

FS2 Logic Navigator OCI Generator

July 25, 2006



First Silicon Solutions
MIPS Technologies, Inc.
4000 SW Kruse Way Place
Bldg 3, Suite 210
Lake Oswego, OR 97035
voice: +1-503-489-0311
fax: +1-503-489-0315
<http://www.fs2.com>
info@fs2.com
support@fs2.com

Copyright (c) 2004-2006 MIPS Technologies, Inc. All rights reserved.

Unpublished rights (if any) reserved under the copyright laws of the United States of America and other countries.

This document contains information that is proprietary to MIPS Technologies, Inc. ("MIPS Technologies"). Any copying, reproducing, modifying or use of this information (in whole or in part) that is not expressly permitted in writing by MIPS Technologies or an authorized third party is strictly prohibited. At a minimum, this information is protected under unfair competition and copyright laws. Violations thereof may result in criminal penalties and fines.

Any document provided in source format (i.e., in a modifiable form such as in FrameMaker or Microsoft Word format) is subject to use and distribution restrictions that are independent of and supplemental to any and all confidentiality restrictions. UNDER NO CIRCUMSTANCES MAY A DOCUMENT PROVIDED IN SOURCE FORMAT BE DISTRIBUTED TO A THIRD PARTY IN SOURCE FORMAT WITHOUT THE EXPRESS WRITTEN PERMISSION OF MIPS TECHNOLOGIES, INC.

MIPS Technologies reserves the right to change the information contained in this document to improve function, design or otherwise. MIPS Technologies does not assume any liability arising out of the application or use of this information, or of any error or omission in such information. Any warranties, whether express, statutory, implied or otherwise, including but not limited to the implied warranties of merchantability or fitness for a particular purpose, are excluded. Except as expressly provided in any written license agreement from MIPS Technologies or an authorized third party, the furnishing of this document does not give recipient any license to any intellectual property rights, including any patent rights, that cover the information in this document.

The information contained in this document shall not be exported, reexported, transferred, or released, directly or indirectly, in violation of the law of any country or international law, regulation, treaty, Executive Order, statute, amendments or supplements thereto. Should a conflict arise regarding the export, reexport, transfer, or release of the information contained in this document, the laws of the United States of America shall be the governing law.

The information contained in this document constitutes one or more of the following: commercial computer software, commercial computer software documentation or other commercial items. If the user of this information, or any related documentation of any kind, including related technical data or manuals, is an agency, department, or other entity of the United States government ("Government"), the use, duplication, reproduction, release, modification, disclosure, or transfer of this information, or any related documentation of any kind, is restricted in accordance with Federal Acquisition Regulation 12.212 for civilian agencies and Defense Federal Acquisition Regulation Supplement 227.7202 for military agencies. The use of this information by the Government is further restricted in accordance with the terms of the license agreement(s) and/or applicable contract terms and conditions covering this information from MIPS Technologies or an authorized third party.

MIPS, MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS-3D, MIPS16, MIPS16e, MIPS32, MIPS64, MIPS-Based, MIPSsim, MIPSpro, MIPS Technologies logo, MIPS RISC CERTIFIED POWER logo, MIPS-VERIFIED, 4K, 4Kc, 4Km, 4Kp, 4KE, 4KEc, 4KEm, 4KEp, 4KS, 4KSc, 4KSd, M4K, 5K, 5Kc, 5Kf, 20K, 20Kc, 24K, 24Kc, 24Kf, 24KE, 24KEc, 24KEf, 25Kf, 34K, 34Kc, 34Kf, R3000, R4000, R5000, ASMACRO,

Atlas, "At the core of the user experience.", BusBridge, CorExtend, CoreFPGA, CoreLV, EC, JALGO, Malta, MDMX, MGB, PDtrace, the Pipeline, Pro Series, QuickMIPS, SEAD, SEAD-2, SmartMIPS, SOC-it, and YAMON are trademarks or registered trademarks of MIPS Technologies, Inc. in the United States and other countries.

FS2, the FS2 First Silicon Solutions logo, FS2 Navigator, Bus Navigator, Logic Navigator, System Navigator, Clam, FPGAVIEW, HyperDebug, HyperJTAG, MED, and OCI are trademarks or registered trademarks of MIPS Technologies, Inc. in the United States and other countries.

All other trademarks referred to herein are the property of their respective owners.

Table of Contents

1. PURPOSE	5
2. KNOWN LIMITATIONS	5
3. INSTALLATION.....	6
4. STARTING THE OCI GENERATOR.....	6
5. USING THE OCI GENERATOR.....	7
5.1. CREATING A PROJECT	7
5.2. SELECTING PARAMETERS FOR CLAM IP INSTANTIATION	7
5.3. IDENTIFYING THE HDL SOURCE FILES	8
5.4. IDENTIFYING THE ROOT MODULE.....	9
5.5. SELECTING THE INPUTS TO CLAM.....	10
5.6. GENERATING THE INSTRUMENTED HDL OUTPUT.....	13
5.7. SAVING THE PROJECT	14
6. RUNNING BATCH MODE	14
7. CHARACTERISTICS OF THE GENERATED RTL FILE OUTPUT	15
7.1. THE .CLAM FILE OUTPUT.....	15
APPENDIX A – CONNECTION POINTS FOR ON-CHIP CLAM IP	16
APPENDIX B – CONNECTION POINTS FOR OFF-CHIP CLAM IP	17

1. Purpose

The FS2 Logic Navigator OCI (On-Chip Instrumentation) Generator is a software tool designed to aid hardware designers in incorporating the FS2 CLAM® IP into designs that utilize an Actel FPGA or other device. The OCI generator supports insertion of FS2 CLAM IP into Verilog and VHDL designs.

Two related documents, [FS2 Off-Chip Logic Navigator™ Instantiation, Simulation and Synthesis Guide](#) and [FS2 On-Chip Logic Navigator™ Instantiation, Simulation and Synthesis Guide](#), are available as references for designers who wish to incorporate the FS2 CLAM (Configurable Logic Analyzer Module) IP blocks manually rather than through the OCI Generator software, or who simply wish to understand the requirements of incorporating the FS2 CLAM IP at the HDL source level.

