

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



fast machine learning
for science

From DNN Models to ASIC Devices with SODA-OPT (Hands-on)

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

September 1, 2025

Antonino Tumeo

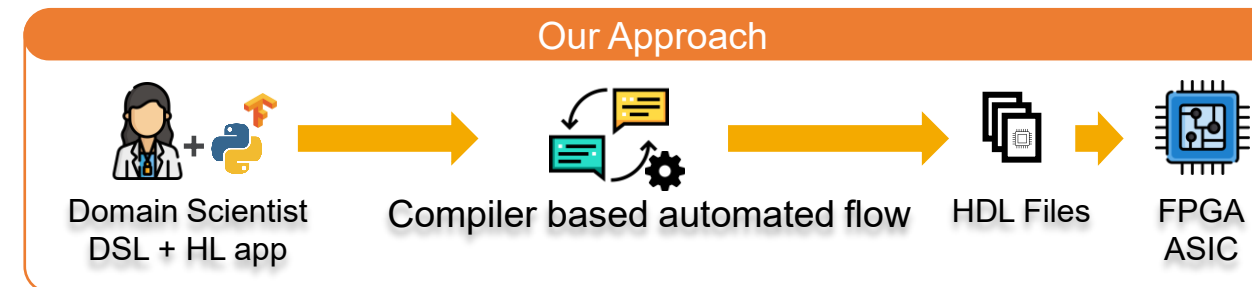
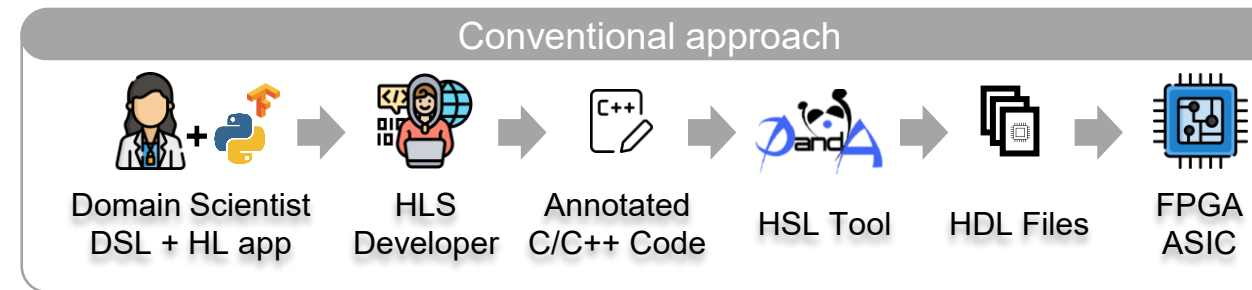
Nicolas Bohm Agostini



