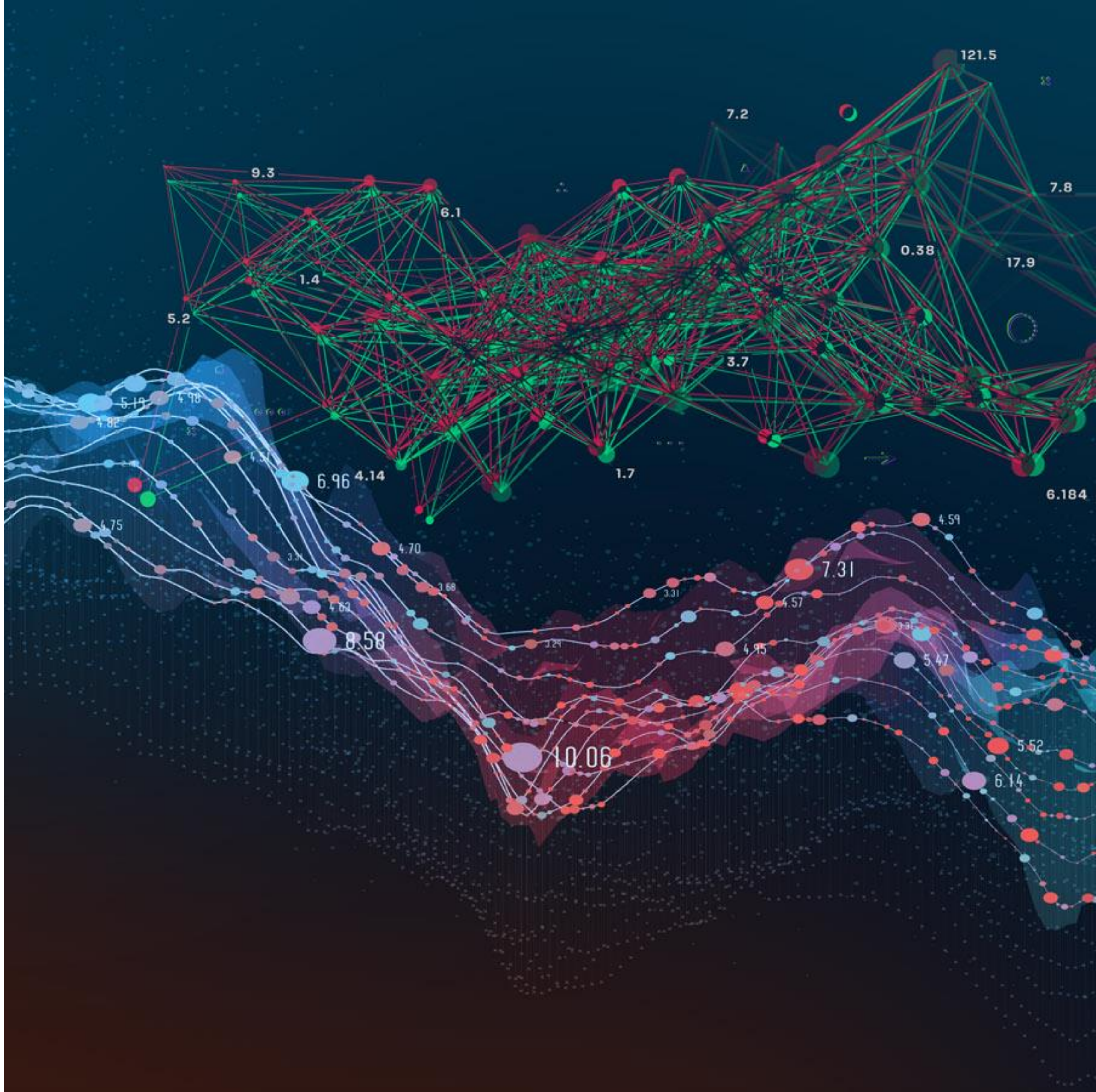
Pacific Northwest NATIONAL LABORATORY

Agile Hardware Design for Complex Data Science Applications: Opportunities and Challenges

