Verilog to Routing (VTR):
A Flexible Open-Source CAD Flow to Explore and Target Diverse FPGA Architectures

Vaughn Betz

With thanks to all VTR contributors, and particularly Kevin Murray for assistance in creating this presentation

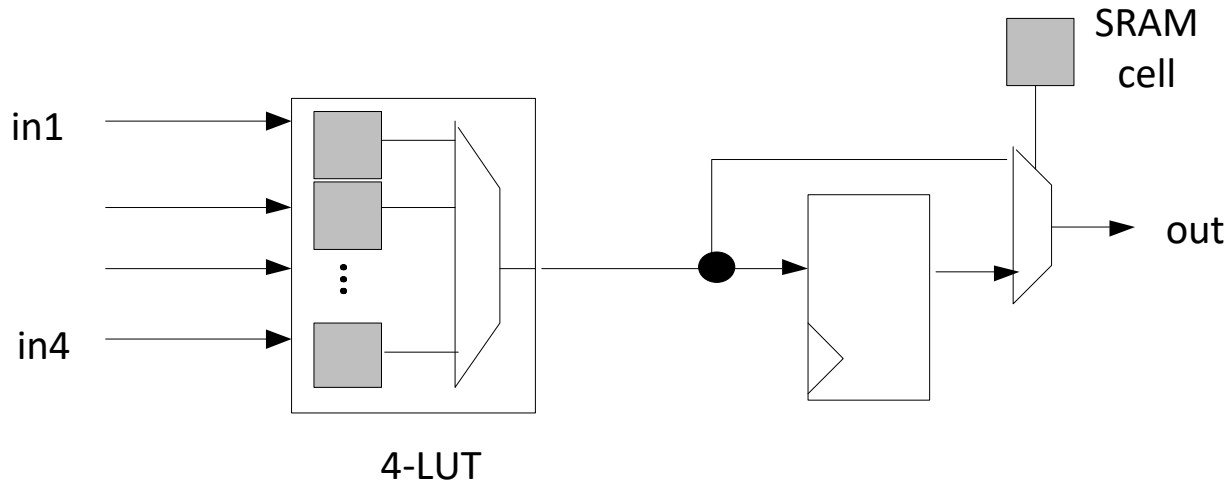
Agenda

- FPGAs: Basics & Utility
- Flexible CAD: Motivation and Challenges
- Verilog to Routing: Recent Progress
  - Architecture Generality
  - AIR: Faster, More Adaptive Routing
  - VTR 8.0: QoR Improvements
  - Reinforcement Learning for Adaptive Placement
- Summary

FPGA Basics & Utility

FPGA Basics

Programmable Hardware



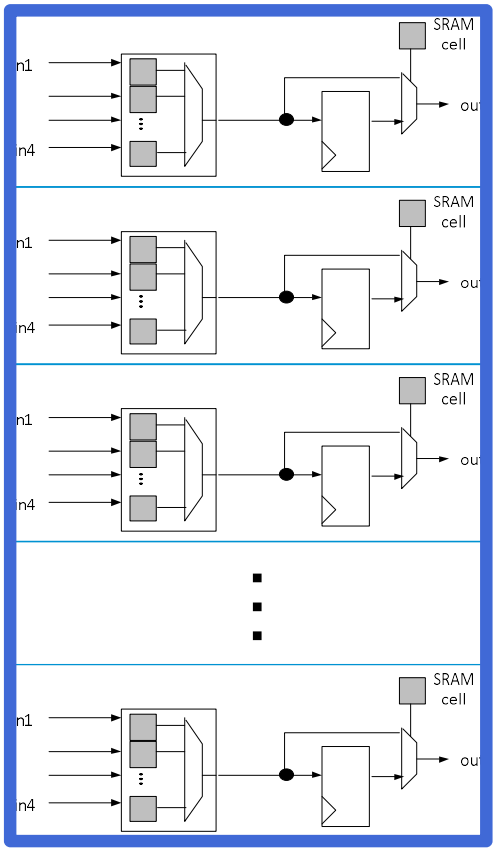
Can implement arbitrary circuits

- Look-Up Table (LUT) implements logic
- Flip-Flop (FF) stores sequential state

FPGA Basics

Logic Block (LB)

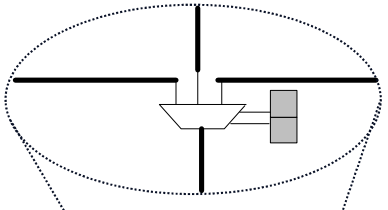
- Group of Logic Elements (LEs)



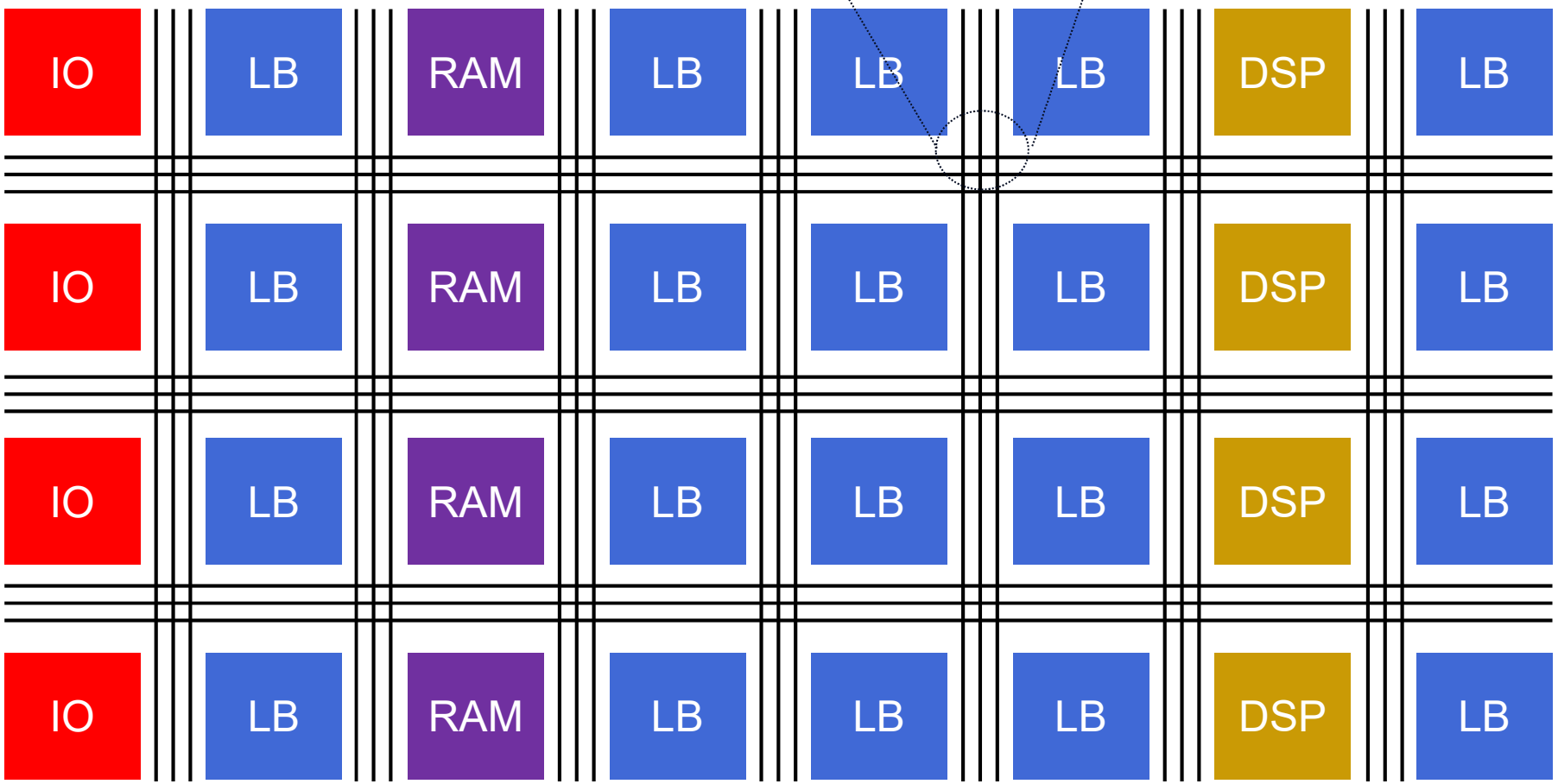
FPGA Basics

Device

- Grid of Function Blocks
- Interconnect



Routing Switch

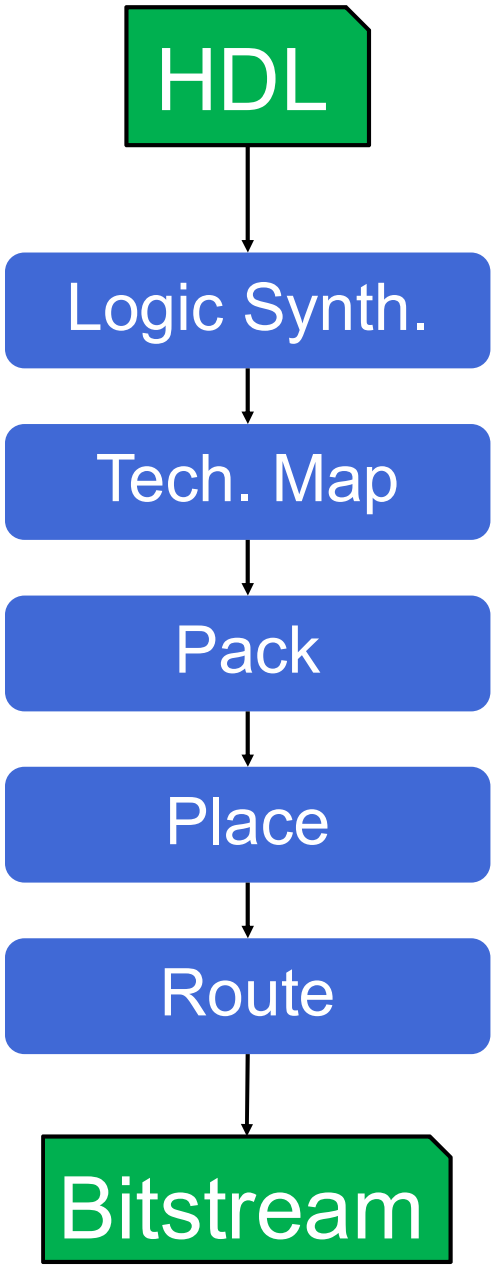


FPGA Basics

Mapping Applications to an FPGA

- Modern FPGAs are large (>10M LEs)

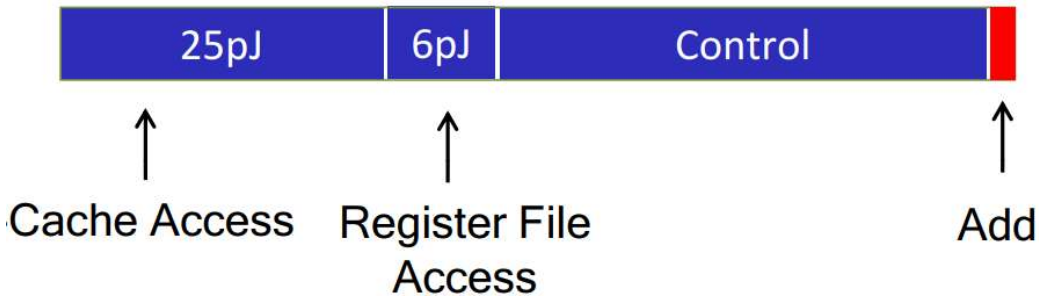
- Use Computer Aided Design (CAD) flow!
  - *Designer: provides behavioural specification*
  - *Algorithms & tools: figure out the details*



Why Use an FPGA?

1. Flexible I/O
   - No off-the-shelf chip with right I/O
2. Power Efficiency
   - CPU/DSP/GPU performance per watt too low

45 nm CPU energy breakdown [1]



0.03 pJ for 8-bit int add
0.90 pJ for 32-bit fp add

3. Programmability
   - Dedicated ASIC not practical (e.g. changing workload)

[1] M. Horowitz, "Computing's Energy Problem," *ISSCC,* 2014.

Widely Used Across Many Domains

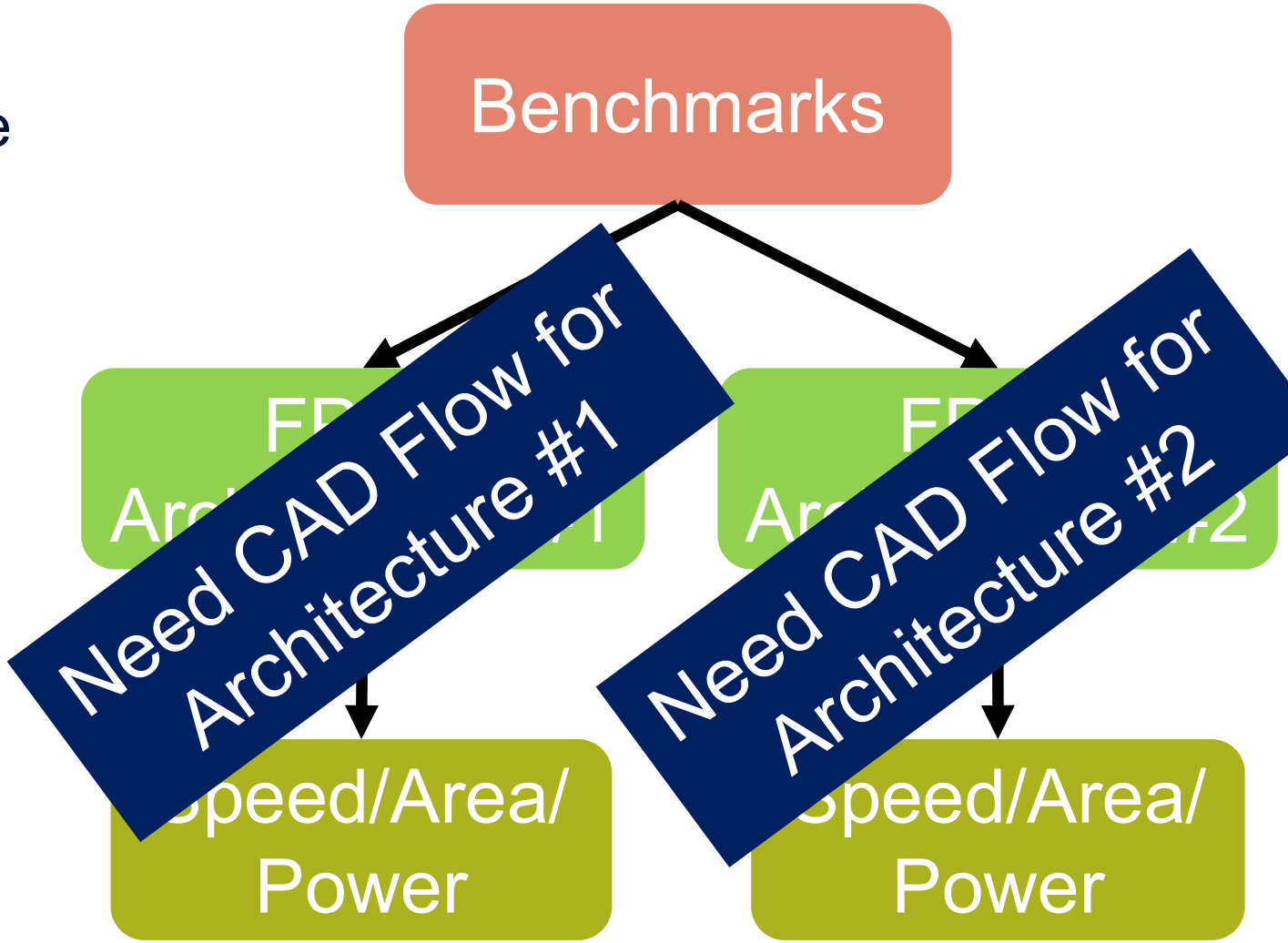
CAD flow key to success

Flexible, Open Source CAD: Motivation and Challenges

I: FPGA Architecture Research

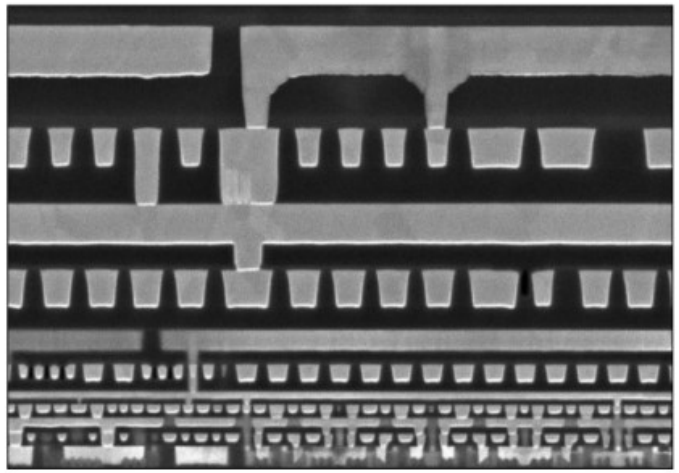
Challenges:

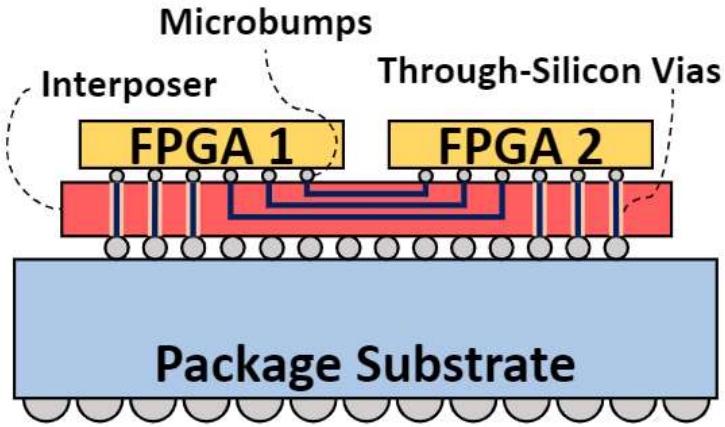
- Need representative benchmarks
- Describe FPGA Architecture
- **Need high quality CAD flow**



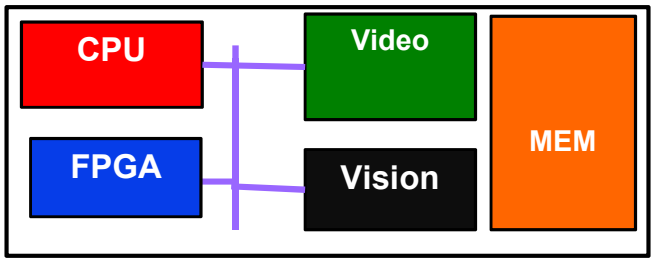
Why New FPGA Architectures?

1. Process Technology Changes
   - Wire resistance, power limits, interposers, ..





2. Application Changes[2]
   - New standards, faster processing: 4G → 5G, …
   - Deep Learning → new compute demands
   - Embedded FPGAs



3. Innovative Architecture Ideas!

II: FPGA CAD Research

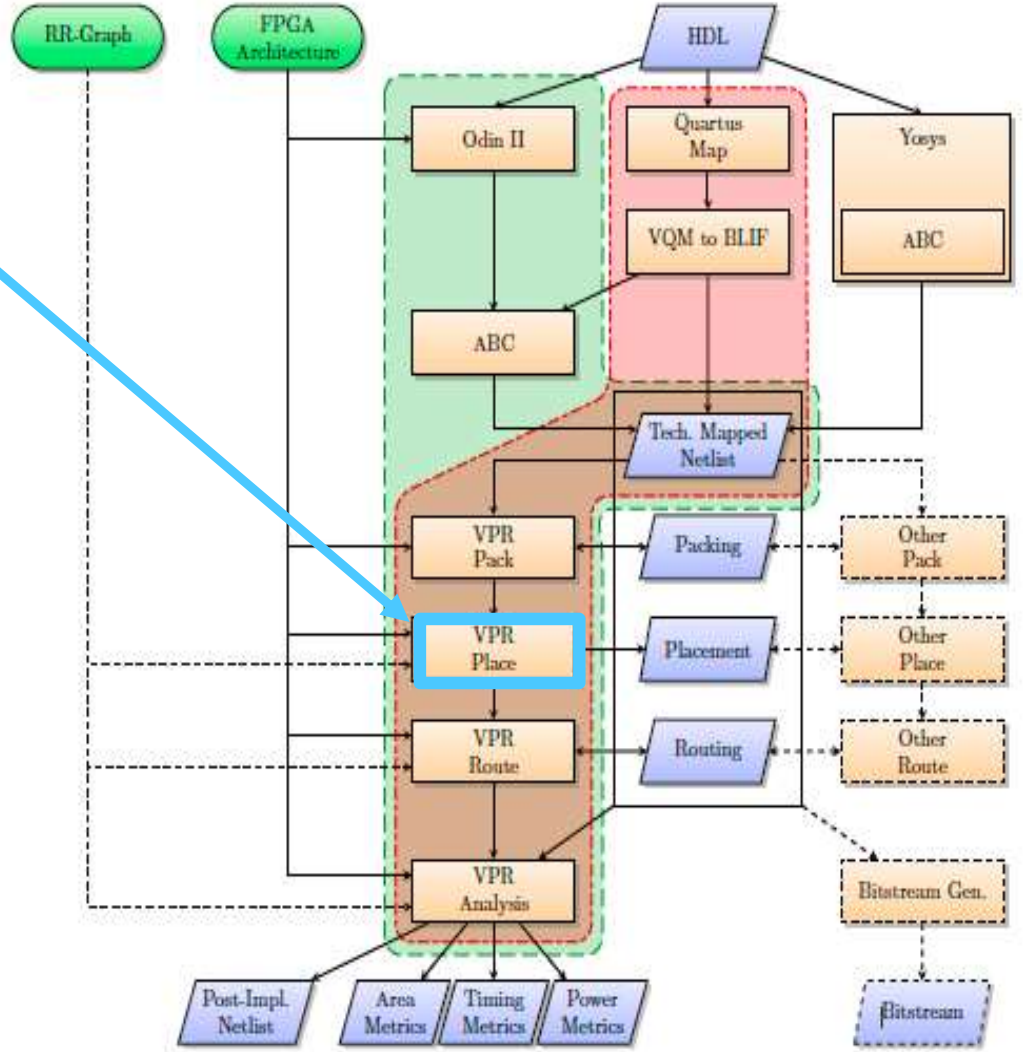
New idea for CAD algorithm

Need infrastructure!

