A Holistic FPGA Architecture Exploration Framework for Deep Learning Acceleration

Jiadong Zhu, Dongsheng Zuo, Yuzhe Ma

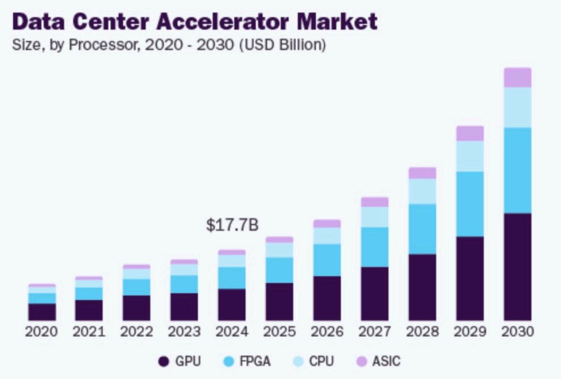
January 23, 2025

The Hong Kong University of Science and Technology (Guangzhou)

Table of Contents

FPGA Architecture Through A DL Lens

Accelerator Market Trends



- The FPGA accelerators are expected to grow steadily over the forecast period.[1]

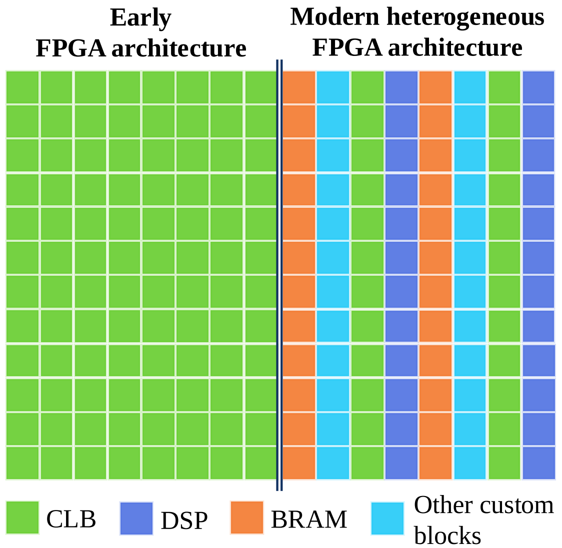
[1] Grand View Research, *Data center accelerator market size, share & trends analysis report by processor (cpu, gpu, fpga, asic)*, 2024. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/data-center-accelerator-market-report.

Comparison between FPGA and Other Platforms

	GPUs	ASICs	FPGAs
Generality	Turing-complete	Specific domain	Any custom HW
Architecture	Many cores / threads	Suits target domain	Spatial
HW Specialization	Fixed datapath & memory subsystem	Full flexibility	Reconfigurable
Power Consumption	High power	Most efficient	Moderate
NRE Cost	Off-the-shelf	Very high	Off-the-shelf

- FPGAs occupy an intermediate position on the spectrum of efficiency versus programmability, striking a unique balance in DL acceleration

