ECE260B Winter 22

Static Timing Analysis (STA)

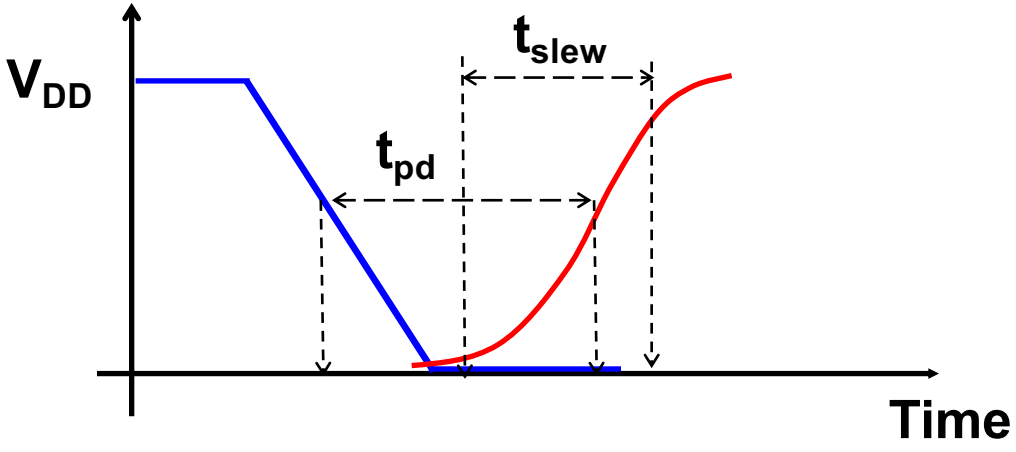
Prof. Mingu Kang

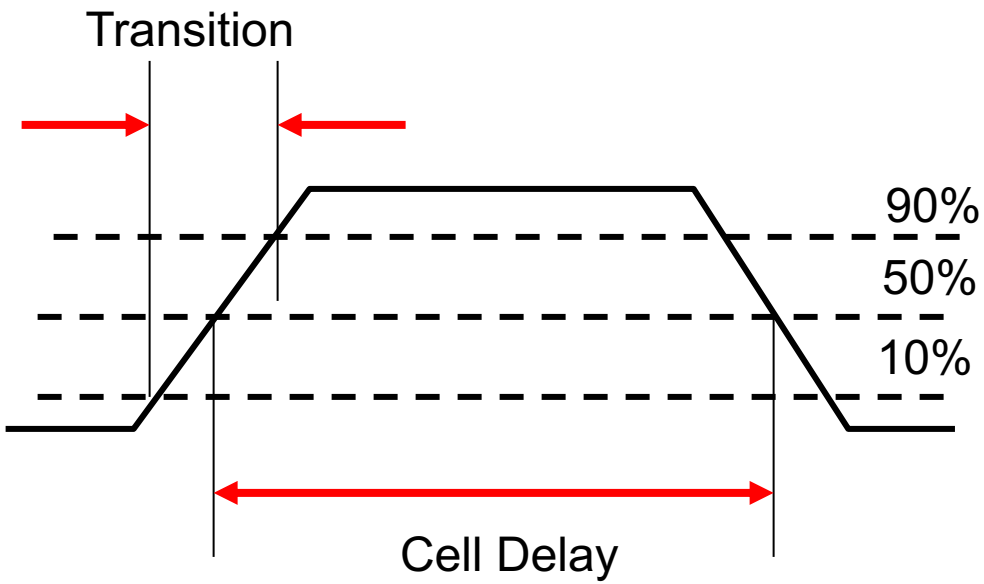
- Slides are modified based on Prof. Andrew Kahng's material

Timing Modeling

Cell Timing Characterization

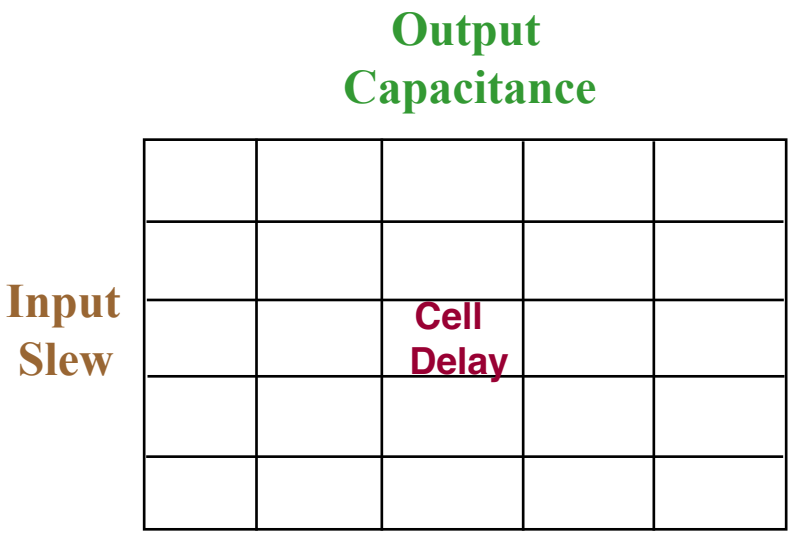
- Delay and output slew tables are generated using a transistor-level circuit simulator such as HSPICE / Spectre
- For a number of different input slews and load capacitances, simulate the circuit of the cell
  - Propagation time (50% V_{DD} at input to 50% at output)
  - Output transition/slew (e.g., 10% V_{DD} to 90% V_{DD} at output = rise time)

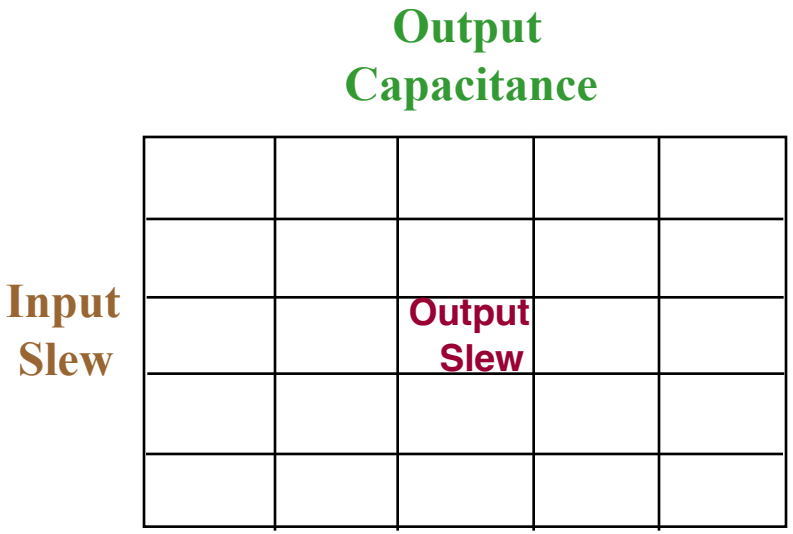


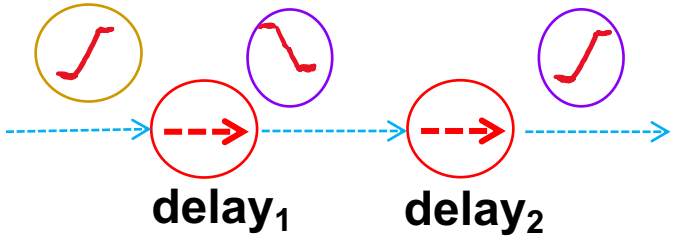


Delay and Slew Calculation with Look-up-table

- Delay = f (C_L, S_{in}) and $Slew_{out}$ = f (C_L, S_{in})
- Timing tool interpolates between such table entries
- Interpolation error is usually “less than 10% w.r.t. SPICE”







Timing Library Example (.lib = Liberty file)

```
    capacitive_load_unit (1,pf) ;
    voltage_unit : "1V" ;
    current_unit : "1mA" ;
    time_unit : "1ns" ;
    pulling_resistance_unit : "1kohm";
lu_table_template(delay_template_7x7) {
  variable_1 : input_net_transition;
  variable_2 : total_output_net_capacitance;
  index_1 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0, 1005.0, 1006.0");
  index_2 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0, 1005.0, 1006.0");
}
pin(Z)  {
  direction : output;
  max_capacitance : 0.0421;
  function : "(A1 A2)";
  timing() {
    related_pin : "A1";
    timing_sense : positive_unate;
    cell_rise(delay_template_7x7) {
      index_1 ("0.0040, 0.0120, 0.0272, 0.0576, 0.1184, 0.2408, 0.4848");
      index_2 ("0.0005, 0.0011, 0.0024, 0.0051, 0.0104, 0.0209, 0.0421");
  values("0.0296, 0.0341, 0.0421, 0.0570, 0.0860, 0.1436, 0.2588", \
         "0.0310, 0.0356, 0.0435, 0.0584, 0.0875, 0.1450, 0.2606", \
         "0.0342, 0.0387, 0.0466, 0.0615, 0.0906, 0.1482, 0.2634", \
         "0.0403, 0.0447, 0.0527, 0.0675, 0.0966, 0.1542, 0.2697", \
         "0.0477, 0.0524, 0.0606, 0.0756, 0.1046, 0.1622, 0.2778", \
         "0.0563, 0.0616, 0.0703, 0.0853, 0.1144, 0.1721, 0.2874", \
         "0.0652, 0.0715, 0.0816, 0.0980, 0.1271, 0.1850, 0.3003");
    }
    rise_transition(delay_template_7x7) {
      index_1 ("0.0040, 0.0120, 0.0272, 0.0576, 0.1184, 0.2408, 0.4848");
      index_2 ("0.0005, 0.0011, 0.0024, 0.0051, 0.0104, 0.0209, 0.0421");
  values("0.0168, 0.0230, 0.0357, 0.0621, 0.1169, 0.2272, 0.4473", \
         "0.0168, 0.0230, 0.0356, 0.0620, 0.1169, 0.2279, 0.4484", \
         "0.0168, 0.0231, 0.0356, 0.0621, 0.1170, 0.2279, 0.4484", \
         "0.0176, 0.0236, 0.0360, 0.0622, 0.1170, 0.2274, 0.4483", \
         "0.0202, 0.0258, 0.0377, 0.0633, 0.1171, 0.2271, 0.4484", \
         "0.0251, 0.0304, 0.0408, 0.0650, 0.1184, 0.2282, 0.4484", \
         "0.0334, 0.0393, 0.0495, 0.0706, 0.1213, 0.2294, 0.4492");
    }
```

Index_1 (input trans) ↓

Index_2 (output load) →

- "cell rise/fall" section: 50%-50% delay
- "rise/fall transition": rising / falling slew

Delay Calculation

Fall delay of INV cell: look at Rising input transition.
T_slew = **0.1ns**, C_load = **1.0pF**
Fall delay = 2/3 * (2/3 * 0.04 + 1/3 * 0.08) + 1/3 * (2/3 * 0.32 + 1/3 * 0.64) = 0.178ns

