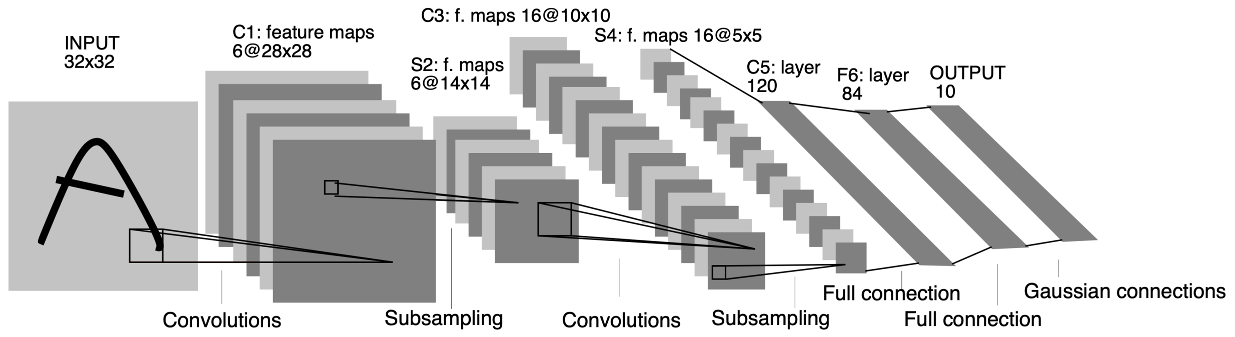


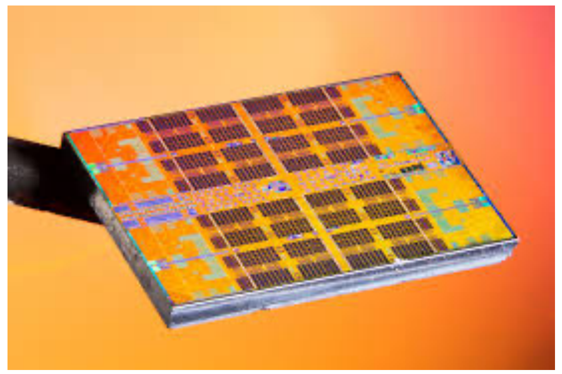
Motivations

- Data science algorithms, approaches, and frameworks are quickly evolving
- Domain-specific accelerators are the only possible approach to keep increasing performance in tight constraints
- Existing accelerators start from specific models (i.e., mostly deep neural networks) or only try to accelerate specific computational patterns coming from high-level frameworks
- Designing hardware by hand is complex and time-consuming
- Depending on the application, a designer may want to explore performance, area, energy, accuracy, and more…

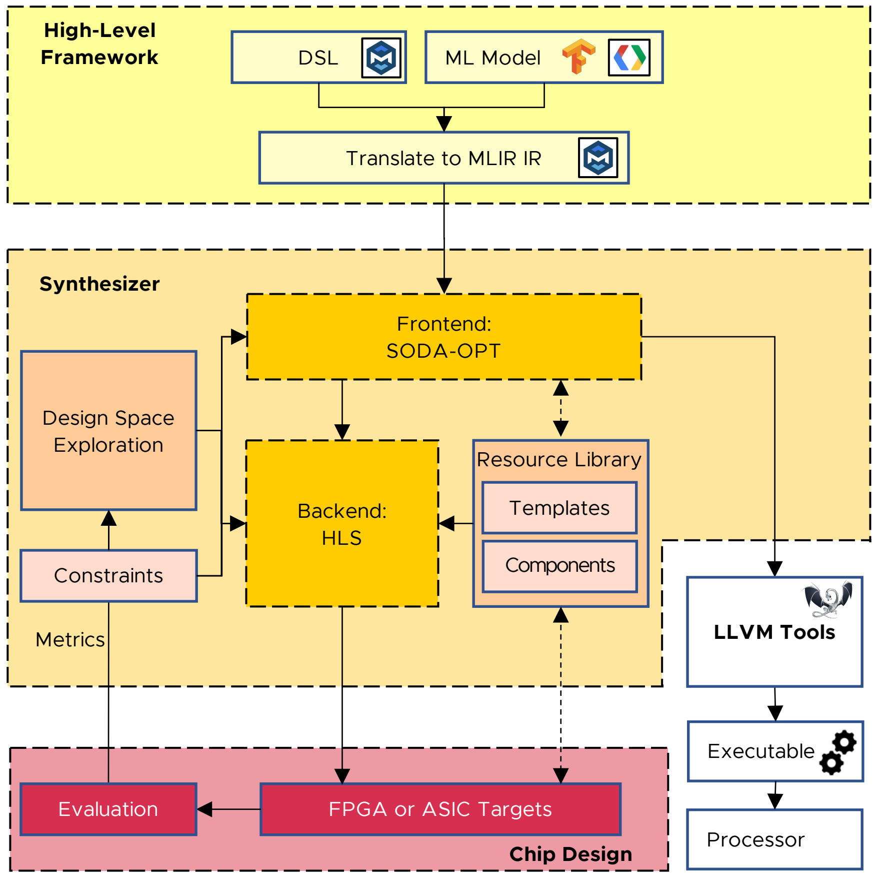
- ***Need tools to quickly transition from formulation of an algorithm to the accelerator implementation and explore the accelerator design along different dimensions***

