Interactive Timing Analysis Using TCL in Lattice Radiant Design Software

Application Note

FPGA-AN-02091-1.0

July 2024

Disclaimers

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
TCL	Tool Command Language
SDC	Synopsys Design Constraints
STA	Static Timing Analysis

1. Introduction

The Lattice Radiant™ software is the complete design environment for Lattice Semiconductor FPGAs. The software includes a comprehensive set of tools for all design tasks, including project management, design entry, simulation, synthesis, place and route, in-system logic analysis, and more.

The Radiant software is equipped with the Tool Command Language (TCL) for scripting. TCL is a key component of the Synopsys® Design Constraints (SDC) format, which provides a versatile language in communicating timing constraints to FPGA synthesis tools from various vendors, including Synopsys Synplify. This widespread use of TCL for FPGA designs has established it as a "Best Practice" in the scripting language domain.

The standout feature in TCL is its ability to conduct interactive queries with design tools and execute automated scripts. This dual functionality makes it a comprehensive platform for managing design databases, including tool settings and design states. For instance, Radiant TCL allows real-time querying of certain timing analysis report commands, applying incremental constraints, and conducting immediate post-application queries. This ensures that the expected behavior is verified without the need to re-run any tool steps. This interactive capability makes TCL an invaluable tool in managing and navigating design databases.

This document covers the interactive static timing analysis (STA) using the TCL commands in the Lattice Radiant software.

1.1. Audience

The intended audience for this document includes FPGA design engineers who are using the Lattice Radiant design software. The technical guidelines assumes that the readers have some basic knowledge of TCL and the SDC constraints usage.

2. Launching the Radiant Software in TCL Mode

The Radiant software includes TCL as its integrated scripting language, which enables you to launch the Radiant software in TCL mode without opening the Radiant software graphical user interface (GUI). This feature is useful for scripting the compilation flow.

You can launch the Radiant software in TCL mode using one of the following methods:

- Using the radiantc application (also known as pnmainc.exe), which is located in the Radiant software installation directory <radiant>/bin/nt64/radiant.
- Using the integrated Radiant TCL console, which can be accessed via the **Windows start menu > Radiant > TCL Console**

Figure 2.1. Integrated TCL Console in the Radiant Software

- Using the Windows command line to call the pnmainc.exe application.

  For instance, using a batch file to call the radiantc application to run the TCL scripts. This approach allows you to automate and run the tasks without opening the Radiant software GUI. Make sure to edit the directories in the batch file to match the directories on the workstation.

  For example, C:/lscc/radiant/<version>/bin/nt64/radiantc -source C/projects/script.tcl

  If you intend to open the GUI during the non-project flow, use the -gui option along with radiantc.

  For example, C:/lscc/radiant/<version>/bin/nt64/radiantc -gui -source C/projects/script.tcl

The TCL console in the Radiant software offers two distinct design flows:

- Project flow
  The project flow allows you to perform the following functions:
  - Create a new project or load a previously saved project from an existing project file.
  - Add, delete, or list project files, which include RTL, constraint, and strategy setting files.
  - Save the ongoing project into a project (.rdf) file at any point during the flow.

- Non-project flow
  The non-project flow can be initiated from an existing project flow design commands. The flow begins by reading a Radiant software database (.udb) file as input, after which the design implementation process can continue by running MAP, or place & route depending on the loaded .udb file.

The following table shows the key differences between project flow and non-project flow.

Table 2.1. Project vs Non-project flow differences

Feature	Project Flow	Non-Project Flow
Design Storage	Stores design in disk files. Allows different strategies to implement the same project.	Keeps design in memory. Enabling interactive access and changes using command. For example, using timing analysis commands to report timing results at different paths.
File Management	Requires adding project files and checks timestamps for changes. Prompts for rerunning specific stages based on the changes.	Starts with a Radiant software database (.udb) file as input. The timestamp of this file is not checked during the design flow.
Design Stages	prj_run commands (prj_run_bitstream, prj_run_map, prj_run_par, prj_run_synthesis) can be used to automatically run the design flow from the current stage to the target stage.	Requires running all stages sequentially. For example, placement (plc_run) must run before routing (rte_run).
Implementation Strategies and Options	Uses implementation strategies, which are a subset of all the available options.	Utilizes the full range of implementation options.

