Verification Continuum™

HAPS® Prototyping Debugging Environment Reference

April 2020

solvnetplus.synopsys.com

Copyright Notice and Proprietary Information

Free and Open-Source Licensing Notices

Destination Control Statement

Disclaimer

Trademarks

Third-Party Links

Synopsys, Inc.
690 East Middlefield Road
Mountain View, CA 94043
www.synopsys.com

April 2020

Contents

CHAPTER 1

Introduction

The tool set consists of an instrumentor and a 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 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 an alphabetical listing of all the commands that the tool set accepts in the next chapter. The remainder of this chapter describes the conventions used within this manual to convey command usage.

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. These symbols are:

This convention...	Organizes this information...
I	Any command that is used in the instrumentor only. This symbol is located following any instrumentor command in the command listing chapter.
D	Any command that is used in the debugger only. This symbol is located following any debugger command in the command listing chapter.
I D	Any command that is used in both the instrumentor and debugger tools. This symbol is located following the commands that have applications in both tools in the command listing chapter.

Tool Conventions

There are tool concepts you must familiarize yourself with when using the 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 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.

CHAPTER 2

Instrumentor and Debugger Commands

All commands are listed alphabetically in this chapter. Each command contains syntax, argument return values, default values, and examples. The table below is linked to the individual command descriptions.

activation	breakpoints	cd	chain
clear	clock_group	com	compile
device	encryption	exit	fpga
haps	help	hierarchy	idcode
iice	instrumentation	jtag_server	launch_verdi
licenseinfo	log	project	pwd
readback	remote_trigger	run	searchpath
setsys	show	signals	source
statemachine	stop	transcript	umrbus_server
userprefs	userprefs	watch	waveform
write fsdb	write instrumentation	write samples	write vcd
write vhdlmodel			

The commands are divided into several specific categories. These categories separate the commands in terms of which tool (instrumentor or debugger) utilizes the command. These symbols are:

I	Command available only in the instrumentor
D	Command available only in the debugger
I D	Command available in both the instrumentor and the debugger

activation D

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 *activation-Name* 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 tool 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**] [**-silent**] *breakpointName* [*breakpointName* ...]

breakpoints preconfigure [**-iice** *iiceID*|**all**] **-condition {***integer***}|all -state 0|1**

breakpoints map *breakpointName MictorPinName*

Arguments and Options

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

For the add and delete options, one or more breakpoints can be added or deleted at the same time. The **-iice** argument specifies a particular IICE (*iiceID*) where the breakpoint is to be added or deleted, when more than one IICE is defined. The **all** argument specifies that the corresponding breakpoint is added to or deleted from each IICE.

The **-silent** argument suppresses the display of resource estimates when a breakpoint is added. By default, adding a breakpoint automatically updates the estimate of IICE resources displayed in the console window. The resources represent the total number of bits marked as breakpoints (breakpoints command) or watchpoints (sample only, trigger only, or sample and trigger through the signals command).

For information on specifying *breakpointName,* see *Breakpoint Names,* on page 18.

breakpoints preconfigure [**-iice** *iiceID*|**all**] **-condition {***integer***}|all -state 0|1**

The preconfigure option identifies specific breakpoint conditions to set in advance or to *preset* into the debugger. Pre-configure triggers arm your IICE to run immediately after configuration in order to trap error conditions that can occur during the first several cycles. The -condition argument identifies the trigger conditions to enable and accepts a Tcl list of conditions (e.g., {1 2 5}) or all for all conditions, and the -state argument identifies the trigger transition. For more information on pre-configured triggers, see Capturing Startup Errors with Pre-Armed Triggers, on page 551 in the *ProtoCompiler User Guide*.

The **-iice** argument specifies a particular IICE (*iiceID*) where the breakpoint is to be configured, when more than one IICE is defined. The **all** argument specifies that the corresponding breakpoint is configured in each IICE.

breakpoints map *breakpointName MictorPinName*

The **map** option is used exclusively with real-time debugging, to assign a breakpoint to a Mictor connector pin. *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.

