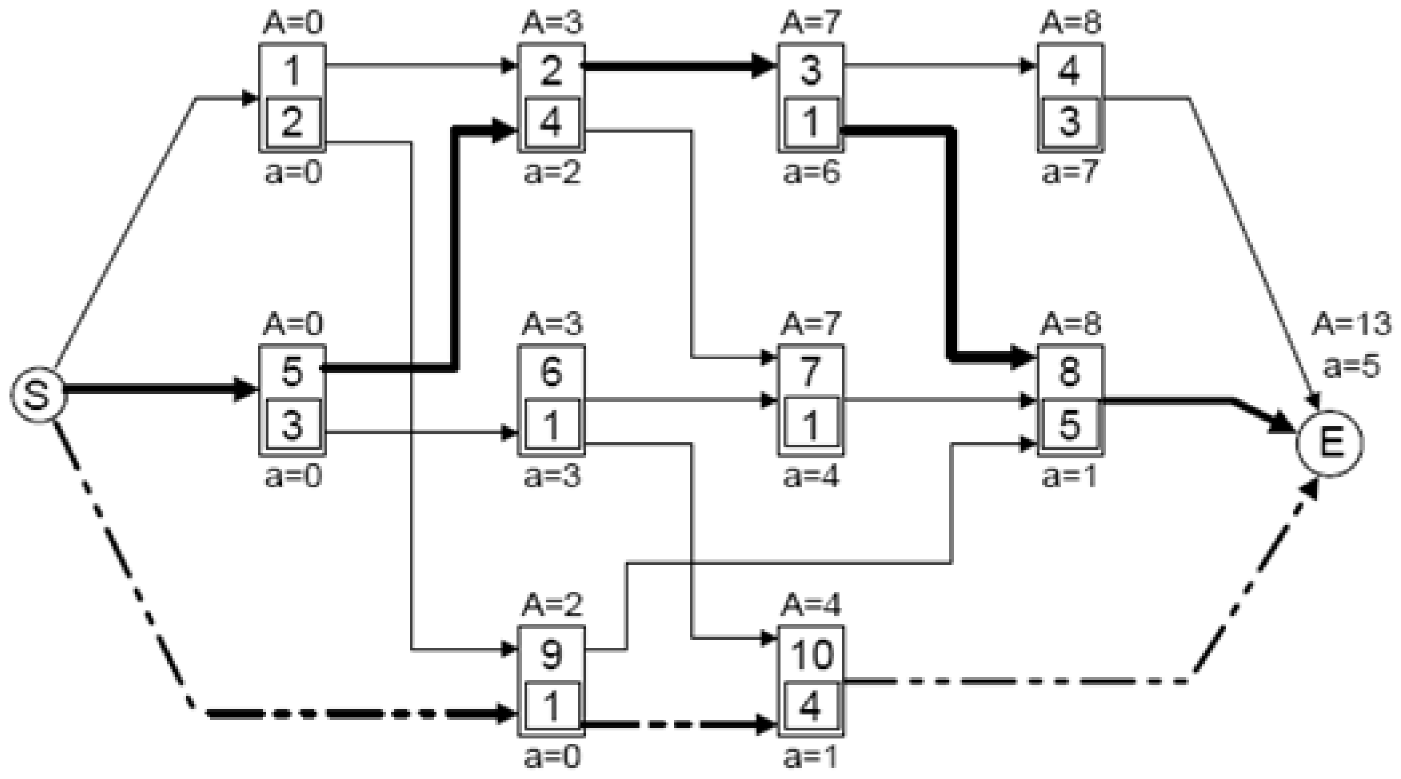
Unit 8: Timing

- Course contents:
  - Delay model
  - Static timing analysis



Function vs. Performance

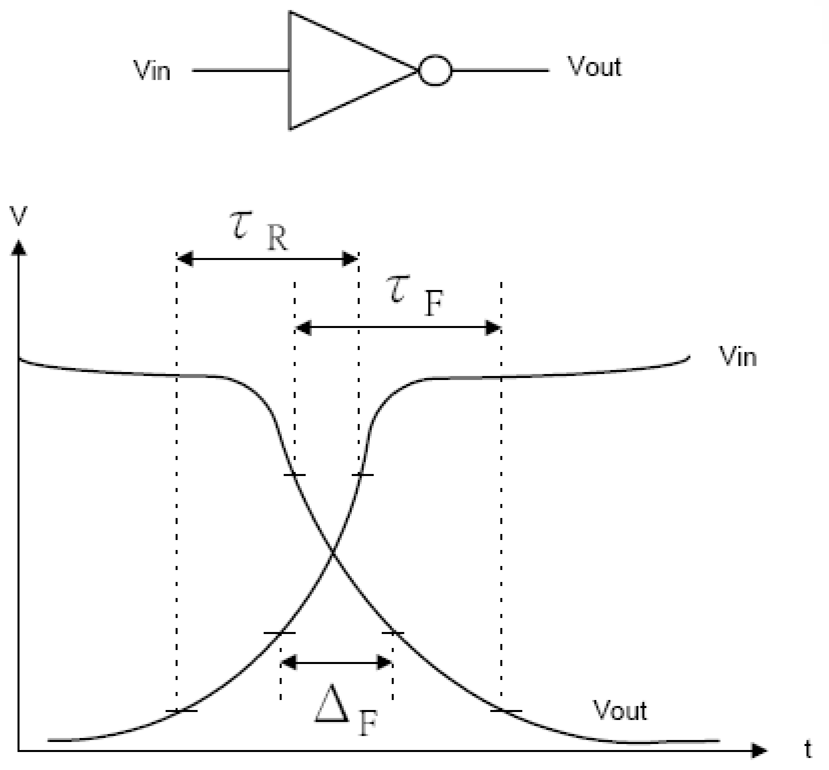
- Function verification
  - System correctly implements specified function
  - Checked by formal or informal ways (logic, RTL, behavior)
    - Simulation is the most popular one
- Performance verification
  - System correctly implements specified function at specified speed
  - Can be checked by simulation, but …

Performance Verification

- The simulation (dynamic) approach
  - Pattern-dependent
  - Accurate
  - Incomplete coverage
  - Slow
- The timing analysis (static) approach
  - Functionality assumed correct
  - Pattern-independent timing check
  - Complete coverage
  - Fast
  - Maybe inaccurate due to false paths

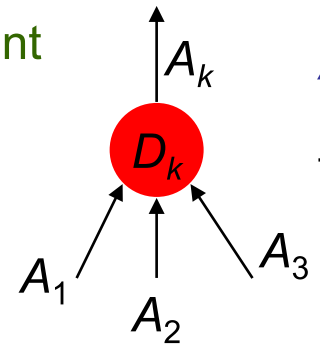
Cell Delay

- Definition
  - Propagation: Δ_R, Δ_F
  - Switching: τ_R, τ_F
- Common practice
  - Simulation based
    - Spice simulation
  - Input waveforms
    - Ramp input
    - Exponential input
  - Data fitting
    - Input slope, output capacitance
    - Linear vs. nonlinear



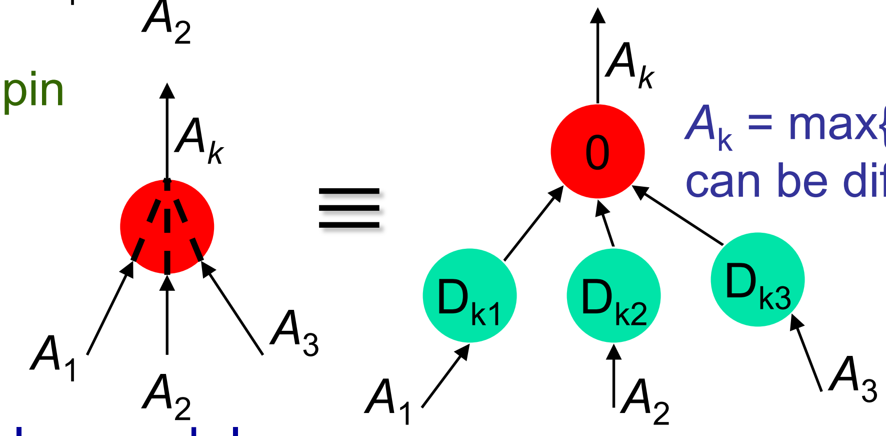
Cell Delay Model

- Cell model
  - Gate delay model
    - Constant



A_k = arrival time = max(A_1,A_2,A_3) + D_k
D_k = delay at node k, parameterized by function f_k and node k's fanout

    - Pin-to-pin



A_k = max{A_1+D_{k1},A_2+D_{k2},A_3+D_{k3}}
can be different for rise from fall

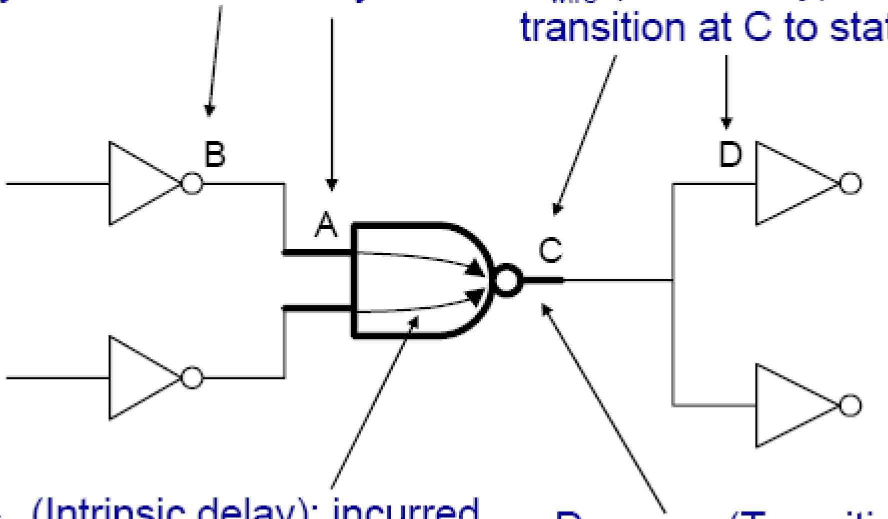
  - Library delay model
    - Use more accurate delay data given in the library
    - May use linear or nonlinear models
    - Table looked-up + interpolation/extrapolation

Linear Delay Model

Delay = Dslope + Dintrinsic + Dtransition + Dwire

D_{slope} (Slope delay): delay at input A caused by the transition delay at B

D_{wire} (Wire delay): time from state transition at C to state transition at D



$D_{intrinsic}$ (Intrinsic delay): incurred from cell input to cell output

$D_{transition}$ (Transition delay): output pin loading, output pin drive

Linear Delay Model

- Delay = $D_{slope} + D_{intrinsic} + D_{transition} + D_{wire}$
- Slope delay
  - The delay due to a slow logic transition at the input pin
  - $D_{slope} = D_{Tprevious} * S$ ($D_{Tprevious}$: previous-stage transition; S: slope sensitivity)
- Intrinsic delay
  - The built-in delay, fixed
- Transition delay
  - Output resistance times load
  - $D_{transition} = R_{drive} * (C_{pin} + C_{wire})$
- Wire delay
  - The time to propagate a logic transition through an interconnect network
  - $D_{wire} = R_{wire} * \boxed{(C_{pin} + C_{wire})}$

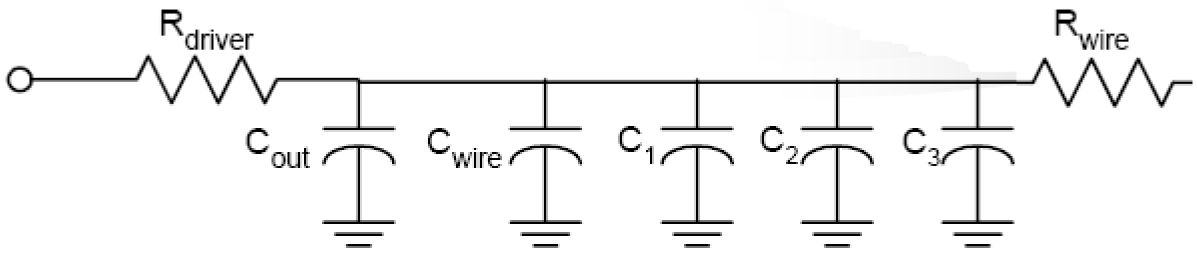
Downstream capacitance depends on interconnect model

Wire Delay Model

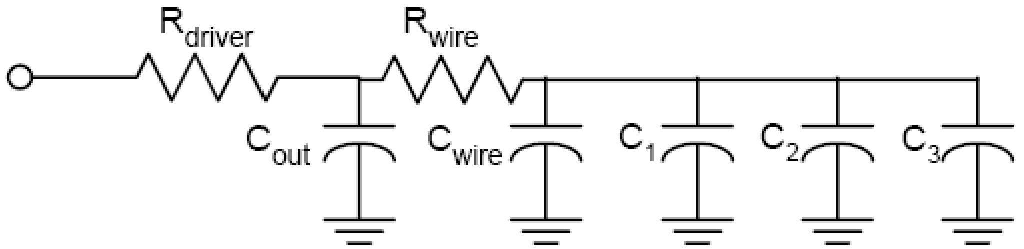
- Wire load model
  - Unit fanout delay model
    - Incorporate an additional delay for each fanout
  - RC model
    - Calculate delay according to physical information (distance, loading, etc.)

