Timing Optimization for Virtual FPGA Configurations

Linus Witschen, Tobias Wiersema, Masood Raeisi Nafchi,
Arne Bockhorn, and Marco Platzner
Computer Engineering Group
Paderborn University

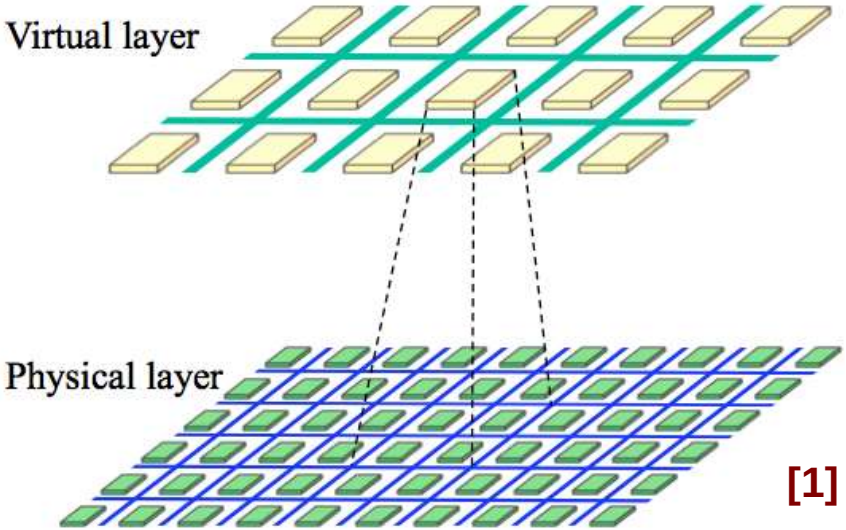
{witschen, wiersema, mraeisi, xoar, platzner}@mail.upb.de

PADERBORN UNIVERSITY
The University for the Information Society

SFB901
ON - THE - FLY COMPUTING

Virtual FPGAs (a.k.a. FPGA Overlays)

- Different abstraction levels
  - Multi-node level
  - Node level
  - Resource level
- Active research field **[2]**
  - Multi-tenancy
  - Resource management
  - Flexibility
  - Isolation
  - Scalability
  - Performance
  - Security
  - Resilience
  - Programmer's productivity
- Fine-grained virtual FPGAs
  - Very high virtualization costs (area and delay)
  - Area expansion improved in last decades (> 100x ⟶ ≤ 40x)
  - Delay reduction only reduced as byproduct (6x ⟶ ~ 5x)



[1]

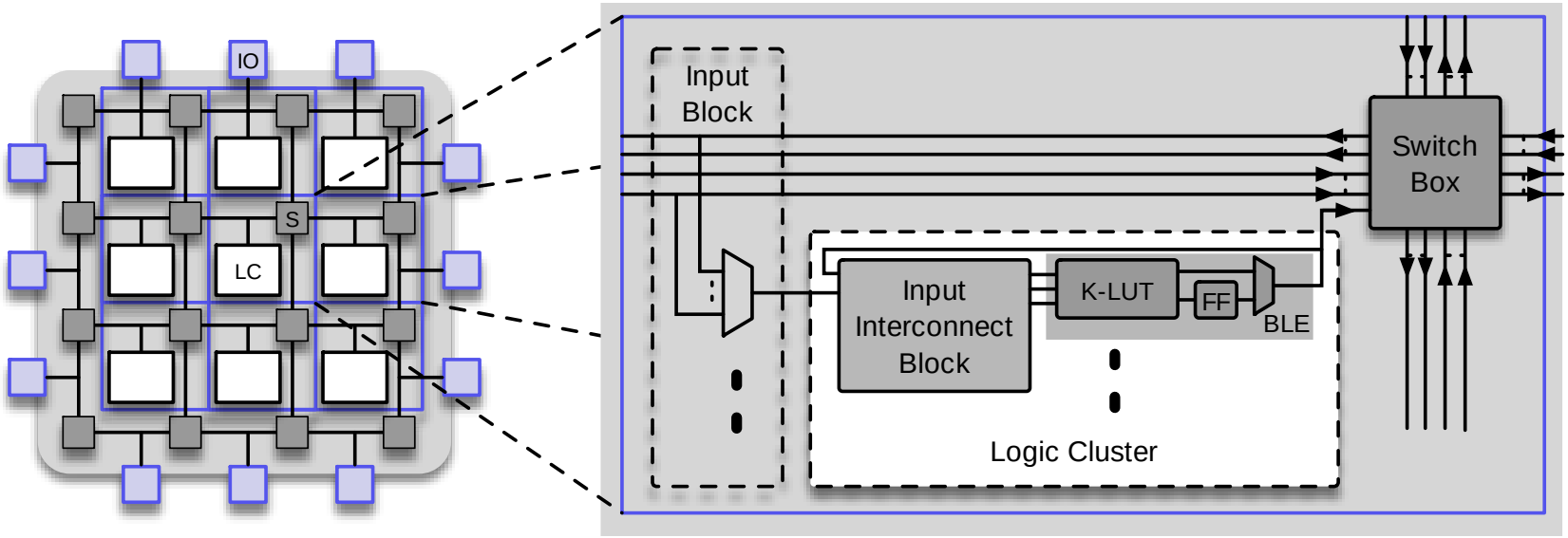
[1] Lagadec et al.: *Placing, routing, and editing virtual FPGAs*. FPL 2001.
[2] Vaishnav et al.: *A survey on FPGA virtualization*. FPL 2018.