LeNet architecture from the original paper





SODA Synthesizer: Overview



[Marco Minutoli, Vito Giovanni Castellana, Cheng Tan, Joseph B. Manzano, Vinay Amatya, Antonino Tumeo, David Brooks, Gu-Yeon Wei: SODA: a New Synthesis Infrastructure for Agile Hardware Design of Machine Learning Accelerators. ICCAD 2020: 98:1-98:7]

[Jeff Jun Zhang, Nicolas Bohm Agostini, Shihao Song, Cheng Tan, Ankur Limaye, Vinay Amatya, Joseph B. Manzano, Marco Minutoli, Vito Giovanni Castellana, Antonino Tumeo, Gu-Yeon Wei, David Brooks: Towards Automatic and Agile AI/ML Accelerator Design with End-to-End Synthesis. ASAP 2021: 218-225]

- A modular, multi-level, interoperable, extensible, **open-source hardware compiler** from **high-level programming frameworks to silicon**
- Compiler-based frontend, leveraging the MultiLevel Intermediate Representation (MLIR)
- **Compiler-based backend**, leveraging state-of-the-art High-Level Synthesis (HLS) techniques
- Generates **synthesizable Verilog** for a variety of targets, from Field Programmable Gate Arrays (FPGAs) to Application Specific Integrated Circuits (ASICs)
- Optimizations at all levels are performed as **compiler optimization** passes

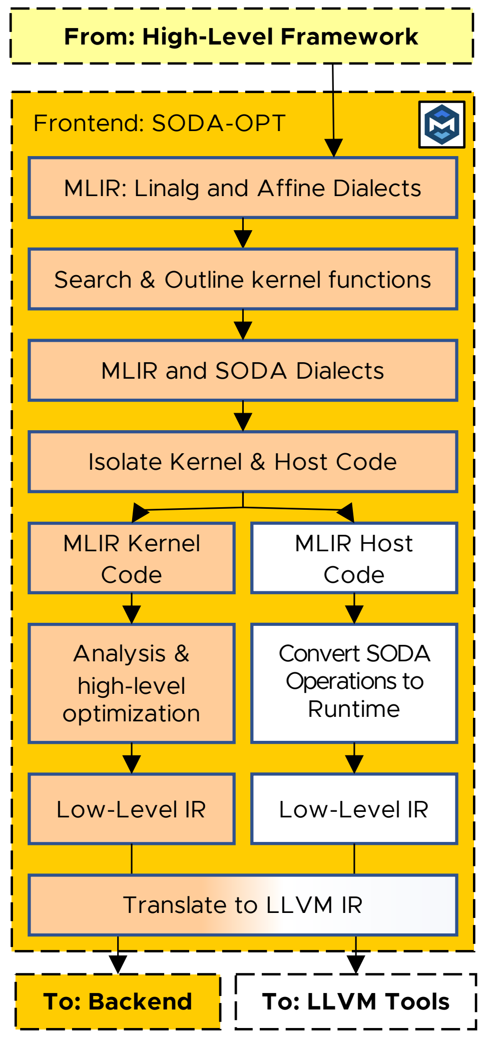
[N. Bohm Agostini, S. Curzel, J. Zhang, A. Limaye, C. Tan, V. Amatya, M. Minutoli, V.G. Castellana, J. Manzano, G-Y. Wei, D. Brooks, A. Tumeo: Bridging Python to Silicon: The SODA Toolchain. IEEE Micro Magazine, Sept/Oct 2022.]

SODA-OPT: Frontend and High-Level IR

- **SODA-OPT: S**earch, **O**utline, **D**ispatch, **A**ccelerate frontend **opt**imizer "generates" the SODA High-Level IR
- Employs and embraces the **MLIR** framework
  - MLIR: Multi-Level Intermediate Representation
  - Used in TensorFlow, TFRT, ONNX-MLIR, NPComp, others
  - Several architecture independent dialects (Linalg, Affine, SCF) and optimizations
- Interfaces with **high-level ML frameworks** through 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)
  - Generate interfacing code and runtime calls for microcontroller

[N. Bohm Agostini, D. Kaeli, A. Tumeo: SODA-OPT: System-Level Design in MLIR for HLS. SC 21 Poster]

[N. Bohm Agostini, S. Curzel, V.C. Amatya, C. Tan, M. Minutoli, V. G. Castellana, J. Manzano, D. Kaeli, A. Tumeo, An MLIR-based Compiler Flow for System-Level Design and Hardware Acceleration. To appear at ICCAD 2022]

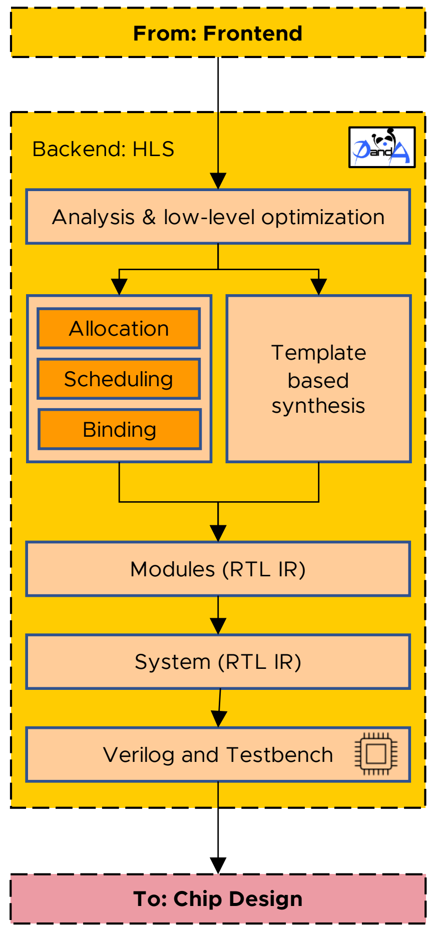

From: High-Level Framework
Frontend: SODA-OPT
MLIR: Linalg and Affine Dialects
Search & Outline kernel functions
MLIR and SODA Dialects
Isolate Kernel & Host Code
MLIR Kernel Code
MLIR Host Code
Analysis & high-level optimization
Convert SODA Operations to Runtime
Low-Level IR
Low-Level IR
Translate to LLVM IR
To: Backend
To: LLVM Tools

