IDEA: LSOracle
A learning-based Oracle for Automatic Logic Optimization

Prof. Pierre-Emmanuel Gaillardon, Dr. Xifan Tang
Scott Temple, Walter Lau, Max Austin
University of Utah

IDEA and POSH Phase I Integration Session and Demos
Detroit, MI

15 July 2019 – 19 July 2019

LSOracle: OPDB Chip Bridge Demo

- LSOracle is available on GitHub: https://github.com/LNIS-Projects/LSOracle

Getting Started with LSOracle

build passing docs passing

Introduction

The Logic Synthesis oracle is a framework developed on the top of
manipulation by using different logic optimizers.To do so, the flow
different optimizers for different logics attributes. Currently, it supp
possible the use of sequential MIGs, sequential support has been a

How to compile:

To compile, it is needed support to CMake 3.9 and C++ 17. For the
current version has been tested on Red Hat 7.5 and MacOS Mojave

```
git clone https://github.com/LNIS-Projects/LSOracle.git
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=RELEASE
make
```

MIG Sequential Support Example:

Welcome to LSOracle's documentation!

Motivation

- Motivation

Tool Guide

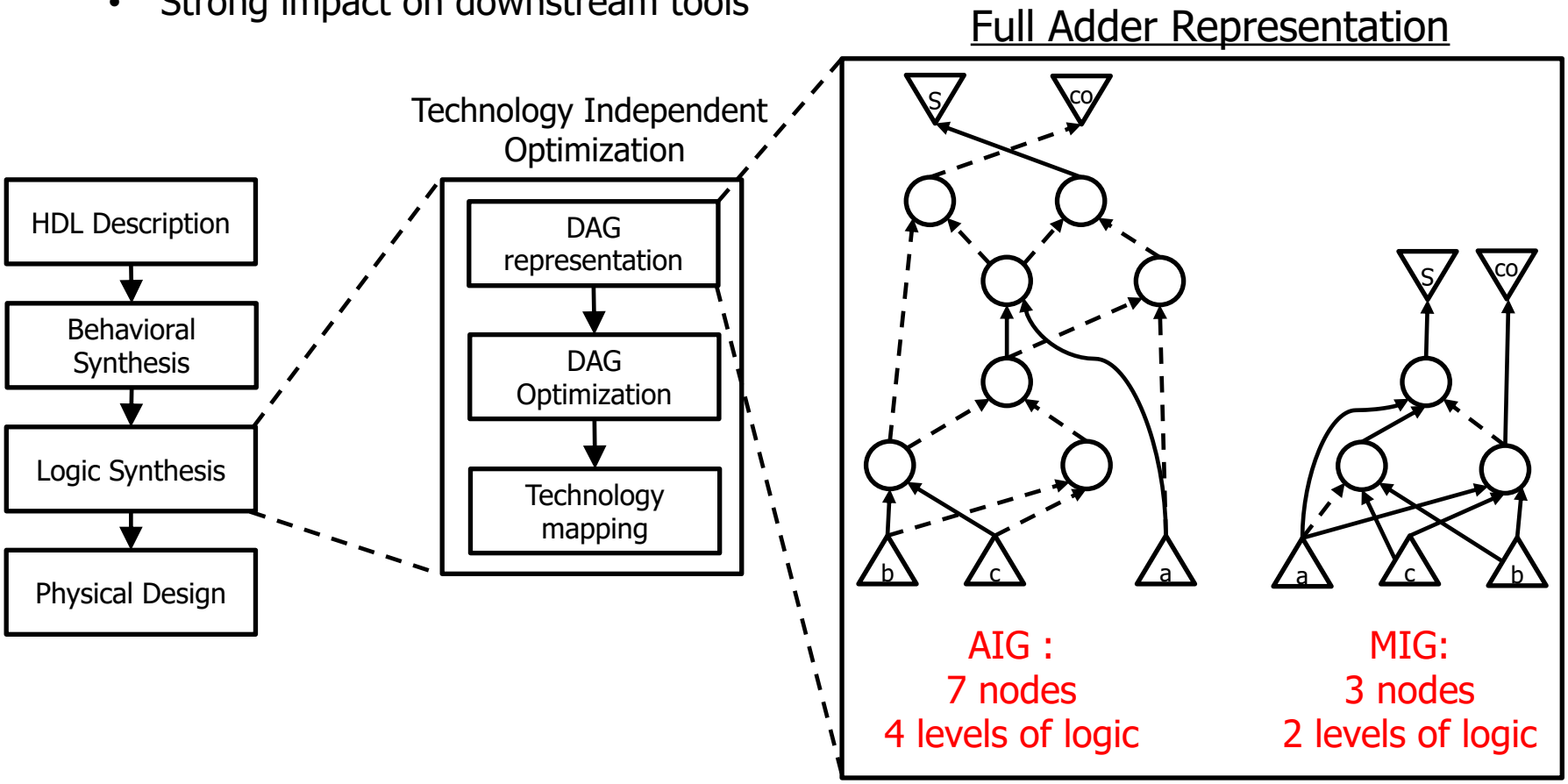
- EPFL Logic Synthesis Libraries
  - Alice
  - Kitty
  - Lorina
  - Mockturtle
- Logic Synthesis Oracle
  - Convulational Neural Network Classifier
  - Partition Manager
  - Partition View
  - Optimization
    - Optimization with Classification
      - Partitions with only AIG Optimization
      - Partitions with only MIG Optimization
      - Brute Force Classification
    - Two-Step Mixed Synthesis
- Sequential MIG Support