Note:

The TCL console inside the Radiant software GUI cannot be used to execute non-project mode commands. This document describes how to use various commands of interactive STA in the non-project mode.

3. TCL Commands for Timing Analysis

The non-project mode TCL in the Radiant software provides a set of *sta* commands to perform timing analysis in an interactive manner. These commands are primarily divided into the following categories:

- sta_get command: The sta_get command is designed to return a value based on the specified parameters. This returned value can be utilized in other *sta* commands to perform further operations or analysis. Additionally, it can also be printed out to the standard output for immediate viewing or logging purposes.
  The following table lists the sta_get commands that are supported in the Radiant software.

Table 3.1. sta_get Command Description

Command	Description
sta_get_clocks	To get a list of clocks that matches the specified name.
sta_get_coverage	To get the timing constraints coverage.
sta_get_info	To get information about a clock, port, or a pin.
sta_get_paths	To get the details of the paths.
sta_get_slack	To get the total negative slack, or terminal worst slack information.
sta_get_terms	To get a list of ports and pins that matches the specified names.

- sta_report command: The sta_report command is used to send the information directly to the standard output. This allows you to instantly view the results of your timing analysis, making it a valuable tool for real-time monitoring and adjustments.

  The following table lists the sta_report commands that are supported in the Radiant software.

Table 3.2. sta_report Command Description

Command	Description
sta_report_clocks	To report the clocks.
sta_report_constraints	To report timing constraints.
sta_report_timing	To report detailed timing or timing summary.
sta_report_unconstrained	To report unconstrained ports. Pins or nets.

- sta_* command: Use the help {sta_*} command to access a list of all the sta commands that are available in the Radiant software along with the descriptions and usage instructions. This feature ensures that you can easily find and understand the commands you need for your specific timing analysis tasks.
  The following table lists the other *sta* commands that are supported in the Radiant software.

Table 3.3. sta_* Command Description

Command	Description
sta_path_info	To extract data from a given path.
sta_set_option	To change the style of the timing report.

3.1. sta_set_option Command

The sta_set_option command is used to change the style of the timing report. This allows you to customize the presentation of the timing report based on your preferences.

Syntax:

```
sta_set_option [-help] [-name <logical/physical>] [-worst_delays]
```

The following table lists the flags that you can use with the sta_set_option command.

Table 3.4. sta_set_option Usage Description

Flag	Description
-help	Provides help information related to the sta_set_option command.
-name	A string parameter that represents the type of names to use in reports. It can be *logical* or *physical*. When specified, the command uses logical or physical names in reports.
-worst_delays	A boolean flag. When specified, the command enables worst case delay if true.

3.1.1. Use Cases for the sta_set_option Command

The following lists the use case examples for this command:

- Changing Name Style: To change the name style in the timing report, use the -name flag.
  For example, the following command will use logical names in the report:
  ```
  sta_set_option -name logical
  ```
  Or, the following command will use physical names in the report:
  ```
  sta_set_option -name physical
  ```

- Enabling Worst Case Delay To enable worst case delay in the timing report, use the -worst_delays flag.
  For example, this command enables worst case delay:
  ```
  sta_set_option -worst_delays
  ```

3.2. sta_get_clocks Command

The sta_get_clocks command is used to retrieve a list of clocks that matches a specified name.

Syntax:

The syntax information for the sta_get_clocks command is as follows:

```
sta_get_clocks [-help] [name]
```

The following table lists the flags that you can use with the sta_get_clocks command.

Table 3.5. sta_get_clocks Usage Description

Flag	Description
-help	Provides help information related to the sta_get_clocks command.
name	This is a string parameter that represents the name of the clocks. When specified, the command returns a list of clocks that matches this name.

3.2.1. Use Case: Retrieve Clocks with Specific Names or Patterns

Suppose there are multiple clocks in the design and you want to retrieve all clocks that have a specific name or pattern in their name. You can use the sta_get_clocks command to do this.

The following lists the use case examples for this command:

- Example 1: In a design with multiple clocks, use the sta_get_clocks command to retrieve all clocks containing a specific name. For example, this command example gets all clocks with 'clk' in their name:
  ```
  sta_get_clocks clk
  ```
  This command returns a list of all clocks in the design that have 'clk' in their name. This can be particularly useful when you perform operations or analysis on a specific set of clocks in your design.

