Tessent® SiliconInsight® User's Manual for Tessent Shell

Software Version 2014.2

June 2014

This manual is part of a fully-indexed Tessent documentation set. To search across all Tessent manuals, click on the "binocular" icon or press Shift-Ctrl-F. Note that this index is not available if you are viewing this PDF in a web browser.

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/

Send Feedback on Documentation: supportnet.mentor.com/doc_feedback_form

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction to Tessent SiliconInsight

This chapter provides an overview of the Tessent SiliconInsight product.

Overview

Tessent SiliconInsight operates within the Tessent Shell environment and provides interactive capabilities for controlling, debugging, and characterizing BIST-related memories, logic, PLLs, and SerDes. It uses the Mentor Graphics Tessent IJTAG implementation of the IEEE P1687 (IJTAG) proposed standard, which provides a unified mechanism for accessing and debugging embedded designs.

The tool uses an IJTAG-based open ATE interface that enables Tessent SiliconInsight to directly interact with embedded IPs. The open interface is called the *Characterization and Debug Package* (CDP). The CDP contains enough data in terms of IJTAG test pattern files, Tcl procedures, and optional side files to allow access and control of any instrument at any design hierarchy level.

Related Topics

Tessent IJTAG Users Manual

Interactive Debug with Tessent SiliconInsight Desktop

Tessent SiliconInsight Desktop provides an automated interactive environment for test bring-up, debug, and characterization of devices containing Tessent BIST capabilities. You can:

- Validate the performance of your silicon prototypes to ensure that they function as expected.
- Quickly modify and apply test patterns for debugging embedded instruments.
- Characterize the functional silicon prototype to define its design specifications.

Related Topics

Interactive Debugging with Tessent SiliconInsight Desktop

Pre-Silicon Simulation with SimDUT

The Tessent SiliconInsight SimDUT tool allows you to validate patterns before you receive the silicon prototype. SimDUT is a software package that simulates the DUT and provides responses as if you are interacting with a real tester. The simulator converts the responses to failures that you can then diagnose with Tessent SiliconInsight Desktop.

Using SimDUT reduces the time required for bring-up-and-debug validation of the silicon prototype, allowing you to begin the characterization process sooner.

SimDUT supports the ModelSim and Questa Advanced Simulator from Mentor Graphics.

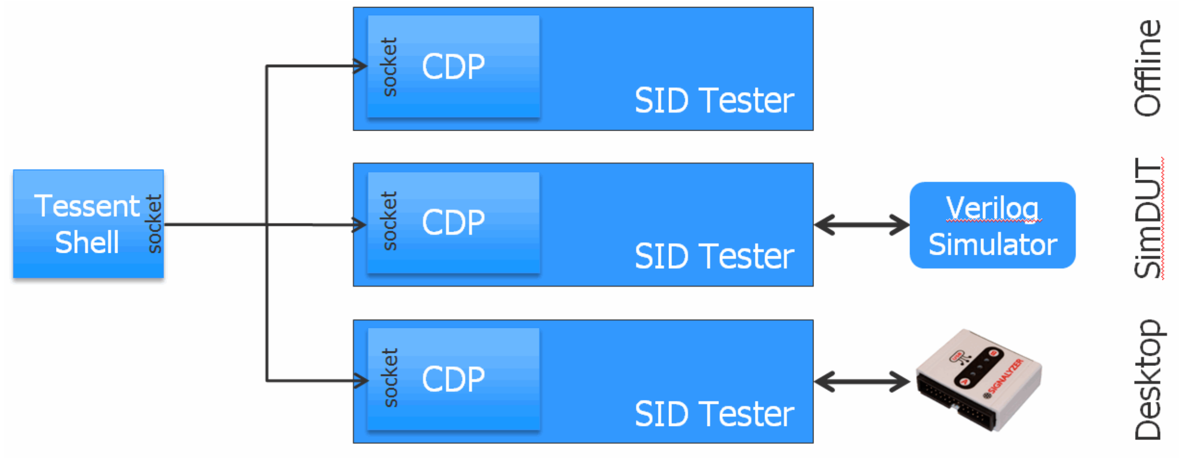
Related Topics

Using Tessent SiliconInsight in a Simulated Environment with SimDUT

Architecture

Figure 1-1shows the Tessent SiliconInsight architecture.

Figure 1-1. Tessent SiliconInsight Architecture



You use the Tessent Shell environment to perform Tessent SiliconInsight operations, such as creating the CDP. Tessent Shell is a Tcl shell environment and design data model that provides a unified Tcl command set with data model interaction.

The Tessent SiliconInsight Desktop tester process (SID tester) communicates with and controls the hardware during execution of a specified test. It acts as an interface between Tessent Shell and the Tessent SiliconInsight adaptor hardware and uses the information stored within the CDP to execute particular patterns for simulation, debug, or diagnostic purposes.

In offline mode, you can use Tessent SiliconInsight to verify that you can run the test patterns without any issues (including syntax errors) before you use them with SimDUT and Tessent SiliconInsight Desktop.

Related Topics

Tessent Shell User's Manual *Tessent Shell Reference Manual*

Chapter 2
Interactive Debugging with Tessent SiliconInsight Desktop

This chapter describes how to use Tessent SiliconInsight Desktop to execute and debug IJTAG test patterns in the following sections:

Note
You can find complete hardware and operating system requirements for Tessent SiliconInsight Desktop in the "Tessent SiliconInsight Configuration" section of the *Managing Mentor Graphics Tessent Software* manual.

Bring-Up, Debug and Characterization

Preparing a design for volume production does not end after tape-out. In fact, the time spent from the arrival of the first silicon prototypes to achieving high-volume production is steadily growing. One of the main tasks during this stage is called *bring-up*, which is the process for validating that a silicon prototype functions as expected. Reducing the bring-up phase is crucial for getting ICs to market within an acceptable budget and timeframe.

During the bring-up-and-debug validation process, your goal is to discover under what conditions the DUT fails. This is an iterative process in which you adjust and re-execute your IJTAG test patterns so that they exercise particular settings in the DUT. By default, SiliconInsight Desktop executes the test patterns in go-nogo (pass-fail) mode.

When you discover a condition in which the DUT always fails—for example, the DUT always fails when it runs at faster than 120 MHz—you must diagnose the problem to isolate the root cause defect for the failure. In diagnosis mode, Tessent SiliconInsight executes an IJTAG test pattern and returns a list of the actual-versus-expected results for the failing cycles. You can use this data to refine and re-execute the test patterns until you isolate the root cause defect that the foundry can then fix. For example, a root cause defect for a DUT that fails at speeds greater than 120 MHz could be a slow flop that was introduced into the design due to a defect in the foundry's manufacturing process.

The end result of the bring-up-and-debug validation process is a functional silicon prototype. Once you have a functional silicon prototype, you perform characterization. Similar to the validation process, you adjust and re-execute your IJTAG test patterns so that you exercise particular conditions. In this case, you want to isolate how the DUT performs under extreme conditions. For example, you may test for the fastest speed at which the DUT runs, the lowest voltage at which it runs, and so on. The characterization results are the performance specifications for the DUT.

Related Topics

Manually Debugging the Failure Results

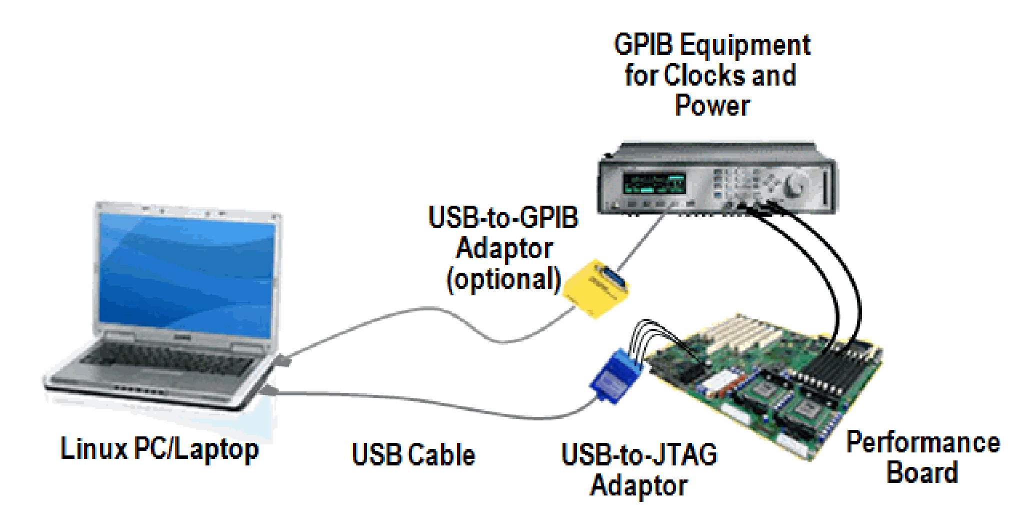
Preparing the Computer for Tessent SiliconInsight Desktop

Before executing an IJTAG test pattern, you must prepare for Tessent SiliconInsight Desktop as described in this section. You only need to perform the following tasks once for each computer.

Tessent SiliconInsight Desktop runs in a Red Hat® Enterprise Linux® environment on any standard Linux PC or laptop. As shown in Figure 2-1, to control the DUT, you attach a USB cable from the PC or laptop to an adaptor, and then connect the adaptor to standard IEEE 1149.1 pins on the performance board.

For more detailed characterization activities, you can also control instruments such as power supplies and clock generators with interface buses implemented according to the General Purpose Interface Bus (GPIB) standard IEEE-488. To do this, connect a USB-to-GPIB adaptor to a GPIB instrument, which you then connect to the performance board.

Figure 2-1. Tessent SiliconInsight Desktop Hardware Configuration


GPIB Equipment for Clocks and Power
USB-to-GPIB Adaptor (optional)
Linux PC/Laptop
USB Cable
USB-to-JTAG Adaptor
Performance Board

When you execute a test from Tessent SiliconInsight Desktop, the tool retrieves information about the BIST resources and design from the Tessent Shell environment, and it instructs the BIST or TAP inside the device to run the test. Tessent SiliconInsight Desktop then interprets the results and displays them on your computer.

Prerequisites

- A Linux PC or laptop configured with one of the following Red Hat Enterprise Linux operating systems:
  - Red Hat Enterprise Linux 5
  - Red Hat Enterprise Linux 6
- USB cable(s). You need two if you are using a GPIB instrument.
- The performance board you will be testing.