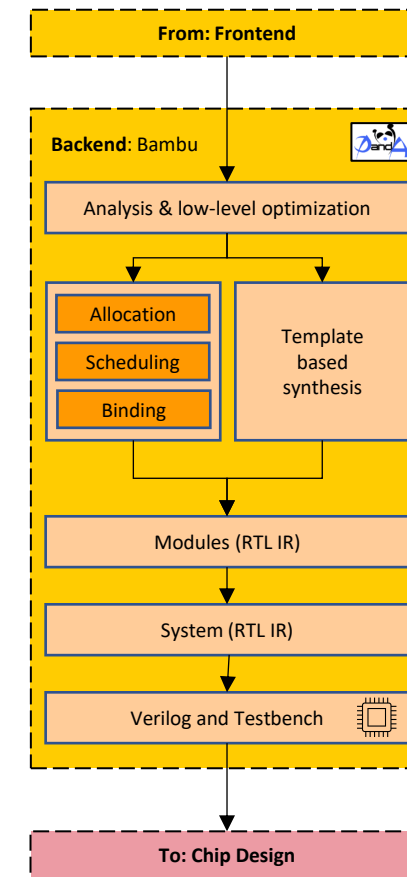
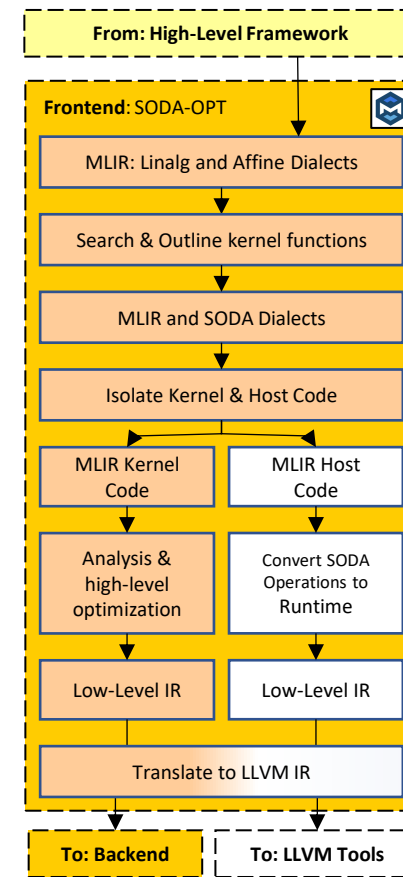
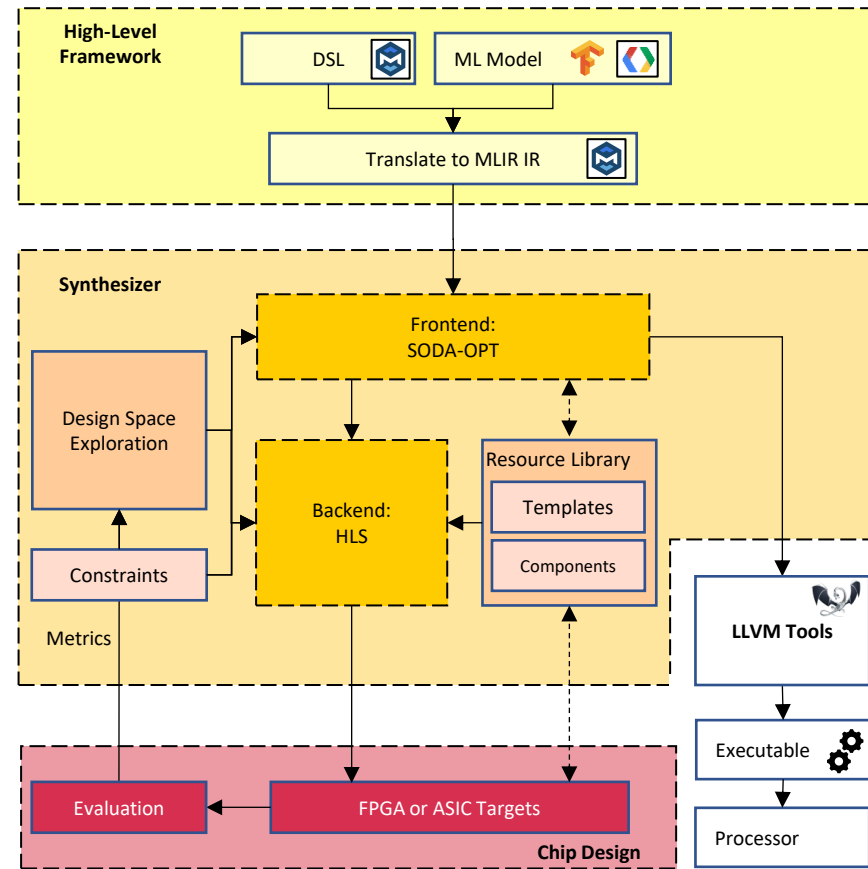
PNNL is operated by Battelle for the U.S. Department of Energy



Motivation



SODA End-To-End Toolchain



Goals

1. Learn how to use the SODA toolchain and the SODA-OPT compiler to perform end-to-end synthesis of high-level applications
2. Experiment with different high-level optimization strategies provided by SODA-OPT

