SmartFusion2, IGLOO2, RTG4 Tcl Commands Reference Guide

Libero SoC v12.2

NOTE: PDF files are intended to be viewed on the printed page; links and cross-references in this PDF file may point to external files and generate an error when clicked. **View the online help included with software to enable all linked content.**

a MICROCHIP company

About Microsemi

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at **www.microsemi.com.**

5-02-00529-4/08.19

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Fax: +1 (949) 215-4996
Email:
sales.support@microsemi.com
www.microsemi.com

Table of Contents

Introduction to Tcl Scripting

Tcl, the Tool Command Language, pronounced *tickle*, is an easy-to-learn scripting language that is compatible with Libero SoC software. You can run scripts from either the Windows or Linux command line or store and run a series of commands in a *.tcl batch file.

This section provides a quick overview of the main features of Tcl:

- Basic syntax
- Types of Tcl commands
- Variables
- Command substitution
- Quotes and braces
- Lists and arrays
- Control structures
- Print statement and Return values
- Running Tcl scripts from the command line
- Exporting Tcl scripts
- Project Manager Tcl Commands

For complete information on Tcl scripting, refer to one of the books available on this subject. You can also find information about Tcl at web sites such as http://www.tcl.tk.

Libero SoC provides additional capabilities and built-in Tcl Commands:

- Running Tcl scripts from the command line
- Exporting Tcl scripts
- extended_run_lib
- Tcl Commands as specified in this document

Tcl Commands and Supported Families

When we specify a family name, we refer to the device family and all its derivatives, unless otherwise specified.

See Supported Families in the Tcl command help topics for the families supported for a specific Tcl command.

Tcl Command Documentation Conventions

The following table shows the typographical conventions used for the Tcl command syntax.

Syntax Notation	Description
`command -argument`	Commands and arguments appear in Courier New typeface.
`variable`	*Variables appear in blue, italic Courier New typeface. You must substitute an appropriate value for the variable.*
[`-argument`*`value`*] [*`variable`*]+	Optional arguments begin and end with a square bracket with one exception: if the square bracket is followed by a plus sign (+), then users must specify at least one argument. The plus

Syntax Notation	Description
	sign (+) indicates that items within the square brackets can be repeated. Do not enter the plus sign character.

Note: All Tcl commands are case sensitive. However, their arguments are not.

Examples

Syntax for the get_clocks command followed by a sample command:

```
get_clocks variable
```

```
get_clocks clk1
```

Syntax for the backannotate command followed by a sample command:

```
backannotate -name file_name -format format_type -language language -dir directory_name [-
netlist] [-pin]
```

```
backannotate -dir \
 {..\design} -name "fanouttest_ba.sdf" -format "SDF" -language "VERILOG" \
-netlist
```

Wildcard Characters

You can use the following wildcard characters in names used in Tcl commands:

Wildcard	What it Does
\	Interprets the next character literally
?	Matches any single character
*	Matches any string
[]	Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range)

Note: The matching function requires that you add a slash (\) before each slash in the port, instance, or net name when using wildcards in a PDC command. For example, if you have an instance named “A/B12” in the netlist, and you enter that name as “A\\/B*” in a PDC command, you will not be able to find it. In this case, you must specify the name as A\\\\/B*.

Special Characters [ ], { }, and \

Sometimes square brackets ([ ]) are part of the command syntax. In these cases, you must either enclose the open and closed square brackets characters with curly brackets ({ }) or precede the open and closed square brackets ([ ]) characters with a backslash (\). If you do not, you will get an error message.

For example:

```
pin_assign -port {LFSR_OUT[0]} -pin 15
or
pin_assign -port LFSR_OUT\[0\] -pin 180
```

Note: Tcl commands are case sensitive. However, their arguments are not.

Entering Arguments on Separate Lines

To enter an argument on a separate line, you must enter a backslash (\) character at the end of the preceding line of the command as shown in the following example:

```
backannotate -dir \
{..\design} -name "fanouttest_ba.sdf" -format "SDF" -language "VERILOG" \
-netlist
```

See Also

Introduction to Tcl scripting

Basic syntax

Project Manager Tcl Command Reference

A Tcl (Tool Command Language) file contains scripts for simple or complex tasks. You can run scripts from either the Windows or UNIX command line or store and run a series of Tcl commands in a *.tcl batch file. You can also run scripts from within the GUI in Project Manager.

Note: Tcl commands are case sensitive. However, their arguments are not.

The following table lists some of the Tcl scripting commands supported by the Libero SoC Project Manager:

Command	Action
add_file_to_library	Adds a file to a library in your project
add_library	Adds a VHDL library to your project
add_modelsim_path	Adds a ModelSim simulation library to your project
add_profile	Adds a profile; sets the same values as the Add or Edit Profile dialog box
associate_stimulus	Associates a stimulus file in your project
change_link_source	Changes the source of a linked file in your project
check_hdl	Checks the HDL in the specified file
close_project	Closes the current project in Libero SoC
configure_tool (SmartFusion2,IGLOO2, and RTG4)	Tcl command to set the parameters for any tool called by Libero for the SmartFusion2, IGLOO2, and RTG4 families.
create_links	Creates a link (or links) to a file/files in your project
delete_files	Deletes files from your Libero SoC project
download_core	Downloads a core and adds it to your repository
edit_profile	Edits a profile; sets the same values as the Add or Edit Profile dialog box
export_as_link	Exports a file to another directory and links to the file
export_ba_files	Exports the backannotated files.
export_bitstream_file	Configures parameters for your exported bitstream.

Command	Action
export_bsdl_file	Exports the BSDL to a specified file.
export_design_summary	Exports an HTML file containing information about your root SmartDesign in your project.
export_firmware	Exports design firmware configuration data.
export fp_pdc	Exports the Floorplanning Physical Design Constraint (*.pdc) File.
export_io_pdc	Exports the I/O constraints Physical Design Constraint (*.pdc) File.
export_netlist_file	Exports the netlist after the compile state has completed.
export_profiles	Exports your tool profiles; performs the same action as the Export Profiles dialog box
export_prog_job	Exports your programming job.
export_script	Explicitly exports the Tcl command equivalents of the current Libero session.
export_sdc_file	Exports the SDC (Synopsys Design Constraint) file for timing constraints.
import_component_data	Imports component data into an existing Libero project.
import_files (Libero SoC)	Imports files into your Libero SoC project
new_project	Creates a new project in the Libero SoC
open_project	Opens an existing Libero SoC project
organize_constraints	Organizes the constraint files in your project
organize_sources	Organizes the source files in your project
organize_tool_files	Specifies specific constraint files to be passed to and used by a Libero tool.
project_settings	Modifies project flow settings for your Libero SoC project
read_active_probe	Reads active probe values from the device.
refresh	Refreshes your project, updates the view and checks for updated links and files.
remove_core	Removes a core from your project
remove_library	Removes a VHDL library from your project
remove_profile	Deletes a tool profile

Command	Action
rename_library	Renames a VHDL library in your project
run_drc	Runs the design rule check on your netlist and generates an HDL file
run_simulation	Runs simulation on your project with your default simulation tool and creates a logfile
run_tool (SmartFusion2, IGLOO2, and RTG4)	Starts the specified tool.
save_log	Saves your Libero SoC log file
save_project	Saves your project
save_project_as	Saves your project with a different name
select_active_probe	Manages the current selection of active probe points to be used by active probe READ operations.
select_profile	Selects a profile to use in your project
set_actel_lib_options	Sets your simulation library to default, or to another library
set_device (Project Manager)	Sets your device family, die, and package in the Project Manager
set_live_probe	Channels A and/or B to the specified probe point(s).
set_modelsim_options	Sets your ModelSim simulation options
set_option	Sets your synthesis options on a module
set_userlib_options	Sets your user library options during simulation
set_root	Sets the module you specify as the root
synplify	Runs Synplify in batch mode and executes a Tcl script.
synplify_pro	Runs Synplify Pro in batch mode and executes a Tcl script.
unlink	Removes a link to a file in your project
use_file	Specifies which file in your project to use
use_source_file	Defines a module for your project
write_active_probe	Sets the target probe point on the device to the specified value.

Basic Syntax

Tcl scripts contain one or more commands separated by either new lines or semicolons. A Tcl command consists of the name of the command followed by one or more arguments. The format of a Tcl command is:

```
command arg1 ... argN
```

The command in the following example computes the sum of 2 plus 2 and returns the result, 4.

```
expr 2 + 2
```

The **expr** command handles its arguments as an arithmetic expression, computing and returning the result as a string. All Tcl commands return results. If a command has no result to return, it returns an empty string.

To continue a command on another line, enter a backslash (\) character at the end of the line. For example, the following Tcl command appears on two lines:

```
import -format "edif" -netlist_naming "Generic" -edif_flavor "GENERIC" {prepi.edn}
```

Comments must be preceded by a hash character (#). The comment delimiter (#) must be the first character on a line or the first character following a semicolon, which also indicates the start of a new line. To create a multi-line comment, you must put a hash character (#) at the beginning of each line.

Note: Be sure that the previous line does not end with a continuation character (\). Otherwise, the comment line following it will be ignored.

Special Characters

Square brackets ([ ]) are special characters in Tcl. To use square brackets in names such as port names, you must either enclose the entire port name in curly braces, for example, pin_assign -port {LFSR_OUT[15]} -iostd lvttl -slew High, or lead the square brackets with a slash (\) character as shown in the following example:

```
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High
```

Sample Tcl Script

```
#Create a new project and set up a new design
new_project -location {C:/sf2} -name {sf2} -project_description {} -block_mode 0 -
standalone_peripheral_initialization 0 \
-instantiate_in_smartdesign 1 -ondemand_build_dh 1 -hdl {VERILOG} -family {SmartFusion2}
-die {M2S010} -package {484 FBGA} -speed {STD} \
-die_voltage {1.2} -part_range {IND} -adv_options {IO_DEFT_STD:LVCMOS 2.5V} -adv_options
{RESTRICTPROBEPINS:1} -adv_options {RESTRICTSPIPINS:0} \
-adv_options {TEMPR:IND} -adv_options {UNUSED_MSS_IO_RESISTOR_PULL:None} -adv_options
{VCCI_1.2_VOLTR:COM} -adv_options {VCCI_1.5_VOLTR:COM} \
-adv_options {VCCI_1.8_VOLTR:COM} -adv_options {VCCI_2.5_VOLTR:COM} -adv_options
{VCCI_3.3_VOLTR:COM} -adv_options {VOLTR:IND}
#Import the component file
import_component -file {C:/g4_ccf_119/component/work/top/top.cxf}
# generate the component
generate_component -component_name {top} -recursive 1
# build design hierarchy
build_design_hierarchy
#set the top level design name
set_root -module {top::work}
#Import the pdc file
import_files -io_pdc {C:/g4_ccf_119/constraint/io/top.io.pdc}
# tun Synthesis
run_tool -name {SYNTHESIZE}
```

```
# organize the constraint file
organize_tool_files -tool {PLACEROUTE} -file {C:/sf2/constraint/io/top.io.pdc} -module
{top::work} -input_type {constraint}
#run the Place and Route tool
run_tool -name {PLACEROUTE}
# run verify timing
run_tool -name {VERIFYTIMING}
#export the bitstream file
export_bitstream_file -file_name {top} -export_dir {C:\sf2\designer\top\export} -format
{STP DAT}
#save the design
save_project
```

Types of Tcl commands

This section describes the following types of Tcl commands:

- Built-in commands
- Procedures created with the proc command

Built-in commands

Built-in commands are provided by the Tcl interpreter. They are available in all Tcl applications. Here are some examples of built-in Tcl commands:

- Tcl provides several commands for manipulating file names, reading and writing file attributes, copying files, deleting files, creating directories, and so on.
- exec - run an external program. Its return value is the output (on stdout) from the program, for example:

```
set tmp [ exec myprog ]
puts stdout $tmp
```

- You can easily create collections of values (lists) and manipulate them in a variety of ways.
- You can create arrays - structured values consisting of name-value pairs with arbitrary string values for the names and values.
- You can manipulate the time and date variables.
- You can write scripts that can wait for certain events to occur, such as an elapsed time or the availability of input data on a network socket.

Procedures created with the proc command

You use the proc command to declare a procedure. You can then use the name of the procedure as a Tcl command.

The following sample script consists of a single command named **proc**. The proc command takes three arguments:

- The name of a procedure (myproc)
- A list of argument names (arg1 arg2)
- The body of the procedure, which is a Tcl script

```
proc myproc { arg1 arg2 } {
# procedure body
}
myproc a b
```

Variables

With Tcl scripting, you can store a value in a variable for later use. You use the set command to assign variables. For example, the following set command creates a variable named x and sets its initial value to 10.

```
set x 10
```

A variable can be a letter, a digit, an underscore, or any combination of letters, digits, and underscore characters. All variable values are stored as strings.

In the Tcl language, you do not declare variables or their types. Any variable can hold any value. Use the dollar sign ($) to obtain the value of a variable, for example:

```
set a 1
set b $a
set cmd expr
set x 11
$cmd $x*$x
```

The dollar sign $ tells Tcl to handle the letters and digits following it as a variable name and to substitute the variable name with its value.

Global Variables

Variables can be declared global in scope using the Tcl global command. All procedures, including the declaration can access and modify global variables, for example:

```
global myvar
```

Command substitution

By using square brackets ([]), you can substitute the result of one command as an argument to a subsequent command, as shown in the following example:

```
set a 12
set b [expr $a*4]
```

Tcl handles everything between square brackets as a nested Tcl command. Tcl evaluates the nested command and substitutes its result in place of the bracketed text. In the example above, the argument that appears in square brackets in the second set command is equal to 48 (that is, 12* 4 = 48).

Conceptually,

```
set b [expr $a * 4]
```

expands to

```
set b [expr 12 * 4 ]
```

and then to

```
set b 48
```

Quotes and braces

The distinction between braces ({ }) and quotes (" ") is significant when the list contains references to variables. When references are enclosed in quotes, they are substituted with values. However, when references are enclosed in braces, they are not substituted with values.

Example

With Braces	With Double Quotes
set b 2	set b 2
set t { 1 $b 3 }	set t " 1 $b 3 "
set s { [expr $b + $b] }	set s " [expr $b + $b] "

With Braces	With Double Quotes
puts stdout $t	puts stdout $t
puts stdout $s	puts stdout $s

will output

```
1 $b 3                    vs.        1 2 3
[ expr $b + $b ]                     4
```

Filenames

In Tcl syntax, filenames should be enclosed in braces { } to avoid backslash substitution and white space separation. Backslashes are used to separate folder names in Windows-based filenames. The problem is that sequences of “\n” or “\t” are interpreted specially. Using the braces disables this special interpretation and specifies that the Tcl interpreter handle the enclosed string literally. Alternatively, double-backslash “\\n” and “\\t” would work as well as forward slash directory separators “/n” and “/t”.For example, to specify a file on your Windows PC at c:\newfiles\thisfile.adb, use one of the following:

```
{C:\newfiles\thisfile.adb}
C:\\newfiles\\thisfile.adb
"C:\\newfiles\\thisfile.adb"
C:/newfiles/thisfile.adb
"C:/newfiles/thisfile.adb"
```

If there is white space in the filename path, you must use either the braces or double-quotes. For example:

```
C:\program data\thisfile.adb
```

should be referenced in Tcl script as

```
{C:\program data\thisfile.adb} or "C:\\program data\\thisfile.adb"
```

If you are using variables, you cannot use braces { } because, by default, the braces turn off all special interpretation, including the dollar sign character. Instead, use either double-backslashes or forward slashes with double quotes. For example:

```
"$design_name.adb"
```

Note: To use a name with special characters such as square brackets [ ], you must put the entire name between curly braces { } or put a slash character \ immediately before each square bracket.

The following example shows a port name enclosed with curly braces:

```
pin_assign -port {LFSR_OUT[15]} -iostd lvttl -slew High
```

The next example shows each square bracket preceded by a slash:

```
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High
```

Lists and arrays

A list is a way to group data and handle the group as a single entity. To define a list, use curly braces { } and double quotes “ “. For example, the following set command {1 2 3 }, when followed by the list command, creates a list stored in the variable "a." This list will contain the items "1," "2," and "3."

```
set a { 1 2 3 }
```

Here's another example:

```
set e 2
set f 3
set a [ list b c d [ expr $e + $f ] ]
puts $a
```

displays (or outputs):

```
b c d 5
```

Tcl supports many other list-related commands such as lindex, linsert, llength, lrange, and lappend. For more information, refer to one of the books or web sites available on this subject.

Arrays

An array is another way to group data. Arrays are collections of items stored in variables. Each item has a unique address that you use to access it. You do not need to declare them nor specify their size.

Array elements are handled in the same way as other Tcl variables. You create them with the set command, and you can use the dollar sign ($) for their values.

```
set myarray(0) "Zero"
set myarray(1) "One"
set myarray(2) "Two"
for {set i 0} {$i < 3} {incr i 1} {
```

Output:

```
Zero
One
Two
```

In the example above, an array called "myarray" is created by the set statement that assigns a value to its first element. The for-loop statement prints out the value stored in each element of the array.

Special arguments (command-line parameters)

You can determine the name of the Tcl script file while executing the Tcl script by referring to the $argv0 variable.

```
puts “Executing file $argv0”
```

To access other arguments from the command line, you can use the `lindex` command and the *`argv`* variable:

To read the the Tcl file name:

```
lindex $argv 0
```

To read the first passed argument:

```
lindex $argv 1
```

Example

```
puts "Script name is $argv0" ; # accessing the scriptname
puts "first argument is [lindex $argv 0]"
puts "second argument is [lindex $argv 1]"
puts "third argument is [lindex $argv 2]"
puts "number of argument is [llength $argv]"
set des_name [lindex $argv 0]
puts "Design name is $des_name"
```

Control structures

Tcl control structures are commands that change the flow of execution through a script. These control structures include commands for conditional execution (if-then-elseif-else) and looping (while, for, catch).

An "if" statement only executes the body of the statement (enclosed between curly braces) if the Boolean condition is found to be true.

if/else statements

```
if { “$name” == “paul” } then {
…
# body if name is paul
} elseif { $code == 0 } then {
…
# body if name is not paul and if value of variable code is zero
```

```
} else {
…
# body if above conditions is not true
}
```

for loop statement

A "for" statement will repeatedly execute the body of the code as long as the index is within a specified limit.

```
for { set i 0 } { $i < 5 } { incr i } {
…
# body here
}
```

while loop statement

A "while" statement will repeatedly execute the body of the code (enclosed between the curly braces) as long as the Boolean condition is found to be true.

```
while { $p > 0 } {
…
}
```

catch statement

A "catch" statement suspends normal error handling on the enclosed Tcl command. If a variable name is also used, then the return value of the enclosed Tcl command is stored in the variable.

```
catch { open "$inputFile" r } myresult
```

Print statement and Return values

Print Statement

Use the puts command to write a string to an output channel. Predefined output channels are "stdout" and "stderr." If you do not specify a channel, then puts display text to the stdout channel.

Note: The STDIN Tcl command is not supported by Microsemi SoC tools.

Example:

```
set a [ myprog arg1 arg2 ]
puts "the answer from myprog was $a (this text is on stdout)"
puts stdout "this text also is on stdout"
```

Return Values

The return code of a Tcl command is a string. You can use a return value as an argument to another function by enclosing the command with square brackets [ ].

Example:

```
set a [ prog arg1 arg2 ]
exec $a
```

The Tcl command "exec" will run an external program. The return value of "exec" is the output (on stdout) from the program.

```
Example:
set tmp [ exec myprog ]
puts stdout $tmp
```

Running Tcl Scripts from the GUI

Instead of running scripts from the command line, you can use Execute Script dialog box to run a script in the software.

To run a Tcl script from the GUI:

1. In Libero SoC, from the **File** menu choose **Execute Script**.

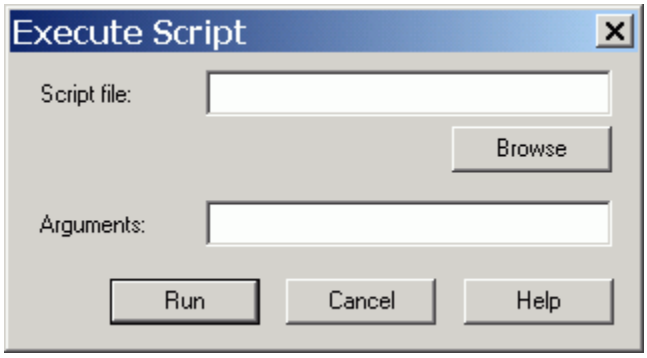


Figure 1 · Execute Script Dialog Box

2. Click **Browse** to display the **Open** dialog box, in which you can navigate to the folder containing the script file to open. When you click **Open**, the software enters the full path and script filename into the Execute Script dialog box for you.
3. In the Arguments edit box, enter the arguments to pass to your Tcl script as shown in the following sample Execute Script dialog box. Separate each argument by a space character. For information about accessing arguments passed to a Tcl script, see **"Running Scripts from the command line."**

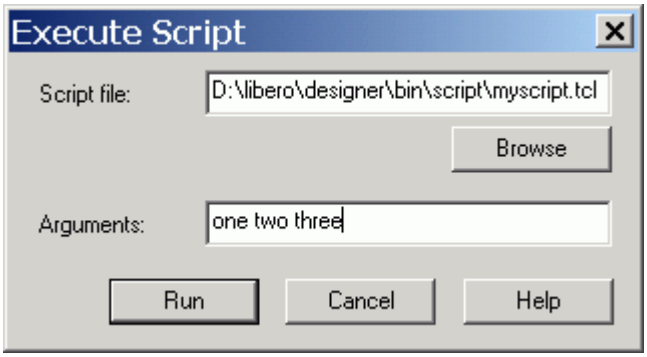


Figure 2 · Execute Script Dialog Box Example

4. Click **Run**.

Specify your arguments in the Execute Script dialog box. To get those argument values from your Tcl script, use the following:

```
puts "Script name: $argv0"
puts "Number of arguments: $argc"
set i 0
foreach arg $argv {
  puts "Arg $i : $arg"
  incr i
}
```

Running Tcl Scripts from the Command Line

You can run Tcl scripts from your Windows or Linux command line as well as pass arguments to scripts from the command line.

To execute a Tcl script file in the Libero SoC Project Manager software from a shell command line:

At the prompt, type the path to the Microsemi SoC software followed by the word "SCRIPT" and a colon, and then the name of the script file as follows:

```
<location of Microsemi SoC software>\bin\libero SCRIPT:<filename>
```

where <location of Microsemi SoC software> is the root directory in which you installed the Microsemi SoC software, and <filename> is the name, including a relative or full path, of the Tcl script file to execute. For example, to run the Tcl script file "myscript.tcl", type:

```
C:\libero\designer\bin\libero SCRIPT:myscript.tcl
```

If `myscript.tcl` is in a particular folder named "mydesign", you can use SCRIPT_DIR to change the current working directory before calling the script, as in the following example:

```
C:\libero\designer\bin\libero SCRIPT:myscript.tcl "SCRIPT_DIR:C:\actelprj\mydesign"
```

To pass arguments from the command line to your Tcl script file:

At the prompt, type the path to the Microsemi SoC software followed by the SCRIPT argument:

```
<location of Microsemi SoC software>\bin\designer SCRIPT:<filename "arg1 arg2 ...>" <-- For Libero
```

where `<location of Microsemi SoC software>` is the root directory in which you installed the Microsemi SoC software, and `<filename arg1 arg2 ...>`is the name, including a relative or full path, of the Tcl script file and arguments you are passing to the script file.

For example,

```
C:\libero\designer\bin\designer SCRIPT:myscript.tcl SCRIPT_ARGS:”one two three"
```

To obtain the output from the log file:

At the prompt, type the path to the Microsemi SoC software followed by the SCRIPT and LOGFILE arguments.

```
<location of Microsemi SoC software> SCRIPT:<filename> SCRIPT_ARGS:"a b c"
LOGFILE:<output.log>
```

where

- `location of Microsemi SoC software` is the root directory in which you installed the Microsemi SoC software
- `filename` is the name, including a relative or full path, of the Tcl script file
- `SCRIPT_ARGS` are the arguments you are passing to the script file
- `output.log` is the name of the log file

For example,

```
C:\libero\designer\bin\designer SCRIPT:testTCLparam.tcl SCRIPT_ARGS:"a b c"
LOGFILE:testTCLparam.log
```

Exporting Tcl Scripts

You can write out a Tcl script file that contains the commands executed in the current session. You can then use this exported Tcl script to re-execute the same commands interactively or in batch. You can also use this exported script to become more familiar with Tcl syntax.

You can export Tcl scripts from the Project Manager.

To export a Tcl session script from the Project Manager:

1. From the **File** menu, choose **Export Script File**. The **Export Script** dialog box appears.
2. Click **OK**. The **Script Export Options** dialog box appears:

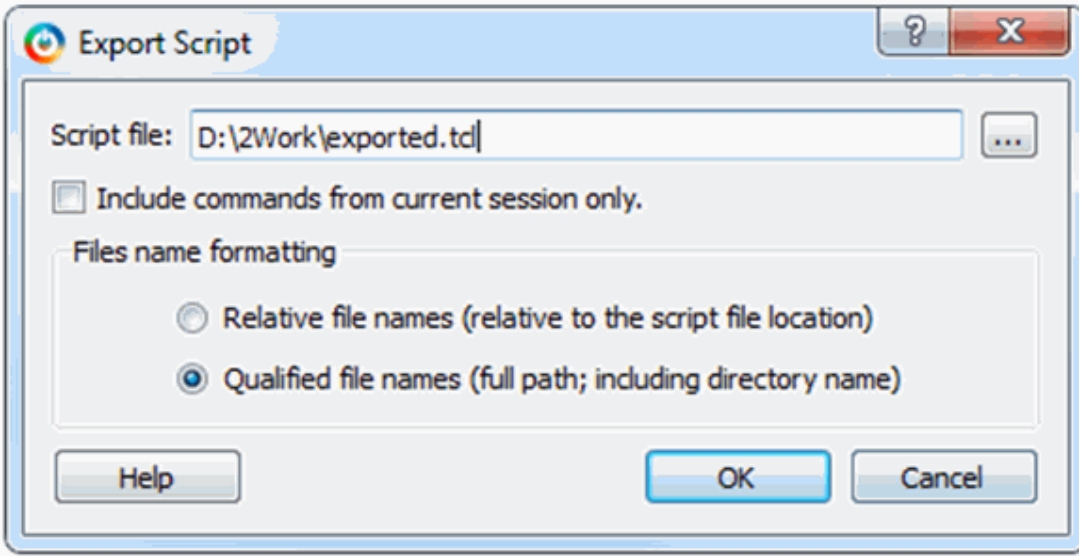


Figure 3 · Script Export Options

3. Check the **Include Commands from Current Design [Project] Only** checkbox. This option applies only if you opened more than one design or project in your current session. If so, and you do not check this box, Project Manager exports all commands from your current session.
4. Select the radio button for the appropriate filename formatting. To export filenames relative to the current working directory, select **Relative filenames (default)** formatting. To export filenames that include a fully specified path, select **Qualified filenames (full path; including directory name)** formatting.

   Choose **Relative filenames** if you do not intend to move the Tcl script from the saved location, or **Qualified filenames** if you plan to move the Tcl script to another directory or machine.
5. Click **OK**.

Project Manager saves the Tcl script with the specified filename.

Note: Notes:

- When exporting Tcl scripts, Project Manager always encloses filenames in curly braces to ensure portability.
- Libero SoC software does not write out any Tcl variables or flow-control statements to the exported Tcl file, even if you had executed the design commands using your own Tcl script. The exported Tcl file only contains the tool commands and their accompanying arguments.

extended_run_lib

Note: This is not a Tcl command; it is a shell script that can be run from the command line.

The extended_run_lib Tcl script enables you to run the multiple pass layout in batch mode from a command line.

```
$ACTEL_SW_DIR/bin/libero script:$ACTEL_SW_DIR/scripts/extended_run_lib.tcl
logfile:extended_run.log "script_args:-root path/designer/module_name [-n numPasses] [-
starting_seed_index numIndex] [-compare_criteria value] [-c clockName] [-analysis value] [-
slack_criteria value] [-stop_on_success] [-timing_driven|-standard] [-power_driven value]
[-placer_high_effort value]"
```

Note:

- There is no option to save the design files from all the passes. Only the (Timing or Power) result reports from all the passes are saved.

Arguments

-root *path/designer/module_name*

The path to the root module located under the designer directory of the Libero project.

[-n *numPasses*]

Sets the number of passes to run. The default number of passes is 5.

[-starting_seed_index *numIndex*]

Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Value may range from 1 to 100. If not specified, the default behavior is to continue from the last seed index that was used.

[-compare_criteria *value*]

Sets the criteria for comparing results between passes. The default value is set to frequency when the –c option is given or timing constraints are absent. Otherwise, the default value is set to violations.

Value	Description
frequency	Use clock frequency as criteria for comparing the results between passes. This option can be used in conjunction with the -c option (described below).
violations	Use timing violations as criteria for comparing the results between passes. This option can be used in conjunction with the -analysis, -slack_criteria and -stop_on_success options (described below).
power	Use total power as criteria for comparing the results between passes, where lowest total power is the goal.

[-c *clockName*]

Applies only when the clock frequency comparison criteria is used. Specifies the particular clock that is to be examined. If no clock is specified, then the slowest clock frequency in the design in a given pass is used. The clock name should match with one of the Clock Domains in the Summary section of the Timing report.

[-analysis *value*]

Applies only when the timing violations comparison criteria is used. Specifies the type of timing violations (the slack) to examine. The following table shows the acceptable values for this argument:

Value	Description
max	Examines timing violations (slack) obtained from maximum delay analysis. This is the default.
min	Examines timing violations (slack) obtained from minimum delay analysis.

[-slack_criteria *value*]

Applies only when the timing violations comparison criteria is used. Specifies how to evaluate the timing violations (slack). The type of timing violations (slack) is determined by the -analysis option. The following table shows the acceptable values for this argument:

Value	Description
worst	Sets the timing violations criteria to Worst slack. For each pass obtains the most amount of negative slack (or least amount of positive slack if all constraints are met) from the timing violations report. The largest value out of all passes will determine the best pass. This is the default.
tns	Sets the timing violations criteria to Total Negative Slack (tns). For each pass it obtains the sum of negative slack values from the first 100 paths from the timing violations report. The largest value out of all passes determines the best pass. If no negative slacks exist for a pass, then the worst slack is used to evaluate that pass.

[-stop_on_success]

Applies only when the timing violations comparison criteria is used. The type of timing violations (slack) is determined by the -analysis option. Stops running the remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).

[-timing_driven|-standard]

Sets layout mode to timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

[-power_driven *value*]

Enables or disables power-driven layout. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

Value	Description
off	Does not run power-driven layout.
on	Enables power-driven layout.

[-placer_high_effort *value*]

Sets placer effort level. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

Value	Description
off	Runs layout in regular effort.
on	Activates high effort layout mode.

Return

A non-zero value will be returned on error.

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

None

See Also

Place and Route - SmartFusion2, IGLOO2, RTG4

Multiple Pass Layout

Multiple Pass Layout - SmartFusion2, IGLOO2, RTG4

See the online help for more information.

Sample Tcl Script - Project Manager

The following Tcl commands create a new project and set your project options.

SmartFusion2, IGLOO2, RTG4:

```
#Create a new project and set up a new design
new_project -location {C:/sf2} -name {sf2} -project_description {} -block_mode 0 -
standalone_peripheral_initialization 0 \
```

```
-instantiate_in_smartdesign 1 -ondemand_build_dh 1 -hdl {VERILOG} -family {SmartFusion2}
-die {M2S010} -package {484 FBGA} -speed {STD} \
-die_voltage {1.2} -part_range {IND} -adv_options {IO_DEFT_STD:LVCMOS 2.5V} -adv_options
{RESTRICTPROBEPINS:1} -adv_options {RESTRICTSPIPINS:0} \
-adv_options {TEMPR:IND} -adv_options {UNUSED_MSS_IO_RESISTOR_PULL:None} -adv_options
{VCCI_1.2_VOLTR:COM} -adv_options {VCCI_1.5_VOLTR:COM} \
-adv_options {VCCI_1.8_VOLTR:COM} -adv_options {VCCI_2.5_VOLTR:COM} -adv_options
{VCCI_3.3_VOLTR:COM} -adv_options {VOLTR:IND}
#Import the component file
import_component -file {C:/g4_ccf_119/component/work/top/top.cxf}
# generate the component
generate_component -component_name {top} -recursive 1
# build design hierarchy
build_design_hierarchy
#set the top level design name
set_root -module {top::work}
#Import the pdc file
import_files -io_pdc {C:/g4_ccf_119/constraint/io/top.io.pdc}
# tun Synthesis
run_tool -name {SYNTHESIZE}
# organize the constraint file
organize_tool_files -tool {PLACEROUTE} -file {C:/sf2/constraint/io/top.io.pdc} -module
{top::work} -input_type {constraint}
#run the Place and Route tool
run_tool -name {PLACEROUTE}
# run verify timing
run_tool -name {VERIFYTIMING}
#export the bitstream file
export_bitstream_file -file_name {top} -export_dir {C:\sf2\designer\top\export} -format
{STP DAT}
#save the design
save_project
```

Tcl Flow in the Libero SoC

Use the following commands to manage and build your project in the Libero SoC.

Manage Profiles in the Project Manager

add_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"]

edit_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"] [-new_name *name*]

export_profiles -file *name* [-export "*predefined | user | all*"]

remove_profile -name *profile_name*

select_profile -name *profile_name*

Linking Files

change_link_source -file *filename* -path *pathname*

create_links [-hdl_source *file*]* [-stimulus *file*]* [-sdc *file*]* [-pin *file*]* [-dcf *file*]* [-gcf *file*]* [-pdc *file*]* [-crt *file*]* [-vcd *file*]*

export_as_link -file *filename* -path *link_path*

unlink -file *file* [-local *local_filename*]

Set Simulation Options in the Project Manager

add_modelsim_path -lib *library_name* [-path *library_path*] [-remove " "]

Set Device in the Project Manager

set_device [-family *family*] [-die *die*] [-package *package*]

Miscellaneous Operations in the Project Manager

project_settings [-hdl "*VHDL | VERILOG*"] [-auto_update_modelsim_ini "*TRUE | FALSE*"] [-auto_update_viewdraw_ini "*TRUE | FALSE*"] [-block_mode "*TRUE | FALSE*"] [-auto_generate_synth_hdl "*TRUE | FALSE*"] [-auto_run_drc "*TRUE | FALSE*"] [-auto_generate_viewdraw_hdl "*TRUE | FALSE*"] [-auto_file_detection "*TRUE | FALSE*"]

refresh

set_option [-synth "*TRUE | FALSE*"] [-module "*module_name*"]

remove_core -name *core_name*

Project Manager Tcl Commands

add_file_to_library

Tcl command; adds a file to a library in your project.

```
add_file_to_library
-library name
-file name
```

Arguments

`-library name`

Name of the library where you wish to add your file.

`-file name`

Specifies the new name of the file you wish to add (must be a full pathname).

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

Add a file named foo.vhd from the ./project/hdl directory to the library 'my_lib'

```
add_file_to_library -library my_lib -file ./project/hdl/foo.vhd
```

See Also

add_library

remove_library

rename_library

Project Manager Tcl Command Reference

add_library

Tcl command; adds a VHDL library to your project.

```
add_library
-library name
```

Arguments

`-library name`

Specifies the name of your new library.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

Create a new library called 'my_lib'.

```
add_library -library my_lib
```

See Also

remove_library

rename_library

Project Manager Tcl Command Reference

add_profile

Tcl command; sets the same values as the Add or Edit Profile dialog box. The newly added profile becomes the active tool profile for the specified *type* of tool.

```
add_profile -name profilename -type value -tool profiletool -location tool_location [-args tool_parameters] [-batch value]
```

Arguments

-name *profilename*

Specifies the name of your new profile.

-type *value*

Specifies your profile type, where value is one of the following:

Value	Description
synthesis	New profile for a synthesis tool
simulation	New profile for a simulation tool
stimulus	New profile for a stimulus tool
flashpro	New FlashPro tool profile

-tool *profiletool*

Name of the tool you are adding to the profile.

-location *tool_location*

Full pathname to the location of the tool you are adding to the profile.

-args *tool_parameters*

Profile parameters (if any).

-batch *value*

Runs the tool in batch mode (if TRUE). Possible values are:

Value	Description
TRUE	Runs the profile in batch mode
FALSE	Does not run the profile in batch mode

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

Create a new FlashPro tool profile called 'myflashpro' linked to a FlashPro installation in my c:\programs\actel\flashpro\bin directory

```
add_profile -name myflashpro -type flashpro -tool flashpro.exe -location
c:\programs\actel\flashpro\bin\flashpro.exe -batch FALSE
```

See Also

Project Manager Tcl Command Reference

add_modelsim_path

Tcl command; adds a ModelSim simulation library to your project.

```
add_modelsim_path -lib library_name [-path library_path] [-remove " "]
```

Arguments

`-lib library_name`

Name of the library you want to add.

`-path library_path`

Path to library that you want to add.

`-remove " "`

Name of library you want to remove (if any).

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

Add the ModelSim library 'msim_update2' located in the c:\modelsim\libraries directory and remove the library 'msim_update1':

```
add_modelsim_path -lib msim_update2 [-path c:\modelsim\libraries] [-remove msim_update1]
```

See Also

Project Manager Tcl Command Reference

associate_stimulus

Tcl command; associates a stimulus file in your project.

```
associate_stimulus
[-file name]*
[-mode value]
-module value
```

Arguments

`-file name`

Specifies the name of the file to which you want to associate your stimulus files.

`-mode value`

Specifies whether you are creating a new stimulus association, adding, or removing; possible values are:

Value	Description
new	Creates a new stimulus file association
add	Adds a stimulus file to an existing association
remove	Removes an stimulus file association

`-module value`

Sets the module, where value is the name of the module.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

The example associates a new stimulus file 'stim.vhd' for stimulus.

```
associate_stimulus -file stim.vhd -mode new -module stimulus
```

See Also

Project Manager Tcl Command Reference

change_link_source

Tcl command; changes the source of a linked file in your project.

```
change_link_source -file filename -path new_source_path
```

Arguments

`-file filename`

Name of the linked file you want to change.

`-path new_source_path`

Location of the file you want to link to.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

Change the link to a file 'sim1.vhd' in your project and link it to the file in c:\microsemi\link_source\simulation_test.vhd

```
change_link_source -file sim1.vhd -path c:\microsemi\link_source\simulation_test.vhd
```

See Also

Project Manager Tcl Command Reference

change_vault_location

Tcl command; changes the location of the vault.

Note: This command overrides the vault location for all projects.

```
change_vault_location \
-location location
```

Arguments

`-location location`

Specifies the new vault location. Value must be a file path.It is mandatory

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

```
change_vault_location -location {../vault}
```

See Also

Tcl Command Documentation Conventions

check_fdc_constraints

This Tcl command checks FDC constraints files associated with the Synthesis tool.

```
check_fdc_constraints -tool {synthesis}
```

Arguments

`-tool {synthesis}`

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
check_fdc_constraints -tool {synthesis}
```

Return Value

This command returns “0” on success and “1” on failure.

check_ndc_constraints

This Tcl command checks NDC constraints files associated with the Synthesis tool. NDC constraints are used to optimize the post-synthesis netlist with the Libero SoC Compile engine.

```
check_ndc_constraints -tool {synthesis}
```

Arguments

-tool {synthesis}

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
check_ndc_constraints -tool {synthesis}
```

See Also

set_ioff

check_pdc_constraints

This Tcl command checks PDC constraints files associated with the Libero Place and Route tool.

```
check_pdc_constraints -tool {designer}
```

Arguments

```
-tool {designer}
```

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
check_pdc_constraints -tool {designer}
```

Return Value

This command returns “0” on success and “1” on failure.

check_sdc_constraints

This Tcl command checks SDC constraints files associated with the Libero tools: designer, synthesis, or timing.

```
check_sdc_constraints -tool {tool_name}
```

Arguments

```
-tool {synthesis|designer|timing}
```

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

This command checks the SDC constraint files associated with Timing Verifcation.

```
check_sdc_constraints -tool {timing}
```

This command checks the SDC constraint files associated with Place and Route.

```
check_sdc_constraints -tool {designer}
```

This command checks the SDC constraint files associated with Synthesis.

```
check_sdc_constraints -tool {synthesis}
```

Return Value

The command returns “0” on success and “1” on failure.

configure_core

Tcl command; modifies the configuration of an existing core component in the SmartDesign. This command works for core components created for different types of cores namely, Sg cores, System Builder cores and Direct cores.

Limitations: The command does not work for SmartFusion2/IGLOO2 System Builder components, SmartFusion2 MSS component, and RTG4 PCIE_SERDES_IF_INIT(RTG4 High Speed Serial Interface 1 - EPCS and XAUI - with Initialization), NPSS_SERDES_IF_INIT(RTG4 High Speed Serial Interface 2 - EPCS and XAUI - with Initialization) and RTG4FDDRC_INIT(RTG4 DDR Memory Controller with initialization) core components.

```
configure_core \
-component_name component_name \
-params core_parameters
```

Arguments

`-component_name component_name`

Specifies the name of the component to be configured. It is mandatory.

`-params core_parameters`

Specifies the parameters needed to configure the core component. It is mandatory. This command will fail if none of the core parameters are specified.

Examples

```
configure_core -component_name {PF_CCC_C0} -params "GL1_0_IS_USED:false"
"GL0_0_IS_USED:true” “GL0_0_OUT_FREQ:200”}
configure_core -component_name {Core_UART} -params {"BAUD_VAL_FRCTN_EN:false"
"RX_FIFO:0" "RX_LEGACY_MODE:0" "TX_FIFO:1" "USE_SOFT_FIFO:1"}
```

See Also

Tcl Command Documentation Conventions

configure_tool

configure_tool is a general-purpose Tcl command that is used to set the parameters for any tool called by Libero **for the SmartFusion2, IGLOO2, RTG4 families**. The command requires the name of the tool and one or more

parameters in the format *tool_parameter*:*value*. These parameters are separated and passed to the tool to set up its run.

