Innovus Clock Concurrent Optimization Technology

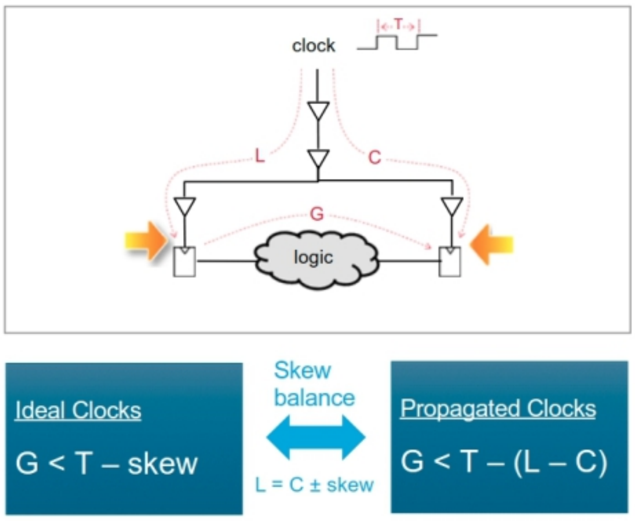
1. generate clock constraints from SDC constraints
2. implement the clock tree using CCOpt technology using the generated constraints
3. differentiate between traditional and modern clocking methods from balanced skew to leveraging useful skew
4. fix the clock tree postroute
5. specifies clock properties to customize the clock tree including:
   a) define routing types, CTS cells, stop and ignore pins
   b) modifying source latency settting in the hierachical implementation to meet timing at the block level

Traditional CTS Timing Closure compared to CCOpt Timing Closure

Traditional CTS is focused on building a skew balanced clock tree with separate optimization step after the clcok tree is built.

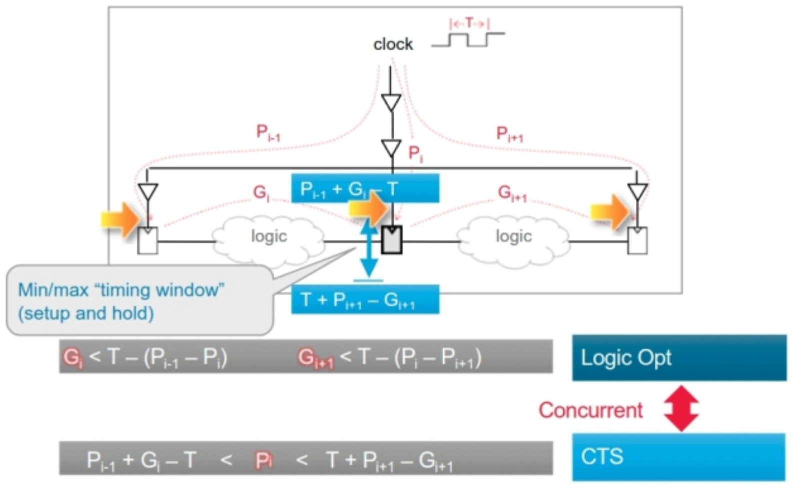
CCOpt balances the clock tree, optionally, with usedful skew(time borrowing) and post-CTS optimization

If the useful skew feature(time borrowing) is enabled in CCOpt, then timing closure utilizes borrowed slack to close timing.



In traditional timing closure, you minimize the skew, which in the diagram is the difference between the launch clock(L) and capture clock(C)

