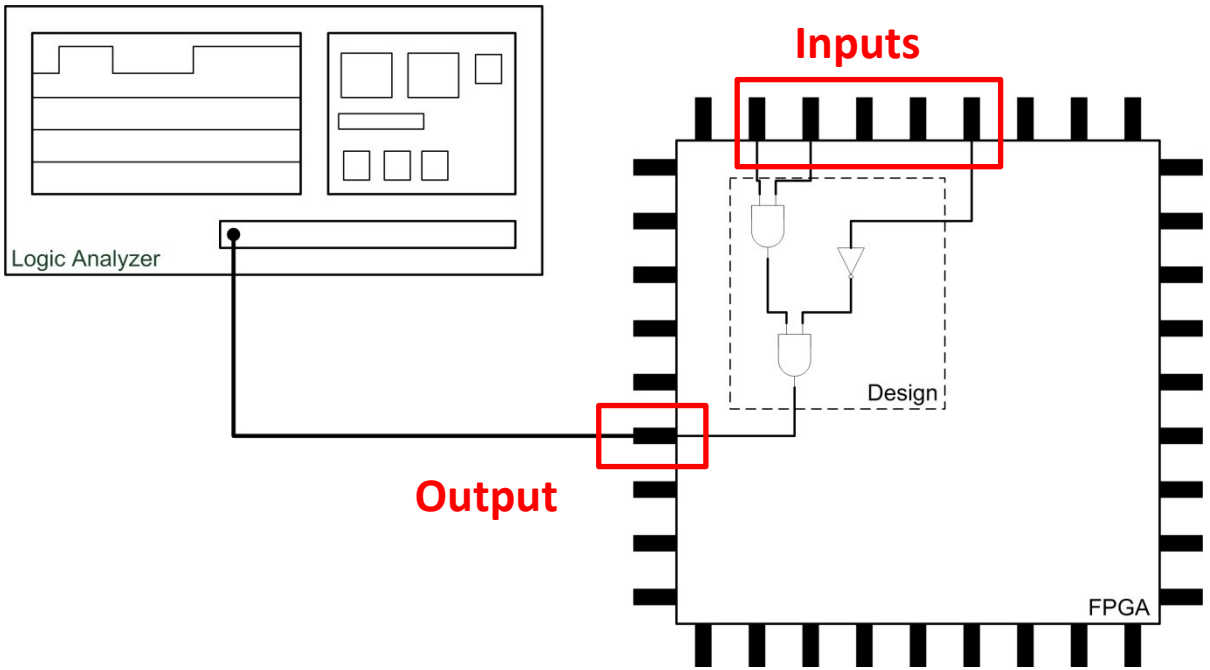
COE 758 – Xilinx ISE 13.4 Tutorial 2

ChipScope Pro Overview

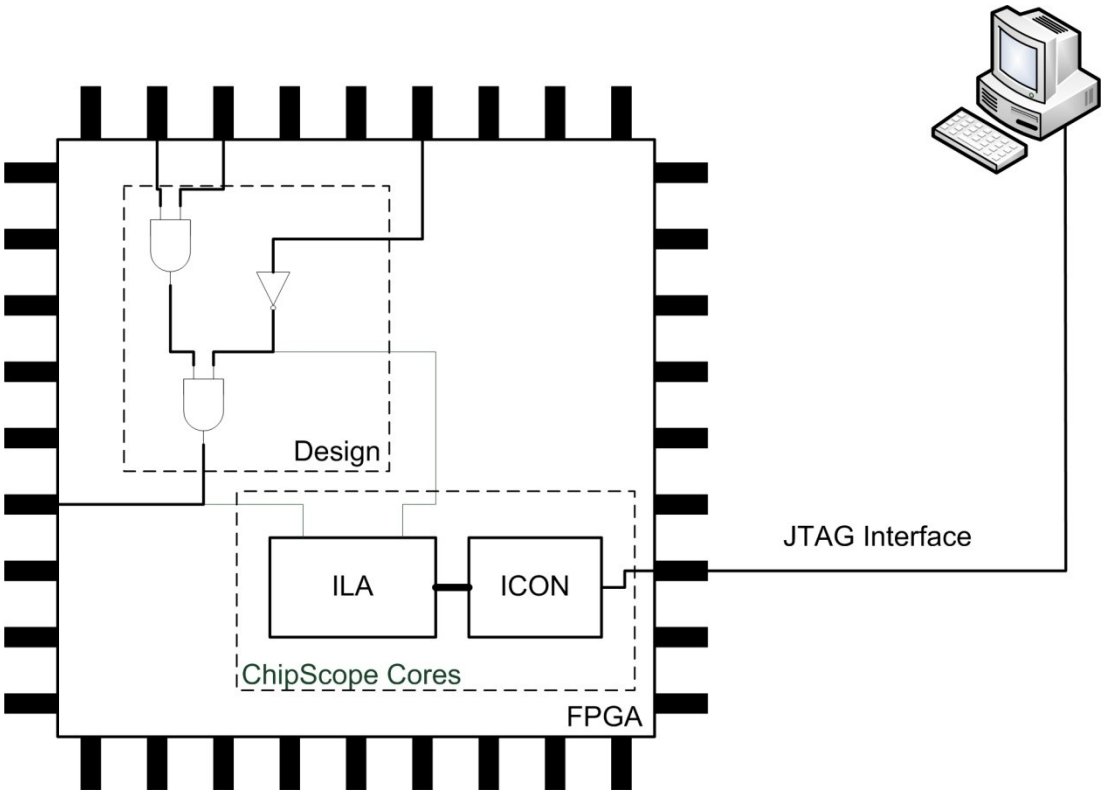
Integrating ChipScope components into a design

Conventional Testing and Debug of Digital Designs



Traditionally, the debugging and verification of digital circuits is accomplished using Logic Analyzers, connected to a given design via the physical I/O pins of the FPGA being used (this applies equally to situations where ASIC implementation is used). If internal signals need to be monitored, these signals must be assigned to additional I/O pins, which can increase the cost and complexity of the design.

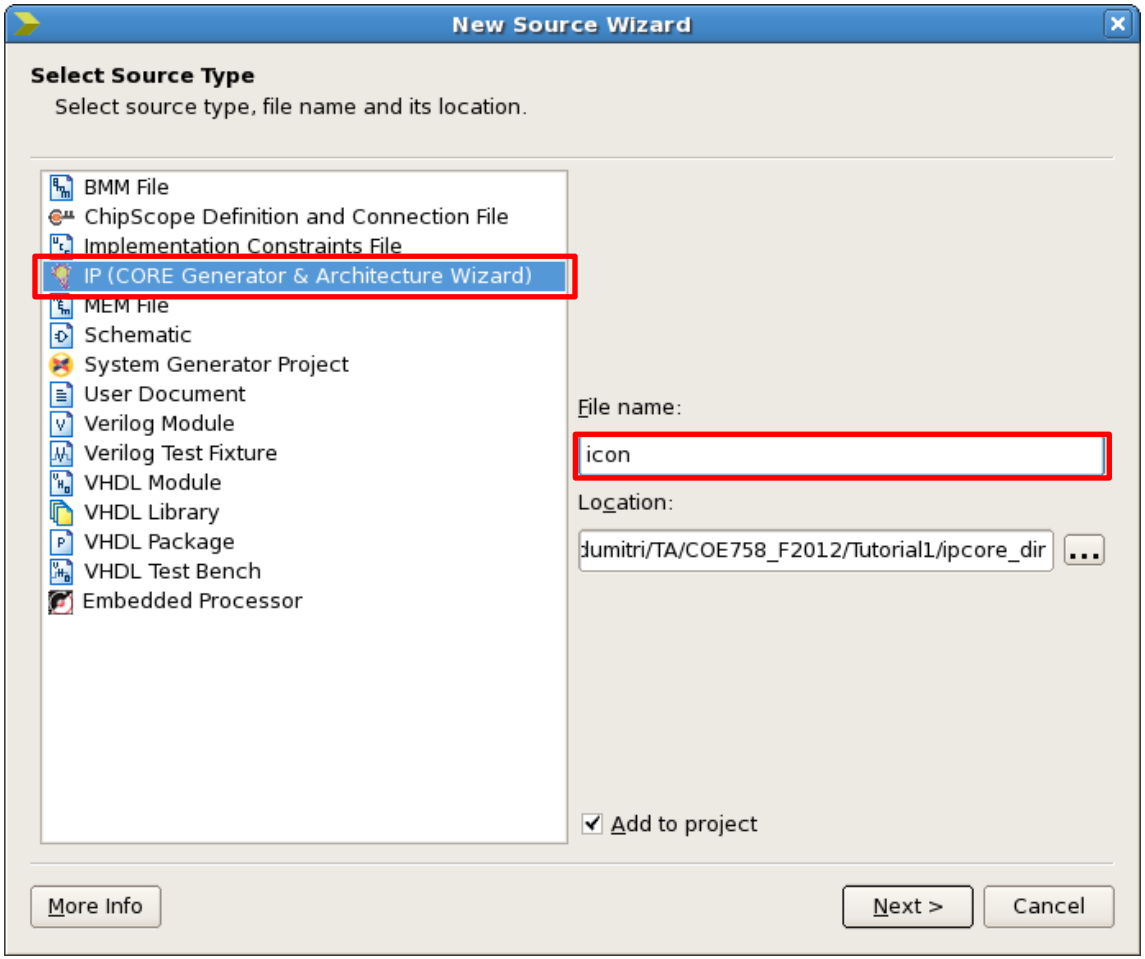
Testing and Debug of Digital Designs using ChipScope



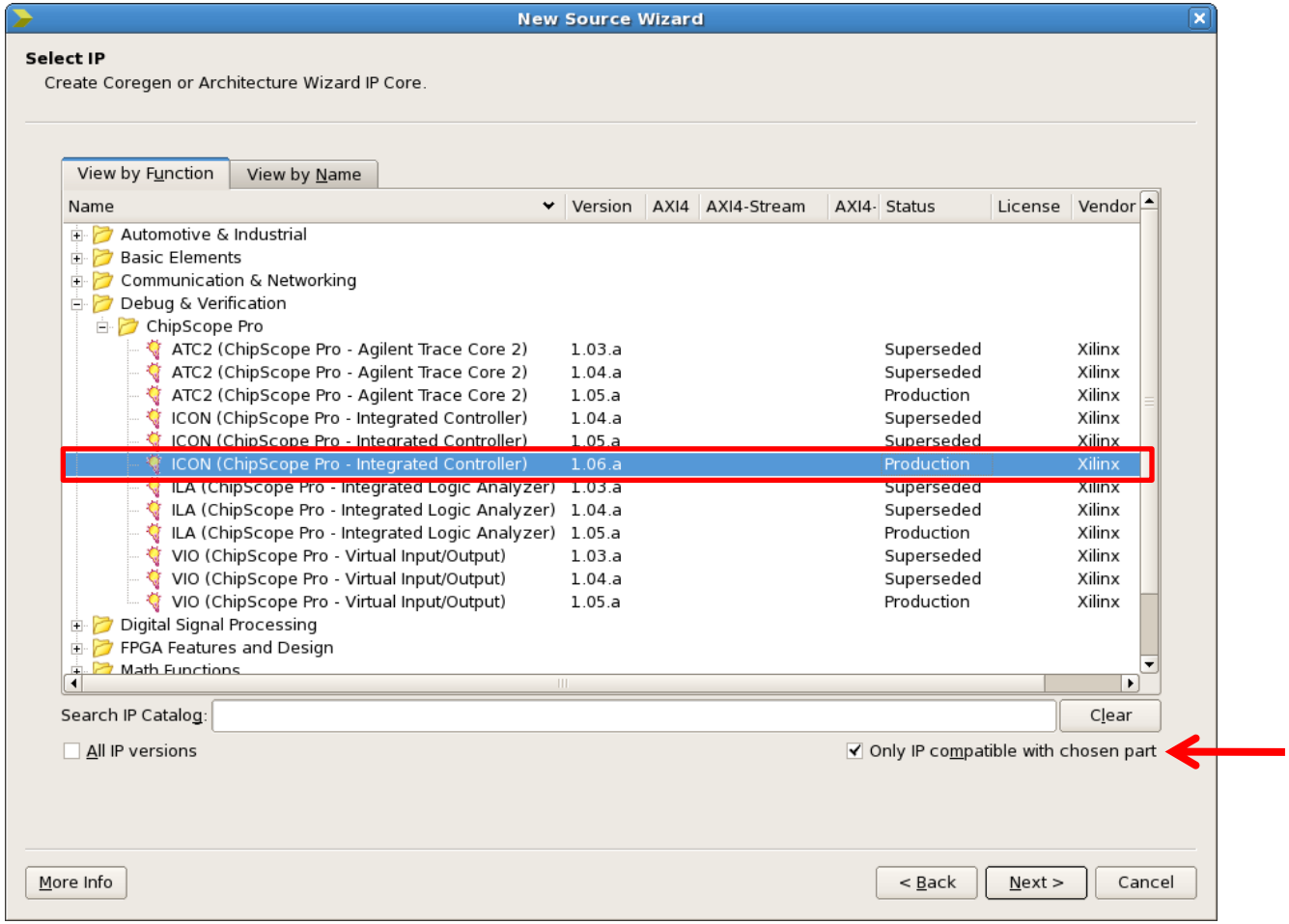
The use of FPGAs allows for an alternative way to perform testing and debug activities which excludes the use of a logic analyzer. The method relies on incorporating specific circuit *cores* into the design, which can monitor the rest of the system, and send information to a host computer. The ChipScope Pro collection of IP cores accomplishes this task; by instantiating the Integrated Controller (ICON) and one or more Integrated Logic Analyzers (ILAs) into a design, any signals in the design can be sampled; the data thus captured can then be sent to a host computer for analysis.