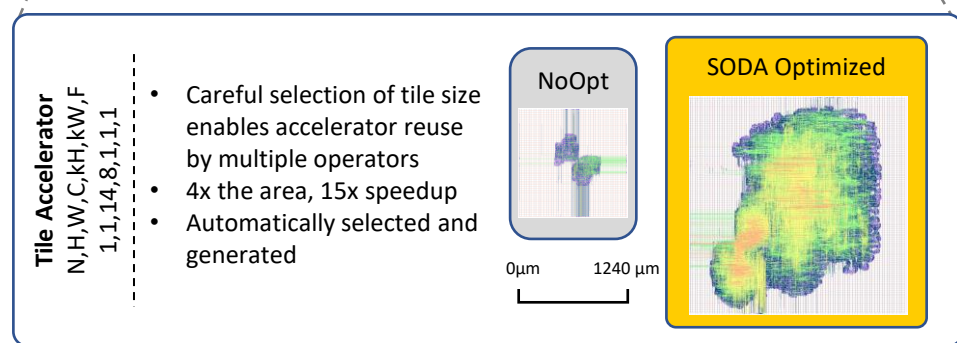
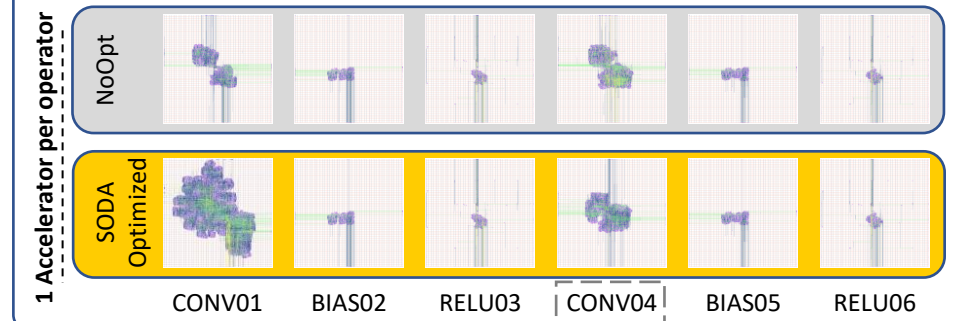
Why are HL code transformations important?

PolyBench

EXECUTION TIME (IN CLOCK CYCLES) FOR POLYBENCH KERNELS WITH ASIC TARGET - OPENPDK 45NM @ 500MHZ.
SPEEDUP SHOWN IN PARENTHESIS.

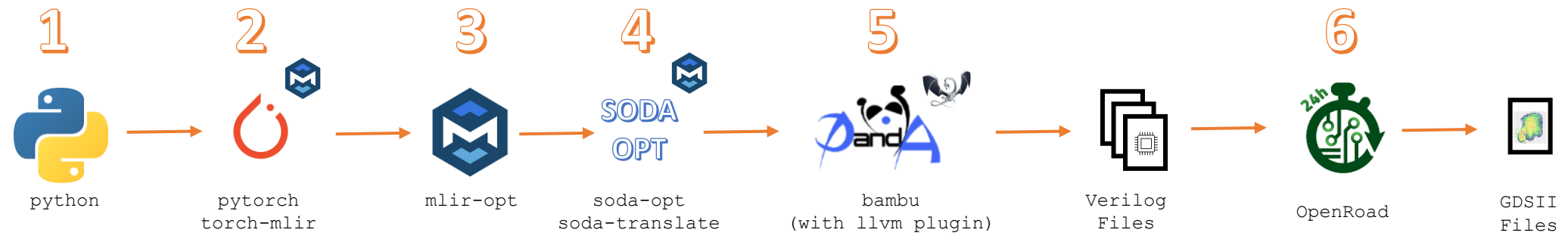
Opt. Strategy	No High Level Opts.				SODA-OPT Pipeline				
	Kernel Size	2	4	8	16	2	4	8	16
symm	421	2,928	21,400	163,368	31 (13.6x)	34 (86.1x)	325 (65.8x)	2,600 (62.8x)	
three_mm	388	3,087	25,010	211,298	47 (8.3x)	82 (37.6x)	656 (38.1x)	5,248 (40.3x)	
two_mm	315	2,475	20,258	167,490	52 (6.1x)	86 (28.8x)	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)	

Lenet



* Bambu target: OpenPDK 45nm ASIC @500MHz, 2 memory channels

Tutorial Structure



<https://github.com/pnnl/soda-benchmarks/tree/main/tutorials/pytorch>

Tutorial Structure



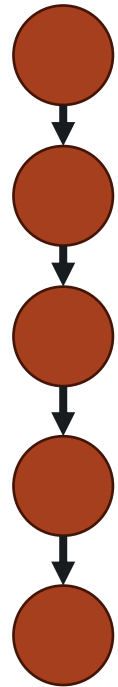
- 1 • Write a simple DNN model in python
- 2 • Convert model to MLIR (with `torch_mlir`)
• Lower model to TOSA MLIR dialect
- 3 • Lower model to `linalg` MLIR dialect
• Visualize the intermediate representation

