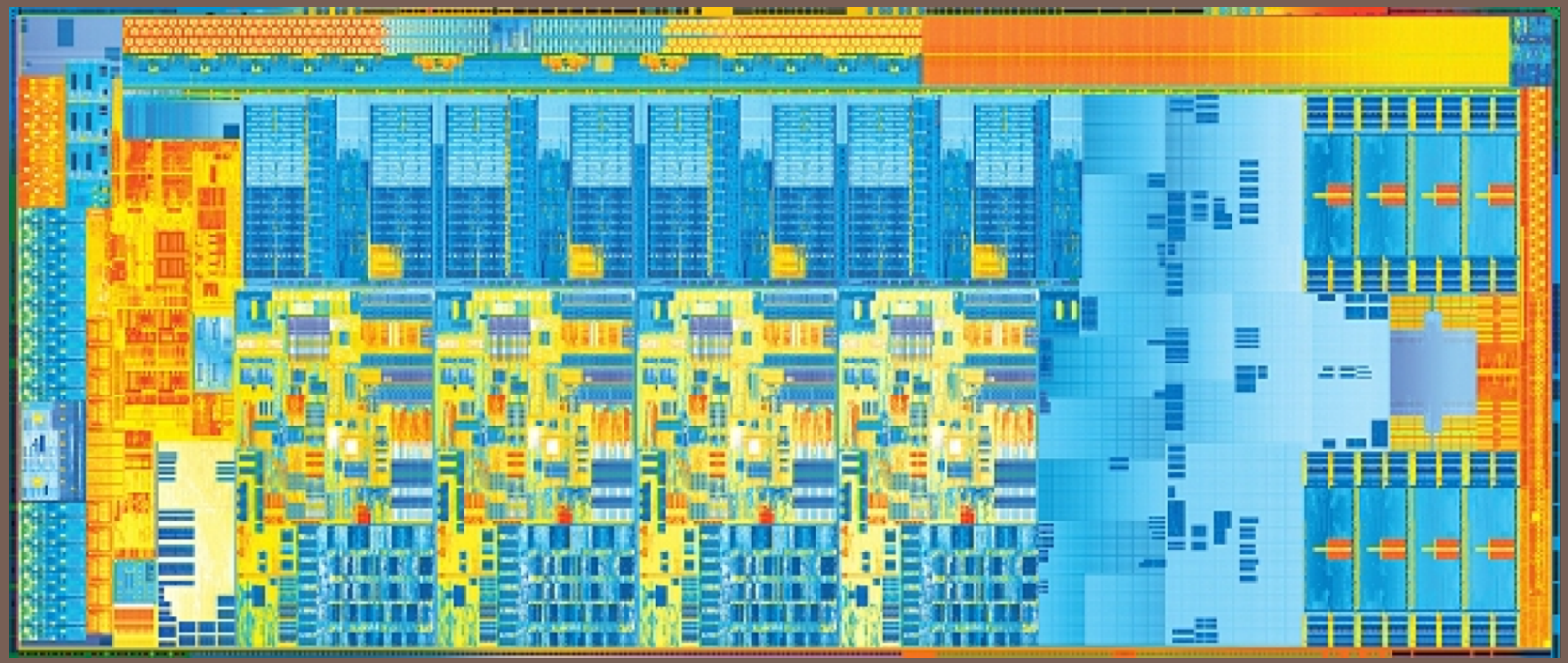


CSCE337 – Digital Design II

Lectures 19-21: Timing & Clocking Issues in Digital Systems

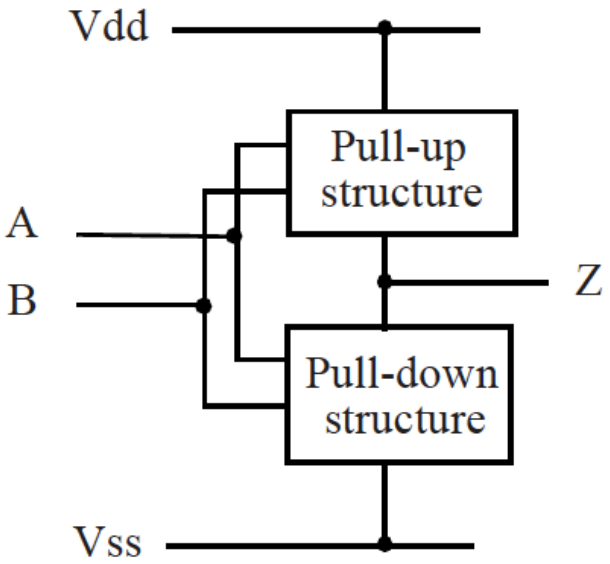
Spring 2019

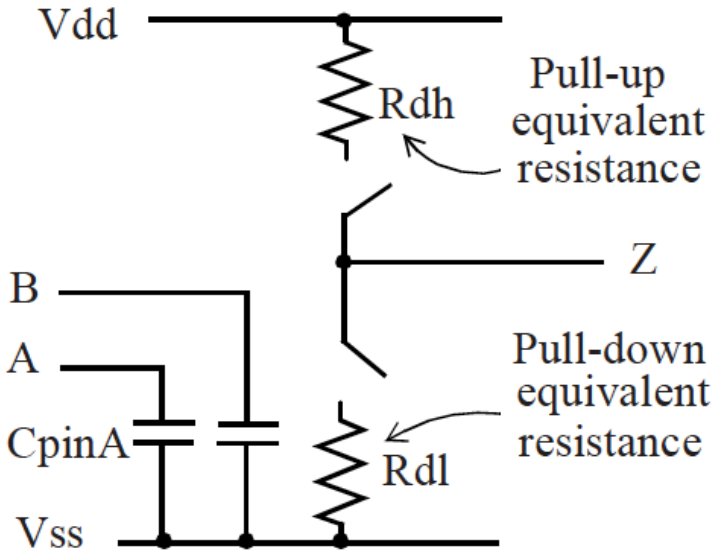
Mohamed Shalan

Agenda

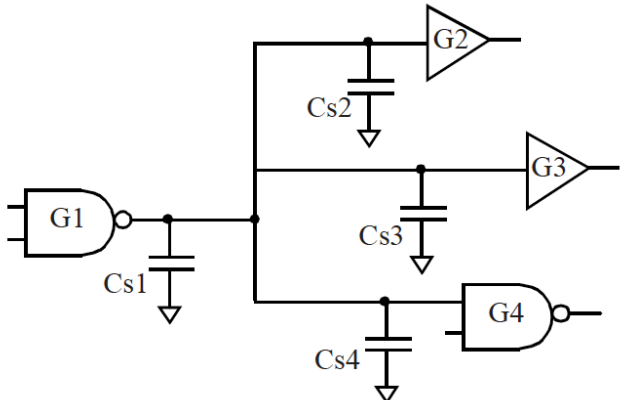
- Some Background Info
  - Clocking
  - Setup & Hold Times
  - Clock Signal
- Timing & Logic Synthesis
- Static Timing Analysis
- Fixing Timing Violations

CMOS Cell Model

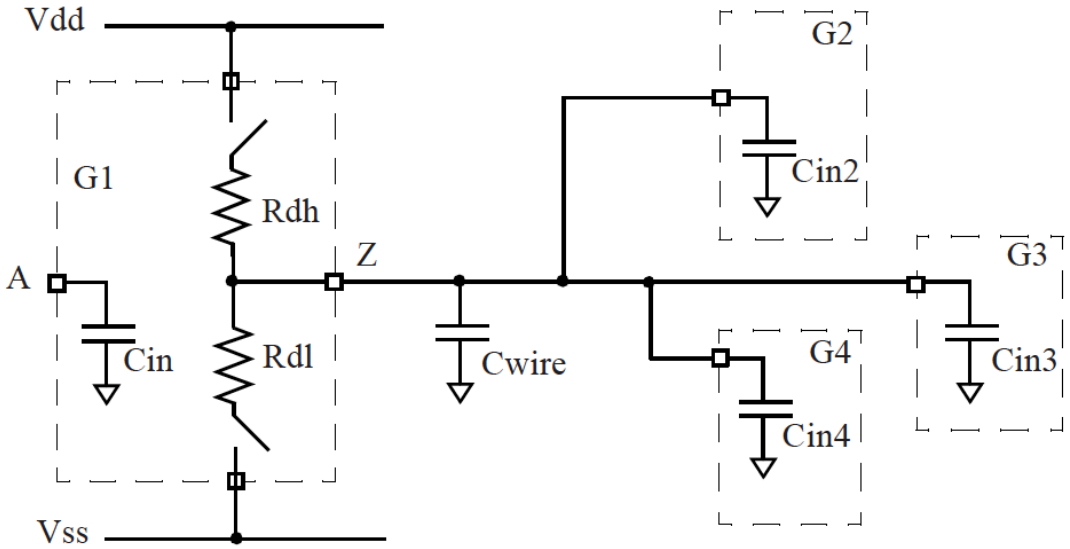




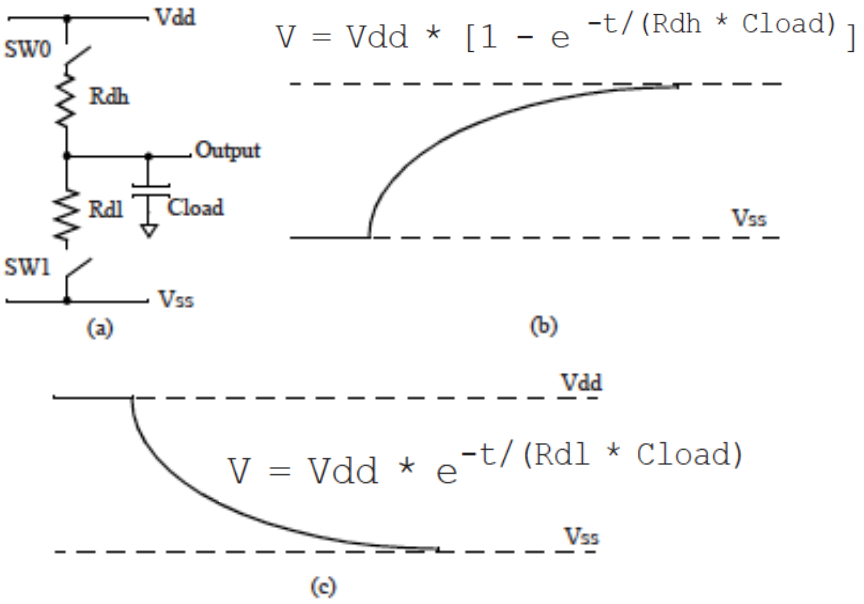
CMOS Delay Model

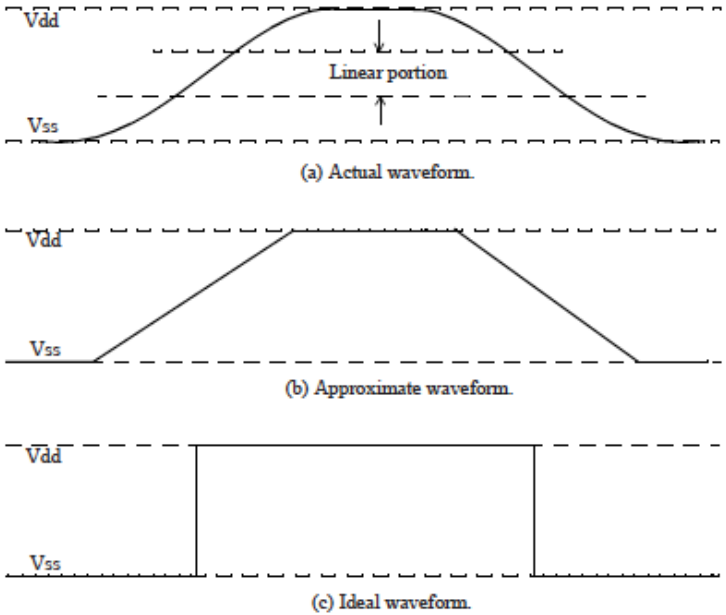


- $C_{wire} = C_{s1} + C_{s2} + C_{s3} + C_{s4}$
- tpd = R x C_{load}
- R is either Rdl or Rdh
- $C_{load} = C_{wire} + C_{in2} + C_{in3} + C_{in4}$

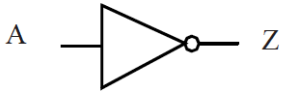


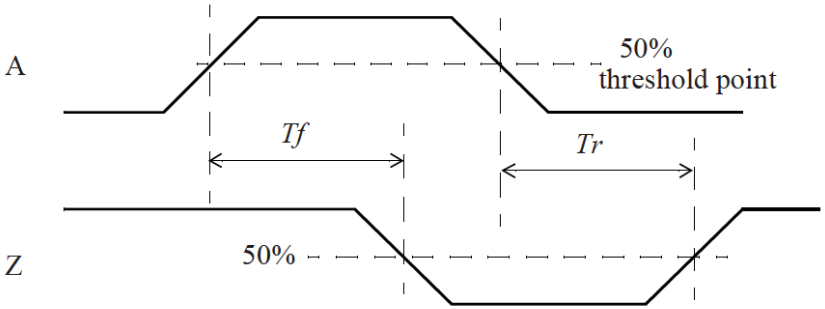
CMOS Delay Model

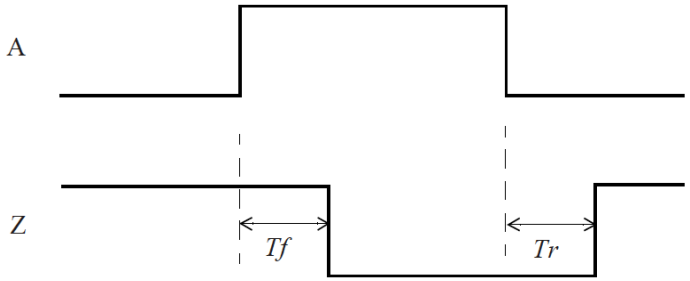




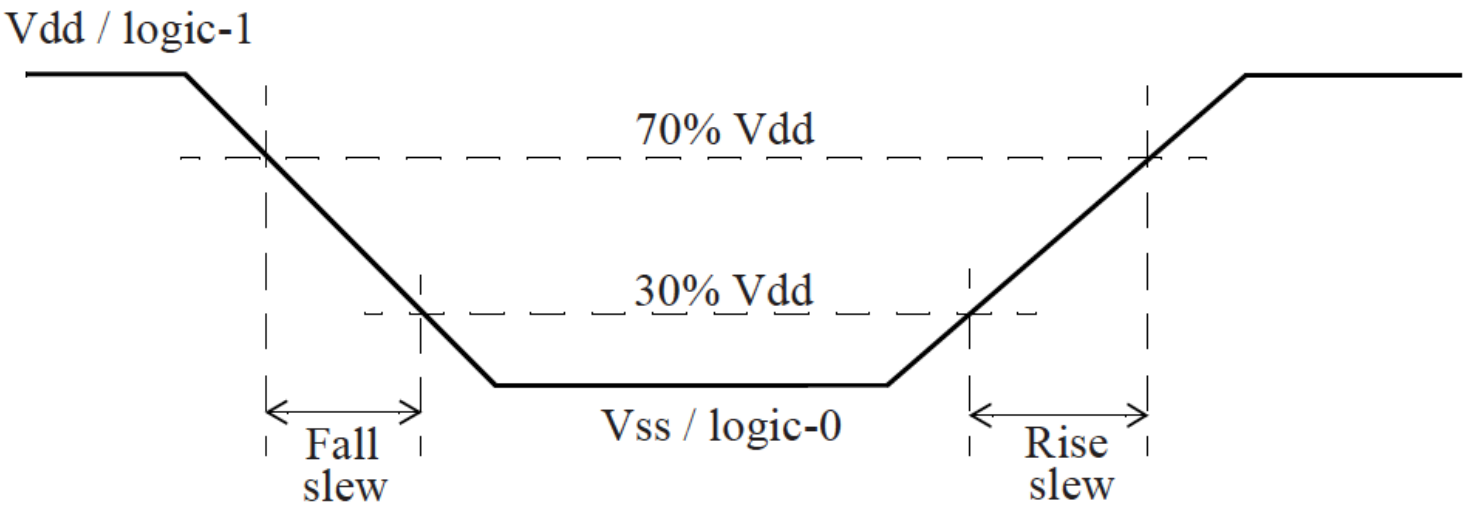
Rise and Fall Times







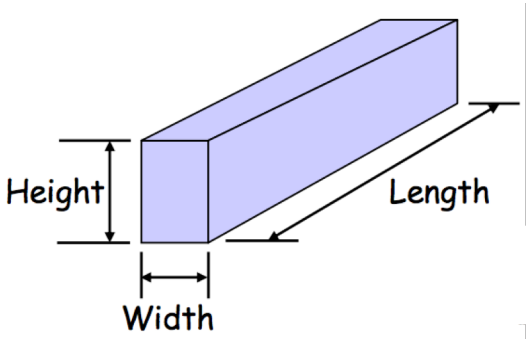
Slew (Transition Time)



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

Wire Model

- Wire Resistance
  - Resistance = L x Resistivity / (W x H)
  - Rs = (Resistivity/H) (sheet resistance)
  - For any given process, every metal layer Rs is given as Ω/square
    - The height (thickness) is fixed for a given metal layer
  - For TSMC180nm Process
    - M1: 0.08 Ω/square
      - 0.5µm x 1mm wire ➔ 160Ω
    - M4: 0.03 Ω/square
      - 0.5µm x 1mm wire ➔ 60Ω



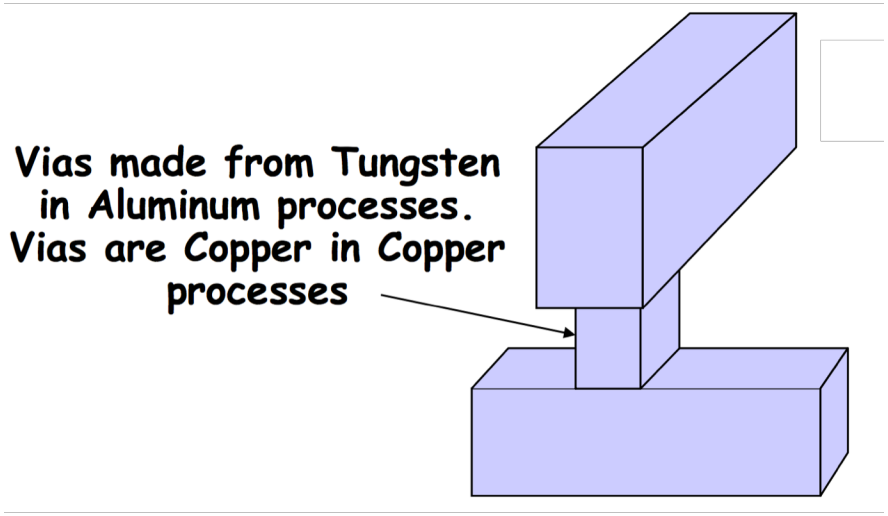
Metal	Resistivity (ρ)
Aluminum	2.8×10^{-8} Ωm
Copper	1.7×10^{-8} Ωm
Silver	1.6×10^{-8} Ωm

