Lecture 14
Performance Metrics for Digital Circuits and Systems

Major Metrics

- Time
  - Delay, latency, throughput
- Power
- Area
- Temperature (Heat Dissipation)
  - Function of Power Density = Power /Area

Timing in Digital Systems

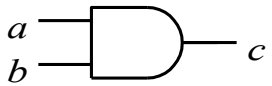
- Delay in combinational paths
- Clock cycle time

Delay in Combinational Circuits

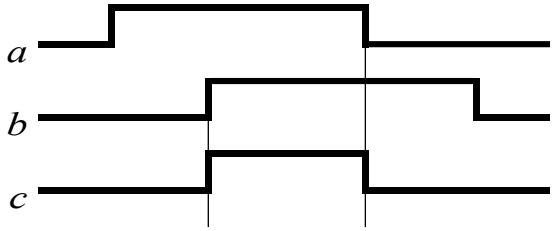
- ***Propagation Delay***
  - Physical characteristics of a logic circuit to be considered:
    - Propagation delays
    - Gate fan-in and fan-out restrictions
    - Interconnect
    - Total chip size (relates to distance to be covered)
  - *Propagation delay*: The delay between the time of an input change and the corresponding output change.
  - Typical two propagation delay parameters:
    - t_{PLH} = propagation delay time, low-to-high-level output
    - t_{PHL} = propagation delay time, high-to-low-level output
  - Approximation:
    - $t_{PD} = \frac{t_{PLH} + t_{PHL}}{2}$

Delay in Combinational Circuits

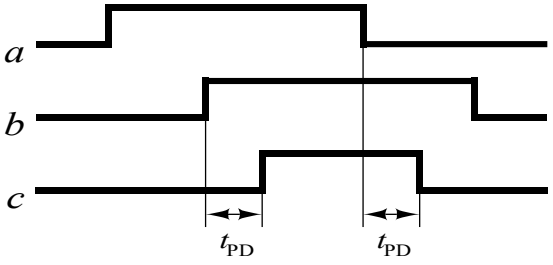
- Propagation delay through a logic gate



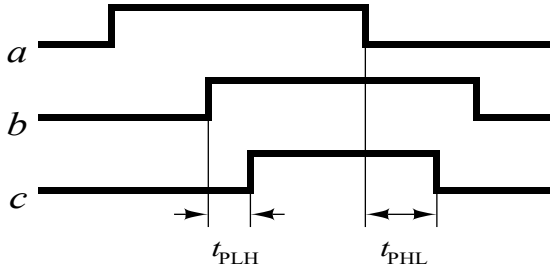
(a) Two-input AND gate



(b) Ideal (zero) delay



(c) $t_{PD} = t_{PLH} = t_{PHL}$

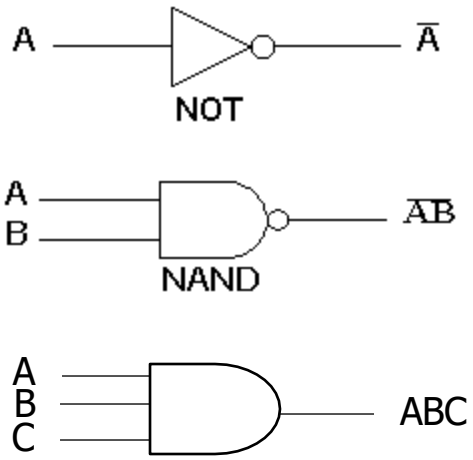


(d) $t_{PLH} < t_{PHL}$

Delay in Logic

Fan-in and Fan-out:

Fan-in: the number of inputs of a gate that it can handle without impairing its normal operation.



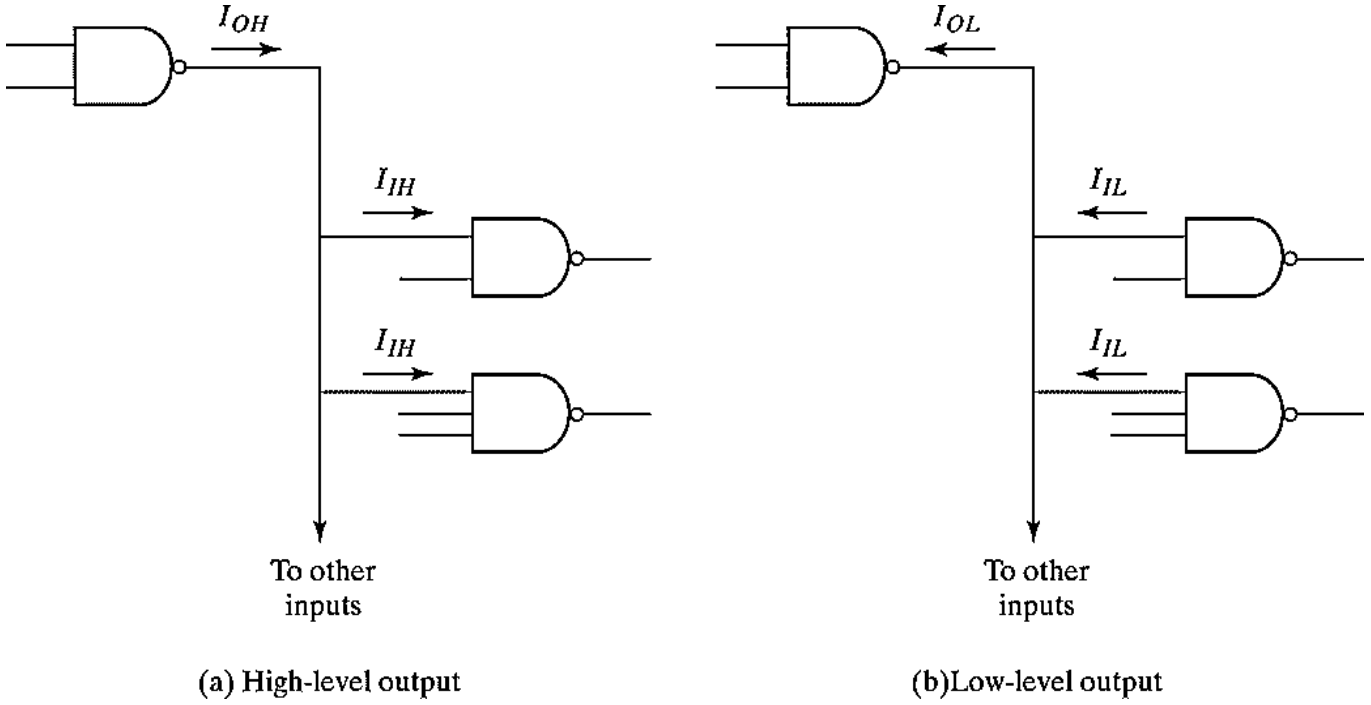
Fan-in = number of inputs

Propagation delay increases with number of inputs. Therefore a 2 input NAND gate is faster than 4 input NAND gate.

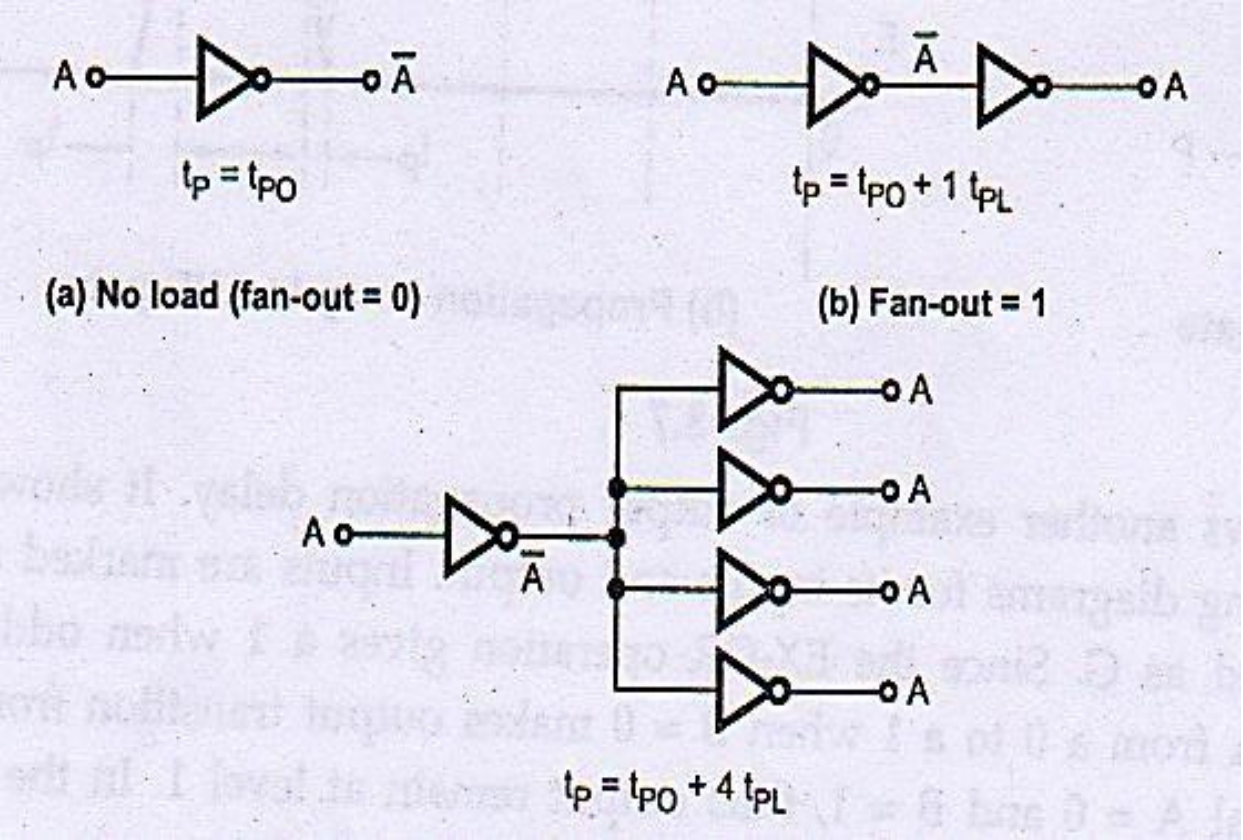
Delay in Logic

Fan-in and Fan-out:

Fan-out: The number of standard loads can be connected to the output of the gate without affecting its normal operation. Sometimes the term loading is used.



