Lecture 20: Timing Analysis – II

Tsung-Wei (TW) Huang

Department of Electrical and Computer Engineering

University of Utah, Salt Lake City, UT

In-class Presentation: 12/5

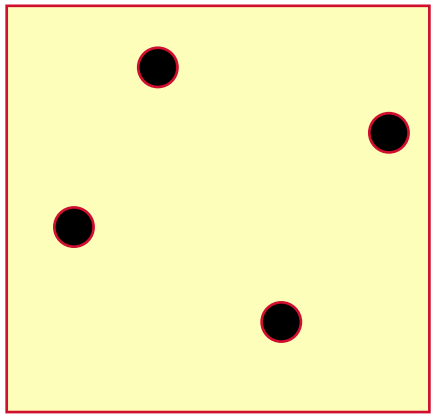
- **Routing research presentation on 12/5 (in class)**
  - Siting Liu, Yuan Pu, Peiyu Liao, Hongzhong Wu, Rui Zhang, Zhitang Chen, Wenlong Lv, Yibo Lin, Bei Yu, "FastGR : Global Routing on CPU-GPU with Heterogeneous Task Graph Scheduler," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2022
  - Chris Chu and Yiu-Chung Wong. FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. In *IEEE Transactions on Computer-Aided Design*, vol. 27, no. 1, pages 70-83, January 2008.
- Upload your pptx to https://github.com/tsung-wei-huang/ece5960-physical-design/issues/14 before presentation

In-class Presentation: 12/7

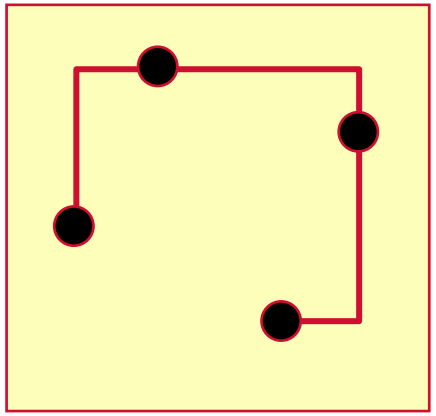
- **Routing research presentation on 12/7 (in class)**
  - Shiju Lin, Jinwei Liu, and Martin D F Wong, "GAMER: GPU-accelerated Maze Routing", *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021
  - Zizheng Guo, Feng Gu, and Yibo Lin, "GPU-Accelerated Rectilinear Steiner Tree Generation," *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022
- Upload your pptx to https://github.com/tsung-wei-huang/ece5960-physical-design/issues/15 before presentation

Programming Assignment #3: Routing

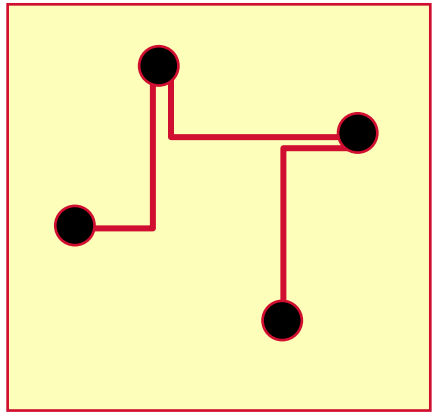
- Goal: Implement a Steiner Tree Construction Algorithm



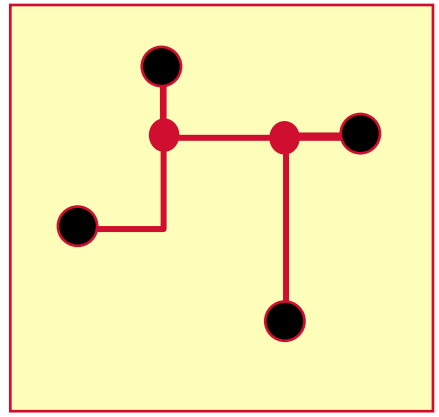
Pins to connect



Route it so we guarantee each 2-point path is shortest;



Redraw it--different orientations of 2-point paths

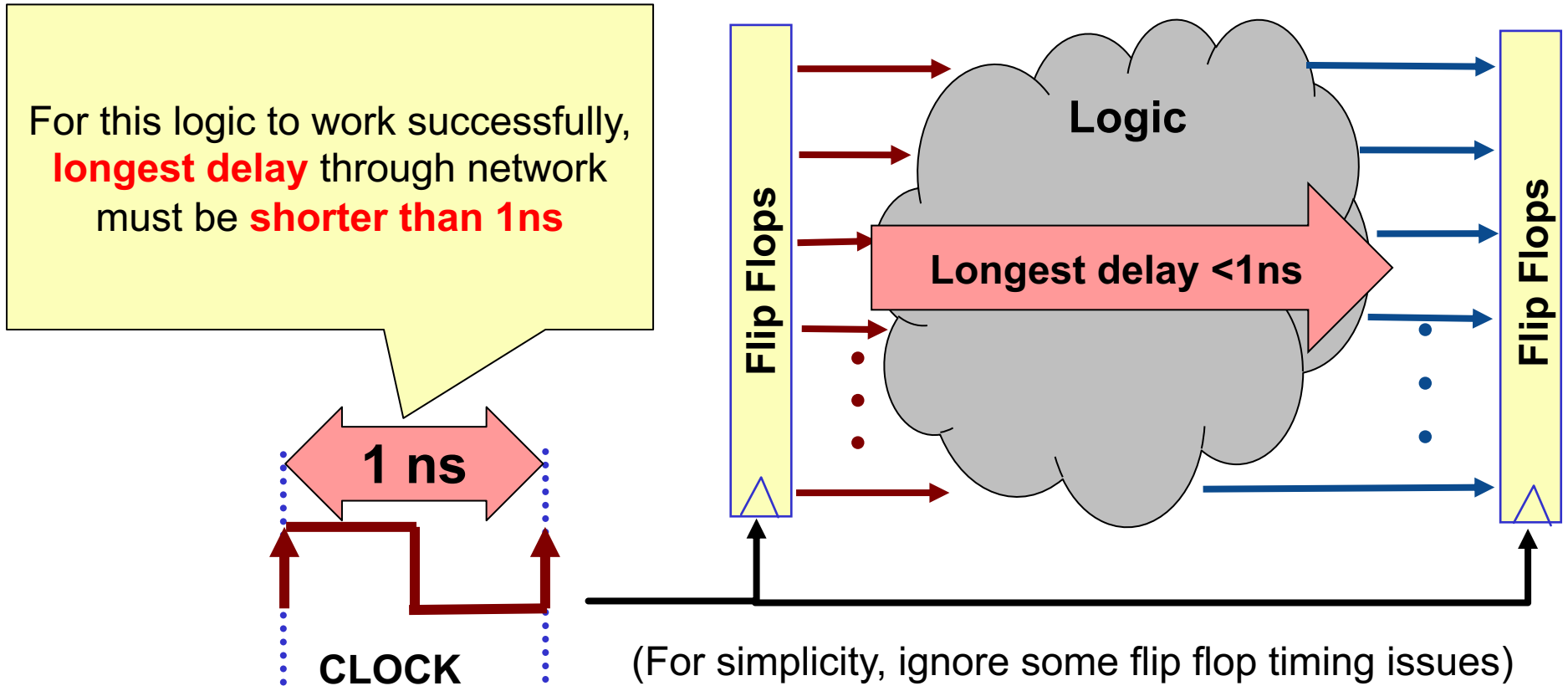


Now we can see the better (shorter) Steiner tree

Due 12/16: https://github.com/tsung-wei-huang/ece5960-physical-design/tree/main/PA3