Copper is used for processes less than 180nm

VIA Resistance

- Via resistance significant
- TSMC 0.18µm 6-Al

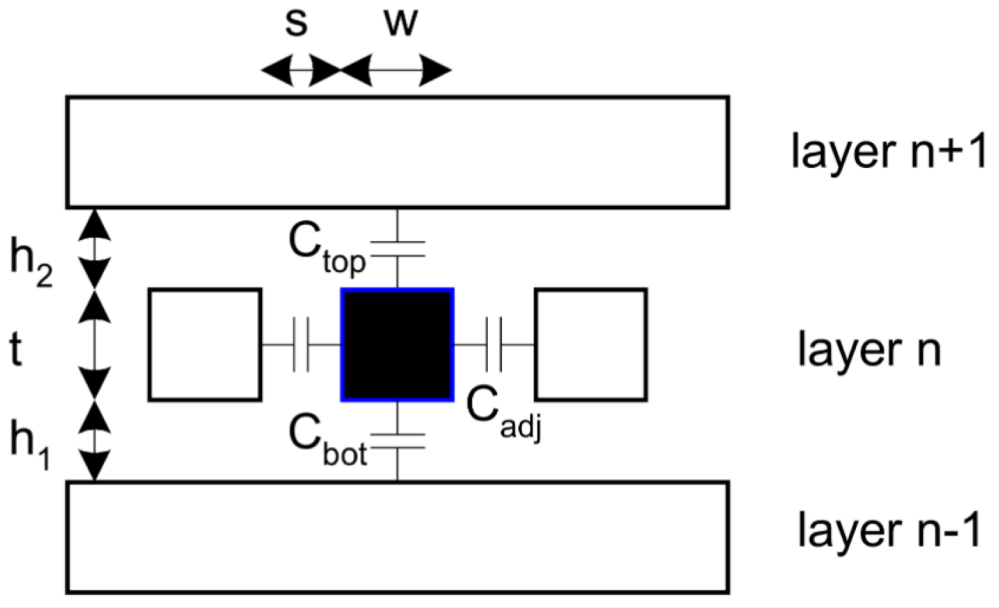
Diff-M1	11.0 Ω
Poly-M1	10.4 Ω
M2-M1	4.5 Ω
M3-M1	9.5 Ω
M4-M1	15.0 Ω
M5-M1	19.6 Ω
M6-M1	21.8 Ω

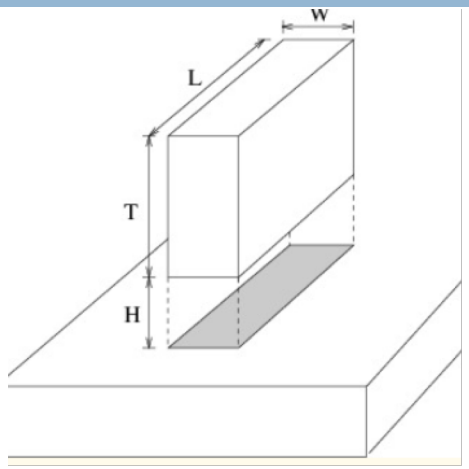


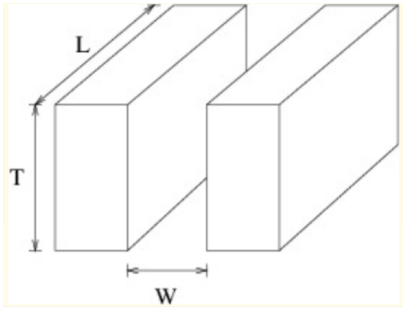
Use many contacts
for lower R

Wire Model

- Wire Capacitance







- $C_{total} = C_{top} + C_{bot} + 2C_{adj}$
- Every process defines C/μ (0.2fF/μm for M2 in 180nm)

Capacitance Trends

- Parallel plate equation: $C = \varepsilon_{ox} A/d$
  - Wires are not parallel plates, but obey trends
  - Increasing area (W, t) increases capacitance
  - Increasing distance (s, h) decreases capacitance
- Dielectric constant
  - $\varepsilon_{ox} = k\varepsilon_0$
    - $\varepsilon_0 = 8.85 \times 10^{-14}$ F/cm
    - k = 3.9 for SiO_2
- Processes are starting to use low-k dielectrics
  - $k \approx 3$ (or less) as dielectrics use air pockets

65nm Wire Attributes

Layer	W (μm)	T (μm)	H (μm)	Pitch (μm)	r (Ω)	c_{pp} ($fF/\mu m^2 \times 10^{-3}$)	c_{fringe} ($fF/\mu m \times 10^{-3}$)
M1	0.09	0.18	0.59	0.20	0.16	0.171	0.089
M2	0.10	0.22	0.95	0.20	0.14	0.232	0.078
M3	0.10	0.22	1.34	0.20	0.14	0.232	0.078
M4	0.10	0.22	1.74	0.20	0.14	0.232	0.078
M5	0.10	0.22	2.13	0.20	0.14	0.232	0.078
M6	0.10	0.22	2.53	0.20	0.14	0.232	0.078
M7	0.10	0.22	2.92	0.20	0.14	0.232	0.081
M8	0.40	0.90	3.74	0.80	0.02	0.063	0.092
M9	0.40	0.90	5.23	0.80	0.02	0.063	0.096

65nm Tech LEF


```
LAYER M4
        TYPE ROUTING ;
        DIRECTION VERTICAL ;
        PITCH 0.200 ;
        HEIGHT 1.7350 ;
        THICKNESS 0.2200 ;
        WIDTH 0.1 ;
        MAXWIDTH 12.0 ;
        ...
        ...
        RESISTANCE RPERSQ 0.1399000000 ;
        CAPACITANCE CPERSQDIST 0.0002320000 ;
        EDGECAPACITANCE 0.0000778000 ;
END M4
```

65nm Example (L=12mm)

$$RPERSQ = 0.1399\Omega$$
$$CPERSQDIST = 0.000232pF/\mu m^2$$
$$EDGECAPACITANCE = 0.0000778pF/\mu m$$

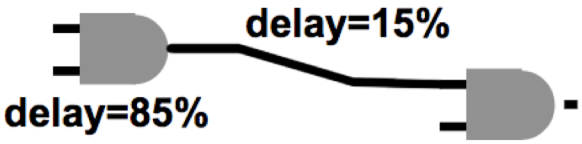
$$R = RPERSQ \times \frac{L}{W} = 0.1399\Omega \times \frac{12mm}{0.1\mu m} = 17K\Omega$$

$$C = CPERSQDIST \times L \times W + EDGECAPACITANCE \times L$$
$$= 0.000232\frac{pF}{\mu m^2} \times 12mm \times 0.1\mu m + 0.0000778\frac{pF}{\mu m} \times 12mm$$
$$= 0.278pF + 0.934pF = 1.2pF$$

80% edge capacitance

Why Bother?

- Technology Trends



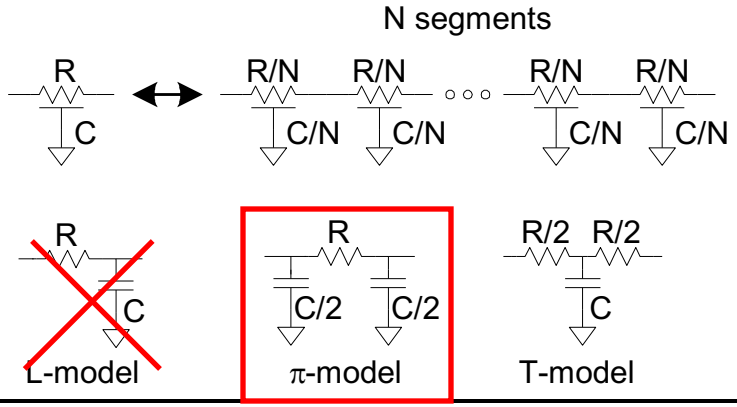
80's – early 90's

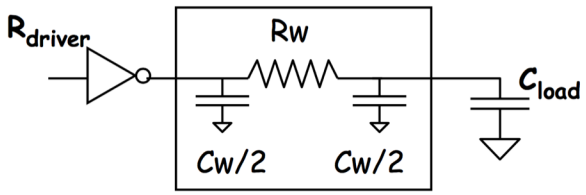
250nm

Now

Wire Delay Model

- Wire has distributed R and C per unit length
  - wire delay increases quadratically with length
  - edge rate also degrades quadratically with length
- Simple lumped π (pi) model gives reasonable approximation
  - Actually, the 3-segment π-model accurate to 3% in simulation
  - Rw is lumped resistance of wire
  - Cw is lumped capacitance (put half at each end)





$$\text{Delay} = \text{Rdriver} \times \frac{\text{Cw}}{2} + (\text{Rdriver} + \text{Rw}) \times \left(\frac{\text{Cw}}{2} + \text{Cload} \right)$$

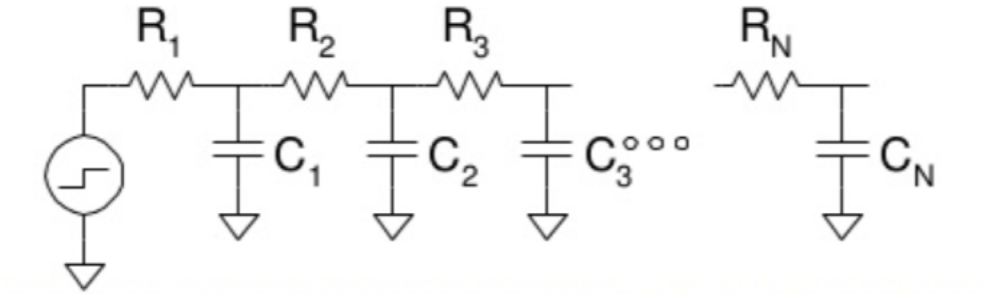
Wire Delay Model

- Example:
  - In 0.18µm TSMC, 5x minimum inverter with effective resistance of 3 kΩ, driving FO4 load (25fF)
    - Delay = Rdriver x Cload = 75 ps
  - Now add 1mm M1 wire, 0.25µm wide, and a via
    - Rw=320Ω wire+22Ω via = 344Ω
    - Cw=160fF

$$\begin{aligned} \text{Delay} &= \text{Rdriver} \times \frac{\text{Cw}}{2} + (\text{Rdriver} + \text{Rw}) \times \left(\frac{\text{Cw}}{2} + \text{Cload} \right) \\ &= 3\text{k}\Omega \times 80\text{fF} + (3\text{k}\Omega + 344\Omega) \times (80\text{fF} + 25\text{fF}) \\ &= 591\text{ps} \end{aligned}$$

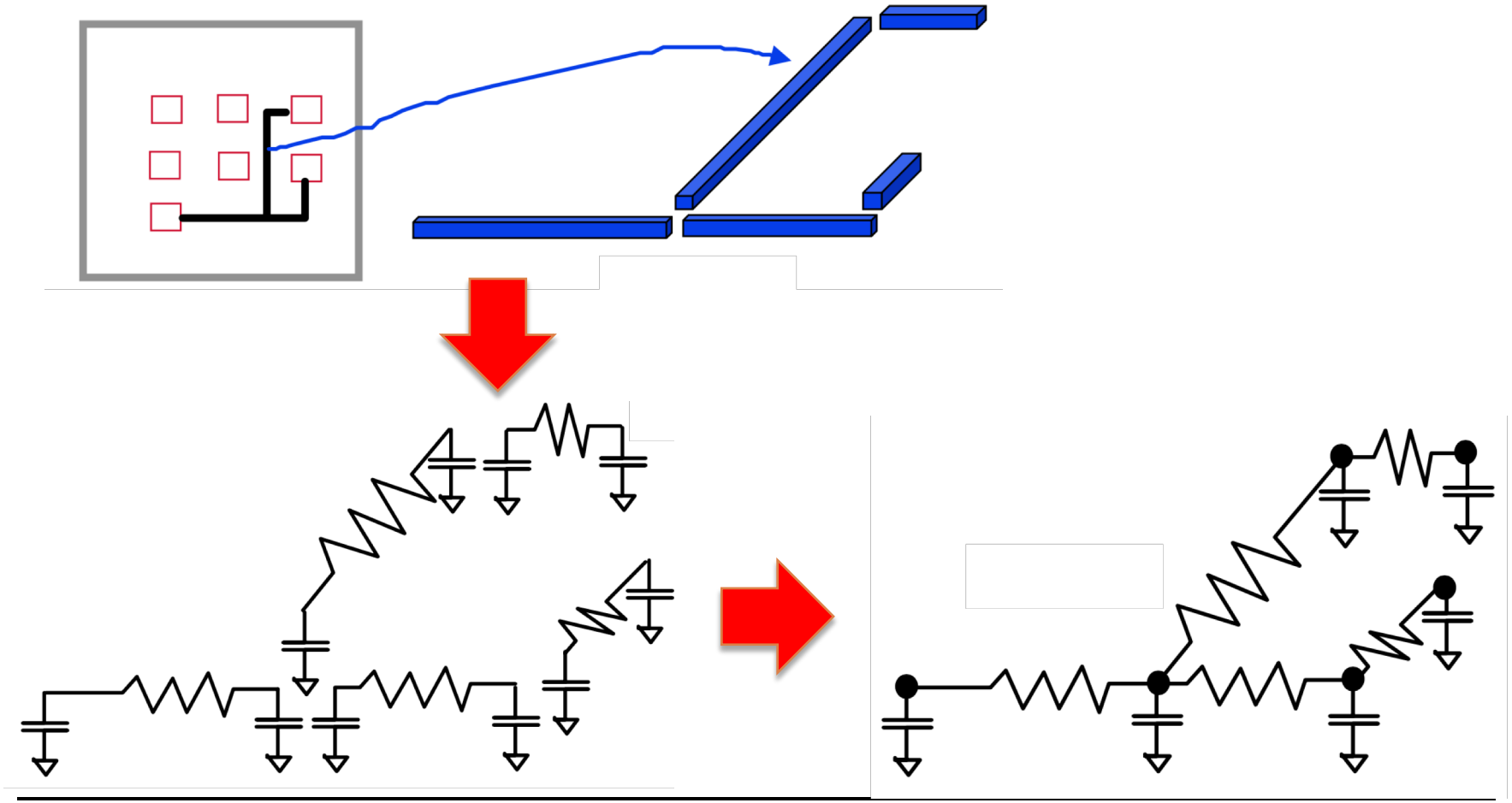
Elmore Delay

- A good approximation for the RC delay of complex RC circuits
  - $t_{pd} = \Sigma R_{i\text{-to-src}} * C_i$
- Example

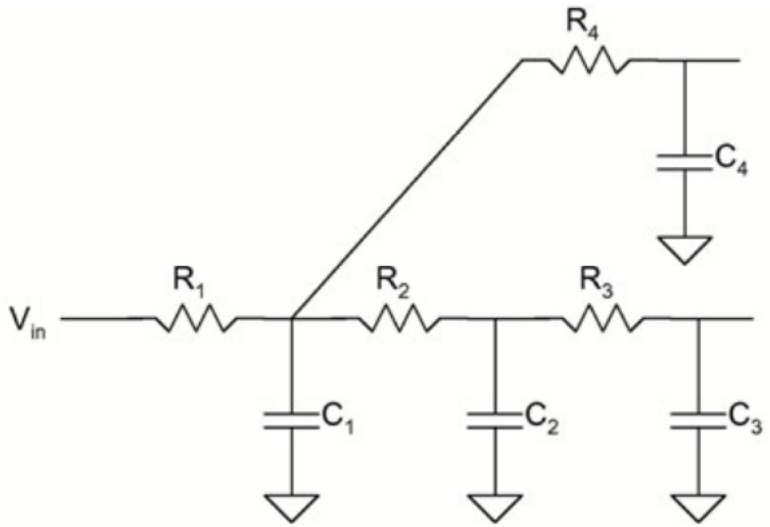


  - $t_{pd} = R_1C_1 + (R_1+R_2)C_2 + (R_1+R_2+R_3)C_3 + \ldots$

RC Tree



Elmore Delay for RC Tree

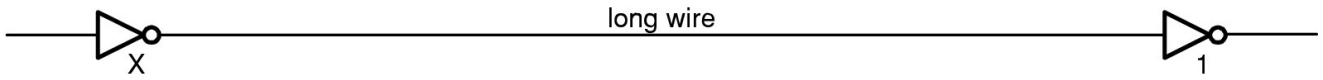


$$t_{d3} = R_1C_1 + (R_1 + R_2)C_2 + (R_1 + R_2 + R_3)C_3 + R_1C_4$$

$$t_{d4} = R_1C_1 + R_1C_2 + R_1C_3 + (R_1 + R_4)C_4$$

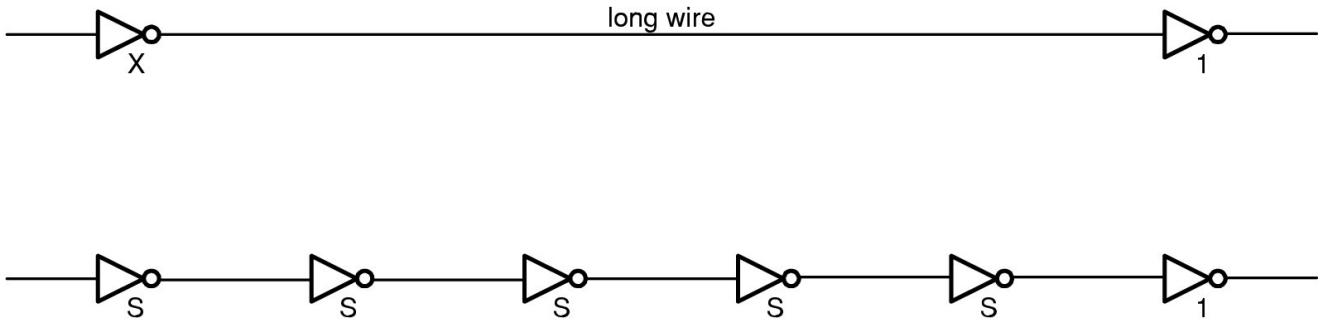
Wire Delay

- How to deal with long wires connecting gates?



Wire Delay

- How to deal with long wires connecting gates?
  - Buffers insertion.



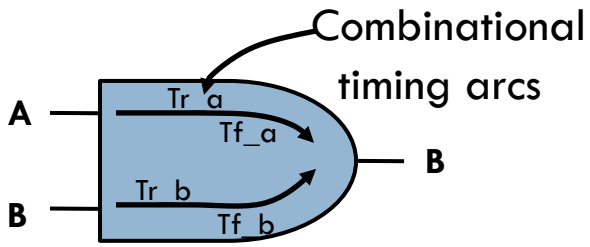
Liberty (.lib) Format –Revisted

- The .lib file is an ASCII representation of the standard cells electrical models (timing, power, …)
- The timing and power parameters are obtained by simulating the cells under a variety of conditions (characterization) and the data is represented in the .lib format

Timing Models for Combinational Cells

- Pin Capacitance
  - Input and output cells can specify capacitance at the pin. Generally capacitance is specified only for the cell inputs and not for the outputs.
  - There are also specified separate values for rise_capacitance and fall_capacitance, which refer to the values used for rising and falling transitions at the pin INP1

```
pin (INP1)  {
   capacitance :  0.5;
   rise_capacitance : 0.5;
   rise_capacitance_range : (0.48, 0.52);
   fall_capacitance : 0.45;
   fall_capacitance_range :  (0.435, 0.46);
   .  .  .
  }
```



Timing Models for Combinational Cells

- Unate
  - The timing arc is negative unate when the corresponding output pin transition direction is opposite of the input pin transition direction
  - The timing arc is positive unate when the output transition is in the same direction as the input transition
  - Ex:
    - For 2 input AND cell, Both the timing arcs for AND are positive_unate, therefore an input rise corresponds to an output rise.

