Using the Synopsys® Design Constraints Format Application Note

Version Z-2007.03, June 2007

Comments?
Send comments on the documentation by going to http://solvnet.synopsys.com, then clicking "Enter a Call to the Support Center."

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Cadabra, CATS, CRITIC, CSim, Design Compiler, DesignPower, DesignWare, EPIC, Formality, HSIM, HSPICE, iN-Phase, in-Sync, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Photolynx, Physical Compiler, PrimeTime, SiVL, SNUG, SolvNet, System Compiler, TetraMAX, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, Columbia, Columbia-CE, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, Direct Silicon Access, Discovery, Encore, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, HSIM^{plus}, HSPICE-Link, i-Virtual Stepper, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Raphael-NES, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, Taurus, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (^{SM})

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
ARM and AMBA are registered trademarks of ARM Limited.
Saber is a registered trademark of SabreMark Limited Partnership and is used under license.
All other product or company names may be trademarks of their respective owners.

Contents

Preface

This preface includes the following sections:

What’s New in This Release

This section describes the enhancements included in Synopsys Design Constraints (SDC) version 1.7.

Enhancements

SDC version 1.7 includes the following enhancements:

- Support for unit specification in the SDC file (`set_units` command)
- Support for the following commands:
  - `all_registers` (supported by `read_sdc` only)
  - `group_path`
  - `set_clock_groups`
  - `set_clock_sense`
  - `set_ideal_latency`
  - `set_ideal_network`
  - `set_ideal_transition`
- Support for the `-combinational` option for the `create_generated_clock` command
- Support for the `-rise_from`, `-rise_to`, `-rise_through`, `-fall_from`, `-fall_to`, and `-fall_through` options for the following commands:
  - `set_false_path`
  - `set_max_delay`

- `set_min_delay`
- `set_multicycle_path`

Known Limitations and Resolved STARs

Information about known problems and limitations, as well as about resolved Synopsys Technical Action Requests (STARs), is available in the product release notes in SolvNet.

To see the product release notes,

1. Go to http://solvnet.synopsys.com/ReleaseNotes. (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.)
2. Click the product name, then click the release you want in the list that appears at the bottom.

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 version 1.7 of the SDC format.

SDC version 1.7 was introduced in the Z-2007.03 release of Design Compiler, IC Compiler, Astro, and Jupiter XT. It is the recommended SDC version to use with version Z-2007.03 and later of these tools.

SDC version 1.6 is the recommended SDC version to use with version Y-2006.06 and later of PrimeTime.

Audience

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

Related Publications

For additional information about SDC, see

- Synopsys Online Documentation (SOLD), which is included with the software for CD users or is available to download through the Synopsys electronic software transfer (EST) system
- Documentation on the Web, which is available through SolvNet at http://solvnet.synopsys.com/DocsOnWeb
- The Synopsys MediaDocs Shop, from which you can order printed copies of some Synopsys documents, at http://mediadocs.synopsys.com

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

- Design Compiler
- IC Compiler
- Astro
- Jupiter XT
- PrimeTime

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates command syntax.
`Courier italic`	Indicates a user-defined value in Synopsys syntax, such as *`object_name`*. (A user-defined value that is not Synopsys syntax, such as a user-defined value in a Verilog or VHDL statement, is indicated by regular text font italic.)
`Courier bold`	Indicates user input—text you type verbatim—in Synopsys syntax and examples. (User input that is not Synopsys syntax, such as a user name or password you enter in a GUI, is indicated by regular text font bold.)
[ ]	Denotes optional parameters, such as *`pin1`* [*`pin2 ... pinN`*]
|	Indicates a choice among alternatives, such as `low | medium | high` (This example indicates that you can enter one of three possible values for an option: low, medium, or high.)
_	Connects terms that are read as a single term by the system, such as `set_annotated_delay`
Control-c	Indicates a keyboard combination, such as holding down the Control 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.

Note:

SolvNet online customer support and the Synopsys Technical Support Center are available to Synopsys customers only. TAP-in partners might not have access to these resources.

