Fast Path-Based Timing Analysis for CPPR (special session paper)

Tsung-Wei Huang, Pei-Ci Wu, and Martin D. F. Wong
Department of Electrical and Computer Engineering (ECE)
University of Illinois at Urbana-Champaign (UIUC), IL, USA

2014 IEEE/ACM International Conference on Computer-Aided Design

ECE ILLINOIS

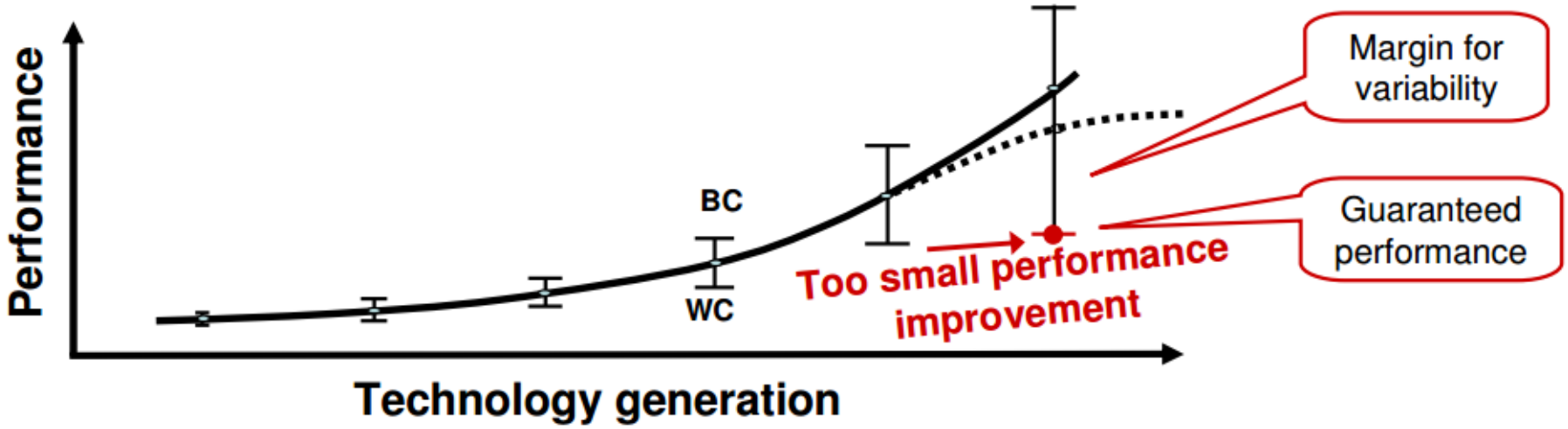
ILLINOIS

Outline

- Recap
  - Static timing analysis (STA) and common pessimism removal
  - Importance of clock network pessimism removal
  - 2014 TAU timing analysis contest
- Algorithm
  - Lookup table for pessimism retrieval
  - Path ranking algorithm
- Experimental result
  - Comparison with top-ranked timers
- Conclusion

Static Timing Analysis (STA)

- Static Timing analysis
  - Verify the expected timing characteristics of integrated circuits
  - Keep track of path slacks and identify the critical path with negative slack
- Increasing significance of variance
  - On-chip variation such as temperature change and voltage drop
  - Perform **dual-mode (min-max)** conservative or pessimistic analysis
  - Degrades the quality of signoff timing report

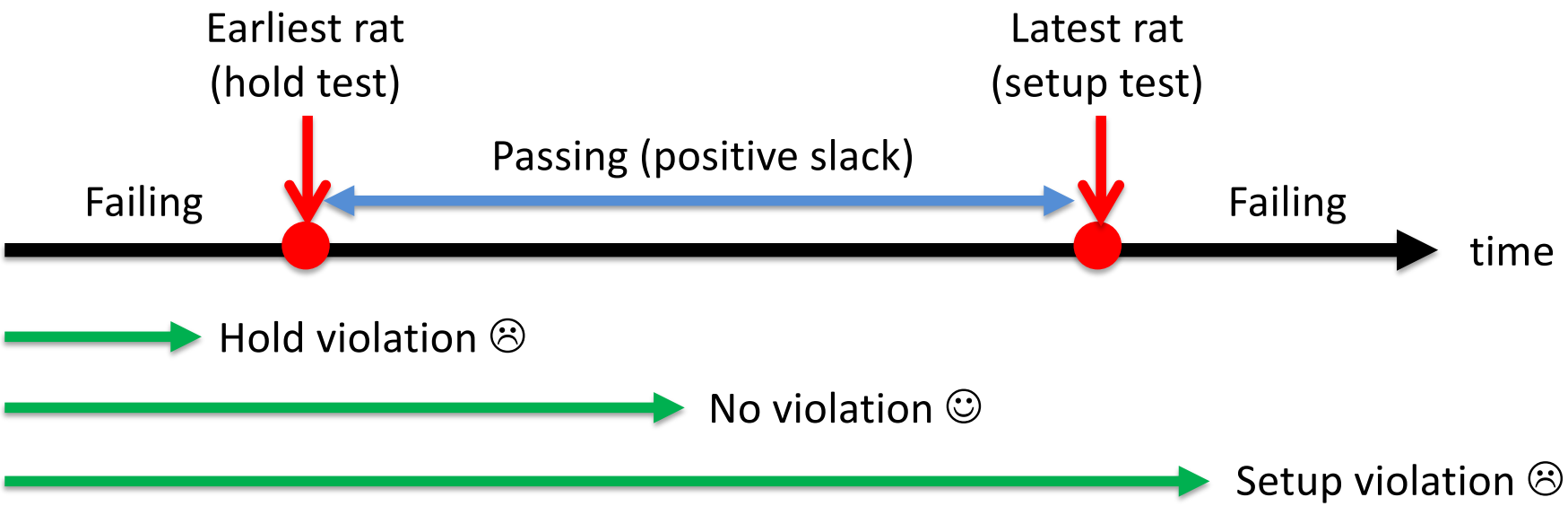


Timing Test and Verification of Setup/Hold Check

- Sequential timing test
  - Setup time check
    - "Latest" arrival time (at) v.s. "Earliest" required arrival time (rat)
  - Hold time check
    - "Earliest" arrival time (at) v.s. "Latest" required arrival time (rat)