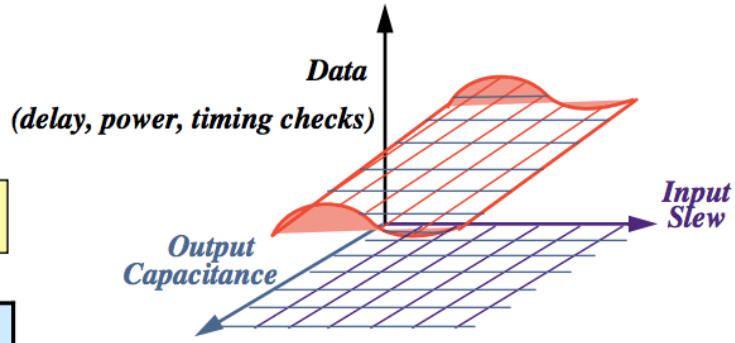
Liberty Timing Model (NLDM)

- The Non-Linear Delay (or NLDM) model is presented in a two-dimensional form, with the two independent variable:
  - The input transition time
  - The output load capacitance
- In the example, the delays of the output pin OUT are described. This portion of the cell description contains the rising and falling delay models for the timing arc from pin INP1 to pin OUT

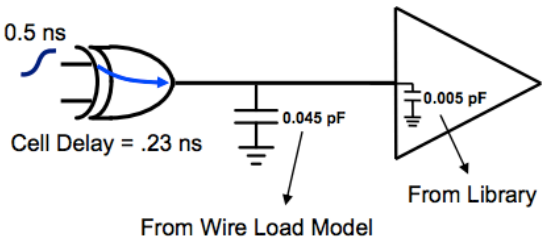
```
pin (OUT) {
  max_transition : 1.0;
  timing () {
    related_pin : "INP1";
    timing_sense : negative_unate;
    cell_rise (delay_template_3x3) {
        index_1 ("0.1, 0.3, 0.7"); /* Input transition */
        index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
        values ( /* 0.16, 0.35, 1.43" */ \
          /* 0.1 */ "0.0513, 0.1537, 0.5280" , \
          /* 0.3 */ "0.1018, 0.2327, 0.6476", \
          /* 0.7 */ "0.1334, 0.2973, 0.7252", ) ;
}
    cell_fall (delay_template_3x3) {
        index_1 ("0.1, 0.3, 0.7"); /* Input transition */
        index_2 ("0.16, 0.35, 1.43") /* Output capacitance */
        values ( /* 0.16 0.35 1.43 */ \
          /* 0.1 */ "0.0617, 0.1537, 0.5280", \
          /* 0.3 */ "0.0918, 0.2027, 0.5676", \
          /* 0.7 */ "0.1034, 0.2273, 0.6452" ) ;
}
```

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)



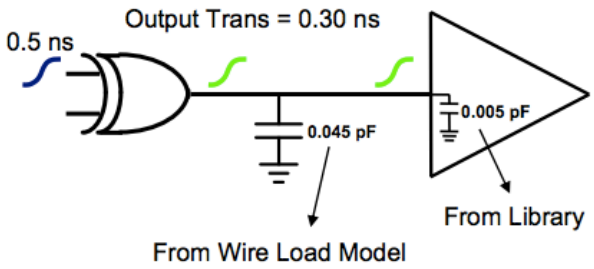
SPICE	Input Trans (ns) \ Output Load (pF)	.005	.05	.10	.15
	0.0	.1	.15	.2	.25
	0.5	.15	.23	.3	.38
	1.0	.25	.4	.55	.75

Cell Delay (ns)

Liberty Timing Model (NLDM)

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

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

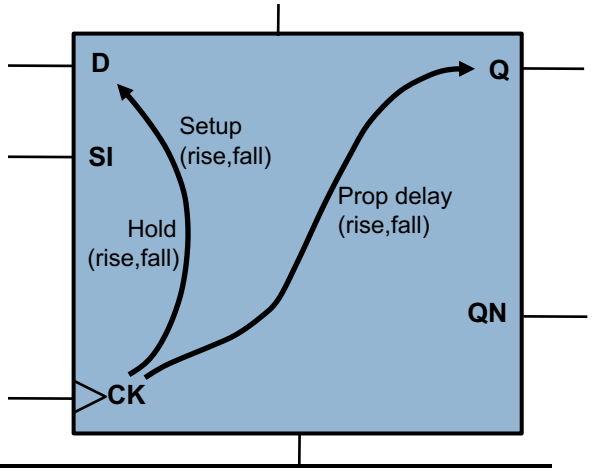


SPICE	Output Load (pF)			
	.005	.05	.10	.15
Input Trans (ns) 0.00	0.10	0.20	0.37	0.60
0.50	0.18	0.30	0.49	0.80
1.00	0.25	0.40	0.62	1.00

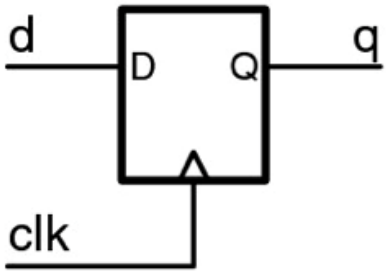
Output Transition (ns)

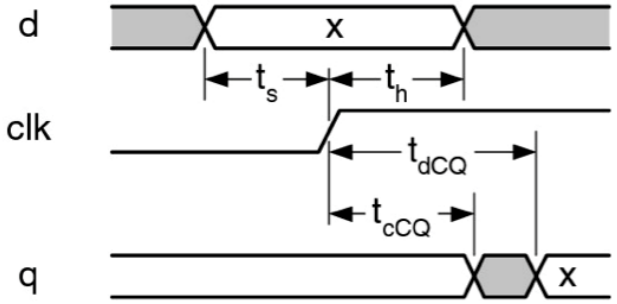
Timing Models for Sequential Cells

- Consider the timing arcs of a sequential cell shown below
- For synchronous inputs (D), there are the following timing arcs:
  - Setup check arc (rising and falling)
  - Hold check arc (rising and falling)
- For synchronous outputs of flip-flop, such as pin Q or QN, there is the following timing arc:
  - CK-to-output propagation delay arcs (rising falling)



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

Synchronous (Setup and Hold) Checks

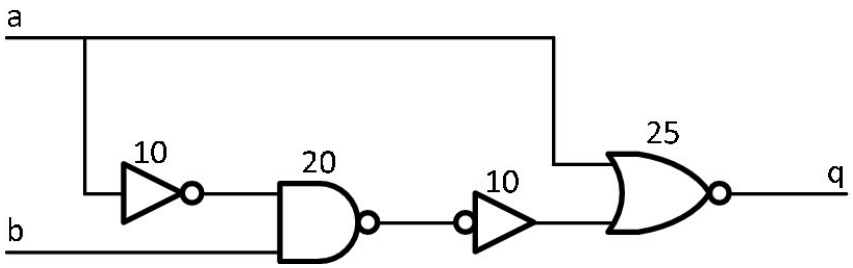
- The setup and hold constraints for a synchronous pin of a sequential cell are normally described in terms of two-dimensional tables as illustrated below.

```
pin (D)  {
direction  :  input;
. . .
timing  () {
related_pin  :  “CK”;
timing_type  :  “setup_rising”:
rise_constraint  (setup_template_3x3)  {
index_1     (“0.4,   0.57,   0.84”);/*Data transition*/
index_2     (“0.4,   0.57,   0.84”);/*Clock transition*/
values (  /*   0.4,   0.57,   0.84   */  \
   /*  0.4  */   “0.063,   0.093,   0.112”,  \
   /*  0.57 */  “0.526,  0.644,   0.824”,  \
   /*  0.84 */  “0.720,  0.839,   0.930”)   ;
}
```

```
fall_constraint  (setuphold_template_3x3)  {
index_1     (“0.4,   0.57,   0.84”);/*Data transition*/
timing () {
related_pin  :  “CK”;
timing_type :  “hold_rising”;
rise_constraint  (setuphold_template_3x3)  {
index_1     (“0.4,   0.57,   0.84”);/*Data transition*/
index_2     (“0.4,   0.57,   0.84”);/*Clock transition*/
values (  /*   0.4,   0.57,   0.84        */     \
   /*  0.4  */   “-0.220,  -0.339,   -0.584”,  \
   /*  0.57 */  “-0.247,  -0.381,   -0.729”,  \
   /*  0.84 */  “-0.398,  -0.516,   -0.864”)   ;
}
```

Path Delay

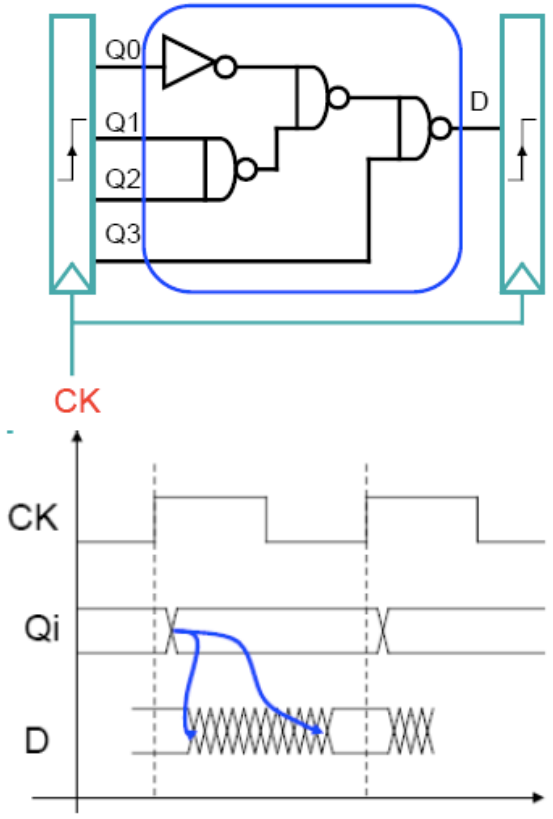
- What is the propagation delay?



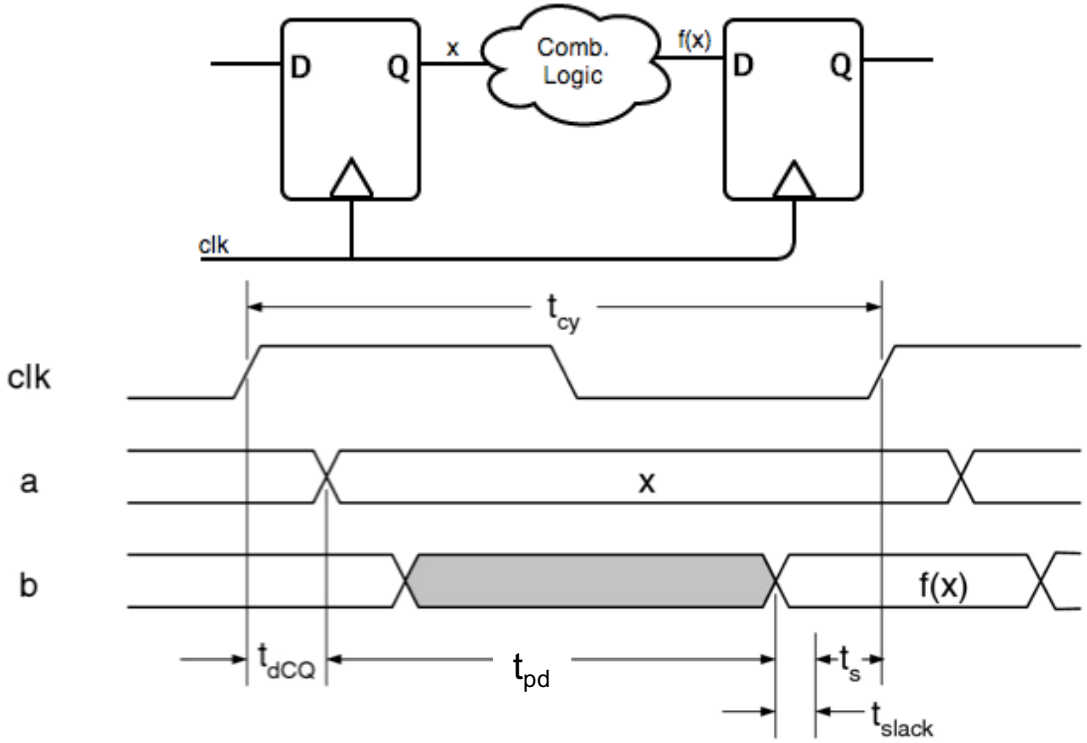
  - When all inputs change, the output starts to change or *glitch* after the minimum delay of 25ns, and settles to the final value after the propagation delay (65ns).
    - t_{cd}: Contamination delay (important for hold constraint)
    - t_{pd}: Propagation delay (important for setup constraint)

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)

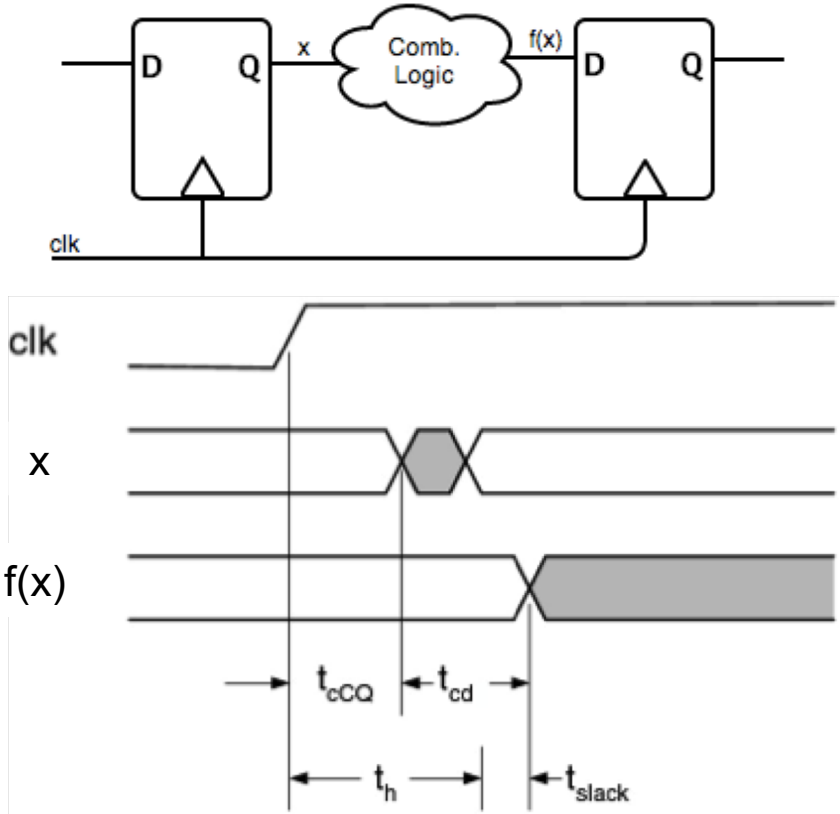


Setup and Hold Constraints



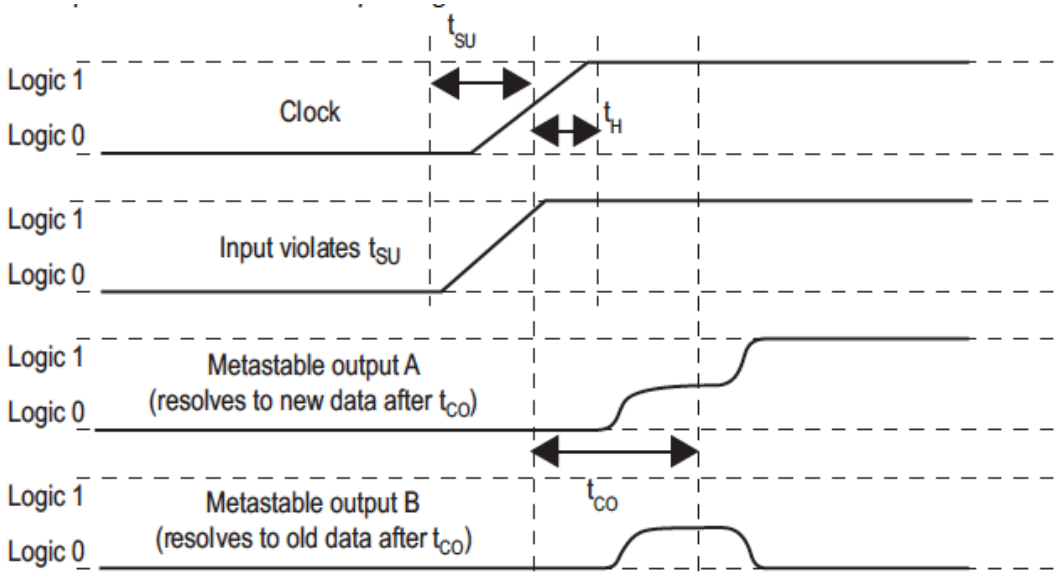
- $t_{cycle} >= t_{dCQ} + t_{pd} + t_s$
- $t_{slack} = t_{cycle} - (t_{dCQ} + t_{pd} + t_s)$

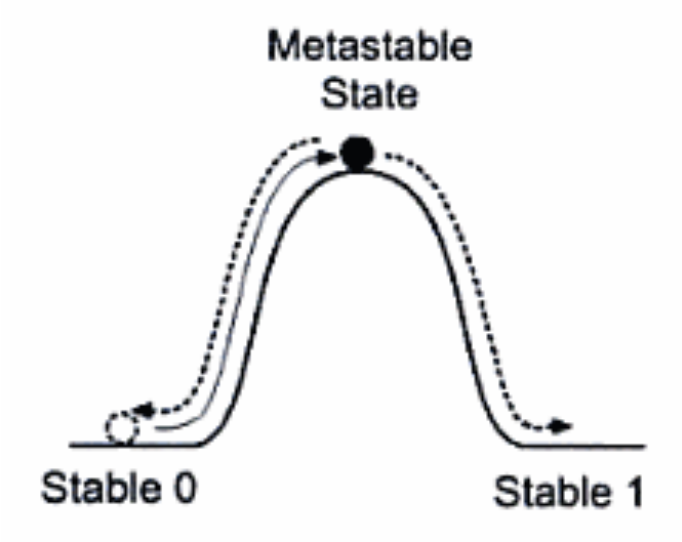
Setup and Hold Constraints



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

Violating Setup & Hold Time Constraints





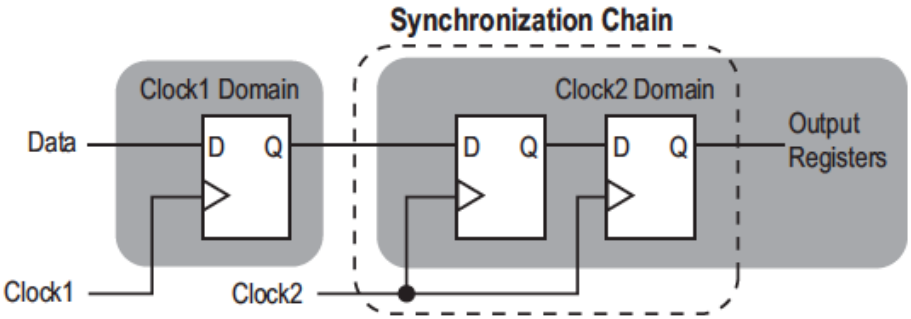
- Violation implies metastability

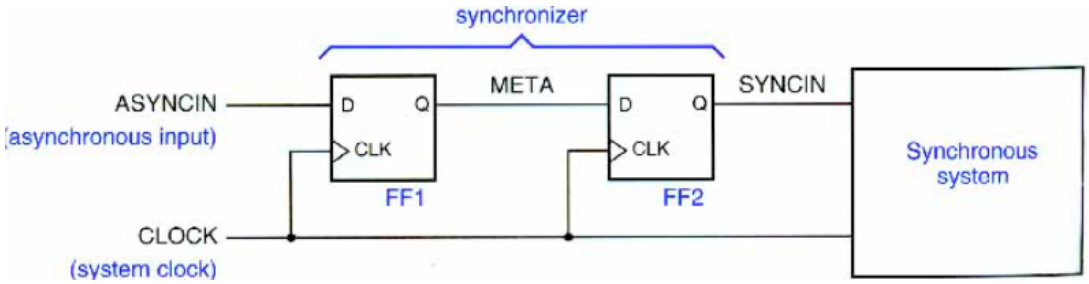
Asynchronous Signals & Metastability

- 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.