- Do *not* build it all
- Focus on your contribution

Common framework:

- Best practise
- Compare results

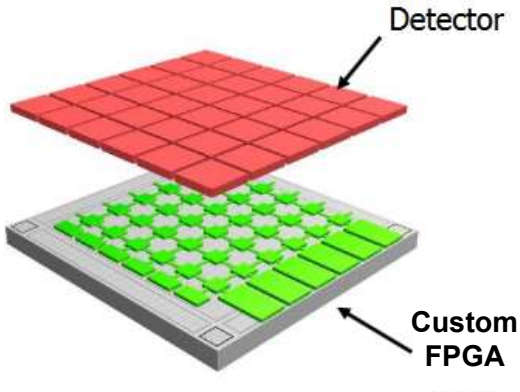


Flexible CAD: more impact

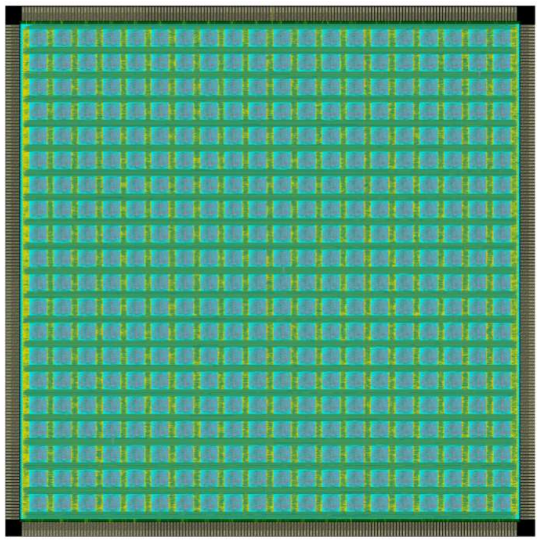
- VTR enhancements help Symbiflow (QuickLogic EOS S3, Xilinx Artix7), start-ups, architecture research, …

III: Program Novel Spatial Compute Architectures

- Have an architecture? Build it!
  - FPGA/FPGA+???/???
  - Need a CAD flow!



ReImagine: FPGA + Imager
(MIT Lincoln Lab) [1]



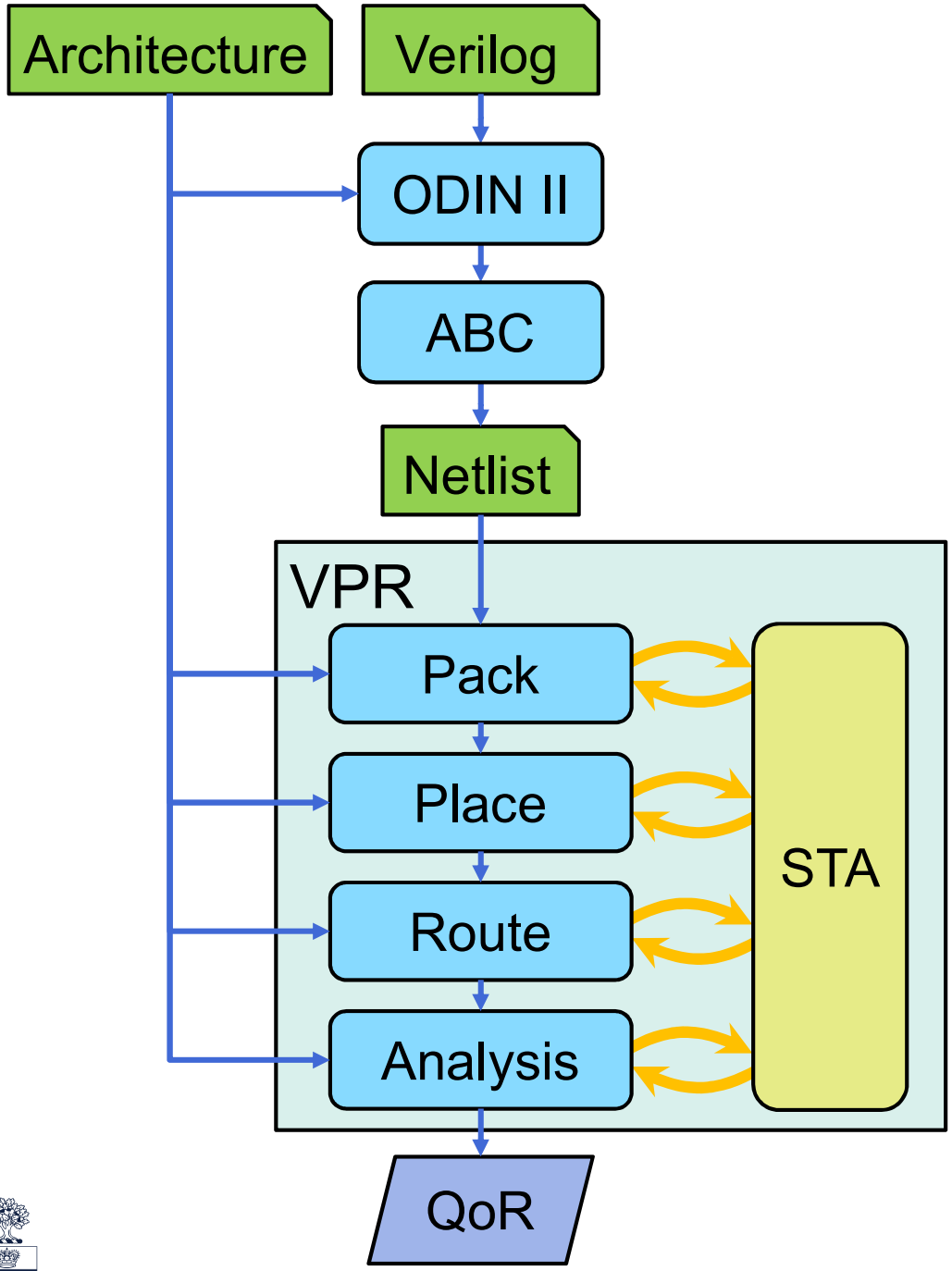
OpenFPGA (U of Utah) [2]

PRGA (Princeton) [3]

[1] https://www.darpa.mil/attachments/Final_Compiled_ReImagineProposersDay.pdf
[2] OpenFPGA: a Complete Open Source Frameworkfor FPGA Prototyping, Open Source Design Automation, 2019 (https://sites.google.com/site/pegaillardon/research/openfpga)
[3] PRGA: An Open-source Framework for Buildingand Using Custom FPGAs, Open Source Design Automation, 2019 (https://prga.readthedocs.io)

VTR: Verilog to Routing

Verilog to Routing (VTR) Project



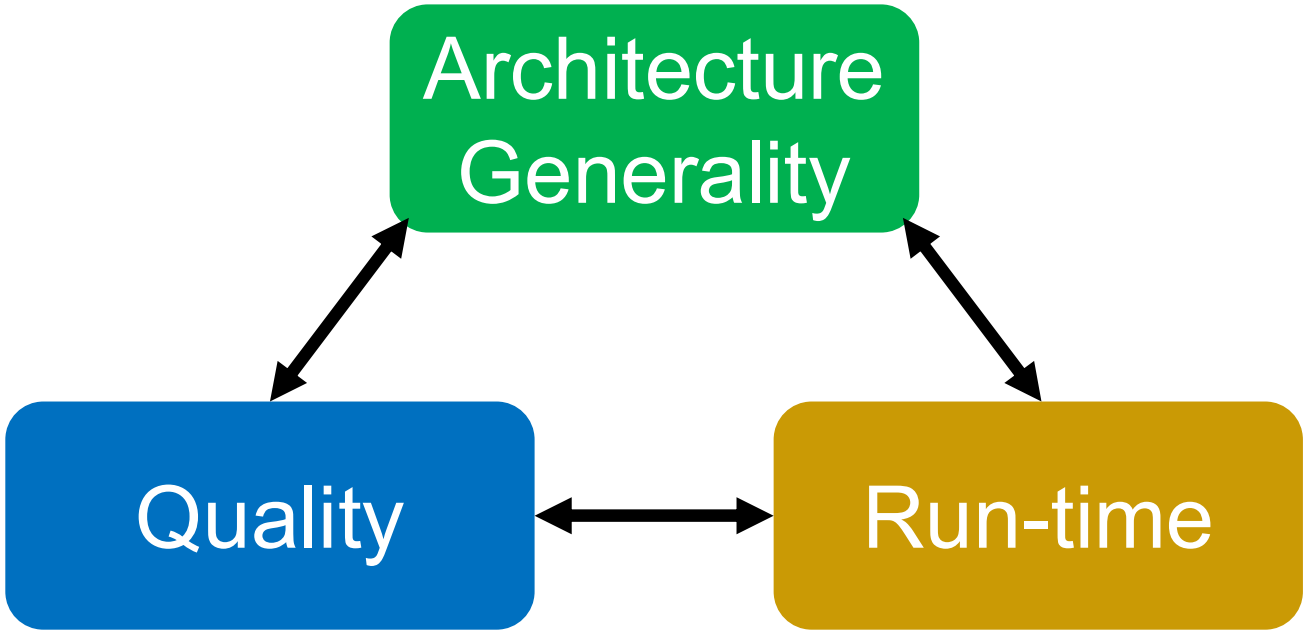
Open Source FPGA CAD Flow

Approach:

- Architecture Agnostic
- Data-driven FPGA Architecture description

VTR Challenges

- Explore larger design space

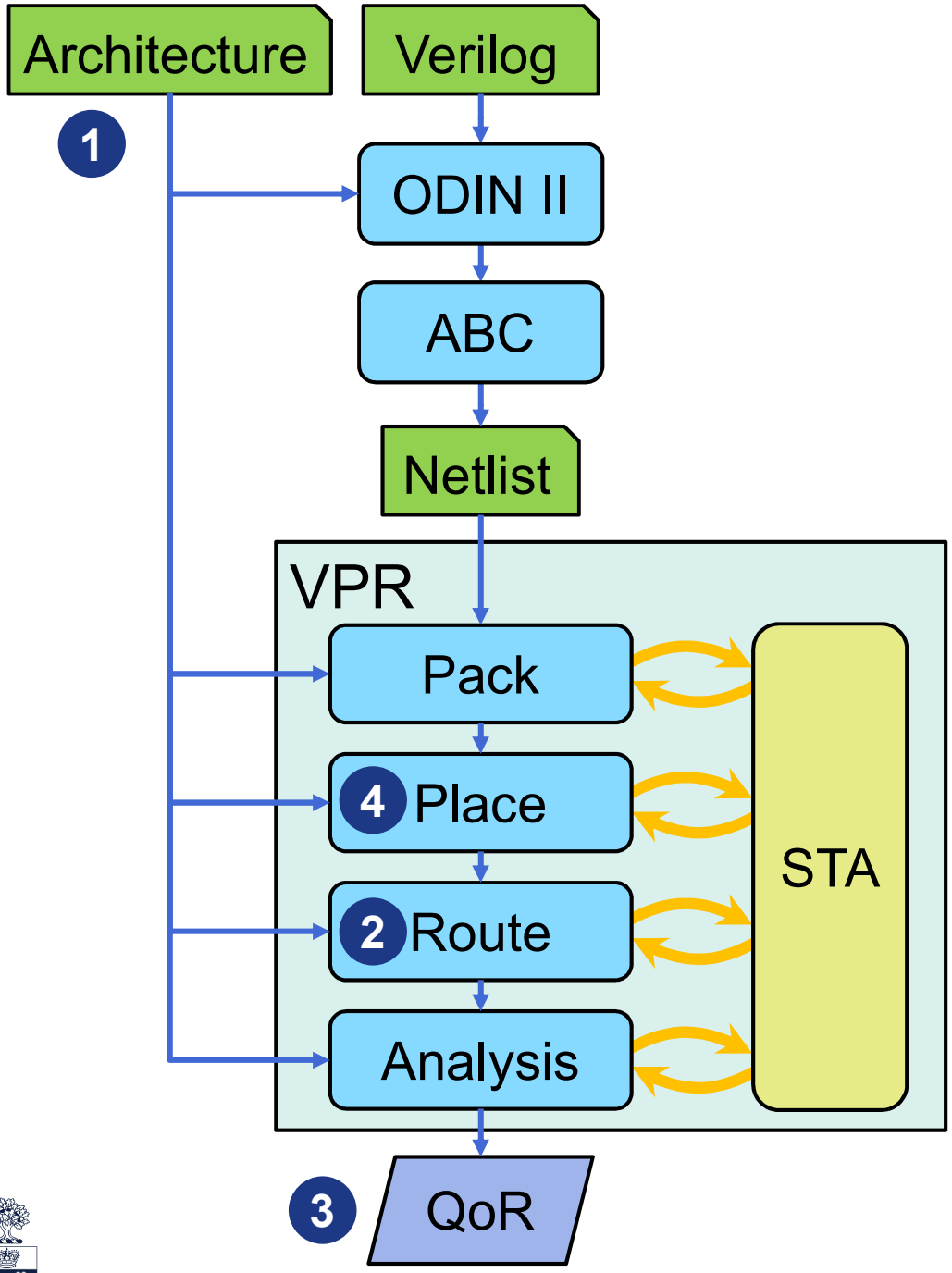


- Accurate architecture
  conc
- Redu
  effort

hours to ***days***
designs)

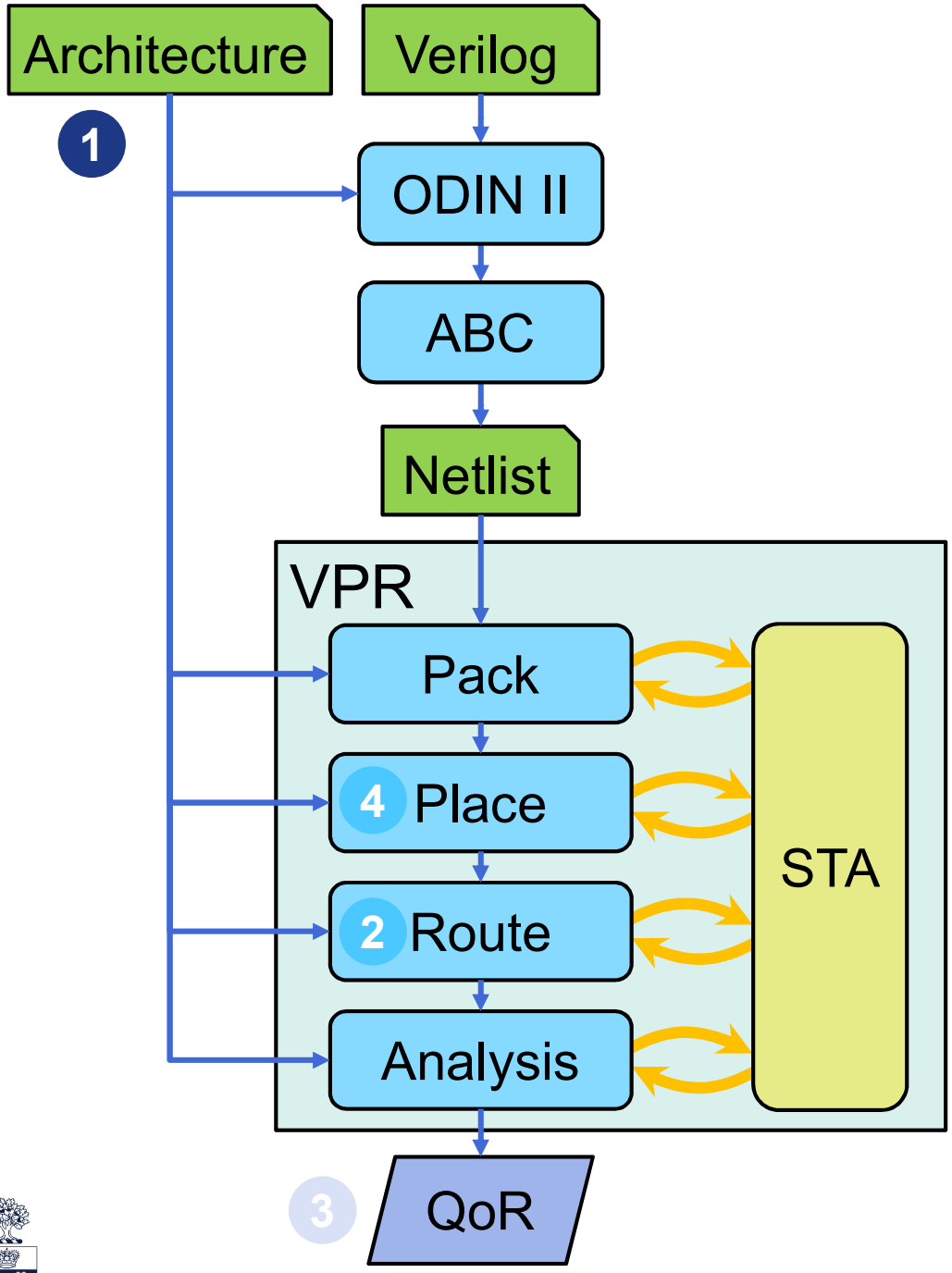
Can we have all three?

Outline



1. VTR 8 Capabilities & New Features
2. AIR: Adaptive Incremental Router
3. VTR 8 QoR, Run-time & CAD Enhancements
4. Reinforcement Learning Enhanced Placement

Outline



1 VTR 8 Capabilities & New Features

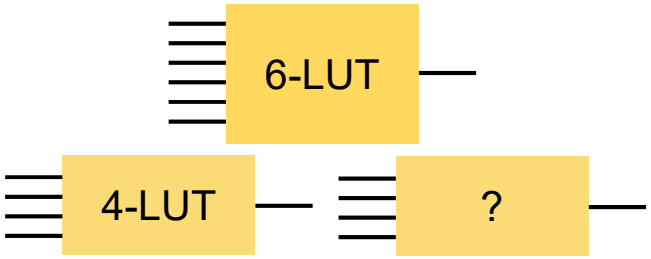
2 AIR: Adaptive Incremental Router

3 VTR 8 QoR, Run-time & CAD Enhancements

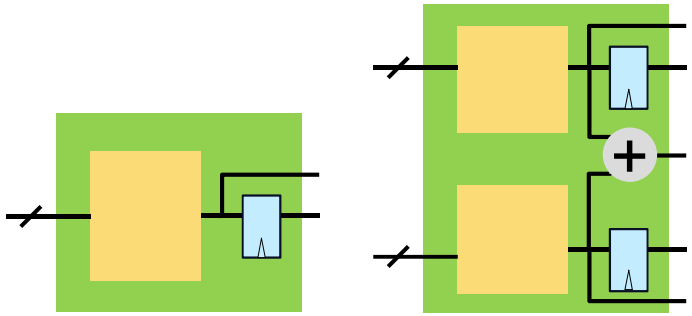
4 Reinforcement Learning Enhanced Placement

K. E. Murray, et al., "VTR 8: High Performance CAD and Customizable FPGA Architecture Modelling", *ACM TRETS, 2020*

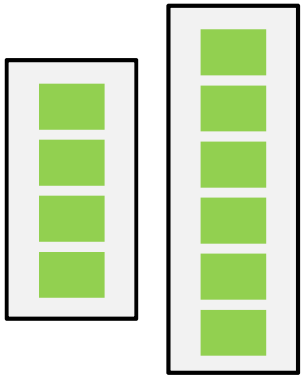
Block Modeling



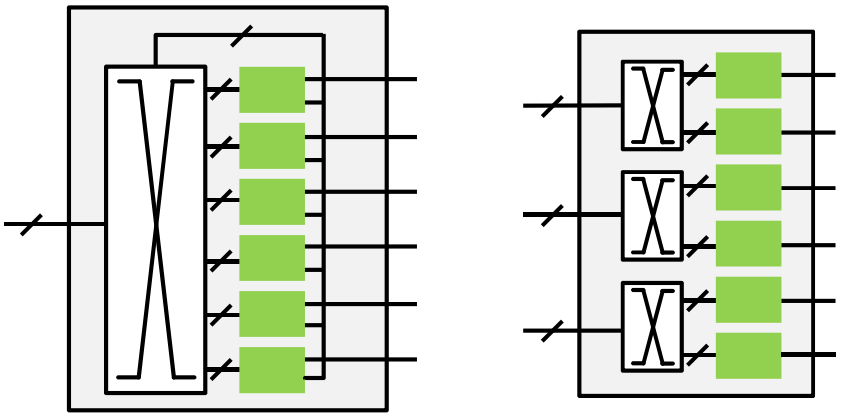
Logic Primitive



Logic Element



Logic Block Size

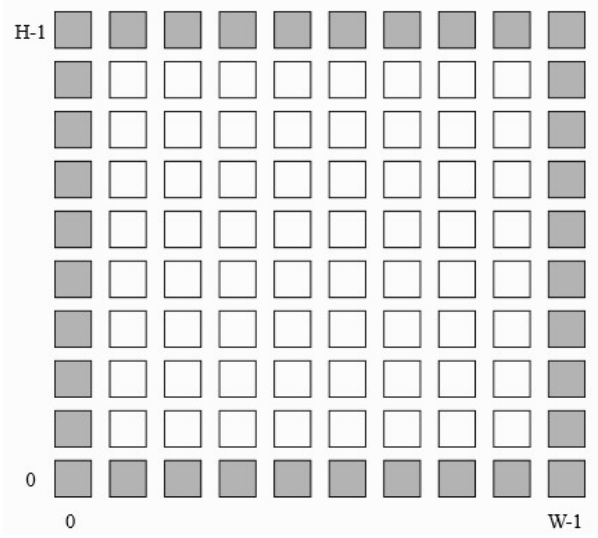


Inter-Element Connectivity

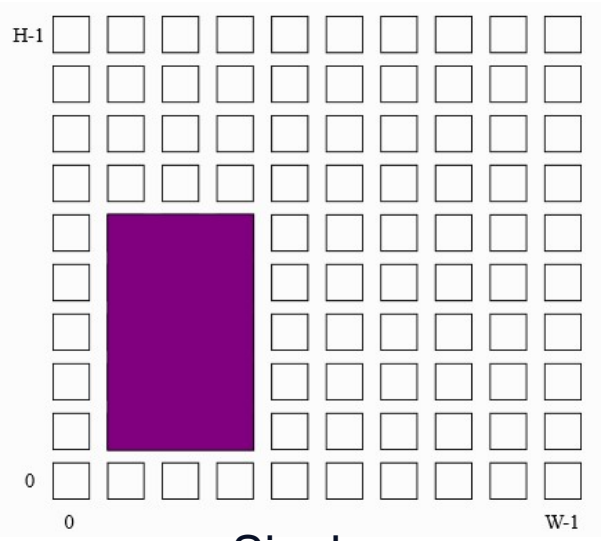
- Supports both soft & hard blocks
- Arbitrary:
  - Netlist Primitives
  - Hierarchy
  - Connectivity

General Device Grids

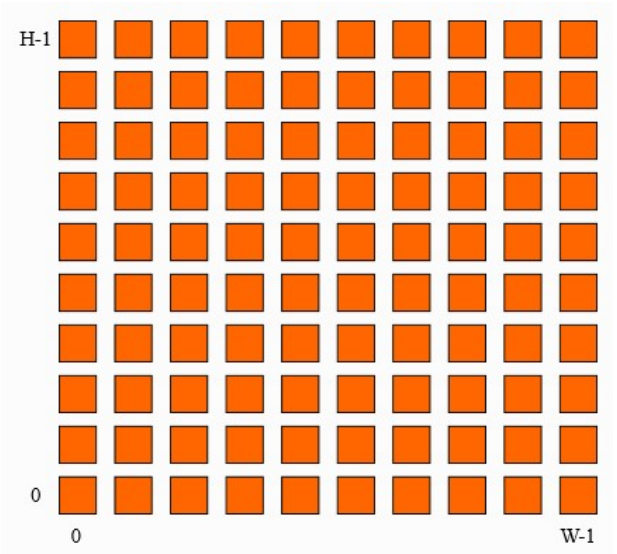
New!