Outline

- Background: Virtual FPGA ZUMA
- Physical Timing-driven Virtual Routing
- Physical Design Optimization Through Floorplanning
- Experimental Results
- Summary and outlook

Background: Virtual FPGA ZUMA

- Open-source resource-level virtual FPGA by Brant & Lemieux **[3]**
- Tiled island-style layout
  - IO pads
  - Extra-cluster routing resources
  - Logic clusters
    - *N* Basic Logic Elements (*K*-LUT + FF)
    - *I* cluster inputs
    - Intra-cluster routing resources

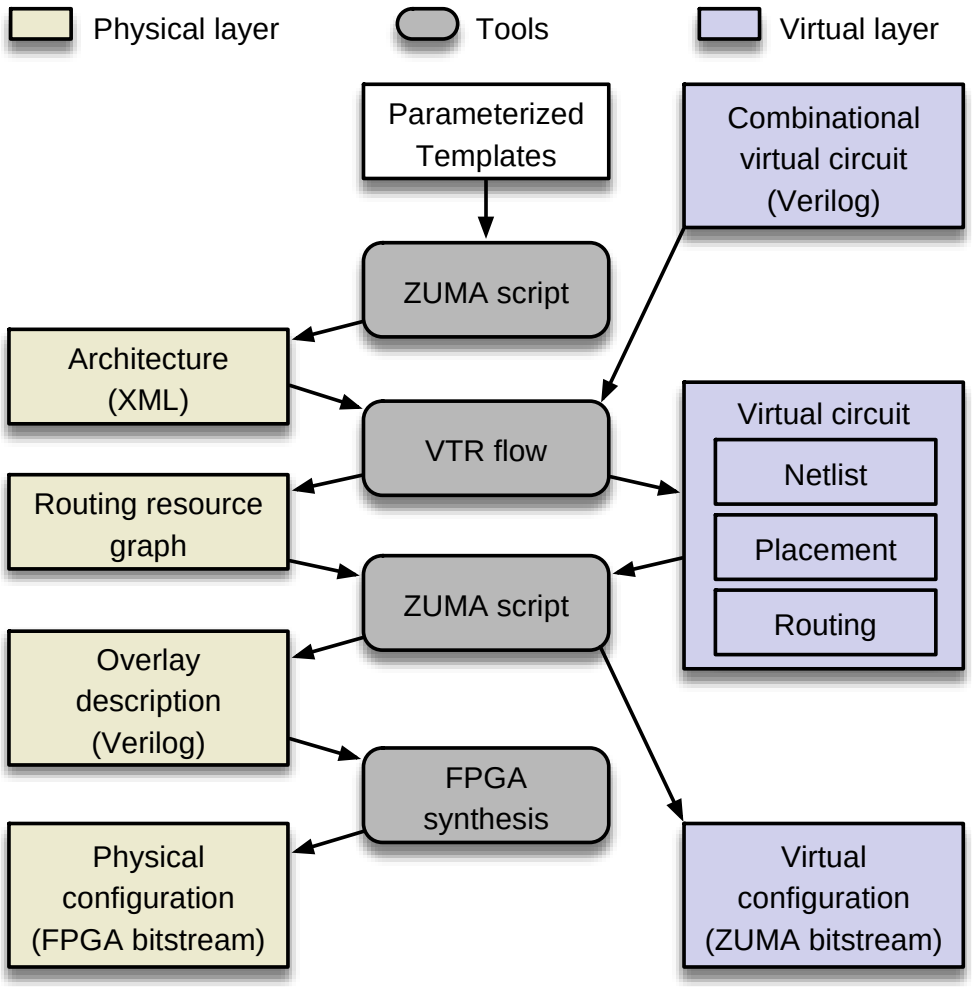


[3] Brant & Lemieux: *ZUMA: An open FPGA overlay architecture*. FCCM 2012.

ZUMA Synthesis Flow

