Logic Synthesis for Energy-Efficient Photonic Integrated Circuits

Zheng Zhao, Zheng Wang, Zhoufeng Ying,
Shounak Dhar, Ray T. Chen, and **David Z. Pan**

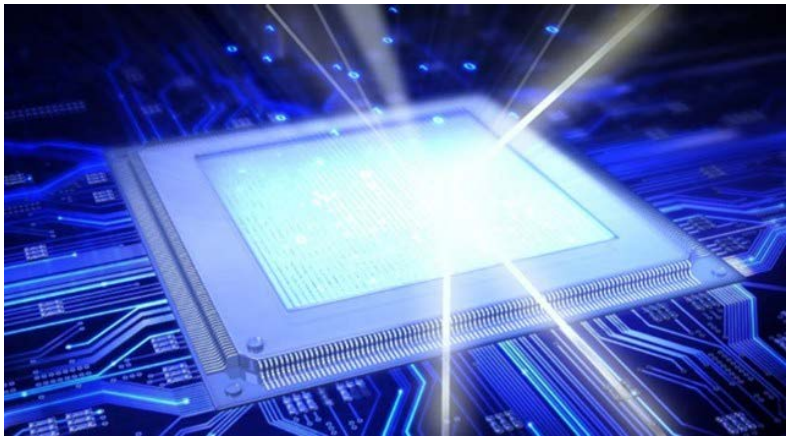
Dept. of Electrical and Computer Engineering
The University of Texas at Austin

Outline

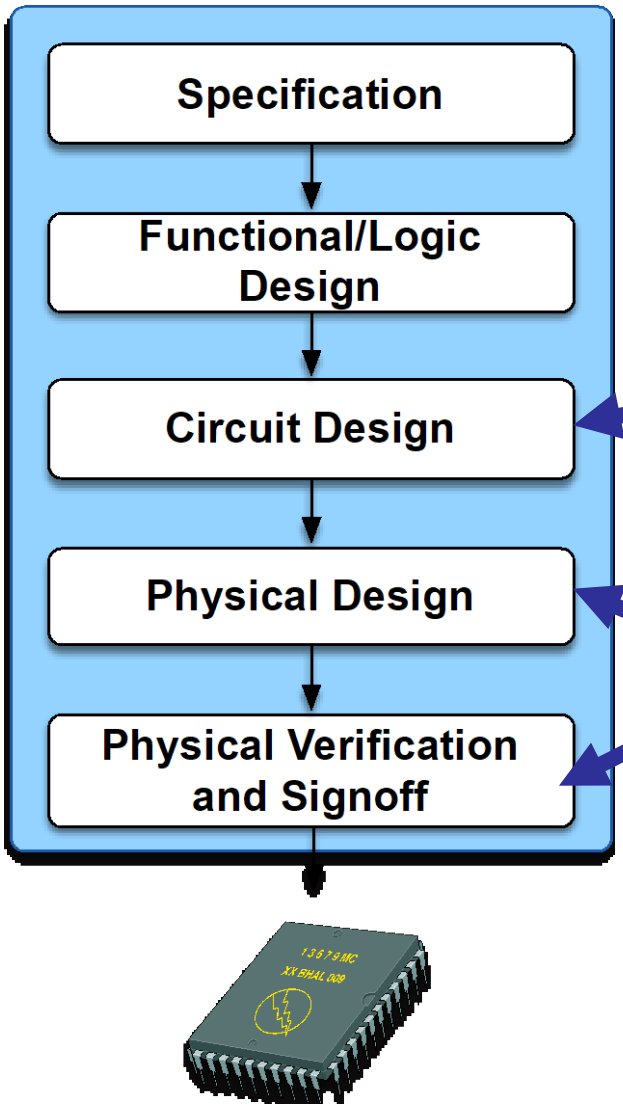
- Introduction and background
- Logic synthesis algorithms
- Experimental results
- Conclusion

What is Optical Computing?

- Use optics/photonics to perform computation
- Information is an optical signal sourced by lasers and detected by photodetectors
- Great potentials
  - Significant reduction of signal transfer latency
  - Ultra-low energy consumption
  - Simplified architecture for many computation tasks



Automate the Chip Design Flow



Logic Synthesis → ?

cadence

SYNOPSYS® Optical

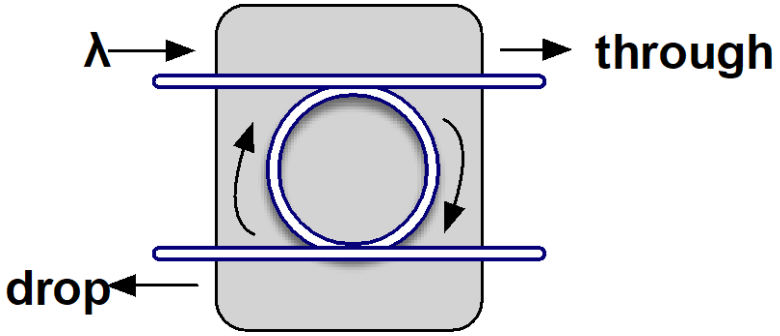
PhoeniX Software
Solutions for Micro and Nano Technologies

