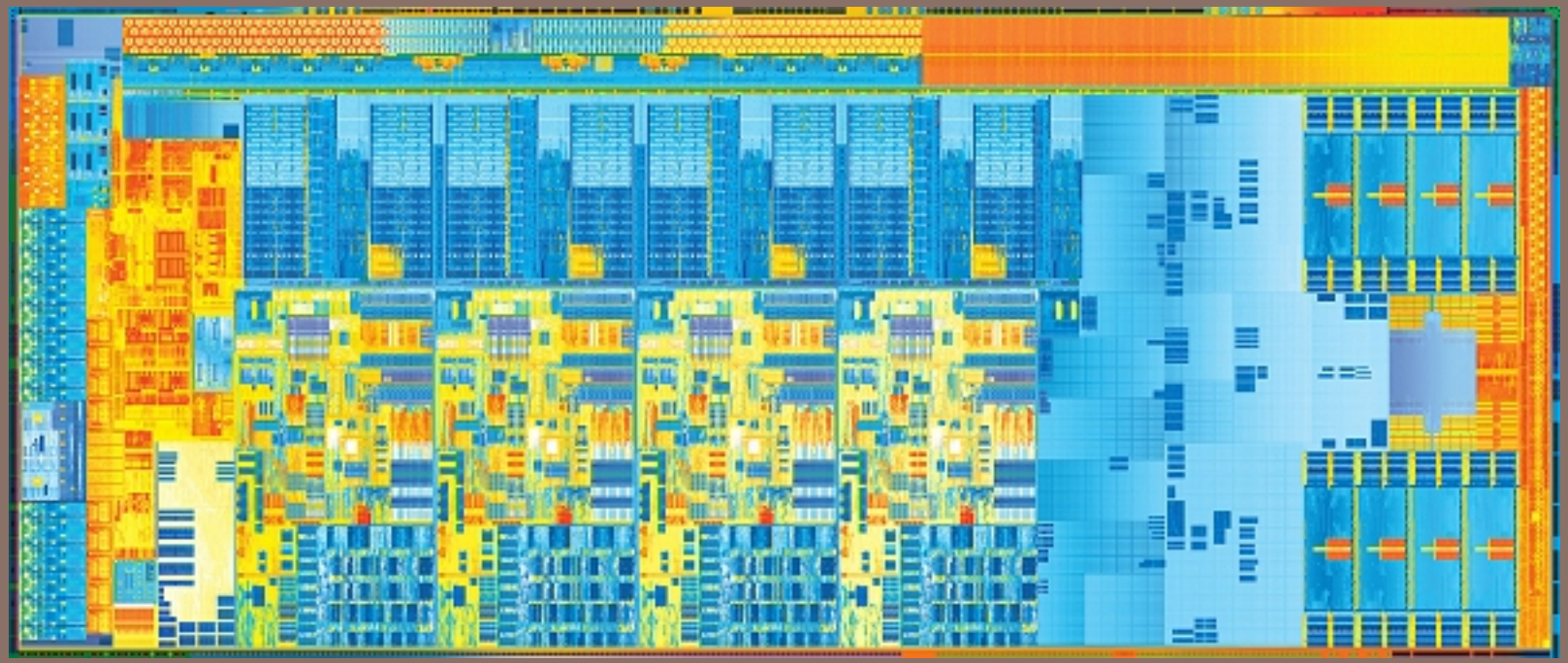


CSCE337 – Digital Design II

LECTURES 15-17: TIMING & CLOCKING ISSUES IN DIGITAL SYSTEMS

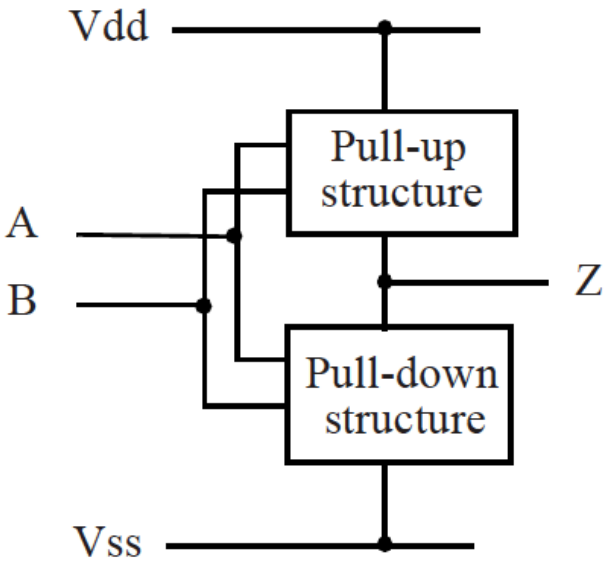
Spring 2015

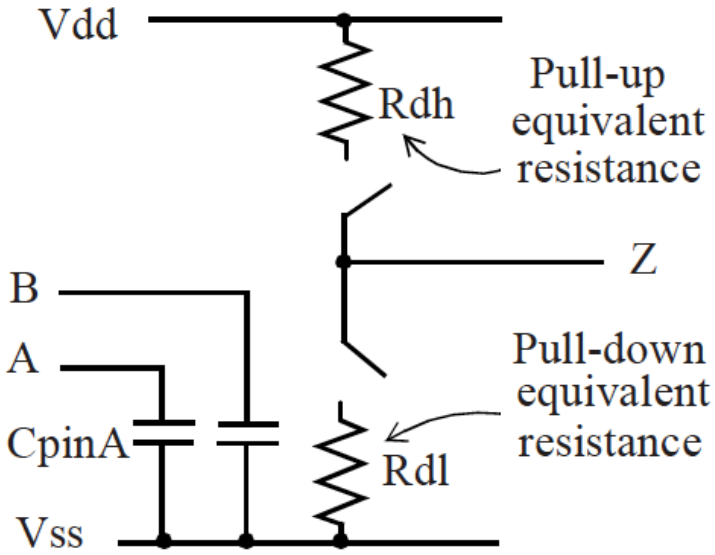
Mohamed Shalan

Agenda

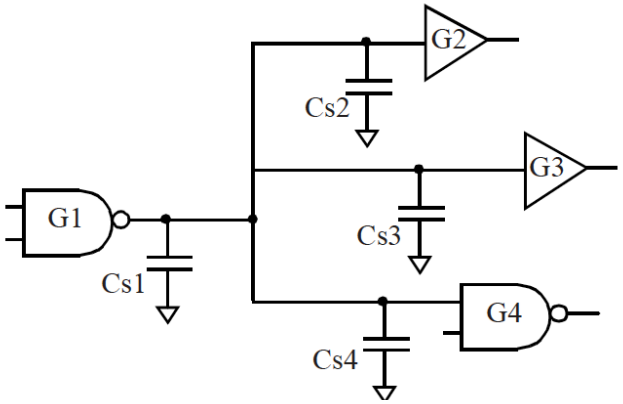
- Introduction
- Clocking
- Setup & Hold Times
- Clock Signal
- Timing & Logic Synthesis
- Static Timing Analysis
- Fixing Timing Violations

CMOS Cell Model

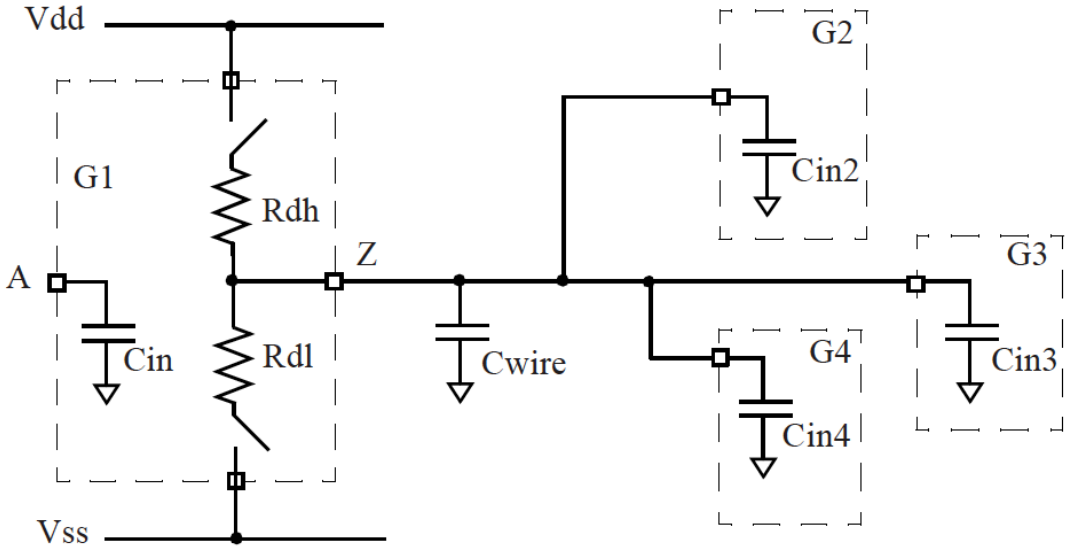




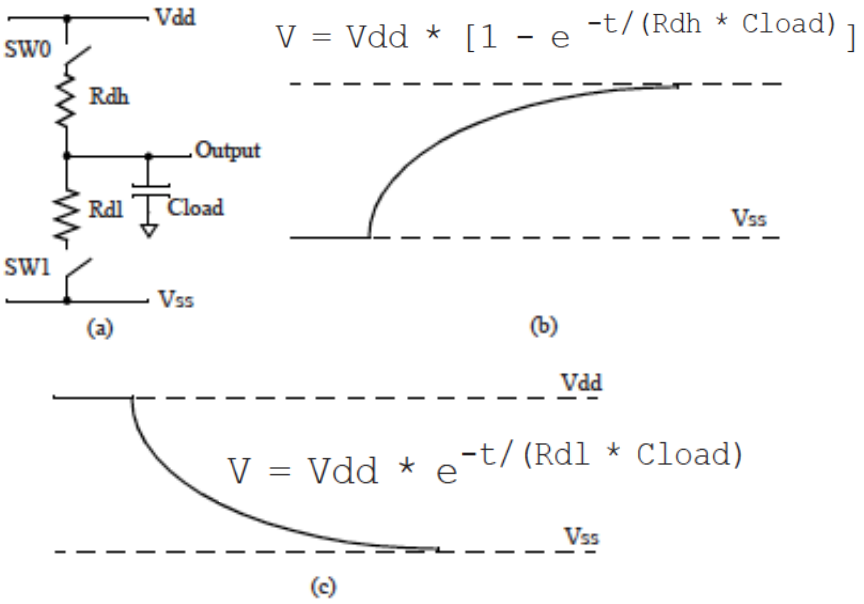
CMOS Delay Model

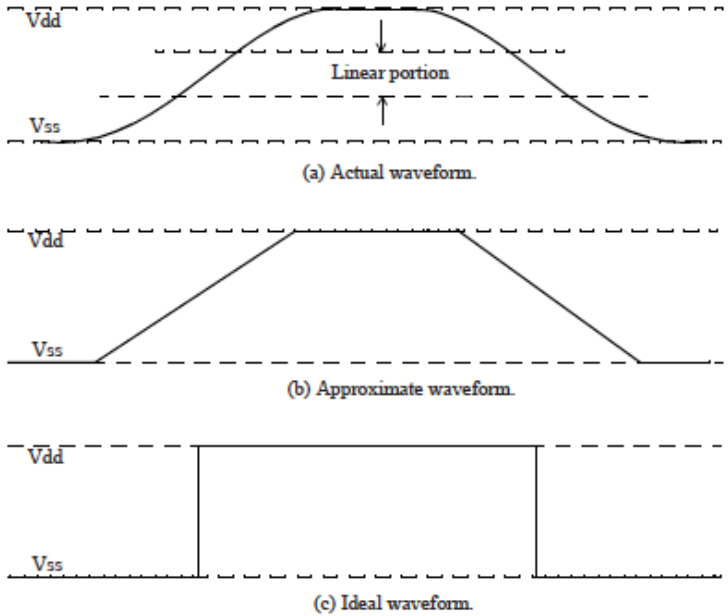


- $C_{wire} = C_{s1} + C_{s2} + C_{s3} + C_{s4}$
- tpd = R x C_{load}
- R is either Rdl or Rdh
- $C_{load} = C_{wire} + C_{in2} + C_{in3} + C_{in4}$

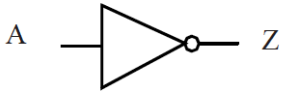


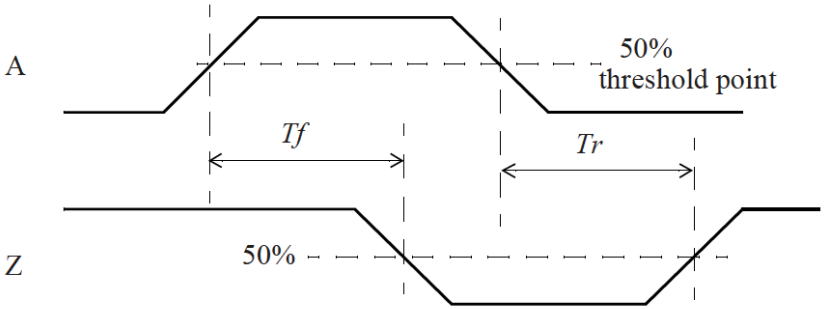
CMOS Delay Model

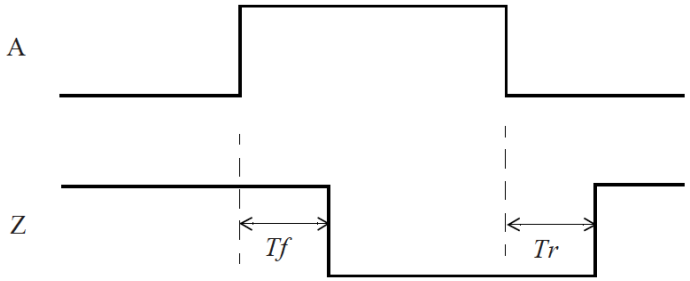




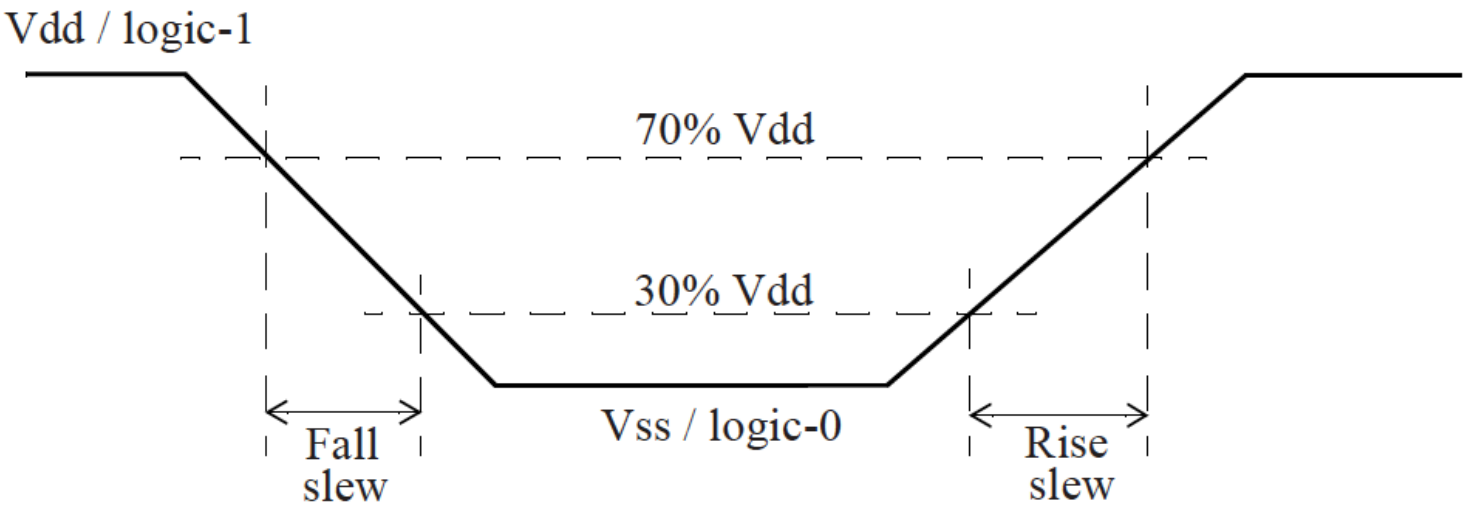
Rise and Fall Times







Slew Rate (Transition Time)



20-80% and 30-70% are in use (10-90% range was used for early technologies)

Cell Delay Specification in Libraries

- Genlib (Used by ABC and SIS)
  - Uses 1-D linear timing model
  - A cell is specified in the following format:

```
GATE <cell-name> <cell-area> <cell-logic-function>
<pin-info>
    .
    .
<pin-info>

PIN <pin-name> <phase> <input-load> <max-load>
    <rise-block-delay> <rise-fanout-delay>
    <fall-block-delay> <fall-fanout-delay>
```

