JOINT INSTITUTE
交大密西根学院

Approximate Logic Synthesis for Area and Delay Optimization

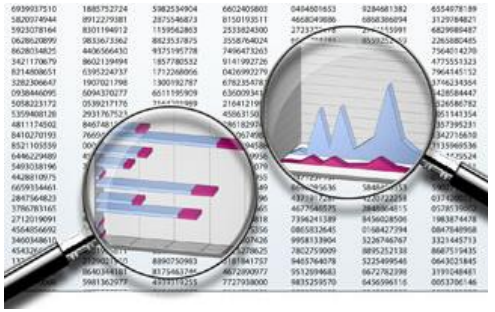
Weikang Qian, Ph.D.
Associate Professor
University of Michigan-SJTU Joint Institute
Shanghai Jiao Tong University

DATE'19 Friday Workshop: Quo Vadis, Logic Synthesis?
Florence, Italy, Mar. 29, 2019

Outline

- Background on Approximate Computing
- Area-driven Approximate Logic Synthesis
- Delay-driven Approximate Logic Synthesis
- Conclusion

Approximate Computing: Motivation



Low-power VLSI

Error-tolerant applications

Exact Results NOT Necessary

- A few erroneous pixels do not affect human recognizing the image.

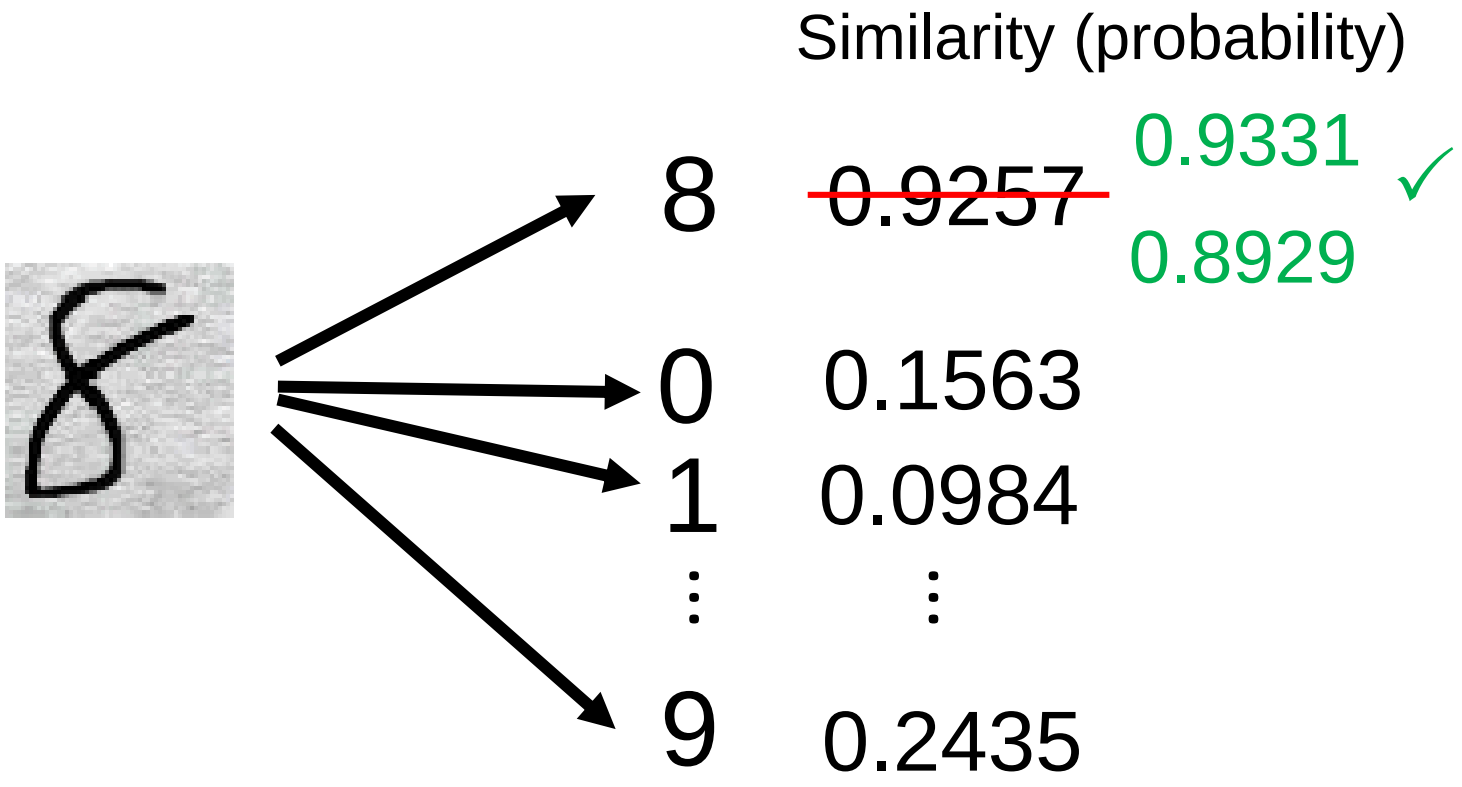
Accurate Result

Inexact Result

[Han and Orshansky, 2013]

Numerical Values NOT Matter

- Handwritten Digit Recognition



NO Golden Standard Answer

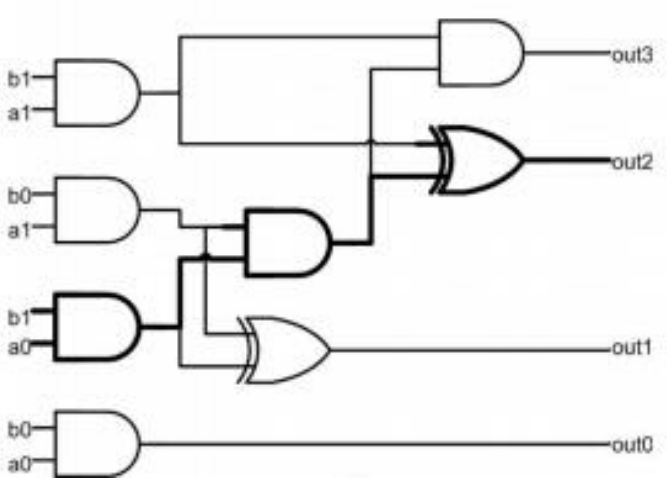
Approximate Computing

- Design a circuit that may not be 100% correct
  - Targeting at error-tolerant applications
  - Trade accuracy for area/delay/power

[Kulkarni et al. 2011]

A1A0 \ B1B0	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	1001	110
10	000	010	110	100

