Cycle-Free Routing Graph

Ang Li and David Wentzlaff

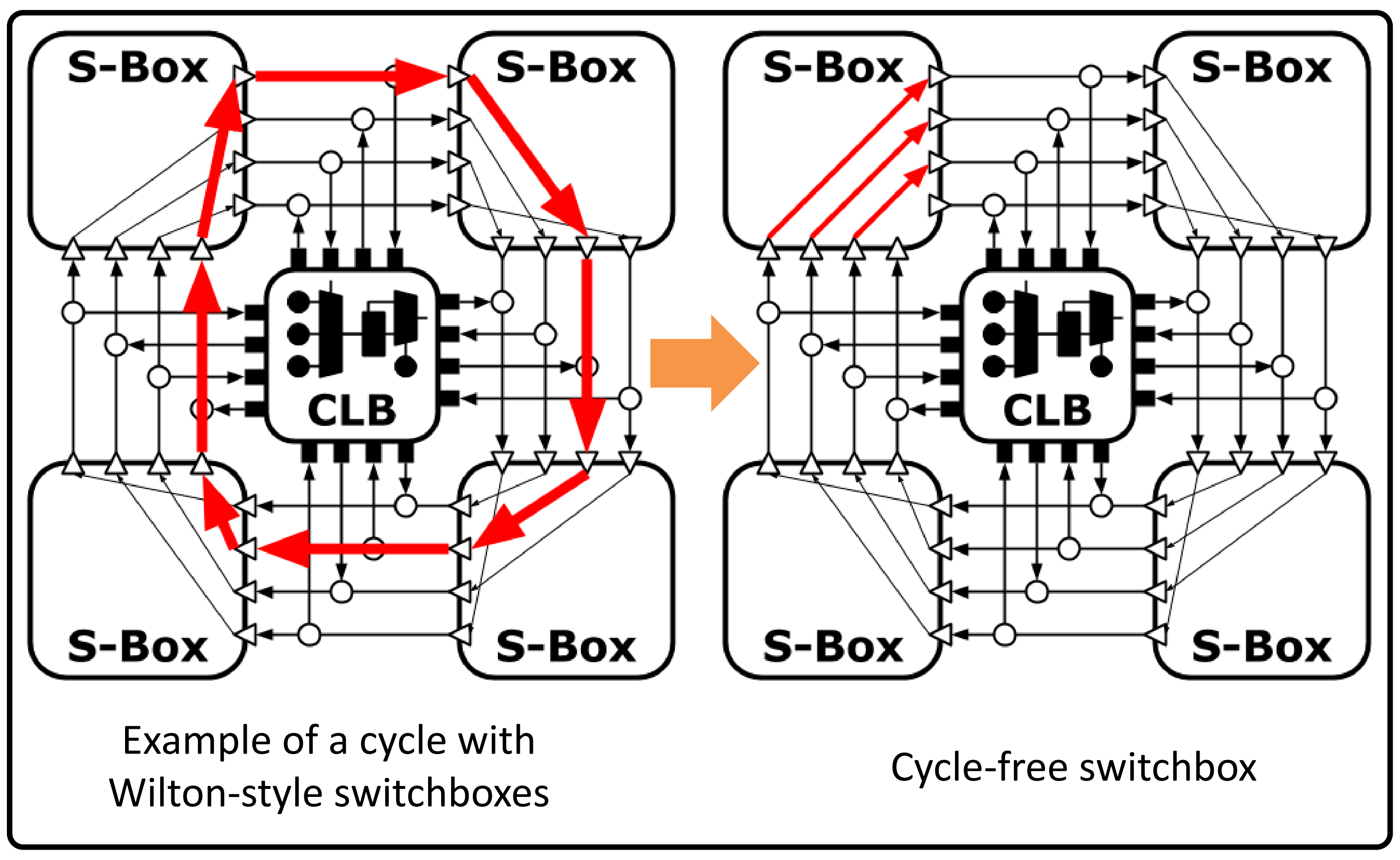
PRINCETON UNIVERSITY

Motivation and Introduction

Goal: Build and characterize an FPGA with **fully-automated** ASIC flow
- This is **NOT** about mapping applications onto an FPGA
- No manual layout
- Accurate timing analysis of wires/switches

Challenge: FPGA routing graph is full of cycles (combinational loops)
- Unable to **constrain** or **measure** wire-to-wire delay