Accessing SolvNet

SolvNet includes an electronic 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 on the Web, and “Enter a Call With the Support Center.”

To access SolvNet,

1. Go to the SolvNet Web page at http://solvnet.synopsys.com.
2. 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 or in the footer.

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 call to your local support center from the Web by going to http://solvnet.synopsys.com (Synopsys user name and password required), then clicking “Enter a Call With the Support Center.”
- 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/support_ctr.
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at http://www.synopsys.com/support/support_ctr.

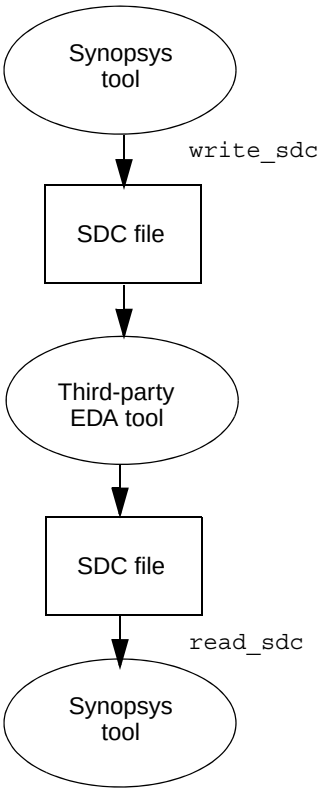
1

Using the Synopsys Design Constraints Format

Synopsys Design Constraints (SDC) is a format 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, Astro, Jupiter XT, 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. This application note describes how to share constraint information between third-party EDA tools and these Synopsys tools using SDC files. Figure 1-1 shows this SDC-based interface.

Use the SDC-based flow described in this document to share constraint information between Synopsys 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 1-13.

This application note describes the SDC-based interface in 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:

- The SDC version (optional)
- The SDC units (optional)
- The design constraints
- 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

The `sdc_version` variable specifies the SDC version for the file. To specify the SDC version, set this variable in the first command in the SDC file:

```
set sdc_version value
```

Note:

Astro and Jupiter XT do not support this variable. Astro and Jupiter XT assume that the file uses SDC version 1.7 syntax.

If the SDC file does not specify a version, Design Compiler, IC Compiler, PrimeTime, Astro, and JupiterXT assume that the file uses SDC version 1.7 syntax. The SDC syntax is specified in Appendix A, “SDC Syntax.”

SDC version 1.7 was introduced in the Y-2006.12 release of PrimeTme and the Z-2007.03 release of Design Compiler, IC Compiler, Astro and Jupiter XT. It is the recommended SDC version to use with version Z-2007.03 and later of these tools.

If the third-party EDA tool you are using supports an earlier version of the SDC format, set this variable to ensure compatibility with the Synopsys tools.

Specifying the 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.

The syntax of the `set_units` command is

```
set_units -capacitance cap_unit -resistance res_unit
   -time time_unit -voltage voltage_unit
   -current current_unit -power power_unit
```

Specifying the Design Constraints

You specify design constraints using Synopsys 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 Synopsys 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 the individual tools. For a listing of the supported arguments, see Appendix A, “SDC Syntax.” 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 1-17.

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_max_fanout`
	`set_max_transition`
	`set_min_capacitance`

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_output_delay`
	`set_propagated_clock`
	`set_resistance`
	`set_timing_derate`

Table 1-1 SDC Commands (Continued)

Type of information	Commands

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`

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 each command's man page. 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 the individual tools. For a listing of the supported syntax, see Appendix A, "SDC Syntax." For information about individual tool support, see SolvNet article 015193.

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:

In SDC version 1.5 and later, if you do not specify an object argument for an object access command, SDC interprets the command as if you specified the * wildcard character. However, in Astro and Jupiter XT, the `get_clocks` command requires an object argument; an error occurs if you do not specify the object argument for this command.

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.

Note:

Astro does not support 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 value.

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 (|), with the slash (/) being the default.

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. In Astro and Jupiter XT, you can also specify the hierarchy separator character in the dialog box for the `ataLoadSDC` Scheme command.