The function of the OCI Generator is severalfold:

- 1) to simplify the interactive selection of signals within a design to connect as inputs to the FS2 CLAM IP.
- 2) to generate instrumented HDL source code that adds trace interfaces and connections to a design's HDL source and instances and interfaces the FS2 CLAM IP to the designer's selected input and administrative signals
- 3) to generate a corresponding configuration (.CLAM) file that allows automates loading the signal names into the Logic Navigator GUI.

2. Known Limitations

There are some HDL source constructs that the OCI Generator does not support. It is safe to use the OCI Generator on HDL source that contains these constructs, but instances and signals within these constructs will not be visible as available signals and will not be available for automated hookup to the logic analyzer IP. Support for some or all of these constructs may be added in the future.

For those cases where OCI Generator does not support automated signal hookup to particular signals of interest, this issue can be overcome by manually incorporating the OCI into your design. The documents [FS2 Off-Chip Logic Navigator Instantiation, Simulation and Synthesis Guide](#) and [FS2 On-Chip Logic Navigator™ Instantiation, Simulation and Synthesis Guide](#) cover the details of manually modifying your HDL to incorporate Logic Navigator OCI.

The known limitations for Verilog are:

1. Signals of type **real** and **time** are currently not selectable for CLAM hookup.
2. Signal vectors or arrays whose bounds are not expressed entirely in terms of the following Verilog elements are not selectable for CLAM hookup: integer constants, integer parameters, standard Verilog operators.
3. Arrays of instances (and their child instances and signals) are not selectable for CLAM hookup.
4. Dynamically generated instances and signals (e.g. constructed within generate blocks) are not selectable for CLAM hookup.

The known limitations for VHDL are:

5. Signals with types *other* than **ieee.std_logic_1164.std_logic** and **ieee.std_logic_1164.std_logic_vector** are currently not selectable for CLAM hookup.

6. Signal vectors whose bounds are not expressed entirely in terms of the following VHDL elements are not selectable for CLAM hookup: integer constants, integer generics, standard VHDL integer operators. Expressions more complex than that (including function calls) are not currently supported.
7. Configuration specifications and configuration declarations are currently ignored. When analyzing instantiation relationships between modules, the OCI generator uses the default VHDL rules for mapping component instantiations to entities. Specifically, it looks for an entity in the work library with a name that is identical to the component name and then uses the most recently seen architecture body that is associated with that entity.
8. Dynamically generated instances and signals (e.g. constructed within generate statements) are not selectable for CLAM hookup.

3. Installation

The OCI Generator is installed as part of the Logic Navigator installation package. Please refer to the Getting Started document for Logic Navigator installation instructions.

For Actel designs, the Actel **ACTGEN** utility is required to be installed and is used by the Logic Navigator OCI Generator to create an instance of an on-chip memory block used for on chip trace data storage. ACTGEN is typically installed as part of the Actel Libero suite; please consult Actel documentation regarding installation details. For any other design type, the user is responsible for creating a compatible memory macro for use with Logic Navigator.

4. Starting the OCI Generator

From the Windows Start menu, select Programs -> FS2 -> Logic Navigator OCI Generator. The window shown in Figure 1 should appear.

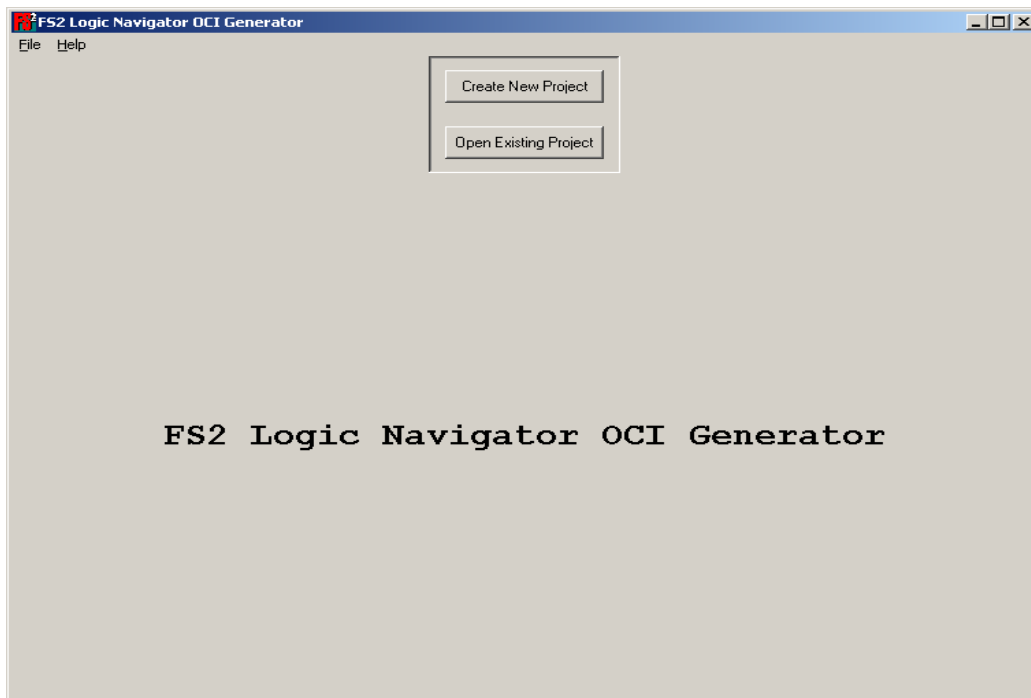


Figure 1 – OCI Generator window

5. Using the OCI Generator

5.1. Creating a project

The first step in using the Logic Navigator OCI Generator for the first time is to create a new *project*, that will hold the design's CLAM IP instantiation configuration and links to the designer's HDL source files. To create a new project, click the "Create New Project" button shown in Figure 1, or activate the File menu and select "New Project". A dialog box prompt for the location and file name to hold the project settings. Once a file name has been supplied and the dialog box is dismissed, the appearance of the window should change to that shown in Figure 2. In the Figure 2 example, a project "tjexe" in directory "c:\RTL\TJDSP" has been created. The tabbed area in the top half of the window will bring up a set of pages where the designer interactively defines the project settings – the properties of the desired CLAM instantiation (Parameters), the set and configuration of source files that comprise the design to be instrumented (Source Files and Root), the set(s) of signals to be connected to the logic analyzer inputs (Signal connections) and the generation of the instrumented RTL files (Generate Output). The white area in the bottom half of the window displays messages and status when the OCI Generator is analyzing source files or generating output.

