Agenda

DAY 2

2 Design Planning (Lab – continued)

3 Placement

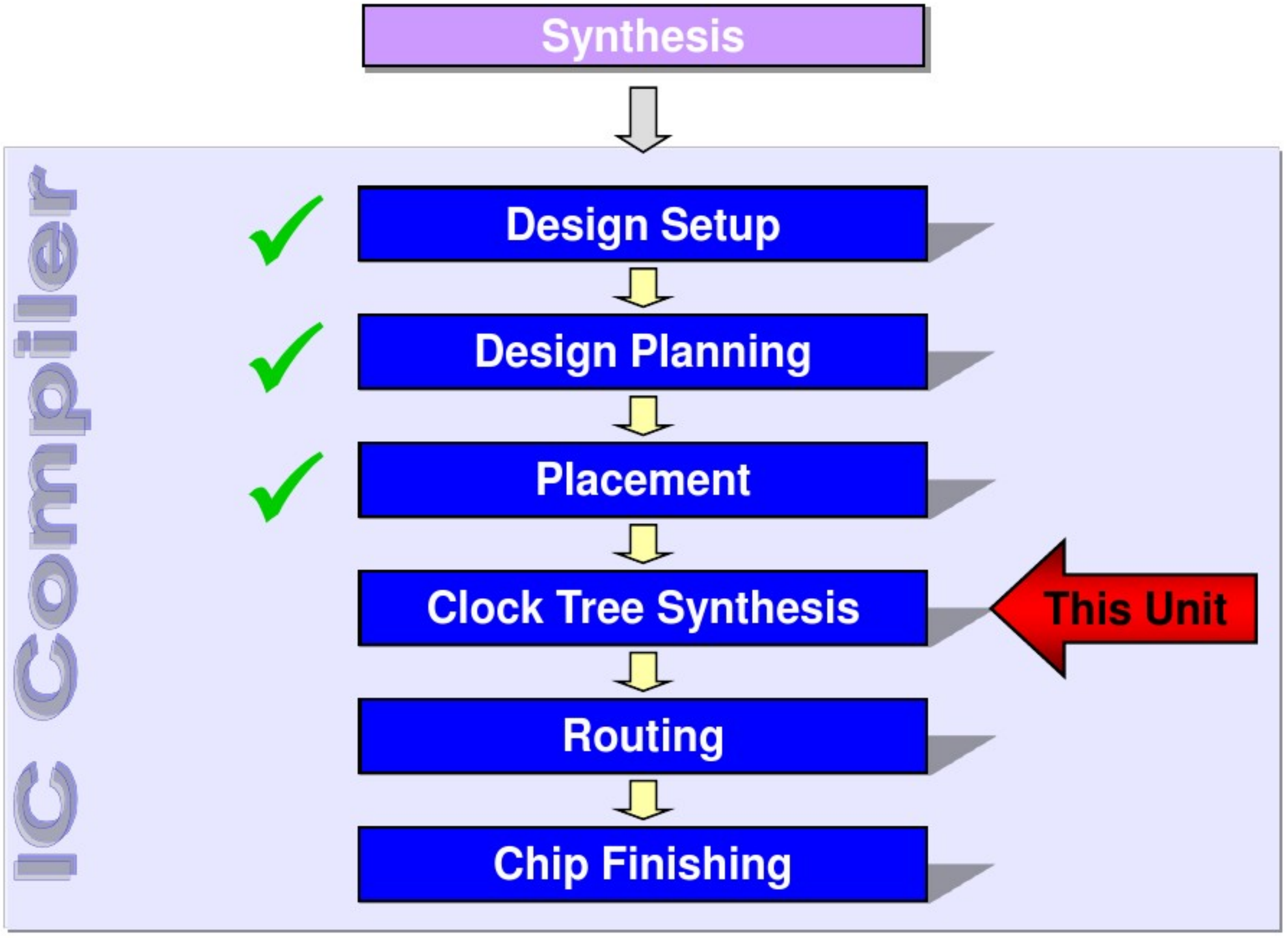
4 Clock Tree Synthesis

Unit Objectives

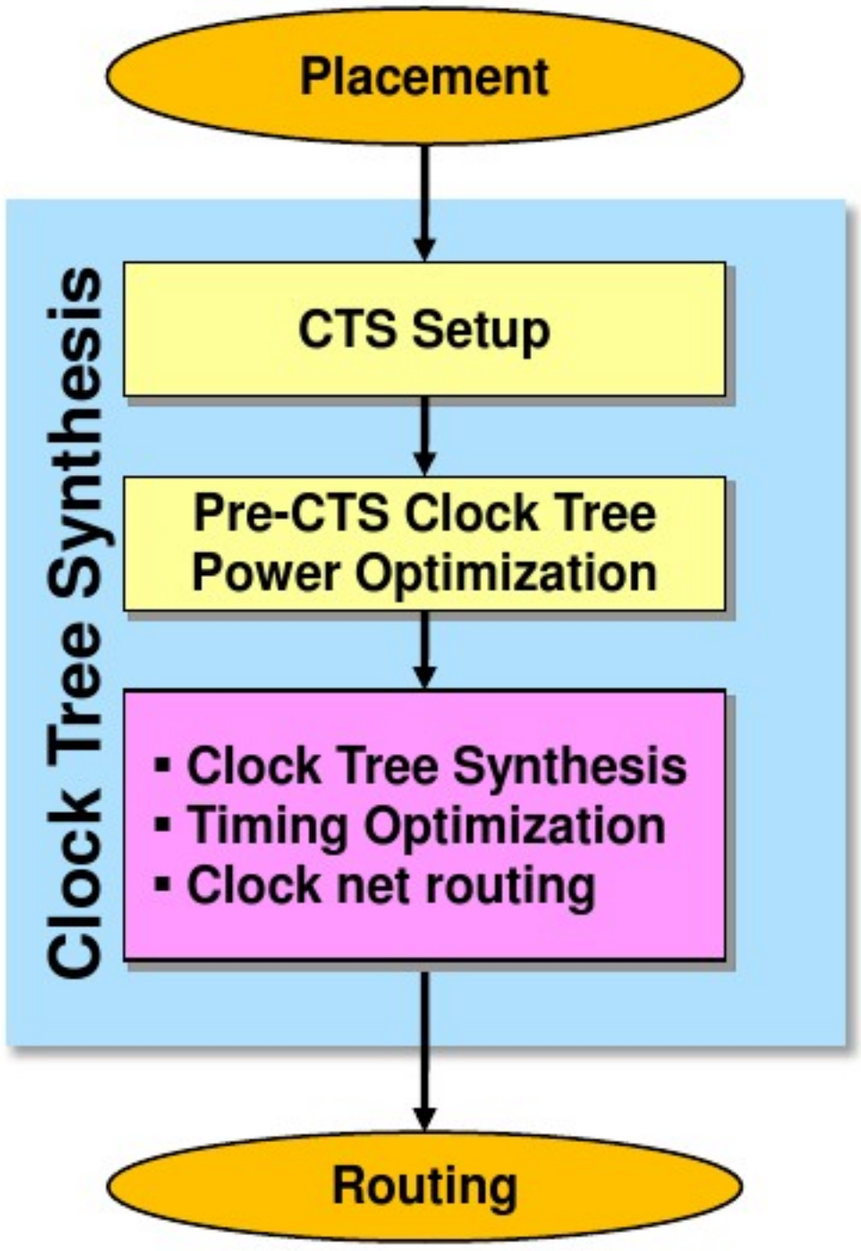
After completing this unit, you should be able to:

- List the status of the design prior to CTS
- Set up the design for clock tree synthesis
- Identify implicit clock tree start/end points and when explicit modifications are needed
- Control the constraints and targets used by CTS
- Execute the recommended clock tree synthesis and optimization flow
- Analyze timing and clock specifications post CTS

General IC Compiler Flow



IC Compiler Clock Tree Synthesis Flow



The "CTS phase" involves several key steps:

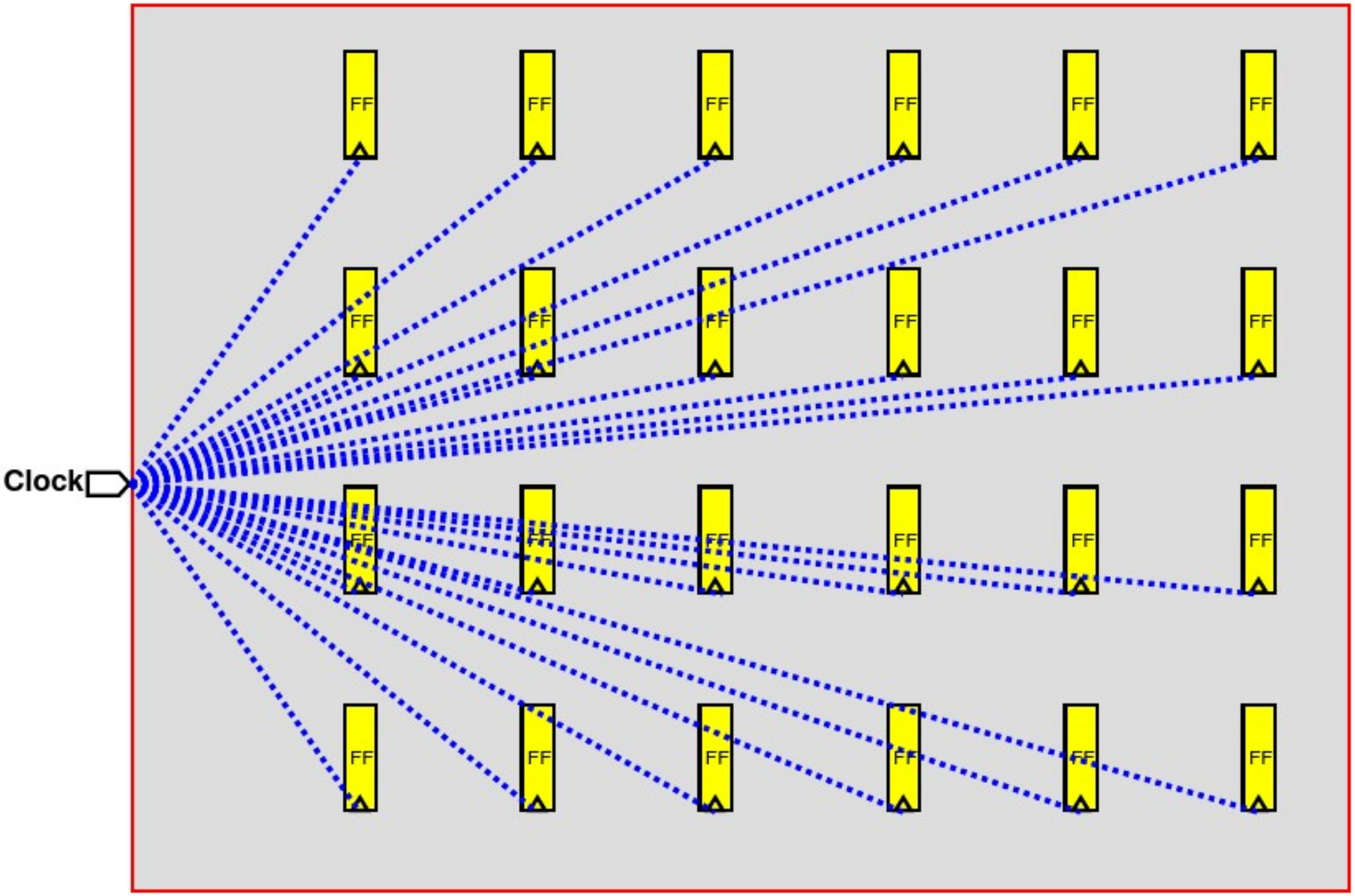
- **Setup steps to control CTS**
- **Optional Pre-CTS power optimization**
- **Clock Tree Synthesis**
- **Timing Optimization**
- **Routing of clock nets**

Note: The flow diagrams included in this unit represent an **example** flow, not the recommended flow

Design Status Prior to Clock Tree Synthesis

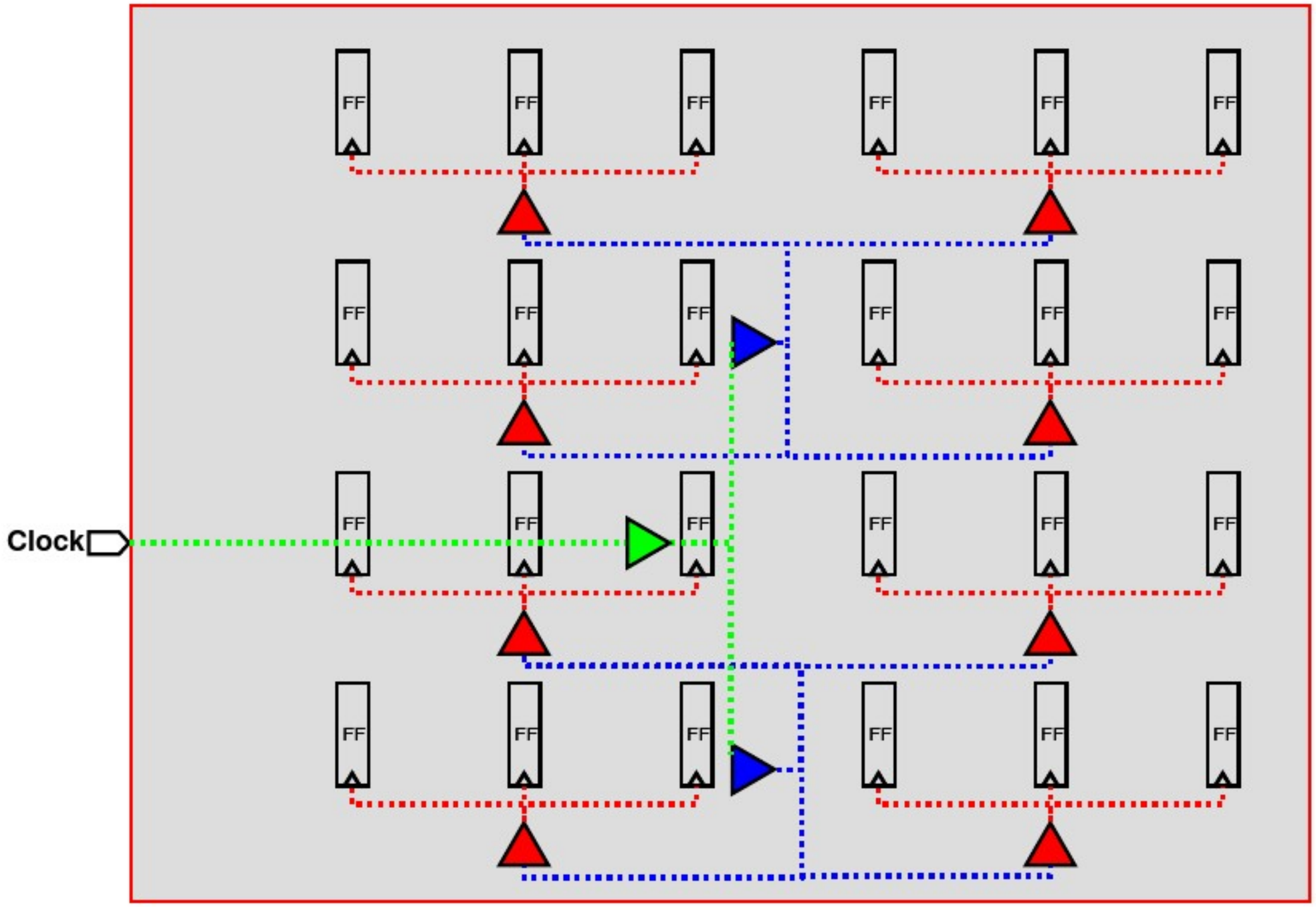
- **Placement - completed**
- **Power and ground nets – prerouted**
- **Estimated congestion – acceptable**
- **Estimated setup timing – acceptable (~0ns slack)**
- **Estimated max cap/transition – no violations**
- **High fanout nets:**
  - Reset, Scan Enable synthesized with buffers
  - Clocks are still not buffered

Starting Point before CTS



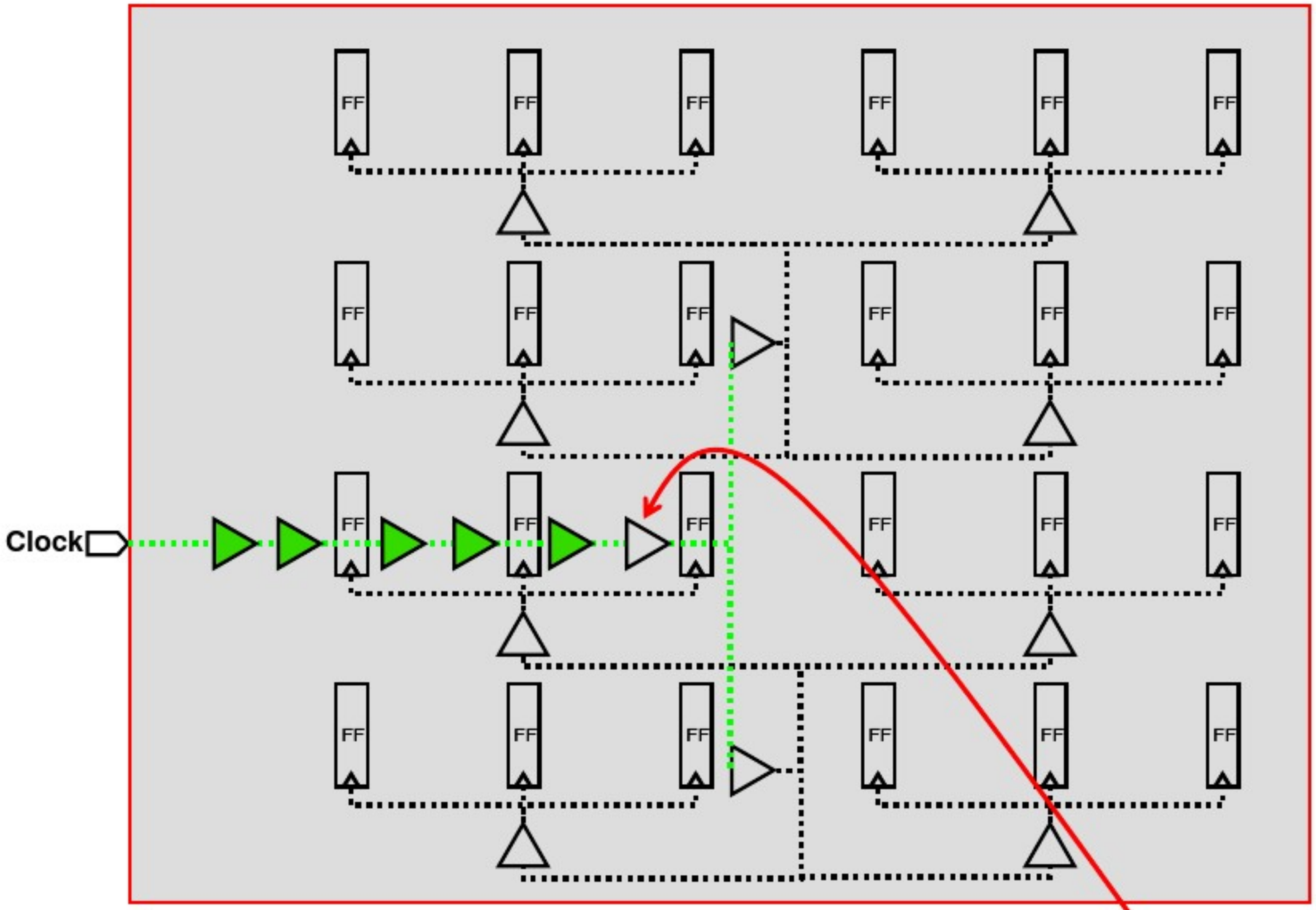
All clock pins are driven by a single clock source.

