DREAMPlaceFPGA: An Open-Source Analytical Placer for Large Scale Heterogeneous FPGAs using Deep-Learning Toolkit

Rachel Selina Rajarathnam[1], Mohamed Baker Alawieh[1], Zixuan Jiang[1], Mahesh A. Iyer[2], David Z. Pan[1]

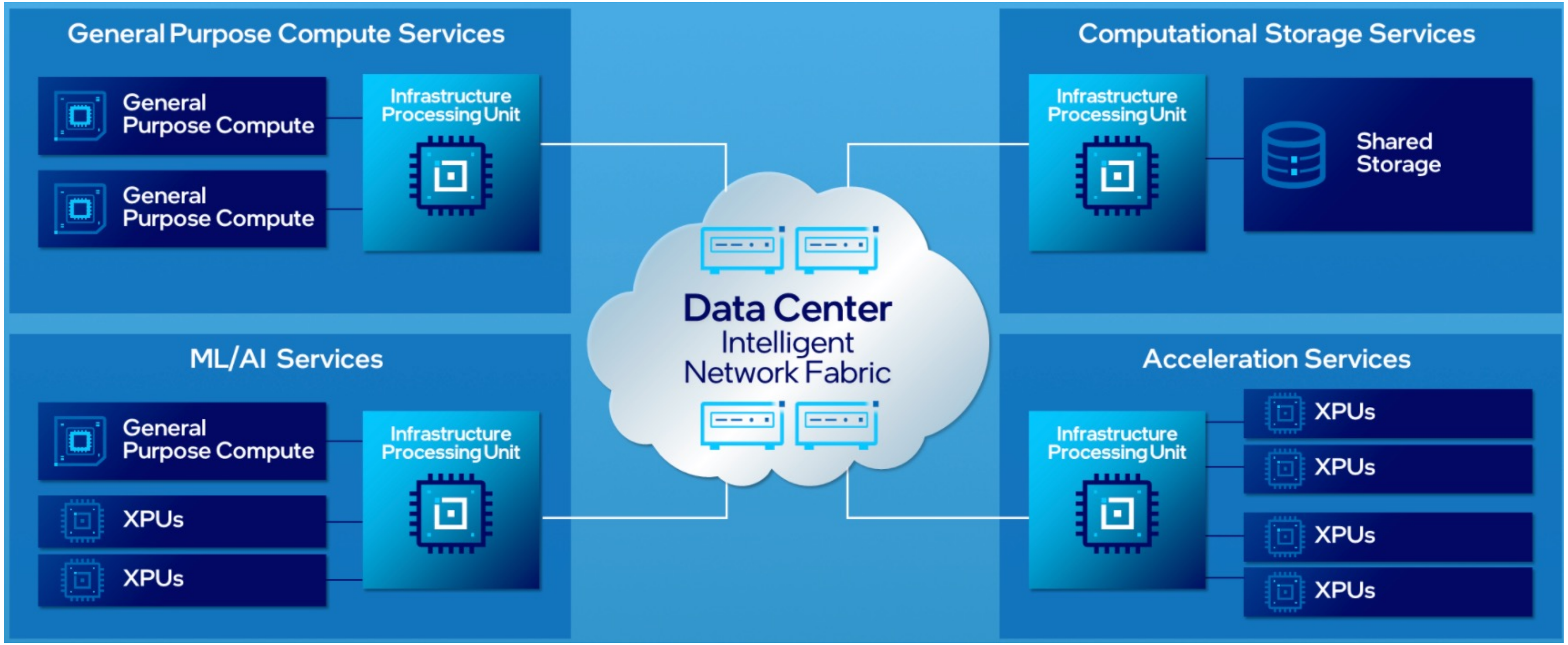
[1]*ECE Department, The University of Texas at Austin, TX, USA*

[2]*Intel Corporation, CA, USA*

This work was supported in part by Intel Strategic Research Alliance, Intel & VMware, and NSF

HETEROGENEOUS PLATFORMS IN DATA CENTERS

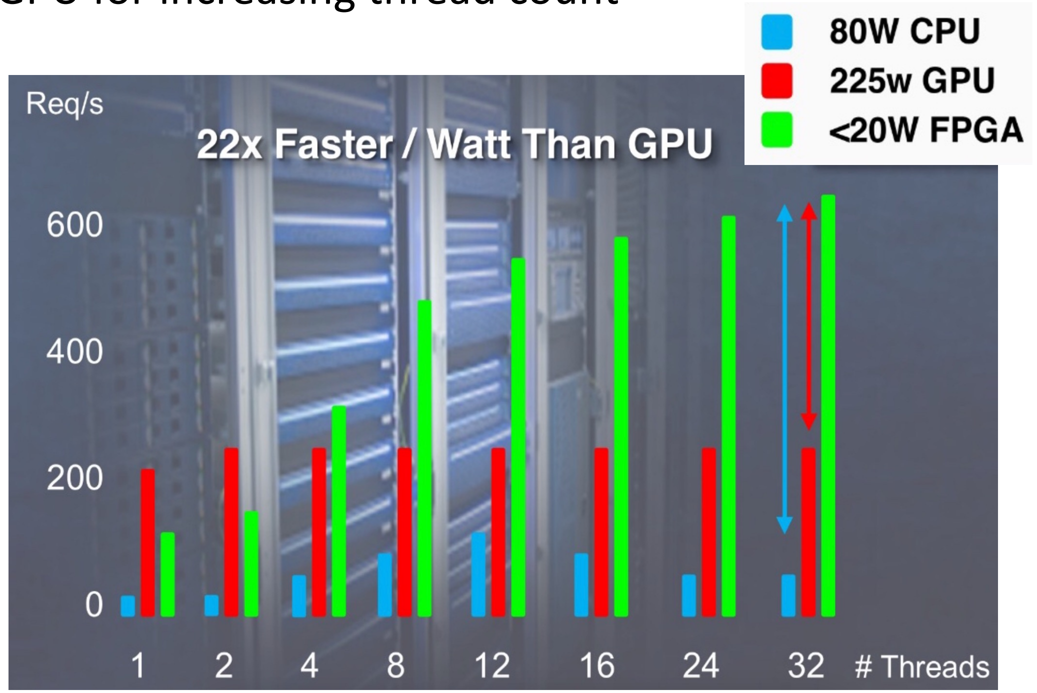
- XPU: Application/workload-specific Processing Unit

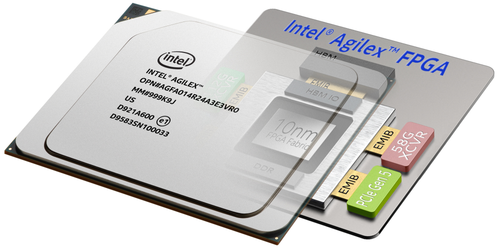

General Purpose Compute Services
General Purpose Compute
General Purpose Compute
Infrastructure Processing Unit
Computational Storage Services
Infrastructure Processing Unit
Shared Storage
Data Center Intelligent Network Fabric
ML/AI Services
General Purpose Compute
XPUs
XPUs
Infrastructure Processing Unit
Acceleration Services
Infrastructure Processing Unit
XPUs
XPUs
XPUs
XPUs

Source: https://www.hpcwire.com/2021/06/15/intel-debuts-infrastructure-processing-unit-as-part-of-broader-xpu-strategy/