Cell Fall

Tran\Cap	0.01	0.5	2.0
0.05	0.02	0.04	0.08
0.2	0.15	0.32	0.64
0.5	0.30	0.60	1.20

0.178

Cell Rise

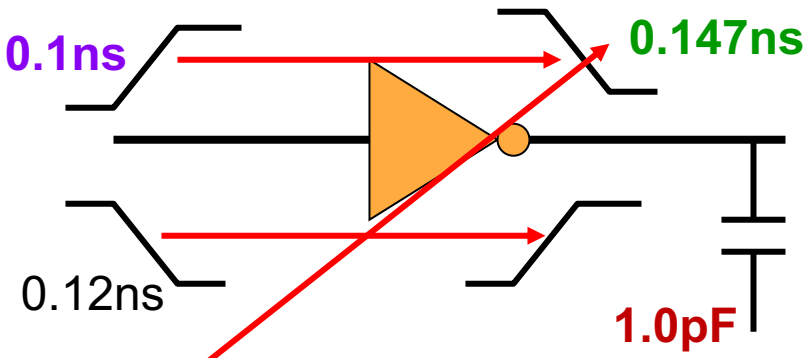
Tran\Cap	0.01	0.5	2.0
0.05	0.03	0.06	0.09
0.2	0.18	0.36	0.72
0.5	0.33	0.66	1.32

0.261

Fall Transition

Tran\Cap	0.01	0.5	2.0
0.05	0.01	0.03	0.06
0.2	0.09	0.27	0.54
0.5	0.15	0.45	0.90

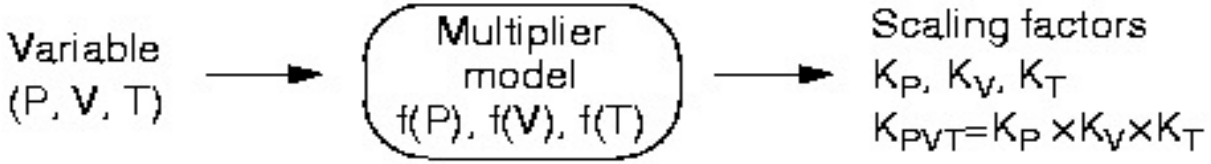
0.147



Fall delay = 0.178ns
Rise delay = 0.261ns
Fall transition = 0.147ns
Rise transition = …

Fall Transition at output: 2/3 * (2/3 * 0.03 + 1/3 * 0.06) + 1/3 * (2/3 * 0.27 + 1/3 * 0.54) = 0.147ns

PVT (Process, Voltage, Temperature) Combinations



Actual cell delay = nominal delay x K_{PVT}

Proc_var (0.5:1.0:1.3)
Voltage (5.5:5.0:4.5)
Temperature (0:20:50)
K_P = 0.80 : 1.00 : 1.30
K_V = 0.93 : 1.00 : 1.08
K_T = 0.80 : 1.00 : 1.35

K_{PVT} = 0.60 : 1.00 : 1.90

Cell delay = 0.261ns
Delay = 0.157 : 0.261 : 0.496 {min : typical : max}

Observation: 0.496/0.157 = **3.16X range of delays** from BC (fastest) to WC (slowest)

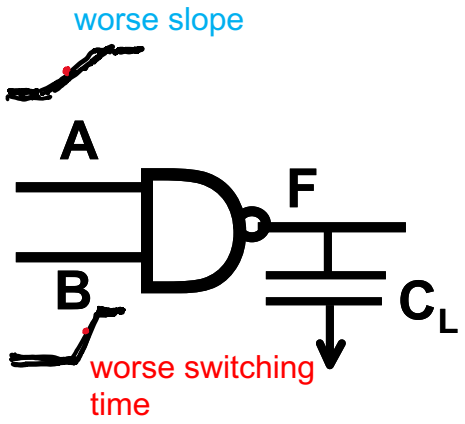
- V = {0.90, 1.0, 1.10}; T = {125C, 25C, -40C}; process = {SS,TT, FF};
- Interconnect = {RCworst, Cworst, Rcnom, Cbest, RCbest}

>100 “corner-mode” combinations

Conservatism of Gate Delay Modeling, Propagation

- True gate delay depends on input arrival time patterns
  - STA classically assumes that only 1 input is switching
  - "Graph-based STA" propagates the worst slope among several inputs

WC example

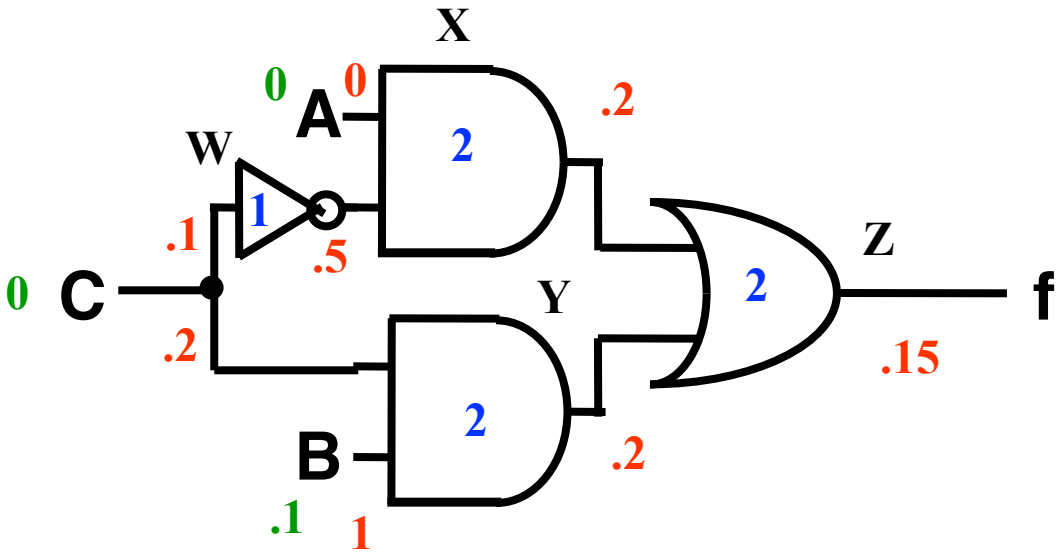


- A: earlier but **slower** transition
- B: **later** but faster transition
- Graph Based (GBA) STA uses **later** + **slower**
- Path-based analysis ("PBA") avoids pessimism of "GBA" at large runtime cost

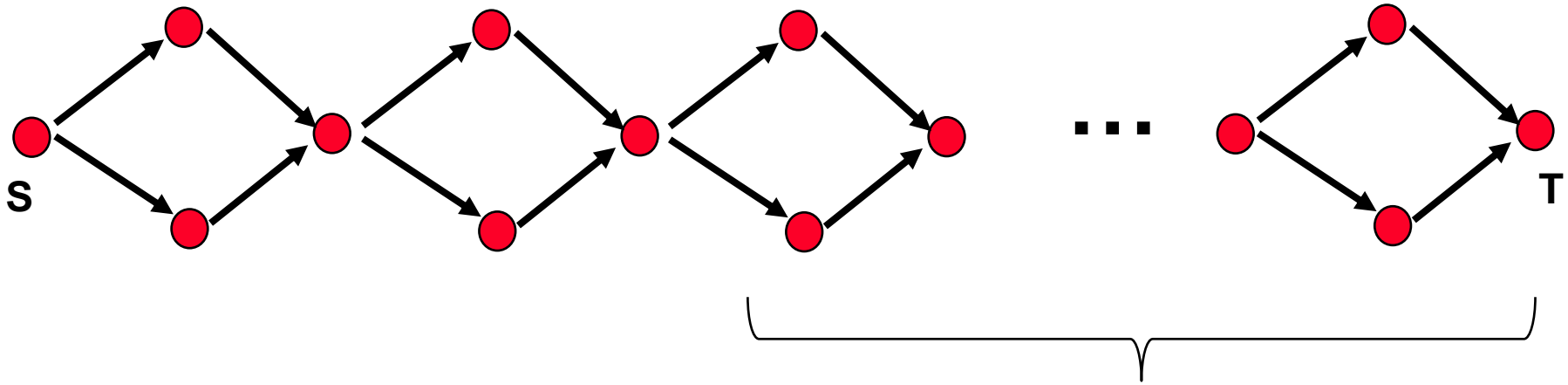
Static Timing Analysis

Calculating the Path Delay

- **Arrival time in green**
- Interconnect delay in red
- Gate delay in blue
- Labeled directed graph representation
  - PIs: A, B, C
  - POs: f
  - Gates: W, X, Y, Z