Clock Cells Are Inserted and then Resized



Buffers are inserted to balance the loads, meet DRCs and minimize the skew

Delay Cells Are Added to Meet Min. Insertion



Delay cells are placed behind the common single buffer to minimize clock skew impact

CTS Goals

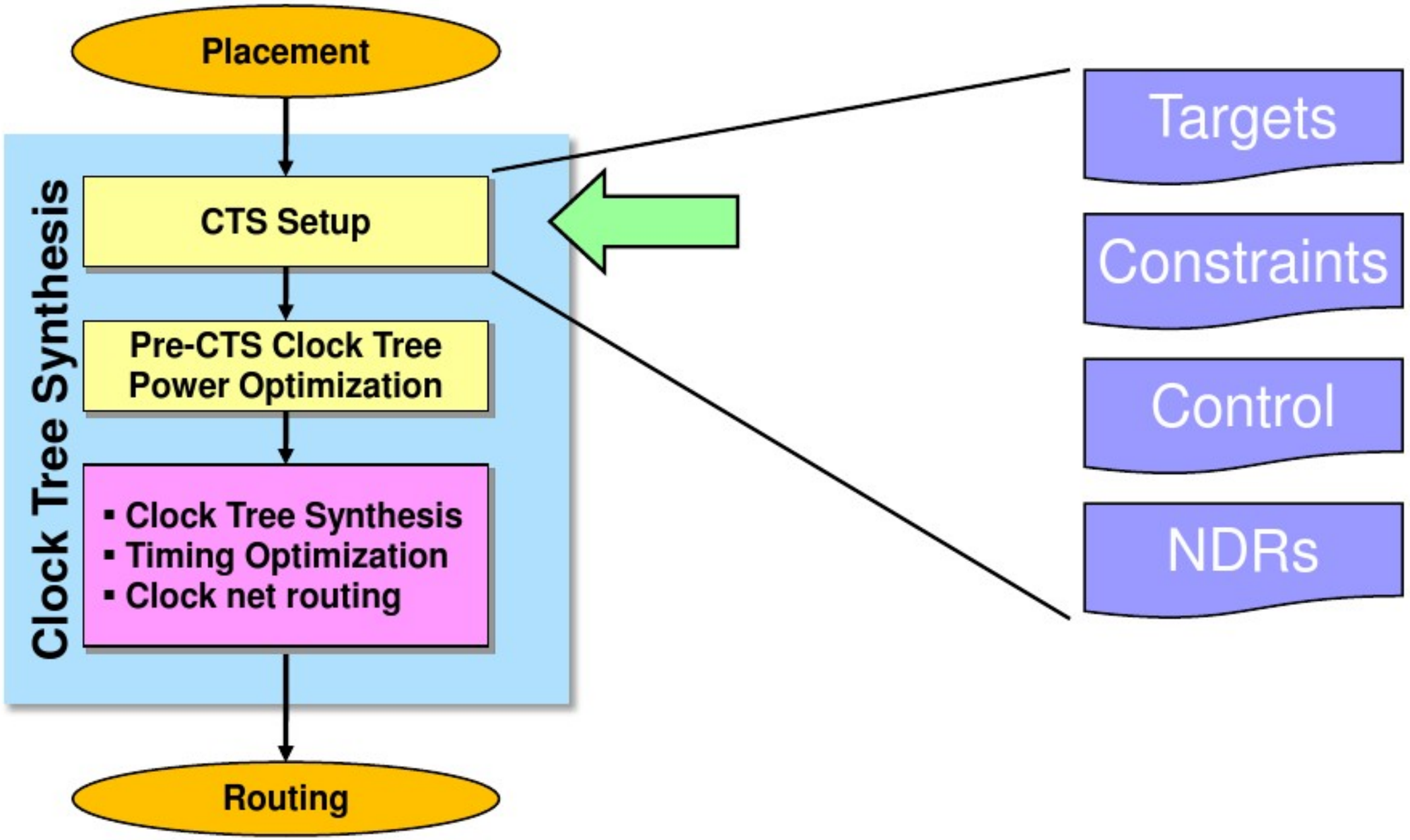
- **Meet the clock tree Design Rule Constraints (DRC):**
  - Maximum transition delay
  - Maximum load capacitance
  - Maximum fanout
  - Maximum buffer levels

Constraints are upper bound goals. If constraints are not met, violations will be reported.

- **Meet the clock tree targets:**
  - Maximum skew
  - Minimum insertion delay

Targets are "nice to have" goals. If targets are not met, no violations will be reported.

IC Compiler Clock Tree Synthesis Flow

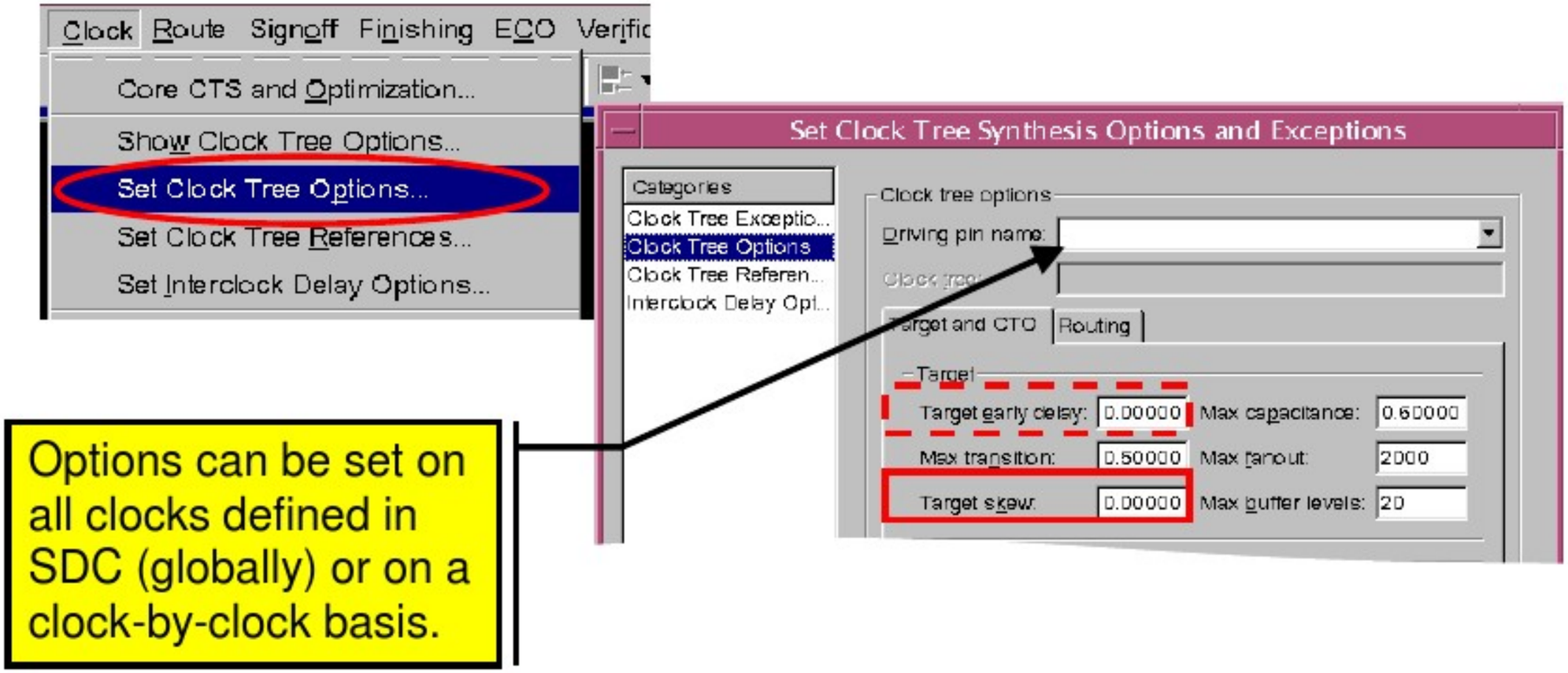


Default Clock Tree Targets

Targets

- **The default CTS target for skew and insertion delay is 0ns**
  - Uncertainty and insertion delay SDC constraints are ignored
- **It is recommended to relax the clock skew target as much as possible**
  - Reduces overall buffer count and run time
- **Specify minimum clock latencies as needed**

Specifying Global Targets




```
icc_shell> set_clock_tree_options \
           -target_early_delay 0.9 \
           -target_skew 0.1
Setting global CTS options...
1
```

Specifying Clock-Specific Targets

How do you set different targets per clock?

- **Use the GUI, select the appropriate clock from the pull-down, and OK your selections for every clock**
- **Better:**

```
set_clock_tree_options \
            -clock_trees clk1 -target_early_delay 0.9
set_clock_tree_options \
            -clock_trees clk2 -target_skew 0.2
```

TIP: Use the GUI preview mode, then cut&paste the echo'd commands into your script.

Control Buffer/Inverter Selection

- **To set specific buffers/inverters to be used during each of the specific CTS optimizations:**

```
set_clock_tree_references
   -references list1 (DRC buffering)
   -references list2 -sizing_only (skew balancing)
   -references list3 -delay_insertion_only
```

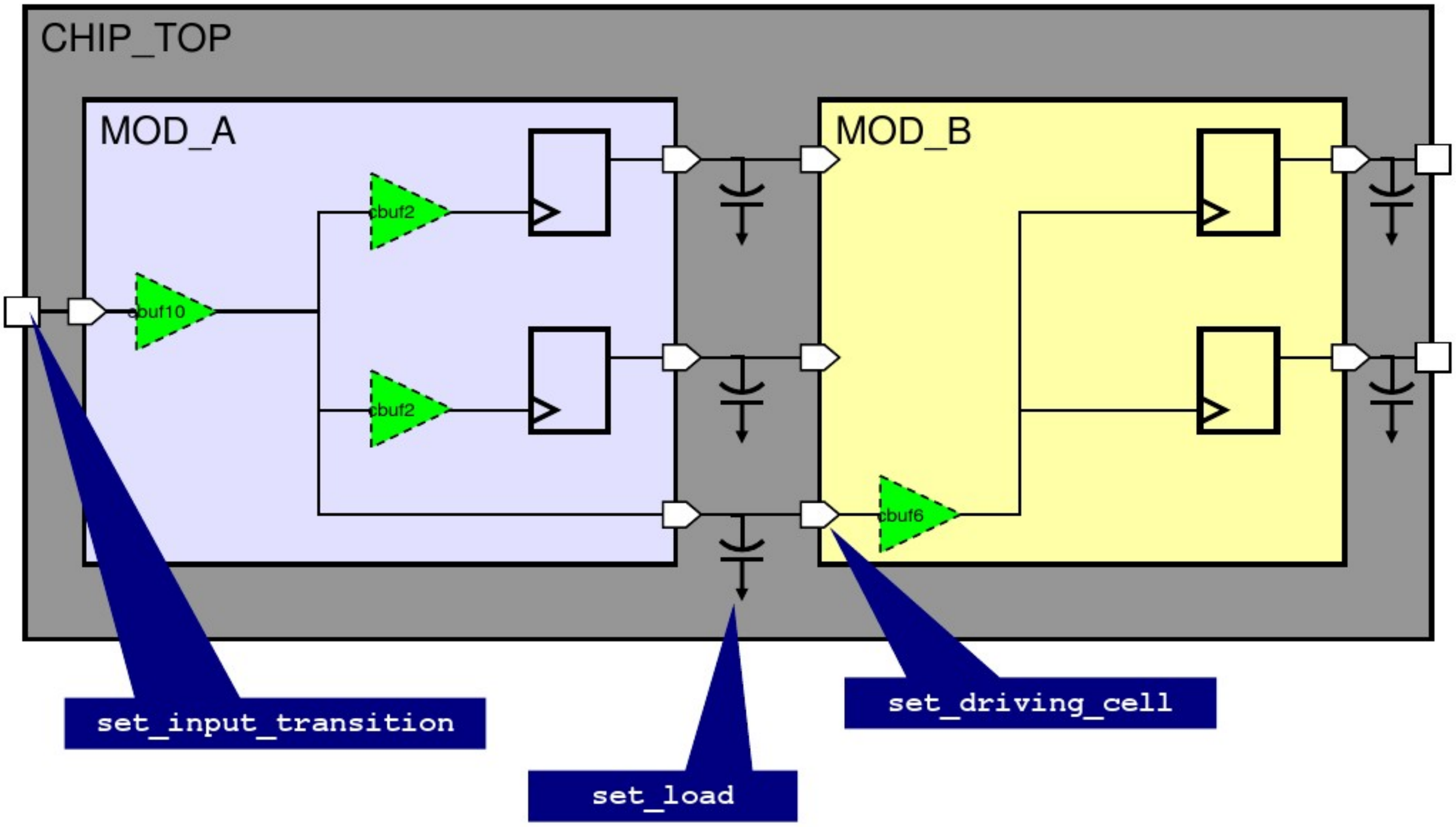
It is recommended to define all lists

  - **There is no priority on how CTS uses the members from each list**
  - **If a list is not specified, by default all buffers/inverters (except `dont_use` cells) in the library can be used during its respective optimization**

Make sure the references are in `target_library`

Are all Clock Drivers and Loads Specified?

Constraints



Remove "Skew" from *Uncertainty*

- **Your SDC constraints will most likely include a `set_clock_uncertainty -setup <number>` applied to each clock**
  - This command is used to model estimated *clock skew*, but can also be used to model the effects of *clock jitter* and to include some additional *timing margin*
  - The specified setup number reduces the effective clock period of all paths captured by the specified clock and is used during synthesis to estimate clock behavior
- **Timing analysis post-CTS will also include the effects of this command, therefore:**
  - Remove clock uncertainty if only skew is included

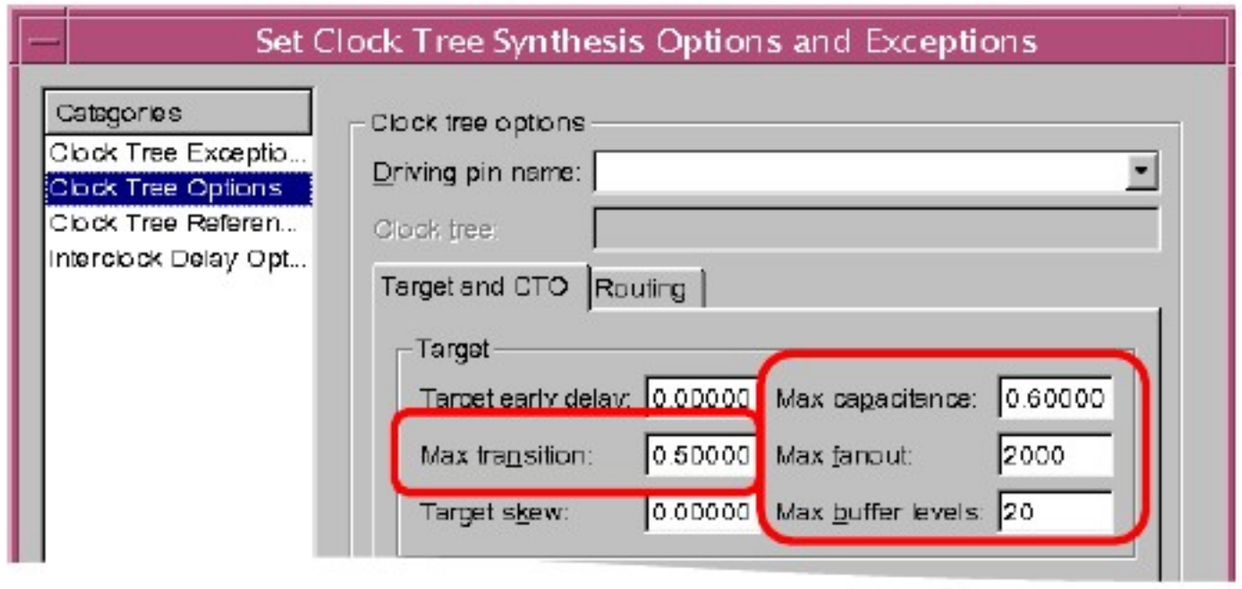
```
remove_clock_uncertainty [all_clocks]
```

OR

  - Reduce the uncertainty number by the estimated skew

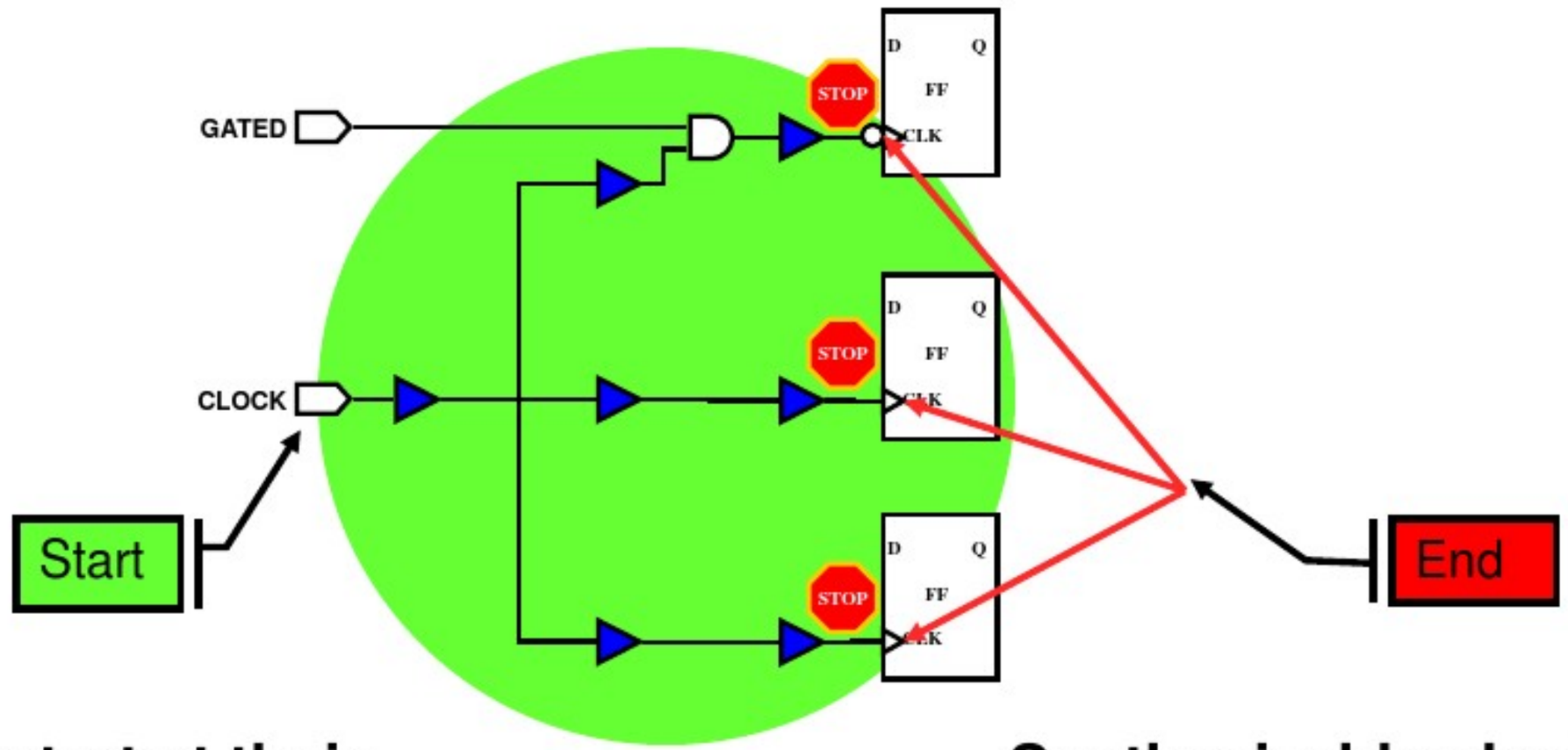
Defining CTS-Specific DRC Values

- **Max transition and max capacitance design rules can be specified in three ways: Library, SDC and CTS setting**
  - By default IC Compiler will use the smallest of the three
- **The default CTS settings are set for today's technologies**
  - If using >> 90nm you may need to relax (increase) the numbers



Where Does the Clock Tree Begin and End?

Control



Clock trees start at their "source" defined by

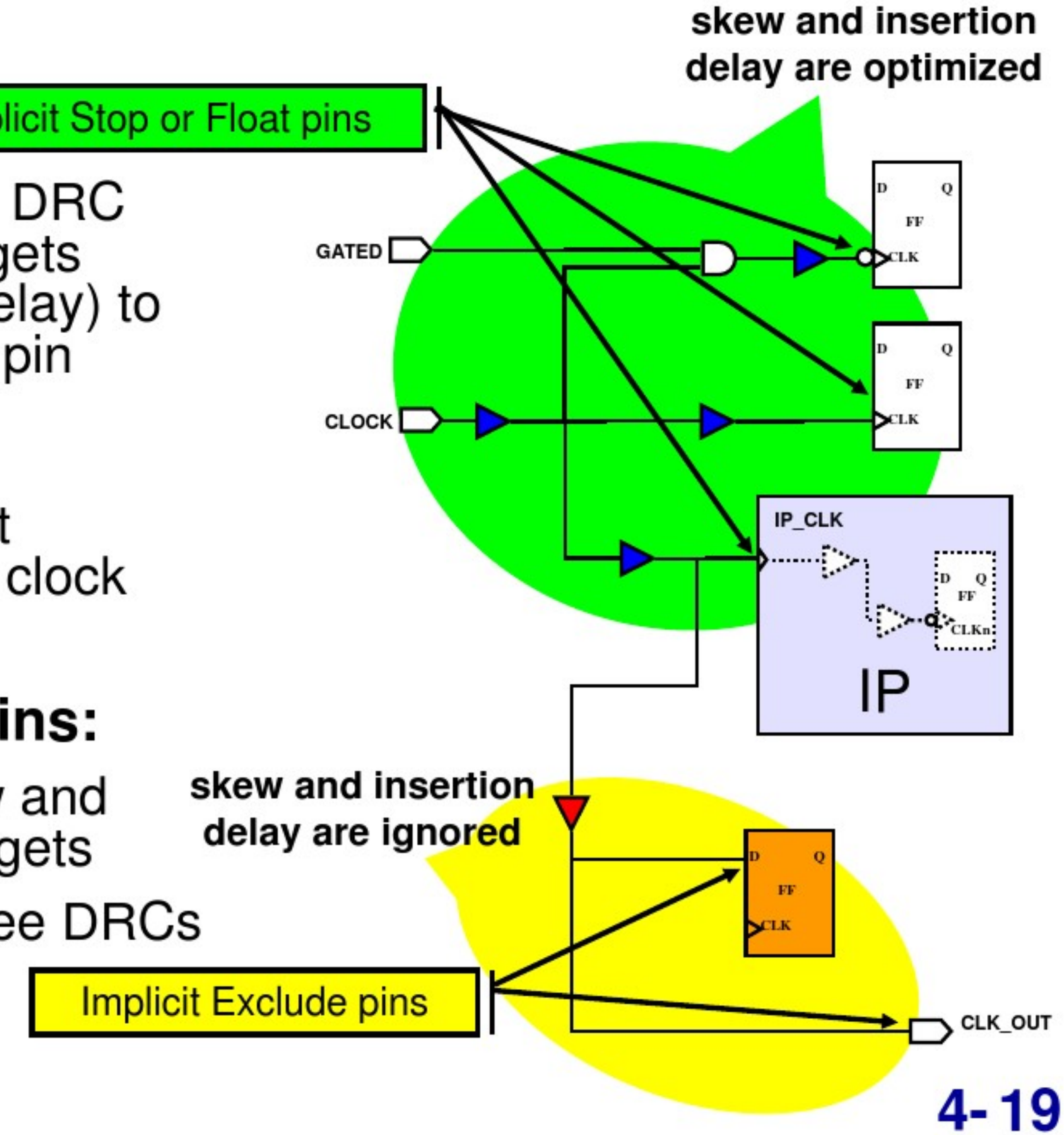
`create_clock` ...

