LogiCORE IP ChipScope Pro Integrated Logic Analyzer (ILA) (v1.04a)

DS299 March 1, 2011 **Product Specification**

Introduction

The LogiCORE™ IP ChipScope™ Pro Integrated Logic Analyzer (ILA) core is a customizable logic analyzer core that can be used to monitor any internal signal of your design. The ILA core includes many advanced features of modern logic analyzers, including boolean trigger equations, trigger sequences, and storage qualification. Because the ILA core is synchronous to the design being monitored, all design clock constraints that are applied to your design are also applied to the components inside the ILA core.

Features

- Provides a communication path between the ChipScope Pro Analyzer software and capture cores via the ChipScope Pro Integrated CONtroller (ICON) core
- Has user-selectable trigger width, data width, and data depth
- Has multiple trigger ports, which can be combined into a single trigger condition or sequence
- Includes storage qualification option that enables the core to store a sample only when a certain condition is met

For more information about the ILA core, refer to the *ChipScope Pro Software and Cores User Guide*.

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family[1]	Kintex®-7, Virtex®-7, Virtex-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3/XA, Spartan-3E/XA, Spartan-3A/3AN/3A DSP/XA				
Supported User Interfaces	Not applicable				
	Resources				**Frequency**
Configuration[2]	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs**	**Max. Freq.**
Config1	156	270	0	1	313.239 MHz
Config2	391	698	0	4	243.858 MHz
Config3	4262	8400	0	228	412.788 MHz
Provided with Core					
Documentation	Product Specification User Guide				
Design Files	Netlist				
Example Design	Verilog /VHDL				
Test Bench	Not Provided				
Constraints File	Xilinx Constraints File				
Simulation Model	Not Provided				
Tested Design Tools					
Design Entry Tools	CORE Generator™ tool, XPS				
Simulation	Not Provided				
Synthesis Tools	Not Provided.				
Support					
Provided by Xilinx, Inc.					

Notes:

1. For a listing of supported devices, see the release notes for this core.
2. For configuration details, see Table 5, page 8.

Applications

The ILA core is designed to be used in any application that requires verification or debugging using the ChipScope Pro software and cores.

Functional Description

Signals in the FPGA design are connected to ILA core inputs, and those signals can be captured at design speeds. Before the design is implemented, select the parameters of the core, including how many signals to capture and how many samples can be captured. Communication with the ILA core is conducted using a connection to the JTAG port through the ICON core, as shown in Figure 1.

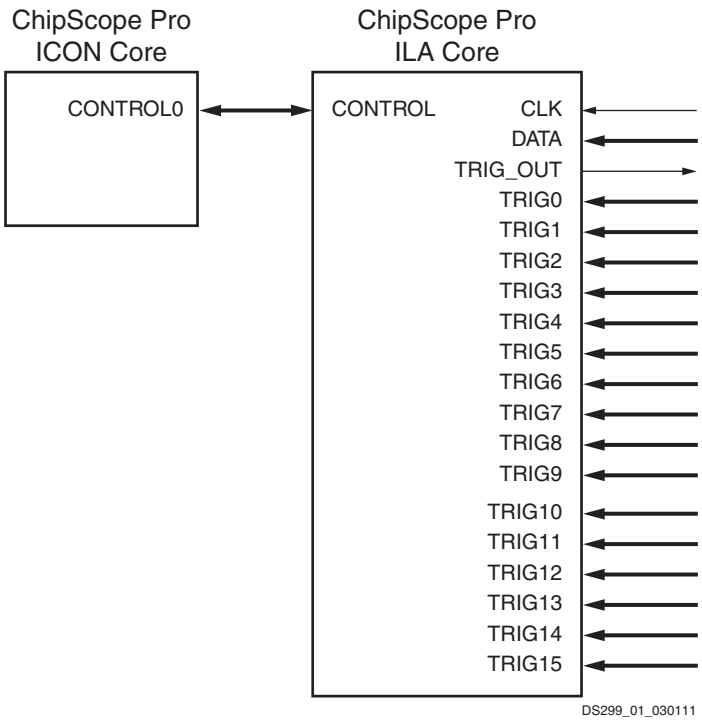

Figure 1: **ILA Core Connection to ICON Core**

After the design is loaded into the FPGA device on the board, you can use the ChipScope Pro Analyzer software to set up trigger conditions that define when and how to capture the signals connected to the ILA core. After the trigger occurs and the sample buffer is filled, the data buffer is uploaded into the Analyzer software. You can display this data in waveform or list format.

Regular FPGA logic is used to implement the match logic, capture control, and status functionality. On-chip block RAM memory stores the data until it is uploaded by the software. No user input or output is required to trigger on events, capture data, or communicate with the ILA core.

Triggering

You must choose when to capture the data that comes into the ILA core. This moment in time is called the trigger event.

Trigger Ports

The ability to monitor different kinds of signals and buses in the design requires the use of multiple trigger ports. For example, if you are instrumenting an internal system bus in your design that is made up of control, address, and data signals, then you could assign a separate trigger port to monitor each signal group. If you connected all of these different signals and buses to a single trigger port, you would not be able to monitor for individual bit transitions on the CE, WE, and OE signals while looking for the address bus to be in a specified range. The flexibility of being able to choose from different types of match units allows you to customize the ILA cores to your triggering needs while keeping resource usage to a minimum.

Match Logic

Each trigger port can have one or more match units, which are blocks of logic that do the actual comparisons. Several types of match units are available (Table 1). The simplest match unit can only do an = (equals) or <> (not equals) comparison, but take up the least amount of FPGA resources. The most complex match unit can do all types of comparisons, including =, <>, >, <, >=, <=, and range comparisons. The match unit can also be configured to include edge detection, whether it be a rising, falling, or either edge comparison.

Table 1: **Match Unit Types**

Match Unit Name	Description
Basic Basic with Edges	The Basic match unit can only do = and <> comparisons. Basic with edges can match R (rising), F (falling), B (either), or N (no) edge.
Extended Extended with Edges	The Extended match unit can do =, <>, >, <, >=, and <= comparison. Extended with edges can do edge matching for the = and <> operators.
Range Range with Edges	The Range match unit can do everything the Extended match unit can do, along with matching on a range of values or matching on values outside a specific range.

Match Unit Counters

The group of match units on a particular trigger port can also be configured to have a match counter on the output of each. The match counter can be used to detect a certain number of events, either sequentially or non-sequentially. The size of the counter is determined at generation time. The number of events to count and whether to count them sequentially is decided at runtime.

Trigger Conditions and Storage Qualification

The ILA core implements both trigger and storage qualification condition logic. The trigger condition is a Boolean or sequential combination of events that is detected by match unit comparators attached to the trigger ports of the core. The trigger condition marks a distinct point of origin in the data capture window and can be located at the beginning, the end, or anywhere within the data capture window.

Similarly, the storage qualification condition is also a Boolean combination of events that is detected by match unit comparators that are subsequently attached to the trigger ports of the core. However, the storage qualification condition differs from the trigger condition in that it evaluates trigger port match unit events to decide whether or not to capture and store each individual data sample. The trigger and storage qualification conditions can be used together to define when to start the capture process and what data is captured.

ILA Core Example

To accomplish the following:

- Trigger on the first memory write cycle (CE = rising edge, WE = 1, OE = 0) to Address = 0xFF0000
- Capture only memory read cycles (CE = rising edge, WE = 0, OE = 1) from Address = 0x23AACC where the Data values are between 0x00000000 and 0x1000FFFF

To implement these conditions successfully, ensure that both the TRIG0 and TRIG1 trigger ports each have two match units attached to them: one for the trigger condition and one for the storage qualification condition. To set up the trigger and storage qualification equations and each individual match unit to satisfy the conditions above, set the following:

- Trigger Condition = M0 && M2, where:
  - M0[2:0] = CE, WE, OE = R10 (where 'R' means 'rising edge').
  - M2[23:0] = Address = 0xFF0000
- Storage Qualification Condition = M1 && M3 && M4, where:
  - M1[2:0] = CE, WE, OE = R10 (where 'R' means 'rising edge')
  - M3[23:0] = Address = 0x23AACC
  - M4[31:0] = Data = in the range of 0x00000000 through 0x1000FFFF

The triggering and storage qualification capabilities of the ILA core allow the user to locate and capture exactly the information needed without wasting valuable on-chip memory resources.

Trigger Output Logic

The ILA core implements a trigger output port called `TRIG_OUT`. The `TRIG_OUT` port is the output of the trigger condition that is set up at runtime using the Analyzer. The shape (level or pulse) and sense (active-High or active-Low) of the trigger output can also be controlled at runtime. The latency of the `TRIG_OUT` port relative to the input trigger ports is 10 clock cycles.

The `TRIG_OUT` port is very flexible and has many uses. For example, to trigger external test equipment such as oscilloscopes and logic analyzers, connect the `TRIG_OUT` port to a device pin. Connecting the `TRIG_OUT` port to an interrupt line of an embedded PowerPC® or MicroBlaze™ processor can be used to cause a software event to occur.

To expand the trigger and data capture capabilities of the on-chip debug solution, connect the `TRIG_OUT` port of one core to a trigger input port of another core .

Data Capture Logic

Each ILA core can capture data using on-chip block RAM resources independently from all other cores in the design. Each ILA core can also capture data using one of two capture modes: Window and N samples.

Window Capture Mode

In Window capture mode, the sample buffer can be divided into one or more equal-sized sample windows. The window capture mode uses a single trigger condition event, such as a Boolean combination of the individual trigger match unit events, to collect enough data to fill a sample window.

In the case where the depth of the sample windows is a power of 2 up to 131,072 samples, the trigger position can be set to the beginning of the sample window (trigger first, then collect), the end of the sample window (collect until the trigger event), or anywhere in between. In the other case where the window depth is not a power of 2, the trigger position can only be set to the beginning of the sample window.

Once a sample window has been filled, the trigger condition of the ILA core is automatically re-armed and continues to monitor for trigger condition events. This process is repeated until all sample windows of the sample buffer are filled or you halt the ILA core.

N Samples Capture Mode

The N Samples capture mode is similar to the Window capture mode except for two major differences:

- The number of samples per window can be any integer `N` from 1 to *<sample buffer size>* - 1
- The trigger position must always be at position `0` in the window

The N sample capture mode is useful for capturing the exact number of samples needed per trigger without wasting valuable capture storage resources.

Trigger Marks

The data sample in the sample window that coincides with a trigger event is tagged with a trigger mark. This trigger mark tells the Analyzer the position of the trigger within the window. This trigger mark consumes one extra bit per sample in the sample buffer.

Data Port

The ILA core provides the capability to capture data on a port that is separate from the trigger ports that are used to perform trigger functions. This feature is useful for limiting the amount of data to be captured to a relatively small amount since it is not always useful to capture and view the same information that is used to trigger the core.

However, in many cases it is useful to capture and view the same data that is used to trigger the core. In this case, choose for the data to consist of one or more of the trigger ports. Using this feature will conserve resources while providing the flexibility to select the useful trigger information to be captured.

Data Depth

Because different depths of block RAM are available across all the architectures supported, different data depths are supported for each, as described in Table 2.

Table 2: **ILA Data Depths Available**

Device	Range	Default
Virtex-4, Spartan-3, Spartan-3E, Spartan-3A, Spartan-3A DSP	512, 1024, 2048, 4096, 8192, 16384	512
Virtex-5, Virtex-6, Virtex-7, Spartan-6, and Kintex-7.	1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072	1024

Interface Ports

The I/O signals of the ILA core consist of a control bus to interface to ICON, and one or more other ports to connect to the surrounding logic, depending on the parameters used when the core was generated. See Table 3.

Table 3: **ILA Interface Ports**

Port Name	Direction	Description
CLK	IN	Design clock that clocks all trigger and storage logic. Mandatory.
CONTROL[35:0]	INOUT(1)	Control bus connection to the ICON core. Mandatory. ***Note:*** For XPS designs, the direction of this port is IN.
DATA[<*m*>-1:0]	IN	Data port: The data port width (denoted by <*m*>) is in the range of 1 to 4096 bits for the Virtex-5 device family, and 1 to 256 for all other device families. Optional (depends on data_same_as_trigger parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use DATA[0:0].
TRIG_OUT	OUT	Trigger output port. Optional (depends on enable_trigger_output_port parameter).
TRIG0[<*m*>-1:0]	IN	Trigger port number 0: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Mandatory. ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG0[0:0].
TRIG1[<*m*>-1:0]	IN	Trigger port number 1: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG1[0:0].
TRIG2[<*m*>-1:0]	IN	Trigger port number 2: The port width (denoted by <*m*>) is in the range of 1 to 256 for all other device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG2[0:0].
TRIG3[<*m*>-1:0]	IN	Trigger port number 3: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG3[0:0].
TRIG4[<*m*>-1:0]	IN	Trigger port number 4: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG4[0:0].
TRIG5[<*m*>-1:0]	IN	Trigger port number 5: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG5[0:0].
TRIG6[<*m*>-1:0]	IN	Trigger port number 6: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG6[0:0].
TRIG7[<*m*>-1:0]	IN	Trigger port number 7: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG7[0:0].
TRIG8[<*m*>-1:0]	IN	Trigger port number 8: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG8[0:0].
TRIG9[<*m*>-1:0]	IN	Trigger port number 9: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG9[0:0].
TRIG10[<*m*>-1:0]	IN	Trigger port number 10: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG10[0:0].

Table 3: **ILA Interface Ports *(Cont'd)***

Port Name	Direction	Description
TRIG11[<*m*>-1:0]	IN	Trigger port number 11: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG11[0:0].
TRIG12[<*m*>-1:0]	IN	Trigger port number 12: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG12[0:0].
TRIG13[<*m*>-1:0]	IN	Trigger port number 13: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG13[0:0].
TRIG14[<*m*>-1:0]	IN	Trigger port number 14: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG14[0:0].
TRIG15[<*m*>-1:0]	IN	Trigger port number 15: The trigger port width (denoted by <*m*>) is in the range of 1 to 256 for all device families. Optional (depends on number_of_trigger_ports parameter). ***Note:*** This port must be declared as a vector. For a one-bit port, use TRIG15[0:0].

Notes:

1. For projects created using Xilinx Platform Sturdio, the direction for CONTROL ports is IN.

XCO Parameters

Table 4: **ILA XCO Parameters**

Parameter Name	Allowable Values	Default Value	Description
component_name	String with A-z, 0-9, and _ (underscore)	ila	Name of instantiated component
counter_width_<*n*>	Disabled or 1-32	Disabled	Width of the match unit counters associated with each of the match units connected to trigger port <n>. The value "Disabled" indicates that no match counters are to be used on that trigger port.
data_port_width	1-4096 for Virtex-5, Virtex-6, Spartan-6, Kintex-6, and Virtex-7; otherwise 1-256	32	Size of optional data port, if used.
data_same_as_trigger	true = Use one or more trigger ports as the data capture bus false = Use the optional data port as the data capture bus	true	Used to specify whether to capture trigger ports as data or to use the optional data port.
enable_storage _qualification	true = Enable storage qualification condition false = Disable storage qualification condition	true	Enable optional storage qualifier.
enable_trigger_output_port	true = Enable trigger output port false = Disable trigger output port	false	Use optional trigger output port.
exclude_from_data_storage<*n*>	true = Exclude trigger port <*n*> from data capture false = Include trigger port <*n*> in data capture	false	Exclude trigger port <*n*> from the data storage if true. Only applicable if data_same_as_trigger is true.
match_type<*n*>	basic, basic_with_edges, extended, extended_with_edges, range, range_with_edges	basic	The match unit type to use for all match units connected to trigger port <*n*>.

Table 4: **ILA XCO Parameters *(Cont'd)***

Parameter Name	Allowable Values	Default Value	Description
match_units_<*n*>	1-16	1	Number of match units in trigger port <*n*>. The total number of match units used across all trigger ports cannot exceed 16.
max_sequence_levels	1-16	1	Number of levels or 'states' in the trigger sequencer. A value of 1 means the trigger sequencer is not used.
number_of_trigger_ports	1-16	1	Number of trigger ports
sample_data_depth	Refer to Table 2, page 5	Refer to Table 2	Depth of the data buffer.
sample_on	rising = Sample on rising edge of clk falling = Sample on falling edge of clk	rising	The edge of the clk port to capture and trigger upon.
trigger_port_width_<*n*>	1-256	8	Size of trigger port <*n*>.
use_rpms	true = Use RPMs false= Don't use RPMs	true	Use relative-placed macro constraints to constrain logic placement.

Restrictions

A maximum of 15 cores can be used in a single design.

Configuration

Table 5: **Configuration Details**

Configuration Name	Device	ILA Setup
Config1	Xc5vlx20t-ff323-2	Default settings
Config2	Xc6vlx240t-ff1156-2	32-bit wide, 4 trigger ports, and trigger output enabled with sample depth of 1024
Config3	Xc7vhx285t-ff1157-2	4096-bit wide sample data with sample depth of 2048

Using the ILA Core in the Embedded Development Kit (EDK)

The ILA core can be inserted into an embedded processor design using EDK. In this case, the ILA core depends on ICON and OPB_MDM component instances already being present in the design (Figure 2.)

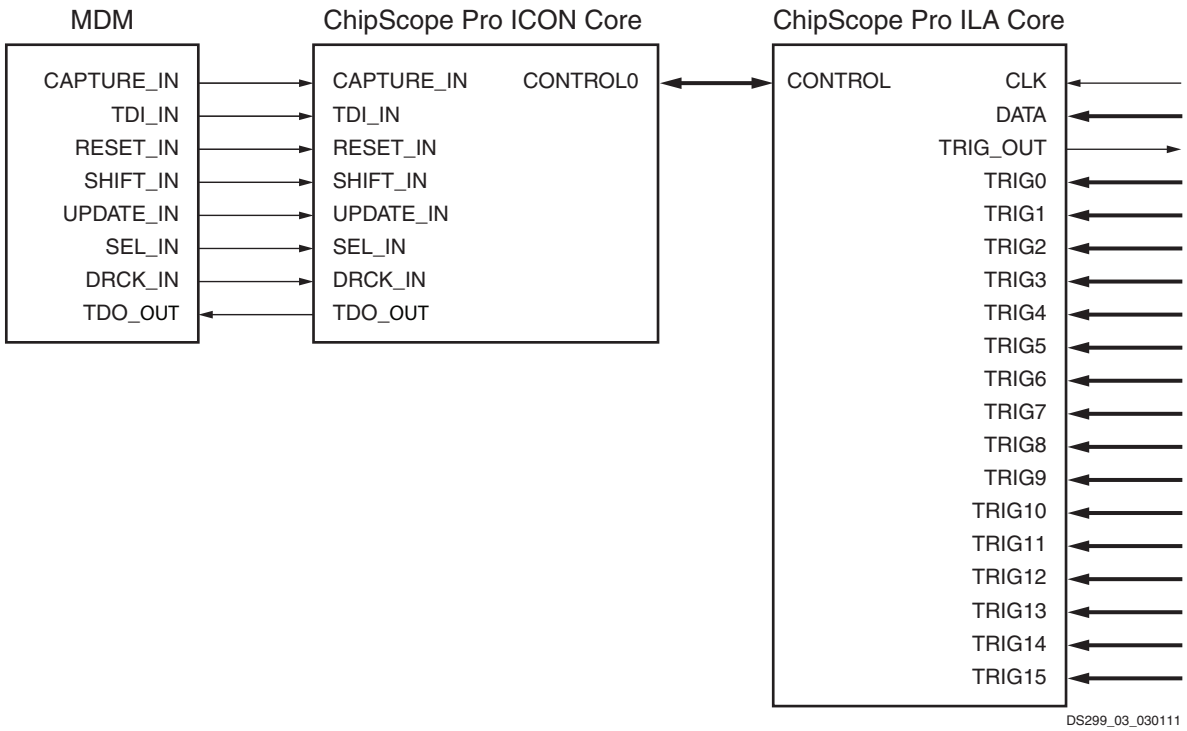

Figure 2: **ILA Core Component in EDK Design**

In EDK, the ILA core is integrated into the tool using a Tcl script. When the EDK Platgen tool is run, the Tcl script is called and the script internally calls Core Generator in command line mode. The Tcl script provides Coregen an arguments file (.xco) to generate the ILA core netlist. The Tcl script also generates a HDL wrapper to match the ILA ports based on the core parameters found in Table 6.

Table 6: **EDK-specific Parameters**

Parameter Name	Allowable Values	Default Value	Description
c_data_in_width	1-4096 for Virtex-5, Virtex-5, Spartan-6, Virtex-7, and Kintex-7; otherwise 1-256	32	Data port width, if used
c_data_same_as_trigger	0, 1	1	1 = data same as trigger 0 = data different than trigger
c_disable_rpm	0, 1	0	0 = enable RPMs 1 = disable RPMs
c_enable_trigger_out	0, 1	0	0 = disable trigger out 1 = enable trigger out
c_family	virtex4, virtex5, virtex6, virtex6l, virtex7, kintex7, spartan3, spartan3a, spartan3adsp, spartan3e, spartan6, spartan6l, aspartan3, aspartan3a, aspartan3adsp, aspartan3e, aspartan6, avirtex4, qspartan6, qspartan6l, qrvirtex4, qspartan6, qspartan6l, qvirtex4, qvirtex5, qvirtex6	virtex5	Device family to use
c_max_sequencer_levels	1-16	16	number of sequencer levels, if used
c_num_data_samples	see Table 2, page 5	see Table 2	Data buffer depth

Table 6: **EDK-specific Parameters *(Cont'd)***

Parameter Name	Allowable Values	Default Value	Description
c_rising_clock_edge	0, 1	1	0 = falling edge 1 = rising edge
c_trig<*n*>_counter_width	0-32	0	0 = disable match counter 1-32 = match counter width for match units connected to trigger port <*n*>
c_trig<*n*>_match_type	basic, basic with edges, extended, extended with edges, range, range with edges	basic	Match unit type for all match units connected to trigger port <*n*>
c_trig<*n*>_trigger_in_width	1-256	8	trigger port <*n*> width, if used
c_trig<*n*>_units	0-16	0	0 = disable unit 1-16 = number of match units for trigger port <*n*>

The XST synthesis tool is used for synthesizing the wrapper HDL generated for the ILA core. The NGC netlist outputs from XST and CORE Generator are subsequently incorporated into the ISE for actual device implementation.

Verification

Xilinx has verified the ILA core in a proprietary test environment, using an internally developed bus functional model.

References

1. More information on the ChipScope Pro software and cores is available in the Software and Cores User Guide, located at http://www.xilinx.com/documentation.
2. Information about hardware debugging using ChipScope Pro in EDK is available in the Platform Studio online help, located at http://www.xilinx.com/documentation.
3. Information about hardware debugging using ChipScope Pro in System Generator for DSP is available in the Xilinx System Generator for DSP User Guide, located at http://www.xilinx.com/documentation.

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx ISE® Design Suite Embedded Edition software under the terms of the Xilinx End User License. The core is generated using the Xilinx ISE Design Suite software. For more information, visit the Chipscope ILA page.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE modules and software, please contact your local Xilinx sales representative.

Revision History

The following table shows the revision history for this document:

Date	Document Version	Description of Revisions
03/24/2008	1.0	Release 10.1 (Initial Xilinx release).
04/25/2008	1.1	Release 10.1 Service Pack 1.
09/19/2008	1.2	Release 10.1 Service Pack 3.
04/07/2009	2.0	Release 11.1.
04/19/2010	3.0	Release 12.1.
03/01/2011	3.1	Updated to v1.04a for the 13.1 release.

Notice of Disclaimer