FPGA IN DATA CENTERS

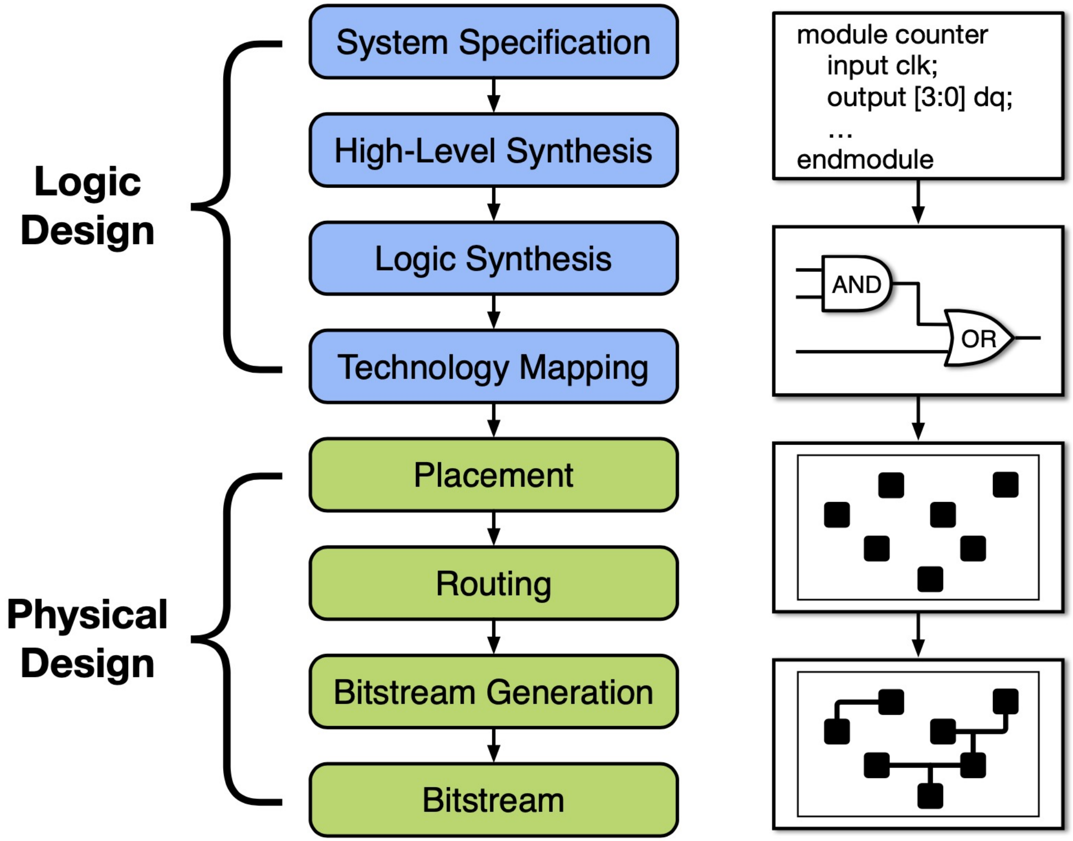
- Better processing performance per Watt for data centric acceleration tasks than CPU or GPU for increasing thread count





Source: Xilinx & Intel

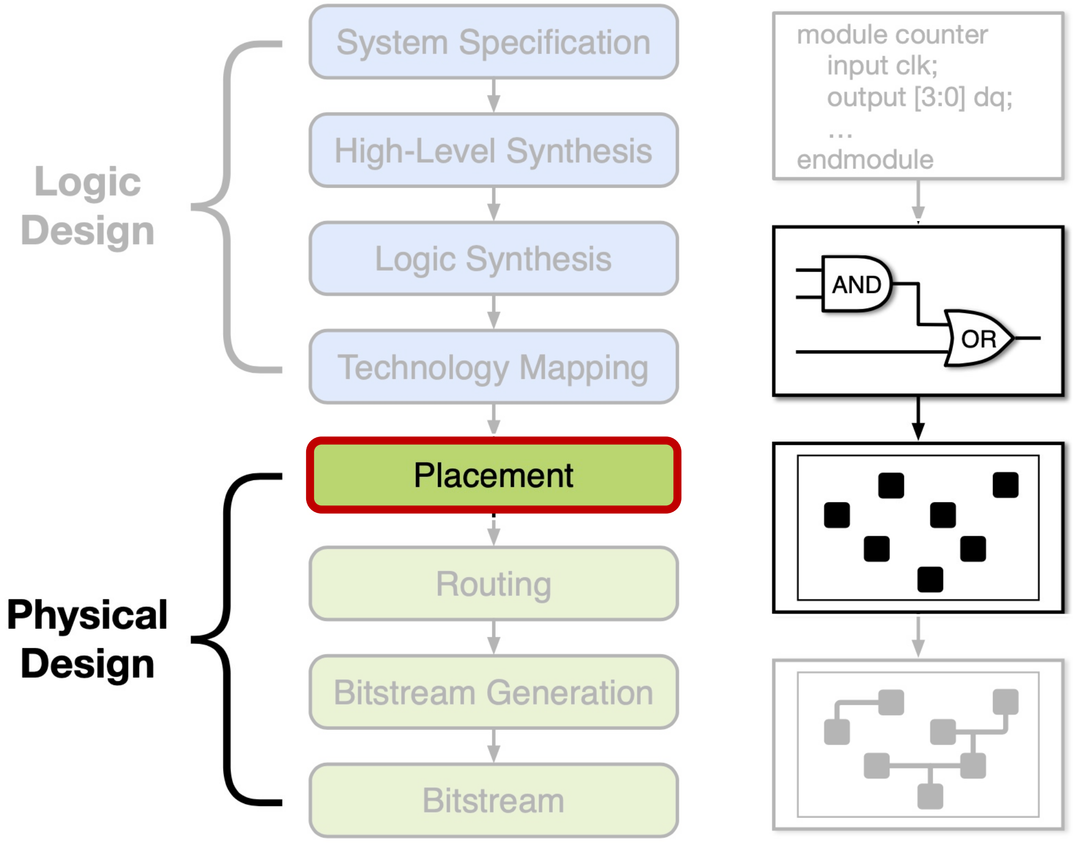
FPGA Design Implementation



```
module counter
   input clk;
   output [3:0] dq;
   ...
endmodule
```

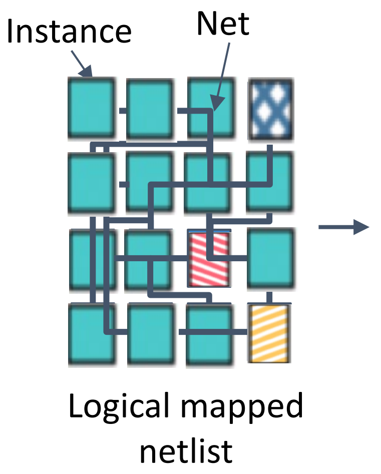
Source: http://wuxili.net/pdf/dissertation.pdf

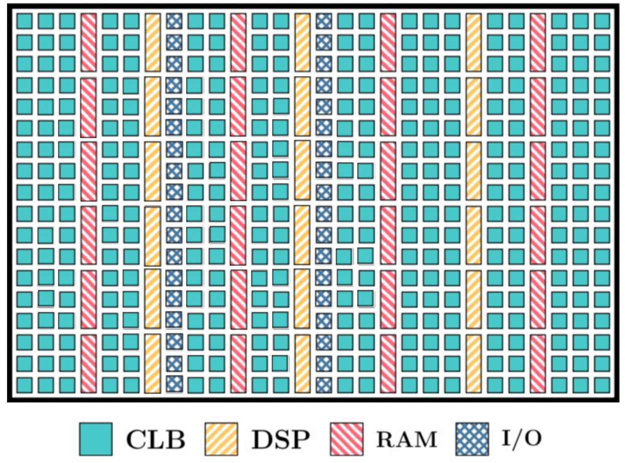
FPGA Design Implementation



Source: http://wuxili.net/pdf/dissertation.pdf

FPGA Placement



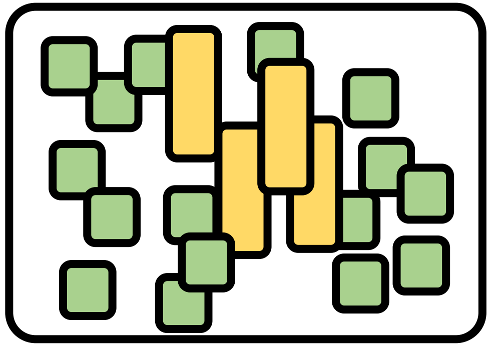


FPGA Layout: Xilinx Ultrascale

- Determine locations of instances on a fixed-floorplan chip with limited resources
- Concurrent optimization of wirelength, routing congestion, etc.
- Placement stages: Global Placement, Legalization, and Detailed Placement

