Time Multiplexing via Circuit Folding

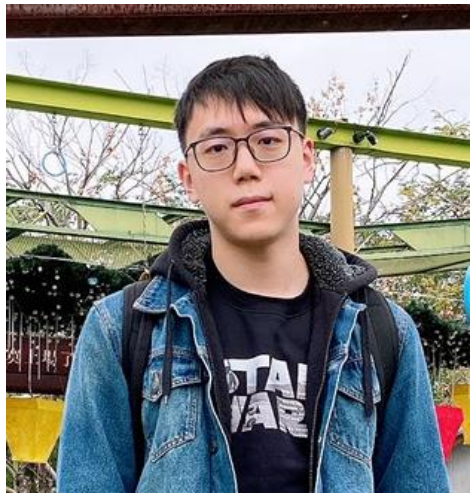
Po-Chun Chien†, Jie-Hong Roland Jiang†‡

ALCom Lab

‡Department of Electrical Engineering
†Graduate Institute of Electronics Engineering
National Taiwan University

Speaker Bio

- Po-Chun Chien is a 2nd-year master's student.
- I am planning to apply for PhD (fall 2021).
- Contact: r07943091@ntu.edu.tw
- Affiliation:
  - ALCom Lab (led by Jie-Hong Roland Jiang)
  - GIEE, NTU
- Research Interests:
  - Logic synthesis
  - System verification
  - Machine learning



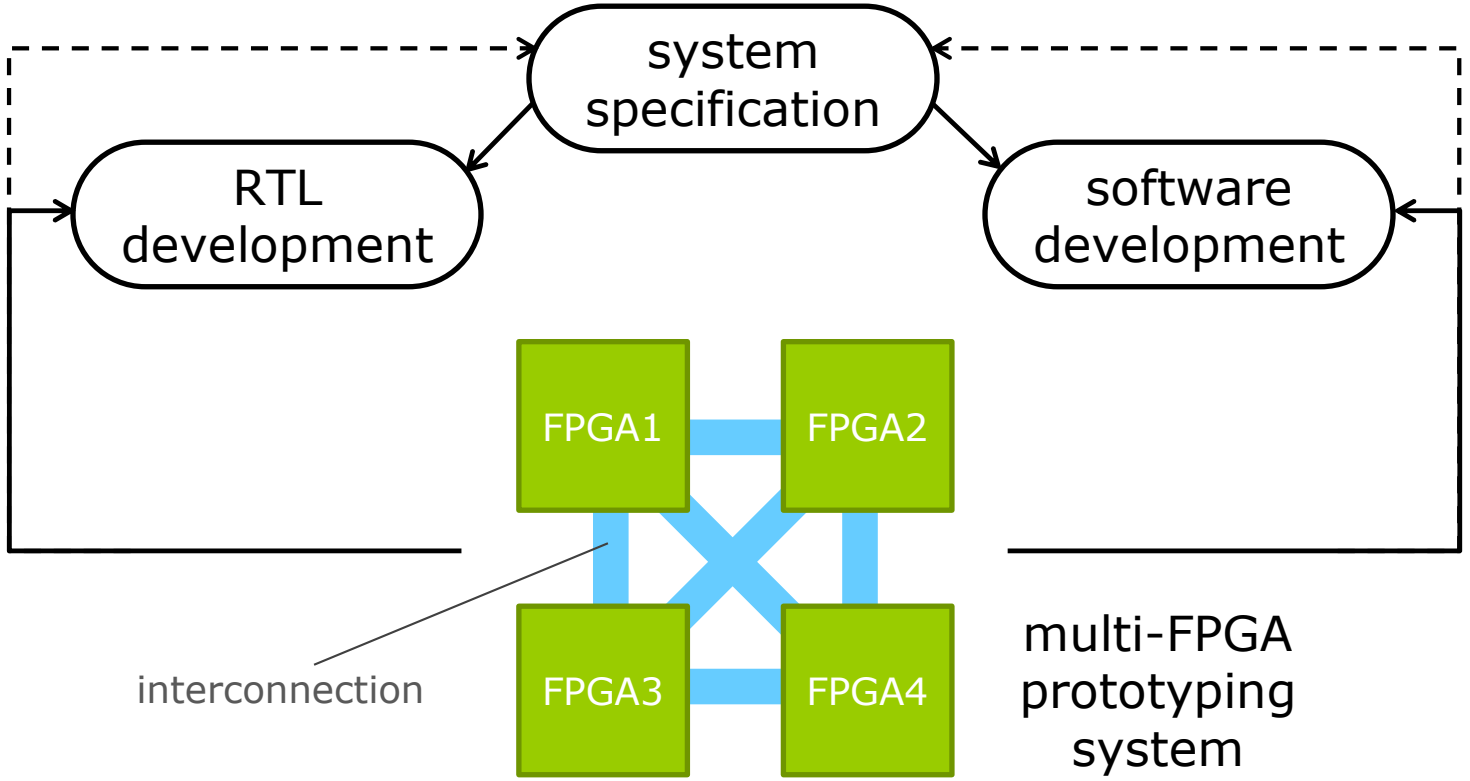
Outline

- Introduction
- Algorithm
  - Structural folding method
  - Functional folding method
- Experiments
- Conclusions & future work

