A Robust FPGA Router with Concurrent Intra-CLB Rerouting

Jiarui Wang[1], Jing Mai[1], Zhixiong Di[2], Yibo Lin[1]

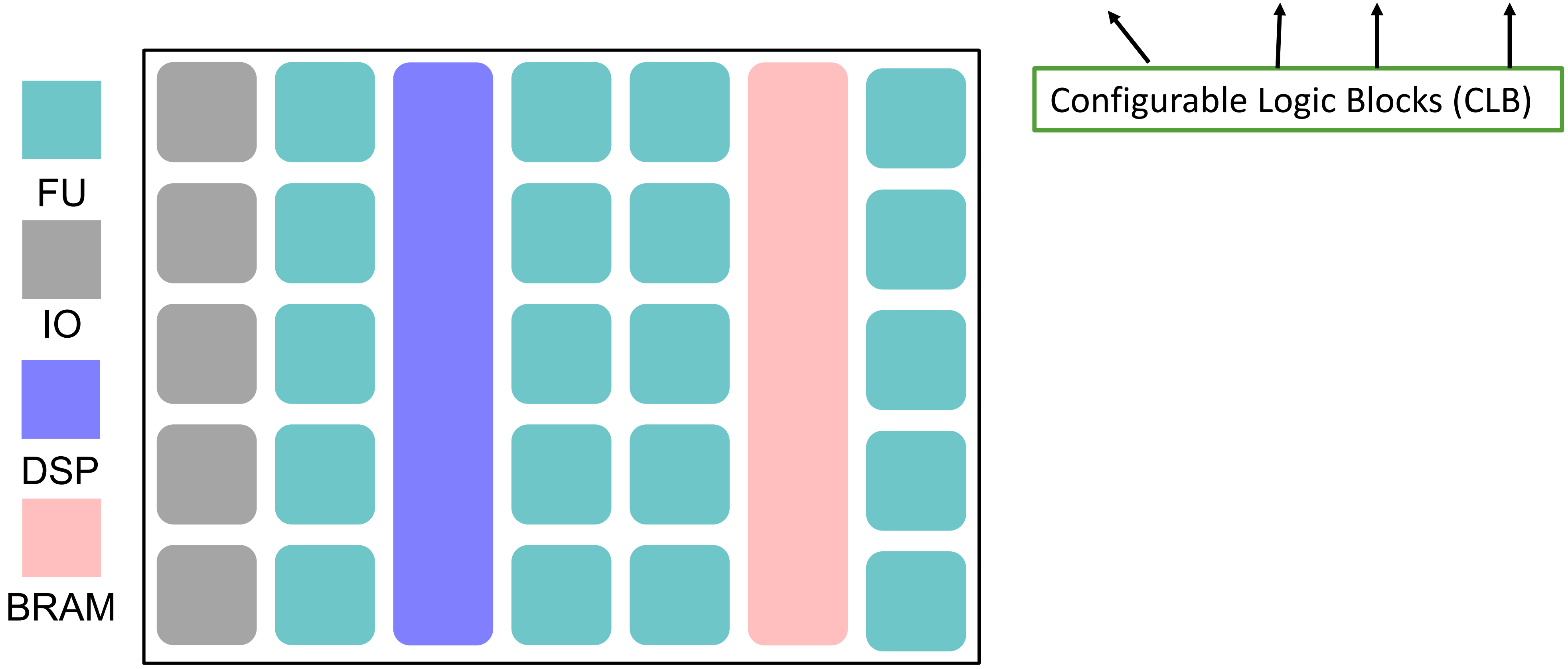
[1]Peking University

[2]Southwest Jiaotong University

jiaruiwang@pku.edu.cn

Modern FPGA Layout

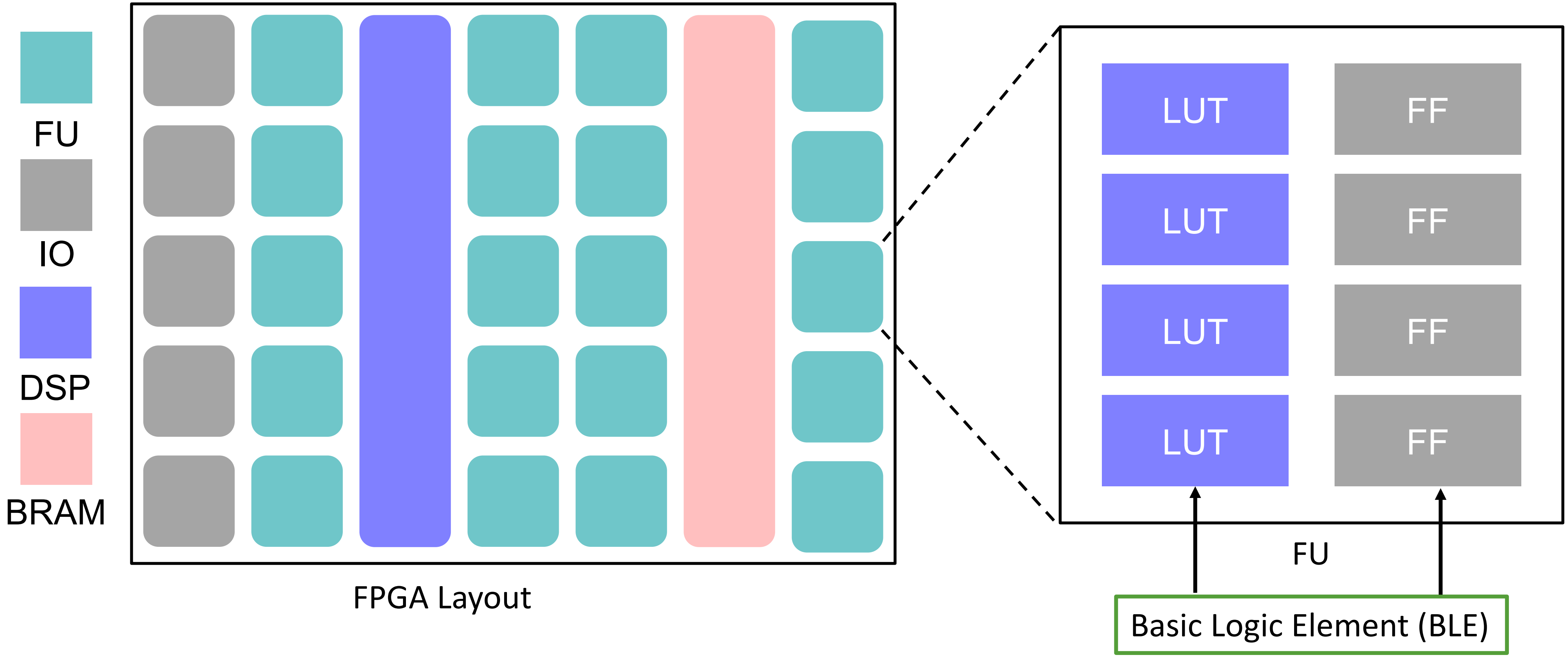
- Contain heterogeneous resources, Like function unit (FU), IO, DSP, BRAM,…



FPGA Layout

Modern FPGA Layout

- Contain heterogeneous resources, Like function unit (FU), IO, DSP, BRAM,…



FPGA Routing Problem

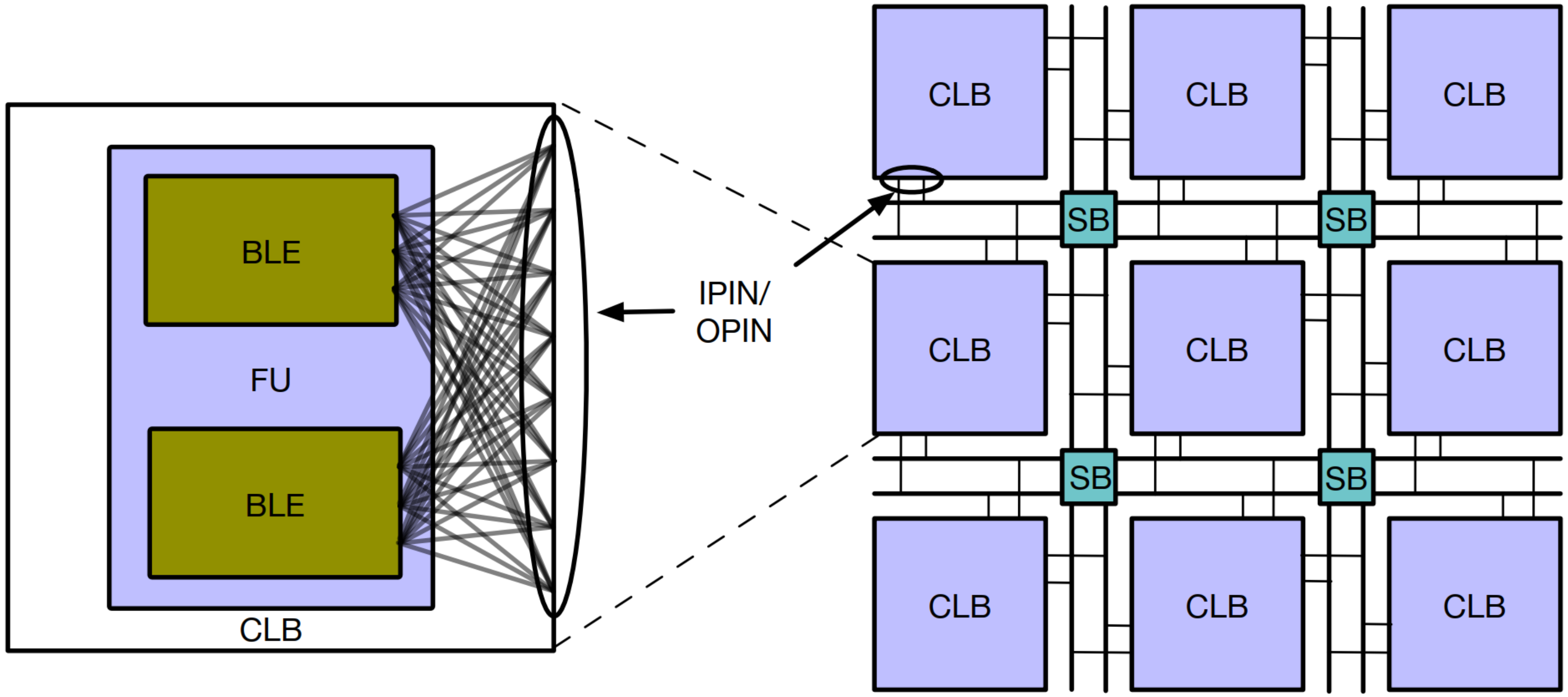
- Target:
  - Find logic paths between logic elements inside CLBs
- Importance:
  - Performance impact
    - Wirelength/Timing/Power/…
  - Runtime consuming
    - 41%~86% runtime in FPGA CAD flow [Murray et. al. TRTS'15]
  - Scalability
    - Millions of logic cells and nets

