



fast machine learning  
for science

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

Tutorial: SODA Synthesizer: Accelerating Artificial Intelligence  
Applications with an End-to-End Silicon Compiler

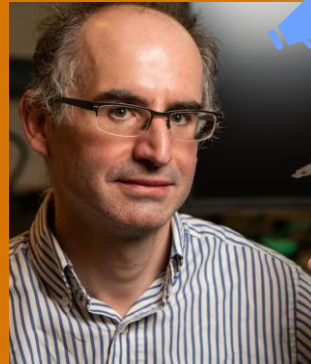
September 1, 2025  
**Antonino Tumeo**  
Chief Scientist



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



# Tutorial Team and Presenters



**Antonino Tumeo**



**Fabrizio Ferrandi**



**Nicolas  
Bohm Agostini**



**Vito Giovanni  
Castellana**



**Serena Curzel**



**Michele Fiorito**



**Giovanni Gozzi**

# Tutorial Schedule

9:00 – 9:30	Antonino Tumeo	Agile Hardware Design for Complex Data Science Applications: Opportunities and Challenges.
9:30 – 10:00	Fabrizio Ferrandi	Bambu: An Open-Source Research Framework for the High-Level Synthesis of Complex Applications.
10:00 – 10:30	Antonino Tumeo	SODA-OPT: Enabling System-Level Design in MLIR for HLS and Beyond. Hands-on: From DNN Models to ASIC Devices with SODA-OPT
10:30 – 11:00		Coffee Break
11:00 – 12:00	Serena Curzel Michele Fiorito	Hands-on: Productive High-Level Synthesis with Bambu, Compiler Based Optimizations, Tuning and Customization of Generated Accelerators
12:00 - 12:30	Antonino Tumeo & Fabrizio Ferrandi	New features in SODA-OPT and Bambu

## Pre-Tutorial Survey

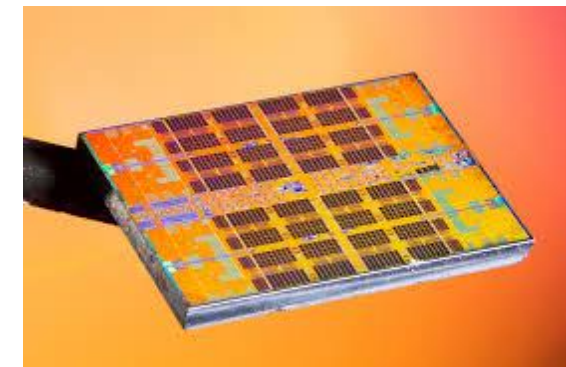
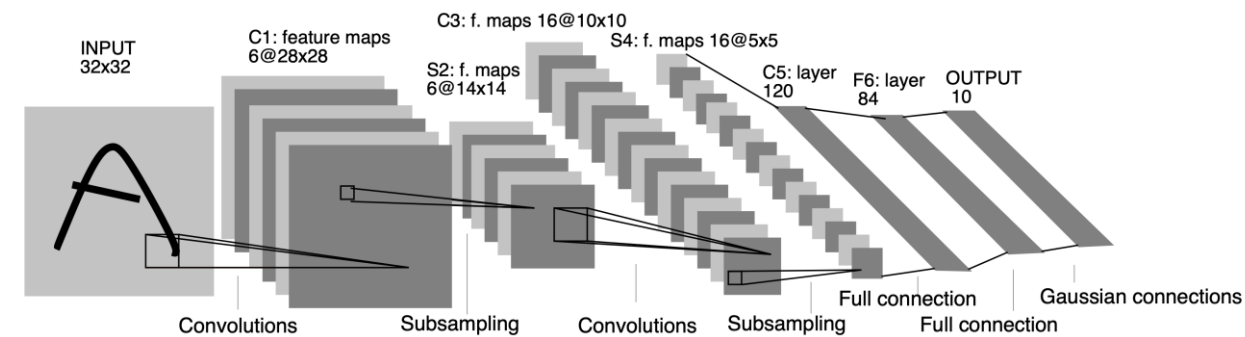
Please join the survey to let us improve the presentation