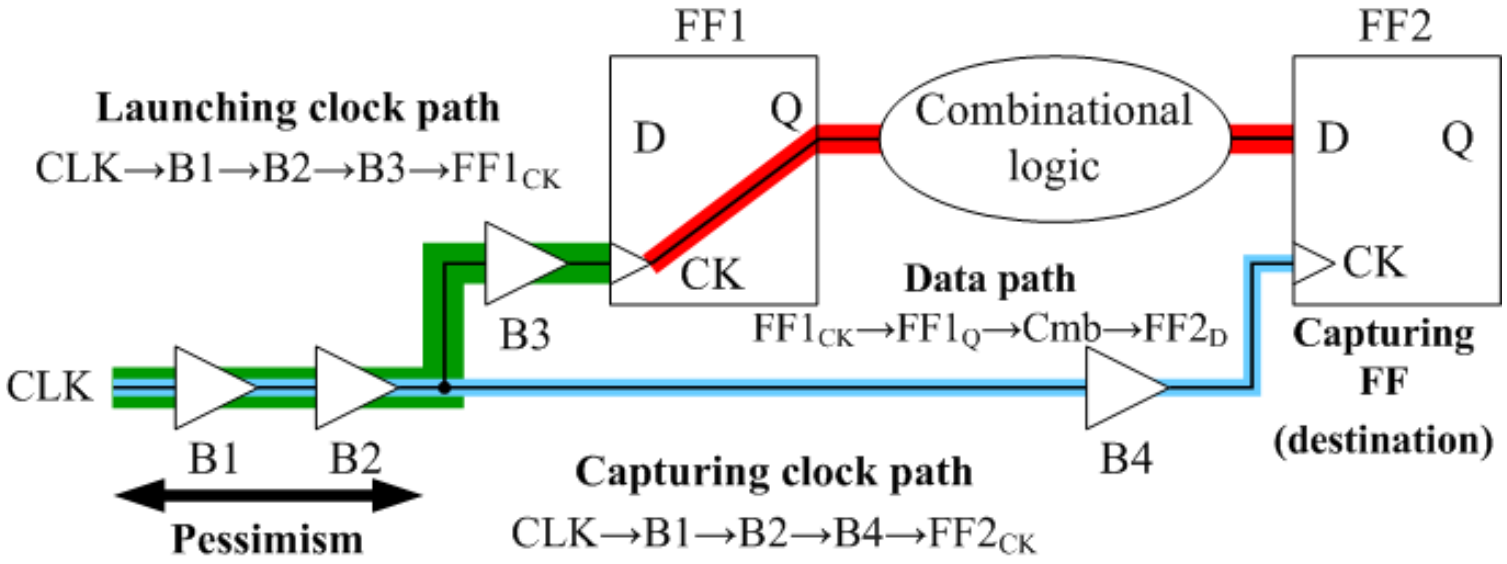
$$rat_t^{early} = at_o^{late} + T_{hold},\ rat_t^{late} = at_o^{early} + T_{clk} - T_{setup} \quad (1)$$

$$slack_{worst}^{hold} = at_d^{early} - rat_t^{early},\ slack_{worst}^{setup} = rat_t^{late} - at_d^{late} \quad (2)$$



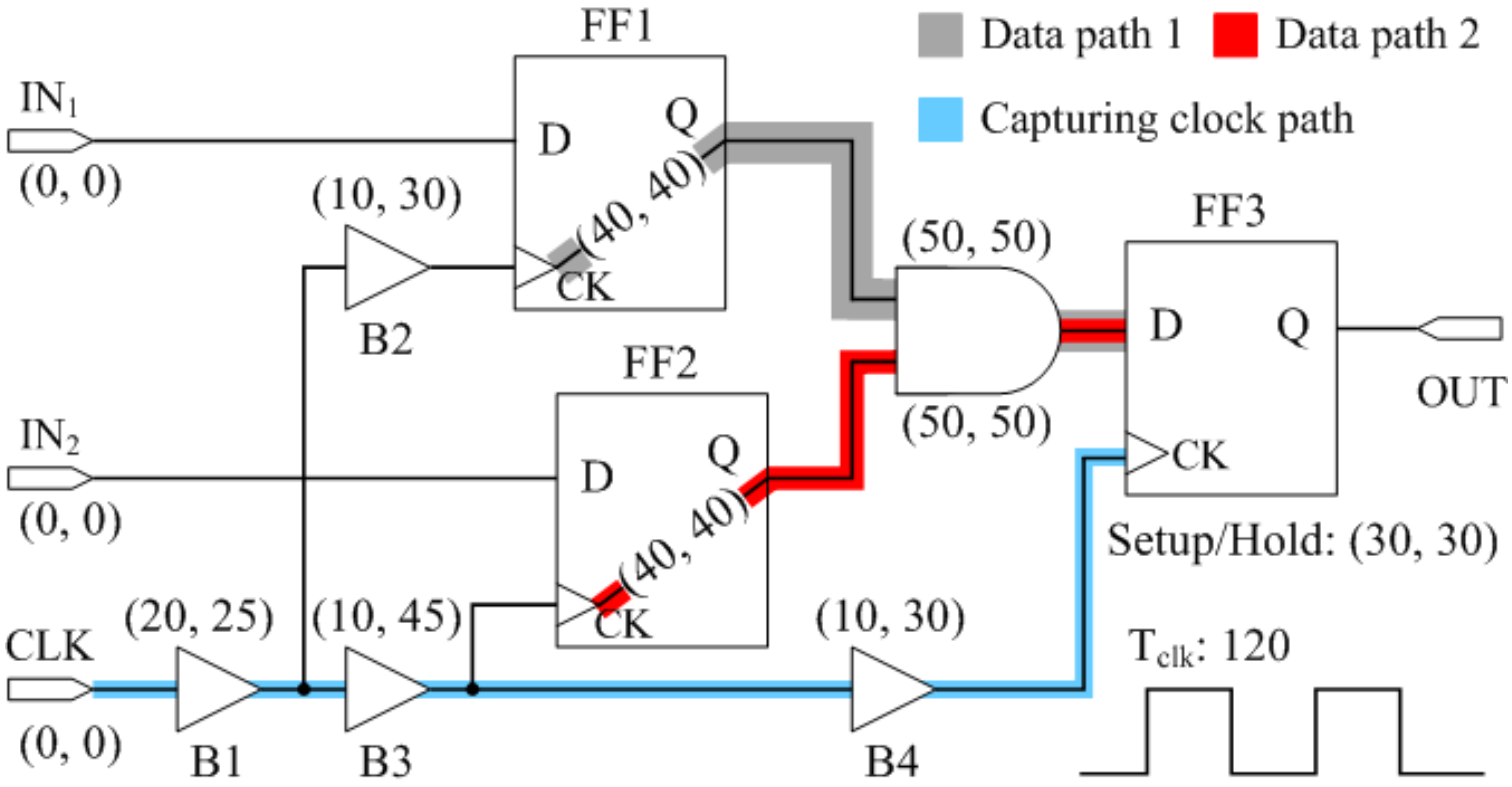
Clock Network (Common Path) Pessimism

- Common path pessimism (CPP)
  - Simultaneous min-max variation along the common clock paths
    - Impossible in reality
  - Unnecessary pessimism is included into the path slack
    - Test marking failing might be passing in actuality