```
configure_tool
-name {<tool_name>} # Each tool_name has its own set of parameters
-params {<parameter>:<value>} # List of parameters and values
tool_name ::= COMPILE | CONFIGURE_PROG_OPTIONS | CONFIGURE_PROG_OPTIONS_RTG4 | SYNTHESIZE
| PLACEROUTE | GENERATEPROGRAMMINGDATA | GENERATEPROGRAMMINGFILE | PROGRAMDEVICE |
PROGRAM_OPTIONS | PROGRAMMER_INFO |IO_PROGRAM_STATE | SPM | FLASH_FREEZE |
PROGRAM_RECOVERY | USER_PROG_DATA | VERIFYTIMING | INIT_LOCK
```

Supported tool_names

The following table lists the supported tool_names.

tool_name	Parameter (-params)	Description
COMPILE	See the topic for parameter names and values.	See the topic for description.
"CONFIGURE_PROG_OPTIONS " on page **101**	See the topic for parameter names and values.	See the topic for description.
"CONFIGURE_PROG_OPTIONS_RTG4 (RTG4 only)" on page **102**	See the topic for parameter names and values.	See the topic for description.
SYNTHESIZE	See the topic for parameter names and values.	See the topic for description.
PLACEROUTE	See the topic for parameter names and values.	See the topic for description.
GENERATEPROGRAMMINGDATA	See the topic for parameter names and values.	See the topic for description.
	See the topic for parameter names and values.	See the topic for description.
"GENERATEPROGRAMMINGFILE " on page **104**	See the topic for parameter names and values.	See the topic for description.
PROGRAMDEVICE	See the topic for parameter names and values.	See the topic for description.
PROGRAM_OPTIONS	See the topic for parameter names and values.	See the topic for description.
PROGRAMMER_INFO	See the topic for parameter names and values.	See the topic for description.
IO_PROGRAMMING_STATE	See the topic for parameter names and values.	See the topic for description.
SPM	See the topic for parameter names and values.	See the topic for description.
FLASH_FREEZE	See the topic for parameter names and values.	See the topic for description.

tool_name	Parameter (-params)	Description
PROGRAM_RECOVERY	See the topic for parameter names and values.	See the topic for description.
USER_PROG_DATA	See the topic for parameter names and values.	See the topic for description.
VERIFYTIMING	See the topic for parameter names and values.	See the topic for description.
"INIT_LOCK" on page **104**	See the topic for parameter names and values.	See the topic for description.

See the SmartFusion2, IGLOO2, and RTG4 Tcl for SoC document for the full list of parameters and values.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
configure_tool -name {COMPILE} \
  -params { DISPLAY_FANOUT_LIMIT:10}\
  -params {MERGE_SDC:true}
configure_tool -name {SYNTHESIZE} -params {LANGUAGE_SYSTEM_VLOG:true}
configure_tool -name {PLACEROUTE} -params {PDPR:false} -params \
  {TDPR:true} -{EFFORT_LEVEL:false} -params {INCRPLACEANDROUTE:false}
```

For example, the command:

```
configure_tool \
  -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10} \
  -params {MERGE_SDC:true}
```

sets the COMPILE command options DISPLAY_FANOUT_LIMIT to 10 and MERGE_SDC to true.

There are alternative ways to write these commands to fit your coding style, as shown in the following examples.

Method 1 - single line

```
configure_tool -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10} -params {MERGE SDC:true}
```

Method 2 - one statement, multiple lines

```
configure_tool \
  -name {COMPILE} \
  -params {DISPLAY_FANOUT_LIMIT:10} \
  -params {MERGE_SDC:true}
```

Method 3 - multiple statements

```
configure_tool -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10}
configure_tool -name {COMPILE} -params {MERGE_SDC:true}
```

See Also

Project Manager Tcl Command Reference
Tcl documentation conventions

create_and_configure_core

Tcl command; creates a configured core component for a core selected from the Libero Catalog.

To use this command to create a configured core component with valid parameters and values, it is recommended to use the GUI to configure the core as desired. Then export the core configuration Tcl description by selecting the “Export Component Description(Tcl)” action on the right-click menu of the component in the Design Hierarchy. You can then use the exported Tcl command to create the configured core in a regular Tcl script.

```
create_and_configure_core \
-core_vlnv Vendor:Library:Name:version \
-component_name component_name \
[-params core_parameters]
```

Arguments

`-core_vlnv` *`Vendor:Library:Name:Version`*

Specifies the version identifier of the core being configured. It is mandatory.

`-component_name` *`component_name`*

Specifies the name of the configured core component. It is mandatory.

`-params` *`core_parameters`*

Specifies the parameters that need to be configured for the core component. It is optional. If the core parameters are not specified with this argument, the component is configured and generated with the core's default configuration. It is recommended to specify all the core parameters of interest as a part of this argument in this command.

Examples

```
create_and_configure_core -core_vlnv {Actel:SgCore:PF_CCC:1.0.115} -
component_name {PF_CCC_C3} -params { \
"PLL_IN_FREQ_0:25" \
"GL0_0_IS_USED:true" \
"GL0_0_OUT_FREQ:150" \
"GL0_1_IS_USED:true" \
"GL0_1_OUT_FREQ:50" }
```

Notes

For DirectCore and Solutions cores, refer to the core handbook or the core user guide for a list of valid parameters and values.

See Also

Tcl Command Documentation Conventions

create_set

Tcl command; creates a set of paths to be analyzed. Use the arguments to specify which paths to include. To create a set that is a subset of a clock domain, specify it with the `-clock` and `-type` arguments. To create a set that is a subset of an inter-clock domain set, specify it with the `-source_clock` and `-sink_clock` arguments. To create a set that is a subset (filter) of an existing named set, specify the set to be filtered with the `-parent_set` argument.

```
create_set\ -name <name>\ -parent_set <name>\ -type <set_type>\ -clock <clock name>\ -
source_clock <clock name>\ -sink_clock <clock name>\ -in_to_out\ -source <port/pin pattern>\
-sink <port/pin pattern>
```

Arguments

`-name <name>`

Specifies a unique name for the newly created path set.

`-parent_set <name>`

Specifies the name of the set to filter from.

`-clock <clock_name>`

Specifies that the set is to be a subset of the given clock domain. This argument is valid only if you also specify the `-type` argument.

`-type <value>`

Specifies the predefined set type on which to base the new path set. You can only use this argument with the `-clock` argument, not by itself.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_async	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
external_hold	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-in_to_out`

Specifies that the set is based on the “Input to Output” set, which includes paths that start at input ports and end at output ports.

`-source_clock <clock_name>`

Specifies that the set will be a subset of an inter-clock domain set with the given source clock. You can only use this option with the `-sink_clock` argument.

`-sink_clock <clock_name>`

Specifies that the set will be a subset of an inter-clock domain set with the given sink clock. You can only use this option with the `-source_clock` argument.

`-source <port/pin_pattern>`

Specifies a filter on the source pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

`-sink <port/pin_pattern>`

Specifies a filter on the sink pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

```
create_set -name { my_user_set } -source { C* } -sink { D* }
create_set -name { my_other_user_set } -parent_set { my_user_set } -source { CL* }
create_set -name { adder } -source { ALU_CLOCK } -type { REG_TO_REG } -sink { ADDER*}
create_set -name { another_set } -source_clock { EXTERN_CLOCK } -sink_clock {
MY_GEN_CLOCK }
```

create_smartdesign

Tcl command; creates a SmartDesign.

```
create_smartdesign \
-sd_name smartdesign_component_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component to be created. It is mandatory.

Examples

```
create_smartdesign -sd_name {top}
```

See Also

Tcl Command Documentation Conventions

delete_component

Tcl command; deletes a component from the Design Hierarchy.

```
delete_component \
-component_name component_name
```

Arguments

`-component_name component_name`

Specifies the name of the component to be deleted. It is mandatory.

Examples

```
delete_component -component_name {component}
delete_component -component_name {shifter}
```

See Also

Tcl Command Documentation Conventions

download_latest_cores

This Tcl command is used to download the latest cores into the vault. A project does not need to be open to run this command.

```
download_latest_cores
```

This command takes no arguments.

If there are no cores to be downloaded, you will see the following message:

```
Info:All the latest cores are present in the vault.
```

export_ba_files

Tcl command to export the backannotated files. The backannotated files are <design_name>_ba.v (Verilog backannotated netlist) or <design_name>_ba.vhd (VHDL backannotated netlist) and <design_name>_ba.sdf (Standard Delay Format) timing file. These files are passed to the default simulator for postlayout simulation.

```
export_ba_files
-export_dir {absolute path to folder location}
-export_file_name {name of file}
-vhdl {value}
-min_delay {value}
```

Arguments

-export_dir {*absolute path to directory/folder location*}

Folder/directory location.

-export_file_name {*name of file*}

File name to generate the files. If not specified, it takes <design_name> as the default.

-vhdl {*value*}

Generates the <design_name>_ba.v and <design_name>_ba.sdf when set to 0 and <design_name>_ba.vhd and <design_name>_ba.sdf when set to 1. Default is 0.

-min_delay {*value*}

Set to 1 to export enhanced min delays to include your best-case timing results in your Back Annotated file. Default is 0.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_ba_files\
  -export_dir {E:\designs\export\sd1}\
  -export_file_name {test}\
  -vhdl 0\
  -min_delay 1
```

export_bitstream_file

Configures the parameters for the bitstream to be exported from Libero.

Note: RTG4 devices do not support the security, SPI directory, or serialization options that SmartFusion2 and IGLOO2 devices support.

The syntax for the Export Bitstream File TCL command for SmartFusion2 and IGLOO2 is below:

```
 export_bitstream_file
[-file_name file]
[-export_dir dir]
[-format PPD | STP | CHAIN_STP | DAT | SPI | HEX]
```

```
[-for_ihp 0 | 1]
[-force_rtg4_otp 0 | 1]
[-master_file 0 | 1]
[-master_file_components SECURITY | FABRIC | ENVM]
[-encrypted_uek1_file 1 | 0]
[-encrypted_uek1_file_components FABRIC | ENVM]
[-encrypted_uek2_file 1 | 0]
[-encrypted_uek2_file_components FABRIC | ENVM]
[-encrypted_uek3_file 1 | 0]
[-encrypted_uek3_file_components FABRIC | ENVM]
[-trusted_facility_file 1 | 0]
[-trusted_facility_file_components FABRIC | ENVM]
[-add_golden_image 1 | 0]
[-golden_image_address golden_image_address]
[-golden_image_design_version golden_image_design_version]
[-add_update_image 1 | 0]
[-update_image_address update_image_address]
[-update_image_design_version update_image_design_version]
[-serialization_stapl_type SINGLE | MULTIPLE]
[-serialization_target_solution Flashpro_3_4_5 | generic_STAPL_player]
[-master_include_plaintext_passkey 0 | 1]
[-uek1_include_plaintext_passkey 0 | 1]
[-uek2_include_plaintext_passkey 0 | 1]
[-uek3_include_plaintext_passkey 0 | 1]
```

The syntax for the Export Bitstream File TCL command for RTG4 is below:

```
export_bitstream_file
[-file_name file]
[-export_dir dir]
[-format PPD | STP | CHAIN_STP | DAT | HEX]
[-for_ihp 0 | 1]
[-force_rtg4_otp 0 | 1]
```

Arguments

`-file_name file`

The name of the file. File name must start with design name. If omitted, design name will be used.

`-export_dir dir`

Location where the bitstream file will be exported. If omitted, design export folder will be used.

`-format PPD | STP | CHAIN_STP | DAT | SPI | HEX`

Specifies the bitstream file formats to be exported. Space is used as a delimiter. If omitted, PPD and DAT files will be exported.

`-for_ihp 0 | 1`

Specifies to export the bitstream files for Microsemi In House Programming(IHP).

`-force_rtg4_otp 0 | 1`

Enforces the use of One-time programming (OTP). It is optional.

Security-related options:

Note: One of the trusted_facility file or master_file or encrypted_uek1_file or encrypted_uek2_file **or encrypted_uek3_file** must be set to “1”. 1 indicates that this particular file type will be exported; 0 indicates that it will not be exported. For example, if `trusted_facility_file` is set to 1, all other file types must be set to 0.

Or, if `trusted_facility_file` is set to 0, a combination of `master_file` and `uek1_file`,`uek2_file` **and** `uek3_file` can be set to 1. In this case, `master_file` must be set to 1.

Bitstream encryption with default key (default security):

`-trusted_facility_file 1 | 0`

Specifies the bitstream file to be exported.

-trusted_facility_file_components *FABRIC | ENVM*

Specifies the components of the design that will be saved to the bitstream file. The value can **be any one or both of FABRIC and ENVM** .

Custom security options:

-master_file *0 | 1*

Specifies the bitstream files to be exported. Depends on the selected security.

Note: If -master_file is 1, SECURITY must be selected.

-master_file_components *SECURITY | FABRIC | ENVM*

Specifies the components in the design that will be saved to the bitstream file. The value can be **one or any combination of SECURITY, FABRIC, ENVM.**

Notes:

1. The SECURITY option is available in -bitstream_file_components only when file type is MASTER in –bitstream_file_type.

-encrypted_uek1_file *0 | 1*

-encrypted_uek1_file_components *FABRIC | ENVM*

Specifies the components of the design that will be saved to uek1 bitstream. **The value can be any one or both of FABRIC and ENVM .**

-encrypted_uek2_file *0 | 1*

-encrypted_uek2_file_components *FABRIC | ENVM*

Specifies the components of the design that will be saved to uek2 bitstream. **The value can be any one or both of FABRIC and ENVM .**

-encrypted_uek3_file *0 | 1*

-encrypted_uek3_file_components *FABRIC | ENVM*

Specifies the components of the design that will be saved to uek3 bitstream. The value can be any one or both of FABRIC and **ENVM** .

-master_include_plaintext_passkey *0 | 1*

Specifies that the master file includes plaintext passkey. This argument is optional.

-uek1_include_plaintext_passkey *0 | 1*

Specifies that uek1 includes plaintext passkey. This argument is optional.

-uek2_include_plaintext_passkey *0 | 1*

Specifies that uek2 includes plaintext passkey. This argument is optional.

-uek3_include_plaintext_passkey *0 | 1*

Specifies that uek3 includes plaintext passkey. This argument is optional.

Bitstream file to be exported and the components of the design that will be saved to the bitstream file are required.

SPI-related options:

-add_golden_image *1 | 0*

To enable/disable golden SPI image in SPI directory.

-golden_image_address

Hexadecimal address for golden image.

-golden_image_design_version

Decimal value for golden image design version.

-add_update_image *1 | 0*

To enable/disable update SPI image.

-update_image_address

Hexadecimal value for update image address.

-update_image_design_version

Decimal value for update image design version.

SPI-related options are optional.

Serialization options:

-serialization_stapl_type

Serialization stapl file type either single or multiple.

-serialization_target_solution

Target programming hardware – Flashpro_3_4_5 or generic_STAPL_player.

Serialization options are optional.

Note: A TCL script file exported from Libero will include all command options. You can modify options you need and remove options you do not need.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

Export a bitstream file:

Export bitstream file for design with default security

```
export_bitstream_file \
  -trusted_facility_file 1
  -trusted_facility_file_components {FABRIC}
```

Export bitstream file for design with custom security options

Export bitstreams to master and uek1 encrypted files. Master file to include security and fabric components, uek1 encrypted file to include FABRIC along with Export Pass Keys in Plaintext option.

```
export_bitstream_file \
  -file_name {top} \
  -export_dir
  {X:\10_docs_review\12.0_Release\102018\g4_clkint_fftousram\designer\top\export} \
  -format {PPD DAT HEX} \
  -for_ihp 1 \
  -master_file 1 \
  -master_file_components {SECURITY FABRIC} \
  -encrypted_uek1_file 1 \
  -encrypted_uek1_file_components {FABRIC} \
  -encrypted_uek2_file 0 \
  -encrypted_uek2_file_components {} \
  -trusted_facility_file 0 \
  -trusted_facility_file_components {} \
  -add_golden_image 0 \
  -golden_image_address {} \
  -add_update_image 0 \
  -golden_image_design_version {} \
  -update_image_address {} \
  -update_image_design_version {} \
  -serialization_stapl_type {SINGLE} \
  -serialization_target_solution {FLASHPRO_3_4_5} \
  -master_include_plaintext_passkey 1 \
  -uek1_include_plaintext_passkey 1 \
  -uek2_include_plaintext_passkey 0
```

Export SPI directory for programming recovery

```
export_bitstream_file \
  -add_golden_image 1 \
```

```
  -golden_image_address {1111} \
  -golden_image_design_version {1} \
  -add_update_image 1 \
  -update_image_address {1211} \
  -update_image_design_version {1} \
```

Export bitstream file for design with MSS/serialization clients

```
export_bitstream_file \
  -file_name {mss1} \
  -format {STP} \
  -trusted_facility_file 1 \
  -trusted_facility_file_components {FABRIC ENVM} \
  -serialization_stapl_type {SINGLE} \
  -serialization_target_solution {FLASHPRO_3_4_5}
```

Export bitstream file for RTG4

```
export_bitstream_file \
  -file_name {sd1} \
  -export_dir {D:\sd_prj\jade_rtg4\designer\sd1\export} \
  -format {STP DAT HEX PPD} \
  -for_ihp 1 \
  -force_rtg4_otp 0
```

export_bsdl_file

Tcl command to export the BSDL to a specified file. The exported file has a *.bsd file name extension.

```
export_bsdl_file
-file {absolute path and name of BSDL file}
```

Arguments

```
-file {absolute path and name of BSDL file}
```

Specifies the *.bsd file.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_bsdl_file\
  -file {E:/designs/export/sd1.bsd}
```

export_component_to_tcl

Tcl command; exports the Tcl command for the selected component. The components can be SmartDesign components, configured cores and HDL+ cores.

```
export_component_to_tcl \
-component component_name \
```

```
[-library library_name] \
[-package package_name] \
-file file_path
```

Arguments

`-component component_name`

Specifies the name of the component for which the Tcl command is exported. It is mandatory.

`-library library_name`

Specifies the name of the library the component belongs to. It is optional.

`-package package_name`

Specifies the name of the package the HDL+core belongs to. It is optional.

`-file file_path`

Specifies the path where you wish to export the Tcl file. It is mandatory.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_component_to_tcl -component {pattern_gen_checker} -library {work} -package {} -
file {./pattern_gen_checker.tcl}
```

export_firmware

This Tcl command exports design firmware configuration data, which consists of:

- Component configuration for MSS/HPMS, FDDR and SERDES blocks instantiating your design
- Compatible firmware drivers for your peripherals

It also creates a workspace and project specific to the IDE tool of your choice (SoftConsole, Keil or IAR).

To open your exported firmware projects you must invoke the third-party development tool (SoftConsole, Keil or IAR) outside Libero SoC.

If you make any changes to your design, you must re-export firmware.

```
export_firmware
-export_dir {D:\Designs\software_drivers}
-create_project {0|1}
-software_ide {SoftConsole|Keil|IAR}
```

Arguments

`-export_dir {absolute or relative path of the folder location}`

Specifies the path and name of folder for the exported firmware.

`-create_project {0|1}`

Generates the workspace and project for the specified IDE tool. Default is 0.

`-software_ide {IDE_toolname}`

Specifies one of three IDE tool name: SoftConsole | IAR | Keil.

If you use -create_project parameter and -software_ide paramter at the same time, Libero exports the workspace and project for that Software IDE tool to the *export_path*/{*SoftConsole|Keil|IAR*} folder.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
export_firmware \
  -export_dir {D:\Designs\software_drivers} \
  -create_project {1} \
  -software_ide {SoftConsole}
```

export_fp_pdc

Tcl command to export the Floorplanning Physical Design Constraint (*.pdc) File. The exported file has a *_fp.pdc file name extension.

```
export_fp_pdc
-file {absolute path and name of *_fp.pdc file}
-mode {PDC_PLACE | PDC_FULL_PLACEMENT}
```

Arguments

`-file {absolute path and name of *_fp.pdc file}`

Specifies the *_fp.pdc file.

`-mode {PDC_PLACE | PDC_FULL_PLACEMENT}`

Use PDC_PLACE to export user's floorplanning constraints, for example, fixed logic and regions.

Use PDC_FULL_PLACEMENT to export information about all of the physical design constraints (I/O constraints, I/O Banks, routing constraints, region constraints, global and local clocks).

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_fp_pdc\
  -file {E:/designs/export/sd1_fp.pdc}\
  -mode {PDC_FULL_PLACEMENT}
```

export_ibis_file

Tcl command to export the IBIS (Input/Output Buffer Information Specification) model report. The exported file has a *.ibs file name extension.

```
export_ibis_file
-file {absolute path and name of *.ibs file}
```

Arguments

-file {*absolute path and name of *.ibs file*}

Specifies the IBIS file to export.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_ibis_file\
  -file {E:/designs/export/sd1.ibs}
```

export_io_pdc

Tcl command to export the I/O constraints Physical Design Constraint (*.pdc) File. The exported file has a *_io.pdc file name extension.

```
export_io_pdc
-file {absolute path and name of *_io.pdc file}
```

Arguments

-file {*absolute path and name of *_io.pdc file*}

Specifies the *_io.pdc file.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_io_pdc\
  -file {E:/designs/export/sd1_io.pdc}
```

export_job_data

Tcl command; configures the parameters for the Job Manager Data Container file (JDC) to be exported from Libero and used by Job Manager.

```
export_job_data -file_name name -export_dir path -components  "SECURITY | FABRIC | ENVM"
```

Arguments

All parameters are optional. Default values are used if parameters are omitted.

`-file_name` *name*

Name of the file that will be saved. If omitted, it will be the design name.

`-export_dir` *path*

Location where the file will be saved. If omitted, it will be the Libero export folder.

`-components` *SECURITY* | *FABRIC* | *ENVM*

Specifies the components of the design that will be saved to the file. The value can be any one or a combination of SECURITY and FABRIC and ENVM if they are available in the design. If the parameter is omitted, all available components of the design will be saved.

Note:The SECURITY component must be selected if user security is initialized for the current Libero design.

Supported Families

SmartFusion2, IGLOO2

Example

```
export_job_data \
 -file_name {sd1} \
 -export_dir {D:\sd_prj\test3T\designer\sd1\export} \
 -components {FABRIC}
```

See Also

Programming Job Manager User Guide

SPPS User Guide

export_netlist_file

Tcl command to export the netlist after the compile state has completed. The netlist can be either Verilog or VHDL. Microsemi recommends exporting the netlist after the compile state has successfully completed.

```
export_netlist_file
-file {absolute path and filename for netlist}
-vhdl {value}
```

Arguments

`-file {`*absolute path and filename*`}`

Specifies the path and name of netlist file.

`-vhdl {`*value*`}`

Generates the netlist in VHDL (when set to 1) or Verilog (when set to 0). Default is 0 (Verilog netlist).

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_netlist_files\
```

```
-file {E:/designs/export/sd1/sd1.v}\
-vhdl 0
```

export_pin_reports

Tcl command to configure and export a pin report file to a specified folder/directory location.

```
export_pin_reports
-export_dir {absolute path to folder location}
-pin_report_by_name {value}
-pin_report_by_pkg_pin {value}
-bank_report {value}}
-io_report {value}
```

Arguments

`-export_dir {absolute or relative path to the folder for pin report file}`

Specifies the folder.

`-pin_report_by_name {value}`

Set to 1 to have the pin report sorted by pin name. Default is 1.

`- pin_report_by_pkg_pin {value}`

Set to 1 to have pin report sorted by package pin number, 0 to not sort by package pin number. Default is 1.

`- bank_report {value}`

Set to 1 to generate the I/O bank report, 0 to not generate the report. Default is 1.

`- io_report {value}`

Set to 1 to generate the I/O report, 0 to not generate the report. Default is 1.

At least one argument must be specified for this command.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
export_pin_reports\
  -export_dir {E:/designs/export}\
  -pin_report_by_name {1}\
  -pin_report_by_pkg_pin {0}\
  -bank_report {1}\
  -io_report {1}
```

export_prog_job

Tcl command; configures the parameters for the FlashPro Express programming job to be exported.

Note: The Programming Mode (JTAG/SPI-Slave) setting from the Programming Connectivity and Interface tool will be exported in the job file.

Note: RTG4 devices do not support the security options supported by SmartFusion2 and IGLOO2 devices.

The syntax for the export programming job TCL command for SmartFusion2 and IGLOO2 is shown below:

```
 export_prog_job
-job_file_name file
-export_dir dir
-bitstream_file_type TRUSTED_FACILITY | MASTER | UEK1 | UEK2
-bitstream_file_components SECURITY | FABRIC | ENVM
-include_plaintext_passkey 0 | 1
-design_bitstream_format PPD | STP
```

The syntax for the export programming job TCL command for RTG4 is below:

```
export_prog_job
-job_file_name file
-export_dir dir
-force_rtg4_otp 0 | 1
-design_bitstream_format PPD | STP
```

Arguments

`-job_file_name file`

The name of the file. Name must start with design name. If omitted, design name will be used.

`-export_dir dir`

Location where the job file will be saved; any folder can be specified. The default folder is the Libero export folder.

`-force_rtg4_otp 0 | 1`

Enforces the use of one-time programming (OTP). This argument is optional. The default value is 0.

`-bitstream_file_type TRUSTED_FACILITY | MASTER | UEK1 | UEK2`

Bitstream file to be included in the programming job. Only one bitstream file can be included in a programming job.

`-bitstream_file_components SECURITY | FABRIC | ENVM`

The list of components to be included in the programming job. Components should be delimited by space.

bitstream_file_components can be any one of SECURITY**, FABRIC, ENVM or any combination of them.**

Notes:

1. The SECURITY option is available in -bitstream_file_components only when file type is MASTER in –bitstream_file_type.

`-include_plaintext_passkey 0 | 1`

Includes plaintext passkey. This argument is optional.

`-design_bitstream_format PPD | STP`

Specifies the Bitstream file format. If omitted, the bitstream file will be in PPD format.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

SmartFusion2/IGLOO2

```
export_prog_job \
-job_file_name {top} \
```

```
        -export_dir
        {X:\10_docs_review\12.0_Release\102018\g4_clkint_fftousram\designer\top\export} \
        -bitstream_file_type {MASTER} \
        -bitstream_file_components {SECURITY FABRIC} \
        -include_plaintext_passkey 1 \
        -design_bitstream_format {PPD}
```

RTG4

```
        export_prog_job \
        -job_file_name {top} \
        -export_dir {X:\10_docs_review\12.0_Release\102018\rtg4_ff_usram\designer\top\export} \
        -force_rtg4_otp 1 \
    -design_bitstream_format {PPD}
```

export_sdc_file

Tcl command to export the SDC (Synopsys Design Constraint) file for timing constraints. The exported file has a *.sdc file name extension.

```
export_sdc_file
-file {absolute path and name of *.sdc file}
```

Arguments

-file {*absolute path and name of *.sdc file*}

Specifies the SDC file to export.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
export_sdc_file \
  -file {E:/designs/export/sd1.sdc}
```

generate_component

Tcl command; generates a SmartDesign or a core component.

```
generate_component \
-component_name component_name \
[-recursive 0|1]
```

Arguments

-component_name *component_name*

Specifies the name of the SmartDesign component or the core component to be generated. It is mandatory.

-recursive *0|1*

Specifies if a SmartDesign component needs to be generated recursively. It is optional. It is ‘0’ by default and generates only the specified component. If set to ‘1’, all the dependent components which are in

ungenerated state will be generated along with the SmartDesign component. It is recommended to generate all components individually.

Examples

The following command generates SmartDesign "sd2" only.

```
generate_component -component_name {sd2}
```

The following command generates SmartDesign "TOP" and all its dependent components which are in ungenerated state.

```
generate_component -component_name {TOP} -recursive 1
```

See Also

Tcl Command Documentation Conventions

generate_sdc_constraint_coverage

Tcl command to generate the constraint coverage report. The constraint coverage report contains information about the coverage of the paths from associated SDC constraints in the design. Two constraints coverage reports can be generated, one for Place and Route and one for Timing Verification.

To run this command, there is no need to run Place-and-Route first, but the design must be in the post-synthesis state. The generated constraint coverage reports (*.xml) are listed in the Reports tab and are physically located in <prj_folder>/designer/<module>/*constraints_coverage.xml.

```
generate_sdc_constraint_coverage -tool {PLACEROUTE | VERIFYTIMING}
```

Arguments

```
-tool {PLACEROUTE|VERIFYTIMING}
```

Specifies whether the constraint coverage report is based on the SDC constraint file associated with Place and Route or associated with Timing Verification.

Returns

Returns 0 on success, 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

This command generates the SDC Constraint Coverage report for the SDC file associated with Place and Route:

```
generate_sdc_constraint_coverage -tool {PLACEROUTE}
```

This command generates the SDC Constraint Coverage report for the SDC file associated with Timing Verification:

```
generate_sdc_constraint_coverage -tool {VERIFYTIMING}
```

See Also

Understanding Constraints Coverage Reports

get_libero_release

This Tcl command returns the release number of the Libero SoC release. The value that is returned is the same as the release number that is displayed in the Help > About Libero Window.

```
get_libero_release
```

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
get_libero_release
#save into a variable
set var1 [get_libero_release]
#display the variable
puts "Libero Release is $var1"
```

Output

You will see output similar to this:

```
Libero Release is v11.9
```

get_libero_version

This Tcl command returns the version number of the Libero SoC version. The value that is returned is the same as the version number that is displayed in the Help > About Libero Window.

```
get_libero_version
```

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
get_libero_version
#save into a variable
set var2 [get_libero_version]
#display variable
puts "Libero Version is $var2"
```

Output

You will see output similar to the following:

```
Libero Version is 11.9.0.4
```

import_component

This Tcl command imports a component *.cxf file into the Libero project. After import, the .cxf file is placed in the <project_folder>/component/work/<component_name> folder.

```
import_component -file <path_to_component.cxf>
```

Note: Only the *.cxf file format is supported for component import.

Arguments

-file <path_to_component *.cxf file>

The -file argument specifies the location of the component *.cxf file to import. Both absolute and relative paths are supported.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
import_component -file {D:/test/my_design/my_mult.cxf}
```

See Also

import_component_data

generate_component

import_component_data

A Libero SoC general-purpose Tcl command to import component data into an existing Libero project. Component refers to MDDR, FDDR and SERDES peripherals in SmartFusion2 devices. Component Data refers to initialization/configuration register values (*init_reg or *init.mem files) of those peripherals. Use this command if and when:

- The synthesized netlist or HDL files in the existing Libero SoC project contains no component (MDDR, FDDR and SERDES) information AND
- You want to add component s (MDDR, FDDR or SERDES) into the existing design.

```
import_component_data
-module root # name of the top_level (root)
-fddr file_path_and_name # has to be FDDR_init.reg or .mem
-mddr file_path_and_name # has to be MDDR_init.reg or .mem
-serdes0 file_path_and_name # has to be SERDESIF_0_init.reg or .mem
-serdes1 file_path_and_name # has to be SERDESIF_1_init.reg or .mem
-serdes2 file_path_and_name # has to be SERDESIF_2_init.reg or .mem
-serdes3 file_path_and_name # has to be SERDESIF_3_init.reg or .mem
-envm_cfg file_path_and_name # SmartFusion2, IGLOO2 only
-uprom_cfg file_path_and_name # RTG4 only
```

Note: The eNVM config file can have any name.

Note: Either *_init.reg (register configuration file) or *.mem files (memory files) can be used. The two cannot be mixed in the same import_component_data command.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

The component name for IGLOO2 devices may have different file extension (*.mem or *.reg), depending on the Libero SoC release version used to generate the components.

The following is an example of importing design components created with a Libero SoC pre-v11.4 release into an IGLOO2 project.

```
import_component_data \
  -module <root> \
  -fddr <file_path>/FDDR_init.mem \
  -mddr <file_path>/MDDR_init.mem \
  -serdes0 <file_path>/SERDESIF_0_init.mem \
  -serdes1 <file_path>/SERDESIF_1_init.mem \
  -serdes2 <file_path>/SERDESIF_2_init.mem \
  -serdes3 <file_path>/SERDESIF_3_init.mem \
  -envm_cfg <user_cfg_filepath>
```

The following is an example of importing design components created with Libero SoC v11.4 or subsequent releases into an IGLOO2 project. Note the *.reg file extension.

```
import_component_data
  -module <root> \
  -fddr <file_path>/FDDR_init.reg \
  -mddr <file_path>/MDDR_init.reg \
  -serdes0 <file_path>/SERDESIF_0_init.reg \
  -serdes1 <file_path>/SERDESIF_1_init.reg \
  -serdes2 <file_path>/SERDESIF_2_init.reg \
  -serdes3 <file_path>/SERDESIF_3_init.reg \
  -envm_cfg <user_cfg_file_path>
```

The following is an example of importing design components created with a Libero SoC pre-v11.4 release into a SmartFusion2 project.

```
import_component_data \
  -module <root> \
  -fddr <file_path>/FDDR_init.reg \
  -mddr <file_path>/MDDR_init.reg \
  -serdes0 <file_path>/SERDESIF_0_init.reg \
  -serdes1 <file_path>/SERDESIF_1_init.reg \
  -serdes2 <file_path>/SERDESIF_2_init.reg \
  -serdes3 <file_path>/SERDESIF_3_init.reg \
  -envm_cfg <user_cfg_file_path>
```

The following is an example of importing design components created with Libero SoC v11.4 or a subsequent release into a SmartFusion2 project.

```
import_component_data \
  -module <root> \
  -envm_cfg <user_cfg_file_path>
```

Return Value

Returns 0 on success and 1 on failure.

import_files (Libero SoC)

Tcl command; enables you to import design source files and constraint files.

For importing constraint files, import_files has retired the -pdc parameter for SmartFusion2 and IGLOO2. It has been replaced with two new parameters to match the new design flow. Physical Design Constraints (PDC) Tcl must now be divided between I/O attribute and pin information from all floorplanning and timing constraints. These

commands must now reside in and be imported as separate files. The new parameters specify the type of *.pdc file being imported.

Use of the -pdc parameter with Smartfusion2 or IGLOO2 families will cause an error. The path to the file can be absolute or relative but must be enclosed in curly braces { }.

Use the -convert_EDN_to_HDL parameter to convert the EDIF file to HDL and then import the converted HDL file.

Note: The EDIF File is not imported.

```
import_files
-schematic {file}
-symbol {file}
-smartgen_core {file}
-ccp {file}
-stimulus {file}
-hdl_source {file}
-io_pdc {<absolute or relative path to file>} # For PDC containing I/O attribute and pin info
-fp_pdc {<absolute or relative path to file>} # For PDC containing timing and placement info
-edif {file}
-sdc {file}
-pin {file}
-dcf {file}
-pdc {file}
-gcf {file}
-vcd {file}
-saif {file}
-crt {file}
-simulation {file}
-profiles {file}
-cxf {file}
-templates {file}
-ccz {file}
-wf_stimulus {file}
-modelsim_ini {file}
-library {file}
-convert_EDN_to_HDL {true | false}
```

Arguments

`-schematic {file}`

Specifies the schematics you wish to import into your IDE project. Type parameter must be repeated for each file.

`-symbol {file}`

Specifies the symbols you wish to import into your IDE project. Type parameter must be repeated for each file.

`-smartgen_core {file}`

Specifies the cores you wish to import into your project. Type parameter must be repeated for each file.

`-ccp {file}`

Specifies the ARM or Cortex-M1 cores you wish to import into your project. Type parameter must be repeated for each file.

`-stimulus {file}`

Specifies HDL stimulus files you wish to import into your project. Type parameter must be repeated for each file.

`-hdl_source {file}`

Specifies the HDL source files you wish to import into your project. Type parameter must be repeated for each file.

`-io_pdc {<absolute or relative path to file>}`

SmartFusion2 and IGLOO2 only - Specifies the PDC file that contains the I/O attribute and pin information.

`-fp_pdc {<absolute or relative path to file>}`

SmartFusion2 and IGLOO2 only - Specifies the PDC file that contains the timing and placement information.

`-edif {file}`

Specifies the EDIF files you wish to import into your project. Type parameter must be repeated for each file. This is a mandatory option if you want to convert EDIF to HDL with the –can_convert_EDN_to_HDL option.

`-convert_EDN_to_HDL {true |false |1 | 0} #Boolean {true | false | 1 | 0}`

The –edif option is mandatory. If the –edif option is not specified or the –convert_EDN_to_HDL is used with another option, EDIF to HDL conversion will fail.

`-constraint_sdc {file}`

Specifies the SDC constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_pin {file}`

Specifies the PIN constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_dcf {file}`

Specifies the DCF constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_pdc {file}`

Specifies the PDC constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_gcf {file}`

Specifies the GCF constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_vcd {file}`

Specifies the VCD constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_saif {file}`

Specifies the SAIF constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_crt {file}`

Specifies the CRT constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-simulation {file}`

Specifies the simulation files you wish to import into your Libero SoC project. Type parameter must be repeated for each file.

`-profiles {file}`

Specifies the profile files you wish to import into your Libero SoC project. Type parameter must be repeated for each file.

`-cxf {file}`

Specifies the CXF file (such as SmartDesign components) you wish to import into your Libero SoC project. Type parameter must be repeated for each file.

`-templates {file}`

Specifies the template file you wish to import into your project.

`-ccz {file}`

Specifies the IP core file you wish to import into your project.

`-wf_stimulus {file}`

Specifies the WaveFormer Pro stimulus file you wish to import into your project.

`-modelsim_ini {file}`

Specifies the ModelSIM INI file that you wish to import into your project.

`-library {file}`

Specifies the library file that you wish to import into your project. If a library file is not available it will be created and added to the library.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

The command below imports the HDL source files file1.vhd and file2.vhd:

```
import_files -hdl_source file1.vhd -hdl_source file2.vhd
```

See Also

Project Manager Tcl Command Reference

loopback_test

Tcl command; used to start and stop the loopback tests.

```
loopback_test [-deviceName device_name] -start -serdes num -lane num -type LoopbackType
loopback_test [-deviceName device_name] -stop -serdes num -lane num
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User's Guide for details).

`-start`

Starts the loopback test.

`-stop`

Stops the loopback test.

`-serdes num`

Serdes block number. Must be between 0 and 4 and varies between dies.

`-lane num`

Serdes lane number. Must be between 0 and 4

`-type LoopbackType`

Specifies the loopback test type. Must be *meso* (PCS Far End PMA RX to TX Loopback)

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
loopback_test -start -serdes 1 -lane 1 -type meso
loopback_test -start -serdes 0 -lane 0 -type plesio
loopback_test -start -serdes 1 -lane 2 -type parallel
loopback_test -stop -serdes 1 -lane 2
```

new_project

Tcl command; creates a new project in Libero SoC. If you do not specify a location, Libero SoC saves the new project in your current working directory.

```
new_project -name project_name\
-use_enhanced_constraint_flow {1 | 0} \
-location project_location -family family_name\
-project_description brief text description of project\
-die device_die -package package_name -hdl HDL_type\
-speed speed_grade -die_voltage value\

-adv_options value\
-standalone_peripheral_initialization {1 | 0}\
-block_mode {1 | 0}\
-instantiate_in_smartdesign {1 | 0}
```

Arguments

`-name project_name`

The name of the project. This is used as the base name for most of the files generated from Libero SoC.

`-use_enhanced_constraint_flow {1 | 0}`

Set to 1 to use the Enhanced Constraint Flow or 0 to use the Classic Constraint Flow. Libero SoC's Enhanced Constraint Flow provides a single centralized view for you to import, link, edit, check, and create design constraints and associate the constraints to different design tools in Libero. SoC.

`-location project_location`

The location of the project. Must not be an existing directory.

`-project_description project_description`

A brief text description of the design in your project.

`-family family_name`

The Microsemi SoC device family for your targeted design.

`-die device_die`

Die for your targeted design.

`-package package_name`

Package for your targeted design.

`-hdl HDL_type`

Sets the HDL type for your new project.

Value	Description
VHDL	Sets your new projects HDL type to VHDL
VERILOG	Sets your new projects to Verilog

`-speed speed_grade`

Sets the speed grade for your project. Possible values depend on your device, die and package. See your device datasheet for details.

`-die_voltage value`

Sets the die voltage for your project. Possible values depend on your device. See your device datasheet for details.

`-standalone_peripheral_initialization {1 | 0}` *(for SmartFusion2 and IGLOO2 only)*

Set this option to 1 if you want to build your own peripheral initialization logic in SmartDesign to initialize each of the peripherals (MDDR/FDDR/SERDES) independently. Set this option to 0 to instruct System Builder to build the initialization circuitry for MDDR/FDDR/SERDES peripherals.

`-block_mode {1 | 0}`

Enter "1" to enable or "0" (default) to disable design block creation.

`-instantiate_in_smartdesign {1 | 0}`

Enter "1" to enable or "0" (default) to disable Instantiate SystemBuilder/MSS components in a Smart Design. When set to "1", a System Builder or MSS component is auto-instantiated in a SmartDesign component upon creation. The default is 1.

`-adv_options value`

Sets your advanced options, such as operating conditions.