Tutorial 2 Overview

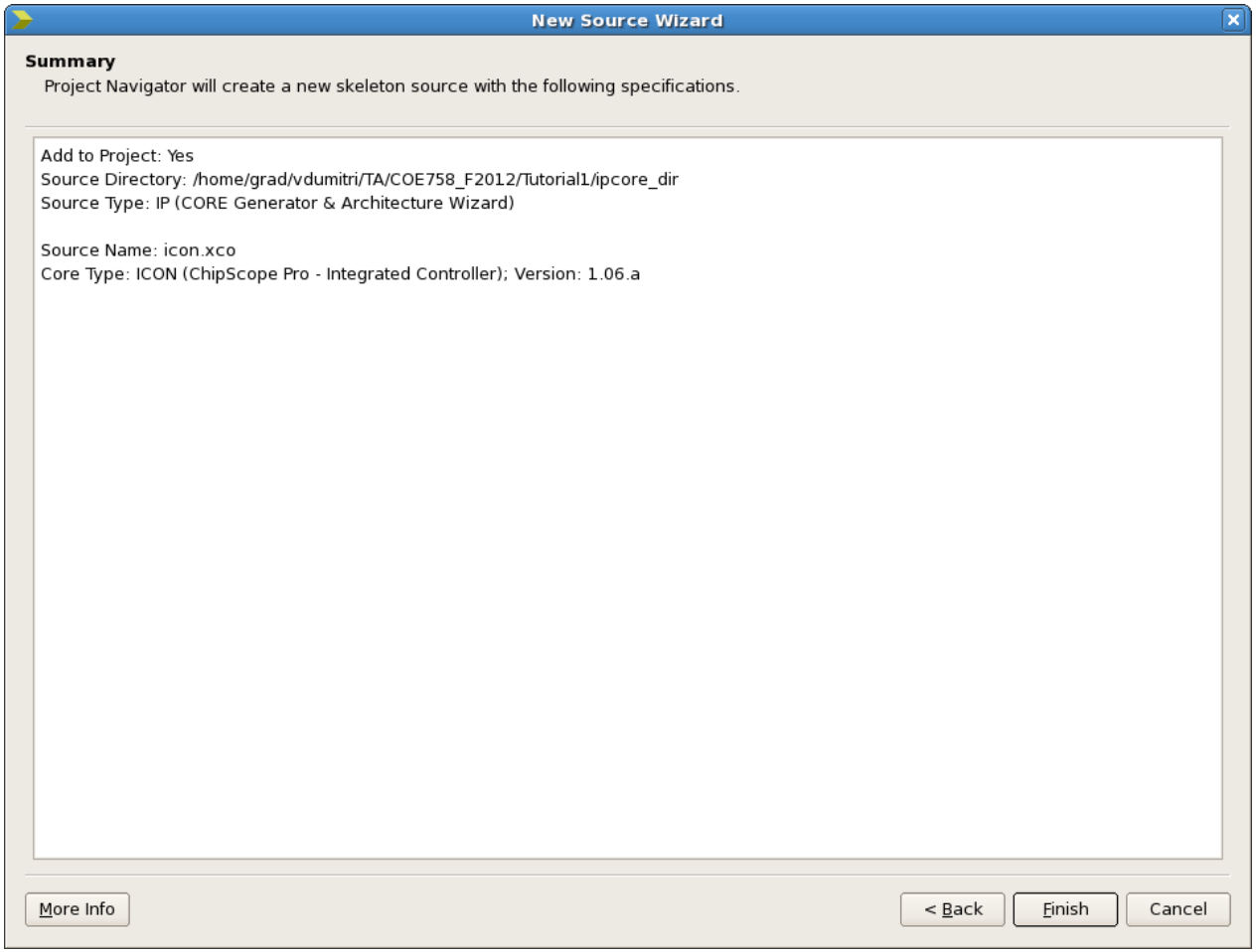
- The remainder of this Tutorial will outline how to generate and instantiate ChipScope Pro Cores into an existing design.
- Once the design is updated with ChipScope cores, the ChipScope Analyzer will be used to sample data from a running on-chip design.
- Note that this tutorial updates the project created in Tutorial 1. You should either use the same project, or make a new copy of the project for the current Tutorial.



The first step is to add the ICON integrated controller to the project. Start by adding a new source to the project (as shown in Tutorial 1). When selecting the source type, select **IP (CORE Generator & Architecture Wizard)**, and name the new component accordingly. Finally, click **Next**, which will launch the Core selection wizard.

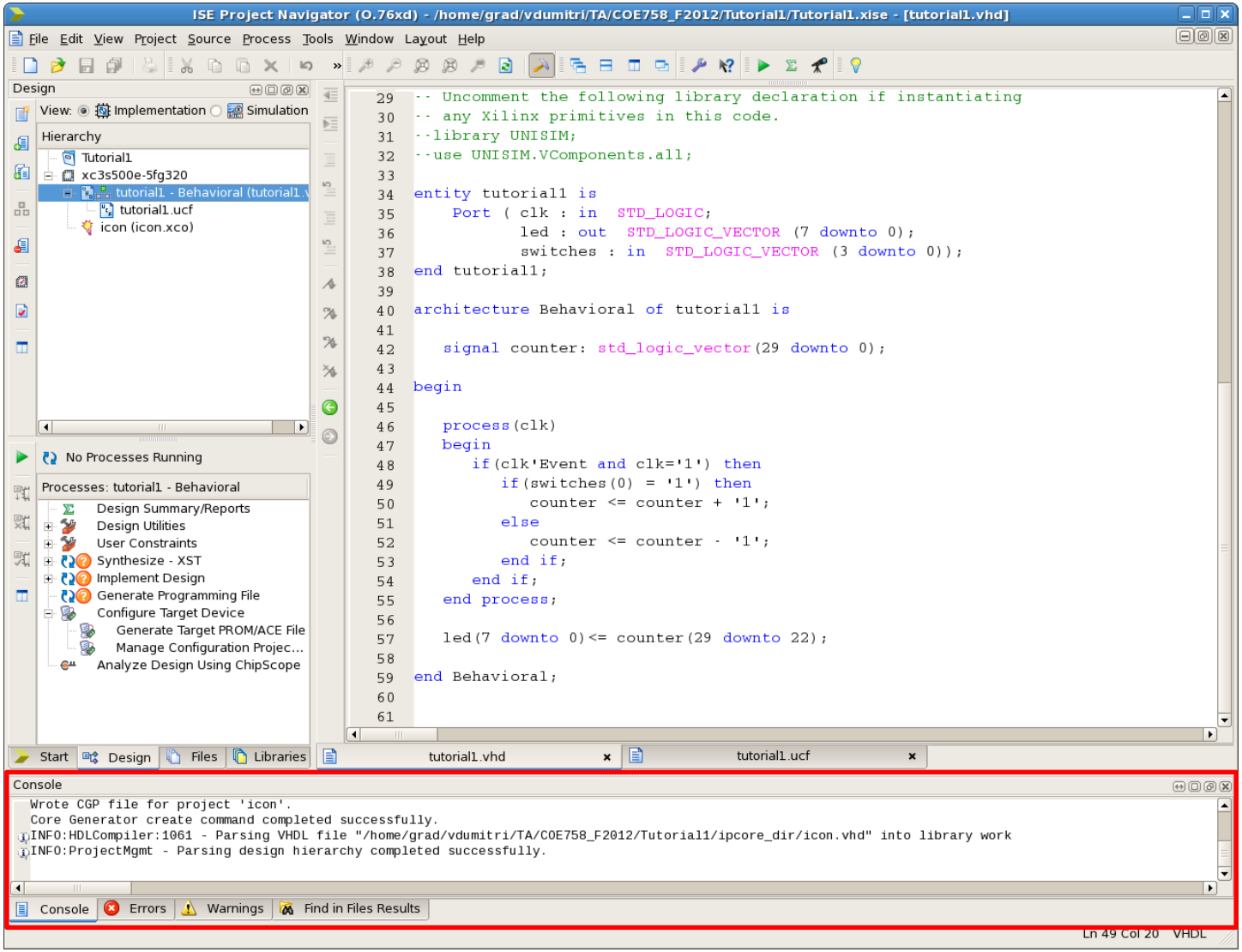


Once the IP core selection wizard appears (shown above), expand the **Debug & Verification** folder, then the **Chipscope Pro** folder, and select the appropriate version of the ICON core, as shown above; for the purposes of this tutorial (and the rest of the course), always select Chipscope cores with the status **Production**. Also ensure that the option "**Only IP compatible with the chosen part"** is selected. Once the correct core is selected, click **Next**.

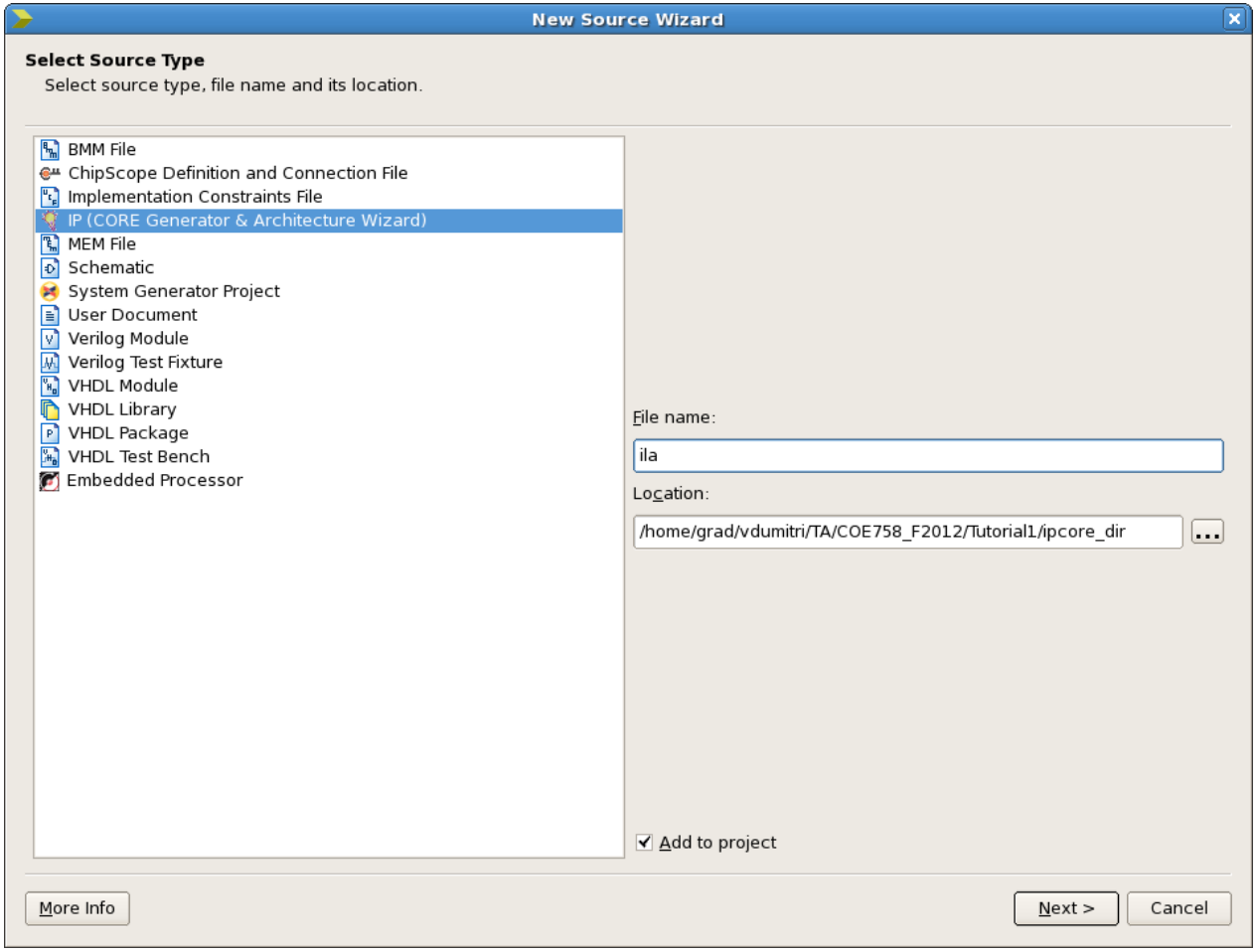


Once the correct core is selected, a final summary window will appear (similar to the one above). This summary lists what core will be generated, where it will be stored, and what it is called. Click **Finish** to launch the core generator for the ICON integrated controller.