- Example 2: To retrieve all the clocks that start with a specific suffix clk_in, use the following command:
  ```
  sta_get_clocks clk_in*
  ```

- Example 3: To use the sta_get_clocks in a Tcl script, refer to the following example:
  ```
  set clocks [ sta_get_clocks <clock name pattern> ]

  foreach clock $clocks
  {
      puts "Clock name: $clock"
  }
  ```

In this script example, the sta_get_clocks command is used to retrieve a list of clocks that match the specified name pattern. The foreach loop then iterates over each clock in the list and prints out its name.

You can replace <clock name pattern> with the specific name or pattern of the clocks you want to retrieve. For example, to retrieve all clocks that start with 'clk', you can use clk* as the name pattern.

3.3. sta_get_coverage Command

The sta_get_coverage command is used to retrieve the timing constraints coverage for connections in your design.

Syntax:

The syntax information for the sta_get_coverage command is as follows:

```
sta_get_coverage [-help]
```

The following table lists the flag that you can use with the sta_get_coverage command.

Table 3.6. sta_get_coverage Usage Description

Flag	Description
-help	Provides help information related to the sta_get_coverage command.

3.3.1. Use Case: sta_get_coverage Command

Suppose you have a complex design with numerous timing constraints and connections. You want to ensure that all connections in your design are covered by your timing constraints. The sta_get_coverage command can be used to retrieve the timing constraints coverage for all connections in your design. This can help you identify any connections that are not covered by your timing constraints, allowing you to adjust your constraints accordingly.

For example, you can use the following command to get the timing constraints coverage for your design:

```
sta_get_coverage
```

This command returns the timing constraints coverage for all connections in your design. This can be particularly useful when you want to verify the completeness of your timing constraints and ensure that all connections in your design are properly constrained.

3.4. sta_get_info Command

The sta_get_info command is used to retrieve information about a clock or a terminal (port/pin).

Syntax: The syntax information for the sta_get_info command is as follows:

```
sta_get_info [-help] [-clock <clock_name>] [-waveform] [-period] [-duty_cycle] [-terminal
            <terminal_name>] [-isclockpin] [-isinput] [-isoutput] [-net] [-cell]
```

The following table lists the flags that you can use with the sta_get_info command.

Table 3.7. sta_get_info Usage Description

Flag	Description
-help	Provides help information related to the sta_get_info command.
-clock	This is a string parameter that represents the name of the clock. When specified, the command returns information about this clock.
-waveform	This command returns the waveform of the specified clock.
-period	This command returns the period of the specified clock.
-duty_cycle	This command returns the duty cycle of the specified clock.
-terminal	This is a string parameter that represents the name of the terminal. When specified, this command returns information about this terminal.
-isclockpin	This command returns 1 if the specified terminal is a clock pin else returns 0.
-isinput	This command returns 1 if the specified terminal is an input pin else returns 0.
-isoutput	This command returns 1 if the specified terminal is an output pin else returns 0.
-net	This is a string parameter that represents the name of the net. When specified, this command returns information about this terminal.
-cell	This is a string parameter that represents the name of the cell. When specified, this command returns information about this terminal.

Note:
Currently only one option is supported at any given time.

3.4.1. Use Cases for the sta_get_info Command

Suppose there are multiple clocks and terminals in the design and you want to retrieve specific information about them. You can use the sta_get_info command to do this.

The following lists the use case examples for this command:

- Example 1: To get the waveform and period of a clock named 'clk', use the following command:

  ```
  sta_get_info -clock clk -waveform
  ```

  This command will return the waveform the clock named 'clk'. This can be particularly useful when you want to analyze the behavior of a specific clock in your design.

- Example 2: To check if a terminal named 'term1' is an input pin, use the following command:

  ```
  sta_get_info -terminal term1 -isinput
  ```

  This command returns 1 if the terminal named 'term1' is an input pin. This can help you understand the role of specific terminals in your design.

