JOINT INSTITUTE
交大密西根学院

Approximate Logic Synthesis for FPGA by Wire Removal and Local Function Change

Yi Wu, Chuyu Shen, Yi Jia, and Weikang Qian
University of Michigan-SJTU Joint Institute
Shanghai Jiao Tong University

Asia and South Pacific Design Automation Conference
Chiba/Tokyo, Japan, Jan 17, 2017

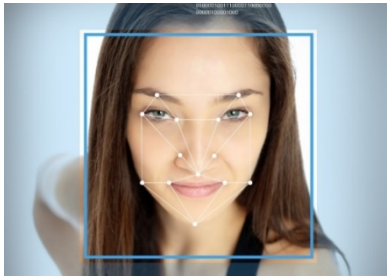
Outline

- Introduction and Motivation
- Proposed Method
- Experimental Results
- Conclusion

Outline

- Introduction and Motivation
- Proposed Method
- Experimental Results
- Conclusion

Error-Tolerant Applications



Low-power VLSI

Error-tolerant applications

Exact Results *Not* Necessary

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

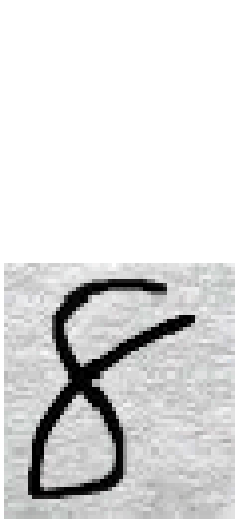
(a) From accurate processing

(b) From inaccurate processing

[Han and Orshansky, 2013]

Numerical Values *Not* Matter

- Handwritten Digit Recognition



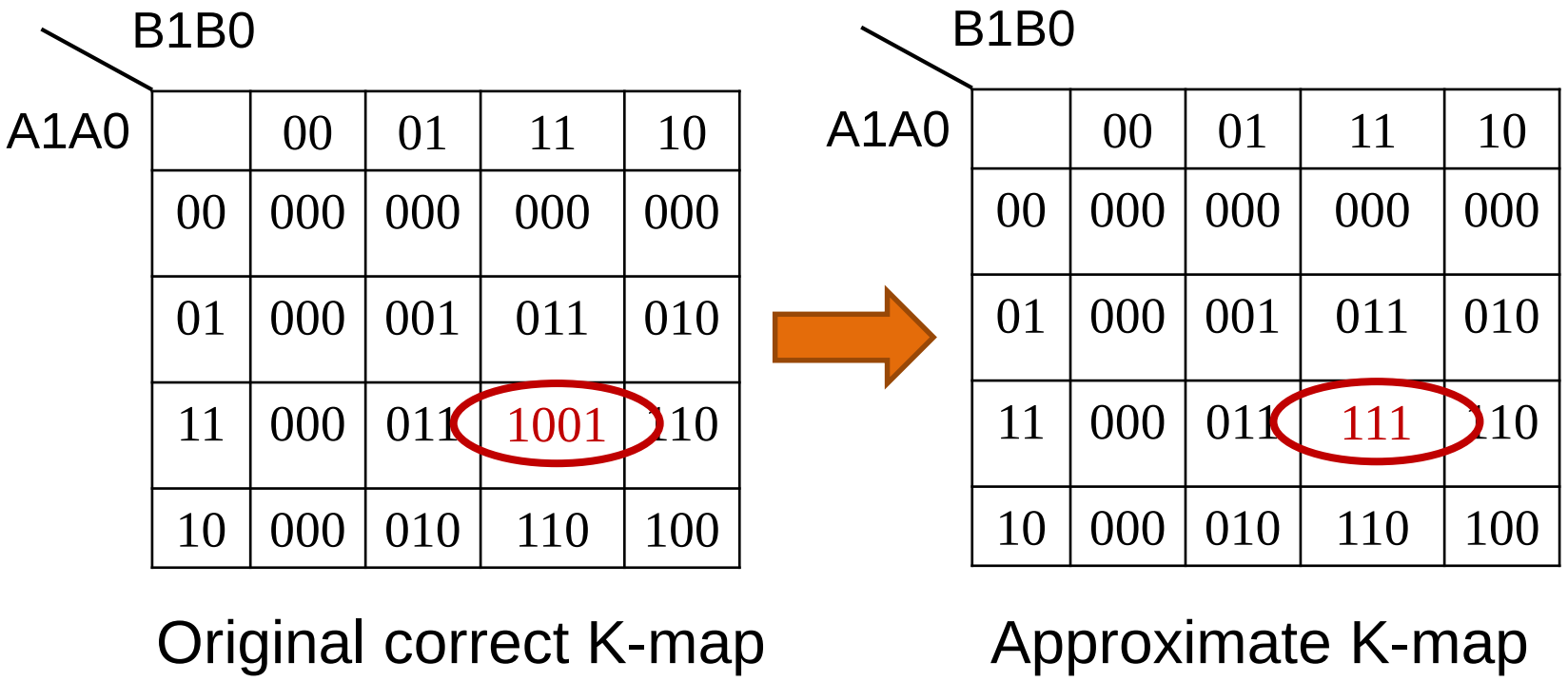
Similarity (probability)

