# Intel® Quartus® Prime Pro Software Timing Analysis - Lecture



intel

1

## Intel® FPGA Products


| FPGA Mid-range FPGAs SoC & Transceivers | FPGA Optimized for High Bandwidth | FPGA Optimized for High Bandwidth | FPGA Cost/Power Balance SoC & Transceivers | FPGA/CPLD Lowest Cost, Lowest Power |
|---|---|---|---|---|

Resources

| Embedded Soft and Hard Processors | Design Software | Acceleration Boards & Development Kits | Intellectual Property (IP) |
|---|---|---|---|
| Nios® II<br>ARM* | Intel® Quartus® Prime Design Software<br>Intel® FPGA SDK for OpenCL™ | | ▪ Industrial<br>▪ Computing<br>▪ Enterprise |

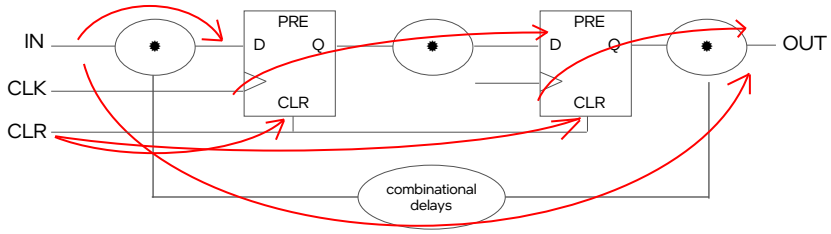


intel 2

2

## Purpose of Timing Analysis (Review)

- To analyze every design path against required performance specifications
  - Catch timing-related errors faster and easier than gate-level simulation & board testing
- Designer must enter timing requirements & exceptions
  - Used to guide Fitter during placement & routing
  - Used to compare against actual results (post-fit)







3

## FPGA Design Challenges

- Properly constraining a design is one of the most critical aspects of FPGA design development
- FPGA Designers may be unfamiliar with using Intel® Quartus® Prime Pro software Timing Analyzer effectively
- FPGA Designers may be unfamiliar with writing Synopsys* Design Constraints (SDC) when targeting Intel FPGAs
- Designers need to get up and running quickly due to shorter development cycles




4

## Intel® Quartus® Prime Pro Software Timing Analysis – Objectives

**The purpose of this class is for users new to FPGAs or Intel® FPGAs to...**

- Learn the flow for setting up and performing timing analysis in the Intel® Quartus® Prime Pro software Timing Analyzer
- Gain understanding of SDC terminology and syntax
- Use the latest recommendations to write clear and effective SDC files for properly constraining and analyzing Intel FPGA designs for timing




5

## Intel® Quartus® Prime Pro Software Timing Analysis – Agenda

- Intel® Quartus® Prime Pro Software Timing Analyzer
- Writing SDC Files




6

## Prerequisites

- Understanding basic Timing Analyzer parameters and equations
  - See *Timing Analyzer: Introduction to Timing Analysis* online training
- Knowledge of FPGA architecture
- Knowledge of FPGA design flow
- Familiarity with using Intel® Quartus® Prime Pro software design flow




7

# Intel® Quartus® Prime Pro Timing Analysis

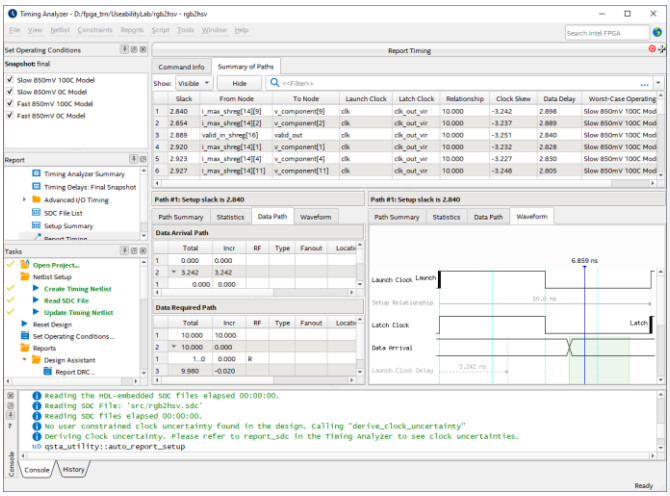
Intel Quartus Prime Pro Software Timing Analyzer




8

# Timing Analyzer

- Timing analysis engine in Intel® Quartus® Prime design software providing timing analysis solution for all levels of experience
- Features
  - Synopsys* Design Constraints (SDC) support
    - Standardized constraint methodology
  - Easy-to-use interface
    - Constraint entry
    - Standard, on-the-fly reporting
  - Scripting fully supported
    - Presentation focuses on using GUI



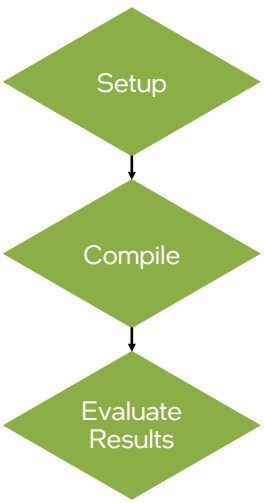






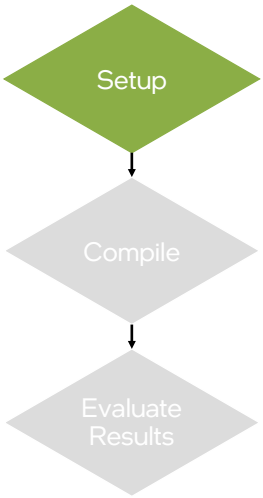
# Intel® Quartus® Prime Pro Software Timing Analyzer – Section Agenda

1. Set up project for timing analysis
2. Compile design
3. Evaluate results and generate detailed Timing Analyzer reports



# 1) Setting Up Timing Analyzer

- Create SDC file(s)
- Add SDC file(s) to project
- Enable/Disable additional Timing Analyzer features







11

# Constraining

- Recommendation: Ensure all design paths are constrained
  - Constraints guide the Fitter during placement & routing design
    - Without guidance, Fitter can make incorrect optimization choices
  - Design paths must be constrained to fully analyze design
    - Timing Analyzer only performs slack analysis on constrained design paths
- Not as difficult a task as it may sound
  - Wildcards are supported
  - Single, generalized constraints cover many paths, even all paths in an entire clock domain




12

# Create SDC File(s)

- Define top-level SDC file
  - Start with initial clock understanding
  - Add more detailed constraints as needed and when known
- Use additional SDC files for key modules/IP/interfaces
  - Most Intel® FPGA IP create their own SDC files, if required, which get automatically added to the project

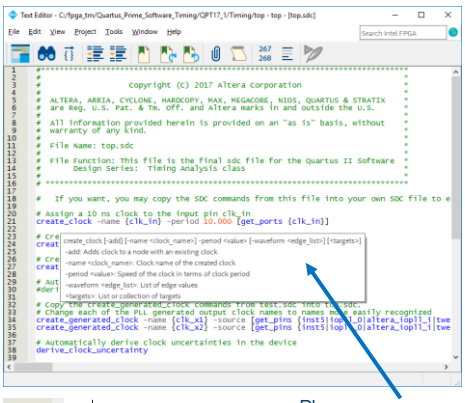




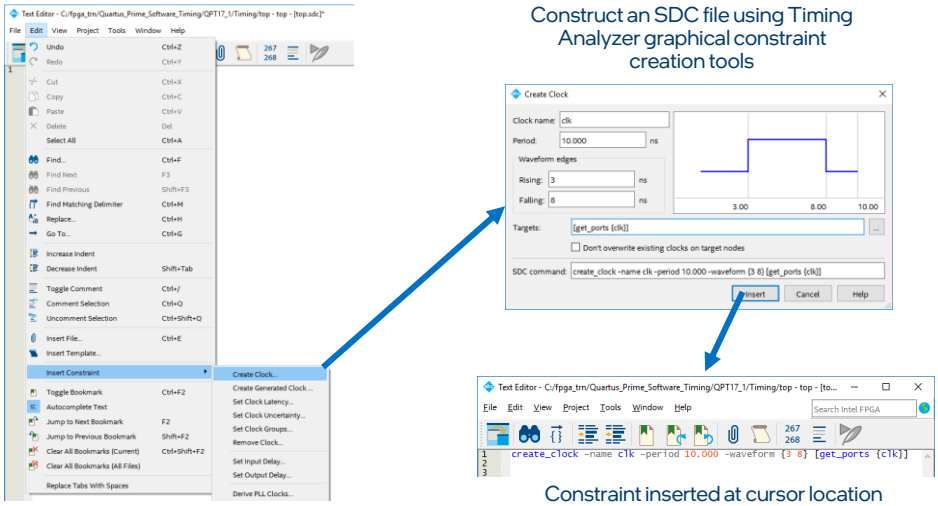
# SDC File Editing

- Intel® Quartus® Prime software text editor features for editing SDC files
  - Access to GUI dialog boxes for constraint entry (Edit → Insert Constraint)
  - SDC templates
  - Syntax coloring
  - Delimiter matching
  - Tooltip syntax help
  - Auto-complete

Timing Analyzer File menu → Open/New SDC File
Intel® Quartus® Prime software File menu → New → Other Files



# SDC File Editor GUI Constraint Entry





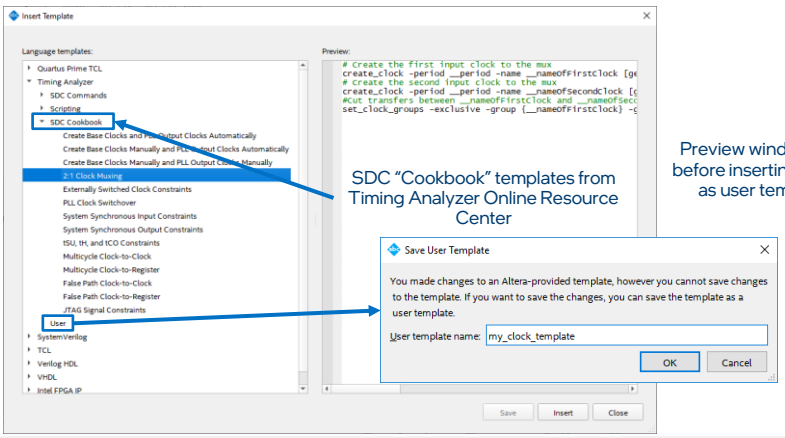



15

# SDC Templates

- Quickly add pre-defined and customized constraint templates

Toolbar button or Edit menu → Insert Template





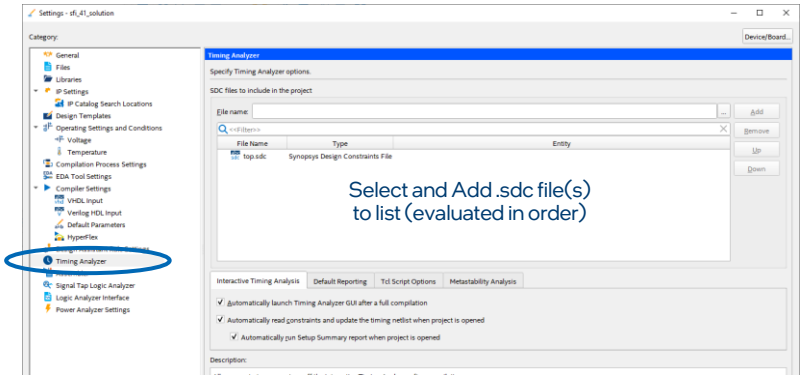



16

# Add SDC File(s) to Project

- Tells Intel® Quartus® Prime Design software which SDC constraints to use for compilation and evaluation

Assignments menu → Settings




QSF: set_global_assignment -name SDC_FILE <filename>.sdc



intel

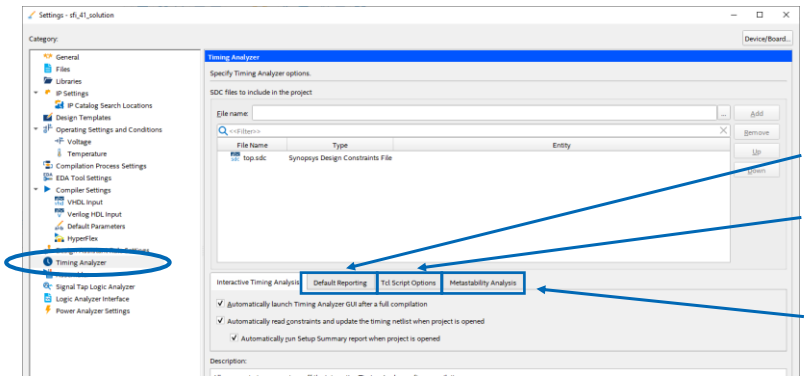
17

17

# Enable/Disable Additional Timing Analyzer Features

- Control when Timing Analyzer runs and perform additional non-standard analysis

Assignments menu → Settings




QSF: set_global_assignment -name SDC_FILE <filename>.sdc



intel

18

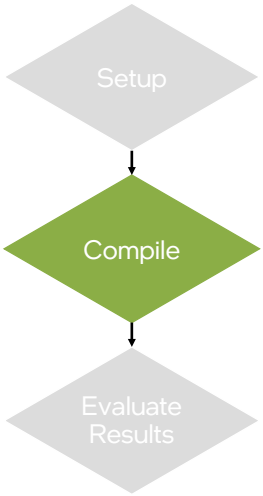
18

## 2) Compile Design

- Full compilation

OR

- Use incremental optimization flow
  - Timing analysis can be performed after running Plan, Retime or Fitter (Finalize) stages
  - Recommendation: use Plan stage when creating and editing SDC files for faster iterations

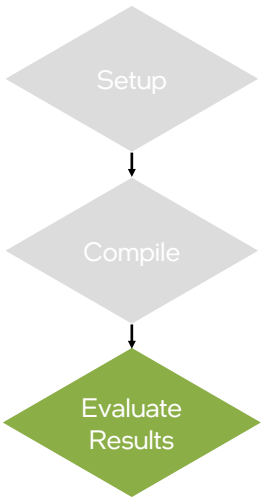






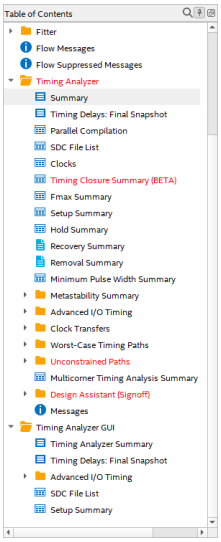
## 3) Evaluate Timing Analysis Results

- View Timing Analyzer summary information directly in Intel® Quartus® Prime Design Software Compilation Report
  - When full compilation is performed
- Open Timing Analyzer GUI …
  - When more detailed analysis required
  - When using incremental optimization flow



# Timing Analyzer Folder in Compilation Report

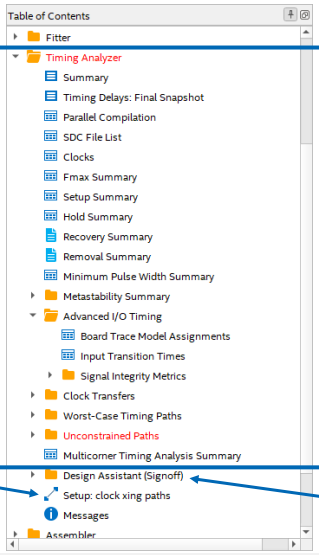
- Use to quickly verify compilation results able to meet timing requirements
  - Failing and unconstrained reports appear in red
- Displays subset of reports available in Timing Analyzer GUI
- Includes any customized reports generated from Tcl Script File setting








# Timing Analyzer Folder in Compilation Report




Standard reports generated during compile (as specified in the Settings window). For example:

- .sdc files used during fitting
- Clocks generated
- Timing violations
- Unconstrained paths
- Worst-case paths

Report generated by Tcl script in Settings

Design Assistant analysis results

# Reports in the Timing Analyzer Folder of Compilation Report

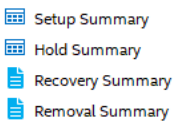
- Is the design meeting performance?
  - Synchronous analysis
    - Setup
    - Hold
  - Asynchronous analysis
    - Recovery
    - Removal
  - Fmax
  - IP-specific reports
- What could be issues leading to unexpected results?
  - SDC file list
  - Design Assistant checks
  - Clocks (discussed later)
  - Unconstrained paths (discussed later)
  - Worst-case timing paths




23

# Timing Slack Reports

- Simplest, most common type of timing report
- Each row lists
  - Worst case (least positive or most negative) slack on a clock domain in the design
  - End point total negative slack (TNS), sum of most negative slacks for each failing endpoint
  - Timing model producing the worst-case slack value is displayed




Setup Summary

Show: Visible | Hide | <<Filter>>

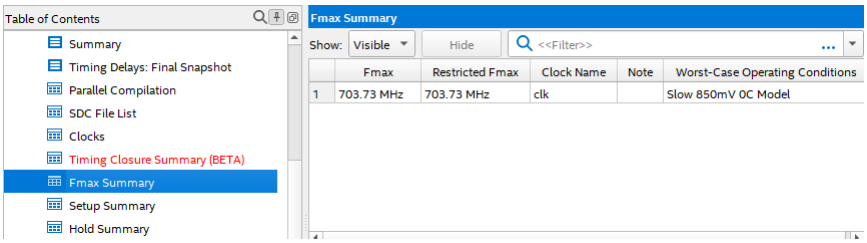
| | Clock | Slack | End Point TNS | Worst-Case Operating Conditions |
|---|---|---|---|---|
| 1 | clk_500 | 0.813 | 0.000 | Slow 900mV 100C Model |
| 2 | clk_250 | 1.111 | 0.000 | 1 Slow vid1 0C Model |




24

# Fmax Report

- Shows the maximum clock speed possible by clock domain based on current layout
  - Translates clock period and slack into easily recognizable value
  - Excludes cross-domain timing paths
- Use to quickly determine how fast your design can be clocked



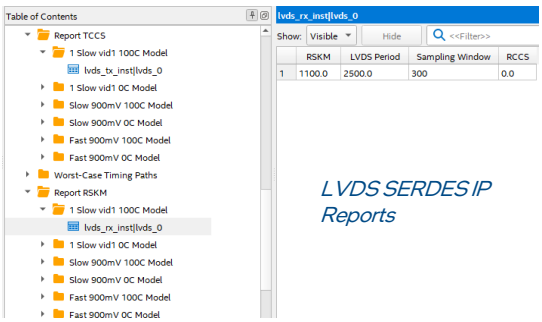




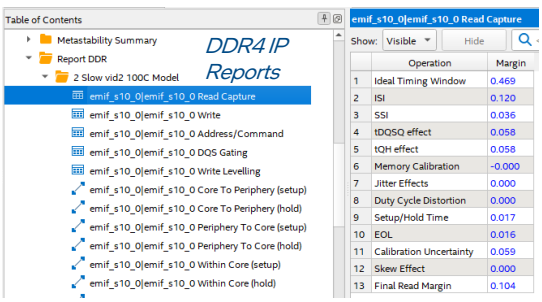
25

# IP-Specific Reports

- IP targeting FPGA-specific hardware resources generate special reports to check their timing
  - IP detected by software and included in summary reports
- Use to confirm specialized analysis margins are met for correct hardware operation




*LVDS SERDES IP Reports*



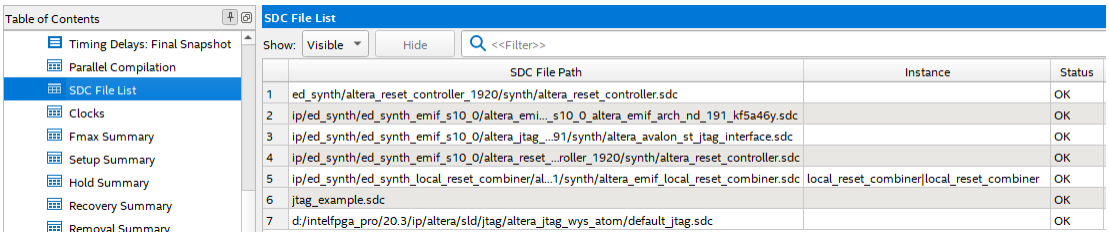

*DDR4 IP Reports*




26

# SDC File List

- Reports all the SDC files read during compilation and analysis
- Check for missing or incorrect constraint files



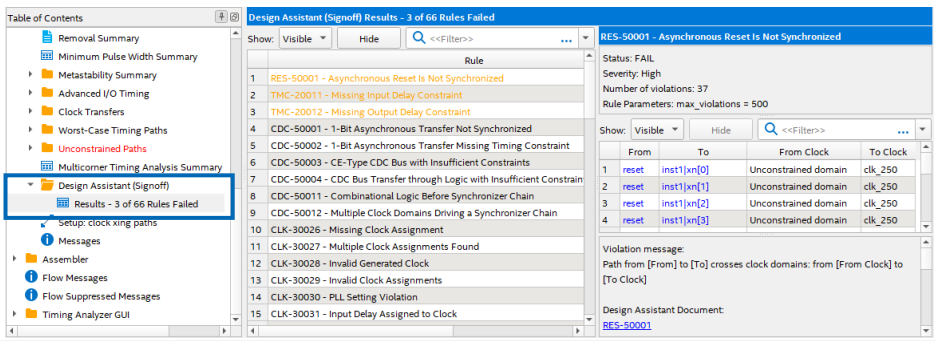




27

# Design Assistant (Signoff) Folder

- Reports if any enabled Design Assistant rules checked during Timing Signoff have been violated
- Use to find design issues that could affect functionality and timing



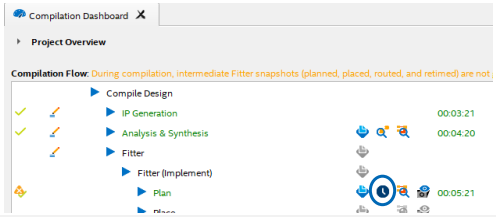




28

# Open Timing Analyzer GUI

- Timing Analyzer GUI opens by default at the end of full compilation ready to begin generating reports
- Manually launch GUI from Compilation Dashboard
- Timing netlist/database automatically generated for design
- Project SDC files automatically read in