March 21, 2022

Antonino Tumeo



Tutorial Presenters

Antonino Tumeo

Vito Giovanni Castellana

Serena Curzel

Fabrizio Ferrandi

Michele Fiorito

POLITECNICO MILANO 1863

Marco Minutoli

Nicolas Bohm Agostini

Tutorial Schedule

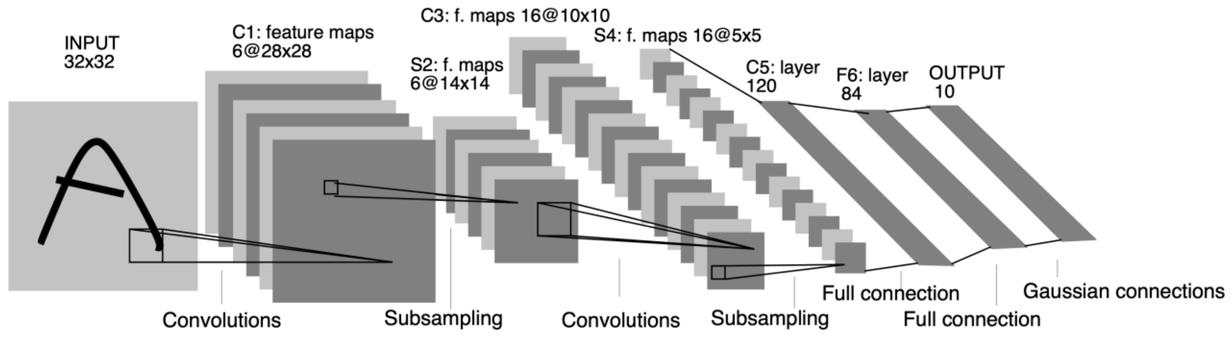
Time	Presenter	Title
13:15 - 13:45	Antonino Tumeo	Agile Hardware Design for Complex Data Science Applications: Opportunities and Challenges
13:45 - 14:15	Fabrizio Ferrandi	Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications
14:15 - 15:00	Serena Curzel	Hands-on: Productive High-Level Synthesis with Bambu
15:00 - 15:15		Coffee Break
15:15 - 16:00	Michele Fiorito	Hands-on: Compiler Based Optimizations, Tuning and Customization of Generated Accelerators
16:00 - 16:45	Nicolas Bohm Agostini	Hands-on: SODA-OPT: Enabling System-Level Design in MLIR for High-Level Synthesis and Beyond
16:45 - 17:15	Marco Minutoli Vito Giovanni Castellana	Technical presentation and Hands-on: Svelto: High-Level Synthesis of Multi-Threaded Accelerators for Graph Analytics

