Reducing FPGA Router Run-Time Through Algorithm and Architecture

Marcel Gort and Jason Anderson
University of Toronto
FPL 2011

Motivation

- FPGA CAD is getting harder and slower for two main reasons:
  - FPGA capacities are increasing.
  - Single-core CPU speed improvements have all but ceased.
- Long CAD times increase development time, thereby increasing design cost.

Reducing Run-time

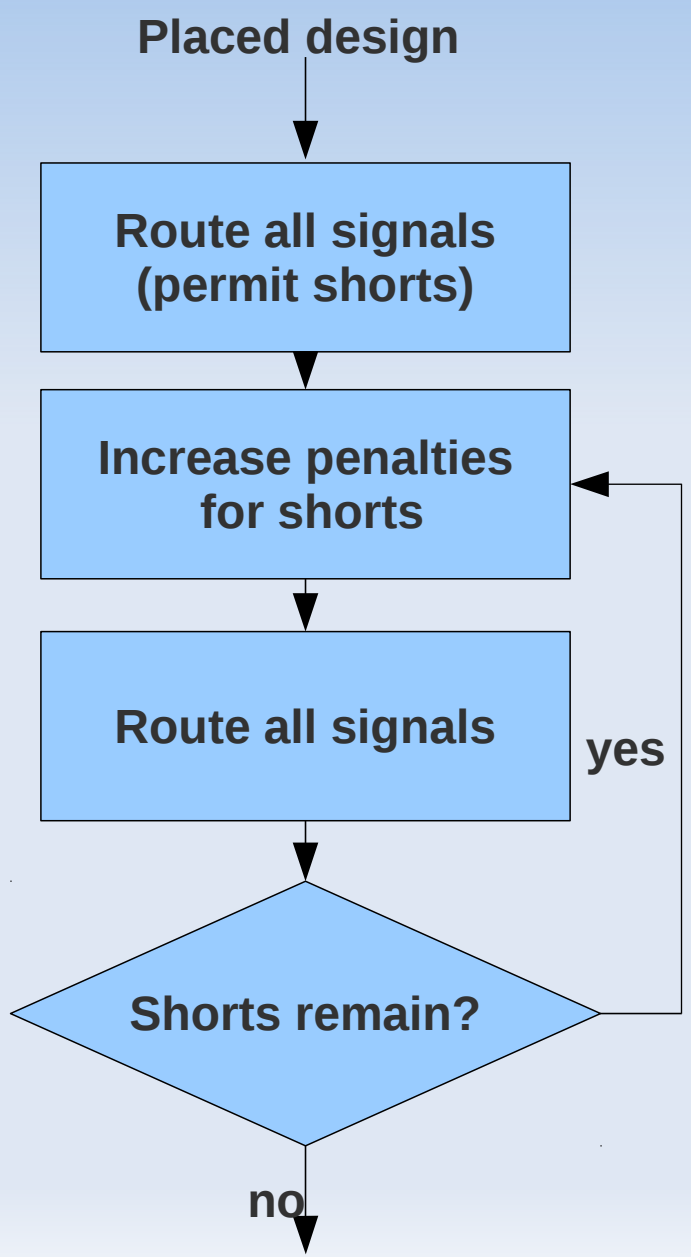
- Prior work mostly improves run-time by:
  - Making algorithmic changes
  - Parallelizing existing algorithms
- Our work uses a combined CAD and FPGA architecture approach to reduce run-time.

PathFinder[1] Routing

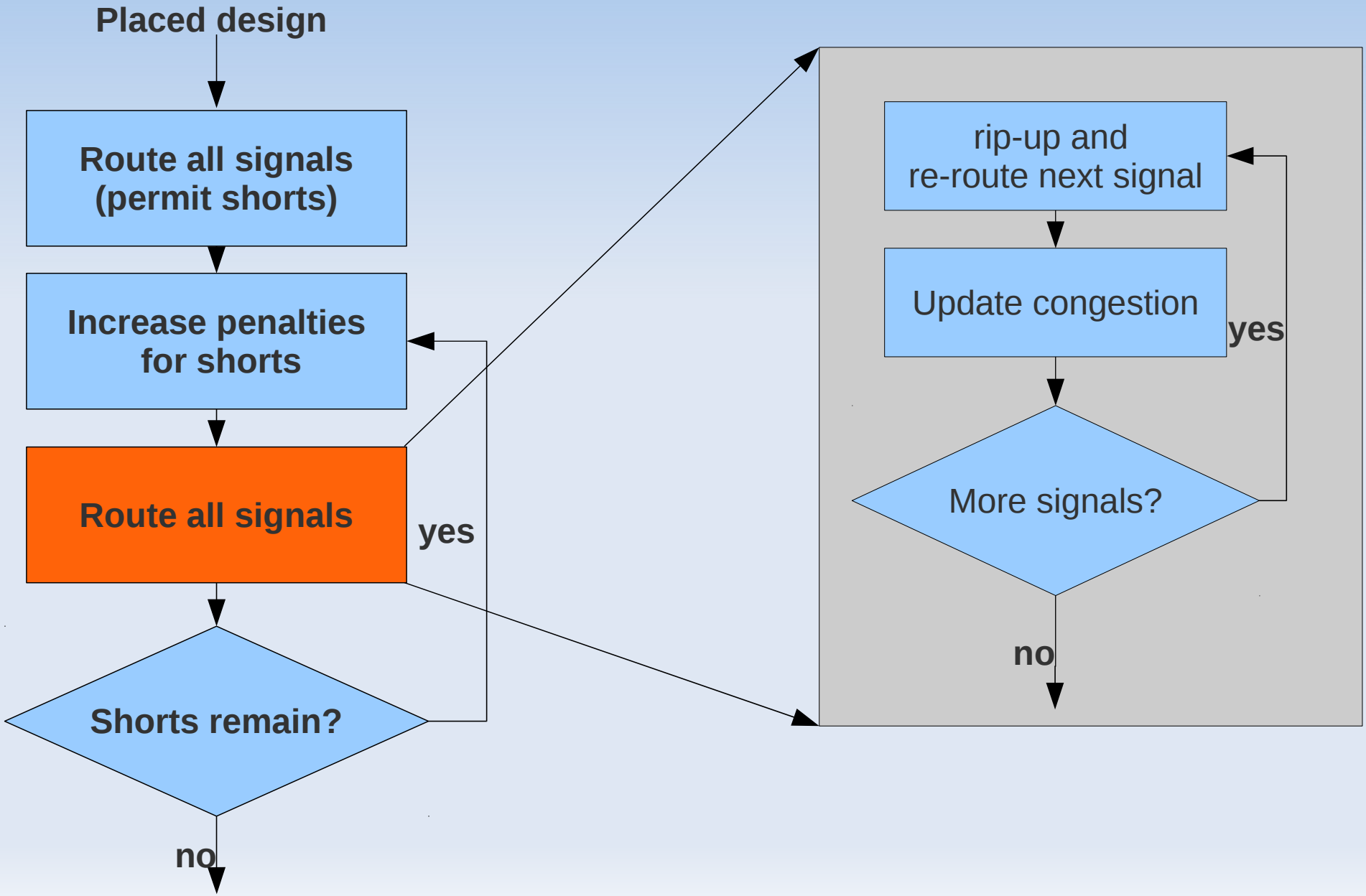
- Forms the basis for both Xilinx and Altera commercial routers, as well as VPR routing.
- Route signals, allowing shorts on routing resource nodes but discouraging their use by making them high cost.
- Gradually eliminate routing resource congestion until no shorts exist.

