AIR: A Fast but Lazy Timing-Driven FPGA Router

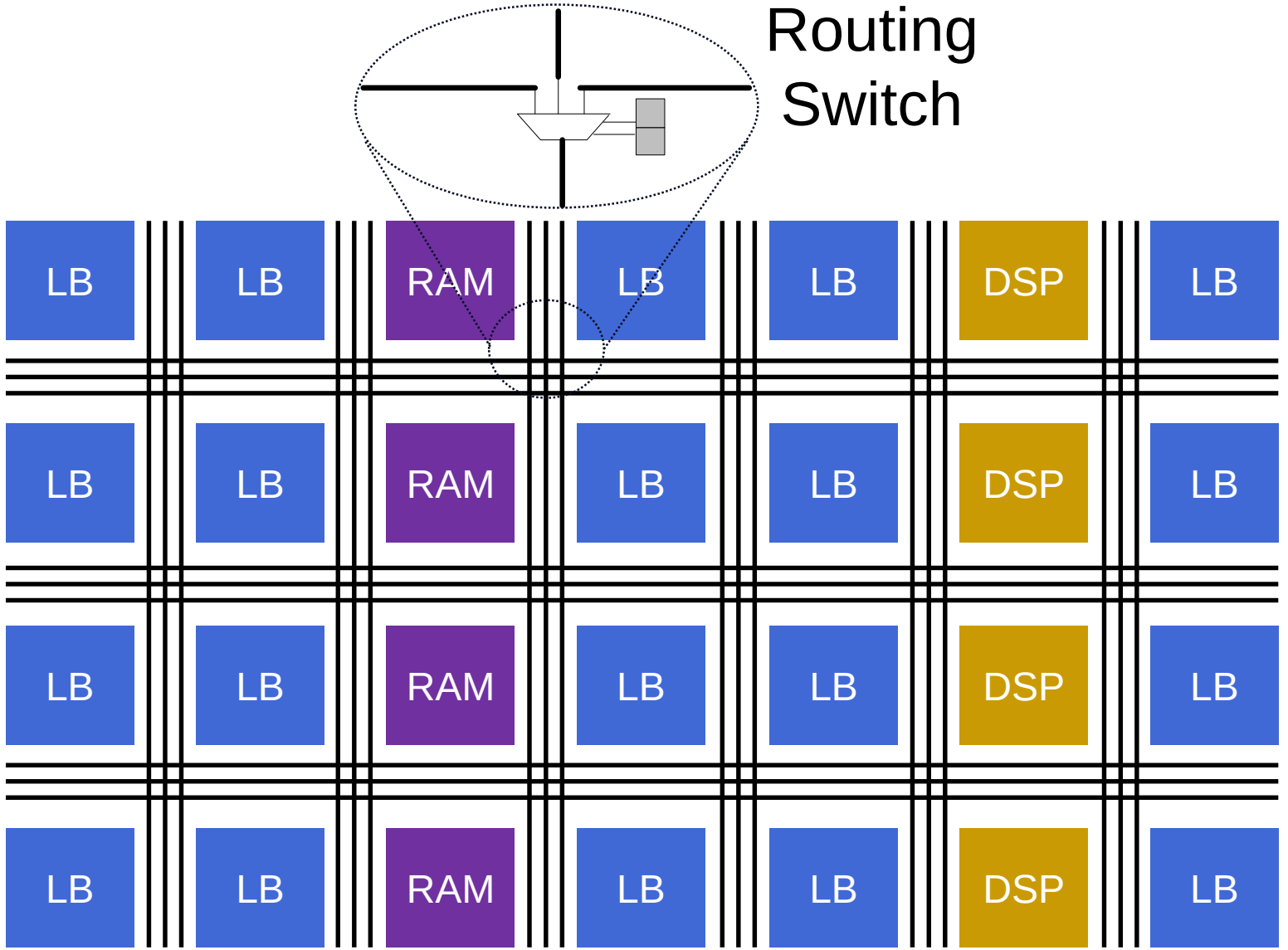
Kevin E. Murray, Sheng Zhong, Vaughn Betz

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

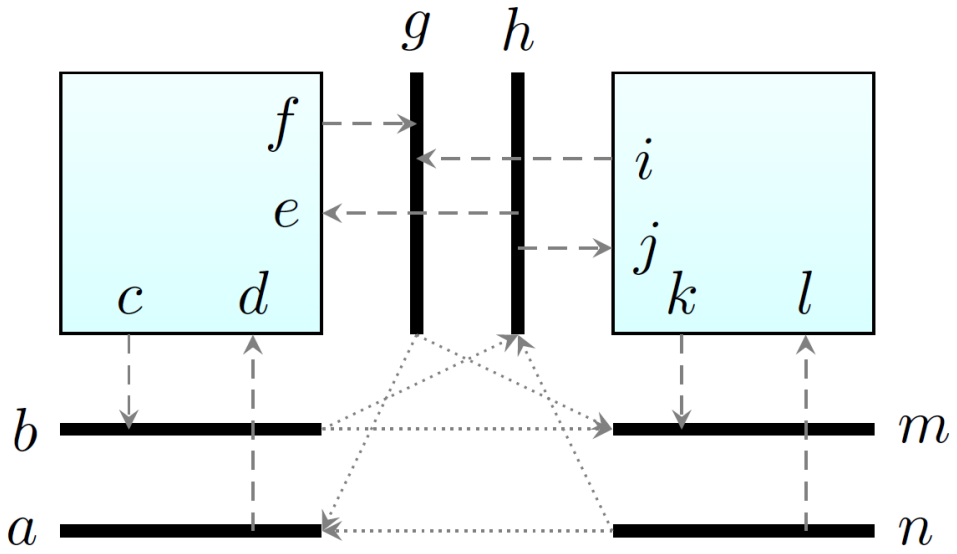
FPGA Architecture & CAD

Field Programmable Gate Arrays (FPGAs)

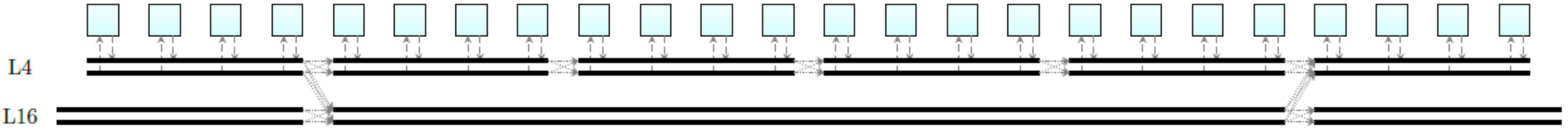
- Pre-fabricated Logic:
  - Logic Blocks (LB)
  - RAM Blocks
  - DSP Blocks
- Pre-fabricated Routing:
  - Wires
  - Muxes
- Routing & Logic *configurable*
  - Enables implementation of arbitrary digital circuits



FPGA Interconnect Network

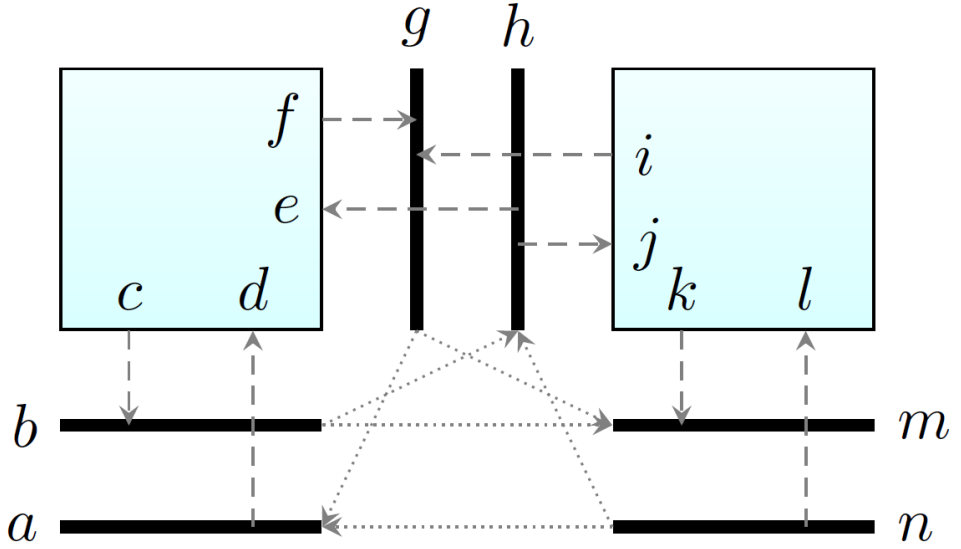


- Significant
  - >50% silicon area
  - Dominates timing
- Routing challenges:
  - Fixed/pre-fabricated wires
  - Sparse connectivity (leads to congestion)
  - Different resource types (electrical characteristics, wire lengths, connectivity)
  - Large networks (100Ms of conductors)
  - No later fix-ups (no buffer insertion etc.)



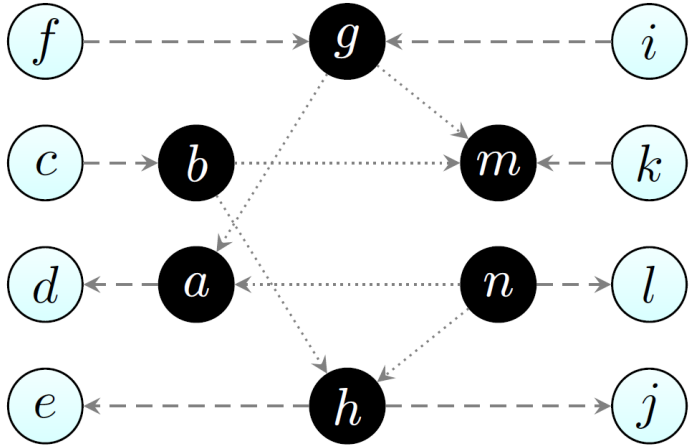
FPGA Routing Problem

- Model interconnect network as Routing Resource (RR) graph:
  - Nodes: Conductors (wires/pins)
  - Edges: Configurable switches



Routing Architecture

- FPGA Routing Problem:
  - Find embedding of netlist in RR graph
  - Requires finding many non-overlapping trees in RR graph
  - Minimize timing and wiring/power
  - Reasonable run-time



Routing Resource (RR) Graph

Negotiated Congestion routing

- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other

Connection Router

Negotiated Congestion routing

- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other

Connection Router

Negotiated Congestion routing

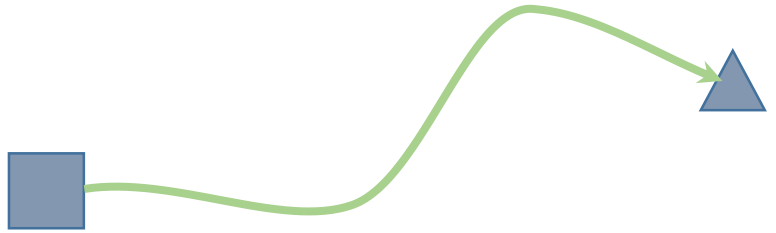
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other

Connection Router

Negotiated Congestion routing

- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other

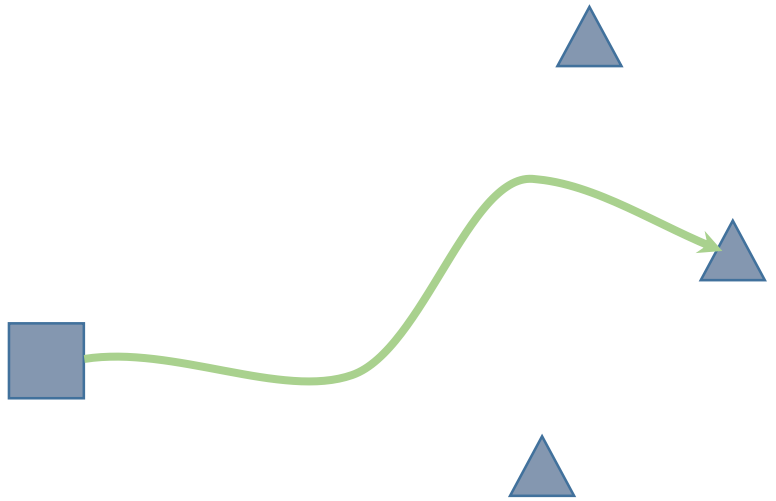
Connection Router



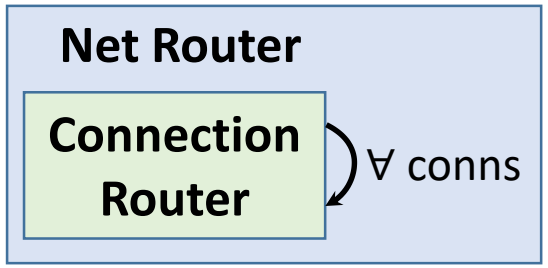
Negotiated Congestion routing

- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other

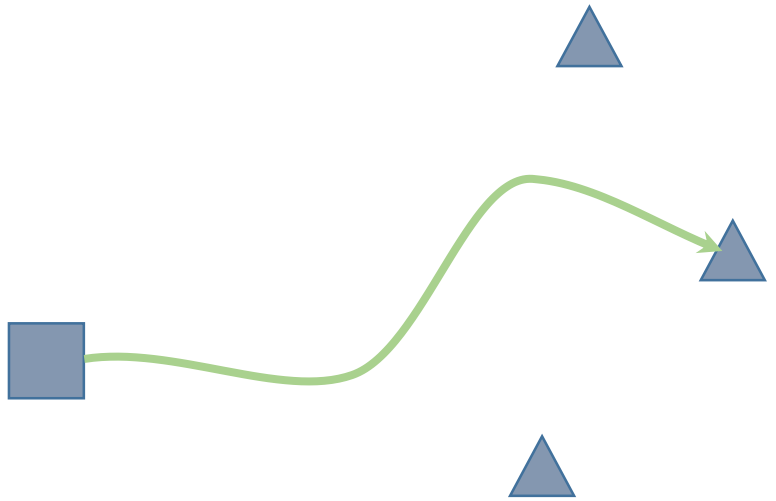
Connection Router



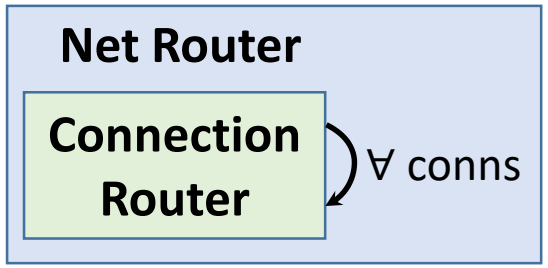
Negotiated Congestion routing



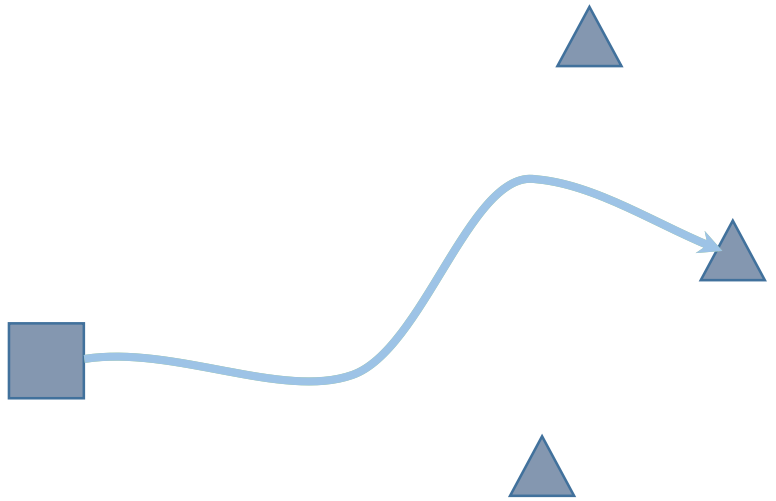
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



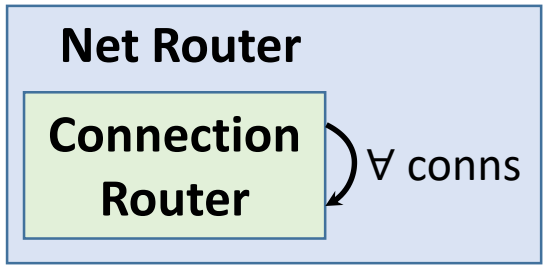
Negotiated Congestion routing



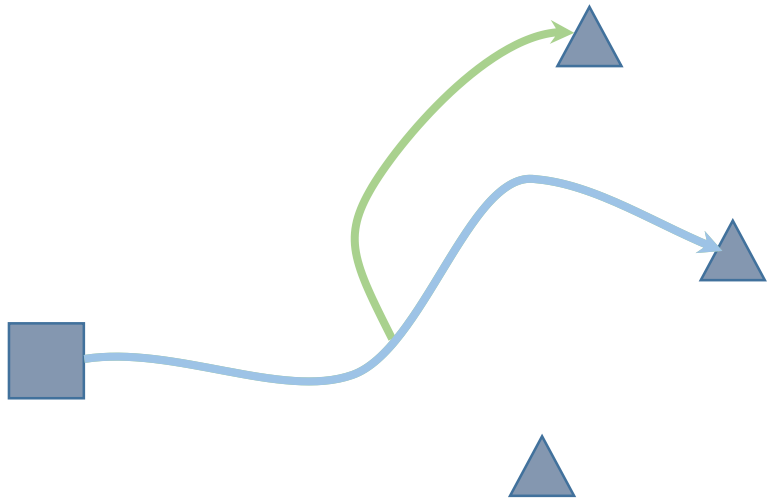
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



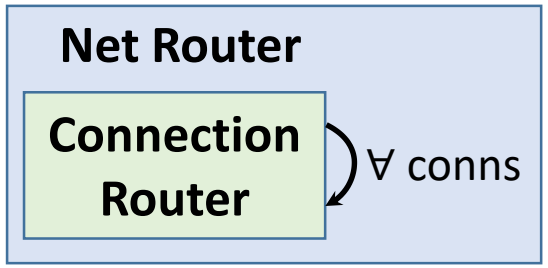
Negotiated Congestion routing



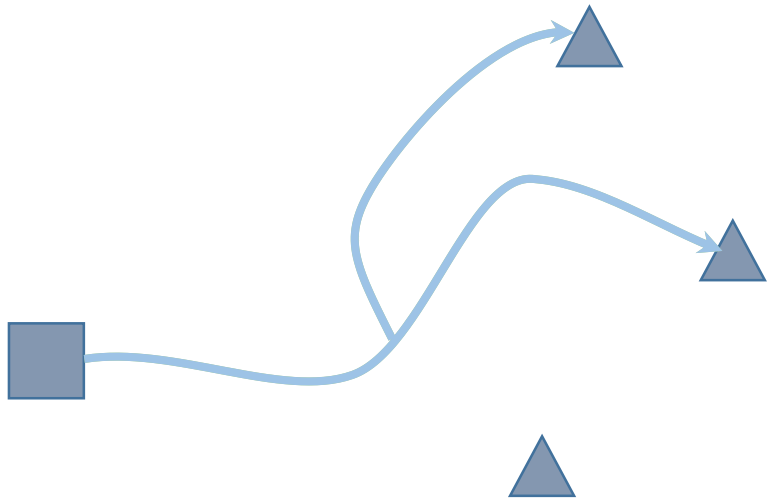
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



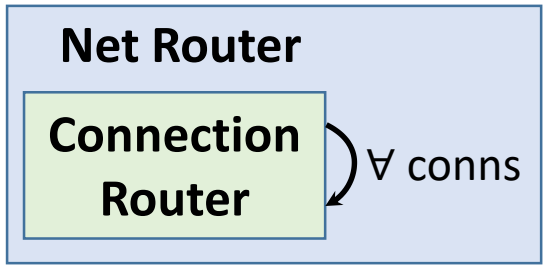
Negotiated Congestion routing



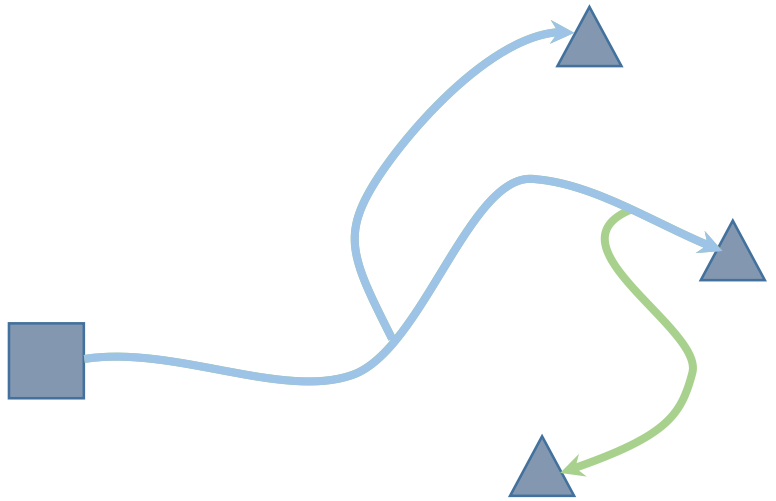
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



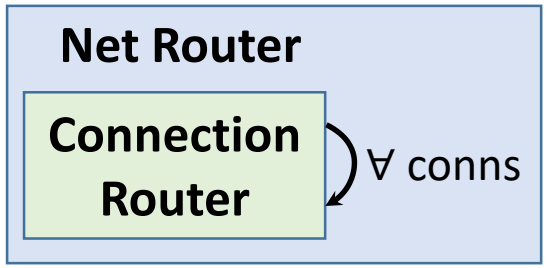
Negotiated Congestion routing



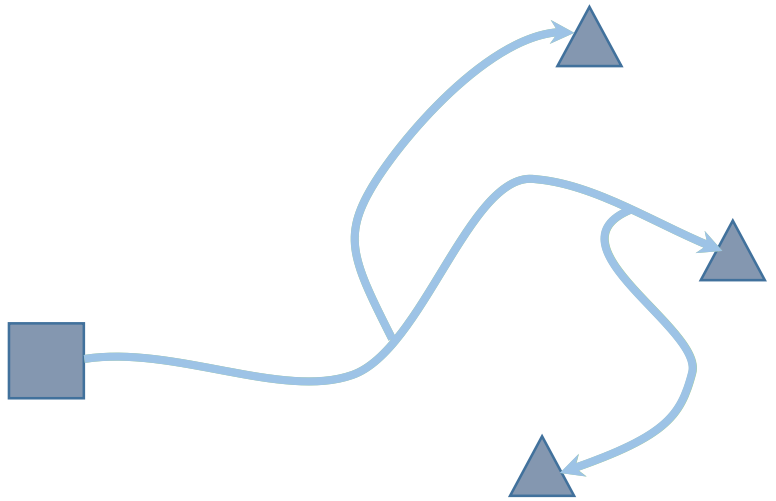
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



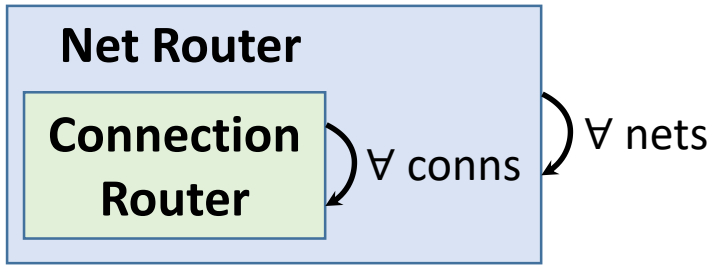
Negotiated Congestion routing



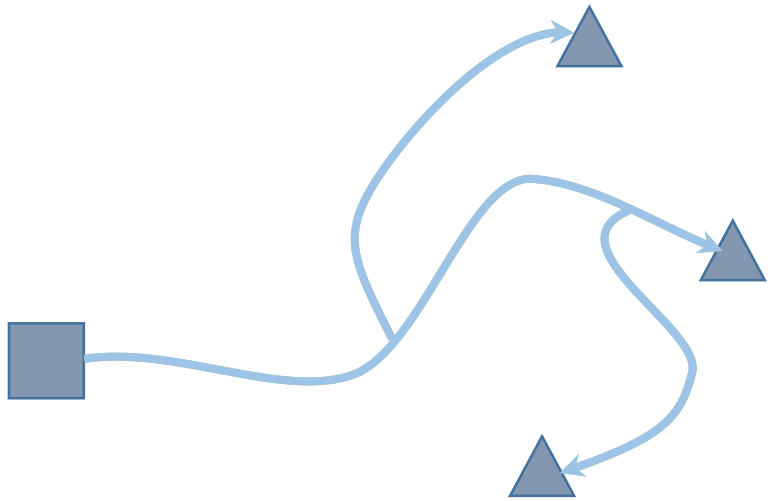
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



