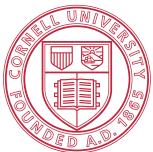


PIMap

A Parallelized Iterative Improvement Approach to Area Optimization for LUT-Based Technology Mapping

Gai Liu and Zhiru Zhang

Computer Systems Lab
Electrical and Computer Engineering
Cornell University



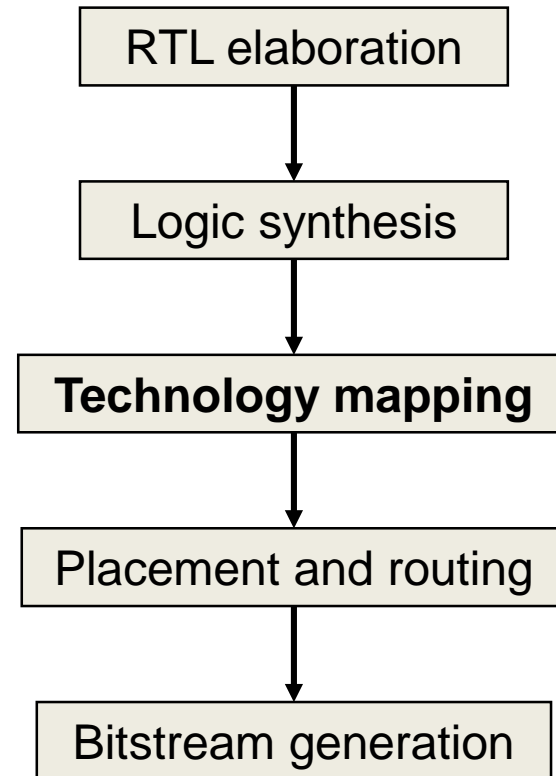
Cornell University



Technology Mapping for FPGAs

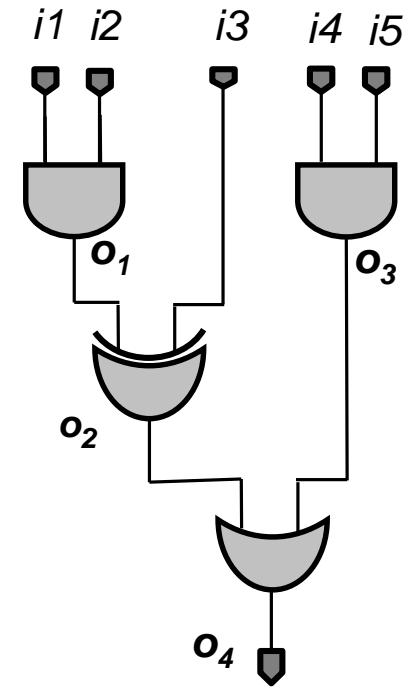
- ▶ Technology mapping is an essential step in FPGA CAD flow
 - Dictates the design area (i.e., number of LUTs)
 - Large impact on timing of the final design

A typical FPGA CAD flow



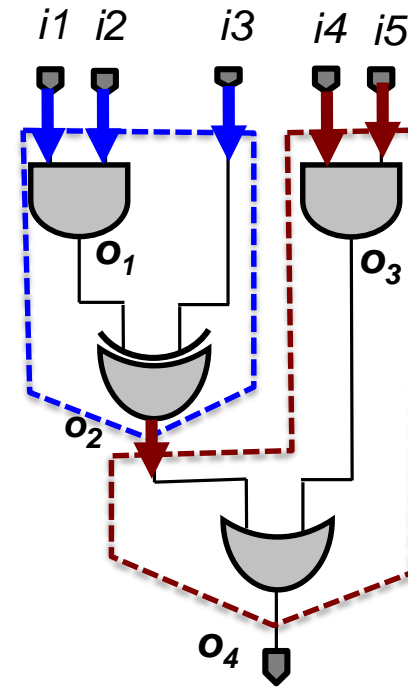
Technology Mapping for FPGAs

- ▶ Cover a gate-level logic network using LUTs
 - K-input LUT (k-LUT) can implement any k-input 1-output combinational logic network



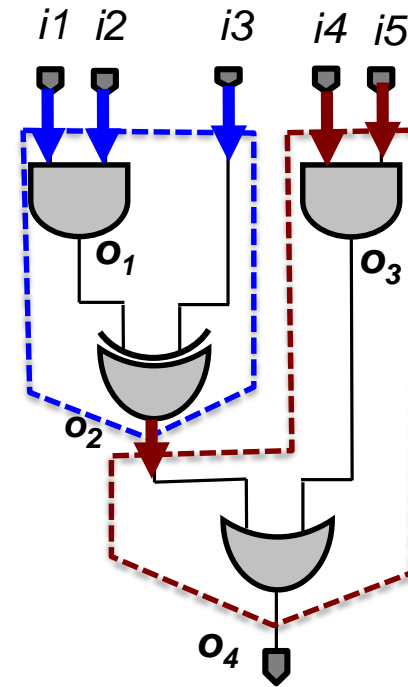
Technology Mapping for FPGAs

- ▶ Cover a gate-level logic network using LUTs
 - K-input LUT (k-LUT) can implement any k-input 1-output combinational logic network



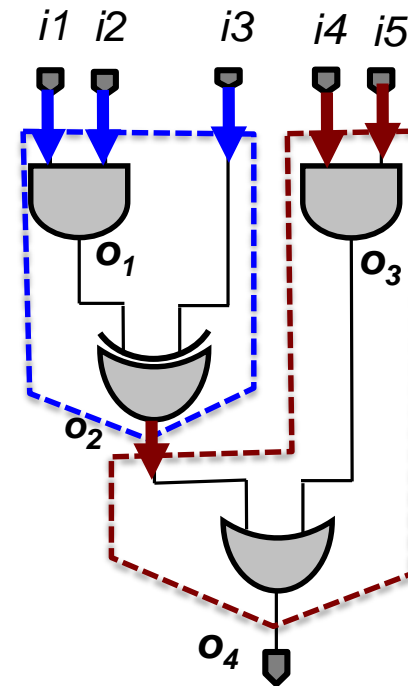
Technology Mapping for FPGAs

- ▶ Cover a gate-level logic network using LUTs
 - K-input LUT (k-LUT) can implement any k-input 1-output combinational logic network
 - This work focuses on combinational circuit



Technology Mapping for FPGAs

- ▶ Cover a gate-level logic network using LUTs
 - K-input LUT (k-LUT) can implement any k-input 1-output combinational logic network
 - This work focuses on combinational circuit
- ▶ Quality metrics for technology mapping
 - **Area:** number of LUTs needed
 - **Depth:** longest path from PI to PO in # of LUTs



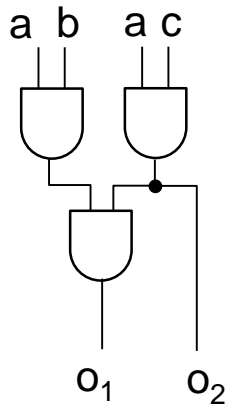
Tech Mapping for Area is a Hard Problem

- ▶ Case 1: no logic restructuring

Tech Mapping for Area is a Hard Problem

- ▶ Case 1: no logic restructuring

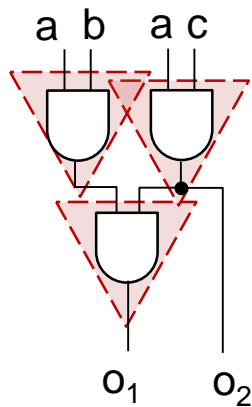
Goal: map to 3-input LUTs



Tech Mapping for Area is a Hard Problem

- ▶ Case 1: no logic restructuring

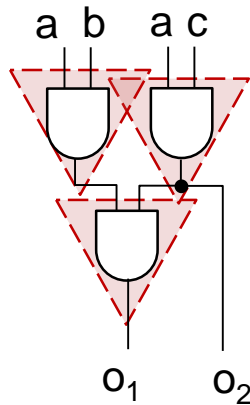
Goal: map to 3-input LUTs



Tech Mapping for Area is a Hard Problem

- ▶ Case 1: no logic restructuring
 - Already NP-hard [1]

Goal: map to 3-input LUTs

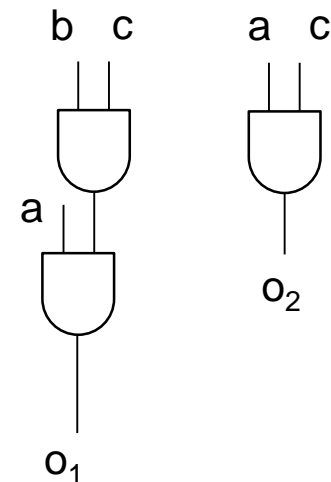
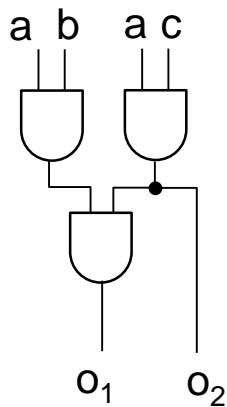


[1] Farrahi and Sarrafzadeh, TCAD'02

Tech Mapping for Area is a Hard Problem

- ▶ Case 1: no logic restructuring
 - Already NP-hard [1]
- ▶ Case 2: with logic restructuring

Goal: map to 3-input LUTs

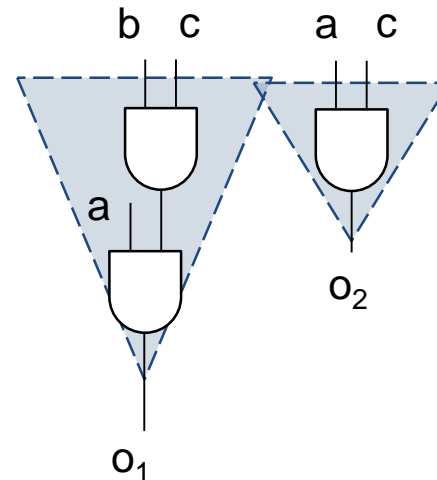
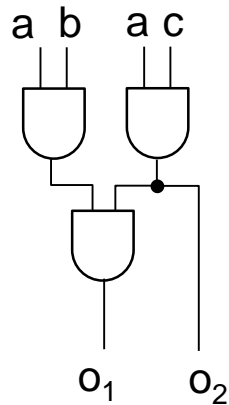


[1] Farrahi and Sarrafzadeh, TCAD'02

Tech Mapping for Area is a Hard Problem

- ▶ Case 1: no logic restructuring
 - Already NP-hard [1]
- ▶ Case 2: with logic restructuring

Goal: map to 3-input LUTs

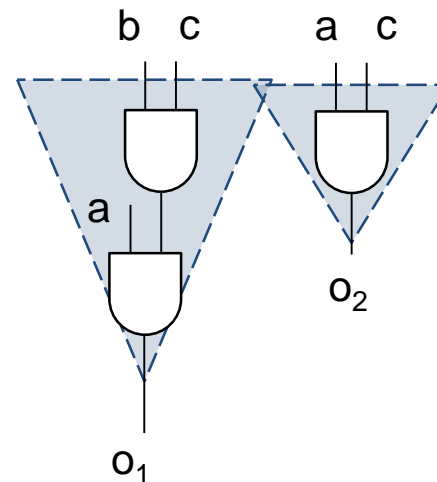
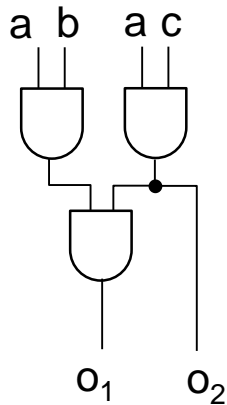


[1] Farrahi and Sarrafzadeh, TCAD'02

Tech Mapping for Area is a Hard Problem

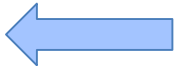
- ▶ Case 1: no logic restructuring
 - Already NP-hard [1]
- ▶ Case 2: with logic restructuring
 - Even harder to find optimal solution
 - Existing approach: heuristically transform logic network for better mapping quality

Goal: map to 3-input LUTs

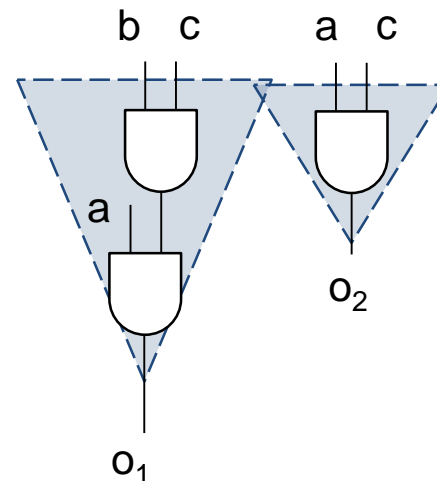
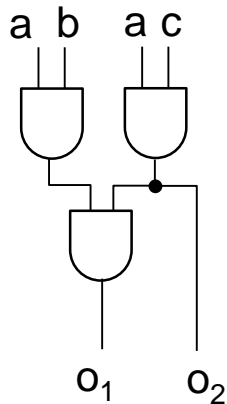


[1] Farrahi and Sarrafzadeh, TCAD'02

Tech Mapping for Area is a Hard Problem

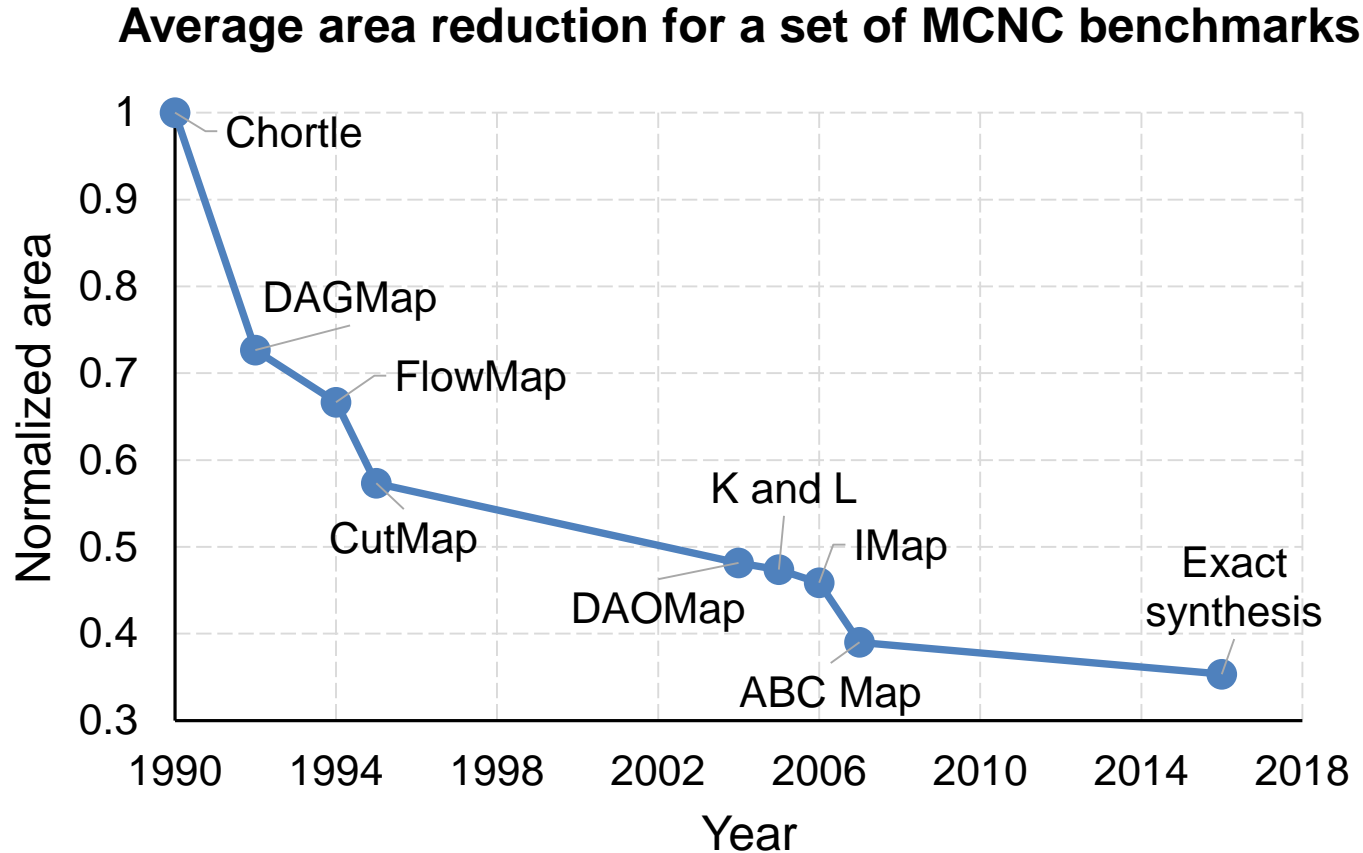
- ▶ Case 1: no logic restructuring
 - Already NP-hard [1]
- ▶ Case 2: with logic restructuring  Focus of this work
 - Even harder to find optimal solution
 - Existing approach: heuristically transform logic network for better mapping quality