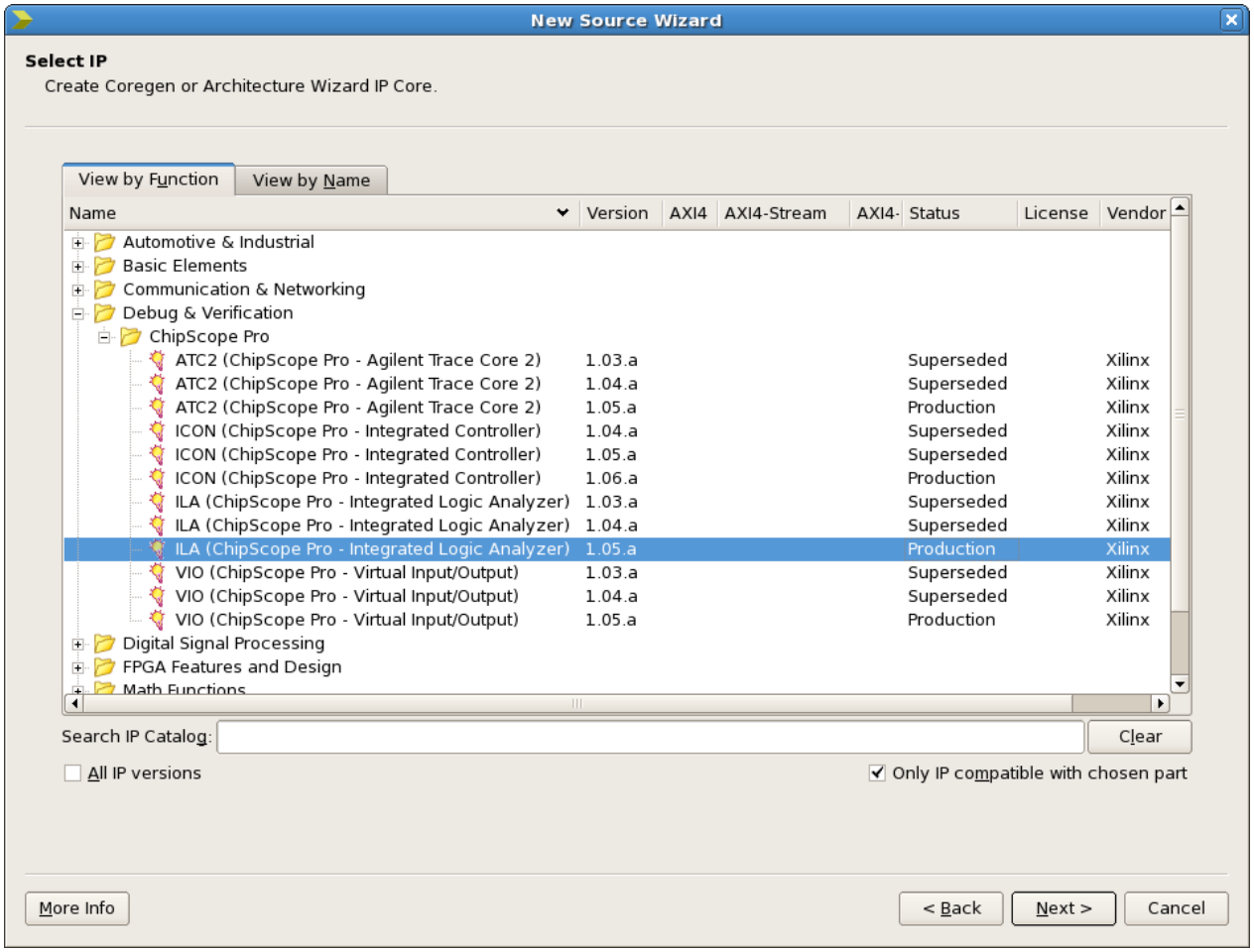
The ICON core generator, shown above, allows you to configure certain parameters associated with your ICON core. The most important of these parameters is the **Number of Control Ports**, which should be set to **1** for this tutorial. All other options should be left as they are. Click **Generate** to actually generate the core.



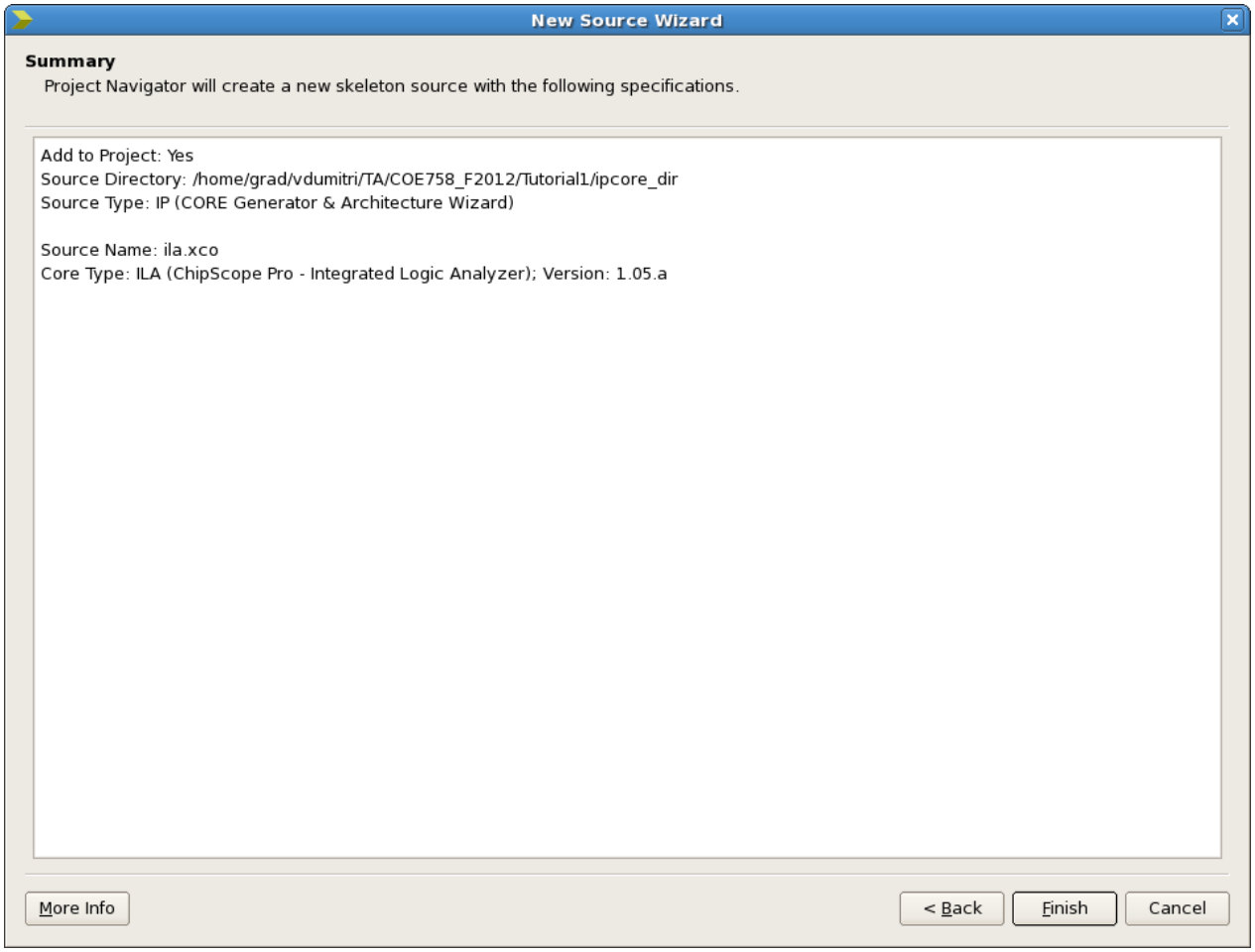
As the core is generated, status messages will be printed to the console. Once the core generation process completes, the message highlighted above will be displayed. Your project should now look similar to what is shown above.



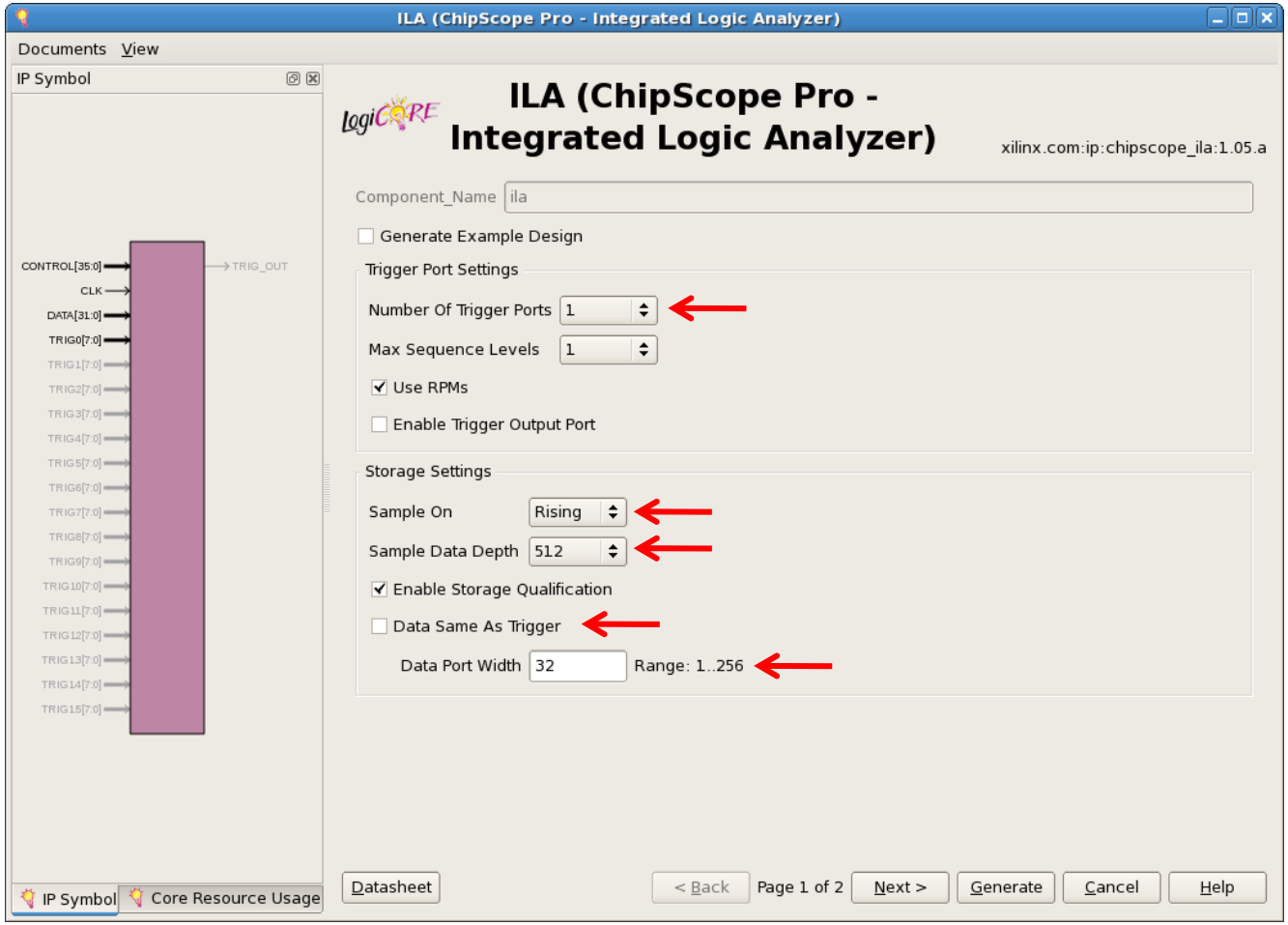
The next step is to add an Integrated Logic Analyzer (ILA) core to your project. As before, add a new source to the project, once again select **IP (Core Generator & Architecture Wizard)** as the source type, and name your core appropriately. Finally, click **Next**.



Open the same folder as you did for the ICON core, and select the ILA core, as shown above. Once again, ensure the status of the core is **Production**. Once the correct core is selected, click **Next**.