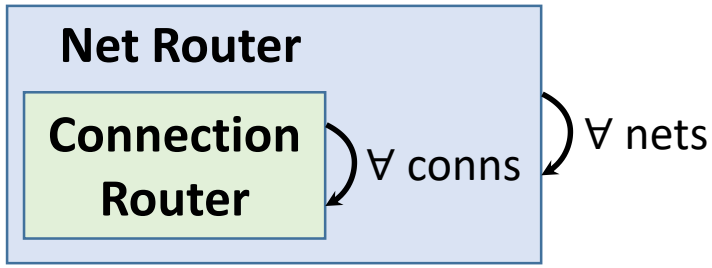
Negotiated Congestion routing



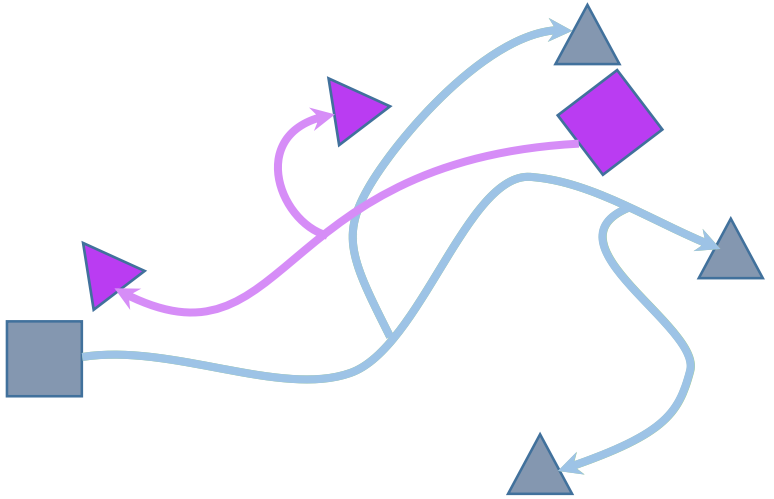
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



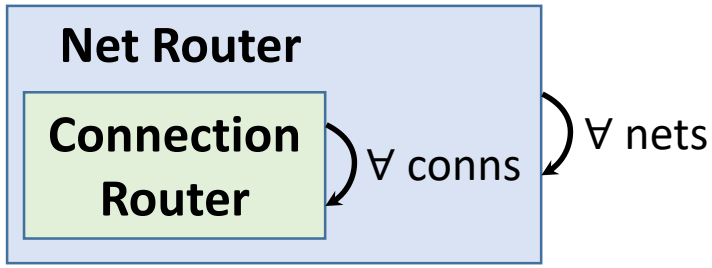
Negotiated Congestion routing



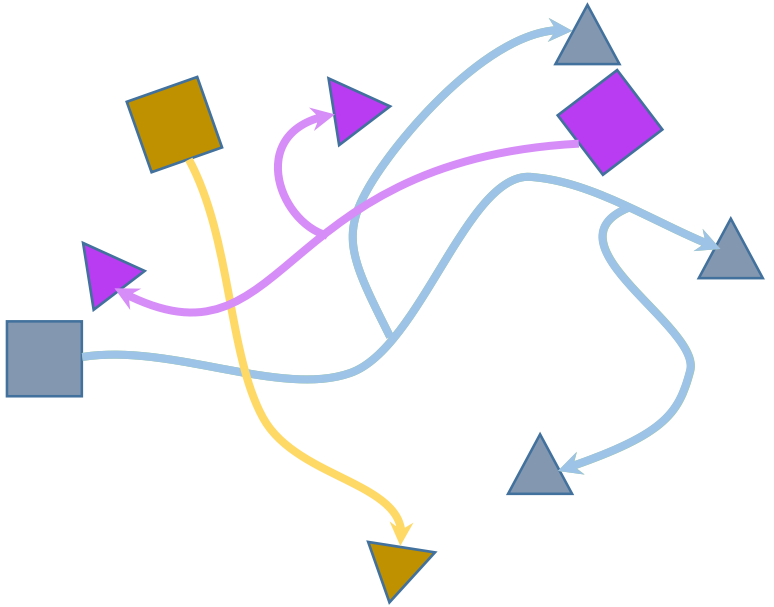
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



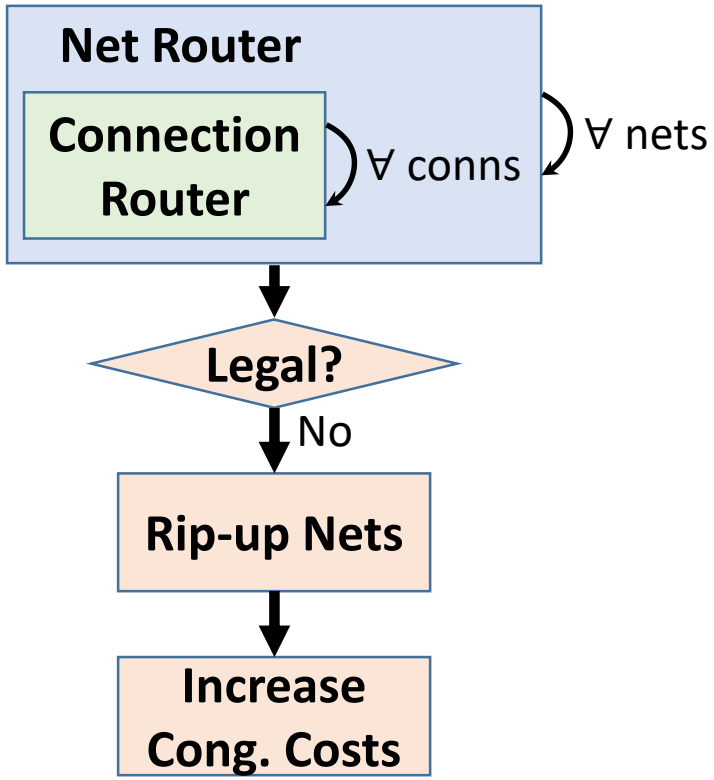
Negotiated Congestion routing



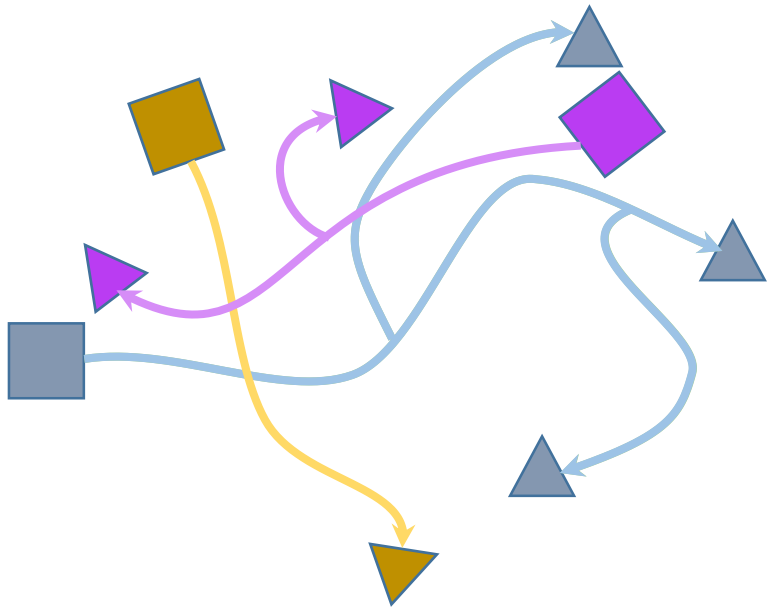
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



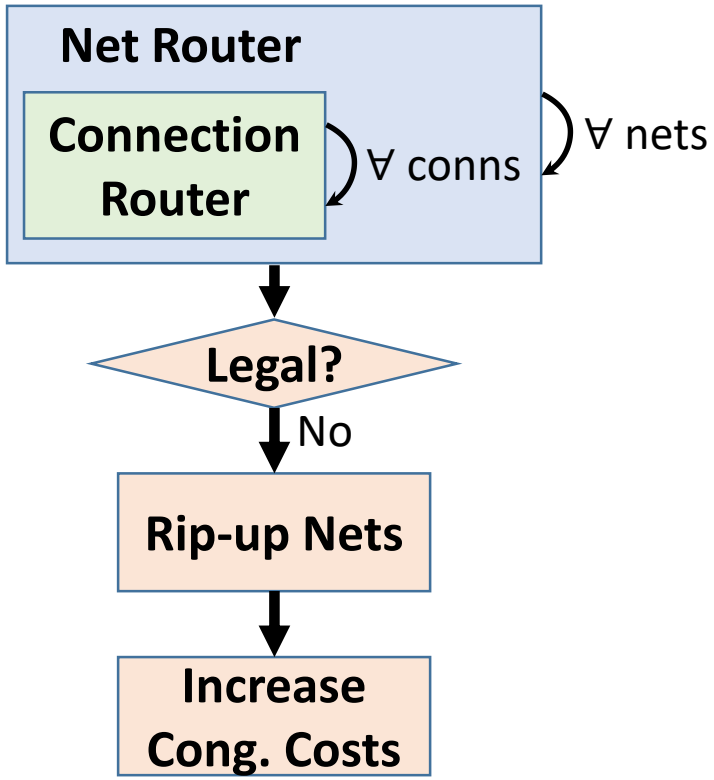
Negotiated Congestion routing



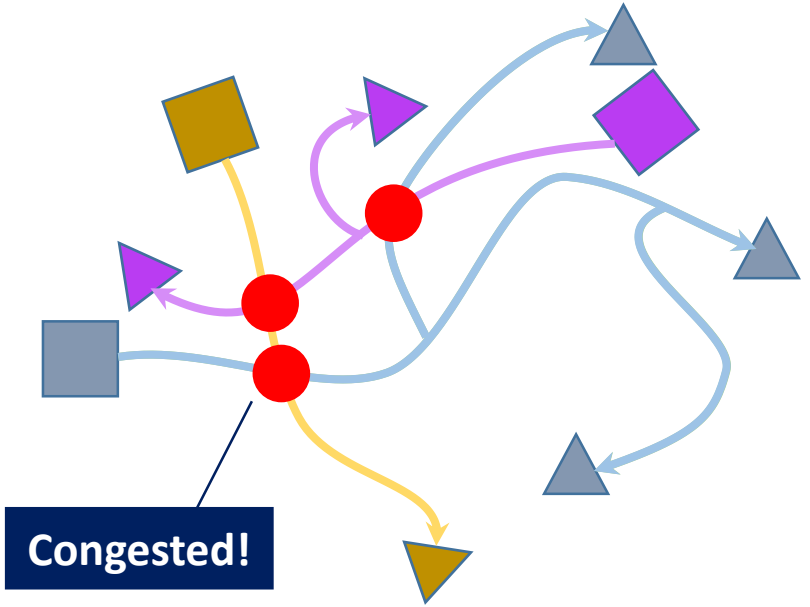
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



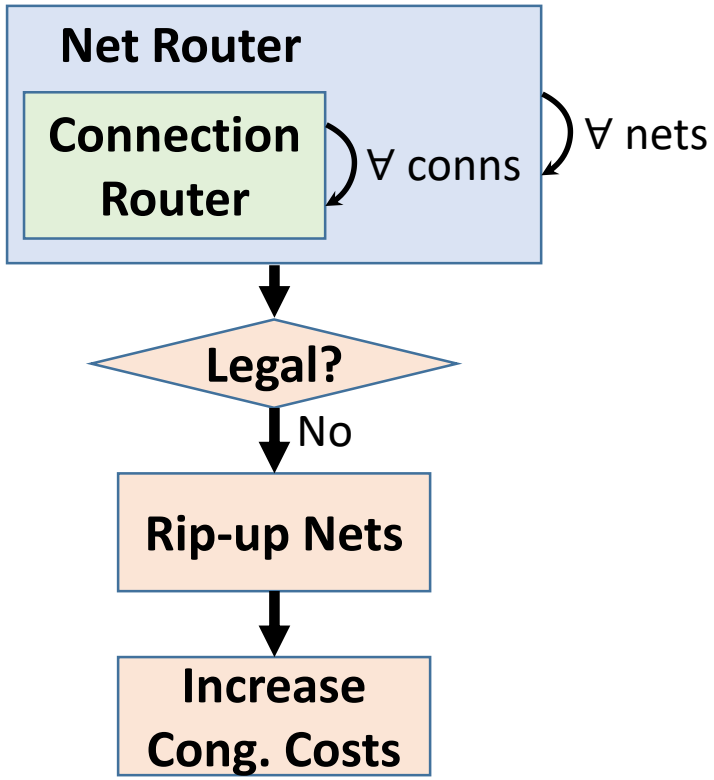
Negotiated Congestion routing



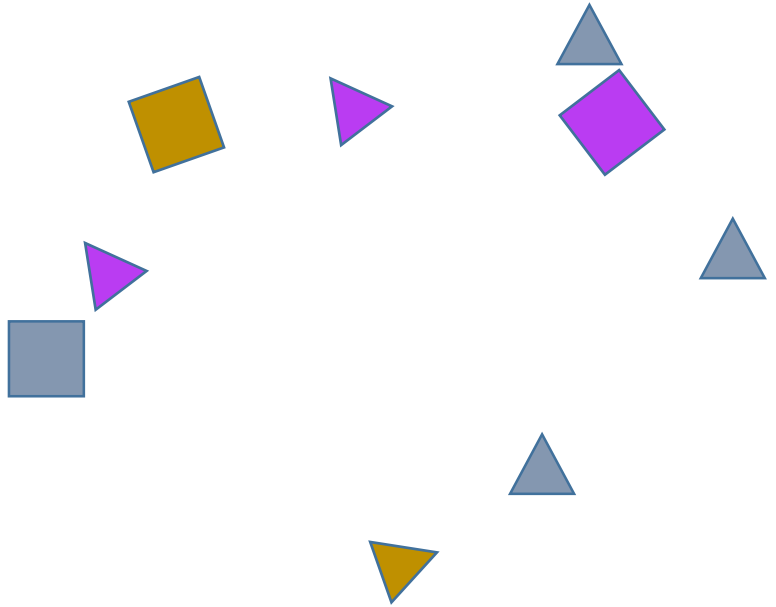
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



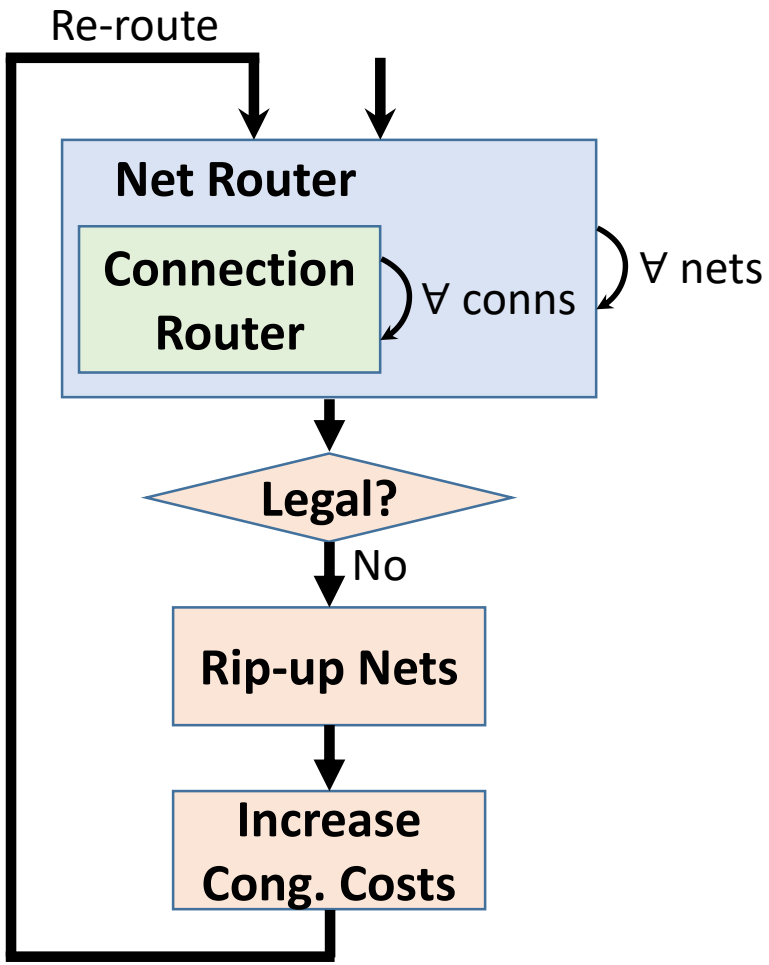
Negotiated Congestion routing



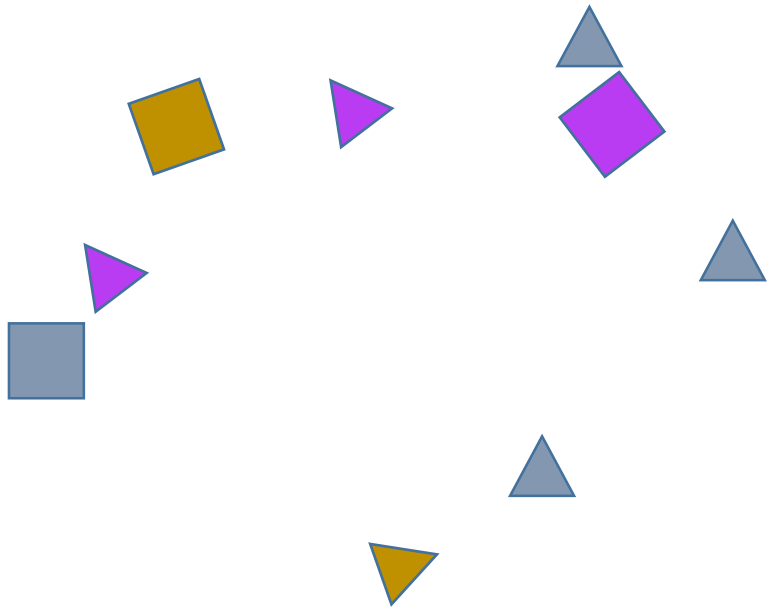
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



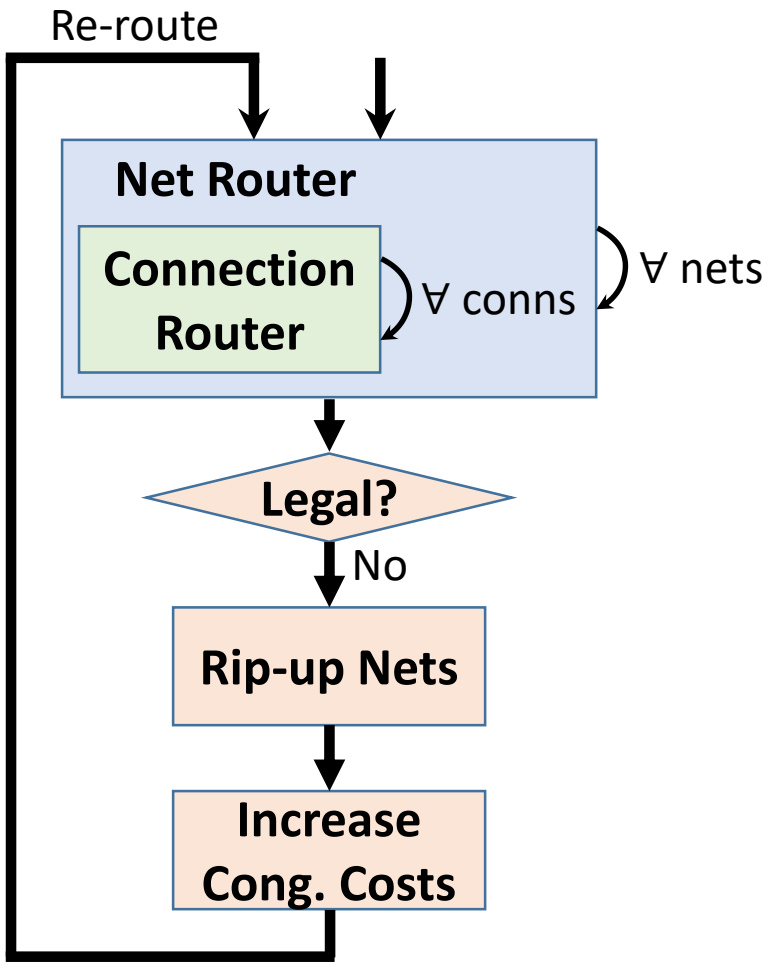
Negotiated Congestion routing



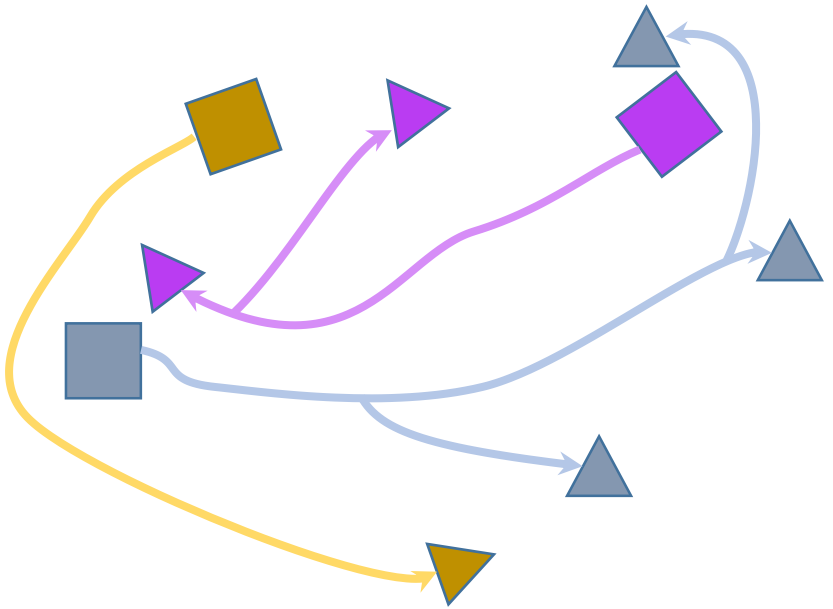
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



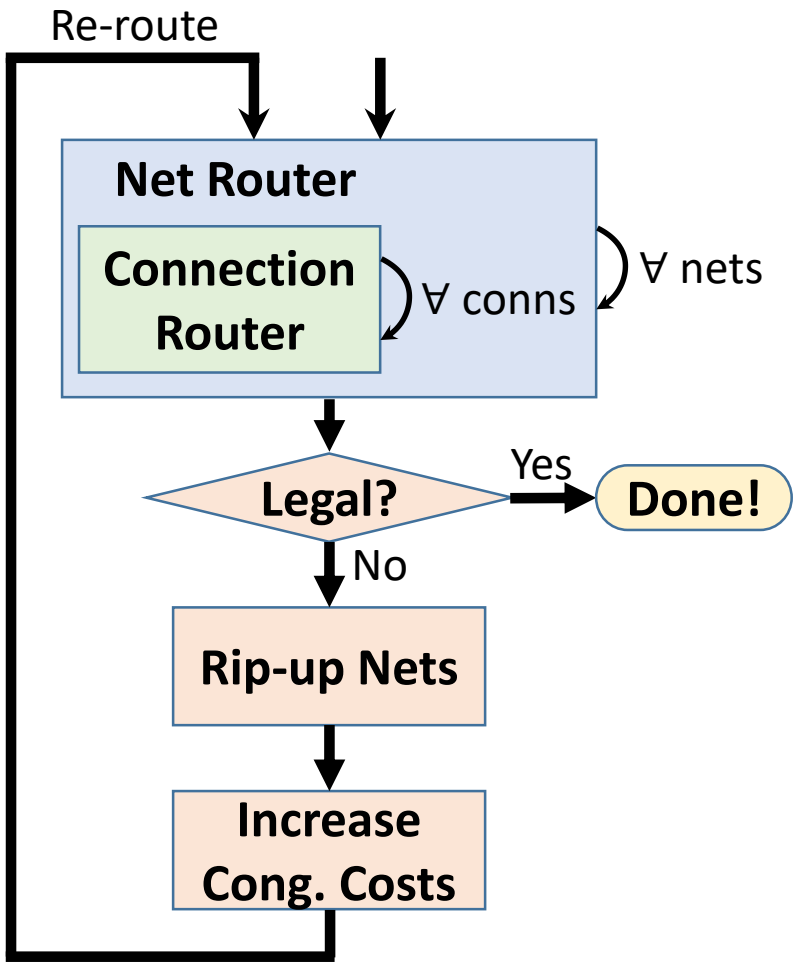
Negotiated Congestion routing



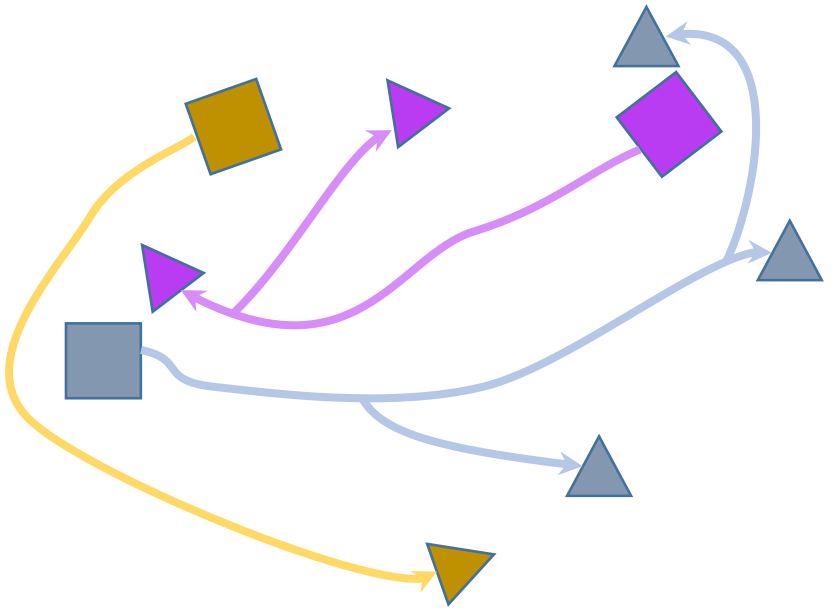
- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



Negotiated Congestion routing



- Allow multiple nets to use same resources (*congestion*)
  - Nets negotiate for resources
  - Nets do not block each other



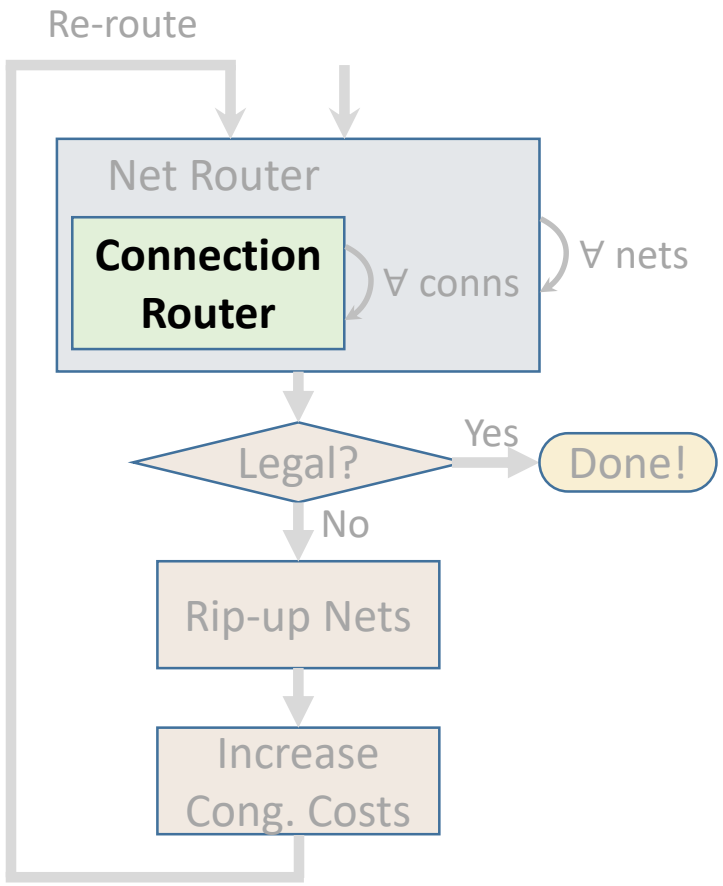
Congestion Free!