SODA-OPT: System Overview

https://gitlab.pnnl.gov/sodalite/soda-opt

SODA Synthesizer: HLS Backend

- The synthesizer backend take as input the properly optimized low-level IR and generate the hardware descriptions of the accelerators
  - SODA supports an open-souce HLS backend, PandA-Bambu, and a the commercial AMD/Xilinx Vitis HLS
- PandA-Bambu is an open-source state-state-of-the-art high-level synthesis (HLS)
  - Unique features include **parallel accelerator designs**, **modular HLS**, and **ASIC support**
  - Provides automated testing and verification of the generated designs



https://panda.dei.polimi.it

[Fabrizio Ferrandi, Vito Giovanni Castellana, Serena Curzel, Pietro Fezzardi, Michele Fiorito, Marco Lattuada, Marco Minutoli, Christian Pilato, Antonino Tumeo: Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications. DAC 2021: 1327-1330]

Why an HLS Backend?

- Provides the necessary **generality** to deal with novel algorithms
- Provides opportunities for **specialized** and optimized templates by recognizing specific computational patterns

- The SODA Approach relies on **progressive lowerings** of compiler intermediate representations (IRs), rather than rewriting annotated C/C++
  - Reduces semantic mismatches between high-level and low-level descriptions
  - Provides further opportunities to apply optimizations at the right level of abstraction

- New optimizations as additional compiler passes
- Design space exploration formulated as a compiler optimization problem

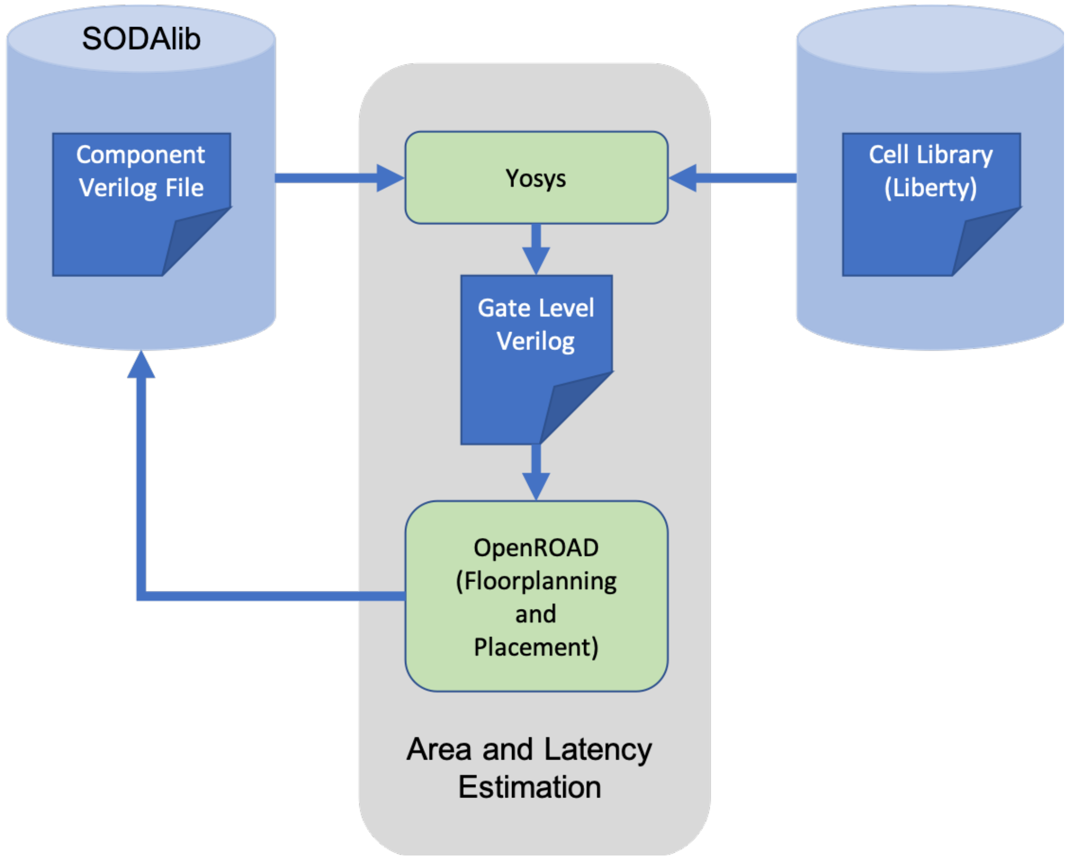
SODA Synthesizer: ASIC targets

- The multi-level approach of the SODA toolchain allows supporting different target technologies (FPGA, ASIC) for actual generation of the designs

- ASIC targets:
  - **Commercial Tools** (Synopsys Design Compiler with Global Foundries 12/14 nm cells)
  - **OpenROAD suite** (OpenPDK 45nm and ASAP 7nm cell libraries)

- Backend' resources characterized for the target technology:
  - **Eucalyptus** tool in Bambu, allows driving hardware synthesis algorithms to optimize for area, latency, etc.

