Identify®
Reference

June 2013

http://solvnet.synopsys.com

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only.

Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

"This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

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

Disclaimer

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

Registered Trademarks (®)

Synopsys, AEON, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, CoMET, CODE V, Design Compiler, DesignWare, EMBED-IT!, Formality, Galaxy Custom Designer, Global Synthesis, HAPS, HapsTrak, HDL Analyst, HSIM, HSPICE, Identify, Leda, LightTools, MAST, METeor, ModelTools, NanoSim, NOVeA, OpenVera, ORA, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Sonic Focus, STAR Memory System, Syndicated, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, ARC, ASAP, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomExplorer, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Eclypse, Encore, EPIC, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIMplus, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Intelli, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Macro-PLUS, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, ORAengineering, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, RippledMixer, Saturn, Scirocco, Scirocco-i, SiWare, Star-RCXT, Star-SimXT, StarRC, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCSi, VHDL Compiler, VMC, and Worksheet Buffer are trademarks of Synopsys, Inc.

Service Marks (sm)

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

SystemC is a trademark of the Open SystemC Initiative and is used under license.

ARM and AMBA are registered trademarks of ARM Limited.

Saber is a registered trademark of SabreMark Limited Partnership and is used under license.

All other product or company names may be trademarks of their respective owners.

Printed in the U.S.A
June 2013

Contents

Chapter 4: Alphabetical Command Reference

SYNOPSYS®
Accelerating Innovation

CHAPTER 1

Command Reference Introduction

The Identify® tool set consists of the Identify instrumentor and the Identify debugger. These two tools allow you to debug your HDL design:

- In the target system
- At the target speed
- At the VHDL/Verilog RTL Source level

The Identify tool set increases your debugging capabilities of high-end FPGA designs, FPGA-based prototypes, and system-on-a-chip designs. For the first time you will be able to debug live hardware with the internal design visibility you need while using intuitive debugging techniques.

To efficiently use the system and its underlying tools, this manual provides you with a comprehensive listing of all the commands the Identify tool set accepts. You can access this command information by concept listing or alphabetically.

In addition to easy look-up access to the commands, this manual also contains conventions that help display the command information easily and quickly. These manual conventions organize each command so that the information can be attained easily.

The remainder of this chapter describes:

Manual Conventions

There are several conventions this manual uses in order to organize command information effectively. These conventions are:

This convention...	Organizes this information....
Text convention	Text containing paths, directories, command names, and menu selections. All system specific command and path information has its own text convention to make is decipherable throughout the manual.
Syntax convention	Symbols used to separate command examples and other important system text. Syntax conventions include any type of brackets and parentheses that separate commands and functions from the rest of the text.

Text Conventions

There are several text conventions this manual uses to organize command, path and directory information. These conventions or text styles are:

This convention...	Organizes this information...
Bold	command titles
`Monospacing type`	command, path name, and directory examples
Sans-serif type	commands, literals, and keywords
Italics	variable arguments

Syntax Conventions

There are several conventions this manual uses to convey command syntax. These conventions are:

This convention...	Organizes this information...
bold	Commands and literal arguments entered as shown.
italics	User-defined arguments or example command information.
[ ]	Optional information or arguments for command use. Do not use these brackets with the command within the command line.
...	Items that can be repeated any number of times.
|	Choices you can make between two items or commands. The items are located on either side of this of this symbol.
#	Comments concerning the code or information within the command line.
{}	Escape characters for search strings; also entered in bold font as a literal in some commands

Symbol Conventions

This manual contains symbol conventions detailing the tools that use these commands. These symbols are located adjacent to the command name in any of the command listing chapters. These symbols are:

This convention...	Organizes this information...
	Any command that is used in the Identify instrumentor only. This symbol is located underneath any Identify instrumentor command in the command listing chapters.
	Any command that is used in the Identify debugger only. This symbol is located underneath any Identify debugger command in the command listing chapters.
	Any command that is used in both the Identify instrumentor and Identify debugger tools. This symbol is located underneath the commands that have applications in both tools in the command listing chapters.

Tool Conventions

There are tool concepts you must familiarize yourself with when using the Identify tool set. These concepts help you to decipher structural and HDL-related information.

File System Conventions

The term file system refers to any command that uses file, directory, or path name information in its argument. A file system command must contain specific conventions.

Path Separator "/"

All file system commands that contain a directory name use only forward slashes, regardless of the underlying operating system:

```
/usr/data.dat
```

```
c:/Synopsys/data.dat
```

Wildcards

A wildcard is a command element you can use to search for specific file information. You can use these wildcards in combination with the file system commands. Conventions for wildcards are as follows:

Syntax	Description
*	Matches any sequence of characters
?	Matches any single character

Square brackets are used in pattern matching as follows:

Syntax	Description
[abcd]	Matches any character in the specified set.
[a-d]	Matches any character in a specified range.

To use square brackets in wildcard specifications, you must delimit the entire name with curly braces {}. For example

```
{[a-d]1}
```

matches any character in the specified range (a-d) preceding the character 1.

Design Hierarchy Conventions

Design hierarchy refers to the structure of your design. Design hierarchy conventions define a way to refer to objects within the design hierarchy.

The Identify tool set supports VHDL and Verilog. These languages vary in their hierarchy conventions. The VHDL and Verilog languages contain design units and hierarchies of these design units. In VHDL, these design units are entity/architecture pairs, in Verilog they are modules. VHDL and Verilog design units are organized hierarchically. Each of the following HDL design units creates a new level in the hierarchy:

VHDL

- The top-level entity
- Architectures
- Component instantiation statements
- Process statements
- Control flow statements: if-then-else, and case
- Subprogram statements
- Block statements

Verilog

- The top-level module
- Module instantiation statements
- Always statements
- Control flow statements: if-then-else, and case
- Functions and tasks

Design Hierarchy References

A reference to an element in the design hierarchy consists of a path made up of references to design units (similar to a file reference described earlier). Regardless of the underlying HDL (VHDL or Verilog) the path separator character is always "/":

```
/inst/reset_n
```

Absolute path names begin with a path separator character. The top-level design unit is represented by the initial "/". Thus, a port on the top-level design unit would be represented:

```
/port_name
```

The architecture of the top-level VHDL design unit is represented:

```
/arch
```

Relative path names do not start with the path separator, and are relative to the current location in the design hierarchy. Initially, the current location is the top-level design unit, but commands exist that allow you to change the location.

Note: Design unit and hierarchy information can be case sensitive depending on the HDL language. VHDL names are not case sensitive. In contrast, all Verilog names are case sensitive.

Wildcards

A wildcard is a command element you can use to search for specific design hierarchy information. You can use these wildcards in combination with the design hierarchy commands. Conventions for wildcards are as follows:

Syntax	Description
*	Matches any sequence of characters
?	Matches any single character

Square brackets are used in hierarchy pattern matching as follows:

Syntax	Description
[abcd]	Matches any character in the specified set.
[a-d]	Matches any character in a specified range.

To use square brackets in pattern matching, you must delimit the entire name with curly braces {}. For example

```
{[a-d]1}
```

matches any character in the specified range (a-d) preceding the character 1.

SYNOPSYS®
Accelerating Innovation

CHAPTER 2

Startup Modes

The Identify instrumentor and the Identify debugger can be started in any of three execution modes as outlined below:

- `identify_instrumentor`

  Opens the Identify instrumentor in the graphical interface

- `identify_instrumentor -f` *`fileName`*`.tcl`

  Runs a Tcl startup file and then opens the Identify instrumentor in the graphical user interface.

- `identify_instrumentor_shell [-version]`

  Opens the Identify instrumentor in the shell and/or script mode. If the optional -version argument is included, reports the software version without opening the Identify instrumentor.

- `identify_debugger`

  Opens the Identify debugger in the graphical interface.

- `identify_debugger -f` *`fileName`*`.tcl`

  Runs a Tcl startup file and then opens the Identify debugger in the graphical user interface.

- `identify_debugger_shell [-version]`

  Opens the Identify debugger in the shell and/or script mode. If the optional -version argument is included, reports the software version without opening the Identify debugger.

Note: Depending on the command shell, you may be required to provide the full path name to the program executable as well as the full path name to the files specified in any of the filename arguments.

Synthesis Tool Pointers

When the Identify instrumentor or the Identify debugger is started from the command line, two additional arguments can be passed to the tool to identify the location and type of the synthesis tool associated with the project.

Note: The path to the synthesis tool and the tool type can also be defined after the Identify instrumentor or Identify debugger has been started as described in Command Line Configuration, on page 18.

The syntax of the two command-line arguments is:

-synplify_install *synthesisToolPath*

-synplify_tool *synplify* | *synplify_pro* | *synplify_premier* | *synplify_premier_dp*

The arguments are entered on the same command line in any order.

HAPS Board Bring-up Utility

The Identify debugger can be launched in the board bring-up mode using the -board_bringup option to the identify_debugger GUI command:

```
identify_debugger -board_bringup
```

Configuring the Synthesis Tool

When the Identify instrumentor or Identify debugger in launched from the synthesis tool GUI, the path to the synthesis tool and its tool type are defined.

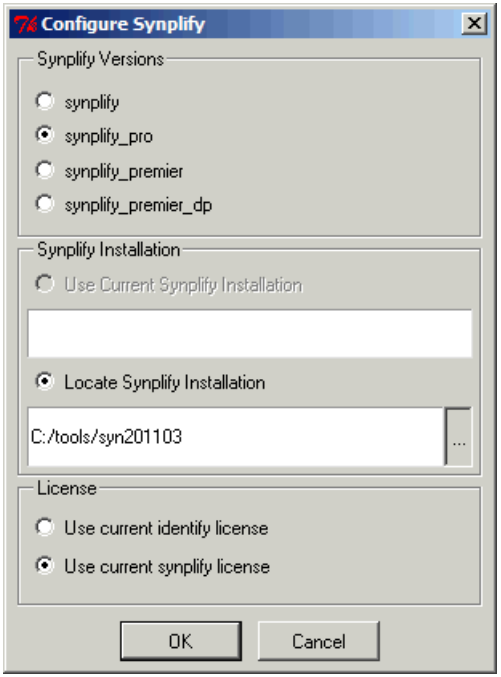
When the Identify instrumentor or Identify debugger is started independently and then opens a synthesis project file, the path to the synthesis tool executable and the tool type must be defined either when starting the Identify instrumentor or Identify debugger (see Synthesis Tool Pointers above) or after the tool is started as defined in the following subsections.

Note: The Identify instrumentor uses the synthesis tool compiler to compile the design.