Related Works

- Open-source academic routers
  - VTR 8.0 [Murray et. al. ASPDAC ’20]
  - CRoute [Vercruyce et. al. FCCM ’19]
- Routing metric enhancement
  - Rip-up & reroute enhancement [Wang et. al. TCAD ’18]
  - GPU acceleration [Shen et. al. ICCD ’18]
  - Improved routing cost function [Zha et. al. FPGA ’22]

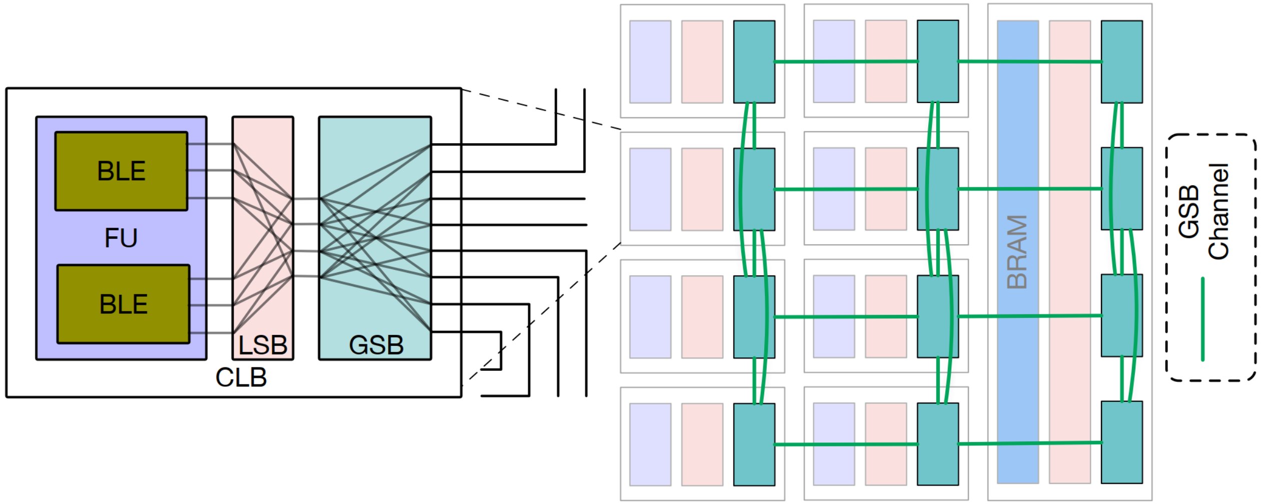
Limitation of Prior Works

- Can only deal with logic-**equivalence** FPGA architecture
  - Each logic pin of a logic element is logic equivalent
  - Can be connected to any input/output pin of CLB



Non Logic-Equivalence FPGA Architecture

- Each logic pin of a logic element can be connected to different I/O logic pin
- Challenge
  - Large search space
  - Limited routing resources
  - Intra-CLB routing congestion



Problem Formulation

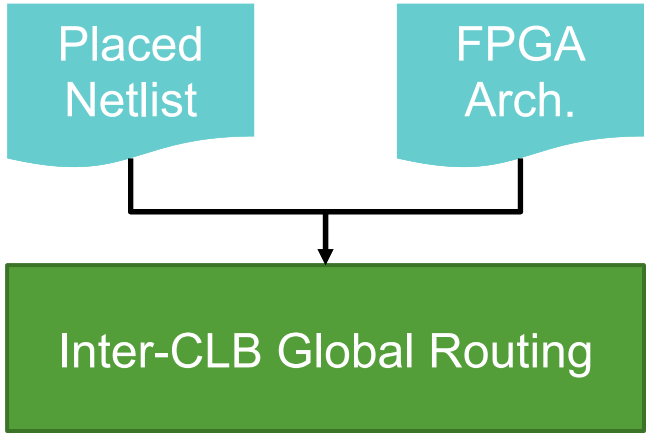
- Input
  - Non logic-equivalence FPGA routing architecture
  - Placed FPGA design
- Output
  - Routed logic path for each logic net
- Target
  - Minimize wirelength
  - Ensure no routing congestion

Our Contribution

- A robust FPGA router can deal with non logic-equivalence FPGA architecture
  - 2-stage **robust** router to generate **logic element level** routing result
  - **ILP-based concurrent tile assignment** to deal with logic tiles difficult to route
  - **Stencil-based parallelization** to accelerate tile assignment
- Result in **less runtime and wirelength** than SOTA
  - **100%** routability
  - **8.87x** faster
  - **16.25%** less wirelength

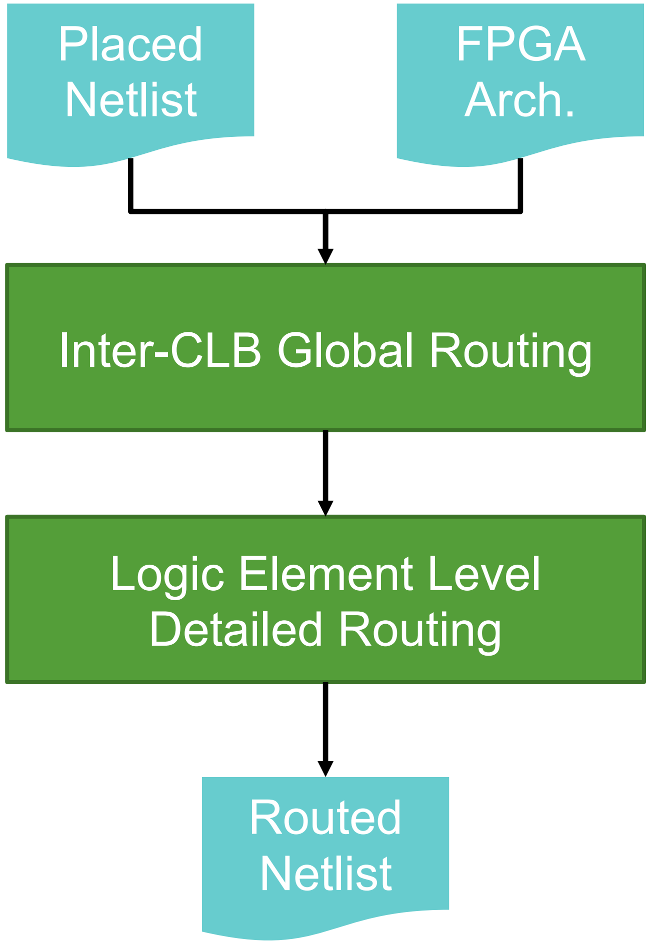
Algorithm Framework

- 2-Stage router to generate routing result
  - Global routing to assign inter-CLB topology



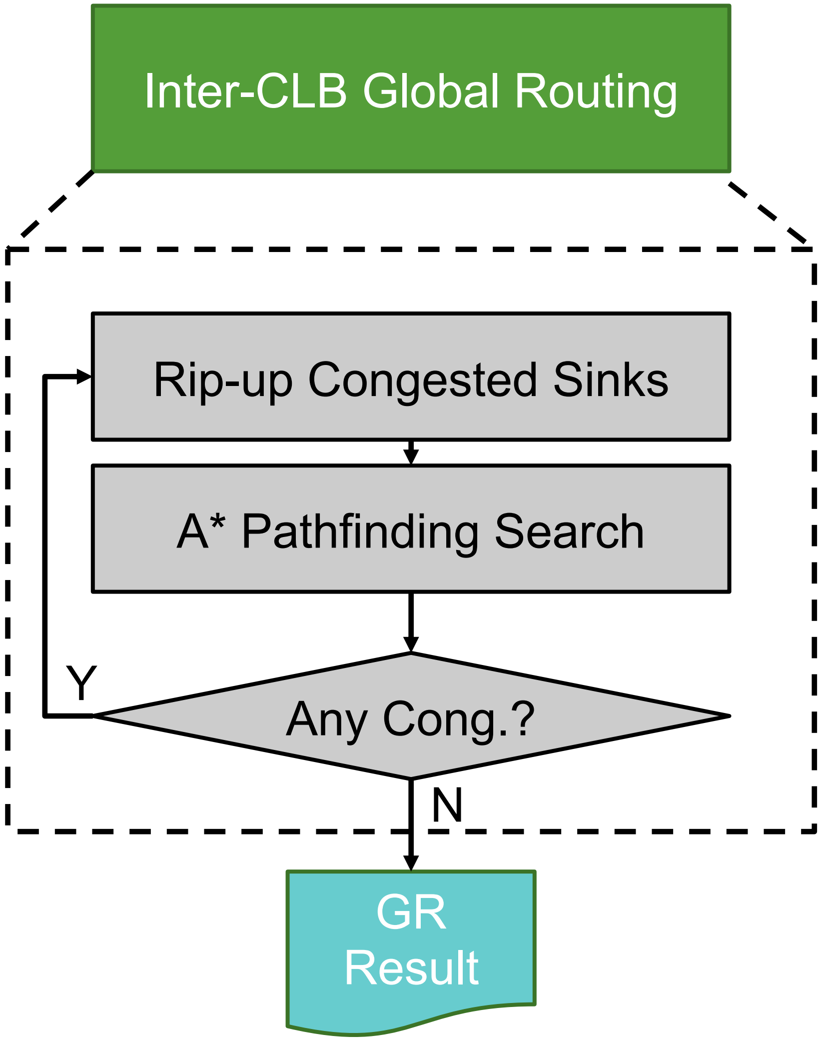
Algorithm Framework

- 2-Stage router to generate routing result
  - Global routing to assign inter-CLB topology
  - Detailed routing to generate routing result