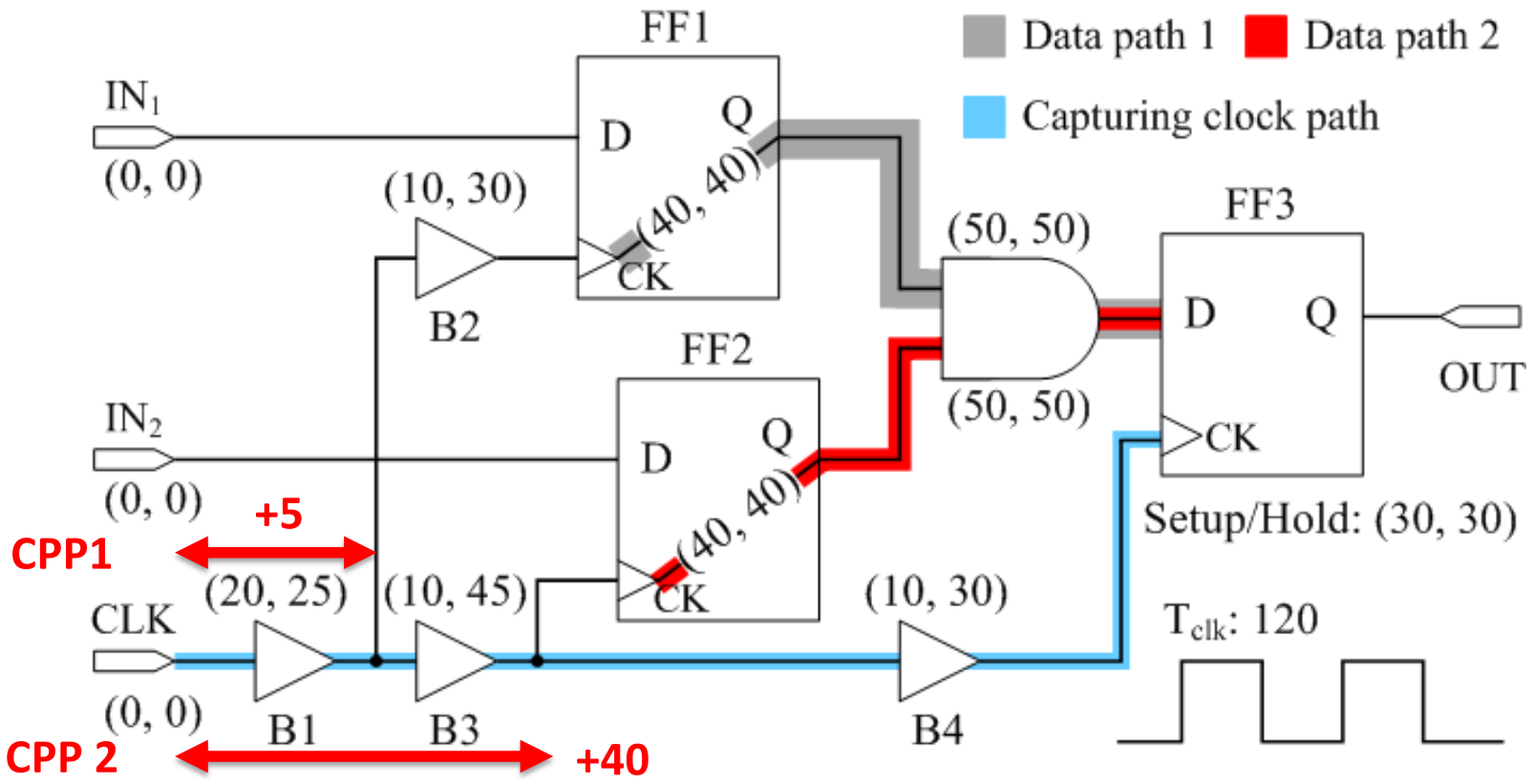
Example – Data Path Slack with CPPR Off

- Pre common-path-pessimism-removal (CPPR) slack
  - Data path 1: ((120+(20+10+10))-30) – (25+30+40+50) = -15 (critical)
  - Data path 2: ((120+(20+10+10))-30) –(25+45+40+50) = -30 (critical)

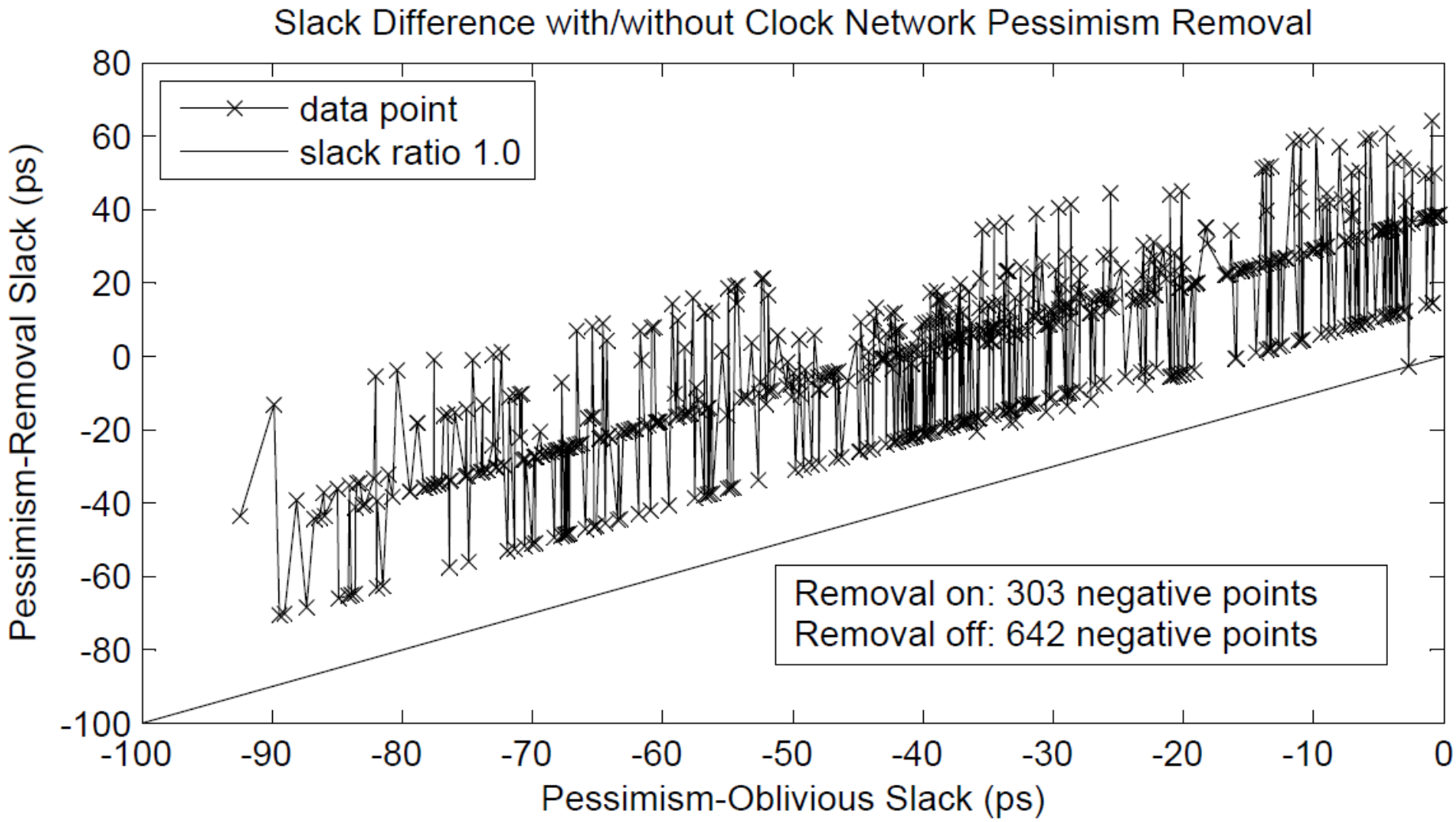


Example – Data Path Slack with CPPR On

- Post common-path-pessimism-removal (CPPR) slack
  - Data path 1: ((120+(20+10+10))-30) – (25+30+40+50)**+5** = **-10** (critical)
  - Data path 2: ((120+(20+10+10))-30) –(25+45+40+50)**+40** = **10**