GUI Configuration

To set the path to the synthesis tool from the Identify GUI:

1. Select Options->Configure Synplify from the menu to display the Configure Synplify dialog box.



2. In the dialog box:
   - Select the corresponding synthesis tool radio button in the Synplify Versions section.
   - Select the Locate Synplify Installation radio button and enter the path to the synthesis tool installation in the Synplify Installation section.
   - Select the appropriate license radio button in the License section.
3. Click the OK button to accept the values and close the dialog box.

Command Line Configuration

The path to the synthesis tool can also be set from the command line with a set_synplify_configuration command. The syntax for the command is:

set_synplify_configuration -type *synToolVersion* {**-locate** *installPath* |**-current**} **-license identify|synplify** [**-help**]

In the syntax, *synToolVersion* is synplify, synplify_pro, synplify_premier, or synplify_premier_dp and *installPath* is the path to the synthesis tool installation directory.

License Types

All of the startup execution mode commands accept an optional -licensetype argument to specify a license type other than the default. The license type can be either a vendor-specific license or a full license.

Note: The -licensetype argument is required when initially starting the Identify instrumentor or Identify debugger in the shell mode.

The following values are accepted by the -licensetype argument:

identinstrumentor	requests a full Identify instrumentor license
identinstrumentor_actel	requests a Microsemi-only Identify instrumentor license
identinstrumentor_altera	requests an Altera-only Identify instrumentor license
identinstrumentor_xilinx	requests a Xilinx-only Identify instrumentor license

identdebugger	requests a full Identify debugger license
identdebugger_actel	requests an Microsemi-only Identify debugger license
identdebugger_altera	requests an Altera-only Identify debugger license
identdebugger_xilinx	requests a Xilinx-only Identify debugger license

The following examples illustrate using the -licensetype argument. The first example opens the Identify instrumentor in the graphical interface with an Altera-only license, and the second example opens the Identify debugger in the shell mode with a full license.

```
identify_instrumentor -licensetype identinstrumentor_altera

identify_debugger_shell -licensetype identdebugger
```

To verify the license currently being used by the Identify instrumentor or Identify debugger, enter the command licenseinfo in the console window or at the shell prompt.

Note: Changing the license type with the -licensetype argument is valid only for the current session and does not change the default license type defined in the Select available license dialog box.

Synthesis Tool Command-Line Options

When starting the synthesis tool from the command line, an additional argument can be included to specify the location of the Identify installation directory. For example, the command

pathToSynplifyInstall/synplify_pro -identify_dir *pathToIdentifyInstall*

starts Synplify Pro in the GUI and sets the default installation path in the synthesis tool for launching the Identify tool set. The installation path specified appears in the Configure Identify Launch dialog box (Options->Configure Identify Launch) in the synthesis tool GUI.

The -identify_dir argument overrides any SYN_IDENTIFY_EXE environment variable default setting.

Custom Initialization Script

The Identify tool set can be customized using a TCL-style initialization script. This script is sourced on tool startup and allows you to define custom procedures and variables, or to program start-up behavior.

Script Locations

The Identify software first looks for initialization scripts, titled synrc.tcl, in the `/etc` directory of the Identify installation path and then in the user's home directory. On a Linux-based platform, this is the standard home directory. On Windows, the value of the environment value `USERPROFILE` is used. On a standard Windows installation, this path is generally represented as:

`c:/Documents and Settings/`*`userName`*`/synrc.tcl`

Scripting Priority

The script is first sourced from the installation directory and then from the home directory. Since the format is Tcl, a second user-specific script overrides Tcl procedures and variables previously defined.

Sample Initialization File

A sample synrc file is distributed with the software. The file resides in the /etc directory of the installation path. The file is named synrc.template.tcl and must be renamed to synrc.tcl to enable its functionality. This file contains some sample functions and can be used as an example of how to provide a custom waveform viewer for use in the Identify debugger. If you are interested in interfacing and using your own waveform viewer with the Identify debugger, refer to the application note "Interfacing Your Waveform Viewer" available on the Synopsys SolvNet web site.

SYNOPSYS®
Accelerating Innovation

CHAPTER 3

Command Concepts

All commands that the Identify tool set uses are divided into several specific categories. These categories, or concepts, separate the commands in terms of how these commands impact the system and also in terms of the tools within the system that utilize them. These concepts are defined in their respective sections below.

Each section in this chapter also contains a table that lists the commands alphabetically. The tables separate the commands into three different columns of information. These columns are:

- Tool Usage – describes in symbol form the tools that can utilize the specific commands. The symbols are:

Command available only in Identify instrumentor

Command available only in Identify debugger

Command available in both Identify instrumentor and Identify debugger

- Use this command ... – lists the command name or command example.
- To do this ... – defines the uses of the command and lists command conventions and standards.

This chapter contains:

General Commands

A general command is any command that allows you to control and access aspects of both the Identify instrumentor and Identify debugger. These commands access aspects of these tools, such as accessing the online command help and exiting the program.

	Use this command ...	To do this ...
	clear	Remove all the console output in the graphical user interface.
	exit	Exit the program and close its window.
	help	Display the online help system and a help topic about a command, if given.
	instrumentation	Manipulate incremental instrumentations.
	licenseinfo	Display product version and license status.
	log	Record commands and their output into a log file.
	project	Create, open, and save projects.
	searchpath	Set a search path to find HDL design files.
	setsys	Manipulate internal customization variables.
	source	Run a TCL script.
	transcript	Record commands into a transcript file.

File System Commands

File system commands allow you to navigate through the file system on your computer.

	Use this command ...	To do this ...
	cd	Change the working directory.
	pwd	Display the present working directory

Design Hierarchy Commands

Design hierarchy handling commands allow you to navigate through the design hierarchy of your HDL design.

	Use this command ...	To do this ...
	hierarchy	Navigate through the design hierarchy of your HDL design. A find option allows you to search for specific HDL design units within an HDL design.
	show	Display the HDL source code in the source window.

Design Instrumentation Commands

Design entry and instrumentation commands allow you to manipulate and instrument your HDL design and to insert the IICE.

	Use this command ...	To do this ...
	breakpoints	Instrument breakpoints as HDL source level trigger conditions of the IICE.
	compile	Import and compile HDL design files.
	device	Describe device-specific parameters used to implement your design.
	encryption	Set the current password to use before encrypting or decrypting a file.
	iice	Duplicate IICE Configuration dialog box functions including defining sample clock, setting counter width for complex triggering, defining trigger conditions and states, and sampling method.
	signals	Instrument signals for sampling and/or triggering in the IICE.
	write instrumentation	Write the instrumented HDL design files to a specified directory.

Design Debugging Commands

Design debugging commands allow you to debug your HDL design, interacting with the IICE and analyzing sample data at the RTL HDL source level.

	Use this command ...	To do this ...
	activation	Save or reload a set of trigger settings
	chain	Set up the JTAG chain of devices to be debugged.
	com	Set up the communication to the on-chip hardware, including cable and port settings.
	device	Query the type of target device(s) used to implement your design.
	encryption	Set the current password to use before encrypting or decrypting a file.
	haps	Queries the hardware to generate the requisite Tcl file for Certify board generation.
	idcode	Set up and maintain a table of device ID codes.
	iice	Query the IICE Configuration dialog box settings selected during the instrumentation phase including sample clock, counter width for complex triggering, trigger conditions and states, and selected sampling method.
	logicanalyzer	Configures the logic analyzer for real-time debugging.
	remote_trigger	Sets/resets events on a specified debugger instantiation, on all debuggers in a multiple debugger configuration, or on specific IICE units in a multi-IICE configuration.

	Use this command ...	To do this ...
	run	Arm the IICE with currently activated trigger conditions and wait for a trigger. When the trigger occurs, acquire and display the sample data.
	statemachine	Define the behavior of the state machine triggering hardware.
	stop	Activate or deactivate a HDL source level breakpoint.
	watch	Activate or deactivate a HDL source level watchpoint.
	waveform	Implement a waveform viewer
	write samples	Write the sample data in text format.
	write vcd	Write the sample data to a Verilog Change Dump (vcd) format.
	write vhdlmodel	Write the sample data in a VHDL model format.

SYNOPSYS®
Accelerating Innovation

CHAPTER 4

Alphabetical Command Reference

All commands are listed alphabetically in this chapter. Each command contains syntax, argument return values, default values, and examples.

activation 

Allows you to save or reload a set of trigger settings (enabled watchpoints and breakpoints). Including the -sample option causes the sample data to be loaded or saved with the trigger settings. If the optional *activationName* argument is included, the named activation is loaded or saved; if *activationName* is omitted, last_run.adb is used as the default activation name. The activation clear and activation list commands clear the current trigger settings and list all of the saved activations for the current instrumentation, respectively.

Syntax

activation load|**save** [**-sample**] [*activationName*]

activation clear|**list**

Command Example

```
activation load -sample instr_trial1
```

breakpoints 

Instructs the Identify instrumentor to add or delete special debug logic to or from the specified IICE. This debug logic implements breakpoint-style RTL source-level trigger conditions.

Syntax

breakpoints add|**delete** [**-iice** *iiceID*|**all**] *breakpointName* [*breakpointName* ...]

breakpoints map *breakpointName MictorPinName*

Arguments and Options

For the add and delete options, one or more breakpoints can be added or deleted at the same time.

add *breakpointName* [*breakpointName* ...]

delete *breakpointName* [*breakpointName* ...]

The map option is used exclusively with the real-time debugging feature to assign a breakpoint to a Mictor connector pin. In the above syntax, *MictorPin-Name* is the concatenation of the Mictor board HapsTrak connector location, the Mictor connector name, and the Mictor pin name separated with periods. For example, 3.M1.D3e is the D3e pin of Mictor connector M1 on the Mictor board installed in HapsTrak connector 3.

Breakpoint names consists of two components:

- The full hierarchical path of the HDL design unit that denotes the underlying control statement of the breakpoint.
- The HDL source code location given by the filename and the line number of the breakpoint.

These two components together ensure that each breakpoint has a unique name for identification purposes.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the IICE (*iiceID*) where the breakpoint is to be added or deleted. If the argument all is specified, the corresponding breakpoint is added to or deleted from each IICE.

Command Example

```
breakpoints add /beh/arb_inst/beh/process_83/case_88/arb.vhd:90

breakpoints delete -iice trap2
   /beh/blk_xfer_inst/beh/process_85/case_97/xfer.vhd:107

breakpoints map /beh/process_50/case_88/if_90/alu.v:72 3.M1.D5e
```

See Also

- stop, on page 88

cd 

Changes the present working directory in the file system to a different designated directory.