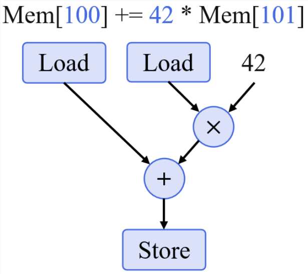
FPGA Architecture Overview



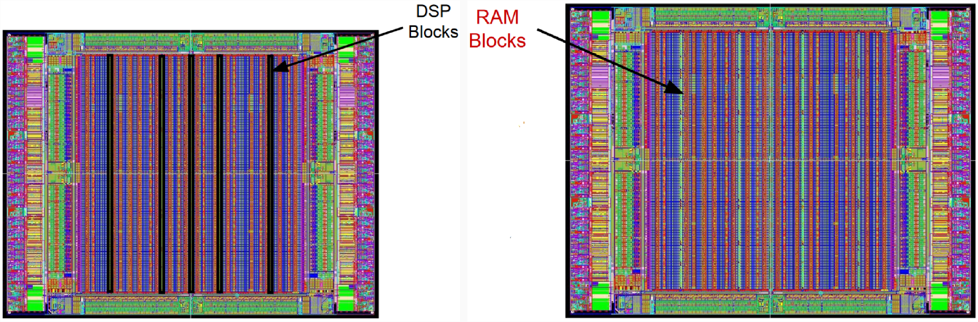
- Blocks and their strength for DL

Strength: Flexible Precision & Efficient Computing Implementation

- CLB
  - Most numerous
  - Can program to realize hardware of any bit width
    $\longrightarrow$ Use lowest precision that meets accuracy for each network / layer
  - Programmable routing: directly wire data from one unit to another
  - Programmable logic: perform only necessary operation



Strength: Hard Blocks & Low Latency Memory

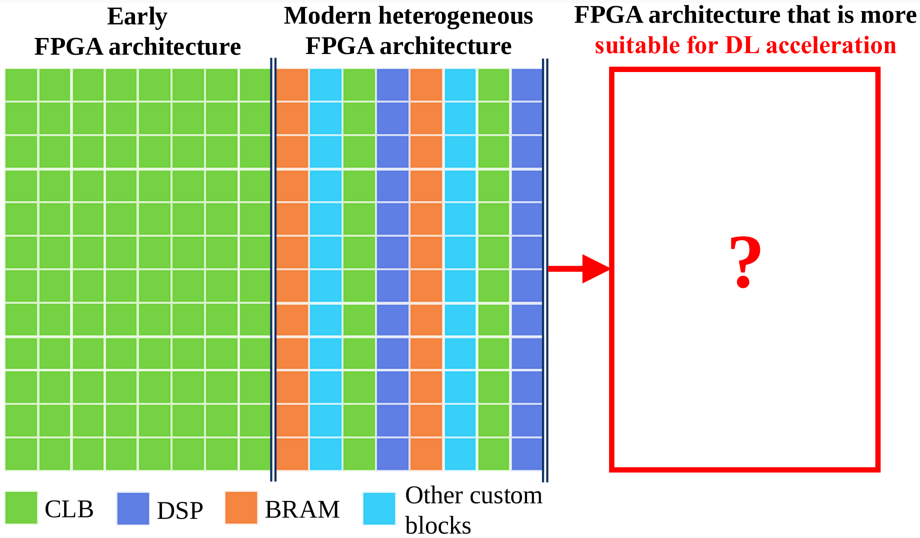


Source: Vaughn Betz's slides of the tutorial on Deep Learning-Optimized FPGA Architectures at MICRO 2022

- Hard block
  - DSP: designed to speed up multiply–accumulate (MAC) operations
- Massive bandwidth BRAM
  - $\sim$Pb/s of on-chip bandwidth (in a large chip) $\longrightarrow$ little or no batching
  - GPUs batch inputs to amortize weight re-loading $\longrightarrow$ latency increase

How to Make FPGA Architecture More Suitable for DL Acceleration?

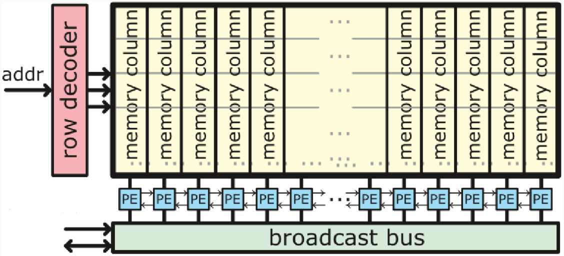
- Existing FPGA architectures are not designed specifically for DL workloads



Previous Work on Improving FPGA Architectures

Manually Improving Existing Blocks

- CLB $\longrightarrow$ adding adders and shadow multipliers[2].
- DSP $\longrightarrow$ optimizing for low-precision multiplications[3].
- BRAM $\longrightarrow$ integrating in-memory compute capabilities[4].