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]}]
```

Using 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 (#).

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

To add a comment after a command, end the command using a semicolon, then precede the comment with a pound sign (#).

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

Generating SDC Files

You can generate an SDC file in the following ways:

- Using the Synopsys Design Compiler, IC Compiler, Astro, Jupiter XT, 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 1-17.

Generating SDC Files From a Synopsys Tool

To generate an SDC file from Design Compiler, IC Compiler, Astro, Jupiter XT, or PrimeTime, use the `write_sdc` command.

```
write_sdc file_name
```

Note:

In Astro and Jupiter XT, you can also generate an SDC file by using the `ataWriteTC` Scheme command.

The `write_sdc` command 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 (version 1.7 for version Z-2006.12 and later releases of PrimeTime and version Z-2007.03 and later releases of Design Compiler, IC Compiler, Astro, and Jupiter XT). To generate the file with an earlier SDC version, use the `-version` option when you invoke the `write_sdc` command.

Note:

Astro and Jupiter XT do not support the `-version` option; these tools always generate the SDC file using version 1.7 syntax.

When you generate an SDC file using syntax version 1.7, 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 1-8 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 details about the expanded format, see “About the Generated SDC File” on page 1-13.

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. See “Managing Large SDC Files” on page 1-21 for tips on how to use these large files.

Note:

The commands generated by the `write_sdc` command might differ between 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 will not be 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 1-8 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 following command,

    ```
    create_clock -period 10 CLK
    ```

    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 by using 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 corresponding SDC command is (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, you can use the `read_sdc -syntax_only` command. The `read_sdc -syntax_only` 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.

Note:

Astro and Jupiter XT do not support `read_sdc -syntax_only`. To validate an SDC file in Astro or Jupiter XT, check for errors and warnings when reading the SDC file, and fix reported problems.

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

Reading SDC Files Into a Synopsys Tool

To read an SDC file into Design Compiler, IC Compiler, Astro, Jupiter XT, or PrimeTime, use the `read_sdc` command.