8	~~0.9257~~ 0.9306 ✓ 0.8928
0	0.1563
1	0.0984
⋮	⋮
9	0.2435

No Golden Standard Answer

Approximate Computing

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



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

Original correct K-map

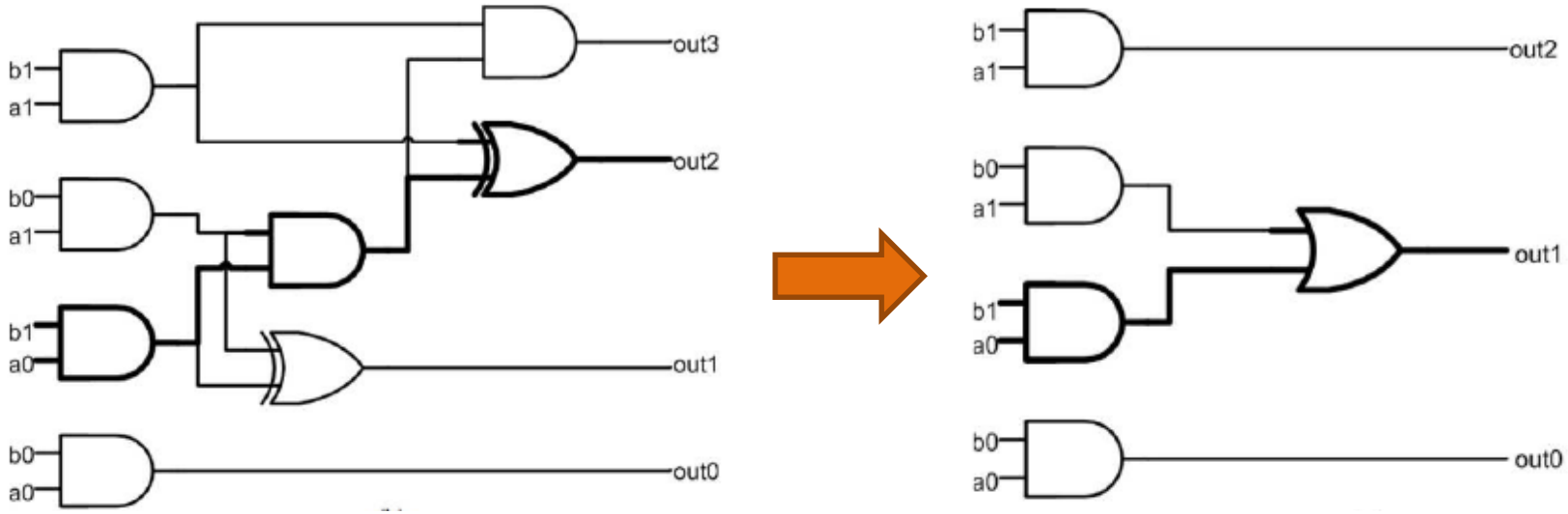
A1A0 \ B1B0	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	111	110
10	000	010	110	100

Approximate K-map

[Kulkarni et al. 2011]

Approximate Computing

- A 2-bit approximate multiplier



Accurate digital circuit

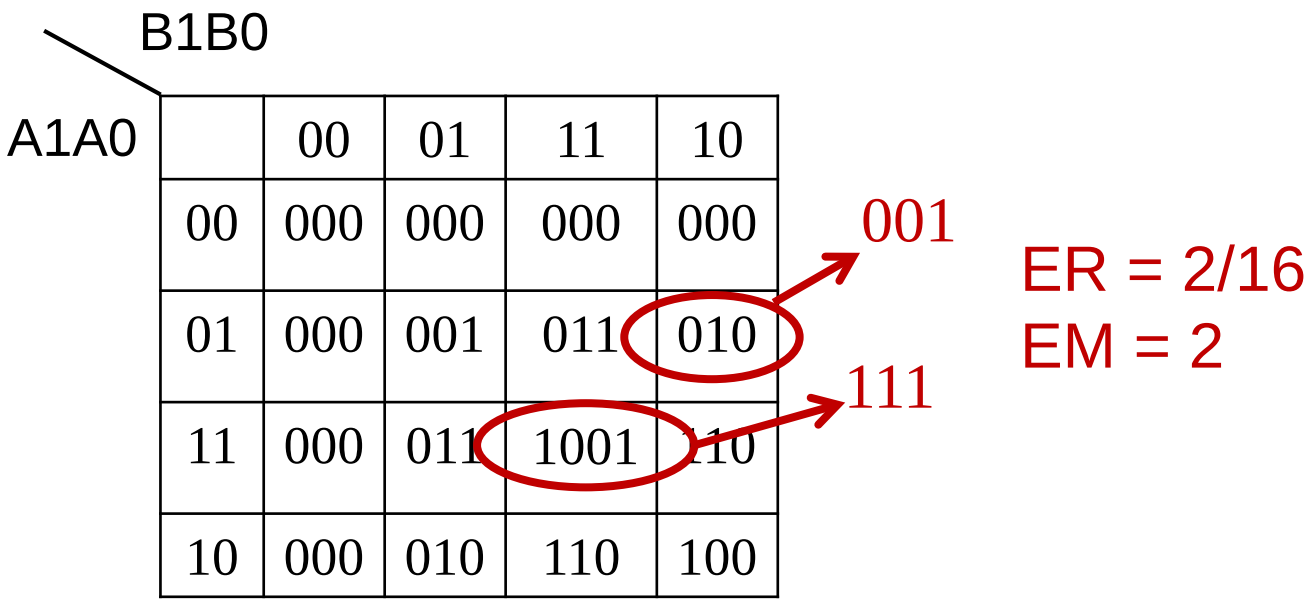
Approximate digital circuit

Comparison

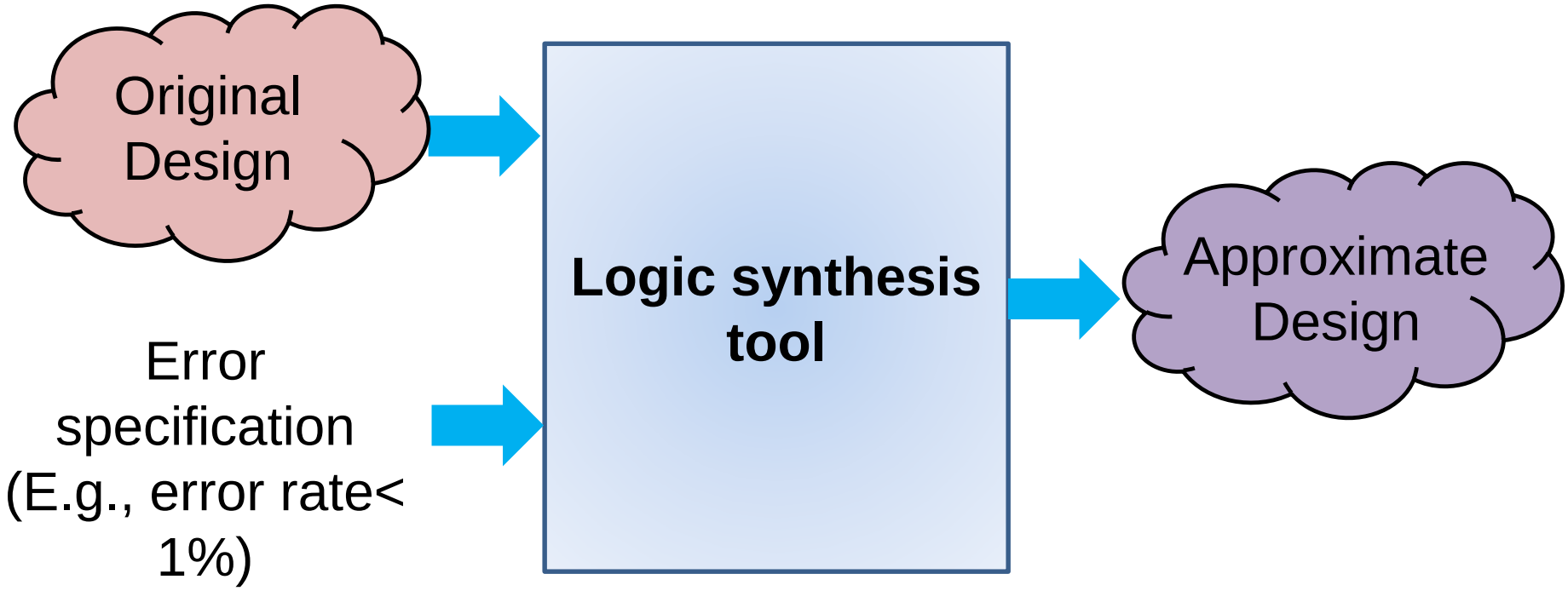
Nearly half area; Shorter critical path; Smaller switching capacitance (less dynamic power)

