CAL POLY
SAN LUIS OBISPO

EE 531: ADVANCED VLSI DESIGN

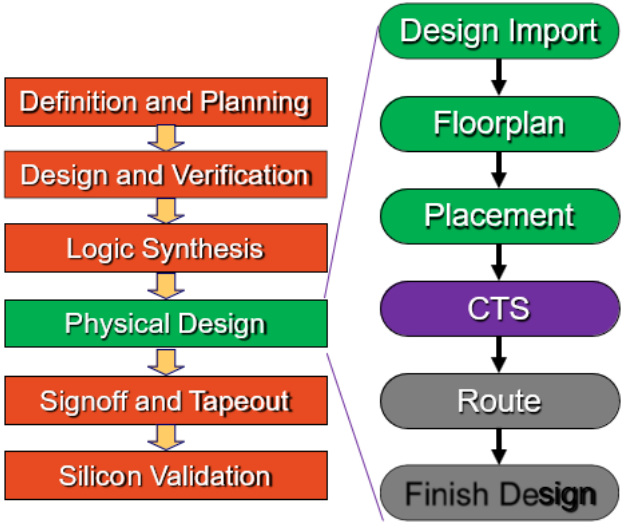
Clock Tree Synthesis

Nishith N. Chakraborty

February, 2025

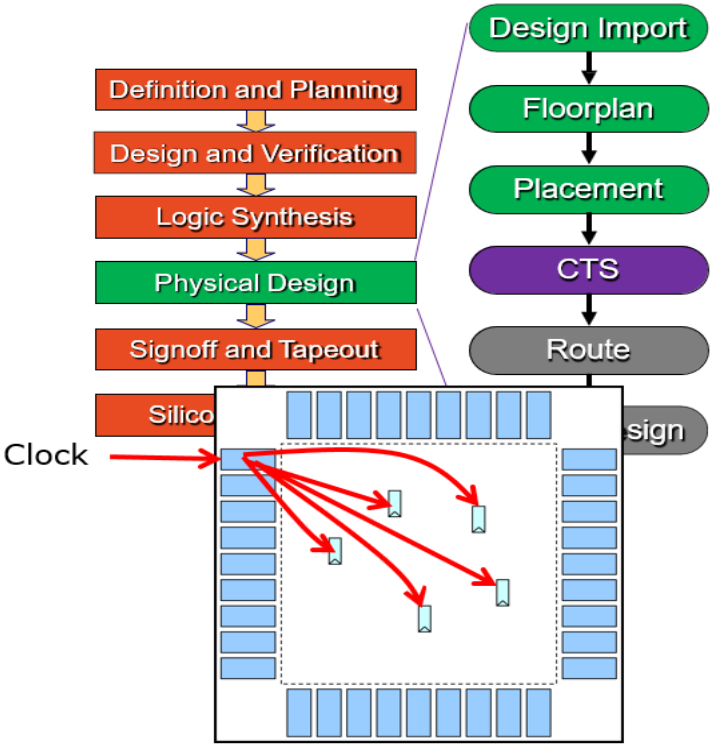
DESIGN FLOW: OUR PROGRESS

- We have:
  - Synthesized our design into a technology mapped gate-level netlist
  - Designed a floorplan for physical implementation.
  - And provided a location for each and every gate.
- During all stages:
  - We analyzed timing constraints and optimized the design according to these constraints.



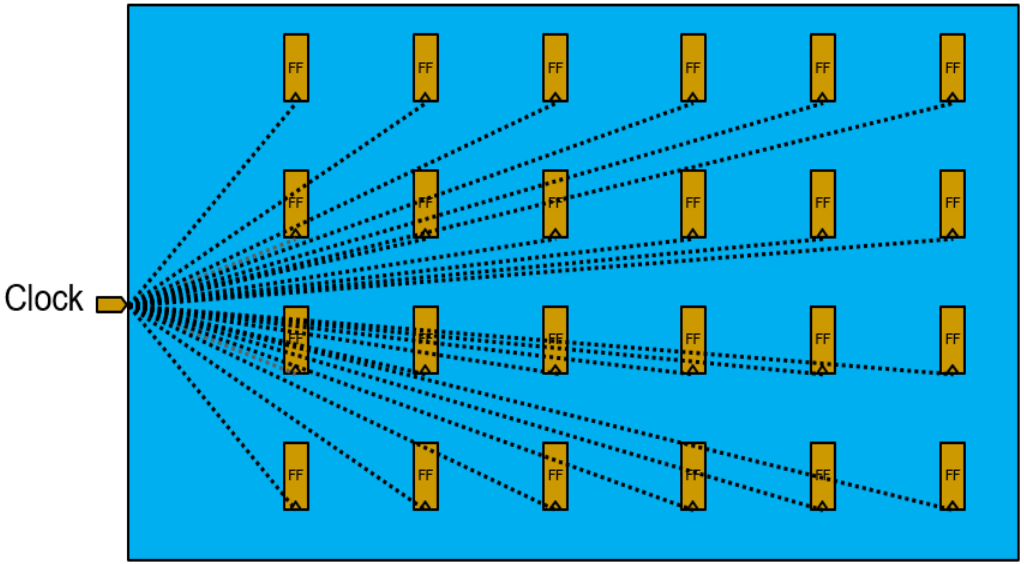
DESIGN FLOW: OUR PROGRESS

- We have:
  - Synthesized our design into a technology mapped gate-level netlist
  - Designed a floorplan for physical implementation.
  - And provided a location for each and every gate.
- During all stages:
  - We analyzed timing constraints and optimized the design according to these constraints.
- However…
  - Until now, we have assumed an ideal clock.
  - Now we have all sequential elements placed, so we have to provide them with a real clock signal.



TRIVIAL APPROACH?

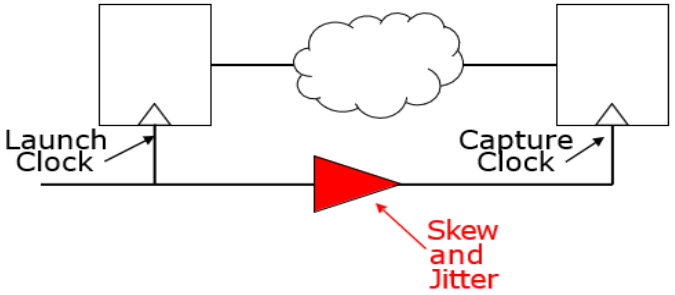
- **Question:**
  - Why not just route the clock net to all sequential elements, just like any other net?
- **Answer:**
  - Timing
  - Power
  - Area
  - Signal Integrity
  - etc…

