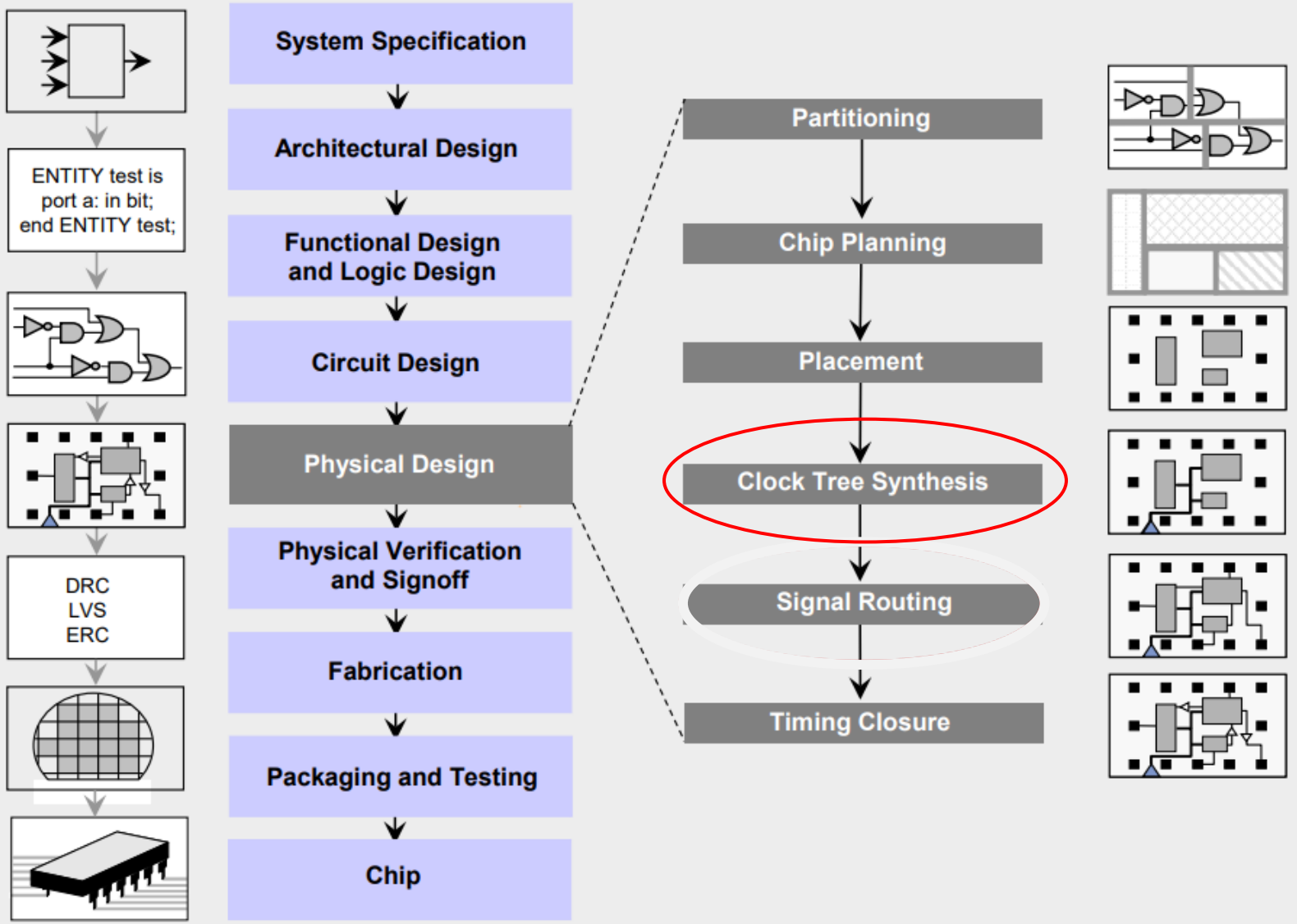
Clock Tree Synthesis

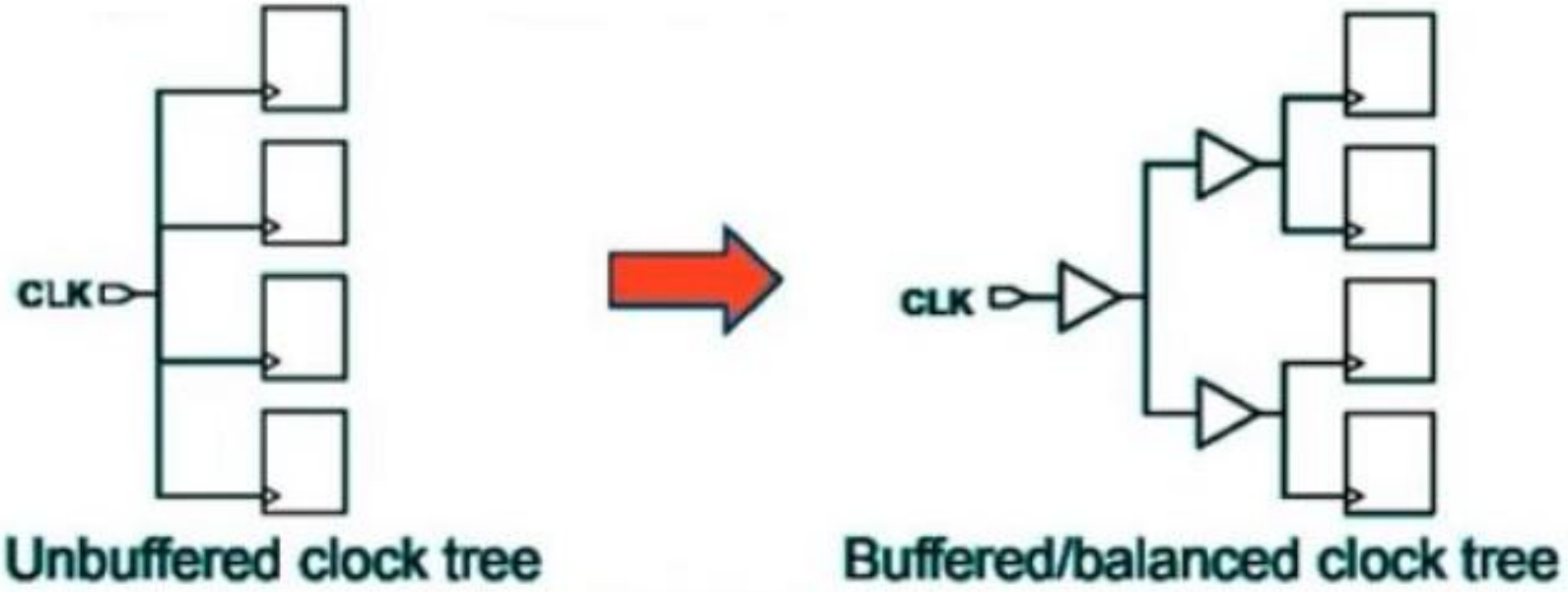
https://vlsi-backend-adventure.com/cts.html#1

5.1 Introduction



Clock Tree Synthesis (CTS)

- Clock Tree Synthesis (CTS) is one of the most important stages in PnR. CTS QoR decides timing convergence & power. In most of the ICs clock consumes 30-40 % of total power. So efficient clock architecture, clock gating & clock tree implementation helps to reduce power.
- The process of distributing the clock and balancing the load is called CTS. Basically, delivering the clock to all sequential elements. CTS is the process of insertion of buffers or inverters along the clock paths of ASIC design in order to achieve zero/minimum skew or balanced skew. Before CTS, all clock pins are driven by a single clock source. CTS starting point is clock source and CTS ending point is clock pins of sequential cells.



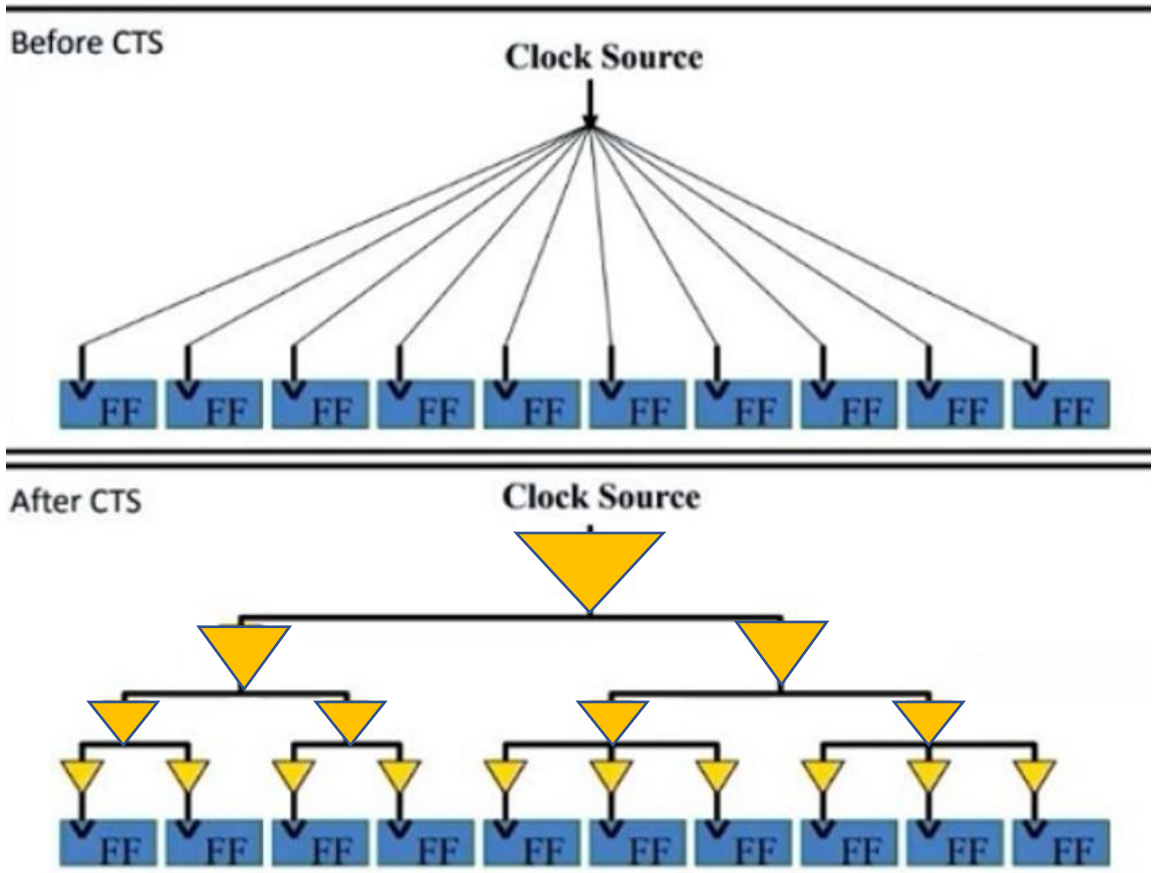
Clock Tree Synthesis Is Challenging!