Goal: map to 3-input LUTs



[1] Farrahi and Sarrafzadeh, TCAD'02

Representative Academic Mappers



Chortle: Francis, et al., DAC'90

DAGMap: Chen, et al., DT'92

FlowMap: Cong and Ding, TCAD'94

CutMap: Cong and Hwang, FPGA'95

DAOMap: Chen and Cong, ICCAD'04

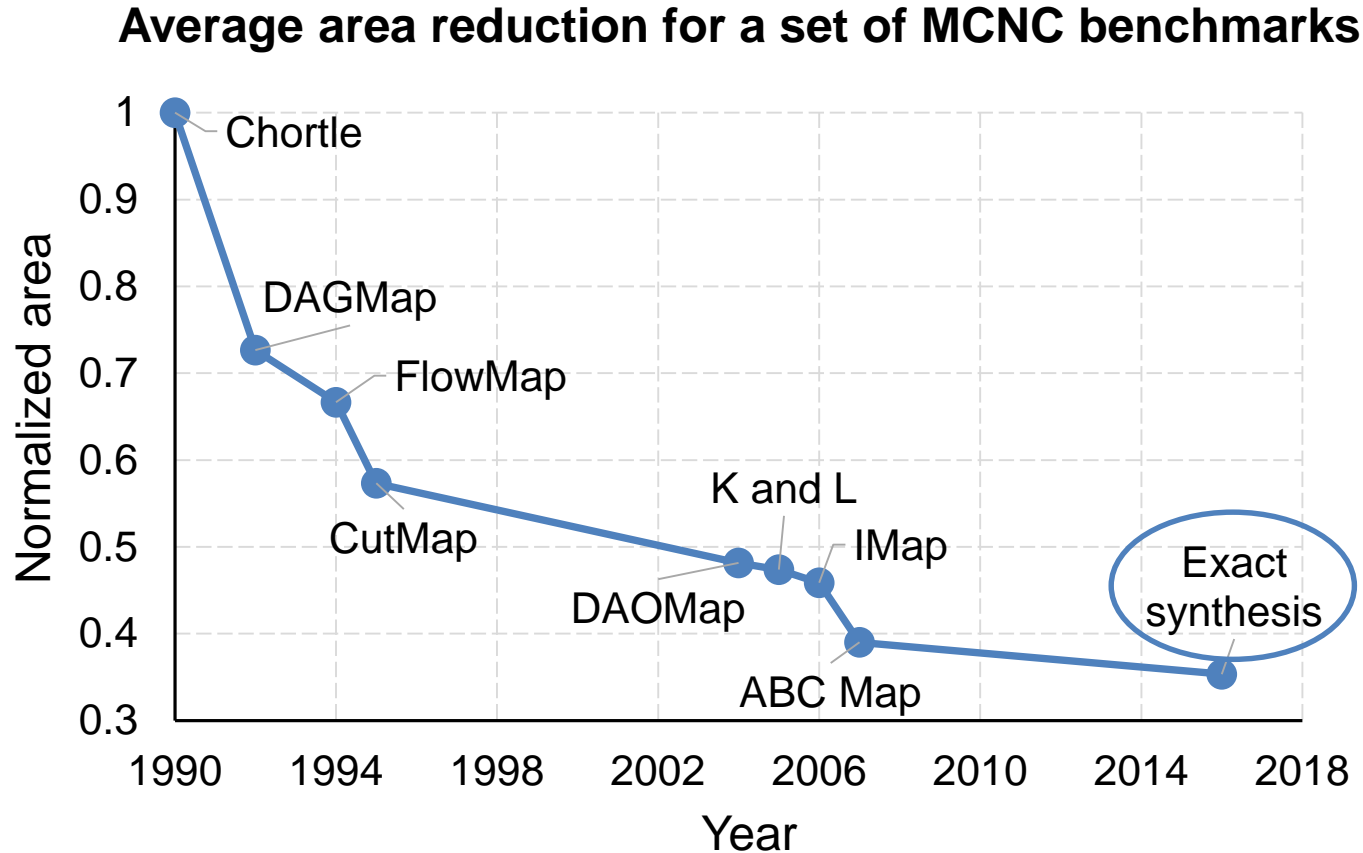
K and L: Kao and Lai, TDAES'05

Imap: Manohararajah, et al., TCAD'06

ABC Map: Mishchenko, et al., TCAD'07

Exact synthesis: Haaswijk, et al., ASPDAC'17

Representative Academic Mappers



Chortle: Francis, et al., DAC'90
DAGMap: Chen, et al., DT'92
FlowMap: Cong and Ding, TCAD'94

CutMap: Cong and Hwang, FPGA'95
DAOMap: Chen and Cong, ICCAD'04
K and L: Kao and Lai, TDAES'05

Imap: Manohararajah, et al., TCAD'06
ABC Map: Mishchenko, et al., TCAD'07
Exact synthesis: Haaswijk, et al., ASPDAC'17

World Record for Area Optimization

Best LUT-6 implementation for EPFL benchmark suite [1]

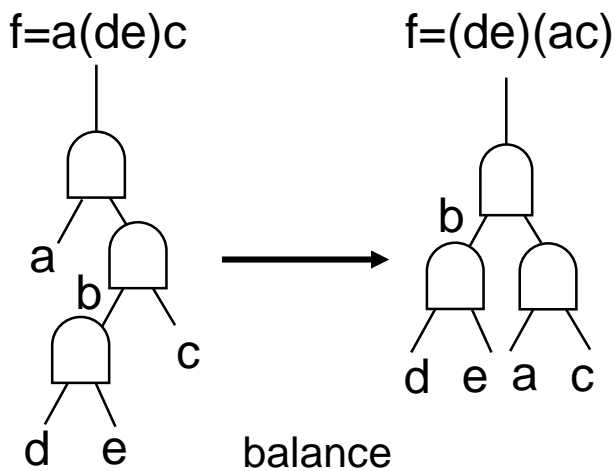
Best results for LUT6 count

Arithmetic					
Benchmark name	Author's name	Author's affiliation	Synthesis Method	Size	Depth
Adder	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	201	73
Barrel Shifter	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	512	4
Divisor	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	3813	1542
Hypotenuse	***	***	***	***	***
Log2	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	7344	142
Max	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	532	192
Multiplier	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	5681	120
Sine	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	1347	62
Square-root	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	3286	1180
Square	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	3800	116
Random-control					
Benchmark name	Author's name	Author's affiliation	Synthesis Method	Size	Depth
Round-robin arbiter	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	429	24
ALU ctrl	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	29	2
Coding-CAVLC	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	107	6
Decoder	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	272	2
i2c controller	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	230	7
Int2float	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	34	4
Mem ctrl	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	2399	23
Priority encoder	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	118	27
Lookahead XY router	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	53	6
Voter	Robert K. Brayton & Alan Mishchenko	UC Berkeley	ABC Extreme Mapper	1521	18

[1] Amarù, et al., <http://lsi.epfl.ch/benchmarks>

Common Restructuring Techniques

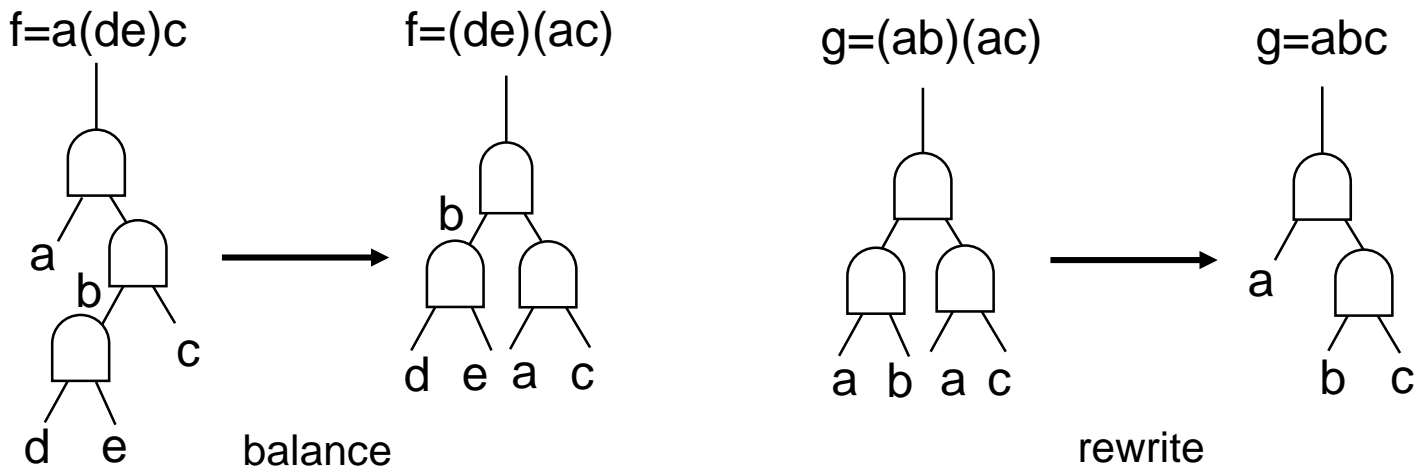
- ▶ Heuristically shrink logic network by logic rewriting (e.g., [1])



[1] Mishchenko, Chatterjee, Brayton, DAC'06

Common Restructuring Techniques

- ▶ Heuristically shrink logic network by logic rewriting (e.g., [1])



[1] Mishchenko, Chatterjee, Brayton, DAC'06

Typical Pre-mapping Transformation Sequence

- ▶ A typical area-minimizing script in ABC^[1]:

[1] Mishchenko, et al., TCAD'07

balance → rewrite → balance → rewrite

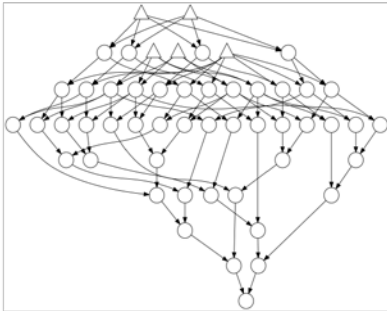
Typical Pre-mapping Transformation Sequence

- ▶ A typical area-minimizing script in ABC^[1]:

[1] Mishchenko, et al., TCAD'07

balance → rewrite → balance → rewrite

Initial and-inverter graph for *xor5*



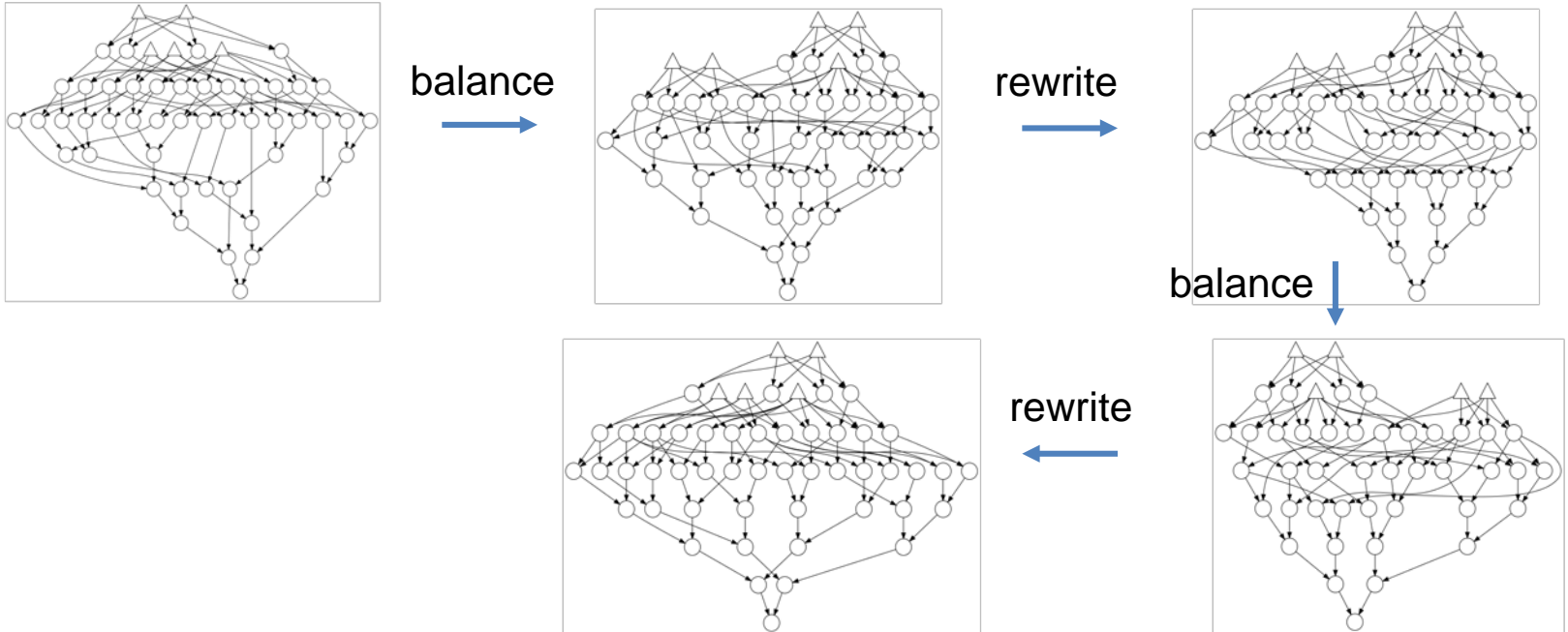
Typical Pre-mapping Transformation Sequence

- ▶ A typical area-minimizing script in ABC^[1]:

[1] Mishchenko, et al., TCAD'07

balance → rewrite → balance → rewrite

Initial and-inverter graph for *xor5*



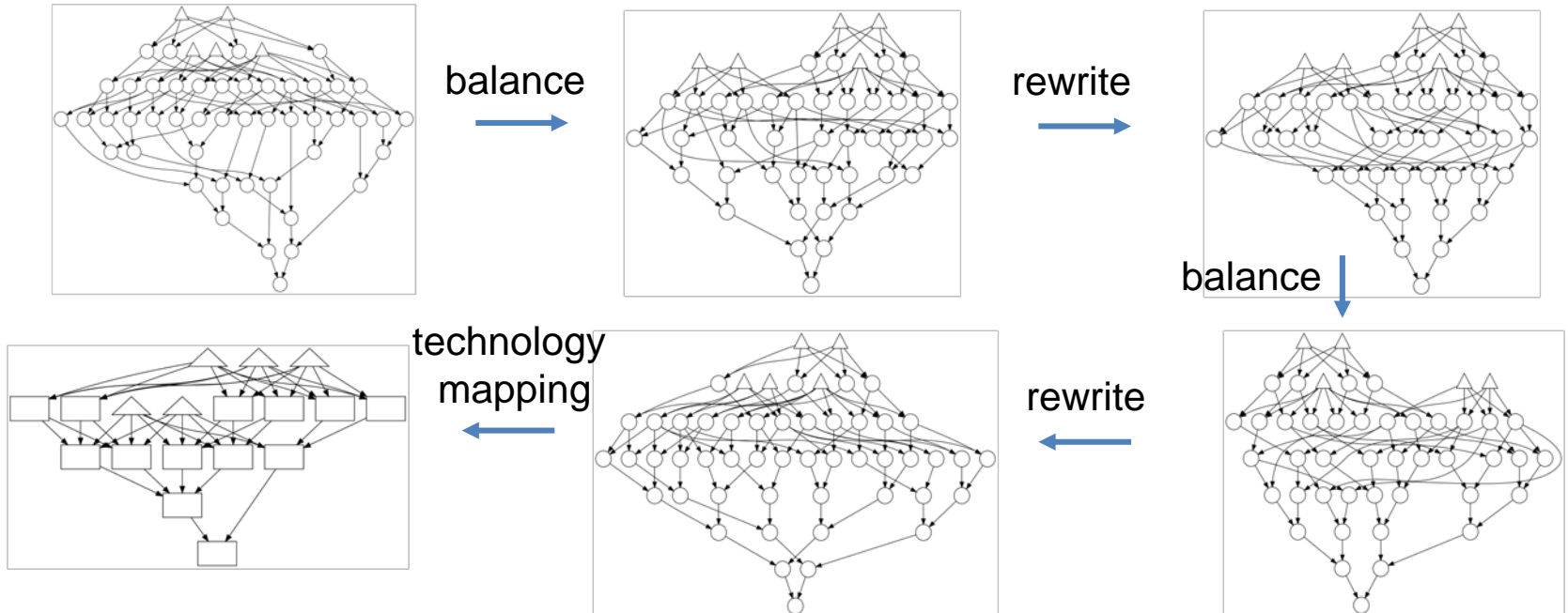
Typical Pre-mapping Transformation Sequence