1. Larry McMurchie, Carl Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," FPGA '95

PathFinder


Placed design
Route all signals (permit shorts)
Increase penalties for shorts
Route all signals
yes
Shorts remain?
no

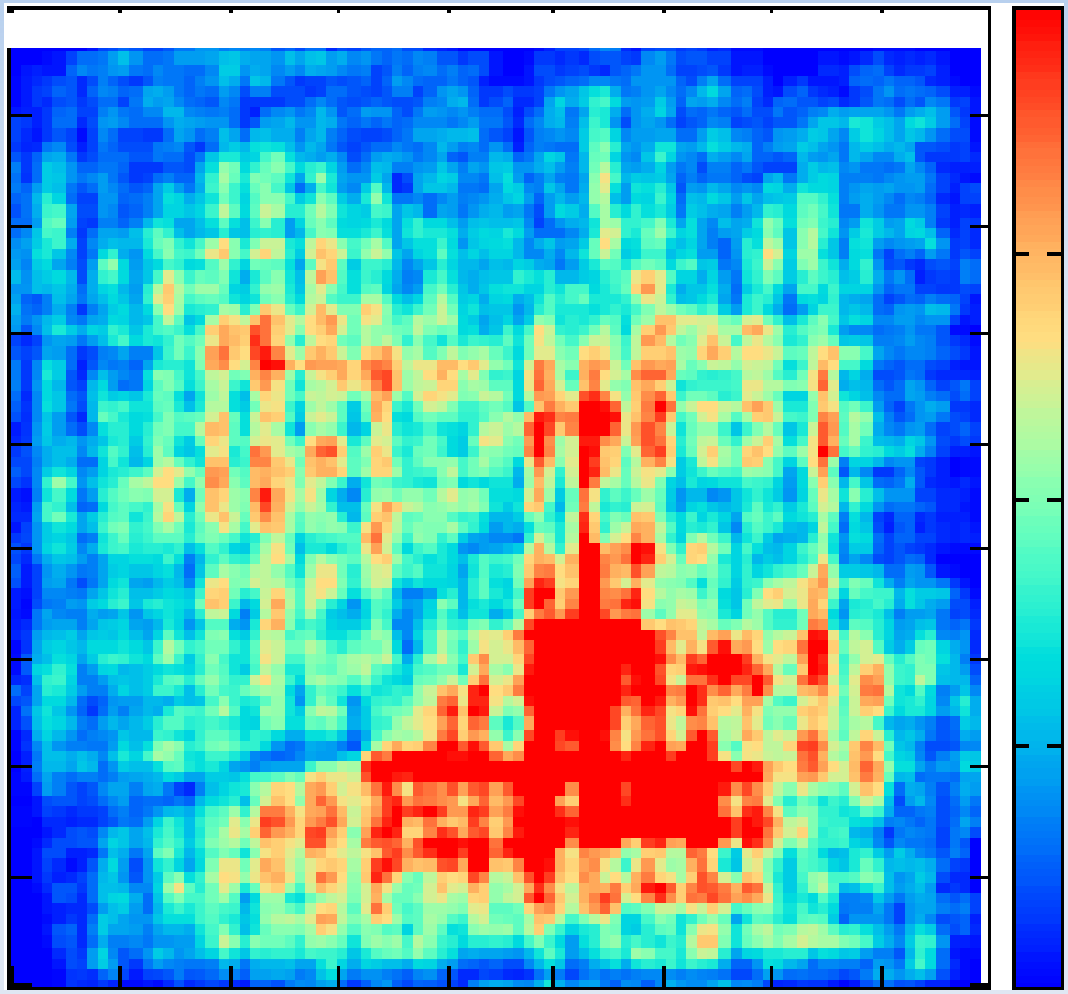
PathFinder


Placed design
Route all signals (permit shorts)
Increase penalties for shorts
Route all signals
yes
Shorts remain?
no
rip-up and re-route next signal
Update congestion
yes
More signals?
no

PathFinder Congestion Resolution

- PathFinder gradually resolves congestion in routing resource nodes, but how fast?

PathFinder Congestion Resolution



congestion_resolution.gif

PathFinder Congestion Resolution

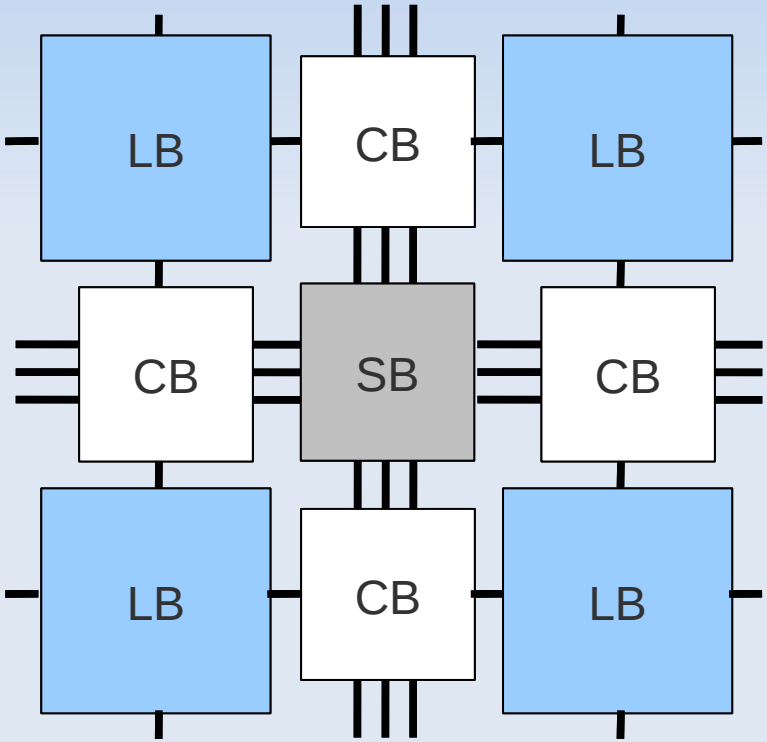
- PathFinder very quickly reduces majority of congestion.
- Spends a large part of run-time "fine-tuning" an almost legal routing solution.
- Heuristic nature means PathFinder is not directly resolving conflicts.