Value	Description
IO_DEFT_STD:LVTTL	Sets your I/O default value to LVTTL. This value defines the default I/O technology to be used for any I/Os that the user does not explicitly set a technology for in the I/O Editor. It could be any of : • LVTTL • LVCMOS 3.3V • LVCMOS 2.5V • LVCMOS 1.8V • LVCMOS 1.5V • LVCMOS 1.2V
DSW_VCCA_VOLTAGE_RAMP_RATE	(SmartFusion2 and IGLOO2 only) This value defines the Maximum VDD and VPP power supply ramp rate . Power-up management circuitry is designed into every SmartFusion2 and IGLOO2 SoC FPGA. These circuits ensure easy transition from the powered-off state to powered-up state of the device. The SmartFusion2, IGLOO2 system controller is responsible for systematic power-on reset whenever the device is powered on or reset. All the I/Os are held in a high-impedance state by the system controller until all power supplies are at their required levels and the system controller has completed the reset sequence. The power-on reset circuitry in SmartFusion2 and IGLOO2 devices requires the VDD and VPP supplies to ramp monotonically from 0 V to the minimum recommended operating voltage within a predefined time. There is no sequencing requirement on VDD and VPP. Four ramp rate options are available during design generation: • 50 µs • 1 ms • 10 ms • 100 ms Each selection represents the maximum ramp rate to apply to VDD and VPP.
PLL_SUPPLY	(SmartFusion2, IGLOO2 only) This value sets the voltage for the power supply you plan to connect to all the PLLs in your design, such as MDDR, FDDR, SERDES and FCCC. Two Values are available: • 2.5 • 3.3

Value	Description
RESTRICTPROBEPINS	This value reserves your pins for probing if you intend to debug using SmartDebug. Two values are available: • 1 (Probe pins are reserved) • 0 (No probe pins are reserved)
RESTRICTSPIPINS:1	(RTG4 only) Sets to 1 to reserve pins for SPI functionality in Programming. This reserved SPI pin option is displayed in the Compile Report when the compile process completes.
SYSTEM_CONTROLLER_SUSPEND_MODE	(SmartFusion2, IGLOO2 only) Enables **SmartFusion2 and IGLOO2** designers to suspend operation of the System Controller. Enabling this bit instructs the System Controller to place itself in a reset state once the device is powered up. This effectively suspends all system services from being performed. For a list of system services, refer to the **SmartFusion2 or IGLOO2 System Controller** User Guide for your device on the Microsemi website. Two values are available: • 1 (System Controller Suspend Mode is enabled) • 0 (System Controller Suspend Mode is disabled
The following options are for Analysis Operating Conditions **(SmartFusion2, IGLOO2, and RTG4)** so that Timing and Power analysis can be performed at different operating conditions.	
TEMPR	Sets your default temperature range for operating condition analysis **; can be** • COM (Commercial) • MIL (Military) • IND (Industrial)
VCCI_1.2_VOLTR	Sets the Default I/O Voltage Range for 1.2V **which could be** • COM • IND • MIL • Custom These settings are propagated to Verify Timing, Verify Power and Backannotated Netlist to perform Timing/Power Analysis
VCCI_1.5_VOLTR	Sets the Default I/O Voltage Range for 1.5V **which could be** • COM • IND • MIL • Custom These settings are propagated to Verify Timing, Verify Power and Backannotated Netlist to perform Timing/Power Analysis
VCCI_1.8_VOLTR	Sets the Default I/O Voltage Range for 1.8V **which could be** • COM

Value	Description
	• IND • MIL • Custom These settings are propagated to Verify Timing, Verify Power and Backannotated Netlist to perform Timing/Power Analysis
VCCI_2.5_VOLTR	Sets the Default I/O Voltage Range for 2.5V **which could be** • COM • IND • MIL • Custom These settings are propagated to Verify Timing, Verify Power and Backannotated Netlist to perform Timing/Power Analysis
VCCI_3.3_VOLTR	Sets the Default I/O Voltage Range for 3.3V **which could be** These settings are propagated to Verify Timing, Verify Power and Backannotated Netlist to perform Timing/Power Analysis
VOLTR	Sets the core voltage range for operating condition analysis **; These settings are** propagated to Verify Timing, Verify Power and Backannotated Netlist to perform Timing/Power Analysis. **Can be one of the following:** • COM (Commercial) • MIL (Military) • IND (Industrial)
PART_RANGE	Sets your default temperature range for your project **; can be COM (Commercial), MIL (Military) or IND (Industrial).**

Supported Families

SmartFusion2, IGLOO2, RTG4

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Creates a new project in the directory ./designs/mydesign, with the HDL type Verilog for the SmartFusion2 family.

```
new_project -location {./designs/mydesign} -name {mydesign}
-use_enhanced_constraint_flow 1
-standalone_peripheral_initialization 1 -hdl {VERILOG} -family
{SmartFusion2} -die {M2S150TS} -package {FCS536} -speed {-1} -die_voltage {1.2}
-adv_options {DSW_VCCA_VOLTAGE_RAMP_RATE:100_MS} -adv_options
{IO_DEFT_STD:LVCMOS 2.5V} -adv_options {PLL_SUPPLY:PLL_SUPPLY_25} -adv_options
{RESTRICTPROBEPINS:1} -adv_options {SYSTEM_CONTROLLER_SUSPEND_MODE:0}
-adv_options {TEMPR:IND} -adv_options {VCCI_1.2_VOLTR:IND} -adv_options
{VCCI_1.5_VOLTR:IND} -adv_options {VCCI_1.8_VOLTR:IND} -adv_options
{VCCI_2.5_VOLTR:IND} -adv_options {VCCI_3.3_VOLTR:IND} -adv_options {VOLTR:IND}
```

See Also

Project Manager Tcl Command Reference

open_smartdesign

Tcl command; opens a SmartdDesign. You must either open or create a SmartDesign before using any of the SmartDesign specific commands "sd_*".

```
open_smartdesign \
-sd_name smartdesign_component_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component to be opened. It is mandatory.

Examples

```
open_smartdesign -sd_name {top}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

organize_tool_files

This Tcl command is used to specify specific constraint files to be passed to and used by a Libero tool.

```
organize_tool_files \
-tool {tool_name}
-params {tool parameters}
-file {<absolute or relative path to constraint file>} \
-module {$design::work} \
-input_type {value}
```

Arguments

`-tool {<tool_name>}`

Specifies the name of the tool files you want to organize. Valid values are:

`SYNTHESIZE | PLACEROUTE | SIM_PRESYNTH | SIM_POSTSYNTH | SIM_POSTLAYOUT | VERIFYTIMING`

`-file {<absolute or relative path to constraint file>}`

Specifies the absolute or relative path to the constraint file; there may be multiple `-file` arguments (see example below).

`-module {<design::work>}`

Module definition, format is <*$design:work*>.

`-input_type {<constraint>}`

Specifies type of input file. Possible values are: `constraint | source | simulation | stimulus | unknown`

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

The following command organizes the `test_derived.sdc` and `user.sdc` files of SDC file type for the tool VERIFYTIMING for the sd1: work design.

```
organize_tool_files \
   -tool {VERIFYTIMING} \
   -file {D:/Designs/my_proj/constraints/test_derived.sdc} \
   -file {D:/Designs/my_proj/constraints/user.sdc} \
   -module {sd1::work} \
   -input_type {constraint}
```

prbs_test

Tcl command; used in PRBS test to start, stop, reset the error counter and read the error counter value.

```
prbs_test [-deviceName device_name] -start -serdes num -lane num [-near] -pattern PatternType
prbs_test [-deviceName device_name] -stop -serdes num -lane num
prbs_test [-deviceName device_name] -reset_counter -serdes num -lane num
prbs_test [-deviceName device_name] -read_counter -serdes num -lane num
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User Guide for details).

`-start`

Starts the prbs test.

`-stop`

Stops the prbs test.

`-reset_counter`

Resets the prbs error count value to 0.

`-read_counter`

Reads and prints the error count value.

`-serdes num`

Serdes block number. Must be between 0 and 4 and varies between dies.

`-lane num`

Serdes lane number. Must be between 0 and 4.

`-near`

Corresponds to near-end (on-die) option for prbs test. Not specifying implies off-die.

`-pattern PatternType`

The pattern sequence to use for PRBS test. It can be one of the following:

prbs7, *prbs11*, *prbs23*, or *prbs31*

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
prbs_test -start -serdes 1 -lane 0 -near -pattern prbs11
prbs_test -start -serdes 2 -lane 2 -pattern custom -value all_zeros
prbs_test -start -serdes 0 -lane 1 -near -pattern user -value 0x0123456789ABCDEF0123
```

rename_file

This Tcl command renames a constraint file specified by the `-file` parameter to a different name specified by the `-target` parameter.

```
rename_file -file {filename} -target {new_filename}
```

Arguments

`-file {filename}`

Specifies the original name of the file.

`-target {new_filename}`

Specifies the new name of the file.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

This command renames the file `a.sdc` to `b.sdc`.

```
rename_file -file {c:/user/a.sdc} -target {c:/user/b.sdc}
```

Return Value

This command returns 0 on success and 1 on failure.

run_tool

run_tool starts the specified tool. For tools that support command files, an optional command file can be supplied through the -script parameter.

```
run_tool
-name {<tool_name >} \
-script {<absolute or relative path to script file>}
```

-script is an optional parameter.

```
tool_name ::= SYNTHESIZE | COMPILE | SIM_PRESYNTH | SIM_POSTSYNTH | PLACEROUTE |
VERIFYTIMING | VERIFYPOWER | GENERATEPROGRAMMINGFILE | GENERATE_MEMORY_MAP |
PROGRAMDEVICE | CONFIGURE_CHAIN | SMARTDEBUG | SSNANALYZER | UPDATE_ENVM | UPDATE_UPROM
|
```

Return

run_tool returns 0 on success and 1 on failure.

Supported tool_names

The following table lists tool_names for run_tool –name {*tool_name*}.

tool_name	Parameter	Description
SYNTHESIZE	-script {*script_file*}	Runs synthesis on your design.
COMPILE	N/A	Runs Compile with default or configured settings.
SIM_PRESYNTH	N/A	Runs pre-synthesis simulation with your default simulation tool
SIM_POSTSYNTH	N/A	Runs post-synthesis simulation with your default simulation tool.
PLACEROUTE	N/A	Runs Layout with default or configured settings.
VERIFYTIMING	-script {*script_file*}	Runs timing analysis with default settings/configured settings in *script_file*.
VERIFYPOWER	-script {*script_file*}	Runs power analysis with default settings/configured settings in *script_file*.
GENERATEPROGRAMMINGFILE	N/A	Generates the bitstream used for programming within Libero.
GENERATE_MEMORY_MAP	N/A	Exports an XML file in <prj_folder> component/work/<design> /<design>_DataSheet.xml. The file contains information about your root SmartDesign in your project.
PROGRAMDEVICE	N/A	Programs your device with configured parameters.
CONFIGURE_CHAIN	-script {*script_file*}	Takes a script that contains FlashPro-specific Tcl commands and passes them to FlashPro Express for execution.
SMARTDEBUG	-script {*script_file*}	Takes a script that contains SmartDebug-specific Tcl commands and passes them to SmartDebug for execution.
SSNANALYZER	-script {*script_file*}	Takes a script that contains Simultaneous Switching Noise (SSN)-specific Tcl commands and passes them to the SSN tool for execution. Simultaneous Switching Noise (SSN) is a Libero SoC tool that analyzes and generates a Nosie Margin report for I/Os after layout.
UPDATE_ENVM (SmartFusion2 and IGLOO2 only)	-script {*update_config_file*}	Takes a script file that updates the client(s) in the ENVM. In the script file, the client(s) to be updated may be a serialization client or a data storage client or a mix of serialization clients and data storage clients.

tool_name	Parameter	Description
UPDATE_UPROM (RTG4 Only)	-script {*update_config_file*}	Takes a script that updates the data storage client(s) in RTG4 UPROMs.

`-script {absolute or relative path to script file}`

Script file location.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
run_tool \
  -name {COMPILE}
run_tool \
  -name {SYNTHESIZE} -script {./control_synopsys.tcl}
  #control _synopsys.tcl contains the synthesis-specific Tcl commands
run_tool \
  -name {VERIFYTIMING} \
  -script {./SmartTime.tcl}
  # Script file contains SmartTime-specific Tcl commands
run_tool \
  -name {VERIFYPOWER} \
  -script {./SmartPower.tcl}
  # Script file contains SmartPower-specific Tcl commands
run_tool \
  -name {SMARTDEBUG}
  -script {./sd_test.tcl}
  # Script file contains SmartDebug-specific Tcl commands
run_tool \
  -name {SSNANALYZER}
  -script {<full_path>/ssn.tcl}
  # Script file contains the SSN-specific Tcl commands
```

Note

Where possible, the value of *tool_name* corresponds to the name of the tool in Libero SoC.

Invoking some tools will cause Libero SoC to automatically run some upstream tools in the design flow. For example, invoking Place and Route will invoke Synthesis (if not already run) before it runs Place and Route.

See Also

Project Manager Tcl Command Reference

save_smartdesign

Tcl command; saves all the changes made in a SmartDesign component.

```
save_smartdesign \
-sd_name smartdesign_component_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component to be saved. It is mandatory.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

```
save_smartdesign -sd_name {top}
```

See Also

Tcl Command Documentation Conventions

select_libero_design_device

This command selects the Libero design device for the Programming Connectivity and Interface tool within Libero. This command is needed when the tool cannot automatically resolve the Libero design device when there are two or more identical devices that match the Libero design device in the configured JTAG chain.

```
select_libero_design_device -name {device_name}
```

Arguments

`-name {device_name}`

Specifies a user-assigned unique device name in the JTAG chain.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

```
select_libero_design_device -name {M2S050TS (2)}
select_libero_design_device -name {my_design_device}
```

Note

This Tcl command is typically used in a Tcl command script file that is passed to the Libero run_tool command.

```
run_tool -name {CONFIGURE_CHAIN} -script {<flashPro_cmd>.tcl}
```

set_as_target

This Tcl command sets a SDC, PDC or FDC file as the target file to receive and store new constraints.

```
set_as_target -type {constraint_file_type} \
-file {constraint_file_path}
```

Arguments

`-type {sdc | pdc | fdc}`

Specifies the file type: SDC, PDC, or FDC.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

This command sets the SDC file <project_folder> /constraints/user.sdc as the target to receive and store new SDC commands.

```
set_as_target -type {sdc} -file {./constraint/user.sdc}
```

This command sets the PDC file <project_folder> /constraints/user.pdc as the target to receive and store new PDC commands.

```
set_as_target -type {pdc} -file {./constraint/user.pdc}
```

Return Value

This command returns 0 on success and 1 on failure.

set_live_probe

Tcl command; set_live_probe channels A and/or B to the specified probe point(s). At least one probe point must be specified. Only exact probe name is allowed (i.e. no search pattern that may return multiple points).

```
set_live_probe [-deviceName device_name] [-probeA probe_name] [-probeB probe_name]
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug **(see SmartDebug user guide for details)**.

`-probeA probe_ name`

Specifies target probe point for the probe channel A.

-probeB *probe_ name*

Specifies target probe point for the probe channel B.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

- The array must be programmed and active
- Active probe read or write operation will affect current settings of Live probe since they use same probe circuitry inside the device
- Setting only one Live probe channel affects the other one, so if both channels need to be set, they must be set from the same call to set_live_probe
- Security locks may disable this function
- In order to be available for Live probe, ProbeA and ProbeB I/O's must be reserved for Live probe respectively

Example

Sets the Live probe channel A to the probe point A12 on device **sf2**.

```
set_live_probe [-deviceName sf2] [-probeA A12]
```

unset_as_target

This Tcl command unsets a target file in the Constraints view.

```
unset_as_target -file {filename}
```

Arguments

`-file {filename}`

Specifies the name of the file to be unset as a target.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

This command unsets the PDC file <project_folder> /constraints/user.pdc:

```
unset_as_target -file {c:/user/a_io.pdc}
```

Return Value

This command returns 0 on success and 1 on failure.

SmartDesign Tcl Commands

The SmartDesign Tcl commands can be used to create a design in the SmartDesign. You must either create or open a SmartDesign before you can use any of the SmartDesign commands - sd_* .

All SmartDesign Tcl commands are supported by the **SmartFusion2, IGLOO2 and RTG4 familes.**

sd_add_pins_to_group

Tcl command; adds one or more pins to a pin group on an instance in a SmartDesign component.

```
sd_add_pins_to_group \
-sd_name smartdesign_component_name \
-instance_name instance_name \
-group_name group_name \
-pin_names pin_names
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the name of the instance on which the pin group is present. It is mandatory.

`-group_name` *`group_name`*

Specifies the name of the group to add the pins to. It is mandatory.

`-pin_names` *`pin_names`*

Specifies the list of instance pins to be added to the pin group. It is mandatory.

Examples

```
sd_add_pins_to_group -sd_name {TOP} -instance_name
{COREAXI4INTERCONNECT_C0_0} -group_name {Group} -pin_names {ARESETN ACLK}
```

See Also

Tcl Command Documentation Conventions

sd_clear_pin_attributes

Tcl command; clears all attributes on one or more pins/ports in a SmartDesign. Pin attributes include pin inversion, mark as unused and constant value settings.

```
sd_clear_pin_attributes \
-sd_name smartdesign_component_name \
-pin_names port_or_pin_names
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_names` *`port_or_pin_names`*

Specifies the name of the port/pin for which all attributes must be cleared. It is mandatory.

Examples

```
sd_clear_pin_attributes -sd_name {sd1} -pin_names {RAM1K18_0:A_DOUT_CLK}
sd_clear_pin_attributes -sd_name {top} -pin_names {CARRY_OUT}
```

Notes

This command will not work on multiple pins/ports in this release. Support for multiple pins/ports will be provided in the next Libero release. This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_configure_core_instance

Tcl command; configures the parameters of a core instance (Direct Instantiation) in a SmartDesign component. This command is typically used after instantiating a core from the catalog directly into a SmartDesign component (Direct Instantiation) without first creating a component for the core (using sd_instantiate_core). This command can configure multiple core parameters at a time.

```
sd_configure_core_instance \
-sd_name smartdesign_component_name \
-instance_name core_instance_name \
-params core_parameters \
[-validate_rules 0|1]
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name instance_name`

Specifies the name of the core instance in the SmartDesign which needs to be configured. It is mandatory.

`-params core_parameters`

Specifies the parameters that need to be configured for the core instance. It is mandatory.

`-validate_rules 0|1`

Validates the rules of the updated configuration. It is optional.

Examples

```
sd_configure_core_instance -sd_name {SD1} -instance_name {COREFIFO_0} -
params {"SYNC:0" "param2:value2" "param3:value3"} -validate_rules 0
```

See Also

Tcl Command Documentation Conventions

sd_connect_instance_pins_to_ports

Tcl command; connects all pins of an instance to new SmartDesign top level ports.

```
sd_connect_instance_pins_to_ports \
-sd_name smartdesign_component_name \
-instance_name instance_name
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the instance name for which all the pins must be connected to top level ports. It is mandatory. The instance pins are connected to new top level ports created with the same instance pin names. If a top level port with the same name already exists, then the tool automatically creates a new port with name <port_name>_<index> (index is an automatically generated integer starting at 0 such that the port name is unique in the SmartDesign).

Examples

```
sd_connect_instance_pins_to_ports -sd_name {top} -instance_name
{CORESPI_C0_0}
sd_connect_instance_pins_to_ports -sd_name {top} -instance_name
{ddr_out_0}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_connect_net_to_pins

Tcl command; connects a list of SmartDesign top level ports and/or instance pins to a net.

```
sd_connect_net_to_pins \
-sd_name smartdesign_component_name \
-net_name net_name \-pin_names port_or_pin_names
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-net_name` *`net_name`*

Specifies the name of the net to be connected to pins/ports in the SmartDesign component. It is mandatory.

`-pin_names` *`port_or_pin_names`*

Specifies the name of the ports/pins to be connected to the net in the SmartDesign. It is mandatory. The command will fail if:

- The ports/pins do not exist.
- The ports/pins and the net being connected are of different range/size.
- There is more than one port/pin driving the net.

Examples

```
sd_connect_net_to_pins -sd_name {shifter} -net_name {ready_net} -pin_names {"READY"}
sd_connect_net_to_pins -sd_name {top} -net_name {clk_net} -pin_names {CLK
RAM64x12_0:R_CLK RAM64x12_0:W_CLK}
```

Notes

This command is not required to build a SmartDesign component. It is not exported when you select Libero **Project** - `**Export Script File**' or '**Export Component Description(Tcl)**' on a SmartDesign component. This command is typically used in conjunction with 'sd_create_*_net' command to connect two or more ports/pins to a net.

See Also

Tcl Command Documentation Conventions

sd_connect_pins_to_constant

Tcl command; connects SmartDesign top level output ports or input instance pins to constant values.

```
sd_connect_pins_to_constant \
-sd_name smartdesign_component_name \
-pin_names port_or_pin_names \
-value constant_value
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_names` *`port_or_pin_names`*

Specifies the names of the top level output ports or the instance level input pins to be tied to constant values. It is mandatory. Bus pins/ports and pin/port slices can also be tied to constant values. This command will fail if the specified port/pin does not exist. The command will also fail if the assigned object is a port of direction IN/INOUT or a pin of direction OUT/INOUT.

`-value` *`constant_value`*

Specifies the constant value to be assigned to the port/pin. It is mandatory. The acceptable values to this argument are GND/VCC/hexadecimal numbers.

Examples

```
sd_connect_pins_to_constant -sd_name {top} -pin_name {bypass} -value
{GND}
sd_connect_pins_to_constant -sd_name {top} -pin_name {sle_0:en} -value
{VCC}
sd_connect_pins_to_constant -sd_name {top} -pin_name {ram64x12_0:w_data}
-value {0x7f}
```

Notes

This command will not work on multiple pins/ports in this release. Support for multiple pins/ports will be provided in the next Libero release.

See Also

Tcl Command Documentation Conventions

sd_connect_pin_to_port

Tcl command; connects a SmartDesign instance pin to a new top level port. This command is equivalent to the 'Promote to Top Level` GUI action on an instance pin.

```
sd_connect_pin_to_port \
-sd_name smartdesign_component_name \
```

```
-pin_name pin_name \
[-port_name port_name]
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_name pin_name`

Specifies the name of the instance level pin that needs to be connected to a top level port. It is mandatory.

`-port_name port_name`

Specifies the name of the new top level port that the instance pin will be connected to. It is optional. If the port name is not specified, the new port takes the name of the instance pin. If the port name as defined by these rules already exists, the tool automatically creates a new port with name <port_name>_<index> (index is an automatically generated integer starting at 0 such that the port name is unique in the SmartDesign).

Examples

```
sd_connect_pin_to_port -sd_name {top} -pin_name {DFN1_0:D}
sd_connect_pin_to_port -sd_name {top} -pin_name {DFN1_0:Q} -port_name
{Q_OUT}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_connect_pins

Tcl command; connects a list of SmartDesign top level ports and/or instance pins together.

```
sd_connect_pins \
-sd_name smartdesign_component_name \
-pin_names port_or_pin_or_slice_names_
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_names port_or_pin_or_slice_names`

Specifies the port names, pin names and/or slice names to be connected together. It is mandatory. This command will fail if the ports, pins or slices do not exist. This command will also fail if the ports, pins and/or slices are not of the same size/range.

Examples

```
sd_connect_pins -sd_name {top} -pin_names {CLK MACC_PA_0:CLK DFN1_0:CLK}
sd_connect_pins -sd_name {top} -pin_names {MACC_PA_0:A
RAM1K20_0:A_DIN[17:0]}
```

See Also

Tcl Command Documentation Conventions

sd_create_bif_net

Tcl command; creates a bus interface (BIF) net in a SmartDesign component. Any net created must be connected to two or more ports/pins using the command “sd_connect_net_to_pins”.

```
sd_create_bif_net \
-sd_name smartdesign_component_name \
-net_name net_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-net_name net_name`

Specifies the name of the net to be added in the SmartDesign component. It is mandatory. The command will fail if there is an already existing net with the same name.

Examples

```
sd_create_bif_net -sd_name {TOP} -net_name {bifnet1}
```

Note: This new bif net is visible in the UI only when it is connected to two or more ports/pins using the command “sd_connect_net_to_pins” as shown below.

```
sd_connect_net_to_pins -sd_name {TOP} -net_name {bifnet1} -pin_names {"AHBmmaster0"
"CoreAHBLite_C0_0:AHBmmaster0"}
```

Notes

This command is not required to build a SmartDesign component. It is not exported when you select Libero **Project** - `**Export Script File**’ or ‘**Export Component Description(Tcl)**’ on a SmartDesign component. This command is used to manually create a Tcl script and specify a new name to the net that connects two or more ports/pins.

See Also

Tcl Command Documentation Conventions

sd_create_bif_port

Tcl command; creates a SmartDesign Bus Interface port of a given type. This command is used to create top level Bus Interface ports in a SmartDesign component to connect to the instance level Bus Interface ports of the same type.

To use this command, it is recommended to first use the GUI to instantiate the core component or the HDL module with Bus Interface port to be promoted in the SmartDesign. Then use the UI action "Promote to Top Level" on the Bus Interface port of interest and export the Tcl script for the SmartDesign component by selecting " Export Component Description(Tcl)" on the right-click menu of the SmartDesign component in the Design Hierarchy. You can then use the Tcl command ‘sd_create_bif_port’ from the exported Tcl script (note to change the SmartDesign name in the command) to create a bus interface port anywhere in a regular Libero script. Note that there can be different Bus Interface types and roles defined by the arguments –port_bif_vlnv and –port_bif_role.

```
sd_create_bif_port \
-sd_name smartdesign_component_name \
-port_name port_name \
-port_bif_vlnv vendor:library:name:version \
-port_bif_role port_bif_role \
-port_bif_mapping [bif_port_name:port_name]+
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-port_name` *`port_name`*

Specifies the name of the Bus Interface port to be added in the SmartDesign. It is mandatory.

`-port_bif_vlnv {vendor:library:name:version}`

Specifies the version identifier of the Bus Interface port to be added in the SmartDesign. It is mandatory.

`-port_bif_role {port_bif_role}`

Specifies the role of the Bus Interface port to be added in the SmartDesign. Role values depend on the type of Bus Interface (VLNV) that is being defined for the port. The figure below shows the roles for different Bus Interface ports supported by Libero.

Name	Vendor	Library	Role
AHB	AMBA	AMBA2	master
AHB	AMBA	AMBA2	slave
AHB	AMBA	AMBA2	mirroredMaster
AHB	AMBA	AMBA2	mirroredSlave
APB	AMBA	AMBA2	master
APB	AMBA	AMBA2	slave
APB	AMBA	AMBA2	mirroredMaster
APB	AMBA	AMBA2	mirroredSlave
AXI	AMBA	AMBA3	master
AXI	AMBA	AMBA3	slave
AXI	AMBA	AMBA3	mirroredMaster
AXI	AMBA	AMBA3	mirroredSlave
AXI	AMBA	AMBA3	system
AXI4	AMBA	AMBA4	master
AXI4	AMBA	AMBA4	slave
AXI4	AMBA	AMBA4	mirroredMaster
AXI4	AMBA	AMBA4	mirroredSlave
DDR3	Actel	busdef.memory	master
DDR3	Actel	busdef.memory	slave
PF_APB_LINK	Actel	busdef.link	master
PF_APB_LINK	Actel	busdef.link	slave
PF_CDR_CLK	Actel	busdef.clock	master
PF_CDR_CLK	Actel	busdef.clock	slave
PF_DRI	Actel	busdef.dri	master
PF_DRI	Actel	busdef.dri	slave
PF_DRI	Actel	busdef.dri	mirroredMaster
PF_DRI	Actel	busdef.dri	mirroredSlave
PF_TXPLL_XCVR_CLK	Actel	busdef.clock	master
PF_TXPLL_XCVR_CLK	Actel	busdef.clock	slave

`-port_bif_mapping {[bif_port_name:port_name ]+}`

Specifies the mapping between the bus interface formal names and the SmartDesign ports mapped onto that bus interface port. It is mandatory.

Examples

```
sd_create_bif_port -sd_name {sd1} -port_name {BIF_1} -port_bif_vlnv
{AMBA:AMBA2:APB:r0p0} -port_bif_role {slave} -port_bif_mapping {\
"PADDR:PADDR" \
"PSELx:pselx" \
"PENABLE:PENABLE" \
```

```
"PWRITE:PWRITE" \
"PRDATA:PRDATA" \
"PWDATA:PWDATA" \
"PREADY:PREADY" \
"PSLVERR:PSLVERR" }
```

See Also

Tcl Command Documentation Conventions

sd_create_bus_net

Tcl command; creates a bus net of a given range in a SmartDesign component. Any net created must be connected to two or more ports/pins using the command “sd_connect_net_to_pins”.

```
sd_create_bus_net \
-sd_name smartdesign_component_name \
-net_name net_name \
-net_range [left_index_range:right_index_range]
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-net_name net_name`

Specifies the name of the net to be added in the SmartDesign component. It is mandatory.

`-net_range [left_index_range:right_index_range]`

Specifies the range of the net added to the SmartDesign component. The range is defined by its left and right range indices. It is mandatory.

Examples

```
sd_create_bus_net -sd_name {top} -net_name {ab1} -net_range {[5:0]]}
```

Note: This new net is visible in the UI only when it is connected to two or more ports/pins using the command “sd_connect_net_to_pins” as shown below.

```
sd_connect_net_to_pins -sd_name {top} -net_name {ab1} -pin_names {a RAM64x12_0:R_ADDR}
```

Notes

This command is not required to build a SmartDesign component. It is not exported when you select Libero **Project** - `**Export Script File**’ or ‘**Export Component Description(Tcl)**’ on a SmartDesign component. This command is used to manually create a Tcl script and specify a new name to the net that connects two or more ports/pins.

See Also

Tcl Command Documentation Conventions

sd_create_bus_port

Tcl command; creates a bus port of a given range in a SmartDesign component.

```
sd_create_bus_port \
-sd_name smartdesign_component_name \
-port_name port_name \-port_direction IN|OUT|INOUT \
-port_range {[left_range_index:right_range_index]}
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-port_name` *`port_name`*

Specifies the name of the bus port added to be SmartDesign component. It is mandatory.

`-port_direction` *`IN|OUT|INOUT`*

Specifies the direction of the bus port added to the SmartDesign component. It is mandatory.

`-port_range` *`{[left_range_index:right_range_index]}`*

Specifies the range of the bus port added to the SmartDesign component. The range is defined by the left and right indices. It is mandatory. The range must be specified inside the square brackets.

Examples

```
sd_create_bus_port -sd_name {top} -port_name {test_port13} -port_direction {OUT} -
port_range {[9:36]}
sd_create_bus_port -sd_name {top} -port_name {test_port4} -port_direction {IN} -
port_range {[31:0]}
```

See Also

Tcl Command Documentation Conventions

sd_create_pin_group

Tcl command; creates a group of pins in a SmartDesign component. A pin group is only used to manage the complexity of the SmartDesign canvas. There is no actual netlist functionality related to pin group commands. Pin groups cannot be created for top level ports.

```
sd_create_pin_group \
-sd_name smartdesign_component_name \
-instance_name instance_name \
[-group_name group_name] \
[-pin_names pin_to_be_added_to_the_group]
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the name of the instance on which the pin group is added. It is mandatory.

`-group_name` *`group_name`*

Specifies the name of the pin group. It is optional. If the group name is not specified, the default name will be 'Group'. If the name 'Group' is already taken, then the group name will be 'Group_<index>' (index is auto-incremented).

`-pin_names` *`pins_to_be_added_to_the_group`*

Specifies the list of instance pins to be added to the pin group. It is optional.

Examples

```
sd_create_pin_group -sd_name {TOP} -instance_name
{COREAXI4INTERCONNECT_C0_0} -group_name {MyGroup} -pin_names {ACLK
ARESETN}
```

See Also

Tcl Command Documentation Conventions

sd_create_pin_slices

Tcl command; creates slices for a SmartDesign top level bus port or an instance level bus pin.

```
sd_create_pin_slices \
-sd_name smartdesign_component_name \
-pin_name port_or_pin_name \
-pin_slices port_or_pin_slices
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_name port_or_pin_name`

Specifies the name of the bus port or bus pin to be sliced. It is mandatory. This command will fail if the port/pin is scalar or if the bus port/pin does not exist.

`-pin_slices port_or_pin_slices`

Specifies the port/pin slices as a list of bus ranges which must be contained within the port/pin bus range. It is mandatory. This command will fail if the sliced object is top level OUT/INOUT port and the slice ranges overlap. This command will also fail if the sliced object is an instance level IN/INOUT pin and the slice ranges overlap.

Examples

```
sd_create_pin_slices -sd_name {sub} -pin_name {Rdata} -pin_slices {[4:3] [2:0]} # top level port slicing
```

```
sd_create_pin_slices -sd_name {sub} -pin_name {DDR_memory_arbiter_C0_0:VIDEO_RDATA_4_O} -pin_slices {[3:3] [2:0]} # instance level pin slicing
```

See Also

Tcl Command Documentation Conventions

sd_create_scalar_net

Tcl command; creates a scalar net in a SmartDesign component. Any net created must be connected to two or more ports/pins using the command “sd_connect_net_to_pins”.

```
sd_create_scalar_net \
-sd_name smartdesign_component_name \
-net_name net_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-net_name net_name`

Specifies the name of the net added to the SmartDesign component. It is mandatory.

Examples

```
sd_create_scalar_net -sd_name {top} -net_name {clk_net}
```

Note: This new net is visible in the UI only when it is connected to two or more ports/pins using the command “sd_connect_net_to_pins” as shown below

```
sd_connect_net_to_pins -sd_name {top} -net_name {clk_net} -pin_names {CLK RAM64x12_0:R_CLK RAM64x12_0:W_CLK}
```

Notes

This command is not required to build a SmartDesign component. It is not exported when you select Libero **Project** - `**Export Script File**' or '**Export Component Description(Tcl)**' on a SmartDesign component. This command is used to manually create a Tcl script and specify a new name to the net that connects two or more ports/pins.

See Also

Tcl Command Documentation Conventions

sd_create_scalar_port

Tcl command; creates a scalar port in a SmartDesign component.

```
sd_create_scalar_port \
-sd_name smartdesign_component_name \
-port_name port_name \
-port_direction IN|OUT|INOUT
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-port_name port_name`

Specifies the name of the port added to the SmartDesign component. It is mandatory.

`-port_direction IN|OUT|INOUT`

Specifies the direction of the port added to the SmartDesign component. It is mandatory.

Examples

```
sd_create_scalar_port -sd_name {main} -port_name {po2} -port_direction {INOUT}
```

See Also

Tcl Command Documentation Conventions

sd_delete_instances

Tcl command; deletes one or more instances from a SmartDesign component.

```
sd_delete_instances \
-sd_name smartdesign_component_name \
-instance_names instance_names
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_names instance_names`

Specifies the instance names to be deleted. It is mandatory.

Examples

```
sd_delete_instances -sd_name {top} -instance_names {RAM64X12_0}
sd_delete_instances -sd_name {SUB} -instance_names {coreahblite_c0_0
coreriscv_axi4_c0_0 pf_ccc_c0_0}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_delete_nets

Tcl command; deletes one or more nets from the SmartDesign component.

```
sd_delete_nets \
-sd_name smartdesign_component_name \
-net_names net_names
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-net_names net_names`

Specifies the net names to be deleted. It is mandatory.

Examples

```
sd_delete_nets -sd_name {topp} -net_names {B_REN_0}
```

Notes

This command will not delete multiple nets in this release. Support for deleting multiple nets will be provided in the next Libero release. This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_delete_pin_group

Tcl command; deletes a pin group from an instance in a SmartDesign component.

```
sd_delete_pin_group \
-sd_name smartdesign_component_name \
-instance_name instance_name \
-group_name group_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-group_name group_name`

Specifies the name of the pin group to be deleted. It is mandatory.

`-instance_name instance_name`

Specifies the name of the instance from which the group pin needs to be deleted. It is mandatory.

Examples

```
sd_delete_pin_group -sd_name {TOP} -instance_name
{COREAXI4INTERCONNECT_C0_0} -group_name {Group}
```

See Also

Tcl Command Documentation Conventions

sd_delete_pin_slices

Tcl command; deletes SmartDesign top level port slices or instance pin slices.

```
sd_create_pin_slices \
-sd_name smartdesign_component_name \
-pin_name port_or_pin_name \
-pin_slices port_or_pin_slices
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_name` *`port_or_pin_name`*

Specifies the name of the bus port or bus pin for which the slices must be deleted. It is mandatory.

`-pin_slices` *`port_or_pin_slices`*

Specifies the ranges of the port and/or pin slices to be deleted. It is mandatory.

Examples

```
sd_delete_pin_slices -sd_name {top} -pin_name {MACC_pa_0:p} -pin_slices
{[21] [13] [28]} # deletes instance pin slices
sd_delete_pin_slices -sd_name {top} -pin_name {A} -pin_slices {[17:16]
[15:1] [0]} # deletes top level port slices
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_delete_ports

Tcl command; deletes one or more ports from the SmartDesign component.

```
sd_delete_ports \
-sd_name smartdesign_component_name \
-port_names port_names
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-port_names` *`port_names`*

Specifies the names of the ports to be deleted. It is mandatory.

Examples

```
sd_delete_ports -sd_name {sd1} -port_names {REF_CLK_0}
```

Notes

This command will not work on multiple ports in this release. Support for multiple ports will be provided in the next Libero release. This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_disconnect_instance

Tcl command; clears all the connections on an instance in a SmartDesign component.

```
sd_disconnect_instance \
-sd_name smartdesign_component_name \
-instance_name instance_name
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the name of the instance for which all the connections must be cleared. It is mandatory.

Examples

```
sd_disconnect_instance -sd_name {sd1} -instance_name {RAM1K18_1}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_disconnect_pins

Tcl command; disconnects a list of SmartDesign top level ports and/or instance pins from the net they are connected to.

```
sd_disconnect_pins \
-sd_name smartdesign_component_name \
-pin_names port_or_pin_or_slice_names
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_names` *`port_or_pin_or_slice_names`*

Specifies the port, pin and/or slice names to be disconnected. It is mandatory. This command will fail if the ports, pins and/or slices do not exist.

Examples

```
sd_disconnect_pins -sd_name {topp} -pin_names {B_ren
RAM1K20_0:B_ADRR[12]}
sd_disconnect_pins -sd_name {SD1} -pin_names {AND2_0:B AND3_0:B AND3_0:A
PF_XCVR_ERM_C0_0:LANE0_RX_READY}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_duplicate_instance

Tcl command; creates a new instance in a SmartDesign with the same module/component as the original instance.

```
sd_duplicate_instance \
-sd_name smartdesign_component_name \
-instance_name instance_name \[-duplicate_instance_name duplicate_instance_name]
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the name of the instance to be duplicated. It is mandatory.

`-duplicate_instance_name` *`duplicate_instance_name`*

Specifies the name of the duplicate instance. It is optional. If the duplicate_instance_name is not specified, it will be automatically generated as <instance_name><index> (index is an automatically generated integer starting at 0 such that the instance name is unique in the SmartDesign).

Examples

```
sd_duplicate_instance -sd_name {top} -instance_name {PF_CCC_C0_0}
sd_duplicate_instance -sd_name {top} -instance_name {SUB_0} -
duplicate_instance_name {T1}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_hide_bif_pins

Tcl command; hides one or more already exposed internal scalar or bus pins/ports of a Bus Interface pin/port.

```
sd_hide_bif_pins \
-sd_name smartdesign_component_name \
-bif_pin_name name_of_the_bif_pin_or_port \-pin_names pins_or_ports_to_be_exposed
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-bif_pin_name name_of_the_bif_pin_name`

Specifies the name of the Bus Interface pin for which the internal pins must be hidden. It is mandatory.

`-pin_name pins_to_be_exposed`

Specifies the bus interface internal pin/port names to be hidden. It is mandatory.

Examples

```
sd_hide_bif_pins -sd_name {sd1} -bif_pin_name {COREAXI4INTERCONNECT_C0_0:AXI4mmaster0} -
pin_names {COREAXI4INTERCONNECT_C0_0:MASTER0_AWADDR}
```

```
sd_hide_bif_pins -sd_name {SD1} -bif_pin_name {CLKS_FROM_TXPLL_0} -pin_names
{TX_PLL_LOCK_0}
```

Notes

This command will not hide multiple pins/ports in this release. Support to hide multiple pins/ports will be provided in the next Libero release. This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_instantiate_component

Tcl command; instantiates a Libero SmartDesign component or a core component into another SmartDesign component.

```
sd_instantiate_component \
-sd_name smartdesign_component_name \
-component_name component_module_name \
[-instance_name instance name]
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component in which other components will be instantiated. It is mandatory.

`-component_name component_module_name`

Specifies the name of the component being instantiated in the SmartDesign component. It is mandatory. The components include SmartDesign components, core components created for different types of cores from the catalog and blocks.

`-instance_name instance name`

Specifies the instance name of the Libero component being instantiated in the SmartDesign component. It is optional. By default, the instance name is <component_module_name>_<index> (index is an automatically generated integer starting at 0 such that the instance name is unique in the SmartDesign).

Examples

```
sd_instantiate_component -sd_name {sub} -component_name {sd1} -
instance_name {sd1_0}
sd_instantiate_component -sd_name {top} -component_name {PF_CCC_C0}
```

See Also

Tcl Command Documentation Conventions

sd_instantiate_core

Tcl command; instantiates a core from the catalog directly into a SmartDesign component (Direct Instantiation) without first having to create a component for the core. The file-set related to the core is generated only when the SmartDesign in which the core is instantiated is generated. The GUI equivalent of this command is not currently supported in Libero. To instantiate a core in a SmartDesign component in the GUI, you have to first create a component for the core.

```
sd_instantiate_core \
-sd_name smartdesign_component_name \-core_vlnv vendor:library:name:verison \[-instance_name
instance_name]
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-core_vlnv vendor:library:name:verison`

Specifies the version identifier of the core being instantiated in the SmartDesign component. It is mandatory.

`-instance_name instance_name`

Specifies the instance name of the core being instantiated in the SmartDesign. It is optional. By default, the instance name is <core_name>_<index> (index is an automatically generated integer starting at 0 such that the instance name is unique in the SmartDesign).

Examples

```
sd_instantiate_core -sd_name {top} -core_vlnv
{Actel:DirectCore:COREAXI4INTERCONNECT:2.5.100} -instance_name
{COREAXI4INTERCONNECT_C0_0}
```

See Also

Tcl Command Documentation Conventions

sd_instantiate_hdl_core

Tcl command; instantiates a HDL+ core in a SmartDesign component. HDL+ core definition must be created on a HDL module before using this command.

```
sd_instantiate_hdl_core \
-sd_name smartdesign_component_name \
-hdl_core_name hdl_core_module_name \
[-instance_name instance_name]
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-hdl_core_name` *`hdl_core_module_name`*

Specifies the name of the HDL+ core module being instantiated in the SmartDesign component. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the instance name of the HDL+ core being instantiated in the SmartDesign. It is optional. By default, the instance name is <hdl_core_module_name>_<index> (index is an automatically generated integer starting at 0 such that the instance name is unique in the SmartDesign).