Interconnect Models

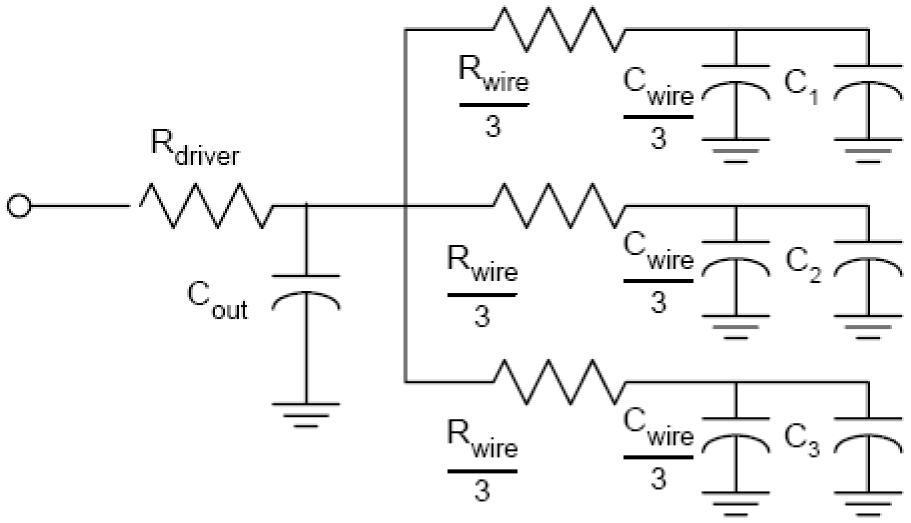
- Best-case RC tree



- Worst-case RC tree



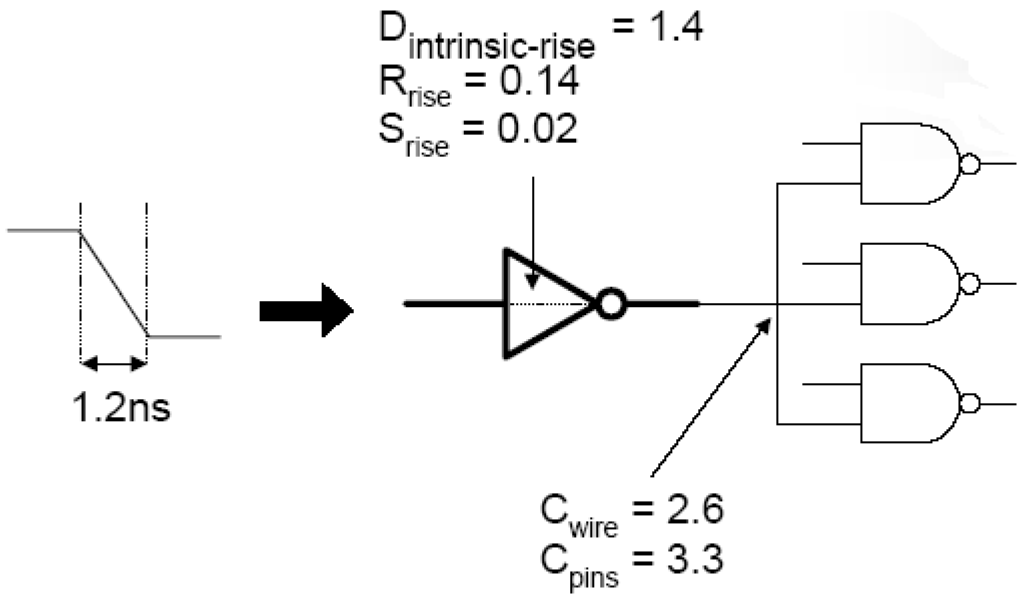
- Balanced-case RC tree



Example: Linear Delay Calculation

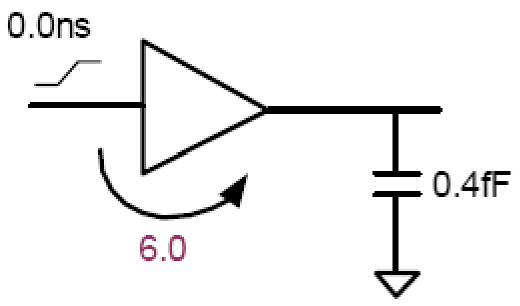
- Assume an inverter input is driven by a falling signal as follows, based on best-case RC tree, we have

$$D_{rise} = D_{slope} + D_{intrinsic} + D_{transition} + D_{wire}$$
$$= (1.2)(0.02) + 1.4 + (0.14)(2.6 + 3.3) + 0$$
$$= 2.25$$



Tabular Delay Model (NLDM)

- Delay values are obtained by a look-up table
  - Two-dimensional table of delays (m by n)
    - W.r.t input slope (m) and total output capacitance (n)
  - Each value in the table is obtained by Spice simulation



Total Output Load (fF)

Input Transition (ns)	0.2	0.3	0.4	0.5
0	3	4.5	6	7
0.1	5	8	10.7	13

Cell Delay (ps)

- Can be more precise than linear delay model
  - Table size ↑ ➔ accuracy ↑
  - Interpolation required for outbounded values
- Require more space to store the table (.lib format)

Operating Conditions (PVT Variations)

- Cases: best, typical, worst
- Process variation
  - Treated as a straight percentage variation in any performance calculation
- Voltage variation
  - Speed increased with higher voltage
- Temperature variation
  - Delay increased with higher temperature

Arrival Time and Required Time

- Arrival time: the time signal arrives
  - Calculated from input to output
- Required time: the time signal must be ready
  - Calculated from output to input
- Slack = required time – arrival time
  - Timing flexibility margin

A(j): arrival time of signal j

R(k): required time or for signal k

S(k): slack of signal k

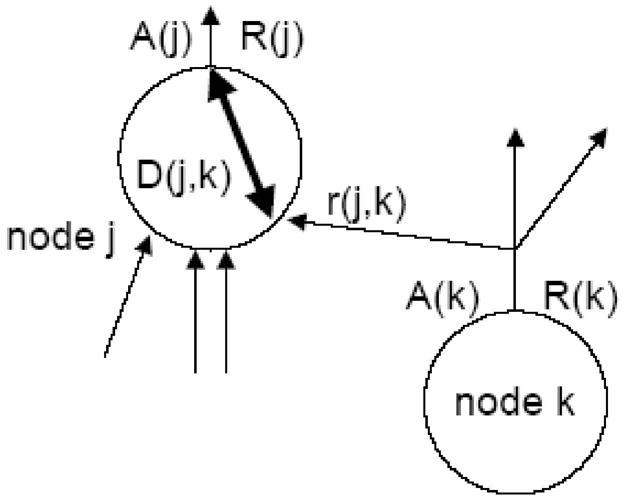
D(j,k): delay of node j from input k

$A(j) = \max_{k \in FI(j)} [A(k) + D(j,k)]$

$r(j,k) = R(j) - D(j,k)$

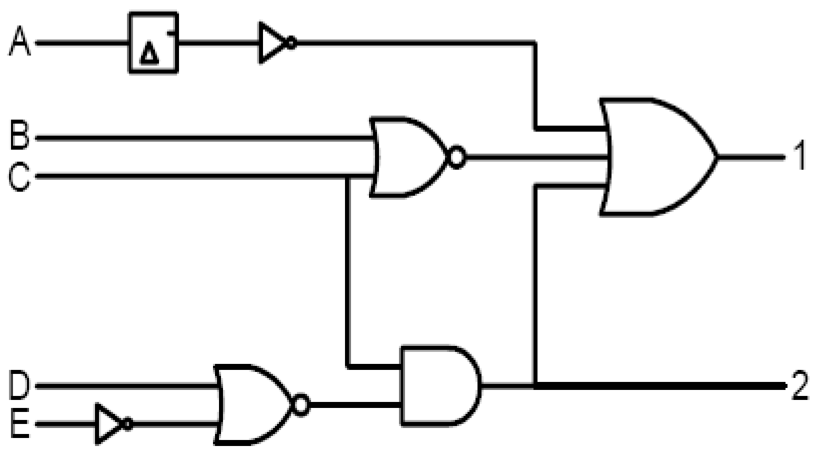
$R(k) = \min_{j \in FO(k)} [r(j,k)]$

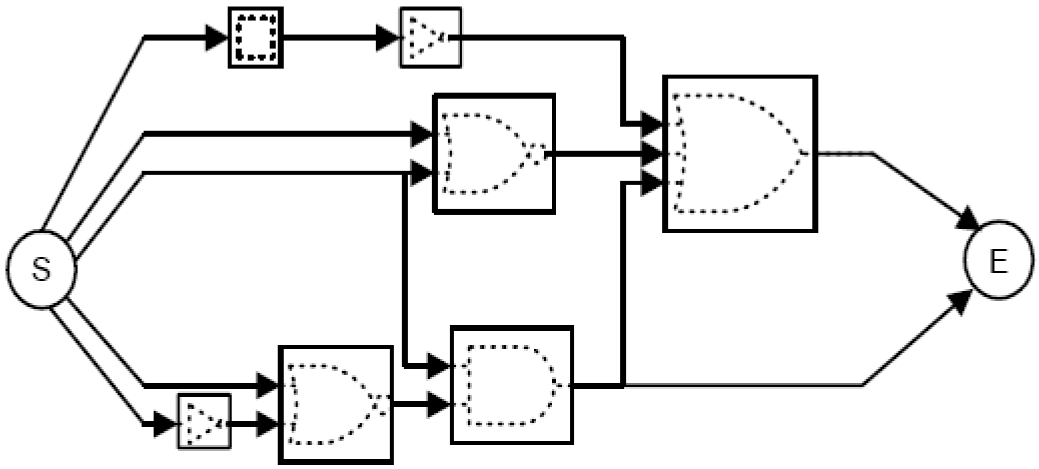
$S(k) = R(k) - A(k)$



Delay Graph

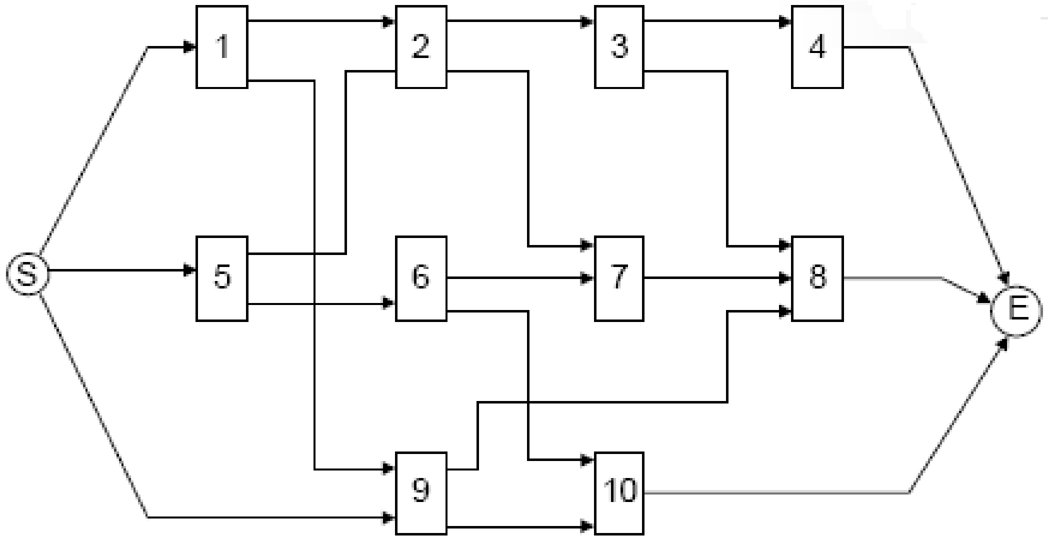
- Replace logic gates with delay nodes
- Add start (S) and end (E) nodes
- Indicate signal flow with directed arcs