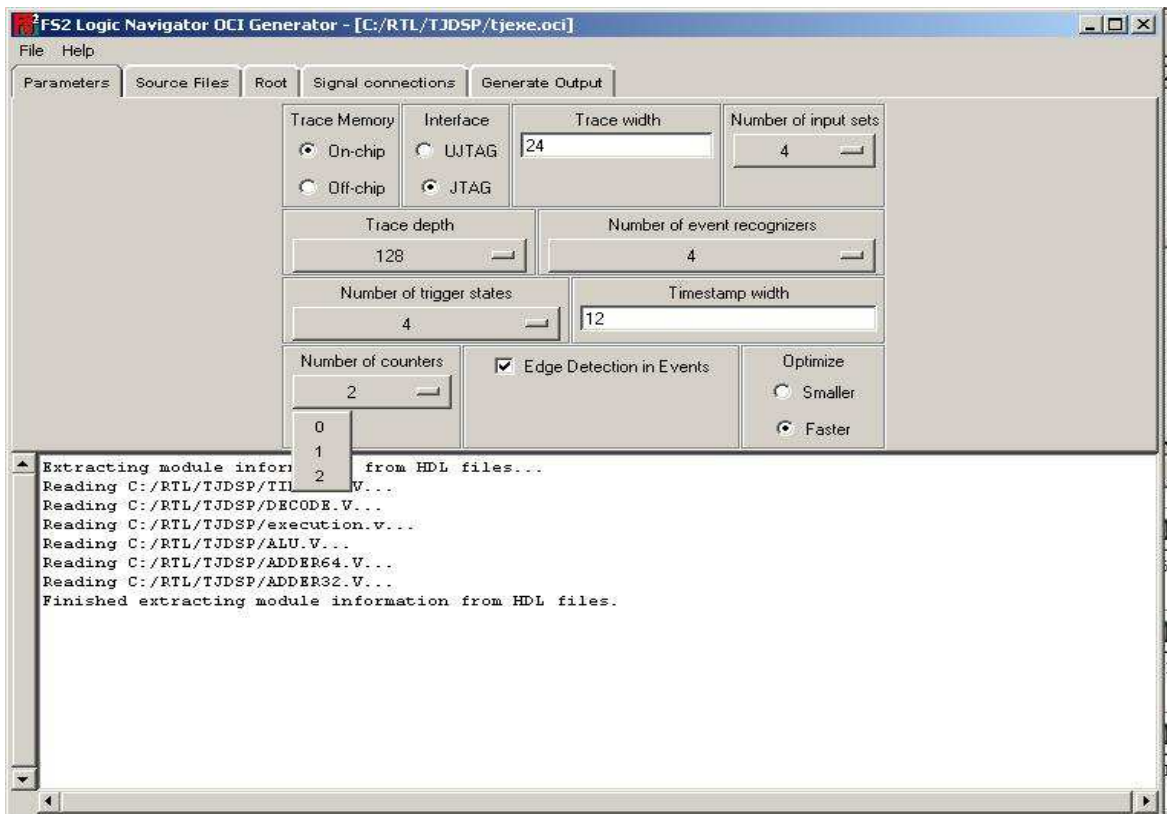


Figure 2 – New Project

5.2. Selecting parameters for CLAM IP instantiation

The first tabbed page of OCI Generator (Figure 2) allows the designer to select the desired instantiation parameters of the FS2 CLAM IP.

In the **Trace Memory** section, on-chip refers to the configuration that implements data storage and a triggering system entirely within the target device, and with which host software controls

the CLAM IP through the Actel FlashPro family of programmer hardware. Off-chip refers to a configuration in which the data storage and triggering systems are implemented in an external FS2 Logic Navigator hardware probe, which receives streaming data from the target device.

The interface types UJTAG or JTAG refers to the type of interface that Logic Navigator uses for on chip configuration. UJTAG is specific to Actel FLASH (ProASIC) devices and interfaces to the Actel FlashPro. All other applications will require a JTAG interface and JTAG probe. For off chip configuration, this interface defaults to a parallel debug port of a width defined by the number of signals being traced.

The **Trace width** is the number of signals interfaced to the CLAM for trace.

The CLAM IP supports the concept of **input sets**, which are separate sets of signals (the width of each set is exactly the trace width) that can be multiplexed onto the CLAM inputs, where only a single input set is connected within the scope of a single acquisition. The designer can select the number of input sets supported by his instantiation of CLAM.

Trace depth is the maximum number of samples that can be stored by on-chip CLAM.

On-chip CLAM supports a designer-selectable number of distinct **event recognizers, counters,** and **trigger states** as elements of its trigger system. Event recognizers can optionally implement **edge detection** or level detection. All of these elements can be used alone or in combination to construct simple or complex triggering scenarios at device testing time.

On-chip CLAM supports a **timestamp** of designer-selectable width. During an acquisition, the timestamp is incremented on each clock, and is stored with the sample data.

For the Off-Chip **Trace Memory** category, there are a few different CLAM IP configurations: clamext1x, clamext2x, and clamext4x. Please refer to the [FS2 Off-Chip CLAM™ Instantiation, Simulation and Synthesis Guide](#) document for an explanation of these different configurations.

5.3. Identifying the HDL source files

OCI Generator will read RTL files and allow selecting signals for attachment to FS2 CLAM IP. Either Verilog or VHDL is supported. The user should select the proper type so that the OCI Generator can properly read the RTL files. In the case of Verilog, if the source code uses the ``include` directive to pull in header files, the designer should specify the set of directories that the OCI Generator should search for include files. For Verilog source, the OCI Generator allows non-parameterized ``define` directives to be defined; a preprocessor definition specified within the OCI Generator will override a ``define` directive with the same name in the source.

Figure 3 shows the **Source Files** page, in which the source files, include paths, and defines are specified. Use the **Add** button to add one or more source files to the project. The example project in Figure 3 has had a single source file added to the project. It is not strictly necessary to add all of the design files to the OCI Generator project. At a minimum, it is necessary to make the OCI generator aware of the source file that contains the root module of the design, as well as the files that contain any signals that the designer will wish to connect to the CLAM inputs.

For VHDL source, the order of files in the list can be significant. For example, assuming that files named *upper.vhd* and *lower.vhd* are included in a design, if *upper.vhd* refers to a design unit or package declared in file *lower.vhd*, then *lower.vhd* should precede *upper.vhd* in the list of files. The **Move Up** and **Move Down** buttons can be used to change the order of the files.