Breakpoint Names

Breakpoint names consist of two components that together ensure that each breakpoint has a unique name:

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

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

breakpoints preconfigure -iice IICE4 -condition {2,12,22}
   -state 0
```

See Also

- *stop*, on page 87

cd ◆I◆D

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 67

chain D

Sets up and manipulates the UMRBus® or JTAG chain of devices. Because more than one device can be connected in a chain, the commands allows you to setup the chain representation in the 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 27
- *com*, on page 24

clear I D

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

clock_group

Specifies the clock signal for a clock domain or clears all clock groups.

Syntax

clock_group add -name *groupName* **-freq** *frequency* [**-edge positive|negative**]
[*signalName*] [**-iice** *iiceID*]

clock_group clear

Arguments and Options

-name *groupName*

Specifies the name for a clock group.

-freq *frequency*

Specifies the frequency for the clock signals in the clock group. The *frequency* value is in kHz unless specified otherwise. The sample clock frequency must be at least equal to the fastest user sample clock domain.

-edge positive|negative

Specifies the active edge of the clock (positive or negative) when an IICE clock group is specified. The default edge is rising (positive).

signalName

An optional clock signal name for the clock group.

-iice *iiceID*

Used when more than one IICE is defined to specify which IICE (*iiceID*) to connect to the clock group. The current IICE unit is used by default.

Command Example

```
clock_group add -name grpA -freq 100 grpAmstr
```

See Also

- *signals add -clkgroup*

com D

Sets up and manipulates communication settings between the 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 xilinxparallel, xilinxusb, xilinxauto, and demo. The umrbus selection indicates that the UMRBus is to be used as the communication interface between the hardware and the host machine running the debugger.

cableoptions *option* [*value*]

Specifies or reports cable-specific option settings:

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.

See Also

- *chain,* on page 20

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 75

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 stop_on_signal_not_found [0|1]

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

device xilinxinsertbufg

device xilinxjtagaddr1

device xilinxjtagaddr2

device xilinxusesrl16

device capimbaseaddr [*address*]

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 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 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 debugger (the JTAG port is not used). To change or report the UMRBus CAPIM address, see *capimbaseaddr [baseAddress]* , on page 30.

prepare_incremental

Sets up the 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 instrumentor to NOT explicitly define the JTAG clock as requiring global clock resources. The skewfree option is disabled (0) by default.

stop_on_signal_not_found [0|1]

When the device stop_on_signal_not_found 1 command is specified in the .idc or Instrumentor TCL file, it will force the instrumentor to error out when the signals to be instrumented in the idc file (RTL or SRS) are not found in the compiled design. By default, stop_on_signal_not_found will be set to 0 and the instrumentor will flag a warning message if the signals to be instrumented are not found in the compiled design.

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

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

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

Due to the timing requirements of the JTAG clock, the 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 boundary scan cells for the built-in JTAG controller. The default is user3. This option is available only in the instrumentor.

xilinxjtagaddr2 [user1|user2|user3|user4]

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

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

capimbaseaddr [*baseAddress*]

Specifies or reports the base address of a CAPIM inserted for UMRBus communication. In a multi-FPGA debugging environment, different FPGAs on the same UMRBus require unique CAPIM addresses when using post partitioned instrumentation (by default, FPGAs on the same board share the same CAPIM address which would disrupt communications with the debugger). The capinbaseaddr argument is used to assign different CAPIMs when two or more FPGAs have independent instrumentations. The CAPIM base address value ranges up to 63 with a default base address of 57. This value is decremented with each CAPIM added (user CAPIM addresses are incremented beginning with 1).

Command Example

```
device estimate -iice IICE -noresources

device jtagport builtin

device skewfree 1
```

See Also

- *chain*, on page 20
- *com*, on page 24

encryption I D

Sets the current password to use before encrypting or decrypting a file. In the instrumentor, this command sets the password to be used when writing out an encrypted file with the write instrumentation command. In the 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 instrumentor or 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 99
- *project*, on page 66

exit ◆I ◆D

Exits the program and closes the window.

Syntax

exit

Arguments and Options

None

Command Example