# 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



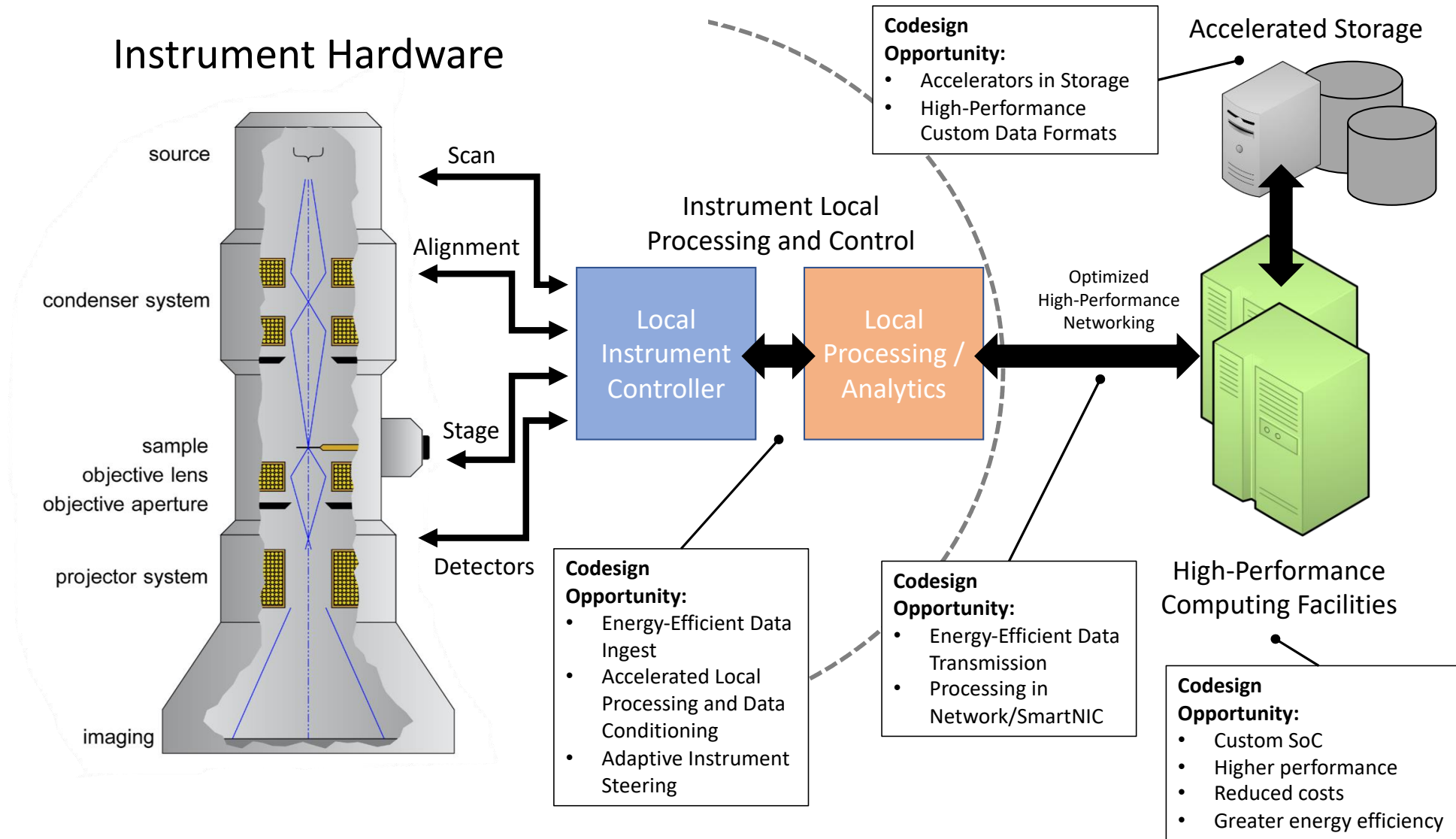
# Why Data Science?