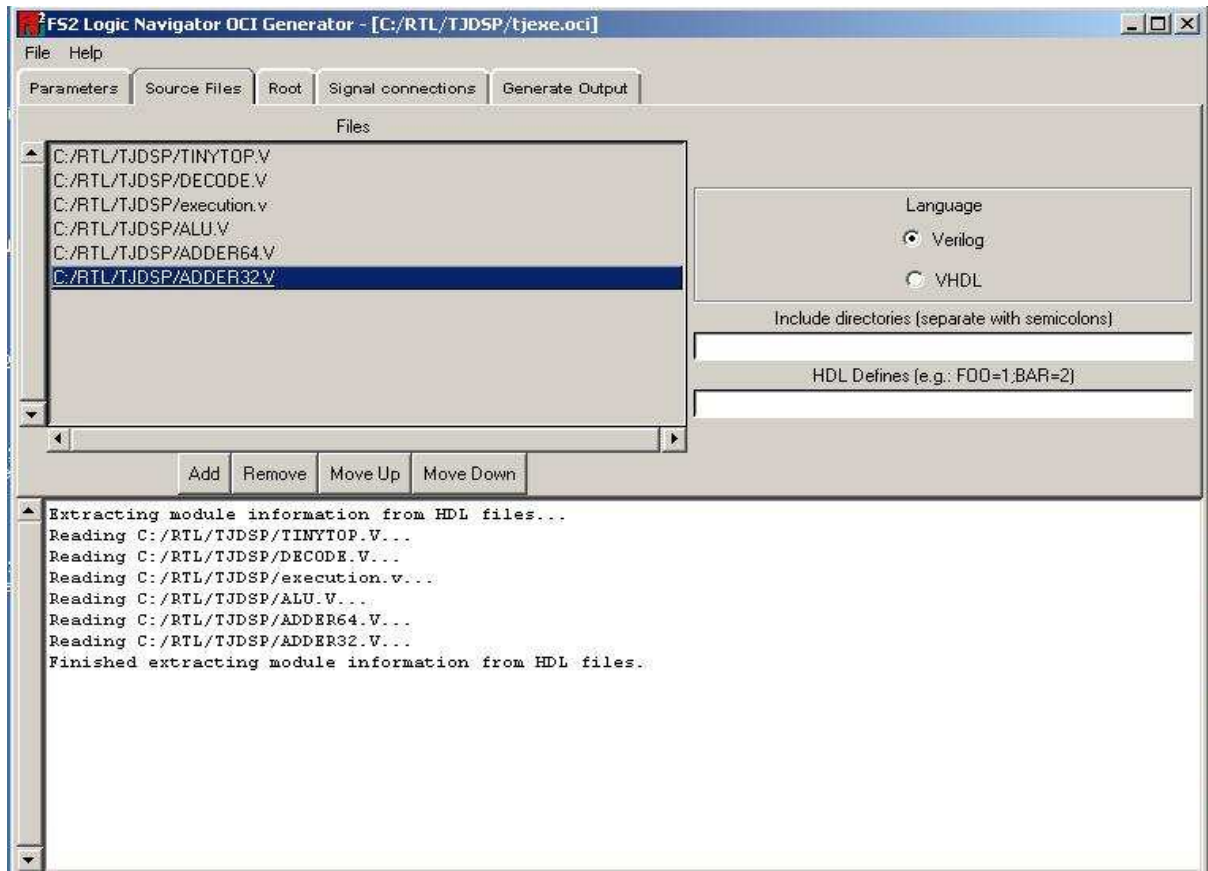


Figure 3 – Source Files page

5.4. Identifying the Root Module

Specifying the root module of the user's design defines the block level for implementing the CLAM logic and the on chip destination for routing signals between files and levels of hierarchy. for attachment to the CLAM. Figure 4 shows the **Root** page, in which the designer identifies the root module by selecting it from the menu button labeled **Select root module**. The OCI generator displays a list of candidate modules or entity/architecture pairs by analyzing the set of HDL files specified on the Source Files page (note the messages in the bottom status area of the OCI Generator window).

One important thing to note is that it is possible that there may be errors reflected in the status pane at this stage. For example, if the include path was not specified in the Source Files page, or the source file holding the actual root module was omitted from the project and is therefore missing from the list of modules in the menu button, an incorrect OCI implementation may be created. The recommended approach is to go back to the Source Files page and make required adjustments, and then return to the Root page. In general, the user at any time can go to any of the pages to make corrections or updates to an OCI.