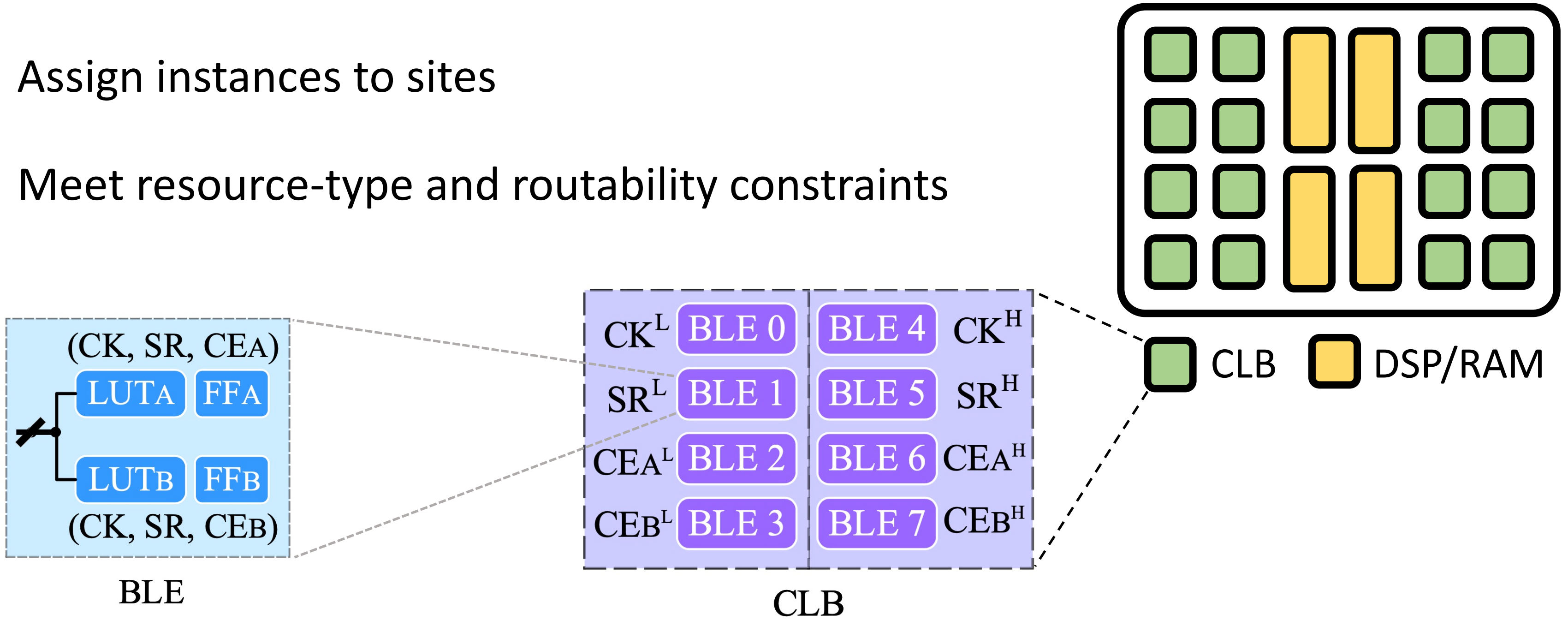
GLOBAL PLACEMENT

- Obtain rough legal locations for instances
- Optimization problem formulated as:
  - Simulated Annealing
  - Quadratic Placement
  - Non-linear Placement



LEGALIZATION

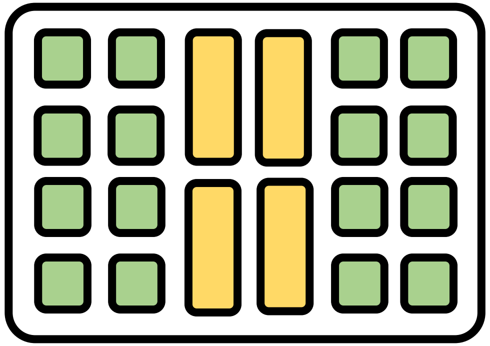
- Assign instances to sites
- Meet resource-type and routability constraints



Source: Xilinx Ultrascale Architecture

DETAILED PLACEMENT

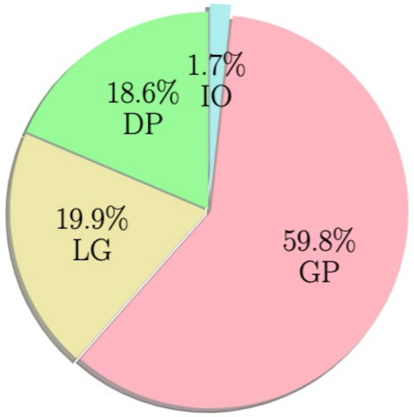
- Further improve metrics of legalized placement
- Formulated as:
  - Independent Set Matching
  - Slot Assignment



FPGA Placement

- Global Placement (GP):
  - Obtain rough legal locations

  Focus of this work
- Legalization (LG):
  - Assign instances to sites
- Detailed Placement (DP):
  - Further improve metrics of legalized placement



elfPlace Placement runtime breakdown

W.Li+, ICCAD'2019

Existing FPGA Placement Acceleration Schemes

- Multi-threaded CPU [W. Li+, TCAD'2018; W. Li+, ICCAD'2019]

  Runtime benefit plateaus after certain thread count

- FPGA [S. Dhar+, FPL'2019; S. Dhar+, HPC'2019]
  - Hybrid CPU + FPGA platform

  Limited by total memory on FPGA

- GPU [C. Fobel+, NEWCAS'2012; Y. Meng+, TCAD'2021]

  Expertise on underlying hardware for efficient kernel development

Existing FPGA Placement Acceleration Schemes

- Multi-threaded CPU [W. Li+, TCAD'2018; W. Li+, ICCAD'2019]
  - Runtime benefit plateaus after certain thread count
- FP
- GF
  - Expertise on underlying hardware for efficient kernel development

Non-trivial effort to adapt acceleration schemes to changes in placement formulations!

Can we do better?

DREAMPlaceFPGA

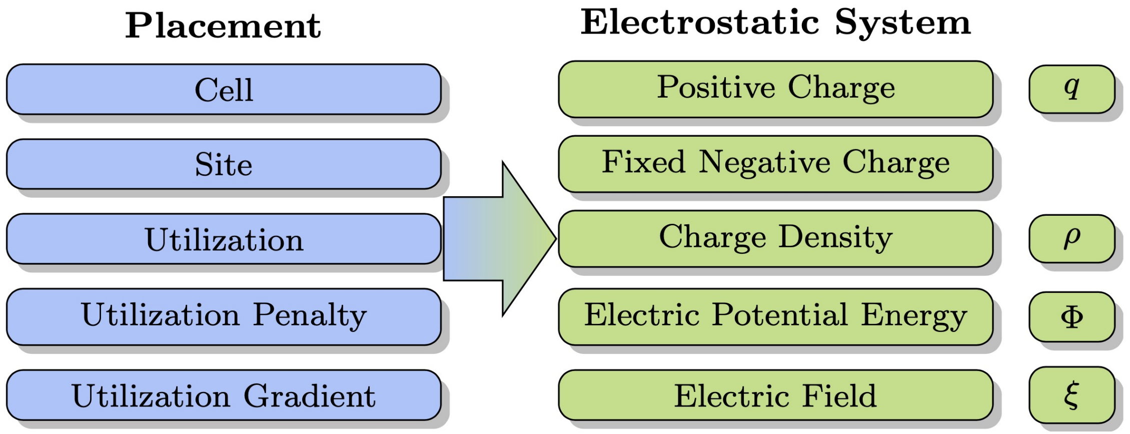
- Adapt the state-of-the-art elfPlace placer on a DL toolkit
  - Inspired by DREAMPlace framework for ASIC
- Re-use advanced libraries available in deep learning toolkit
- Consists of low-level optimized operators and high-level programming interface => *low development cost*
- Contribute to FPGA open-source ecosystem

ELFPLACE OVERVIEW

- A flat, nonlinear placement algorithm for large-scale heterogeneous FPGAs

$$\min_{x,y} W(x,y)\ \ s.t.\, D(x,y) \le D_0 \qquad \Rightarrow \qquad \min_{x,y} f(x,y) = \min\left(\sum_{e\in E} W_e(x,y) + \lambda D(x,y)\right)$$

- Casts the placement density cost $D(x,y)$ to the potential energy $\boldsymbol{\Phi}(x,y)$ of an electrostatic system.



W.Li[+], ICCAD'2019

ELFPLACE OVERVIEW

- Each resource type solved as a separate electrostatic system

  {LUT, FF, DSP, RAM}

- Formulate using Augmented Lagrangian Method

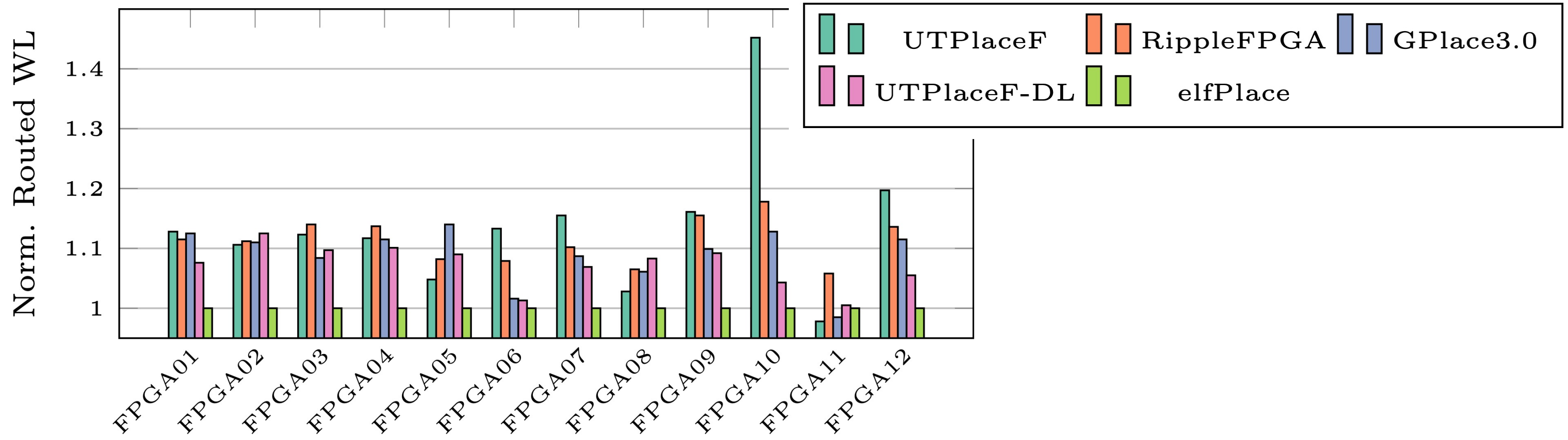
$$\min_{x,y} f(x,y) = \widetilde{W}(x,y) + \sum_{s \in S} \lambda_s \left(\Phi_s(x,y) + \frac{c_s}{2} \Phi_s(x,y)^2 \right)$$