Propagation Delay



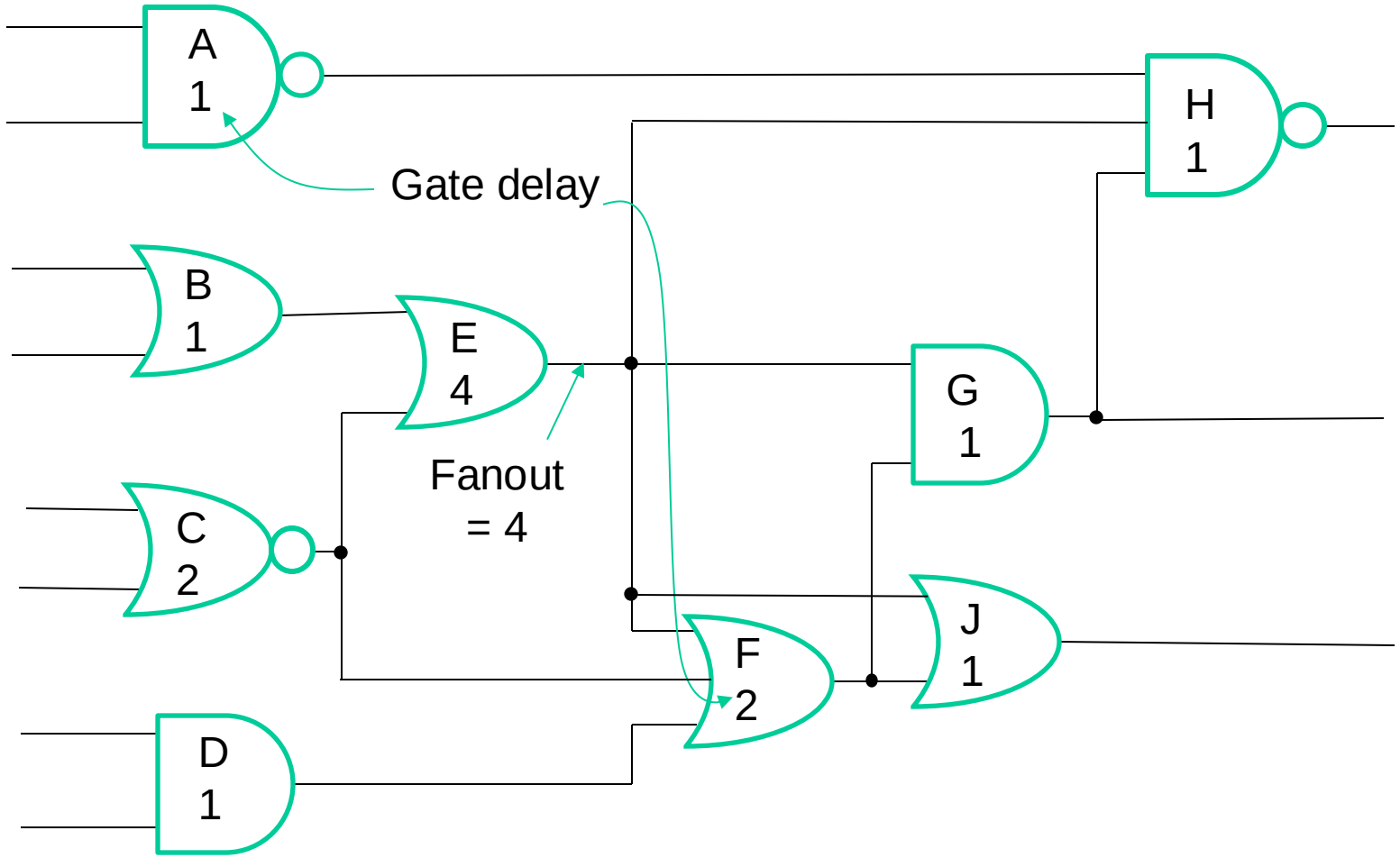
Timing Analysis

- There are two main methods
  - Static Timing Analysis (STA)
  - Dynamic Timing Analysis
- STA: Method of validating the timing performance of a design by checking all possible paths for worst case delay assumptions
- Dynamic Timing Analysis: Determines the timing behavior of the circuit for a given set of input stimulus vectors

Static Timing Analysis (STA)

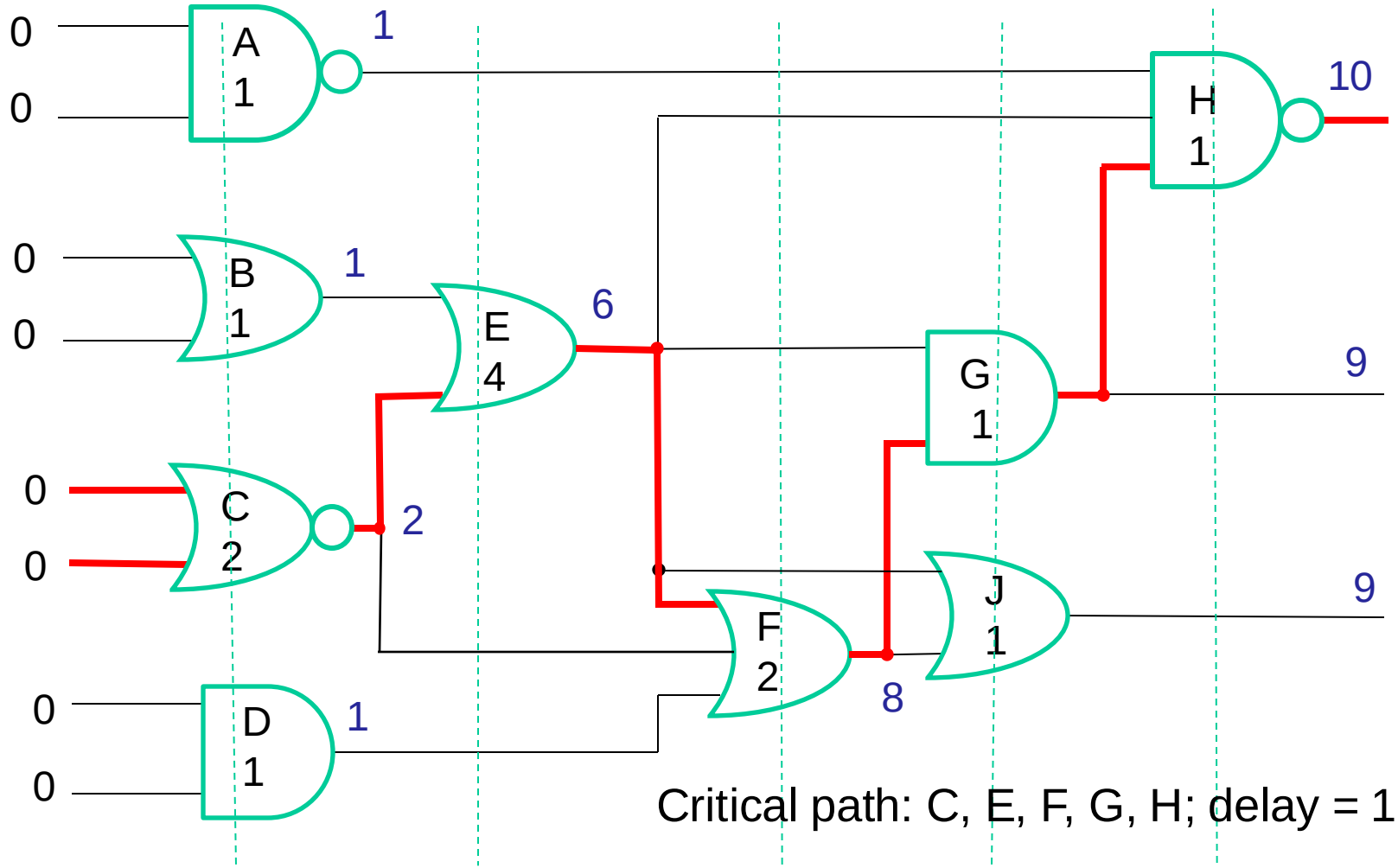
- Combinational logic for critical path delays
- Circuit represented as an acyclic directed graph (DAG)
- Gates characterized by delays; gate function ignored
- No inputs are used – worst-case analysis – static analysis (simulation would be dynamic)

Combinational Circuit



Static Timing Analysis (STA)

Trace critical paths from the output with longest arrival time.



Static Timing Analysis

Timing analysis reports potential issues with a list of timing violations, where specific delay paths with the error are marked

Two kinds of timing errors are possible:

- A **hold time violation**, when a signal arrives too early, and advances one clock cycle before it should.
- A **setup time violation**, when a signal arrives too late, and misses the time when it should advance.

Pros and Cons

- **Static Timing Analysis**

  + faster because it is not necessary to simulate the logical operation of the circuit

  + checks all the possible paths

  - only checks the timing, cannot validate the functionality

- **Dynamic Timing Analysis**

  + can check both the timing and the functionality

  - consumes more run time

  - dependent on choice of stimulus vectors

When do we perform Timing Analysis?

- First - gate-level design (after mapping functionality to logic gates, a.k.a. Logic Synthesis)
  - If free of timing violations → placement and routing of the gates on chip (upon which we obtain accurate delay information or detailed parasitic information)
- Second – after place&route - back-annotated design used in more accurate timing analysis

Static Timing Analysis

Inputs:

- gate-level netlist (.db, Verilog, VHDL); delay information in SDF format; timing constraints (example: Synopsys Design Constraints (SDC) format); gate technology library
- Verifies the design timing using information provided in the technology library

Outputs:

- Timing Report
- Timing Models

Timing Checks performed:

- Setup, hold, recovery, and removal constraints
- User-specified data-to-data timing constraints
- Clock-gating setup and hold constraints
- Minimum period and minimum pulse width for clocks
- Design rules (minimum/maximum transition time, capacitance, and fanout)
- Bus contention and floating net conditions

Timing Paths

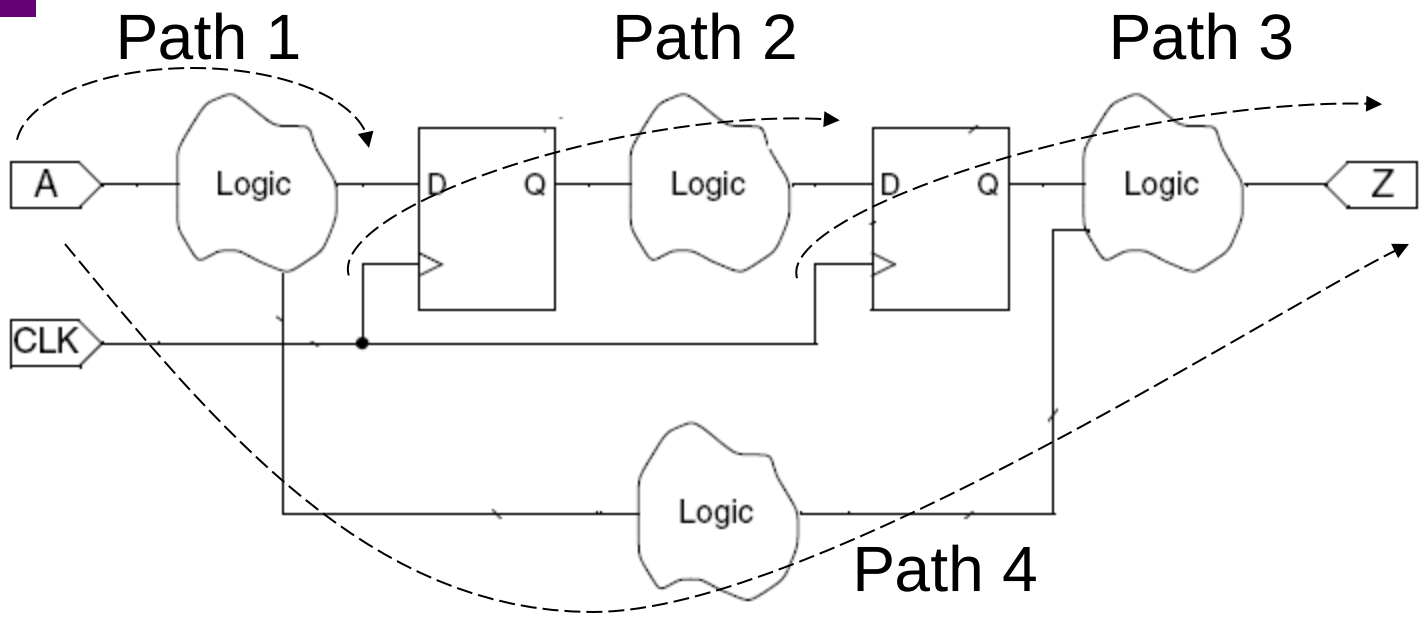
Each path has a startpoint and an endpoint.

