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

DS299 June 22, 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

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family (1)	Kintex-7(6), Virtex-7, Virtex-6(4), Virtex-5, Virtex-4, Spartan-6(5), Spartan-3/XA, Spartan-3E/XA, Spartan-3A/3AN/3A DSP/XA				
Supported User Interfaces	Not applicable				
	Resources				**Frequency**
Configuration (3)	**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 (2)					
Design Entry Tools	CORE Generator tool, XPS				
Simulation	Not Provided				
Synthesis Tools	Not Provided.				
Support					
Provided by Xilinx, Inc.					

Notes:

1. For a list of supported derivative devices, see http://www.xilinx.com/ise/embedded/ddsupport.htm.
2. For the supported versions of the tools, see the ISE Design Suite 13: Release Notes Guide.
3. For configuration details, see Table 4, page 13.
4. For more information, see the DS150 Virtex-6 Family Overview Product Specification.
5. For more information, see DS160 Spartan-6 Family Overview Product Specification.
6. For more information, see DS180 7 Series FPGAs Overview.

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.

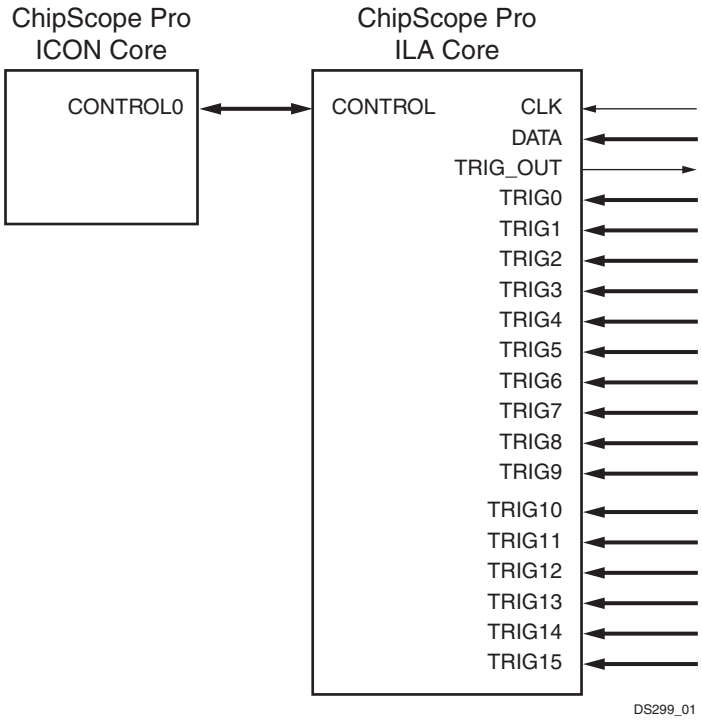

Figure 1: **ILA Core Connection to ICON Core**

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.

There are a maximum of 16 trigger ports on each ILA. Each trigger port can consist of between 1 and 256 signals. The trigger port input is compared against a set of expected values known as match units. The comparison can be a test for equality, inequality, within a range, outside a range, greater than, greater than or equal to, less than, or less than or equal to. The result of all of the match units is then sent to the trigger event detector. This device combines the results of match units in either a logical or sequential fashion. If the match equations evaluate to true, then a trigger event occurs and data is collected and stored into trace memory.

The user gains access to the collected data stored in block RAM via the ICON interface that connects to the ICON core which finally connects to the PC running the ChipScope Pro Analyzer via the JTAG cable. This interface also allows the user to set the conditions on which the match unit tests the various triggers. The specific settings of the match units and the trigger event detector are programmable via the ChipScope Pro Analyzer; however, the match unit capabilities must be defined when the ILA core is created-either through ChipScope Pro Core Inserter or CORE Generator tool.

CORE Generator

The CORE Generator tool provides the ability to define and generate a customized ILA capture core to use with HDL designs. You can customize the number, width, and capabilities of the trigger ports. You can also customize the maximum number of data samples stored by the ILA core and the width of the data samples.

ILA Core Trigger and Storage Parameters

The CORE Generator tool is used to set up the ILA core parameters, including the general trigger and storage parameters and the trigger port parameters.

Entering the Component Name

The Component Name field, stored as component_name in the generated XCO parameter file, can consist of any combination of alpha-numeric characters in addition to the underscore symbol. However, the underscore symbol cannot be the first character in the component name.