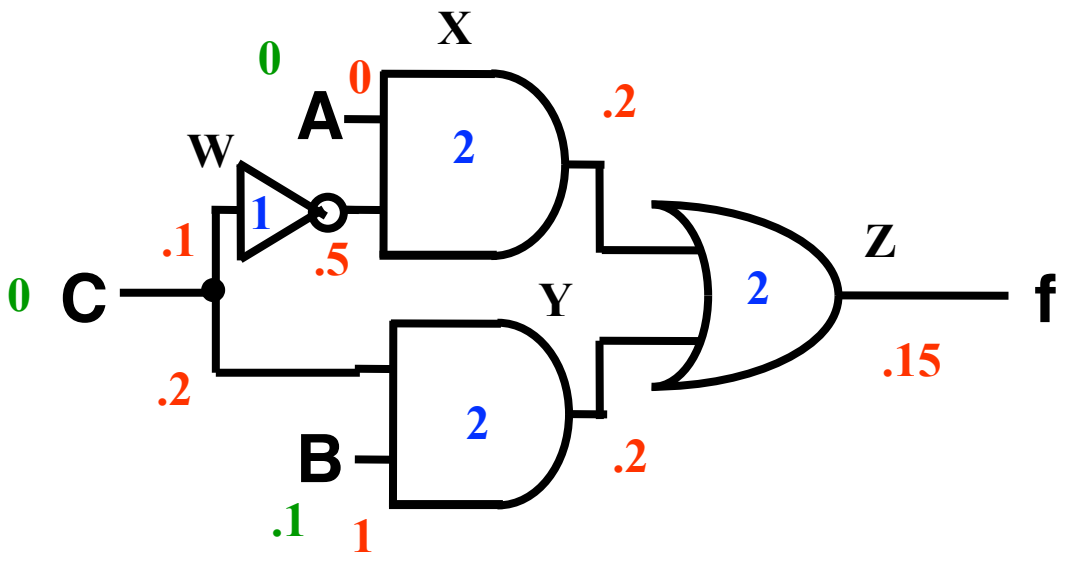
Difficult in Calculating Timing Paths

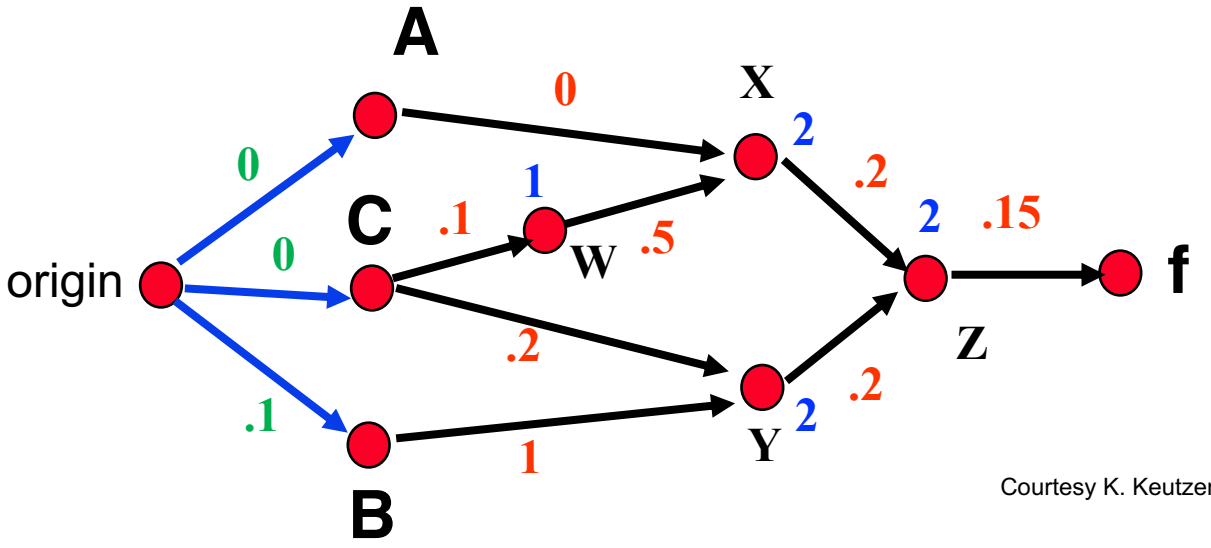


→ Number of timing paths can be exponential in size of circuit (2^N)

Problem Formulation

- Use a labeled *directed* graph
- *G* = <*Vertices, Edges*>
- *Vertices* represent gates, primary inputs and primary outputs
- *Edges* represent wires
- *Labels on the edge* represent delays



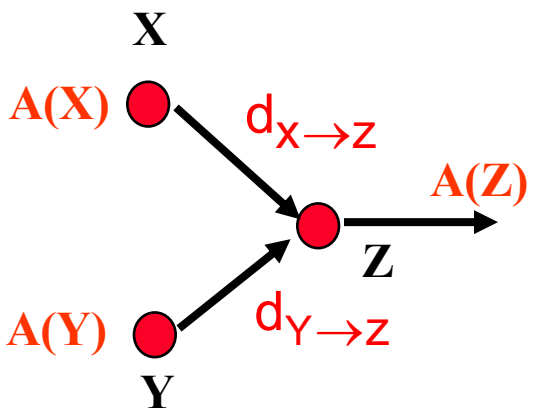


Courtesy K. Keutzer et al. UCB

Note: This is a simplistic view of the timing graph! A real timing graph will (at least) have a vertex for each input or output pin of each gate, and will also distinguish rise and fall delays.

Problem Formulation – Actual Arrival Time

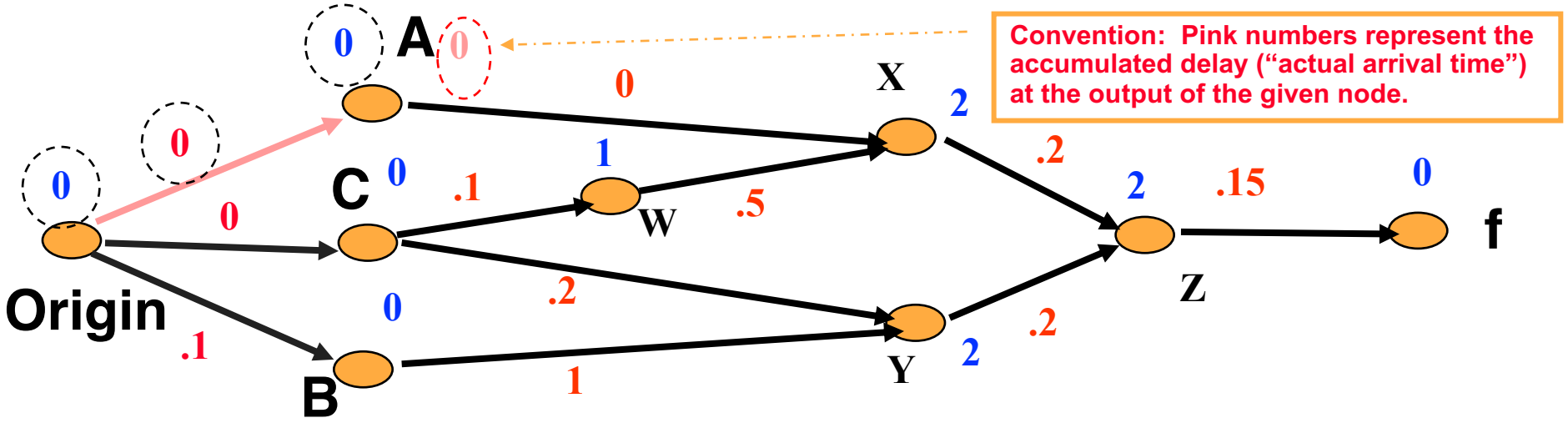
- Arrival time A(v) for a node v is <u>latest</u> time that signal <u>can</u> arrive at node v



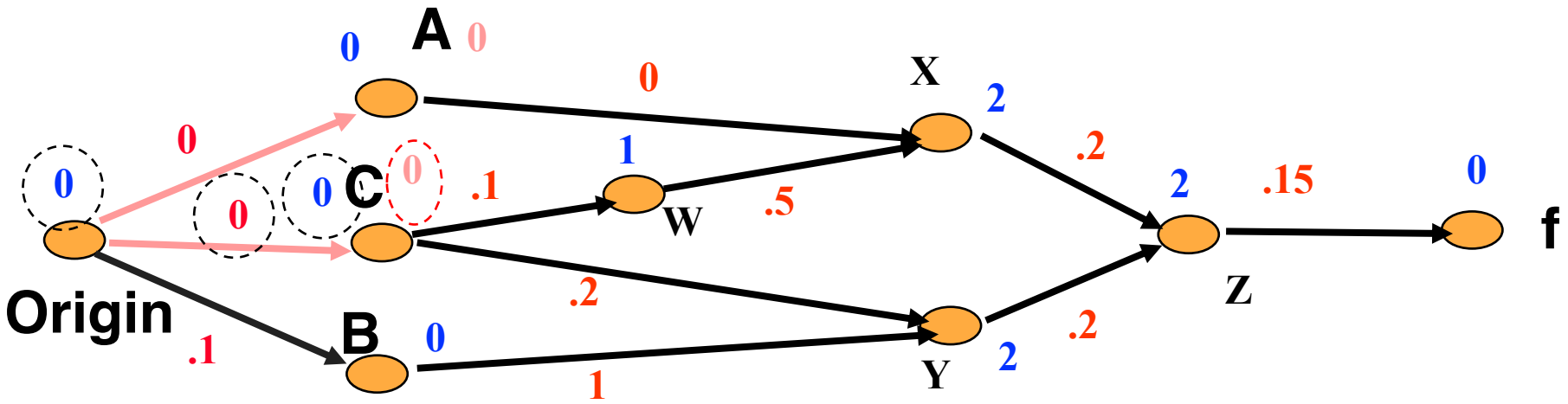
$$A(\upsilon) = \max_{u \in FI(\upsilon)} (A(u) + d_{u\to\upsilon})$$

where $d_{u\to v}$ **is delay from** u **to** v, $FI(V) = \{X, Y\}$, **and** $\upsilon = \{Z\}$.