CCOpt Timing Closure

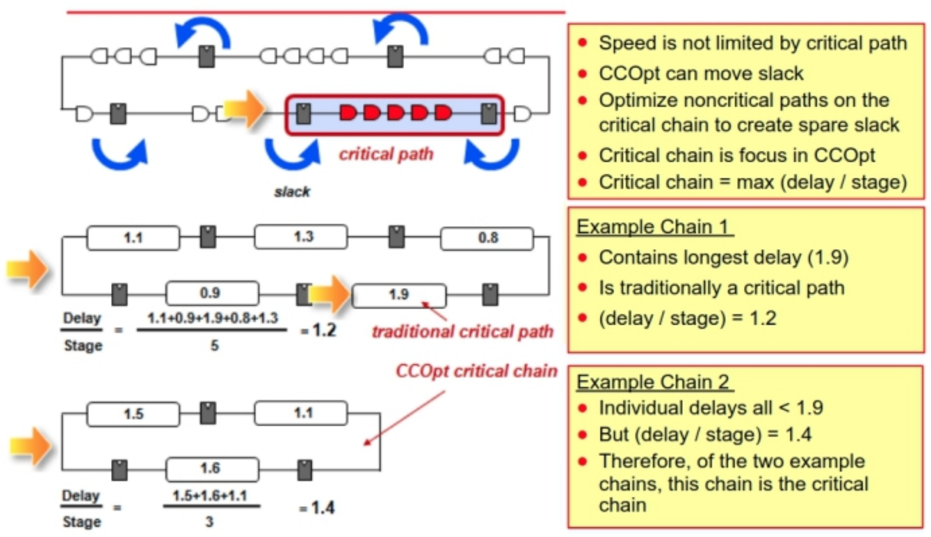


With CCOpt timing closure, to close timing at P_i, you can borrow timie from the launch clock P_{i-1} and the capture clock P_{i+1}, which increases the skew but allows for the datapath to meet timing if there is negative slack. There is also concurrent optimization of the datapath along clock tree synthesis.

Time Borrowing in CCOpt

1. In CCOpt, slack can flow across register boundaries
2. CCOpt algorithms focus on optimizing the entire logic chan(not just the critical path)
3. logic chains often loop
4. speed is not limited by critical path
5. CCOpt can move slack
6. Optimize noncritical paths on the critical chain to create spare slack
7. critical chain is focus in CCOpt
8. Critical chain = max(delay/stage)

   focus **critical chain** rather than **critical path,** critical chain is determined by **max(delay/stage)**


critical path
slack
1.1
1.3
0.8
0.9
1.9
Delay / Stage = (1.1+0.9+1.9+0.8+1.3) / 5 = 1.2
traditional critical path
CCOpt critical chain
1.5
1.1
1.6
Delay / Stage = (1.5+1.6+1.1) / 3 = 1.4

- Speed is not limited by critical path
- CCOpt can move slack
- Optimize noncritical paths on the critical chain to create spare slack
- Critical chain is focus in CCOpt
- Critical chain = max (delay / stage)

Example Chain 1
- Contains longest delay (1.9)
- Is traditionally a critical path
- (delay / stage) = 1.2

Example Chain 2
- Individual delays all < 1.9
- But (delay / stage) = 1.4
- Therefore, of the two example chains, this chain is the critical chain

chain 2 is critical chain

reporting cell filtering reasons

    report_ccopt_cell_filtering_reasons

Setting up CCOpt properties and command flow
As a part of the clock tree synthesis flow, the clock tree transitions from ideal to propagated.
clock tree synthesis:

1. adds buffering and sizes/places clock cells for drive and delay

   after the insertion of the clock tree, timing is with propagated clocks:
2. SOCV/AOCV/derates impact timing from non-common clock path
3. clock gate enable timing is no longer ideal
4. inter clock timing depends on achievable insertion delays
5. clock generator control logic timing is no longer ideal
6. single CTS unbufferable net can impact entire design timing