```
exit
```

fpga ◆I

Adds an FPGA for distributed instrumentation.

Syntax

fpga add [**-iice** *iiceID*] **-type** *fpgaType*

-iice *iiceID* **| all**

Used when more than one IICE is defined to specify which IICE (*iiceID*) to use for distributed instrumentation. If the argument all is specified, the FPGA type applies to each IICE unit.

-type *fpgaType*

The type of FPGA to use for distributed instrumentation.

haps D

Queries the hardware to generate the requisite Tcl file for board generation and performs the verification tests. For additional information, see Chapter 4, *Board Bring-up* in the *HAPS® ProtoCompiler Debugger User Guide*.

Syntax

haps
  boardstatus [*boardID*]
  settings {*setting value* [*setting value* **...**]**}**
  prog *binFile devID*
  setvcc *voltage* [*region*]
  setclk *clockName frequency*
  clear
  tssgen *tclFile*
  hstdm_report [**capim "***device bus address***"|ALLFPGAS** [verbose]]
    [**poll** *interval*] [**device** *index*]
  clock_check
  con_check [*connectivityFileName*] [*logFileName*]
  con_speed *speed* {**fast|sweep**} [*connectivityFileName*] [***logFileName***]
  umr_check *fpgaID*

Arguments and Options

boardstatus [*boardID*]

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

settings {*setting value* [*setting value* **...**]**}**

Specifies the HAPS port (PORT_NAME), device (DEV_ID), and bus (BUS_NUM) settings. With no arguments, reports the current settings. The curly braces enclosing the arguments are required.

prog *binFile devID*

Programs the FPGA identified by *devID* with the specified bin file. The *devID* value ranges from 1 to 32 with 1 corresponding to the first FPGA on the board.

setvcc *voltage* [*region*]

Sets the I/O voltage for the board regions. The voltage value and region are selected from the corresponding drop-down menus and differ with the board/system selected. Multiple regions can be selected using the Ctrl key. If *region* is omitted, all regions are set to *voltage*.

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

Clears the entire board/system configuration including the FPGA configuration, voltage, and clock settings.

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.

tssgen *tclFile*

Queries the HAPS system and generates a Tcl file that describes the hardware setup. Use this file to check the hardware setup.

hstdm_report [**capim "***device bus address***"**|**ALLFPGAS** [verbose]]
[**poll** *interval*] [**device** *index*]

Queries and reports the current training status of the HSTDM blocks running in the user design.

capim "*device bus address***"** – query the training report file from the FPGA identified by the CAPIM. The capim argument can be repeated to address other CAPIMs.

ALLFPGAS – query the training report files from all FPGAs.

verbose – include additional content in the training report.

poll *interval* – poll the FPGAs for the specified interval in seconds; the poll argument is not supported from the GUI.

device *index* – list training status for the FPGAs on the indexed board device.

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.

con_check [*connectivityFileName*] [*logFileName*] – verifies that the cabling between HapsTrak connectors is consistent with the defined connectivity file.

> *connectivityFileName* – specifies the Tcl script that describes the connections between HapsTrak connectors in your current system set-up. The default script in the current working directory is named connectivity.tcl.
>
> *logFileName* – specifies the name of the log file. The default file in the current working directory is named hapstest.log. Note that if you use a non-default log file, you must also explicitly specify the connectivity file even if you intend to use the default; for example:

```
haps run con_check {} ./logfiles/hapstest2.log
```

con_speed *speed* {**fast|sweep**} [*connectivityFileName*] [*logFileName*] – verifies the connectivity between HapsTrak connectors as well as the hypothetical speed at which HSTDM can run. In the syntax:

> *speed* – the raw data transfer speed in Mbps; the acceptable values are 840, 960, 1080, or ALL (the ALL selection scans a range of speeds to determine the highest rate possible).
>
> **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.
>
> *connectivityFileName* – specifies the Tcl script that describes the connections between HapsTrak connectors in your current system set-up. The default script in the current working directory is named connectivity.tcl.
>
> *logFileName* – specifies the name of the log file. The default file in the current working directory is named hapstest.log. If you use a non-default log file, you must also explicitly specify the connectivity file even if you intend to use the default.

umr_check *fpgaID* – 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 first board.

Command Example

```
haps run umr_check 2 180

haps setclk GCLK1 150MHz

haps setvcc 1v8

haps help con_speed

haps settings {PORT_NAME emu:1 DEV_ID 4}

haps hstdm_report capim "8 0 58" verbose
```

help I D

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.

hierarchy I D

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 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 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 the debugger).

– **enabled** – limits search to enabled watchpoints, breakpoints, and other enabled HDL design units (available only in the debugger).

– **instrumented** – limits search to the sampling clock, and watchpoints and breakpoints that have been marked as instrumented (available only in the instrumentor).

– **not-instrumented** – limits search to watchpoints and breakpoints that have not been instrumented (available only in the instrumentor).

– **sample_only** – limits search to sample-only watchpoints (available only in the instrumentor).

– **trigger_only** – limits search to trigger-only watchpoints (available only in the 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 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 77

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 00100000001110001000100010001000 device_type 8

idcode add -quiet 00100000001110001000100010001000 "device type" 8

idcode clear
```

See Also

- *device*, on page 27
- *chain*, on page 20

iice

Duplicates the functionality of the IICE Configuration dialog box.

Syntax

**iice clock|controller|current|delete|info|list|mgb|new|preconfigure|rename|
sampler**

iice clock 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 (the clock selected as the sample clock is defined as the fast 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 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 Arguments and Options

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 -counterval option (applies only to the 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 -counterval.