AIR: Adaptive Incremental Router

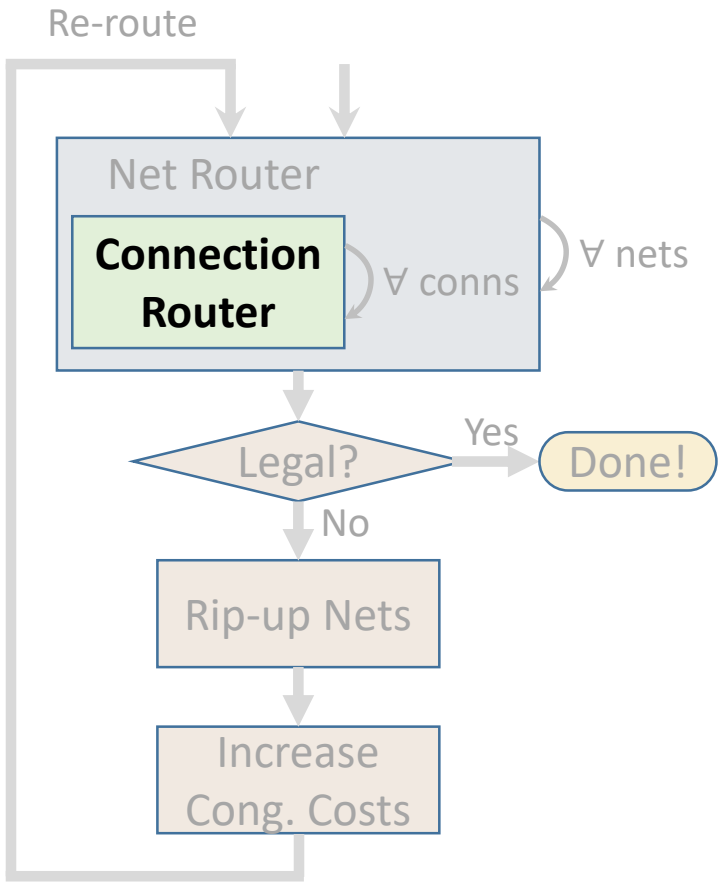
AIR

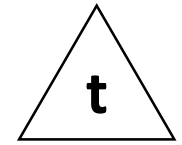
- Negotiated congestion router
- Adaptive Routing → **Improve quality & robustness**
- Lazy Routing → **Avoid unnecessary work!**

Connection Router: Architecture Adaptation



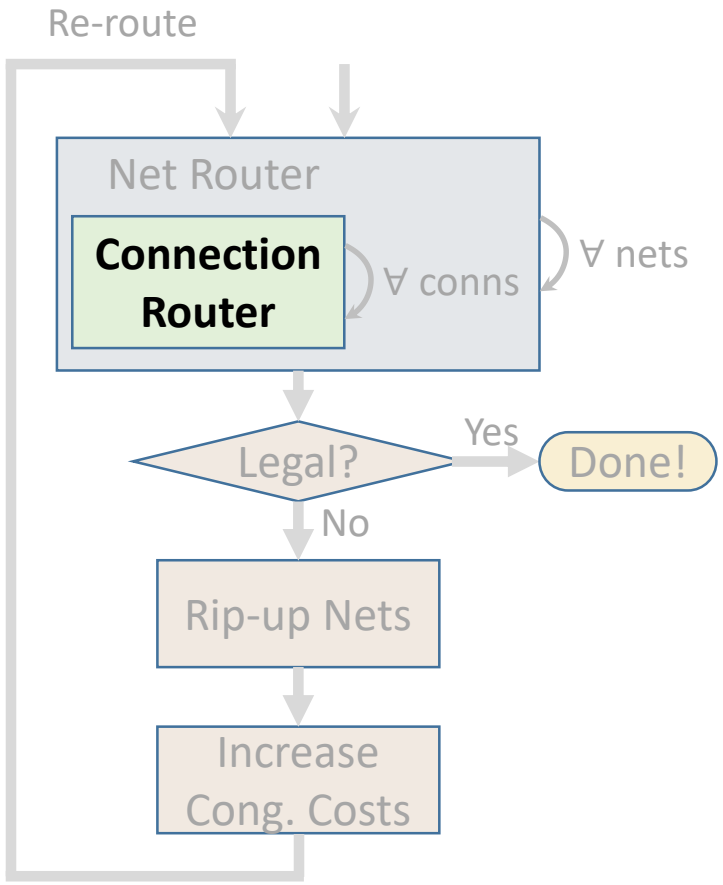
Connection Router: Architecture Adaptation

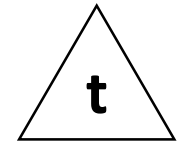




Connection Router: Architecture Adaptation

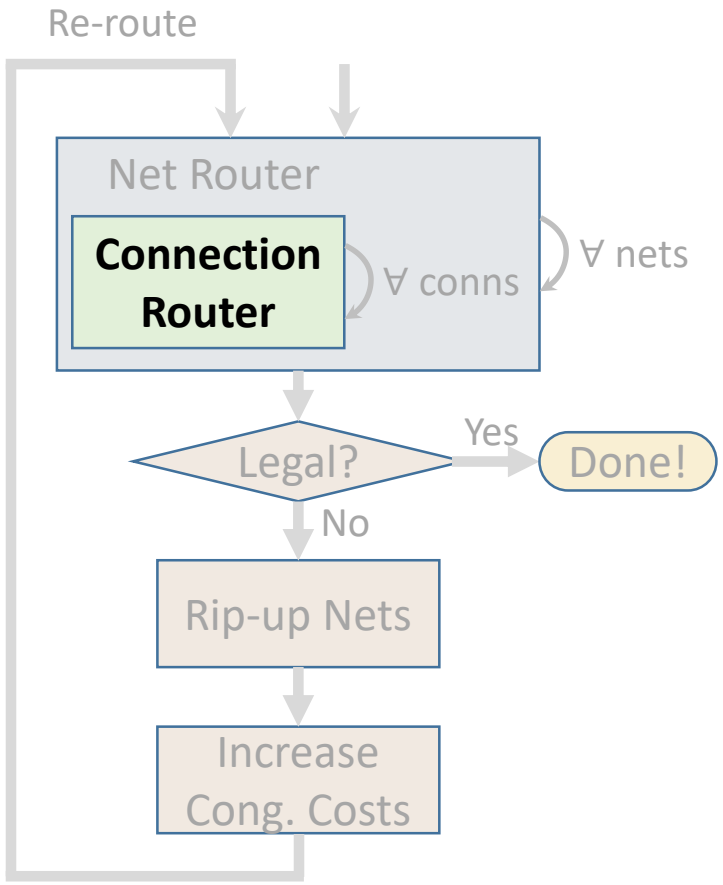
$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$

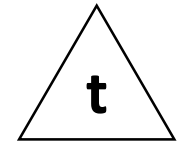




Connection Router: Architecture Adaptation

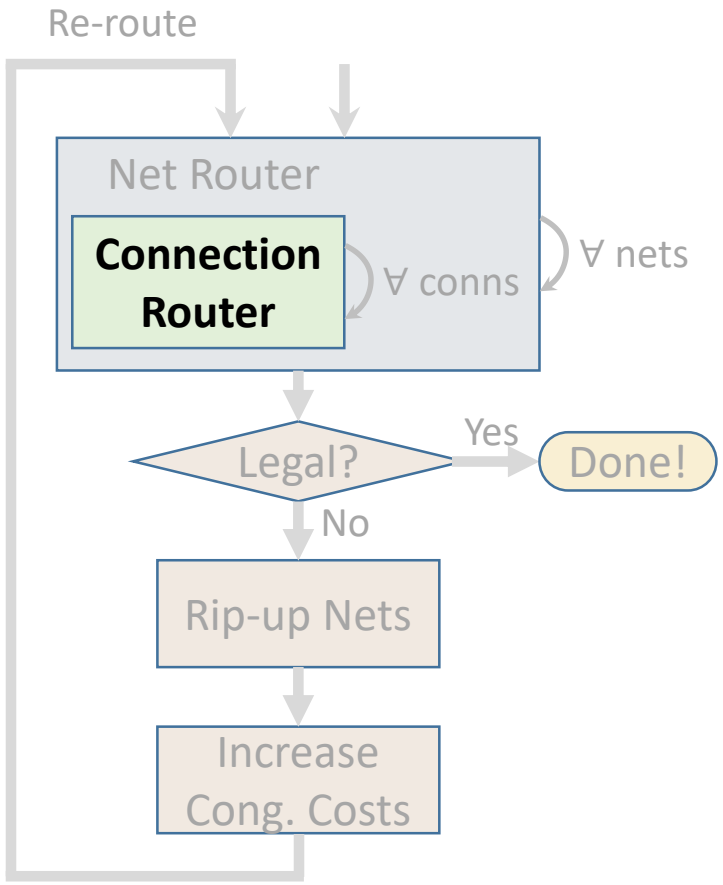
$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$

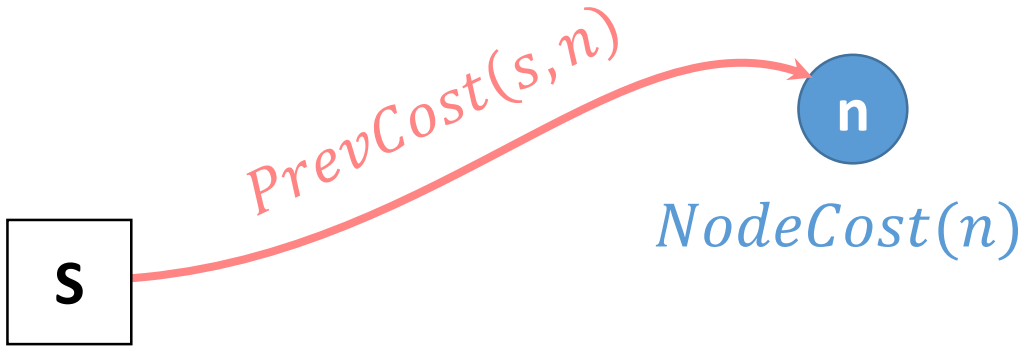


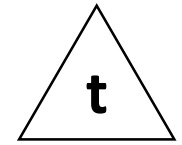


Connection Router: Architecture Adaptation

$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$

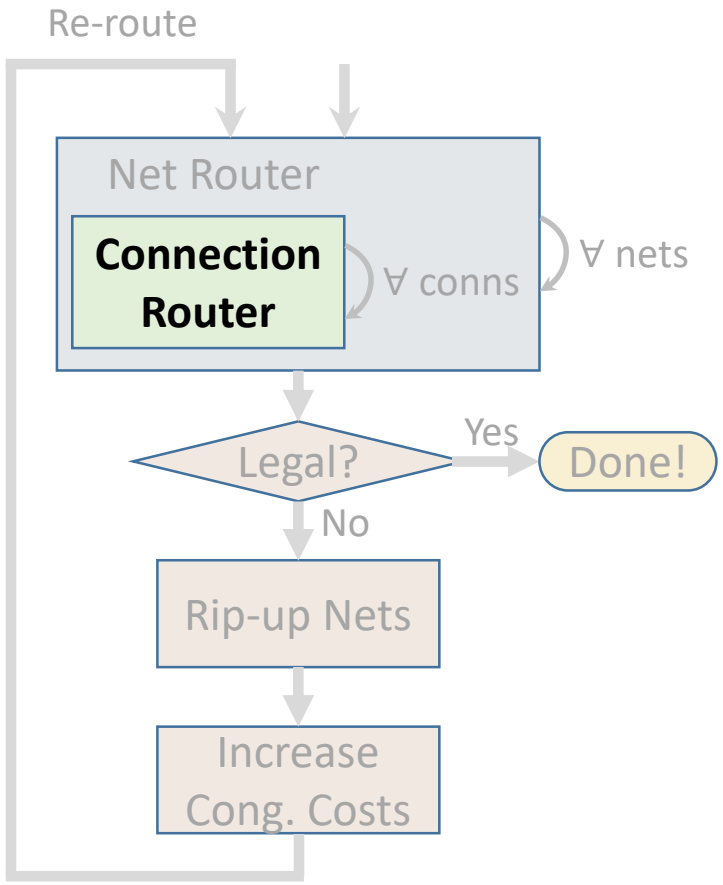


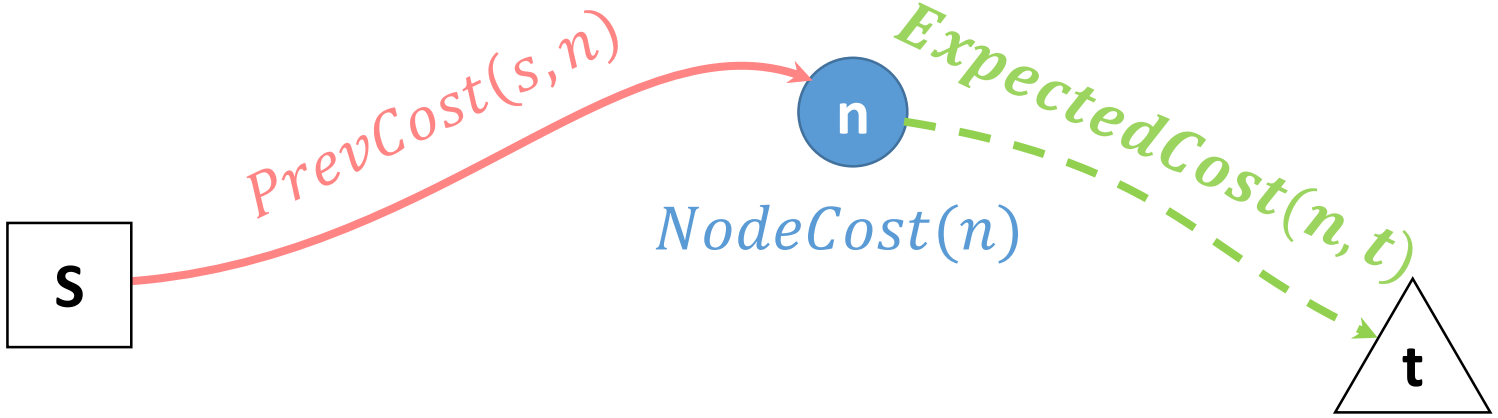




Connection Router: Architecture Adaptation

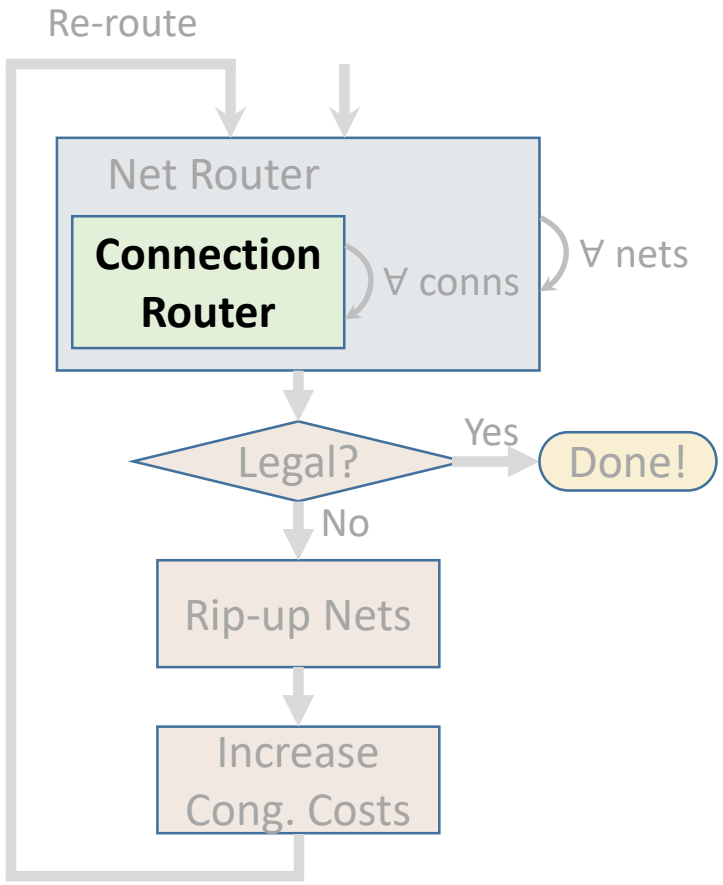
$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$