- 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

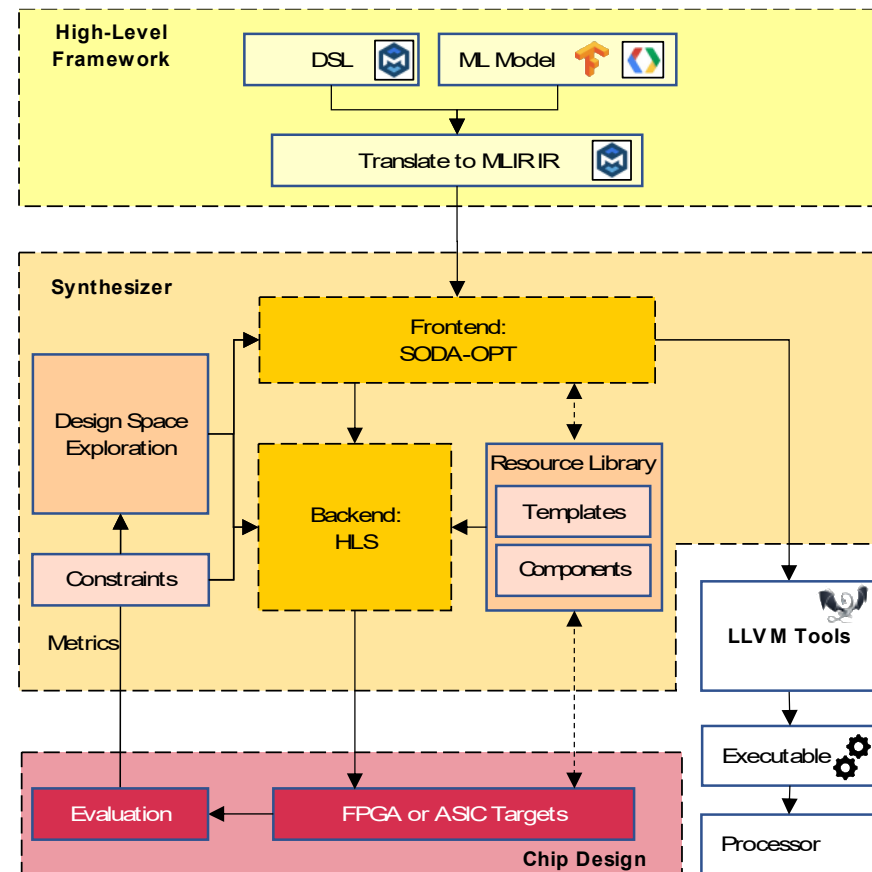
# Possible Applications: Experimental Scientific Workflows



# 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



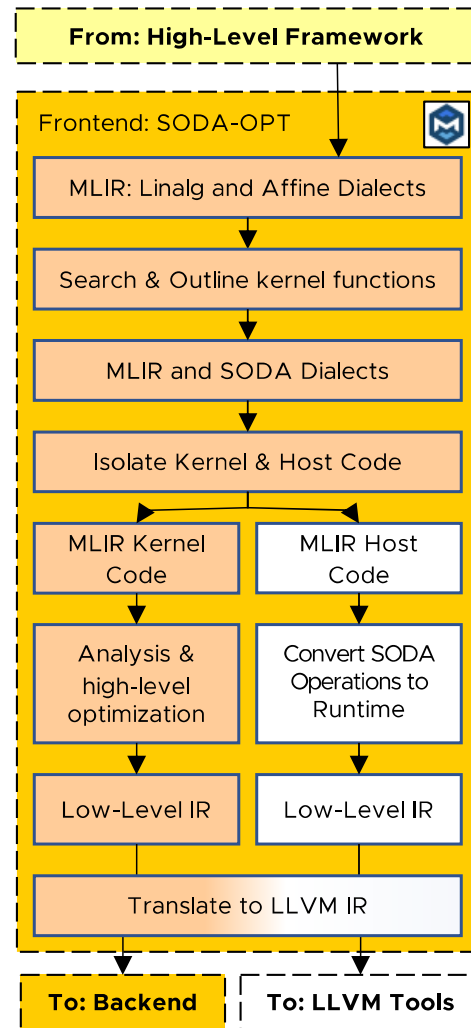
- A modular, multi-level, interoperable, extensible, **open-source hardware compiler** from **high-level programming frameworks to silicon**
- Compiler-based frontend, leveraging the MultiLevel Intermediate Representation (MLIR)
- **Compiler-based backend**, leveraging state-of-the-art High-Level Synthesis (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**

[M. Minutoli, V. G. Castellana, C. Tan, J. Manzano, V. Amaty, A. Tumeo, D. Brooks, G-Y. Wei: SODA: a New Synthesis Infrastructure for Agile Hardware Design of Machine Learning Accelerators. ICCAD 2020: 98:1-98:7]