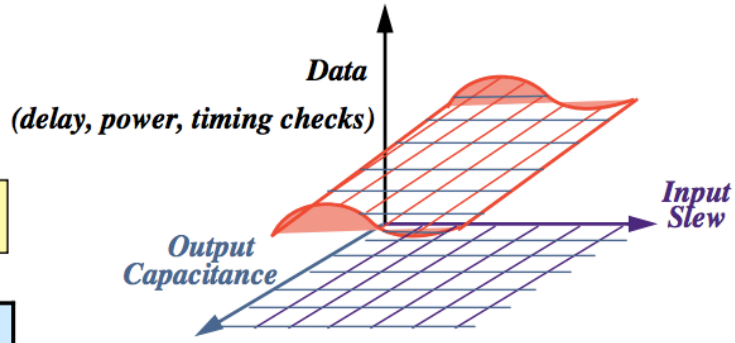
Genlib Example


```
GATE inv      1      O=!a;
       PIN * INV 1 999 0.9 0.3 0.9 0.3

GATE nand2    2      O=!(a*b);
       PIN * INV 1 999 1.0 0.2 1.0 0.2

GATE mux21    4      O=a*s+b*!s;
    PIN a NONINV 1 999 1.6 0.4 1.6 0.4
    PIN b NONINV 1 999 1.6 0.4 1.6 0.4
    PIN s UNKNOWN 2 999 2.0 0.4 1.6 0.4
```

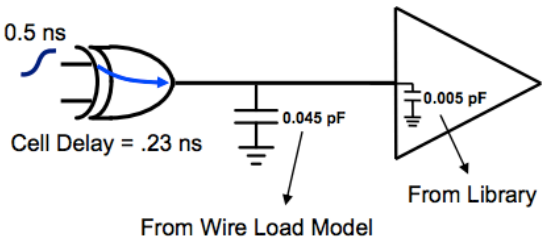
$t_r = 1.6 + 0.4 \times load$
$t_f = 1.6 + 0.4 \times load$

Liberty Timing Model (NLDM)

- Cell delays are calculated from a Non Linear Delay Model (NLDM) table in the technology library
- Tables are indexed by input transition and total output load for each gate



Cell Delay = f (Input Transition Time, Output Load)



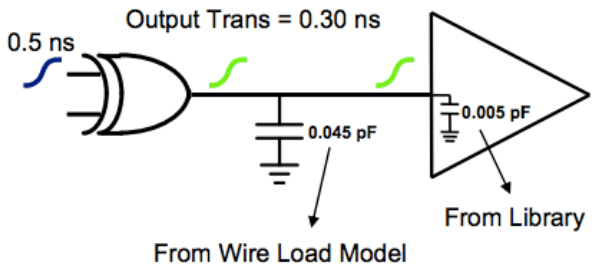
SPICE	Output Load (pF)			
Input Trans (ns)	.005	.05	.10	.15
0.0	.1	.15	.2	.25
0.5	.15	.23	.3	.38
1.0	.25	.4	.55	.75

Cell Delay (ns)

Liberty Timing Model (NLDM)

- There is another NLDM table in the library to calculate output transition
- Output transition of a cell becomes the input transition of the next cell down the chain

Output Transition = f (Input Transition Time, Output Load)

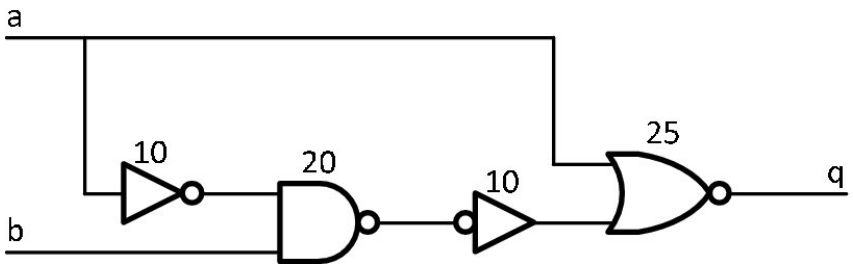


SPICE	Output Load (pF)			
	.005	.05	.10	.15
Input Trans (ns) 0.00	0.10	0.20	0.37	0.60
0.50	0.18	0.30	0.49	0.80
1.00	0.25	0.40	0.62	1.00

Output Transition (ns)

Path Delay

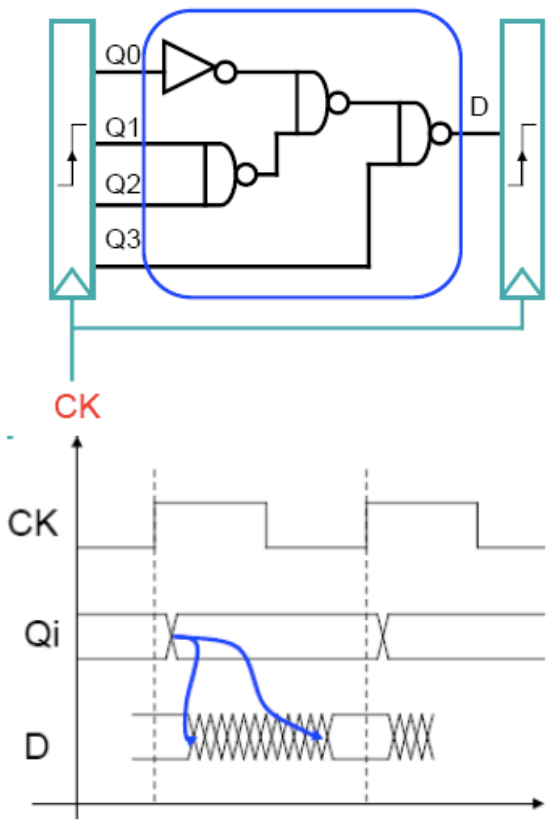
- What is the propagation delay?



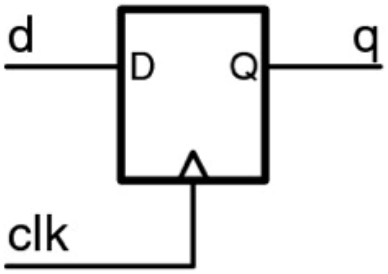
  - When all inputs change, the output starts to change or *glitch* after the minimum delay of 25ns, and settles to the final value after the propagation delay (65ns).
    - t_{cd}: Contamination delay (important for setup violations)
    - t_{pd}: Propagation delay (important for hold violations)

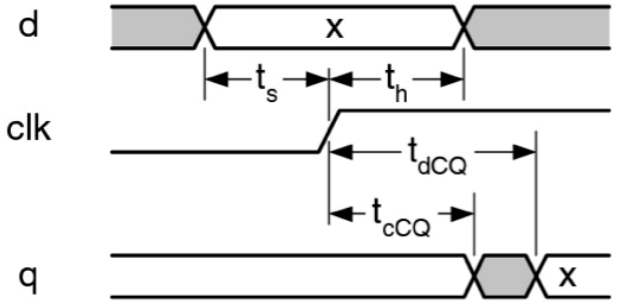
Clocking

- Clocks provide the means to synchronize
- By allowing events to happen at known timing boundaries, we can sequence these events
- Greatly simplifies building of state machines
- No need to worry about variable delay through combinational logic (CL)
- All signals delayed until clock edge (clock imposes the worst case delay)



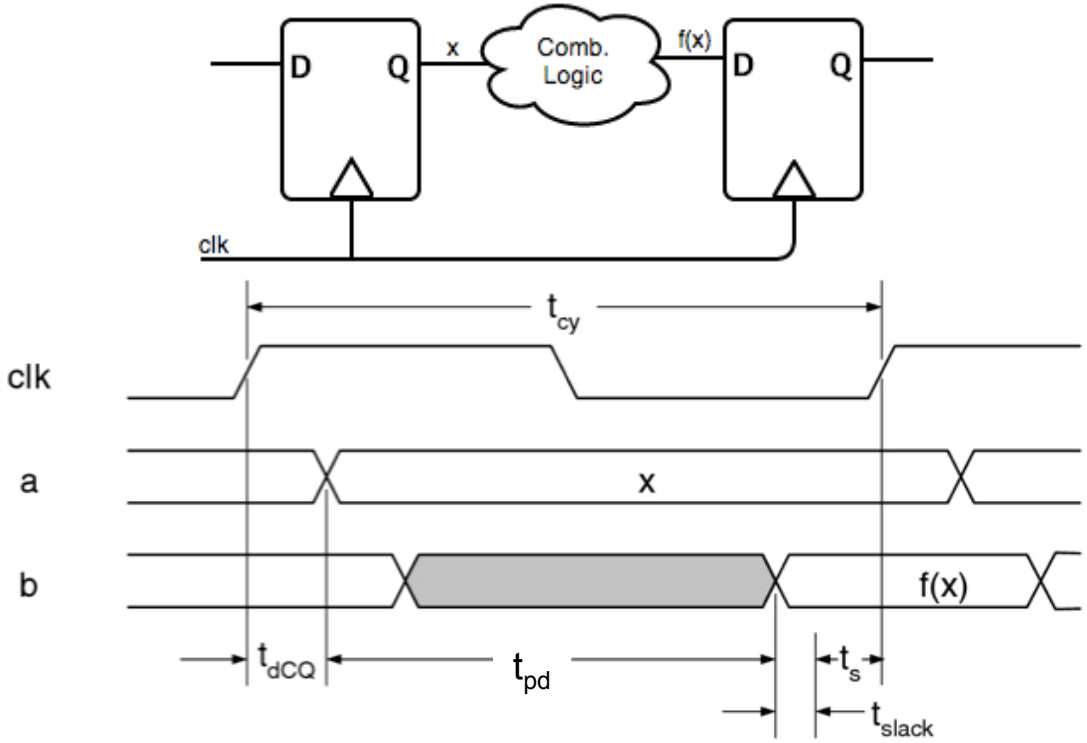
Flip-Flop Timing Constraints





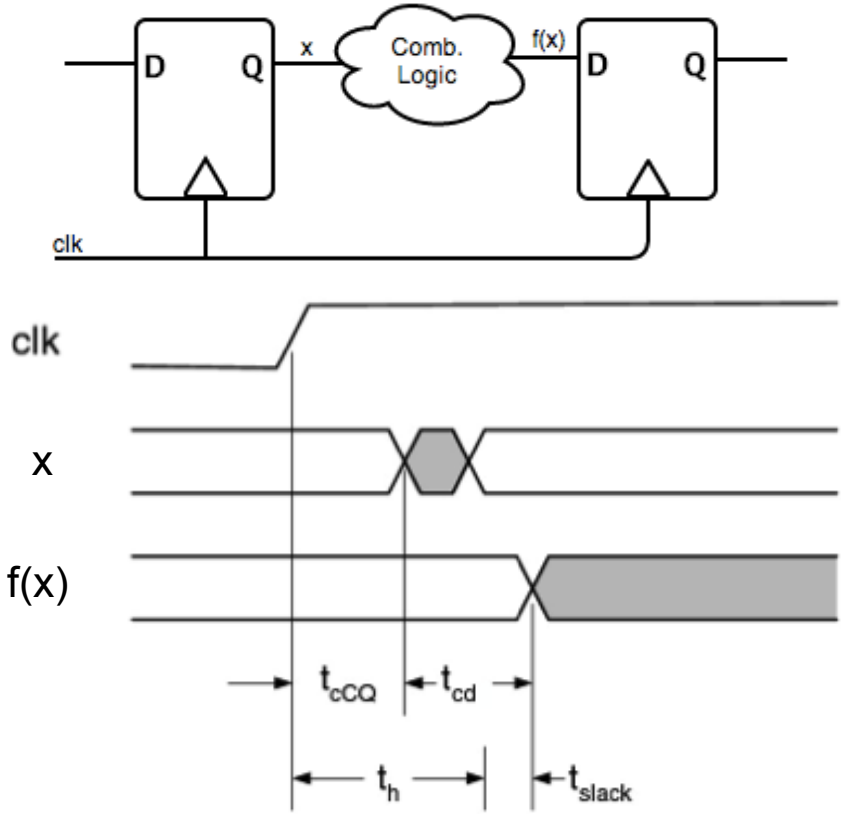
- t_s: Setup Time
  - The minimum amount of time the data signal should be held steady before the clock edge
- t_h: Hold Time
  - the minimum amount of time the data signal should be held steady after the clock edge
- t_{CQ}: Time to output (Q) to settle after the Clock (C) edge
  - t_{cCQ}: minimum delay (similar to contamination delay)
  - t_{dCQ}: Maximum delay (similar to propagation delay)

Setup and Hold Constraints



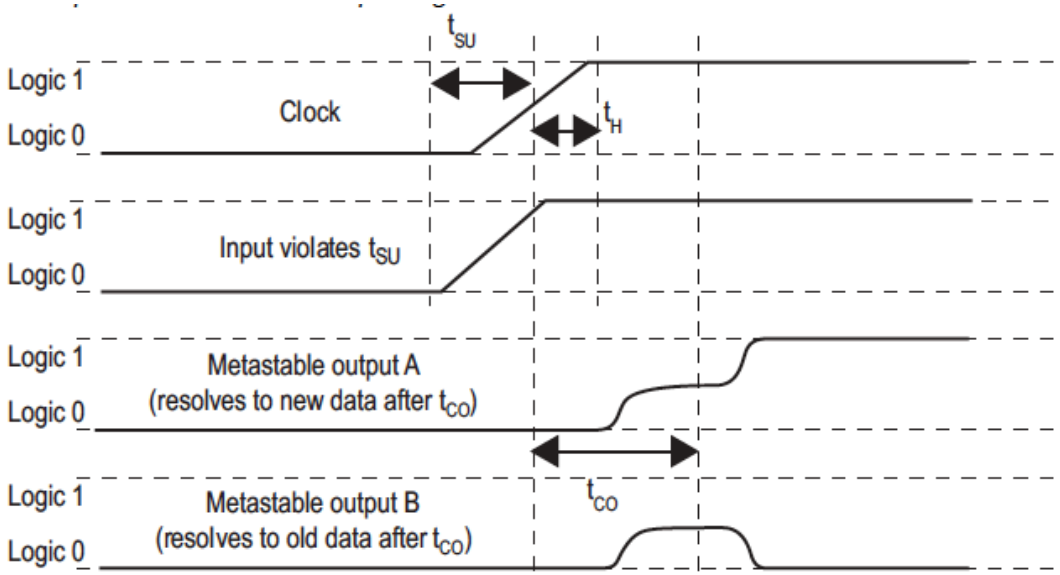
- $t_{cycle} >= t_{dCQ} + t_{pd} + t_s$
- $t_{slack} = t_{cycle} - (t_{dCQ} + t_{pd} + t_s)$

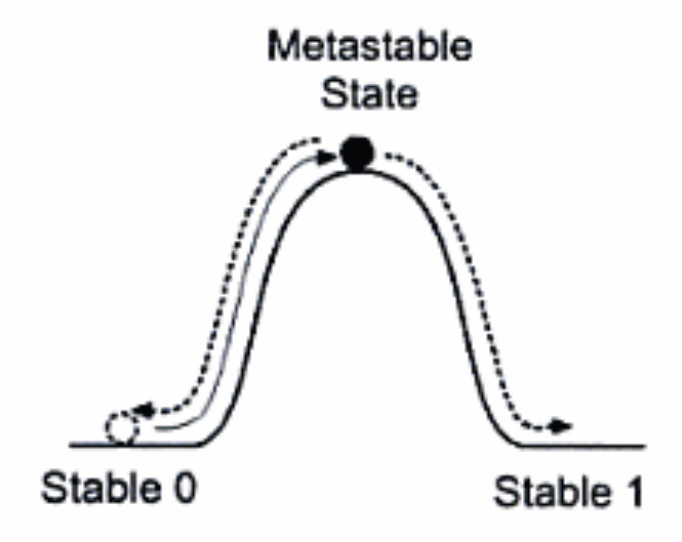
Setup and Hold Constraints



- $t_{cCQ} + t_{cd} >= t_{hold}$

Violating Setup & Hold Time Constraints





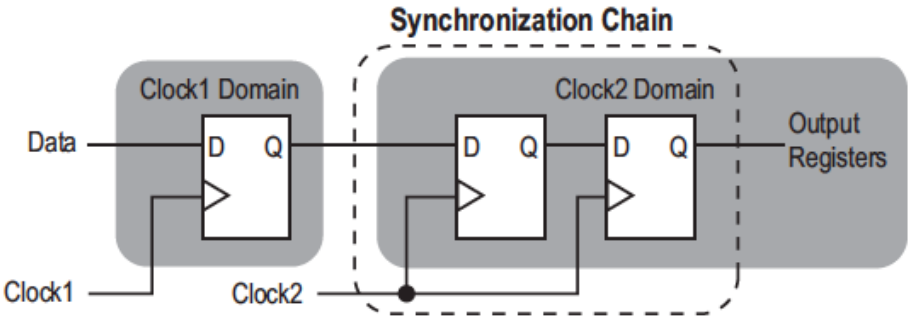
- Violation implies metastability

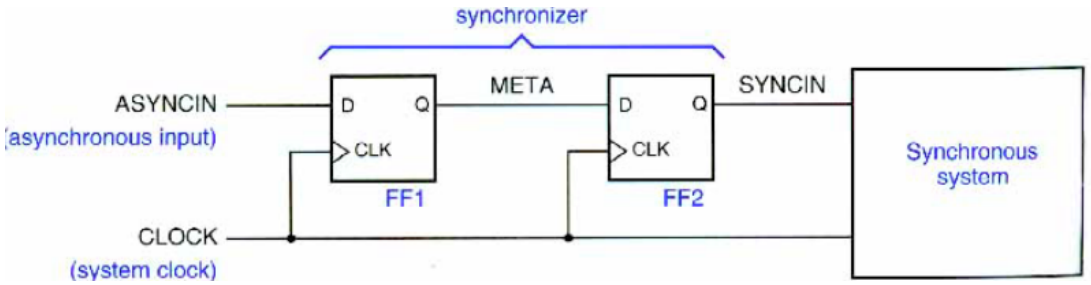
Asynchronous Signals & Metastability

- Many synchronous systems need to interface to asynchronous input signals
  - Consider a computer system running at 1GHz with interrupts from I/O devices, keystrokes, etc.
  - Data transfers from devices with their own clocks. For example, Ethernet has its own 100MHz clock
- The probability that a flip-flop stays in the metastable state decreases exponentially with time.