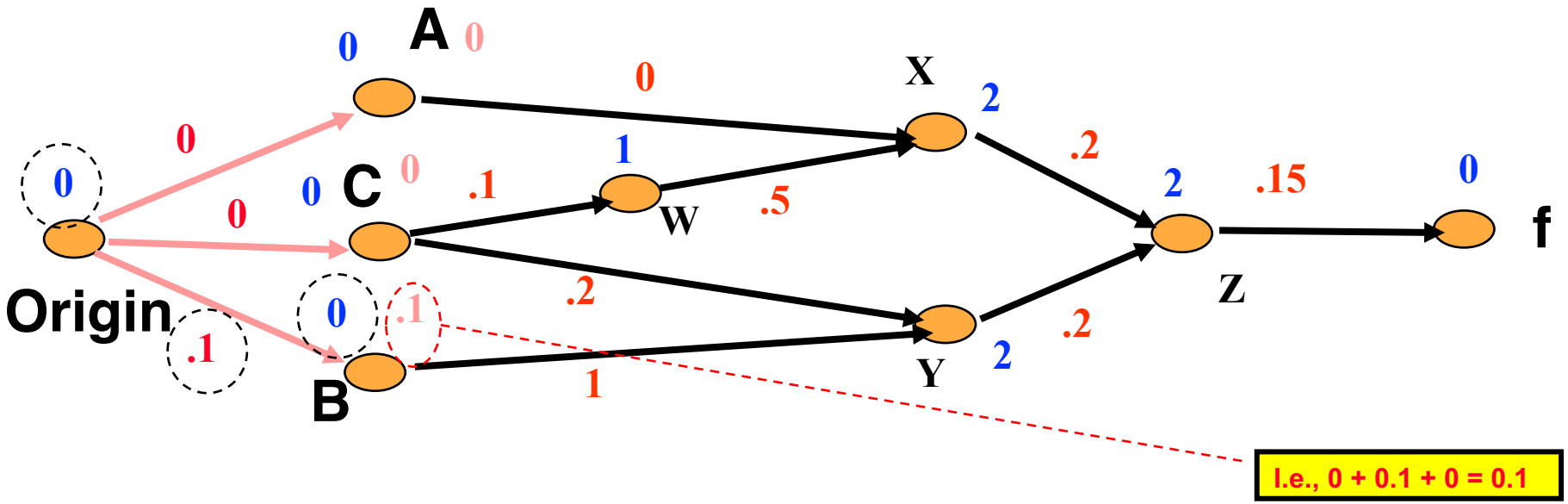
Calculating Arrival Time Example



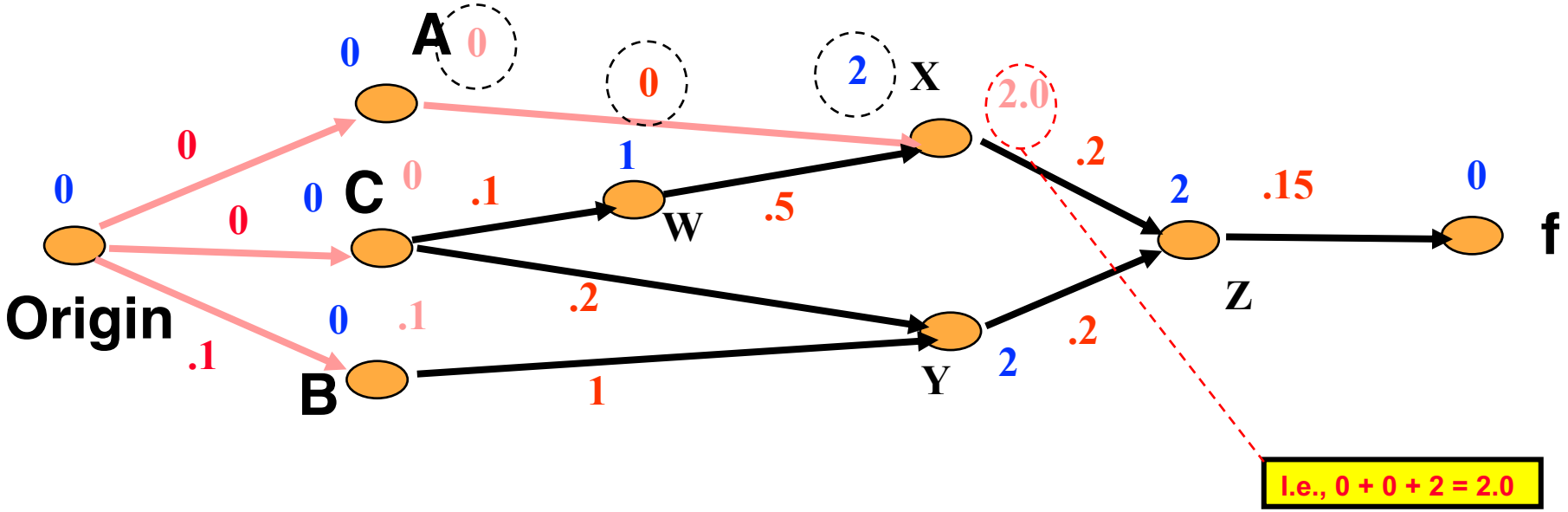
Calculating Arrival Time Example



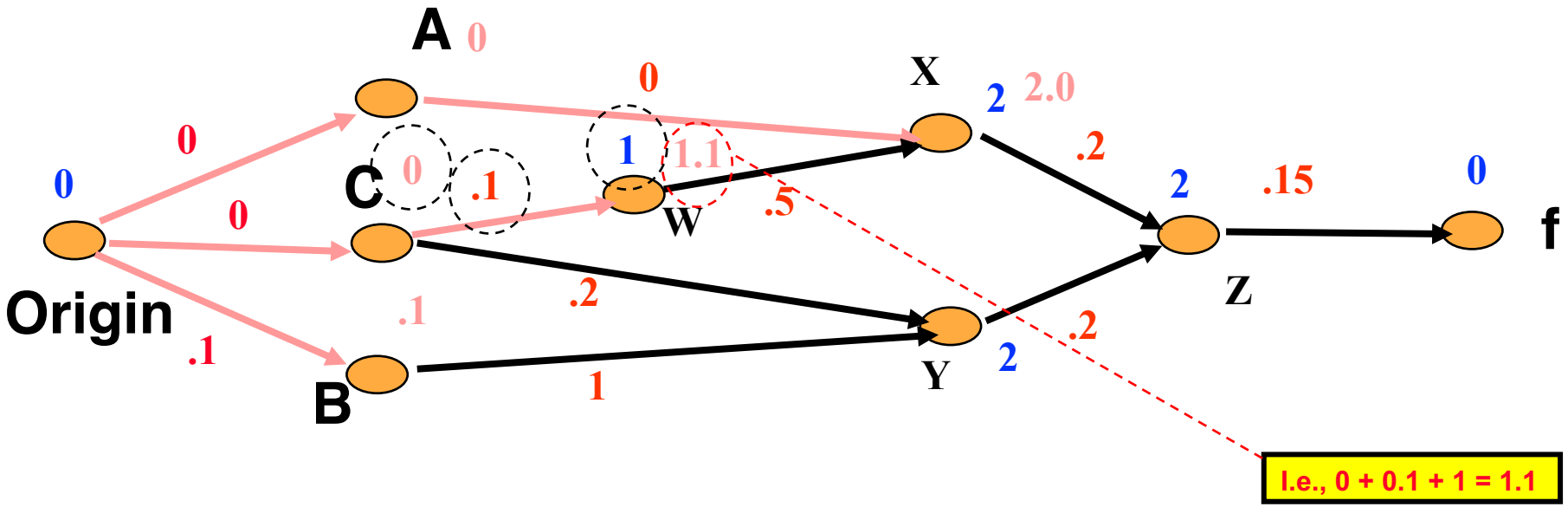
Calculating Arrival Time Example



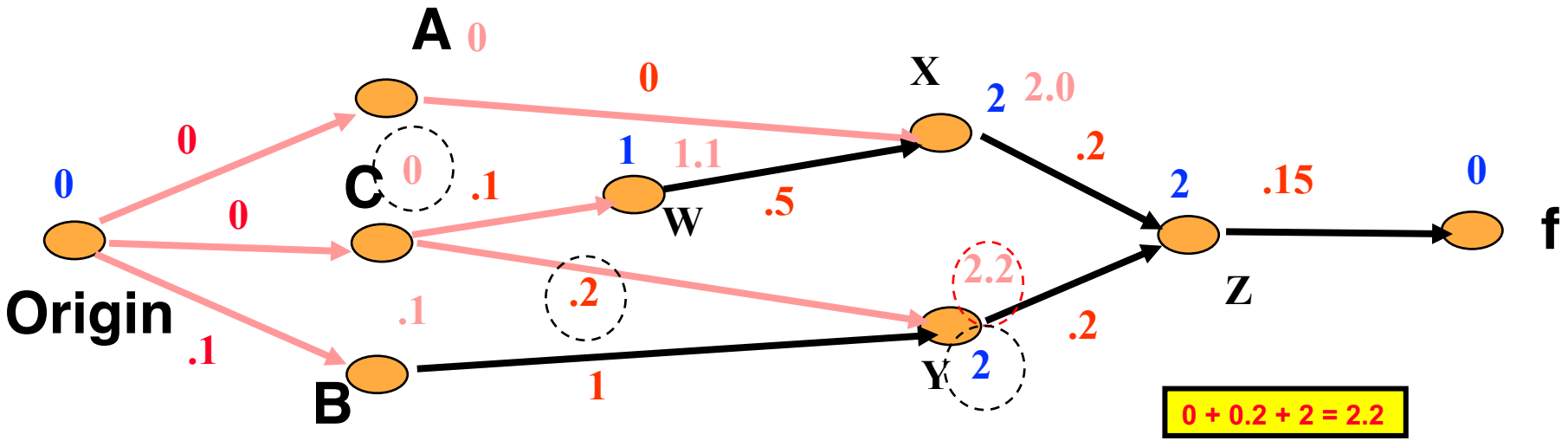
Calculating Arrival Time Example



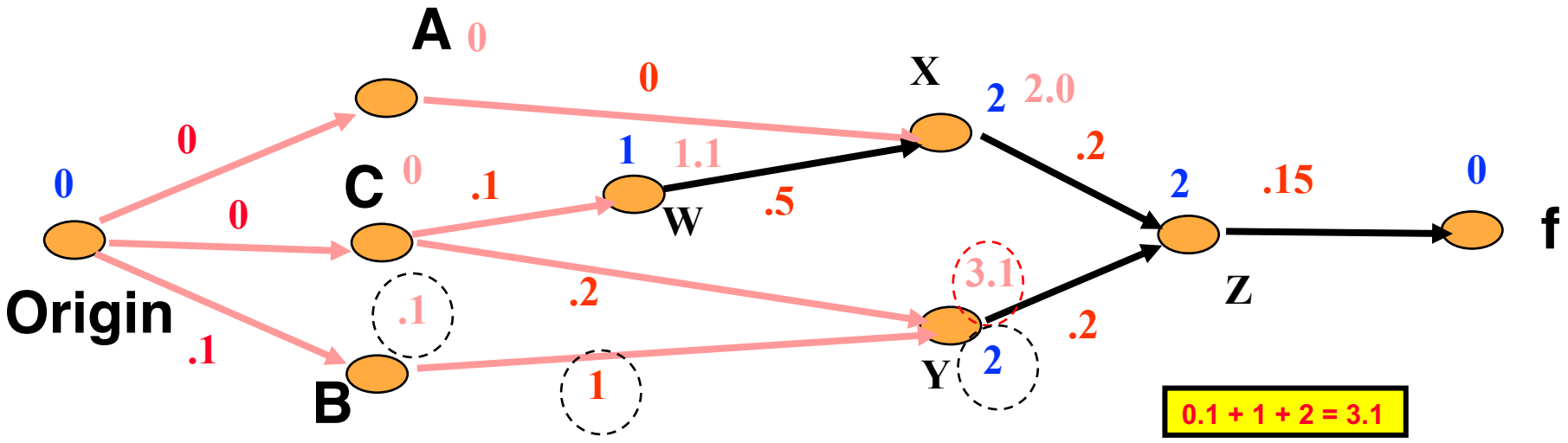
Calculating Arrival Time Example



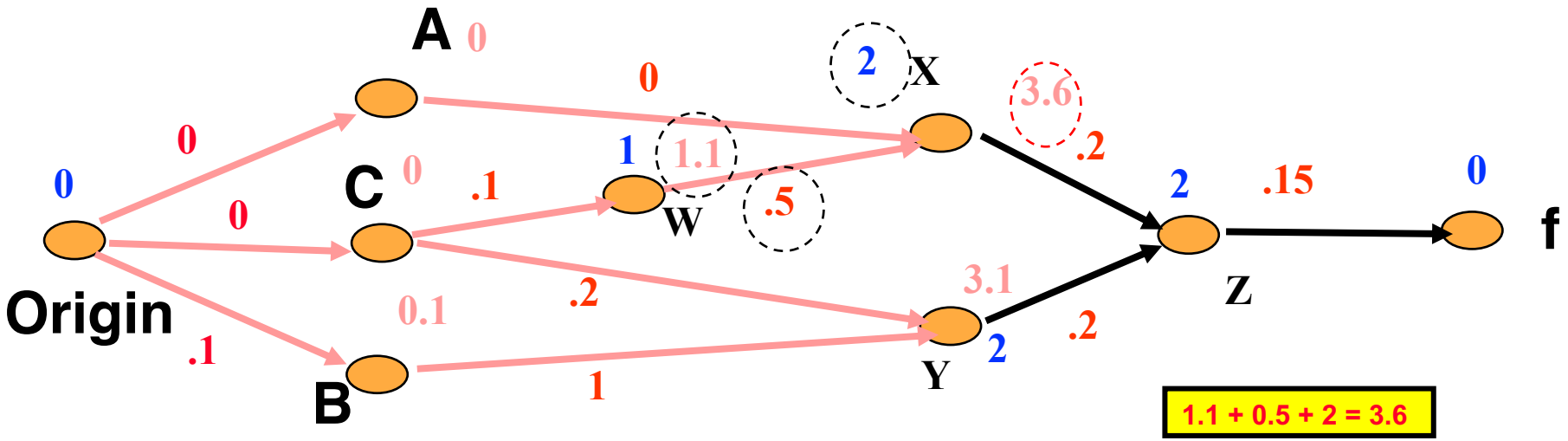
Calculating Arrival Time Example



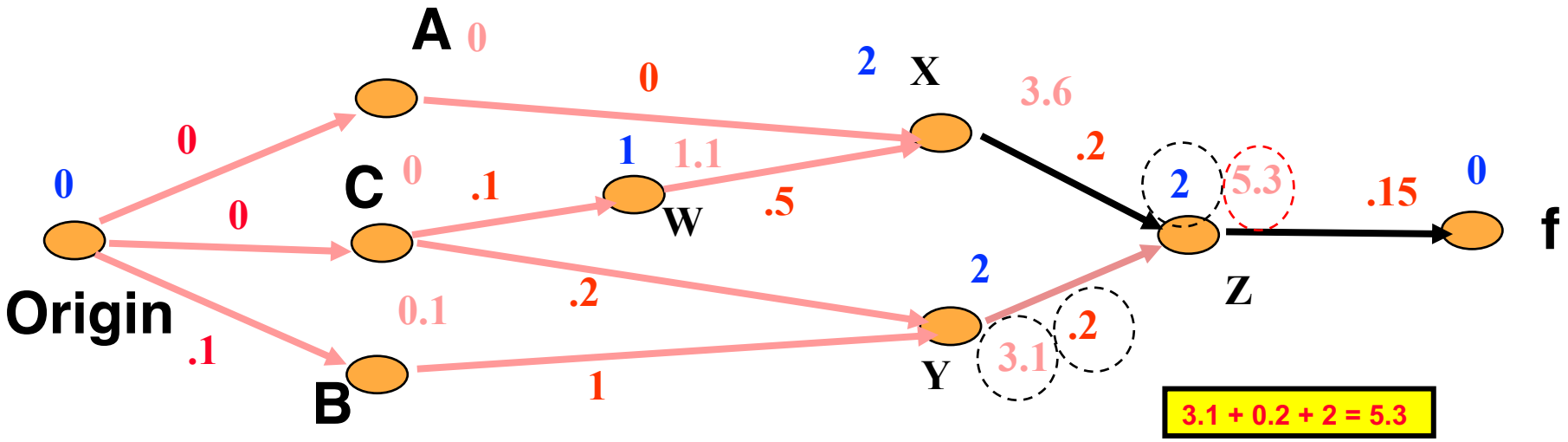
Calculating Arrival Time Example