Introduction

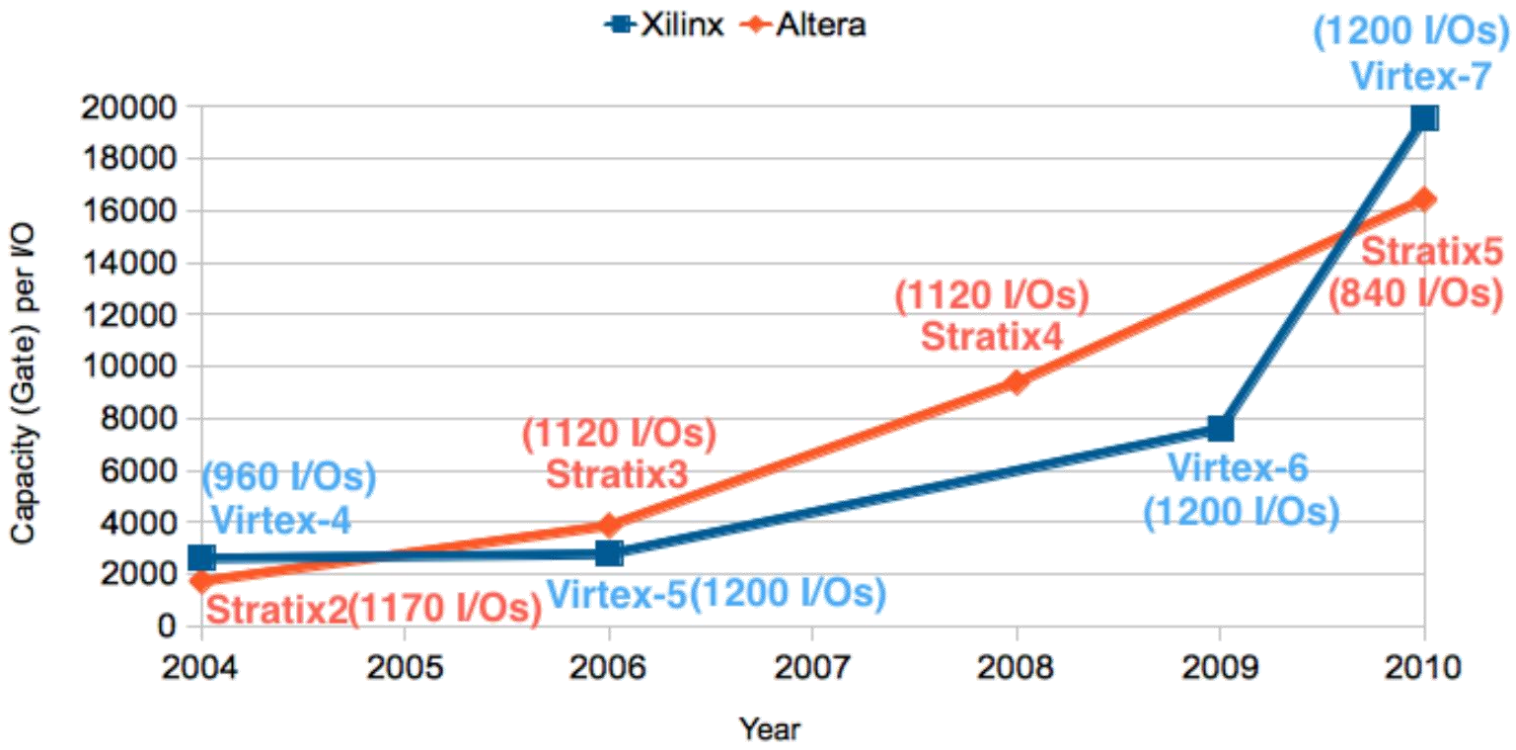
Multi-FPGA System

- Multi-FPGA boards are commonly used for system emulation [1] and prototyping.



[1] Myaing et al., 2011.

FPGA I/O bottleneck



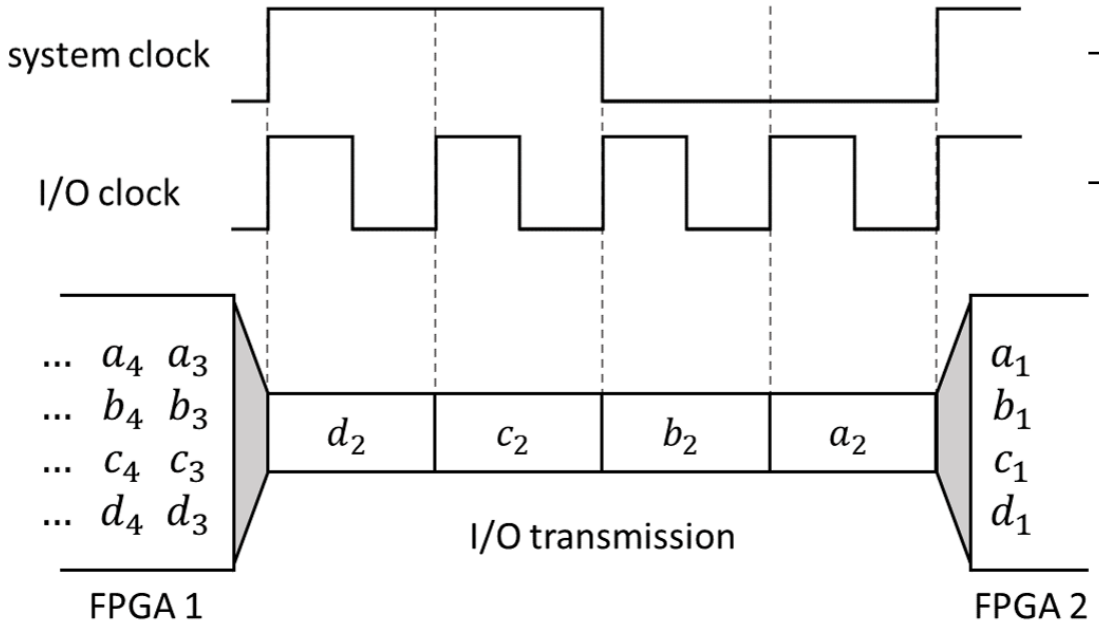
ratio of FPGA logic capacity over I/Os (retrieved from [2])

→ I/Os become scarce resources

[2] Hung et al., 2018.

Time Division Multiplexing (TDM)

- The system requires 2 separate clocks.



TDM [3] I/O transmission with ratio 4

→ system operation

→ I/O data transmission

↓

4x #I/O can be transmitted in 1 system clock

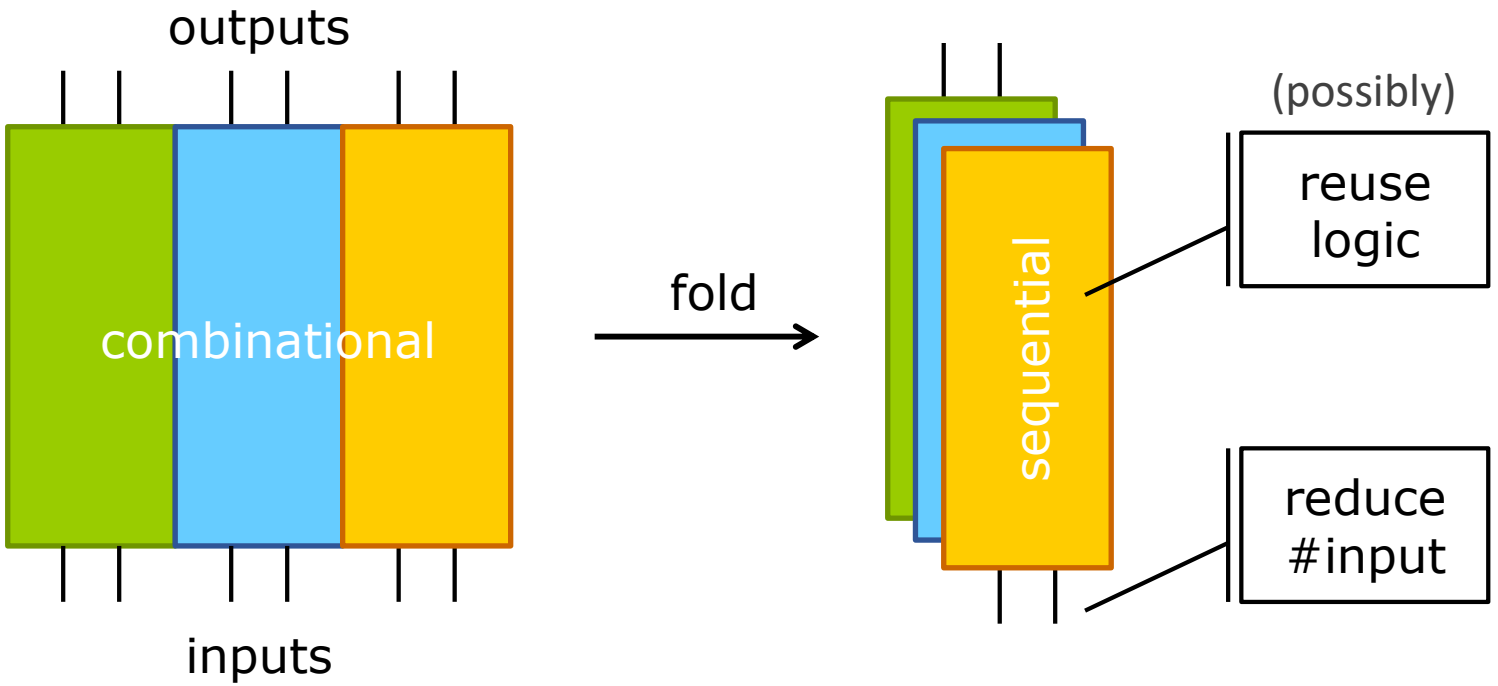
↓

increase the effective I/O of FPGAs

[3] Babb et al., 1993.