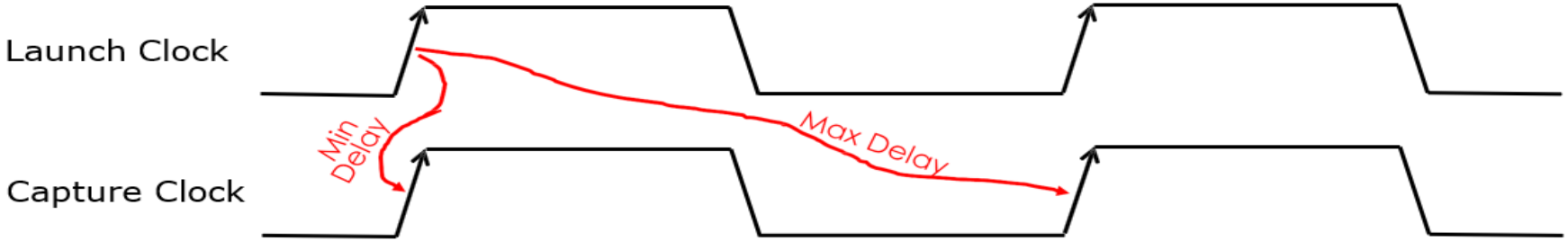


Implications of Clocking

IMPLICATIONS ON TIMING

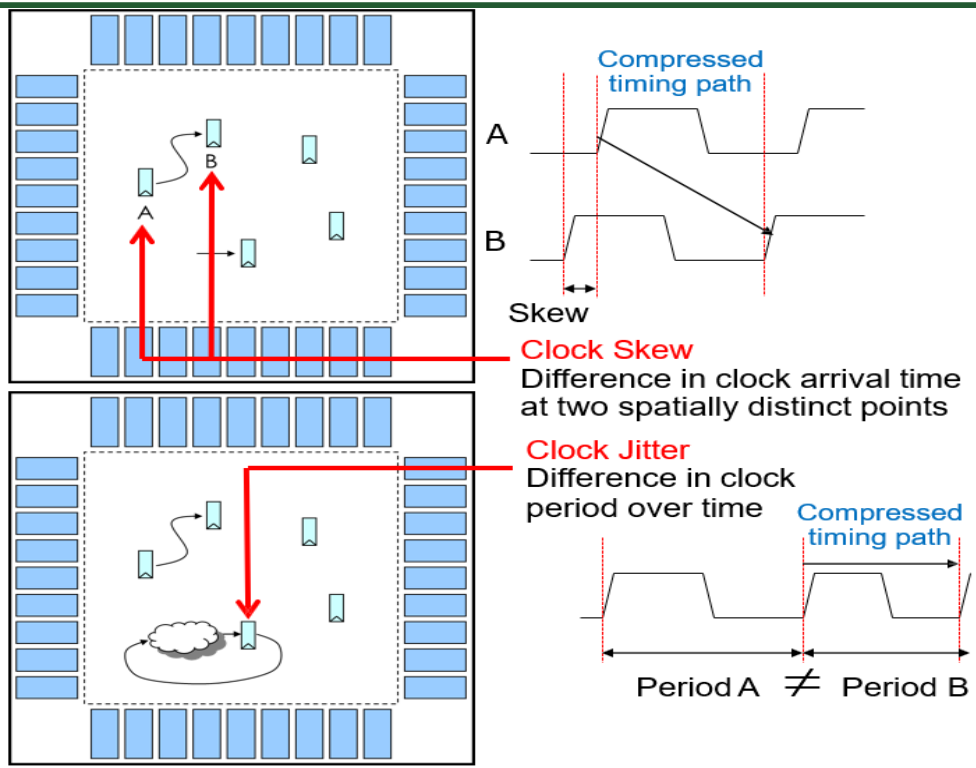
- **Let's remember the timing constraints:**
  - Max Delay: $T + \delta_{skew} > t_{CQ} + t_{logic} + t_{setup} + \delta_{margin}$
  - Min Delay: $t_{CQ} + t_{logic} - \delta_{margin} > t_{hold} + \delta_{skew}$





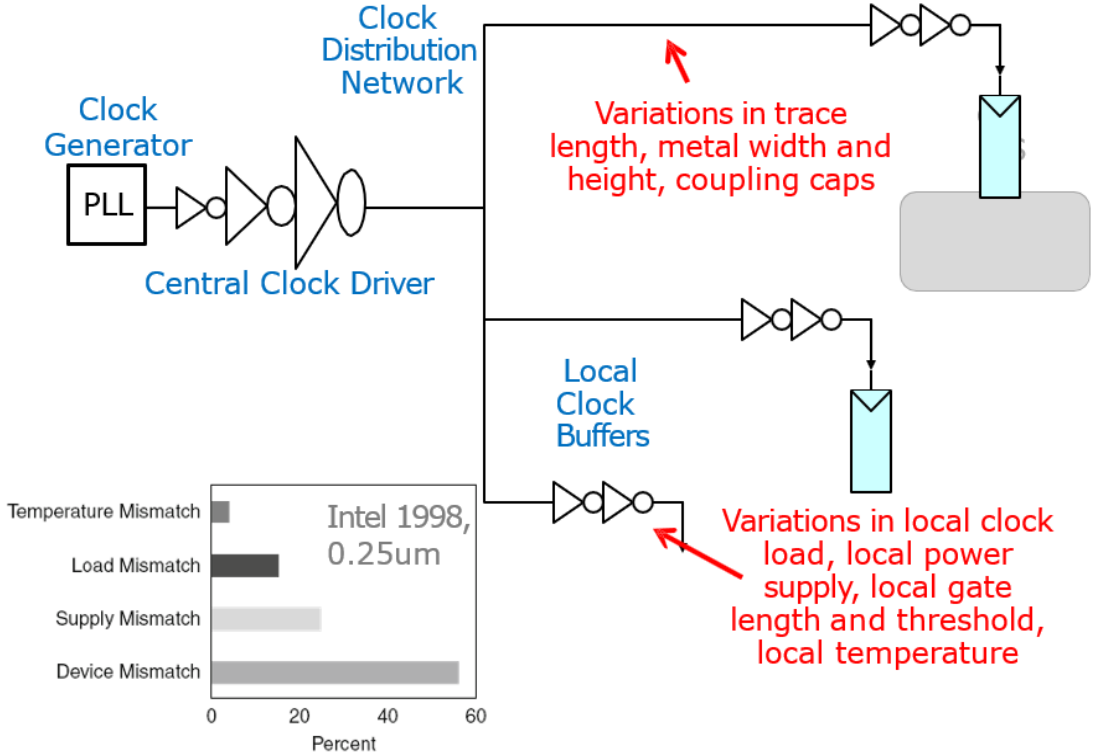
CLOCK PARAMETERS

- Skew
  - Difference in clock arrival time at two different registers.
- Jitter
  - Difference in clock period between different cycles.
- Slew
  - Transition (t_{rise}/t_{fall}) of clock signal.
- Insertion Delay
  - Delay from clock source until registers.



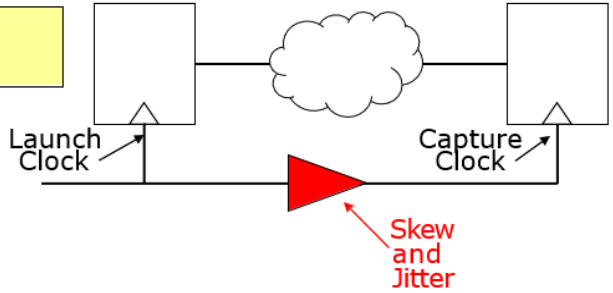
HOW DO CLOCK SKEW AND JITTER ARISE?