Recap: Timing Analysis Model

- Assume we know clock cycle: e.g., 1GHz clock, cycle = 1ns

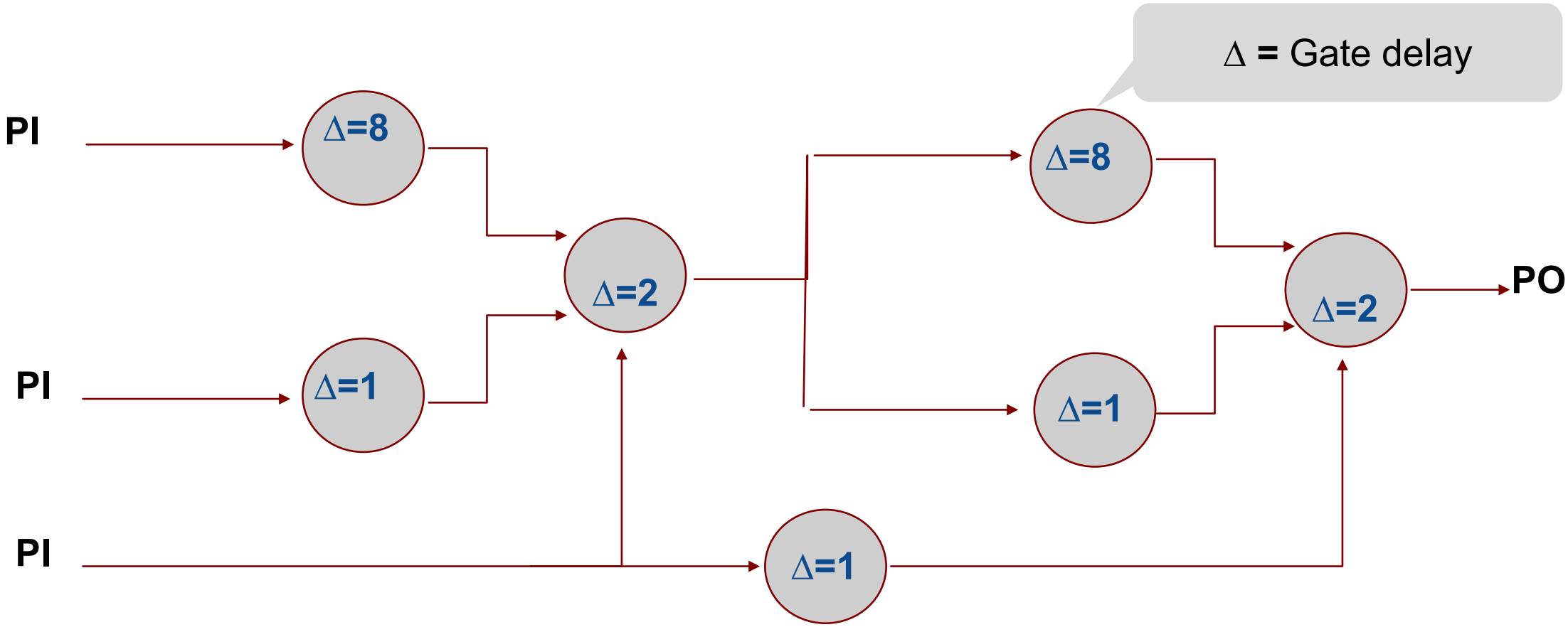


Static Timing Analysis (STA)

- **When we ignore logic, this is called Topological Analysis**
  - We only work with the graph and the delays – **don't** consider the logic
  - We can get wrong answers: what we found was called a **False Path**
- **Going forward: we ignore the logic**
  - Assume that all paths are **statically sensitizable**
    - **Means**: Can find a constant pattern of inputs to *other* PIs that makes some output sensitive to some input
- **This timing analysis is called Static Timing Analysis (STA)**
  - Consider only the best- and worst-case timing results
  - Consider no logic (otherwise called dynamic timing analysis)