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

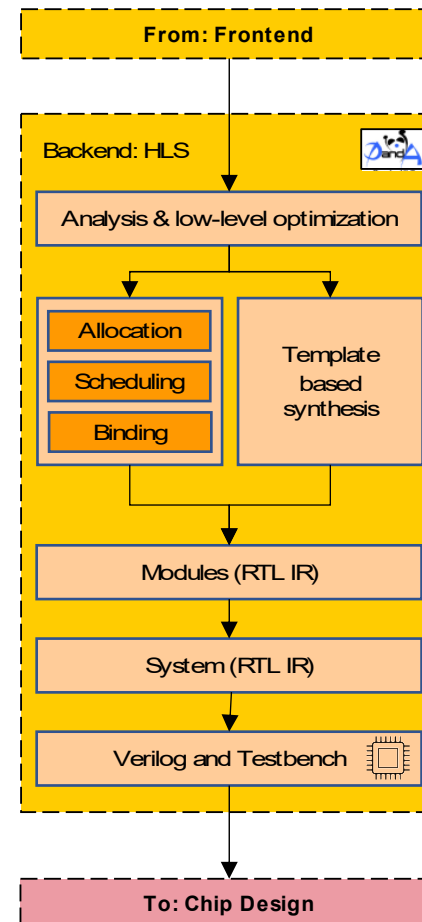
[N. Bohm Agostini, S. Curzel, J. Zhang, A. Limaye, C. Tan, V. Amaty, M. Minutoli, V.G. Castellana, J. Manzano, A. Tumeo: Bridging Python to Silicon: The SODA Toolchain. IEEE Micro Magazine 2022]  
 [N. Bohm Agostini, S. Curzel, V. Amaty, C. Tan, M. Minutoli, V. G. Castellana, J. Manzano, D. Kaeli, A. Tumeo : An MLIR-based Compiler Flow for System-Level Design and Hardware Acceleration. ICCAD 22]  
 [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: Components