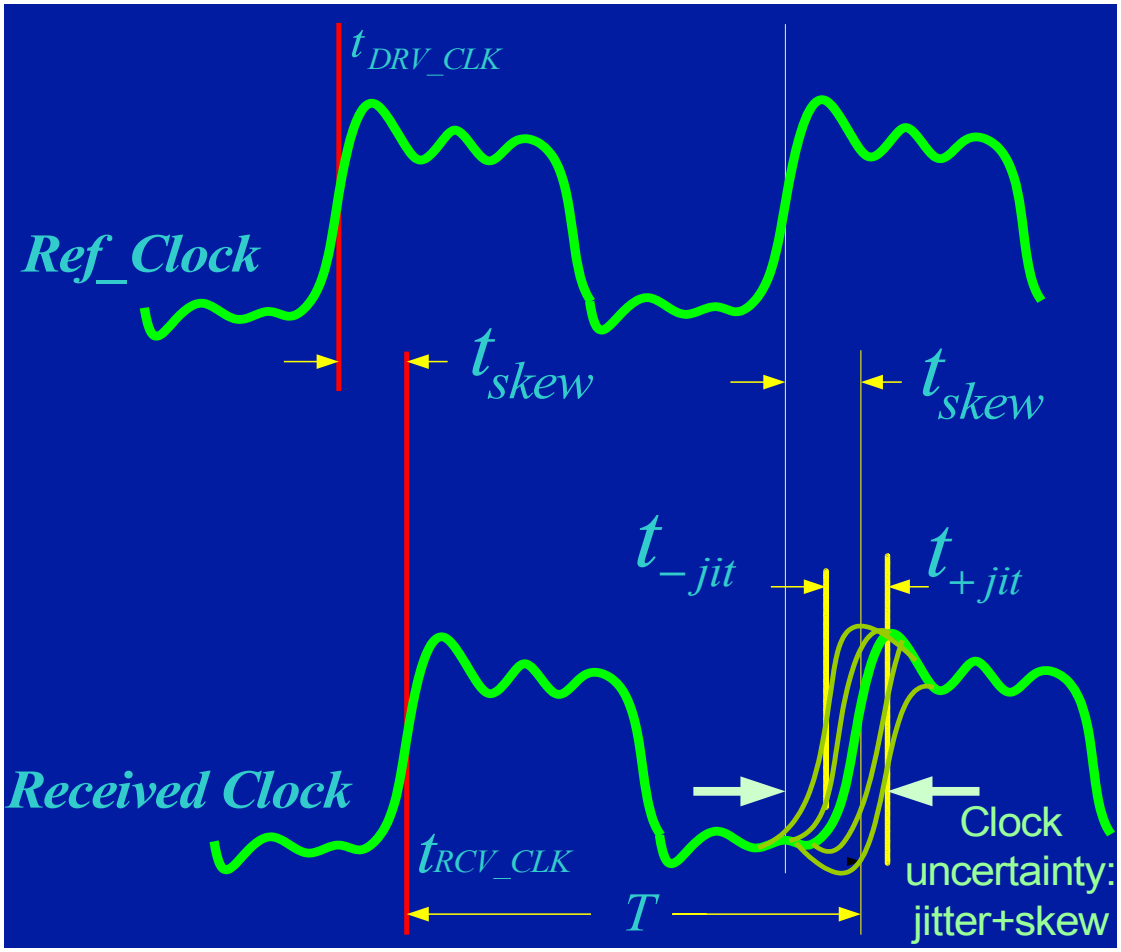
Asynchronous Signals & Metastability

- Therefore, any scheme that delays using the signal can be used to decrease the probability of failure.





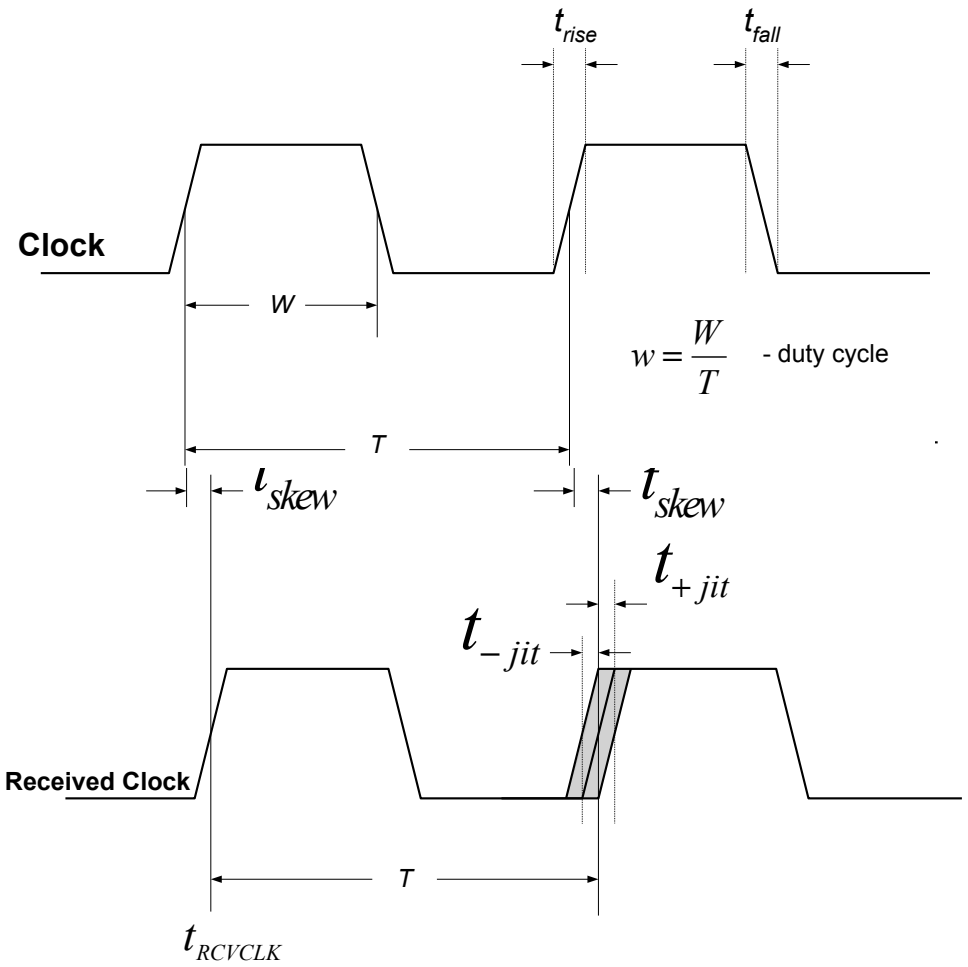
Clock Signal non-ideal Behavior



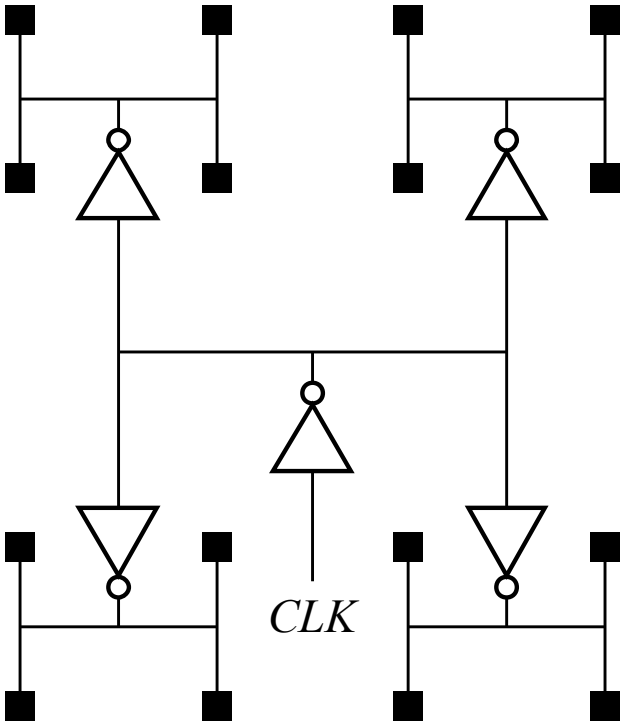
Clock Signal non-ideal Behavior

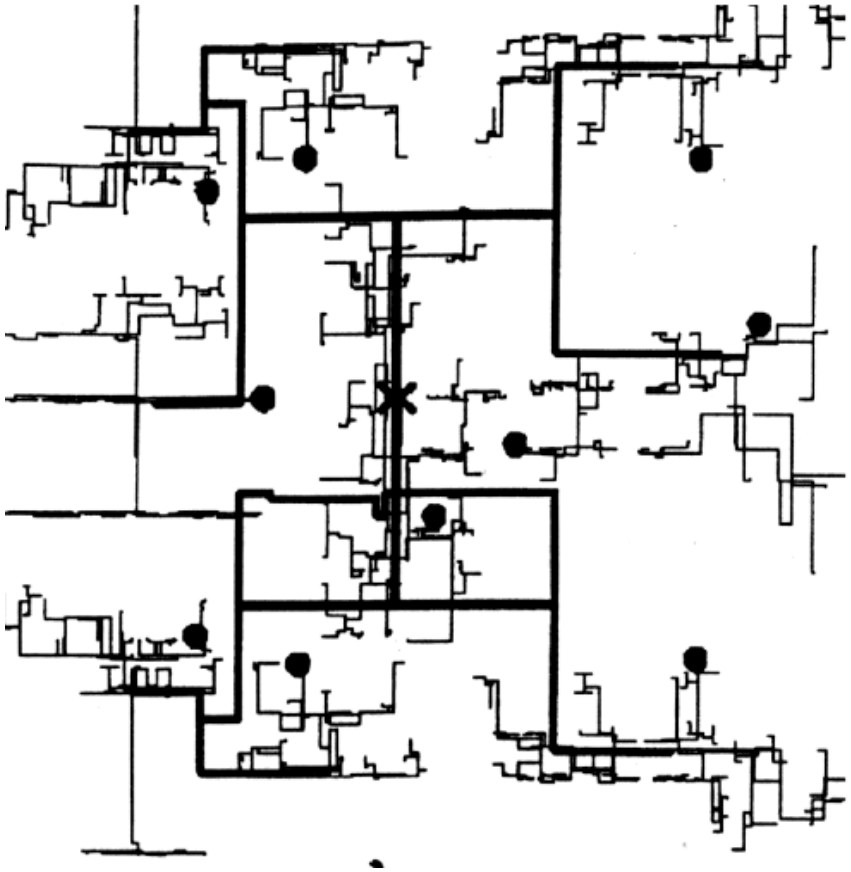
- Skew
  - Differences in clock signal arrival times across the chip
  - Cause: Interconnect delay (unbalanced Clock Tree)
- Jitter
  - Temporal variations in clock edges at a given point on the chip.
    - Results in continuous clock period variation.
    - In general, each clock period has a different jitter.
    - t_{jitter} is the maximum absolute jitter.
  - Cause: EM Interference, Cross-Talk, ….

Clock Signal non-ideal Behavior



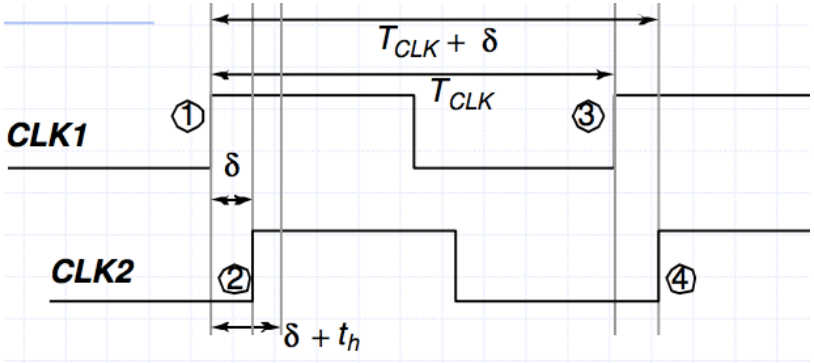
Clock Tree and Skew



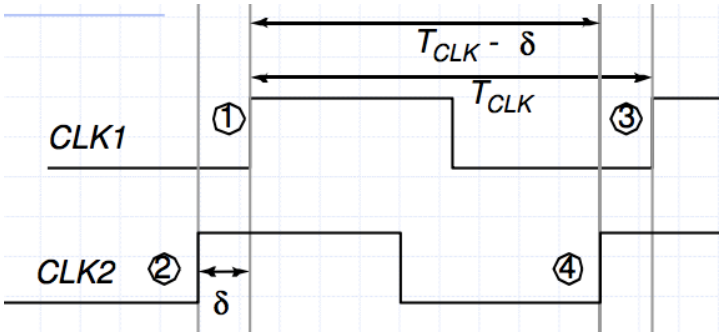


Positive Skew & Negative Skew

- +ve Skew

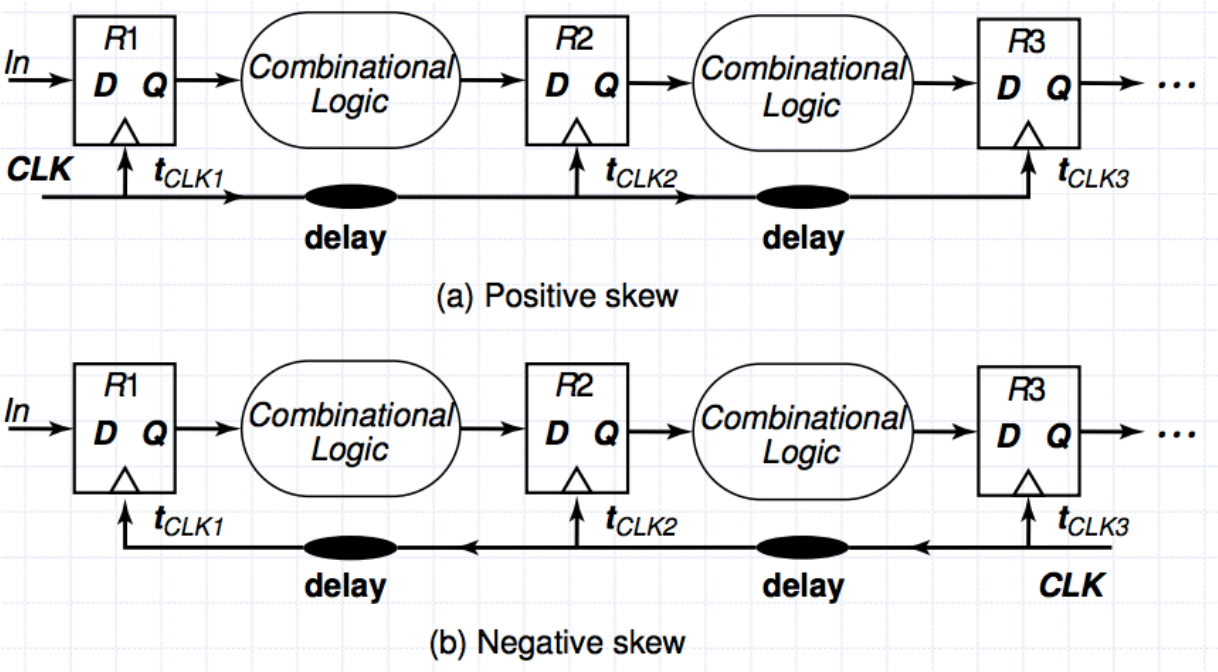


- -ve Skew



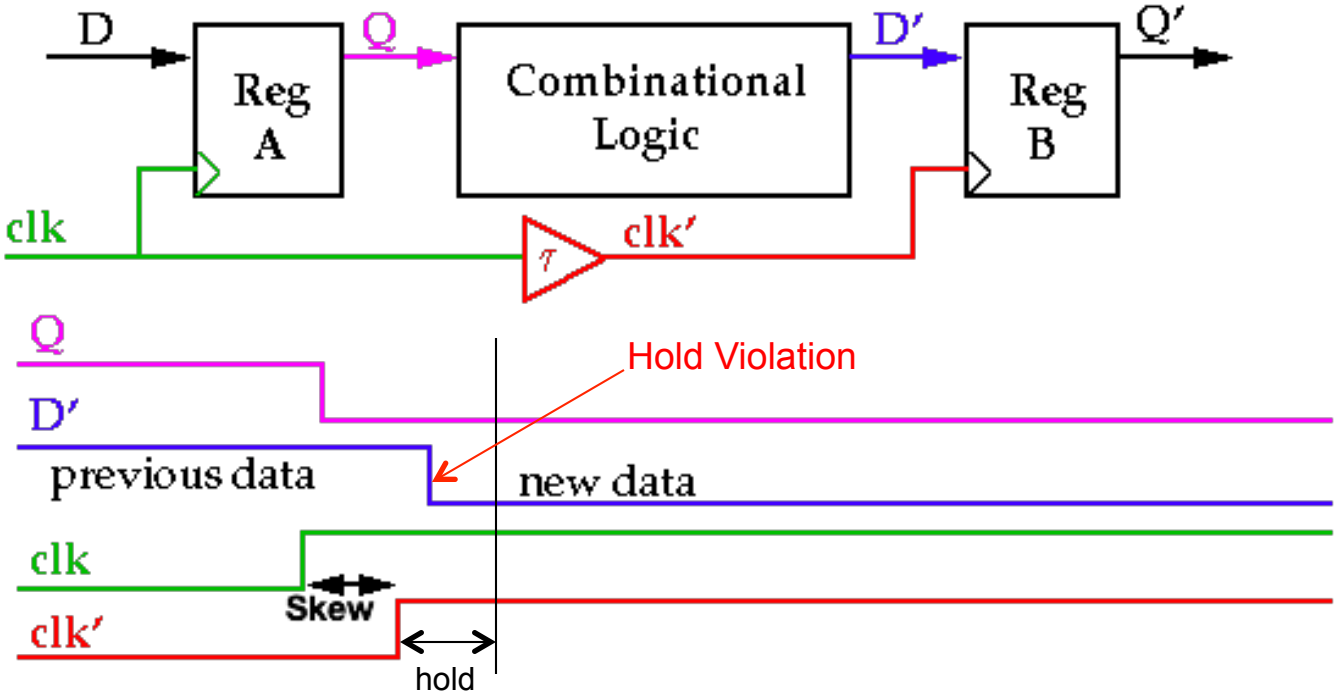
Positive Skew & Negative Skew

- +ve skew can occur when data & clock travel in same direction
- -ve skew can occur when data & clock travel in opposite directions



