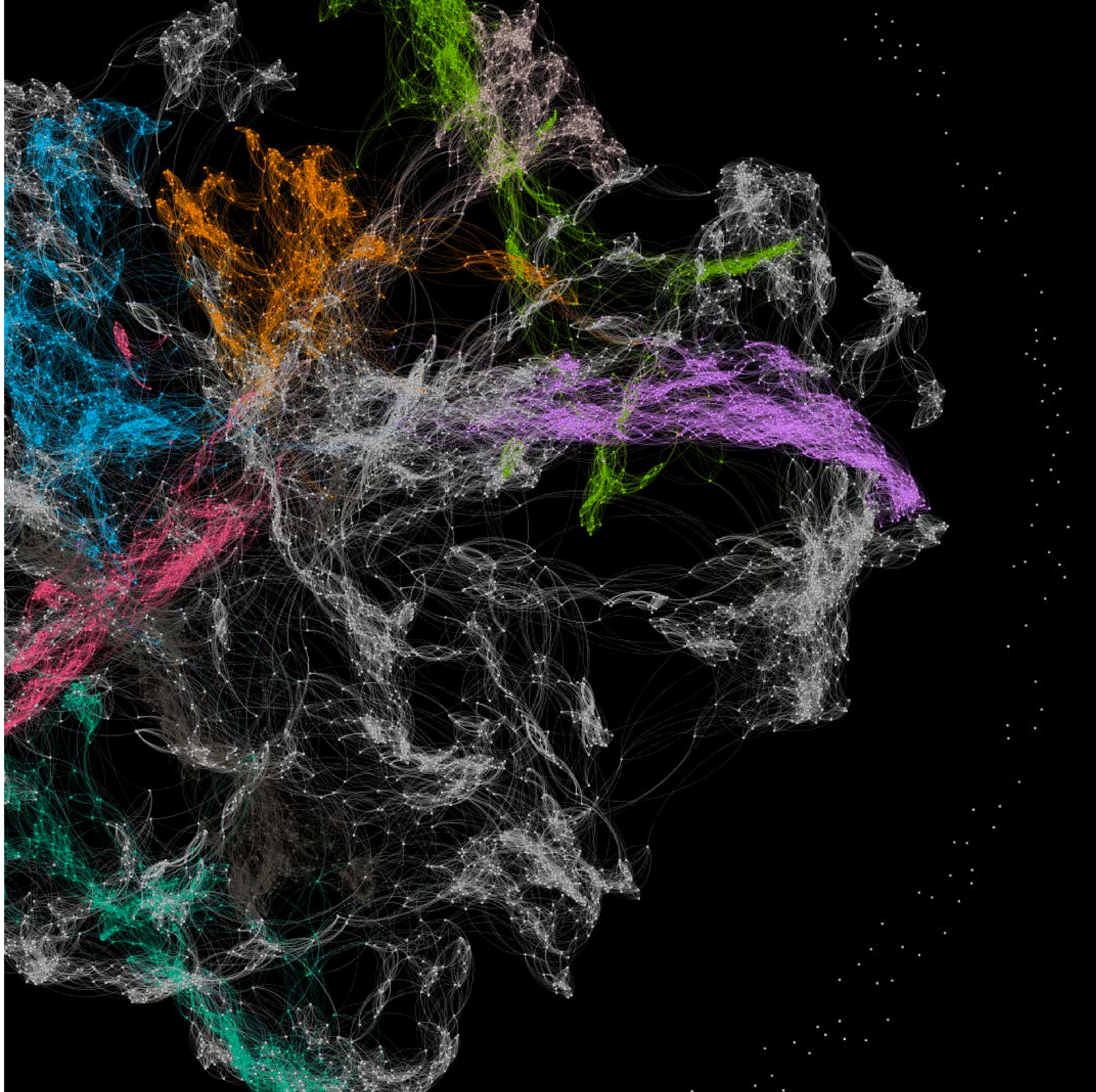
Pacific Northwest NATIONAL LABORATORY

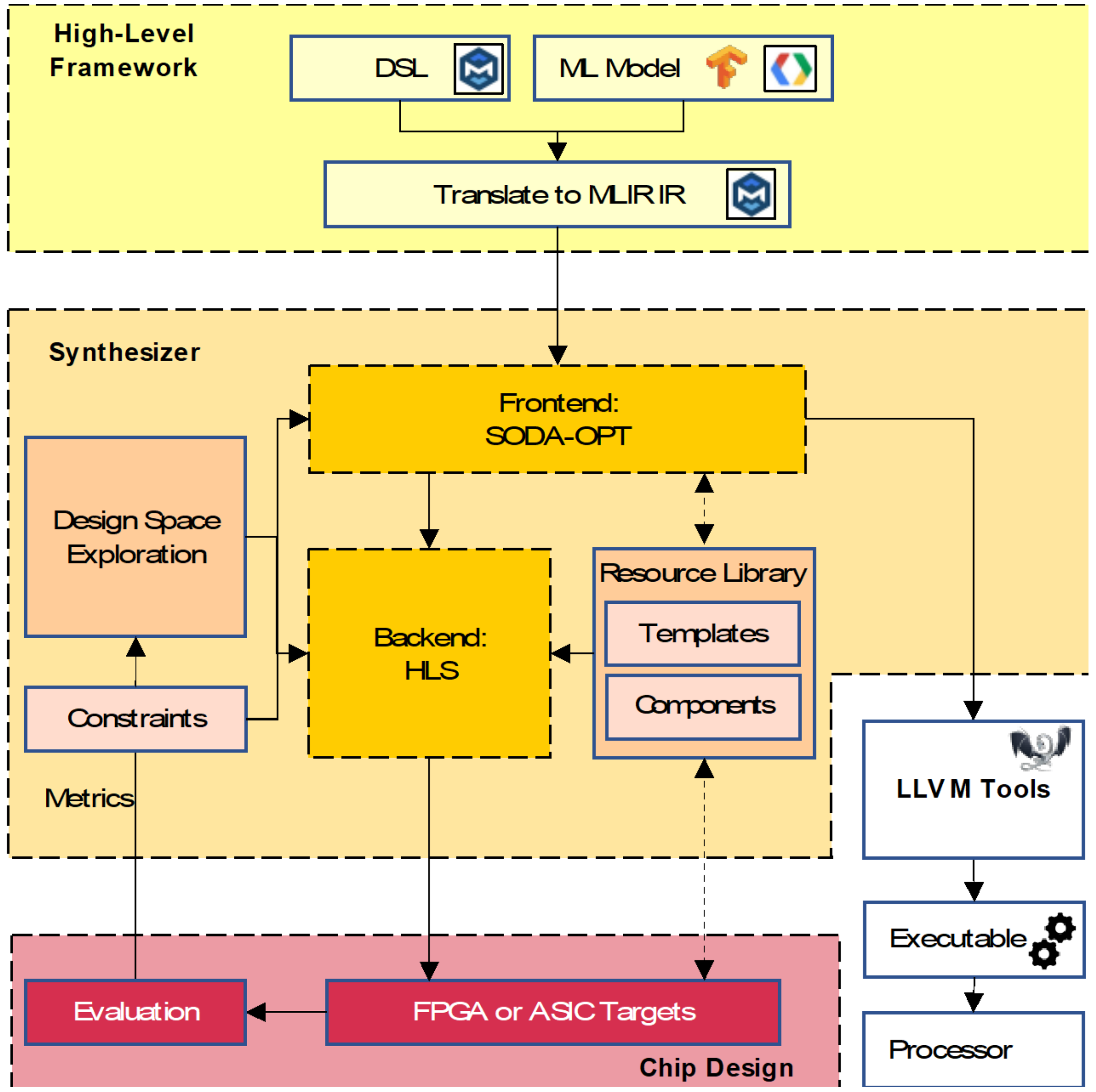
From High-Level Frameworks to custom Silicon with SODA

Serena Curzel, Nicolas Bohm Agostini, Reece Neff, Ankur Limaye, Jeff (Jun) Zhang, Vinay Amatya, Marco Minutoli, Vito Giovanni Castellana, Joseph Manzano, David Brooks, Gu-Yeon Wei, Fabrizio Ferrandi, Antonino Tumeo