The startpoint is a place in the design where data is launched by a clock edge. The data is propagated through combinational logic in the path and then captured at the endpoint by another clock edge.

The startpoint of a path is a clock pin of a sequential element, or possibly an input port of the design (because the input data can be launched from some external source).

The endpoint of a path is a data input pin of a sequential element, or possibly an output port of the design (because the output data can be captured by some external sink).

How many timing paths?



Path 1 starts at an input port and ends at the data input of a sequential element.

Path 2 starts at the clock pin of a sequential element and ends at the data input of a sequential element.

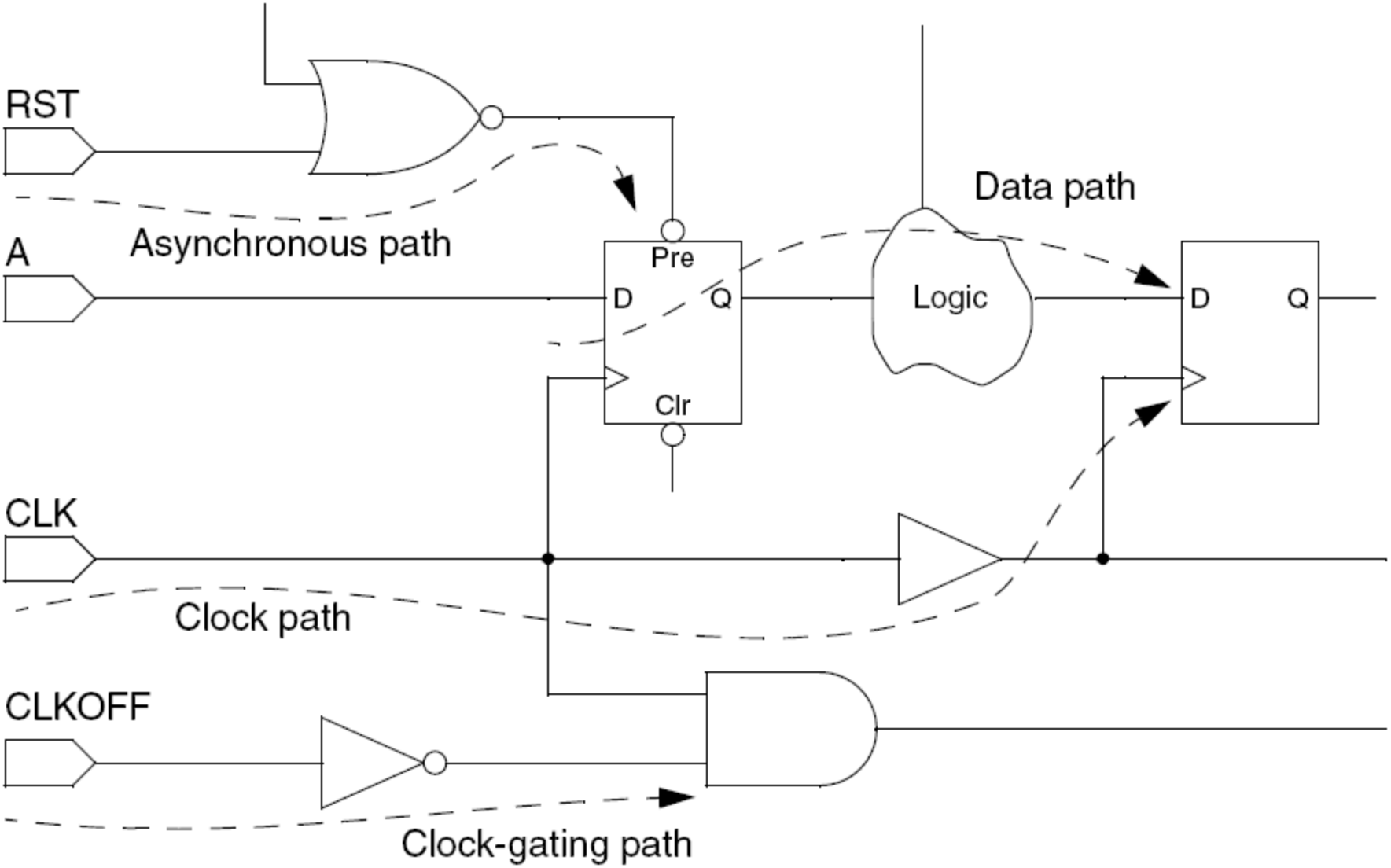
Path 3 starts at the clock pin of a sequential element and ends at an output port.

Path 4 starts at an input port and ends at an output port.

A combinational logic cloud might contain multiple paths. Analysis tools use the longest path to calculate a maximum delay or the shortest path to calculate a minimum delay.

- **Clock path** (a path from a clock input port or cell pin, through one or more buffers or inverters, to the clock pin of a sequential element) for data setup and hold checks
- **Clock-gating path** (a path from an input port to a clock-gating element) for clock-gating setup and hold checks
- **Asynchronous path** (a path from an input port to an asynchronous set or clear pin of a sequential element) for recovery and removal checks
- **Critical Path** (path between an input and an output with the maximum delay) for finding the maximum frequency

Path Types:


RST
A
Asynchronous path
Pre
D
Q
Clr
Data path
Logic
D
Q
CLK
Clock path
CLKOFF
Clock-gating path

Sequential Circuit Timing

- We already discussed timing constraints governing sequential circuits
  - Setup and Hold Constraints

Constraint Checking - Recap

- A **setup** constraint specifies how much time is necessary for data to be present and stable at the input of a sequential device **ahead of the** clock edge that captures the data in the device. This constraint enforces a **maximum** delay on the data path relative to the clock path.

- A **hold** constraint specifies how much time is necessary for data to remain stable at the input of a sequential device **after** the clock edge that captures the data in the device. This constraint enforces a **minimum** delay on the data path relative to the clock path.

Constraint Checking

The amount of time by which a violation is avoided is called the **slack**.

E.g.: for a setup constraint, if a signal must reach a cell input at no later than 8 ns and is determined to arrive at 5 ns, **the slack is +3 ns**.

A slack of 0 means that the constraint is just barely satisfied.

A negative slack indicates a timing violation.

Multi-corner STA

- Analysis with a set of assumptions on the operating conditions
  - Slow, average, fast
  - slow (high temp, low voltage)
  - fast (low temp, high voltage)

Advanced considerations in timing analysis

- **Statistical Static Timing Analysis**

The statistical models for gate delays are built by examining the delay behavior of the gate at different values of the parameters influenced by process and operating conditions.

Process: Fabricated device and interconnect characteristics (variation in the effective gate length, doping concentration, oxide thickness, etc.)

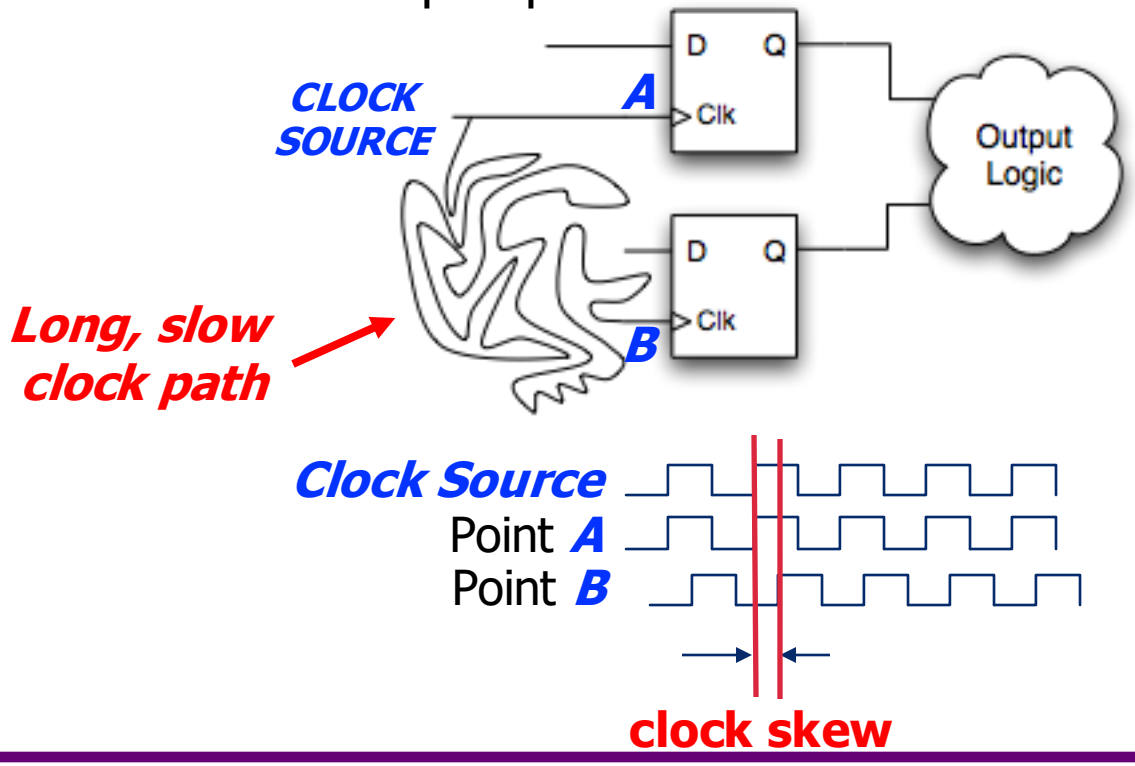
Operating conditions: Uncertainty in environmental conditions during the operation of a chip (power supply and temperature fluctuations)

Additional Digital Circuit Concepts

- A few additional considerations to be discussed
- Clock skew
- Clock Jitter
- Noise Margin
- Glitches