- ▶ A typical area-minimizing script in ABC^[1]:

[1] Mishchenko, et al., TCAD'07

balance → rewrite → balance → rewrite

Initial and-inverter graph for *xor5*



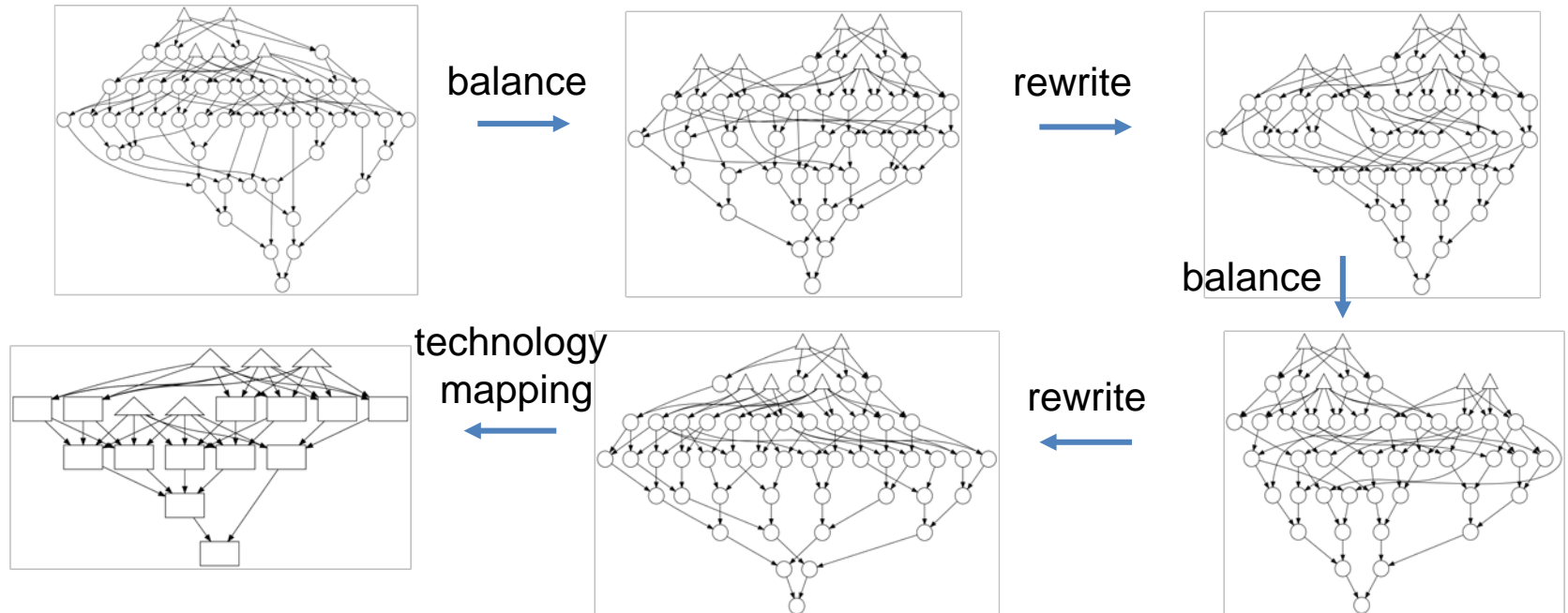
Typical Pre-mapping Transformation Sequence

- ▶ A typical area-minimizing script in ABC^[1]:

[1] Mishchenko, et al., TCAD'07

balance → rewrite → balance → rewrite

Initial and-inverter graph for *xor5*



Key rationale: smaller logic network → smaller post-mapping circuit

Study of (Mis)Correlation

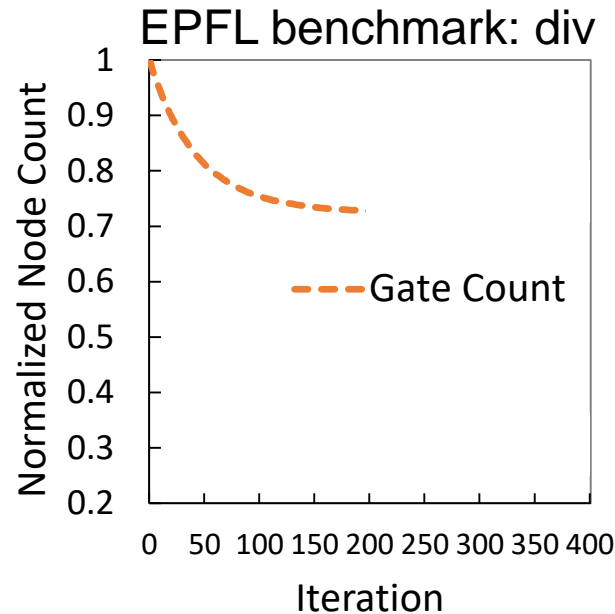
Key rationale of previous techniques:
smaller logic network → smaller post-mapping circuit

Study of (Mis)Correlation

Key rationale of previous techniques:
smaller logic network → smaller post-mapping circuit ?

Study of (Mis)Correlation

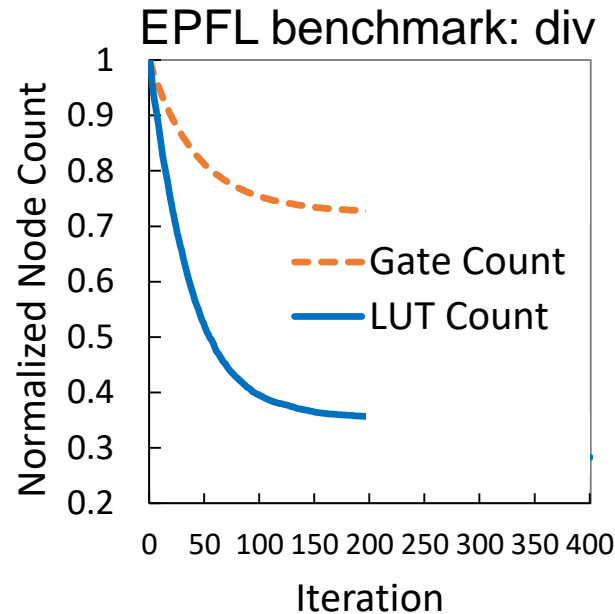
Key rationale of previous techniques:
smaller logic network \longrightarrow smaller post-mapping circuit ?



Methodology:
generate sequence of equivalent designs
with varying gate/LUT count

Study of (Mis)Correlation

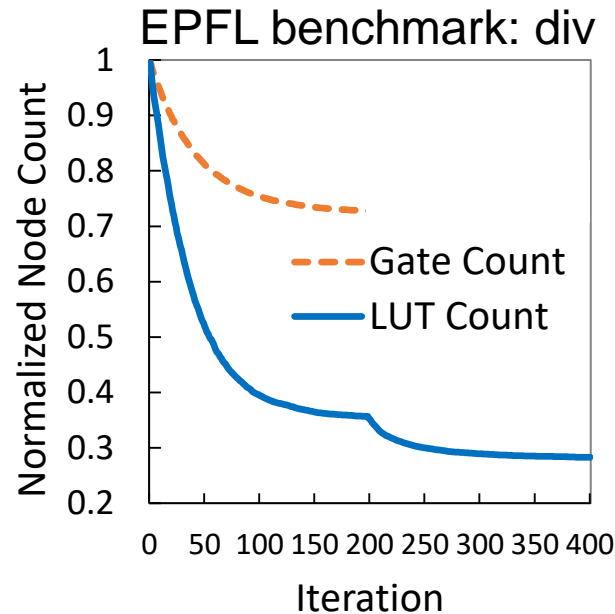
Key rationale of previous techniques:
smaller logic network \longrightarrow smaller post-mapping circuit ?



Methodology:
generate sequence of equivalent designs
with varying gate/LUT count

Study of (Mis)Correlation

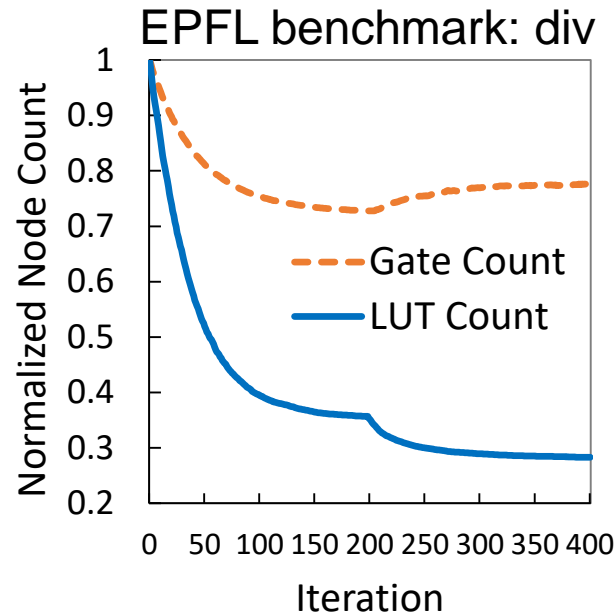
Key rationale of previous techniques:
smaller logic network \longrightarrow smaller post-mapping circuit ?



Methodology:
generate sequence of equivalent designs
with varying gate/LUT count

Study of (Mis)Correlation

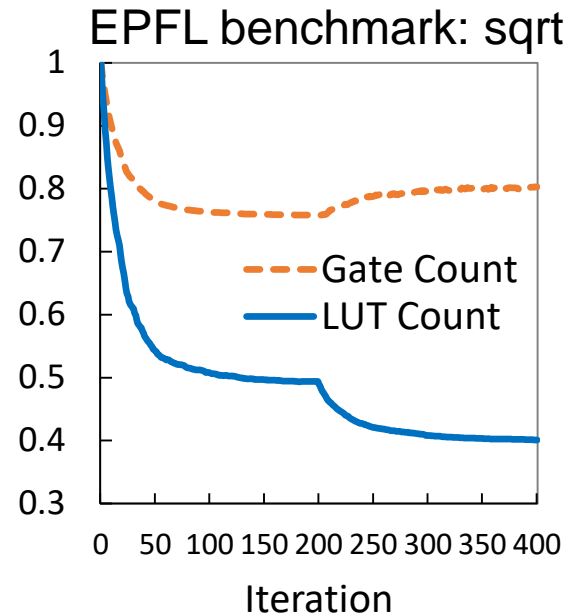
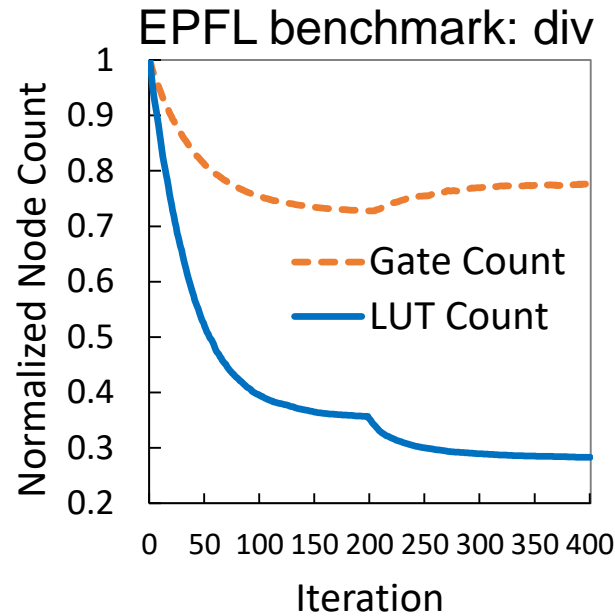
Key rationale of previous techniques:
smaller logic network \longrightarrow smaller post-mapping circuit ?



Methodology:
generate sequence of equivalent designs
with varying gate/LUT count

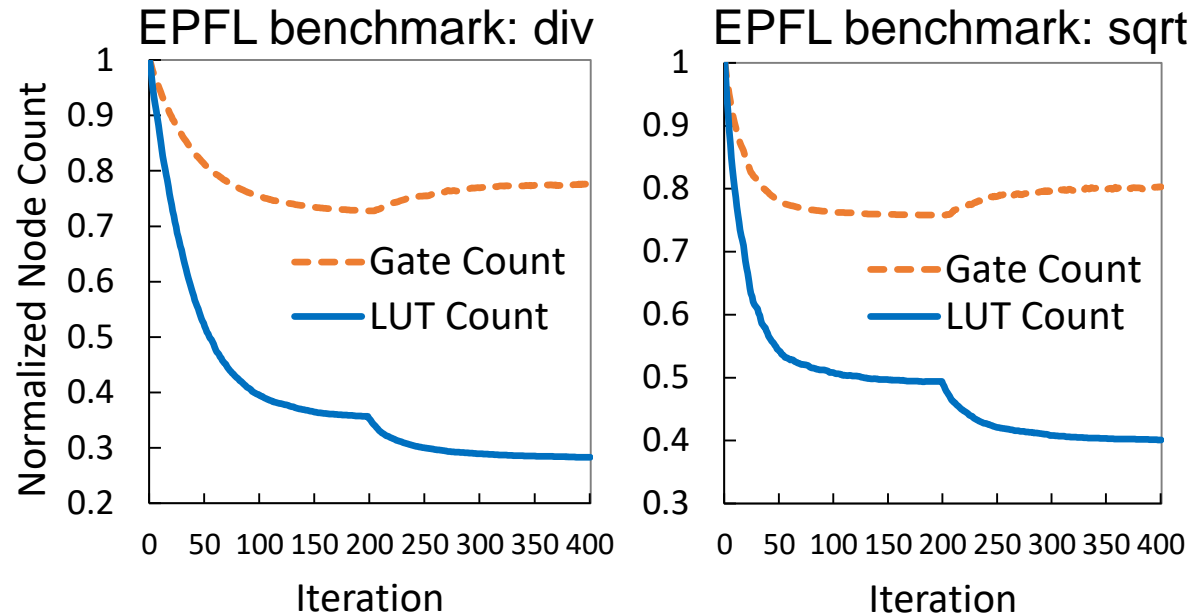
Study of (Mis)Correlation

Key rationale of previous techniques:
smaller logic network \longrightarrow smaller post-mapping circuit ?



Study of (Mis)Correlation

Key rationale of previous techniques:
smaller logic network \longrightarrow smaller post-mapping circuit ?



Our study:
smaller logic network **not necessarily** leads to smaller post-mapping circuit

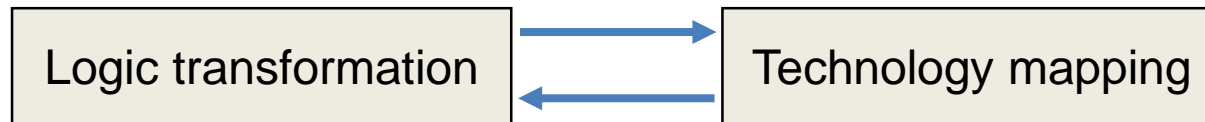
PIMap: A Parallelized Iterative Improvement Approach to LUT-Based Tech Mapping

- ▶ Couple mapping and logic transformation
 - Close the gap between logic optimization and tech mapping
 - Incrementally improve area

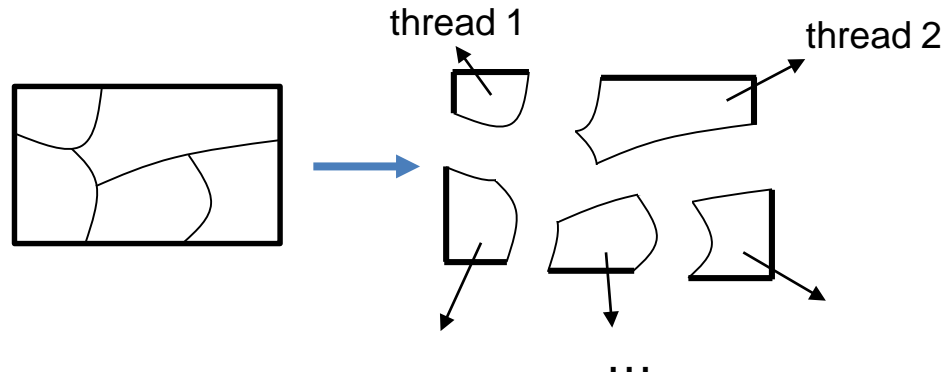


PIMap: A Parallelized Iterative Improvement Approach to LUT-Based Tech Mapping

- ▶ Couple mapping and logic transformation
 - Close the gap between logic optimization and tech mapping
 - Incrementally improve area



- ▶ Effective partitioning and parallelization technique
 - Improve both runtime and design quality