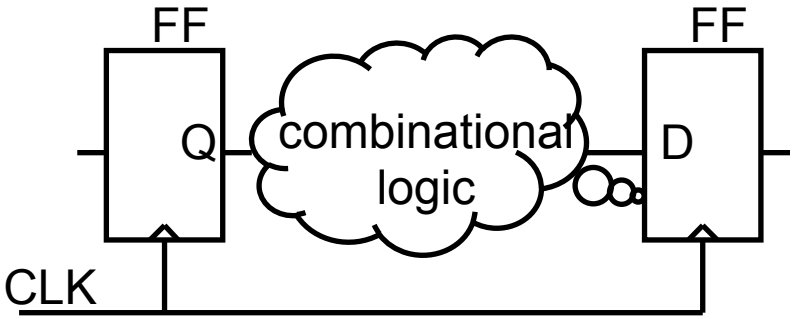
Clock Skew & Hold Violations

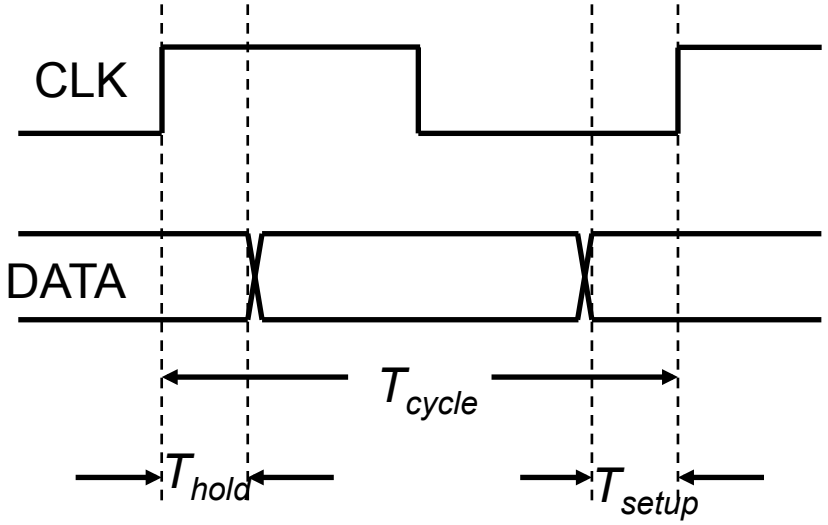
- +ve Clock Skew may cause Hold violations if the CL delay is small



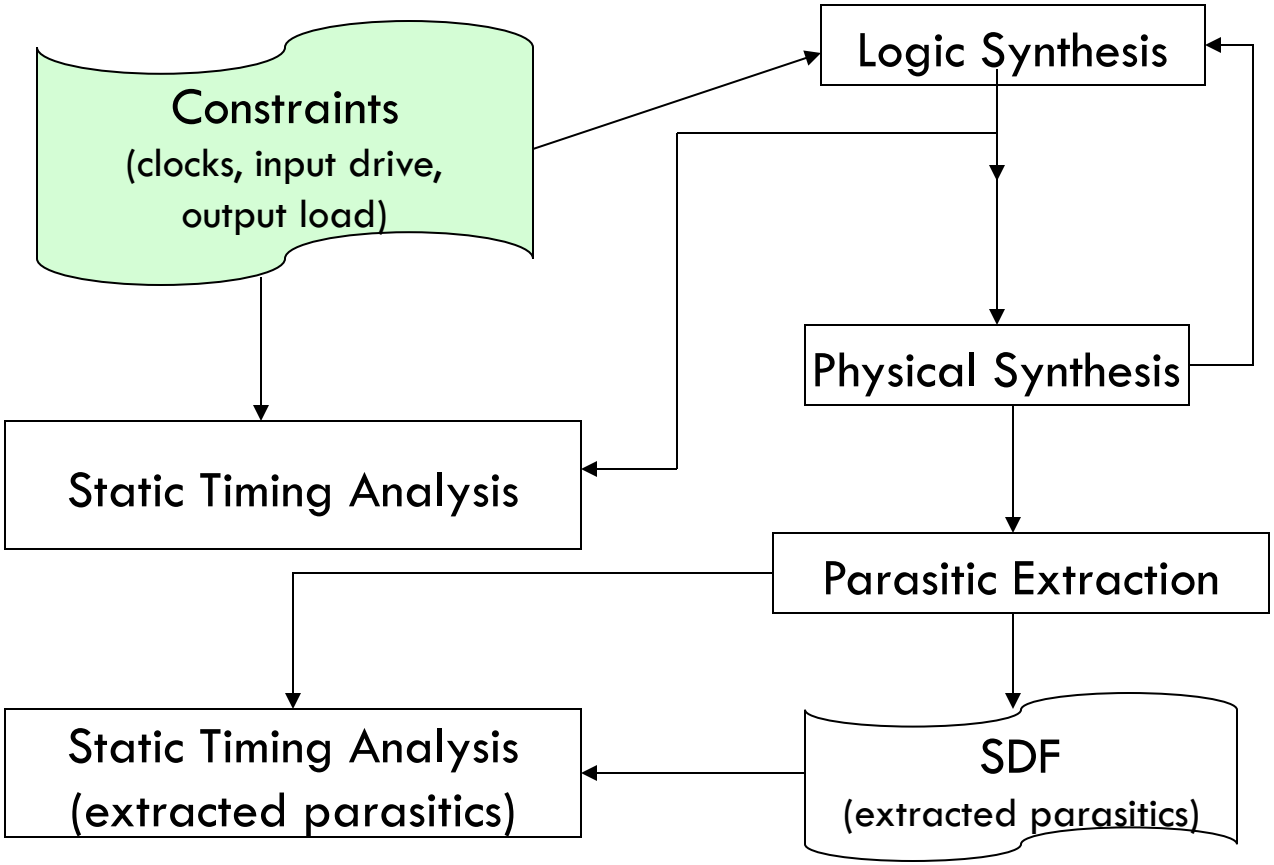
Clock Skew Timing Checks

- For each Flip Flop:
  - $t_{cd} + t_{cCQ} > t_{hold} + t_{skew}$
  - $t_{pd} + t_s + t_{dCQ} < t_{cycle} - t_{skew}$



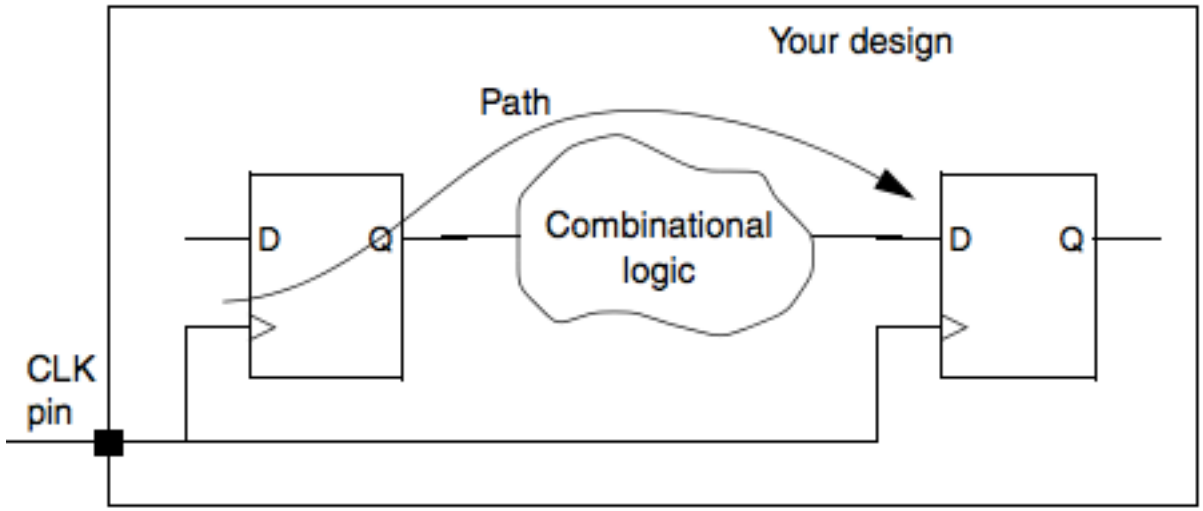


Digital Flow (Revisited)



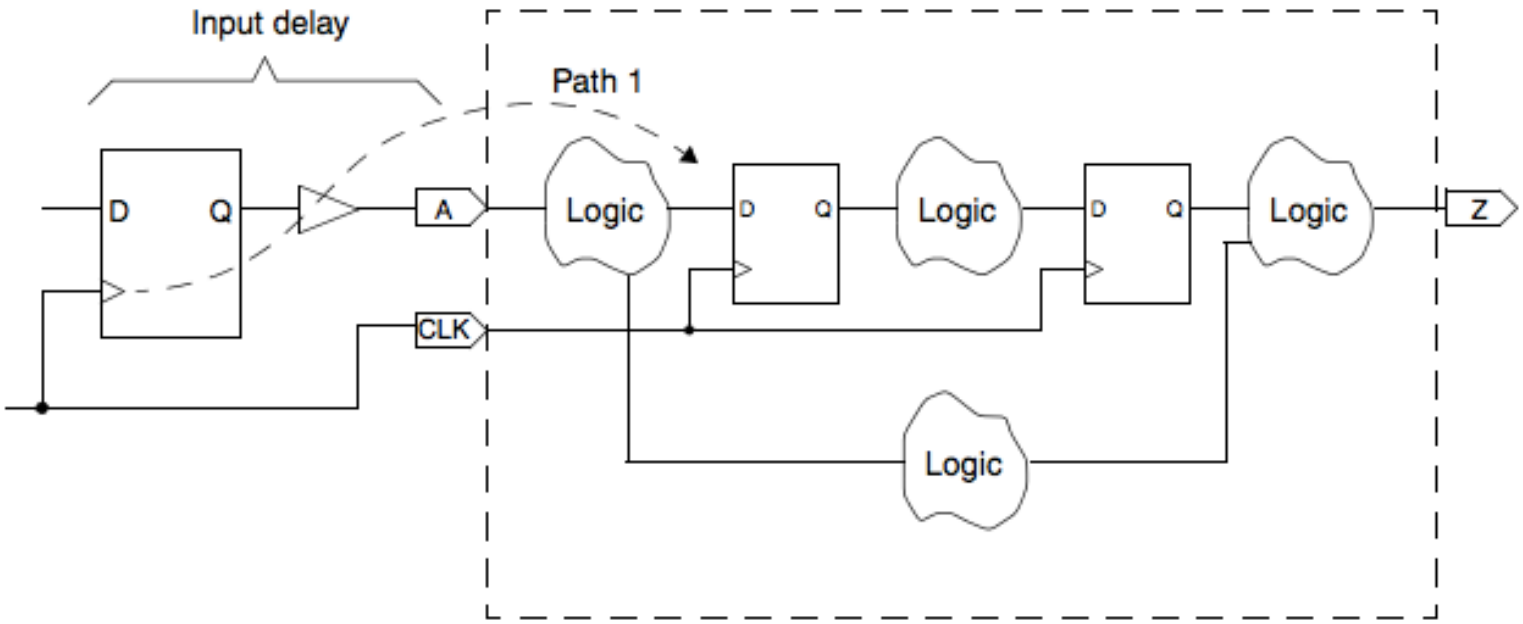
Timing and Logic Synthesis (constraints)

- Setting the Clock (`create_clock -period 10 [get_ports CLK]`)



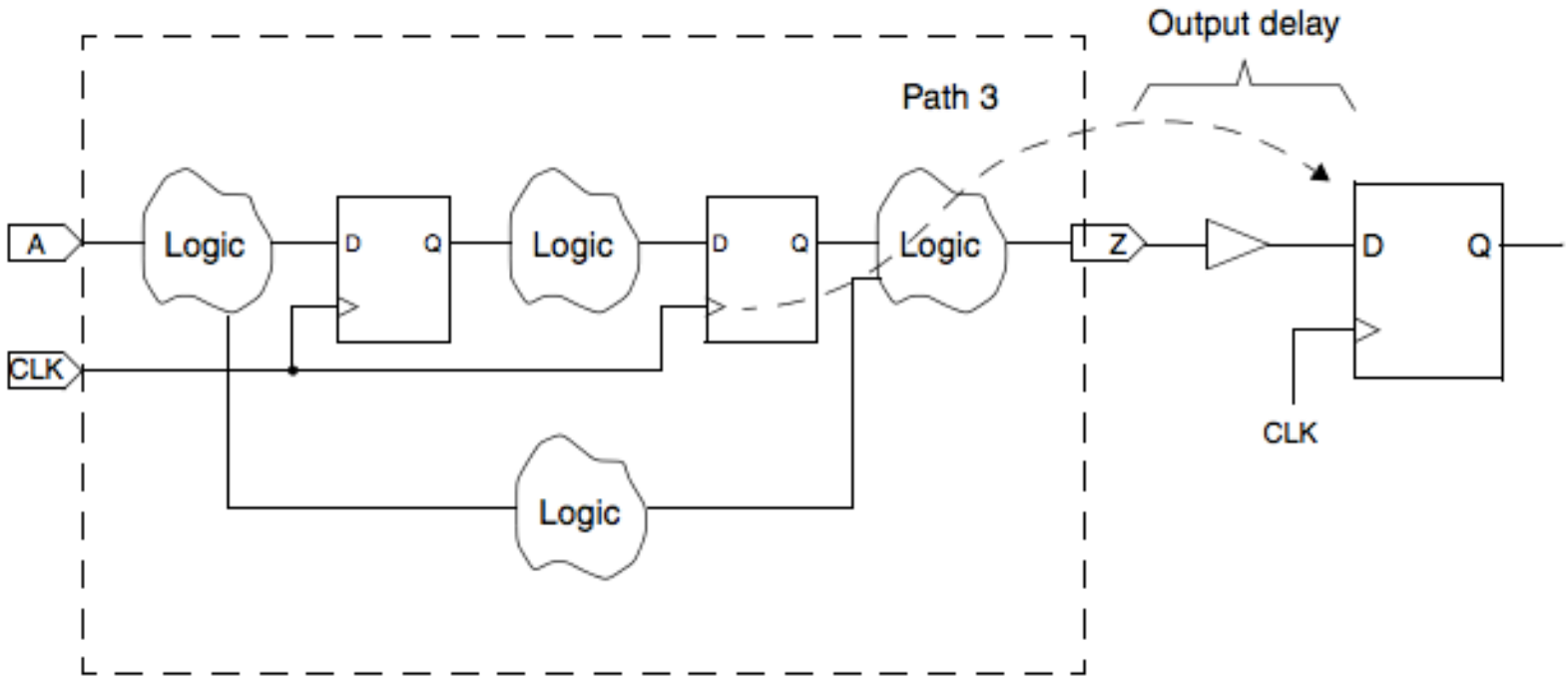
Timing and Logic Synthesis (constraints)

- Input Delay (`set_input_delay 0.2 -clock CLK A`)