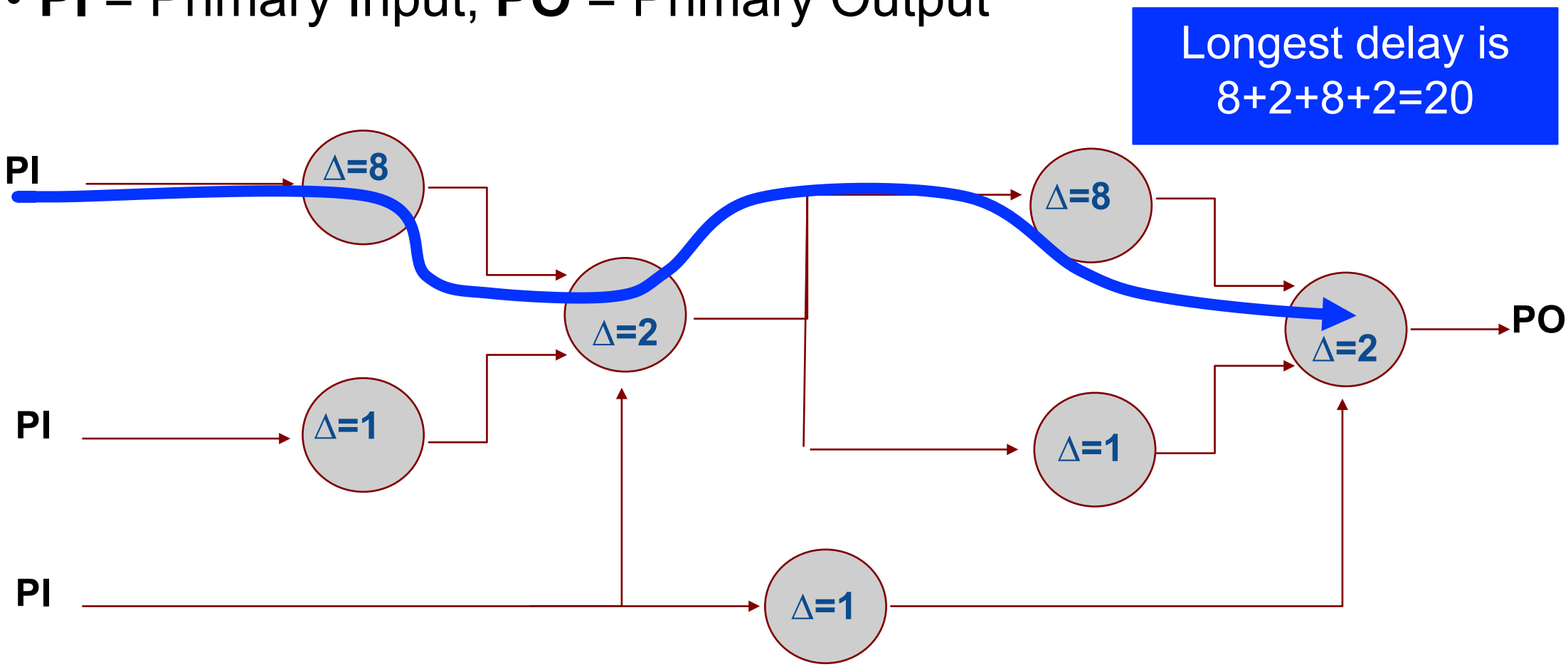
Recap: STA Example

- Consider only worst-/base-case delays



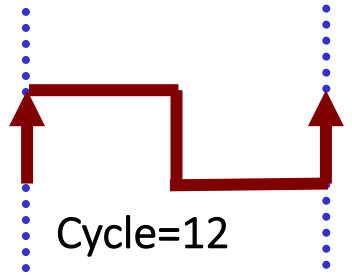
Recap: STA = Shortest Path Finding

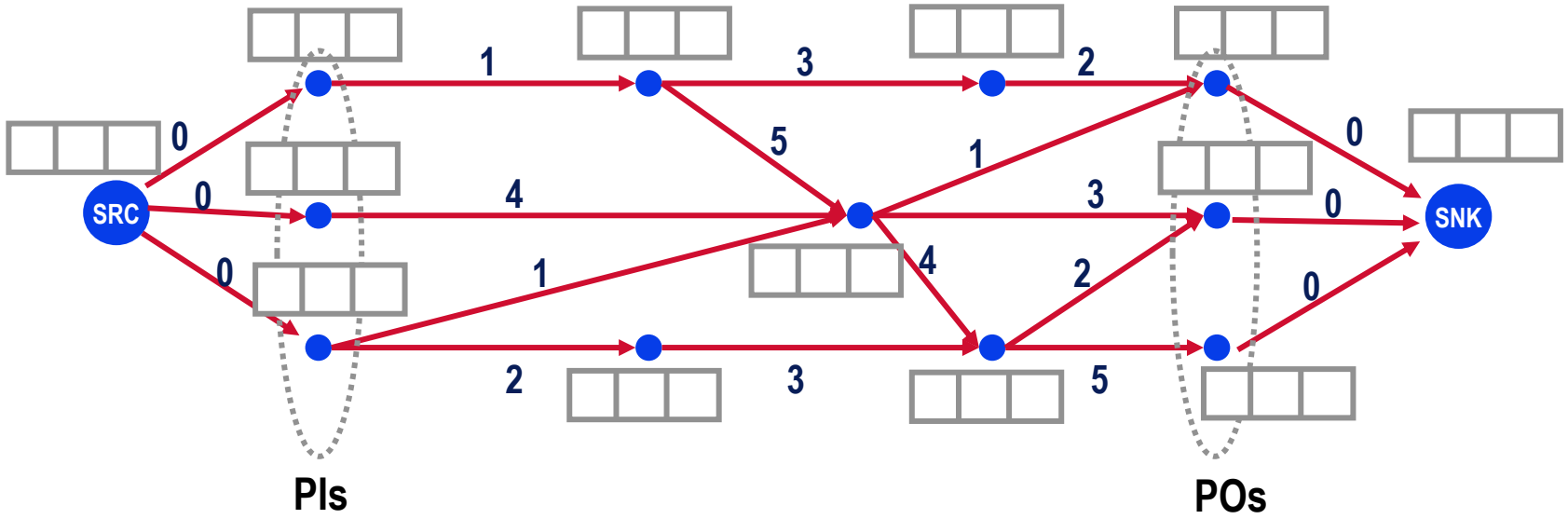
- **PI** = Primary Input, **PO** = Primary Output



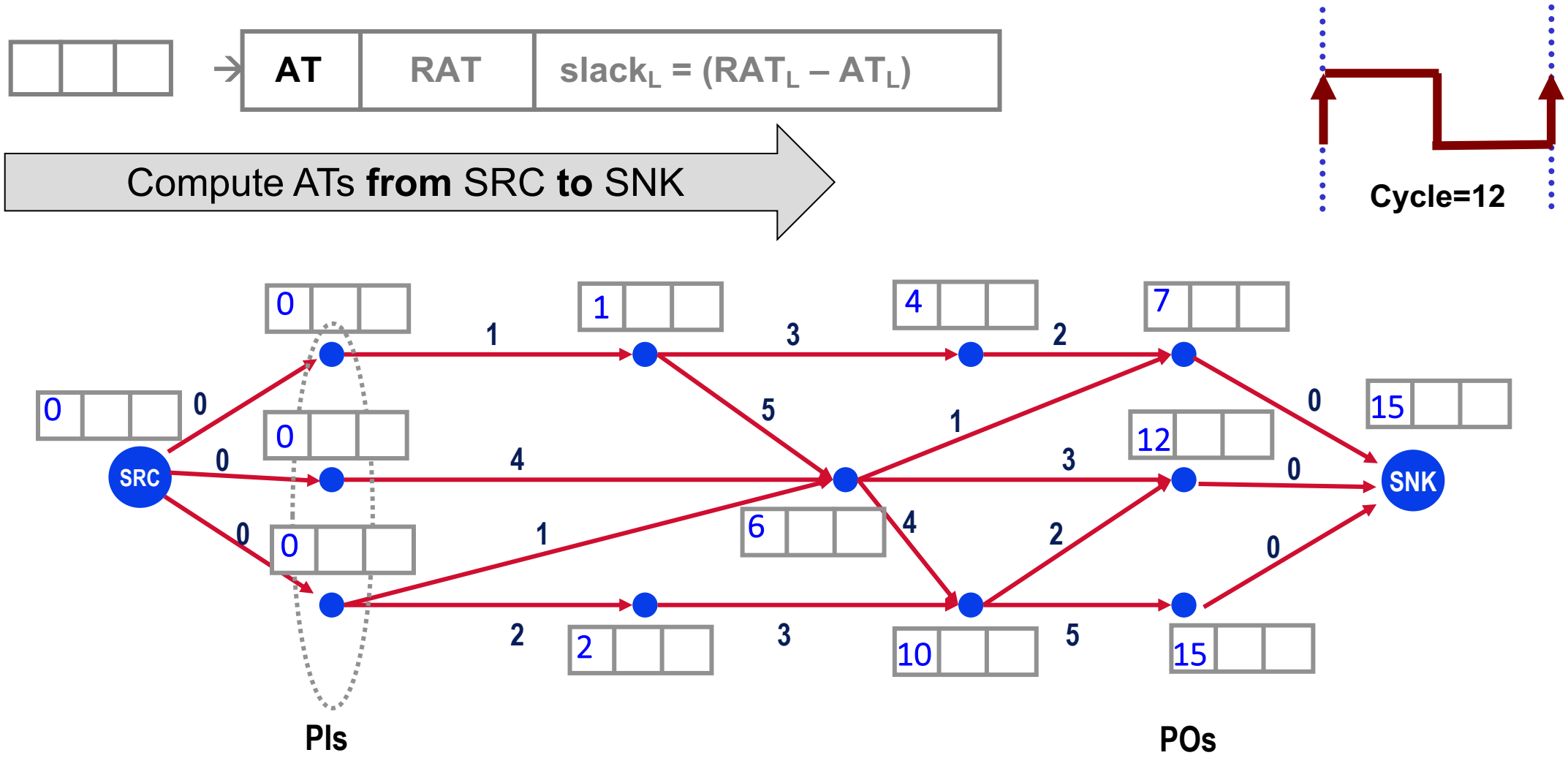
Recap: STA Terminology

- **Delays are on edges; let clock cycle be 12**
  - Compute the min/max delays “by eye” for now
  - **AT**=longest path from SRC **TO** node;
  - **RAT**=(cycle time 12) – (longest path **FROM** node to SNK)
  - **Slack** = RAT - AT