Syntax

cd *directory*

Arguments and Options

directory

Specifies the designated directory name. You must use forward slashes to describe relative and absolute path names irrespective of the operating system. On a Windows-based platform, the directory may include a drive letter followed by a colon.

Command Example

```
cd c:/temp
```

```
cd ../homedirs/adam
```

See Also

- pwd, on page 72

chain 

Sets up and manipulates the JTAG chain of devices. Because more than one device can be connected in a JTAG chain, the commands allows you to setup the JTAG chain representation in the Identify debugger to select the particular device to be debugged.

Syntax

chain add *deviceName instructionRegisterWidth*

chain clear

chain info [**-raw**|**-active**]

chain replace *position chipID instructionRegisterLength*

chain select *chipID*

Arguments and Options

add *deviceName instructionRegisterWidth*

Creates and labels a device and assigns that device with an instruction register width. Every device attached to the JTAG must be identified by a unique name. This device name can include any alpha-numeric characters. Spaces and other characters cannot be used.

The instruction register is an N-bit register that holds the OPCODE for the JTAG controller. Every device has a specific instruction register width, which can be found in the device's Data Book.

clear

Deletes the current chain description.

info

Displays the chain description.

info -raw

Returns a machine readable JTAG chain description. The chain is represented by a Tcl list of chain elements where each element is a two-item Tcl list specifying the device name and instruction register width. Example:

```
{{device_a 8} {device_b 10}}
```

info -active

Returns the name of the device that is currently selected for debugging.

chain replace *position chipID instructionRegisterLength*

Changes the name or register length of a device that has been previously defined using the chain add command. In the command syntax, *position* is the value shown by the chain info command for the device to be replaced.

select *deviceName*

Selects a device for system debugging. Only devices added and labeled using chain add can be selected.

Command Example

```
chain add fpga 5

chain select fpga

chain info -active

chain replace 1 new_fpga 8
```

See Also

- device, on page 39
- com, on page 35

clear 

Removes all the console output in the graphical user interface. This command is only supported in the graphical modes.

Syntax

clear

Arguments and Options

none

com 

Sets up and manipulates communication settings between the Identify debugger and the Intelligent In-Circuit Emulator (IICE).

Syntax

com cabletype [*type*]

com cableoptions *option* [*value*]

com check

com port [**lpt1**|**lpt2**|**lpt3**|**lpt4**]

Arguments and Options

cabletype [*type*]

Describes the type of cable connecting the system to the hardware being analyzed. The supported cable types are byteblaster, xilinxparallel, xilinxusb, xilinx-auto, Microsemi_BuiltinJTAG, JTAGTech3710, Altera_BuiltinJTAG, and demo. A umrbus selection in also available to indicate that the UMRBus is to be used as the communication interface between the hardware and the host machine running the Identify debugger.

cableoptions *option* [*value*]

Specifies or reports cable-specific option settings:

byteblaster_port [*integer*] – specifies the parallel port number; default is 1 (lpt1).

flashPro_trst [*string*] – specifies the setting of the TRST (tristate) pin. Accepted values (*string*) are off, toggle, low, and hi; the default is off.

flashProLite_trst [*string*] – specifies the setting of the TRST (tristate) pin. Accepted values (*string*) are off, toggle, low, and hi; the default is off.

flashPro3_trst [*string*] – specifies the setting of the TRST (tristate) pin. Accepted values (*string*) are off, toggle, low, and hi; the default is off.

JTAGTech_port [*integer*] – specifies the interface card address (0 to 255). The default address is 0.

JTAGTech_tapnum [*integer*] – specifies the active tap port on the JTAG Technologies tap pod. Values (*integer*) range from 1 to 4; the default is 1.

JTAGTech_type [**PCI**|**USB**] – specifies the type of computer interface connection (parallel or USB); the default is PCI (parallel).

xilinxparallel_port [*integer*] – specifies the parallel port number from 1 to 4; the default is 1 (lpt1).

xilinxparallel_speed [*integer*] – specifies the parallel port communications speed; acceptable values are 5000000 (5MHz), 2500000 (2.5 MHz, and 200000 (200kHz); the default is 5000000.

xilinxusb_speed [*integer*] – specifies the USB port communications speed; acceptable values are 24000000 (24MHz), 12000000 (12 MHz), 6000000 (6 MHz), 3000000 (3 MHz), 1500000 (1.5MHz), or 750000 (750 kHz); the default is 12000000.

check

Performs a connectivity check on the JTAG cable connection.

port [**lpt1**|**lpt2**|**lpt3**|**lpt4**]

Specifies the host computer parallel port to which the JTAG cable is connected. The supported ports are lpt1, lpt2, lpt3, and lpt4.

Command Example

```
com cabletype byteblaster
com cableoptions byteblaster_port 2
com port lpt1
```

See Also

- chain, on page 33

compile 

Prints a list of the design files and the respective order in which they are read.

Syntax

compile list

Arguments and Options

list [-vhdl|-verilog]

Prints a list of the design files and the respective order in which they are read. If the -vhdl or -verilog option is included, limits the list to only the specified file type.

Command Example

```
compile list -vhdl
```

See Also

- *searchpath*, on page 76

device 

Defines device-specific parameters used to implement the instrumented HDL design.

Syntax

device estimate [**-iice all**|*iiceName*] [**-resources** | **-noresources** | **-raw**]

device jtagport [**builtin**|**soft**|**umrbus**]

device prepare_incremental [**0**|**1**]

device skewfree

device technologydefinitions [**0**|**1**]

device xilinxinsertbufg

device xilinxjtagaddr1

device xilinxjtagaddr2

device xilinxusesrl16

device capimbaseaddr

Arguments and Options

estimate or **estimate -iice all**

Reports total number of instrumented signals and estimated resource utilization for the current implementation.

estimate -iice *iiceName*

Reports total number of instrumented signals and estimated resource utilization for the named IICE (*iiceName*) for the current implementation.

estimate -resources or **estimate -resources -iice all**

Reports only the estimated resource utilization for current implementation.

estimate -resources -iice *iiceName*

Reports only estimated resource utilization for the named IICE (*iiceName*) for the current implementation.

estimate -noresources or **estimate -noresources -iice all**

Reports only the number of instrumented signals for current implementation.

estimate -resources -iice *iiceName*

Reports only the number of instrumented signals for the named IICE (*iiceName*) for the current implementation.

estimate -raw

Displays the instrumented signal information and estimated resource utilization for the current implementation in a machine-readable format.

jtagport [**builtin**|**soft**|**umrbus**]

Determines if the built-in JTAG port of the target device is used for the IICE connection or if the Synopsys test port is used. Selection can only be set in the Identify instrumentor. With no argument specified, the current setting is displayed. The following selections are available:

builtin

Specifies that the JTAG port built into the target device is the port used. No extra user pin is required. This is the default value when the device family specified is other than generic.

soft

Specifies that the IICE communicates through a JTAG TAP controller that is automatically inserted by the Identify instrumentor. The Synopsys JTAG port requires four additional user pins.

umrbus

Specifies that the UMRBus is to be used as the communication interface between the hardware and the host machine running the Identify debugger (the JTAG port is not used).

prepare_incremental

Sets up the Identify instrumentor to support incremental changes to the instrumented signals.

skewfree [**0**|**1**]

Causes the IICE to be built using skew-resistant hardware when no global clock resources are available for the JTAG clock. When this option is enabled (1), master-slave flip-flops are used on the JTAG chain to prevent clock skew from affecting the logic. This setting also causes the Identify instrumentor to NOT explicitly define the JTAG clock as requiring global clock resources. The skewfree option is disabled (0) by default.

technologydefinitions [**0**|**1**]

Disables/enables the generation of black boxes for undefined module definitions. This option is available only in Identify instrumentor and is enabled by default.

xilinxinsertbufg [**0**|**1**]

Due to the timing requirements of the JTAG clock in Xilinx designs, the Identify instrumentor automatically adds a BUFG component to this clock signal to ensure that the signal is implemented using the chip's global clock resources.

If you prefer to have the synthesis tool detect and add the BUFG component, disable (0) this option to change this behavior. Use caution with this option; if the JTAG clock is either not in a global clock buffer or is implemented using skew-free hardware, the debug logic will not function properly. The skewfree option overrides the behavior of this setting, as no BUFG is inserted for skew-free hardware. The xilinxinsertbufg option is enabled (1) by default.

xilinxjtagaddr1 [**user1**|**user2**|**user3**|**user4**]

Selects the first user instruction register for Virtex-4, Virtex-5, and Virtex-6 boundary scan cells for the built-in JTAG controller. The default is user3. This option is available only in the Identify instrumentor.

xilinxjtagaddr2 [**user1**|**user2**|**user3**|**user4**]

Selects the second user instruction register for Virtex-4/Virtex-5 boundary scan cells for the built-in JTAG controller. The default is user4. This option is available only in the Identify instrumentor.

xilinxusesrl16 [**0**|**1**]

Determines if the IICE uses shift registers for the debug logic in Xilinx Virtex-II designs when enabled (1). If shift registers are used, the area cost of the debug logic can potentially be reduced. The xilinxusesrl16 option is enabled (1) by default.

Note: Valid values must be set for the above options before you instrument your design.

---

capimbaseaddr *baseAddress*

Specifies the address of a CAPIM inserted for UMRBus communication. In a multi-FPGA debug environment, a CAPIM is inserted into each FPGA where signals are instrumented. The base address value is decremented with each CAPIM added. The default base address is 57.

Command Example

```
device estimate -iice IICE -noresources

device jtagport builtin

device skewfree 1
```

See Also

- chain, on page 33
- com, on page 35

encryption 

Sets the current password to use before encrypting or decrypting a file. In the Identify instrumentor, this command sets the password to be used when writing out an encrypted file with the write instrumentation command. In the Identify debugger, this command is used to set the password to enable encrypted files to be displayed.

Note: Setting the password with this command displays the password on the screen and in any log files that you create. If this is a concern, use only the graphical interface when instrumenting and debugging designs that use the encryption feature.

Syntax

encryption set_passwd *password*

Arguments and Options

set_passwd *password*

The set_passwd argument requires a single string (*password*) entry. The new password is stored for decrypting/encrypting until it is changed or until the Identify instrumentor or Identify debugger is shut down.

Note: Passwords are the user's responsibility; Synopsys cannot recreate a lost or forgotten password.

Command Example

```
encryption set_passwd xyzzy
```

See Also

- write instrumentation, on page 96
- project, on page 71

exit 

Exits the program and closes the window.

Syntax

exit

Arguments and Options

None

Command Example

```
exit
```

help 

Displays the online help system and a help topic about a command.