U.S. DEPARTMENT OF ENERGY BATTELLE

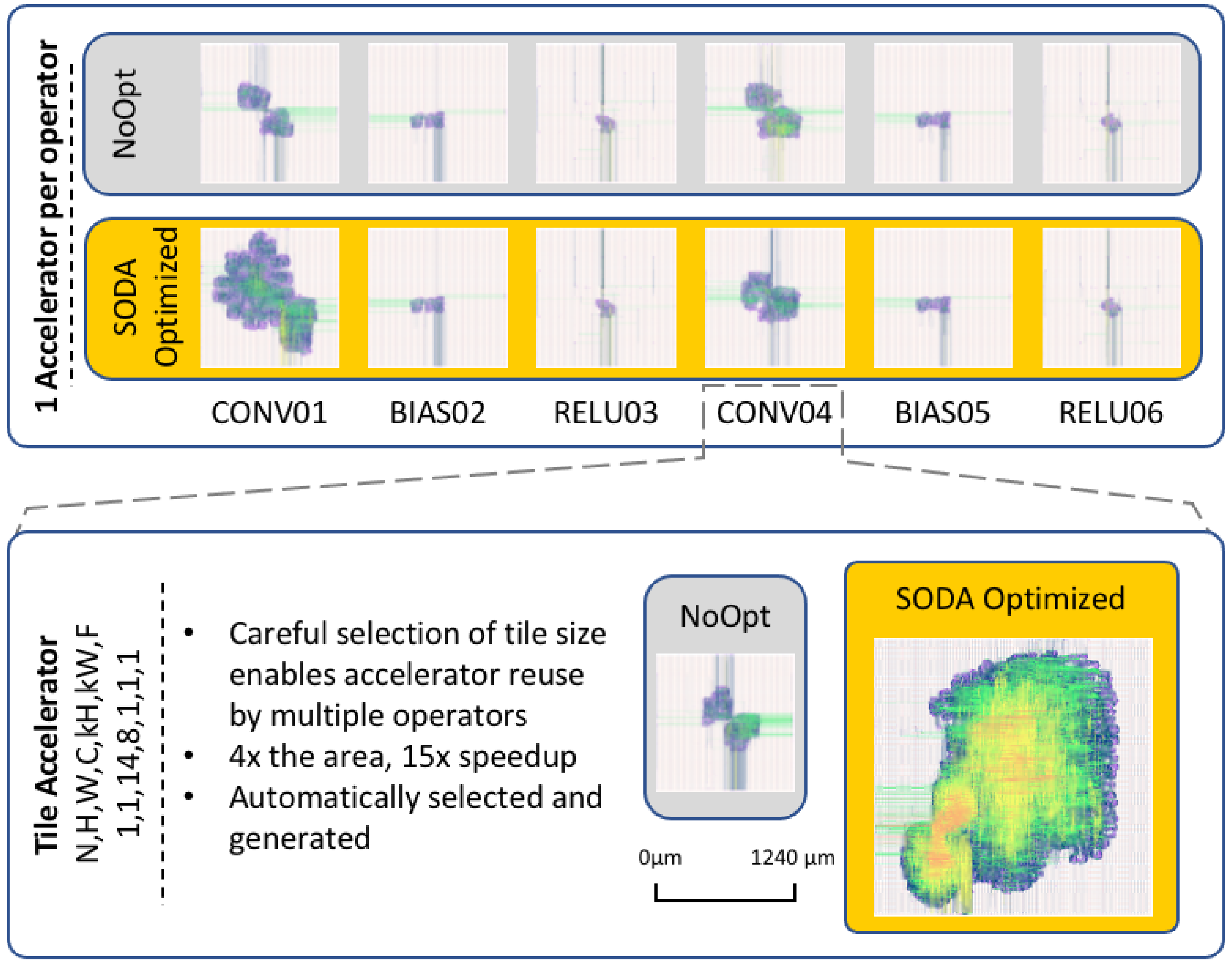


Overview



- The SODA Synthesizer is 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

Results



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

ASIC accelerators for LeNet layers

Useful links

SODA-OPT

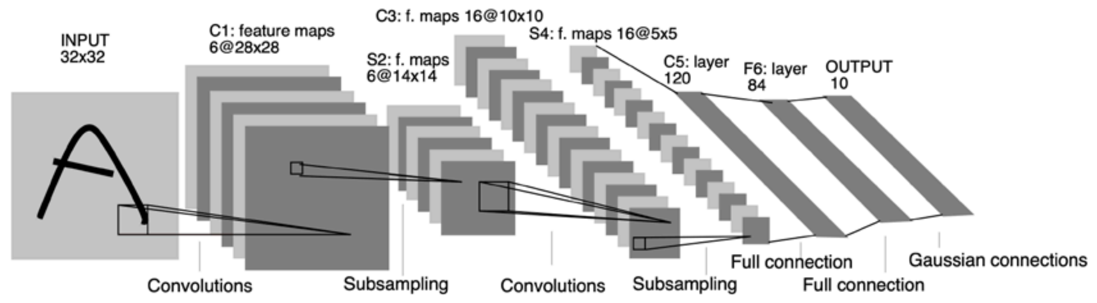
Panda-Bambu HLS (v 0.9.7)

SODA Docker Image

SODA Tutorial: *DATE 2022*

Motivations

- Data Science algorithms, Machine Learning models and frameworks are quickly evolving



[Y. Lecun, *et al.*, "Gradient-based learning applied to document recognition," *Proc. IEEE*, 1998}

Increasing number of layers and parameters

(ResNet VGG, Transformers...)

New network architectures

(GNN, LSTM, Reinforcement Learning...)

Compression techniques

(Quantization, pruning...)