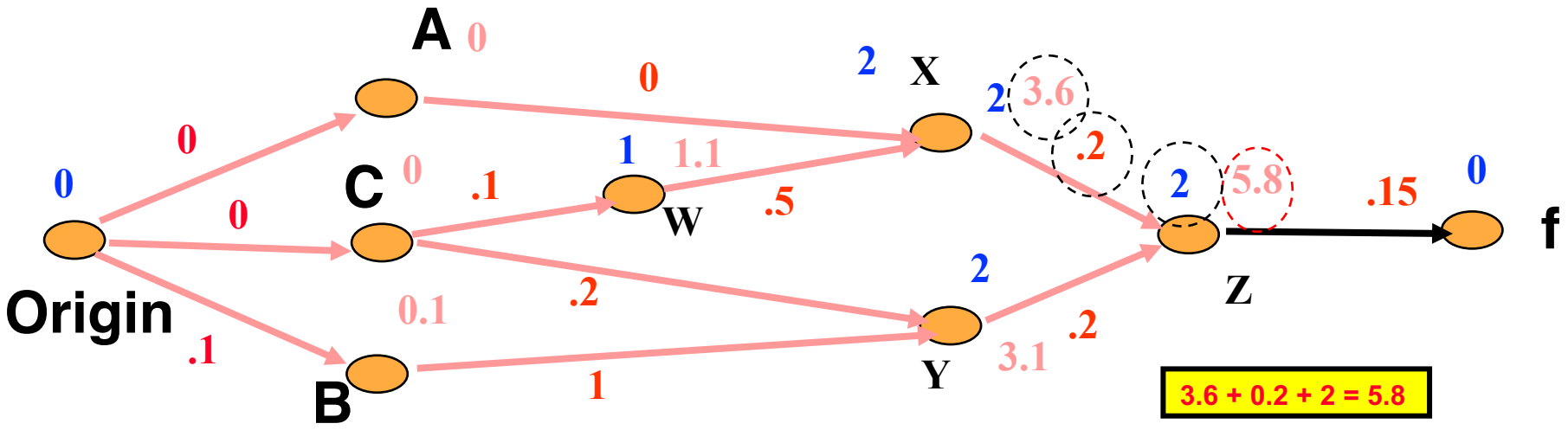
Calculating Arrival Time Example



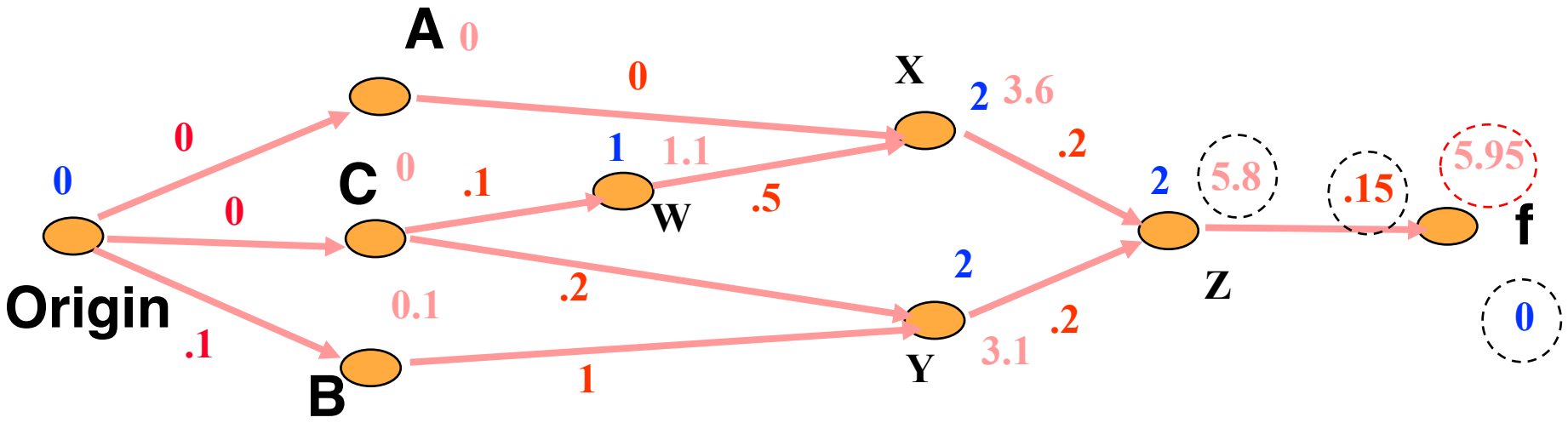
Calculating Arrival Time Example



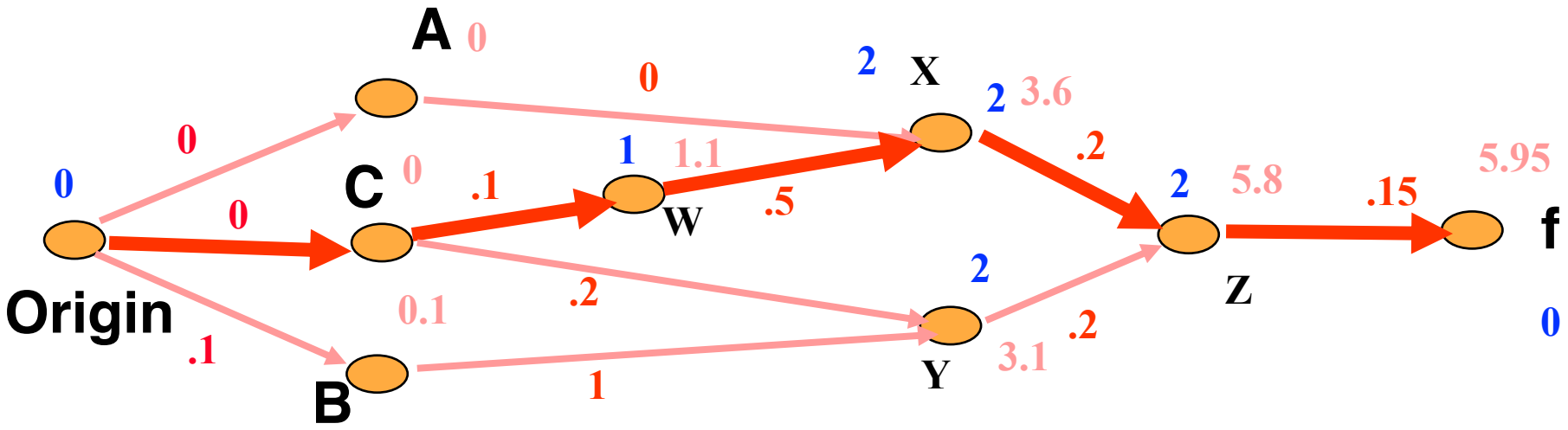
Calculating Arrival Time Example



Calculating Arrival Time Example



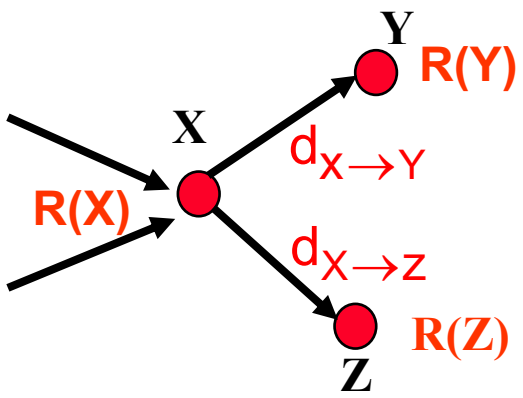
Calculating Arrival Time Example



In this way, rather than considering all the possible path combinations, only focus on the worst-case delay
Don’t need to compute 2^N combination, but linearly increased computations