Motivation

- Instead of **increasing the effective I/O**, we try to **decrease the required input pin** by folding the circuit.



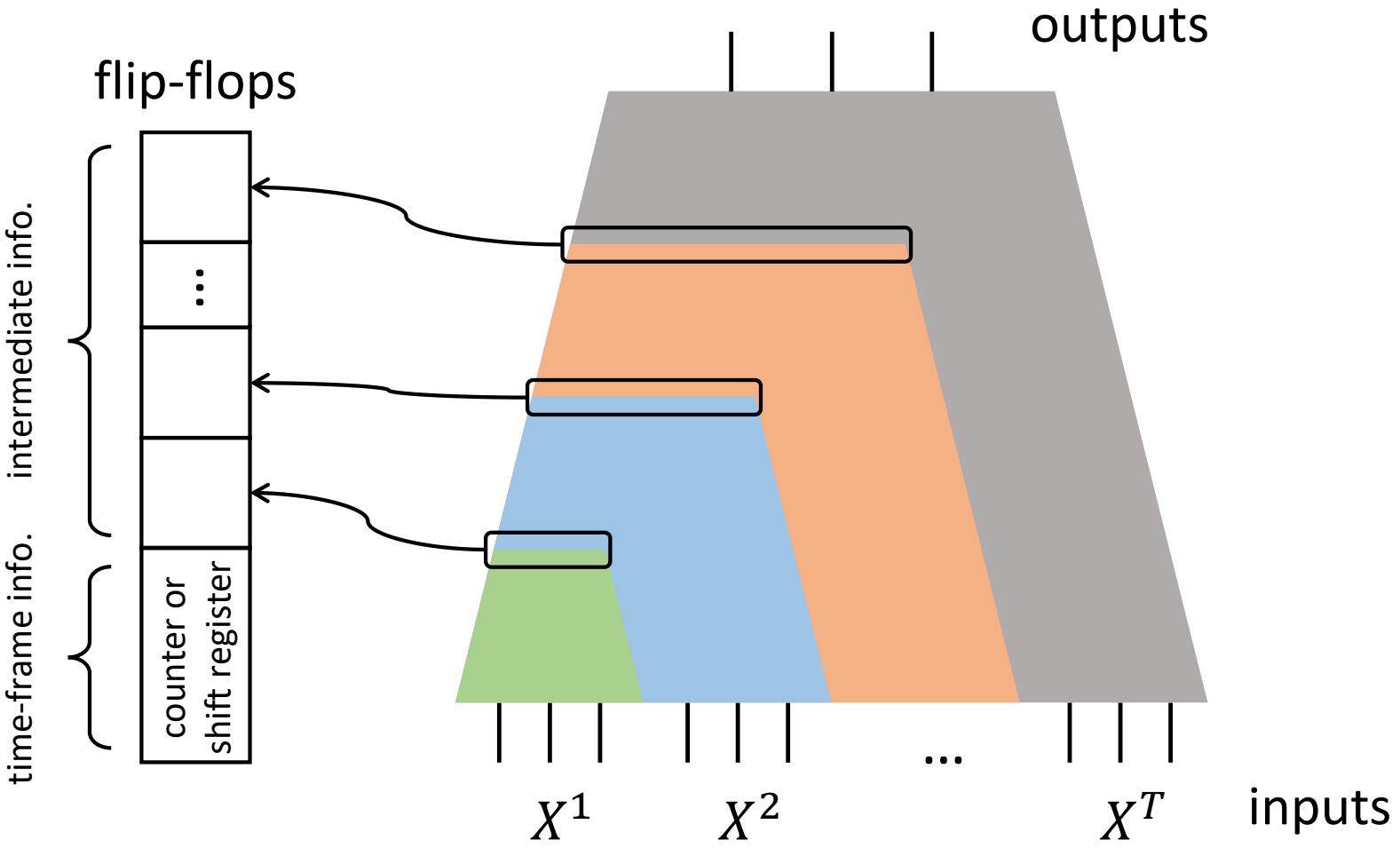
Problem Formulation

- Given a combinational circuit C_c with n inputs and m outputs, and a folding number T, we are asked to fold C_c into a sequential circuit C_S, which
  - has $\boldsymbol{n/T}$ **inputs** and ≥ m outputs, and
  - after expanding for T time-frames, becomes functionally equivalent to C_c under proper association of their inputs and outputs.

