Using the Synopsys® Design Constraints Format Application Note

Version 2.0, December 2012

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

Disclaimer

Trademarks

Synopsys, Inc.
700 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

Appendix A. SDC Syntax

Preface

This preface includes the following sections:

What's New in This Release

The Synopsys Design Constraints (SDC) format version 2.0 includes the following enhancements:

- Support for the `-reference_pin` option of the `set_input_delay` and `set_output_delay` commands
- Support for the `set_min_pulse_width` command

About This Application Note

This application note describes the methodology and commands used to transfer constraint information between Synopsys tools and third-party tools using the SDC format.

SDC version 2.0 was introduced in December 2012. It is the recommended SDC version to use with the following Synopsys tools:

- PrimeTime version H-2012.12 and later releases
- Design Compiler and IC Compiler version H-2013.03 and later releases

Audience

This application note is for engineers who use the SDC format to transfer constraint information between Design Compiler, IC Compiler, or PrimeTime and third-party tools.

Related Publications

For additional information about SDC, see the documentation on SolvNet at the following address:

https://solvnet.synopsys.com/DocsOnWeb

You might also want to see the documentation for the following related Synopsys products:

- Design Compiler
- IC Compiler
- PrimeTime

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates syntax, such as `write_file`.
`Courier italic`	Indicates a user-defined value in syntax, such as `write_file` *`design_list`*.
`Courier bold`	Indicates user input—text you type verbatim—in examples, such as `prompt>` **`write_file top`**
[ ]	Denotes optional arguments in syntax, such as `write_file [-format` *`fmt`*`]`
...	Indicates that arguments can be repeated as many times as needed, such as *`pin1 pin2 ... pinN`*
|	Indicates a choice among alternatives, such as `low | medium | high`
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access SolvNet, go to the following address:

https://solvnet.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.

If you need help using SolvNet, click HELP in the top-right menu bar.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a support case to your local support center online by signing in to SolvNet at https://solvnet.synopsys.com, clicking Support, and then clicking “Open A Support Case.”
- Send an e-mail message to your local support center.
  - E-mail support_center@synopsys.com from within North America.
  - Find other local support center e-mail addresses at http://www.synopsys.com/Support/GlobalSupportCenters/Pages
- Telephone your local support center.
  - Call (800) 245-8005 from within North America.
  - Find other local support center telephone numbers at http://www.synopsys.com/Support/GlobalSupportCenters/Pages

1

Using the Synopsys Design Constraints Format

The Synopsys Design Constraints (SDC) format is used to specify the design intent, including the timing, power, and area constraints for a design. SDC is based on the tool command language (Tcl). The Synopsys Design Compiler, IC Compiler, and PrimeTime tools use the SDC description to synthesize and analyze a design. In addition, these tools can generate SDC descriptions for and read SDC descriptions from third-party tools.

Figure 1-1 shows the SDC-based flow for sharing constraint information between Synopsys tools and third-party EDA tools.

Figure 1-1 SDC-Based Constraint Interface

Note:

There are slight differences between the SDC files generated by the Synopsys tools. For more information, see “About the Generated SDC File” on page 1-10.

To learn about the SDC-based interface, see the following sections:

- About the SDC Format
- Generating SDC Files
- Reading SDC Files Into a Synopsys Tool
- Managing Large SDC Files

About the SDC Format

SDC is a Tcl-based format. All commands in an SDC file conform to the Tcl syntax rules.

You use an SDC file to communicate the design intent, including timing and area requirements between EDA tools. An SDC file contains the following information:

- SDC version (optional)
- SDC units (optional)
- Design constraints
- Design objects
- Comments (optional)

Note:
An SDC file does not contain commands to load or link the design. You must perform these tasks before reading an SDC file.

Specifying the SDC Version

SDC version 2.0 was introduced in December 2012. It is the recommended SDC version to use with the following Synopsys tools:

- PrimeTime version H-2012.12 and later releases
- Design Compiler and IC Compiler version H-2013.03 and later releases

If you do not specify a version, Design Compiler, IC Compiler, and PrimeTime assume that the file uses SDC version 2.0 syntax. For a listing of the SDC syntax, see "SDC Syntax" in Appendix A.

If you are using a third-party EDA tool that requires an earlier version of the SDC format, set the `sdc_version` variable to ensure compatibility with the Synopsys tools.