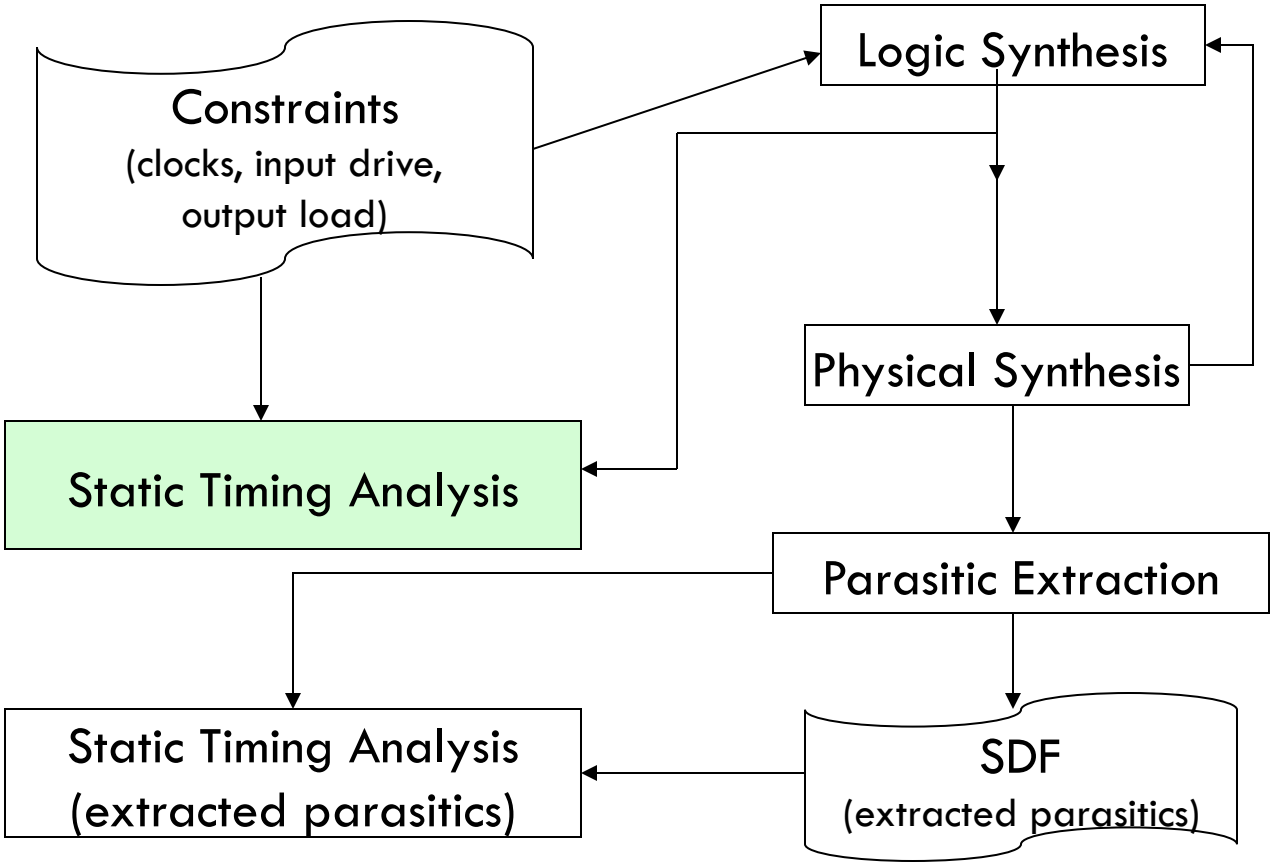


Timing and Logic Synthesis (constraints)

- Output Delay (`set_output_delay 0.5 -clock CLK Z`)



Digital Flow (Revisited)



Static Timing Analysis

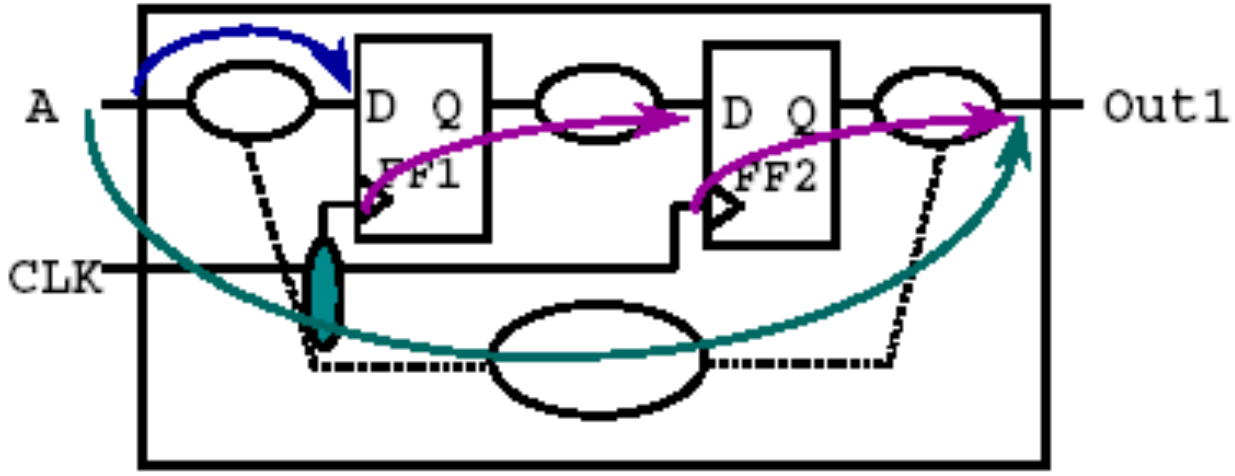
- Problem: Given a circuit, find the path(s) with the largest delay (critical paths)
  - Solution: run SPICE and report the results of the simulation
    - Problem: SPICE is computationally expensive to run except for small-size circuits plus it needs test vector
- WANTED: We need a fast method that produces relatively accurate timing results compared to SPICE
  - Solution: Static Timing Analysis (STA)
    - Typically takes a fraction of the time it takes to run

STA

- Static Timing Analysis is a method for determining if a circuit meets timing constraints without having to simulate
  - Much faster than timing-driven, gate-level/Spice simulation
  - Proper circuit functionality is not checked
  - Vector generation NOT required

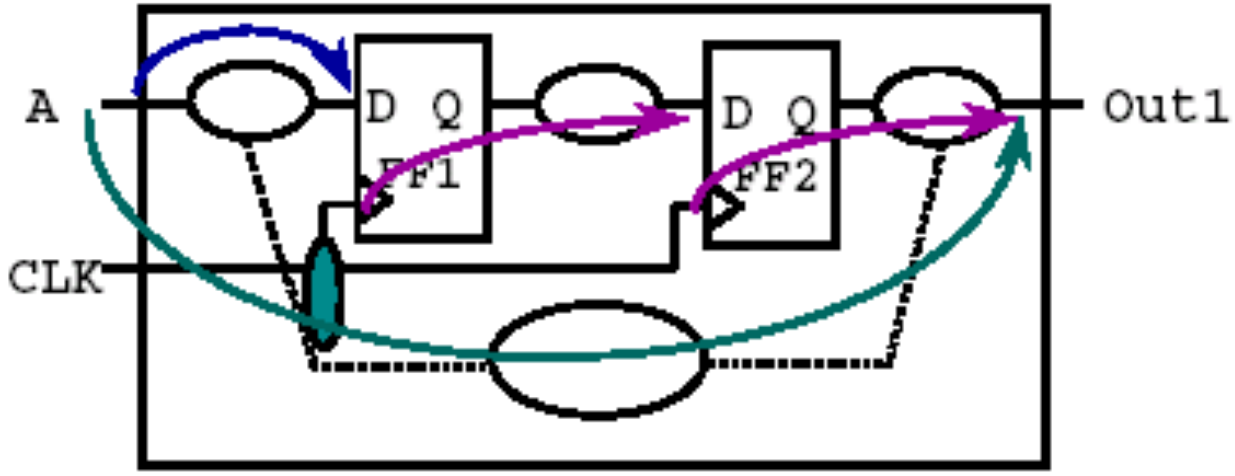
STA Steps

- Circuit is broken down into sets of timing paths
- Delay of each path is calculated
- Path delays are checked to see if timing constraints have been met



Timing Path

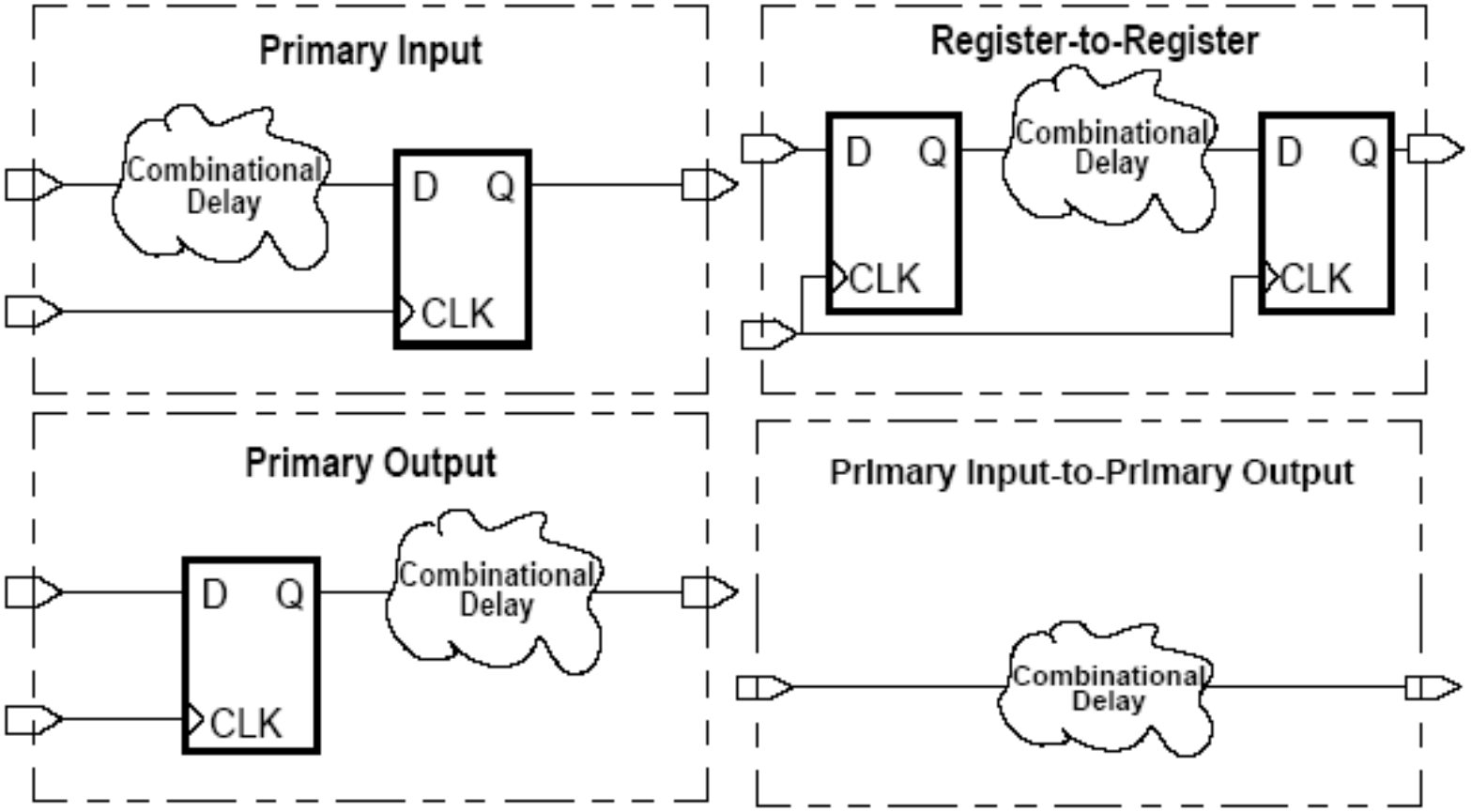
- A Timing Path is a point-to-point path in a design which can propagate data from one flip-flop to another
- Each path has a start point and an endpoint
  - Start point: Input ports Clock pins of flip-flops
  - Endpoints: Output ports Data input pins of flip-flops



Types of Timing Paths

- Depending on the logic through which data propagates a design can be considered to comprise of 4 paths
  - Input → reg : Data from Input port to the first flip-flop
  - Reg → reg : Data from one flip-flop to another ff
  - Reg → out : Data from last flip-flop to output port
  - Input → output : Data from Input – Output with no sequential components in between

Types of Timing Paths



STA Terminologies

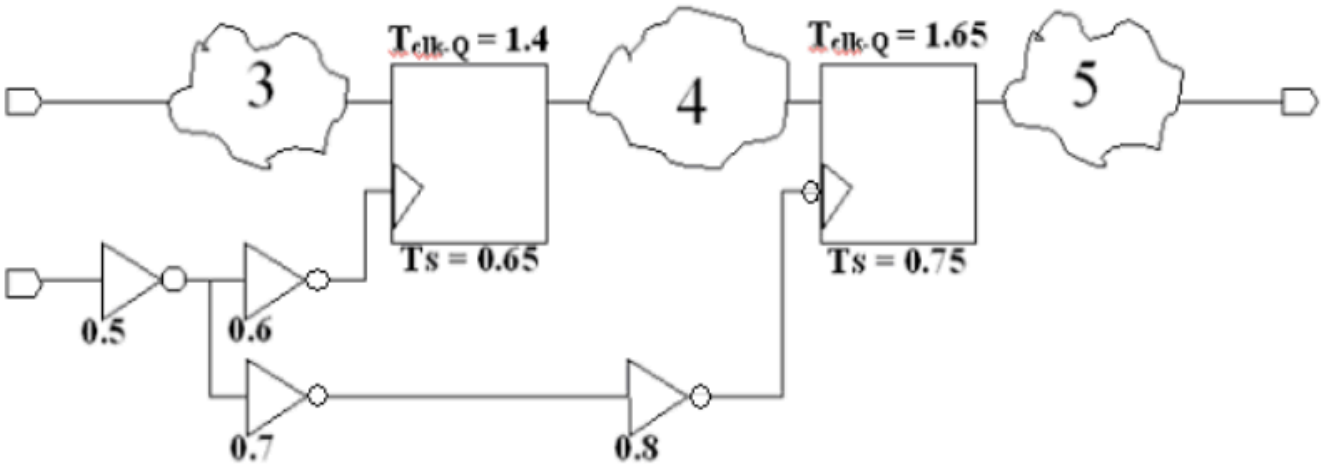
- Critical Path: Theoretically path which has maximum delay
- Arrival Time: Time taken by data to reach a end point from a specific start point.
- Required Time: Time at which data is required at a particular end point. Depends on the requirements / specifications
- Slack: Difference in required time and arrival time.
  - For a design to work the slack value should always be positive

Clock Frequency

- For a circuit to meet its specifications the minimum time period or maximum frequency at which it works should be identified
- The frequency of operation of a circuit depends on the logic of the paths
- Each of the four kinds of paths have different required times

Problem 1

- Find the maximum frequency



Solution Methodology

1) Find the total number of paths in the design
2) Identify the paths that are constrained by clock
3) Identify the arrival times of all the paths identified in step 2 above
4) Identify the required time equations of all paths identified in step 2
5) Equate the arrival times and required times of all relevant paths to obtain the time periods
6) Select the maximum applicable clock frequency from the clocks identified in step 5

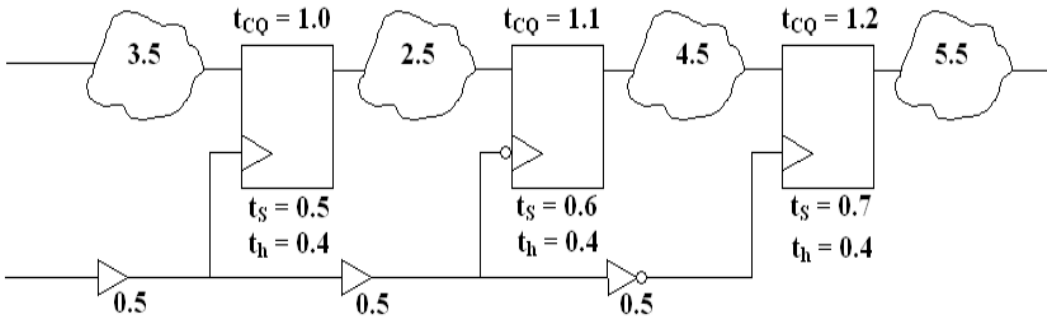
Solution

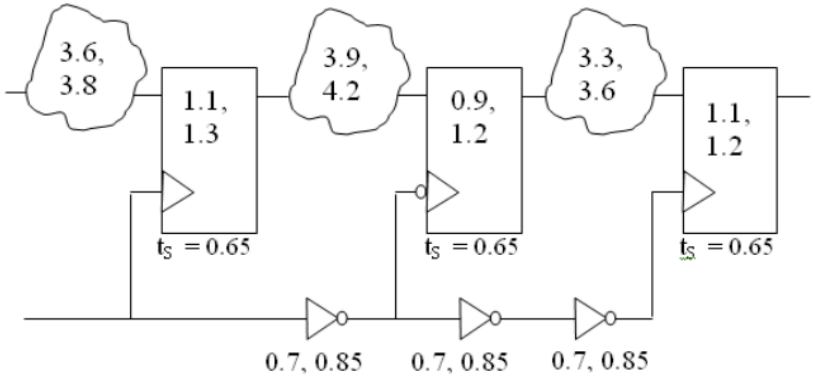
- Input – Reg Path
  - Arrival Time = 3 (assuming no input delay constraint)
  - Required Time = CP + 0.5 + 0.6 $-t_{setup}$ = CP + 1.1 –0.65
  - Equating the TWO, CP = 3 – 1.1 + 0.65 = 2.55
- Reg – Reg Path
  - Arrival Time = 0.5 + 0.6 + 1.4 + 4 = 6.5
  - Required Time = 0.5 + 0.7 + 0.8 + CP – 0.75 = CP + 1.25
  - Equating the two, CP = 6.5 – 1.25 = 5.25
- Reg – Output Path
  - Path is not constrained (output delay constraint)