clock definitions

    clock tree

determine physical characteristics of the clock

skew group

determine balancing or initial balancing

clock, STA clock or SDC clock

clocks for timing analysis

skew group definitions

skew group defines groups of sinks to balance

skew group name is clockName/mode

modifying skew group ignore pins

modify_ccopt_skew_group -skew_group <skewGroup> -add_ignore_pins <pins>

set_ccopt_property

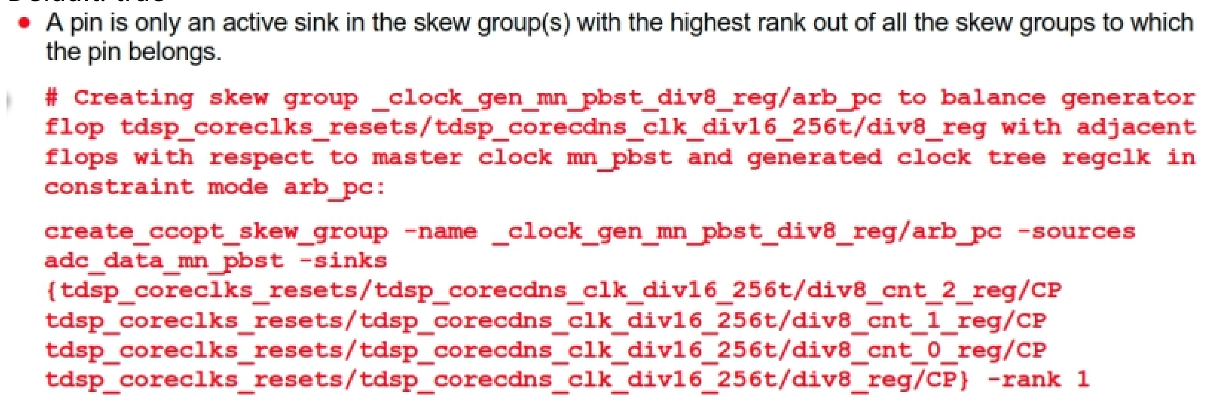
extract_clock_generator_skew_groups

This property will cause the create_ccopt_clock_tree_spec command to create skew groups for sequential generators and their adjacent registers. Such skew groups will be specified with the same highest rank so that they can be balanced from the other normal skew groups that share some sinks of them. The adjacent registers of a generator are registers that have a datapath timing path to talk with the generator directly. When this property is set to true, one skew group will be created per sequential generator instance, master clock and generated clock tree triple. The resulting skew groups will by default be named in the pattern:

_clock_gen_<master_clock_name>_<generator_local_name><_optional_number>/<constraint_mode_name>.

Type: boolean

Default: true



- A pin is only an active sink in the skew group(s) with the highest rank out of all the skew groups to which the pin belongs.

```
# Creating skew group _clock_gen_mn_pbst_div8_reg/arb_pc to balance generator
flop tdsp_coreclks_resets/tdsp_corecdns_clk_div16_256t/div8_reg with adjacent
flops with respect to master clock mn_pbst and generated clock tree regclk in
constraint mode arb_pc:

create_ccopt_skew_group -name _clock_gen_mn_pbst_div8_reg/arb_pc -sources
adc_data_mn_pbst -sinks
{tdsp_coreclks_resets/tdsp_corecdns_clk_div16_256t/div8_cnt_2_reg/CP
tdsp_coreclks_resets/tdsp_corecdns_clk_div16_256t/div8_cnt_1_reg/CP
tdsp_coreclks_resets/tdsp_corecdns_clk_div16_256t/div8_cnt_0_reg/CP
tdsp_coreclks_resets/tdsp_corecdns_clk_div16_256t/div8_reg/CP} -rank 1
```

skew groups on generated clocks used only for reporting

set_ccopt_property

constraints

The special value "none" is a synonym for an empty list, and specifies that the skew group will only be used for reporting purposes.