Required Arrival Time

- Required arrival time R(v) is the time before which a signal must arrive to avoid a timing violation

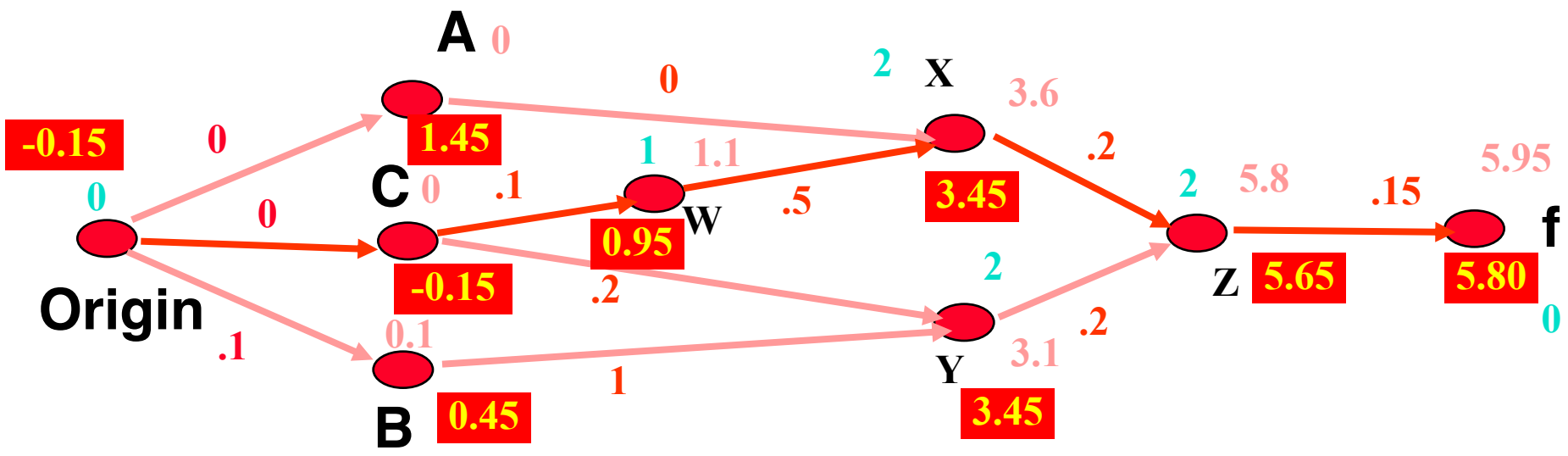


$$R(\upsilon) = \min_{u \in FO(\upsilon)} (R(u) - d_{\upsilon \to u})$$

- Then recursively

  where *FO(X) = {Y, Z}*

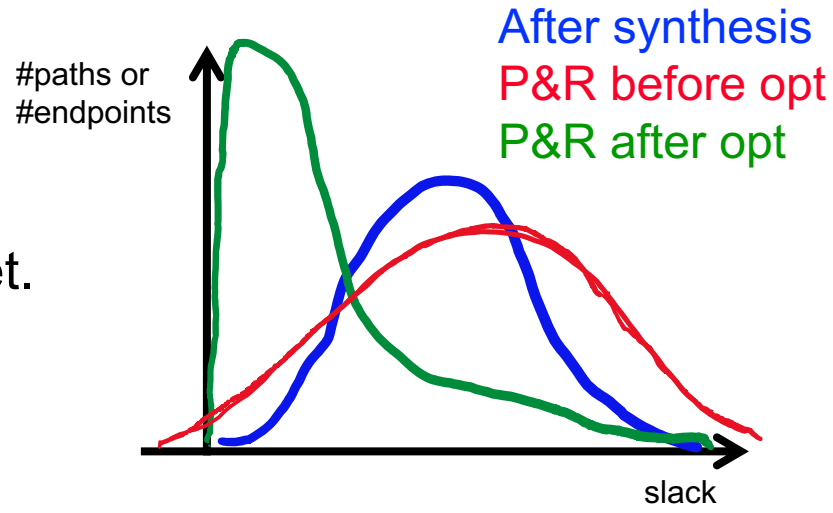
Required Time Propagation



- Assume required time at output R(f) = 5.80
- Propagate required times backwards

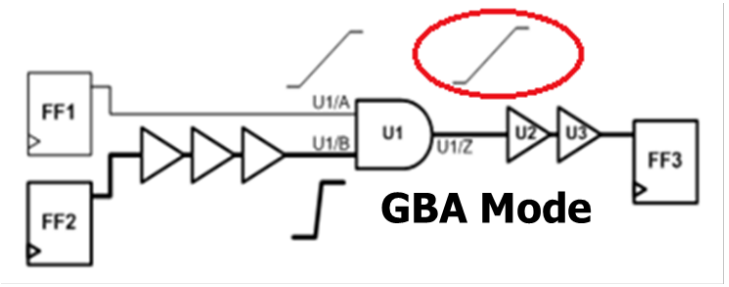
Timing Slack

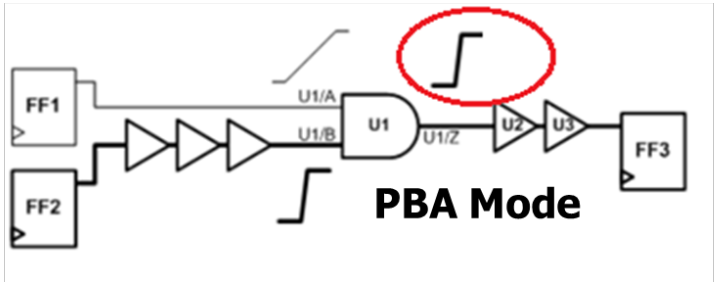
- Can compute slack from arrival and required times.
- Slack reflects *criticality* of a node
- Positive slack
  - Node is not on critical path. Timing constraints met.
- Zero slack
  - Node is on critical path. Timing constraints are barely met.
- Negative slack
  - There is a timing violation
- Slack distribution is key for timing optimization
  - Slack does not need to be large number
  - Should avoid over design to save the power

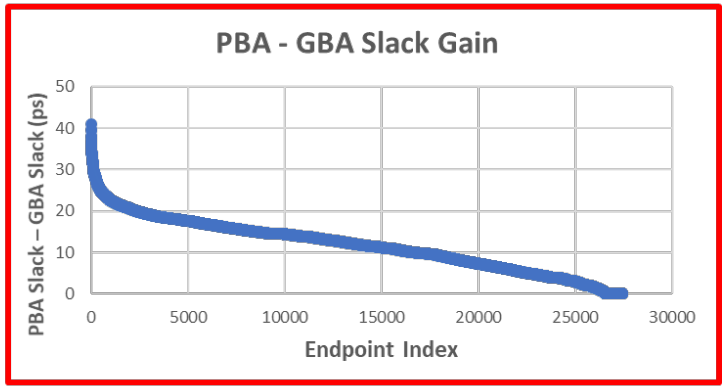


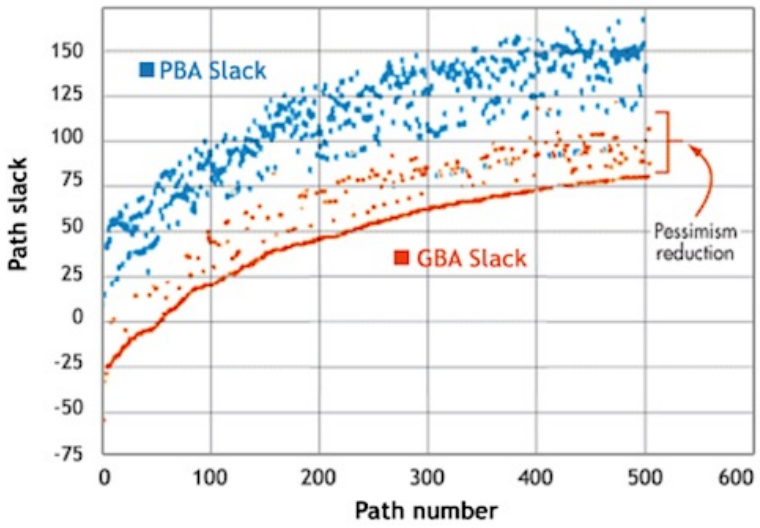
STA Enhancement

- **Incremental** timing analysis ("iSTA")
- by considering interference and crosstalk
- Multiple inputs switching
- by applying "PBA" instead of "GBA"
- by considering process variation (by probabilistic or statistical STA)