Procedure

1. Configure one of the following supported adaptors:
   - Xverve SignalyzerSHA40 and SignalyzerH2/H4 Adaptors
   - Xverve Signalyzer Adaptor
   - Xverve Signalyzer SP
   - Amontec Adaptors
2. Optionally, if you plan to use a GPIB instrument, configure the supported Prologix 6.0 adaptor as described in "Prologix 6.0 Adaptor."
3. Configure your hardware as shown in Figure 2-1.

4. Based on your Red Hat Linux operating system version, configure your computer to run Tessent Silicon Insight Desktop.

   Completing this task allows all users in addition to *root* to access Tessent SiliconInsight Desktop. Perform this task while logged on as *root*. Refer to:

   - Configuring Red Hat Enterprise Linux 5 to Run Tessent SiliconInsight Desktop
   - Configuring Red Hat Enterprise Linux 6 to Run Tessent SiliconInsight Desktop

5. Generate a configuration data file for Tessent SiliconInsight Desktop as described in "The Configuration Data File for Tessent SiliconInsight Desktop."

Results

You can now proceed to "Executing an IJTAG Test Pattern."

Executing an IJTAG Test Pattern

The following procedure takes an IJTAG test pattern that you have saved to SVF, stores it in the CDP, creates a test for the test pattern, and executes the test on the DUT.

Prerequisites

- Computer prepared as described in "Preparing the Computer for Tessent SiliconInsight Desktop."
- A Tessent SiliconInsight Desktop configuration data file. See "The Configuration Data File for Tessent SiliconInsight Desktop."

Procedure

1. From a shell, invoke Tessent Shell using the following syntax:

   ```
   % tessent -shell
   ```

   After invocation, the tool is in setup mode. Refer to the "tessent" command description in the *Tessent Shell Reference Manual* for additional invocation options.

2. Set the tool context to enable Tessent SiliconInsight within Tessent Shell as follows:

   SETUP>set_context patterns -ijtag -silicon_insight

   You do not need the -ijtag switch if you are not generating patterns in the current session.

3. Read in the Tessent SiliconInsight Desktop configuration data file as follows:

   SETUP>read_config_data *my_config_file*.cfg

4. Launch the SID tester process and create a new CDP as follows:

```
SETUP>launch_sid_tester -cdp my_cdp_path -new
```

When using a previously generated CDP, you do not need the -new switch.

Note

SVF-based CDPs are configured with asynchronous clocking. The tool does not provide clock waveforms to the DUT, therefore you must source the appropriate clocks yourself.

5. Generate your IJTAG test pattern and save it as an annotated SVF test pattern with the write_patterns command. Refer to “Writing PDL, Pattern, and Test Bench Files” in the *Tessent IJTAG User’s Manual*.

6. Add the SVF pattern file to the CDP and create a test for that pattern file.

   ```
   SETUP>add_cdp_test test1 -pattern my_pattern_path.svf
   ```

   The test manipulates and executes test pattern files. Currently, you can only specify one test pattern file per test.

   To view your tests, specify the get_cdp_test_list command. To delete a test, specify the delete_cdp_test command.

7. Optionally, use a GPIB instrument to change settings such as the power supply. You must use instrument-specific commands as shown in the following examples.

   In the following example, the GPIB power supply sets the pin voltage levels to 3.9 volts. The power supply was configured to listen at address 19.

   ```
   SETUP>execute_gpib_command “SetVoltage 3.9V” -address 19
   ```

   The following example sets the voltage for an Agilent Power Supply (connected to GPIB address 19) to 1.03 volts.

   ```
   SETUP>execute_gpib_command \
       "SOURCE:VOLTAGE:LEVEL:IMMEDIATE:AMPLITUDE 1.03" -address 19
   ```

   The following example sets the low voltage level for the clock pulse (0) for a Stanford Research System (SRS) CG635 clock generator (connected to GPIB address 6) to 1.5 volts. It also sets the high voltage for the clock pulse (1) to 2.7 volts. Thus, the clock pulse will swing between 1.5 volts to 2.7 volts.

   ```
   SETUP>execute_gpib_command "QOUT 0,1" -address 19
   SETUP>execute_gpib_command "QOUT 1,2.7" -address 19
   ```

   The following example sets the frequency for the same clock generator (SRS CG635 at GPIB address 6) to 25MHZ.

   ```
   SETUP>execute_gpib_command "FREQ 25000000" -address 19
   ```

8. Execute the test. For example:

   ```
   SETUP>execute_cdp_test test1 -collect_data_type_list variable
   ```

This command executes the test named "test1" on the Desktop. It collects the failures during test pattern execution and maps the failures into variable cycles as defined in the annotated SVF pattern file.

If you do not specify any switch, the tool executes the test and returns the pass/fail status for the specified test.

The execute_cdp_test command also stores the data as a config data object that you can introspect using the report_config_data command.

9. Shut down the SID tester when you are done.

    SETUP>shutdown_sid_tester

Results

After executing a given test, Tessent SiliconInsight Desktop generates a tabulated report. The following example shows the results for the test pattern associated with the test named "test1."

Example 2-1. Failures Results for a Test Named "test1"

```
//  command: execute_cdp_test test1 -collect_data_type_list variable
//  Test 'test1' failed.
//  Variable failures and unmapped failures of pattern
// 'DLV2_1S0_ETMemory_Full.svf' :
//
//  PatternSet  Variable                           Pin   Expected
//                                                         Actual
//  ----------  ------------------------------------  ---   --------
//   pattern_1  MyBlock1.MyBlock1Instrument.go(1)    tdo         b1
//                                                               b0
//  ----------  ------------------------------------  ---   --------
//   pattern_1  MyBlock1.MyBlock1Instrument.done(1) tdo         b1
//                                                               b0
//  ----------  ------------------------------------  ---   --------
//
//  Command ID  Bit Offsets
//  ----------  -----------
//           4            0
//  ----------  -----------
```

Related Topics

Sample Dofile to Execute an IJTAG Pattern   Prologix 6.0 Adaptor

Manually Debugging the Failure Results

This section describes the iterative process you can use to manually debug failure results. In this process, you want to resolve the failure results so that the instrument returns the expected values as defined in your procedural description language (PDL) file. You do this by adjusting the parameters in the PDL file and re-executing the test, which may take many iterations.

Refer to the *Tessent IJTAG Users Manual* for details about PDL and pattern retargeting.

Prerequisites

- You have executed an IJTAG test pattern as described in "Executing an IJTAG Test Pattern" and have a failure report that lists the variable failures. These results are shown in Example 2-1.
- You have the PDL file. Suppose the results in Example 2-1 pertain to the following PDL file, instrument.pdl. This file contains one procedure named run_testa with one argument, mode, whose default value is "red."

```
iProcsForModule instrument

iProc run_testa {{mode red}} {
  iNote "-- Start of run_testa called from ICL instance
[get_icl_scope] -- "

  iNote "Verify that ClockPort clk is started"
  iClock clk
  set cycles(red) 50
  set cycles(blue) 90
  set cycles(green) 70

  iNote "Check that go is Fail and done is No at the beginning of the
test"
  iRead go Fail
  iRead done No
  iApply

  iNote "Set mode to $mode"
  iWrite mode $mode
  if {$mode == "red"} {
    iWrite p1 0b0
    iWrite p1[2] 0b1
  }
  iApply

  iNote "Start the test"
  iWrite enable Yes
  iApply

  iRunLoop $cycles($mode) -tck

  iNote "Check that done is Yes and go is Pass at the end of the
test"
  iRead done Yes
  iRead go Pass
  iRead count
  iApply
}
```

Procedure

1. Investigate why the tool reports an actual 0 for the go and done bits. There are many possible reason for failure, including:

    - Incorrect PDL and ICL definitions.
    - Incorrect clock period was specified.
    - Incorrect configuration data file specifications.
    - Simulation was not performed to correct failures in the design.

    Assume that your configuration and clock specifications are correct, and also that your design passed simulation. You suspect that the PDL definition in instrument.pdl is incorrect. Specifically, because the done bit has failed, you suspect that the silicon requires more cycles to complete execution than what is specified in the PDL file.

2. Examine the PDL file, instrument.pdl.

    As shown in "Sample Dofile to Execute an IJTAG Pattern," suppose you called for run_testa to run in default (red) mode.

    ```
    # Call an iProc registered against the ICL module to compose the pattern
    iCall MyBlock1.MyBlock1Instrument.run_testa red
    ```

    In the PDL file you see that the specified clock is defined to run for 50 cycles in red mode.

    ```
    iClock clk
      set cycles(red) 50
      set cycles(blue) 90
      set cycles(green) 70
    ```

3. Adjust the PDL file. To test your theory about the clock cycles, you decide to increase the clock cycle to 60, as follows:

    ```
    iClock clk
      set cycles(red) 60
      set cycles(blue) 90
      set cycles(green) 70
    ```

4. Regenerate the test pattern, create a new test with the add_cdp_test command, and then execute the IJTAG pattern again. Check the results. The test still fails.

```
//  command: execute_cdp_test test1 -collect_data_type_list variable
//  Test 'test1' failed.
//  Variable failures and unmapped failures of pattern
// 'DLV2_1S0_ETMemory_Full.svf' :
//
//  PatternSet  Variable                            Pin   Expected
//                                                          Actual
//  ----------  ----------------------------------  ---   --------
//   pset_1  MyBlock1.MyBlock1Instrument.go(1)      tdo         b1
//                                                              b0
//  ----------  ----------------------------------  ---   --------
```

```
//   pset_1  MyBlock1.MyBlock1Instrument.done(1)    tdo          b1
//                                                               b0
//  ----------  -----------------------------------  ---  --------
//
//  Command ID  Bit Offsets
//  ----------  -----------
//           4            0
//  ----------  -----------
```

5. Adjust the PDL file again by increasing the red clock cycles to 70.

```
iProc run_testa {{mode red}} {
iClock clk
 set cycles(red) 70
 set cycles(blue) 90
 set cycles(green) 70
```

6. Execute the IJTAG pattern again and check the results. This time test1 passes.

```
execute_cdp_test test1 -collect_data_type_list variable
//  Test 'test1' passed.
//  command: system echo -n "Time after execution of test1:";system date
Time after execution of test1:Thu May 8 12:17:04 PDT 2013
```

Related Topics

Sample Dofile to Execute an IJTAG Pattern

Supported Adaptors

Tessent SiliconInsight Desktop supports the following adaptors as described in this section:

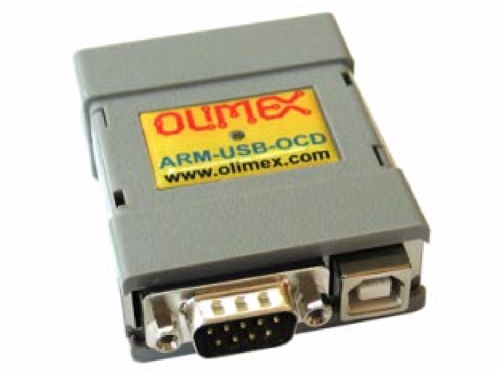
- Olimex ARM-USB-OCD and ARM-USB-OCD-H Adaptors
- Tin Can Tools Flyswatter2 Adaptor
- Xverve SignalyzerSHA40 and SignalyzerH2/H4 Adaptors
- Xverve Signalyzer Adaptor
- Xverve Signalyzer SP
- Amontec Adaptors
- Prologix 6.0 Adaptor

For adaptors, the pin map specification that you include in the configuration data file depends on the adaptor you choose.