Apendix

- Contact

Electronic Design Automation (EDA) Workflow

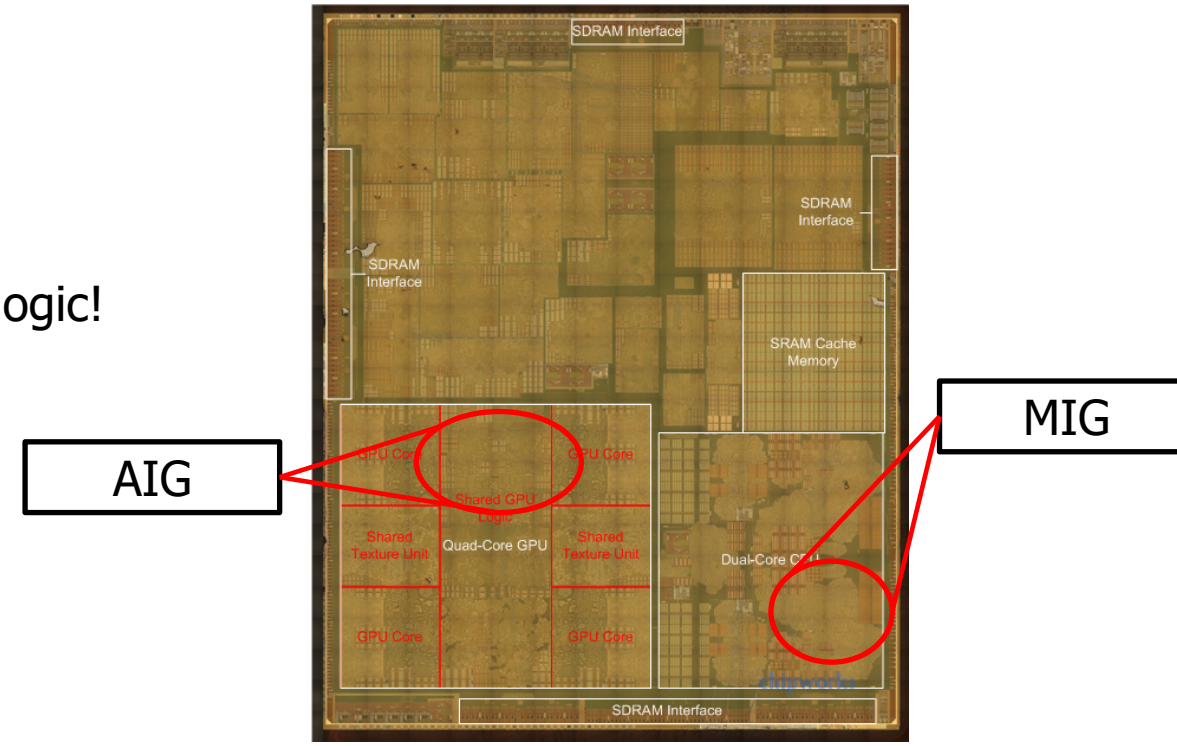
- A typical EDA flow is comprised of a complex chain of tools
  - Three main steps: behavioral synthesis, logic synthesis and physical synthesis
- Synthesis is at the forefront of EDA:
  - Strong impact on downstream tools



Technology Independent Logic Optimization

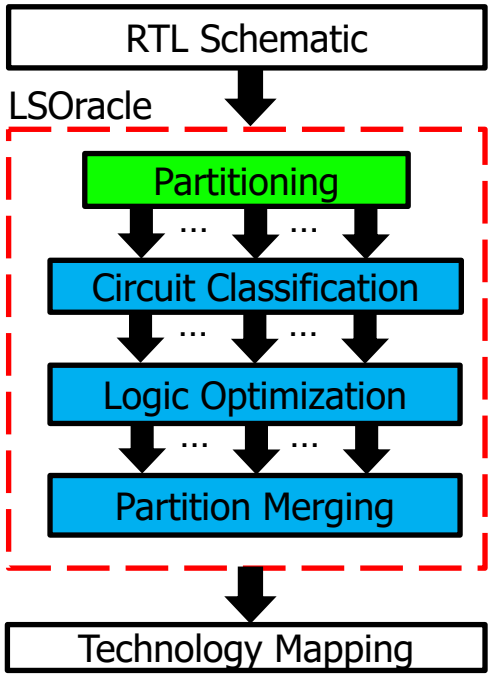
- For the full adder, MIG is more compact, but what about other circuits?
- Considering a dataset of 8,327 combinational circuits
  - **35.7%** perform better with **MIG** : usually arithmetic logic
  - **64.3%** perform better with **AIG** : control/random logic

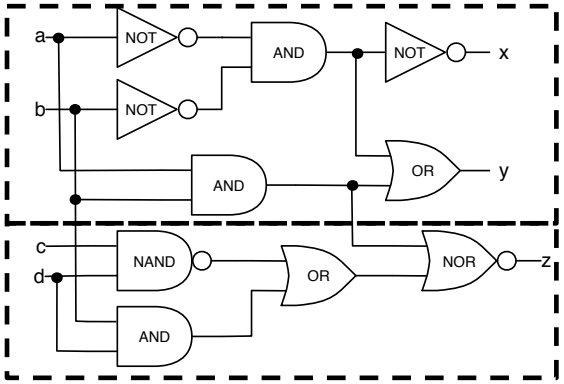
Complex designs have mixed logic!



Can we automatically select the best DAG representation and the best optimization approach for different logic blocks on the circuit?

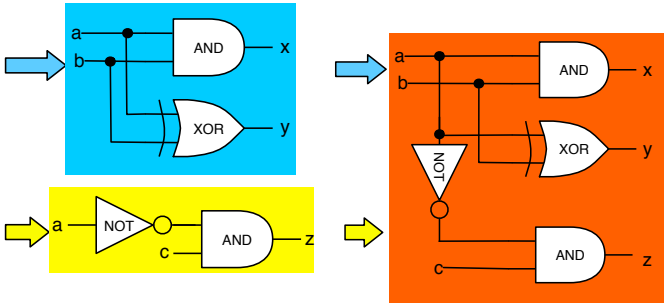
Automatic learning-based logic synthesis framework