Clock tree consumes up to 40% power
→ aggressive power reduction
→ complex clock tree with clock logic cells (CLCs) such as, clock gating, divider, MUXes

Complex timing constraints across process, voltage, temperature and operating scenarios

On-chip variation → more design margin

- Clock Tree is a path from the Clock Source (Root) to Clock Sinks (Leaf)
- Clock Tree Synthesis is the process of creating this Clock Path from Clock Source to Clock Sinks
- All Clock pins of flip Flop are considered as Clock Sinks (Leaf); where the Clock Tree Synthesis ends

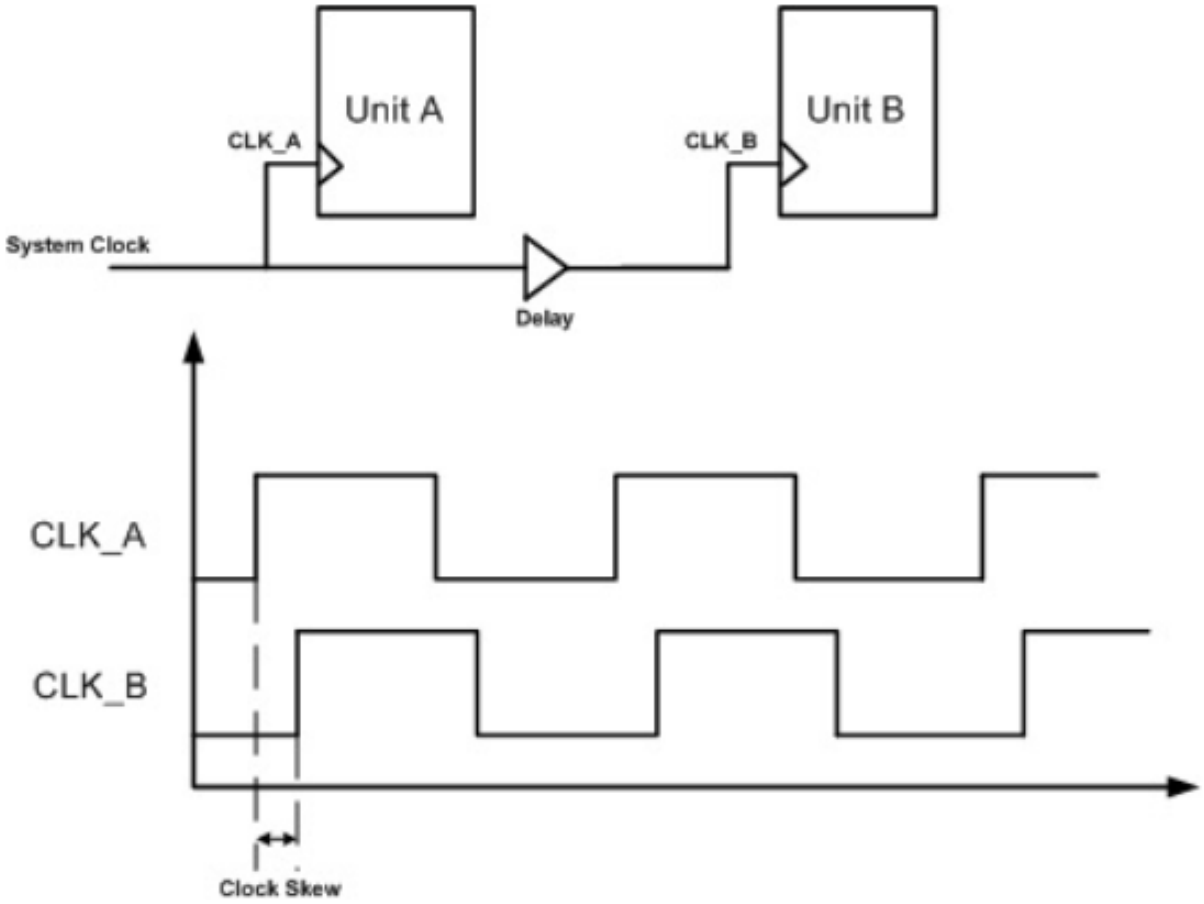


- How many clock buffer levels?
- What kind of clock buffers?
- At the bottom level, how many FFs can be driven by one clock buffer?
- For higher level, buffers will drive buffers. How many of them can form a group?
- How do we group those FFs?
  1) Usually after placement
  2) What buffers to be used?
- Where to place those buffers?
  1) Usually after placement
  2) Now need to place the newly inserted clock buffers
- How to route them?
- …

2. clock skew

The Clock Problem

- Clock skew
- Long clock insertion delay
- Skew across clocks
- Heavy clock net loading
- Clock is power hungry
- Clock to signal coupling effect (CrossTalk)
- Electromigration on clock net



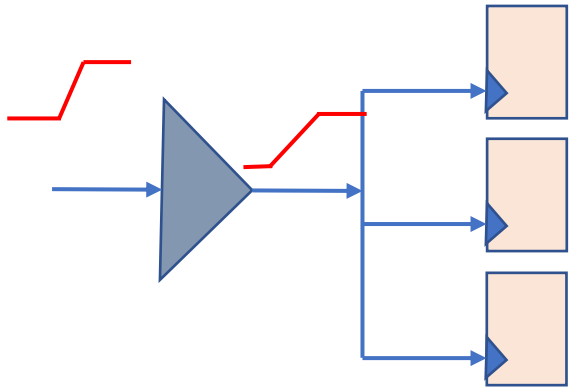
Constraints To Follow In CTS

Clocks tree constraints:

https://teamvlsi.com/2021/06/clock-tree-constraints-in-vlsi-ccopt-file-in-physical-design-cts-constraints.html

1) **Target Skew**
2) **Target Maximum Transition**
3) **Target Maximum Capacitance**
4) **Min/Max Insertion Delay (Latency)**
5) **Clock tree cell list**
6) **Maximum Fanout**
7) **Preferred clock tree routing layers and Non-Default Rules (NDR)**
8) **Cell Density**
9) **Cell Halo**

- ❑ Need to provide a halo around the clock tree instance to avoid a denser placement which may cause IR / Crosstalk issue. So provide halo constraints in the x and y direction over the cells.



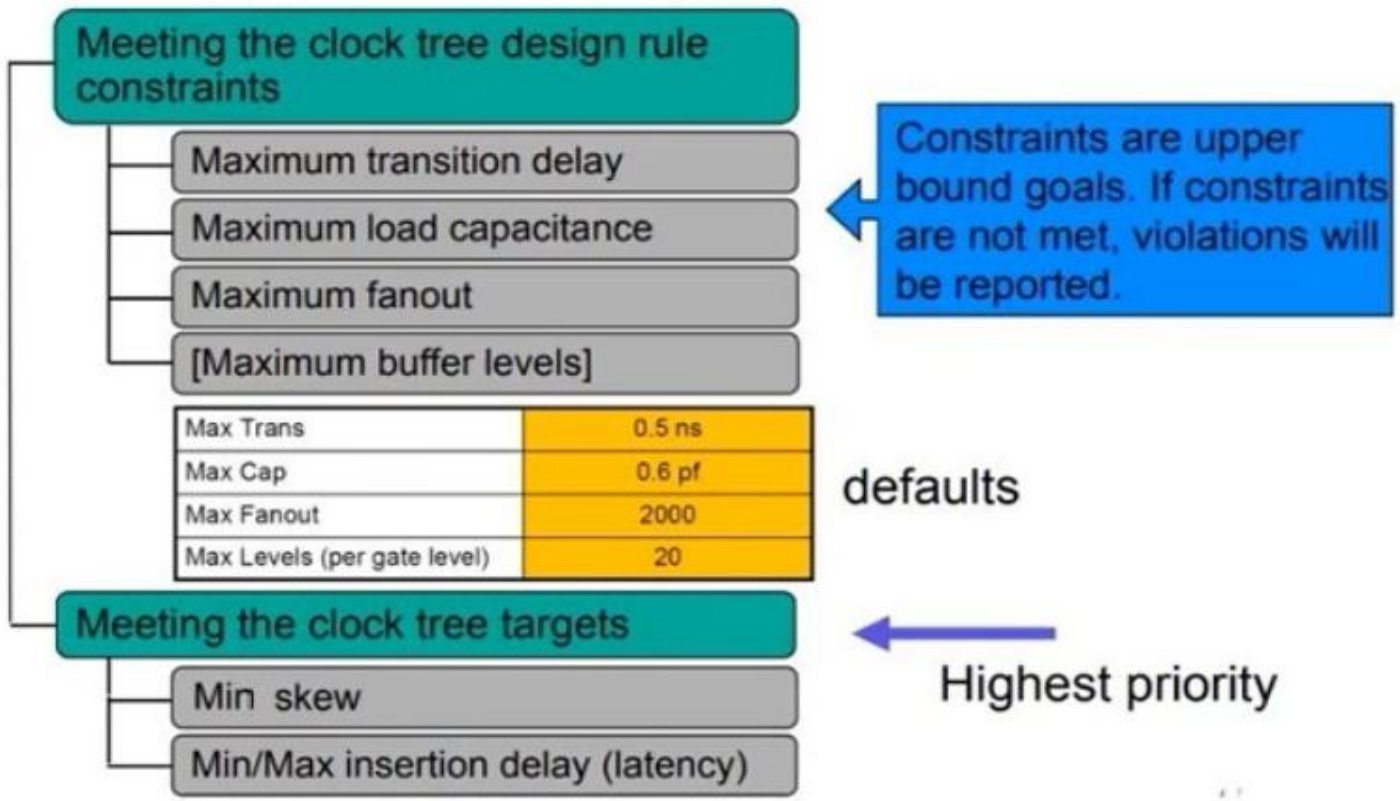
- Need timing library for cells
- Given a clock buffer, assuming its input transition delay, see how much total cap it can drive and meets output max transition delay.
- Then, check the input cap of FFs, and estimated wire length and its RC