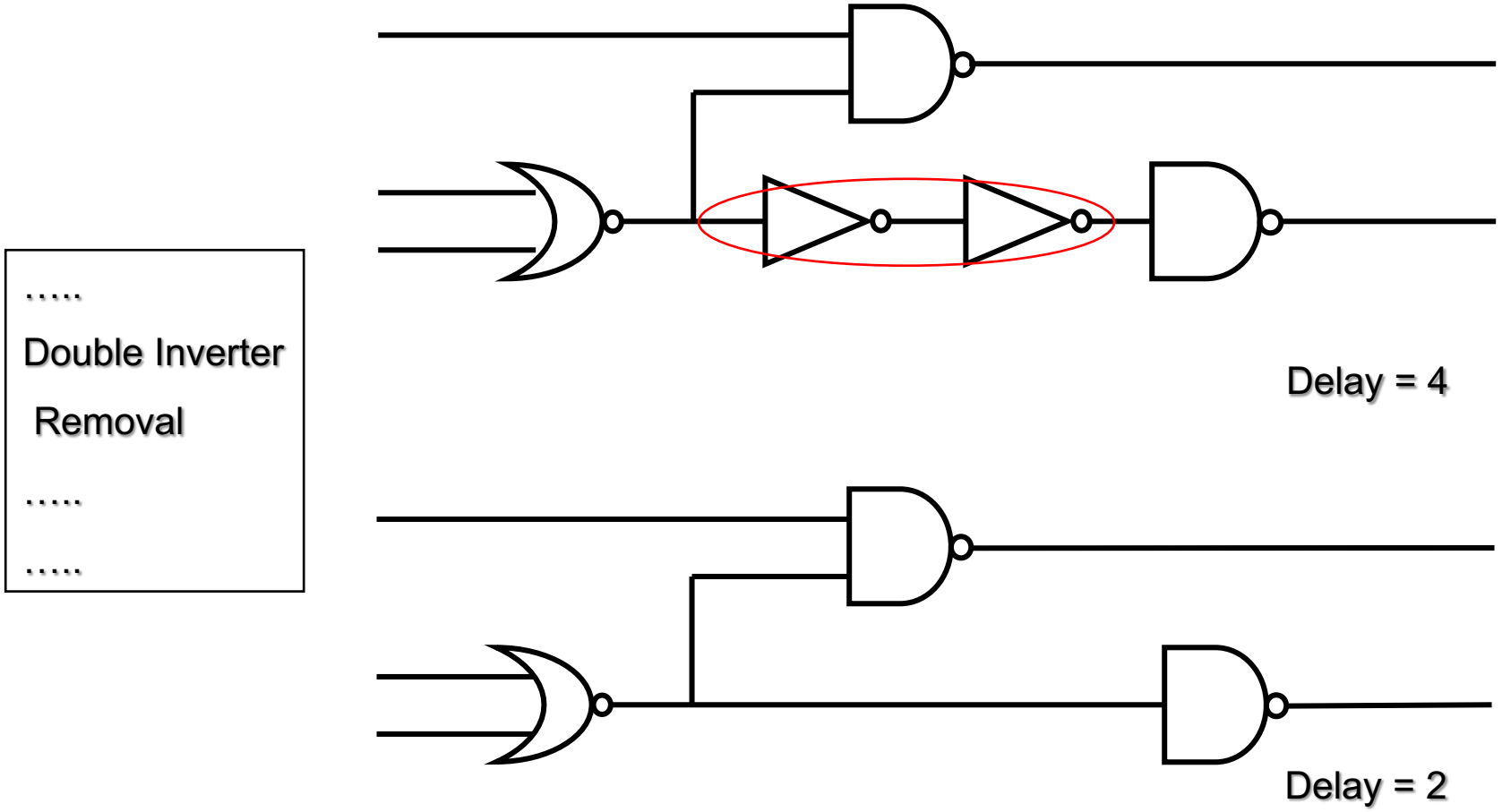


Performance Optimization

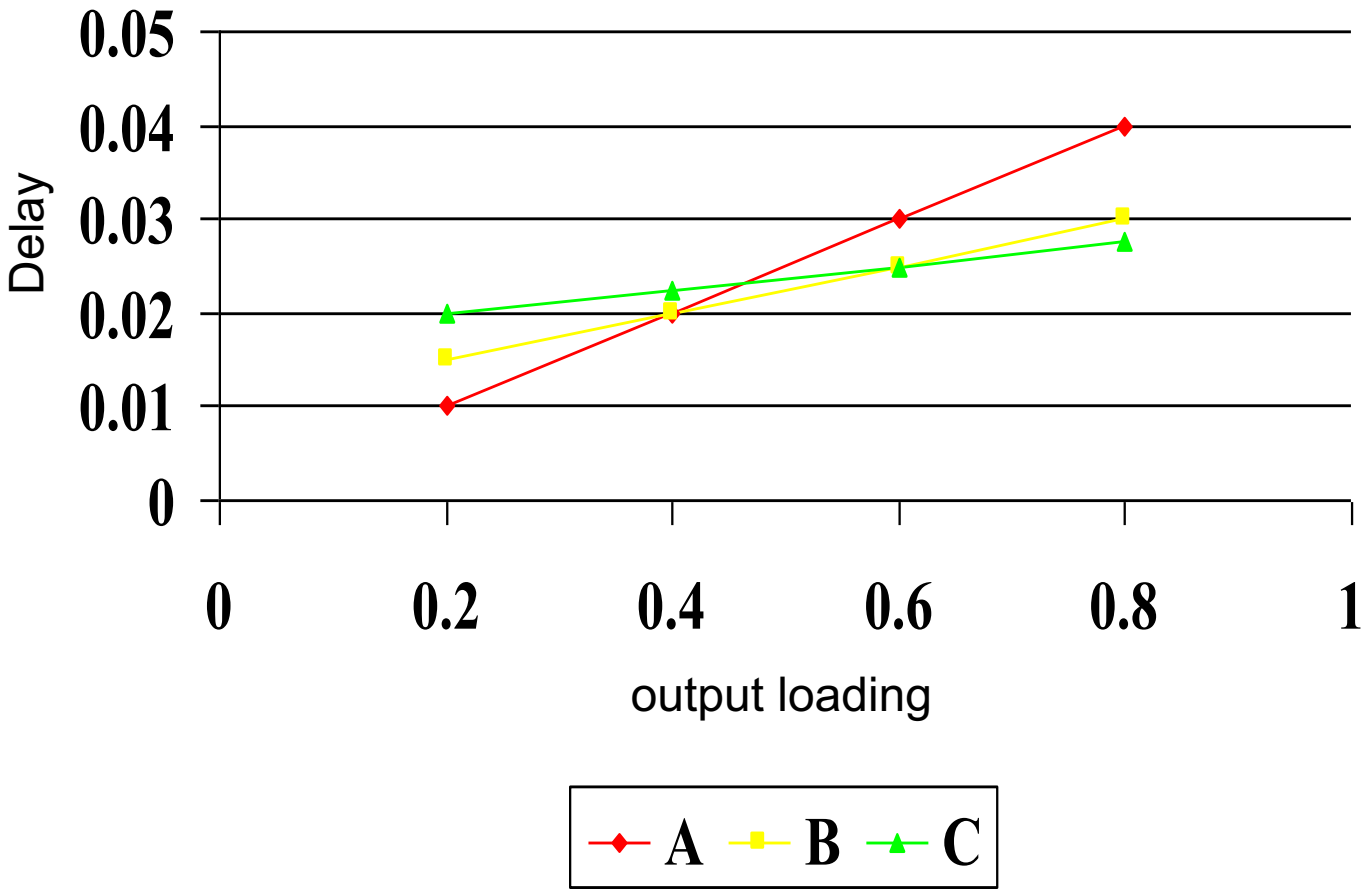
Performance Optimization

- Driven by STA
  - Incremental performance analysis
- Fix electrical violations
  - Resize cells
  - Buffer nets
  - Copy (clone) cells
  - Vt swap, Lgate swap, etc.
- Fix timing problems
  - Local transforms
  - Path-based transforms

Transform Example



Resizing



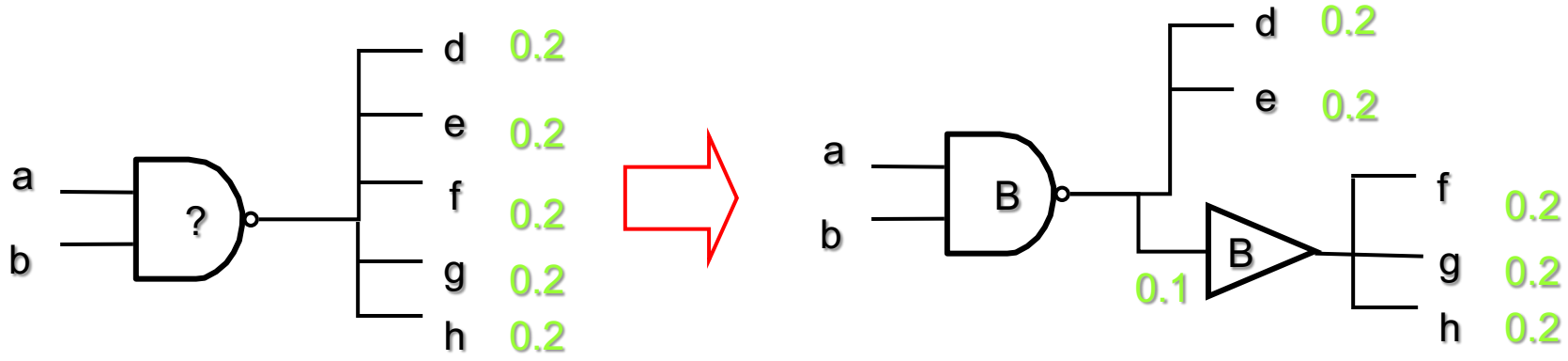
Gate size: C > B > A

- For loading 0.8, it is better to use gate C
- For loading 0.2, it is better to use gate A

Cloning

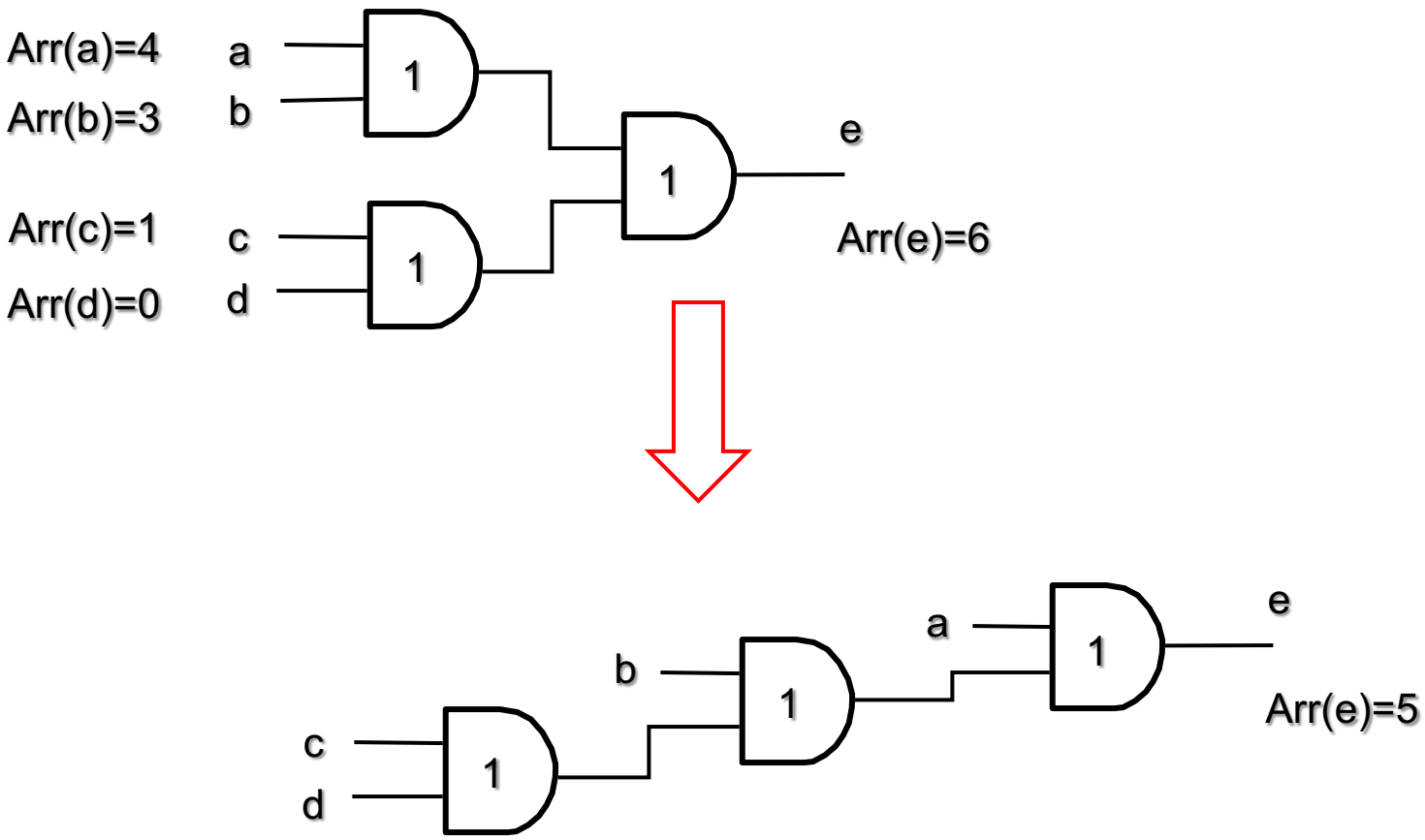
Note: especially if sinks d,e are located in a different region than sinks f,g,h