-counterval *unsignedInteger*

Sets a value for the complex counter and loads that value into the complex counter (applies only to the debugger). The value must fit into the complex counter width as defined in the instrumentor. The default value for the complex counter is 16.

-counterwidth *integer*

Instruments a versatile counter of variable size for triggering (applies only to the instrumentor). An integer parameter in the range between 1 and 32 specifies a new counter width, and 0 suppresses the creation of a state-machine counter (a -counterwidth value of 0 is not recognized for complex-counter triggering). All other values are invalid. The default value for the counter width is 16 for both complex and state-machine counter triggering.

-triggerconditions *integer*

Used when instrumenting a design for state-machine triggering (applies only to the 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 the instrumentor). This option specifies the maximum number of states instrumented in the state machine. The range is 2 to 10. 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 the 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 the instrumentor). Specifying a value between 1 and 8 creates a corresponding number of input ports.

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 the 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 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 Arguments and Options

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 Arguments and Options

iice delete *iiceID*

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

iice info Arguments and Options

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 Arguments and Options

iice list

Lists the IDs (names) of each defined IICE.

iice mgb Arguments and Options

iice mgb -iice {*iiceID***}** [*options*]

Configures the multi-FPGA DTD module (*iiceID* identifies the IICE unit and is required). The following options are supported:

-add_hub {*hubID***}**

Sets the debug hub name for the associated IICE.

-clear_hubs

Removes all debug hubs.

-connect {*userFPGAconn***} {***hubConn***}**

Describes the MGB connectivity between a user FPGA on the HAPS system and the debug hub. This option is mutually exclusive of the -mgb_reserved option. Example:

```
iice mgb -iice {idxA} -connect {mb_1.uA.MGB2.J2A}
   {hub_idxA.MGB1.J2A}
```

-mgb_internal all|*hubID* **{***mgbLinks***} ...**]

Sets up the on-board MGB link locations reserved for debugging. The all option sets up the links for all hubs; including the *hubID* argument sets up only the links for the specified hub. This option only applies to the multi-FPGA DTD module buffer type.

-mgb_loopback *hubID* **{***mgbLinks***} ...**

Sets up the MGB link locations that are reserved as loopback links for the specified debug hub. This option only applies to the multi-FPGA DTD module buffer type.

-mgb_link_speed 3|6

Specifies speed of the MGB link. Recognized values are either 3 (SATA card) or 6 (QSFP card). This option only applies to the multi-FPGA DTD module buffer type.

-mgb_max_links 8|16

Limits the number of MGB links to either 8 (SATA card, the default) or 16 (QSFP card). This option only applies to the multi-FPGA DTD module buffer type.

-mgb_muxfactor auto|1|2|4|8

Defines the MGB multiplexing factor. This option only applies to the multi-FPGA DTD module buffer type. Values are 1 for up to 32 signals per link, 2 for up to 64 signals per link, 4 for up to 128 signals per link, and 8 for up to 256 signals per link. Selecting auto (the default) uses the smallest value that can accommodate the number of signals per link.

-mgb_reserved { {*FPGA* **{***mgbLink***} } ...}**

Identifies the MGB link locations reserved for the user design (more than one location can be specified). This option only applies to the multi-FPGA DTD module buffer type and is mutually exclusive of the -connect option. Example:

```
iice mgb -iice {link0} -mgb_reserved {{mb1.uA
   {MGB1.J3.X2 MGB1.J3.X3 MGB1.J3.X4 MGB1.J3.X5}}
   {mb1.uB {MGB1.J2.X2 MGB1.J2.X3 MGB1.J2.X4 MGB1.J2.X5 }} }
```

-mgb_use_predefined_links 1|2

Reserves only predefined links on the system for debug. This option only applies to the multi-FPGA DTD module buffer type.

-num_ddr3_locations 1|2

Specifies the number of DDR3 memory modules used on the specified debug hub (the default is 1). This option only applies to the multi-FPGA DTD module buffer type.

iice new Arguments and Options

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 instrumentor.

iice preconfigure Arguments and Options

iice preconfigure [*options*]

Identifies specific conditions to set in advance or *preset* into the debugger. Pre-configure triggers arm your IICE to run immediately after configuration in order to trap error conditions that occur during the first several cycles. For more information on pre-configured triggers, see Capturing Startup Errors with Pre-Armed Triggers, on page 551 in the *ProtoCompiler User Guide*.

-iice *iiceID***|all**

Used when more than one IICE is defined to identify the IICE unit; if all is specified, effective for all IICE units.

-countermode events|cycles|pulsewidth|watchdog

Pre-configures the mode for the counter.