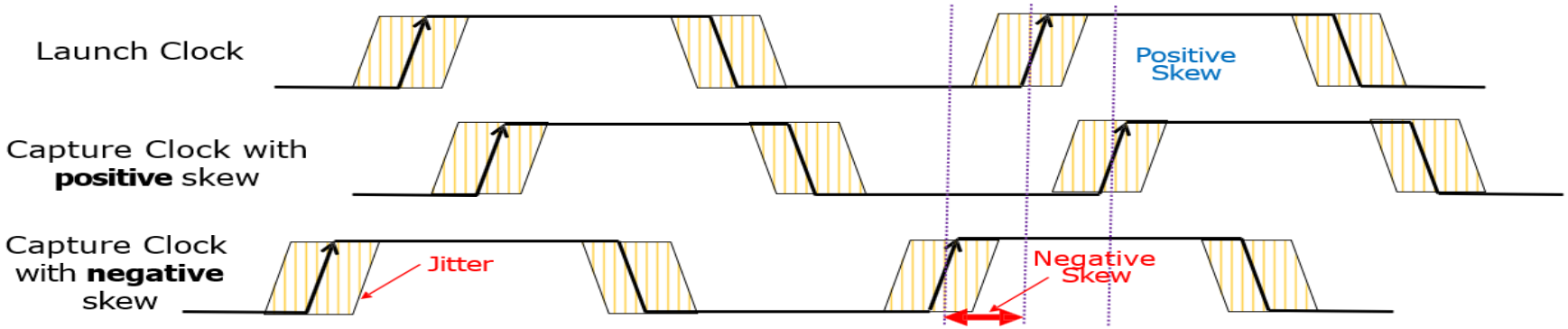
- Clock Generation
- Distribution network
  - Number of buffers
  - Device Variation
  - Wire length and variation
  - Coupling
  - Load
- Environment Variation
  - Temperature
  - Power Supply



IMPLICATIONS ON TIMING

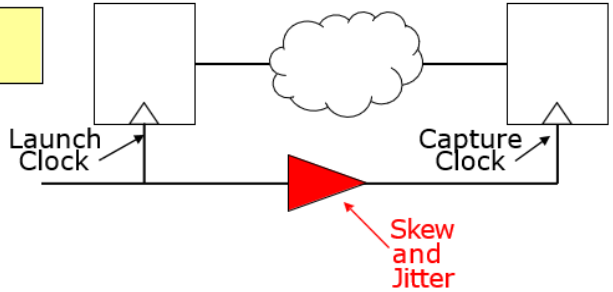
- **Let's remember the timing constraints:**
  - Max Delay: $T + \delta_{skew} - 2\delta_{jitter} > t_{CQ} + t_{logic} + t_{setup} + \delta_{margin}$
  - Min Delay: $t_{CQ} + t_{logic} - \delta_{margin} > t_{hold} + \delta_{skew}$

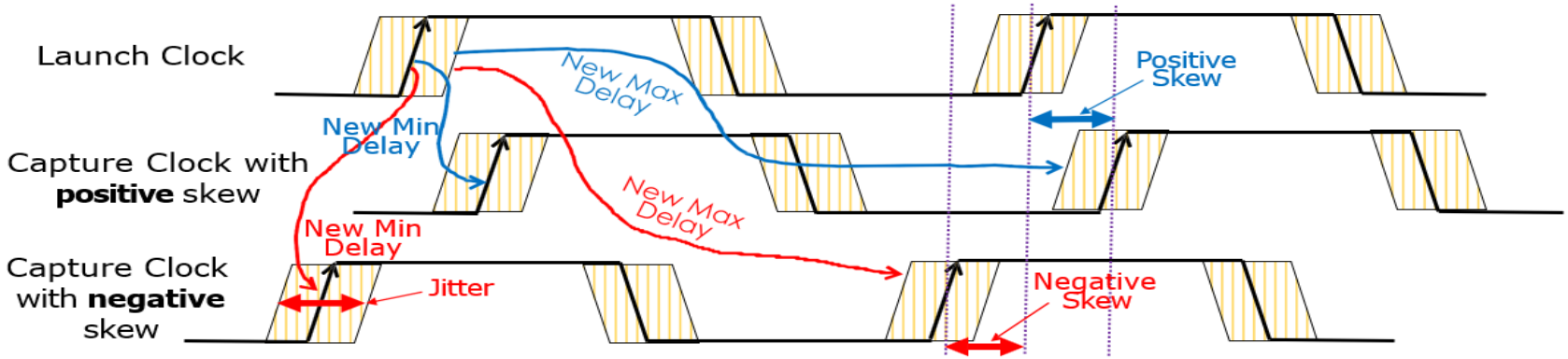




IMPLICATIONS ON TIMING

- **Let's remember the timing constraints:**
  - Max Delay: $T + \delta_{skew} - 2\delta_{jitter} > t_{CQ} + t_{logic} + t_{setup} + \delta_{margin}$
  - Min Delay: $t_{CQ} + t_{logic} - \delta_{margin} > t_{hold} + \delta_{skew}$





IMPLICATIONS ON POWER

- Let's remember how to calculate dynamic power:

$$P_{dyn} = \alpha_{0 \rightarrow 1} C_L V_{DD}^2 f$$

- The activity factor (α) of the clock network is 100%!
- The clock capacitance consists of:
  - Clock generation (i.e., PLL, clock dividers, etc.)
  - Clock elements (buffers, muxes, clock gates)
  - Clock wires
  - Clock load of sequential elements
- Clock networks are huge
  - And therefore, the clock is responsible for a large percentage of the total chip power.

IMPLICATIONS ON SIGNAL INTEGRITY

- Signal Integrity is an obvious requirement for the clock network:
- Noise on the clock network can cause:
  - In the worst case, additional clock edges
  - Lower coupling can still slow down or speed up clock propagation
  - Irregular clock edges can impede register operation

IMPLICATIONS ON SIGNAL INTEGRITY

- Signal Integrity is an obvious requirement for the clock network:
- Noise on the clock network can cause:
  - In the worst case, additional clock edges
  - Lower coupling can still slow down or speed up clock propagation
  - Irregular clock edges can impede register operation
- Slow clock transitions (slew rate):
  - Susceptibility to noise (weak driver)
  - Poor register functionality (worse t_{cq}, t_{setup}, t_{hold})
- Too fast clock transitions
  - Overdesign → power, area, etc.
  - Bigger aggressor to other signals
- Unbalanced drivers lead to increased skew.

Best practice:

Keep t_{rise} and t_{fall} between 10-20%of clock period (e.g., 100-200ps @1GHz)

IMPLICATIONS ON AREA

- To reiterate, clock networks consist of:
  - Clock generators
  - Clock elements
  - Clock wires
- All of these consume area
  - Clock generators (e.g., PLL) can be very large
  - Clock buffers are distributed all over the place
  - Clock wires consume a lot of routing resources

Source: Universitat Bonn

IMPLICATIONS ON AREA

- Routing resources are most vital
  - Require low RC (for transition and power)
    - Benefit of using high, wide metals
  - Need to connect to every clock element (FF)
    - Distribution all over the chip
    - Need Via stack to go down from high metals

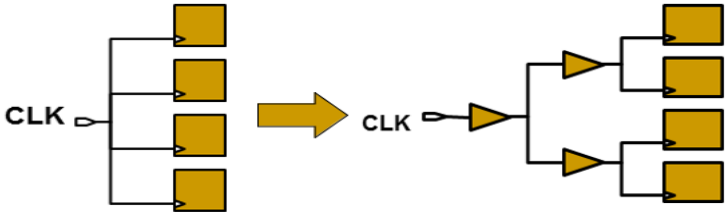
For example: **Intel Itanium** 4% of M4/M5 used for clock routing

Clock Distribution

THE CLOCK ROUTING PROBLEM

Given a source and n sinks:

- Connect all sinks to the source by an interconnect network so as to minimize:
  - Clock Skew = $\max_{i,j} |t_i - t_j|$
  - Delay = $\max_i (t_i)$
  - Total wirelength
  - Noise and coupling effect



Clock Tree goals:
1) Minimize Skew
2) Meet target insertion delay (min/max)