Examples

```
sd_instantiate_hdl_core -sd_name {top} -hdl_core_name {temp} -instance_name {temp3}
```

See Also

Tcl Command Documentation Conventions

sd_instantiate_hdl_module

Tcl command; instantiates a HDL module in a SmartDesign component. The HDL file in which the HDL module is defined must be imported/linked before running this command.

```
sd_instantiate_hdl_module \
-sd_name smartdesign_component_name \-hdl_module_name hdl_module_name \-hdl_file hdl_file \
[-instance_name instance_name]
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-hdl_module_name` *`hdl_module_name`*

Specifies the name of the HDL module being instantiated in the SmartDesign component. It is mandatory.

`-hdl_file` *`hdl_file`*

Specifies the path of the HDL file in which the HDL module is defined. The HDL file path can be relative to project folder for imported files but the path has to be complete for linked files. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the instance name of the HDL module. It is optional. By default, the instance name is <hdl_module_name>_<index> (index is an automatically generated integer starting at 0 such that the instance name is unique in the SmartDesign).

Examples

```
sd_instantiate_hdl_module -sd_name {top} -hdl_module_name {and1} -hdl_file {hdl\and1.v}
sd_instantiate_hdl_module -sd_name {top} -hdl_module_name {and_ex} -hdl_file
{hdl\and_ex.v} -instance_name {test_hdl_hdl_module_name_plus1_1}
```

See Also

Tcl Command Documentation Conventions

sd_instantiate_macro

Tcl command; instantiates a Microsemi primitive macro in a SmartDesign component.

```
sd_instantiate_macro \
-sd_name smartdesign_component_name \
-macro_name macro_module_name |
[-instance_name instance_name]
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-macro_name macro_module_name`

Specifies the name of the macro being instantiated in the SmartDesign component. It is mandatory.

`-instance_name instance_name`

Specifies the instance name of the macro. It is optional. By default, the instance name is <macro name>_<index> (index is an automatically generated integer starting at 0 such that the instance name is unique in the SmartDesign).

Examples

```
sd_instantiate_macro -sd_name {TOP} -macro_name {MX2} -instance_name {MX2_0}
```

sd_instantiate_macro -sd_name {TOP} -macro_name {MACC_PA}

See Also

Tcl Command Documentation Conventions

sd_invert_pins

Tcl command; inverts one or more top level ports or instance level pins in a SmartDesign.

```
sd_invert_pins \
-sd_name smartdesign_component_name \
-pin_names port_or_pin_names
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_names port_or_pin_names`

Specifies the port or pin names to be inverted. It is mandatory. This parameter can take multiple values. This command will fail if the port/pin does not exist.

Examples

```
sd_invert_pins -sd_name {main} -pin_names {A}
sd_invert_pins -sd_name {main} -pin_names {MX2_1:S MX2_1:Y A B}
```

See Also

Tcl Command Documentation Conventions

sd_mark_pins_unused

Tcl command; marks one or more SmartDesign instance level output pins as unused. When an output pin is marked as unused, no Design Rule Check (DRC) warning will be printed for floating output pins while generating the SmartDesign.

```
sd_mark_pins_unused \
-sd_name smartdesign_component_name \
-pin_names port_or_pin_names
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-pin_names port_or_pin_names`

Specifies the names of the instance pins to be marked as unused. It is mandatory.

Examples

```
sd_mark_pins_unused -sd_name {top} -pin_names {PF_CCC_C0_0:PLL_LOCK_0}
```

Notes

This command will not work on multiple pins in this release. Support for multiple pins will be provided in the next Libero release.

See Also

Tcl Command Documentation Conventions

sd_remove_pins_from_group

Tcl command; removes one or more pins from a pin group on an instance in a SmartDesign.

```
sd_remove_pins_from_group \
-sd_name smartdesign_component_name \
-instance_name instance_name \
-group_name group_name \
-pin_names pin_names
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name instance_name`

Specifies the name of the instance on which the pin group is present. It is mandatory.

`-group_name group_name`

Specifies the name of the pin group from which pins need to be removed. It is mandatory.

`-pin_names pin_names`

Specifies the list of pin names to be removed from the pin group. It is mandatory.

Examples

```
sd_remove_pins_from_group -sd_name {TOP} -instance_name
{COREAXI4INTERCONNECT_C0_0} -group_name {Group} -pin_names {ARESETN ACLK}
```

See Also

Tcl Command Documentation Conventions

sd_rename_instance

Tcl command; renames an instance in a SmartDesign component. This command can be used to rename any type of instances (instances of other SmartDesigns components, core components, HDL modules, HDL+ cores and Microsemi macros) in a SmartDesign.

```
sd_rename_instance \
-sd_name component_name \
-current_instance_name instance_name \
-new_instance_name new_instance_name
```

Arguments

`-sd_name component_name`

Specifies the name of the SmartDesign component in which the instance name has to be renamed. It is mandatory.

`-current_instance_name instance_name`

Specifies the name of the instance to be renamed. It is mandatory.

`-new_instance_name new_instance_name`

Specifies the new instance name. It is mandatory.

Examples

```
sd_rename_instance -sd_name {top} -current_instance_name {DFN1_0} -
new_instance_name {DFN1_new}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_rename_net

Tcl command; renames a net in a SmartDesign component.

```
sd_rename_net \
-sd_name smartdesign_component_name \
-current_net_name current_net_name \-new_net_name_new_net_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-current_net_name current_net_name`

Specifies the name of the net to be renamed in the SmartDesign. It is mandatory.

`-new_net_name new_net_name`

Specifies the new name of the net in the SmartDesign. It is mandatory.

Examples

```
sd_rename_net -sd_name {top} -current_net_name {clk_net} -new_net_name {clk_rclk_wclk}
sd_rename_net -sd_name {PCIe_EP_Demo} -current_net_name {USER_RESETN} -new_net_name
{reset_input}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_rename_pin_group

Tcl command; renames a pin group on an instance in a SmartDesign component.

```
sd_rename_pin_group \
-sd_name smartdesign_component_name \
-instance_name instance_name \
-current_group_name current_pin_group_name \
-new_pin_group_name new_pin_group_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name instance_name`

Specifies the name of the instance on which the pin group is present. It is mandatory.

`-current_group_name current_pin_group_name`

Specifies the name of the pin group to be renamed. It is mandatory.

`-new_group_name new_group_name`

Specifies the new name of the pin group. It is mandatory.

Examples

```
sd_rename_pin_group -sd_name {TOP} -instance_name
{COREAXI4INTERCONNECT_C0_0} -current_group_name {Group} -new_group_name
{MyNewGroup}
```

See Also

Tcl Command Documentation Conventions

sd_rename_port

Tcl command; renames a SmartDesign port.

```
sd_rename_port \
-sd_name smartdesign_component_name \
-current_port_name port_name \
-new_port_name new_port_name
```

Arguments

`-sd_name smartdesign_component_name`

Specifies the name of the SmartDesign component. It is mandatory.

`-current_port_name port_name`

Specifies the name of the port to be renamed in the SmartDesign component. It is mandatory. Note that only port names can be renamed, and not port types (scalar ports cannot be renamed as bus ports and vice versa).

`-new_port_name` *new_port_name*

Specifies the new name of the specified port. It is mandatory.

Examples

```
sd_rename_port -sd_name {top} -library {work} -current_port_name {c1} -new_port_name
{c2}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

sd_save_core_instance_config

Tcl command; this command is used to save the core instance configuration specified using one or more 'sd_configure_core_instance' commands. This command is typically used after configuring a core instance in a SmartDesign, to save that core instance's configuration.

```
sd_save_core_instance_config \
-sd_name smartdesign_component_name \
-instance_name core_instance_name
```

Arguments

`-sd_name` *smartdesign_component_name*

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name` *instance_name*

Specifies the name of the core instance in the SmartDesign for which the configuration must be saved. It is mandatory.

Examples

```
sd_save_core_instance_config -sd_name {SD1} -instance_name {COREFIFO_0}
```

See Also

Tcl Command Documentation Conventions

sd_show_bif_pins

Tcl command; exposes one or more internal scalar or bus pins/ports of a Bus Interface pin/port. A Bus Interface pin/port is usually a group of normal scalar or bus pins/ports grouped together and used to connect instances that have similar interfaces. The internal pins/ports underneath the Bus Interface pin/port may have to be exposed in some cases to connect to some logic in the design.

```
sd_show_bif_pins \
-sd_name smartdesign_component_name \
-bif_pin_name name_of_the_bif_pin_or_port \
-pin_names pins_or_ports_to_be_exposed
```

Arguments

`-sd_name` *smartdesign_component_name*

Specifies the name of the SmartDesign component. It is mandatory.

`-bif_pin_name` *`name_of_the_bif_pin_or_port`*

Specifies the name of the Bus Interface pin/port for which the internal pins/ports need to be exposed. It is mandatory.

`-pin_names` *`pins_or_ports_to_be_exposed`*

Specifies the names of the Bus Interface internal pins/ports to be exposed. It is mandatory.

Examples

```
sd_show_bif_pins -sd_name {TOP} -bif_pin_name {COREAXI4INTERCONNECT_C0_0:AXI4mmaster0} -
pin_names {COREAXI4INTERCONNECT_C0_0:MASTER0_AWADDR}
```

```
sd_show_bif_pins -sd_name {SD1} -bif_pin_name {CLKS_FROM_TXPLL_0} -pin_names
{TX_PLL_LOCK_0}
```

Notes

This command will not expose multiple pins/ports in this release. Support to expose multiple scalar or bus pins/ports will be provided in the next Libero release.

See Also

Tcl Command Documentation Conventions

sd_update_instance

Tcl command; updates an instance in a SmartDesign with its latest definition. This command is useful when the interface (port-list) of the component/module instantiated in a SmartDesign has changed. This command can be used to update any type of instance such as instances of other SmartDesign components, core components, HDL modules and HDL+ cores in a SmartDesign.

```
sd_update_instance \
-sd_name smartdesign_component_name \
-instance_name instance_name
```

Arguments

`-sd_name` *`smartdesign_component_name`*

Specifies the name of the SmartDesign component. It is mandatory.

`-instance_name` *`instance_name`*

Specifies the name of the instance to be updated. It is mandatory.

Examples

```
sd_update_instance -sd_name {top} -instance_name {CORESMIP_C0_0}
```

Notes

This command is not required to build a SmartDesign component. This command maps to an interactive user action in the SmartDesign Canvas and will not be present in the exported SmartDesign component Tcl description.

See Also

Tcl Command Documentation Conventions

HDL Core Tcl Commands

create_hdl_core

This Tcl command is used to create a core component from an HDL core.

```
create_hdl_core \
-module {module_name} \
-file {file_path}
-library {library_name} \
-package {package_name}
```

Arguments

`-module {module_name}`

Specify the module name for which you want to create a core component. This is a mandatory argument.

`-file {file_path}`

Specify the file path of the module from which you create a core component. This is a mandatory argument.

`-library {library_name}`

Specify the library name from which you want to create a HDL core. This is an optional argument.

`-package {package_name}`

Specify the package name from which you want to create a core component. This is an optional argument.

Example

```
create_hdl_core -file {./HDL_CORE_TEST/hdl/hdl_core.v} -module {test_hdl_core}
```

See Also

Tcl Command Documentation Conventions

hdl_core_add_bif

This Tcl command adds a bus interface to an HDL core.

```
hdl_core_add_bif \
-hdl_core_name {hdl_core_name} \
-bif_definition {Name:Vendor:Library:Role} \
-bif_name {bus_interface_name} \
[-signal_map {signal_map}]
```

Arguments

`-module {module_name}`

Specify the HDL core name to which the bus interface needs to be added. This is a mandatory argument.

`-bif_definition {Name:Vendor:Library:Role}`

Specify the Bus Interface Definition Name, Vendor, Library and Bus Role of the core in the format {N:V:L:R}. This is a mandatory argument.

`-bif_name {bus_interface_name}`

Specify the bus interface port name being added to the HDL core. This is a mandatory argument.

-signal_map {*signal_map*}

This argument is used to specify the signal map of the bus interface. This is an optional argument.

Example

```
hdl_core_add_bif -hdl_core_name {test_hdl_core} -bif_definition {AHB:AMBA:AMBA2:master} -
bif_name {BIF_1}
```

See Also

Tcl Command Documentation Conventions

hdl_core_assign_bif_signal

Maps a bus interface signal definition name to an HDL core module port name.

```
hdl_core_assign_bif_signal
-hdl_core_name {hdl_core_name} \
-bif_name {bus_interface_name} \
-bif_signal_name {bif_signal_name} \
-core_signal_name {core_signal_name}
```

Arguments

-hdl_core_name {*hdl_core_name*}

Specify the HDL core name to which the bus interface signal needs to be added. This is a mandatory argument.

-bif_name {*bus_interface_name*}

Specify the bus interface name for which you want to map a core signal. This is a mandatory argument.

-bif_signal_name {*bus_interface_signal_name*}

Specify the bus interface signal name that you want to map with the core signal name. This is a mandatory argument.

-core_signal_name {*core_signal_name*}

Specify the core signal name for which you want to map the bus interface signal name. This is a mandatory argument.

Example

```
hdl_core_assign_bif_signal -hdl_core_name {test_hdl_core} -bif_name {BIF_1} -
bif_signal_name {HWRITE} -core_signal_name {myHRESULT}
```

See Also

Tcl Command Documentation Conventions

hdl_core_delete_parameters

This Tcl command deletes parameters from a HDL core definition.

```
hdl_core_delete_parameters
-hdl_core_name {module_name} \
-parameters {parameter_list}
```

Arguments

-hdl_core_name {*hdl_core_name*}

Specify the HDL core name from which you want to delete parameters. This is a mandatory argument.

-parameters {*parameter_list*}

Specify the list of parameters from a HDL core. This is typically done to remove parameters from the list of parameters that was automatically extracted using the hdl_core_extract_ports_and_params command. This is a mandatory argument.

Example

```
hdl_core_delete_parameters -hdl_core_name {test_hdl_core} -parameters {WIDTH}
```

See Also

Tcl Command Documentation Conventions

hdl_core_extract_ports_and_parameters

This Tcl command automatically extracts ports and generic parameters from an HDL core module description.

```
hdl_core_extract_ports_and_parameters \
-hdl_core_name {hdl_core_name}
```

Arguments

-hdl_core_name *hdl_core_name*}

Specifies the HDL core name from which you want to extract signal names and generic parameters. This is a mandatory argument.

Example

```
hdl_core_extract_ports_and_params -hdl_core_name {test_hdl_core}
```

See Also

Tcl Command Documentation Conventions

hdl_core_remove_bif

Remove an existing bus interface from an HDL core.

```
hdl_core_remove_bif \
-hdl_core_name {hdl_core_name} \
-bif_name {bus_interface_name}
```

Arguments

-module {*module_name*}

Specify the HDL core name from which the bus interface needs to be removed. This is a mandatory argument.

-bif_name {*bus_interface_name*}

Specify the bus interface name that needs to be removed from the HDL core. This is a mandatory argument.

Example

```
hdl_core_remove_bif -hdl_core_name {mod1} -bif_name {BIF_1}
```

See Also

Tcl Command Documentation Conventions

hdl_core_rename_bif

Rename an existing bus interface port of a HDL core.

```
hdl_core_rename_bif
-hdl_core_name {hdl_core_name} \
-current_bif_name {current_bus_interface_name} \
-new_bif_name {new_bus_interface_name}
```

Arguments

`-hdl_core_name {hdl_core_name}`

Specify the HDL core name for which the bus interface needs to be renamed. This is a mandatory argument.

`-current_bif_name {current_bus_interface_name}`

Specify the bus old bus interface name that needs to be renamed for the HDL core. This is a mandatory argument.

`-new_bif_name {new_bus_interface_name}`

Specify the new bus interface name that needs to be updated for the HDL core. This is a mandatory argument.

Example

```
hdl_core_rename_bif -hdl_core_name {test_hdl_plus} -current_bif_name {BIF_2} -
new_bif_name {BIF_3}
```

See Also

Tcl Command Documentation Conventions

hdl_core_unassign_bif_signal

Unmap an existing bus interface signal from a bus interface.

```
hdl_core_unassign_bif_signal
-hdl_core_name {hdl_core_name} \
-bif_name {bus_interface_name} \
-bif_signal_name {bif_signal_name}
```

Arguments

`-hdl_core_name {hdl_core_name}`

Specify the HDL core name from which the bus interface signal needs to be deleted. This is a mandatory argument.

`-bif_name {bus_interface_name}`

Specify the bus interface name for which you want to unassign a core signal. This is a mandatory argument.

`-bif_signal_name {bus_interface_signal_name}`

Specify the bus interface signal name for which you want to unassign a core signal. This argument is mandatory.

Example

```
hdl_core_unassign_bif_signal -hdl_core_name {test_hdl_plus} -bif_name {BIF_2} -
bif_signal_name {PENABLE}
```

See Also

Tcl Command Documentation Conventions

remove_hdl_core

This Tcl command removes an HDL core component from the current project.

```
remove_hdl_core \
-hdl_core_name {hdl_core_name}
```

Arguments

`-hdl_core_name {hdl_core_name}`

Specify the module name from which you want to delete a core component. This is a mandatory argument.

Example

```
remove_hdl_core -hdl_core_name {test_hdl_core}
```

See Also

Tcl Command Documentation Conventions

Command Tools

CONFIGURE_CHAIN

CONFIGURE_CHAIN is a command tool used in run_tool. The command run_tool -name {CONFIGURE_CHAIN} takes a script file that contains specific Tcl commands and passes them to FlashPro Express for execution.

```
run_tool -name {CONFIGURE_CHAIN} -script {fpro_cmds.tcl}
```

fpro_cmds.tcl is a Tcl script that contains specific Tcl commands to configure JTAG chain. For details on JTAG chain programming Tcl commands, refer to the Tcl commands section in the Libero SoC Online Help.

Do not include any project-management commands such as open_project, save_project, or close_project in this *fpro_cmds.tcl* script file. The run_tool –name {CONFIGURE_CHAIN} command generates these project-management commands for you.

Note: For a new Libero project without a JTAG chain, executing this command causes Libero to first add the existing design device to the JTAG chain and then execute the commands from the script. If, for example, the script *fpro_cmds.tcl* contains commands to add four devices, executing the command run_tool –name {CONFIGURE_CHAIN} -script {*fpro_cmds.tcl*} will create a JTAG chain of the Libero design device and the four devices. For existing Libero projects that already have a JTAG chain, the command is executed on the existing JTAG chain.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
run_tool -name {CONFIGURE_CHAIN} -script {d:/fpro_cmds.tcl}
```

#Example fpro_cmds.tcl command file for the –script parameter

```
add_actel_device \
  -file {./sd_prj/sp_g3/designer/impl1/sd1.stp} \
  -name {dev1}
enable_device -name {M2S050TS_5} -enable 0
add_non_actel_device \
  -ir 2 \
  -tck 1.00 \
  -name {Non-Microsemi Device}
add_non_actel_device \
  -ir 2 \
  -tck 1.00 \
  -name {Non-Microsemi Device (2)}
remove_device -name {Non-Microsemi Device}
set_device_to_highz -name {M2S050TS_5} -highz 1
add_actel_device \
  -device {M2S050TS} \
  -name {M2S050TS (3)}
select_libero_design_device -name {M2S050TS (3)}
```

Return

Returns 0 on success and 1 on failure.

CONFIGURE_PROG_OPTIONS

CONFIGURE_PROG_OPTIONS is a command tool used in configure_tool. Configure_tool -name {CONFIGURE_PROG_OPTIONS} sets the programming options.

```
configure_tool -name {CONFIGURE_PROG_OPTIONS}
-params {design_version:<value>}
-params {silicon_signature:<value>}
-params {enable_auto_update:true | false}
-params {enable_prog_recovery:true | false}
-params {spi_clk_freq:<value>}
-params {spi_data_transfer_mode: <value>}
```

The following table lists the parameter names and values.

configure_tool –name {CONFIGURE_PROG_OPTIONS} parameter:value pair

Name	Value	Description
design_version	Integer {0 through 65535}	Sets the design version. It must be greater than the Back level version in SPM Update Policy.
enable_auto_update	boolean {true | false}	Enables auto update when set to true and disables it when set to false
enable_prog_recovery	boolean {true | false}	Enables programming recovery when set to true and disables it when set to false
silicon_signature	Hex {<max length 8 Hex characters>}	32-bit (8 hex characters) silicon signature to be programmed into the device. This field can be read from the device using the JTAG USERCODE instruction.
spi_clock_freq	Sets SPI clock frequency from a list of possible values {1.00 | 2.08 |3.13|4.16|5.00| 6.25| 8.30|12.50|25.00}	
spi_data_transfer_mode	{100 | 101|111|110}	SPI data transfer mode sets the values for SPS, SPO and SPH in the UI. SPS has a fixed value of 1 and cannot be changed. The user can change the value of only SPO and SPH to 0 or 1.

Supported Families

SmartFusion2

IGLOO2

Example

```
configure_tool -name {CONFIGURE_PROG_OPTIONS}\
  -params {design_version:255}
  -params {enable_auto_update:true}
  -params {enable_prog_recovery:true}
```

```
-params {silicon_signature:abcdef}
-params {spi_clk_freq:25.00}
-params {spi_data_transfer_mode:100}
```

Return

Returns 0 on success and 1 on failure.

See Also

Configure Programming Options (SmartFusion2 and IGLOO2)

CONFIGURE_PROG_OPTIONS_RTG4 (RTG4 only)

CONFIGURE_PROG_OPTIONS_RTG4 is a command tool used in configure_tool. Configure_tool -name {CONFIGURE_PROG_OPTIONS_RTG4} sets the programming options for RTG4 devices.

```
configure_tool -name {CONFIGURE_PROG_OPTIONS_RTG4}
-params {design_version:1}
-params {silicon_signature: abcd}
-params {disable_digest_check:true}
-params {disable_fabric_erase_write_verify:true}
-params {disable_jtag:true}
-params {disable_probe_read_write:false}
-params {disable_spi:false}
-params {one_time_programmable:false}
-params {system_controller_supsend_mode:false}
```

The following table lists the parameter names and values.

configure_tool –name {CONFIGURE_PROG_OPTIONS_RTG4} parameter:value pair

Name	Value	Description
design_version	Integer {0 through 65535}	Sets the design version. It must be greater than the Back level version in SPM Update Policy.
silicon_signature	Hex {<max length 8 Hex characters>}	32-bit (8 hex characters) silicon signature to be programmed into the device. This field can be read from the device using the JTAG USERCODE instruction.
disable_digest_check	boolean {true | false}	
disable_fabric_erase_write_verify	boolean {true | false}	
disable_jtag	boolean {true | false}	
disable_probe_read_write	boolean {true | false}	
disable_spi	boolean {true | false}	

Name	Value	Description
one_time_programmable	boolean {true | false}	
system_controller_suspsend_mode	boolean {true | false}	

Supported Families

RTG4

Example

```
configure_tool -name {CONFIGURE_PROG_OPTIONS_RTG4}\
  -params {design_version:255}\
  -params {silicon_signature: abcd}\
  -params {disable_digest_check:true}\
  -params {disable_fabric_erase_write_verify:true}\
  -params {disable_jtag:true}\
  -params {disable_probe_read_write:false}\
  -params {disable_spi:false}\
  -params {one_time_programmable:false}\
  -params {system_controller_supsend_mode:false}
```

Return

Returns 0 on success and 1 on failure.

See Also

Configure Programming Options (RTG4 Only)

FLASH_FREEZE

FLASH_FREEZE is a command tool used in configure_tool. You use the configure_tool -name {FLASH_FREEZE} command to specify:

- The state of the uRAM and LSRAM when the FPGA fabric is in the Flash Freeze state.
- The MSS clock source when the FPGA fabric is in the Flash Freeze state.

```
configure_tool -name {FLASH_FREEZE}
-params {name:value}
-params {name:value}
```

configure_tool –name {FLASH_FREEZE} parameter:value pair

Params_name	<Params_value>	Description
FF_RAM_STATE	Enum {SUSPEND | SLEEP }	Specifies the uRAM and LSRAM state during Flash Freeze. Default is SUSPEND.
FF_MSS_CLOCK	Enum {RCOSC_1MHZ | RCOSC_50MHZ}	Specifies the MSS Clock Source during Flash Freeze. Default is RCOSC_1MHZ.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
configure_tool -name {FLASH_FREEZE}\
  -params {FF_RAM_STATE:SUSPEND}\
  -params {FF_MSS_CLOCK:RCOSC_1MHZ}
```

Return

Returns 0 on success and 1 on failure.

GENERATEPROGRAMMINGFILE

GENERATEPROGRAMMINGFILE is a command tool used in the configure_tool command and the run_tool command. The configure_tool -name {GENERATEPROGRAMMINGFILE} Tcl command configures tool options. The run_tool -name {GENERATEPROGRAMMINGFILE} Tcl command generates the Bitstream used for programming.

```
run_tool -name {GENERATEPROGRAMMINGFILE}
```

This command takes no parameters.

Supported Families

SmartFusion2, IGLOO2, RTG4

Return

Returns 0 on success and 1 on failure.

INIT_LOCK

INIT_LOCK is a TCL equivalent command name for the Configure Register Lock Bits tool. It is passed as a parameter to the configure_tool command.

The INIT_LOCK command imports a Lock Bit Configuration File (*.txt) and configures the Register Lock Bits of FDDR, MSS, and SERDES blocks for SmartFusion2 and IGLOO2 devices so that they cannot be overwritten by Fabric or MSS Masters that have write access to these registers. This command takes only one parameter, INIT_LOCK_FILE, which has the configuration file's full path or relative path as its value.

```
configure_tool -name {INIT_LOCK} -params {INIT_LOCK_FILE: \
<full _or_relative_path_to_config_lock_bit_file>}
```

Supported Families

SmartFusion2 and IGLOO2

Example

```
configure_tool -name {INIT_LOCK}\
   -params {INIT_LOCK_FILE:D:/designs/g4_fclk_mddr_clk/sb_init_config_lock_bits_src.txt}
# full path
configure_tool -name {INIT_LOCK}\
   -params {INIT_LOCK_FILE: ../ sb_init_config_lock_bits_src.txt} # relative path from
project folder
```

Return

```
configure_tool -name { INIT_LOCK } -params {INIT_LOCK_FILE: \
<full _or_relative_path_to_config_lock_bit_file>}
```

Returns 0 on success and 1 on failure.

PROGRAMDEVICE

PROGRAMDEVICE is a command tool used in configure_tool and run_tool. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. Run_tool executes the tool with the configured parameters.

To program the design in Libero SoC, you must first configure the PROGRAMDEVICE tool with configure_tool command and then execute the PROGRAMDEVICE command with the run_tool command.

Use the commands to configure your programming action and the programming procedures associated with the program action.

```
configure_tool -name {PROGRAMDEVICE}
-params {prog_action:params_value}
-params {prog_optional_procedures:params_value}

run_tool -name {PROGRAMDEVICE}
```

configure_tool –name {PROGRAMDEVICE} parameter:value pair

Name	Value	Description
prog_action	String { PROGRAM | VERIFY | ERASE | DEVICE_INFO | READ_IDCODE | ENC_DATA_AUTHENTICATION | VERIFY_DIGEST}	PROGRAM – Programs all selected family features: FPGA Array, targeted eNVM clients and security settings. VERIFY – Verifies all selected family features: FPGA Array, targeted eNVM clients and security settings. ERASE – Erases the selected family features: FPGA Array and security settings. DEVICE_INFO – Displays the IDCODE, the design name, the checksum, and device security settings and programming environment information programmed into the device.READ_IDCODE – Reads the device ID code from the device. ENC_DATA_AUTHENTICATION - Encrypted bitstream authentication data. VERIFY_DIGEST – Calculates the digests for the components included in the bitstream and compares them against the programmed values
prog_optional_procedures	Depends on the action from the prog_action parameter.	This parameter is optional. It is only required when the user wants to enable optional procedure.

run_tool –name {PROGRAMDEVICE} Parameter:value pair

Name	Value	Description
None		

Supported Families

SmartFusion2

IGLOO2

Example

```
configure_tool \
    -name {PROGRAMDEVICE} \
  -params {prog_action:PROGRAM} \
  -params {prog_optional_procedures:DO_VERIFY}\
configure_tool -name {PROGRAMDEVICE} -params {prog_action:DEVICE_INFO}
run_tool -name {PROGRAMDEVICE} #Takes no parameters
```

Return

`configure_tool -name {PROGRAMDEVICE}` returns 0 on success and 1 on failure.

`run_tool -name {PROGRAMDEVICE}` returns 0 on success and 1 on failure.

PLACEROUTE

To place and route a design in Libero SoC, you must first configure the PLACEROUTE tool with the configure_tool command and then execute the PLACEROUTE tool with the run_tool command.

configure_tool

```
configure_tool -name {PLACEROUTE} [-params {[name:value ]+}]+
```

Parameters

Name	Value	Description
TDPR	true | false | 1 | 0	Set to true or 1 to enable Timing-Driven Place and Route. Default is 1.
PDPR	true | false | 1 | 0	Set to true or 1 to enable Power-Driven Place and Route. Default is false or 0.
IOREG_COMBINING	true | false | 1 | 0	Set to true or 1 to enable I/O Register Combining. **Default is true or 1.**
EFFORT_LEVEL	true | false | 1 | 0	Set to true or 1 to enable High Effort Layout to optimize design performance. Default is false or 0.
INCRPLACEANDROUTE	true | false | 1 | 0	Set to true or 1 to use previous placement data as the initial placement for the next run. Default is false or 0.
REPAIR_MIN_DELAY	true | false | 1 | 0	Set to 1 to enable Repair Minimum Delay violations for the router when TDPR option is set to true or 1. Default is false.

Name	Value	Description
NUM_MULTI_PASSES	1-25	Specifies the number of passes to run. The default is 5. Maximum is 25.
START_SEED_INDEX	1-100	Indicates the random seed index which is the starting point for the passes. Its value should range from 1 to 100. If not specified, the default behavior is to continue from the last seed index which was used.
MULTI_PASS_LAYOUT	true | false | 1 | 0	Set to true or 1 to enable Multi-Pass Layout Mode for Place and Route. Default is false or 0.
MULTI_PASS_CRITERIA	SLOWEST_CLOCK SPECIFIC_CLOCK VIOLATIONS TOTAL_POWER	Specifies the criteria used to run multi-pass layout: • hat has the lowest total power (static + dynamic) out of all layout passes.
SPECIFIC_CLOCK	Clock_Name	Applies only when MULTI_PASS_CRITERIA is set to SPECIFIC_CLOCK. It specifies the name of the clock in the design used for Timing Violation Measurement.
DELAY_ANALYSIS	max | min	Used only when MULTI_PASS_CRITERIA is set to “VIOLATIONS”. Specifies the type of timing violations (slacks) to be examined. The default is 'max'. • max: Use timing violations (slacks) obtained from maximum delay analysis • min: Use timing violations (slacks) obtained from minimum delay analysis.
STOP_ON_FIRST_PASS	true | false | 1 | 0	Applies only when MULTI_PASS_CRITERIA is set to “VIOLATIONS”. It stops performing remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report). Note: The type of timing violations (slacks) used is determined by the 'DELAY_ANALYSIS ' parameter.
SLACK_CRITERIA	WORST_SLACK TOTAL_NEGATIVE_SLACK	Applies only when MULTI_PASS_CRITERIA is set to VIOLATIONS. Specifies how to evaluate the timing violations (slacks). The default is WORST_SLACK. • WORST_SLACK: The largest amount of negative slack (or least amount of positive slack if all constraints are met) for each pass is identified and then the largest value out of all passes

Name	Value	Description
		will determine the best pass. This is the default. • TOTAL_NEGATIVE_SLACK: The sum of negative slacks from the first 100 paths for each pass in the Timing Violation report is identified. The largest value out of all passes will determine the best pass. If no negative slacks exist for a pass, then use the worst slack to evaluate that pass. Note: The type of timing violations (slacks) used is determined by the 'DELAY_ANALYSIS' parameter.
RGB_COUNT	1-18	Allows an entity to override the placer's RGB/RCLK bandwidth constraint. This option is useful for Block Creation.

Return Value

Returns 0 on success and 1 on failure.

run_tool

run_tool -name {PLACEROUTE}

Parameters

None

Return Value

Returns 0 on success and 1 on failure.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
configure_tool -name {PLACEROUTE}\
  -params {EFFORT_LEVEL:true}\
  -params {INCRPLACEANDROUTE:false}\
  -params {IOREG_COMBINING:false}\
  -params {MULTI_PASS_CRITERIA:VIOLATIONS}\
  -params {MULTI_PASS_LAYOUT:false}\
  -params {NUM_MULTI_PASSES:5}\
  -params {PDPR:false}\
  -params {REPAIR_MIN_DELAY:true}\
  -params {SLACK_CRITERIA:WORST_SLACK}\
  -params {SPECIFIC_CLOCK:}\
  -params {START_SEED_INDEX:1}\
```

```
  -params {STOP_ON_FIRST_PASS:false}\
  -params {TDPR:true}\
  -params {USE_RAM_MATH_INTERFACE_LOGIC:false}
run_tool -name{PLACEROUTE}
```

PROGRAM_OPTIONS (SmartFusion2 and IGLOO2)

PROGRAM_OPTIONS is a command tool used in configure_tool. Configure_tool allows you to configure the tool's parameters and values. This Tcl command is the Tcl equivalent of the Configure Bitstream options in the Design Flow window.

```
configure_tool -name {PROGRAM_OPTIONS}
-params {name:value}
[-params {name:value}]
```

The following table lists the parameter names and values.

configure_tool –name {PROGRAM_OPTIONS} parameter:value pair

Name	Value	Description
program_envm	Boolean {true | false | 1 | 0}	Include eNVM component in the programming bitstream ("true" only if eNVM available in the design).
program_fabric	Boolean {true | false | 1 | 0}	Include fabric component in the programming bitstream.
program_mode	String {selected feature}	Only "selected_features" is supported.
program_security	Boolean {true | false | 1 | 0}	Include custom security component in the programming bitstream ("true" only if custom security was defined).

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {PROGRAM_OPTIONS}\
  -params {program_envm:false}\
  -params {program_fabric:true}\
  -params {program_mode:selected_features}\
  -params {program_security:true}
```

Return

Returns 0 on success and 1 on failure.

PROGRAM_RECOVERY

Configures the parameters in the Configure User Programming Data dialog box within Libero. See the online help for more information

```
configure_tool -name {PROGRAM_RECOVERY}
-params {enable_auto_update:value}
-params {enable_prog_recovery:value}
-params {spi_clk_freq:value}
-params {spi_data_transfer_mode:value}
```

Arguments

`Enable_auto_update {true|false}`

Enables auto update when set to true and disables it when set to false.

`Enable_prog_recovery {true|false}`

Enables programming recovery when set to true and disables when set to false.

`Spi_clk_freq {1.00|2.08|3.13|4.16|5.00|6.25|8.30|12.50|25.00}`

SPI clock frequency can be set to one of the values specified above.

`Spi_data_transfer_mode {100}`

SPI data transfer mode will set the values for SPS, SPO and SPH in the UI. SPS has a fixed value of 1. The user can change the values of only SPO and SPH to 0 or 1.

Supported Families

SmartFusion2, IGLOO2

Exceptions

None

Example

The following example sets the auto update and programming recovery to be ON. SPI clock frequency is set to 25MHz. SPO and SPH are set to 0.

```
configure_tool -name {PROGRAM_RECOVERY}\
  -params {enable_auto_update:true}\
  -params {enable_prog_recovery:true}\
  -params {spi_clk_freq:25.00}\
  -params {spi_data_transfer_mode:100}
```

PROGRAMMER_INFO

PROGRAMMER_INFO is a command tool used in configure_tool. Configure_tool -name {PROGRAMMER_INFO} sets the programmer settings, similar to the way FlashPro commands set the programmer settings. This command supports all five programmers: FlashPro3, FlashPro4, FlashPro5, and FlashPro6.

```
configure_tool -name {PROGRAMMER_INFO}
-params [{name:value}]
```

The following tables list the parameter names and values.

configure_tool –name {PROGRAMMER_INFO} Parameter:value (FlashPro6)

Name	Value	Description
flashpro6_force_freq	String {OFF | ON}	For FlashPro6 Programmer only.
flashpro6_ freq	Integer (Hertz)	For FlashPro6 Programmer only.

configure_tool –name {PROGRAMMER_INFO} Parameter:value (FlashPro5)

Name	Value	Description
flashpro5_clk_mode	String {free_running_clk | discrete_clocking}	For FlashPro5 Programmer only.
flashpro5_force_freq	String {OFF | ON}	For FlashPro5 Programmer only.
flashpro5_ freq	Integer (Hertz)	For FlashPro5 Programmer only.
flashpro5_vpump	String {ON | OFF}	For FlashPro5 Programmer only.

configure_tool –name {PROGRAMMER_INFO} Parameter:value (FlashPro4)

Name	Value	Description
flashpro4_clk_mode	String {free_running_clk | discrete_clocking}	For FlashPro4 Programmer only.
flashpro4_force_freq	String {OFF | ON}	For FlashPro4 Programmer only.
flashpro4_ freq	Integer (Hertz)	For FlashPro4 Programmer only.
flashpro4_vpump	String {ON | OFF}	For FlashPro4 Programmer only.

configure_tool –name {PROGRAMMER_INFO} parameter:value (FlashPro3)

Name	Value	Description
flashpro3_clk_mode	String {free_running_clk | discrete_clocking}	For FlashPro3 Programmer only.
flashpro3_force_freq	String {OFF | ON}	For FlashPro3 Programmer only.
flashpro3_ freq	Integer (Hertz)	For FlashPro3 Programmer only.
flashpro3_vpump	String {ON | OFF}	For Flash For FlashPro3 Programmer only.

For a detailed description of the parameters and values, refer to Programmer Settings.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

For FlashPro3 programmer

```
configure_tool -name {PROGRAMMER_INFO}\
```

```
  -params {flashpro3_clk_mode:free_running_clk}\
  -params {flashpro3_force_freq:OFF}\
  -params {flashpro3_freq:400000}\
  -params {flashpro3_vpump:ON}
```

For FlashPro4 programmer

```
configure_tool -name {PROGRAMMER_INFO}\
  -params {flashpro4_clk_mode:free_running_clk\}
  -params {flashpro4_force_freq:OFF}\
  -params {flashpro4_freq:400000}\
  -params {flashpro4_vpump:ON}
```

For FlashPro5 programmer

```
configure_tool -name {PROGRAMMER_INFO}\
  -params {flashpro5_clk_mode:free_running_clk}\
  -params {flashpro5_force_freq:OFF}\
  -params {flashpro5_freq:400000}\
  -params {flashpro5_vpump:ON}
```

For FlashPro6 programmer

```
configure_tool -name {PROGRAMMER_INFO}\
  -params {flashpro6_force_freq:OFF}\
  -params {flashpro6_freq:400000}
```

Return

Returns 0 on success and 1 on failure.

SPM

To configure security using Tcl, you must use the configure_tool Tcl command to pass the SPM configuration parameters.

```
configure_tool -name {SPM}
-params {name:value}
[-params {name:value}]+
```

configure_tool –name {SPM} parameter:value pair

Note: true | 1 will select the checkbox in the SPM UI

Name	Type	Value	Description
back_level_bypass	bool	false | true | 1 | 0	If true, will bypass the backlevel protection; Update Policy
back_level_protection	bool	false | true | 1 | 0	If true, will set back level protection; Update Policy
back_level_update_version	Integer	0 - 65535	Set back level version; Update Policy

Name	Type	Value	Description
debug_cortex_m3	bool	false | true | 1 | 0	If true, will disable Cortex M3 debug. This lock bit is protected by DPK; Debug Policy; SmartFusion2 only
debug_digest_request	bool	false | true | 1 | 0	If true, disables design digest check request via JTAG and SPI. Use FlashLock/UPK1 to allow digest check; Debug Policy
debug_disable_jtag	bool	false | true | 1 | 0	If true, disables JTAG (1149.1) test instructions (HIGHZ, EXTEST, INTEST, CLAMP, SAMPLE, and PRELOAD). I/Os will be tri-stated when in JTAG programming mode. Use FlashLock/UPK1 to unlock; Debug Policy
debug_passkey	hex	64 hex characters	Value of DPK; Debug Policy
debug_ujtag_access	bool	false | true | 1 | 0	If true, disables access to UJTAG. Use FDPK to unlock; Debug Policy
disable_user_encryption_key_1	bool	false | true | 1 | 0	If true, disables UEK1; Key Mode Policy
disable_user_encryption_key_2	bool	false | true | 1 | 0	If true, disables UEK2; Key Mode Policy
disable_user_encryption_key_3	bool	false | true | 1 | 0	If true, disables UEK3 Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices. All other devices will set this to false by default. See the SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide for details. Key Mode Policy
factory_access	string	Flashlock | permanent | open	Sets Microsemi factory test mode access level Open: All Microsemi factory test mode access without FlashLock/UPK1 FlashLock (default): Microsemi factory test mode is disabled. FlashLock/UPK1 is required to unlock. Permanent: Permanently disable Microsemi factory test mode access

Name	Type	Value	Description
iap_isp_services	bool	false | true | 1 | 0	If true, disables access to IAP/ISP services; Update Policy
security_key_mode	string	Default | custom	Default: Bitstream encrypted with default key. No security lock bits are set. Custom: Custom security settings. Allows user encryption keys, security policy settings, and Microsemi factory test mode access level.
smartdebug_access	string	Full | none	Full: SmartDebug has full access to debug features None: Disable read/write access to SmartDebug architecture. DPK is required for read/write access. Debug Policy
update_auto_prog_lock	bool	false | true | 1 | 0	If true, disables Auto Programming; Update Policy
update_envm_protection	bool	Passkey | open	Open: Updates to eNVM are allowed using UEK1 or UEK2; FlashLock/UPK1 is NOT required for updates Passkey: eNVM updates are disabled. Use FlashLock/UPK1 to unlock Write/Verify/Read operations. Update Policy
update_fabric_protection	bool	Passkey | open	Open: Updates to Fabric are allowed using UEK1 or UEK2; FlashLock/UPK1 is NOT required for updates Passkey: Fabric updates are disabled. Use FlashLock/UPK1 to unlock Erase/Write/Verify operations. Update Policy
update_jtag_lock	bool	false | true | 1 | 0	If true, disables access to JTAG programming. Use FlashLock/UPK1 to unlock; Update Policy
update_spi_slave_lock	bool	false | true | 1 | 0	If true, disables access to SPI Slave. Use FlashLock/UPK1 to unlock; Update Policy
use_debug_policy	bool	false | true | 1 | 0	If true, Debug Policy will be used

Name	Type	Value	Description
use_key_mode_policy	bool	false | true | 1 | 0	If true, Key Mode Policy will be used
use_update_policy	bool	false | true | 1 | 0	If true, Update Policy will be used
use_user_key_set_1	bool	false | true | 1 | 0	If true, enables User Key Set 1
use_user_key_set_2	bool	false | true | 1 | 0	If true, enables User Key Set 2
use_user_key_set_3	bool	false | true | 1 | 0	If true, enables User Key Set 3 Note: User Key Set 3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.
user_encryption_key_1	hex	64 hex characters	Value of UEK1
user_encryption_key_2	hex	64 hex characters	Value of UEK2
user_encryption_key_3	hex	64 hex characters	Value of UEK3 Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices. All other devices will set this to false by default. See the SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide for details.
user_passkey_1	hex	64 hex characters	Value of Flashlock/UPK1
user_passkey_2	hex	64 hex characters	Value of UPK2
user_security_policy_protection	string	Flashlock|permanent	Flashlock: User keys and Security policies will be protected from erase/write by FlashLock/UPK1 Permanent: Permanently protect UEK1, UEK2, Security Policies, and Microsemi factory test mode access level. NOTE: Once programmed, these settings cannot be changed.