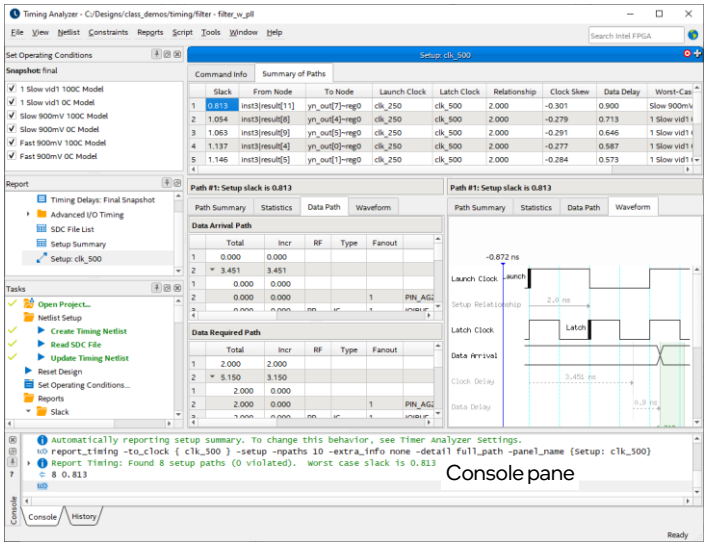



# Timing Analyzer GUI

Operating Conditions pane

Report pane

Tasks pane

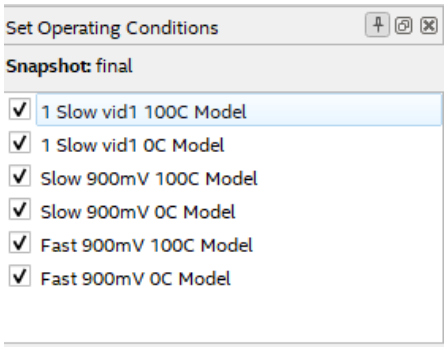



View pane

Console pane

# Timing Analyzer GUI: Set Operating Conditions Pane

- Various timing models/corners available
- All timing models available for Routed and Final netlists and selected by default
- Only slow (low voltage, high temperature) model available for Plan netlist

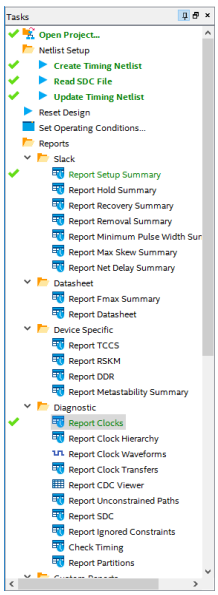






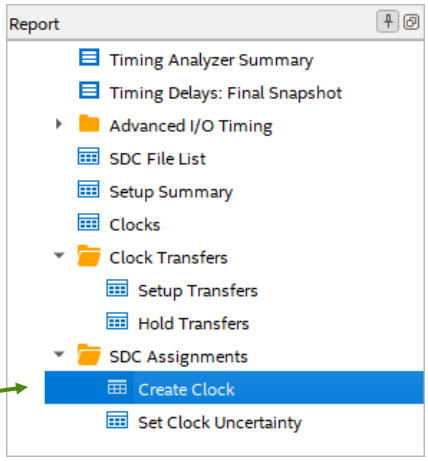
# Timing Analyzer GUI: Tasks Pane

- Provides quick access to common and useful operations
  - Command execution
  - Report generation
- Executes most commands with default settings
  - Use menus for non-default settings
- Double-click to execute any command



# Timing Analyzer GUI: Report Pane

- Displays list of generated reports currently available for viewing
  - Reports generated by Tasks pane
  - Reports generated using report commands from Report menu




Select report to display in View pane



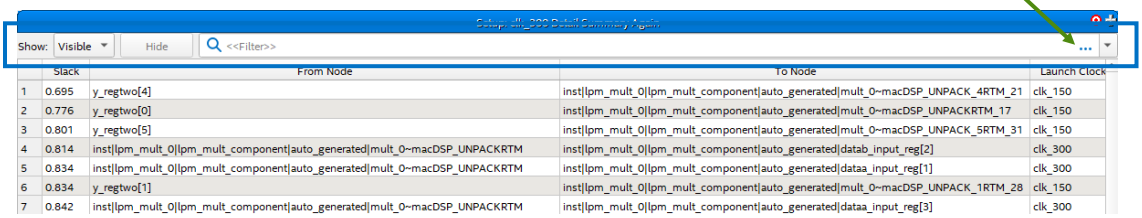




# Timing Analyzer GUI: View Pane

- Main viewing area displaying results of selected report
- Additional filter control for parsing larger report results

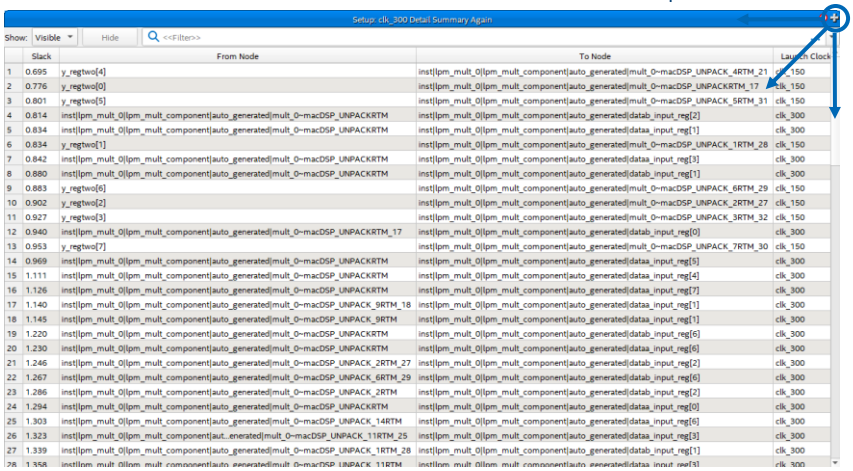
Opens Filter dialog for more filtering options




| | Slack | From Node | To Node | Launch Clock |
|---|---|---|---|---|
| 1 | 0.695 | y_regtwo[4] | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~macDSP_UNPACK_4RTM_21 | clk_150 |
| 2 | 0.776 | y_regtwo[0] | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~macDSP_UNPACKRTM_17 | clk_150 |
| 3 | 0.801 | y_regtwo[5] | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~macDSP_UNPACK_5RTM_31 | clk_150 |
| 4 | 0.814 | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~macDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|datab_input_reg[2] | clk_300 |
| 5 | 0.834 | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~macDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|dataa_input_reg[1] | clk_300 |
| 6 | 0.834 | y_regtwo[1] | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~macDSP_UNPACK_1RTM_28 | clk_150 |
| 7 | 0.842 | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~macDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|dataa_input_reg[3] | clk_300 |

# Timing Analyzer GUI: Viewing Multiple Reports

Click & drag '+' sign to divide view pane into multiple windows







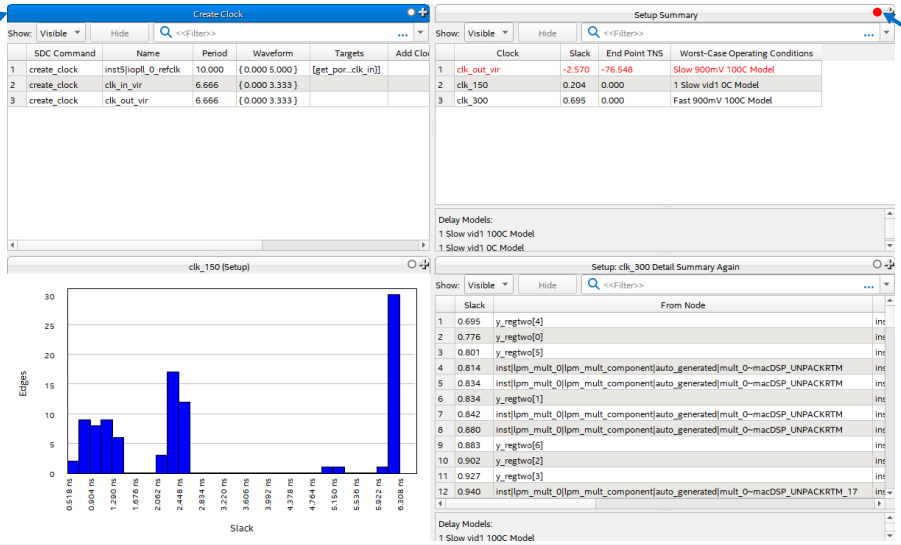



# Timing Analyzer GUI: Viewing Multiple Reports

Highlight window, then select report in Reports pane you would like to appear there

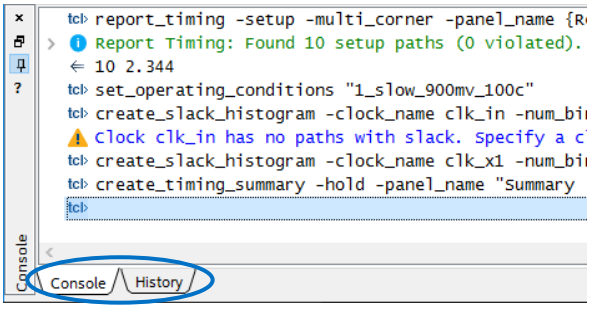
Drag bars to edges to remove splits

Use Target Pane button to force a selected report to appear in a pane



# Timing Analyzer GUI: Console Pane

- Displays Timing Analyzer output messages
- Allows direct entry and execution of SDC & Tcl commands
  - Also displays equivalent of commands executed in GUI
- History tab to record all executed SDC & Tcl commands
  - Copy & paste to create scripts or SDC files
  - Run scripts from Script menu



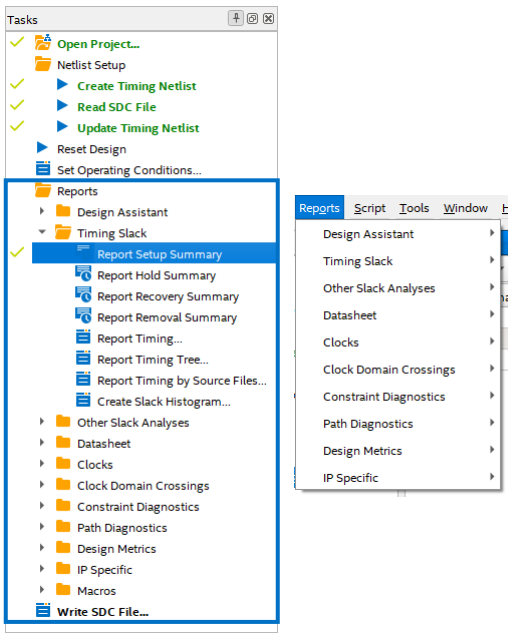




37

# Generating Timing Reports

- Double click on report in Tasks pane or select from Reports menu
  - GUI automatically runs all prior necessary steps for generating report
- Dozens of reports available to help understand design timing
  - Reports to check for constraint or design issues
  - Reports to verify timing and locate violations







38

## Task Pane Report Categories

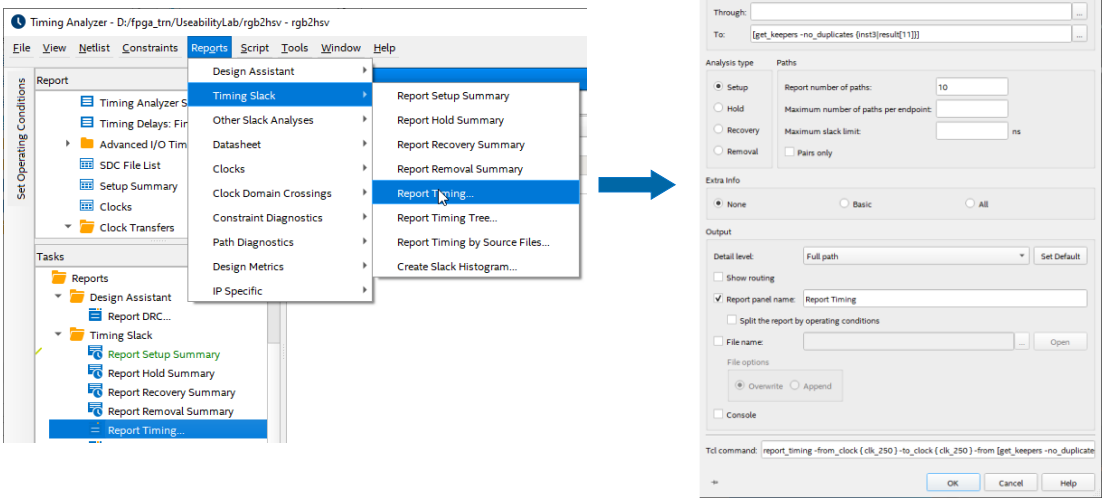
- Timing Slack – Standard slack analysis (single value per domain) or net analysis (discussed later)
- Datasheet – View FPGA device timing as standalone application specific standard product (ASSP)
- Device Specific – Review FPGA hardened resource timing
- Constraint Diagnostic – Look for constraint issues
- Design Metrics – Reports characteristics of your design beyond just a path analysis such as logic depth, pipelining, and so forth
- And more.....




## Report Example: Report Timing for Detailed Slack/Path Analysis

- Create more specific/detailed reports
  - Example: details on a specific clock domain
  - Example: view timing paths between particular I/O & registers
- Create using GUI, menu or Tcl commands
  - Use GUI or menu to get help building command and to see report immediately
  - Store as Tcl command for repeatability
  - All commands executed by Timing Analyzer appear in Console pane History tab regardless of source

# Report Timing (GUI)








41

Create report based on source or destination clock domains

Create report based on timing path endpoints or intermediate point

Select type of slack analysis

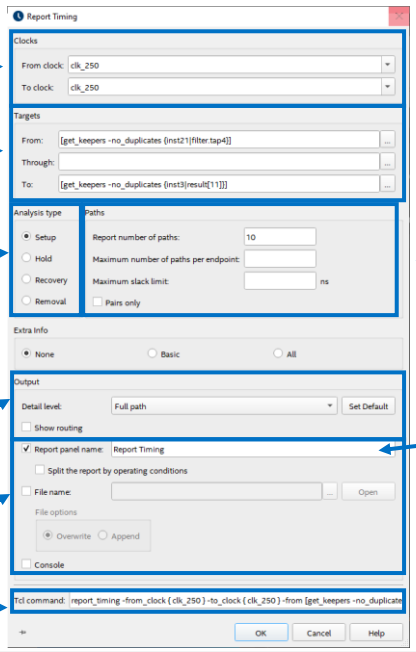
Select level of detail (next slide)

Select where to send output report

Equivalent Tcl command (copy & paste for scripting)

Control the number of results or paths in report

Name of report to use in Timing Analyzer panel








42

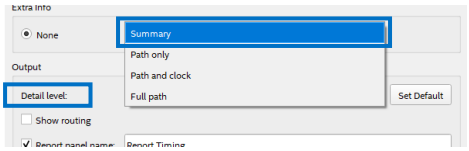
## Report Timing Detail Level Descriptions

- `summary`: lists individual paths in a table with select path information
- `path_only`: reports timing path without any clock path detail
- `path_and_clock`: reports timing path including clock detail tracing back to the launch and latch clocks
- `full_path`: reports timing path including clock detail tracing back to base clock, through any generated clocks *(default option)*




43

## Summary Slack/Path Report




Launch to latch clock relationship (may be adjusted by exceptions discussed later)

Operating condition used for slack value

Setup: clk_300 Detail Summary

Show: Visible | Hide | <<Filter>>

| | Slack | From Node | To Node | Launch Clock | Latch Clock | Relationship | Clock Skew | Data Delay | Worst-Case Operating Conditions |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.695 | y_regtwo[4] | inst\|lpm_mult_0\|lpm_mul...0~macDSP_UNPACK_4RTM_21 | clk_150 | clk_300 | 6.666 | 0.048 | 6.062 | Fast 900mV 100C Model |
| 2 | 0.776 | y_regtwo[0] | inst\|lpm_mult_0\|lpm_mult...lt_0~macDSP_UNPACKRTM_17 | clk_150 | clk_300 | 6.666 | 0.030 | 5.962 | Fast 900mV 100C Model |
| 3 | 0.801 | y_regtwo[5] | inst\|lpm_mult_0\|lpm_mul...0~macDSP_UNPACK_5RTM_31 | clk_150 | clk_300 | 6.666 | 0.043 | 5.952 | Fast 900mV 100C Model |
| 4 | 0.814 | inst\|lpm_mult_0\|lp...0~macDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_c...nerated\|datab_input_reg[2] | clk_300 | clk_300 | 3.333 | 0.047 | 2.536 | Slow 900mV 100C Model |
| 5 | 0.834 | inst\|lpm_mult_0\|lp...0~macDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_c...nerated\|dataa_input_reg[1] | clk_300 | clk_300 | 3.333 | 0.047 | 2.516 | Slow 900mV 100C Model |
| 6 | 0.834 | y_regtwo[1] | inst\|lpm_mult_0\|lpm_mul...0~macDSP_UNPACK_1RTM_28 | clk_150 | clk_300 | 6.666 | 0.028 | 5.904 | Fast 900mV 100C Model |
| 7 | 0.842 | inst\|lpm_mult_0\|lp...0~macDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_c...nerated\|dataa_input_reg[3] | clk_300 | clk_300 | 3.333 | 0.047 | 2.508 | Slow 900mV 100C Model |
| 8 | 0.880 | inst\|lpm_mult_0\|lp...0~macDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_c...nerated\|datab_input_reg[1] | clk_300 | clk_300 | 3.333 | 0.047 | 2.470 | Slow 900mV 100C Model |
| 9 | 0.883 | y_regtwo[6] | inst\|lpm_mult_0\|lpm_mul...0~macDSP_UNPACK_6RTM_29 | clk_150 | clk_300 | 6.666 | 0.031 | 5.856 | Fast 900mV 100C Model |
| 10 | 0.902 | y_regtwo[2] | inst\|lpm_mult_0\|lpm_mul...0~macDSP_UNPACK_2RTM_27 | clk_150 | clk_300 | 6.666 | 0.029 | 5.874 | 1 Slow vid1 0C Model |

Calculated slack

Source & destination nodes

Source (launch) & destination (latch) clocks

Skew between clock arrival at source and destination

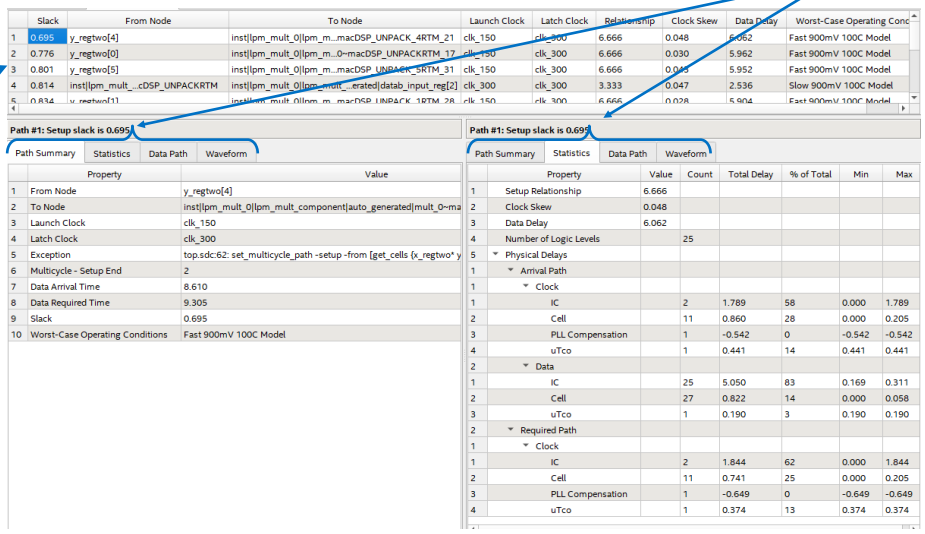



44

# Detailed Slack/Path Report

4 detailed views of selected path available

Select path from list for viewing



| | Slack | From Node | To Node | Launch Clock | Latch Clock | Relationship | Clock Skew | Data Delay | Worst-Case Operating Conc |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.695 | y_regtwo[4] | inst\|lpm_mult_0\|lpm_m...macDSP_UNPACK_4RTM_21 | clk_150 | clk_300 | 6.666 | 0.048 | 6.062 | Fast 900mV 100C Model |
| 2 | 0.776 | y_regtwo[0] | inst\|lpm_mult_0\|lpm_m...0~macDSP_UNPACKRTM_17 | clk_150 | clk_300 | 6.666 | 0.030 | 5.962 | Fast 900mV 100C Model |
| 3 | 0.801 | y_regtwo[5] | inst\|lpm_mult_0\|lpm_m...macDSP_UNPACK_5RTM_31 | clk_150 | clk_300 | 6.666 | 0.043 | 5.952 | Fast 900mV 100C Model |
| 4 | 0.814 | inst\|lpm_mult_...cDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_...erated\|datab_input_reg[2] | clk_300 | clk_300 | 3.333 | 0.047 | 2.536 | Slow 900mV 100C Model |
| 5 | 0.834 | y_regtwo[1] | inst\|lpm_mult_0\|lpm_m...macDSP_UNPACK_1RTM_28 | clk_150 | clk_300 | 6.666 | 0.028 | 5.904 | Fast 900mV 100C Model |

**Path #1: Setup slack is 0.695**

Path Summary | Statistics | Data Path | Waveform

| | Property | Value |
|---|---|---|
| 1 | From Node | y_regtwo[4] |
| 2 | To Node | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~ma |
| 3 | Launch Clock | clk_150 |
| 4 | Latch Clock | clk_300 |
| 5 | Exception | top.sdc:62: set_multicycle_path -setup -from [get_cells {x_regtwo* y |
| 6 | Multicycle - Setup End | 2 |
| 7 | Data Arrival Time | 8.610 |
| 8 | Data Required Time | 9.305 |
| 9 | Slack | 0.695 |
| 10 | Worst-Case Operating Conditions | Fast 900mV 100C Model |