Two-stage Approach

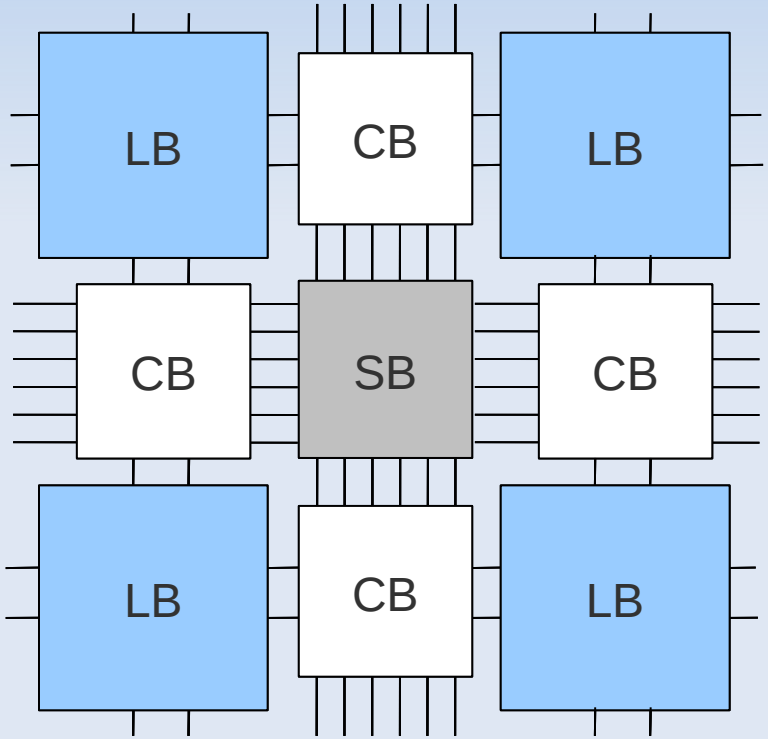
- Use PathFinder to remove the majority of congestion.
  - Run PathFinder on coarsened RR graph.
  - RR nodes represent groups of wire-segments called "wide wires".
  - PathFinder maps signals to wide wires.
- Separate SAT-based embedding step fine-tunes solution to eliminate remaining congestion

RR Graph Coarsening

Coarse Architecture (n=2)

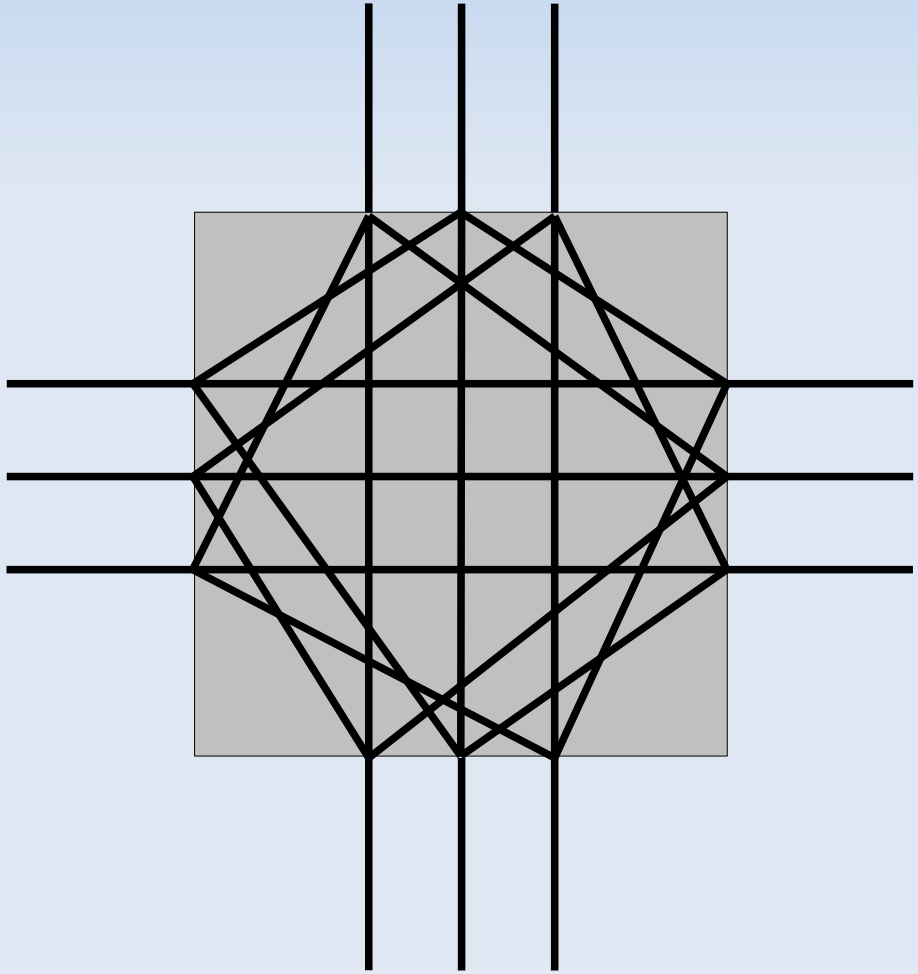


Standard Architecture



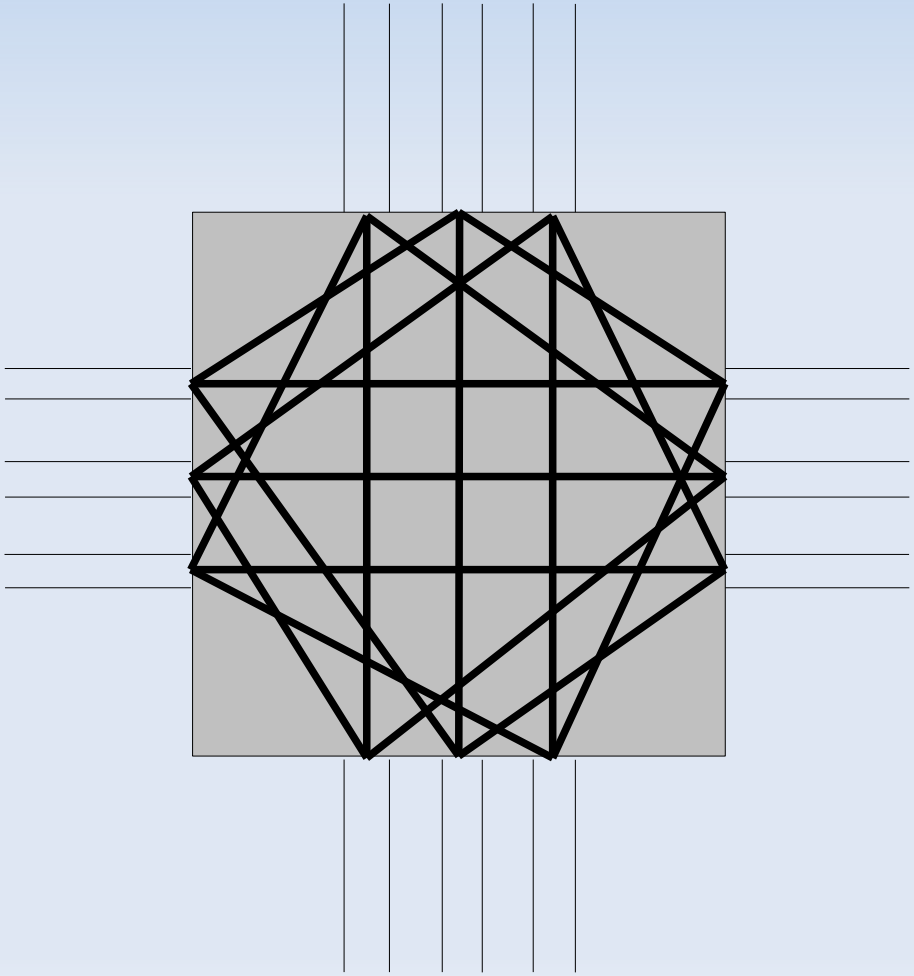
Routing Architecture

Coarse switch block architecture (n=2)