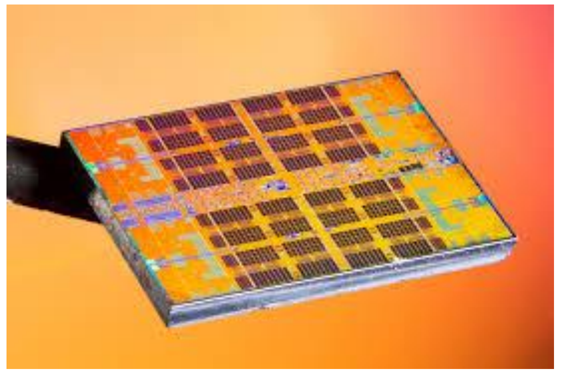
Motivations

- 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





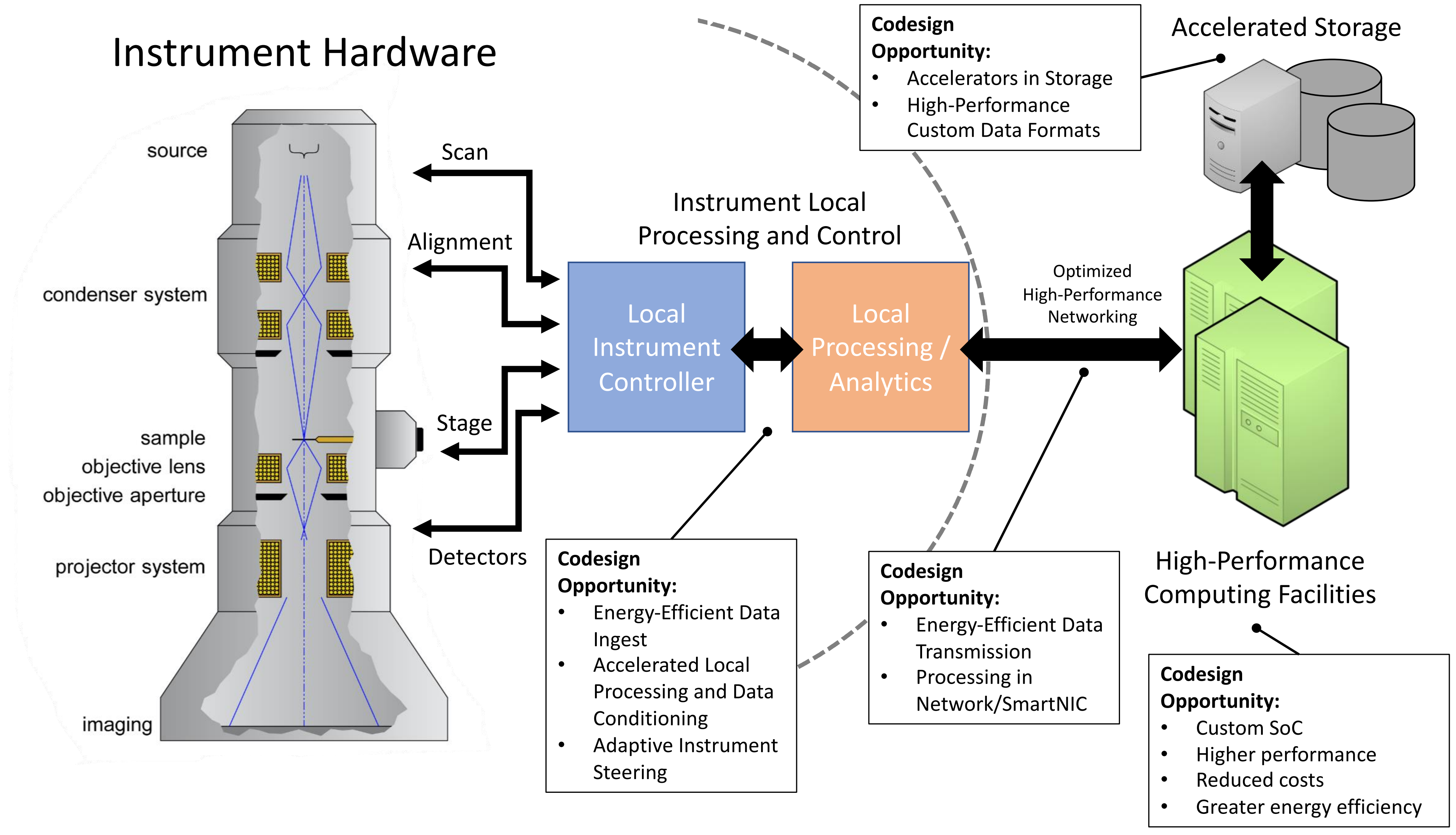
Why Data Science?

- Increasingly complex data analysis pipelines
- May include algorithms with significantly different behaviors
  - Deep neural networks, graph analytics, graph representation learning…
- Algorithmic research in the area is quickly evolving
- Algorithms are data-intensive
  - Significant amount of data per computation
- Some algorithms exhibit irregular behaviors
  - Graph algorithms are the prototypical irregular kernel

Possible Applications

- Inference in the cloud (Brainwave, Bing, Alibaba, Amazon…)
- High-performance computing
  - Converged applications (Scientific simulation, machine learning, and graph analytics)
  - Near data / near network data analysis
- Autonomous systems
  - Low latency reasoning for decision making
  - Federated learning

Possible Applications: Experimental Scientific Workflows


Instrument Hardware
source
condenser system
sample
objective lens
objective aperture
projector system
imaging
Scan
Alignment
Stage
Detectors
Instrument Local Processing and Control
Local Instrument Controller
Local Processing / Analytics
Optimized High-Performance Networking
Accelerated Storage
High-Performance Computing Facilities

Codesign Opportunity:
- Accelerators in Storage
- High-Performance Custom Data Formats

Codesign Opportunity:
- Energy-Efficient Data Ingest
- Accelerated Local Processing and Data Conditioning
- Adaptive Instrument Steering