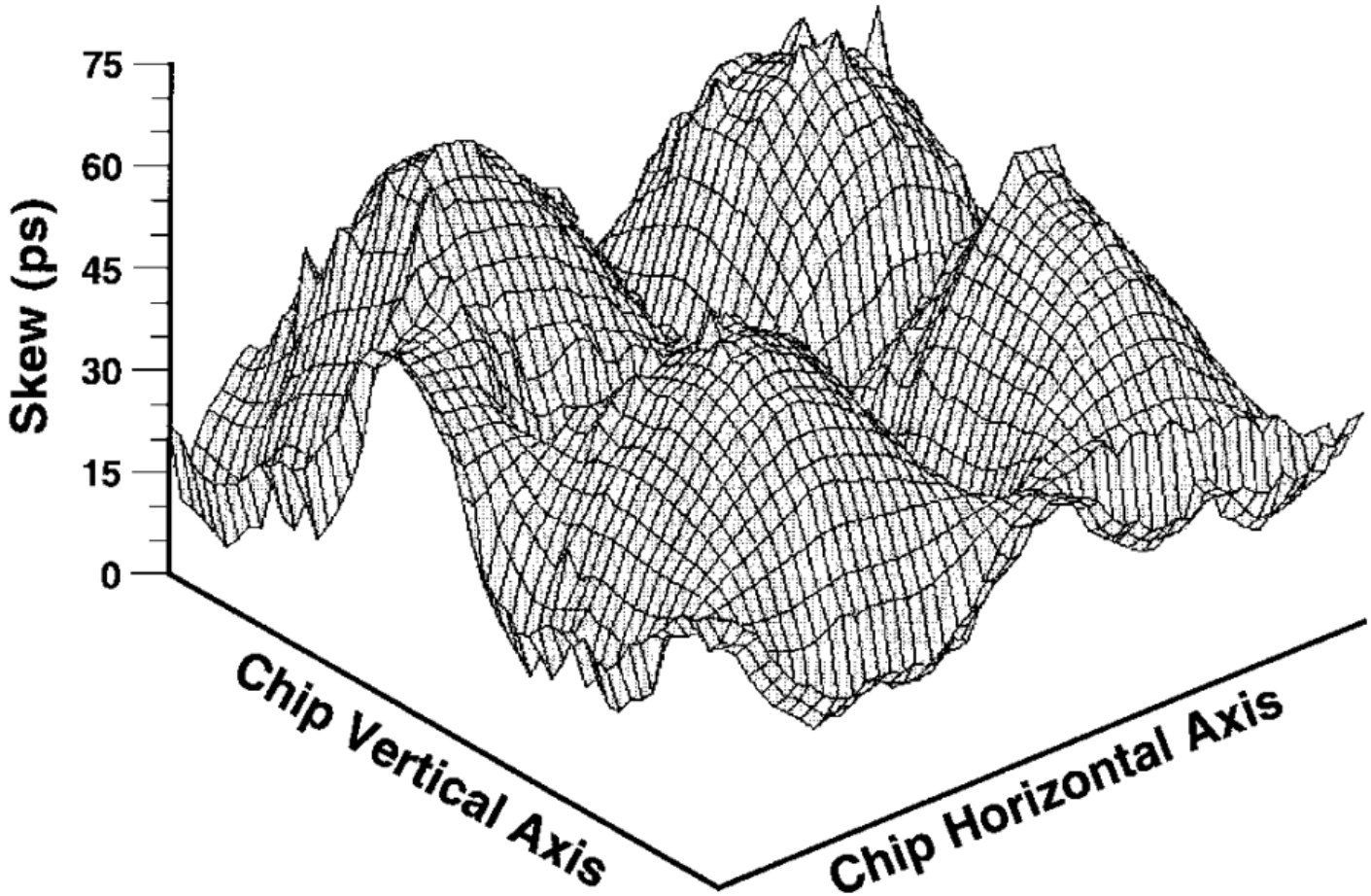
Clock Skew

- To make matters worse, **clock networks have delay** too!
  - The clock does **not** reach all parts of the chip at the same time!
- **Clock skew:** time difference in the arrival time between two clock edges at two different flip flop locations



Clock Skew Example

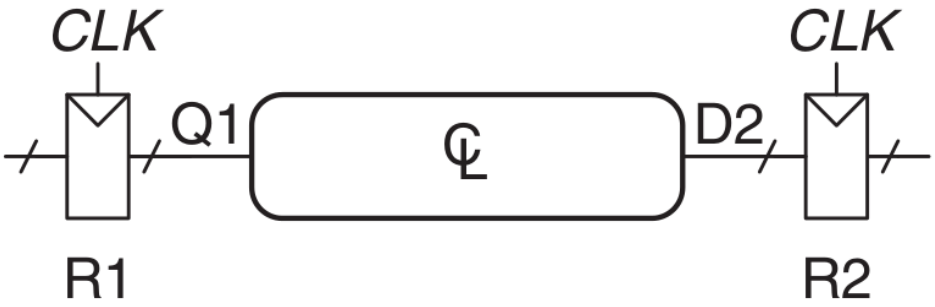
- Example of the **Alpha 21264 processor** clock skew spatial distribution

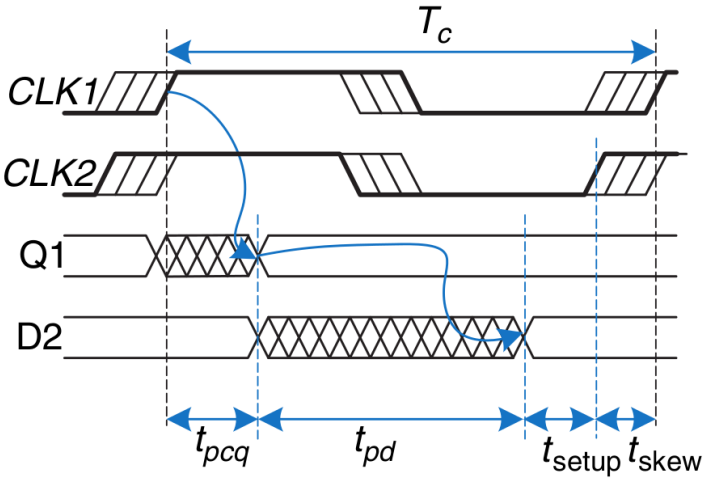


P. E. Gronowski+, "High-performance Microprocessor Design," JSSC'98.

Clock Skew: Setup Time Revisited

- Safe timing requires considering the worst-case skew
  - Clock arrives at R2 *before* R1
  - Leaves as little time as possible for the combinational logic





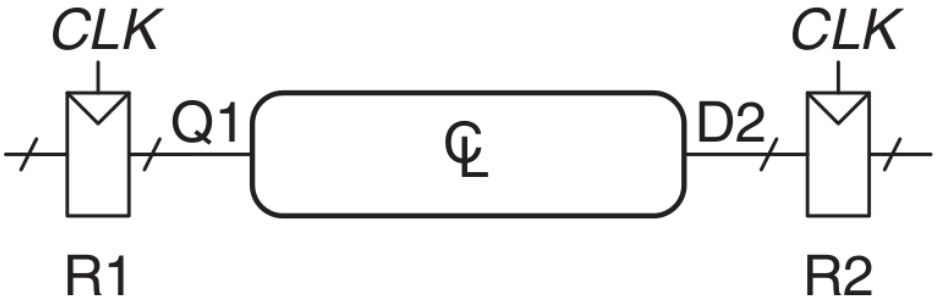
Signal must arrive at D2 ***earlier***!
This effectively ***increases*** t_{setup}:

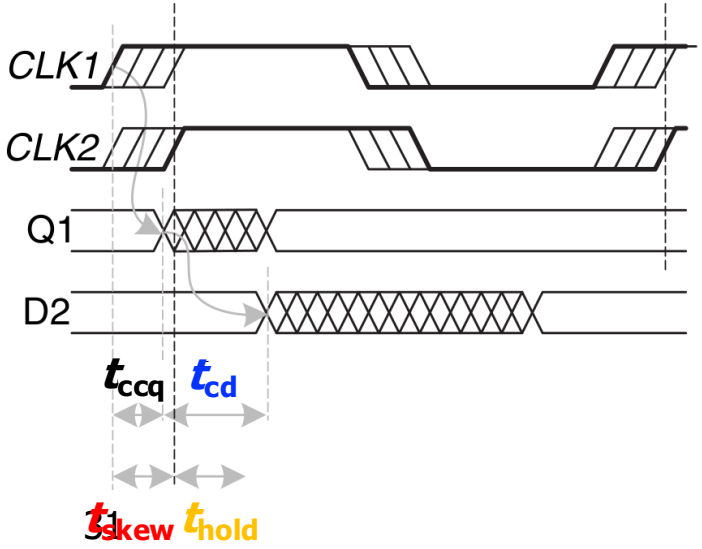
$$T_c > t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$T_c > t_{pcq} + t_{pd} + t_{setup,\ effective}$$

Clock Skew: Hold Time Revisited

- Safe timing requires considering the worst-case skew
  - Clock arrives at R2 *after* R1
  - Increases the minimum required delay for the combinational logic





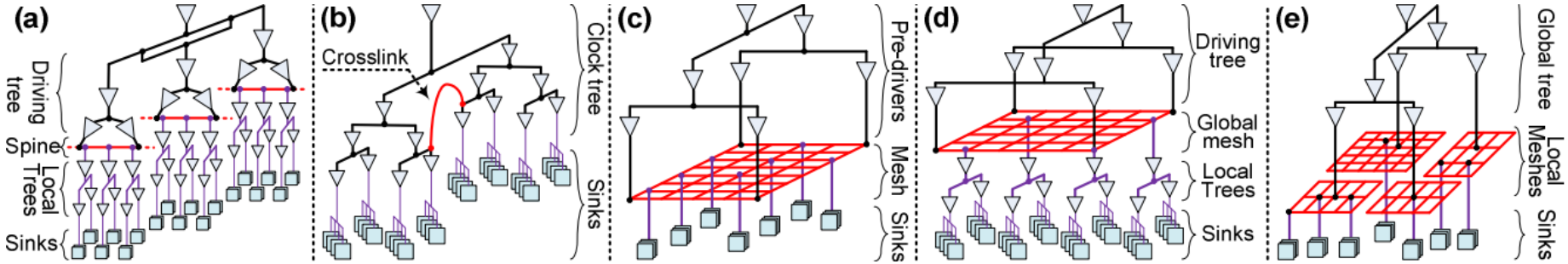
Signal must arrive at D2 ***later***!
This effectively ***increases*** t_{hold}:

$$t_{cd} + t_{ccq} > t_{hold} + t_{skew}$$

$$t_{cd} + t_{ccq} > t_{hold,\ effective}$$

Clock Skew: Summary

- **Skew** effectively forces an **increase in** both $\mathbf{t_{setup}}$ and $\mathbf{t_{hold}}$
  - Increased **sequencing overhead**
  - i.e., less useful work done per cycle
- Designers must keep skew to a **minimum**
  - Requires intelligent **"clock network"** across a chip
  - **Goal: clock** arrives at all locations at roughly the **same time**



Source: Abdelhadi, Ameer, et al. "Timing-driven variation-aware nonuniform clock mesh synthesis." GLSVLSI'10.