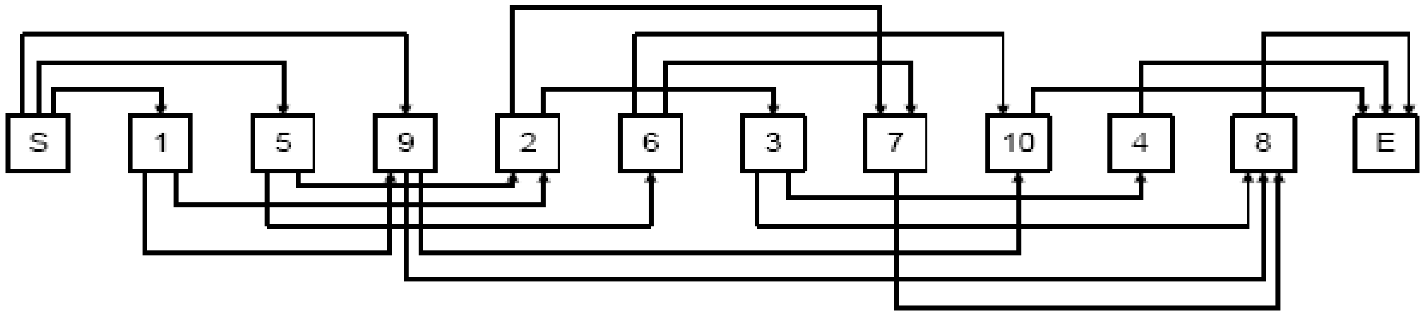




Delay Graph and Topological Sort

- Extract the topological order from a delay graph



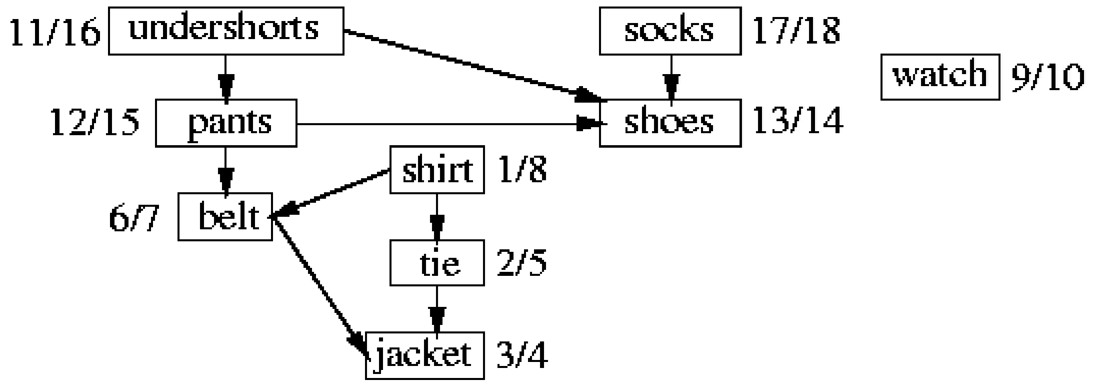


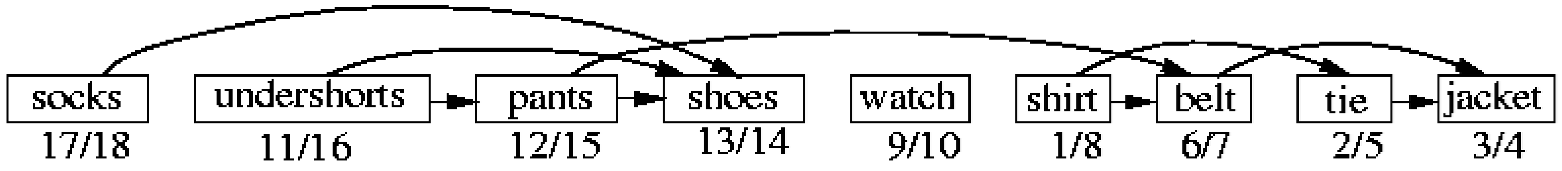
Topological Sort: A Quick Reivew

- A **topological sort** of a directed acyclic graph (DAG) *G* = (*V*, *E*) is a linear ordering of *V* s.t. (*u*, *v*) ∊ *E* □ *u* appears before *v*.

```
Topological-Sort(G)
1. call DFS(G) to compute finishing times f[v] for each vertex v
2. as each vertex is finished, insert it onto the front of a linked list
3. return the linked list of vertices
```

- Time complexity: O(V+E) (adjacency list).
- Correctness: Any edge (u, v) in a dag, we have f[v] < f[u].



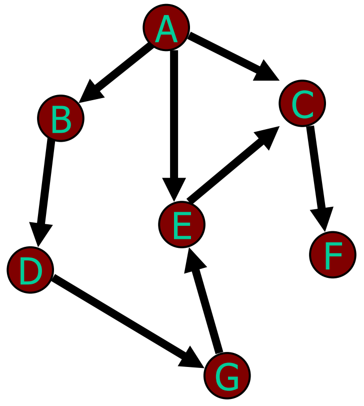


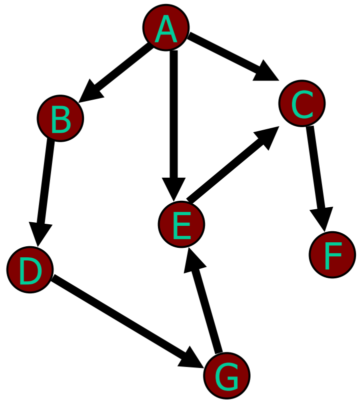
Vertices are arranged from left to right in order of decreasing finishing times.

Topological Sort

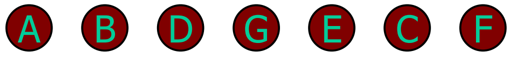
- A **topological sort** of a directed acyclic graph (DAG) $G = (V, E)$ is a linear ordering of V s.t. $(u, v) \in E$ □ u appears before v.
- A directed acyclic graph always contains a vertex with indegree 0.

```
Topological-Sort(G)
1. Compute indegree[v] for each vertex v ∈ V[G]
2. Q ← ∅
3. label ← 0
4. for each vertex v ∈ V[G] do
5.       if indegree[v] = 0 then
6.           Enqueue(Q, v)
7. while Q ≠ ∅ do
8.     u ← Dequeue(Q)
9.     label[u] ← label ← label+1
10.   for each v ∈ Adj[u] do
11.         indegree[v] = indegree[v]-1
12.         if indegree[v] = 0 then
13.             Enqueue(Q, v)
```





- Time complexity: $O(V+E)$ (adjacency list).



Static Timing Analysis

- Arrival times can be computed in the topological order from inputs to outputs
  - When a node is visited, its output arrival time is: **the max of its fanin arrival times + its own delay**
- Required times can be computed in the topological order from outputs to inputs
  - When a node is visited, its input required time is: **the min of its fanout required times – its own delay**

Topological Timing Analysis

- Procedure for computing the (longest) topological delay of a circuit

```
1.  Construct timing graph
2.  Sort nodes in topological order
3.  For each node in sorted order
        Compute (latest) arrival time $AT(v_i)$ as
        $\max_{v_j \in inputs(v_i)} \{AT(v_j) + D(v_j, v_i)\}$
```

NOTE: Above pseudo-code assumes pin-to-pin delay model.
If using load and slew rate dependent model, also need to propagate slew rates and use them along with loads to compute delay of each gate.
For transition-dependent delay model, propagate separate rising and falling arrival times

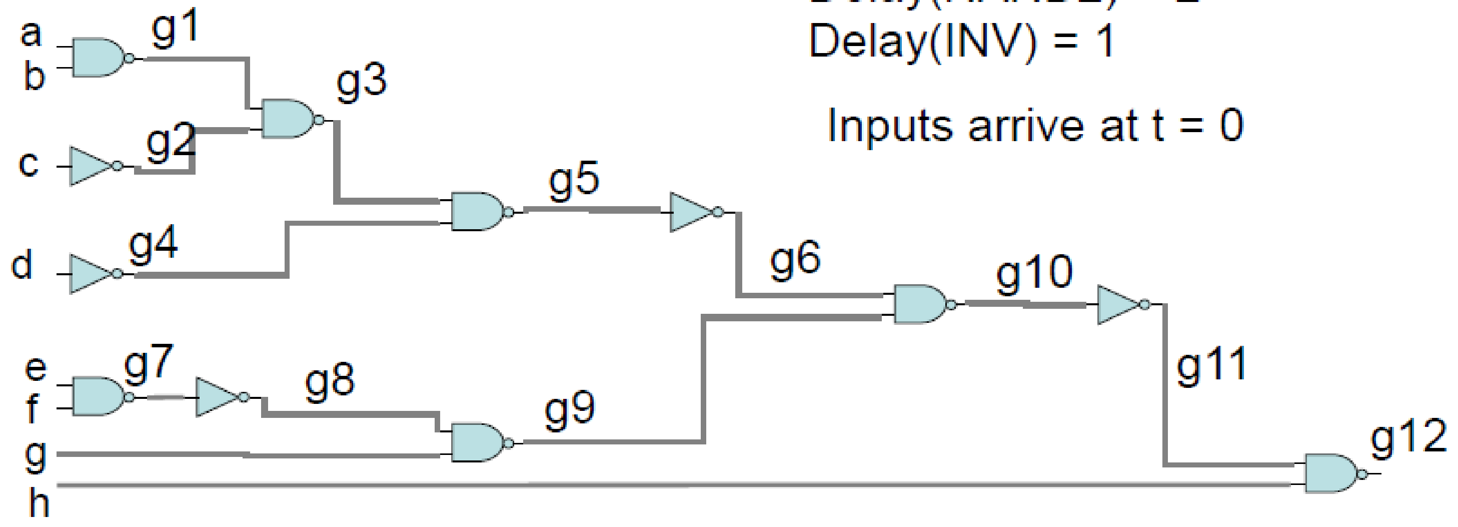
https://nanohub.org/resources/15542/download/2012.04.03-ECE595Z-L22.pdf

Topological Timing Analysis

- Example:

Delay(NAND2) = 2
Delay(INV) = 1

Inputs arrive at t = 0



Topological order:

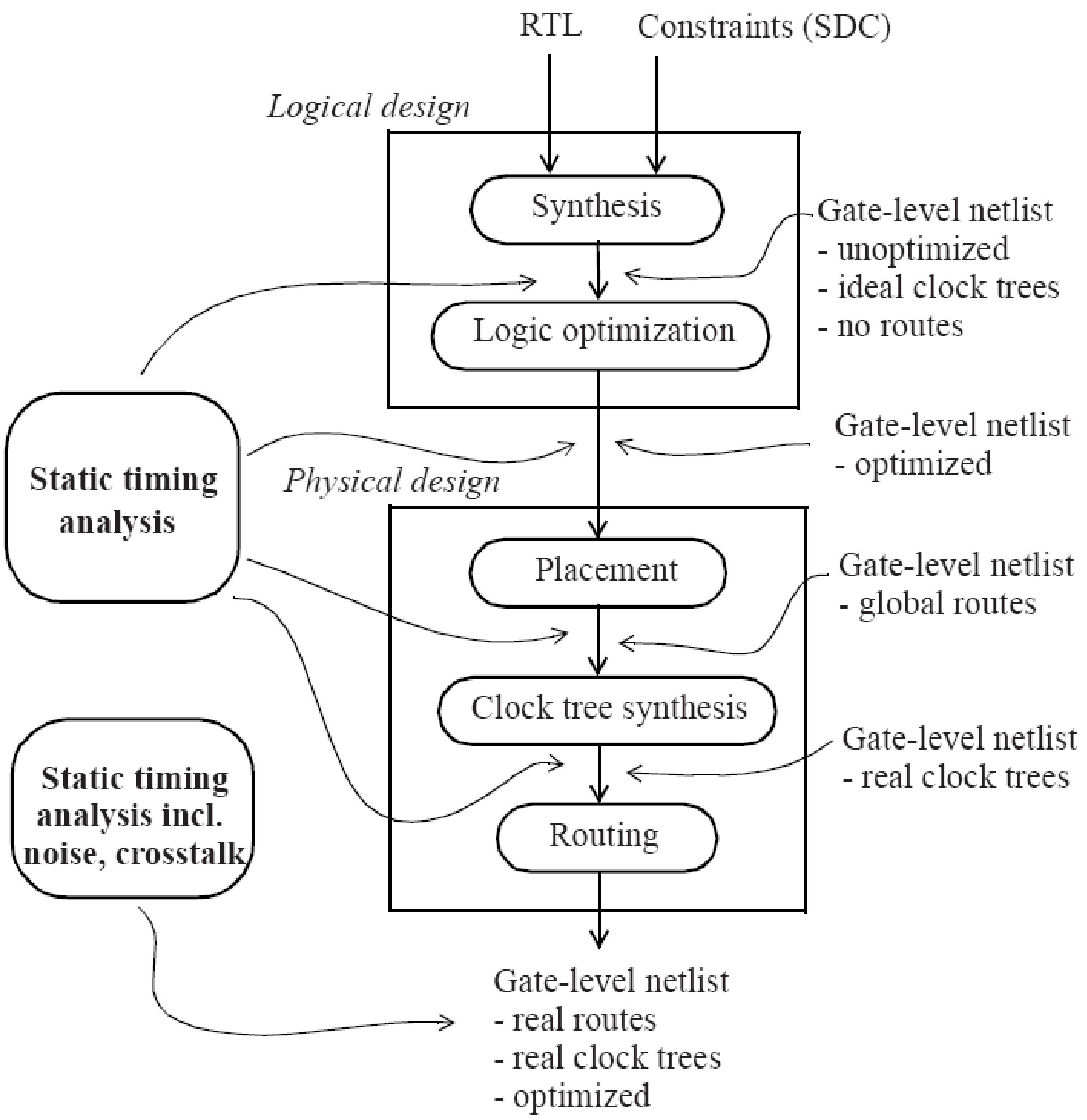
a, b, c, d, e, f, g, h, g1, g2, g4, g7, g3, g8, g5, g9, g6, g10, g11, g12

Latest Arrival Times:

a: 0, b: 0, c: 0, d: 0, e: 0, f: 0, g: 0, h: 0
g1: 2, g2: 1, g4: 1, g7: 2
g3: 4, g8: 3
g5: 6, g9: 5, g6: 7
g10: 9, g11: 10, g12: 12

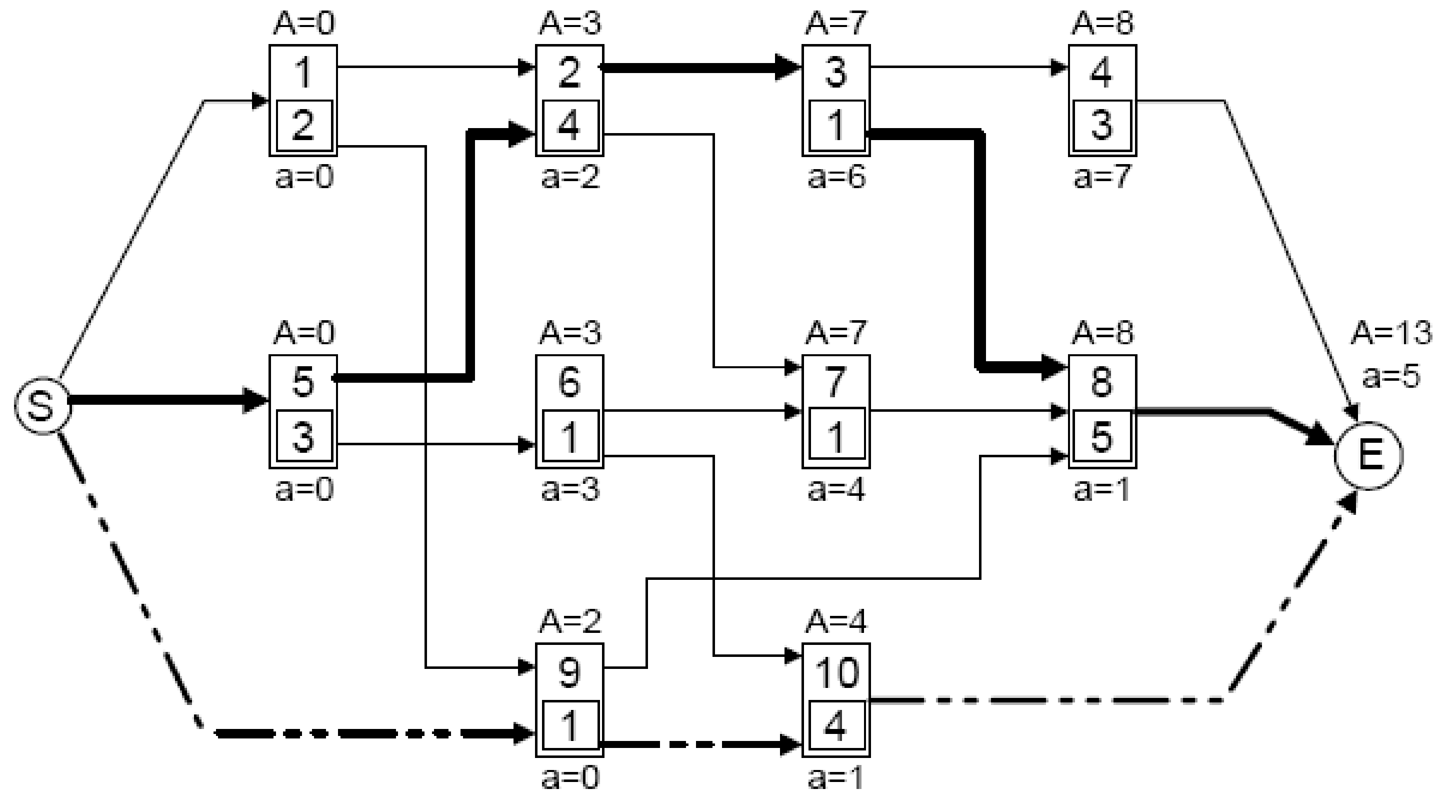
https://nanohub.org/resources/15542/download/2012.04.03-ECE595Z-L22.pdf

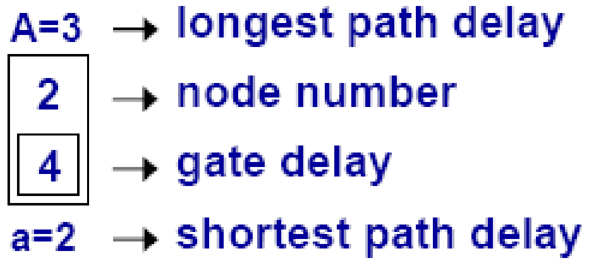
Static Timing Analysis in Design Flow



Source: Static Timing Analysis for Nanometer Designs, 2009

Delay Calculation





A=3 → longest path delay

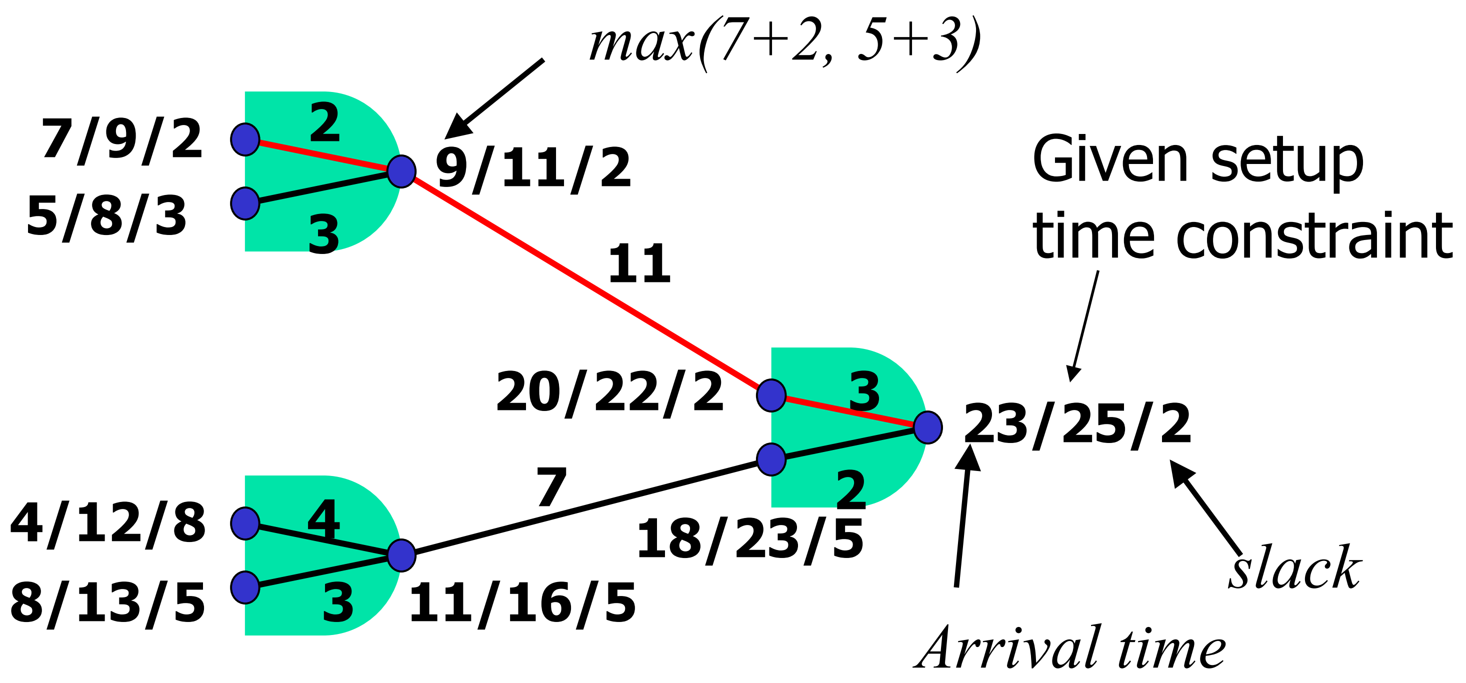
2 → node number

4 → gate delay

a=2 → shortest path delay

P.S: The longest delay and shortest delay of each gate are assumed to be the same.