Syntax

help [*commandName*]

Arguments and Options

commandName

Displays help text about the specified command. If the *commandName* argument is omitted, help descriptions for all commands are printed to the screen.

haps 

Queries the hardware to generate the requisite Tcl file for Certify board generation and to performs the verification tests.

Syntax

haps
board
prog *binFile devID*
setvcc *voltage*
setclk *clockName frequency*
restart
list [*testName*]
run [*testName*|**All**]
confscr *scriptFileName*
vbgen *tclFile*
help [*testName*]

Arguments and Options

board

Displays the board status to the screen. Status includes clock and voltage settings, reset configuration, daughter card connections, firmware version, and board serial number.

prog *binFile devID*

Programs the FPGA identified by *devID* with the specified `bin` file. The *devID* value begins with 1 which corresponds to the first FPGA on the board.

setvcc *voltage*

Sets the I/O voltage for the board regions. The acceptable values for *voltage* are 1v5, 1v8, 2v5, and 3v3 (HAPS-70 system voltage regions are limited to a maximum of 1.8 volts).

setclk *clockName frequency*

Sets the frequency for the global input clock identified by *clockName* to the specified frequency. The *frequency* value is in kHz unless specified otherwise. For example, the command haps setclk GCLK1 150MHz specifies a clock frequency of 150 MHz for GCLK1.

restart

Restarts the board.

list

Lists the available local board tests.

run [*testName*|**All**]

Runs a particular test or runs all local board tests. In the above syntax, *testName* is one of:

umr_check *fpgaID frequency* – verifies the basic functionality of the UMRBus. In the syntax:

fpgaID – indicates which FPGA device is to be tested. The default is 1, which is the first FPGA device on the board.

frequency – sets the frequency for GCLK1 which is used in the test of the design. The default frequency is 140MHz.

con_speed *frequency* **fast|sweep** – verifies the connectivity between HapsTrak connectors as well as the speed at which HSTDM can run. In the syntax:

frequency – sets the frequency at which HSTDM is to be verified.

fast|sweep – sets the run mode. The default is fast mode. When mode is set to sweep, the test sweeps every channel of the connection which can require up to four hours to complete.

self_test – replaces the traditional self test with an STB2 test card.

clock_check – reports the clock frequency of each GCLK output to allow of all of the GCLK frequencies to be verified.

When the All argument is used, runs all local tests with the individual test parameter defaults. For more complete test details, see the individual test descriptions under the haps command in the *Certify Command Reference* and see *HAPS Board Bring-up Utility* in Chapter 11, *Connecting to the Target System*, in the Identify User Guide.

confscr *scriptFileName*

Runs confprosh tcl scripts. For example, the confscr option can be used to source a HAPS clock and voltage-region configuration script; the user could then run clock checks to verify the on-board clock configuration.

vbgen *tclFile*

Queries the HAPS system and generates a corresponding Tcl file for Certify board file generation.

help [*testName*]

Shows help information. If *testName* is included, shows help for the specified test.

Command Example

```
haps run umr_check 2 180

haps setclk GCLK1 150MHz

haps setvcc 1v8
```

hierarchy 

Navigates through the design hierarchy and shows design and hierarchy elements in the HDL design. These design elements include the following types, depending on the HDL language used to describe the design:

- Entity – VHDL design unit type.
- Module – Verilog design unit type.
- Instance – VHDL or Verilog design unit type.

Syntax

hierarchy add [*options*] *element* [*element* ...]

hierarchy cd *hierarchyPath*

hierarchy delete [*options*] *element* [*element* ...]

hierarchy find [*options*] [*hierarchyPath*]

hierarchy ls [**-long**] [**-recursive**] [**-all**] [*hierarchyPath*]

hierarchy pwd

hierarchy toplevel

Arguments and Options

add [*options*] *element* [*element* ...]

Connects all signals or breakpoints in the specified hierarchical element to the IICE. The add argument applies only to the Identify instrumentor.

The following add argument options are available:

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify which IICE (*iiceID*) to connect. If the argument all is specified, the signals or breakpoints are connected to each IICE.

-sample

Connects all signals in the specified hierarchical element to the IICE sample buffer.

-trigger

Connects all signals in the specified hierarchical element to the IICE trigger logic.

-breakpoint

Connects all breakpoints in the specified hierarchical element to the IICE.

-recursive

Allow hierarchies to be traversed when a wildcard is included in the *element* argument.

Note: The -sample, -trigger, -breakpoint, and -recursive options can be combined in a single add argument.

---

cd *hierarchyPath*

Changes the current design hierarchy to the one specified by *hierarchyPath*. Either a relative or an absolute hierarchical path name can be used.

```
cd /
```

Changes the current design hierarchy to the top level of the hierarchy.

```
cd ..
```

Changes the current design hierarchy to next higher level.

delete [*options*] *element* [*element* ...]

Disconnects all signals or breakpoints in the specified hierarchical element from the IICE. The delete argument applies only to the Identify instrumentor.

The following delete argument options are available:

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify which IICE (*iiceID*) to disconnect. If the argument all is specified, the signals or breakpoints are disconnected from each IICE.

-signal

Disconnects all sample and trigger signals in the specified hierarchical element from the IICE.

-breakpoint

Disconnects all breakpoints in the specified hierarchical element from the IICE.

Note: The -signal and -breakpoint options can be combined in a single delete argument.

---

find [*options*] [*hierarchyPath*]

Searches for specific HDL design units and lists those elements. Use this command to locate specified design units in the compiled HDL design file. The search is started from the specified hierarchical path. If you do not provide *hierarchyPath*, the search starts from the current working hierarchy.

The following find options are available:

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the IICE (*iiceID*) to be searched. If the argument all is specified, each IICE is searched.

-name *elementName*

The HDL element name to be located.

-noequiv

Limits the search to named path only and does not search equivalent paths.

-type instance|breakpoint|signal|*

The type of HDL element for the target search. If * is entered, search includes all elements.

-ls

Prints verbose information for each HDL element found.

-stat *status*|*

Serves as a filter to search for an HDL element with a specific instrumentation status. If * is entered, any instrumentation status is included in the search. The *status* argument takes the following options:

- **disabled** – limits search to disabled watchpoints, breakpoints, and other disabled HDL design units (available only in Identify debugger).
- **enabled** – limits search to enabled watchpoints, breakpoints, and other enabled HDL design units (available only in Identify debugger).
- **instrumented** – limits search to the sampling clock, and watchpoints and breakpoints that have been marked as instrumented (available only in Identify instrumentor).
- **not-instrumented** – limits search to watchpoints and breakpoints that have not been instrumented (available only in Identify instrumentor).
- **sample_only** – limits search to sample-only watchpoints (available only in Identify instrumentor).
- **trigger_only** – limits search to trigger-only watchpoints (available only in Identify instrumentor).

-maxdepth *integer*

Limits search to a maximum depth within the hierarchy tree.

-all

Lists “hidden” HDL design units, such as signals/breakpoints within dead code or, in the Identify debugger, breakpoints that were not instrumented. By, default, HDL elements with enabled status are searched.

ls [**-long**] [**-recursive**] [**-all**] [*hierarchyPath*]

Displays all information about the HDL design units within the current design hierarchy. You can display this design unit information in a long listing using the -long option or you can display this information recursively using the -recursive option. The -all option shows all HDL elements including hidden elements.

pwd

Lists the current HDL design hierarchy.

toplevel

Shows top-level hierarchy name.

Command Example

```
hierarchy cd ..

hierarchy cd /top/u1/arui

hierarchy ls -recursive

hierarchy find -type breakpoint -stat instrumented
```

See Also

- show, on page 78

idcode 

Sets up and maintains a table of device ID codes. The ID code information is used for auto-detection of the devices on the JTAG chain during debugging. If the chain can be successfully detected, you do not need to manually specify the chain using the chain command.

Syntax

idcode add [**-quiet**] *idcode deviceName instructionRegisterWidth*

idcode clear

idcode info [**-raw**]

Arguments and Options

add [**-quiet**] *idcode deviceName instructionRegisterWidth*

Creates an entry in the device table for a given device.

The *idcode* argument should be a binary representation of a 32-bit number in the form of a string. The string can contain 'x' entries for bits that are irrelevant.

The *deviceName* argument can be any descriptive string. The string must be quoted if it includes spaces.

The *instructionRegisterWidth* argument takes an integer value. Every device has a specific instruction register width, which can be found in the device's Data Book.

The -quiet option adds the device, but does not display a user notification.

clear

Deletes the entire ID code table.

info [**-raw**]

Returns a description of the device table. The table is represented by a Tcl list of device elements where each element is a three item Tcl list specifying the ID code, device name, and instruction register width. Example:

```
{11001100110011001100110011001100 device_a 8}
{00001100110011001100110011001111 device_b 10}
```

The optional -raw option generates the description in a machine-readable format.

Command Example

```
idcode add 00100000001110001000100010000 device_type 8

idcode add -quiet 00100000001110001000100010000 "device type" 8

idcode clear
```

See Also

- device, on page 39
- chain, on page 33

iice 

Duplicates the functionality of the IICE Configuration dialog box.

Syntax

iice clock|**controller**|**current**|**delete**|**info**|**list**|**new**|**rename** |
sampler|**assignmentsreport** [*option*]

Arguments and Options

iice clock [*options*] [*signalName*]

Defines the signal to be used for the IICE sample clock. The *signalName* is the full hierarchical path name to the signal. You can select any signal within the HDL design as the sample clock. However, this signal cannot be sampled itself while used as the sample clock. This option can only be used during instrumentation. If *signalName* is not specified, the option returns the name of the IICE clock.

-edge positive|**negative**

Specifies the active edge of the clock (positive or negative) when an IICE sample clock is specified. The -edge option is only available in the Identify instrumentor; the default edge is rising (positive).

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify/report the controller parameters for the specified IICE (*iiceID*). If the argument all is specified, the controller parameters apply to each IICE.

iice controller [*options*] [**none**|**counter**|**statemachine**]

Specifies IICE controller configuration; simple triggering (none), complex triggering (counter), or state machine. The following options are supported:

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify/report the controller parameters for the specified IICE (*iiceID*). If the argument all is specified, the controller parameters apply to each IICE.

-countermode [**events**|**cycles**|**watchdog**|**pulsewidth**]

Selects the complex counter mode. The value *n* referenced below is the value set by the countervalue option (applies only to Identify debugger).

events

Stops sampling after the trigger condition occurs for the *n*+1'th time. This is the default value for -countermode.

cycles

Stops sampling *n* cycles after the trigger condition occurs.

watchdog

Stops sampling if the trigger condition does not occur for *n* consecutive cycles.

pulsewidth

Stops sampling when the trigger condition has met *n* consecutive cycles. The number *n* is controlled by the current setting of countervalue.