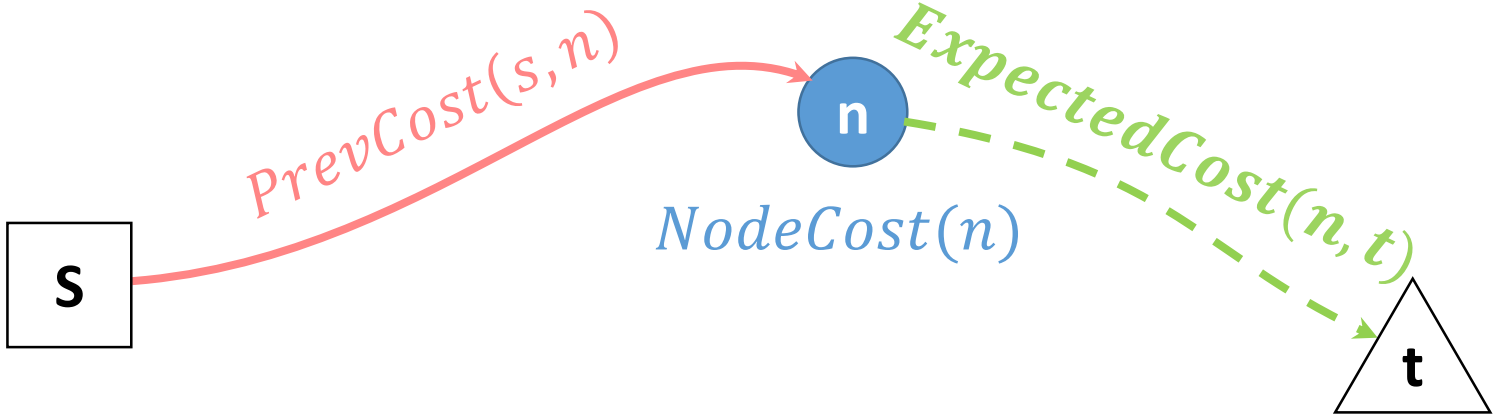




Connection Router: Architecture Adaptation

$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$

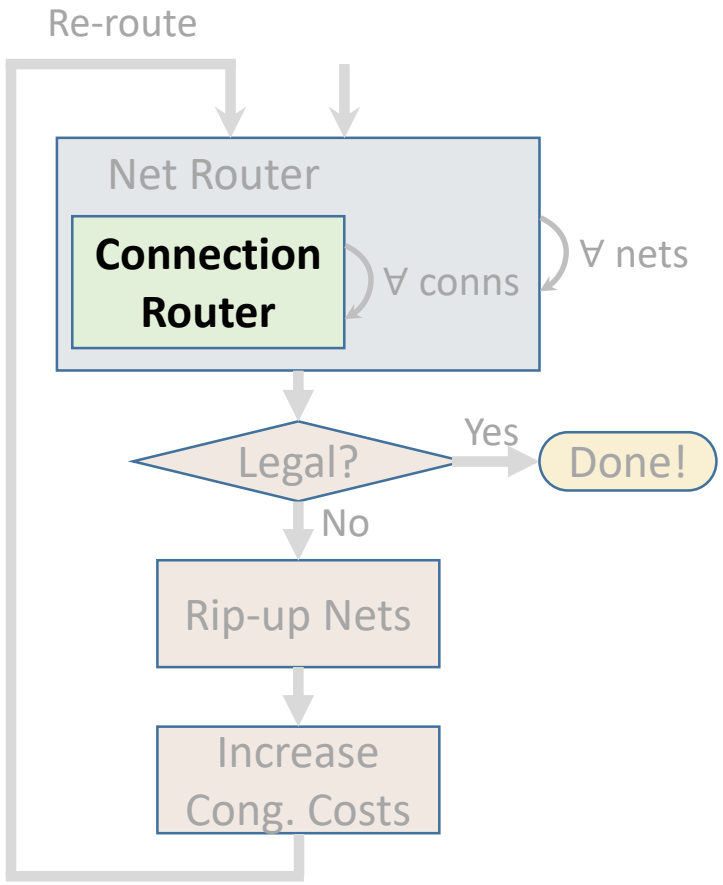


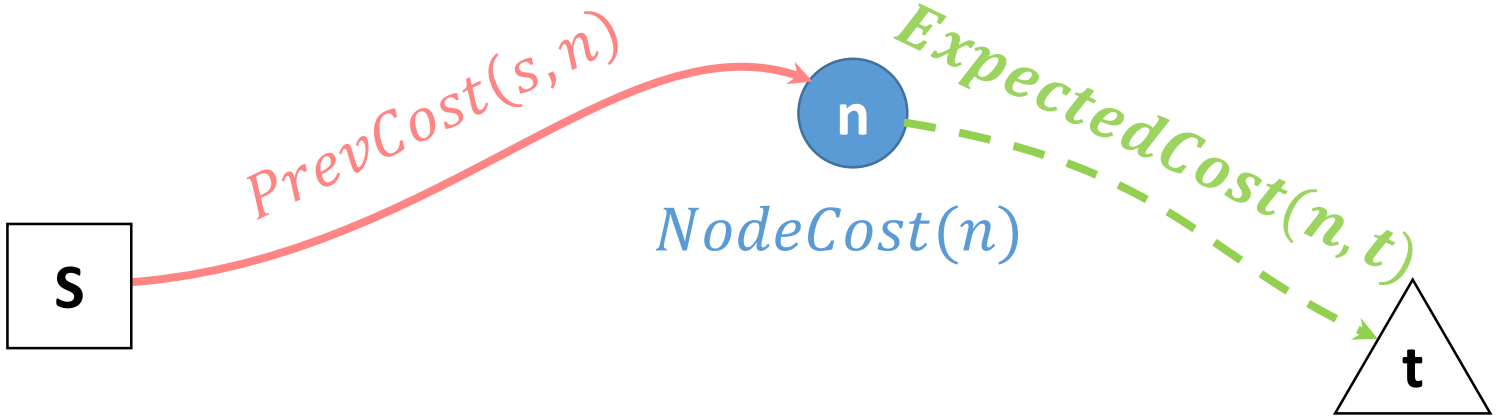


$$NodeCost(n) =$$

Connection Router: Architecture Adaptation

$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$

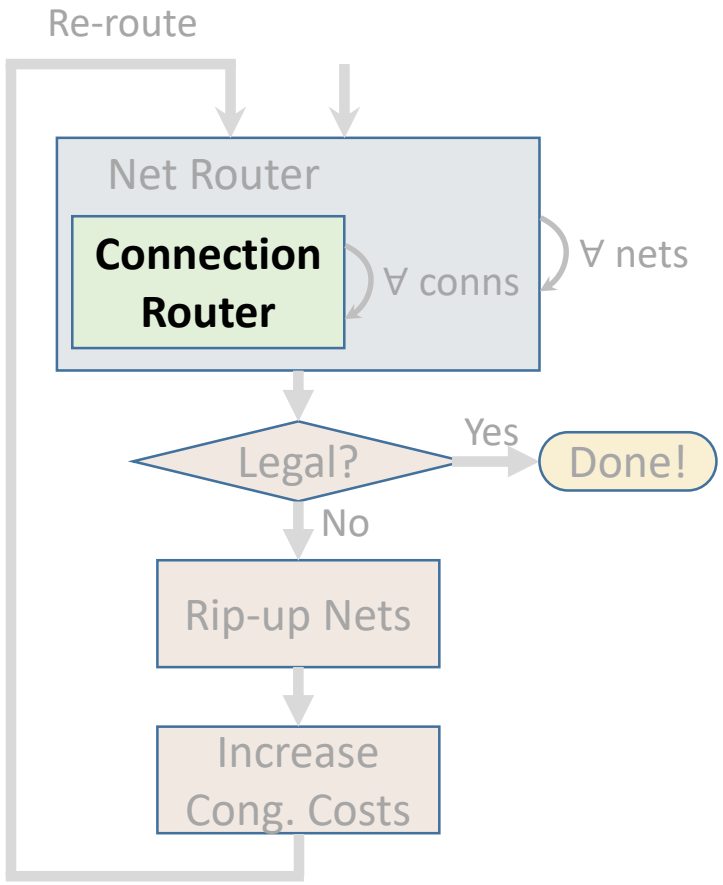


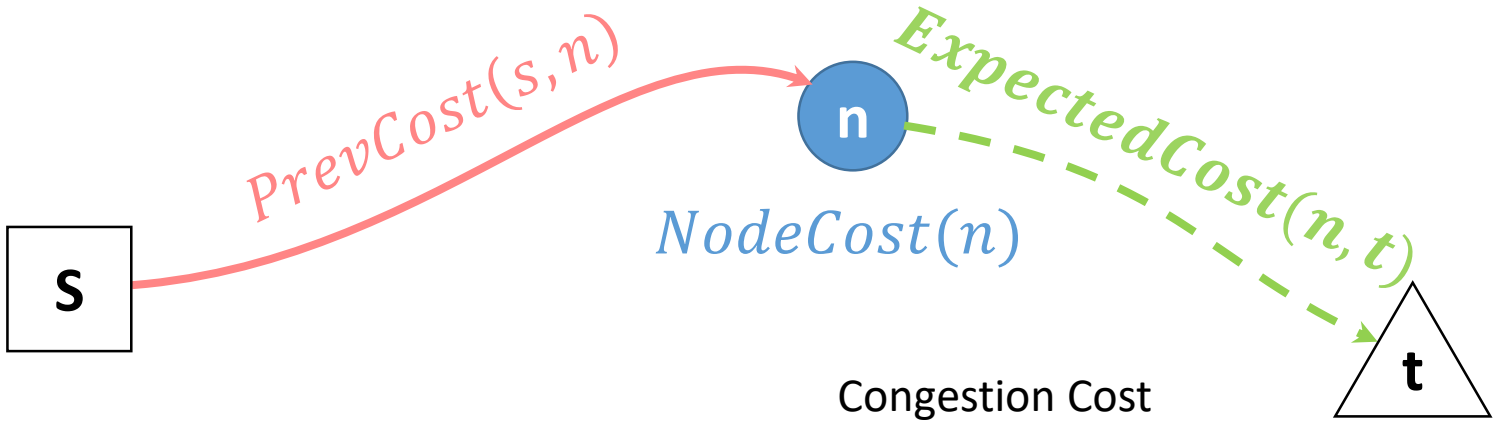


$$NodeCost(n) = \underbrace{b(n)}_{\text{Base Resource Cost}}$$

Connection Router: Architecture Adaptation

$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$

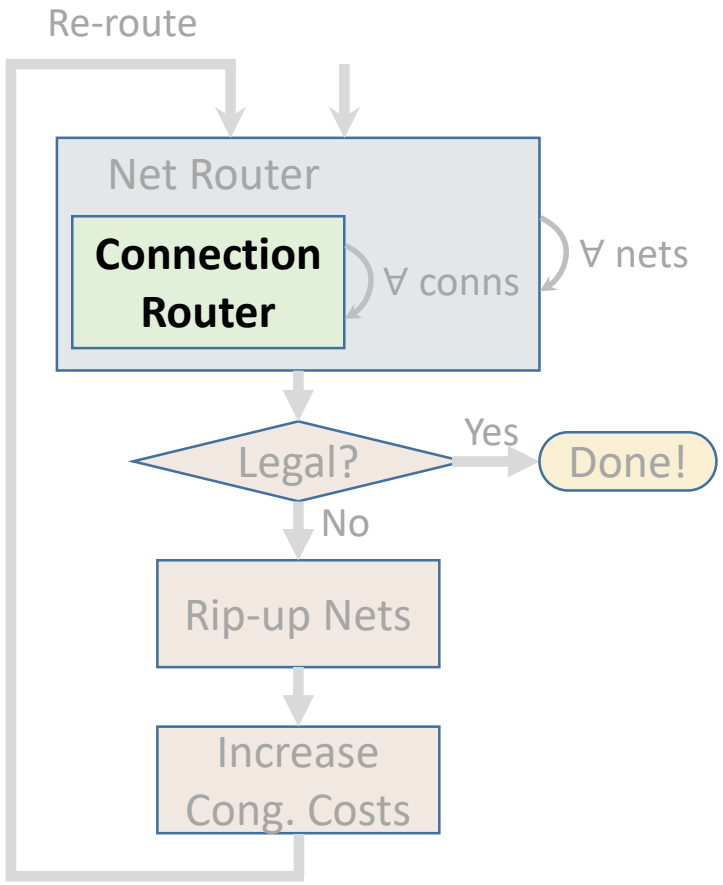


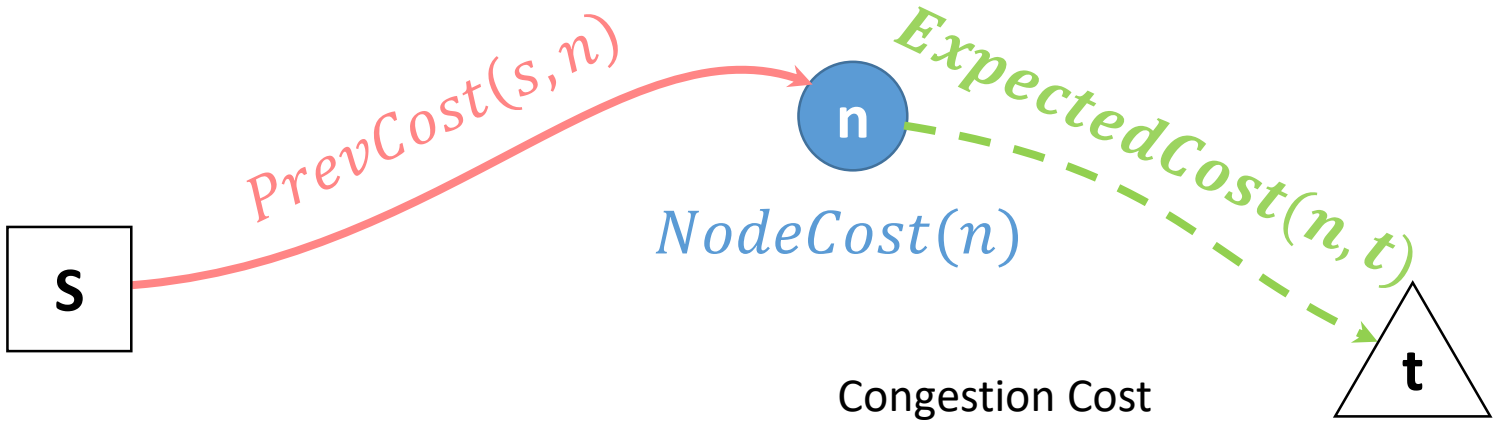


$$NodeCost(n) = \underbrace{b(n)}_{\text{Base Resource Cost}} \cdot \overbrace{c(n)}^{\text{Congestion Cost}}$$

Connection Router: Architecture Adaptation

$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$





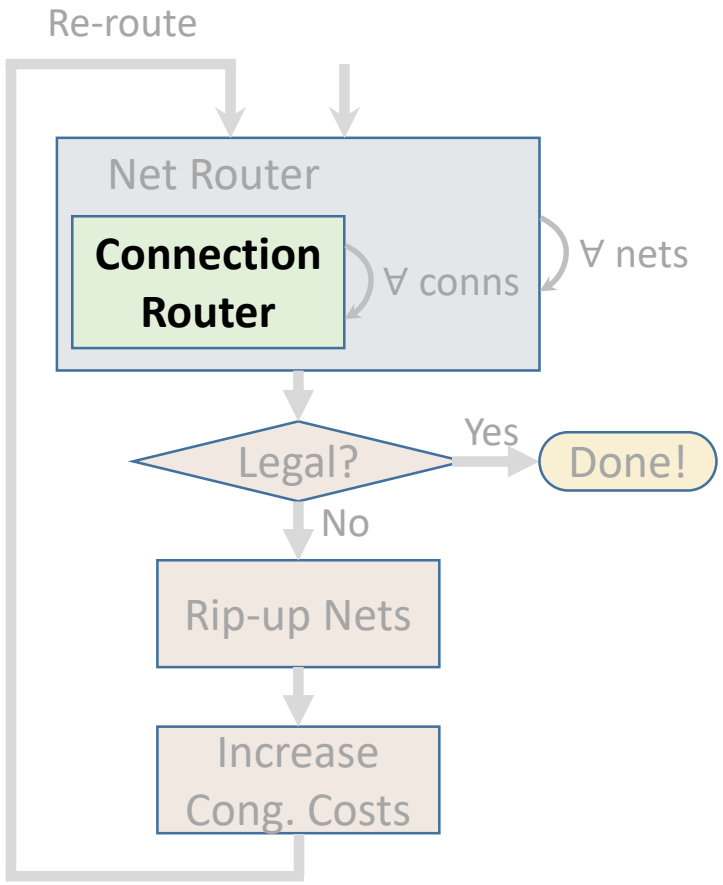
Congestion Cost

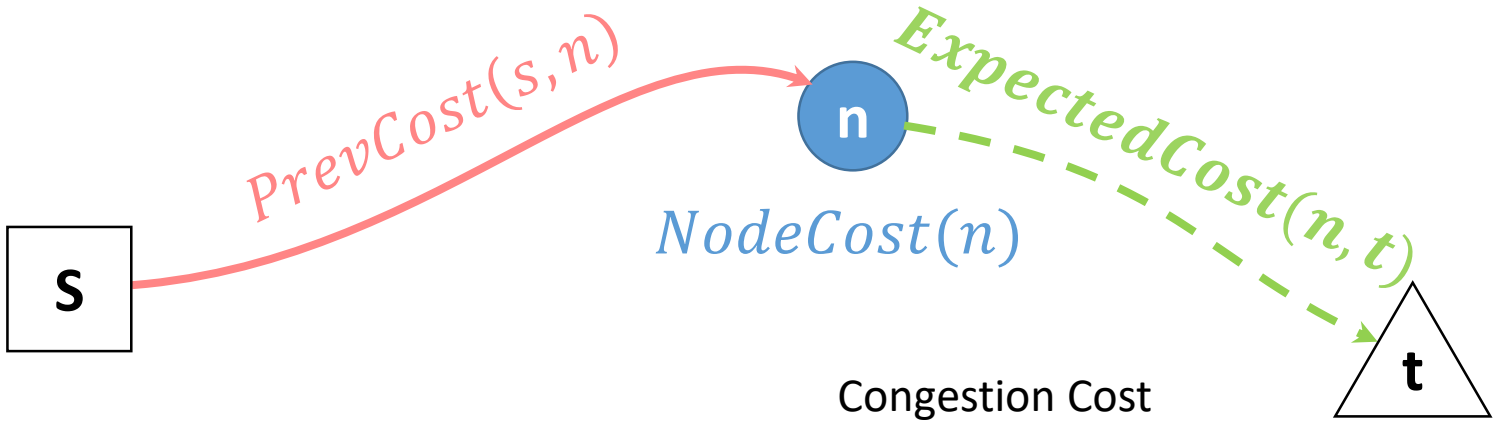
$$NodeCost(n) = \mathbf{b(n)} \cdot c(n) + Delay(n)$$

Base Resource Cost

Connection Router: Architecture Adaptation

$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$





Congestion Cost

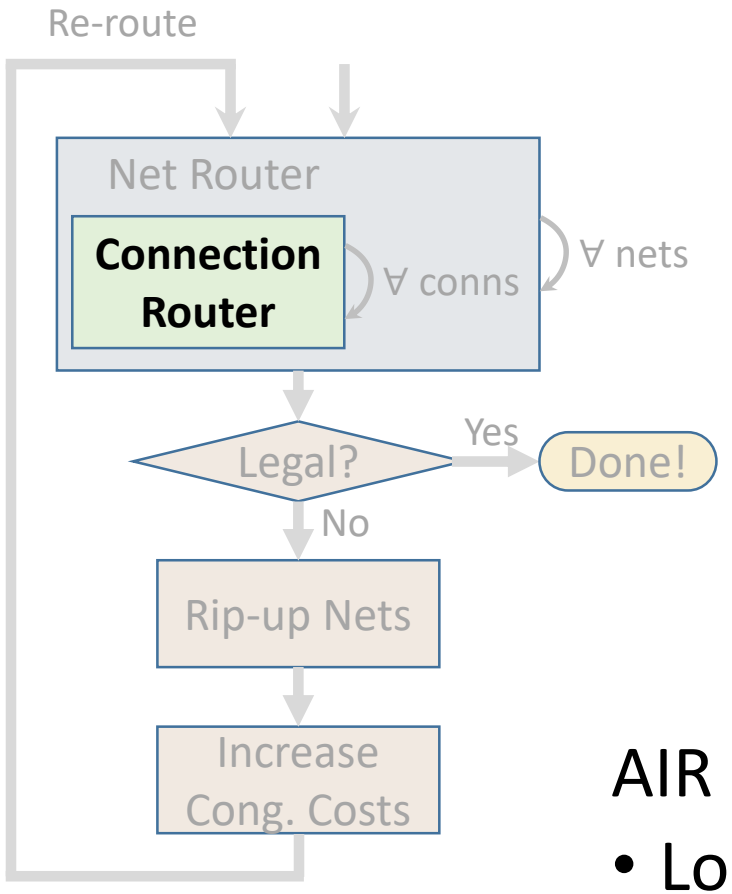
$$NodeCost(n) = (1-\gamma) \cdot \boldsymbol{b(n)} \cdot c(n) + \gamma \cdot Delay(n)$$

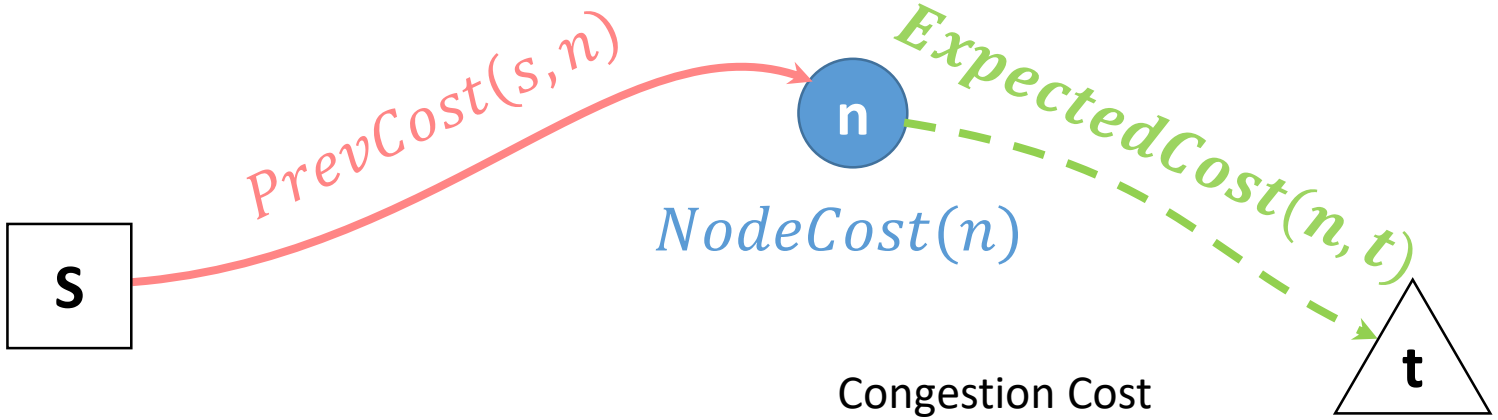
Base Resource Cost

Criticality

Connection Router: Architecture Adaptation

$$TotalCost(s,n,t) = PrevCost(s,n) + NodeCost(n) + ExpectedCost(n,t)$$





Congestion Cost

$$NodeCost(n) = (1-\gamma) \cdot \boldsymbol{b(n)} \cdot c(n) + \gamma \cdot Delay(n)$$

Base Resource Cost

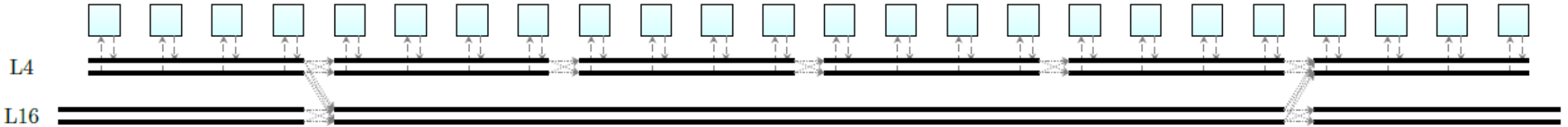
Criticality

AIR uses adapt***in)***)

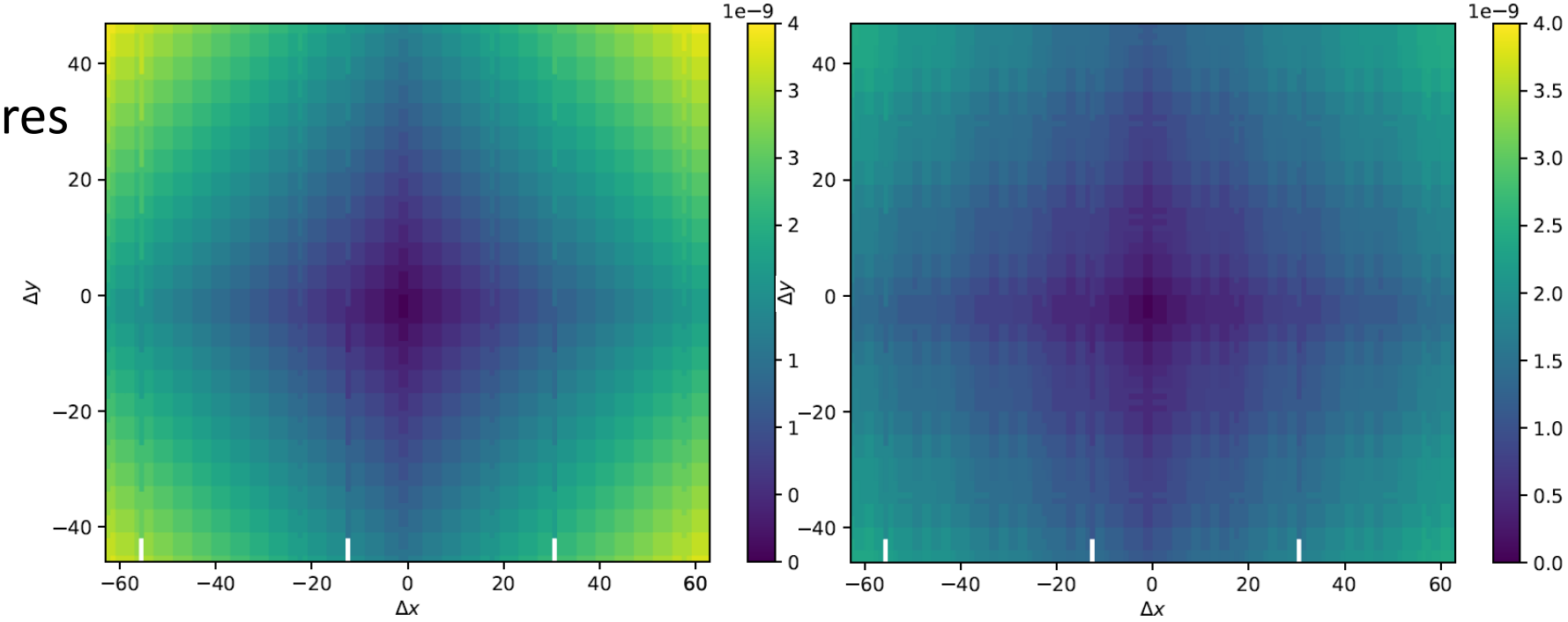
- Lookahead (***ExpectedCost***)
- Base Costs (***b(n)***)

Adaptive Lookahead

- Observation: Lookahead should adapt to routing architecture



- Challenges:
  - Many FPGA Architectures
  - Huge Graph
  - Reasonable build time
  - Fast evaluation
- Approach:
  - Profile graph with un-directed Djikstra flood
  - Adapts to graph
  - Faster for equivalent quality