Impact of Clock Network Pessimism – (I)



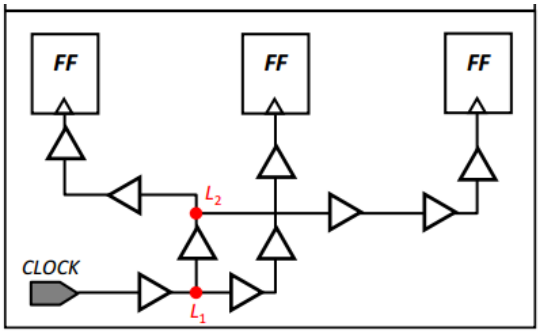
Importance of Clock Network Pessimism Removal

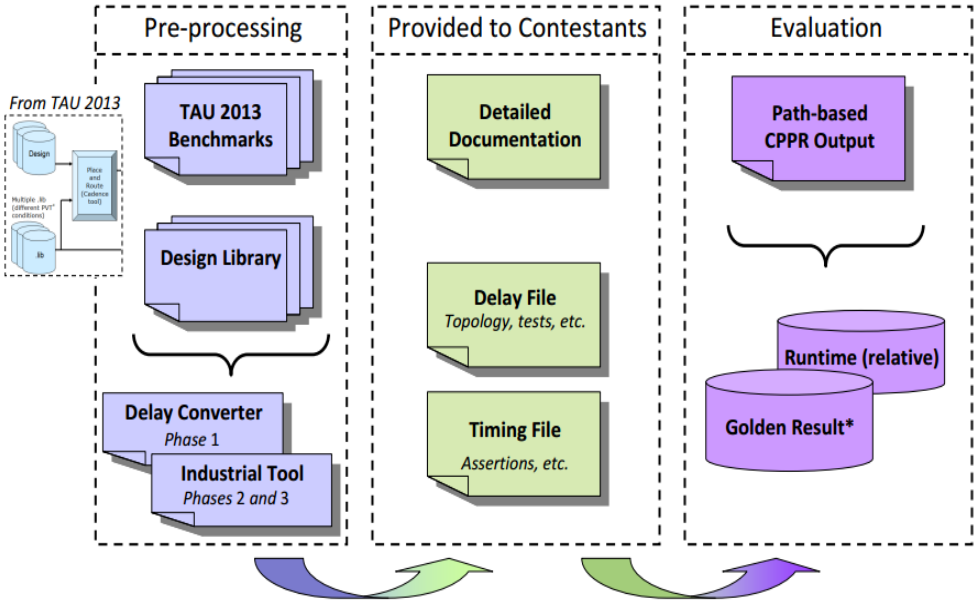
- Over-pessimistic timing report
  - Report failure paths but in actuality passing
  - Mislead designer into an inaccurate timing result
  - Decrease the productivity and legality of turnaround
  - Most critical pre-CPPR path(s) could be positive post-CPPR path(s)
  - Mislead CAD tools and result in wastage of optimization efforts
  - Leaving performance on Table

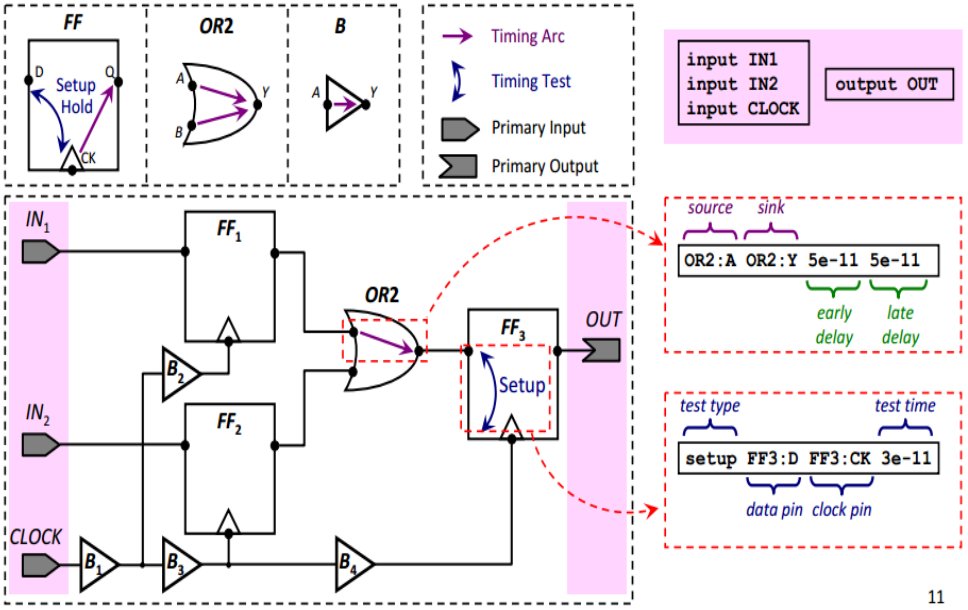
- Increasing significance in deep sub-micro era
  - Industry people seek novel ideas for CPPR algorithm
  - 2014 TAU CAD Contest on CPPR
    - Worldwide teams participated in the contest
    - **1st place winner – UI-Timer**

Problem Formulation

- Input file
  - Delay file for circuit topology and tests
  - Timing file for assertion and clock properties
- Goal
  - Identify the top *k* critical paths with CPPR (i.e., post-CPPR critical paths)





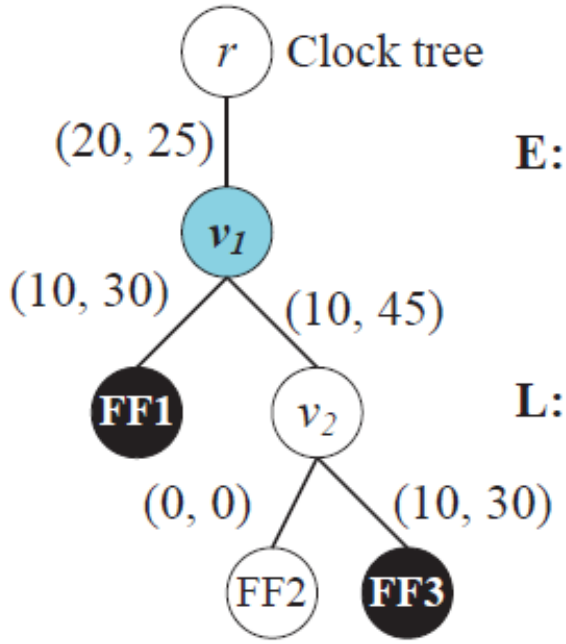


Algorithm - Overview

- Step 1: look-up-table preprocessing
  - Tabulate the common path for quick pessimism lookup
- Step 2: pessimism-free graph formulation
  - Facilitate the search on true post-CPPR slack
- Step 3: path extraction
  - Search the top *k* critical paths in the pessimism-free graph

Algorithm – Step 1: Lookup Table Preprocessing

- Pessimism incurs on a clock re-convergence node
  - Equivalently finding the lowest common ancestor (LCA) in the clock tree
  - Range minimum query with dynamic programming
  - O(nlogn) preprocessing and O(1) per query