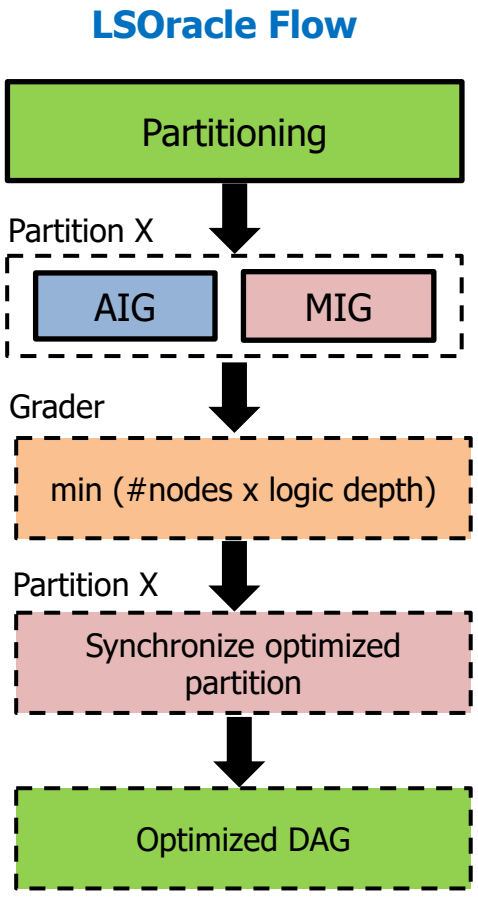
Open-source library KaHyPar

Open-source library
Frugallydeep



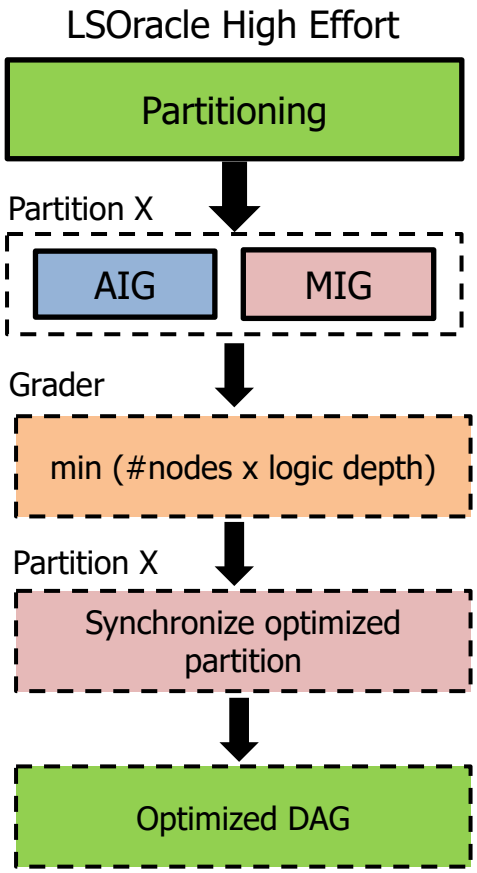
EPFL Logic
Synthesis Libraries

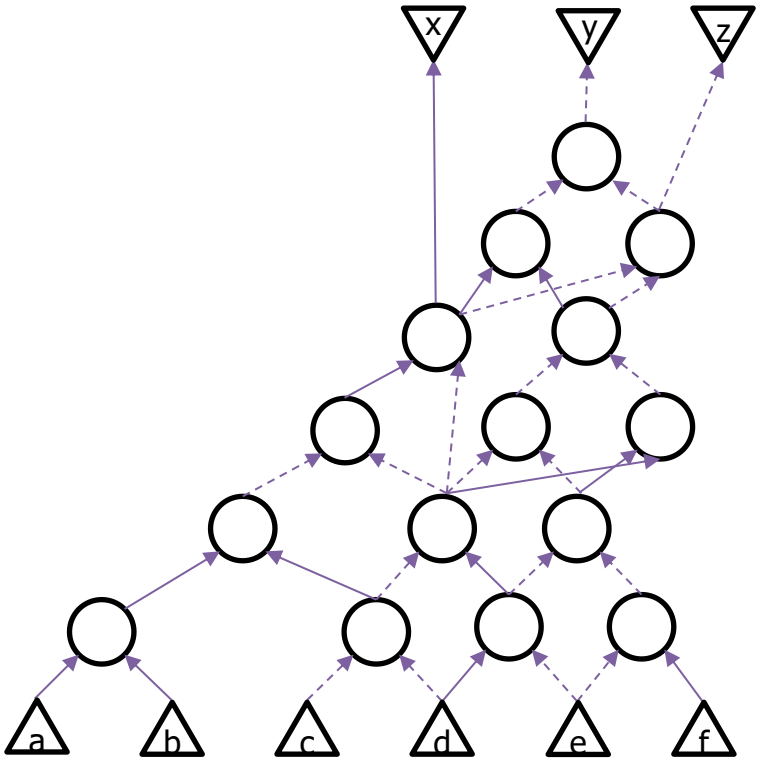
LSOracle Framework: Current Status



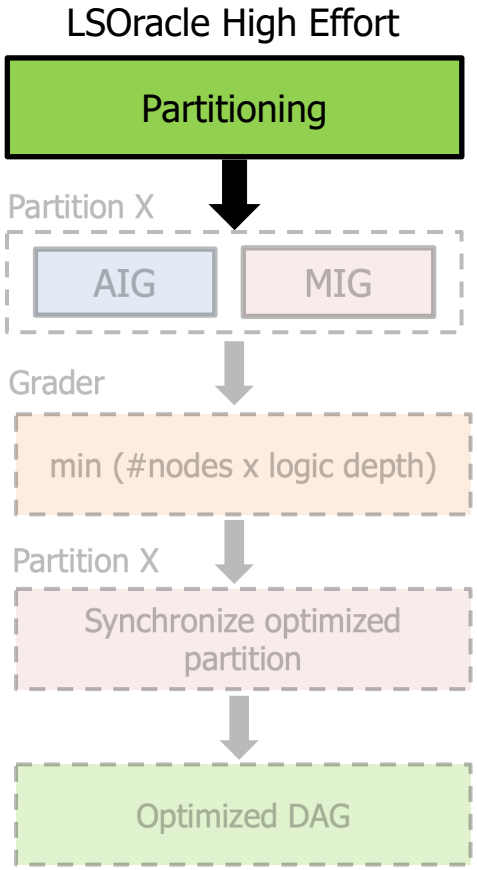
All partitions are treated and optimized by different methods, taking advantage of different optimizers for different structures