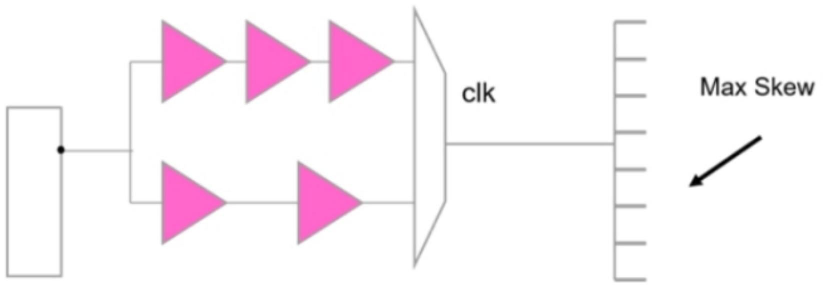
A constraints noe skew group is used only for reporting because it corresponds to a generated clock that is synchronous with its master clock, and will be balanced as part of thke skew group corresponding to the master clock

Balancing self-reconvergent clocks

The CCOpt tool automatically balances self-reconvergent cloks

First, the software analyzes the clk and determines the delay to the MUX inputs

Next, it balances the clock paaths to the MUX



add_ndr

create_route_type

set_ccopt_property route_type

Non-default routing rules have to be first deined via LEF or by using the add_ndr command

The CCOpt clock tree state is stored when the saveDesign command is run, including:

clock trees, skew groups, sink pin types

Analyzing the clock spec file

It is possible to delete the clock spec and to regenerate it after making changing to the setting using ghe commands

delete_ccopt_clock_tree_spec

reset_ccopt_config

setting transition targets

set_ccopt_property target_max_trans <value>

The transition target can be specified by net type, clock tree and delay corner. For example,

It may be desirable to have a tighter transition target at sink pins to improve flop CK->Q arc timing, but relax the transition target in trunk nets to reduce clock area and power.

cloning clock gating cells

set_ccopt_property clone_clock_gates true|false

default: false

cloning can improve clock gate enable timing but may worsen pwoer consumption, congestion and utilization

clock tree halo support

set_ccopt_property cell_halo_x -cell <cell> -clock_tree <clock_tree> <x_spacing>

set_ccopt_property cell_halo_y -cell <cell> -clock_tree <clock_tree> <y_spacing>

To check the halos, run the command:

report_ccopt_cell_halo_violations

limiting useful skew range

```
set_ccopt_property auto_limit_insertion_delay_factor <factor>
```

To not permit cells to abut on adjacent rows, run the command;

```
set_ccopt_property adjacent_rows_legal false
```

To limit cell density

```
set_ccopt_property cell_density <value>
```

Balancing skews of unrelated clocks

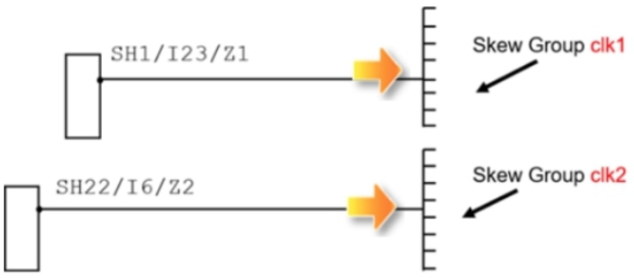
```
create_ccopt_skew_group -balance_skew_groups
```

To balance the skews of unrelated clocks, run the *create_ccopt_skew_group* command with the

–balance_skew_groups option.

Example

```
create_ccopt_skew_group –name skew_group_new
-balance_skew_groups {   clk1   clk2 }
-target_skew 20
```



setting stop pins

```
set_ccopt_property sink_type -pin <pin name> stop
```

setting ignore pins

```
set_ccopt_property sink_type -pin <pin name> ignore
```

```
report_clock_propagation -clock <clockName> -to <pinName> -verbose
```

what is clock latency or insertion delay>

clock latency or insertion delay is the time for a clock signal to propagate from the clock definiton point to a register clock pin

There are two types fof latency

1. source latency: clock source to the clock port in the design
2. network latency: clock port to the register (clock tree delay)

Source Latency updated after CTS

Sourece latency update is performmed to ensure that after CTS, when clocks are switched to propgagated mode, that I/O timing and inter-clock timing is consistent with the ideal mode timing model.