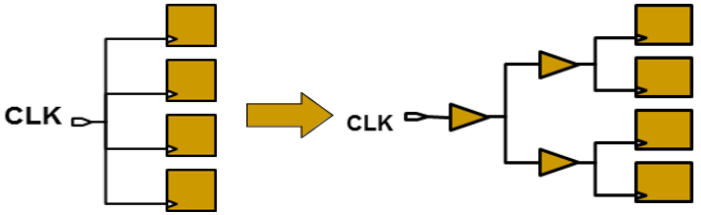
THE CLOCK ROUTING PROBLEM

Given a source and n sinks:

- Connect all sinks to the source by an interconnect network so as to minimize:
  - Clock Skew = $\max_{i,j} |t_i - t_j|$
  - Delay = $\max_i (t_i)$
  - Total wirelength
  - Noise and coupling effect

The Challenge:

- Synchronize millions (billions) of separate elements
  - Within a time scale on order of ~10 ps
  - At distances spanning 2-4 cm
  - Ratio of synchronizing distance to element size on order of 10^5
  - Reference: light travels <1 cm in 10 ps



Clock Tree goals:
1) Minimize Skew
2) Meet target insertion delay (min/max)

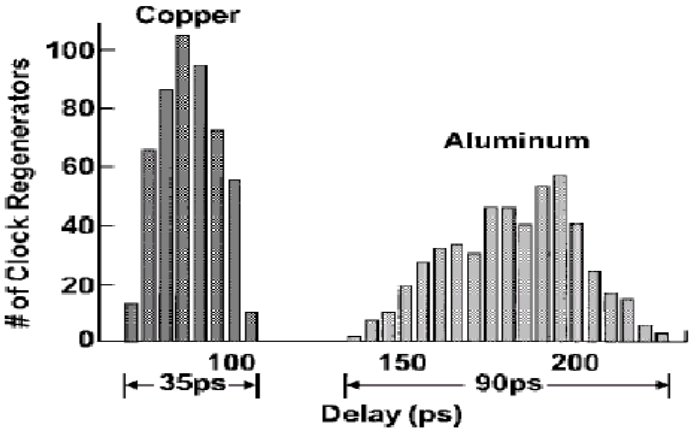
Clock Tree constraints:
1) Maximum Transition
2) Maximum load cap
3) Maximum Fanout
4) Maximum Buffer Levels

TECHNOLOGY TRENDS

- Timing
  - Higher clock frequency ⟶ Lower skew
  - Higher clock frequency ⟶ Faster transitions
  - Jitter – PLL's get better with CMOS scaling
  - But other sources of noise increase
    - Power supply noise more important
    - Switching-dependent temperature gradients

TECHNOLOGY TRENDS

- Timing
  - Higher clock frequency ⟶ Lower skew
  - Higher clock frequency ⟶ Faster transitions
  - Jitter – PLL's get better with CMOS scaling
  - But other sources of noise increase
    - Power supply noise more important
    - Switching-dependent temperature gradients



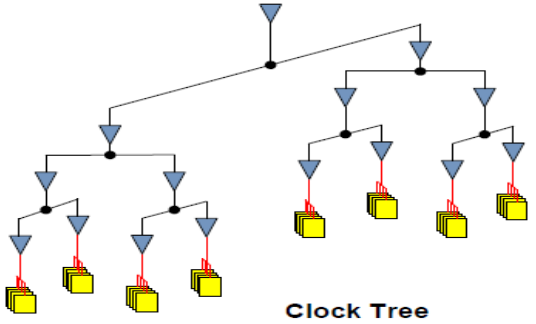
- New Interconnect Materials
  - Copper Interconnect ⟶ Lower RC ⟶ Better slew and potential skew
  - Low-k dielectrics ⟶ Lower clock power, better latency/skew/slew rates

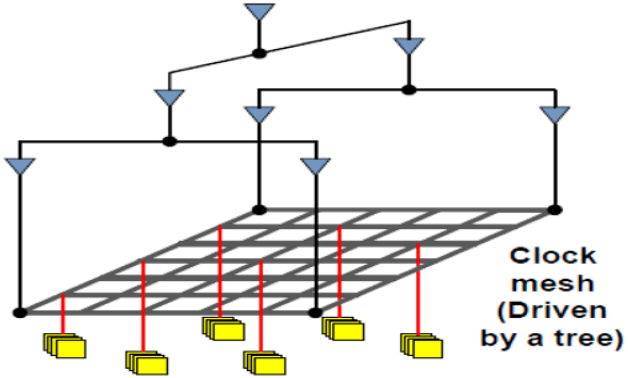
TECHNOLOGY TRENDS

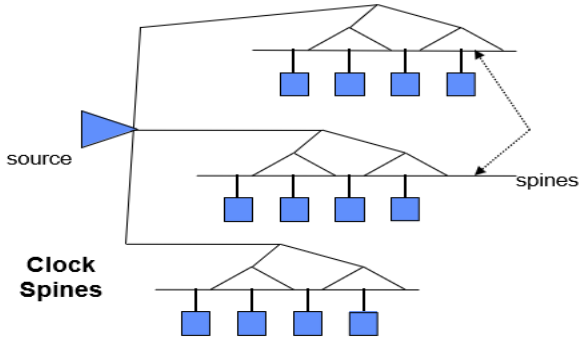
- Power
  - Heavily pipelined design → more registers → more capacitive load for clock
  - Larger chips → more wire-length needed to cover the entire die
  - Complexity → more functionality and devices → more clocked elements

APPROACHES TO CLOCK SYNTHESIS

- Broad classification:
  - Clock Tree
  - Clock Mesh (Grid)
  - Clock Spines

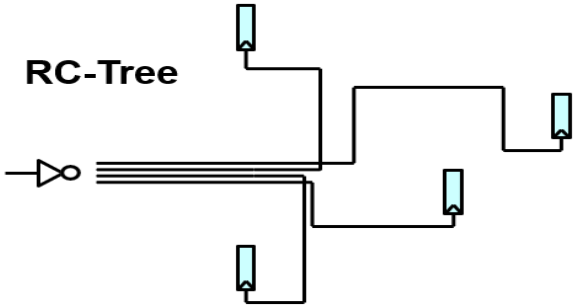






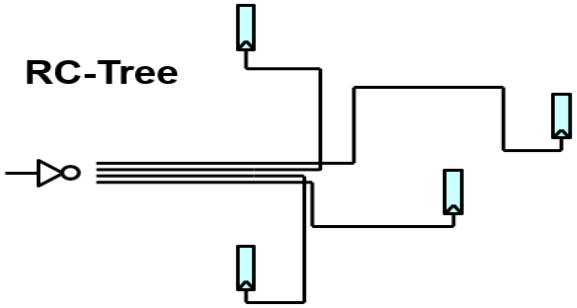
CLOCK TREES

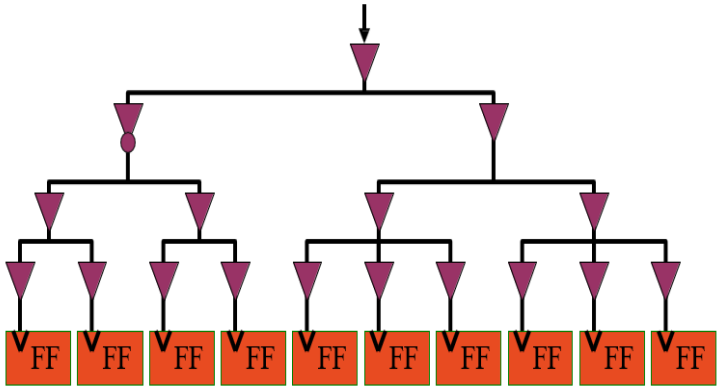
- Naïve approach:
  - Route an individual clock net to each sink and balance the RC-delay
  - However, this would burn excessive power and the large RC of each net would cause signal integrity issues.