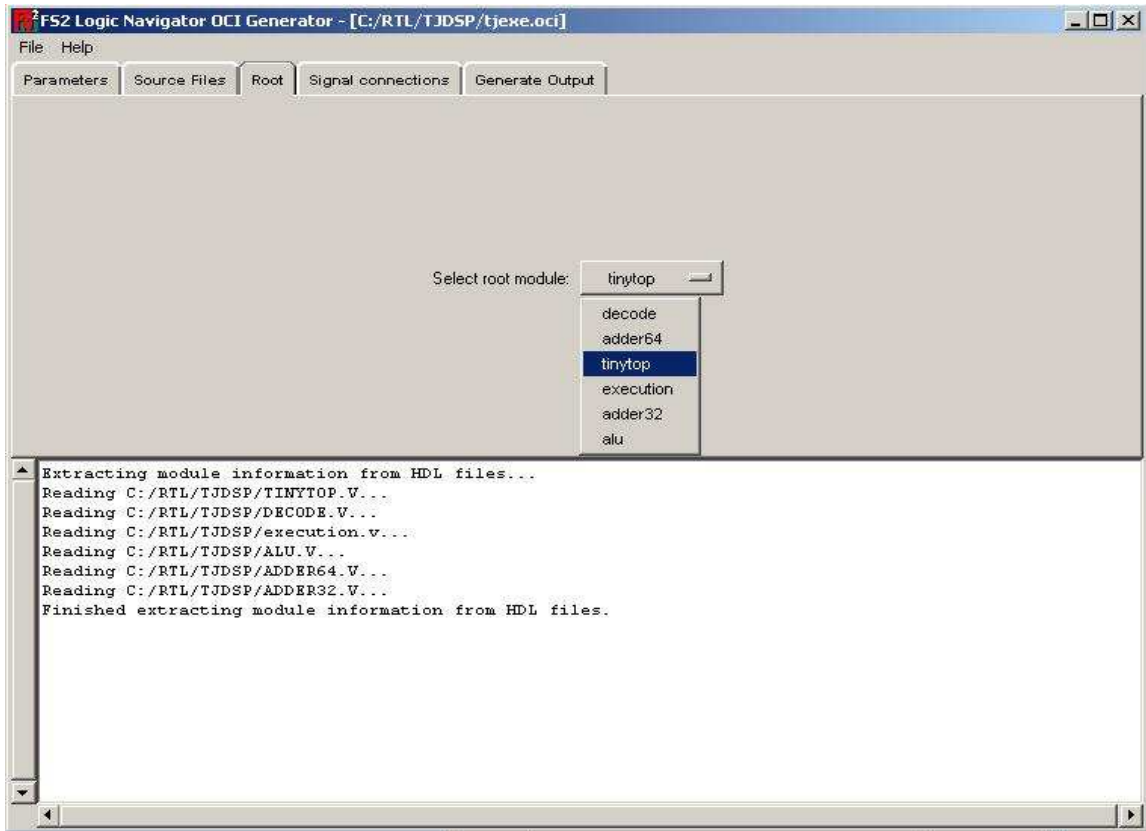


Figure 4 – Root page

5.5. Selecting the Inputs to CLAM

After the source files, OCI parameters and options have been defined, and root selected, the signal connections from the design to the OCI must be defined. Figure 5 shows the Signal Connections page of the OCI Generator. On the left side of the page appears the **Available Signals** tree: a hierarchical tree of instance and signal names as extracted from the set of source files that have been added to the project. On the right side of the page appears the **Selected Inputs** list, which reflects the set of signals that the designer has selected to connect to various CLAM inputs (clock, reset, data inputs, trigger in) and outputs (trigger out, gpreg). These are discussed in detail in the Logic Navigator ISS document. The menu button directly above the **Selected Inputs** lists which CLAM connection is currently reflected for a given CLAM input; changing the menu selection causes the list to update accordingly. Some of the CLAM inputs are mandatory, requiring the designer to specify signal connections for these inputs. (clock) Other CLAM inputs are optional (reset, triggers) and need not be associated with signals in the user design. See Appendices A and B for a list of CLAM connection points (ports) for on-chip and off-chip CLAM, respectively.

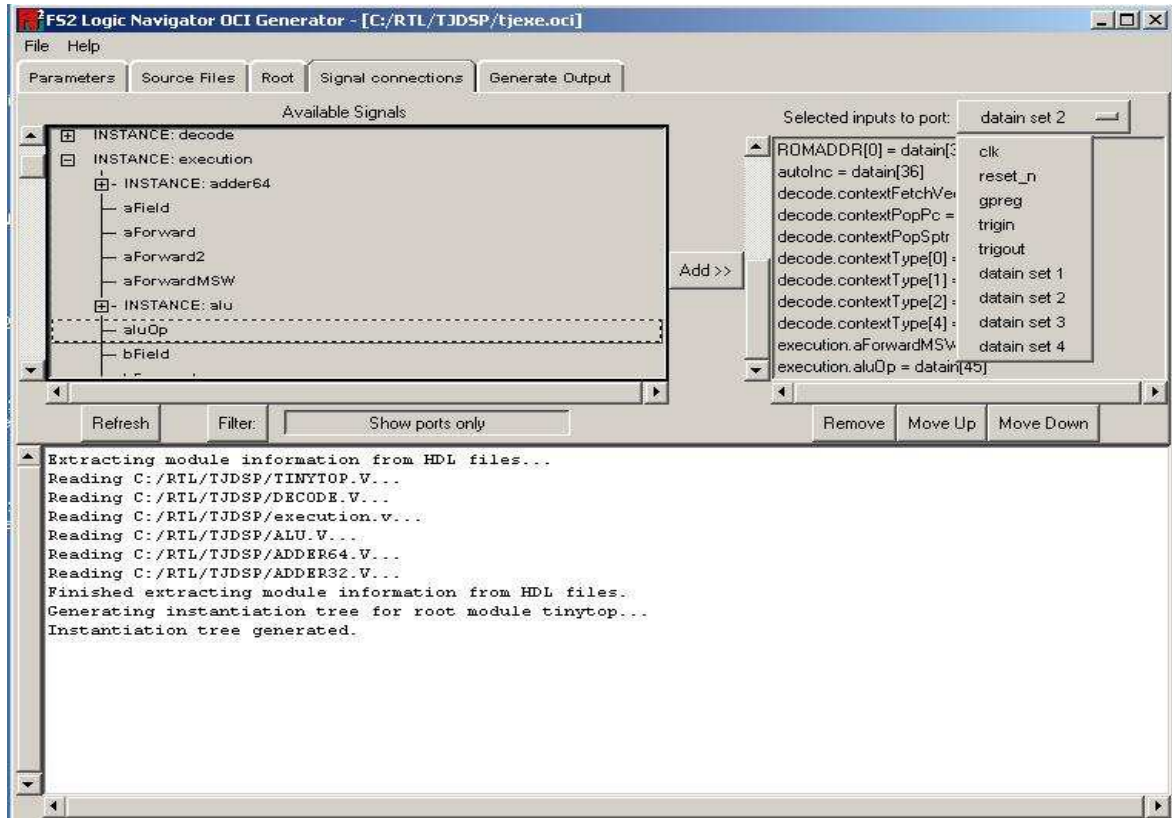


Figure 5 – Signal Connections page showing selected inputs pull down menu

In the Available Signals tree, the top level of the hierarchy represents the set of signals and instances that are directly contained within the root module as identified in step 4.4. The names of hierarchical instances (components) are preceded by the key word INSTANCE. Child instances, signal arrays, and signal vectors can be expanded into their contained elements by clicking on the (+) symbol to the left of the element's name. By expanding the tree in this fashion, signals at various hierarchical levels can be reached and selected for connection to the FS2 CLAM. Individual signals appear at the leaves of the tree, and cannot be expanded further.