New programming environments

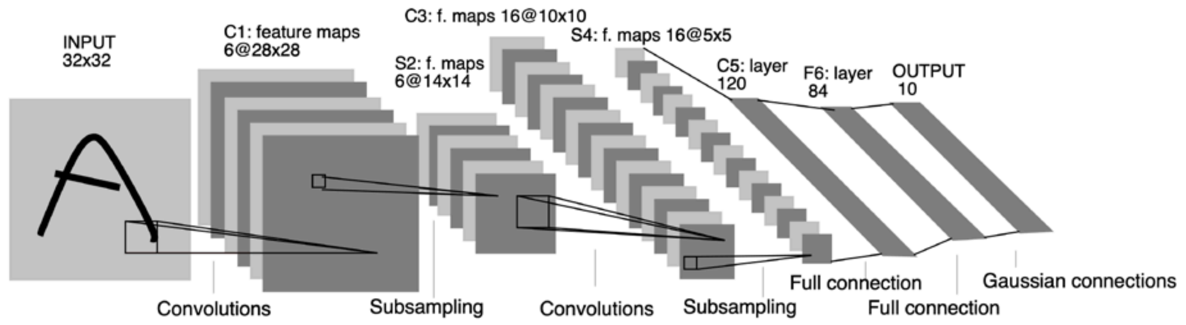
(TensorFlow, PyTorch, MXNet...)

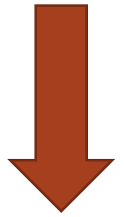
- Increased complexity and tight performance/power/area constraints (especially on edge devices) require domain-specific accelerators

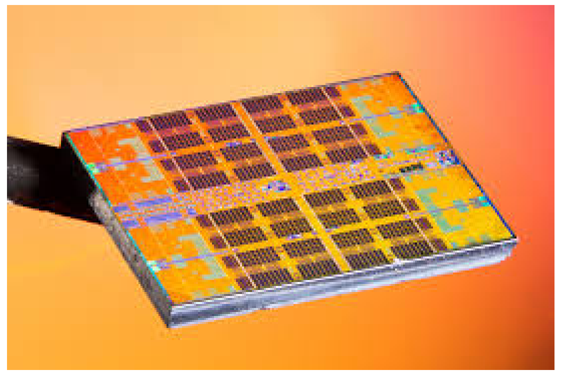
Motivations

- Existing accelerators start from specific models (e.g., CNNs) or only try to accelerate specific computational patterns
  - Designing hardware accelerators by hand is complex and time-consuming
  - Hardware designers may want to explore different design trade-offs, depending on the application requirements
- Agile Hardware Design and Prototyping is required
  - Quickly transition from algorithm formulation to accelerator implementation
  - Sufficient design space exploration knobs
  - Minimal human interaction