**Path #1: Setup slack is 0.695**

Path Summary | Statistics | Data Path | Waveform

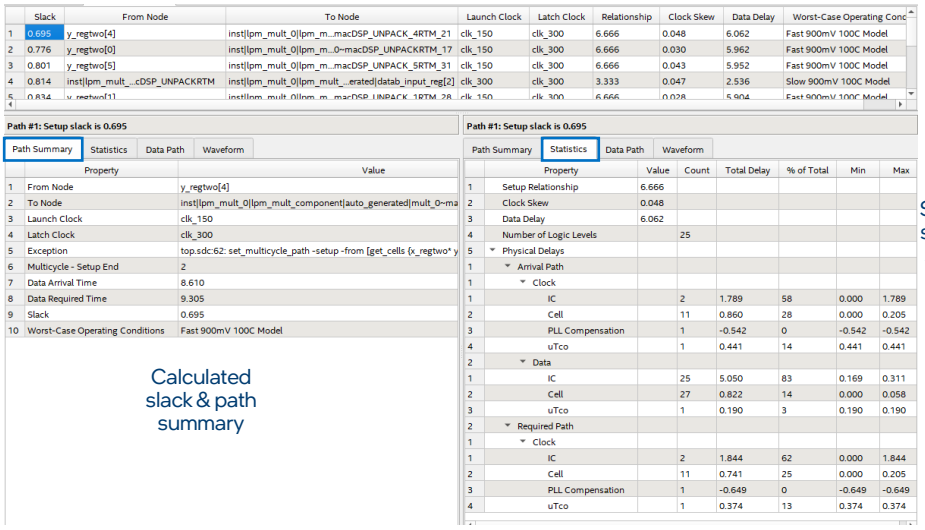
| | Property | Value | Count | Total Delay | % of Total | Min | Max |
|---|---|---|---|---|---|---|---|
| 1 | Setup Relationship | 6.666 | | | | | |
| 2 | Clock Skew | 0.048 | | | | | |
| 3 | Data Delay | 6.062 | | | | | |
| 4 | Number of Logic Levels | | 25 | | | | |
| 5 | Physical Delays | | | | | | |
| 1 | Arrival Path | | | | | | |
| 1 | Clock | | | | | | |
| 1 | IC | | 2 | 1.789 | 58 | 0.000 | 1.789 |
| 2 | Cell | | 11 | 0.860 | 28 | 0.000 | 0.205 |
| 3 | PLL Compensation | | 1 | -0.542 | 0 | -0.542 | -0.542 |
| 4 | uTco | | 1 | 0.441 | 14 | 0.441 | 0.441 |
| 2 | Data | | | | | | |
| 1 | IC | | 25 | 5.050 | 83 | 0.169 | 0.311 |
| 2 | Cell | | 27 | 0.822 | 14 | 0.000 | 0.058 |
| 3 | uTco | | 1 | 0.190 | 3 | 0.190 | 0.190 |
| 2 | Required Path | | | | | | |
| 1 | Clock | | | | | | |
| 1 | IC | | 2 | 1.844 | 62 | 0.000 | 1.844 |
| 2 | Cell | | 11 | 0.741 | 25 | 0.000 | 0.205 |
| 3 | PLL Compensation | | 1 | -0.649 | 0 | -0.649 | -0.649 |
| 4 | uTco | | 1 | 0.374 | 13 | 0.374 | 0.374 |





45

# Detailed Slack/Path Report



| | Slack | From Node | To Node | Launch Clock | Latch Clock | Relationship | Clock Skew | Data Delay | Worst-Case Operating Conc |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.695 | y_regtwo[4] | inst\|lpm_mult_0\|lpm_m...macDSP_UNPACK_4RTM_21 | clk_150 | clk_300 | 6.666 | 0.048 | 6.062 | Fast 900mV 100C Model |
| 2 | 0.776 | y_regtwo[0] | inst\|lpm_mult_0\|lpm_m...0~macDSP_UNPACKRTM_17 | clk_150 | clk_300 | 6.666 | 0.030 | 5.962 | Fast 900mV 100C Model |
| 3 | 0.801 | y_regtwo[5] | inst\|lpm_mult_0\|lpm_m...macDSP_UNPACK_5RTM_31 | clk_150 | clk_300 | 6.666 | 0.043 | 5.952 | Fast 900mV 100C Model |
| 4 | 0.814 | inst\|lpm_mult_...cDSP_UNPACKRTM | inst\|lpm_mult_0\|lpm_mult_...erated\|datab_input_reg[2] | clk_300 | clk_300 | 3.333 | 0.047 | 2.536 | Slow 900mV 100C Model |
| 5 | 0.834 | y_regtwo[1] | inst\|lpm_mult_0\|lpm_m...macDSP_UNPACK_1RTM_28 | clk_150 | clk_300 | 6.666 | 0.028 | 5.904 | Fast 900mV 100C Model |

**Path #1: Setup slack is 0.695**

Path Summary | Statistics | Data Path | Waveform

| | Property | Value |
|---|---|---|
| 1 | From Node | y_regtwo[4] |
| 2 | To Node | inst\|lpm_mult_0\|lpm_mult_component\|auto_generated\|mult_0~ma |
| 3 | Launch Clock | clk_150 |
| 4 | Latch Clock | clk_300 |
| 5 | Exception | top.sdc:62: set_multicycle_path -setup -from [get_cells {x_regtwo* y |
| 6 | Multicycle - Setup End | 2 |
| 7 | Data Arrival Time | 8.610 |
| 8 | Data Required Time | 9.305 |
| 9 | Slack | 0.695 |
| 10 | Worst-Case Operating Conditions | Fast 900mV 100C Model |

Calculated slack & path summary

**Path #1: Setup slack is 0.695**

Path Summary | Statistics | Data Path | Waveform

| | Property | Value | Count | Total Delay | % of Total | Min | Max |
|---|---|---|---|---|---|---|---|
| 1 | Setup Relationship | 6.666 | | | | | |
| 2 | Clock Skew | 0.048 | | | | | |
| 3 | Data Delay | 6.062 | | | | | |
| 4 | Number of Logic Levels | | 25 | | | | |
| 5 | Physical Delays | | | | | | |
| 1 | Arrival Path | | | | | | |
| 1 | Clock | | | | | | |
| 1 | IC | | 2 | 1.789 | 58 | 0.000 | 1.789 |
| 2 | Cell | | 11 | 0.860 | 28 | 0.000 | 0.205 |
| 3 | PLL Compensation | | 1 | -0.542 | 0 | -0.542 | -0.542 |
| 4 | uTco | | 1 | 0.441 | 14 | 0.441 | 0.441 |
| 2 | Data | | | | | | |
| 1 | IC | | 25 | 5.050 | 83 | 0.169 | 0.311 |
| 2 | Cell | | 27 | 0.822 | 14 | 0.000 | 0.058 |
| 3 | uTco | | 1 | 0.190 | 3 | 0.190 | 0.190 |
| 2 | Required Path | | | | | | |
| 1 | Clock | | | | | | |
| 1 | IC | | 2 | 1.844 | 62 | 0.000 | 1.844 |
| 2 | Cell | | 11 | 0.741 | 25 | 0.000 | 0.205 |
| 3 | PLL Compensation | | 1 | -0.649 | 0 | -0.649 | -0.649 |
| 4 | uTco | | 1 | 0.374 | 13 | 0.374 | 0.374 |

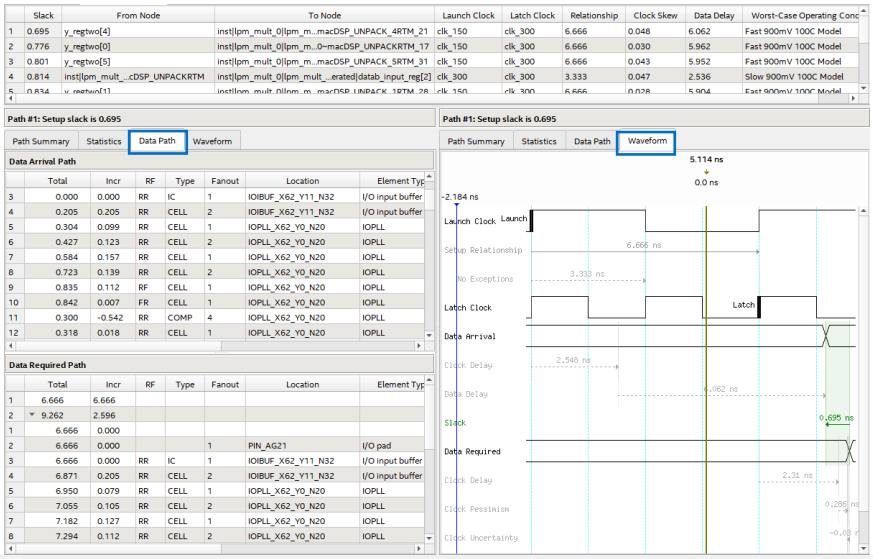
Statistics table shows general statistical info about path delays





46

# Detailed Slack/Path Report



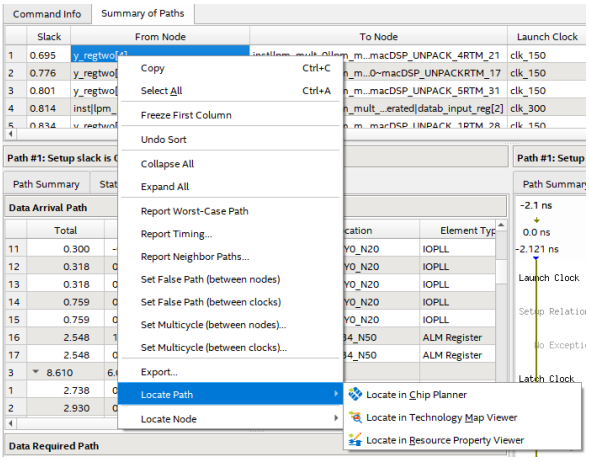







# Further Path Analysis

- Right-click path(s) to cross-probe to other Intel® Quartus® Prime Design Software tools or design files
  - **Locate Path**: view timing information in target tool
  - **Locate Node**: just highlights node/path in target tool or design file



## Other Example Reports Available in Timing Analyzer GUI

- Report All Summaries
  - Generates all slack and timing summary reports at once
- Report Timing Tree
  - Displays timing from design hierarchy perspective
- Report Bottlenecks
  - Displays design bottlenecks based on user criteria
- Report Metastability Summary
  - Displays results of metastability analysis (Mean Time Between Failures or MTBF) of synchronization register chains
- Report Skew
  - Displays results of skew analysis on selected paths
- Report Neighbor Paths
  - Displays timing critical paths along with timing critical neighboring paths




49

## Other Example Reports Available in Timing Analyzer GUI (cont.)

- Other reports discussed later and in follow-up timing optimization class (ask instructor for details)
- See Intel® Quartus® Prime Pro Edition User Guide: Timing Analyzer for detailed list and description of other reports




50

## Other Timing Analyzer In Session Use Cases

- Modifying project SDC file during Timing Analyzer session
- Applying new SDC constraints during Timing Analyzer session

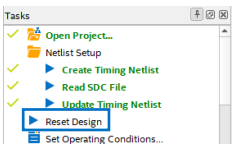
- Care should be used as constraints added or changed during Timing Analyzer session not reflected in compilation
  - Re-compilation necessary if new place and route needed due to change




51

## Modifying Project SDC File During Session

1. Modify and save file
2. Reset Design (Tasks pane)



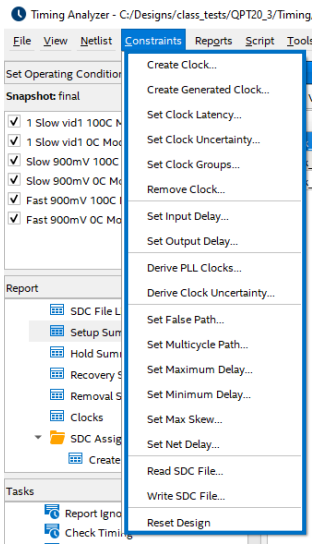

   - Flushes all timing constraints from current timing netlist and recreates original netlist
   - Otherwise unchanged and updated constraints applied on top of existing constraints and warnings generated
3. Double-click on report in Tasks pane to regenerate any reports
   - SDC files automatically re-read
   - Netlist automatically updated




52

# Applying New SDC Constraints During Session (1)

- New constraints may be loaded during an active Timing Analyzer session
  - Using Constraints menu
  - Entering SDC command in Console pane
  - Reading non-project SDC file (Read SDC File in Constraints menu)
- Useful to immediately see constraint's affect on the netlist before modifying project SDC file
- Use remove_<command> to remove an applied constraints before applying new one



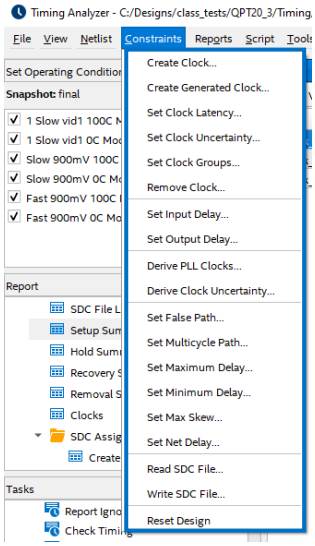




53

# Applying New SDC Constraints During Session (2)

- Constraints entered this way will only be applied to the netlist
  - No automatic transfer to project SDC file
  - Use Write SDC file Task/command to have Timing Analyzer write out single SDC file with all currently loaded constraints
    - Copy and paste commands you want to retain into project SDC (do not use SDC file generated by Write SDC File directly in project)



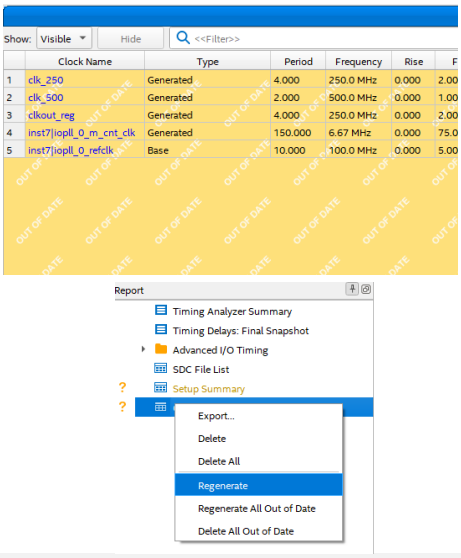




54

## Steps for Applying New SDC Constraints During Session

1. Read in new constraint(s)
   - Adding new constraints interactively in GUI or Console pane causes current reports to appear as "out of date"
2. Run Update Timing Netlist (Tasks pane) to apply new constraints
3. Run new reports and/or right-click on report in Report pane and select Regenerate or Regenerate All Out of Date







55

# Intel Quartus Prime Pro Timing Analysis

## Writing SDC Files




56

## Importance of Constraining (Again)

- Timing analysis reports how a circuit will behave in your system
- Providing SDC timing constraints tells tools how you want the design to behave
  - Describe the system environment (i.e. board, other devices, connections) and how design should operate in that system
    - Based on design specs & specs from the PCB and other devices on PCB
  - Provide goals for Fitter to target during compilation
  - Provide values for comparing results




57

## Effects of Incorrect SDC Files

- Decreases timing performance
  - Fitter tries to optimize most critical paths (worst timing slack) first
  - Non-critical or unnecessary paths with bad timing prevent Fitter from working on "real" critical paths
- Increases Fitter processing time
  - Fitter tries to achieve incorrect or even impossible timing values
  - Constraints may negate or override others, causing mismatches
- Increases timing analysis processing time
  - Unnecessary paths analyzed during timing signoff
  - Duplicate constraints reprocessed unnecessarily




58

## Writing SDC Files – Section Objectives

- Learn common SDC commands and arguments used in Intel® FPGA designs
- Write valid SDC files for common FPGA design situations




59

## SDC References

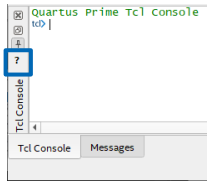
- Intel® Quartus® Prime Pro Edition User Guide: Timing Analyzer
- Intel Quartus Prime Pro Edition online help
  - https://www.intel.com/content/www/us/en/programmable/quartushelp/current/index.htm#tafs/tafs/tcl_pkg_sdc_ver_1.5.htm
  - https://www.intel.com/content/www/us/en/programmable/quartushelp/current/index.htm#tafs/tafs/tcl_pkg_sdc_ext_ver_1.0.htm
- Intel Quartus Prime Timing Analyzer Cookbook
- Tcl and Command Line Help online tool (QHelp)
  - Installed with the Intel Quartus Prime Pro software




60

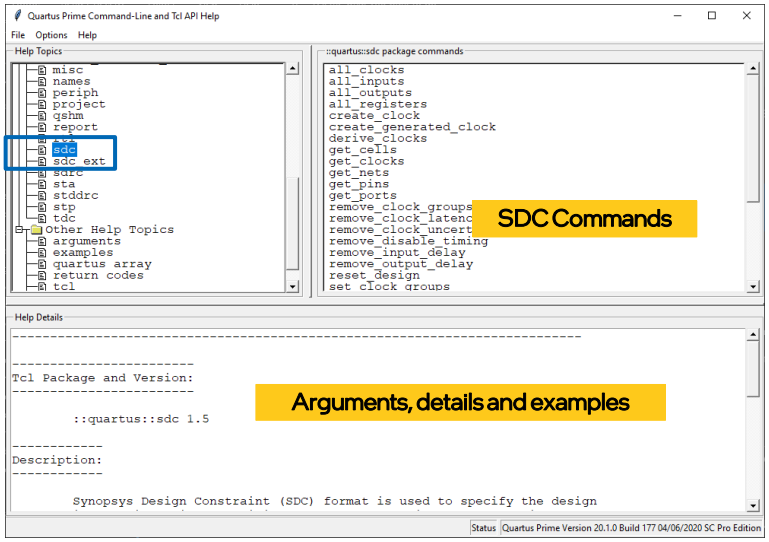
# SDC References – Tcl and Command Line Help

*Launch from Tcl Console*




*Launch from Command Line*

C:\<install dir>\quartus\bin64\qpro_sh –qhelp








61

# Writing SDC Constraints – Section Agenda

- Collections
- Clock constraints
- I/O interfaces
- Timing exceptions





62

# Writing SDC Constraints

## Collections




63

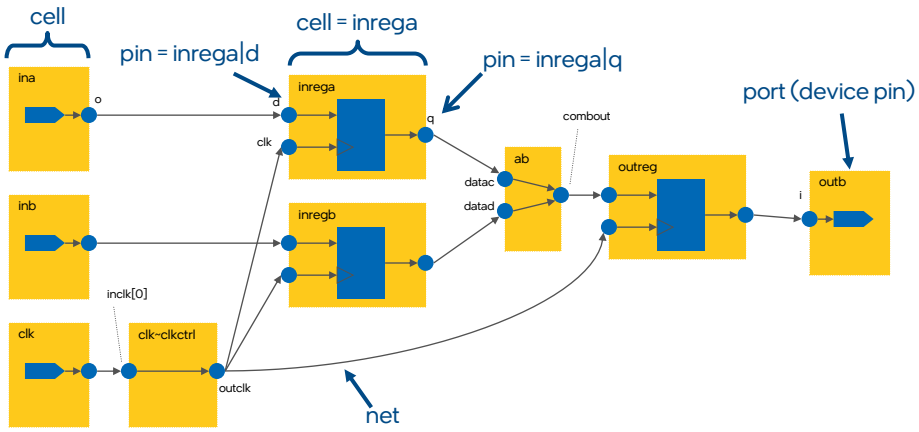
## SDC Netlist Terminology

| Term | Definition |
|---|---|
| Cell | Device building blocks (e.g. look-up tables, registers, embedded multipliers, memory blocks, I/O elements, PLLs, etc.) |
| Pin | Input or outputs of cells |
| Net | Connections between pins |
| Port | Top-level inputs and outputs (e.g. device pins) |
| Registers | Registers in a design (subset of cells) |
| Keepers | Non-combinational design nodes (superset of I/O and registers); timing paths begin and end with keepers |




64

# SDC Netlist Example




- Target design paths by applying constraints to path endpoints (or clocks)
  - i.e. ports, cells, pins, nets, registers





65

# SDC Naming Conventions

| Term | Naming Convention | Examples (from SDC Netlist Example slide) | |
|---|---|---|---|
| Port | <io_name> | ina<br>inb | clk<br>outb |
| Cell | <hierarchy_path>\|<cell_name> | ina~input<br>clk~input<br>inrega | ab<br>outreg<br>outb~output |
| Pin | <hierarchy_path>\|<cell_name>\|<pin_name> | ina~input\|o<br>clk~input\|o<br>inrega\|d<br>inrega\|q | ab\|datac<br>ab\|combout<br>outreg\|clk<br>outb~output\|i |
| Net | <hierarchy_path>\|<cell_output_name> | ina~input<br>ab | clk~ctrl<br>inrega |
| Registers | <hierarchy_path>\|<cell_register_name> | inrega<br>inregb<br>outreg | |
| Keepers | <io_name><br><hierarchy_path>\|<cell_register_name | ina<br>inb<br>inrega | inregb<br>outreg<br>outb |

- '|' used for hierarchy, cell and pin separation





66

## Collections

- Search the timing netlist and return a list of names meeting criteria (i.e. collection type and optional character string or wildcard)
- Use as targets in SDC commands to target correct objects type
- Recommendation: include collections to ensure constraint targets the correct object (and to make constraint file more readable)

```
create_clock -name sys_clk -period 20.0 -waveform {10.0 20.0} \
          [get_ports sys_clk]
set_multicycle_path -from [get_clocks sys_clk] -to [get_clocks sys_clk] 2
```




## Collection Examples

- get_clocks
- get_ports
- get_keepers
- get_cells
- get_pins
- get_registers
- get_nets
- all_clocks
- all_registers
- all_inputs
- all_outputs