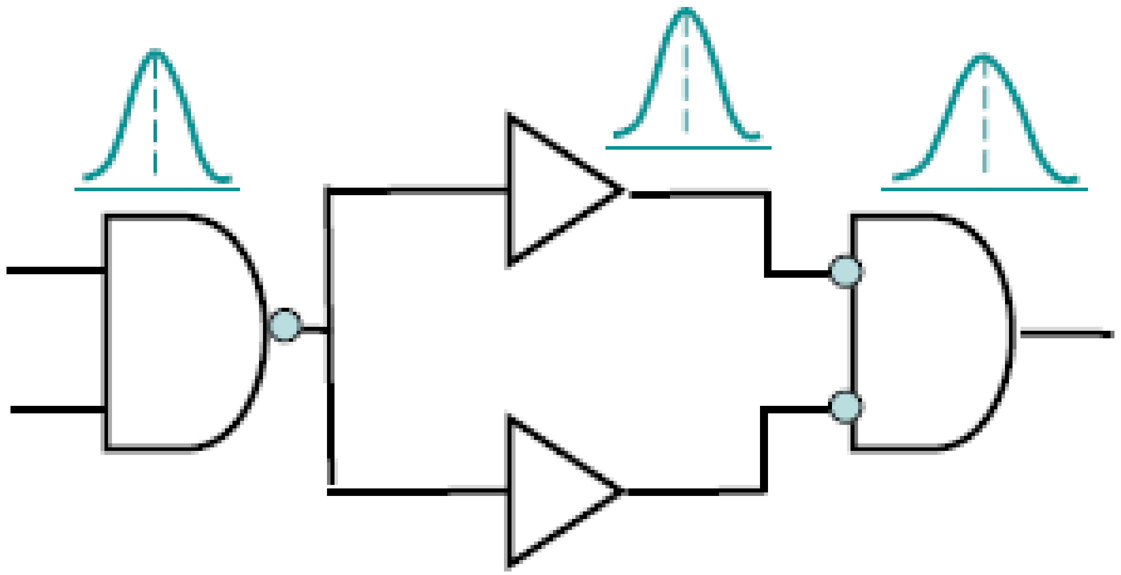
Delay Calculation: Another Example



- In STA, the basic operations are “max” and “+”
- This is a fixed-delay STA
  - Each cell pin-to-pin delays are pre-characterized
  - Interconnect delays are pre-calculated before STA
  - After STA, critical paths can be identified

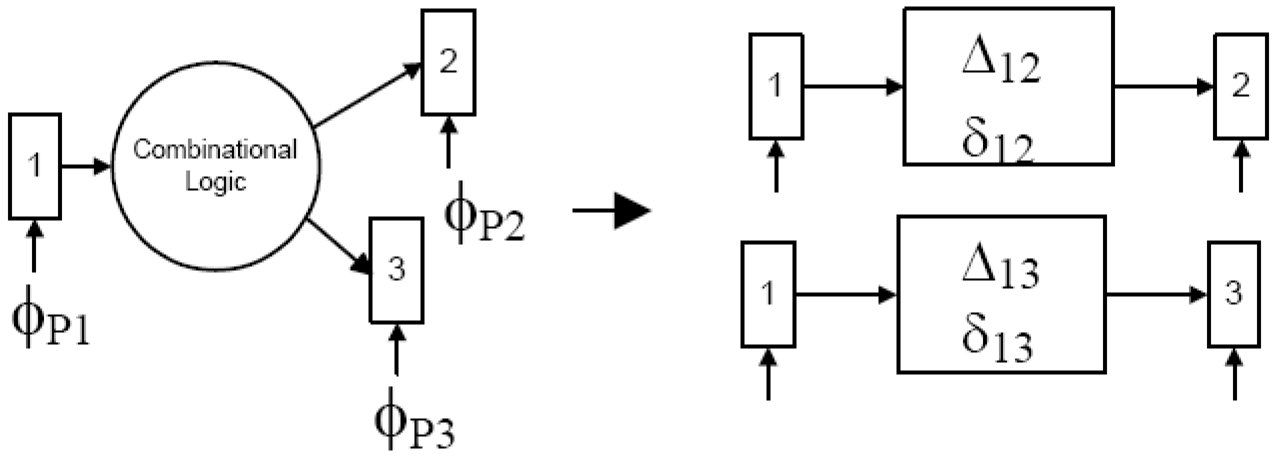
Statistical Static Timing Analysis

- Gate delays are in probabilistic distributions rather than fixed constants
- Aims to compute the statistical criticalities of delay paths
  - Avoid overestimating critical paths
  - Especially used in advanced technology nodes



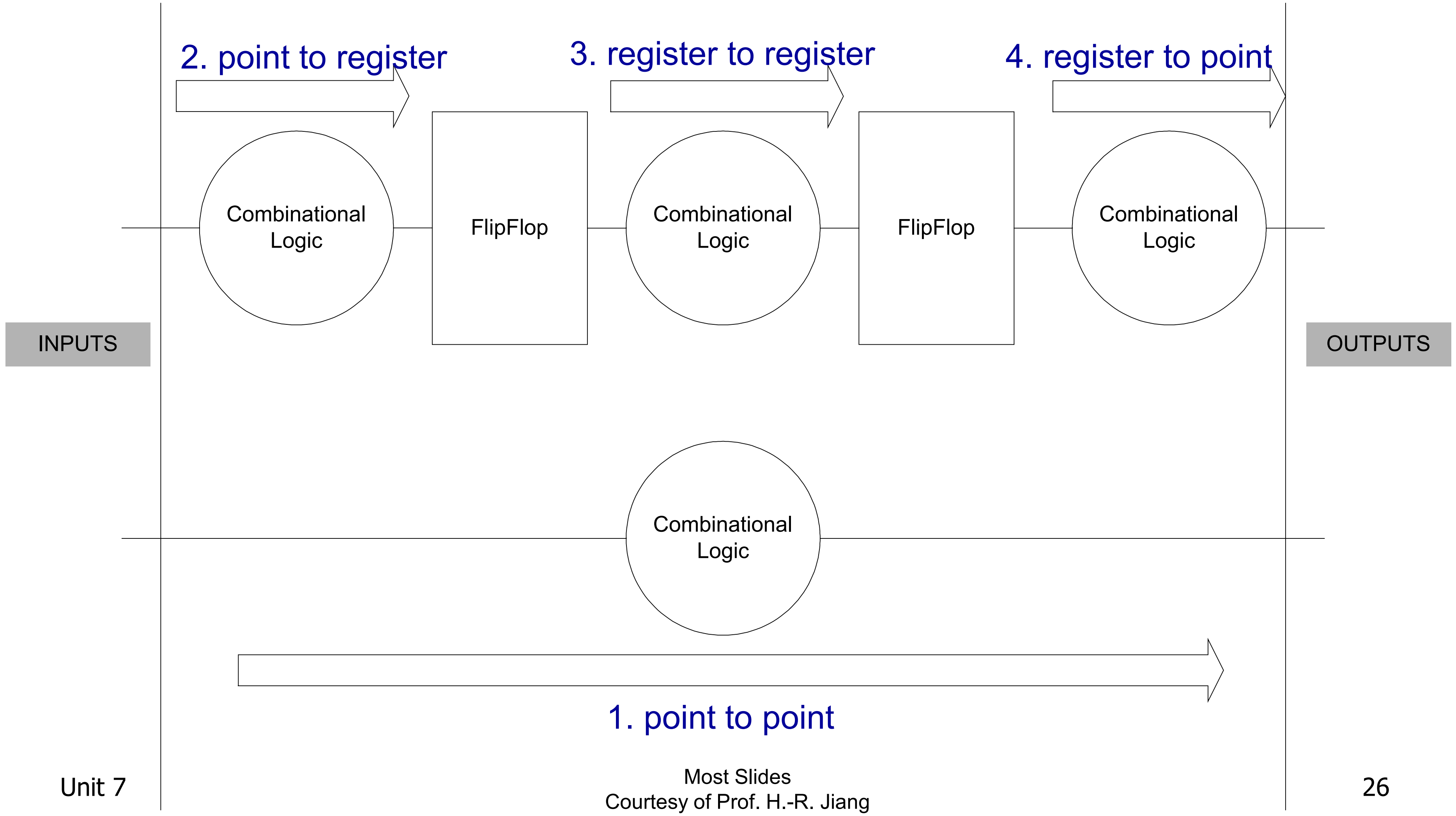
System Timing Model

- Allow three main components of digital logic systems to be treated separately
  - Clock distribution subsystem
  - Sequential elements
  - Combinational logic blocks
- Partitioning simplifies analysis for long-and-short path
  - Combinational logic to be analyzed as manageable acyclic (cyclic) graphs



Delay Path Types

- Begin points: inputs, register CK pins
- End points: outputs, register D pins

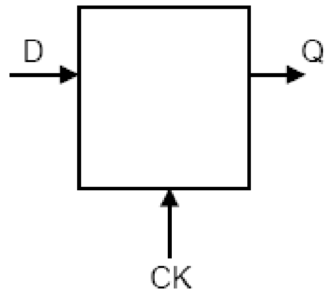


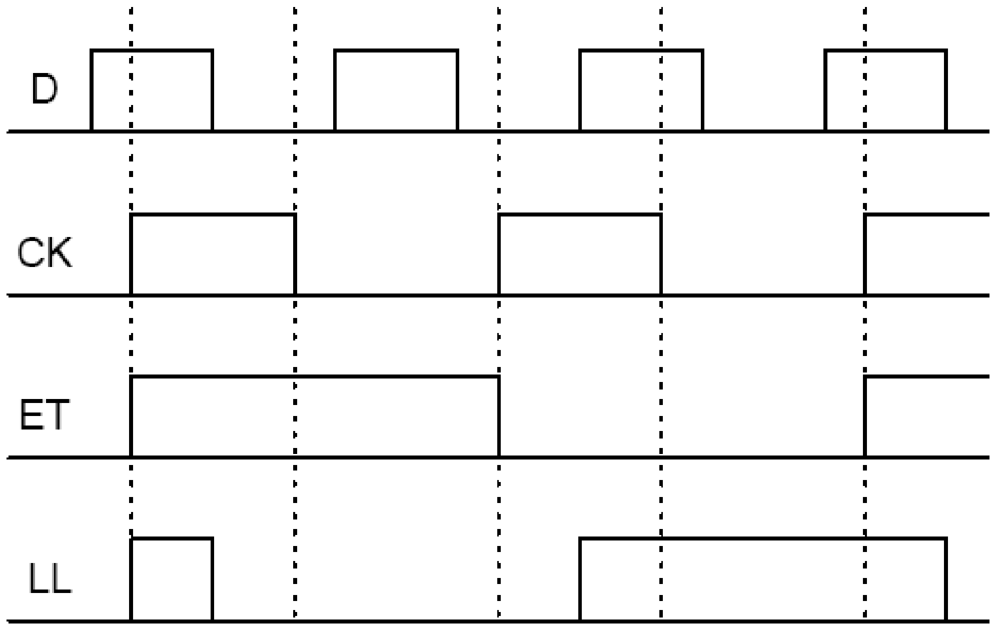
Launch & Capture Edges

- **Launch edge**: active clock edge to launch data
  - Triggering edge of a flip-flop
  - Open edge of a transparent latch
- **Capture (latch) edge**: active clock edge to capture data
  - Triggering edge of a flip-flop
  - Close edge of a transparent latch

Synchronizer

- Edge-trigger-rigid synchronizer
  - Same launch and capture edge
- Level-sensitive-soft synchronizer
  - Eg. Transparent latch
    - When the clock is high, an active-high transparent latch transmits the signal from D to Q as if it were a delay element
  - *Transparent* during its active period
  - Open edge: the edge when it starts to be active
  - Close edge: the edge when it closes