The general flow of using the Signal Connections page is:

- 1) Using the pull down menu button labeled "Selected Inputs to port: "X"", the designer selects the CLAM interface to which he wishes to connect signal(s).
- 2) In the Available Signals tree, the designer locates and selects a signal by clicking on the appropriate item in the tree. Multiple contiguous items can be selected by clicking with the mouse while pressing the Shift key. Multiple noncontiguous items can be selected by clicking with the mouse while pressing the Ctrl key.
- 3) The designer clicks the "Add>>" button to add the selected signal(s) to the selected CLAM input.
- 4) Repeat at step 1 as required to define all of the signals connected to the CLAM block. The OCI Generator keeps track of the maximum number of signals for each CLAM interfaces, and does not allow addition of more signals for a given port than is define in the Parameter pages.

In the **Selected Inputs** list, signals may be removed using the **Remove** button, or reordered using the **Move Up** and **Move Down** buttons.

For larger designs the number of signals that are extracted from a design can be significant. The **Available Signals** tree can be optionally filtered to display only port level signals, only registered signals (this option is not supported for VHDL), signals matching a particular name with asterisk acting as a wildcard, or signal blocks delineated using an FS2 specific source pragma embedded in the HDL.

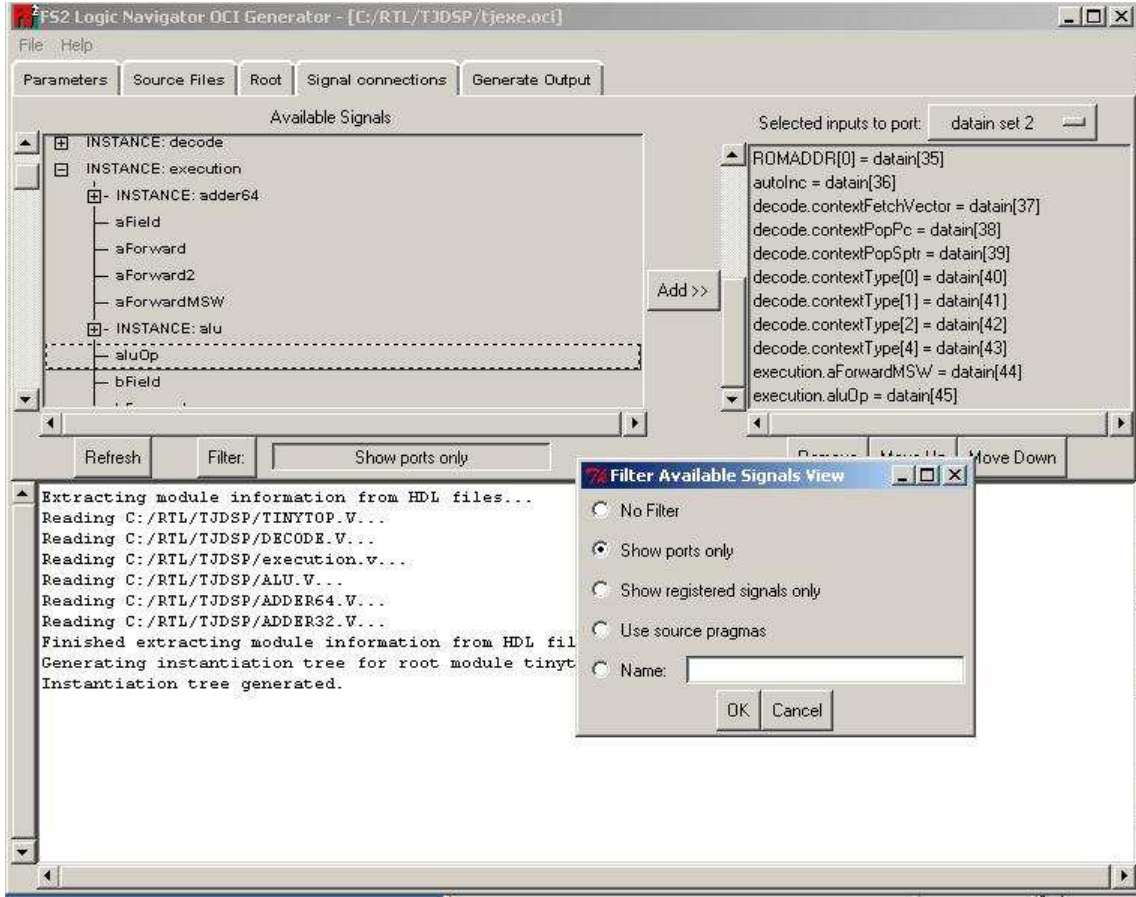


Figure 6 – Signal Connections page showing the Filtering menu

For both Verilog and VHDL, occurrences of the comment strings **logic_navigator pragma filter off** and **logic_navigator pragma filter on** within a valid HDL comment (// for Verilog, - - for VHDL) are used to control pragma based filtering. By default, the pragma setting is considered to be “on” at the top of each source file, so only signals bracketed by “filter off” and “filter on” pragmas will show up in the Available Signals tree when this particular filter mode is in effect.

As an example, in the following signal lists, only signals C3, D4, F6, and G7 would be included in the signal list

```
-- VHDL Example
Signal A1 : std_logic;
Signal B2 : std_logic;
-- logic_navigator pragma filter off
Signal C3 : std_logic;
Signal D4 : std_logic;
-- logic_navigator pragma filter on
Signal E5 : std_logic;
-- logic_navigator pragma filter off
Signal F6 : std_logic;
Signal G7 : std_logic;
```

```

-- logic_navigator pragma filter on
Signal H8 : std_logic;

-- Verilog Example
wire A1;
wire B2
-- logic_navigator pragma filter off
wire C3;
reg D4;
-- logic_navigator pragma filter on
reg E5;
-- logic_navigator pragma filter off
reg F6;
wire G7;
-- logic_navigator pragma filter on
reg H8;

```

5.6. Generating the Instrumented HDL Output

Figure 7 shows the “Generate Output” page of OCI Generator. All output files are written to the output directory appearing in the text field labeled “Directory for output files” (which defaults to the directory holding the OCI Generator project file). In addition, the polarity of the signal connection to the OCI reset input can be specified here; an inverter will be added as necessary to match the active-low expectation of the OCI. To generate the output files, click the “Go!” button.