K-Map for 2-bit multiplier



Digital circuit

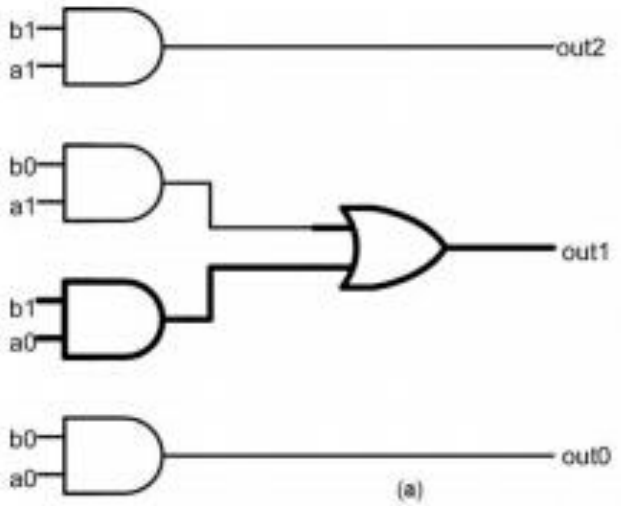
Approximate Computing

- Design a circuit that may not be 100% correct
  - Targeting at error-tolerant applications
  - Trade accuracy for area/delay/power

[Kulkarni et al. 2011]

A1A0 \ B1B0	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	111	110
10	000	010	110	100

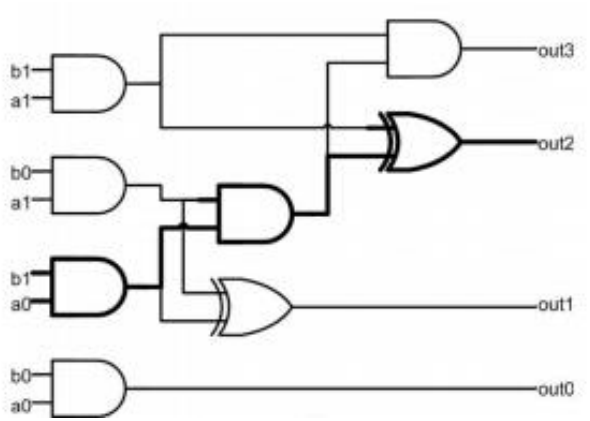
K-Map for 2-bit multiplier



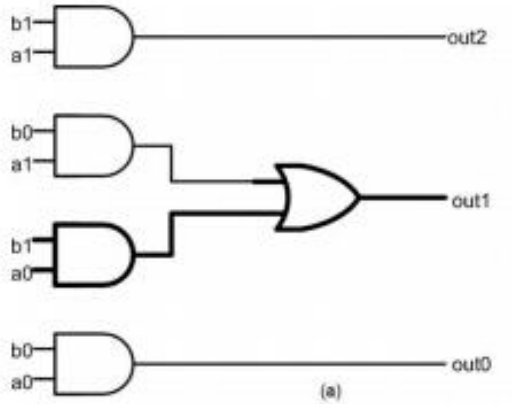
Digital circuit

Approximate Computing

- Design a circuit that may not be 100% correct
  - Targeting at error-tolerant applications
  - Trade accuracy for area/delay/power



Accurate
2-bit multiplier



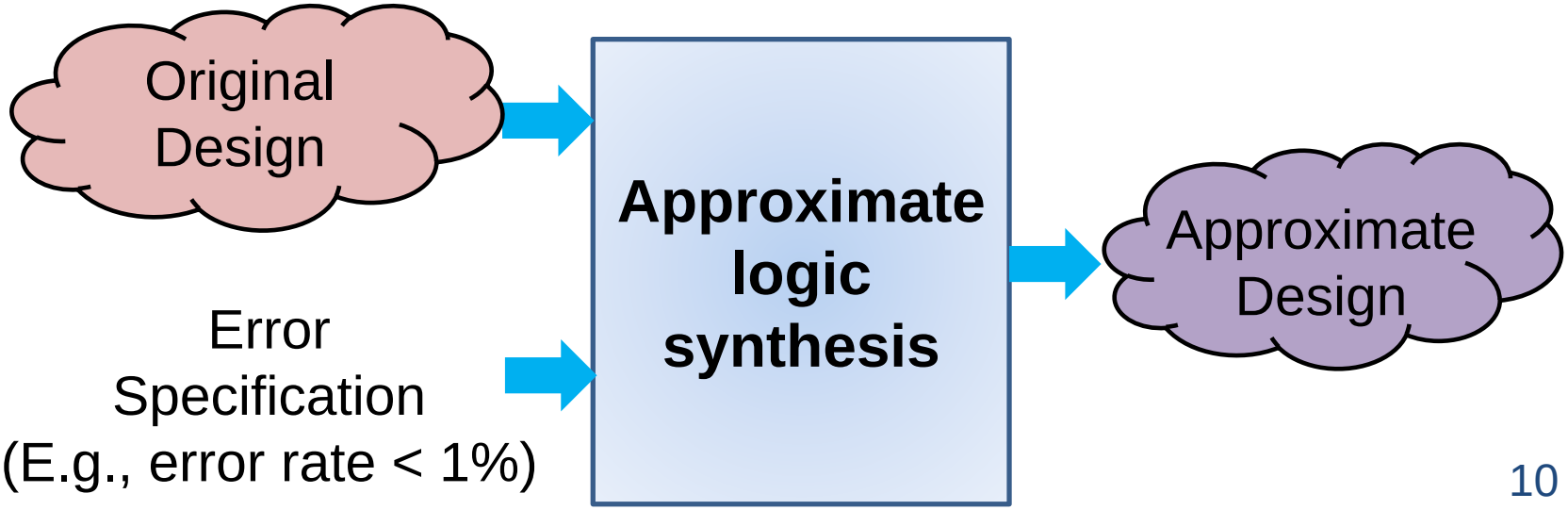
Approximate
2-bit multiplier

[Kulkarni et al., 2011]

Approximate Logic Synthesis

A1A0 \ B1B0	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	1001	110
10	000	010	110	100