Error Metrics for Approximate Computing

- Error rate (ER): $\Pr(\vec{f}(\vec{X}) \neq \vec{f}'(\vec{X}))$
- Error magnitude (EM): $\max_{\vec{X}} |\vec{f}(\vec{X}) - \vec{f}'(\vec{X})|$
  - Applicable to arithmetic circuits



Approximate Logic Synthesis (ALS)



- Input: an original circuit and error specification
- Output: an optimal approximate circuit, satisfying the error constraint

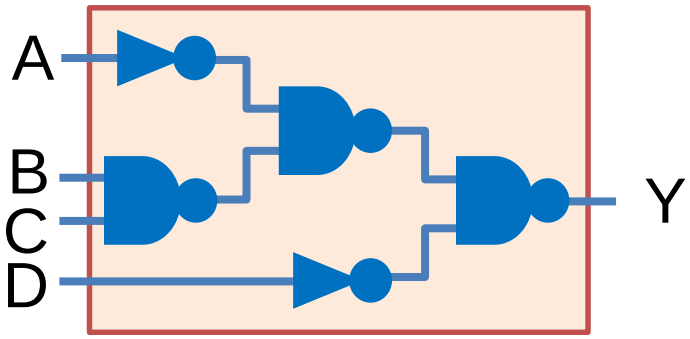
Related Works: ALS for Multi-level Circuits

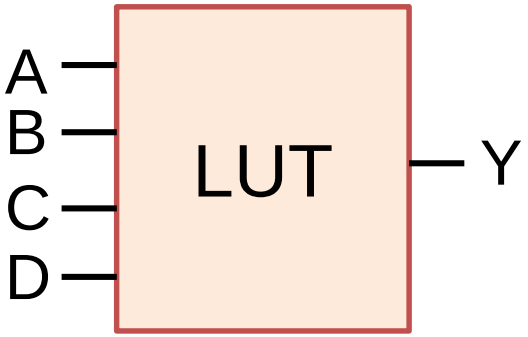
- Inject stuck-at-faults, composite constraint on ER & EM [Shin and Gupta, DATE'2011]
- SALSA — Encode EM constraint as a funciton [Venkataramani et al., DAC'2012]
- SASIMI — Substitute a signal with another one that takes the same value with high probability, ER constraint [Venkataramani et al., DATE'2013]
- Solve ER-unconstrained ALS first: ER + EM [Miao et al., ICCAD'2014]
- Remove literals from factored-form expressions of local nodes: ER [Wu and Qian, DAC'2016]

All of these works target at ASICs !

FPGA Architecture

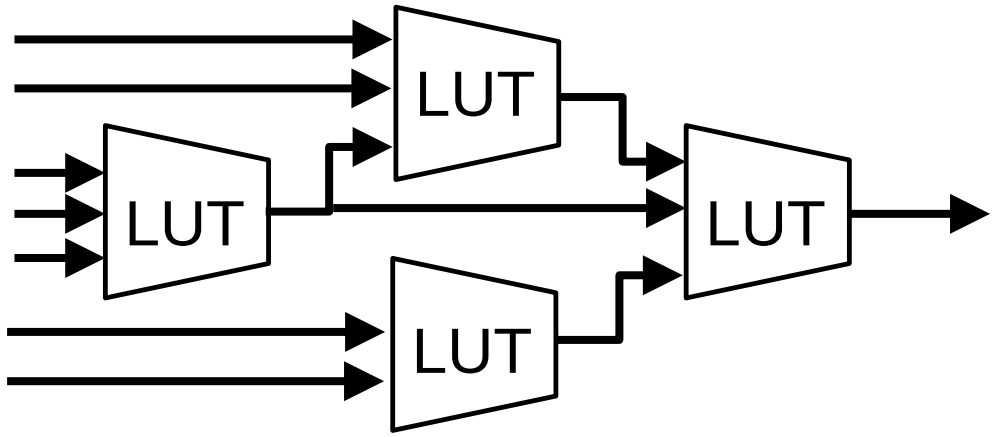
- Basic building block: k-input look-up table (k-LUT)
  - Can implement any Boolean function with up to k variables



