SmartFusion2, IGLOO2, and RTG4 Tcl for SoC

Tcl Documentation

Microsemi

Table of Contents

Introduction

Microsemi's Libero System on Chip (SoC) is a comprehensive software toolset for designing with Microsemi FPGA and SoC FPGAs. Libero SoC enables you to manage the entire design flow from design entry, synthesis and simulation, through place-and-route, timing and power analysis, with enhanced integration of the embedded design flow.

Libero SoC also has a rich set of Tcl commands that support the design flow process in batch mode. This document introduces you to frequently-used Tcl commands available to hardware designers targeting Microsemi's SmartFusion2, IGLOO2, and RTG4 families.

For a general introduction to Tcl commands, see the Tcl Command Reference in the Libero SoC online help.

Lexical Conventions and Syntax

This document uses a subset of the BNF syntax to describe the usage of the Libero SoC Tcl commands.

The Tcl command import_source is shown below to illustrate the syntax.

import_source

–format *imp_fmt* \

[-edif_flavor *flavor_kinds*] {*filename*$_n$} \

imp_fmt ::= adl | edif | verilog | vhdl | pin | crt | gcf

flavor_kinds ::= generic | viewlogic | mgc | orcad | workview

filename$_n$ ::= a filename with an absolute or relative path

Where:

- **import_source** is the Tcl command name
- -format is the parameter name
- *imp_fmt* is one of the valid parameter values indicated and preceded by the ::= notation. Alternative values are separated by the vertical bar |. In this example adl, edif, verilog, vhdl, pin, crt and gcf are valid parameter values for the parameter imp_fmt.
- The backslash symbol \ breaks long Tcl commands and continues on the next line.
- Square brackets [ ] denote an expression that can appear in a statement 0 (optional) or 1 time.
- Curly braces { } denote an expression that can appear in a statement 0 (optional) or multiple times. A subscript is used with the expression to further denote a potential list.

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

1 – Frequently-Used Libero SoC Tcl Commands for SmartFusion2, IGLOO2, and RTG4

The following is a list of frequently-used Libero SoC Tcl commands and their syntax.

To understand the context of these commands in the Libero SoC design flow, refer to "Sample Script #1" on page 12 and "Sample Script #2" on page 14.

For a general description of Libero SoC Tcl commands, refer to the Libero SoC online help.

import_files

Syntax

import_files \

-edif {<absolute or relative path to file>}	# For EDIF files
-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
-sdc {<absolute or relative path to file>}	# For SDC files
-hdl_source {<absolute or relative path to file>}	# For Verilog or VHDL files

Example

```
import_files -edif {D:/Designs/gugu/synthesis/sd1.edn}
import_files -io_pdc {./my_attributes.pdc}
import_files -fp_pdc {./my_timing.pdc}
```

Description

This command enables you to import design source files and constraint files.

For importing constraint files, import_files has retired the -pdc parameter for SmartFusion2, IGLOO2, and RTG4. 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 will cause an error. The path to the file can be absolute or relative but must be enclosed in curly braces { }.

configure_tool

Syntax

configure_tool \

-name {<*tool_name*>} \	# Each tool_name will have its own set of parameters
-params {<*parameter*>:<*value*>}\	# List of all parameters and the values they can be set to

tool_name ::= COMPILE | SIM_PRESYNTH | SYNTHESIZE | SIM_POSTSYNTH | EXPORTNETLIST | EXPORTSDF | PLACEROUTE | SIM_POSTLAYOUT | VERIFYTIMING | VERIFYPOWER | EXPORTPIN | GENERATEPROGRAMMINGFILE | EXPORTPROGRAMMINGFILE | GENERATEPROGRAMMINGDATA | PROGRAMDEVICE | SPM

Example

```
configure_tool -name {COMPILE} -params {PDC_IMPORT_HARDERROR:true}
```

```
configure_tool -name {PLACEROUTE} -params {PDPR:false} -params {TDPR:true}
```

Description

configure_tool is a general purpose tool for setting the parameters for any tool called by Libero for the SmartFusion2, IGLOO2, and RTG4 families. The command requires the name of the tool and one or more parameters in the format of <tool_parameter>:<value>. These parameters are separated and passed to the tool to set up its run.

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. The following three examples all do the same thing.