→ This buffer can drive how many FF clk pins

Inputs of CTS

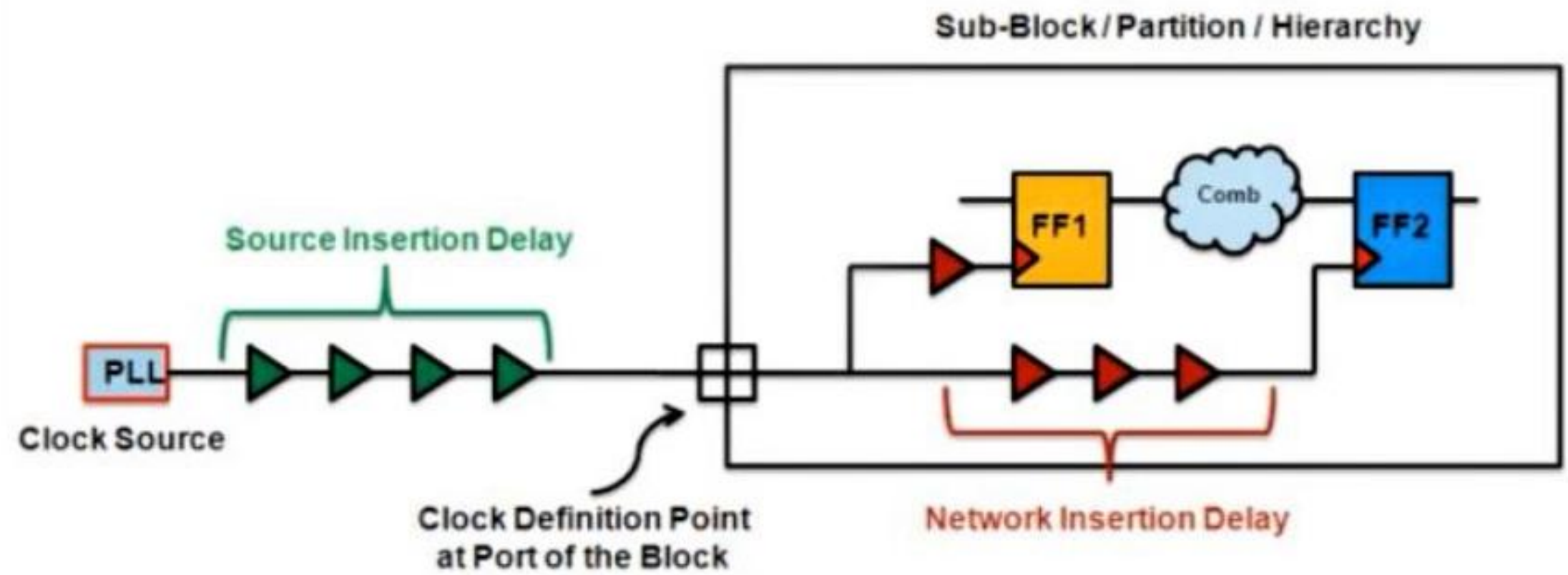
- Technology file (.tf)
- Netlist
- SDC
- Library files (.lib & .lef) & TLU+ file
- Placement DEF file
- Clock specification file which contains Insertion delay, skew, clock transition, clock cells, NDR, CTS tree type, CTS exceptions, list of buffers/inverters etc...

Goals of CTS



Clock Latency / Insertion Delay

Clock Latency

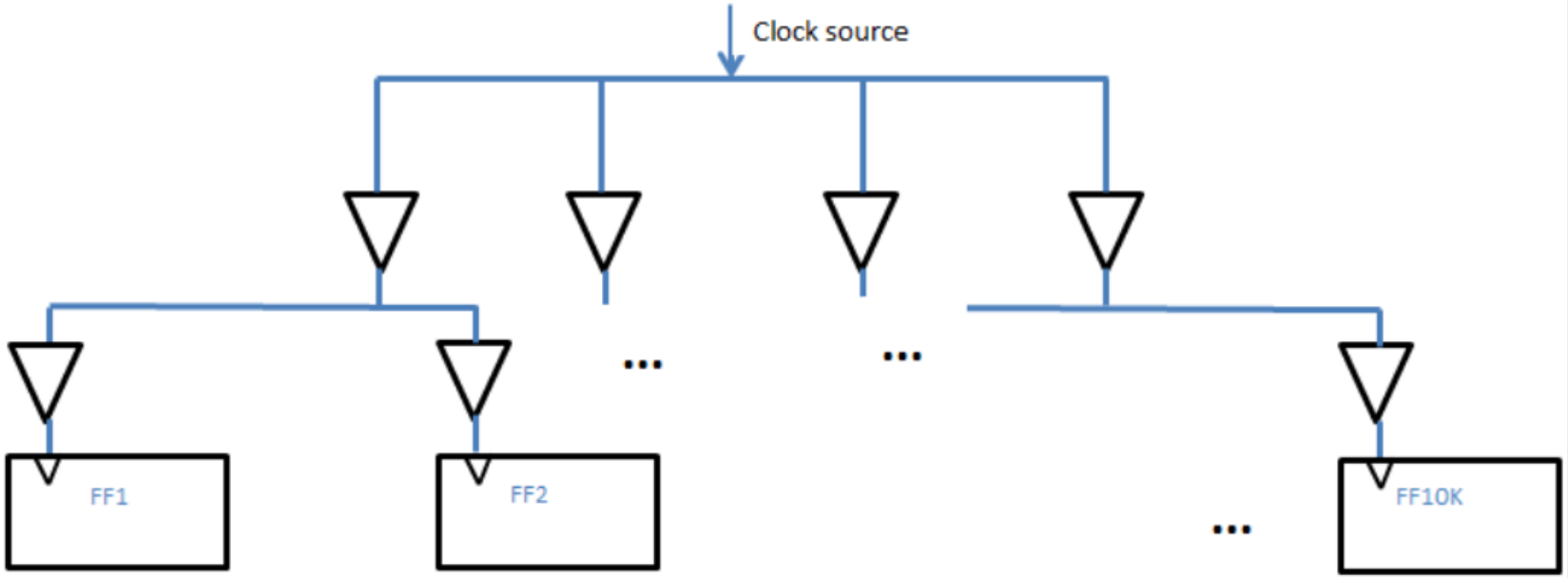


Total Latency = Source Insertion Delay + Network Insertion Delay

- Total time taken by the clock signal to reach the input of the register
- Source latency is the time between clock sources to clock definition ports
- Network latency is the time between clock definition ports to clock leaf cells in the design

After Trees Are Built → Routing

Please think about how to synthesize a clock tree



Clock Net After CTS.

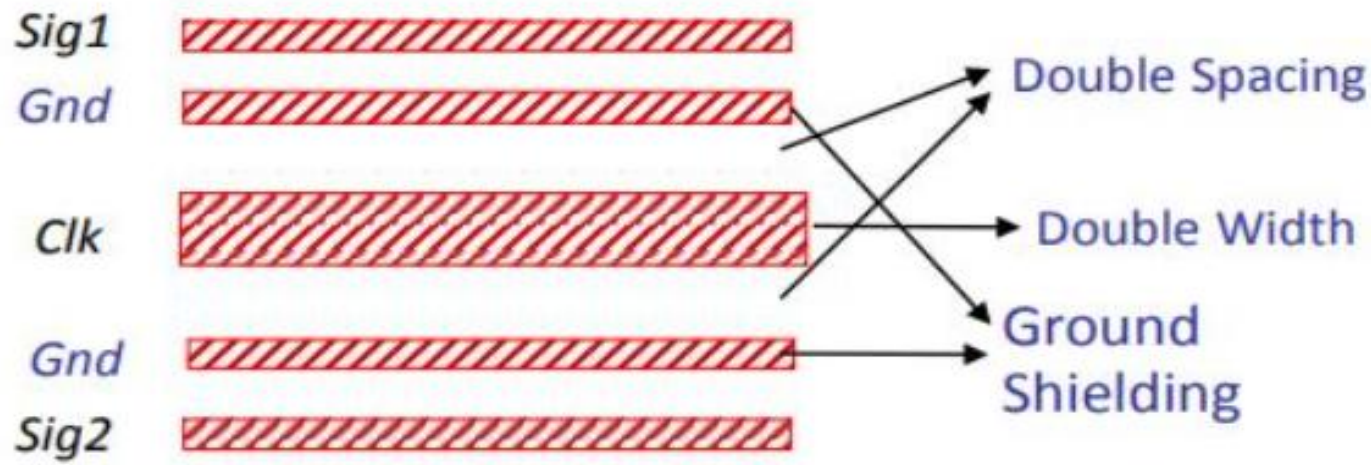
Clock tree routing usually is being done before regular signal nets, since they are more critical

Non-Default Rule (NDR)

- The user-defined Routing rules apart from the default Routing Rule
- Often used to “harden” the sensitive nets like Clock Nets
- NDRs make the Clock Routes less sensitive to CrossTalk or EM effects
- Double/ Triple Width for avoiding Electromigration
- Double/ Triple Spacing for avoiding Crosstalk
- NDRs will improve Insertion Delay