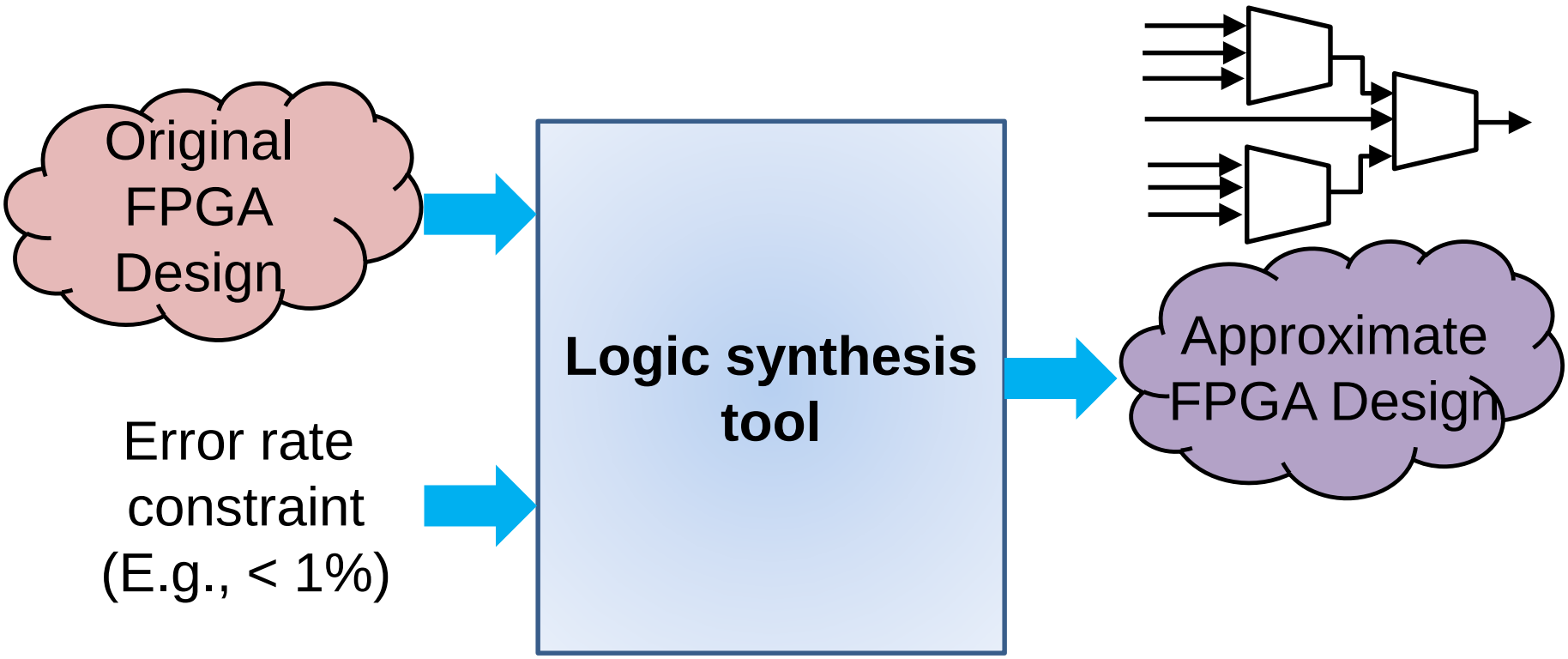


A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
		...		
1	1	1	0	1
1	1	1	1	0

In FPGA, a circuit is a LUT network



Problem Formulation



Approximate Logic Synthesis for FPGAs

- Input: an original FPGA design and error rate constraint ER
- Output: an FPGA design with minimal area, satisfying the error rate constraint

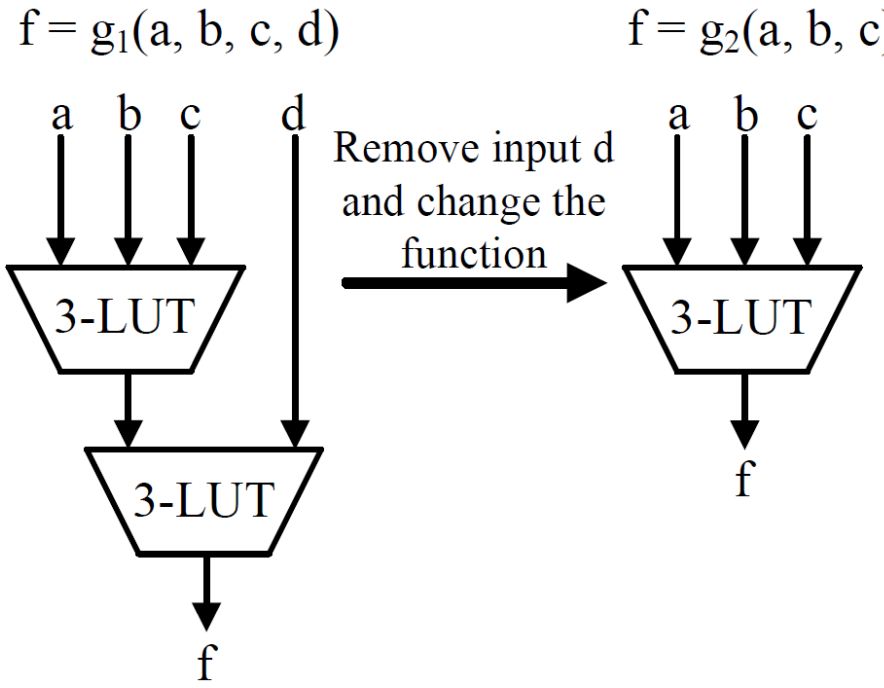
$\propto$ power, usually

Outline

- Introduction and Motivation
- Proposed Method
- Experimental Results
- Conclusion

Basic Idea

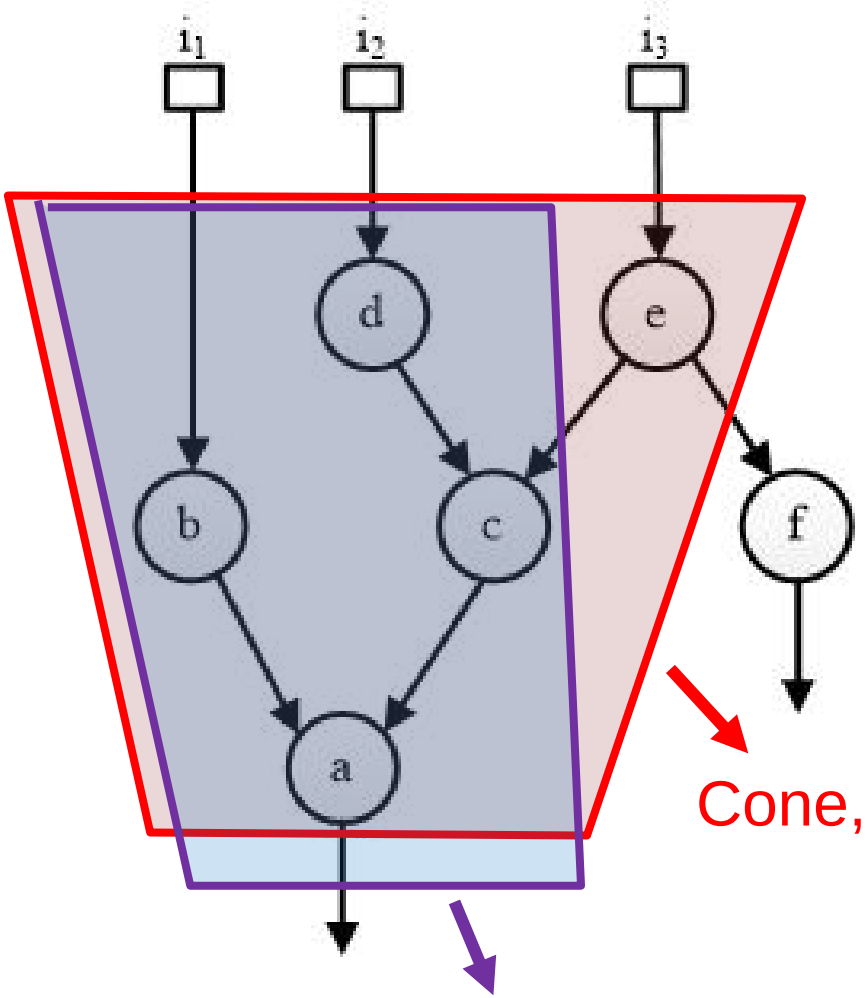
- Remove inputs + Change function
  - Remove inputs: can reduce #LUT
  - Change function: minimize error



Where to apply the basic idea?
— Fanout-free cone (FFC) in the network

Note: when # inputs from 4 to 3, the number of LUTs needed is guaranteed to be reduced by 1! *Not* guaranteed for ASIC!