Synthesizable clock trees end (by default) on clock pins of registers or macros

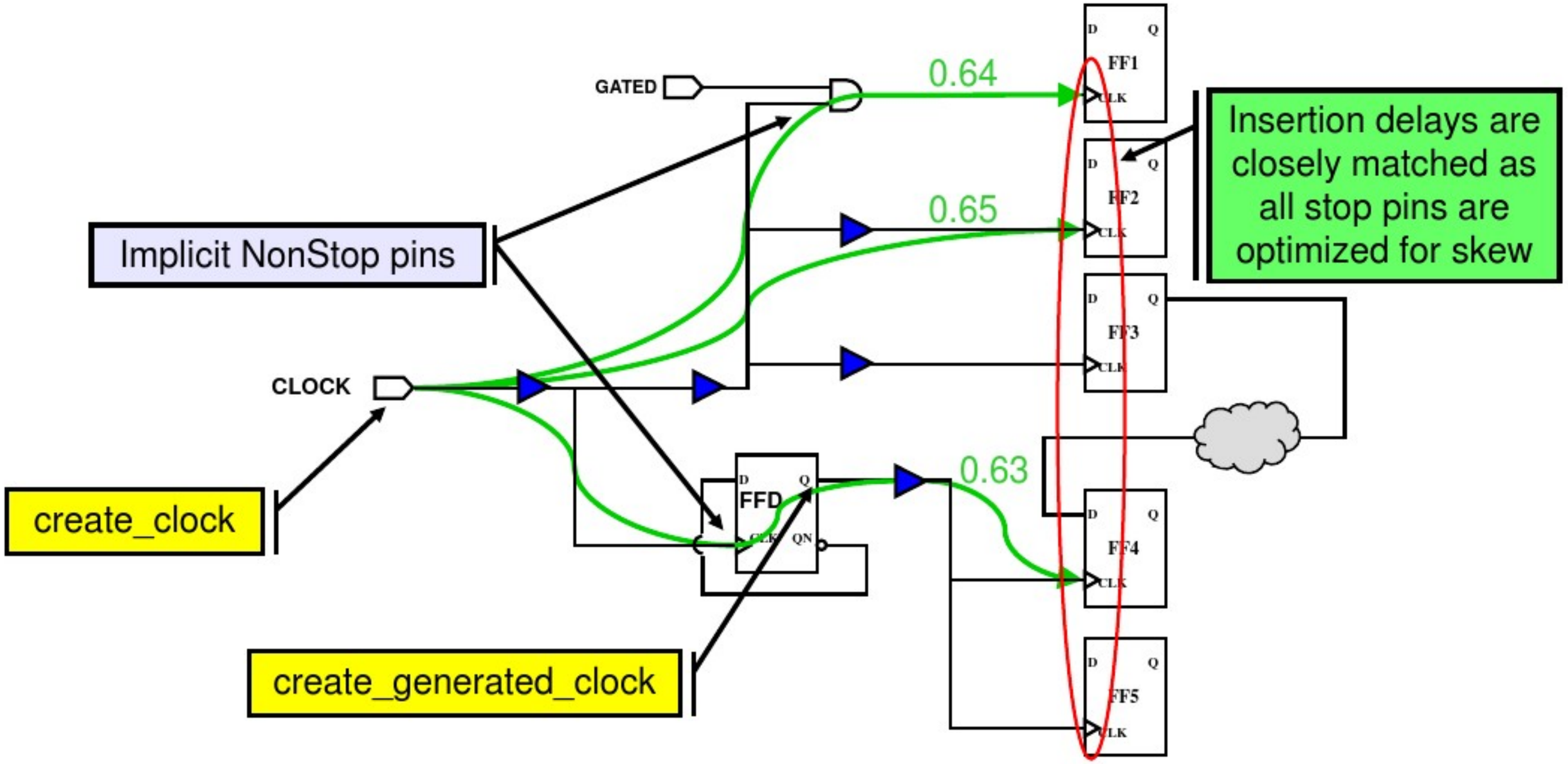
(*stop* or *float* pins)

Stop, Float and Exclude Pins

- **Stop Pins:**
  - CTS optimizes for DRC and clock tree targets (skew, insertion delay) to the *external* clock pin
- **Float Pins:**
  - Like Stop pins, but considers *internal* clock latency
- **Exclude (Ignore) Pins:**
  - CTS ignores skew and insertion delay targets
  - CTS fixes clock tree DRCs



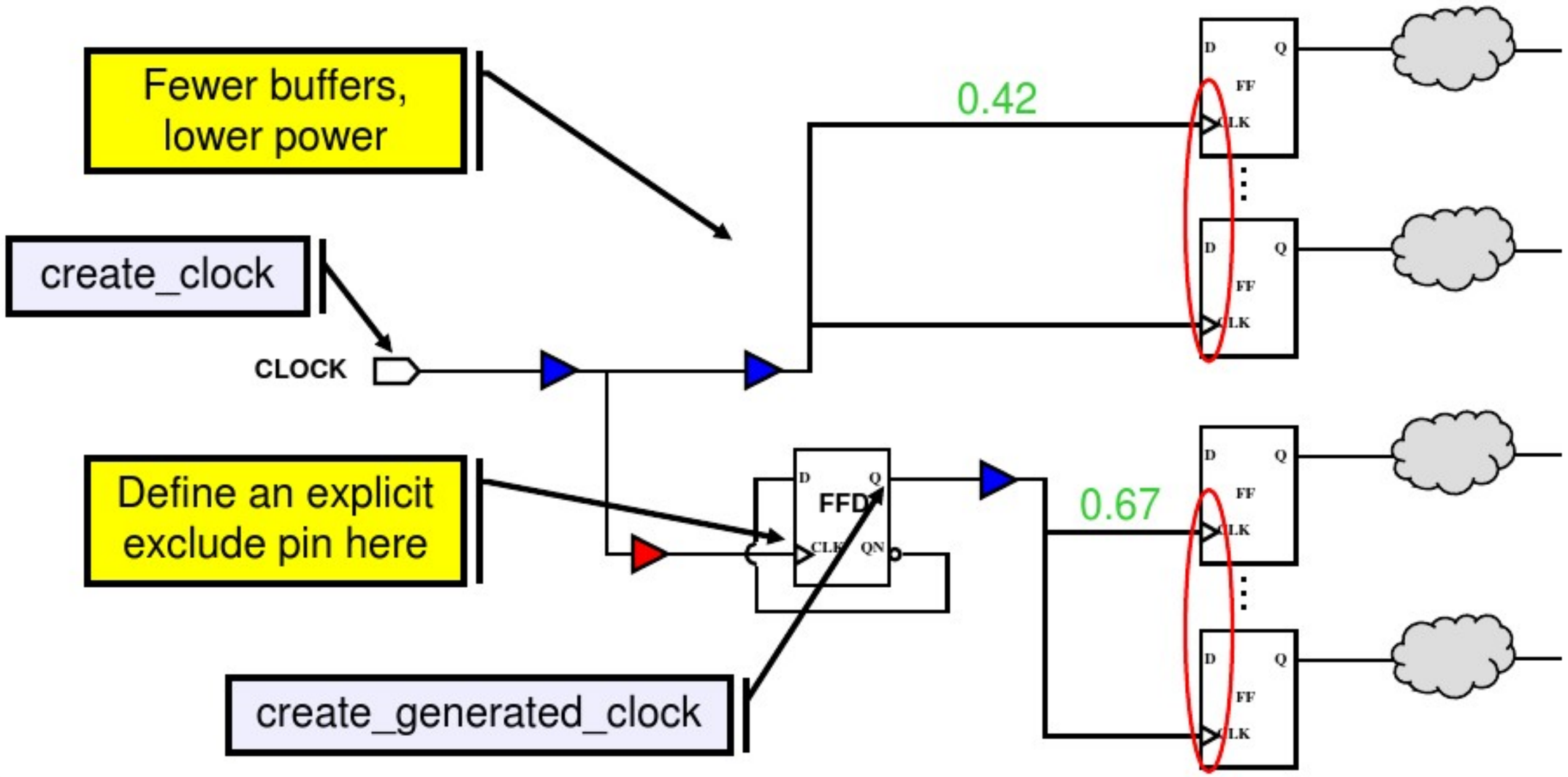
Generated and Gated Clocks



Skew will be balanced ‘globally’, within each clock domain, across all clock-pins of both master and generated clock.

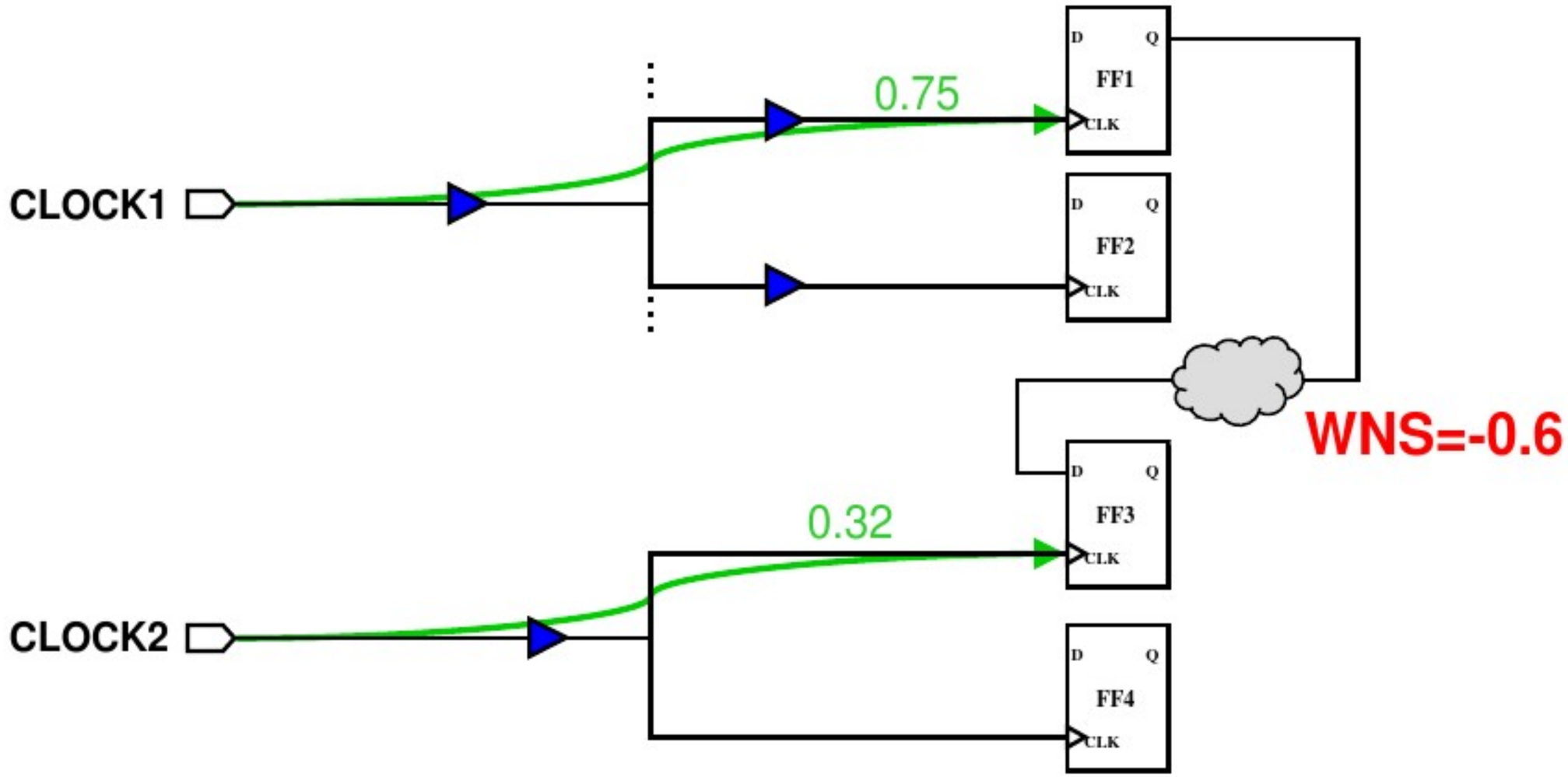
Skew Balancing not Required?

If a generated clock domain is independent of the master domain (no crossing paths) skew balancing may not be important




```
set_clock_tree_exceptions -exclude_pins [get_pins FFD/CLK]
```

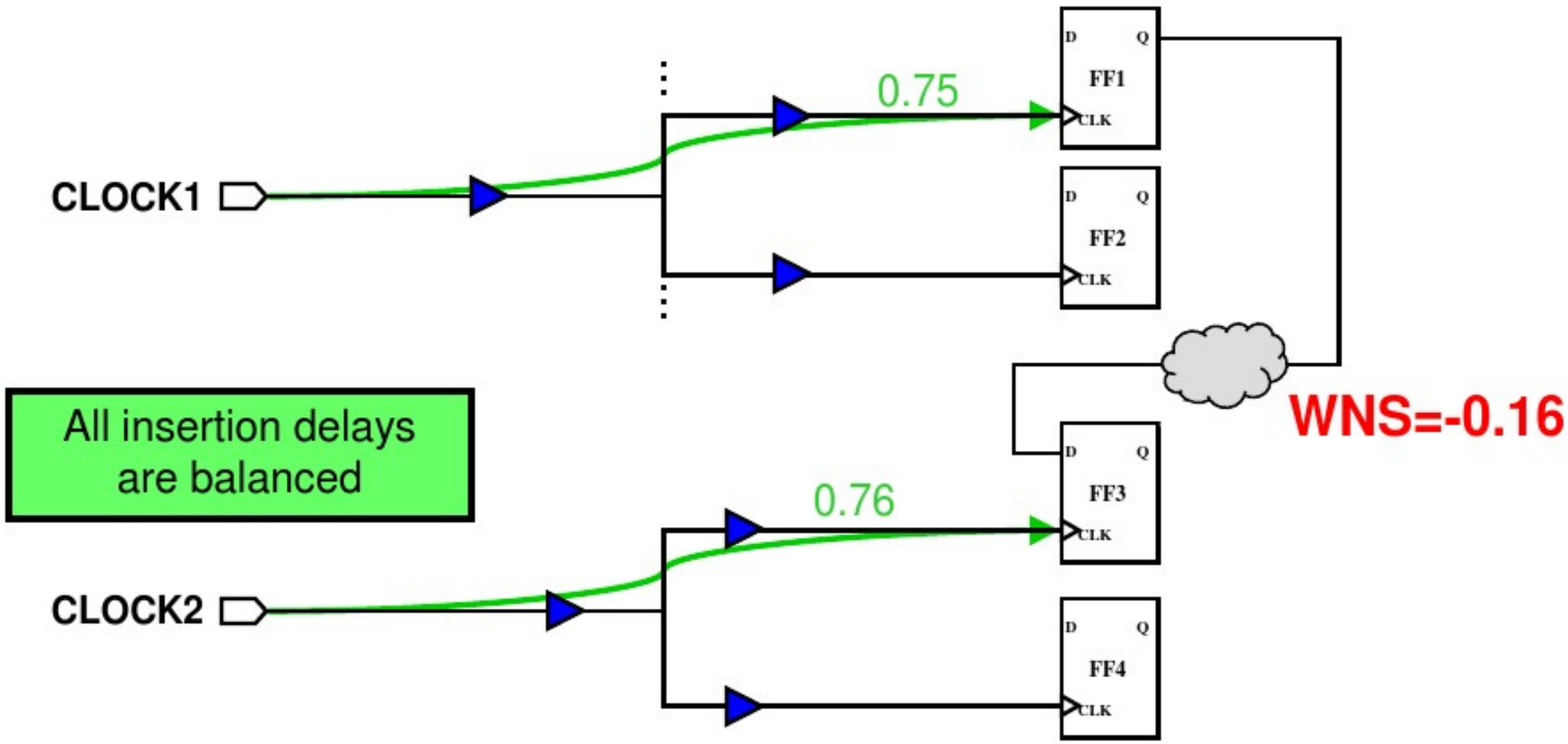
No Inter-Clock Skew Balancing by Default



By default CTS does not perform inter-clock skew balancing → May result in worse setup timing violations

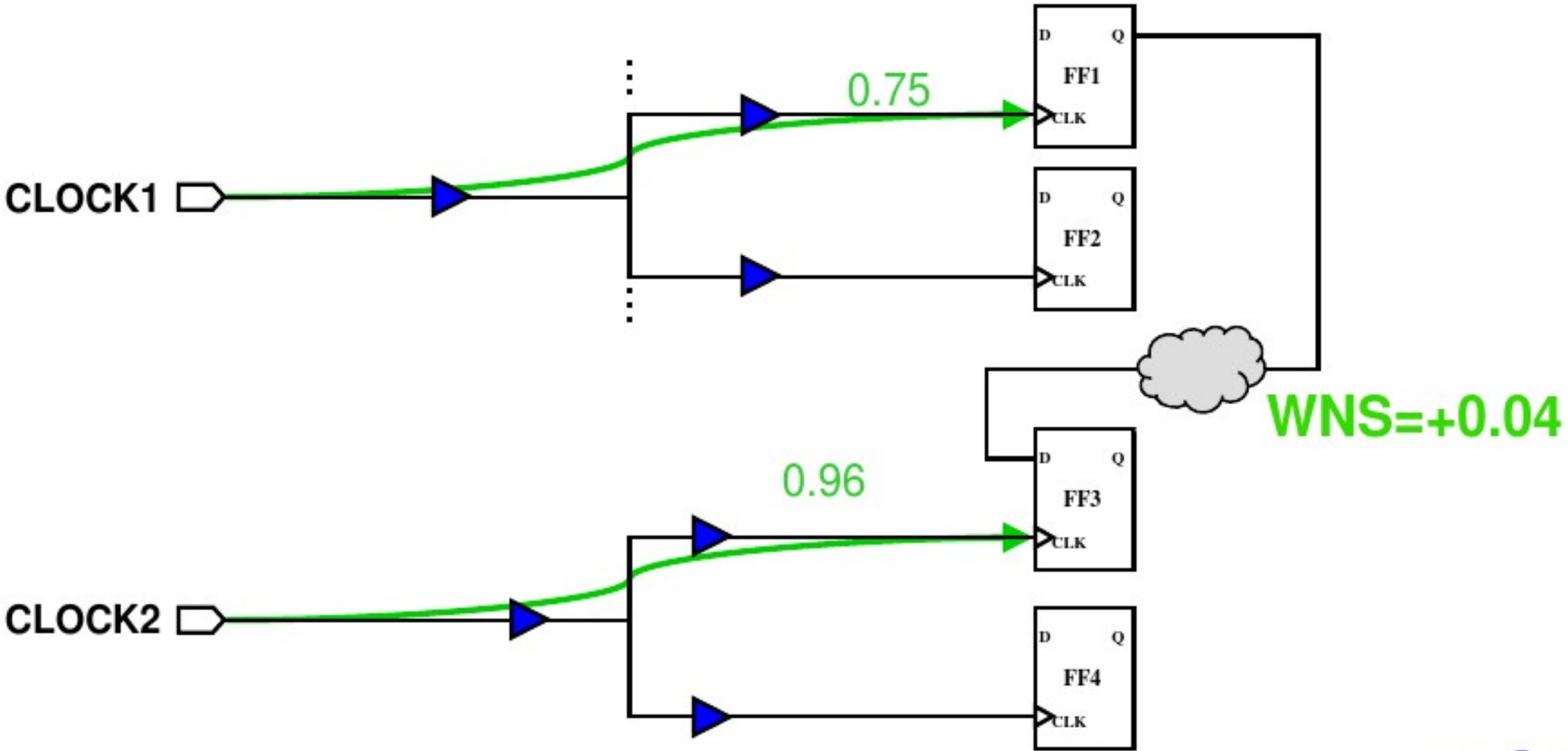
Setup for Inter-Clock Delay Balancing

```
# Balances skew between specified clocks
set_inter_clock_delay_options \
      -balance_group "CLOCK1 CLOCK2"
clock_opt -inter_clock_balance
```



Setup for Inter-Clock Delay Balancing: Offset

```
set_inter_clock_delay_options \
  -offset_from CLOCK1 -offset_to CLOCK2 \
  -delay_offset 0.2
clock_opt -inter_clock_balance
```



SDC Latencies

Does CTS respect SDC `set_clock_latency`?