The source latency updates are reflected in the innovus timing constraints and will be saved in any saved database or exported in SDC as normal

Source Latency Update implications

what about multiple clocks?

source and network adjustment is performed per clock what about virtual clocks?

virtual clocks do not need latency modification

Does this make sense with useful skew?

latency modifications are applied after initial global skew balancing

what about at the top-level chip?

turn off the latency update by setting set_ccopt_property update_io_latency false

default is true

where is the latency saved?

The latency is saved in a file referenced by the viewdefinition.tcl with a -latency_file under the analysis view

```
create_analysis_view -name dtmf_view_hold -constraint_mode common -delay_corner dtmf_corner_min
-latency_file ${::IMEX::dataVar}/mmmc/views/dtmf_view_hold/latency.sdc
create_analysis_view -name dtmf_view_setup -constraint_mode common -delay_corner dtmf_corner_max
-latency_file ${::IMEX::dataVar}/mmmc/views/dtmf_view_setup/latency.sdc
set_analysis_view -setup [list dtmf_view_setup] -hold [list dtmf_view_hold]
```

get_ccopt_clock_tree_cells

This command returns a list of clock tree cells - cell instances that form a part of the clock tree network - whose names match the specified pattern. By default, the command returns all clock tree cells with matching names, but you can use optional parameters to filter the list based on cell type and clock tree membership.

```
innovus> llength [get_ccopt_clock_tree_cells *]
39
```

What are chains?

chains are launch and capture paths that stop when they either loop back on themselves or when they reach an IO

What are IO chains?

The IO chains are chains from the primary input to the primary output

clock latency or insertion delay is the sum of the clock source latency and the clock network latency

To report insertion delay and skew information, run

report_ccopt_skew_groups

```
Skew Group Structure:
=====================

-------------------------------------------------------------------------------------
Skew Group                          Sources    Constrained Sinks    Unconstrained Sinks
-------------------------------------------------------------------------------------
div_clk/functional_func_slow_max      1             1922                  1
my_clk/functional_func_slow_max       1             11536                 1
test_clk/functional_func_slow_max     1             11536                 1
-------------------------------------------------------------------------------------
```

```
Skew Group Summary:
===================

----------------------------------------------------------------------------------------------------------------------
------------------------------------------------------
Timing Corner    Skew Group                         ID Target    Min ID    Max ID    Avg ID
Std.Dev. ID    Skew Target Type    Skew Target    Skew    Skew window occupancy
```

To report DRV, sinks , buffer and other clock tree stats, run:
report_ccopt_clock_trees

```
Clock DAG stats:
================
--
Clock DAG wire lengths:
=======================
--
Clock DAG hp wire lengths:
==========================
--
Clock DAG capacitances:
=======================
--
Clock DAG sink capacitances:
============================
--
Clock DAG net violations:
=========================
--
Clock DAG primary half-corner transition distribution:
======================================================
--
Clock DAG library cell distribution:
====================================
--
Clock Tree Summary:
===================
--
Clock Sink Summary:
===================
--
Summary across all clock trees:
===============================
--
Clock Sink Summary across all clock trees:
```

```
================================================
--
Physical metrics across all clock trees:
==============================================
--
Transition distribution for half-corner fast_min:hold.late:
--
Transition distribution for half-corner slow_max:setup.late:
=====================================================================
--
Count of violations across all clock trees:
==================================================
--
Slew violation summary across all clock trees - Top 6 violating pins:
================================================================================
--
Clock Timing Summary:
========================
--
Total Transition Slacks Summary:
=====================================
```