Olimex ARM-USB-OCD and ARM-USB-OCD-H Adaptors

Mentor Graphics supports two adaptors manufactured by Olimex Ltd.—the ARM-USB-OCD and ARM-USB-OCD-H adaptors as shown in Figure 2-2.

Figure 2-2. Olimex ARM-USB-OCD and OCD-H Adaptors



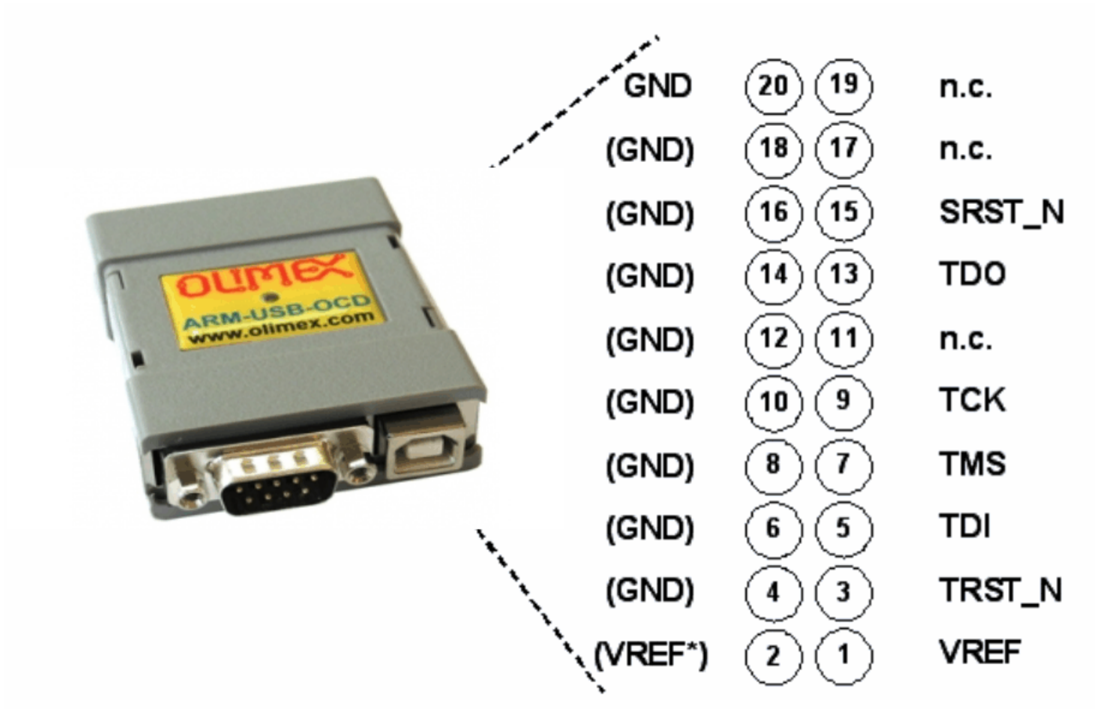
Both adaptors have the same pinout which is provided in Figure 2-3. The only adaptor pins that you must connect to the performance board signals are listed below in Table 2-1.

Table 2-1. Olimex Adaptor Connections

Pin Name	Pin Number	Description
TDI	5	Test Data Input (to DUT)
TMS	7	Test Mode Select (to DUT)
TCK	9	Test Clock (to DUT)
TDO	13	Test Data Output (from DUT)
VREF	1	Voltage Reference
GND	20	Ground

Note
Ensure that all ground pins (4, 6, 8, 10, 12, 14 16, 18, 20) are grounded for good signal fidelity.

Figure 2-3. Olimex Adaptor Pinout



Note

The Adaptor TRST_N (Test Reset) pin is not used. Tessent SiliconInsight Desktop performs a test reset using the 1149.1 standard's reset sequence. The DUT's TRST pin must be pulled low on the board to ensure that the device can operate in normal functional mode when the adaptor is not being used.

The adaptor's SRST_N pin is not used and need not be connected to the DUT.

For detailed information about these adaptors, refer to the Olimex Ltd. website at the following URL:

http://www.olimex.com

The devices are available through several distributors world-wide:

https://www.olimex.com/Distributors/

Tin Can Tools Flyswatter2 Adaptor

Mentor Graphics supports the Tin Can Tools Flyswatter2 adaptor as shown in Figure 2-4.

Figure 2-4. Tin Can Tools Flyswatter2 Adaptor

The pin connections and pinout for the Flyswatter2 adaptor are the same as that for the Olimex ARM-USB-OCD and ARM-USB-OCD-H adaptors. Refer to Table 2-1 and Figure 2-3 for details.

For detailed information about this adaptor, refer to the Tin Can Tools website at the following URL:

http://www.tincantools.com

Xverve SignalyzerSHA40 and SignalyzerH2/H4 Adaptors

The SignalyzerSHA40 and SignalyzerH2/H4 adaptors (Figure 2-5) are produced by Xverve Technologies, Inc. Signalyzer H2 and Signalyzer H4 are no longer available. They have been replaced by the Signalyzer SHA40.

Figure 2-5. SignalyzerSHA40 and SignalyzerH2/H4 Adaptors

SignalyzerSHA40 and SignalyzerH4 support up to 32 I/O pins divided into 4 channels of 8 pins while SignalyzerH2 supports up to 16 I/O pins divided into 2 channels of 8 pins. Since these adaptors are based on the USB 2.0 “high speed” protocol, they are at least 5 times faster than plain Signalyzer which is based on “full speed” version of USB 2.0.

For the adaptors’ operating parameters, refer to the Xverve Technologies website at the following URL:

`http://www.xverve.com`

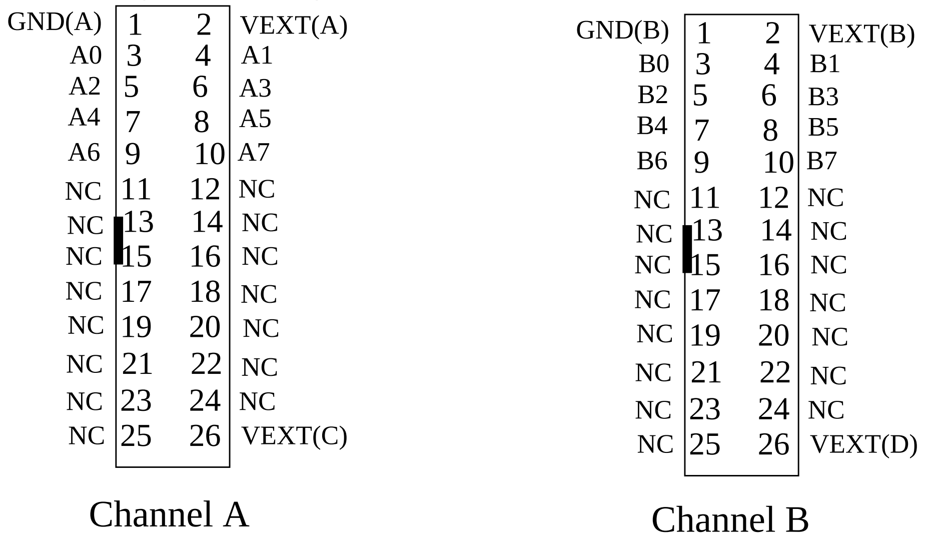
Caution

For the SignalyzerSHA40 and SignalyzerH2/H4 adaptors, Pin 2 (VEXT) and Pin 26 (VEXT) are 5.0V DC supply pins from the USB port of the computer. These pins, in contrast to older Signalyzer models, are not for VREF input and could damage the Signalyzer device if you use them as such.

SignalyzerH2 Pin Map

Figure 2-6 shows the SignalyzerH2 channel pin map. Tessent SiliconInsight Desktop can only simultaneously drive the pins on the same channel. In the SignalyzerH2, these channels are A and B.

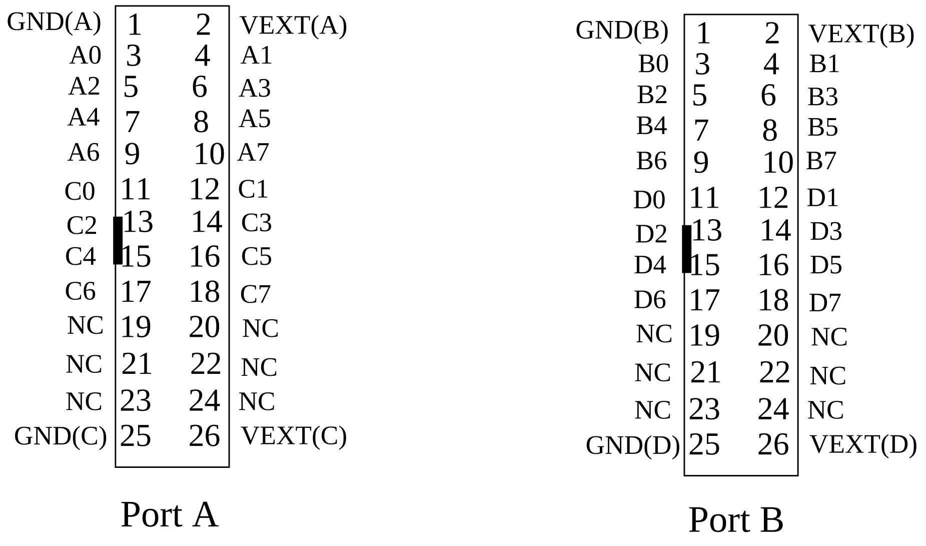
Figure 2-6. SignalyzerH2 Channels Pin Map



SignalyzerSHA40 and SignalyzerH4 Pin Map

Figure 2-7 shows the Signalyzer SHA40 and SignalyzerH4 channel pin map. Tessent SiliconInsight Desktop can only simultaneously drive the pins on the same channel. In the Signalyzer SHA40 and SignalyzerH4 adaptors, these channels are A, B, C, and D.