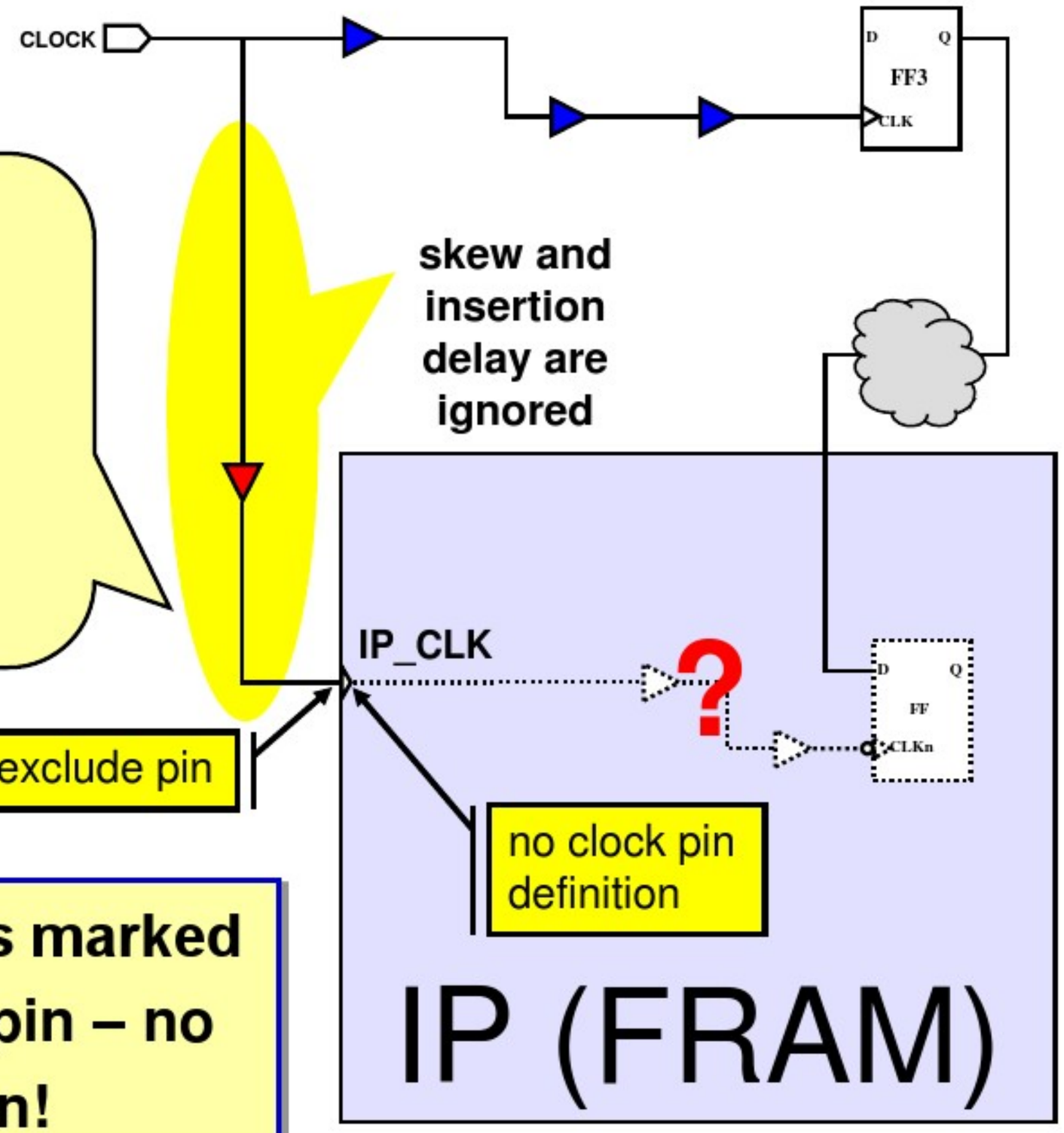
- **CTS does not respect SDC latencies by default!**
- **If you need your minimum insertion delays to match the SDC specified latencies:**

```
set_inter_clock_delay_options -honor_sdc true
clock_opt -inter_clock_balance
```

- **Note: Insertion delay will not be minimized if given SDC latency is less than initial CTS insertion delay**

User-defined or Explicit Stop Pins

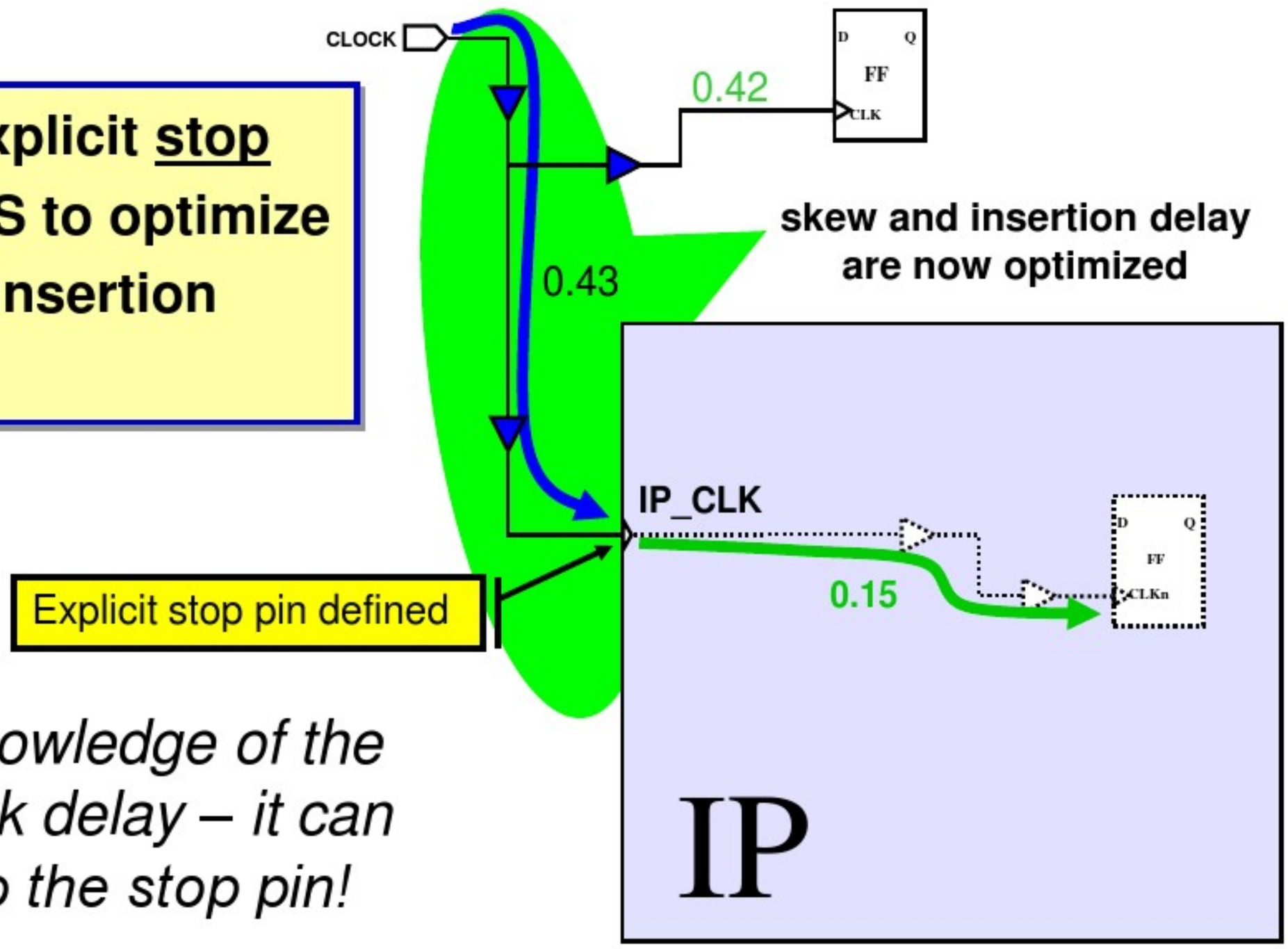
Scenario: If a macro cell clock pin is defined, CTS will treat that pin as an implicit stop pin. In this example the clock pin is not defined. What is the problem here?



The macro's clock pin is marked as an implicit exclude pin – no skew optimization!

Defining an Explicit Stop Pin

Defining an explicit <u>stop</u> pin allows CTS to optimize for skew and insertion delay targets.

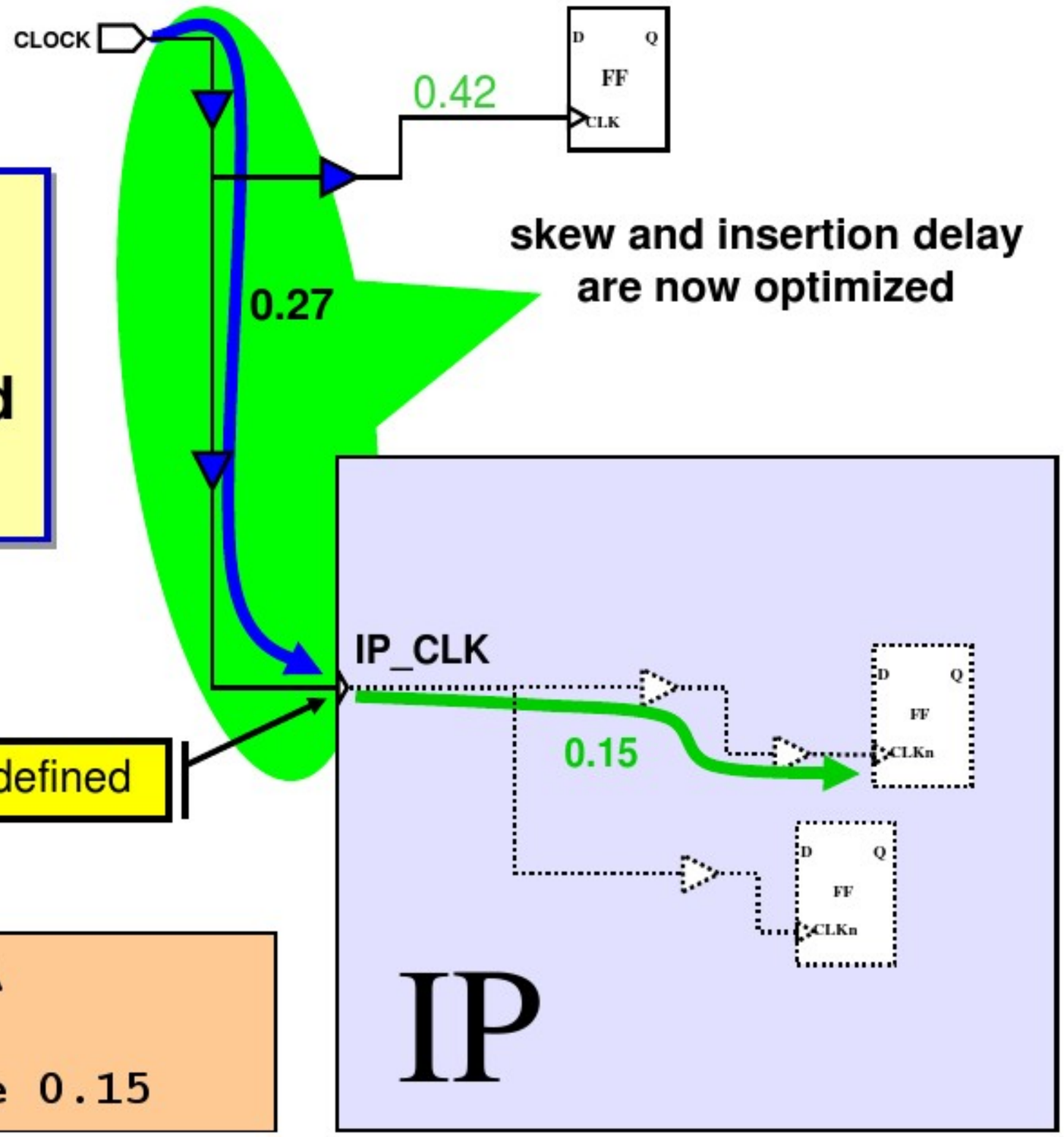


CTS has no knowledge of the IP-internal clock delay – it can only "see" up to the stop pin!

```
set_clock_tree_exceptions -stop_pins [get_pins IP/IP_CLK]
```

Defining an Explicit Float Pin

Defining an explicit <u>float</u> pin allows CTS to adjust the insertion delays based on specification.

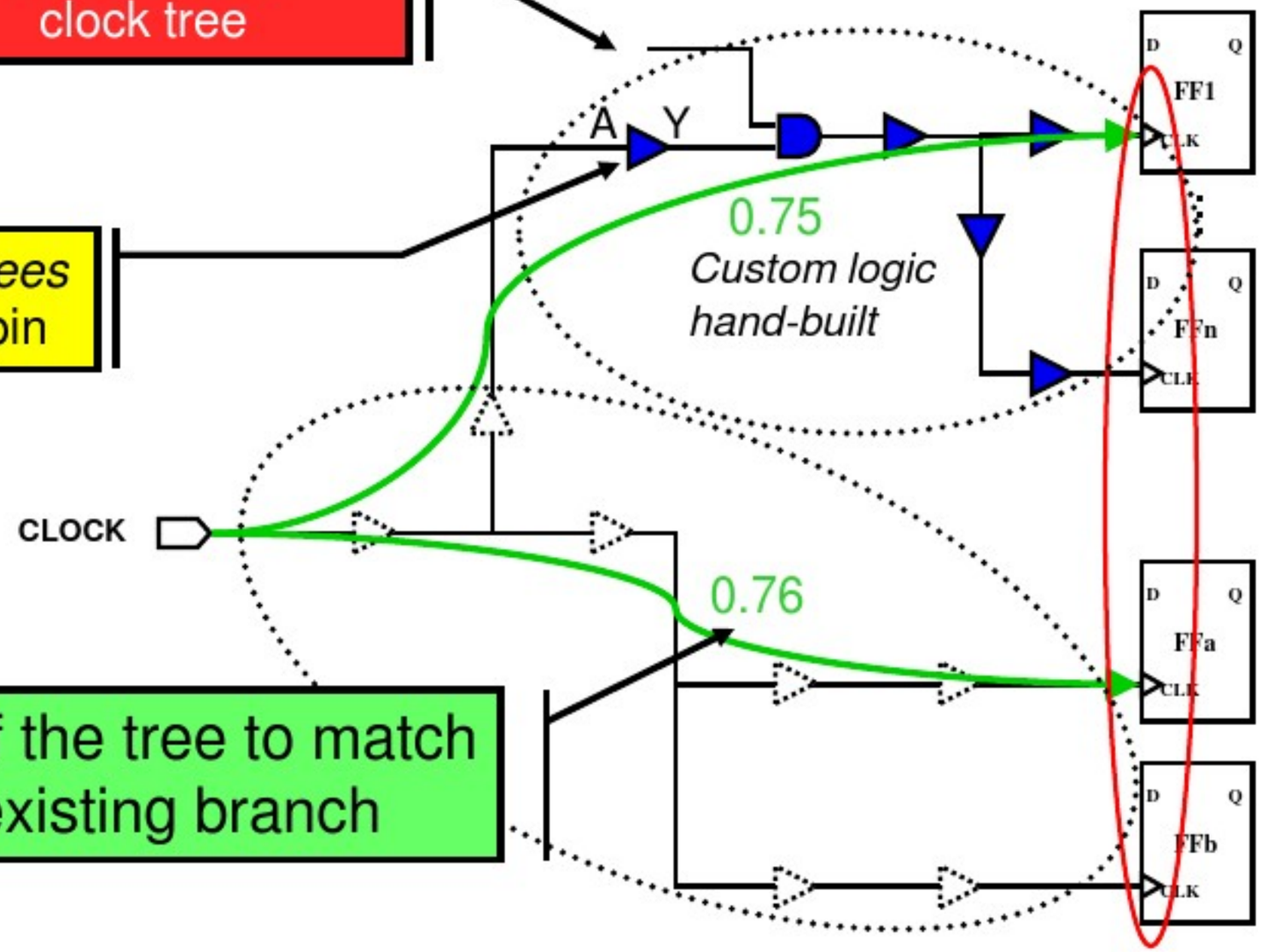



```
set_clock_tree_exceptions \
  -float_pins IP/IP_CLK \
  -float_pin_max_delay_rise 0.15
```

Preserving Pre-Existing Clock Trees

Pre-existing clock tree

Set a *dont_touch_subtrees* exception at the input pin

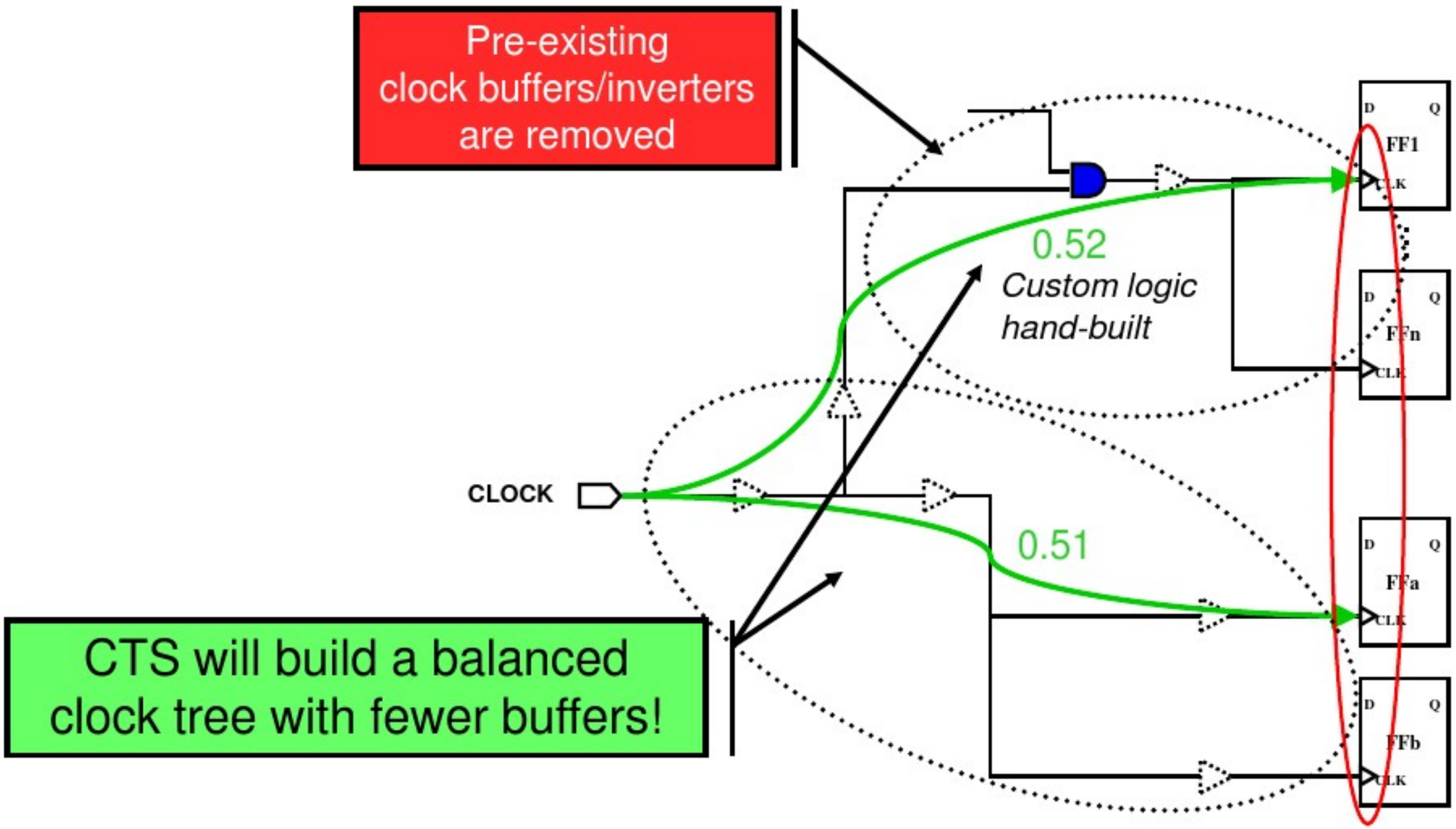


CTS will build this part of the tree to match the delay of the pre-existing branch

```
set_clock_tree_exceptions -dont_touch_subtrees buf/A
```

May create unnecessary additional buffers!