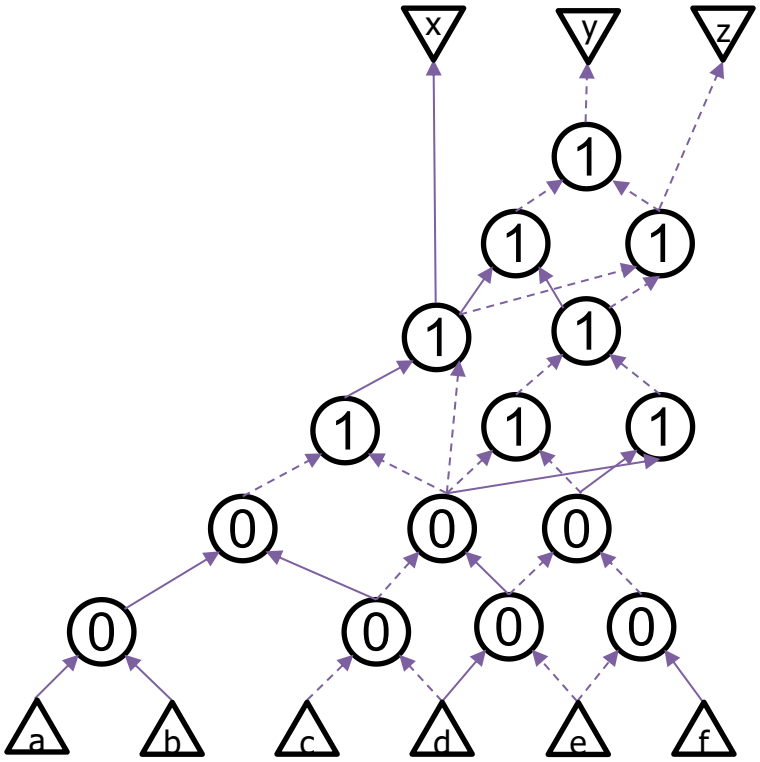
LSOracle Flow Example



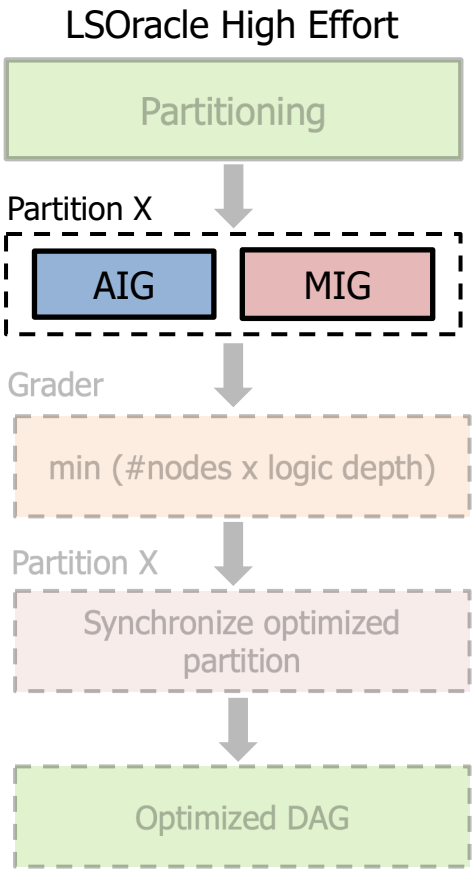


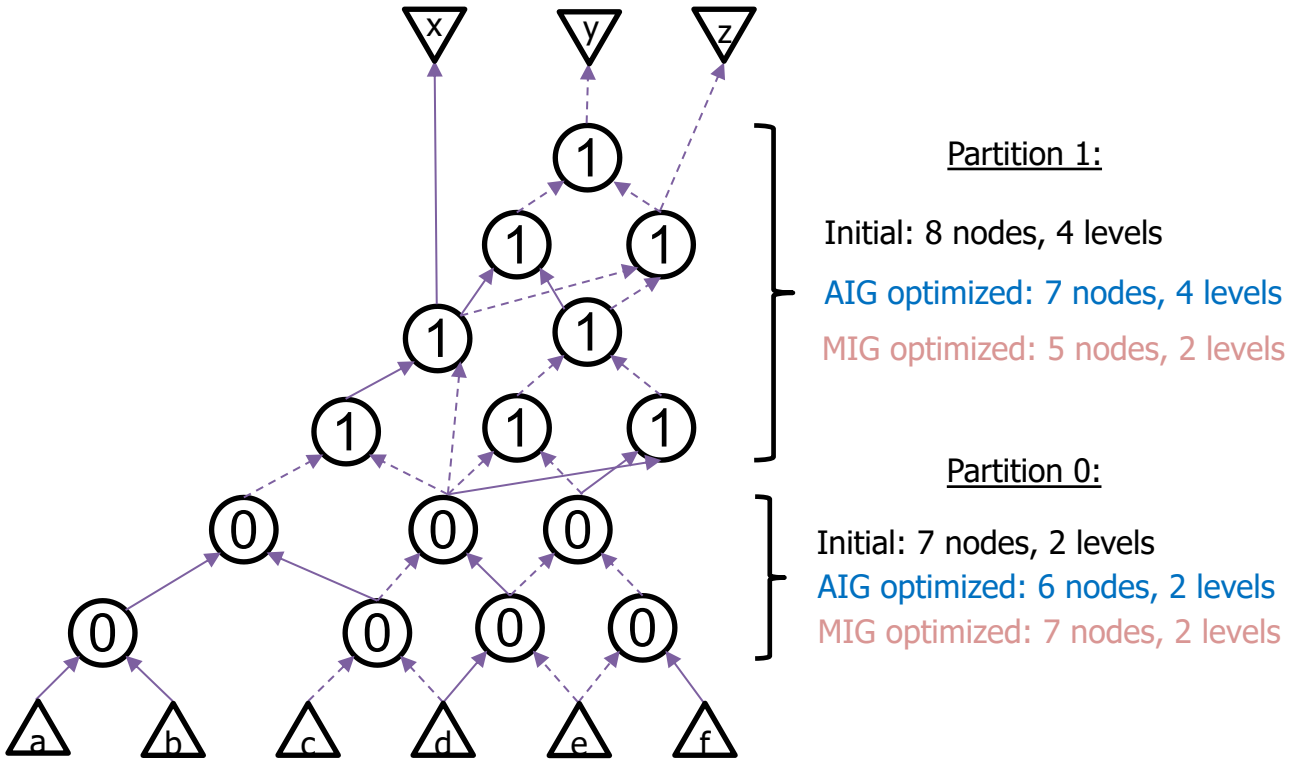
LSOracle Flow Example – cont'd



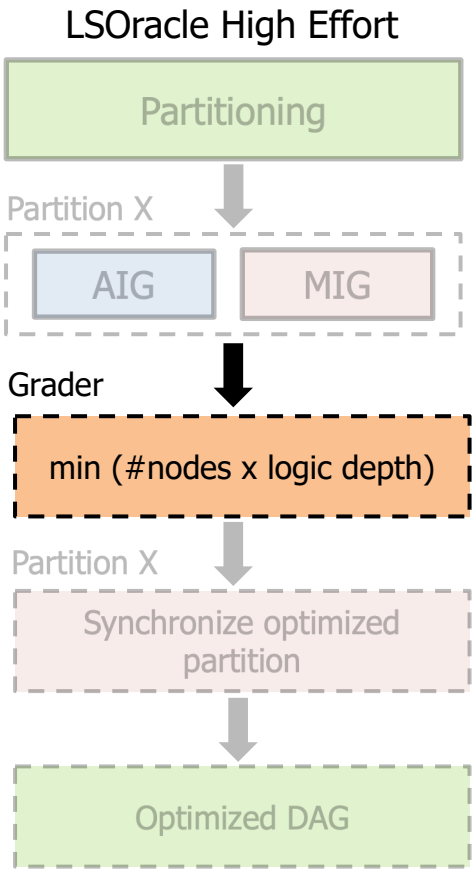


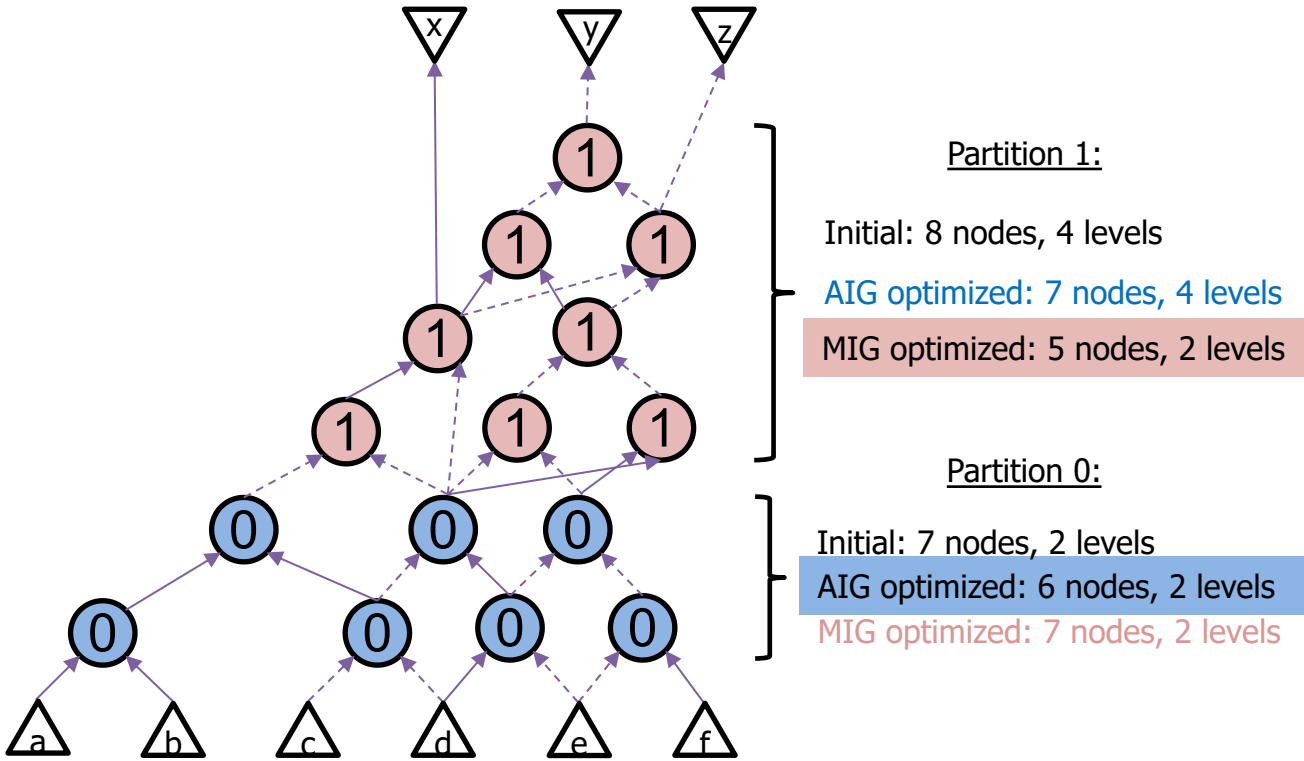