3.4.2. Use Case: To Create Constraints

You can use Radiant TCL commands to create timing constraints using the sta_get_info command. Here's a sample script to generate the create_clock constraint for clocks, set_input_delay constraint for input ports, and set_output_delay for output ports.

```
 # Open the .sdc file for writing
set sdc_file [open "my_design.sdc" w]

# Create a dummy virtual clock
puts $sdc_file "create_clock -period 10 -name virt_clk"

# Get all ports
set all_ports [all_inputs]
set all_outputs [all_outputs]
foreach port1 $all_outputs {
       lappend all_ports $port1

}

foreach port $all_ports {
    # Query the port for clock pin
    set isclockpin [sta_get_info -terminal $port -isclockpin]
    # Query the port for input
    set isinput [sta_get_info -terminal $port -isinput]
    # Query the port for output
    set isoutput [sta_get_info -terminal $port -isoutput]

    # Check if it's a clock pin
    if {$isclockpin} {
        # Write out a create_clock command
        puts $sdc_file "create_clock -period 10 \[get_ports $port\]"
    }
       puts $isclockpin
    if {$isinput} {
          # Add a set_input_delay constraint
```

```
            puts $sdc_file "set_input_delay -clock virt_clk 1 \[get_ports $port\]"
    }
    if {$isoutput} {
            # Add a set_output_delay constraint
            puts $sdc_file "set_output_delay -clock virt_clk 1 \[get_ports $port\]"
    }
}

# Close the .sdc file
close $sdc_file
```

3.5. sta_get_paths Command

The sta_get_paths command is used to collect detailed paths for a query.

Syntax:

```
sta_get_info sta_get_paths [-help] [-n <number_of_paths>] [-from <from_name>] [-from_clock
             <from_clock_name>] [-to <to_name>] [-to_clock <to_clock_name>] [-hold]
```

The following table lists the flags that you can use with the sta_get_path command.

Table 3.8. sta_get_paths Usage Description

Flag	Description
-help	Provides help information related to the sta_get_paths command.
-n	This is an integer parameter that represents the number of paths to report.
-from	This is a string parameter that represents the name/s of ports and/or pins. When specified, the command returns paths from these ports/pins.
-from_clock	This is a string parameter that represents the name/s of clocks. When specified, the command returns paths from these clocks.
-to	This is a string parameter that represents the name/s of ports and/or pins. When specified, the command returns paths to these ports/pins.
-to_clock	This is a string parameter that represents the name/s of clocks. When specified, the command returns paths to these clocks.
-hold	To report hold paths. If not specified, the command defaults to reporting setup paths.

3.5.1. Use Cases for the sta_get_paths Command

The following lists the use cases for the sta_get_paths command:

- Reporting Specific Paths: To report a specific number of paths in your design, use the -n flag.
  For example, to report the top 5 paths:
  ```
  sta_get_paths -n 5
  ```
- Analyzing Paths from Specific Ports/Pins: To analyze paths originating from specific ports or pins, use the -from flag.
  For example, this command reports paths from the start_pin:
  ```
  sta_get_paths -from start_pin
  ```
- Analyzing Paths to Specific Ports/Pins: To analyze the paths leading to specific ports or pins, use the -to flag.
  For example, this command reports paths to the end_pin:
  ```
  sta_get_paths -to end_pin
  ```

- Analyzing Paths from/to Specific Clocks: To analyze paths from or to specific clocks, use -from clock and -to clock flags. For example, these commands report paths from the clk1 and clk2 clocks:
  ```
  sta_get_paths -from_clock clk1
  sta_get_paths -to_clock clk2
  ```
- Hold Time Analysis: In hold time analysis, use the -hold flag to report hold paths.
  For example, this command reports the hold paths in your design:
  ```
  sta_get_paths -hold.
  ```
- Analyzing Paths from Specific Clocks to Specific Pins: To analyze paths from a specific clock to a specific pin, use -from_clock and -to flags.
  For example, this command reports paths from the clk1 clock to the end_pin:
  ```
  sta_get_paths -from_clock clk1 -to end_pin
  ```

3.6. sta_get_slack Command

The sta_get_slack command is used to retrieve the total negative, worst, or terminal slack for the design. Slack is a critical parameter in digital design, representing the amount of time that you can delay a signal without causing a design to fail.

Syntax:
```
sta_get_slack [-help] [-tns] [-worst] [-hold] [-terminal <terminal_name>]
```

The following table lists the flags that you can use with the sta_get_slack command.