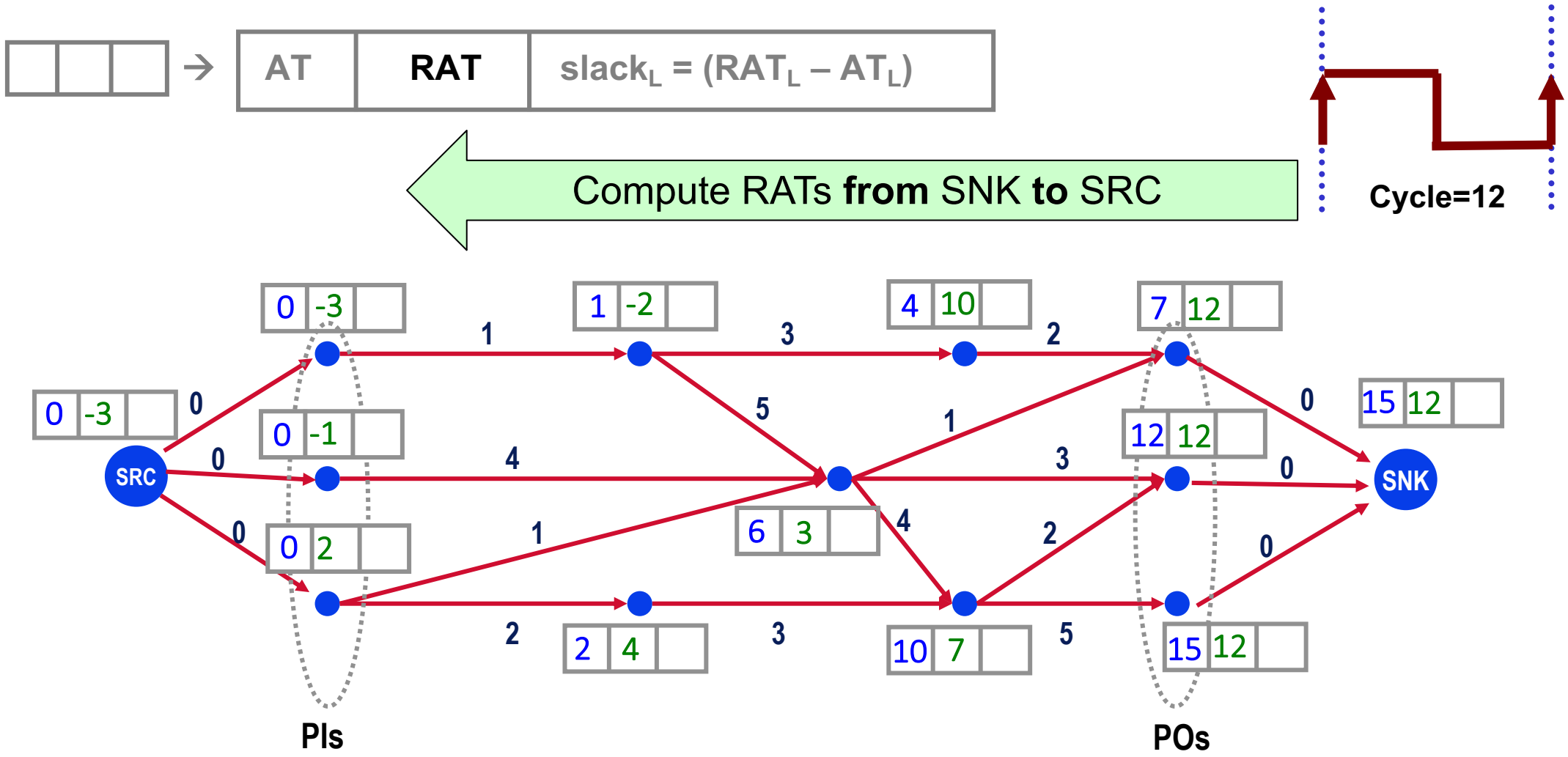




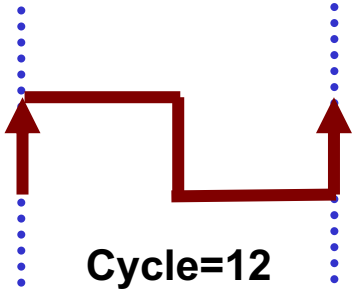
Recap: Compute ATs …



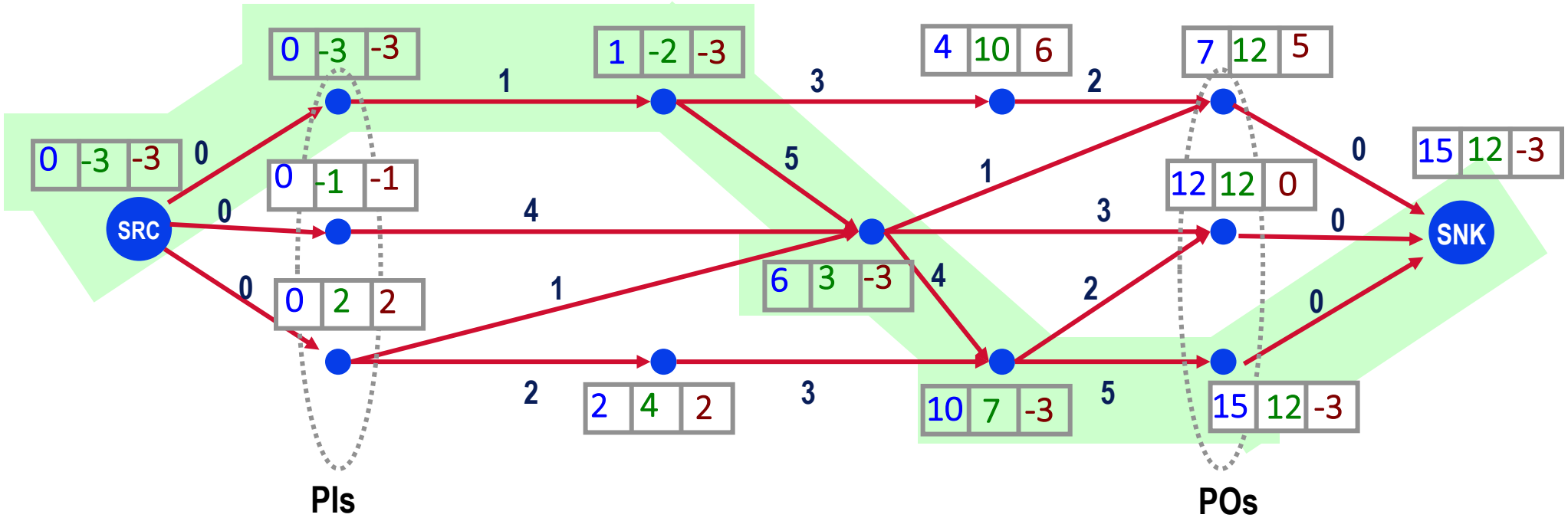
Recap: Compute RATs …



Recap: Compute Slacks …

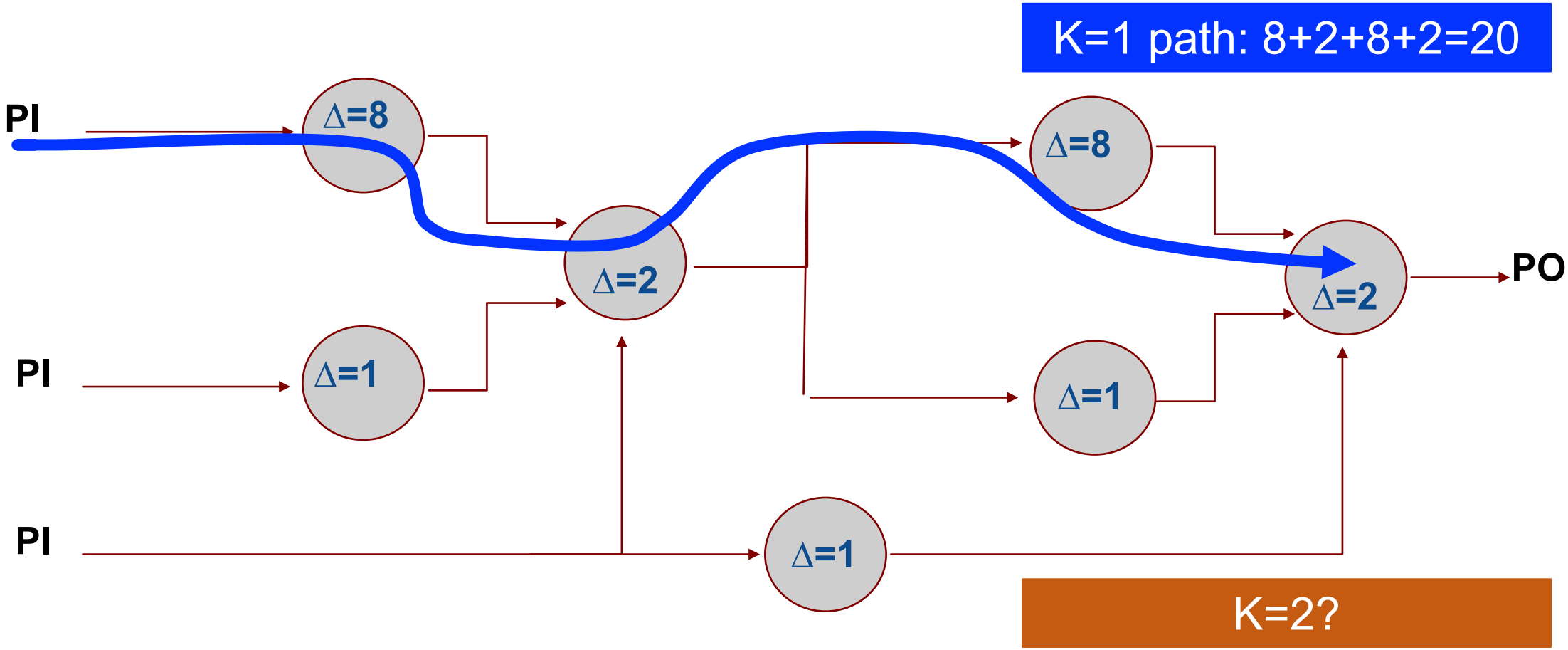


