

FS2 On-Chip Logic Navigator™ Instantiation, Simulation and Synthesis Guide

for Actel ProASIC^{Plus} and AX Devices

March 24, 2005



First Silicon Solutions, Inc.
4000 SW Kruse Way Place
Building 3, Suite 210
Lake Oswego, OR 97035
(503) 489-0311
Fax (503) 489-0315
<http://www.fs2.com>
sales@fs2.com
support@fs2.com

Table of Contents

1. PURPOSE	3
2. INTERNAL LOGIC NAVIGATOR DESCRIPTION	3
2.1. INTERNAL LOGIC NAVIGATOR PORT DESCRIPTIONS	4
2.2. INTERNAL LOGIC NAVIGATOR PARAMETER DESCRIPTIONS	6
3. INTERNAL LOGIC NAVIGATOR VERILOG DESIGN FLOW	8
3.1. INSTANTIATING THE INTERNAL LOGIC NAVIGATOR IN A VERILOG DESIGN	9
3.2. BEHAVIORAL SIMULATION.....	10
3.3. SYNTHESIS	11
3.4. FUNCTIONAL SIMULATION.....	12
4. INTERNAL LOGIC NAVIGATOR VHDL DESIGN FLOW	12
4.1. INSTANTIATING THE INTERNAL LOGIC NAVIGATOR IN A VHDL DESIGN	13
4.2. BEHAVIORAL SIMULATION.....	16
4.3. SYNTHESIS	16
4.4. FUNCTIONAL SIMULATION.....	17
APPENDIX A – TIMING REQUIREMENTS	18
1. LOGIC NAVIGATOR IP TIMING SPECIFICATIONS	18
2. INTERNAL LOGIC NAVIGATOR IP TIMING CONSTRAINTS.....	18
APPENDIX B – AREA REQUIREMENTS	19
INTERNAL LOGIC NAVIGATOR AREA REQUIREMENTS	19
APPENDIX C – PIN REQUIREMENTS	20
VOLTAGE LEVEL OF PINS INTERFACING TO THE FS2 PROBE.....	20
APPENDIX D – EXAMPLE TOP-LEVEL DESIGNS.....	21
1. EXAMPLE VERILOG DESIGN FOR INTERNAL LOGIC NAVIGATOR REFERENCE DESIGNS	21
2. EXAMPLE VHDL DESIGN FOR INTERNAL LOGIC NAVIGATOR REFERENCE DESIGNS	21
3. STP FILES FOR INTERNAL LOGIC NAVIGATOR REFERENCE DESIGN.....	21

1. Purpose

The purpose of this document is to provide aid in Logic Navigator trace and debug RTL integration to a designer working with Actel parts. Logic Navigator is an on-chip instrumentation based trace and debug tool based on FS2 Configurable Logic Analysis Monitor IP and tools. On-Chip Logic Navigator usage based on both Verilog and VHDL design flows are covered in the document. The On-Chip Logic Navigator™ is part of the FS2 Navigator® Suite of Tools and has been optimized to work with both Actel ProASIC^{Plus} and related flash based FPGAs and with high end Actel antifuse based (notably AX) FPGAs. The Logic Navigator IP routes internal signals (defined in HDL code of an application design implemented in an Actel FPGA) through a JTAG port connected to a probe. The probe interfaces to a trace display GUI where the signals can be examined. The routing and integration of Logic Navigator into an application design is automated through the use of a Logic Navigator OCI Generator tool which is discussed in the *FS2 Logic Navigator™ OCI Generator User Guide* and is included as part of this product release. This document can also be used to support manual instantiation and integration of Logic Navigator IP blocks into an Actel design.

In section 2, the Internal Logic Navigator IP is discussed, including a complete description of the ports and parameters for the IP blocks. The Internal Logic Navigator buffers and controls trace data on-chip and exports trace data through a serial JTAG port. In section 3 a description of the Verilog design flow is discussed including specific simulation examples using the Model Technology ModelSim simulator. The sub-section on synthesis includes examples using Simplicity Synplify. Corresponding use of the Internal Logic Navigator IP through a VHDL design flow is discussed in section 4.

Note: “On-Chip” and “internal” are interchangeable terms in this document.

2. Internal Logic Navigator Description

The Internal Logic Navigator is a multi-function on-chip logic analyzer, which includes event recognition logic, on-chip memory to hold trace data, and control logic to start, stop and read the trace memory. The Internal Logic Navigator communicates off chip using the device JTAG port to set up a serial link.