- Spectral methods (Discrete Cosine Transform) used to numerically solve the Poisson's equations for the electrostatic system

W.Li[+], ICCAD'2019

ELFPLACE OVERVIEW

- Best Routed WL: UTPlaceF (-13.6%), RippleFPGA (-11.3%), GPlace3.0 (-8.9%), & UTPlaceF-DL (-7.1%)

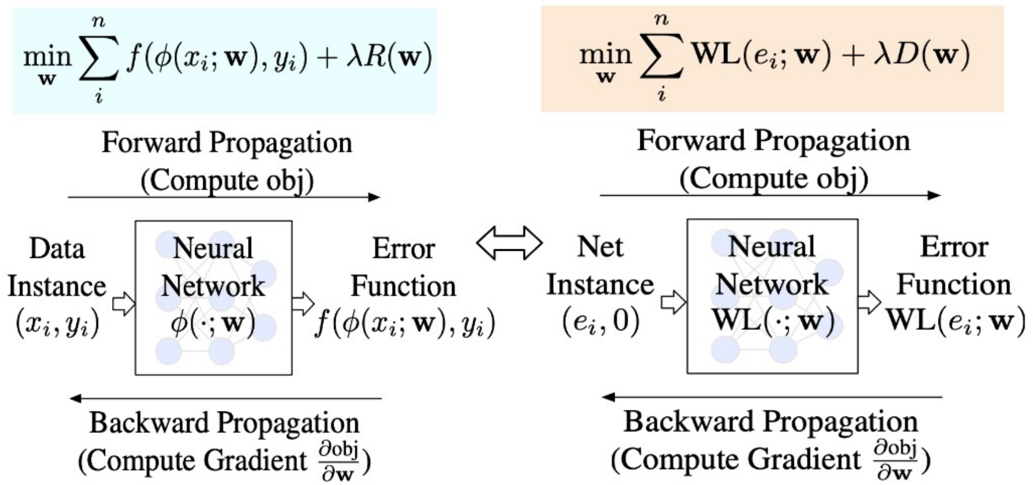


- Placement Runtime: UTPlaceF (3.97x), RippleFPGA (1.04x), GPlace3.0 (3.42x), & UTPlaceF-DL (3.63x)

W.Li+, ICCAD'2019

DREAMPlace Overview

- Novel **Analogy** by casting the nonlinear placement optimization into a neural network training problem
- Greatly leverage deep learning hardware (GPU) and software (e.g., PyTorch)



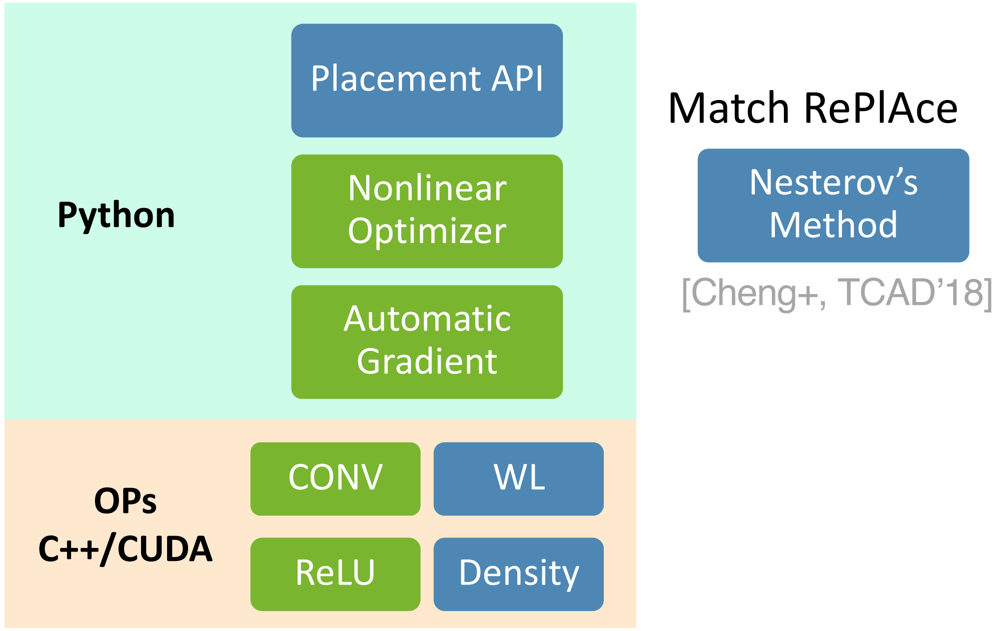
$\min_{\mathbf{w}} \sum_{i}^{n} f(\phi(x_i;\mathbf{w}), y_i) + \lambda R(\mathbf{w})$
Forward Propagation (Compute obj)
Data Instance (x_i, y_i)
Neural Network $\phi(\cdot;\mathbf{w})$
Error Function $f(\phi(x_i;\mathbf{w}), y_i)$
Backward Propagation (Compute Gradient $\frac{\partial obj}{\partial \mathbf{w}}$)
$\min_{\mathbf{w}} \sum_{i}^{n} \mathrm{WL}(e_i;\mathbf{w}) + \lambda D(\mathbf{w})$
Forward Propagation (Compute obj)
Net Instance $(e_i, 0)$
Neural Network $\mathrm{WL}(\cdot;\mathbf{w})$
Error Function $\mathrm{WL}(e_i;\mathbf{w})$
Backward Propagation (Compute Gradient $\frac{\partial obj}{\partial \mathbf{w}}$)

Train a Neural Network **Solve a Placement**

Y. Lin+, DAC'2019

DREAMPlace Overview

Leverage highly optimized deep learning toolkit PyTorch



Y. Lin+, DAC'2019

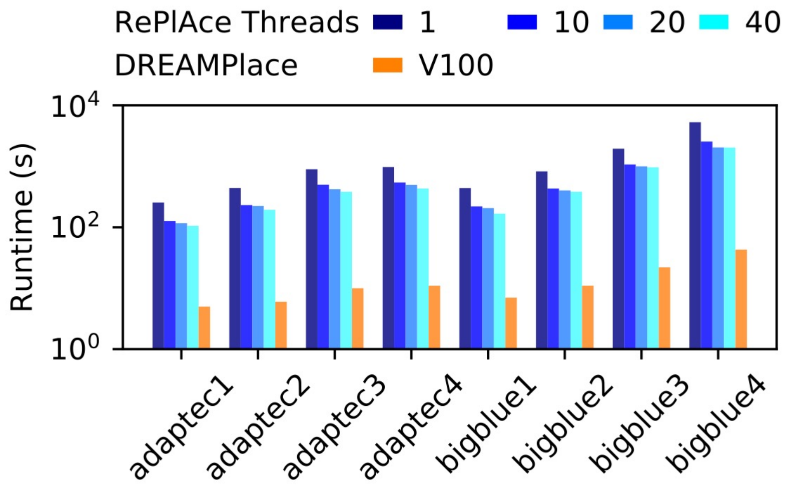
DREAMPLACE OVERVIEW

RePlAce **[Cheng+, TCAD'18]**
- CPU: 24-core 3.0 GHz Intel Xeon
- 64GB memory allocated

DREAMPlace
- CPU: Intel v4 @2.20GHz
- GPU: 1 NVIDIA Tesla V100
- Single CPU thread used

Same quality of results!

34x speedup



10M-cell design finishes within **5min c.f. 3h**

Y. Lin+, DAC'2019

Why Integrate elfPlace & DREAMPlace ?