## Using Collections in SDC Constraints

- `get_clocks`
- `get_ports`
- `get_registers`
- `get_keepers`

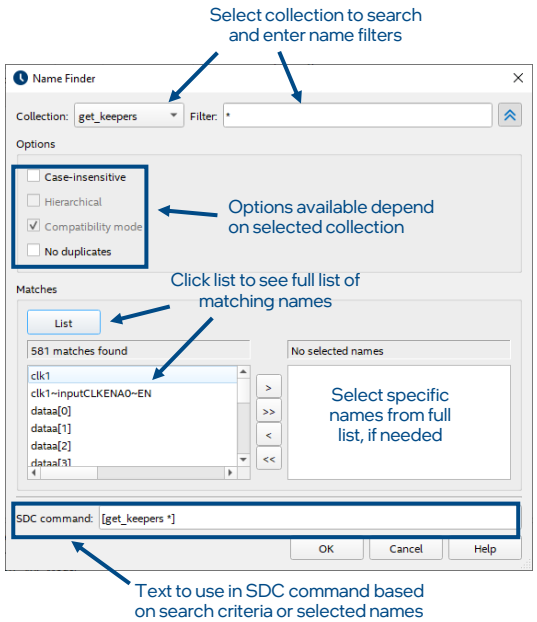
- Most of the FPGA design constraints can be managed with these collections
- Constraining certain logic structures may require more specific collection (e.g. `get_pins`)
- Examples shown in remainder of class




69

## Name Finder

- Graphical tool in Timing Analyzer that aids in defining collection search criteria
- Access from
  - Browse button ... in any constraint dialog box
  - Timing Analyzer View menu



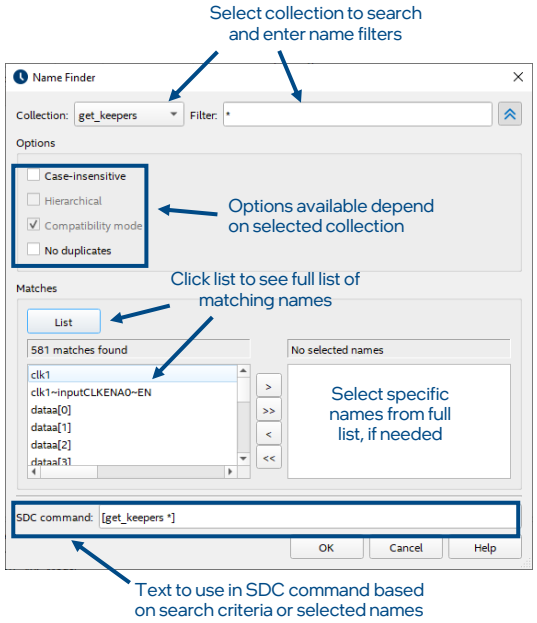




70

# Name Finder Uses

- Locate names of nodes to use in SDC commands
  - Combinational node names (if needed) may be challenging to find
- Ensure use of correct collection syntax
  - Some collection arguments may be tricky to get right
- Verify collection search criteria captures desired node names
  - See Intel® Quartus® Prime Pro Edition User Guide: Timing Analyzer for more details on collection search criteria







71

# Writing SDC Constraints

## Clock Constraints




72

## SDC Clock Constraints

- Good starting point for any design SDC file is to constrain clocks
- Once clocks defined, all or nearly all core design paths are completely constrained




73

## SDC Clock Constraints Agenda

- Create constraints for
  - Base clocks
  - Generated clocks
  - PLL clocks
- Constrain clock uncertainty
- Clock groups




74

## SDC Clocks

- Defined, repeating signal properties applied to a design location (target)
  - Typically ports, pins, registers and keepers
- Create clock domains fed by constraint target (or output of constraint target)
- Define edge to edge timing relationships that serve as the basis for all timing analysis
- Typical starting point for most design SDC files




75

## Clock Types

- Clock
  - Alternate name(s): base clock; absolute clock
  - Frequency or phase always as defined
  - Frequency or phase not dependent on any other defined clock
- Generated clock
  - Alternate name(s): derived clock
  - Frequency or phase based on another previously defined clock or generated clock (source)
  - Relation to source clock either implied (x1) or explicitly defined
  - Apply to register output used as clock or output of logic function that modifies clock input
    - e.g. PLLs, output clocks




76

# `create_clock` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Creates a base clock domain used for constraining compilation and/or for timing analysis |
| Common Arguments | [-name <clock_name>]<br>-period <time_or_freq><br>[-waveform {<rise_time> <fall_time>}]<br>[<target>]<br>[-add] |

[ ] = optional argument (under all or select conditions)





77

# `create_clock` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| [-name <clock_name>] | • Assigns user defined name to clock domain to be used in other constraints and in timing reports<br>• Defaults to target name if unused |
| -period <value> | • clock period defined in time (ns - default) or frequency* (e.g "300 MHz") |
| [-waveform {rise_time, fall_time}] | • Specifies when in time the rising and falling edges of the clock occur for defining clock waveform, assuming clocking starts at 0 time (default)<br>• Default = {0, period/2} (i.e. clock rises at 0 and falls at period/2)<br>• Use to define phase shifted (0 < rise_time ≤ period)<br>• Use to define non-50% duty-cycle clocks (fall_time – rise_time ≠ period/2) |
| [<-target>] | • Point in FPGA to be used as source of this clock domain during analysis<br>• Apply to port, register, keeper, pin or net |
| [-add] | • Allows multiple clocks to be assigned to same point without overwriting or ignoring (simultaneous multi-frequency analyses)<br>• Default: clock with **same** name assigned to point with existing clock overwrites existing clock (warning given)<br>• Default: clock with **different** name assigned to point with existing clock is ignored (warning given) |



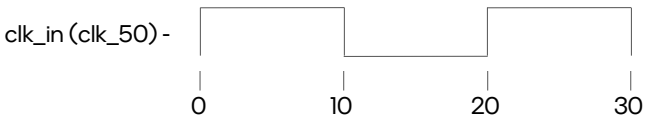
* non-standard SDC



78

# create_clock Examples

```
create_clock -period 20.0 -name clk_50 [get_ports clk_in]
    # 50 MHz clock domain named clk_50
```



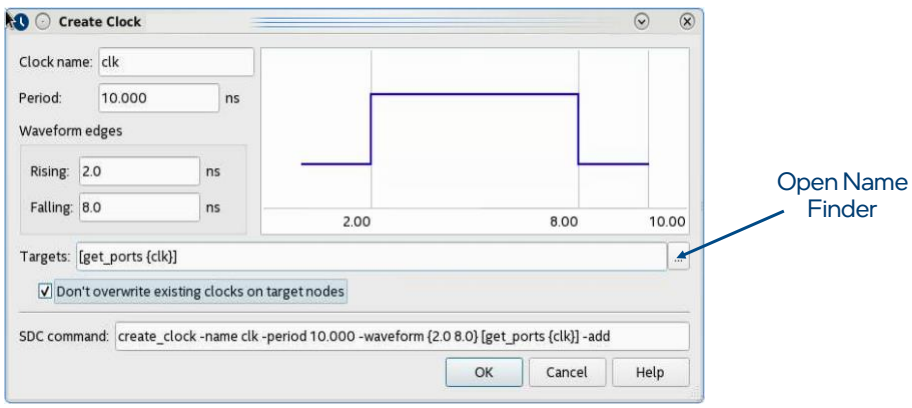

```
create_clock -period 100Mhz -waveform {2.0 8.0} [get_ports sysclk]
    # 100 MHz, 60% duty cycle clock shifted by 2 ns
```





79

# create_clock Dialog Box for Constraint Entry




Edit any field
(change values; use wildcards in targets or command)




80

# `create_generated_clock` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Creates an internally generated, derived clock domain used for constraining compilation and/or for timing analysis<br>• Defines relationship with source clock domain |
| Common Arguments | [-name <clock_name>]<br>-source <clock_source><br>[-master_clock <clock_name>]<br>[-divide_by <factor>]<br>[-multiply_by <factor>]<br>[-duty_cycle <percent>]<br>[-invert]<br>[-phase <degrees>]<br>[<targets>]<br>[-add] |

[ ] = optional argument (under all or select conditions)



intel

81

81

# `create_generated_clock` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| [-name <clock_name>] | • Same as create_clock command |
| -source <clock_source> | • Specifies location in design at which the clock waveform is used as reference for derived clock<br>• Source must be a port, pin, register, keeper or net and should be the target of a clock constraint |
| [-master_clock <clock_name>] | • Chooses correct source clock domain if multiple domains exist at <clock_source> location (i.e. [-add] was used when creating clocks) |
| [<-target>] | • Same as create_clock command |
| [-add] | • Same as create_clock command |



intel

82

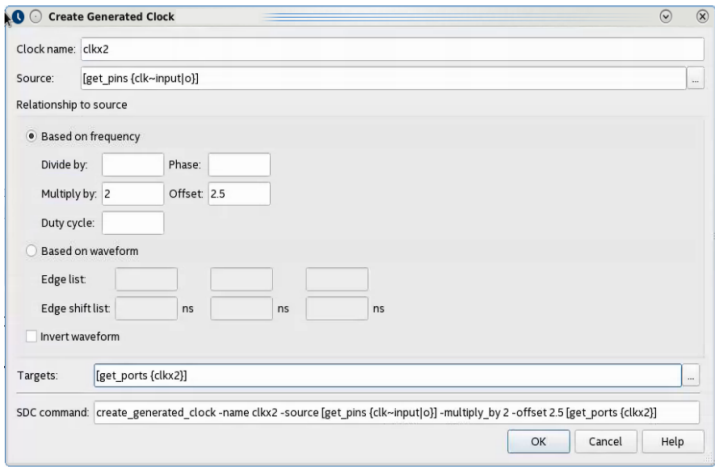
82

# `create_generated_clock` Command Argument Details

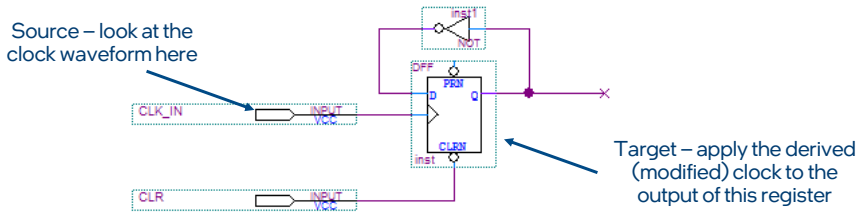
| Argument | Argument Definition and Details |
|---|---|
| `[-divide_by <factor>]` | • Divides source clock frequency by <factor> to determine derived clock frequency |
| `[-multiply_by <factor>]` | • Multiplies source clock frequency by <factor> to determine derived clock frequency |
| `[-duty_cycle <percent>]` | • Sets duty cycle of derived clock to <percent><br>• Derived clock duty cycle defaults to 50% |
| `[-invert]` | • Inverts edges of source clock to create derived clock |
| `[-offset <time>]` | • Shifts first edge of derived clock by the value of <time> with respect to source clock first edge<br>• First edge of source and derived clocks aligned by default |
| `[-phase <degrees>]` | • Shifts phase of the derived clock by <degrees> with respect to derived clock period |




# `create_generated_clock` Dialog Box for Constraint Entry



# Generated Clock Example 1



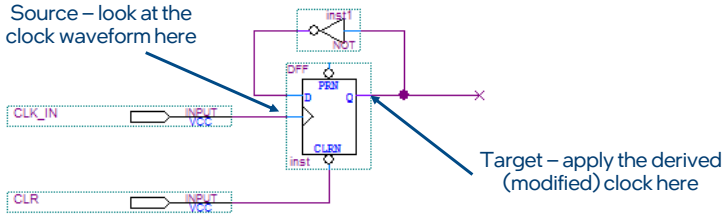

```
create_clock -name clk_in -period 10 [get_ports clk_in]
create_generated_clock -name clk_div -source [get_ports clk_in] -divide_by 2 \
          [get_registers inst]
```

*Line continuation*





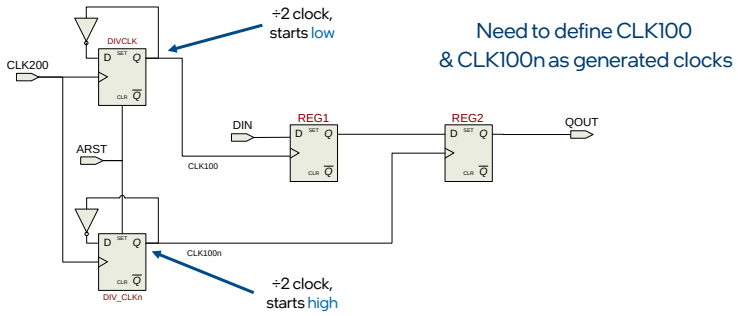
# Generated Clock Example 1 (Alternative)




```
create_clock -name clk_in -period 10 [get_ports clk_in]
create_generated_clock -name clk_div -source [get_pins inst|clk] -divide_by 2 \
          [get_pins inst|q]
```

- *Demonstrates another (less common but valid) way to write constraint based on pins*
- *Produces the same constraint as the previous slide*
- *Difference: clock constraint automatically changes if register connected to different clock in RTL*

# Inverted Clock Example




```
create_clock -name clk200 -period 5 [get_ports clk200]

create_generated_clock -name clk100 -source [get_ports clk200] \
          -divide_by 2 [get_registers divclk]
create_generated_clock -name clk100n -source [get_ports clk200] \
          -divide_by 2 -invert [get_registers div_clkn]
```




87

# PLL Clocks

- Intel® Quartus® Prime software automates constraining designs with internal PLLs using the user-defined settings contained in PLL or IO-related IP
  - I/O PLLs and fPLLs
  - Transceiver PLLs
  - Memory Interface PLLs
- What constraints are created?
  - Reference clock port/pin constrained with base clock
  - Output clocks constrained with generated clock




88

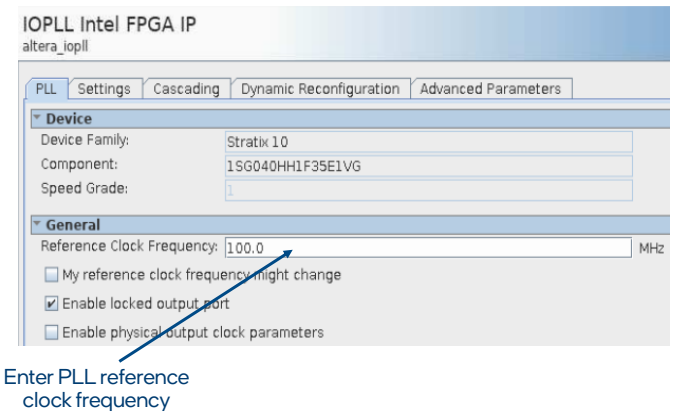
# PLL Constraint Methodology

- Intel Agilex™ and Intel Stratix® 10 FPGAs and later families
  - Automatic constraint generation for all PLL input and output clocks
- Older device families
  - `derive_pll_clocks` constraint used to generate all PLL input and output clocks
- (Optional) Define names in IOPLL IP core parameter editor to use for PLL output clock constraint name arguments
  - Other clock names come from internal cell or pin name connected to clock output




89

# PLL Parameter Editor (IOPLL IP)

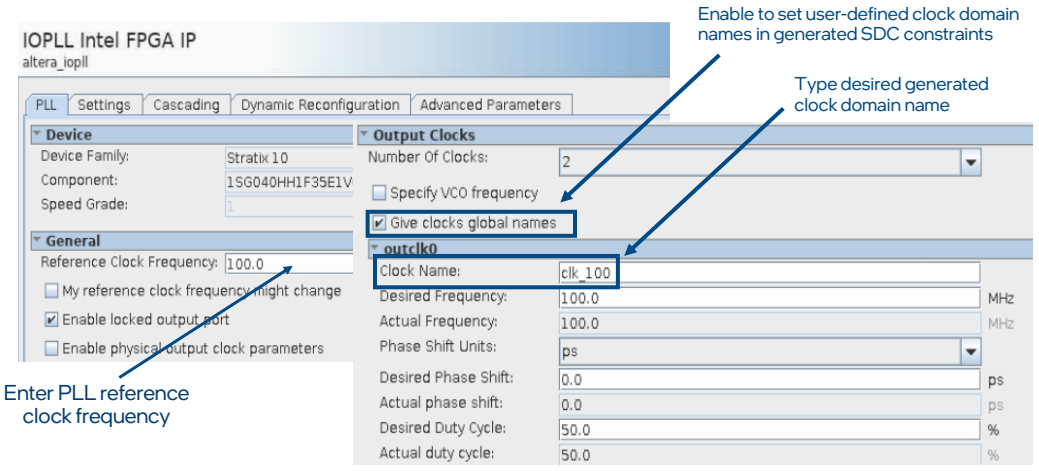



Enter PLL reference clock frequency




90

# PLL Parameter Editor (IOPLL IP)

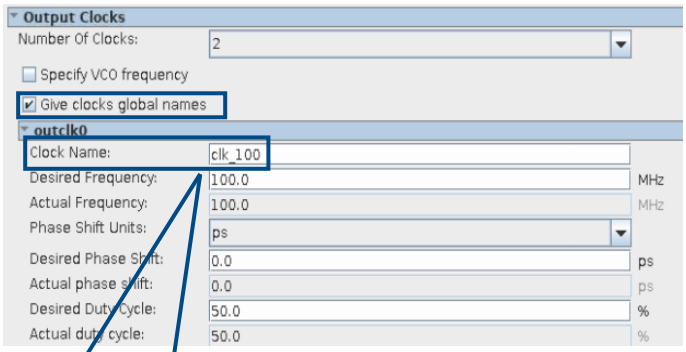







91

# PLL Parameter Editor (IOPLL IP)




Timing Analyzer clocks

| Clock Name | Type | Period | Frequency | Rise | Fall | Duty Cycle | Divide by | Multiply by |
|---|---|---|---|---|---|---|---|---|
| clk_100 | Generated | 10.000 | 100.0 MHz | 0.000 | 5.000 | 50.00 | 16 | 16 |
| clk_200 | Generated | 5.000 | 200.0 MHz | 0.000 | 2.500 | 50.00 | 8 | 16 |
| inst7\|iopll_0_m_cnt_clk | Generated | 160.000 | 6.25 MHz | 0.000 | 80.000 | 50.00 | 16 | 1 |
| inst7\|iopll_0_refclk | Base | 10.000 | 100.0 MHz | 0.000 | 5.000 | | | |





92

# PLL Clock Constraints (Intel® Agilex™ and Intel Stratix® 10 FPGAs and later families)

- No PLL clock constraints required in SDC
- Clocks constraints automatically created on input reference clock and all output clocks based on PLL settings





93

# Example Auto SDC Clock Constraints for PLL

```
#**************************************************************
# Create Clock
#**************************************************************
create_clock -name {pll_inst|iopll_0_refclk} -period 10.000 -waveform { 0.000 5.000 } [get_ports {clk}]

#**************************************************************
# Create Generated Clock
#**************************************************************
create_generated_clock -name {pll_inst|iopll_0_m_cnt_clk} -source [get_ports {clk}] \
   -duty_cycle 50/1 -multiply_by 1 -divide_by 16 -master_clock {inst7|iopll_0_refclk} \
   [get_registers {pll_inst|iopll_0|stratix10_altera_iopll_i|s10_iopll.fourteennm_pll~mcntr_reg}]

create_generated_clock -name {clk_200} -source [get_ports {clk}] \
   -duty_cycle 50/1 -multiply_by 16 -divide_by 8 -master_clock {pll_inst|iopll_0_refclk} \
   [get_pins {pll_inst|iopll_0|stratix10_altera_iopll_i|s10_iopll.fourteennm_pll|outclk[0]}]

create_generated_clock -name {clk_100} -source [get_ports {clk}] \
   -duty_cycle 50/1 -multiply_by 16 -divide_by 16 -master_clock {pll_inst|iopll_0_refclk} \
   [get_pins {pll_inst|iopll_0|stratix10_altera_iopll_i|s10_iopll.fourteennm_pll|outclk[1]}]
```

*Base clock automatically created from PLL IP input reference clock settings*

*Generated clocks automatically created from PLL IP settings*





94

## Manual Clock Constraint Entry (Override PLL Constraints) Uses

- Analyzing multi-clock or clock switching designs
  - PLL settings and configuration unchanged
- Manually naming clocks
  - Design contains IP that configure PLLs and create clocks but provide no means to name them
  - i.e. auto-generated SDC clocks results in long clock domain names




95

## Manual Clock Constraint Entry (Override PLL Constraints) Methods

- Override auto created input reference clock constraint only
  - Manually define PLL input reference clock constraint in SDC file
  - Output clock constraints created applying PLL IP multiply, divide and shift values to input clock constraint
- Override auto created input reference and output PLL clock constraints
  - Manually define PLL input reference and output clocks in SDC file




96

## derive_pll_clocks Command (Older Device Families)

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Creates generated clock constraints on all PLL outputs based on settings defined in PLL IP core(s)<br>• Does not overwrite existing clocks<br>• Creates multiple output clocks if PLL IP core switchover feature used |
| Common Arguments | [-create_base_clocks] |




97

## derive_pll_clocks Command Argument Details (Older Device Families)

| Argument | Argument Definition and Details |
|---|---|
| [-create_base_clocks] | • Instructs command to also create base clock on PLL input reference pin |




98

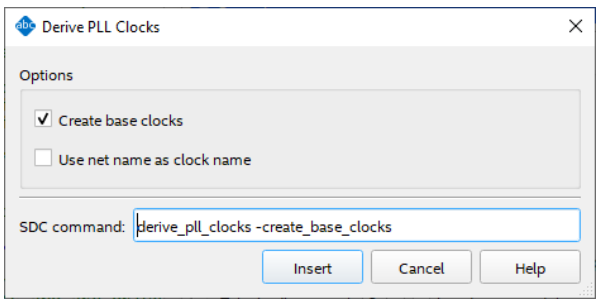
## derive_pll_clocks Usage

- derive_pll_clocks -create_base_clocks runs automatically if no clock constraints in SDC file
  - PLL input reference and output clocks constrained
  - Similar in behavior to automatic constraint creation in newer devices
- Should manually include derive_pll_clocks in SDC file if including other clocks constraints
  - derive_pll_clocks -create_base_clocks does not run automatically when other clocks defined in SDC file




99

## derive_pll_clocks Dialog Box for Constraint Entry

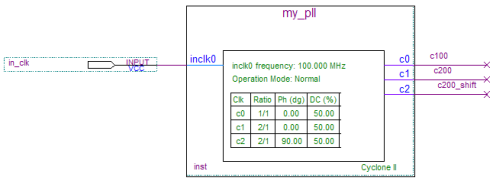






100

# derive_pll_clocks Examples




Derive PLL command with base clock argument

```
derive_pll_clocks -create_base_clocks
```