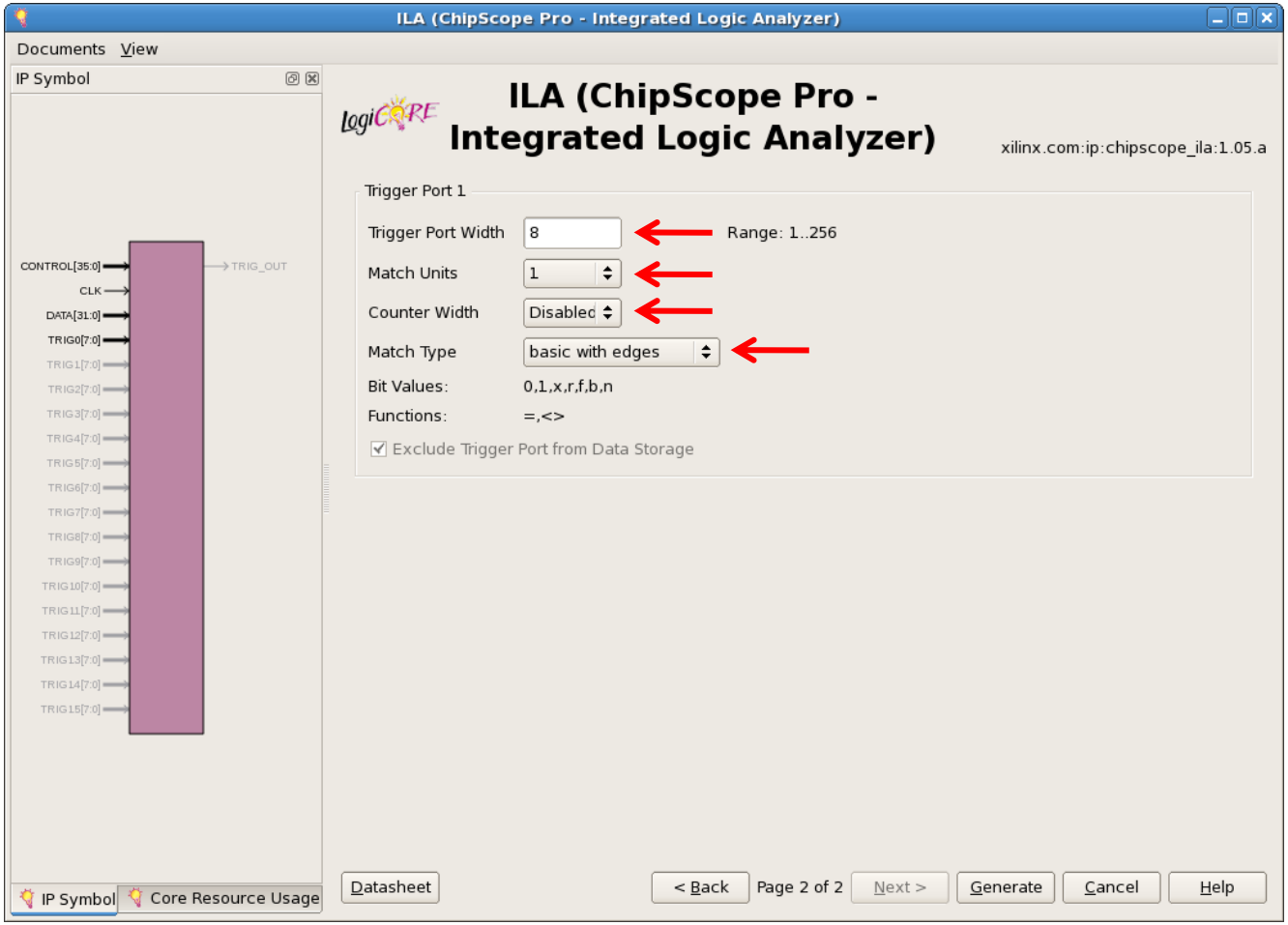


Like before, a summary will appear listing pertinent information about the core being generated. Click **Finish** to launch the ILA core generator.



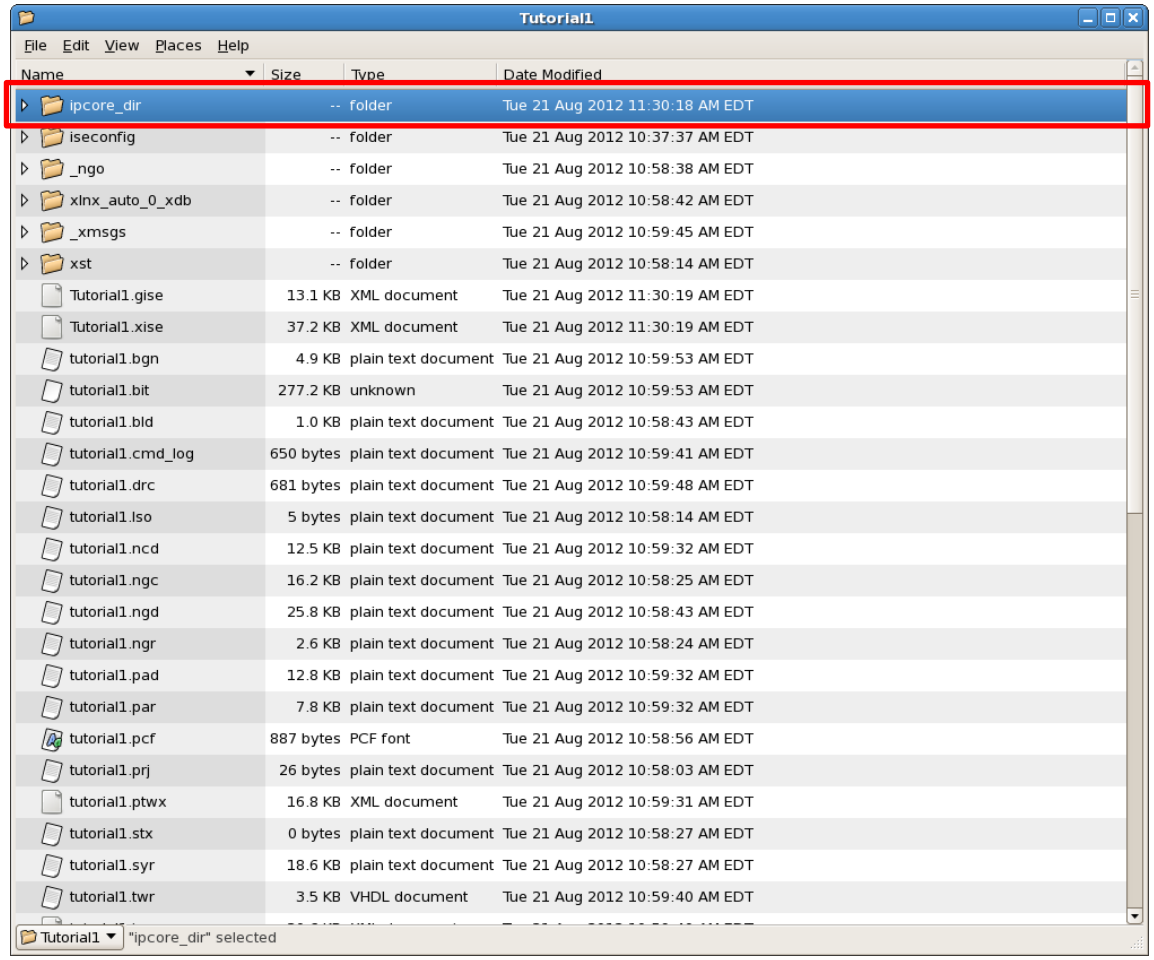
Number of trigger Ports: ***1***
Sample On: ***Rising***
Sample Data Depth: ***512***
Data Same As Trigger: ***Disabled***
Data Port Width: ***32***

The ILA core generator allows you to configure parameters related to the data and trigger port of the core. The majority of the options will be left as they are; the appropriate settings and their parameters are listed above, next to the image. Once the correct settings are selected, click **Next**.

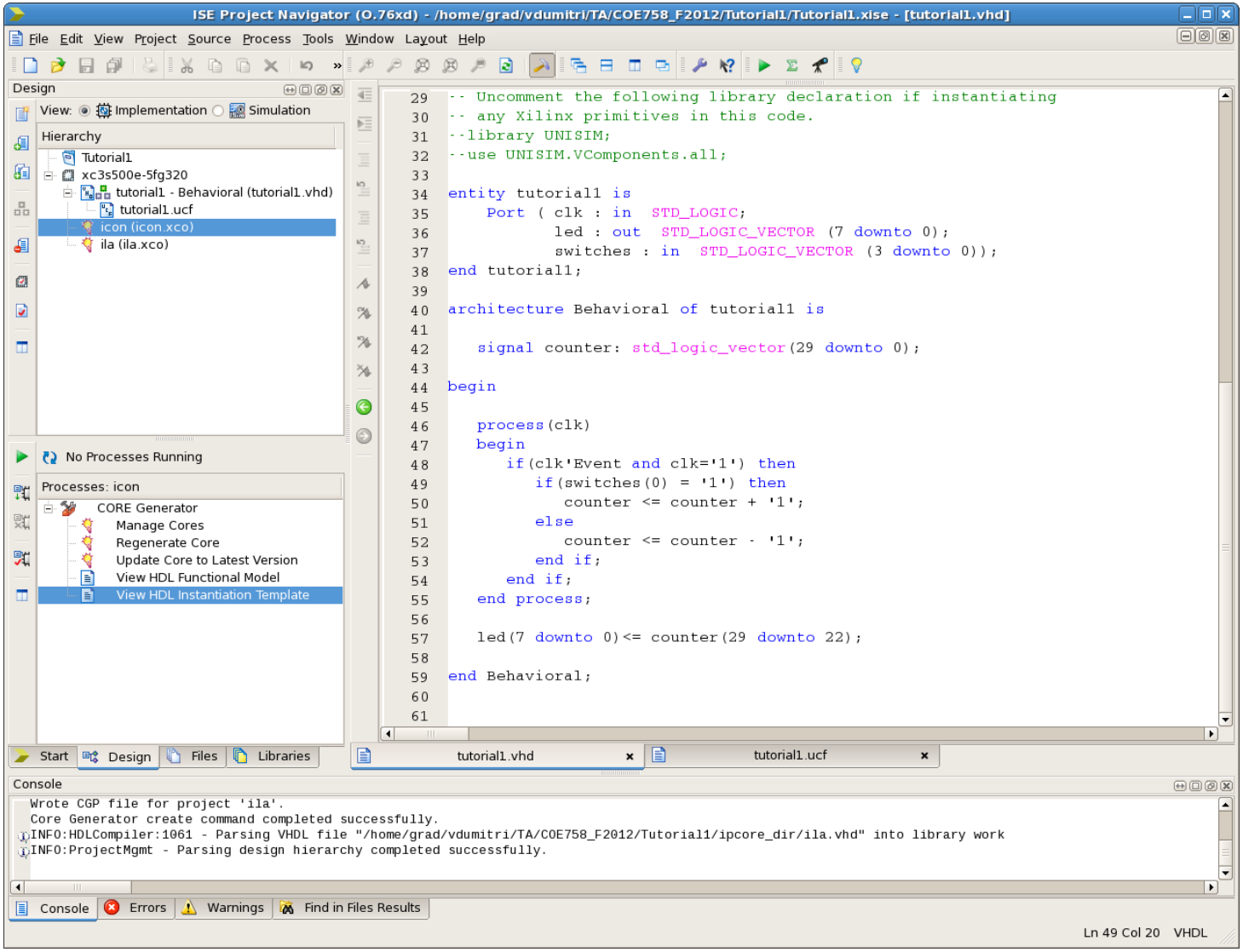


Trigger Port Width: ***8***
Match Units: ***1***
Counter Width: ***Disabled***
Math Type: ***basic with edges***

The second page of the ILA core generator allows you set parameter for the trigger port. Set the parameters as described above, and then click **Generate** to create the core.

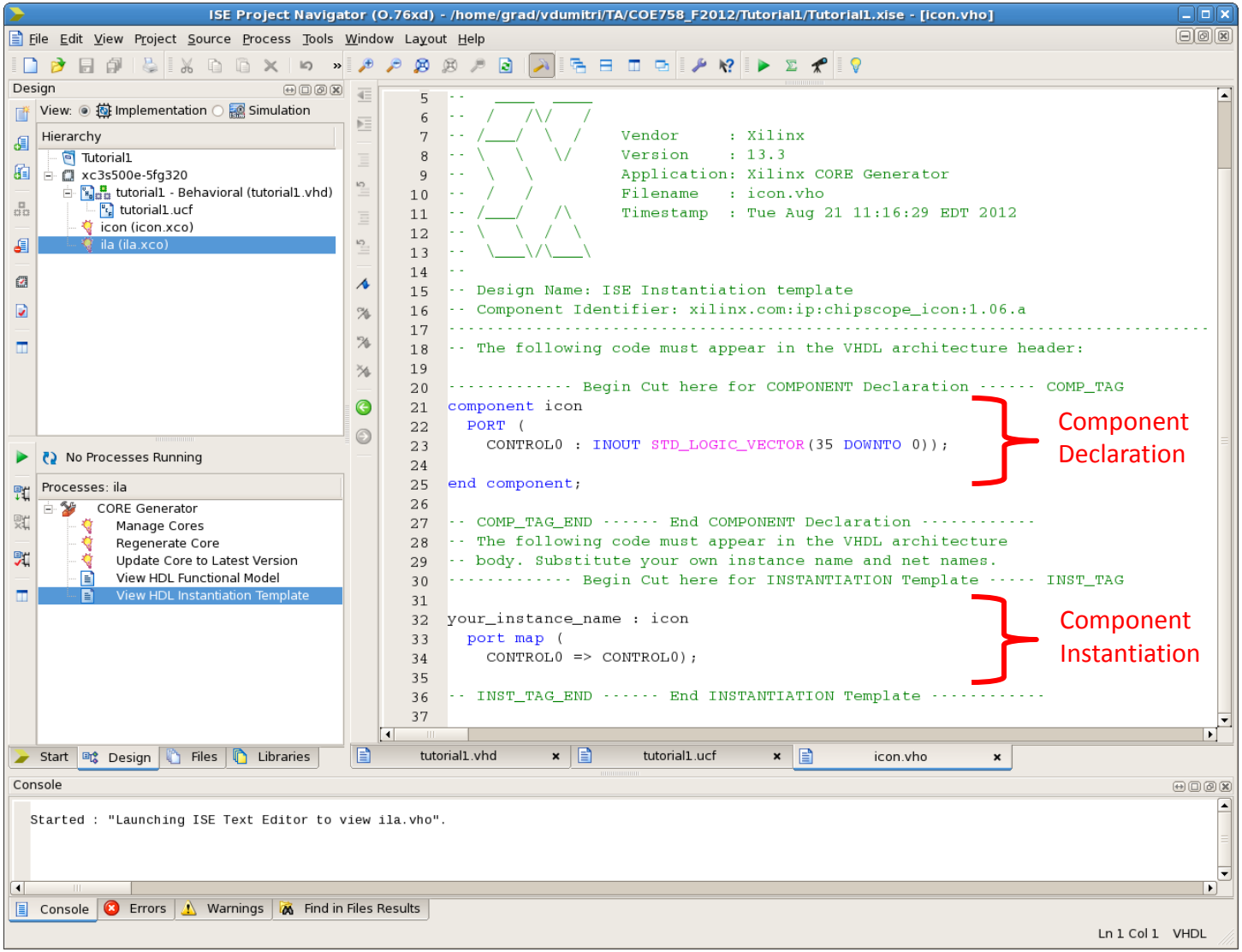


For future reference, note that all cores created through the Core Generator are placed inside a folder in your project directory, named ***ipcore_dir***. All files associated with a specific core will be found here.