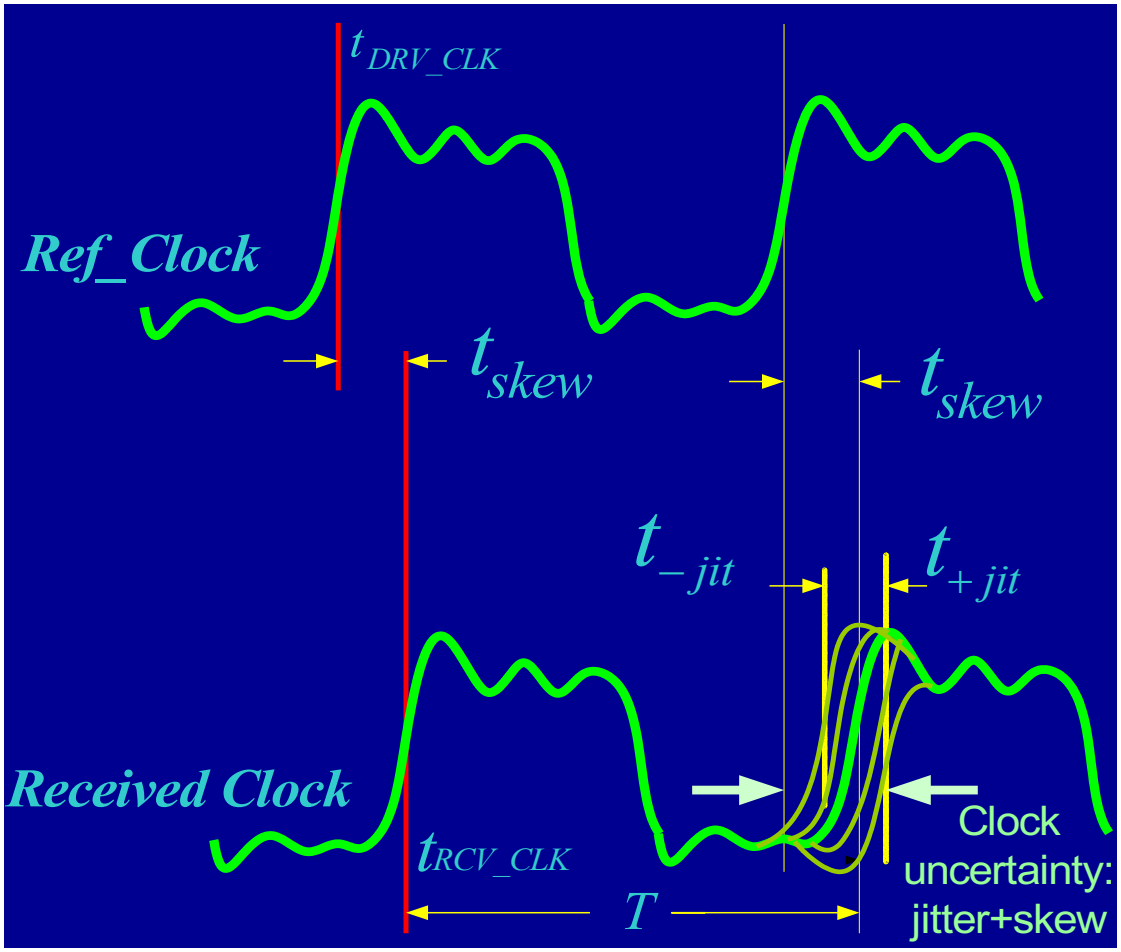
Asynchronous Signals & Metastability

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





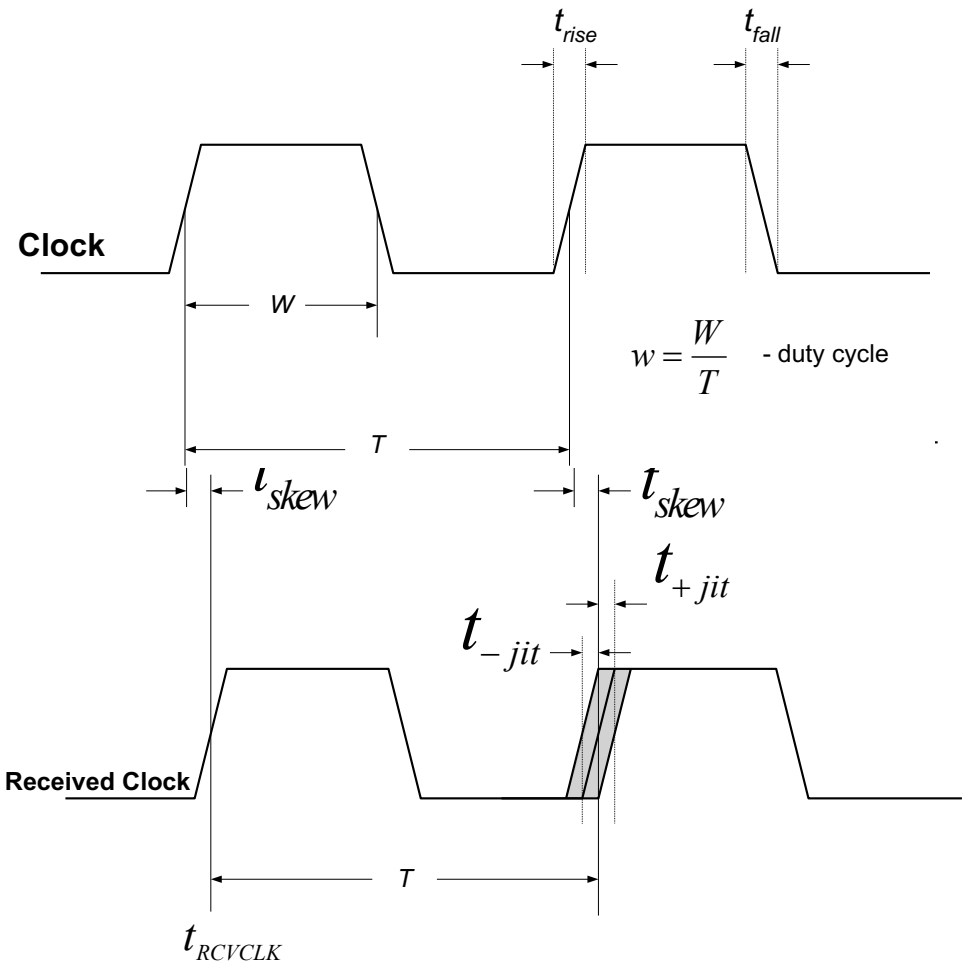
Clock Signal non-ideal Behavior



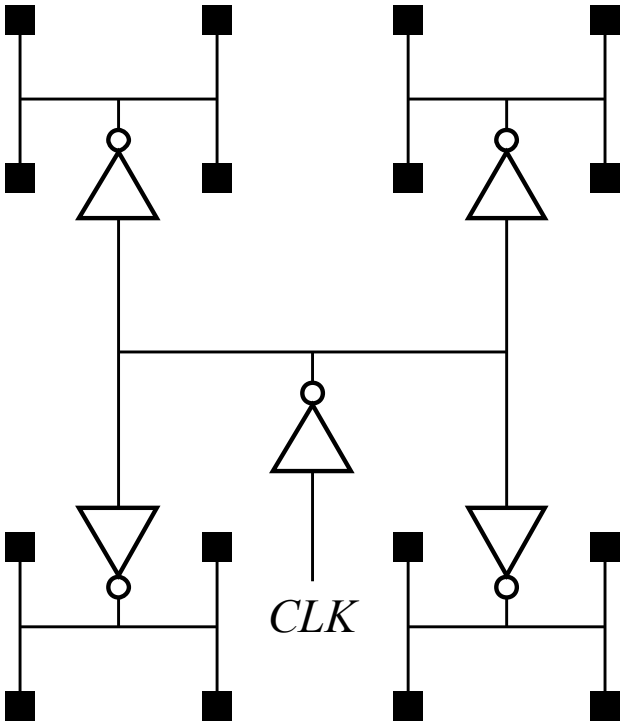
Clock Signal non-ideal Behavior

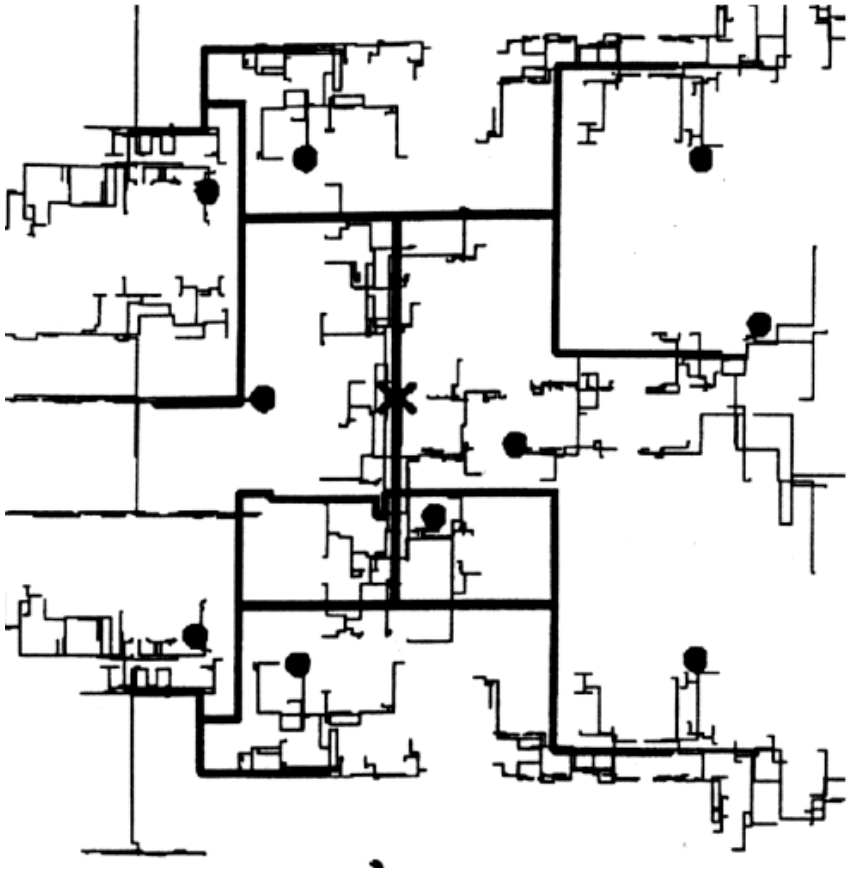
- Skew
  - Differences in clock signal arrival times across the chip
  - Cause: Interconnect delay (unbalanced Clock Tree)
- Jitter
  - Temporal variations in clock edges at a given point on the chip.
    - Results in continuous clock period variation.
    - In general, each clock period has a different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

Clock Signal non-ideal Behavior



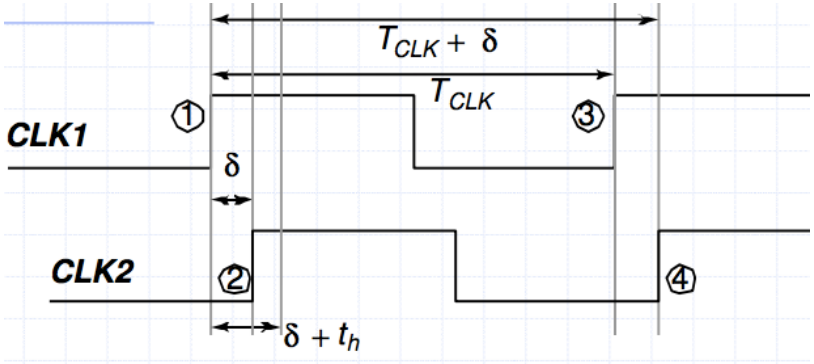
Clock Tree and Skew



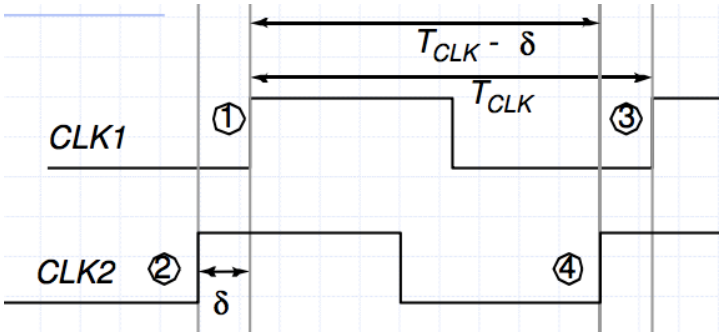


Positive Skew & Negative Skew

- +ve Skew



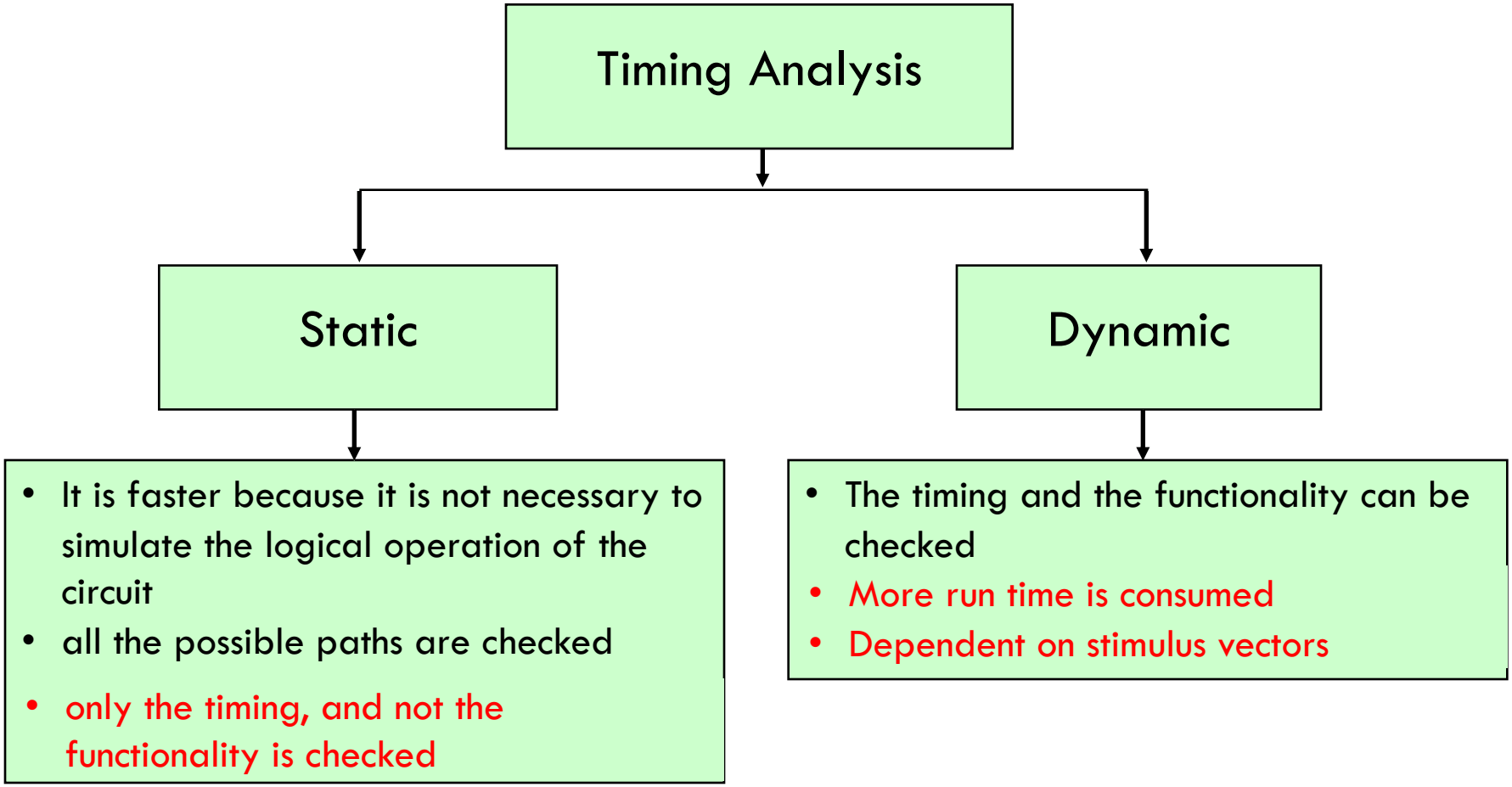
- -ve Skew



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

Timing Verification



Static Timing Analysis

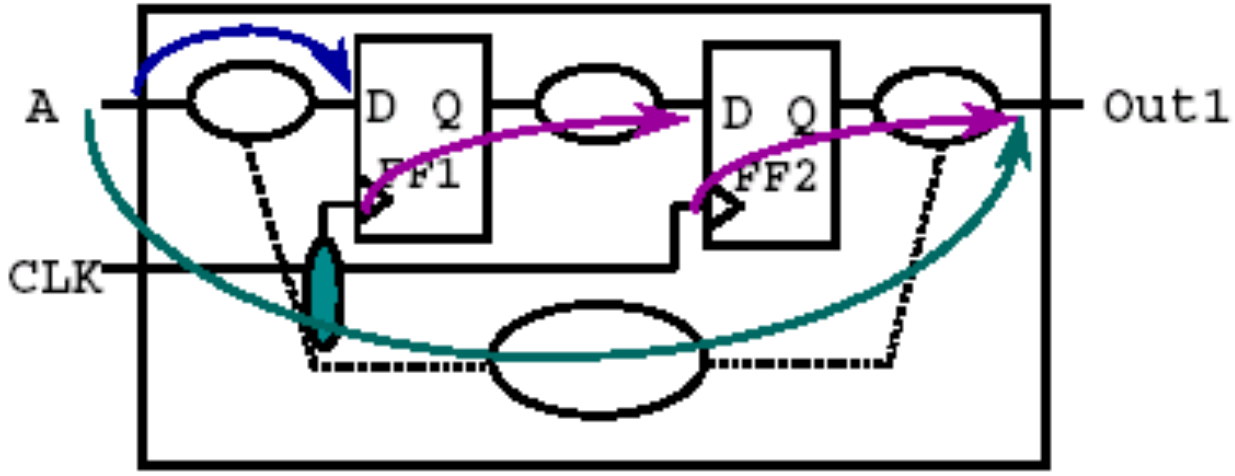
- Problem: Given a circuit, find the path(s) with the largest delay (critical paths)
  - Solution: run SPICE and report the results of the simulation
    - Problem: SPICE is computationally expensive to run except for small-size circuits plus it needs test vector
- WANTED: We need a fast method that 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 (Dynamic Analysis)
  - Proper circuit functionality is not checked
  - Vector generation NOT required