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}
```

organize_tool_files

Syntax

organize_tool_files \

-tool {<*tool_name*>} \

-file {<absolute or relative path to constraint file>} \ # There may be multiple -file parameter entries

-module {sd1::work} \

-input_type {constraint} # Specifies type of input file

Example

```
organize_tool_files \
   -tool {COMPILE} \
   -file {D:/Designs/tests/constraint/io/io_0.pdc} \
   -file {D:/Designs/tests/constraint/fp/io_1.pdc} \
   -file {D:/Designs/tests/constraint/arbiter.sdc} \
   -module {sd1::work} \
   -input_type {constraint}
```

Description

organize_tool_files is used to specify specific constraint files to be passed to and used by a Libero tool.

run_tool

Syntax

run_tool \

-name {<*tool_name* >} \ # Name of a Libero tool

-script {<absolute or relative path to script file>} \ # Location of a script file

tool_name ::= COMPILE | SIM_PRESYNTH | SYNTHESIZE | SIM_POSTSYNTH | EXPORTNETLIST | EXPORTSDF | PLACEROUTE | SIM_POSTLAYOUT | VERIFYTIMING | VERIFYPOWER | EXPORTPIN | GENERATEPROGRAMMINGFILE | EXPORTPROGRAMMINGFILE | GENERATEPROGRAMMINGDATA | PROGRAMDEVICE | SPM

Example

```
run_tool -name {COMPILE}
run_tool -name {SYNTHESIZE} -script {./control_synopsys.tcl}
```

Description

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

export_script

Syntax

export_script \

-file {<*absolute or relative path to file*>} \

-relative _path <value> # 1 for using relative path, 0 for absolute path in the
exported script

value ::= 1 | 0

Example