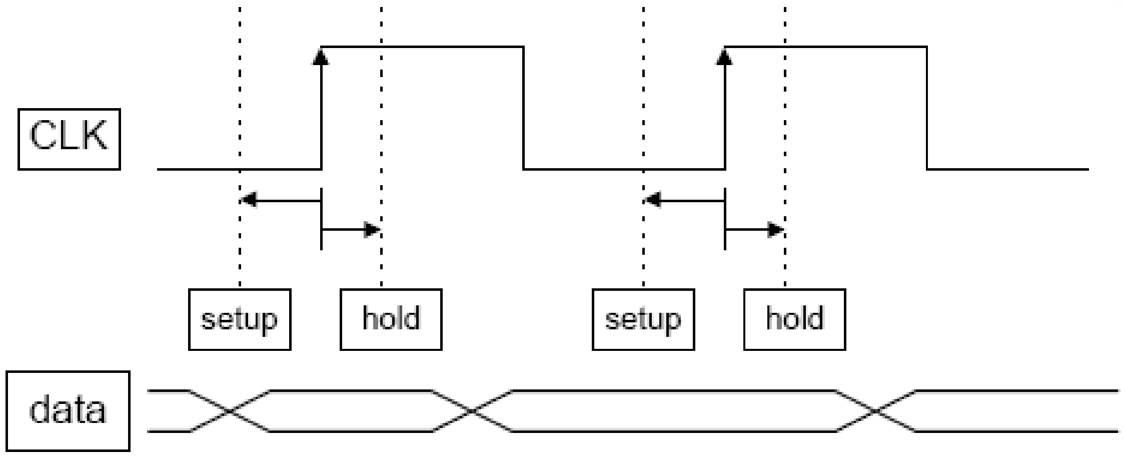


Setup & Hold Time

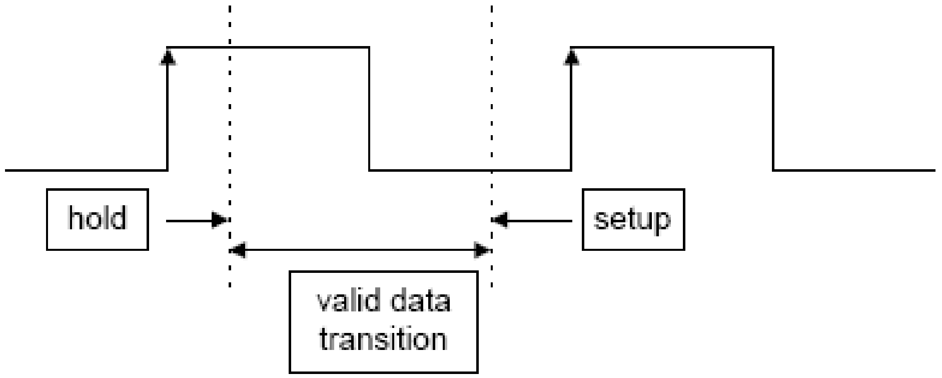
- **Setup time**: the length of time that data must stabilize before the clock transition
  - The longest data path is used to determine if the setup constraint is met
- **Hold time**: the length of time that data must remain stable at the input pin after the active clock transition
  - The shortest data path is used to determine if hold time is met

Setup & Hold Time

- Timing diagram

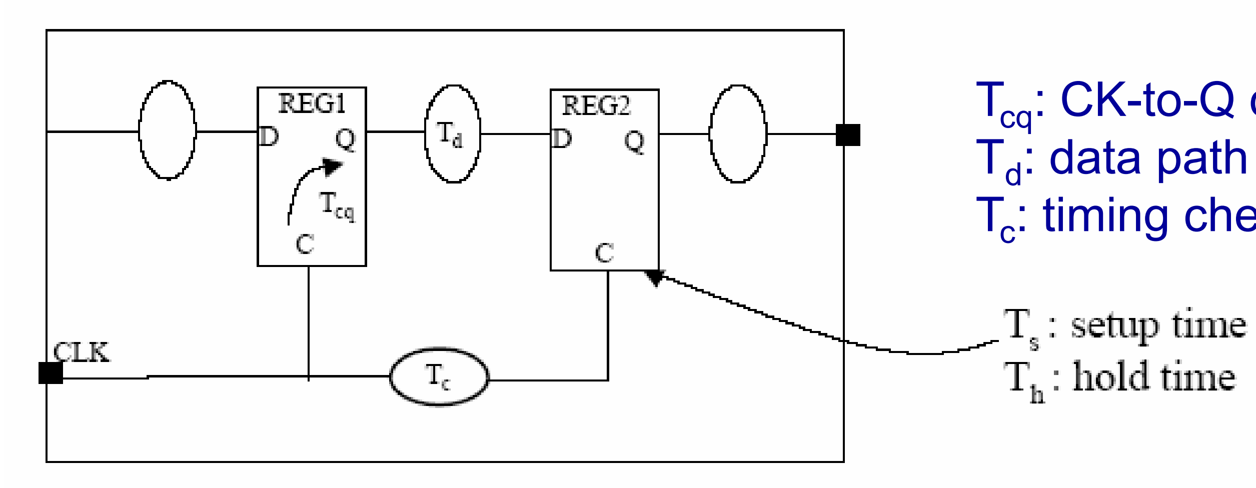


- Valid data transition



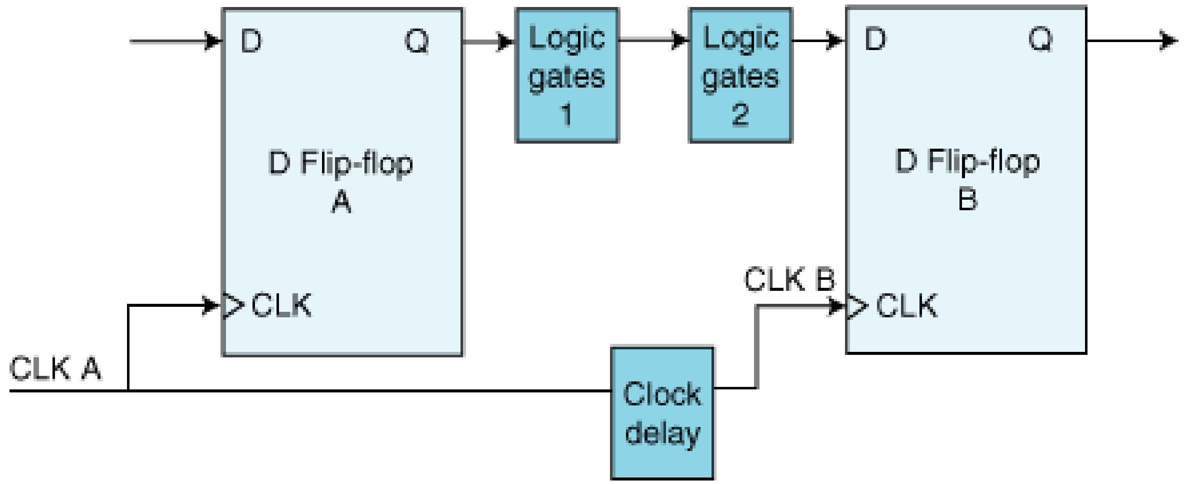
Setup & Hold Time Checks

- Setup capture edge: capture edge at an end point for a setup time check
  - $T_{cq} + \max T_d + T_s < T_{cycle} + \min T_c$
- Hold capture edge: capture edge at an end point for a hold time check
  - $T_{cq} + \min T_d > \max T_c + T_h$

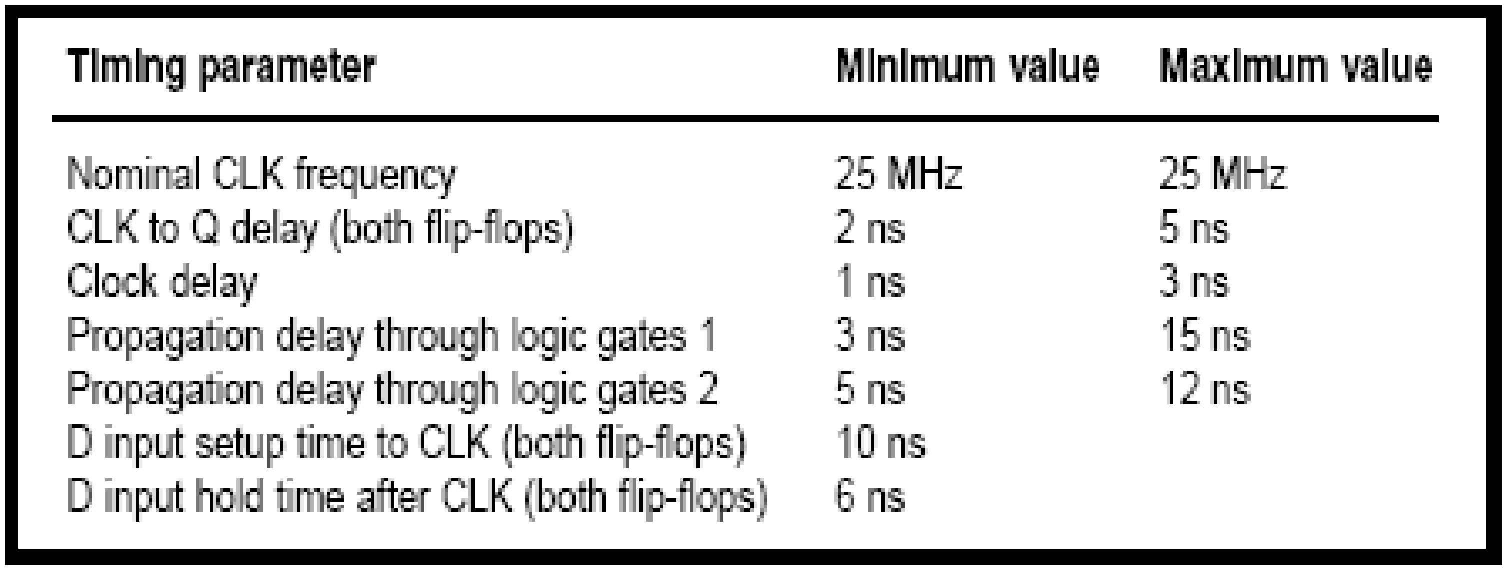


T_{cq}: CK-to-Q delay
T_d: data path delay
T_c: timing check

Setup & Hold Time Example (1/2)