- 4 • Select MLIR code for custom accelerator generation
• Optimize kernel code and generate IR for Bambu
- 5 • Synthesize baseline and optimized code into Verilog
• Compare results
- 6 • Place and route synthesized code
• Visualize final GDSII files

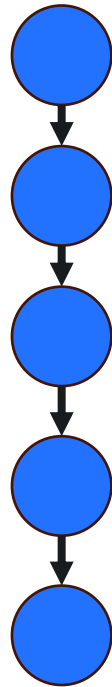
<https://github.com/pnnl/soda-benchmarks/tree/main/tutorials/pytorch>

In a Nutshell

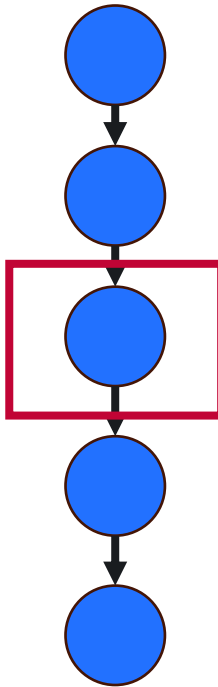
Computational Graph



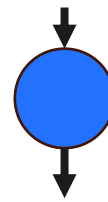
MLIR Representation



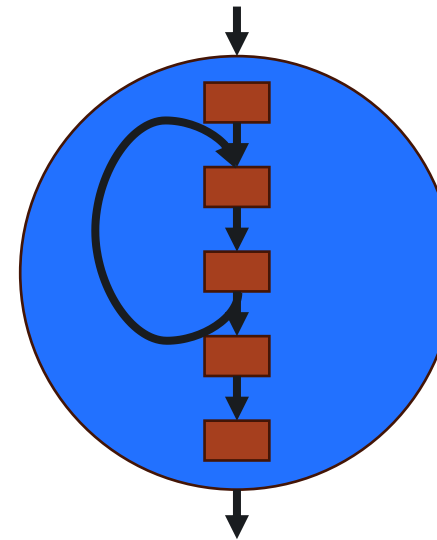
SODA-OPT Selects Code



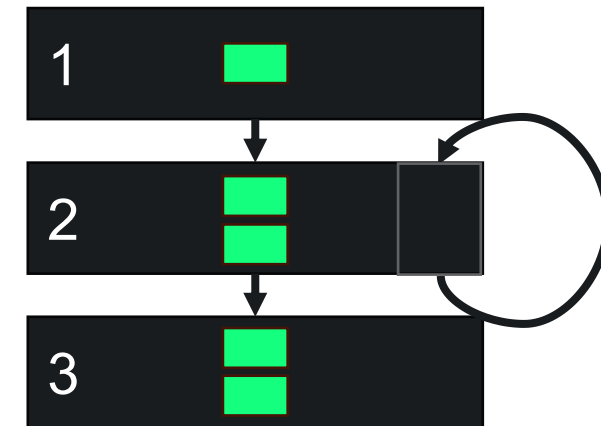
SODA-OPT Outlines Code



SODA-OPT Lowers to LLVM

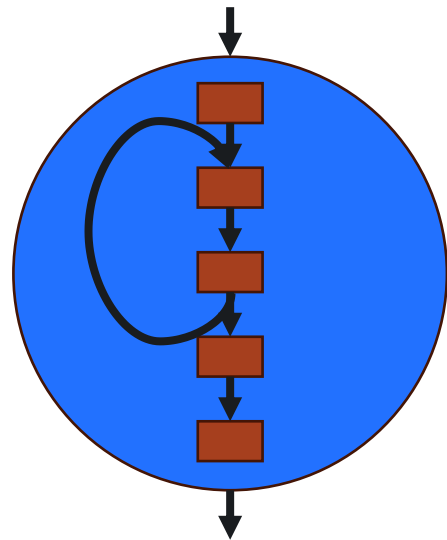


Bambu performs HLS: maps LLVM instructions to HW modules and schedules in a FSM