LSOracle Flow Example – cont'd



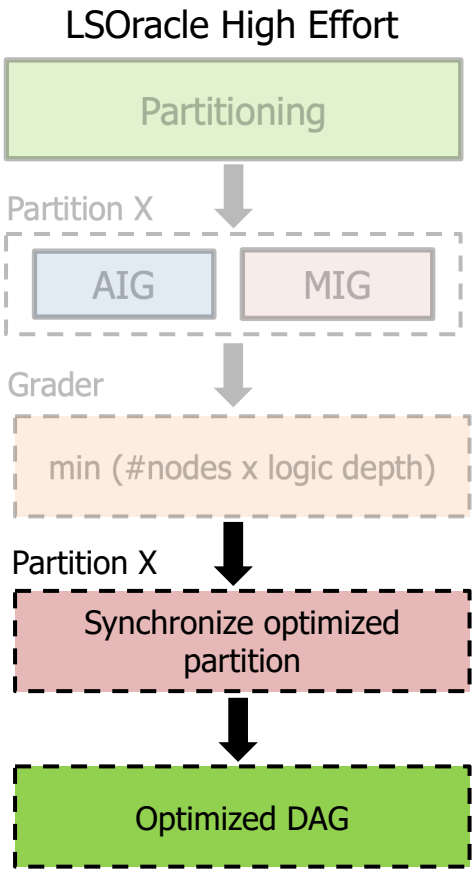


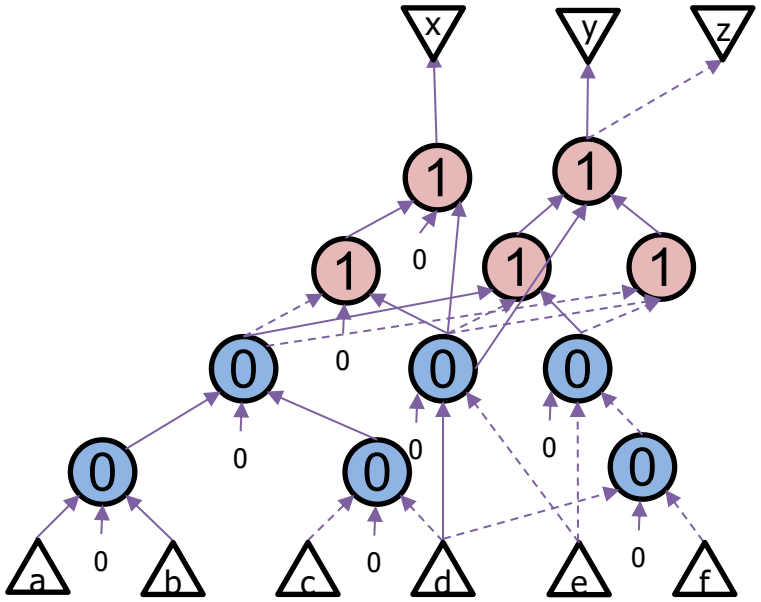
LSOracle Flow Example – cont'd





LSOracle Flow Example – cont'd





From 15 nodes and 6 levels of logic to 11 nodes and 4 levels

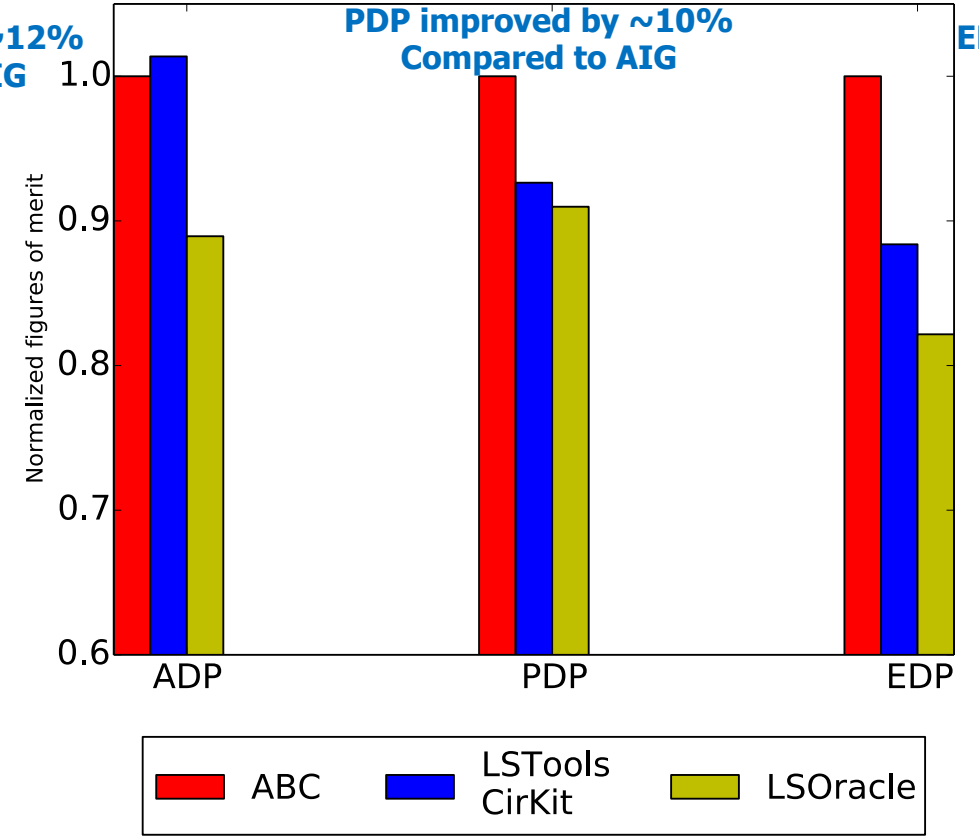
Integrated Flow Demo

- Benchmark: Chip Bridge from OPDB mapped using ASAP 7nm
  - 124k nodes; 7× > than PicoRV; 1000× > than the example presented in January
  - Runtime ~5 minutes on mobile core i5
- All optimization flows based on ABC's *resyn2* script