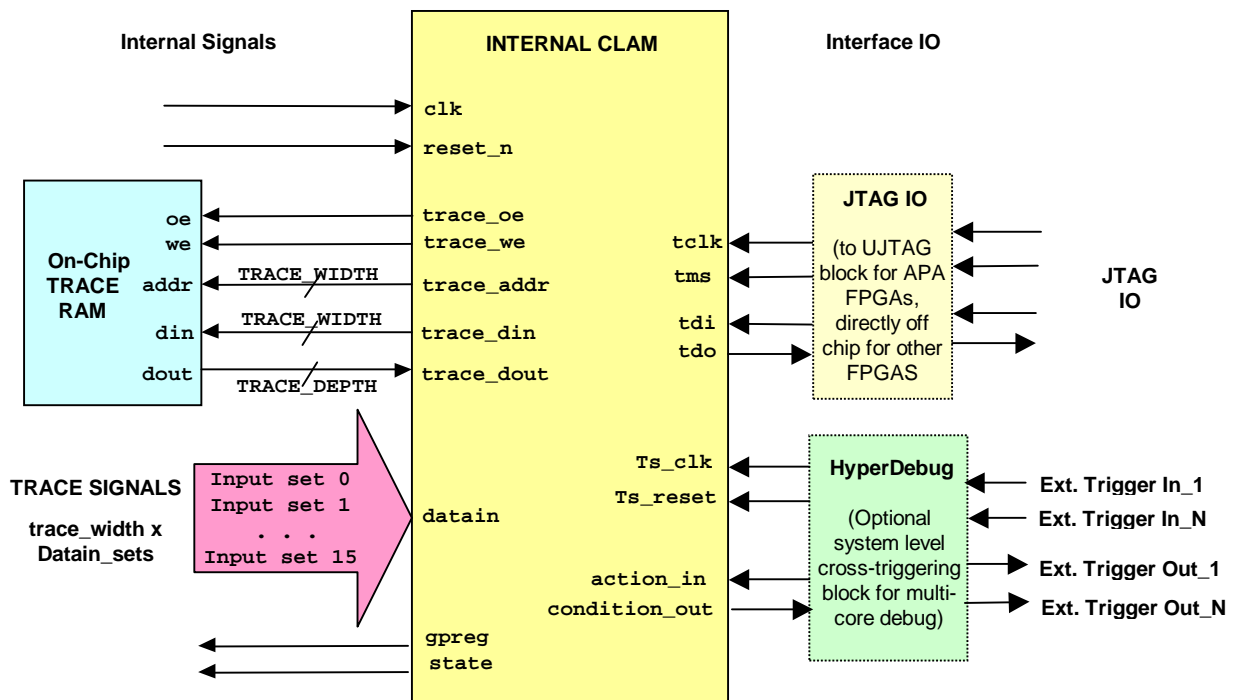


Figure 3. Diagram of Internal Logic Navigator IP

The Internal Logic Navigator can be implemented in two configurations, which differ only in the type of JTAG interface that is available on the FPGA.

1. For APA and other FLASH based Actel FPGAs with an integrated, internally accessible JTAG port, the Internal Logic Navigator includes a JTAG interface that connects to the APA UJTAG logic and IO port.
2. For devices that do not have an internally accessible JTAG port interface (this includes ACTEL antifuse parts, and AX family) the Internal Logic Navigator provides a fully functional (secondary) JTAG interface which includes a TAP controller.

The Actel FlashPro JTAG Probe used for programming APA and other FLASH based parts can also be used to configure the Internal Logic Navigator and to route data from the on-chip trace to the FS2 host. The FlashPro JTAG Probe can also be used with antifuse parts (having internal memory blocks) as a probe to configure the Internal Logic Navigator and route data from the on-chip trace to the FS2 host.

2.1. Internal Logic Navigator Port Descriptions

Table 1 contains a complete list of the ports on the Internal Logic Navigator design along with a description of each.

Table 1. Internal Logic Navigator Port descriptions

Name	Width	Direction	Description
clk	1	In	Drive this port with the system clock. All registers within the Internal Logic Navigator module are clocked on the positive edge of this clock signal.
reset_n	1	In	Drive this port with an active low reset signal.
datain	datain_width * datain_sets	In	Trace data input port driven from the user's design. The actual width of this port is determined by two parameter values: datain_width and datain_sets.
tclk	1	In	External JTAG clock driven to the chip by the Logic Navigator Probe.
tdi	1	In	External JTAG Serial data input from the Probe.
tdo	1	Out	External JTAG Serial data output to the Probe.
tms	1	In	External JTAG TAP State Machine selector
trace_addr	trace_depth	In	Trace memory port driving the address lines of the on-chip trace memory. The width of this port is determined by the TRACE_DEPTH parameter.
trace_din	trace_width	Out	Trace memory port driving the data in lines of the on-chip trace memory. Its width is determined by the TRACE_WIDTH parameter plus two status bits.
trace_dout	trace_width	In	Trace memory port driven by the data out of the on-chip trace memory. Its width is determined by the TRACE_WIDTH parameter plus two status bits.
trace_oe	1	Out	Trace memory port driving the output (read) enable of the trace memory
trace_we	1	Out	Trace memory port driving the write enable port of the trace memory
clam2_state	2	Out	Operational state of Logic Navigator system 00 = Logic Navigator IP has stopped 01 = Logic Navigator IP has started 10 = Logic Navigator IP running 11 = Event has triggered and trace is postfilling

action_in	hdnodes	In	This port is driven by external (off chip) actions
condition_out	hdnodes	Out	This port is driven high when a trigger event is recognized. It may be directly routed off-chip to the probe for use with other test equipment
Ts_clk	1	In	This port is clk for the timestamp counter (width of which is defined by parameter ts_width)
Ts_reset	1	In	This port is the reset signal for timestamp counter
gpreg	gpreg_width	Out	This port is general-purpose output register that can be set through JTAG.

The Logic Navigator reset_n port is an active low reset, which is used to set registers inside the Logic Navigator IP. If the user's design does not contain a reset signal, the registers and state machines inside the Logic Navigator IP are designed to reach known values after a few clock cycles (10 max) without the use of a reset signal. This works well in physical silicon, however simulators still require the reset signal to be asserted to bring state machines inside the Logic Navigator into a known state for simulation. If there is no reset signal in the top-level design it is necessary to create one in the designer's testbench that is used for simulation only. The reset_n port can be driven with a constant value of '1' for synthesis. OCI Generator users can select reset interfaces between Logic Navigator and the rest of the system when they generate instrumented RTL files.

The designer should drive the datain port with the signals that are to be traced. The size of the datain port is determined by the value of two parameters passed into the design. (A description of all parameters in the design is included in table 6.) The TRACE_WIDTH parameter indicates the number of bits in each data set routed to the Logic Navigator IP. The INPUT_SETS parameter indicates the number of data sets routed to the Logic Navigator. The Logic Navigator IP will automatically set the size of the datain port equal to (TRACE_WIDTH * INPUT_SETS).

The trace_addr, trace_din, trace_dout, trace_we, and trace_oe ports are used to connect the Internal Logic Navigator IP to the memory block instantiation that the designer will use for trace memory. The designer can use the ACTgen tool to create a RAM block of the desired proportions and then instance the RAM block the design. The memory can be connected to the Logic Navigator IP as shown in Figure 3.

The clam2_state port is an output of Logic Navigator block and provides status information on Logic Navigator operation.

- 00 = Logic Navigator stopped
- 01 = Logic Navigator starting
- 10 = Logic Navigator running
- 11 = Logic Navigator has triggered and is postfilling

action_in and condition_out ports are respectively trigger in and trigger out ports whose size is determined by the hdnodes parameter. Up to 32 trigger in and trigger out ports can be instantiated. Trigger in and trigger out features may not be supported in evaluation licenses. Contact FS2 for details.

Ts_clk and Ts_reset ports are inputs to the Logic Navigator timestamp logic. If the ts_width = 0, they are not implemented, for any other ts_width value they are implemented. Ts_clk and Ts_reset signals are driven by on-chip clk and reset signals.

Gpreg port is a JTAG controlled output register whose size is defined by gpreg_width parameter. When gpreg_width parameter = 0, Gpreg is not implemented. The gpreg feature may not be supported in evaluation licenses. Contact FS2 for details.

2.2. Internal Logic Navigator Parameter Descriptions

Table 2 contains a complete list of the parameters used in the on-Chip Logic Navigator design along with a description of each.

Table 2. Internal Logic Navigator Parameter Descriptions

Name	Default Value	Valid Range	Description
datain_width	32	1 to 256	Indicates the size of each trace data set routed to the datain port.
trace_depth	10	4 to 16	Sets the width of the trace_addr port. The trace memory instantiated by the designer should support 2 ^{TRACE_DEPTH} memory locations.
datain_sets	1	1 to 16	Indicates the number of data sets routed to the Logic Navigator IP on the datain port.
hdnodes	1	0 to 32	Number of (optional) external trigger interfaces. Each interface node pair has action_in (trigger input) and condition_out (trigger output) signals that may connect directly to off-chip signals, on-chip signals, or to an FS2 HyperDebug node cross-triggering IP block (contact FS2 for details on HyperDebug). This optional logic is removed by synthesis when set to 0.
event_width	2	1, 2, 3, or 4	Size of event recognizer index corresponding to 2, 4, 8, or 16 event recognizers available for triggering
state_width	2	1, 2, 3, or 4	number of bits in the trigger sequencer state index corresponding to 2, 4, 8, or 16 states for sequential event detection and sequential trigger control
ts_width	32	0 to 63	width of (optional) timestamp implemented – size of the timestamp counter implemented. This optional logic is removed by the synthesis tool when set to 0.
counters	2	0,1,2	Defines number of 32 bit counters used for trigger operations. This optional logic is removed by the synthesis tool when set to 0.
gpreg_width	0	0 to 32	width of general purpose register output. This optional logic is removed by the synthesis tool when set to 0.
clam2_optimize	1	0/1	Allows tradeoff between in a non-pipelined (smaller but slower) debug critical path (0) and pipelined (larger but faster) debug critical path (1) in implementation
edge_events	0	0/1	Support edge detection in events (1). If (0), then provides logic level-only event detection, which requires fewer gates.

The following 5 parameters are required and cannot be set to 0.

The “trace_width” parameter is used to set the width of each data set that is connected to the Logic Navigator IP datain port. The minimum trace_width parameter value is 1; that is, one signal is traced per each input set defined.

The “datain_sets” parameter indicates how many separate data sets the Logic Navigator datain port consists of. Only one data set is traced for a given trace run. Together with trace_width, the number of input sets determines the overall width on the Logic Navigator IP datain ports.

The “trace_depth” parameter determines the Logic Navigator width of the trace_addr port for on-chip trace memory. The memory block instantiated in the designer’s code will need to support this address width.

The “event_width” parameter sets the number of the event recognition trigger blocks inside the Logic Navigator IP. The number of trigger events is defined as $2^{\text{event_width}}$. The minimum number of trigger events in Logic Navigator is 2, (event_width parameter = 1). Up to 16 trigger events can be specified (event_width = 4).

The “state_width” parameter enables sequential event trigger logic as part of Logic Navigator event detection and triggering. The number of states is $2^{\text{state_width}}$. The minimum number of states in Logic Navigator is 2 (state_width parameter = 1). Up to 16 states can be specified (state_width = 4).

The following 4 parameters are optional and can be set to 0.

The “ts_width” parameter defines the length of timestamp counter that is supported for trace. The timestamp is appended to data being traced, so in addition to logic used and timing considerations for a very long counter, the memory requirement to implement the counter should be considered. The counter width supported can vary from 0 bits (in which case, no timestamping is implemented) to 63 bits in length. The timestamp counter is driven by an on-chip clock that is typically the highest frequency clock in a given design. This is optional logic in the Logic Navigator and is not implemented when ts_width = 0.

The “counters” parameter defines the number (from 0 to 2) of 32 bit counters implemented in the Logic Navigator IP. The counters are used for interval timing between events or for counting of events prior to triggering or during trace capture. This is optional logic in the Logic Navigator and is not included when counters=0.

The “gpreg_width” parameter defines the size of a general-purpose register that is included in Logic Navigator. The register value is set via JTAG and can be used to control any on-chip user logic or set static values. For example, it could be used to set one of several test modes or set a constant for a test. It cannot be updated by on-chip user logic. This is optional logic in the Logic Navigator and is not included when gpreg_width = 0. The gpreg feature may not be supported in evaluation licenses.

The “hdnodes” parameter defines the number of external trigger pair interfaces that are implemented in the Logic Navigator IP. Each interface node pair has an action_in (trigger input) and condition_out (trigger output) signal. This is optional logic in the Logic Navigator and is not included when hdnodes = 0. The hdnodes feature may not be supported in evaluation licenses.

The following 2 (edge_events and clam2_optimize switch) parameters allow user defined tradeoffs for Logic Navigator implementation and are set to 0 or 1

The “edge_events” parameter defines whether the event recognition adds or doesn’t add signal edge recognition (rising (0 then 1) or falling edge (1 then 0)) and double high/low recognition (input 0 or 1 for two cycles) to the standard level recognition. Edge detection is enabled with edge_events = 1. If edge_events = 0 then the event condition is based only on logic levels (high or low) for one clock cycle. Logic level-only event detection requires fewer gates than full edge recognition.

The “clam2_optimize” parameter defines whether a pipeline stage is inserted in the Logic Navigator critical path, which increases gate count by approximately 10%, but increases maximum operating speed by approximately 30%. The only operational difference between optimize values of 0 and 1 is when optimize is set to 1, the postfill amount can not be zero, due to the increased pipelining. If postfill is set to zero, trace will have one frame of postfill even if it is not required.

Note that when manually integrating Logic Navigator with clam2_optimize =1, a register stage must be added to the input of ts_rst signal to synchronize timestamp and trace. Examples of this are shown in Verilog code in section 3.1 and VHDL example in section 4.1.

3. Internal Logic Navigator Verilog Design Flow

Table 3 gives a complete listing of the files included in the Internal Logic Navigator Verilog design flow. Since the Internal Logic Navigator supports both APA FPGAs (that have an inherent UJTAG port) and Actel Antifuse FPGAs (that do not have a JTAG port), the flow has two corresponding variants which will be discussed. Note that Internal Logic Navigator only supports Actel Antifuse with embedded on-chip memory resources.

Table 3. Internal Logic Navigator Verilog Files

Name	Description
clam2_oci_defs.v	Common definitions for Logic Navigator IP
clam2_oci.v	Contains top level definition of the internal Logic Navigator module.
clam2_oci_timestamp.v	Logic Navigator timestamp generator sub-module to clam2_oci
clam2_oci_inputset.v	Logic Navigator select input set sub-module to clam2_oci
clam2_oci_jtag.v	Logic Navigator OCI JTAG TAP sub-module to clam2_oci – used to add JTAG TAP for antifuse design flow
clam2_oci_jtagapa.v	Logic Navigator OCI JTAG TAP sub-module to clam2_oci- used for ProASIC through UJTAG interface
clam2_oci_trace.v	Logic Navigator trace manager sub-module to clam2_oci
clam2_oci_trigger.v	Logic Navigator trigger manager sub-module to clam2_oci
jtagnav_int.v	target-side reference design for general Logic Navigator implementation (including TAP controller)
apalnav_int.v	target-side reference design for APA specific Logic Navigator implementation (ProASIC ^{Plus} device) top level module for the Actel APA evaluation board, both APA075 and APA300 versions.
apalnav_int_isp.v	This is the toplevel module for the Actel ISP eval board
clam2_oci_tb.v	(Optional) It contains a testbench suitable for stimulating the Internal Logic Navigator by itself.
simulation script files	
simclam2apa.tcl	TCL script for Logic Navigator, APA UJTAG version for use with ModelSim simulators. It compiles Verilog source files for APA configuration and then loads the testbench and starts a simulation. This script should be edited to match the design directory structure
simclam2.tcl	TCL script driver for Logic Navigator, JTAG version for use with ModelSim simulators. It compiles Verilog source files for antifuse configuration and then loads the testbench and starts a simulation. This script should be edited to match the design directory structure
Memory support files	
mem_wrapper_behavioral.v	A generic behavioral description of on-chip memory, used for behavioral simulation. It accepts the trace_width and trace_depth parameters, so it can work with any memory configuration.
mem_wrapper_proasic.v wrapper_apa.v	This file contains Actgen ProASIC ^{Plus} memory block output for the trace memory for apalnav_int and apalnav_int_isp designs, configured to width of 34 bits (trace_width = 32) and depth of 256 words (trace_depth = 8).
mem_wrapper_ax.v	This file contains an example instantiation of an Actel AX memory block, to width of 34 bits (trace_width = 32)

3.1. Instantiating the Internal Logic Navigator in a Verilog Design

The Logic Navigator OCI Generator tool will automatically create Verilog code to interface the Internal Logic Navigator subsystem in a user design, with all appropriate parameters and trace connections as defined by the designer through the OCI Generator GUI.

If manually integrating the Logic Navigator IP, interface the top level clam2_oci module in your design, similar to any other instance of an IP block. If the target FPGA is APA, the file clam2_oci_jtagapa.v should be precompiled in the design. If the target FPGA is not APA, the file clam2_oci_jtag.v should be precompiled in the design. Both files contain a module named clam2_oci_jtag. This allows a common set of top level models to support either target FPGA, but does require that the designer pay attention that the correct version of the clam2_oci_jtag module is implemented. Use of the OCI Generator ensures that the proper module is included in the design.

The APA design flow assumes the inclusion of the Actel provided apa.v library file in a design. The apa.v include file includes a module for the UJTAG block, which provides the JTAG IO for APA parts. The antifuse design flow has a full JTAG Test Access Port (TAP) in the clam2.jtag.v file and does not depend on any Actel provided models.

The designer will need to ensure that the proper parameters with the correct values for the application are included. The parameters are independent of whether APA or Antifuse parts are being used. Below is an example of setting parameters for trace width to 32 and other parameters set as shown in the comments.

```
// example instantiation of internal Logic Navigator for case of
// clam2_datain_width = 32      clam2_datain_sets = 2
// clam2_trace_depth = 7      clam2_hdnodes = 3
// clam2_event_width = 1      clam2_state_width = 4
// clam2_ts_width = 28        clam2_counters = 1
// clam2_gpreg_width = 8      clam2_edge_events = 0
// clam2_optimize = 1

// *****
// // Remove comments and use this reg. only if clam2_optimize = 1
// always @(posedge clam_connection_clk) begin
//     clam_connection_ts_reset <= ~clam_connection_state[1];
// end
// *****

clam2_oci #(32, 2, 7, 3, 1, 4, 28, 1, 8, 0, 1) clamint_inst
( .reset_n(clam_connection_rst),
  .clk(clam_connection_clk),
  .tck(tck),
  .tms(tms),
  .tdi(tdi),
  .tdo(tdoclam2),
  .datain(datain[63:0]),      // 2 input sets * 32 bit datain
  .ts_clk(clk),
  .ts_reset(~clam2_state[1]), // use only if clam2_optimize = 0
// .ts_reset (clam_connection_ts_reset)// use only if clam2_optimize = 1
  .clam2_trace_addr(trace_addr[6:0]), // trace depth
  .clam2_trace_din(trace_din[31:0]), // trace width in
  .clam2_trace_dout(trace_dout[31:0]), // trace width out
  .clam2_trace_we(trace_we), // trace write enable
  .clam2_trace_oe(trace_oe), // trace read enable
  .clam2_action_in(clam2_action_in[2:0]),
  .clam2_condition_out(clam2_condition_out[2:0]),
  .clam2_gpreg(clam2_gpreg[7:0]),
  .clam2_state(clam2_state)
);
```

The OCI Generator tool automatically generates batch script files for ACTgen that are used to create the memory block and its appropriate wrappers. If OCI Generator is not used, the designer must create an instance of a memory block of the correct size for the application and connect the memory to the Logic Navigator IP (using declared wires). The designer can use the ACTgen tool to create a memory of the desired width and depth. In ACTgen set read access to “Sync Transp”, write access to “Sync”, optimization to “Speed” and Parity to “None”. The designer may wish to create a “wrapper” module, which hides the actual RAM port definitions for behavioral and structural versions of the RAM code from the design. The mem_wrapper files in this release demonstrate how this is implemented. The following example of the mem_wrapper module instantiation can be used with any memory model.

```
// example instantiation of trace memory
mem_wrapper mem_wrapper_inst (
    .addr(trace_addr),
    .clk(clk),
    .din(trace_din),
    .dout(trace_dout),
    .we(trace_we),
    .we(trace_oe)
);
```

An example design which instances the Internal Logic Navigator IP is described In Appendix D.

3.2. Behavioral Simulation

To simulate your design with a simulator such as ModelSim, include the following files in your project similar to any of your other design files and compile them.

```
clam2_oci_tb.v
clam2_oci.v
clam2_oci_timestamp.v
clam2_oci_inputset.v
clam2_oci_jtagapa.v (for APA FPGAs) or clam2_oci_jtag.v
                    (for Antifuse FPGAs)
clam2_oci_trace.v
clam2_oci_trigger.v
clam_oci_defs.v
mem_wrapper_behavioral.v (or equivalent form Actgen)
apa.v (for APA FPGAs - this includes UJTAG module for simulation)
```

clam2_oci.v and some of its sub modules contains a ‘include statement for the clam2_oci_defs.v file, which contains Logic Navigator definitions. If the simulation directory is different from the source file directory you will have to include a command line option on the compile command (or set the option with the ModelSim GUI) so that ModelSim can locate the clam2_oci_defs.v file for inclusion. The compile command from the ModelSim prompt for clam2_oci.v would look something like this:

```
vlog -reportprogress 300 -work work \
+incdir+C:/source/file/directory \
{C:/source/file/directory/ clam2_oci.v}
```

Remember, the reset_n port must be pulsed low at the start of simulation by your testbench in order for proper Logic Navigator simulation.

TCL scripts that direct compilation and execution design flows of Internal Logic Navigator for several sets of parameters are included in the Logic Navigator release. The simclam2apa.tcl file supports the APA design flow and simclam2.tcl supports simulation of the Antifuse design flow. Directory definitions in the TCL files should be modified to the directory structure used for the given design.

3.3. Synthesis

To synthesize Logic Navigator IP for your design with a synthesis tool such as Synplify, include the following files in your project similar to any of your other design files and proceed with your synthesis as usual. `Clam2_oci_defs.v` need not be included in the file list as it is automatically included by the include statement inside `clam2_oci.v`.

```
clam2_oci.v
clam2_oci_timestamp.v
clam2_oci_inputset.v
clam2_oci_jtagapa.v (for APA FPGAs) or clam2_oci_jtag.v (for Antifuse
FPGAs)
- Do not synthesize both clam2_oci_jtagapa.v and clam2_oci_jtag.v in a
given design. Since they have same entity name, they can overwrite each
other.
clam2_oci_trace.v
clam2_oci_trigger.v
clam_oci_defs.v
mem_wrapper.v (this file is created by OCI Generator or the designer)
```

The OCI generator creates a model named `your_design_inav.v` which integrates a module and mapping of the Logic Navigator subsystem IP to the top level target design file that is named `your_design.v`. `Your_design_inav.v` file and any associated files (that may have trace mapping) should be synthesized instead of `your_design.v` and its associated files to include Logic Navigator debug IP to be implemented.

A `mem_wrapper_proasic.v` file is provided as an example which instances an Actel ProASIC^{Plus} memory block created by the ACTgen tool with the trace depth set to 128 words and the memory width set to 34 bits. The designer will need to create a `mem_wrapper` file similar to this which implements an (APA or Antifuse) Actel memory block for the design application that is compatible with the memory width and depth defined as parameters in the `clam2_oci` module.

In cases where Logic Navigator is interfacing to Actel UJTAG port, the synthesis tools may not recognize that UJTAG TDO is an output. Synplicity, as an example, assumes that modules that produce no outputs have no effect and can therefore be pruned.

To force Synplicity to keep the `clam2_oci` module, two directives should be added to the module declaration of `clam2_oci`. These are only needed for On-Chip Logic Navigator APA implementation since only Actel APA uses the UJTAG macro. The OCI Generator will automatically add these directives for On-Chip Logic Navigator APA implementations.

For manual integration in an APA design, the directives should be added to `clam2_oci.v` as follows:

```
...
// General purpose register output
clam2_gpreg,
// Operational state of CLAM system
clam2_state
)
// Add these directives after ) and before ; of the module definition
/* synthesis syn_hier="hard" */
/* synthesis syn_noprune=1 */
;
// clam2 implementation options
```

3.4. Functional Simulation

Note: information on functional simulation is included for completeness; simulating the synthesized netlist does not generally need to be done.

If you wish to simulate your synthesized netlist with a simulator such as ModelSim you will first need to compile the Actel primitives into a library. The following sequence of commands will create a library and compile the Actel primitives for the ProASIC^{Plus} family into it. From the ModelSim prompt:

```
cd c:/your/full/path
vlib apa_ver
vmap apa_ver C:/your/full/path/apa_ver
vlog -reportprogress 300 -work apa_ver \
{C:/libero/designer/lib/vlog/apa.v}
```

A similar design flow to the above with different libraries would need to be implemented for other FPGA types.

Next compile the synthesized design into your work library. Designer will create a synthesized Verilog netlist with the same name as your top-level file with the extension in a directory called "designer" below your project directory. An example command for ModelSim would be:

```
vlog -reportprogress 300 -work work \
{C:/your/project/directory/designer/your_design_name.v}
```

Now, load the testbench into ModelSim with a command similar to this:

```
vsim -L C:/ your/full/path/apa_ver work.your_test_bench
```

Begin your simulation as previously discussed.

4. Internal Logic Navigator VHDL Design Flow

Table 4 gives a complete listing of the files included in the Internal Logic Navigator VHDL design flow.

Table 4. Internal Logic Navigator VHDL Files

Name	Description
clam2_oci_defs.vhd	Library element file containing constant definitions for Internal Logic Navigator
clam2_oci.vhd	Contains the top level entity and architecture of the internal Logic Navigator.
clam2_oci_timestamp.vhd	Logic Navigator timestamp generator entity and architecture - which is component to clam2_oci
clam2_oci_inputset.vhd	Logic Navigator input set entity and architecture - which is component to clam2_oci
clam2_oci_jtag.vhd	Logic Navigator OCI JTAG TAP entity and (behavioral) architecture - component to clam2_oci to add JTAG TAP for antifuse design flow.
clam2_oci_jtagapa.vhd	Logic Navigator IP OCI JTAG TAP entity and (behavioral_apa) architecture component to clam2_oci- for ProASIC (with UJTAG interface) design flow
clam2_oci_trace.vhd	Logic Navigator trace manager entity and architecture - which is component to clam2_oci
clam2_oci_trigger.vhd	Logic Navigator trigger manager entity and architecture - which is component to clam2_oci

jtaglnav_int.vhd	Top level LNAV Reference design module for general clam2 (including TAP controller) implementation
apalnav_int_isp.vhd	Top level LNAV Reference design module for the Actel APA ISP evaluation board
apalnav_int.vhd	Top level LNAV Reference module for the Actel APA both APA075 and APA300 evaluation board,
clam2_oci_tb.vhd	(Optional) It contains a testbench suitable for stimulating the Internal Logic Navigator by itself.

Support Script Files

simclam2apavhdl.tcl	TCL script for Logic Navigator, APA UJTAG version for use with ModelSim simulators. It compiles VHDL source files for APA configuration, and then loads the testbench and starts a simulation. The script should be edited to match the designer's directory structure.
simclam2vhdl.tcl	TCL script driver for Logic Navigator, JTAG version for use with ModelSim simulators. It compiles VHDL source files for antifuse (non UJTAG), configuration, loads the testbench and starts a simulation. The script should be edited to match the designer's directory structure

Memory support files

mem_wrapper_behav.vhd	A generic behavioral description of on-chip memory, used for behavioral simulation. It accepts the trace_width and trace_depth parameters, so it can work with any memory configuration.
mem_wrapper_proasic.vhd	(This file contains Actgen ProASIC ^{Plus} memory block output for the trace memory for apalnav_int and apalnav_int_isp designs, configured to width of 34 bits (trace_width = 32) and depth of 256 words (trace_depth = 8).
mem_wrapper_ax.vhd	This file contains an example instantiation of an Actel AX memory block, to width of 34 bits (trace_width = 32)

4.1. Instantiating the Internal Logic Navigator in a VHDL Design

The Logic Navigator OCI Generator tool will automatically create VHDL code to interface the Internal Logic Navigator subsystem in a user design, with all appropriate generics and trace connections as defined by the designer through the OCI Generator GUI.

If manually integrating the Logic Navigator IP, interface the clam2_oci entity/component in your design, similar to any other IP block component. If the target FPGA is APA, the file clam2_oci_jtagapa.vhd should be precompiled in the design. This file contains entity named clam2_oci_jtag and an architecture named behavioral_apa. If the target FPGA is not APA, the file clam2_oci_jtag.v should be precompiled in the design. This file contains entity named clam2_oci_jtag and an architecture named behavioral_apa. This allows a common set of top level models to support either target FPGA, but does require that the designer pay attention that the correct version of the clam2_oci_jtag architecture is being implemented. Use of the OCI Generator ensures that the proper architecture is included in the design.

The APA design flow assumes the inclusion of the Actel provided comp.vhd file in a design. The comp.vhd file includes a component for the UJTAG block, which provides the JTAG IO for APA parts. The antifuse design flow has a full JTAG Test Access Port (TAP) in the clam2.jtag.vhd file, and does not depend on any Actel provided models.

The designer will need to ensure that the proper generics with the correct values for the application are included. The generic values are independent of whether APA or Antifuse parts are being used. Below is an example of setting generics for trace width to 32 and other parameters set as shown in the example. You will need to drive the generics with the correct values for your application.

```

-- example component declaration of internal Logic Navigator
component clamint is
generic (
    clam2_datain_width : integer := 32;
    clam2_datain_sets  : integer := 2;
    clam2_trace_depth  : integer := 7;
    clam2_hdnodes      : integer := 3;
    clam2_event_width  : integer := 1;
    clam2_state_width  : integer := 4;
    clam2_ts_width     : integer := 28;
    clam2_counters     : integer := 1;
    clam2_gpreg_width  : integer := 8;
    clam2_edge_events  : integer := 0;
    clam2_optimize     : integer := 1
);
port (
    reset_n           : in  std_logic;
    clk                : in  std_logic;
    tck                : in  std_logic;
    tms                : in  std_logic;
    tdi                : in  std_logic;
    tdo                : out std_logic;
    datain            : in  std_logic (trace_width*input_sets-1 downto 0);
    ts_clk            : in  std_logic;
    ts_reset          : in  std_logic;
    clam2_trace_addr  : out std_logic (trace_depth-1  downto 0);
    clam2_trace_din   : out std_logic (trace_width-1  downto 0);
    clam2_trace_dout  : in  std_logic (trace_width-1  downto 0);
    clam2_trace_we    : out std_logic;
    clam2_trace_oe    : out std_logic;
    clam2_action_in   : in  std_logic (hdnodes-1  downto 0);
    clam2_condition_out : out std_logic (hdnodes-1  downto 0);
    clam2_gpreg       : out std_logic (gpreg_wdith-1 downto 0);
    clam2_state       : out std_logic (1  downto 0);
);
end component;
. . .
begin
. . .
-- *****
-- Remove comments and use this process only if clam2_optimize = 1
-- process (clk) begin
--   if rising_edge(clk) then
--     ts_reset <= not clam2_state(1);
--     -- trace stop resets timestamp
--   end process;
-- *****

-- example instantiation of internal Logic Navigator
clamint_inst: clamint
generic map (
    clam2_datain_width => 32,
    clam2_datain_sets  => 2,
    clam2_trace_depth  => 7,
    clam2_hdnodes      => 3,
    clam2_event_width  => 1,
    clam2_state_width  => 4,
    clam2_ts_width     => 28,
    clam2_counters     => 1,
    clam2_gpreg_width  => 8,

```

```

    clam2_edge_events => 0,
    clam2_optimize   => 1
  )
  port map (
    reset_n          => reset_n,
    clk              => clk,
    tck              => tck,
    tms              => tms,
    tdi              => tdi,
    tdo              => tdoclam2,
    datain           => datain (63 downto 0), -- input sets x datain
    ts_clk           => clk,
    ts_reset         => not clam2_state(1), -- use if clam2_optimize = 0
    -- ts_reset      => ts_reset,         -- use if clam2_optimize = 1
    clam2_trace_addr => trace_addr(6 downto 0), -- trace depth
    clam2_trace_din  => trace_din(31 downto 0), -- trace width
    clam2_trace_dout => trace_dout(31 downto 0), -- trace width
    clam2_trace_we   => trace_we,
    clam2_trace_oe   => trace_oe,
    clam2_action_in  => action_in(2 downto 0), -- hdnodes
    clam2_condition_out => condition_out(2 downto 0), -- hdnodes
    clam2_gpreg      => clam2_gpreg(7 downto 0),
    clam2_state      => clam2_state(1 downto 0)
  );

```

The OCI Generator tool automatically generates batch script files for ACTgen that are used to create the memory block and its appropriate wrappers. If OCI Generator is not used, the designer must declare a component for and mapped instance of a memory block of the correct size for the application in the design and connect the memory to the Logic Navigator (using declared signals). A good way to do this is to create a “wrapper” module which hides the actual RAM port definitions from the design. The benefit of doing things this way is that it can require fewer changes to the main code to switch to a different technology or between behavioral and structural versions of the RAM code. Look at the mem_wrapper files included with this design for an example of how this can work. Below is an example instantiation of the mem_wrapper module that will work with both mem_wrapper_behavioral.vhd and mem_wrapper_proasic.vhd.

```

-- example component declaration for trace memory
component mem_wrapper is
  port (
    addr : IN std_logic_VECTOR(6 downto 0);
    clk  : IN std_logic;
    din  : IN std_logic_VECTOR(31 downto 0);
    dout : OUT std_logic_VECTOR(31 downto 0);
    we   : IN std_logic,
    oe   : IN std_logic
  );
end component;

-- example instantiation for trace memory
mem_wrapper_inst: mem_wrapper
port map (
  addr => trace_addr(6 downto 0),
  clk  => clk,
  din  => trace_din(31 downto 0),
  dout => trace_dout(31 downto 0),
  we   => trace_we,
  oe   => trace_oe
);

```

An example design which instantiates the Internal Logic Navigator IP is described in Appendix D.

4.2. Behavioral Simulation

To simulate your design with a simulator such as ModelSim, include the following files in your project design files and compile them in order as shown (lowest level to highest) along with other design files.

- mem_wrapper_behavioral.vhd (or equivalent from Actgen)
- comp.vhd (for APA FPGAs – this includes UJTAG component for simulation)
- clam2_oci_defs.vhd (package)
- clam2_oci_trigger.vhd
- clam2_oci_trace.vhd
- clam2_oci_jtagapa.vhd (with architecture behavioral_apa for APA FPGAs) or
- clam2_oci_jtag.vhd (with architecture behavioral for Antifuse FPGAs)
- Do not synthesize both clam2_oci_jtagapa.vhd and clam2_oci_jtag.vhd in a given design. Since they have same entity name, they can overwrite each other.**
- clam2_oci_inputset.vhd
- clam2_oci_timestamp.vhd
- clam2_oci.vhd
- clam2_oci_tb.vhd

Remember that the reset_n port must be pulsed low at the start of simulation by your testbench in order for proper simulation.

TCL scripts that direct compilation and execution design flows of Internal Logic Navigator for several sets of parameters are included in the Logic Navigator release. The simclam2apavhdl.tcl file supports the APA design flow and simclam2vhdl.tcl supports simulation of the Antifuse design flow. Directory definitions in the TCL files should be modified to the directory structure used for the given design.

4.3. Synthesis

To synthesize your design with a synthesis tool such as Synplify, include the following files in your project similar to any of your other design files and proceed with synthesis as usual.

- mem_wrapper_behavioral.vhd (or equivalent from Actgen)
- comp.vhd (for APA FPGAs – this includes UJTAG component for simulation)
- clam2_oci_defs.vhd (package)
- clam2_oci_trigger.vhd
- clam2_oci_trace.vhd
- clam2_oci_jtagapa.vhd (with architecture behavioral_apa for APA FPGAs) or
- clam2_oci_jtag.vhd (with architecture behavioral for Antifuse FPGAs)
- Do not synthesize both clam2_oci_jtagapa.vhd and clam2_oci_jtag.vhd in a given design. Since they have same entity name, they will overwrite each other.**
- clam2_oci_inputset.vhd
- clam2_oci_timestamp.vhd
- clam2_oci.vhd

The OCI generator creates a model named your_design_inav.vhd which integrates a component and mapping of the Logic Navigator subsystem IP to the top level target design file that is named your_design.vhd. Your_design_inav.vhd and any associated files (that may have trace mapping) should be synthesized instead of your_design and its associated files to include Logic Navigator debug IP to be implemented.

A mem_wrapper_proasic.vhd file is provided as an example of RAM instantiated with an Actel ProASIC^{Plus} memory block created by the ACTgen tool (with trace depth set = 128 words and

memory width set = 36 bits). The designer will need to create a mem_wrapper file similar to this which implements an (APA or Antifuse) Actel memory block for the design application that is compatible with the memory width and depth passed in as parameters in clam2_oci generics

In cases where Logic Navigator is interfacing to the Actel UJTAG port, the synthesis tools may not recognize that UJTAG TDO is an output. Synplicity, as an example, assumes that modules that produce no outputs have no effect and can therefore be pruned.

To force Synplicity to keep the clam2_oci module, two directives should be added to the architecture declaration of clam2_oci. These are only needed for On-Chip Logic Navigator APA implementation since only Actel APA uses the UJTAG macro. The OCI Generator will automatically add these directives for On-Chip Logic Navigator APA implementations.

For manual integration in an APA design, the directives should be added to clam2_oci.v as follows:

```
entity clam2_oci is
port (
  ..
  clam2_gpreg      : out  std_logic (gpreg_width-1 downto 0);
  clam2_state      : out  std_logic (1 downto 0);
);
end clam2_oci;

architecture behavioral_apa of clam2_oci is
-- Add these directives in architecture declaration section
-- before begin statement)

attribute syn_hier of behavioral_apa: architecture is "firm";
attribute syn_noprune of behavioral_apa : architecture is true;
```

4.4. Functional Simulation

Note: information on functional simulations is included for completeness; simulating the synthesized netlist does not generally need to be done.

If you wish to simulate your synthesized netlist with a simulator such as ModelSim you will first need to compile the Actel primitives into a library. The following sequence of commands will create a library and compile the Actel primitives for the ProASIC^{Plus} family into it. From the ModelSim prompt:

```
cd c:/your/full/path
vlib apa_ver
vmap apa_ver C:/your/full/path/apa_ver
vlog -reportprogress 300 -work apa_ver \
{C:/libero/designer/lib/vlog/apa.v}
```

Next compile the synthesized design into your work library. Designer will create a synthesized Verilog netlist with the same name as your top-level file with the extension in a directory called "designer" below your project directory. An example command for ModelSim would be:

```
vlog -reportprogress 300 -work work \
{C:/your/project/directory/designer/your_design_name.v}
```

Now, load the testbench into ModelSim with a command similar to this:

```
vsim -L C:/ your/full/path/apa_ver work.your_test_bench
```

Begin your simulation as usual.

Appendix A – Timing Requirements

1. Logic Navigator IP Timing Specifications

The Internal Logic Navigator operation has a longer on-chip critical path and is highly dependent on the parameters and trace width selected for maximum speed of operation, along with synthesis and place and route optimizations. An application note discussing these tradeoffs is forthcoming.

2. Internal Logic Navigator IP Timing Constraints

Table 6 shows the minimum and maximum frequencies for the On-Chip Logic Navigator IP.

Table 6. Internal Logic Navigator IP Timing Specifications

Max Trace Width	Min Input Frequency	Max Input Frequency
256 per set x 16 sets	0	Depends on technology

Table 7. Internal Logic Navigator IP Timing Constraints

Logic Navigator Version	Maximum Delay			
	Clock Signal	In to Reg	Reg to Reg	Reg to out
Internal	clk	none	1 clk period	none
(APA family)	udrck ¹	none	25ns	none
(AX family)	tck ¹	10ns	25ns	7.5ns

¹ These constraints assume a probe TckRate of 4MHz. A slower TckRate allows more relaxed timing.

All critical registers in the Internal Logic Navigator IP are clocked with clk. In most cases, the existing system timing constraints with respect to the system clock will also cover the Logic Navigator. For systems without a system timing constraint, define the clock period constraint using an SDC file. Example:

```
# Example Frequency Constraints for Internal Logic Navigator
create_clock -period 20.0000 [get_pins {clk_pll/Core:GLA}]
```

Appendix B – Area Requirements

Internal Logic Navigator Area Requirements

Below is a table showing the approximate area requirements for different configurations of the Logic Navigator. Area is not significantly impacted by variations in `datain_sets` or `trace_depth` (excluding the effect that `trace_depth` has on the amount of RAM required). Other parameters typically have a direct impact on area required.

This table has baseline Logic Navigator parameters configured as follows: (`trace_width`, `datain_sets`, `trace_depth`, `hdnodes`, `event_width`, `state width`, `ts_width`, `counters`, `gpreg_width`, `edge_events`, (pipeline)optimization). Parameters are described in section 5 (table 6).

The initial baseline example (first row on table) creates Logic Navigator IP with 32 bit wide trace, single input set, trace depth of 10 (1024 cycles), 2 events, 2 sequential states. Other rows show effect of changes in other parameters.

Table 8. Area Requirement for the Internal Logic Navigator IP vs. different configurations

Internal Logic Navigator Configuration	Notes	APA Core Cells	AX R+C Cells
(32,1,10,0,1,1,0,0,0,0,0)	Baseline (<code>trace_width</code> = 32 bit) , no edge triggering, unpipelined,	1580	1132
(32,1,10,0,1,1,0,0,0,0,1)	Baseline (<code>trace_width</code> = 32 bit) no edge triggering, pipelined	1681	1154
(32,1,10,1,1,1,0,0,0,0,1)	Adding single ext, trigger set	1763	1246
(32,1,10,1,1,1,32,0,0,0,1)	Adding 32 bit timestamp	2127	1455
(32,1,10,1,1,1,32,1,0,0,1)	Adding single counter	2669	1850
(32,1,10,1,1,1,32,1,8,0,1)	Adding 8 bit gpreg	2699	1858
(32,1,10,1,1,1,32,1,8,1,1)	Adding edge event triggering	2894	1977
(32,1,10,1,3,1,32,1,8,1,1)	Changing to 8 trigger events	6423	4202
(32,1,10,1,3,3,32,1,8,1,1)	Changing to 8 sequential states	12044	6134
(16,1,10,0,1,1,0,0,0,0,1)	Baseline (<code>trace_width</code> = 16 bit) no edge triggering, pipelined	1162	830
(16,1,10,0,1,1,0,0,0,1,1)	adding edge triggering	1244	865
(8,1,10,0,1,1,0,0,0,1,1)	Baseline (<code>trace_width</code> = 8 bit) edge triggering, pipelined	971	654

Appendix C – Pin Requirements

Voltage level of pins interfacing to the FS2 Probe

The FS2 Logic Navigator Probe uses LVTTTL I/O signals to interface with the Logic Navigator IP instantiated in the FPGA. All of the pins on the FPGA, which connect with the Probe, should be compatible with this I/O standard. Acceptable options for a ProASIC^{Plus} devices include LVTTTL, and PCI.

For APA devices, the JTAG signals TCK, TMS, TDI, and TDO are not specified as external ports from the top-level design. Instead, these signals are implicitly connected to the APA device's dedicated JTAG pins through the UJTAG hardware module present in all APA devices.

For antifuse devices, JTAG signals do appear in the top-level design and should connect directly to pads.

Appendix D – Example Top-Level Designs

1. Example Verilog Design for Internal Logic Navigator Reference designs

Top-level Reference designs, which instantiate different FS2 Logic Navigator IP versions, are included in the Verilog file set. These files are intended as a reference to the different steps that are required to complete a design with the Logic Navigator IP. Table 11 lists the files in the Verilog file set for internal Logic Navigator top-level Reference designs.

Table 11. Files for Example Verilog Top-Level Design

jtaglnav_int.v	Root file for the design which instances Internal Logic Navigator IP, full JTAG tap version (for use with Antifuse FPGAs)
apalnav_int.v	Root file for the design which instances Internal Logic Navigator IP, with JTAG interface to UJTAG (for use with APA FPGAs)

2. Example VHDL Design for Internal Logic Navigator Reference designs

Top-level Reference designs, which instantiate different FS2 Logic Navigator IP versions, are included in the VHDL file set. These files are intended as a reference to the different steps that are required to complete a design with the Logic Navigator IP. Table 12 lists the files in the VHDL file set for internal Logic Navigator top-level Reference designs.

Table 12. Files for Example VHDL Top-Level Design

jtaglnav_int.vhd	Root file for the design which instances Internal Logic Navigator IP, full JTAG tap version (for use with Antifuse FPGAs)
apalnav_int.vhd	Root file for the design which instances Internal Logic Navigator IP, with JTAG interface to UJTAG (for use with APA FPGAs)

3. STP files for Internal Logic Navigator Reference design

Precompiled STP (STAPL) files, which instantiate the FS2 apalnav_int.v reference design and Logic Navigator IP, are included in the Logic Navigator release. These files are intended as a reference for test and training purposes. Table 13 lists the STP files for different APA chip configurations.

Table 13. Logic Navigator STP Reference Files

apalnav_int_apa075.stp	STP file of apalnav_int.v design targeted for APA075 part.
apalnav_int_apa300.stp	STP file of apalnav_int.v design targeted for APA300 part.
apalnav_int_isp.stp	STP file of apalnav_int_isp.v design targeted for APA750 ISP board.