CLOCK TREES

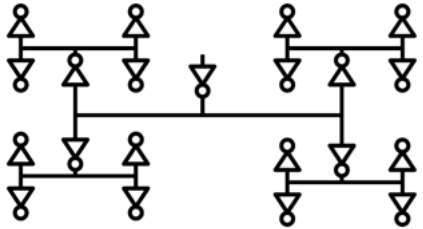
- Naïve approach:
  - Route an individual clock net to each sink and balance the RC-delay
  - However, this would burn excessive power and the large RC of each net would cause signal integrity issues.
- Instead use a buffered tree
  - Short nets mean lower RC values
  - Buffers restore the signal for better slew rates
  - Lower total insertion delay
  - Less total switching capacitance



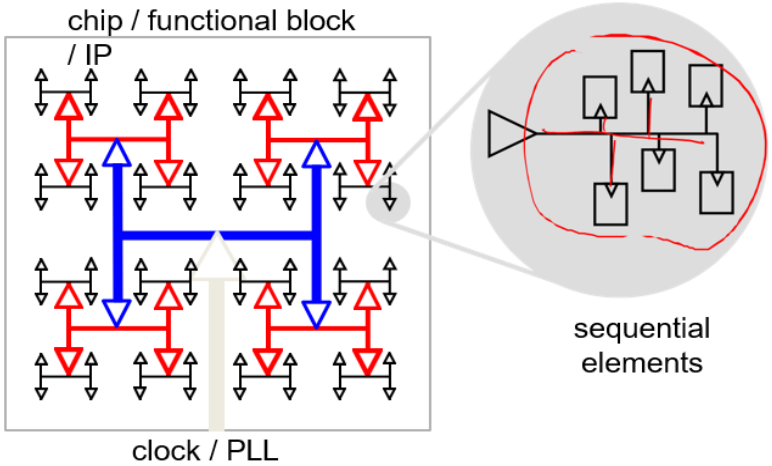


BUILDING AN ACTUAL CLOCK TREE

- Perfectly balanced approach: H-Tree
  - One large central driver
  - Recursive H-style structure to match wire-lengths
  - Halve wire width at branching points to reduce reflections

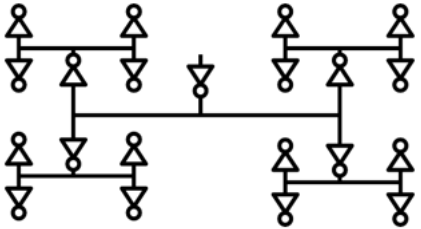


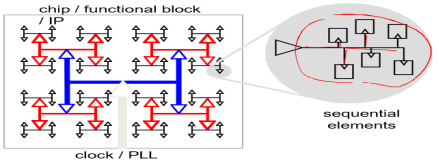
- More realistic:
  - Tapered H-Tree, but still hard to do.

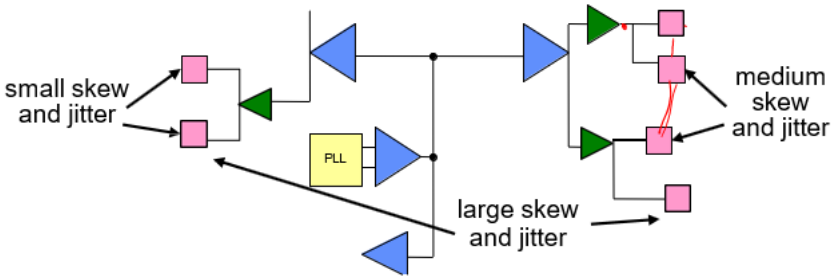


BUILDING AN ACTUAL CLOCK TREE

- Perfectly balanced approach: H-Tree
  - One large central driver
  - Recursive H-style structure to match wire-lengths
  - Halve wire width at branching points to reduce reflections
- More realistic:
  - Tapered H-Tree, but still hard to do.
- Standard CTS approach:
  - Flip flops aren't distributed evenly.
  - Try to build a balanced tree

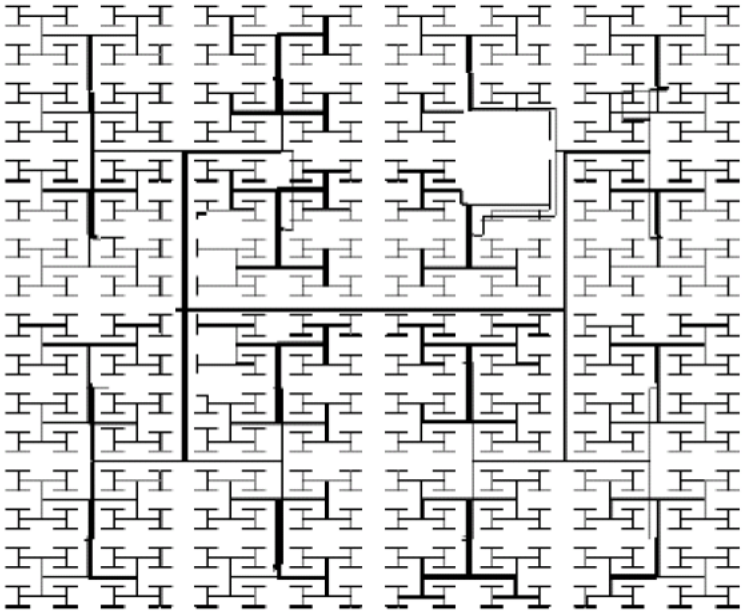






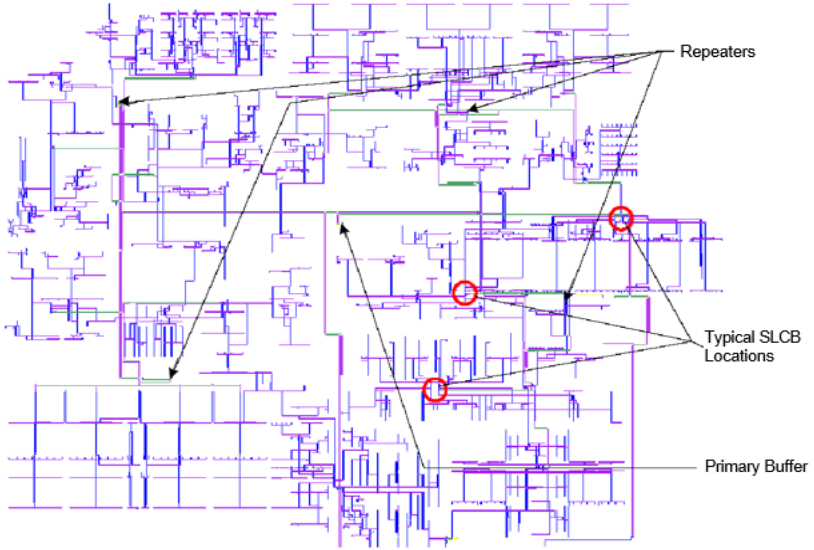
INDUSTRIAL H-TREE EXAMPLES

- IBM PowerPC (2002)