Worst (most negative slack) is **-3**. Trace **worst path**, SRC→SNK



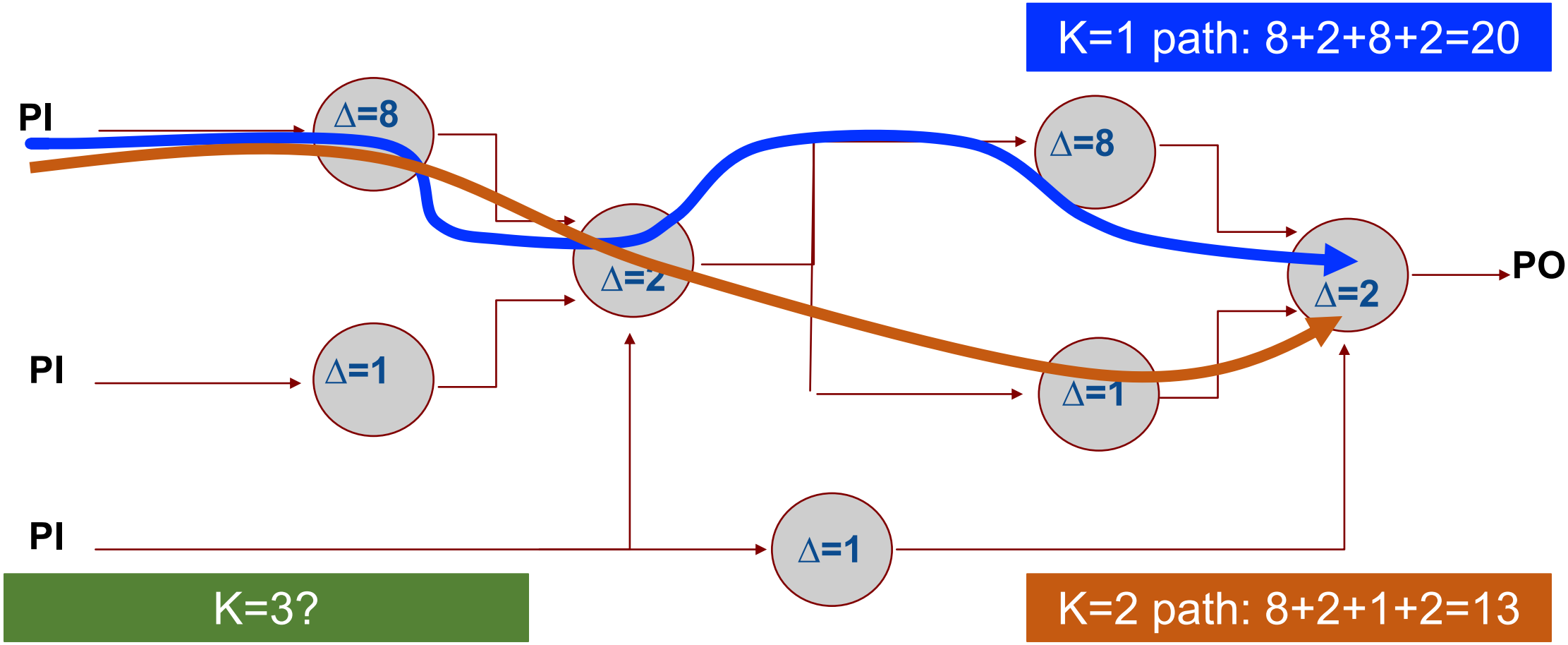
Top-k Critical Path Finding

- STA tools need to identify top-k critical paths (not just one)