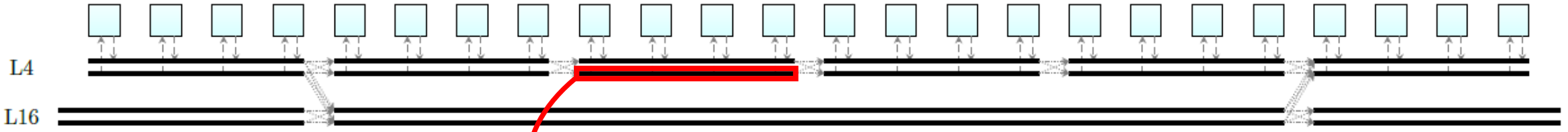


Classic Lookahead Delay

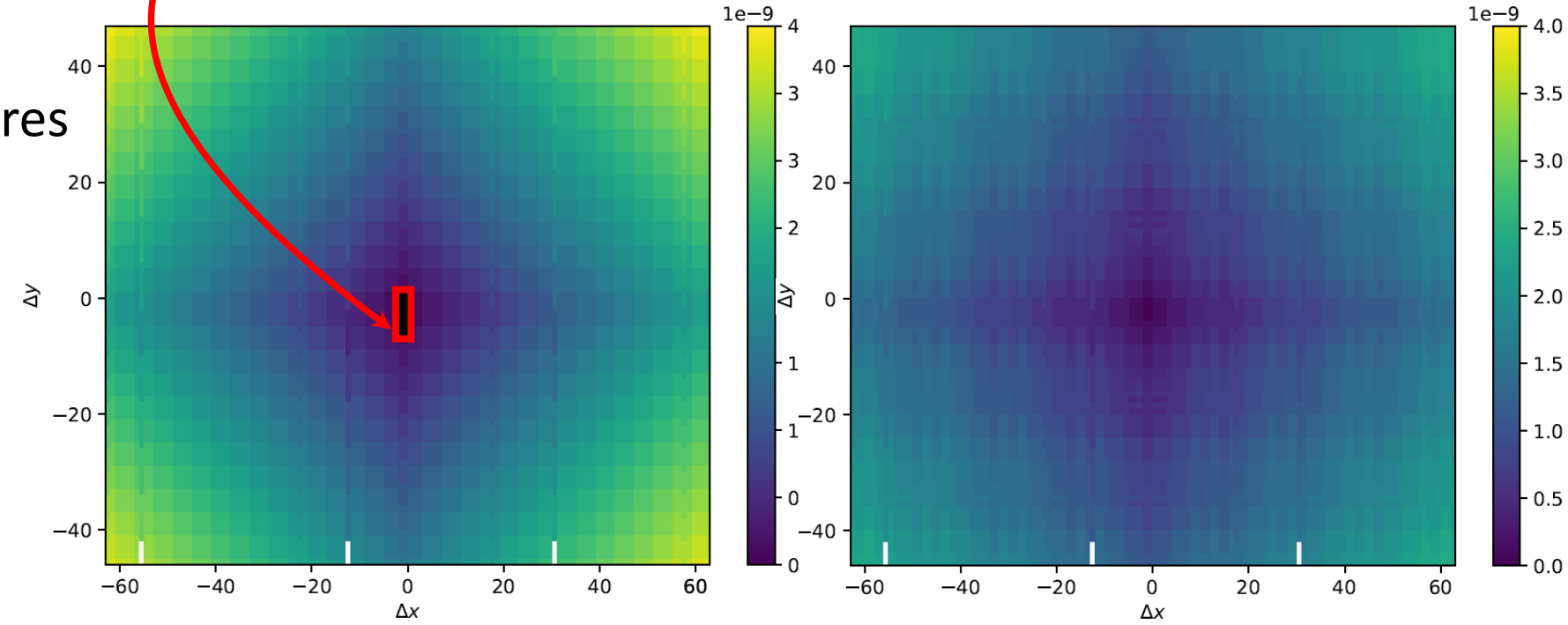
Map Lookahead Delay

Adaptive Lookahead

- Observation: Lookahead should adapt to routing architecture



- Challenges:
  - Many FPGA Architectures
  - Huge Graph
  - Reasonable build time
  - Fast evaluation
- Approach:
  - Profile graph with un-directed Djikstra flood
  - Adapts to graph
  - Faster for equivalent quality

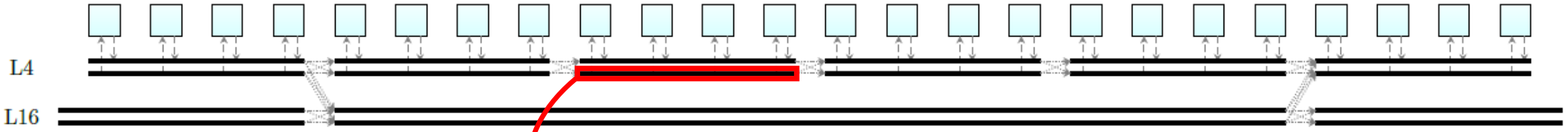


Classic Lookahead Delay

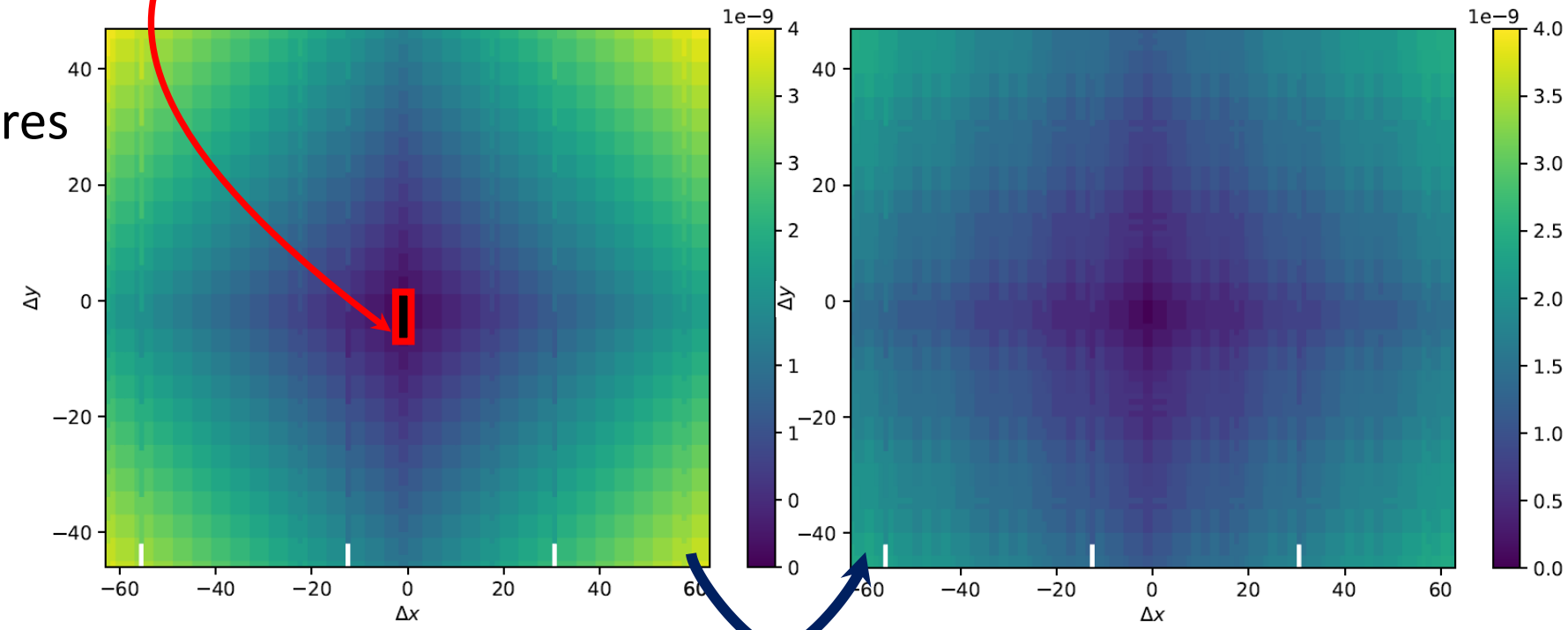
Map Lookahead Delay

Adaptive Lookahead

- Observation: Lookahead should adapt to routing architecture

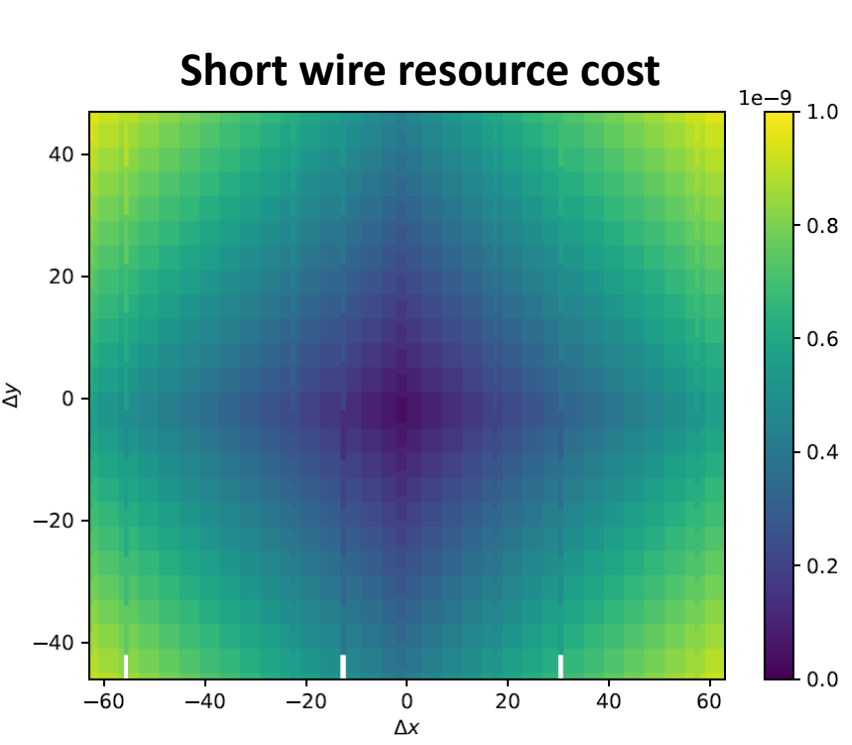


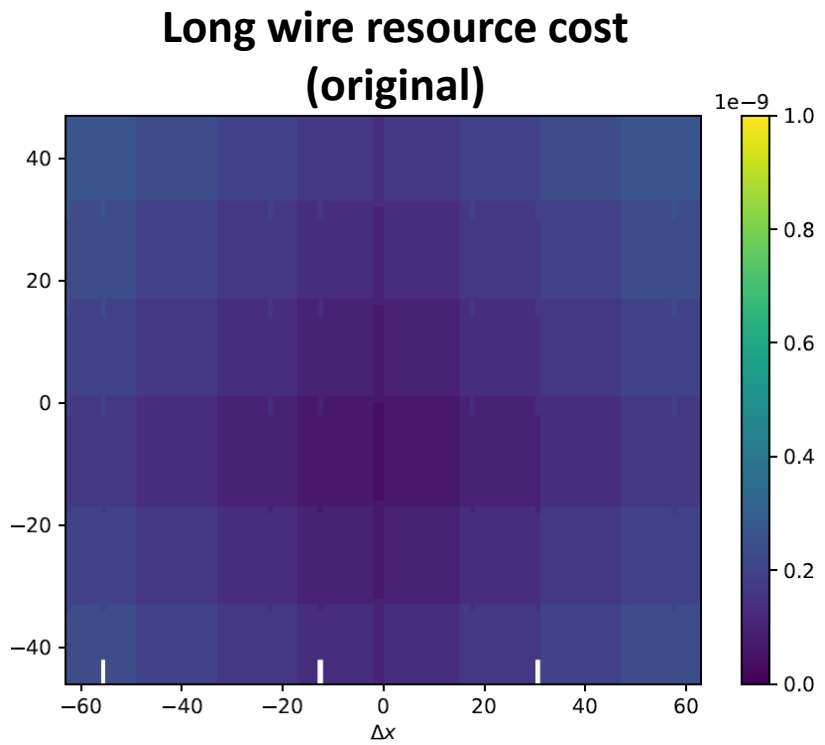
- Challenges:
  - Many FPGA Architectures
  - Huge Graph
  - Reasonable build time
  - Fast evaluation
- Approach:
  - Profile graph with un-directed Djikstra flood
  - Adapts to graph
  - Faster for equivalent quality

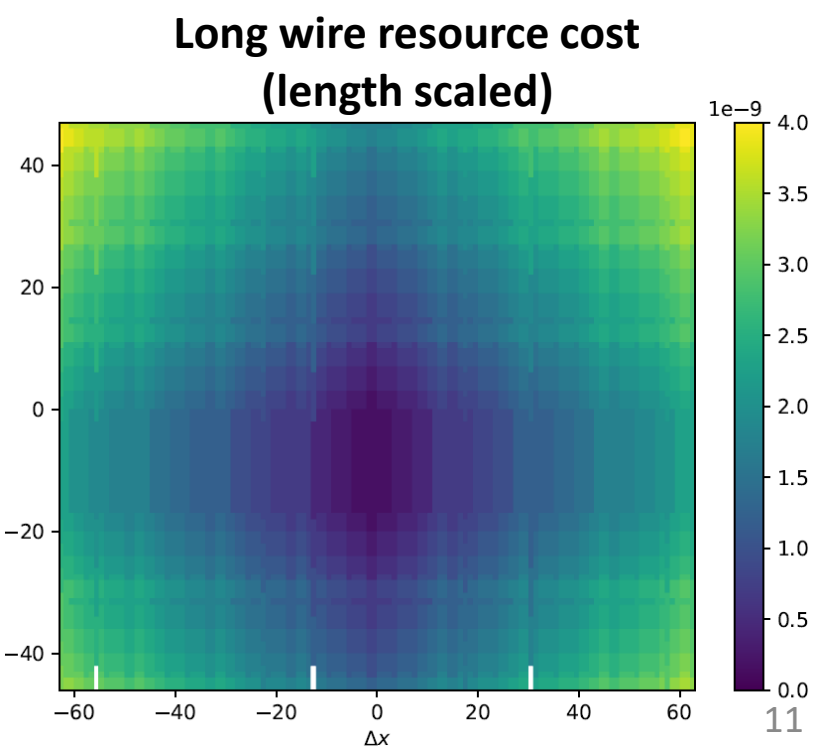


Resource Base Costs

- Observation: Strong preference for long wires leads to congestion
  - Long wires were:
    - Fast (preferred for timing)
    - Cheap (preferred for wirelength)
- Optimization: Scale resource cost by length



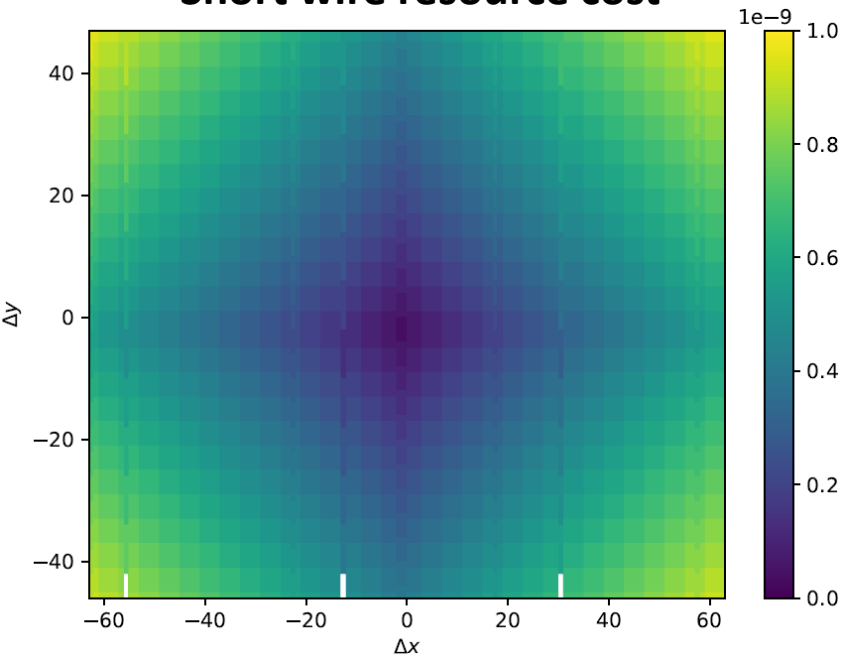


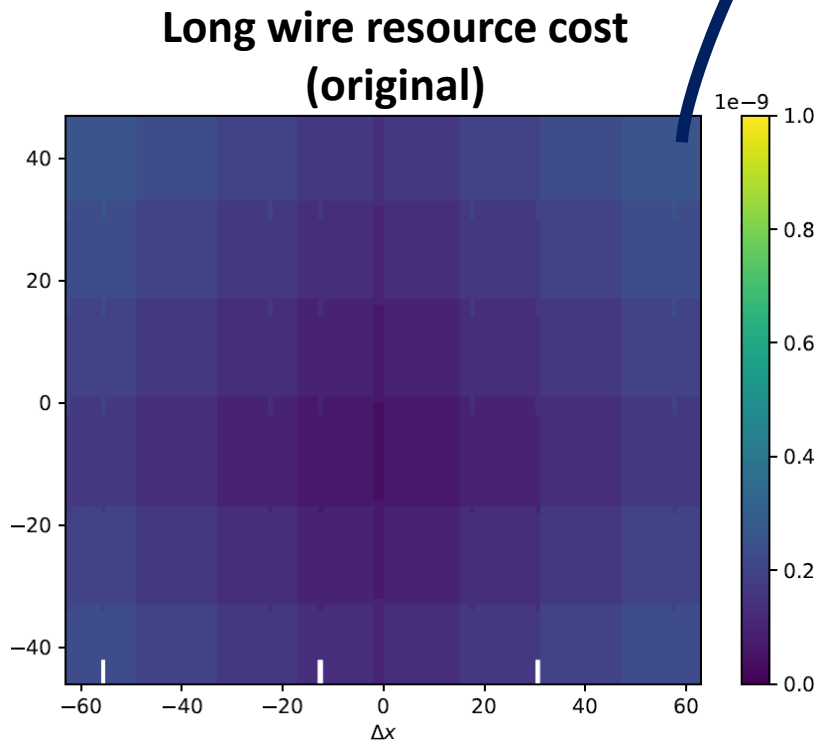


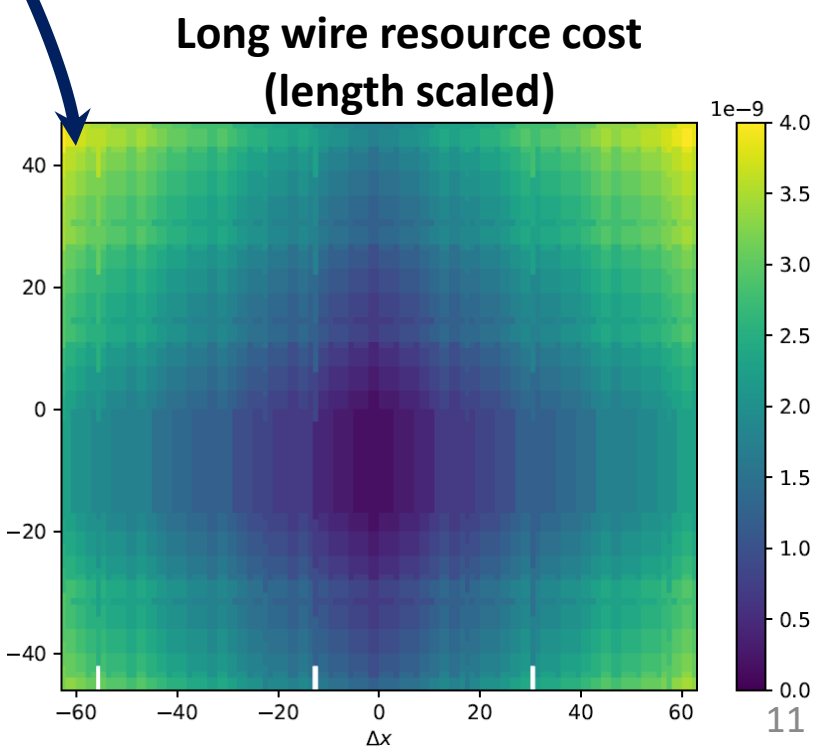
Resource Base Costs

- Observation: Strong preference for long wires leads to congestion
  - Long wires were:
    - Fast (preferred for timing)
    - Cheap (preferred for wirelength)
- Optimization: Scale resource cost by length

No longer cheaper to use long wires!



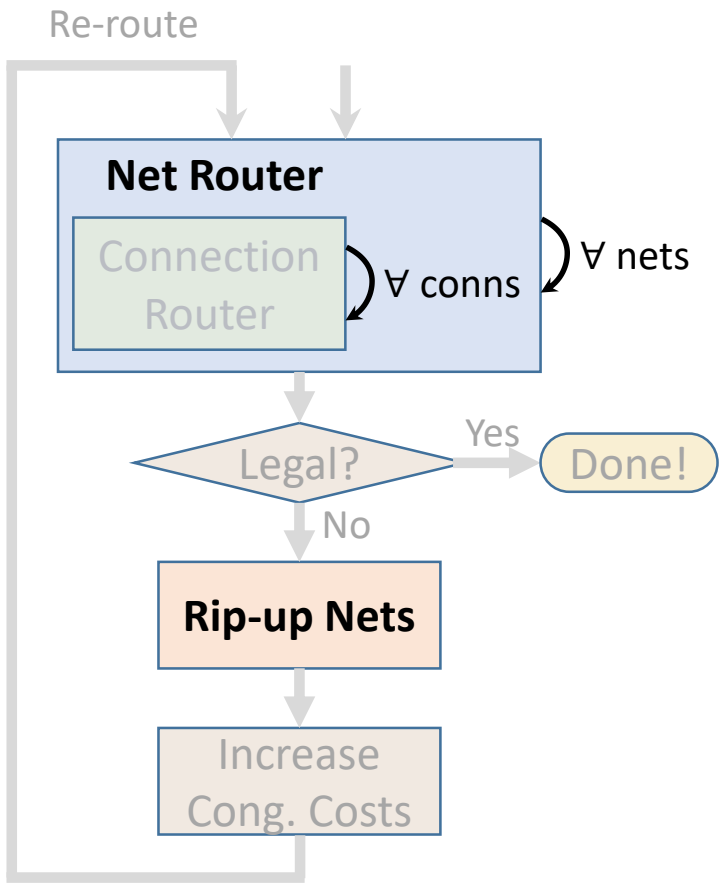




Resource Base Costs

- Observation: Strong preference for long wires leads to congestion
  - Long wires were:
    - Fast (preferred for timing)
    - Cheap (preferred for wirelength)
- Optimization: Scale resource cost by length

Net Router: Lazy Routing

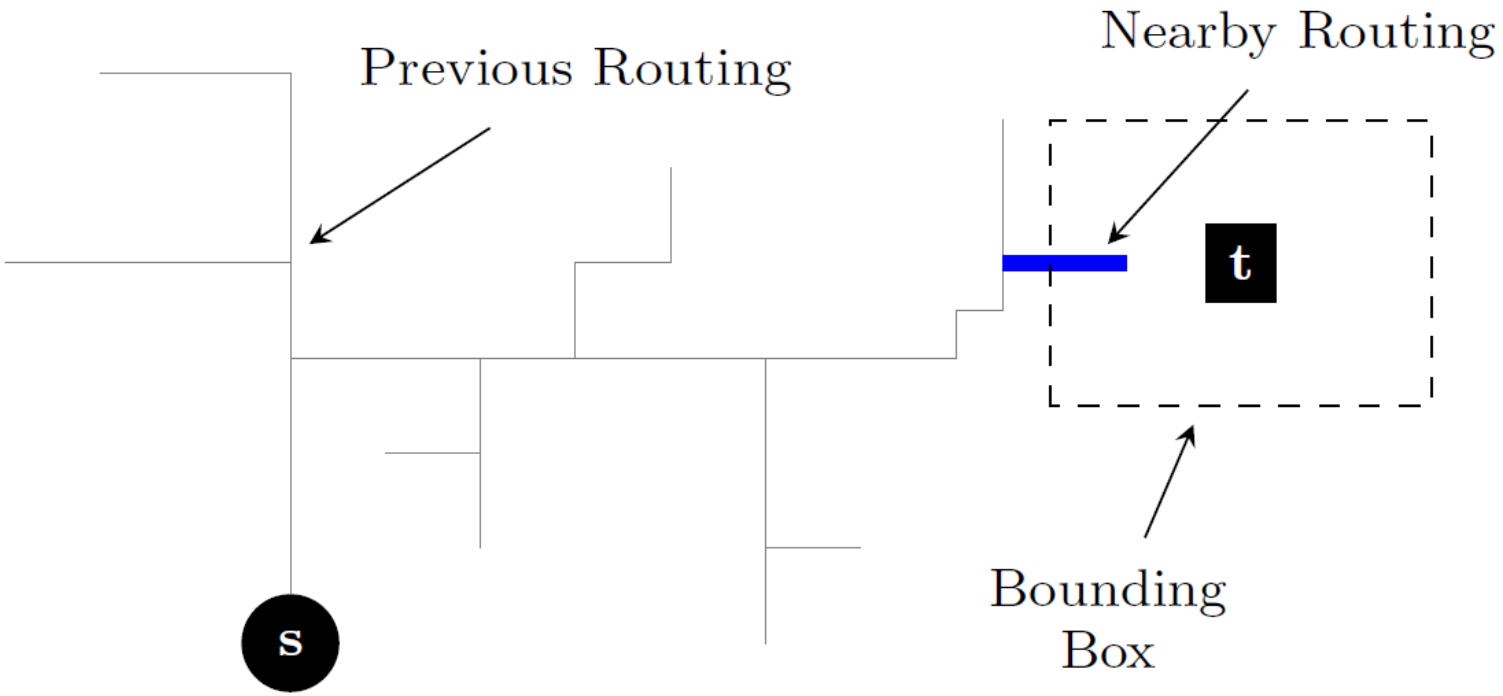


AIR Routes Nets Lazily:

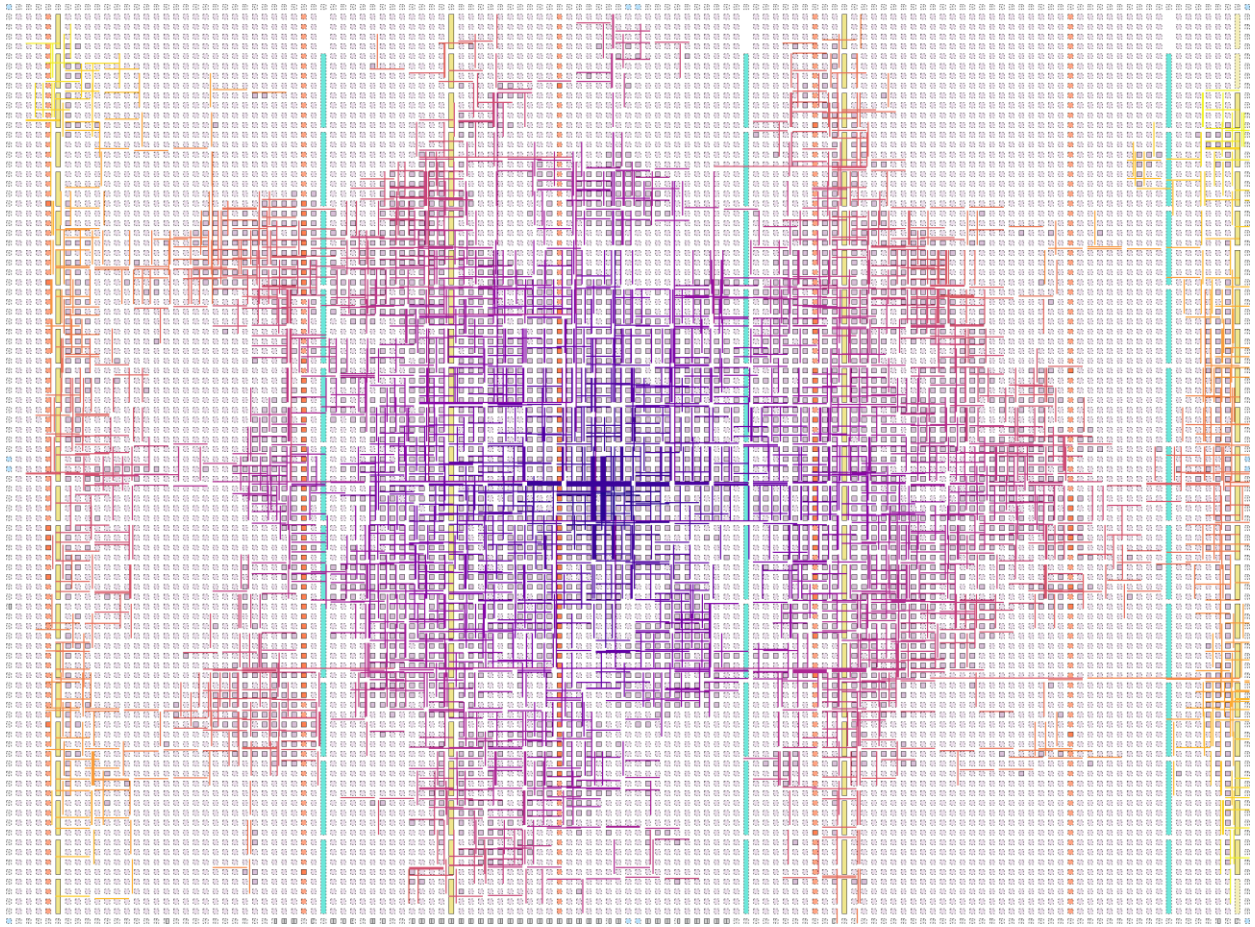
- Efficient for High-Fanout nets
- Incrementally re-route congested nets

High-Fanout (HF) Routing

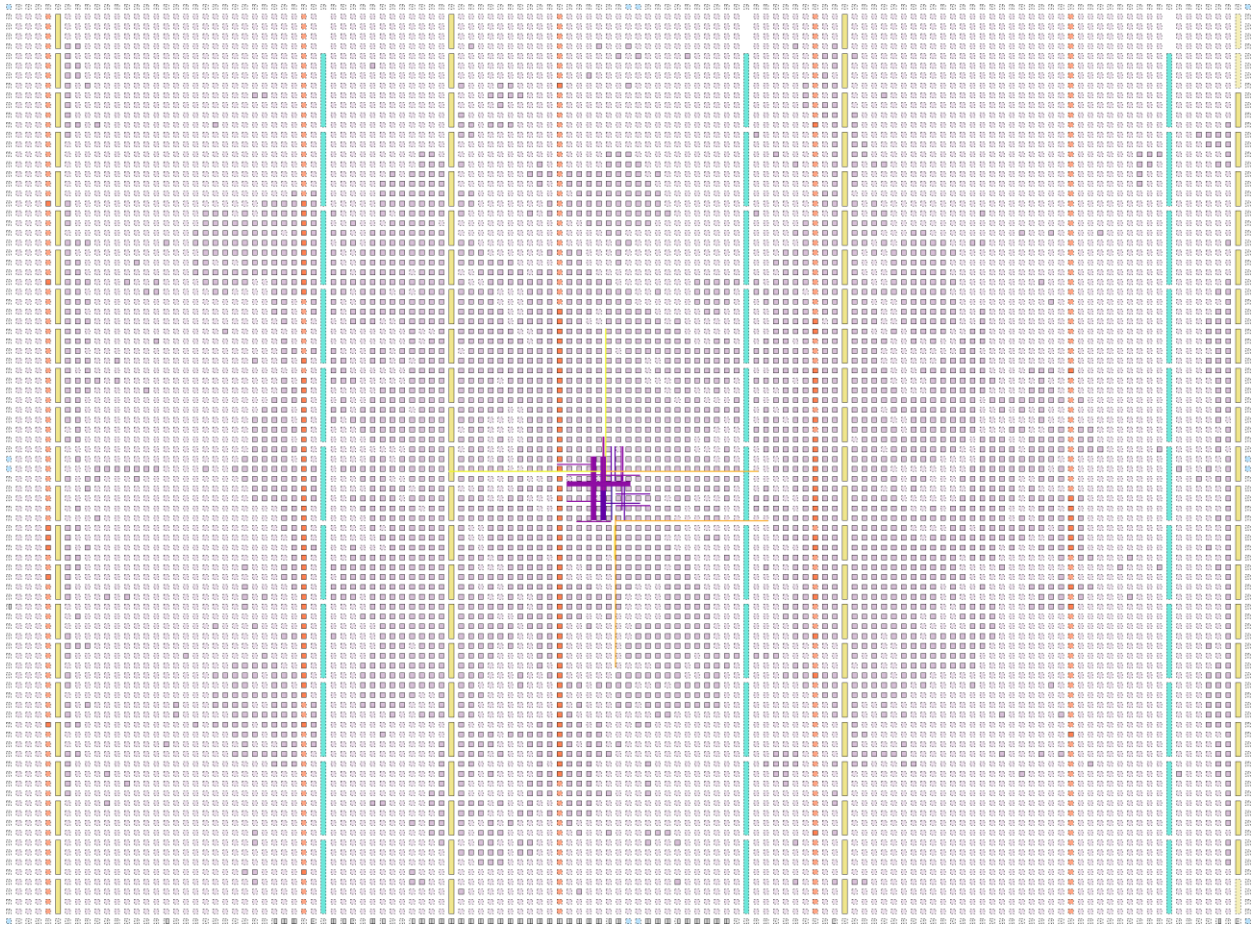
- Observation: Most routing of HF net irrelevant for a particular sink
  - But still consider as potential branch points (expensive)
- Optimization:
  - Only consider branch points spatially *nearby* target
  - Don't even look at others
  - Limit search to region near target



HF Routing Example



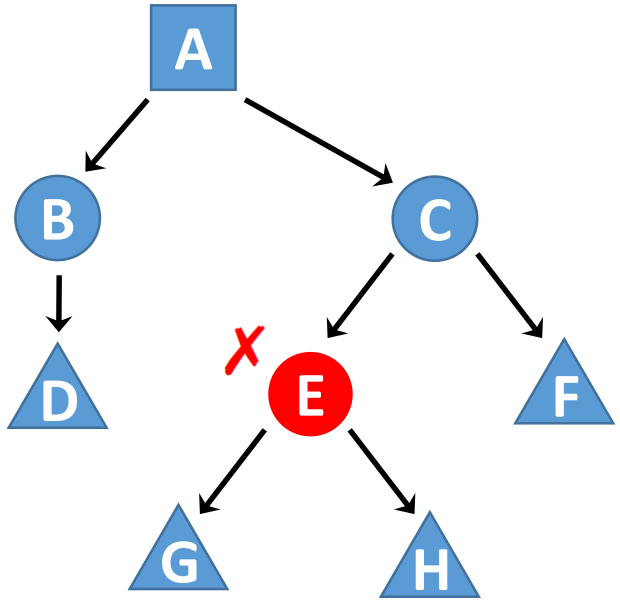
Traditional



Spatial Lookup

Incremental Routing

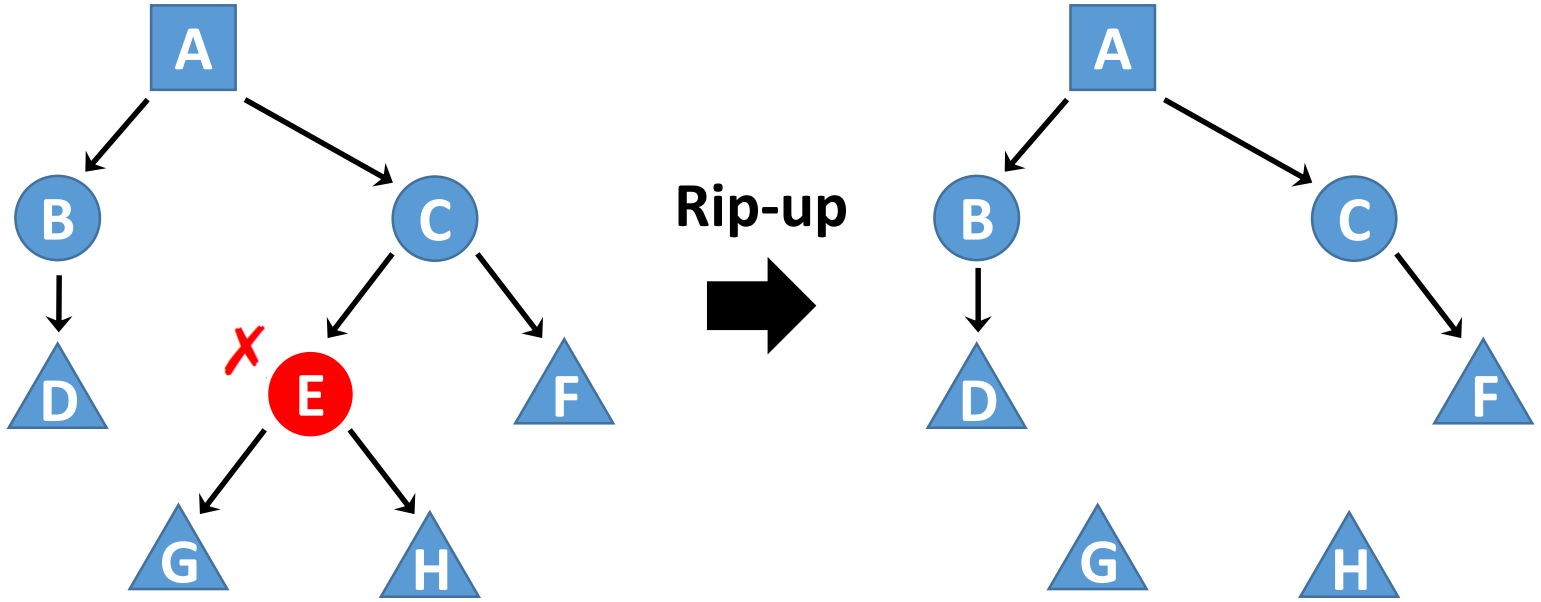
- Observation: Most net connections routed legally
- Optimization:
  - Rip-up illegal sub-trees
  - Re-route *only* those sub-tree connections



- Challenge: May degrade critical path
  - Fix: Also rip-up delay sub-optimal connections

Incremental Routing

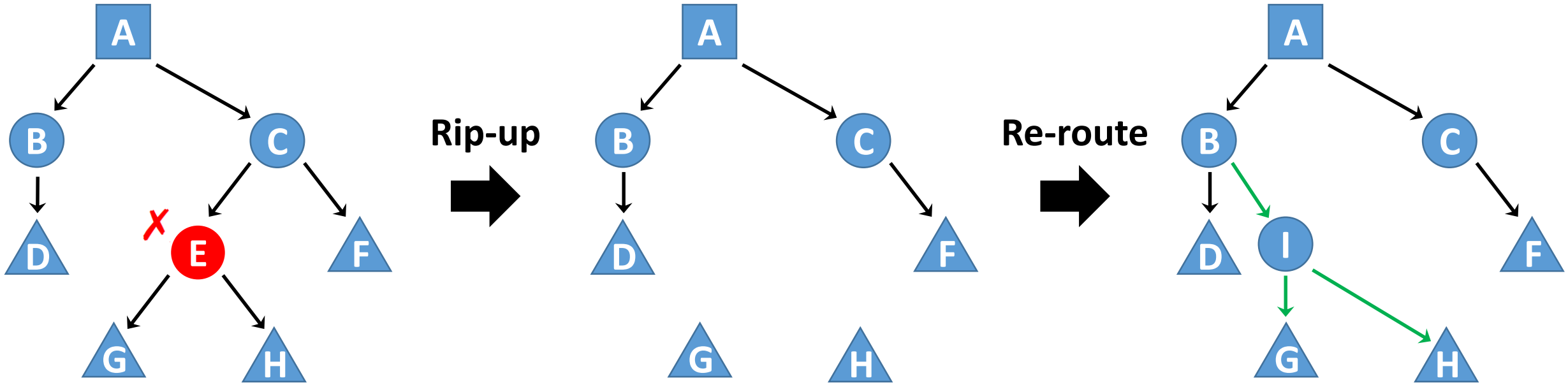
- Observation: Most net connections routed legally
- Optimization:
  - Rip-up illegal sub-trees
  - Re-route *only* those sub-tree connections



- Challenge: May degrade critical path
  - Fix: Also rip-up delay sub-optimal connections

Incremental Routing

- Observation: Most net connections routed legally
- Optimization:
  - Rip-up illegal sub-trees
  - Re-route *only* those sub-tree connections



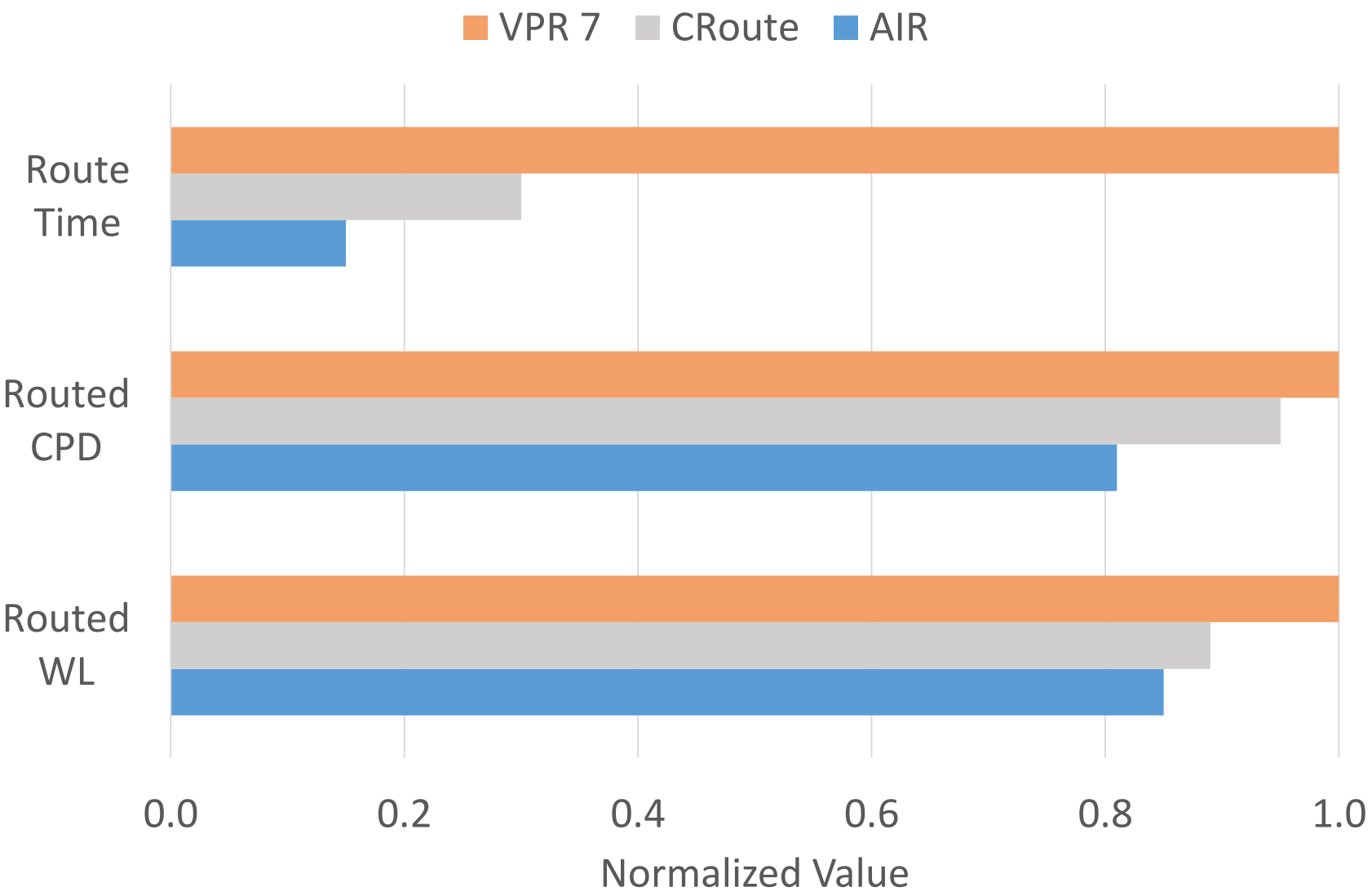
- Challenge: May degrade critical path
  - Fix: Also rip-up delay sub-optimal connections

Evaluation

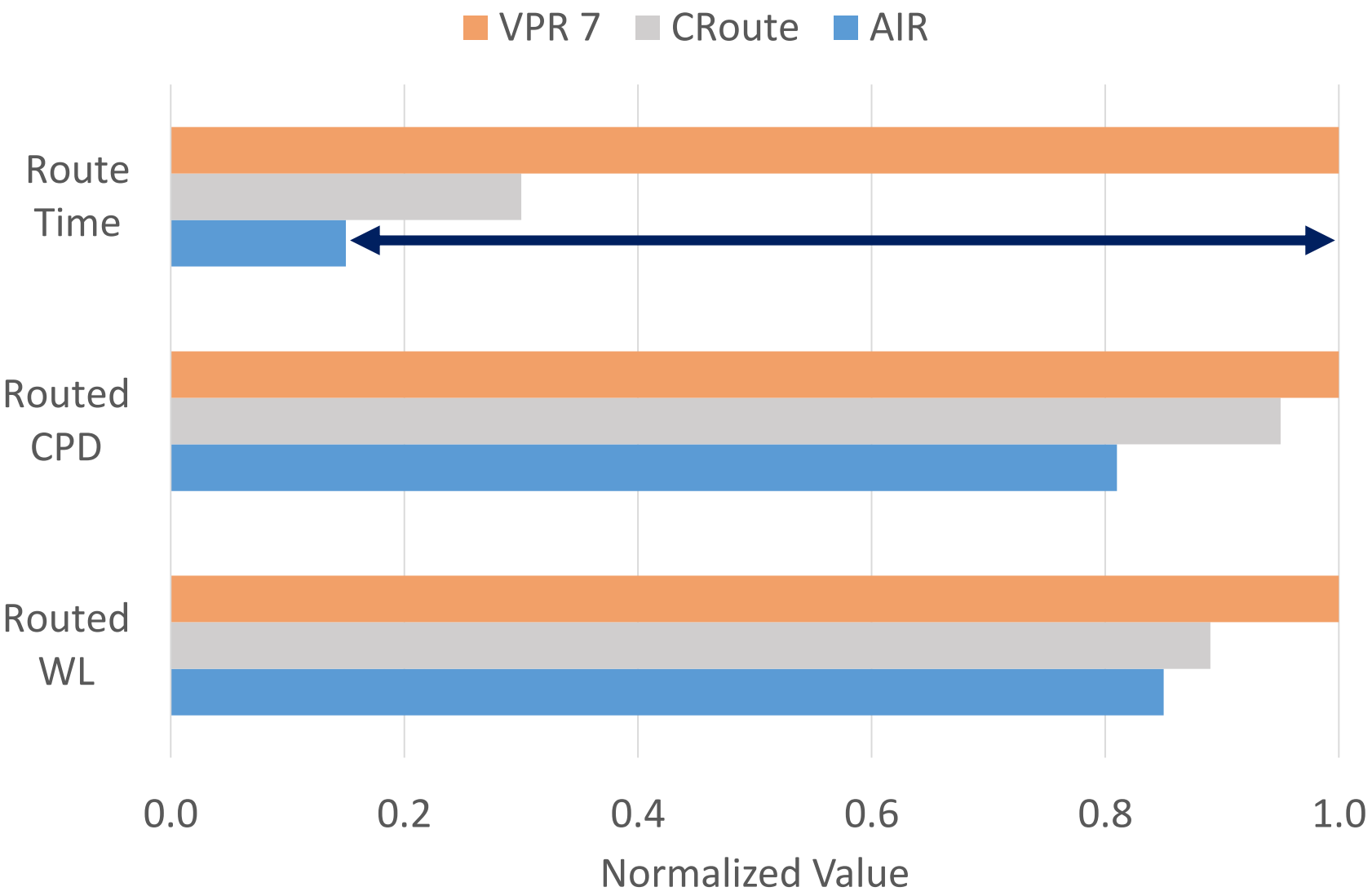
Experimental Setup

- Titan Benchmarks:
  - 23 modern heterogeneous FPGA designs
  - Large Scale: 90K-1.9M netlist primitives
- Targeting Stratix IV FPGA Architecture Model
- Academic Routers:
  - VPR 7
  - CRoute
  - AIR
- Commercial Routers:
  - Intel Quartus 18.0