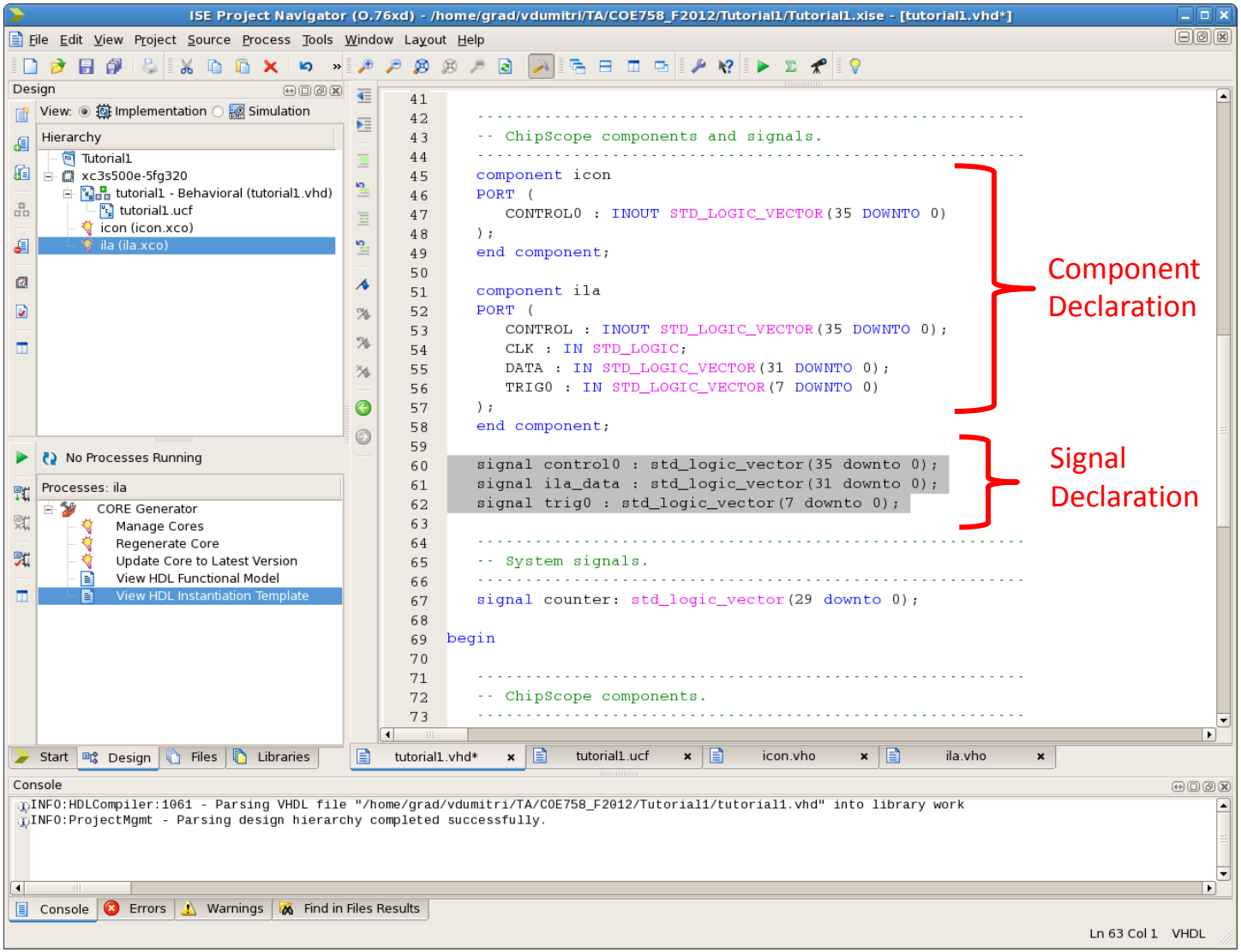



```
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx primitives in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity tutorial1 is
35      Port ( clk : in  STD_LOGIC;
36             led : out  STD_LOGIC_VECTOR (7 downto 0);
37             switches : in  STD_LOGIC_VECTOR (3 downto 0));
38  end tutorial1;
39
40  architecture Behavioral of tutorial1 is
41
42     signal counter: std_logic_vector(29 downto 0);
43
44  begin
45
46     process(clk)
47     begin
48        if(clk'Event and clk='1') then
49           if(switches(0) = '1') then
50              counter <= counter + '1';
51           else
52              counter <= counter - '1';
53           end if;
54        end if;
55     end process;
56
57     led(7 downto 0)<= counter(29 downto 22);
58
59  end Behavioral;
60
61
```

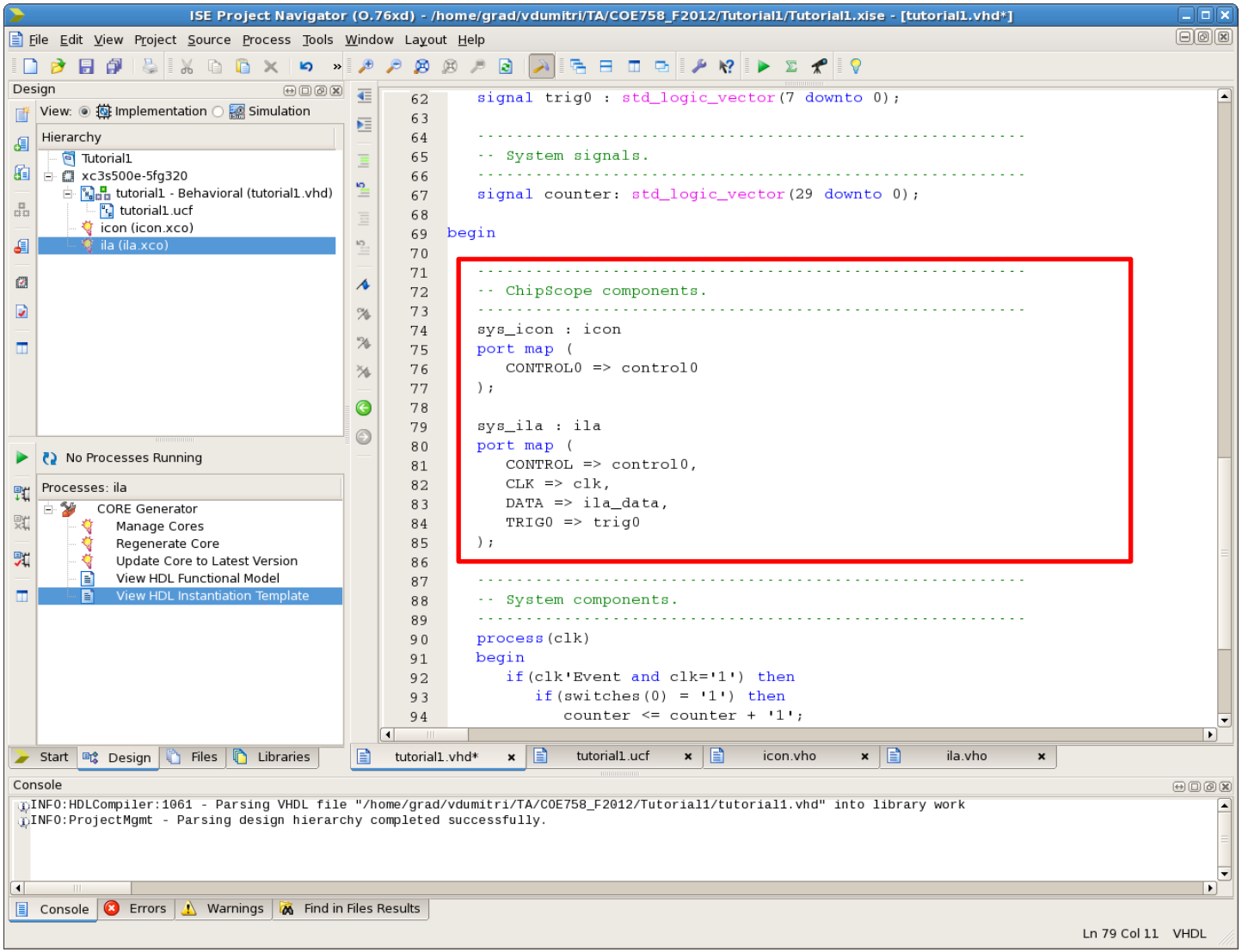
Once the two ChipScope cores have been generated, they must be instantiated into the project. The Core Generator helps with this process by creating a template for each IP core it creates. Open this template for the ICON core by selecting the core in your design navigator, expanding the **CORE Generator** item in the activities pane, and double-clicking on the **View HDL Instantiation Template** item.



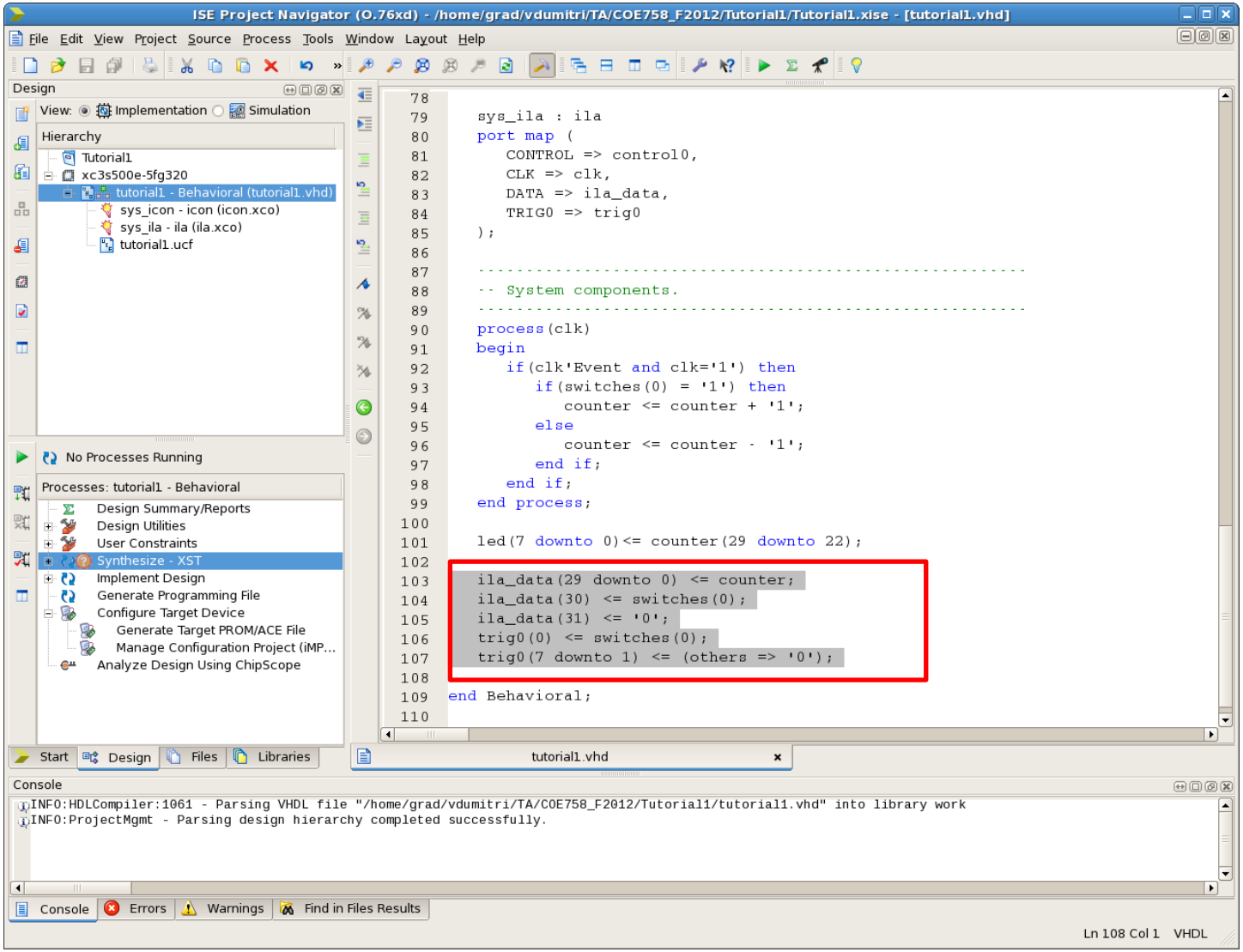
The HDL Instantiation Template contains two core items: the components declaration, and the component's actual instantiation statement. You can now copy these two pieces of code and past them into your tutorial, as discussed in the following steps.