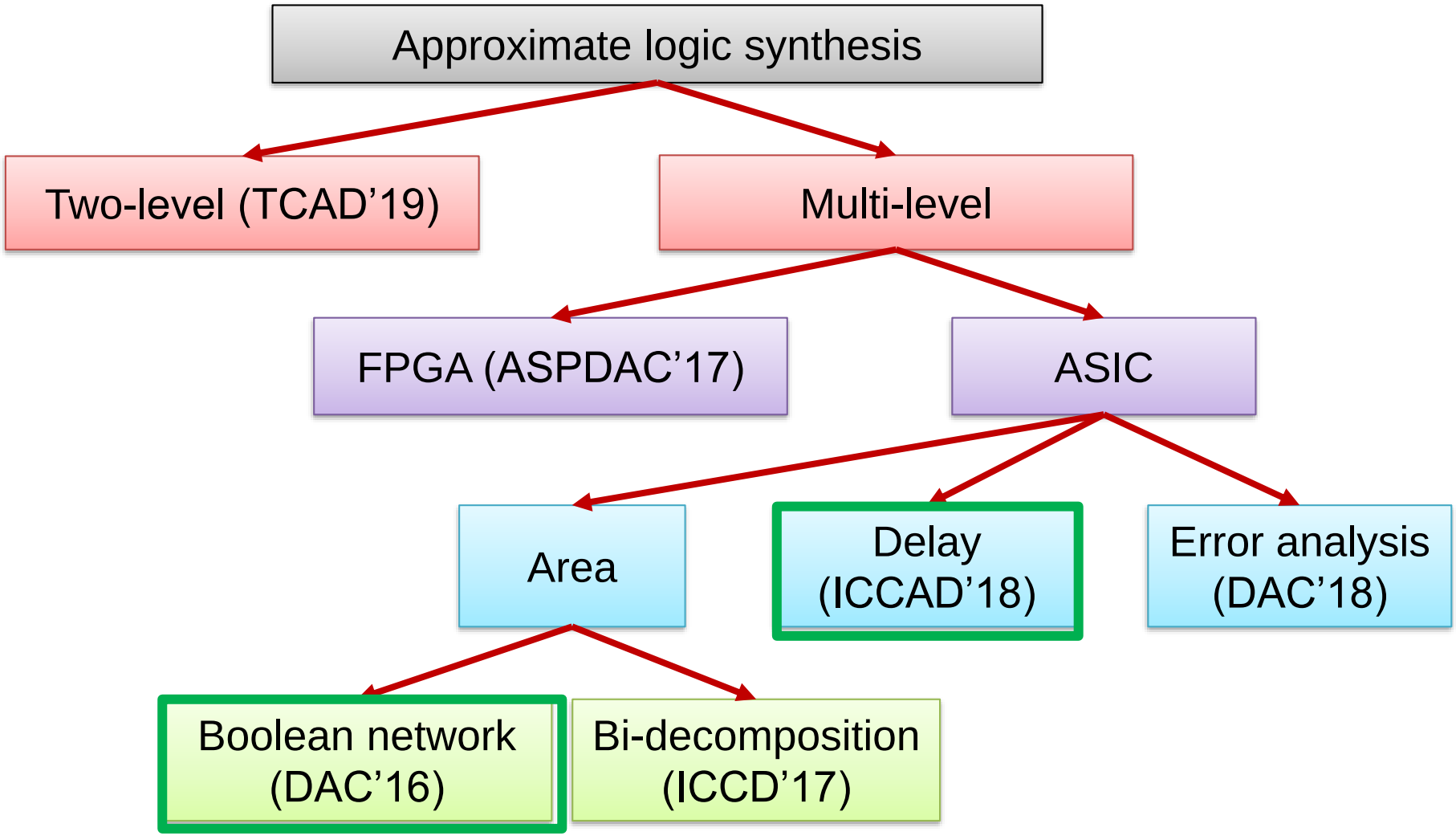
What's the optimal way to introduce error?



Existing Works on ALS

- Inject stuck-at-faults [Shin and Gupta, DATE '11]
- SALSA: Encode the error constraint as a function and use exploit don't cares [Venkataramani et al., DAC'12]
- SASIMI: Identify similar signal pairs and substitute one signal by the other in the pair [Venkataramani et al., DATE'13]
- Multi-level logic synthesis under general error constraint [Miao et al., ICCAD'14]
- Approximate AIG Rewriting [Chandrasekharan et al., ICCAD'16]
- SCALS: Statistically certified approach with stochastic optimization [Liu and Zhang, ICCAD'17]
- BLASYS: Boolean matrix factorization [Hashemi, Tann, and Reda, DAC'18]
- Approximate logic synthesis by symmetrization [Bernasconi, Ciriani, and Villa, DATE'19]
- …

Our Work on Approximate Logic Synthesis

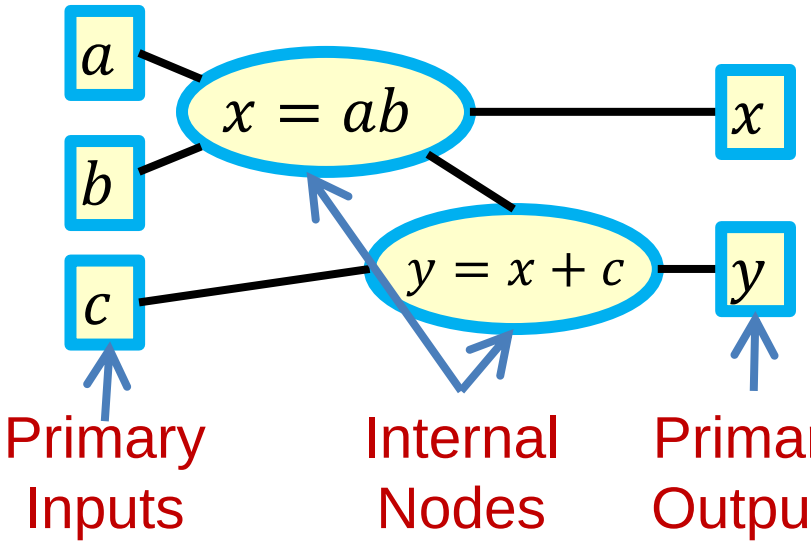


Outline

- Background on Approximate Computing
- Area-driven Approximate Logic Synthesis
  - Y. Wu and W. Qian, *Design Automation Conference (DAC)*, 2016
- Delay-driven Approximate Logic Synthesis
  - Z. Zhou, Y. Yao, S. Huang, S. Su, C. Meng, and W. Qian, *International Conference on Computer-Aided Design (ICCAD)*, 2018
- Conclusion

Background: Boolean Logic Network Model

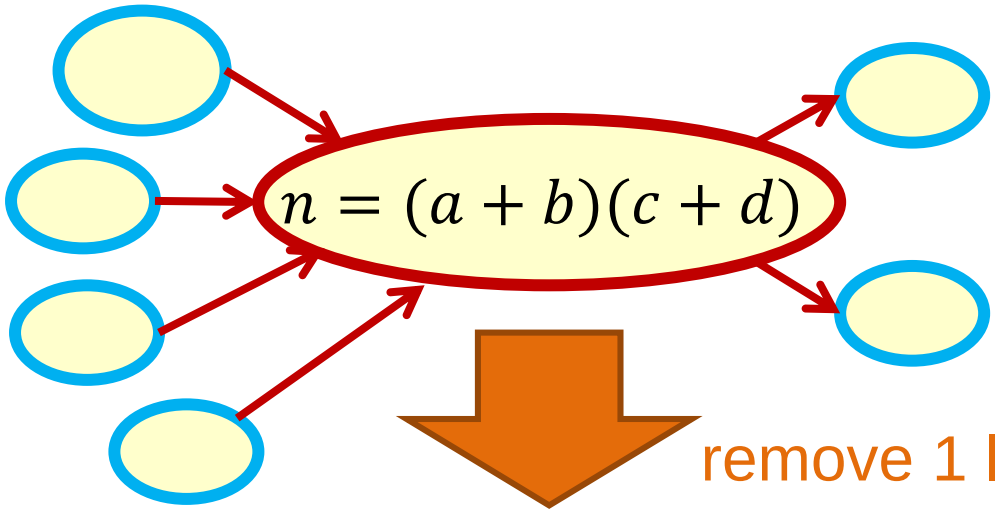
- A direct acyclic graph. Each node is a Boolean function
  - Boolean function could be in either sum-of-product (SOP) form or factored form
- Quality metric: literal count