```
read_sdc file_name
```

Note:

In Astro and Jupiter XT, you can also read an SDC file by using the `ataLoadSDC` Scheme command.

For information about SDC file requirements, see "About the SDC Format" on page 1-3.

Important:

SDC files generated with PrimeTime version Y-2006.12 are not compatible with IC Compiler version Z-2007.03. To read the SDC file into IC Compiler, you must regenerate the SDC file with PrimeTime version Y-2006.12-SP1.

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

   Note:

   This option is not supported by Astro or Jupiter XT.

2. The `sdc_version` variable specified in the SDC file

   Note:

   This variable is not supported by Astro or Jupiter XT.

3. The default version

   The default SDC version is the latest available syntax (version 1.7 for version Z-2006.12 and later releases of PrimeTime and version Z-2007.03 and later releases of Design Compiler, IC Compiler, Astro, and Jupiter XT).

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. When using the `ataLoadSDC` command in Astro or Jupiter XT, the hierarchy separator character specified in the ataLoadSDC dialog box.
4. The SDC default hierarchy separator (/)

Managing Large SDC Files

Because the constraints are written in expanded form, the SDC file can become quite 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.

To write an SDC file directly to a compressed file in PrimeTime, use the `-compress gzip` option on the `write_sdc` command.

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

To write an SDC file directly to a compressed file in Design Compiler, IC Compiler, Astro, or Jupiter XT, define a Tcl procedure that pipes the `write_sdc` output to a program like gzip. Example 1-1 provides an example procedure.

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.

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

A

SDC Syntax

The following tables list the commands and arguments supported by SDC version 1.7. Commands and options added since version 1.5 are noted.

Note:

For information about individual tool support, see SolvNet article 015193.

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` (added in version 1.7)	[-capacitance *cap_units*] [-resistance *res_unit*] [-time *time_unit*] [-voltage *voltage_units*] [-current *current_unit*] [-power *power_unit*]

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` (added in version 1.7, supported only by `read_sdc`)	`[-no_hierarchy]`
	`[-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]`
	`[-hsc` *`separator`*`]`
	`[-regexp]`
	`[-nocase]`
	`-of_objects` *`objects`*
	`patterns`

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

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

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

`get_libs`	`[-regexp]`
	`[-nocase]`
	`patterns`

Table A-2 Object Access Commands (Continued)

Command	Supported arguments

`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`

Table A-3 Timing Constraints

Command	Supported arguments
create_clock	-period *period_value* [-name *clock_name*] [-waveform *edge_list*] [-add] [*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*] *source_objects* Option added in version 1.7: [-combinational]
group_path (added in version 1.7)	[-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*]
set_clock_gating_check	[-setup *setup_value*] [-hold *hold_value*] [-rise] [-fall] [-high] [-low] [*object_list*]

Table A-3 Timing Constraints (Continued)

Command	Supported arguments

`set_clock_groups`
(added in version 1.7)
	`[-name` *`name`*`]`
	`[-logically_exclusive]`
	`[-physically_exclusive]`
	`[-asynchronous]`
	`[-allow_paths]`
	`-group` *`clock_list`*

`set_clock_latency`	`[-rise]`
	`[-fall]`
	`[-min]`
	`[-max]`
	`[-source]`
	`[-late]`
	`[-early]`
	`[-clock` *`clock_list`*`]`
	`delay`
	`object_list`

`set_clock_sense`
(added in version 1.7)
	`[-positive]`
	`[-negative]`
	`[-pulse pulse]`
	`[-stop_propagation]`
	`[-clock` *`clock_list`*`]`
	`pin_list`

`set_clock_transition`	`[-rise]`
	`[-fall]`
	`[-min]`
	`[-max]`
	`transition`
	`clock_list`

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
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*] Options added in version 1.7: [-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*]
set_ideal_latency (added in version 1.7)	[-rise] [-fall] [-min] [-max] *delay* *object_list*
set_ideal_network (added in version 1.7)	[-no_propagate] *object_list*
set_ideal_transition (added in version 1.7)	[-rise] [-fall] [-min] [-max] *transition_time* *object_list*

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_input_delay	[-clock *clock_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*] *delay_value* Options added in version 1.7: [-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*]
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*] *delay_value* Options added in version 1.7: [-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*]
set_multicycle_path	[-setup] [-hold] [-rise] [-fall] [-start] [-end] [-from *from_list*] [-to *to_list*] [-through *through_list*] *path_multiplier* Options added in version 1.7: [-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*]

Table A-3 Timing Constraints (Continued)

Command	Supported arguments
set_output_delay	[-clock *clock_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*

Table A-4 Environment Commands

Command	Supported arguments
set_case_analysis	*value* *port_or_pin_list*
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*
set_input_transition	[-rise] [-fall] [-min] [-max] [-clock *clock_name*] [-clock_fall] *transition* *port_list*

Table A-4 Environment Commands (Continued)

Command	Supported arguments
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*
set_min_capacitance	*value* *object_list*
set_operating_conditions	[-library *lib_name*] [-max *max_condition*] [-min *min_condition*] [-max_library *max_lib*] [-min_library *min_lib*] [-object_list *objects*] [*condition*]

Table A-4 Environment Commands (Continued)

Command	Supported arguments
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] *derate_value* [*object_list*]
set_wire_load_min_block_size	*size*
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*]

Table A-5 Multivoltage and Power Optimization Commands

Command	Supported arguments

`create_voltage_area`
(added in version 1.6)
	`-name` *`name`*
	`[-coordinate`
	`coordinate_list``]`
	`[-guard_band_x` *`float`*`]`
	`[-guard_band_y` *`float`*`]`
	`cell_list`

`set_level_shifter_strategy` (added in version 1.6)	`[-rule` *`rule_type`*`]`
`set_level_shifter_threshold` (added in version 1.6)	`[-voltage` *`float`*`]` `[-percent` *`float`*`]`
`set_max_dynamic_power`	*`power`* `[`*`unit`*`]`
`set_max_leakage_power`	*`power`* `[`*`unit`*`]`