Figure 2-7. Signalyzer SHA40 and SignalyzerH4 Channels Pin Map

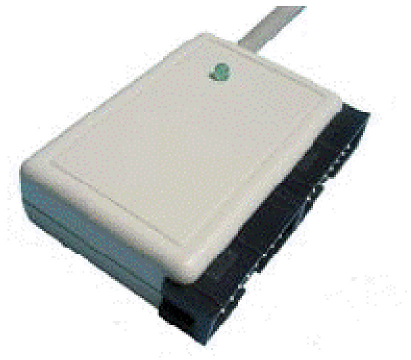


Xverve Signalyzer Adaptor

The Xverve Signalyzer is no longer available. It has been replaced by SignalyzerSHA4. The following section is intended for existing Xverve Signalyzer users.

The Xverve Signalyzer shown in Figure 2-8 is produced by Xverve Technologies.

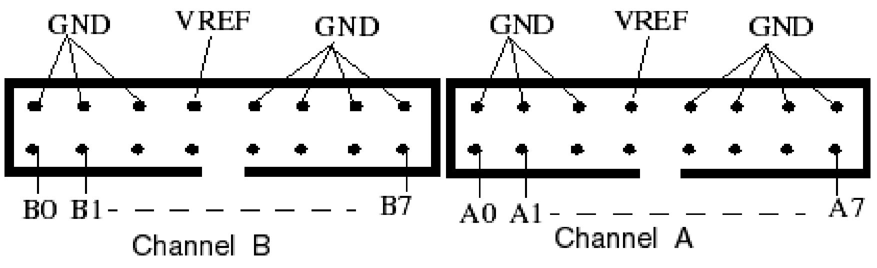
Figure 2-8. Xverve Signalyzer



This Signalyzer supports up to 16 I/O pins divided into two channels, each supporting 8 I/O pins. Only one channel pins might be burst in a single pattern.

Figure 2-9 is displays the Xverve Signalyzer channel pinmap.

Figure 2-9. Signalyzer Channels Pinmap



For example, while Channel *A* might connect to 4 or 5 TAP pins (depending on whether TRST is contacted) and 3 or 4 other dynamic/static pins, Channel B might be used to control other user pins that are not used by the LVDB. Additionally, the Channel B pins might be used to toggle a user sequence to allow access for the Mentor Graphics TAP controller, which will then be accessed through Channel A.

Note
For the best signal fidelity, ensure that all ground pins are grounded.

Xverve Signalyzer SP

The Signalyzer SP has 64 general purpose I/O pins. All pins may be toggled at the same time. Each pin may be independently assigned to be input or output. The Signalyzer SP adaptor is produced by Xverve Technologies.

Signalyzer SP Pinout for Tessent SiliconInsight Desktop

Note
For a complete pinout of the Signalyzer SP pins, see the manufacturer's website at www.xverve.com.

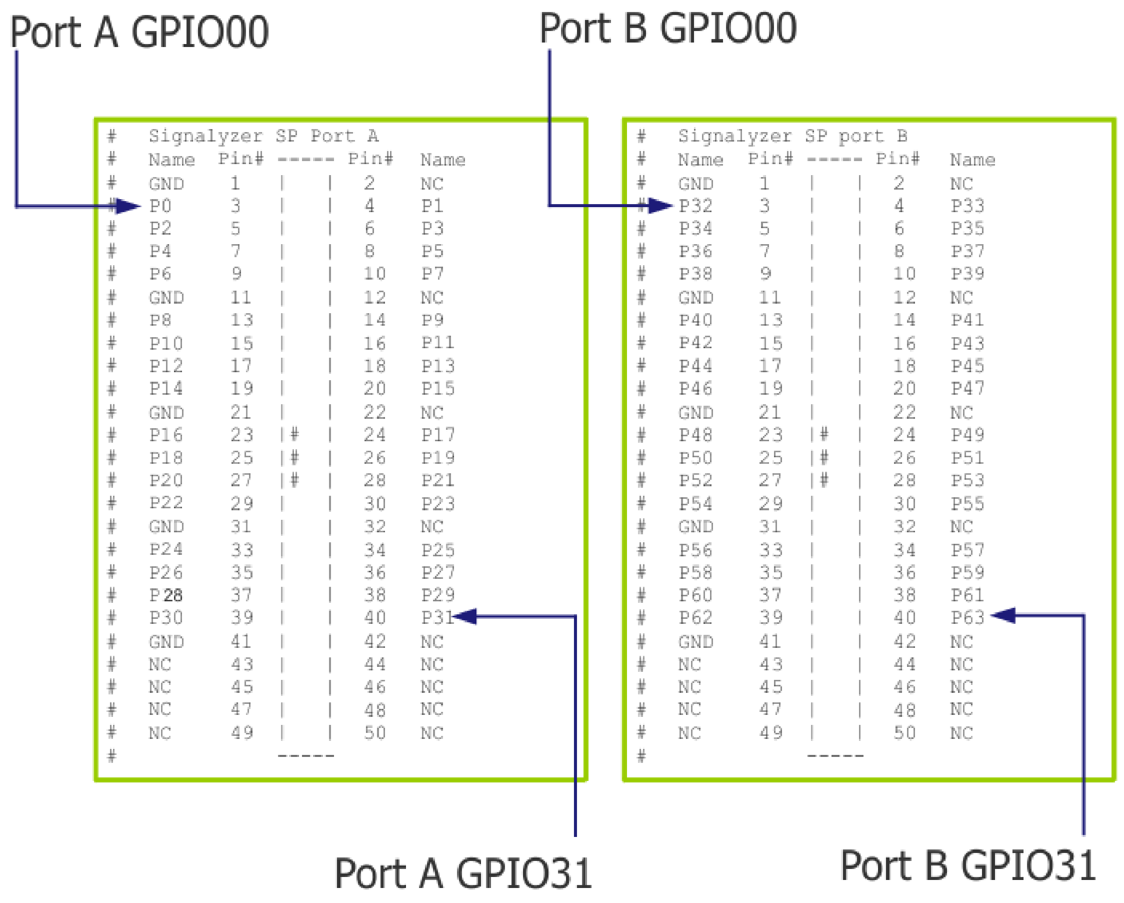
For Tessent SiliconInsight Desktop, the following pin-out guidelines apply:

- You can configure the Signalyzer SP hardware in two independent or one expanded GPIO port. Tessent SiliconInsight Desktop only uses the Signalyzer SP as one expanded 64-pin port. You can set voltage levels for each of the two ports independently.
- You only need to connect one of the Signalyzer SP GND pins on each Signalyzer SP port to ground (1, 11, 21, 31, 41, 49). These pins are connected together internally inside the Signalyzer SP.
- Pin 2 is a Signalyzer SP output that you can use to validate the I/O level, which is controlled by the invocation switch. Pin 2 is otherwise not used by Tessent SiliconInsight Desktop.
- Pins 12, 22, 32, and 42 are unused by Tessent SiliconInsight Desktop.

To support Signalyzer SP, the Tessent SiliconInsight software tree contains a hardware definition called *default.hwdef.signalyzerSP*. This file describes the names of the hardware channels that are specified in your pin map file for multi-site testing. See "Creating the Tessent SiliconInsight Pin Map File for Signalyzer SP" in the *Tessent SiliconInsight User's Manual for the LV Flow* manual for complete information.

Figure 2-10 shows the Signalyzer SP pinout for Tessent SiliconInsight Desktop as described in the *default.hwdef.signalyzerSP* file.

Figure 2-10. Hardware Definition File Signalyzer SP Pinout



```
#  Signalyzer SP Port A
#  Name  Pin# ----- Pin#  Name
#  GND    1   |    |  2   NC
#  P0     3   |    |  4   P1
#  P2     5   |    |  6   P3
#  P4     7   |    |  8   P5
#  P6     9   |    |  10  P7
#  GND    11  |    |  12  NC
#  P8     13  |    |  14  P9
#  P10    15  |    |  16  P11
#  P12    17  |    |  18  P13
#  P14    19  |    |  20  P15
#  GND    21  |    |  22  NC
#  P16    23  |#   |  24  P17
#  P18    25  |#   |  26  P19
#  P20    27  |#   |  28  P21
#  P22    29  |    |  30  P23
#  GND    31  |    |  32  NC
#  P24    33  |    |  34  P25
#  P26    35  |    |  36  P27
#  P28    37  |    |  38  P29
#  P30    39  |    |  40  P31
#  GND    41  |    |  42  NC
#  NC     43  |    |  44  NC
#  NC     45  |    |  46  NC
#  NC     47  |    |  48  NC
#  NC     49  |    |  50  NC
#             -----
```

```
#  Signalyzer SP port B
#  Name  Pin# ----- Pin#  Name
#  GND    1   |    |  2   NC
#  P32    3   |    |  4   P33
#  P34    5   |    |  6   P35
#  P36    7   |    |  8   P37
#  P38    9   |    |  10  P39
#  GND    11  |    |  12  NC
#  P40    13  |    |  14  P41
#  P42    15  |    |  16  P43
#  P44    17  |    |  18  P45
#  P46    19  |    |  20  P47
#  GND    21  |    |  22  NC
#  P48    23  |#   |  24  P49
#  P50    25  |#   |  26  P51
#  P52    27  |#   |  28  P53
#  P54    29  |    |  30  P55
#  GND    31  |    |  32  NC
#  P56    33  |    |  34  P57
#  P58    35  |    |  36  P59
#  P60    37  |    |  38  P61
#  P62    39  |    |  40  P63
#  GND    41  |    |  42  NC
#  NC     43  |    |  44  NC
#  NC     45  |    |  46  NC
#  NC     47  |    |  48  NC
#  NC     49  |    |  50  NC
#             -----
```

This file is located in the Tessent SiliconInsight software tree at the following directory path:

```
Tessent_software_tree/lib/tools/etas_usb
```

For the adaptor's operating parameters, refer to the Xverve Technologies website at the following URL:

http://www.xverve.com

Debugging Signalyzer SP Voltage Level Issues

You should measure the GPIO pin voltages after you have connected the Signalyzer SP to the DUT. If the measured voltage for any GPIO pins is below the preset voltage, then the Signalyzer SP setting may not be correct.

All GPIO pins are referenced to two power pins: portA GPIO pins are referenced to pin2 (VA) and portB pins are referenced to pin22 (VB). Both of the power pins have a reference voltage defined by the user and are usually between 1.2v and 3.3v.

Prerequisites

- Signalyzer SP connected to the DUT.