LUCEDA PHOTONICS

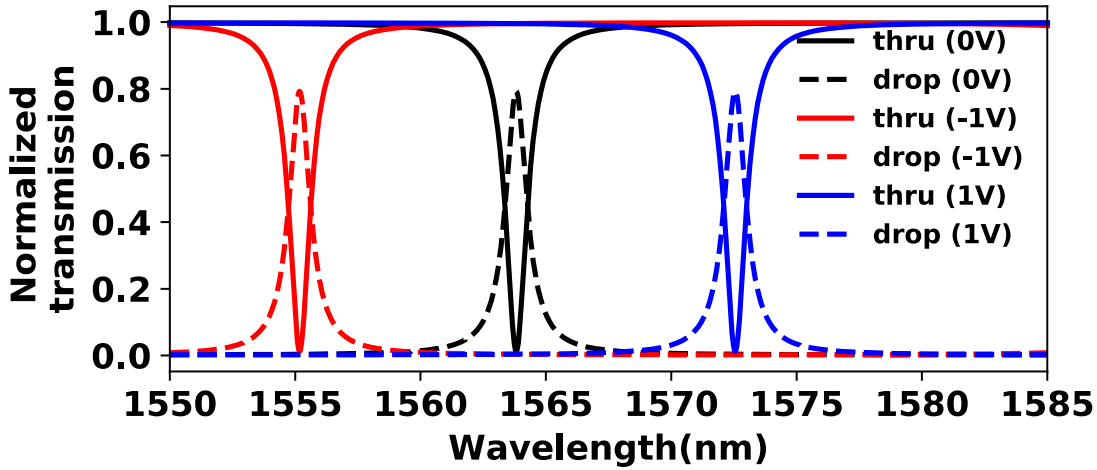
Mentor Graphics®

Optical Computing Components

- Microresonator-based Optical Switches
  - Can be implemented with microrings/microdisks

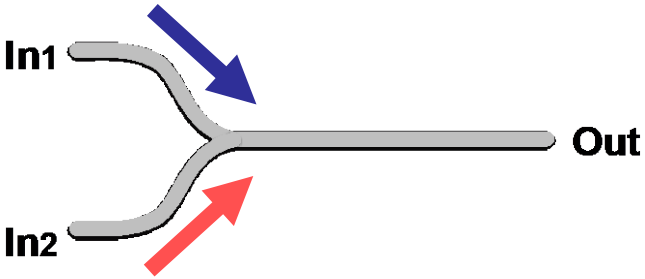


2X2 optical switch

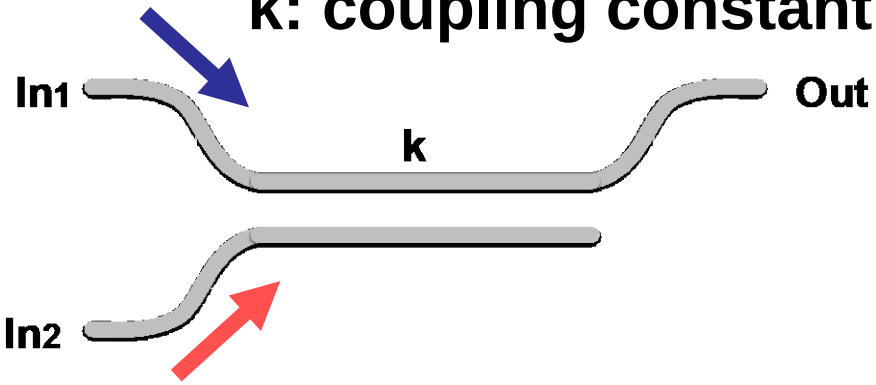


Optical Computing Components

- Y-branch combiner and directional coupler
  - When there is **only one** light input



k: coupling constant



$P_{out} = 0.5\ P_{In1}$
$P_{out} = 0.5\ P_{In2}$

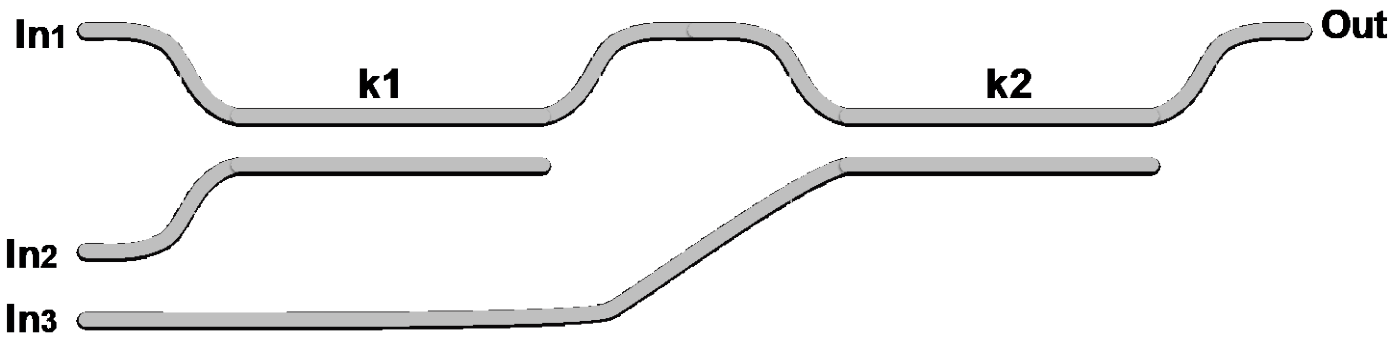
$P_{out} = k\ P_{In1}$
$P_{out} = (1-k)\ P_{In2}$