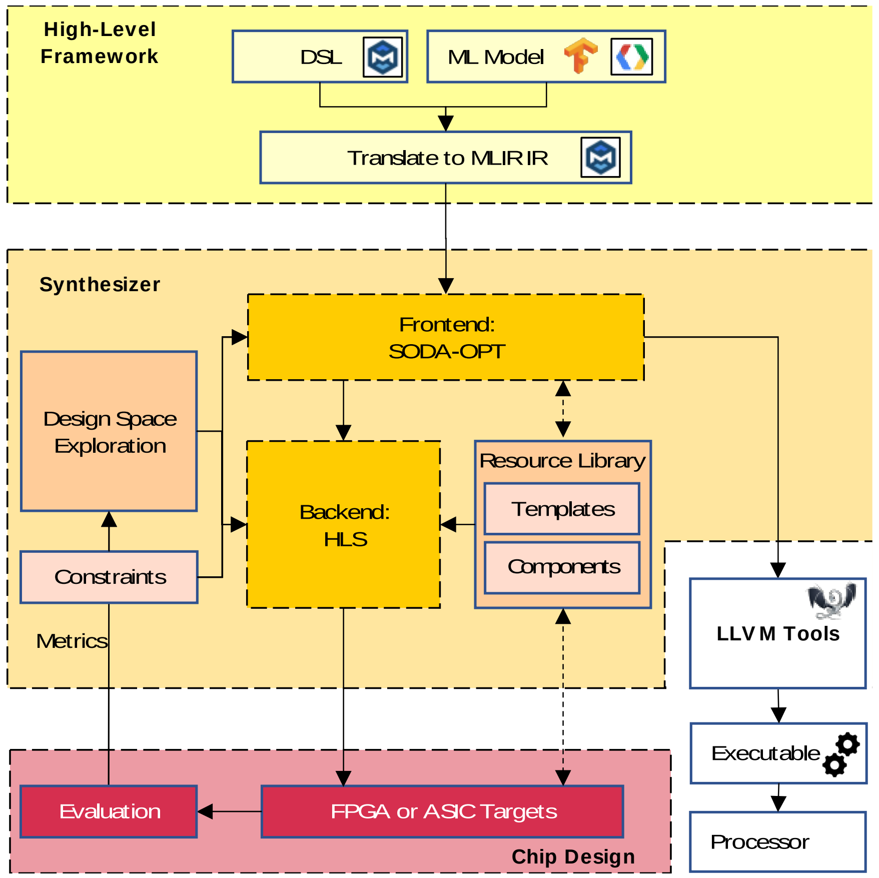
Codesign Opportunity:
- Energy-Efficient Data Transmission
- Processing in Network/SmartNIC

Codesign Opportunity:
- Custom SoC
- Higher performance
- Reduced costs
- Greater energy efficiency

Challenges

- Need to go from the high-level data science frameworks to the hardware implementation
  - Tension between domain specificity and generality applies to both hardware and the hardware generators/tools
  - Python frameworks typically based on tensor representations
    - ✓ What about graphs and sparse data structures?
- Generating only the accelerators is not sufficient, we need to consider the system level implications
- Many levels, many tools, often not directly interoperating
- Verification and testing
- Many efforts to reduce costs to access tools and IPs, but still a long road

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]

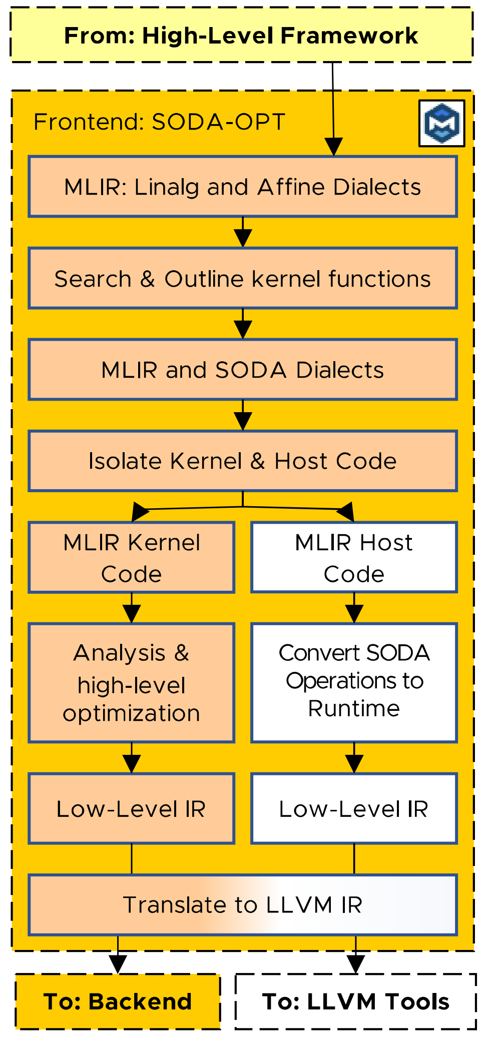
[Jeff Jun Zhang, Nicolas Bohm Agostini, Shihao Song, Cheng Tan, Ankur Limaye, Vinay Amatya, Joseph B. Manzano, Marco Minutoli, Vito Giovanni Castellana, Antonino Tumeo, Gu-Yeon Wei, David Brooks: Towards Automatic and Agile AI/ML Accelerator Design with End-to-End Synthesis. ASAP 2021: 218-225]