FFC (Fanout-Free Cone)

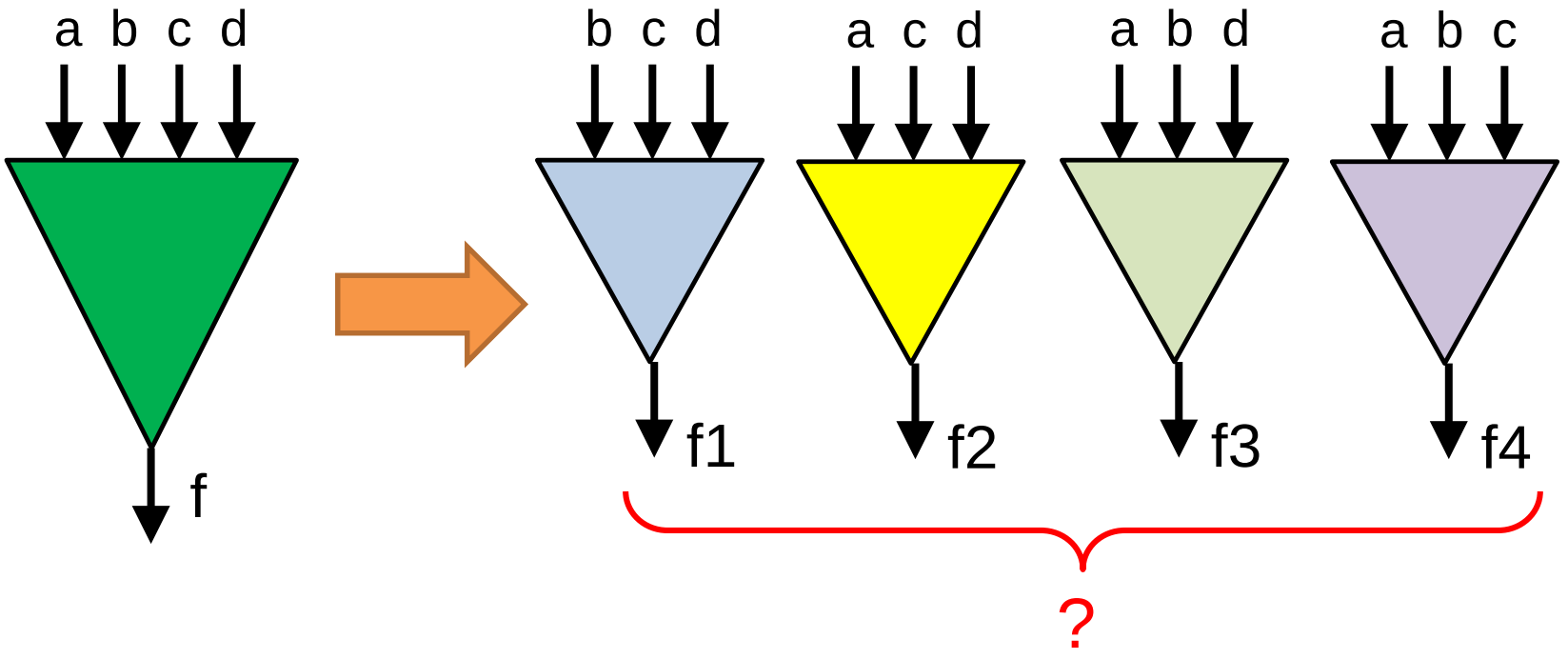


Cone of v : for any node w in C_v, there is a path from w to v that lies entirely in C_v.

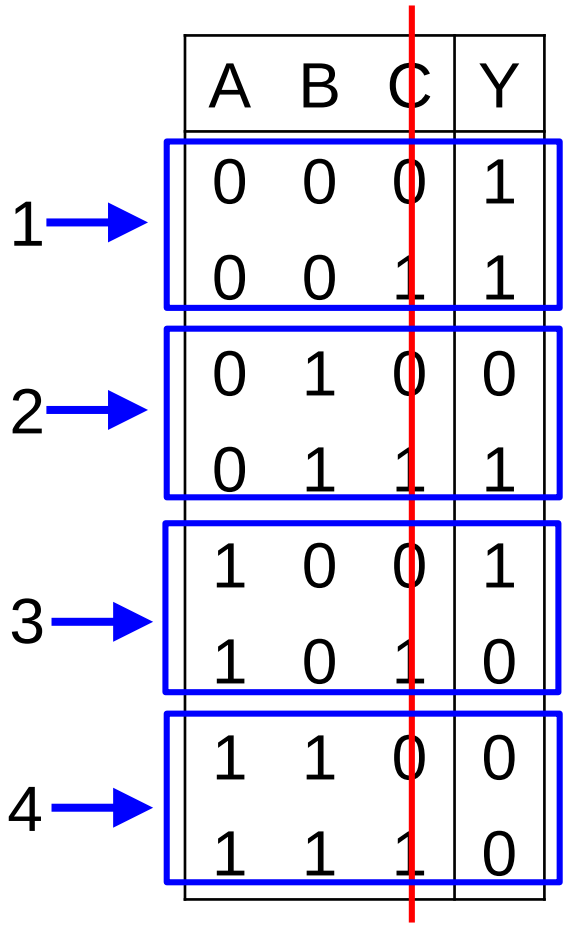
FFC of v : a cone in which the fanouts of every node other than the root v are in the cone.

Proposed Method

- Apply the basic idea to FFCs in a network
- For each FFC of each node, consider all cases in which one input of the FFC is removed



Determining the New Function



Suppose Pr(010)>Pr(011)

Suppose Pr(100)>Pr(101)

A	B	Y
0	0	? → 1
0	1	? → 0
1	0	? → 1
1	1	? → 0

Rule: if output values differ, always keep the output value of the input pattern with larger probability

⇒ the induced error rate is minimal

Candidate Transformations

- Map the modified function of an FFC into a new LUT network
  - If fewer LUTs than original FFC
  - If the error rate <= error margin
  - If using it won't increase delay

  } *candidate transformation*

- Many candidate transformations exist
  - We will pick multiple candidate transformations to apply
  - Formulated as an ***Integer Linear Programming (ILP)***

ILP Formulation

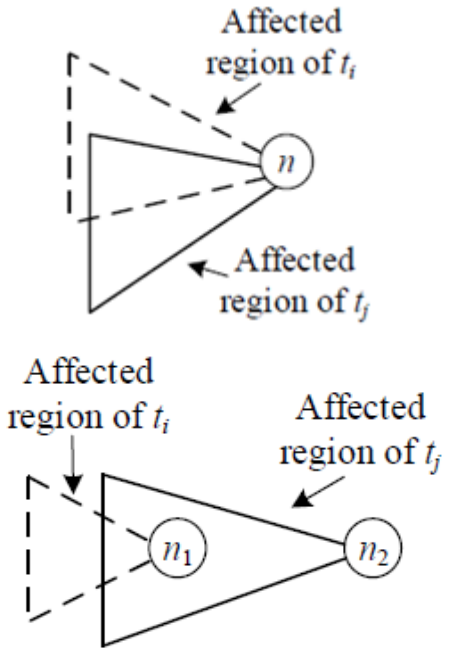
- m candidate transformations: $t_1, t_2, \ldots, t_m$, each associated with a binary variable v_i;
- For each t_i, its error rate is e_i, LUT number reduction is a_i.