LeNet architecture



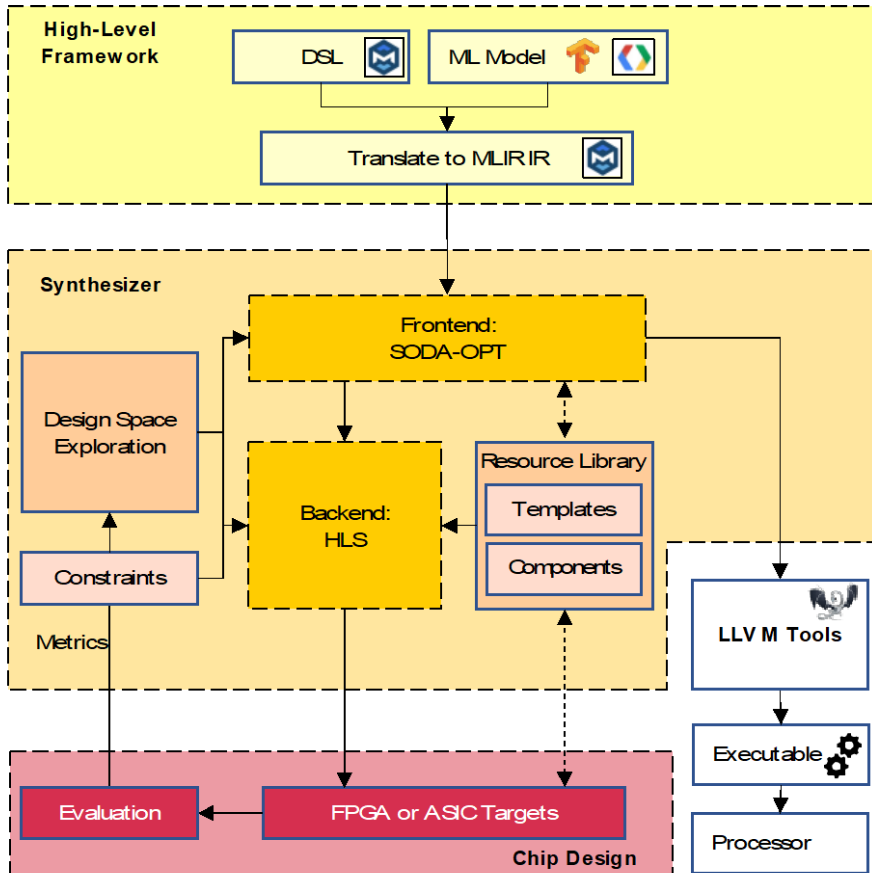




ASIC

Pacific Northwest NATIONAL LABORATORY

Our solution: the SODA Synthesizer



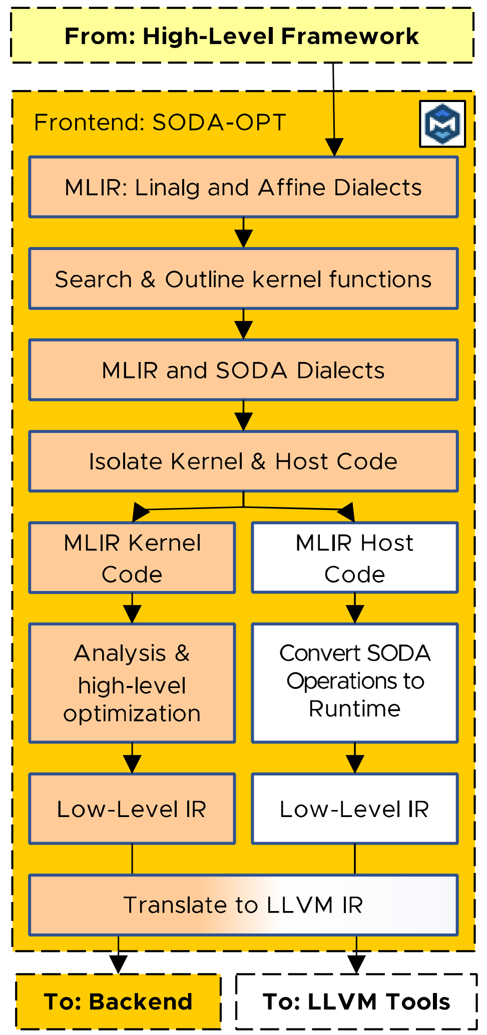
[J. Zhang, N. Bohm Agostini, S. Song, C. Tan, A. Limaye, V. Amatya, J. B. Manzano, M. Minutoli, V. G. Castellana, A. Tumeo, G. Wei, D. Brooks: Towards Automatic and Agile AI/ML Accelerator Design with End-to-End Synthesis. ASAP 2021: 218-225]

[N. Bohm Agostini, S. Curzel, J.Zhang, A. Limaye, C. Tan, V. Amatya, M. Minutoli, V. G. Castellana, J. B. Manzano , D. Brooks, G. Wei, A. Tumeo: Bridging Python to Silicon: The SODA Toolchain. To appear in IEEE Micro Magazine]