-counterval *integer*

Pre-configures the value for the counter.

-crosstriggermode disabled|any|after|all

Pre-configures the cross-triggering mode.

-crosstriggeriice *iiceID*

IICE to be used when crosstriggermode is set to after.

-triggertime early|middle|late

Sets/gets trigger time for next run.

-samplemode normal|qualified_fill|qualified_intr|always_armed

Sets/gets the iice sample mode.

-datacompression 0|1|off|on|false|true

Sets data compression.

-group 1|2|3|4|5|6|7|8

Sets the specified sample group for sampling.

-notification 0|1|off|on|false|true

Disables/enables notification of a trigger event via an LED. If enabled, a top-level port identify_trig_notification_LED_n_*iiceID* (active low) will be added. It is then the user's responsibility to set up an appropriate location constraint for this port.

iice rename Arguments and Options

iice rename *iiceID*

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

iice sampler Arguments and Options

iice sampler [*options*]

The following table shows the iice sampler options supported in the instrumentor and debugger:

Instrumentor iice sampler Options

Instrumentor: Supported iice sampler Options

-iice {*iiceID*|**all**} **internal_memory|haps_DTD|haps_DTD2|haps_DTD_builtin haps_DTD_cross_trigger**
-always_armed 0|1
-depth *sampleMemoryDepth*
-qualified_sampling 0|1
-compression 0|1
-ram *options*
-rtd {mictorlocs {*location* [*location* ...]**}|board** *boardType***}**
-dtd_clock *clockName*
-dtd_reset_type {*type***}**
-dtd_reset {*resetSignalNSource***}**
-dtd_reset_bins {*binName***}**
-pipe *integer*

Debugger: Supported iice sampler Options

-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*

-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|haps_DTD|haps_DTD2|haps_DTD_builtin

Specifies the buffer type used to capture the sample data as internal_memory, haps_DTD (synthesis flow), haps_DTD2 (partition flow), or haps_DTD_builtin (HAPS-80 built-in memory).

haps_DTD_cross_trigger

Specifies the buffer type used to capture the sample data as external_memory.

-always_armed 0|1

Enables/disables always-armed sampling. When enabled (1), the 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.

-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), a single sample of all sampled signals is collected each time the trigger condition is true. When a trigger condition occurs, instead of filling the entire buffer, the IICE collects the single sample and then waits for the next trigger to acquire the next sample to allow design operation to be monitored over an extended period of time.

Qualified sampling is impacted by the -samplemode setting; selecting the qualified_fill mode allows single samples to be acquired at each trigger event until the sample buffer is full, and selecting the qualified_intr mode allows samples to continue to be acquired (and possibly overwritting existing samples) until interrupted.

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.

-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.

-ram {*option value***}**

The -ram option applies when the *bufferType* is set to haps_DTD (synthesis flow), haps_DTD2 (partition flow), or haps_DTD_builtin. The -ram options are described below. The -ram option applies only to regular IICE units and is not supported by real-time debug IICE. The GUI equivalents to the -ram options are included on the DTD tab (synthesis flow) or MGB tab (partition flow).

board|debugboard *boardType*

sramlocations {*locations***}**

Used in the synthesis flow to identify the connector location where a daughter card is physically installed. *Locations* is a set of three adjacent HapsTrak 3 connectors on the HAPS system board that align with an SLR region within the FPGA (for example, 1_2_3, 4_5_6, etc.).

numberboardstack *integer*

The number of stacked daughter boards (always 1)

type *integer*

The RAM daughter board type installed on a synthesis-based system according to the following table:

Type Value	Memory Card/Daughter Board
9	ONBOARD_DX/DDR3_SODIMM_2R_HT3 (8GB)
10	DDR3_SODIMM_HT3 (4GB)

clocktype

Specifies either an external HDL signal or an internal GCLK clock source for the partition flow.

-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.

-dtd_clock {*clockName***}**

The -dtd_clock option applies only when the *bufferType* is set to haps_DTD2 for the partition flow to specify the source of the GCLK reference clock in a multi-FPGA system.

-dtd_reset {*resetSignalName***}**

Specifies the name of the reset signal for the gclk reset type. The option applies only when the *bufferType* is set to haps_DTD2 for the partition flow in a multi-FPGA system.

-dtd_reset_bins {*binName***}**

For designs where some but not all the FPGAs contain signals that have been tagged for debug, this option defines the FPGAs (bins) where reset logic must be distributed or replicated because they contain the tagged debug signals for the design. Use -dtd_reset_bins in conjunction with the bin_attribute -locked PCF constraint (used to lock the FPGAs that are not part of the design).