Clock Jitter

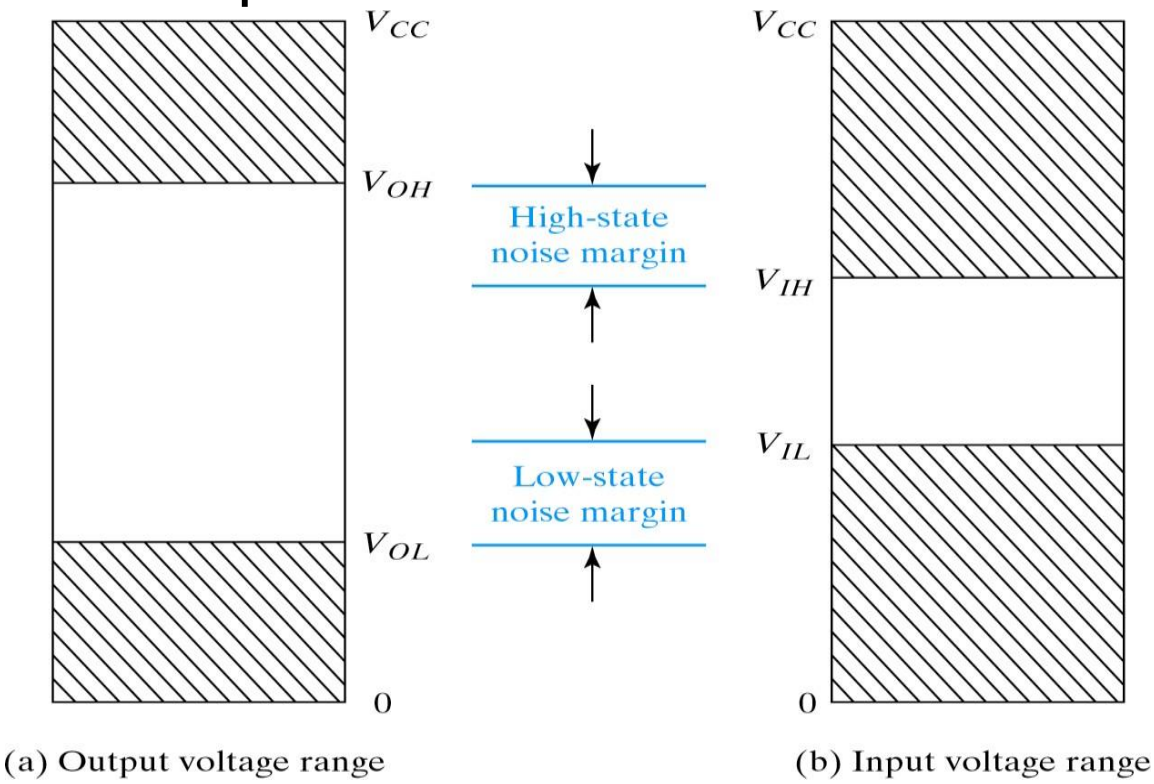
- The random (unknowable, except statistical distribution σ) difference in clock arrival times at the same flip-flop
- Caused by V_{dd}, temperature variation, PLL (a circuit used to create/manage the clock signal) jitter, crosstalk, etc.
- Accounted for in STA by
  - *subtracting* (~3 σ) from the cycle time in setup constraint analysis
  - *adding* to receiving clock arrival time in hold constraint analysis
- So, jitter is always bad in either consideration

Noise Margin

- Noise: Unwanted random disturbance in the digital signals
  - Several sources; some examples are
    - Resistive elements that pass current through them generate a noise signal with fluctuation of their thermal state
    - Moving electric charge generates noise (flow of electric current is actually a sequence of discrete electric charges moving with fluctuations, it is not fluid)

Noise Margin

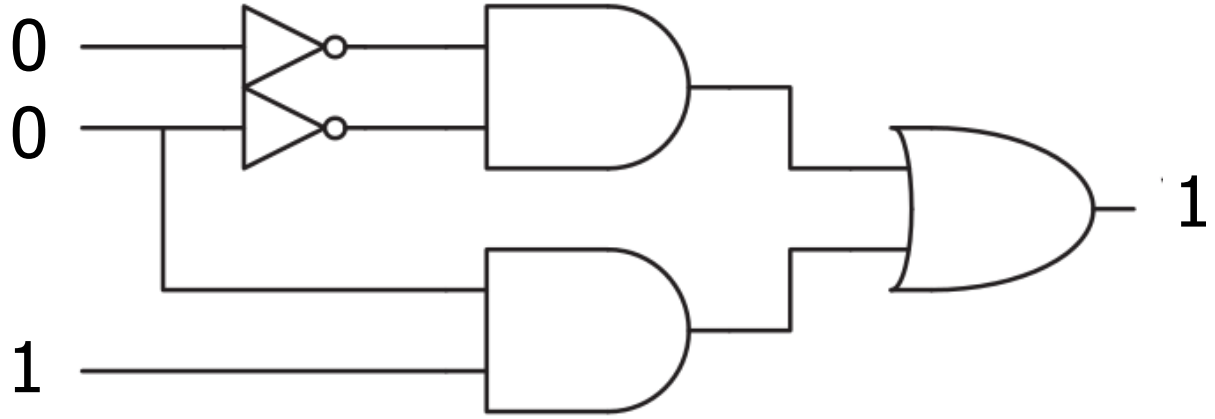
- Noise Margin: the maximum noise added to an input signal of a digital circuit that does not cause an undesirable change in the circuit's output



Glitches

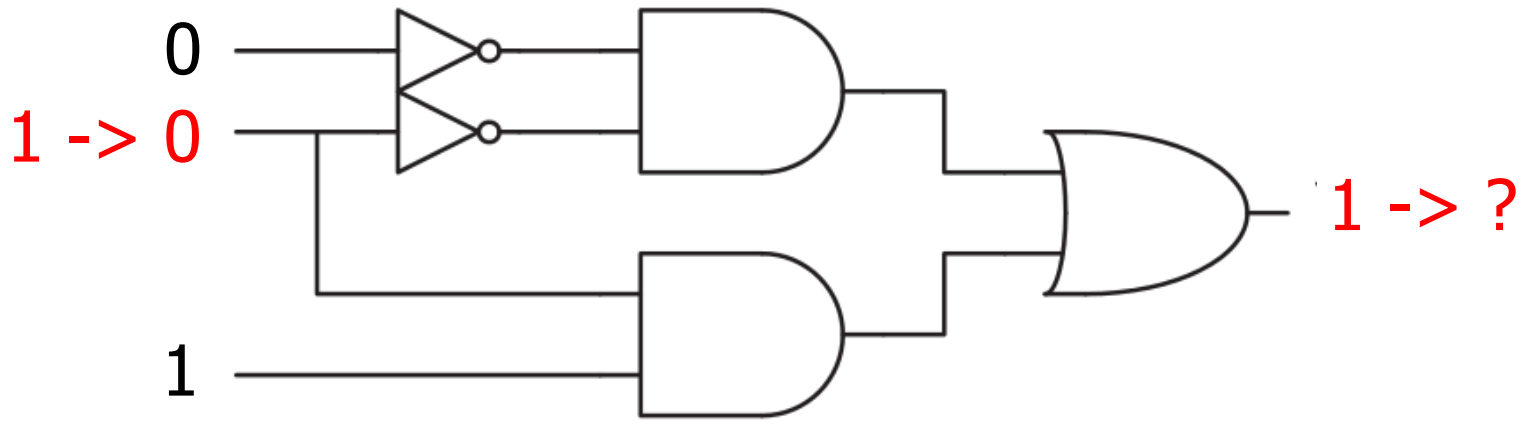
- **a single** input transition causing **multiple** output transitions in a sequence

Circuit initial state



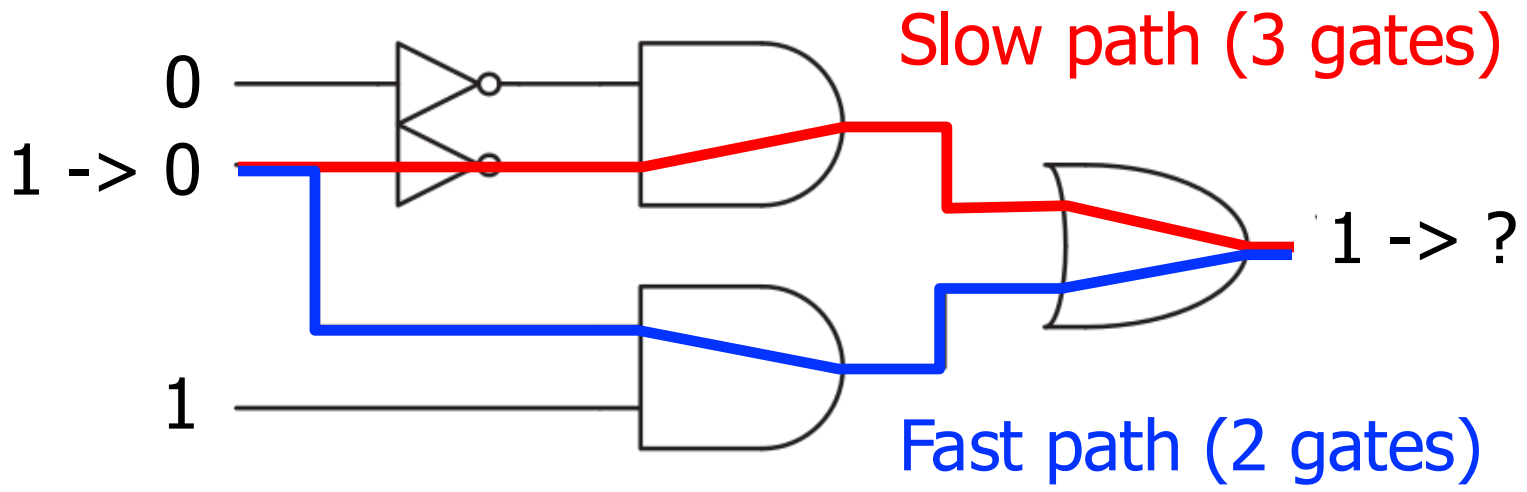
Glitches

- **a single** input transition causing **multiple** output transitions in a sequence



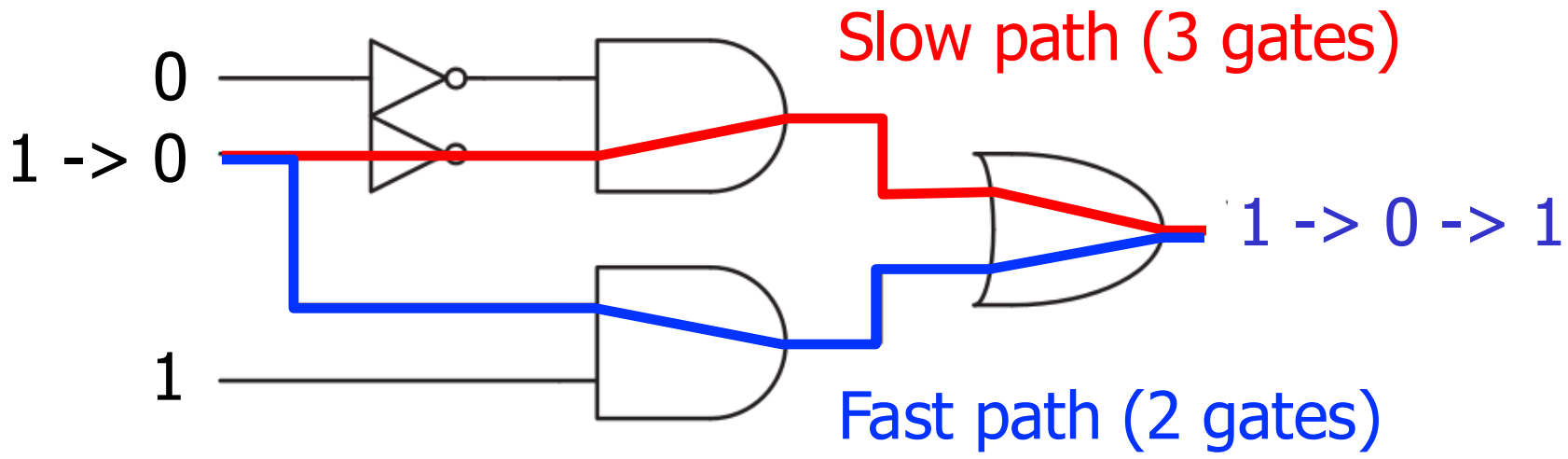
Glitches

- **Glitch: a single** input transition causing **multiple** output transitions in a sequence