Derive PLL command with separate base clock constraint

```
create_clock -name clk_in -period 10.0 [get_ports in_clk]
derive_pll_clocks
```

Equivalent separate clock commands

```
create_clock -period 10.0 [get_ports in_clk]
create_generated_clock -name c100 \
    -source [get_pins {inst|altpll_component|pll|inclk[0]}] \
    -divide_by 1 \
    [get_pins {inst|altpll_component|pll|clk[0]}]
create_generated_clock -name c200 \
    -source [get_pins {inst|altpll_component|pll|inclk[0]}] \
    -multiply_by 2 \
    [get_pins {inst|altpll_component|pll|clk[1]}]
create_generated_clock -name c200_shift \
    -source [get_pins {inst|altpll_component|pll|inclk[0]}] \
    -multiply_by 2 \
    -phase 90 \
    [get_pins {inst|altpll_component|pll|clk[2]}]
```




101

# SDC Clock Uncertainty

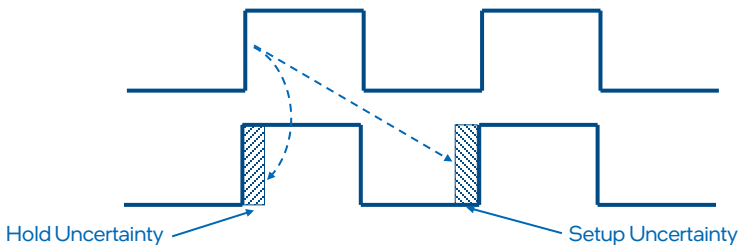
- Accounts for irregularities in clock waveform to create non-ideal clocks
  - e.g. clock tree jitter, PLL jitter, skew
  - Clock constraints define only regular, ideal clock waveforms
- Increases accuracy of timing analysis
- Reduces margins for setup and hold analysis
- Support for automatic and manual definition




102

## Clock Uncertainty Types

- Setup uncertainty decreases setup required time
  - Moves destination capturing clock edge earlier for setup analysis
- Hold uncertainty increases hold required time
  - Moves destination capturing clock edge later for hold analysis







103

## Automatically Derived Uncertainties

- Software automatically runs `derive_clock_uncertainty` command to include internal uncertainty values in analyses
  - Values determined from the timing model and/or characterization
  - Values incorporate datasheet parameters and operating conditions
- Applies uncertainty to inter-clock, intra-clock and I/O transfers
- View derived uncertainties with Report SDC (`report_sdc`) command
  - Results only seen after place and route




104

## Manually Added Uncertainty

- `set_clock_uncertainty` constraint
- Use to provide additional information on external clock for analysis
  - e.g. account for external jitter components from clock source in excess of device specs
- Use to add guard band to existing clock constraint
  - Tightens (over-constrains) setup and hold without changing target clock frequency




105

## `set_clock_uncertainty` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Defines an uncertainty or skew value to use for timing analysis and how the value should be applied |
| Common Arguments | [-setup \| -hold]<br>[-from <from_clock>]<br>[-to <to_clock>]<br>[-add]<br><uncertainty_value> |




106

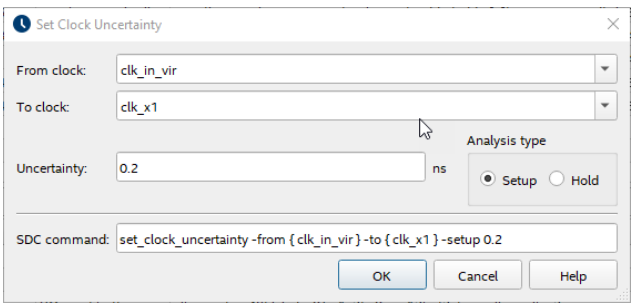
# `set_clock_uncertainty` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| `<uncertainty_value>` | • Value to use in uncertainty constraint |
| `[-add]` | • Adds the uncertainly value to the automatically derived value instead of the default which is to override it<br>• Not cumulative; if multiple constraints with `-add` to same clock, only one is processed |
| `[-from <from_clock>]` | • Applies uncertainty value only to paths that originate in `<from_clock>` domain |
| `[-to <to_clock>]` | • Applies uncertainty value only to paths that end in `<to_clock>` domain |
| `[-setup]` | • Applies uncertainty value only to setup analysis instead of the default which is to apply uncertainty value to both setup and hold |
| `[-hold]` | • Applies uncertainty value only to hold analysis instead of the default which is to apply uncertainty value to both setup and hold |




107

# Clock Uncertainty (GUI)

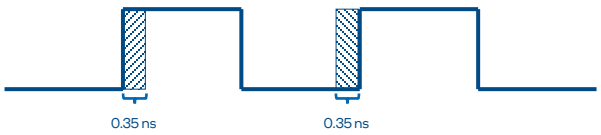






108

# set_clock_uncertainty Examples




*Both setup and hold analyses are tightened by additional uncertainty*

```
set clock_jitter-ptp 0.7
set_clock_uncertainty [exec $clock_jitter-ptp/2] -to [get_clocks clk_in] -add
    # Increases both setup and hold uncertainty values by 0.35 ns when clk_in is capturing clock
    #  i.e. increases both setup and hold required times by 0.35 ns
```


*Only setup analysis is tightened by additional uncertainty*

```
set_clock_uncertainty -setup 0.2 -to [get_clocks clk_in] -add
    # Implements 0.2 ns guard band when clk_in is capturing clock
    # Increases clock uncertainty to reduce data required time
    #      on all paths in the capturing clock domain by 0.2
```




109

# Clock Relationships

- All SDC clocks are related by default
- How is this possible?
  - At 0 time, all clocks assumed to "begin" toggling according to clock definitions
- What does this mean?
  - All clocks related by the common starting point (all edges aligned at 0, unless clocks defined with offset or shift)
  - All cross-domain transfers will be analyzed
    - Using smallest positive launch-to-latch edge relationship between clock waveforms
  - May need to adjust or "cut" inter-domain timing paths if default cross-domain analysis not valid for circuit operation




110

# `set_clock_groups` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Defines which clocks are not related and should not be analyzed together<br>• Timing analysis on all paths between selected domains is ignored (any direction) |
| Common Arguments | -group<br>-asynchronous \| -logically_exclusive \| -physically_exclusive |




111

# `set_clock_groups` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| -group <clock_name(s)> | • Clarifies which clock domains specifically are related and which are not related<br>• Each group argument to same `set_clock_groups` command creates a collection of related clocks that are unrelated to the clocks in other group arguments<br>  • Clock names listed within single group argument are related (analysis is performed on clock names within group)<br>  • Clock names listed in different groups are not related (analysis not performed between groups) |
| [-asynchronous] | • Specifies that clocks are active at same time but with no phase relationship (so timing between clocks cannot be accurately analyzed) |
| [-logically_exclusive] | • Specifies that clocks are physically present but not actively used at same time (e.g. internally muxed clocks) |
| [-physically_exclusive] | • Specifies that clocks are not physically present at the same time (e.g. externally muxed clocks) |




112

## `set_clock_groups` Notes

- Use `set_clock_groups` command to separate unrelated clock domains
- One `-asynchronous`, `-logically_exclusive` OR `-physically_exclusive` argument is required for each `set_clock_groups` command
  - Analyses and report values may change depending on which argument is used
- A clock may appear in only one group argument for single command




113

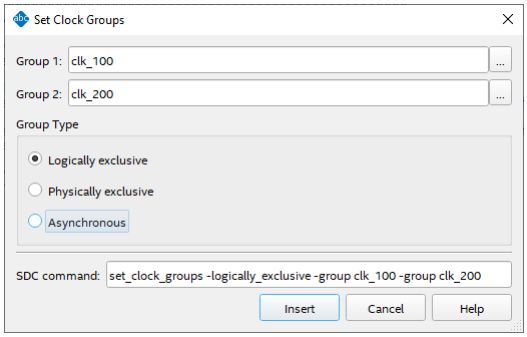
## `set_clock_groups` Notes (cont.)

- Use `set_clock_groups` command to separate unrelated clock domains
- Use as many group arguments with single `set_clock_groups` command as needed to define correct relationships
- Single group argument means all paths to/from any clock within group will not be analyzed to/from any clock not in group
- Clock not listed in any group remains related to all




114

# set_clock_groups Dialog Box for Constraint Entry

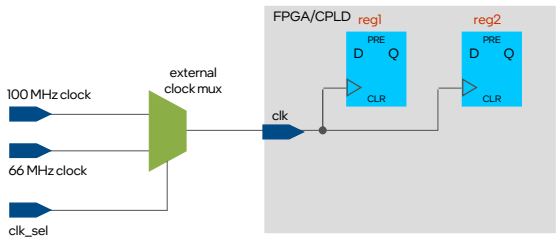



Two groups provided; additional groups must be added manually




115

# External Clock Mux Example

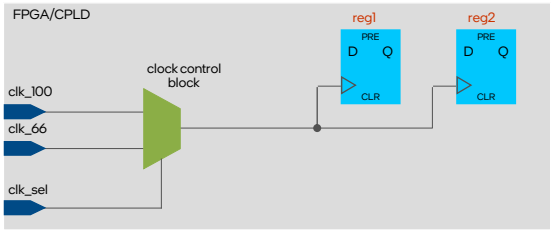



```
create_clock -name clk_100 -period 10.0 [get_ports clk]
create_clock -name clk_66 -period 15.0 [get_ports clk] -add

set_clock_groups -physically_exclusive \
   -group [get_clocks clk_100] \
   -group [get_clocks clk_66]

# Since clocks are muxed, cross-domain paths not analyzed
#   as only one clock will be driving registers at one time
# Since mux is external, their separation is physical
```




116

# Clock Mux Examples




```
create_clock -name clk_100 -period 10.0 [get_ports clk_100]
create_clock -name clk_66 -period 15.0 [get_ports clk_66]

create_generated_clock -name clkmux_100 \
          -source [get_ports clk_100 ] \
         [get_pins clkmux|clkout]
create_generated_clock -name clkmux_66 \
          -source [get_ports clk_66 ] \
         [get_pins clkmux|clkout] -add

set_clock_groups -logically_exclusive \
   -group [get_clocks clkmux_100] \
   -group [get_clocks clkmux_66]

# Since clocks are muxed, cross-domain paths not analyzed
#   as only one clock will be driving registers at one time
# Since mux is internal, their separation is logical
```






# Clock Grouping Examples

- Five clocks defined: `clk_A`, `clk_B`, `clk_C`, `clk_D`, `clk_E`

```
set_clock_groups -asynchronous \
   -group [get_clocks {clk_A clk_B}] \
   -group [get_clocks {clk_C clk_D}]
# Single clock groups command
```

| Analyzed? | | Destination | | | | |
|---|---|---|---|---|---|---|
| | | clk_A | clk_B | clk_C | clk_D | clk_E |
| Source | clk_A | Y | Y | | | Y |
| | clk_B | Y | Y | | | Y |
| | clk_C | | | Y | Y | Y |
| | clk_D | | | Y | Y | Y |
| | clk_E | Y | Y | Y | Y | Y |

```
set_clock_groups -asynchronous \
   -group [get_clocks {clk_A clk_B}]
set_clock_groups -asynchronous \
   -group [get_clocks {clk_C clk_D}]
# Two individual clock groups commands
```

| Analyzed? | | Destination | | | | |
|---|---|---|---|---|---|---|
| | | clk_A | clk_B | clk_C | clk_D | clk_E |
| Source | clk_A | Y | Y | | | |
| | clk_B | Y | Y | | | |
| | clk_C | | | Y | Y | |
| | clk_D | | | Y | Y | |
| | clk_E | | | | | Y |

# Verifying Clock Constraints

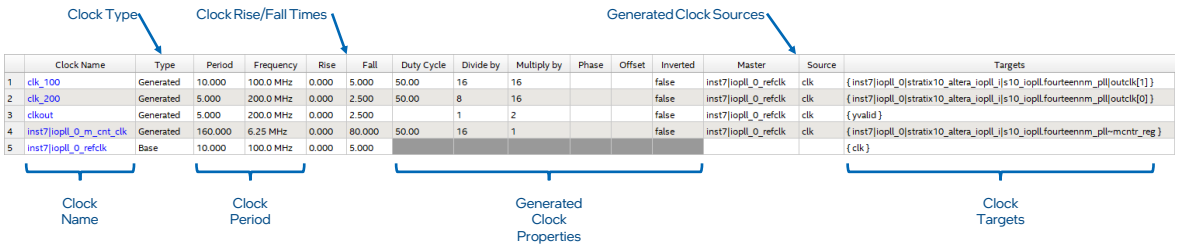
- Use these reports to manage the many related and unrelated clocks found in large FPGA designs
- Report Clocks
- Report Clock Waveforms
- Report Clock Hierarchy
- Report Unconstrained Paths
- Report SDC
- Report Clock Transfers
- Report CDC Viewer




119

# Report Clocks

- Displays clocks (all types) created automatically and from SDC commands and their detailed settings
- Use to verify/confirm clock parameters and relationships




| | Clock Name | Type | Period | Frequency | Rise | Fall | Duty Cycle | Divide by | Multiply by | Phase | Offset | Inverted | Master | Source | Targets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | clk_100 | Generated | 10.000 | 100.0 MHz | 0.000 | 5.000 | 50.00 | 16 | 16 | | | false | inst7\|iopll_0_refclk | clk | { inst7\|iopll_0\|stratix10_altera_iopll_i\|s10_iopll.fourteennm_pll\|outclk[1] } |
| 2 | clk_200 | Generated | 5.000 | 200.0 MHz | 0.000 | 2.500 | 50.00 | 8 | 16 | | | false | inst7\|iopll_0_refclk | clk | { inst7\|iopll_0\|stratix10_altera_iopll_i\|s10_iopll.fourteennm_pll\|outclk[0] } |
| 3 | clkout | Generated | 5.000 | 200.0 MHz | 0.000 | 2.500 | | 1 | 2 | | | false | inst7\|iopll_0_refclk | clk | { yvalid } |
| 4 | inst7\|iopll_0_m_cnt_clk | Generated | 160.000 | 6.25 MHz | 0.000 | 80.000 | 50.00 | 16 | 1 | | | false | inst7\|iopll_0_refclk | clk | { inst7\|iopll_0\|stratix10_altera_iopll_i\|s10_iopll.fourteennm_pll~mcntr_reg } |
| 5 | inst7\|iopll_0_refclk | Base | 10.000 | 100.0 MHz | 0.000 | 5.000 | | | | | | | | | { clk } |

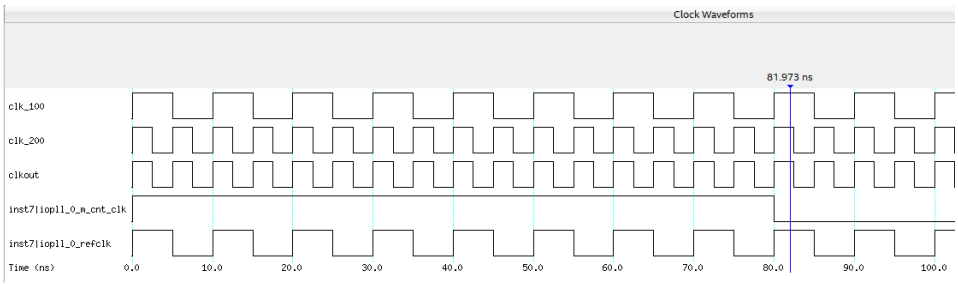
Tcl: `report_clocks`




120

# Report Clock Waveforms

- Provides visual inspection of clocks relationships
- Use to confirm expected clock edge relationships for timing paths




Tcl: `report_clocks -waveform`





121

# Report Clock Hierarchy

- Displays clock relationships in nested, hierarchical style
- Look for expected (and unexpected) hierarchical relationships)

Clock Hierarchy Summary

<<Filter>>

| | Clock Name | Type | Period | Frequency | Rise | Fall | Duty Cycle | Divide by | Multiply by | Phase | Offse | Inverted | Master | Source | Targets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | inst7\|iopll_0_refclk | Base | 10.000 | 100.0 MHz | 0.000 | 5.000 | | | | | | | | | { clk } |

| | Clock Name | Type | Period | Frequency | Rise | Fall | Duty Cycle | Divide by | Multiply by | Phase | Offse | Inverted | Master | Source | Targets |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | inst7\|iopll_0_refclk | Base | 10.000 | 100.0 MHz | 0.000 | 5.000 | | | | | | | | | { clk } |
| 1 | clk_100 | Generated | 10.000 | 100.0 MHz | 0.000 | 5.000 | 50.00 | 16 | 16 | | | false | inst7\|iopll_0_refclk | clk | { inst7\|iopll_0\|stratix10_altera_iopll_i\|s10_iopll.fourteennm_pll\|outclk[1] } |
| 2 | clk_200 | Generated | 5.000 | 200.0 MHz | 0.000 | 2.500 | 50.00 | 8 | 16 | | | false | inst7\|iopll_0_refclk | clk | { inst7\|iopll_0\|stratix10_altera_iopll_i\|s10_iopll.fourteennm_pll\|outclk[0] } |
| 3 | clkout | Generated | 5.000 | 200.0 MHz | 0.000 | 2.500 | | 1 | 2 | | | false | inst7\|iopll_0_refclk | clk | { yvalid } |
| 4 | inst7\|iopll_0_m_cnt_clk | Generated | 160.000 | 6.25 MHz | 0.000 | 80.000 | 50.00 | 16 | 1 | | | false | inst7\|iopll_0_refclk | clk | { inst7\|iopll_0\|stratix10_altera_iopll_i\|s10_iopll.fourteennm_pll~mcntr_reg } |

Tcl: `report_clocks -hierarchy`



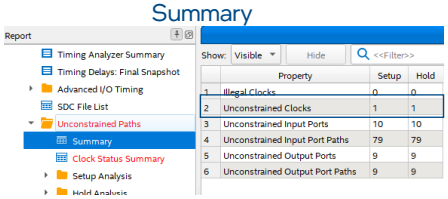


122

# Report Unconstrained Paths

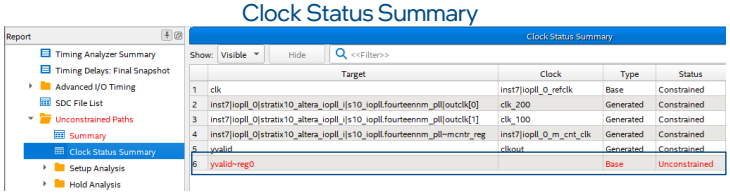
- Reports missing or invalid clocks
- Look for design clocks that are illegal or unconstrained

Summary




*Report indicates how many clock nodes are unconstrained (along with other unconstrained paths)*

Clock Status Summary




*Report lists each clock found and whether it was constrained*

Tcl: `report_ucp`

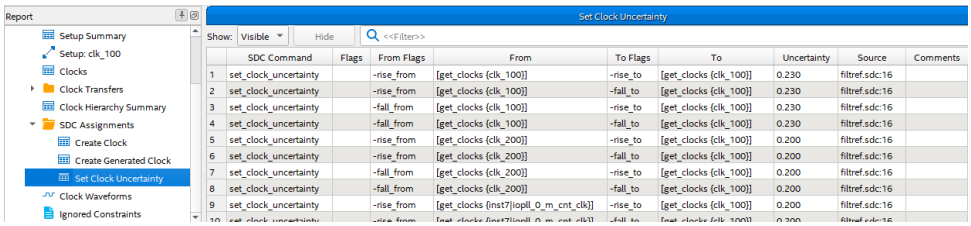






# Report SDC

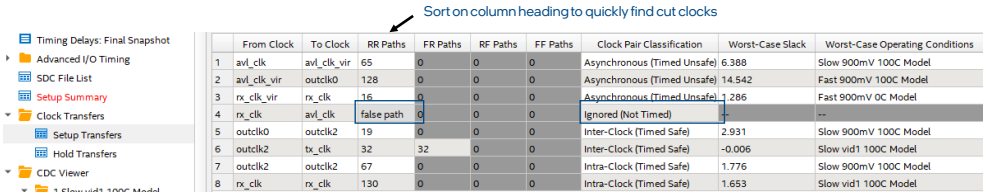
- Reports all SDC constraints currently applied to netlist
- Use to confirm uncertainty constraints read and applied correctly
  - Report also displays clock constraints




Tcl: `report_sdc`

# Report Clock Transfers

- Reports a table of all defined clocks and number of paths between them
  - Displays "false path" when clocks unrelated using `set_clock_groups`
- Use to quickly check if expected groups (un)analyzed




| | From Clock | To Clock | RR Paths | FR Paths | RF Paths | FF Paths | Clock Pair Classification | Worst-Case Slack | Worst-Case Operating Conditions |
|---|---|---|---|---|---|---|---|---|---|
| 1 | avl_clk | avl_clk_vir | 65 | 0 | 0 | 0 | Asynchronous (Timed Unsafe) | 6.388 | Slow 900mV 100C Model |
| 2 | avl_clk_vir | outclk0 | 128 | 0 | 0 | 0 | Asynchronous (Timed Unsafe) | 14.542 | Fast 900mV 100C Model |
| 3 | rx_clk_vir | rx_clk | 16 | 0 | 0 | 0 | Asynchronous (Timed Unsafe) | 1.286 | Fast 900mV 0C Model |
| 4 | rx_clk | avl_clk | false path | 0 | 0 | 0 | Ignored (Not Timed) | -- | -- |
| 5 | outclk0 | outclk2 | 19 | 0 | 0 | 0 | Inter-Clock (Timed Safe) | 2.931 | Slow 900mV 100C Model |
| 6 | outclk2 | tx_clk | 32 | 32 | 0 | 0 | Inter-Clock (Timed Safe) | -0.006 | Slow vid1 100C Model |
| 7 | outclk2 | outclk2 | 67 | 0 | 0 | 0 | Intra-Clock (Timed Safe) | 1.776 | Slow 900mV 100C Model |
| 8 | rx_clk | rx_clk | 130 | 0 | 0 | 0 | Intra-Clock (Timed Safe) | 1.653 | Slow vid1 100C Model |