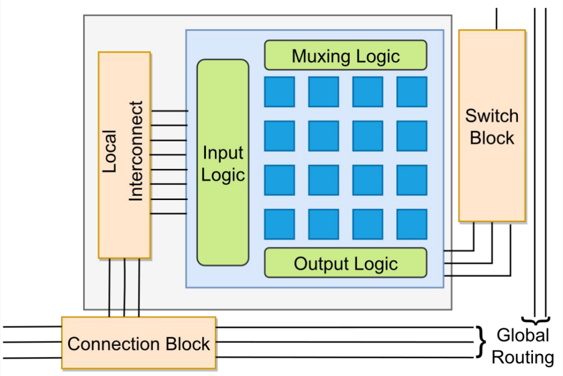
[2] A. Boutros *et al.*, “Math doesn’t have to be hard: Logic block architectures to enhance low-precision multiply-accumulate on fpgas,” in *Proc. FPGA*, 2019, pp. 94–103.

[3] A. Boutros *et al.*, “Embracing diversity: Enhanced dsp blocks for low-precision deep learning on fpgas,” in *Proc. FPL*, 2018, pp. 35–357.

[4] A. Arora *et al.*, “Comefa: Deploying compute-in-memory on fpgas for deep learning acceleration,” *ACM TRETS*, vol. 16, no. 3, pp. 1–34, 2023.

Mannually Adding New Blocks

- The Xilinx Versal architecture[5] and Intel Stratix 10 NX FPGA[6]
- Tensor Slices[7]



[5] B. Gaide *et al.*, "Xilinx adaptive compute acceleration platform: Versaltm architecture," in *Proc. FPGA*, 2019, pp. 84–93.

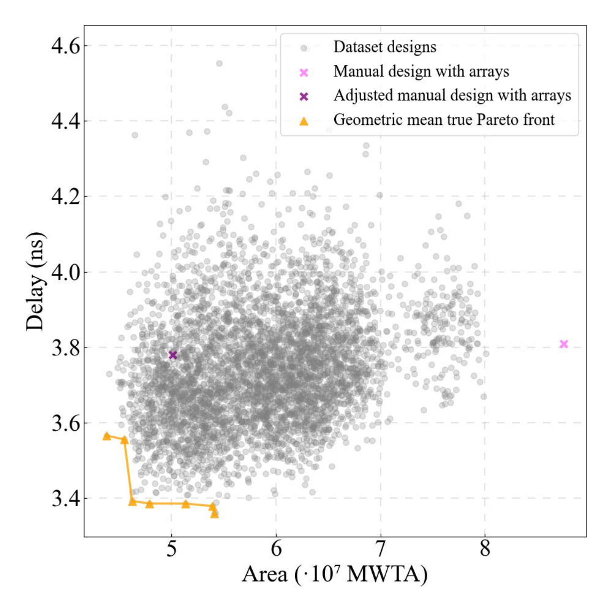
[6] M. Langhammer *et al.*, "Stratix 10 nx architecture and applications," in *Proc. FPGA*, 2021, pp. 57–67.

[7] A. Arora *et al.*, "Tensor slices: Fpga building blocks for the deep learning era," *ACM TRETS*, vol. 15, no. 4, pp. 1–34, 2022.

Mannually Optimizing FPGA Global Architecture? Too Vast Design Space!

Type	Parameter	Description
Logic Block	N	number of BLEs per CLB
	K	number of LUT inputs
	I	number of CLB inputs
	F_{clocal}	sparse crossbar flexibility
PE array	S_{array}	size of the PE array
RAM	S_{RAM}	size of the BRAM
Routing	R_l	$L16$ routing wire segment ratio
Layout	$Layout$	layout strategy
	$Fill$	whether fill empty grids with CLB
	Asp	aspect ratio of the layout

Mannually Optimizing FPGA Global Architecture? Too Vast Design Space!