To specify the SDC version in an SDC file, begin the file with the following command:

```
set sdc_version value
```

Specifying Units

The `set_units` command specifies the units used in the SDC file. You can specify the units for capacitance, resistance, time, voltage, current, and power. For the complete list of options for the `set_units` command, see "SDC Syntax" in Appendix A.

Specifying Design Constraints

Specify design constraints using constraint commands. You can break up a long command line into multiple lines by using the backslash character (\) to indicate command continuation. The SDC format consists of the constraint commands listed in Table 1-1.

Note:

The SDC format supports a subset of the command arguments, as compared to the arguments supported by individual tools. For a listing of the SDC arguments, see "SDC Syntax" in Appendix A. For information about individual tool support, see SolvNet article 015193. For information about validating your SDC file, see "Using Synopsys Tools to Validate SDC Files" on page 1-13.

Table 1-1 SDC Commands

Type of information	Commands
Operating conditions	`set_operating_conditions`

Wire load models	`set_wire_load_min_block_size`
	`set_wire_load_mode`
	`set_wire_load_model`
	`set_wire_load_selection_group`

System interface	`set_drive`
	`set_driving_cell`
	`set_fanout_load`
	`set_input_transition`
	`set_load`
	`set_port_fanout_number`

Design rule constraints	`set_max_capacitance`
	`set_min_capacitance`
	`set_max_fanout`
	`set_max_transition`

Table 1-1 SDC Commands (Continued)

Type of information	Commands

Timing constraints	`create_clock`
	`create_generated_clock`
	`group_path`
	`set_clock_gating_check`
	`set_clock_groups`
	`set_clock_latency`
	`set_clock_sense`
	`set_clock_transition`
	`set_clock_uncertainty`
	`set_data_check`
	`set_disable_timing`
	`set_ideal_latency`
	`set_ideal_network`
	`set_ideal_transition`
	`set_input_delay`
	`set_max_time_borrow`
	`set_min_pulse_width`
	`set_output_delay`
	`set_propagated_clock`
	`set_resistance`
	`set_timing_derate`

Timing exceptions	`set_false_path`
	`set_max_delay`
	`set_min_delay`
	`set_multicycle_path`

Area constraints	`set_max_area`

Multivoltage and power optimization constraints	`create_voltage_area`
	`set_level_shifter_strategy`
	`set_level_shifter_threshold`
	`set_max_dynamic_power`
	`set_max_leakage_power`

Table 1-1 SDC Commands (Continued)

Type of information	Commands

Logic assignments	`set_case_analysis`
	`set_logic_dc`
	`set_logic_one`
	`set_logic_zero`

Specifying Design Objects

Most of the constraint commands require a design object as a command argument. SDC supports both implicit and explicit object specification.

If you specify a simple name for an object, the Synopsys tools determine the object type by searching for the object using a prioritized object list. The priority order varies by command and is documented in the tool's man page of each command. This is called implicit object specification.

To avoid ambiguity, explicitly specify the object type by using a nested object access command. For example, if you have a cell in the current instance named U1, the implicit specification is `U1`, while the explicit specification is `[get_cells U1]`.

Table 1-2 shows the design objects supported by the SDC format and the access commands used for explicit object specification.

Note:

The SDC format supports a subset of the access command syntax, as compared to the syntax supported by individual tools. For a listing of the SDC syntax, see "SDC Syntax" in Appendix A. For information about individual tool support, see SolvNet article 015193, "Synopsys Design Constraints (SDC) for Implementation Tools."

Table 1-2 SDC Design Objects

Design object	Access command	Description
design	`current_design`	A container for cells. A block.
clock[1]	`get_clocks` `all_clocks`	A clock in a design. All clocks in a design.

port	`get_ports`
	`all_inputs`
	`all_outputs`
		An entry point to or exit point from a design.
		All entry points to a design.
		All exit points from a design.

cell	`get_cells`	An instance of a design or library cell.
pin	`get_pins`	An instance of a design port or library cell pin.
net	`get_nets`	A connection between cell pins and design ports.
library	`get_libs`	A container for library cells.
lib_cell	`get_lib_cells`	A primitive logic element.
lib_pin	`get_lib_pins`	An entry point to or exit point from a lib_cell.
register	`all_registers`	A sequential logic cell.

1. The clock design object includes both standard clocks and generated clocks.

Specifying Multiple Objects

Both the constraint commands and the object access commands follow the Tcl syntax rules. Use a Tcl list or wildcard characters to specify multiple objects. SDC supports the following wildcard characters:

`?` Matches exactly one character.