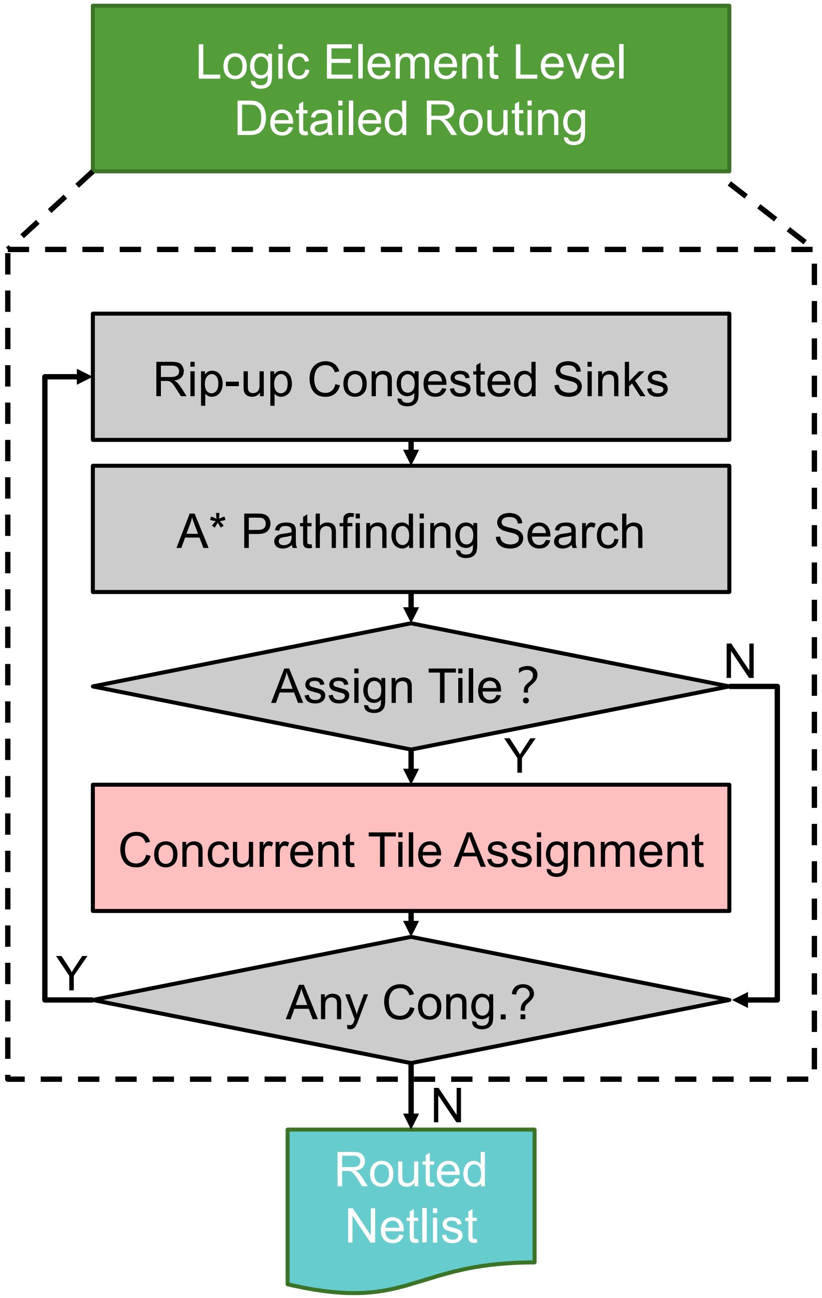
Algorithm Framework

- 2-Stage router to generate routing result
  - Global routing to assign inter-CLB topology
  - Detailed routing to generate routing result



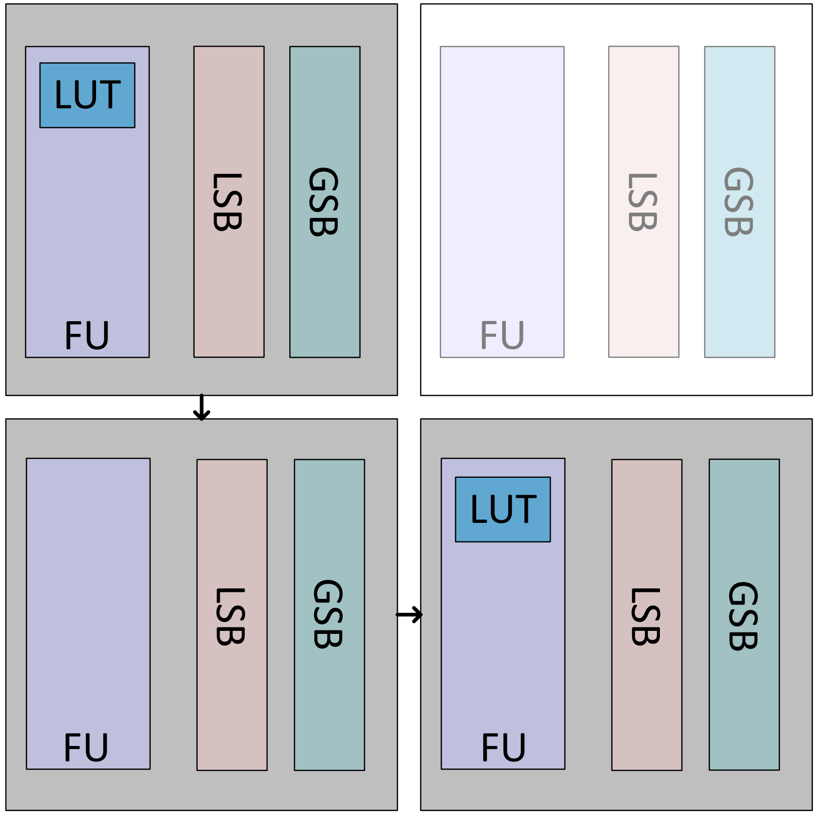
Algorithm Framework

- 2-Stage router to generate routing result
  - Global routing to assign inter-CLB topology
  - Detailed routing to generate routing result
- Concurrent tile assignment
  - Resolve congestions inside CLBs difficult to route



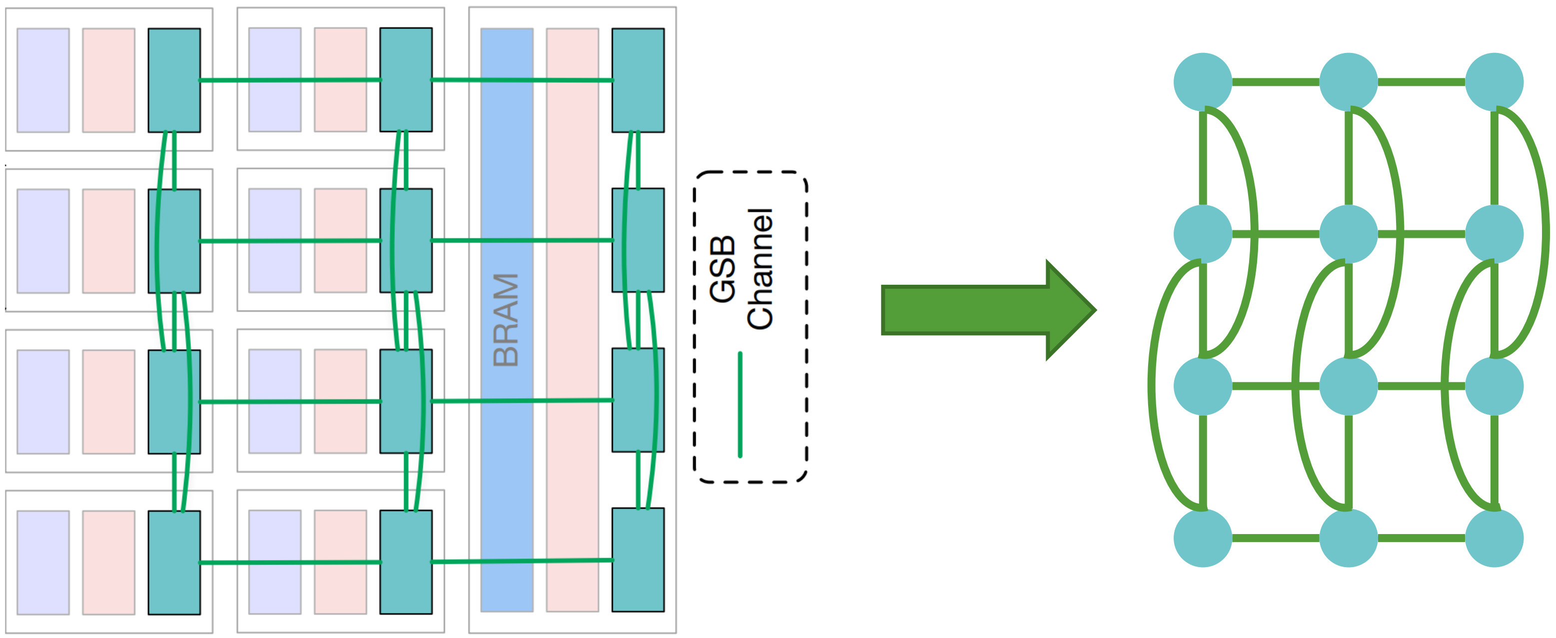
Global Routing

- Target: Generate inter-CLB level coarsen routing result
  - Main idea: Pathfinder [L. McMurchie et. al. FPGA '95]



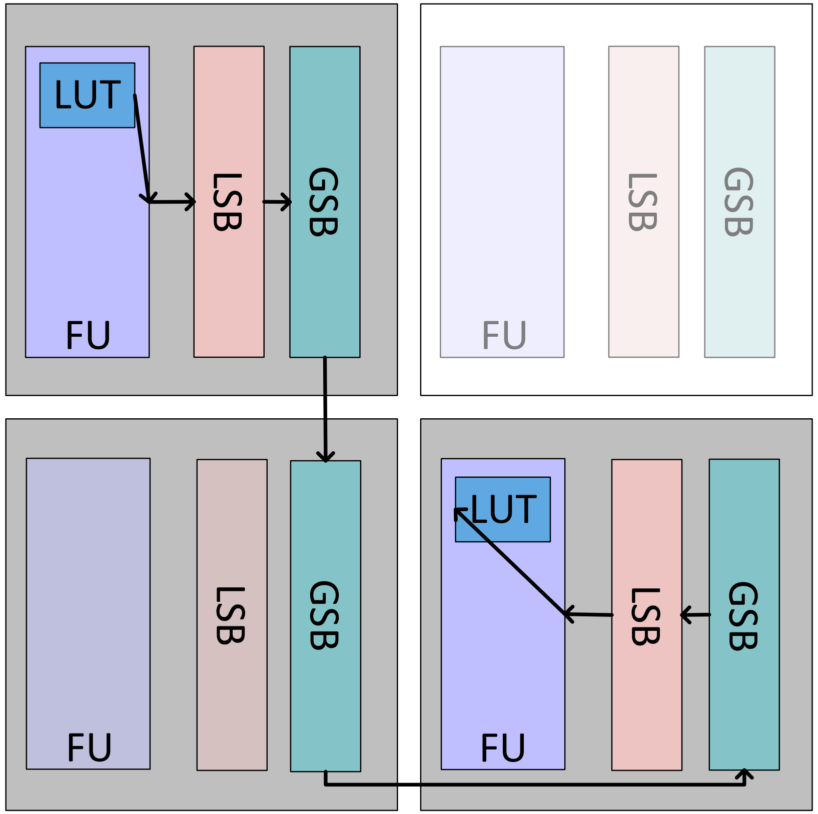
Global Routing

- Target: Generate inter-CLB level coarsen routing result
  - Main idea: Pathfinder [L. McMurchie et. al. FPGA'95]
- Regard logic blocks as a grid graph



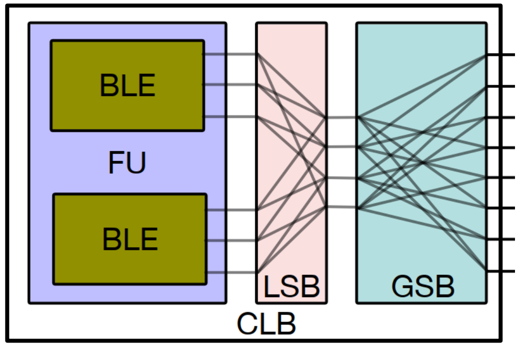
Detailed Routing

- Decide logic element level routing path for each net following guide of global routing