LCA(FF1, FF3) = E[3] = v_1

E:

0	1	2	3	4	5	6	7	8	9	10
r	v_1	FF1	v_1	v_2	FF2	v_2	FF3	v_2	v_1	r

MinL(2, 7) = 3

L:

0	1	2	3	4	5	6	7	8	9	10
0	1	2	1	2	3	2	3	2	1	0

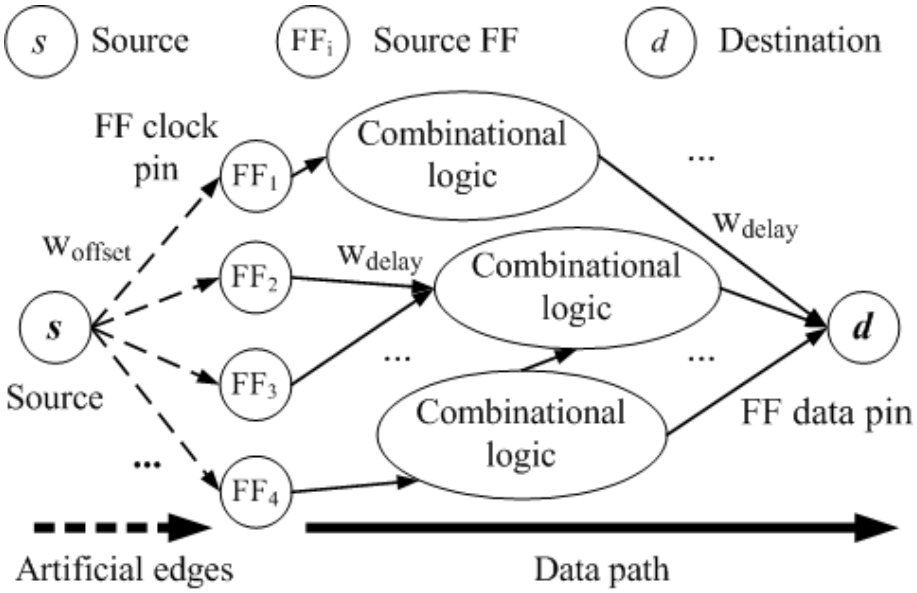
H[FF1] = 2
H[FF3] = 7

H:

r	v_1	FF1	v_2	FF2	FF3
0	1	2	4	5	7

Algorithm – Step 2: Pessimism-Free Graph Formulation

- Pessimism-free graph formulation
  - Isolate the constant part of numerical slack value
    - Offset weight for required arrival time and common-path pessimism
  - Facilitate the search
    - The cost of path is equivalent to post-CPPR slack



Pessimism-free graph

Constant per test

$$slack_{hold} = at_d^{early} - rat_t^{early} + CPP_d^s$$

$$slack_{setup} = rat_t^{late} - at_d^{late} + CPP_d^s$$

Constant per test

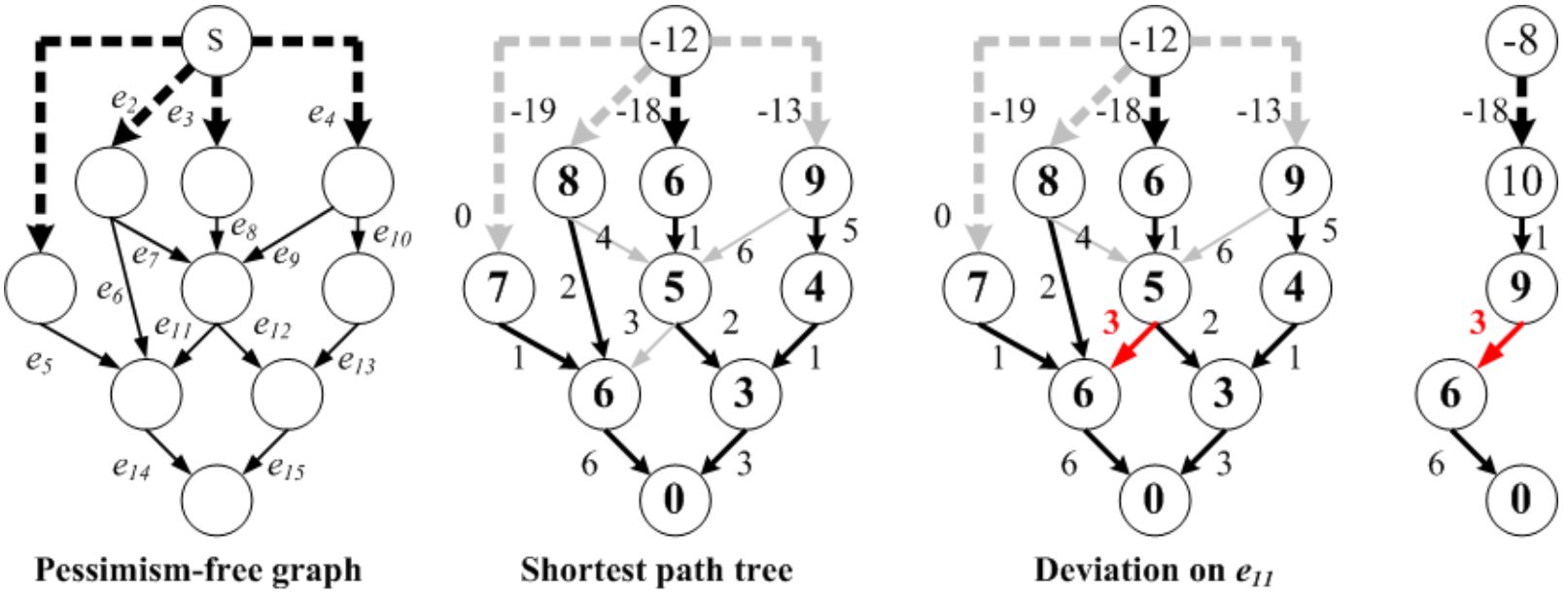
$$W_{hold}^{offset} = -rat_t^{early} + CPP_d^s$$

$$W_{setup}^{offset} = rat_t^{late} + CPP_d^s$$

Algorithm – Step 3: Extraction of Top *k* Critical Paths (I)