Power efficiency factor = P_{out}/P_{in}

- Size of a typical coupler ≈ 2X size of a typical micro-resonator-based switch

Optical Computing Components

- An NX1 combiner/coupler can be implemented by connecting an array of 2X1 combiners/couplers
- NX1 coupler of **arbitrary power efficiency factors** is achievable by cascading (N-1) 2X1 couplers



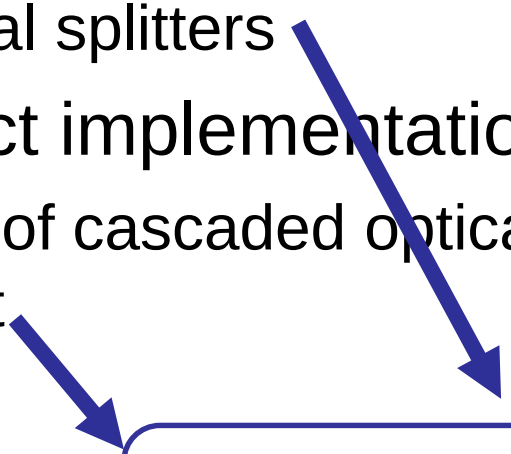
3X1 directional coupler

Outline

- Introduction and background
- **Logic synthesis algorithms**
- Experimental results
- Conclusion

Previous Works and Problems

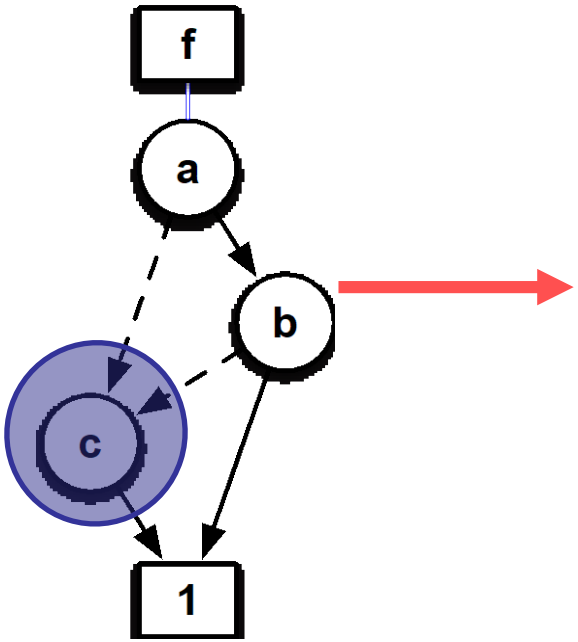
- Synthesis using virtual gates [Condrat+, GLSVLSI'2011]
  - A large number of optical components and
  - Cascaded optical splitters
- BDD-based direct implementation [Wille+, ASPDAC'2015]
  - A large number of cascaded optical combiners with single light input



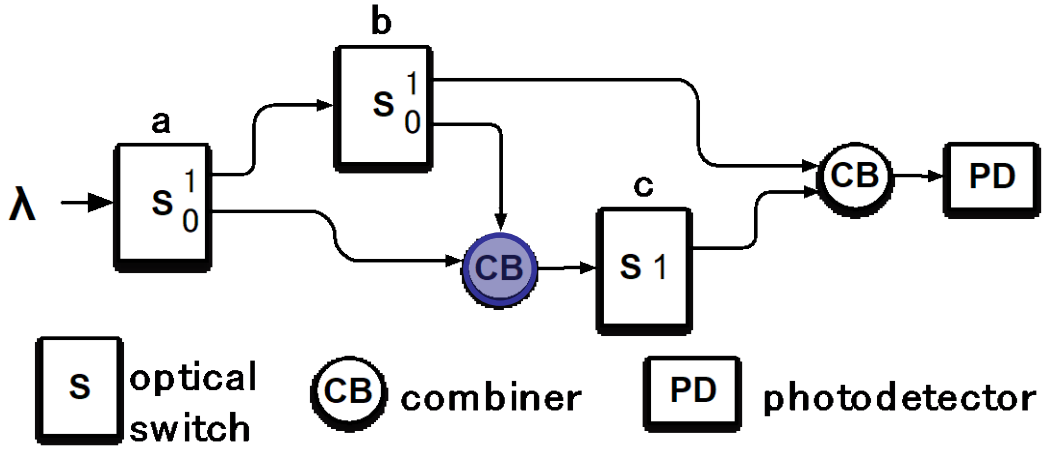
Each has 3dB loss cascaded quickly!