If no argument is given, the current mode is returned.

-counterval *unsignedInteger*

Sets a value for the complex counter (applies only to Identify debugger).
unsignedInteger

Load the given value into the complex counter. This value must fit into the complex counter width as defined in the Identify instrumentor. The default value for the complex counter is 0 which disables the counter.

If no value is given, the current setting is returned.

-counterwidth *integer*

Instruments a versatile counter of variable size for complex triggering (applies only to Identify instrumentor). If no argument is given, the command shows the current counter width. An integer parameter in the range between 1 and 32 specifies a new counterwidth. 0 suppresses the creation of any counter. All other values are invalid. The default value for the counterwidth is 16 (the IICE contains a 16-bit complex counter).

-triggerconditions *integer*

Used when instrumenting a design for state-machine triggering (applies only to Identify instrumentor). This command specifies the number of trigger conditions available for state-machine triggering. The range is from 1 to 16. The default value is 4. If no argument is given, the command shows the current pattern-tree setting.

This option is a critical setting with respect to instrumentation cost. Choosing a trigger setup with the minimum amount of trigger conditions is recommended to reduce resource usage in the instrumentation. Choosing a trigger-condition value greater than 1 requires that multiple trigger states be created. Use the triggerstates option to specify the desired number of states.

-triggerstates [*integer*]

Used when instrumenting a design to use state-machine triggering (applies only to Identify instrumentor). This option specifies the maximum number of states instrumented in the state machine. The range is 2 to 16; powers of 2 are preferable as other integers limit functionality and do not provide any cost savings. The default is 4. If no argument is given, the option shows the current triggerstates setting.

-exporttrigger 0|1

Determines if the master trigger signal of the IICE hardware is exported to the top-level of the instrumented design (applies only to Identify instrumentor). Enables (1) or disables the creation of a trigger port. Export trigger port creation is disabled by default.

-importtrigger *integer*

Determines if the master trigger signal of the active IICE hardware includes any triggers received from external sources (applies only to Identify instrumentor). Specifying a value between 1 and 8 creates a corresponding number of input ports.

Note: When using an external trigger, the pin assignment for the corresponding input port must be defined in the synthesis or place and route tool.

-crosstrigger 0|1 [**-iice** *iiceID*]

Enables (1) or disables an IICE to include trigger signals from other IICE units when determining its trigger condition (applies only to Identify instrumentor). If the -iice argument is omitted, the command applies to the current IICE.

-crosstriggermode disabled|any|all|after -crosstriggeriice *iiceID*|**all**

Determines the trigger conditions in the Identify debugger when the IICE controller is set to simple or complex-counter triggering. The following options are supported:

disabled

Destination IICE triggers normally (triggers from source IICE units are ignored).

any

Destination IICE triggers when any source IICE triggers or on its own internal trigger.

all

Trigger occurs when all events, irrespective of order, occur at all IICE units including local IICE unit.

after -crosstriggeriice *iiceID*|**all**

Trigger occurs after source IICE triggers coincident with next destination IICE trigger. The -crosstriggeriice argument specifies a specific source IICE unit (*iiceID*) or all source IICE units (all).

iice current [*iiceID*]

Used when more than one IICE is defined to select the active IICE (*iiceID*). If the *iiceID* argument is omitted, reports the ID of the currently active IICE. Note that *iiceID* is case sensitive.

iice delete *iiceID*

Deletes the specified IICE (*iiceID*). The iice delete command is only available in the Identify instrumentor.

iice info [*iiceID*]

Reports the status of the specified IICE (*iiceID*). If the *iiceID* argument is omitted, reports the status of the currently active IICE.

iice list

Lists the IDs (names) of each defined IICE.

iice new [*iiceID*] [**-type rtd**|**regular**]

Creates a new IICE with the name *iiceID*. If the *iiceID* argument is omitted, the new IICE is named IICE_*n* where *n* is the next sequential integer. The -type option indicates if the IICE is to be configured for real-time debugging (rtd) or normal debugging (regular). For more information on the real-time debugging feature, see the *User Guide*.

The iice new command is only available in the Identify instrumentor.

iice rename *iiceID*

Renames the currently active IICE to the name specified (*iiceID*). The iice rename command is only available in the Identify instrumentor.

iice sampler [*options*]

The following iice sampler options are supported in the Identify instrumentor:

-iice {*iiceID*|**all**} **internal_memory**|**hapssram**
-compression 0|**1**
-sram *option* [*value*]
-rtd {mictorlocs {*location* [*location* ...]**}**|**board** *boardType***}**
-depth *depthValue*
-qualified_sampling 0|**1**
-always_armed 0|**1**

The following iice sampler options are supported in the Identify debugger:

-triggertime early|**middle**|**late**
-samplemode normal|**qualified_fill**|**qualified_intr**|**always_armed**
-runselftest 0|**1**
-datacompression 0|**1**
-enablemask 0|**1** [**-msb** *integer* **-lsb** *integer*] *signalName*
-group *interger*

Identify Instrumentor iice sampler Options

-iice *iiceID*|**all**

Used when more than one regular IICE is defined to specify/report the IICE sampler parameters for the specified IICE (*iiceID*). If the argument all is specified, the IICE sampler parameters apply to each qualified IICE.

internal_memory|**hapssram**

The internal_memory/hapssrm argument specifies the type of RAM used to capture the sample data; internal_memory uses local RAM (the default) and hapssram uses the memory on a HAPS SRAM daughter board (available only with HAPS board configurations; unavailable with real-time debug IICE).

-compression 0|**1**

The -compression option determines if data compression is to be applied when the sample data is unchanged between cycles (the data is automatically decompressed when viewed). A value of 1 enables data compression. An internal default is set to force an update after 64 cycles of unchanging data. The -compression option applies only to regular IICE units and is not supported by real-time debug IICE.

-sram *option* [*value*]

The -sram option applies only when the *bufferType* is set to hapssram. The -sram options are described below. If *value* is not specified, the current setting is reported. The -sram option applies only to regular IICE units and is not supported by real-time debug IICE.

sramlocations [*location*]

The connector location where the daughter card or cards are physically connected. *Location* is one or more integers between 1 and 6 that represent the HapsTrak connectors of the FPGA under debug. Eligible locations are board dependent.

numberboardstack [*integer*]

The number of daughter cards stacked at the specified SRAM locations. *Integer* can be 1 or 2 and applies to all connector locations (the stack depth must be the same at all locations).

type [*value*]

The SRAM daughter card type. Currently, only the SRAM 1x1HTII daughter card is supported (type = 6).

clockfreq [*value*]

Specifies the frequency of the clock source for the SRAM. For more information on SRAM clocks, see *SRAM Clocks*, on page 50 in the *User Guide*.

clocktype external|internal *clockName*

Specifies the source of the SRAM clock. When internal is specified, the clock source must be identified.

-rtd *arguments*

The -rtd option applies only when the IICE type is set to rtd. The -rtd arguments for the real-time debugging feature are described below. The -rtd option applies only to real-time debug IICE and is not supported by regular IICE units.

mictorlocs *location* [*location ...*]

Specifies the location of the Mictor board (or boards) installed in the HapsTrak connectors. Values range from 1 through 6 and more than one location can be specified by separating the values with spaces.

board *boardType*

Specifies the HAPS board type. The *boardType* entered must be in all caps and must be appropriate for the Synopsys HAPS device setting in the Synopsys synthesis tool.

-depth *depthValue*

Changes the default sample depth of the IICE sample buffer to an assigned value *depthValue*. This option can only be used during instrumentation and is only supported by regular IICE units. The default setting for *depthValue* is 128.

-qualified_sampling 0|1

Enables/disables qualified sampling. When enabled (1), causes the Identify instrumentor to build an IICE block that is capable or performing qualified sampling. With qualified sampling, one data value is sampled each time the trigger condition is true so that you can follow the operation of the design over a longer period of time (for example, you can observe the addresses in a number of bus cycles by sampling only one value for each bus cycle instead of a full trace). The -qualified_sampling option applies only to regular IICE units and is not supported by real-time debug IICE. Using qualified sampling includes a minimal area and clock-speed penalty.

-always_armed 0|1

Enables/disables always-armed sampling. When enabled (1), the Identify instrumentor saves the sample buffer for the most recent trigger and waits for the next trigger or until interrupted. With always-armed sampling, a snapshot is taken each time the trigger condition becomes true so that you always acquire the data associated with the last trigger condition prior to the interrupt. The -always_armed option applies only to regular IICE units and is not supported by real-time debug IICE. Using always-armed sampling includes a minimal area and clock-speed penalty.

Identify Debugger iice sampler Options

-triggertime [**early**|**middle**|**late**]

Controls how a detected trigger affects data sampling (applies only to Identify debugger).

early

Approximately 10 percent of the sample data is pre-trigger and approximately 90 percent is post-trigger.

middle

Approximately 50 percent of the sample data is pre-trigger and approximately 50 percent is post-trigger. This is the default sample trigger.

late

Approximately 90 percent of the sample data is pre-trigger and approximately 10 percent is post-trigger.

-samplemode [normal|qualified_fill|qualified_intr|always_armed]

Selects the trigger mode (applies only to Identify debugger).

qualified_fill

Performs qualified sampling until the buffer is full.

qualified_intr

Performs qualified sampling until interrupted.

always_armed

Always-on triggering.

-runselftest 0|1

Runs self-test to verify the deep trace debug hardware configuration. The self-test writes data patterns to the external memory and reads back the data pattern written to detect configuration errors, connectivity problems, and SRAM frequency mismatches.

-datacompression 0|1

Compresses debugger data when the sample data is unchanged between cycles (the data is automatically decompressed when viewed). A value of 1 enables data compression. An internal default is set to force an update after 64 cycles of unchanging data.

-enablemask 0|1 [**-msb** *interger* **-lsb** *integer*] *signalName*

Used when -datacompression option is enabled to selectively mask individual bits or buses from being considered as changing values within the sample data.

-group *integer*

Selects multiplexed group of instrumented signals defined in the Identify instrumentor for activation in the Identify debugger. *Integer* is the number of the multiplexed group which ranges from 1 to 8.

iice assignmentsreport [**-filename** *fileName*]

Prints the real-time debugging IICE assignments report showing signal, breakpoint, and connector assignment information. Executing this command prior to assigning logic analyzer pods to Mictor pin groups lists only the signals/breakpoints assignments information. Executing the command after assigning logic analyzer pods to Mictor pin groups additional lists the pod assignments. Including the optional -filename argument writes the report to the specified file instead of to standard out.