Literal count = 4

Approximating by Local Change

- Work on factored-form expression of a node
  - Simplify it by removing some literals
  - An approximation; can cause error
  - Call the result approximate simplified expression (ASE)



Each node has multiple ASEs

remove 1 literal

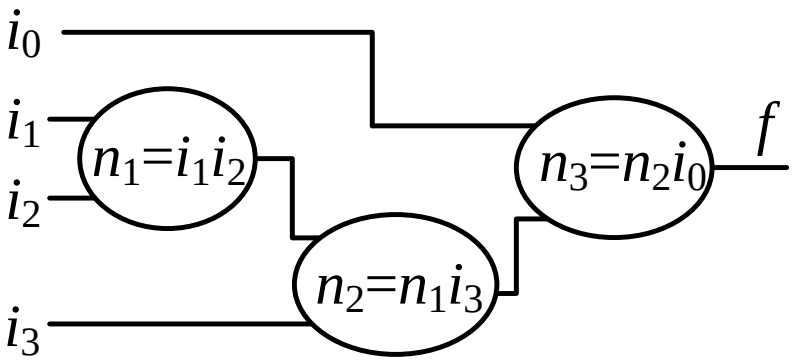
$n = b(c + d)$ $n = a(c + d)$ $n = (a + b)d$ $n = (a + b)c$

Single-Selection Algorithm

- Each round select one node for shrinking
  - Check all nodes; for each node, check all ASEs
  - Pick a node n and an associated ASE of n with the largest score
  - Score = #saved_literal / error_rate

- However, it is slow ...
  - Accelerate it by selecting multiple nodes for simultaneous change in each round

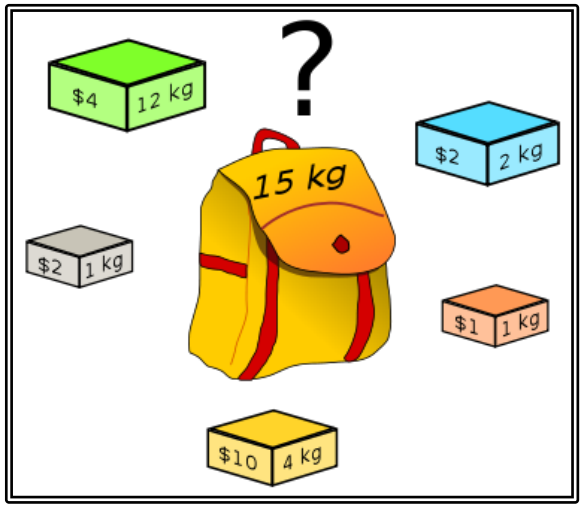
Optimal Choice of Multiple Changes



- Questions:
  1. Which set of nodes should we choose to make change?
  2. For these chosen nodes, which of their ASEs should we pick?
- Proposed solution: model this as a 0/1 multi-state knapsack problem

0/1 Multi-state Knapsack Problem

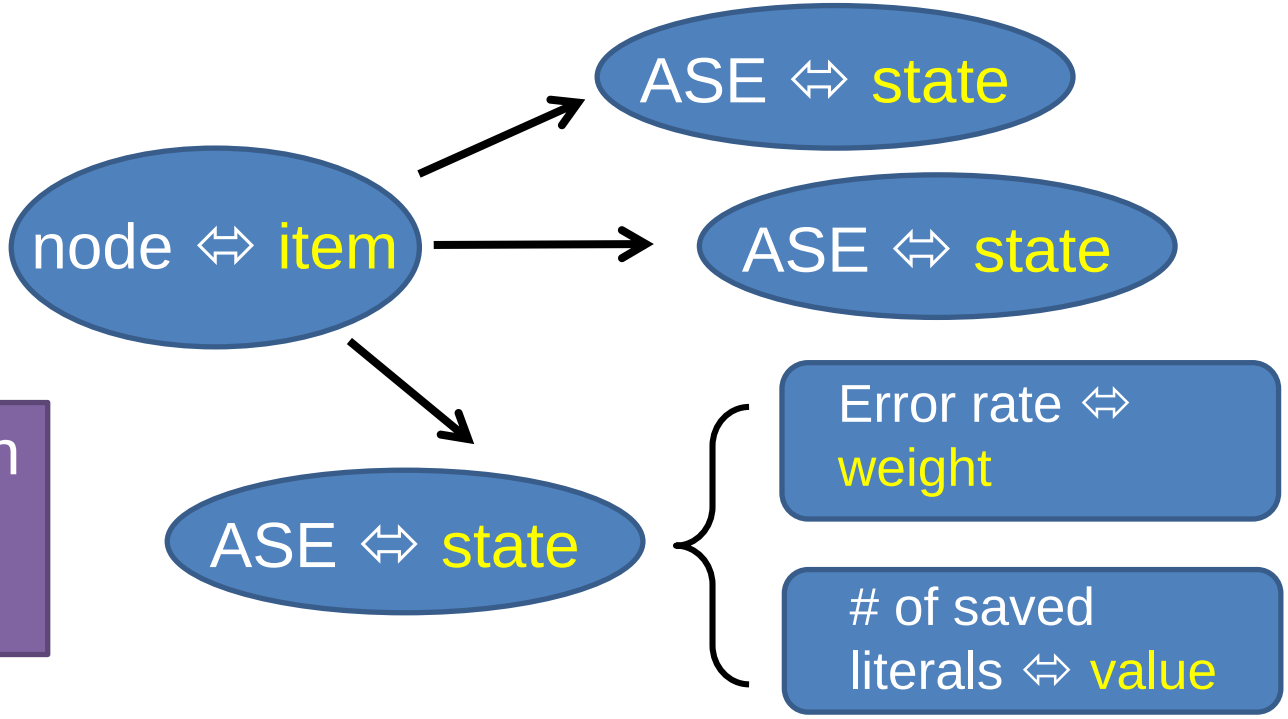
- n items. Each item c_i has m_i states $(w_{i1}, v_{i1}), (w_{i2}, v_{i2}) \ldots (w_{im_i}, v_{im_i})$
  - w_{ij} &v_{ij}: weight and value of the state j of item i
- Choose a set of items and their associated states to put into a knapsack with capacity W to maximize the total value