To report worst timing chain, run:

    report_ccopt_worst_chain

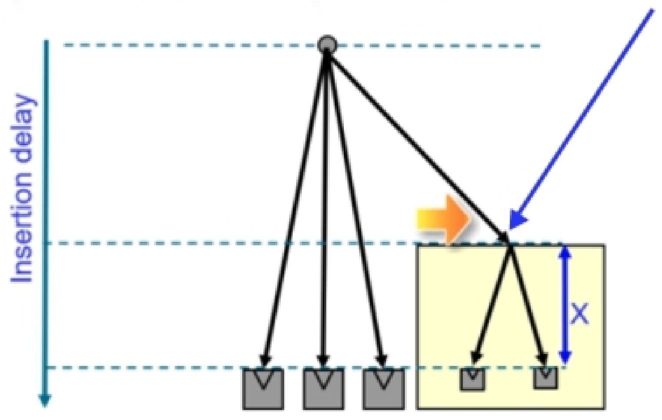
Tuning Early sinks for macros

```
set_ccopt_property insertion_delay X -pin <macro clock pin name>
```

Important to specify very early sinks and gates

- The *ccopt_design* command will not automatically tune early sinks
- Better starting point means better optimization and less runtime

```
set_ccopt_property insertion_delay X
-pin <macro clock pin name>
```



Tips: get_ccopt_property -help to get help on the properties

• get_ccopt_property * -help

- to list all properties

• get_ccopt_property *string* -help

- for help on properties matching *string*. For example,

get_ccopt_property *skew* -help

• get_ccopt_property <property_name> -help

- for help on a specific property

set_ccopt_property balance_mode cluster|trial|full

The following command enables you to run CTS in a different mode for debugging purposes:

cluster:

enables DRV buffering but does not perform any balancing or optimization of the clock tree.

trial:

enables DRV buffering and uses virtual delays to approximate how a full CTS will balance the clock trees.

full: default

a full CTS or optimization is performed and this is the default type.

set_ccopt_property

get_ccopt_property

Running CCOpt-CTS with cluster mode	set_ccopt_property balance_mode cluster ccopt_design -cts ;# global skew,low effort without optDesign step.	
Running CCOpt-CTS with trial mode	set_ccopt_property balance_mode trial ccopt_design -cts ;# global skew,low effort without optDesign step.	
Running CCOpt-CTS with full mode	set_ccopt_property balance_mode full ccopt_design -cts ;# global skew,low effort without optDesign step.	
Running CCOpt	set_ccopt_property balance_mode full setOptMode -usefulSkewCCOpt medium ccopt_design	

ccopt / ccopt -cts log file

section	**description**
Initial Summary	Initial timing before timing optimization
skew group insertion delays	insertion delay for each skew group
Guided max path lengths	Breakdown of the route guide lengths
Deviation of routing from guided max path lengths	Shows how the actual routing deviates from the route guides
Top 10 notable deviations of routed length from guided length	Specific nets whose length deviates most from their route guides
Clock DAG stats at the end of CTS	Count and area of cells used in the initial clock tree
Clock DAG capacitances at the end of CTS	Wire and Gate capacitance of the initial clock tree
Clock DAG primary half-corner transition distribution at the end of CTS	Max/Min trunk and leaf slew Target, Count, and Distribution at primary half-corner
Skew group summary at the end of	Max/Min ID and skew information for all

CTS	skew groups
GigaOpt + CCOpt summary information	WNS, TNS, and runtime summary of GigaOpt + CCOpt optimization
optDesign Final Summary	Final timing results including WNS, TNS, and DRV information

Note that the green highlighted items are only for CCOpt flow.

report_ccopt_skew_groups

- Skew Group Structure: Information on the number of sources and sinks for each skew group.
- Skew Group Summary: Insertion Delay (ID) and skew information for each skew group respective to each delay corner.
- Skew Group Min/Max path pins: End points with a minimum and maximum insertion delay for each skew group respective to each delay corner. This is followed by the detailed path information.

report_ccopt_clock_trees

report the slew, area, and buffer count for each clock tree

report_ccopt_worst_chain

Worst Chain is reported from time-to-time in the log during CCOpt, and an examination of the log may help identify the reasons limiting timing optimization. In addition, the report_ccopt_worst_chain command can be used to report the worst timing chain after ccopt_design has completed. However, note that this will reflect the current worst chain and not the worst chain during optimization.