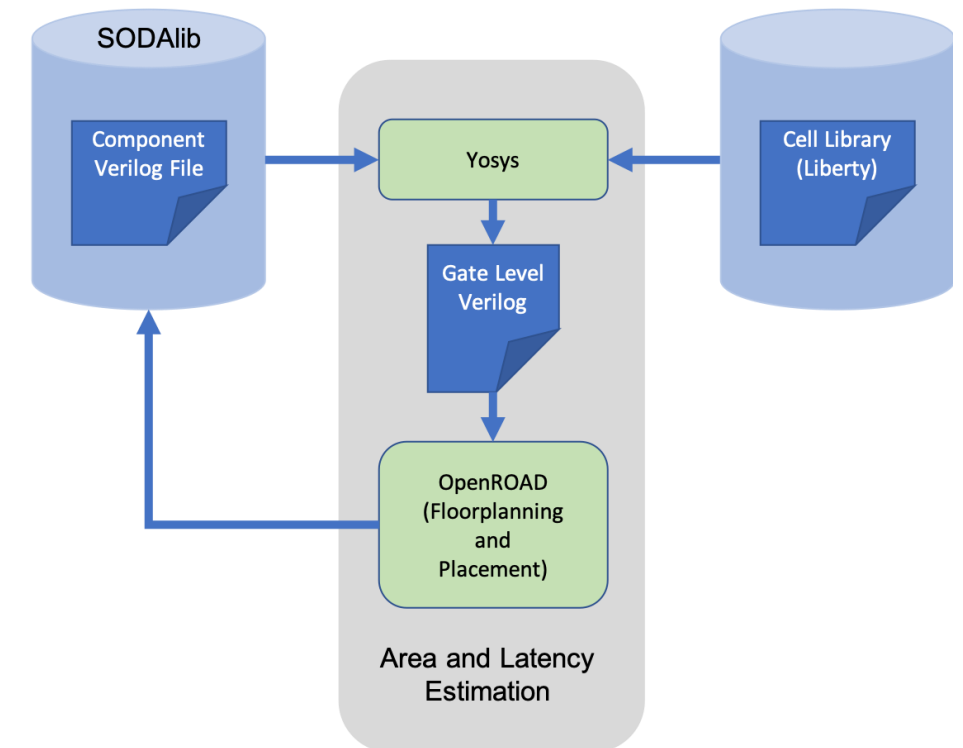


SODA-OPT

<https://github.com/pnnl/soda-opt>



<https://panda.dei.polimi.it>



OpenROAD

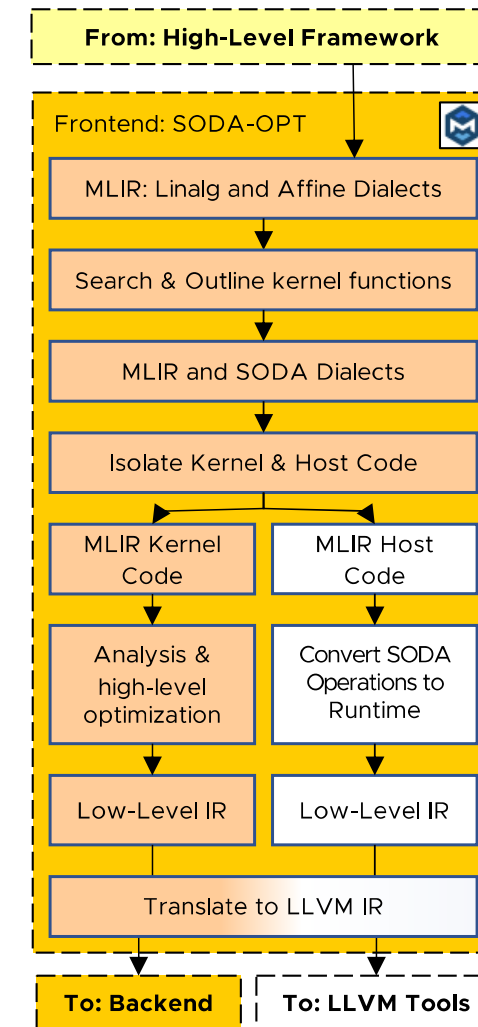
<https://theopenroadproject.org>

# 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
  - 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, S. Curzel, V. 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. ICCAD 22]

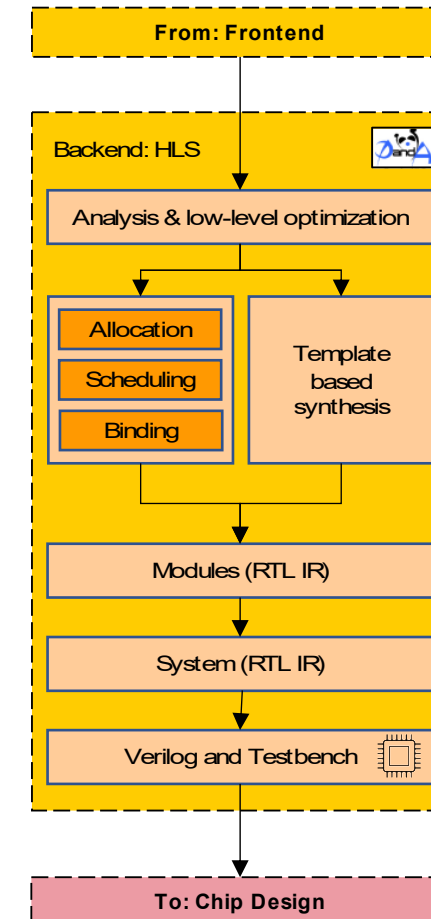
[N. Bohm Agostini, S. Curzel, D. Kaeli, A. Tumeo: SODA-OPT an MLIR based flow for co-design and high-level synthesis. CF 2022: 201-202 - **Best Poster Award.**]



SODA-OPT: System Overview

# 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-of-the-art High-Level Synthesis (HLS) framework
  - Key features: **parallel accelerator designs**, **modular HLS**, and **ASIC support**
- The HLS backend provides automated testing and verification of the generated designs