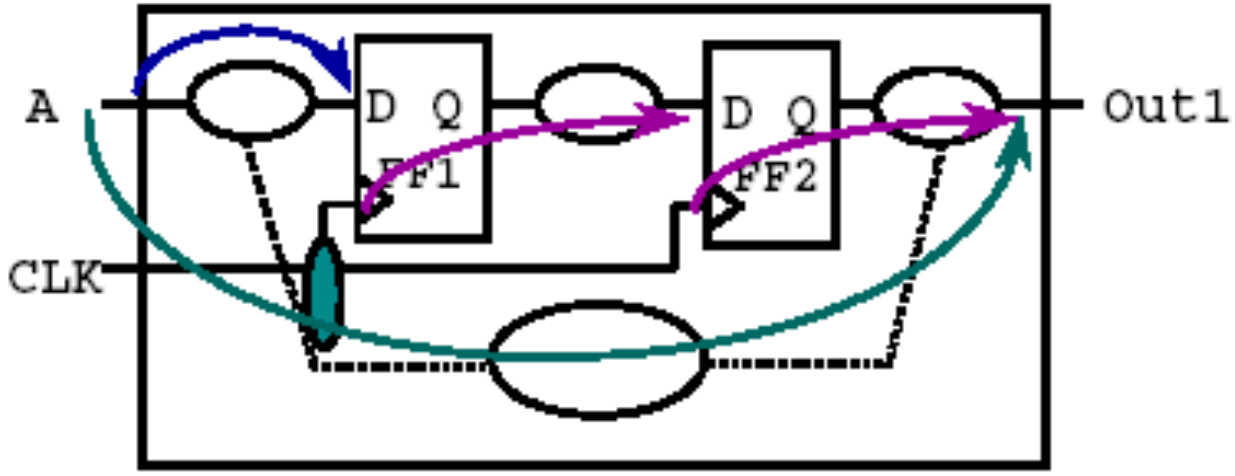
STA Steps

- 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

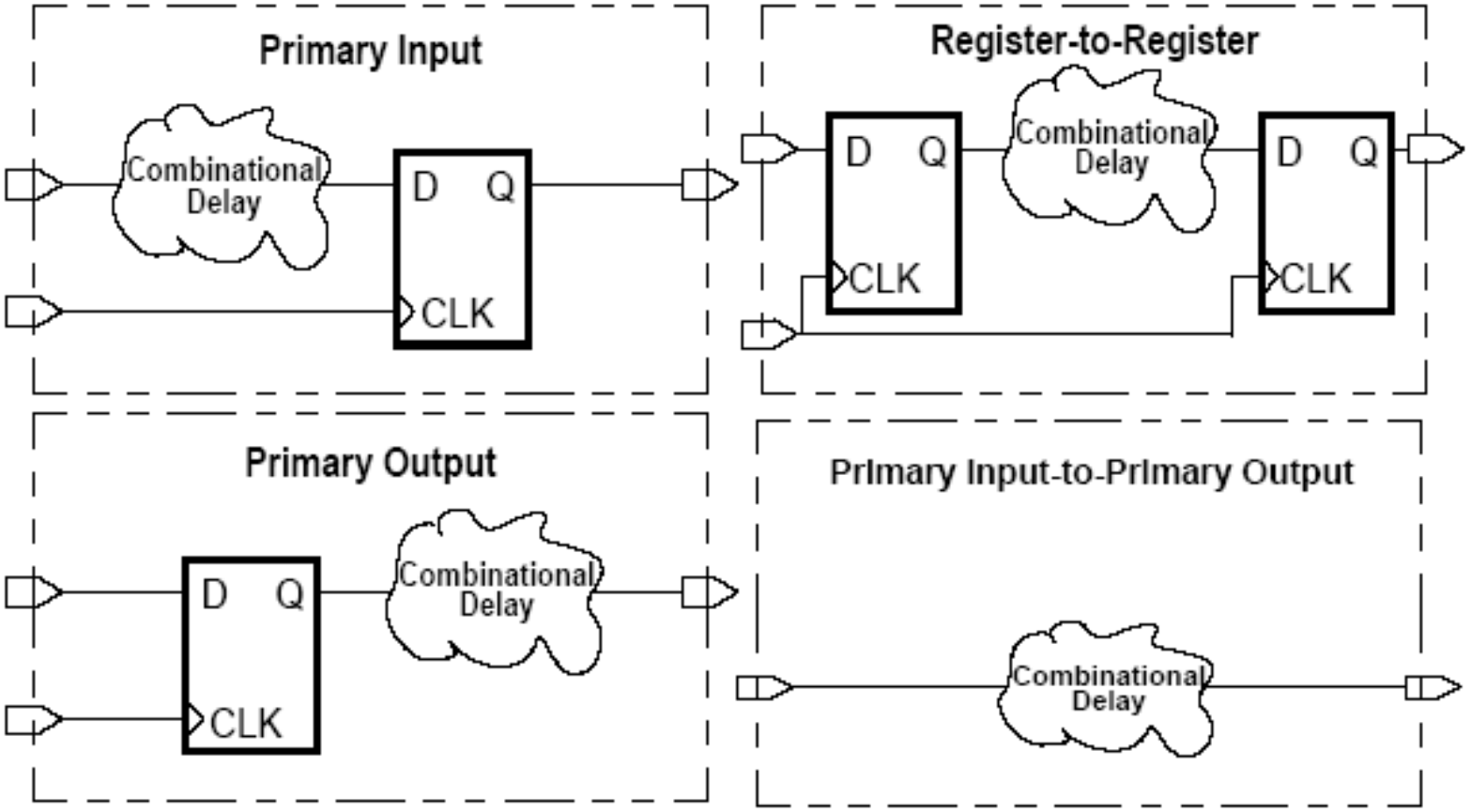
- A Timing Path is a point-to-point path in a design which can propagate data from one flip-flop to another
- Each path has a start point and an endpoint
  - Start point: Input ports Clock pins of flip-flops
  - Endpoints: Output ports Data input pins of flip-flops



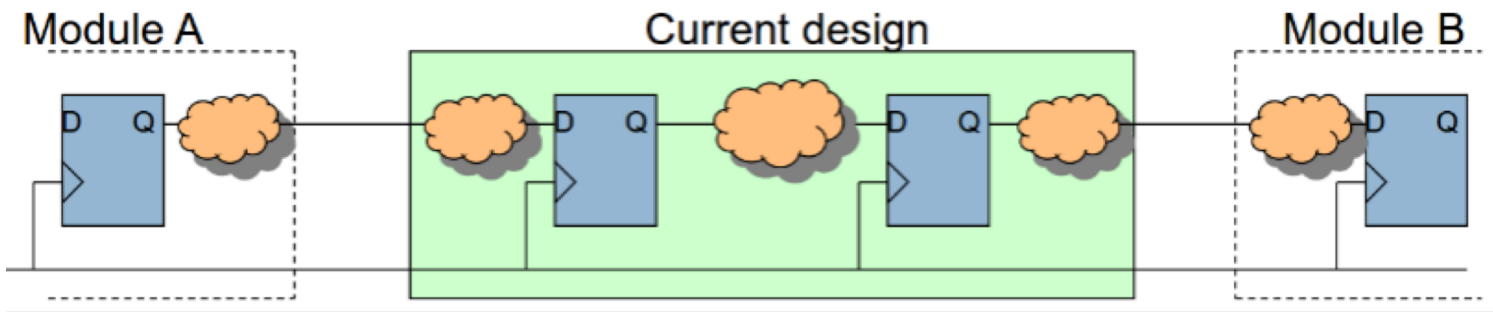
Types of Timing Paths

- 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



Types of Timing Paths

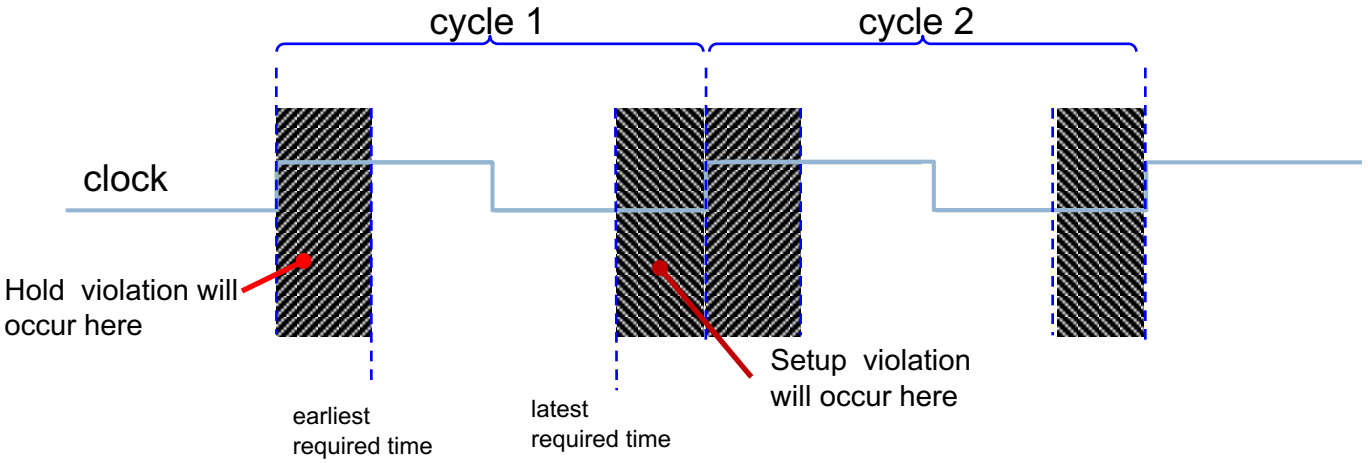


STA Terminologies

- Critical Path: Theoretically path which has maximum delay
- Arrival Time: Time taken by data to reach an 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

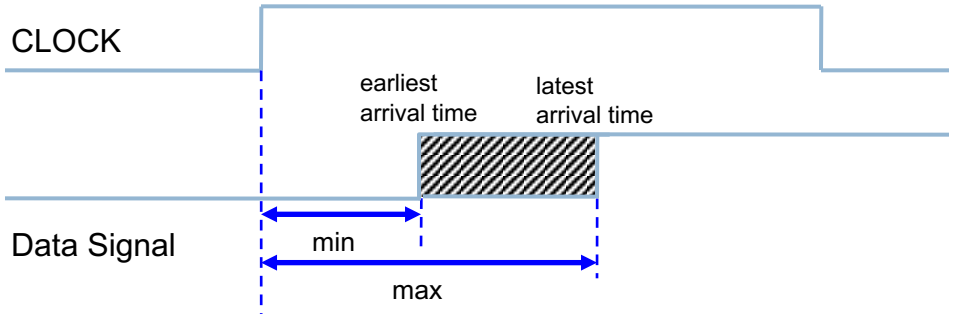
Required Time

- Required time specifies the time point (interval) at which data is required to arrive at end point (data is required to be stable after arrival).
  - Time point after which data can become unstable (change) is called earliest required time
  - Time point after which data cannot become unstable (change) is called latest required time
- The requirement is set by timing constraints like setup/hold, etc.



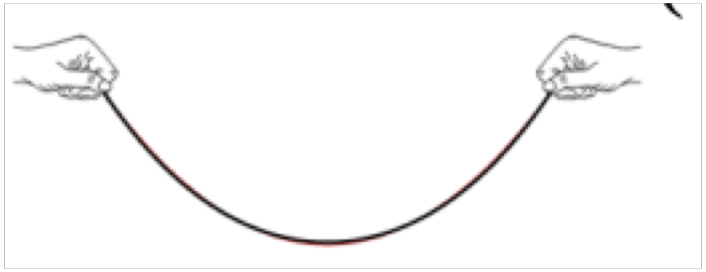
Arrival Time

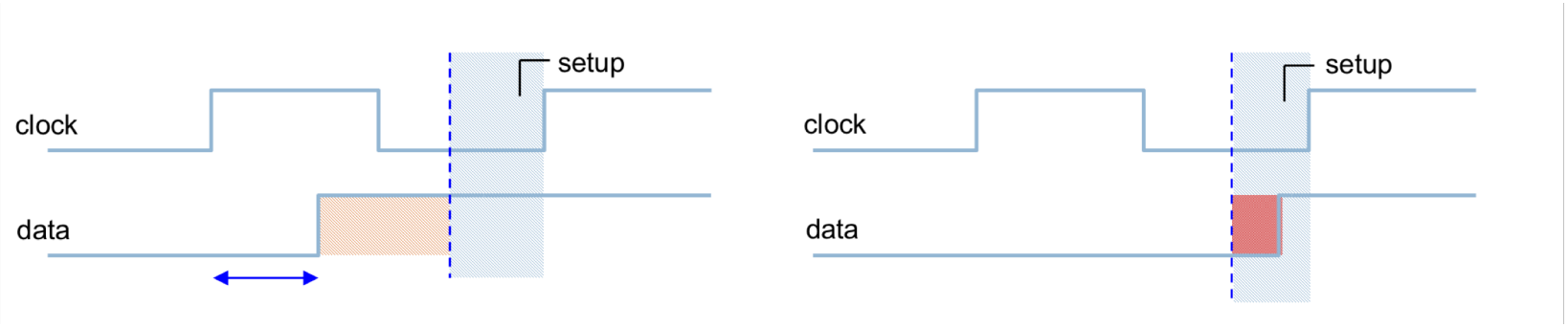
- Arrival time defines the time interval during which a data signal will arrive at a path endpoint (after arrival time signal will be stable).
- Data is normally triggered by clock edge
- Data arrival depend on circuit delay, which vary (depend on temperature, supply voltage, etc.)
- Minimum delay, early arrival
- Maximum delay, late arrival



Slack & Critical Path

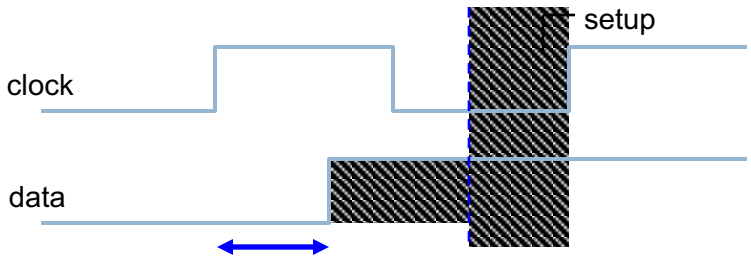
- The slack is the difference between the required and arrival times
  - Negative Slack ➔ Constraints Violation
  - Positive Slack ➔ Constraints are met
- The path with the smallest slack is called the critical path.

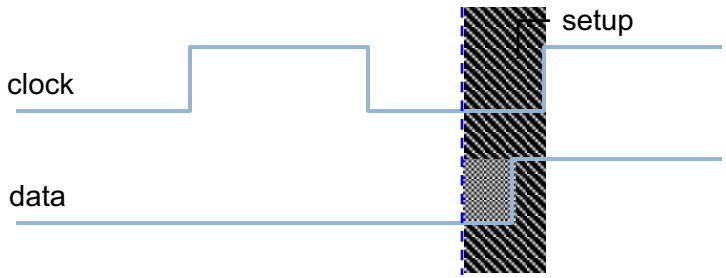




Early and Latest Analysis

- STA tool calculates the slack of each logic path, in order to find critical path.
- Early and Latest analysis approaches:
  - Assumes circuits have minimum delay, compares arrival time to earliest required time (hold check)
  - Assumes circuits have maximum delay, compares arrival time to latest required time (setup check)



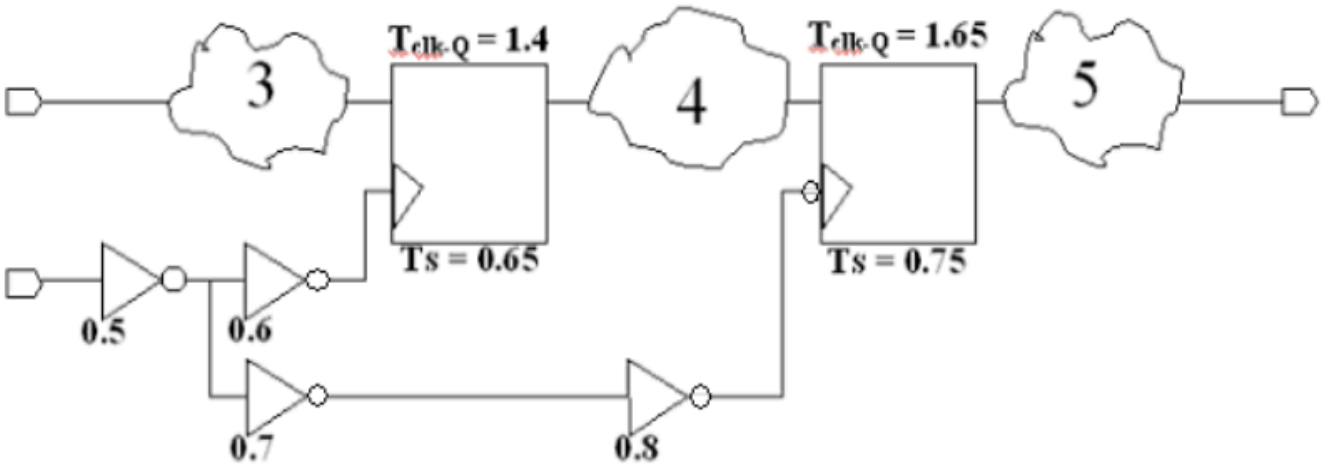


Solution Methodology

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

Problem 1

- Find the maximum frequency?