- PandA-Bambu now also the opensource C frontend for **ZeroASIC' SIliconCompiler** (https://www.siliconcompiler.com)



SODA characterization flow. The characterization flow can be extended to synthesize HLS generated designs, or used to estimate their area-latency-power profiles to drive the Design Space Exploration engine

OpenROAD

https://theopenroadproject.org

SODA-Opt Optimization Passes

Optimization	Benefit for HLS	Passes
Single basic block containing the compute intensive part of the kernel	More freedom to schedule operations	Tiling, Unrolling
Increased instruction-level parallelism	Schedule independent arithmetic operations on the same cycle when their inputs are available	Unrolling
Increased data level parallelism	Schedule operations into different memory units on the same cycle	Tiling, Unrolling, Temporary Buffer Allocation
Avoid unnecessary reads from kernel arguments	Reduce expensive accesses to external memory	Temporary Buffer Allocation, Alloca Buffer Promotion
Reuse read results, aggregate on scalars	Save scalar values loaded from memory and intermediate results in registers rather than performing repeated memory accesses	Scalar Replacement of Aggregates
Early alias analysis	Schedule memory operations independently on regions that don't alias	Early Alias Analysis, Outlining pass
Remove redundant or unnecessary operations	Avoid wasting resources	Common Sub-expression Elimination, Dead Code Elimination

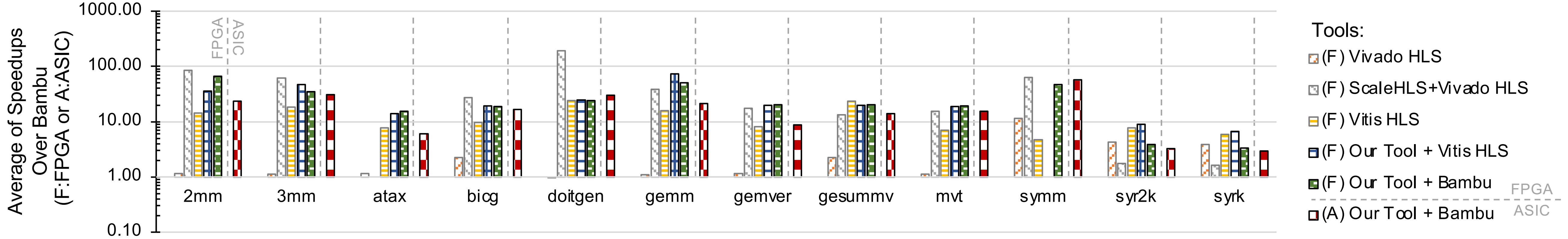
- **SODA-Opt** provides passes to: **S**earch, **O**utline, Isolate, Optimize, **D**ispatch and **A**ccelerate kernels for the hardware synthesis backend
- SODA-Opt **defines the SODA dialect** to perform search and outlining of the kernels to accelerate
- SODA-Opt **performs high-level optimizations** for high-level synthesis

Examples of generated accelerators

- PolyBench kernels
- Outperforming state-of-the-art HLS tools and frontends

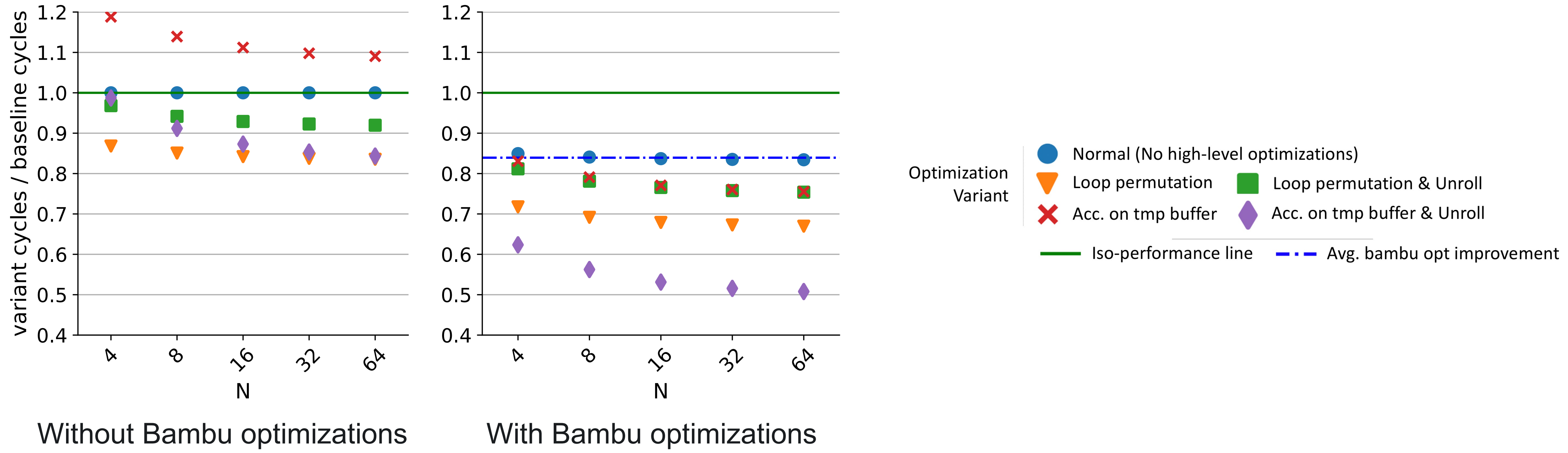
Execution time (in clock cycles) for PolyBench kernels with ASIC Target - OpenPDK 45nm @ 500MHz. Speedup shown in parenthesis.