Challenges

Designing a competitive FPGA architecture is challenging

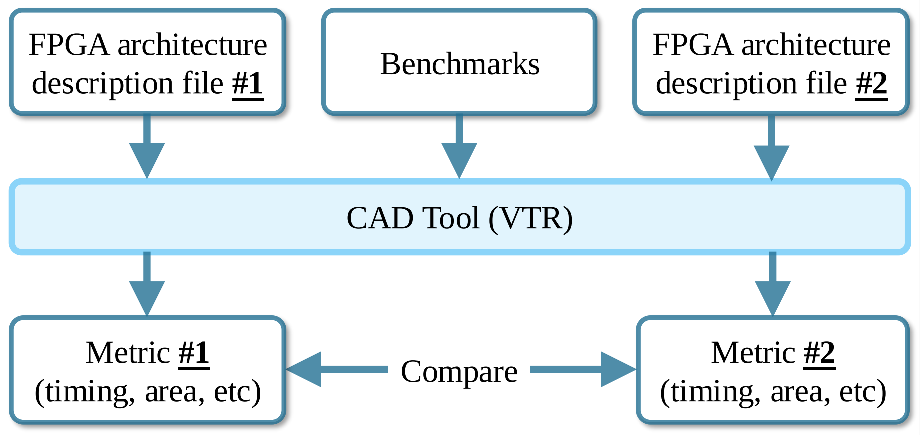
- Require navigating a vast design space to achieve an optimal balance between metrics

Manual design is inefficient for exploring large design spaces

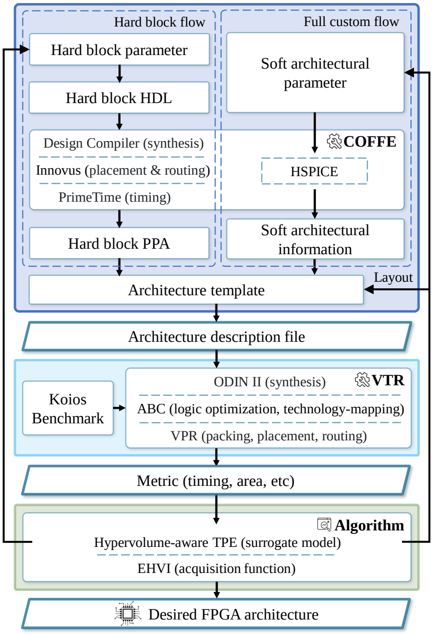
- A suitable automatic framework with design space exploration (DSE) algorithm is essential

FPGA Architecture Exploration Framework Overview

FPGA Architecture Evaluation and Exploration

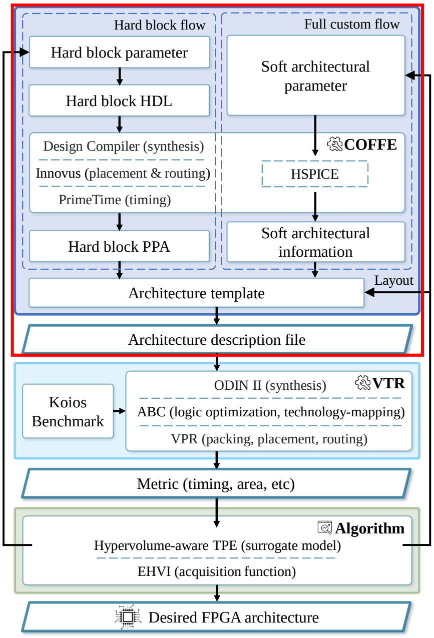


Proposed Exploration Framework



- Integrated flow: COFFE & VTR generate architecture description files and output the metrics
- The hypervolume-aware TPE method iterates the flow