Procedure

1. Generate a configuration data file that contains the parameters for the SignalyzerSP adaptor. In the following example, the pio_levels_a parameter sets up reference voltage pin2, and the pio_levels_b parameter sets up reference voltage pin22. They can have different voltages.

```
Tester( my_sig_sp ) {
    operation_mode : online;
    Adapter {
      signalyzerSP {
        pio_levels_a : 3.30V;         //portA pin reference voltage
        pio_levels_b : 2.50V;         //portB pin reference voltage
        serial_no : "ttty:0xxxfff";
        Pinmap {
          P0 : jtag[0];
          P1 : jtag[1];
          P2 : jtag[2];
          P3 : jtag[3];
        }
      }
    }
  }
```

   See "The Configuration Data File for Tessent SiliconInsight Desktop" for details about how to create the configuration data file.

2. Invoke Tessent SiliconInsight and launch the SID tester process as described in "Executing an IJTAG Test Pattern."

3. For any GPIO portA pin problem, start measuring the voltage level at pin2 of the Port A connector. (Similarly for GPIO portB pins, measure pin 2 of the Port B connector.) Ensure that the power pins are at the requested level as specified by the pio_levels_a and pio_levels_b parameters. If not, repeat steps 1 and 2 with Signalyzer SP disconnected from the target device(s). If the problem persists, contact the Signalyzer SP manufacturer.

4. Assume you are debugging issues for portA pins. If pin2 has the proper voltage level as specified by the pio_levels_a parameter, then disconnect all the other GPIO pins, including the portB GPIO pins, from the DUT except for one GPIO portA pin. Measure the voltage to make sure it is 1.8v. If it works fine, then connect more GPIO portA pins to the DUT and measure the voltage.

5. If a pin has a partial voltage, do the following:

   a. Measure the voltage between the problematic GPIO pin and other GPIO pins to verify that there are no shortages.

b. Check whether the DUT board is configured correctly, most especially if there is an interposer board placed on top of the DUT board.

c. Check for any pull-up or pull-downs for the associated pin on the DUT board.

d. Chect whether the design itself has a pull-up or a pull-down that connects to the problematic GPIO pin.

6. If all the GPIO pins having consistent partial voltage, reset the DUT board and the interposer board, if any. If the problems persist, try a different Signalyzer SP, and again, if the problem persists, contact the Signalyzer SP manufacturer.

Results

The measured voltage for all the GPIO pins should be below the preset voltage.

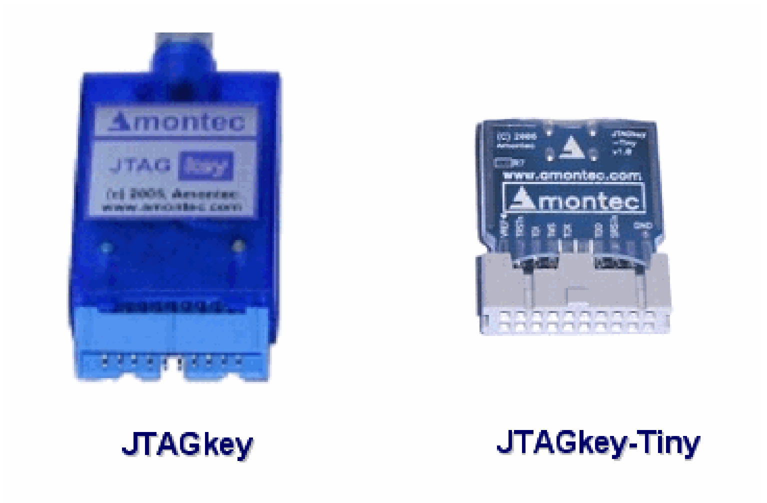
Amontec Adaptors

The Amontec adaptors are no longer available. The following section is intended for existing users. The Olimex and Tin Can Tools adapters described in "Olimex ARM-USB-OCD and ARM-USB-OCD-H Adaptors" and "Tin Can Tools Flyswatter2 Adaptor" offer comparable capabilities.

Mentor Graphics currently supports two USB-to-JTAG adaptors manufactured by Amontec—the JTAGkey and JTAGkey-Tiny adaptors that are shown in Figure 2-11. The JTAGkey adaptor has a few more features than the JTAGkey-Tiny, but these are not used by Tessent SiliconInsight Desktop. Therefore, either adaptor can be used.

Any number of adaptors can be purchased directly from Amontec at ***www.amontec.com***.

Figure 2-11. Amontec JTAG Adaptors



The pin connections and pinout for the Amontec adaptors are the same as those for the Olimex ARM-USB-OCD and ARM-USB-OCD-H adaptors. Refer to Table 2-1 and Figure 2-3 for details.

Prologix 6.0 Adaptor

The Tessent SiliconInsight Desktop GPIB option enables integration and usage of any GPIB-enabled instrument in a plug-and-play manner. Examples of these instruments include clock generators, power supplies, and parametric measurement units (PMUs).

For GPIB integration, Tessent SiliconInsight Desktop currently supports the Prologix 6.0 adaptor manufactured by Prologix as shown in Figure 2-12.

Figure 2-12. Prologix 6.0 Adaptor

You can purchase this adaptor from Prologix at the following URL:

`www.prologix.biz`

Using standard GPIB cables, you can daisy chain multiple GPIB instruments together. For example, you can connect 20 power supplies to Tessent SiliconInsight Desktop in serial.

Note
When you use multiple GPIB instruments through the GPIB bus, a "wait" statement (normally in Tcl scripts) is required so that Tessent SiliconInsight Desktop can correctly gain control of the GPIB instruments.

Configuring the Red Hat Enterprise Linux Operating Systems

This section describes how to configure the following Red Hat Enterprise Linux operating systems to run Tessent SiliconInsight Desktop:

- Configuring Red Hat Enterprise Linux 5 to Run Tessent SiliconInsight Desktop
- Configuring Red Hat Enterprise Linux 6 to Run Tessent SiliconInsight Desktop

Configuring Red Hat Enterprise Linux 5 to Run Tessent SiliconInsight Desktop

To allow access of Tessent SiliconInsight Desktop for all users in addition to *root*, perform the following operations when logged on as *root*.

Prerequisites

- A Linux laptop or PC configured with the Red Hat Enterprise Linux 5 operating system.
- Your hardware configured as shown in Figure 2-1.

Procedure

1. Copy the file <*Tessent Shell Install Dir*>*/share/SiliconInsight/ATE/USB/Drivers/10-sid.rules* to the */etc/udev/rules.d* directory.
2. Run *udevcontrol reload_rules.*

Note
If you are using GPIB equipment, you do not need to perform extra setup steps.

Results

You can now generate your Tessent SiliconInsight Desktop configuration data file as described in "The Configuration Data File for Tessent SiliconInsight Desktop."

Configuring Red Hat Enterprise Linux 6 to Run Tessent SiliconInsight Desktop

To allow access of Tessent SiliconInsight Desktop for all users in addition to *root*, perform the following operations when logged on as *root*.

Prerequisites

- A Linux laptop or PC configured with the Red Hat Enterprise Linux 6 operating system.
- Your hardware configured as shown in Figure 2-1.

Procedure

1. Copy the file <*Tessent Shell Install Dir*>*/share/SiliconInsight/ATE/USB/Drivers/10-sid.rules* to the */etc/udev/rules.d* directory.
2. Run *udevadm control –reload-rules.*

Note
If you are using GPIB equipment, you do not need to perform extra setup steps.

Results

You can now generate your Tessent SiliconInsight Desktop configuration data file as described in "The Configuration Data File for Tessent SiliconInsight Desktop."

The Configuration Data File for Tessent SiliconInsight Desktop

The configuration data file for Tessent SiliconInsight Desktop contains Tessent SiliconInsight-specific commands that define the pin map, testers, adaptor types, and other data that are required to prepare Tessent SiliconInsight Desktop to communicate with the DUT.

Figure 2-13shows the configuration data file structure for Tessent SiliconInsight Destop.

Note
Device design pins that are not specified in the configuration data file are assumed to not be connected to adaptor pins.

Figure 2-13. Configuration Data File Structure for Tessent SiliconInsight Desktop

```
SidSetupSpecification {
   selected_tester : <nameOfSelectedTester>; // specify tester name
                               // that corresponds with adapter type
   Protocol {
      SvfTapPins { // specify design pins that correspond to SVF TAP ports
         TCK : <designPinName>;
         TDI : <designPinName>;
         TDO : <designPinName>;
         TMS : <designPinName>;
         TRST : <designPinName>; // use "-" if it does not exist
      }
   }
   Tester( <testerName> ) { // you can specify multiple testers
      port : <auto | 0..65535>; // a port number for inter-process \
         // communication, where 0 is equivalent to auto
      operation_mode : < online | offline >; // default is online
   Adapter {
      {olimex_arm_usb_ocd | Flyswatter2 | JtagKey | Signalyzer | \
       SignalyzerH2 | SignalyzerSHA40 | SignalyzerH4 | SignalyzerSP}{
      // the following pio_levels_a and pio_levels_b are only required
      // and valid for SignalyzerSP
      pio_levels_a : { 1.20V | 1.25V | 1.30V | 1.35V | 1.40V | 1.45V
                     | 1.50V | 1.60V | 1.65V | 1.70V | 1.80V | 1.90V
                     | 2.50V | 2.80V | 3.00V | 3.30V };
      pio_levels_b : { 1.20V | 1.25V | 1.30V | 1.35V | 1.40V | 1.45V
```

```
                         | 1.50V | 1.60V | 1.65V | 1.70V | 1.80V | 1.90V
                         | 2.50V | 2.80V | 3.00V | 3.30V };
            Pinmap {
               <adapterPinName> : <designPinName>;
               ......
                  }
               }
            }
         }
      }
```

You can specify multiple testers within one configuration data file. However, you can only specify one of the testers as the selected_tester. The selected_tester must be the name of the tester that has your chosen adaptor type. The selected_tester launches Tessent SiliconInsight Desktop.

By default, the SID tester automatically picks an unused port number for inter-process communication.

Note

Preferably, let the SID tester choose an unused port number. If you want to specify a port number, ensure that no other applications are using the port and choose a port number greater than 50000 to avoid port conflict.

Table 2-2 lists the valid adaptor pin names for the supported adaptor types.