Opt. Strategy	No High Level Opts.				SODA-OPT Pipeline			
Kernel Size	**2**	**4**	**8**	**16**	**2**	**4**	**8**	**16**
symm	421	2,928	21,400	163,368	31 (13.6x)	34 (86.1x)	325 (65.8x)	2,600 (62.8x)
three_mm	388	3,087	25,010	211,298	47 (8.3x)	82 (37.6x)	656 (38.1x)	5,248 (40.3x)
two_mm	315	2,475	20,258	167,490	52 (6.1x)	86 (28.8x)	688 (29.4x)	5,504 (30.4x)
gemm	186	1,446	11,922	95,376	31 (6.0x)	56 (25.8x)	448 (26.6x)	3,584 (26.6x)
doitgen	277	4,282	67,666	999,698	29 (9.6x)	258 (16.6x)	2,064 (32.8x)	16,512 (60.5x)
bicg	129	518	2,058	8,482	26 (5.0x)	43 (12.0x)	85 (24.2x)	340 (24.9x)
mvt	130	514	2,051	8,195	26 (5.0x)	45 (11.4x)	89 (23.0x)	356 (23.0x)
gemver	283	1,118	4,393	17,617	77 (3.7x)	106 (10.5x)	424 (10.4x)	1,696 (10.4x)
gesummv	162	578	2,178	8,722	39 (4.2x)	56 (10.3x)	105 (20.7x)	420 (20.8x)
atax	132	523	2,067	8,227	44 (3.0x)	73 (7.2x)	292 (7.1x)	1,168 (7.0x)
syr2k	186	1,310	9,018	68,986	38 (4.9x)	567 (2.3x)	3,033 (3.0x)	24,264 (2.8x)
syrk	142	990	6,714	49,250	31 (4.6x)	453 (2.2x)	2,581 (2.6x)	20,648 (2.4x)
trmm	46	532	4,402	34,018	24 (1.9x)	532 (1.0x)	4,402 (1.0x)	34,018 (1.0x)



[N. Bohm Agostini, S. Curzel, V.C. Amatya, C. Tan, M. Minutoli, V. G. Castellana, J. Manzano, D. Kaeli, A. Tumeo, An MLIR-based Compiler Flow for System-Level Design and Hardware Acceleration. To appear at ICCAD 2022]