Academic Router Comparison



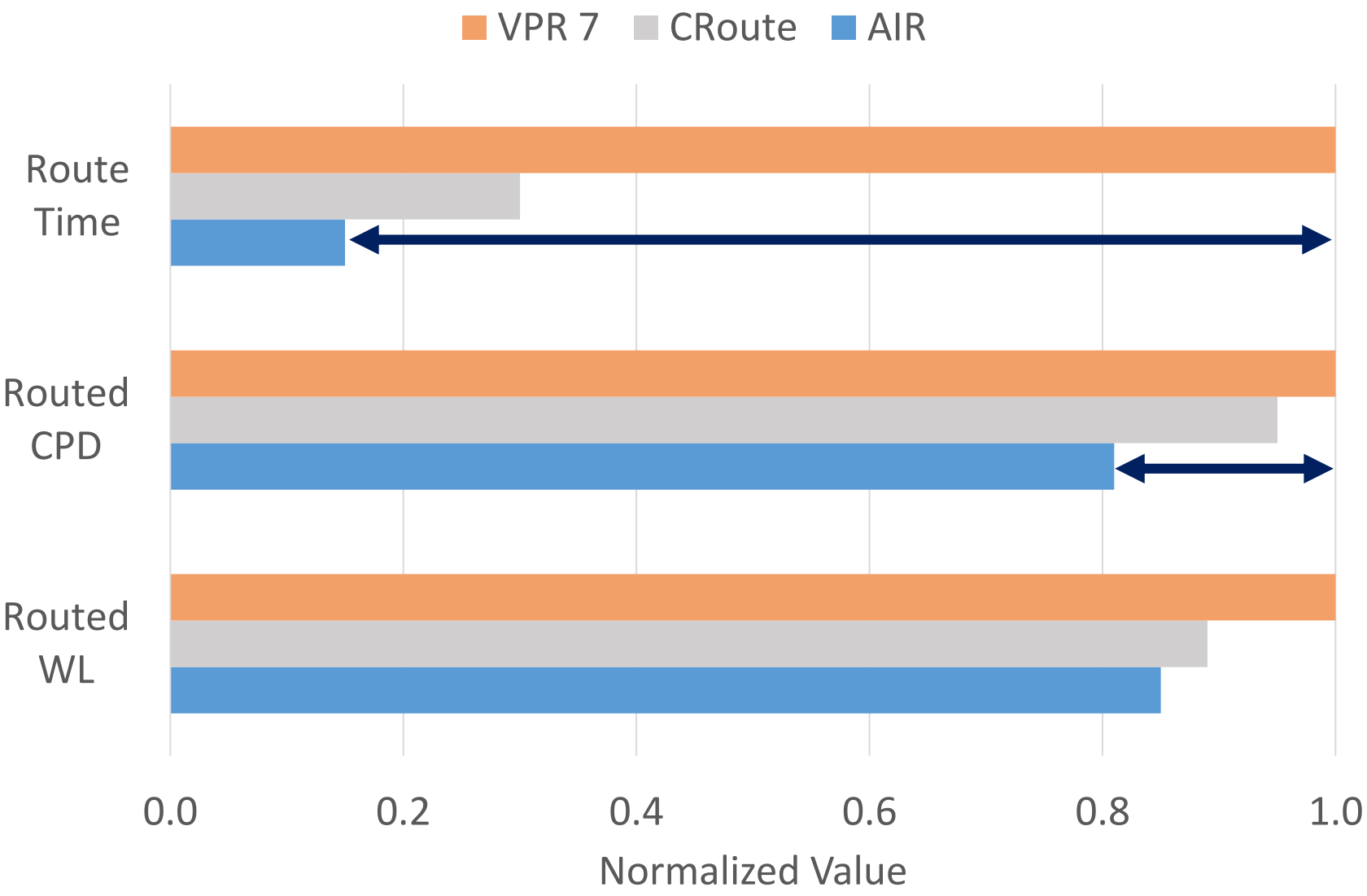
Academic Router Comparison



> 6.6x run-time reduction

- Incr. Routing
- HF Routing
- Lookahead

Academic Router Comparison



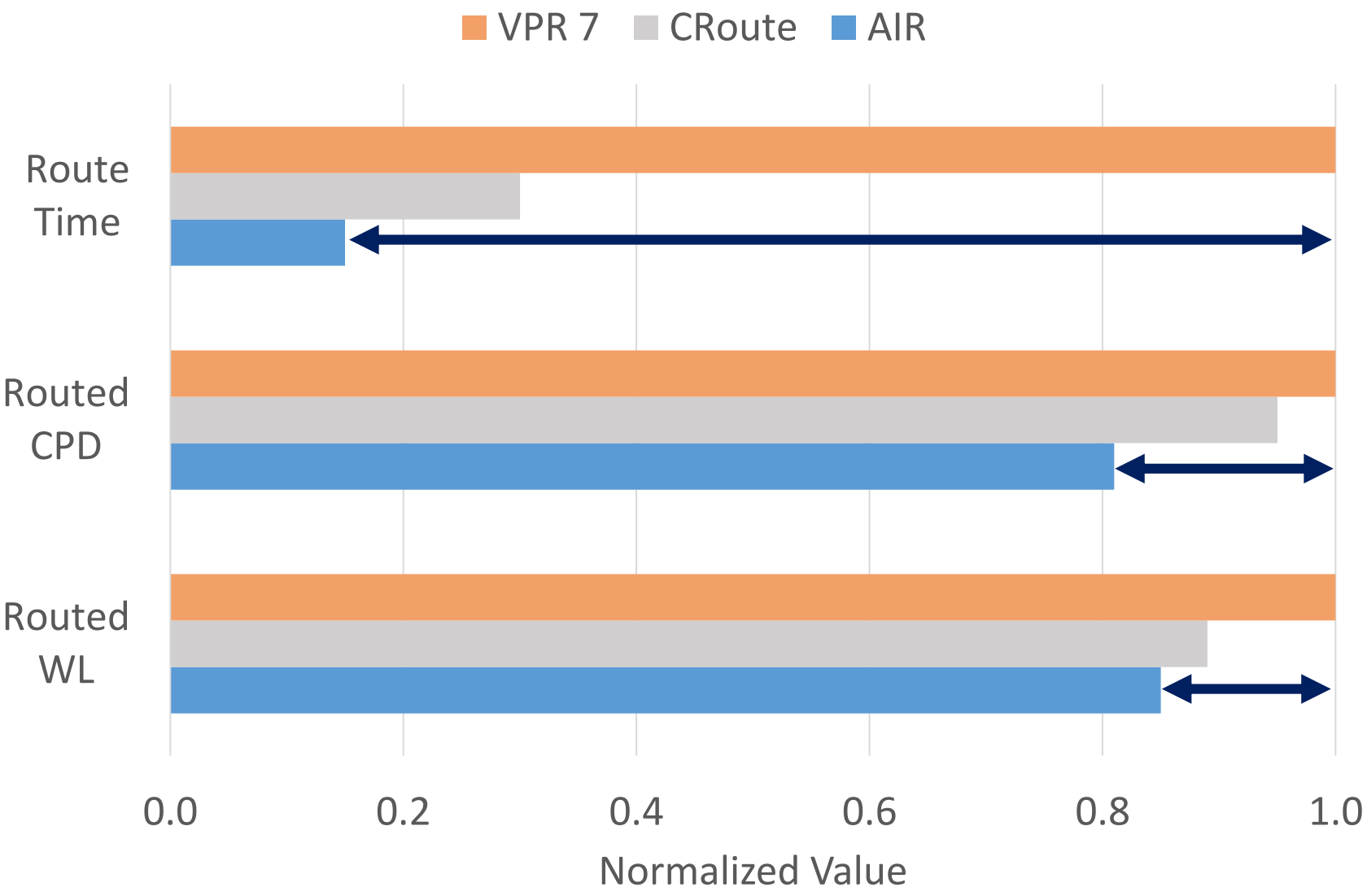
> 6.6x run-time reduction

- Incr. Routing
- HF Routing
- Lookahead

19% faster critical path

- Lookahead

Academic Router Comparison



> 6.6x run-time reduction

- Incr. Routing
- HF Routing
- Lookahead

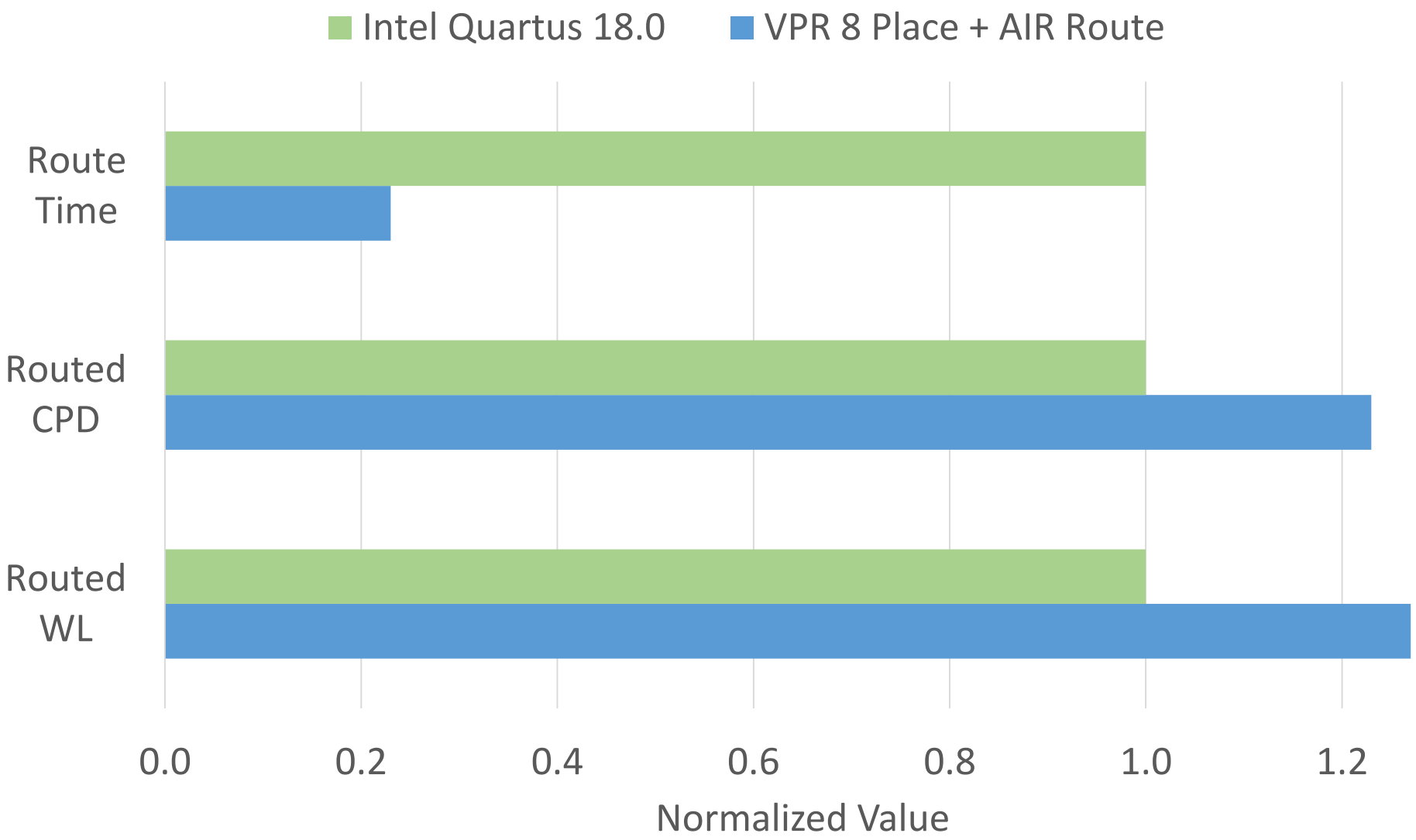
19% faster critical path

- Lookahead

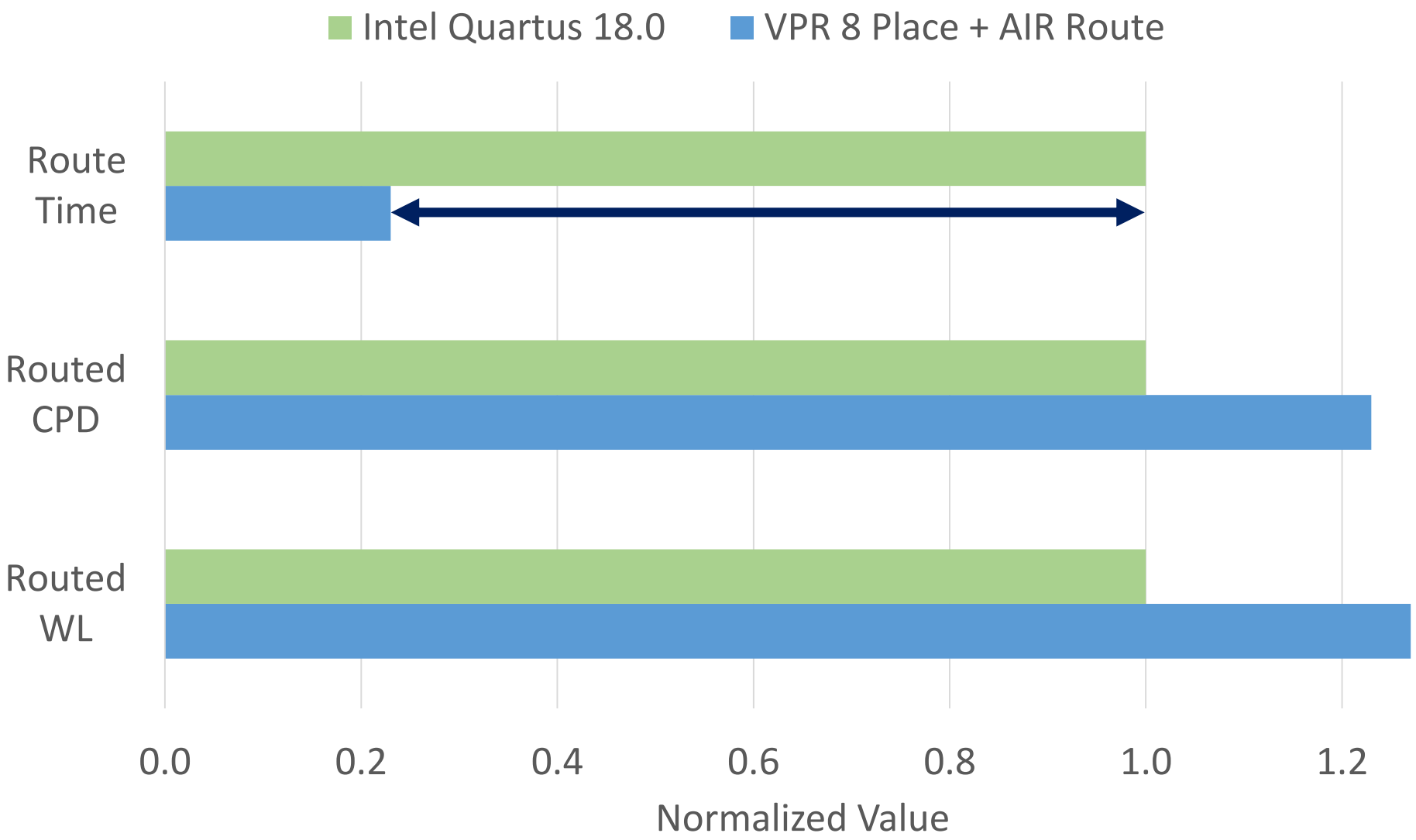
15% less wiring

- Resource Base Costs
- Lookahead

Commercial Router Run-time Comparison

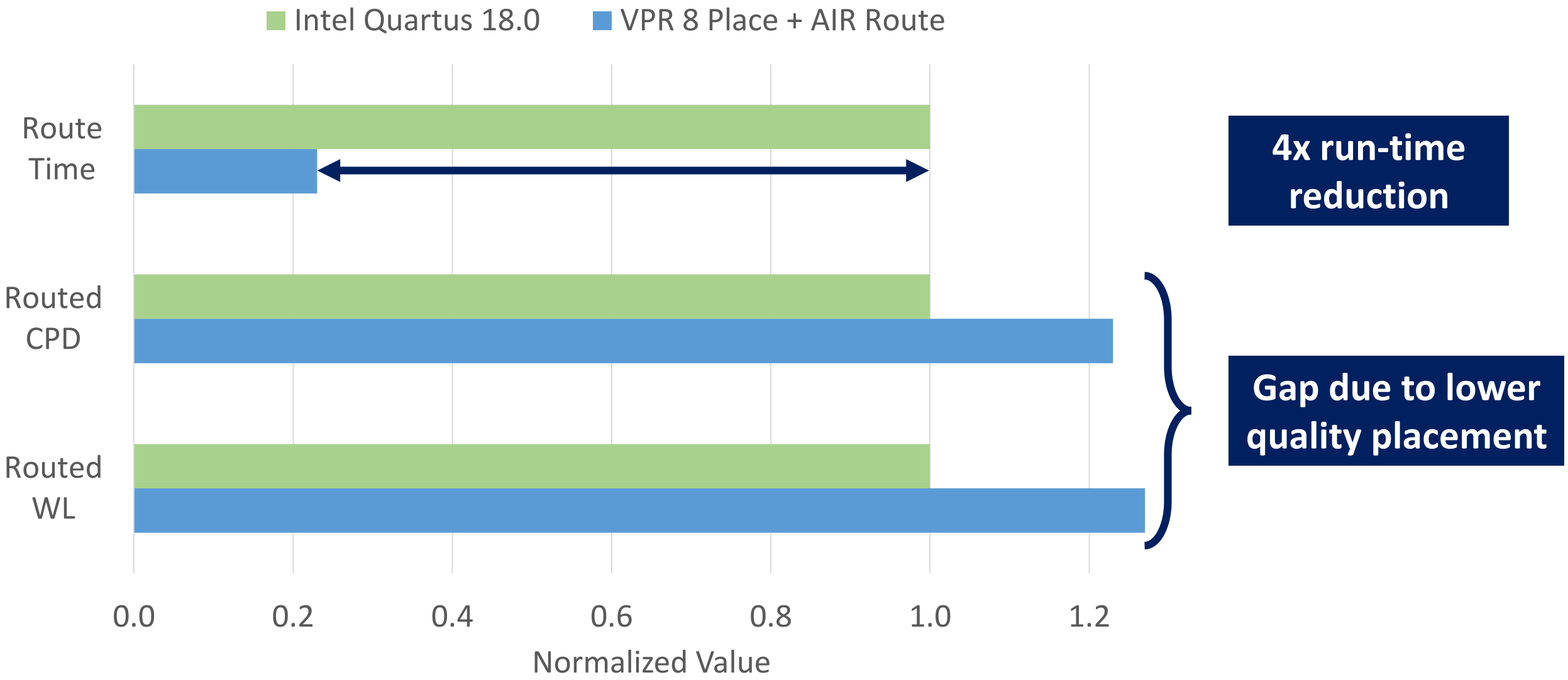


Commercial Router Run-time Comparison

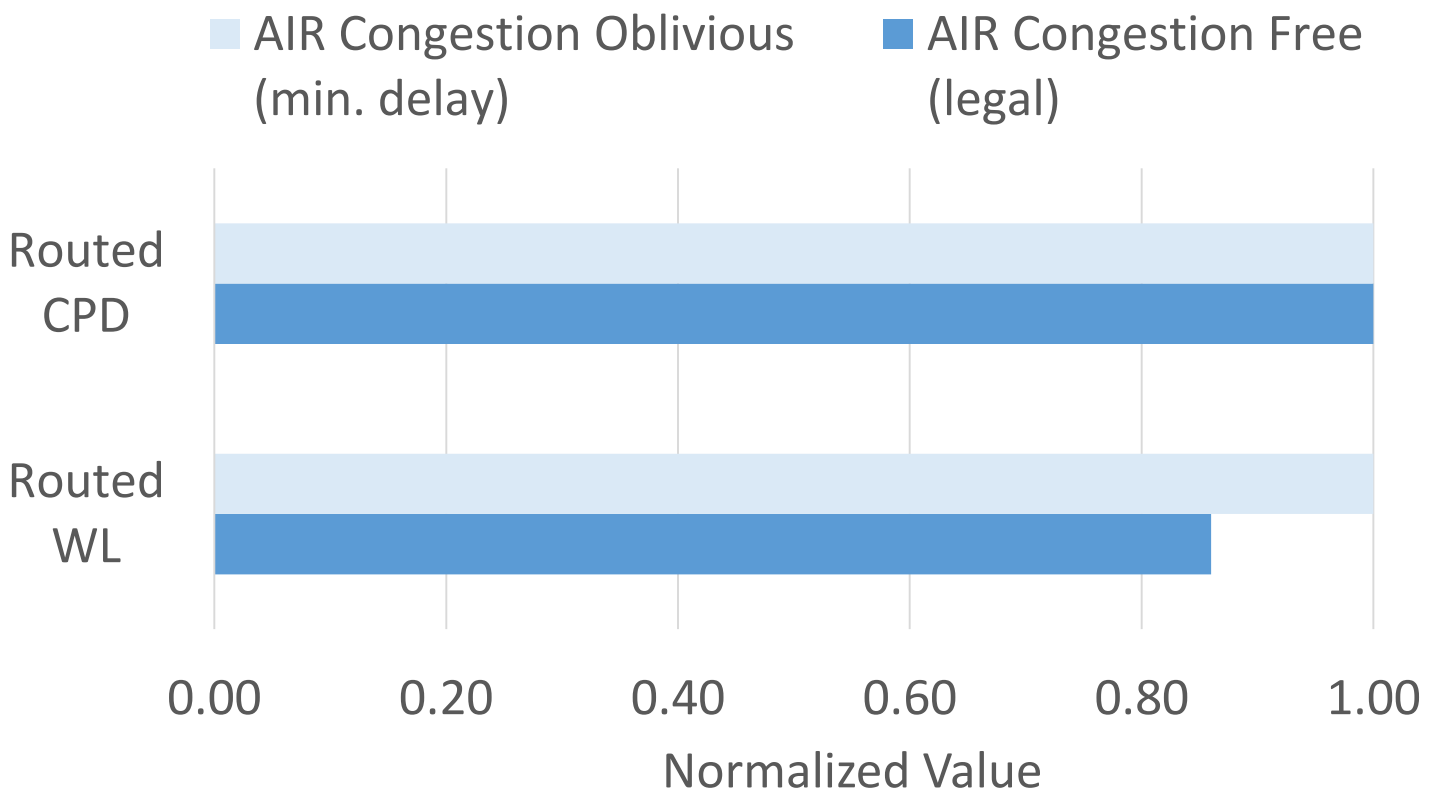


4x run-time reduction

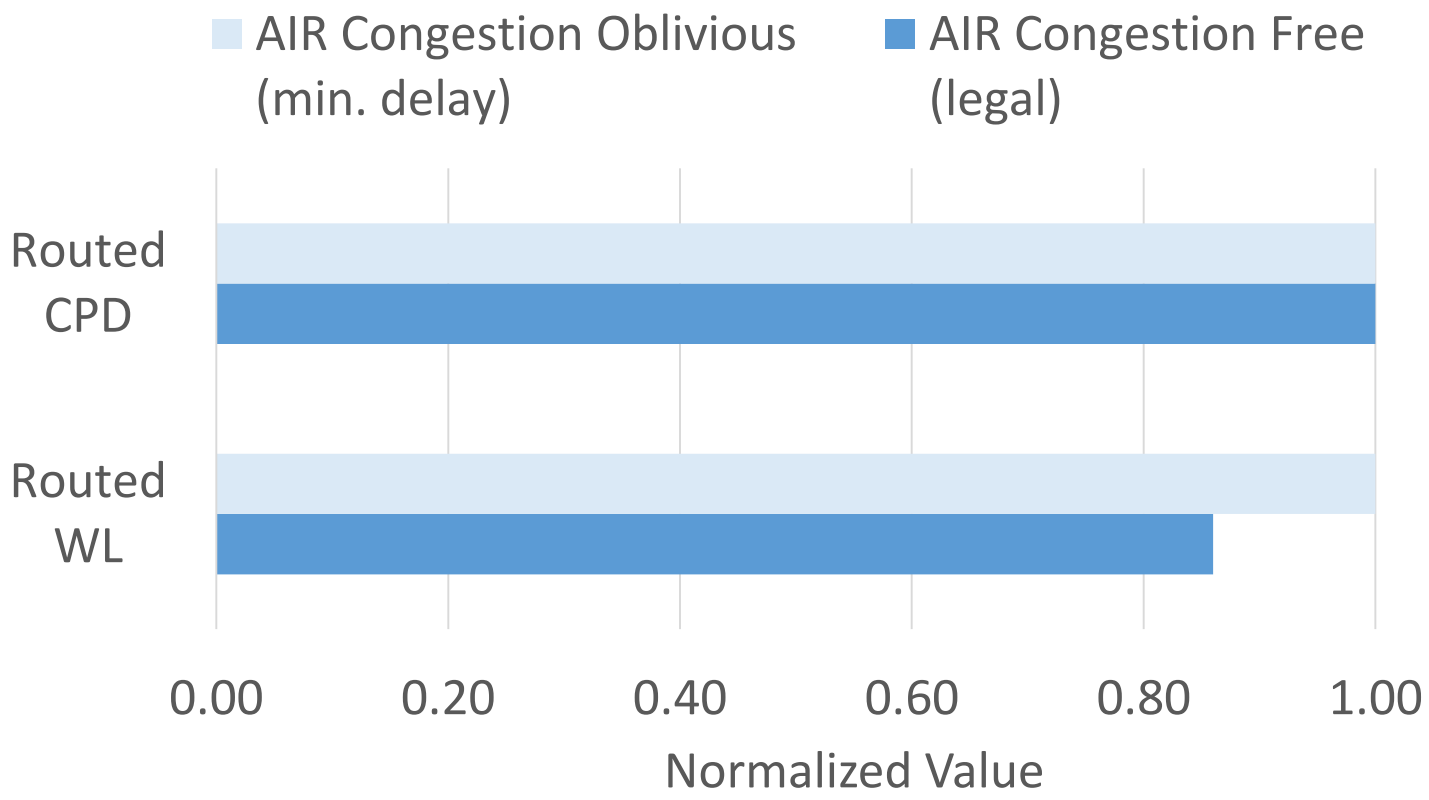
Commercial Router Run-time Comparison



AIR Congestion Resolution

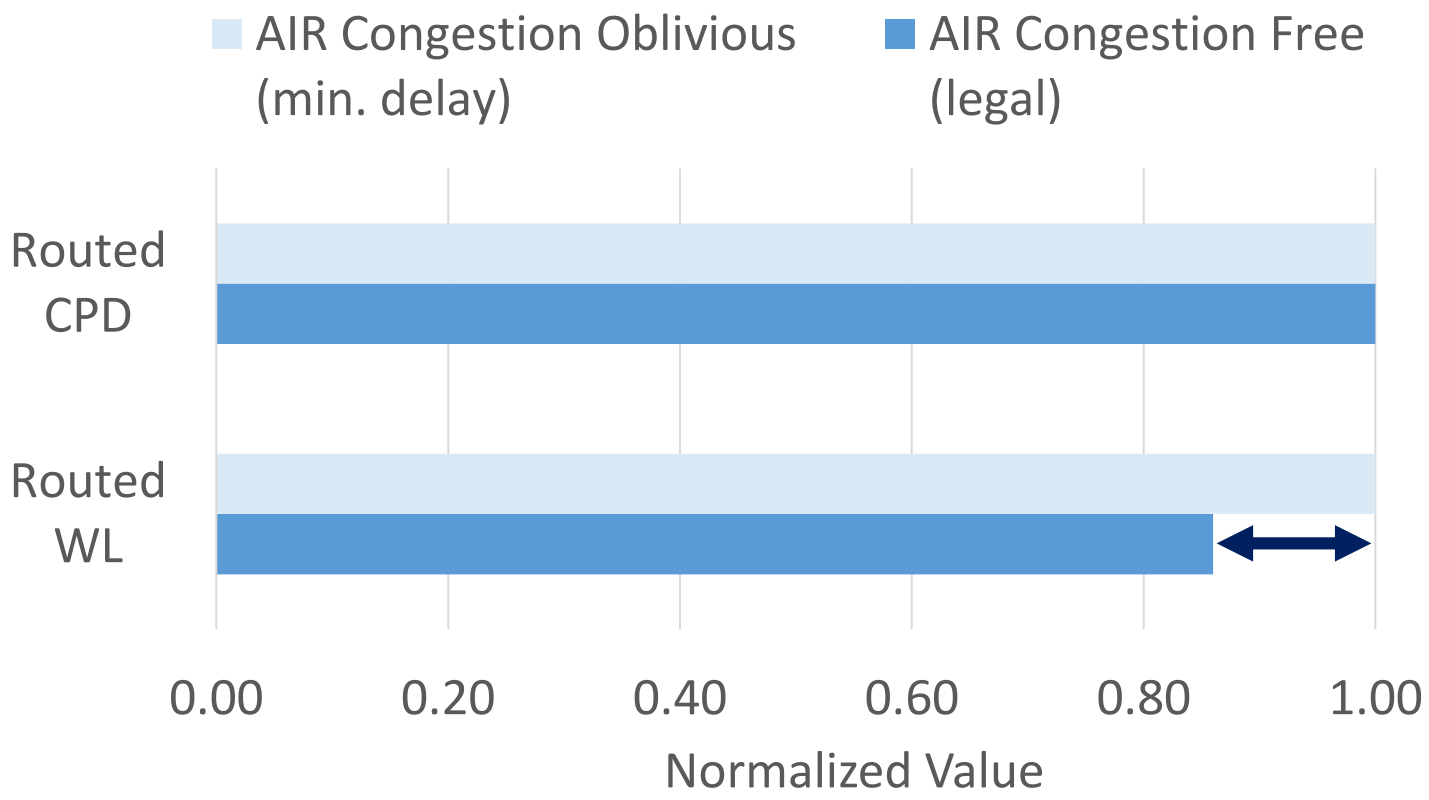


AIR Congestion Resolution



Critical Path not degraded

AIR Congestion Resolution



Critical Path not degraded

Wiring reduced

Conclusion

Conclusion

- AIR Timing-Driven FPGA Router
  - Lazy: Incremental Routing, High-Fanout
  - Adaptive: Architecture (lookahead and base costs), Congestion
- QoR:
  - vs VPR 7: > 6.6x run-time reduction, 19% faster CPD, 15% less wiring
