2023 IEEE 15th International Conference on ASIC

OpenPARF: An Open-Source Placement and Routing Framework for Large-Scale Heterogeneous FPGAs with Deep Learning Toolkit

Jing Mai[1], Jiarui Wang[1], Zhixiong Di[3], Guojie Luo[1], Yun Liang[1], Yibo Lin[1]

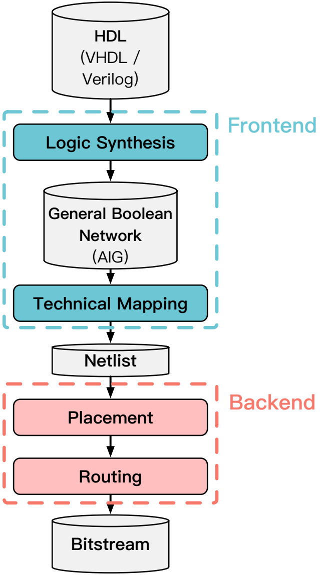
[1]Peking University
[2]Southwest Jiaotong University

jingmai@pku.edu.cn

October 26, 2023

Outline

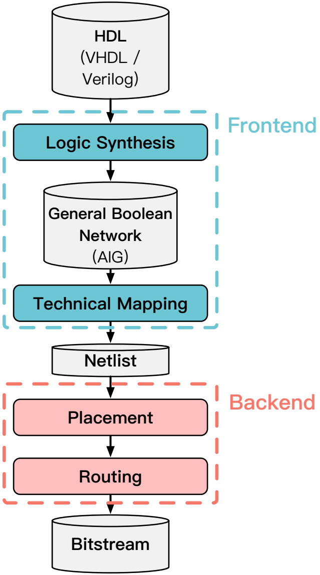
Introduction



HDL

- Hardware design is modeled in a *Hardware Description Language* (HDL)

FPGA CAD Flow

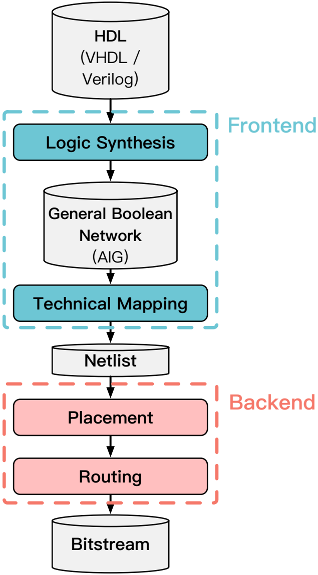


HDL

- Hardware design is modeled in a *Hardware Description Language* (HDL)

Frontend

- A FPGA "compiler" (synthesis tool) translates the HDL into a general Boolean network, e.g, *And-Inverter Graph* (AIG)

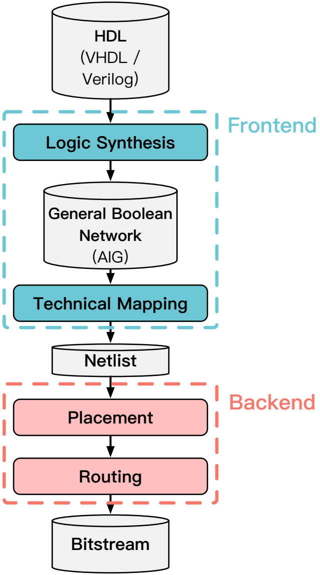


HDL

- Hardware design is modeled in a *Hardware Description Language* (HDL)

Frontend

- A FPGA "compiler" (synthesis tool) translates the HDL into a general Boolean network, e.g, *And-Inverter Graph* (AIG)
- which is then **mapped** to a FPGA technology tailored netlist



HDL

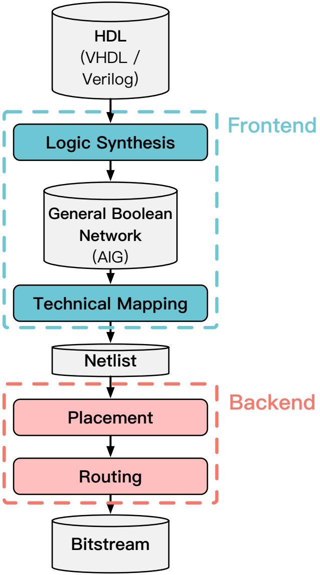
- Hardware design is modeled in a *Hardware Description Language* (HDL)

Frontend

- A FPGA "compiler" (synthesis tool) translates the HDL into a general Boolean network, e.g, *And-Inverter Graph* (AIG)
- which is then **mapped** to a FPGA technology tailored netlist

Backend

- The netlist components are **placed** on the FPGA laytout



HDL

- Hardware design is modeled in a *Hardware Description Language* (HDL)