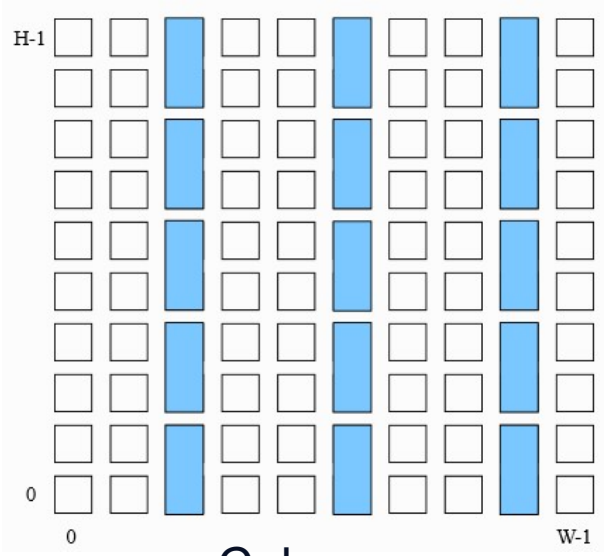
Perimeter



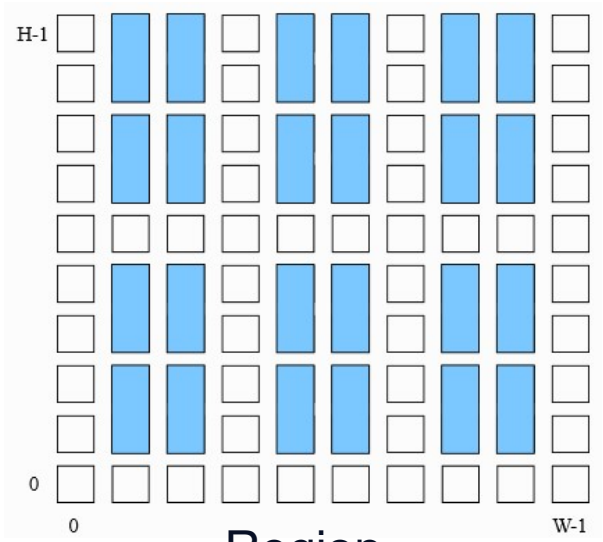
Single



Fill



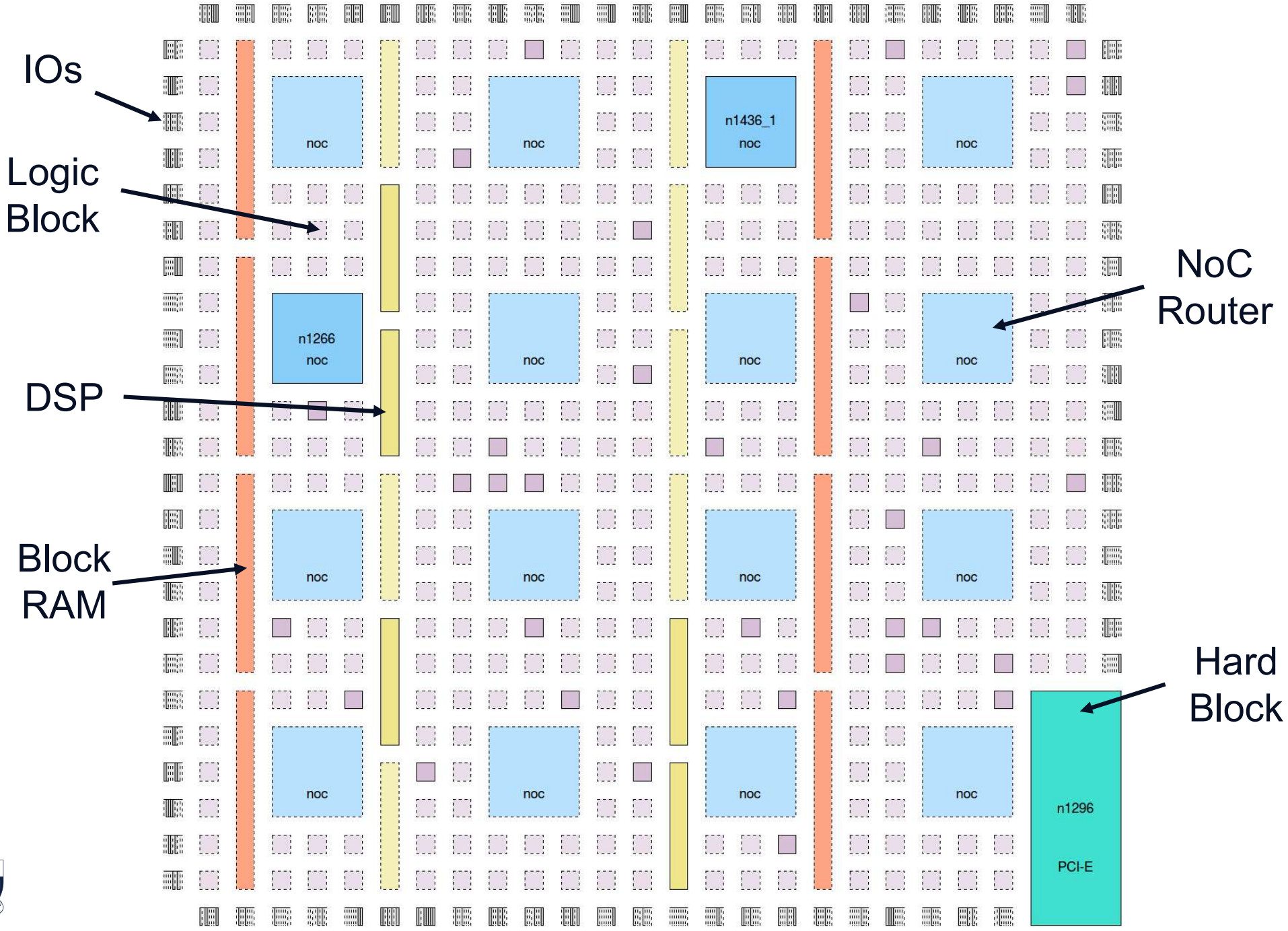
Column



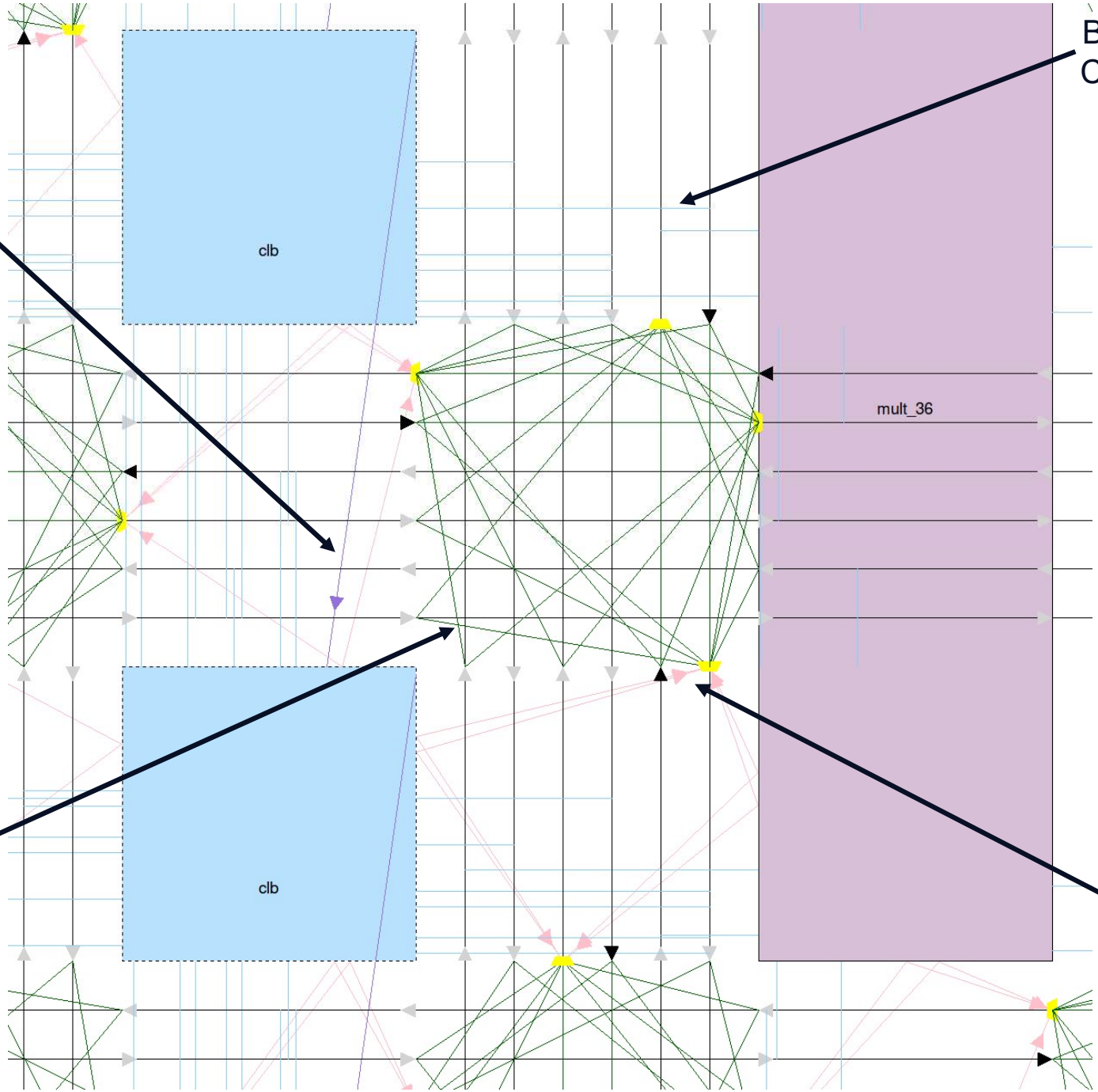
Region

Arbitrary Grid Example

New!



Detailed Routing Architecture



Block Input Connection

Direct Block-Block Connection

Routing Wires

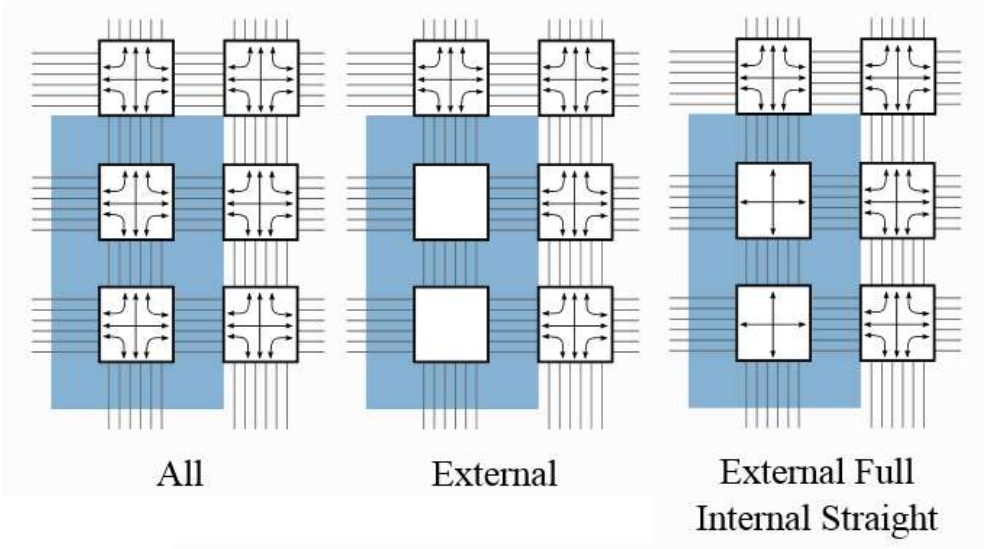
Wire-Wire Connections (switch-block)

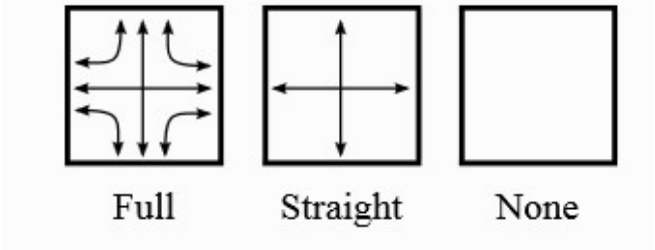
Block Output Connection

Customizable Routing Architectures

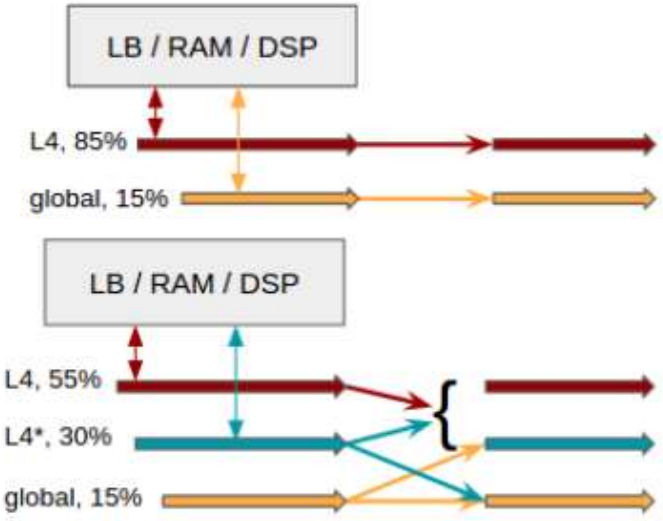
New!

Switch-block Locations

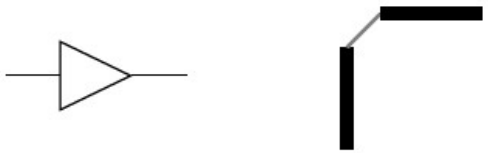




Wire Connectivity

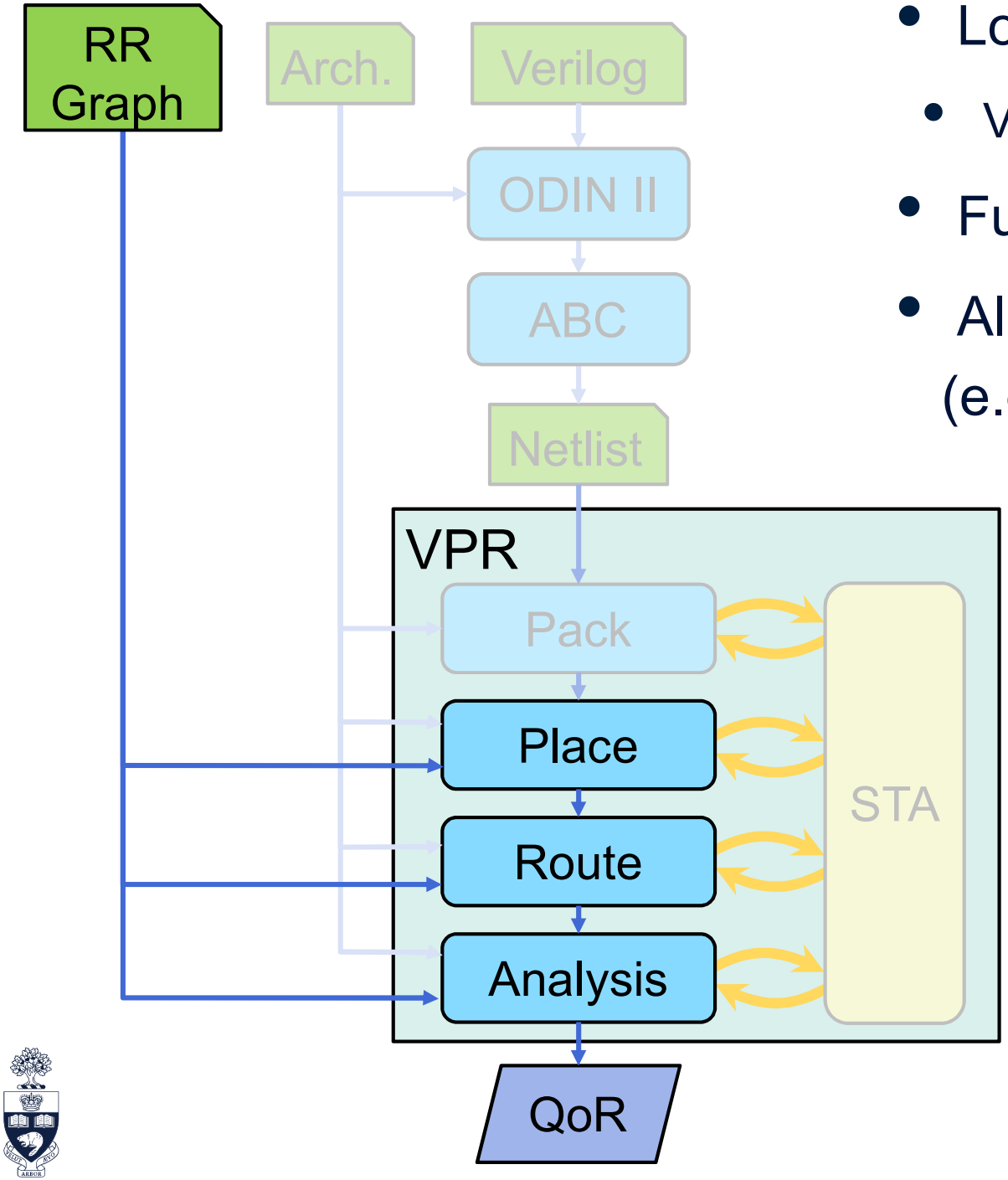


Non-Configurable Edges



Fully Custom Routing Architectures

New!



- Load RR graph from file
  - VPR or non-VPR generated
- Full control of routing network
- Allows freezing RR graph (e.g. taped-out)

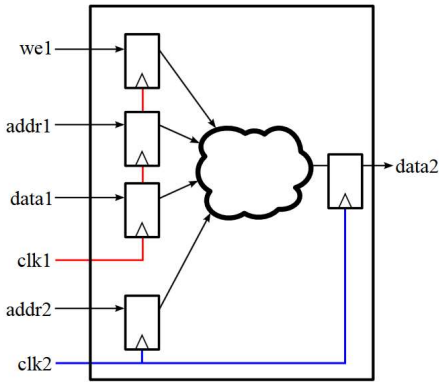
Analysis & Modeling

New & Enhanced!