- Intertwined physical and virtual synthesis
  - Based on the VTR (Verilog-To-Routing) flow
  - Tile instantiation
  - *Overlay Description Graph*
    - Based on RRG
    - k_{host}-feasible routing
    - Intra-cluster routing

- Result
  - One host FPGA bitstream describing the virtual fabric
  - One ZUMA bitstream describing the virtual circuit



PHYSICAL TIMING-DRIVEN VIRTUAL ROUTING

Physical Timing-driven Virtual Routing

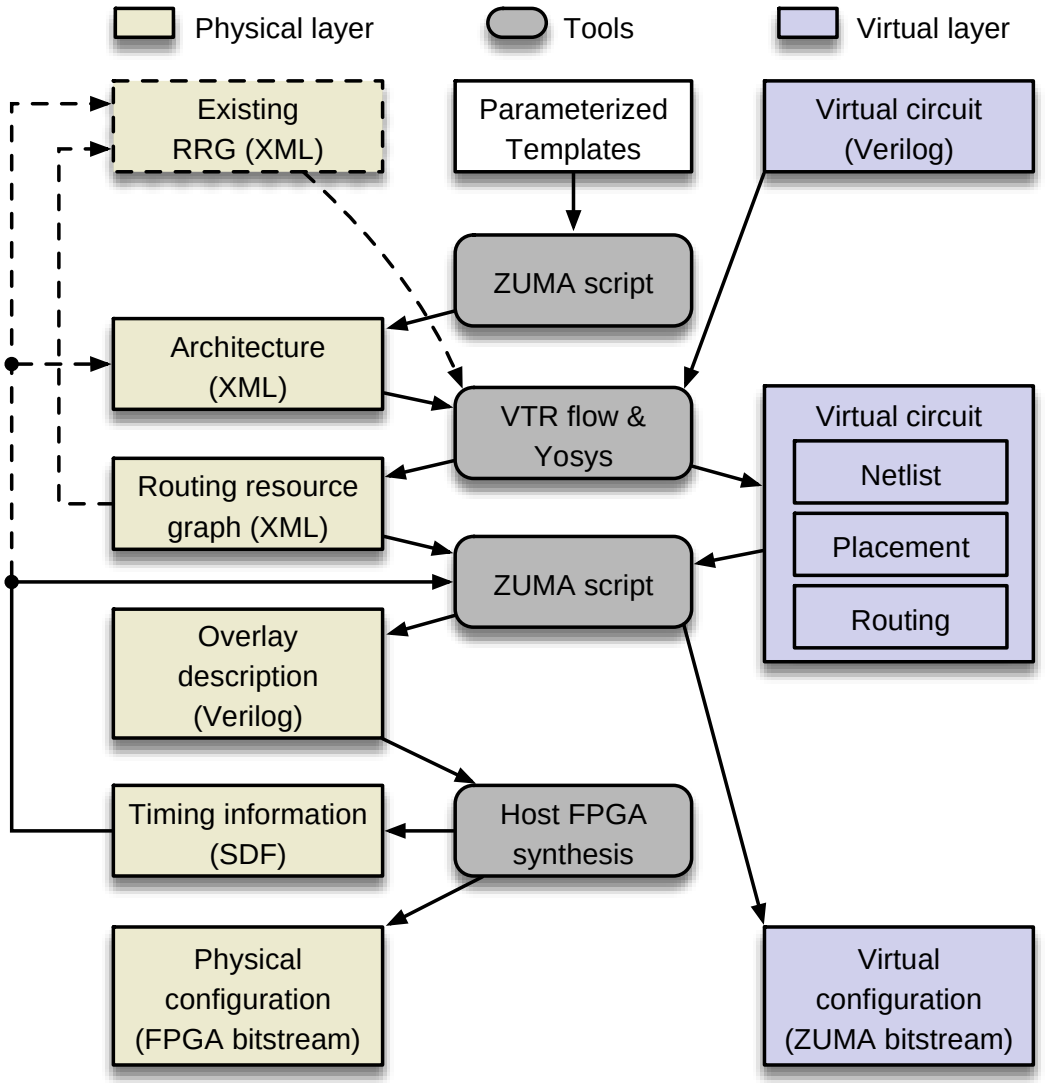
- Considerations and Challenges
  - Portability vs. performance trade-off
  - Generic tiles vs. individual resources
  - Required tool support on the virtual side ⟶  VTR 8