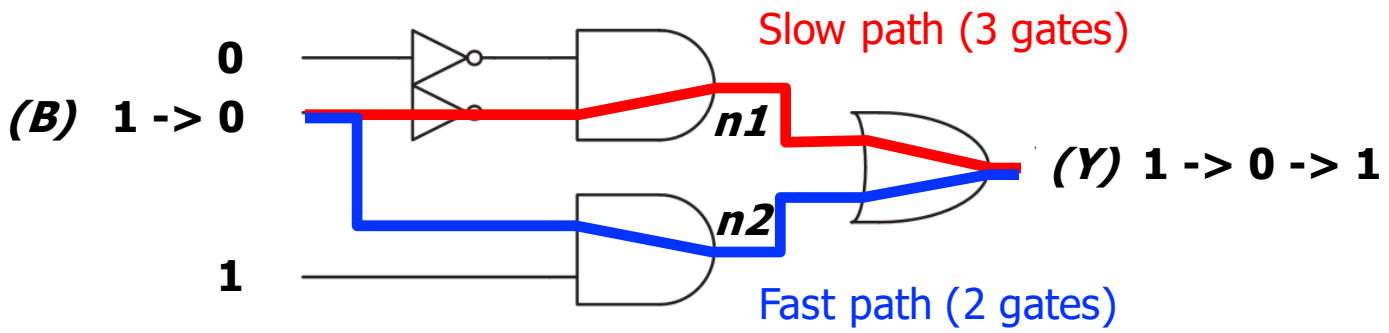


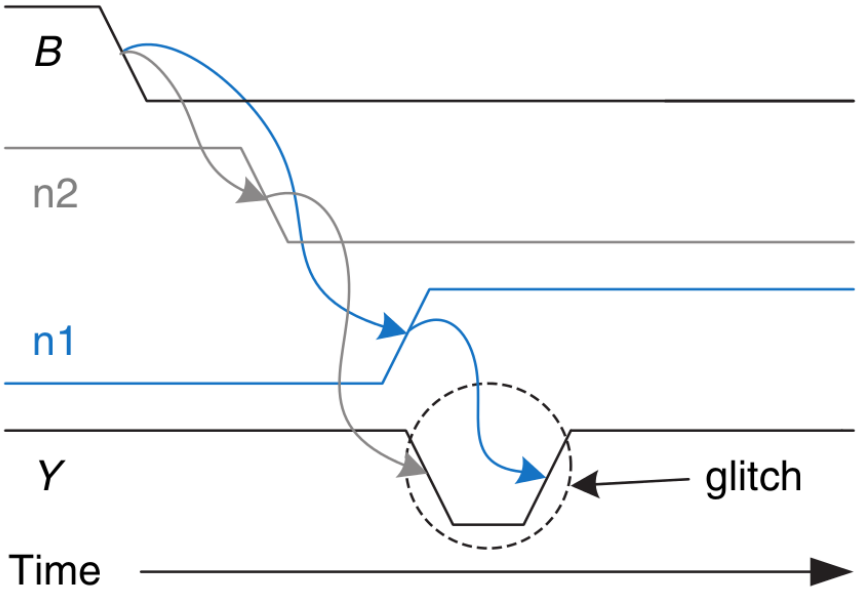
Glitches

- **a single** input transition causing **multiple** output transitions in a sequence



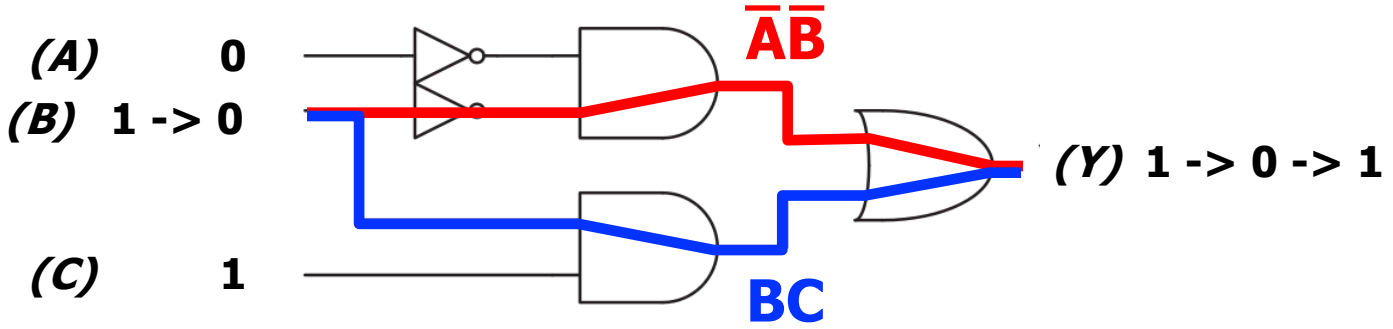
Glitches

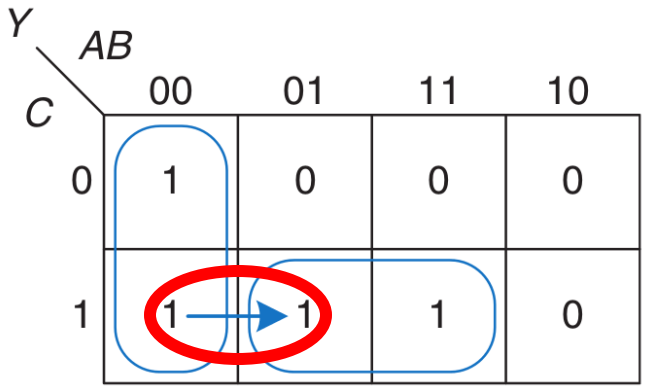

Slow path (3 gates)
0
(B) 1 -> 0
n1
(Y) 1 -> 0 -> 1
n2
1
Fast path (2 gates)


B
n2
n1
Y
glitch
Time

Avoiding Glitches Using K-Maps

- Glitches are **visible** in **K-maps**
  - Recall: K-maps illustrate the consequence of a change in a **single input**
  - A glitch occurs when **transitioning between prime implicants**

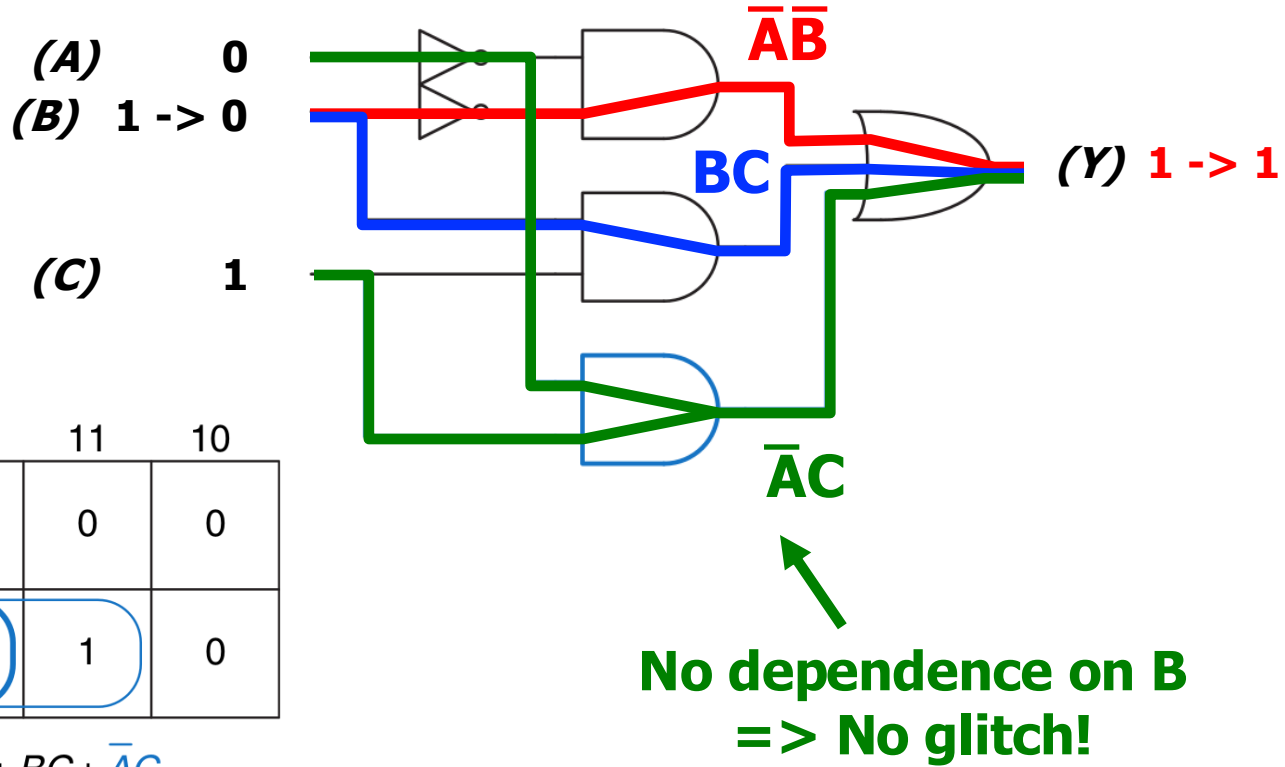


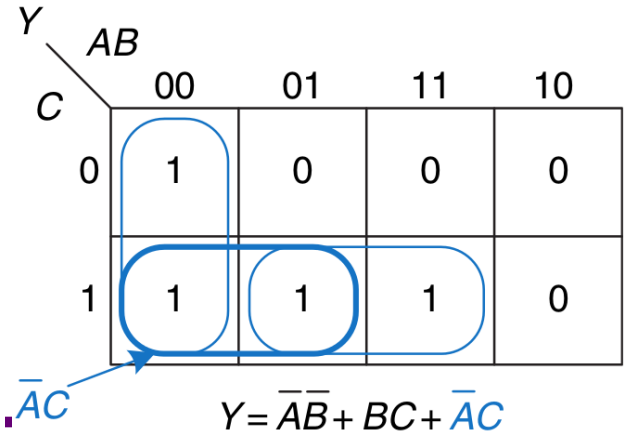


$Y = \bar{A}\bar{B} + BC$

Avoiding Glitches Using K-Maps

- We can fix the issue by adding the **consensus** term
  - Avoids **transition** between different **prime implicants**