Table 3.9. sta_get_slack Usage Description

Flag	Description
-help	To provide help information related to the sta_get_slack command.
-tns	To get the total negative setup or hold slack.
-worst	To get the worst setup or hold slack of the design.
-hold	To get hold slack.
-terminal	This is a string parameter that represents the name of the pin or port. When specified, the command gets the worst slack for the given pin or port. The name must represent exactly one port or pin.

3.6.1. Use Cases for the sta_get_slack Command

The following lists the use cases for the sta_get_slack command:

- Total Negative Slack (TNS) Analysis: To analyze the total negative slack in your design, use the -tns flag.
  For example, this command returns the total negative setup or hold slack:
  ```
  sta_get_slack -tns
  ```
- Worst Slack Analysis: To analyze paths originating from specific ports or pins, use the -from flag.
  For example, this command reports paths from the start_pin:
  ```
  sta_get_slack -worst
  ```
- Hold Slack Analysis: To analyze the hold slack in your design, use the -hold flag.
  For example, this command returns the hold slack:
  ```
  sta_get_slack -hold
  ```

- Terminal Slack Analysis: To analyze the slack for a specific pin or port, use the -terminal flag.
  For example, this command returns the worst slack for the pin1:
  ```
  sta_get_slack -terminal pin1
  ```

3.7. sta_get_terms Command

The sta_get_terms command is used to retrieve a list of ports and pins that matches a specified name. This is a critical feature in the FPGA design, which allows you to quickly locate and work with specific elements of your design.

Syntax:
```
sta_get_terms [-help] [-ports] [-pins] [-name <terminal_name>]
```

The following table lists the flags that you can use with the sta_get_term command.

Table 3.10. sta_get_terms Usage Description

Flag	Description
-help	To provide help information related to the sta_get_terms command.
-ports	To return ports only.
-pins	To return pins only.
-name	This is a string parameter that represents the name of the terminals. When specified, the command gets the terminals that matches the given name.

3.7.1. Use Cases for the sta_get_terms Command

The following lists the use cases for the sta_get_terms command:

- Filtering Ports: To get only the ports in your design, use the -ports flag:
  ```
  sta_get_terms -ports
  ```
- Filtering Pins: To get only the pins in your design, use the -pins flag:
  ```
  sta_get_terms -pins
  ```
- Searching for Specific Terminals: To get terminals with a specific name, use the -name flag.
  For example, the following command returns terminals with the name *CLK*:
  ```
  sta_get_terms -name CLK
  ```

3.8. sta_path_info Command

The sta_path_info command is used to extract data from a given path. This is useful for analyzing specific paths in a design for timing, constraints, and other parameters.

Syntax:
```
sta_path_info [-help] [-path <path_name>] [-slack] [-constraint] [-skew] [-launch_clock] [-
      latch_clock] [-levels]
```

The following table lists flags that you can use with the sta_path_info command.

Table 3.11. sta_path_info Usage Description

Flag	Description
-help	Provides help information related to the sta_path_info command.
-path	This is a string parameter that represents the path in question.
-slack	To get the slack of the path.
-constraint	To get the constraint of the path.
-skew	To get the clock skew – common skew not removed.
-launch_clock	To get the launch clock.
-latch_clock	To get the latch clock.
-levels	To get the number of logic levels.

Note:
sta_get_paths must be used before using the sta_path_info command.

3.8.1. Use Cases for the sta_path_info Command

The following lists the use cases for the sta_path_info command:

- Slack Analysis: To analyze the slack of a specific path, use the -slack flag.
  For example, this command returns the slack of path1:
  ```
  sta_path_info -path path1 -slack
  ```
- Constraint Analysis: To find the constraint of a specific path, use the -constraint flag.
  For example, this command returns the constraint of path1:
  ```
  sta_path_info -path path1 -constraint
  ```
- Clock Skew Analysis: To analyze the clock skew of a specific path, use the -skew flag.
  For example, this command returns the clock skew of path1:
  ```
  sta_path_info -path path1 -skew
  ```
- Launch Clock Analysis: To analyze the launch clock of a specific path, use the -launch_clock flag.
  For example, this command returns the launch clock of path1:
  ```
  sta_path_info -path path1 -launch_clock
  ```
- Latch Clock Analysis: To analyze the latch clock of a specific path, use the -latch_clock flag. For example, this command returns the latch clock of path1:
  ```
  sta_path_info -path path1 -latch_clock
  ```
- Logic Levels Analysis: To analyze the number of logic levels of a specific path, use the -levels flag. For example, this command returns the number of logic levels of path1:
  ```
  sta_path_info -path path1 -levels
  ```

```
# Example 1: Get paths from pin1 to pin2 and print their launch clock, latch clock, and
skew

set paths [ sta_get_paths -n 10 -from_clock CLK -to_clock clock_in ]

foreach path $paths {
    puts "Launch clock: [ sta_path_info -path $path -launch_clock ]"
    puts "Latch clock: [ sta_path_info -path $path -latch_clock ]"
    puts "Slack: [ sta_path_info -path $path -slack ]"
}
```

```
# Example 2: Get paths from a list of pins to another list of pins and print their launch
clock, latch clock, and skew

set from_pins {pin1 pin2 pin3}
set to_pins {pin4 pin5 pin6}
set paths [ sta_get_paths -n 10 -from $from_pins -to $to_pins ]

foreach path $paths {
    puts "Launch clock: [ sta_path_info -path $path -launch_clock ]"
    puts "Latch clock: [ sta_path_info -path $path -latch_clock ]"
    puts "Slack: [ sta_path_info -path $path -slack ]"
}
```

3.9. sta_report_clocks Command

The sta_report_clocks command is used to generate a report on a specific clock or multiple clocks. This is useful for analyzing the characteristics of your clocks, such as their frequency, source, and edges.

Syntax:

```
sta_report_clocks [-help] [-clocks <clock_name>]
```

The following table lists the flags that you can use with the sta_report_clocks command.

Table 3.12. sta_report_clocks Usage Description

Flag	Description
-help	Provides help information related to the sta_report_clocks command.
-clocks	This is a string parameter that represents the name of the clock or clocks. When specified, the command generates a report for the given clock or clocks.
-app	Used to append the clocks to a list or file.
-file	This is a string parameter that represents the name of the file for output. When specified, the command writes the output to the given file.

3.9.1. Use Cases for the sta_report_clocks Command

The following lists the use cases for the sta_report_clocks command:

- Clock Analysis: To analyze a specific clock in your design, use the -clocks flag. For example, this command returns a report for the *clk* clock:

```
sta_report_clocks -clocks clk
```

- Multiple Clock Analysis: To analyze multiple clocks in your design, specify multiple clock names separated by spaces. For example, this command returns a report for the *clk1* and *clk2* clocks:

```
sta_report_clocks -clocks "CLK clock_in"
```

This will provide a report like:

```
% sta_report_clocks -clocks {CLK clock_in}


=======================
create_generated_clock -name {CLK} -source [get_pins
{pll_inst/lscc_pll_inst/gen_no_refclk_mon.u_PLL.PLL_inst/REFCK}] -multiply_by 5 [get_pins
{pll_inst/lscc_pll_inst/gen_no_refclk_mon.u_PLL.PLL_inst/CLKOP }]

Single Clock Domain
-------------------------------------------------------------------------------------------------------
               Clock CLK                |                    |       Period       |     Frequency
-------------------------------------------------------------------------------------------------------
 From CLK                               |             Target |          10.000 ns |        100.000 MHz
                                        | Actual (all paths) |           4.805 ns |        208.117 MHz
grant_0io[7].PIC_inst/CLK (MPW)         |   (50% duty cycle) |           4.000 ns |        250.000 MHz
-------------------------------------------------------------------------------------------------------

Clock Domain Crossing
------------------------------------------------------------------------------------------------------
               Clock CLK                |   Worst Time Between Edges   |           Comment
------------------------------------------------------------------------------------------------------
 From clock_in                          |                         ---- |                      No path
------------------------------------------------------------------------------------------------------



=======================
create_clock -name {clock_in} -period 50 -waveform {0.000 25.000} [get_ports clock_in]

Single Clock Domain
-------------------------------------------------------------------------------------------------------
             Clock clock_in             |                    |       Period       |     Frequency
-------------------------------------------------------------------------------------------------------
 From clock_in                          |             Target |          50.000 ns |         20.000 MHz
                                        | Actual (all paths) |           5.000 ns |        200.000 MHz
clock_in_pad.bb_inst/B (MPW)            |   (50% duty cycle) |           5.000 ns |        200.000 MHz
-------------------------------------------------------------------------------------------------------

Clock Domain Crossing
------------------------------------------------------------------------------------------------------
             Clock clock_in             |   Worst Time Between Edges   |           Comment
------------------------------------------------------------------------------------------------------
 From CLK                               |                     9.999 ns |             slack = 6.263 ns
------------------------------------------------------------------------------------------------------------
```