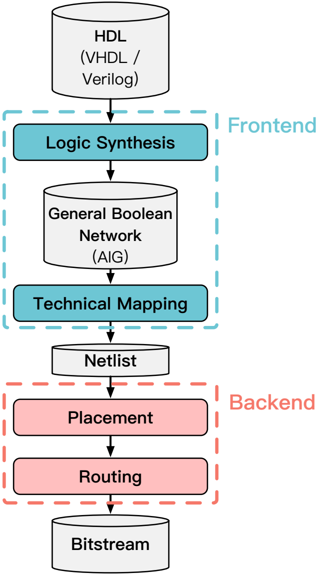
Frontend

- A FPGA "compiler" (synthesis tool) translates the HDL into a general Boolean network, e.g, *And-Inverter Graph* (AIG)
- which is then **mapped** to a FPGA technology tailored netlist

Backend

- The netlist components are **placed** on the FPGA laytout
- and the connecting signals are **routed** through the interconnection network

FPGA CAD Flow



HDL

- Hardware design is modeled in a *Hardware Description Language* (HDL)

Frontend

- A FPGA "compiler" (synthesis tool) translates the HDL into a general Boolean network, e.g, *And-Inverter Graph* (AIG)
- which is then **mapped** to a FPGA technology tailored netlist

Backend

- The netlist components are **placed** on the FPGA laytout
- and the connecting signals are **routed** through the interconnection network
- A bitstream is finally generated for the FPGA configuration

FPGA P&R: the CORE of the FPGA backend CAD flow

- **Placement** significantly determines the final routability and timing performance[1]

[1] Shih-Chun Chen and Yao-Wen Chang (2017). "FPGA placement and routing". In: *Proc. ICCAD*, pp. 914–921.

[2] Kevin E Murray et al. (2015). "Timing-driven titan: Enabling large benchmarks and exploring the gap between academic and commercial CAD". In: *ACM TRETS* 8.2, pp. 1–18.

FPGA P&R: the CORE of the FPGA backend CAD flow

- **Placement** significantly determines the final routability and timing performance[1]
- **Routing** is generally the most time-consuming step, accounting for 41-86% runtime[2]

[1] Shih-Chun Chen and Yao-Wen Chang (2017). "FPGA placement and routing". In: *Proc. ICCAD*, pp. 914–921.

[2] Kevin E Murray et al. (2015). "Timing-driven titan: Enabling large benchmarks and exploring the gap between academic and commercial CAD". In: *ACM TRETS* 8.2, pp. 1–18.

FPGA P&R: the CORE of the FPGA backend CAD flow

- **Placement** significantly determines the final routability and timing performance[1]
- **Routing** is generally the most time-consuming step, accounting for 41-86% runtime[2]
- FPGA P&R is deeply tied to the hardware architecture,
- and every FPGA manufacturer needs tailored P&R software → Source of advantage!

FPGA Manufacturer	Tailored P&R Tool
AMD XILINX	XILINX VITIS
intel	ALTERA QUARTUS II
Lattice Semiconductor Corporation	LATTICE DIAMOND DESIGN SOFTWARE

[1] Shih-Chun Chen and Yao-Wen Chang (2017). "FPGA placement and routing". In: *Proc. ICCAD*, pp. 914–921.

[2] Kevin E Murray et al. (2015). "Timing-driven titan: Enabling large benchmarks and exploring the gap between academic and commercial CAD". In: *ACM TRETS* 8.2, pp. 1–18.

Contributions

- We propose **OpenPARF**, an open-source academic FPGA P&R engine that supports complex industrial FPGA architectures with *state-of-the-art* (SOTA) placement and routing algorithms.

Contributions

- We propose **OpenPARF**, an open-source academic FPGA P&R engine that supports complex industrial FPGA architectures with *state-of-the-art* (SOTA) placement and routing algorithms.
- We implement it with the deep learning toolkit `PyTorch`, running on both CPU and GPU platforms with highly flexibility and efficiency.