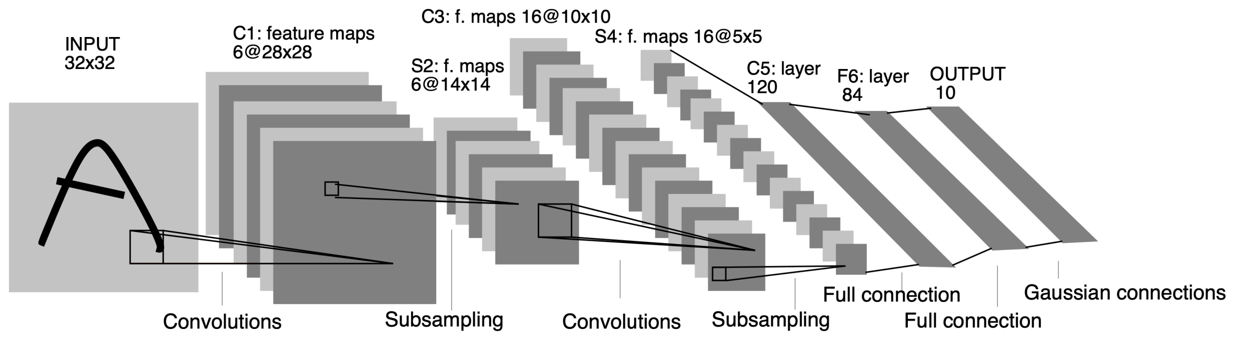
Benefits of high-level optimizations



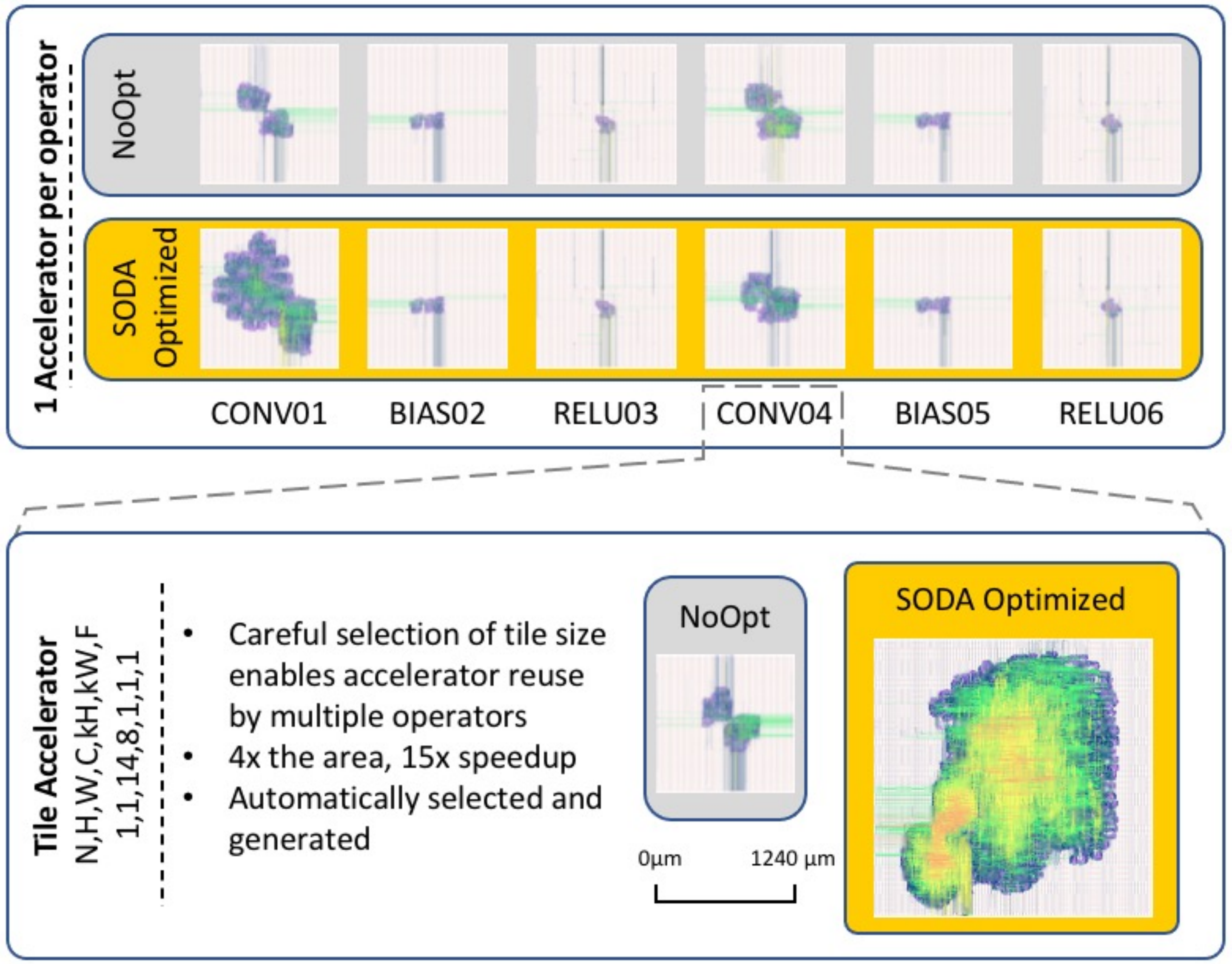
- Matrix multiplication kernel with different input sizes
- Optimization passes in SODA-OPT prepare the input IR for Bambu exposing more parallelism (loop permutation, loop unrolling, accumulation on temporary buffers)
- Even greater effect when **combined** with low-level HLS optimizations
- **Significant impact on the performance** of the generated accelerators

From Python to optimized ASIC

LeNet architecture from the original paper



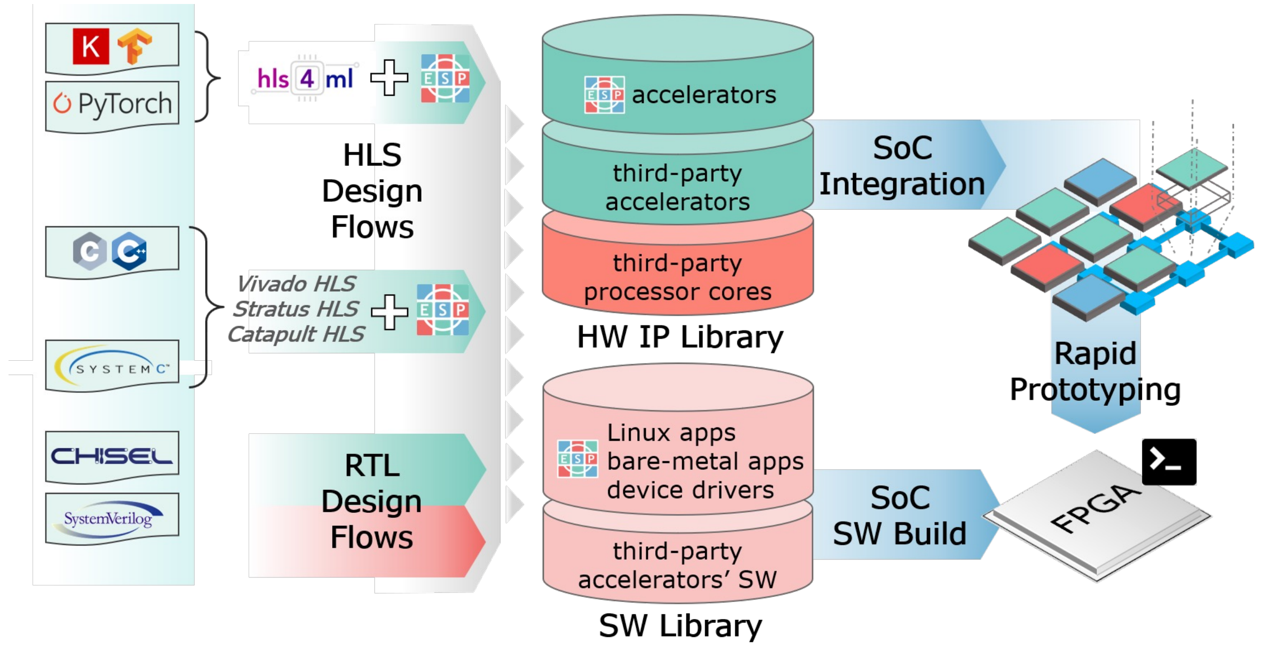
- LeNet example
- Each of the operator is synthesized to an **ASIC accelerator**
- SODA-Opt optimized accelerators are bigger, but also much **faster**