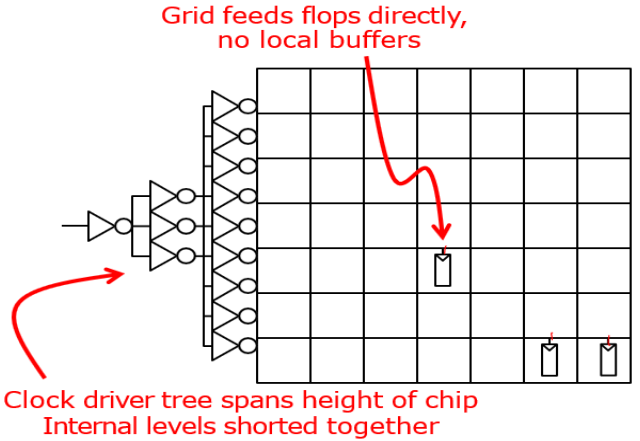
IBM, ISSCC 2000

- Intel Itanium 2 (2005)



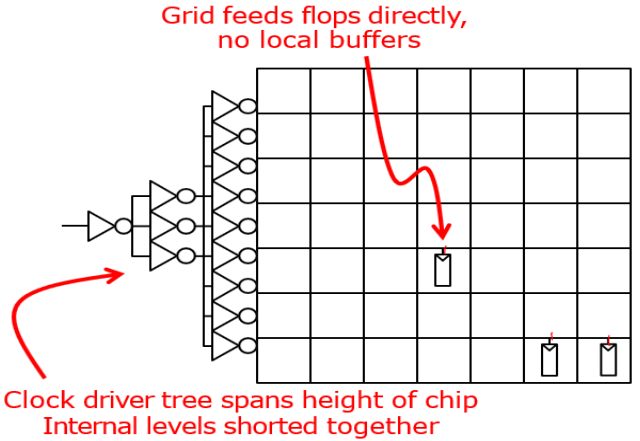
LOWER SKEW – CLOCK GRIDS

- Advantages:
  - Skew determined by grid density and not overly sensitive to load position
  - Clock signals are available everywhere
  - Tolerant to process variations
  - Usually yields extremely low skew values



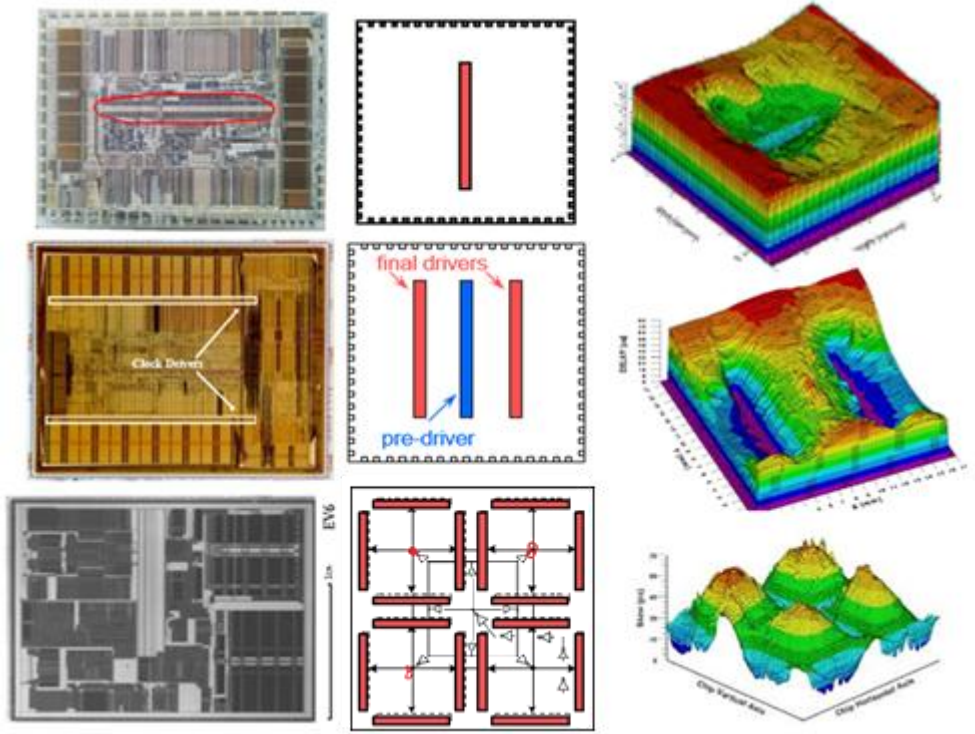
LOWER SKEW – CLOCK GRIDS

- Disadvantages
  - Huge amounts of wiring & power
    - Wire cap large
    - Strong drivers needed – pre-driver cap large
    - Routing area large
  - To minimize all these penalties, make grid pitch coarser
    - Skew gets worse
    - Losing the main advantage
  - Don't overdesign – let the skew be as large as tolerable
  - Still – grids seem non-feasible for SoC's



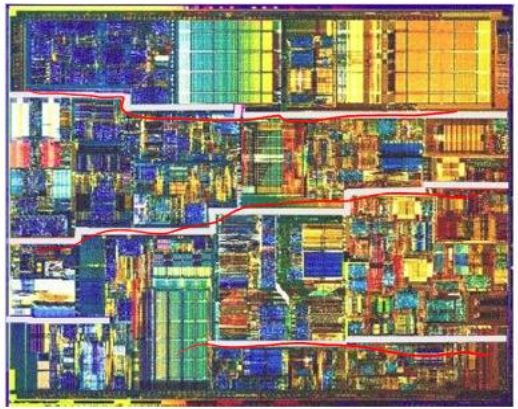
DEC ALPHA – GENERATIONS OF GRIDS

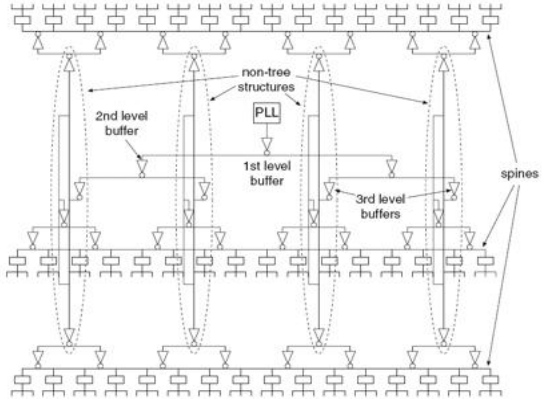
- 21064 (EV4) – 1992
  - 0.75µm, 200MHz, 1.7M trans.
  - Big central driver, clock grid
    - 240ps skew
- 21164 (EV5) - 1995
  - 0.5 µm, 300MHz, 9.3M trans.
  - Central driver, two final drivers, clock grid
    - Total driver size – 58 cm!
    - 120ps skew
- 21264 (EV6) - 1998
  - 0.35 µm, 600MHz, 15.2M trans.
  - 4 skew areas for gating
    - Total driver size: 40 cm
    - 75ps skew



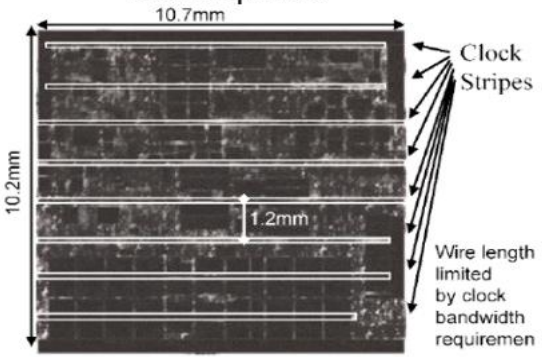
CLOCK SPINES

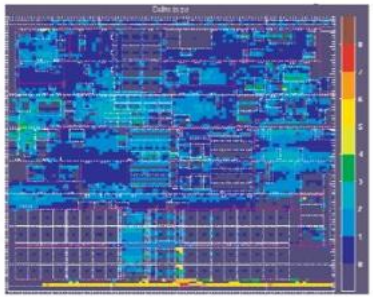
- Clock grids are too power (and routing) hungry.
- A different approach is to use spines
  - Build an H-Tree to each spine
  - Radiate local clock distribution from spines
- Pentium 4 (2001) used the clock spine approach.