- Virtual synthesis is typically area-driven
  - Physical timing information abstracted away
  - Previous work **[4]** introduced accurate timing estimation
    - Back-annotated timing information
    - Identify physically critical virtual path to find f_{max}

[4] Wiersema et al.: *An architecture and design tool flow for embedding a virtual FPGA into a rSoC*. Comput. Electr. Eng. 55.

Timing-driven Synthesis Flow

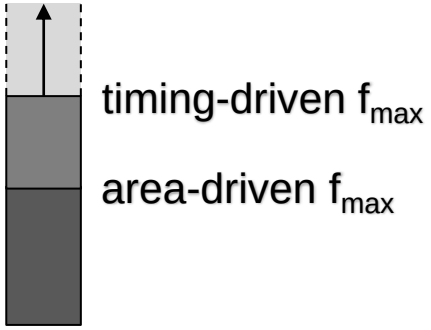
- Back-annotate timing information (second run)
  - Aggregate from fine-grained ODG
  - Extra-LC ⟶ RRG
  - Intra-LC ⟶ Architecture
  - Individualize arch elements in all tiles
- Run VTR 8 in timing-driven mode
- Caveat: Individual LCs
  - Modeling limitation of VTR
  - Timing information stored in LCs, not tiles
  - Placer's effectiveness severely limited



PHYSICAL DESIGN OPTIMIZATION THROUGH FLOORPLANNING

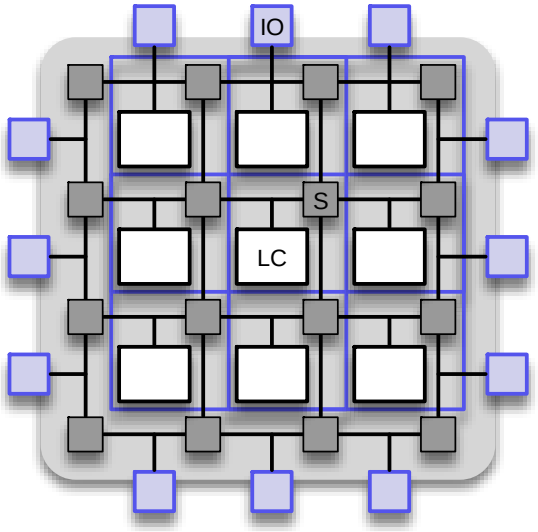
Physical Design Optimization Through Floorplanning