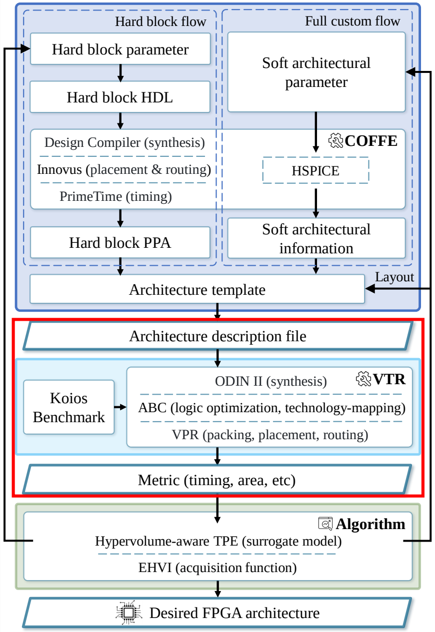
Proposed Exploration Framework——COFFE Part



- COFFE[a] models heterogeneous FPGA architectures
- Each architecture design is abstracted into two inputs:
  - hard block design parameters
  - soft architectural parameters

[a] S. Yazdanshenas and V. Betz, “Coffe 2: Automatic modelling and optimization of complex and heterogeneous fpga architectures,” *ACM TRETS*, vol. 12, no. 1, pp. 1–27, 2019.

Proposed Exploration Framework——VTR Part

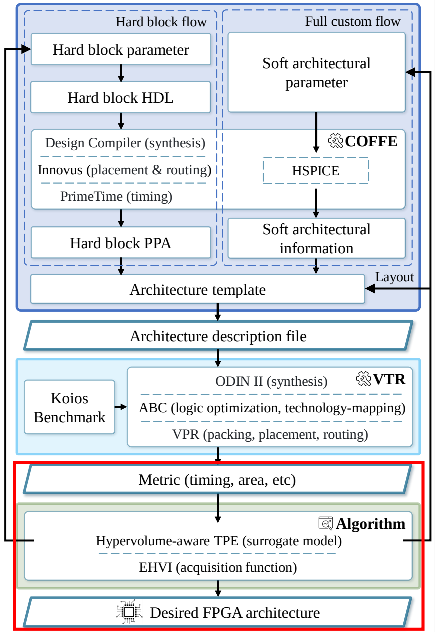


- VTR[a]: a suite of CAD tools for FPGA architecture
- Koios[b]: a suite of DL acceleration benchmark circuits for FPGA architecture

[a] K. E. Murray *et al.*, "Vtr 8: High-performance cad and customizable fpga architecture modelling," *ACM TRETS*, vol. 13, no. 2, pp. 1–55, 2020.

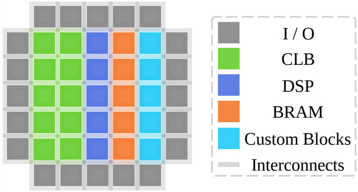
[b] A. Arora *et al.*, "Koios 2.0: Open-source deep learning benchmarks for fpga architecture and cad research," *IEEE TCAD*, 2023.

Proposed Exploration Framework——Algorithm Part



- The DSE algorithm iterates the flow
  - Take metrics as the inputs
  - Select the next sampling point (a set of parameters)

Architecture Template



- The template includes columns of CLBs, DSPs, BRAMs, and PE arrays, with I/Os positioned along the FPGA perimeter.

- The complex DSP[8] supports fixed-point and floating-point precisions
- The PE array[9] supports int8 and int16 precisions, as well as matrix-matrix and matrix-vector multiplication.
  - Employ Schoolbook method[10] to split 16-bit mult $\longrightarrow$ 4 fewer 8-bit adders

---

[8] Intel, "Intel agilex fpgas and socs," (2019), [Online]. Available: https://www.intel.com/content/www/us/en/products/programmable/fpga/agilex.html.

[9] A. Arora *et al.*, "Tensor slices: Fpga building blocks for the deep learning era," *ACM TRETS*, vol. 15, no. 4, pp. 1–34, 2022.

[10] E. Ustun *et al.*, "Impress: Large integer multiplication expression rewriting for fpga hls," in *Proc. FCCM*, 2022, pp. 1–10.