Default Routing Rule

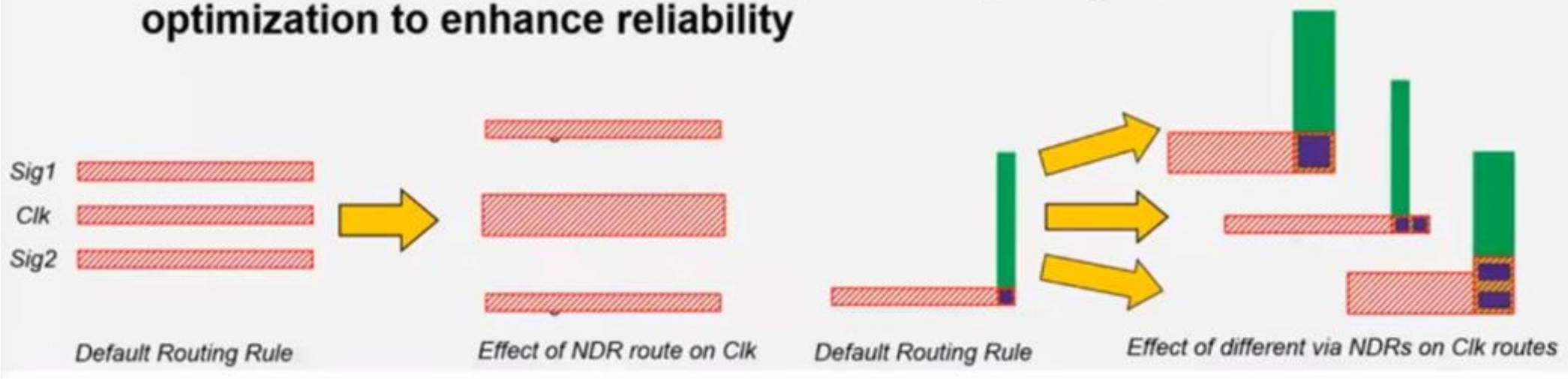
NDR Route on Clock net



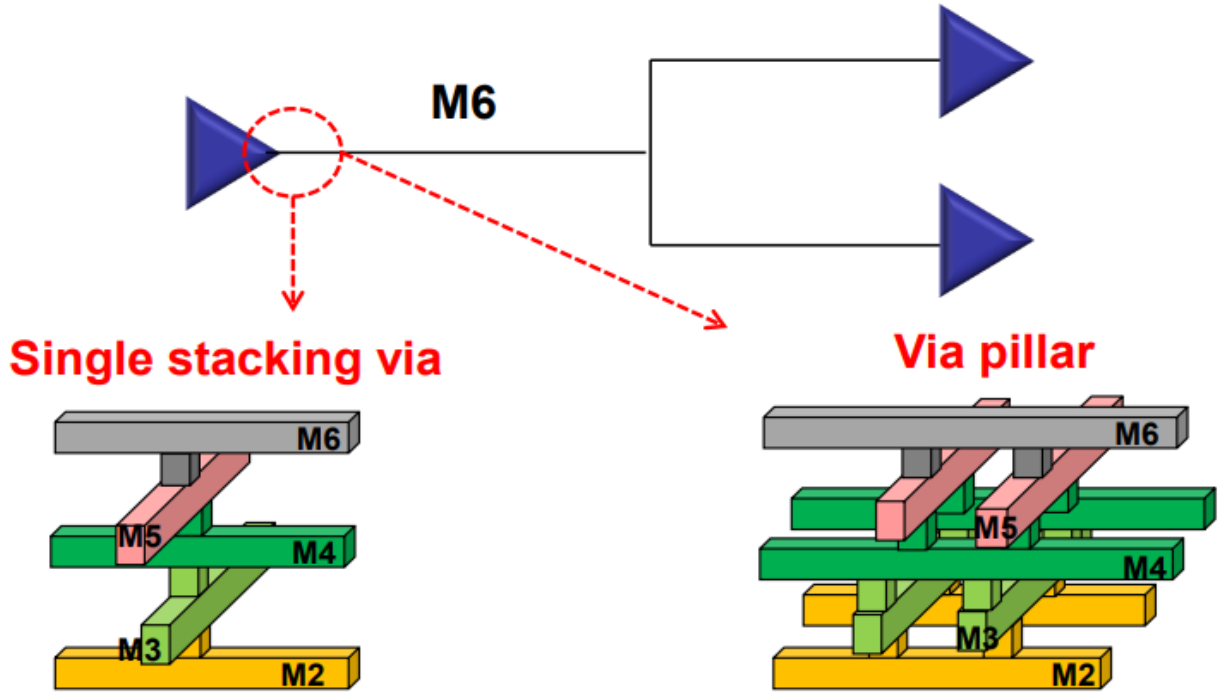
https://www.programmersought.com/article/86877197473/

Non-Default Rules

- *ICC II* can route the clock nets using *non-default routing* (NDR) rules, e.g. double-spacing, double-width, shielding
- NDR rules are often used to "harden" the clock, e.g. to make the clock routes less sensitive to *cross-talk* and *electro-migration* (*EM*) effects
- Clock nets are hardened further by often requiring 100% via optimization to enhance reliability



VIA Pillar



- Large drivers + upper thick metal + via pillar to reduce transistor, metal and via resistance
- VIA-Pillar enablement is complex and requires EDA enablement across all Implementation phases

For Long Interconnect – Why Adding Buffer

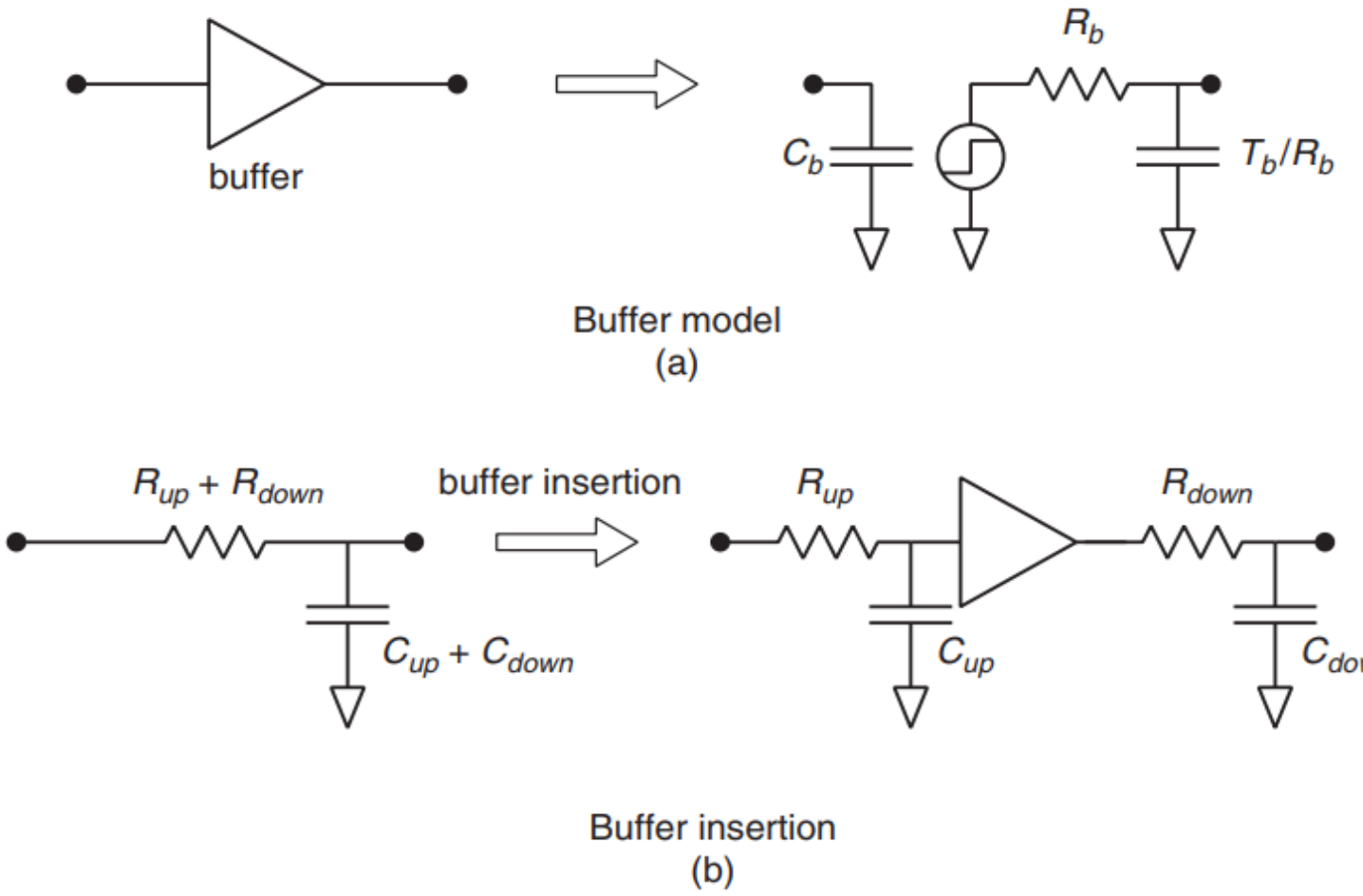
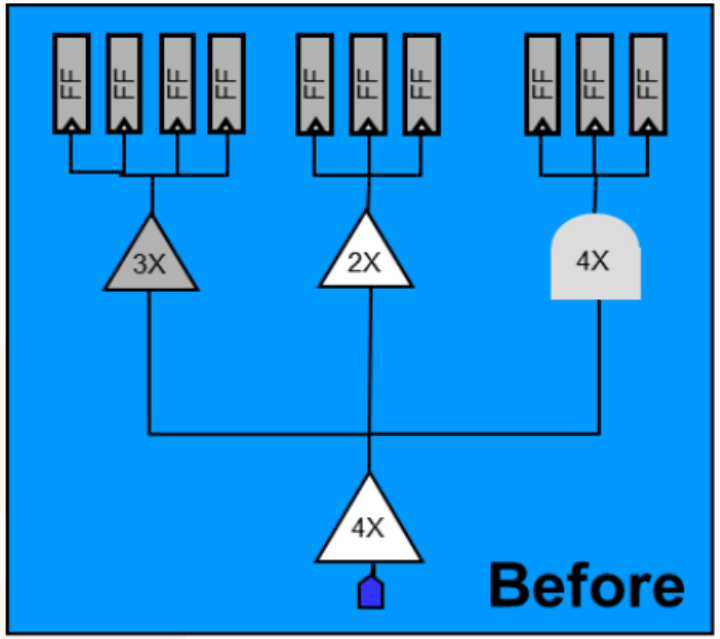


FIGURE 13.44

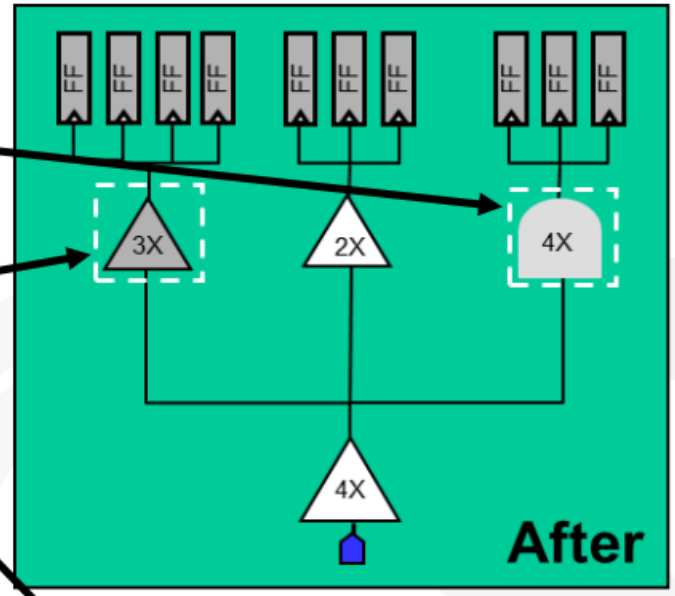
(a) A switch-level model for a clock buffer/driver. (b) The insertion of a buffer to break up a long interconnect.