Table 2-2. Valid Adaptor Pin Names

Adaptor	Valid Pin Names
Olimex	TCK, TDI, TDO, TMS
Flyswatter2	TCK, TDI, TDO, TMS
JtagKey	TCK, TDI, TDO, TMS
Signalyzer	A0, A1, A2, A3, A4, A5, A6, A7, B0, B1, B2, B3, B4, B5, B6, B7
SignalyzerH2	A0, A1, A2, A3, A4, A5, A6, A7, B0, B1, B2, B3, B4, B5, B6, B7
SignalyzerSHA40 and SignalyzerH4	A0, A1, A2, A3, A4, A5, A6, A7, C0, C1, C2, C3, C4, C5, C6, C7, B0, B1, B2, B3, B4, B5, B6, B7, D0, D1, D2, D3, D4, D5, D6, D7
SignalyzerSP	P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20, P21, P22, P23, P24, P25, P26, P27, P28, P29, P30, P31, P32, P33, P34, P35, P36, P37, P38, P39, P40, P41, P42, P43, P44, P45, P46, P47, P48, P49, P50, P51, P52, P53, P54, P55, P56, P57, P58, P59, P60, P61, P62, P63

Example Configuration Data File

The following example illustrates a configuration data file for SignalyzerSHA40, SignalyzerH4 and SignalyzerH2 adaptors.

Example 2-2. Configuration Data File for SignalyzerSHA40, SignalyzerH4 and SignalyzerH2 Adaptors

```
SidSetupSpecification {
   selected_tester : my_sig_sha40; // specify the tester to be used to
                                  // launch_sid_tester
   Protocol {
      SvfTapPins { // specify design pins for the adaptor
         TMS  : jtag[0];
         TDI  : jtag[1];
         TDO  : jtag[2];
         TCK  : jtag[3];
         TRST : -;
      }
   }
   Tester( my_sig_sha40 ) { // one test
      port: auto;
      operation_mode: online; // default is online but can be offline too
      Adapter {
         SignalyzerSHA40 {
            Pinmap {
               A0 : jtag[0];
               A1 : jtag[1];
               A2 : jtag[2];
               A3 : jtag[3];
               A4 : D[0];
               A5 : D[1];
               A6 : A[0];
               A7 : A[1];
            }
         }
      }
   }
   Tester( my_sig_h4 ) { // one test
      port: auto;
      operation_mode: online; // default is online but can be offline too
      Adapter {
         SignalyzerH4 {
            Pinmap {
               A0 : jtag[0];
               A1 : jtag[1];
               A2 : jtag[2];
               A3 : jtag[3];
               A4 : D[0];
               A5 : D[1];
               A6 : A[0];
               A7 : A[1];
            }
         }
      }
   }
```

```
        Tester( my_sig_h2 ) { // another tester
            port: auto;
            operation_mode: online; // default is online but can be offline too
            Adapter {
                SignalyzerH2 {
                    Pinmap {
                        A0 : jtag[0];
                        A1 : jtag[1];
                        A2 : jtag[2];
                        A3 : jtag[3];
                        A4 : D[0];
                        A5 : D[1];
                        A6 : A[0];
                        A7 : A[1];
                    }
                }
            }
        }
    }
```

Sample Dofile to Execute an IJTAG Pattern

The following dofile illustrates executing an IJTAG pattern.

```
##########################################
#### 1. Set context to pattern and ijtag and define modules
##########################################

# Tell Tessent Shell that want to perform
# Pattern retargeting and use Tessent SiliconInsight
# to execute the IJTAG pattern
set_context patterns -ijtag -silicon_insight
set modules {sib tap tdr2 block1 instrument chip2}

#############################################
#### 2. Setup SID hardware and CDP
#############################################

# Read in configuration data file, SidSetup.cfg
read_config_data SidSetup.cfg

# Launch SID tester process and create the new
# configuration data file
launch_sid_tester -cdp mycdp.cdp -new

#############################################
#### 3. Pattern retargeting
#############################################

# Read in all relevant ICLs and PDLs
read_icl ../data/icl/sib.icl
read_icl ../data/icl/tap.icl
read_icl ../data/icl/tdr2.icl
read_icl ../data/icl/block1.icl
read_icl ../data/icl/instrument.icl
read_icl ../data/icl/chip2.icl
source ../data/pdl/instrument.pdl

# Retarget these PDL commands to the chosen ICL hierarchy level
# referred to as the current_design
set_current_design chip2

# Define appropriate control signals
```

```
add_clocks ClkA -free_running -period 10ns

# Set system mode to analysis mode so that retarget analysis will
# be conducted
set_system_mode analysis

# Name pattern set and make it ready to be populated with the specified
# PDL  commands
open_pattern_set pset_1 -tester_period 50 -tck_ratio 1

# Call an iProc registered against the ICL module to compose the pattern
iCall MyBlock1.MyBlock1Instrument.run_testa red

# Close the previously opened pattern set
close_pattern_set

# To report the names of the pattern sets, the associated timeplates,
# the number of vectors, and whether the pattern set has been saved
report_pattern_sets

#############################################
#### 4. Write out the retargeted patterns to appropriate formats
#############################################

write_patterns pattern.v -pattern_sets pset_1 -verilog -replace
write_patterns pattern.stil -pattern_sets pset_1 -stil -replace
write_patterns pattern.svf -pattern_sets pset_1 -svf -replace
write_patterns pattern.pdl -pattern_sets pset_1 -pdl -replace

#############################################
#### 5. Add and execute the test
#############################################

# Create a CDP test “test1” with pattern_1.svf
add_cdp_test test1 -patterns pattern.svf

# Report variables as defined in the pattern for cdp_test test1
report_cdp_test_variables -cdp_test test1

# Execute cdp_test “test1”
# Executes pattern.svf and returns only the PASS/FAIL status
execute_cdp_test test1

# Execute pattern.svf and returns all failures
execute_cdp_test test1 -collect_data_type_list variable

# Shut down the SID test process
shutdown_sid_tester
```

Chapter 3
Using Tessent SiliconInsight in a Simulated Environment with SimDUT

Rather than attaching hardware adapters as described in Chapter 2, "Interactive Debugging with Tessent SiliconInsight Desktop," you can use SimDUT to simulate the application of test patterns on a Tessent SiliconInsight Desktop tester. This chapter describes how to use SimDUT to validate a device and contains the following sections:

Preparing for SimDUT Test Pattern Simulation

SimDUT validation requires you to generate the following two files:

- Configuration data file for SimDUT—Specifies the Verilog simulator as the tester type.
- SIMDUT.v top-level Verilog netlist—Specifies the host machine and socket information for the Verilog simulator and initializes the simulator as a separate process.

Prerequisites

- You must have a licensed installation of the ModelSim or Questa Verilog simulator from Mentor Graphics.
- You must be able to load the design's Verilog netlist into ModelSim or Questa. As a recommendation, simulate your Verilog netlist with a testbench file to ensure that the netlist is a functional design netlist.

Procedure

1. Generate a SIMDUT.v top-level netlist file as described in "The SIMDUT.v Top-Level Verilog Netlist File."

   Start the simulator and extract the simdutport value. Refer to your Mentor Graphics ModelSim or Questa documentation for further information.

2. Generate a configuration data file named “SidSetup.cfg” for SimDUT as described in “The Configuration Data File for SimDUT.”

Results

You can now use SimDUT to simulate an IJTAG test pattern and return the failure results as described in “Performing SimDUT Test Pattern Simulation.”

Performing SimDUT Test Pattern Simulation

The following procedure executes and simulates an IJTAG test pattern for the DUT and returns the simulated failure results.

Prerequisites

- You have Verilog design files.
- You have prepared for SimDUT simulation as described in “Preparing for SimDUT Test Pattern Simulation.”

Procedure

1. From a shell, invoke Tessent Shell using the following syntax:

   ```
   % tessent -shell
   ```

   After invocation, the tool is in setup mode. Refer to the “tessent” command description in the *Tessent Shell Reference Manual* for additional invocation options.

2. Set the tool context to enable Tessent SiliconInsight within Tessent Shell as follows:

   SETUP>set_context patterns -ijtag -silicon_insight

   When using a previously generated CDP, you do not need the -ijtag switch.

3. Read in the configuration data file for SimDUT as follows:

   SETUP>read_config_data *my_config_file*.cfg

4. Compile the Verilog designs, the library, and the SIMDUT.v netlist file.

5. Invoke the ModelSim or Questa Verilog simulator, ensuring that you:

   - Link the libSimdutVpi.so library to the simulator. The library contains SimDUT functions based on standard Verilog Procedural Interface functions. This library receives the test stimuli from the SID tester process and returns the failing cycle data to the SID tester process.

     Specify the pathname, as follows:

     ```
     <sts_tree>/share/SiliconInsight/Simdut/lnx-x86/[lib64 | lib32]
     ```

- Specify the module name as defined in the SIMDUT.v top-level netlist file.
- Specify the port name if you are not using the default 2111 port. Do this by specifying the following argument when you invoke the Verilog simulator:

```
+simdutport:<N>          // where N is the port number
```

For example, for the Mentor Graphics Questa simulator, suppose the sts_tree is /tessent/tsi/test_area, and you are using port 5555 as defined in your configuration data file. You want to simulate the SIMDUT1 module as defined in your SIMDUT.v file. Invoke the simulator as follows:

```
vsim -c -voptargs=”+acc” +nowarnTFMPC +simdutport:5555
-do “run -all” SIMDUT1 -pli
tessent/tsi/test_area/share/SiliconInsight/Simdut/lnx-
x86/lib64/libSimdutVpi.so
```

Refer to the Mentor Graphics Questa documentation for details.

6. Launch the SID tester process and create a new CDP as follows:

   SETUP>launch_sid_tester -cdp *my_cdp_path* -new

   When using a previously generated CDP, you do not need the -new switch.

Note
SVF-based CDPs are configured with asynchronous clocking. The tool does not provide clock waveforms to the DUT, therefore you must source the appropriate clocks yourself.

7. Generate your IJTAG test pattern and save it as an annotated SVF test pattern with the write_patterns command. Refer to the *Tessent IJTAG Users Manual* for details about this process.

8. Add the SVF pattern file to the CDP and create a test program for that pattern file as follows:

   SETUP>add_cdp_test test1 -pattern *my_pattern_path*.svf

   The test program manipulates and executes test pattern files. You can only specify one test pattern file per test.

   To view your tests, specify the get_cdp_test_list command. To delete a test, specify the delete_cdp_test command.

9. Execute the test. For example:

   SETUP>execute_cdp_test test1 -collect_data_type_list variable

   This command executes the collect_data_type_list procedure for the test pattern associated with the test named “test1.” SimDUT converts the SVF file into simulation events, and after simulation completes, it converts the failure results back to SVF.

By default, the tool executes a go-nogo procedure and returns the pass/fail status for the specified test.

10. Optionally, inject a fault and execute the test again. For example:

```
SETUP>add_simdut_fault -signal SIMDUT1.CHIP_INST.memory00.q_a -fault 0
SETUP>execute_cdp_test test1 -collect_data_type_list variable
```

These commands first inject a stuck-at 0 fault at the SIMDUT_TB.CHIP.memory00.q_a signal, and then execute test1 and return the variable failing cycle results.