Generating an Example Design

The ILA core generator normally generates standard Xilinx CORE Generator output files only, such as netlist and instantiation template files. To generate an example design that uses the ILA core, in addition to the normal generated files, select the Generate Example Design check box. This parameter is stored as example_design in the generated XCO parameter file.

Selecting the Number of Trigger Ports

Each ILA core can have up to 16 separate trigger ports that can be set up independently. Select the number of trigger ports from Number of Trigger Ports pull-down list. This parameter is stored as number_of_trigger_ports in the generated XCO parameter file.

Enabling the Trigger Condition Sequencer

The trigger condition sequencer can be either a Boolean equation, or an optional trigger sequencer that is controlled by the Max Sequence Levels pull-down, stored as max_sequence_levels in the XCO parameter file. A block diagram of the trigger sequencer is shown in Figure 2.

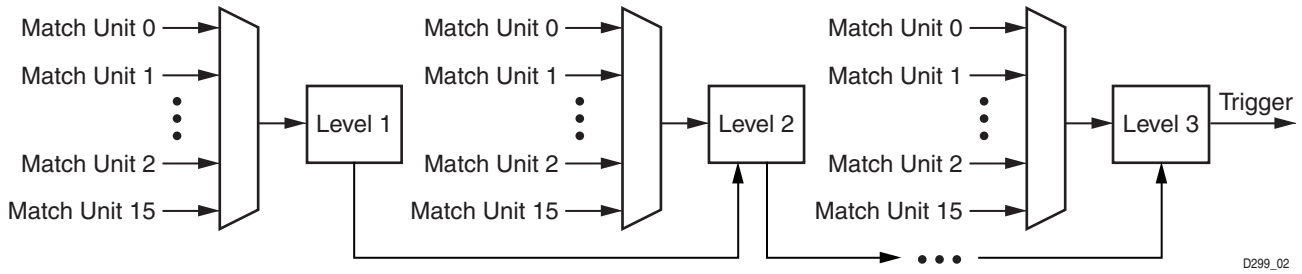

Figure 2: **Trigger Sequencer Diagram**

The Max Sequence Levels sets the maximum levels of Match Units that can be placed in sequence by ChipScope Analyzer to activate the trigger condition.

Using RPMs

The ILA core normally uses relationally placed macros (RPMs) to increase the performance of the core. The usage of RPMs by the ILA core can be disabled by deselecting the Use RPMs check box. It is recommended that the Use RPMs check box remain enabled. This parameter is stored as use_rpms in the generated XCO parameter file.

Enabling the Trigger Output Port

To output the result the ILA trigger condition, select the Enable Trigger Output Port check box. This parameter is stored as enable_trigger_output_port in the generated XCO parameter file. The result of the ILA trigger output has 10 clock cycles latency in respect to the trigger input ports. The shape (level or pulse) and sense (active-High or active-Low) of the trigger output is controlled by ChipScope Analyzer. The trigger output is often required to trigger external test equipment or used as a trigger, interrupt, or other control signal in the design.

Selecting the Clock Edge

The ILA unit can use either the rising or falling edges of the CLK signal to trigger and capture data. Select either the rising or falling edge from the Sample On pull-down list. This parameter is stored as sample_on in the generated XCO parameter file.

Selecting the Sample Data Depth

The maximum number of data sample words that the ILA core can store in the sample buffer is controlled by the Sample Data Depth pull-down list. This parameter is stored as sample_data_depth in the generated XCO parameter file.

Enabling the Storage Qualification Condition

In addition to the trigger condition, the ILA core can also implement a storage qualification condition. 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 to capture. The storage qualification condition can be enabled by checking the Enable Storage Qualification check box. This parameter is stored as enable_storage_qualification in the generated XCO parameter file.

Selecting the Data Type

The data captured by the ILA trigger port can come from two different source types and is controlled by the Data Same as Trigger check box:

- When the Data Same as Trigger check box is selected:
  - The data and trigger ports are identical.
  - Individual trigger ports can be selected to be excluded from the data port.
  - This mode conserves CLB and routing resources in the ILA core, but is limited to a maximum aggregate data sample word width of 4,096 bits (or 256 bits for Spartan®-3, Spartan-3E, Spartan-3A, Spartan-3A DSP, and Virtex®-4 devices).
- When the Data Same as Trigger check box is not selected
  - The data port is completely independent of the trigger ports.
  - This mode is useful when you want to limit the amount of data being captured.
  - In the case of data not same as trigger, the Data Port Width parameter needs to be specified.