Clock Tree Optimizations



Gate relocation

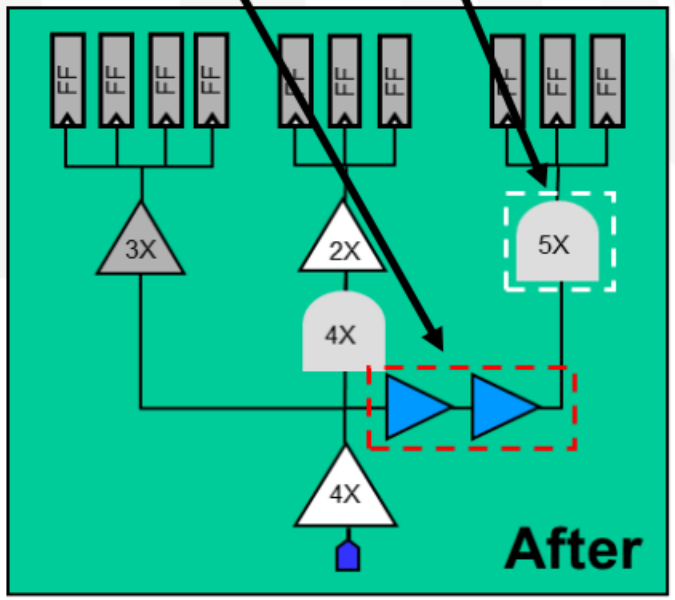
Buffer relocation

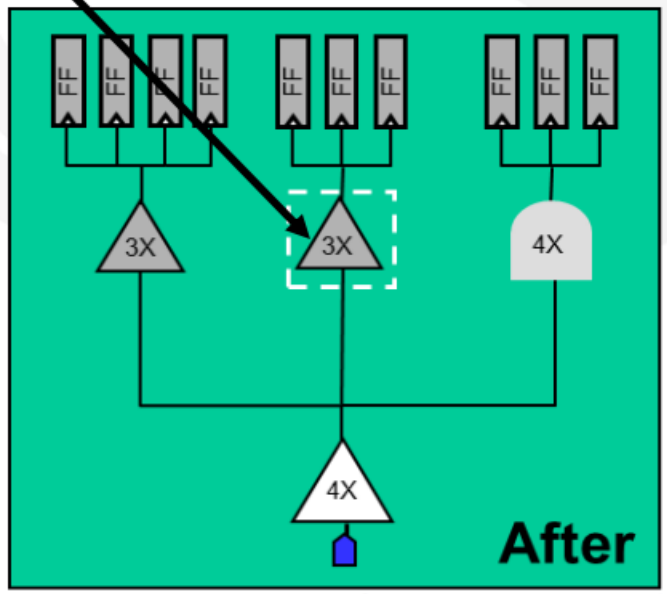


Delay insertion

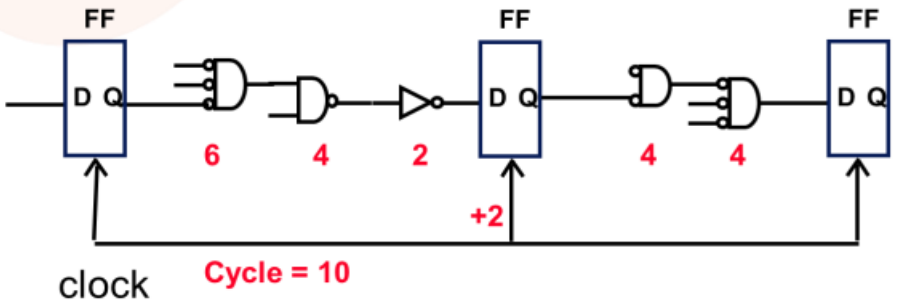
Gate sizing

Buffer sizing





Useful Skew



https://vlsibegin.blogspot.com/p/cts-clock-tree-synthesis.html (CTS – worthwhile)

Difference between High Fanout Net Synthesis (HFNS) & clock tree Synthesis.

- Buffers and clock inverter with equal rise and fall times are used. whereas HFNS uses buffers and inverters with a relaxed rise and fall times.
- HFNS are used mostly for reset, scan enable and other static signals having fanouts. there is not stringent requirement of balancing and power reduction.
- Clock tree power is given special attention as it is a constantly switching signal. HFNS are mostly performed for static signals and hence not much attention to power is needed.
- NDR rules are used for clock tree routing

Difference between clock buffer and normal buffer

Many spice simulations needed

- Clock buffer have equal rise time and fall time, therefore pulse width violation is avoided. In clock buffer Beta ratio is adjusted such that rise and fall time are matched. this may increase size of clock buffer compared to normal buffer.
- Normal buffer may not have equal rise and fall time. Clock buffers are usually designed such that an input signal with 50% duty cycle produces an output with 50% duty cycle.

Pre-CTS Optimization

Note: first read the place opt and then continue with Pre CTS.

Set the Optimization Directives

- don't_use, size_only

Perform High Fanout Nets Synthesize (HFNS)

- High Fanout Nets are Synthesized before Clock Tree Synthesis
- HFNS is the Buffering of High Fanout Nets
- Usually High Fanout Nets may have Fanout of more than 1000
  Eg., Reset, Clear etc.

Set CTS Routing Rules

- Shielding
- Non Default Rules (NDR)

Set RC Delay Models

https://lmr.fi/int/high-fanout-net-synthesis-hfns/

HFNS can be also done in synthesis.

Make sure an appropriate fanout limit is set

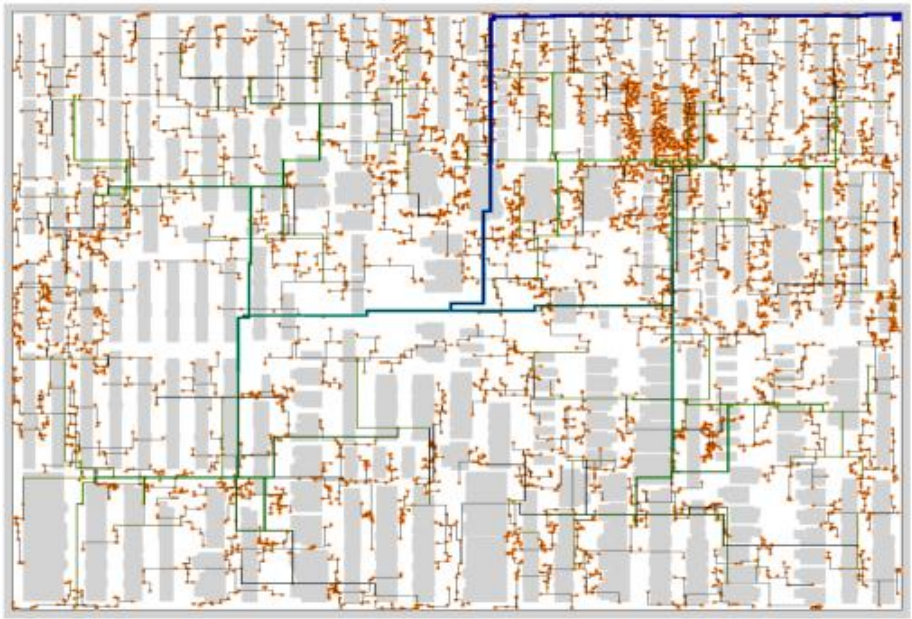
Sanity Checks need to be done before CTS

- Check legality
- Check power stripes, standard cell rails & also verify PG Connections.
- Timing QOR (Setup should be under control)
- Timing DRVs
- High Fanout nets (like scan enable/any static signal)
- Congestion (running CTS on congestion design / design with congestion hotspot can create more congestion and other issues (Noise/IR))
- Remove Don't_use attribute on clock buffers and inverters
- Check whether all pre-existing cells in clock path are balanced cells (CK* Cells)
- Check & qualify don't_touch, don't_size attributes on clock components.

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
- **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