`*` Matches zero or more characters.

Note:

If you do not specify an object argument for an object access command, SDC interprets the command as if you specified the `*` wildcard character.

Specifying Hierarchical Objects

The reference point for all object specifications is the current instance. By default, the top-level design is the current instance. You can change the current instance by using the `current_instance` command.

Design Compiler and IC Compiler always use a slash (`/`) as the hierarchy separator. PrimeTime supports a user-defined hierarchy separator (as specified by the `hierarchy_separator` variable), with a slash (`/`) being the default.

In some cases, the character used to indicate hierarchy levels (the hierarchy separator character) is also used within design object names. This can lead to an ambiguous hierarchy definition within the SDC file.

Note:

The hierarchy definition is never ambiguous within the Synopsys tool, because the search engines within these tools can correctly decode the object names.

The SDC format supports the following characters as hierarchy separator characters: slash (`/`), at sign (`@`), caret (`^`), pound sign (`#`), period (`.`), and vertical bar (`|`). By default, the hierarchy separator is the slash (`/`).

To create an unambiguous hierarchy definition, the SDC file uses another character as the hierarchy separator character whenever a design uses a slash (`/`) within object names. Within the SDC file, a nondefault hierarchy separator character is specified either globally, using the `set_hierarchy_separator` statement, or locally, by using the `-hsc` option on the object access commands.

Specifying Buses

Specify buses using the Verilog-style naming convention *name*[*index*] and enclose the name in curly braces. For example,

```
create_clock -period 10 [get_clocks {CLK[0]}]
```

Adding Comments

You can add comments to an SDC file either as complete lines or as fragments after a command. To identify a line as a comment, start the line with a pound sign (#). For example,

```
# This is an SDC comment line.
```

Add inline comments using a semicolon to end the command, followed by the pound sign (#) to begin the comment. For example,

```
create_clock -period 10 [get_ports CLK]; # comment fragment
```

Using the -comment Option

To include user-specific comments, use the `-comment` option. The comment string associated with the specific command is written out when you use the `write_sdc` or `write_script` command. The tool issues a message if a comment is too long or the overall allocated storage is reached. The following SDC commands support the `-comment` option:

- `create_clock`
- `create_generated_clock`
- `group_path`
- `set_clock_groups`
- `set_false_path`
- `set_max_delay`
- `set_min_delay`
- `set_multicycle_path`

The following example shows how to use the `-comment` option:

```
create_clock -period 10 [get_ports CLK] -comment "for blk1 in Test Mode"
```

Generating SDC Files

You can generate an SDC file in the following ways:

- Using the Synopsys Design Compiler, IC Compiler, or PrimeTime tools
- Using a third-party EDA tool that supports the SDC format
- Writing the file manually

The SDC files generated by Synopsys tools always meet the SDC format requirements. If you generate an SDC file using a third-party tool or by writing the file manually, you should validate the file syntax. For information about validating the file syntax, see “Using Synopsys Tools to Validate SDC Files” on page 1-13.

Generating SDC Files From a Synopsys Tool

To generate an SDC file from Design Compiler, IC Compiler, or PrimeTime, use the `write_sdc` command, which writes the constraints for the current design and its hierarchy to the specified file. By default, the `write_sdc` command generates the file with the latest syntax. To generate a file with an earlier SDC version, use the `-version` option with the `write_sdc` command.

When you generate an SDC file, the `write_sdc` command writes the design units, as specified in the main library file, to the SDC file.

The constraints can either be set from a script file or derived through characterization or budgeting. The order of commands in the SDC file does not indicate constraint precedence.

The `write_sdc` command writes the design constraints to the SDC file in expanded format. This means that the generated SDC files contain a command for each constraint attribute that exists on each design object. Each design object is represented by its full hierarchical name and is selected by using the appropriate object access function (see Table 1-2 on page 1-7 for a listing of object access functions). Each command line contains all command options; those that are not specified on the design are assigned default values. For information about the expanded format, see “About the Generated SDC File” on page 1-10.

Buses that have constraints set on them get expanded when you run the `write_sdc` command. For example, if you use the `set_input_delay` command on a bus, when you run the `write_sdc` command, Design Compiler, IC Compiler, and PrimeTime expands the bus name to all bits of the bus.