- 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, as well as a Coarse-Grained Reconfigurable Array (CGRA) generator
- 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

Pacific Northwest NATIONAL LABORATORY

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


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

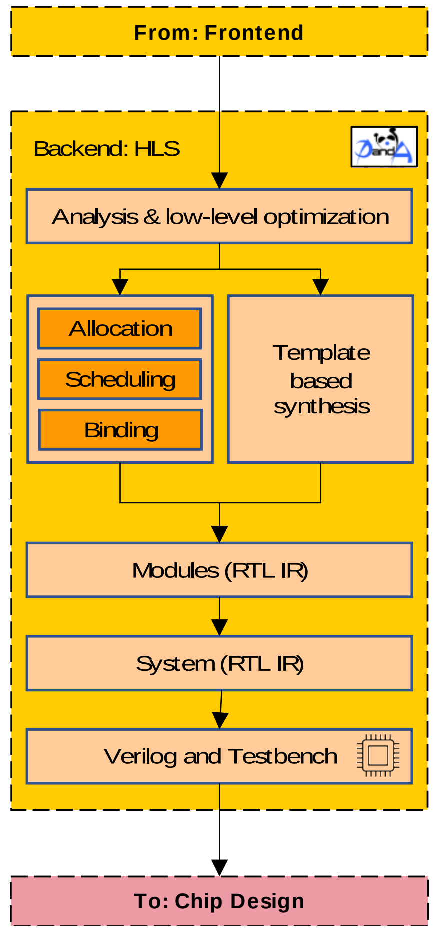
SODA-OPT: System Overview

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

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
- The HLS backend is PandA-Bambu, an open-source state-state-of-the-art high-level synthesis (HLS)
  - Key features: **parallel accelerator designs**, **modular HLS**, and **ASIC support**
- The HLS backend provides automated testing and verification of the generated designs



Bambu HLS

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]