-dtd_reset_type {*type***}**

Specifies the type (origin) of the reset signal in a multi-FPGA configuration. The option applies only when the *bufferType* is set to haps_DTD2 for the partition flow. Recognized reset types are:

- AUTO – identifies the partitioner-assigned source of the generated reset. Refer to the TSS file entry. Optionally, a master FPGA can be identified to limit the source of the generated reset to the indicated device.
- GCLK – clock-defined reset source referencing a GCLK signal. The clock entry format is *boardName*.GCLK*n* (for example, mb1.GLCK3). Refer to the TSS file entry.

Note that when using a GCLK resource for the DTD2 reset, you must define the source of the GCLK clock net as fpga in the TSS file and not as pll.

-pipe *integer*

Adds additional pipeline registers for high-speed sampling of the instrumented signals that cannot meet timing. The specified number of pipeline registers are dynamically inserted on the input signal of every bit to be instrumented; the default is 3.

Debugger iice sampler Options

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

Controls how a detected trigger affects data sampling (applies only to the 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 the 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 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 instrumentor for activation in the debugger. *Integer* is the number of the multiplexed group which ranges from 1 to 8.

Command Examples

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

iice sampler -ram {ramlocations {7_8_9}}

iice sampler -DTD2_reset_type {user}
iice sampler -DTD2_reset {mb2.my_rst_all}
```

instrumentation

Manipulates incremental instrumentations.

Syntax

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

instrumentation current

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

instrumentation list

instrumentation load *name*

instrumentation save

Arguments and Options

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

current

Returns a key value-pair TCL list with information of current instrumentation.

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 the instrumentor).

load *name*

Loads an existing instrumentation into the the instrumentor.

save

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

Command Example

```
instrumentation load instr_2

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

jtag_server D

Configures the JTAG server.

Syntax

jtag_server set -addr {*hostName***|***IP_address***} -port {***serverPort***} -logf {***logFfileName***} -usecs 1|0**

jtag_server get

jtag_server start -standalone 0|1 -cabletype xilinxusb|*otherValidType*

jtag_server stop -forced 0|1

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.

-usecs 1|0

Enables or disables the client-server configuration for the USB-based UMRBus.

get

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

start

Selects the server startup mode.

-standalone 0|1

Selects the server startup mode. If set to 1, the debugger application is closed, and the JTAG server runs in the background.

-cabletype xilinxusb|*otherValidType*

Selects the cable type.

stop

Stops the server.

-forced 0|1

A value of 1 immediately stops all communications.

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

jtag_server start -standalone 1

jtag_server stop
```

launch_verdi

Launches automated verdi debug environment.

-run_dir <value>

Run directory containing the debug data for executing Verdi.

-fsdb <value>

Absolute path of the FSDB file.

-fsdbtop <value>

Design top module name.

-fsdbtype <value>

FSDB type. gsv, pnr, and vcs are supported.

gsv

For FSDB file generated using backannotation ll file(s) in runtime.

pnr

For FSDB file generated using raw ll file(s) from Vivado in runtime.

vcs

For FSDB file generated using VCS.

-gui

Launches Verdi GUI; default is batch.

Syntax

launch_verdi