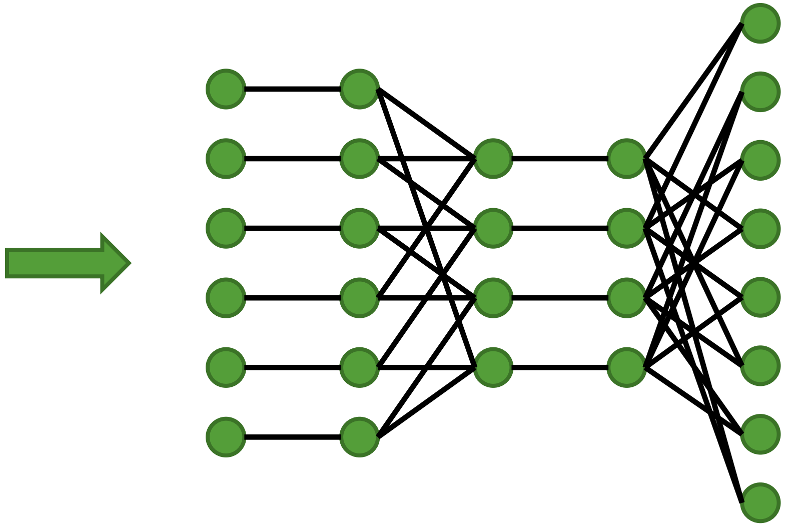


Detailed Routing

- Decide logic element level routing path for each net following guide of global routing
- Regard each logic pin as a vertex in the RRG



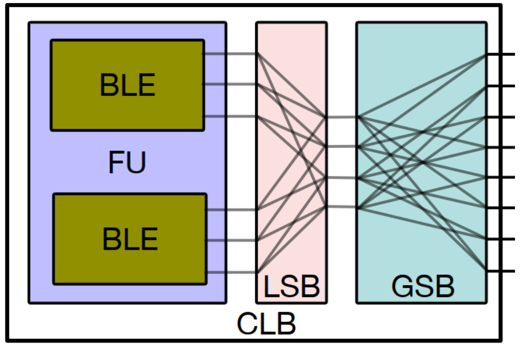
CLB Layout



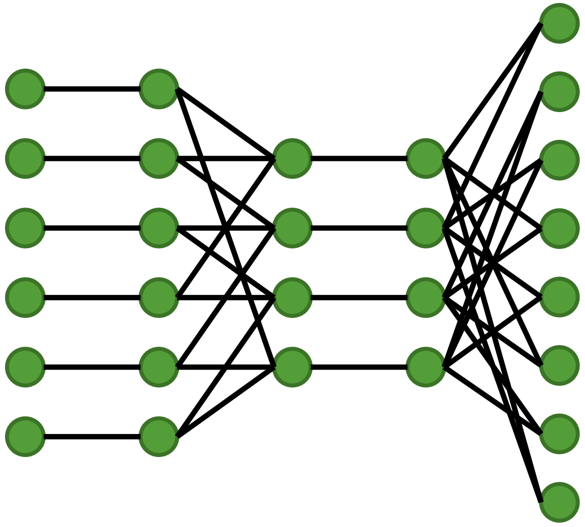
Routing Resource Graph

Detailed Routing

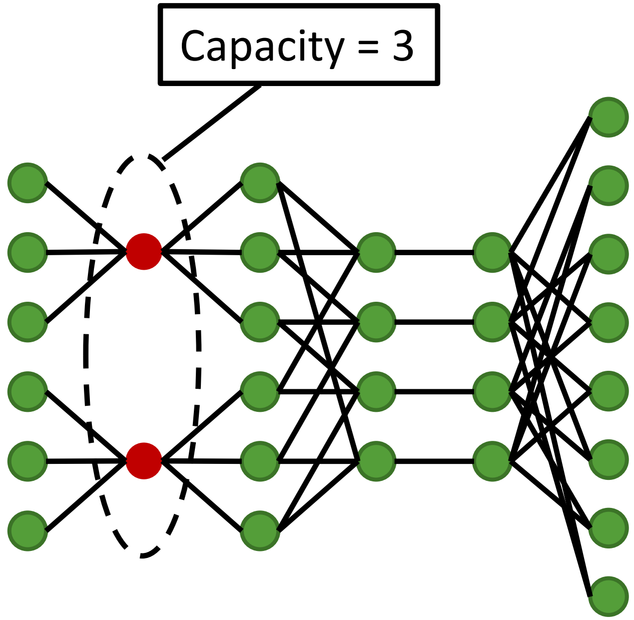
- Decide logic element level routing path for each net following guide of global routing
- Regard each logic pin as a vertex in the RRG
- Pin merging and swapping to improve routability



CLB Layout



Routing Resource Graph Before Pin Merging



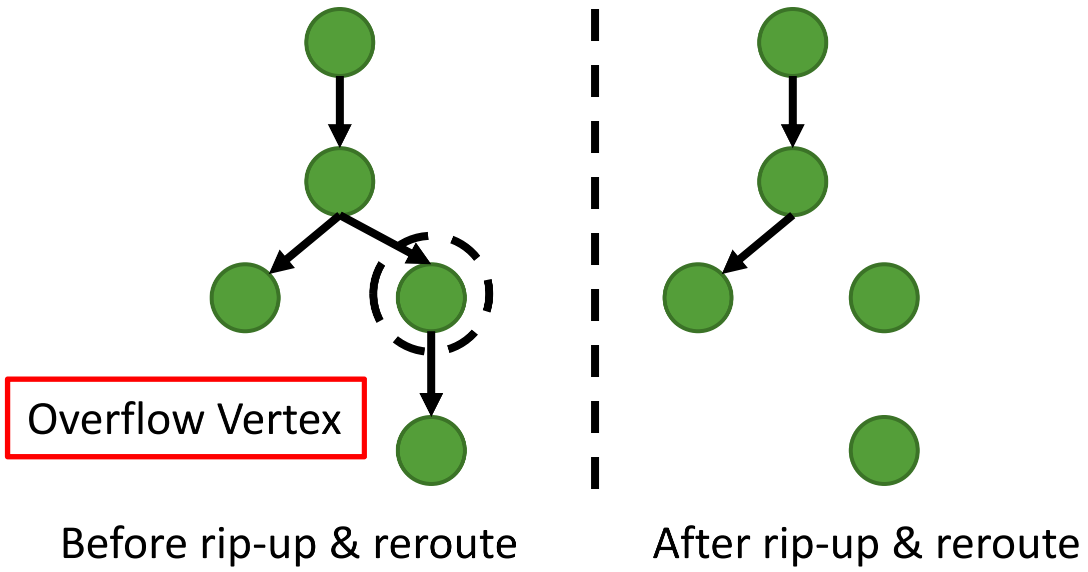
Routing Resource Graph After Pin Merging

Detailed Routing – Routing Enhancement

- Decide logic element level routing path for each net following guide of global routing
- Regard each logic pin as a vertex in the RRG
- Pin merging and swapping to improve routability
- Other enhancement technique to reduce runtime and improve quality

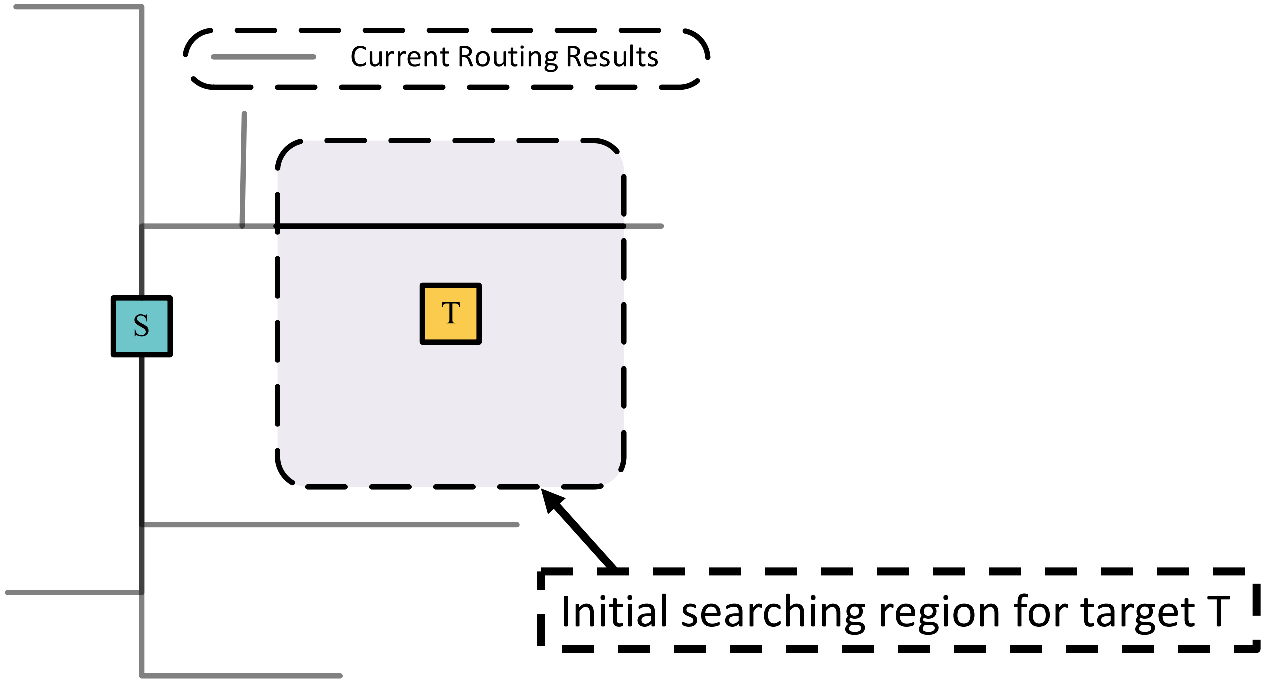
Detailed Routing – Routing Enhancement

- Decide logic element level routing path for each net following guide of global routing
- Regard each logic pin as a vertex in the RRG
- Pin merging and swapping to improve routability
- Other enhancement technique to reduce runtime and improve quality
  - Rip-up & reroute enhancement