Contributions

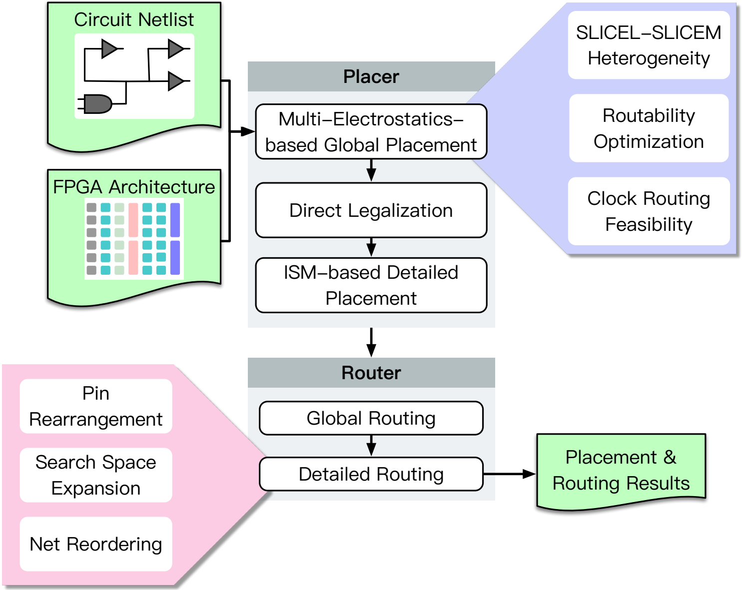
- We propose **OpenPARF**, an open-source academic FPGA P&R engine that supports complex industrial FPGA architectures with *state-of-the-art* (SOTA) placement and routing algorithms.
- We implement it with the deep learning toolkit `PyTorch`, running on both CPU and GPU platforms with highly flexibility and efficiency.
- We are capable of achieving superior placement results under various constraints such as routability, clock feasibility, and SLICEL-SLICEM heterogeneity

Contributions

- We propose **OpenPARF**, an open-source academic FPGA P&R engine that supports complex industrial FPGA architectures with *state-of-the-art* (SOTA) placement and routing algorithms.
- We implement it with the deep learning toolkit `PyTorch`, running on both CPU and GPU platforms with highly flexibility and efficiency.
- We are capable of achieving superior placement results under various constraints such as routability, clock feasibility, and SLICEL-SLICEM heterogeneity
- We can reduce 0.4-12.7% routed wirelength as well as more than 2$\times$ speedup in placement efficiency compared with other SOTA academic P&R engines.

The OpenPARF Framework

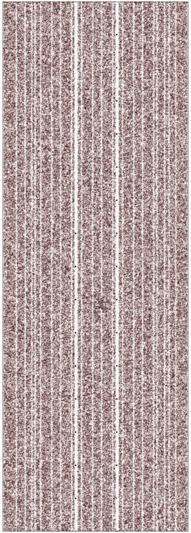
OpenPARF Overview



Hightlighted Features

State-of-the-art P&R Algorithms

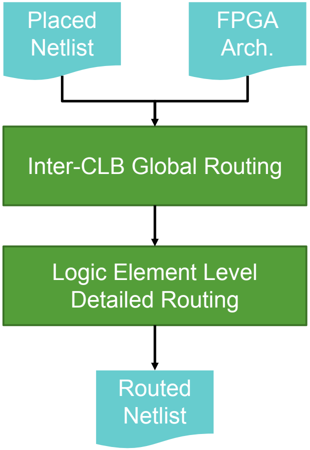
- **SOTA** multi-electrostatics-based global placement



Hightlighted Features

State-of-the-art P&R Algorithms

- **SOTA** multi-electrostatics-based global placement
- **SOTA** two-stage CLB-level FPGA routing



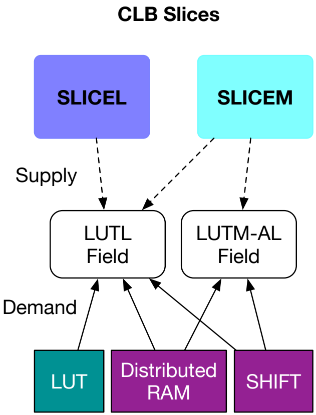
Hightlighted Features

State-of-the-art P&R Algorithms

- **SOTA** multi-electrostatics-based global placement
- **SOTA** two-stage CLB-level FPGA routing

More P&R Constraints

- SLICEL-SLICEM heterogeneity



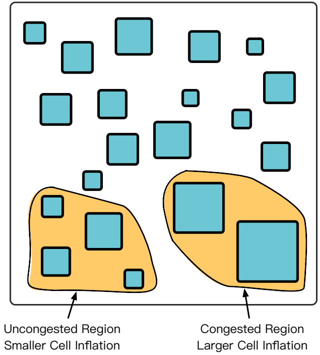
Hightlighted Features

State-of-the-art P&R Algorithms

- **SOTA** multi-electrostatics-based global placement
- **SOTA** two-stage CLB-level FPGA routing

More P&R Constraints

- SLICEL-SLICEM heterogeneity
- Routability optimization



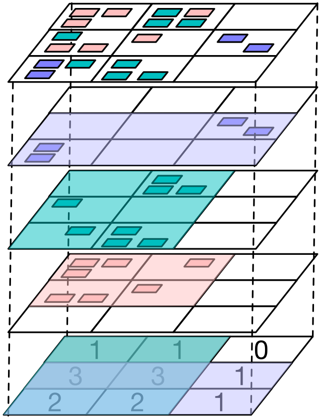
Hightlighted Features

State-of-the-art P&R Algorithms

- **SOTA** multi-electrostatics-based global placement
- **SOTA** two-stage CLB-level FPGA routing

More P&R Constraints

- SLICEL-SLICEM heterogeneity
- Routability optimization
- Clock routing feasibility
- ...