- All Clocks Analysis: To analyze all clocks in your design, use the sta_report_clocks command without the -clocks flag. This command generates a report for all clocks in your design:
  ```
  sta_report_clocks
  ```

3.10. sta_report_constraints Command

The sta_report_constraints command is used to list all the timing constraints, which allows you to quickly locate and work with specific elements of your design.

Syntax:
```
sta_report_constraints [-help] [-file <filename>] [-app]
```

The following table lists the flags that you can use with the sta_report_clocks command.

Table 3.13. sta_report_constraints Usage Description

Flag	Description
-help	Provides help information related to the sta_report_constraints command.
-file	This is a string parameter that represents the name of the file for output. When specified, the command writes the output to the given file.
-app	This is a boolean flag. When specified, the command appends the output to a list or file.

3.10.1. Use Cases for the sta_report_constraints Command

The following lists the use cases for the sta_report_constraints command:

- Constraint Verification: This command lists all the constraints and verifies their correctness.
  For example, the following command sta_report_constraints command will list all the constraints:
  ```
  sta_report_constraints
  ```

- Writing Output to a File: To write the output to a file, use the -file flag.
  For example, this command writes the output to output.txt:
  ```
  sta_report_constraints -file output.txt
  ```

- Appending Output to a File: To append the output to an existing file, use the -file and -app flags.
  For example, this command appends the output to output.txt:
  ```
  sta_report_constraints -file output.txt -app
  ```

3.11. sta_report_timing Command

The sta_report_timing command is used to report the detailed or summary of timing, which allows you to quickly understand the timing performance of your design.

Syntax:

```
sta_report_timing [-help] [-n <number_of_paths>] [-endpoints] [-from <from_ports/pins>] [-
from_clock <from_clocks>] [-to <to_ports/pins>] [-to_clock <to_clocks>] [-hold] [-summary] [-
file <filename>] [-app]
```

The following table lists the flags that you can use with the sta_report_clocks command.

Table 3.14. sta_report_timing Usage Description

Flag	Description
-help	Provides help information related to the sta_report_timing command.
-n	This is an integer parameter that represents the number of paths to report. The default value is -2.
-endpoints	This is a boolean flag. When specified, the command reports the endpoints only.
-from	This is a string parameter that represents the names of ports and/or pins for the starting point of the paths.
-from_clock	This is a string parameter that represents the names of clocks for the starting point of the paths.
-to	This is a string parameter that represents the names of ports and/or pins for the ending point of the paths.
-to_clock	This is a string parameter that represents the names of clocks for the ending point of the paths.
-hold	This is a boolean flag. When specified, the command reports hold paths.
-summary	This is a boolean flag. When specified, the command reports the timing summary.
-file	This is a string parameter that represents the name of the file for output. When specified, the command writes the output to the given file.
-app	This is a boolean flag. When specified, the command appends the output to the existing content of the file. If the file does not exist, it will be created.

3.11.1. Use Cases for the sta_report_timing Command

The following lists the use cases for the sta_report_timing command:

- Detailed Timing Analysis: To perform a detailed timing analysis of your design, use the sta_report_timing command.
  For example, this command will report the detailed timing:
  ```
  sta_report_timing
  ```

- Reporting Specific Number of Paths: To report a specific number of paths, use the -n flag.
  For example, this command will report the top 10 paths:
  ```
  sta_report_timing -n 10
  ```

- Reporting Endpoints Only: To report only the endpoints, use the -endpoints flag.
  For example, this command will report only the endpoints:
  ```
  sta_report_timing -endpoints
  ```

- Reporting From Specific Ports/Pins or Clocks: To report from specific ports/pins or clocks, use the -from or -from_clock flags.
  For example, this command will report from the port port1 and the clock clk1:
  ```
  sta_report_timing -from port1 -from_clock clk1
  ```