Objective: maximize $\sum_{i=1}^{m} a_i v_i$ (a) (total reduction in LUT number)

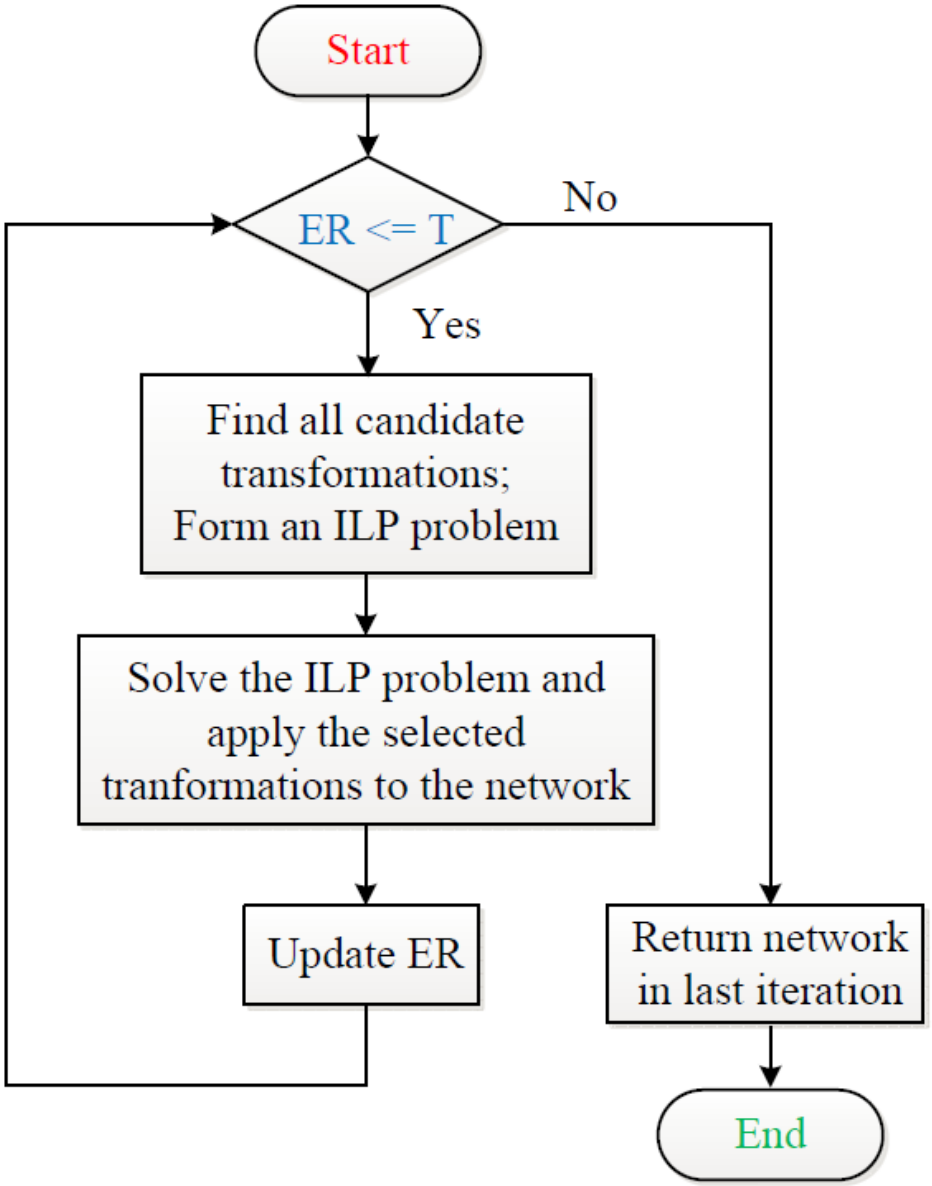
Constraints: $v_i + v_j \leq 1$ (b) (overlapped transformations cannot be selected at same time)

$\sum_{i=1}^{m} e_i v_i \leq t$ (c) (total error rate should be within error rate margin t)