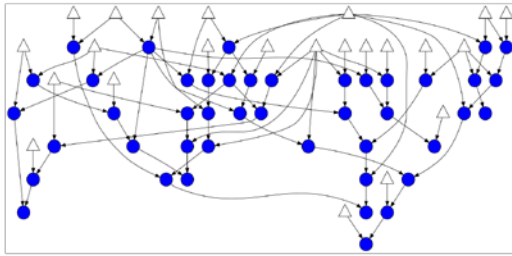


PIMap Technique: Iterative Area Minimization

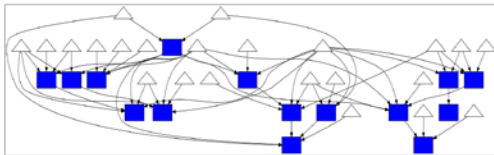
Intuition: use mapping result to
guide randomly proposed logic
transformations

PIMap Technique: Iterative Area Minimization

Intuition: use mapping result to
guide randomly proposed logic
transformations

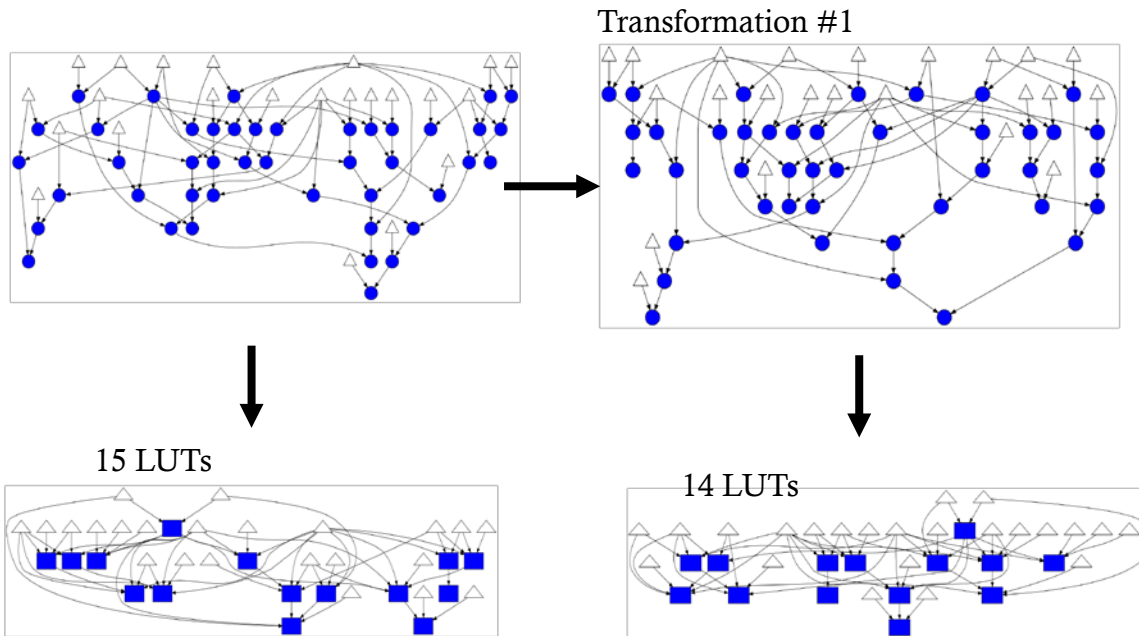


15 LUTs



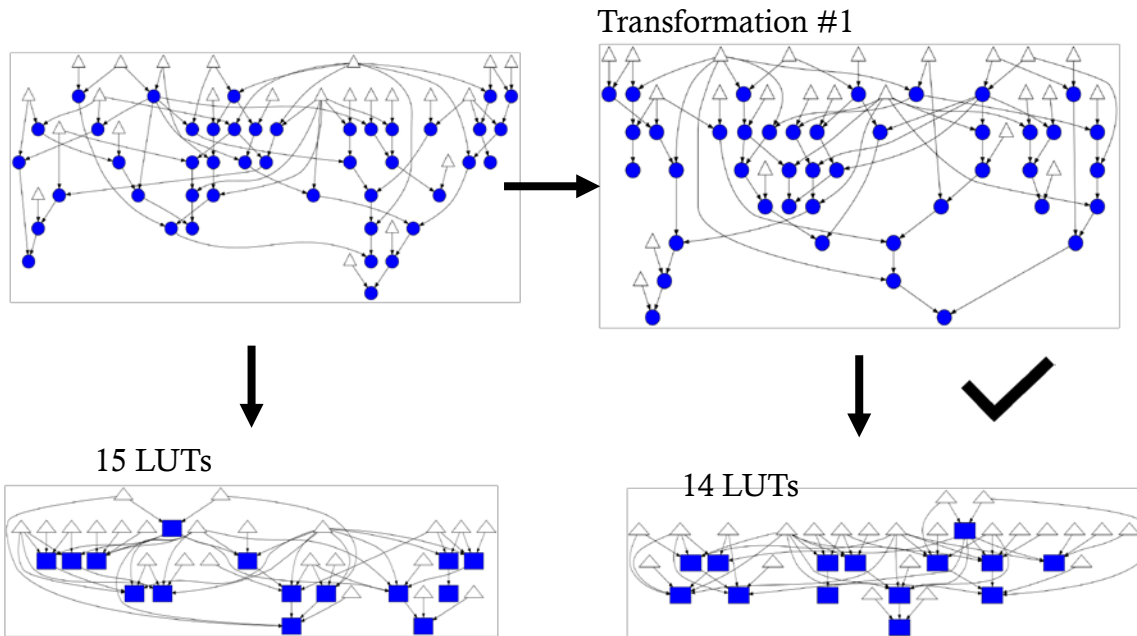
PIMap Technique: Iterative Area Minimization

Intuition: use mapping result to
guide randomly proposed logic
transformations



PIMap Technique: Iterative Area Minimization

Intuition: use mapping result to guide randomly proposed logic transformations



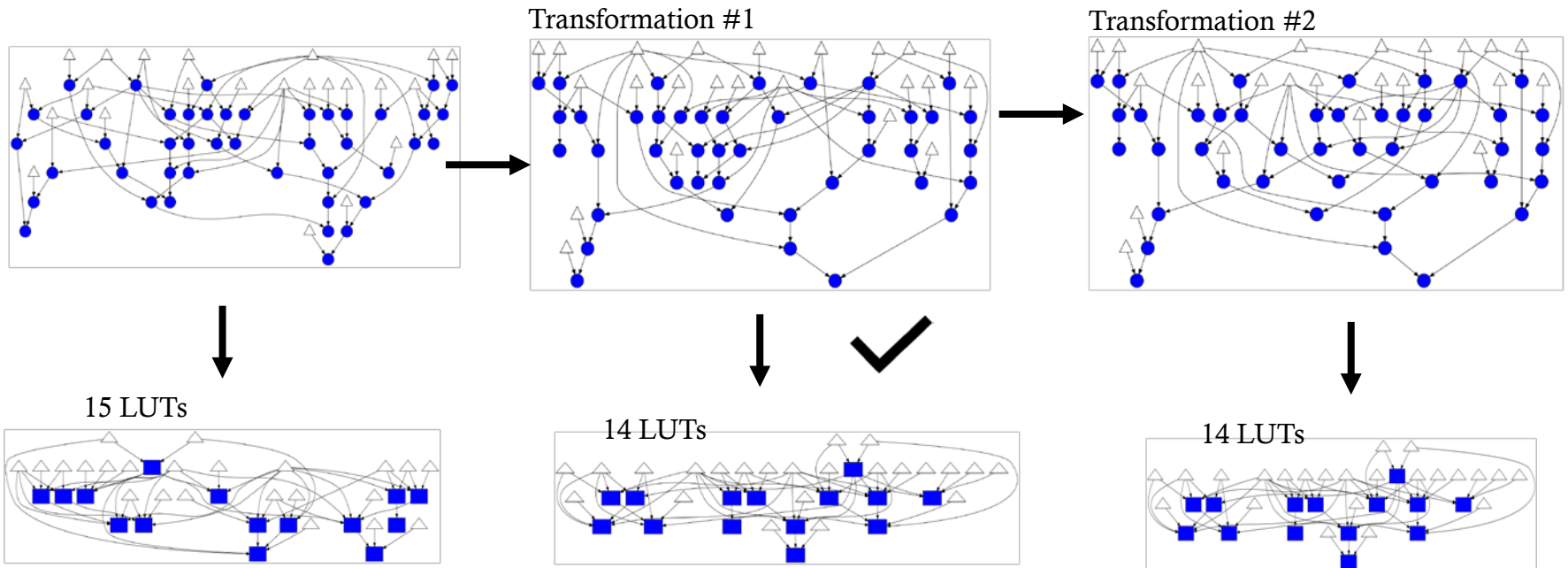
Metropolis-Hastings algorithm^[1]:

Accept current transformation if $\text{rand}(0,1) < \exp\left(-\gamma \frac{N_{LUT_new}}{N_{LUT_old}}\right)$

[1] Hastings, Biometrika'70

PIMap Technique: Iterative Area Minimization

Intuition: use mapping result to guide randomly proposed logic transformations

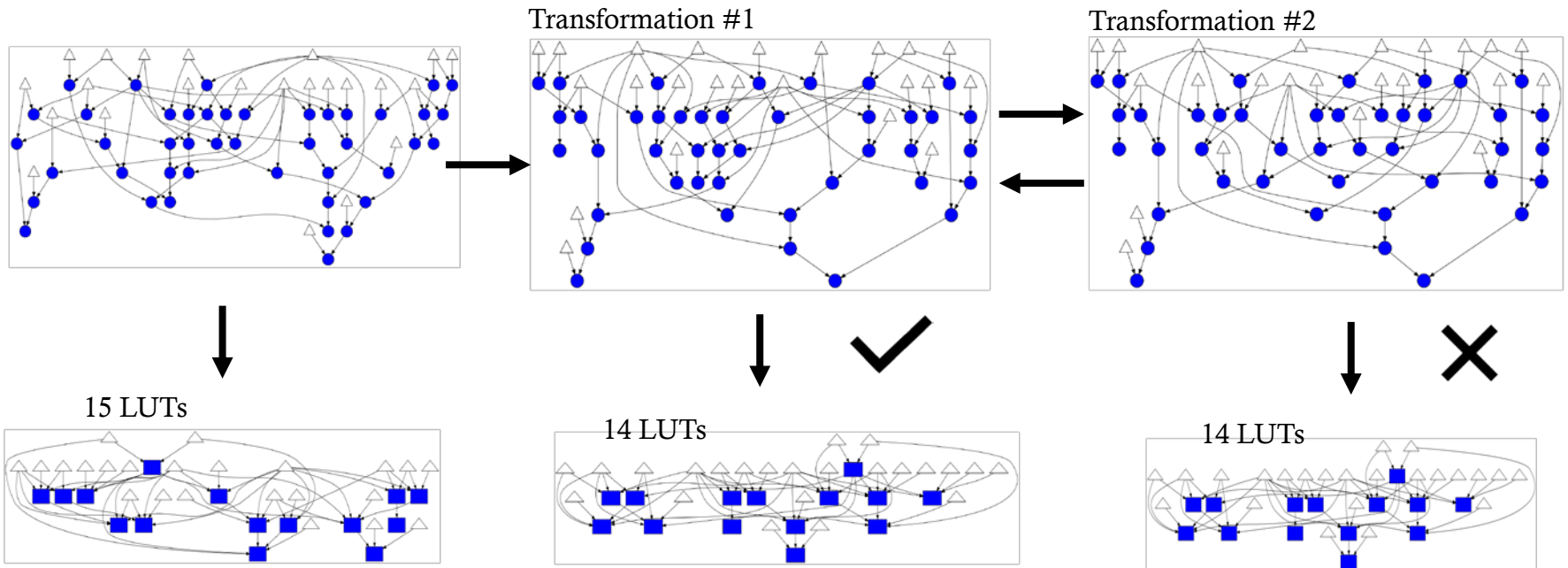


Metropolis-Hastings algorithm^[1]:
Accept current transformation if $rand(0,1) < \exp\left(-\gamma \frac{N_{LUT_new}}{N_{LUT_old}}\right)$

[1] Hastings, Biometrika'70

PIMap Technique: Iterative Area Minimization

Intuition: use mapping result to guide randomly proposed logic transformations

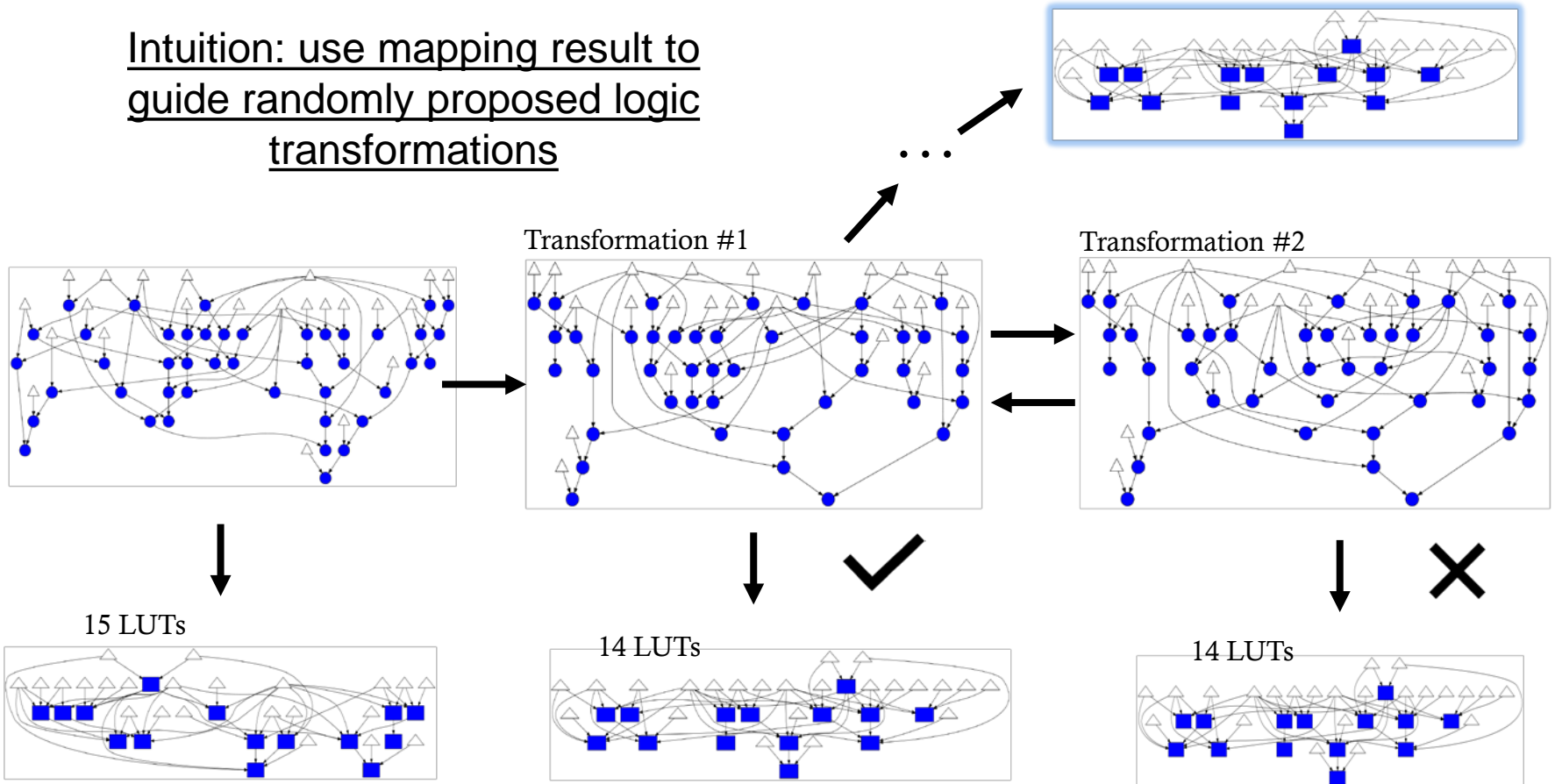


Metropolis-Hastings algorithm^[1]:
Accept current transformation if $rand(0,1) < \exp(-\gamma \frac{N_{LUT_new}}{N_{LUT_old}})$