- Timing-driven routing exploits available f_{max} optimization headroom
  - Size of headroom dictated by host FPGA‘s vendor tools

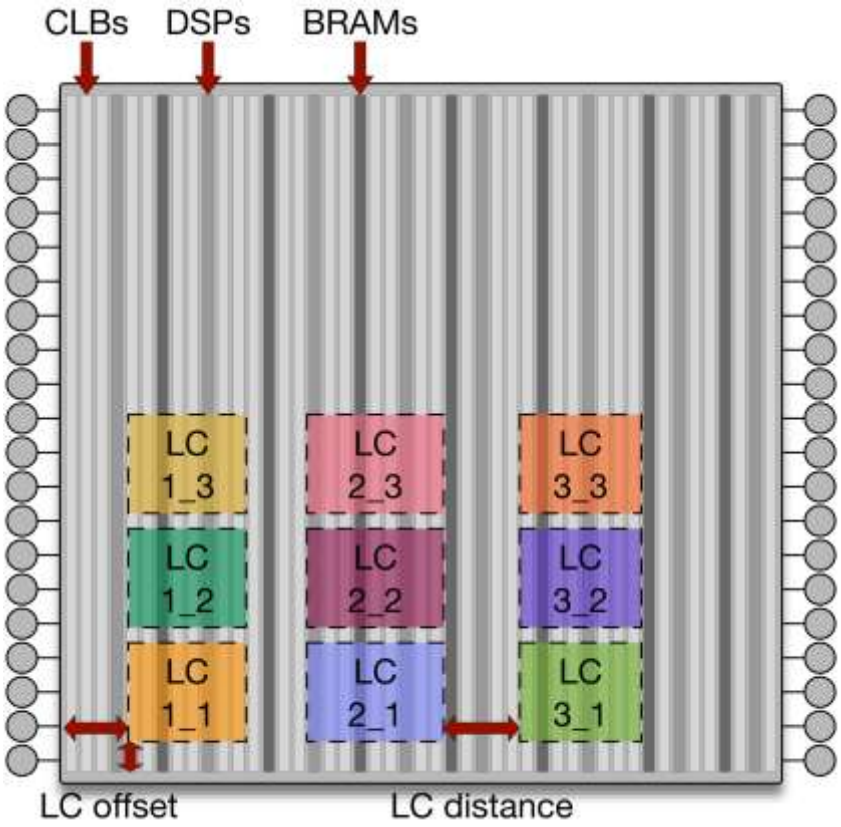


- Physical synthesis suboptimal
  - Combinational loops
  - Tiles
  - Length of critical path vs. balanced virtual wires

- Optimize host FPGA synthesis to improve headroom
  - Preserve vFPGA’s regularity
  - Hierarchical fabric designs
    - Guide placement
    - Enable out-of-context builds
  - Floorplanning
    - Coarse: Plan complete LCs
    - Fine: Plan each LC’s IIB and BLEs
  - Automatable