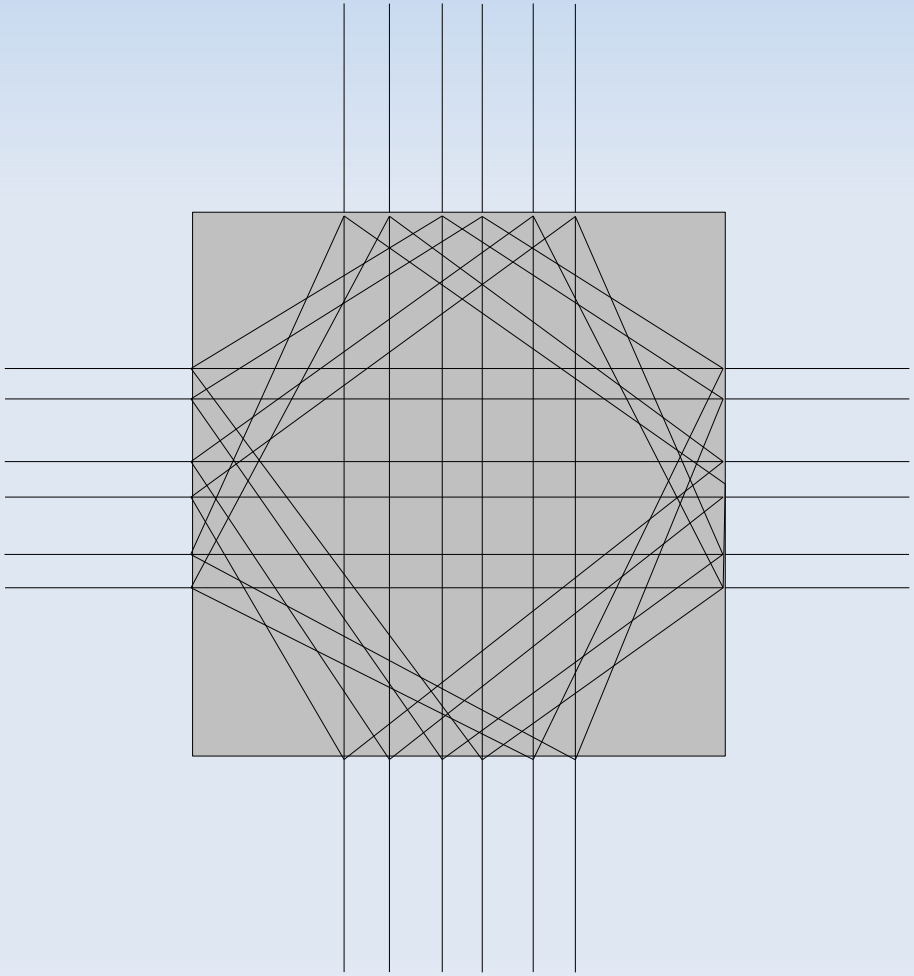
Routing Architecture

Coarse switch block architecture (n=2)

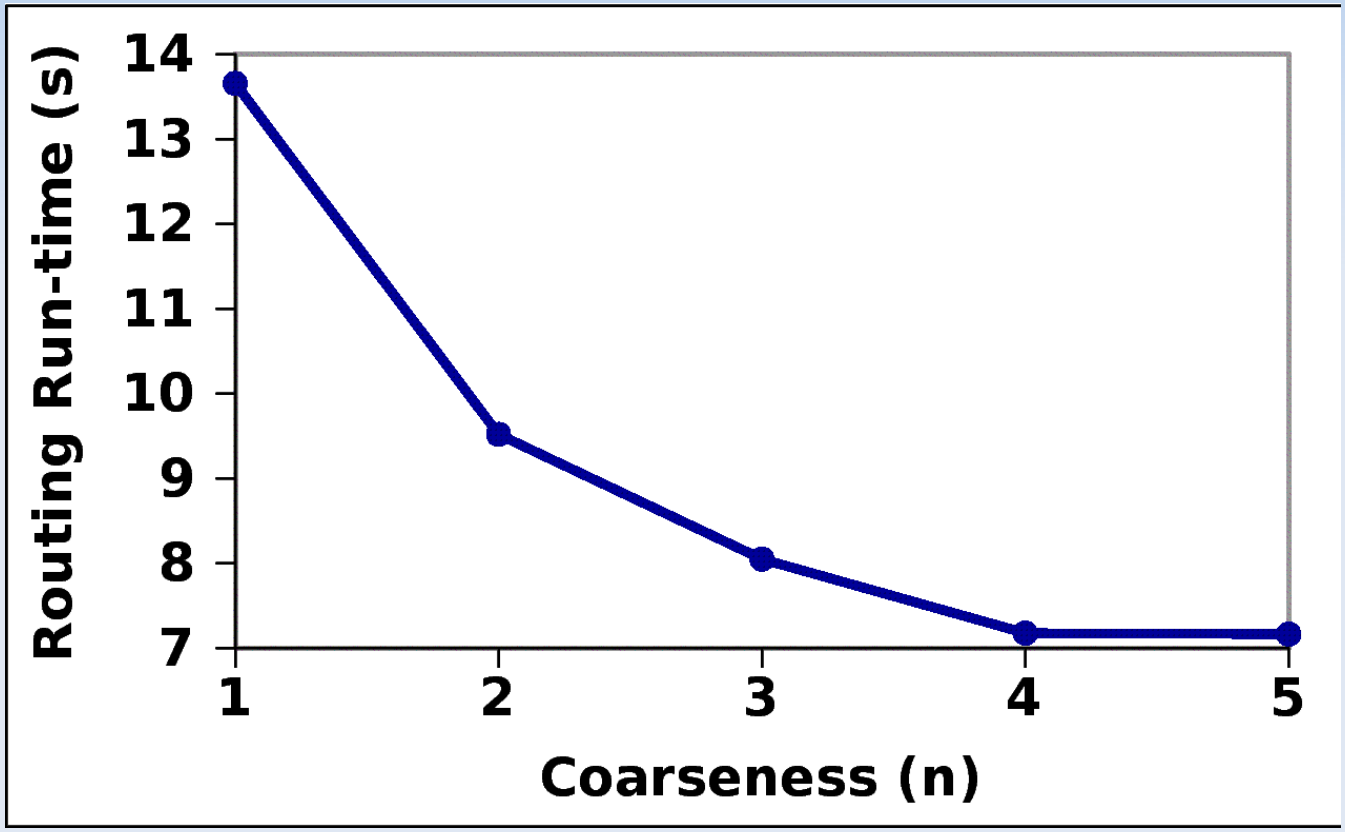


Routing Architecture

Low level switch block architecture (n=2) with only 1-to-1 connections



Coarse Routing Run-time



From n=1 to 5, run-time reduced by 50%

Embedding

- Embedding is the problem of assigning signals in a wide wire to individual wire segments on the FPGA.
- Possible embeddings are constrained to the coarse routes chosen in PathFinder.
- Formulate embedding problem as a SAT instance.