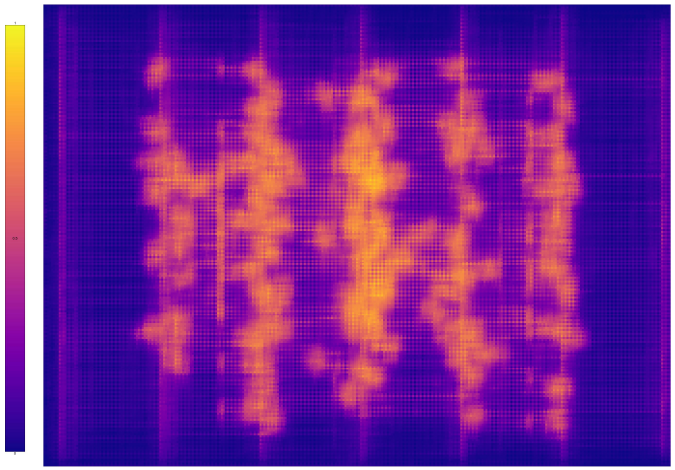
- Run analysis independent of optimization
- Improved visualizations & reporting
- Improved area models
- Full featured timing analysis
  - Min & max timing models
  - Multi-clock primitives

```
Startpoint: FFC.Q[0]
Endpoint  : out:out1.outpad[0]
Path Type : setup

Point                                    Incr      Path
-----------------------------------------------------------
clock clk (rise edge)                    0.000     0.000
clock source latency                     0.000     0.000
clk.inpad[0] (.input)                    0.000     0.000
FFC.clk[0] (.latch)                      0.042     0.042
FFC.Q[0] (.latch) [clock-to-output]      0.124     0.166
out:out1.outpad[0] (.output)             0.550     0.717
data arrival time                                  0.717

clock virtual_io_clock (rise edge)       0.000     0.000
clock source latency                     0.000     0.000
clock uncertainty                        0.000     0.000
output external delay                    0.000     0.000
data required time                                 0.000
-----------------------------------------------------------
data required time                                 0.000
data arrival time                                 -0.717
-----------------------------------------------------------
slack (VIOLATED)                                  -0.717
```

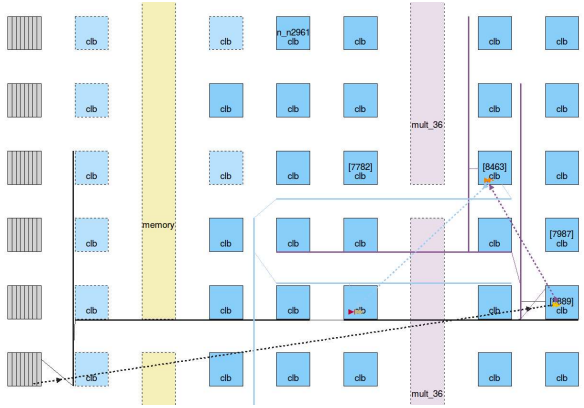
Timing Reports



Multi-clock Primitives

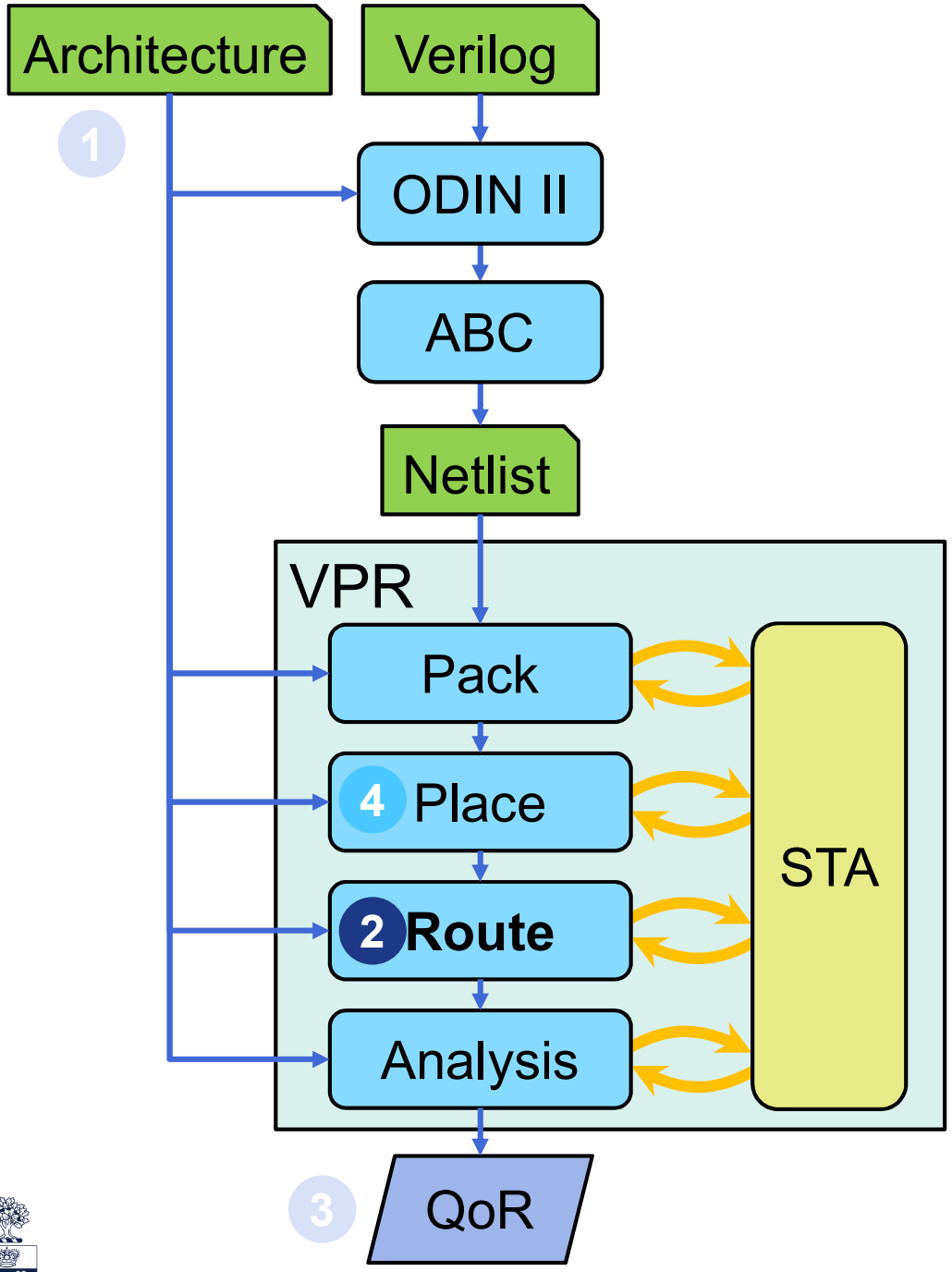


Routing Utilization



Critical Path

Outline



1 VTR 8 Capabilities & New Features

2 AIR: Adaptive Incremental Router

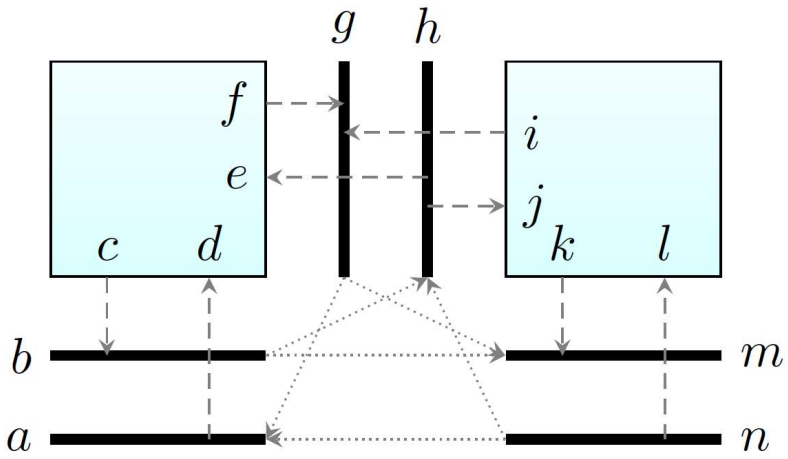
3 VTR 8 QoR, Run-time & CAD Enhancements

4 Reinforcement Learning Enhanced Placement

K. E. Murray, et al., "AIR: A Fast but Lazy Timing-Driven FPGA Router", *ASP-DAC*, 2020, 1-7.

FPGA Routing Problem

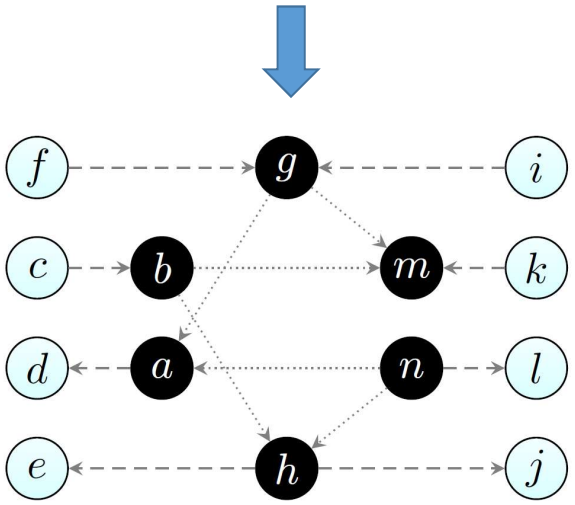
- Model interconnect network as Routing Resource (RR) graph:
  - Nodes: Conductors (wires/pins)
  - Edges: Configurable switches



Routing Architecture

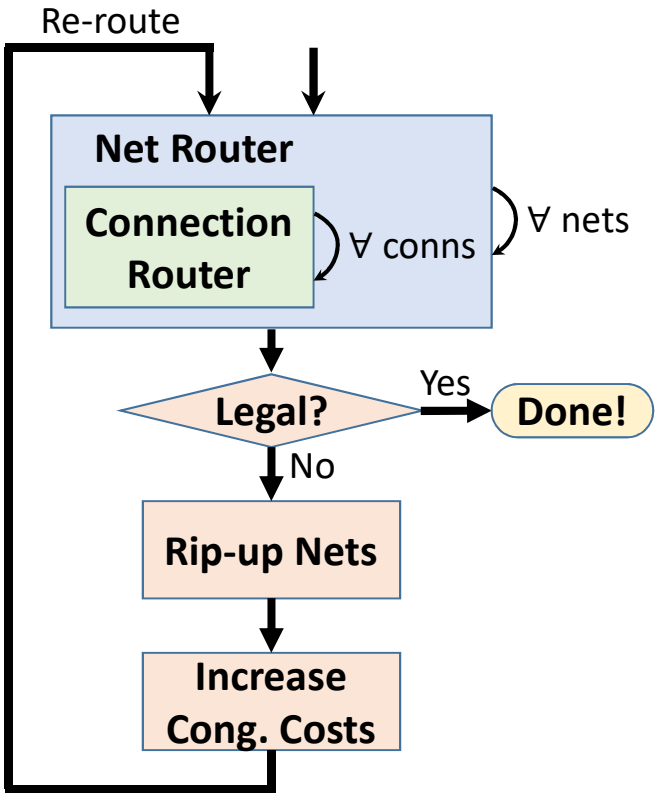
- FPGA Routing Problem:
  - Find embedding of netlist in RR graph
  - Requires finding many non-overlapping trees in RR graph
  - Minimize timing and wiring (power)
  - Reasonable run-time

- Usually solved as single combined global/detailed routing stage

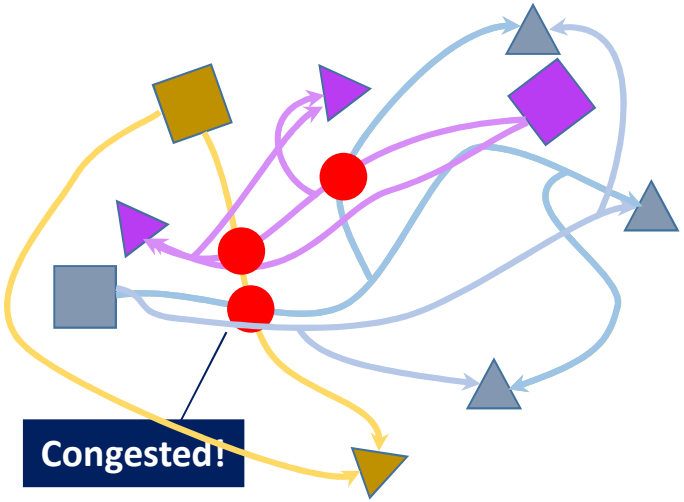


Routing Resource (RR) Graph

Negotiated Congestion Routing



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



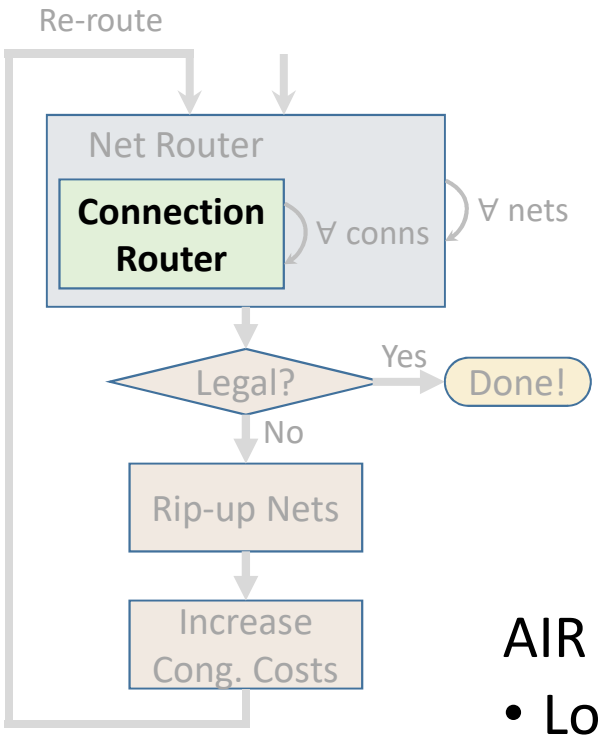
Congestion Free!

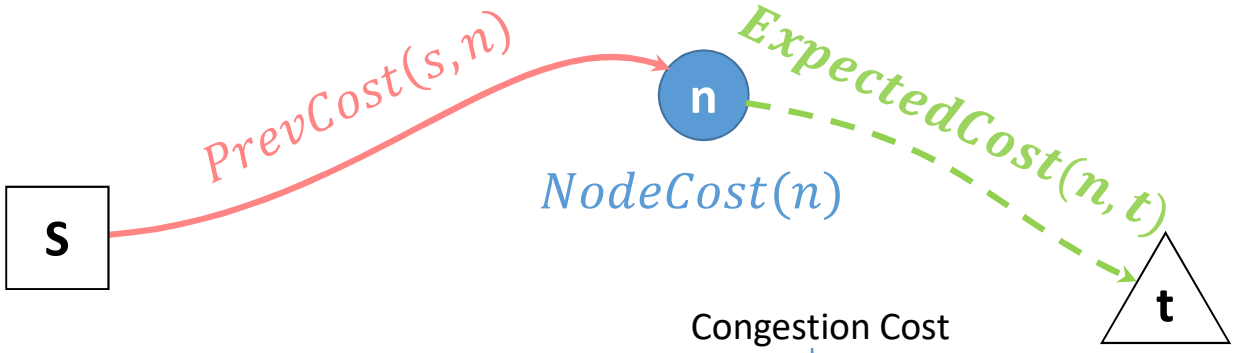
AIR: Adaptive Incremental Router

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

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

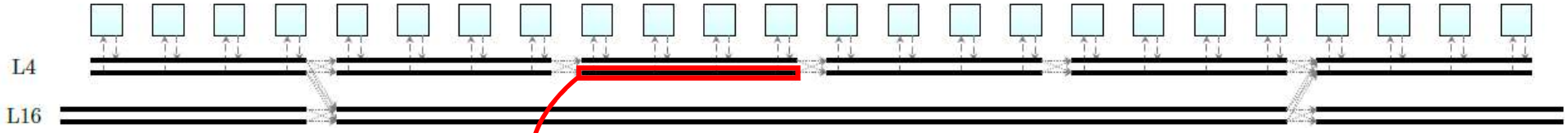
Timing Criticality

AIR uses adaptive:

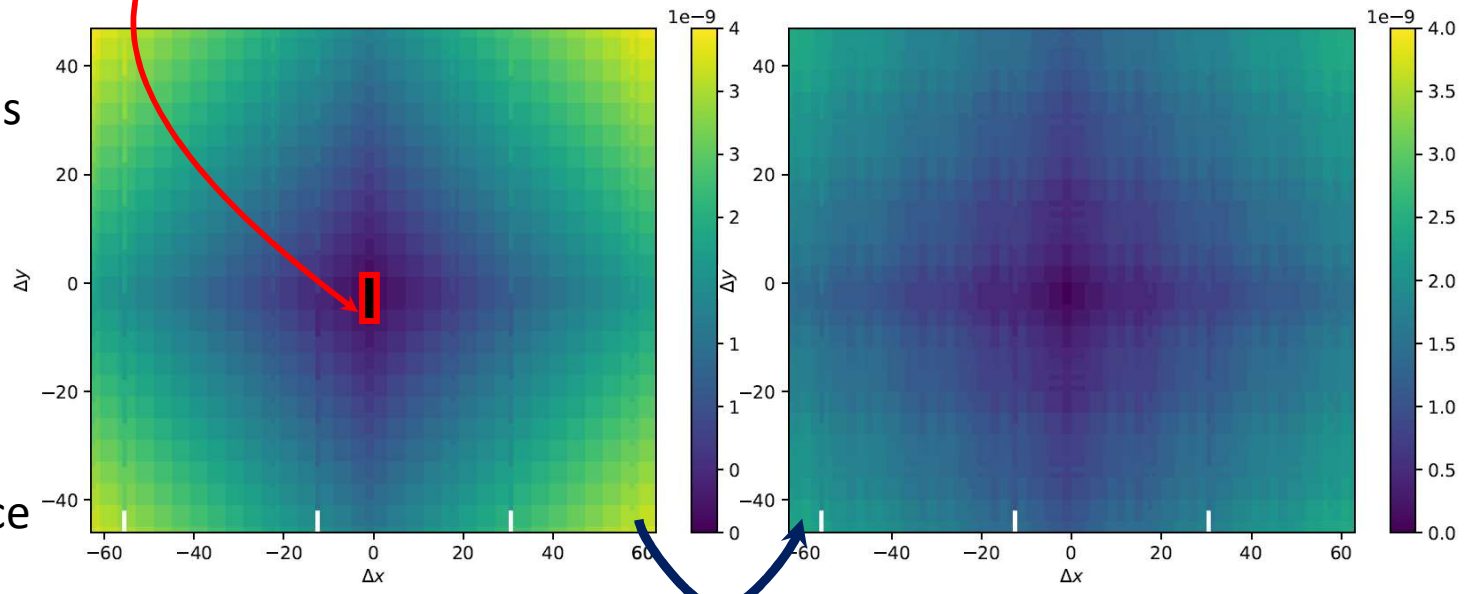
- Lookahead: $\boldsymbol{ExpectedCost}$
- Base Costs : $\boldsymbol{b(n)}$ (see paper)

Adaptive Lookahead

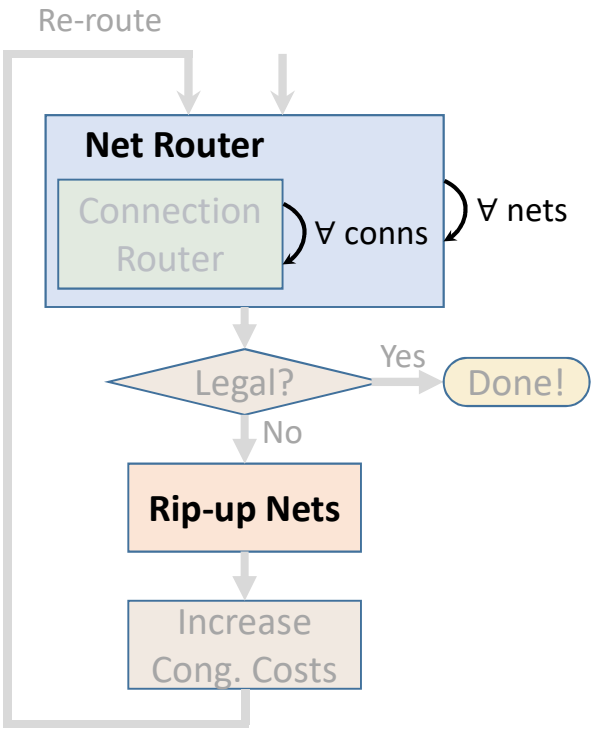
- Observation: Lookahead should adapt to routing architecture



- Challenges:
  - Many FPGA Architectures
  - Huge Graph
  - Reasonable build time
  - Fast evaluation
- Approach:
  - Profile graph with un-directed Djikstra flood & automatically data reduce
  - Adapts to graph
  - More accurate → faster for equivalent/better quality



Net Router: Lazy Routing

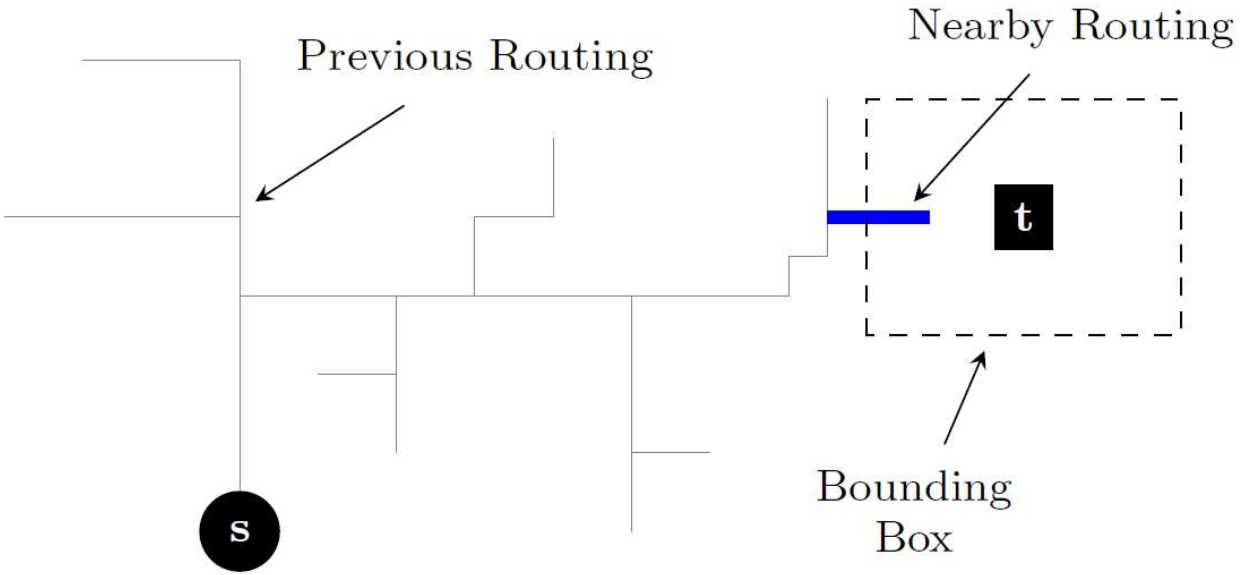


AIR Routes Nets Lazily:

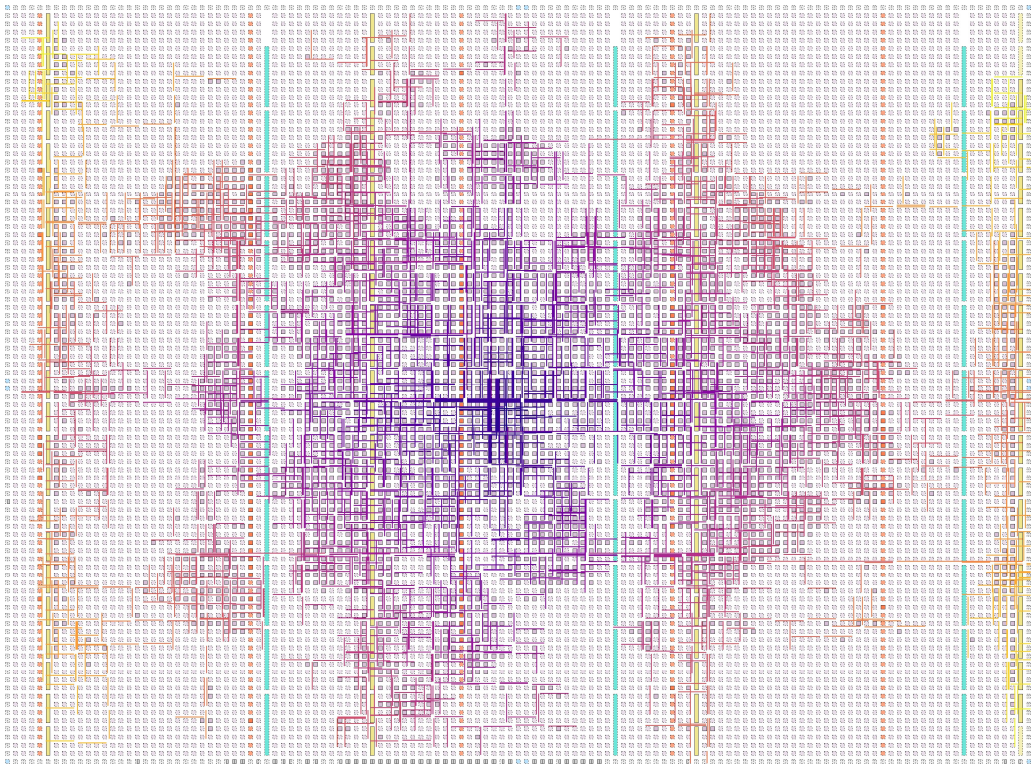
- Efficiently handle High-Fanout nets
- Incrementally re-route congested nets

High-Fanout (HF) Routing

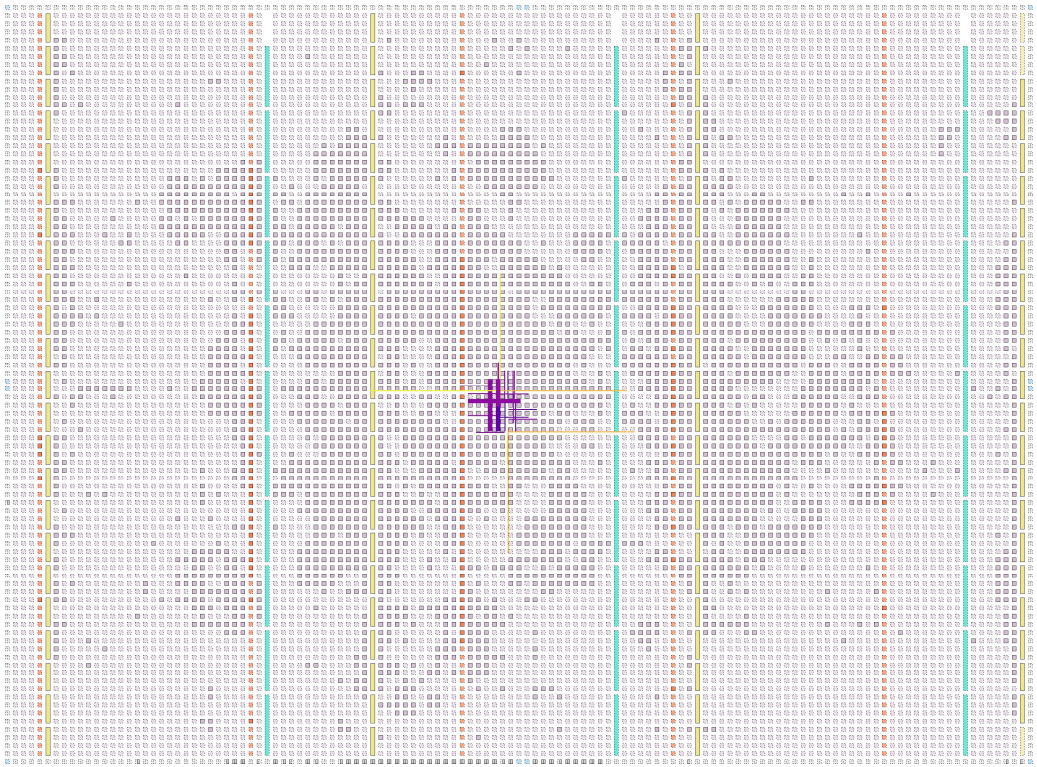
- Observation: Most routing of HF net irrelevant for a particular sink
  - But still consider as potential branch points (expensive)
- Optimization:
  - Only consider branch points spatially *nearby* target
  - Don't even look at others
  - Limit search to region near target



HF Router Expansion



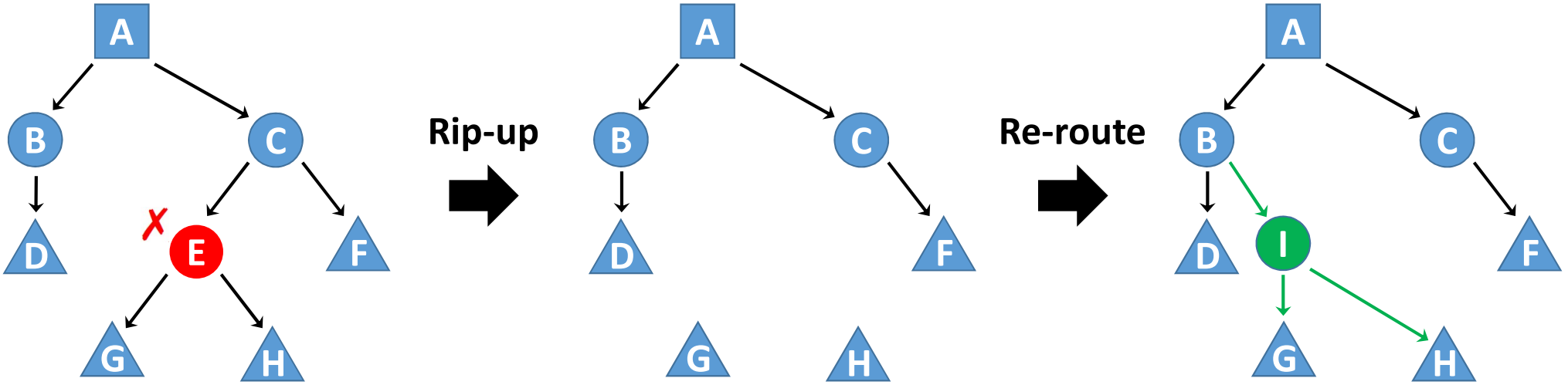
Traditional



Spatial Lookup

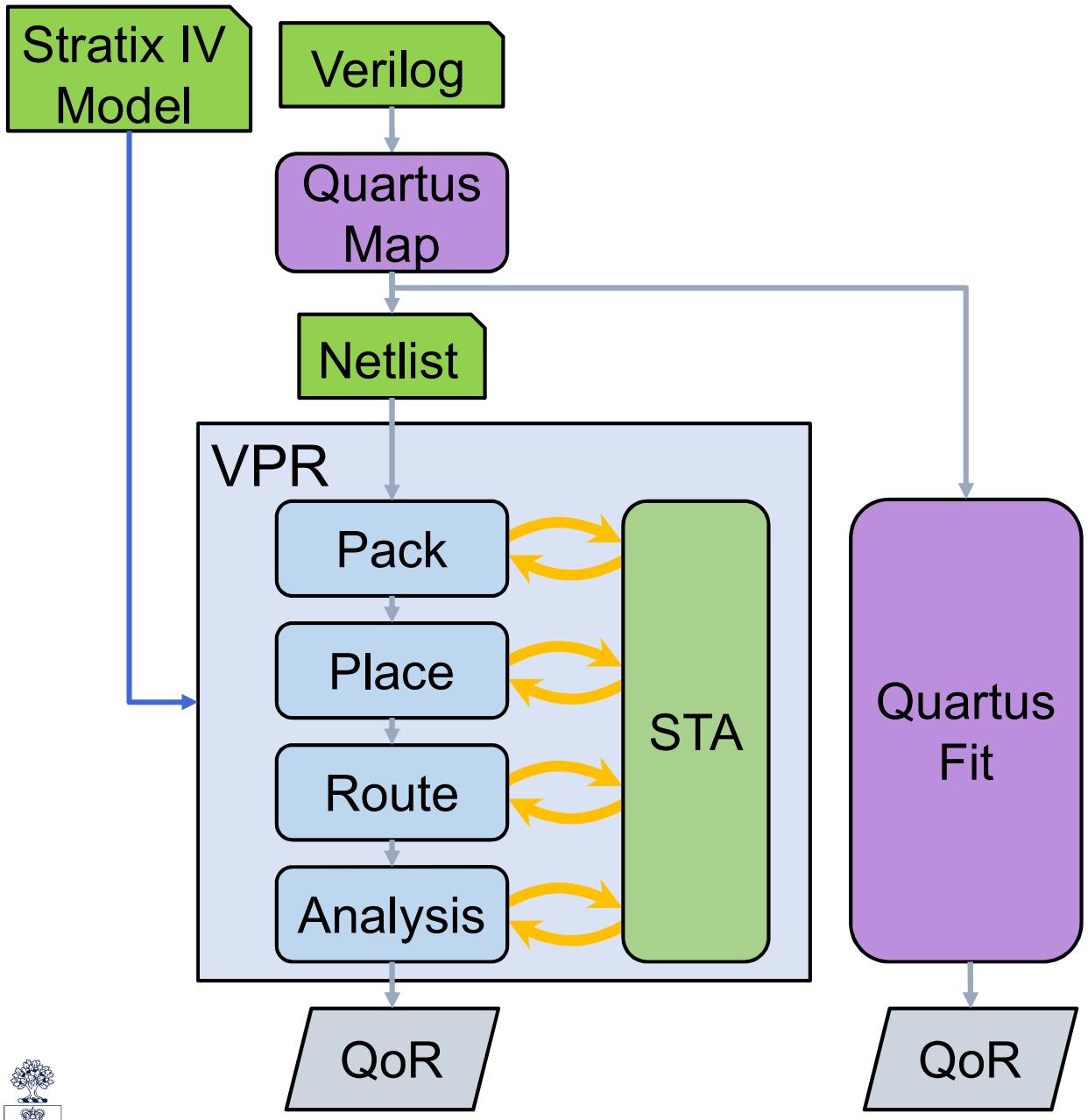
Incremental Routing

- 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

Evaluation Titan Flow & Benchmarks



Titan Flow:

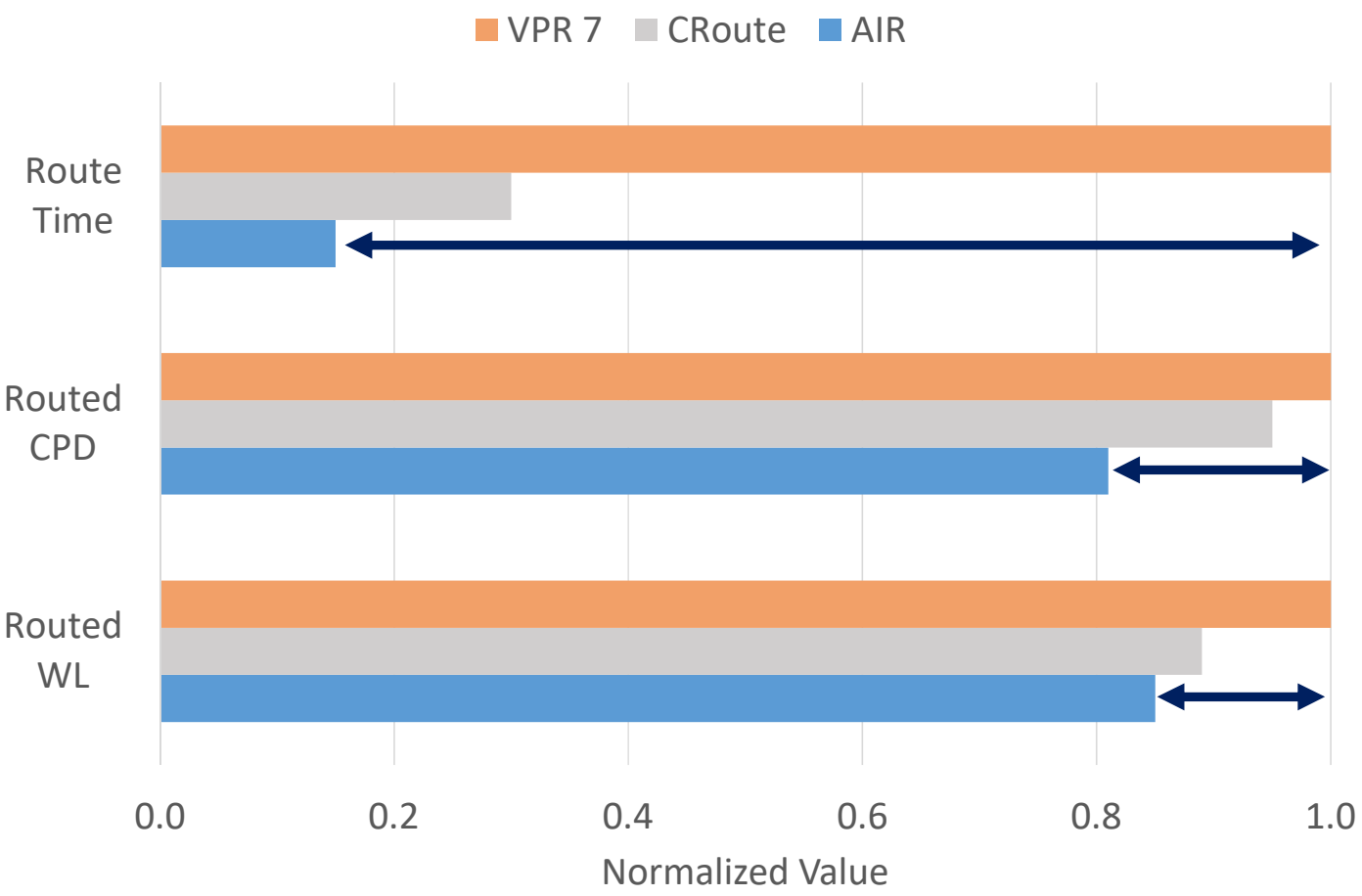
- Synth. & Tech-Map with Quartus
- Pack/Place/Route with Quartus or VPR

Titan Benchmarks:

- Target Stratix IV
- 23 designs, 90K-1.9M primitives

- Academic Routers:
  - VPR 7
  - CRoute
  - AIR
- Commercial Routers:
  - Intel Quartus 18.0