- Quality of Results: elfPlace is a state-of-the-art academic placer
- Acceleration using Deep Learning (DL) Hardware & Software
- Low development overhead
  - High-level Python Programming + low-level operators in C++/CUDA
- Extensible
  - Easy to incorporate new algorithms and acceleration techniques
  - Build on existing framework – similar to DREAMPlace (ASIC)

W. Li+, ICCAD'2019; Y. Lin+, DAC'2019

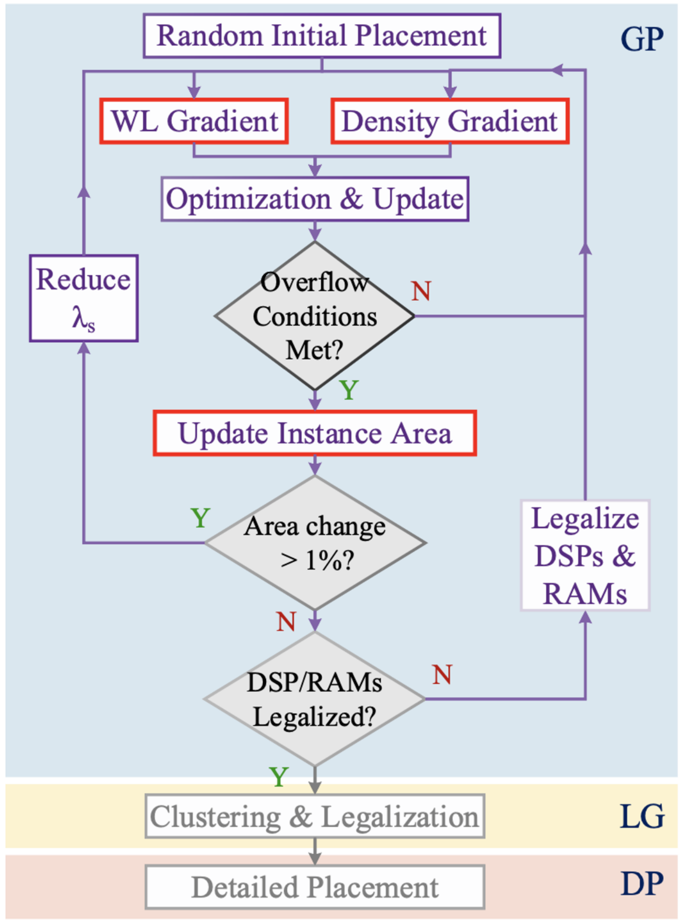
DREAMPlaceFPGA

Adopt ASIC framework for FPGA ⇒ Address various challenges

- Heterogenous resource types with discrete site locations
  - Implement different resources as separate electrostatic systems
- Legality constraints for FFs & LUTs
- High fanout nets: Clock nets have > 100k pin connections
- GPU-friendly data structures
  - Connectivity representation & maps → Multiple flat arrays

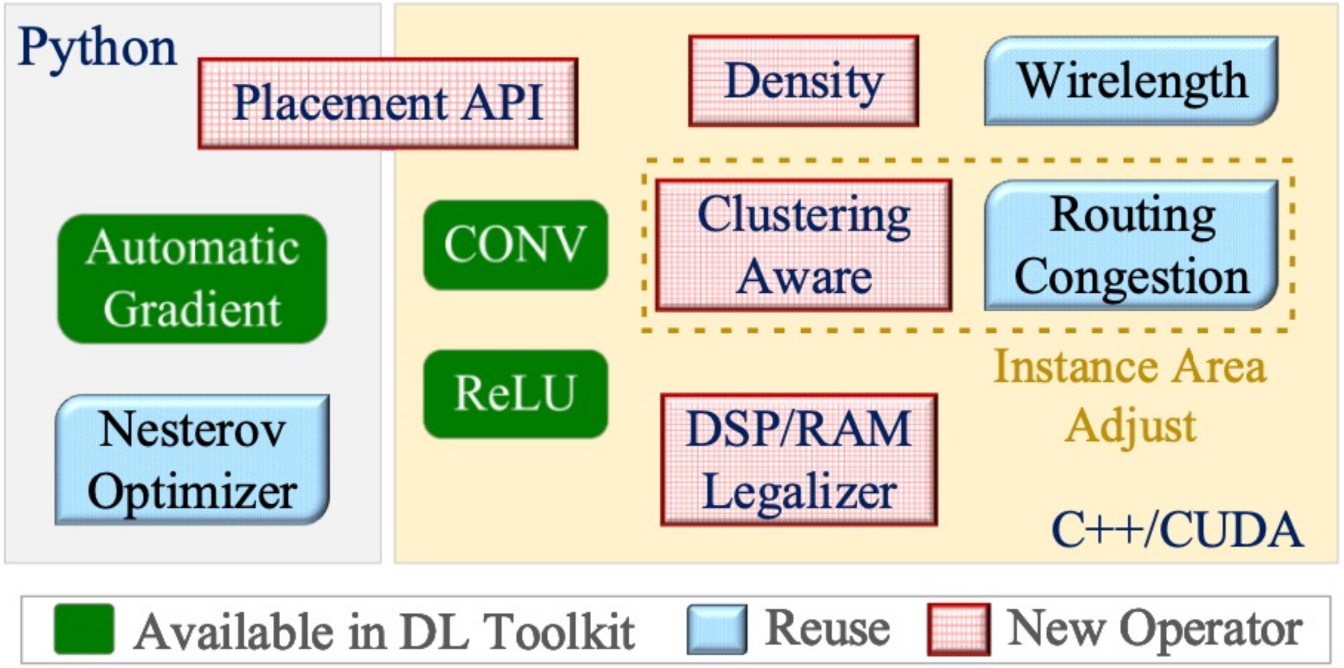
DREAMPlaceFPGA Flow

- Overflow conditions based on instance size:
  - LUT/FF instances: 10%
  - Large DSP/RAM instances: 20%
- Operators in red are GPU accelerated:
  - Wirelength
  - Density
  - Instance Area Update