```
export_script -file {./exported.tcl} -relative_path 1
```

Description

export_script is a new command that explicitly exports the Tcl command equivalents of the current Libero session. You must supply a file name with the -file parameter. You may supply the optional -relative_path parameter to specify whether an absolute or relative path is used in the exported script file.

2 – Tcl Commands and Parameters

Symbol Definitions

filename ::= a filename with an absolute or relative path

dirname ::= a directory name with an absolute or relative path

parameter ::= a string representing a keyword passed as an argument by command parameter

parameter_value ::= a string representing a value for a *parameter.*

pv_pair ::= *parameter*:*parameter_value*

module_def ::=

torf ::= ‘true’|’1’|’false’|’0’

Table 2-1 • **configure_tool Parameters and Values**

Tcl Command	Parameter	Parameter Values	Notes
configure_tool	-name *tool_name*	*tool_name* ::= ‘COMPILE’ ‘EXPORTNETLIST’ ‘EXPORTPIN’ ‘EXPORTSDF’ ‘GENERATEPROGRAMMINGDATA’ ‘GENERATEPROGRAMMINGFILE’ ‘PLACEROUTE’ ‘PROGRAMDEVICE’ ‘SIM_POSTLAYOUT’ ‘SIM_POSTSYNTH’ ‘SIM_PRESYNTH’ ‘SPM’ ‘SYNTHESIZE’ ‘VERIFYTIMING’ ‘VERIFYPOWER’	Each *tool_name* may have a set of parameters specified with a -params parameter. See Table 2-2 on page 8 for a list of tool_name and parameter values.
	-params *pv_pair*	Tool_specific	*pv pairs* are specific to the *tool_name* specified. See Table 2-2 on page 8.
	-file *filename*	Tool_specific	-file use is specific to the tool_name specified. See Table 2-2 on page 8
	-module *module_def*	Tool_specific	-module use is specific to the tool_name specified. See Table 2-2 on page 8
	-input_type *ip_type_def*	Tool_specific	-input_type use is specific to the tool_name specified. See Table 2-2 on page 8

***Table 2-2* • configure_tool Parameters and Values for tool_name**

tool_name	**Command Parameter**	**Tool Parameter (*parameter)***	**Tool Parameter Value (*parameter_value*)**
COMPILE	-file	Not a pv_pair type	*filename*
	-module	Not a pv_pair type	*module_def*
	-input_type	Not a pv_pair type	constraint
	-params	DISPLAY_FANOUT_LIMIT	<integer>
	-params	MERGE_SDC	*torf*
	-params	PDC_IMPORT_HARDERROR	*torf*
EXPORTNETLIST	-params	EXPORT_HDL_TYPE	VERILOG VHDL
EXPORTPIN	-params	PINRPT_BY_NAME	*torf*
	-params	PINRPT_BY_NUMBER	*torf*
EXPORTPROGRAMMINGFILE	-params	DAT	*torf*
	-params	SPI	*torf*
	-params	STP	*torf*
	-params	SVF	*torf*
	-params	SVF_max_filesize	
	-params	SVF_max_vectors	
	-params	file_name	sd1
	-params	limit_SVF_file_size	*torf*
	-params	limit_SVF_max_filesize_or_vector	
EXPORTSDF	-params	DELAY_TYPE	*torf*
	-params	EXPORT_HDL_TYPE	VERILOG VHDL

***Table 2-2* • configure_tool Parameters and Values for tool_name (continued)**

tool_name	Command Parameter	Tool Parameter (*parameter*)	Tool Parameter Value (*parameter_value*)
PLACEROUTE	-params	INCRPLACEANDROUTE	*torf*
	-params	PDPR	*torf*
	-params	TDPR	*torf*
	-params	EFFORT_LEVEL	*torf*
	-params	MULTI_PASS_LAYOUT	*torf*. Default is false
	-params	NUM_MULTI_PASSES	<*integer 1 through 25*> Default is 5
	-params	START_SEED_INDEX	<*integer 1 through 101*> Default behavior is to continue from the last seed index which was used.
	-params	MULTI_PASS_CRITERIA	{SLOWEST_CLOCK | SPECIFIC_CLOCK | VIOLATIONS | TOTAL_POWER} Refer to Libero online help for details.
	-params	SPECIFIC_CLOCK	{*Name of Clock*}
	-params	DELAY_ANALYSIS	{MAX | MIN} Refer to Libero online help for details.
	-params	STOP_ON_FIRST_ PASS	*torf*. Refer to Libero online help for details.
	-params	SLACK_CRITERIA	{WORST_SLACK | TOTAL_NEGATIVE_SLACK } Refer to Libero online help for details.
PROGRAM_OPTIONS	-params	program_envm	*torf*
	-params	program_fabric	*torf*
	-params	program_mode	selected_features
	-params	program_security	*torf*
PROGRAMDEVICE	-params	program_action	DEVICE_INFO
	-params	program_procedures	VERIFY_IDCODE DO_DEVICE_INFO DO_EXIT
SPM	-params	spm_file_name	<*filename*>

***Table 2-3* • run_tool Parameter and Values**

Tcl command	Parameter	Parameter values	Notes
run _tool	-name *tool_name*	*tool_name* ::= ‘COMPILE’ ‘EXPORTNETLIST’ ‘EXPORTPIN’ ‘EXPORTPROGRAMMINGFILE’ ‘EXPORTSDF’ ‘GENERATEPROGRAMMINGDATA’ ‘GENERATEPROGRAMMINGFILE’ ‘PLACEROUTE’ ‘PROGRAMDEVICE’ ‘SIM_POSTLAYOUT’ ‘SIM_POSTSYNTH’ ‘SIM_PRESYNTH’ ‘SPM’ ‘SYNTHESIZE’ ‘VERIFYTIMING’ ‘VERIFYPOWER’ 'SMARTDEBUG'	N/A
	-script *filename*	*filename* ::= a filename with an absolute or relative path	A *tool_name* may have the ability to execute a set of commands saved in a script file. If included in the command the file will be passed to the tool.
	-defvar		List of user def variables
	-defvars		List of Libero def variables

***Table 2-4* • organize_tool_files Parameter and Values**

Tcl Command	Parameter	Parameter Values	Notes
organize_tool_files	-tool *tool_name*	COMPILE	
	-file *filename*	*filename* ::= a filename with an absolute or relative path	Multiple -file parameters may be used in one command
	module *module_def*	*module_def*	${*design*}::work
	-input_type *input_type_def*	*input_type_def* ::=’constraint’ | ‘source’ | ‘simulation’ | ‘stimulus’ | ‘unknown’	

***Table 2-5* • Miscellaneous Tcl Commands**

Tcl Command	Parameter	Parameter Values	Notes
defvar_set	-name *def_var*	*def_var* ::= variable name	
	-value *def_var_value*	*def_var_value ::*= value for def_var	
export_script	-file	*filename*	
	-relative_path	*torf*	*torf* or 1 and 0 only
import_files	-edif	*filename*	For *.edn files
	-io_pdc	*filename*	For *.pdc containing I/O attribute and pin info
	-fp_pdc	*filename*	*.pdc file that contains timing and placement info
	-sdc	*filename*	For *.sdc files
	-hdl_source	*filename*	For Verilog or VHDL source files
set_actel_lib_options	-use_default_sim_path	1 | 0 or *torf*	
	-sim_path		

3 – Sample Tcl Scripts

The two sample scripts below contain some of the Tcl commands listed in the tables in "Tcl Commands and Parameters" on page 7. Relevant context is embedded in the script comments.

Sample Script #1

```
# ------------------------------------------------------------
# Save a design as a verilog netlist:
#
# 1- Delete the project and recreate.
# 2- Import and Organize EDIF file
# 3- Run Compile
# 4- Export netlist
# 5- Run Place and Route,then export back annotation files.
# ------------------------------------------------------------