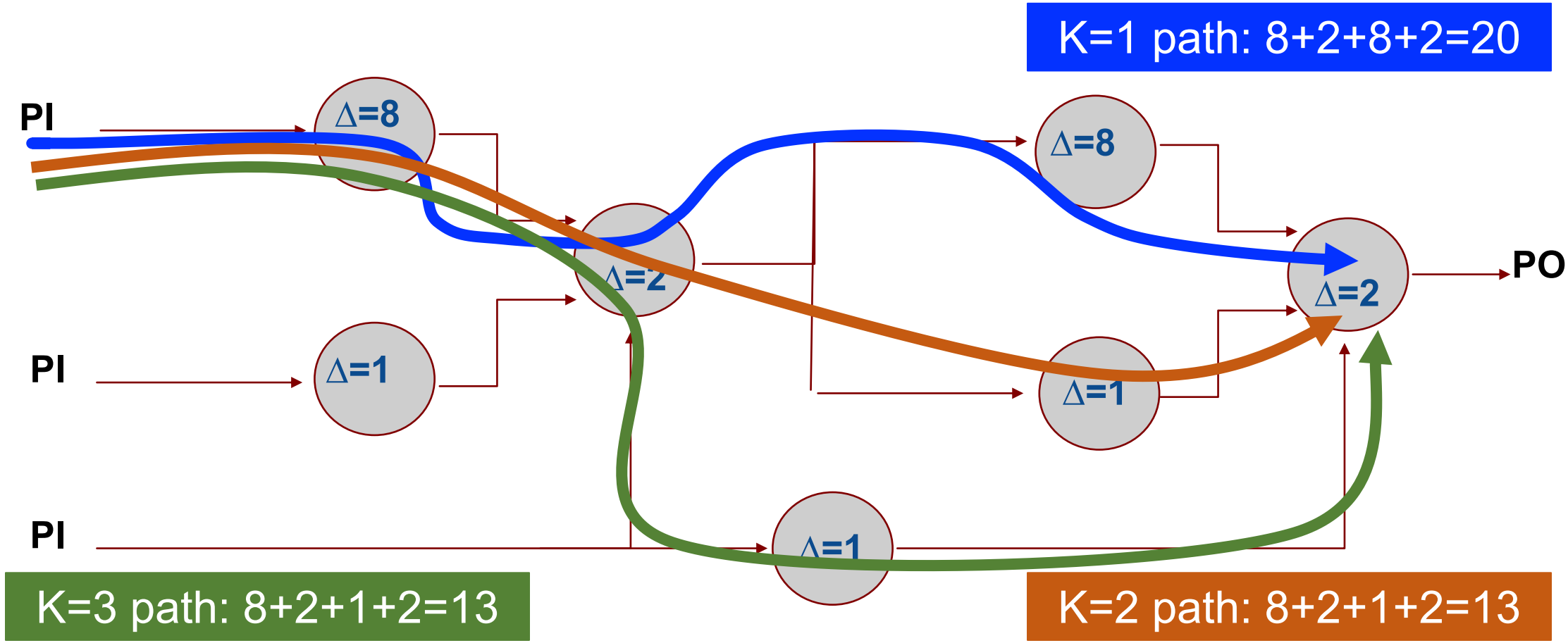
Top-k Critical Path Finding (cont'd)

- STA tools need to identify top-k critical paths (not just one)



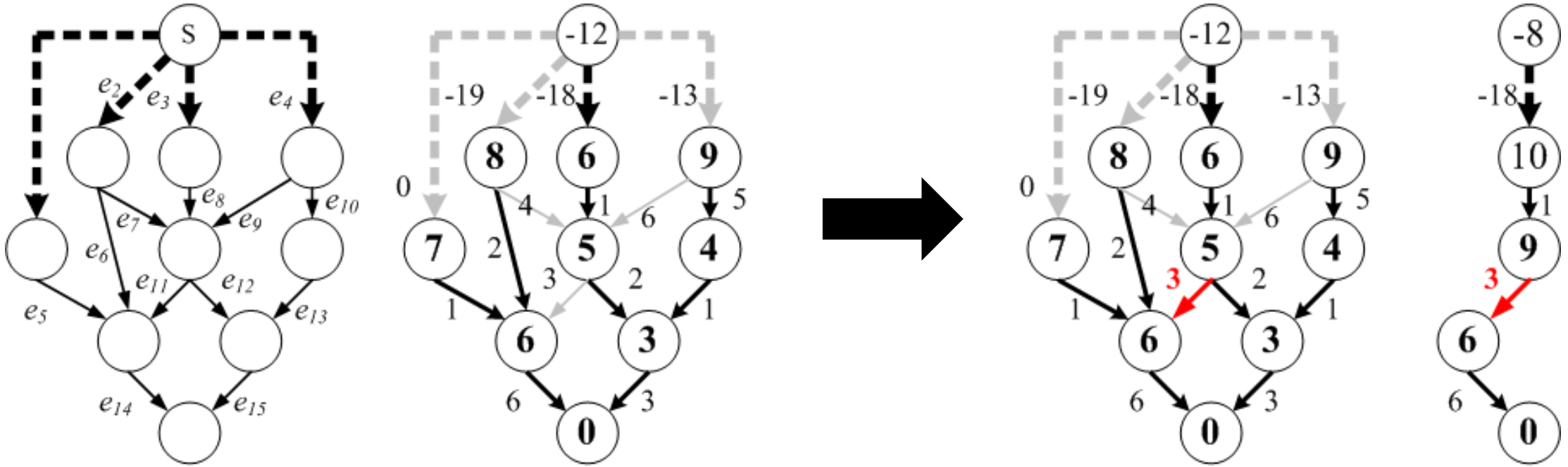
Top-k Critical Path Finding (cont'd)

- STA tools need to identify top-k critical paths (not just one)



Top-k Shortest Path Finding Algorithm

- O(N) explicit representation: Path = <e_1, e_2, e_3, … e_m>
- O(1) implicit representation: Path = <e_i>, wrt shortest path tree