Because the constraints are written in expanded format, the size of the SDC file increases proportionately with the number of constraints. In particular, the use of timing exceptions increases the size of the generated SDC file. For tips about how to use these large files, see “Managing Large SDC Files” on page 1-14.

Note:

The commands generated by the `write_sdc` command might differ amongst Synopsys tools. However, the generated commands meet the SDC requirements and capture the same intent.

About the Generated SDC File

Although the SDC file generated by the `write_sdc` command captures the same intent as the constraints you specified, the format of the constraints are not identical to the input format you used. In addition, there are slight differences between the SDC file generated by the different Synopsys tools. For example, assume you enter the following constraint:

```
create_clock -period 100 clk
```

The SDC file generated by Design Compiler represents this constraint as

```
create_clock -period 100 -waveform {0 50} [get_ports {clk}]
```

The SDC file generated by PrimeTime represents this constraint as

```
create_clock -name clk -period 100.000000 \
   -waveform { 0.000000 50.000000 } [get_ports {clk}]
```

The SDC file generated by the `write_sdc` command might differ from the input constraints in the following ways:

- Specification of design objects
  - Explicit specification

    The SDC file specifies all design objects using object access commands (see Table 1-2 on page 1-7 for the listing of object access commands for each design object). Because the argument to the object access commands is a Tcl list, the SDC file expresses the design objects as a Tcl list (either as a list of strings within curly braces ({}) or by using the Tcl list command). For example, if you specified clock CLK using the `create_clock -period 10 CLK` command, the corresponding SDC command is (the added text is shown in bold):

    ```
    create_clock -period 10 [get_clocks {CLK}]
    ```

  - Direct specification

    Direct specification of a design object uses the object name as the argument to the object access command. You can indirectly specify design objects through use of the `-of_objects` option of an object access command. The SDC file specifies all objects directly. For example, if you specified port IN1 using the following command,

    ```
    set_input_delay 5 -clock [get_clocks CLK] \
       [get_ports -of_objects [get_nets n_in1]]
    ```

    the corresponding SDC command is (changed text is shown in bold):

    ```
    set_input_delay 5 -clock [get_clocks {CLK}] [get_ports {IN1}]
    ```

  - Wildcard expansion

    The generated SDC file does not include wildcard characters. In some cases, the SDC file includes a separate command for each design object represented by a wildcard specification. In other cases, the SDC file includes a single command with a list of design objects as its argument. For example, if you specified ports IN1, IN2, and IN3 using the following command,

    ```
    set_input_delay 5 -clock [get_clocks CLK] [get_ports IN*]
    ```

the corresponding SDC commands are (changed text is shown in bold):

```
set_input_delay 5 -clock [get_clocks {CLK}] [get_ports {IN1}]
set_input_delay 5 -clock [get_clocks {CLK}] [get_ports {IN2}]
set_input_delay 5 -clock [get_clocks {CLK}] [get_ports {IN3}]
```

If you specified ports IN1, IN2, and IN3 using the following command,

```
set_false_path -from [get_ports IN*]
```

the corresponding SDC command is (changed text is shown in bold):

```
set_false_path -from [get_ports {IN1 IN2 IN3}]
```

- Hierarchy separator character

  If the hierarchy separator character is used in an object name, the tool uses a different hierarchy separator character in the SDC file to make the hierarchy definition unambiguous. For example, assume the design contains a cell named U1/U2, where / is part of the cell name and does not indicate hierarchy. To specify a false path on pin A of this cell, you enter the following command:

  ```
  set_false_path -to [get_pins {U1/U2/A}]
  ```

  The corresponding SDC command is (changed text is shown in bold):

  ```
  set_false_path -to [get_pins -hsc "@" {U1/U2@A}]
  ```

  Note:

  If you are using a third-party tool that does not support the unambiguous hierarchical names feature of SDC, you can disable this feature by setting the `sdc_write_unambiguous_names` variable to false. The `write_sdc` command issues a warning if you have set this variable to false.

- Object conversion

  In some cases, when you apply a constraint to a cell, the Synopsys tools interpret this as applying the constraint to the cell pins. In these cases, the `write_sdc` command specifies the constraints on the pins, not on the cells. For example, if you use the `set_disable_timing` command on a cell, the Synopsys tools interpret this as setting the `disable_timing` constraint on the cell output pins. Therefore, if you specify the following input constraint,

  ```
  set_disable_timing U1/buf2
  ```

  The following shows the corresponding SDC command whereby the changed text is shown in bold:

  ```
  set_disable_timing [get_pins {U1/buf2/Z}]
  ```

Using Synopsys Tools to Validate SDC Files

To validate the syntax of an SDC file, use the `read_sdc -syntax_only` command. This command generates warning messages if your SDC file contains unsupported commands or arguments. Fix any reported problems before using the SDC file to share constraint information.