Previous Works and Problems

- Data structure and the direct implementation



Binary decision diagram (**BDD**)

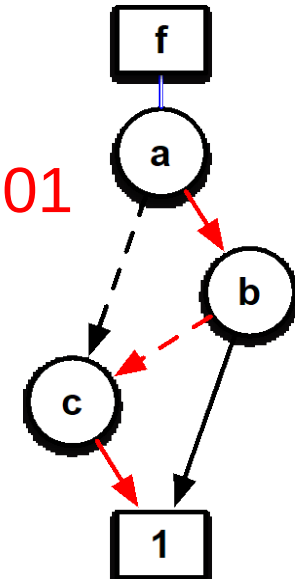


Optical direct implementation: each multi-parent BDD node has a combiner

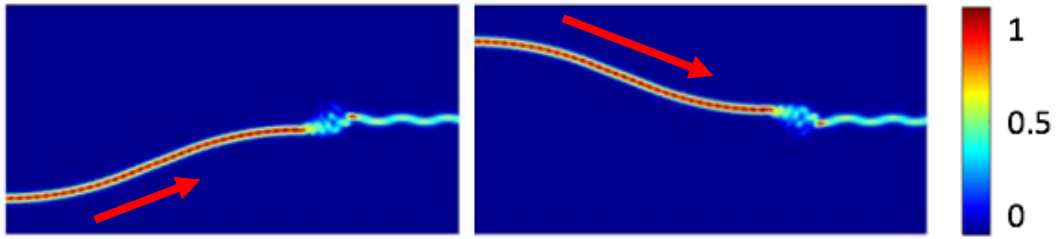
Problem with Direct Implementation

- Due to *BDD's* **single-path property,** any combiners have **at most one** light input
- Power is cut by half (3dB)

abc = 101



Binary decision diagram (**BDD**)



Light stream to the lower/upper input port

Problem with Direct Implementation

- **Power efficiency** is a big issue
  - Optical power depletes fast due to device loss
  - Optical power become too small to be detected
  - Requires more amplifiers which leads to greater overhead

Power Efficiency Factor

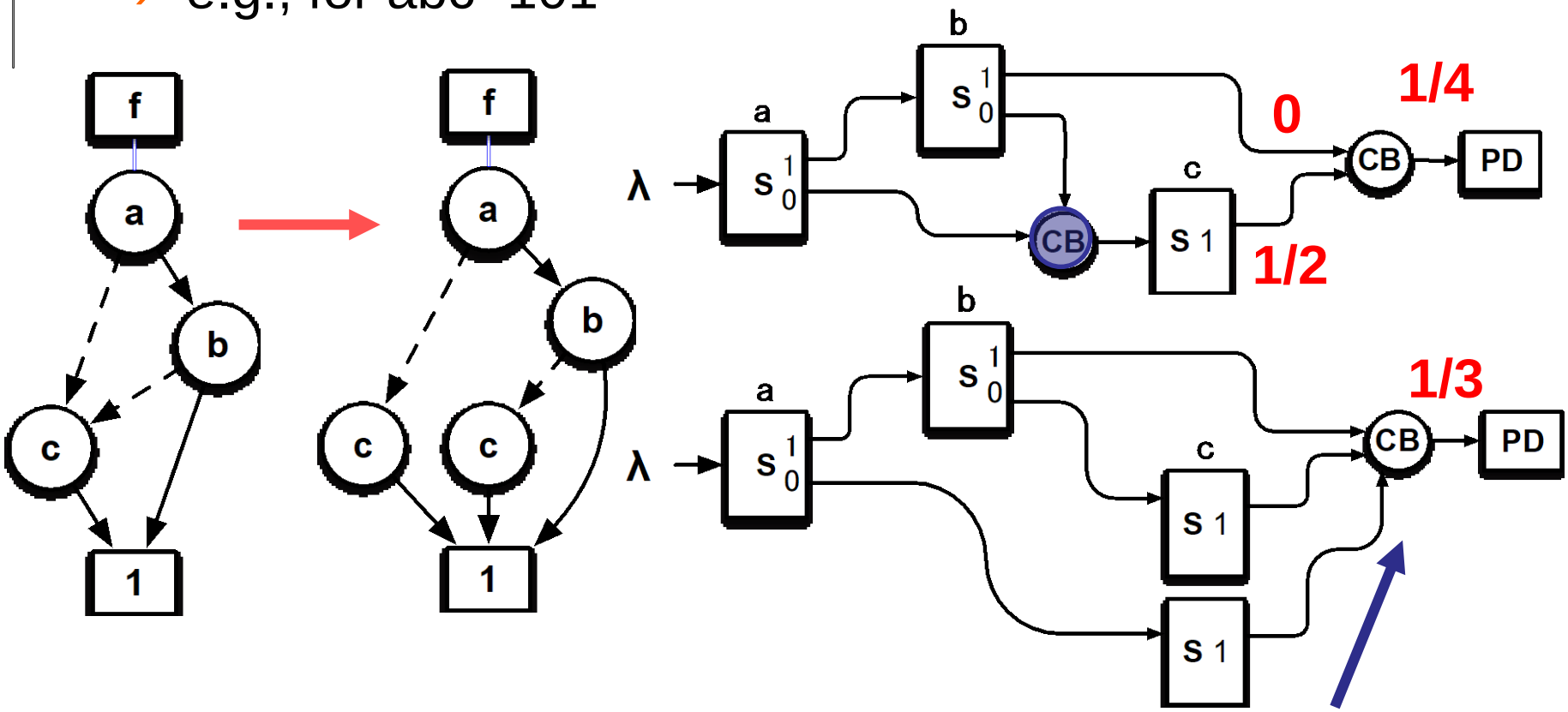
- For an general optical network, **power efficiency factor** $\gamma = P_{out}/P_{in}$
- In a BDD-based optical network, γ can be defined for
  - **node**, as an optical switch
  - **edge**, as an input branch of a combiner/coupler
  - **path** and the **whole network**