Command Example

```
iice clock -edge falling clk2

iice controller -counterwidth 8 statemachine

iice current IICE_2

iice sampler -triggertime late

iice sampler -datacompression 1

iice sampler -enablemask 1 -msb 3 -lsb 0 ctrlbus1a

iice sampler -sram sramlocations {2 4 6}

iice sampler -iice IICE_2 -rtd {mictorloc {1 3 5}}

iice sampler -iice myIICE -rtd {board HAPS-64}
```

instrumentation 

Manipulates incremental instrumentations.

Syntax

instrumentation new -instr {*baseName***}**
-ncdfile {*pathtoFilename***.ncd** [*fpgaName*]**}** |
-dcpfile {*pathtoFilename***.dcp** [*fpgaName*]**}**

instrumentation info [**-raw**] *name*

instrumentation list

instrumentation load *name*

instrumentation save

Arguments and Options

new -instr {*baseName***}**
-ncdfile {*pathtoFilename***.ncd** [*fpgaName*]**}** |
-dcpfile {*pathtoFilename***.dcp** [*fpgaName*]**}**

info [*options*] *name*

Shows information about the specified instrumentation. If the -raw option is included, returns information in a machine readable format

list

Lists the existing instrumentations (applies only to Identify instrumentor).

load *name*

Loads an existing instrumentation into the Identify instrumentor.

save

Saves the current instrumentation settings (applies only to Identify instrumentor).

Command Example

```
instrumentation load instr_2

instrumentation new -instr {rev_1} -ncdfile {
   {./synplify_identify/mb_uA/pr_1/cnt_demo.ncd mb.uA}
   {./synplify_identify/mb_uB/pr_1/cnt_demo.ncd mb.uB} }

instrumentation new -instr {rev_1} -dcpfile
   {./synplify_identify/pr_1/post_route.dcp}
```

jtag_server 

Configures the JTAG server.

Syntax

jtag_server set -addr {*hostName|IP_address***} -port {***serverPort***} -logf {***logFfileName***}**

jtag_server get

Arguments and Options

set

Configures the JTAG server

-addr {*hostName|IP_address***}**

The IP address or the name of the server.

-port {*serverPort***}**

The port number over which the client and server communicate.

-logf {*logFfileName***}**

The name of the log file.

get

Returns the server host name or IP address, port number, and log file name.

Command Example

```
jtag_server -set -addr myhost -port 58015 -logf servercom.log

jtag_server get
   INFO: addr 127.0.0.1 port 57015 logf ipc_tcp_xilinx.log
```

licenseinfo 

Displays information about the product version and license status.

Syntax

licenseinfo

logicanalyzer 

Configures the logic analyzer for real-time debugging. The scan options define the target logic analyzer, the assignpod option describes the analyzer interface, and the submit option sends the data to the logic analyzer. Additional options display the most recently used logic analyzer scan settings (lastscansettings option) and show the logic analyzer's presently scanned pod and module information (pods option).

Syntax

logicanalyzer scan -latype tla -hostname *hostName* **-username** *userName* **-script** *scriptName* **-assignpodsauto yes**|**no**

logicanalyzer scan -latype la16700|**la16900 -hostname** *hostName* **-assignpodsauto yes**|**no**

logicanalyzer assignpod -micconpingrp *groupName* **-module** *moduleNumber* **-pod** *podIdentifier*

logicanalyzer submit

logicanalyzer lastscansettings

logicanalyzer pods

Arguments and Options

-latype

The type of logic analyzer interfaced to the Mictor connector. Recognized types are tla, la16700, and la16900.

-hostname

The name or IP address (*hostName*) for the Identify debugger host.

-username

The user name (*userName*) on the logic analyzer (Tektronix only).

-script

The name of the script (*scriptName*) to run to set up logic analyzer (Tektronix only).

-assignpodsauto

Determines if pods are automatically assigned to the Mictor connectors.

-micconpingrp

The Mictor connector pin group (*groupName*). The connector pin group is identified by the concatenation of the Mictor board HapsTrak connector location, the Mictor connector name, and the Mictor odd/even pin bank separated with periods. For example, 3.M1.e addresses the even bank of Mictor connector M1 on the Mictor board installed in HapsTrak connector 3.

-module

The module name.

-pod

The connector pod (slot) on the logic analyzer.

Command Examples

```
logicanalyzer scan -latype la16900 -hostname sisyphus
   -assignpodsauto yes

logicanalyzer assignpod -micconpingrp 2.M1.e -module 1
   -pod A2A3CK0
```

log 

Allows logging the console output in the graphical user interface to a file.

Syntax

log *fileName*|**on**|**off**

Arguments and Options

fileName

Starts logging to the specified file.

on

Starts logging to the last specified file or to the default files syn_di.log or syn_hhd.log.

off

Stops logging.

Command Example

```
log on

log off

log mylog.log
```

See Also

- transcript, on page 90

Note: This command is not supported in the command-line tools. Use the operating system capability to pipe the console input into a file.

project 

Opens existing projects and displays project information.

Syntax

project open [**-password** *password*] [*fileName*]

project name [**-path**]

Arguments and Options

open [**-password** *password*] *SynopsysFPGAprojectFile*

Performs a simple import of a Synopsys FPGA synthesis project (`prj`) file by extracting the design files, the device technology, and the design top level. This data is used to create an Identify implementation (applies only to Identify instrumentor). After extracting the files, the design is automatically compiled.

-**password** *password*

Specifies the password to use to decrypt an encrypted source file (applies only to Identify debugger). Note that setting the password with this command displays the password on the screen and in any log files that you create. If this is a concern, use only the graphical interface when instrumenting and debugging designs that use the encryption feature.

name [**-path**]

Returns the name of the current project. If the -path option is specified, includes the full path to the project.

Command Example

```
project open C:/space/designs/mydesign.prj

project open -password xyzzy demo_design.prj
```

See Also

- encryption, on page 43

pwd 

Displays the current working directory.

Syntax

pwd

See Also

- cd, on page 32

remote_trigger 

Triggers the event (stops data collection and downloads data).

Syntax

remote_trigger [**-all**|**-info**|**-pid** *processID*|**-iice** *iiceID*]

remote_trigger -set|**-reset** [**-pid** *processID*|**-iice** *iiceID*]

Arguments and Options

-all

Triggers the event for every IICE in all debugger instantiations on the corresponding machine.

-info

Lists the names of the triggers in the current debugger instantiation.

-pid *processID*

Triggers every IICE on the debugger instantiation identified by *processID*. To identify the process ID of the active debugger instantiation, enter pid at the command prompt. The default is to trigger every IICE in all debugger instantiations (-all).

-iice *iiceID*

Triggers the event only on the specified IICE in the current debugger instantiation. The default is to trigger every IICE in all debugger instantiations (-all).

-set [**-pid** *processID* |**-iice** *iiceID*]

Sets the trigger. If the -pid argument is specified, sets the trigger on every IICE on the debugger instantiation identified by *processID*; if the -iice argument is specified, sets the trigger only on the IICE unit specified by *iiceID*.

-reset [**-pid** *processID* |**-iice** *iiceID*]

Clears the trigger. If the -pid argument is specified, resets the trigger on every IICE on the debugger instantiation identified by *processID*; if the -iice argument is specified, resets the trigger only on the IICE unit specified by *iiceID*.

Command Example

```
remote_trigger

remote_trigger -info

remote_trigger -set -pid 12

remote_trigger -reset -pid 12

remote_trigger -set -iice IICE0
```

See also

- triggermode option – iice, on page 55
- triggertime option – iice, on page 55

run 

Arms the IICE with the current trigger settings and waits until the trigger condition has occurred and has been detected by the IICE. Once the trigger condition has occurred, the sample data is downloaded from the IICE and is displayed on the screen.

Syntax

run -iice *iiceID*|**all**

run -timeout *integer*

run -wait

run -remote_trigger *pid*|**0**

Arguments and Options

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) for triggering. If the argument all is specified, triggering applies to each IICE.

-timeout *integer*

Specifies the number of seconds that the Identify debugger waits for a trigger before stopping. Whenever a time-out occurs, the data buffer is automatically updated. A value of 0 disables the time-out feature.

-wait

Causes the IICE to wait for the hardware to stop running before returning.

-remote_trigger *pid*|**0**

Used when running multiple debugger instantiations to send a trigger to either the debugger instantiation identified by *pid* or to all debugger instantiations if 0 is specified when a local trigger condition is detected. To identify the process ID of the active debugger instantiation, enter pid at the command prompt.

Note: The run command does not stop running until the trigger occurs. If the trigger does not occur, the run command does not stop. To cancel the run command, you must click the Stop button in the Identify debugger menu bar. There is no stop command in the command shell.

Command Example

```
run -remote_trigger 1336
```

searchpath 

Sets a search path to find HDL design files during instrumentation or debugging.

Syntax

searchpath [*directoryList*]

Arguments and Options

Without an argument, the current search path is displayed.

[*directoryList*]

Searches the specified directories, in order, for design files. *DirectoryList* can take the form of the following:

- On a Windows platform: a semicolon-separated list of valid directories. Note that the Windows "\" separator is not allowed in path names.
- On a Linux platform: a colon-separated list of valid directories.

Default Value

By default, the search path is the current working directory.

Command Example

```
searchpath {C:/temp;D:/user/joe}
searchpath {/home/john:/home/designs}
```

See Also

- add option – compile, on page 38

setsys 

Sets and queries user customization variables.

Syntax

setsys list

setsys variables

setsys set lpt_address [*value*]

Arguments and Options

list

Lists all available variables with their respective values.

variables

Lists all available variables with a short description explaining their function.

set lpt_address [*value*]

Specifies the device address for the parallel port. This setting overrides the operating system defaults. *Value* ranges from 0 to 65535; the default value is "0". If no value is supplied, returns the current value.

Command Example

```
setsys set lpt_address 4095
```

show 

Displays an HDL source code context on the command line.

Syntax

show [**-***integer*] [**+***integer*] *fileName*:*lineNumber*

show *hierarchyPath*

Arguments and Options

[**-***integer*]

Displays a designated number (*integer*) of lines before the *lineNumber* listed. The default number of lines displayed is 5.

[**+***integer*]

Displays a designated number (*integer*) of lines after the *lineNumber* listed. The default number of lines displayed is 5.

fileName:*lineNumber*

Displays an HDL design file at the selected *lineNumber*.

hierarchyPath

Displays the HDL design unit described by the given hierarchical path.

Command Example

```
show -8 +16 cpu.vhd:29
show /top/u1/reset_n
```

signals 

Instructs the Identify instrumentor to create special debug logic for the IIICE to sample a signal from your HDL design or to delete the debug logic and return the signal to its "not instrumented" status. The group options assign and report signals in multiplexed groups.