  - Assume no input delays or output delays are specified

Solution (1)

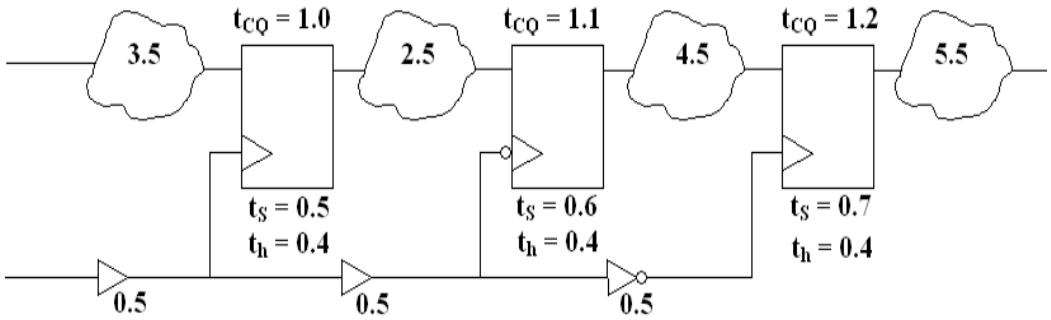
- Input – Reg Path
  - Arrival Time = 3 (assuming no input delay constraint)
  - Required Time = CP + 0.5 + 0.6 $-t_{setup}$ = CP + 1.1 –0.65
  - Equating the TWO, CP = 3 – 1.1 + 0.65 = 2.55
- Reg – Reg Path
  - Arrival Time = 0.5 + 0.6 + 1.4 + 4 = 6.5
  - Required Time = 0.5 + 0.7 + 0.8 + CP – 0.75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

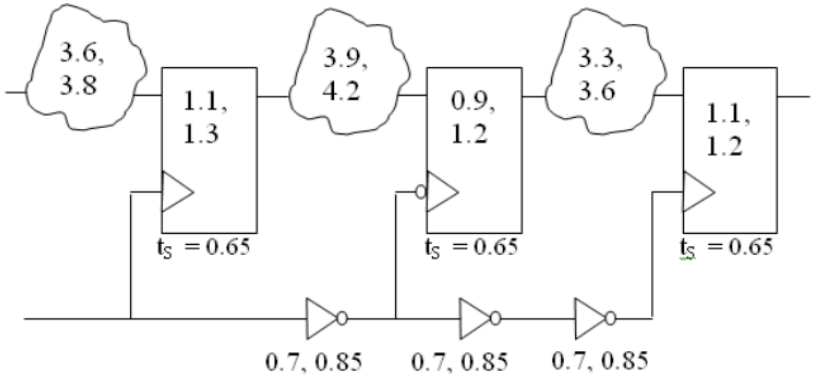
CP: Clock Period

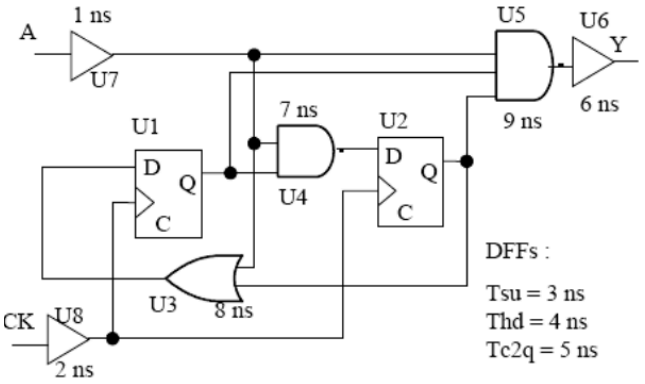
Solution (2)

- 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

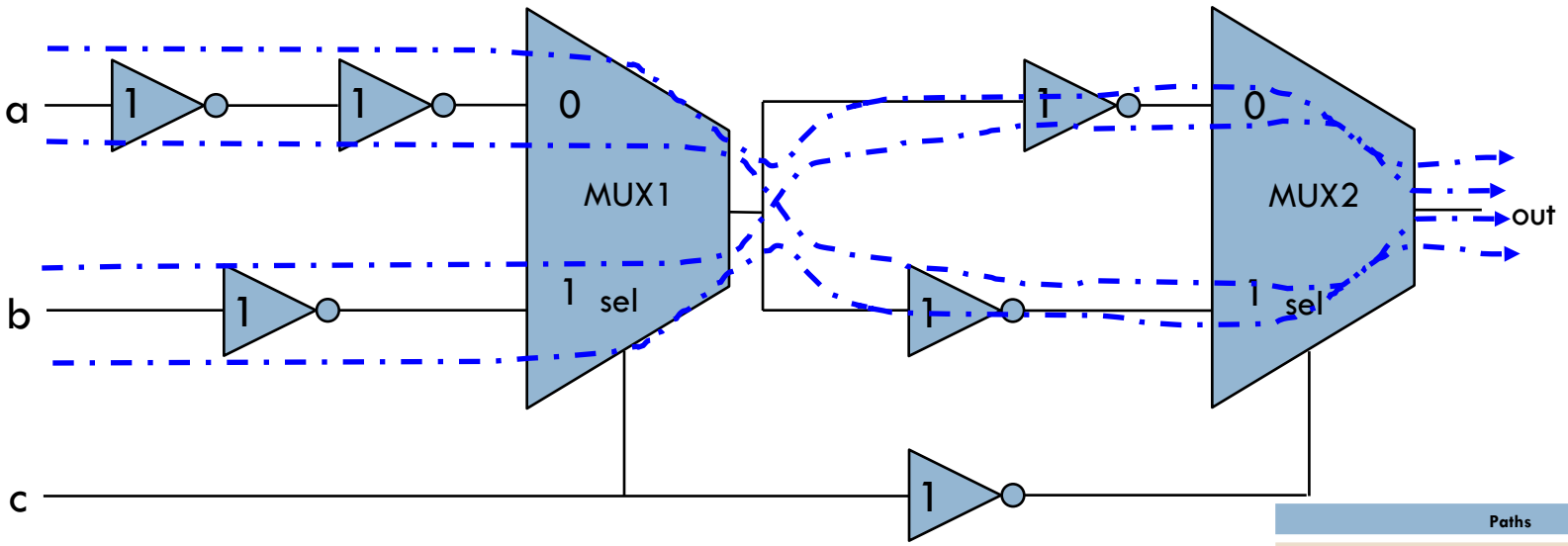
More Problems







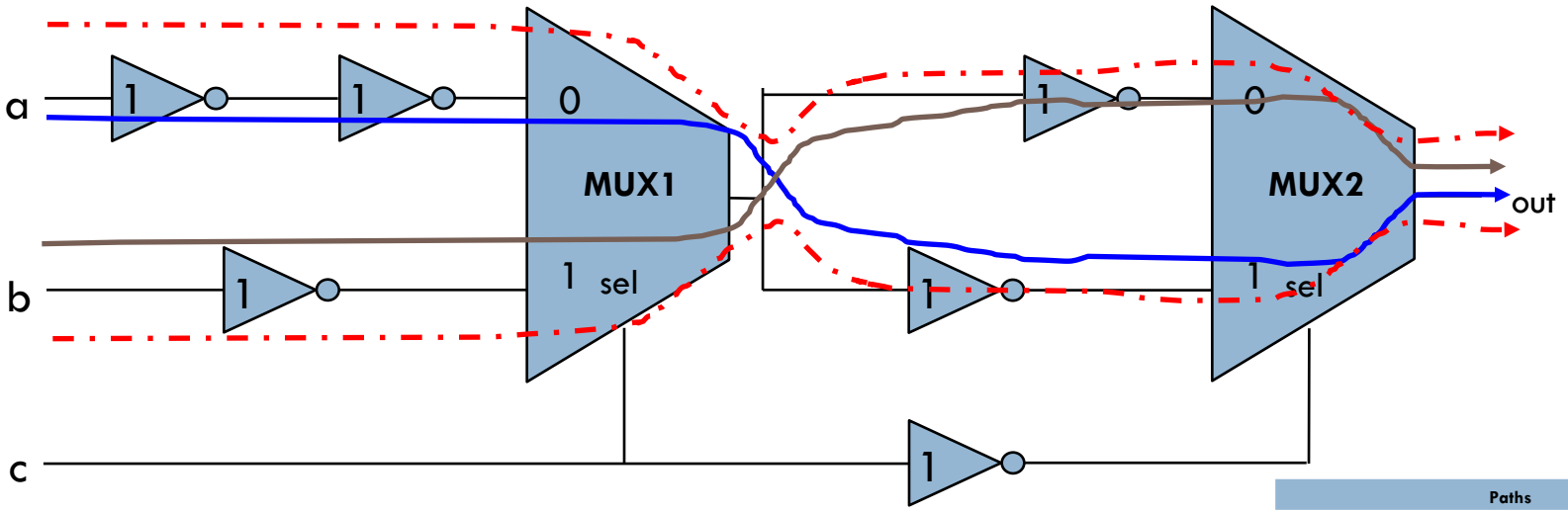
Possible Paths



Paths
a → MUX1.0 → MUX2.0 → out
a → MUX1.0 → MUX2.1 → out
b → MUX1.1 → MUX2.0 → out
b → MUX1.1 → MUX2.1 → out

False Paths

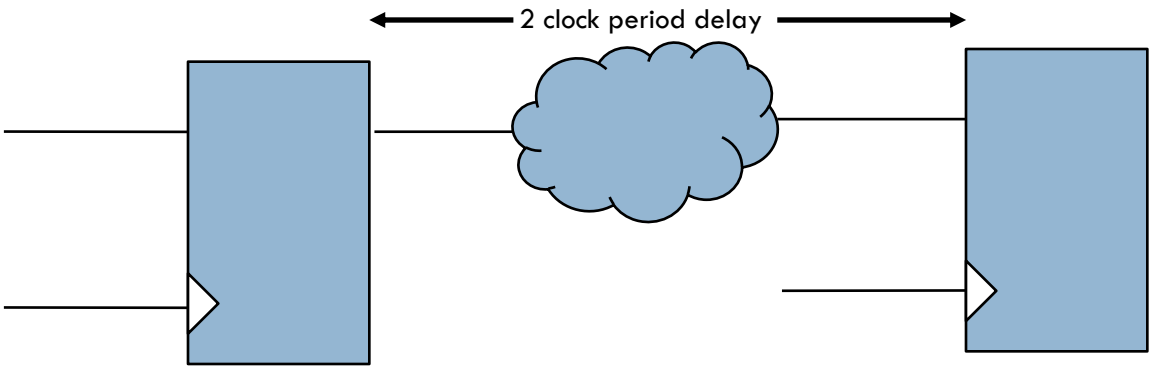
- Paths which physically exist in a design but are not logic paths. These paths never get sensitized under any input condition



Paths
~~a → MUX1.0 → MUX2.0 → out~~
a → MUX1.0 → MUX2.1 → out
b → MUX1.1 → MUX2.0 → out
~~b → MUX1.1 → MUX2.1 → out~~

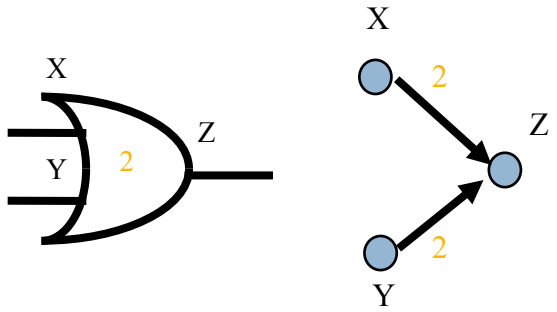
Multi-cycle Paths

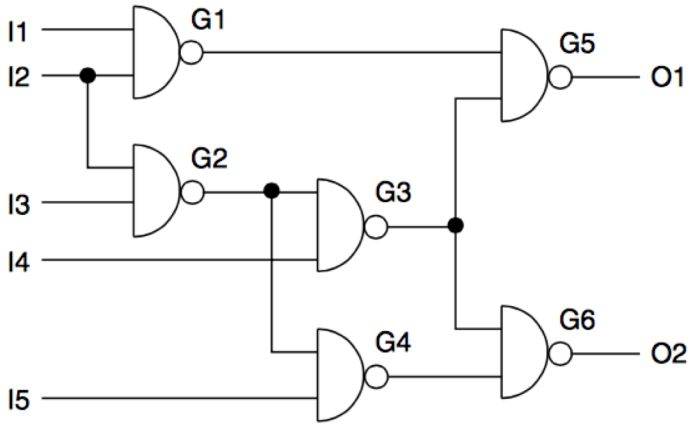
- There are data paths that require more than one clock period for execution.

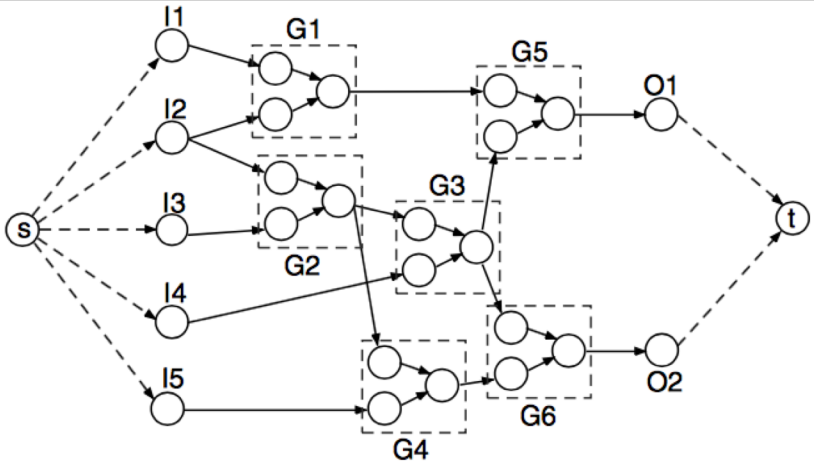


How is STA Performed in S/W?

- 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

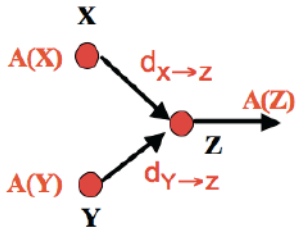




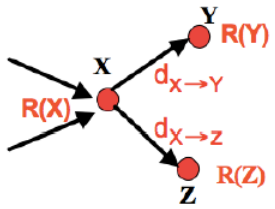


How is STA Performed?

- 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

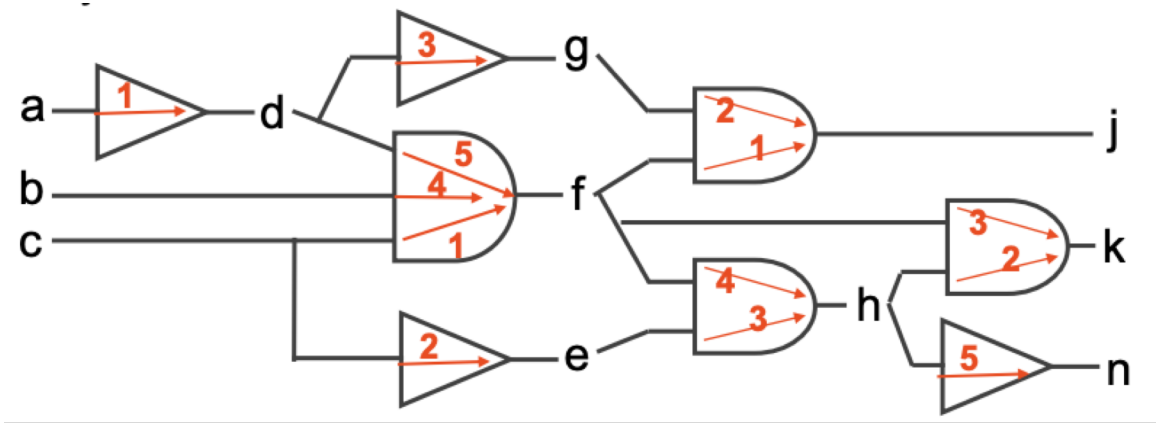


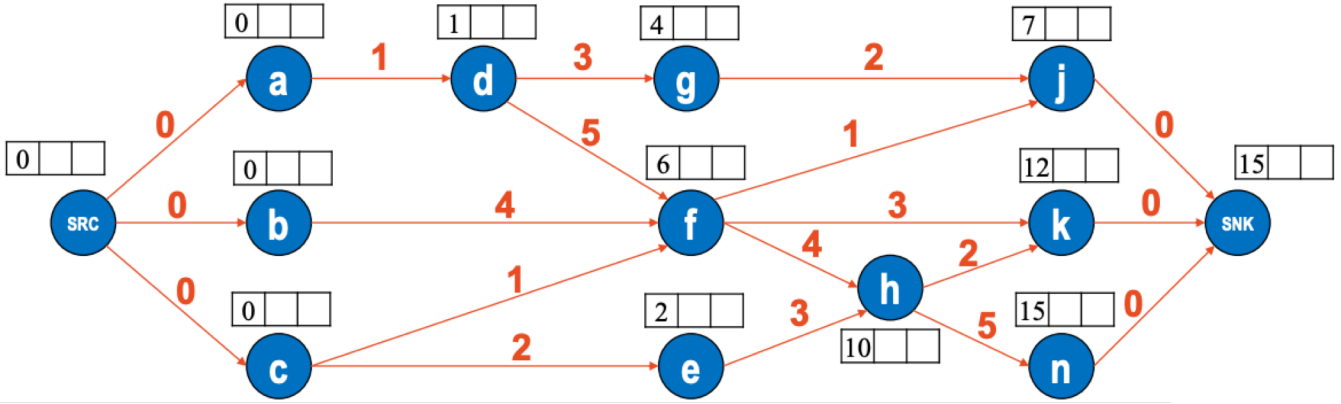
$$A(v) = \max_{u \in FI(v)} (A(u) + d_{u \to v})$$



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

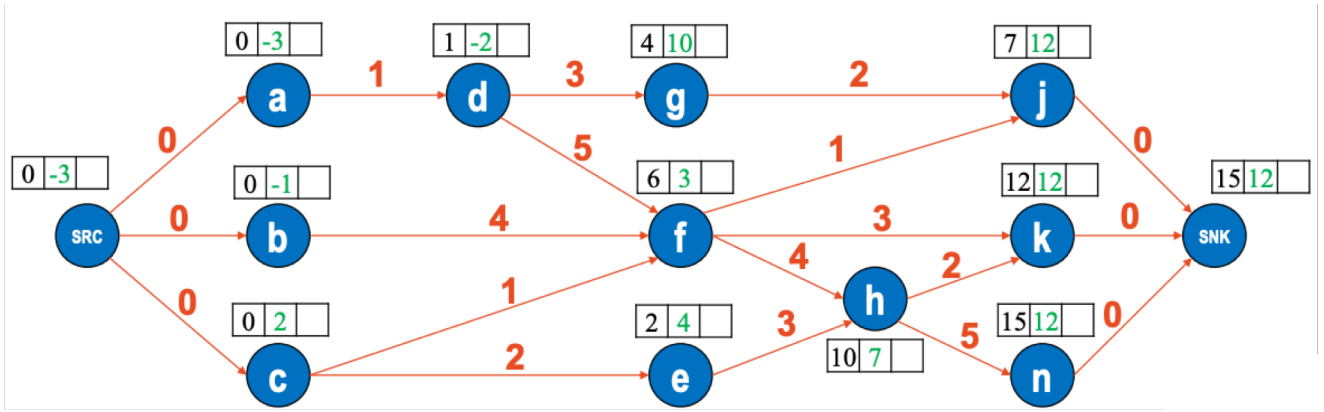
How STA is Performed?