Multi-Electrostatic-based Placement Flow[4]

Constrained Optimization Formulation

- Nonlinear placers minimize wirelength

$$\min_{\boldsymbol{x},\boldsymbol{y}} \widetilde{W}(\boldsymbol{x},\boldsymbol{y}) = \sum_{e \in E} \mathrm{WL}_e(\boldsymbol{x},\boldsymbol{y}), \tag{1}$$

- ... while subject to `ePlace`-series [Lu+, TCAD'15] density constraints for each object type $s \in S = \{\mathrm{LUT}, \mathrm{FF}, \mathrm{DSP}, \mathrm{BRAM}, \mathrm{IO}\}$

$$\text{s.t. } \Phi_s(\boldsymbol{x},\boldsymbol{y}) = 0, \quad \forall s \in S, \tag{2}$$

Augmented Lagrangian Method

- Constrained $\rightarrow$ unconstrained [3]

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad \mathcal{L}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\lambda}) = \widetilde{W}(\boldsymbol{x},\boldsymbol{y}) + \sum_{s \in S} \lambda_s \left(\Phi_s + \frac{1}{2} \mathcal{C}_s \Phi_s^2\right) \tag{3}$$

- Update $\boldsymbol{x}$ and $\boldsymbol{y}$ by nonlinear optimization method (e.g., *Nestrov* method)
- ... and gradually increase $\boldsymbol{\lambda}$ to resolve the constraints

[3] $\mathcal{C}_s$: penalty coefficient, $\boldsymbol{\lambda}$: Lagrangian multipliers.

[4] Jing Mai et al. (2022). "Multi-electrostatic FPGA placement considering SLICEL-SLICEM heterogeneity and clock feasibility". In: *Proc. DAC*, pp. 649–654.

SLICEL-SLICEM Heterogeneity

Extended Electrostatic Fields

- Recall the multi-electrostatic-based placement flow ...

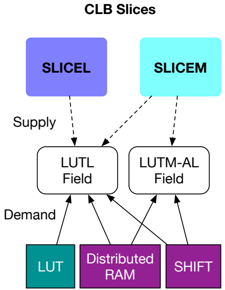
$$\min_{\boldsymbol{x},\boldsymbol{y}} \widetilde{W}(\boldsymbol{x},\boldsymbol{y}) \quad \text{s.t. } \Phi_s(\boldsymbol{x},\boldsymbol{y}) = 0, \quad \forall s \in S \tag{4}$$

- Extend the electrostatic fields as

$$S = \{\text{LUTL}, \text{LUTM-AL}, \text{DSP}, \text{BRAM}, \text{IO}\} \tag{5}$$

Asymmetrical Demand and Supply Attriburtes

- Demand (cell type)
  - `LUTL` field: LUT, Distributed RAM, and SHIFT
  - `LUTM-AL` field: Distributed RAM and SHIFT
- Supply (site type)
  - `LUTL` field: SLICEL and SLICEM
  - `LUTM-AL` field: SLICEM



SLICEL-SLICEM Heterogeneity (II)

Asymmetrical coeffect of `LUTL` and `LUTM-AL`

- LUT can be placed in SLICEL or SLICEM
- Distributed RAM and SHIFT can only be placed in SLICEM

LUT | SHIFT | SLICEL Column | SLICEM Column

	LUTL Field		LUTM-AL Field		Two Fields
	ρ_{LUT}	Φ_{LUT}	ρ_{LUTM_AL}	Φ_{LUTM_AL}	$\Phi_{LUT} + \Phi_{LUTM\_AL}$
Initial		-		-	-
Solution I ✓		Low		Low	Low
Solution II ✓		Low		Low	Low
Solution III ✗		Low		High	High
Solution IV ✗		Low		High	High

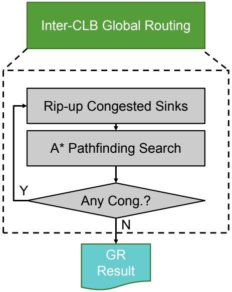
Two-stage CLB-level Routing

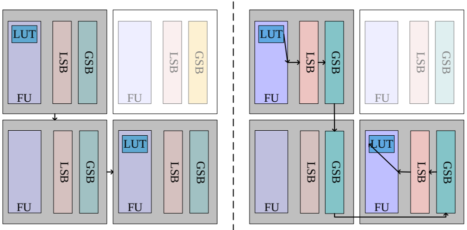
Inter-CLB level global routing

- Coarse-grained routing graph
- Provide inter-CLB routing topology

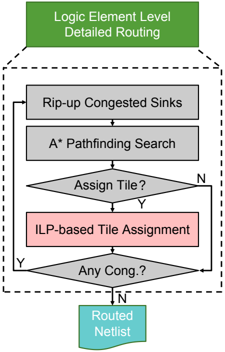
Logic element level detailed routing

- Fine-grained routing graph
- Generate final routing results
- ILP-based tile assignment is proposed to remedy congestion