[1] Hastings, Biometrika'70

PIMap Technique: Iterative Area Minimization

Intuition: use mapping result to guide randomly proposed logic transformations

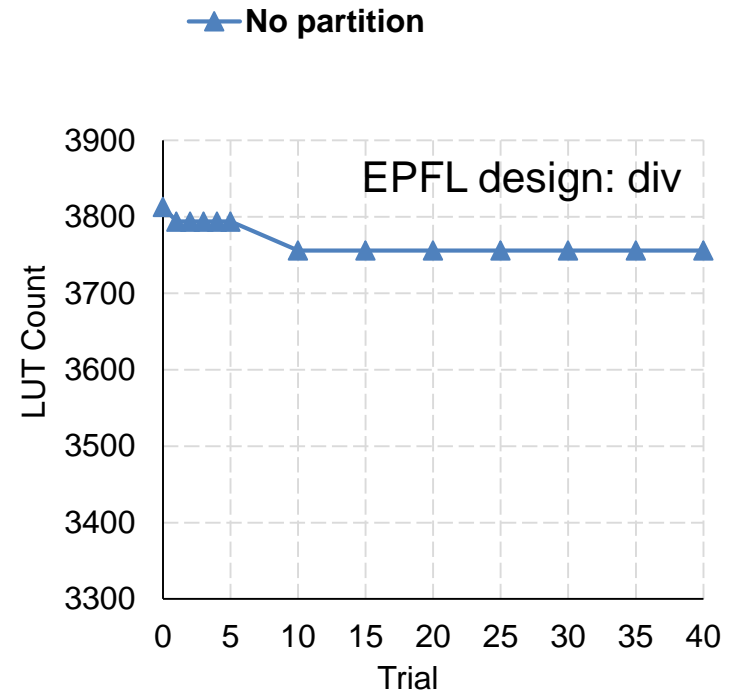


Metropolis-Hastings algorithm^[1]:
Accept current transformation if $rand(0,1) < \exp(-\gamma \frac{N_{LUT_new}}{N_{LUT_old}})$

[1] Hastings, Biometrika'70

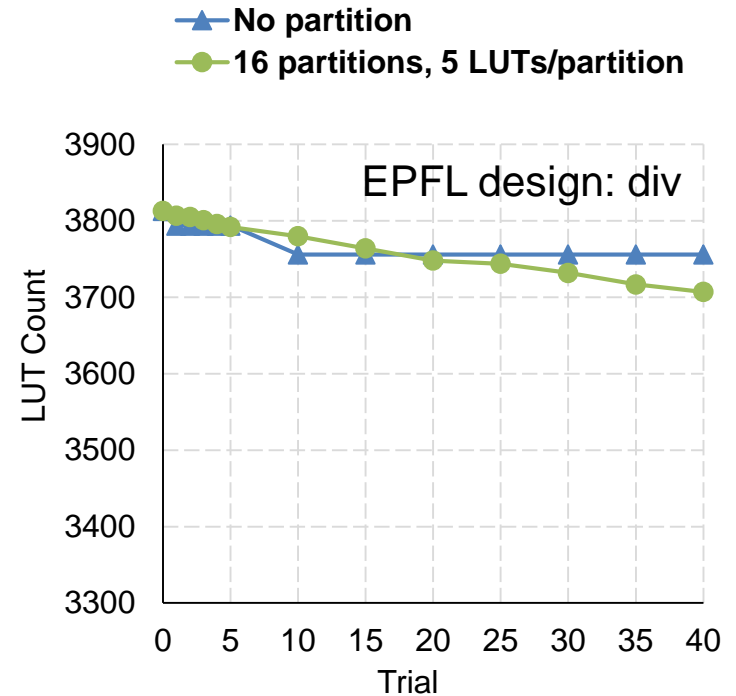
Partitioning Schemes

- ▶ No circuit partitioning
 - Long runtime per trial
 - Easily stuck at local minimum



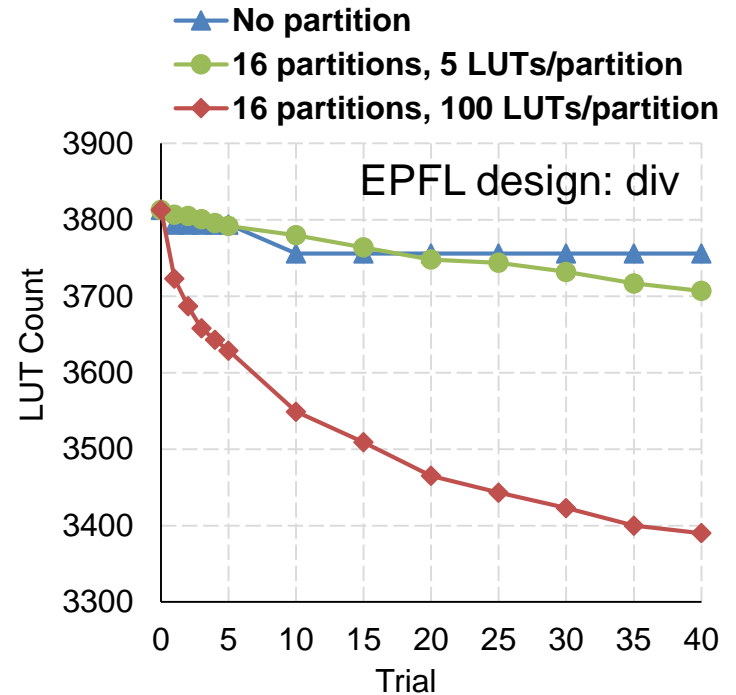
Partitioning Schemes

- ▶ No circuit partitioning
 - Long runtime per trial
 - Easily stuck at local minimum
- ▶ Fine-grained partition
 - Similar concept to exact synthesis
 - Fast runtime per trial
 - Slow progress overall



Partitioning Schemes

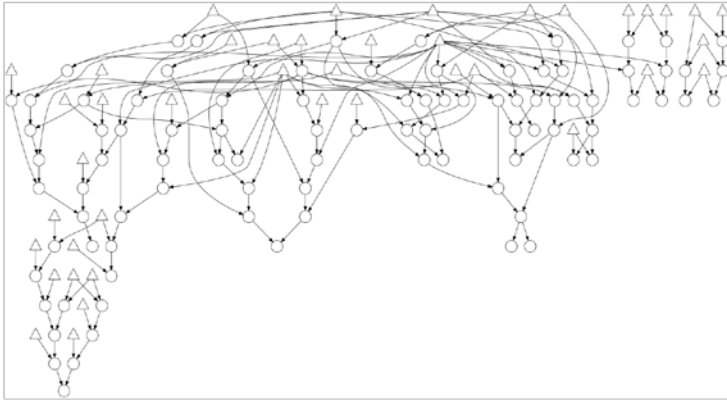
- ▶ No circuit partitioning
 - Long runtime per trial
 - Easily stuck at local minimum
- ▶ Fine-grained partition
 - Similar concept to exact synthesis
 - Fast runtime per trial
 - Slow progress overall
- ▶ **Coarse-grained partition**
 - Balance runtime and solution quality
 - Repartition between trials to further improve quality



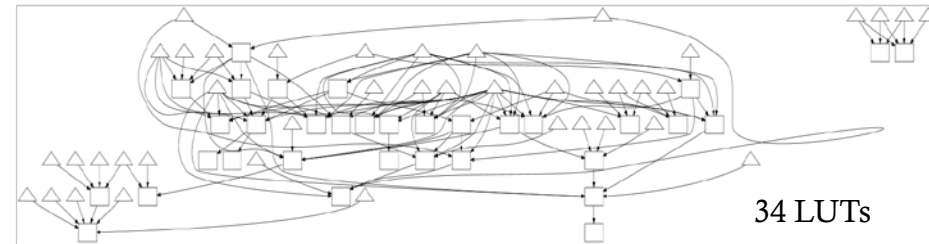
PIMap Technique: Partitioning and Parallelization

Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs

AIG of design b9

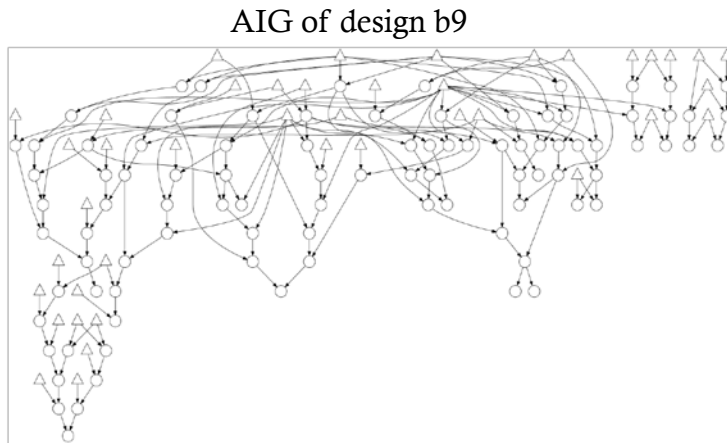


Mapped netlist

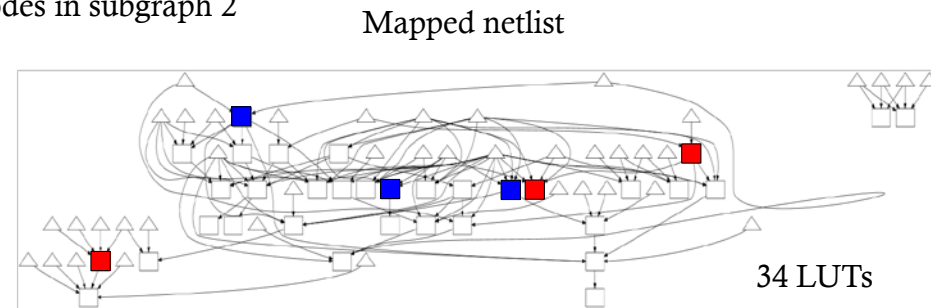


PIMap Technique: Partitioning and Parallelization

Initial mapping to LUT **Subgraph extraction** Iterative area minimization Recombine subgraphs

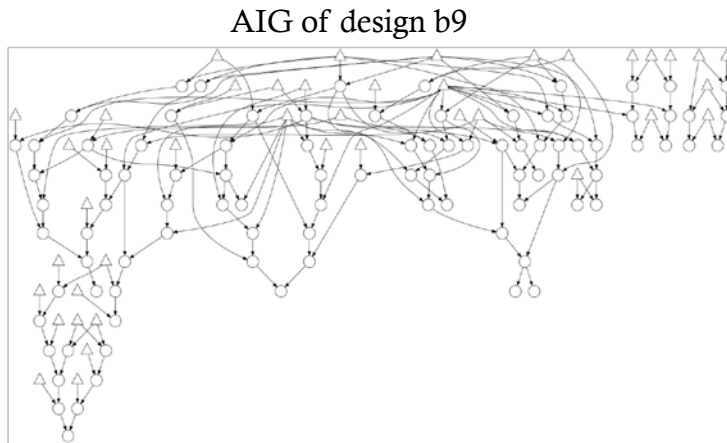


- Nodes in subgraph 1
- Nodes in subgraph 2

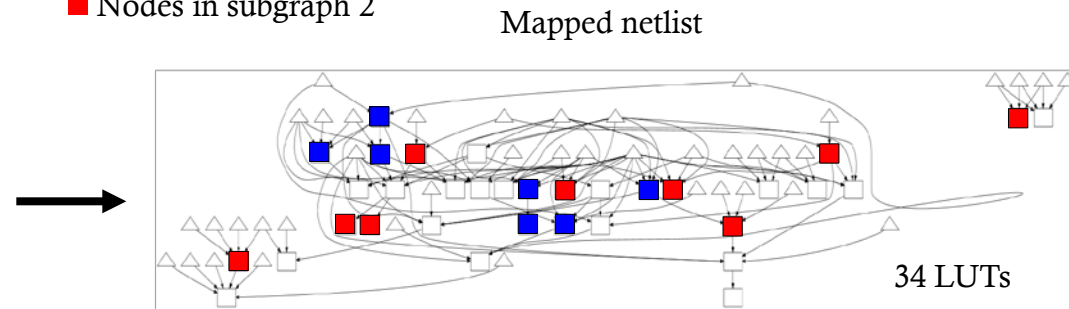


PIMap Technique: Partitioning and Parallelization

Initial mapping to LUT **Subgraph extraction** Iterative area minimization Recombine subgraphs

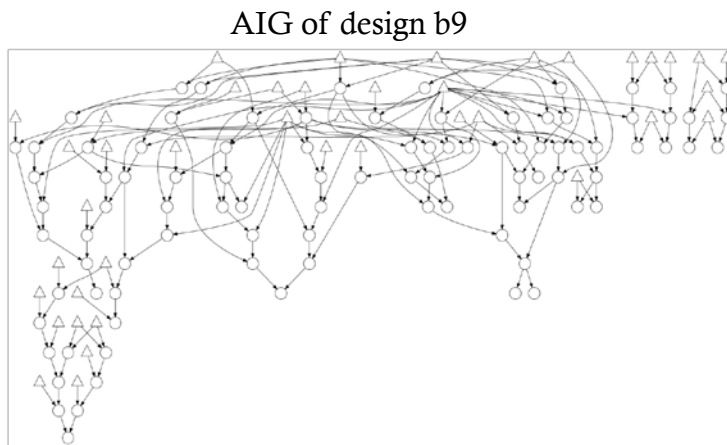


- Nodes in subgraph 1
- Nodes in subgraph 2

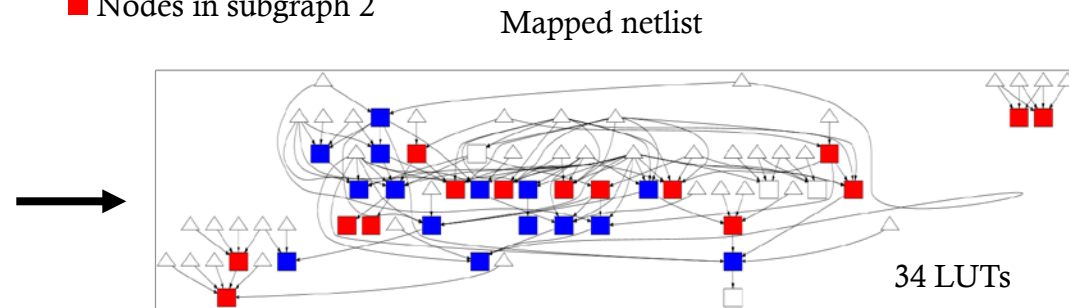


PIMap Technique: Partitioning and Parallelization

Initial mapping to LUT **Subgraph extraction** Iterative area minimization Recombine subgraphs