Proposed Algorithms

- **Our goal**: to improve the worst-case network efficiency under a reasonable overhead and computational budget
- Two techniques
  - **Combiner elimination** to avoid cascaded combiner loss
  - **Coupler assignment** to redistribute the power resource

Technique 1: Combiner Elimination

- Idea: avoid **cascaded** combiner loss
  - e.g., for abc=101



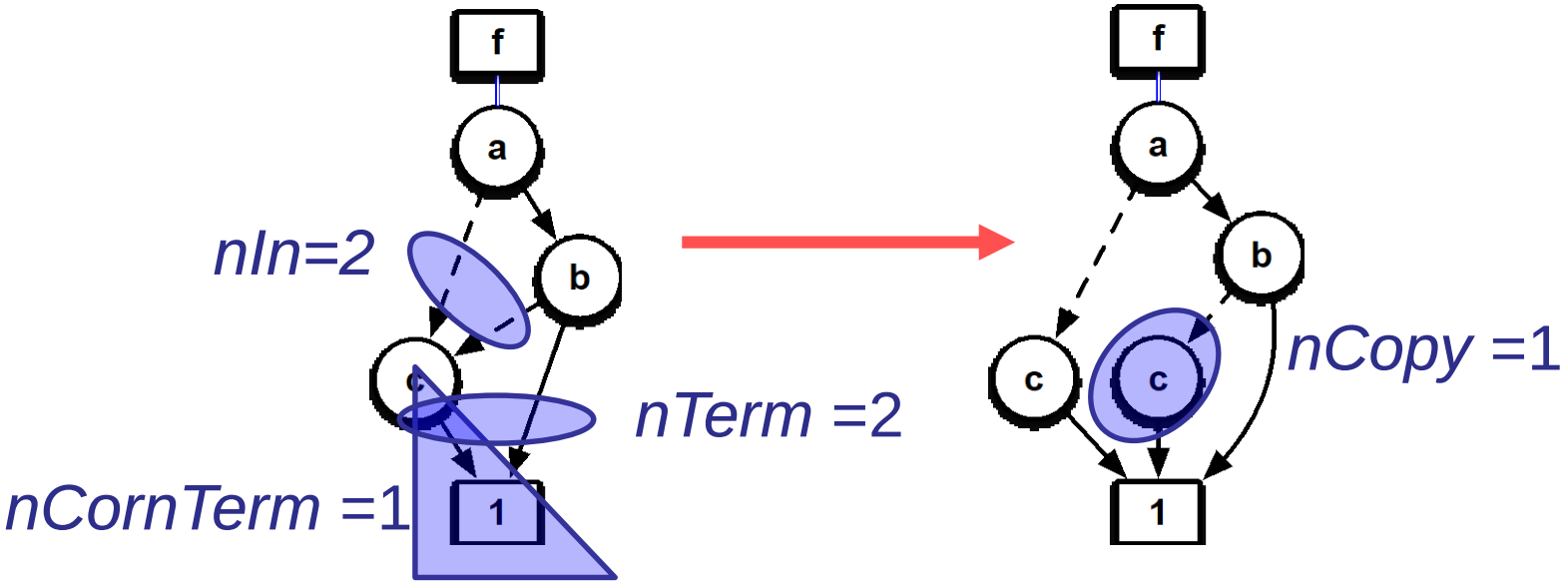
Greater combiner loss at the terminal but not cascaded