Source: Universitat Bonn

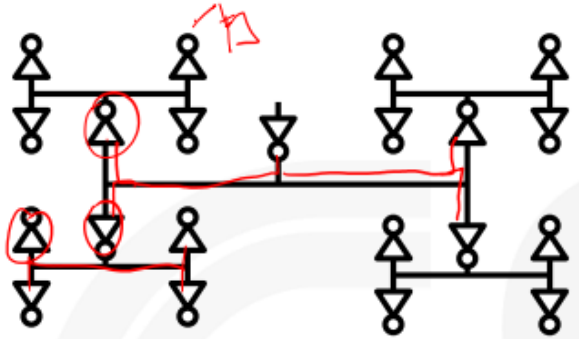
Building an actual Clock Tree

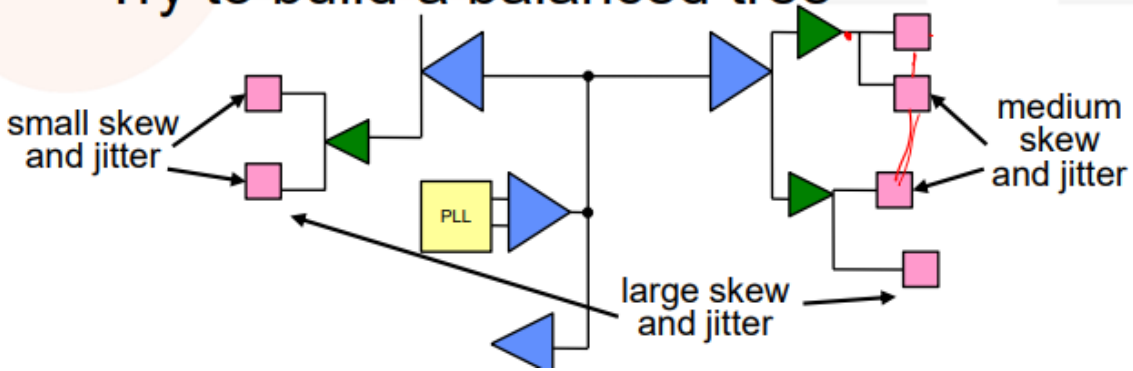
Clock Tree

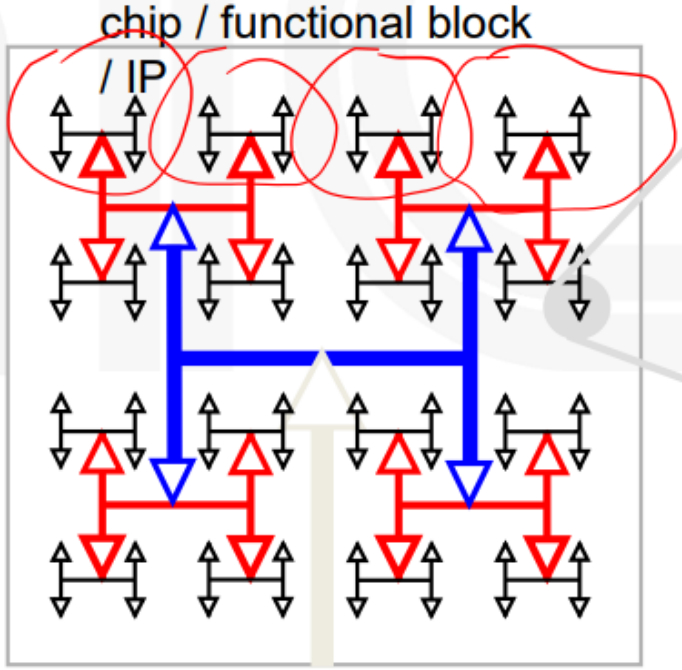
Clock Grid

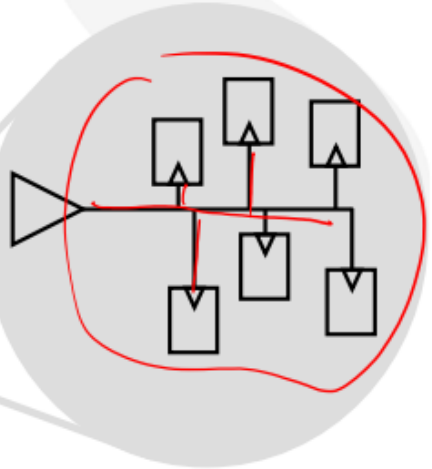
Clock Spine

- **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









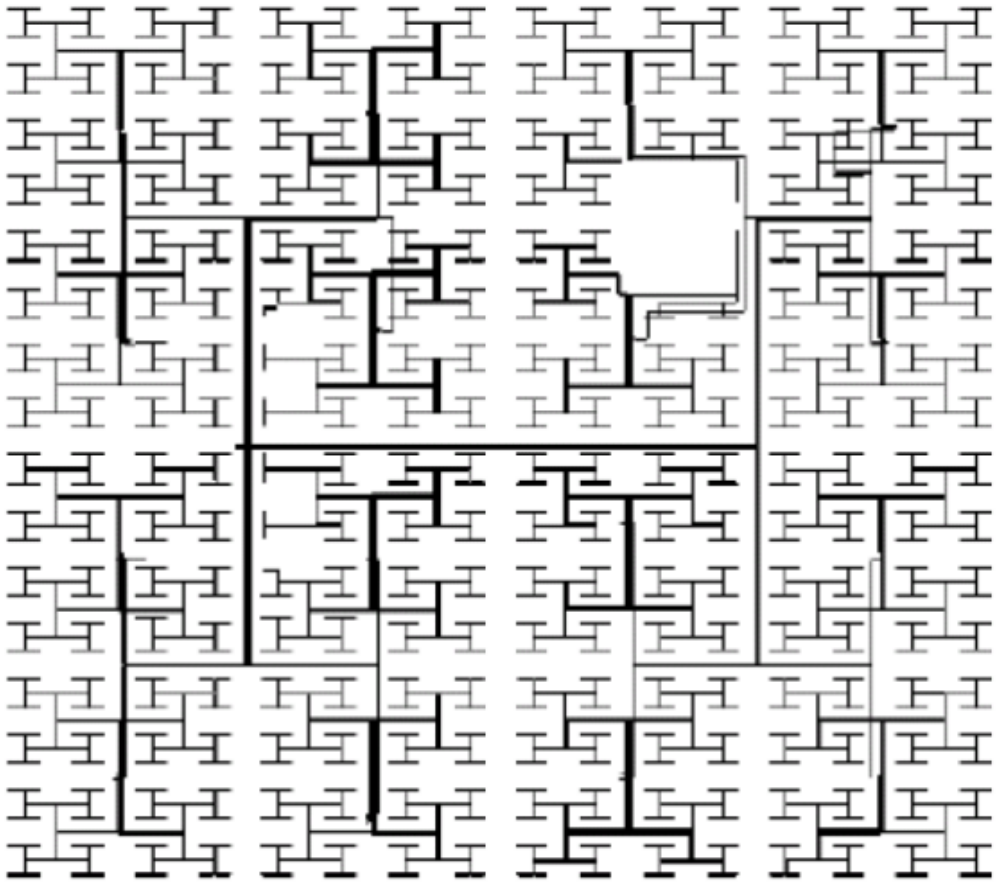
Industrial H-Tree Examples

Clock Tree

Clock Grid

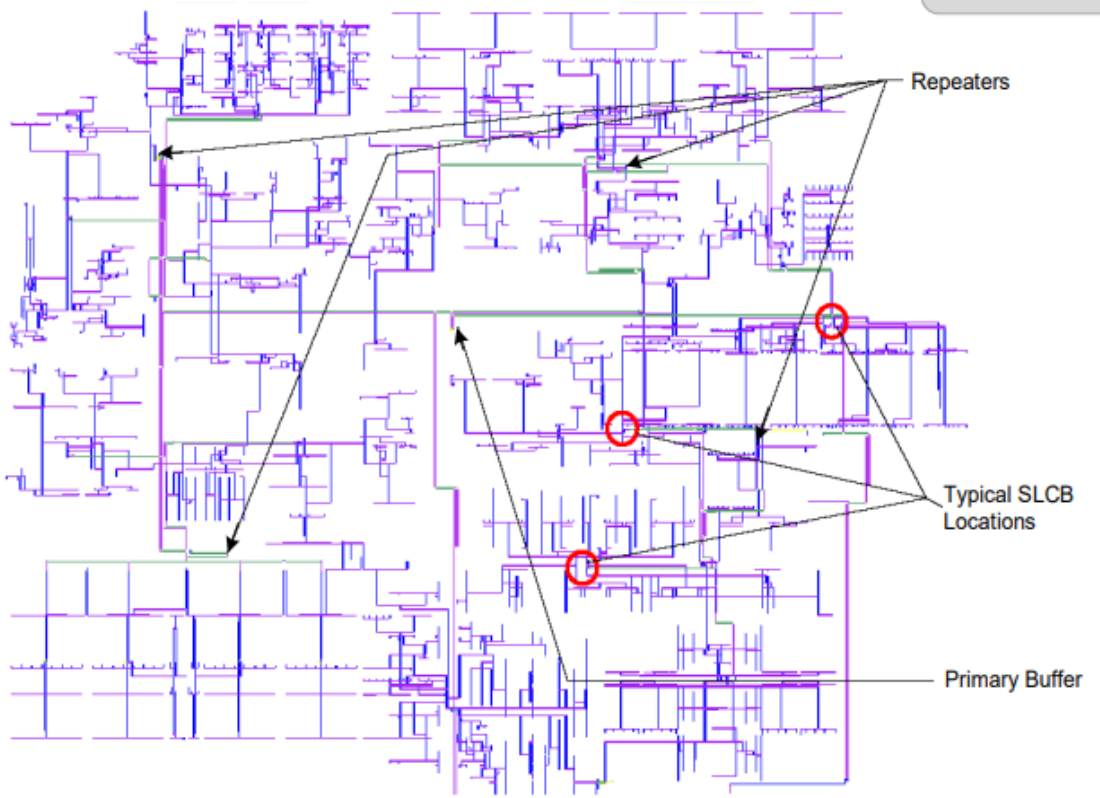
Clock Spine

- IBM PowerPC (2002)



IBM, ISSCC 2000

- Intel Itanium 2 (2005)



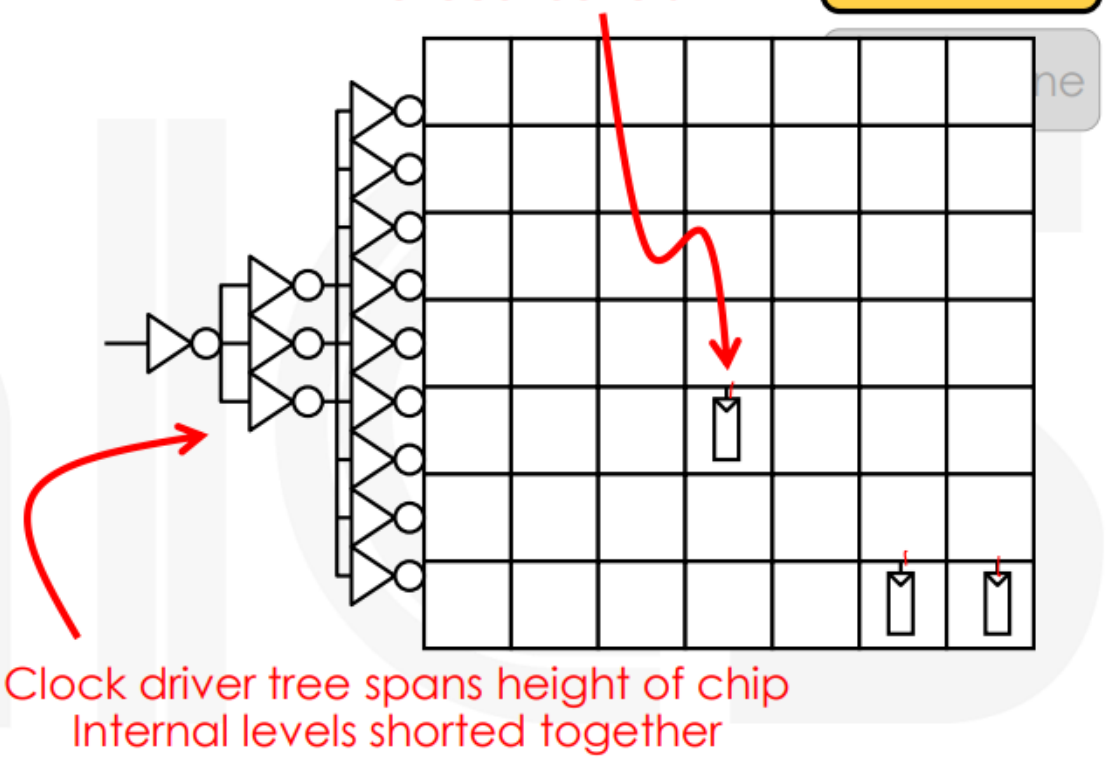
Source: CMOS VLSI Design, 4th Ed.