SAT Formulation

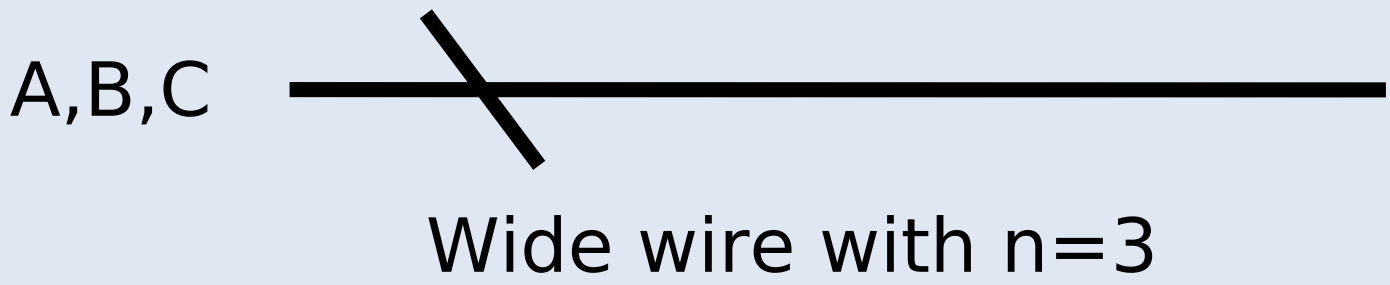
- Boolean formula.
- Formula is SAT if there is an assignment of TRUE or FALSE to boolean variables such that the entire formula evaluates to TRUE.
- SAT F = (A+B)
- UNSAT F = $(A+B)(\overline{A}+\overline{B})(A+\overline{B})(\overline{A}+B)$

SAT Formulation

- Generate set of clauses to represent routing constraints then solve them with MiniSAT.
  - Exclusivity constraints ensure that no shorts exist on any routing track.
  - Liveness constraints ensure that there exists a path from every source to sink.
- Create a variable for each signal to wire-segment assignment.
  - TRUE means the signal uses the wire-segment.

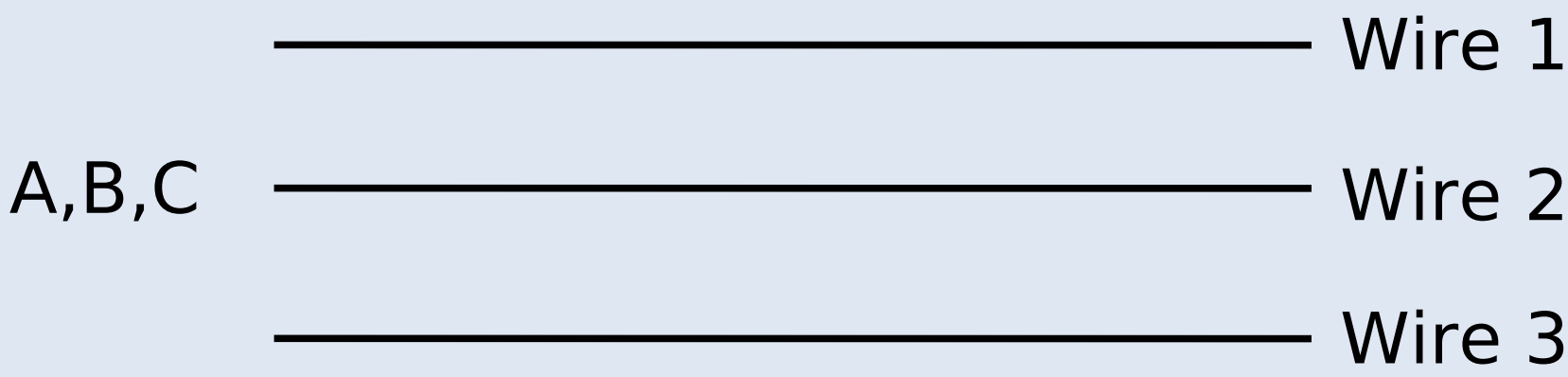
Example: Exclusivity Constraints

- Embed signals A,B,C onto constituent wires of wide wire with n = 3.