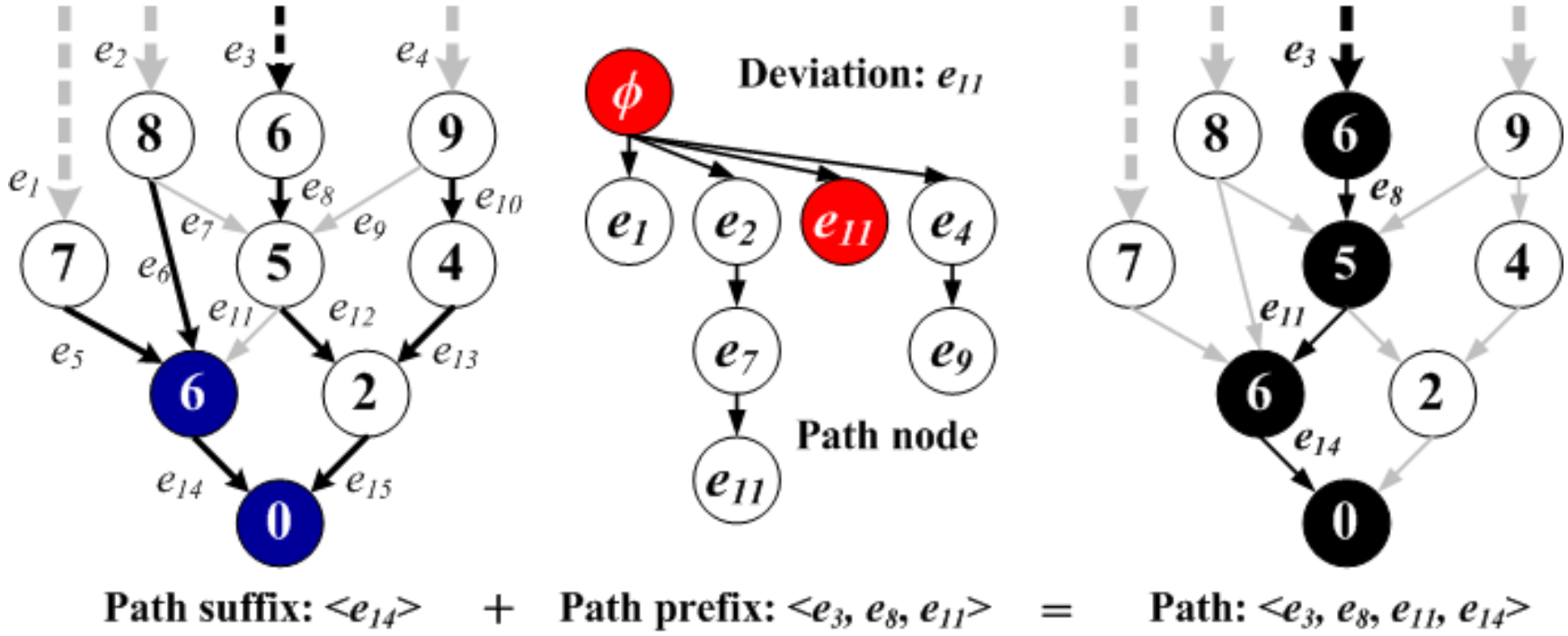


Shortest Path tree

Deviation on e_{11}

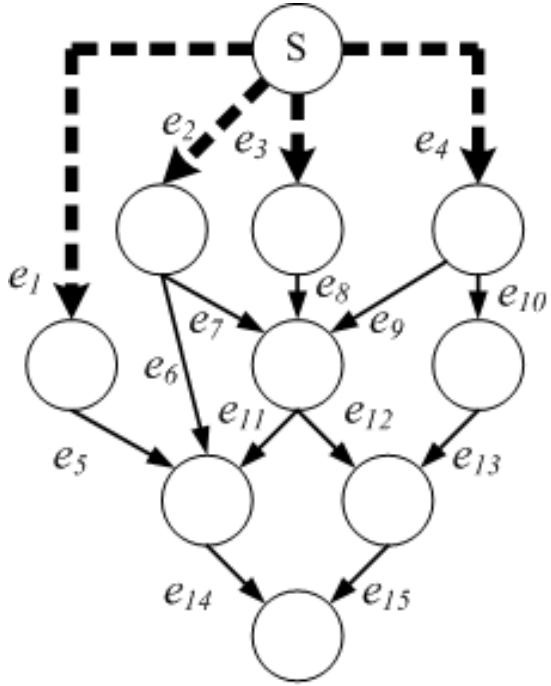
Suffix Tree and Prefix Tree

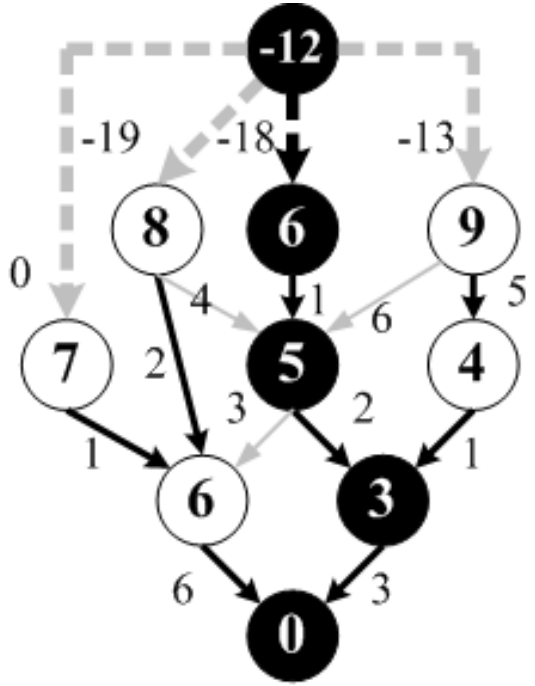
- Suffix tree: shortest path tree rooted at destination node
- Prefix tree: tree order of non-suffix-tree nodes (deviation)



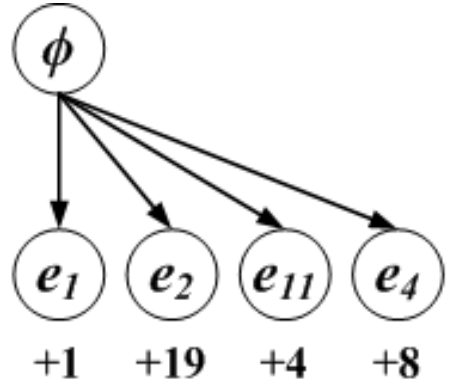
Extract Top-k Shortest Path: Supr

- Take a path *p* and generate all other paths deviated from *p*
- Grow the prefix tree based on deviation cost