- Careful selection of tile size enables accelerator reuse by multiple operators
- 4x the area, 15x speedup
- Automatically selected and generated

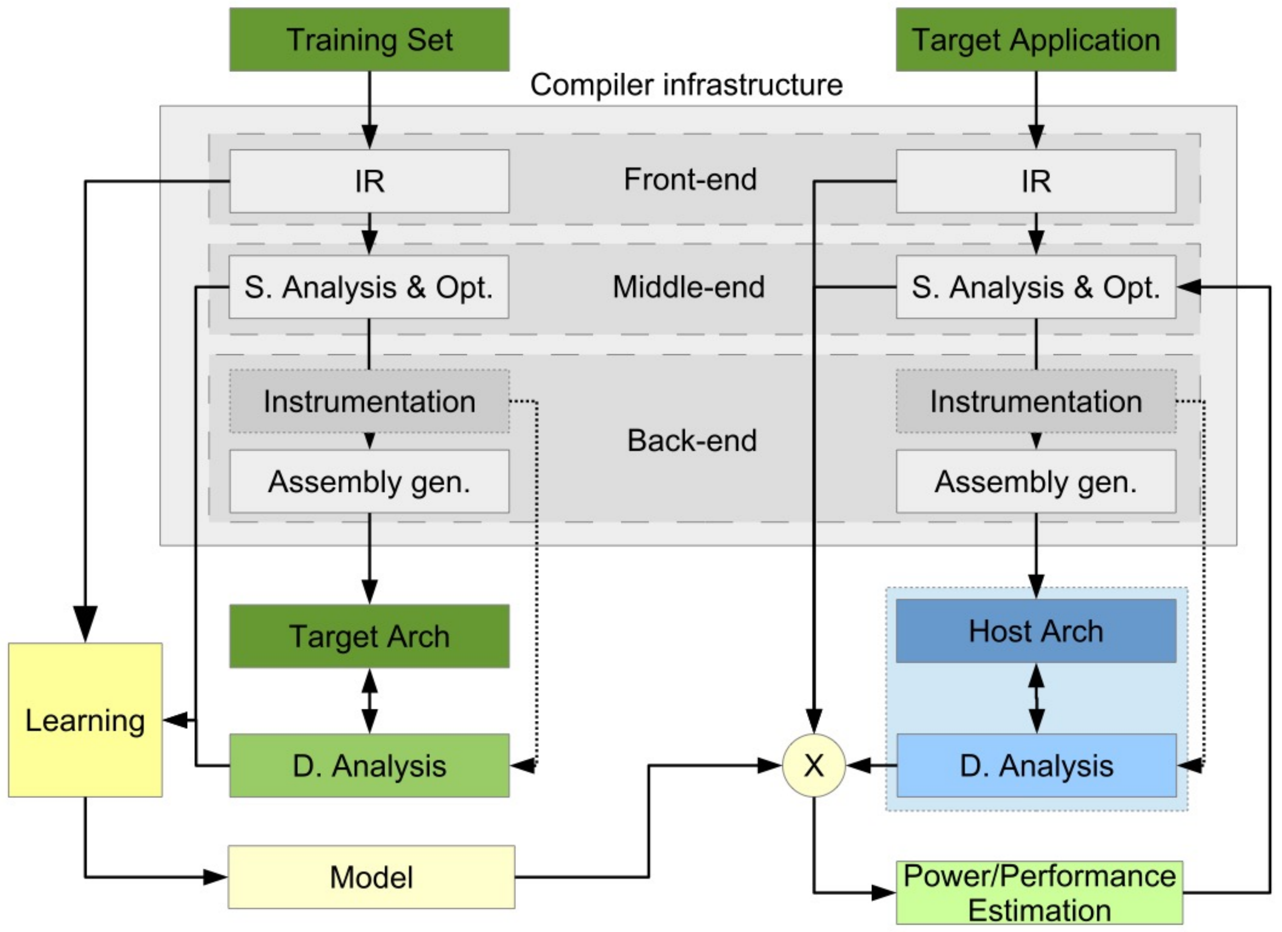
Research Opportunities: System-Level Design

- Integrating with open-source fast prototyping platforms: Columbia University Embedded Scalable Platforms (ESP)
- SODA-OPT
  - MLIR is naturally modular and hierarchical
  - Can lower to multiple targets, including runtimes
  - A more comprehensive tool than HLS4ML
- Bambu
  - Provides a fully open-source HLS backend for ESP
- Enables end-to-end fast prototyping from algorithmic concept to system implementation



Research Opportunities: Profile Driven Synthesis

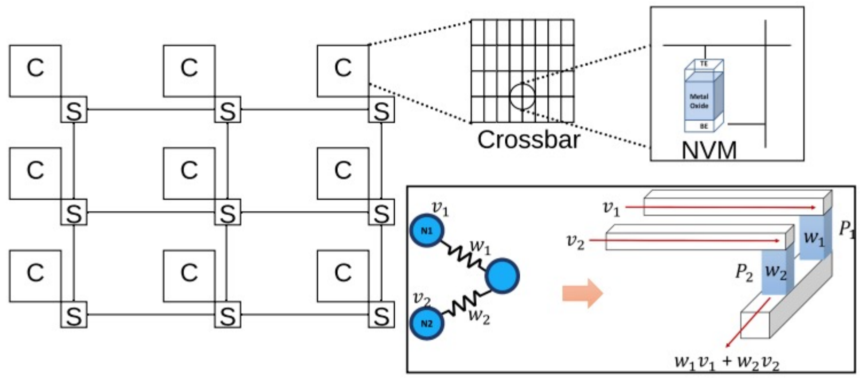
- A multi-level compiler infrastructure provides **static analysis**
- A compiler infrastructure provides opportunities to implement **dynamic analysis** through automated instrumentation and profiling
  - E.g., capturing data-dependent patterns and memory transactions
  - Information can be feed back to the hardware generation engine to facilitate exploration of the memory and the overall architecture design