Solution: Cycle-free routing graph
- Applicable to analyzing existing FPGAs
  - Accurate per-wire/switch timing analysis
- Applicable to designing new FPGA architectures
  - Cycle-free architectures enabling **fully-automated** ASIC flow
  - Minimal impact to routability

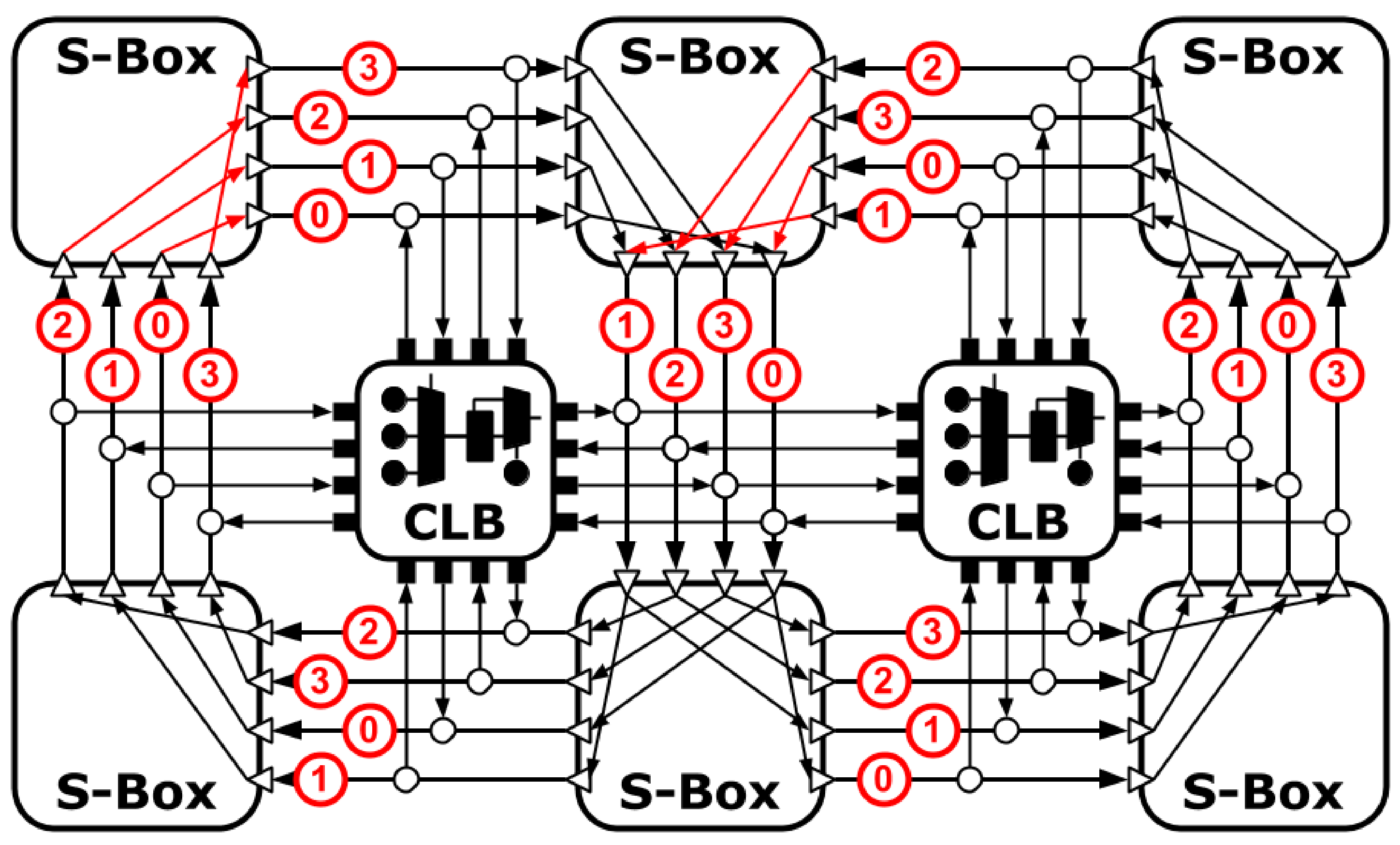


Example of a cycle with Wilton-style switchboxes

Cycle-free switchbox

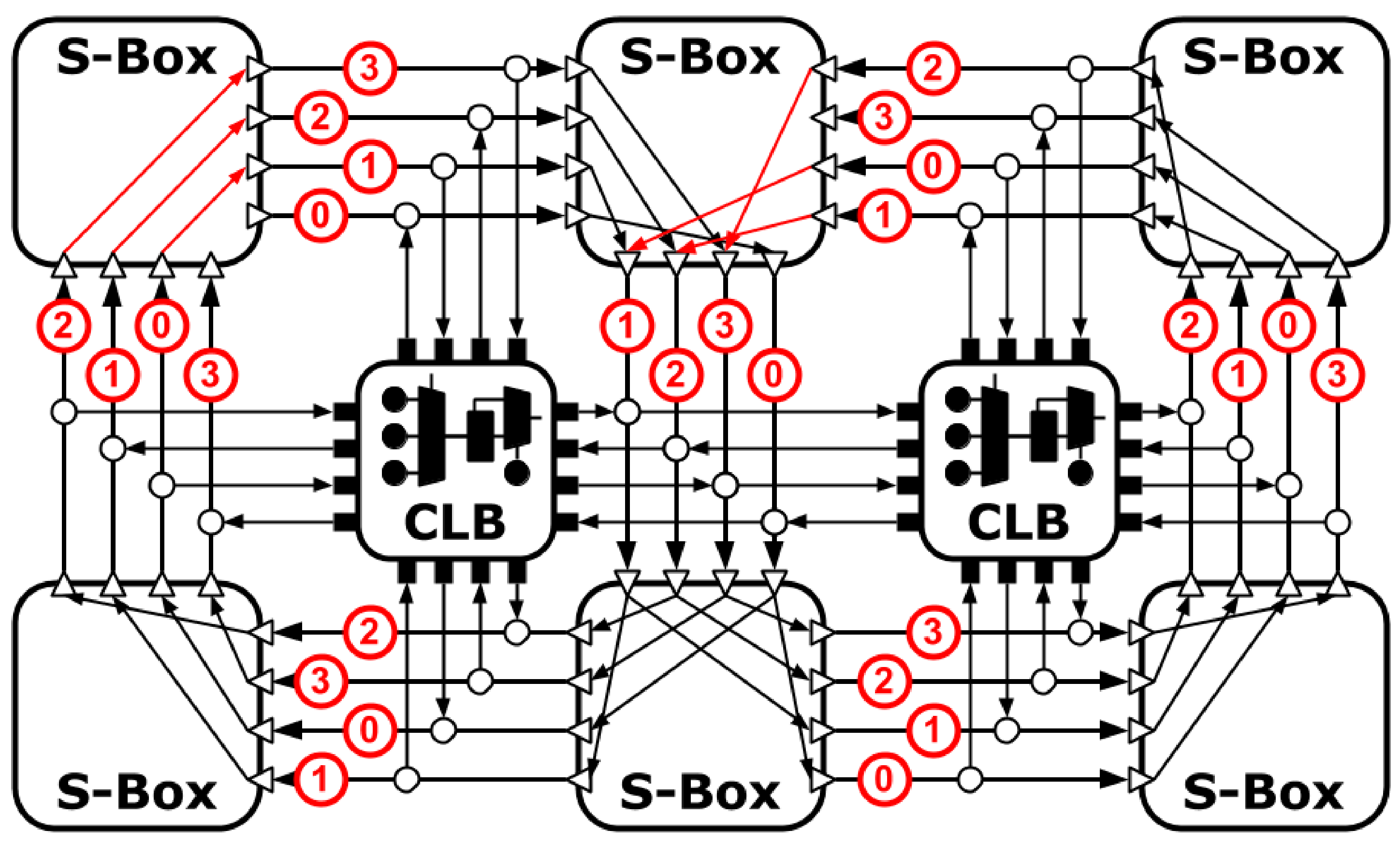
Cycle-Free Routing Graph

Find the Minimally-Sufficient Set of Cycle-Breaking Edges:



- Assign **Logical Class** (LC) to wire segments
- **No going-back**: Remove edges from higher LC to lower LC
- **Break same-LC cycle:** Remove north-east and west-south‡ same-LC turns

Design a Cycle-Free Switchbox



- Use Wilton-style switchbox except for north-east and west-south‡ turns (same-LC connections)
- Connect LC to LC + 1 for north-east and west-south‡ turns