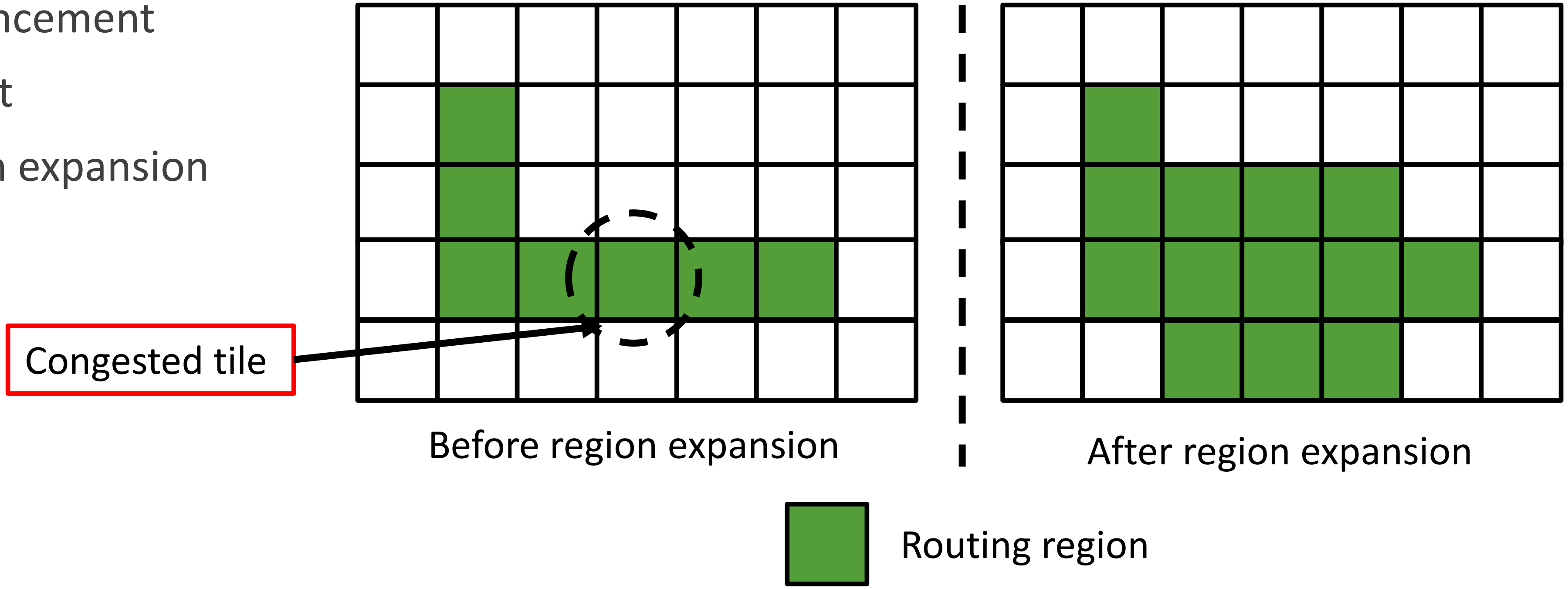
Detailed Routing – Routing Enhancement

- Decide logic element level routing path for each net following guide of global routing
- Regard each logic pin as a vertex in the RRG
- Pin merging and swapping to improve routability
- Other enhancement technique to reduce runtime and improve quality
  - Rip-up & reroute enhancement
  - Large net enhancement



Detailed Routing – Routing Enhancement

- Decide logic element level routing path for each net following guide of global routing
- Regard each logic pin as a vertex in the RRG
- Pin merging and swapping to improve routability
- Other enhancement technique to reduce runtime and improve quality
  - Rip-up & reroute enhancement
  - Large net enhancement
  - Dynamic routing region expansion



Detailed Routing – Routing Enhancement

- Decide logic element level routing path for each net following guide of global routing
- Regard each logic pin as a vertex in the RRG
- Pin merging and swapping to improve routability
- Other enhancement technique to reduce runtime and improve quality
  - Rip-up & reroute enhancement
  - Large net enhancement
  - Dynamic routing region expansion
  - Historical-based cost function calculation

$$c(u,v) = (1 + p * overuse(v)) * (b(v) + h(v)) * weight(u,v)$$

Cost of edge from u to v

Basic cost

Historical cost

Edge weight

Concurrent Tile Assignment

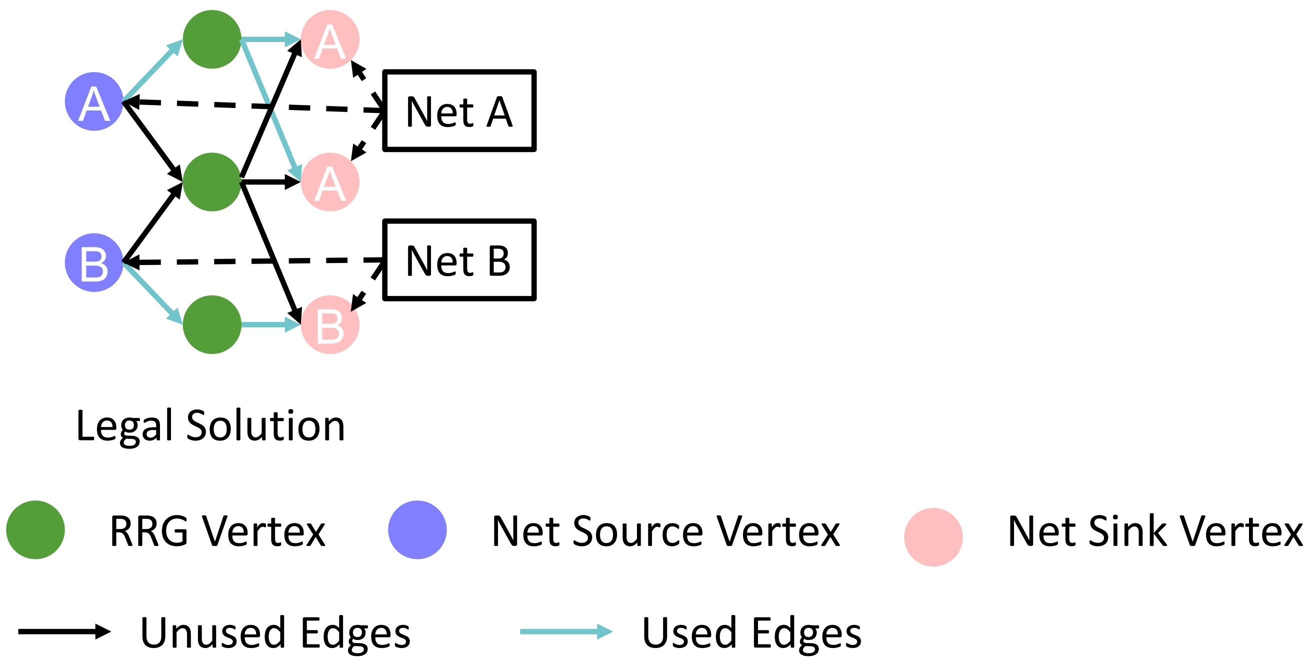
- Most congestion can be resolved in first few iterations
  - Congestion remains in few logic tiles
- Use ILP to concurrently generate routing result for those tiles
  - Consider a tile and its neighbor tile to improve quality

Target of ILP

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

Target of ILP

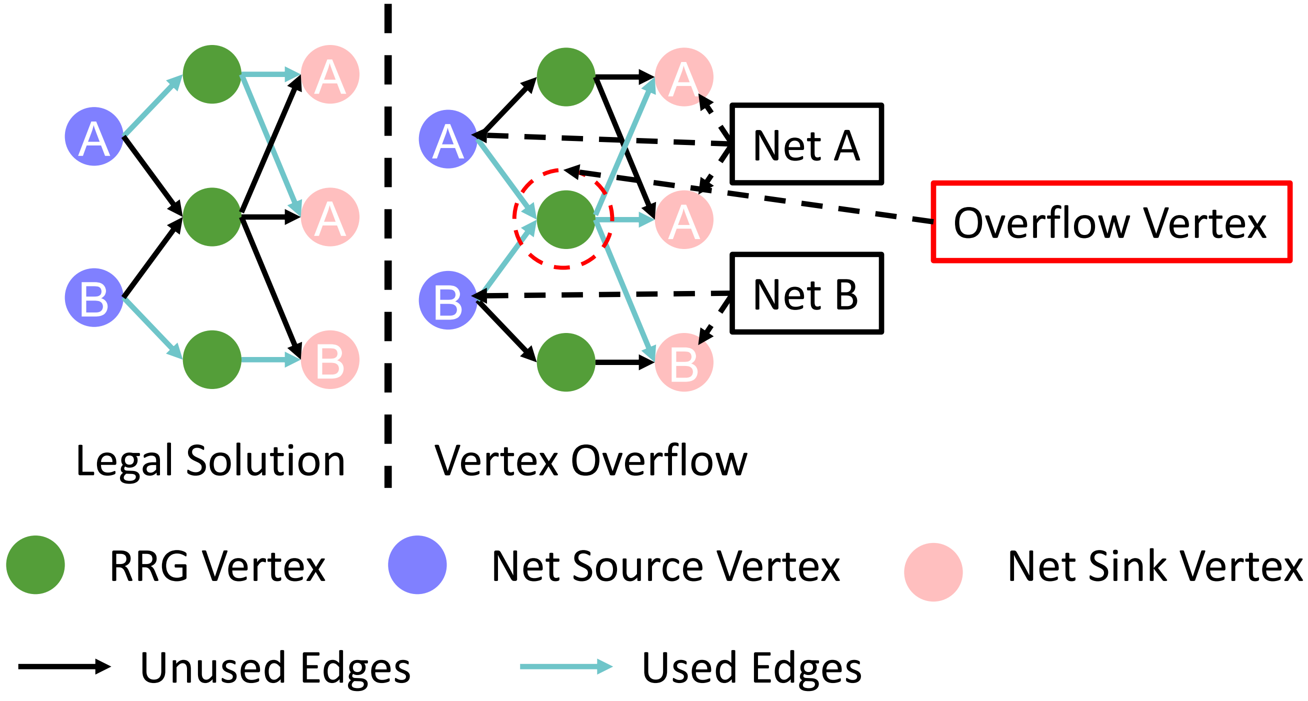
- Route multiple nets inside a tile and its neighbor tile concurrently
  - No overflow vertices
  - No loop in the paths