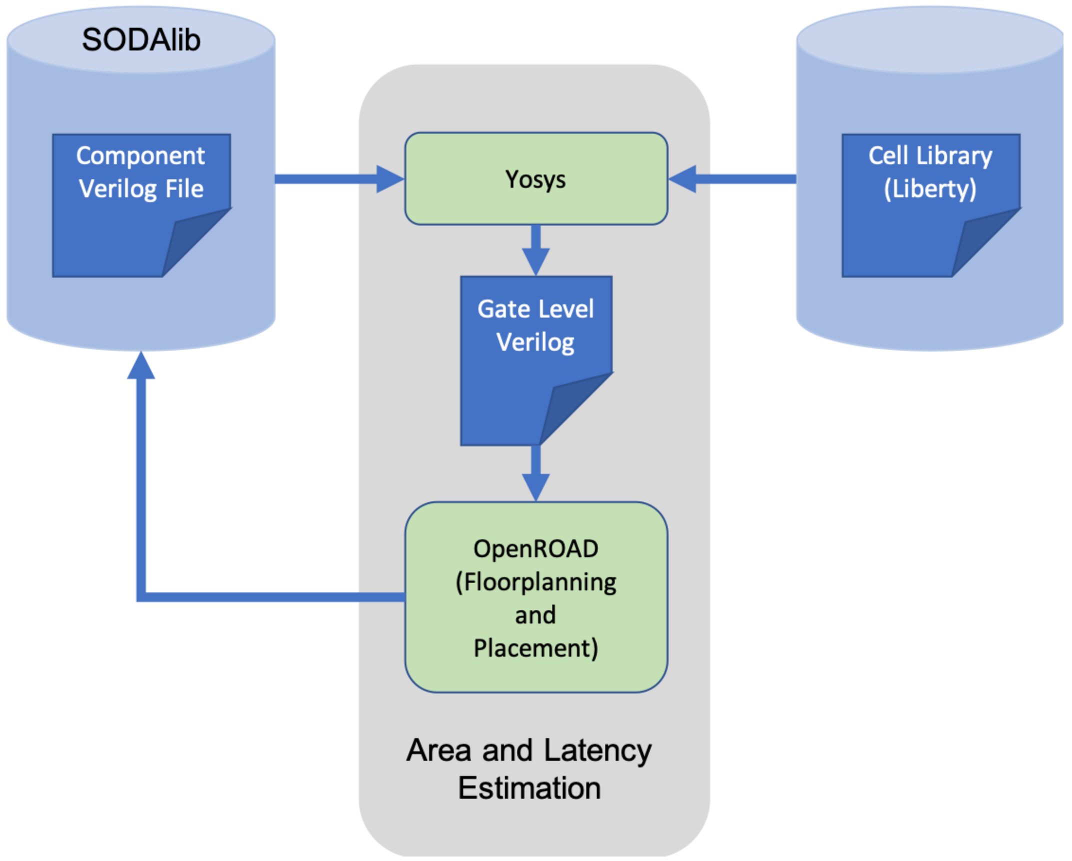
SODA Synthesizer: ASIC targets

- 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)

- Backends' resources characterized for the target technology:
  - **HLS Backend: 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)

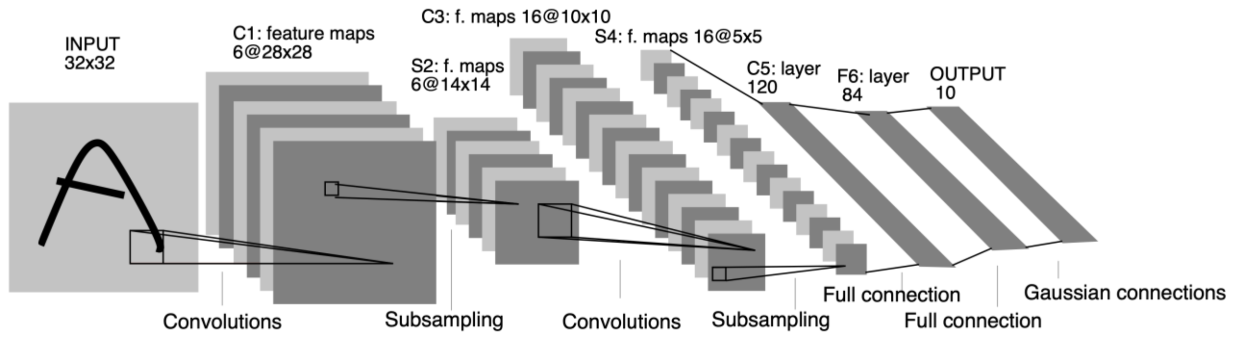


SODA characterization flow. The characterization flow can be extended to synthesize HLS generated designs, or used to estimate their area-latency-power profiles to drive the Design Space Exploration engine