Buffering

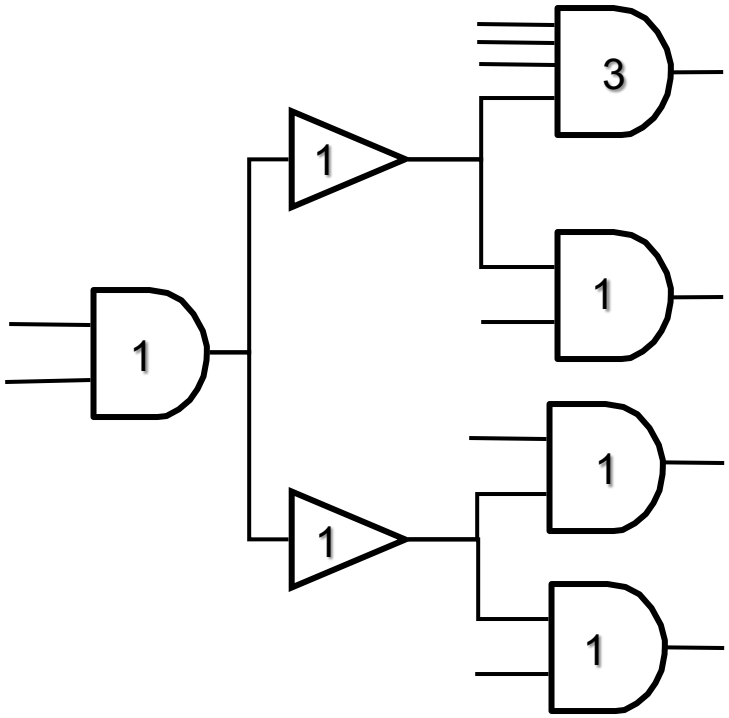


Note: especially if sink d or e is a “critical sink”

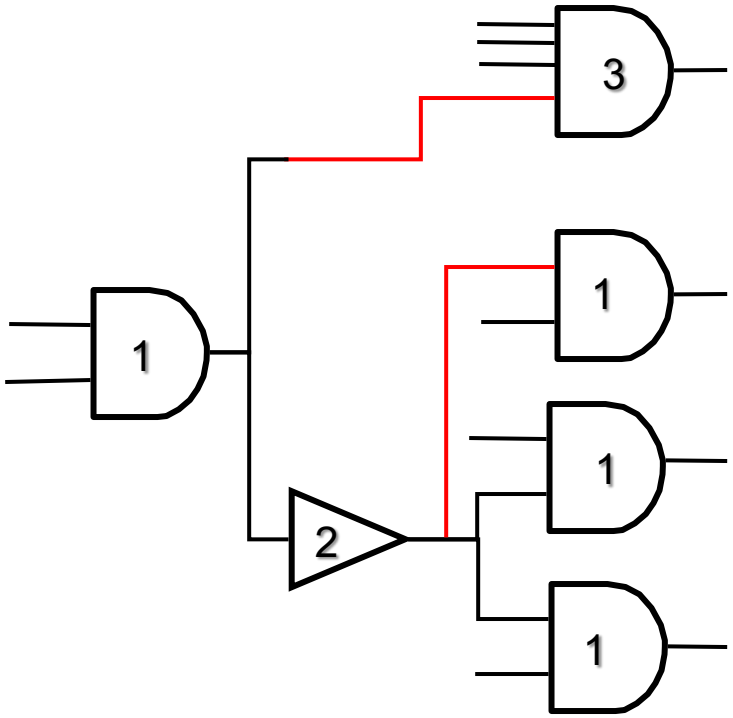
Redesign Fan-in Tree



Redesign Fan-out Tree



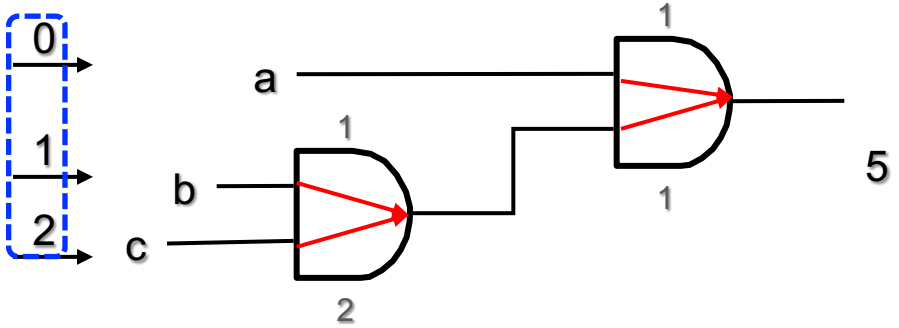
Longest Path = 5



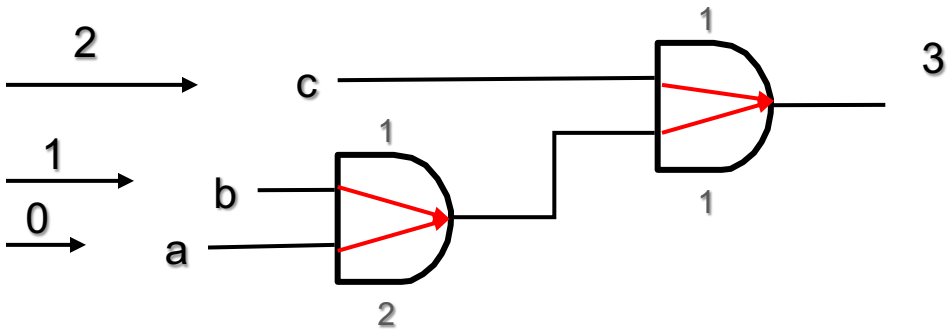
Longest Path = 4
Slowdown of buffer (so 1->2) due to load

Swap Commutative Pins

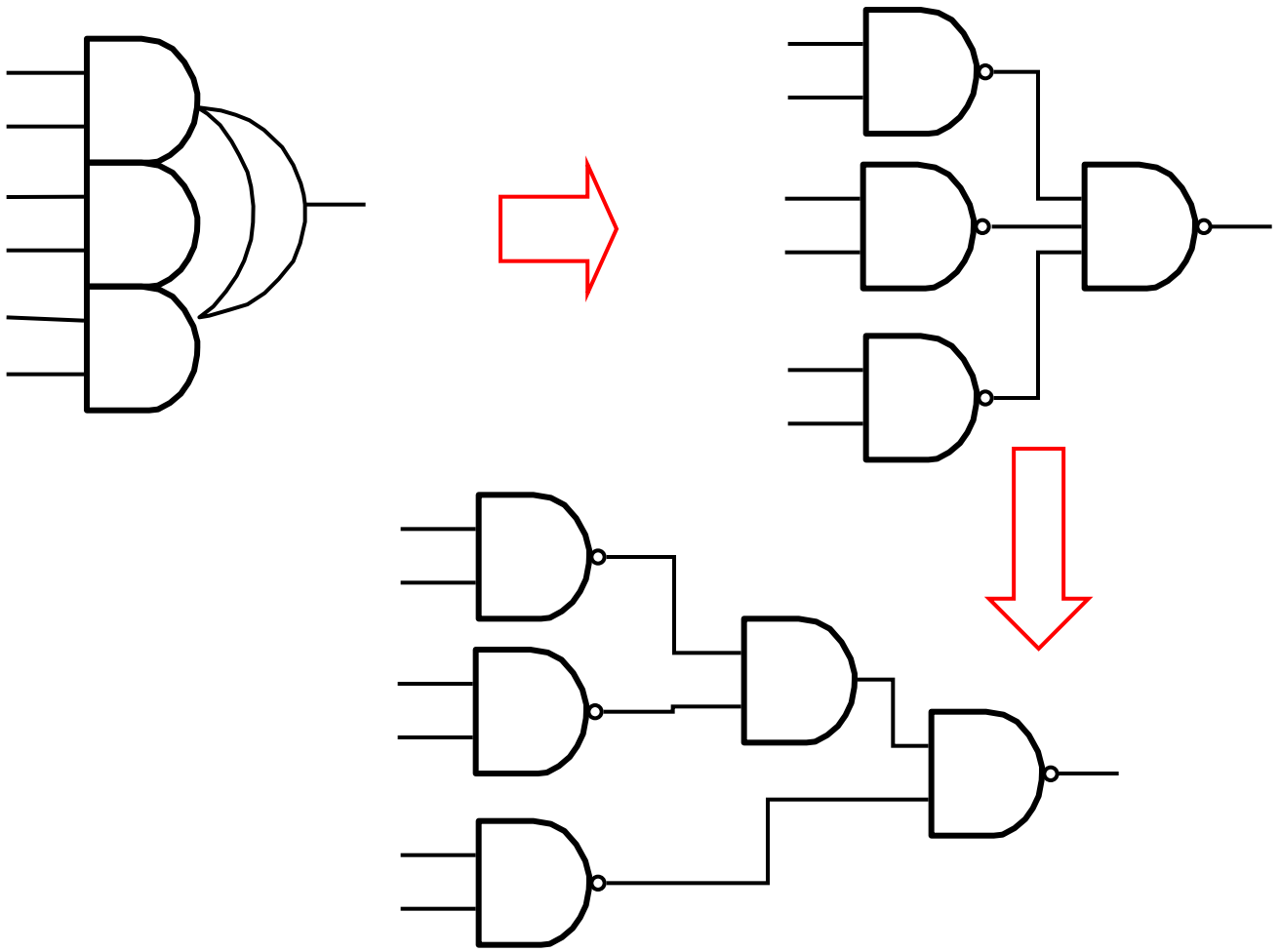
input arrival time



Simple sorting on arrival times and delay works



Decomposition



Knobs for Power/Delay Optimization

- Effective approach to power, delay optimization
- Objective: minimize total/leakage power
  - Satisfy constraints: slack, slew, max load capacitance, …
  - **Tunable** cell/gate parameters: width, Vth, channel length (L_{gate})
  - Select a proper library cell for each gate

gate-width (drive-strength)	INVX2	INVX4	INVX8	INVX16
multi-Vth	HVT	NVT	LVT	
L_{gate}-bias	L=65nm	L=60nm	L=55nm	

lower (leakage) power ⟷ *higher (leakage) power*

lower speed ⟷ *higher speed*