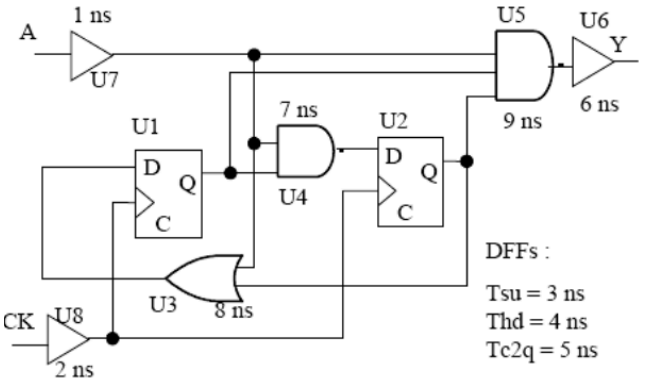
Solution

- Two time periods available from the calculations 5.25, 2.55. Though first path can work if clock time period is 5.25 the second path can not work if clock time period is 2.55
- Hence the maximum clock frequency at which the circuit can work is 1/5.25 GHz

More Problems

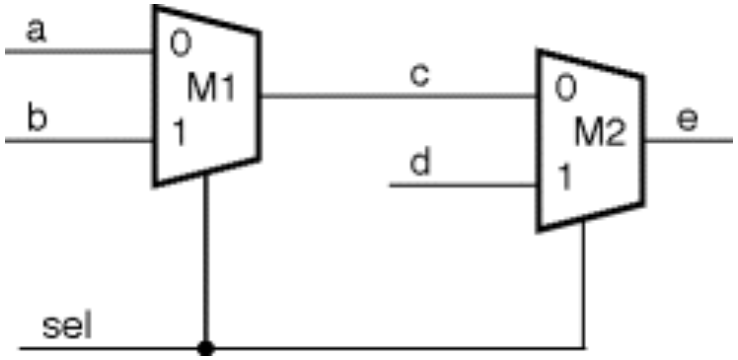






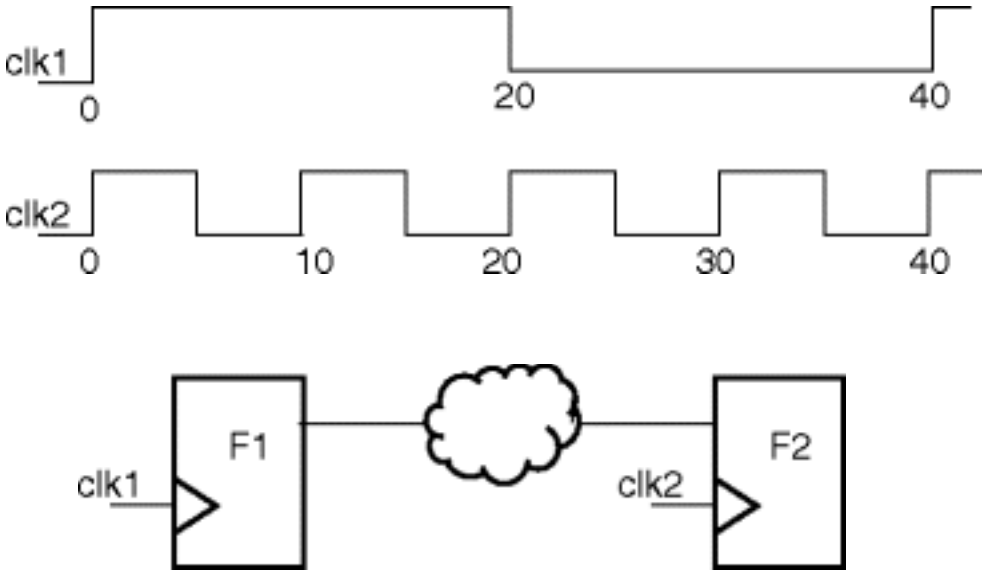
False Path

- Paths that physically exist in a design but are not logic/functional paths
- These paths never get sensitized under any input conditions
- False Paths Due to Paths Being Unsensitizable



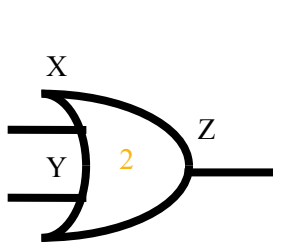
Multi-Cycle Path

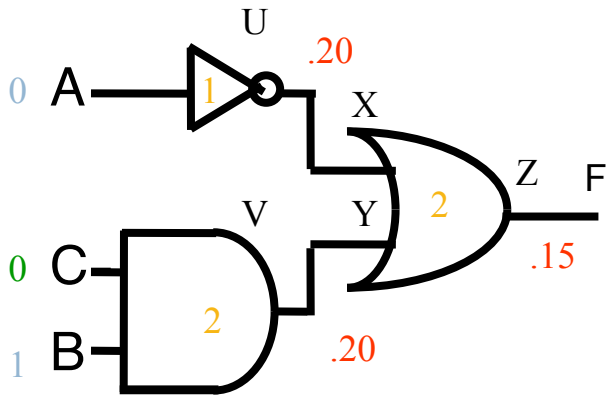
- Multicycle paths are paths which intentionally require more than one clock cycle to propagate.

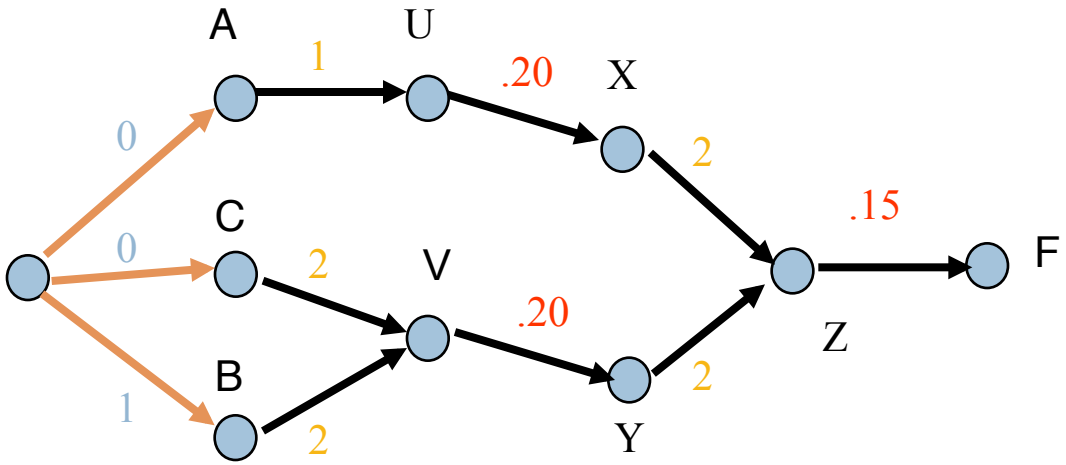


How is STA Performed?

- Timing Graph
  - Data paths with timing constraints
    - Starting from primary inputs/FF outputs
    - Ending at primary outputs/FF inputs
  - Represented by a labeled *directed* graph (DAG) $G = \langle V,E \rangle$
    - *Timing node V* ~ pin/primary input/output
    - *Timing edge E* ~ gate/wire delay

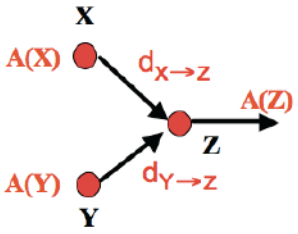




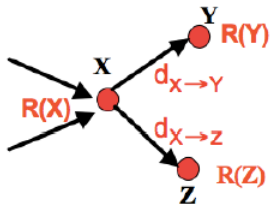


How is STA Performed?

- Timing Analysis Terminology
  - Actual arrival time (AAT): forward propagation
  - Required arrival time (RAT): backward propagation
  - Slack = RAT - AAT
    - A measure of how much timing margin exists at each node
    - Slack<0 ➔ timing violation
    - Can optimize a particular branch
    - Can trade slack for power, area, robustness



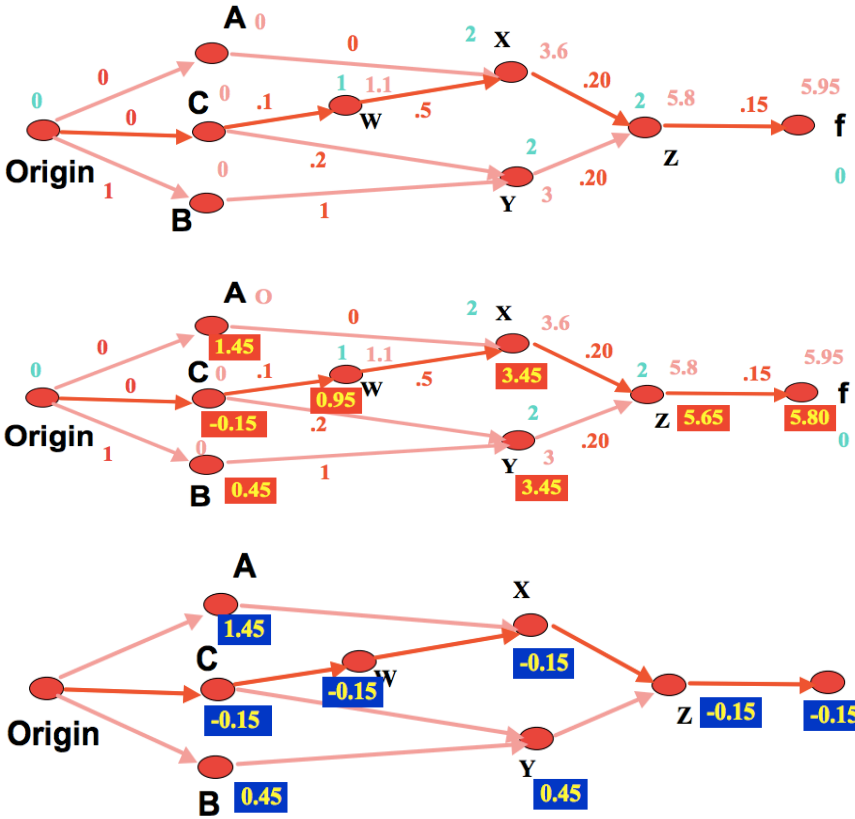
$$A(v) = \max_{u \in FI(v)} (A(u) + d_{u \to v})$$

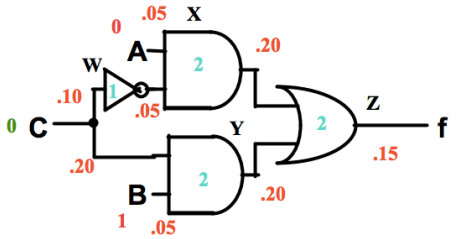


$$R(v) = \min_{u \in FO(v)} (R(u) - d_{v \to u})$$

How is STA Performed?

- Example

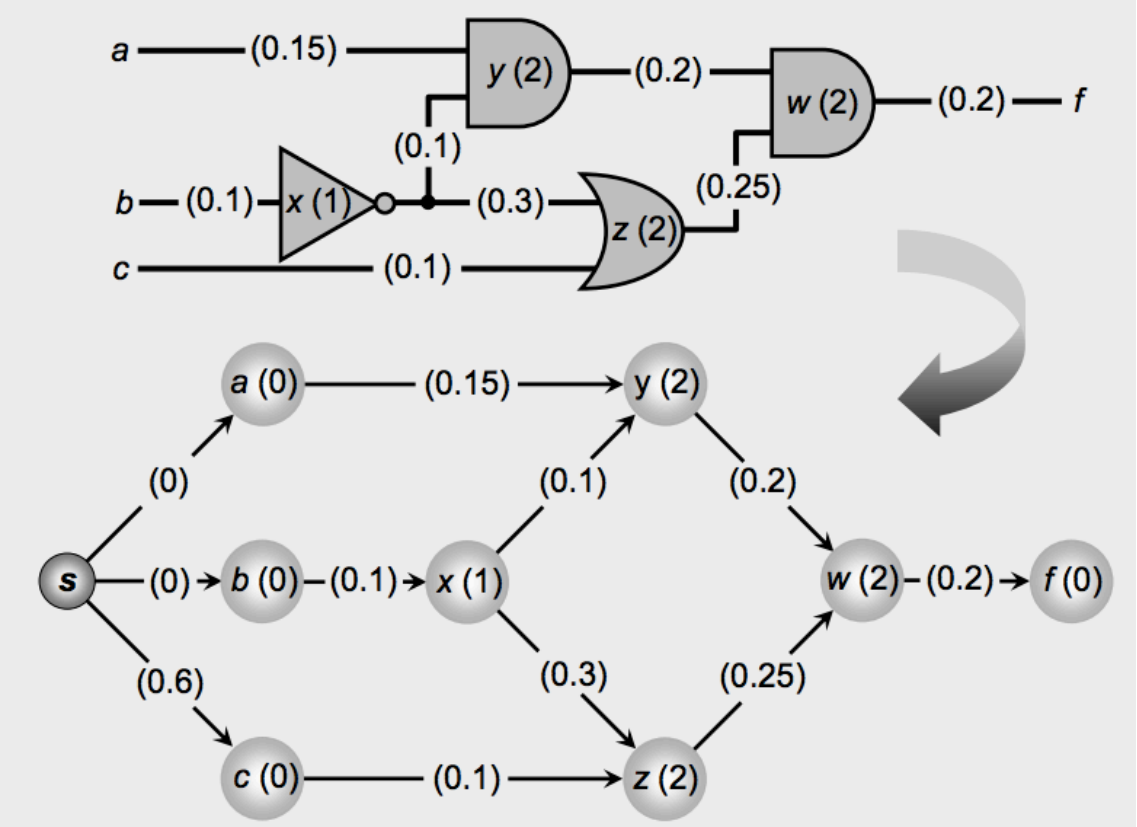




In this graph the timing arcs inside the gate are not included; assuming that the gate delay is represented by the delay of the longest timing arc

How is STA Performed?