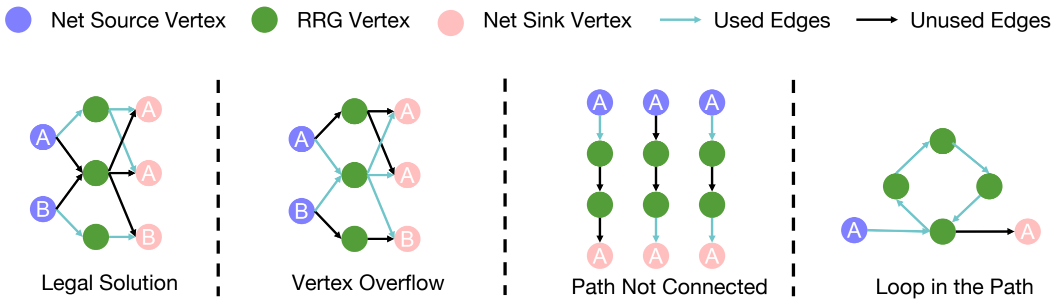


(a) (b)



Problem Formulation

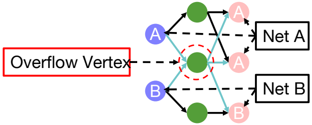
- Route multiple nets inside a tile and its neighbor tile concurrently
  - No overflow vertices
  - Paths must be connected
  - No loop in the paths



Integer Linear Programming (ILP) Modeling

1. No overflow vertex

$$\sum_{e,j} R_{e,j} \leq \mathrm{cap}(v), e \in \mathrm{FI}(v) \quad (6)$$



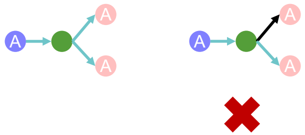
Integer Linear Programming (ILP) Modeling

1. No overflow vertex

$$\sum_{e,j} R_{e,j} \leq \text{cap}(v), e \in \text{FI}(v) \tag{6}$$

2. Each sink of each net is routed

$$S_{e,j,k} \leq R_{e,j}, k \in \text{SINK}(j) \tag{7}$$



Integer Linear Programming (ILP) Modeling

1. No overflow vertex

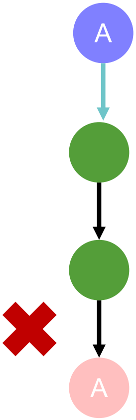
$$\sum_{e,j} R_{e,j} \leq \text{cap}(v), e \in \text{FI}(v) \tag{6}$$

2. Each sink of each net is routed

$$S_{e,j,k} \leq R_{e,j}, k \in \text{SINK}(j) \tag{7}$$

3. The signal is sent from source pin of each net

$$\sum_{e,j,k} S_{e,j,k} = 1, e \in \text{FO}(v), v = \text{SOURCE}(j), \forall k \in \text{SINK}(j) \tag{8}$$



Integer Linear Programming (ILP) Modeling

1. No overflow vertex

$$\sum_{e,j} R_{e,j} \leq \text{cap}(v), e \in \text{FI}(v) \tag{6}$$

2. Each sink of each net is routed

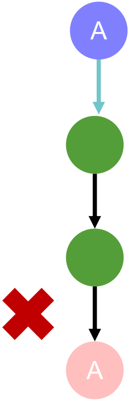
$$S_{e,j,k} \leq R_{e,j}, k \in \text{SINK}(j) \tag{7}$$

3. The signal is sent from source pin of each net

$$\sum_{e,j,k} S_{e,j,k} = 1, e \in \text{FO}(v), v = \text{SOURCE}(j), \forall k \in \text{SINK}(j) \tag{8}$$

4. The signal is received at each sink pin of each net

$$\sum_{e,j,k} S_{e,j,k} = 1, e \in \text{FI}(v), v = \text{SINK}(j,k) \tag{9}$$



Integer Linear Programming (ILP) Modeling

1. No overflow vertex

$$\sum_{e,j} R_{e,j} \leq \text{cap}(v), e \in \text{FI}(v) \tag{6}$$

2. Each sink of each net is routed

$$S_{e,j,k} \leq R_{e,j}, k \in \text{SINK}(j) \tag{7}$$

3. The signal is sent from source pin of each net

$$\sum_{e,j,k} S_{e,j,k} = 1, e \in \text{FO}(v), v = \text{SOURCE}(j), \forall k \in \text{SINK}(j) \tag{8}$$