Supported Families

SmartFusion2

IGLOO2

Example

```
configure_tool \
  -name {SPM} \
  -params {back_level_bypass:false} \
  -params {back_level_protection:false} \
  -params {back_level_update_version:} \
  -params {debug_cortex_m3:false} \
  -params {debug_digest_request:false} \
  -params {debug_disable_jtag:false} \
  -params {debug_passkey:} \
  -params {debug_ujtag_access:false} \
  -params {disable_user_encryption_key_1:false} \
  -params {disable_user_encryption_key_2:false} \
  -params {disable_user_encryption_key_3:false} \
  -params {factory_access:flashlock} \
  -params {iap_isp_services:true} \
  -params {security_key_mode:custom} \
  -params {smartdebug_access:full} \
  -params {update_auto_prog_lock:true} \
  -params {update_envm_protection:passkey} \
  -params {update_fabric_protection:passkey} \
  -params {update_jtag_lock:false} \
  -params {update_spi_slave_lock:false} \
  -params {use_debug_policy:false} \
  -params {use_key_mode_policy:false} \
  -params {use_update_policy:false} \
  -params {use_user_key_set_1:true} \
  -params {use_user_key_set_2:false} \
  -params {use_user_key_set_3:false} \
  -params
  {user_encryption_key_1:9E108123949848EC7453336DFBBC0CAE60C8541C2AFA7010FA209F7396F3EA17}
  \
  -params {user_encryption_key_2:} \
  -params {user_encryption_key_3:} \
  -params
  {user_passkey_1:B52EED23B1C4C5BAE1384791CE4F7A069D940A6933329A0A9CE5B24E21C13D39} \
  -params {user_passkey_2:} \
  -params {user_security_policy_protection:flashlock}
```

Return

Returns 0 on success and 1 on failure.

SYNTHESIZE

SYNTHESIZE is a command tool used in configure_tool and run_tool. Configure_tool is a general-purpose Tcl command that allows you to configure a tool's parameters and values prior to executing the tool. The run_tool Tcl command then executes the specified tool with the configured parameters.

To synthesize your design in Libero SoC, you first configure the synthesize tool with the configure_tool command and then execute the command with the run_tool command.

```
configure_tool -name {SYNTHESIZE}
    -params {name:value}
    [-params {name:value}]
```

```
run_tool -name {SYNTHESIZE}
```

The following tables list the parameter names and values.

configure_tool –name {SYNTHESIZE} parameter:value pair

Name	Value	Description
CLOCK_ASYNC	**Integer**	**Specifies the threshold value for asynchronous pin promotion to a global net. The default is 12.**
CLOCK_GLOBAL	Integer	Specifies the threshold value for Clock pin promotion. The default is 2.
CLOCK_DATA	Integer value between 1000 and 200,000.	Specifies the threshold value for data pin promotion. The default is 5000.
RAM_OPTIMIZED_FOR_POWER	Boolean {true | false | 1 | 0}	Set to true or 1 to optimize RAM for Low Power; RAMS are inferred and configured to ensure the lowest power consumption. Set to false or 0 to optimize RAM for High Speed at the expense of more FPGA resources.
RETIMING	Boolean {true | false | 1 | 0}	Set to true or 1 to enable Retiming during synthesis. Set to false or 0 to disable Retiming during synthesis.
SYNPLIFY_OPTIONS	String	Specifies additional synthesis-specific options. Options specified by this parameter override the same options specified in the user Tcl file if there is a conflict.
SYNPLIFY_TCL_FILE	String	Specifies the absolute or relative path name to the user Tcl file containing synthesis-specific options.
BLOCK_MODE	Boolean {true | false | 1 | 0}	Set to true or 1 when you have blocks in your design and you want to enable the Block mode. Set it to false or 0 if you don't have blocks in your design. Default is false or 0.
BLOCK_PLACEMENT_CONFLICTS	String {ERROR|KEEP|LOCK|DISCARD}	Instructs the COMPILE engine what to do when the software encounters a placement conflict. When set to: ERROR - Compile errors out if any instance from a Designer block becomes unplaced. This is the default. KEEP - If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked. Therefore, the placer can move them into another location if necessary. LOCK - If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved and locked. DISCARD – Discards any placement from the block, even if there are no conflicts.

Name	Value	Description
BLOCK_ROUTING_CONFLICTS	String {ERROR|KEEP|LOCK|DISCARD}	Instructs the COMPILE engine what to do when the software encounters a routing conflict. When set to: ERROR - Compile errors out if any route in any preserved net from a Designer block is deleted. This is the default. KEEP – If a route is removed from a net for any reason, the routing for the non-conflicting nets is kept unlocked. The router can re-route these nets. LOCK – If routing is removed from a net for any reason, the routing for the non-conflicting nets is kept as locked, and the router will not change them. DISCARD - Discards any routing from the block, even if there are no conflicts.
PA4_GB_COUNT	Integer	The number of available global nets is reported. Minimum for all dies is “0”. Default and Maximum values are die-dependent: 005/010 die: Default = Max = 8 025/050/060/090/150 die: Default=Max=16 RT4G075/RT4G150: Default=24, Max=48. **Note: For RTG4, default is 48.**
PA4_GB_MAX_RCLKINT_INSERTION	Integer	Specifies the maximum number of global nets that could be demoted to row-globals. Default is 16, Min is 0 and Max is 50.
PA4_GB_MIN_GB_FANOUT_TO_USE_RCLKINT	Integer	Specifies the Minimum fanout of global nets that could be demoted to row-globals. Default is 300. Min is 25 and Max is 5000.
LANGUAGE_SYSTEM_VLOG	Boolean {true | false}	Set to true if the Verilog files contain System Verilog constructs.
LANGUAGE_VERILOG_2001	Boolean {true | false}	Set to true if Verilog files contain Verilog 2001 constructs.

run_tool –name {SYNTHESIZE}

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
configure_tool -name {SYNTHESIZE} -params {BLOCK_MODE:false}\
     -params {BLOCK_PLACEMENT_CONFLICTS:ERROR} -params\
     {BLOCK_ROUTING_CONFLICTS:ERROR} -params {CLOCK_ASYNC:12}\
     -params {CLOCK_DATA:5010} -params {CLOCK_GLOBAL:2} -params\
     -params {PA4_GB_MAX_RCLKINT_INSERTION:16} -params\
     {PA4_GB_MIN_GB_FANOUT_TO_USE_RCLKINT:299} -params\
```

```
{RAM_OPTIMIZED_FOR_POWER:false} -params {RETIMING:false}
 -params {SYNPLIFY_OPTIONS:
set_option -run_prop_extract 1;
set_option -maxfan 10000;
set_option -clock_globalthreshold 2;
set_option -async_globalthreshold 12;
set_option -globalthreshold 5000;
set_option -low_power_ram_decomp 0;}\
-params {SYNPLIFY_TCL_FILE:C:/Users/user1/Desktop/tclflow/synthesis/test.tcl}

run_tool -name {SYNTHESIZE} #Takes no parameters
```

Return

`configure_tool -name {SYNTHESIZE}`

Returns 0 on success and 1 on failure.

`run_tool -name {SYNTHESIZE}`

Returns 0 on success and 1 on failure.

USER_PROG_DATA (SmartFusion2, IGLOO2)

USER_PROG_DATA is a command tool used in configure_tool. Configure_tool -name {USER_PROG_DATA} sets the Design Version and Silicon Signature in your device.

```
configure_tool -name {USER_PROG_DATA}
-params {name:value}
-params {name:value}
```

The following table lists the parameter names and values.

configure_tool –name {USER_PROG_DATA} parameter:value pair

Name	Value	Description
design_version	Integer (0 through 65535)	Sets the design version. It must be greater than the Back level version in SPM Update Policy.
silicon_signature	Hex {<max length 8 Hex characters>}	32-bit (8 hex characters) silicon signature to be programmed into the device. This field can be read from the device using the JTAG USERCODE instruction.

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {USER_PROG_DATA}\
  -params {design_version:255}\
  -params {silicon_signature:abcdffff}
```

Return

Returns 0 on success and 1 on failure.

VERIFYPOWER

VERIFYPOWER is a command tool used in run_tool. The command run_tool passes a script file that contains power-specific Tcl commands to the VERIFYPOWER command and executes it.

```
run_tool -name {VERIFYPOWER} -script {power_analysis.tcl}
```

where

<power_analysis.tcl> is a script that contains power-specific Tcl commands. You can include power-specific Tcl commands to generate power reports. See the sample power_analysis Tcl Script below for details.

Return

Returns 0 on success and 1 on failure.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
run_tool -name {VERIFYPOWER} -script {<power_analysis.tcl>}
```

Sample power_analysis Tcl Script <power_analysis.tcl>

The following example changes SmartPower operating condition settings from the default to 40C junction temperature and 1.25V VDD.

It then creates a report called A4P5000_uSRAM_POWER_64X18_power_report.txt.

Change from pre-defined temperature and voltage mode (COM,IND,MIL) to SmartPower custom

```
smartpower_set_temperature_opcond -use "design"
smartpower_set_voltage_opcond -voltage "VDD" -use "design"
```

Set the custom temperature to 40C ambient temperature.

```
smartpower_temperature_opcond_set_design_wide -typical 40 -best 40 -worst 40
```

Set the custom voltage to 1.25V

```
smartpower_voltage_opcond_set_design_wide -voltage "VDD" -typical 1.25 -best 1.25 -worst
1.25
```

VERIFYTIMING

VERIFYTIMING is a command tool used in run_tool. Run_tool passes a script file that contains timing-specific Tcl commands to the VERIFYTIMING command and executes it.

```
run_tool -name {VERIFYTIMING} -script {timing.tcl}
```

where

<timing.tcl> is a script that contains SmartTime-specific Tcl commands. You can include SmartTime-specific Tcl commands to create user path sets and to generate timing reports. See sample the Sample SmartTime Tcl Script below for details.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
run_tool -name {VERIFYTIMING} -script {<timing.tcl>}
```

Return

Returns 0 on success and 1 on failure.

Sample SmartTime Tcl Script <*timing.tcl*>

```
# Create user path set -from B_reg
create_set -name from_B_reg \
  -source {B_reg*[*]:CLK} \
  -sink {*}
# Create user set -from A, B, C
create_set -name from_in_ports \
  -source {A B C} \
  -sink {*}
# Generate Timing Reports
report \
  -type timing \
  -analysis min \
  -format text \
  -max_paths 10 \
  -print_paths yes \
  -max_expanded_paths 10 \
  -include_user_sets yes \
  min_timing.rpt
# Export SDC
write_sdc -scenario {Primary} exported.sdc
#save the changes
save
```

SmartTime Tcl Commands

all_inputs

Tcl command; returns an object representing all input and inout pins in the current design.

```
all_inputs
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay -from [all_inputs] -to [get_clocks ck1]
```

See Also

Tcl documentation conventions

all_outputs

Tcl command; returns an object representing all output and inout pins in the current design.

```
all_outputs
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay -from [all_inputs] -to [all_outputs]
```

See Also

Tcl documentation conventions

all_registers

Tcl command; returns an object representing register pins or cells in the current scenario based on the given parameters.

```
all_registers [-clock clock_name]
[-async_pins][-output_pins][-data_pins][-clock_pins]
```

Arguments

-clock *clock_name*

Specifies the name of the clock domain to which the registers belong. If no clock is specified, all registers in the design will be targeted.

-async_pins

Lists all register pins that are async pins for the specified clock (or all registers asynchronous pins in the design).

-output_pins

Lists all register pins that are output pins for the specified clock (or all registers output pins in the design).

-data_pins

Lists all register pins that are data pins for the specified clock (or all registers data pins in the design).

-clock_pins

Lists all register pins that are data pins for the specified clock (or all registers clock pins in the design).

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2.000 -from { ff_m:CLK ff_s2:CLK } -to [all_registers -clock_pins -clock {
ff_m:Q }]
```

See Also

Tcl documentation conventions

check_constraints

Tcl command; checks all timing constraints in the current scenario for validity. This command performs the same checks as when the constraint is entered through SDC or Tcl.

```
check_constraints
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
check_constraints
```

clone_scenario (SmartFusion2, IGLOO2, and RTG4)

Tcl command; creates a new timing scenario by duplicating an existing one. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
clone_scenario original  new_scenario_name
```

Arguments

original

Specifies the name of the source timing scenario to clone (copy). The source must be a valid, existing timing scenario.

new_scenario_name

Specifies the name of the new scenario to be created.

Supported Families

SmartFusion2
IGLOO2
RTG4

Description

This command creates a timing scenario with the *new_scenario_name*, which includes a copy of all constraints in the original scenario. The new scenario is then added to the list of scenarios.

Example

```
clone_scenario primary my_new_scenario
```

See Also

create_scenario
delete_scenario
Tcl documentation conventions

create_clock

Tcl command; creates a clock constraint on the specified ports/pins, or a virtual clock if no source other than a name is specified.

```
create_clock [-name clock_name] -period period_value
[-waveform> edge_list][source_objects]
```

Arguments

-period *period_value*

Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The period_value must be greater than zero.

`-name clock_name`

Specifies the name of the clock constraint. You must specify either a clock name or a source.

`-waveform edge_list`

Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify -waveform option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant (period_value/2)ns.

source_objects

Specifies the source of the clock constraint. The source can be ports, pins, or nets in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. You must specify either a source or a clock name.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates two clocks, one on port CK1 with a period of 6, and the other on port CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} -period 6 -waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```
create_clock -period 7 -waveform {2 4} [get_ports {CK3}]
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

create_generated_clock

Tcl command; creates an internally generated clock constraint on the ports/pins and defines its characteristics.

```
create_generated_clock [-name clock_name] -source reference_pin [-divide_by divide_factor] [-
multiply_by multiply_factor] [-invert] source[-edges values] [-edge_shift values]
```

Arguments

`-name clock_name`

Specifies the name of the clock constraint.

`-source reference_pin`

Specifies the reference pin in the design from which the clock waveform is to be derived.

`-divide_by` *`divide_factor`*

Specifies the frequency division factor. For instance if the *divide_factor* is equal to 2, the generated clock period is twice the reference clock period.

`-multiply_by` *`multiply_factor`*

Specifies the frequency multiplication factor. For instance if the *multiply_factor* is equal to 2, the generated clock period is half the reference clock period.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`source`

Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

`-edges` *`values`*

Specify the integer values that represent the edges from the source clock that form the edges of the generated clock. Three values must be specified to generate the clock. If you specify less than three, a tool tip indicates an error.

`-edge_shift` *`values`*

Specify a list of three floating point numbers that represents the amount of shift, in nanoseconds, that the specified edges are to undergo to yield the final generated clock waveform. These floating point values can be positive or negative. Positive value indicates a shift later in time, while negative indicates a shift earlier in time.

For example: An edge shift of {1 1 1} on the LSB generated clock, would shift each derived edge by 1 nanosecond.

To create a 200MHz clock from a 100MHz clock, use edge { 1 2 3} and edge shift {0 -2.5 -5.0}.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```
create_generated_clock -name {my_user_clock} -divide_by 2 -source [get_ports
{CLK}]  U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```
create_generated_clock -divide_by 3 -multiply_by 4  -source clk [get_pins {myPLL:CLK1}]
```

See Also

`create_clock`

Tcl Command Documentation Conventions

create_scenario

Tcl command; creates a new timing scenario with the specified name. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
create_scenario name
```

Arguments

name

Specifies the name of the new timing scenario.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

A timing scenario is a set of timing constraints used with a design. Scenarios enable you to easily refine the set of timing constraints used for Timing-Driven Place-and-Route, so as to achieve timing closure more rapidly.

This command creates an empty timing scenario with the specified name and adds it to the list of scenarios.

Example

```
create_scenario scenario_A
```

See Also

clone_scenario

Tcl Command Documentation Conventions

create_set

Tcl command; creates a set of paths to be analyzed. Use the arguments to specify which paths to include. To create a set that is a subset of a clock domain, specify it with the `-clock` and `-type` arguments. To create a set that is a subset of an inter-clock domain set, specify it with the `-source_clock` and `-sink_clock` arguments. To create a set that is a subset (filter) of an existing named set, specify the set to be filtered with the `-parent_set` argument.

```
create_set\ -name <name>\ -parent_set <name>\ -type <set_type>\ -clock <clock name>\ -
source_clock <clock name>\ -sink_clock <clock name>\ -in_to_out\ -source <port/pin pattern>\
-sink <port/pin pattern>
```

Arguments

`-name <name>`

Specifies a unique name for the newly created path set.

`-parent_set <name>`

Specifies the name of the set to filter from.

`-clock <clock_name>`

Specifies that the set is to be a subset of the given clock domain. This argument is valid only if you also specify the `-type` argument.

`-type <value>`

Specifies the predefined set type on which to base the new path set. You can only use this argument with the `-clock` argument, not by itself.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_async	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
external_hold	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-in_to_out`

Specifies that the set is based on the “Input to Output” set, which includes paths that start at input ports and end at output ports.

`-source_clock <clock_name>`

Specifies that the set will be a subset of an inter-clock domain set with the given source clock. You can only use this option with the `-sink_clock` argument.

`-sink_clock <clock_name>`

Specifies that the set will be a subset of an inter-clock domain set with the given sink clock. You can only use this option with the `-source_clock` argument.

`-source <port/pin_pattern>`

Specifies a filter on the source pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

`-sink <port/pin_pattern>`

Specifies a filter on the sink pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

```
create_set -name { my_user_set } -source { C* } -sink { D* }
create_set -name { my_other_user_set } -parent_set { my_user_set } -source { CL* }
create_set -name { adder } -source { ALU_CLOCK } -type { REG_TO_REG } -sink { ADDER*}
create_set -name { another_set } -source_clock { EXTERN_CLOCK } -sink_clock {
MY_GEN_CLOCK }
```

expand_path

Tcl command; displays expanded path information (path details) for paths. The paths to be expanded are identified by the parameters required to display these paths with list_paths. For example, to expand the first path

listed with list_paths -clock {MYCLOCK} -type {register_to_register}, use the command expand_path -clock {MYCLOCK} -type {register_to_register}. Path details contain the pin name, type, net name, cell name, operation, delay, total delay, and edge as well as the arrival time, required time, and slack. These details are the same as details available in the SmartTime Expanded Path window.

```
expand_path
-index value
-set name
-clock clock name
-type set_type
-analysis {max| min}
-format {csv | text}
-from_clock clock name
-to_clock clock name
```

Arguments

`-index value`

Specify the index of the path to be expanded in the list of paths. Default is 1.

`-analysis {max | min}`

Specify whether the timing analysis is done is max-delay (setup check) or min-delay (hold check). Valid values: max or min.

`-format {csv | text}`

Specify the list format. It can be either text (default) or csv (comma separated values). The former is suited for display the latter for parsing.

`-set name`

Displays a list of paths from the named set. You can either use the -set option to specify a user set by its name or use both -clock and -type to specify a set.

`-clock clock name`

Displays the set of paths belonging to the specified clock domain. You can either use this option along with -type to specify a set or use the -set option to specify the name of the set to display.

`-type set_type`

Specifies the type of paths in the clock domain to display in a list. You can only use this option with the -clock option. You can either use this option along with -clock to specify a set or use the -set option to specify a set name.

Value	Description
reg_to_reg	Paths between registers in the design
external_setup	Path from input ports to registers
external_hold	Path from input ports to registers
clock_to_out	Path from registers to output ports
reg_to_async	Path from registers to asynchronous pins
external_recovery	Set of paths from inputs to asynchronous pins
external_removal	Set of paths from inputs to asynchronous pins
async_to_reg	Path from asynchronous pins to registers

`-from_clock clock_name`

Displays a list of timing paths for an inter-clock domain set belonging to the source clock specified. You can only use this option with the -to_clock option, not by itself.

`-to_clock clock_name`

Displays a list of timing paths for an inter-clock domain set belonging to the sink clock specified. You can only use this option with the -from_clock option, not by itself.

`-analysis name`

Specifies the analysis for the paths to be listed. The following table shows the acceptable values for this argument.

Value	Description
maxdelay	Maximum delay analysis
mindelay	Minimum delay analysis

`-index list_of_indices`

Specifies which paths to display. The index starts at 1 and defaults to 1. Only values lower than the max_paths option will be expanded.

`-format value`

Specifies the file format of the output. The following table shows the acceptable values for this argument:

Value	Description
text	ASCII text format
csv	Comma separated value file format

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

Note: The following example returns a list of five paths:

```
puts [expand_path -clock { myclock } -type {reg_to_reg }]
puts [expand_path -clock {myclock} -type {reg_to_reg} -index { 1 2 3 } -format text]
```

See Also

list_paths

get_cells

Tcl command; returns an object representing the cells (instances) that match those specified in the pattern argument.

```
get_cells pattern
```

Arguments

`pattern`

Specifies the pattern to match the instances to return. For example, "get_cells U18*" returns all instances starting with the characters "U18", where “*” is a wildcard that represents any character string.

Supported Families

SmartFusion2
IGLOO2
RTG4

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path -through [get_cells {Rblock/muxA}]
```

See Also

get_clocks
get_nets
get_pins
get_ports
Tcl Command Documentation Conventions

get_clocks

Tcl command; returns an object representing the clock(s) that match those specified in the pattern argument in the current timing scenario.

```
get_clocks pattern
```

Arguments

pattern

Specifies the pattern to use to match the clocks set in SmartTime.

Supported Families

SmartFusion2
IGLOO2
RTG4

Description

- If this command is used as a –from argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the synchronous pins of all the registers related to this clock are used as path endpoints.

Example

```
set_max_delay -from [get_ports datal] -to \
[get_clocks ck1]
```

See Also

create_clock
create_generated_clock
Tcl Command Documentation Conventions

get_current_scenario

Tcl command; returns the name of the current timing scenario.

```
get_current_scenario
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
get_current_scenario
```

See Also

set_current_scenario
Tcl documentation conventions

get_nets

Tcl command; returns an object representing the nets that match those specified in the pattern argument.

```
get_nets pattern
```

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where “*” is a wildcard that represents any character string.

Supported Families

SmartFusion2
IGLOO2
RTG4

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock (create_clock) or create generated clock (create_generated_clock) constraints and as `-through` arguments in the set false path, set minimum delay, set maximum delay, and set multicycle path constraints.

Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
```

```
set_false_path -through [get_nets {Tblk/rm/n*}]
create_clock -name mainCLK -period 2.5 [get_nets {cknet}]
```

See Also

create_clock
create_generated_clock
set_false_path
set_min_delay
set_max_delay
set_multicycle_path
Tcl documentation conventions

get_pins

Tcl command; returns an object representing the pin(s) that match those specified in the pattern argument.

```
get_pins pattern
```

Arguments

pattern

Specifies the pattern to match the pins to return. For example, "get_pins clock_gen*" returns all pins starting with the characters "clock_gen", where “*” is a wildcard that represents any character string.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

See Also

create_clock
create_generated_clock
set_clock_latency
set_false_path
set_min_delay
set_max_delay
set_multicycle_path
Tcl documentation conventions

get_ports

Tcl command; returns an object representing the port(s) that match those specified in the pattern argument.

```
get_portspattern
```

Argument

pattern

Specifies the pattern to match the ports.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
create_clock -period 10 [get_ports CK1]
```

See Also

create_clock
set_clock_latency
set_input_delay
set_output_delay
set_min_delay
set_max_delay
set_false_path
set_multicycle_path
Tcl documentation conventions

list_clock_groups

This Tcl command lists all existing clock groups in the design.

```
list_clock_groups
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
list_clock_groups
```

See Also

set_clock_groups
remove_clock_groups

list_clock_latencies

Tcl command; returns details about all of the clock latencies in the current timing constraint scenario.

```
list_clock_latencies
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clock_latencies]
```

See Also

set_clock_latency
remove_clock_latency
Tcl documentation conventions

list_clock_uncertainties

Tcl command; returns details about all of the clock uncertainties in the current timing constraint scenario.

```
list_clock_uncertainties
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
list_clock_uncertainties
```

See Also

set_clock_uncertainty
remove_clock_uncertainty

list_clocks

Tcl command; returns details about all of the clock constraints in the current timing constraint scenario.

```
list_clocks
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_clocks]
```

See Also

create_clock
remove_clock
Tcl documentation conventions

list_disable_timings

Tcl command; returns the list of disable timing constraints for the current scenario.

```
list_disable_timings
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
list_disable_timings
```

list_false_paths

Tcl command; returns details about all of the false paths in the current timing constraint scenario.

```
list_false_paths
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_false_paths]
```

See Also

set_false_path
remove_false_path
Tcl documentation conventions

list_generated_clocks

Tcl command; returns details about all of the generated clock constraints in the current timing constraint scenario.

```
list_generated_clocks
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_generated_clocks]
```

See Also

create_generated_clock
remove_generated_clock
Tcl documentation conventions

list_input_delays

Tcl command; returns details about all of the input delay constraints in the current timing constraint scenario.

```
list_input_delays
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_input_delays]
```

See Also

set_input_delay
remove_input_delay
Tcl documentation conventions

list_max_delays

Tcl command; returns details about all of the maximum delay constraints in the current timing constraint scenario.

```
list_max_delays
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_max_delays]
```

See Also

set_max_delay
remove_max_delay
Tcl documentation conventions

list_min_delays

Tcl command; returns details about all of the minimum delay constraints in the current timing constraint scenario.

```
list_min_delays
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_min_delays]
```

See Also

set_min_delay
remove_min_delay
Tcl documentation conventions

list_multicycle_paths

Tcl command; returns details about all of the multicycle paths in the current timing constraint scenario.

```
list_multicycle_paths
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_multicycle_paths]
```

See Also

set_multicycle_path
remove_multicycle_path
Tcl documentation conventions

list_objects

Tcl command; returns a list of object matching the parameter. Objects can be nets, pins, ports, clocks or instances.

```
list_objects <object>
```

Arguments

Any timing constraint parameter.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

The following example lists all the inputs in your design:

```
list_objects [all_inputs]
```

You can also use wildcards to filter your list, as in the following command:

```
list_objects [get_ports a*]
```

See Also

Tcl documentation conventions

list_output_delays

Tcl command; returns details about all of the output delay constraints in the current timing constraint scenario.

```
list_output_delays
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
puts [list_output_delays]
```

See Also

set_output_delay

remove_output_delay

Tcl documentation conventions

list_paths

Tcl command; returns a list of the *n* worst paths matching the arguments. The number of paths returned can be changed using the set_options -limit_max_paths <*value*> command.

```
list_paths
-analysis <max | min>
-format <csv | text>
-set <name>
-clock <clock name>
-type <set_type>
-from_clock <clock name>
-to_clock <clock name>
-in_to_out
-from <port/pin pattern>
-to <port/pin pattern>
```

Arguments

-analysis <*max* | *min*>

Specifies whether the timing analysis is done for max-delay (setup check) or min-delay (hold check). Valid values are: max or min.

-format < *text* | *csv* >

Specifies the list format. It can be either text (default) or csv (comma separated values). Text format is better for display and csv format is better for parsing.

-set <*name*>

Returns a list of paths from the named set. You can either use the -set option to specify a user set by its name or use both -clock and -type to specify a set.

-clock <*clock name*>

Returns a list of paths from the specified clock domain. This option requires the -type option.

-type <*set_type*>

Specifies the type of paths to be included. It can only be used along with -clock. Valid values are:

reg_to_reg -- Paths between registers

external_setup -- Path from input ports to data pins of registers

external_hold -- Path from input ports to data pins of registers

clock_to_out -- Path from registers to output ports

reg_to_async -- Path from registers to asynchronous pins of registers

external_recovery -- Path from input ports to asynchronous pins of registers

external_removal -- Path from input ports to asynchronous pins of registers

async_to_reg -- Path from asynchronous pins to registers

-from_clock <*clock name*>

Used along with -to_clock to get the list of paths of the inter-clock domain between the two clocks.

-to_clock <*clock name*>

Used along with -from_clock to get the list of paths of the inter-clock domain between the two clocks.

-in_to_out

Used to get the list of path between input and output ports.

-from <*port/pin pattern*>

Filter the list of paths to those starting from ports or pins matching the pattern.

`-to <port/pin pattern>`

Filter the list of paths to those ending at ports or pins matching the pattern.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

The following command displays the list of register to register paths of clock domain clk1:

```
puts [ list_paths -clock clk1 -type reg_to_reg ]
```

See Also

create_set

expand_path

set_options

list_scenarios

Tcl command; returns a list of names of all of the available timing scenarios.

```
list_scenarios
```

Arguments

None

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

```
list_scenarios
```

See Also

get_current_scenario

Tcl documentation conventions

read_sdc

The read_sdc Tcl command evaluate an SDC file, adding all constraints to the specified scenario (or the current/default one if none is specified). Existing constraints are removed if -add is not specified.

```
read_sdc
-add
-scenario scenario_name
-netlist (user | optimized)
-pin_separator (: | /)
file name
```

Arguments

`-add`

Specifies that the constraints from the SDC file will be added on top of the existing ones, overriding them in case of a conflict. If not used, the existing constraints are removed before the SDC file is read.

`-scenario` *scenario_name*

Specifies the scenario to add the constraints to. The scenario is created if none exists with this name.

`-netlist (user | optimized)`

Specifies whether the SDC file contains object defined at the post-synthesis netlist (user) level or physical (optimized) netlist (used for timing analysis).

`-pin_separator` *sep*

Specify the pin separator used in the SDC file. It can be either ':' or '/'.

file name

Specify the SDC file name.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

The following command removes all constraints from the current/default scenario and adds all constraints from design.sdc file to it:

```
read_sdc design.sdc
```

See Also

write_sdc

remove_all_constraints

Tcl command; removes all timing constraints from analysis.

```
remove_all_constraints
```

Arguments

None

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
remove_all_constraints
```

See Also

remove_scenario

remove_clock

Tcl command; removes the specified clock constraint from the current timing scenario.

```
remove_clock -name clock_name| -id constraint_ID
```

Arguments

`-name clock_name`

Specifies the name of the clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes the specified clock constraint from the current scenario. If the specified name does not match a clock constraint in the current scenario, or if the specified ID does not refer to a clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock constraint named "my_user_clock":

```
remove_clock -name my_user_clock
```

The following example removes the clock constraint using its ID:

```
set clockId [create_clock -name my_user_clock -period 2]
remove_clock -id $clockId
```

See Also

create_clock

Tcl Command Documentation Conventions

remove_clock_groups

This Tcl command removes a clock group by name or by ID.

```
remove_clock_groups [-id id# | -name groupname] \
[-physically_exclusive | -logically_exclusive | -asynchronous]
```

Note: The exclusive flag is not needed when removing a clock group by ID.

Arguments

`-id id#`

Specifies the clock group by the ID.

`-name groupname`

Specifies the clock group by name (to be always followed by the exclusive flag).

`[-physically_exclusive | -logically_exclusive | - asynchronous]`

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

Removal by group name

```
remove_clock_groups -name mygroup3 -physically_exclusive
```

Removal by goup ID

```
remove_clock_groups -id 12
```

See Also

set_clock_groups

list_clock_groups

remove_clock_latency

Tcl command; removes a clock source latency from the specified clock and from all edges of the clock.

```
remove_clock_latency {-source clock_name_or_source |-id constraint_ID}
```

Arguments

`-source clock_name_or_source`

Specifies either the clock name or source name of the clock constraint from which to remove the clock source latency. You must specify either a clock or source name or its constraint ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either a clock or source name or its constraint ID.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes a clock source latency from the specified clock in the current scenario. If the specified source does not match a clock with a latency constraint in the current scenario, or if the specified ID does not refer to a clock with a latency constraint, this command fails.

Do not specify both the source and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock source latency from the specified clock.

```
remove_clock_latency -source my_clock
```

See Also

set_clock_latency

Tcl Command Documentation Conventions

remove_clock_uncertainty

Tcl command; removes a clock-to-clock uncertainty from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_clock_uncertainty -from | -rise_from | -fall_from from_clock_list -to | -rise_to| -fall_to to_clock_list -setup {value} -hold {value}
remove_clock_uncertainty -id constraint_ID
```

Arguments

-from

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-rise_from

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-fall_from

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

from_clock_list

Specifies the list of clock names as the uncertainty source.

-to

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the -to, -rise_to , or -fall_to arguments can be specified for the constraint to be valid.

-rise_to

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the -to, -rise_to , or -fall_to arguments can be specified for the constraint to be valid.

-fall_to

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the -to, -rise_to , or -fall_to arguments can be specified for the constraint to be valid.

to_clock_list

Specifies the list of clock names as the uncertainty destination.

-setup

Specifies that the uncertainty applies only to setup checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

-hold

Specifies that the uncertainty applies only to hold checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

-id *constraint_ID*

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either the exact parameters to set the constraint or its constraint ID.

Supported Families

SmartFusion2

IGLOO2
RTG4

Description

Removes a clock-to-clock uncertainty from the specified clock in the current scenario. If the specified arguments do not match clocks with an uncertainty constraint in the current scenario, or if the specified ID does not refer to a clock-to-clock uncertainty constraint, this command fails.

Do not specify both the exact arguments and the ID.

Examples

```
remove_clock_uncertainty -from Clk1 -to Clk2
remove_clock_uncertainty -from Clk1 -fall_to { Clk2 Clk3 } -setup
remove_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
remove_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 } -setup
remove_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
remove_clock_uncertainty -id $clockId
```

See Also

remove_clock
remove_generated_clock
set_clock_uncertainty

remove_disable_timing

Tcl command; removes a disable timing constraint by specifying its arguments, or its ID. If the arguments do not match a disable timing constraint, or if the ID does not refer to a disable timing constraint, the command fails.

```
remove_disable_timing -from value -to value name -id name
```

Arguments

-from *from_port*

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

-to *to_port*

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

name

Specifies the cell name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

-id *name*

Specifies the constraint name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
remove_disable_timing -from port1 -to port2 -id new_constraint
```

remove_false_path

Tcl command; removes a false path from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_false_path [-from from_list] [-to to_list] [-through through_list] [-id constraint_ID]
remove_false_path -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the false path constraint to remove from the current scenario. You must specify either the exact false path to remove or the constraint ID that refers to the false path constraint to remove.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes a false path from the specified clock in the current scenario. If the arguments do not match a false path constraint in the current scenario, or if the specified ID does not refer to a false path constraint, this command fails.

Do not specify both the false path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command such as get_pins or get_ports.

Examples

The following example specifies all false paths to remove:

```
remove_false_path -through U0/U1:Y
```

The following example removes the false path constraint using its id:

```
set fpId [set_false_path -from [get_clocks c*] -through {topx/reg/*} -to [get_ports
out15] ]
remove_false_path -id $fpId
```

See Also

set_false_path

Tcl Command Documentation Conventions

remove_generated_clock

Tcl command; removes the specified generated clock constraint from the current scenario.

```
remove_generated_clock {-name clock_name | -id constraint_ID }
```

Arguments

`-name clock_name`

Specifies the name of the generated clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the generated clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes the specified generated clock constraint from the current scenario. If the specified name does not match a generated clock constraint in the current scenario, or if the specified ID does not refer to a generated clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a generated clock name.

Examples

The following example removes the generated clock constraint named "my_user_clock":

```
remove_generated_clock -name my_user_clock
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

remove_input_delay

Tcl command; removes an input delay a clock on a port by specifying both the clocks and port names or the ID of the input_delay constraint to remove.

```
remove_input_delay -clock clock_name port_pin_list
remove_input_delay -id constraint_ID
```

Arguments

`-clock clock_name`

Specifies the clock name to which the specified input delay value is assigned.

`port_pin_list`

Specifies the port names to which the specified input delay value is assigned.

`-id constraint_ID`

Specifies the ID of the clock with the input_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the input_delay constraint ID .

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes an input delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an input delay constraint in the current scenario, or if the specified ID does not refer to an input delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the input delay from CLK1 on port data1:

```
remove_input_delay -clock [get_clocks CLK1] [get_ports data1]
```

See Also

set_input_delay

Tcl Command Documentation Conventions

remove_max_delay

Tcl command; removes a maximum delay constraint from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_max_delay [-from from_list] [-to to_list] [-through through_list]
remove_max_delay -id constraint_ID
```

Arguments

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-through *through_list*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

-to to_list

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

-id *constraint_ID*

Specifies the ID of the maximum delay constraint to remove from the current scenario. You must specify either the exact maximum delay arguments to remove or the constraint ID that refers to the maximum delay constraint to remove.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes a maximum delay value from the specified clock in the current scenario. If the arguments do not match a maximum delay constraint in the current scenario, or if the specified ID does not refer to a maximum delay constraint, this command fails.

Do not specify both the maximum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command.

Examples

The following example specifies a range of maximum delay constraints to remove:

```
remove_max_delay -through U0/U1:Y
```

See Also

set_max_delay

Tcl Command Documentation Conventions

remove_min_delay

Tcl command; removes a minimum delay constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_min_delay [-from from_list] [-to to_list] [-through through_list]
remove_min_delay -id constraint_ID
```

Arguments

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-through *through_list*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

-to*to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

-id *constraint_ID*

Specifies the ID of the minimum delay constraint to remove from the current scenario. You must specify either the exact minimum delay arguments to remove or the constraint ID that refers to the minimum delay constraint to remove.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes a minimum delay value from the specified clock in the current scenario. If the arguments do not match a minimum delay constraint in the current scenario, or if the specified ID does not refer to a minimum delay constraint, this command fails.

Do not specify both the minimum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example specifies a range of minimum delay constraints to remove:

```
remove_min_delay -through U0/U1:Y
```

See Also

set_min_delay

Tcl Command Documentation Conventions

remove_multicycle_path

Tcl command; removes a multicycle path constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_multicycle_path [-from from_list] [-to to_list] [-through through_list]
remove multicycle_path -id constraint_ID
```

Arguments

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-through *through_list*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

-to*to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

-id *constraint_ID*

Specifies the ID of the multicycle path constraint to remove from the current scenario. You must specify either the exact multicycle path arguments to remove or the constraint ID that refers to the multicycle path constraint to remove.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes a multicycle path from the specified clock in the current scenario. If the arguments do not match a multicycle path constraint in the current scenario, or if the specified ID does not refer to a multicycle path constraint, this command fails.

Do not specify both the multicycle path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes all paths between reg1 and reg2 to 3 cycles for setup check.

```
remove_multicycle_path -from [get_pins {reg1}] -to [get_pins {reg2}]
```

See Also

set_multicycle_path

Tcl Command Documentation Conventions

remove_output_delay

Tcl command; removes an ouput delay by specifying both the clocks and port names or the ID of the output_delay constraint to remove.

```
remove_output_delay -clock clock_name port_pin_list
remove_output_delay -id constraint_ID
```

Arguments

-clock *clock_name*

Specifies the clock name to which the specified output delay value is assigned.

port_pin_list

Specifies the port names to which the specified output delay value is assigned.

-id *constraint_ID*

Specifies the ID of the clock with the output_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the output_delay constraint ID .

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Removes an output delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an output delay constraint in the current scenario, or if the specified ID does not refer to an output delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the output delay from CLK1 on port out1:

```
remove_output_delay -clock [get_clocks CLK1] [get_ports out1]
```

See Also

set_output_delay

Tcl Command Documentation Conventions

remove_scenario

Tcl command; removes a scenario from the constraint database.

```
remove_scenario <name>
```

Arguments

name

Specifies the name of the scenario to delete.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

The following command removes the scenario named my_scenario:

```
remove_scenario my_scenario
```

See Also

create_scenario

remove_set

Tcl command; removes a set of paths from analysis. Only user-created sets can be deleted.

```
remove_set -name name
```

Parameters

-name *name*

Specifies the name of the set to delete.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

The following command removes the set named my_set:

```
remove_set -name my_set
```

See Also

create_set

rename_scenario

Tcl command; renames an existing timing scenario to a new name. You must provide a unique name (that is, it cannot already be used by another timing scenario) for the new name.

```
rename_scenario old_name  new_name
```

Arguments

old_name

Specifies the name of the existing timing scenario to be renamed.

new_name

Specifies the new name for the scenario.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

This command renames an existing scenario name to a new name..

Example

```
rename_scenario my_old_scenario my_new_scenario
```

See Also

create_scenario

delete_scenario

Tcl documentation conventions

report

Tcl command; specifies the type of reports to generate and what to include in the reports.

```
report -type (timing | violations | datasheet | bottleneck | constraints_coverage |
combinational_loops) \
       -analysis <max|min> \
       -format (csv|text) \
        <filename>
        timing options
           -max_parallel_paths <number>
           -max_paths <number>
          -print_summary (yes|no)
          -use_slack_threshold (yes|no)
          -slack_threshold <double>
           -print_paths (yes|no)
          -max_expanded_paths <number>
          -include_user_sets (yes|no)
           -include_clock_domains (yes|no)
           -select_clock_domains <clock name list>
           -limit_max_paths (yes|no)
           -include_pin_to_pin (yes|no)
        bottleneck options
           -cost_type (path_count|path_cost)
           -max_instances <number>
           -from <port/pin pattern>
           -to <port/pin pattern>
           -set_type <set_type>
           -set_name <set name>
          -clock <clock name>
           -from_clock <clock name>
```

```
          -to_clock <clock name>
           -in_to_out
```

Arguments

`-type`

Specifies the type of the report to be generated. It is mandatory.