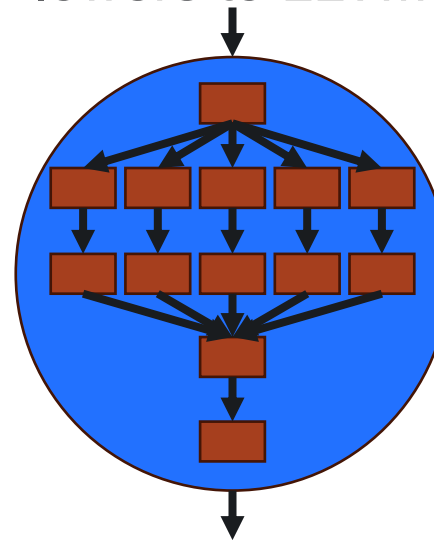


In a Nutshell

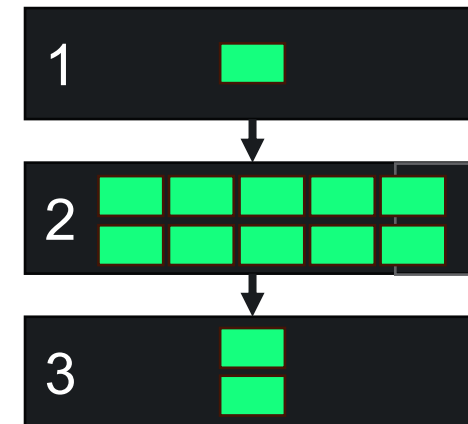
SODA-OPT
Outlines Code



SODA-OPT
Optimizes the code and
lowers to LLVM



Bambu performs
HLS: maps LLVM
instructions to
HW modules and
schedules in a
FSMD

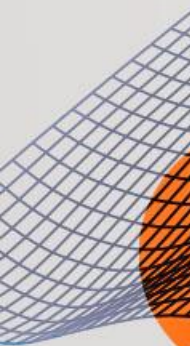


Thank you



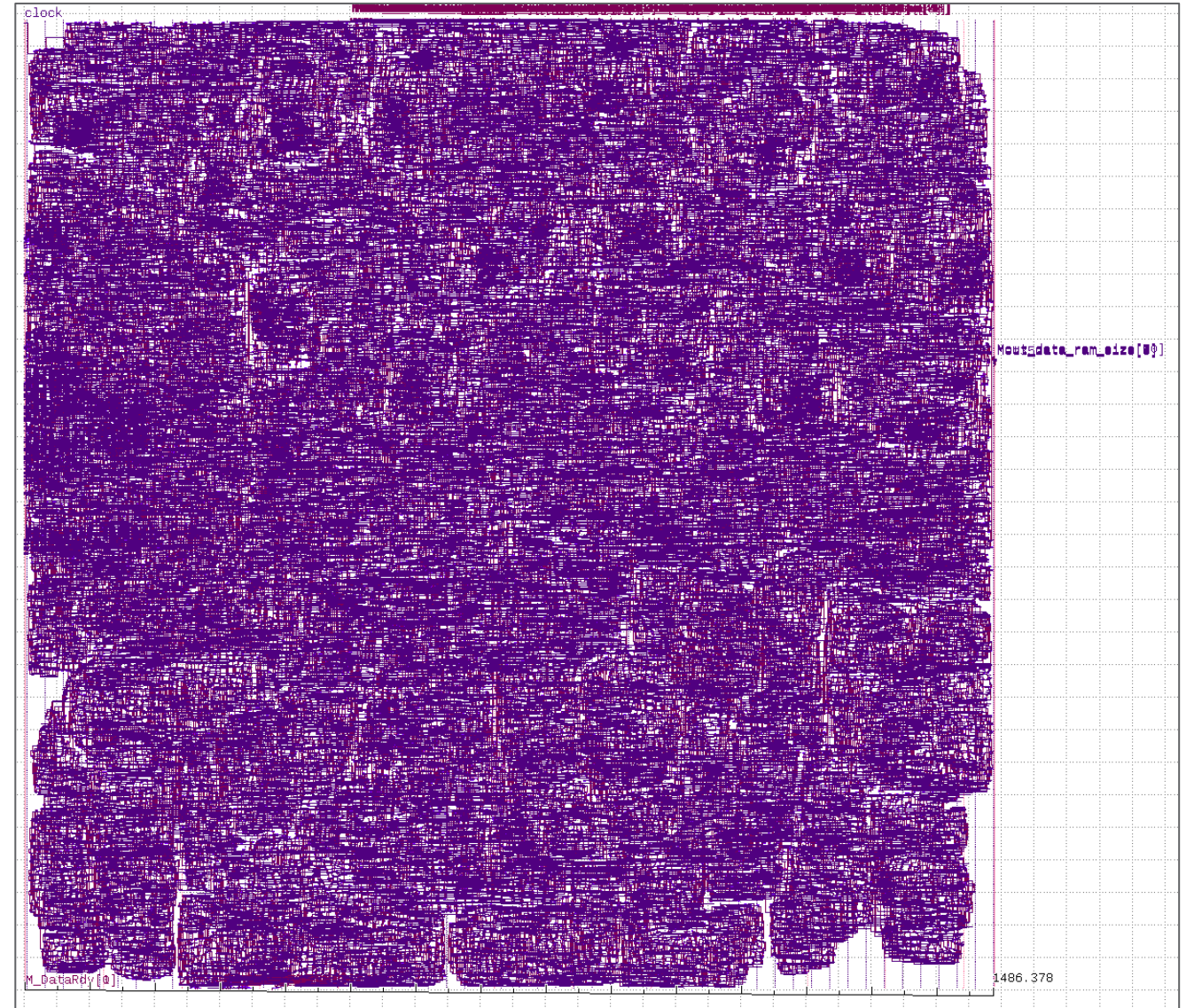
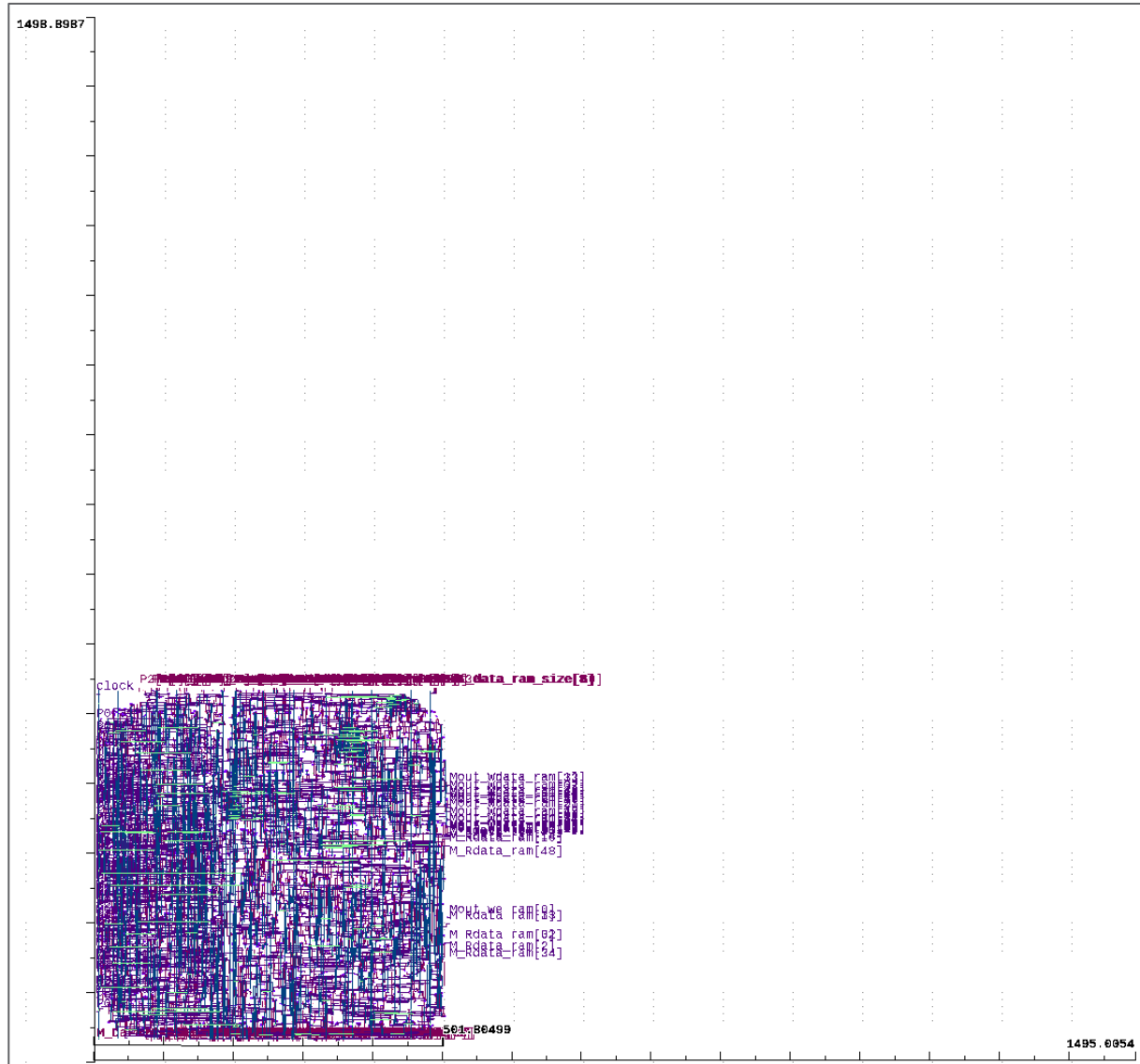