For more information about the `read_sdc` command, see the next section, "Reading SDC Files Into a Synopsys Tool" and the specific man page.

Reading SDC Files Into a Synopsys Tool

To read an SDC file into Design Compiler, IC Compiler, or PrimeTime, use the `read_sdc` command. For information about SDC file requirements, see "About the SDC Format" on page 1-3.

Determining the SDC Version

The `read_sdc` command determines the version of the SDC file in the following ways (listed in order of priority):

1. The `-version` option specified on the `read_sdc` command line
2. The `sdc_version` variable specified in the SDC file
3. The default SDC version, which is the latest available syntax

Determining the Hierarchy Separator Character

The `read_sdc` command determines the hierarchy separator character used in the SDC file in the following ways, listed in order of priority:

1. The `-hsc` option on the object access commands

   This option specifies the hierarchy separator character used in that object access command.

2. The `set_hierarchy_separator` statement

   This statement specifies the default hierarchy separator character used within the SDC file.

3. The SDC default hierarchy separator (/)

Managing Large SDC Files

Because constraints are written in expanded form, the SDC file size can become large. In particular, using wildcard characters to specify timing exceptions can result in large SDC files. One way to reduce the disk space required for an SDC file is to compress the file using the UNIX gzip utility, as shown in Example 1-1.

If you are using PrimeTime, you can write an SDC file directly to a compressed file by using the `write_sdc` command with the `-compress gzip` option. For example,

```
write_sdc -compress gzip design.sdc.gz
```

Note:

You can use this method only on UNIX platforms with a Tcl-based tool.

Example 1-1 Tcl Procedure for Writing a Compressed SDC File

```
proc write_sdc_gzip {fname} {
  sh mknod my_pipe p
  sh gzip -c < my_pipe > $fname &
  write_sdc -output my_pipe
  sh rm my_pipe
}
```

The `read_sdc` command automatically detects gzip compressed files and uncompresses the files as it reads them. For example,

```
read_sdc design.sdc.gz
```

A

SDC Syntax

The following sections list the commands and arguments supported by SDC version 2.0:

- General-Purpose Commands
- Object Access Commands
- Timing Constraints
- Environment Commands
- Multivoltage and Power Optimization Commands

Note:

For information about individual tool support, see SolvNet article 015193, “Synopsys Design Constraints (SDC) for Implementation Tools.”

General-Purpose Commands

Table A-1 General-Purpose Commands

Command	Supported arguments
current_instance	[*instance*]
expr	*arg1 arg2 ... argn*
list	*arg1 arg2 ... argn*
set	*variable_name value*
set_hierarchy_separator	*separator*
set_units	[-capacitance *cap_units*] [-resistance *res_unit*] [-time *time_unit*] [-voltage *voltage_units*] [-current *current_unit*] [-power *power_unit*]

Object Access Commands

Table A-2 Object Access Commands

Command	Supported arguments
all_clocks	
all_inputs	[-level_sensitive] [-edge_triggered] [-clock *clock_name*]
all_outputs	[-level_sensitive] [-edge_triggered] [-clock *clock_name*]

Table A-2 Object Access Commands (Continued)

Command	Supported arguments

`all_registers`
(supported only by `read_sdc`)
	`[-no_hierarchy]`
	`[-hsc` *`separator`*`]`
	`[-clock` *`clock_name`*`]`
	`[-rise_clock` *`clock_name`*`]`
	`[-fall_clock` *`clock_name`*`]`
	`[-cells]`
	`[-data_pins]`
	`[-clock_pins]`
	`[-slave_clock_pins]`
	`[-async_pins]`
	`[-output_pins]`
	`[-level_sensitive]`
	`[-edge_triggered]`
	`[-master_slave]`

`current_design`	

`get_cells`	`[-hierarchical]`
	`[-regexp]`
	`[-nocase]`
	`-of_objects` *`objects`*
	`patterns`