Value	Description
timing	Timing Report
violations	Timing Violation Report
datasheet	Datasheet Report
bottleneck	Bottleneck Report
constraints_coverage	Constraints Coverage Report
combinational_loops	Combinational Loops Report

`-analysis`

Specifies the type of Analysis(Max Delay or Min Delay) Performed to generate the reports. It is optional.

Note: This argument should not be used to generate datasheet reports. The command may fail if this argument is used to generate datasheet report.

Value	Description
max	Timing report considers maximum analysis (default).
min	Timing report considers minimum analysis.

`-format`

Specifies the format in which the report is generated. It is optional.

Value	Description
text	Generates a text report (default).
csv	Generates the report in a comma-separated value format which you can import into a spreadsheet.

`-filename`

Specifies the file name of the generated report. It is mandatory.

Timing Options and Values

Parameter/Value	Description
-max_parallel_paths <number>	Specifies the max number of parallel paths. Parallel paths are timing paths with the same start and end points.

Parameter/Value	Description
-max_paths <number>	Specifies the max number of paths to display for each set. This value is a positive integer value greater than zero. Default is 100.
-print_summary <yes|no>	Yes to include and No to exclude the summary section in the timing report.
-use_slack_threshold <yes|no>	Yes to include slack threshold and no to exclude threshold in the timing report. The default is to exclude slack threshold.
-slack_threshold <double>	Specifies the threshold value to consider when reporting path slacks. This value is in nanoseconds (ns). By default, there is no threshold (all slacks reported).
-print_paths (yes|no)	Specifies whether the path section (clock domains and in-to-out paths) will be printed in the timing report. Yes to include path sections (default) and no to exclude path sections from the timing report.
-max_expanded_paths <number>	Specifies the max number of paths to expand per set. This value is a positive integer value greater than zero. Default is 100.
-include_user_sets (yes|no)	If yes, the user set is included in the timing report. If no, the user set is excluded in the timing report.
-include_clock_domains (yes|no)	Yes to include and no to exclude clock domains in the timing report.
-select_clock_domains <clock_name_list>	Defines the clock domain to be considered in the clock domain section. The domain list is a series of strings with domain names separated by spaces. Both the summary and the path sections in the timing report display only the listed clock domains in the clock_name_list.
-limit_max_paths (yes|no)	Yes to limit the number of paths to report. No to specify that there is no limit to the number of paths to report (the default).
-include_pin_to_pin (yes|no)	Yes to include and no to exclude pin-to-pin paths in the timing report.

Bottleneck Options and Values

Parameter/Value	Description
-cost_type <path_count|path_cost>	Specifies the cost_type as either path_count or path_cost. For path_count, instances with the greatest number of path violations will have the highest bottleneck cost. For path_cost, instances with the largest combined timing violations will have the highest bottleneck cost.
-max_instances <number>	Specifies the maximum number of instances to be reported. Default is 10.
-from <port/pin pattern>	Reports only instances that lie on violating paths that start at locations specified by this option.

Parameter/Value	Description
-to <port/pin pattern>	Reports only instances that lie on violating paths that end at locations specified by this option.
-clock <clock name>	This option allows pruning based on a given clock domain. Only instances that lie on these violating paths are reported.
-set_name <set name>	Displays the bottleneck information for the named set. You can either use this option or use both -clock and -type. This option allows pruning based on a given set. Only paths that lie within the named set will be considered towards bottleneck.
-set_type <set_type>	This option can only be used in combination with the -clock option, and not by itself. The options allows you to filter which type of paths should be considered towards the bottleneck: • reg_to_reg - Paths between registers in the design • async_to_reg - Paths from asynchronous pins to registers • reg_to_async - Paths from registers to asynchronous pins • external_recovery - The set of paths from inputs to asynchronous pins • external_removal - The set of paths from inputs to asynchronous pins • external_setup - Paths from input ports to registers • external_hold - Paths from input ports to registers • clock_to_out - Paths from registers to output ports
-from_clock <clock name>	Reports only bottleneck instances that lie on violating timing paths of the inter-clock domain that starts at the source clock specified by this option. This option can only be used in combination with -to_clock.
-to_clock <clock name>	Reports only instances that lie on violating paths that end at locations specified by this option.
-in_to_out	Reports only instances that lie on violating paths that begin at input ports and end at output ports.

Example

The following example generates a timing violation report named timing_viol.txt. The report considers an analysis using maximum delays and does not filter paths based on slack threshold. It reports two paths per section and one expanded path per section.

```
report \
  -type violations \
  -analysis max \
  -use_slack_threshold no \
  -limit_max_paths yes \
  -max_paths 2 \
  -max_expanded_paths 1 \
  timing_viol.txt
```

The following example generates a datasheet report named datasheet.csv in CSV format.

```
report \
  -type datasheet \
  -format csv \
  datasheet.csv
```

save

Tcl command; saves all changes made prior to this command. This includes changes made on constraints, options and sets.

```
save
```

Arguments

None

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

The following script sets the maximum number of paths reported by list_paths to 10, reads an SDC file, and save both the option and the constraints into the design project:

```
set_options -limit_max_paths 10
read_sdc somefile.sdc
save
```

See Also

set_options

set_clock_groups

set_clock_groups is an SDC command which disables timing analysis between the specified clock groups. No paths are reported between the clock groups in both directions. Paths between clocks in the same group continue to be reported.

```
set_clock_groups [-name name]
                 [-physically_exclusive | -logically_exclusive | -asynchronous]
                 [-comment comment_string]
                 -group clock_list
```

Note: If you use the same name and the same exclusive flag of a previously defined clock group to create a new clock group, the previous clock group is removed and a new one is created in its place.

Arguments

-name *name*

Name given to the clock group. Optional.

-physically_exclusive

Specifies that the clock groups are physically exclusive with respect to each other. Examples are multiple clocks feeding a register clock pin. The exclusive flags are all mutually exclusive. Only one can be specified.

-logically_exclusive

Specifies that the clocks groups are logically exclusive with respect to each other. Examples are clocks passing through a mux.

`-asynchronous`

Specifies that the clock groups are asynchronous with respect to each other, as there is no phase relationship between them. The exclusive flags are all mutually exclusive. Only one can be specified.

Note: The exclusive flags for the arguments above are all mutually exclusive. Only one can be specified.

`-group clock_list`

Specifies a list of clocks. There can any number of groups specified in the set_clock_groups command.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set_clock_groups -name mygroup3 -physically_exclusive \
-group [get_clocks clk_1] -group [get_clocks clk_2]
```

See Also

list_clock_groups

remove_clock_groups

set_clock_latency

Tcl command; defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

Arguments

`-source`

Specifies the source latency on a clock pin, potentially only on certain edges of the clock.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-fall`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`-late`

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`-early`

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and

"-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

delay

Specifies the latency value for the constraint.

clock

Specifies the clock to which the constraint is applied. This clock must be constrained.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a"late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

Examples

The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency -source -rise -early 0.4 { main_clock  }
set_clock_latency -source -fall    1.2 { main_clock }
```

See Also

create_clock

create_generated_clock

set_clock_to_output

SDC command; defines the timing budget available inside the FPGA for an output relative to a clock.

```
set_clock_to_output delay_value -clock clock_ref [-max] [-min] output_list
```

Arguments

delay_value

Specifies the clock to output delay in nanoseconds. This time represents the amount of time available inside the FPGA between the active clock edge and the data change at the output port.

-clock *clock_ref*

Specifies the reference clock to which the specified clock to output is related. This is a mandatory argument.

-max

Specifies that *delay_value* refers to the maximum clock to output at the specified output. If you do not specify –max or –min options, the tool assumes maximum and minimum clock to output delays to be equal.

-min

Specifies that *delay_value* refers to the minimum clock to output at the specified output. If you do not specify –max or –min options, the tool assumes maximum and minimum clock to output delays to be equal.

output_list

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

SmartFusion2

IGLOO2

RTG4

set_clock_uncertainty

Tcl command; specifies simple clock uncertainty for single clock and clock-to-clock uncertainty between two clocks (from and to).

```
set_clock_uncertainty [-setup] [-hold] uncertainty [object_list -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]
```

Arguments

uncertainty

Specifies the time in nanoseconds that represents the amount of variation between two clock edges.

object_list

Specifies a list of clocks, ports, or pins for simple uncertainty; the uncertainty is applied either to destination flops clocked by one of the clocks in the object list option , or destination flops whose clock pins are in the fanout of a port or a pin specified in the object_list option.

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

from_clock/rise_from_clock/fall_from_clock

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

to_clock/rise_to_clock/fall_to_clock

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

set_clock_uncertainty command sets the timing uncertainty of clock networks. It can be used to model clock jitter or add guard band in timing analysis. Either simple clock uncertainty or clock-to-clock uncertainty can be specified.

Simple clock uncertainty can be set on a clock or on any pin in the clock network. It will then apply to any path with the capturing register in the forward cone of the uncertainty. If multiple simple uncertainty applies to a register, the last one (in the propagation order from the clock source to the register) is used.

Clock-to-clock uncertainty applies to inter-clock paths. Both “from” clock and “to” clock must be specified. Clock-to-clock uncertainty has higher priority than simple uncertainty. If both are set (a clock-to-clock uncertainty and a simple clock uncertainty on the “to” clock), the simple clock uncertainty will be ignored for inter-clock paths, only the clock-to-clock uncertainty will be used.

Examples

Simple Clock Uncertainty constraint examples:

```
set_clock_uncertainty 2 -setup [get_clocks clk]
set_clock_uncertainty 2 [get_clocks clk]
```

Clock to Clock Uncertainty constraint examples:

```
set_clock_uncertainty 10 -from Clk1 -to Clk2
set_clock_uncertainty 0 -from Clk1 -fall_to { Clk2 Clk3 } -setup
set_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
set_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 }
-setup
set_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
```

set_current_scenario

Tcl command; specifies the timing scenario for the Timing Analyzer to use. All commands that follow this command will apply to the specified timing scenario.

```
set_current_scenario name
```

Arguments

name

Specifies the name of the timing scenario to which to apply all commands from this point on.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

A timing scenario is a set of timing constraints used with a design. If the specified scenario is already the current one, this command has no effect.

After setting the current scenario, constraints can be listed, added, or removed, the checker can be invoked on the set of constraints, and so on.

This command uses the specified timing scenario to compute timing analysis.

Example

```
set_current_scenario scenario_A
```

See Also

get_current_scenario

Tcl Command Documentation Conventions

set_disable_timing

Tcl command; disables timing arcs within a cell and returns the ID of the created constraint if the command succeeded.

```
set_disable_timing -from value -to value name
```

Arguments

-from *from_port*

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

-to *to_port*

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

name

Specifies the cell name where the timing arcs will be disabled.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set_disable_timing -from A -to Y a2
```

See Also

Tcl documentation conventions

set_external_check

SDC command; defines the external setup and hold delays for an input relative to a clock.

```
set_external_check delay_value -clock clock_ref [-setup] [-hold] input_list
```

Arguments

delay_value

Specifies the external setup or external hold delay in nanoseconds. This time represents the amount of time available inside the FPGA for the specified input after a clock edge.

-clock *clock_ref*

Specifies the reference clock to which the specified external check is related. This is a mandatory argument.

-setup or -hold

Specifies that *delay_value* refers to the setup/hold check at the specified input. This is a mandatory argument if –hold is not used. You must specify either the -setup or -hold option.

input_list

Provides a list of input ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

SmartFusion2

IGLOO2

RTG4

set_false_path

Tcl command; identifies paths that are considered false and excluded from the timing analysis in the current timing scenario.

```
set_false_path [-ignore_errors] [-from from_list] [-through through_list] [-to to_list]
```

Arguments

-ignore_errors

Specifies to avoid reporting errors for derived constraints targeting the logic that becomes invalid due to logic optimization. It is an optional argument. Some IPs may have extra logic present depending on other IPs used in the design but the synthesis tool will remove this logic if fewer IPs were used. In such cases, the implementation flow will halt without -ignore_errors flag.

Note: It is not recommended to use this flag outside similar use cases.

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-through *through_list*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

The set_false_path command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these

false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one -through option, the path can pass through any objects.

You must specify at least one of the `-from`, `-to`, or `-through` arguments for this constraint to be valid.

Examples

The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

```
set_false_path -from [get_clocks {clk1}] -to reg_2:D
```

The following example specifies all paths through the pin U0/U1:Y to be false:

```
set_false_path -through U0/U1:Y
```

The following example specifies a derived false path constraint through the pin PCIe_Demo_0/SYSRESET_POR/POWER_ON_RESET_N

```
set_false_path -ignore_errors -through [ get_pins
{PCIe_Demo_0/SYSRESET_POR/POWER_ON_RESET_N } ]
```

See Also

Tcl Command Documentation Conventions

set_input_delay

Tcl command; creates an input delay on a port list by defining the arrival time of an input relative to a clock in the current scenario.

```
set_input_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] input_list
```

Arguments

delay_value

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

`-clock` *clock_ref*

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

SmartFusion2

IGLOO2
RTG4

Description

The set_input_delay command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance reg1:CLK. This name can be hierarchical (for instance toplevel/block1/reg2:CLK)
- an object accessor that will refer to one clock: [get_clocks {clk}]

Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:

```
set_input_delay 1.0 -clock_fall -clock CLK2 -min {IN1}
set_input_delay 1.4 -clock_fall -clock CLK2 -max {IN1}
```

See Also

set_output_delay

Tcl Command Documentation Conventions

set_max_delay

Tcl command; specifies the maximum delay for the timing paths in the current scenario.

```
set_max_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

-through *through_list*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual maximum delay targets from clock waveforms and port input or output delays.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `-from`, `-to`, or `-through` arguments for this constraint to be valid.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

See Also

set_min_delay

remove_max_delay

Tcl Command Documentation Conventions

set_min_delay

Tcl command; specifies the minimum delay for the timing paths in the current scenario.

```
set_min_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual minimum delay targets from clock waveforms and port input or output delays.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `-from`, `-to`, or `-through` arguments for this constraint to be valid.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

See Also

set_max_delay

remove_min_delay

Tcl Command Documentation Conventions

set_multicycle_path

Tcl command; defines a path that takes multiple clock cycles in the current scenario.

```
set_multicycle_path ncycles [-setup] [-hold] [-setup_only] [-from from_list] [-through
through_list] [-to to_list]
```

Arguments

ncycles

Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.

`-setup`

Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another set_multicycle_path command for the hold value.

`-hold`

Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

Note: If you do not specify "-setup" or "-hold", the cycle value is applied to the setup check and the default hold check is performed (*ncycles* -1).

`-setup_only`

Optional. Specifies that the path multiplier is applied to setup paths only. The default value for hold check (which is 0) is applied.

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins or ports through which the multiple cycle paths must pass.

`-to` *`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Description

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one -through option, the path passes through any of the objects.

You must specify at least one of the `-from`, `-to`, or `-through` arguments for this constraint to be valid.

Exceptions

Multiple priority management is not supported in Microsemi SoC designs. All multiple cycle path constraints are handled with the same priority.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] -to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain ck1. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

The following example specifies that four cycles are needed for setup only check on all paths starting at the registers in the clock domain REF_CLK_0.

```
set_multicycle_path -setup_only 4 -from [ get_clocks { REF_CLK_0 } ]
```

See Also

`remove_multicycle_path`

Tcl Command Documentation Conventions

set_options

SmartTime-specific Tcl command; sets options for timing analysis. Some options will also affect timing-driven place-and-route. The same parameters can be changed in the SmartTime Options dialog box in the SmartTime GUI.

```
set_options
 [-max_opcond value ]
 [-min_opcond value]
 [-interclockdomain_analysis value]
 [-use_bibuf_loopbacks value]
 [-enable_recovery_removal_checks value]
 [-break_at_async value]
 [-filter_when_slack_below value]
 [-filter_when_slack_above value]
 [-remove_slack_filters]
 [-limit_max_paths value]
 [-expand_clock_network value]
 [-expand_parallel_paths value]
 [-analysis_scenario value]
 [-tdpr_scenario value]
 [-reset]
```

Arguments

-max_opcond *value*

Sets the operating condition to use for Maximum Delay Analysis.

The acceptable Values for max_opcode for SmartFusion2, IGLOO2 and RTG4 is shown in the below table. Default is worst.

Value	Description
worst	Use Worst Case conditions for Maximum Delay Analysis
typical	Use Typical conditions for Maximum Delay Analysis
best	Use Best Case conditions for Maximum Delay Analysis

-min_opcond *value*

Sets the operating condition to use for Minimum Delay Analysis.

The acceptable Values for min_opcode for SmartFusion2, IGLOO2 and RTG4 is shown in the below table. Default is best.

Value	Description
best	Use Best Case conditions for Minimum Delay Analysis
typical	Use Typical conditions for Minimum Delay Analysis
worst	Use Worst Case conditions for Minimum Delay Analysis

-interclockdomain_analysis *value*

Enables or disables inter-clock domain analysis. Default is *yes*.

Value	Description
yes	Enables inter-clock domain analysis
no	Disables inter-clock domain analysis

-use_bibuf_loopbacks *value*

Instructs the timing analysis whether to consider loopback path in bidirectional buffers (D->Y, E->Y)as false-path {no}. Default is *yes*; i.e., loopback are false paths.

Value	Description
yes	Enables loopback in bibufs
no	Disables loopback in bibufs

-enable_recovery_removal_checks *value*

Enables recovery checks to be included in max-delay analysis and removal checks in min-delay analysis. Default is *yes*.

Value	Description
yes	Enables recovery and removal checks
no	Disables recovery and removal checks

-break_at_async *value*

Specifies whether or not timing analysis is allowed to cross asynchronous pins (clear, reset of sequential elements). Default is *no*.

Value	Description
yes	Enables breaking paths at asynchronous ports
no	Disables breaking paths at asynchronous ports

-filter_when_slack_below *value*

Specifies a minimum slack value for paths reported by list_paths. Not set by default.

-filter_when_slack_above *value*

Specifies a maximum slack value for paths reported by list_paths. Not set by default.

-remove_slack_filters

Removes the slack minimum and maximum set using -filter_when_slack_below and filter_when_slack_above.

-limit_max_paths *value*

Specifies the maximum number of paths reported by list_paths. Default is *100*.

-expand_clock_network *value*

Specify whether or not clock network details are reported in expand_path. Default is *yes*.

Value	Description
yes	Enables expanded clock network information in paths

Value	Description
no	Disables expanded clock network information in paths

-expand_parallel_paths *value*

Specify the number of parallel paths {paths with the same ends} to include in expand_path. Default is *1*.

-analysis_scenario *value*

Specify the constraint scenario to be used for timing analysis. Default is *Primary*, the default scenario.

-tdpr_scenario *value*

Specify the constraint scenario to be used for timing-driven place-and-route. Default is Primary, the default scenario.

-reset

Reset all options to the default values, except those for analysis and TDPR scenarios, which remain unchanged.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

The following script commands the timing engine to use best operating conditions for both max-delay analysis and min-delay analysis:

```
set_options -max_opcond {best} -min_opcond {best}
```

The following script changes the scenario used by timing-driven place-and-route and saves the change in the Libero project for place-and-route tools to see the change.

```
set_options -tdpr_scenario {My_TDPR_Scenario}
```

See Also

save

set_output_delay

Tcl command; defines the output delay of an output relative to a clock in the current scenario.

```
set_output_delay [-max] [-min] delay_value -clock clock_ref [-clock_fall] output_list
```

Arguments

-max

Specifies that delay_value refers to the longest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

-min

Specifies that delay_value refers to the shortest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

delay_value

Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

-clock *clock_ref*

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

`output_list`

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

SmartFusion2

IGLOO2

RTG4

Description

The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

Examples

The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay -min {OUT1} 1.0 -clock_fall -clock CLK2
set_output_delay -max {OUT1} 1.4 -clock_fall -clock CLK2
```

See Also

remove_output_delay

set_input_delay

Tcl Command Documentation Conventions

write_sdc

Tcl command; writes timing constraints into an SDC file. If multiple constraint scenarios are defined, -scenario allows the user to specify which scenario to write. By default, the current scenario is written.

```
write_sdc
-scenario scenario name
-pin_separator (: | /)
file name
```

Arguments

`-scenario` *`scenario name`*

Specify the scenario to write. By default the current scenario is used.

`-pin_separator` *`sep`*

Specify the pin separator used in the SDC file. It can be either ':' or '/'.

`file name`

Specify the SDC file name.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

The following script merges two SDC files and writes the result into a third SDC file:

```
read_sdc first.sdc
read_sdc -add second.sdc
write_sdc merged.sdc
```

See Also

read_sdc, set_current_scenario
VERIFYTIMING (SmartFusion2 , IGLOO2, RTG4,)

SmartPower Tcl Commands

smartpower_add_new_scenario

Tcl command; creates a new scenario.

```
smartpower_add_new_scenario -name {value} -description {value} -mode {value}
```

Arguments

`-name {value}`

Specifies the name of the new scenario.

`-description {value}`

Specifies the description of the new scenario.

`-mode {<operating mode>:<duration>}+`

Specifies the mode(s) and duration(s) for the specified scenario.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example creates a new scenario called myscenario:

```
smartpower_add_new_scenario -name "MyScenario" -mode "Custom_1:50.00"
"Custom_2:25.00" -mode "Active:25.00"
```

See Also

Tcl documentation conventions

smartpower_add_pin_in_domain

Tcl command; adds a pin into a clock or set domain.

```
smartpower_add_pin_in_domain -pin_name {pin_name} -pin_type {value} -domain_name
{domain_name} -domain_type {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to add to the domain.

`-pin_type {value}`

Specifies the type of the pin to add. The following table shows the acceptable values for this argument:

Value	Description
clock	The pin to add is a clock pin
data	The pin to add is a data pin

`-domain_name {domain_name}`

Specifies the name of the domain in which to add the specified pin.

`-domain_type {value}`

Specifies the type of domain in which to add the specified pin. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

Supported Families

SmartFusion2
IGLOO2
RTG4

Notes

- The `domain_name` must be a name of an existing domain.
- The `pin_name` must be a name of a pin that exists in the design.

Examples

The following example adds a clock pin to an existing Clock domain:

```
smartpower_add_pin_in_domain -pin_name { XCMP3/U0/U1:Y } -pin_type {clock} -domain_name {clk1} -domain_type {clock}
```

The following example adds a data pin to an existing Set domain:

```
smartpower_add_pin_in_domain -pin_name {XCMP3/U0/U1:Y} -pin_type {data} -domain_name {myset} -domain_type {set}
```

See Also

Tcl documentation conventions

smartpower_remove_pin_of_domain

smartpower_battery_settings

This SmartPower Tcl command sets the battery capacity in SmartPower. The battery capacity is used to compute the battery life of your design.

```
smartpower_battery_settings -capacity {decimal value}
```

Parameters

`-capacity {decimal value}`

Value must be a positive decimal.

This parameter is mandatory.

Exceptions

None

Returns

This command does not return a value.

Supported Families

SmartFusion2
IGLOO2
RTG4

Usage

The following table lists the parameters for the command, their types, and the values they can be set to.

smartpower_battery_settings	Type	Value	Description
capacity	Decimal	Positive decimal	Specify the battery capacity in mA*Hours

Example

This example sets the battery capacity to 1800 mA * Hours.

```
smartpower_battery_settings –capacity {1800}
```

smartpower_change_clock_statistics

Tcl command; changes the default frequencies and probabilities for a specific domain.

```
smartpower_change_clock_statistics -domain_name {value} -clocks_freq {value} -
clocks_proba {value} -registers_freq {value} -registers_proba {value} -set_reset_freq
{value} -set_reset_proba {value} -primaryinputs_freq {value} -primaryinputs_proba {value} -
combinational_freq {value} -combinational_proba {value}
```

Arguments

`-domain_name{value}`

Specifies the domain name in which to initialize frequencies and probabilities.

`-clocks_freq {value}`

Specifies the user input frequency in Hz, KHz, or MHz for all clocks.

`-clocks_proba {value}`

Specifies the user input probability in % for all clocks.

`-registers_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-registers_proba {value}`

Specifies the user input probability in % for all registers.

`-set_reset_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-set_reset_proba {value}`

Specifies the user input probability in % for all set/reset nets.

`-primaryinputs_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-primaryinputs_proba {value}`

Specifies the user input probability in % for all primary inputs.

`-combinational_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-combinational_proba {value}`

Specifies the user input probability in % for all combinational combinational output.

Supported Families

SmartFusion2

IGLOO2

RTG4

Note: This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks with:

```
smartpower_change_clock_statistics -domain_name {my_domain} -clocks_freq {10 MHz} -
clocks_proba {20} -registers_freq {10 MHz} -registers_proba {20} -set_reset_freq {10
MHz} -set_reset_proba {20} -primaryinputs_freq {10 MHz} -primaryinputs_proba {20} -
combinational_freq {10 MHz} -combinational_proba {20}
```

See Also

Tcl documentation conventions

smartpower_change_setofpin_statistics

Tcl command; changes the default frequencies and probabilities for a specific set.

```
smartpower_change_setofpin_statistics -domain_name {value} -data_freq {value} -
data_proba {value}
```

Arguments

`-domain_name{value}`

Specifies the domain name in which to initialize data frequencies and probabilities.

`-data_freq {value}`

Specifies the user input data frequency in Hz, KHz, or MHz for all sets of pins.

`-data_proba {value}`

Specifies the user input data probability in % for all sets of pins.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks withs:

```
smartpower_change_setofpin_statistics -domain_name {my_domain} -data_freq {10 MHz} -
data_proba {20}
```

See Also

Tcl documentation conventions

smartpower_commit

Tcl command; saves the changes to the design **(.adb)** file.

```
smartpower_commit
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
smartpower_commit
```

See Also

Tcl documentation conventions

smartpower_compute_vectorless

This Tcl command executes a vectorless analysis of the current operating mode.

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
smartpower_compute_vectorless
```

See Also

Tcl Command Documentation Conventions

smartpower_create_domain

Tcl command; creates a new clock or set domain.

```
smartpower_create_domain -domain_type {value} -domain_name {domain_name}
```

Arguments

`-domain_type {value}`

Specifies the type of domain to create. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

`-domain_name {domain_name}`

Specifies the name of the new domain.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

The `domain name` cannot be the name of an existing domain.

The domain `type` must be either clock or set.

Examples

The following example creates a new clock domain named "clk2":

```
smartpower_create_domain -domain_type {clock} -domain_name {clk2}
```

The following example creates a new set domain named "myset":

```
smartpower_create_domain -domain_type {set} -domain_name {myset}
```

See Also

Tcl documentation conventions

smartpower_remove_domain

smartpower_edit_scenario

Tcl command; edits a scenario.

```
smartpower_edit_scenario -name {value} -description {value} -mode {value} -new_name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

`-description {value}`

Specifies the description of the scenario.

`-mode {<operating mode>:<duration>}`

Specifies the mode(s) and duration(s) for the specified scenario.

`-new_name {value}`

Specifies the new name for the scenario

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

This example edits the name of myscenario to finalscenario:

```
smartpower_edit_scenario -name myscenario -new_name finalscenario
```

See Also

Tcl documentation conventions

smartpower_import_vcd

This SmartPower Tcl command imports into SmartPower a VCD file generated by a simulation tool. SmartPower extracts the frequency and probability information from the VCD.

```
import_vcd -file "VCD file" [-opmode "mode name"] [-with_vectorless "TRUE | FALSE"] [-
partial_parse\ "TRUE | FALSE"] [-start_time "decimal value"] [-end_time "decimal value"]
\
[-auto_detect_top_level_name "TRUE | FALSE"] [-top_level_name "top level name"] [-
glitch_filtering\ "false | auto | true"] [-glitch_threshold "integer value"] [-stop_time
"decimal value"]
```

Parameters

`-file "VCD file"`

Value must be a file path. This parameter is mandatory.

`[-opmode "mode name"]`

Value must be a string. This parameter is optional.

`[-with_vectorless "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-partial_parse "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-start_time "decimal value"]`

Value must be a positive decimal. This parameter is optional.

`[-end_time "decimal value"]`

Value must be a positive decimal. This parameter is optional.

`[-auto_detect_top_level_name "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-top_level_name "top level name"]`

Value must be a string. This parameter is optional.

`[-glitch_filtering "false | auto | true"]`

Value must be one of false | auto | true. This parameter is optional.

`[-glitch_threshold "integer value"]`

Value must be a positive integer. This parameter is optional.

Exceptions

None

Returns

This command does not return a value.

Usage

This section lists all the parameters for the command, their types, and the values they can be set to. The default value is always listed first.

smartpower_import_vcd	Type	Values	Description
file	String	Path to a VCD file	Path to a VCD file.
opmode	String	Operating mode name “Active” by default	Operating mode in which the VCD will be imported. If the mode doesn’t exist, it will be created.
with_vectorless	Boolean	TRUE|FALSE	Specify the method to set the frequency and probability information for signals not annotated by the VCD TRUE: use the vectorless analysis FALSE: use average value computed from the VCD.
partial_parse	Boolean	FALSE|TRUE	Enable partial parsing of the VCD. Start time and end time need to be specified when TRUE.
start_time	Decimal value	positive decimal nanoseconds (ns)	Specify the starting timestamp of the VCD extraction in ns. It must be lower than the specified end_time. It must be lower than the last timestamp in the VCD file.
end_time	Decimal value	positive decimal nanoseconds (ns)	Specify the end timestamp of the VCD extraction in ns. It must be higher than the specified start_time.
auto_detect_top_level_name	Boolean	TRUE|FALSE	Enable the auto detection of the top level name in the VCD file. Top_level_name needs to be specified when FALSE.
top_level_name	Boolean	Full hierarchical name	Specify the full hierarchical name of the instance of the design in the VCD file.
glitch_filtering	Boolean	Auto|FALSE|TRUE	AUTO: Enable glitch filtering with predefined thereshold based on the family TRUE: Enable glitch filtering, glitch_threshold must be specified FALSE: Disable glitch filtering.
glitch_threshold	Integer	Positive integer	Specify the threshold in ps below which glitches are filtered out.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

The Tcl command below imports the power.vcd file generated by the simulator into SmartPower:

```
smartpower_import_vcd -file "../../simulation/power.vcd"
```

The Tcl command below extracts information between 1ms and 2ms in the simulation, and stores the information into a custom mode:

```
smartpower_import_vcd -file "../../simulation/power.vcd" -partial_parse TRUE -start_time 1000000 -end_time 2000000 -opmode "power_1ms_to_2ms"
```

smartpower_init_do

Tcl command; initializes the frequencies and probabilities for clocks, registers, set/reset nets, primary inputs, combinational outputs, enables and other sets of pins, and selects a mode for initialization.

```
smartpower_init_do -with {value} -opmode {value} -clocks {value} -registers {value} -set_reset {value} -primaryinputs {value} -combinational {value} -enables {value} -othersets {value}
```

Arguments

`-with{value}`

This sets the option of initializing frequencies and probabilities with vectorless analysis or with fixed values. The following table shows the acceptable values for this argument:

Value	Description
vectorless	Initializes frequencies and probabilities with vectorless analysis
fixed	Initializes frequencies and probabilities with fixed values

`-opmode {value}`

Optional; specifies the mode in which to initialize frequencies and probabilities. The value must be Active or Flash*Freeze.

`-clocks {value}`

This sets the option of initializing frequencies and probabilities for all clocks. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all clocks
false	Does not initialize frequencies and probabilities for all clocks

`-registers {value}`

This sets the option of initializing frequencies and probabilities for all registers. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all registers
false	Does not initialize frequencies and probabilities for all registers

`-set_reset {value}`

This sets the option of initializing frequencies and probabilities for all set/reset nets. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all set/reset nets
false	Does not initialize frequencies and probabilities for all set/reset nets

`-primaryinputs{value}`

This sets the option of initializing frequencies and probabilities for all primary inputs. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all primary inputs
false	Does not initialize frequencies and probabilities for all primary inputs

`-combinational {value}`

This sets the option of initializing frequencies and probabilities for all combinational outputs. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all combinational outputs
false	Does not initialize frequencies and probabilities for all combinational outputs

`-enables {value}`

This sets the option of initializing frequencies and probabilities for all enable sets of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all enable sets of pins
false	Does not initialize frequencies and probabilities for all enable sets of pins

`-othersets {value}`

This sets the option of initializing frequencies and probabilities for all other sets of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all other sets of pins
false	Does not initialize frequencies and probabilities for all other sets of pins

Supported Families

SmartFusion2

IGLOO2

RTG4

Note: This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks with:

```
smartpower_init_do -with {vectorless} -opmode {my_mode} -clocks {true} -registers {true} -asynchronous {true} -primaryinputs {true} -combinational {true} -enables {true} -othersets {true}
```

See Also

Tcl documentation conventions

smartpower_init_set_clocks_options

Tcl command; initializes the clock frequency options of all clock domains.

```
smartpower_init_set_clocks_options -with_clock_constraints {value} -with_default_values {value} -freq {value} -duty_cycle {value}
```

Arguments

`-with_clock_constraints {value}`

This sets the option of initializing the clock frequencies with frequency constraints from SmartTime. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with clock constraints ON
false	Sets initialize clock frequencies with clock constraints OFF

`-with_default_values {value}`

This sets the option of initializing the clock frequencies with a user input default value. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with default values ON
false	Sets initialize clock frequencies with default values OFF

`-freq {value}`

Specifies the user input frequency in Hz, KHz, or MHz.

`-duty_cycle {value}`

Specifies the user input duty cycles in %.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks after executing `smartpower init do` with `-clocks {true}`:

```
smartpower_init_set_clocks_options -with_clock_constraints {true} -with_default_values {true} -freq {10 MHz} -duty_cycle {20}
```

See Also

Tcl documentation conventions

smartpower_init_set_combinational_options

Tcl commands; initializes the frequency and probability of all combinational outputs.

```
smartpower_init_set_combinational_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all combinational signals after executing `smartpower init do` with `-combinational {true}`:

```
smartpower_init_set_combinational_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

smartpower_init_set_enables_options

Tcl command; initializes the clock frequency of all enable clocks with the initialization options.

```
smartpower_init_set_enables_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz).

`-proba {value}`

Specifies the user input probability in %.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks after executing `smartpower init do` with `-enables {true}`:

```
smartpower_init_set_enables_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

smartpower_init_set_primaryinputs_options

Tcl command; initializes the frequency and probability of all primary inputs.

```
smartpower_init_set_primaryinputs_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all primary inputs after executing smartpower_init_do with -primaryinputs {true}:

```
smartpower_init_set_primaryinputs_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

smartpower_init_set_registers_options

Tcl command; initializes the frequency and probability of all register outputs.

```
smartpower_init_set_registers_options -freq {value} -proba {value}
```

Arguments

-freq {*value*}

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

-proba {*value*}

Specifies the user input probability in %.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Exceptions

None

Examples

The following example initializes all register outputs after executing smartpower_init_do with -registers {true}:

```
smartpower_init_set_registers_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

smartpower_init_setofpins_values

Tcl command; initializes the frequency and probability of all sets of pins.

```
smartpower_init_setofpins_values -domain_name {name} -freq {value} -proba {value}
```

Arguments

-domain_name{*name*}

Specifies the set of pins that will be initialized. The following table shows the acceptable values for this argument:

Value	Description
IOsEnableSet	Specifies that the IOsEnableSet set of pins will be initialized
MemoriesEnableSet	Specifies that the MemoriesEnableSet set of pins will be initialized

`-freq {value}`

Specifies the user input frequency in Hz, MHz, or KHz.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all primary inputs after executing `smartpower init do` with `-othersets {true}`:

```
smartpower_init_setofpins_values -domain_name {IOsEnableSet} -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

smartpower_remove_all_annotations

Tcl command; removes all initialization annotations for the specified mode.

```
smartpower_remove_all_annotations -opmode {value}
```

Arguments

`-opmode {value}`

Removes all initialization annotations for the specified mode, where value must be Active or Flash*Freeze.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks with opmode Acitve:

```
smartpower_remove_all_annotations -opmode {Active}
```

See Also

Tcl documentation conventions

smartpower_remove_file

Tcl command; removes a VCD file from the specified mode or all operating mode. Frequency and probability information of signals annotated by the VCD are set back to the default value.

```
remove_file
-file {value} \
-format {value} \
-opmode {value} \
```

Arguments

`-file {value}`

Specifies the file to be removed. This is mandatory.

`-format VCD`

Specifies that the type to be removed is a VCD file. This is mandatory.

`[-opmode {value}]`

Specifies the operating mode. This is optional. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to active
Standby	The operating mode is set to Static
Flash*Freeze	The operating mode is set to Flash*Freeze

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example removes the file test.vcd from the Active mode.

```
smartpower_remove_file -file "test.vcd" -format VCD -opmode "Active"
```

This example removes the VCD file power1.vcd from all operating modes:

```
smartpower_remove_file -file “power1.vcd” -format VCD
```

See Also

Tcl documentation conventions

smartpower_remove_pin_probability

Tcl command; removes the probability value associated with a specific pin. This pin will have a default probability based on the domain set it belongs to.

```
smartpower_remove_pin_probability -pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin with the probability to remove. This pin must be the direct driver of an enable pin.

Supported Families

SmartFusion2, IGLOO2, RTG4

Examples

The following example removes the probability of the pin driving the enable pin of a bidirectional I/O:

```
Smartpower_remove_pin_probability -pin_name mybibuf/U0/U1:EOUT
```

See Also

Tcl documentation conventions

`smartpower_set_pin_probability`

See the online help for more information.

smartpower_remove_scenario

Tcl command; removes a scenario from the current design.

```
smartpower_remove_scenario -name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example removes a scenario from the current design:

```
smartpower_remove_scenario -name myscenario
```

See Also

Tcl documentation conventions

smartpower_report_power

Tcl command; creates a Power report, which enables you to determine if you have any power consumption problems in your design. It includes information about the global device and SmartPower preferences selection,

and hierarchical detail (including gates, blocks, and nets), with a block-by-block, gate-by-gate, and net-by-net power summary SmartPower results.

```
smartpower_report_power\
[-powerunit {value}] \
[-frequnit {value}] \
[-opcond {value}] \
[-opmode {value}] \
[-toggle {value}] \
[-power_summary {value}] \
[-rail_breakdown{value}] \
[-type_breakdown{ value}] \
[-clock_breakdown{value}] \
[-thermal_summary {value}] \
[-battery_life {value}] \
[-opcond_summary {value}] \
[-clock_summary {value}] \
[-style {value}] \
[-sortorder {value}] \
[-sortby {value}] \
[-instance_breakdown {value}] \
[-power_threshold {value}] \
[-filter_instance {value}] \
[-min_power {number}] \
[-max_instance {integer >= 0}] \
[-activity_sortorder {value}] \
[-activity_sortby {value}] \
[-activity_summary {value}] \
[-frequency_threshold {value}] \
[-filter_pin {value}] \
[-min_frequency {value}] \
[-max_pin {value}] \
[-enablerates_sortorder {value}] \
[-enablerates_sortby {value}] \
[-enablerates_summary {value}] \
[-with_annotation_coverage {value}] \
{filename}
```

Arguments

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-opcond {value}`

Specifies the operating condition. The following table shows the acceptable values for this argument:

Value	Description
worst	The operating condition is set to worst case
typical	The operating condition is set to typical case
best	The operating condition is set to best case

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

`-power_summary {value}`

Specifies whether to include the power summary, which shows the static and dynamic values in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the power summary in the report
false	Does not include the power summary in the report

`-rail_breakdown {value}`

Specifies whether to include the breakdown by rail summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by rail summary in the report
false	Does not include the breakdown by rail summary in the report

`-type_breakdown {value}`

Specifies whether to include the breakdown by type summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by type summary in the report
false	Does not include the breakdown by type summary in the report

`-clock_breakdown {value}`

Specifies whether to include the breakdown by clock domain in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by clock domain summary in the report
false	Does not include the breakdown by clock domain summary in the report

`-thermal_summary {value}`

Specifies whether to include the thermal summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the thermal summary in the report
false	Does not include the thermal summary in the report

`-battery_life {value}`

Specifies whether to include the battery life summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the battery life summary in the report
false	Does not include the battery life summary in the report

`-opcond_summary {value}`

Specifies whether to include the operating conditions summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the operating conditions summary in the report
false	Does not include the operating conditions summary in the report

`-clock_summary {value}`

Specifies whether to include the clock domains summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the clock summary in the report
false	Does not include the clock summary in the report

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-sortby {value}`

Specifies how to sort the values in the report. The following table shows the acceptable values for this argument:

Value	Description
power values	Sorts based on the power values
alphabetical	Sorts in an alphabetical order

`-sortorder {value}`

Specifies the sort order of the values in the report. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

`-instance_breakdown {value}`

Specifies whether to include the breakdown by instance in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by instance in the report
false	Does not include the breakdown by instance in the report

`-power_threshold {value}`

This specifies whether to include only the instances that consume power above a certain minimum value. When this command is set to true, the `-min_power` argument must also be used to specify that only the instances that consume power above this minimum power value are the ones that are included in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the power threshold in the report
false	Does not include the power threshold in the report

`-filter_instance {value}`

This specifies whether to have a limit on the number of instances to include in the Power report. When this command is set to true, the `-max_instance` argument must also be used to specify the maximum number of instances to be included into the Power report. The following table shows the acceptable values for this argument:

Value	Description
true	Indicates that you want to have a limit on the number of instances to include in the Power report
false	Indicates that you do not want to have a limit on the number of instances to include in the Power report

`-min_power {number}`

Specifies which block to expand based on the minimum power value of a block.

`-max_instance {integer >= 0}`

Sets the maximum number of instances to a specified integer greater than or equal to 0 (zero). This will limit the maximum number of instances to be included in the Power report.

`-activity_sortorder {value}`

Specifies the sort order for the activity summary. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

`-activity_sortby {value}`

Specifies how to sort the values for the activity summary. The following table shows the acceptable values for this argument:

Value	Description
pin name	Sorts based on the pin name
net name	Sorts based on thepnet name
domain	Sorts based on the clock domain
frequency	Sorts based on the clock frequency
source	Sorts based on the clock frequency source

`-activity_summary {value}`

Specifies whether to include the activity summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the activity summary in the report
false	Does not include the activity summary in the report

`-frequency_threshold {value}`

Specifies whether to add a frequency threshold. The following table shows the acceptable values for this argument:

Value	Description
true	Adds a frequency threshold
false	Does not add a frequency threshold

`-filter_pin {value}`

Specifies whether to filter by maximum number of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Filters by maximum number of pins
false	Des not filter by maximum number of pins

`-min_frequency {value}`

Sets the minimum frequency to {decimal value [unit { Hz | KHz | MHz}]}.

`-max_pin {value}`

Sets the maximum number of pins.

`-enablerates_sortorder {value}`

Specifies the sort order for the probabilities summary. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order

Value	Description
descending	Sorts the values in descending order

`-enablerates_sortby {value}`

Specifies how to sort the values for the probabilities summary. The following table shows the acceptable values for this argument:

Value	Description
pin name	Sorts based on the pin name
net name	Sorts based on the net name
domain	Sorts based on the clock domain
frequency	Sorts based on the clock frequency
source	Sorts based on the clock frequency source

`-enablerates_summary {value}`

Specifies whether to include the probabilities summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the activity summary in the report
false	Does not include the activity summary in the report

`-with_annotation_coverage {value}`

Specifies whether to include the annotation coverage summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the annotation coverage summary in the report
false	Does not include the annotation coverage summary in the report

`{filename}`

Specifies the name of the report.

Supported Families

SmartFusion2

IGLOO2

RTG4

Notes

- The following arguments have been removed. Running the script will trigger a warning message: **Warning: Invalid argument: -argname "argvalue" Ignored**. Ignore the warning.