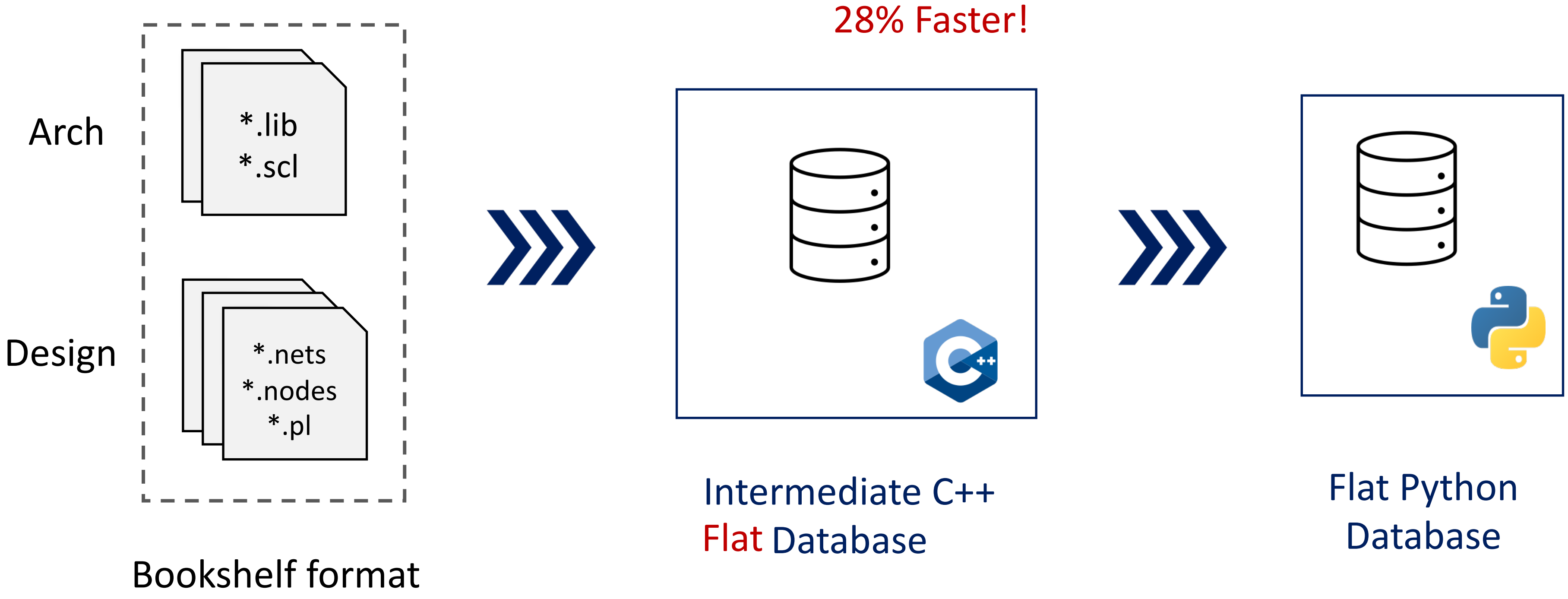
DREAMPLACEFPGA OVERVIEW

Leverage highly optimized deep learning toolkit `PyTorch`

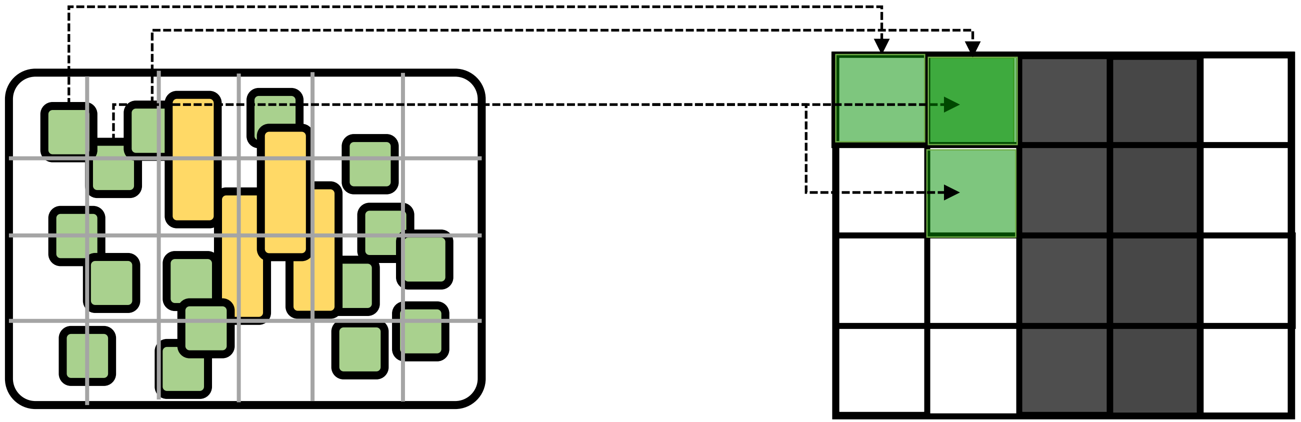


DREAMPlace, Y. Lin^{+}, DAC'2019

DREAMPLACEFPGA: PLACEMENT API



DREAMPlaceFPGA: Density Operator

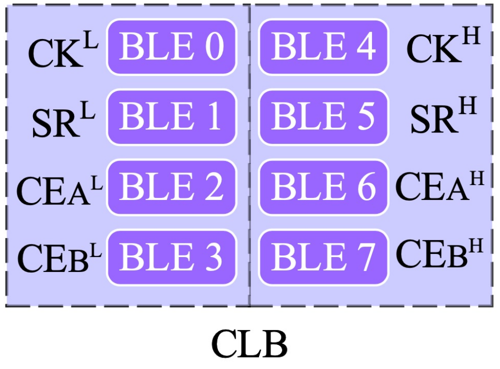


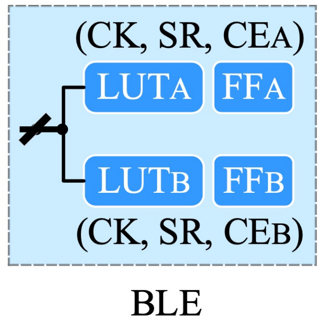
- Each resource type treated as a separate electrostatic system
- Demand map computation done in parallel for instances of a resource type

DREAMPlaceFPGA: Clustering Aware Area Update

Adjust LUT/FF instance areas to ensure legality constraints are met

- Configurable Logic Block (CLB) consists of 8 Basic Logic Elements (BLEs)
  - BLE consists of 2 LUTs & 2 FFs
- For LUTs: maximum I/P count
  - LUT6 occupies the entire BLE
- For FFs: Control set
  - Shared clock (CK), set/reset (SR) and control enable (CE)





Source: Xilinx Ultrascale Architecture

DREAMPLACEFPGA: DSP/RAM LEGALIZER

- Large instances (DSPs & block RAMs) are legalized at the end of GP
- Sparse resources in the considered Xilinx Ultrascale Architecture
  - 1728 block RAMs
  - 768 DSPs
- Min-cost flow (CPU) used to legalize these sparse resource types

Source: Xilinx Ultrascale Architecture

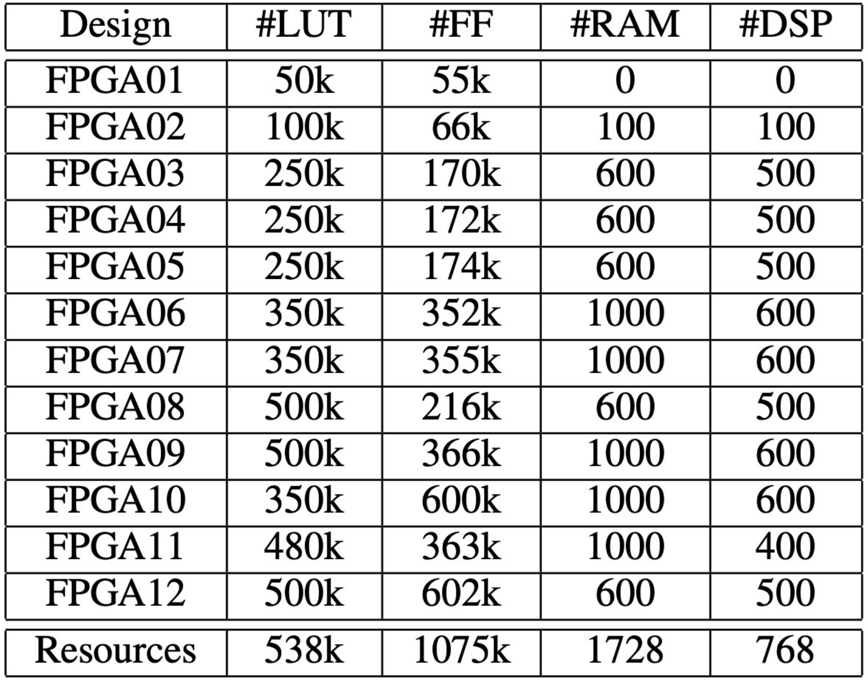
EXPERIMENTAL SETUP:

- Design suite: ISPD'2016 benchmarks
- Placers: elfPlace-CPU, elfPlace-GPU
- Router: *Xilinx Vivado* $^{v2015.4}$

CPU (8T): Intel Core i9-7900X @ 3.30 GHz

GPU: NVIDIA TITAN Xp (Pascal)