- The SODA Synthesizer is 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

Frontend: SODA-OPT

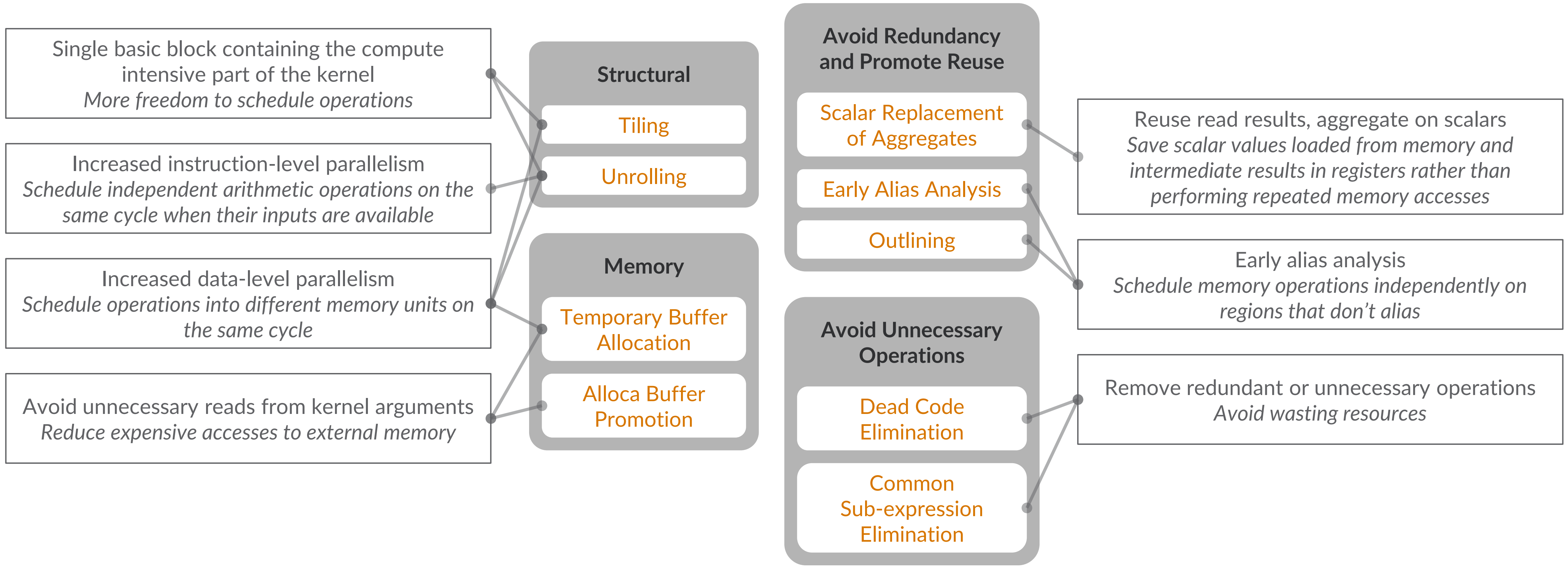
- **SODA-OPT: S**earch, **O**utline, **D**ispatch, **A**ccelerate frontend **opt**imizer
- Employs and embraces the MLIR framework
  - ✓ MLIR: Multi-Level Intermediate Representation
  - ✓ Used in TensorFlow, TFRT, ONNX-MLIR, others
- Uses MLIR and compiler passes to:
  - ✓ **Identify code regions** for hardware generation
  - ✓ Perform **high-level optimizations** (dataflow transformations, data-level and instruction-level parallelism extraction)
  - ✓ Generate **interfacing code** and runtime calls for microcontroller


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

[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]

Frontend: SODA-OPT

- SODA-OPT implements optimizations as compiler passes



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

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

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

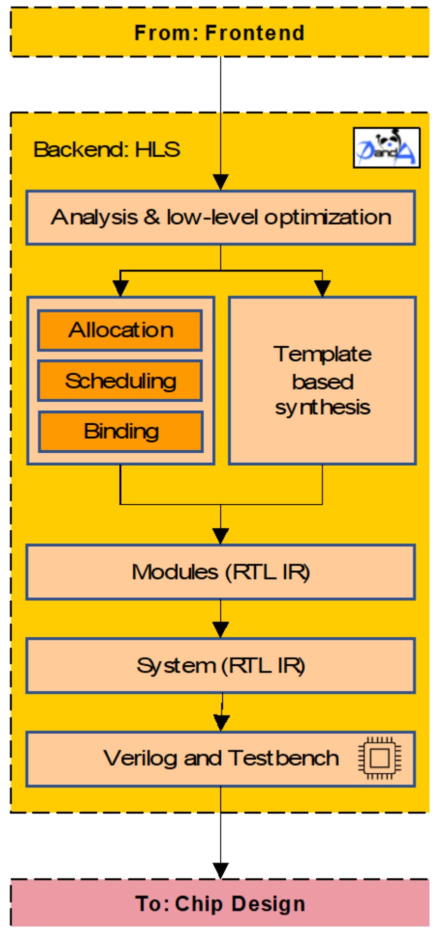
Early alias analysis
Schedule memory operations independently on regions that don't alias

Remove redundant or unnecessary operations
Avoid wasting resources

Pacific Northwest NATIONAL LABORATORY

Backend: High-Level Synthesis

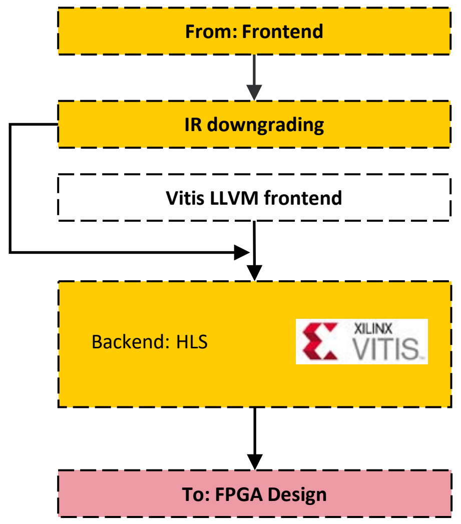
- The synthesizer **backend** takes as input the optimized low-level IR and generates the hardware descriptions of the accelerators
- The main HLS backend is PandA-Bambu, an open-source state-state-of-the-art high-level synthesis (HLS) tool
  - ✓ We are key contributors to Bambu, with **parallel accelerator designs**, **modular HLS**, and **ASIC support**
  - ✓ Automated testing and verification