Quantify the Benefit

- After eliminating a combiner at an internal node

$$\gamma_{new}/\gamma_{org} = \frac{nIn}{1 + nCopy \cdot nCornTerm/nTerm}$$

- Example: eliminating c's combiner



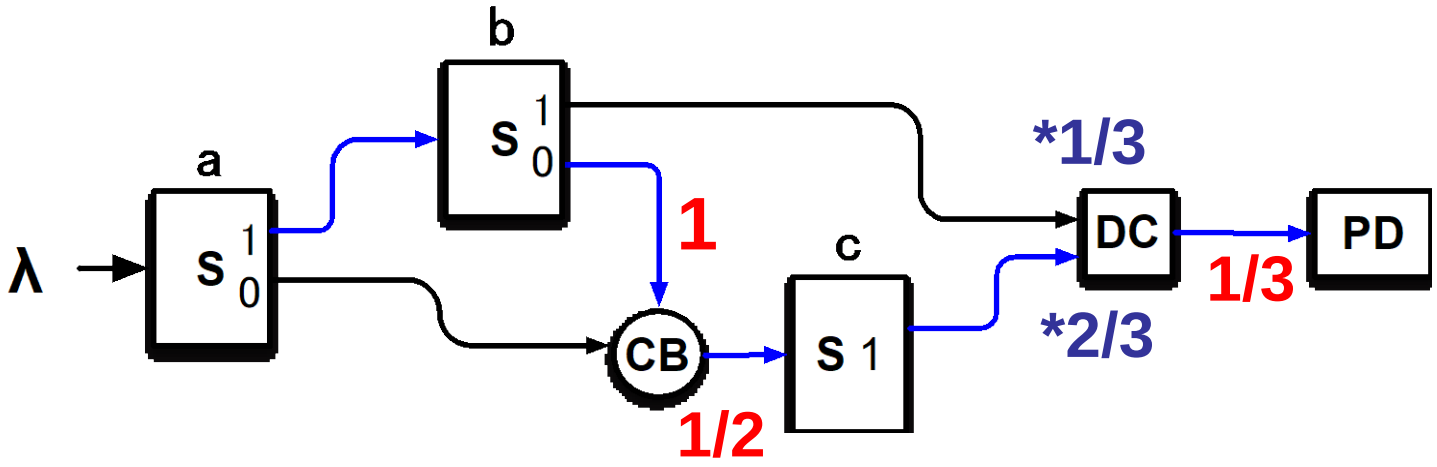
$$\gamma_{new}/\gamma_{org} = \frac{2}{1 + 1 \cdot \frac{1}{2}} = 4/3$$

Technique 1: Combiner Elimination

- How to select the node
  - **Benefit ratio**: $\gamma_{new}/\gamma_{org} > 1$
  - **Overhead**: the duplicated node number is controlled
- Heuristic: for paths with the lowest power first
  - Compute the **ratio** and **overhead** for nodes closer to the terminal first (generally have smaller overhead)
  - If both meet the set criteria, copy the node cone
  - Stop until the overhead budget is reached

Technique 2: Coupler Assignment

- Idea: redistribute the power with directional couplers (DCs) instead of combiners
- Assign the coupling efficiency for each DC



General Coupler Assignment Formulation

- Polynomial programming formulation
  - On path $\boldsymbol{p_i}$, $\boldsymbol{x_j}$ is an assignment of coupling efficiency for the edge $\boldsymbol{e_j}$
- Objective function

Power efficiency for path p_i

$$Maximize \min_i \{\gamma_i \cdot \prod_{j:e_j \in p_i} x_j\}$$

Other source of power loss on path p_i

- Transform the max-min objective to **path constraints** with dummy variable f

$$f \leq \gamma_i \cdot \prod x_j, \forall i$$

General Coupler Assignment Formulation

- Other constraints
  - **Node constraint:** rule of power conservation

$$\text{s.t.} \sum_{i \in In(n)} x_i = 1, \ \forall n$$

  - Power efficiency for each coupler

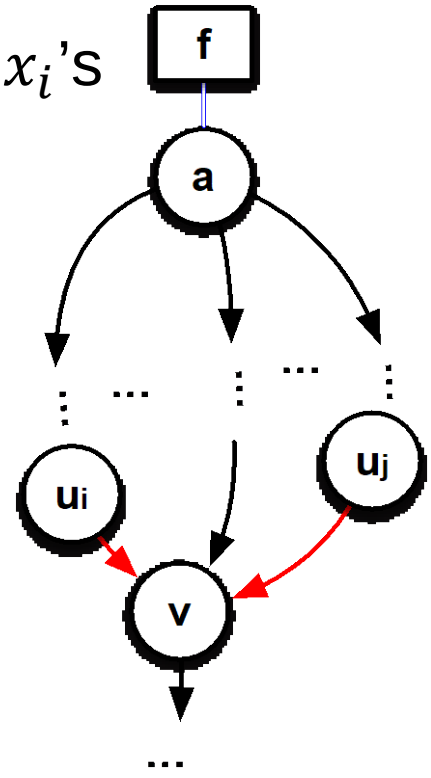
$$0 \leq x_i \leq 1, \ \forall i$$

- Solvable by semidefinite programming (SDP) relaxation, but very time consumptive

Fast Solution

- **Iteratively** solve by quadratically constrained programming (QCP)
  - In each iteration, optimize **a small set** of x_i's
- For each critical path,
  - Evaluate the **divergence factor** for each multi-input node v

$$div(v) = \max_{\substack{(u_i,v),(u_j,v)\in E,\\ u_i \neq u_j}} \frac{\gamma_{top \to u_i}}{\gamma_{top \to u_j}}$$