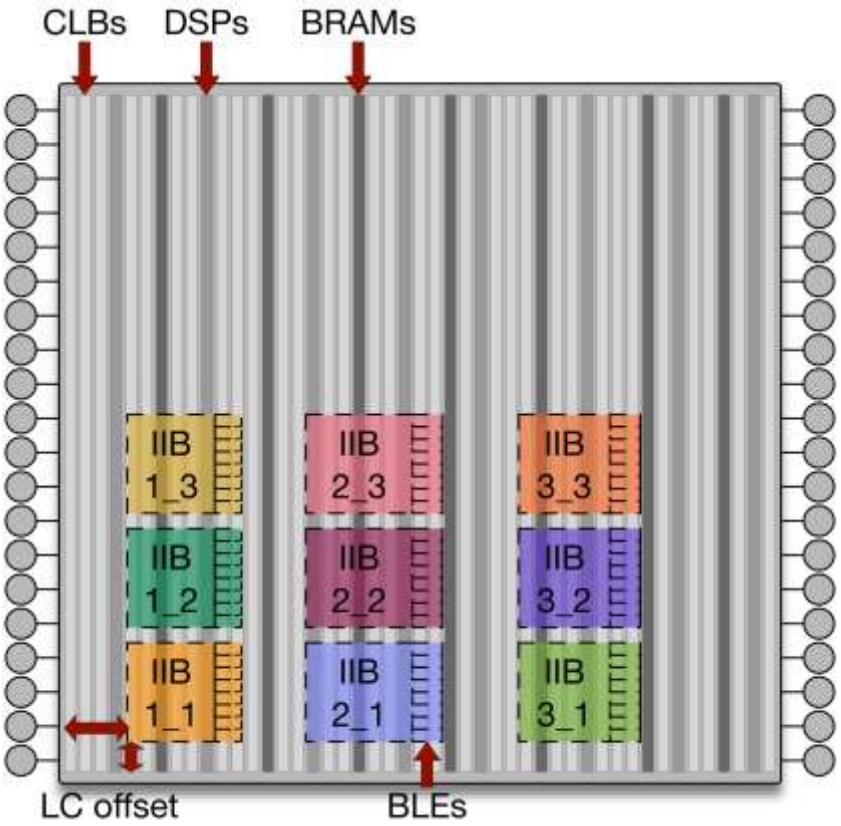


Floorplanning Strategy



- Coarse strategy
  - One Pblock per LC
  - Automatically placed
- Parameter: Pblock shape & size
  - Smaller ⟶ less delay
  - Minimum size
  - Host FPGA columns
- Parameter: Gap size
  - Smaller gaps ⟶ less delay
  - Implemented in gaps
    - Virtual IOs
    - Extra-LC routing resources
    - Configuration distribution

Floorplanning and Synthesis Strategy



- Fine strategy
  - One Pblock per IIB / BLE
  - Trade-offs as for coarse strategy
- Pblock minimum size
  - Determined automatically
  - Module pre-synthesis
  - Check resource requirements
- Synthesis strategies
  - Global (*glb*)
    - Entire design at once
  - Out-of-context (*ooc*)
    - Each module individually
    - Entire design as stitching run
    - Longer runtimes (2.3x up to 18x)