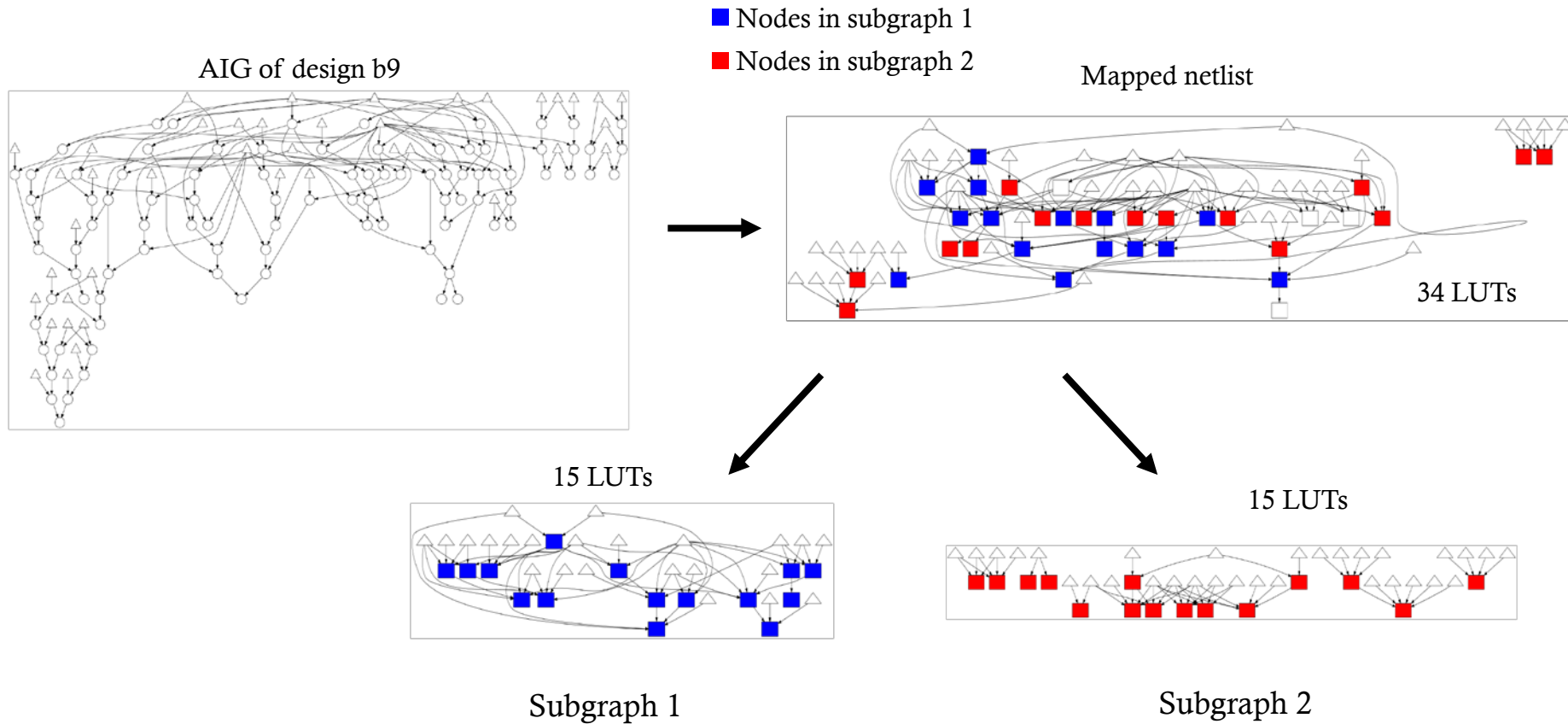


- Nodes in subgraph 1
- Nodes in subgraph 2



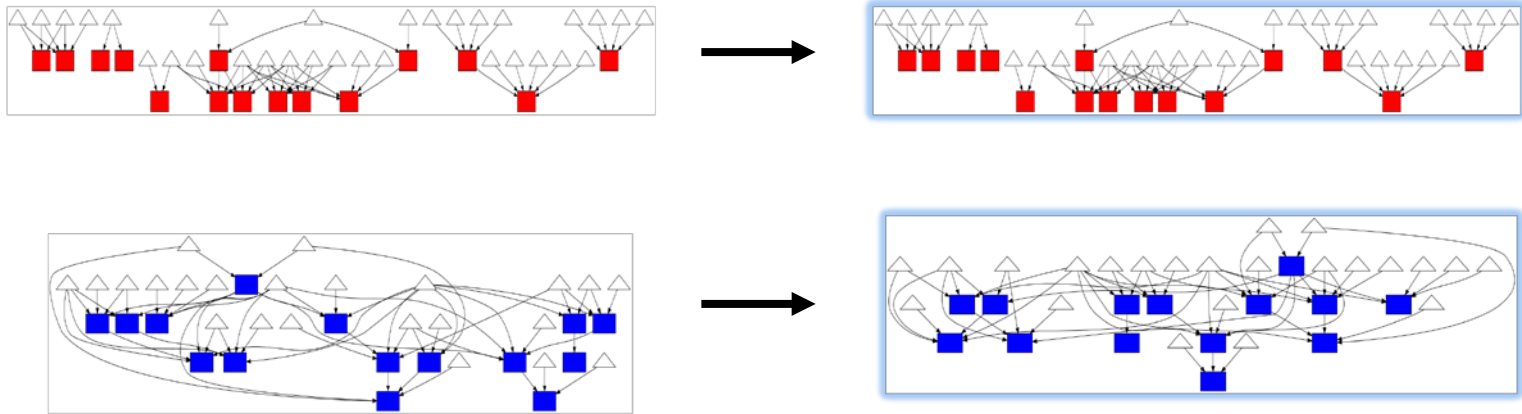
PIMap Technique: Partitioning and Parallelization

Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs



PIMap Technique: Partitioning and Parallelization

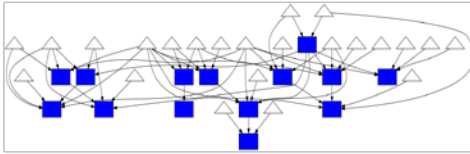
Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs



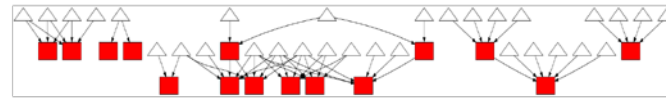
PIMap Technique: Partitioning and Parallelization

Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs

14 LUTs

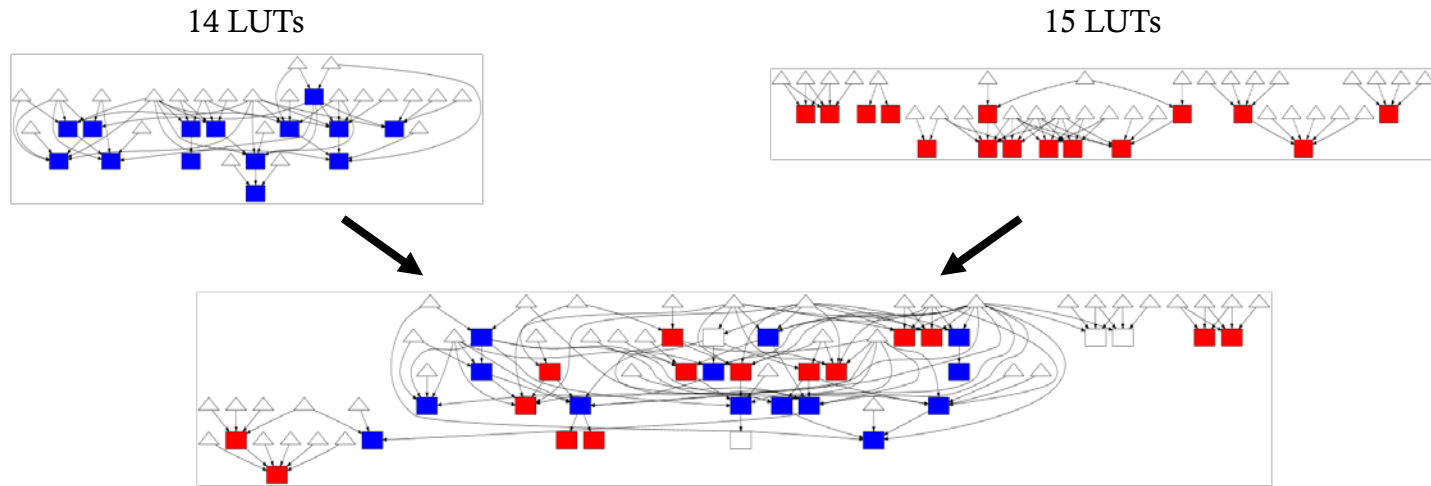


15 LUTs



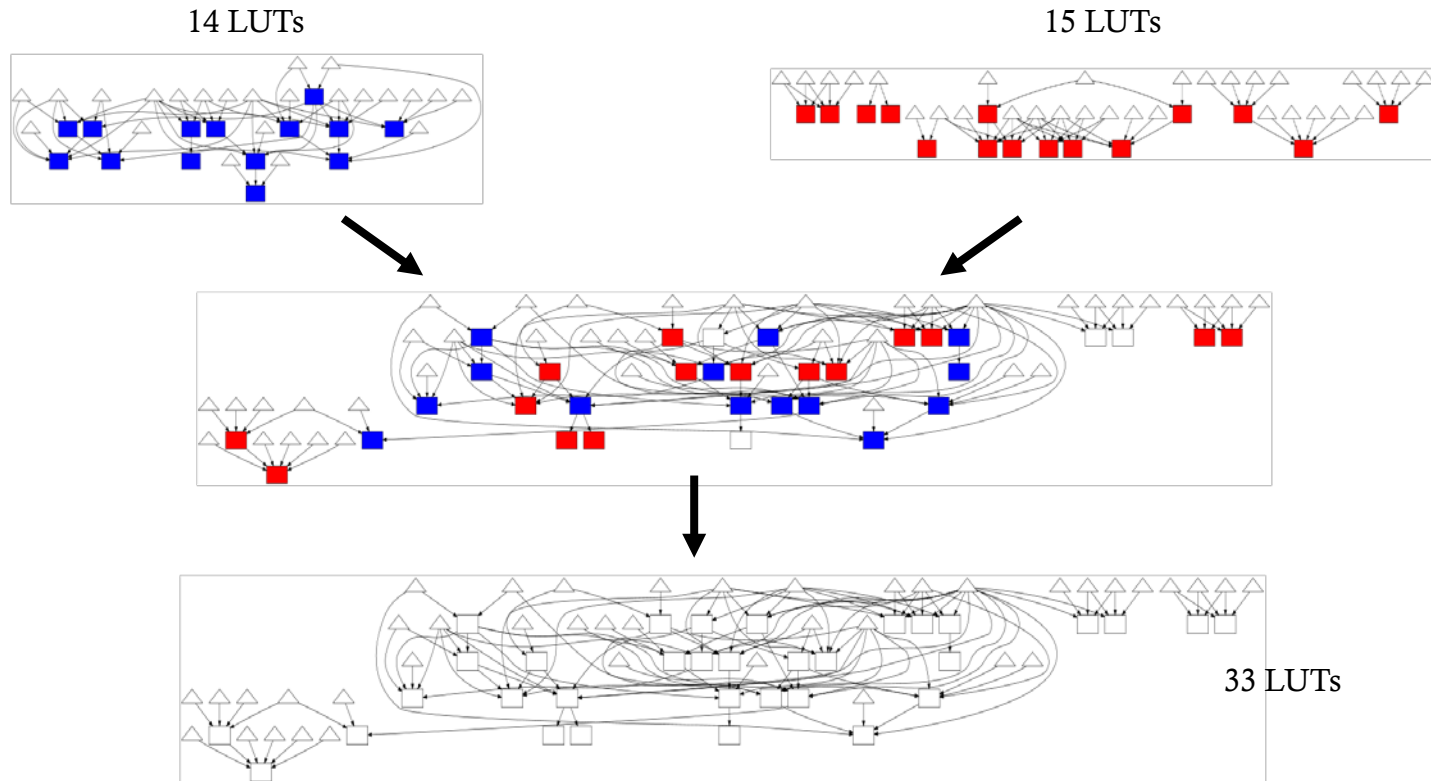
PIMap Technique: Partitioning and Parallelization

Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs



PIMap Technique: Partitioning and Parallelization

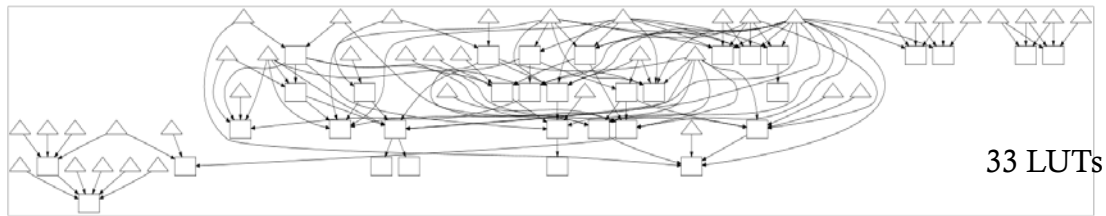
Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs



PIMap Technique: Repartition

Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs

Optimized design after trial 1

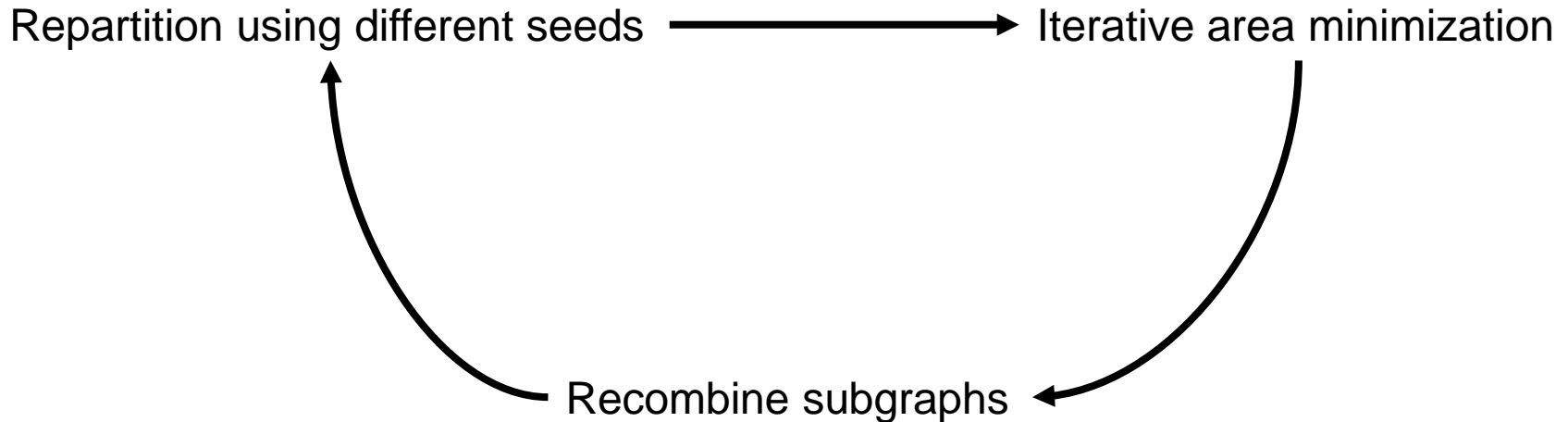
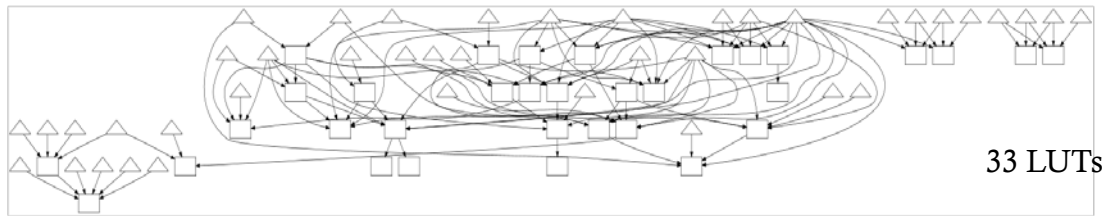


Repartition using different seeds

PIMap Technique: Repartition

Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs

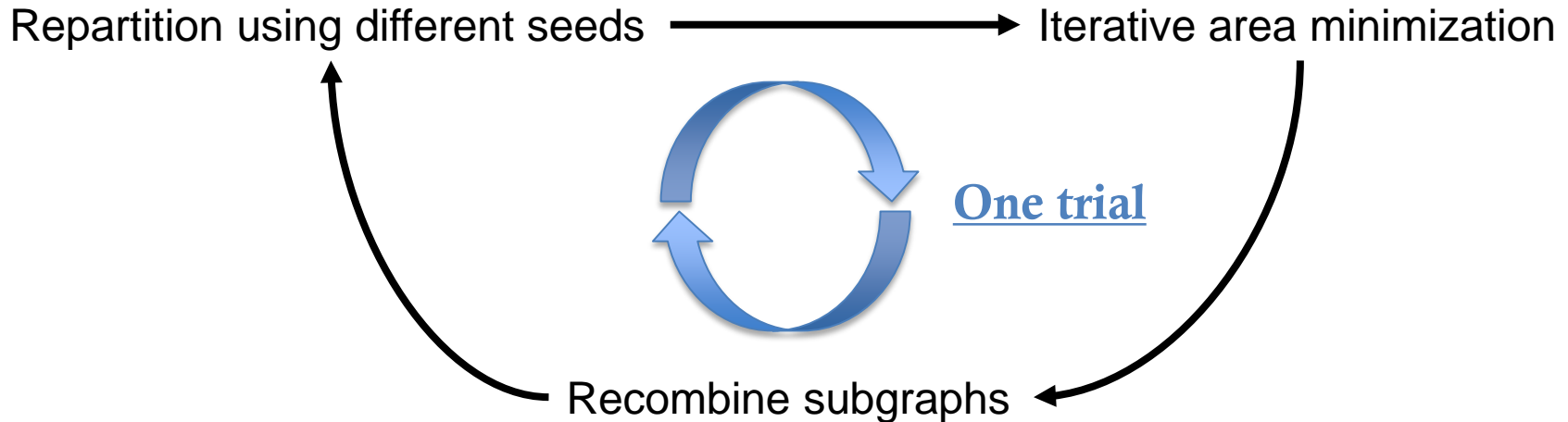
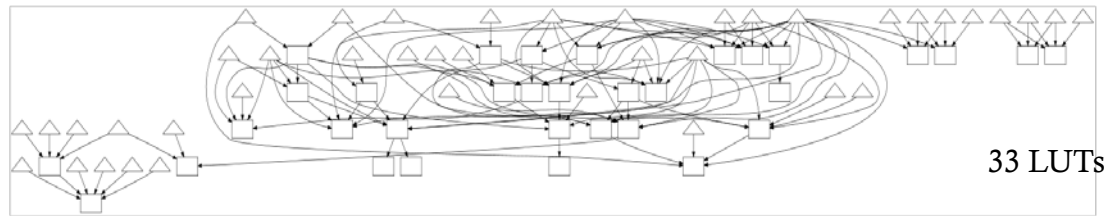
Optimized design after trial 1