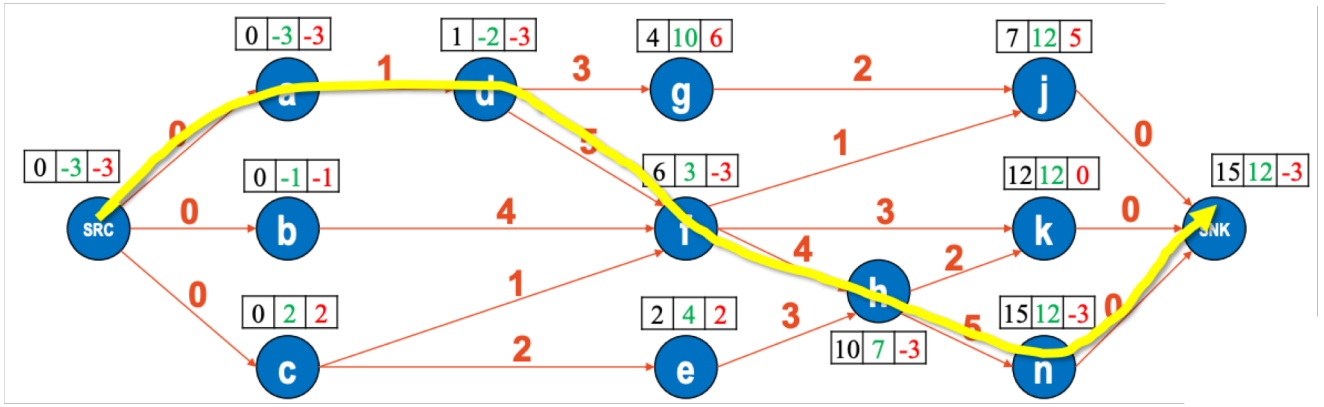


Ignoring wire delays

How STA is Performed?



RAT @ Sink is 12

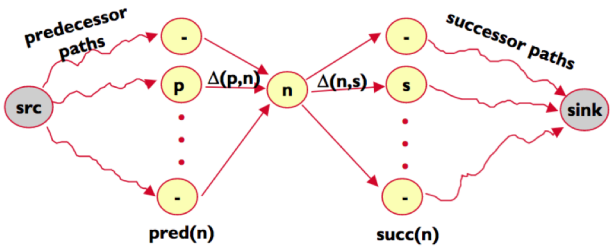


Critical path nodes have the same slack

Issues?

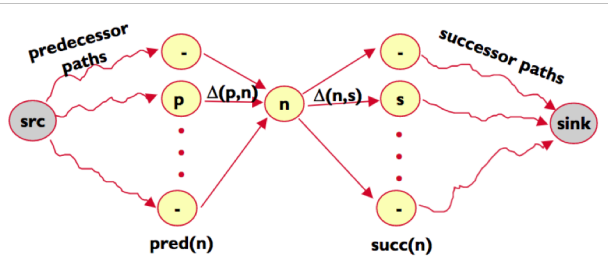
- How about Hold Constraints?
  - What we calculated was the late AAT and RAT which is good for setup constraints check
  - For Hold constraints check, we use the Early AAT and Early RAT
- Ordering?
  - Topological order for AAT
  - Reverse Topological order for RAT

Late and Early AAT and RAT



$$AT_E(n) = \text{min delay to } n = \begin{cases} 0 \text{ if } n == src \\ \underset{p = pred(n)}{Min} \{AT_E(p) + \delta(p,n)\} \end{cases}$$

$$AT_L(n) = \text{max delay to } n = \begin{cases} 0 \text{ if } n == src \\ \underset{p = pred(n)}{Max} \{AT_L(p) + \Delta(p,n)\} \end{cases}$$



$$RAT_E(n) = \begin{cases} 0 \text{ if } n == sink \\ \underset{s = succ(n)}{Max} \{RAT_E(s) - \delta(n,s)\} \end{cases}$$

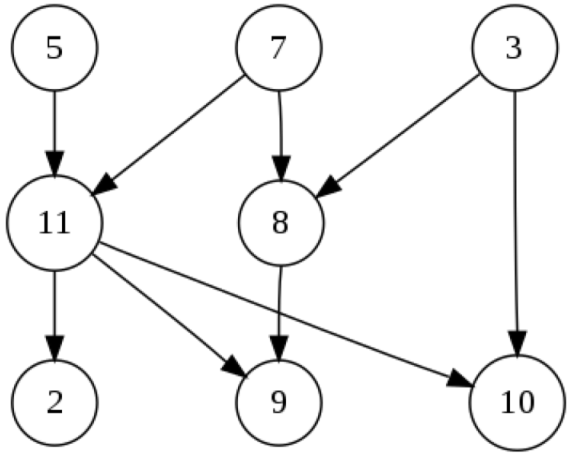
$$RAT_L(n) = \begin{cases} \text{Cycle time if } n == sink \\ \underset{s = succ(n)}{Min} \{RAT_L(s) - \Delta(n,s)\} \end{cases}$$

Note reversal of min and max for early and late modes; this is because we're subtracting delays instead of adding them

Source: R. Rutenbar

Topological Sort

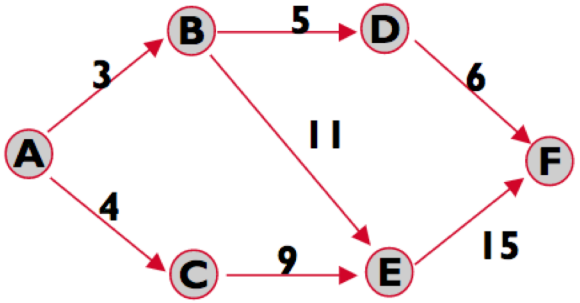
- A topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge *uv* from vertex *u* to vertex *v*, *u* comes before *v* in the ordering.

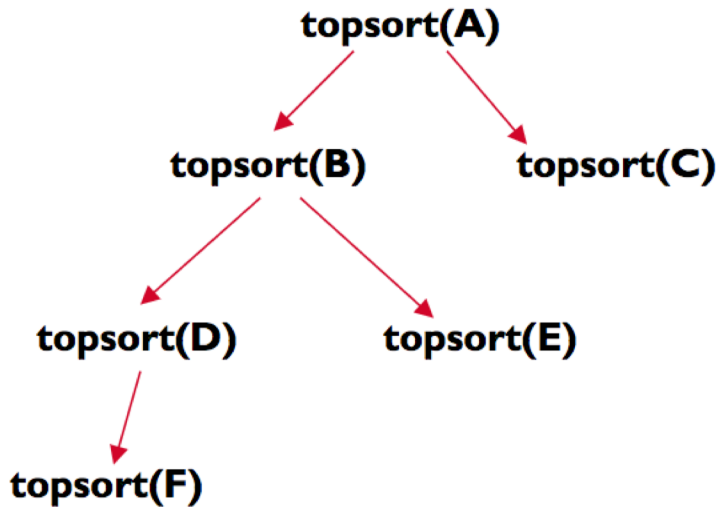


5, 7, 3, 11, 8, 2, 9, 10
3, 5, 7, 8, 11, 2, 9, 10
5, 7, 3, 8, 11, 10, 9, 2
•
•
•

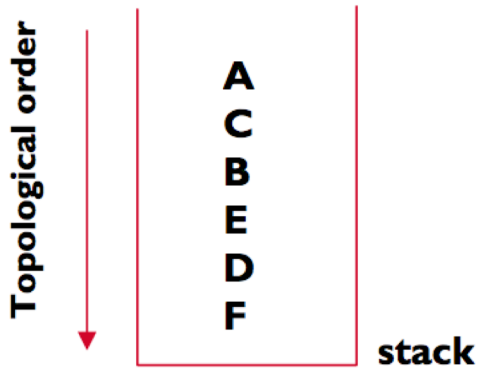
Src: Wikipedia

Topological Sort (DFS)



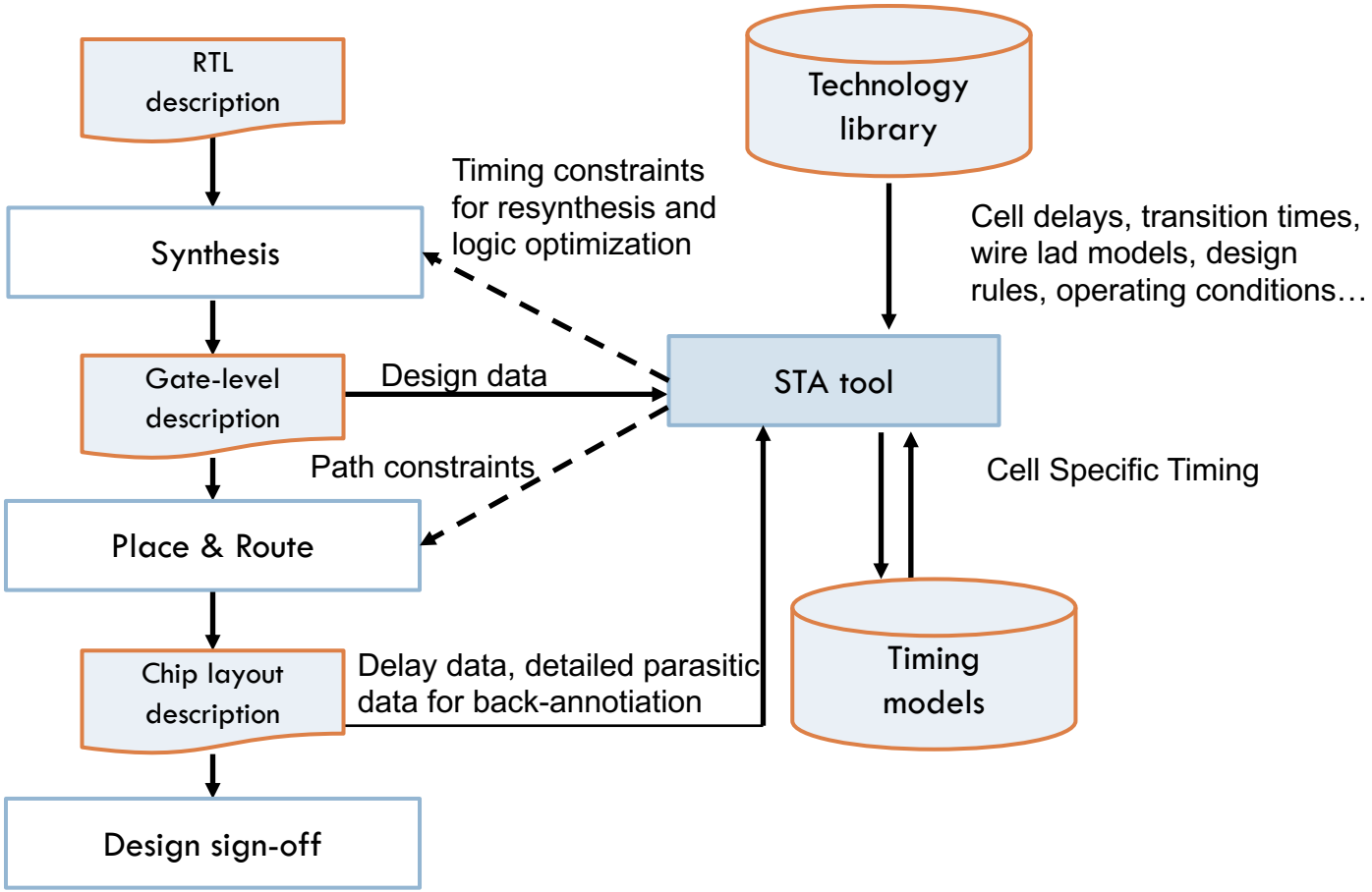



```
topsort( node n ) {
  for each s in succ(n) {
    if s has not been visited {
      topsort( s );
      push n on stack ;
      mark n as visited;
    }
  }
}

topsort(SRC);
```



Source: R. Rutenbar

STA Tools


RTL description
Synthesis
Gate-level description
Place & Route
Chip layout description
Design sign-off
Technology library
Timing constraints for resynthesis and logic optimization
Cell delays, transition times, wire lad models, design rules, operating conditions…
Design data
STA tool
Path constraints
Cell Specific Timing
Timing models
Delay data, detailed parasitic data for back-annotiation

Synopsys Prime Time (PT)

- Industry standard STA tool
- Sample Timing Report

report_timing

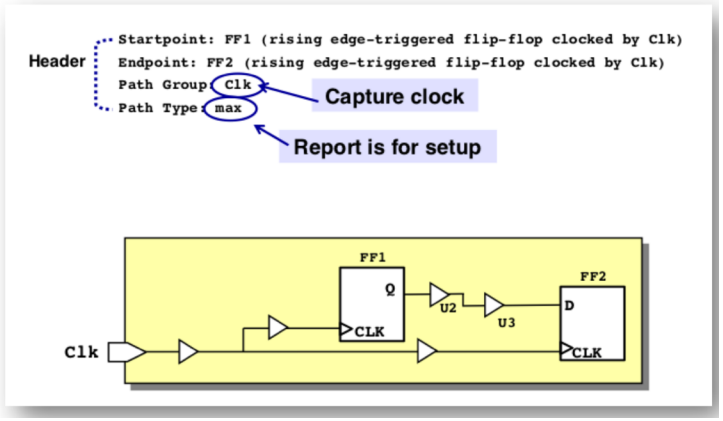
```
Startpoint: FF1 (rising edge-triggered flip-flop clocked by Clk)
Endpoint: FF2 (rising edge-triggered flip-flop clocked by Clk)
Path Group: Clk
Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock Clk (rise edge)                    0.00       0.00
  clock network delay (propagated)         1.10 *     1.10
  FF1/CLK (fdef1a15)                       0.00       1.10 r
  FF1/Q (fdef1a15)                         0.50 *     1.60 r
  U2/Y  (buf1a27)                          0.11 *     1.71 r
  U3/Y  (buf1a27)                          0.11 *     1.82 r
  FF2/D (fdef1a15)                         0.05 *     1.87 r
  data arrival time                                   1.87

  clock Clk (rise edge)                    4.00       4.00
  clock network delay (propagated)         1.00 *     5.00
  FF2/CLK (fdef1a15)                                  5.00 r
  library setup time                      -0.21 *     4.79
  data required time                                  4.79
  -----------------------------------------------------------
  data required time                                  4.79
  data arrival time                                  -1.87
  -----------------------------------------------------------
  slack (MET)                                         2.92
```

Header

Data arrival

Data required

Slack



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

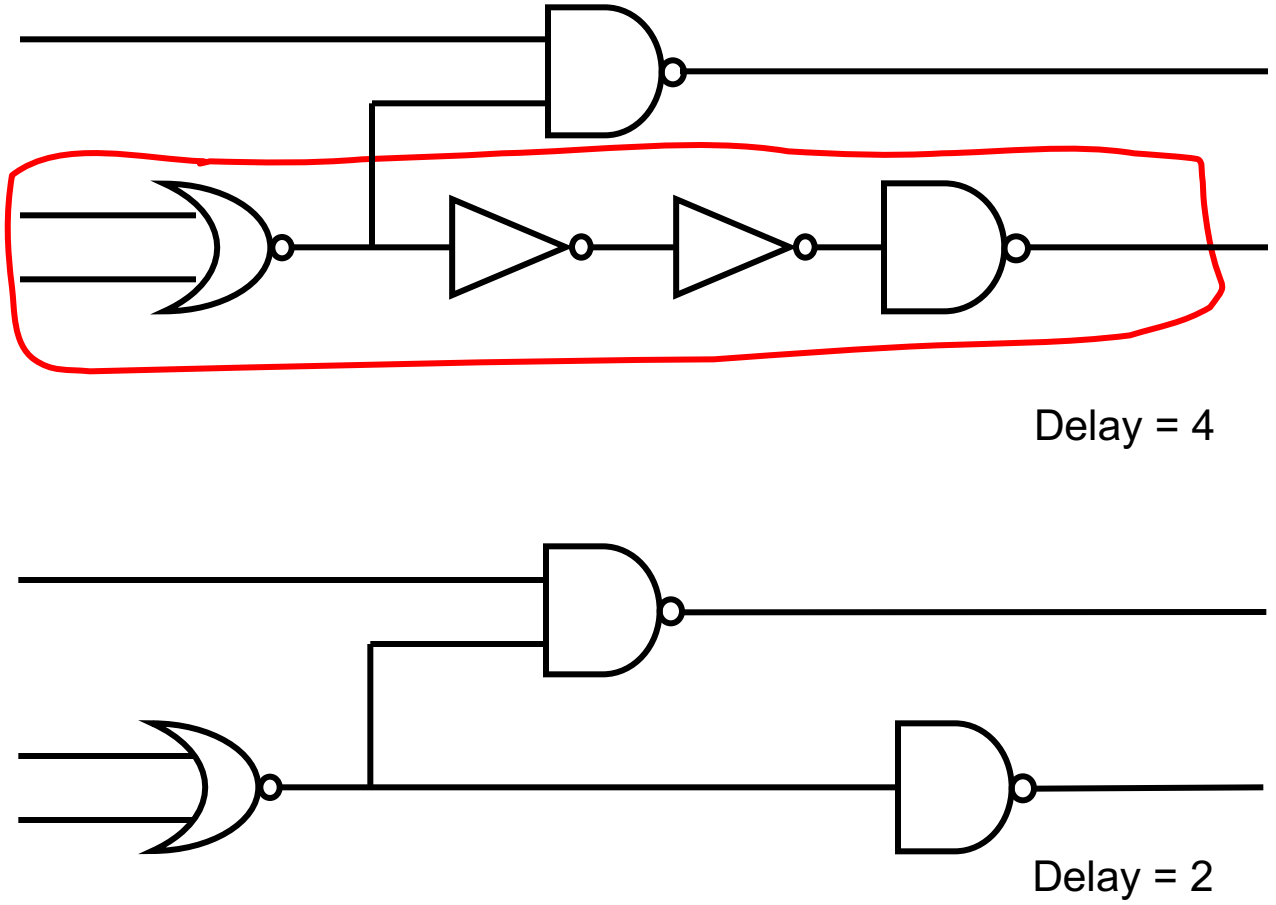
.....

Double Inverter

Removal

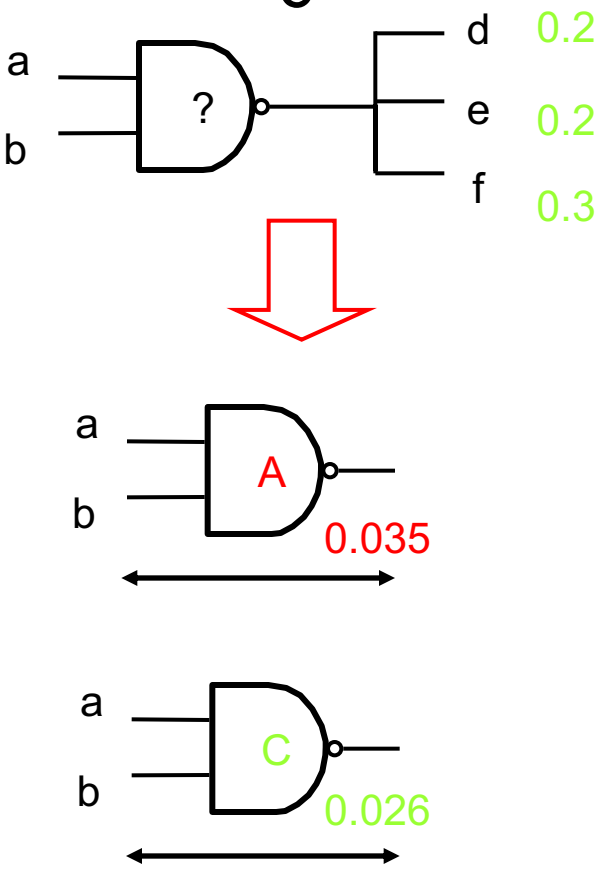
.....