Avoiding Glitches

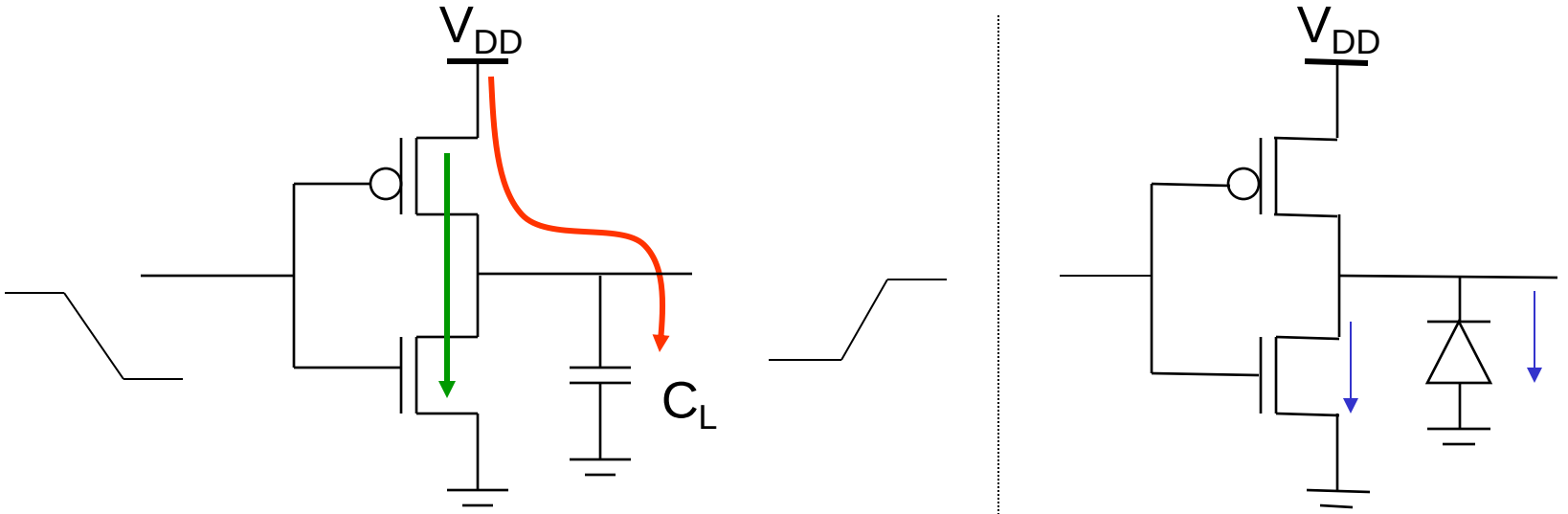
- Do we always care about glitches?
  - Fixing glitches is costly
    - More chip area (more power consumption)
      - **But note that glitches will also waste power**
  - The circuit is eventually guaranteed to converge to the correct value regardless of glitches
- No, not always!
  - If we only care about the long-term steady state output, we can safely ignore glitches
  - Up to the designer to decide if glitches matter in their application

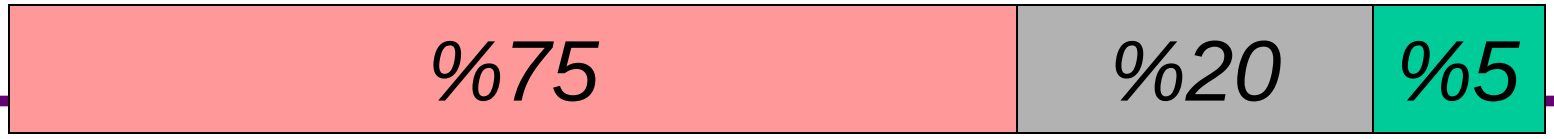
Power Analysis

- Components of Power
- Dynamic
  - Signal transitions
    - Logic activity
    - Glitches
  - Short-circuit (often neglected)
- Static
  - Leakage

Power Dissipation in CMOS Logic

$$P_{total}\ (0 \rightarrow 1) = C_L V_{DD}^2 + t_{sc} V_{DD} I_{peak} + V_{DD} I_{leakage}$$





CMOS Dynamic Power

$$\text{Dynamic Power} = \sum_{\text{All gates } i} 0.5\,\alpha_i\, f_{clk}\, C_{Li}\, V_{DD}^2$$

$$\approx 0.5\,\alpha\, f_{clk}\, C_L\, V_{DD}^2$$

$$\approx \alpha_{01}\, f_{clk}\, C_L\, V_{DD}^2$$

where

α	average gate activity factor
α_{01}	= 0.5α, average 0→1 trans.
f_{clk}	clock frequency
C_L	total load capacitance
V_{DD}	supply voltage

Degrees of Freedom

- The three degrees of freedom are:
  - Supply Voltage
  - Switching Activity
  - Physical capacitance

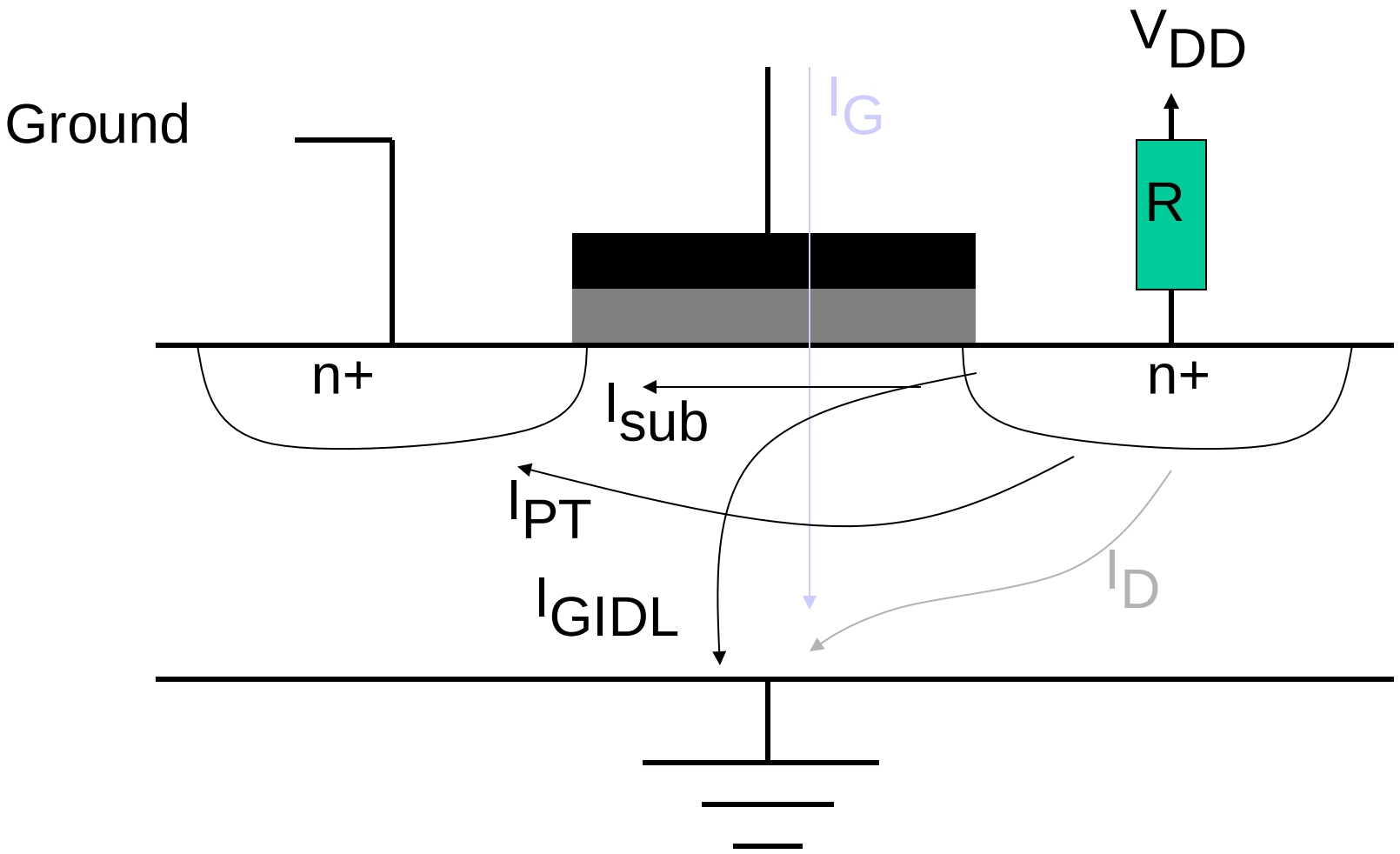
Peak Short Circuit Current

- Increases with the size (or gain, β) of transistors
- Decreases with load capacitance, C_L
- Largest when $C_L = 0$

Short-Circuit Energy

- Increases with rise and fall times of input
- Decreases for larger output load capacitance
- Decreases and eventually approaches zero as V_{DD} is scaled down but the threshold voltages are not scaled down

Leakage Power



Leakage Current Components

- Subthreshold conduction, I_{sub}
- Reverse bias pn junction conduction, I_D
- Gate induced drain leakage, I_{GIDL} due to tunneling at the gate-drain overlap
- Drain source punchthrough, I_{PT} due to short channel and high drain-source voltage
- Gate tunneling, I_G through thin oxide

Subthreshold Current

$$I_{sub} = \mu_0 C_{ox} (W/L) V_t^2 \exp\{(V_{GS}-V_{TH})/nV_t\}$$

μ_0: carrier surface mobility
C_{ox}: gate oxide capacitance per unit area
L: channel length
W: gate width
$V_t = kT/q$: thermal voltage
n: a technology parameter

I_{DS} for Short Channel Device

$$I_{sub} = \mu_0 C_{ox} (W/L) V_t^2 \exp\{(V_{GS}-V_{TH}+\eta V_{DS})/nV_t\}$$

V_{DS} = drain to source voltage
η: a proportionality factor

Summary: Leakage Power

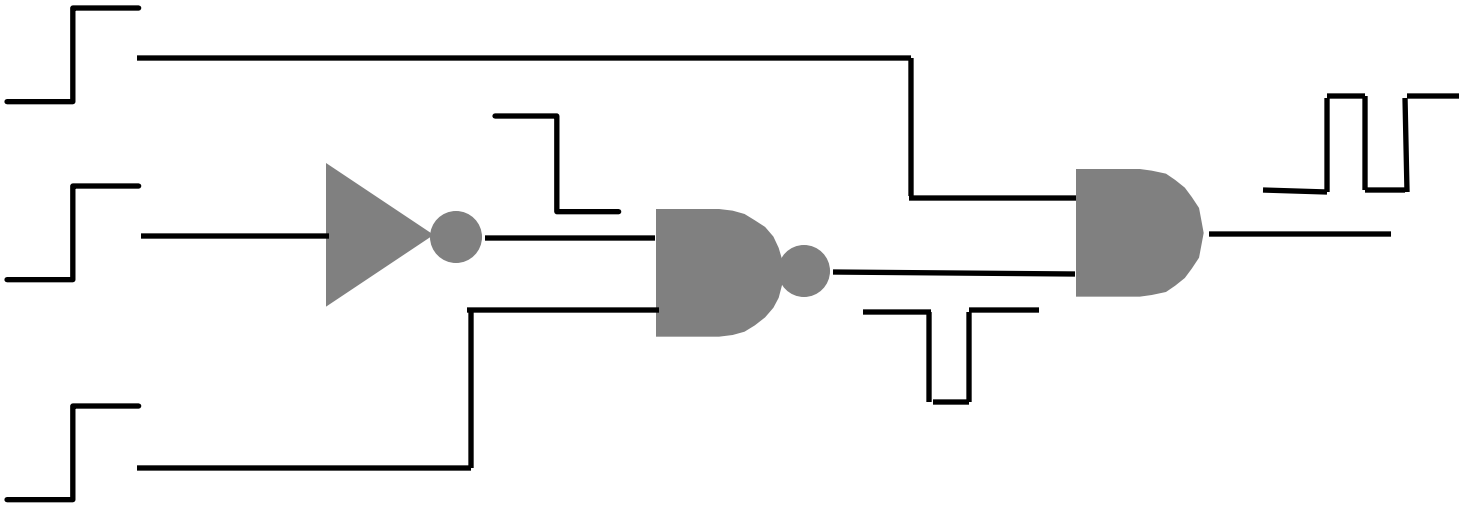
- Leakage power as a fraction of the total power increases as clock frequency drops. *Turning supply off in unused parts can save power*.
- For a single gate it is a small fraction of the total power; it can be significant for very large circuits.
- Scaling down feature sizes requires lowering the threshold voltage, which increases leakage power.
- Multiple-threshold devices and novel transistor structures are used to reduce leakage power.