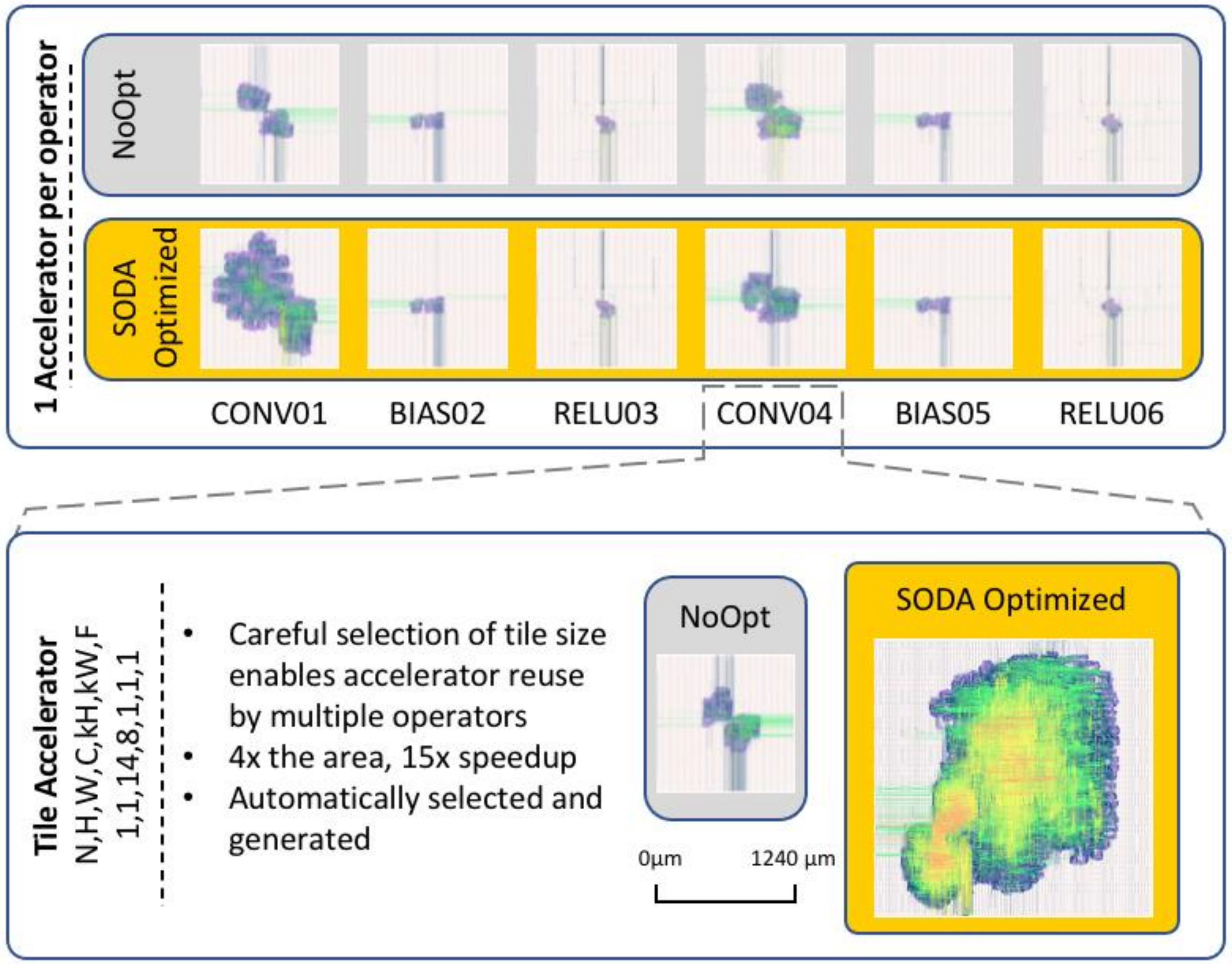
OpenROAD
https://theopenroadproject.org

From Python to optimized ASIC

LeNet architecture from the original paper

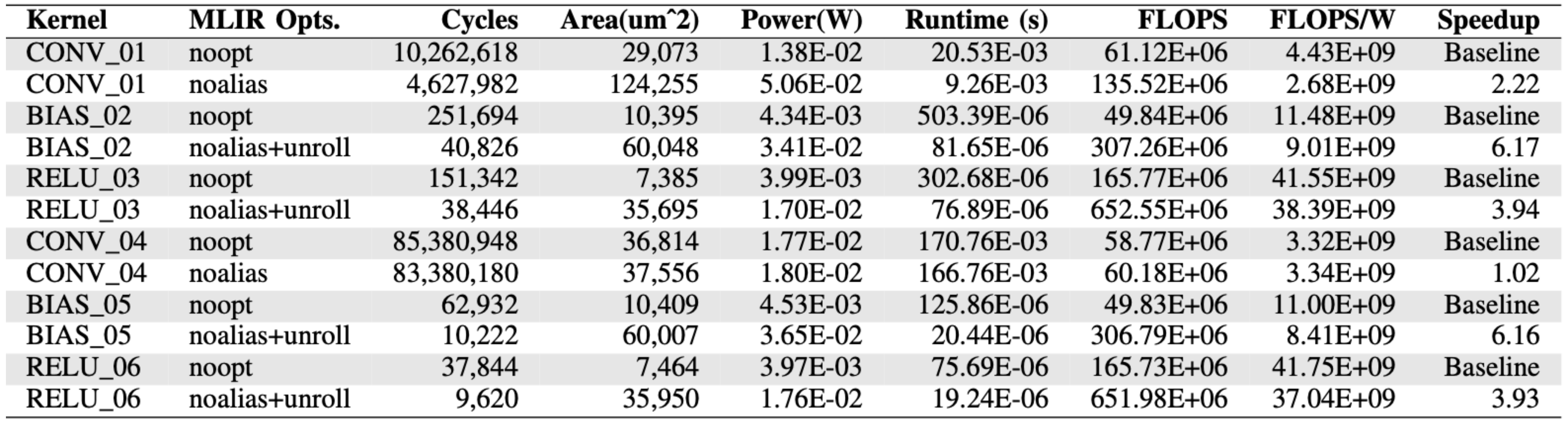


- LeNet example
- Each of the operator is synthesized to an **ASIC accelerator**
- SODA optimized accelerators are bigger, but also much **faster**