** Report CDC Viewer report provides more in-depth and interactive clock domain transfer information.*

Tcl: `report_clock_transfers`

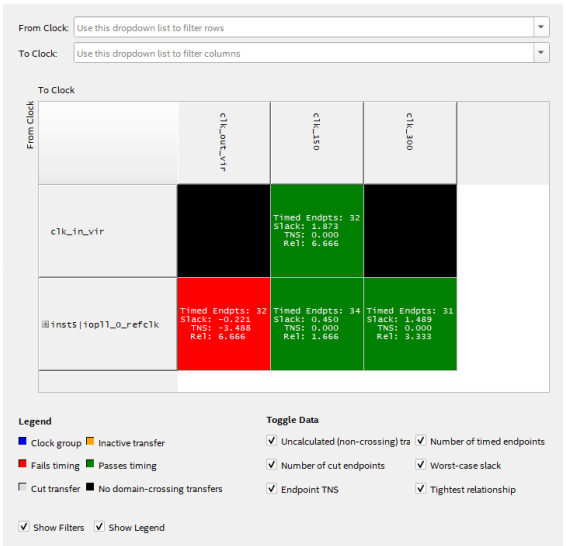






# Report CDC Viewer

- Displays clock transfers in an easy-to-read, interactive grid
- Color-coded to show passing & failing transfers, as well as fully-cut transfers & clock groups
- Can organize clocks in a tree structure based on their hierarchy
- Right-click to apply additional constraints to clock crossing



# Writing SDC Constraints

## I/O Interfaces




127

## I/O Interfaces Agenda

- Basic I/O interfaces
- Advanced I/O interfaces




128

## Basic I/O Interfaces

- I/O interfaces should be constrained when incoming or outgoing data has a known timing relationship to FPGA clock
  - Ensures success of data transfers between other devices
- Virtual clocks
- Synchronous input interfaces
- Synchronous output interfaces




129

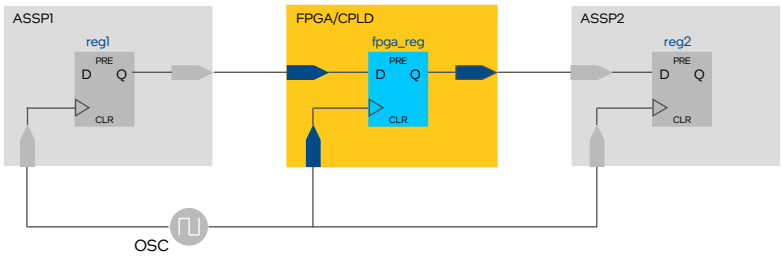
## Virtual Clock

- Definition: SDC clock defined without a target
  - Argument `-name` is required in definition
- Represents clock in system that does not directly interact with FPGA design
  - Clock interacts indirectly through other signals controlled by clock
- Use in I/O constraints to represent clock driving external device in system transferring data to/from FPGA




130

# System Diagram

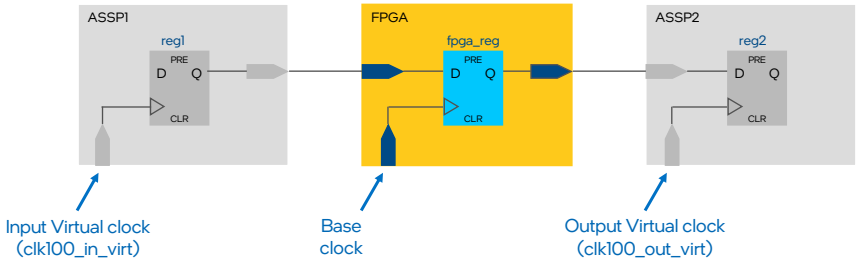



- Devices use common oscillator or reference clock
- ASSP1 and ASSP2 might be same device




131

# Virtual Clock Block Diagram Example




```
create_clock -name clk100_in -period 10.0 [get_ports in_clk]
create_clock -name clk100_in_virt -period 10.0
create_clock -name clk100_out_virt -period 10.0
```

*All related and aligned by default*

- Input virtual clock launches data from upstream device for capture by FPGA
- Output virtual clock captures data from FPGA into downstream device




132

## I/O Timing: Virtual Clocks

- Provide more accurate analysis of I/O timing
- Separating I/O clock from core clock means different numbers used for I/O transfer uncertainties than core uncertainties
  - I/O uncertainties smaller than core uncertainties since only fraction of timing path in FPGA
- Other benefits
  - Easier to identify I/O paths in timing reports by virtual clocks at launch or latch edge
  - In some cases (e.g. DDR), difficult (or not possible) to accurately constrain I/O without using virtual clock



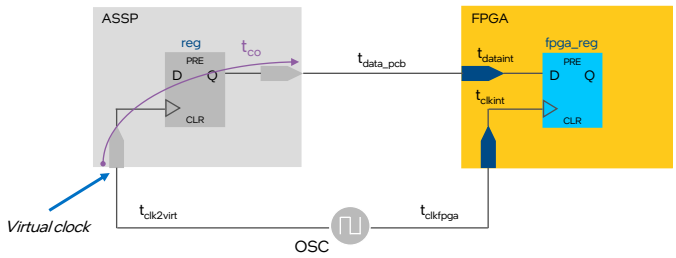

## Constraining Synchronous I/O

- Two methods to constrain I/O, depending on what I/O information is known

1. Using external timing parameters
   - Surrounding chips, board delays, chip-to-chip skews are known
   - Intel® Quartus® Prime design software derives FPGA I/O timing
2. Using FPGA timing requirements
   - Target $t_{su}$, $t_h$, $t_{co}$ specs for FPGA already known
   - Useful when environment or I/O standard provides the timing for the FPGA

# Synchronous Inputs

- Need to specify timing relationship from ASSP to FPGA to guarantee setup/hold met in FPGA
- Fitter adjusts placement and routing of input register to meet timing







135

# `set_input_delay` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Describes how much time is consumed external delays before arriving at input port<br>• Allows timing analyzer to determine timing needed to satisfy internal input register requirements |
| Common Arguments | [-clock <clock_name>]<br>[-max \| -min]<br><delay_value><br><targets><br>[-clock_fall] |




136

# `set_input_delay` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| `[-clock <clock_name>]` | • Identifies clock launching data from source device, usually input virtual clock |
| `[-max \| -min]` | • Indicates whether constraint is providing maximum delays (for setup or recovery) or minimum delays (for hold or removal)<br>• Same value used for both max and min if only one (or neither) is provided |
| `<delay_value>` | • Time value being used in constraint for max/min delay (as determined by equations or system) |
| `<targets>` | • Input I/O port names being constrained |
| `[-clock_fall]` | • Specifies that register in source device launches data on falling edge of clock<br>• Default: source register is rising edge triggered |




137

# Constraining Synchronous Inputs

- Use `set_input_delay` (`-max` option) to constrain FPGA input setup time (maximum time to arrive and still meet tsu)
  - Calculated maximum input delay value representing all external delays to device
- Use `set_input_delay` (`-min` option) to constrain FPGA input hold time (minimum time to stay active and still meet th)
  - Calculated minimum input delay value representing all external delays to device




138

# Calculating Input Delay Max/Min

| | 1. Using external timing parameters* | 2. Using FPGA timing requirements |
|---|---|---|
| Input delay (max) | **board delay (max) - board clock skew (min) + $t_{co(max)}$**<br>board delay (max) = $t_{data\_pcb(max)}$<br>board clock skew (min) = $t_{clkfpga(min)} - t_{clk2virt(max)}$ | **$t - t_{su}$**<br>$t = (t_{latch} - t_{launch})$<br>$t_{su}$ is required FPGA setup time |
| Input delay (min) | **board delay (min) - board clock skew (max) + $t_{co(min)}$**<br>board delay (min) = $t_{data\_pcb(min)}$<br>board clock skew (max) = $t_{clkfpga(max)} - t_{clk2virt(min)}$ | **$t_h$**<br>$t_h$ is required FPGA hold time |

** Timing parameters represented on prior Synchronous Inputs slide*

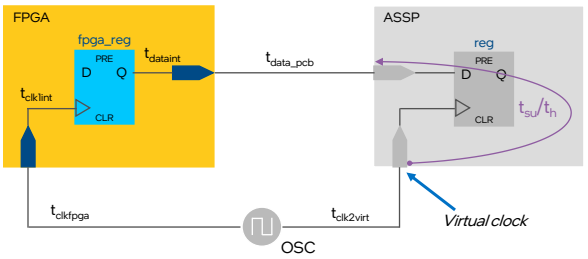




139

# Synchronous Outputs

- Need to specify timing relationship from FPGA to ASSP to guarantee setup/hold met in FPGA
- Fitter adjusts placement and routing of output register to meet timing








140

# `set_output_delay` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Describes how much time is consumed by external delays after leaving device output port<br>• Allows timing analyzer to determine if signal leaves FPGA with enough time to satisfy setup/recovery/hold/removal of external device |
| Common Arguments | [-clock <clock_name>]<br>[-max \| -min]<br><delay_value><br><targets><br>[-clock_fall] |




141

# `set_output_delay` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| [-clock <clock_name>] | • Identifies clock latching data into destination device, usually output virtual clock |
| [-max \| -min] | • Indicates whether constraint is providing maximum delays (for setup or recovery) or minimum delays (for hold or removal)<br>• Same value used for both max and min if only one (or neither) is provided |
| <delay_value> | • Time value being used in constraint for max/min delay (as determined by equations or system) |
| <targets> | • Output I/O port names being constrained |
| [-clock_fall] | • Specifies that register in destination device latches data on falling edge of clock<br>• Default: destination register is rising edge triggered |




142

## Constraining Synchronous Outputs

- Use **`set_output_delay`** (**`-max`** option) to constrain FPGA maximum clock-to-output (maximum time to leave and meet ASSP $t_{su}$)
  - Calculated maximum output delay value represents all delays external to device
- Use **`set_output_delay`** (**`-min`** option) to constrain FPGA minimum clock-to-output (minimum time to stay active and meet ASSP $t_h$)
  - Calculated minimum output delay value represents all delays external to device




143

## Calculating Output Delay Max/Min

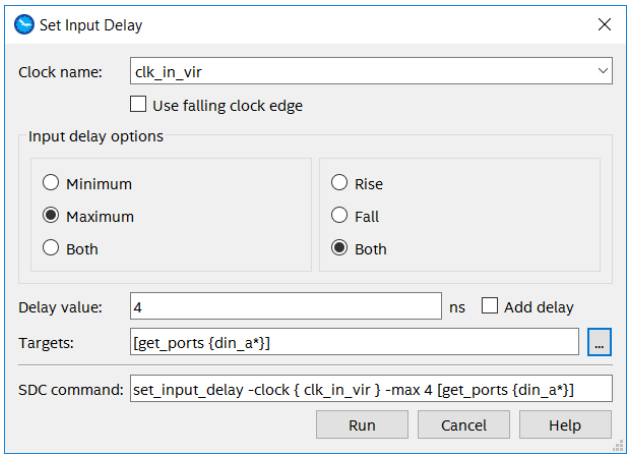
| | 1. Using external timing parameters* | 2. Using FPGA timing requirements |
|---|---|---|
| Output delay (max) | **board delay (max) - board clock skew (min) + $t_{su}$**<br><br>board delay (max) = $t_{data\_pcb(max)}$<br>board clock skew (min) = $t_{clk2virt(min)} - t_{clkfpga(max)}$ | **$t - t_{co(max)}$**<br><br>$t = (t_{latch} - t_{launch})$<br>$t_{co(max)}$ is required FPGA maximum clock-to-output time |
| Output delay (min) | **board delay (min) - board clock skew (max) - $t_h$**<br><br>board delay (min) = $t_{data\_pcb(min)}$<br>board clock skew (max) = $t_{clk2virt(max)} - t_{clkfpga(min)}$ | **$-t_{co(min)}$**<br><br>$t_{co(min)}$ is required FPGA minimum clock-to-output time |

** Timing parameters represented on prior Synchronous Outputs slide*




144

# Input/Output Delays (GUI)

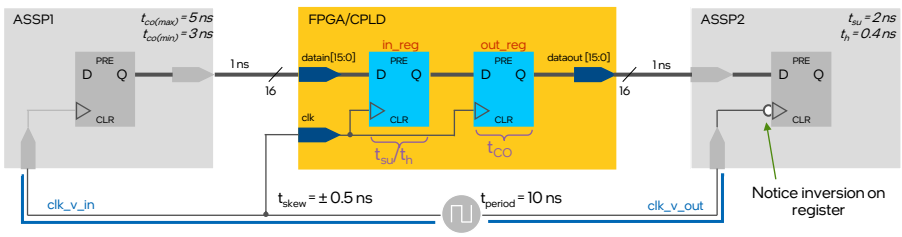









# Synchronous I/O Example




```
create_clock -name clk -period 10 [get_ports clk]
create_clock -name clk_v_in -period 10;  # virtual clock for input constraint
create_clock -name clk_v_out -period 10; # virtual clock for output constraint

set_input_delay -clock clk_v_in -max [expr {1 - (-0.5) + 5}] [get_ports datain*]
set_input_delay -clock clk_v_in -min [expr {1 - 0.5 + 3}]    [get_ports datain*]

set_output_delay -clock clk_v_out -max [expr {1 - (-0.5) + 2}] \
           -clock_fall [get_ports dataout*]
set_output_delay -clock clk_v_out -min [expr {1 - 0.5 - 0.4}] \
           -clock_fall [get_ports dataout*]
```

Note: `expr` in these constraints is used to simply calculate the value of the equation broken down into the 3 parts defined by the input/output delay equations

## Constrain I/O using Tcl Variables

- Use Tcl variables in SDC file for clarity and ease of reuse when constraining I/Os

```
# Enter device and PCB info related to the I/O
set tco_max 5.000
set tco_min 3.000
set td_max 1.0
set td_min 1.0
set longest_src_clk 1.0
set shortest_src_clk 0.5
set longest_dest_clk 1.0
set shortest_dest_clk 0.5

# Calculate the clock skew
set tcs_smallest [expr {$shortest_dest_clk - $longest_src_clk}]
set tcs_largest  [expr {$longest_dest_clk  - $shortest_src_clk}]

# Calculate the input min and max values
set input_max [expr {$td_max - $tcs_smallest + $tco_max}]
set input_min [expr {$td_min - $tcs_largest + $tco_min}]

# Create the input delay constraint
set_input_delay -max -clock vir_clk_in $input_max [get_ports datain]
set_input_delay -min -clock vir_clk_in $input_min [get_ports datain]
```




147

## Verifying I/O Constraints

- Use these reports to ensure I/O are constrained correctly

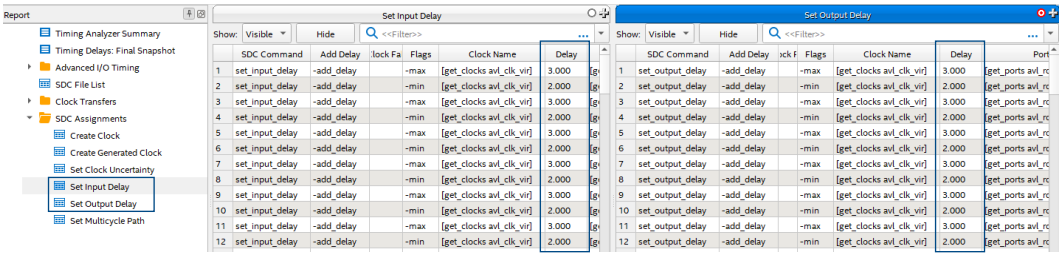
- Report SDC
- Report Unconstrainted Paths




148

# Report SDC (Review)

- Reports all SDC constraints currently applied to netlist
- Use to confirm I/O constraint delay values and ports




Tcl: `report_sdc`

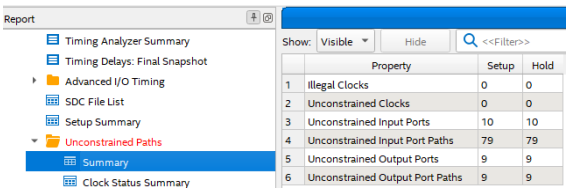


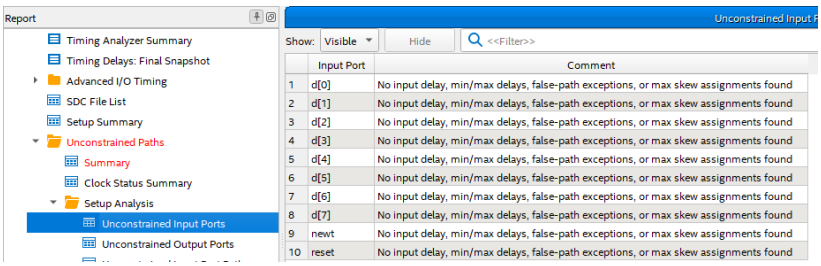




# Report Unconstrained Paths (Review)

- Reports missing I/O constraints
- Use to make sure all design ports are properly constrained



## Advanced I/O Interfaces

- Modern FPGAs have many types of advanced I/O interfaces with specific, pre-built hardware to implement these interfaces
  - e.g., external memory controller, serial transceivers
  - Synchronous I/O constraints and equations do not apply or apply easily
- Most Intel® FPGA IP used to configure these interfaces generate SDC files when specific constraints are needed
- See specific Intel® FPGA IP documentation for details or for any additional constraint requirements




151

## Example: External Memory Interface (EMIF) Hardware

- Use External Memory Interfaces Intel® FPGA IP
- Process
  1. Enter IP and board-level timing parameters during IP configuration
     - Parameters used to configure hardware settings
     - Parameters used to generate timing constraints for SDC file
  2. Add IP files to project including generated SDC file
- See External Memory Interfaces User Guide for target FPGA or EMIF technical training courses




152

# Example: High-Speed Source Synchronous Hardware

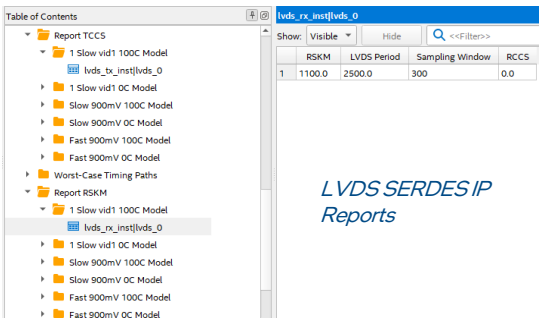
- Use LVDS SERDES Intel® FPGA IP core
- Process
  1. Enter IP parameters and board-level skew timing during IP configuration
     - Parameters used to configure hardware settings
     - Parameters used to generate timing constraints for SDC file
  2. Add IP files to project including generated SDC file
- See Intel FPGA High-Speed LVDS I/O User Guide for target FPGA



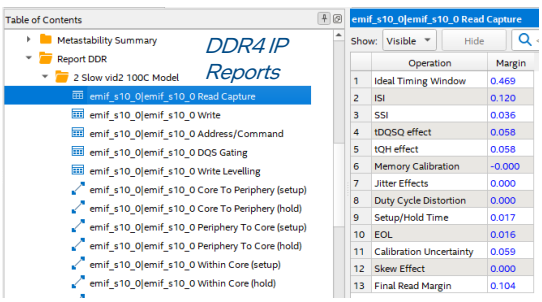

153

# IP-Specific Reports (Review)

- IP targeting FPGA-specific hardware resources generate special reports to check their timing
  - IP detected by software and included in summary reports
- Use to confirm specialized analysis margins are met for correct hardware operation




*LVDS SERDES IP Reports*




*DDR4 IP Reports*




154

# Writing SDC Constraints

## Timing Exceptions




155

## SDC Timing Exceptions

- Additional constraints to more accurately describe slack relationships according to the way the design works
  - Clock and I/O constraints alone not "good enough"
- Not required for fully constrained design, but their absence (when needed) may cause timing failures and/or an inaccurate timing analysis




156

## SDC Timing Exceptions (cont.)

- False paths
  - Perform no analysis on path
  - Sometimes referred to as "cut paths"
  - Fitter is free to place and route with no timing restrictions but still chooses as direct a path as possible
- Multicycle paths
  - Adjust clock relationship of path based on specified number of launch-to-latch edges
- Max/min delay paths
  - Apply arbitrary timing relationship to path using `set_max[min]_delay`
  - No deep discussion in class, see SDC references quoted earlier




157

## SDC Timing Exceptions Agenda

- False paths
- Multicycle paths
- Exception priorities
- Verifying exceptions




158

## Examples of Where Timing Exceptions Used

- Asynchronous input paths to synchronizer logic
- Paths between asynchronous clock domains
- Synchronization hardware
- Over-constraining individual path(s)
- Path requires > 1 cycle to complete transfer
- Enable controlled logic
- "Static" registers




159

## Note: Timing Exceptions Prevent Hyper-Retiming

- Hyper-Retiming (i.e. retiming using Hyper-Registers) is a key optimization in Intel® FPGAs with Intel Hyperflex™ architecture
  - e.g. Intel Agilex™ FPGAS; Intel Stratix® 10 FPGAs
- Hyper-Retiming not performed on registers/timing paths using timing exceptions
- High-performance designs should minimize timing exceptions on timing-critical logic to only what is needed to describe design behavior
  - Consider alternative functionally equivalent logic that does not require exception
  - e.g. Adding intermediate registers vs. using multi-cycle logic




160

## Timing Exceptions: False Paths

- Logic-based
  - Paths not relevant during normal circuit operation
  - e.g. test logic, static or quasi-static registers
- Timing-based
  - Paths intentionally not analyzed by designer
  - e.g. bridging asynchronous clock domains using synchronizer circuits
- Must be marked by constraint to tell Timing Analyzer to ignore them
- Remember: all clock domains are related by default!