Dynamic Power

- Each transition of a gate consumes $CV^2/2$.
- Methods of power saving:
  - Minimize load capacitances
    - Transistor sizing
    - Library-based gate selection
  - Reduce transitions
    - Minimize logic design
    - Glitch reduction

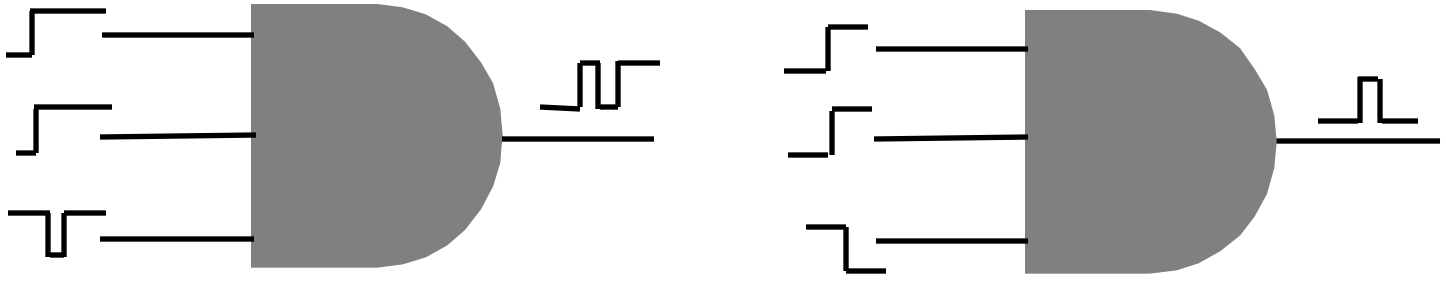
Glitch Power Reduction

- Design a digital circuit for minimum transient energy consumption by eliminating hazards



Theorem 1

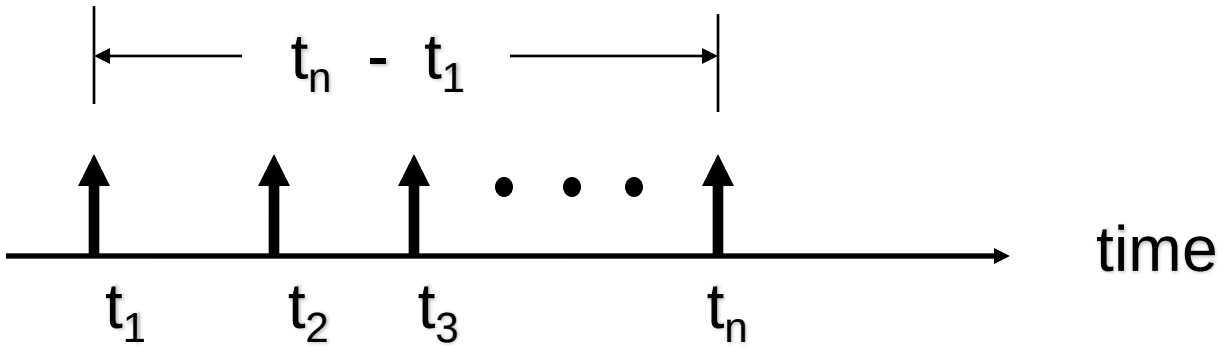
- For correct operation with minimum energy consumption, a Boolean gate must produce no more than *one* event per transition



Theorem 2

- Given that events occur at the input of a gate (inertial delay = d) at times $t_1 < \ldots < t_n$, the number of events at the gate output cannot exceed

$$\min\left(n,\ 1 + \left\lfloor \frac{t_n - t_1}{d} \right\rfloor\right)$$



Minimum Transient Design

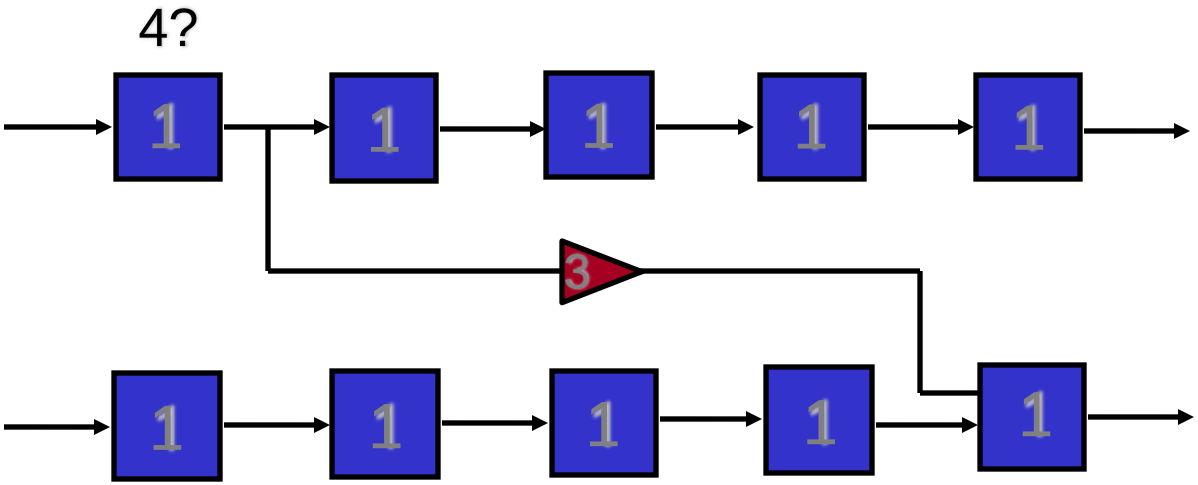
- Minimum transient energy condition for a Boolean gate:

$$| t_i - t_j | < d$$

Where t_i and t_j are arrival times of input events and d is the inertial delay of gate

Balanced Delay Method

- All input events arrive simultaneously
- Overall circuit delay not increased
- Delay buffers may have to be inserted



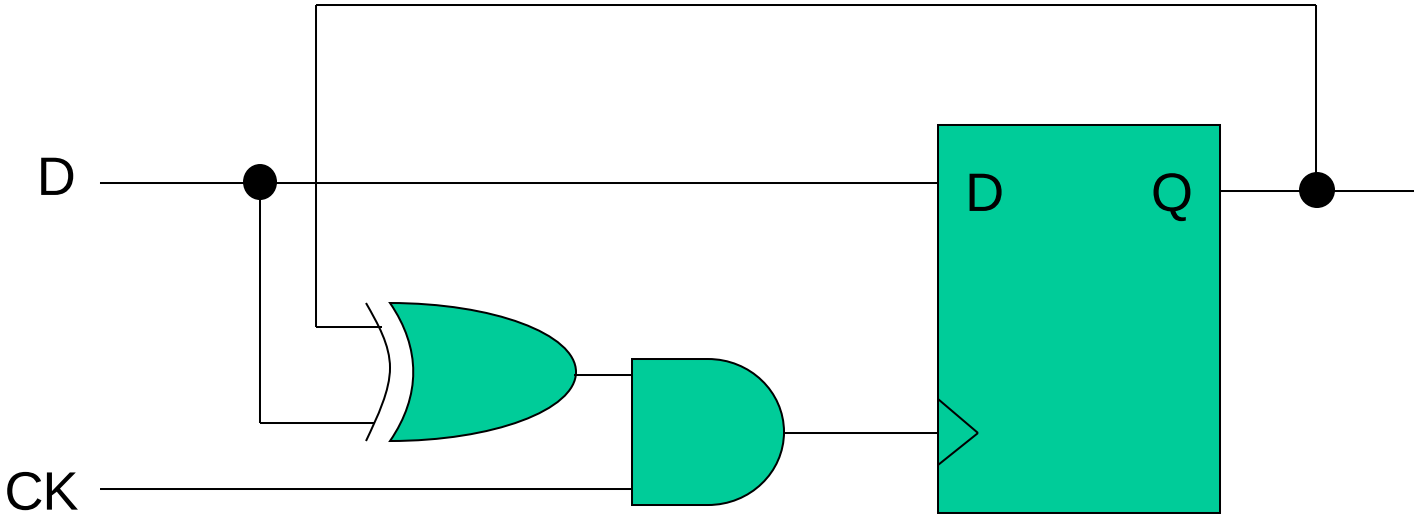
Avoiding Wasted Power on Glitches

- Prevent glitches by design
  - As discussed earlier in this lecture

Low-Power System Design

- State encoding
  - Bus encoding
    - Control frequency of bits flipping as they get transmitted for communication
  - Finite state machine
    - Control bit flips to generate "next state" codes
- Clock gating
  - Flip-flop
  - Shift register

Clock-Gating in Low-Power Flip-Flop



- If no new (different) data is arriving, do not activate the FF

Power Reduction in Digital Circuits

- Hardware methods:
  - Voltage reduction for dynamic power
  - Dual-threshold devices for leakage reduction
  - Clock gating, frequency reduction
  - Sleep mode

Moore's Law and Dennard Scaling

The way things should work…

- Moore's Law: transistor density doubles every N years (currently N ~ 2)
- Dennard Scaling (constant electric field)
  - Shrink feature size by k (typ. 0.7), hold electric field constant
  - Area scales by k^2 (1/2) , C, V, delay reduce by k
  - $P \cong CV^2f \Rightarrow$ P goes down by k^2
  - **Power density = P/A = 1**

The Real Power Wall

- V_{dd} scaling is coming to a halt
  - Currently 0.9-1.0V, scaling only ~2.5%/gen
- V_{dd} reductions were stopped by leakage
- Lower V_{dd} => Vth must be lower
- Leakage depends exponentially on V_{th}
- Leakage is also exponential in Temperature T

The Real Power Wall

- Even if we generously assume C scales and frequency is flat
  - $P \cong CV^2f \Rightarrow 0.7\ (0.975^2)\ (1) = 0.66$
- Power *density* goes up
  - P/A = 0.66/0.5 = 1.33
  - And this is very optimistic, because C probably scales more like 0.8 or 0.9, and we want frequency to go up, so a more likely number is 1.5-1.75X
- But max TDP (Thermal Design Point) for air cooling is expected to stay flat
  - Around 200-250 W total and around 1.5 W/mm^2
  - Viable, *affordable* alternatives for cooling still not yet apparent