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

Detailed Results of the Optimization Process



Kernel	MLIR Opts.	Cycles	Area(um^2)	Power(W)	Runtime (s)	FLOPS	FLOPS/W	Speedup
CONV_01	noopt	10,262,618	29,073	1.38E-02	20.53E-03	61.12E+06	4.43E+09	Baseline
CONV_01	noalias	4,627,982	124,255	5.06E-02	9.26E-03	135.52E+06	2.68E+09	2.22
BIAS_02	noopt	251,694	10,395	4.34E-03	503.39E-06	49.84E+06	11.48E+09	Baseline
BIAS_02	noalias+unroll	40,826	60,048	3.41E-02	81.65E-06	307.26E+06	9.01E+09	6.17
RELU_03	noopt	151,342	7,385	3.99E-03	302.68E-06	165.77E+06	41.55E+09	Baseline
RELU_03	noalias+unroll	38,446	35,695	1.70E-02	76.89E-06	652.55E+06	38.39E+09	3.94
CONV_04	noopt	85,380,948	36,814	1.77E-02	170.76E-03	58.77E+06	3.32E+09	Baseline
CONV_04	noalias	83,380,180	37,556	1.80E-02	166.76E-03	60.18E+06	3.34E+09	1.02
BIAS_05	noopt	62,932	10,409	4.53E-03	125.86E-06	49.83E+06	11.00E+09	Baseline
BIAS_05	noalias+unroll	10,222	60,007	3.65E-02	20.44E-06	306.79E+06	8.41E+09	6.16
RELU_06	noopt	37,844	7,464	3.97E-03	75.69E-06	165.73E+06	41.75E+09	Baseline
RELU_06	noalias+unroll	9,620	35,950	1.76E-02	19.24E-06	651.98E+06	37.04E+09	3.93

- OpenRoad with OpenPKD 45 nm, targeting 500 MHz
- Significant speed ups in almost all cases
- Efficiency (FLOPS/W) may reduce with the optimized designs because of increased power consumption and area, however performance also increases

Other Approaches (an incomplete list)

- A multitude of hand-designed domain specific accelerators
  - Trying to recover some levels of flexibility either using extensible Instruction Set Architectures (e.g., RISC-V) or novel reconfigurable designs
- Approaches that generate custom hardware from Python framework mapping on parametric templates
  - GEMMINI: parametric template
  - VeriGOOD-ML: compiler maps on three different architectures
  - VTA: specialized coprocessor (GEMM unit) generated with HLS
- Convert code to imperative languages (C/C++) annotated for HLS
  - PyLog: Python to C/C++ for Vivado HLS
  - HeteroCL: partitions code between CPU and FPGA, provides a library of functions to insert hardware-specific information in the source code, generates C/C++ for HLS
  - ScaleHLS: MLIR to annotate C/C++ for Vivado HLS
- CIRCT (Circuit IR Compilers and Tools): MLIR to build interoperable tools for hardware design

Some Opportunities

- Circuit-level Intermediate Representations
  - To enable hardware level transformation, modularity, generation of better RTL code
- Profile-driven synthesis
  - Interface with instrumentation and profiling tools; profile on the host, employ the information for the synthesis
  - Especially for data science: many data-intensive, data-dependent algorithms
- Memory-centric optimizations
  - Data-intensive algorithms focused at how data are accessed and moved rather than on the compute
- AI-driven design space exploration
  - Compilers facilitate development of estimation models

Conclusions

- Discussed the benefit of end-to-end synthesis tools for data science application
- Introduced the SODA Synthesizer, a modular, multilevel, extensible, and opensource hardware compiler
  - Composed of a high-level compiler and an HLS tool
  - Supports FPGAs and ASIC
- Discussed the SODA framework in the context of Data Science Applications as a tool for agile hardware development
  - And some of the other ongoing research in the area
- Discussed challenges and opportunities for hardware generators and compiler-based design tools
- The next presentation will specifically focus on High-Level Synthesis, and we will then dive into the hands-on Bambu and SODA-Opt

Pacific Northwest NATIONAL LABORATORY

Thank you