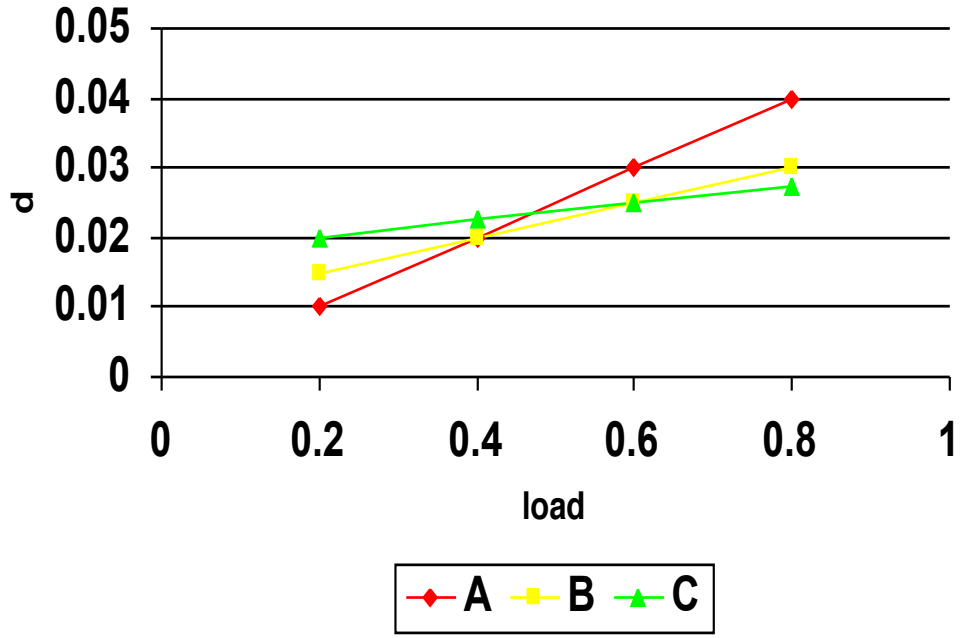
.....



Local Synthesis Transforms

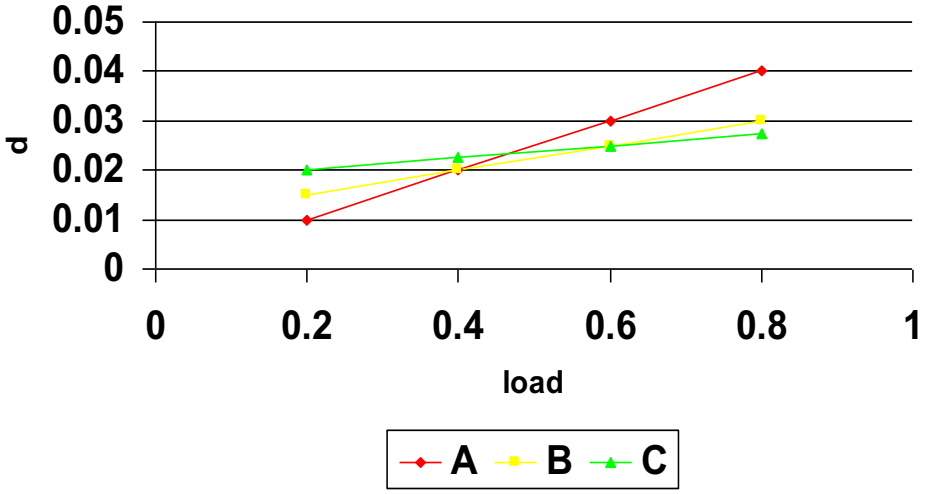
- Resizing





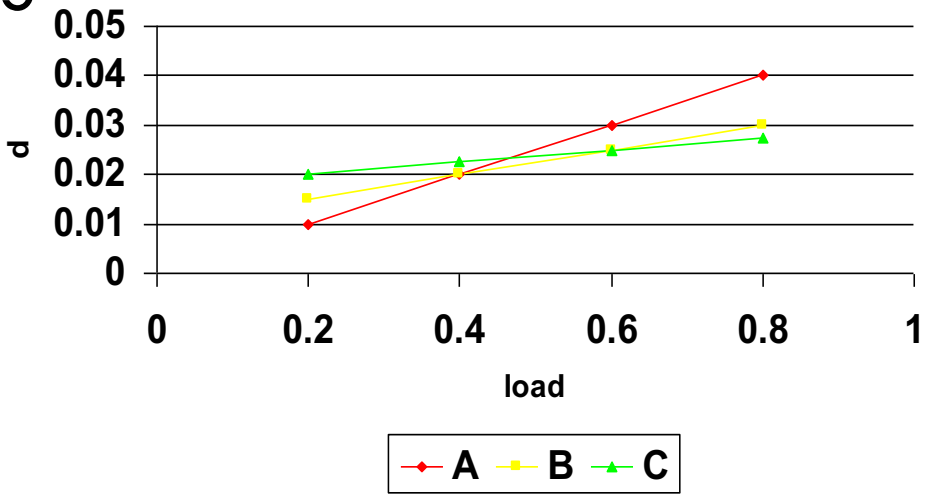
Local Synthesis Transforms

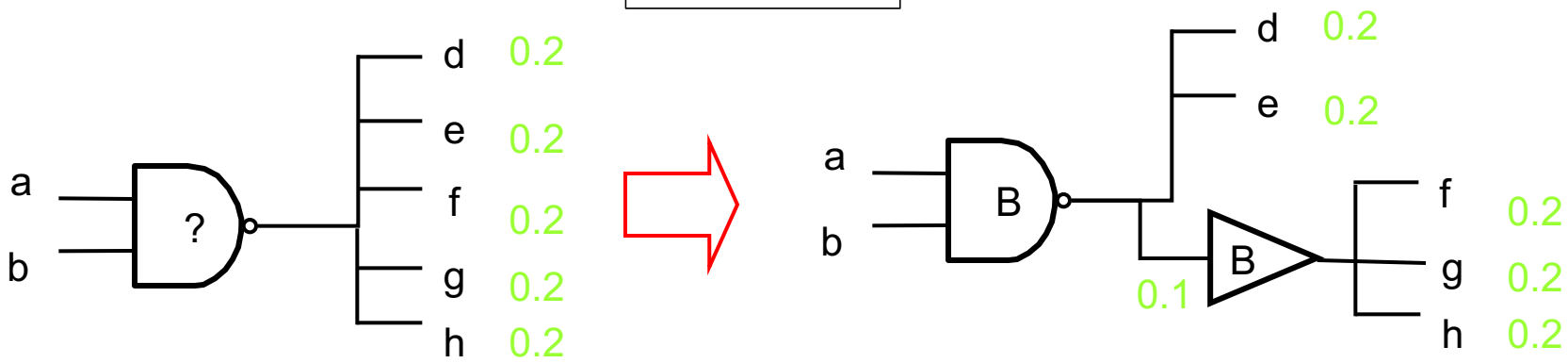
- Cloning



Local Synthesis Transforms

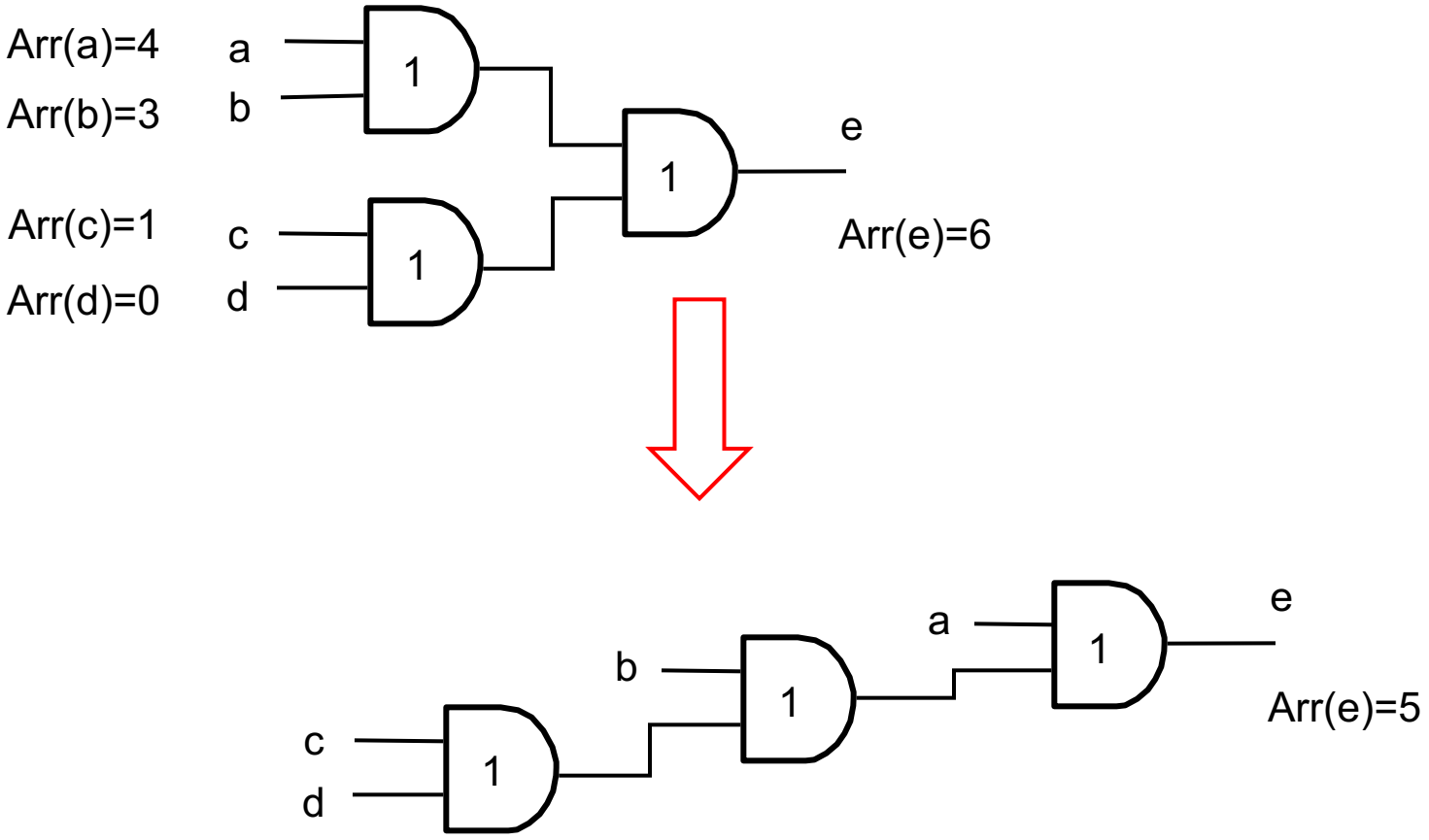
- Buffering





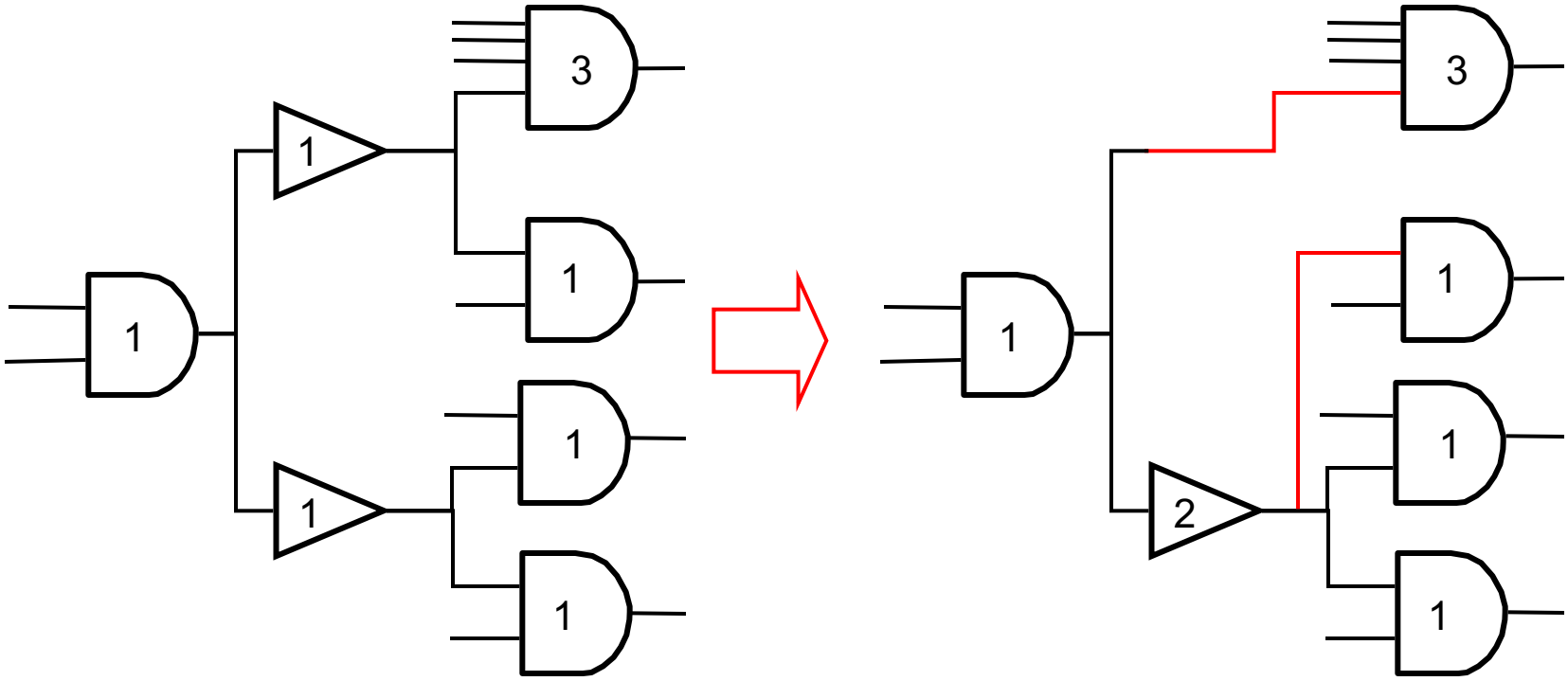
Local Synthesis Transforms

- Redesign Fan-in Tree



Local Synthesis Transforms

- Redesign Fan-out Tree

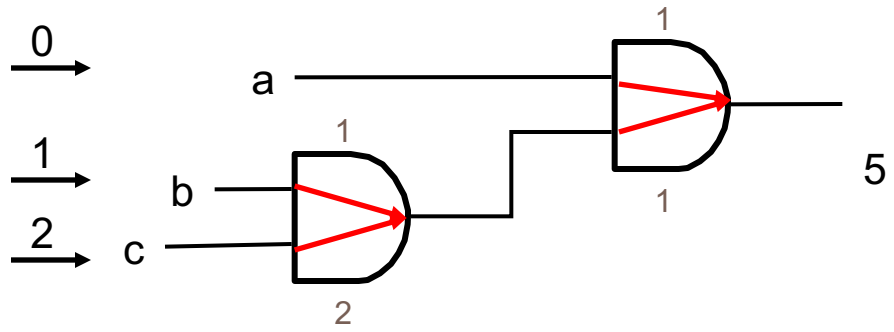


Longest Path = 5

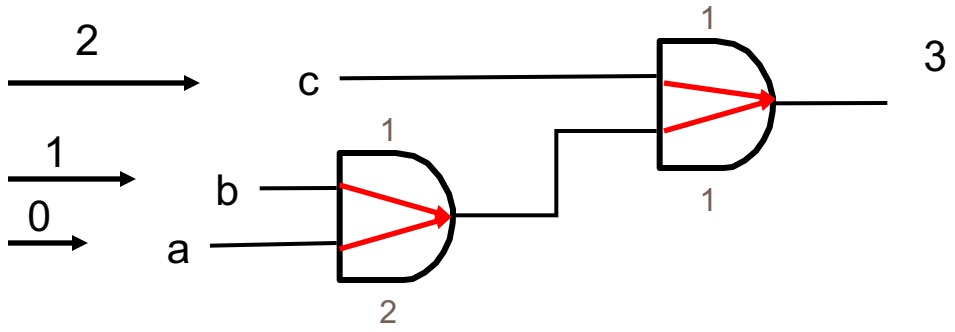
Longest Path = 4
Slowdown of buffer due to load

Local Synthesis Transforms

- Swap Pins

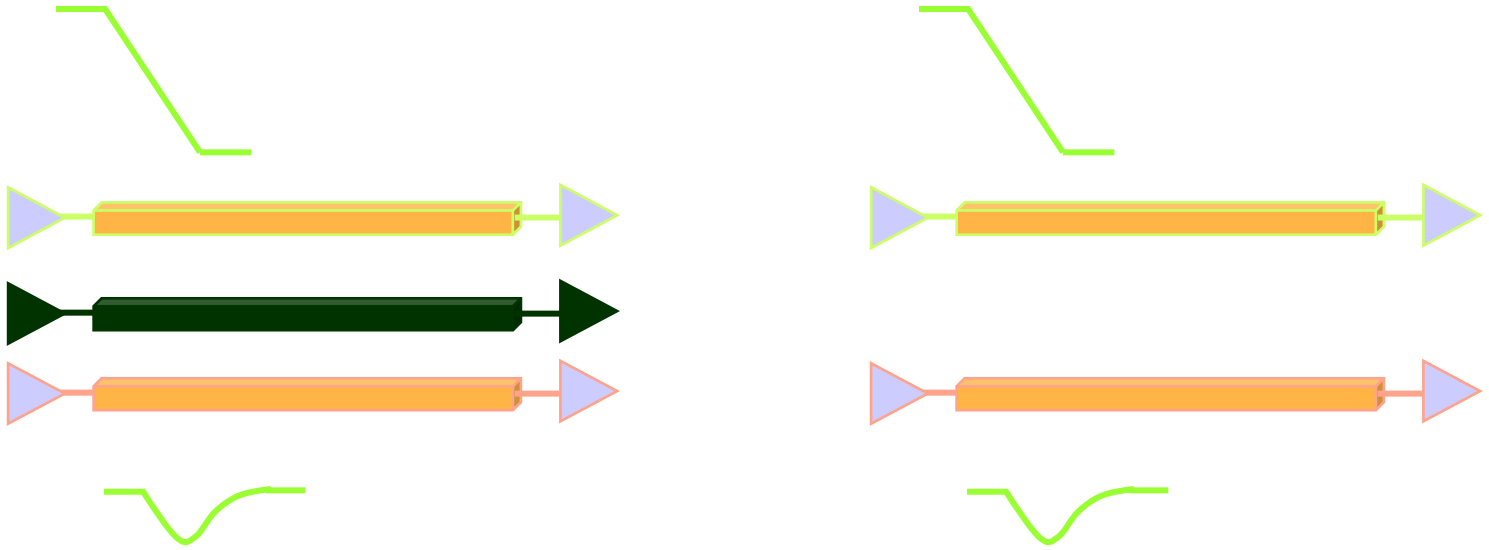


Simple sorting on arrival times and delay works



Post Layout Optimizations

- Re-route to reduce cross-talk effect



Post Layout Optimizations

- Useful Skew