```
launch_verdi -run_dir debug_data -fsdb top.fsdb -fsdbtop top
-fsdbtype pnr -gui
```

licenseinfo I D

Displays information about the product version and license status.

Syntax

licenseinfo

logicanalyzer D

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 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 I D

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 89

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 I D

Opens existing projects and displays project information.

Syntax

project import *projectFile*

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

project name [**-path**]

Arguments and Options

import *projectFile*

Imports the specified project file.

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

Performs a simple import of a project (`prj`) file by extracting the design files, the device technology, and the design top level. This data is used to create an implementation (applies only to the 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 the 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 31

pwd ◆I◆D

Displays the current working directory.

Syntax

pwd

See Also

- *cd*, on page 19

readback

Initiates a readback operation.

Syntax

readback sample_buffer {**-append**|**-overwrite**}

readback set -iice {*iiceID*|**all**} {**-clear**|**-apply**} **-llFile** *llFilePath* **-fpga** *boardLocation*

readback scan_ccm_capims

readback set_ccm_capim -device *number* **-bus** *number* **-capim** *number*

readback enable_ccm

readback memory -mem {memoryName} [-file fileName] [-format bin|hex]

Arguments and Options

sample_buffer [**-append**|**-overwrite**]

Appends or overwrites samples to the sample buffer when performing GVS (Global State Visibility). The -overwrite option clears the sample buffer prior to execution of the run command; the -append option leaves the contents of the sample buffer intact, effectively appending the samples.

set -iice {*iiceID*|**all**} **-clear -apply -llFile** *llFilePath* **-fpga** *boardLocation*

Sets the path to the logic location (**.ll**) file and defines the FPGA mapping. The individual options are:

-iice {*iiceID*|**all**} – selects the IICE (*iiceID*) or all IICE units for readback

-clear – clears the previous assignments

-apply – loads the specified logic location file and applies the signals to the user design on readback command execution (by default, logic location files are automatically loaded by the run command)

-llFile *llFilePath* – specifies the logic location (**.ll**) file to be loaded

-fpga *boardLocation* – specifies the physical FPGA device; for example, FPGA C on board 1 is specified as either fb1_c or FB1_C.

scan_ccm_capims

Scans all CAPIMs in the system with a clock-control module type and creates a table of eligible CAPIM entries for the set_ccm_capim argument.

set_ccm_capim -device *number* **-bus** *number* **-capim** *number*

Specifies the CAPIM (for the individual argument options, enter values from the table generated by the scan_ccm_capims argument).

enable_ccm

Enables the clock-control module (CCM). Including the enable_ccm argument with the readback command allows the CCM to start generating the controlled clocks. This runtime command must be exercised during the runtime session and only after all of the FPGAs on the target system or systems have been configured.

readback memory

Lists the different RTL memories in the design along with their hierarchical paths in the Tcl Console.

readback memory -mem {memoryName} – To readback content of a specific memory and view the memory data in TCL Console window.

> **[-file fileName]** – To write the readback data to a file. Writing the readback data to a file is optional.
>
> **[-format bin|hex]** – To specify the readback data format. Specifying the readback data format is optional. By default, the data is displayed in hex format.

See Also

- *run*, on page 72
- *Running GSV with Clock Control* in the *HAPS ProtoCompiler User Guide.*

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 44
- triggertime option – *iice*, on page 44

run D

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. In the interactive shell mode, this command does not detect the trigger conditions occurred on the hardware. Hence, use the update TCL command in loop to capture the trigger condition and download the data or use the -wait command to block the trigger.

Syntax

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

run -timeout *integer*

run -wait

run -skip_config

run -readback *integer*

run -stepping *integer*

run -wait_for_trigger 0|1

run -release_clock 0|1

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 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.

-skip_config

Used when a pre-configure trigger is set to ignore the pre-configured trigger and enable sampling through the normal, user-defined trigger. For more information on pre-configured triggers, see Capturing Startup Errors with Pre-Armed Triggers, on page 551 in the *ProtoCompiler User Guide*.

-readback *integer*

Number of readback cycles to perform. For more information on using the Xilinx readback debugging capability, see Using Global State Visibility (GSV), on page 641 in the *ProtoCompiler User Guide*.

-stepping *integer*

Enables the stopped clocks for *integer* cycles.

-wait_for_trigger 0|1

When set to 1, readback waits for the trigger to occur.

-release_clock 0|1

When set to 1, releases the stopped clocks after the readback cycle is complete.

-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 debugger menu bar. There is no stop command in the command shell.

Command Example

```
run -remote_trigger 1336
```

searchpath I D

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 26

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 I D

Displays an HDL source code context on the command line.

Syntax

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

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*.

Command Example

```
show -8 +16 cpu.vhd:29
```

signals

Instructs the instrumentor to create special debug logic for the IICE 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*

signals preconfigure [*options*]

signals add: Arguments and Options

signals add [*options*] *sigName* [*sigName* ...]
signals add [*options*] **-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.

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 every IICE.

-sample

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

-silent

Suppresses the display of resource estimates when a signal is added. By default, adding a signal automatically updates the estimate of IICE resources displayed in the console window. The resources represent the total number of bits marked as watchpoints (sample only, trigger only, or sample and trigger through the signals command) or breakpoints (breakpoints command).

-field *fieldName*

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

-clkgroup *clockGroupName*

Uses the *clockGroupName* for the instrumented signal. Signals without a defined -clockgroup argument are sampled in the fast-clock domain.

-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.

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 alone (without -lsb) 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.

signals delete: Arguments and Options

signals delete *sigName* [*sigName* ...]
signals 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).

signals group: Arguments and Options

signals group {*groupNumber* [*groupNumber*]**}** *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.

signals group -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.

signals group -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.

signals map: Arguments

signals 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.

signals preconfigure: Arguments and Options

signals preconfigure [*options*] *signalName* [**-msb** *integer* **-lsb** *integer*]

The preconfigure option identifies specific signals to set in advance or to *preset* into the debugger. Pre-configure triggers arm your IICE to run immediately after configuration in order to trap error conditions that occur during the first several cycles. For more information on pre-configured triggers, see Capturing Startup Errors with Pre-Armed Triggers, on page 551 in the *ProtoCompiler User Guide*.

The following arguments and options can be used:

-iice *iiceID***|all**

The IICE unit to use or all IICE units.

-condition {*triggerCondition***}|all**

Specifies the trigger conditions to enable. Accepts a Tcl list of conditions (e.g., {1 2 5}) or all for all defined conditions.

-msb *integer*

The msb of a bus slice to instrument.

-lsb *integer*

The lsb of a bus slice to instrument.

Command Example