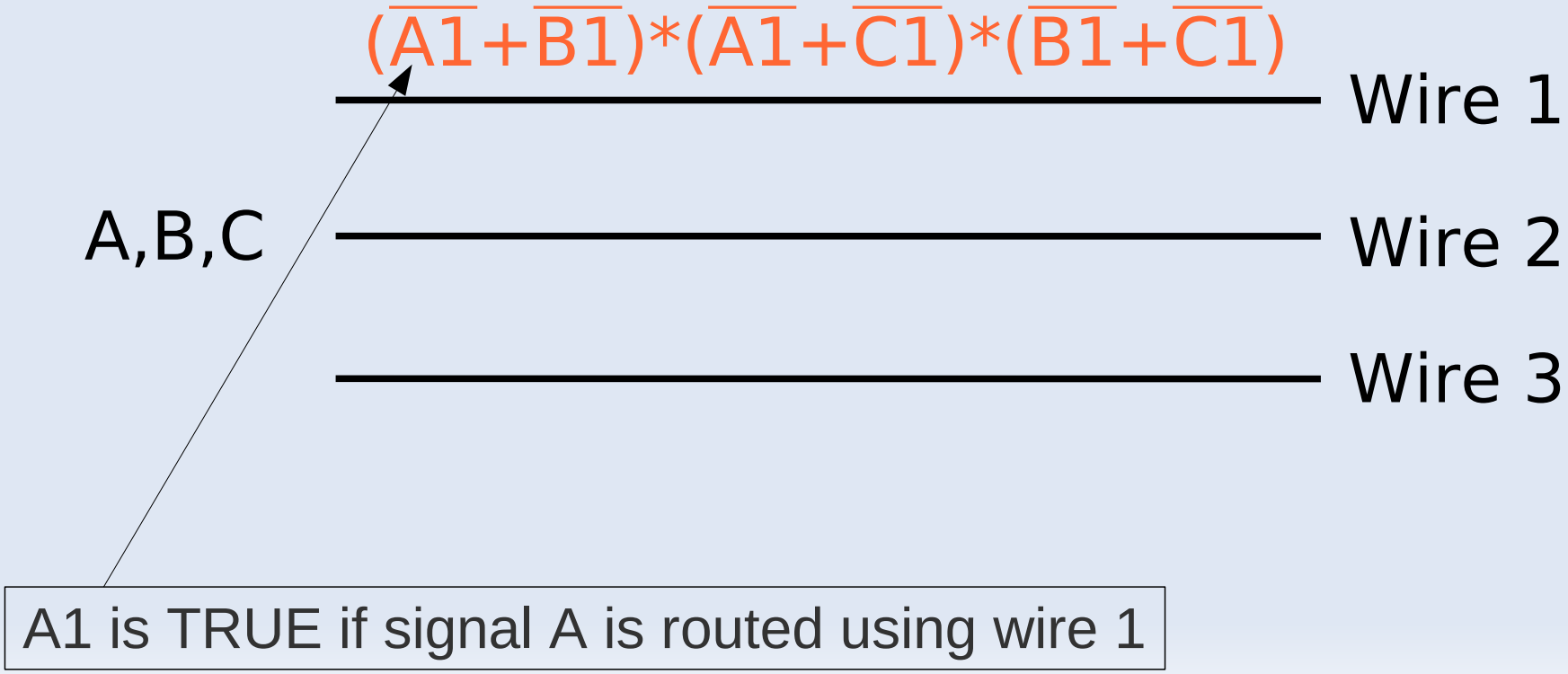
Example: Exclusivity Constraints

- Embed signals A,B,C onto constituent wires of wide wire with n = 3.



Example: Exclusivity Constraints

- Embed signals A,B,C onto constituent wires of wide wire with n = 3.
- If one wire variable is TRUE, all others are FALSE.



Example: Liveness Constraints

- A signal uses at least one constituent wire in each wide-wire through which it passes.

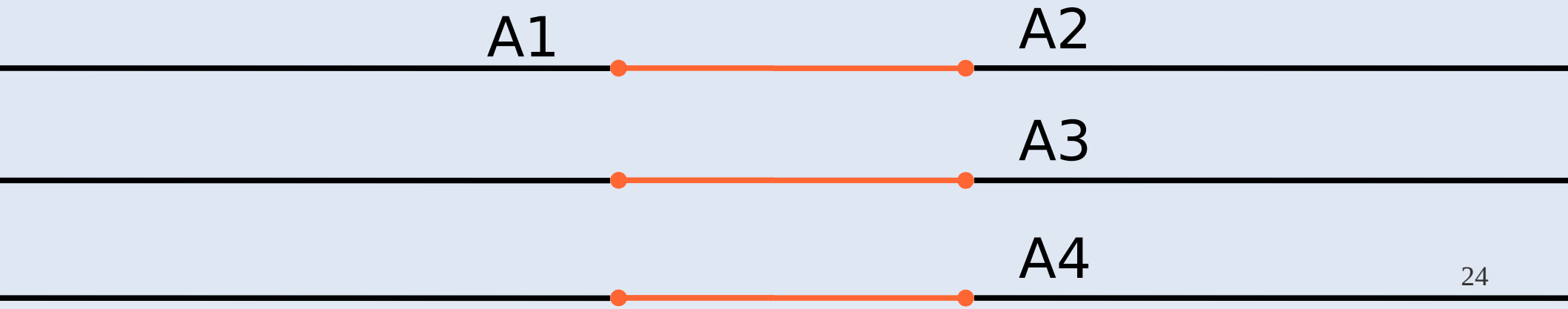
For signal A using wires 1,2,and 3: (A1+A2+A3)

Example: Liveness Constraints

- Only valid connections through switch blocks are made.
  - For a signal, variables representing adjacent wires with no connecting switch cannot both be TRUE.

Example: Liveness Constraints

- Only valid connections through switch blocks are made.
  - For a signal, variables representing adjacent wires with no connecting switch cannot both be TRUE.



Example: Liveness Constraints

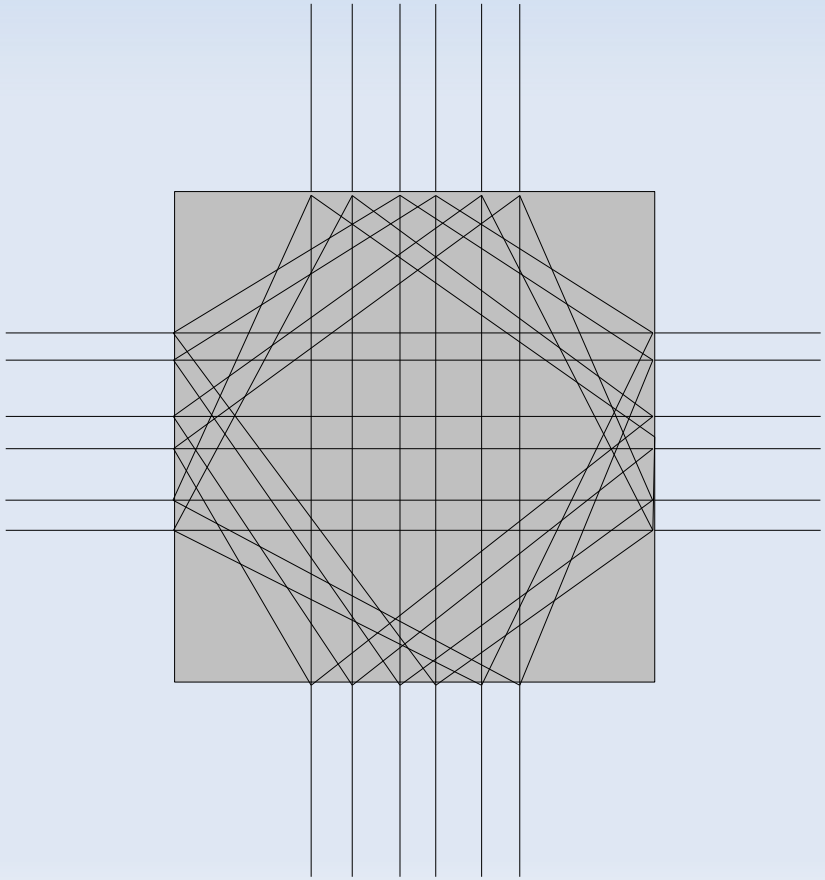
- Only valid connections through switch blocks are made.
  - For a signal, variables representing adjacent wires with no connecting switch cannot both be TRUE.

SAT Constraints

- F = exclusivity * liveness
- If F is SAT, value of variables define a legal routing solution.
- If F is UNSAT, no embedding exists given the routing architecture and the coarse routing.

Architectural Changes

- Using a standard switch block architecture (Fs=3, Wilton topology, 1-to-1 connections), almost all benchmarks result in UNSAT.