The component declarations for both your ICON and ILA cores should be copied into the declaration portion of your top-level tutorial design. The ICON is connected to the ILA via a dedicated control bus which is 36 bits wide. The ILA core will require buses for its data and trigger ports. These busses should also be declared at this time, as shown in the figure above.



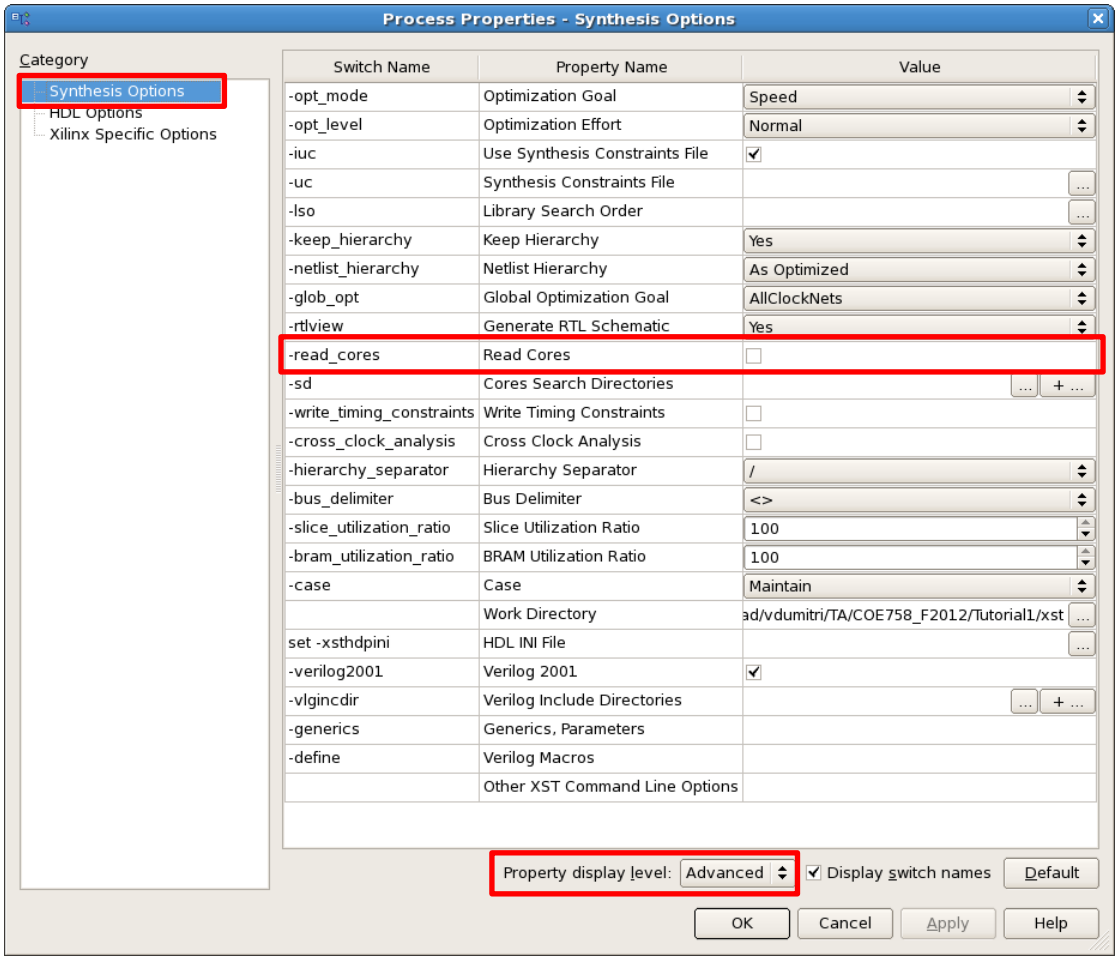
Once the two ChipScope cores have been declared, they can be instantiated as shown in the figure above.



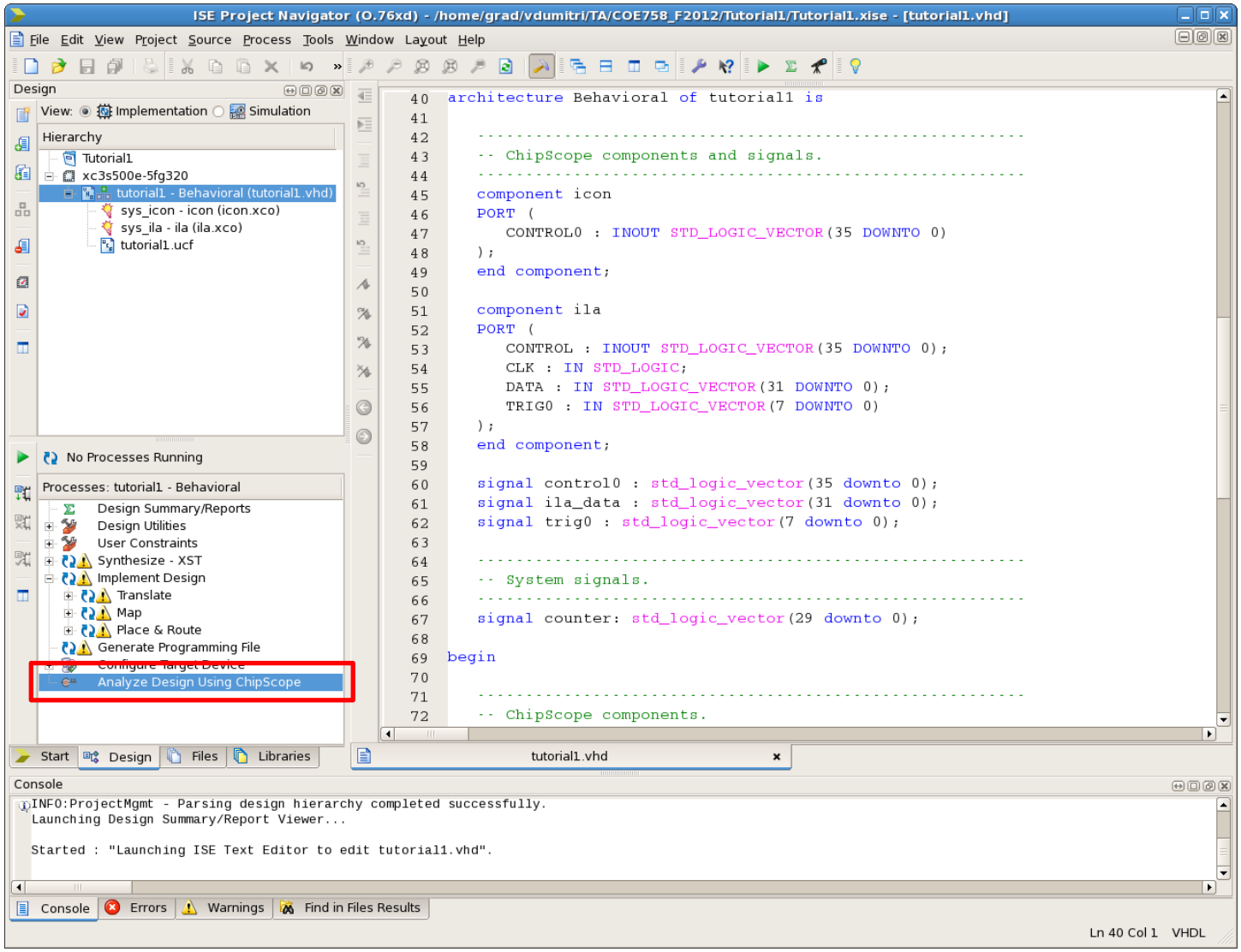
Finally, internal signals of interest can be connected to the ILA data and trigger ports. The data port is used to monitor signals of interest. The trigger port is used to trigger data capture in the ILA based on specific conditions. For the current tutorial, connect the counter signal to the data port, and the switch inputs to the data and trigger port, as shown.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity tutorial1 is
   Port ( clk : in  STD_LOGIC;
          led : out  STD_LOGIC_VECTOR (7 downto 0);
          switches : in  STD_LOGIC_VECTOR (3 downto 0));
end tutorial1;

architecture Behavioral of tutorial1 is

	------------------------------------------------------------
	-- ChipScope components and signals.
	------------------------------------------------------------
	component icon
	PORT (
		CONTROL0 : INOUT STD_LOGIC_VECTOR(35 DOWNTO 0)
	);
	end component;

	component ila
	PORT (
		CONTROL : INOUT STD_LOGIC_VECTOR(35 DOWNTO 0);
		CLK : IN STD_LOGIC;
		DATA : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
		TRIG0 : IN STD_LOGIC_VECTOR(7 DOWNTO 0)
	);
	end component;

	signal control0 : std_logic_vector(35 downto 0);
	signal ila_data : std_logic_vector(31 downto 0);
	signal trig0 : std_logic_vector(7 downto 0);

	------------------------------------------------------------
	-- System signals.
	------------------------------------------------------------
	signal counter: std_logic_vector(29 downto 0);

begin

	------------------------------------------------------------
	-- ChipScope components.
	------------------------------------------------------------
	sys_icon : icon
	port map (
		CONTROL0 => control0
	);

	sys_ila : ila
	port map (
		CONTROL => control0,
		CLK => clk,
		DATA => ila_data,
		TRIG0 => trig0
	);

	------------------------------------------------------------
	-- System components.
	------------------------------------------------------------
	process(clk)
	begin
		if(clk'Event and clk='1') then
			if(switches(0) = '1') then
				counter <= counter + '1';
			else
				counter <= counter - '1';
			end if;
		end if;
	end process;

	led(7 downto 0)<= counter(29 downto 22);

	ila_data(29 downto 0) <= counter;
	ila_data(30) <= switches(0);
	ila_data(31) <= '0';
	trig0(0) <= switches(0);
	trig0(7 downto 1) <= (others => '0');

end Behavioral;
```

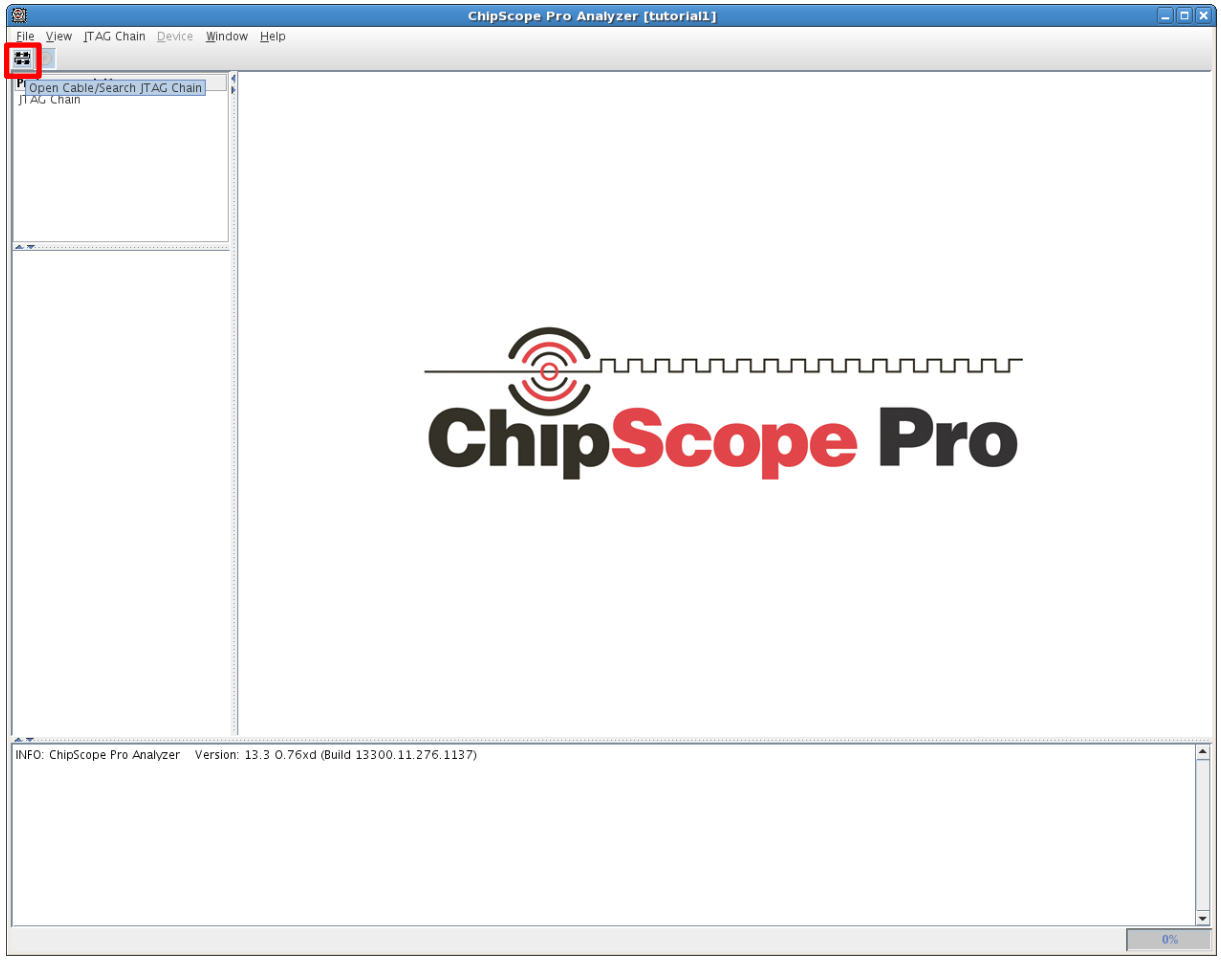
The complete code for Tutorial 2 is listed above. Please note that the Implementation Constraints File from Tutorial 1 is also used, but is not listed here.



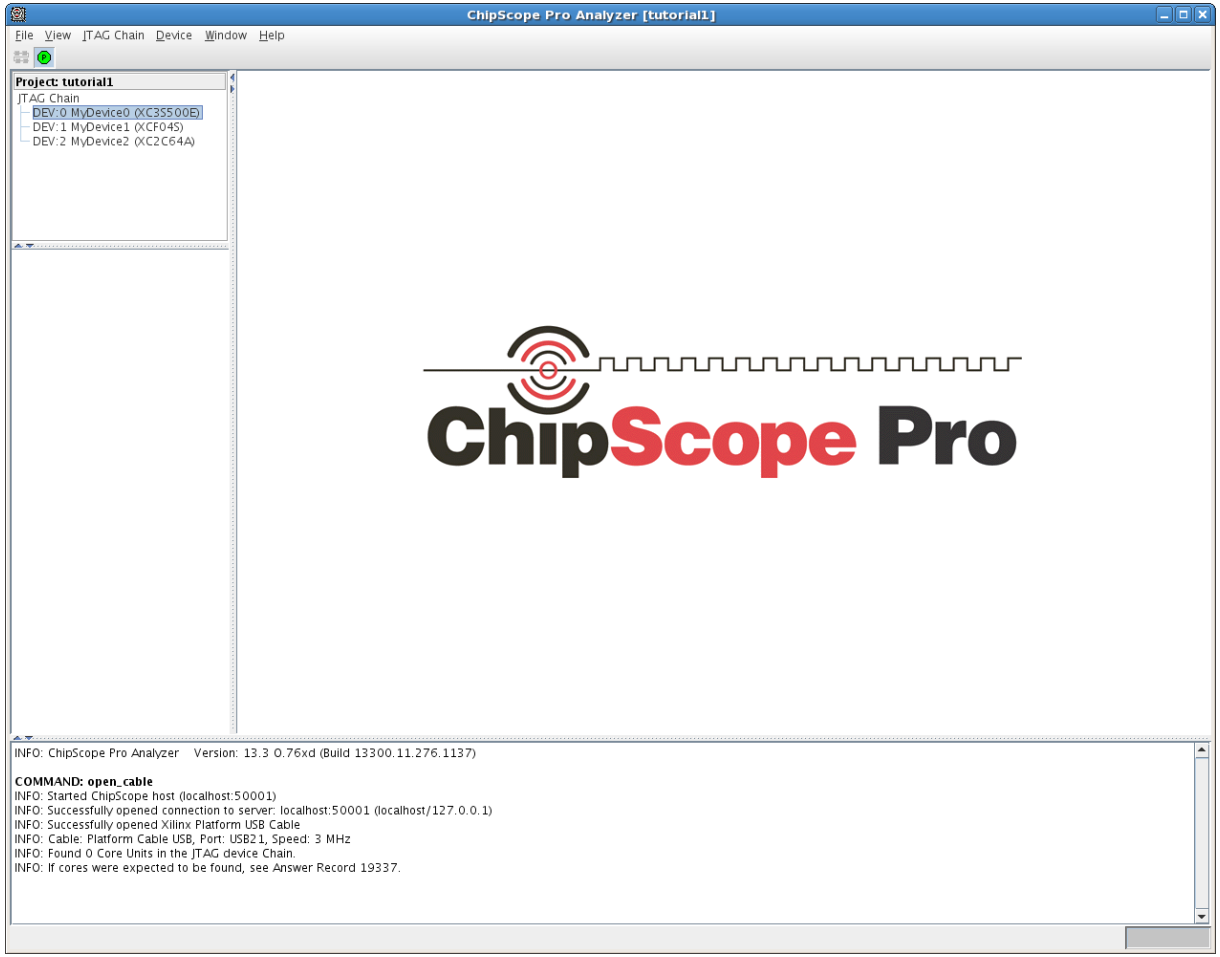
To ensure that the project is implemented correctly, a specific synthesis option must be set. To change synthesis options, right click on the **Synthesize - XST** item in the activities pane, and select **Process Properties...** . This will bring up and options window. First, set the **Property Display Level** setting to **Advanced**, as shown above. Ensure that the **Synthesis Options** category is selected, and disable the **Read Cores** option; finally click **Apply** and then **OK**.



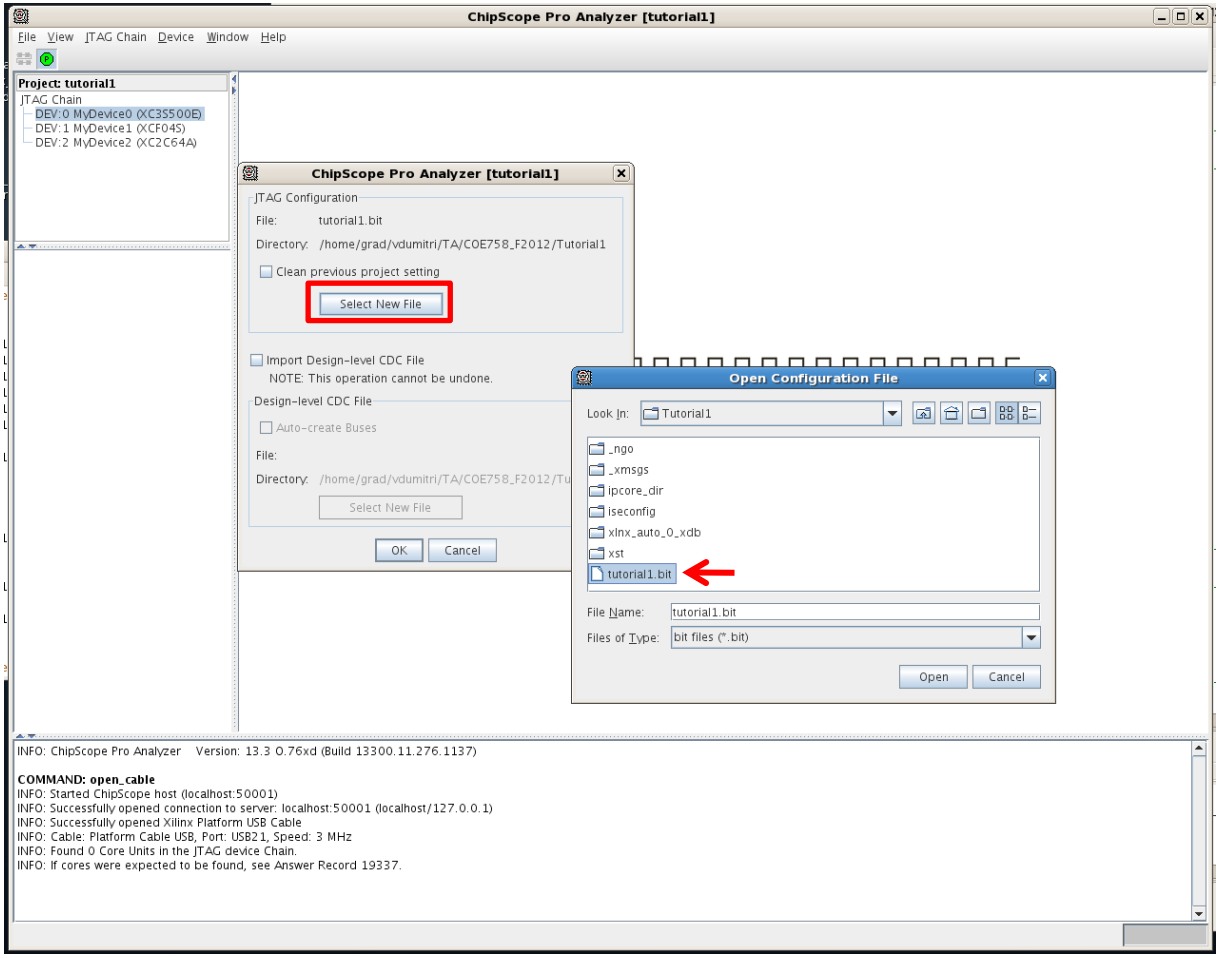
You can now synthesize and implement your updated design, as described in the previous tutorial. Once the programming file has been generated, you are ready to analyze and debug your design. Launch the ChipScope analyzer by double-clicking the **Analyze Design Using ChipScope** item. Note that you do not need to use iMPACT to program the FPGA. The ChipScope analyzer also acts as a programmer and will allow you to program the device.