Removing Pre-Existing Clock Buffers




```
remove_clock_tree -clock_tree CLOCK
```

Specify Arnoldi-based Model on Clock Nets

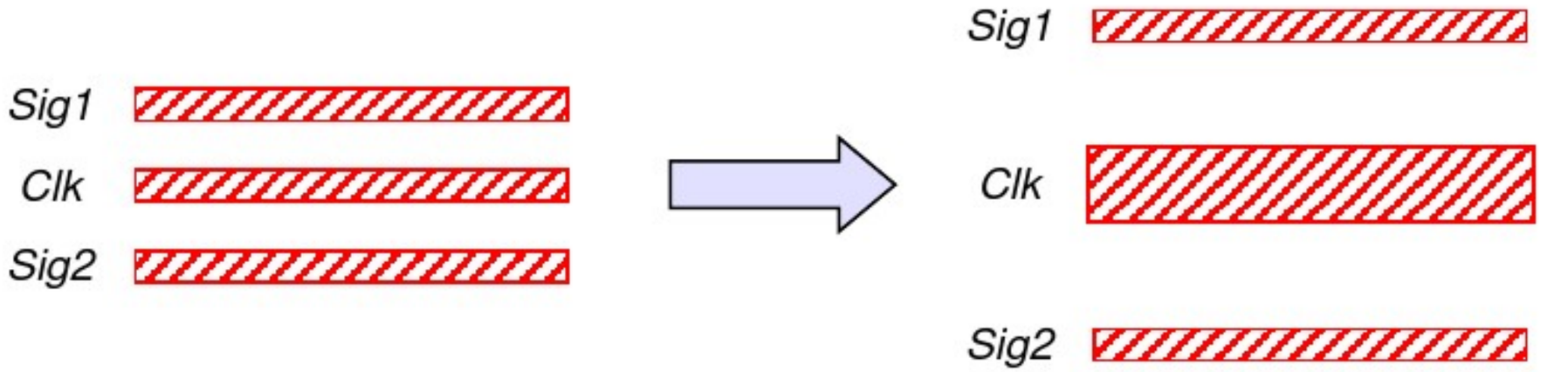
- Delays are computed via Elmore delay model by default
- Specify the delay calculation to use Arnoldi-based delay models on clock nets in a post-CTS or a post-Route design

```
set_delay_calculation -clock_arnoldi
```

Non-Default Routing Rules

NDRs

- IC Compiler 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 or EM effects



Default Routing Rule

Effect of NDR route on Clk

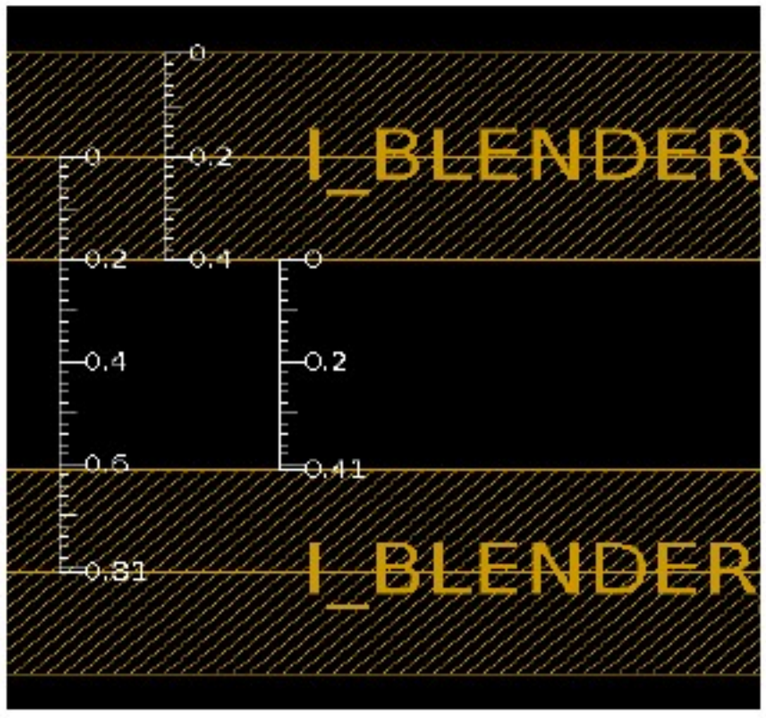
Put NDR on Pitch for Accurate RC Estimation

- Metal traces are always routed “on pitch”
- With clock NDR rules, pre-routing RC estimates of clock nets use NDR width and spacing numbers
- If NDR [spacing + width] numbers are not integer multiples of pitch (i.e. off-pitch), timing estimates pre-route may not correlate well with post-route timing

- Make sure your NDR numbers are on pitch!

```
...
Layer       "METAL5" {
            layerNumber             = 30
            maskName                = "metal5"
            pitch                   = 0.81
            defaultWidth            = 0.4
            minWidth                = 0.38
            minSpacing              = 0.34
```

For example: For double spacing and double width, do not simply double the minimum DRC numbers and use an NDR width of 0.76 and spacing of 0.68!!



Defining and Applying NDR Rule Example

- **Define the NDR rules:**

```
define_routing_rule MY_ROUTE_RULES  \
  -widths {METAL3 0.40 METAL4 0.40 METAL5 0.80} \
  -spacings {METAL3 0.42 METAL4 0.63 METAL5 0.82}
```

2x pitch – 2x defaultWidth

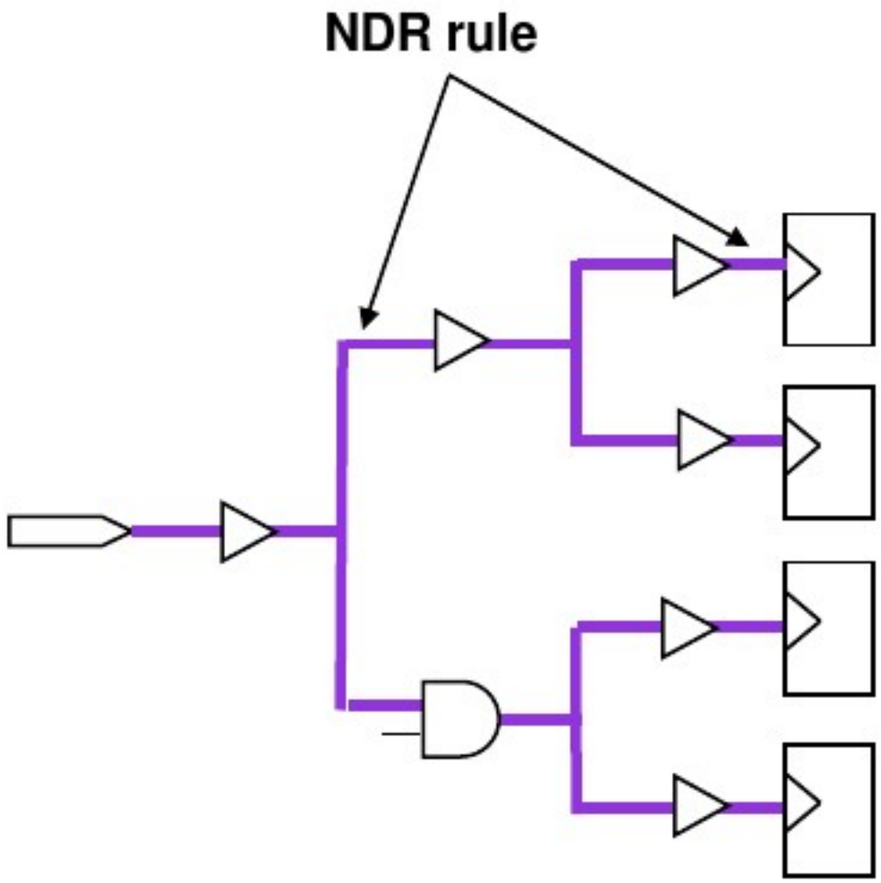
- **Assign the routing rule to clock(s):**

```
set_clock_tree_options -clock_tree clk  \
  -routing_rule MY_ROUTE_RULES  \
  -layer_list "METAL3 METAL5"
```

You may also specify the min/max layers to be used for clock tree routing

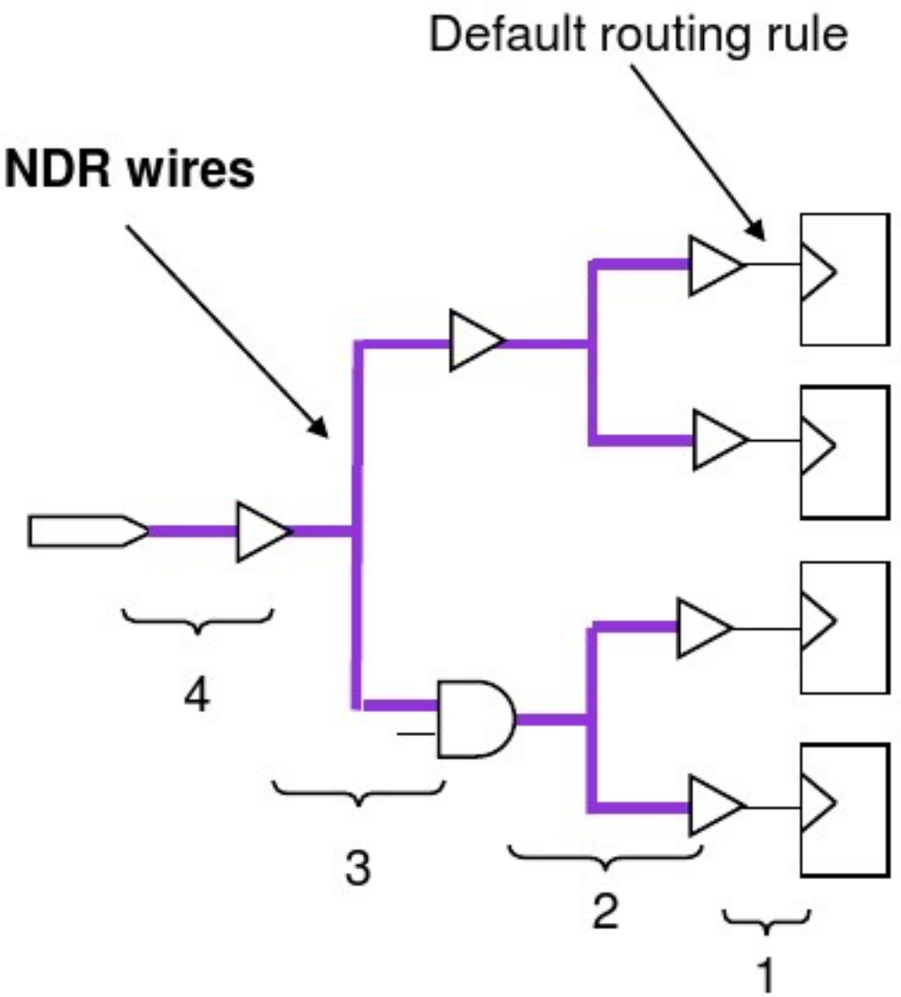
Default Routing Rule for Sink Pins Option

```
set_clock_tree_options \
  -routing_rule my_route_rule
```



Default Behavior

```
set_clock_tree_options \
  -routing_rule my_route_rule \
  -use_default_routing_for_sinks 1
```



`-use_default_routing_for_sinks`
can only be used globally!

NDR Recommendations

- **Always route clock on metal 3 and above**
- **Consider using double width to reduce resistance**
- **Consider using double spacing to reduce crosstalk**
- **Consider double via insertion to reduce resistance and improve yield**
- **Avoid NDRs on clock sinks:**

```
set_clock_tree_options -use_default_routing_for_sinks 1
```

- **Avoid NDRs on Metal 1**
  - May have trouble accessing metal 1 pins on clock buffers and gates
- **Put NDR spacing on pitch**

Test for Understanding

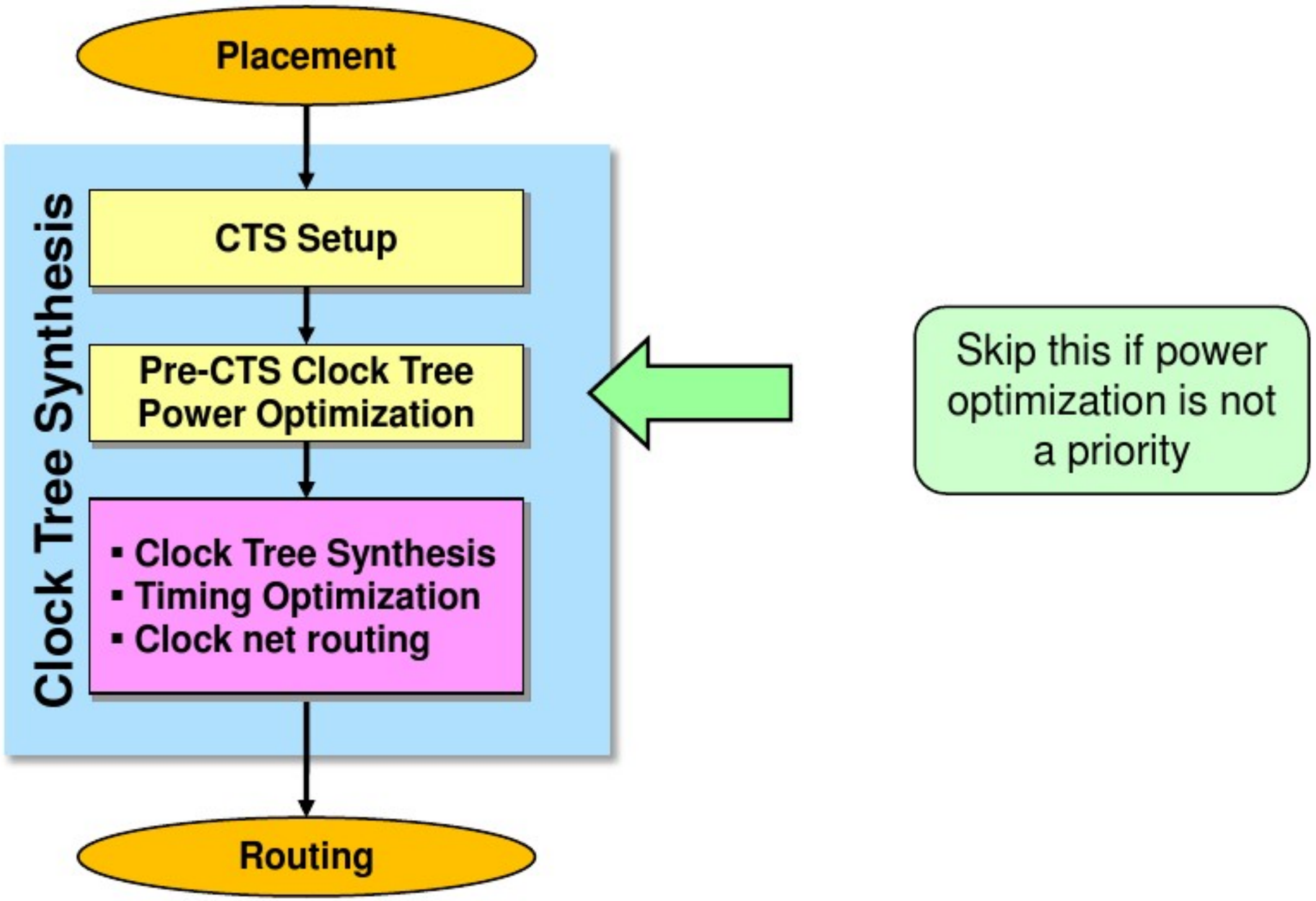
1. **What are the two main goals of CTS?**
   →
   →

2. **What is the difference between *stop* and *exclude pins*? List some examples of implicit stop/exclude pins.**

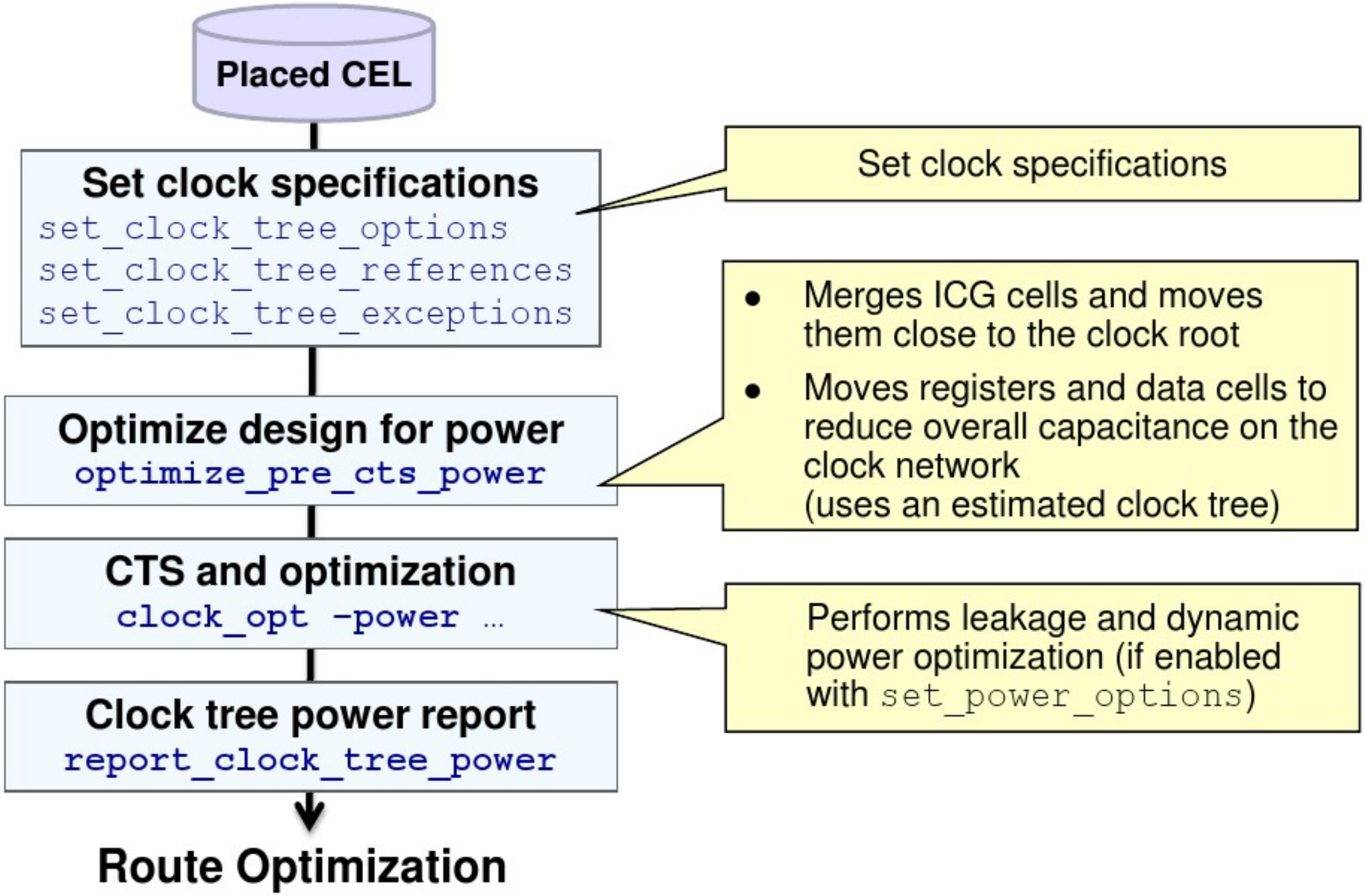
3. **How is a float pin different from a stop pin?**

10 Minute Break

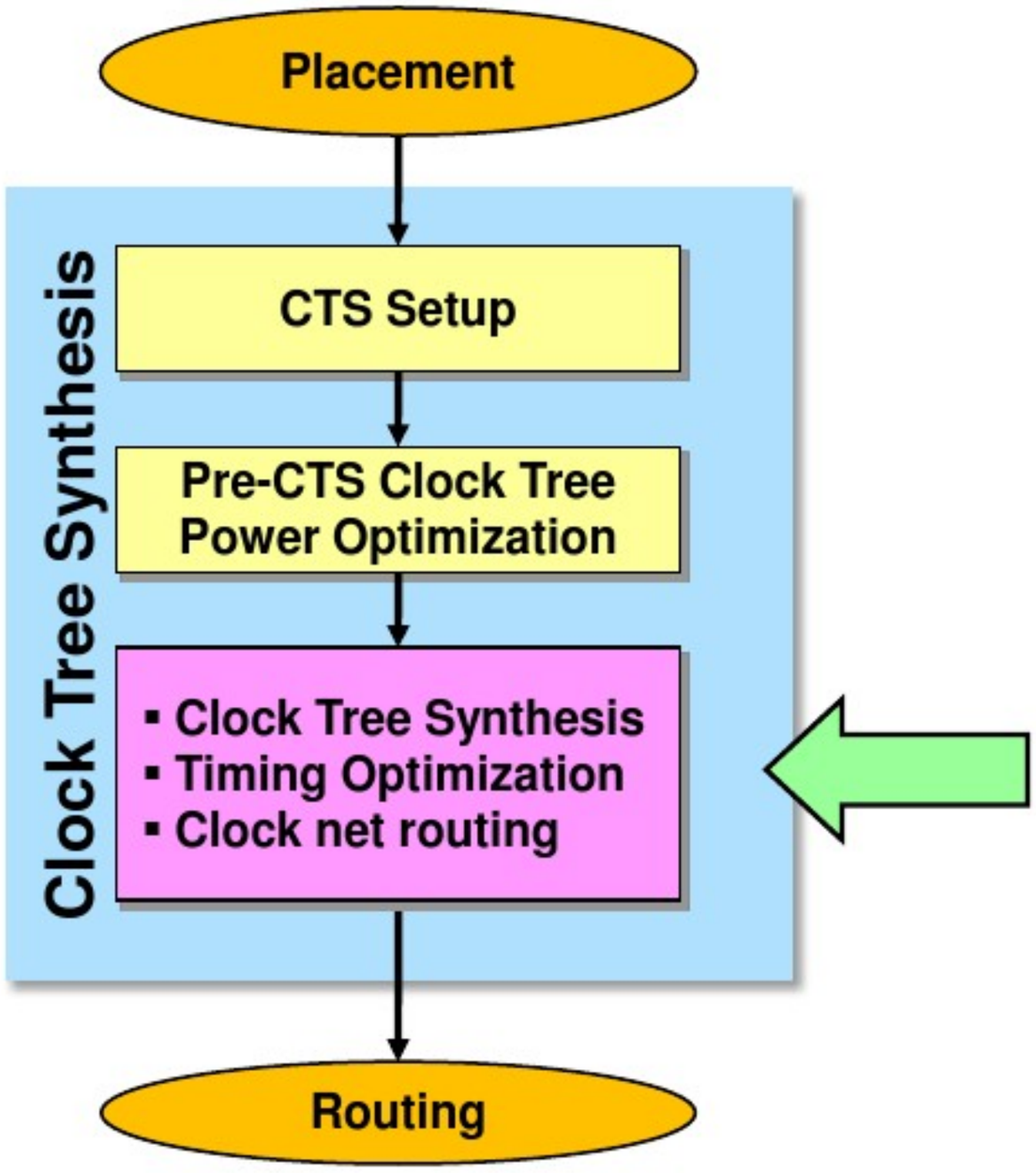
Optional Pre-CTS Power Optimization



Clock Tree Power Optimization Pre-CTS



CTS and Timing Optimization



Is the Design Ready for CTS?