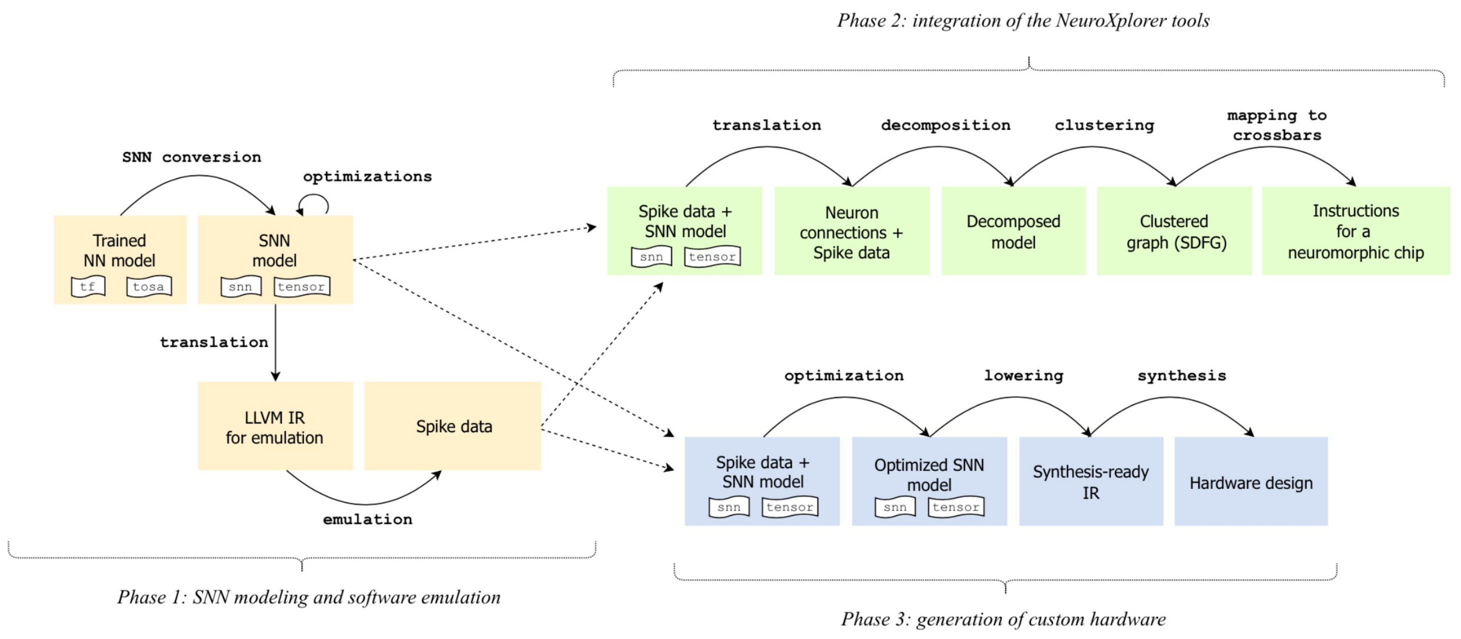
[A. Tumeo: Architecture independent integrated early performance and energy estimation. IGSC 2017: 1-6]

Research Opportunities: Support for Spiking Neural Networks

- Extend the SODA approach to **Spiking Neural Networks**
  - Integration with Drexel's NeuroXplore framework
- Defined a new **MLIR dialect** to perform conversion from ANN to SNN
- The SODA framework then allows mapping to (synthesized) digital neurons



[Spiking Neural Network implementation]



[S. Curzel *et al.*, "Automated Generation of Integrated Digital and Spiking Neuromorphic Machine Learning Accelerators," ICCAD 2021]

Public Software Repositories

- SODA frontend: https://gitlab.pnnl.gov/sodalite/soda-frontend
- SODA-Opt: https://gitlab.pnnl.gov/sodalite/soda-opt
- Panda-Bambu HLS: https://panda.dei.polimi.it (latest release 0.9.7)
- OpenROAD: https://theopenroadproject.org (external tool, leveraged by SODA toolchain to achieve end-to-end synthesis to ASIC in a fully opensource compiler toolchain)
- SiliconCompiler: https://siliconcompiler.com (external tool developed by ZeroASIC, Bambu has been included as a frontend from C/C++)

Conclusions

- SODA implements an **end-to-end** (high-level frameworks to silicon) **compiler-based toolchain** for the generation of domain-specific accelerators
  - Modular, multi-level, extensible
  - All based on interoperating open-source technologies
  - Targets reconfigurable architectures FPGAs as well ASICs
  - Consider specialization and partial dynamic reconfiguration as key optimization metrics
  - Considers system-level implications
  - Enables automated design space exploration and agile hardware design

- **The SODA Synthesizer provides a no-human-in-the-loop toolchain from algorithmic formulation to hardware implementation for complex workloads**

Pacific Northwest
NATIONAL LABORATORY

Thank you