Multi-objective FPGA Architecture Search

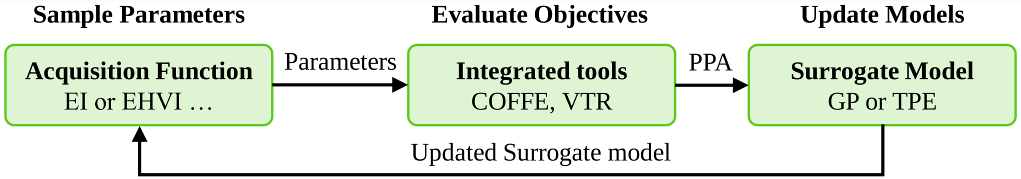
Design Space Definition

Type	Parameter	Description	Range of values
Logic Block	N	number of BLEs per CLB	6, 8, 10, 12
	K	number of LUT inputs	5, 6
	I	number of CLB inputs	32: 68: 4
	F_{clocal}	sparse crossbar flexibility	0.25, 0.5
PE array	S_{array}	size of the PE array	4×4, 8×8
RAM	S_{RAM}	size of the BRAM	16Kb, 20Kb, 32Kb, 40Kb
Routing	R_l	$L16$ routing wire segment ratio	0.1, 0.15, 0.2
Layout	*Layout*	layout strategy	spatial, clustered
	Fill	whether fill empty grids with CLB	0, 1
	Asp	aspect ratio of the layout	0.5, 1, 2

* The values are either listed individually or start : end : stride.

- Most of them are restricted to the most common options

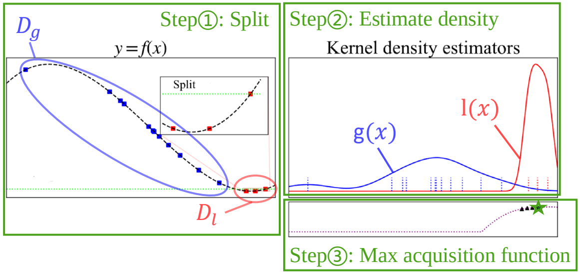
Bayesian Optimization (BO)



- Gaussian Process (GP) models $p(y \mid x)$ directly by assuming a multivariate normal distribution over the search space $\longrightarrow$ struggles with discrete or categorical variables due to its smoothness assumption.

Tree-Structured Parzen Estimator (TPE)

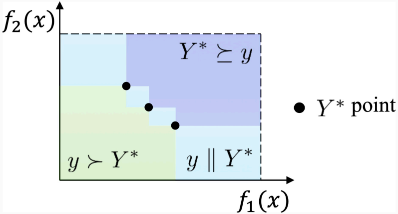
- TPE splits observations
  - Good observation: D_l
  - Bad observation: D_g
- Estimate two density functions
  - good density $l(x)$
  - bad density $g(x)$



- y: objective values in observations, y^*: value to split observations

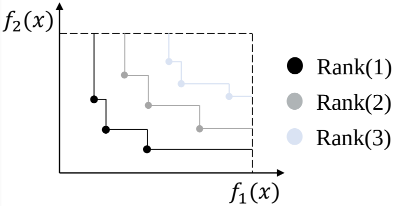
$$p(x \mid y) = \begin{cases} l(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases} \quad p(y < y^*) = \gamma \tag{1}$$

S. Watanabe, "Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance," *arXiv preprint*, 2023.



- y: objective values in observations
- Y^*: points to split observations

$$p(x \mid y) = \begin{cases} l(x) & \text{if } (y \succ Y^*) \cup (y \parallel Y^*) \\ g(x) & \text{if } Y^* \succeq y \end{cases} \qquad p(y \succ Y^* \cup y \parallel Y^*) = \gamma \tag{2}$$



- Split mainly by nondomination rank (take a certain rank points as Y^*)

- Acquisition function: Expected Hypervolume Improvement (EHVI)

Experiments

DL Acceleration Benchmark

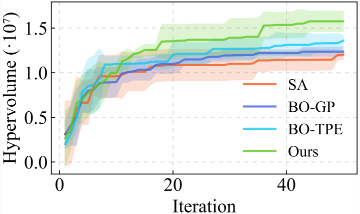
- Selected from the Koios benchmark suite[11]
- Various applications, precisions, and operation modes for PE arrays