Note
The simulator injects faults on nets, not ports. Refer to "SimDUT Fault Injection and Signal Naming Convention" for more information about SimDUT fault injection.

For information about SimDUT fault injection for memory models, see "SimDUT Fault Injection for Memory Models."

If a net consists of escaped identifiers, you can inject a fault as shown in the following example. The curly brackets prevent the tool from flattening the pathname thus making it possible to use fault injection.

```
SETUP>add_simdut_fault -signal { /TB/CHIP/m8051_i/u4/\LDATAA[3]   } -fault 1
```

You can specify the add_simdut_fault command multiple times. The tool supports stuck-at 0 (SA0) and stuck-at 1 (SA1) fault injection.

To delete the injected fault, specify the delete_simdut_fault command.

11. Shut down the SID tester process when you are done.

```
SETUP>shutdown_sid_tester
```

Results

After executing a given test, the tool generates a tabulated report that contains the results from the simulated test execution. While the results are based on simulation, the format of the results is identical to those returned by Tessent SiliconInsight when you test the actual silicon.

Example 3-1 shows the variable failure results for test1.

Example 3-1. SimDUT Simulation Results

```
//  sub-command: execute_cdp_test test1 -collect_data_type_list variable
//  Test 'test1' failed.
//  Variable failures and unmapped failures of pattern 'test1.svf' :
//
//  PatternSet    Variable            Pin                   Expected
//                                                            Actual
//  ------------  ------------------  ---  -------------------------
//  lbist_normal  top_edt_i.misr(51)  tdo  b110011111100010101101101
//                                         b010101101100101001101101
//  ------------  ------------------  ---  -------------------------
//  lbist_normal  top_edt_i.misr(77)  tdo  b000110101100111110010010
//                                         b000110111000111110000110
//  ------------  ------------------  ---  -------------------------
//  lbist_normal  top_edt_i.misr(92)  tdo  b101101110100001110001011
//                                         b111010101110101001010101
//  ------------  ------------------  ---  -------------------------
```

Sample Dofiles

The following example executes two tests, test3 and test4.

Example 3-2. Dofile Example for SimDUT Simulation

```
set_context patterns -ijtag -silicon_insight
 SidSetup.cfg
launch_sid_tester -cdp tty -new
sys mkdir tty/SVF

// test CDP with the "help" command
execute_cdp_cmd help

// Execute TAP test tapTest1_noTRST
sys cp ../src/SVF/tapTest1_noTRST.svf tty/SVF/tapTest1_noTRST.svf
add_cdp_test test3 -pattern tty/SVF/tapTest1_noTRST.svf
execute_cdp_test test3
execute_cdp_test test3 -collect_data_type_list variable

// Execute test readID
system cp ../src/SVF/readID.svf tty/SVF/readID.svf
add_cdp_test test4 -pattern tty/SVF/readID.svf
execute_cdp_test test4 -collect_data_type_list variable

shutdown_sid_tester
system sleep 10
exit
```

The following example performs fault injection before executing the test6 test and then deletes the injected faults before shut down.

```
set env(MENTOR_DOMAIN_ENABLE) 1
set_access_code -code [ create_access_code TS_SiliconInsight ]

 SidSetup.cfg
sys rm -rf tty
launch_sid_tester -cdp tty -new

sys mkdir tty/SVF
sys cp ../src/SVF/membistpv_1.tck.svf tty/SVF/membistpv_1.tck.svf
add_cdp_test test6 -pattern tty/SVF/membistpv_1.tck.svf
execute_cdp_test test6 -collect_data_type_list variable

simdut_add_fault -signal SIMDUT_TB.CHIP.memory00.q_a   -fault 0
simdut_add_fault -signal SIMDUT_TB.CHIP.memory01.q_a   -fault 0
simdut_add_fault -signal SIMDUT_TB.CHIP.memory02.q_a   -fault 0
simdut_add_fault -signal SIMDUT_TB.CHIP.memory03.q_a   -fault 0
execute_cdp_test test6 -collect_data_type_list variable

simdut_delete_fault -signal SIMDUT_TB.CHIP.memory03.q_a
simdut_delete_fault -signal SIMDUT_TB.CHIP.memory02.q_a
simdut_delete_fault -signal SIMDUT_TB.CHIP.memory01.q_a
simdut_delete_fault -signal SIMDUT_TB.CHIP.memory00.q_a
execute_cdp_test test6 -collect_data_type_list variable

shutdown_sid_tester
exit
```

The Configuration Data File for SimDUT

The configuration data file for SimDUT contains Tessent SiliconInsight Desktop-specific commands that define the pin map, testers, and other data that are required to prepare Tessent SiliconInsight Desktop to communicate with the simulator process.

Figure 3-1 shows the configuration data file structure for SimDUT.

Figure 3-1. Configuration Data File Structure for SimDUT

```
SidSetupSpecification {
   selected_tester : <nameOfSelectedTester>; // specify tester name
                                 // for adaptor of type “SIMDUT”
   Protocol {
      SvfTapPins { // specify design pins that correspond to SVF TAP ports
         TCK : <designPinName>;
         TDI : <designPinName>;
         TDO : <designPinName>;
         TMS : <designPinName>;
         TRST : <designPinName>; // use "-" if it does not exist
      }
   }
   Tester( <nameOfSelectedTester> ) { // same name as the selected tester
   Adapter {                               // only define one adapter
      SIMDUT {
         simdut_host : <name of host>;     // default is localhost
         simdut_port : <port number>;               // default is 2111
      }
   }
}
```

You can specify multiple testers within one configuration data file. However, you can only specify one of the testers as the selected_tester. The selected_tester must be the name of the tester that has an adaptor of type SIMDUT. The selected_tester launches SimDUT.

By default, the SimDUT process uses port 2111 for inter-process communication with the Verilog simulator process. You can specify any unused port number between 2000 and 65536.

Note

If you want to specify a port number, ensure that no other applications are using the port and choose a port number greater than 50000 to avoid port conflict.

For SimDUT, the Pinmap section in the configuration data file is optional. If omitted, the tool assumes that all DUT pins are connected. If you do not plan to connect all DUT pins to the hardware adaptor, you can simulate this scenario by including the Pinmap section. SimDUT will then only assume that the specified pins are connected.

The following example illustrates a configuration data file for a SimDUT tester named my_simdut.

Example 3-3. Configuration Data File for SimDUT my_simdut

```
SidSetupSpecification {
  selected_tester : my_simdut;
  Protocol {
    SVFTapPins {
      TCK : TCK;
      TDI : TDI;
      TDO : TDO;
      TMS : TMS;
      TRST : TRST;
    }
  }
  Tester( my_simdut ) {
    // host : localhost;
    // Port : 30000;
    Adapter {
      SIMDUT {
        simdut_host : localhost;
        simdut_port : 5555;
      }
    }
  }
```

The SIMDUT.v Top-Level Verilog Netlist File

The SimDUT flow requires a SIMDUT.v top-level Verilog netlist file. This file provides a top-level design wrapper and starts the SIMDUT library function by calling $SimdutInit in the initial block.

Example 3-4 shows the format of the SIMDUT.v wrapper for a design whose top-level name is CHIP. You can include a free running clock if the free running clock waveforms are not provided in any other existing design netlist files.

Example 3-4. SIMDUT.v Top-Level Netlist File for a Router Design

```
`timescale 100ps/10ps
```

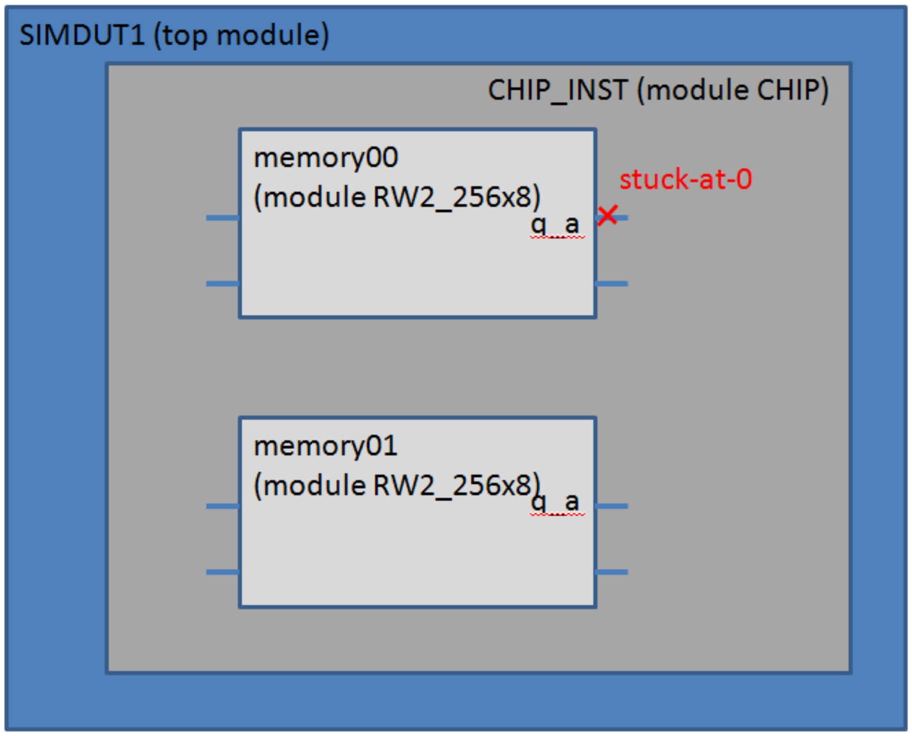
```
module SIMDUT1;                          // declare ports
    wire          TDI;
    wire          TDO;
    wire          TMS;
    wire          TCK;
    wire          CLKA;
    wire          TRST;
    wire          FI_RST_EN;
    wire          AuxOut;

assign FI_RST_EN = 1'b1;  // static pin as specified in P1687 SVF file
CHIP_INST(                // create one instance of top-level design
         .TDI(TDI), .TDO(TDO), .TMS(TMS), .TCK(TCK), .CLKA(CLKA),
         .TRST(TRST), .FI_RST_EN(FI_RST_EN), .AuxOut(AuxOut));

      membistpv_1_clka mbist_clka (.CLKA(CLKA) );  // system clock

      initial
         begin
            $SimdutInit;  // initialize the SimDUT VPI library
         end
      endmodule

module membistpv_1_clka ( CLKA );  // generate the system clock
      output         CLKA;
      reg           CLKA;

      initial begin
           #0   CLKA = 1'b0;
      forever begin
           #50  CLKA = ~CLKA;
      end
   end
endmodule
```

SimDUT Fault Injection and Signal Naming Convention

Given the following command, Figure 3-2 shows how SimDUT injects a stuck-at fault on a specified signal.

add_simdut_fault -signal SIMDUT1.CHIP_INST.memory00.q_a -fault 0

Figure 3-2. SimDUT Stuck-At Fault Insertion



When you inject a stuck-at-0 fault or a stuck-at-1 fault into the specified signal, SimDUT uses the Verilog simulation capabilities to force the signal to 0 or 1, respectively. The Verilog capability has the following behavior: When you specify a pin pathname for the fault you want to inject, Verilog forces the associated signal to the specified value.