ADP improved by ~12% Compared to AIG

PDP improved by ~10% Compared to AIG

EDP improved by ~18% Compared to AIG



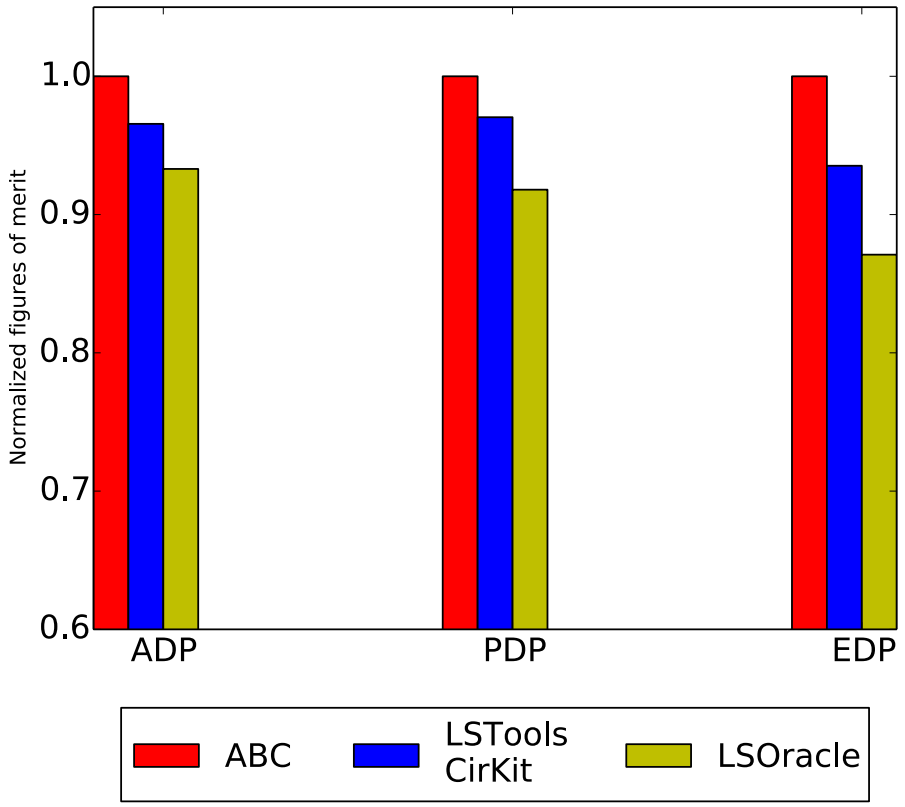
Post-tech Mapping ASIC Results

- 10 circuits from OPDB, OpenCores and EPFL
- Circuits mapped using ASAP 7nm
- All optimization flows based on ABC's *resyn2* script

ADP improved by ~7.3% Compared to AIG

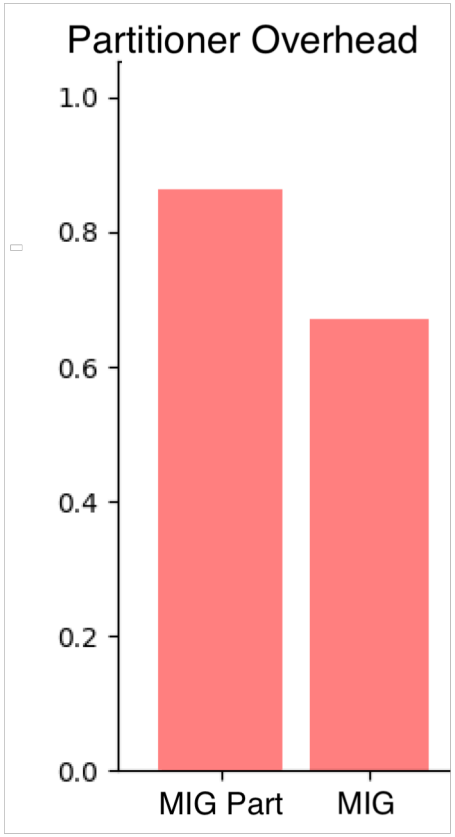
PDP improved by ~8.17% Compared to AIG

EDP improved by ~12.9% Compared to AIG



Improving Performance Through Partition Merging

- In Mixed Synthesis mode, partitions may cause a loss in QoR
- Optimizing the largest regions possible is desirable
- We have implemented a feature that merges adjacent partitions which have been classified as the same type
- This gives larger partitions which are less likely to break up important functionality