- `check_physical_design -stage pre_clock_opt` **checks for:**
  - All previous placement requirements
  - Design is placed
  - Clocks have been defined
- `check_clock_tree` **checks and warns if:**
  - A generated-clock with improperly specified master-clock
  - A clock tree has no synchronous pins
  - There are multiple clocks per register
  - There is a master-clock that terminates at a pin without a corresponding generated-clock
  - …more

How Should `clock_opt` be Executed?

- **The mega command `clock_opt` performs, by default:**
  - Synthesis and balancing of individual clock tree networks
  - Timing and DRC optimization of non-clock logic
  - Routing of clock tree network
- **Additional options exist to perform:**
  - Inter-clock delay balancing
  - Scan-chain re-ordering
  - Power optimization
  - And more …
- **It is recommended to perform `clock_opt` in three steps**

```
clock_opt
    -only_cts
    -only_psyn
    -no_clock_route
    -inter_clock_balance
    -optimize_dft
    -power
    ...
```

Recommended Three-Step `clock_opt` Flow

Using `clock_opt` in the following manner allows early analysis and intervention, which can lead to increased quality of clock tree synthesis results:

```
clock_opt -no_clock_route -only_cts ...
```
analyze...

```
...
clock_opt -no_clock_route -only_psyn ...
```
analyze...

```
route_zrt_group -all_clock_nets ...
```

clock_opt Functionality

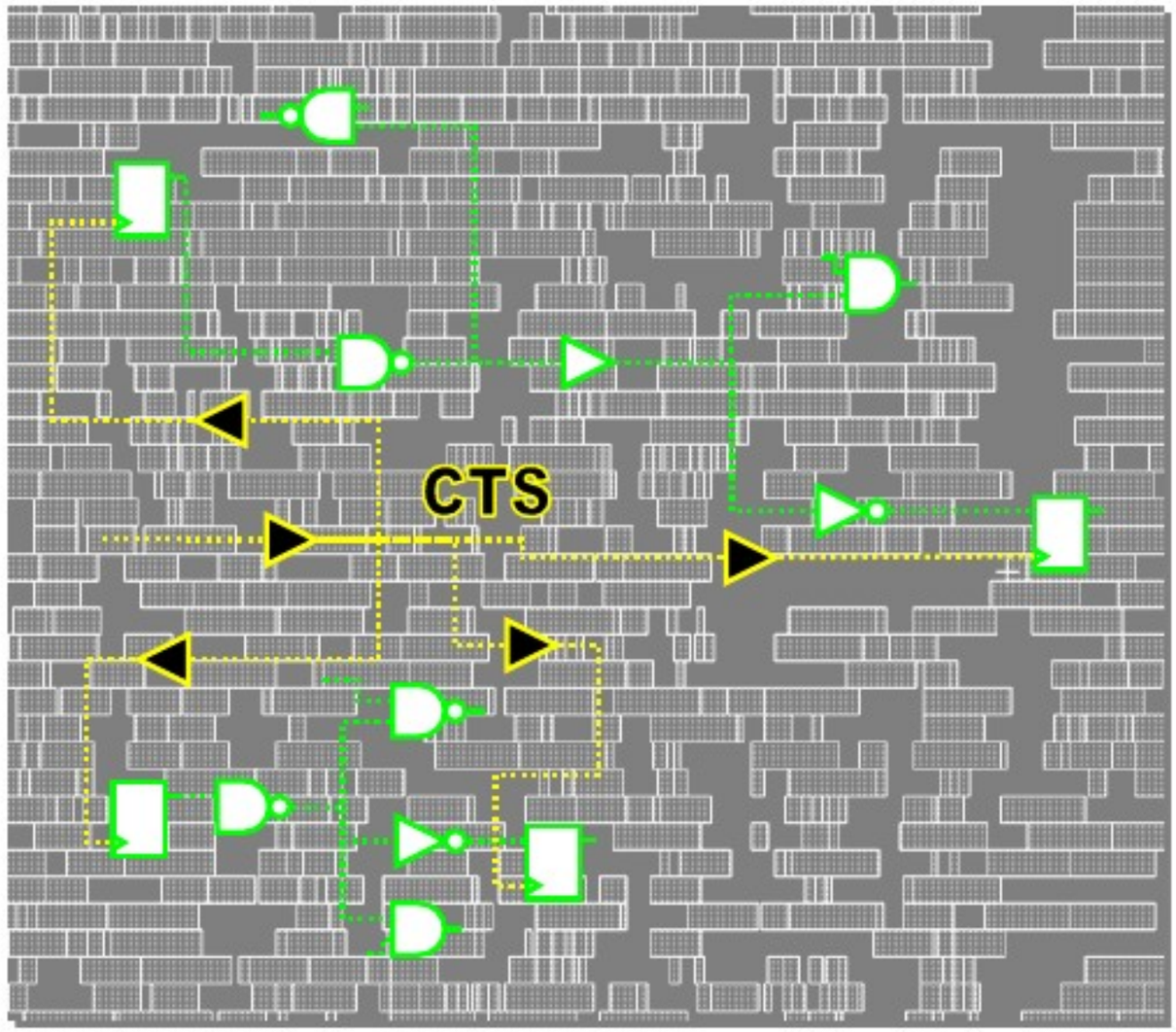
	clock_opt	clock_opt \ -only_cts \ -no_clock_route	clock_opt \ -only_psyn \ -no_clock_route	route_zrt_group -all_clock_nets
Clock Tree Synthesis Builds initial clock tree by: - Load balancing - Logic-level balancing	✓	✓		
Clock Tree Optimization* Meets clock tree targets by: - Cell sizing - Delay insertion	✓	✓		
Timing/DRC optimization - Optimizes logic and placement of data cells - Clock cells are fixed	✓		✓	
Routing of Clock Nets	✓			✓

Clock buffers and inverters are fixed (locked down) after `clock_opt -only_cts`

Effects of Clock Tree Synthesis

```
clock_opt -no_clock_route -only_cts -inter_clock_balance
```

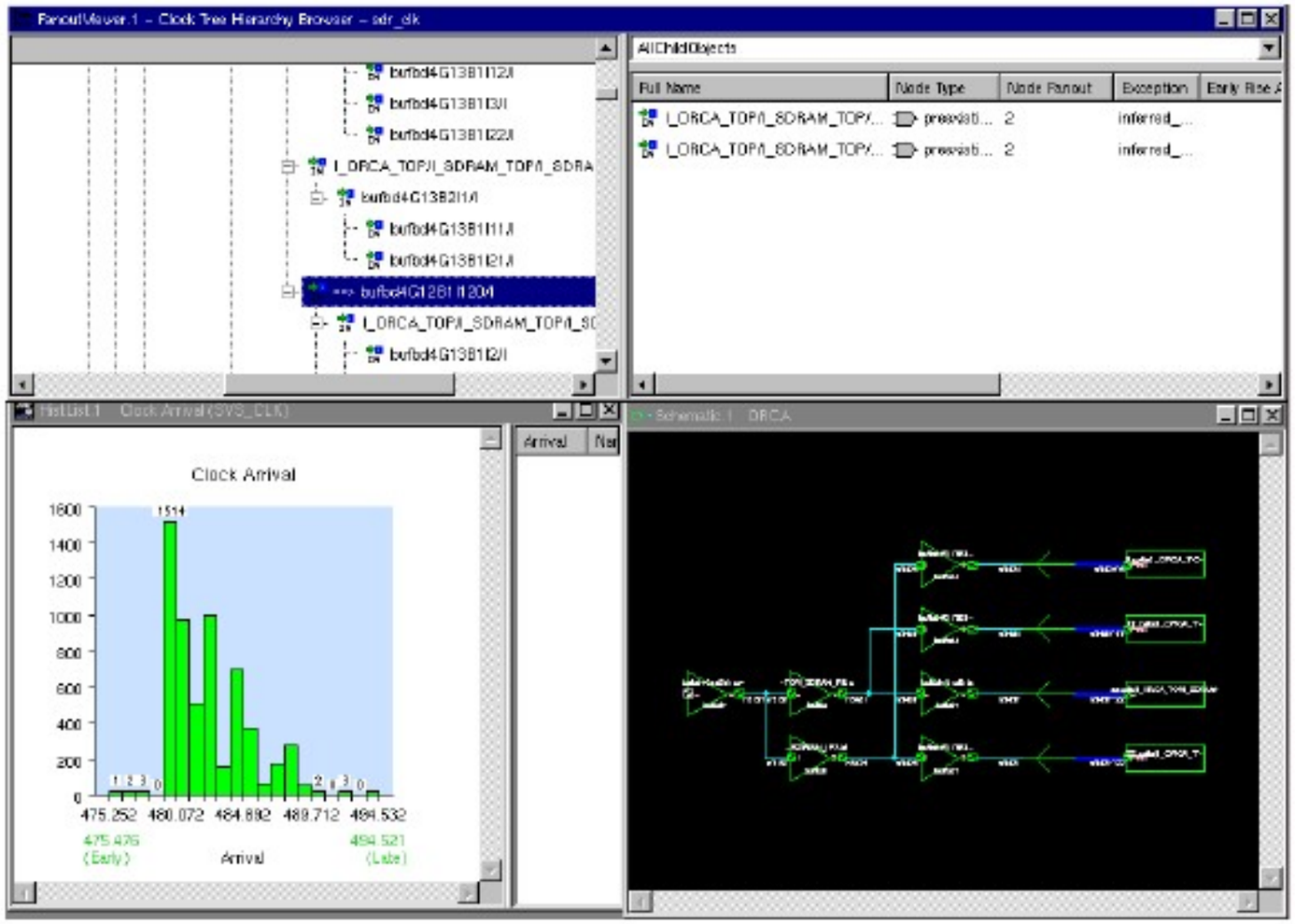
- Builds the clock tree – lots of buffers are added!
- Congestion may increase
- Non clock cells may have been moved to less ideal locations
- Can introduce new timing and max tran/cap violations to non clock paths



? How do you analyze clock trees?

Analysis Using the CTS GUI

- **CTS browser**
  - Identify clock tree object properties and attributes
  - Traverse clock tree levels
- **CTS schematic**
  - Trace clock paths
  - Cross-probe with the layout view
- **Clock arrival histogram**
  - Analyze insertion delays and maximum skew
  - X-axis shows delays of clock paths
  - Y-axis shows the # of paths



Analyzing CTS Results

Details covered in the lab!

- `report_clock_tree`
  `-summary`
  `-settings`
  `-...`
  - Reports Max global skew, Late/Early insertion delay, Number of levels in clock tree, Number of clock tree references (Buffers), Clock DRC violations
- `report_clock_timing`
  - Reports actual, relevant skew, latency, inter-clock latency, etc. for paths that are related.
  - Example: `report_clock_timing -type skew`

How do you handle timing violations?

Timing/Power/Scan/Congestion Optimiz'n

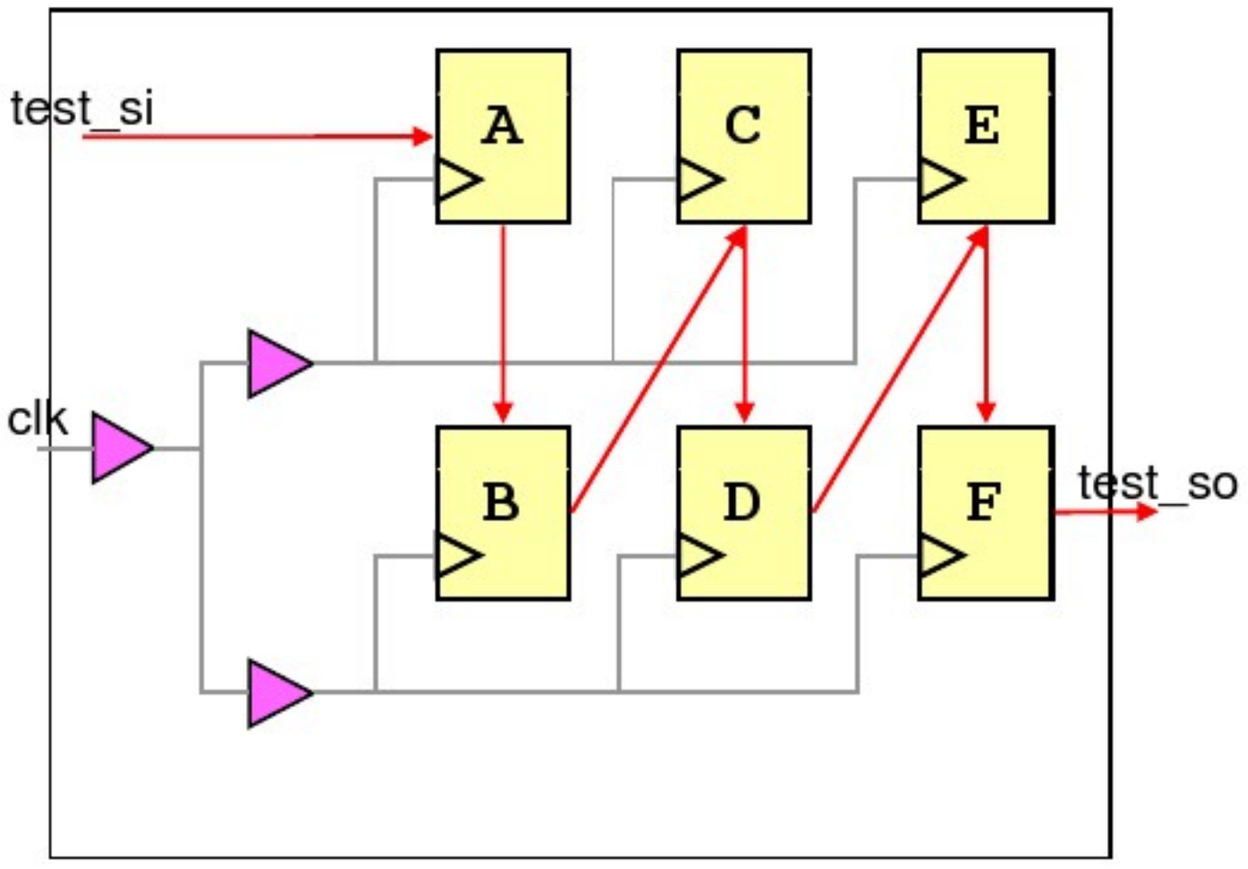
```
clock_opt -no_clock_route -only_psyn \
          -optimize_dft -area_recovery -power
```

- **Performs logic and placement optimization of non-clock paths to fix setup timing and max tran/cap (DRC) violations**
  - Tries to minimize cell disturbances
- **Can also perform:**
  - Scan chain re-ordering (`-optimize_dft`)
  - Congestion reduction (`-area_recovery`)
  - Power optimization (`-power`)

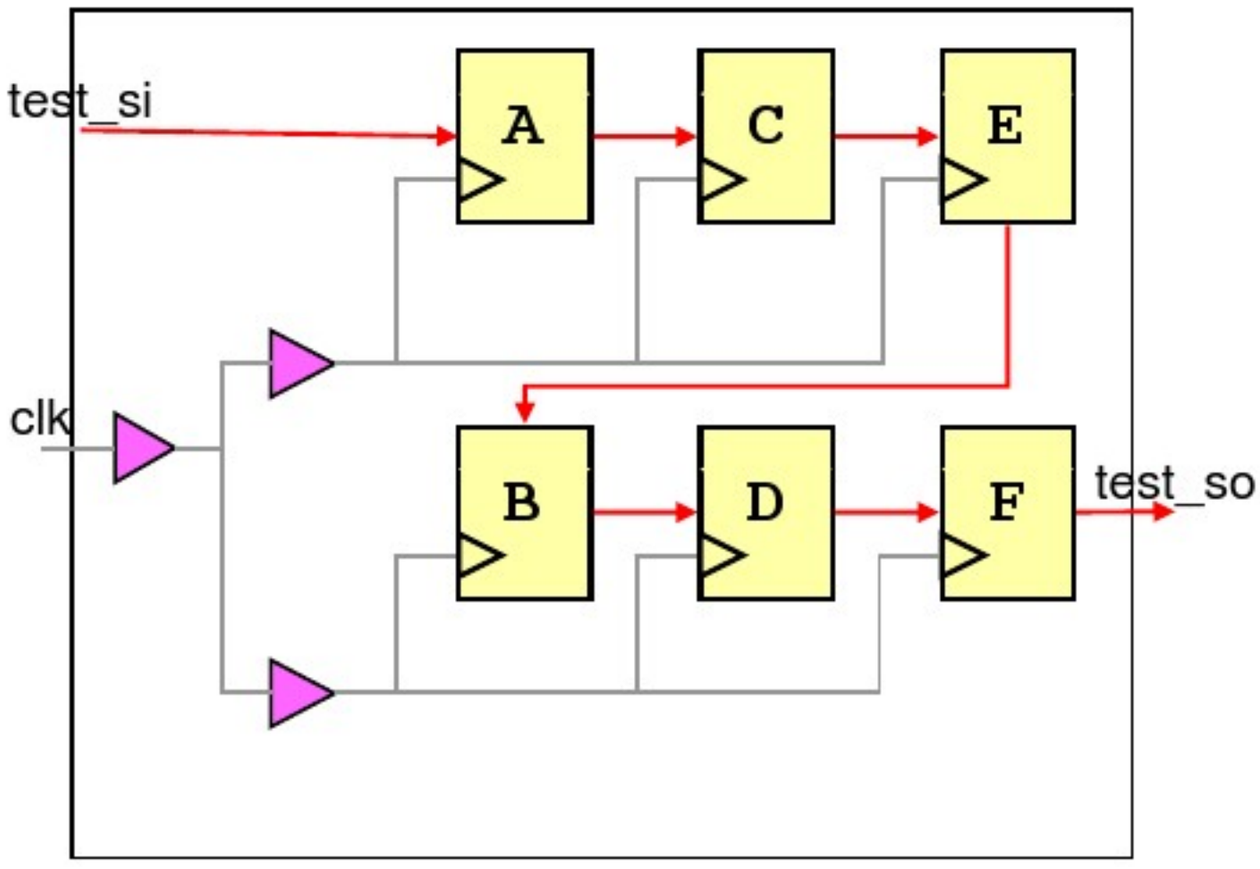
Minimize Hold Time Violations in Scan Paths

```
clock_opt ... -optimize_dft ...
```

- Reorders scan chains to minimize crossings between clock buffers
- Can reduce hold time violations in the scan chain



Without clock tree based reordering



With clock tree based reordering

Enable Hold Time Fixing

- No hold time fixing has been performed on the design up to this point
- Now that the clock tree has been synthesized it is generally considered good practice to enable hold time fixing:

```
set_fix_hold [all_clocks]
extract_rc
clock_opt -no_clock_route -only_psyn \
   -optimize_dft -area_recovery -power
```

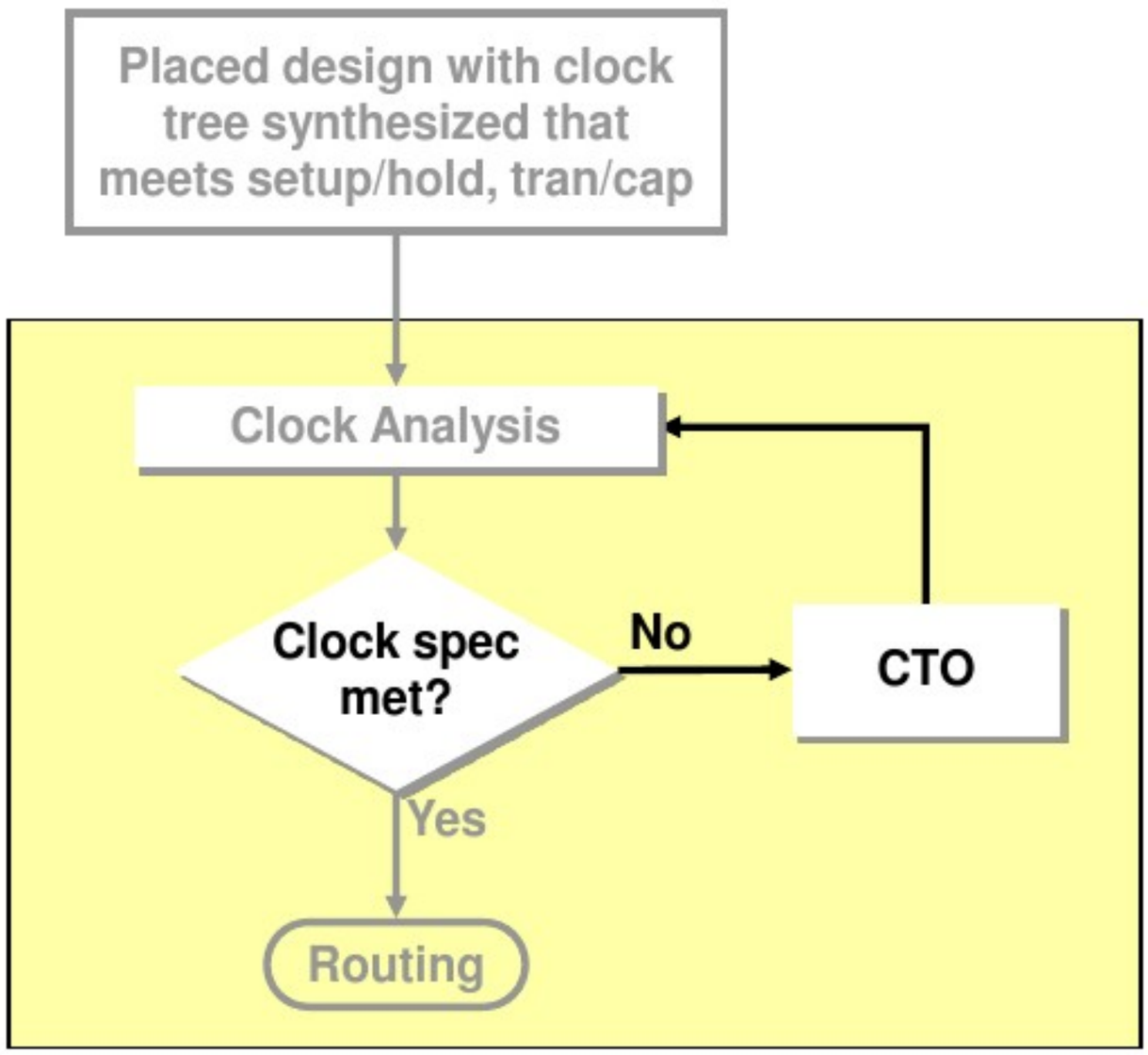
Force global route instead of virtual route extraction for timing optimization during the next step