Lower Skew – Clock Grids

Clock Tree

Clock Grid

- **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
- **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



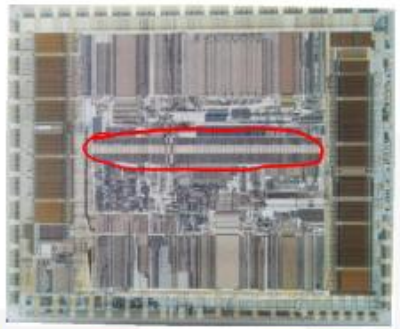
DEC Alpha – Generations of Grids

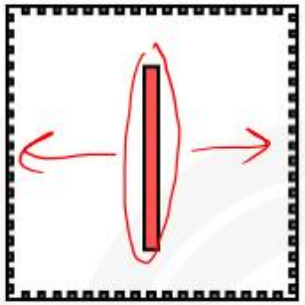
Clock Tree

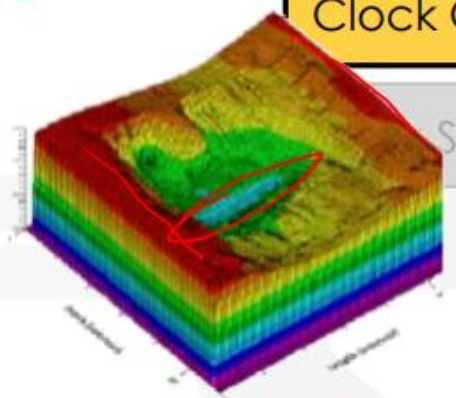
Clock Grid

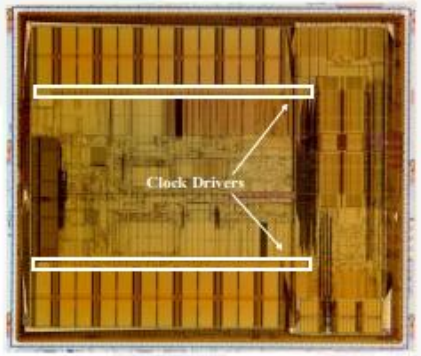
Spine

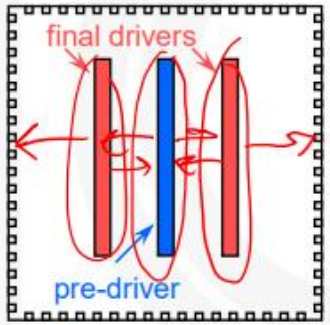
- **21064 (EV4) – 1992**
  - 0.75µm, 200MHz, 1.7M trans.
  - Big central driver, clock grid
    - 240ps skew
- **21164 (EV5) - 1995**
  - 0.5 µm, 300MHz, 9.3M trans.
  - Central driver, two final drivers, clock grid
    - Total driver size – 58 cm!
    - 120ps skew
- **21264 (EV6) - 1998**
  - 0.35 µm,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

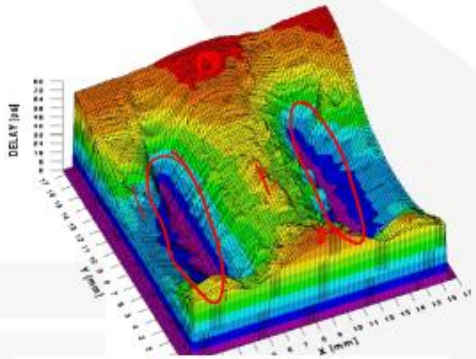


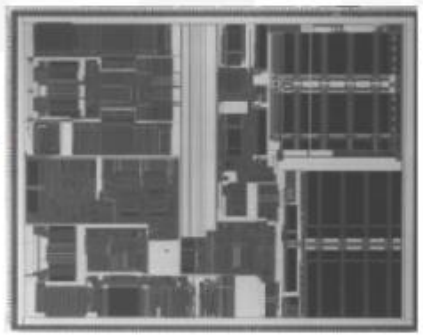


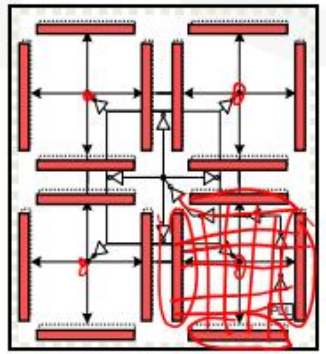


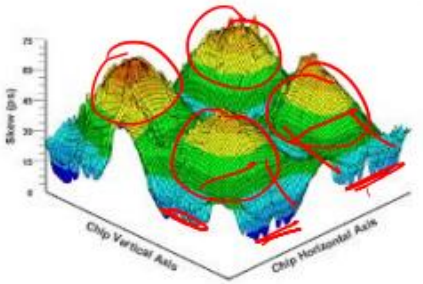












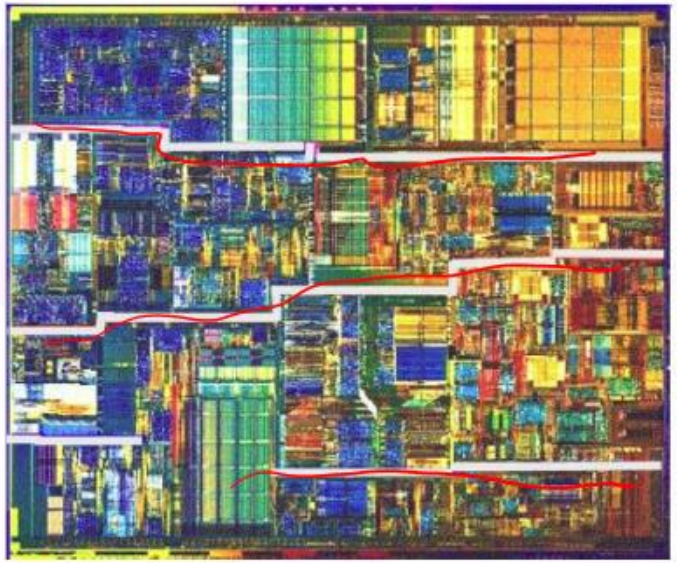
Clock Spines

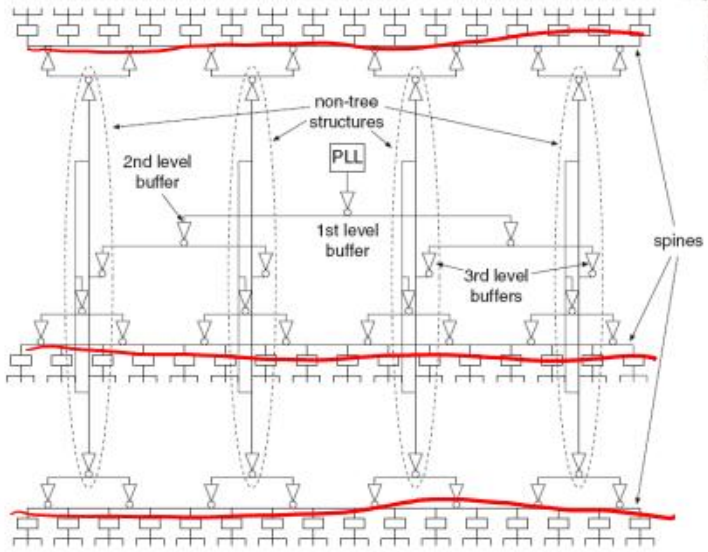
- **Clock grids are too power (and routing) hungry.**
- **A different approach is to use spines**
  - Build an H-Tree to each spine
  - Radiate local clock distribution from spines
- **Pentium 4 (2001) used the clock spine approach.**