PIMap Technique: Repartition

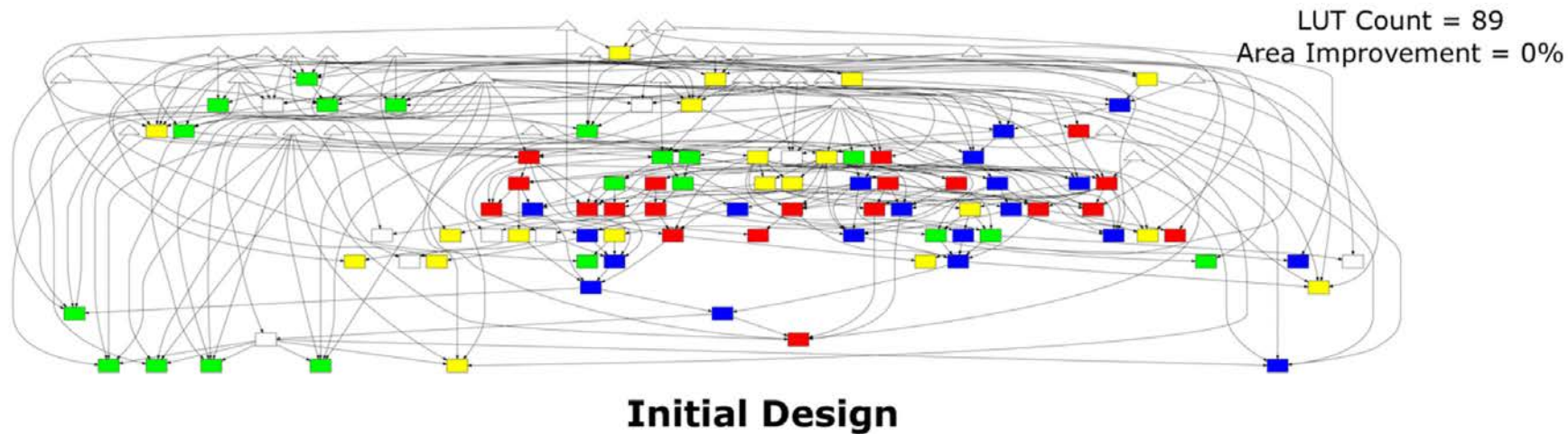
Initial mapping to LUT Subgraph extraction Iterative area minimization Recombine subgraphs

Optimized design after trial 1



PIMap Overall Flow

Design C1908 from the MCNC benchmark suite
5 trials in total

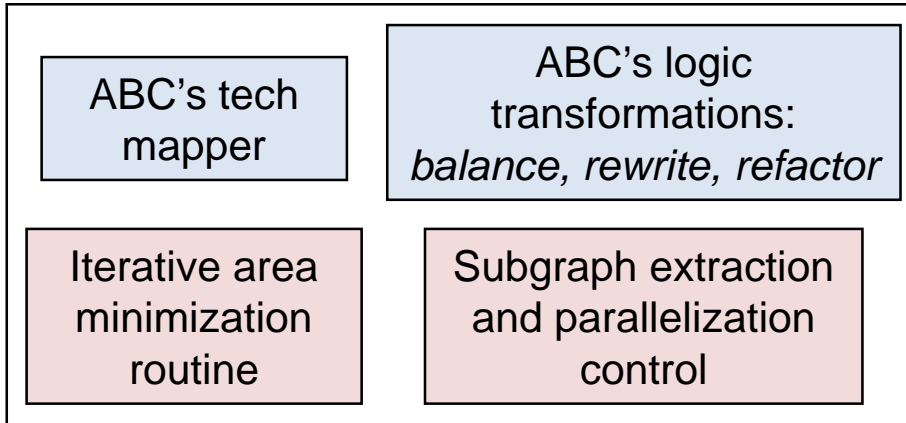


Observations:

1. Partition boundaries vary between trials
 → Uncover better structure
2. Overall network structure differ significantly between trials
 → Discover a wide range of designs

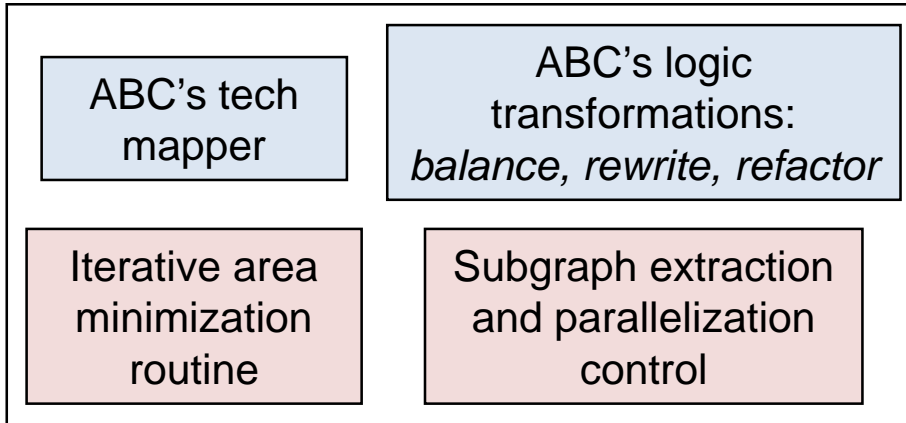
Experimental Setup

PIMap toolchain



Experimental Setup

PIMap toolchain



Benchmarks

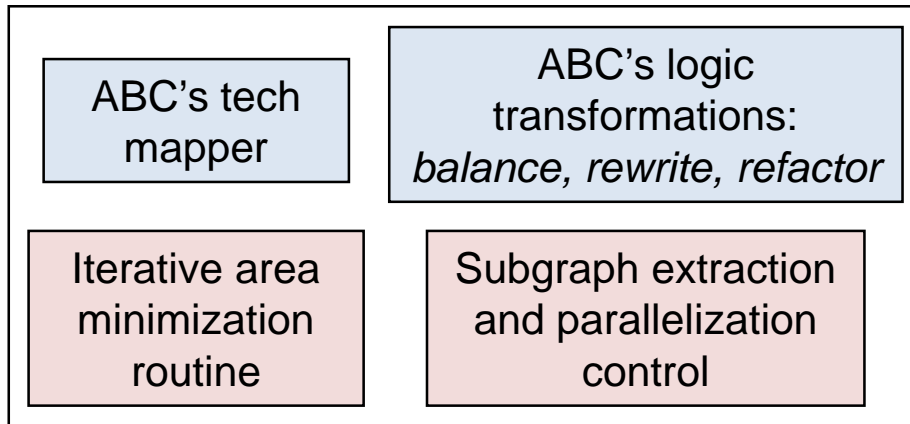
Benchmark	Initial design
10 largest MCNC designs ^[1]	pre-synthesized using ABC's <i>compress2rs</i> script
EPFL arithmetic designs ^[2]	best-known mapping designs ^[2]

[1] Yang, MCNC'91

[2] Amarù, et al., <http://lsi.epfl.ch/benchmarks>

Experimental Setup

PIMap toolchain



Benchmarks

Benchmark	Initial design
10 largest MCNC designs [1]	pre-synthesized using ABC's <i>compress2rs</i> script
EPFL arithmetic designs [2]	best-known mapping designs [2]

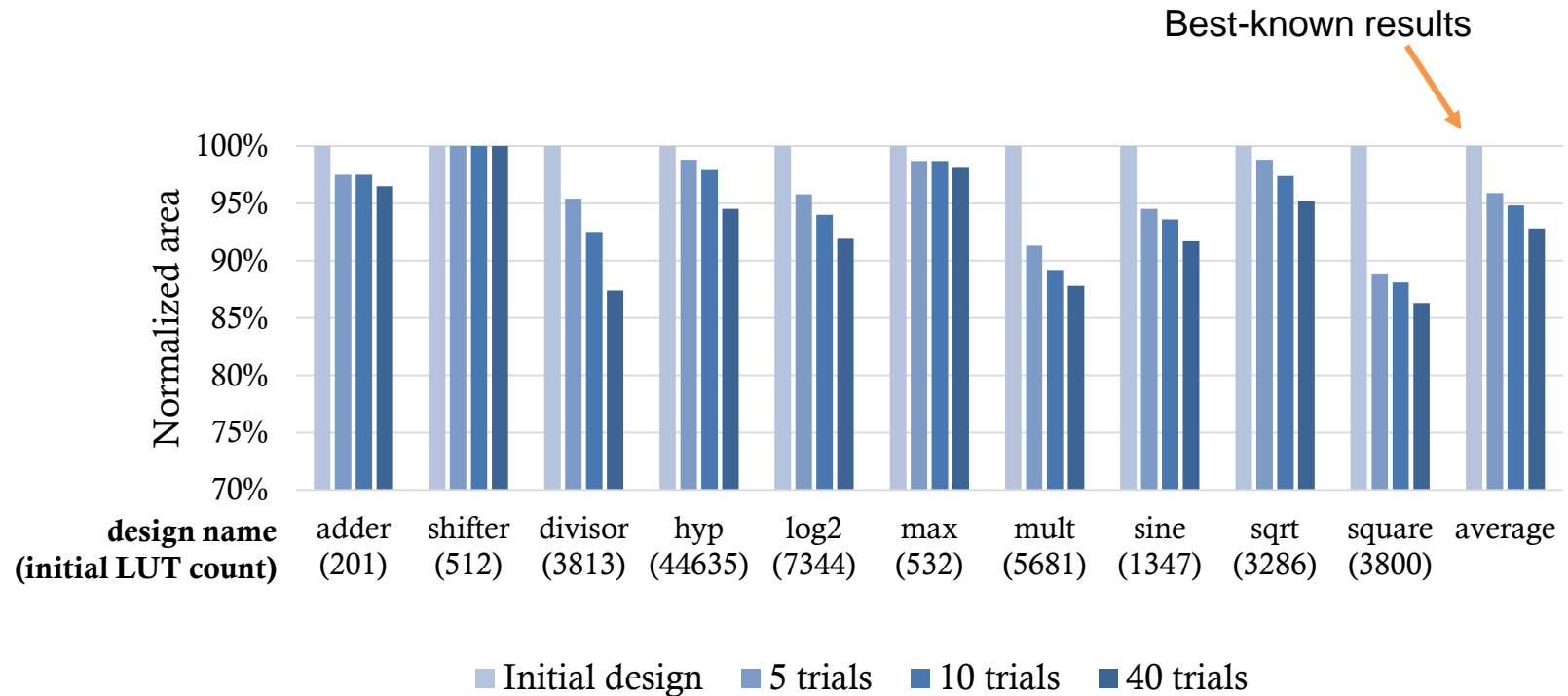
[1] Yang, MCNC'91

[2] Amarù, et al., <http://lsi.epfl.ch/benchmarks>

Setup

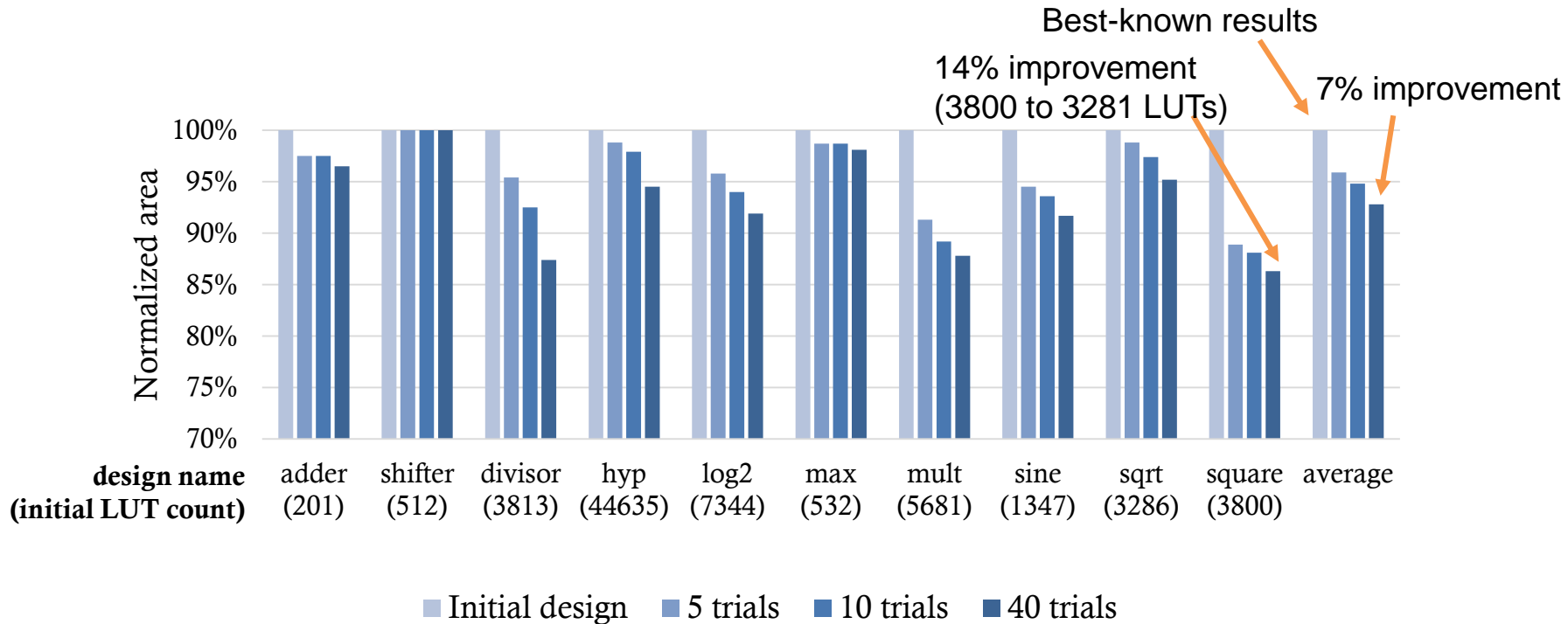
Configuration	40 trials, 100 iterations of area minimization per trial
Partitioning	up to 16 subgraphs, each with up to 100 LUTs
Computing resource	up to 8 machines, each with a quad-core Xeon processor

Unconstrained Area Minimization



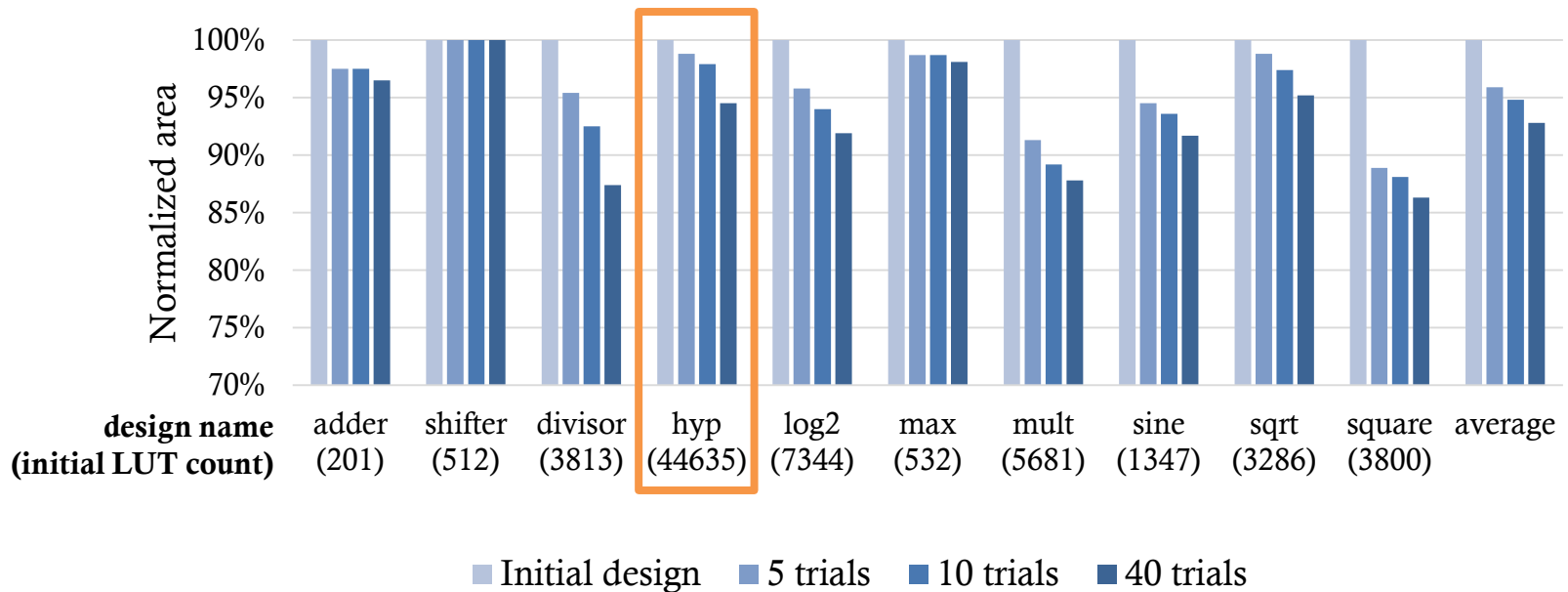
- ▶ Initial design: best-known results from EPFL record