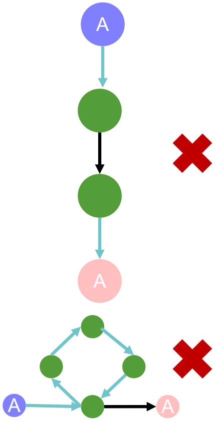
4. The signal is received at each sink pin of each net

$$\sum_{e,j,k} S_{e,j,k} = 1, e \in \text{FI}(v), v = \text{SINK}(j,k) \tag{9}$$

5. There is a path from source pin to each sink pin and no loop

$$\sum_{e_{in}} S_{e_{in},j,k} = \sum_{e_{out}} S_{e_{out},j,k}, \tag{10}$$

$$e_{in} \in \text{FI}(v), e_{out} \in textFO(v), v \neq \text{SOURCE}(j), v \notin \text{SINK}(j)$$



Code Components

1. **openparf**
   The core placement and routing tool
2. **openparf.ops**
   A collection of operators that allow the implementation of various PR algorithms
3. **openparf.placement**
   A set of APIs for performing placement tasks
4. **openparf.routing**
   A set of APIs for performing routing tasks
5. **openparf.py_utils**
   Provides other utility functions for Python convenience

README.md

OpenPARF

OpenPARF is an open-source FPGA placement and routing framework build upon the deep learning toolkit PyTorch. It is designed to be flexible, efficient, and extensible.

Experimental Results

Experiments Setup (I)

Implementation

- C++ & Python
- Build upon `Pytorch` for agile gradient computation

Machine

- Intel(R) Xeon(R) Gold 6230 CPUs (2.10 GHz, 40 cores)
- 512GB RAM
- One NVIDIA RTX 2080Ti GPU

Experiments Setup (II)

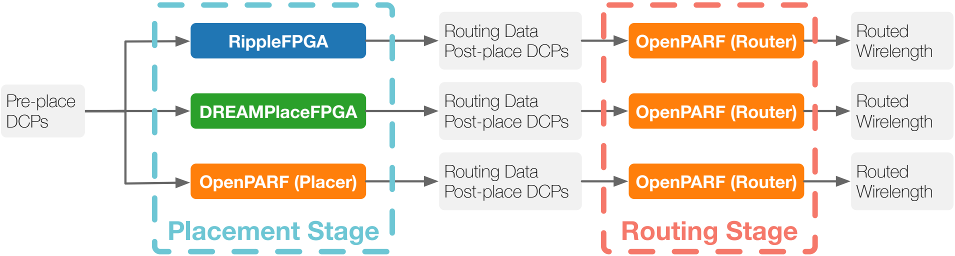
Benchmark Suite

- ISPD 2016 Routability-Driven FPGA Placement Contest [Yang+, ISPD'16]
- ISPD 2017 Clock-Aware FPGA Placement Contest [Yang+, ISPD'17]
- SLICEL-SLICEM Structure-Aware Industrial Benchmarks

Placers for Comparison

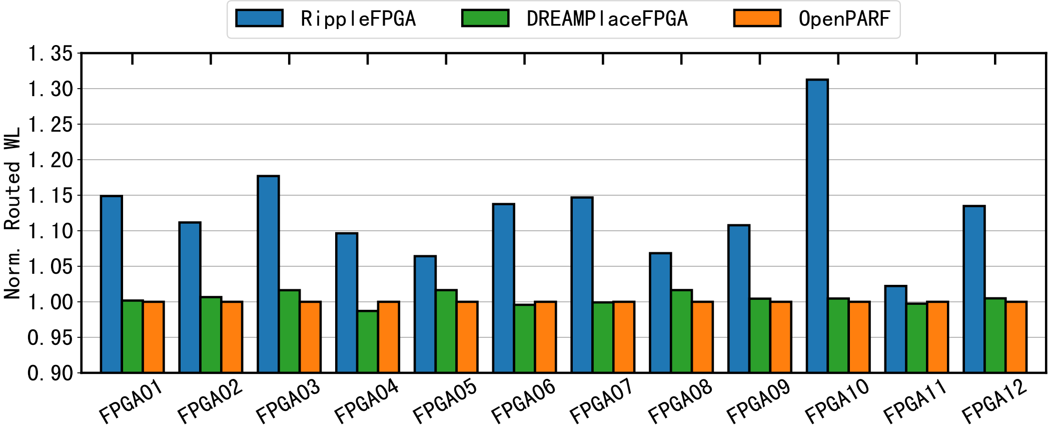
- `RippleFPGA` [Chen+, TCAD'18]
- `DREAMPlaceFPGA` [Rajarathnam+, ISPD'23]