161

## `set_false_path` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Defines paths that should be ignored during fitting and timing analysis<br>• Allows Fitter to prioritize other paths |
| Common Arguments | [-from <names>]<br>[-fall_from <clocks>]<br>[-rise_from <clocks>]<br>[-through <names>]<br>[-to <names>]<br>[-fall_to <clocks>]<br>[-rise_to <clocks>]<br>[-latency_insensitive] |




162

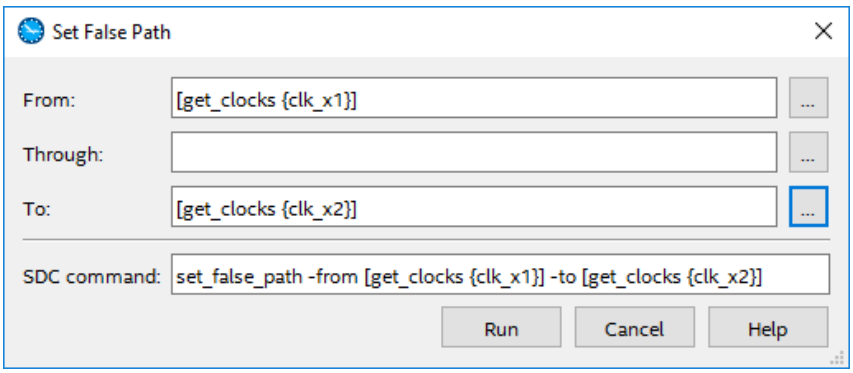
# `set_false_path` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| `[-from <names>]` | • Identifies source node or clock domain from which all paths will be cut |
| `[-rise_from <clocks>]` | • Indicates source clock domain from which paths launched by a rising edge will be cut |
| `[-fall_from <clocks>]` | • Indicates source clock domain from which paths launched by a falling edge will be cut |
| `[-through <names>]` | • Identifies intermediate path node through which paths will be cut |
| `[-to <names>]` | • Identifies target node or clock domain to which all paths will be cut |
| `[-rise_to <clocks>]` | • Indicates target clock domain to which paths captured by a rising edge will be cut |
| `[-fall_to <clocks>]` | • Indicates target clock domain to which paths captured by a falling edge will be cut |
| `[-latency_insensitive]` | • Indicates path is a false path from a timing perspective, but Fitter (retiming) path optimization may still be performed<br>• False path must be clock-based |




163

# Set False Path (GUI)



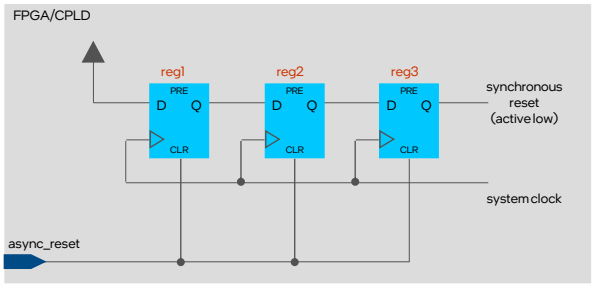




164

# False Path Example 1 : Asynchronous Reset Input to Device

- Synchronizing logic added to reset path to synchronize into internal clock domain
  - Asynchronous assertion
  - Synchronous de-assertion
- Reset input to register becomes false paths

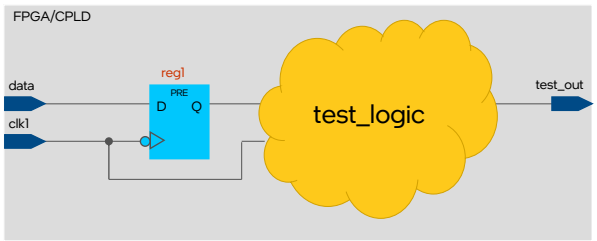



```
set_false_path -from [get_ports async_reset]
```




165

# False Path Example 2: Removing Test Logic



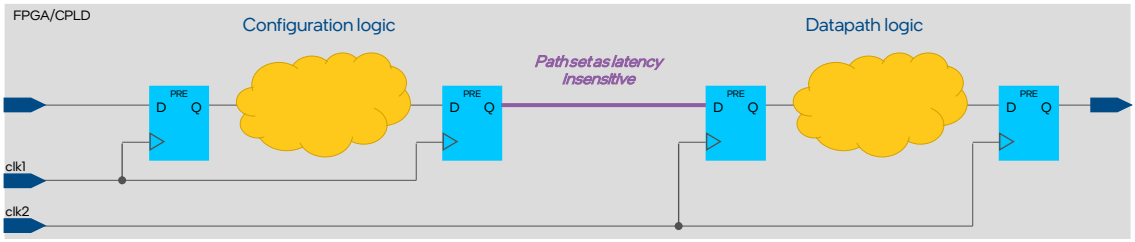

Cutting analysis of inserted test logic

```
set_false_path -from [get_clocks clk1] -to [get_registers test_logic|*]
set_false_path -from [get_registers test_logic|*] -to [get_registers test_logic|*]
set_false_path -from [get_registers test_logic|*] -to [get_ports test_out]
```




166

# False Path Example 3: Latency Insensitive False Path

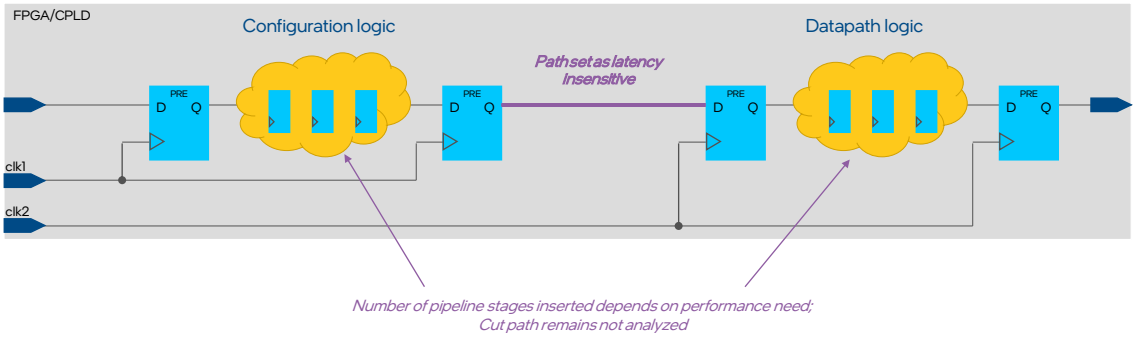



```
set_false_path -latency_insensitive -from [get_clocks clk1] -to [get_clocks clk2]
```




167

# False Path Example 4: Latency Insensitive False Path (cont.)




```
set_false_path -latency_insensitive -from [get_clocks clk1] -to [get_clocks clk2]
```




168

## False Paths vs. Clock Groups

- Both commands prevent analysis on paths

- `set_false_path` command
  - Provides more filter control over targeted paths
  - Unidirectional
- `set_clock_groups` command
  - Targets all paths between domains
  - Cuts both directions




169

## SDC Timing Exceptions Agenda

- False paths
- Multicycle paths
- Exception priorities
- Verifying exceptions




170

## Timing Exceptions: Multicycle Paths

- Paths where the default selected launch or latch edge causes an incorrect timing analysis
  - e.g. more than one cycle for data to propagate
- Causes timing analyzer to select another latch or launch edge
- Designer specifies number of cycles to move edge




171

## Remember…

- Expected design behavior should determine the constraints to use
  - i.e. logic must be designed to work this way!
  - Constraint simply informs timing analysis how logic is supposed to function
- Yes, multicycles make the red go away!
  - But resulting design could be non-functional




172

## Instances Where Multicycle Paths Used

- Design does not require single cycle to transfer data (non-critical paths)
  - Otherwise needlessly over-constrain paths
- Clocks are integer multiples of each other with or without offset
  - Demonstrated in Workshop Lab 4
- Clock enables ensuring register(s) not sampling data every clock edge




173

## `set_multicycle_path` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Defines multicycle paths and by how many cycles/edges their analysis need to be adjusted<br>• Allows Fitter to change path priority |
| Common Arguments | [-start \| -end]<br>[-setup \| -hold]<br>[-from <names>]<br>[-fall_from <clocks>]<br>[-rise_from <clocks>]<br>[-through <names>]<br>[-to <names>]<br>[-fall_to <clocks>]<br>[-rise_to <clocks>]<br><value> |




174

## `set_multicycle_path` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| `[-start \| -end]` | • Indicates whether source or destination clock edge is being shifted<br>• Default = `-end` |
| `[-setup \| -hold]` | • Indicates whether setup analysis or hold analysis is being adjusted<br>• Default = `-setup` |
| `<value>` | • Specifies the number of edges to move the launching or capturing edge |




175

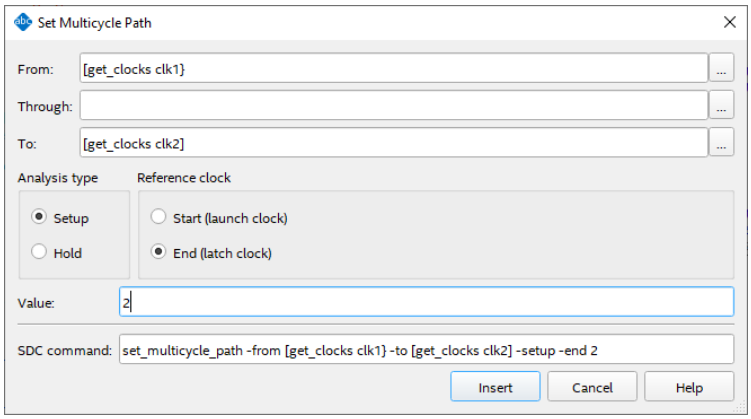
## `set_multicycle_path` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| `[-from <names>]` | • Identifies source node or clock domain from which all paths will be cut |
| `[-rise_from <clocks>]` | • Indicates source clock domain from which paths launched by a rising edge will be cut |
| `[-fall_from <clocks>]` | • Indicates source clock domain from which paths launched by a falling edge will be cut |
| `[-through <names>]` | • Identifies intermediate path node through which paths will be cut |
| `[-to <names>]` | • Identifies target node or clock domain to which all paths will be cut |
| `[-rise_to <clocks>]` | • Indicates target clock domain to which paths captured by a rising edge will be cut |
| `[-fall_to <clocks>]` | • Indicates target clock domain to which paths captured by a falling edge will be cut |




176

# Set Multicycle Path (GUI)







177

# Common Multicycle Constraint Cases

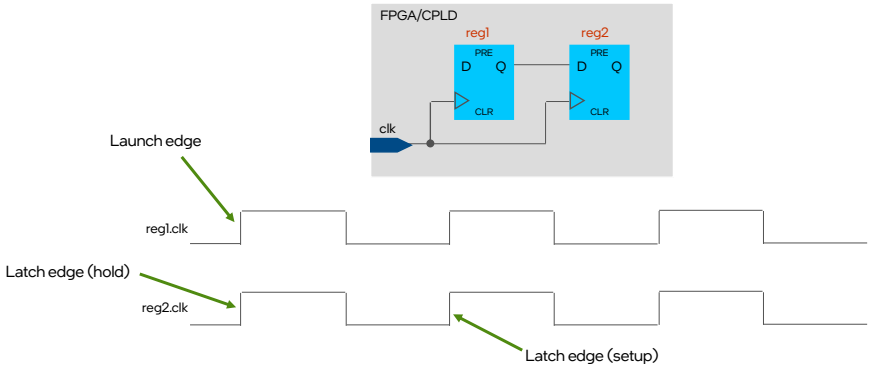
- Opening the capture window
- Shifting the capture window




178

# Understanding Multicycle

- Standard single-cycle register transfer



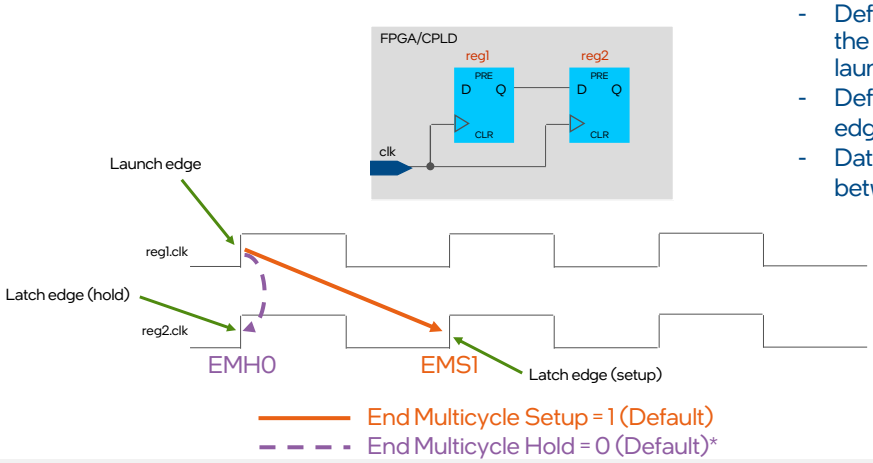




179

# Understanding Multicycle

- Standard single-cycle register transfer

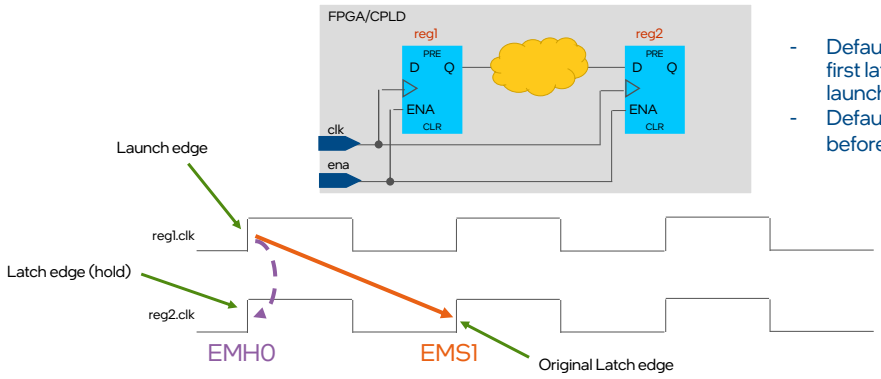



- Default setup edge of 1 is the first latch edge after the launch
- Default hold of 0 is the edge before the setup
- Data must arrive to reg2 between EMH0 and EMS1




180

# Case 1: Opening the Window (Default)



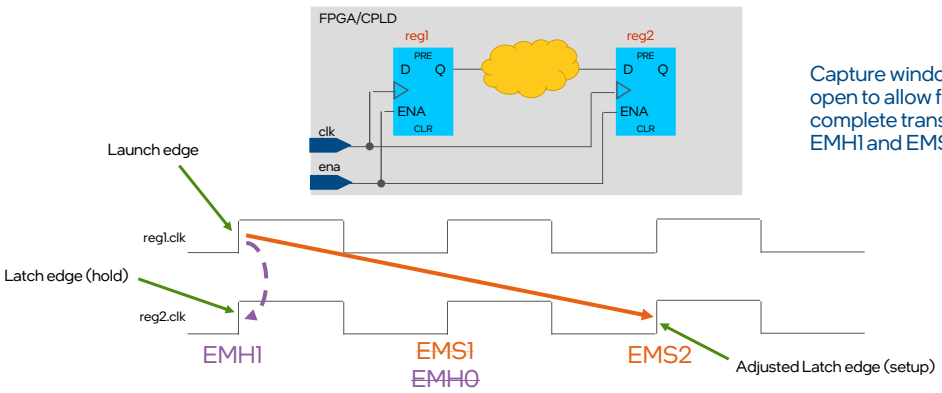

- Default setup edge of 1 is the first latch edge after the launch
- Default hold of 0 is the edge before the setup





181

# Case 1: Opening the Window (2 Cycle Transfer)




Capture window is now open to allow for 2 cycles to complete transfer (between EMH1 and EMS2)

```
set_multicycle_path -from [get_keepers reg1] -to [get_keepers reg2] -setup 2
set_multicycle_path -from [get_keepers reg1] -to [get_keepers reg2] -hold 1
```

End Multicycle Setup = 2

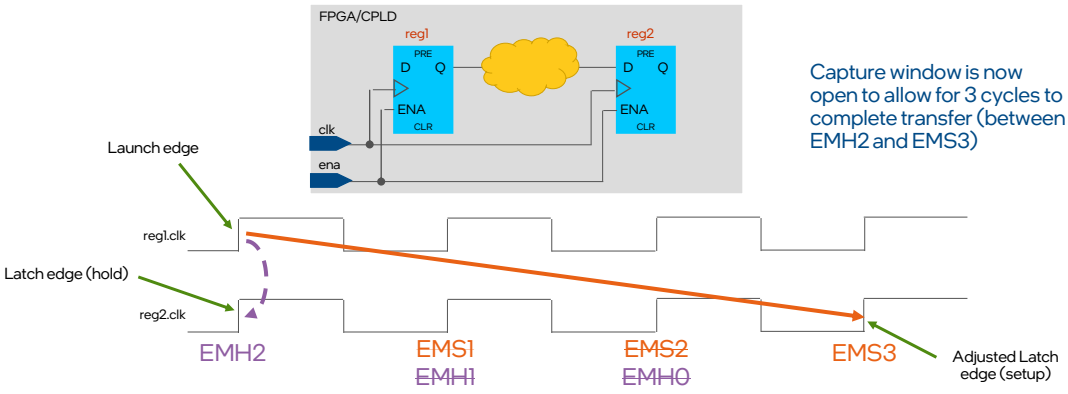
End Multicycle Hold = 1





182

# Case 1: Opening the Window (3 Cycle Transfer)




```
set_multicycle_path -from [get_keepers reg1] -to [get_keepers reg2] -setup 3
set_multicycle_path -from [get_keepers reg1] -to [get_keepers reg2] -hold 2
```

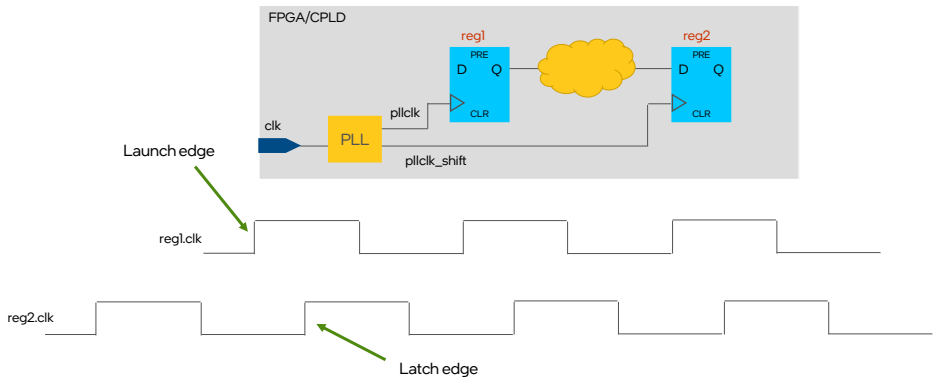
End Multicycle Setup = 3
End Multicycle Hold = 2




183

# Case 2: Shifting the Window

- PLL inserts small positive phase shift
- Transfer is still single cycle, but capture cycle is delayed

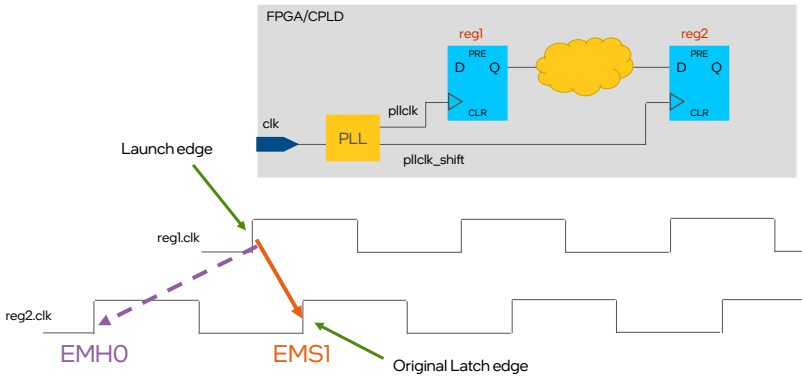






184

# Case 2: Shifting the Window (Default)




- Default setup edge of 1 is the first latch edge after the launch
- Default hold of 0 is the edge before the setup

Default setup edge timing cannot be met

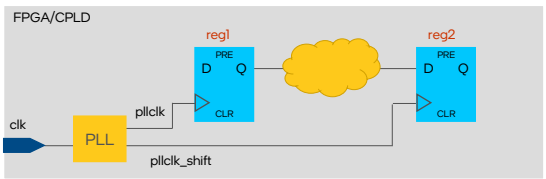


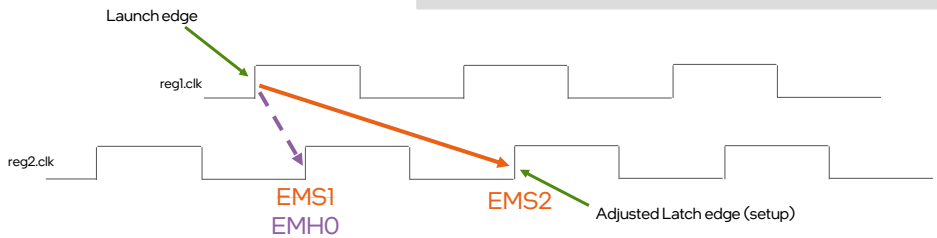


185

# Case 2: Shifting the Window (Adjusted)

- Setup multicycle gives data more time to arrive
- Hold constraint ensures data does not arrive too early (within a single cycle of EMH0 and EMS2)




```
set_multicycle_path -from [get_clocks pllclk] -to [get_clocks pllclk_shift] -setup 2
```

End Multicycle Setup = 2

End Multicycle Hold = 0 (Default)

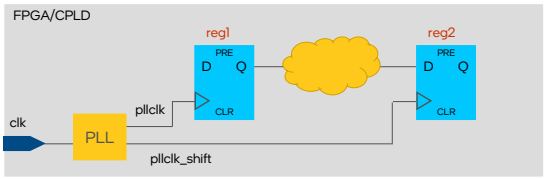


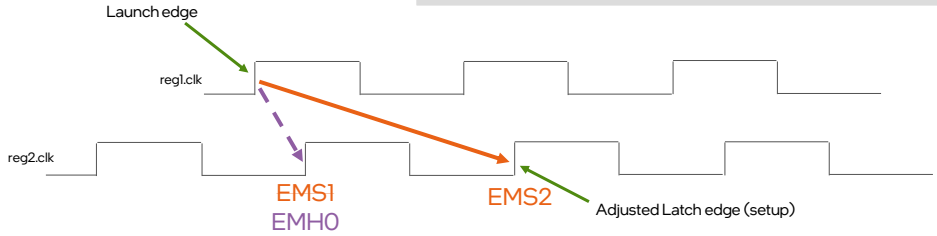


186

# Case 2: Shifting the Window (Adjusted)

- Setup multicycle gives data more time to arrive
- Hold constraint ensures data does not arrive too early (within a single cycle of EMH0 and EMS2)




```
set_multicycle_path -from [get_clocks pllclk] -to [get_clocks pllclk_shift] -setup 2
set_multicycle_path -from [get_clocks pllclk] -to [get_clocks pllclk_shift] -hold 0
```