Algorithm

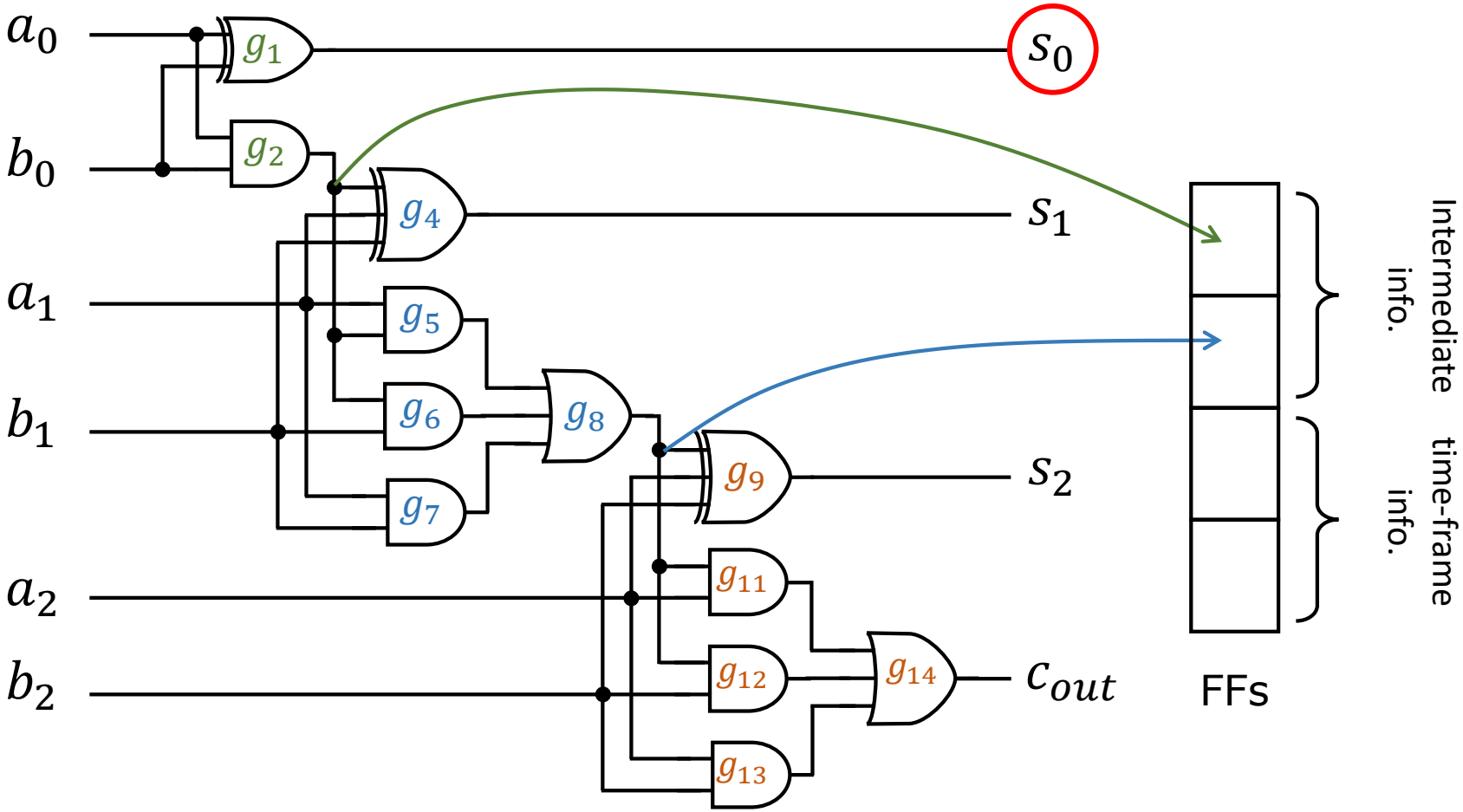
Structural Method

- Illustration



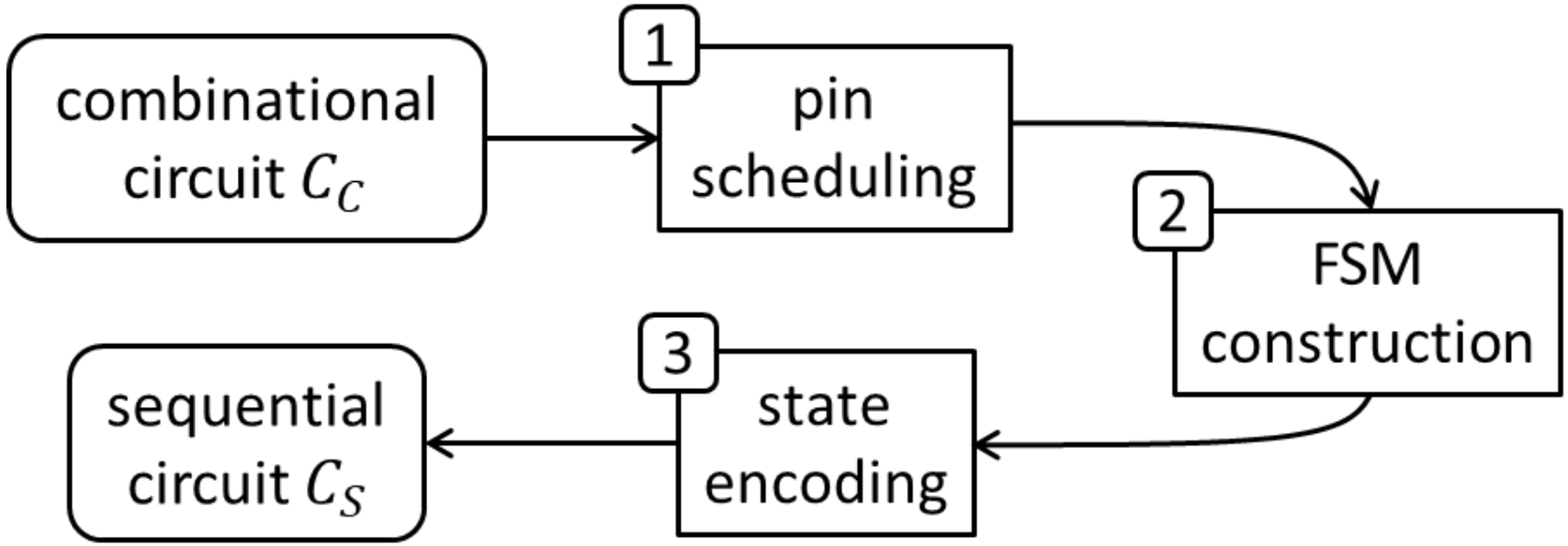
Structural Method

- 3-adder example ($T = 3$)

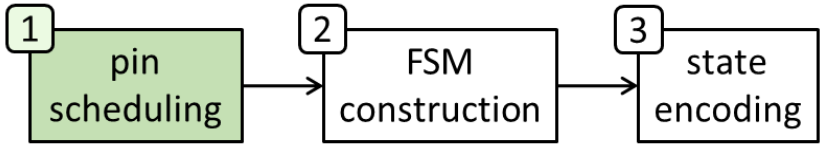


Functional Method

- Computation flow



Functional Method

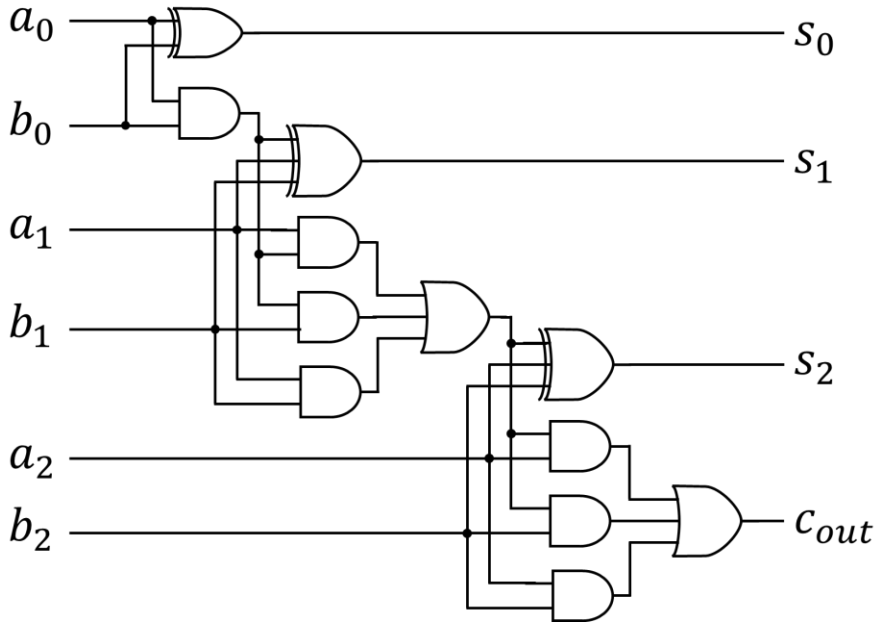


- Pin scheduling heuristic:

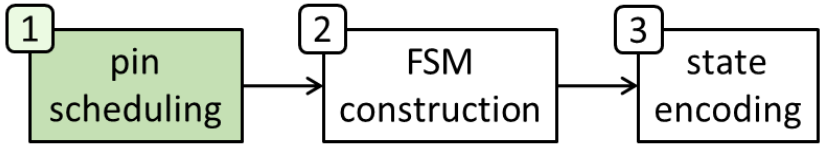
  Convert the given combinational circuit into **virtual iterative** form.