`get_clocks`	`[-regexp]`
	`[-nocase]`
	`patterns`

`get_lib_cells`	`[-regexp]`
	`[-hsc` *`separator`*`]`
	`[-nocase]`
	`patterns`

`get_lib_pins`	`[-regexp]`
	`[-nocase]`
	`patterns`

Table A-2 Object Access Commands (Continued)

Command	Supported arguments
get_libs	[-regexp] [-nocase] *patterns*
get_nets	[-hierarchical] [-hsc *separator*] [-regexp] [-nocase] -of_objects *objects* *patterns*
get_pins	[-hierarchical] [-hsc *separator*] [-regexp] [-nocase] -of_objects *objects* *patterns*
get_ports	[-regexp] [-nocase] *patterns*

Timing Constraints

Table A-3 Timing Constraints

Command	Supported arguments
create_clock	-period *period_value* [-name *clock_name*] [-waveform *edge_list*] [-add] [-comment *comment_string*] [*source_objects*]
create_generated_clock	[-name *clock_name*] -source *master_pin* [-edges *edge_list*] [-divide_by *factor*] [-multiply_by *factor*] [-duty_cycle *percent*] [-invert] [-edge_shift *shift_list*] [-add] [-master_clock *clock*] [-combinational] [-comment *comment_string*] *source_objects*
group_path	[-name *group_name*] [-default] [-weight *weight_value*] [-from *from_list*] [-rise_from *from_list*] [-fall_from *from_list*] [-to *to_list*] [-rise_to *to_list*] [-fall_to *to_list*] [-through *through_list*] [-rise_through *through_list*] [-fall_through *through_list*] [-comment *comment_string*]

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_clock_gating_check	[-setup *setup_value*] [-hold *hold_value*] [-rise] [-fall] [-high] [-low] [*object_list*]
set_clock_groups	-group *clock_list* [-logically_exclusive] [-physically_exclusive] [-asynchronous] [-allow_paths] [-name *name*] [-comment *comment_string*]
set_clock_latency	[-rise] [-fall] [-min] [-max] [-source] [-late] [-early] [-clock *clock_list*] *delay* *object_list*
set_clock_sense	[-positive] [-negative] [-pulse pulse] [-stop_propagation] [-clock *clock_list*] *pin_list*

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_clock_transition	[-rise] [-fall] [-min] [-max] *transition* *clock_list*
set_clock_uncertainty	[-from *from_clock*] [-rise_from *rise_from_clock*] [-fall_from *fall_from_clock*] [-to *to_clock*] [-rise_to *rise_to_clock*] [-fall_to *fall_to_clock*] [-rise] [-fall] [-setup] [-hold] *uncertainty* [*object_list*]
set_data_check	[-from *from_object*] [-to *to_object*] [-rise_from *from_object*] [-fall_from *from_object*] [-rise_to *to_object*] [-fall_to *to_object*] [-setup] [-hold] [-clock *clock_object*] *value*
set_disable_timing	[-from *from_pin_name*] [-to *to_pin_name*] *cell_pin_list*

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_false_path	[-setup] [-hold] [-rise] [-fall] [-from *from_list*] [-to *to_list*] -through *through_list*] [-rise_from *rise_from_list*] [-rise_to *rise_to_list*] [-rise_through *rise_through_list*] [-fall_from *fall_from_list*] [-fall_to *fall_to_list*] [-fall_through *fall_through_list*] [-comment *comment_string*]
set_ideal_latency	[-rise] [-fall] [-min] [-max] *delay* *object_list*
set_ideal_network	[-no_propagate] *object_list*
set_ideal_transition	[-rise] [-fall] [-min] [-max] *transition_time* *object_list*

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_input_delay	[-clock *clock_name*] [-reference_pin *pin_port_name*] [-clock_fall] [-level_sensitive] [-rise] [-fall] [-max] [-min] [-add_delay] [-network_latency_included] [-source_latency_included] *delay_value* *port_pin_list*
set_max_delay	[-rise] [-fall] [-from *from_list*] [-to *to_list*] [-through *through_list*] [-rise_from *rise_from_list*] [-rise_to *rise_to_list*] [-rise_through *rise_through_list*] [-fall_from *fall_from_list*] [-fall_to *fall_to_list*] [-fall_through *fall_through_list*] [-comment *comment_string*] *delay_value*
set_max_time_borrow	*delay_value* *object_list*