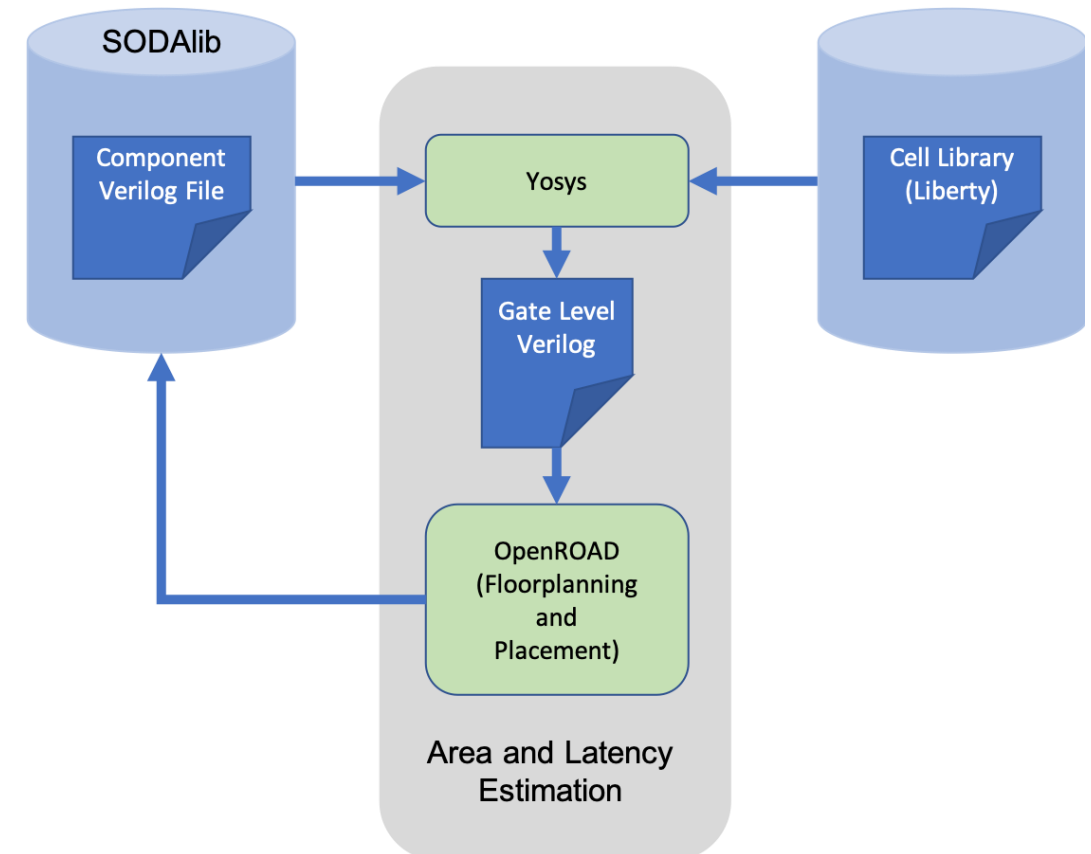


<https://panda.dei.polimi.it>

[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]

# 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
- SODA also supports 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 interfaces with 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