Syntax

signals add [*options*] *sigName* [*sigName* ...]
signals add [*options*] **-msb** *value* [**-lsb** *value*] *sigName*

signals delete [*options*] *sigName* [*sigName* ...]
signals delete [*options*] **-msb** *value* [**-lsb** *value*] *sigName*

signals group {*groupNumber***}** *sigName* [*sigName* ...]
signals group -show all|*sigName* [*sigName* ...]
signals group -show_tab all|*sigName* [*sigName* ...]

signals map {*sigName***}** *connectorPin*

Arguments and Options

add *sigName* [*sigName* ...]
add -msb *value* [**-lsb** *value*] *sigName*

SigName is the full hierarchical path name of the signal. In the first syntax statement, more than one signal can be specified for sampling or triggering by including additional signal names separated by spaces. In the second syntax statement, the -msb and -lsb arguments specify a bit or bit range of a bus. Note that when specifying partial buses:

- Use the -msb argument (without an -lsb argument) to specify a single bit
- Observe the index order of the bus. For example, when defining a partial bus range for bus [63:0] (or "63 downto 0"), the MSB value specified must be greater than the LSB value. Similarly, for bus [0:63] (or "0 upto 63"), the MSB value specified must be less that the LSB value.

The following options are available with the add argument:

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) for signal sampling/triggering. If the argument all is specified, signal sampling/triggering applies to each IICE.

-sample

Connects the specified signal or signals to the IICE sample buffer.

-silent

Suppresses resource estimation display when a signal is added (by default, adding a signal automatically updates the total instrumentation requirements in the console window).

-field *fieldName*

Instruments the named field or record for the specified signal (partial instrumentation).

-trigger

Connects the specified signal or signals to the IICE trigger logic.

Note: The -sample and -trigger options can be combined or both options can be omitted to specify a signal for both sampling and triggering.

---

delete *sigName* [*sigName* ...]
delete -msb *value* [**-lsb** *value*] *sigName*

SigName is the full hierarchical path name of the signal. In the first syntax statement, more than one signal can be specified for deletion by including additional signal names. In the second syntax statement, the -msb and -lsb arguments identify a previously specified bit or bit range of a bus.

Note: When a partial bus is defined, you must explicitly delete the individual bus segments to return their status to non-instrumented.

The following options are available with the delete argument:

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) for sample signal deletion. If the argument all is specified, sample signal deletion applies to each IICE.

-field *fieldName*

Removes the instrumentation from the named field or record for the specified signal (partial instrumentation).

group {*groupNumber* [*groupNumber*]**}** *sigName* [*sigName* ...]
group -show all|*sigName* [*sigName* ...]
group -show_tab all|*sigName* [*sigName* ...]

In the first syntax statement, *groupNumber* is an integer value specifying the assigned multiplexed group number from 1 through 8, and *sigName* is the full hierarchical path name of the instrumented signal to be assigned to that group. Multiple signals can be assigned to a group by separating the signal names with spaces, and signals can be assigned to more than one group by including additional group numbers separated by spaces and enclosed in curly braces.

The following options are available with the group argument:

-show all|*sigName* [*sigName* ...]

Lists the group or groups assigned to *sigName*. If the all argument is included, lists all of the signals that have been assigned to groups and their group numbers.

-show_tab all|*sigName* [*sigName* ...]

Lists the group or groups assigned to *sigName* in tabular format. If the all argument is included, lists all of the signals that have been assigned to groups and their group numbers.

map {*sigName***}** *connectorPin*

Assigns *sigName* to the specified Mictor connector pin location. In the above syntax, *sigName* is the full hierarchical path name to the signal or bus and *MictorPinName* is the concatenation of the Mictor board HapsTrak connector location, the Mictor connector name, and the Mictor pin name separated with periods. For example, 3.M1.D3e is the D3e pin of Mictor connector M1 on the Mictor board installed in HapsTrak connector 3.

Command Example

```
signals add /top/u1/reset_n
signals add -iice IICE_2 -trigger /top/u1/clken
signals add -sample -field iport_mem {/Struc_P_Signed_LDDT_iport}
signals delete -msb 63 -lsb 32 /top/data_in
signals group {2 3} /top/data_in top_data_out
signals group -show_tab all
signals map /beh/blk_xfer_cntrl/req_o 4.M1.D13o
signals map /beh/blk_xfer_inst/beh/{slave_bus[0]} 4.M1.D13o
```

See Also

- breakpoints, on page 30
- clock option – iice, on page 55

source 

Runs a TCL script of commands.

Syntax

source *fileName*

Arguments and Options

FileName contains a script of TCL commands plus commands for the Identify debugger.

Command Example

```
source /home/joe/syn.tcl
```

```
source E:/counter/load.tcl
```

statemachine 

Configures the state machine with the desired behavior.

Syntax

statemachine addtrans -from *state* [**-iice** *iiceID*|**all**] [**-to** *state*]
[**-cond "***equation*|**ti***triggerInID***"**] [**-cntval** *integer*] [**-cnten**] [**-trigger**]

statemachine clear [**-iice** *iiceID*|**all**] **-all**|*state* [*state ...*])

statemachine info [**-iice** *iiceID*|**all**] [**-raw**] **-all**|*state* [*state ...*])

Arguments and Options

addtrans -from *state*

Specifies the state from which the transition is exiting. This option is required to add a transition to the state machine.

addtrans -iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) for state-machine configuration. If the argument all is specified, state-machine configuration applies to each IICE.

addtrans [**-to** *state*]

Specifies the state to which the transition goes. If the -to option is not given, the state defaults to the state given by the -from option, thus creating a transition back to the -from state.

addtrans [**-cond "***equation*|**ti***triggerInID***"**]

Specifies the condition or external trigger under which the transition is to be taken. The default is “true” (that is, the transition is taken regardless of any input data).

The conditions are specified using boolean expressions comprised of variables and operators. The available variables are:

- **c0, ... c***n*: where *n* is the number of trigger conditions instrumented. These variables represent the trigger output of the respective trigger condition.

- **cntnull**: true whenever the counter is equal to '0' (only available if a counter has been instrumented using the -counterwidth option of the iice controller command).
- *iiceID*: this variable is used with cross triggering to define the source IICE units to be included in the equation for the destination IICE trigger.
- **ti***triggerInID*: the ID (0 thru 7) of an external trigger input.

Operators are:

- Negation: `not, !, ~`
- AND operators: `and, &&, &`
- OR operators: `or, ||, |`
- XOR operators: `xor, ^`
- NOR operators: `nor, ~|`
- NAND operators: `nand, ~&`
- XNOR operators: `xnor, ~^`
- Equivalence operators: `==. =`
- Constants: `0, false, 1, true`

Parentheses '(', ')' are recommended whenever the operator precedence is in question. Use the state info command to verify the conditions specified.

addtrans [**-cntval** *integer*]

Specifies that in the case when the transition is taken, the counter must be loaded with the given value. This option is only valid if a counter was instrumented using the iice controller -counterwidth option.

addtrans [**-cnten**]

If this flag is given, the counter is decremented by '1' during this transition. This flag is only valid if a counter was instrumented using the iice controller -counterwidth option.

addtrans [**-trigger**]

If this flag is given, the trigger occurs during this transition.

clear [**-iice** *iiceID*|**all**] **-all**|*state* [*state* ...]

Deletes state transitions.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) for state-machine transition deletion. If the argument all is specified, transition deletion applies to each IICE.

-all|*state* [*state* ...]

Deletes the state transitions from the states given in the argument, or from all states if the argument -all is specified.

info [**-iice** *iiceID*|**all**] [**-raw**] **-all**|*state* [*state* ...]

Prints the current state-machine settings.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) reporting the state-machine settings. If the argument all is specified, the settings for each IICE are reported.

-**all**|*state* [*state* ...]

Reports the settings for the states given in the argument or, if the option -all is specified, for the entire state machine.

-raw

Reports the settings in a machine-processible form.

Command Example

```
statemachine addtrans -from 0 -to 1 -cntval 9
statemachine addtrans -from 0 -cond "(c1 | c2)" -trigger
statemachine addtrans -from 1 -cond "c1 && c2" -cnten
statemachine addtrans -from 2 -cond "c2 && cntnull" -trigger
statemachine addtrans -from 0 -cond "IICE_1 and IICE_2" -trigger
statemachine clear 1
```

```
statemachine info -all
```

See Also

- iice controller -counterwidth option – iice, on page 55
- iice controller -triggerconditions option – iice, on page 55
- iice controller -crosstrigger option – iice, on page 55

Note: The order in which the transitions are added is important. In each state, the first transition condition that matches the current data, is taken. There may be other transitions later in the list that also match the current data, but they are ignored.

stop 

Activates/deactivates an HDL source-level breakpoint that has been added by the Identify instrumentor. All activated breakpoints are used to form the trigger condition of the IICE. Only breakpoints that have been instrumented using the breakpoints add command can be activated. One or more breakpoints can be activated/deactivated at the same time. A breakpoint name consists of two components:

- The fully hierarchical path of the HDL design unit that denotes the underlying control statement of the breakpoint.
- The HDL source code location given by the file name and the line number of the breakpoint.

The combination of these two components ensures that each breakpoint has a unique name.

Syntax

stop disable [*options*] *breakpointName* [*breakpointName* ...]

stop enable [*options*] *breakpointName* [*breakpointName* ...]

stop info [**-raw**] *breakpointName*

Arguments and Options

disable [*options*] *breakpointName* [*breakpointName* ...]

Deactivates one or more HDL source-level breakpoints.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the breakpoint to be disabled. If the argument all is specified, disabling the breakpoint applies to each IICE.

-condition all|**{***conditionList***}**

Specifies a list of trigger conditions in which to disable the breakpoint or breakpoints. If only one trigger condition exists in the current design, this option can be omitted, otherwise it is required. The identifier all disables the breakpoint(s) from all trigger conditions.

enable [*options*] *breakpointName* [*breakpointName* ...]

Activates one or more HDL source-level breakpoints.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the breakpoint to be enabled. If the argument all is specified, enabling the breakpoint applies to each IICE.

-condition all|**{***conditionList***}**

Specifies a list of trigger conditions in which to enable the breakpoint(s). If only one trigger condition exists in the current design, this option can be omitted, otherwise it is required. The identifier all enables the breakpoints from all trigger conditions.

info [**-raw**] *breakpointName* [*breakpointName* ...]

Displays information about the settings for the given HDL breakpoint. The -raw option provides the information in a machine-readable format.