Legal Solution

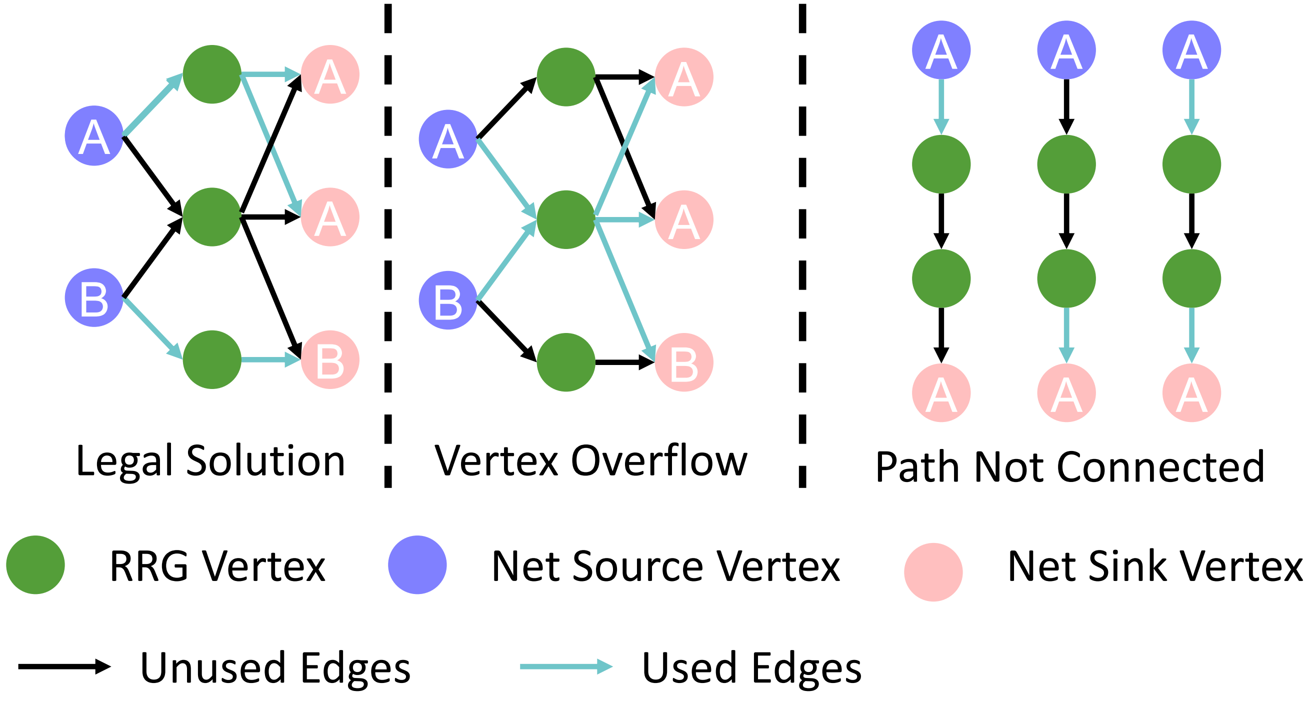
Target of ILP

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



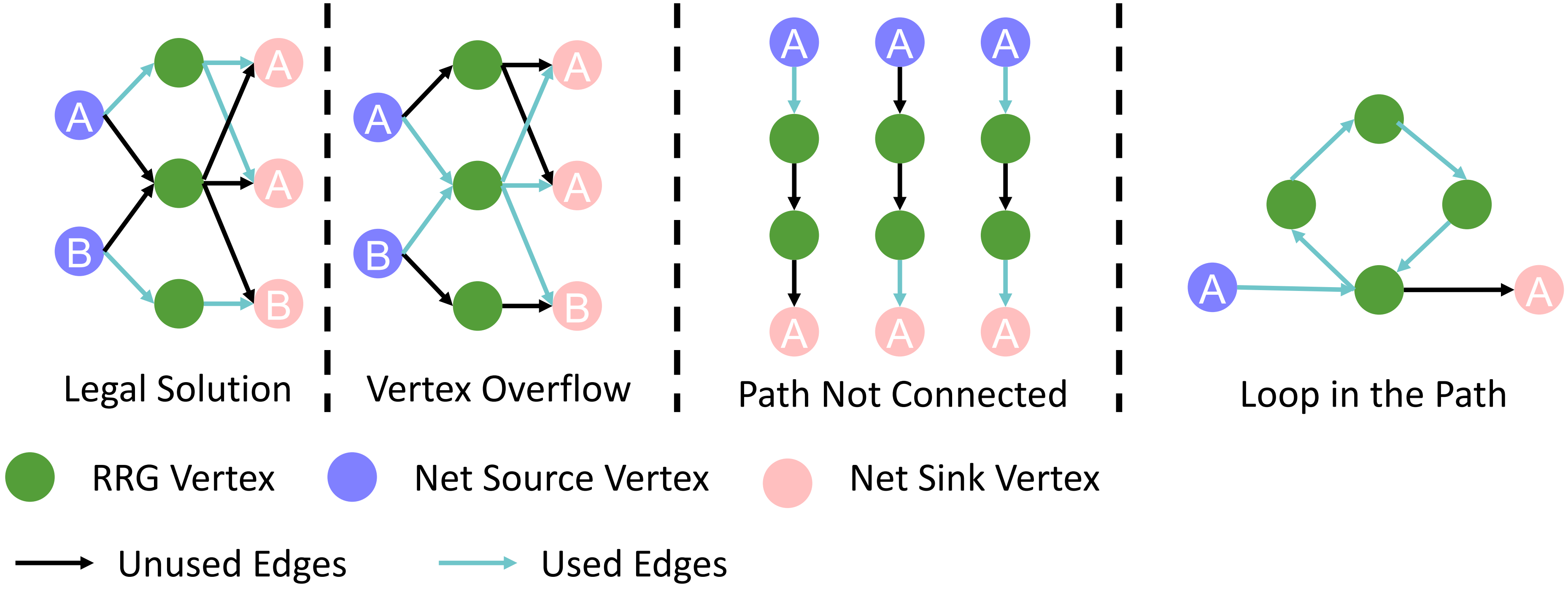
Target of ILP

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



Target of ILP

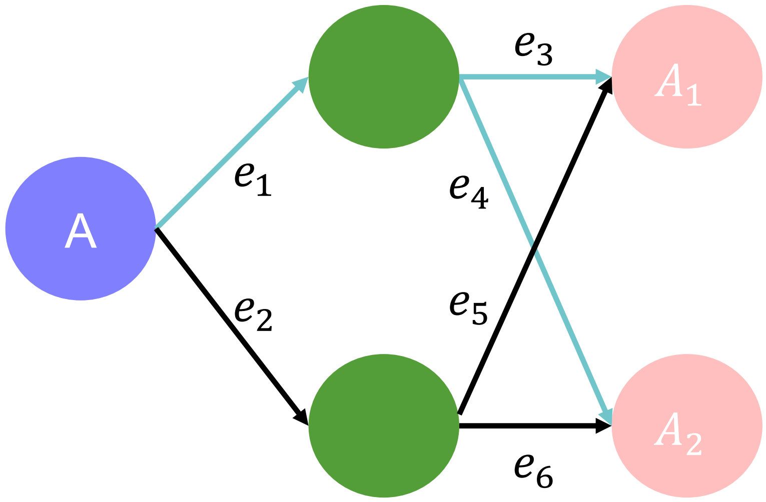
- Route multiple nets inside a tile and its neighbor tile concurrently
  - No overflow vertices
  - No loop in the paths



ILP Variables & Objective

- Variables of ILP:

$R_{e,j}$	whether edge e is used to route net j
$S_{e,j,k}$	whether edge e is used to route sink k of net j



$$R_{e_1,A} = R_{e_3,A} = R_{e_4,A} = 1$$
$$S_{e_1,A,A_1} = S_{e_3,A,A_1} = 1$$
$$S_{e_1,A,A_2} = S_{e_4,A,A_2} = 1$$
Other binary variables are 0

ILP Variables & Objective

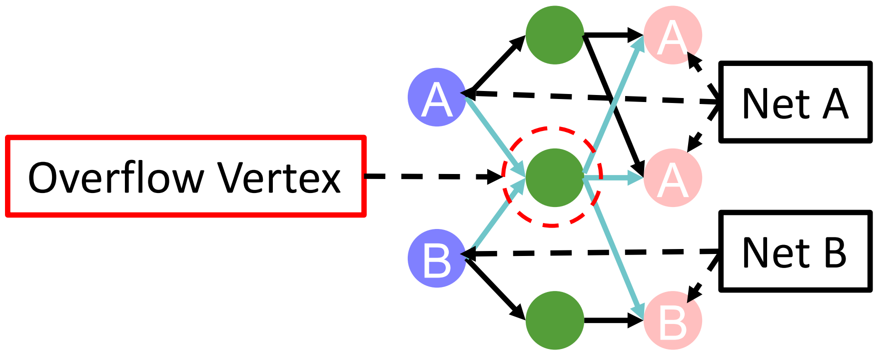
- Variables of ILP:

$R_{e,j}$	whether edge e is used to route net j
$S_{e,j,k}$	whether edge e is used to route sink k of net j

- ILP Objective
  - Minimize $\sum_{e,j} R_{e,j} \cdot COST(e)$ ← Cost of RRG edge e

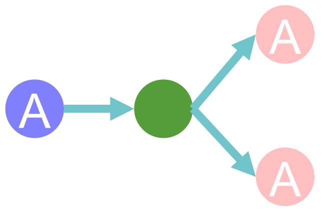
ILP Formulation of Tile Assignment

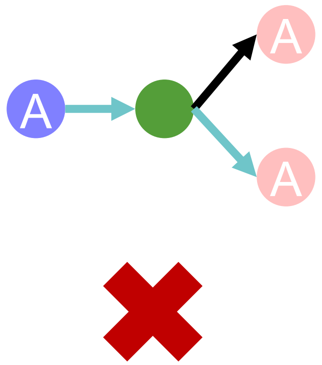
- ILP constraints
  - $\sum_{e,j} R_{e,j} \leq cap(v), e \in FI(v)$ Ensure **no overflow** vertex



ILP Formulation of Tile Assignment

- ILP constraints
  - $\sum_{e,j} R_{e,j} \leq cap(v), e \in FI(v)$
  - $S_{e,j,k} \leq R_{e,j}, k \in SINK(j)$ Ensure **each sink of each net is routed**

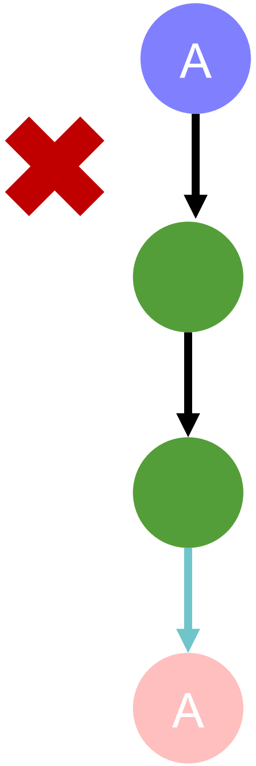




ILP Formulation of Tile Assignment

- ILP constraints
  - $\sum_{e,j} R_{e,j} \leq cap(v), e \in FI(v)$
  - $S_{e,j,k} \leq R_{e,j}, k \in SINK(j)$
  - $\sum_{e,j,k} S_{e,j,k} = 1, e \in FO(v), v = SOURCE(j), \forall k \in SINK(j)$

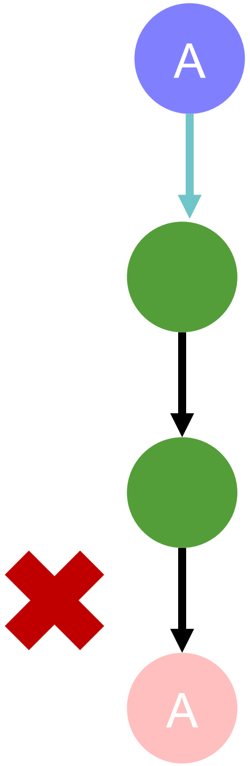
  Ensure signal is **sent from source pin** of each net