When you inject a fault at a specific pin pathname, SimDUT actually injects the fault at the whole net, not only on the specific pin. The fault acts like a fault injected on the upstream driver pin of the associated net. If you specify the input port of a cell and this port is attached to a net with fanout, then the fault acts like a fault injected on the driver port of that net.

SimDUT Signal Naming Convention

For the add_simdut_fault command, derive the name for the signal using the Verilog source code. You must use "." to separate the names of the modules and pins. Example 3-5 shows a Verilog source code snippet that has three modules, corresponding to Figure 3-2. The signal name starts with the top-module name followed by the lower-level module names based on the hierarchy of the Verilog netlist. By following the hierarchy shown in the sample Verilog netlist, you derive the signal name "SIMDUT1.CHIP_INST.memory00.q_a."

Example 3-5. Sample Verilog Netlist

```
module SIMDUT1;
  CHIP CHIP_INST (…);
```

```
    initial
        $SimdutInit;
      end
  endmodule

  module CHIP (…);
  ……
      RW2_256x8 memory00 (…);
      RW2_256x8 memory01 (…);

  ……
  endmodule

  module RW2_256x8 (......);
  ......
          output  q_a;
  ......
  endmodule
```

SimDUT Fault Injection for Memory Models

SimDUT fault injection relies on the Verilog simulator to force values on specified signals. This means that the actual resolution of injected faults depends on the memory model you use.

- High-level RTL memory models: Injecting faults might be limited to the external interface of the memory—that is, the address or data ports.
- Low-level gate memory models: To inject faults at the cell level, the internal structure of the memory module must be represented in a model you are using.

Example: High-Level RTL Memory Model

Example 3-6 shows a high-level RTL memory model in which fault injection at the cell level is not possible. With this model, you can only inject faults on the memory interface, as follows:

add_simdut_fault -signal { /mem_inst/q[3] } -fault 0

Example 3-6. Verilog Source Code for High-Level RTL Memory Model

```
module SYNC_1RW_64x8 (
   Q,
   CLK,
   CEN,
   WEN,
   A,
   D,
   OEN
);
   reg [BITS-1:0]          MEM [word_depth-1:0];
   reg [BITS-1:0]          Q_REG;

assign Q = (~OEN) ? Q_REG : wordz ;
integer i;
```

```
initial begin
    for (i=0;i<word_depth; i=i+1)
        begin
            MEM[i]={BITS{1'b0}};
        end
    Q_REG={BITS{1'b0}};
end

always @ (posedge CLK) begin
   if (~CEN) begin
     if (~WEN) begin
         MEM[A]  <= D;
         Q_REG   <= D;
      end else begin
         Q_REG   <= MEM[A];
      end
   end
end
endmodule
```

Example: Low-Level Gate Memory Model

To inject faults at the cell level, you must use a memory model similar to that shown in Example 3-7. In this example, all memory cells are represented by a distinct entity in the memory model. You would perform fault injection at the cell level as follows:

add_simdut_fault -signal { /mem_instance/mem_cell[63][4]/q } -fault 0

You can generate low-level memory models by synthesizing high-level RTL models.

Example 3-7. Verilog Source Code for Low-Level Gate Memory Model

```
module SYNC_1RW_64x8 ( Q, CLK, CEN, WEN, A, D, OEN );
  output [7:0] Q;
  input [5:0] A;
  input [7:0] D;
  input CLK, CEN, WEN, OEN;
  wire \MEM[63][7];
  wire \MEM[63][6];
  wire \MEM[63][5];
  ...
  wire \MEM[0][0];

  DFF \MEM_reg[63][7]  ( .D(n682), .CLK(CLK), .Q(\MEM[63][7] ) );
  DFF \MEM_reg[63][6]  ( .D(n681), .CLK(CLK), .Q(\MEM[63][6] ) );
  DFF \MEM_reg[63][5]  ( .D(n680), .CLK(CLK), .Q(\MEM[63][5] ) );
  ...
endmodule
```

Appendix 4
Getting Help

There are several ways to get help when setting up and using Tessent software tools. Depending on your need, help is available from documentation, online command help, and Mentor Graphics Support.

Documentation

The Tessent software tree includes a complete set of documentation and help files in PDF format. Although you can view this documentation with any PDF reader, if you are viewing documentation on a Linux file server, you must use only Adobe® Reader® versions 8 or 9, and you must set one of these versions as the default using the MGC_PDF_READER variable in your *mgc_doc_options.ini* file.

For more information, refer to "Specifying Documentation System Defaults" in the *Managing Mentor Graphics Tessent Software* manual.

You can download a free copy of the latest Adobe Reader from this location:

http://get.adobe.com/reader

You can access the documentation in the following ways:

- **Shell Command** — On Linux platforms, enter **mgcdocs** at the shell prompt or invoke a Tessent tool with the -Manual invocation switch. This option is available only with Tessent Shell and the following classic point tools: Tessent FastScan, Tessent TestKompress, Tessent Diagnosis, and DFTAdvisor.
- **File System** — Access the Tessent bookcase directly from your file system, without invoking a Tessent tool. From your product installation, invoke Adobe Reader on the following file:

  ```
  $MGC_DFT/doc/pdfdocs/_bk_tessent.pdf
  ```
- **Application Online Help** — You can get contextual online help within most Tessent tools by using the "help -manual" tool command:

  ```
  > help dofile -manual
  ```

  This command opens the appropriate reference manual at the "dofile" command description.

Mentor Graphics Support

Mentor Graphics software support includes software enhancements, access to comprehensive online services with SupportNet, and the optional On-Site Mentoring service.

For details, refer to this page:

http://supportnet.mentor.com/about

If you have questions about a software release, you can log in to SupportNet and search thousands of technical solutions, view documentation, or open a Service Request online:

http://supportnet.mentor.com

If your site is under current support and you do not have a SupportNet login, you can register for SupportNet by filling out a short form here:

http://supportnet.mentor.com/user/register.cfm

All customer support contact information is available here:

http://supportnet.mentor.com/contacts/supportcenters/index.cfm

Third-Party Information

For information about third-party software included with this release of Tessent products, refer to the *Third-Party Software for Tessent Products*.

End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/eula

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. **ORDERS, FEES AND PAYMENT.**

1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement (each an "Order"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not those documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order or presented in any electronic portal or automated order management system, whether or not required to be electronically accepted, will not be effective unless agreed in writing and physically signed by an authorized representative of Customer and Mentor Graphics.

1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice. Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

1.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 5.2); (b) for Customer's internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer provides any feedback or requests any change or enhancement to Products, whether in the course of receiving support or consulting services, evaluating Products, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

3. **ESC SOFTWARE.** If Customer purchases a license to use development or prototyping tools of Mentor Graphics' Embedded Software Channel ("ESC"), Mentor Graphics grants to Customer a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESC compilers, including the ESC run-time libraries distributed with ESC C and C++ compiler Software that are

linked into a composite program as an integral part of Customer's compiled computer program, provided that Customer distributes these files only in conjunction with Customer's compiled computer program. Mentor Graphics does NOT grant Customer any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into Customer's products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

4. **BETA CODE.**

4.1. Portions or all of certain Software may contain code for experimental testing and evaluation (which may be either alpha or beta, collectively "Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. Mentor Graphics may choose, at its sole discretion, not to release Beta Code commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **RESTRICTIONS ON USE.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer becomes aware of such unauthorized disclosure or use. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any source code from Software. Log files, data files, rule files and script files generated by or for the Software (collectively "Files"), including without limitation files containing Standard Verification Rule Format ("SVRF") and Tcl Verification Format ("TVF") which are Mentor Graphics' trade secret and proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Products or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer's employees or on-site contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense, or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer with updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics' then current End-User Support Terms located at http://supportnet.mentor.com/supportterms.

7. **LIMITED WARRANTY.**

7.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor

Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification, improper installation or Customer is not in compliance with this Agreement. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

7.2. THE WARRANTIES SET FORTH IN THIS SECTION 7 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

8. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 8 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

9. **HAZARDOUS APPLICATIONS.** CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING ITS PRODUCTS USED IN APPLICATIONS WHERE THE FAILURE OR INACCURACY OF ITS PRODUCTS MIGHT RESULT IN DEATH OR PERSONAL INJURY ("HAZARDOUS APPLICATIONS"). EXCEPT TO THE EXTENT THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **INDEMNIFICATION.** CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **INFRINGEMENT.**

11.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to such action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

11.2. If a claim is made under Subsection 11.1 Mentor Graphics may, at its option and expense: (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

11.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by Customer that is deemed willful. In the case of (h), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.

11.4. THIS SECTION 11 IS SUBJECT TO SECTION 8 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, FOR DEFENSE, SETTLEMENT AND DAMAGES, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

12. **TERMINATION AND EFFECT OF TERMINATION.**

12.1. If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

12.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

13. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and United States ("U.S.") government agencies, which prohibit export, re-export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export or re-export Products in any manner without first obtaining all necessary approval from appropriate local and U.S. government agencies. If Customer wishes to disclose any information to Mentor Graphics that is subject to any U.S. or other applicable export restrictions, including without limitation the U.S. International Traffic in Arms Regulations (ITAR) or special controls under the Export Administration Regulations (EAR), Customer will notify Mentor Graphics personnel, in advance of each instance of disclosure, that such information is subject to such export restrictions.

14. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. The parties agree that all Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to U.S. FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. government or a U.S. government subcontractor is subject solely to the terms and conditions set forth in this Agreement, which shall supersede any conflicting terms or conditions in any government order document, except for provisions which are contrary to applicable mandatory federal laws.

15. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

16. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 16 shall survive the termination of this Agreement.

17. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the U.S. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, U.S., if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. Nothing in this section shall restrict Mentor Graphics' right to bring an action (including for example a motion for injunctive relief) against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

18. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

19. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements. Some Software may contain code distributed under a third party license agreement that may provide additional rights to Customer. Please see the applicable Software documentation for details. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 140201, Part No. 258976