# ------------------------------------------------------------
# 1. Be clean.Start from scratch.Delete the project
#    and recreate.
# ------------------------------------------------------------
file delete -force {D:/Designs/hellos}

new_project \
   -location    {D:/Designs/hellos} \
   -name        {hellos} \
   -hdl         {VERILOG} \
   -family      {SmartFusion2} \
   -die         {M2S050T} \
   -package     {896 FBGA} \
   -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 {RESTRICTPROBEPINS:1} \
   -adv_options {TEMPR:COM} \
   -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:COM}

# ------------------------------------------------------------
# 2. Import and Organize EDIF file
# ------------------------------------------------------------
import_files  -edif {D:/Designs/gugu/synthesis/sd1.edn}
use_file -file {D:/Designs/hellos/synthesis/sd1.edn} \
   -designer_view {Impl1} \
   -module        {sd1::work}

# ------------------------------------------------------------
# 3. Run Compile (Use files organized for current root)
# ------------------------------------------------------------
run_tool -name {COMPILE}

# ------------------------------------------------------------
# 4. Export netlist after Compile.
# ------------------------------------------------------------
```

```
configure_tool \
   -name {EXPORTNETLIST} \
   -params {EXPORT_HDL_TYPE:VERILOG}

run_tool -name {EXPORTNETLIST}

# ------------------------------------------------------------
# 5. Run Place and Route, then export back annotation files.
# ------------------------------------------------------------
run_tool -name {PLACEROUTE}

configure_tool \
   -name {EXPORTSDF} \
   -params {DELAY_TYPE:false} \
   -params {EXPORT_HDL_TYPE:VERILOG}

run_tool -name {EXPORTSDF}
```

Sample Script #2

```
# ------------------------------------------------------------
# 1- open an existing Libero project
# 2- import an edif netlist (they synthesize outside Libero
#    using their own synplify_pro)
# 3- import SDC and PDC (or mark them as used)
# 4- compile (for now default options will do)
# 5- timing driven place and route
# 6- export reports: timing, violation, max delay paths, pin report.
# 7 - export stapl file.
# 8- save and close the design
# ------------------------------------------------------------

# ------------------------------------------------------------
# 1. open the project
# ------------------------------------------------------------
open_project \
   -file {../msstest/msstest.prjx} \
   -do_backup_on_convert 1 \
   -backup_file {../msstest.zip}
# ------------------------------------------------------------
# 2. Import EDIF
# ------------------------------------------------------------
import_files -edif {D:/Designs/gugu/synthesis/sd1.edn}
# ------------------------------------------------------------
# 3. Import constraints
#
# -io_pdc is for io pad constraints and attributes
# -fp_dpc is for floorplanning constraints
# these constraints must be in separate files and imported
# with their individual commands - new to SF2
# ------------------------------------------------------------
import_files -io_pdc {D:/Designs/io_0.pdc}
import_files -fp_pdc {D:/Designs/io_1.pdc}
import_files -sdc {D:/Designs/arbiter.sdc}

# ------------------------------------------------------------
# 4. compiling
# ------------------------------------------------------------
organize_tool_files \
   -tool {COMPILE} \
   -file {D:/Designs/tests/constraint/io/io_0.pdc} \
   -file {D:/Designs/tests/constraint/fp/io_1.pdc} \
   -file {D:/Designs/tests/constraint/arbiter.sdc} \
   -module {sd1::work} \
   -input_type {constraint}
run_tool -name {COMPILE}

# ------------------------------------------------------------
# 5. place and route
# ------------------------------------------------------------
configure_tool \
   -name   {PLACEROUTE} \
   -params {EFFORT_LEVEL:false} \
   -params {INCRPLACEANDROUTE:false} \
   -params {PDPR:false} -params {TDPR:true}
run_tool -name {PLACEROUTE}

# ------------------------------------------------------------
# 6. exporting reports
# this automatically makes a number of reports
# under <project>/designer/<design>
# some are xml, some are text
```

```
# ------------------------------------------------------------
run_tool -name {VERIFYTIMING}
run_tool -name {EXPORTPIN}

# ------------------------------------------------------------
# 7. export programming file (*.pdb)
# SPL file under <project>/designer/<design>/export
# ------------------------------------------------------------
run_tool -name {EXPORTPROGRAMMINGFILE}

# ------------------------------------------------------------
# 8. Save and close project
#    Shown as two commands but could be one with lower example
# ------------------------------------------------------------
save_project
close_project -save 1
```

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and security, aerospace, 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; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

5-02-00616-0/03.15