**Pacific
Northwest**
NATIONAL LABORATORY





Pacific Northwest
NATIONAL LABORATORY



The Effects of Our Automatic Optimization Pipeline

PolyBench

Opt. Strategy	FPGA Target: Xilinx xc7vx690t-ffg1930-3 @ 100MHz									FPGA Target: Xilinx xc7vx690t-ffg1930-3 @ 100MHz											
	No High Level Opts.				SODA-OPT Pipeline				Avg. speedup	No HL Opts. [#LUTs]				LUTs Ratio				Speedup / LUTs Ratio			
	Kernel Size	2	4	8	16	2	4	8		16	2	4	8	16	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	2,300	3,496	2,824	2,382	1.81	3.14	22.75	26.98*	6.09	4.76	1.48	1.38
<i>three_mm</i>	220	1,743	14,042	111,410	22	40	320 (F4)	2,560 (F4)	35.2x	3,122	7,108	7,696	2,014	1.68	4.07	30.05*	114.84*	5.95	10.71	1.46	0.38
<i>two_mm</i>	176	1,375	11,218	87,842	25	43	98	784 (F8)	66.4x	2,917	5,700	5,275	1,641	1.09	3.40	20.95	67.35*	6.45	9.41	5.46	1.66
<i>gemm</i>	103	794	6,538	42,514	16	28	71	568 (F8)	50.4x	1,929	3,385	2,693	1,005	1.32	3.51	22.22	59.55*	4.89	8.09	4.14	1.26
<i>bicg</i>	73	294	1,162	4,626	13	28	45	141	18.7x	1,370	1,642	2,614	1,027	1.68	3.55	5.92	40.52*	3.34	2.96	4.36	0.81
<i>mvt</i>	74	290	1,155	4,611	13	21	45	141	19.5x	1,464	3,893	2,966	5,461	1.57	1.47	5.23	7.78	3.63	9.39	4.91	4.20
<i>gesummv</i>	92	326	1,226	5,026	18	25	46	142	20.0x	1,690	1,838	2,883	1,197	1.56	3.38	5.27	35.45	3.29	3.85	5.05	1.00
<i>gemver</i>	154	606	2,377	9,620	36	49	75	300 (F8)	20.1x	2,728	6,182	6,584	8,509	1.26	2.20	5.48	4.24	3.40	5.63	5.79	7.57
<i>atax</i>	76	299	1,171	4,643	21	34	60	157	15.4x	1,371	1,764	2,668	4,775	1.04	2.51	5.61	8.80	3.49	3.50	3.48	3.36
<i>syr2k</i>	99	706	4,834	35,650	19	270	1,417	8,835	3.8x	2,049	2,440	4,147	1,294	1.59	1.26	1.98	23.19	3.28	2.07	1.72	0.17
<i>syrk</i>	79	546	3,682	22,594	16	222	1,211	7,896	3.3x	1,403	1,530	2,626	1,040	1.48	0.97	1.52	14.08	3.33	2.53	1.99	0.20
<i>symm</i>	92	729	5,570	42,754	19	526	3,338	22,610	2.4x	2,024	3,787	1,622	1,619	1.51	0.75	0.97	0.98	3.21	1.85	1.72	1.93
<i>trmm</i>	26	301	2,138	15,730	12	288	2,138	15,730	1.3x	975	2,517	1,010	1,005	1.23	0.69	0.82	0.83	1.76	1.52	1.22	1.20

Speedups across a wide range of algorithms **without** manual code annotation

$$LUTs\ Ratio = \frac{\#LUT_{OurTool}}{\#LUT_{NoHighLevelOpts.}}$$

$$Speedup = \frac{\#CYCLES_{NoHighLevelOpts.}}{\#CYCLES_{OurTool}}$$

On average, Speedup observed is bigger than Resource (LUT) consumption increase

* Bambu target: Xilinx xc7vx690t @100MHz, 2 memory channels per argument