  - Output pin scheduling
  - Input pin scheduling

follow the previous 3-adder example for illustration



Functional Method

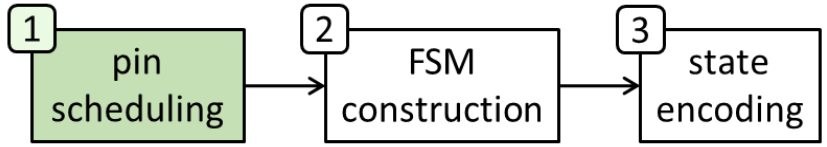


- Output pin scheduling:
  1. sort the outputs according to their support sizes in an ascending order.

output	s_0	s_1	s_2	c_{out}
support	a_0, b_0	a_0, b_0, a_1, b_1	$a_0, b_0, a_1, b_1, a_2, b_2$	$a_0, b_0, a_1, b_1, a_2, b_2$
|support|	2	4	6	6

ascending order

Functional Method

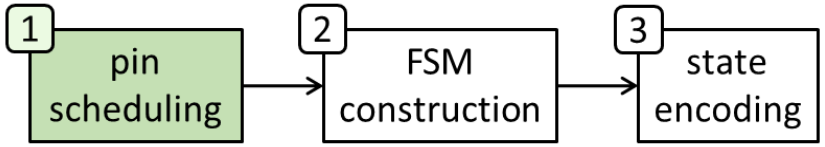


- Output pin scheduling:
  2. determine the iteration of each output to be scheduled at.

#input of the folded circuit = 2

output	s_0	s_1	s_2	c_{out}
support	a_0, b_0	a_0, b_0, a_1, b_1	$a_0, b_0, a_1, b_1, a_2, b_2$	$a_0, b_0, a_1, b_1, a_2, b_2$
|support|	2 / 2	4 / 2	6 / 2	6 / 2
iteration	1	2	3	3

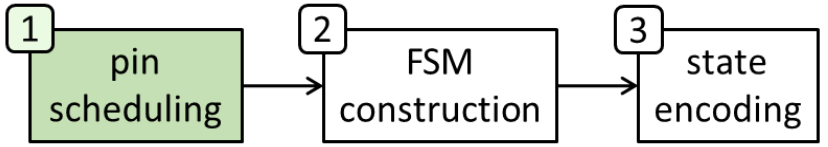
Functional Method



- Output pin scheduling:
  3. null outputs insertion.

iteration	1	2	3
scheduled outputs	$s_0, null$	$s_1, null$	s_2, c_{out}

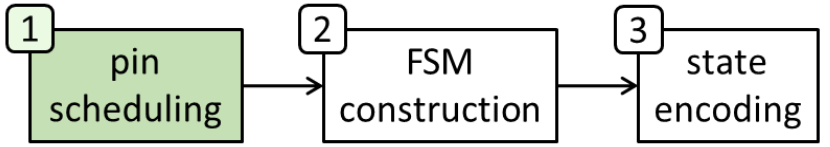
Functional Method



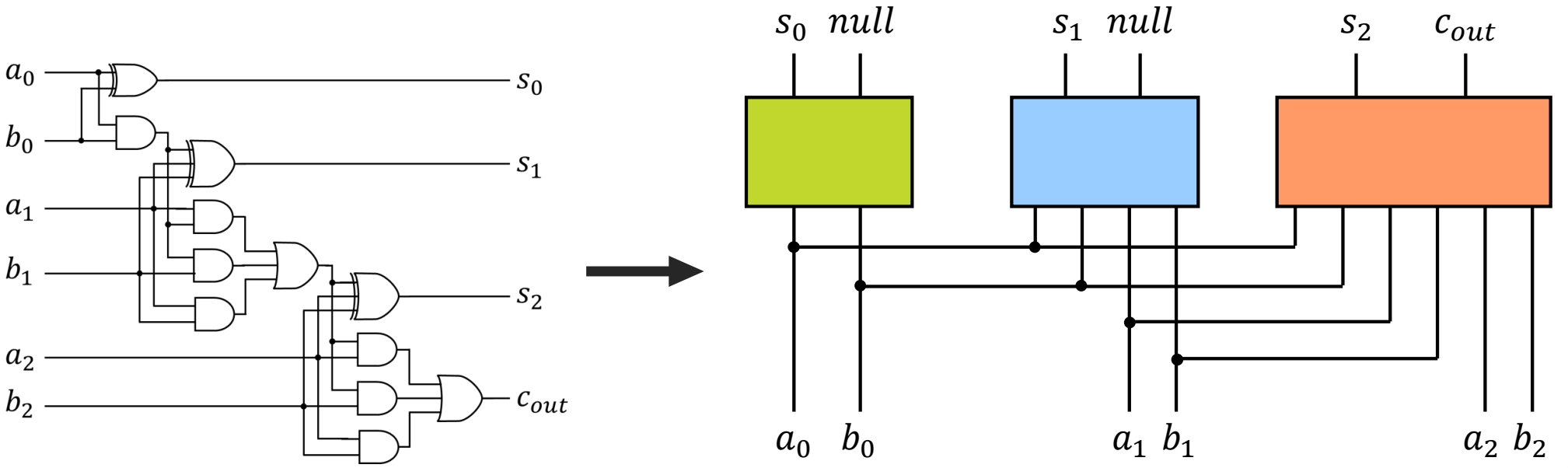
- Input pin scheduling:
  - schedule the inputs according to the outputs.

iteration	1	2	3
scheduled outputs	$s_0, null$	$s_1, null$	s_2, c_{out}
scheduled inputs	a_0, b_0	a_1, b_1	a_2, b_2

Functional Method

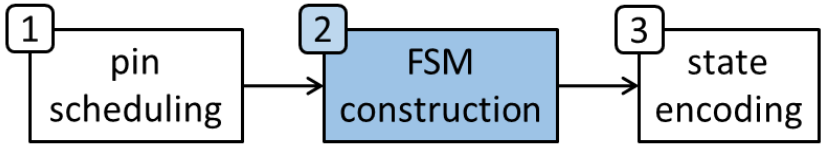


- After pin scheduling, the 3-adder becomes virtual iterative.

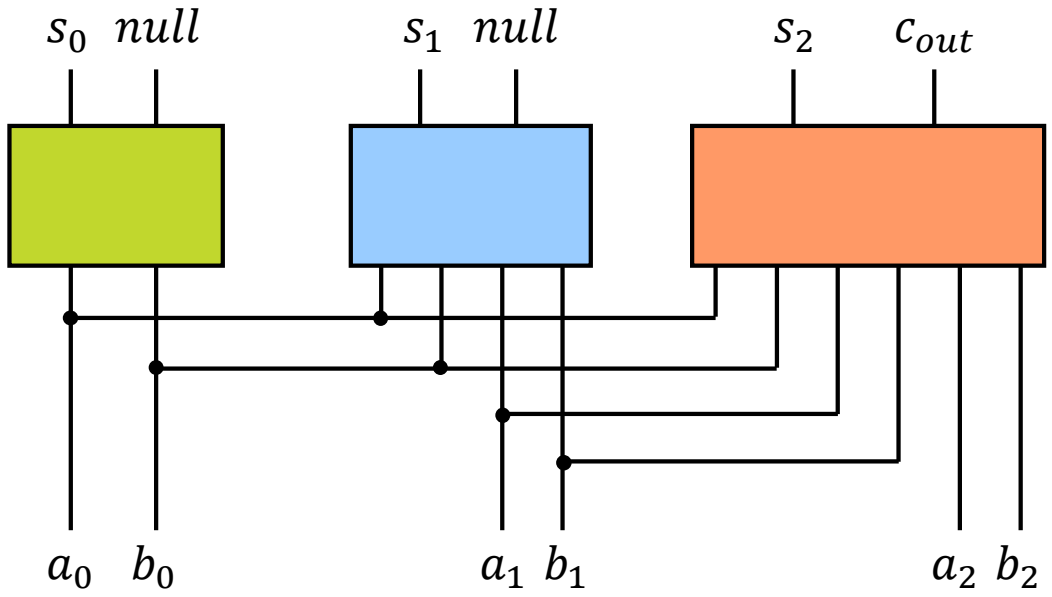


iterative form
(time-frame expanded form)