```
-annotated_pins {value}
-stat_pow {value}
-dyn_pow {value}
```

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best are available only for certain families and devices.

Examples

This example generates a Power report named **report.rpt**.

```
smartpower_report_power -powerunit "uW" -frequnit "MHz" -opcond "Typical" -opmode
"Active" -toggle "TRUE" -rail_breakdown "TRUE" -battery_life "TRUE" -style "Text" -
power_summary "TRUE" -activity_sortby "Source" text_report.txt
```

smartpower_set_mode_for_pdpr

This SmartPower Tcl command sets the operating mode used by the Power Driven Place and Route (PDPR) tool during power optimization.

```
smartpower_set_mode_for_pdpr -opmode {value}
```

Parameters

```
-opmode {value}
```

Value must be a valid operating mode.

This parameter is mandatory.

Sets the operating mode for your power driven place and route.

Exceptions

None

Supported Families

SmartFusion2

IGLOO2

RTG4

Return Value

This command does not return a value.

Examples

This example sets the Active mode as the operating mode for Power Driven Place and Route.

```
set_mode_for_pdpr -opmode "Active"
```

This example creates a custom mode and set it to be used by Power Driven Place and Route (PDPR).

```
smartpower_add_new_custom_mode -name “MyCustomMode” \
-description “for PDPR” -base_mode “Active”
smartpower_set_mode_for_pdpr -opmode “MyCustomMode
```

See Also

Tcl Command Documentation Conventions

smartpower_set_operating_condition

Tcl command; sets the operating conditions used in SmartPower to one of the pre-defined types.

```
smartpower_set_operating_condition -opcond {value}
```

Arguments

`-opcond {value}`

Specifies the value of the operating condition. The following table shows the acceptable values for this argument:

Value	Description
best	Sets the operating conditions to best
typical	Sets the operating conditions to typical
worst	Sets the operating conditions to worst

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example sets the operating conditions to best:

```
smartpower_set_operating_condition -opcond {best}
```

See Also

Tcl documentation conventions

smartpower_set_operating_conditions

Tcl command; sets the operating conditions used in SmartPower.

```
smartpower_set_operating_conditions "still_air | 1.0_mps | 2.5_mps | custom" -heatsink
"None | custom | 10mm_Low_Profile | 15mm_Medium_Profile | 20mm_High_Profile" -boardmodel
"None_Conservative | JEDEC_2s2p" [-teta_ja "decimal value"] [-teta_sa "decimal value"]
```

Arguments

`-still_air {value}`

Specifies the value for the still air operating condition. The following table shows the acceptable values for this argument:

Value	Description
1.0_mps	Sets the operating conditions to best
2.5_mps	Sets the operating conditions to typical
custom	Sets the operating conditions to worst

-heatsink {*value*}

Specifies the value of the operating condition. The following table shows the acceptable values for this argument:

Value	Description
none	No heat sink
custom	Sets a custom heat sink size
10mm_Low_Profile	10 mm heat sink
15mm_Low_Profile	15 mm heat sink
20mm_High_Profile	20 mm heat sink

-boardmodel {*value*}

Specifies your board model. The following table shows the acceptable values for this argument:

Value	Description
None_Conservative	No board model, conservative routing
JEDEC_2s2p	JEDEC 2s2p board model

-teta_ja {*decimal_value*}

Optional; sets your teta ja value; must be a positive decimal

-teta_sa {*decimal_value*}

Optional; sets your teta sa value; must be a positive decimal.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example sets the operating conditions to best:

```
set_operating_conditions -airflow "still_air" -heatsink "None" -boardmodel
"None_Conservative "
```

See Also

Tcl documentation conventions

smartpower_set_process

Tcl command; sets the process used in SmartPower to one of the pre-defined types.

```
smartpower_set_process -process {value}
```

Arguments

-process {*value*}

Specifies the value of the operating condition. The following table shows the acceptable values for this argument:

Value	Description
Typical	Sets the process for SmartPower to typical
Maximum	Sets the process for SmartPower to maximum

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

This example sets the operating conditions to typical:

```
smartpower_set_process -process {Typical}
```

See Also

Tcl documentation conventions

smartpower_set_temperature_opcond

Tcl command; sets the temperature in the operating conditions to one of the pre-defined types.

```
smartpower_set_temperature_opcond -use{value}
```

Arguments

`-use{value}`

Specifies the temperature in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the temperature in the operating conditions as specified in your Project Settings.
design	Sets the temperature in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the temperature in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

This example sets the temperature in the operating conditions as specified in the custom mode-settings:

```
smartpower_set_temperature_opcond -use{mode}
```

See Also

Tcl documentation conventions

smartpower_set_voltage_opcond

Tcl command; sets the voltage in the operating conditions.

```
smartpower_set_voltage_opcond -voltage{value} -use{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions for VDDI 2.5
VPP	Sets the voltage operating conditions for VPP

`-use{value}`

Specifies the voltage in the operating conditions for each voltage supply. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the voltage in the operating conditions as specified in your Project Settings.
design	Sets the voltage in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the voltage in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example sets the VCCA as specified in the SmartPower mode-specific settings:

```
smartpower_set_voltage_opcond -voltage{vcca} -use{mode}
```

See Also

Tcl documentation conventions

smartpower_temperature_opcond_set_design_wide

Tcl command; sets the temperature for SmartPower design-wide operating conditions.

```
smartpower_temperature_opcond_set_design_wide -best{value} -typical{value} -worst{value} -
thermal_mode{value}
```

Arguments

`-best{value}`

Specifies the best temperature (in degrees Celsius) used for design-wide operating conditions.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) used for design-wide operating conditions.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) used for design-wide operating conditions.

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example sets the temperature for design-wide operating conditions to Best 20, Typical 30, and Worst 60:

```
smartpower_temperature_opcond_set_design_wide -best{20} -typical{30} -worst{60}
```

See Also

Tcl documentation conventions

smartpower_temperature_opcond_set_mode_specific

Tcl command; sets the temperature for SmartPower mode-specific operating conditions.

```
smartpower_temperature_opcond_set_mode_specific -opmode{value} -thermal_mode{value} -
best{value} -typical{value} -worst{value} -thermal_mode{value}
```

Arguments

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to **Standby**
Flash*Freeze	The operating mode is set to Flash*Freeze

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

`-best{value}`

Specifies the best temperature (in degrees Celsius) for the selected mode.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) for the selected mode.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) for the selected mode.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example sets the temperature for mode-specific operating conditions for mode1:

```
smartpower_temperature_opcond_set_mode_specific -mode{mode1} -best{20} -typical{30} -worst{60}
```

See Also

Tcl documentation conventions

smartpower_voltage_opcond_set_design_wide

Tcl command; sets the voltage settings for SmartPower design-wide operating conditions.

```
smartpower_voltage_opcond_set_design_wide -voltage{value} -best{value} -typical{value} -worst{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions for VDDI 2.5
VPP	Sets the voltage operating conditions for VPP
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for design-wide operating conditions.

`-typical{value}`

Specifies the typical voltage used for design-wide operating conditions.

`-worst{value}`

Specifies the worst voltage used for design-wide operating conditions.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example sets VCCA for design-wide to best 20, typical 30 and worst 40:

```
smartpower_voltage_opcond_set_design_wide -voltage{VCCA} -best{20} -typical{30} -
worst{40}
```

See Also

Tcl documentation conventions

smartpower_voltage_opcond_set_mode_specific

Tcl command; sets the voltage settings for SmartPower mode-specific use operating conditions.

```
smartpower_voltage_opcond_set_mode_specific -opmode{value} -voltage{value} -best{value} -
typical{value} -worst{value}
```

Arguments

`-opmode {value}`

Use this option to specify the mode from which the operating conditions are extracted to generate the report.

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to **Standby**
Flash*Freeze	The operating mode is set to Flash*Freeze

`-voltage{value}`

Specifies the voltage in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions for VDDI 2.5
VPP	Sets the voltage operating conditions for VPP
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for mode-specific operating conditions.

`-typical{value}`

Specifies the typical voltage used for mode-specific operating conditions.

`-worst{value}`

Specifies the worst voltage used for mode-specific operating conditions.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

This example sets the voltage for the static mode and sets best to 20, typical to 30 and worst to 40:

```
smartpower_voltage_opcond_set_mode_specific -opmode{active} -voltage{VCCA} -best{20} -
typical{30} -worst{40}
```

See Also

Tcl documentation conventions

FlashPro Express Tcl Commands

close_project

Closes the FlashPro or FlashPro Express project.

```
close_project
```

Arguments

None

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

```
close_project
```

complete_prog_job

Tcl command; completes the current open job and generates a Job Status container including cryptographically signed Job Ticket end certifiers and Certificates of Conformance (if enabled) of the programmed devices. It archives ticket data from the HSM database. The resultant Job Status container can be imported into Job Manager and validated using U-HSM. If the job status file is not specified, the information is printed in the log window, and no Job Status container is created for subsequent verification.

The HSM Job can only be completed if the number of devices in each HSM ticket has been exhausted. If devices remain, the job can only be terminated by using the “-terminate” option.

```
complete_prog_job [-job_status_file path]\
[-terminate]
```

NOTE: This command will fail if there are devices left in any HSM ticket and the terminate option is not used.

Arguments

`[-job_status_file path]`

Full path to the output Job Status container which contains End-Job Certifier and CofCs. If not specified, information will be printed in the log window.

`[-terminate]`

This option will terminate the HSM job even if there are devices left in any HSM ticket. This parameter is optional if the number of devices in all tickets have been exhausted.

Supported Families

SmartFusion2, IGLOO2

See Also:

SPPS User Guide

User HSM Installation Guide

Manufacturer HSM Installation Guide

Job Manager User Guide

configure_flashpro3_prg

Changes FlashPro3 programmer settings.

```
configure_flashpro3_prg [-vpump {ON|OFF}] [-clk_mode {discrete_clk|free_running_clk}] [-force_freq {ON|OFF}] [-freq {freq}]
```

Arguments

-vpump {ON|OFF}

Enables FlashPro programmer to drive VPUMP. Set to ON to drive VPUMP.

-clk_mode {discrete_clk|free_running_clk}

Specifies free running or discrete TCK.

-force_freq {ON|OFF}

Forces the FlashPro software to use the TCK frequency specified by the software rather than the TCK frequency specified in the programmer file.

-freq {*freq*}

Specifies the TCK frequency in MHz.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example sets the VPUMP option to ON, TCK to free running, and uses the TCK frequency specified in the programmer file (force_freq is set to OFF):

```
configure_flashpro3_prg -vpump {ON} -clk_mode {free_running_clk} -force_freq {OFF} -freq {4}
```

The following example sets VPUMP to ON, TCK to discrete, forces the FlashPro software to use the TCK frequency specified in the software (-force_freq is set to ON) at a frequency of 2 MHz.

```
configure_flashpro3_prg -vpump {ON} -clk_mode {discrete_clk} -force_freq {ON} -freq {2}
```

configure_flashpro4_prg

Changes FlashPro4 programmer settings.

```
configure_flashpro4_prg [-vpump {ON|OFF}] [-clk_mode {discrete_clk|free_running_clk}] [-force_freq {ON|OFF}] [-freq {freq}]
```

Arguments

-vpump {ON|OFF}

Enables FlashPro4 programmer to drive VPUMP. Set to ON to drive VPUMP.

`-clk_mode {discrete_clk|free_running_clk}`

Specifies free running or discrete TCK.

`-force_freq {ON|OFF}`

Forces the FlashPro software to use the TCK frequency specified by the software rather than the TCK frequency specified in the programmer file.

`-freq {freq}`

Specifies the TCK frequency in MHz.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example sets the VPUMP option to ON and uses a free running TCK at a frequency of 4 MHz (force_freq is set to OFF).

```
configure_flashpro4_prg -vpump {ON} -clk_mode {free_running_clk} -force_freq {OFF} -freq {4}
```

The following example sets the VPUMP option to ON, uses a discrete TCK and sets force_freq to ON at 2 MHz.

```
configure_flashpro4_prg -vpump {ON} -clk_mode {discrete_clk} -force_freq {ON} -freq {2}
```

configure_flashpro5_prg

Tcl command; changes FlashPro5 programmer settings.

```
configure_flashpro5_prg [-vpump {ON|OFF}] [-clk_mode {free_running_clk}]
[-programming_method {jtag | spi_slave}] [-force_freq {ON|OFF}] [-freq {freq}]
```

Arguments

`-vpump {ON|OFF}`

Enables FlashPro5 programmer to drive VPUMP. Set to ON to drive VPUMP. Default is ON.

`-clk_mode {free_running_clk}`

Specifies free running TCK. Default is free_running_clk.

`-programming_method {jtag | spi_slave}`

Specifies the programming method to use. Default is jtag.

Note: spi_slave works only with SmartFusion2 and IGLOO2.

`-force_freq {ON|OFF}`

Forces the FlashPro software to use the TCK frequency specified by the software rather than the TCK frequency specified in the programmer file. Default is OFF.

`-freq {freq}`

Specifies the TCK frequency in MHz. Default is 4.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example sets the VPUMP option to ON and uses a free running TCK at a frequency of 4 MHz (force_freq is set to OFF).

```
configure_flashpro5_prg -vpump {ON} -clk_mode {free_running_clk} -force_freq {OFF} -freq {4}
```

The following example sets the VPUMP option to ON, uses a free running TCK and sets force_freq to ON at 2 MHz.

```
configure_flashpro5_prg -vpump {ON} -clk_mode {free_running_clk} -force_freq {ON} -freq {2}
```

configure_flashpro6_prg

Tcl command; changes FlashPro6 programmer settings.

```
configure_flashpro6_prg
[-force_freq {ON|OFF}] [-freq {freq}]
```

Arguments

`-force_freq {ON|OFF}`

Forces the FlashPro software to use the TCK frequency specified by the software rather than the TCK frequency specified in the programmer file. Default is OFF.

`-freq {freq}`

Specifies the TCK frequency in MHz. Default is 4.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example sets TCK at a frequency of 4 MHz and sets force_freq to OFF.

```
configure_flashpro6_prg -force_freq {OFF} -freq {4}
```

The following example sets TCK at a frequency of 2 MHz and sets force_freq to ON.

```
configure_flashpro6_prg -force_freq {ON} -freq {2}
```

create_job_project

Tcl command; creates a Flashpro Express job using the programming job exported from Libero.

```
create_job_project -job_project_location location -job_file path -overwrite 0|1
```

Arguments

`-job_project_location location`

Specifies the location for your FlashPro Express job project.

`-job_file path`

Path to the Libero job file that is used as input to create the Flashpro Express job project.

`-overwrite 0|1`

Set value to 1 to overwrite your existing job project. .

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example creates a job project named test.job in the \fpexpress directory. It does not overwrite the existing job project.

```
 create_job_project \
-job_project_location {D:\fpexpress} \
-job_file {D:\test\designer\test\export\test.job} –overwrite 0\
```

dump_tcl_support

Unloads the list of supported FlashPro or FlashPro Express Tcl commands.

```
dump_tcl_support -file {file}
```

Arguments

`-file {file}`

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example dumps your Tcl commands into the file 'tcldump.tcl'

```
dump_tcl_support -file {tcldump.tcl}
```

enable_serialization

This Tcl command enables or disables serialization programming.

```
enable_serialization –name {device_name} -enable {true|false}
```

Arguments

`-name`

Specifies the device name.

`-enable`

Enables (true) or disables (false) serialization programming.

Exceptions

Must be a Microsemi Device

Supported Families

SmartFusion2, IGLOO2

Example

```
enable_serialization -name M2S/M2GL050{T|S|TS} -enable true
```

get_job_status

Tcl command; exports status of current open job. The job status contains number of devices left for each HSM ticket. If job status file is not specified, the information is printed in the log window.

```
get_job_status [-job_status_file path] \
               [-archive]
```

Arguments

`[-job_status_file path]`

Path to the output FlashPro Express job status container. The job status file can be sent to the Job Manager application and Certificates of Conformance (if available) validated using the U-HSM.

`[-archive]`

Moves the HSM ticket log files from the HSM ticket database to the HSM ticket archive. The archive folder was specified during HSM installation and setup.

NOTE: If no job_status_file is specified, the archive option prints the Certificates of Conformance in the log window without exporting them.

Supported Families

SmartFusion2, IGLOO2

See Also

SPPS User Guide

Manufacturer HSM Installation Guide

User HSM Installation Guide

Job Manager User Guide

open_project

Opens a FlashPro or FlashPro Express project.

```
open_project -project {project}
```

Arguments

`-project {project}`

Specifies the location and name of the project you wish to open.

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

None

Example

Opens the 'FPPrj1.pro' project from the FPProject1 directory

```
open_project -project {./FPProject1/FPPrj1.pro}
```

ping_prg

Pings one or more programmers.

```
ping_prg (-name {name})*
```

Arguments

`-name {name}`

Specifies the programmer to be pinged. Repeat this argument for multiple programmers.

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

None

Example

The following example pings the programmers 'FP300085' and 'FP30086'.

```
ping_prg -name {FP300085} -name {FP300086}
```

process_job_request

Tcl command; processes a job request received from Job Manager. It is part of the Job Ticket generation process.

NOTE1: This command does not require a FlashPro Express project to be created or opened.

NOTE2: HSM parameters must be configured using set_hsm_params before processing job request.

```
process_job_request -request_file path \
        -reply_file path \
        [-overwrite_reply {TRUE | FALSE}]
```

Arguments

`-request_file path`

Full file name of job request file.

`-reply_file path`

Full file name of job reply file.

`-overwrite_reply {TRUE | FALSE}`

TRUE allows overwriting of any pre-existing reply_file.

Supported Families

SmartFusion2 and IGLOO2

Example

```
process_job_request \
   -request_file {D:/flashpro_files/jobmgr_project12/cm_request.req} \
   -reply_file {D:/flashpro_files/jobmgr_project12/cm_reply.rep} \
   -overwrite_reply {TRUE}
```

See Also

SPPS User Guide

Job Manager User Guide

set_hsm_params

refresh_prg_list

Refreshes the programmer list. This is most often used to have FlashPro or FlashPro Express detect a programmer that you have just connected.

```
refresh_prg_list
```

Arguments

None

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

```
refresh_prg_list
```

remove_hsm_tickets

Tcl command; removes HSM tickets from the HSM using one of the following methods:

- By specifying the job reply file in which case all tickets that are in the reply file will be deleted.
- By specifying each of the ticket IDs value in hexadecimal string.

```
remove_hsm_tickets [-reply_file path] \
[-ticket_ids ticketID+] \
[-all]
```

Arguments

`-reply_file path`

Full file name of job request file.

`-ticket_ids ticketID+`

Hex value of each ticket ID to be removed.

Supported Families

SmartFusion2, IGLOO2

Example

```
remove_hsm_tickets \
   -reply_file {D:flashpro_filesjobmgr_project12cm_reply.rep}
remove_hsm_tickets \
   -ticket_ids {0000000000000000000000899f252d9fb55442aa7e
00000000000000000000b6f385c6a9eeca69705c
00000000000000000000ed5702d0b767ba686b82}
```

NOTES

- This command should be used very carefully since it removes HSM tickets, rendering any FlashPro Express jobs based on those tickets to be unusable.
- This command does not require a FlashPro Express project to be created or opened.

See Also

SPPS User Guide

Job Manager User Guide

remove_prg

Removes the programmer from the programmer list.

```
remove_prg (-name {name})*
```

Arguments

`-name {name}*`

Specifies the programmer to be removed. You can repeat this argument for multiple programmers.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example removes the programmer '03178' from the programmer list:

```
remove_prg (name {03178})*
```

run_selected_actions

Runs the selected action on the specified programmer and returns the exit code from the action. If no programmer name is specified, the action is run on all connected programmers. Only one exit code is returned, so return code cannot be used when action is run on more than one programmer. A programming file must be loaded.

```
run_selected_actions [(-name {name})*]
```

Arguments

`-name {name}`

Optional argument that specifies the programmer name. You can repeat this argument for multiple programmers.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example runs the selected actionS on the programmers 'FP30085' and 'FP30086'.

```
run_selected_actions -name {FP300085} -name {FP300086}
```

Example using return code:

```
if {[catch {run_selected_actions} return_val]} {puts "Error running Action"} else {puts "exit code $return_val"}
```

Example returning exit code to the command line (returns exit 99 on script failure, otherwise returns exit code from selected action):

```
if {[catch {run_selected_actions} return_val]}{exit 99} else {exit $return_val}
```

save_log

Saves the log file.

```
save_log -file {file}
```

Arguments

`-file {file}`

Specifies the log filename.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following example saves the log file with the name 'my_logfile1.log':

```
save_log -file {my_logfile1.log}
```

save_project

Saves the FlashPro or FlashPro Express project.

```
save_project
```

Arguments

None

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

None

Example

```
save_project
```

scan_chain_prg

In single mode, this command runs scan chain on a programmer.

In chain mode, this command runs scan and check chain on a programmer if devices have been added in the grid.

```
scan_chain_prg [(-name {name})+]
```

Arguments

`-name {name}`

Specifies the programmer name.

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

None

Example

The following example runs scan chain on a single programmer (single mode) named '21428':

```
scan_chain_prg -name {21428}
```

select_serial_range

This Tcl command selects the range of indexes to program.

```
select_serial_range -name device_name -from_data start_index_to_program -to_data
end_index_to_program
```

Arguments

`-name`

Specifies the device name.

`-from_data`

Specifies the start index to program.

`-to_data`

Specifies the end index.

Supported Families

SmartFusion2, IGLOO2

Exceptions

Must be a Microsemi Device

Example

```
select_serial_range -name M2S/M2GL050{T|S|TS} -from_data 3 -to_data 5
```

self_test_prg

Runs Self-Test on a programmer.

```
self_test_prg (-name {name})*
```

Arguments

`-name {name}`

Specifies the programmer name. You can repeat this argument for multiple programmers.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

The following examples runs the self test on the programmer '30175':

```
self_test_prg (-name {30175})*
```

set_hsm_params

Tcl command; saves the HSM parameters for the FlashPro Express application. These parameters remain in effect until overridden by another invocation of this command.

```
set_hsm_params -hsm_server_name hsm_server \
               -hsm_type_u {TRUE|FALSE} \
               -m_hsm_uuid m_uuid \
               -ftp_username ftp_username \
               -ftp_password ftp_password
```

NOTE1: The HSM parameters are persistent between multiple FlashPro Express sessions on the same computer.

NOTE2: HSM parameters only need to be set for HSM flow jobs.

Arguments

`-hsm_server hsm_server`

Name or IP address of HSM server computer

`-hsm_type_u {TRUE|FALSE}`

TRUE FlashPro Express will use the Manufacturer features of the User HSM.

FALSE FlashPro Express will use a Manufacturer HSM.

`-m_hsm_uuid m_uuid`

UUID of HSM to be used for FlashPro Express tasks.

`-ftp_username ftp_username`

User name to access the HSM files via FTP server.

`-ftp_password ftp_password`

Password to access the HSM files via FTP server.

Supported Families

SmartFusion2 and IGLOO2

Example

```
set_hsm_params -hsm_server_name {10.241.140.224} \
               -hsm_type_u {0} \
               -m_hsm_uuid {00000000000000000000000000000000000000000002} \
               -ftp_username {hsm} \
               -ftp_password {hsm}
```

set_prg_name

Changes the user name of a programmer.

```
set_prg_name -name {name} -new_name {new_name}
```

Arguments

`-name {name}`

Identifies the old programmer name.

`-new_name {new_name}`

Specifies the new programmer name.

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

None

Example

The following example changes the name of the programmer 'FP300086' to 'FP3Prg2':

```
set_prg_name -name {FP300086} -new_name {FP3Prg2}
```

set_programming_action

Selects the action for a device. The device name parameter must be specified only in chain programming mode. A programming file must be loaded. The device must be a Microsemidevice.

```
set_programming_action [-name {name}] -action {action}
```

Arguments

`-name {name}`

Specifies the device name.

`-action {action}`

Specifies the action.

Supported Families

SmartFusion2
IGLOO2
RTG4

Exceptions

Must be a Microsemi device

Example

The following example sets the programming action in single programming mode:

```
set_programming_action -action {PROGRAM}
```

And in chain programming mode:

```
set_programming_action -name {MyDevice1} -action {ERASE}
```

set_programming_file

Sets the programming file for a device. Either the *file* or the *no_file* flag must be specified. A programming file must be loaded. The device must be a Microsemi device .

```
set_programming_file [-name {name}] [-file {file}] [-no_file { }]
```

Arguments

`-name {name}`

Specifies the device name. This argument must be specified only in chain programming mode.

`-file {file}`

Specifies the programming file.

`-no_file`

Specifies to unload the current programming file.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

Must be a Microsemi device.

Examples

In single programming mode:

```
set_programming_file -file {e:/design/pdb/TopA3P250.pdb}
```

In chain programming mode:

```
set_programming_file -name {MyDevice2} -file {e:/design/pdb/TopA3P250.pdb}
set_programming_file -name {MyDevice1} -no_file
```

set_serialization_log_file

This Tcl command sets the path and name of the serialization log file.

```
set_serialization_log_file -file {log_file_path}
```

Arguments

`-file`

Specifies the serialization log file path and name

Supported Families

SmartFusion2, IGLOO2

Exceptions

Must be a Microsemi Device

Example

```
set_serialization_log_file -file {C:/local_z_folder/work/my_serial_log}
```

SmartDebug Tcl Commands

add_probe_insertion_point

This Tcl command adds probe points to be connected to user-specified I/Os for probe insertion flow.

```
add_probe_insertion_point -net net_name -driver driver -pin package_pin_name -port port name
```

Arguments

`-net net_name`

Name of the net used for probe insertion.

`-driver driver`

Driver of the net.

`-pin package_pin_name`

Package pin name (i.e. I/O to which the net will be routed during probe insertion).

`-port port_name`

User-specified name for the probe insertion point.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
add_probe_insertion_point -net {count_out_c[0]} -driver {Counter_8bit_0_count_out[0]:Q} -
pin {H5} -port {Probe_Insert0}
```

add_to_probe_group

Tcl command; adds the specified probe points to the specified probe group.

```
add_to_probe_group -name probe_name -group group_name
```

Arguments

`-name probe_name`

Specifies one or more probes to add.

`-group group_name`

Specifies name of the probe group.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
add_to_probe_group -name out[5]:out[5]:Q \
                   -name grp1.out[3]:out[3]:Q \
                   -name out.out[1].out[1]:Q \
                   -group my_new_grp
```

construct_chain_automatically

This Tcl command automatically starts chain construction for the specified programmer.

```
construct_chain_automatically -name {programmer_name}
```

Arguments

-name

Specify the device (programmer) name. This argument is mandatory.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

For a single programmer:

```
construct_chain_automatically -name {21428}
```

See Also

scan_chain_prg

enable_device

set_debug_programmer

set_device_name

set_programming_file

set_programming_action

run_selected_actions

create_probe_group

Tcl command; creates a new probe group.

```
create_probe_group -name group_name
```

Arguments

-name *group_name*

Specifies the name of the new probe group.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
create_probe_group -name my_new_grp
```

ddr_read

Tcl command; reads the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR).

```
ddr_read -block ddr_name -name reg_name
```

Arguments

-block <fddr || mddr || east_fddr || west_fddr>

- Specifies which DDR configurator is used in the Libero design.
- SmartFusion2 and IGLOO2 - fddr and mddr
- RTG4 - east_fddr and west_fddr

-name *register_name*

- Specifies which configuration registers need to be read.
- A complete list of registers is available in the DDR Interfaces User Guides for the respective families.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

Read DDR Controller register DDRC_DYN_REFRESH_1_CR for a configured FDDR block on a SmartFusion2 or IGLOO2 device:

```
ddr_read -block fddr -name DDRC_DYN_REFRESH_1_CR
```

Returns

Returns 16-bit hexadecimal value.

The result of the command in the example above will be:

```
Register Name: DDRC_DYN_REFRESH_1_CR Value: 0x1234
"ddr_read" command succeeded.
```

ddr_write

Tcl command; writes the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR).

```
ddr_write-block ddr_name -name reg_name -value hex_value
```

Arguments

-block <fddr || mddr || east_fddr || west_fddr>

- Specifies which DDR configurator is used in the Libero design.
- SmartFusion2 and IGLOO2 - fddr and mddr
- RTG4 - east_fddr and west_fddr

-name *register_name*

- Specifies which configuration registers need to be read.
- A complete list of registers is available in the DDR Interfaces User Guides for the respective families.

-value *hex_value*

- Specifies the value to be written into the specified register of a given block.
- Hex_value in the form of “0x12FA”.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

Write a 16-bit value DDR Controller register DDRC_DYN_REFRESH_1_CR for a configured FDDR block on a SmartFusion2 or IGLOO2 device:

```
ddr_write -block fddr -name DDRC_DYN_REFRESH_1_CR –value 0x123f
```

Returns

Returns if the command succeeded or failed to execute.

```
“ddr_write” command succeeded
```

delete_active_probe

Tcl command; deletes either all or the selected active probes.

Note: You cannot delete an individual probe from the Probe Bus.

```
delete_active_probe -all | -name probe_name
```

Arguments

-all

Deletes all active probe names.

-name *probe_name*

Deletes the selected probe names.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
delete –all      <- deletes all active probe names
delete -name out[5]:out[5]:Q \
       -name my_grp1.out[1]:out[1]:Q            #deletes the selected probe names
delete -name my_grp1 \
       -name my_bus             #deletes the group, bus and their members.
```

enable_device

This Tcl command enables or disables a device in the chain. When the device is disabled, it is bypassed. The device must be a Microsemi device.

```
enable_device -name {device_name} -enable {1 | 0}
```

Arguments

-name

Specify the device name. This argument is mandatory.

`-enable`

Specify the enable device. This argument is mandatory.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
enable_device -name {MPF300 (T_ES|TS_ES)} -enable 1
```

See Also

construct_chain_automatically
scan_chain_prg
set_debug_programmer
set_device_name
set_programming_file
set_programming_action
run_selected_actions

event_counter

The event_counter Tcl command runs on signals that are assigned to channel A on the live probe, and displays the total events. It can be run before or after setting the live probe signal to channel A. The user specifies the duration to run the event_counter command.

```
event_counter -run -stop -after duration_in_seconds
```

Arguments

`-run`

Run event_counter.

`-stop`

Stop event_counter.

`-after duration_in_seconds`

Duration to stop event_counter. Specified by the user. This argument is required when `-stop` is specified.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set_live_probe -probeA {count_out_c[0]:Counter_8bit_0_count_out[0]:Q} -probeB {}
event_counter -run
event_counter -stop -after 10
```

Output

```
Device ID Code = 2F8071CF
The 'read_id_code' command succeeded.
Live probes have been assigned.
Channel A: count_out_c[0]:Counter_8bit_0_count_out[0]:Q
Channel B: Not specified

The 'set_live_probe' command succeeded.

Event Counter = Activated
The 'event_counter' command succeeded.

Event Counter = Stopped
Total Events = 1603561
The 'event_counter' command succeeded.
The Execute Script command succeeded.
```

export_smart_debug_data

Tcl command; exports debug data for the SmartDebug application.

```
export_smart_debug_data [device_components] [bitstream_components] [-file_name {file} [-
export_dir {dir}] [-force_rtg4_otp 0 | 1]
```

The command corresponds to the Export SmartDebug Data tool in Libero. The command creates a file with the extension “ddc” that contains data based on selected options. This file is used by SmartDebug (standalone application) to create a new SmartDebug project, or it can be imported into a device in SmartDebug (standalone application).

- If you do not specify any design components, all components available in the design will be included by default except the bitstream components.
- The generate_bitstream parameter is required if you want to generate bitstream file and include it in the exported file.
    - You must specify the bitstream components you want to include in the generated bitstream file or all available components will be included.
    - If you choose to include bitstream, and the design has custom security, the custom security bitstream component must be included.

Arguments

device_components

The following device components can be selected. Specify "1" to include the component, and "0" if you do not want to include the component.

```
-probes <1|0>
-package_pins <1|0>
-memory_blocks <1|0>
-envm_data <1|0>
-security_data <1|0>
-chain <1|0>
-programmer_settings <1|0>
-io_states <1|0>
```

bitstream_components

The following bitstream components can be selected. Specify "1" to include the component, and "0" if you do not want to include the component.

```
-generate_bitstream <1|0>
-bitstream_security <1|0>
```

-bitstream_fabric <1|0>
-bitstream_envm <1|0>

-file_name *file*

Name of exported file with extension “ddc”.

-export_dir *dir*

Location where DDC file will be exported. If omitted, design export folder will be used.

-force_rtg4_otp *0 | 1*

Enforces the use of one-time programming (OTP).

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

The following examples shows the export_smart_debug_data command with all parameters.

SmartFusion2, IGLOO2, RTG4 example:

```
export_smart_debug_data \
-file_name {sd1} \
-export_dir {d:\sd_prj\test3T\designer\sd1\export} \
-force_rtg4_otp 1 \
-probes 1 \
-package_pins 0 \
-memory_blocks 1 \
-envm_data 0 \
-security_data 1 \
-chain 1 \
-programmer_settings 1 \
-ios_states 1 \
-generate_bitstream 0 \
-bitstream_security 0 \
-bitstream_fabric 0 \
-bitstream_envm 0
```

The following example shows the command with no parameters:

```
export_smart_debug_data
```

fhb_control

This Tcl command provides FPGA Hardware Breakpoint (FHB) feature capability for SmartDebug.

```
fhb_control
   -halt -clock_domain clkDomName(s)/all
   -run -clock_domain clkDomName(s)
   -step number_of_steps -clock_domain clkDomName(s)
   -reset -clock_domain clkDomName(s)
   -arm_trigger -trigger_signal liveProbePoint -trigger_edge_select rising -delay value -
clock_domain clkDomName(s)
   -disarm_trigger -clock_domain clkDomName(s)/all
   -capture_waveform number_of_steps -vcd_file target_file_name
   -clock_domain_status -clock_domain clkDomName(s)/all
```

Arguments

-halt

Specifies to halt the clock.

-clock_domain *clkDomName(s)/all*

Specifies clock domain names to halt. Can be single or multiple clock domains, halted in order specified by user.

-run

Specifies to run the clock.

-clock_domain *clkDomName(s)*

Specifies clock domain names to run. Can be single or multiple clock domains, releasing the user clock based on order specified.

-step *number_of_steps*

Specifies to step the clock “number_of_steps” times. Minimum value is 1.

-clock_domain *clkDomName(s)*

Specifies clock domain names to step. Can be single or multiple clock domains.

-reset

Specifies to reset FHB configuration for the specified clock domain.

-clock_domain *clkDomName(s)*

Specifies clock domain names to reset. Can be single or multiple clock domains.

-arm_trigger

Specifies to arm FHB configuration for the specified clock domain.

-trigger_signal *liveProbePoint*

Set the trigger signal to arm the FHBs.

-trigger_edge_select *rising*

Specifies the trigger signal edge to arm the FHBs. FHBs will be armed on rising edge of trigger signal.

-delay *value*

-clock_domain *clkDomName(s)*

Specifies clock domain names to be armed by the trigger signal. Can be single or multiple clock domains.

-disarm_trigger

Specifies to disarm FHB configuration for the specified clock domain.

-clock_domain *clkDomName(s)*

Specifies clock domain names to be reset by the trigger signal. Can be single or multiple clock domains.

-capture_waveform *number_of_steps*

Specifies to capture waveform of all the added signals to active probes in the specified clock domain for number_of_steps.

- vcd_file *target_file_name*

Target file to save the data and see the waveform.

-clock_domain_status *clkDomName(s)/all*

Specifies to read and display status of specified clock domain(s). Can be single or multiple clock domains.

Supported Families

SmartFusion2

IGLOO2

RTG4

Examples

```
fhb_control -halt -clock_domain {“FCCC_0/GL0_INST “ “FCCC_0/GL1_INST” }
fhb_control -run -clock_domain {“FCCC_0/GL0_INST “ “FCCC_0/GL1_INST” }
fhb_control -step -clock_domain {“FCCC_0/GL0_INST “ “FCCC_0/GL1_INST” }
fhb_control -reset -clock_domain {“FCCC_0/GL0_INST “ “FCCC_0/GL1_INST” }
```

```
fhb_control -arm_trigger -trigger_signal {q_0_c[14]:count_1_q[14]:Q}
-trigger_edge_select {rising} - delay 0 - clock_domain {"FCCC_0/GL0_INST"}
fhb_control -disarm_trigger -trigger_signal {q_0_c[14]:count_1_q[14]:Q}
-trigger_edge_select {rising} - delay 0 - clock_domain {"FCCC_0/GL0_INST"}
fhb_control -capture_waveform {10} -vcd_file {D:/wvf_location/waveform.vcd}
fhb_control - clock_domain_status - clock_domain { "FCCC_0/GL0_INST" "FCCC_0/GL1_INST"
"FCCC_0/GL2_INST" }
```

frequency_monitor

The frequency_monitor Tcl command calculates the frequency of a signal that is assigned to live probe A.

```
run_frequency_monitor -signal signal_name -time duration
```

Arguments

`-signal signal_name`

Specifies the signal name.

`-time duration`

Specifies the duration to run the command. The value can be 0.1, 1, 5, 8, or 10.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
run_frequency_monitor -signal {count_out_c[7]:Counter_8bit_0_count_out[7]:Q} -time {5}
```

Output

```
Device ID Code = 2F8071CF
The 'read_id_code' command succeeded.

Frequency = 0.192716 MHz
The 'run_frequency_monitor' command succeeded.
The Execute Script command succeeded.
```

get_programmer_info

This Tcl command lists the IDs of all FlashPRO programmers connected to the machine.

```
get_programmer_info
```

This command takes no arguments.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set a [get_programmer_info]
```

load_active_probe_list

Tcl command; loads the list of probes from the file.

```
load_active_probe_list -file file_path
```

Arguments

`-file file_path`

The input file location.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
load_active_probe_list -file "./my_probes.txt"
```

loopback_test

Tcl command; used to start and stop the loopback tests.

```
loopback_test [-deviceName device_name] -start -serdes num -lane num -type LoopbackType
loopback_test [-deviceName device_name] -stop -serdes num -lane num
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User's Guide for details).

`-start`

Starts the loopback test.

`-stop`

Stops the loopback test.

`-serdes num`

Serdes block number. Must be between 0 and 4 and varies between dies.

`-lane num`

Serdes lane number. Must be between 0 and 4

`-type LoopbackType`

Specifies the loopback test type. Must be *meso* (PCS Far End PMA RX to TX Loopback)

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
loopback_test -start -serdes 1 -lane 1 -type meso
loopback_test -start -serdes 0 -lane 0 -type plesio
loopback_test -start -serdes 1 -lane 2 -type parallel
loopback_test -stop -serdes 1 -lane 2
```

move_to_probe_group

Tcl command; moves the specified probe points to the specified probe group.

Note: Probe points related to a bus cannot be moved to another group.

```
move_to_probe_group -name probe_name -group group_name
```

Arguments

`-name probe_name`

Specifies one or more probes to move.

`-group group_name`

Specifies name of the probe group.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
move_to_probe_group -name out[5]:out[5]:Q \
                    -name grp1.out[3]:out[3]:Q \
                    -group my_grp2
```

prbs_test

Tcl command; used in PRBS test to start, stop, reset the error counter and read the error counter value.

```
prbs_test [-deviceName device_name] -start -serdes num -lane num [-near] -pattern PatternType
prbs_test [-deviceName device_name] -stop -serdes num -lane num
prbs_test [-deviceName device_name] -reset_counter -serdes num -lane num
prbs_test [-deviceName device_name] -read_counter -serdes num -lane num
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User Guide for details).

`-start`

Starts the prbs test.

`-stop`

Stops the prbs test.

`-reset_counter`

Resets the prbs error count value to 0.

`-read_counter`

Reads and prints the error count value.

`-serdes num`

Serdes block number. Must be between 0 and 4 and varies between dies.

`-lane num`

Serdes lane number. Must be between 0 and 4.

`-near`

Corresponds to near-end (on-die) option for prbs test. Not specifying implies off-die.

`-pattern PatternType`

The pattern sequence to use for PRBS test. It can be one of the following:

prbs7, *prbs11*, *prbs23*, or *prbs31*

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
prbs_test -start -serdes 1 -lane 0 -near -pattern prbs11
prbs_test -start -serdes 2 -lane 2 -pattern custom -value all_zeros
prbs_test -start -serdes 0 -lane 1 -near -pattern user -value 0x0123456789ABCDEF0123
```

program_probe_insertion

This Tcl command runs the probe insertion flow on the selected nets.

```
program_probe_insertion
```

This command takes no arguments.

Supported Families

SmartFusion2

IGLOO2

RTG4

ungroup

Tcl command; disassociates the probes as a group.