```
signals add /top/u1/reset_n
signals add -iice IICE2 -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

signals preconfigure -iice IICE4 -condition {90,94,98}
   -state 1
```

See Also

- *breakpoints*, on page 17
- clock option – *iice*, on page 44

source I D

Runs a TCL script of commands.

Syntax

source *fileName*

Arguments and Options

FileName contains a script of TCL commands plus commands for the 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 44
- iice controller -triggerconditions option – *iice,* on page 44
- iice controller -crosstrigger option – *iice,* on page 44

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 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 all breakpoints 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 breakpoints. 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 17
- *iice*, on page 44

transcript I D

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 instrumentor and syn_hhd.scr for the debugger.

Default Value

By default, command recording is off.

Command Example

```
transcript on
```

See Also

- *log*, on page 65

umrbus_server

Configures the UMRBus server.

Syntax

umrbus_server set -addr {*hostName***|***IP_address***} -port {***serverPort***}**
-logf {*logFfileName***} -usecs 1|0**

umrbus_server get

umrbus_server start -standalone 0|1 -cabletype umrbus|*otherValidType*

umrbus_server stop -forced 0|1

Arguments and Options

set

Configures the JTAG server

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

The IP address or the name of the UMRBus server.

-port {*serverPort***}**

The port number over which the client and server communicate.

-logf {*logFfileName***}**

The name of the log file.

-usecs 1|0

Enables or disables the client-server configuration for the UMRBus.

get

Returns the current server host name or IP address, port number, log file name, and client-server configuration.

start

Selects the server startup mode.

-standalone 0|1

Selects the server startup mode. When set to 1, the UMRBus server runs in the background and continues to run after the debugger application is closed.

-cabletype umrbus

Selects the UMRBus cable type.

stop

Stops the server.

-forced 0|1

A value of 1 immediately stops all communications.

Command Example

```
umrbus_server set -addr myhost -port 57015 -logf servercom.log
```

```
umrbus_server get
   INFO: umrbus_server get
   addr 127.0.0.1 port 57015 logf ipc_tcp_xilinx.log usecs 1
```

```
umrbus_server start -standalone 1
```

```
umrbus_server stop
```

userprefs

Defines the instrumentation preferences and installation path for the instrumentor.

Syntax

userprefs option save_orig_src|encrypt_orig_src

userprefs synplify_install [*installPath*]

Arguments and Options

option save_orig_src|encrypt_orig_src

Sets/gets the preferences option for the instrumentor.

synplify_install [*installPath*]

Sets/gets the path for the Synplify installation.

verdi

Imports or instruments signals from the Verdi essential signal database.

Syntax

verdi getsignals *ESDBpath*
verdi instrument

Arguments and Options

ESDBpath is the location where es.esdb++ is installed.

watch D

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 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 78
- controller -triggerconditions option – *iice*, on page 44

waveform

Configures the waveform preferences and launches the desired waveform viewer once the 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 fsdb D

Writes the sample data of each specified signal in FSDB format for analysis and display in Verdi nWave.

Syntax

write fsdb [*options*] *fsdbFilename*

Arguments and Options

-iice *iiceID*

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

-showequiv

Includes the sample data for all equivalent signals.

fsdbFilename

Writes the sample data to the specified fast signal database output file.

Command Example

```
write fsdb D:/tmp/b.fsdb
```

write instrumentation

Writes the instrumented design files to the project directory.

Syntax

write instrumentation *options*

Options

-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.

-idc_loc *directory*

Save Identify constraints to the specified directory.

-idc_only

Save only the idc file.

-cdc_only

Save only the CDC file.

Command Example

```
write instrumentation -encrypt_orig_scr
```

```
write instrumentation -save_orig_src
```

```
write instrumentation -idc_only
```

See Also

- *encryption*, on page 31

write samples D

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 D

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