- Comparison Metrics
  - Routed Wirelength
  - Placement Runtime

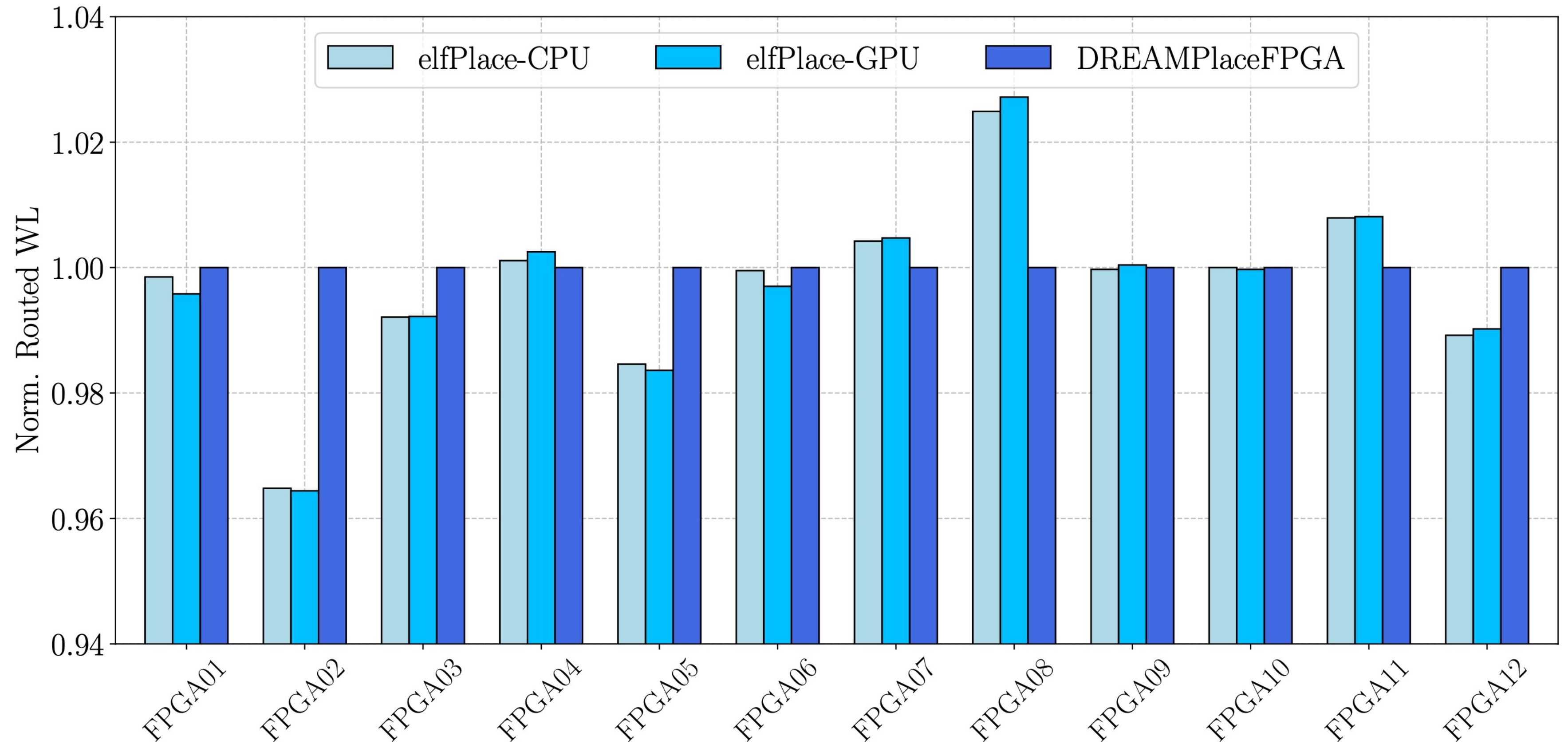


Design	#LUT	#FF	#RAM	#DSP
FPGA01	50k	55k	0	0
FPGA02	100k	66k	100	100
FPGA03	250k	170k	600	500
FPGA04	250k	172k	600	500
FPGA05	250k	174k	600	500
FPGA06	350k	352k	1000	600
FPGA07	350k	355k	1000	600
FPGA08	500k	216k	600	500
FPGA09	500k	366k	1000	600
FPGA10	350k	600k	1000	600
FPGA11	480k	363k	1000	400
FPGA12	500k	602k	600	500
Resources	538k	1075k	1728	768

*LG & DP run on elfPlace-CPU

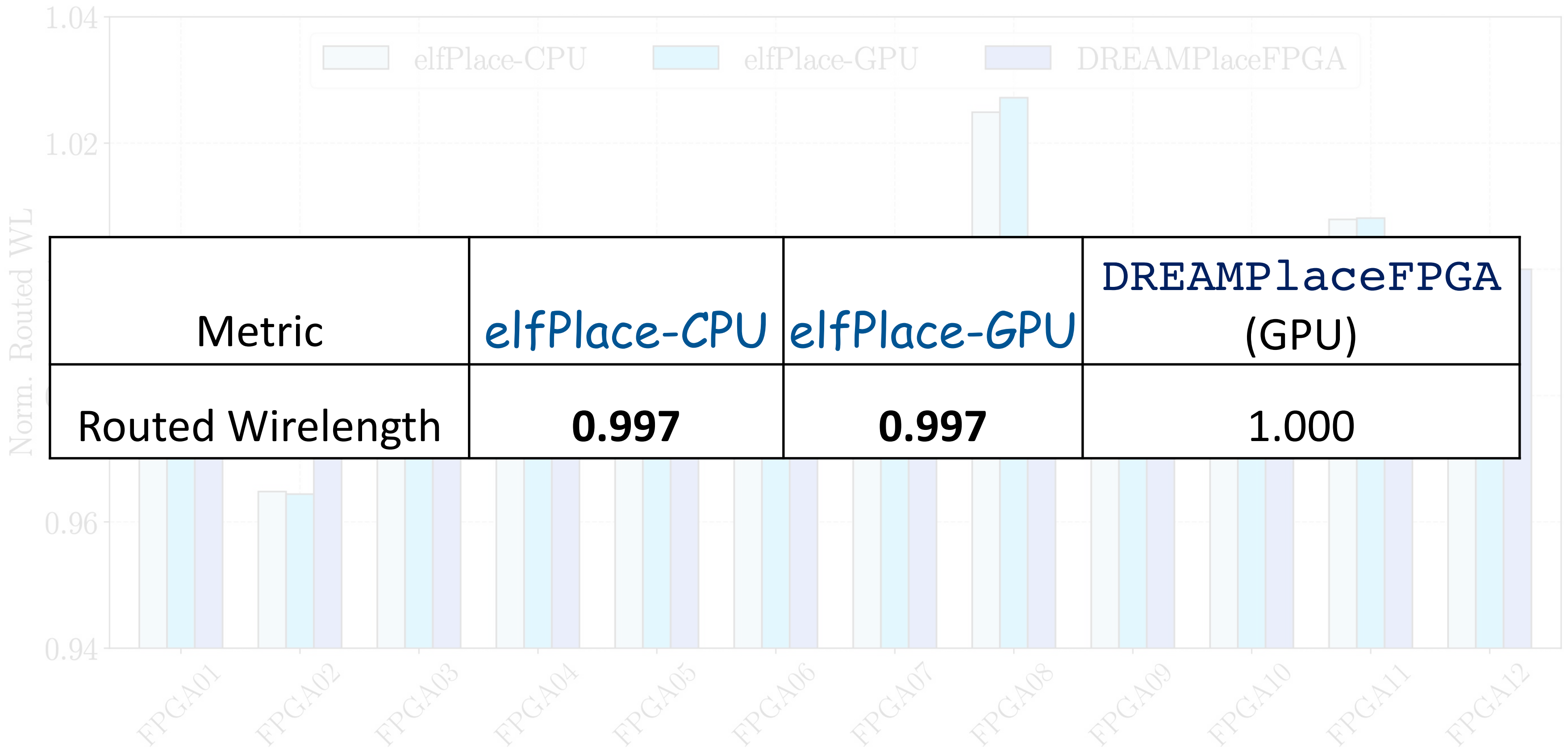
W. Li+, ICCAD'2019; Y. Meng^{+}, TCAD'2021