Functional Method



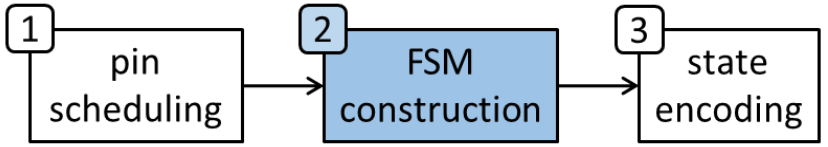
- FSM construction via time-frame folding [4]:
  - State identification via functional decomposition
  - Transition construction
  - FSM minimization



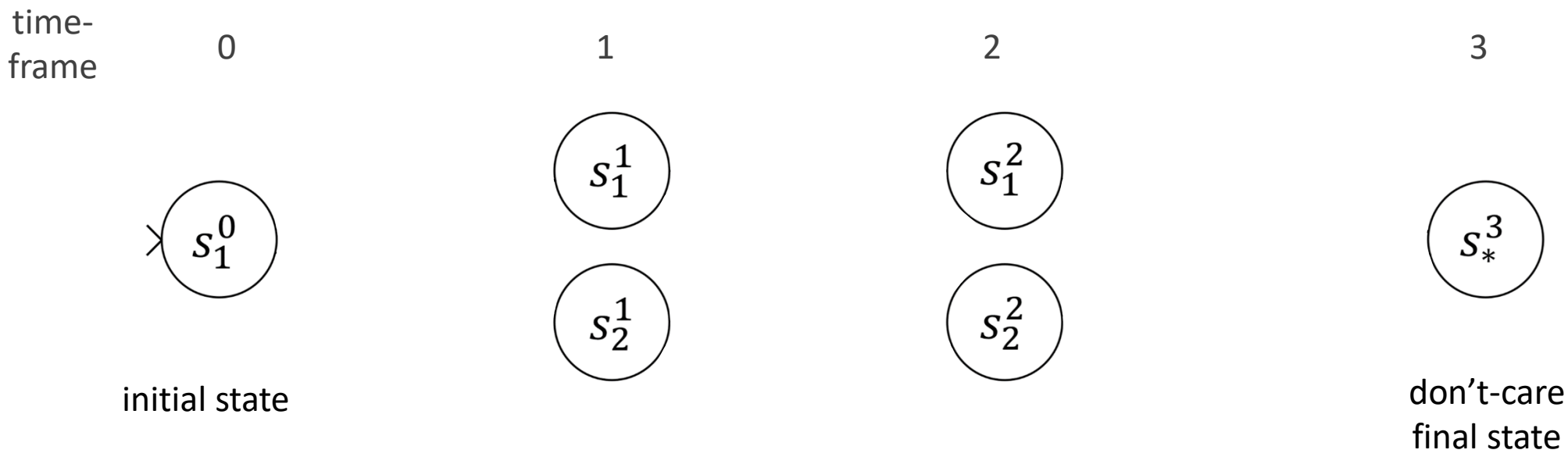
iterative form
(time-frame expanded form)

[4] Chien et al., 2019.