- Reporting To Specific Ports/Pins or Clocks: To report to specific ports/pins or clocks, use the -to or -to_clock flags.
  For example, this command will report to the port2 port and the clk2 clock:
  ```
  sta_report_timing -to port2 -to_clock clk2
  ```

- Reporting Hold Paths: To report hold paths, use the -hold flag.
  For example, this command will report hold paths:
  ```
  sta_report_timing -hold
  ```

- Reporting Timing Summary: To report the timing summary, use the -summary flag.
  For example, this command will report the timing summary:
  ```
  sta_report_timing -summary
  ```

- Writing Output to a File: To write the output to a file, use the -file flag.
  For example, this command will write the output to timing.txt:
  ```
  sta_report_timing -file timing.twr
  ```

- Appending Output to a File: To append the output to an existing file, use the -file and -app flags.
  For example, this command will append the output to timing.txt:
  ```
  sta_report_timing -file timing.twr -app
  ```

3.12. sta_report_unconstrained Command

The sta_report_unconstrained command is used to list unconstrained ports, pins, or nets. The command writes out unconstrained endpoints indicated by the various flags. If no flags are provided, all types of unconstrained objects are dumped.

Syntax:
```
sta_report_unconstrained [-help] [-n <number>] [-start] [-end] [-io] [-clock_nets] [-file
<filename>] [-app]
```

The following table lists the flags that you can use with the sta_report_clocks command.

Table 3.15. sta_report_unconstrained Usage Description

Flag	Description
-help	Provides help information related to the sta_report_unconstrained command.
-n	This is an integer parameter that represents the number of unconstrained objects to report. The default value is 10.
-start	This is a boolean flag. When specified, the command reports unconstrained internal start points.
-end	This is a boolean flag. When specified, the command reports unconstrained internal endpoints.
-io	This is a boolean flag. When specified, the command reports unconstrained I/O start and end points.
-clock_nets	This is a boolean flag. When specified, the command reports clock nets without clock definition. The -start flag has no effect on -clock_nets.
-file	This is a string parameter that represents the name of the file for output. When specified, the command writes the output to the given file.
-app	This is a boolean flag. When specified, the command appends the output to the existing content of the file. If the file does not exist, it will be created.

3.12.1. Use Cases for the sta_report_unconstrained Command

The following lists the use cases for the sta_report_unconstrained command:

- Reporting Unconstrained Objects: To report all types of unconstrained objects, use the sta_report_unconstrained command.
  For example, this command will report all types of unconstrained objects:
  ```
  sta_report_unconstrained
  ```

- Reporting Specific Number of Unconstrained Objects: To report a specific number of unconstrained objects, use the -n flag.
  For example, this command will report the top 10 unconstrained objects:
  ```
  sta_report_unconstrained -n 10
  ```

- Reporting Unconstrained Internal Start Points: To report unconstrained internal start points, use the -start flag.
  For example, this command will report unconstrained internal start points:
  ```
  sta_report_unconstrained -start
  ```

- Reporting Unconstrained Internal Endpoints: To report unconstrained internal endpoints, use the -end flag.
  For example, this command will report unconstrained internal endpoints:
  ```
  sta_report_unconstrained -end
  ```

- Reporting Unconstrained IO Start and End Points: To report unconstrained IO start and end points, use the -io flag.
  For example, this command will report unconstrained IO start and end points:
  ```
  sta_report_unconstrained -io
  ```

- Reporting Clock Nets Without Clock Definition: To report clock nets without clock definition, you can use the -clock_nets flag.
  For example, this command will report clock nets without clock definition:
  ```
  sta_report_unconstrained -clock_nets
  ```

- Writing Output to a File: To write the output to a file, use the -file flag.
  For example, this command will write the output to unconstrained.txt:
  ```
  sta_report_unconstrained -file unconstrained.txt
  ```

- Appending Output to a File: To append the output to an existing file, use the -file and -app flags.
  For example, this command will append the output to unconstrained.txt:
  ```
  sta_report_unconstrained -file unconstrained.txt -app
  ```

You can use the above commands to create your own script and your own timing report formats.

References

- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Lattice Radiant Software web page.
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, July 2024

Section	Change Summary
All	Initial release.

www.latticesemi.com