AIR Outperforms Academic Routers



> 6.6x run-time reduction

- Incr. Routing
- HF Routing
- Lookahead

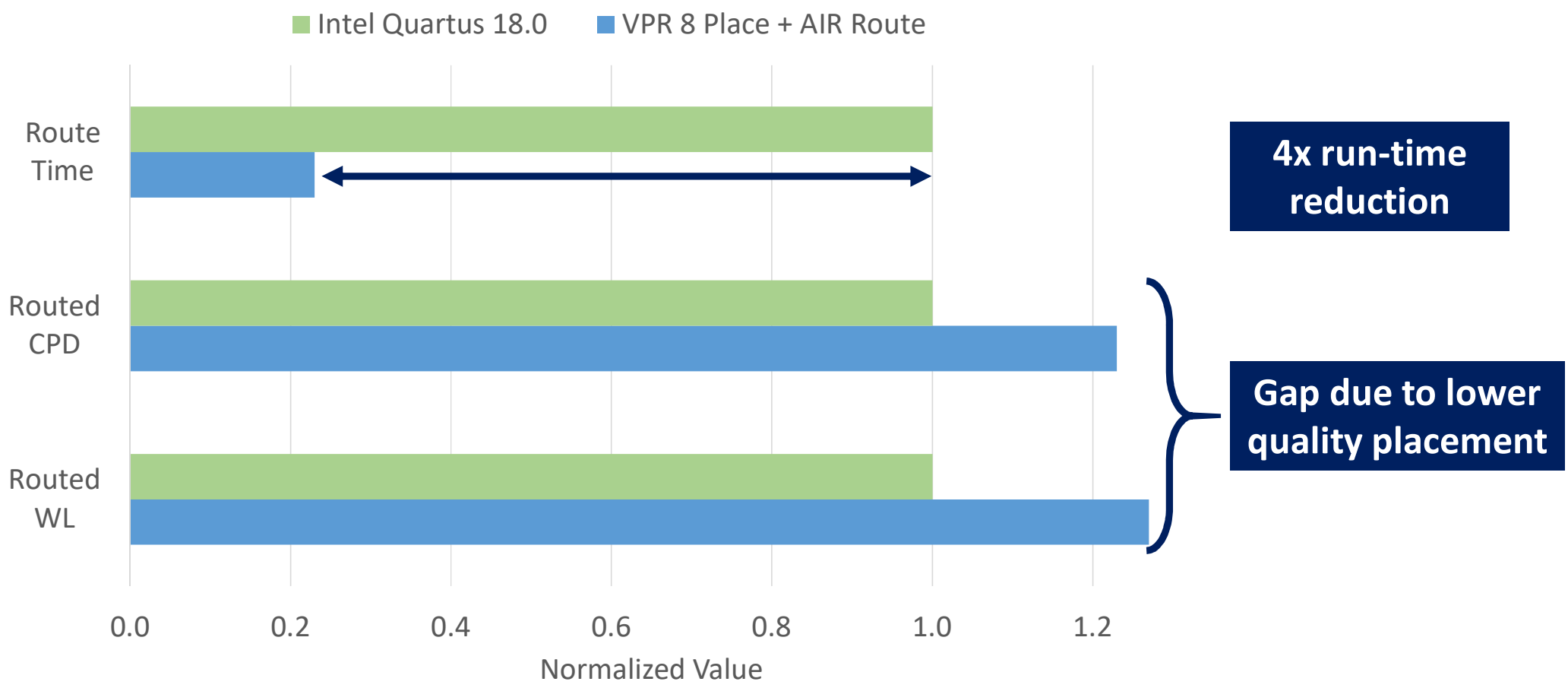
19% faster critical path

- Lookahead

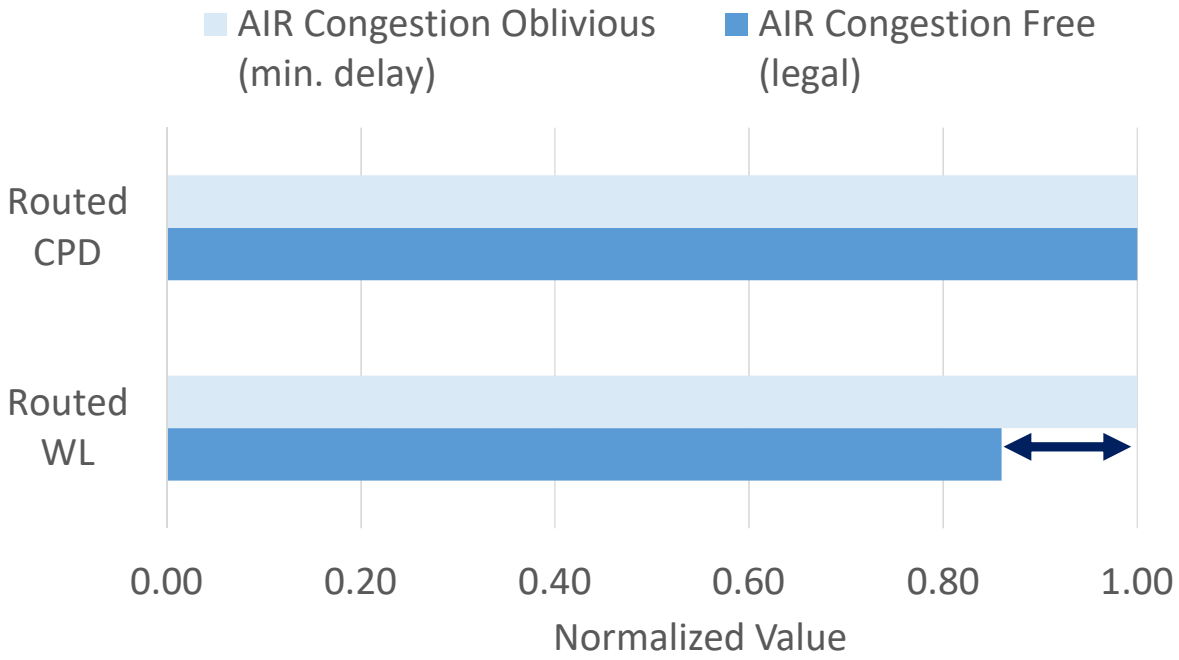
15% less wiring

- Resource Base Costs
- Lookahead

AIR is Faster than Quartus



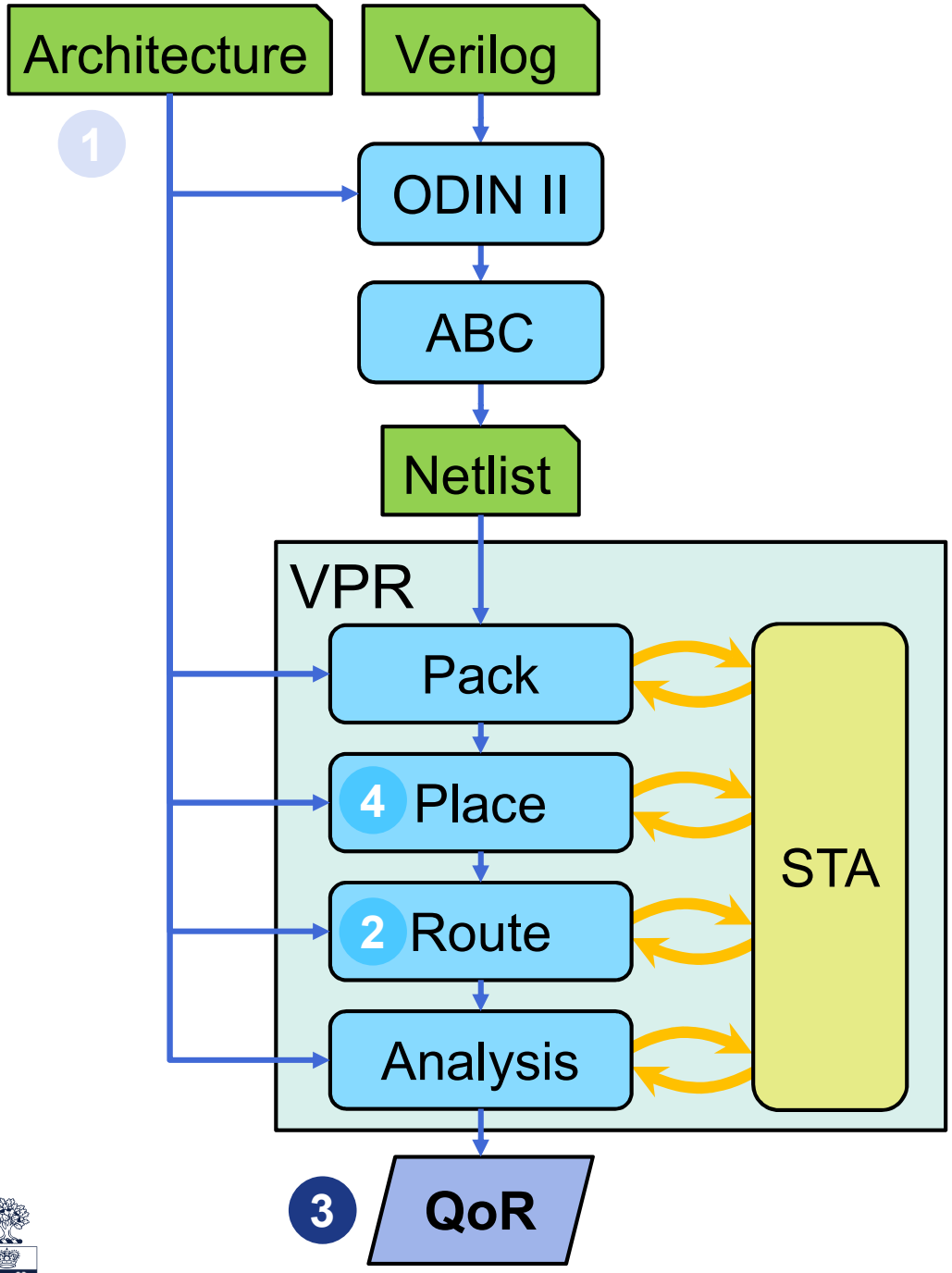
AIR Effectively Resolves Congestion



Critical Path not degraded

Wiring reduced

Outline



1 VTR 8 Capabilities & New Features

2 AIR: Adaptive Incremental Router

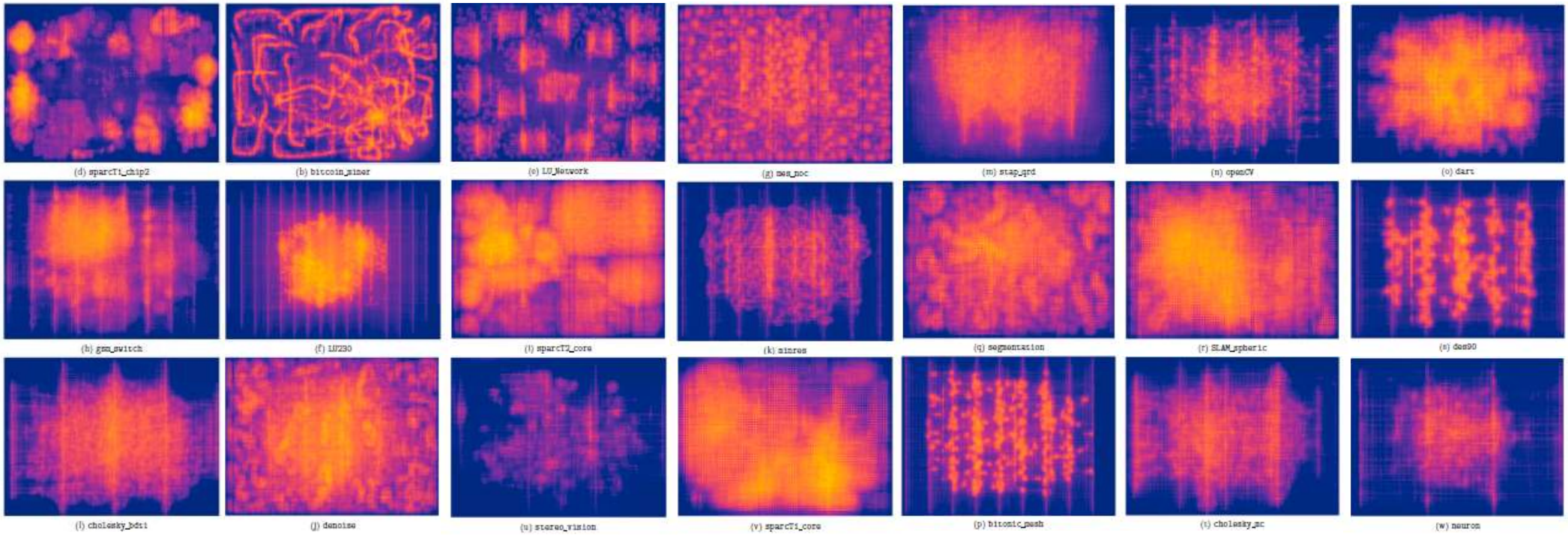
3 VTR 8 QoR, Run-time & CAD Enhancements

4 Reinforcement Learning Enhanced Placement

K. E. Murray, et al., "VTR 8: High Performance CAD and Customizable FPGA Architecture Modelling", *In submission to ACM TRETS, 2020*

Titan Benchmark Completion

Tool	Routed Benchmarks	Unroutable Benchmarks	Device Too Small
VTR 7	14	9	-
VTR 8	23	0	-
Quartus 18.0	20	1	2

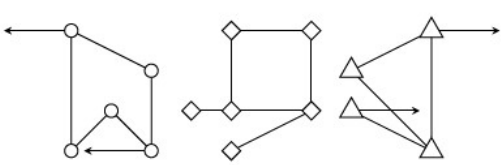


VTR 8 Titan Benchmark Implementations

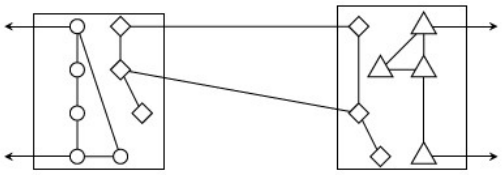
Packing & Placement Enhancements

Avoid highly interconnected (unnatural) clusters

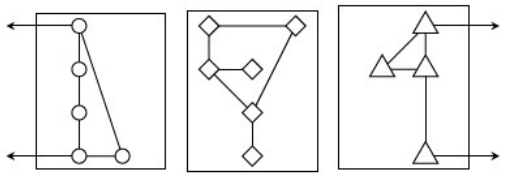
- Avoid packing unrelated logic
- Include more indirect (e.g. transitive) connectivity measures
- De-prioritize high-fanout nets



Unclustered



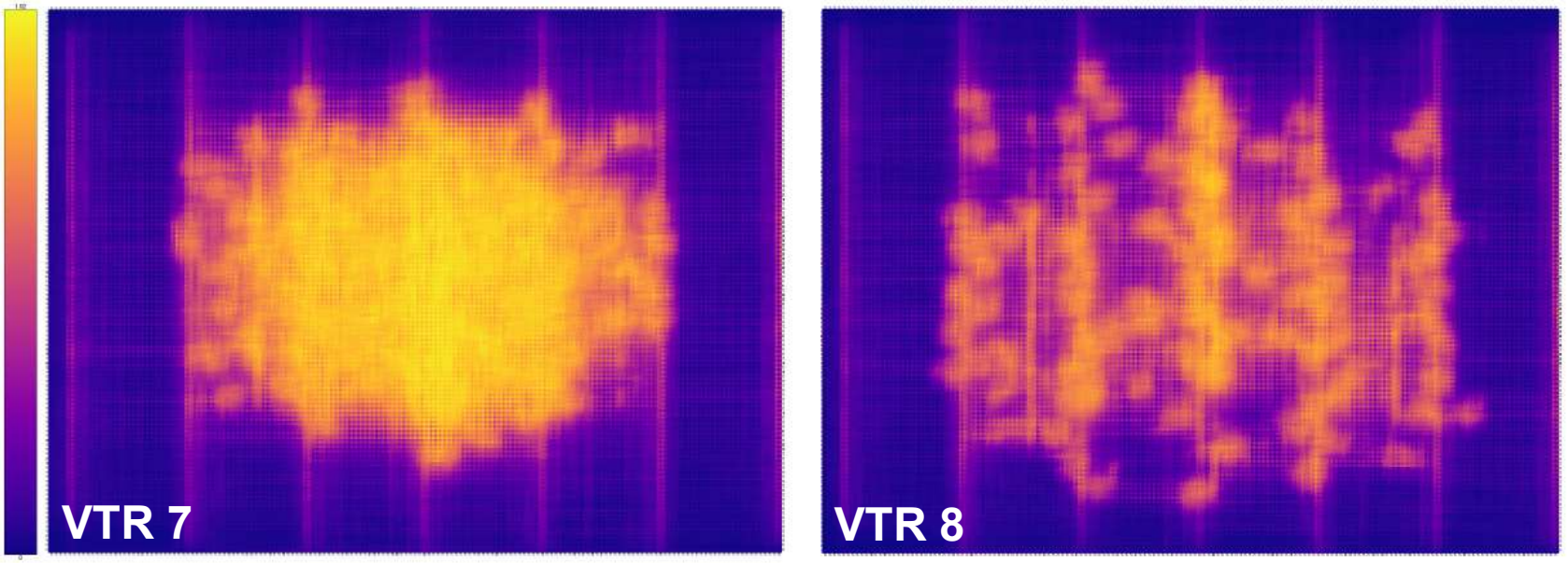
Clustered with Unrelated Logic



Naturally Clustered

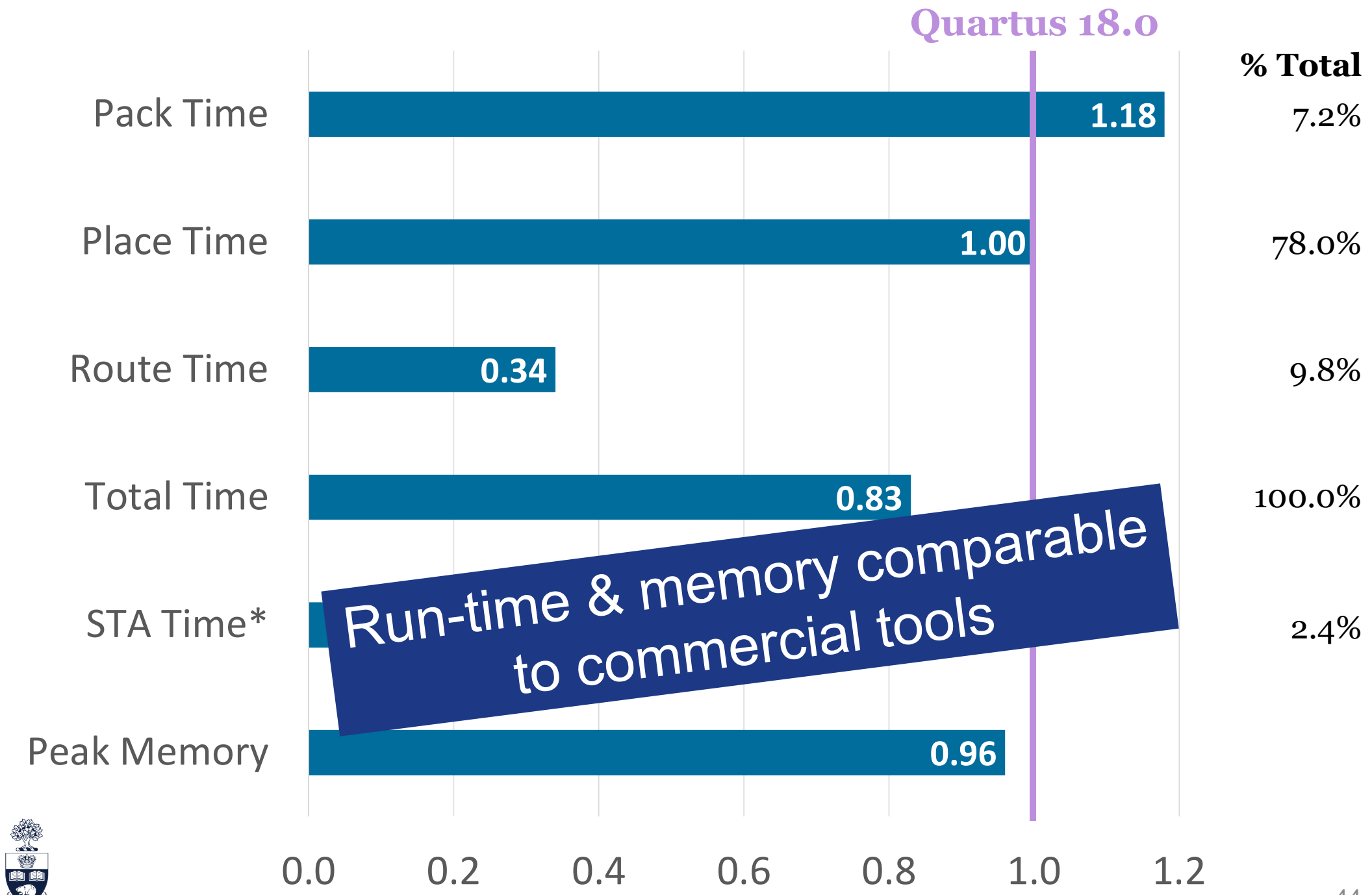
Improved Placement Exploration

- Better macro/carry-chain swapping
- Improved swapping of sparse blocks

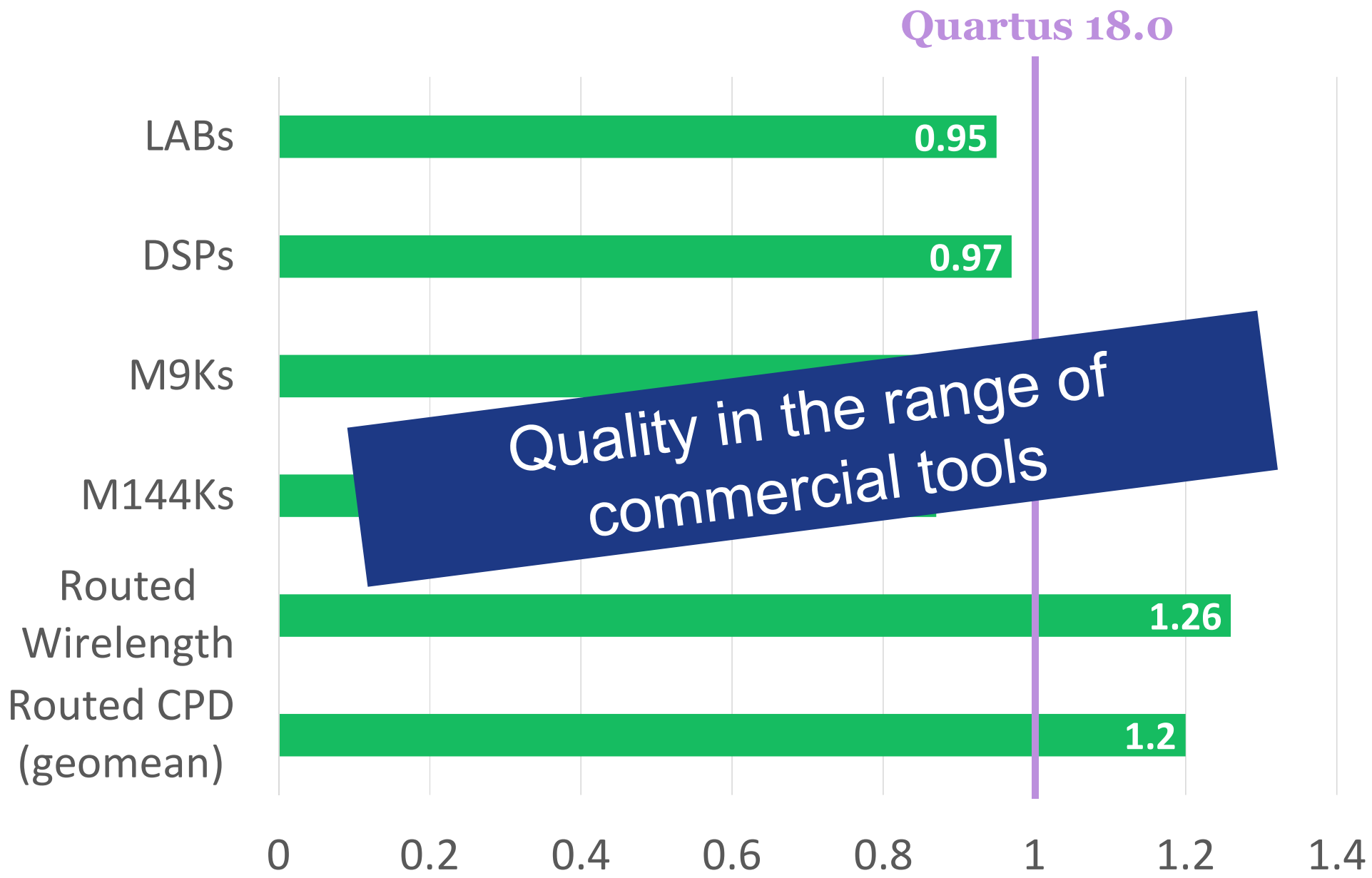


des90 Routing Utilization

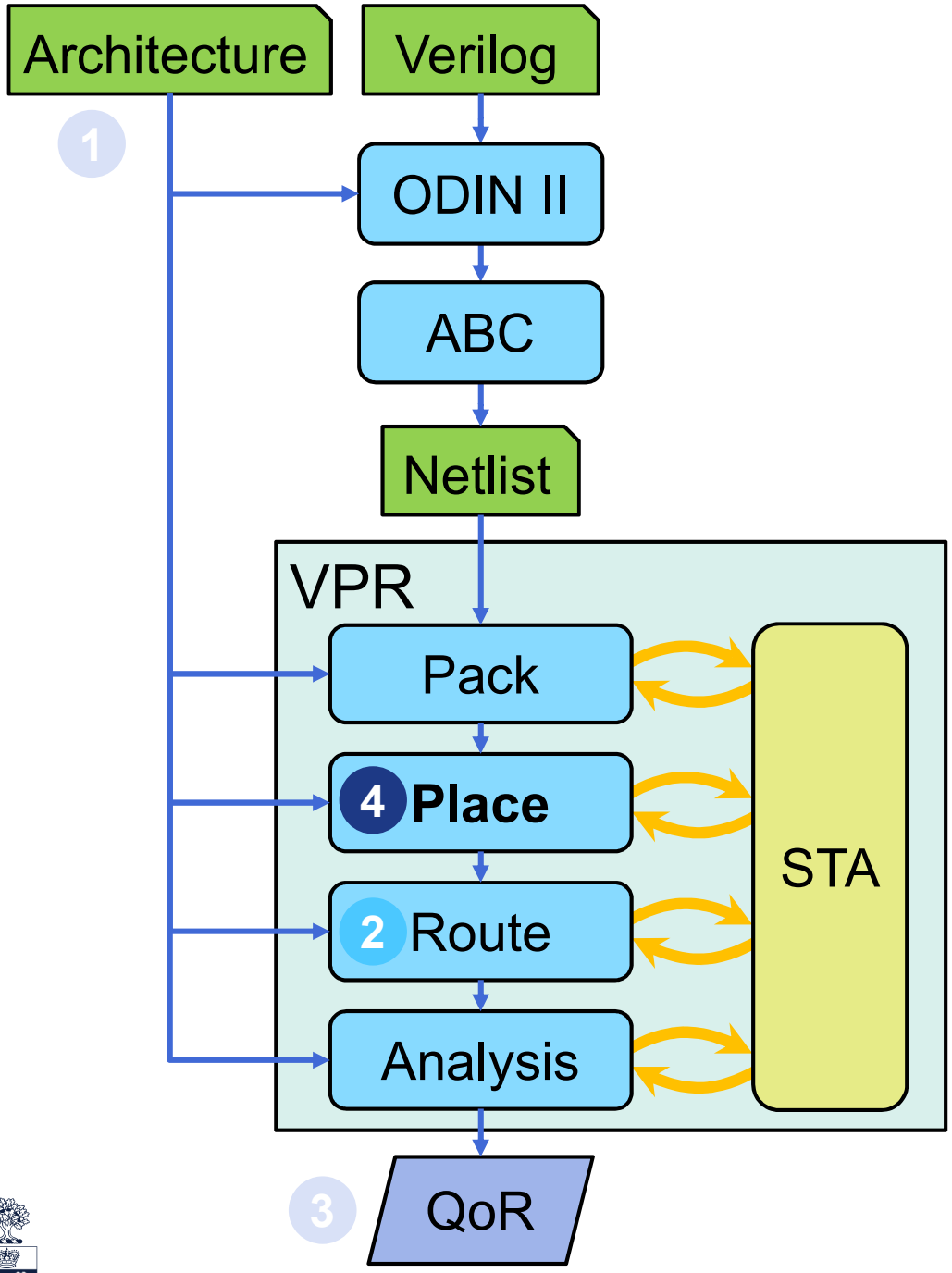
VTR 8 Run-time & Memory Footprint



VTR 8 QoR


Quartus 18.0
LABs
0.95
DSPs
0.97
M9Ks
M144Ks
Quality in the range of commercial tools
Routed Wirelength
1.26
Routed CPD (geomean)
1.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4