Outline

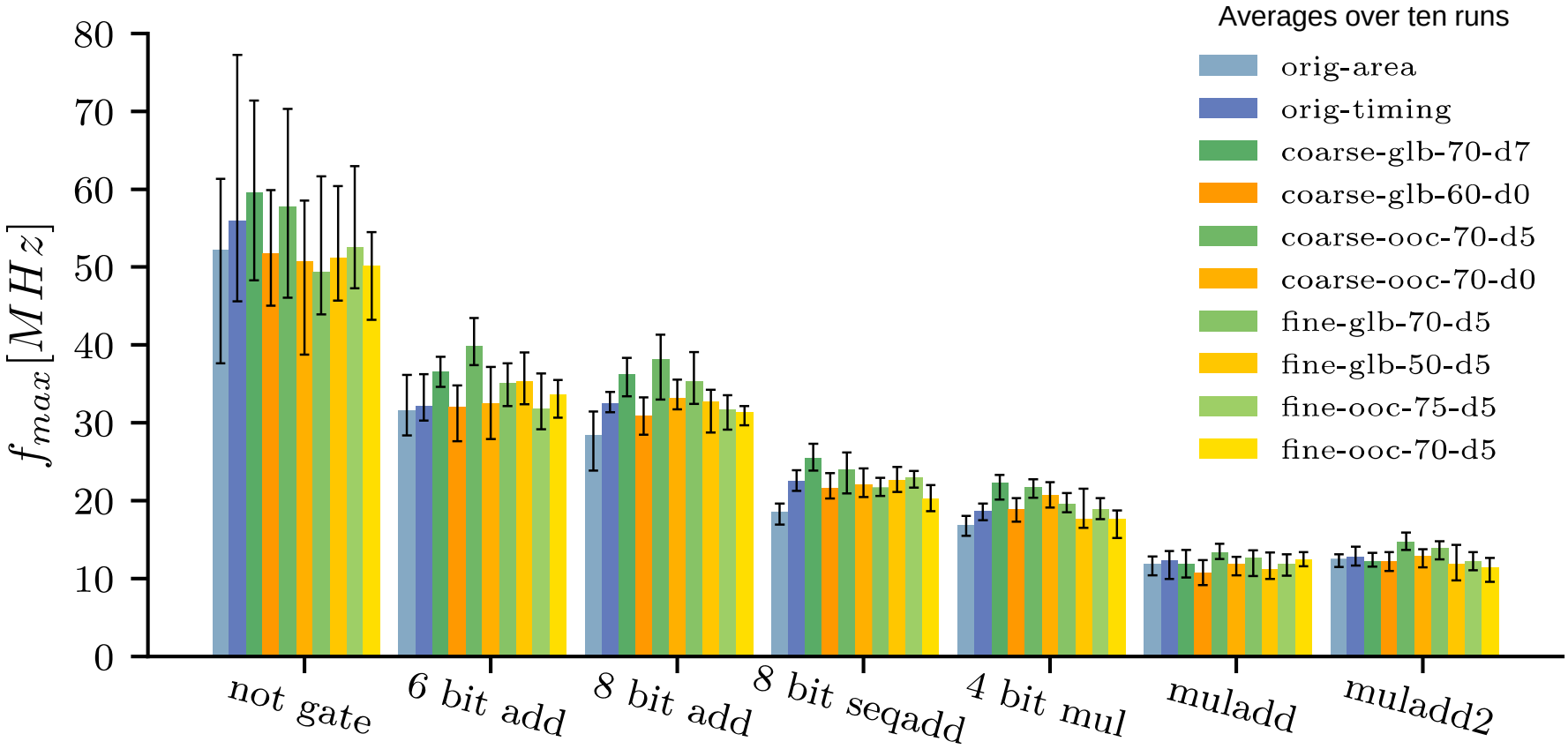
- Background: Virtual FPGA ZUMA
- Physical Timing-driven Virtual Routing
- Physical Design Optimization Through Floorplanning
- Experimental Results
- Summary and outlook

Experimental Results

- Baseline
  - Area-driven VTR
  - Monolithic ZUMA fabric
  - Global synthesis strategy
- Cases
  - Timing-driven VTR
  - Sixteen variations of mixing…
    - coarse or fine floorplanning strategies
    - global or out-of-context synthesis strategies
    - variations of Pblock shapes (0: tall & slim, 50: squares, 100: short & fat)
    - variations of gap sizes between Pblocks (d0: none, d7: large)
- Seven benchmarks on 3×3 and 5×5 vFPGAs
  - Best (worst) performing examples in greens (oranges)
  - Averages of ten runs, span of results indicated by error bars
  - Benchmark circuits sorted by utilization

3×3 Virtual FPGA

- Timing-driven vs. area-driven: -4% up to 32%, avg 9%
- Best overall strategy
  - *coarse-ooc* with larger gaps (≥d5)
  - 17.5% improvement over baseline on average



Summary

- Physical Timing-driven Virtual Routing
  - Reduces portability
  - Increases performance
  - Moderate gains; current VTR 8 modeling impedes utility
- Physical Design Optimization Through Floorplanning
  - Synthesis times prolong significantly
  - Benefit depends on virtual device utilization (lower is better)
  - Trade-offs require careful tuning (Pblock shapes, gap sizes)
  - Correct strategy mix can improve between 20% to 30% over timing-driven mode
- Combination
  - Allows for a wider range of circuits on resource-level virtual FPGAs
  - Can be fully automated with the *coarse-ooc* strategy
- Outlook
  - Unlock even more headroom for automation – at least 30% still untapped

Thank you for your attention. Questions?

5×5 Virtual FPGA

- Lower virtual utilization ⟶ *orig-timing* less competitive
- More virtual IOs, routing and configuration resources
  - Utilization of gaps between Pblocks higher
  - More complex interface ⟶ longer routes ⟶ lower frequencies
  - Placer limitation more prominent in larger vFPGAs