ILP Formulation of Tile Assignment

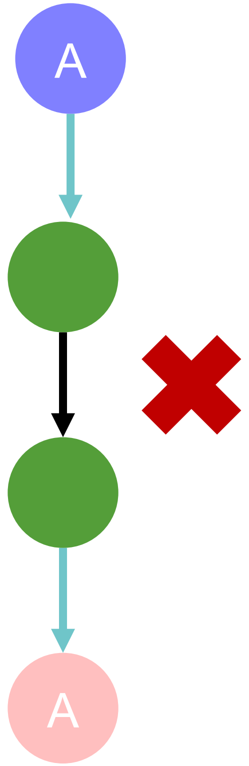
- ILP constraints
  - $\sum_{e,j} R_{e,j} \leq cap(v), e \in FI(v)$
  - $S_{e,j,k} \leq R_{e,j}, k \in SINK(j)$
  - $\sum_{e,j,k} S_{e,j,k} = 1, e \in FO(v), v = SOURCE(j), \forall k \in SINK(j)$
  - $\sum_{e,j,k} S_{e,j,k} = 1, e \in FI(v), v = SINK(j,k)$

  Ensure signal is **received at each sink pin** of each net

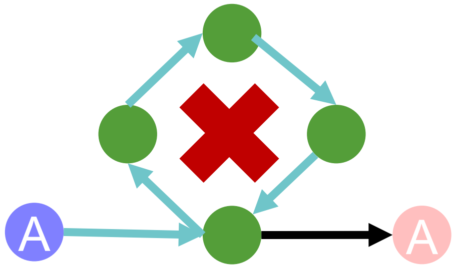


ILP Formulation of Tile Assignment

- ILP constraints
  - $\sum_{e,j} R_{e,j} \leq cap(v), e \in FI(v)$
  - $S_{e,j,k} \leq R_{e,j}, k \in SINK(j)$
  - $\sum_{e,j,k} S_{e,j,k} = 1, e \in FO(v), v = SOURCE(j), \forall k \in SINK(j)$
  - $\sum_{e,j,k} S_{e,j,k} = 1, e \in FI(v), v = SINK(j,k)$
  - $\sum_{e_{in}} S_{e_{in},j,k} = \sum_{e_{out}} S_{e_{out},j,k}, e_{in} \in FI(v), e_{out} \in FO(v), v \neq SOURCE(j), v \notin SINK(j)$

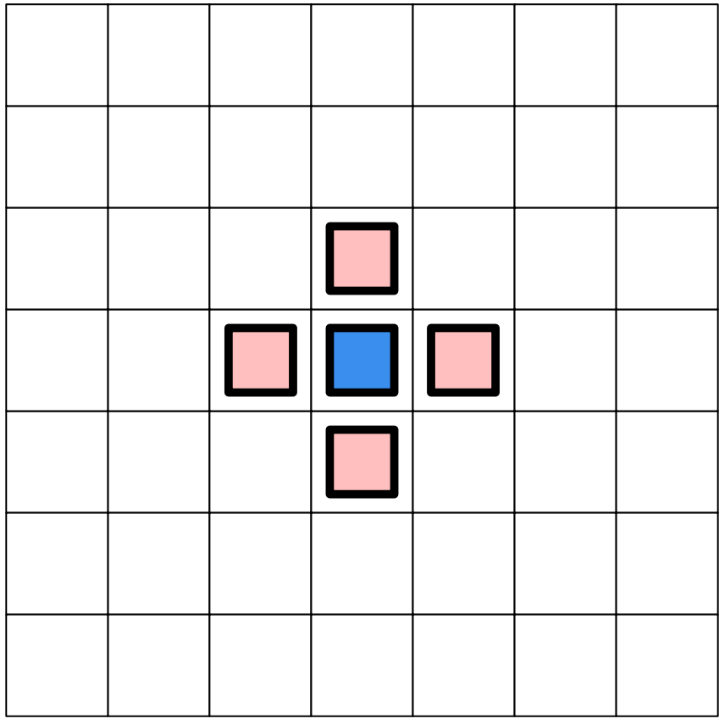


Ensure there is **a path from source pin to each sink pin** and ensure **no loop** in the routing result



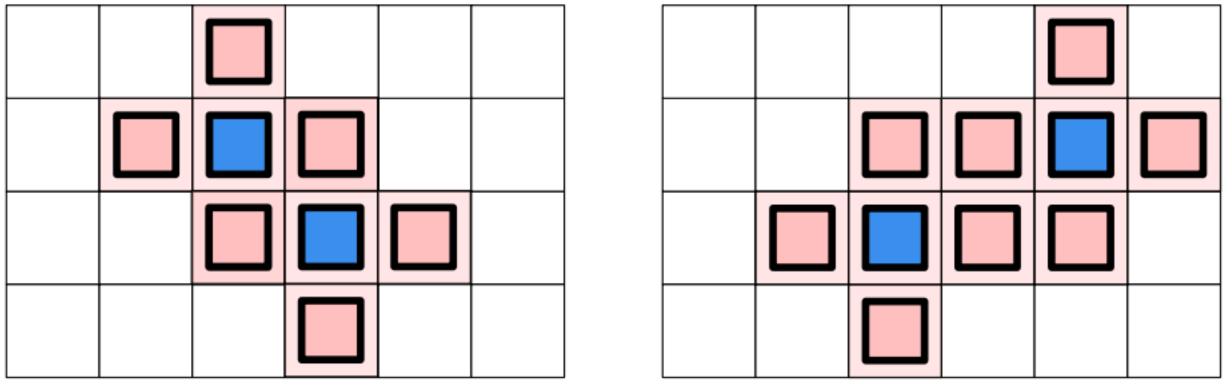
Stencil-Based Parallelization

- Solving ILP during tile assignment takes large amount of time
  - Trying to solve ILP parallelly



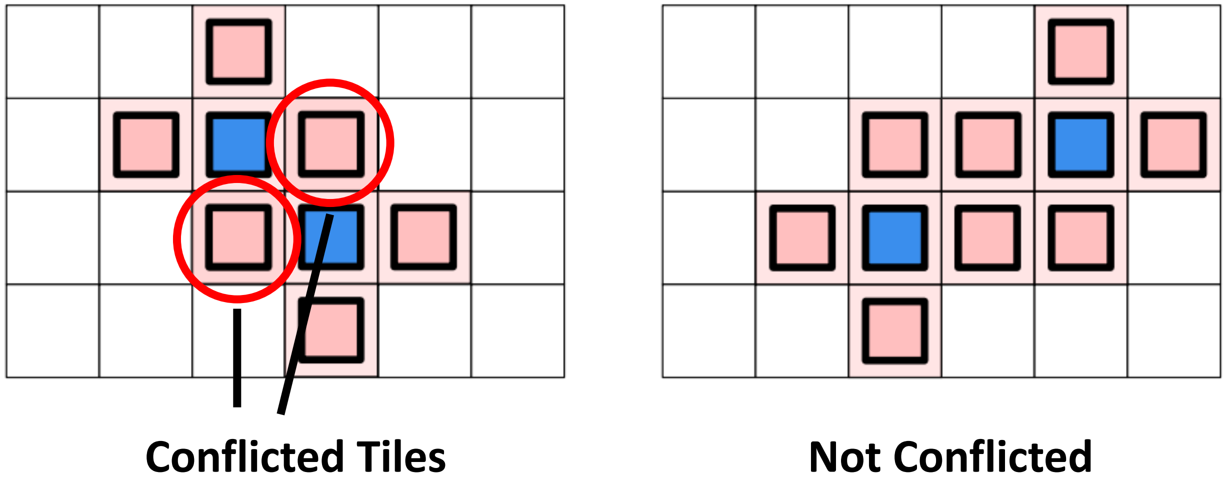
Stencil-Based Parallelization

- Solving ILP during tile assignment takes large amount of time
  - Trying to solve ILP parallelly
- Consider data dependency between different logic tiles



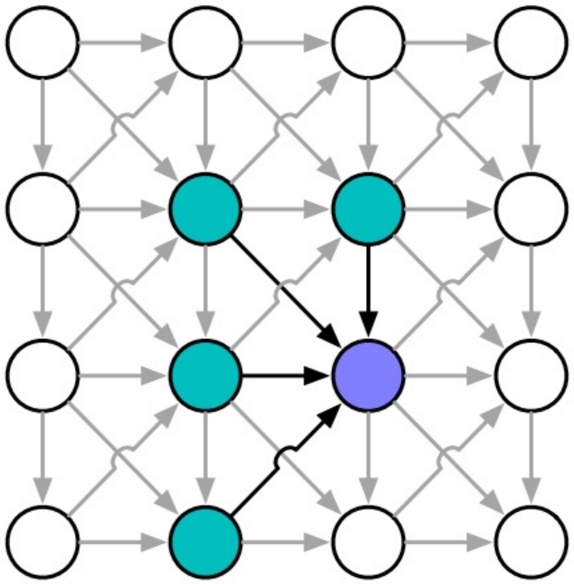
Stencil-Based Parallelization

- Solving ILP during tile assignment takes large amount of time
  - Trying to solve ILP parallelly
- Consider data dependency between different logic tiles



Stencil-Based Parallelization

- Solving ILP during tile assignment takes large amount of time
  - Trying to solve ILP parallelly
- Consider data dependency between different logic tiles
- Stencil-based data dependency graph



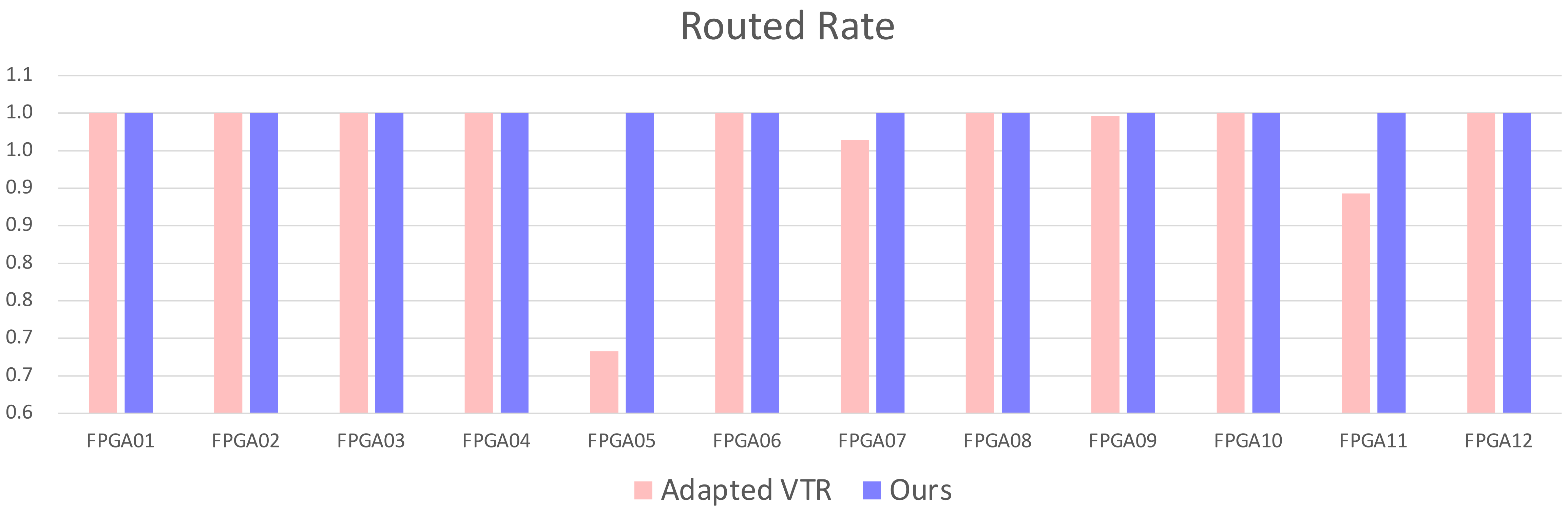
Experimental Setup

- FPGA design: ISPD ’16 contest benchmark excluding control set signals
- Industrial FPGA routing architecture
  - Anonymous due to confidential issues
- Place result from ISPD ’16 contest winner
- Adapted VTR router [Murray et. al. ASPDAC ’20] as baseline