- Another Example

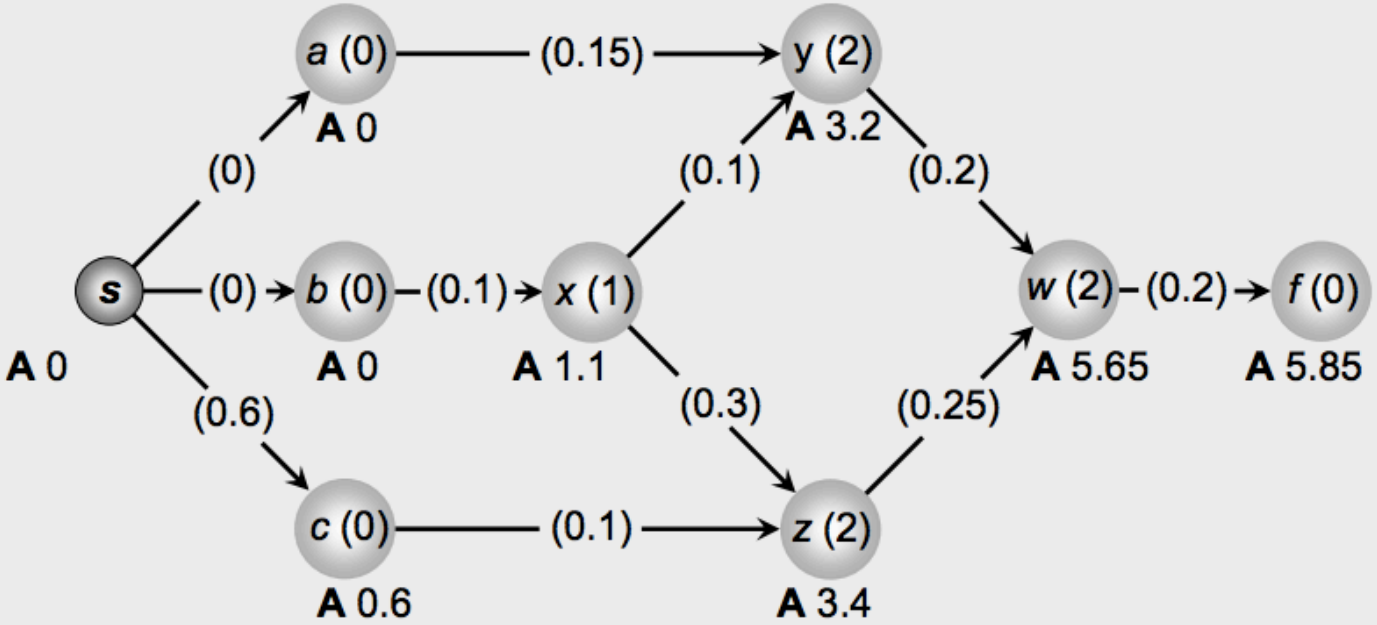


How is STA Performed?

$$AAT(v) = \max_{u \in FI(v)} \left(AAT(u) + t(u,v) \right)$$

where *FI*(*v*) is the fanin nodes, and *t*(*u*,*v*) is the delay between *u* and *v*

(AATs of inputs are given)

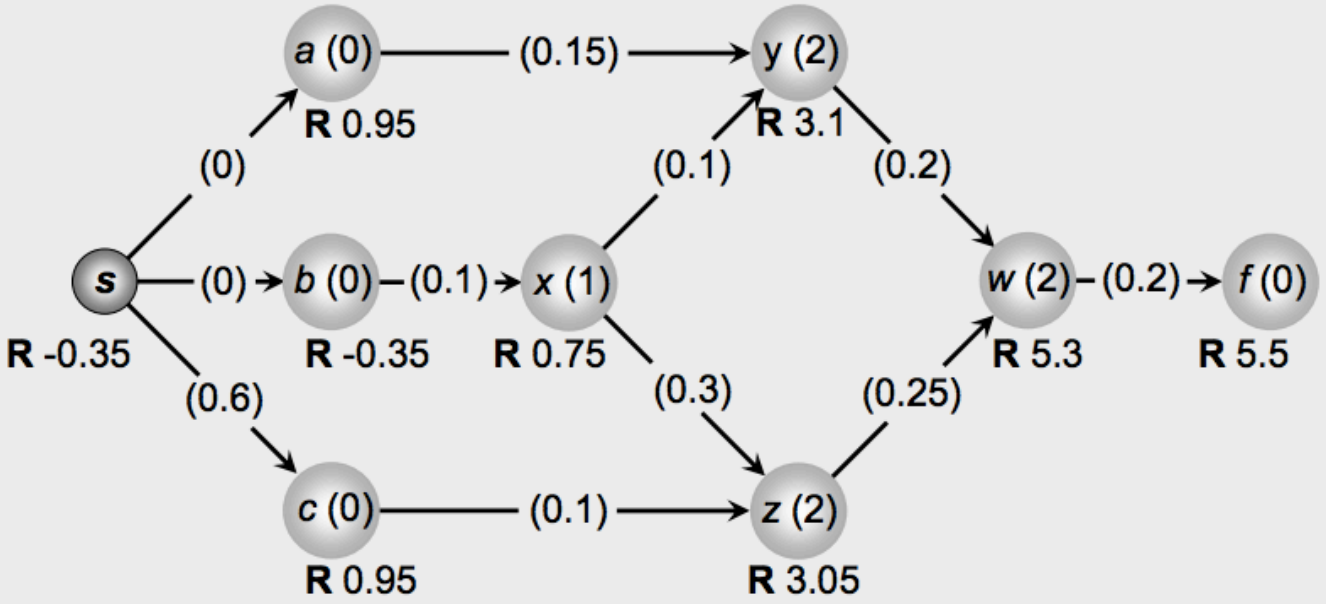


How is STA Performed?

Compute RATs at each node:

$$RAT(v) = \min_{u \in FO(v)} \left(RAT(u) - t(u,v) \right)$$

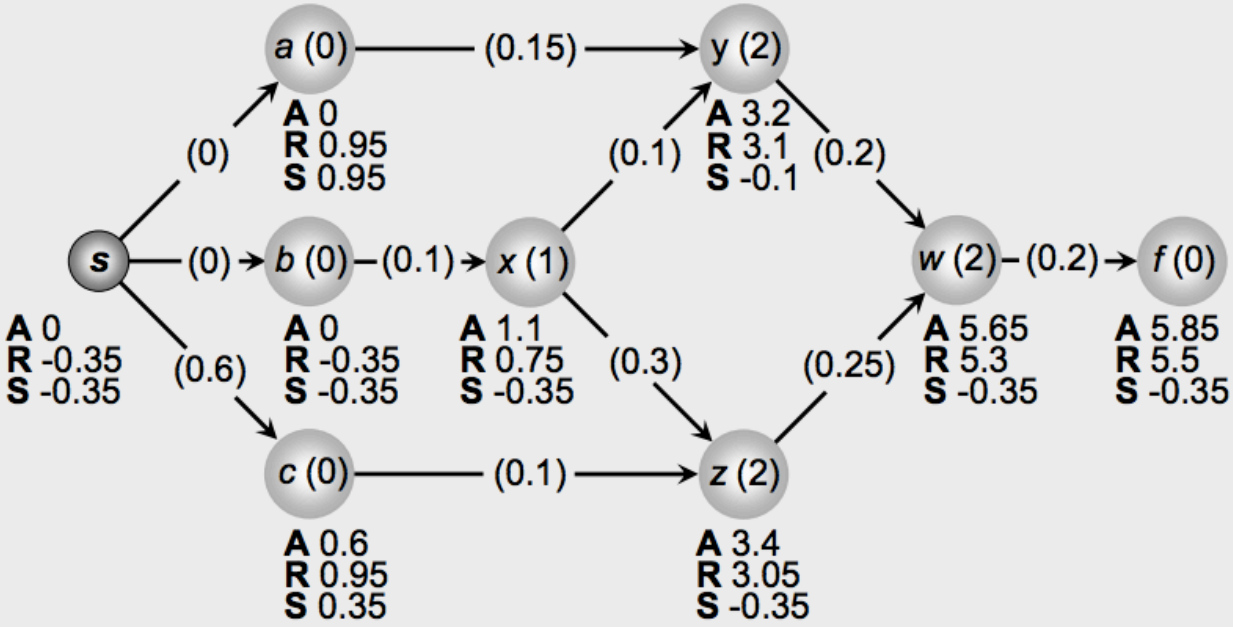
where *FO(v)* are the fanout nodes, and *t(u,v)* is the delay between *u* and *v*
(RATs of outputs are given)



How is STA Performed?

Compute slacks at each node:

$$slack(v) = RAT(v) - AAT(v)$$



Fixing Timing Violations

- Re-synthesis (Before Physical Synthesis)
  - Local synthesis transforms
- Timing-driven placement (During Physical Synthesis)
  - Critical net weighting
- Timing-driven routing (During Physical Synthesis)
  - Net ordering
  - Buffering
  - Topology optimization
- Post-route optimization (IPO)
  - Re-routing
  - Re-timing and useful clock skew
  - Sizing
  - Buffering

Local Synthesis Transforms

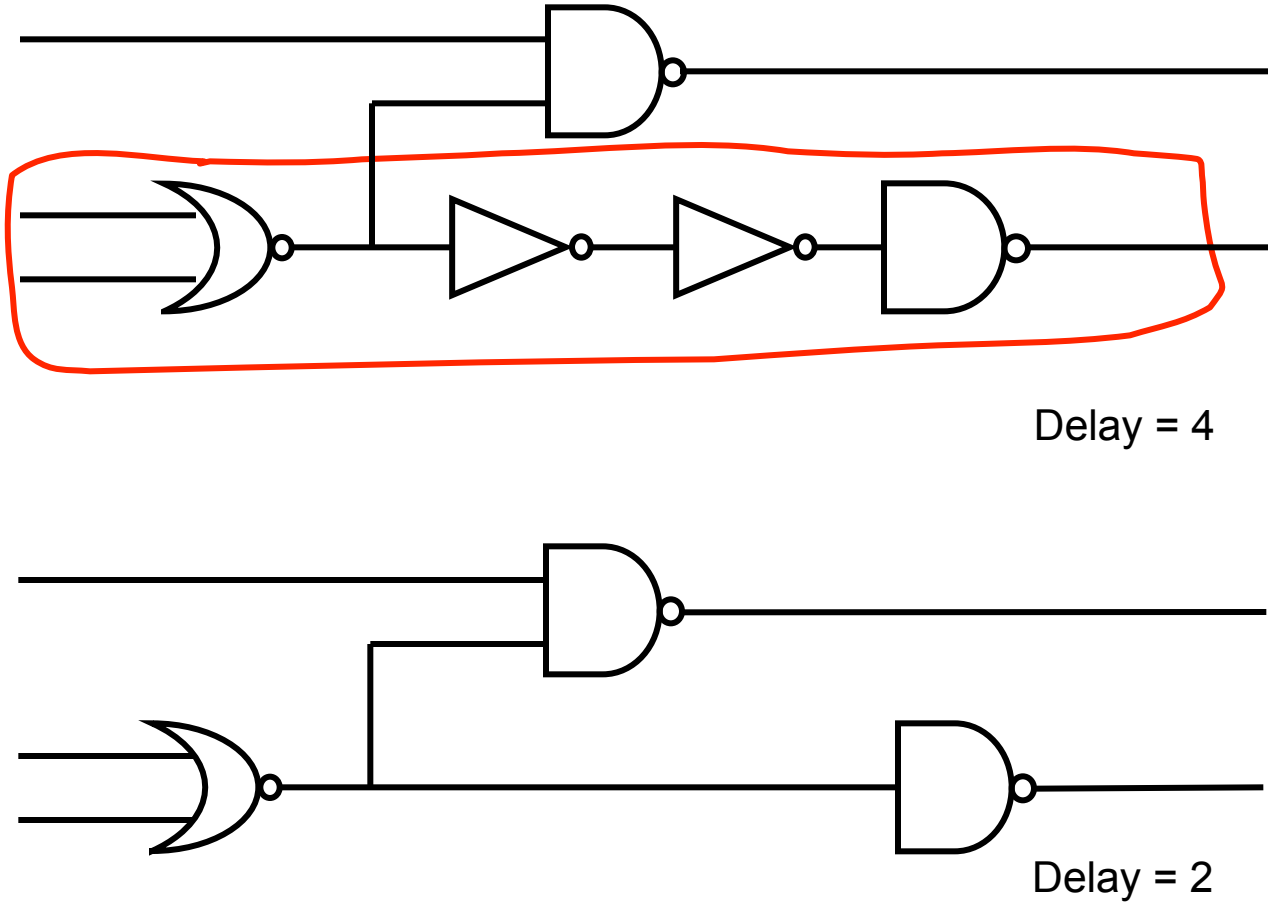
.....

Double Inverter

Removal

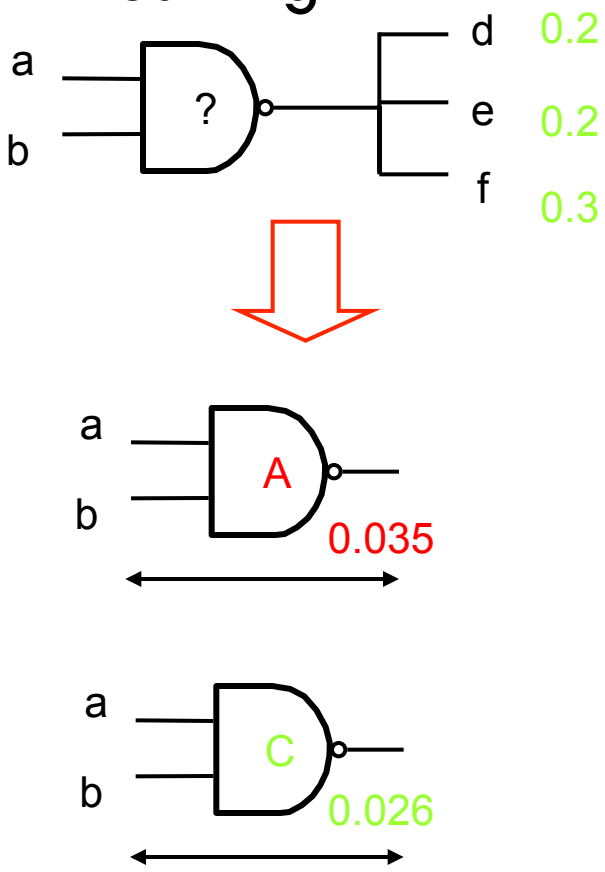
.....

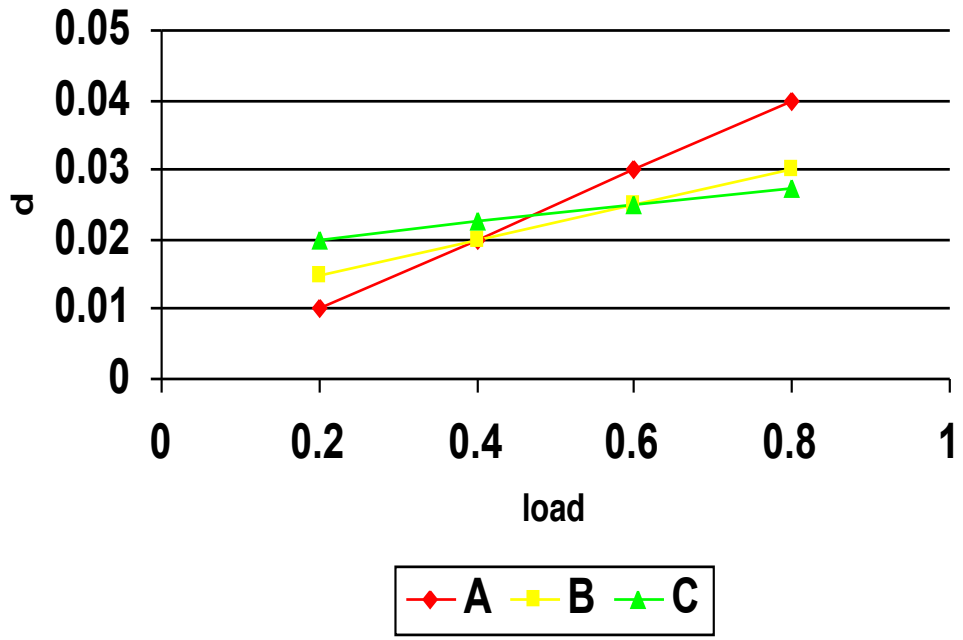
.....