Overlapped Transformations



Algorithm



ER: error rate of current network

T: user-specified error rate threshold

ILP solver: GLPK

Reference:
http://www.gnu.org/software/glpk/

Speed-up Techniques

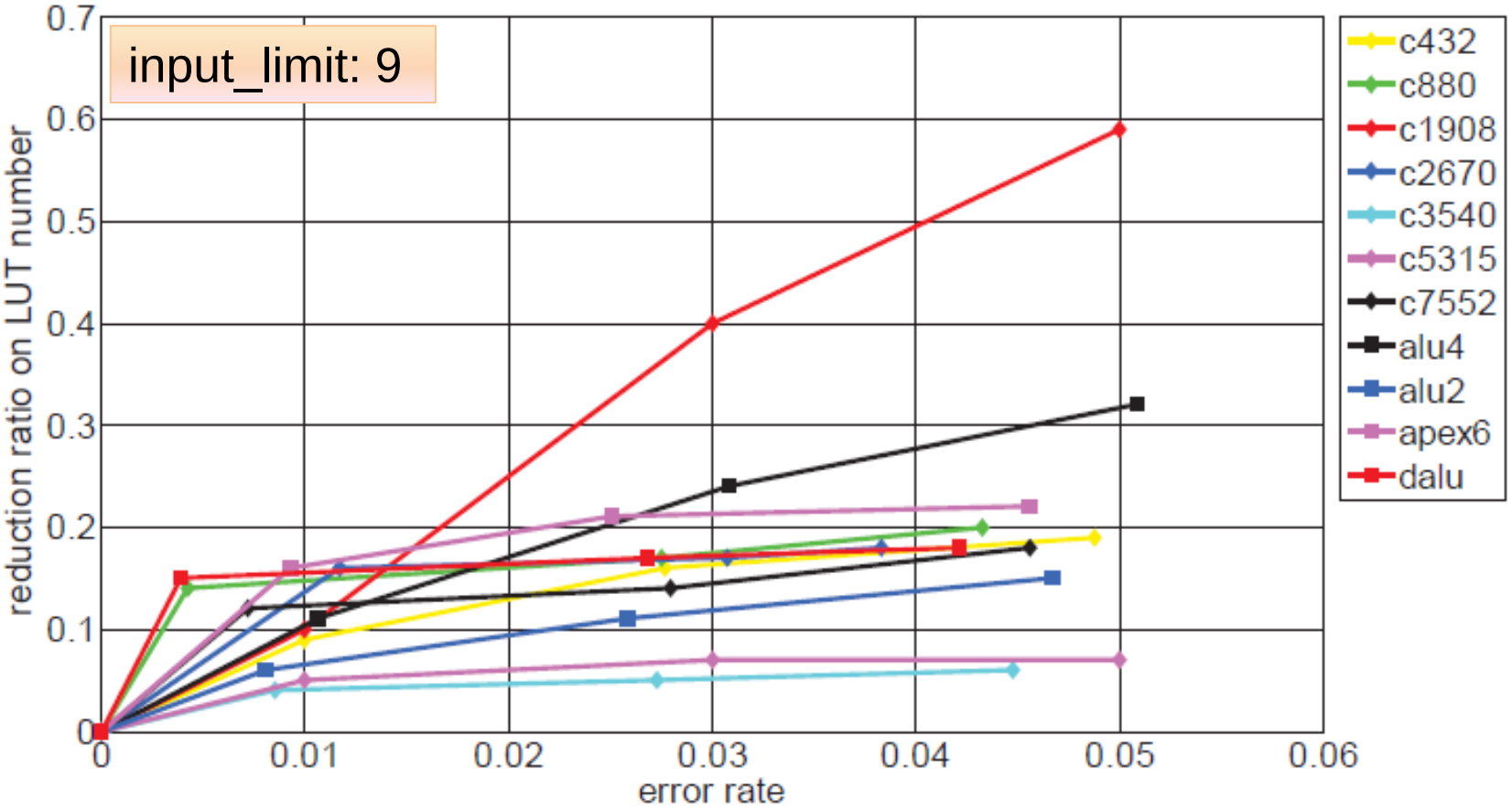
- Set upper bound on the number of inputs of FFC we consider: input_limit = 7~9.
- Discard transformations whose scores are the lowest 40% among all candidate transformations of each node, where score = #LUT_save/error_rate.
- Exclude FFCs whose all input patterns have probability > error rate margin.

Outline

- Introduction and Motivation
- Proposed Method
- Experimental Results
- Conclusion

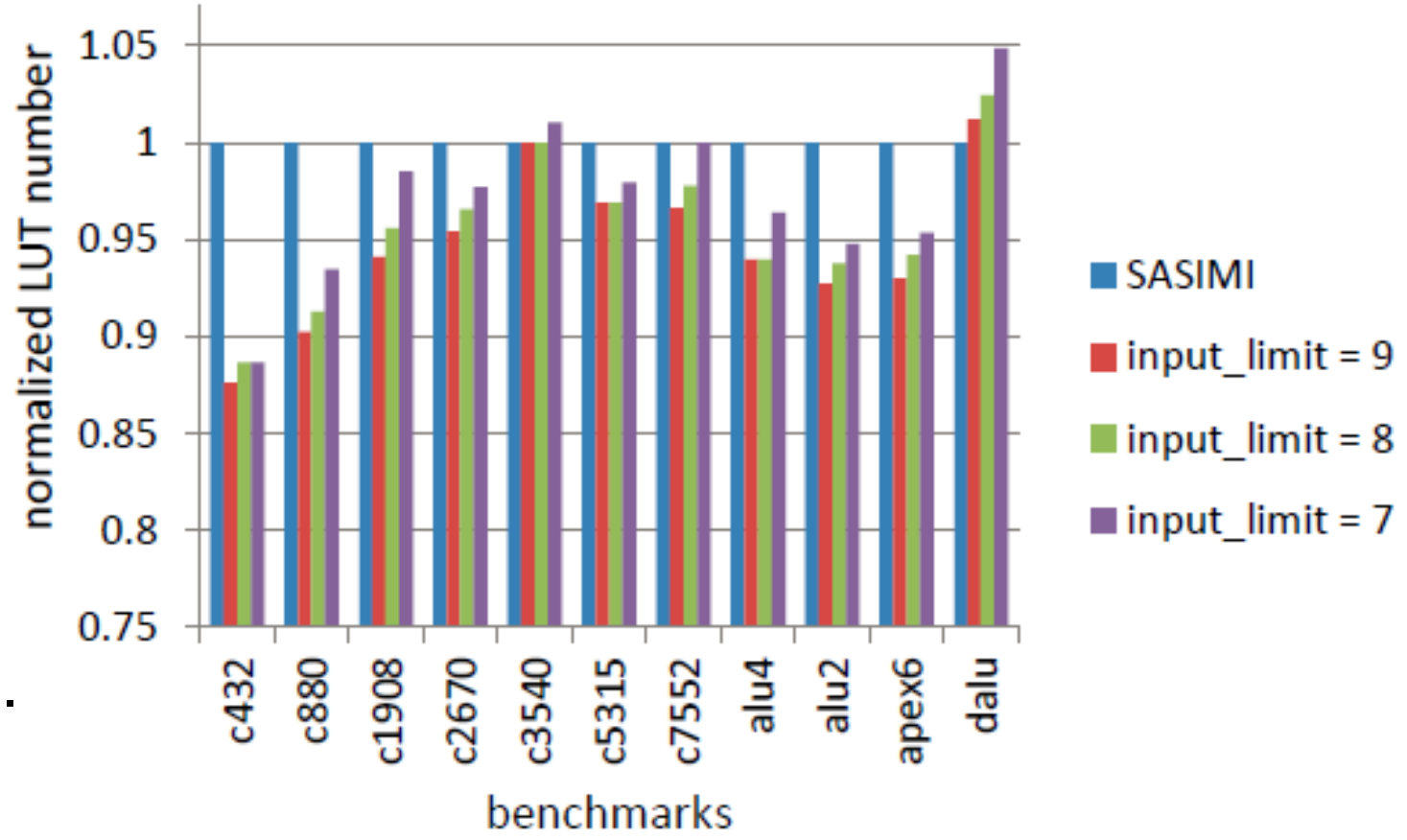
Experimental Results: # LUT Saving

- **Setup:** C++ on a virtual machine running Linux OS (1GB memory); FPGA technology: 4-LUT; Assume all PI patterns are of same probability