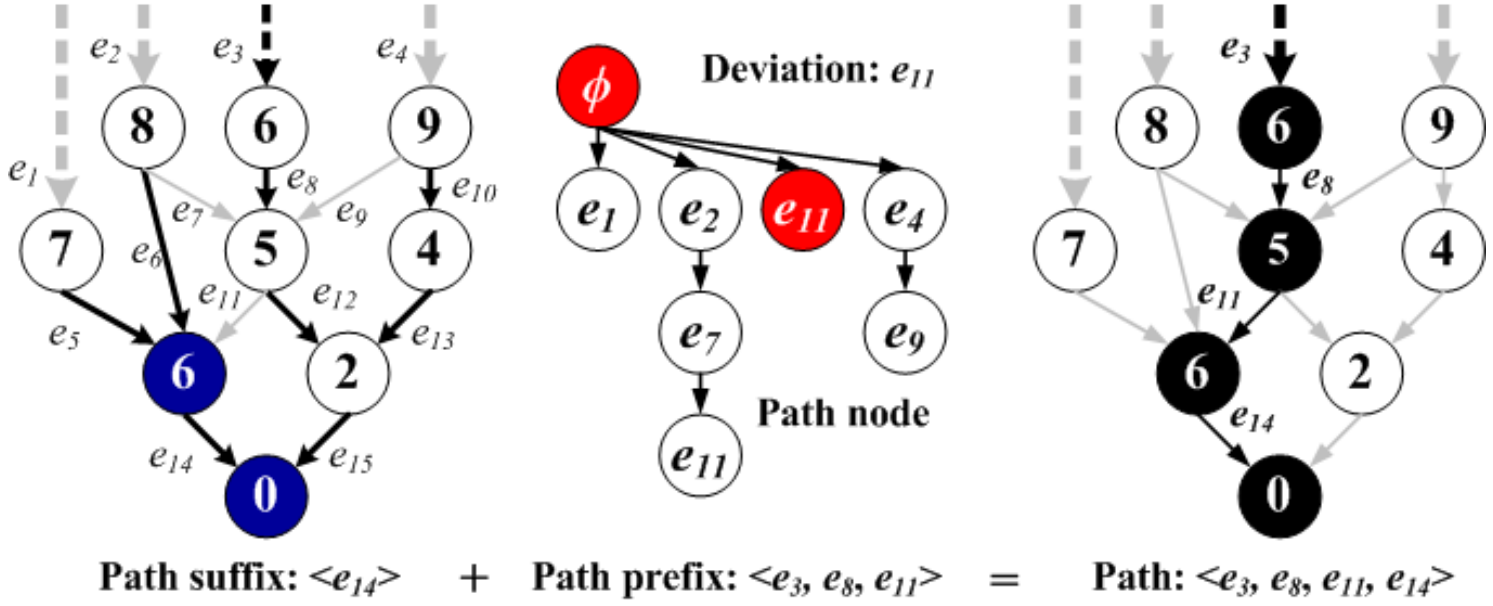
- O(N) explicit path representation
  - Path = <v_1, v_2, v_3, … v_n> or <e_1, e_2, e_3, … e_m>
- O(1) implicit path representation
  - Path = <e_i>

Referencing to the shortest path tree
=> edge as "Deviation"



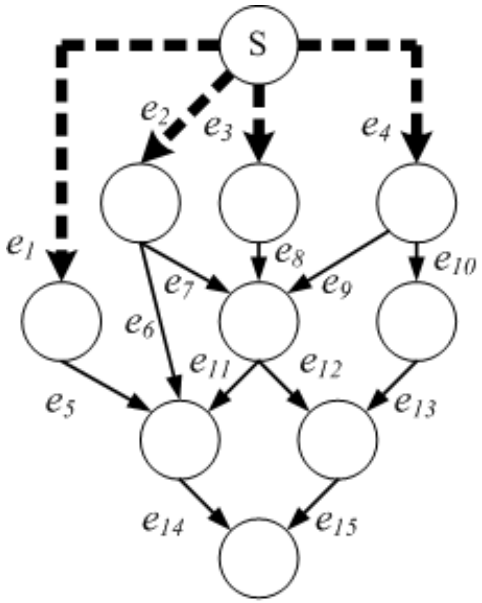
Algorithm – Step 3: Extraction of Top *k* Critical Paths (II)

- Suffix Tree
  - Shortest path tree rooted at destination node (static)
- Prefix Tree
  - Tree order of non-suffix-tree nodes (path deviation)

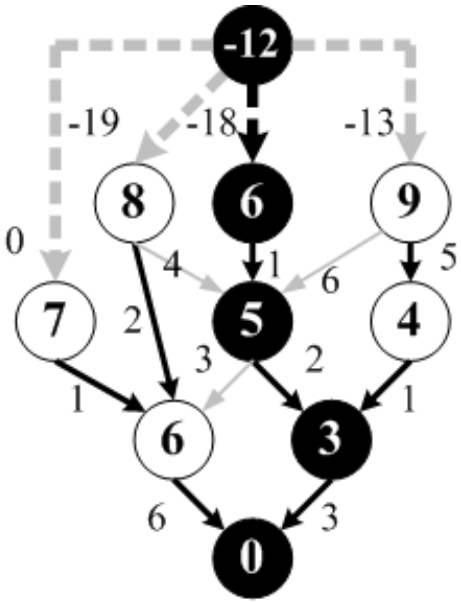


Algorithm – Step 3: Extraction of Top *k* Critical Paths (III)

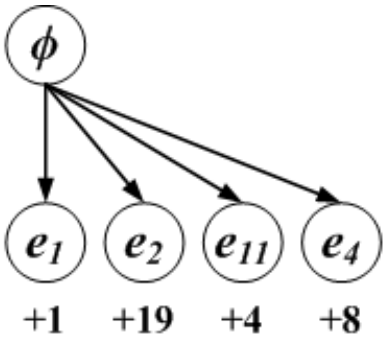
- Spur procedure
  - Growth of the prefix tree (neighboring expansion)
  - Take a path *p* and generate all other paths deviated from this path *p*
  - Mark the deviation cost



Pessimism-free graph



Shortest path tree



4 paths spurred
Shortest path slack = -12
Post-CPPR slack = {-11, 7, -8, -4}

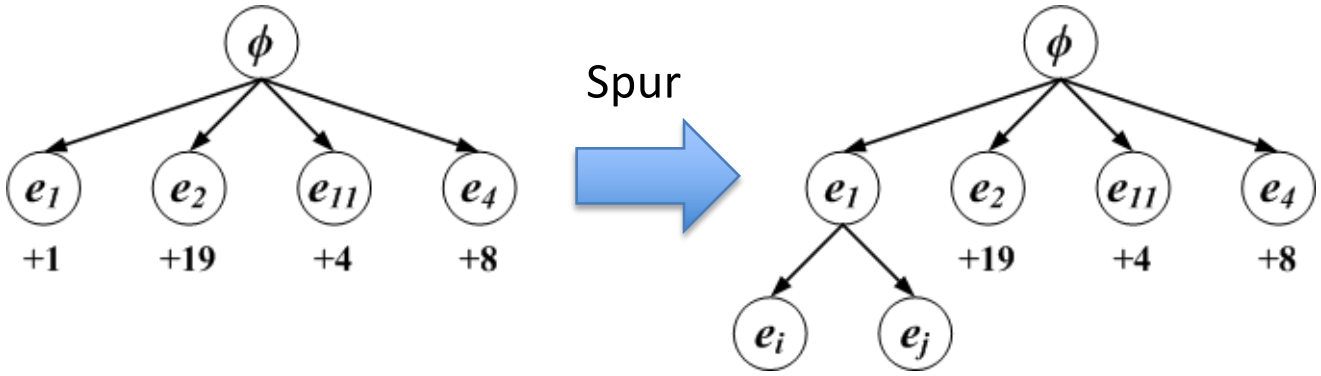
Spur along the shortest path

Algorithm – Step 3: Extraction of Top *k* Critical Paths (IV)