Results: Routed Wirelength



W. Li+, ICCAD'2019; Y. Meng+, TCAD'2021

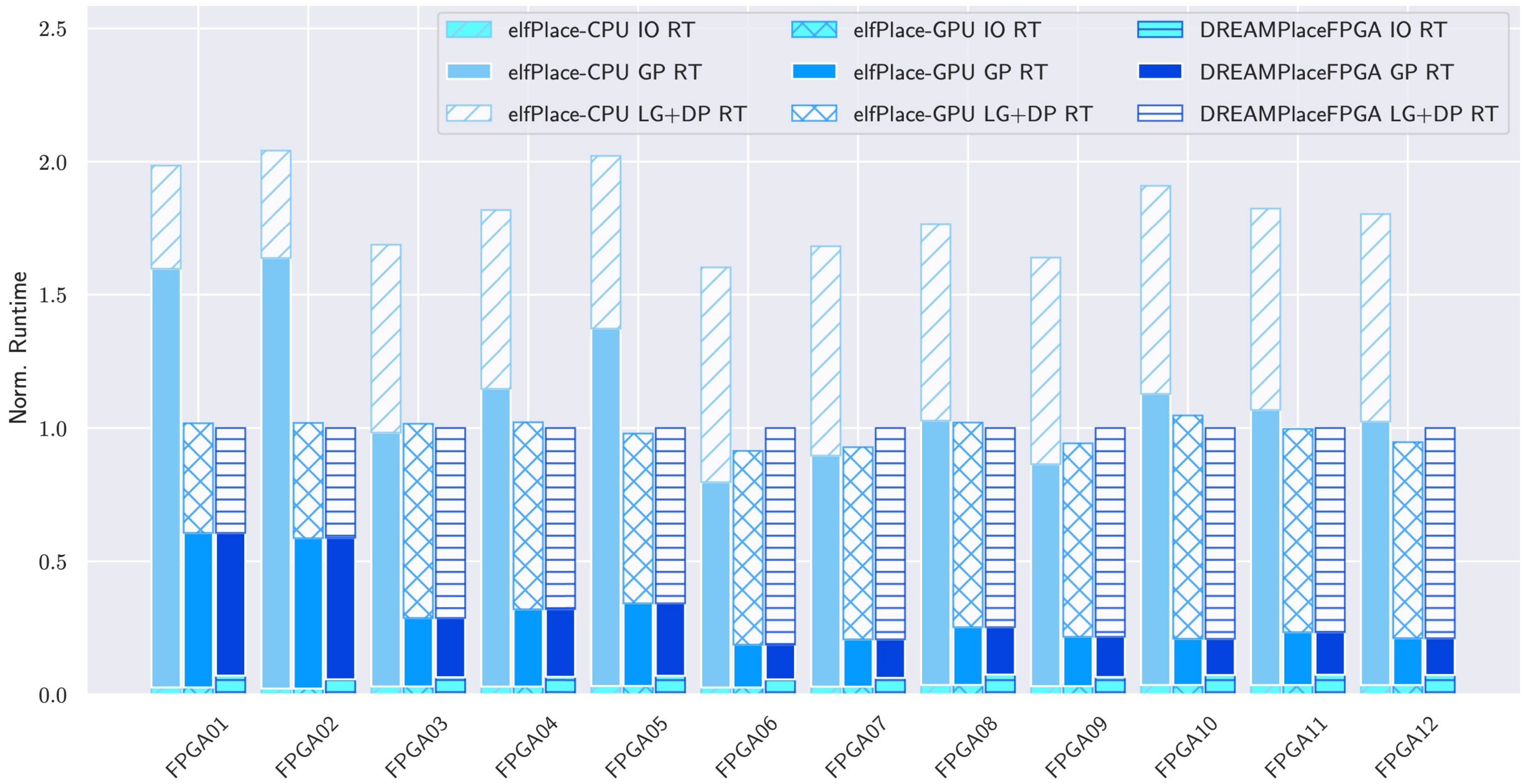
Results: Routed Wirelength



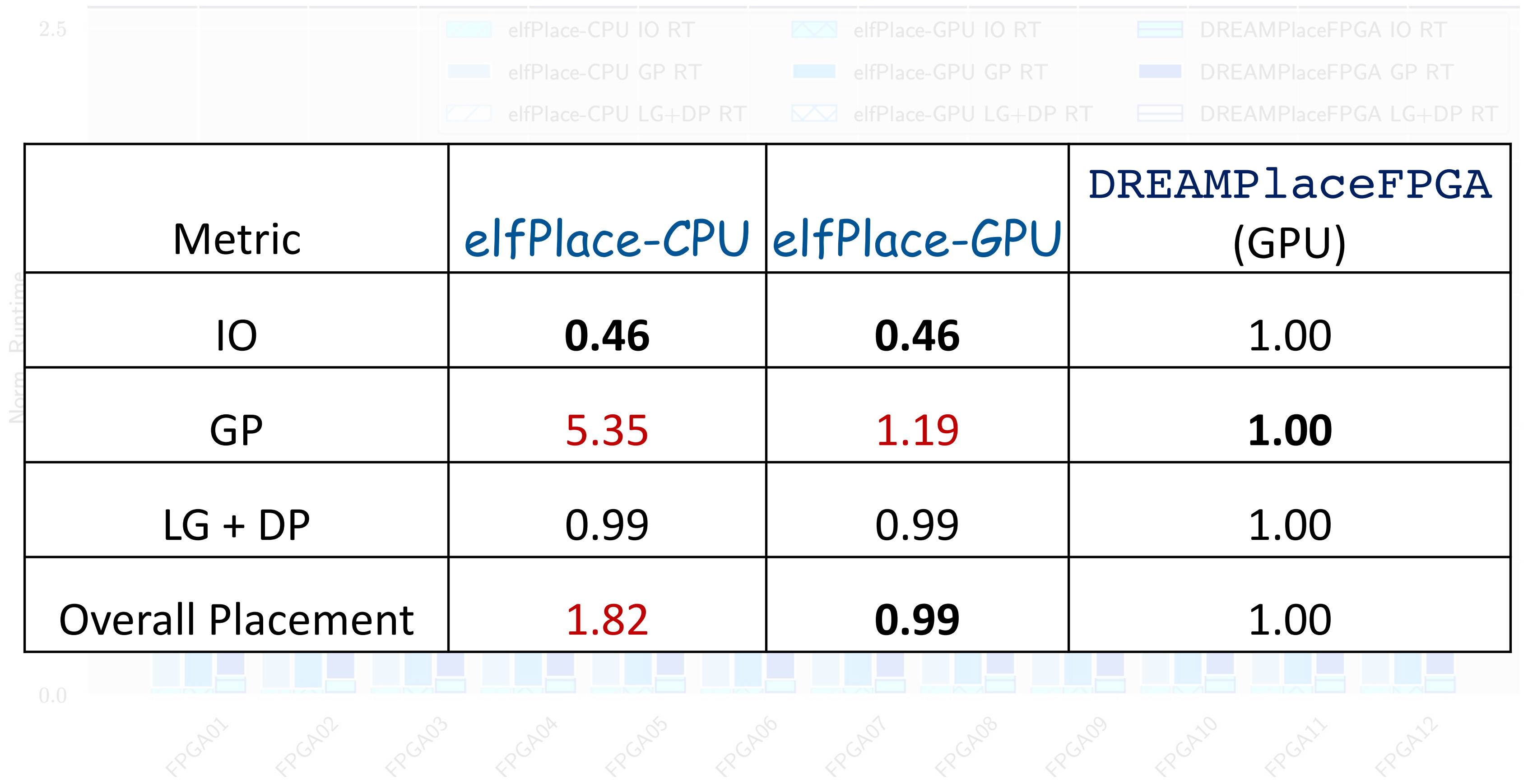
Metric	elfPlace-CPU	elfPlace-GPU	DREAMPlaceFPGA (GPU)
Routed Wirelength	**0.997**	**0.997**	1.000

W. Li+, ICCAD’2019; Y. Meng+, TCAD’2021

Results: Placement Runtime



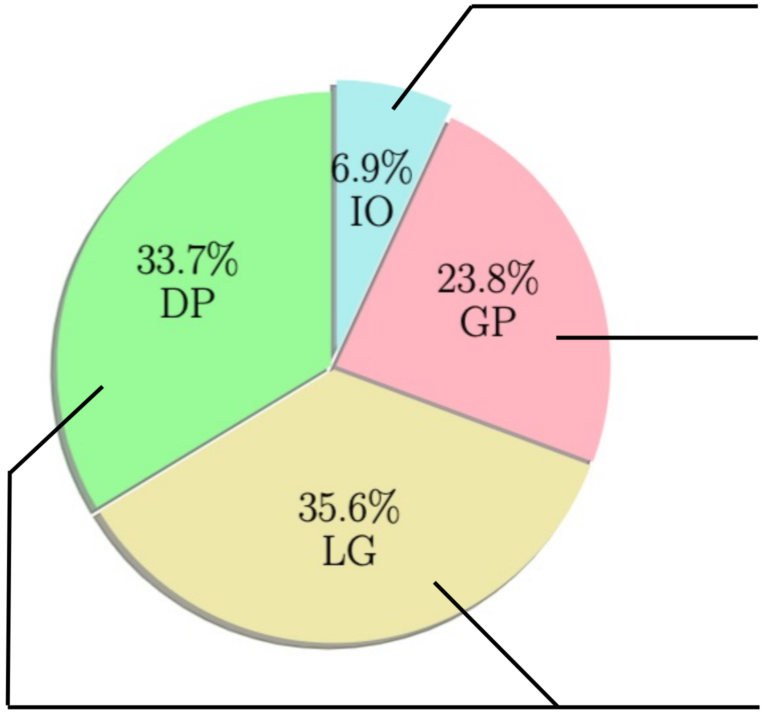
Results: Placement Runtime



Metric	elfPlace-CPU	elfPlace-GPU	DREAMPlaceFPGA (GPU)
IO	**0.46**	**0.46**	1.00
GP	5.35	1.19	**1.00**
LG + DP	0.99	0.99	1.00
Overall Placement	1.82	**0.99**	1.00

W. Li+, ICCAD’2019; Y. Meng^{+}, TCAD’2021

DREAMPlaceFPGA Runtime Breakdown



High data IO movement

Benchmarks → C++ database → Python database

Accelerated Global Placement

~ 1/4th of overall placement runtime

Legalization & Detailed Placement

~ 2/3rd of overall placement runtime

DREAMPlaceFPGA Summary

- Open-source accelerated FPGA placement framework
- Adapts elfPlace to a DL toolkit, inspired by the ASIC DREAMPlace framework
- Outperforms elfPlace-CPU and elfPlace-GPU in terms of global placement runtime with similar quality of results

Metric	elfPlace-CPU	elfPlace-GPU	DREAMPlaceFPGA (GPU)
Routed Wirelength	**0.997**	**0.997**	1.000
Global Placement Runtime	**5.35**	**1.19**	**1.00**
Overall Placement Runtime	**1.82**	**0.99**	1.00

https://github.com/rachelselinar/DREAMPlaceFPGA

Future Work

- Acceleration of other placement stages
  - Legalization
  - Detailed Placement
- Support for different FPGA architectures
  - Only Xilinx Ultrascale architecture support available

Thank You