```
nngroup -name group_name
```

Arguments

`-name group_name`

Name of the group.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
ungroup -name my_grp4
```

read_active_probe

Tcl command; reads active probe values from the device. The target probe points are selected by the select_active_probe command.

```
read_active_probe [-deviceName device_name] [-name probe_name] [-group_name
bus_name|group_name] [-value_type b|h][-file file_path]
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration.

`-name probe_name`

Instead of all probes, read only the probes specified. The probe name should be prefixed with bus or group name if the probe is in the bus or group.

`-group_name bus_name | group_name`

Instead of all probes, reads only the specified buses or groups specified here.

`-value_type b | h`

Optional parameter, used when the read value is stored into a variable as a string.

b = binary

h = hex

`-file file_path`

Optional. If specified, redirects output with probe point values read from the device to the specified file.

Note: When the user tries to read at least one signal from the bus/group, the complete bus or group is read. The user is presented with the latest value for all the signals in the bus/group.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
read_active_probe -group_name {bus1}
read_active_probe -group_name {group1}
```

To save into variable:

```
set a [read_active_probe -group_name {bus_name} -value_type h]
```
#save read data in hex string

If read values are stored into a variable without specifying value_type parameter, it saves values as a binary string by default.

Example

```
set a [read_active_probe ]
```
#sets variable a as binary string of read values after read_active_probe command.

read_lsram (SmartFusion2, IGLOO2, RTG4)

Tcl command; reads a specified block of large SRAM from the device.

Physical block

```
read_lsram [-deviceName device_name] -name block_name [-fileName file_name]
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

`-name block_name`

Specifies the name for the target block.

`-fileName file_name`

Optional; specifies the output file name for the data read from the device.

Exceptions

- Array must be programmed and active
- Security locks may disable this function

Example

Reads the SRAM Block sram_block1 from the sf2 device and writes it to the file sram_block_output.

```
read_lsram [-deviceName sf2] -name sram_block1 [-file sram_block_output]
```

Logical block

```
read_lsram -logicalBlockName block_name -port port_name [-fileName filename]
```

Arguments

`-logicalBlockName block_name`

Specifies the name for the user defined memory block.

`-port port_name`

Specifies the port for the memory block selected. Can be either Port A or Port B.

`-file filename`

Optional; specifies the output file name for the data read from the device.

Example

```
read_lsram -logicalBlockName {Fabric_Logic_0/U2/F_0_F0_U1} -port {Port A}
```

read_usram (SmartFusion2, IGLOO2, RTG4)

Tcl command; reads a uSRAM block from the device.

Physical block

```
read_usram [-deviceName device_name] -name block_name [-fileName file_name]
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

`-name block_name`

Specifies the name for the target block.

`-fileName file_name`

Optional; specifies the output file name for the data read from the device.

Exceptions

- Array must be programmed and active
- Security locks may disable this function

Example

Reads the uSRAM Block usram_block2 from the sf2 device and writes it to the file sram_block_output.

```
read_usram [-deviceName sf2] -name usram_block2 [-fileName sram_block_output]
```

Logical block

```
read_usram -logicalBlockName block_name -port port_name [-file filename]
```

Arguments

`-logicalBlockName block_name`

Specifies the name for the user defined memory block.

`-port port_name`

Specifies the port for the memory block selected. Can be either Port A or Port B.

`-file filename`

Optional; specifies the output file name for the data read from the device.

Example

```
read_usram -logicalBlockName {Fabric_Logic_0/U3/F_0_F0_U1} -port {Port A}
```

remove_from_probe_group

Tcl command; removes the specified probe points from the group. That is, the removed probe points won't be associated with any probe group.

Note: Probes cannot be removed from the bus.

```
remove_from_probe_group -name probe_name
```

Arguments

`-name probe_name`

Specifies one or more probe points to remove from the probe group.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

The following command removes two probes from my_grp2.

```
Move_out_of_probe_group -name my_grp2.out[3]:out[3]:Q \
                        -name my_grp2.out[3]:out[3]:Q
```

remove_probe_insertion_point

This Tcl command deletes an added probe from the probe insertion UI.

```
remove_probe_insertion_point -net net_name -driver driver
```

Arguments

`-net net_name`

Name of the existing net which is added using the `add_probe_insertion_point` command.

`-driver` *`driver`*

Driver of the net.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
remove_probe_insertion_point -net {count_out_c[0]} -driver
{Counter_8bit_0_count_out[0]:Q}
```

run_selected_actions

This Tcl command is used to run the selected action for a device.

```
run_selected_actions
```

This command takes no arguments.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set_programming_action -name {MPF300(T_ES|TS_ES)} -action {DEVICE_INFO}
set_programming_action -name {M2S/M2GL090(T|TS|TV)} -action {ERASE}
```

See Also

construct_chain_automatically

scan_chain_prg

enable_device

set_debug_programmer

set_device_name

set_programming_file

set_programming_action

save_active_probe_list

Tcl command; saves the list of active probes to a file.

```
save_active_probe_list -file file_path
```

Arguments

`-file` *`file_path`*

The output file location.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
save_active_probe_list -file "./my_probes.txt"
```

scan_chain_prg

In single mode, this Tcl command runs scan chain on a programmer. In chain mode, this Tcl command runs scan and check chain on a programmer if devices have been added in the grid.

```
scan_chain_prg -name {programmer_name}
```

Arguments

-name

Specify the device (programmer) name. This argument is mandatory.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
scan_chain_prg -name {21428}
```

See Also

construct_chain_automatically
enable_device
set_debug_programmer
set_device_name.htm
set_programming_file
set_programming_action
run_selected_actions

select_active_probe

Tcl command; manages the current selection of active probe points to be used by active probe READ operations. This command extends or replaces your current selection with the probe points found using the search pattern.

```
select_active_probe [-deviceName device_name] [-name probe_name_pattern] [-reset true|false]
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration..

-name *probe_name_pattern*

Specifies the name of the probe. Optionally, search pattern string can specify one or multiple probe points. The pattern search characters "*" and "?" also can be specified to filter out the probe names.

-reset *true | false*

Optional parameter; resets all previously selected probe points. If name is not specified, empties out current selection.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

The following command selects three probes. In the below example, “grp1” is a group and “out” is a bus..

```
Select_active_probe -name out[5]:out[5]:Q
Select_active_probe -name out.out[1]:out[1]:Q \
                    -name out.out[3]:out[3]:Q \
                    -name out.out[5]:out[5]:Q
```

set_debug_programmer

This Tcl command is used to set the debug programmer.

```
set_debug_programmer -name {programmer_name}
```

Arguments

-name

Specify the programmer. This argument is mandatory.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
set_debug_programmer -name {S201YQST1V}
```

See Also

construct_chain_automatically
scan_chain_prg
enable_device
set_device_name
set_programming_file
set_programming_action
run_selected_actions

set_device_name

Tcl command that is used to set the device name.

```
set_device_name -name {device_name} -new_name {new_name}
```

Arguments

`-name`

Specify the device name. This argument is mandatory.

`-new_name`

Specify the new name for the device. This argument is mandatory.

Supported Families

SmartFusion2, IGLOO2, RTG4

See Also

construct_chain_automatically
scan_chain_prg
enable_device
set_debug_programmer
set_programming_file
set_programming_action
run_selected_actions

set_programming_action

This Tcl command is used to select the action for a device.

```
set_programming_action [-name {device_name}] -action {procedure_action}
```

Arguments

`-name`

Specify the device name. This argument is mandatory.

`-action`

Specify the programming action. This argument is mandatory.

Supported Families

SmartFusion2
IGLOO2
RTG4

Example

```
set_programming_action -name {MPF300(T_ES|TS_ES)} -action {DEVICE_INFO}
set_programming_action -name {M2S/M2GL090(T|TS|TV)} -action {ERASE}
```

See Also

construct_chain_automatically
scan_chain_prg
enable_device
set_debug_programmer
set_device_name
set_programming_file
run_selected_actions

set_programming_file

This Tcl command is used to set the programming file for a device. Either the file or the no_file flag must specified. A programming file must be loaded. The device must be a Microsemi device.

```
set_programming_file -name {device_name} -file {stapl_file_name_with_path}
```

Arguments

`-name`

Specify the device name. This argument is mandatory.

`-file`

Specify the *file* path. This argument is mandatory.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set_programming_file -name {MPF300(T_ES|TS_ES)} -file
{D:/export/CM1_PCIE_TOP_default_uic_12_200_0_12.stp}
```

See Also

construct_chain_automatically
scan_chain_prg

enable_device

set_debug_programmer

set_device_name

set_programming_action

run_selected_actions

serdes_lane_reset

Tcl command. In EPCS mode, this command resets the lane. In PCI mode, this command resets the lane, all other lanes in the link, and the corresponding PCIe controller. The result is shown in the log window/console.

```
serdes_lane_reset -serdes num -lane num
```

Arguments

`-serdes num`

The SERDES block number. It must be between 0 and varies between dies. It must be one of the SERDES blocks used in the design.

`lane num`

The SERDES lane number. It must be between 0 and 3. It must be one of the lanes enabled for the block in the design.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
serdes_lane_reset -serdes 0 -lane 0
```

In EPCS mode, resets Lane 0, for block 0. In PCI mode, resets Lane 0 for block 0, all other lanes in the same link for block 0

```
serdes_lane_reset -serdes 5 -lane 3
```

Errors

The following errors result in the failure of the Tcl command and the corresponding message on the smart debug log window:

When the “-serdes” parameter is not specified:

```
Error: Required parameter 'serdes' is missing.
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]
Error: The Execute Script command failed.
```

When the “-lane” parameter is not specified:

```
Error: Required parameter 'lane' is missing.
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]
Error: The Execute Script command failed.
```

When “block number” is not specified:

```
Error: Parameter 'serdes' has illegal value.
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]
Error: The Execute Script command failed.
```

When “lane number” is not specified:

```
Error: Required parameter 'lane' is missing.
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]
Error: The Execute Script command failed.
```

When “block number” is invalid:

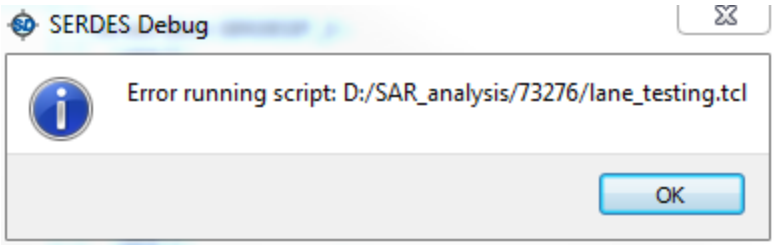
```
Error: Phy Reset: Serdes block number should be one of the following: 0
Error: The command 'serdes_lane_reset' failed.
Error: Failure when executing Tcl script. [Line 26]
Error: The Execute Script command failed.
```

Note: Only the SERDES blocks used the design will be mentioned in the above list.

When “lane number” is invalid:

```
Error: Phy Reset: Serdes lane number should be between 0 and 3.
Error: The command 'serdes_lane_reset' failed.
Error: Failure when executing Tcl script. [Line 26]
Error: The Execute Script command failed.
```

For all the above scenarios, the following message appears:



serdes_read_register

Tcl command; reads the SERDES register value and displays the result in the log window/console.

```
serdes_read_register -serdes num  [ -lane num ] -name REGISTER_NAME
```

Arguments

`-serdes num`

SERDES block number. Must be between 0 and and varies between dies.

`-lane num`

SERDES lane number. Must be between 0 and 3.

The lane number must be specified when the lane register is used. Otherwise, the command will fail.

When the lane number is specified along with the SYSTEM or PCIe register, the command will fail with an error message, as the lane is not applicable to them.

`-name REGISTER_NAME`

Name of the SERDES register.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
serdes_read_register -serdes 0 -name SYSTEM_SER_PLL_CONFIG_HIGH
serdes_read_register -serdes 0 -lane 0 -name CR0
```

serdes_write_register

UG0567: RTG4 High-Speed Serial Interfaces User Guide (includes all SERDES register names)

UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide

serdes_write_register

Tcl command; writes the value to the SERDES register. Displays the result in the log window/console.

```
serdes_write_register -serdes num  [-lane num ] -name REGISTER_NAME -value 0x1234
```

Arguments

`-serdes num`

SERDES block number. Must be between 0 and 5 and varies between dies.

`-lane num`

SERDES lane number. Must be between 0 and 3.

The lane number should be specified when the lane register is used. Otherwise, the command will fail.

When the lane number is specified along with the SYSTEM or PCIe register, the command will fail with an error message, as the lane is not applicable to them.

`-name REGISTER_NAME`

Name of the SERDES register.

`-value`

Specify the value in hexadecimal format.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
serdes_write_register -serdes 0 -name SYSTEM_SER_PLL_CONFIG_HIGH -value 0x5533
```

See Also

serdes_read_register.htm

UG0567: RTG4 High-Speed Serial Interfaces User Guide (includes all SERDES register names)

UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide

set_live_probe

Tcl command; set_live_probe channels A and/or B to the specified probe point(s). At least one probe point must be specified. Only exact probe name is allowed (i.e. no search pattern that may return multiple points).

```
set_live_probe [-deviceName device_name] [-probeA probe_name] [-probeB probe_name]
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug **(see SmartDebug user guide for details)**.

`-probeA probe_ name`

Specifies target probe point for the probe channel A.

-probeB *probe_ name*

Specifies target probe point for the probe channel B.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

- The array must be programmed and active
- Active probe read or write operation will affect current settings of Live probe since they use same probe circuitry inside the device
- Setting only one Live probe channel affects the other one, so if both channels need to be set, they must be set from the same call to set_live_probe
- Security locks may disable this function
- In order to be available for Live probe, ProbeA and ProbeB I/O's must be reserved for Live probe respectively

Example

Sets the Live probe channel A to the probe point A12 on device **sf2**.

```
set_live_probe [-deviceName sf2] [-probeA A12]
```

ungroup

Tcl command; disassociates the probes as a group.

```
nngroup -name group_name
```

Arguments

`-name group_name`

Name of the group.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
ungroup -name my_grp4
```

unset_live_probe

Tcl command; discontinues the debug function and clears both live probe channels (Channel A and Channel B). An all zeros value is shown for both channels in the oscilloscope.

Note: For RTG4, only one probe channel (Probe Read Data Pin) is available.

```
unset_live_probe  [-deviceName device_name]
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User Guide for Libero for details).

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

- The array must be programmed and active.
- Active probe read or write operation affects current of Live Probe settings, because they use the same probe circuitry inside the device.
- Security locks may disable this function.

Example

The following example unsets both live probe channels (Channel A and Channel B) from the device sf2.

```
unset_live_probes [-deviceName sf2]
```

write_active_probe

Tcl command; sets the target probe point on the device to the specified value. The target probe point name must be specified.

```
write_active_probe [-deviceName device_name] -name probe_name -value true|false
-group_name group_bus_name -group_value "hex-value" | "binary-value"
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration.

-name *probe_name*

Specifies the name for the target probe point. Cannot be a search pattern.

-value *true | false hex-value | binary-value*

Specifies values to be written.

True = High

False = Low

-group_name *group_bus_name*

Specify the group or bus name to write to complete group or bus.

-group_value *"hex-value" | "binary-value"*

Specify the value for the complete group or bus.

Hex-value format : " <size>'h<value>"

Binary-value format: " <size>'b<value>"

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
write_active_probe -name out[5]:out[5]:Q -value true <-- write to a single probe
write_active_probe -name grp1.out[3]:out[3]:Q -value low <-- write to a probe in the group
write_active_probe -group_name grp1 -group_value "8'hF0" <-- write the value to complete group
write_active_probe -group_name out -group_value "8'b11110000" \
                   -name out[2]:out[2]:Q -value true <-- write multiple probes at the same time.
```

write_lsram (SmartFusion2, IGLOO2, RTG4)

Tcl command; writes a seven bit word into the specified large SRAM location.

Physical block

```
write_lsram [-deviceName device_name] -name block_name] -offset offset_value -value value
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

-name *block_name*

Specifies the name for the target block.

-offset *offset_value*

Offset (address) of the target word within the memory block.

-value *value*

Nine-bit value to be written to the target location.

Exceptions

- Array must be programmed and active
- The maximum value that can be written is 0x1FF

- Security locks may disable this function

Example

Writes a value of 0x1A to the device sf2 in the block sram_block1 with an offset of 16.

```
write_lsram [-deviceName sf2] -name sram_block1 -offset 16 -value 0x1A
```

Logical block

```
write_lsram -logicalBlockName block_name -port port_name -offset offset_value -logicalValue
hexadecimal_value
```

Arguments

`-logicalBlockName block_name`

Specifies the name for the user defined memory block.

`-port port_name`

Specifies the port for the memory block selected. Can be either Port A or Port B.

`-offset offset_value`

Offset (address) of the target word within the memory block.

`-logicalValue hexadecimal_value`

Specifies the hexadecimal value to be written to the memory block. Size of the value is equal to the width of the output port selected.

Example

```
write_lsram -logicalBlockName {Fabric_Logic_0/U2/F_0_F0_U1} -port {Port A} -offset 1 -
logicalValue {00FFF}
```

write_usram (SmartFusion2, IGLOO2, RTG4)

Tcl command; writes a seven bit word into the specified uSRAM location.

Physical block

```
write_usram [-deviceName device_name] -name block_name] -offset offset_value -value value
```

Arguments

`-deviceName device_name`

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

`-name block_name`

Specifies the name for the target block.

`-offset offset_value`

Offset (address) of the target word within the memory block.

`-value value`

Nine-bit value to be written.

Exceptions

- Array must be programmed and active
- The maximum value that can be written is 0x1FF

- Security locks may disable this function

Example

Writes a value of 0x1A to the device sf2 in the block usram_block2 with an offset of 16.

```
write_usram [-deviceName sf2] -name usram_block2 -offset 16 -value 0x1A
```

Logical block

```
write_usram -logicalBlockName block_name -port port_name -offset offset_value -logicalValue
hexadecimal_value
```

Arguments

`-logicalBlockName block_name`

Specifies the name for the user defined memory block.

`-port port_name`

Specifies the port for the memory block selected. Can be either Port A or Port B.

`-offset offset_value`

Offset (address) of the target word within the memory block.

`-logicalValue hexadecimal_value`

Specifies the hexadecimal value to be written to the memory block. Size of the value is equal to the width of the output port selected.

Example

```
write_usram -logicalBlockName {Fabric_Logic_0/U3/F_0_F0_U1} -port {Port A} -offset 1 -
logicalValue {00FFF}
```

Configure JTAG Chain Tcl Commands

These commands take a script that contains JTAG chain configuration-specific Tcl commands and passes them to FlashPro Express for execution.

Note that these commands cannot be executed directly from Libero.

add_actel_device

Adds an Actel device to the chain. Either the *file* or *device* parameter must be specified. Chain programming mode must have been set.

```
add_actel_device [-file {filename}] [-device {device}] -name {name}
```

Arguments

Where:

`-file{filename}`

Specifies a programming filename.

`-device``{device}`

Specifies the device family (such as MPF300).

`-name``{name}`

Specifies the device user name.

Exceptions

None

Example

```
add_actel_device -file {e:/design/stp/TOP.stp} -name {MyDevice1}
add_actel_device -device {MPF300} -name {MyDevice2}
```

add_non_actel_device

Adds a non-Actel device in the chain. Either the file, or (-tck And -ir) parameters must be specified. The Chain programming mode must have been set.

```
add_non_actel_device [-file {file}] [-ir {ir}] [-tck {tck}] [-name {name}]
```

Arguments

`-file {filename}`

Specifies a BSDL file.

`-ir {ir}`

Specifies the IR length.

`-tck {tck}`

Specifies the maximum TCK frequency (in MHz).

-name {*name*}

Specifies the device user name.

Exceptions

None

Examples

```
add_non_actel_device -file {e:/design/bsdl/DeviceX.bsdl } -name {MyDevice3}
add_non_actel_device -ir 8 - tck 5 -name {MyDevice4}
```

add_non_actel_device_to_database

Imports settings via a BSDL file that adds non-Actel or non-Microsemi devices to the device database so that they are recognized during scan chain and auto-construction operations.

```
add_non_actel_device_to_database [-file {bsdl_filename}]
```

Arguments

-file {**bsdl_filename**}

Specifies the path to the BSDL file and the BSDL filename add to the database.

Supported Families

All non-Microsemi and non-Actel families

Exceptions

N/A

Examples

The following example uses a BSDL file to add a non-Microsemi (1502AS J44) device to the device database:

```
add_non_actel_device_to_database -file {c:/bsdl/atmel/1502AS_J44.bsd}
```

The following example uses a BSDL file to add a non-Microsemi (80200) device to the device database:

```
add_non_actel_device_to_database -file {c:/bsdl/intel/80200_v1.0.bsd}
```

construct_chain_automatically

Automatically starts chain construction for the specified programmer.

```
construct_chain_automatically[(-name {name})+]
```

Arguments

-name {*name*}

Specifies the programmer(s) name(s).

Exceptions

N/A

Example

Example for one programmer:

```
construct_chain_automatically -name {21428}
```

Example for two programmers:

```
construct_chain_automatically -name {21428} -name {00579}
```

copy_device

Copies a device in the chain to the clipboard. Chain programming mode must be set. See the paste_device command for more information.

```
copy_device (-name {name})*
```

Arguments

-name {**name**}

Specifies the device name. Repeat this argument to copy multiple devices.

Exceptions

None

Example

The example copies the device 'mydevice1' to the same location with a new name 'mydevice2'.

```
copy_device -name {MyDevice1} -name {MyDevice2}
```

cut_device

Removes one or more devices from the chain. It places the removed device in the clipboard. Chain programming mode must be set to use this command. See the paste_device command for more information.

```
cut_device (-name {name})*
```

Arguments

-name {*name*}

Specifies the device name. You can repeat this argument for multiple devices.

Exceptions

None

Example

The following example removes the devices 'mydevice1' and 'mydevice2' from the chain.

```
cut_device -name {MyDevice1} -name {MyDevice2}
```

enable_device

Enables or disables a device in the chain (if the device is disabled, it is bypassed). Chain programming mode must be set. The device must be a Microsemi device.

```
enable_device -name {name} -enable {TRUE|FALSE}
```

Arguments

`-name {name}`

Specifies your device name

`-enable {TRUE|FALSE}`

Specifies whether the device is to be enabled or disabled. If you specify multiple devices, this argument applies to all specified devices. (TRUE = enable. FALSE = disable)

Exceptions

None

Example

The following example disables the device 'mydevice1' in the chain.

```
enable_device -name {MyDevice1} -enable {FALSE}
```

paste_device

Pastes the devices that are on the clipboard in the chain, immediately above the *position_name* device, if this parameter is specified. Otherwise it places the devices at the end of the chain. The chain programming mode must be enabled.

```
paste_device [-position_name {position_name}]
```

Arguments

`-position_name {position_name}`

Optional argument that specifies the name of a device in the chain.

Exceptions

None

Examples

The following example pastes the devices on the clipboard immediately above the device 'mydevice3' in the chain.

```
paste_device -position_name {MyDevice3}
```

remove_device

Removes the device from the chain. Chain programming mode must be set.

```
remove_device (-name {name})*
```

Arguments

`-name {name}`

Specifies the device name. You can repeat this argument for multiple devices.

Exceptions

None

Example

Remove a device 'M2S050T' from the chain:

```
remove_device (-name {M2S050T})*
```

remove_non_actel_device_from_database

Removes settings for non-Microsemi or non-Actel device from the device database.

```
remove_non_actel_device_from_database [-name {device_name}]
```

Arguments

`-name {device_name}`

Specifies the non-Actel or non-Microsemi device name to be removed from the database. You can repeat this argument for multiple devices.

Supported Families

Non-Microsemi and non-Actel devices

Exceptions

None

Example

The following example removes the F1502AS_J44 device from the database:

```
remove_non_actel_device_from_database -name {F1502AS_J44}
```

The following example removes the SA2_PROCESSOR device from the database:

```
remove_non_actel_device_from_database -name {SA2_PROCESSOR}
```

select_libero_design_device

This command selects the Libero design device for the Programming Connectivity and Interface tool within Libero. This command is needed when the tool cannot automatically resolve the Libero design device when there are two or more identical devices that match the Libero design device in the configured JTAG chain.

Syntax

```
select_libero_design_device -name {device_name}
```

Arguments

`-name {device_name}`

Specifies a user-assigned unique device name in the JTAG chain.

Supported Families

SmartFusion2

IGLOO2

RTG4

Exceptions

None

Example

```
select_libero_design_device -name {M2S050TS (2)}
select_libero_design_device -name {my_design_device}
```

Note

This Tcl command is typically used in a Tcl command script file that is passed to the Libero run_tool command.

```
run_tool -name {CONFIGURE_CHAIN} -script {<flashPro_cmd>.tcl}
```

set_bsdl_file

Sets a BSDL file to a non-Microsemi device in the chain. Chain programming mode must have been set. The device must be a non-Microsemi device.

```
set_bsdl_file -name {name} -file {file}
```

Arguments

`name {name}`

Specifies the device name.

`-file {file}`

Specifies the BSDL file.

Supported Families

Any non-Microsemi device supported by FlashPro Express.

Exceptions

None

Example

The following example sets the BSDL file /design/bsdl/NewBSDL2.bsdl to the device 'MyDevice3':

```
set_bsdl_file -name {MyDevice3} -file {e:/design/bsdl/NewBSDL2.bsdl}
```

set_device_ir

Sets the IR length of a non-Microsemi device in the chain. Chain programming mode must be set. The device must be a non-Microsemi device.

```
set_device_ir -name {name} -ir {ir}
```

Arguments

`-name {name}`

Specifies the device name.

`-ir {ir}`

Specifies the IR length.

Supported Families

Any non-Microsemi device supported by FlashPro Express.

Exceptions

None

Example

The following example sets the IR length to '2' for the non-Microsemi device 'MyDevice4':

```
set_device_ir -name {MyDevice4} -ir {2}
```

set_device_name

Changes the user name of a device in the chain. Chain programming mode must be set .

```
set_device_name -name {name} -new_name {new_name}
```

Arguments

`-name {name}`

Identifies the old device name.

`-new_name {new_name}`

Specifies the new device name.

Exceptions

None

Example

The following example changes the user name of the device from 'MyDevice4' to 'MyDevice5':

```
set_device_name -name {MyDevice4} -new_name {MyDevice5}
```

set_device_order

Sets the order of the devices in the chain to the order specified. Chain programming mode must have been set. Unspecified devices will be at the end of the chain.

```
set_device_order (-name {name})*
```

Arguments

`-name {name}`

Specifies the device name. To specify a new order you must repeat this argument and specify each device name in the order desired.

Exceptions

None

Example

The following example sets the device order for 'MyDevice1', 'MyDevice2', 'MyDevice3', and 'MyDevice4'. 'MyDevice2' is unspecified so it moves to the end of the chain.

```
set_device_order -name {MyDevice3} -name {MyDevice1} -name {MyDevice4}
```

the new order is:

```
MyDevice3 MyDevice1 MyDevice4 MyDevice2
```

set_device_tck

Sets the maximum TCK frequency of a non-Microsemi device in the chain. Chain programming mode must be set. The device must be a non-Microsemi device.

```
set_device_tck -name {name} -tck {tck}
```

Arguments

`-name {name}`

Specifies the device name.

`-tck {tck}`

Specifies the maximum TCK frequency (in MHz).

Supported Families

Any non-Microsemi device supported by FlashPro Express.

Exceptions

None

Example

The following example sets the maximum TCK frequency of the non-Microsemi device 'MyDevice4':

```
set_device_tck -name {MyDevice4} -tck {2.25}
```

set_device_type

Changes the family of a Microsemi device in the chain. The device must be a Microsemi device. The device parameter below is now optional.

```
set_device_type -name {name} -type {type}
```

Arguments

`-name {name}`

Identifies the name of the device you want to change.

`-type {type}`

Specifies the device family.

Exceptions

None

Example

The following example sets the device 'MyDevice2' to the type MPF300.

```
set_device_type -name {MyDevice2} -type {MPF300}
```

set_programming_action

This Tcl command is used to select the action for a device.

```
set_programming_action [-name {device_name}] -action {procedure_action}
```

Arguments

`-name`

Specify the device name. This argument is mandatory.

`-action`

Specify the programming action. This argument is mandatory.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set_programming_action -name {MPF300(T_ES|TS_ES)} -action {DEVICE_INFO}
set_programming_action -name {M2S/M2GL090(T|TS|TV)} -action {ERASE}
```

See Also

construct_chain_automatically
scan_chain_prg

enable_device

set_debug_programmer

set_device_name

set_programming_file

run_selected_actions

set_programming_file

This Tcl command is used to set the programming file for a device. Either the file or the no_file flag must specified. A programming file must be loaded. The device must be a Microsemi device.

```
set_programming_file -name {device_name} -file {stapl_file_name_with_path}
```

Arguments

`-name`

Specify the device name. This argument is mandatory.

`-file`

Specify the *file* path. This argument is mandatory.

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
set_programming_file -name {MPF300(T_ES|TS_ES)} -file
{D:/export/CM1_PCIE_TOP_default_uic_12_200_0_12.stp}
```

See Also

construct_chain_automatically
scan_chain_prg
enable_device
set_debug_programmer
set_device_name
set_programming_action
run_selected_actions

Additional Tcl Commands

ssn_analyzer_set_pulse_width

Tcl command specific to the Simultaneous Switching Noise (SSN) Analyzer; sets the pulse width for SSN calculation.

```
ssn_analyzer_set_pulse_width -pulseWidth <value>
```

Arguments

`-pulseWidth <value>`

Specifies the threshold value for pulse width. The signal bounce pulse witdth must reach this value before the signal bounce can be recognized for SSN Analysis. Valid values are 0ns or 1ns. A value of 0ns means any signal bounce with pulse width over 0ns is recognized for SSN analysis. A value of 1ns means only signal bounces with pulse width at or above 1ns are recognized for SSN analysis.

Supported Families

SmartFusion2 and IGLOO2

Examples

```
ssn_analyzer_set_pulse_width -pulseWidth 1.0
```

This Tcl command sets the pulse width threshold value to be 1.0 ns.

See Also

Simultaneous Switching Noise

nvm_update_serialization_client

This command updates an existing serialization client in the SmartFusion2 and IGLOO2 eNVM.

```
nvm_update_serialization_client -params {parm:value} [-params {parm:value}]
```

This command is usually put in a configuration “*.cfg” file and passed as an argument to the script parameter of the run_tool command.

```
run_tool -name {UPDATE_ENVM} -script “update.cfg”
```

Parameter and Parameter Values

The following table lists the parameter name and values for this command.

Parameter	Type	Value	Description
client_name	String	valid string	Specifies the name of the eNVM serialization client to update
number_of_words	Integer	Decimal**	Specifies the number of words

Parameter	Type	Value	Description
use_for_simulation	Boolean	0 | 1	Specifies whether or not the serialization client is used for simulation
base_address	Integer	HEX numeral**	Specifies the client base address
maximum_devices_to_program	Integer	See eNVM User Guide	Specify maximum devices to program
reprogram	Boolean	0 | 1	Specifies whether or not to re-program
content_from_file	Boolean	0 | 1	Specify the content from file
number_of_pages	Integer	Decimal**	Specify the number of pages
content_file	String	Valid string	Specify absolute or relative path to content file
content_file_format	String	{Decimal | Hexadecimal} }	Specifies the content file format
start_value	Integer	User input	Specifies the start value
step_value	Integer	User input	Specifies the step value
maximum_value	Integer	See eNVM User Guide**	Specifies the maximum value
use_as_rom	Boolean	0 | 1	Specifies whether or not the serialization client is to be used as ROM

** eNVM address range and available number of words are device dependent. See the eNVM Configuration User Guide for details.

** number_of_words available to users = Number of user pages * BYTES_PER_PAGE *8 / word_size

Supported Families

SmartFusion2, IGLOO2

Example

```
nvm_update_serialization_client \
-client_name {d1} \
-maximum_devices_to_pgoram {6}
-maximum_value {12}
```

See Also

"run_tool" on page 65

nvm_update_storage_client

This command updates an existing data storage client in the SmartFusion2 and IGLOO2 eNVM.

```
nvm_update_storage_client -params {parm:value} [-params {parm:value}]
```

This command is usually put in a configuration "*.cfg" file and passed as an argument to the script parameter of the run_tool command.

```
run_tool -name {UPDATE_ENVM} -script "update.cfg"
```

Parameter and Parameter Values

The following table lists the parameter name and values for this command.

Parameter	Type	Value	Description
client_name	String	valid string	Specifies the name of the eNVM storage client to update
word_size	Integer	8 | 16 | 32	Specifies the number of bits for each word
number_of_words	Integer	User input value**	Specifies the number of words
use_for_simulation	Boolean	0 | 1	Specifies whether or not the data storage client is used for simulation
base_address	Integer	HEX numeral**	Specifies the client start address
retrieve_address	Boolean	0 | 1	Specifies whether or not the address is retrieved from a file
reprogram	Boolean	0 | 1	Specifies whether or not the data storage client is re-programmed
memory_file_format	String	{INTELHEX | MOTOROLAS, SIMPLEHEX | {BINARY}	Specifies the memory file format
memory_file	String	valid string	Specifies the absolute or relative path of the memory file
content_type	String or Integer	{MEMORY_FILE | STATIC_FILL | NO_CONTENT} or {0 | 1 |2}	Specifies the content type

Parameter	Type	Value	Description
lock_address	Boolean	0 | 1	If set to 1, the start address of the client(s) is locked and cannot be changed during optimization.
static_fill_pattern	String	The string consists of 0 and 1	Specifies the static fill pattern – 0 or 1.
use_as_rom	Boolean	0 | 1	Specifies whether or not the data storage client is to be used as ROM

** eNVM address range and available number of words are device dependent. See the eNVM Configuration User Guide for details.

** number_of_words available to users = Number of user pages * BYTES_PER_PAGE *8 / word_size

Supported Families

SmartFusion2, IGLOO2

Example

```
nvm_update_storage_client \
-client_name {c1} \
-word_size 32
-number_of_words 512
```

See Also

"run_tool" on page 65

select_programmer

This Tcl command enables the specified programmer and disables all other connected programmers. This command is useful when multiple programmers are connected.

```
select_programmer -programmer_id {programmer_id} -host_name {host_name} -host_port
{host_port}
```

Arguments

-programmer_id <*programmer_id*>

The programmer to be enabled. See the Select Programmer online help topic

-host_name <*host_name*>

The host name or IP address. This argument is required for a remote programmer and optional for a local programmer. For local programmer, if specified it must be “localhost”.

-host_port <*host_port*>

This argument is required for a remote programmer and optional for a local programmer. If omitted, the default port is used (currently, the default is 80).

For a local host, both “localhost” and its port should be specified or omitted.

Note: The def variable “LOCAL_PROGRAM_DEBUG_SERVER_PORT” is used to set a different default local host port.

Supported Families

SmartFusion2
IGLOO2
RTG4

Examples

```
select_programmer -programmer_id {00557}
select_programmer -programmer_id {00557} \
-host_name {localhost} \
-host_port {80}
```

See Also

Select Programmer

set_programming_interface

This Tcl command sets the programming interface.

```
set_programming_interface -interface {JTAG | SPI_SLAVE}
```

Arguments

`set_programming_interface -interface {JTAG | SPI_SLAVE}`
Specify the programming interface as JTAG or SPI_SLAVE. The default is JTAG.

Supported Families

SmartFusion2
IGLOO2
RTG4

See Also

Programming Connectivity and Interface

See the online help for more information.

ssn_analyzer_noise_report

Tcl command specific to the Simultaneous Switching Noise (SSN) Analyzer; instructs the SSN Analyzer to generate a noise report of all the used I/Os in the design.

```
ssn_analyzer_noise_report -style {file format} -filename {full_path_to_filename}
```

Arguments

`-style file_format`
Specifies the file format for the report. Valid values are “Text”, “CSV”, and “XML”
`-filename full_path_to_filename`
Specifies the *full_path_to_filename* for the report.

Supported Families

SmartFusion2, IGLOO2

Example

The following example generates a noise report in Text format and saves it in D:\2Work\SF2_SSN\MixedIOStd\myreport:

```
ssn_analyzer_noise_report -style {Text} -filename \
{D:\2Work\SF2_SSN\MixedIOStd\myreport}
```

ssn_analyzer_rerun_analysis

Tcl command specific to the Simultaneous Switching Noise (SSN) Analyzer; instructs the SSN Analyzer to run the SSN analysis and compute the noise margin numbers.

```
ssn_analyzer_rerun_analysis
```

Arguments

This command takes no arguments.

Supported Families

SmartFusion2, IGLOO2

Example

The following example reruns the SSN Analyzer and computes the Nosie Margin number:

```
ssn_analyzer_rerun_analysis
```

ssn_analyzer_set_dontcare

Tcl command specific to the Simultaneous Switching Noise (SSN) Analyzer; sets specific I/Os to the dont_care state or resets dont_care I/Os to non-dont_care. A dont_care I/O is considered as an Aggressor only and not as a Victim.

```
ssn_analyzer_set_dontcare -io "ioName" -iobank "ioBankName" -value "integer value"
```

Arguments

`-io "IoName"`

Specifies the I/O to be dont_care (the don't_care I/O will not be considered as a Victim, whereas it will be considered as an Aggressor for the SSN analysis) or resets dont_care I/Os to non-dont _care.

`-iobank "ioBankName"`

Specifies the I/O bank name the specific I/O belongs to.

`-value "0|1"`

Specifies an integer of "0" or "1" where

"1" is used to set an I/O to be dont_care

"0" is used to reset an I/O to be non-dont_care

Supported Families

SmartFusion2, IGLOO2

Examples

The following example sets the I/O named "DATA2" in I/O bank "Bank2" to dont_care:

```
ssn_analyzer_set_dontcare -io "DATA2" -ioBank "Bank2" -value "1"
```

The following example sets the dont_care I/O named "DATA1" in I/O bank "Bank3" to non-dont_care:

```
ssn_analyzer_set_dontcare -io "DATA1" -ioBank "Bank3" -value "0"
```

ssn_analyzer_set_pulse_width

Tcl command specific to the Simultaneous Switching Noise (SSN) Analyzer; sets the pulse width for SSN calculation.

```
ssn_analyzer_set_pulse_width -pulseWidth <value>
```

Arguments

`-pulseWidth <value>`

Specifies the threshold value for pulse width. The signal bounce pulse witdth must reach this value before the signal bounce can be recognized for SSN Analysis. Valid values are 0ns or 1ns. A value of 0ns means any signal bounce with pulse width over 0ns is recognized for SSN analysis. A value of 1ns means only signal bounces with pulse width at or above 1ns are recognized for SSN analysis.

Supported Families

SmartFusion2 and IGLOO2

Examples

```
ssn_analyzer_set_pulse_width -pulseWidth 1.0
```

This Tcl command sets the pulse width threshold value to be 1.0 ns.

See Also

Simultaneous Switching Noise

ssn_analyzer_set_static

Tcl command specific to the Simultaneous Switching Noise (SSN) Analyzer; sets specific I/Os to the static state or resets static I/Os to be non-static.

When the –value is “1”, it sets a specific I/O as static to the SSN Analyzer. A static I/O is considered neither as a victim nor as an aggressor. When the –value is “0”, this commands resets a static I/O to be non-static.

```
ssn_analyzer_set_static –io “ioName” –iobank “ioBankName” –value “integer value”
```

Arguments

`-io “IoName”`

Specifies the I/O name to be marked as static (neither to be considered as a Victim nor as an Aggressor for the SSN analysis) or a static I/O to be non-static.

`-iobank “ioBankName”`

Specifies the I/O bank name the specific I/O belongs to.

`-value “0|1”`

Specifies an integer value of either “0” or “1” where

“1” is used to set a particular I/O to be static

“0” is used to reset a static I/O to be non-static

Supported Families

SmartFusion2, IGLOO2

Examples

The following example sets the I/O named “DATA1” in I/O bank “Bank3” to static:

```
ssn_analyzer_set_static -io “DATA1” -ioBank “Bank3” -value “1”
```

The following example sets the static I/O named “DATA2” in I/O bank “Bank1” to be non-static:

```
ssn_analyzer_set_static -io “DATA2” -ioBank “Bank1” -value “0”
```

ssn_analyzer_summary_report

Tcl command specifc to the Simultaneous Switching Noise (SSN) Analyzer; instructs the SSN Analyzer to generate a SSN Analyzer summary report of all the used I/Os in the design.

```
ssn_analyzer_summary_report -style “file format” -file “filename”
```

Arguments

`-style file format`

Specifies the file format for the report. Valid values are “text”, “csv”, and “xml”.

`-file filename`

Specifies the filename for the report.

Supported Families

SmartFusion2, IGLOO2

Example

The following example generates a summary report in XML format and saves it in the summary_report.xml file in the current directory:

```
ssn_analyzer_summary_report -style “xml” -filename “./summary_report”
```

update_storage_client

This command updates an existing uPROM storage client for the RTG4 uPROM.

```
nvm_update_storage_client -params {parm:value} [-params {parm:value}]
```

This command is usually put in a configuration “*.cfg” file and passed as an argument to the script parameter of the run_tool command.

```
run_tool -name {UPDATE_ENVM} -script “update.cfg”
```

Parameter and Parameter Values

The following table lists the parameter name and values for this command.

Parameter	Type	Value	Description
client_name	String	valid string	Specifies the name of the uPROM data storage client to update
number_of_words	Integer	Decimal 1-10,400	Specifies the number of words (36 bits per word)
use_for_simulation	Boolean	0 | 1	Specifies whether or not the data storage client is used for simulation

Parameter	Type	Value	Description
base_address	Integer	HEX (0-0x289F)**	Specifies the client start address
retrieve_address	Boolean	0 | 1	Specifies whether or not the address is retrieved from a file
memory_file_format	String	{Microsemi Binary}	Specifies the memory file format
memory_file	String	valid string	Specifies the absolute or relative path of the memory file
content_type	String	{MEMORY_FILE | STATIC_FILL | NO_CONTENT}	Specifies the content type

** number_of_words available to users = Number of user pages * BYTES_PER_PAGE *8 / word_size

Supported Families

RTG4

Example

```
update_storage_client \
    -client_name {inc_dat} \
    -number_of_words {10400} \
    -use_for_simulation {0} \
    -content_type {MEMORY_FILE} \
    -memory_file_format {Microsemi-Binary} \
    -memory_file "E:/no-IDE/rtg4_uprom_example/uprom1.mem" \
    -base_address {0}
```

See Also

"run_tool" on page 65