The Data Port Width parameter is stored as data_same_as_trigger in the generated XCO parameter file.

Entering the Data Port Width

The width of each data sample word stored by the ILA core is set by Data Port Width field and stored as data_port_width in the generated XCO parameter file. If the data and trigger words are independent from each other, then the maximum allowable data width depends on the target device type and data depth. However, regardless of these factors, the maximum allowable data width is 4,096 bits (or 256 bits for Spartan-3, Spartan-3E, Spartan-3A, Spartan-3A DSP, and Virtex-4 devices).

ILA Core Trigger Port Parameters

By clicking Next from the trigger and storage ILA core option, a trigger port options screen appears. A separate panel is used to set each trigger port; the number of ports is set by the Number of Trigger Ports pull-down on the trigger and storage ILA core option screen.

Entering the Width of the Trigger Ports

The number of bits used to compose a trigger port is called the trigger width. The width of each trigger port can be set independently using the Trigger Port Width field and is stored as trigger_port_width_XX in the generated XCO parameter file. The range of values that can be used for trigger port widths is 1 to 256.

Selecting the Number of Trigger Match Units

A match unit is a comparator that is connected to a trigger port and is used to detect events on that trigger port. Each trigger port TRIGn can be connected up to 16 match units by using the Match Units pull-down list. This parameter is stored as match_units_XX in the generated XCO parameter file.

Selecting Match Unit Counter Width

The match unit counter is a configurable counter on the output of the each match unit in a trigger port that can be configured at run time to count a specific number of match unit events. To include a match counter select value from 1 to 32 from the pull-down list or to Disabled to disable the counter. This parameter is stored as counter_width_XX in the generated XCO parameter file.

Selecting the Match Unit Type

The different comparisons or match functions that can be performed by the trigger port match units depend on the type of the match unit. Six types of match units are supported by the ILA cores (Table 1).

Table 1: **ILA Trigger Match Unit Types**

Type	Bit Values [1]	Match Function	Bits Per Slice [2]	Description
Basic	0, 1, X	‘=’, ‘<>’	LUT4-based: 8 Virtex-5. Spartan-6: 19 Other LUT6-based: 20	Can be used for comparing data signals where transition detection is not important. This is the most bit-wise economical type of match unit.
Basic w/edges	0, 1, X, R, F, B, N	‘=’, ‘<>’	LUT4-based: 4 LUT6-based: 8	Can be used for comparing control signals where transition detection, such as low-to-high, high-to-low, and so forth, is important.
Extended	0, 1, X	‘=’, ‘<>’, ‘>’, ‘>=’, ‘<’, ‘<=’	LUT4-based: 2 LUT6-based: 16	Can be used for comparing address or data signals or data signals where magnitude is important.
Extended w/edges	0, 1, X, R, F, B, N	‘=’, ‘<>’, ‘>’, ‘>=’, ‘<’, ‘<=’	LUT4-based: 2 LUT6-based: 8	Can be used for comparing address or data signals or data signals where magnitude and transition detection are important.
Range	0, 1, X	‘=’, ‘<>’, ‘>’, ‘>=’, ‘<’, ‘<=’ ‘in range’, ‘not in range’	LUT4-based: 1 LUT6-based: 8	Can be used for comparing address or data signals where a range of values is important.

Table 1: **ILA Trigger Match Unit Types**

Type	Bit Values (1)	Match Function	Bits Per Slice (2)	Description
Range w/ edges	0, 1, X, R, F, B, N	'=', '<>', '>', '>=', '<', '<=' 'in range', 'not in range'	LUT4-based: 1 LUT6-based: 4	Can be used for comparing address or data signals where a range of values and transition detection are important.

Notes:

1. Bit values: '0' = "logical 0"; '1' = "logical 1"; 'X' - "don't care"' 'R' = "0-to1 transition"; 'F' = "1-to0 transition"; 'B' = "any transition"; 'N' = "no transition"
2. The Bites Per Slice value is only an approximation that is used to illustrate the relative resource utilization of the different match unit types. The value should not be used as a definitive estimate of resource utilization. LUT4-based device families are Spartan-3, Spartan-3E, Spartan-3A, Spartan-3A DSP, Virtex-4 FPGAs, and the variants of these families. LUT6-based device families are Virtex-5, Virtex-6, Spartan-6, Kintex®-7, Virtex-7, and the variants of these families.

Use the Match Type pull-down list to select the type of match unit that applies to all match units connected to the trigger port. This parameter is stored as match_type_XX in the generated XCO parameter file.