Unconstrained Area Minimization



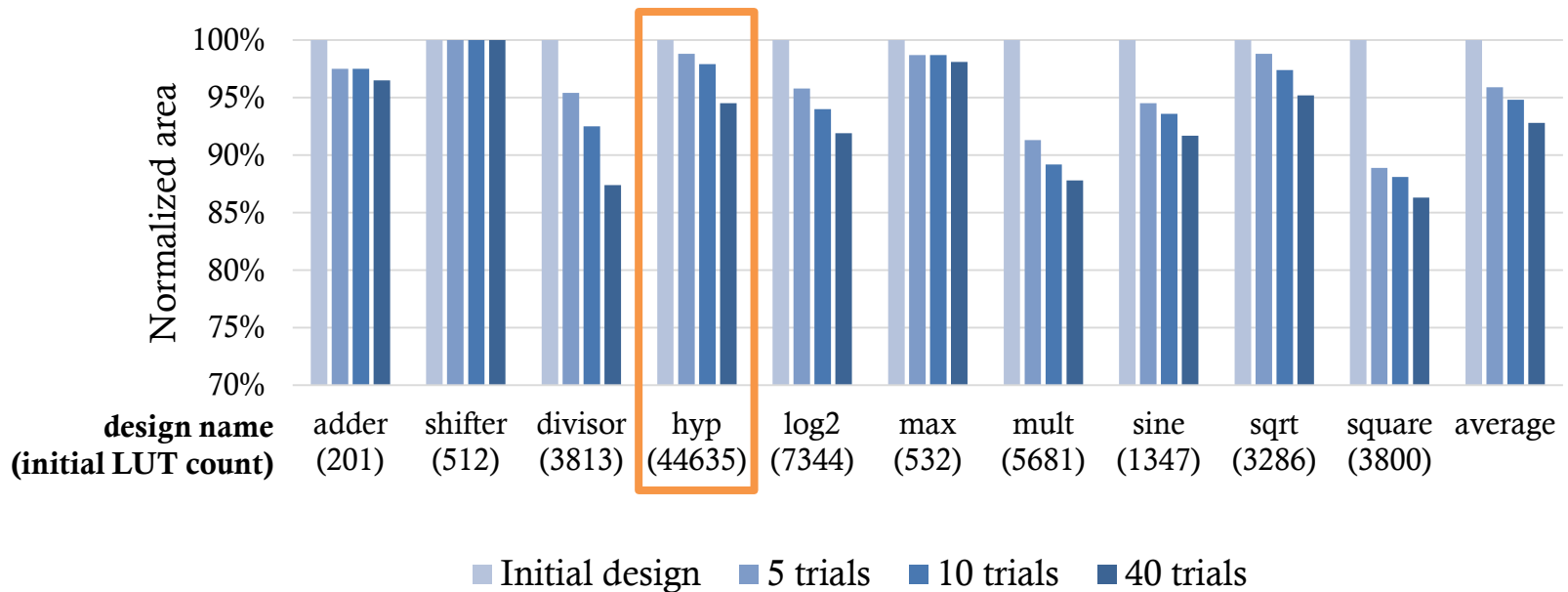
- ▶ Initial design: best-known results from EPFL record
- ▶ Area improvements
 - EPFL: 7% on average, up to 14%

Unconstrained Area Minimization



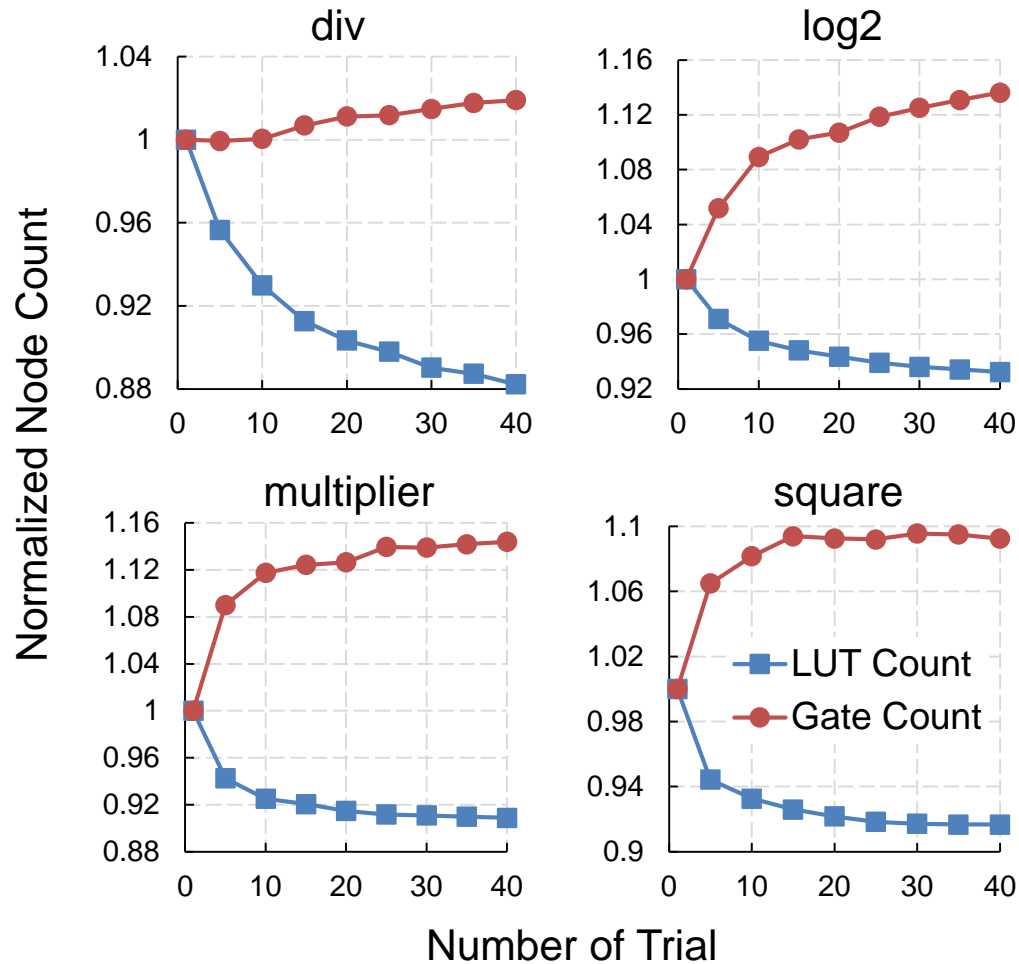
- ▶ Initial design: best-known results from EPFL record
- ▶ Area improvements
 - EPFL: 7% on average, up to 14%
 - Can effectively handle very large circuit (~44k LUTs)

Unconstrained Area Minimization



- ▶ Initial design: best-known results from EPFL record
- ▶ Area improvements
 - EPFL: 7% on average, up to 14%
 - Can effectively handle very large circuit (~44k LUTs)
- ▶ Also able to improve all 10 control-intensive designs in EPFL benchmark suite

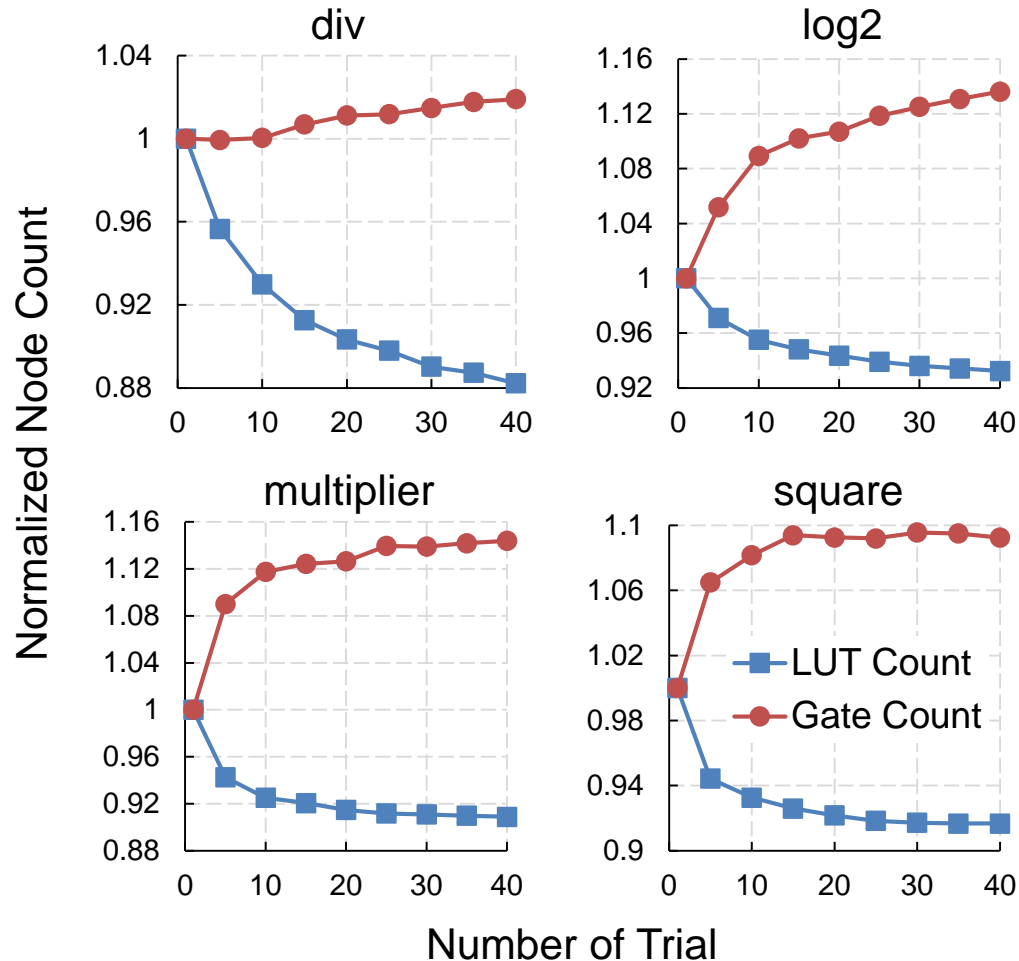
LUT Count vs. Gate Count Reduction



LUT Count vs. Gate Count Reduction

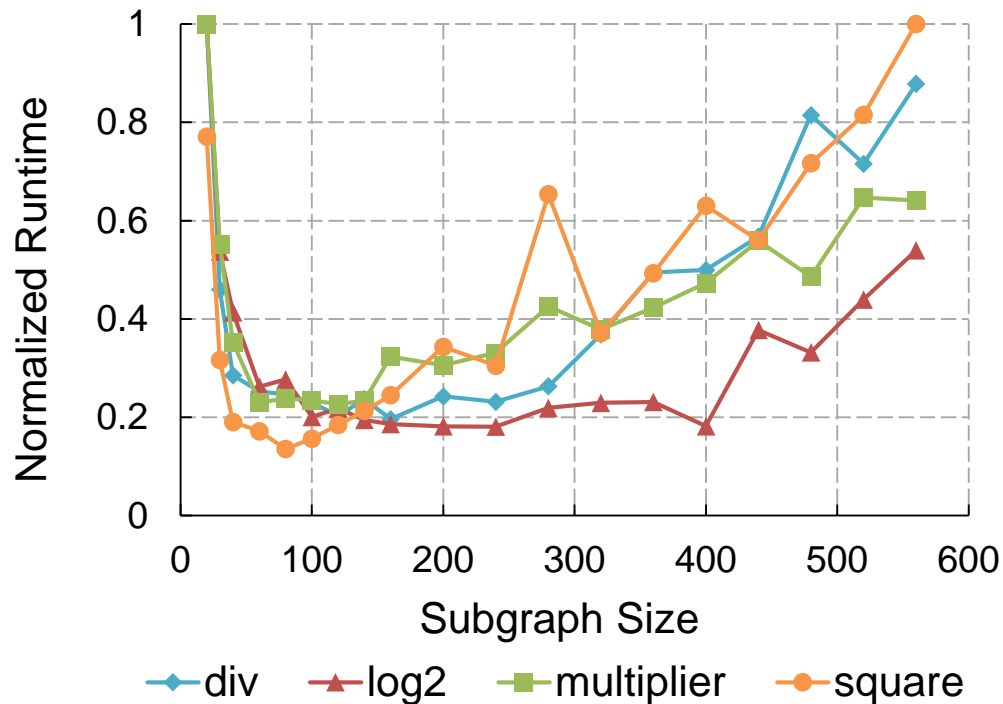
Verified:

post-mapping area does not necessarily correlate with pre-mapping area



Subgraph Size vs. Runtime

- ▶ Tradeoff between runtime vs. progress per trial
 - Optimal subgraph size is around 100 LUTs

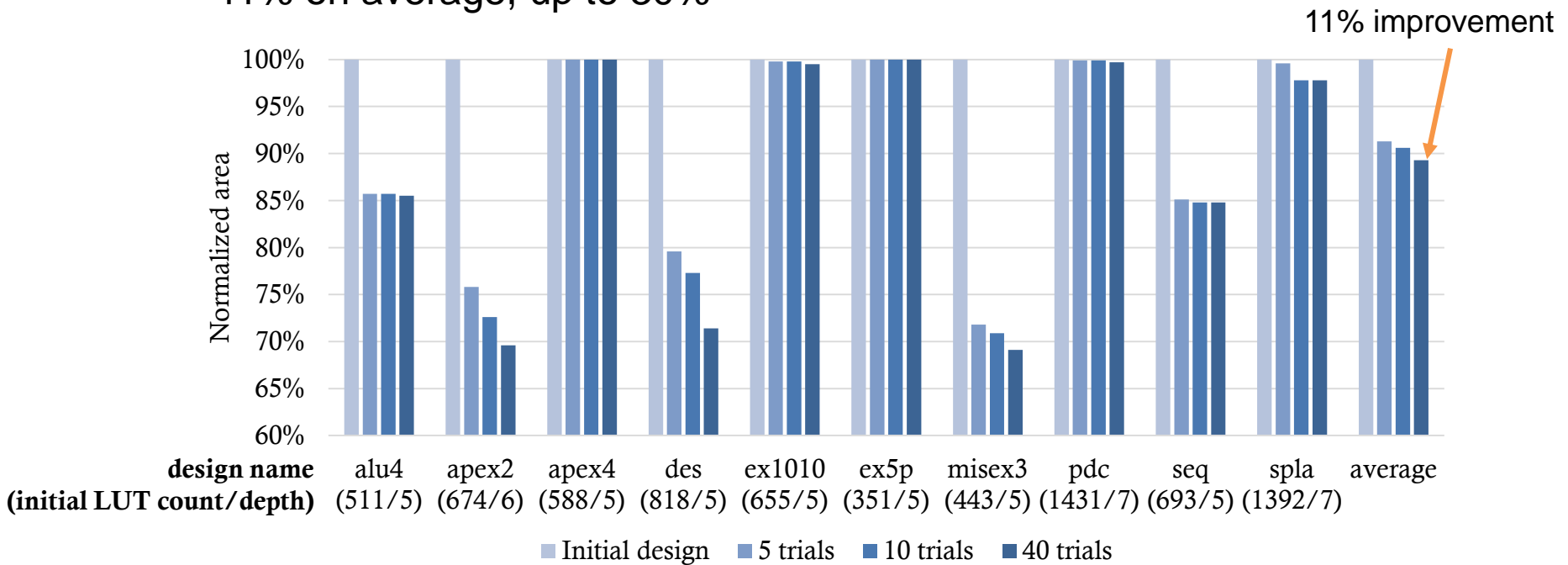


Depth Constrained Area Minimization

- ▶ Constraint: no depth increase compared to initial design
 - Initial designs generated by ABC's depth-minimizing *resyn2* script
 - In PIMap, only accept designs within depth constraint after each trial

Depth Constrained Area Minimization

- ▶ Constraint: no depth increase compared to initial design
 - Initial designs generated by ABC's depth-minimizing *resyn2* script
 - In PIMap, only accept designs within depth constraint after each trial
- ▶ Area improvements under depth constraint
 - 11% on average, up to 30%



Area Reduction under a Tight Runtime Limit

- ▶ In use cases with tight runtime budget
 - Use fewer number of trials and fewer iterations per trial

Area Reduction under a Tight Runtime Limit

- ▶ In use cases with tight runtime budget
 - Use fewer number of trials and fewer iterations per trial
 - PIMap still able to improve most of the best-known results of EPFL benchmark designs

Area reduction using PIMap with tight runtime limit

Designs	Best-known	PIMap
Adder	201	197
Shifter	512	512
Divisor	3813	3787
Hyp	44635	44635
Log2	7344	7305
Max	532	526
Mult	5681	5594
Sine	1347	1309
Sqrt	3286	3279
square	3800	3675

Runtime limit:
10 seconds

Conclusions

- ▶ Circuit area before/after mapping does not necessarily correlate
- ▶ Stochastic mapping-in-the-loop approach for area minimization
- ▶ Sub-circuit extraction and parallelization for runtime reduction
- ▶ Up to 14% and 7% on average over the best-known records for the EPFL arithmetic benchmark suite
- ▶ Future work: depth minimization in tech mapping