Outline



1 VTR 8 Capabilities & New Features

2 AIR: Adaptive Incremental Router

3 VTR 8 QoR, Run-time & CAD Enhancements

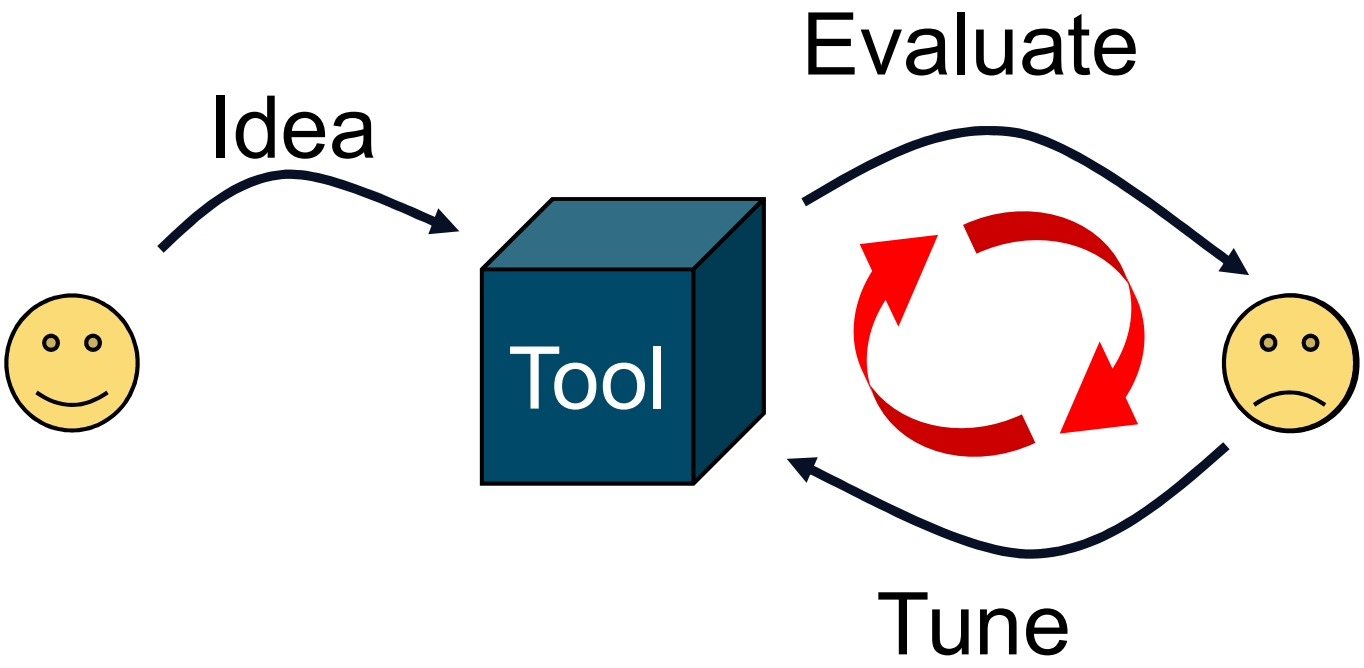
4 Reinforcement Learning Enhanced Placement

K. E. Murray and V. Betz, “Adaptive FPGA Placement Optimization via Reinforcement Learning", *MLCAD, 2019*

Reinforcement Learning & CAD

CAD Tool Development: Human-in-the-loop

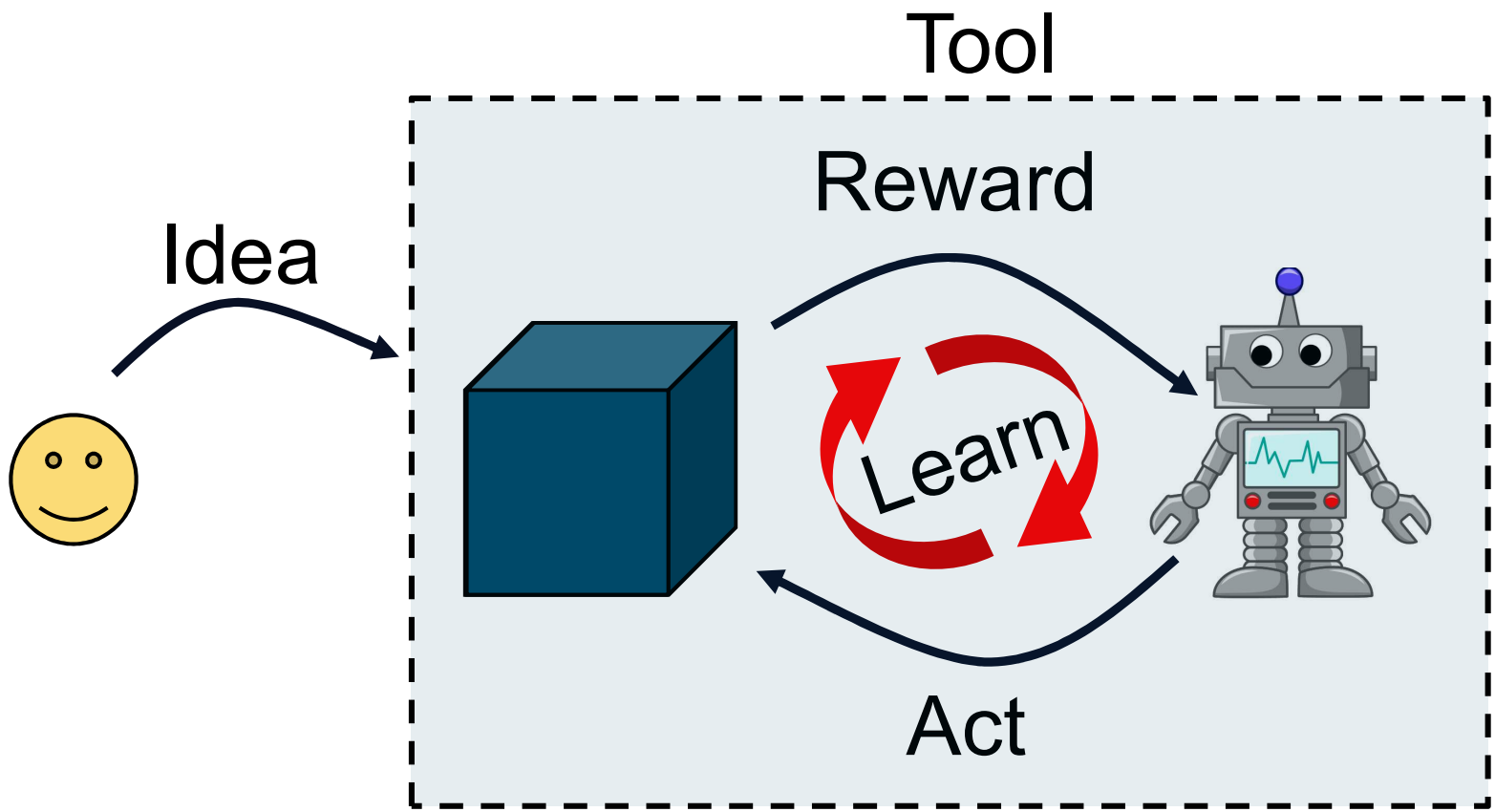
Large solution space → Use heuristics



With human-in-the-loop:

- Slow
- Simple heuristics, limited tool parameters (to keep tractable)
- Tune for average case (can't investigate every benchmark design)

CAD Tool Development: Reinforcement Learning (RL)



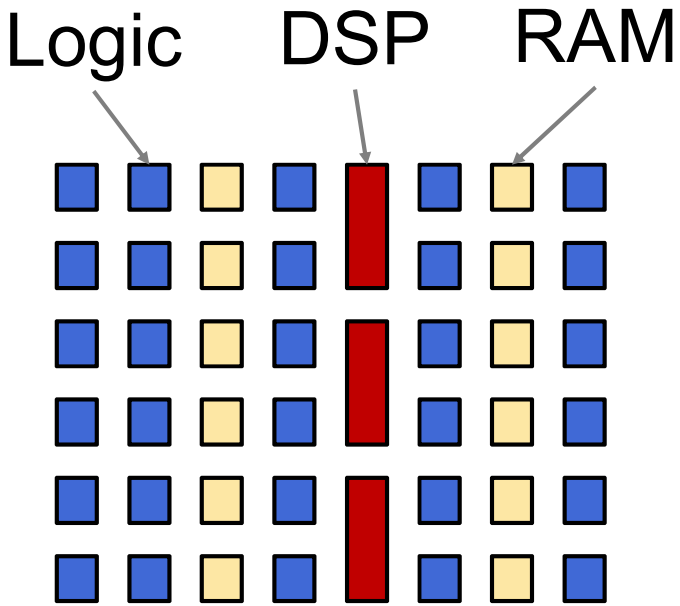
With RL:

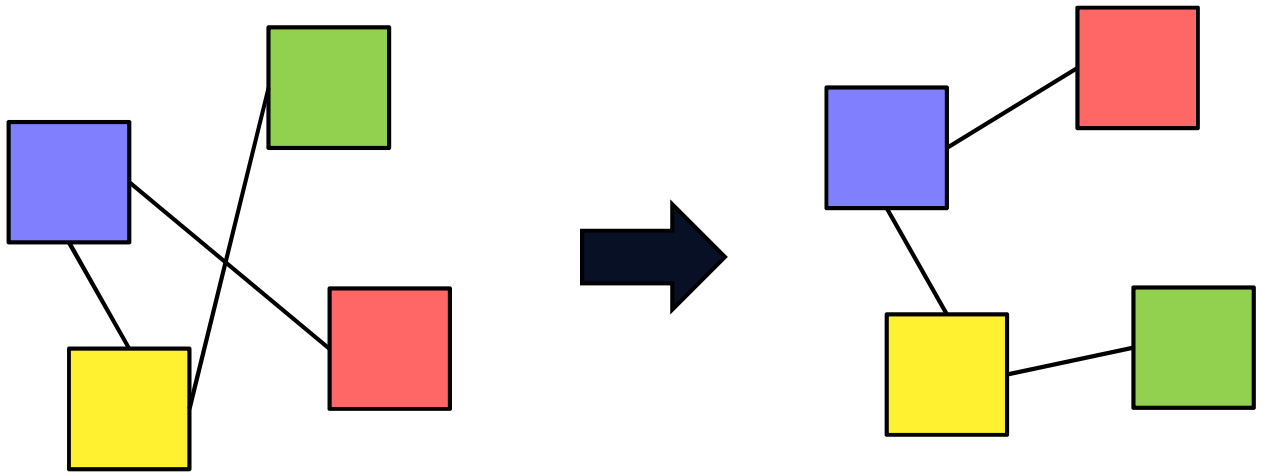
- Human ***out*** of the loop!
- Learn better heuristics: exploit more information, more parameters
- Online adaptation → better than average case

FPGA Placement

Simulated Annealing (SA) Placement

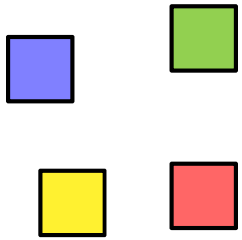
- Modify placement by making ‘moves’
- Accept/Reject move based on:
  - Cost change
  - Temperature (hill climbing)



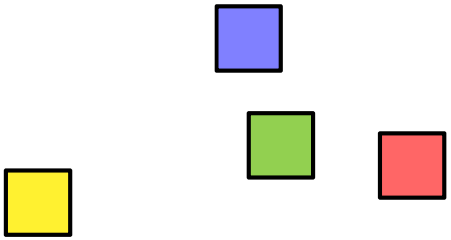


Move Generation

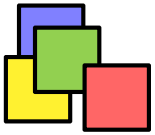
Many possible types of moves!



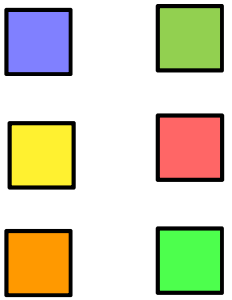
Simple: random swap



Smart: directed move to ‘good’ location (wirelength, timing)



Complex: Analytic



Complex: Assignment

Many considerations:

- Frequencies of different moves
- Situation dependent?
- Move ‘strength’ vs run-time

Treat as RL Problem!

RL Move Generator

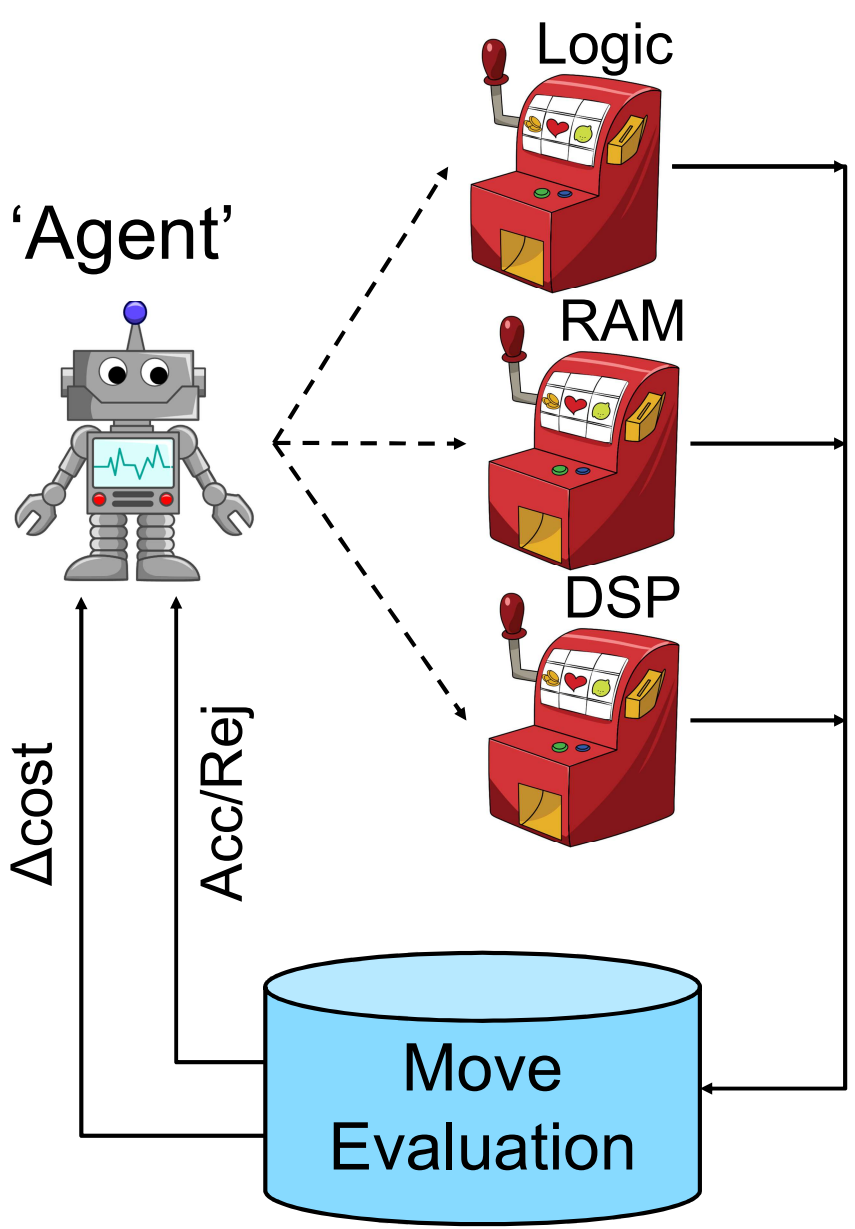
Actions: moved different block types

Reward:

- Accepted: -Δcost
- Rejected: 0

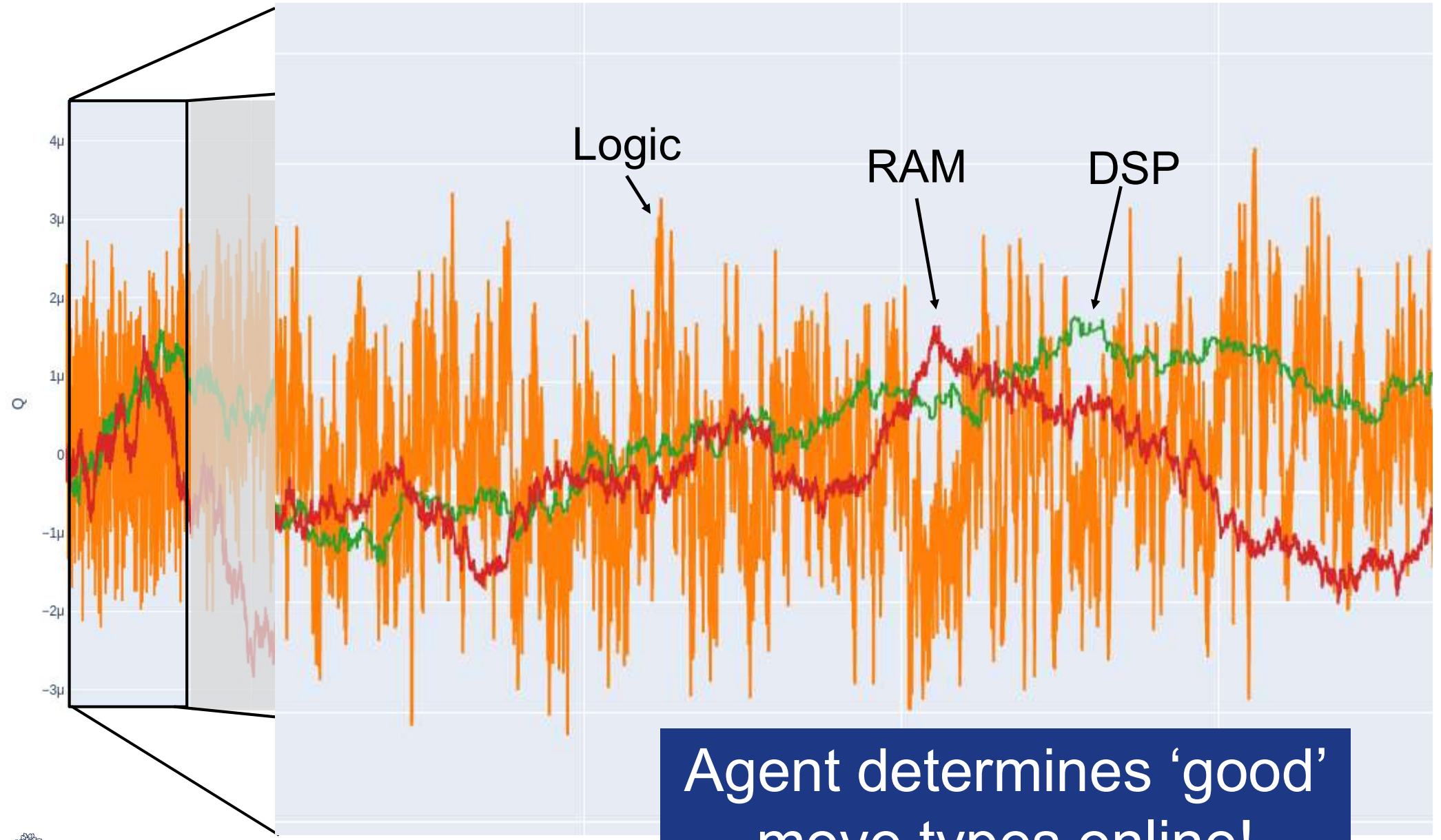
Agent:

- Estimates value of actions
- Selects action to take



Estimating Action Values

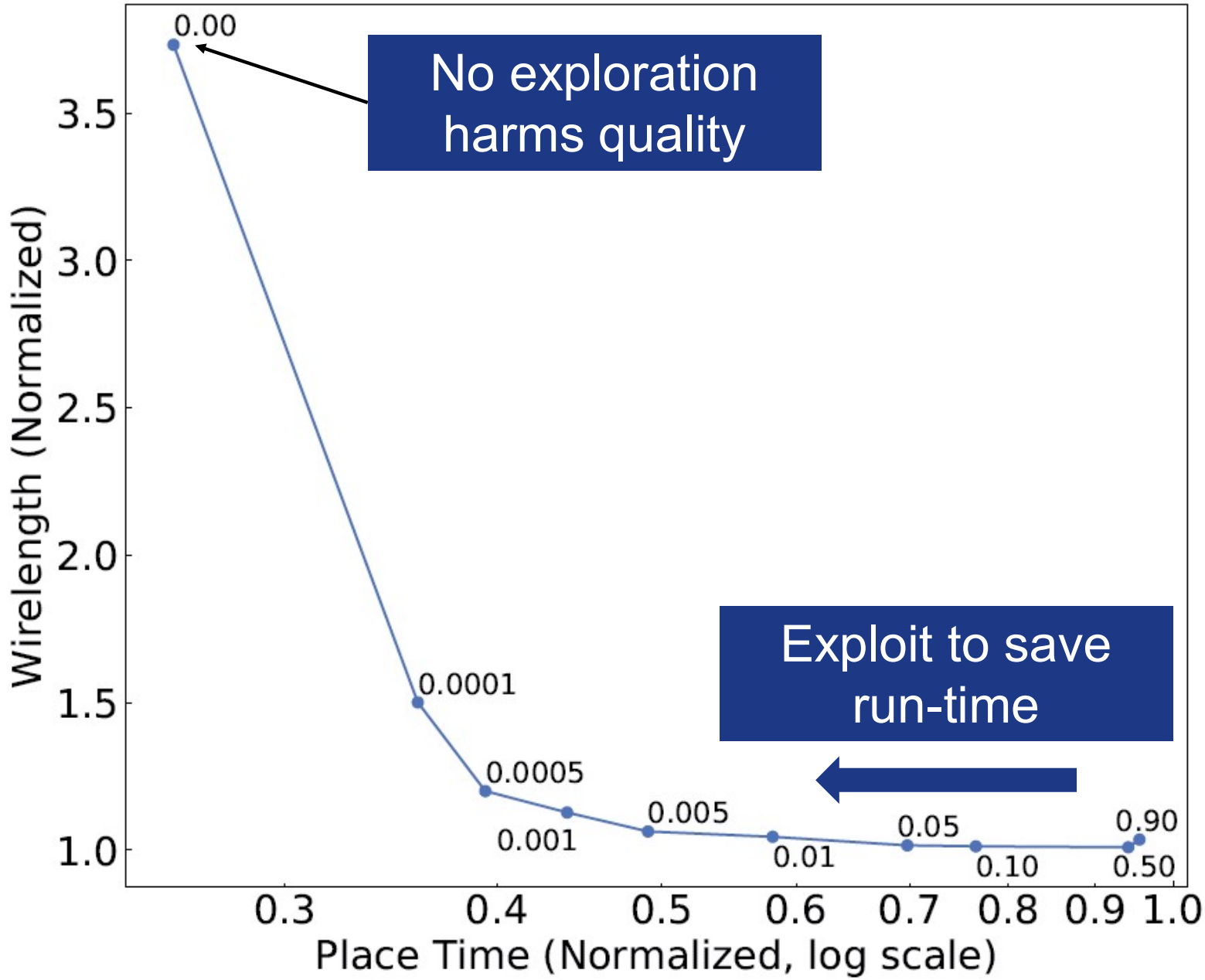
- Values of action are not stationary!