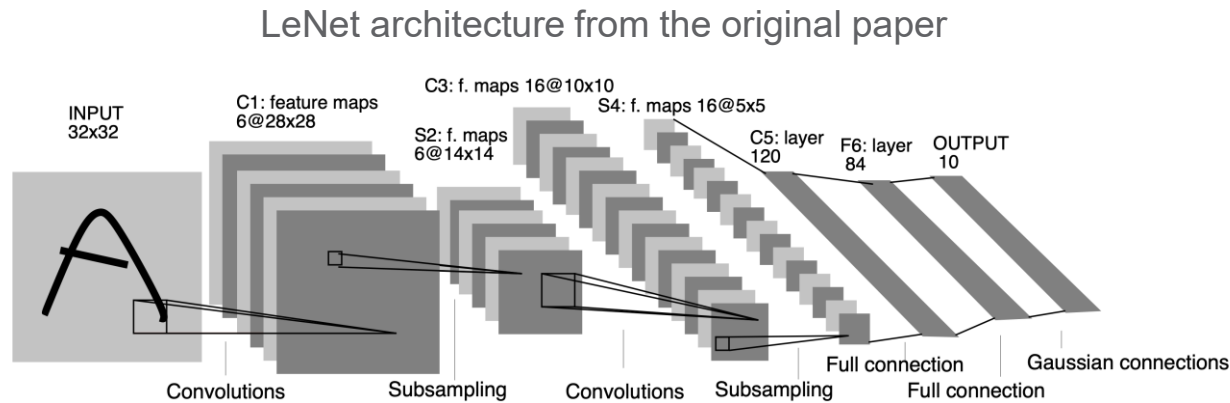
## OpenROAD

<https://theopenroadproject.org>

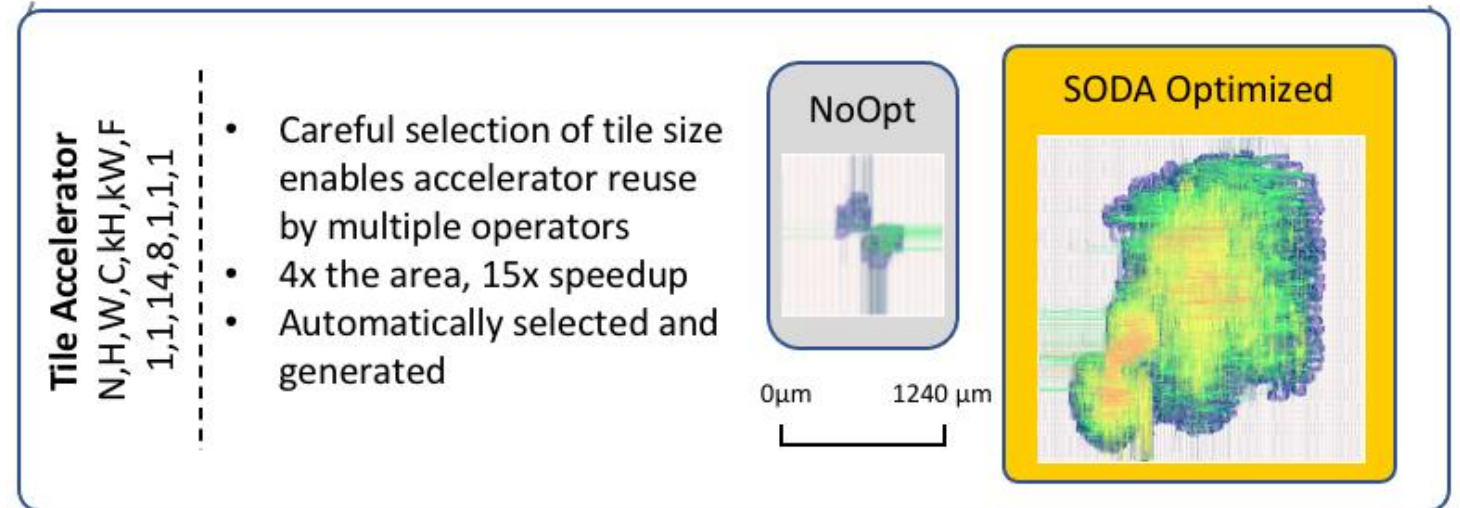
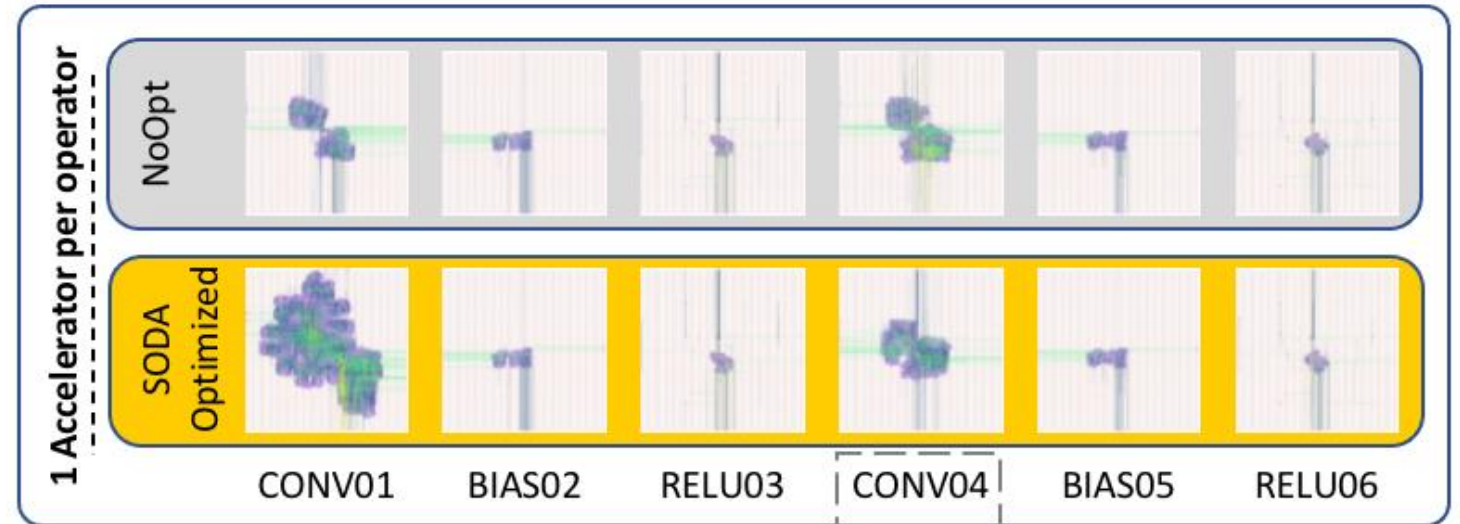
# Why an Opensource High-Level Synthesis Toolchain for High Performance Computing?