  - For each selected node, reassign the two input edges contribute to the div
  - Select two nodes with the highest div for QCP

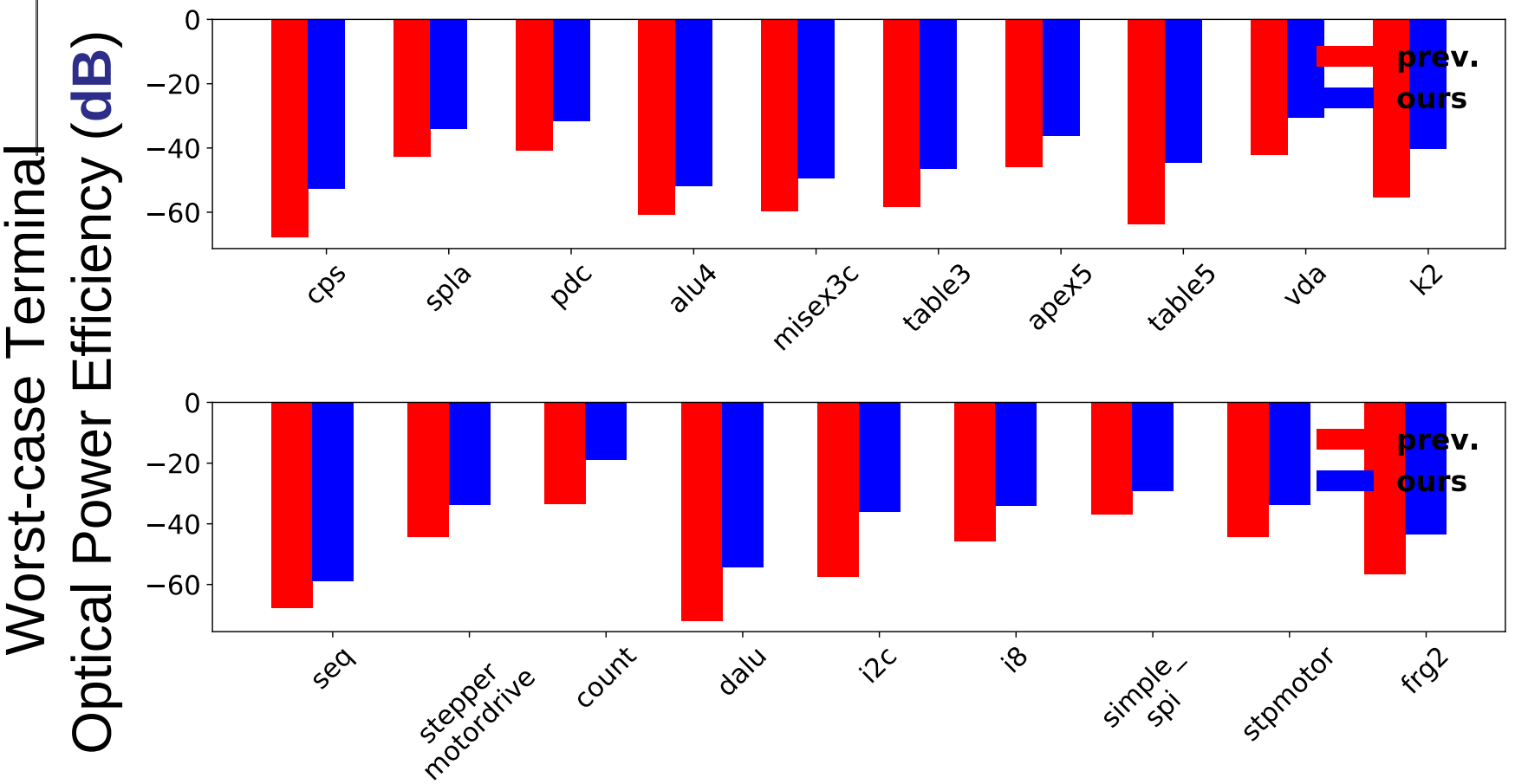
Outline

- Introduction and background
- Logic synthesis algorithms
- **Experimental results**
- Conclusion

Experimental Setup

- Two techniques performed iteratively for each benchmark and stop if no further improvement
- Implementation in C++ with CUDD package
- Linux machine with 8 3.4GHz CPUs
- QCP solver: Gurobi QCP
- Benchmarks
  - Microelectronics Center of North Carolina (MCNC)
  - International Workshop on Logic and Synthesis (IWLS) benchmarks
- BDD-reordering heuristic
  - CUDD_REORDER_SYMM_SIFT