capacity = 9

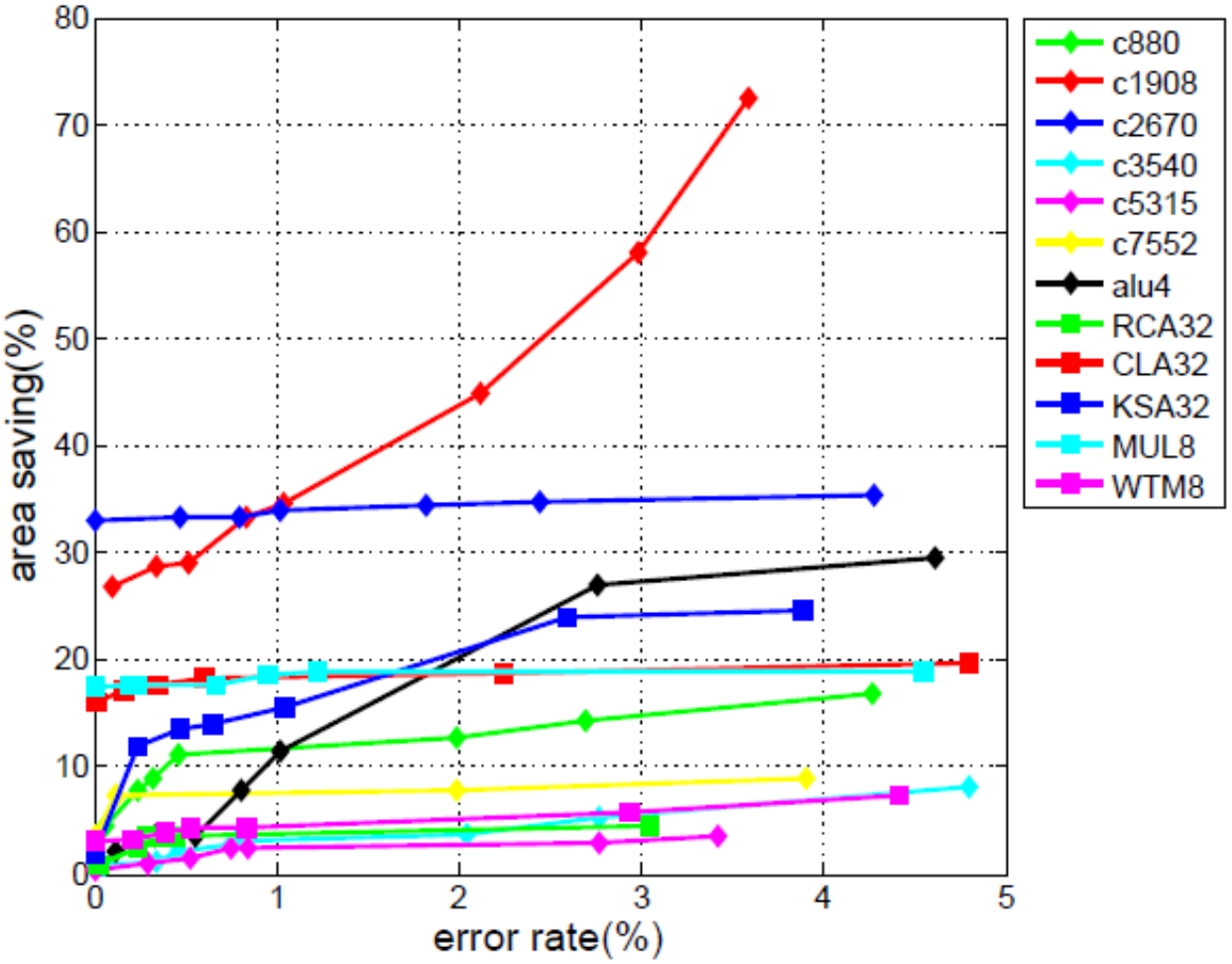
item	state	weight	value
c_1	s_{11}	2	1
	s_{12}	3	2
c_2	s_{21}	4	2
	s_{22}	6	4
c_3	s_{31}	2	1

Mapping to Multi-state Knapsack Problem



Can be solved by extending the classical dynamic programming solution to basic 0/1 knapsack problem

Experimental Results: Area Saving



Comparison to Previous Method

- SASIMI [Venkataramani+, DATE'13] is a state-of-the-art method
- Average over 7 error rate thresholds: 0.1%, 0.3%, 0.5%, 0.8%, 1%, 3%, 5%

	SASIMI		multi-selection	
circuit	area ratio	time/s	area ratio	time/s
c880	0.896	154	0.893	48
c1908	0.610	1090	0.598	181
c2670	0.724	664	0.673	90
c3540	0.975	393	0.965	77
c5315	0.981	996	0.981	85
c7552	0.948	2665	0.941	173
alu4	0.892	645	0.869	186
RCA32	0.972	33	0.969	15
CLA32	0.829	196	0.822	57
KSA32	0.830	553	0.831	39
MUL8	0.829	1095	0.826	151
WTM8	0.959	249	0.956	57
Geomean	0.863	452	0.852	77