- HPC workflows are evolving rapidly and becoming more complex
- Heterogeneous HPC systems are here to stay, domain-specific accelerators are the only way forward
- Conventional HLS tools leverage C/C++ annotated with hardware and tool specific pragmas
  - Developer still need to do significant optimizations by hand, the productivity gap for writing hardware description languages becomes a productivity gap for writing highly specialized C/C++
- We need tools that democratize hardware design **AND** allow to perform research of new methodologies to address new applications and platforms

# From Python to optimized ASIC



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



# From Python to optimized FPGA designs

Opt. Strategy	FPGA Target: Xilinx xc7vx690t-ffg1930-3 @ 100MHz								
	No High Level Opts.				OUR TOOL Pipeline				Avg. speedup
	Kernel Size	2	4	8	16	2	4	8	
<i>doitgen</i>	165	2,486	38,986	344,338	15	166	1,158	9,264 (F8)	24.2x
<i>three_mm</i>	220	1,743	14,042	111,410	22	40	320 (F4)	2,560 (F4)	35.2x
<i>two_mm</i>	176	1,375	11,218	87,842	25	43	98	784 (F8)	66.4x
<i>gemm</i>	103	794	6,538	42,514	16	28	71	568 (F8)	50.4x
<i>bicg</i>	73	294	1,162	4,626	13	28	45	141	18.7x
<i>mvt</i>	74	290	1,155	4,611	13	21	45	141	19.5x
<i>gesummv</i>	92	326	1,226	5,026	18	25	46	142	20.0x
<i>gemver</i>	154	606	2,377	9,620	36	49	75	300 (F8)	20.1x
<i>atax</i>	76	299	1,171	4,643	21	34	60	157	15.4x
<i>syr2k</i>	99	706	4,834	35,650	19	270	1,417	8,835	3.8x
<i>syrk</i>	79	546	3,682	22,594	16	222	1,211	7,896	3.3x
<i>symm</i>	92	729	5,570	42,754	19	526	3,338	22,610	2.4x
<i>trmm</i>	26	301	2,138	15,730	12	288	2,138	15,730	1.3x

Opt. Strategy	FPGA Target: Xilinx xc7vx690t-3ffg1930-VVD @ 100MHz											
	No HL Opts. [#LUTs]				LUTs Ratio				Speedup / LUTs Ratio			
	Kernel Size	2	4	8	16	2	4	8	16	2	4	8
<i>symm</i>	2802	2645	4356	1501	0.63	1.57	13.04	37.83*	24.28	56.22	3.37	1.12
<i>doitgen</i>	2300	3496	2824	2382	1.81	3.14	22.75	26.98*	6.09	4.76	1.48	1.38
<i>three_mm</i>	3122	7108	7696	2014	1.68	4.07	30.05*	114.84*	5.95	10.71	1.46	0.38
<i>two_mm</i>	2917	5700	5275	1641	1.09	3.40	20.95	67.35*	6.45	9.41	5.46	1.66
<i>gemm</i>	1929	3385	2693	1005	1.32	3.51	22.22	59.55*	4.89	8.09	4.14	1.26
<i>bicg</i>	1370	1642	2614	1027	1.68	3.55	5.92	40.52*	3.34	2.96	4.36	0.81
<i>mvt</i>	1464	3893	2966	5461	1.57	1.47	5.23	7.78	3.63	9.39	4.91	4.20
<i>gesummv</i>	1690	1838	2883	1197	1.56	3.38	5.27	35.45	3.29	3.85	5.05	1.00
<i>gemver</i>	2728	6182	6584	8509	1.26	2.20	5.48	4.24	3.40	5.63	5.79	7.57
<i>atax</i>	1371	1764	2668	4775	1.04	2.51	5.61	8.80	3.49	3.50	3.48	3.36
<i>syr2k</i>	2049	2440	4147	1294	1.59	1.26	1.98	23.19	3.28	2.07	1.72	0.17
<i>syrk</i>	1403	1530	2626	1040	1.48	0.97	1.52	14.08	3.33	2.53	1.99	0.20
<i>trmm</i>	975	2517	1010	1005	1.23	0.69	0.82	0.83	1.76	1.52	1.22	1.20

$$LUTs\ Ratio = \frac{\#LUT_{SODA-OPT}}{\#LUT_{NoHighLevelOpts.}} \quad Speedup = \frac{\#CYCLES_{NoHighLevelOpts.}}{\#CYCLES_{SODA-OPT}}$$

- Integration of the modular compiler frameworks allows optimization of the designs as progressive lowerings
- Significant benefits in speed up and area efficiency

## 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 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

Thank you