Source: P. Nowe, "Timing (analysis) is everything," Circuit Cellar, Nov 2003

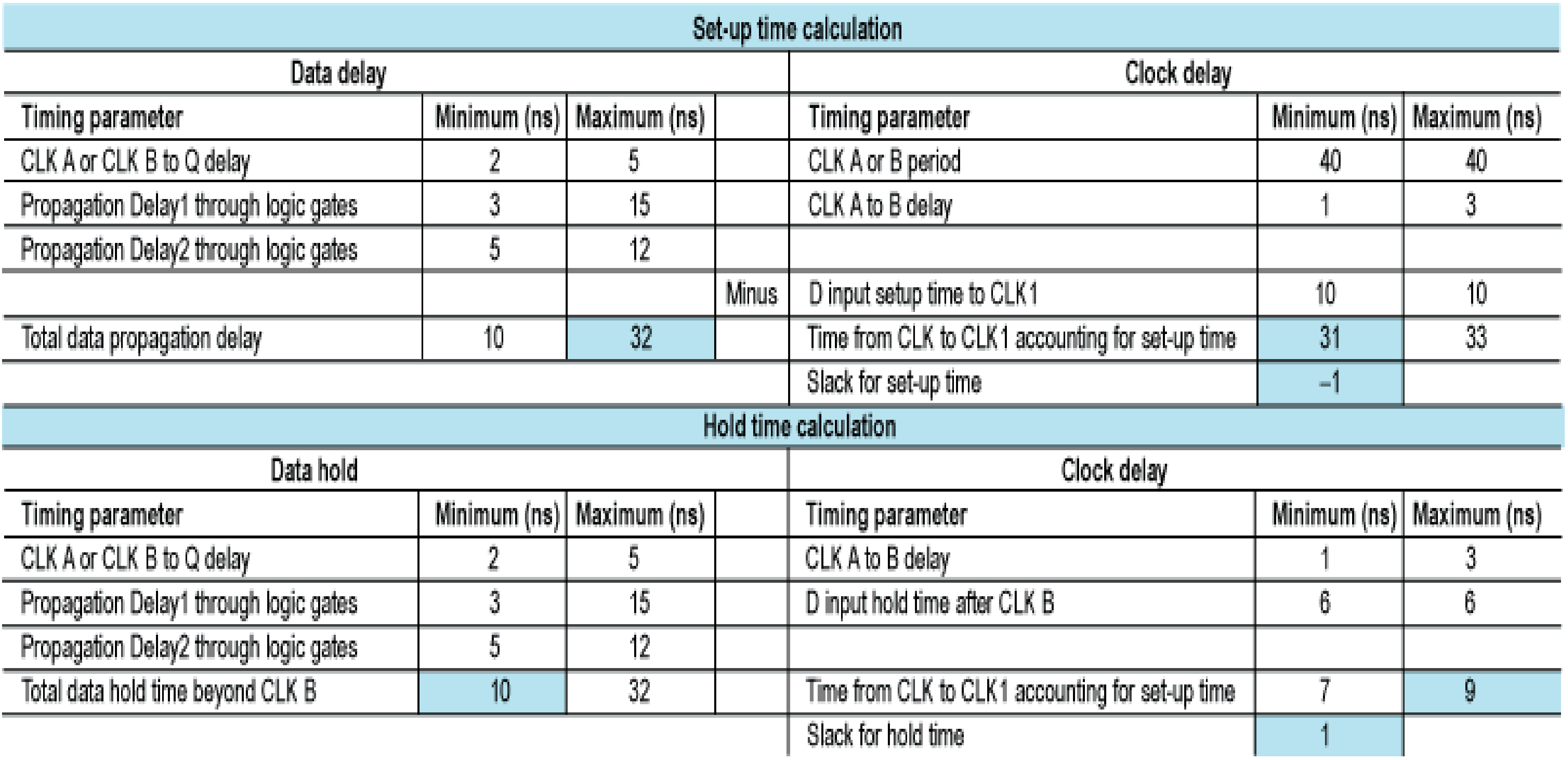


Timing parameter	Minimum value	Maximum value
Nominal CLK frequency	25 MHz	25 MHz
CLK to Q delay (both flip-flops)	2 ns	5 ns
Clock delay	1 ns	3 ns
Propagation delay through logic gates 1	3 ns	15 ns
Propagation delay through logic gates 2	5 ns	12 ns
D input setup time to CLK (both flip-flops)	10 ns	
D input hold time after CLK (both flip-flops)	6 ns	

Table 1—*Here are the timing values for the circuit illustrated in Figure 1.*

Setup & Hold Time Example (2/2)

Source: P. Nowe, "Timing (analysis) is everything," Circuit Cellar, Nov 2003

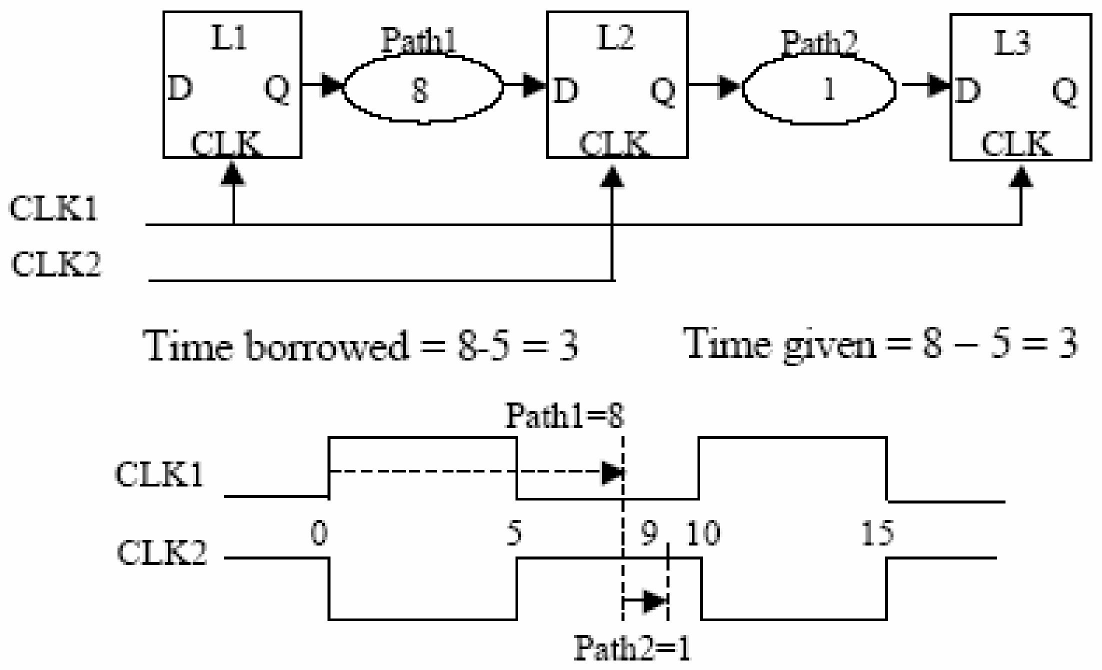


Set-up time calculation						
Data delay				Clock delay		
Timing parameter	Minimum (ns)	Maximum (ns)		Timing parameter	Minimum (ns)	Maximum (ns)
CLK A or CLK B to Q delay	2	5		CLK A or B period	40	40
Propagation Delay1 through logic gates	3	15		CLK A to B delay	1	3
Propagation Delay2 through logic gates	5	12				
			Minus	D input setup time to CLK1	10	10
Total data propagation delay	10	32		Time from CLK to CLK1 accounting for set-up time	31	33
				Slack for set-up time	–1	

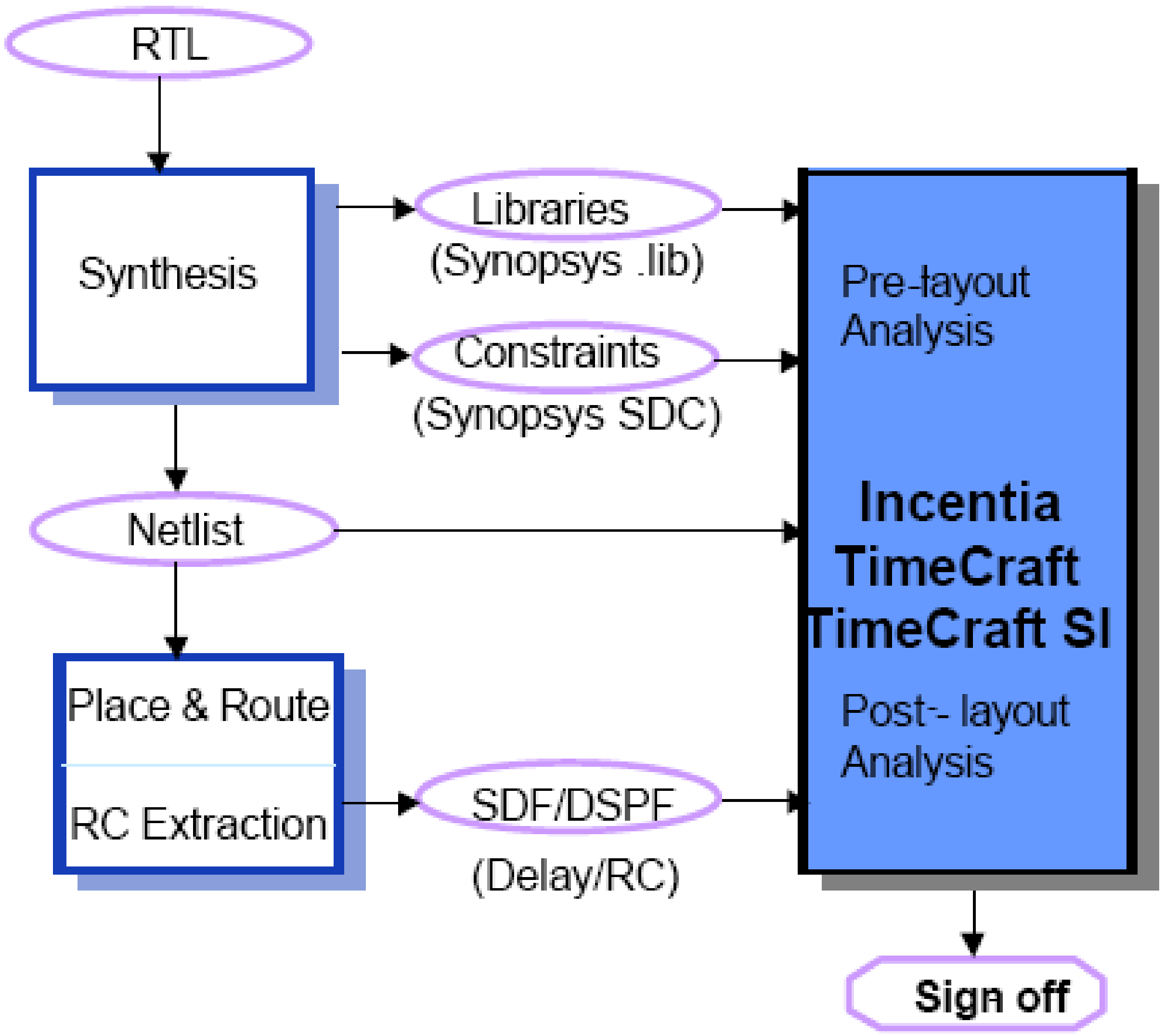
Hold time calculation						
Data hold				Clock delay		
Timing parameter	Minimum (ns)	Maximum (ns)		Timing parameter	Minimum (ns)	Maximum (ns)
CLK A or CLK B to Q delay	2	5		CLK A to B delay	1	3
Propagation Delay1 through logic gates	3	15		D input hold time after CLK B	6	6
Propagation Delay2 through logic gates	5	12				
Total data hold time beyond CLK B	10	32		Time from CLK to CLK1 accounting for set-up time	7	9
				Slack for hold time	1	

Time Borrowing

- Borrow time from the next stage (cycle stealing)
- Without time borrowing: timing violation at L2
- With time borrowing from L3 to L2: no timing violation
- Usually applied in transparent latch, but requires more sophisticated STA



Example: Timing Analysis Tool



Timing Analysis Constraints

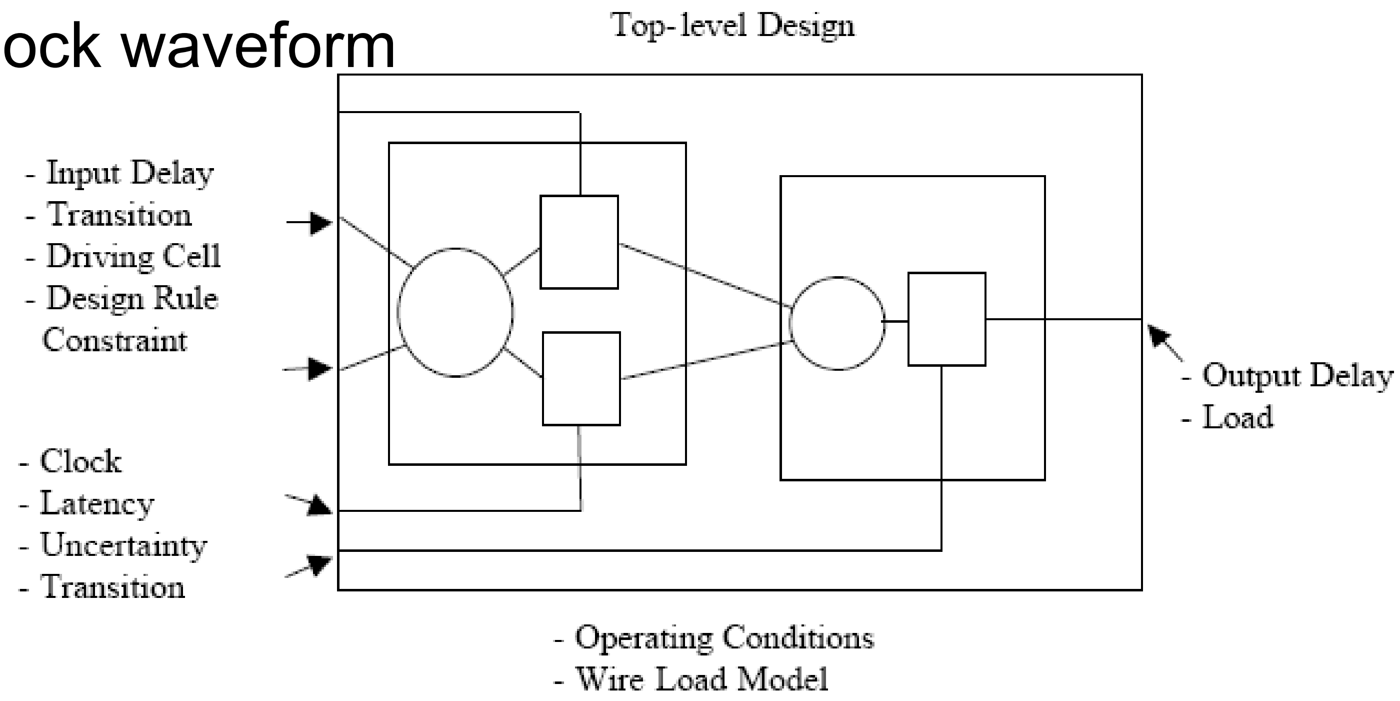
- Environment constraints
- Timing constraints
- Timing exceptions