Example CTS Flow Summary

```
define_routing_rule
set_clock_tree_options ...
set_clock_tree_exceptions ...
set_clock_tree_references ...
set_inter_clock_delay_options
remove_clock_tree ...
remove_clock_uncertainty [all_clocks]; # OR adjust uncertainty
check_physical_design ...
check_clock_tree
optimize_pre_cts_power
set_delay_calculation -clock_arnoldi
clock_opt -no_clock_route -only_cts -inter_clock_balance
report_clock_tree
report_clock_timing
set_fix_hold [all_clocks]
extract_rc
clock_opt -no_clock_route -only_psyn -optimize_dft \
          -area_recovery -power
report_clock_tree
report_clock_timing
report_clock_tree_power
```

Stand-Alone Clock Tree Optimization (CTO)

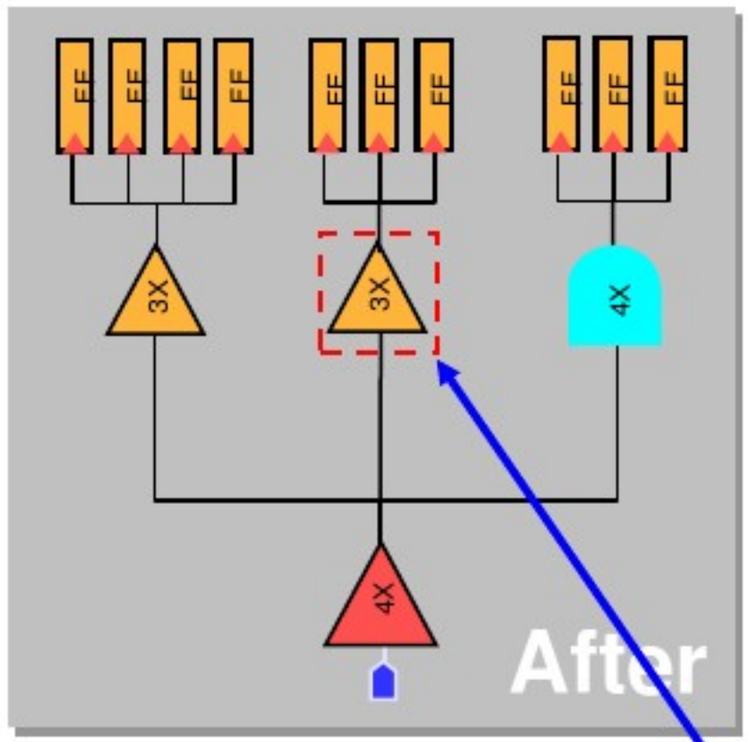
Perform additional Clock Tree Optimization as necessary to further improve clock skew after clock tree synthesis and timing optimization

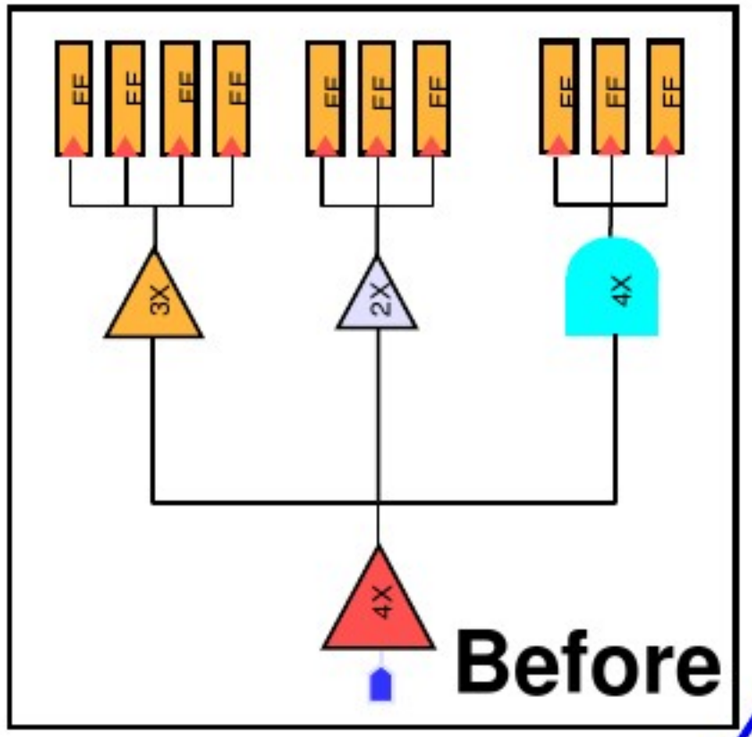


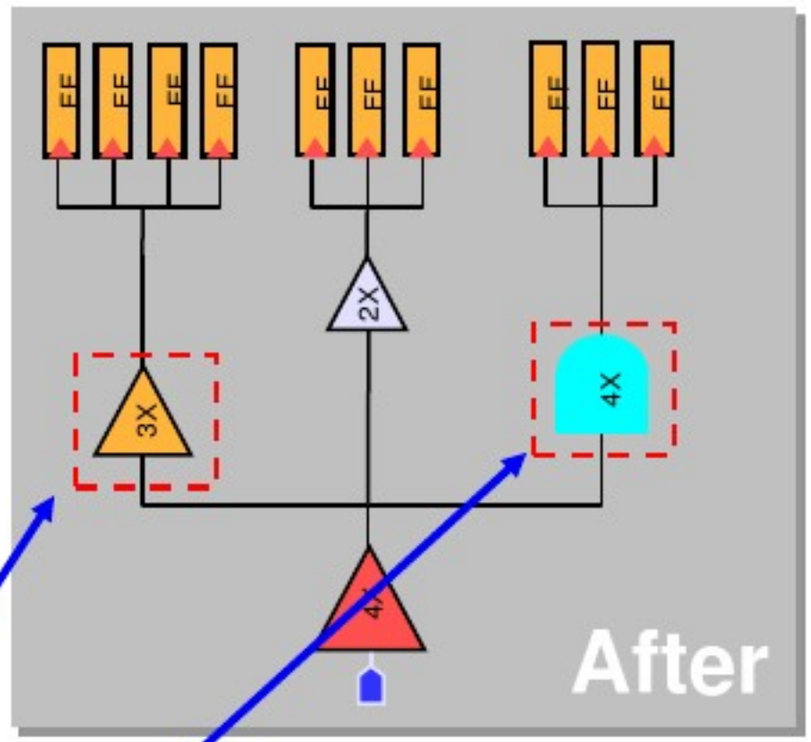
CTO is run inside `clock_opt`, and can be run independently as well:

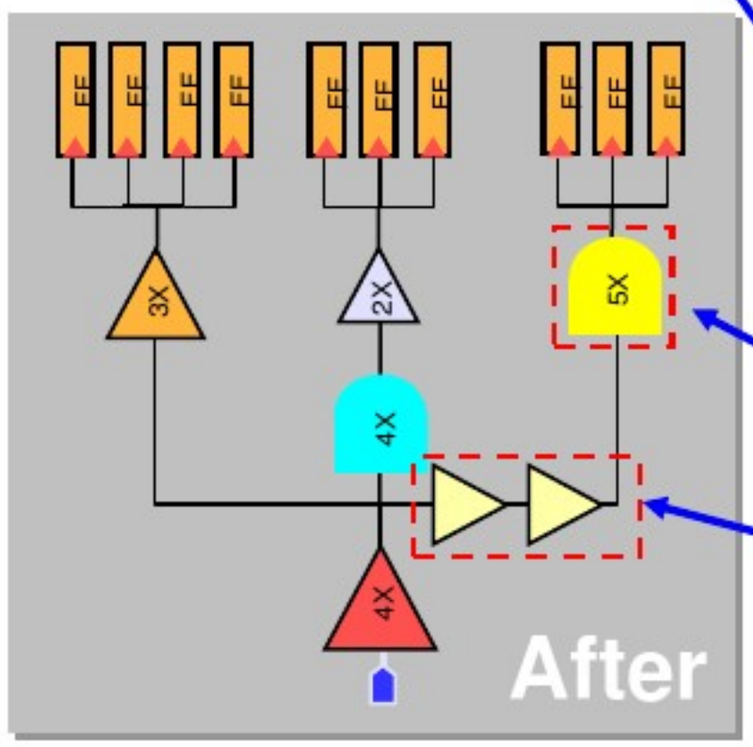
```
optimize_clock_tree
```

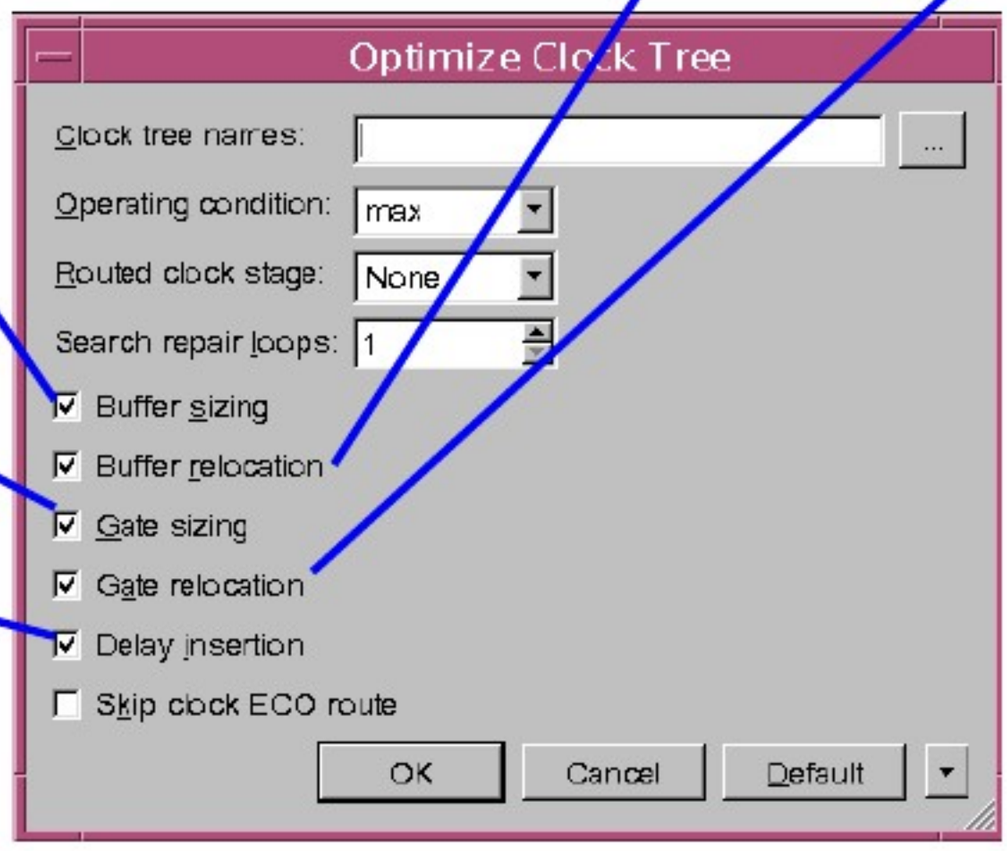
Stand-Alone Clock Tree Optimization Control











Test for Understanding

1. **CTS tries to:**
   a) Minimize skew only
   b) Minimize skew and insertion delay
   c) Minimize skew and maximize insertion delay
   d) Minimize skew and meet minimum insertion delay target
2. **How do you set a clock skew target of 0.1 for clk1, and a minimum insertion delay of 0.7 for clk2? What is the skew target for clk2?**
3. **Write the command(s) to balance the two clock trees clk1 and clk2, so clk2 arrives 0.3 earlier!**
4. **Why is it important to remove or adjust the clock uncertainty *before* executing `clock_opt`?**

Unit Objectives Summary

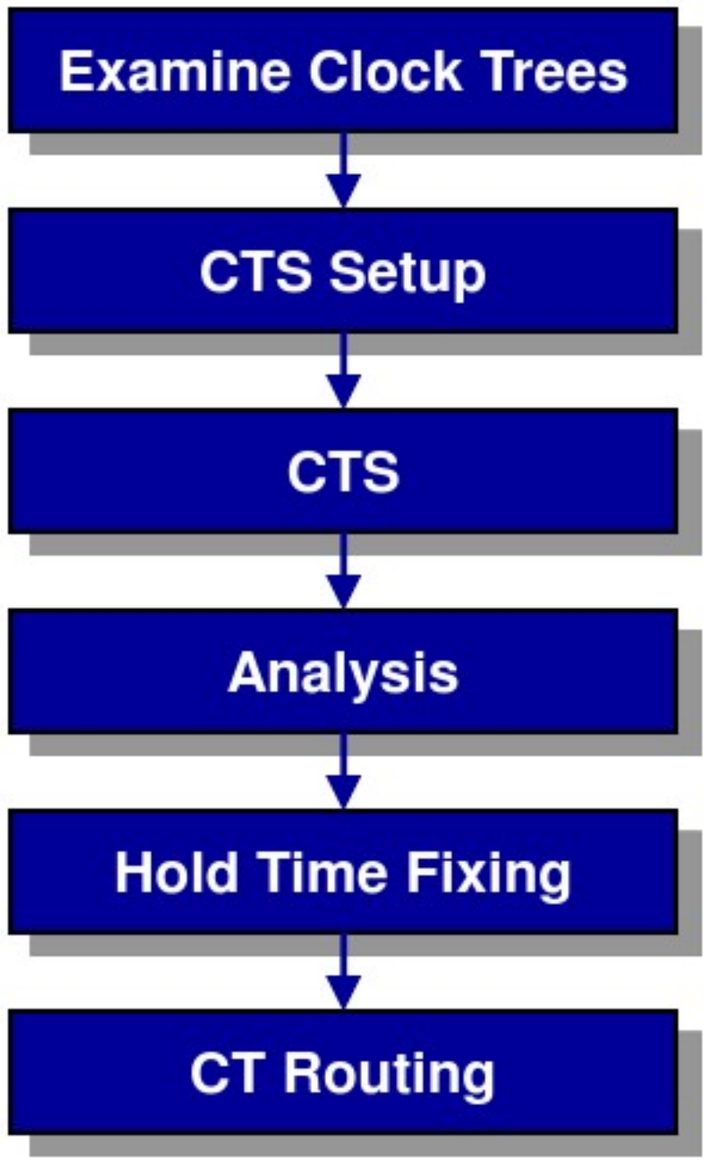
You should now be able to:

- List the status of the design prior to CTS
- Set up the design for clock tree synthesis
- Identify implicit clock tree start/end points and when explicit modifications are needed
- Control the constraints and targets used by CTS
- Execute the recommended clock tree synthesis and optimization flow
- Analyze timing and clock specifications post CTS

Lab 4: Clock Tree Synthesis

75 minutes

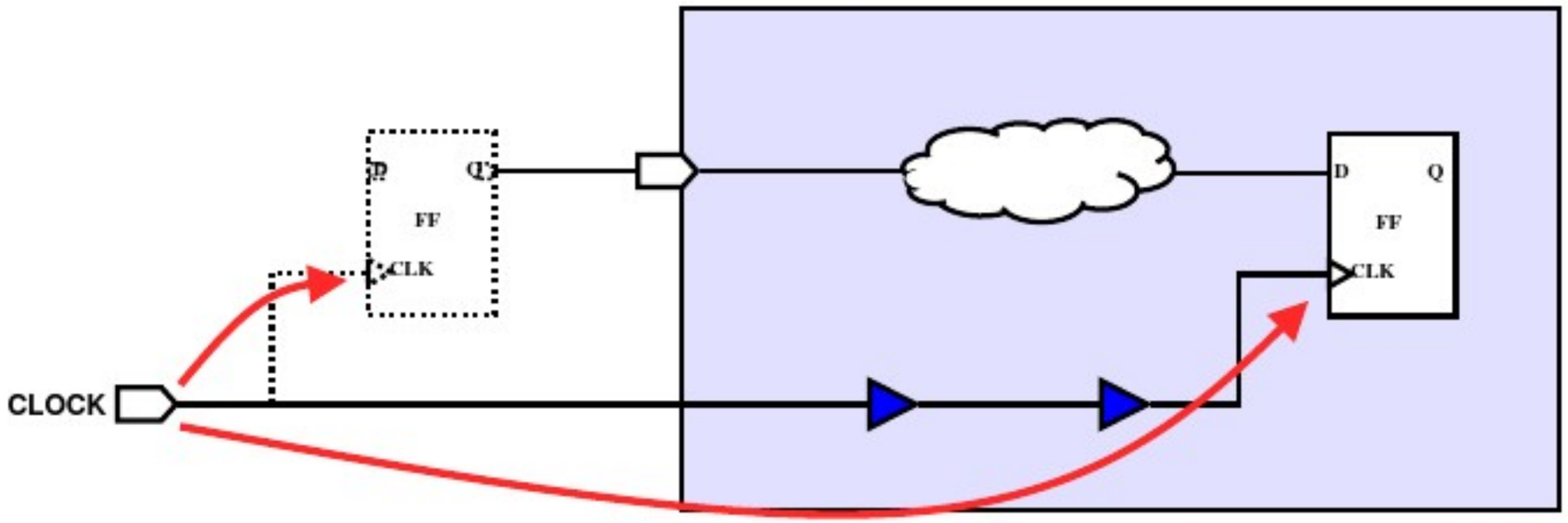
Perform CTS and Optimizations on the ORCA Design.