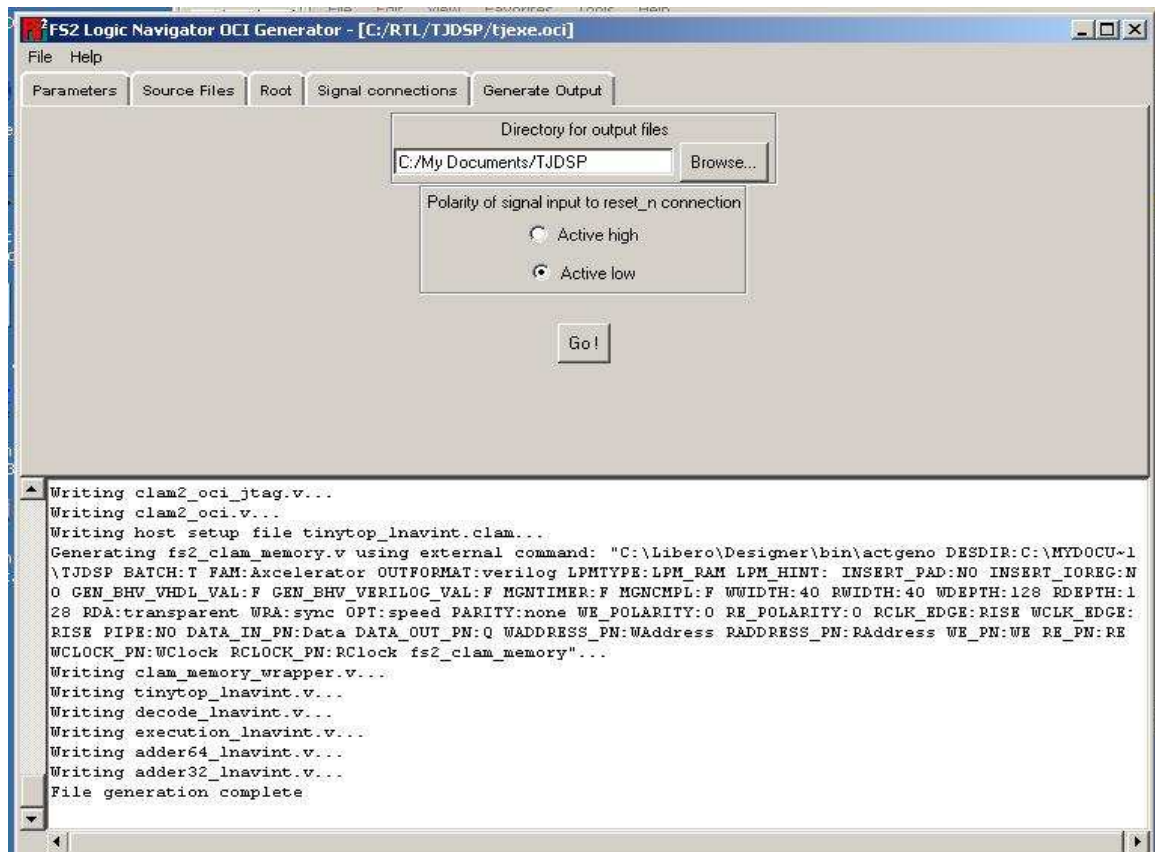


Figure 7 – Generate Output page

Figure 7 shows the Generate Output page after pressing the “Go!” button. After clicking the “Go!” button, it is important to pay attention to the status pane at the bottom of the window. If there are any problems generating the output files (e.g. a mandatory CLAM connection that was left unspecified), problems will be reflected in the status pane. Most problems can typically be resolved by adjusting information on the previous pages (tabs) and then returning to the Generate Output page and trying again.

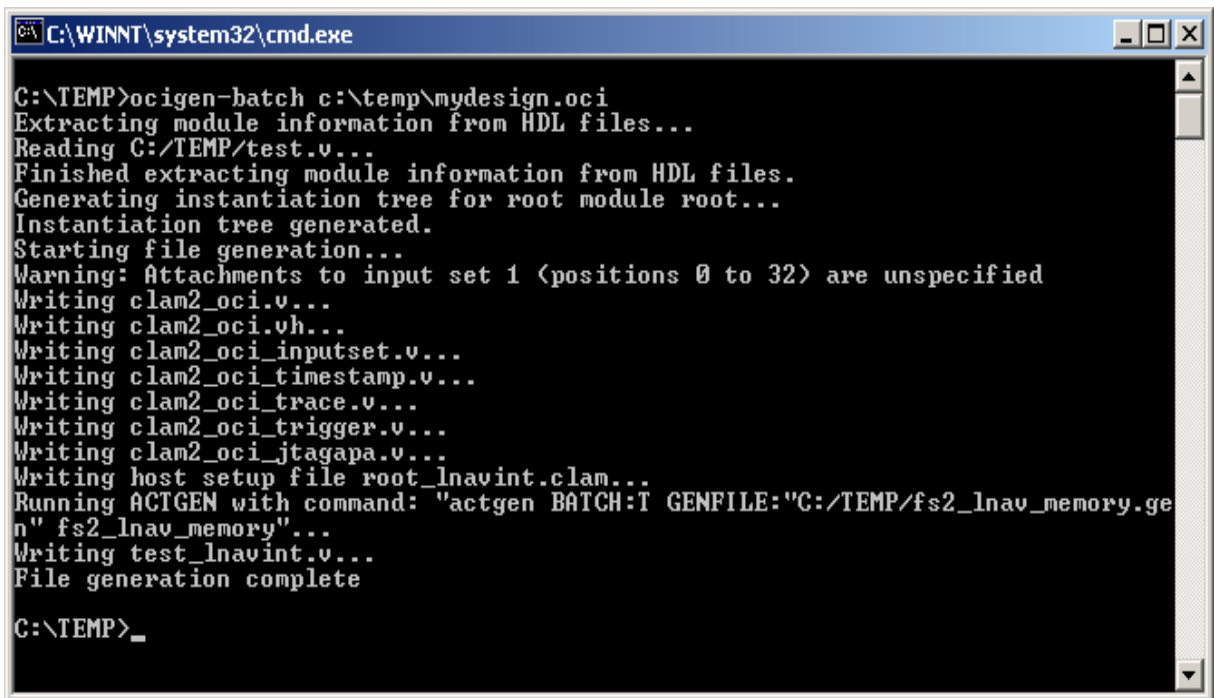
5.7. Saving the Project

It is highly recommended that you save your OCI Generator project for subsequent use. Save the project by using the Save Project option under the File menu. If you exit the OCI Generator without saving your project changes, a prompt will appear to offer another chance to save the project before exiting.