Hold constraint not required as it is default, but may be helpful for readability




187

# Other Multicycle Cases

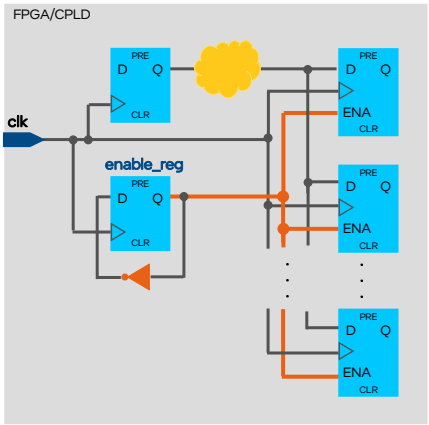
- These two cases cover most multicycle situations
- More complex multicycle timing relationships may still occur
- If needed, refer to Intel® Quartus® Prime Pro Edition User Guide: Timing Analyzer for further examples




188

# Special Case: Clock Enable with Multicycle

- Data paths with destination register controlled by enable
- Need multicycle exceptions
- May be difficult and tedious to do with standard SDC collections and timing netlist hierarchy
  - If naming convention to identify enable controlled registers not used
- Highlights how to use special `get_fanouts` collection







189

# `get_fanouts` Command

| Constraint Property | Constraint Property Details |
|---|---|
| Description | • Finds all registers to which a signal fans out |
| Example Arguments | <filter><br>[-through]<br>[-no_logic]<br>[-non_inverting_paths] |




190

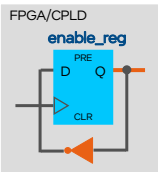
# `get_fanouts` Command Argument Details

| Argument | Argument Definition and Details |
|---|---|
| `<filter>` | • Similar to `-from`; the source or starting point(s) from which to return the fanout |
| `[-through]` | • Indicates to only return fanouts in which the path to them goes through specified intermediate cell |
| `[-no_logic]` | • Specifies not to follow combinational paths |
| `[-non_inverting_paths]` | • Specifies not to follow combinational paths with inversion |




191

# Filtering On Clock Enabled Registers




```
# Create Tcl variable "enabled_regs"
#   Remove enable_reg itself from target collection
set enabled_regs [get_fanouts enable_reg]
set enabled_regs [remove_from_collection $enabled_regs enable_reg]
```

- Returns fanout nodes for all logic that is connected to the output of register `enable_reg` (source of enable signal)
  - Make sure to use clock enable source only as clock enable
  - Mixing with other uses may synthesis to merge signal with other signals
  - Can use `dont_merge` synthesis attribute to guarantee this
- Removes from collection `enable_reg` itself

*Note: there is no `-to` option for `get_fanouts`*




192

## Possible Multicycles for the Enabled Registers

- With registers filtered, user can decide how to apply multicycle based on design behavior

| | |
|---|---|
| Paths between enable-controlled registers | `set_multicycle_path -setup 2 -from $enabled_regs \ -to $enabled_regs` |
| Paths to enable-controlled registers | `set_multicycle_path -setup 2 -to $enabled_regs` |
| Paths from enable-controlled registers | `set_multicycle_path -setup 2 -from $enabled_regs` |
| Paths to and from enable-controlled registers | `set_multicycle_path -setup 2 -to $enabled_regs`<br>`set_multicycle_path -setup 2 -from $enabled_regs` |




193

## SDC Timing Exceptions Agenda

- False paths
- Multicycle paths
- Exception priorities
- Verifying exceptions




194

## Exception Priorities

- Same path, different constraints
  - What happens if both a multicycle and a max (or min) path delay exception are applied to the same path?
- Same constraint, different values
  - What happens if a multicycle setup of 2 is applied to a path by one constraint, but another constraint sets a multicycle setup of 3 to the same path?
- Use Report Exceptions timing report to verify




195

## Same Path, Different Constraint

- Constraint priority determines results

1. (High) `set_false_path/set_clock_groups`
   - Once a path is cut, there's no way to "un-cut" it, so other constraints don't apply
2. `set_max_delay/set_min_delay`
   - Arbitrary time delay overrides multicycles, which are fixed to clock edges
3. `set_multicycle_path`
   - Overrides default setup/hold relationships
4. (Low) `create_clock/create_generated_clock`
   - Creates the default setup/hold relationships




196

# What about `set_input/output_delay`?

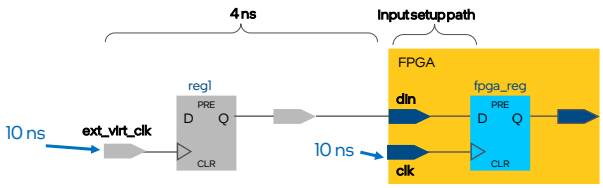
- Not exceptions
- Describes external circuit delay, and applied in addition to all other constraints





197

# Constraint Priority Example (1)



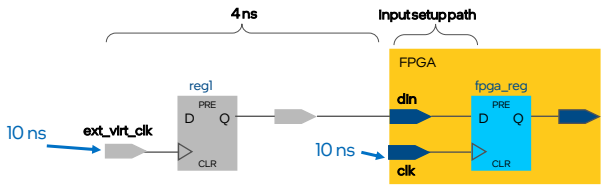

```
# Example uses setup (-max) analysis only; normally
# hold (-min) would be added as well
# Establish default setup/hold relationship; 6 ns (10 - 4) for input setup path
create_clock -period 10.0 -name ext_virt_clk
create_clock -period 10.0 -name clk [get_ports clk]
set_input_delay -clock ext_virt_clk -max 4.0 [get_ports din]
```





198

# Constraint Priority Example (2)

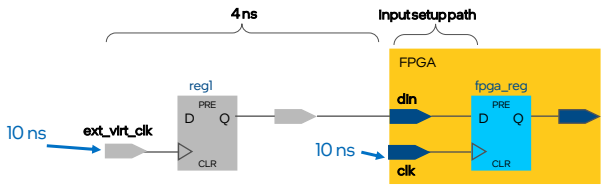



```
# Example uses setup (-max) analysis only; normally
# hold (-min) would be added as well
# Establish default setup/hold relationship; 6 ns (10 - 4) for input setup path
create_clock -period 10.0 -name ext_virt_clk
create_clock -period 10.0 -name clk [get_ports clk]
set_input_delay -clock ext_virt_clk -max 4.0 [get_ports din]

# Higher priority; now 16ns ((2 * 10) - 4) for input setup path
set_multicycle_path -setup 2 -from [get_ports din]
```




199

# Constraint Priority Example (3)

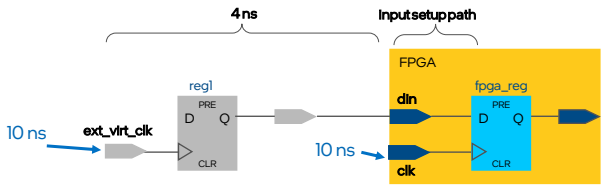



```
# Example uses setup (-max) analysis only; normally
# hold (-min) would be added as well
# Establish default setup/hold relationship; 6 ns (10 - 4) for input setup path
create_clock -period 10.0 -name ext_virt_clk
create_clock -period 10.0 -name clk [get_ports clk]
set_input_delay -clock ext_virt_clk -max 4.0 [get_ports din]

# Higher priority; now 16ns ((2 * 10) - 4) for input setup path
set_multicycle_path -setup 2 -from [get_ports din]

# Still higher priority; now 26ns (30 - 4) for input setup path
set_max_delay-from [get_ports din] 30.0
```




200

# Constraint Priority Example (4)




```
# Example uses setup (-max) analysis only; normally
# hold (-min) would be added as well
# Establish default setup/hold relationship; 6 ns (10 - 4) for input setup path
create_clock -period 10.0 -name ext_virt_clk
create_clock -period 10.0 -name clk [get_ports clk]
set_input_delay -clock ext_virt_clk -max 4.0 [get_ports din]

# Higher priority; now 16ns ((2 * 10) - 4) for input setup path
set_multicycle_path -setup 2 -from [get_ports din]

# Still higher priority; now 26ns (30 - 4) for input setup path
set_max_delay-from [get_ports din] 30.0

# Highest priority; path is now cut and not analyzed at all
set_false_path -from [get_ports din]
```



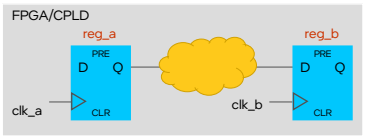




# Same Direct Constraint, Different Values

- Constraints applied in the order they are read
- Last constraint that targets path overrides all previous equivalent constraints which affect that path

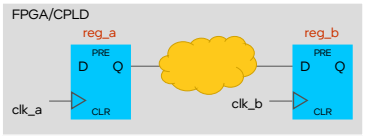
```
# This multicycle constraint directly targets a particular path
#  from reg_a to reg_b
set_multicycle_path -setup 4 -from [get_registers top|mod_a|reg_a] \
     -to [get_registers top|mod_b|reg_b]

# This multicycle also targets the same path directly
#   but uses a different value
# Since this constraint is read last, it overrides the first
set_multicycle_path -setup 2 -from [get_registers top|mod_a|reg_a] \
     -to [get_registers top|mod_b|reg_b]
```



## Same Direct Constraint, Different Values

- Constraints applied in the order they are read
- Last constraint that targets path overrides all previous equivalent constraints which affect that path

```
# This multicycle constraint directly targets a particular path
#  from reg_a to reg_b
set_multicycle_path -setup 4 -from [get_registers top|mod_a|reg_a] \
     -to [get_registers top|mod_b|reg_b]

# This multicycle also targets the same path directly
#   but uses a different value
# Since this constraint is read last, it overrides the first
set_multicycle_path -setup 2 -from [get_registers top|mod_a|reg_a] \
     -to [get_registers top|mod_b|reg_b]
```





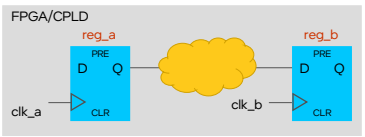


203

## Direct vs Indirect Constraint, Different Values

- Assignments to individual nodes have priority over assignments to clocks

```
# This multicycle constraint directly targets a particular path
#  from reg_a to reg_b
set_multicycle_path -setup 4 -from [get_registers top|mod_a|reg_a] \
     -to [get_registers top|mod_b|reg_b]

# This multicycle constraint indirectly targets the same path by
#  targeting the clocks that drive the path (clk_a -> reg_a,
#  clk_b -> reg_b)
# Since this constraint is targeting the clocks, it is not applied
set_multicycle_path -setup 2 -from [get_clocks clk_a] \
     -to [get_clocks clk_b]
```





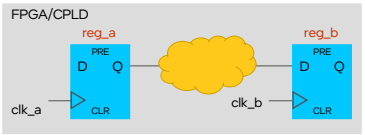


204

## Direct vs Indirect Constraint, Different Values

- Assignments to individual nodes have priority over assignments to clocks

```
# This multicycle constraint directly targets a particular path
#  from reg_a to reg_b
set_multicycle_path -setup 4 -from [get_registers top|mod_a|reg_a] \
     -to [get_registers top|mod_b|reg_b]

# This multicycle constraint indirectly targets the same path by
#  targeting the clocks that drive the path (clk_a -> reg_a,
#  clk_b -> reg_b)
# Since this constraint is targeting the clocks, it is not applied
set_multicycle_path -setup 2 -from [get_clocks clk_a] \
     -to [get_clocks clk_b]
```









## Same Path, Same Timing Exception

- Argument use may also determine priority as determined by this list
- An asterisk wildcard (*) for any of these options applies the same precedence as not specifying the option at all.
  - For example, `-from a -to *` is treated identically to `-from a` with regards to precedence

1. (high) `-from <node>`
2. `-to <node>`
3. `-thru <node>`
4. `-from <clock>`
5. (low) `-to <clock>`

## Constraint Priority Takeaway

- Be aware that exceptions interact in different ways
- Use caution when writing exceptions that can potentially overwrite existing ones
- Use reports (discussed next) to confirm exceptions were applied as intended




207

## SDC Timing Exceptions Agenda

- False paths
- Multicycle paths
- Exception priorities
- Verifying exceptions




208

## Verifying Timing Exceptions

- Use these reports to ensure timing exceptions are being applied correctly
- Report SDC (again)
  - Make sure exceptions constraints read correctly
- Report Exceptions




209

## Report Exceptions

- Provides detail on exceptions applied to netlist
- Analogous to executing Report Timing report on all paths affected by an exception
- Use
  - Confirm exception applied to correct paths (and only those)
  - Confirm correct exception value applied
  - Confirm exception not overridden by higher priority




210

# Report Exceptions - Summary

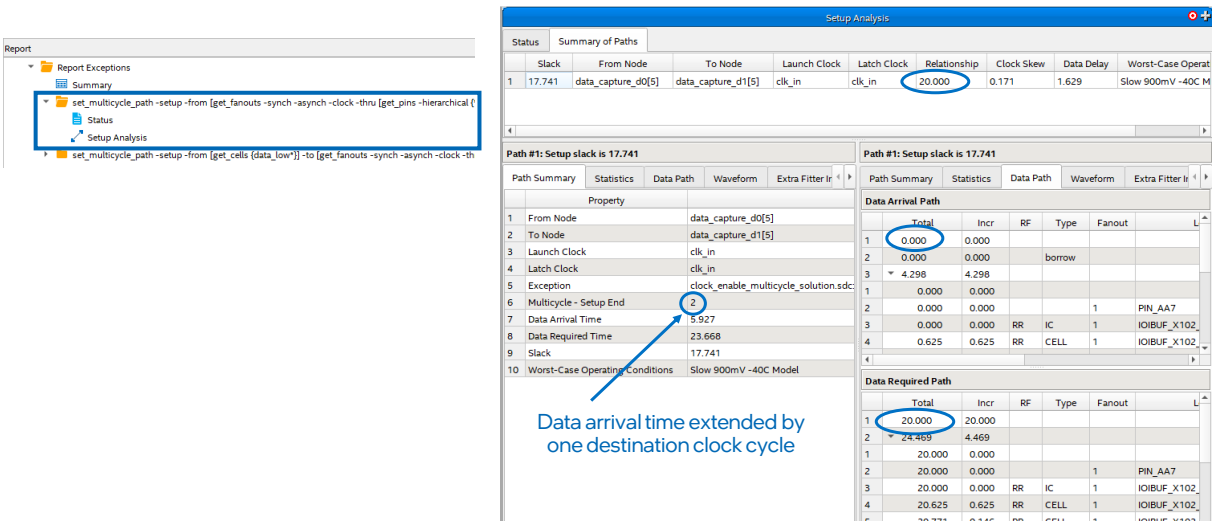
- Returns status on all applied exceptions
  - **Complete**: Valid and not overridden by higher-precedence or subsequent exceptions
  - **Partially Overridden**: Some paths covered by exception overridden by higher-precedence or subsequent exceptions
  - **Fully Overridden**: All paths in report overridden
  - **Invalid**: Valid collections (-from, -to, etc.), but no actual matching paths exist

| | Status | Exception Command | Flags | From Flag | From | Through | To Flag | To | Value | Setup Slack | Hold Slack | Recovery Slack | Removal Slack |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Paths will not be analyzed | set_false_path | | | | | -to | [get_keeper...:*\|din_s1]] | | Invalid | n/a | n/a | n/a |
| 2 | Complete | set_false_path | | | | [get_...ata]] | -to | [get_keeper...st\|din_s1]] | | -1.985 | n/a | n/a | n/a |
| 3 | Complete | set_false_path | | | | [get_...ata]] | -to | [get_keeper...st\|din_s1]] | | -2.100 | n/a | n/a | n/a |
| 4 | Invalid | set_false_path | | -from | [get_registers {...k_data_buffer*}] | | -to | [get_regist...src_data*]] | | Invalid | n/a | n/a | n/a |
| 5 | Partially overridden | set_multicycle_path | -setup -end | -from | [get_fanouts -sy...al {enable\|q*}]] | | -to | [get_fanout...nable\|q*]]] | 2 | 17.741 | n/a | n/a | n/a |
| 6 | Complete | set_multicycle_path | -setup -end | -from | [get_cells {data_low*}] | | -to | [get_fanout...nable\|q*]]] | 1 | 8.939 | n/a | n/a | n/a |




211

# Report Exceptions: Multicycle Path Summary





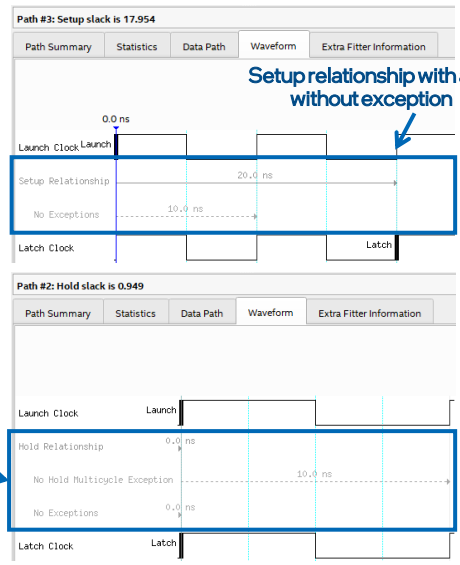


212

# Report Exceptions: Multicycle Waveform Tab

- No Exceptions
  - Launch/latch relationship if no MC exceptions applied
- No Hold Multicycle Exception
  - Launch/latch relationship for hold if setup MC applied without hold MC

Period = 10 ns
EMS = 2; EMH = 1




Hold relationship with and without exceptions




213

# Intel® Quartus® Prime Pro Timing Analysis

## Summary




214

# Intel® Quartus® Prime Pro Software Timing Analysis – Summary

## In this course, we...

- Learned the flow for setting up and performing timing analysis in the Intel® Quartus® Prime Pro software Timing Analyzer
- Gained understanding of SDC terminology and syntax
- Used the latest recommendations to write clear and effective SDC files for properly constraining and analyzing Intel FPGA designs for timing




215

# Additional Training and Support Resources

Visit the Intel® FPGA YouTube channel for more than 1000 trainings and quick videos

Visit the Intel FPGA training website for eLearning courses narrated in a slide-by-slide player

Enroll in free instructor-led courses that include interaction and hands-on lab exercises

Still have questions? Visit our forums that are monitored by skilled applications engineers




216

## We Welcome Your Feedback

- Intel® FPGA Technical Training Team is continuously looking for ways to improve our training
- Your feedback is an important part of this process
- Please email fpgatraining@intel.com with any thoughts you have on the course you attended




217

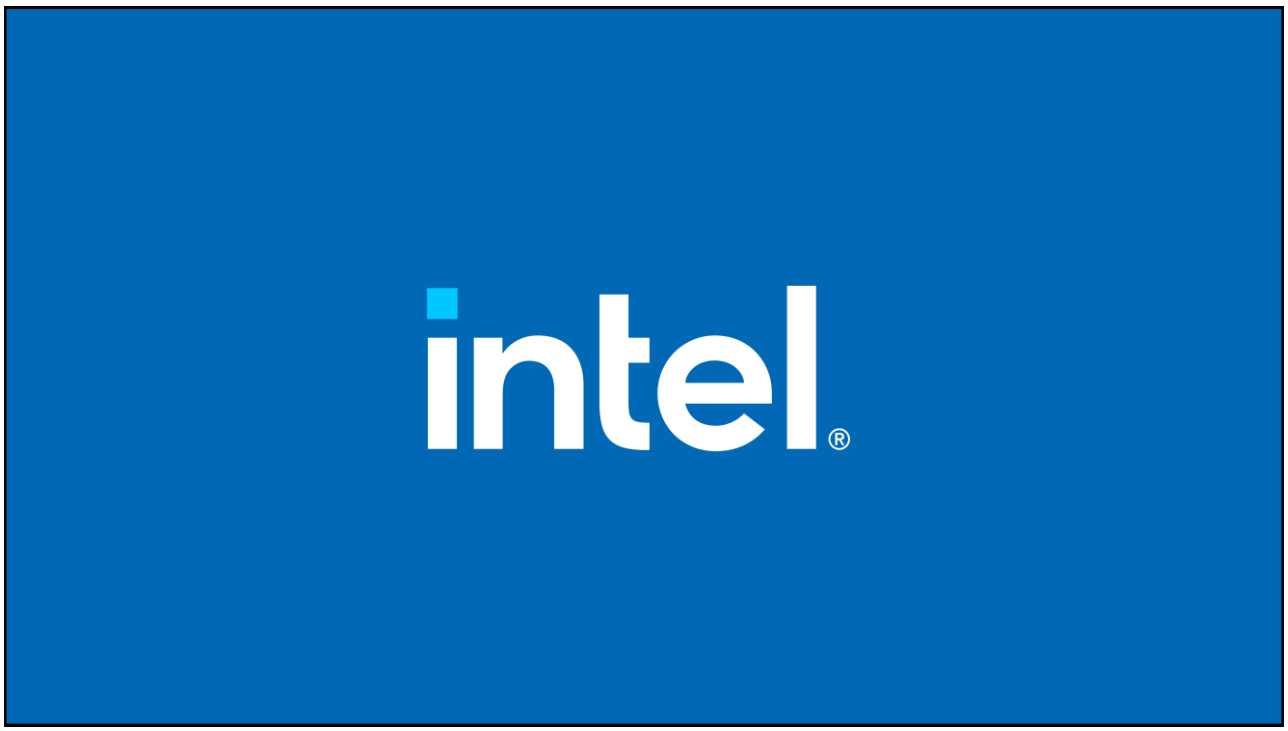
## Legal Disclaimers/Acknowledgements

- Intel technologies may require enabled hardware, software or service activation
- No product or component can be absolutely secure
- Your costs and results may vary
- Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.
- *Other names and brands may be claimed as the property of others
- OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos




218

219




220