[F. Ferrandi, V. G. Castellana, S. Curzel, P. Fezzardi, M. Fiorito, M. Lattuada, M. Minutoli, C. Pilato, A. Tumeo: Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications. DAC 2021: 1327-1330]

Backend: High-Level Synthesis

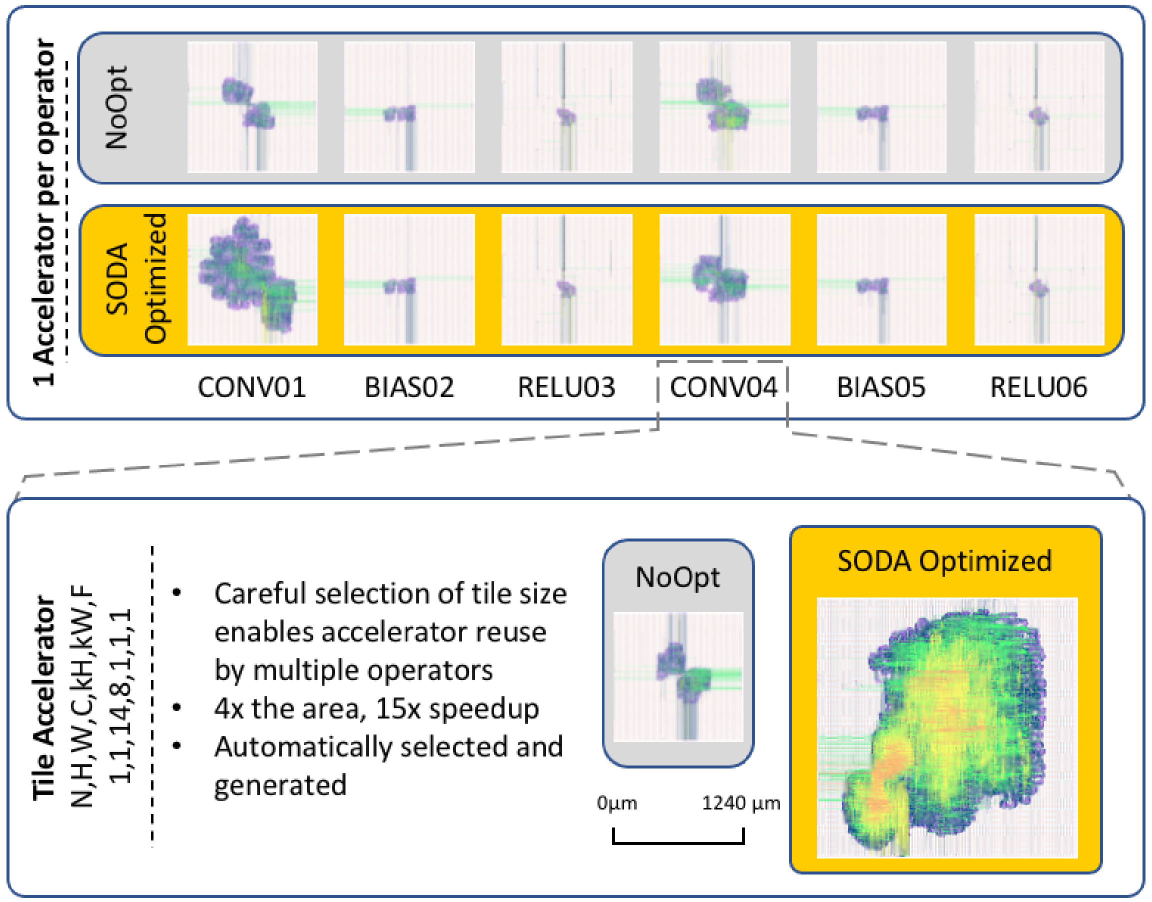
- We also support integration with Xilinx Vitis HLS through its open-source LLVM frontend
- The SODA Synthesizer has interfaces with multiple open-source and commercial backends
  - ✓ Xilinx Vivado, Intel Quartus (FPGA)
  - ✓ OpenROAD, Synopsys Design Compiler (ASIC)
- Automated path to FPGA bitstream or GDS2 files



[https://github.com/Xilinx/HLS]