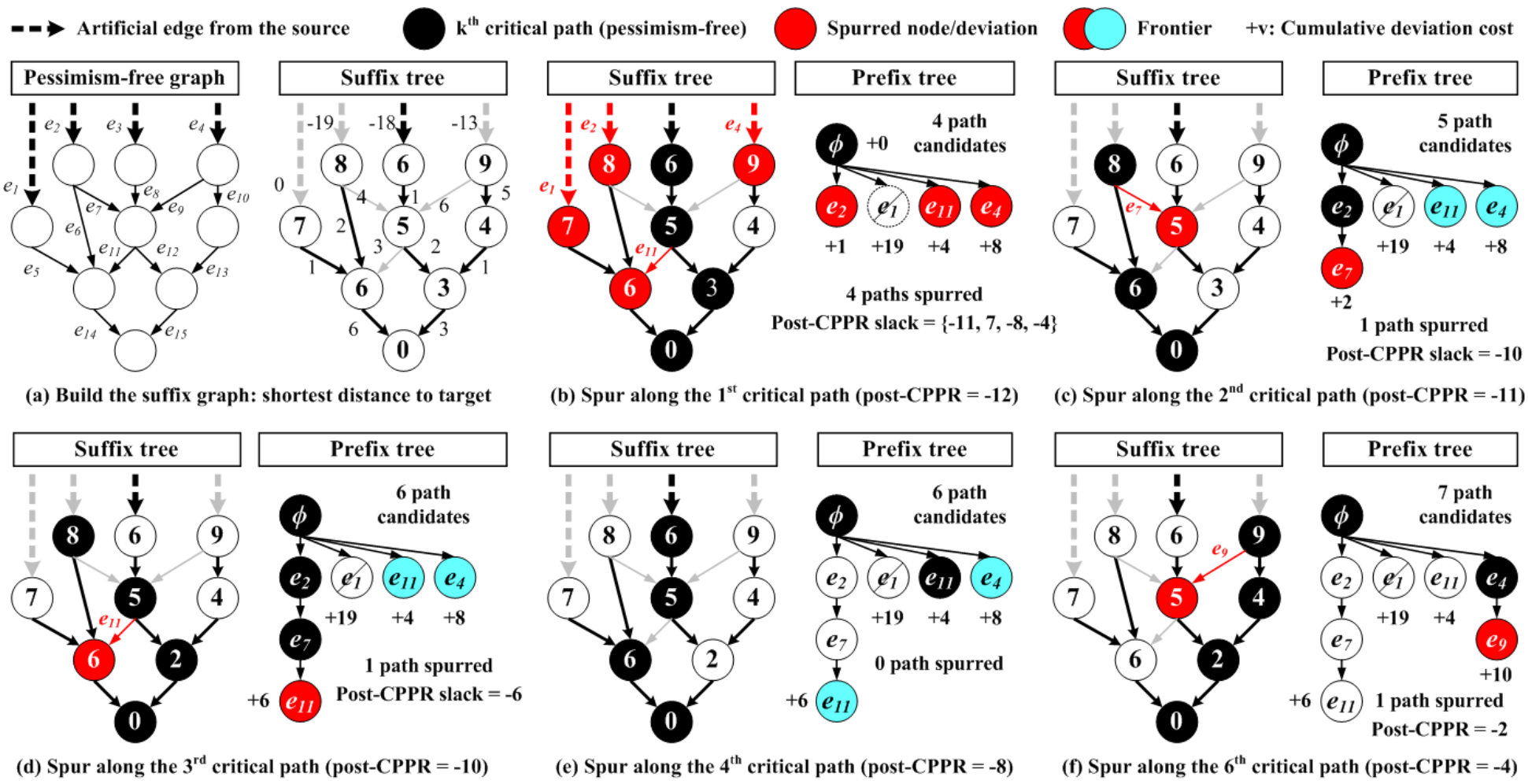
- Priority search
  1. Pick up a prefix-tree node with the minimum cost
  2. Recover the path and mark it as the current most critical path
  3. Perform the spur operation on this path
  4. Repeat until the top *k* critical paths have been found

- Optimality
  - Analogy to typical graph search

Iterative expansion from the frontier node with the least cost



Algorithm - Exemplification



Experimental Results – (I)

- Environment
  - C++ implementation
  - Ubuntu 10.4 Linux system
  - 4 Intel i7 cores and 8GB memory
- Benchmark from TAU 2014 Contest
  - Sequential circuits
  - Open core circuits
  - Combo series
- Baseline timers
  - LightSpeed (2nd place timer)
  - iTimerC (3rd place timer)

Sequential circuits

(6-42 tests)

Design	Number of:			
	PIs	POs	Segments	Tests
s27	6	1	112	6
s344	11	11	658	30
s349	11	11	682	30
s386	9	7	701	12
s400	5	6	813	42
s510	21	7	1091	12
s526	5	6	1097	42
s1196	16	14	2.4K	36
s1494	10	19	2.9K	12

Open core

(380-50K tests)

Design	Number of:			
	PIs	POs	Segments	Tests
systemcdes	132	65	13.3K	380
wb_dma	217	215	17.4K	1374
tv80	14	32	23.7K	838
systemcaes	260	129	29.6K	2.5K
mem_ctrl	115	152	45.0K	3.7K
ac97_ctrl	84	48	55.7K	9.3K
usb_funct	128	121	66.1K	4.3K
pci_bridge32	162	207	78.2K	16.4K
aes_core	260	129	86.7K	2.5K
des_perf	235	64	404.2K	19.7K
vga_lcd	89	109	525.6K	50.1K

Combo series

(8K-110K tests)

Design	Number of:			
	PIs	POs	Segments	Tests
Combo2	170	218	284.4K	29.5K
Combo3	353	215	216.2K	8.2K
Combo4	260	169	866.3K	53.5K
Combo5	432	164	2229.6K	79.0K
Combo6	486	174	3843.9K	128.2K
Combo7	459	148	3012.3K	109.6K

Experimental Results – (II)

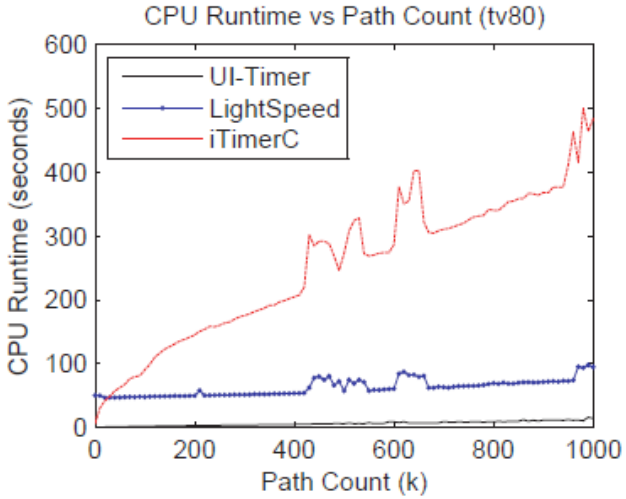
PERFORMANCE COMPARISON BETWEEN UI-TIMER AND TOP-RANKED TIMERS LIGHTSPEED AND ITIMERC FROM TAU 2014 CAD CONTEST [1].