  - vs Intel's Quartus 18.0: 4x run-time reduction
  - Effectively trades-off resource usage to resolve congestion
- Other enhancements (see paper):
  - Congestion adaptation
  - Re-convergent routing

- AIR is the new default router in VPR 8

Thanks!

Questions?

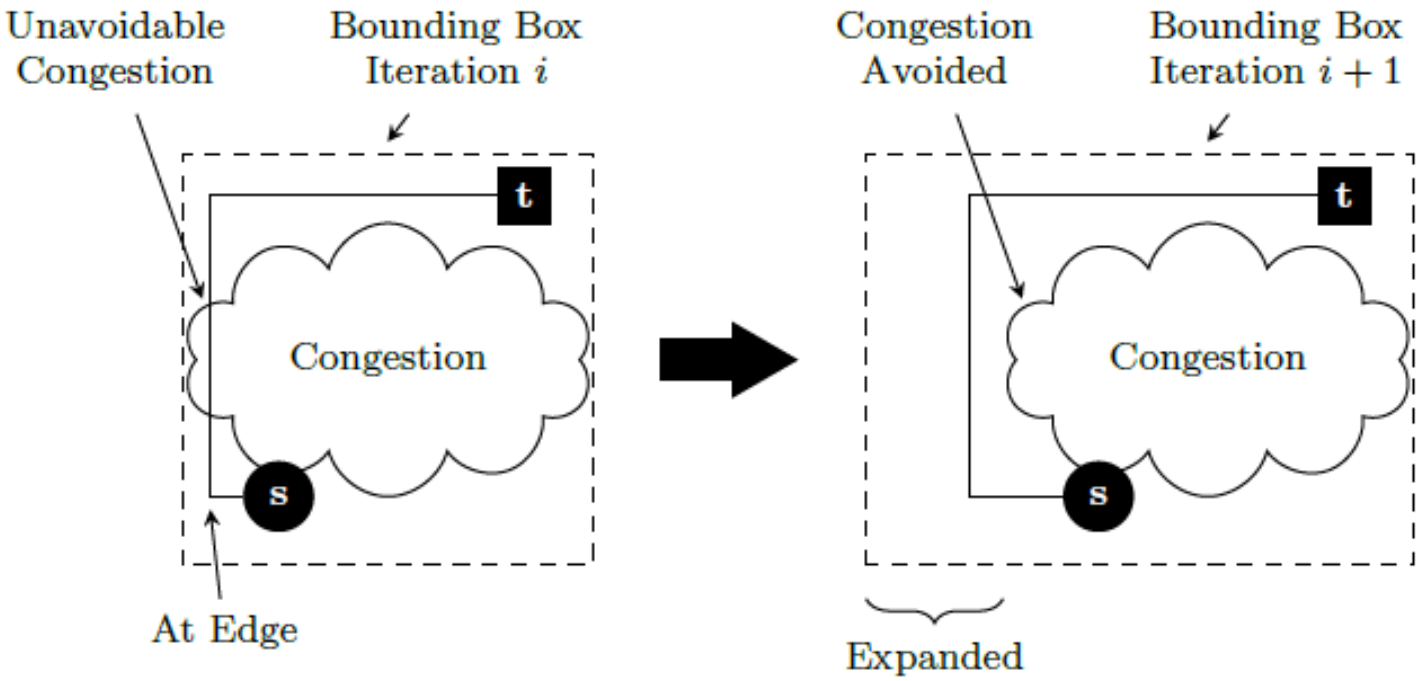
Email: kmurray@ece.utoronto.ca

AIR Source Code: verilogtorouting.org

Appendix

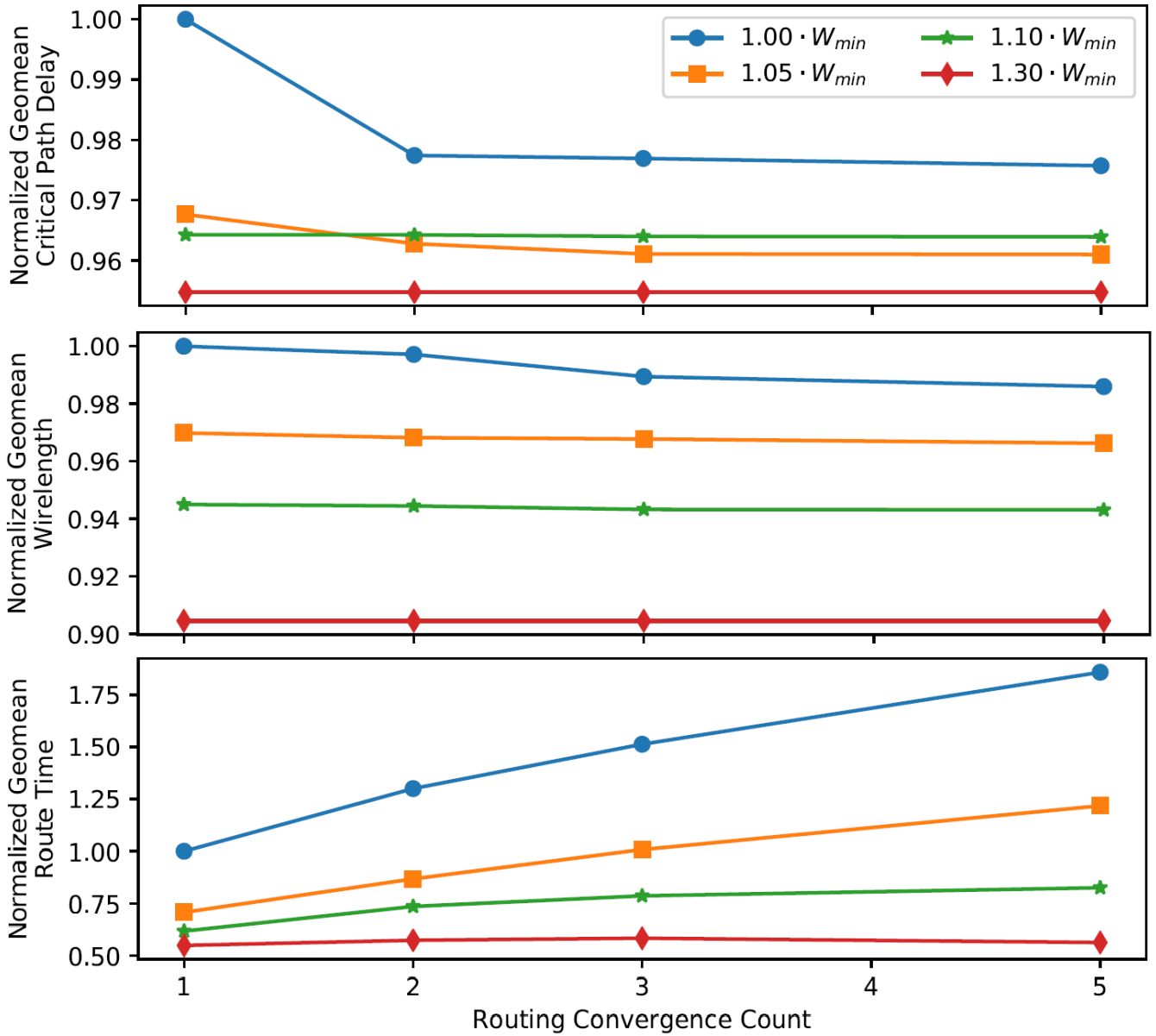
Congestion Adaptation

- Observation: Bounding Box (BB) limits how far a connection could detour
- Optimization: Expand BB when using resources at edge of BB
  - Allows connections to detour away from congestion

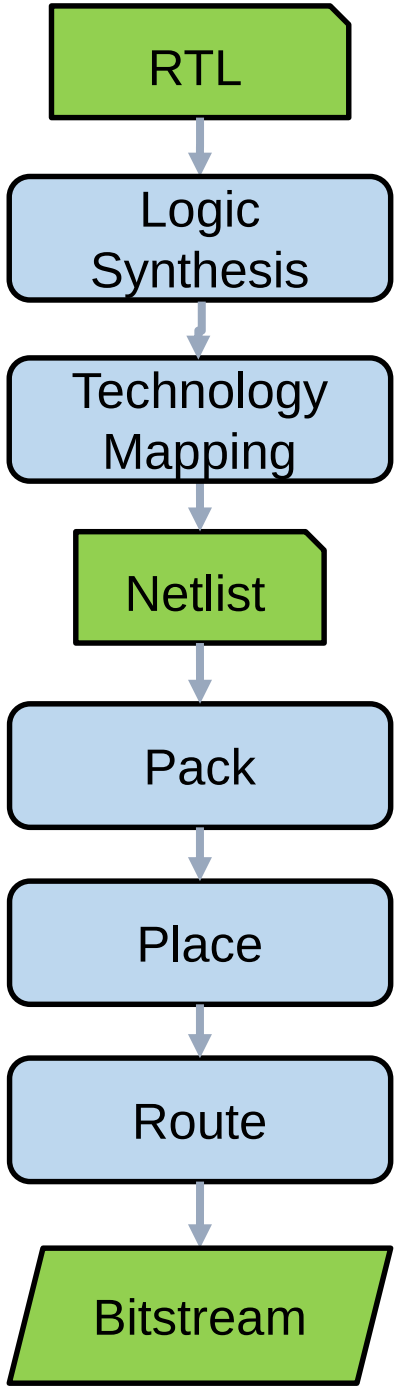


Reconvergent Routing

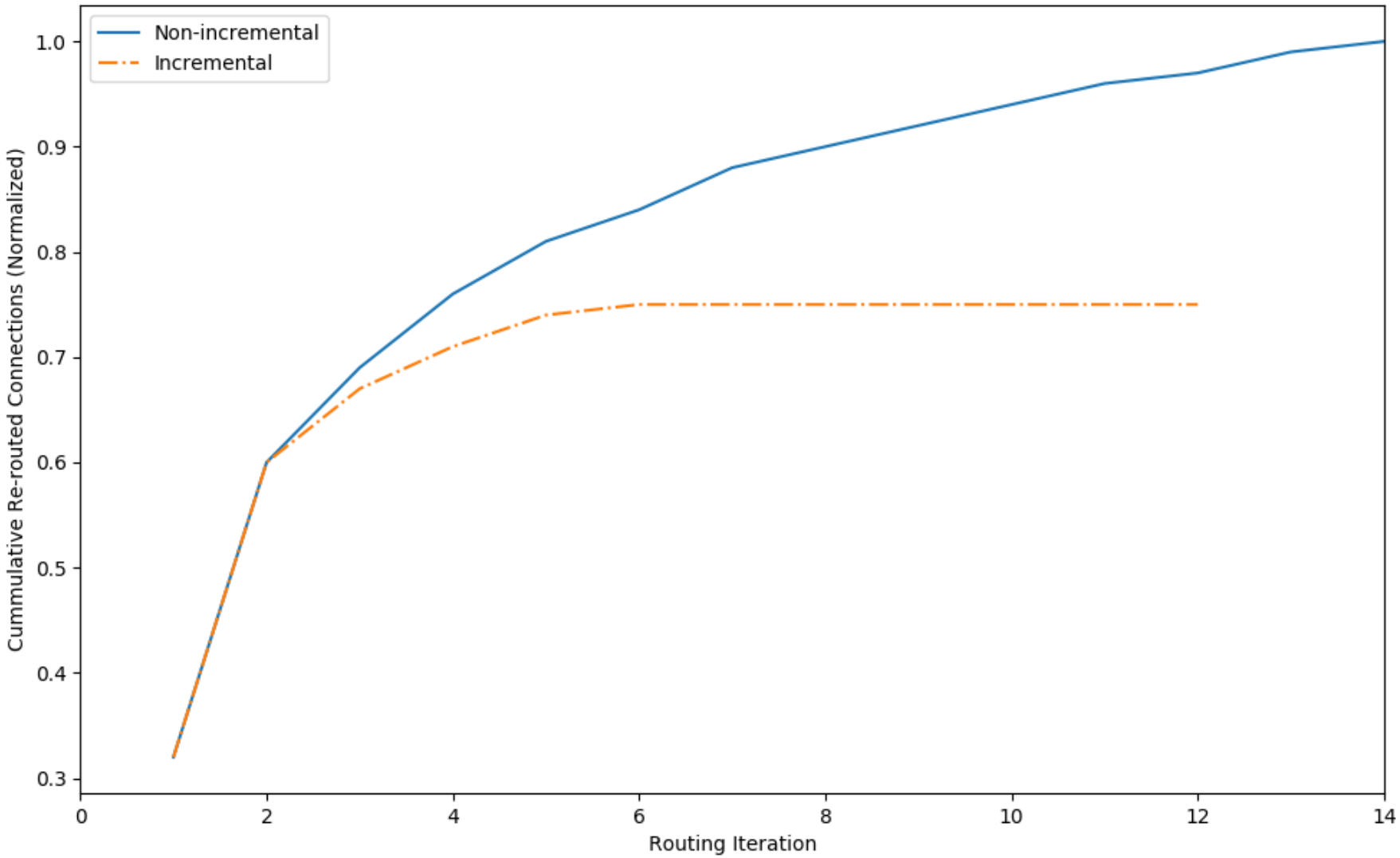
- Observation: In highly congested designs critical path may be detoured
  - Degrades timing
- Optimization: Continue re-routing delay sub-optimal connections after initial convergence
  - Also scale back congestion costs to allow critical path
- Improves CPD & WL
- Incremental routing substantially reduces run-time



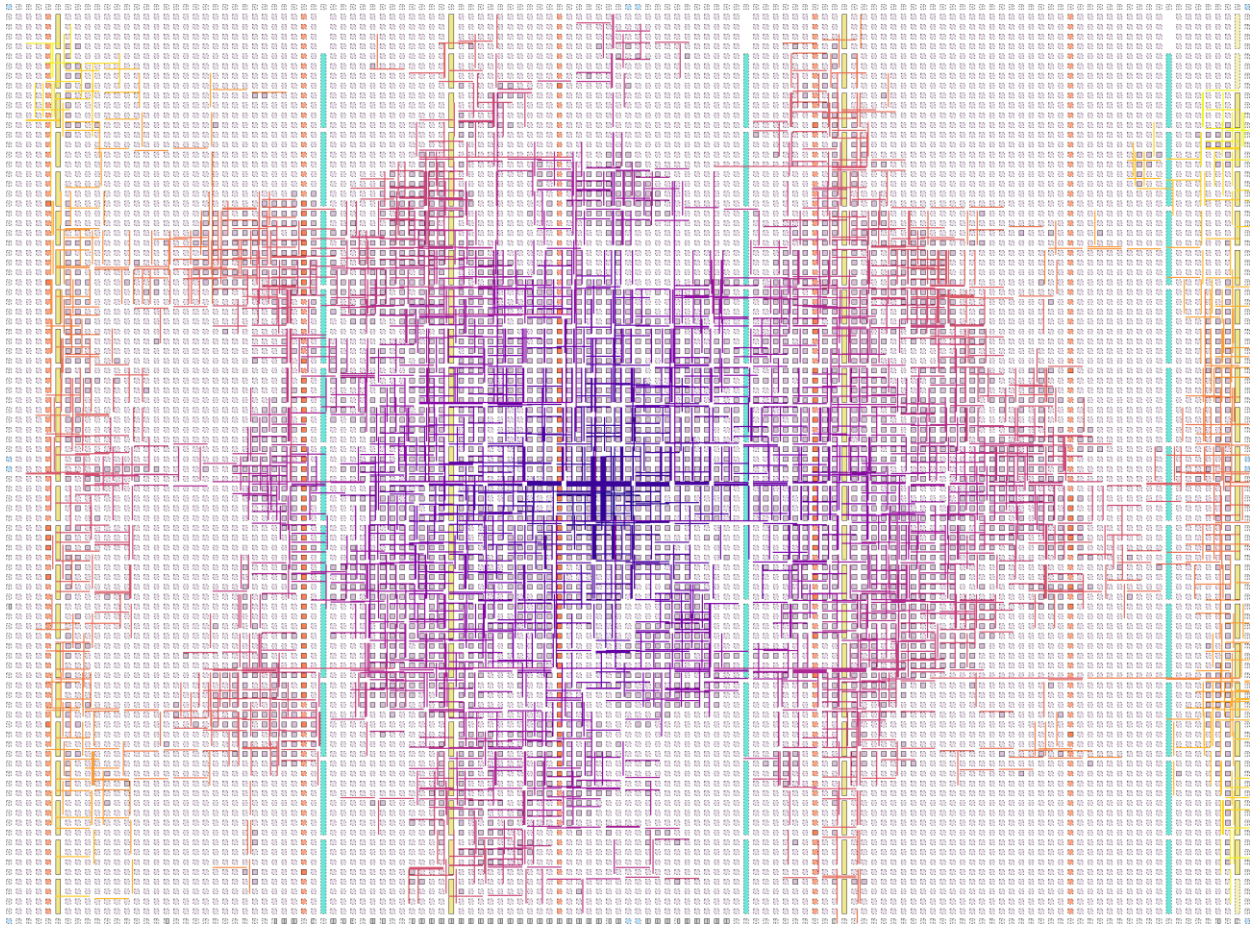
FPGA CAD Flow



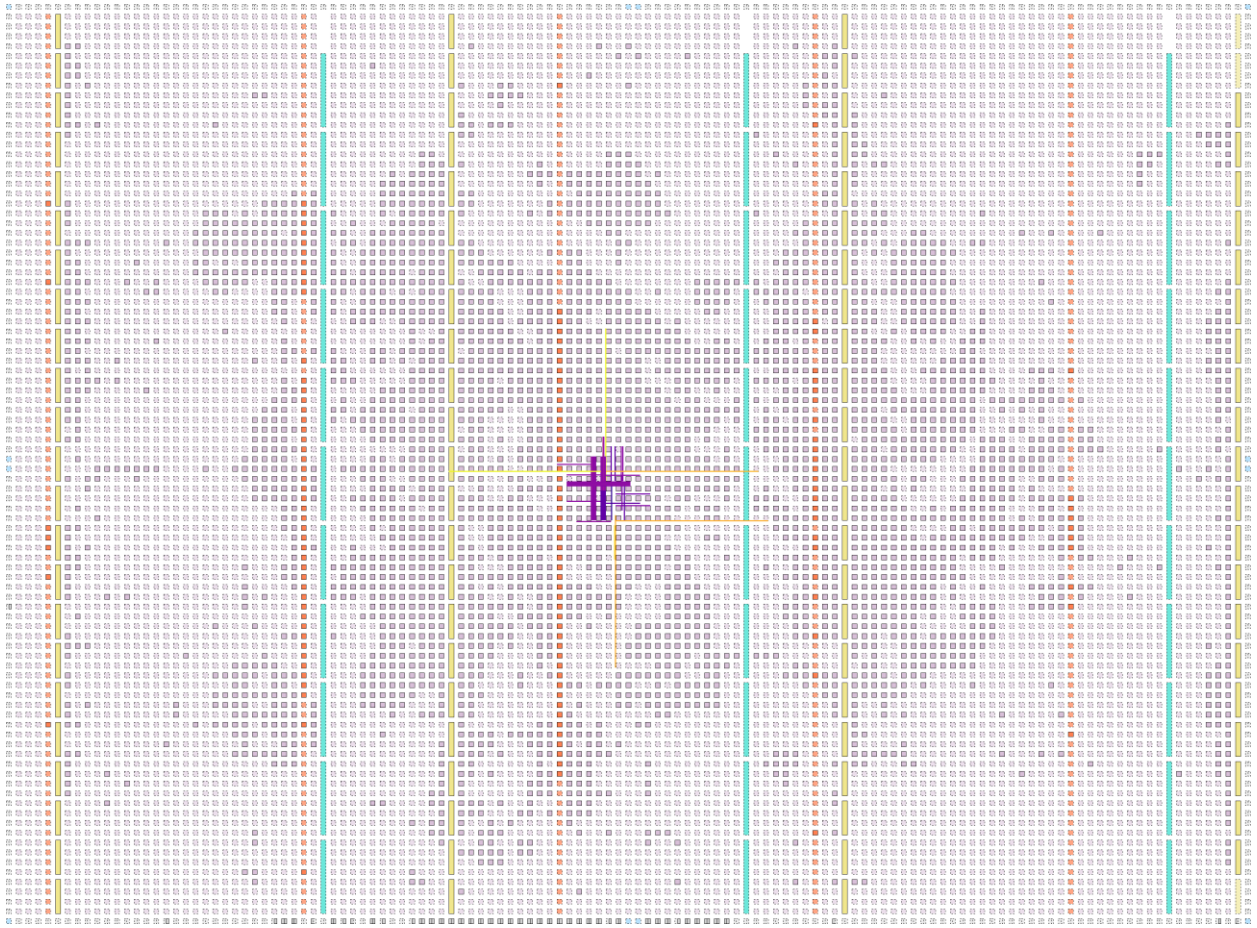
Impact of Incremental Routing on Neuron



HF Net Router Expansion on Neuron



Traditional



Spatial Lookup