Local Synthesis Transforms

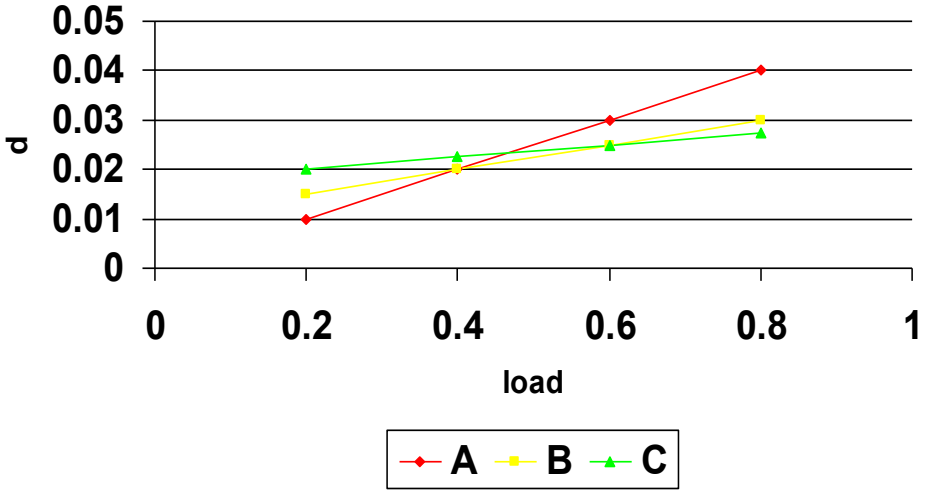
- Resizing





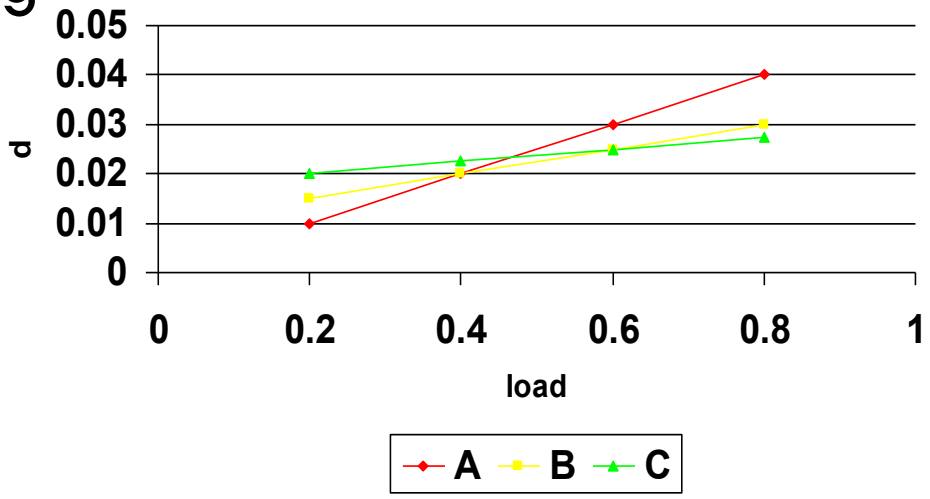
Local Synthesis Transforms

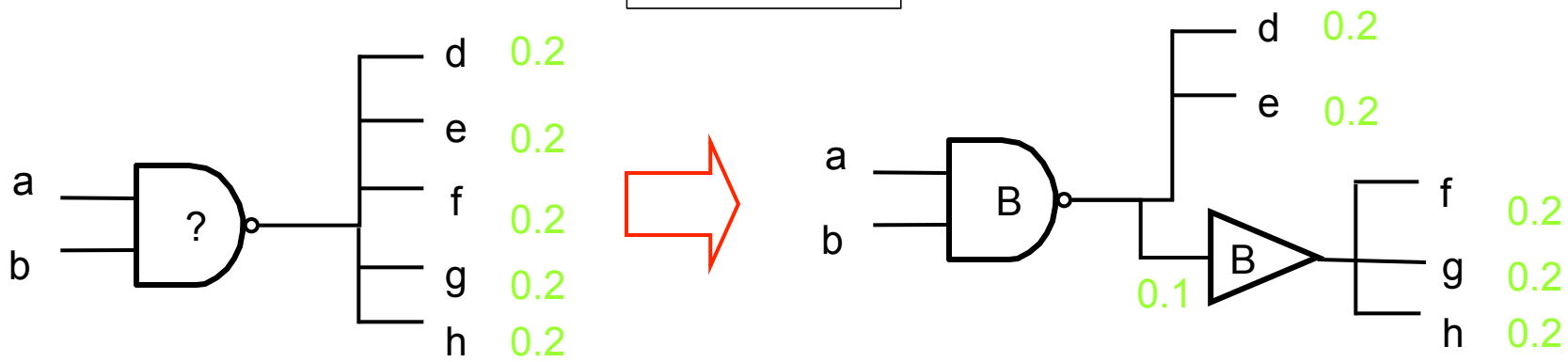
- Cloning



Local Synthesis Transforms

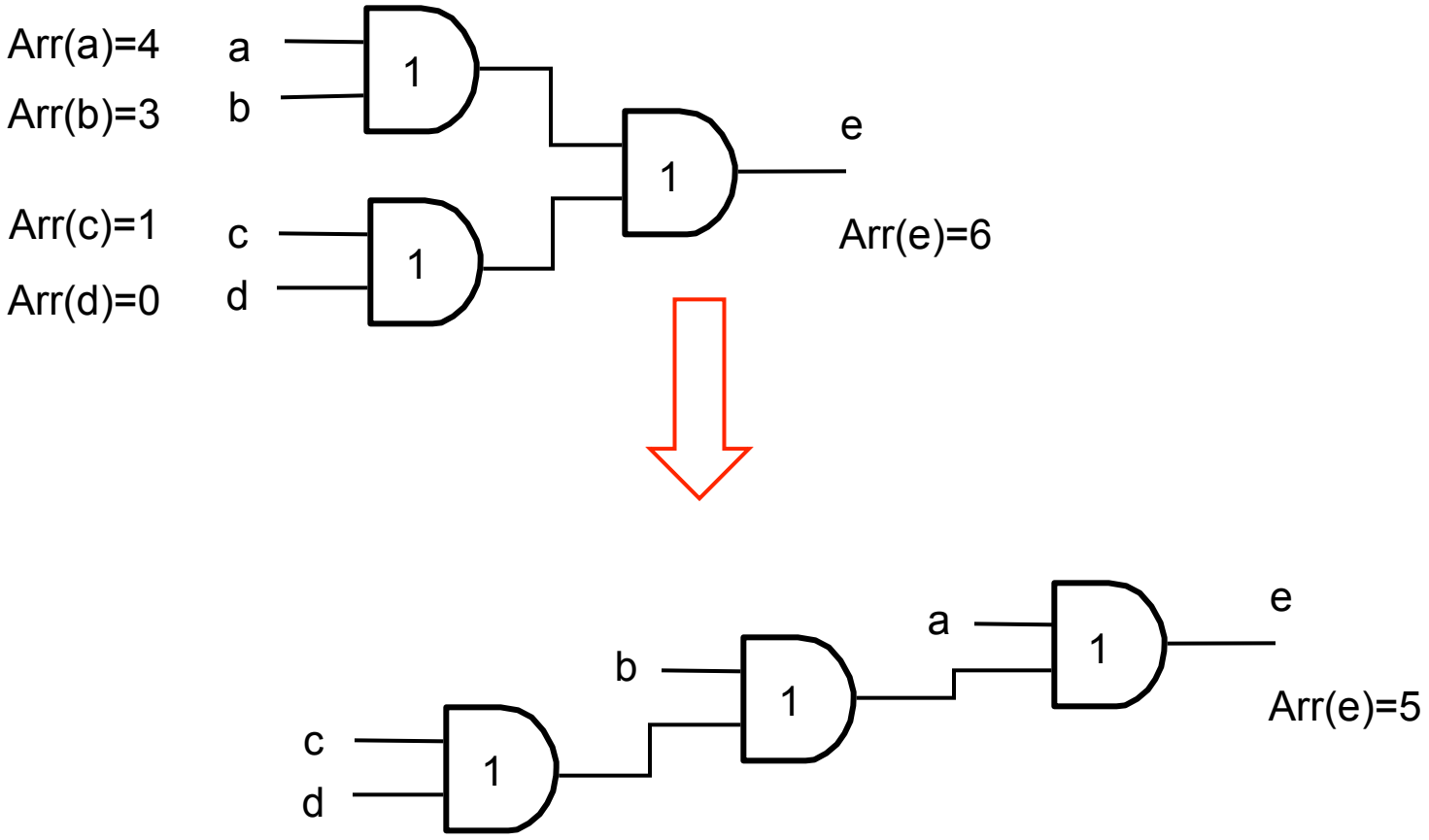
- Buffering





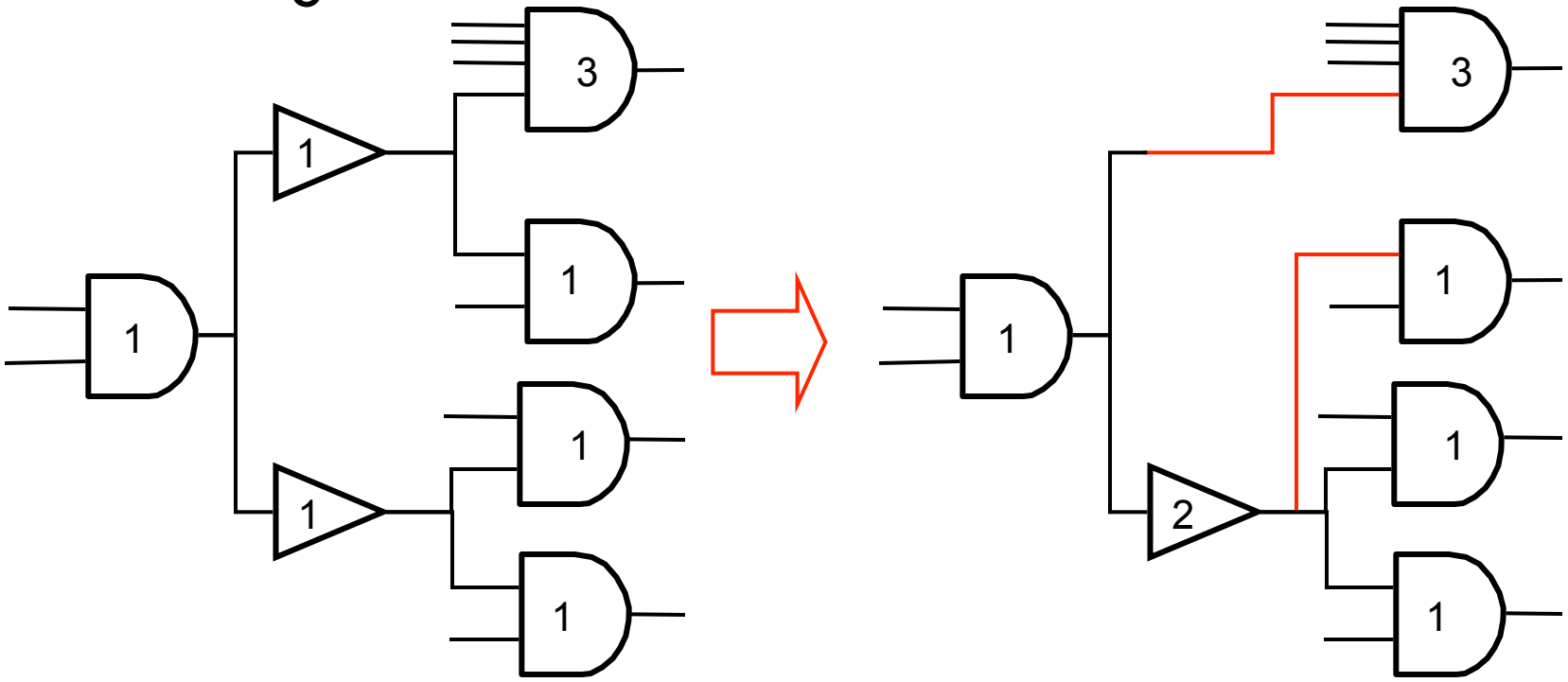
Local Synthesis Transforms

- Redesign Fan-in Tree



Local Synthesis Transforms

- Redesign Fan-out Tree

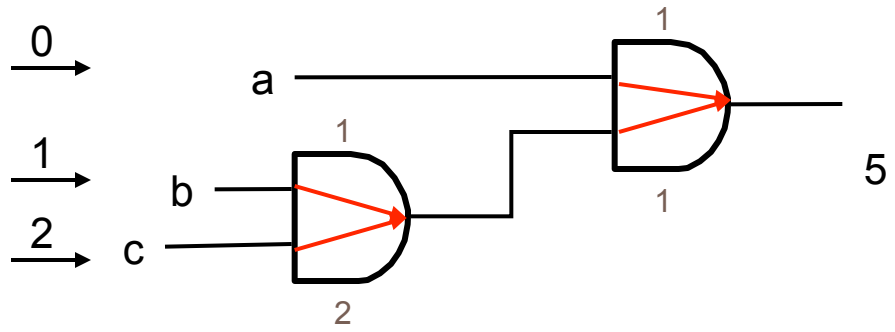


Longest Path = 5

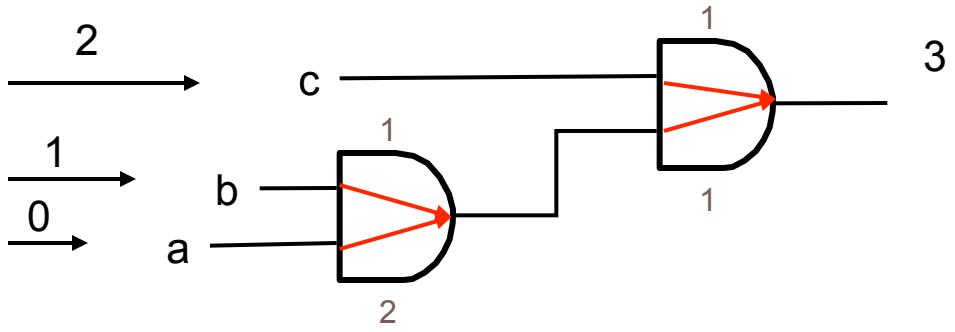
Longest Path = 4
Slowdown of buffer due to load

Local Synthesis Transforms

- Swap Pins

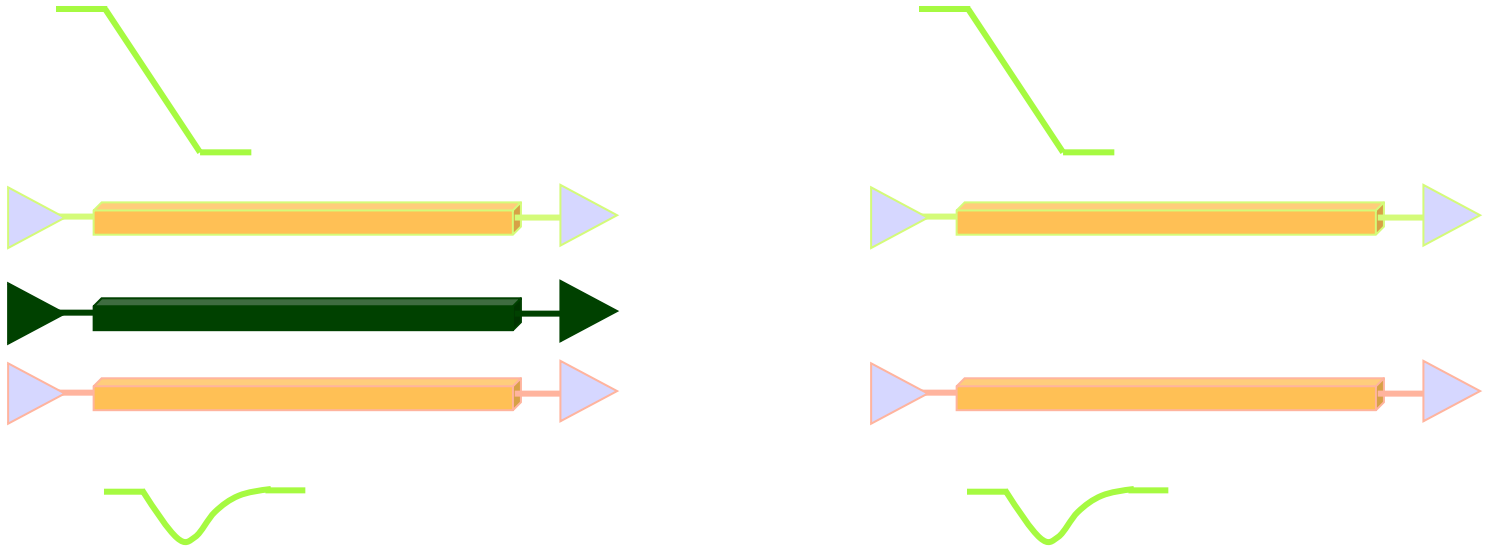


Simple sorting on arrival times and delay works



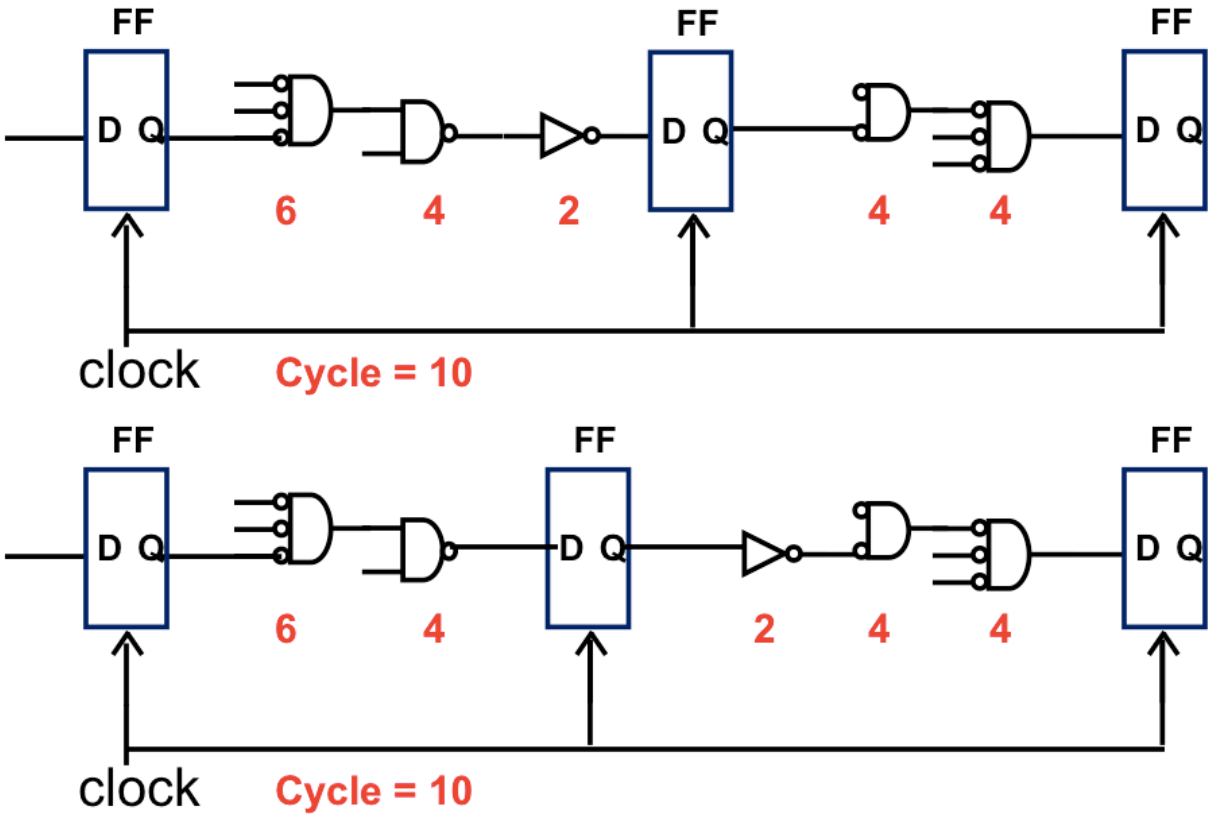
Post Layout Optimizations

- Re-route to reduce cross-talk effect



Post Layout Optimizations

- Re-timing



Post Layout Optimizations

- Useful Skew