‡ Or another pair of opposite turns, e.g. west-north and south-east

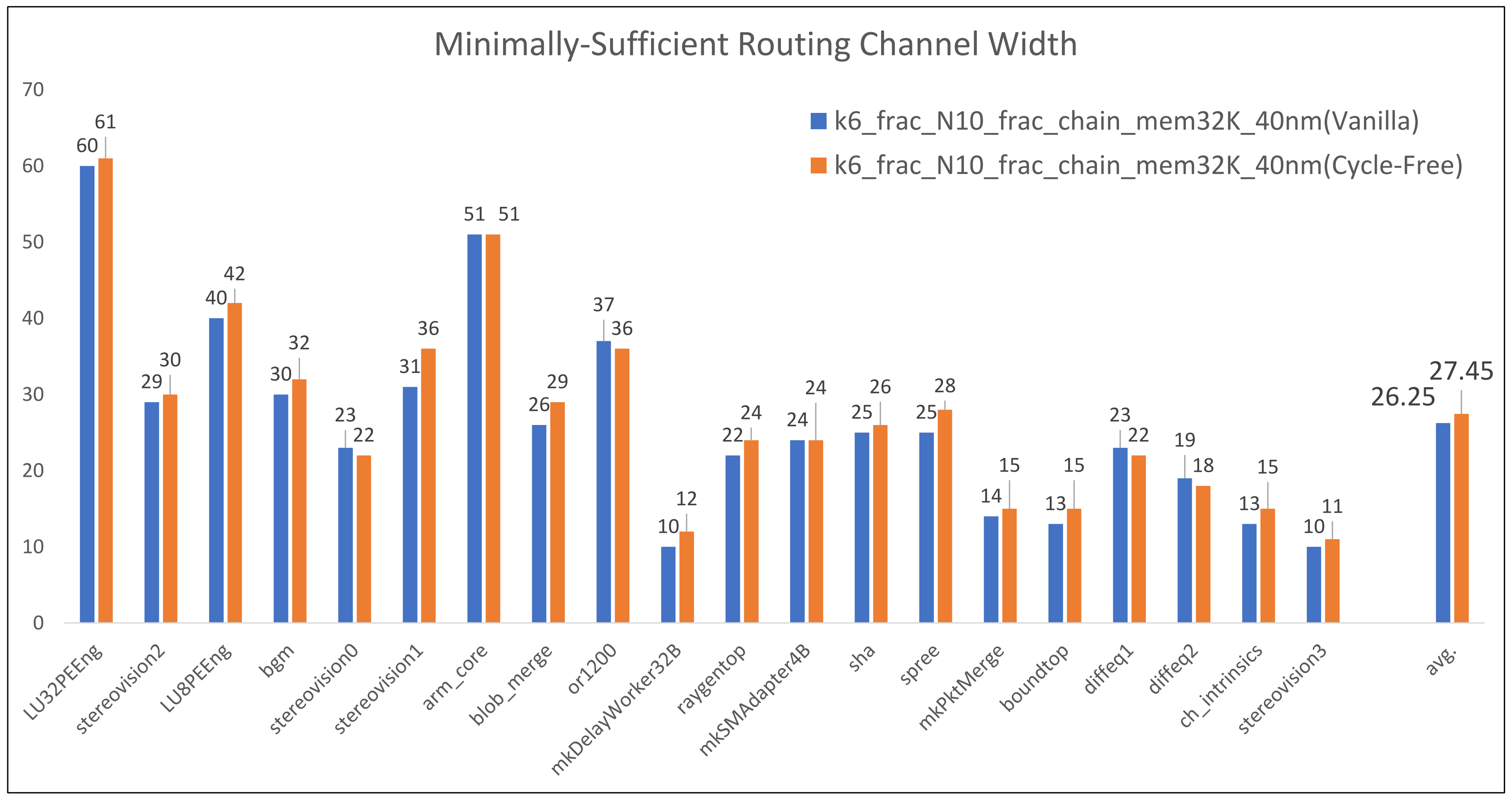
Application

Measure Wire-to-Wire Delay of Existing FPGAs
1. Disable all combinational timing arcs through CLBs
2. Disable all combinational timing arcs of north-east and west-south turns in all switchboxes
3. Run STA and measure wire-to-wire delays
4. Choose another pair of opposite turns and repeat step 2-3