In this report:

- 'x' in the row "x clk_en" indicates a pin that cannot be adjusted because it is outside the defined clock trees. This is commonly an IO pin. clk_en is an input port in the Leon design.
- 'g' in the row "g CGIC_INST" indicates a clock gate. CGIC_INST is a clock gating cell in the design.
- 'o' in the row "o proc0/cmem0/dtags0/u0/id0" indicates a clock tree sink that can be skewed.

Note that this command is not needed in the **CCOpt-CTS** flow. essentialy Time Borrowing and useful-skew

CCOpt Clock Tree Debugger (CTD)

The CCOpt Clock Tree Debugger is a graphical tool for analyzing and debugging the clock tree results. It becomes available once the CCOpt clock tree constraints are defined (that is, after running create_ccopt_clock_tree_spec)

The Control Panel combines the functionality of the **Visibility** and **Color by** menus into a single form

report_clocks shows clock as ideal after CCOpt

For clocks, there is a **waveform object** corresponding to the create_clock object in the SDCs. The waveform is referenced with get_clocks. There is also a clock object applied to the **pins/ports** referred to with get_pins/get_ports.

The clock waveform is not propagated because it will cause incorrect IO timing if a clock is used for both internal and IO timing. report_clocks reports information on the clock waveform and, therefore, does not report the clock as propagated.

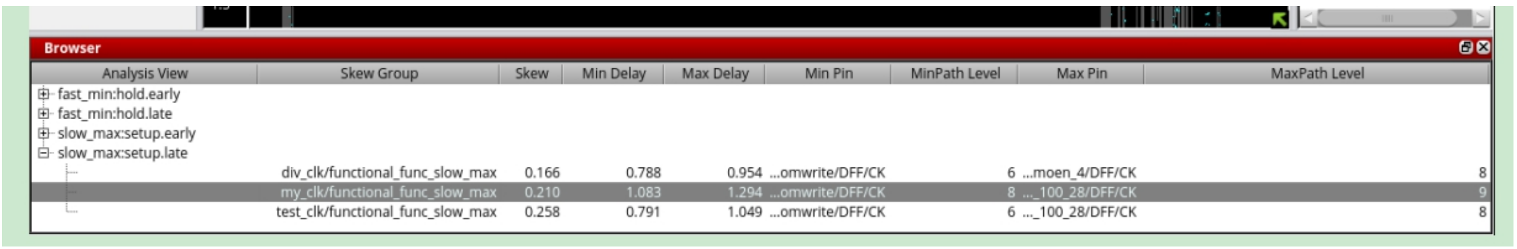
The is_propagated_clock property can be queried to determine if a clock waveform or pin/port is propagated. The following commands show how the property differs for the clock waveform and clock port after CCOpt, and determine if the clock is propagated at a certain pin/port.

```
+------------------------------------------------------------------------------------------+
|                              Clock Descriptions                                          |
|------------------------------------------------------------------------------------------|
|           |               |               |        |       |       |      Attributes    |
|-----------+---------------+---------------+--------+-------+-------+----------------------|
|   Clock   |    Source     |     View      | Period |  Lead | Trail | Generated | Propagated |
|   Name    |               |               |        |       |       |           |            |
|-----------+---------------+---------------+--------+-------+-------+-----------+------------|
|  div_clk  | clk_div_reg/Q | func_slow_max |  8.000 | 0.000 | 4.000 |     y     |      n     |
|  my_clk   |      clk      | func_slow_max |  4.000 | 0.000 | 2.000 |     n     |      n     |
| my_clk_v0 |       -       | func_slow_max |  8.000 | 0.000 | 4.000 |     n     |      n     |
| my_clk_v1 |       -       | func_slow_max |  8.000 | 0.000 | 4.000 |     n     |      n     |
| my_clk_v2 |       -       | func_slow_max |  8.000 | 0.000 | 4.000 |     n     |      n     |
|  test_clk |   scan_clk    | func_slow_max |  8.000 | 0.000 | 4.000 |     n     |      n     |
+------------------------------------------------------------------------------------------+
```

report_clocks
shows clock ...

```
innovus #> get_property [get_clocks my_clk] is_propagated_clock
    false false
innovus #> get_property [get_clocks my_clk] view_name
    func_slow_max func_fast_min
innovus #> get_property [get_clocks my_clk -filter "view_name=~func_slow_max"] is_propagated_clock
    false

innovus #> get_property [get_ports clk] is_propagated_clock
    true
```