Command Example

```
stop disable -condition 1 /top/u1/case_128/cpu.vhd:29
```

See also

- breakpoints, on page 30
- iice, on page 55

transcript 

Controls recording of all typed commands into a transcript file.

Syntax

transcript [*fileName*]

transcript [**off**]

transcript [**on**]

Arguments and Options

transcript *fileName*

Saves all typed commands to the file specified by *fileName*.

transcript off

Commands system to stop recording commands.

transcript on

Commands system to start recording all typed commands and to store them to the default transcript file. The default file is syn_di.scr for the Identify instrumentor and syn_hhd.scr for the Identify debugger.

Default Value

By default, command recording is off.

Command Example

```
transcript on
```

See Also

- log, on page 70

watch 

Activates/deactivates a watchpoint as a trigger condition for the IICE. A watchpoint triggers when the sample value of the watched signal matches the watch value. Only signals that have been instrumented using the signals add command can be used for watchpoints.

Syntax

watch disable [*options*] *signalName* [*signalName* ...]
watch disable [*options*] **-msb** *value* [**-lsb** *value*] *signalName*

watch enable [*options*] *signalName* **{***value***}**|**{***valueFrom***}** **{***valueTo***}**
watch enable [*options*] **-msb** *value* [**-lsb** *value*]
signalName **{***value***}**|**{***valueFrom***}** **{***valueTo***}**

watch info [**-raw**] *signalName*

watch radix [*options*] *signalName* [**default**|**binary**|**octal**|**integer**|**unsigned**|**hex**]

watch width *signalName*

Arguments and Options

Deactivates an HDL source-level watchpoint. One or more watchpoints can be deactivated at the same time.

disable [*options*] *signalName* [*signalName* ...]
disable [*options*] **-msb** *value* [**-lsb** *value*] *signalName*

SigName is the full hierarchical path name of the signal. In the first syntax statement, more than one signal can be deactivated for sampling or triggering by including additional signal names separated by spaces. In the second syntax statement, the -msb and -lsb arguments specify a bit or bit range of a bus. Note that when specifying partial buses:

- Use the -msb argument (without an -lsb argument) to specify a single bit
- Observe the index order of the bus. For example, when defining a partial bus range for bus [63:0] (or “63 downto 0”), the MSB value specified must be greater than the LSB value. Similarly, for bus [0:63] (or “0 upto 63”), the MSB value specified must be less that the LSB value.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the watchpoint to be disabled. If the argument all is specified, disabling the watchpoint applies to each IICE.

-condition all|**{***conditionList***}**

Specifies a list of trigger conditions in which to disable the watchpoint. If only one trigger condition exists in the current design, then this option can be omitted, otherwise it is required. The identifier all can be used to disable the watchpoint from all trigger conditions.

enable [*options*] *signalName* **{***value***}**|**{***valueFrom***}** **{***valueTo***}**
enable [*options*] **-msb** *value* [**-lsb** *value*] *signalName* **{***value***}**|**{***valueFrom***}** **{***valueTo***}**

When only *value* is specified for *signalName*, gives the watchpoint signal an exact value that the system watches for, and enables that watchpoint for triggering. When *valueFrom*/*valueTo* is specified, gives the watchpoint signal two values that the system watches for, and enables the watchpoint for triggering. These formats allow you to specify a trigger condition on the value transition of a signal. In the second syntax statement, the -msb and -lsb arguments specify a bit or bit range of a bus. Note that when specifying partial buses:

- Use the -msb argument (without an -lsb argument) to specify a single bit
- Observe the index order of the bus. For example, when defining a partial bus range for bus [63:0] (or “63 downto 0”), the MSB value specified must be greater than the LSB value. Similarly, for bus [0:63] (or “0 upto 63”), the MSB value specified must be less that the LSB value.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the watchpoint to be enabled. If the argument all is specified, enabling the watchpoint applies to each IICE.

-condition all|**{***conditionList***}**

Specifies a list of trigger conditions in which to enable the one or more watchpoints. If only one trigger condition exists in the current design, this option can be omitted, otherwise it is required. The identifier all enables the watchpoints from all trigger conditions.

info [**-raw**] *breakpointName* [*breakpointName* ...]

Displays information about the settings for the given HDL watchpoint. The -raw option provides the information in a machine readable format.

radix [*options*] *signalName* [**default**|**binary**|**octal**|**integer**|**unsigned**|**hex**]

Displays or changes the radix of the specified watchpoint signal for the sampled data. Specifying default resets the radix to its initial intended value. Note that the radix value is maintained in the "activation database" and that this information will be lost if you fail to save or reload your activation. Also, the radix set on a signal is local to the Identify debugger and is not propagated to any of the waveform viewers. Note that with partial buses, the radix applies to the entire bus.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing *signalName*. If the argument all is specified, the radix is reported/changed for each IICE.

width *signalName*

Reports the width of a vectored (bused) signal. Note that with partial buses, the width reported always applies to the entire bus.

Command Example

```
watch enable /top/u2/current_state {red}

watch enable -condition {1 2} /top/u1/count {"0X01"} {"0010"}

watch radix current_state hex

watch enable /top/bx {4'b0010}

watch enable -msb 3 -lsb 0 /top/u2/data_sel {4'h0}

watch enable -condition all /top/done {1'b0} {1'b1}
```

See also

- signals, on page 79
- controller -triggerconditions option – iice, on page 55

waveform 

Configures the waveform preferences and launches the desired waveform viewer once the Identify debugger has uploaded data from the instrumented design.

Syntax

waveform custom [*userProcedure*]

waveform period [*period_in_ns*]

waveform show [*options*]

waveform viewer [*options*] **aldec**|**verdi**|**dve**|**gtkwave**|**modelsim**|**custom**

Arguments and Options

custom [*userProcedure*]

Sets/gets user-defined TCL procedure (*userProcedure*) that is used to launch a custom waveform viewer. This procedure must be defined in the TCL window or sourced through a startup script prior to launching the waveform viewer. The default value is custom_waveform. This procedure is called by waveform show with the following five arguments:

- *lang* – the language the design is written in -- Verilog or VHDL
- *toplevel* – the name of the top-level module or entity
- *firstcycle* – the cycle number of the first cycle
- *sampledepth* – the total number of samples
- *period* – the period for the waveform display independent of the design speed

period [*period_in_ns*]

Sets/gets the period with which to display the debug data in the waveform viewer. Since the debugger has no information about the timing of the user design, this setting is merely used for customizing the display.

show

Launches the waveform viewer that is currently selected with the current set of sample data.

-iice *iiceID*|**all**

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the sample data to be displayed. If the argument all is specified, the sample data is displayed for each IICE.

-showequiv

Includes all equivalent signals in the sample data.

viewer [*options*] **aldec**|**verdi**|**dve**|**gtkwave**|**modelsim**|**custom**

Selects the user preference for the waveform viewer. The selection custom causes the waveform show command to call the procedure specified by the waveform custom command.

-list

Lists the available waveform viewer choices. An asterisk preceding the waveform viewer name in the list indicates the currently selected viewer.

Command Example

```
waveform viewer -list

waveform show -showequiv
```

write instrumentation 

Writes the instrumented design files to the project directory.

Syntax

write instrumentation [-pretty|-save_orig_src|-encrypt_orig_src]

Options

-pretty

Instrument HDL with human-readable formatting

-save_orig_src

Create an orig_sources directory in the project directory and copy the user's original sources into this directory.

-encrypt_orig_src

Encrypt the original sources in the orig_sources directory. The encryption is based on a password which must previously be set with the encryption set_passwd command. Attempting to use this flag without a valid password set results in an error. Note that the -encrypt_orig_src flag implies and overrides the -save_orig_src flag. When neither flag is set, no orig_sources directory is created in the project directory.

Command Example

```
write instrumentation -encrypt_orig_scr

write instrumentation -save_orig_src

write instrumentation
```

See Also

- encryption, on page 43

write samples 

Writes the sample data of each specified signal.

Syntax

write samples [*options*] *signalName* [*signalName* ...]
write samples [*options*] **-msb** *value* [**-lsb** *value*] *signalName*

In the above syntax statements, *sigName* is the full hierarchical path name of the signal. In the first syntax statement, sample data can be written for more than one signal by including additional signal names separated by spaces. In the second syntax statement, the -msb and -lsb arguments specify a bit or bit range of a bus. Note that when specifying partial buses:

- Use the -msb argument (without an -lsb argument) to specify a single bit
- Observe the index order of the bus. For example, when defining a partial bus range for bus [63:0] (or "63 downto 0"), the MSB value specified must be greater than the LSB value. Similarly, for bus [0:63] (or "0 upto 63"), the MSB value specified must be less that the LSB value.

Arguments and Options

-iice *iiceID*

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the sample data for the specified signal.

-cycle {*cycleFirst cycleLast***}**

Specifies the range of sample data displayed. You can view the data at different points of the trigger event. Enter a negative cycle value to view data sampled before the triggered event. Enter a positive cycle value to view data samples after the trigger event. Enter a zero cycle value to view data sampled during the trigger event.

-file *fileName*

Writes the sample data to a specified output file. If no file is given, the data is displayed on the screen.

-force

Overwrite *fileName* if it exists

-raw

Return machine-readable samples. For each signal specified, the command returns a Tcl list formatted as shown:

{**{***signalName cycleFirst cycleLast***} {***sampleValuesList***}**}

Command Example

```
write samples -file D:/tmp/samples.txt /top/u1/count
write samples -cycle { -10 10 } /top/u2/current_state
write samples -msb 31 -lsb 0 /top/u3/data_outA
```

write vcd 

Writes the sample data of each specified signal to a Verilog Change Dump (vcd) format.

Syntax

write vcd [*options*] *fileName*

Arguments and Options

-iice *iiceID*

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the sample data for the specified signal.

-comment *commentText*

Inserts a text comment into a file. Use curly braces '{}' to group a multi-word comment.

-gtkwave

Creates a GTKWave control file for the VCD output file.

-showequiv

Includes the sample data for all equivalent signals.

fileName

Writes the sample data to the specified output file.

Command Example

```
write vcd -gtkwave D:/tmp/b.vcd
```

write vhdlmodel 

Creates a VHDL model from sample data. This command is not supported in Verilog-based designs or in mixed-language designs when the top-level is a Verilog module.

Syntax

write vhdlmodel [*options*] *fileName*

Arguments and Options

-iice *iiceID*

Used when more than one IICE is defined to specify the active IICE (*iiceID*) containing the sample data for the VHDL model.

-showequiv

Includes the sample data for all equivalent signals.

fileName

Writes the VHDL model to a specified output file.

Command Example

```
write vhdlmodel D:/tmp/b.vhd
```

Index

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

W