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_min_delay	[-rise] [-fall] [-from *from_list*] [-to *to_list*] [-through *through_list*] [-rise_from *rise_from_list*] [-rise_to *rise_to_list*] [-rise_through *rise_through_list*] [-fall_from *fall_from_list*] [-fall_to *fall_to_list*] [-fall_through *fall_through_list*] [-comment *comment_string*] *delay_value*
set_min_pulse_width	[-low] [-high] *value* [*object_list*]
set_multicycle_path	[-setup] [-hold] [-rise] [-fall] [-start] [-end] [-from *from_list*] [-to *to_list*] [-through *through_list*] [-rise_from *rise_from_list*] [-rise_to *rise_to_list*] [-rise_through *rise_through_list*] [-fall_from *fall_from_list*] [-fall_to *fall_to_list*] [-fall_through *fall_through_list*] [-comment *comment_string*] *path_multiplier*

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_output_delay	[-clock *clock_name*] [-reference_pin *pin_port_name*] [-clock_fall] [-level_sensitive] [-rise] [-fall] [-max] [-min] [-add_delay] [-network_latency_included] [-source_latency_included] *delay_value* *port_pin_list*
set_propagated_clock	*object_list*

Environment Commands

Table A-4 Environment Commands

Command	Supported arguments
`set_case_analysis`	*value* *port_or_pin_list* Note: *value* can be `0`, `1`, `rising`, or `falling`.

`set_drive`	`[-rise]`
	`[-fall]`
	`[-min]`
	`[-max]`
	resistance
	port_list

`set_driving_cell`	`[-lib_cell` *lib_cell_name*`]`
	`[-rise]`
	`[-fall]`
	`[-min]`
	`[-max]`
	`[-library` *lib_name*`]`
	`[-pin` *pin_name*`]`
	`[-from_pin` *from_pin_name*`]`
	`[-multiply_by` *factor*`]`
	`[-dont_scale]`
	`[-no_design_rule]`
	`[-clock` *clock_name*`]`
	`[-clock_fall]`
	`[-input_transition_rise` *rise_time*`]`
	`[-input_transition_fall` *fall_time*`]`
	port_list

`set_fanout_load`	*value* *port_list*

Table A-4 Environment Commands (Continued)

Command	Supported arguments
set_input_transition	[-rise] [-fall] [-min] [-max] [-clock *clock_name*] [-clock_fall] *transition* *port_list*
set_load	[-min] [-max] [-subtract_pin_load] [-pin_load] [-wire_load] *value* *objects*
set_logic_dc	*port_list*
set_logic_one	*port_list*
set_logic_zero	*port_list*
set_max_area	*area_value*
set_max_capacitance	*value* *object_list*
set_max_fanout	*value* *object_list*
set_max_transition	[-clock_path] [-data_path] [-rise] [-fall] *value* *object_list*

Table A-4 Environment Commands (Continued)

Command	Supported arguments
set_min_capacitance	*value* *object_list*
set_operating_conditions	[-library *lib_name*] [-analysis_type*analysis_type*] [-max *max_condition*] [-min *min_condition*] [-max_library *max_lib*] [-min_library *min_lib*] [-object_list *objects*] [*condition*]
set_port_fanout_number	*value* *port_list*
set_resistance	[-min] [-max] *value* *net_list*
set_timing_derate	[-cell_delay] [-cell_check] [-net_delay] [-data] [-clock] [-early] [-late] [-rise] [-fall] *derate_value* [*object_list*]
set_voltage	[-min *min_case_value*] [-object_list *list_of_power_nets*] *max_case_voltage*
set_wire_load_min_block_size	*size*

Table A-4 Environment Commands (Continued)

Command	Supported arguments
set_wire_load_mode	*mode_name*
set_wire_load_model	-name *model_name* [-library *lib_name*] [-min] [-max] [*object_list*]
set_wire_load_selection_group	[-library *lib_name*] [-min] [-max] *group_name* [*object_list*]

Multivoltage and Power Optimization Commands

Table A-5 Multivoltage and Power Optimization Commands

Command	Supported arguments
create_voltage_area	-name *name* [-coordinate *coordinate_list*] [-guard_band_x *float*] [-guard_band_y *float*] *cell_list*
set_level_shifter_strategy	[-rule *rule_type*]
set_level_shifter_threshold	[-voltage *float*] [-percent *float*]
set_max_dynamic_power	*power* [*unit*]
set_max_leakage_power	*power* [*unit*]