Selecting the Data-Same-As-Trigger Ports

If the Data Same As Trigger check box is selected, then a check box called Exclude Trigger Port from Data Storage appears in the trigger port options screen. Putting a check mark in this check box will cause the trigger port to be excluded from the aggregate data port. A maximum data width of 4,096 bits (or 256 bits for Spartan-3, Spartan-3E, Spartan-3A, Spartan-3A DSP, and Virtex-4 devices) applies to the aggregate selection of trigger ports. This parameter is stored as exclude_from_data_storage_XX in the generated XCO parameter file.

Generating the Core

After entering the ILA core parameters, click Generate to create the ILA core files. After the ILA core has been generated, a list of files that are generated will appear in a separate window called "Readme <corename>".

Using the ILA Core

To instantiate the example ILA core HDL files into your design, use the following guidelines to connect the ILA core port signals to various signals in your design:

- Connect the ILA core's CONTROL port signal to an unused control port of the ICON core instance in the design
- Connect all unused bits of the ILA core's data and trigger port signals to "0". This prevents the mapper from removing the unused trigger and/or data signals and also avoids any DRC errors during the implementation process
- Make sure the data and trigger source signals are synchronous to the ILA clock signal (CLK)

Example 1: ILA connection in VHDL and Example 2: ILA Connection in Verilog show how the ILA core is connected in vhdl and verilog respectively. Note how the control bus control0 is attached to the control port of the ILA. In the Verilog example an empty module declaration is created for the ICON and ILA module. This is used as a black box declaration so that the synthesis tool properly accounts for the generated netlists.

Example 1: ILA connection in VHDL

```
entity example_chipscope_ila is
port (
    clk_i : in std_logic
    );
end example_chipscope_icon;

architecture ila_arch of example_chipscope_ila is
  ------------------------------------------------------------------------
  --
  --  Component declarations
  --
  ------------------------------------------------------------------------
 component chipscope_icon
   port (
    .CONTROL0 : inout std_logic_vector(35 downto 0));
 end component;

 component chipscope_ila
   port (
     CONTROL →: inout std_logic_vector(35 downto 0);
     CLK:  in  std_logic;
     TRIG0:  in std_logic_vector(7 downto 0));
 end component;

 ------------------------------------------------------------------
 --  Local Signals
 ------------------------------------------------------------------
  signal control0 : std_logic_vector(35 downto 0);
  signal trig0: std_logic_vector(7 downto 0);

begin
 ------------------------------------------------------------------
 --
 --  ICON Pro core instance
 --
 ------------------------------------------------------------------
 ICON_inst:  chipscope_icon
   port map (
     CONTROL0 => control0);
  --
 ------------------------------------------------------------------
  --
  --  ILA Pro core instance
 --
 ------------------------------------------------------------------
  ILA_inst : chipscope_ila
    port map (
      CONTROL=> control0,
      CLK=> clk_i,
      TRIG0=> trig0);
end ila_arch;
```

Example 2: ILA Connection in Verilog

```
module example_chipscope_icon (
      input clk_i
      );
 //----------------------------------------------------------------
 //  Local Signals
 //----------------------------------------------------------------
 wire [35: 0] control0;
  wire [7:0] trig0

 //----------------------------------------------------------------
 //
 //  ICON Pro core instance
 //
 //----------------------------------------------------------------
 chipscope_icon ICON_inst
   (
     .CONTROL0(control0));
 //----------------------------------------------------------------
 //
 //  VIO Pro core instance
 //
 //----------------------------------------------------------------
 chipscope_ila ILA_inst0
   (
     .CONTROL(control0),
     .CLK(clk_i),
     .TRIG0(trig0));
endmodule

//----------------------------------------------------------------------
//
//   ICON Pro core module declaration
//
//----------------------------------------------------------------------
module chipscope_icon
 (
   inout [35:0] CONTROL0);
endmodule

//----------------------------------------------------------------------
//
//  ILA Pro core module declaration
//
//----------------------------------------------------------------------
module chipscope_ila
 (
   inout [35: 0] CONTROL,
   in CLK,
   input [7: 0] TRIG0);
endmodule
```

Xilinx Platform Studio

Using the ILA Core in XPS

The ILA core can be inserted into an embedded processor design using the Xilinx Platform Studio (XPS). In this case, the ILA core depends on a BSCAN component instance whose interface is exported by the OPB_MDM peripheral component (see Figure 3).