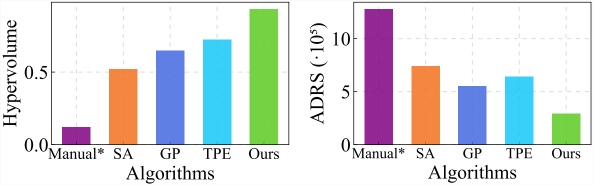
Benchmark	Precision	Array Mode	Description
attention_layer	int16	mat-vec	Self-attention layer
conv_layer	int16	mat-mat	Convolution layer
lstm	int16	mat-vec	LSTM layer
tpu	int8	mat-mat	Google's TPU v1 like
fcl	int8	mat-mat	Fully connected layer

[11] A. Arora *et al.*, "Koios 2.0: Open-source deep learning benchmarks for fpga architecture and cad research," *IEEE TCAD*, 2023.

Hypervolume & Average Distance to Reference Set (ADRS)

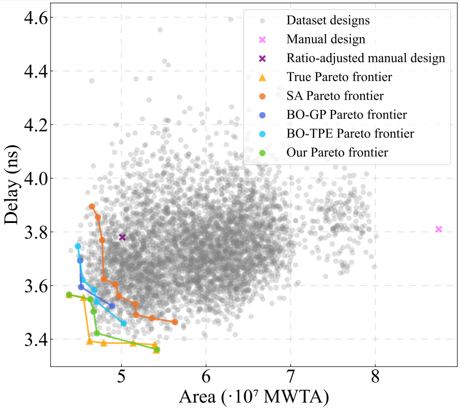
- handle a variety of DL workloads $\longrightarrow$ geometric mean

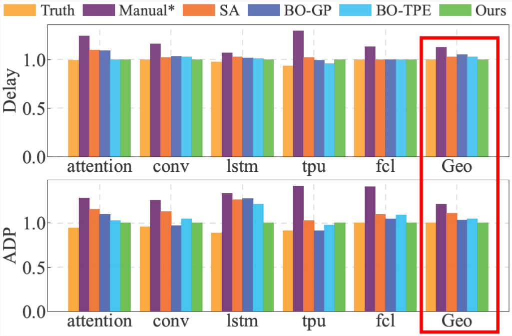




- 29.4% and 89.5% better than the second-best in hypervolume and ADRS

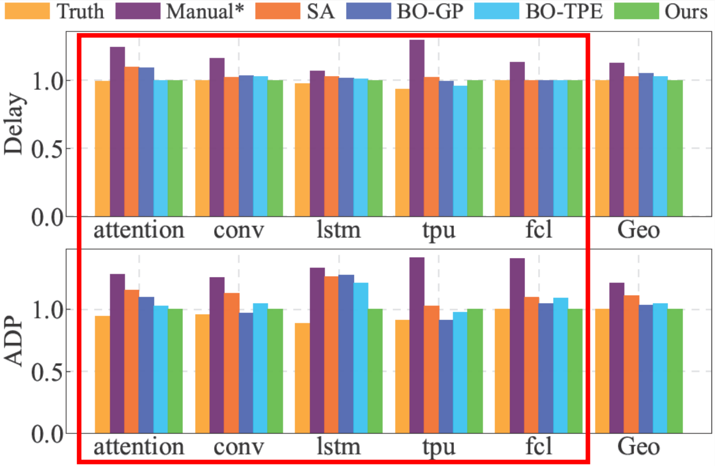
Pareto Frontiers (Geometric Mean)





- Reduce delay by 12.8% and area-delay product (ADP) by 21.4% compared to the manual design with adjusted block ratio
- Outperform all algorithm baselines in both delay and ADP.

Respective Results



- Achieve the best results in 7 out of 10 cases
- Weight of each benchmark's PPA values in the mean calculation can be adjusted

Thank You!