Optical Power Efficiency



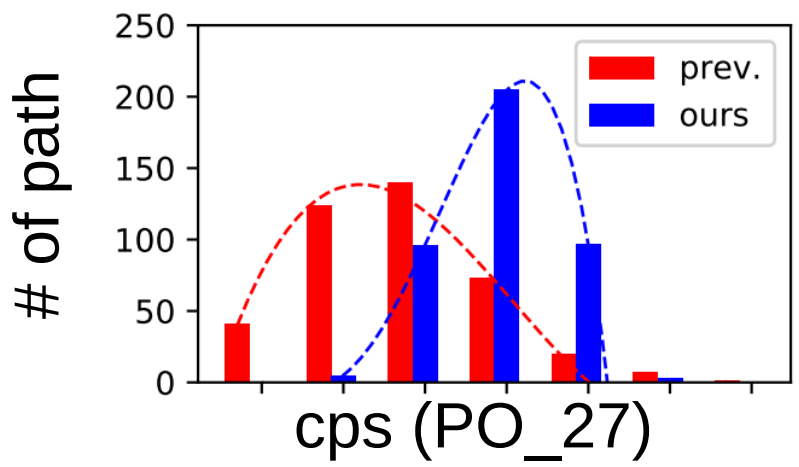
Prev. work: [Wille+,ASPDAC'2015]

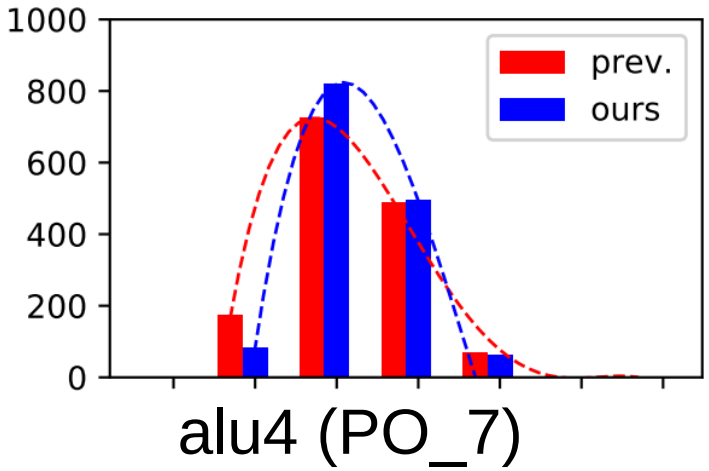
Average power efficiency ratio over prev.: **27.02X**

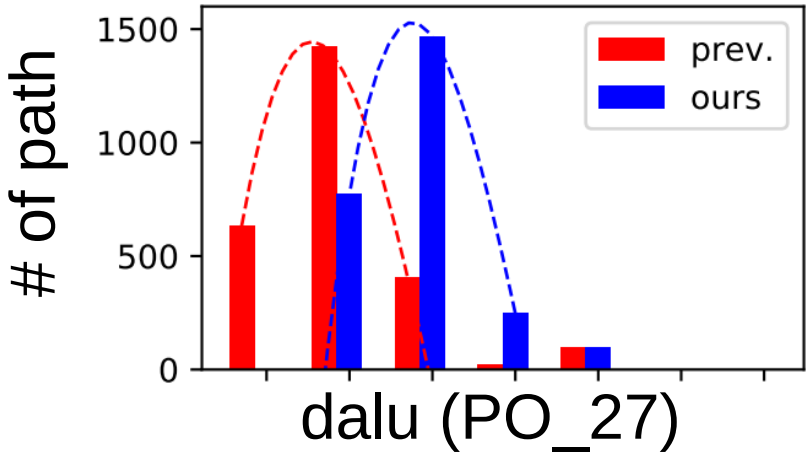
Average/greatest CPU time: **1.88s / 14.5s**

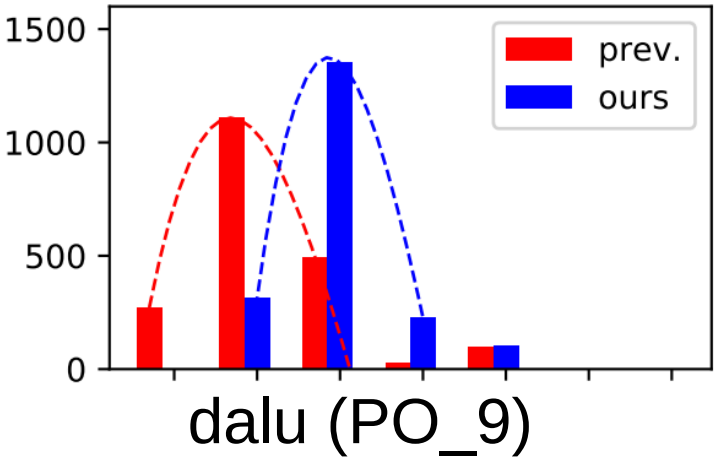
Loss Distribution

- Distribution moves from low efficiency zone to high









Power efficiency zone {0~10^{-6}, 10^{-6}~10^{-5}, … , 10^{-1}~1}

Outline

- Introduction and background
- Logic synthesis algorithms
- Experimental results
- **Conclusion**

Conclusion

- We study the **optical power depletion**, a critical issue of integrated optical circuits
- We address the problem with two techniques, **combiner elimination** and **coupler assignment**
- which also helps to build a much more **noise-resilient** and **scalable** integrated photonic system

Thanks!

Q & A ?