Evaluation Flow



Routed Wirelength Comparison on ISPD 2016

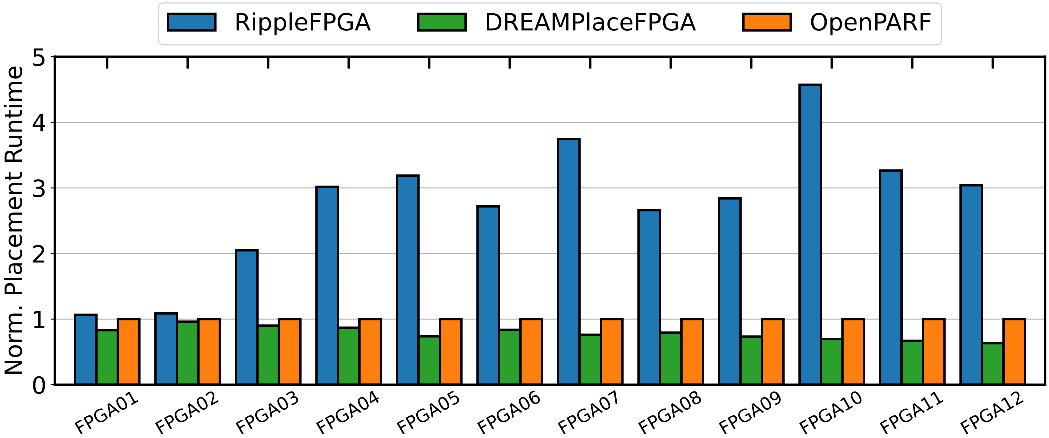
- 12.7% better than `RippleFPGA`
- 0.4% better than `DREAMPlaceFPGA`



`OpenPARF` significantly outperforms other placers on routed wirelength.

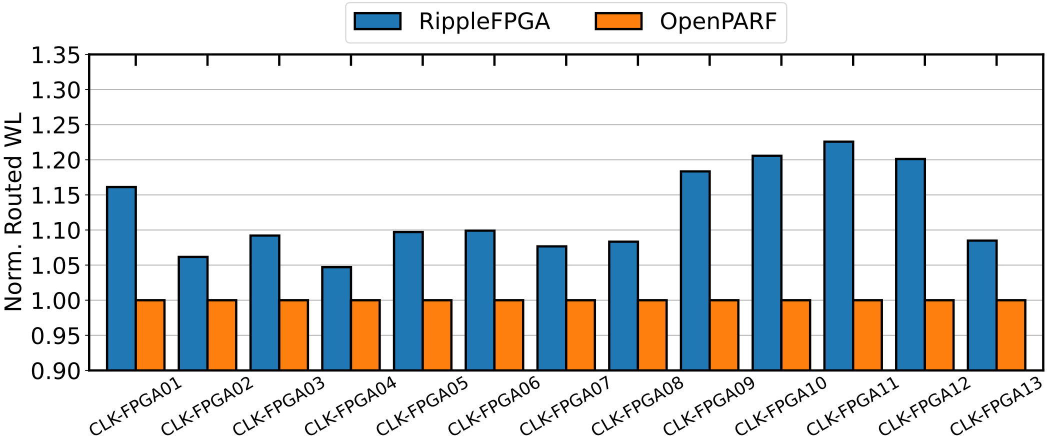
Placement Runtime Comparison on ISPD 2016

- 2.771× faster RippleFPGA
- 1.272× slower than DREAMPlaceFPGA



Routed Wirelength Comparison on ISPD 2017 [5]

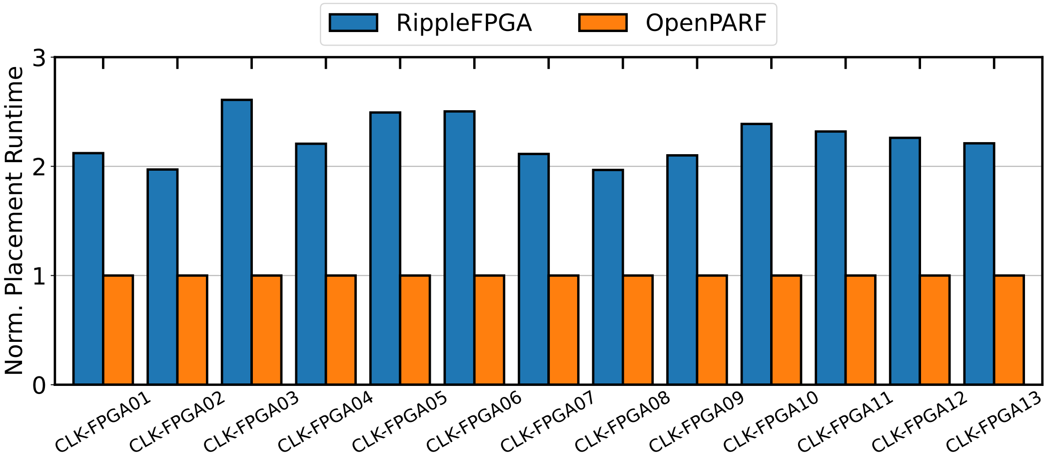
- 12.8% better than `RippleFPGA`



[5] `DREAMPlaceFPGA` is not applicable to this benchmark suite.