Appendix A

Automatic IO Latency Calculation

IO Latency Auto Update



- **Once clock trees are built, delays are propagated:**
  - Latency of the “real” clock path is calculated (lower arrow in diagram)
- **Under normal circumstances, the IOs - or more precisely the flipflops driving/capturing the IOs - have zero latency, unless it is**
  - Specified as source latency in the SDC
  - Included in input delay
- **IC Compiler can compute the median insertion delay and apply it as clock latency on the IO paths** (upper arrow in diagram)

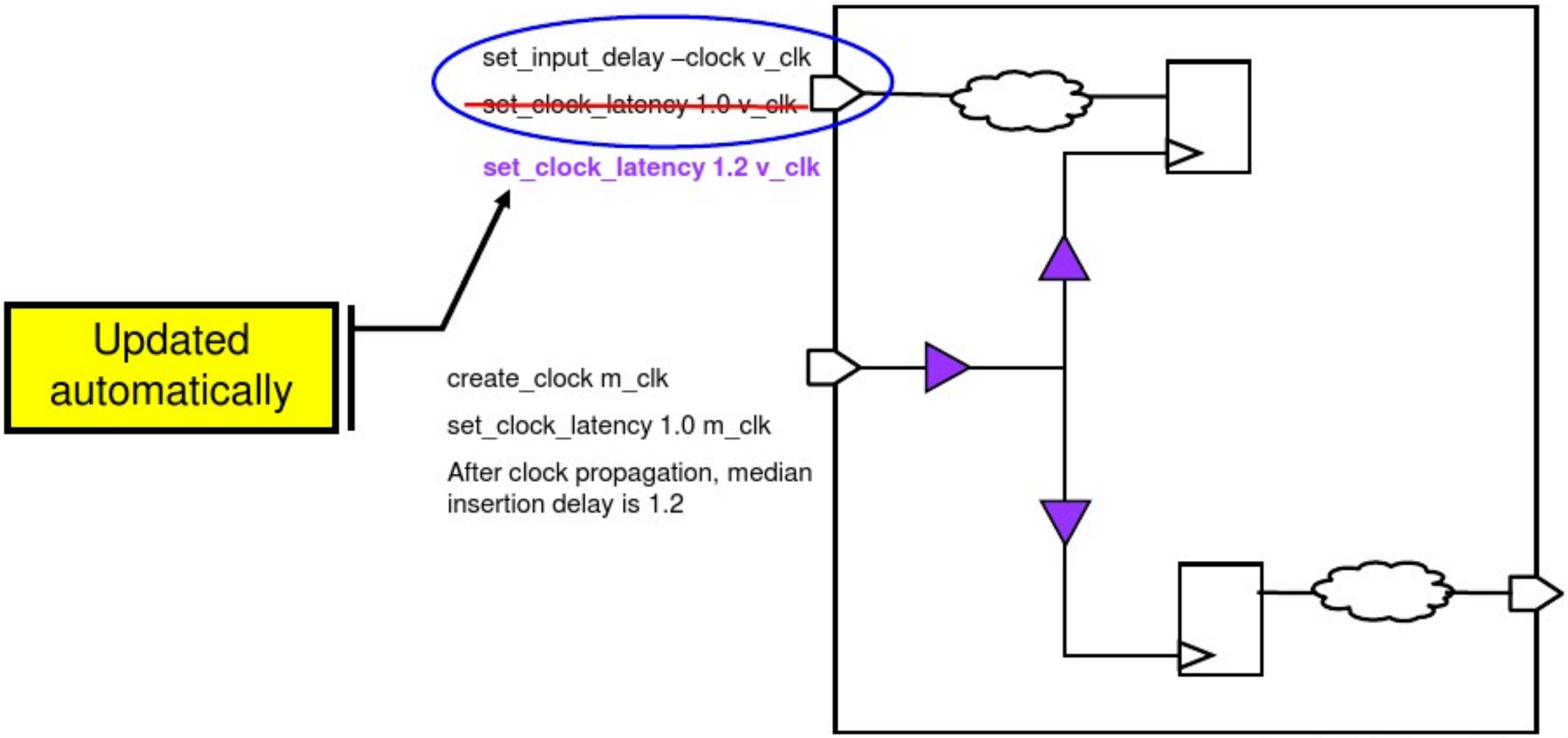
```
clock_opt -update_clock_latency
```
Or after CTS: `update_clock_latency`

Auto Update with Virtual Clocks

- Latency can also be updated for ports that were constrained using a virtual clock. First, define the related virtual clock:

```
set_latency_adjustment_options \
   -from_clock m_clk -to_clock v_clk
```

- Second, use **update_clock_latency**

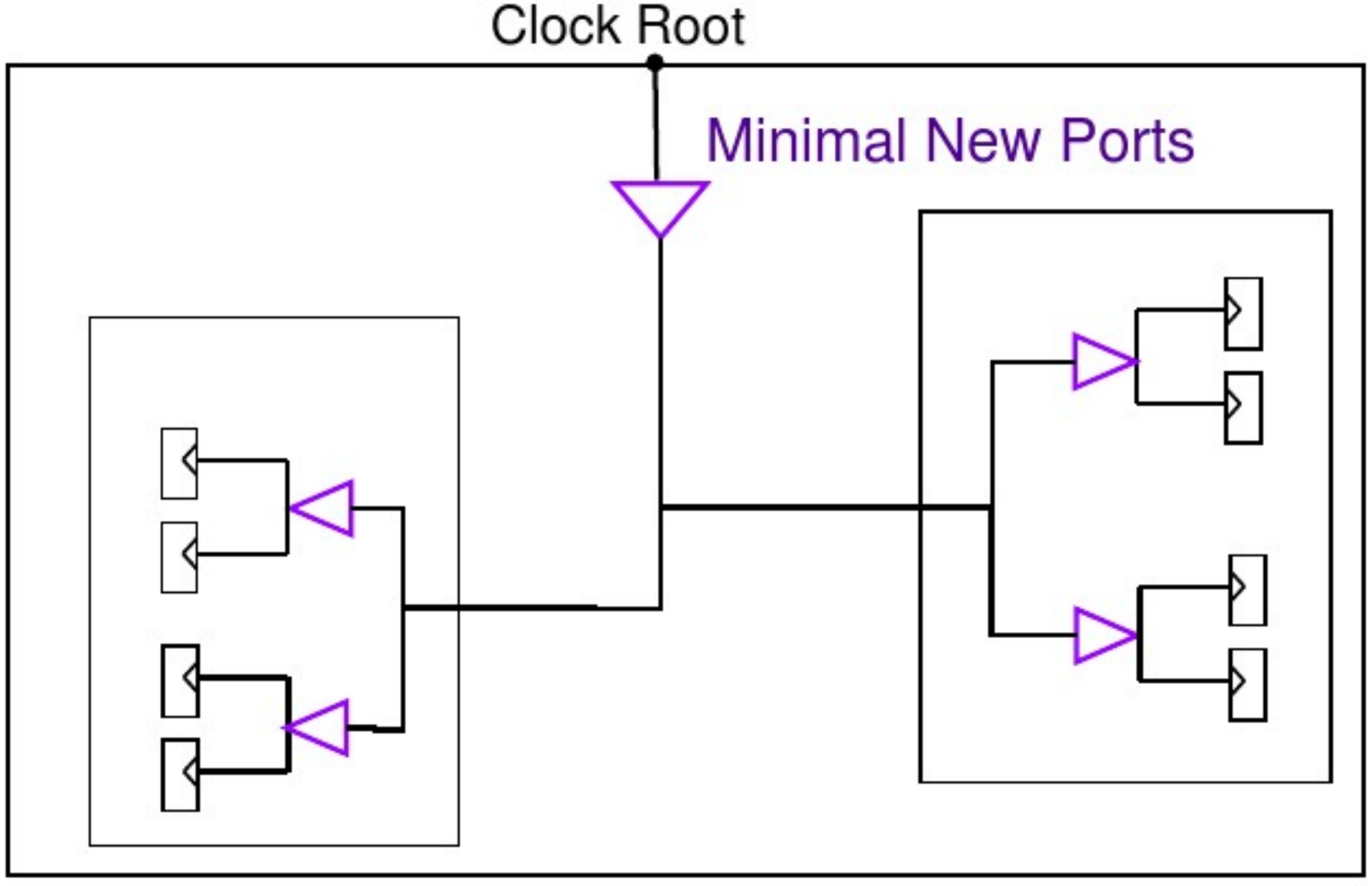


Appendix B

CTS with Logical Hierarchy

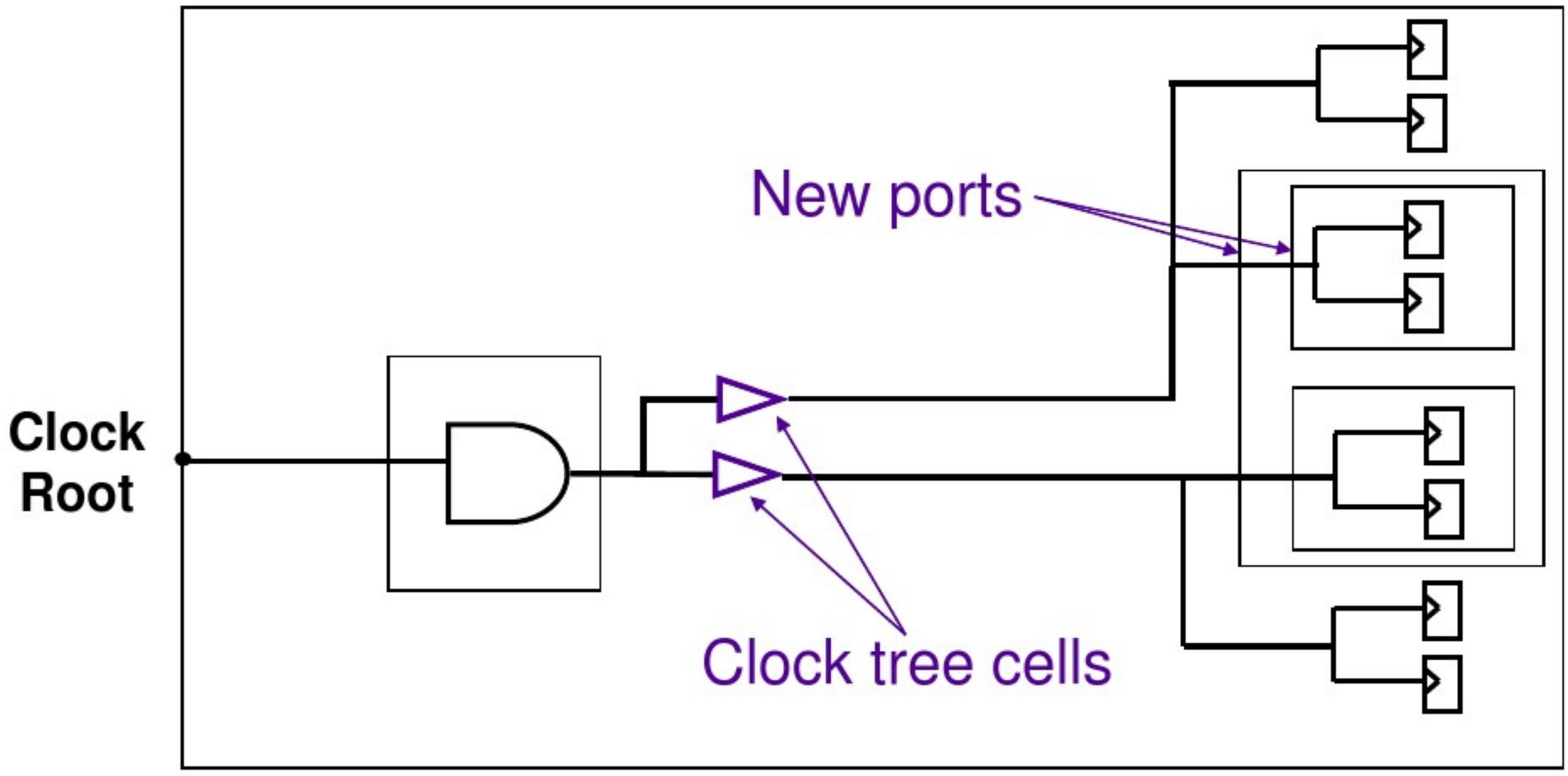
CTS with Logical Hierarchy

- **Clock tree cells are added at the lowest level of common hierarchy**
- **Ports are added as necessary for the sub-modules**
  - Minimal number of new ports created



Clock Tree Cells Added in Top Hier

- New clock tree cells are added at the lowest common hierarchy of the sinks and new ports are created to connect to them (if required)



Appendix C

Clock tree configuration control

Clock Tree Configuration Control

- **Provides additional flexibility to control the results of clock tree synthesis through a predefined clock tree structure (per net basis)**

```
compile_clock_tree \
   -config_file_read <file_name> \
   -config_file_write <file_name>
```

- **Flexibility for providing Hard or Soft clock tree configuration**
- **Embedded CTO operations can change the clock tree reference, but will retain the clock tree structure**

Clock Tree Configuration Syntax

- **Hard Configuration**

```
begin_clock_tree 6
clock_net core/clk
buffer_level bufx6 2
buffer_level invx12 4
buffer_level bufx12 7
buffer_level invx12 11
buffer_level bufx12 48
buffer_level bufx12 268
end_clock_tree
```

- **Soft Configuration**

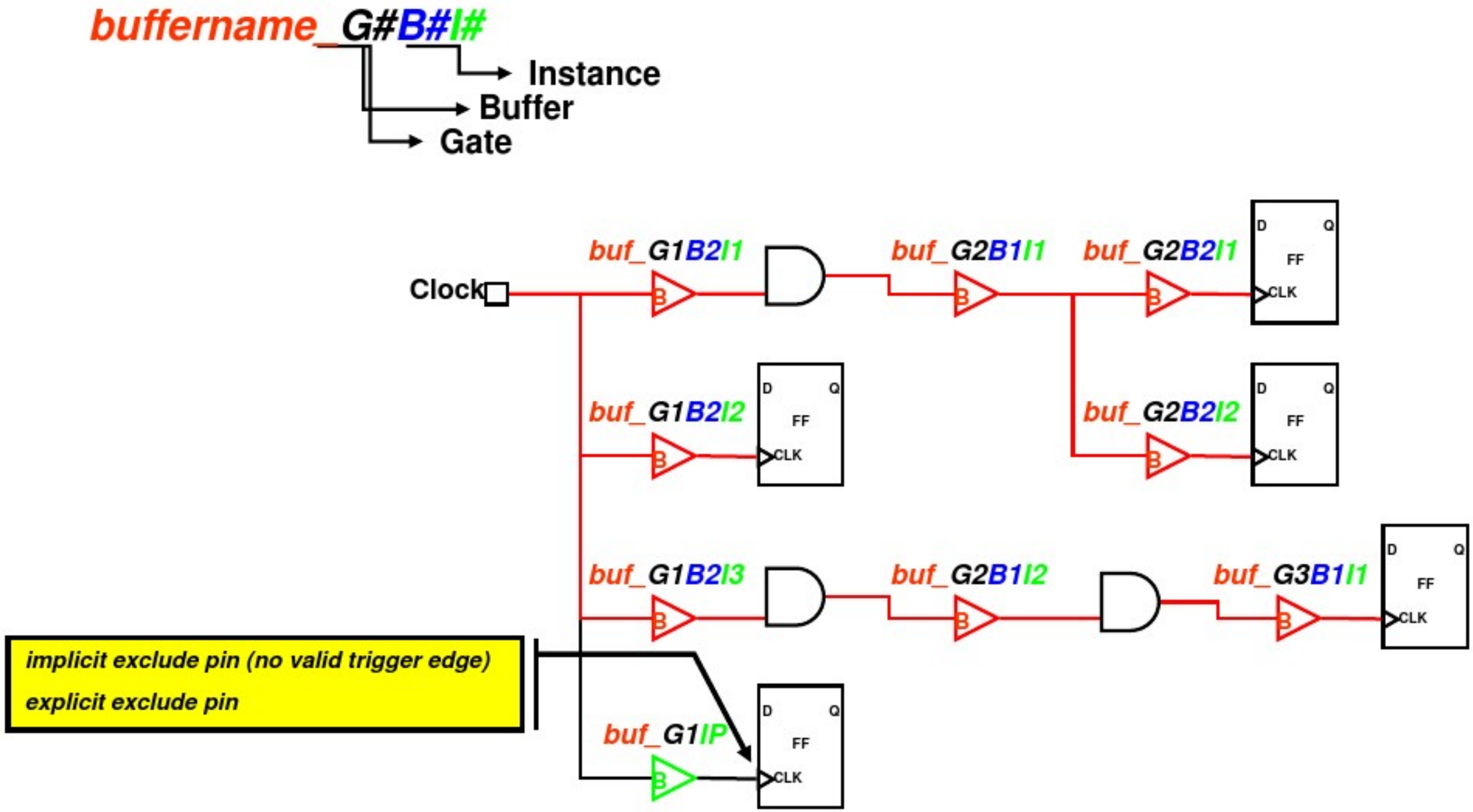
```
begin_clock_tree 0
clock_net core/clk

buffer_level invx12 4
buffer_level bufx12 7
buffer_level invx12 0
buffer_level bufx12 48
buffer_level bufx12 268
end_clock_tree
```

Appendix D

CTS Naming Convention

CTS – Naming Convention



You can further name the clock buffers using:

```
set_app_var cts_instance_name_prefix "MYPREFIX"
```

Which produces: MYPREFIX_clkbuf1x24_G1B1I23

Appendix E

Clock Shielding

Shielding: `create_zrt_shield`

- **Performs same-layer shielding on all nets with predefined shielding rules.**
  - The router routes shielding wires based on the shielding widths and spaces defined in the shielding rules.
- **Shield wires are tied to the ground net, standard cell ground pins, and standard cell rails.**
- **Use the –nets option to specify the subset of nets requiring shields.**
- **To remove existing shielding on specified nets, use the –mode unshield option.**
  - To remove existing shielding on the whole design, use the *remove_route_by_type -shield* command.

Shield routing rules

- **Shielding must be defined in an NDR (non-default routing rule) before use by *create_zrt_shield***
- **Multiple NDRs may be created and attached to different nets.**

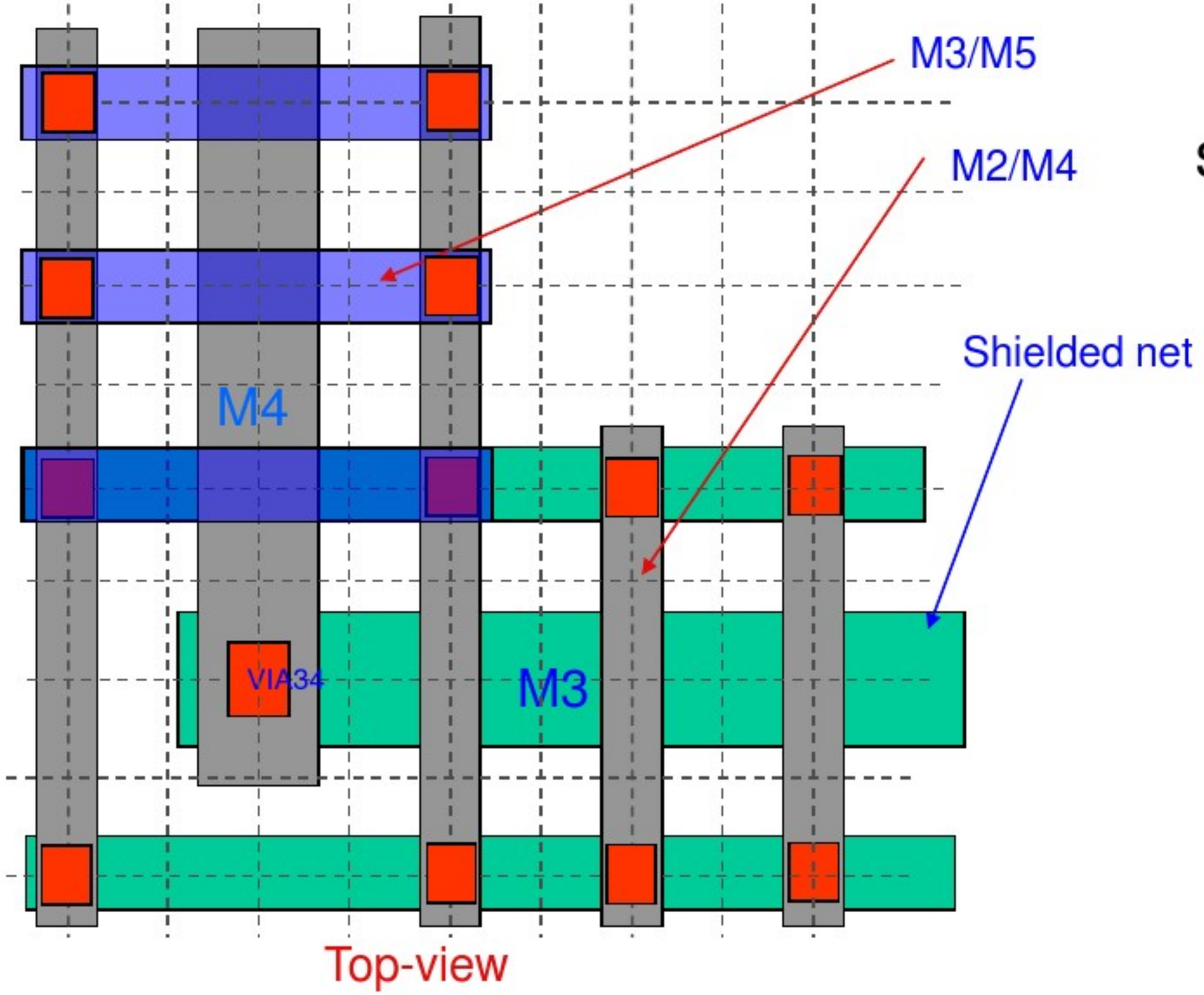
```
define_routing_rule  special_net_shielded   -default_reference_rule \
-taper_level 0 -snap_to_track          \
-via_cuts {VIA12A 1X2 VIA23 2X1  VIA34 1X2} \
-widths {METAL 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2}        \
-spacing {METAL 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21}      \
-shield_width {METAL 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2}  \
-shield_spacing {METAL 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21}

set_net_routing_rule -rule special_net_shielding {n19 analog_integrator_in}

create_zrt_shield -with_ground VSS –nets {n19 analog_integrator_in}
```

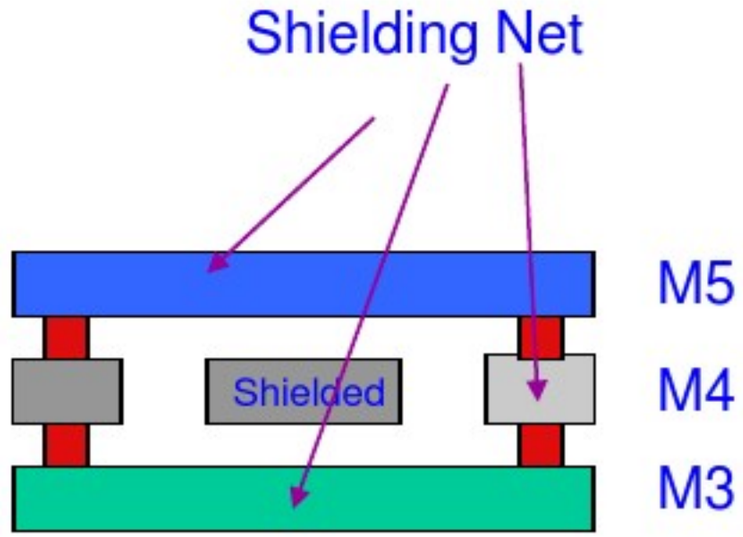
Coaxial Shielding is available but expensive

create_zrt_shield -with_ground VSS -coaxial_below true -coaxial_above true



Top-view

Upper and lower shields are placed at every other track



Cross-view