Later Pentium 4's used more spines

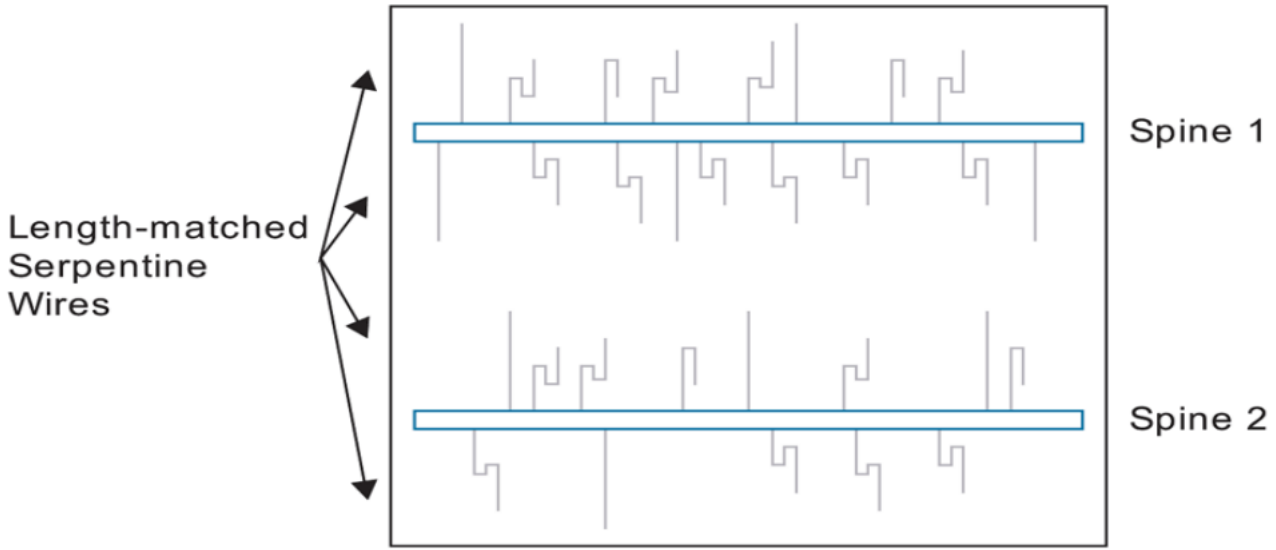




Source: Bindal ISSCC 2003

CLOCK SPINES

- As with a grid, clock buffers located in rows
- Spines drive length-matched serpentine wires to each small group of clocked elements



SUMMARY OF MAIN CLOCK DIST. APPROACHES

- H-tree
  - Low skew, smallest routing capacitance, low power
  - Floorplan flexibility is poor.
- Grid or mesh
  - Low skew, increases routing capacitance, worse power
  - Alpha uses global clock grid and regional clock grids
- Spine
  - Small RC delay because of large spine width
  - Spine has to balance delays; difficult problem
  - Routing cap lower than grid but may be higher than H-tree.

Structure	Skew	Cap/area/ power	Floorplan Flexibility
H-Tree	Low/Med	Low	Low
Grid	Low	High	High
Spine	High	Medium	Medium

CLOCK CONCURRENT OPTIMIZATION (CCOPT)

- What is the main goal of CTS?
  - Attempt to minimize skew
  - The tree that results tends to require a lot of power and area
  - Question: Is minimal skew our actual goal?
- Idea behind CCOpt: Forget about skew and focus on real goals
  - Must meet timing and design rule violation (DRV) constraints
  - Consider system timing constraints while building the clock tree

CLOCK CONCURRENT OPTIMIZATION (CCOPT)

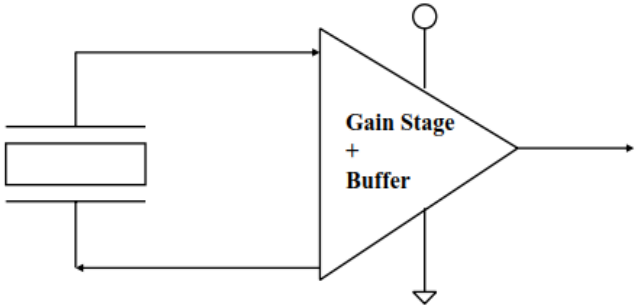
- CCOpt Methodology:
  - First, build a clock tree in order to fix DRVs
  - Then, check timing (setup and hold) and fix any remaining violations
- What makes CCOpt a good approach?
  - Most timing paths are local
  - Thus, they likely come from same local clock branch and don't need a lot of skew balancing to start with – locally, the skew probably isn't bad
  - We also likely don't need low skew for timing paths across chip
- Less skew balancing leads to:
  - Lower insertion delay (helps power, jitter)
  - Fewer clock buffers (helps power, area)

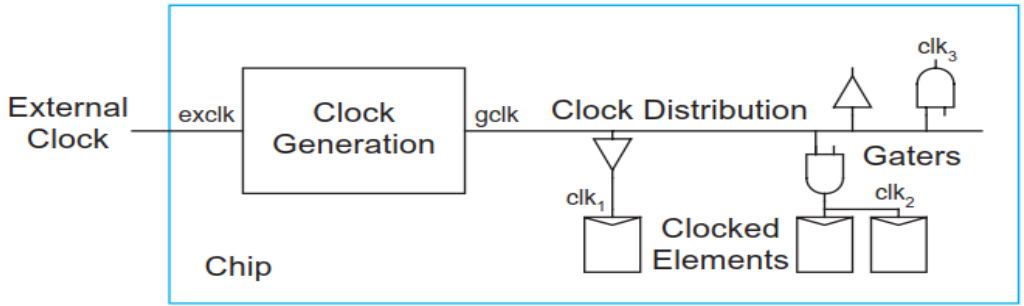
Additional Topics

CLOCK GENERATION

Where does the clock come from?

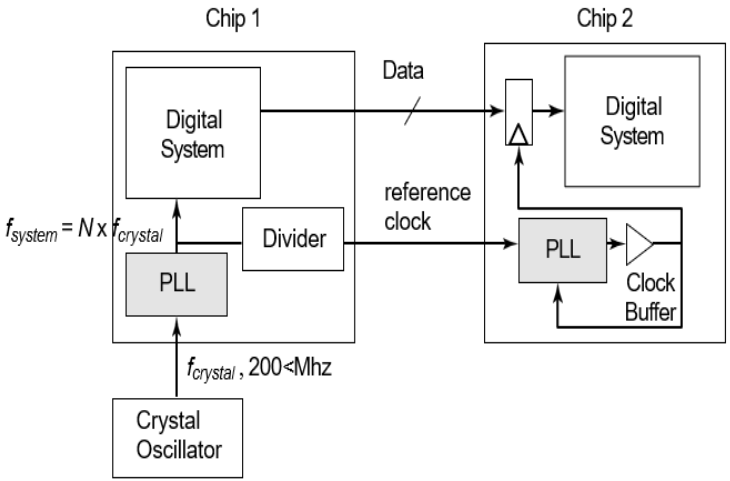
- The easiest way to generate a clock is by using a ring oscillator or some other non-stable circuit, but these are susceptible to PVT variations.
- Therefore, clocks are generally generated off-chip using a crystal and an oscillation circuit.
- However, usually only one off-chip clock can be used (a single frequency) and the frequency that can be input to the chip is limited to around 100 MHz.
- Therefore, on-chip local clock generation is employed, usually with a PLL or DLL.