Clock Tree

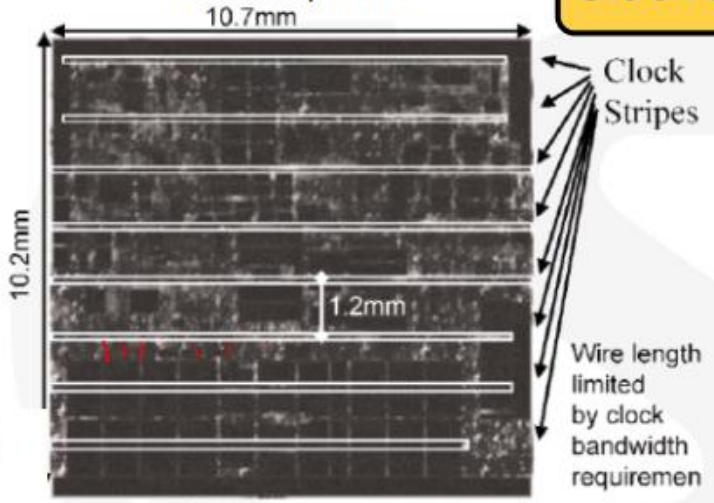
Clock Grid

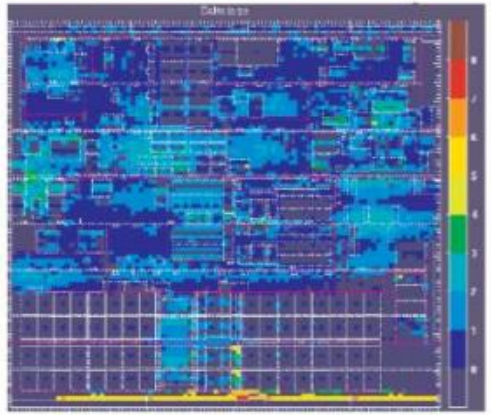
Clock Spine





Later Pentium 4's used more spines





delay in ps

Source: Bindal ISSCC 2003

Summary of main clock dist. approaches

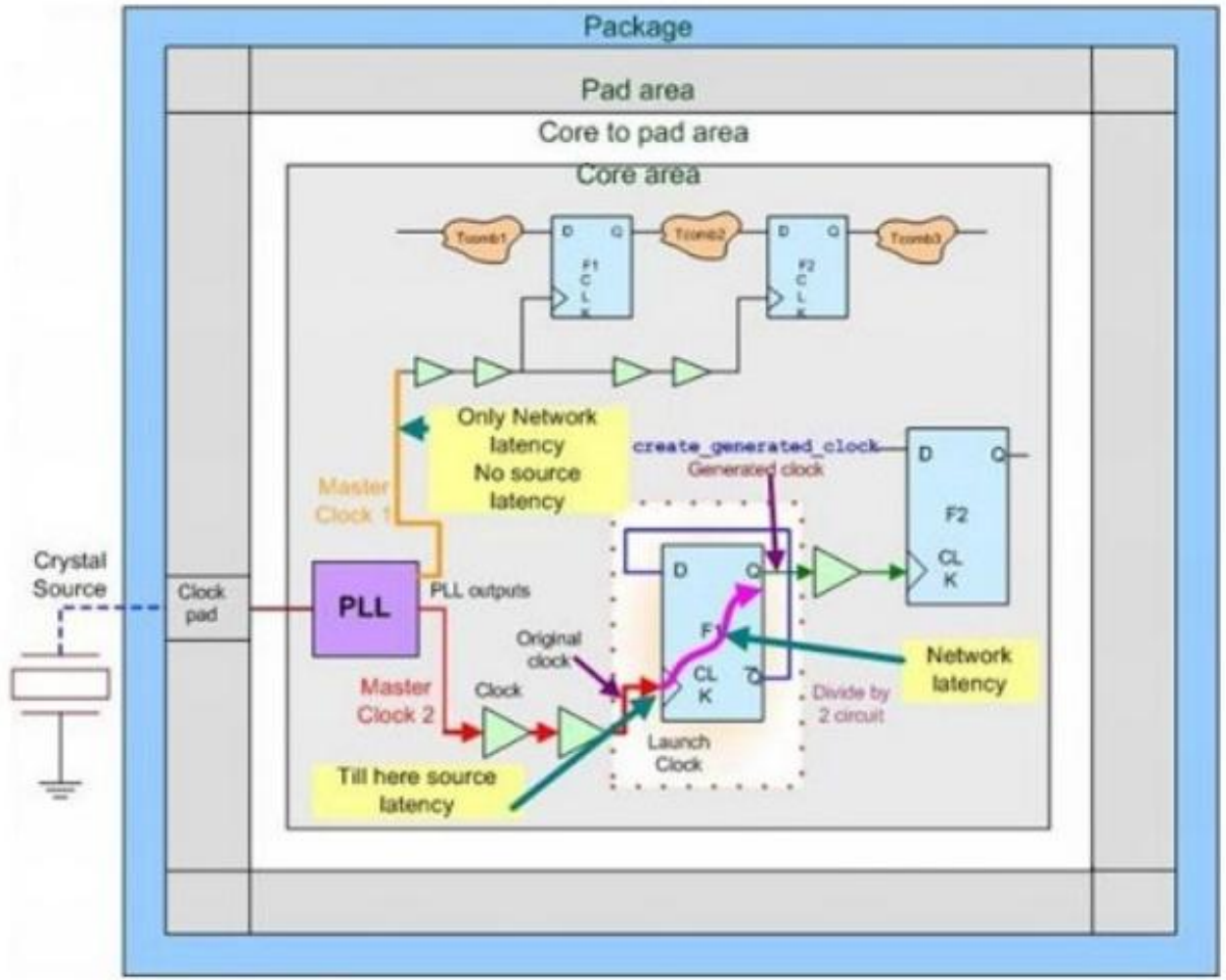
Three basic routing structures for global clock:

- **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

Insertion Delay (ID)

- ID is the clock latency, but after Clock Tree is synthesized
- ID is the physical delay and Clock Latency is the virtual delay
- Latency is a target given to the tool through SDC file or clock tree attribute file and Insertion Delay is the achieved delay value after CTS
- Source and Network Latency (Original Clock & Generated Clock)



Concurrent Clock and Data Optimization (CCD)

https://www.youtube.com/watch?v=pP25JOAxYYA

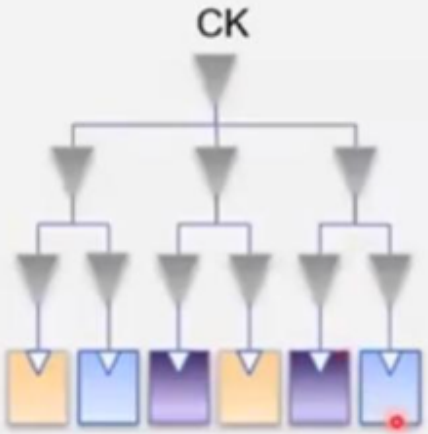
PD Lec 47 - concurrent clock and data optimization| CCD| Timing | placement | VLSI | Physical Design

https://www.programmersought.com/article/86877197473/#google_vignette

Nowadays clock tree and data path are considered together

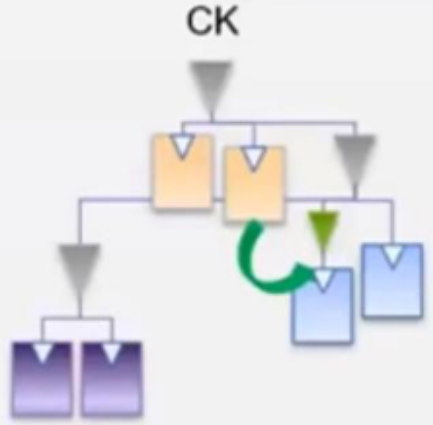
Comparing Classic CTS versus CCD

Classic CTS



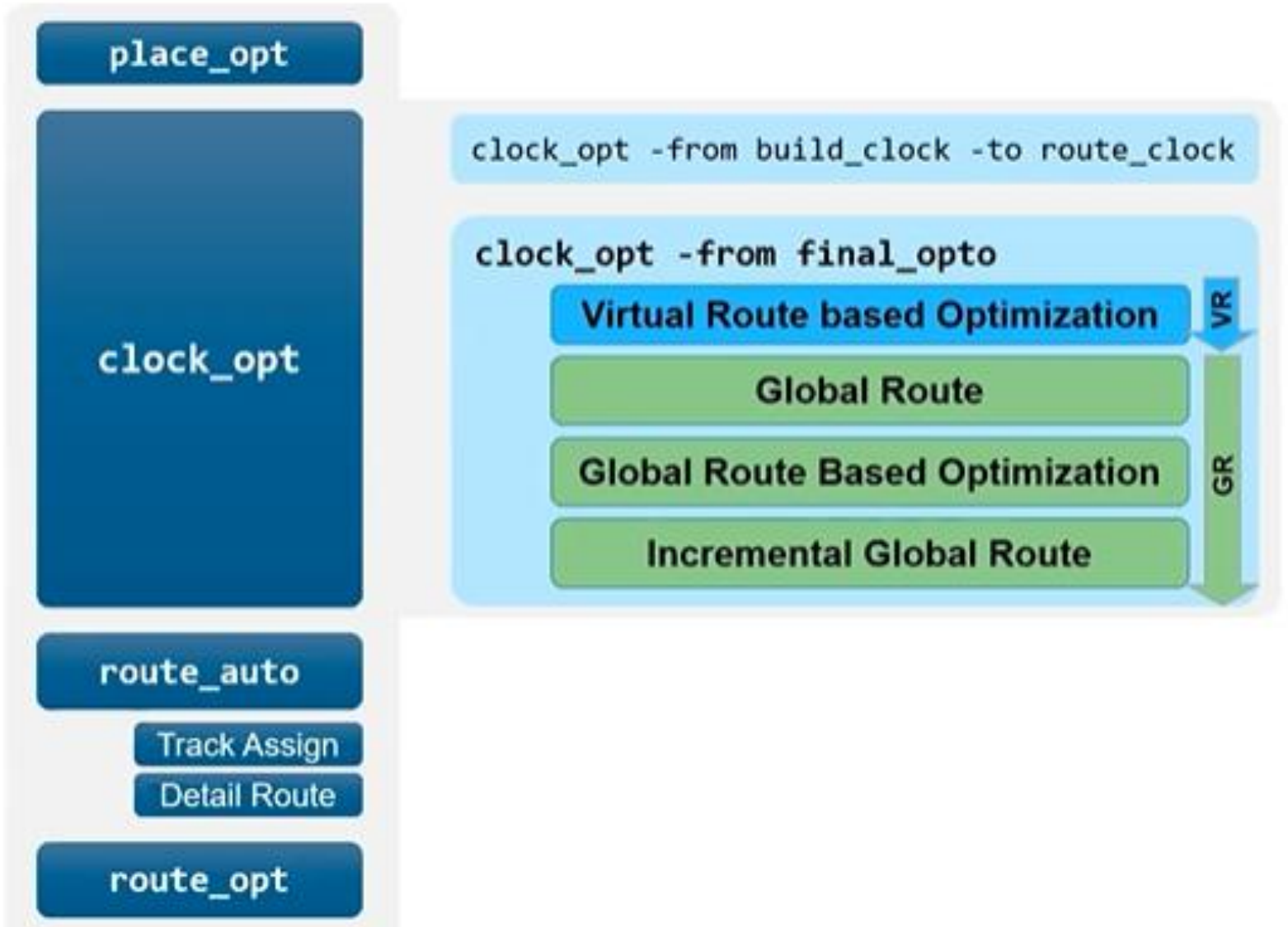
- Clock tree is built while ignoring the data paths
- Goal: Minimize skew
- Data path is optimized post-CTS
  - Clock tree remains untouched

Concurrent Clock&Data



- Clock tree is built with full knowledge of data path timing
- Goal: Meet setup/hold timing
- Data path is optimized along with incremental clock tree modifications *(green buffer)*

Post-CTS Optimization with GRE



Timing Fixes in placement

https://www.youtube.com/watch?v=hzJZuBGC418

PD Lec 43 - Timing Fixes in placement | Part-1 | VLSI | Physical Design

Recap

Welcome suggestions
Please fill out eval form

- Starting from algorithm, complexity analysis
  - Several graph search methods – travelling salesman problem, breadth first, depth first, branch-and-bound, longest path, shortest path, …
  - Dynamic programming, bin packing, topological sort
- Partitioning, floorplan, placement
  - Min-cut with terminal propagation, Gordian (quadratic programming), force-directed, clustering, simulated annealing
  - Slicing (polish expression), non-slicing (B*)
- Routing
  - Maze routing, line search, global route, track assignment, channel route, detailed route, design rules
- Clock tree routing, synthesis
- P/G routing, and co-optimization
- Logic optimization, BDD, tech mapping,
- Delay, timing
- Don’t forget taking courses about ML