Design	#Cells (K)	#Nets(K)	Design	#Cells (K)	#Nets (K)
FPGA01	105	105	FPGA07	707	716
FPGA02	166	167	FPGA08	717	725
FPGA03	421	428	FPGA09	867	876
FPGA04	423	420	FPGA10	952	961
FPGA05	425	433	FPGA11	845	851
FPGA06	704	713	FPGA12	1103	1111

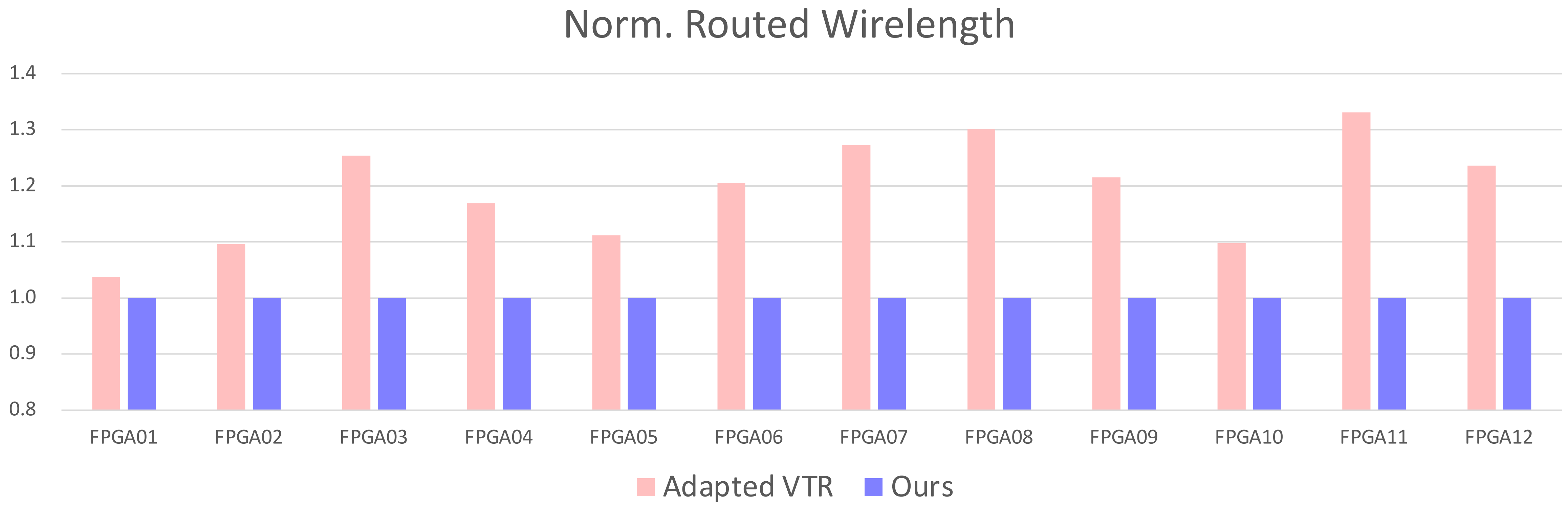
Experimental Results

- Our router successfully routes **all the designs**
- Adapted VTR fails in **4 of 12** designs



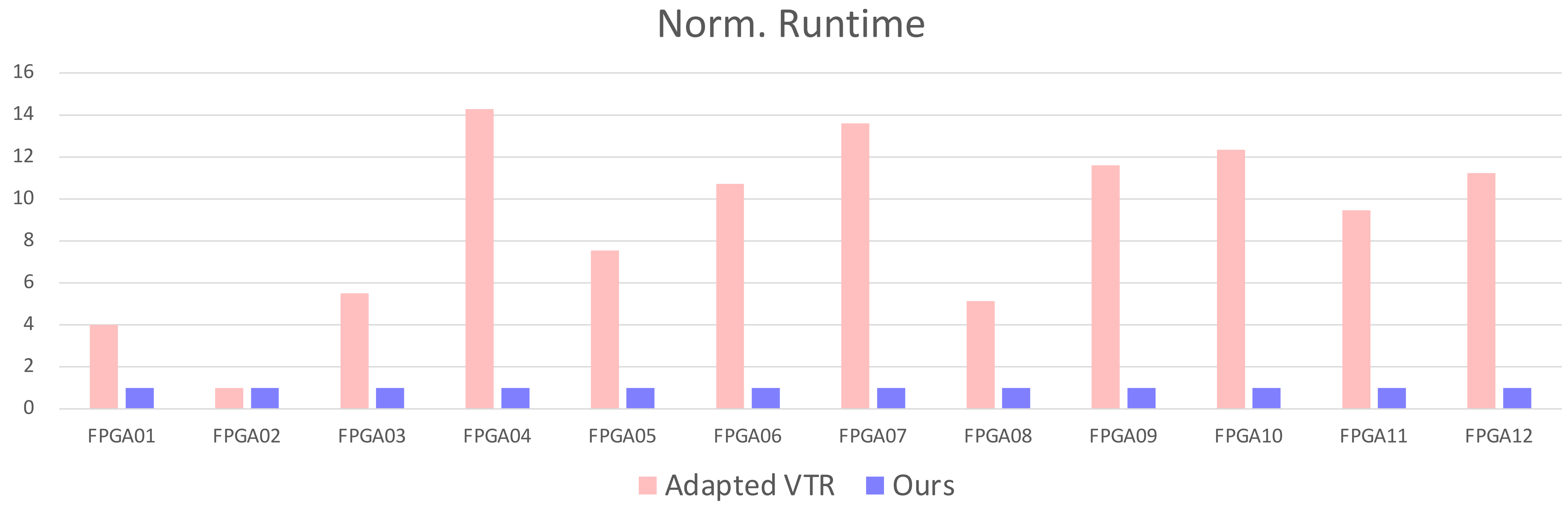
Experimental Results

- Wirelength of our router is **16.25% less** than adapted VTR on average



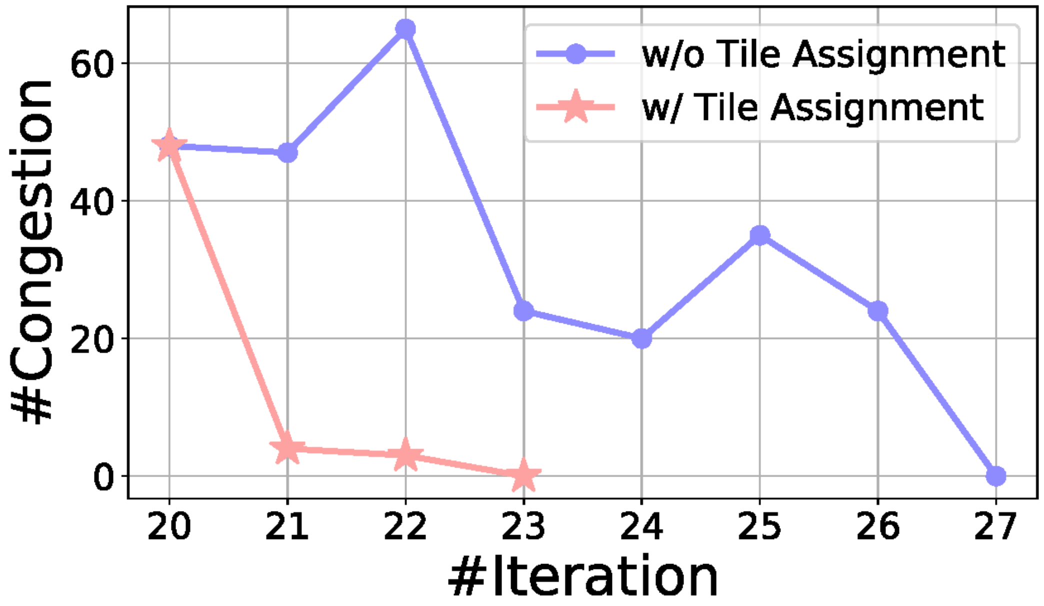
Experimental Results

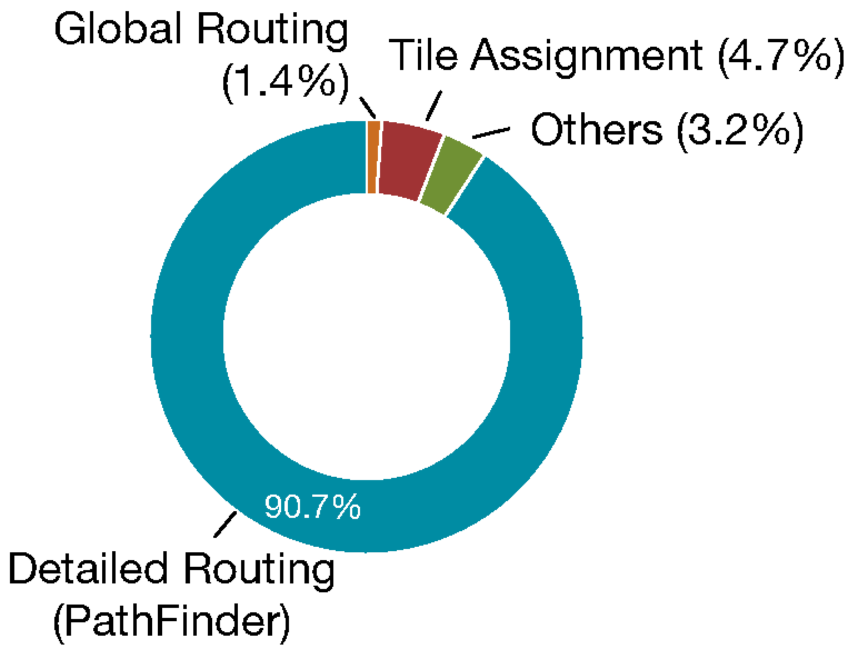
- Speed of our router is **8.87x faster** than adapted VTR on average



Experimental Results

- By applying tile assignment at the 20th rip-up & reroute iteration, our router gain 4 iterations less on FPGA08.





Runtime breakdown on FPGA08

Conclusion

- **Robust** FPGA router for FPGA architecture with **non logic-equivalence** logic pins
  - 2-stage global & detailed routing
  - **Effective concurrent tile assignment** with **stencil based parallelization**
- **8.87x faster** and **16.25% less** wirelength with **100% routability** compare to SOTA

Future work:
- Parallelization during detailed routing
- Support timing-driven routing

Thanks!

Questions are welcome