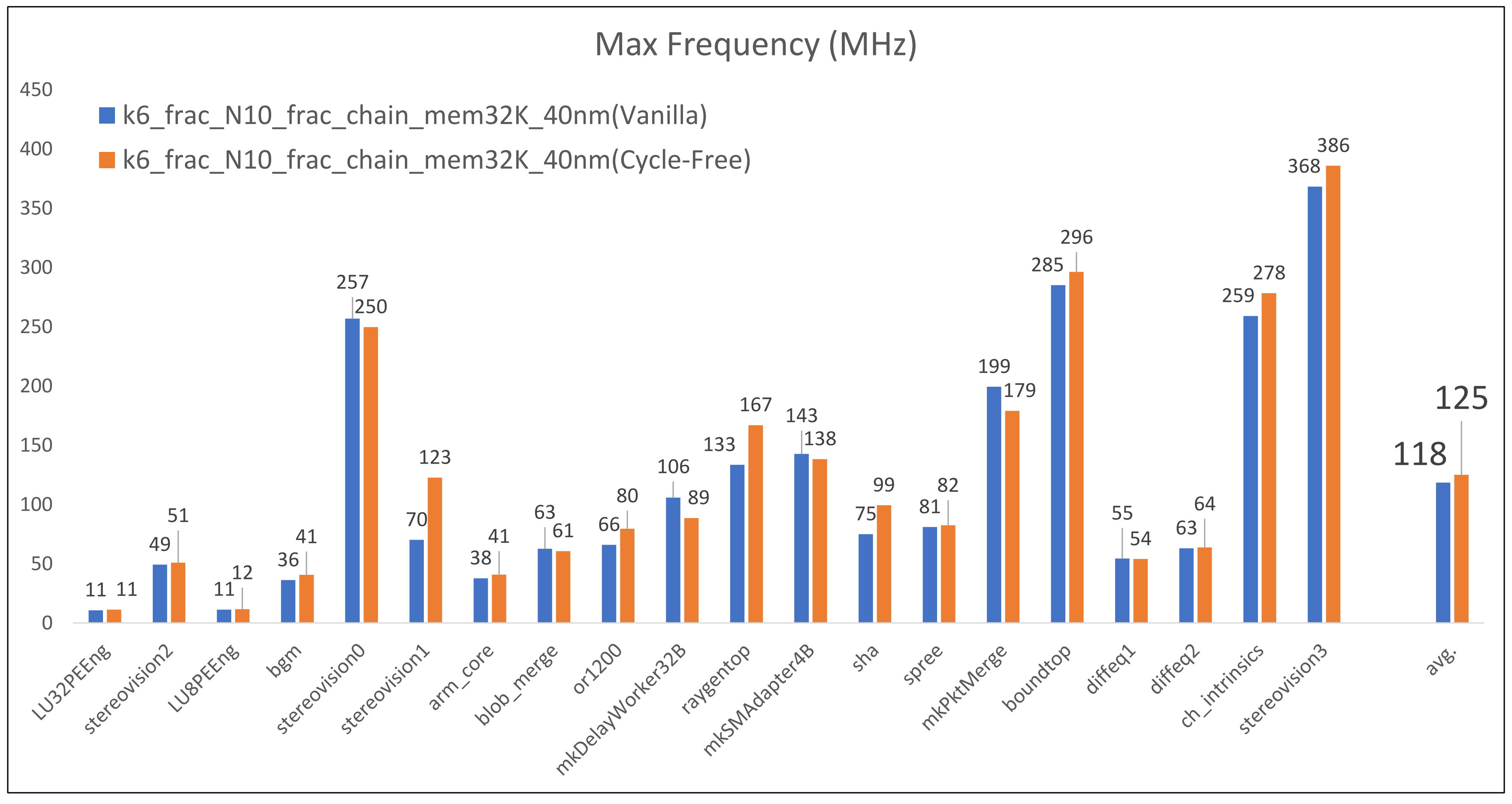
Build a Cycle-Free FPGA with Fully-Automated ASIC Flow
1. Constrain and layout CLBs as hard macros
2. Constrain and layout top-level FPGA design

Advantages of Fully-Automated ASIC flow
- Portable across technology nodes
- Full access to automated EDA optimizations
- Accurate per-wire timing characterization

Evaluation and Conclusion



Minimal Impact on Routability: avg. 4.57% more wires needed



Better Performance: avg. 5.93% higher frequency

PRGA: Open-Source FPGA Workflow

https://prga.readthedocs.io

https://github.com/PrincetonUniversity/prga

Acknowledgements

This material is based on research sponsored by the NSF under Grant No. CCF-1453112, Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement No. FA8650-18-2-7852. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA), the NSF, or the U.S. Government.