Area saving:
Slightly better

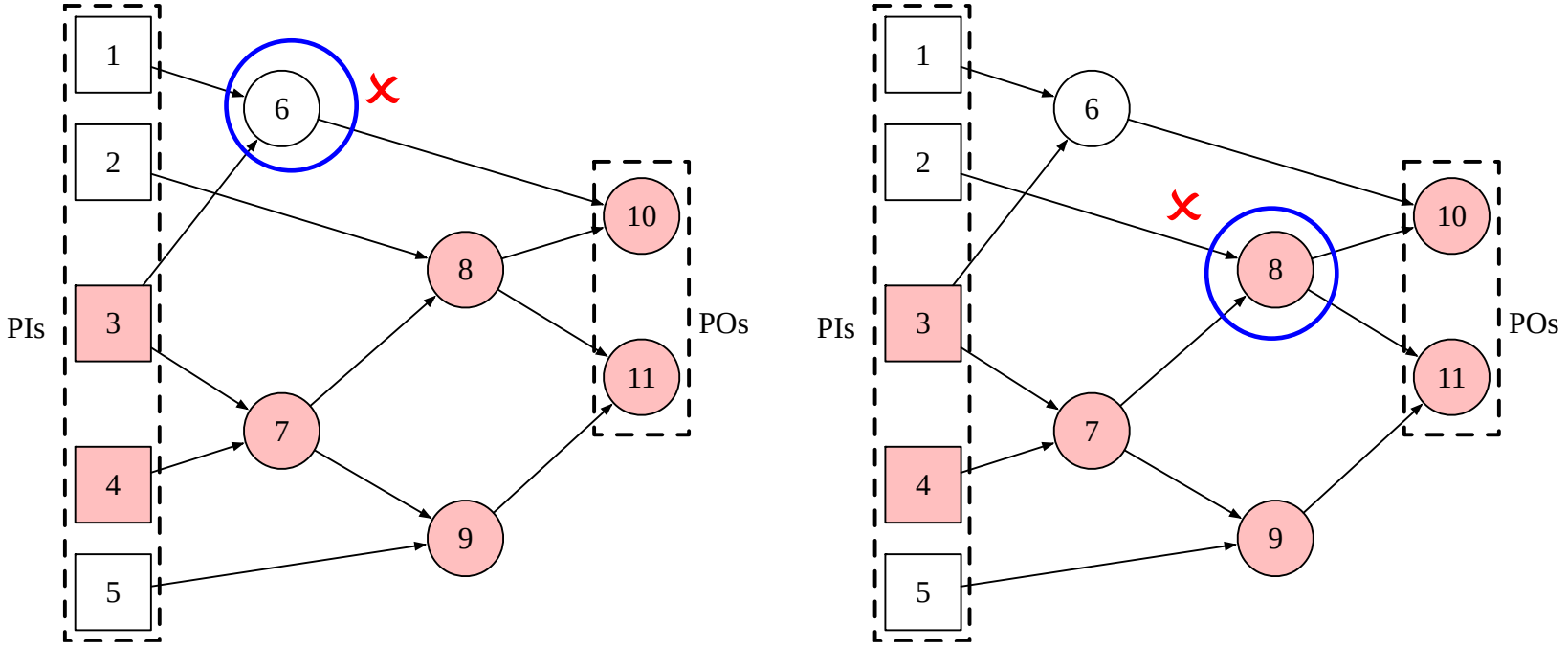
Acceleration:
5.9X

Outline

- Background on Approximate Computing
- Area-driven Approximate Logic Synthesis
- Delay-driven Approximate Logic Synthesis
- Conclusion

Area-Driven ALS Not Good in Reducing Delay

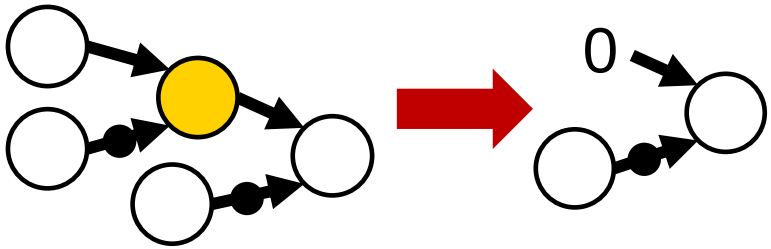
- Applying approximate local changes (ALCs) on non-critical gates
- Applying ALC on a single critical gate is not effective, since there exist multiple critical paths



DALS: Delay-driven ALS

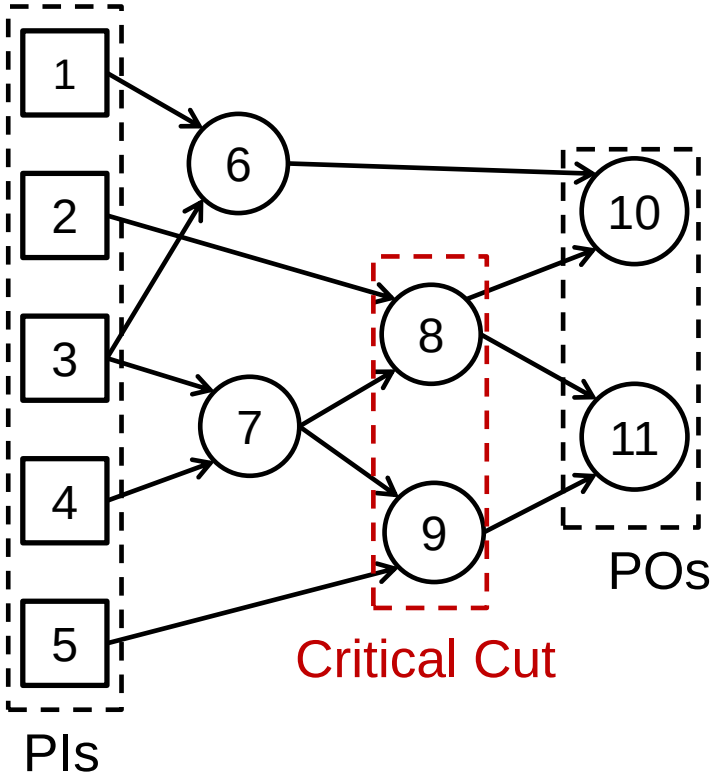
- Basic idea
  - Work on AND-inverter graph (AIG)
  - Apply depth-reduction approximate local changes