Shortest path tree



4 paths spurred from the shortest path of slack -12

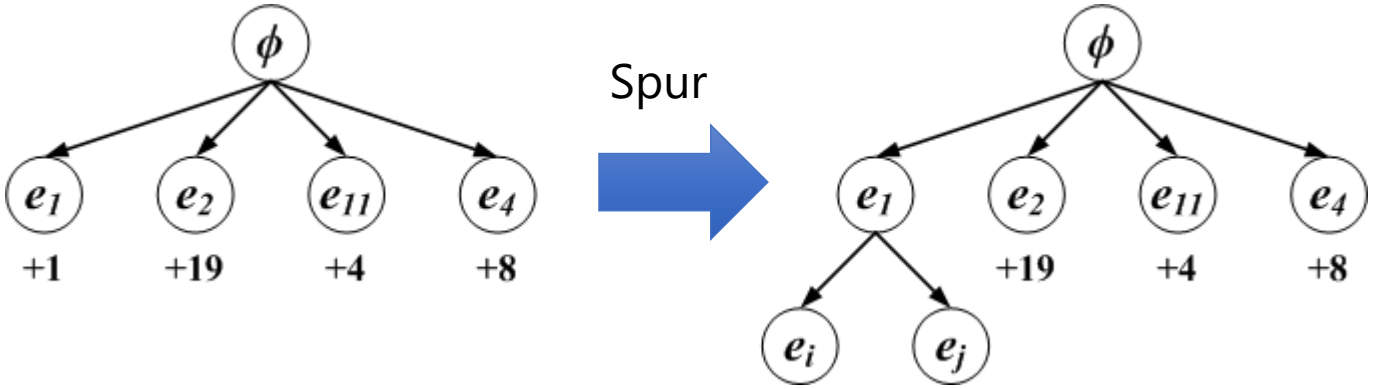
Slacks = {-11, 7, -8, -4}

Spur along the shortest path

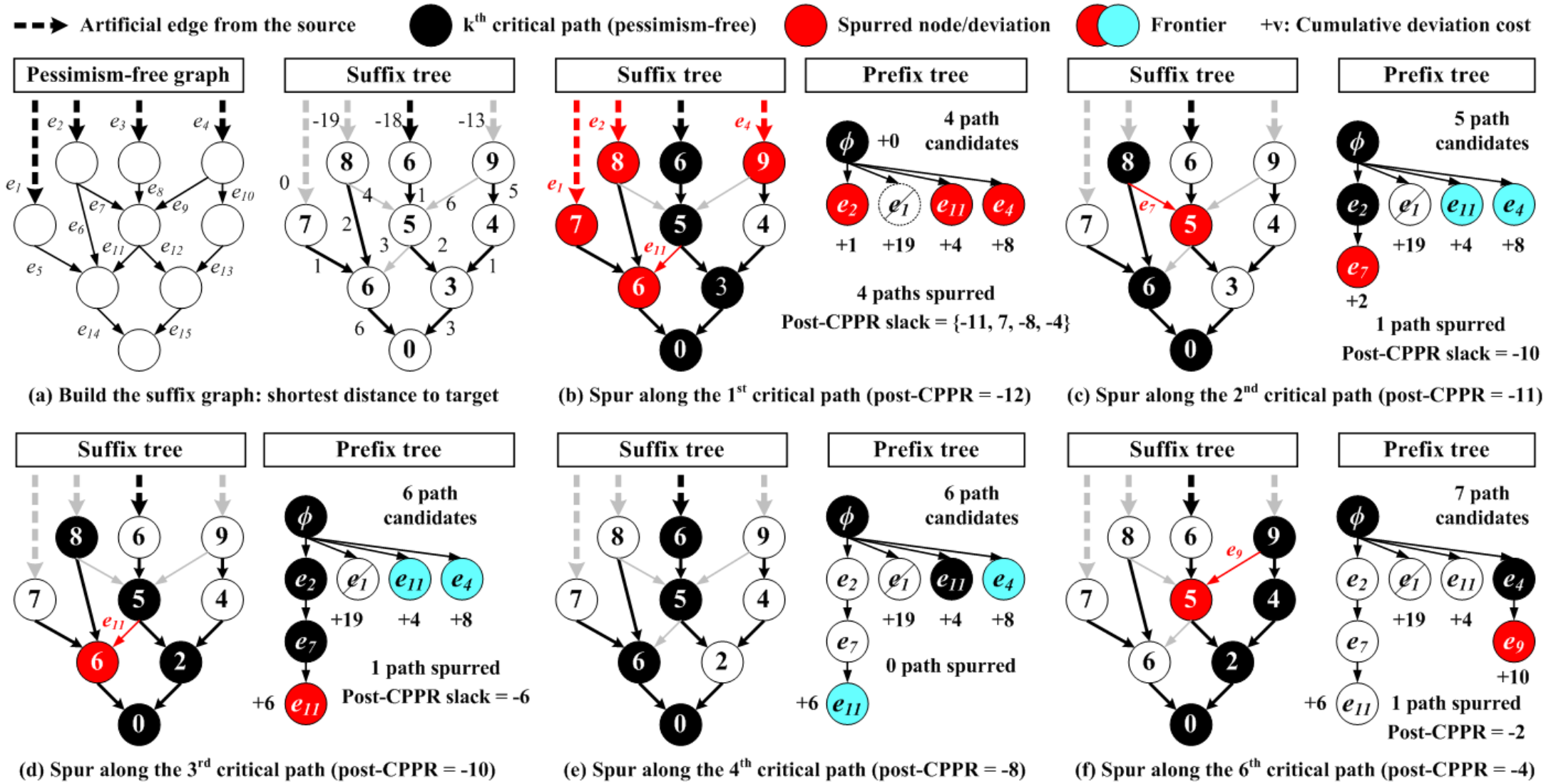
Priority Queue

1. Pick up a prefix-tree node with the minimum cost
2. Recover the path *p* and mark it as the k^{th} critical path
3. Generate all other paths deviated from *p*
4. Attach generated paths to prefix tree

Iterative expansion from the frontier node with the least cost



Example


Artificial edge from the source
k^th critical path (pessimism-free)
Spurred node/deviation
Frontier
+v: Cumulative deviation cost
Pessimism-free graph
Suffix tree
Suffix tree
Prefix tree
4 path candidates
4 paths spurred
Post-CPPR slack = {-11, 7, -8, -4}
Suffix tree
Prefix tree
5 path candidates
1 path spurred
Post-CPPR slack = -10
(a) Build the suffix graph: shortest distance to target
(b) Spur along the 1st critical path (post-CPPR = -12)
(c) Spur along the 2nd critical path (post-CPPR = -11)
Suffix tree
Prefix tree
6 path candidates
1 path spurred
Post-CPPR slack = -6
Suffix tree
Prefix tree
6 path candidates
0 path spurred
Suffix tree
Prefix tree
7 path candidates
1 path spurred
Post-CPPR = -2
(d) Spur along the 3rd critical path (post-CPPR = -10)
(e) Spur along the 4th critical path (post-CPPR = -8)
(f) Spur along the 6th critical path (post-CPPR = -4)

How Good is this Algorithm?

- TAU 2014 Contest: Common Path Pessimism Removal (CPPR) https://sites.google.com/site/taucontest2014/

1st Place: UI-Timer, University of Illinois at Urbana-Champaign, USA

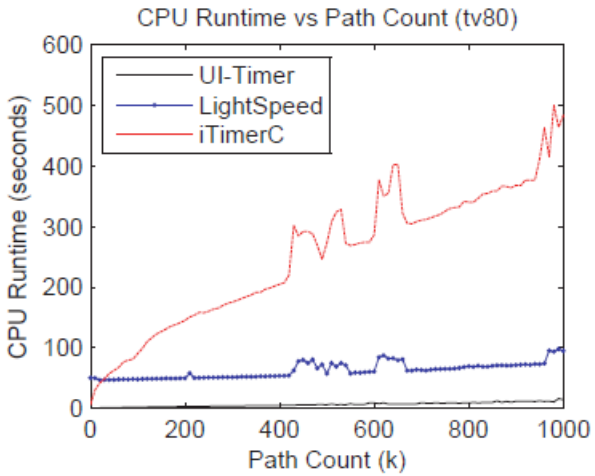
2nd Place: LightSpeed, IIT Madras, India

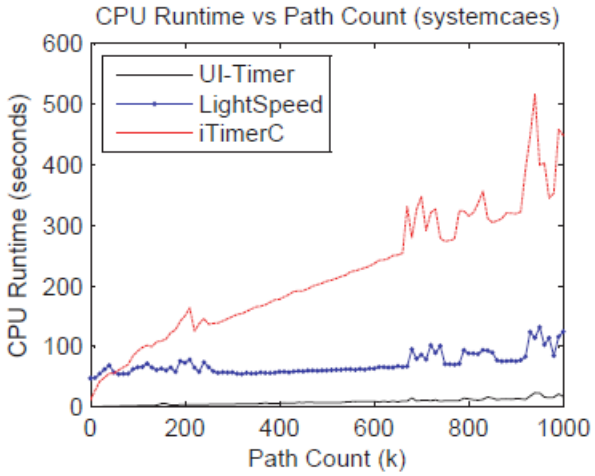
Experimental Results

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.





Tsung-Wei Huang, P.-C. Wu, and Martin Wong, "UI-Timer: An Ultra-Fast Clock Network Pessimism Removal Algorithm," IEEE/ACM ICCAD, pp. 758-765, 2014