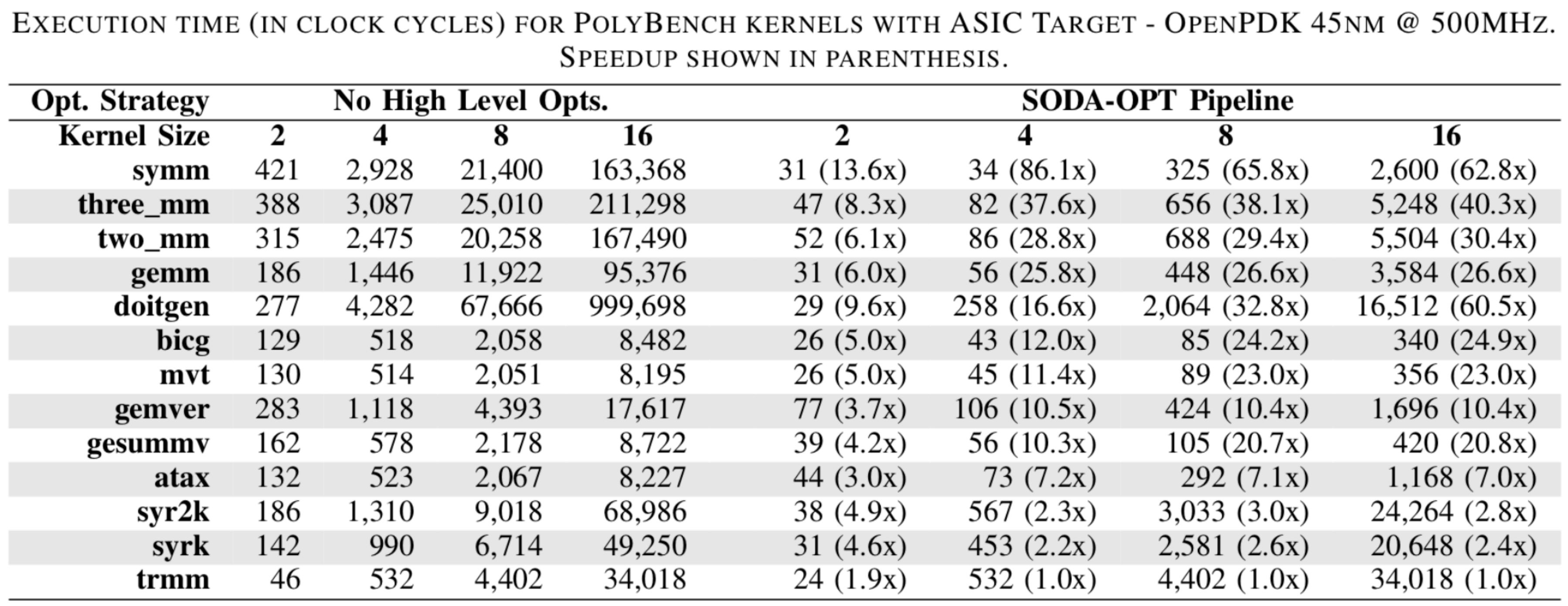
Examples of generated accelerators

- LeNet model imported from TensorFlow
- Each operator is synthesized to an **ASIC accelerator** (OpenROAD FreePDK 45nm)
- SODA-OPT optimized accelerators are bigger, but also much **faster**



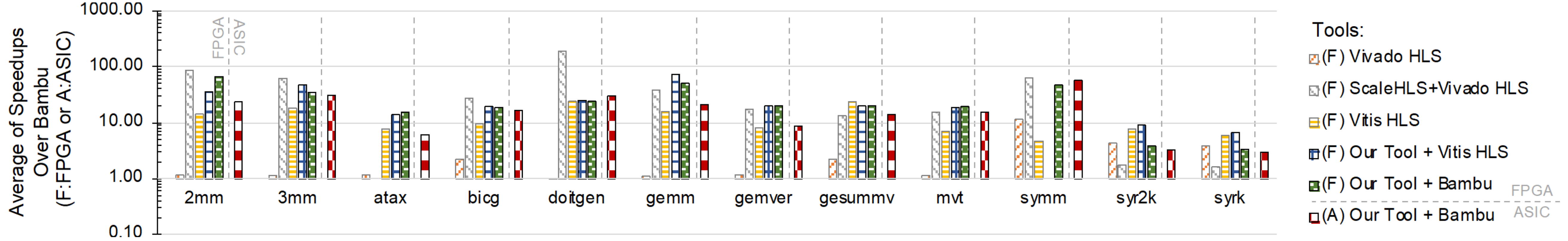
Examples of generated accelerators

- 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]

Conclusions

- The SODA toolchain provides an **end-to-end compiler-based** design flow from the formulation of an algorithm to the deployment of custom hardware accelerators
  - Multi-level, modular, and extensible
  - Promotes **agile hardware design**
  - Based on open-source technologies, and integrated with proprietary tools
- Start using SODA today with these links:

SODA-OPT

Panda-Bambu HLS (v 0.9.7)

SODA Docker Image

SODA Tutorial: *DATE 2022*