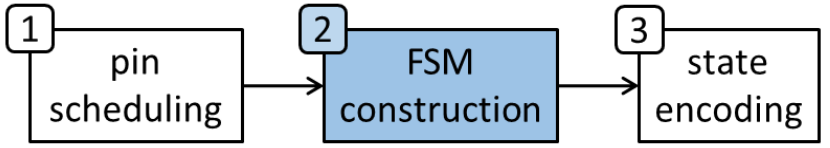
Functional Method



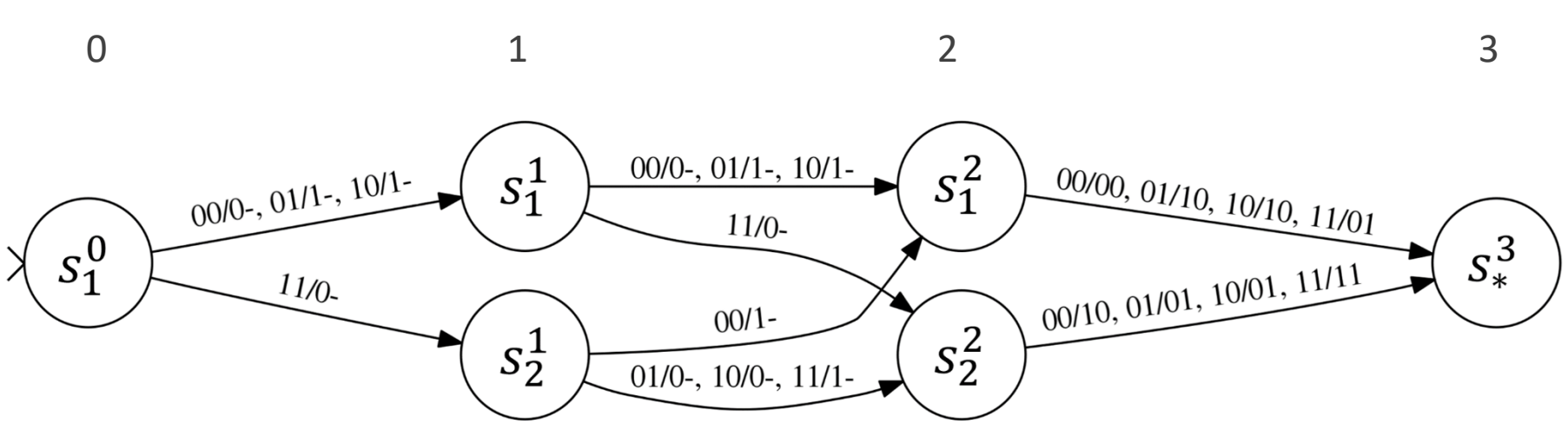
- State identification



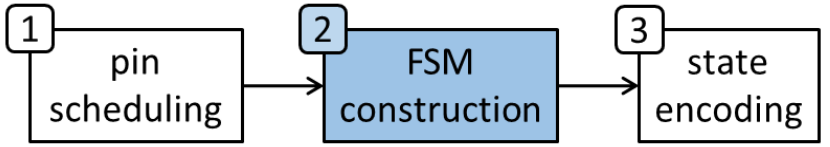
Functional Method



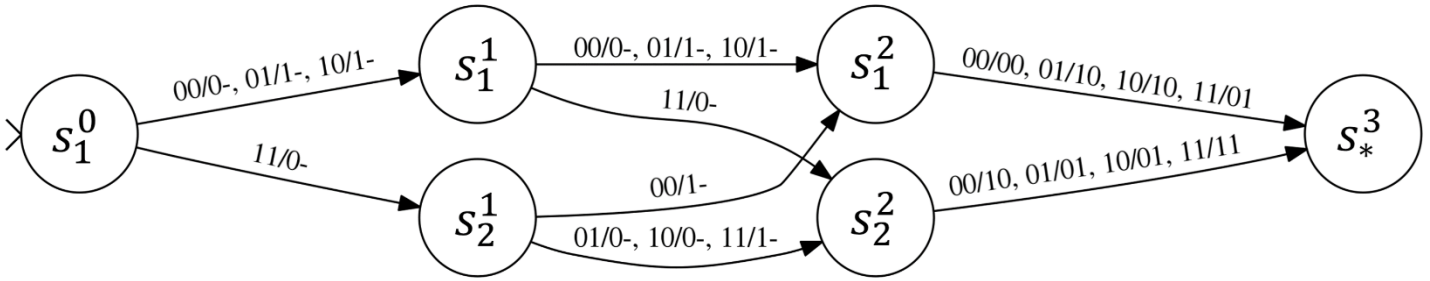
- Transition construction



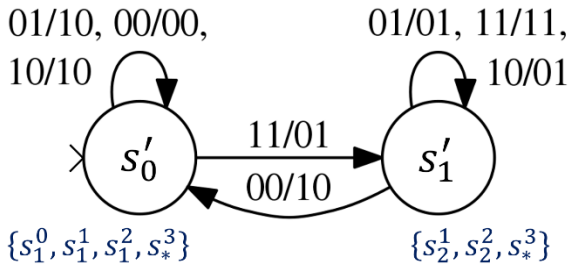
Functional Method



- FSM minimization



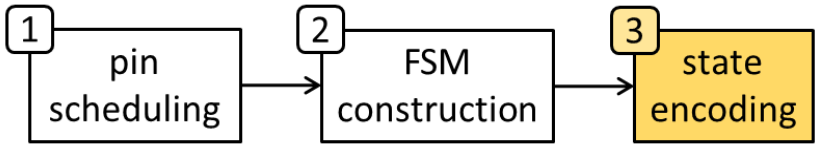
MeMin [5]



Carry-save adder

[5] Abel et al., 2015.

Functional Method



- State encoding
  Encode each state in the state set S with actual bits, 2 schemes are applied:
  - Natural Encoding with $\lceil \log(|S|) \rceil$bits
  - One-hot encoding with $|S|$ bits, each of which represents a state in S

Experiments

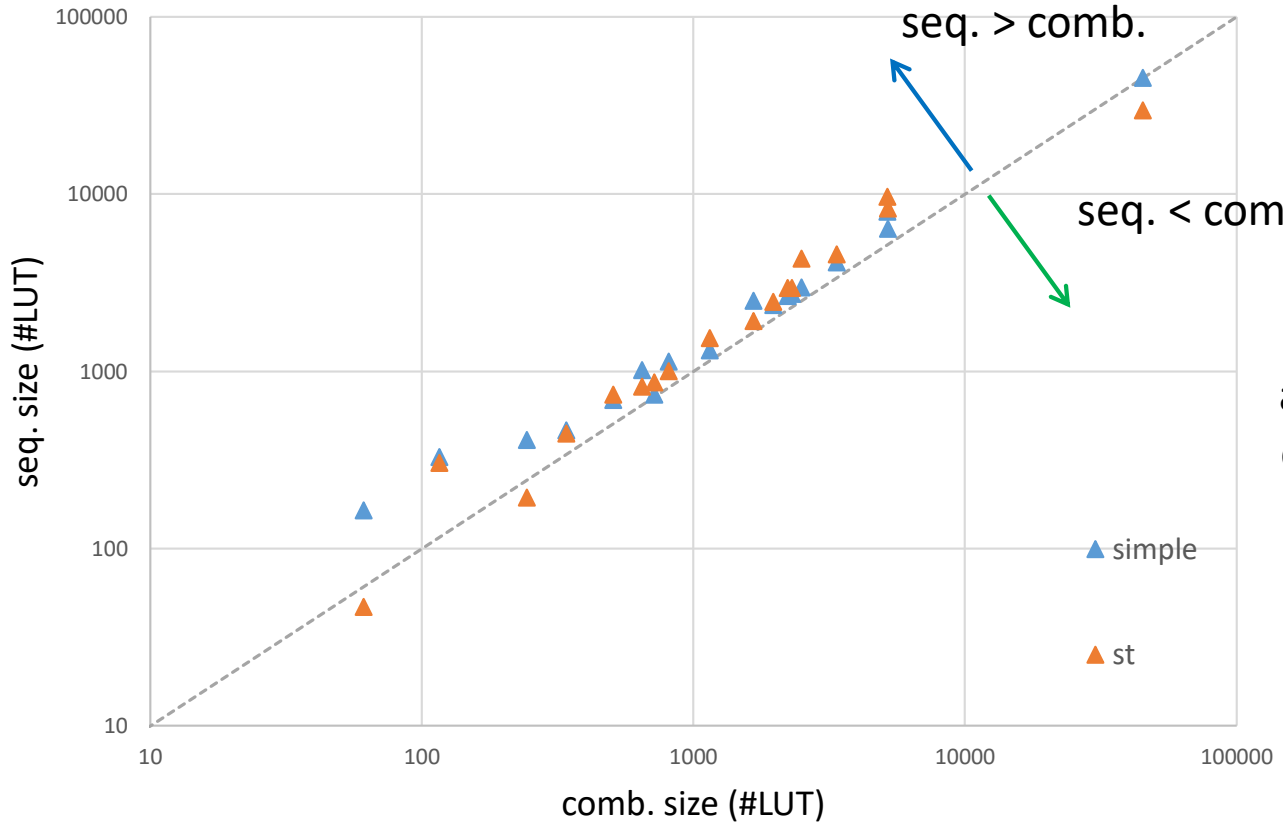
Experimental Setup

- Implemented in C++ within ABC [6] and used CUDD [7] as the BDD package.
- Environment: Intel(R) Core(TM) i7-8700 3.20GHz CPU and 32GB RAM.
- Benchmark circuits: ISCAS, ITC, MCNC(LGSynth), LEKO/LEKU, Adder, and EPFL.
- Structural method:
  - Imposed the input pin count limitation to 200.
  - 11 benchmark circuits with #PI > 200.
- Functional method
  - 11 benchmark circuits, each folded by 4, 8 and 16 time-frames.
  - 300s timeout limit on FSM construction and minimization, individually.

[6] Brayton et al., 2010. [7] Somenzi et al., 2005.

Structural Method - Results

- Impose the input pin count limitation to 200.