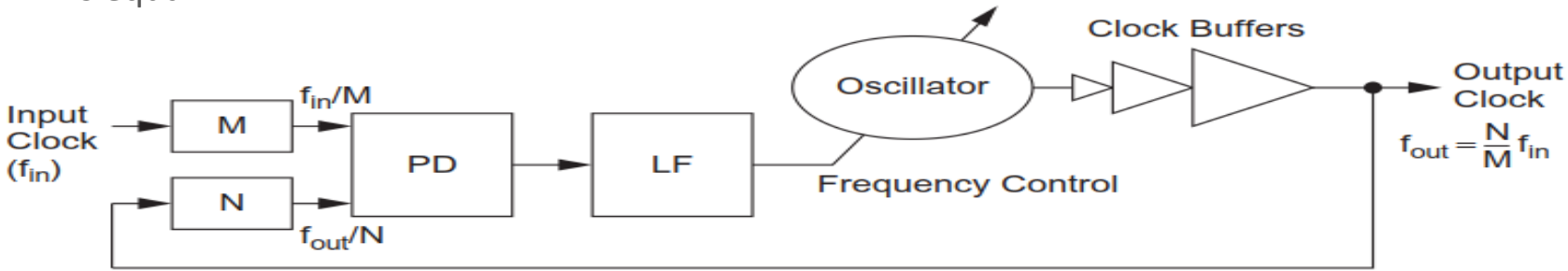
LOCAL CLOCK GENERATION

- Externally generated clocks suffer from two primary problems:
  - Frequency is limited, i.e., a clock multiplier is needed.
  - Clock phase is uncontrolled, such that communication with the external clock domain is unsynchronized.
- To solve both of these problems, a Phase-Locked Loop (PLL) is used.
  - If clock multiplication is not required, a delay-locked loop (DLL) is a more simple solution.



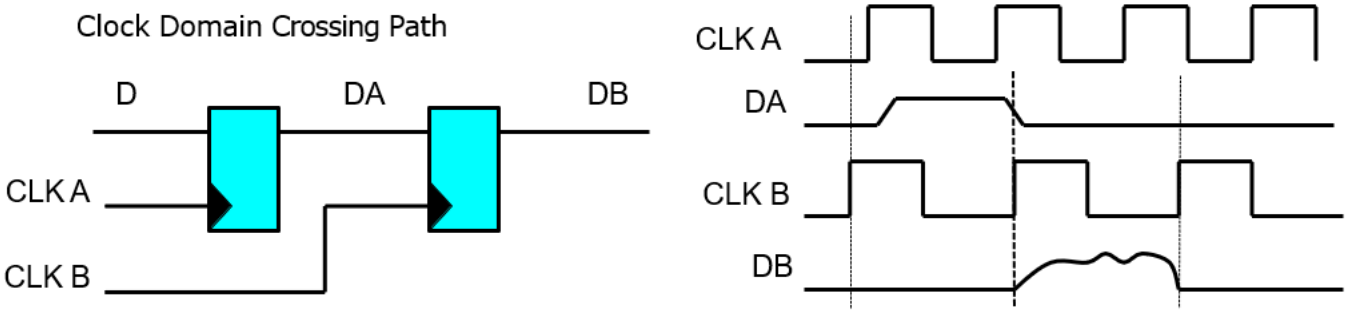
LOCAL CLOCK GENERATION

- How does a PLL work?
  - A phase detector (PD) produces a signal proportional to the phase difference between the input and output clocks.
  - A loop filter (LF) converts the phase error into a control signal (voltage).
  - A voltage-controlled oscillator (VCO), creates a new clock signal based on the error signal.
- What about a DLL?
  - Same principle, but instead of changing the frequency, it just delays the clock until the phase is equal.



CLOCK DOMAIN CROSSING (CDC)

- Most system-on-chips have several components that communicate with different external interfaces and run on different clock frequencies.
  - In other words, they have multiple asynchronous clock domains.
- Asynchronous clocks cannot communicate with each other in a straightforward fashion:



PROBLEMS WITH CDC

The main problems with passing data between asynchronous domains are:

- Metastability:
  - A setup/hold violation in the capture register.
  - May cause:
    - High propagation delay at the fanout.
    - High current flow in the chip (even burnout).
    - Different values of the signal at different parts of the fanout.

What is the probability of metastability?

Let us define:

T_w – an error window (setup+hold) around the clock cycle

f – clock frequency

f_D – Frequency of data change

Now assume that a data (D) change can come anywhere in the clock cycle relative to the capture clock, so:

$$Rate(\text{meta}) = f \cdot f_D \cdot T_w$$

Assume:

$$f = 1\text{GHz} \quad f_D = 0.1f \quad T_w = 20\text{ps}$$

$$Rate(\text{meta}) = 2 \cdot 10^6 / \text{sec}$$

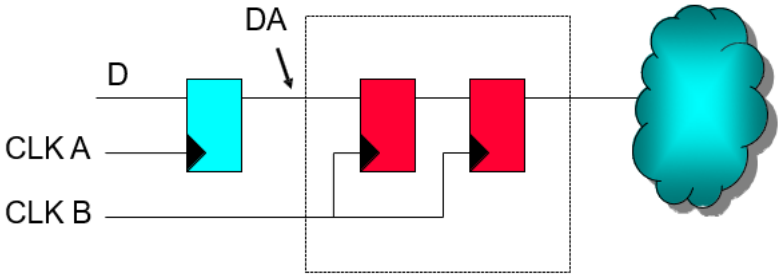
PROBLEMS WITH CDC

The main problems with passing data between asynchronous domains are:

- Data Loss:
  - New data in the source may be generated w/o the data being captured by the destination.
- Data Incoherency:
  - Data may be captured late, causing several coherent signals to be in different states.

SOLUTIONS: SYNCHRONIZERS

- By cascading two or more flip flops, we create a simple synchronizer.
  - The signal has one (or more) clock cycles to stabilize.
  - However, there is a probability that the signal will not settle within the cycle time (T).



What is the probability of failure?

The probability for metastability to pass is:

$P(t > S) = e^{-S/\tau}$ with τ a parameter of the flip flop

We want the metastability to dissipate at least t_{setup} before the next clock edge, so:

$$P(\text{failure}) = P(\text{meta}) \cdot P(\text{exit}) = \frac{T_w}{T} \cdot e^{-\frac{T - t_{setup}}{\tau}}$$

And the mean time between failures (MTBF) is the inverse of the failure rate:

$$MTBF = \frac{1}{Rate(\text{failure})} = \frac{1}{f \cdot f_D \cdot T_w} e^{-\frac{T - t_{setup}}{\tau}}$$

For our previous example, this is about 10^{24} years…

ARE SYNCHRONIZERS ENOUGH?

- No!
  - We may have taken care of metastability, but data loss and data incoherence are still there.
  - So we need to design our logic accordingly.
- To eliminate data loss:
  - Slow to fast clock – we won't lose any data.
  - Fast to slow clock – hold source data for several cycles.
- But for data coherence, we need more thinking:
  - Handshake protocols.
  - First-in First-out (FIFO) interfaces
  - Other solutions (Gray code, Multiplexers, etc.)

Thank you!