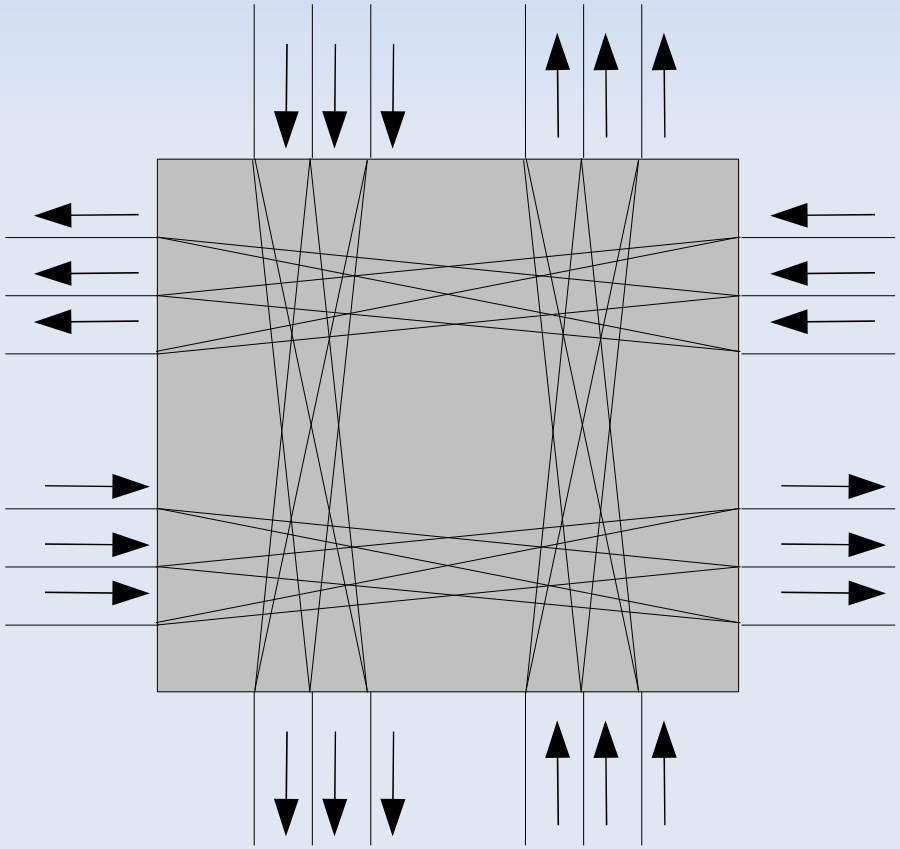


Architectural Changes

- Added extra switches to switch blocks to greatly reduce likelihood of UNSAT.
  - Used same switch block topology across all benchmarks.
- Explored close to 2000 low-level switch block topologies.
- Result: low area overhead switch block topologies with extra switches concentrated in straight connections.
- Further research may reveal interesting ”good” low-level switch block architectures.

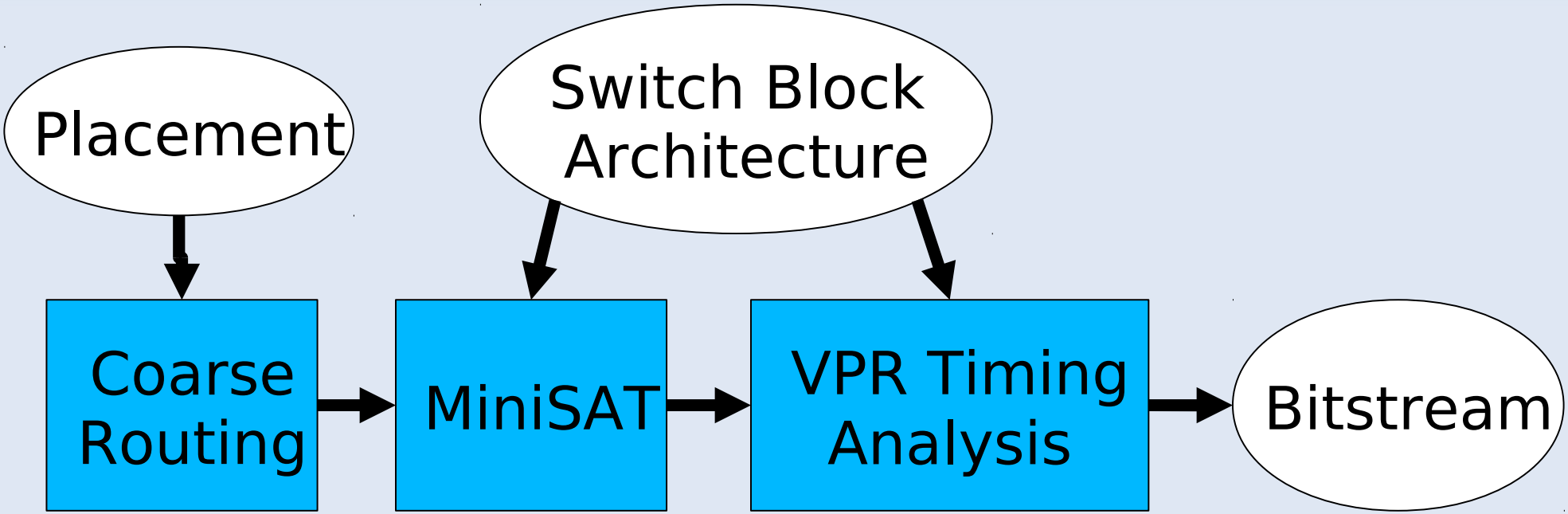
Architectural Changes

n=3 low level switch block connectivity
(excluding 1-to-1 connections)



Experimental Methodology

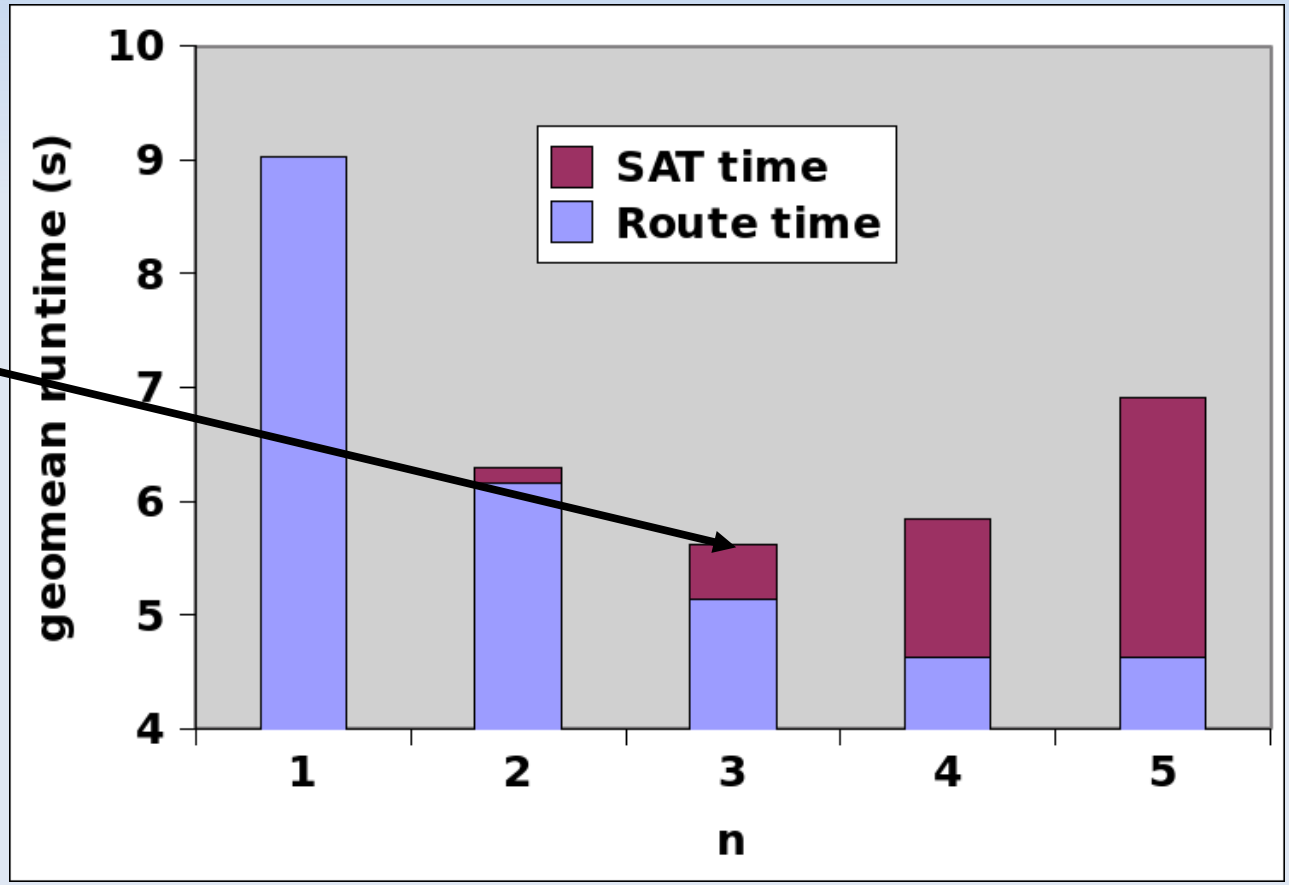
- 16 longest running benchmarks in MCNC20 and VPR 5.0.