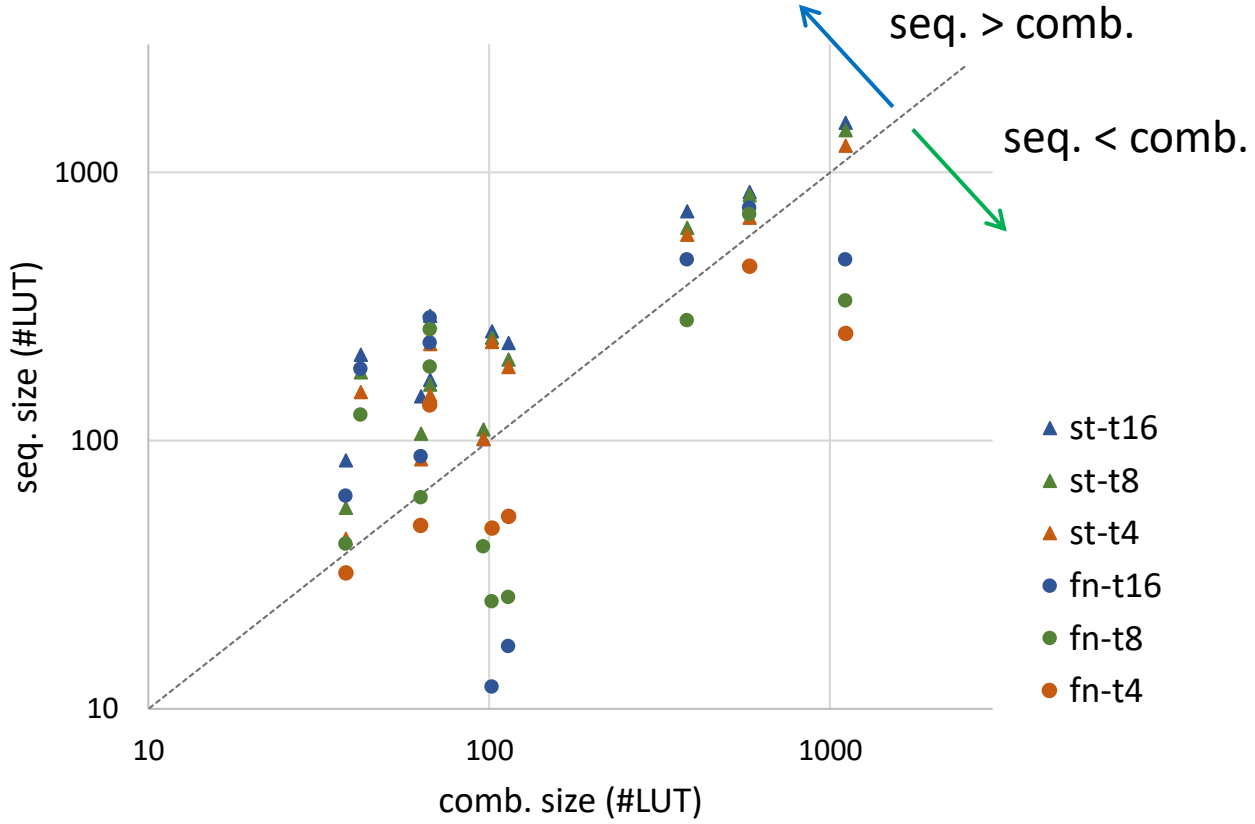
avg. #LUT overhead

46.59%

34.84%

Functional Method - Results

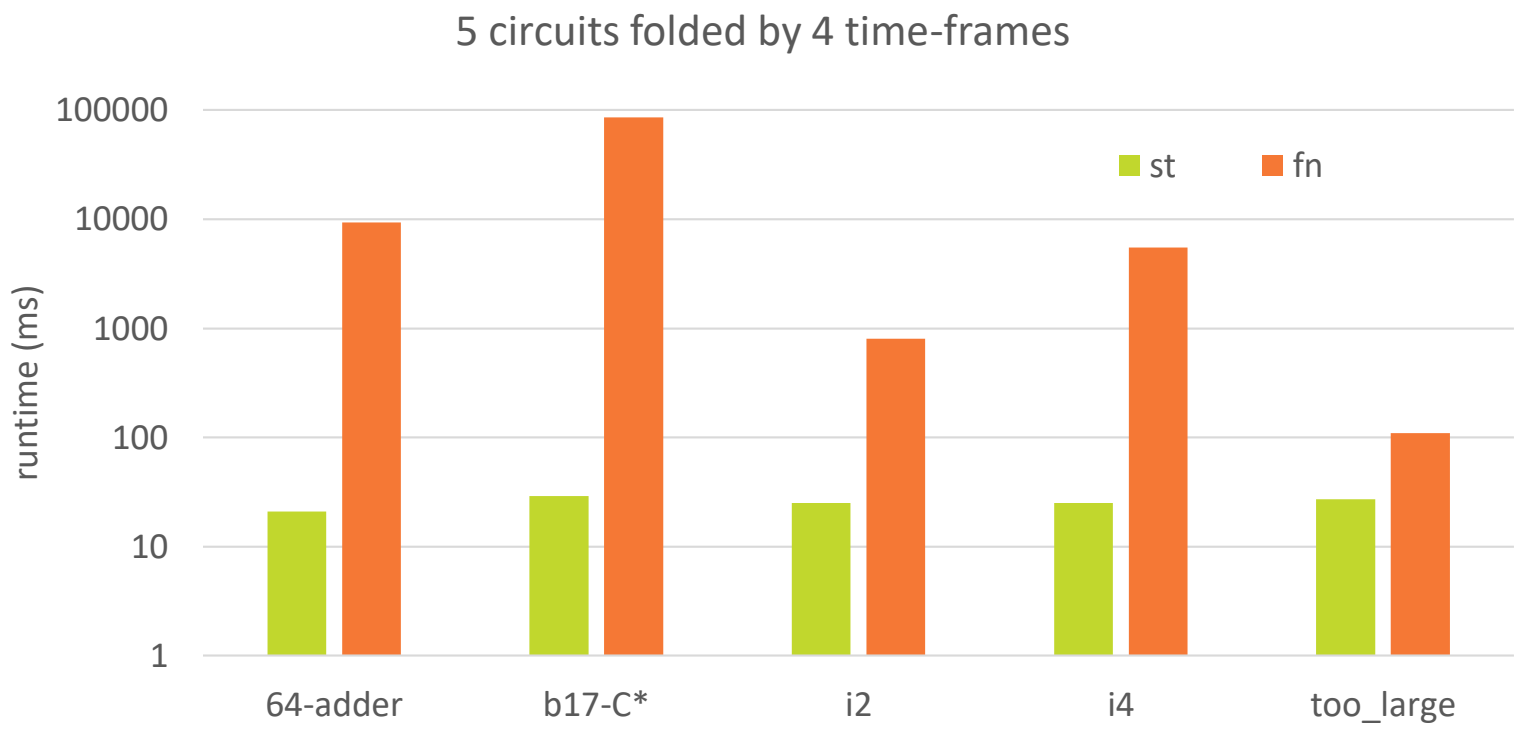
- 29 out of 33 cases done within time limit.



The functional method achieved
44.93% #LUT reduction and
64.93% #FF reduction
over the structural method

Runtime Comparison

- 4 out of 33 cases exceeded time limit when folded by the functional method, while all cases done within 1s by the structural method.



Conclusions & Future Work

Conclusions & Future Work

- We have formulated a circuit folding approach to time multiplexing on FPGAs. The structural and functional methods have been proposed and implemented to demonstrate their potentials to alleviate the I/O-pin bottleneck of FPGAs.
- Experiments suggested that

Structural Method	**Functional Method**
• fast and efficient	• high computational cost
• higher circuit complexity	• less FF and LUT usage

- Future work: combines the 2 methods to achieve both scalability and optimality.

Questions

Feel free to ask any questions.