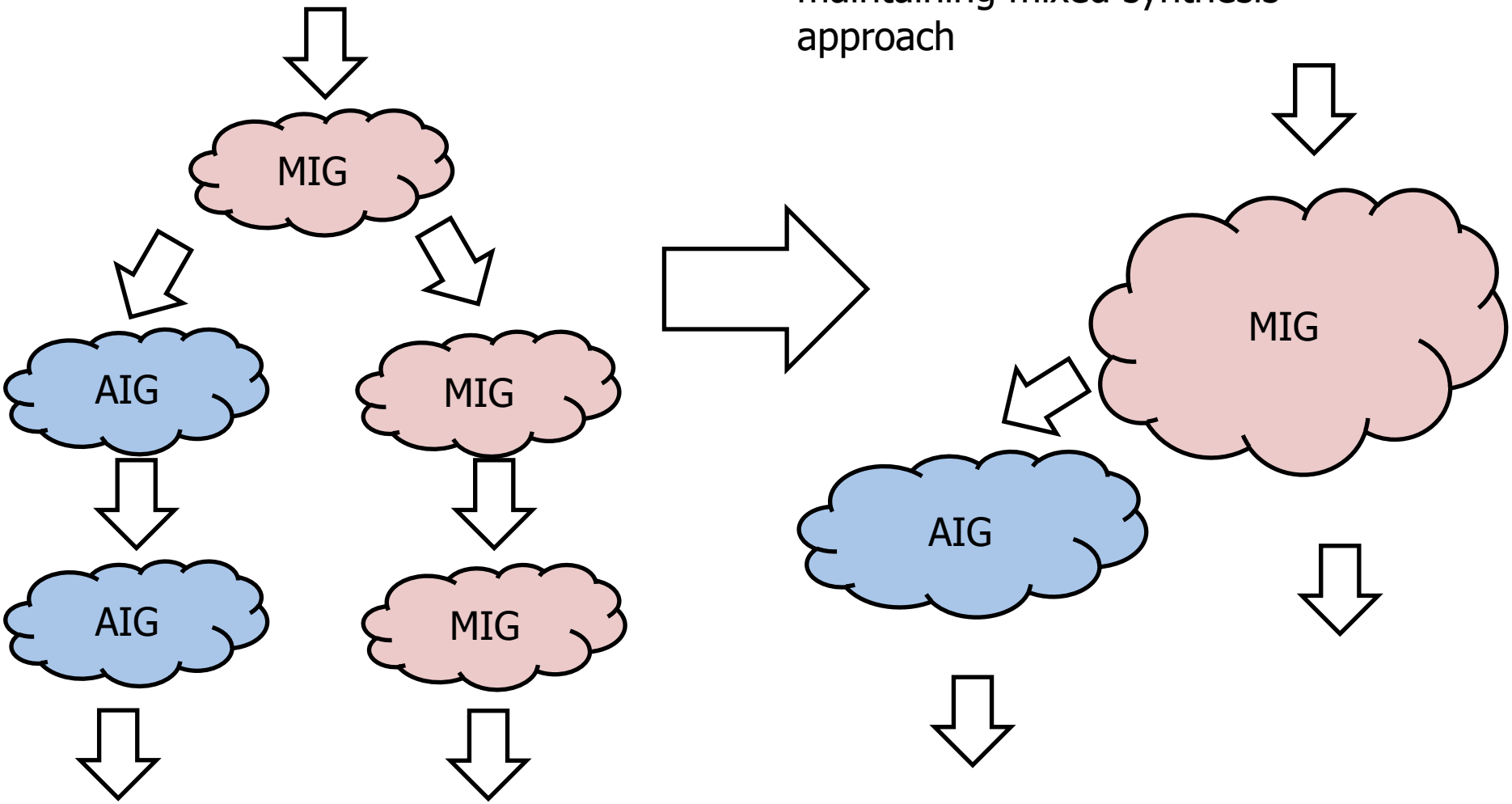


MIG performance on partitioned vs original network, showing loss of global optimization. (AIG normalized to 1)

Partition Merging

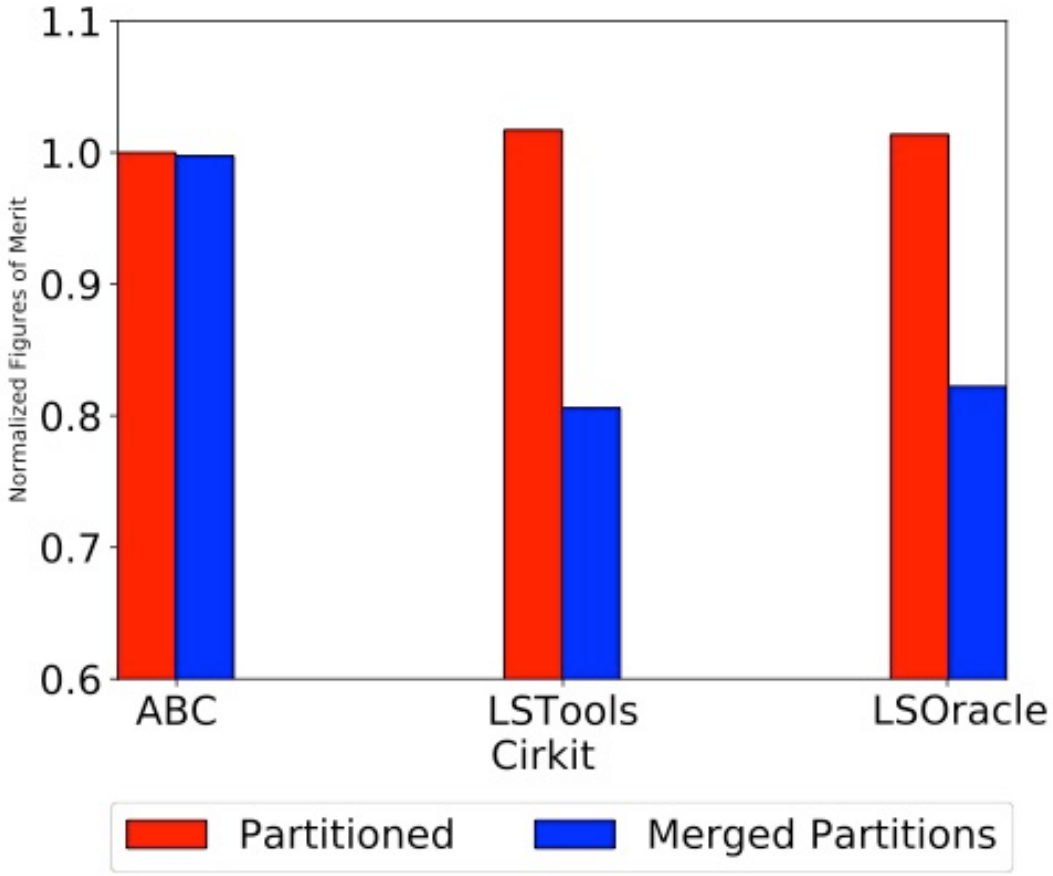
Before merging: Many partitions
Of similar size

After merging: Minimal number of large partitions, while maintaining mixed synthesis approach



Partition Merging: LUT Tech Mapping

Fewer, larger partitions to reduce loss of global optimization



Summary

- LSOracle alpha version released
  - Fully integrated flow
  - Evaluated in circuits with more than 100,000 nodes
  - Handles thousands of partitions
- Two main operation modes
  - LSOracle alpha
  - LSOracle developmental deep learning classifier
- Post-tech mapping results presented over 10 new circuits
  - ADP, PDP and EDP improved when compared to only AIG or MIG optimization
- Fork us: https://github.com/LNIS-Projects/LSOracle

Thank you!
Q & A