6. Running Batch Mode

Once a project file has been created and tested interactively in the OCI Generator, it is possible to re-generate the instrumented HDL output later in a batch context. This might be useful, for example, when incorporating the OCI generation step in automated build processes and/or command line environments. For this purpose, an executable named OCIGEN-BATCH.EXE is included with the installation, and the software installation modifies the system PATH environment variable to include the directory that holds this utility. OCIGEN-BATCH accepts a single command line parameter: an OCI Generator project file. When invoked in this way, OCIGEN-BATCH carries out the equivalent of the “Go!” button of the interactive OCI Generator, using the instantiation parameters, source files, and signal connections captured in the supplied project file.

Figure 8 shows a sample invocation of OCIGEN-BATCH within a command prompt window.



```
C:\WINNT\system32\cmd.exe
C:\TEMP>ocigen-batch c:\temp\mydesign.oci
Extracting module information from HDL files...
Reading C:/TEMP/test.v...
Finished extracting module information from HDL files.
Generating instantiation tree for root module root...
Instantiation tree generated.
Starting file generation...
Warning: Attachments to input set 1 (positions 0 to 32) are unspecified
Writing clam2_oci.v...
Writing clam2_oci.vh...
Writing clam2_oci_inputset.v...
Writing clam2_oci_timestamp.v...
Writing clam2_oci_trace.v...
Writing clam2_oci_trigger.v...
Writing clam2_oci_jtagapa.v...
Writing host setup file root_lnavint.clam...
Running ACTGEN with command: "actgen BATCH:T GENFILE:"C:/TEMP/fs2_lnav_memory.g
n" fs2_lnav_memory"...
Writing test_lnavint.v...
File generation complete

C:\TEMP>_
```

Figure 8 – Sample invocation of ocigen-batch

It is also possible to launch the interactive (GUI) version of the OCI Generator from the command line using the OCIGEN command. OCIGEN accepts an optional command line parameter, an OCI

Generator project file, which causes the OCI Generator to immediately load the specified project file (instead of requiring that the user open the project manually).

7. Characteristics of the Generated RTL file output

The OCI generator does not modify any of the design's existing source files. It generates one or more instrumented modules, each in its own output file for each module that includes a signal that is interfaced to the CLAM including any intermediate levels of hierarchy between the signal and the OCI root. An instrumented module is identical to the original module with the following exceptions:

- in the case of the root module, the corresponding generated module contains an instantiation of FS2 CLAM IP, an instantiation of a memory block used by CLAM (in the case of on-chip CLAM only), as well as administrative signals and structural assignments necessary to attach user signals to the CLAM.
- in the case of any non-top-level module that (either directly or indirectly through a chain of hierarchical child instantiation relationships) contains signals to be attached to FS2 CLAM, the corresponding generated module contains additional administrative signals, ports, and structural assignments necessary to propagate the traced signals to the root level of the hierarchy.

For on-chip CLAM, the name of the instrumented root module is the name of the corresponding non-instrumented root module with the suffix “_Inavint” appended to it. For off-chip CLAM, the name of the instrumented root module is the name of the corresponding non-instrumented root module with the suffix “_Inavext” appended to it.

In addition to files containing instrumented modules that implement the designer's connections to the CLAM IP, the OCI generator also places into the output directory the files that implement the FS2 CLAM IP itself. Any file generated by the OCI Generator, whether an instrumented module file or a file that is part of the FS2 CLAM IP should be included in synthesis configurations in order to implement a correction Logic Navigator OCI implementation. All the files generated by OCI generator will appear in the status pane of the OCI Generator window (as seen in Figure 7).

7.1. The .clam file output

The OCI generator also generates a .clam file that contains information about the OCI options and signals. This .clam file should be loaded to the Logic Navigator GUI prior to running trace operations with Logic Navigator to configure the user interface with the descriptive signal names used in the RTL files. If the .clam file is not loaded, all signals will take on a generic name (channel 0, channel 1, etc.). It is not recommended that the .clam file be manually edited.

Appendix A – connection points for on-chip CLAM IP

Name	Width	Direction	Description
clk	1	In	Drive this port with the system clock. All registers within the CLAM module are clocked on the positive edge of this clock signal.
reset_n	1	In	Drive this port with an active low reset signal. This is an optional connection. The registers and state machines inside the CLAM are designed to reach known values after a few clock cycles without the use of a reset signal. If a user signal is not selected for connection, then the OCI generator will drive this port with a high signal.
datain	Determined by the trace width and the number of input sets selected by the designer.	In	These are the signals that the designer wishes to capture when using the CLAM. In the OCI Generator software, assignment to each input set is made separately. The OCI Generator does not enforce a requirement that all of these inputs be associated with user signals, but in general, the designer will want to specify connections for these.
trigout	0 to 32, as determined by the number of signals assigned to this connection by the designer.	Out	These signals are driven individually by “trigout pulse” actions within the CLAM trigger system.
trigin	0 to 32, as determined by the number of signals assigned to this connection by the designer.	In	These active high signals connect to the CLAM trigger system, allowing them to be used in building complex triggering scenarios.
gpreg	0 to 32, as determined by the number of signals assigned to this connection by the designer.	Out	This is a general purpose output register that can be connected to whatever signals that the designer wishes. The CLAM host software (through the hardware used to connect to the target device) provides a mechanism to query and set the value of this register.

Appendix B – connection points for off-chip CLAM IP

Name	Width	Direction	Description
clk	1	In	Drive this port with the system clock. All registers within the CLAM module are clocked on the positive edge of this clock signal.
reset_n	1	In	Drive this port with an active low reset signal. This is an optional connection. The registers and state machines inside the CLAM are designed to reach known values after a few clock cycles without the use of a reset signal. If a user signal is not selected for connection, then the OCI generator will drive this port with a high signal.
datain	Determined by the trace width and the number of input sets selected by the designer.	In	These are the signals that the designer wishes to capture when using the CLAM. In the OCI Generator software, assignment to each input set is made separately. The OCI Generator does not enforce a requirement that all of these inputs be associated with user signals, but in general, the designer will want to specify connections for these.
clk2x	1	In	This port appears only on the 2x16 configuration of off-chip CLAM. It is a clock signal generated by a PLL which is 2 times the frequency of clk.
clk4x	1	In	This port appears only on the 4x8 configuration of off-chip CLAM. It is a clock signal generated by a PLL which is 4 times the frequency of clk.