Agent determines ‘good’ move types online!

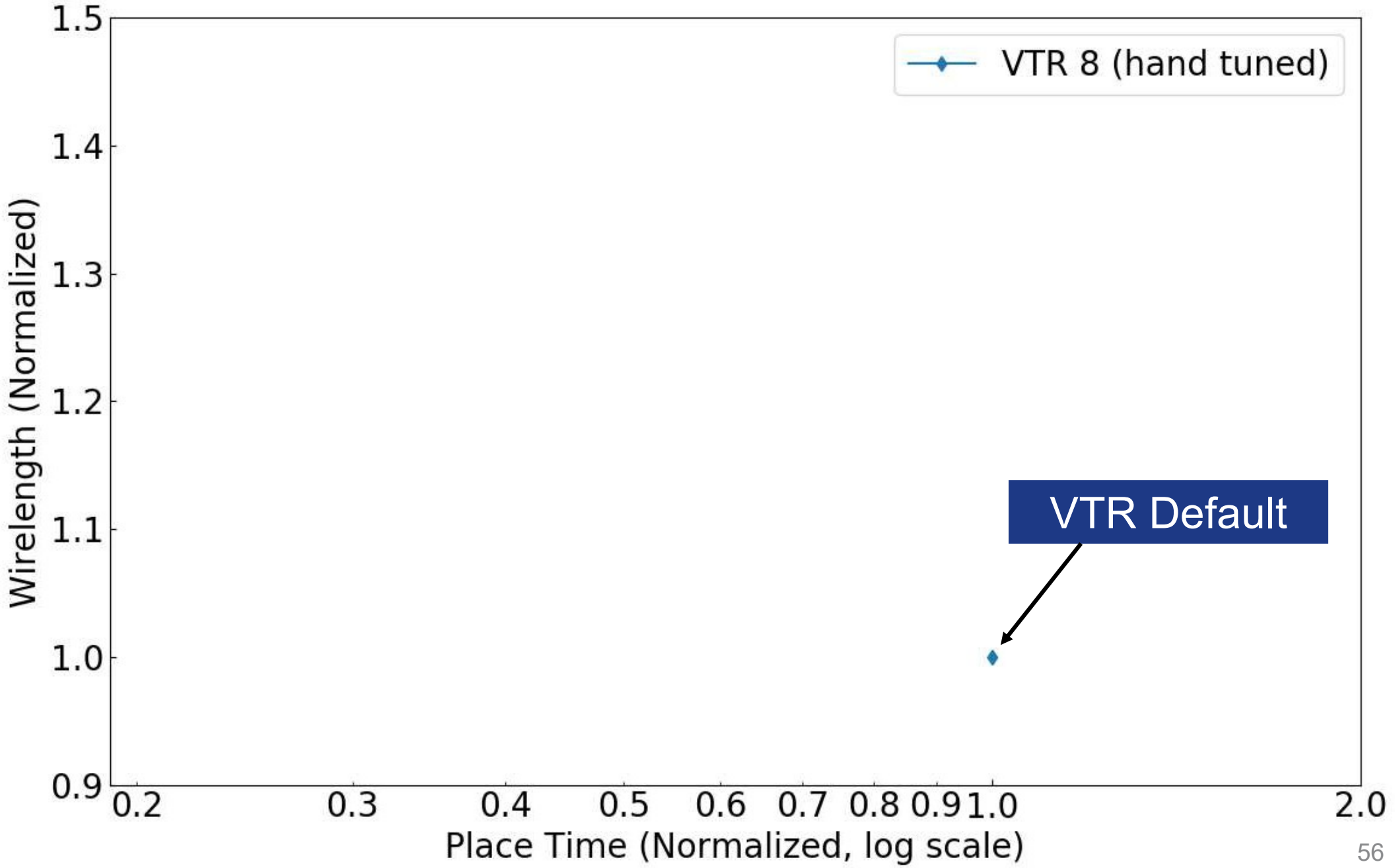
Action Selection: Exploration vs Exploitation

- ε-greedy: Mostly greedy (exploit), occasionally random (explore)
  - ε: fraction of exploratory moves



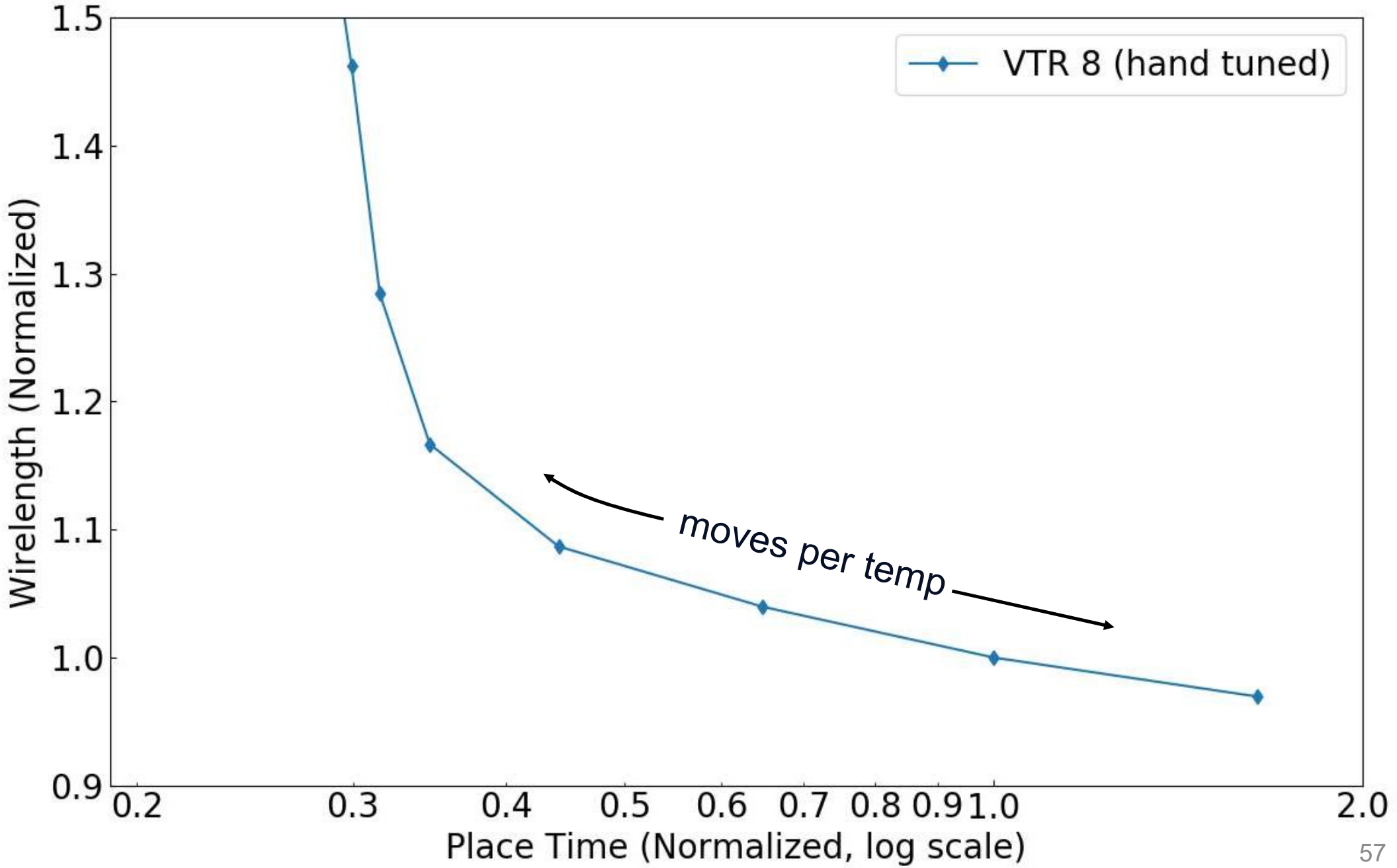
Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds



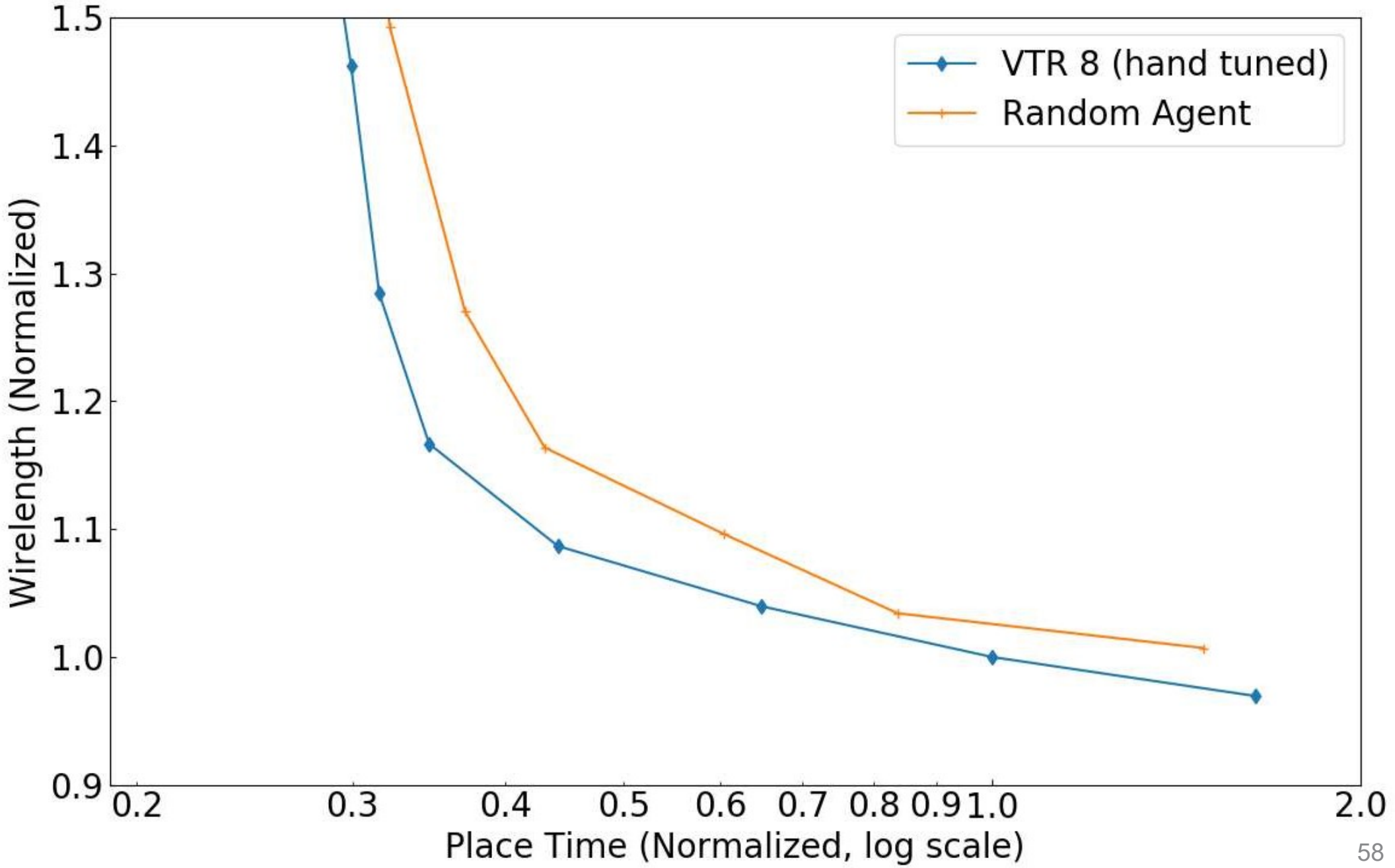
Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds



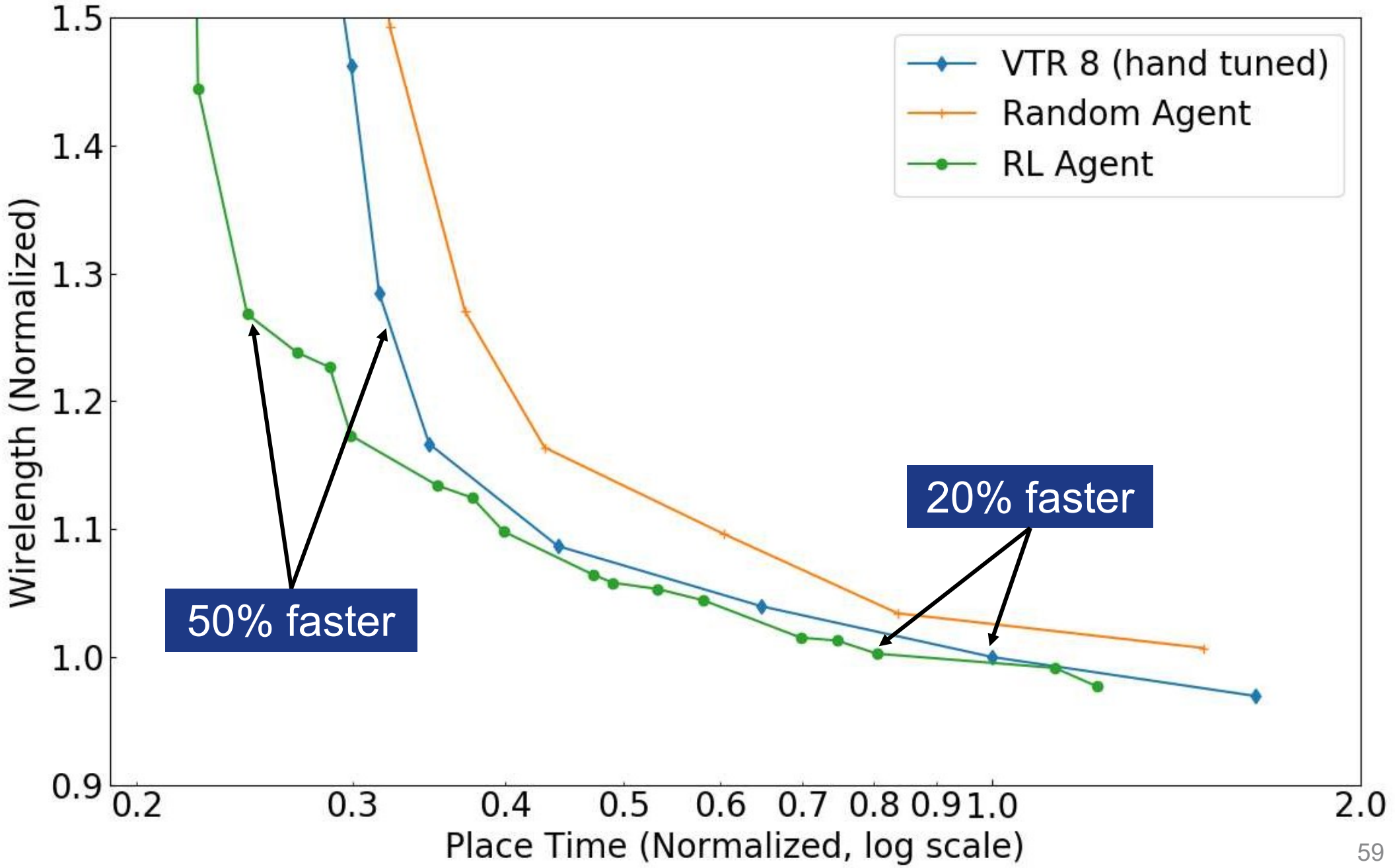
Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds



Quality/Run-time Comparison

- VTR Benchmarks (10K-165K primitives), 3 seeds



Conclusion

- RL-enhanced Simulated Annealing based FPGA Placer
  - RL agent controlled move generator
  - Learns on-line what types of moves are productive
  - Improves run-time/quality trade-offs
    - *Particularly at low run-times*
  - Much more to explore
    - *More diverse placement perturbations / moves*
    - *More powerful agents*

Summary

Summary

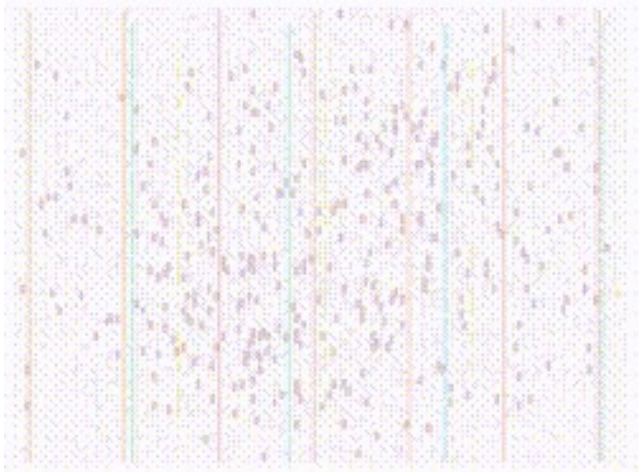
- Open-source, flexible CAD crucial for
  - Evolution/innovation in spatial hardware
  - Enabling CAD research by the community
- VTR
  - Large, very active open-source effort
    - *>12,000 commits, ~100 committers*
  - Linked to other open-source efforts like Symbiflow, OpenFPGA
    - Mix and match your flow
  - Flexible, with good (but can be better!) result quality
  - Contributions welcome!

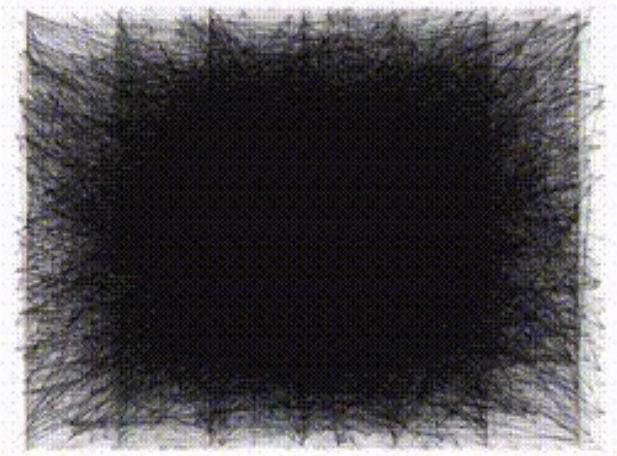
Open source enables innovation across a wider community

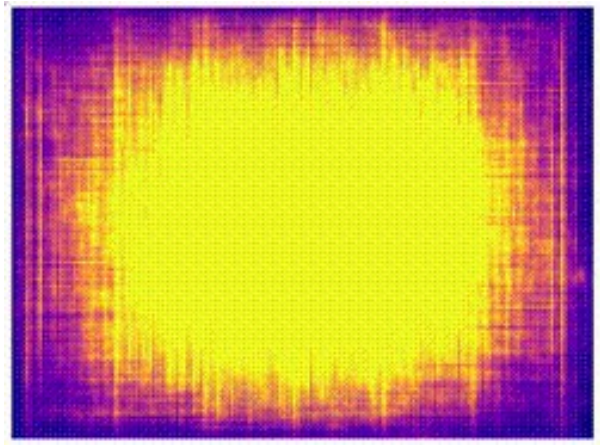
Thanks! | Questions?

Email: vaughn@eecg.utoronto.ca

VTR: verilogtorouting.org







UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING