ChipScope Pro Software and Cores User Manual

(ChipScope Pro Software v4.2i)

UG029 (v1.01) April 10, 2002

XILINX®

ChipScope Pro Software and Cores User Manual

UG029 (v1.01) April 10, 2002

The following table shows the revision history for this document.

	Version	Revision
04/09/02	1.0	Initial Xilinx release.

Contents

Chapter 3: Using the ChipScope Pro Analyzer

Figures

Tables

XILINX®

Chapter 1

Introduction

ChipScope Pro Tools Overview

As the density of FPGA devices increases, so does the impracticality of attaching test equipment probes to these devices under test. The ChipScope™ Pro tools integrate key logic analyzer hardware components with the target design inside the Virtex™-II Series device. The ChipScope Pro tools communicate with these components and provide the designer with a complete logic analyzer, without the need for cumbersome probes or expensive test equipment.

ChipScope Pro Tools Description

Table 1-1: **ChipScope Pro Tools Description**

Tool	Description
ChipScope Pro Core Generator	Provides netlists and instantiation templates for the Integrated Controller Pro (ICON Pro) core and the Integrated Logic Analyzer Pro (ILA Pro) core.
ChipScope Pro Analyzer	Provides device configuration, trigger setup, and trace display for the ILA Pro core. The ILA Pro core provides the trigger and trace capture capability. The ICON Pro core communicates to the dedicated Boundary Scan pins.

The ChipScope Pro Analyzer supports the Parallel Cable III and Parallel Cable IV download cables for communication between the PC and FPGA(s) in the JTAG Boundary Scan chain.

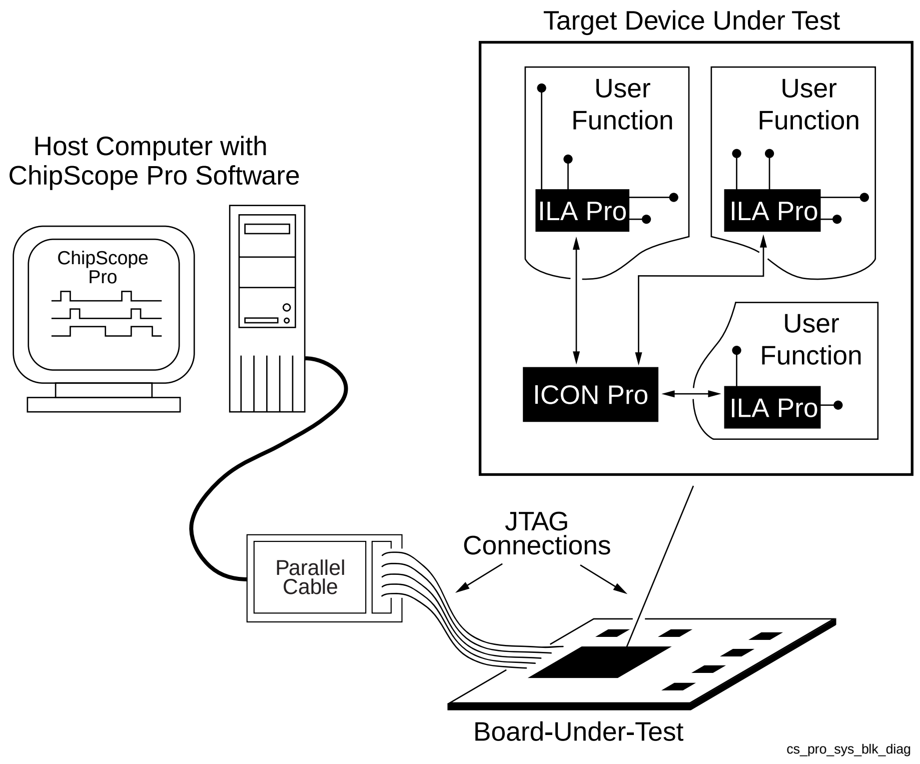


Figure 1-1: **ChipScope Pro System Block Diagram**

Currently, users can place the ICON Pro and ILA Pro cores into their design by generating the cores with the ChipScope Pro Core Generator and instantiating them into the HDL source code. The design is then placed and routed using the Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools. Next, the user downloads the bitstream and analyzes the design with the ChipScope Pro Analyzer software.

The ChipScope Pro Analyzer and corresponding cores contain many features that Xilinx FPGA designers need for thoroughly verifying their logic (Table 1-2). User-selectable data channels range from 1 to 256, and the number of sample sizes ranges from 512 to 16384. Users can change the triggers in real time without affecting their logic. The easy-to-use ChipScope Pro Analyzer leads designers through the process of modifying triggers and analyzing the data.

Table 1-2: **ChipScope Pro Features and Benefits**

Feature	Benefit
1 to 256 user-selectable data channels	Accurately captures wide data bus functionality
User-selectable sample buffers ranging in size from 512 to 16384 samples	Large sample size increases accuracy and probability of capturing infrequent events
Up to four separate trigger ports each with a user-selectable width of 1 to 256 channels (for a total of up to 1024 trigger channels)	Multiple separate trigger ports increase the flexibility of event detection and reduce the need for sample storage
Up to four separate match units per trigger port (for a total of 16 different comparisons per trigger condition)	Multiple match units per trigger ports increase the flexibility of event detection while conserving valuable resources
All data and trigger operations are synchronous to the user clock up to 200 MHz	Capable of high-speed trigger event detection and data capture
Trigger conditions are in-system changeable without affecting the user logic	No need to single step or stop a design for logic analysis
Easy-to-use graphical interface	Guides users through selecting the correct options
Up to 15 independent ILA Pro capture cores per device	Can segment logic and test smaller sections of a large design for greater accuracy
Multiple trigger settings	Records duration and number of events along with matches and ranges for greater accuracy and flexibility
Downloadable from the Xilinx Web site	Tools are easily accessible from the ChipScope Suite

Design Flow

The ChipScope Pro Tools design flow (Figure 1-2) merges easily with any standard FPGA design flow that uses a standard HDL synthesis tool and the Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools.

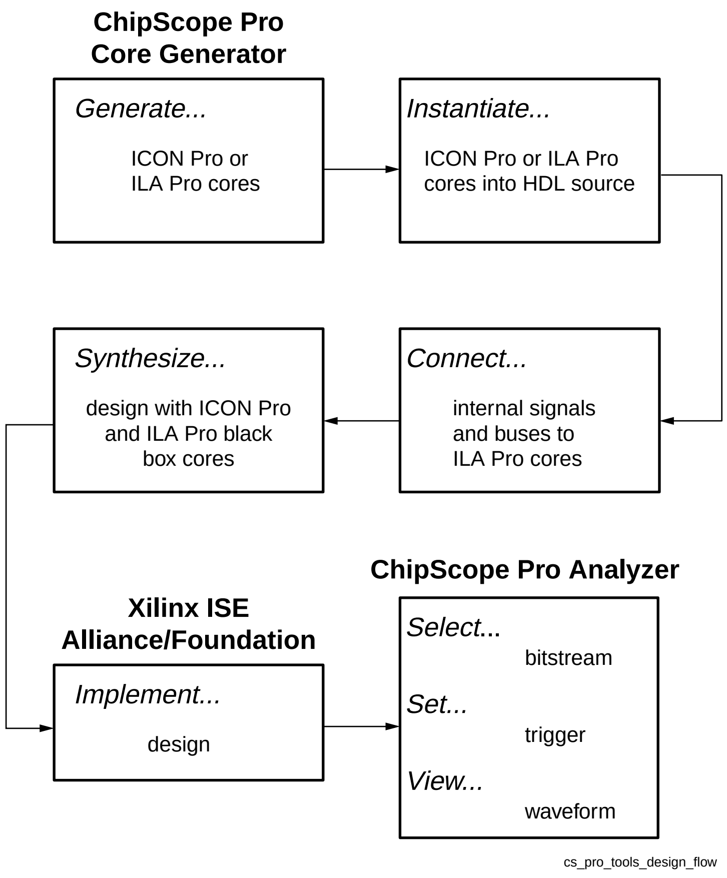

ChipScope Pro Core Generator
Generate... ICON Pro or ILA Pro cores
Instantiate... ICON Pro or ILA Pro cores into HDL source
Synthesize... design with ICON Pro and ILA Pro black box cores
Connect... internal signals and buses to ILA Pro cores
Xilinx ISE Alliance/Foundation
Implement... design
ChipScope Pro Analyzer
Select... bitstream
Set... trigger
View... waveform
cs_pro_tools_design_flow

Figure 1-2: **ChipScope Pro Tools Design Flow**

Trigger Settings

The triggering capabilities of the ILA Pro core include many features that are necessary for detecting elaborate trigger events. These features are described in Table 1-3.

Table 1-3: **ILA Pro Trigger Features**

Feature	Description
Wide trigger ports	Each trigger port can be 1 to 256 bits wide.
Multiple Trigger Ports	Each ILA Pro core can have up to four trigger ports. The ability to support multiple trigger ports is necessary in complex systems where different types of signals or buses need to be monitored using separate match units.
Multiple Match Units per Trigger Port	Each trigger port can be connected to up to four match units. This feature enables multiple comparisons to be performed on the trigger port signals.
Choice of Match Unit Types	The match unit connected to each trigger port can be one of the following types: • **Basic comparator**: - Performs '=' and '<>' comparisons. - Compares up to 8 bits per slice. • **Basic comparator (w/transition)**: - Performs '=' and '<>' comparisons and detects bit-wise transitions. - Compares up to 4 bits per slice. • **Extended comparator**: - Performs '=', '<>', '>', '>=', '<', and '<=' comparisons. - Compares up to 2 bits per slice. • **Extended comparator (w/transition)**: - Performs '=', '<>', '>', '>=', '<', and '<=' comparisons and detects bit-wise transitions. - Compares up to 2 bits per slice. • **Range comparator**: - Performs '=', '<>', '>', '>=', '<', '<=', 'in range', and 'not in range' comparisons. - Compares 1 bit per slice. • **Range comparator (w/transition)**: - Performs '=', '<>', '>', '>=', '<', '<=', 'in range', and 'not in range' comparisons and detects bit-wise transitions. - Compares 1 bit per slice. All match units connected to a given trigger port will be of the same type.
Trigger Output Port	The internal trigger condition of the ILA Pro core can be accessed using the optional trigger output port. This signal can be used as a trigger for external test equipment by attaching the signal to an output pin. However, it can also be used by internal logic as an interrupt, a trigger, or to cascade multiple ILA Pro cores together.

The ability to monitor different kinds of signals and buses in the design requires that multiple trigger ports be used. 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 (as shown in Figure 1-3).

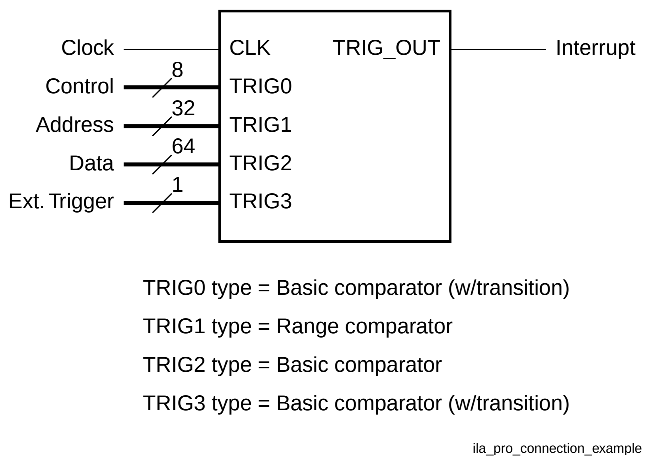

Clock
Control
Address
Data
Ext. Trigger
8
32
64
1
CLK
TRIG0
TRIG1
TRIG2
TRIG3
TRIG_OUT
Interrupt
TRIG0 type = Basic comparator (w/transition)
TRIG1 type = Range comparator
TRIG2 type = Basic comparator
TRIG3 type = Basic comparator (w/transition)
ila_pro_connection_example

Figure 1-3: **ILA Pro Connection Example**

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 Control bus 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 Pro core to your triggering needs while keeping resource usage to a minimum.

Data Capture

Each ILA Pro core can capture data independently from all other ILA Pro cores in the design. Also, the ILA Pro core can capture data using one of two capture modes: *one shot* and *on trigger*.

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 (i.e., a Boolean combination of the individual trigger match unit events) to collect enough data to fill a sample window (up to 16384 samples). The depth of the sample windows can be set up two ways:

- Window depth is powers of 2 from 1 to 16384
- Window depth is any positive integer number from 1 to 16384

In the case where the window depth is a power of two, 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 Pro core is automatically re-armed and will continue to monitor for trigger condition events. This process is repeated until all sample windows of the sample buffer are filled or the user halts the ILA Pro core.

Data Signals

The ILA Pro 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 small 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, you can choose for the data to consist of one or more of the trigger ports. This feature allows you to conserve resources while providing the flexibility to choose what trigger information is interesting enough to capture.

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 ChipScope Pro Analyzer the trigger's position within the window. This trigger mark consumes one extra bit per sample in the sample buffer.

ILA Pro and ICON Pro Core Resource Usage

Tables 1-4, and 1-5 show the ICON Pro core and ILA Pro core resource usage.

Table 1-4: **ICON Pro and ILA Pro Core CLB Resource Usage in Virtex-II Series Devices**

Trigger / Data Width	LUTs	Flip-Flops
8	162	208
16	171	224
32	195	258
64	244	324
128	336	454
256	484	712

Note: A single ILA Pro core with a single trigger port, a single basic match unit, data same as trigger, and 512 data samples was used in this example.

Table 1-5: **ICON Pro and ILA Pro Core Block RAM Resource Usage in Virtex-II Series Devices**

Trigger / Data Width	Data Samples					
	512	1024	2048	4096	8192	16384
8	1	1	2	3	5	9
16	1	2	3	5	9	17
32	2	3	5	9	17	33
64	3	5	9	17	33	65
128	5	9	17	33	65	129
256	9	17	33	65	129	NA

Note: One extra bit per sample is required for the trigger mark (e.g., a trigger/data width of 8 bits requires a sample width of 9 bits, etc.)

Synthesis Requirements

Users can modify many options in the ILA Pro cores without resynthesizing. However, after changing selectable parameters (such as width of the data port or the depth of the sample buffer), the design must be resynthesized with new ILA Pro cores. Table 1-6 shows which design changes require resynthesizing.

Table 1-6: **Design Parameter Changes Requiring Resynthesis**

Design Parameter Change	Resynthesis Required
Change trigger pattern	No
Running and stopping the trigger	No
Enabling the external triggers	No
Changing the trigger signal source	No[1]
Changing the data signal source	No[1]
Changing the ILA Pro clock signal	Yes
Changing the sample buffer depth	Yes

1. The ability to change existing trigger and/or data signal source is supported by the Xilinx ISE Alliance 4.2i (or later) FPGA Editor or Xilinx ISE Foundation 4.2i (or later) FPGA Editor.

System Requirements

Software Tools Requirements

The ChipScope Pro Core Generator requires that Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools be installed on your system.

Note: The Xilinx WebPACK™ implementation tools are not supported by the ChipScope Pro 4.2i software.

Communications Requirements

The ChipScope Pro Analyzer tool can use either the Parallel Cable III or Parallel Cable IV cables to communicate with the target devices in the Boundary Scan chain of the board-under-test.

Table 1-7: **Download Cable Description**

Download Cable	Features
Parallel Cable III	• Uses the parallel port (i.e., printer port) to communicate with the Boundary Scan chain of the board-under-test • Downloads at speeds up to 500 kb/s throughput • Contains an adjustable voltage interface that enables it to communicate with systems and device I/Os operating at 5V down to 3.3V
Parallel Cable IV	• Uses the parallel port (i.e., printer port) to communicate with the Boundary Scan chain of the board-under-test • Downloads at speeds up to 5 Mb/s throughput[1] • Contains an adjustable voltage interface that enables it to communicate with systems and device I/Os operating at 5V down to 1.5V

Note: The Parallel Cable IV cable is available for purchase from the Silicon Xpresso™ Cafe (from www.xilinx.com choose `Buy Online` → `Programming Cables`).

1.The ChipScope Pro Analyzer currently supports only up to 500kb/s throughput with the parallel Cable IV download cable.

Board Requirements

For the ChipScope Pro Analyzer and download cable to work properly with the board-under-test, the following board-level requirements must be met:

- One or more Virtex-II Series target devices must be connected to a JTAG header that contains the TDI, TMS, TCK, and TDO pins
- If another device would normally drive the TDI, TMS, or TDI pins of the JTAG chain containing the target device(s), then jumpers on these signals are required to disable these sources, preventing contention with the download cable
- If using the Parallel Cable III download cable, V_{CC} (3.3V) and GND headers must be available for powering the Parallel Cable III cable
- If using the Parallel Cable IV download cable, VREF (1.5-5.0V) and GND headers must be available for connecting to the Parallel Cable IV cable

Host System Requirements for Windows NT/98/2000

The ChipScope Pro Analyzer and ChipScope Pro Core Generator tools run on PC systems that meet the requirements outlined in Table 1-8.

Table 1-8: **Analyzer and Core Generator PC System Requirements**

OS	Memory	Environment
Windows NT 4.0	128 MB	Java Run-time Environment version 1.3.1_02 (automatically included in ChipScope Pro 4.2i software installation)
Windows 98	64 MB	
Windows 98 SE	64 MB	
Windows 2000	64 MB	

ChipScope Pro Software Installation

Installing ChipScope Pro Software for Windows NT/98/2000

After downloading the ChipScope Pro Tools in the form of a self-extracting executable file (i.e., **`ChipScope_Pro_`*`V`*`_`*`R`*`i_pc.exe`**):

1. Choose **`Start`** → **Run**.
2. Browse for **`ChipScope_Pro_`*`V`*`_`*`R`*`i_pc.exe`**.
3. Choose **Run**.
4. Follow the install wizard instructions.

Notes:

1. The Parallel Cable III / IV driver is automatically installed if it is not already installed on the system.

Installing the Java Run-time Environment for Windows NT/98/2000

The Java Run-time Environment (JRE) version 1.3.1 used by the ChipScope Pro 4.2i tools is automatically included under the ChipScope Pro 4.2i installation directory.

Chapter 2

Using the ChipScope Pro Core Generator

Core Generator Overview

The ChipScope Pro Core Generator tool offers is a graphical user interface to generate the Integrated Controller Pro (ICON Pro) Core and the Integrated Logic Analyzer Pro (ILA Pro) Core. Once the cores are generated, users can use the instantiation templates (that are provided) to quickly and easily insert the cores into their VHDL or Verilog design. After completing the instantiation and running synthesis, users implement the design using the Xilinx implementation tools.

Generating an ICON Pro Core

The Core Generator gives users the ability to define and generate a customized Integrated Controller Pro (ICON Pro) unit to use with one or more Integrated Logic Analyzer Pro (ILA Pro) units in VHDL and Verilog designs. Users can customize control ports (the number of ILA Pro cores to be connected to the ICON Pro Core) and customize the use of the Boundary Scan (BSCAN_VIRTEX2) component that is used for JTAG communication.

After the Core Generator validates the user-defined parameters, it generates an EDIF netlist (***.edn**), a netlist constraint file (***.ncf**), and example code in VHDL and Verilog specific to the synthesis tool used. Users can easily generate the netlist and code examples for use in normal Virtex^{TM}-II Series design flows.

The first screen in the Core Generator offers the choice to generate either an ICON Pro core or an ILA Pro core. Select **`ICON Pro (Integrated Controller)`** core, and click **`Next`**.

General ICON Pro Core Options

The second screen in the Core Generator is used to set up the of the general ICON Pro core options.

Choosing the File Destination

The destination for the ICON Pro EDIF file **`(icon_pro.edn`**) is displayed in the **`Output Netlist`** field. The default directory is the Core Generator install path. To change it, the user can either type a new path in the field, or choose **`Browse`** to navigate to a new destination.

Selecting the Target Device Family

The default target device family is "Virtex-II / Virtex-II Pro". ICON Pro cores generated for this family will work for all Virtex-II Series devices and will not work for Virtex, Virtex-E, Spartan-II and Spartan-IIE devices.

Entering the Number of Control Ports

The ICON Pro core can communicate with up to 15 ILA Pro core units at any given time. However, individual ILA Pro core units cannot share their control ports with any other ILA Pro unit. Therefore, the ICON Pro core needs up to 15 distinct control ports to handle this requirement. Users can select the number of control ports from the **`Number of Control Ports`** pull-down list.

Disabling the Boundary Scan Component Instance

The Boundary Scan (BSCAN_VIRTEX2) primitive component is used to communicate with the JTAG Boundary Scan logic of the Virtex-II Series device. The BSCAN_VIRTEX2 component extends the JTAG TAP interface of the Virtex-II Series device so that up to two internal scan chains can be created. The ChipScope Pro Analyzer communicates with the ChipScope Pro cores by using one of the two internal scan chains (USER1 or USER2) provided by the BSCAN_VIRTEX2 component.

Since ChipScope Pro cores do not use both internal scan chains of the BSCAN_VIRTEX2 component, it is possible to share the BSCAN_VIRTEX2 component with other elements in the user's design. The BSCAN_VIRTEX2 component can be shared with other parts of the design by using one of two methods:

- Instantiate the BSCAN_VIRTEX2 component inside the ICON Pro core and include the unused BSCAN_VIRTEX2 scan chain signals as port signals on the ICON Pro core interface.
- Instantiate the BSCAN_VIRTEX2 component somewhere else in the design and attach either the USER1 or USER2 scan chain signals to corresponding port signals the ICON Pro core interface.

The BSCAN_VIRTEX2 component is instantiated inside the ICON Pro core by default. Use the **`Disable Boundary Scan Component Instance`** check box to disable the instantiation of the BSCAN_VIRTEX2 component.

Selecting the Boundary Scan Chain

The ChipScope Pro Analyzer can communicate with the ChipScope Pro cores using either the USER1 or USER2 boundary scan chains. If the BSCAN_VIRTEX2 component is instantiated inside the ICON Pro core, then you can select the desired scan chain from the **`Boundary Scan Chain`** pull-down list.

Disabling JTAG Clock BUFG Insertion

If the BSCAN_VIRTEX2 component is instantiated inside the ICON Pro core, then it is possible to disable the insertion of a BUFG component on the JTAG clock signal. Disabling the JTAG clock BUFG insertion causes the implementation tools to route the JTAG clock using normal routing resources instead of global clock routing resources. By default, this clock is placed on a global clock resource (BUFG). To disable this BUFG insertion, check select the **`Disable JTAG Clock BUFG Insertion`** check box. This should only be done if global resources are very scarce; placing the JTAG clock on regular routing, even high-speed backbone routing, introduces skew. Make sure the design is adequately constrained to minimize this skew.

Including Boundary Scan Ports

The BSCAN_VIRTEX2 primitive has two sets of ports: USER1 and USER2. These provide an interface to the Boundary Scan TAP controller of the Virtex-II Series device. Since the ICON core uses only one of the USER1 or USER2 scan chain ports for communication purposes, the unused USER2 or USER1 port signals are available for use by other design elements, respectively. If the BSCAN_VIRTEX2 component is instantiated inside the ICON Pro core, then selecting the **`Include Boundary Scan Ports`** check box will provide access to the unused USER1 or USER2 scan chain interfaces of the BSCAN_VIRTEX2 component.

> **Note:** The Boundary Scan ports should be included *only* if the design needs them. If they are included and not used, some synthesis tools do not connect the ICON core properly, causing errors during the synthesis and implementation stages of development.

Selecting the Instantiation Template

After selecting the parameters for the ICON Pro core, you can construct an HDL instantiation template. Click **`Next`** to view the Example Code Generation Options, then select which synthesis tool and language to use. The synthesis tools supported are:

- Exemplar LeonardoSpectrum™
- Synopsys FPGA Compiler™
- Synopsys FPGA Compiler II™
- Synopsys FPGA *Express*™
- Synplicity Synplify®
- XST (Xilinx Synthesis Technology)

Specifically tailored attributes and options are embedded in the HDL instantiation template for the various synthesis tools. To generate the ICON Pro core without any example files, deselect the **`Generate Example Files`** check box.

Generating the Core

After entering the ICON Pro core parameters, click **`Generate Core`** to create the EDIF netlist, NCF constraint file, and applicable code examples. A message window opens, the progress information appears, and the CORE GENERATION COMPLETE message signals the end of the process. The user can then either go back and respecify different options or **`Start Over`** to generate new cores.

Using the ICON Pro Core

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

- Connect one of the ICON Pro core's unused CONTROL* port signals to a control port of only one ILA Pro core instance in the design
- Do not leave any unused CONTROL* ports of the ICON Pro core unconnected as this will cause the implementation tools to report an error. Instead, use an ICON Pro core with the same number of CONTROL* ports as you have ILA Pro cores

Generating an ILA Pro Core

The ChipScope Pro Core Generator allows users to define and generate a customized ILA Pro 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 Pro core, and the width of the data samples (if different from the trigger ports).

After the Core Generator validates the user-defined parameters, it generates an EDIF netlist (**`*.edn`**), a netlist constraint file (**`*.ncf`**), and example code specific to the synthesis tool used. Users can easily generate the netlist and code examples for use in normal Virtex-II Series design flows.

The first screen in the Core Generator offers the choice to generate either an ICON Pro or ILA Pro core. Select ILA Pro (Integrated Logic Analyzer), and click **`Next`**.

General ILA Pro Core Options

The second screen in the Core Generator is used to set up the of the general ILA Pro core options.

Choosing the File Destination

The destination for the ILA Pro EDIF (ila_pro.edn) is displayed in the **`Output Netlist`** field. The default directory is the Core Generator install path. To change it, the user can either type a new path in the field, or choose **`Browse`** to navigate to a new destination.

Selecting the Target Device Family

The target FPGA device family is displayed in the Device Family field. The structure of the ILA core is optimized for the selected device family. Use the pull-down selection to change the device family to the desired architecture. Note that the default target device family is "Virtex-II / Virtex-II Pro". Cores generated for Virtex-II Series devices will not work for Virtex, Virtex-E, Virtex-II, Spartan-II and Spartan-IIE devices.

Selecting the Clock Edge

The ILA unit can use either the rising or falling edges of the CLK signal to trigger and capture data. The **`Clock Settings`** pull-down list is used to select either the rising or falling edge of the CLK signal as the clock source for the ILA core.

ILA Pro Core Trigger Options

Once you are done setting up the general ILA Pro core options, click **`Next`**. This will take you to the third screen in the Core Generator that is used to set up the of the ILA Pro core trigger options.

Selecting the Number of Trigger Ports

Each ILA Pro core can have up to four separate trigger ports that can be set up independently. After using the **`Number of Trigger Ports`** pull-down list to make this selection, you will notice that a group of options will appear for each of the trigger ports. The group of options associated with each trigger port is labeled with **`TRIG`*`n`***, where ***`n`*** is the trigger port number 0 to 3. The trigger port options include trigger width, number of match units connected to the trigger port, and the type of these match units.

Entering the Width of the Trigger Ports

The individual trigger ports are buses that are made up of individual signals or bits. 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 **`TRIG`*`n`*` Width`** field. 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. The results of one or more match units are combined together to form what is called the trigger condition event that is used to start the capture of data. Each trigger port **TRIG*n*** can be connected to 1 to 4 match units by using the **`# Match Units`** pull-down list.

Selecting one match unit conserves resources while still allowing some flexibility in detecting trigger events. Selecting two or more trigger match units allows a more flexible trigger condition equation to be a combination of multiple match units. However, keep in mind that increasing the number of match units per trigger port will increase the utilization of logic resources accordingly.

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 different types of match units are supported by the ILA Pro cores, as shown in Table 2-1.

Table 2-1: **ILA Pro Trigger Match Unit Types**

Type	Match Function	Bits Per Slice[1]	Description
Basic	'=', '<>'	8	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/trans)	'=', '<>', transition detection	4	Can be used for comparing control signals where transition detection (e.g., low-to-high, high-to-low, etc.) is important.
Extended	'=', '<>', '>', '>=', '<', '<='	2	Can be used for comparing address or data signals where magnitude is important.
Extended (w/trans)	'=', '<>', '>', '>=', '<', '<=', transition detection	2	Can be used for comparing address or data signals where a magnitude and transition detection are important.
Range	'=', '<>', '>', '>=', '<', '<=', 'in range', 'not in range'	1	Can be used for comparing address or data signals where a range of values is important.
Range (w/trans)	'=', '<>', '>', '>=', '<', '<=', 'in range', 'not in range', transition detection	1	Can be used for comparing address or data signals where a range of values and transition detection are important.

1.The Bits Per Slice value is only an approximation that is used to illustrate the relative resource utilization of the different match unit types. It should not be used as a hard estimate of resource utilization.

Use the **`TRIGn Type`** pull-down list to select the type of match unit that will apply to all match units connected to the trigger port. Keep in mind that as the functionality of the match unit increases, so does the amount of resources necessary to implement that functionality. This flexibility allows you to customize the functionality of the trigger module while keeping resource utilization in check.

Enabling the Trigger Output Port

The output of the ILA Pro trigger condition module can be brought out to a port signal by checking the **`Enable Trigger Output Port`** check box. The trigger output port is can be used to trigger external test equipment by attaching the port signal to a device pin in the HDL design. The trigger output port can also be attached to other logic or ILA Pro cores in the design to be used as a trigger, an interrupt, or another control signal.

ILA Pro Core Data Options

Once you are done setting up the ILA Pro core trigger options, click **`Next`**. This will take you to the fourth screen in the Core Generator that is used to set up the of the ILA Pro core data options.

Selecting the Data Depth

The maximum number of data sample words that the ILA Pro core can store in the sample buffer is called the *data depth*. The data depth determines the number of data width bits contributed by each block RAM unit used by the ILA unit.

For the Virtex-II and Virtex-II Pro device families, you can set the data depth to one of six values as shown in Table 2-2.

Table 2-2: **Maximum Data Widths (Virtex-II Series Devices)**

	Depth 512	Depth 1024	Depth 2048	Depth 4096	Depth 8192	Depth 16384
1 block RAM	31	15	7	3	1	--
2 block RAMs	63	31	15	7	3	1
4 block RAMs	127	63	31	15	7	3
8 block RAMs	255	127	63	31	15	7
16 block RAMs	--	255	127	63	31	15
32 block RAMs	--	--	255	127	63	31
64 block RAMs	--	--	--	255	127	63
128 block RAMs	--	--	--	--	255	127
216 block RAMs	--	--	--	--	--	215

Note: One extra bit per sample is required for the trigger mark (e.g., a trigger/data width of 7bits requires a full sample width of 8 bits, etc.)

Selecting the Data Type

The data captured by the ILA Pro trigger port can come from two different source types:

- **Data Same as Trigger**
  - The data and trigger ports are identical. This mode is very common in most logic analyzers, since users can capture and collect any data that is used to trigger the core. Individual trigger ports can be selected to be included in the data port.
  - If this selection is made, then the DATA input port will not be included in the port map of the ILA Pro core.
  - This mode conserves CLB and routing resources in the ILA core, but is limited to a maximum aggregate data sample word width of 256 bits.
- **Data Separate from Trigger**
  - 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.

Entering the Data Width

The width of each data sample word stored by the ILA Pro core is called the *data width*. 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 depth is 256 samples.

Selecting the Data-Same-As-Trigger Ports

If the **`Data Same As Trigger`** check box is selected, then a check box for each **`TRIG`*`n`*** port will appear in the data options screen. These check boxes should be used to select the individual trigger ports that will be included in the aggregate data port. Note that selecting the individual trigger ports will automatically update the **`Aggregate Data Width`** field accordingly. A maximum data width of 256 bits applies to the aggregate selection of trigger ports.

Number of Block RAMs

As the data depth and data width selections are changed, the **`Number of Block RAMs`** field will notify you of how many block RAMs will be used by the ILA Pro core. The trigger mark is automatically taken into account when calculating this value.

Selecting the Instantiation Template

After selecting the parameters for the ILA Pro core, you can construct an instantiation template. Click **`Next`** to view the Example Code Generation Options, then select which synthesis tool and language to use. Supported synthesis tools are:

- Exemplar LeonardoSpectrum™
- Synopsys FPGA Compiler™
- Synopsys FPGA Compiler II™
- Synopsys FPGA *Express*™
- Synplicity Synplify®
- XST (Xilinx Synthesis Technology)

Specifically tailored attributes and options are embedded in the HDL instantiation template for the various synthesis tools. To generate the ILA Pro core without any example files, deselect the **`Generate Example Files`** check box.

Generating the Core

After entering the ILA Pro core parameters, click **`Generate Core`** to create the netlist and applicable code examples. A message window opens, the progress information appears, and the CORE GENERATION COMPLETE message signals the end of the process. The user can select to either go back and specify different options or **`Start Over`** to generate new cores.

Using the ILA Pro Core

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

- Connect the ILA Pro core's CONTROL port signal to an **unused** control port of the ICON Pro core instance in the design
- Connect all unused bits of the ILA Pro core's data and trigger port signals to "1". 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 Pro clock signal

Chapter 3

Using the ChipScope Pro Analyzer

Analyzer Overview

The ChipScope Pro Analyzer tool interfaces directly to the ILA Pro and ICON Pro cores. Users can configure their device, choose triggers, and view the results of the capture on the fly. The waveforms and triggers can be manipulated in many ways, providing an easy and intuitive interface to determine the functionality of the design.

Analyzer Interface

The ChipScope Pro Analyzer interface consists of three parts:

- Project tree on the left side of the window
- Message pane at the bottom of the window, and
- Main window area

Both the project tree and the message pane may be hidden by deselecting those options in the **`View`** menu.

The project tree is a graphical representation of the JTAG chain and the ChipScope Pro cores in the devices in the chain. Although all the devices are displayed in the tree, only Virtex-II Series devices can contain ChipScope Pro cores and be operated upon. Leaf nodes in the tree appear when further operations are available. For instance, a folder for each ILA Pro unit appears when that device is configured with an ILA Pro-enabled bitstream.

The message pane displays a scrolling list of status messages. Error messages appear in red.

The main window area can display multiple child windows at once. Such child windows may be trigger and waveform windows. Each window can be resized and moved as needed.

Analyzer Menu Features

Working with Projects

Projects hold important information about the ChipScope Pro Analyzer program state, such as signal naming, signal ordering, bus configurations, and trigger conditions. They allow you to conveniently store and retrieve this information between Analyzer sessions

When you first run the ChipScope Pro Analyzer tool, a new project is created titled ‘new project’. Subsequent launches of the Analyzer will automatically open the last used project. The title bar of the Analyzer displays the project name. If the new project is not saved during the course of the session, a dialog box will appear when the Analyzer is about to exit, asking you if you wish to save the project.

Creating A New Project

To create a new project, select **`File`** → **`New Project.`** A new project called 'new project' will be created and made active in the Analyzer. To save the project under a different name, select **`File`** → **`Save Project`**. The project file will have a **.cpj** extension.

Saving Projects

To rename the current project, or to save a copy to another filename, select **`File`** → **`Save Project As`** (Figure 3-1), type the new name in the dialog box, and click **`Save.`**

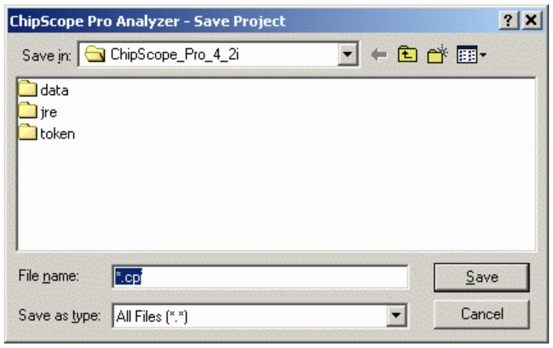


Figure 3-1: **Saving a Project**

Closing and Exiting the Analyzer

To exit the ChipScope Pro Analyzer, select **`File`** → **`Exit`**. The current active project is automatically saved upon exit.

Viewing Options

The project tree on the left of the Analyzer window and the message pane at the bottom of the window can both be hidden or displayed per the user's choice. Both are displayed the first time the Analyzer is launched. To hide the project tree, uncheck it under **`View`** → **`Project Tree`**. To hide the message pane, uncheck it under **`View`** → **`Messages`**.

Opening a Parallel Cable Connection

The ChipScope Pro Analyzer supports the Parallel Cable III and Parallel Cable IV. To open a connection to the Parallel Cable, make sure the cable is connected to one of the computer's parallel ports. Select **`Communication`** → **`Parallel Cable`** (Figure 3-2). The Analyzer prompts you for the port name. Type the printer port name in the port selection box (usually the default LPT1 is correct) and click **`OK`**. If successful, the Analyzer queries the Boundary Scan chain to determine its composition (see **Setting Up the Boundary Scan Chain**, page 3- 3).

If the Analyzer returns the error message "Failed to Open Communication Port", verify that the Parallel Cable III or Parallel Cable IV is connected to the correct LPT port. If you have not installed the Parallel Cable III or Parallel Cable IV driver, follow the instructions in the ChipScope Pro Software installation program to install the required device driver software.

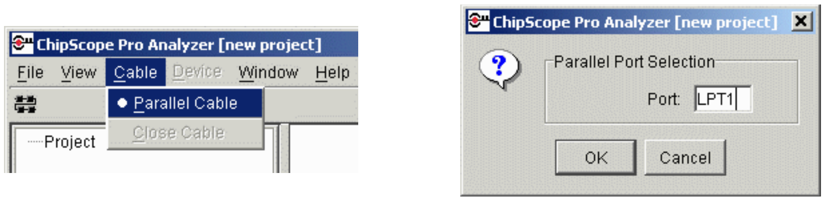


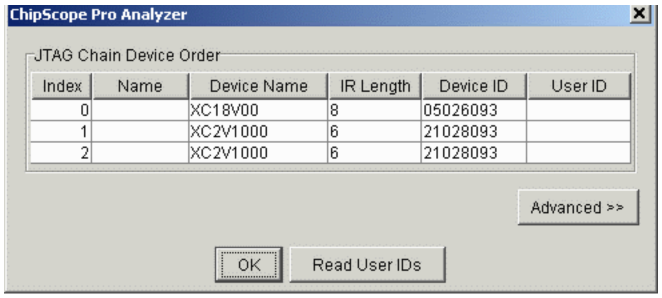
Figure 3-2: **Opening a Parallel Cable Connection**

Configuring the Target Device(s)

You can use the ChipScope Pro Analyzer software with one or more Virtex-II Series devices. The first step is to set up all of the devices in the Boundary Scan chain.

Setting Up the Boundary Scan Chain

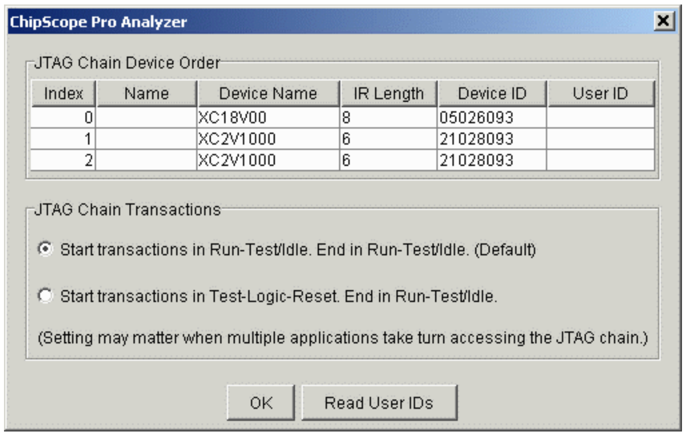
Once the Analyzer has successfully communicated with a download cable, it automatically queries the JTAG chain to find its composition. All Xilinx Virtex/E/EM/II, Spartan-II/IIE, Spartan-XL, 9500/XL/XV, 4000XL/XLA, 18V00, CoolRunner™, CoolRunner-II, and System ACE™ devices are automatically detected. The entire IDCODE can be verified for Virtex-II Series devices. To view the chain composition, select **`Device`** → **`JTAG Device Chain Setup`**. A dialog box appears with all detected devices in order. For devices that are not automatically detected, the IR (Instruction Register) length must be specified to insure proper communication to the ILA and ICON cores This information can be found in the device's BSDL file. The following example has a XC18V00 PROM, and two XC2V1000 Virtex-II FPGAs in a chain (see Figure 3-3). UserID's can be read out of the ILA target devices (only the XC2V1000 devices in this example) by selecting **`Read User IDs`**.



Index	Name	Device Name	IR Length	Device ID	User ID
0		XC18V00	8	05026093	
1		XC2V1000	6	21028093	
2		XC2V1000	6	21028093	

Figure 3-3: **Boundary Scan Setup Window**

The ChipScope Pro Analyzer tool automatically keeps track of the test access port (TAP) state of the devices in the Boundary Scan chain, by default. If the ChipScope Pro Analyzer is used in conjunction with other Boundary Scan controllers (such as the System ACE CF controller or processor debug tools), then the actual TAP state of the target devices can differ from the tracking copy of the ChipScope Pro Analyzer. In this case, the ChipScope Pro Analyzer should always put the TAP controllers into a known state (e.g., the Test Logic Reset state) before starting any Boundary Scan transaction sequences. Clicking on the Advanced button on the Boundary Scan Setup dialog box reveals the parameters that control the start and end states of Boundary Scan transactions (see Figure 3-4). Use the second parameter if the Boundary Scan chain is shared with other Boundary Scan controllers.



Index	Name	Device Name	IR Length	Device ID	User ID
0		XC18V00	8	05026093	
1		XC2V1000	6	21028093	
2		XC2V1000	6	21028093	

[✓ Start transactions in Run-Test/Idle. End in Run-Test/Idle. (Default)]

Start transactions in Test-Logic-Reset. End in Run-Test/Idle.

(Setting may matter when multiple applications take turn accessing the JTAG chain.)

Figure 3-4: **Advanced Boundary Scan Parameters Setup Window**

Device Configuration

The ChipScope Pro Analyzer is able to configure target FPGA devices using the Parallel Cable III or Parallel Cable IV cables in JTAG mode only.

If the target device is to be programmed using the Parallel Cable III or Parallel Cable IV download cables by way of the JTAG port, select **`Device`**→**`Configure`** and the device you wish to configure. Only Virtex-II Series devices can be configured, and show up in the device list for configuration (Figure 3-5).

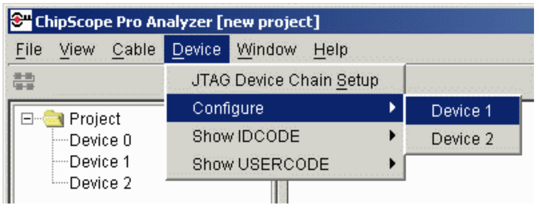


Figure 3-5: **Configuring a Device**

After selecting the configuration mode, the Configuration Selection dialog box (Figure 3-6) opens. This dialog box reflects the configuration choice, and defaults to a blank entry for the configuration file. The Analyzer supports MCS, BIT, and RBT files as inputs.

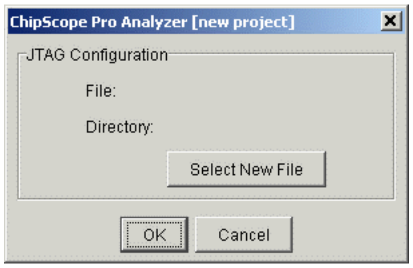


Figure 3-6: **Selecting a Bitstream**

To select the BIT file to download, click on **`Select New File`**. The Open Configuration File dialog box (Figure 3-7) opens. Using the browser, select the device file you want to use to configure the target device. It is important to select a BIT file generated with the proper BitGen settings. Specifically, the **`-g StartupClk:JtagClk`** option must be used in BitGen in order for configuration to be successful.

Once you locate and select the proper device file, click **`Open`** to return to the Configuration Selection dialog box.

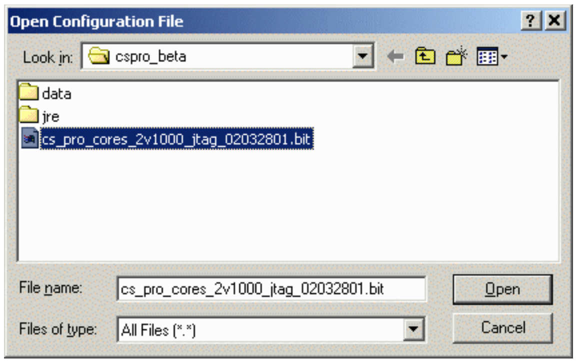


Figure 3-7: **Opening a Configuration File**

Once the mode and BIT file have been chosen, click **`OK`** to configure the device.

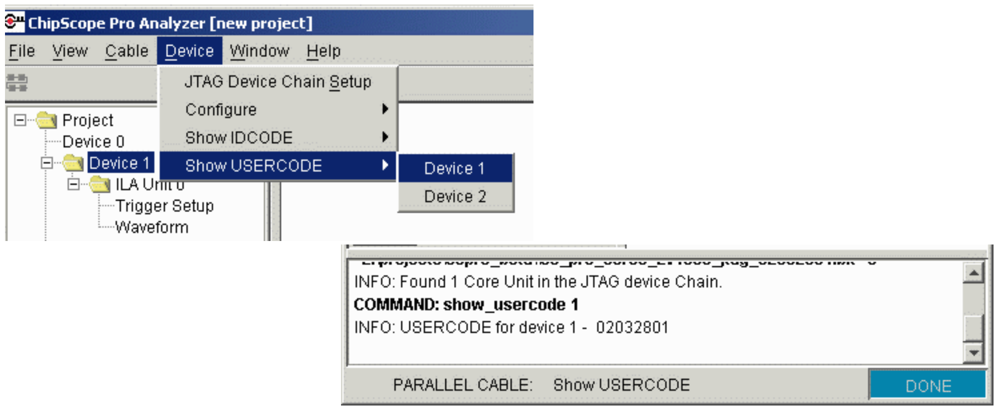
Observing Configuration Progress

While the device is being configured, the status of the configuration is displayed at the bottom of the Analyzer window. If the DONE status is not displayed, a dialog box opens, explaining the problem encountered during configuration. If the download is successful, the target device is automatically queried for ILA Pro cores, and the project tree is updated with the number of cores present. A folder is created for each, and Trigger Setup and Waveform leaf nodes appear under each ILA Pro unit.

Displaying JTAG User and ID Codes

One method of verifying that the target device was configured correctly is to upload the device and user-defined ID codes from the target device.

For instance, to upload and display the user-defined ID code for a particular device (i.e., the 8-digit hexadecimal code that can be set using the BitGen option **`-g UserID`**), select **`Device`** → **Show `USERCODE`** (Figure 3-8). Use **`Device`** → **`Show IDCODE`** to display the fixed device ID code for a particular device. The results of these queries are displayed in the message pane. The IDCODE and USERCODE can also be displayed in the JTAG Chain Setup dialog (**`Device`**→**`JTAG Device Chain Setup`**).



```
INFO: Found 1 Core Unit in the JTAG device Chain.
COMMAND: show_usercode 1
INFO: USERCODE for device 1 -  02032801
```

Figure 3-8: **Displaying the Device USERCODE**

Opening the Trigger Setup Dialog

To set up the trigger for a device that contains ILA Pro cores, select **`Window`**→ **`New Unit Windows`**, and the ILA Pro core desired. A dialog will be displayed for that ILA Pro Unit, and the user can select the Trigger Setup window, Waveform window, both, or neither. Windows cannot be closed from this dialog. The same operation can be achieved by double-clicking on the **`Trigger Setup`** leaf in the project tree, or right-clicking on the **`Trigger Setup`** leaf and selecting **`Open Trigger Setup`**(Figure 3-9).

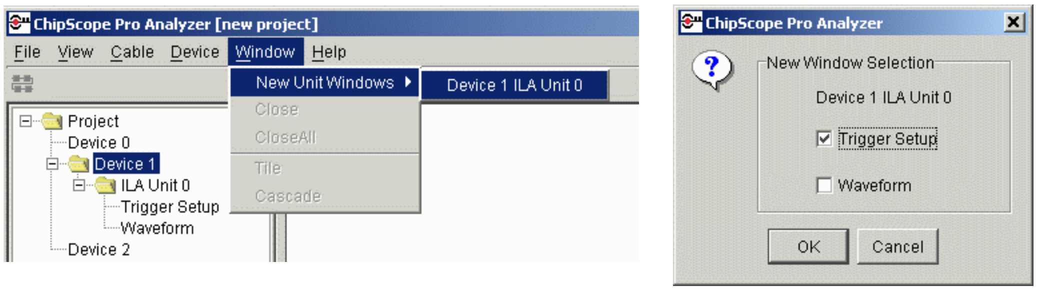


Figure 3-9: **Opening the Trigger Setup dialog**

The Trigger Setup Dialog

Each ILA Pro core has its own trigger setup dialog. The dialog serves two purposes: to display the parameters associated with that ILA Pro core and to provide a graphical interface for the user to set up triggers. The trigger mechanism inside each ILA Pro core can be modified at run-time without having to re-compile the design. The following sections describe how to view the core information and modify the various components that make up the trigger mechanism.

Core Information

All the parameters related to triggering are displayed in the left hand panel of the **`Core Info & Settings`** tab of the trigger setup dialog (Figure 3-10). The core information includes the core type, version, buffer depth, data width, number of trigger ports, and the individual parameters associated with each trigger port. Each match unit type can have transition detection enabled (displayed as '(w/trans)' in the Core Information pane) or disabled. If enabled, the match unit can be set to the following values:

- X : Any value (logical zero or logical one)
- 0 : Logical zero only
- 1 : Logical one only
- R : 0->1 transition only
- F : 1->0 transition only
- T: Any transition

If transition detection is disabled, the match unit bits can only be set to X, 0, and 1.

Basic Match Unit

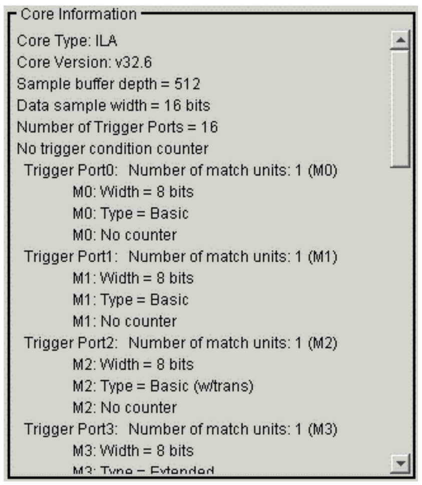
Only an equals (==) or not equals (<>) comparison type is allowed.

Extended Match Unit

Equals (==). not equals (<>), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=) comparison types are allowed. If the comparison type chosen is other than equals or not equals, every bit in the trigger bus must be either 0 or 1 (X, R, F and T are not allowed).

Range Match Unit

All the comparison types of the Extended match unit are allowed, as well as 'In Range' and 'Out of Range'. These two comparison types are used when two separate values are defined (must be made up of only 1's and 0's). The match unit detects when the trigger bus is greater or equal to (>=) the first value, and less than or equal to (<=) the second value.



```
Core Type: ILA
Core Version: v32.6
Sample buffer depth = 512
Data sample width = 16 bits
Number of Trigger Ports = 16
No trigger condition counter
 Trigger Port0:  Number of match units: 1 (M0)
        M0: Width = 8 bits
        M0: Type = Basic
        M0: No counter
 Trigger Port1:  Number of match units: 1 (M1)
        M1: Width = 8 bits
        M1: Type = Basic
        M1: No counter
 Trigger Port2:  Number of match units: 1 (M2)
        M2: Width = 8 bits
        M2: Type = Basic (w/trans)
        M2: No counter
 Trigger Port3:  Number of match units: 1 (M3)
        M3: Width = 8 bits
        M3: Type = Extended
```

Figure 3-10: **Viewing the ILA Pro Core Information**

Capture Settings

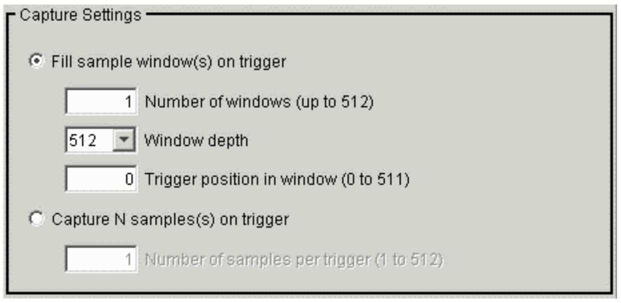
The capture settings pane (Figure 3-11) defines where trigger events occur, and how many trigger events are detected for one entire sample buffer. A window is a contiguous sequence of samples containing one (and only one) trigger event. If an invalid number is entered for any parameter, the text field turns red, and an error is displayed in the message pane.

Fill sample window(s) on trigger

When this radio button is selected, it is possible to choose how many windows should be captured, the number of samples in the window, and the position of the trigger event in each window. The number of samples in the window must be a power of 2.

Capture N sample(s) on trigger

When this radio button is selected, any number of samples may be captured in a single window, and the largest number of complete windows will be captured. For instance, if the sample depth is 512, and the user chooses to have 10 samples captured per trigger, 51 windows will be displayed. For this selection, the trigger event is always the first sample in the window.


Capture Settings

(●) Fill sample window(s) on trigger

1 Number of windows (up to 512)

512 Window depth

0 Trigger position in window (0 to 511)

( ) Capture N samples(s) on trigger

1 Number of samples per trigger (1 to 512)

Figure 3-11: **Capture Settings**

Match Functions

In the **`Match Functions & Trigger Conditions`** tab, both individual match functions and the overall trigger condition are defined. A match function is a definition of a trigger value for a single match unit. A trigger condition is a boolean equation of one or more match functions that determines when the core will capture data. The actual details of where the trigger event occurs in the sample window(s) is defined in the **`Core Info & Settings`** tab.

More than one match function can be defined in the **`Match Functions`** area (Figure 3-12) for a specific match unit, but only one match function for a specific match unit can be included in the active trigger condition. This is helpful when switching between one or more match functions for a specific match unit.

Function Name Field

The Function Name field just provides a mnemonic for a particular match function. MatchFunc*n* is used by default. Double clicking on the Function Name field for a particular match function makes it editable.

Match Unit Field

The Match Unit field selects the match unit that the match function is applied to. Double clicking on an individual field reveals a drop-down list box where the match unit can be chosen. The trigger port corresponding to the match unit is also listed. Clicking on the + sign next to the match unit number will expand that match unit so it is displayed bit by bit in at tree structure. Individual values for each bit can then be viewed and set.

Function Field

The Function field selects which type of comparison is done. Only those comparators that are allowed for that match unit are listed.

Value Field

The Value field selects exactly which trigger value to apply to that match unit. It is displayed according to the Radix field. Double-clicking on the field will make it editable. Place the cursor before the value you want to change, and typing a valid trigger character will overwrite that character. Or, select the field by single-clicking, then proceed by typing the trigger characters.

Radix Field

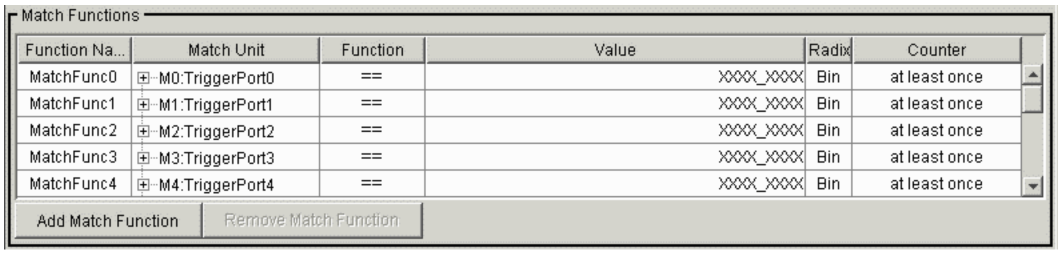
The Radix field selects which radix to display in the Value field. Values are Hex, Octal, Bin, Signed (not allowed for 'In Range' and 'Out of Range' comparisons), and Unsigned.

Counter Field

The Counter field selects how many match function events must occur for the function to be satisfied. If 'at least *n* times' is selected, *n* non-contiguous or *n* non-contiguous events will satisfy the function. If 'for *n* cycles' is selected, only *n* contiguous events will satisfy the function.

Adding and Removing Match Functions

To add a new match function, click the **`Add Match Function`** button. A new function will appear with default name, match unit, and values. Up to 100 match functions can be present for a single core. To delete a particular match function, highlight it by clicking any of the fields, and select **`Remove Match Function`**. The match function will not be removed if it is part of the currently active trigger condition.



Function Na...	Match Unit	Function	Value	Radix	Counter
MatchFunc0	M0:TriggerPort0	==	XXXX_XXXX	Bin	at least once
MatchFunc1	M1:TriggerPort1	==	XXXX_XXXX	Bin	at least once
MatchFunc2	M2:TriggerPort2	==	XXXX_XXXX	Bin	at least once
MatchFunc3	M3:TriggerPort3	==	XXXX_XXXX	Bin	at least once
MatchFunc4	M4:TriggerPort4	==	XXXX_XXXX	Bin	at least once

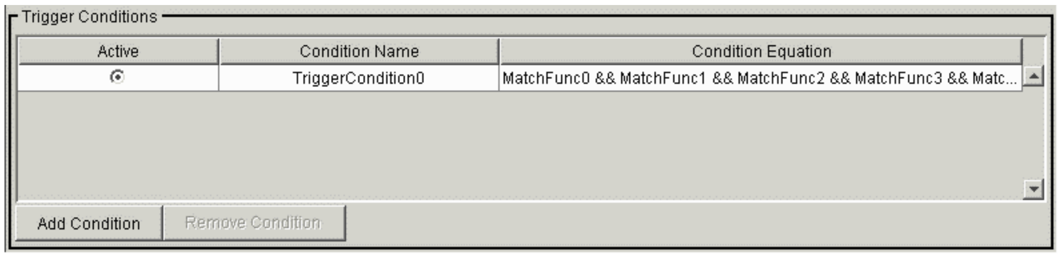
Figure 3-12: **Setting the Match Functions**

Trigger Conditions

The trigger condition area of the **`Match Functions & Trigger Conditions`** tab displays all entered trigger conditions, and the one currently active. The Active radio button indicates which trigger condition is the one selected to be loaded into the core.

Trigger Condition Name Field

The Condition Name field just provides a mnemonic for a particular trigger condition. TriggerCondition*n* is used by default(Figure 3-13).


Trigger Conditions

Active	Condition Name	Condition Equation
●	TriggerCondition0	MatchFunc0 && MatchFunc1 && MatchFunc2 && MatchFunc3 && Matc...

Figure 3-13: **Viewing the Trigger Condition**

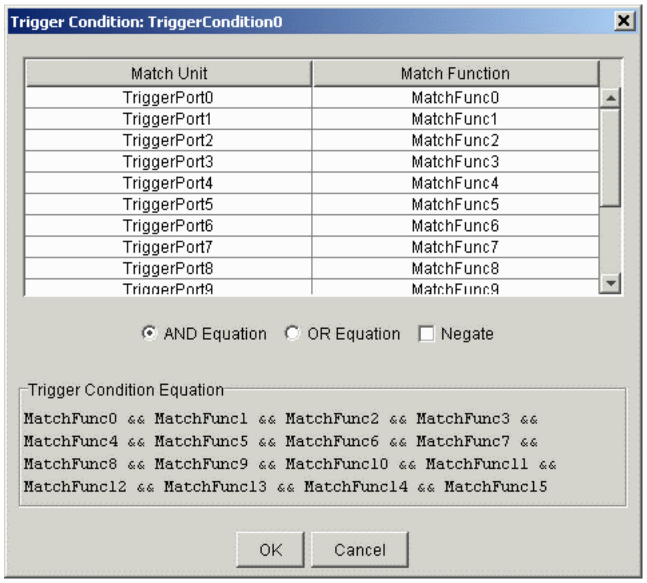
Trigger Condition Equation Field

The Condition Equation field displays the current boolean equation of match functions that make up the overall trigger condition. By default, a logical AND of all the match functions present (one match function for each match unit) is the trigger condition. To change the trigger condition, double-click the Condition Equation field. This will bring up the Trigger Condition dialog.

Trigger Condition Editor Window

The Trigger Condition window has a table of all the match unit and match functions. For each match unit (displayed by the trigger port number) one match function (or none) may be chosen for the trigger condition. To choose a match function for that trigger port, double-click the Match Function field in the table. A drop-down list will appear, listing all the match functions and their inverses for that trigger port and '(unused)'. Choose '(unused)' if you do not want that trigger port to be part of the overall trigger condition.

The overall equation can be the logical AND or OR of all the match functions chosen. Choose AND or OR by selecting that radio button underneath the table. It is possible to negate the entire equation by checking the Negate checkbox. The resulting equation appears in the Trigger Condition Equation pane at the bottom of the window (Figure 3-14).


Trigger Condition: TriggerCondition0

Match Unit	Match Function
TriggerPort0	MatchFunc0
TriggerPort1	MatchFunc1
TriggerPort2	MatchFunc2
TriggerPort3	MatchFunc3
TriggerPort4	MatchFunc4
TriggerPort5	MatchFunc5
TriggerPort6	MatchFunc6
TriggerPort7	MatchFunc7
TriggerPort8	MatchFunc8
TriggerPort9	MatchFunc9

[✓ AND Equation] OR Equation ☐ Negate

Trigger Condition Equation

```
MatchFunc0 && MatchFunc1 && MatchFunc2 && MatchFunc3 &&
MatchFunc4 && MatchFunc5 && MatchFunc6 && MatchFunc7 &&
MatchFunc8 && MatchFunc9 && MatchFunc10 && MatchFunc11 &&
MatchFunc12 && MatchFunc13 && MatchFunc14 && MatchFunc15
```

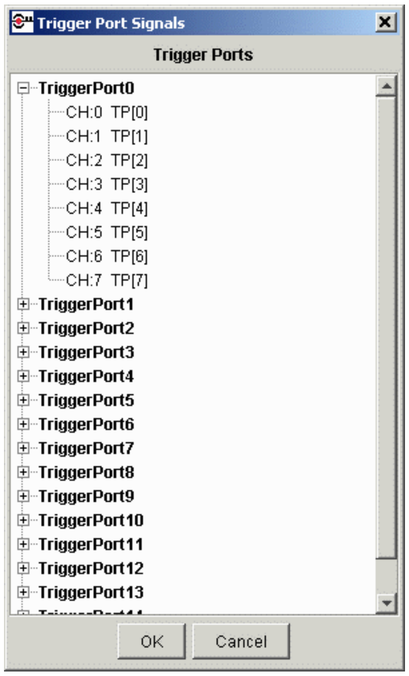
Figure 3-14: **Setting the Trigger Condition Equation**

Saving and Recalling Trigger Setups

All the information in the Trigger Setup window can be saved to a file for recall later with the current project or other projects. To save the current trigger settings, select **`TriggerSetup`** → **`Save Trigger Setup`**. A Save Trigger Setup As file dialog will open, and the trigger settings can be saved in any location, with a **.ctj** extension. To load a trigger settings file into the current project, select **`TriggerSetup`** → **`Read Trigger Setup`**. A Read Trigger Setup file dialog will open, and the user can navigate to the folder where the trigger settings file (with a **.ctj** extension) exists. Once the trigger setting file is chosen, select **`Open`**, and those settings will be loaded into the Trigger Settings window.

Editing Trigger Signals

It possible to name each trigger port and trigger signal on its own, to correspond to the signal or bus names in the design. To edit the names of the trigger signals, select **`TriggerSetup`** → **`Edit Trigger Signals`** (Figure 3-15). The Trigger Port Signals dialog will appear, with all the trigger ports listed as a tree structure. Individual trigger signals can be viewed by clicking the + sign next to a particular trigger port. To change the name of the signal or trigger port, double-click it. This will make the field editable. To save and exit, click **`Ok`**. To exit without saving, click **`Cancel`**.



```
⊟ TriggerPort0
    CH:0  TP[0]
    CH:1  TP[1]
    CH:2  TP[2]
    CH:3  TP[3]
    CH:4  TP[4]
    CH:5  TP[5]
    CH:6  TP[6]
    CH:7  TP[7]
⊞ TriggerPort1
⊞ TriggerPort2
⊞ TriggerPort3
⊞ TriggerPort4
⊞ TriggerPort5
⊞ TriggerPort6
⊞ TriggerPort7
⊞ TriggerPort8
⊞ TriggerPort9
⊞ TriggerPort10
⊞ TriggerPort11
⊞ TriggerPort12
⊞ TriggerPort13
```

Figure 3-15: **Editing Trigger Ports and Signals**

Running and Stopping the Trigger

Running/Arming the Trigger

After setting up the trigger, select **`TriggerSetup`** → **`Run`** to arm it. The trigger stays armed until the trigger condition is satisfied or the user disarms the trigger. Once the trigger condition is satisfied, the trigger automatically disarms and the captured data appears in the waveform window (if displayed).

To force the trigger, select **`TriggerSetup`** → **`Trigger Immediate`**. This causes the ILA unit to ignore the trigger condition and trigger immediately. After the sample buffer fills with data, the trigger disarms and the captured data appears in the waveform window (if displayed).

Stopping/Disarming the Trigger

To disarm the trigger, select **`TriggerSetup`** → **`Stop Acquisition`**. If the trigger condition has been satisfied at least once before the acquisition is stopped, the ChipScope Pro Analyzer program disarms the trigger and displays the captured data. Subsequent selections of **`TriggerSetup`** → **`Run`** cause the trigger to re-arm.

Opening the Waveform Window

To view the waveform for a particular ILA Pro core, select **`Window`**→ **`New Unit Windows`**, and the ILA Pro core desired. A dialog will be displayed for that ILA Pro Unit, and the user can select the Trigger Setup window, Waveform window, both, or neither. Windows cannot be closed from this dialog. The same operation can be achieved by double-clicking on the **`Waveform`** leaf node in the project tree, or right-clicking on the **`Waveform`** leaf and selecting **`Open Waveform`** (Figure 3-16).

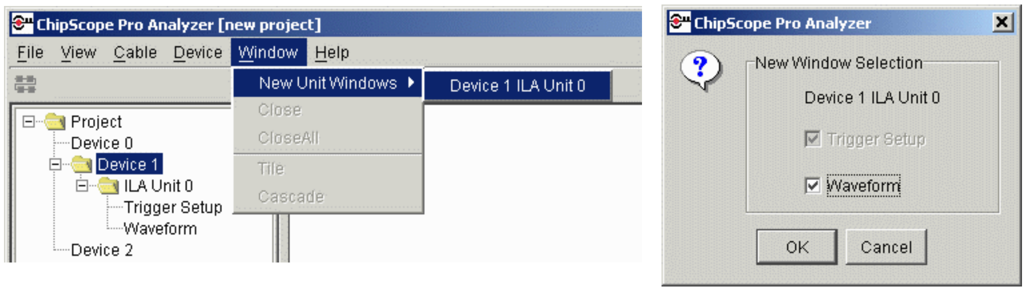


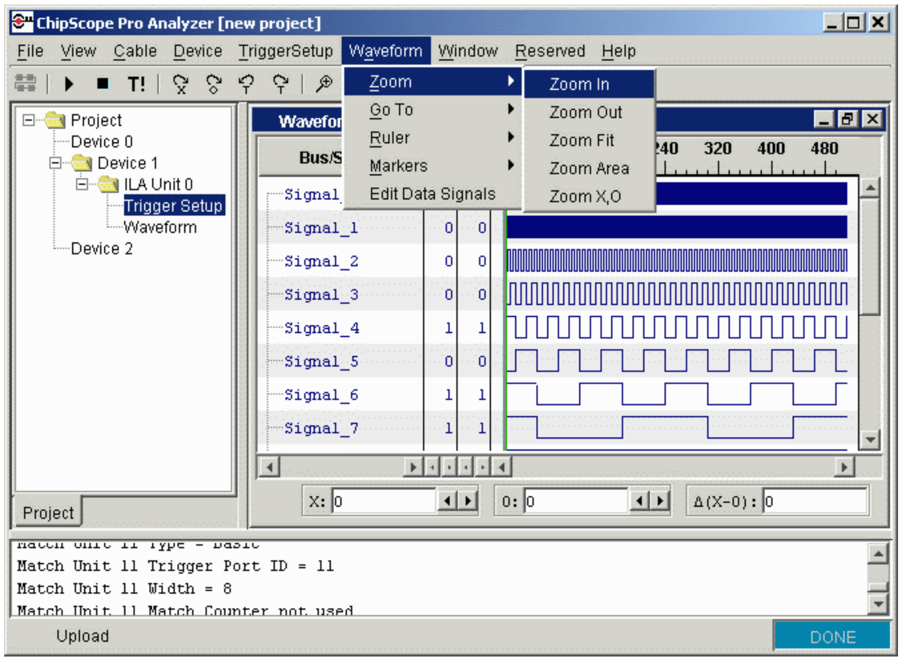
Figure 3-16: **Opening the Waveform Window**

The Waveform Window

The Waveform window displays the sample buffer as a waveform display, similar to many modern simulators and logic analyzers. No signal naming or grouping options are made in the waveform window. Those operations are performed in the Data Signal Browser (**`Waveform`**→ **`Edit Data Signals`**). The only operations performed in the Waveform window itself are waveform viewing and cursor manipulations.

Zooming In and Out

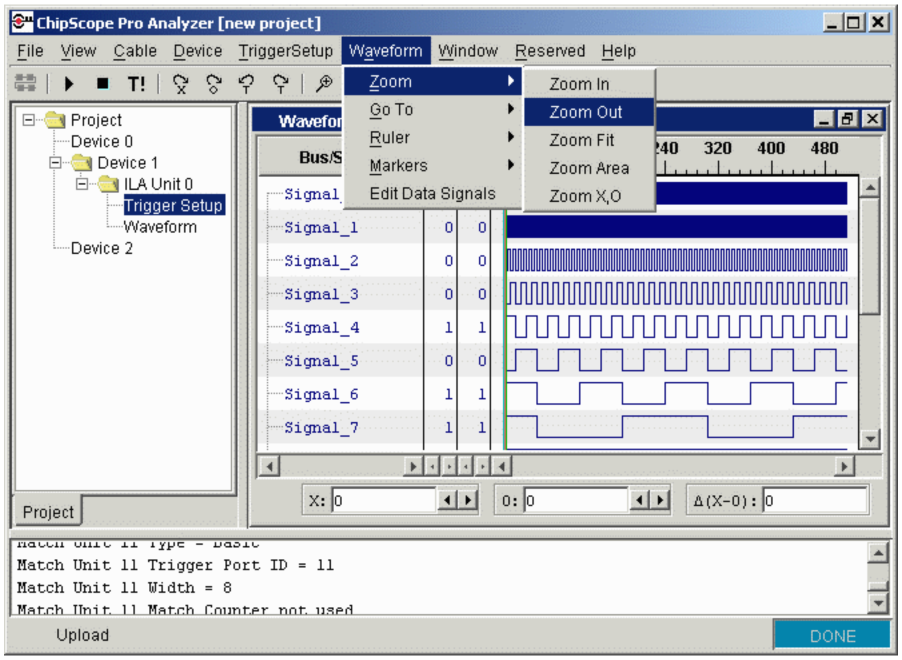
Select **Waveform** → **Zoom** → **Zoom In** to zoom in to the center of the waveform display (Figure 3-17).



```
Match Unit 11 Type = Basic
Match Unit 11 Trigger Port ID = 11
Match Unit 11 Width = 8
Match Unit 11 Match Counter not used
```

Figure 3-17: **Zooming in to the Center of the Waveform Display**

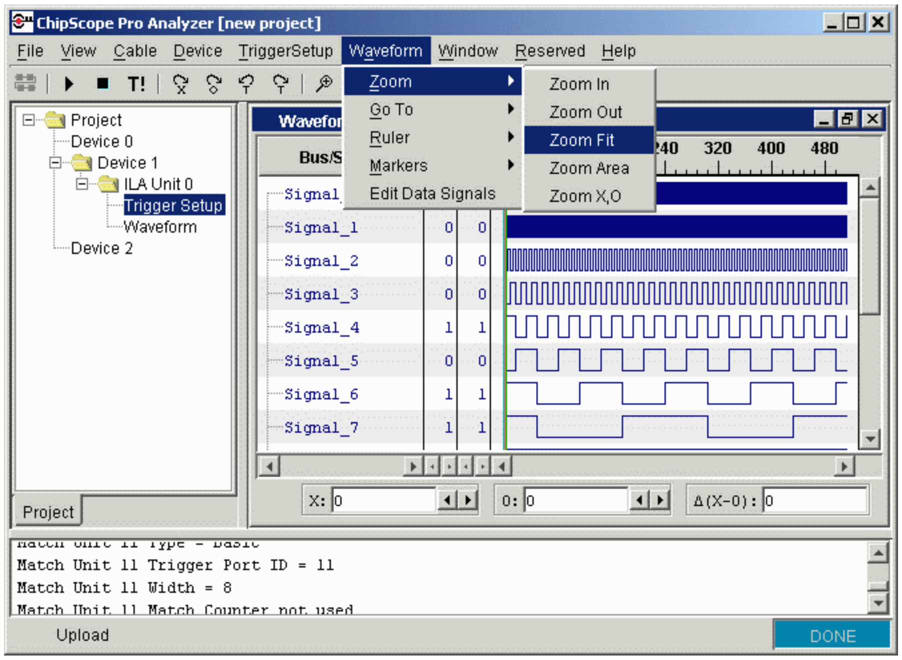
To zoom out from a waveform, use **Waveform** → **Zoom** → **Zoom Out** (Figure 3-18).



```
Match Unit 11 Type = Basic
Match Unit 11 Trigger Port ID = 11
Match Unit 11 Width = 8
Match Unit 11 Match Counter not used
```

Figure 3-18: **Zooming out from the Waveform Display**

To view the entire waveform display select **`Waveform`** → **`Zoom`** → **`Zoom Fit`**(Figure 3-19),



```
Match Unit 11 Type = Basic
Match Unit 11 Trigger Port ID = 11
Match Unit 11 Width = 8
Match Unit 11 Match Counter not used
```

Figure 3-19: **Fitting the Waveform Display in the Window**

To zoom in to a specific area of the waveform, select **`Waveform`** → **`Zoom`** → **`Zoom Area`** (Figure 3-20), then click the left mouse button and drag to select an area of the waveform display. Using this method, the waveform display zooms in to the selected area.

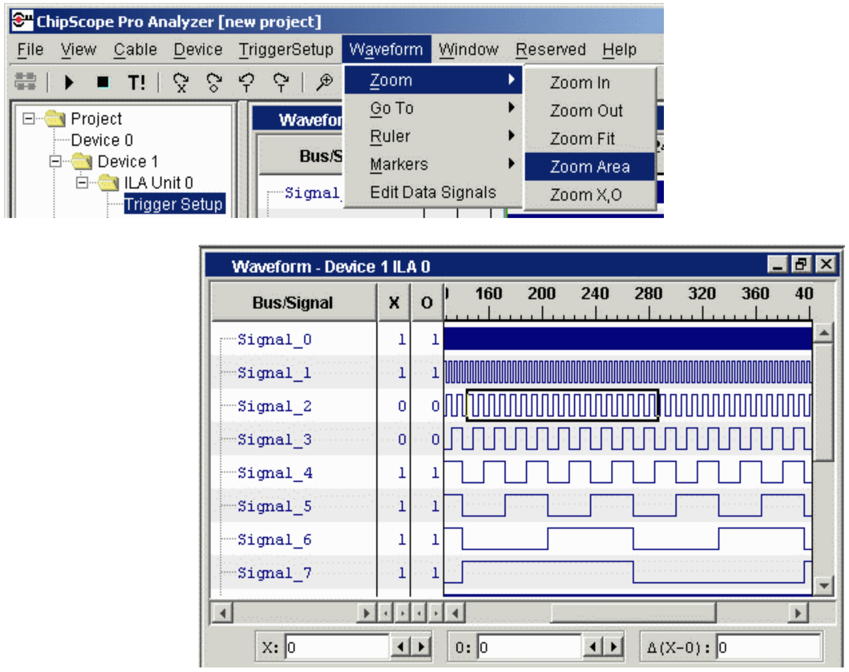
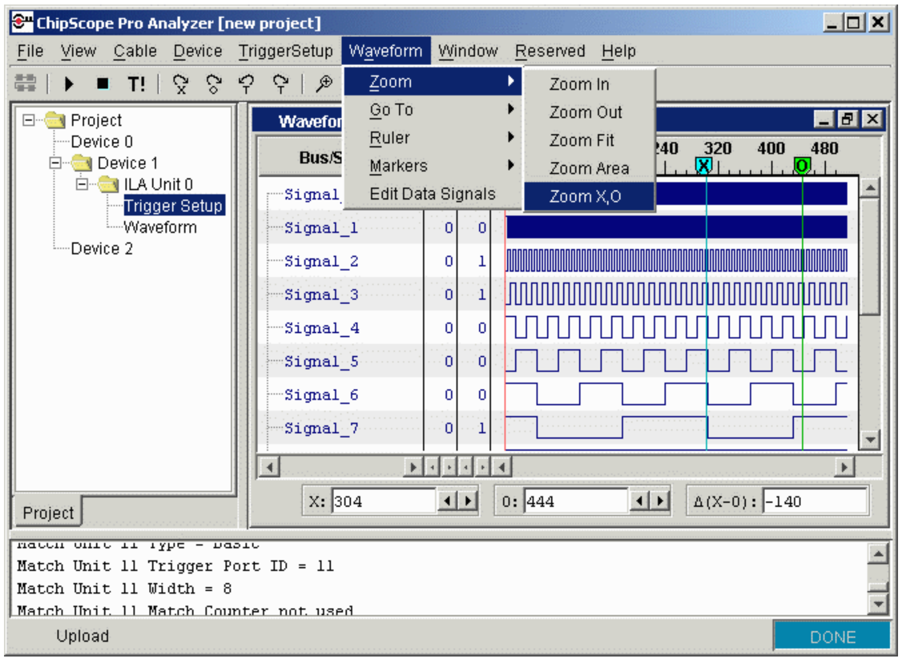


Figure 3-20: **Zooming in to a User-Defined Area**

To zoom in to the space marked by the X and O cursors, select **`Waveform`** → **`Zoom`** → **`Zoom X, O`** (Figure 3-21).



```
Match Unit 11 Trigger Port ID = 11
Match Unit 11 Width = 8
Match Unit 11 Match Counter not used
```

Figure 3-21: **Zooming in to the Area Between the X and O Cursors**

Centering the Waveform

Center the waveform display around a specific point in the waveform by selecting **`Waveform`** → **`Go To`**, then centering the waveform display around the **`X`** and **`O`** markers, as well as the previous or next trigger position (Figure 3-22).

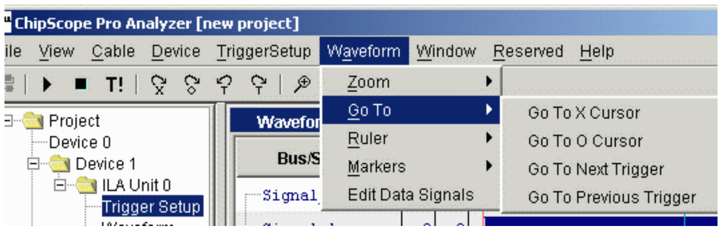


Figure 3-22: **Centering the Waveform on a Marker**

Sample Display Numbering

The x-axis of the waveform can be displayed as the sample number relative to the sample window (default) or by the overall sample number in the buffer. To display the sample number starting over at 0 for each window, select **`Waveform`** → **`Ruler`** → **`Sample # in`**

Window. To display the sample number as an overall sample count in the buffer, select **Waveform** → **Ruler** → **Sample # in Buffer** (Figure 3-23).

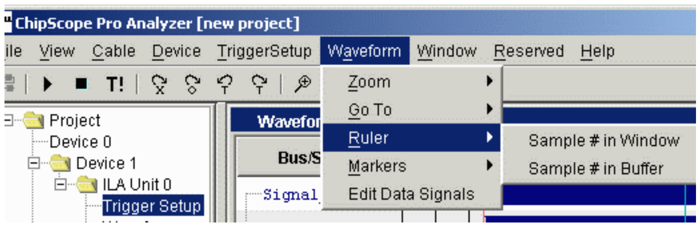


Figure 3-23: **Choosing the Ruler Numbering Scheme**

Displaying Markers

A static red vertical bar is displayed at each trigger position. A static black bar is displayed between two windows, to indicate a period of time where no samples were captured. To not display either of these markers, uncheck them under **Waveform** → **Markers** → **Window Markers** or **Waveform** → **Markers** → **Trigger Markers**.

Editing Signals and Buses - the Data Signal Dialog

Creating, changing, and renaming signals and buses are accomplished in the Data Signal dialog. To open the dialog, select **Waveform** → **Edit Data Signals**.

Grouping Signals Into a Bus

You can group up to 256 signals to form a bus. Hold down the shift key and use the mouse to select one or more ungrouped signals in the Data Signal Browser , then right-click on one of the selected signals. The first right-click menu item is **Group Into Bus**. A new bus is created at the bottom of the dialog box, called BUS0. Subsequently created buses are named BUS1, BUS2, etc. by default (Figure 3-24).

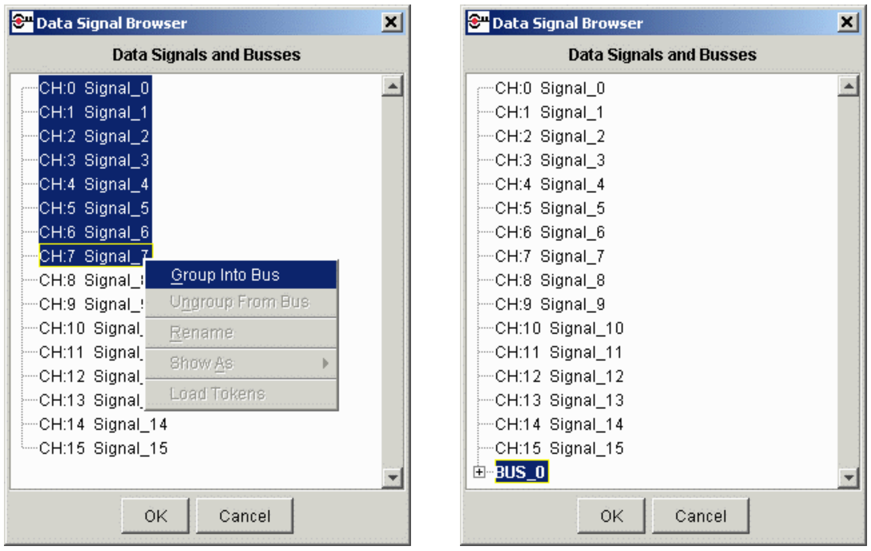


Figure 3-24: **Grouping Signals into a Bus**

Ungrouping Signals From a Bus

To ungroup a signal from a bus, highlight a bus in the **`Data Signal`** dialog, then right-click. Select **`Ungroup From Bus`** from the menu. This removes the bus and places all its signals in the root level (Figure 3-25).

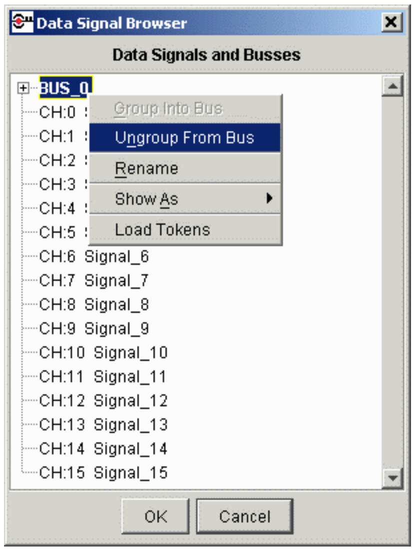


Figure 3-25: **Ungrouping Signals from a Bus**

Moving Buses and Signals

To move buses and signals up and down in the display, highlight a signal or bus, then drag it to a new position. Once a signal or bus is being dragged, red indicator marks appear where the signal or bus will be placed. When the position of a signal in a bus is changed, the bus values are recalculated and the new values appear in the wave display window (Figure 3-26).

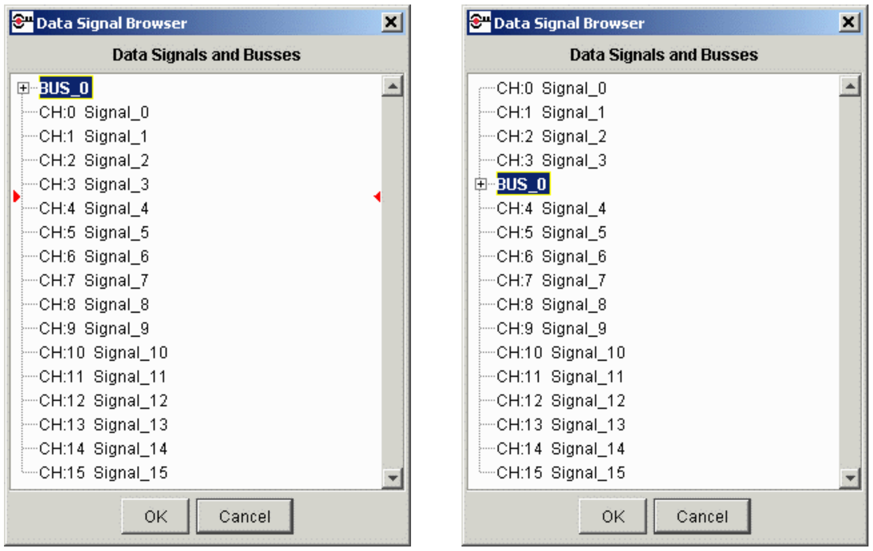


Figure 3-26: **Moving Buses and Signals**

Changing Bus and Signal Names

To rename buses and signals anytime for easy identification, double-click on the bus or signal, or select **Rename** from the right-click menu. A text input box (Figure 3-27) opens, prompting for the new name. Names must be unique and may contain letters, numbers, or underscores. Type the new name, then click **Enter**. To rename the next signal or bus, press the up or down arrow. The previous or next item will become editable.

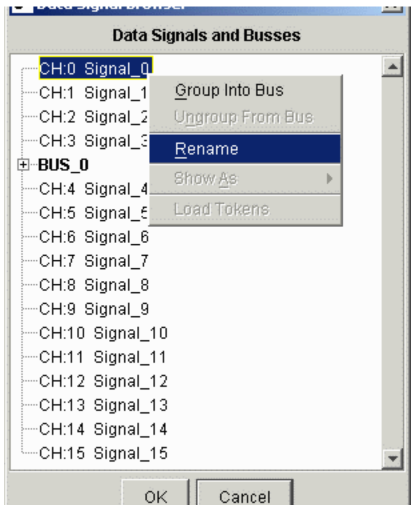


Figure 3-27: **Changing Bus and Signal Names**

Bus Radix Display

Buses can be configured individually to display different radixes in the wave display window. Available bus display options are: hexadecimal, binary, signed decimal, unsigned decimal, octal, ASCII, and Token. By default, bus values are displayed in hexadecimal. ASCII is only available when the bus specified is exactly 8 bits wide. To set the radix for a bus, select a bus in the **Data Signal** dialog, then select **Show As** from the right-click menu. The menu popup (Figure 3-28) allows you to choose from the available bus radix options.

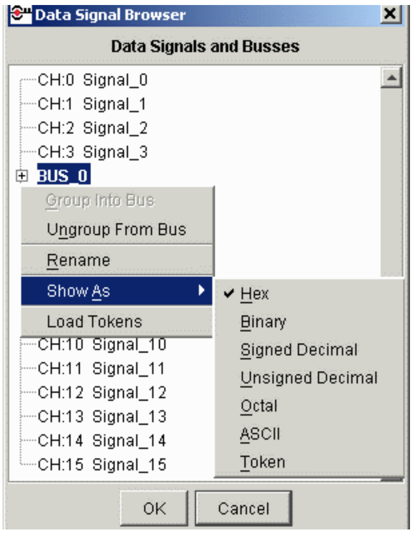
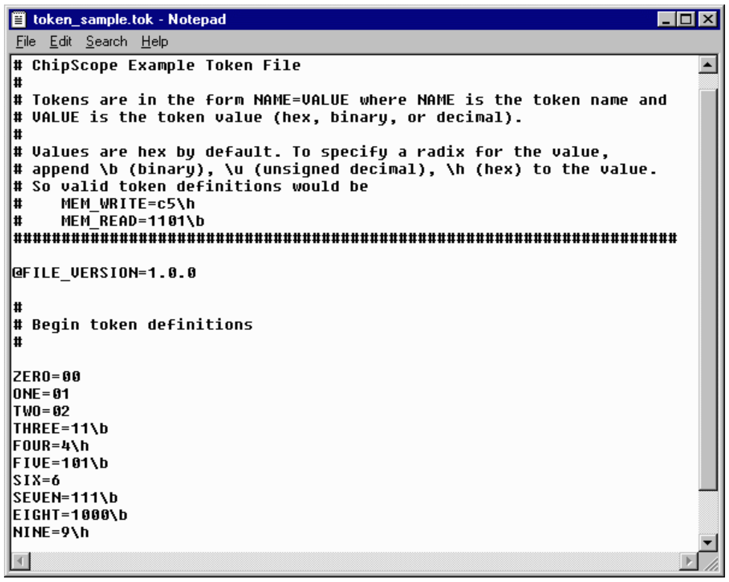


Figure 3-28: **Changing Bus Display Radix**

Using Tokens

Tokens are string labels that can be assigned to a particular bus value. These labels can be very useful in such applications as address decoding and state machines. Tokens are defined in a separate ASCII file, and loaded into the ChipScope Pro Analyzer when appropriate. The token file itself (**.tok** extension) has a very simple format, and can be created or edited in any text editor. An example token file is provided in the token directory in the ChipScope Pro install path (Figure 3-29).



```
# ChipScope Example Token File
#
# Tokens are in the form NAME=VALUE where NAME is the token name and
# VALUE is the token value (hex, binary, or decimal).
#
# Values are hex by default. To specify a radix for the value,
# append \b (binary), \u (unsigned decimal), \h (hex) to the value.
# So valid token definitions would be
#    MEM_WRITE=c5\h
#    MEM_READ=1101\b
#####################################################################

@FILE_VERSION=1.0.0

#
# Begin token definitions
#

ZERO=00
ONE=01
TWO=02
THREE=11\b
FOUR=4\h
FIVE=101\b
SIX=6
SEVEN=111\b
EIGHT=1000\b
NINE=9\h
```

Figure 3-29: **Example Token File**

Tokens are chosen by selecting a bus, then choosing **`Load Tokens`** from the right-click menu. A dialog opens and the user can choose the token file. Once the tokens are loaded, selecting **`Show As`** → **`Token`** from the right-click menu enables the tokens for that particular bus. If the bus is wider than the tokens specify (such as choosing 4-bit tokens for an 8-bit bus) the upper bits are assumed 0 for the tokens to apply. Figure 3-30 shows such a waveform, with the example file in Figure 3-29 applied to a 5-bit bus.

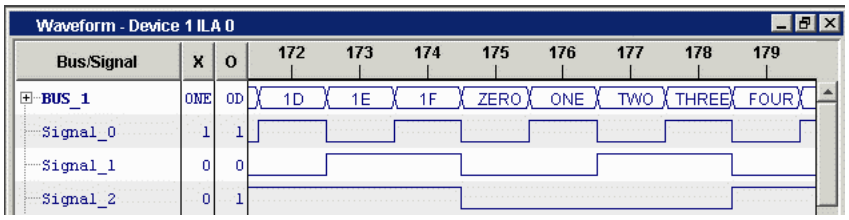
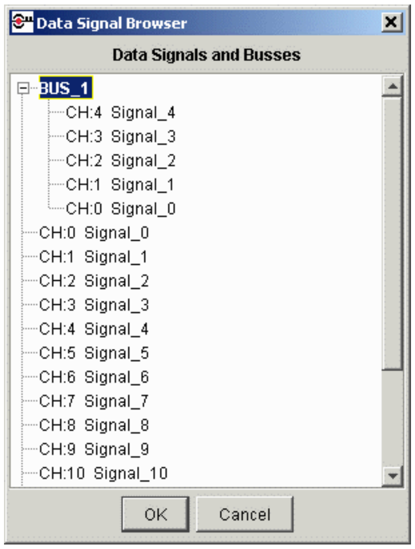


Figure 3-30: **Example Waveform with Tokens**

Bus Bit Ordering

The bits in each bus can be ordered in any fashion. To change the ordering, drag and drop in the individual signals in the bus. The top signal is the least significant. To expand the bus and see the individual signals, click on the + sign next to the bus name (Figure 3-31). To hide the individual signals, click on the - sign next to an expanded bus.



```
BUS_1
    CH:4  Signal_4
    CH:3  Signal_3
    CH:2  Signal_2
    CH:1  Signal_1
    CH:0  Signal_0
CH:0  Signal_0
CH:1  Signal_1
CH:2  Signal_2
CH:3  Signal_3
CH:4  Signal_4
CH:5  Signal_5
CH:6  Signal_6
CH:7  Signal_7
CH:8  Signal_8
CH:9  Signal_9
CH:10  Signal_10
```

Figure 3-31: **Viewing Individual Signals in a Bus**

Viewing the Help Pages

The ChipScope Pro Analyzer help pages contain only the currently opened versions of the ChipScope Pro software and each of the ILA core units. Selecting **`Help`** → **`About: ChipScope Software`** displays the version of the ChipScope Pro Software. Selecting **`Help`** → **`About: ILA Core`** displays the versions of all of the open ILA units.

ChipScope Pro Main Toolbar Features

In addition to the menu options, other ChipScope Pro Analyzer commands are available on a toolbar residing directly below the ChipScope Pro Analyzer menu (Figure 3-32). The second set of toolbar buttons is available only when the Trigger Setup window is open. The third and fourth sets of toolbar buttons are only available when the Waveform window is active.

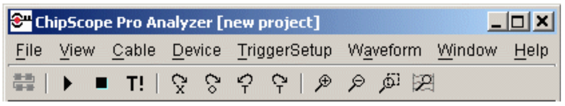

ChipScope Pro Analyzer [new project]
File View Cable Device TriggerSetup Waveform Window Help
T!

Figure 3-32: **Main ChipScope Pro Analyzer Toolbar Display**

The toolbar buttons (from left to right) correspond to the following equivalent menu options:

- **`Open Cable/Search JTAG Chain`** : automatically detects the cable, and queries the JTAG chain to find its composition
- **`Run`** : same as **`TriggerSetup`** → **`Run`** (F5)
- **`Stop`** : same as **`TriggerSetup`** → **`Stop`** (F9)
- **`Trigger Immediate`** : same as **`TriggerSetup`** → **`Trigger Immediate`** (Ctrl + F5)
- **`Go To X Marker`** : same as **`TriggerSetup`** → **`Go To`** → **`X Marker`**
- **`Go To O Marker`** : same as **`TriggerSetup`**→ **`Go To`** → **`O Marker`**
- **`Go To Previous Trigger`** : same as **`Waveform`** → **`Go To`** → **`Previous Trigger`**
- **`Zoom In`** : same as **`Waveform`** → **`Zoom In`**
- **`Zoom Out`** : same as **`Waveform`** → **`Zoom Out`**
- **`Zoom Area`** : same as **`Waveform`** → **`Zoom Area`**
- **`Fit Window`** : same as **`Waveform`** → **`Fit Window`**