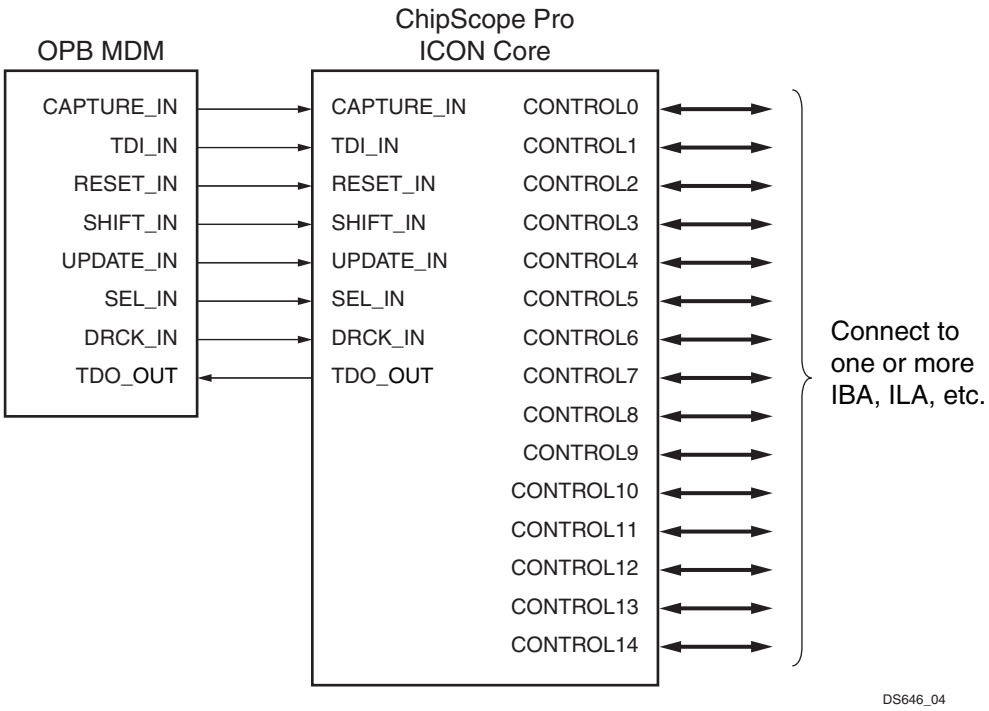

Figure 3: **ICON Core Component in EDK XPS Design**

In XPS, the ICON core is integrated into the tool using a Tcl script. When the EDK Hardware Platform Generator (Platgen) tool runs, the Tcl script is called and the script internally calls the CORE Generator™ (Coregen) tool in the command line mode. The Tcl script provides the CORE Generator software a parameters file (.xco) to generate the ICON core netlist. The Tcl script also generates an HDL wrapper to match the ICON ports based on the core parameters listed in Figure 3. The XST synthesis tool is used for synthesizing the wrapper HDL generated for the ICON core. The NGC netlist outputs from XST and ChipScope Pro Core Generator are subsequently incorporated into the Xilinx ISE® tool suite for actual device implementation.

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.

Ports and Parameters

Ports

The I/O signals of the ILA core listed in Table 2 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.

Table 2: **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).

Table 2: **ILA Interface Ports *(Cont'd)***

Port Name	Direction	Description
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].
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].

Table 2: **ILA Interface Ports *(Cont'd)***

Port Name	Direction	Description
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 Studio, the direction for CONTROL ports is IN.

Parameters

CORE Generator Parameters

The ILA XCO parameters are listed and described in Table 3.

Table 3: **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*>.
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

Table 3: **XCO Parameters *(Cont'd)***

Parameter Name	Allowable Values	Default Value	Description
sample_data_depth	See Table 2, page 10	See Table 2, page 10	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.
example_design	False = do not generate example, True = generate example	false	Enable generation of an example design for the core.

Performance And Resource Utilization

The performance and resource data is listed in Table 4.

Table 4: **Performance and Resource Utilization per Specific Configuration**

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

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.

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
03/24/08	1.0	Release 10.1 (Initial Xilinx release).
04/25/08	1.1	Release 10.1 Service Pack 1.
09/19/08	1.2	Release 10.1 Service Pack 3.
04/07/09	2.0	Release 11.1.
04/19/10	3.0	Release 12.1.
03/01/11	3.1	Updated to v1.04a for the 13.1 release.
06/22/11	3.2	Updated to v1.04a for the 13.2 release.

Notice of Disclaimer