Placement Runtime Comparison on ISPD 2017

- 2.251× faster than `RippleFPGA`



Experiments on SLICEL-SLICEM Industrial Benchmarks

OpenPARF show notable performance and efficiency on industrial benchmarks.

- 21K - 284K cells
- Distributed RAMs and SHIFTs (SLICEM tailored cell types)
- Cascaded DSPs, BRAMs and CARRYs

Design	#LUT/#FF/#BRAM/#DSP	#Distributed RAM + #SHIFT	#Net	OpenPARF		
				PRT [6]	RRT [7]	RWL [8]
IND01	17K/11K/0/13	9	52492	72.36	10	90
IND02	11K/10K/0/24	6	26678	77.82	15	100
IND03	109K/12K/0/0	0	121554	109.54	108	1021
IND04	29K/17K/0/16	218	60968	69.39	19	283
IND05	64K/191K/64/928	29K	371808	126.38	109	2360
IND06	112K/65K/21/0	0	221182	88.28	176	1593
IND07	40K/156K/89/768	26K	294075	140.33	68	1450

[6] Placement Runtime (Seconds).
[7] Routing Runtime (Minutes).
[8] Routed Wirelength.

Conclusion & Future Work

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs

Conclusion & Future Work

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs
- We build `OpenPARF` upon the deep learning toolkit **PyTorch** for agile gradient computation and flexible programming interfaces

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs
- We build `OpenPARF` upon the deep learning toolkit **PyTorch** for agile gradient computation and flexible programming interfaces
- We resolve the SLICEL-SLICEM heterogeneity by the SOTA asymmetrical multi-electrostatic FPGA placement algorithms [Mai+, DAC'22]

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs
- We build `OpenPARF` upon the deep learning toolkit **PyTorch** for agile gradient computation and flexible programming interfaces
- We resolve the SLICEL-SLICEM heterogeneity by the SOTA asymmetrical multi-electrostatic FPGA placement algorithms [Mai+, DAC'22]
- We harness the nested Lagrangian relaxation methodology to resolve multiple placement objectives

Conclusion & Future Work

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs
- We build `OpenPARF` upon the deep learning toolkit **PyTorch** for agile gradient computation and flexible programming interfaces
- We resolve the SLICEL-SLICEM heterogeneity by the SOTA asymmetrical multi-electrostatic FPGA placement algorithms [Mai+, DAC'22]
- We harness the nested Lagrangian relaxation methodology to resolve multiple placement objectives
- We settle the large-scale irregular CLB-level FPGA routing problem by the SOTA two-stage negotiation-based routing algorithms [Wang+, ASP-DAC'23]

Conclusion & Future Work

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs
- We build `OpenPARF` upon the deep learning toolkit **PyTorch** for agile gradient computation and flexible programming interfaces
- We resolve the SLICEL-SLICEM heterogeneity by the SOTA asymmetrical multi-electrostatic FPGA placement algorithms [Mai+, DAC'22]
- We harness the nested Lagrangian relaxation methodology to resolve multiple placement objectives
- We settle the large-scale irregular CLB-level FPGA routing problem by the SOTA two-stage negotiation-based routing algorithms [Wang+, ASP-DAC'23]

Future Work

- GPU-accelerated legalization / routing

Conclusion & Future Work

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs
- We build `OpenPARF` upon the deep learning toolkit **PyTorch** for agile gradient computation and flexible programming interfaces
- We resolve the SLICEL-SLICEM heterogeneity by the SOTA asymmetrical multi-electrostatic FPGA placement algorithms [Mai+, DAC'22]
- We harness the nested Lagrangian relaxation methodology to resolve multiple placement objectives
- We settle the large-scale irregular CLB-level FPGA routing problem by the SOTA two-stage negotiation-based routing algorithms [Wang+, ASP-DAC'23]

Future Work

- GPU-accelerated legalization / routing
- Look-ahead Timing Prediction (LATP) for timing-driven placement

Conclusion & Future Work

Conclusion

- **OpenPARF**: an open-source placement and routing framework for large-scale FPGAs
- We build `OpenPARF` upon the deep learning toolkit **PyTorch** for agile gradient computation and flexible programming interfaces
- We resolve the SLICEL-SLICEM heterogeneity by the SOTA asymmetrical multi-electrostatic FPGA placement algorithms [Mai+, DAC'22]
- We harness the nested Lagrangian relaxation methodology to resolve multiple placement objectives
- We settle the large-scale irregular CLB-level FPGA routing problem by the SOTA two-stage negotiation-based routing algorithms [Wang+, ASP-DAC'23]

Future Work

- GPU-accelerated legalization / routing
- Look-ahead Timing Prediction (LATP) for timing-driven placement
- Fence region-aware placement

THANK YOU!