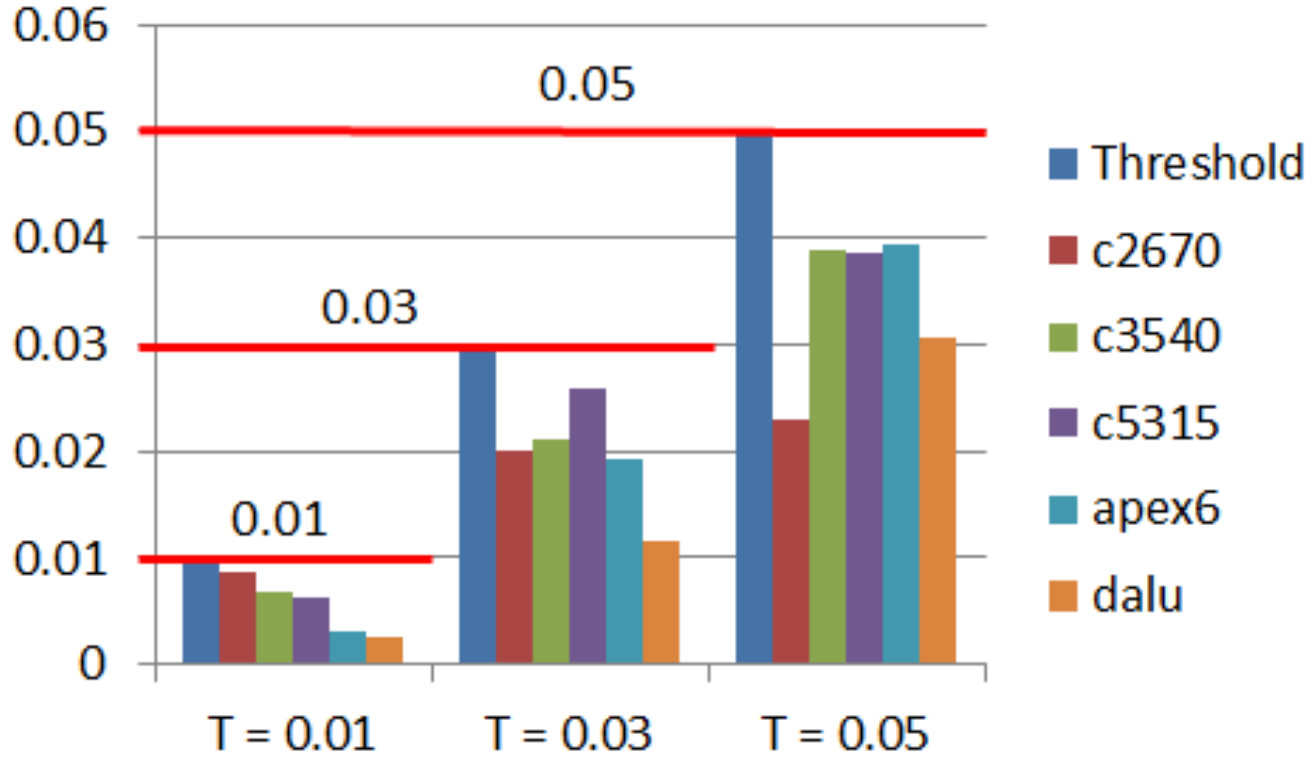
Experimental Results: Comparison

- Compared with SASIMI
  - *input_limit* = *9*: 87%~97% of that produced by SASIMI in terms of LUT number



Remaining Error Rate Margin after SASIMI

Error rate of the final approximate circuits obtained by SASIMI



Observation: some approximate circuits obtained by SASIMI still have error rate margin.

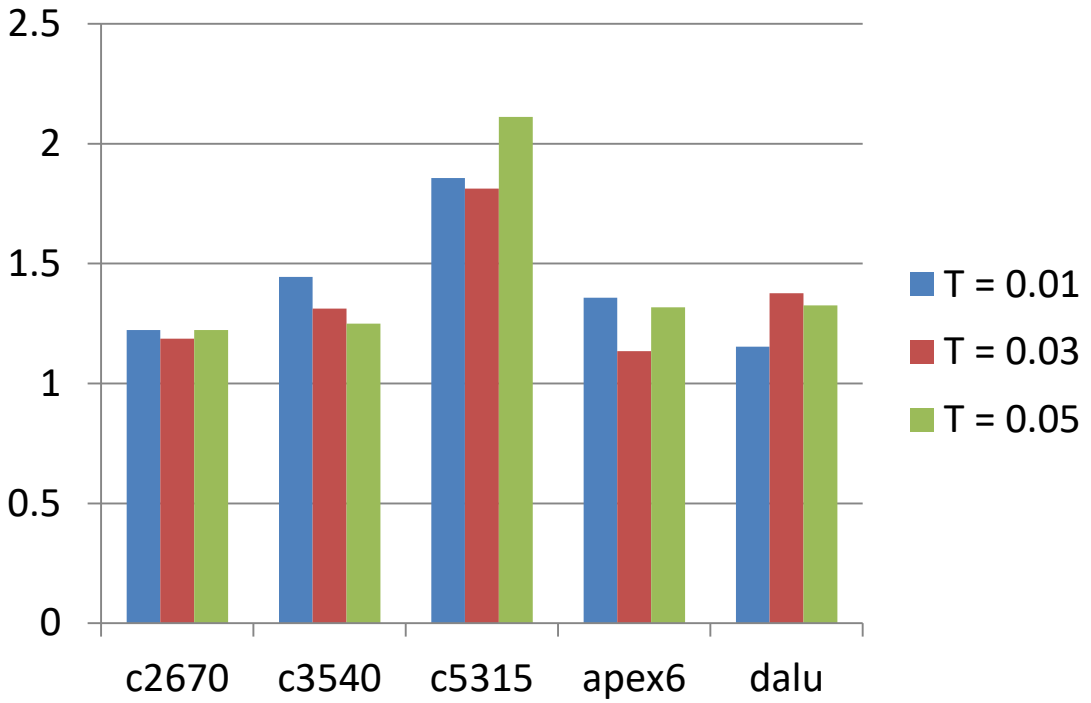
Experimental Results by SASIMI + Ours

$$\frac{\#\ LUT\_save\ \ by\ SASIMI + Ours}{\#\ LUT\_save\ \ by\ SASIMI}$$

Method combination

1. Apply SASIMI with original T

2. Apply Ours with T = remaining error rate margin



Our method is able to search a different solution space and is a good complement to SASIMI.

Outline

- Introduction and Motivation
- Proposed Method
- Experimental Results
- Conclusion

Conclusion

- Multi-level approximate logic synthesis for FPGAs under error rate constraint
- A novel technique which removes an input of an FFC and changes its function simultaneously.
- Formulation of an ILP problem for the selection of multiple changes and a practical flow to synthesize approximate FPGA designs.

Thank You!

Questions?