Environment Constraints

- Operating conditions
- Wire load model
- Design rule constraints
  - Max transition time: max transition time of a driving net
  - Max capacitance: max capacitance of a driving net
  - Max fanout: max number of fanout allowed of a driving net

Timing Constraints

- Input
  - Arrival times (input delays)
  - Input drives
- Output
  - Required times (output delays)
  - Output loads
- Clock waveform



Courtesy of Prof. H.-R. Jiang

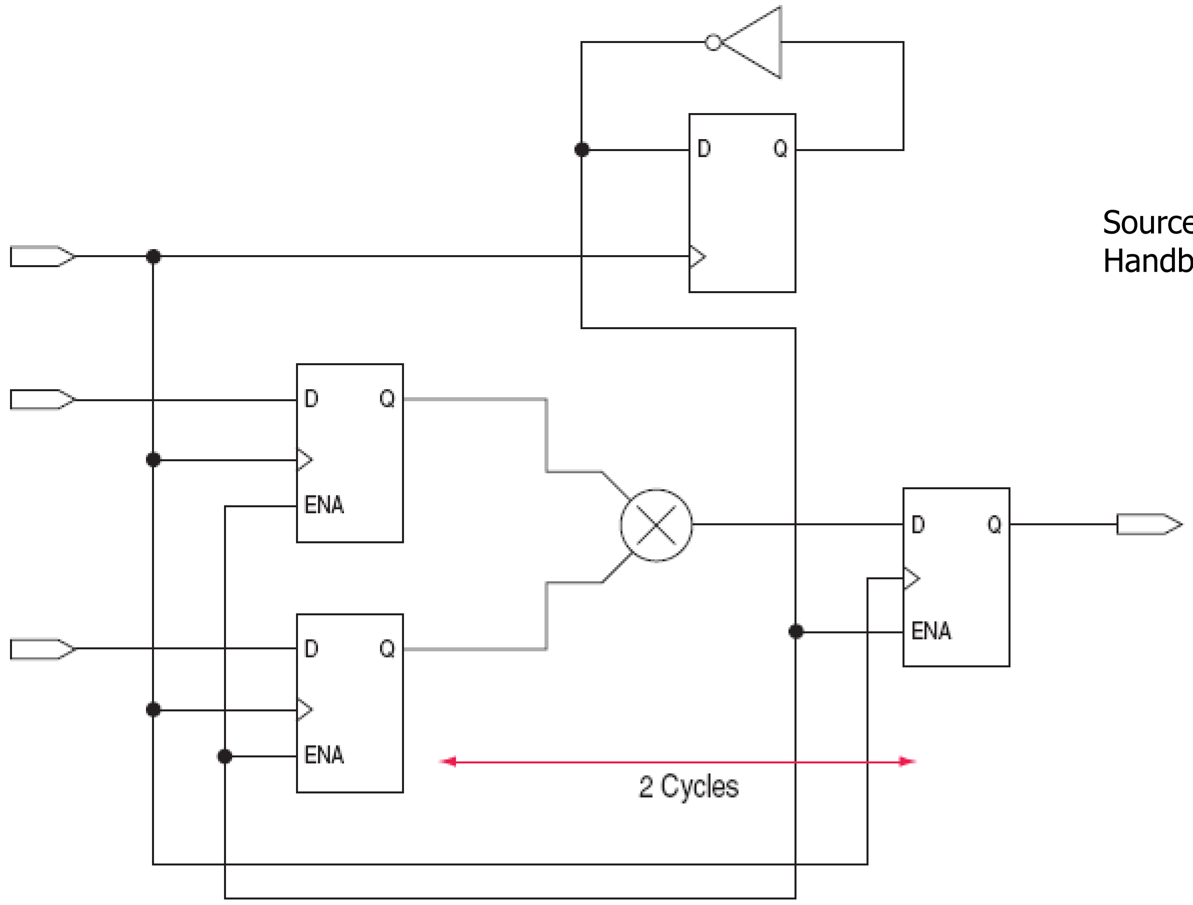
Timing Exceptions

- Override default single-cycle timing behavior
  - Multicycle path
    - Override the default setup & hold relations
  - Test mode path
    - Only active in test mode, usually not consider in regular mode operation
  - False (or unsensitizable) path
    - Prune timing paths which are never be exercised functionally

Multicycle Paths

- Multicycle paths are data paths that require more than one clock cycle to latch data at the destination register

Figure 8–10. Example Diagram of a Multicycle Path

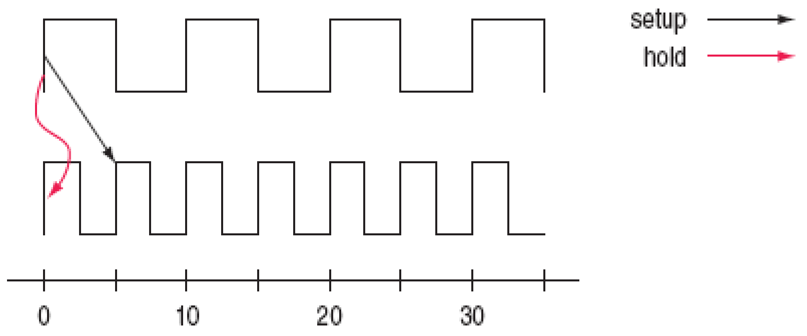


Source: Altera QuartusII Handbook, 2009

Multicycle Paths (cont'd)

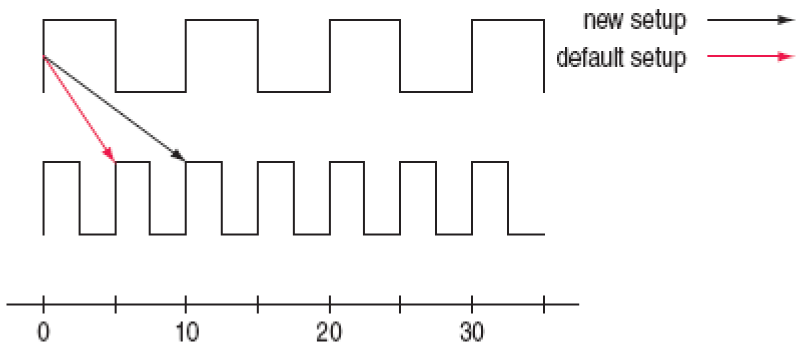
- The timing diagram after a multicycle setup of two has been applied. The command moves the latch edge time to 10 ns from the default 5 ns

Figure 8–12. Default Setup and Hold Timing Diagram



Source: Altera QuartusII Handbook, 2009

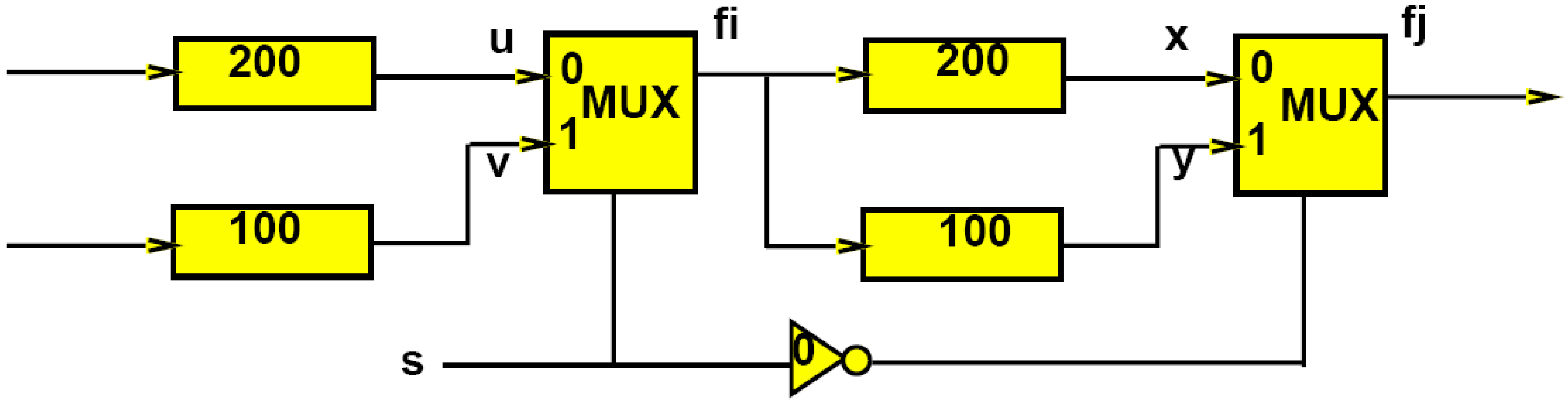
Figure 8–13. Modified Setup Diagram



False Paths

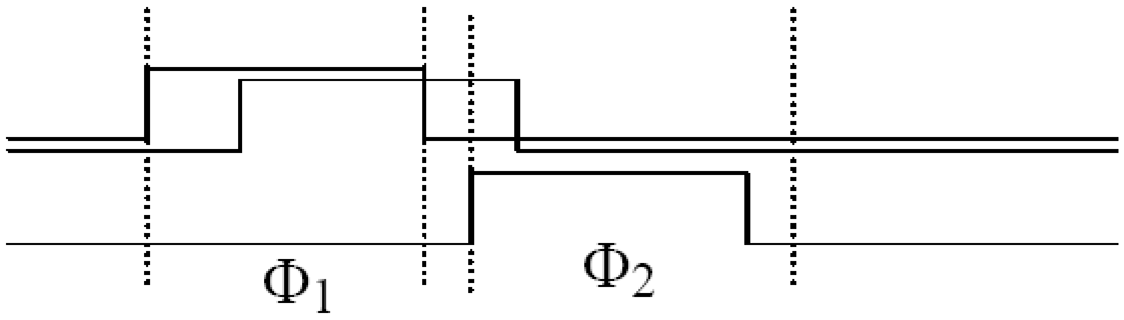
- Static timing analysis maybe inaccurate because of false paths
- Example:

Static analysis is fast but leads to **false paths**
Path of length 400 is never “exercised”

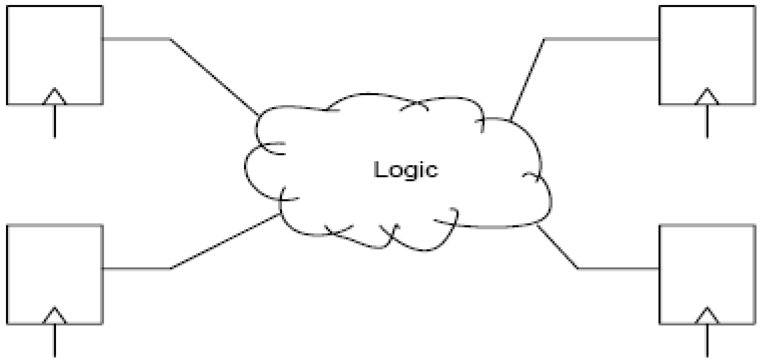


Timing Issues

- Shortest and longest paths
- Clock skew: can be fatal
  - Problem: unintentionally overlapping clock phases



  - Problem: possible reduction of timing budget



Max frequency = 1 / (longest path delay + worst skew delay)