Combined Run-time

- n=3 results in 35% improvement in run-time.

3% area overhead



Area Neutral Experiments

1) Ran VPR routing on uncoarsened RR graph *with* increased flexibility switch block architecture.

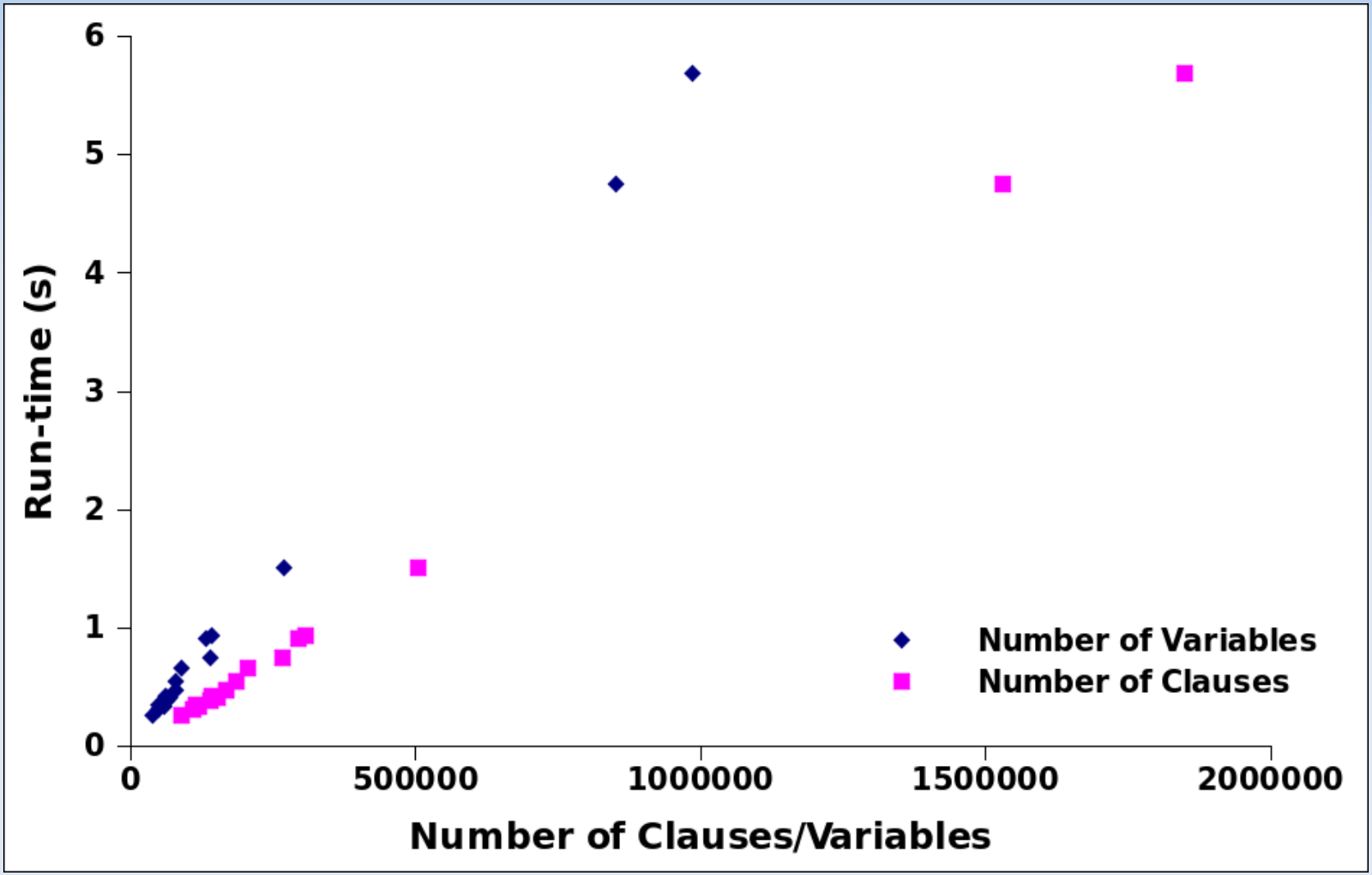
2) Ran VPR routing with a slightly higher channel width so that the area was equal to the area of the increased flexibility switch block architecture.

Area Neutral Experiments

1) Ran VPR routing on uncoarsened RR graph *with* increased flexibility switch block architecture.
   - PathFinder/SAT approach is **35% faster for n=3**

2) Ran VPR routing with a slightly higher channel width so that the area was equal to the area of the increased flexibility switch block architecture.
   - PathFinder/SAT approach is **21% faster for n=3**

SAT Scalability



Conclusions

- 2 stage routing
  - PathFinder-based routing on wide-wires.
  - Embedding onto individual wires as SAT instance.
- 35% run-time reduction with 3% area overhead
- 21% run-time reduction with 0% area overhead
- Considering FPGA architecture and CAD together allows for new run-time reduction strategies.
  - ➔ Open area of research!

Future Work

- More thoroughly explore low level switch block topologies.
- Closer examination of coarse route properties that lead to UNSAT.
  - Can costs in PathFinder be modified to make avoid UNSAT situations?

Questions?

- Marcel Gort and Jason H. Anderson, **Reducing FPGA Router Run-time through Algorithm and Architecture**, FPL 2011