Constant Replacement

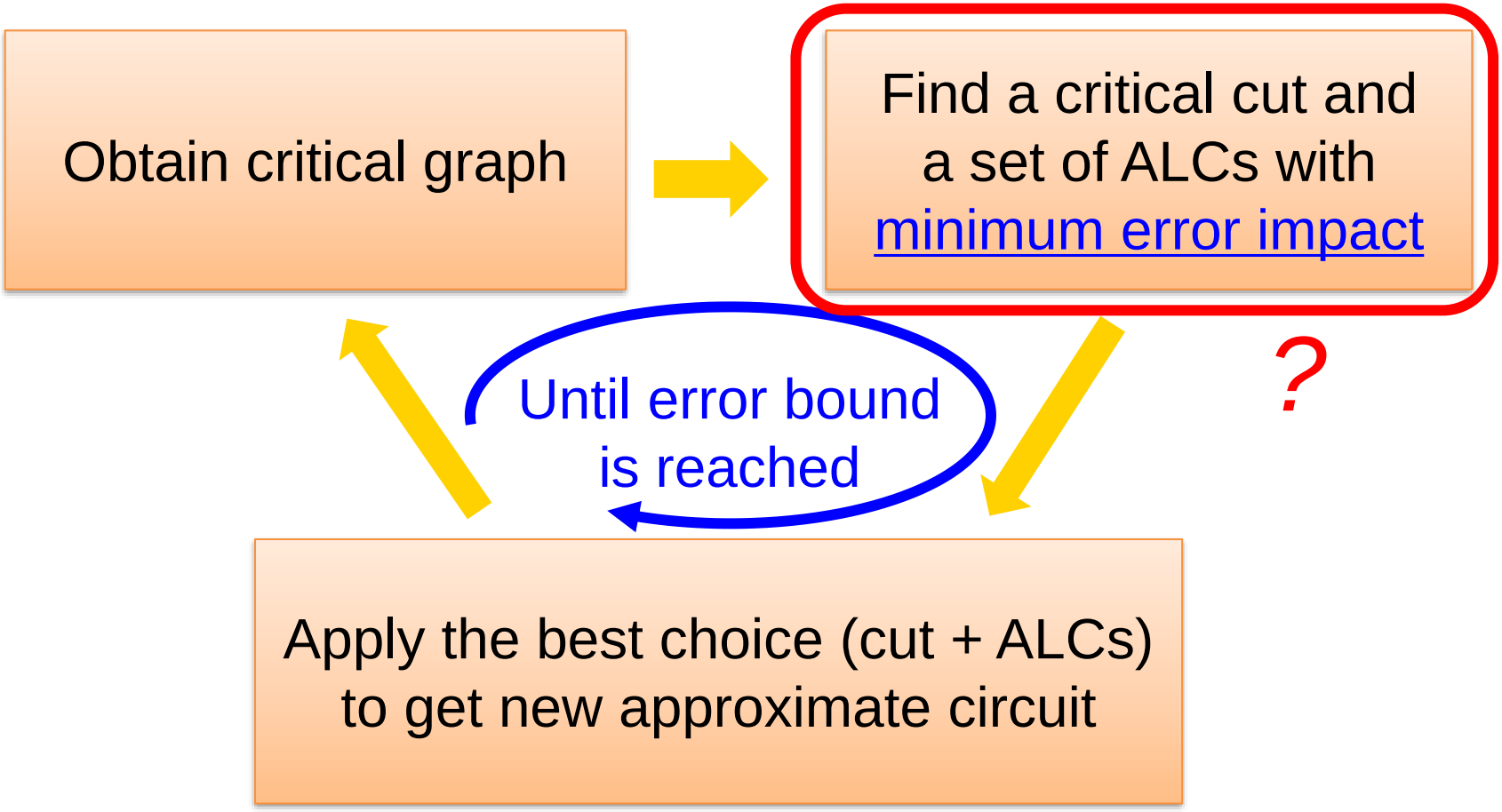


Global Depth Reduction

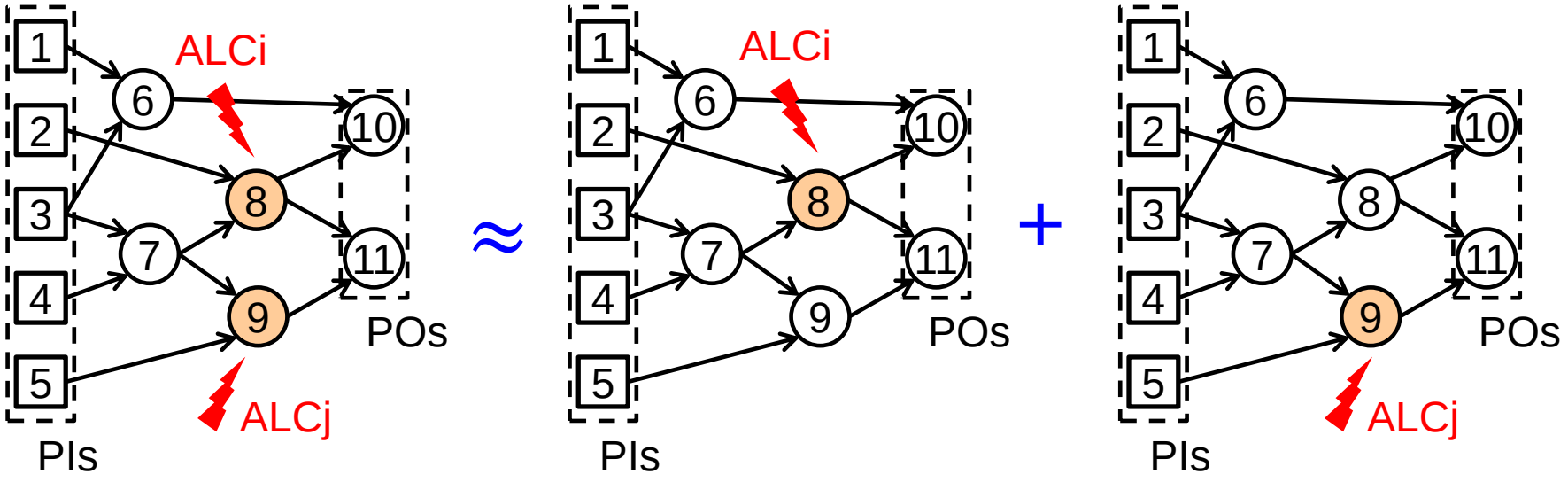
- Approximate local change on a single critical gate is not effective in reducing global delay
- Solution:
  - All the critical paths should be shortened simultaneously
  - Need to find critical cut on the critical graph