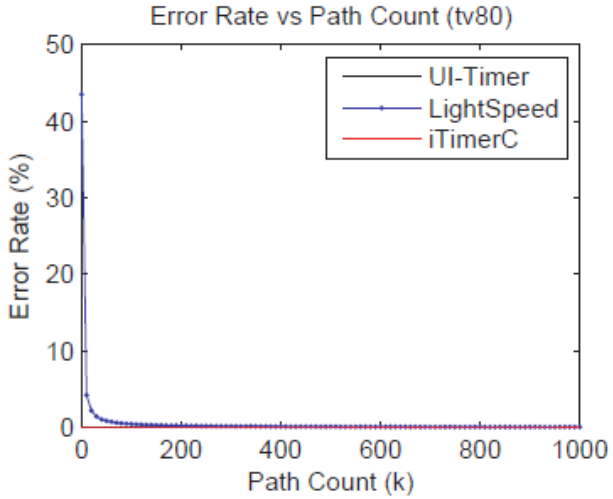
Circuit	$\|V\|$	$\|E\|$	$\|C\|$	$\|I\|$	$\|O\|$	# Tests	# Paths	LightSpeed			iTimerC		UI-Timer	
								AER	MER	CPU	AER	CPU	AER	CPU
s27	109	112	6	6	1	6	9	9.97	50.00	0.20	0	0.40	0	0.20
s344	574	658	16	11	11	30	71	0	0	0.22	0	0.53	0	0.22
s349	598	682	16	11	11	30	71	0	0	0.25	0	0.53	0	0.22
s386	570	701	7	9	7	12	27	0	0	0.20	0	0.49	0	0.20
s400	708	813	22	5	6	42	77	0	0	0.23	0	0.56	0	0.21
s510	891	1091	7	21	7	12	99	0	0	0.18	0	0.40	0	0.18
s526	933	1097	22	5	6	42	44	0	0	0.25	0	0.56	0	0.22
s1196	1928	2400	19	16	14	36	478	0	0	0.25	0	0.59	0	0.22
s1494	2334	2961	7	10	19	12	105	0	0	0.25	0	0.58	0	0.21
systemcdes	10826	13327	1967	132	65	380	41436	6.79	32.89	2.27	0	3.62	0	0.14
wb_dma	14647	17428	5218	217	215	1374	158	7.46	39.30	0.23	0	0.90	0	0.28
tv80	18080	23710	3608	14	32	838	19227963	8.20	43.49	32.38	0	23.13	0	0.23
systemcaes	23909	29673	6643	260	129	2500	13069928	6.53	29.92	33.23	0	22.44	0	0.62
mem_ctrl	36493	45090	10638	115	152	3754	62938	5.41	24.73	0.65	0	3.71	0	0.83
ac97_ctrl	49276	55712	22223	84	48	9370	148	-	-	-	0	2.95	0	1.31
usb_funct	53745	66183	17665	128	121	4392	129854	6.43	37.87	0.94	0	5.64	0	1.41
pci_bridge32	70051	78282	33474	162	207	16450	17296	5.04	25.49	2.27	0	14.49	0	4.71
aes_core	68327	86758	5289	260	129	2528	21064	6.72	31.70	0.68	0	4.46	0	0.96
des_perf	330538	404257	88751	235	64	19764	1682	4.60	11.89	3.37	0	18.37	0	19.24
vga_lcd	449651	525615	172065	89	109	50182	5281	7.94	43.21	16.78	0	119.24	0	159.15
Combo2	260636	284091	171529	170	218	29574	62938	4.70	24.07	9.19	0	49.00	0	56.12
Combo3	181831	284091	73784	353	215	8294	129854	6.71	35.14	3.39	0	20.30	0	11.35
Combo4	778638	866099	469516	260	169	53520	19227963	7.93	42.13	205.69	0	557.81	0	333.04
Combo5	2051804	2228611	1456195	432	164	79050	19227963	-	-	-	N/A	$>$ 3 hrs	0	1225.50
Combo6	3577926	3843033	2659426	486	174	128266	19227963	-	-	-	N/A	$>$ 3 hrs	0	3544.04
Combo7	2817561	3011233	2136913	459	148	109568	19227963	-	-	-	N/A	$>$ 3 hrs	0	2485.81

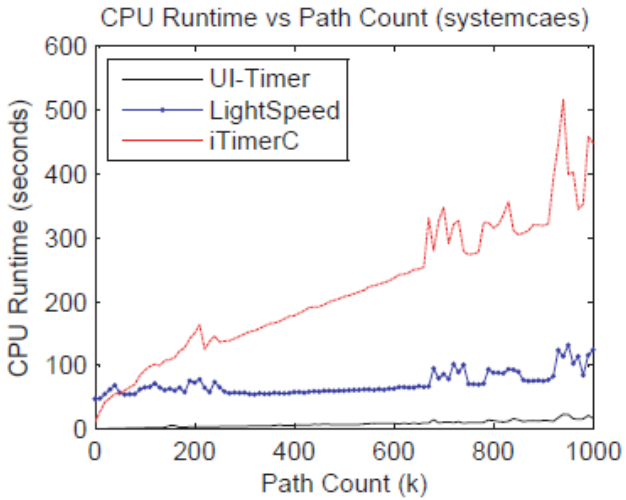
$|V|$: size of node set. $|E|$: size of edge set. $|C|$: size of clock tree. $|I|$: # of primary inputs. $|O|$: # of primary outputs. # Tests: # of setup tests and hold tests.
Paths: max # of data paths per test. AER/MER: avg/max error rate of mismatched paths (%). CPU: avg program runtime (seconds). -: unexpected program fault.

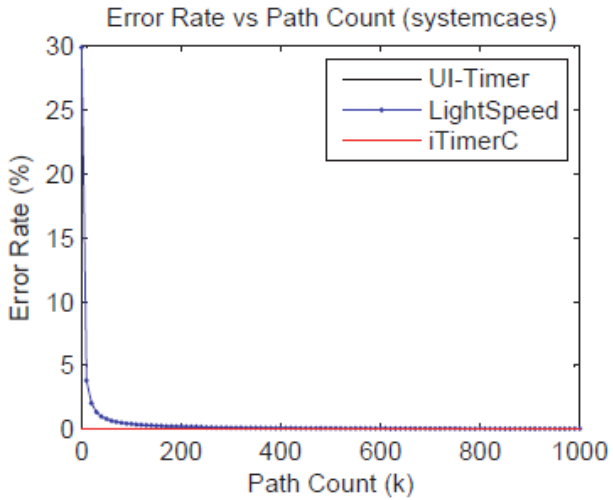
Experimental Results – (III)

- Circuits
  - tv80
  - systemcaes
- Runtime
  - 1st: UI-Timer
  - 2nd: LightSpeed
  - 3rd: iTimerC
- Accuracy
  - 1st: UI-Timer
  - 2nd: iTimerC
  - 3rd: LightSpeed









Conclusion

- A fast and exact algorithm for CPPR
  - Lookup table for clock network pessimism
  - Efficient data structure for path representation
- Evaluation
  - 1st place timer in TAU 2014 CAD contest for timing analysis
  - Handled million-scale graph within 1 hr
    - Other timers either crashed or ran over 3 hrs
- Future work
  - Incremental timing and CPPR
  - Distributed computing for path-based analysis

Acknowledgment

- Binary sharing
  - Team LightSpeed
  - Team iTimerC

- Contest organizers
  - Jin Hu, IBM Corp.
  - Debjit Sinha, IBM Corp.
  - Igor Keller, Cadence