Browser

Analysis View	Skew Group	Skew	Min Delay	Max Delay	Min Pin	MinPath Level	Max Pin	MaxPath Level
fast_min:hold.early								
fast_min:hold.late								
slow_max:setup.early								
slow_max:setup.late								
	div_clk/functional_func_slow_max	0.166	0.788	0.954	...omwrite/DFF/CK	6	...moen_4/DFF/CK	8
	my_clk/functional_func_slow_max	0.210	1.083	1.294	...omwrite/DFF/CK	8	..._100_28/DFF/CK	9
	test_clk/functional_func_slow_max	0.258	0.791	1.049	...omwrite/DFF/CK	6	..._100_28/DFF/CK	8

analysis view: delay_corner:[setup, hold].[early, late]

skew_group: clock/constratint_mode

global skew

    get_ccopt_skew_group_delay -to <DFF/CK> -skew_group <SKEW_GROUP> -check_type setup -delay_corner <DC> -delay_type late

    get_ccopt_skew_group_delay -to <DFF/CK> -skew_group <SKEW_GROUP> -check_type setup -delay_corner <DC> -delay_type late

        -delay_type late: slow path (i.e. launch path of setup check, capture path of hold check)

        -delay_type early: fast path (i.e. capture path of setup check, launch path of hold check)

get_ccopt_clock_tree_sinks

    [-help]

    pattern

    [-in_clock_trees list_of_trees]

        Specifies a TCL list of clock trees whose sinks are to be included in the list.

        If not specified, all the sinks of all the clock trees in the design that satisfy the other conditions of the command will be returned.

    [-not_in_clock_trees list_of_trees]

        Specifies a TCL list of clock trees whose sinks are to be excluded from the list.

        If not specified, all the sinks that satisfy the other conditions of the command will be returned.

    [-regexp]

    This command returns a list of clock tree sinks whose names match the specified pattern. By default, the command returns all clock tree sinks with matching names, but you can use optional parameters to filter the list based on the clock tree membership.

get_ccopt_property sink_type -pin $sink

sink_type

    The type of sink this pin represents.   For this property to take effect, set it before running create_ccopt_clock_tree_spec.

    Valid values are as follows:

        auto: The pin type will be automatically determined by CCOpt

        through: Through pin. Trace the clock tree through this pin.

        stop: Stop pin. When defining clock trees, CCOpt stops searching for parts of the clock tree at stop pins.

        ignore: Ignore pin. CCOpt stops searching for parts of the clock tree at ignore pins and it does not attempt to balance the insertion delay of ignore pins.

        min: Min pin. Keep the pin at minimal insertion delay.

        exclude: Exclude pin. Exclude this pin from the clock tree.

    Valid values: auto through stop ignore min exclude

    Default: auto

Script to report the ignore pin list from a CCOpt clock tree

```
proc usr_rpt_ignore_inclock {file_name} {
  set file1 [open $file_name w]
  set i 0
  set count 0
```

```
        foreach sink [get_ccopt_clock_tree_sinks *] {
          set type [get_ccopt_property sink_type -pin $sink]
          if { $type == "ignore"} {
            puts $file1 "$sink $type"
            incr count
          }
        }
        puts $file1 " INFO IGNORE PINS in CCOPT CLOCK TREE \n ##=============## \n TOTAL IGNORE PINS
COUNT $count \n ##=============## "
        close $file1
      }
```