Overall Flow



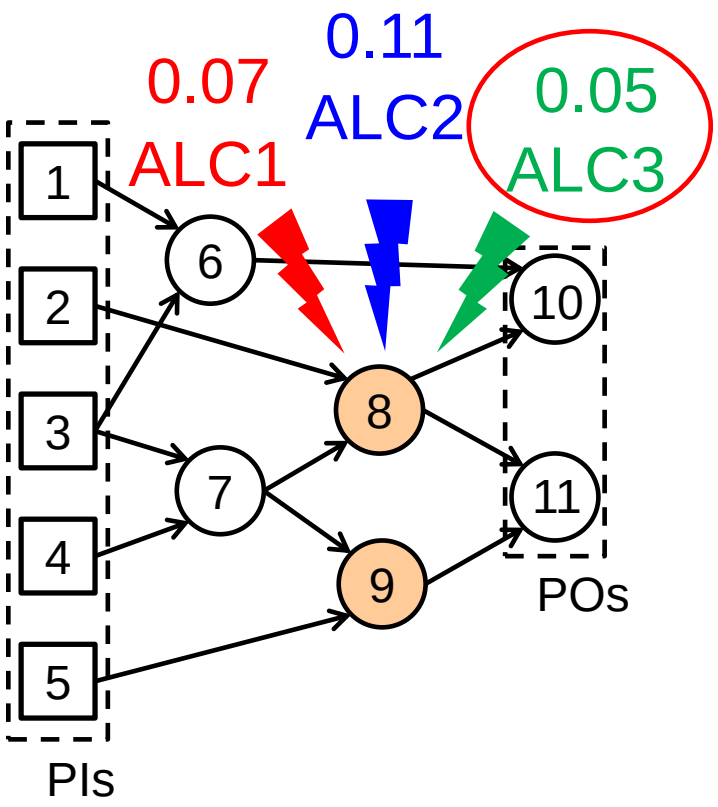
Decompose Error Impact



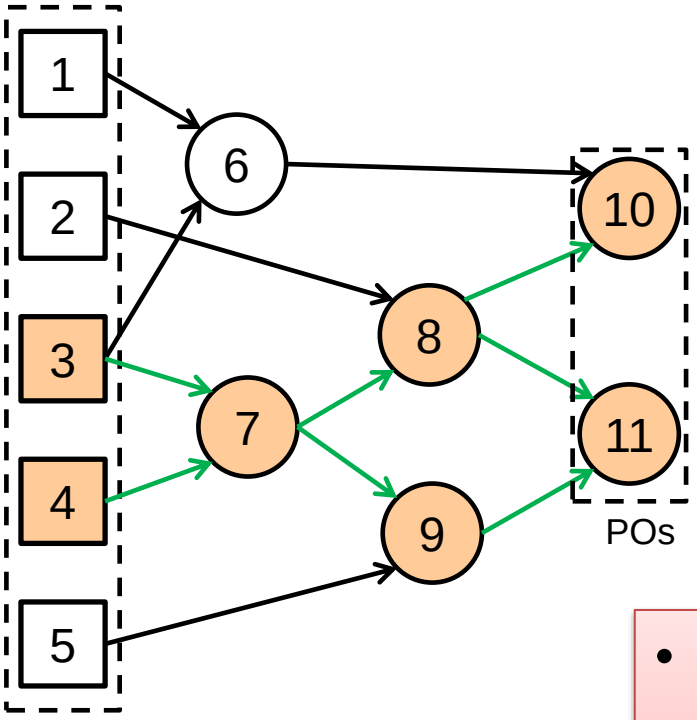
Suitable for error rate (ER) and mean error distance (MED)

Advantages of Error Impact Decomposition

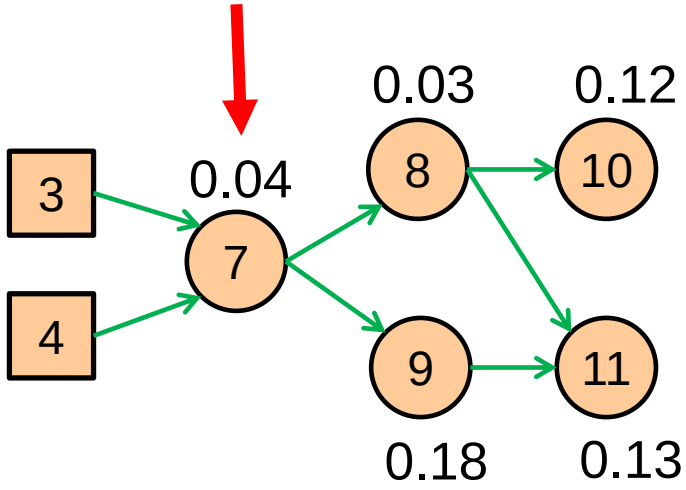
1. Reduce the number of logic simulations (= total number of ALCs over all nodes)
2. Only need to keep the ALC with the **<u>min</u>** error impact for each node



Critical Graph with Min Error Impacts



min error impact over all ALCs for that node



- The problem reduces to finding a min cut
- Min-cut = Max-flow
- Can be solved by a network flow algorithm

Experimental Results

- Compared to [Su+, DAC'18] post-processed by delay-driven traditional logic synthesis

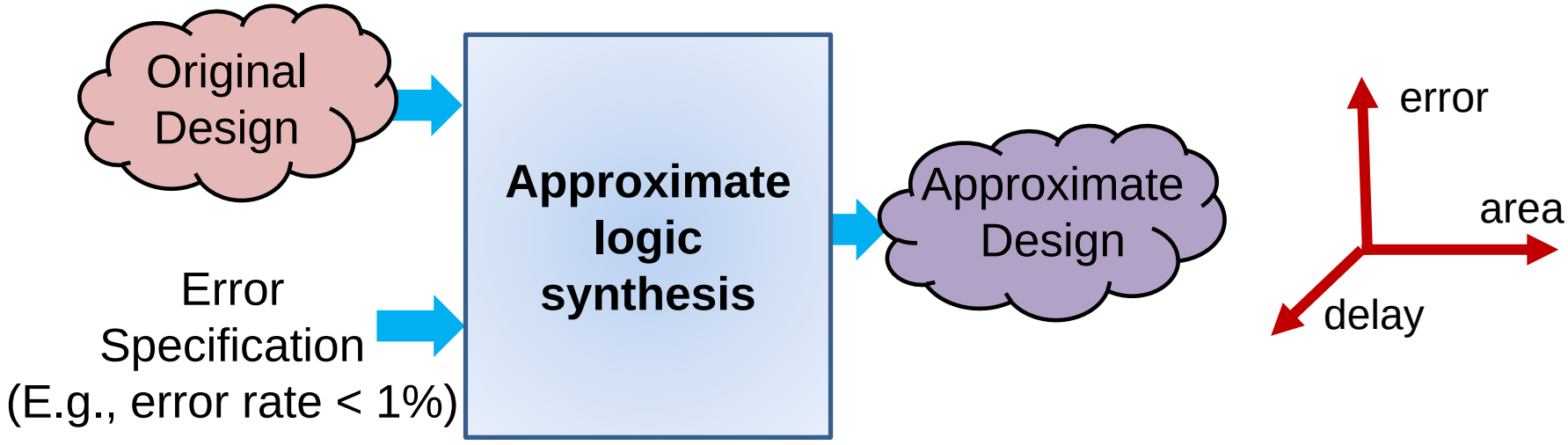
		DALS		[Su+, DAC'18]	
Circuit	**Error rate**	**ΔArea†**	**ΔDelay†**	**ΔArea**	**ΔDelay**
C880	10.73%	17.90%	33.33%	24.04%	16.67%
C1355	12.48%	95.83%	93.83%	41.68%	2.53%
C1908	3.78%	58.24%	55.60%	58.63%	45.14%
C3540	14.31%	19.80%	16.37%	35.67%	8.33%
C5315	15.98%	3.01%	19.92%	13.10%	0.90%
C7552	6.38%	4.43%	16.90%	21.79%	0.91%
ALU4	9.45%	33.86%	19.23%	68.67%	7.69%

Outline

- Background on Approximate Computing
- Area-driven Approximate Logic Synthesis
- Delay-driven Approximate Logic Synthesis
- Conclusion

Conclusion

- Approximate computing
  - Targeting at error-tolerant application
  - Trading accuracy for area/delay/power
- Effective approximate logic synthesis tool
  - Area-driven: formulated as a knapsack problem
  - Delay-driven: formulated as a network flow problem



Acknowledgment

- Faculty: Prof. Jie Han, University of Alberta

- Students: Shuyang Huang, Chang Meng, Chuyu Shen, Sanbao Su, Chen Wang, Yi Wu, Yue Yao, Zhuangzhuang Zhou, Chen Zou

- Funding agency: National Natural Science Foundation of China (NSFC)

Thank You!

Questions?