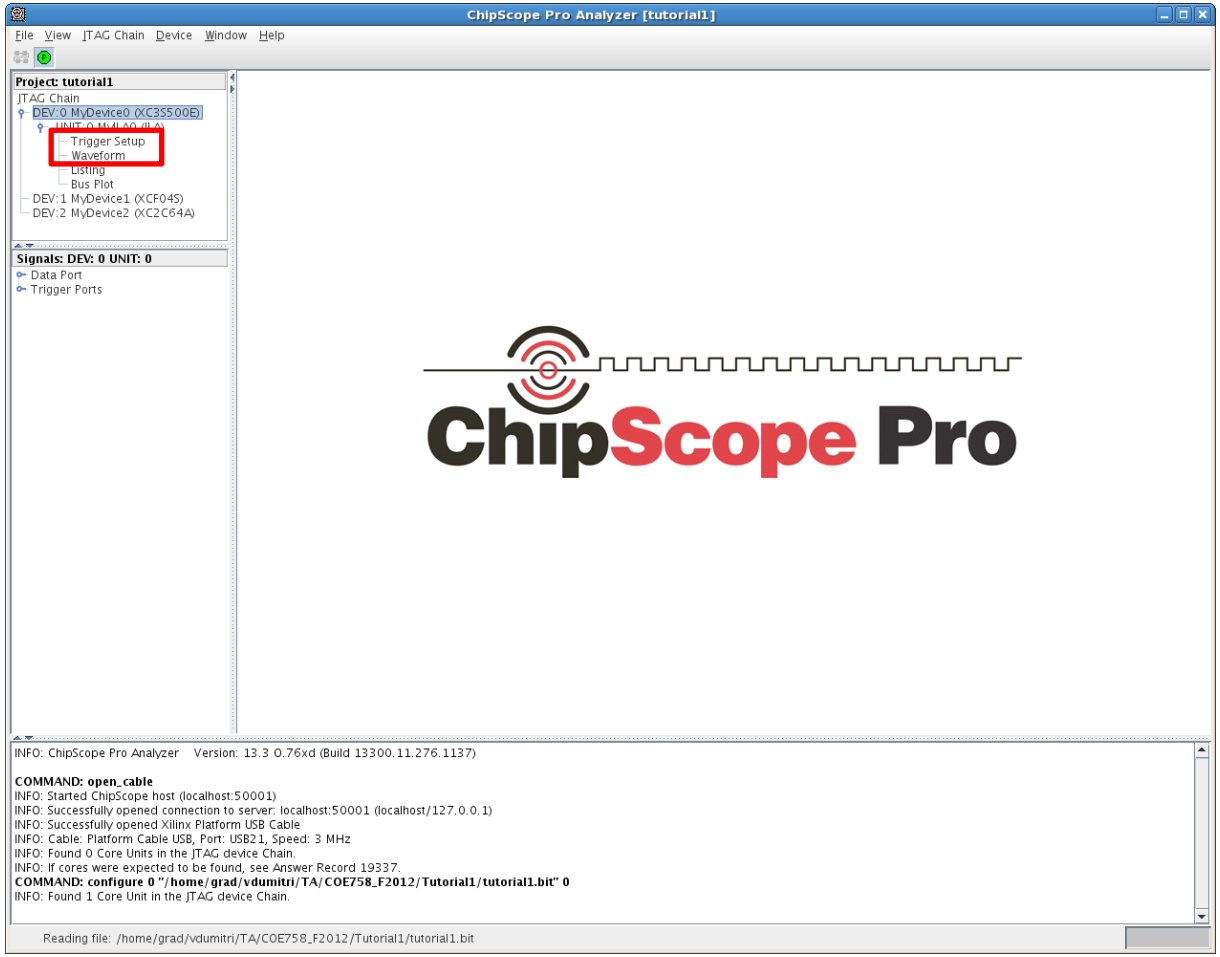
Once the ChpScope Pro Analyzer is running, the JTAG Chain must be initialized. Click the button highlighted above to initialize the chain. Once that step completes, a new window will appear with a list of all found devices, and their ID code. Click **OK** to close this window.



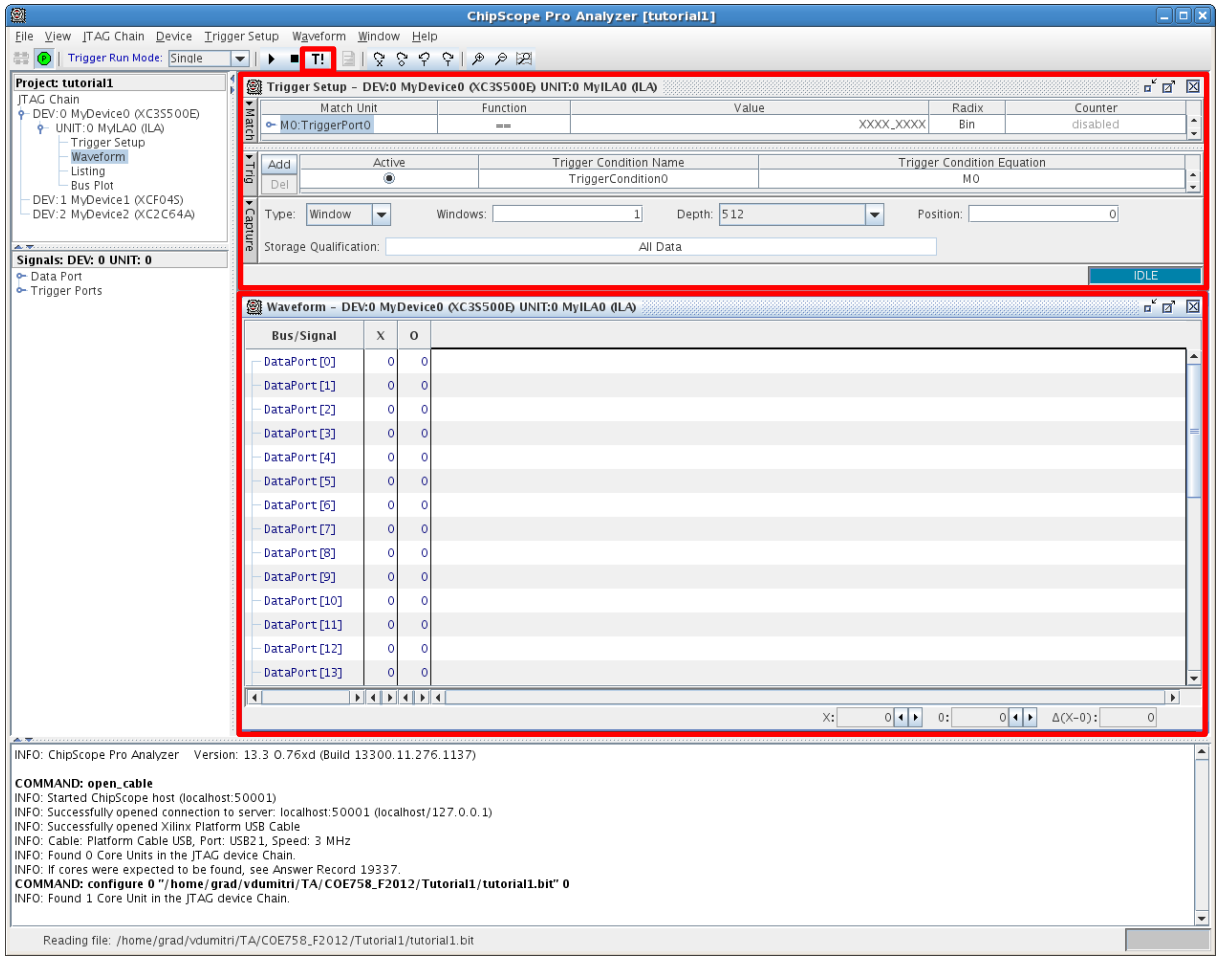
Once the chain is initialized, you are ready to configure the on-board FPGA. Right-click on **Dev: 0 My Device 0 (XC3S500E)** and select **Configure**.



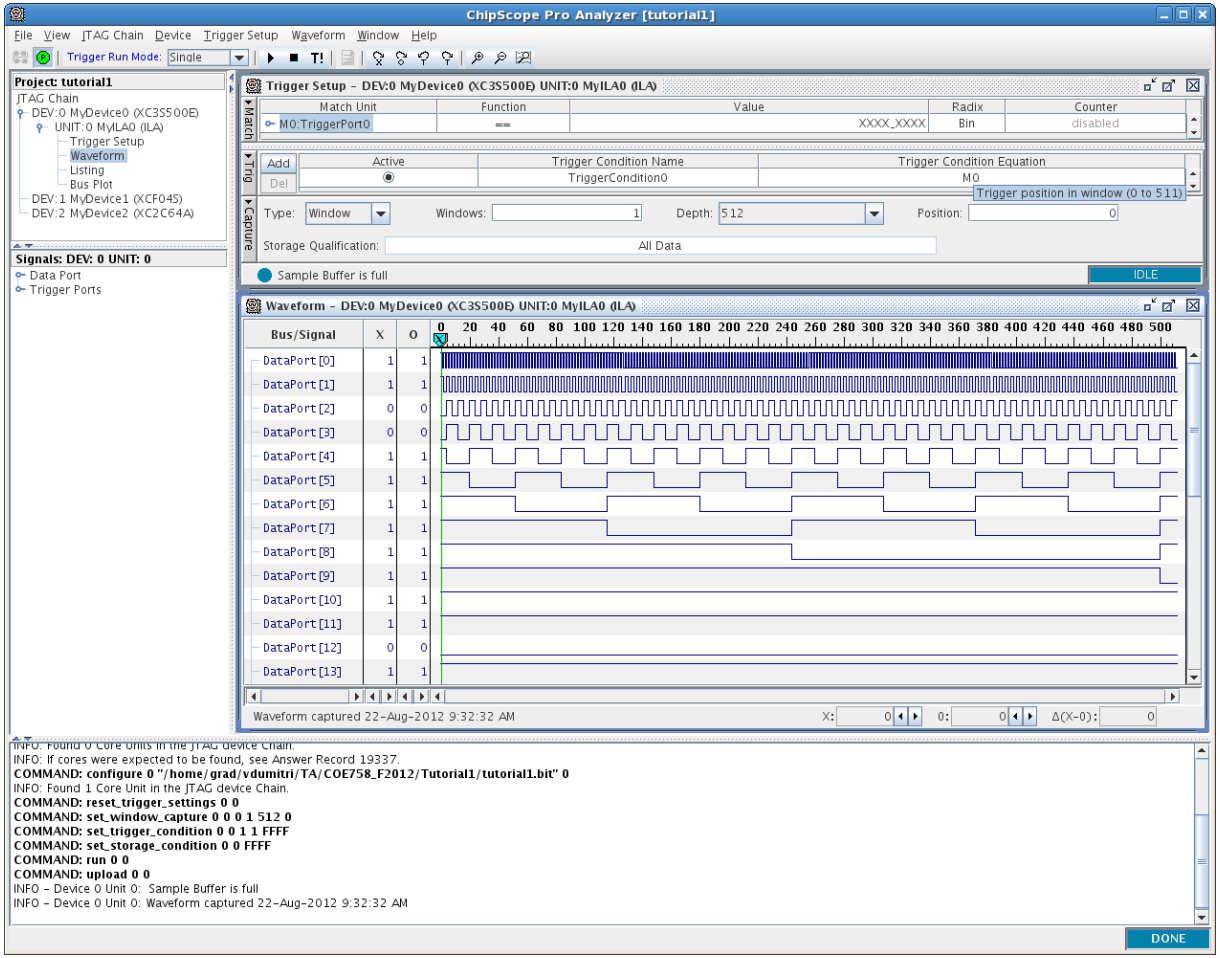
A new prompt will appear, specifying which configuration file should be used to configure the device. Ensure that the correct bit-stream file is selected. If no file (or an incorrect file) is selected, click the **Select New File** button, and select the appropriate configuration file for this tutorial. Once the correct file is selected, click **OK**.



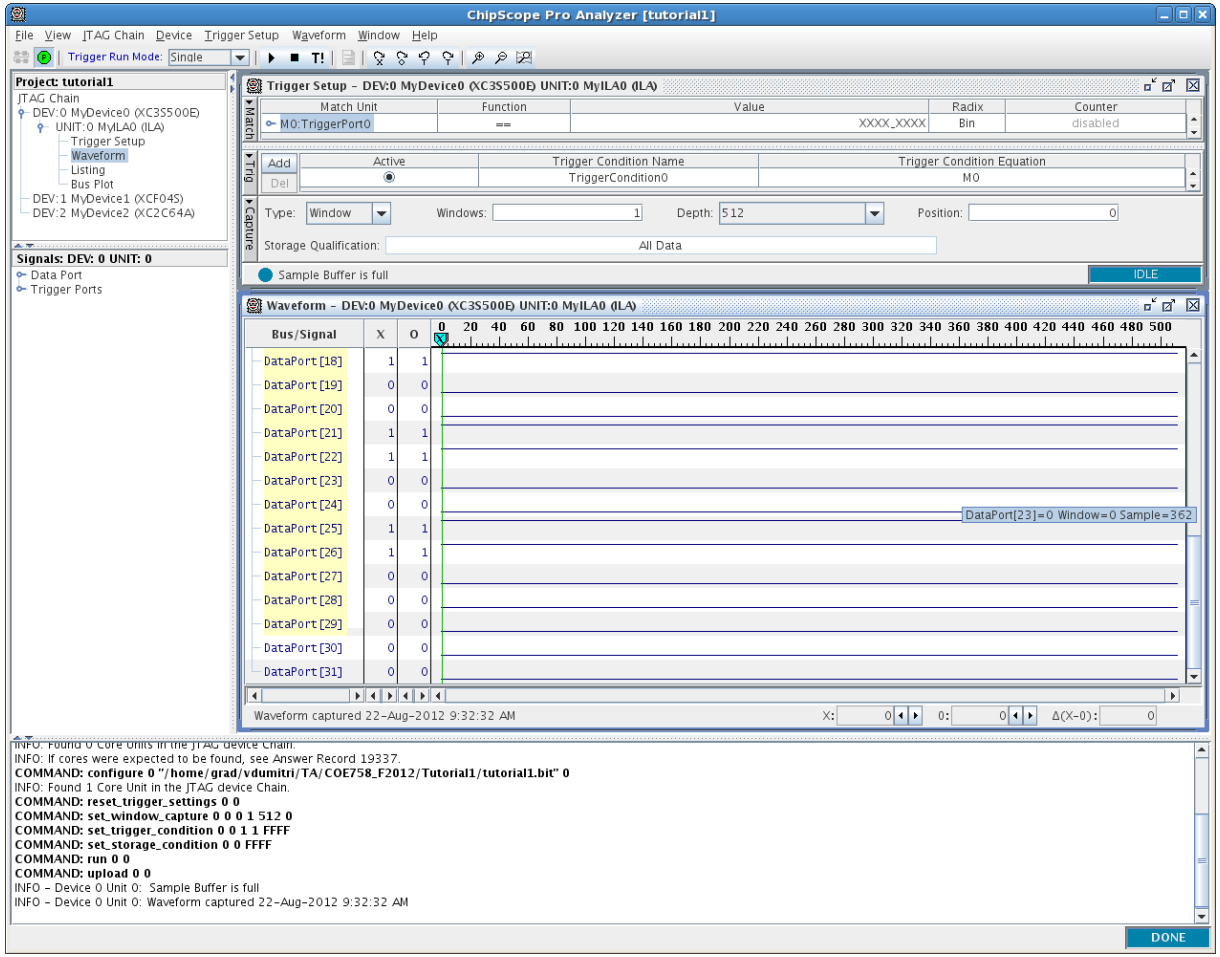
Once the device configuration is complete, **Dev: 0 My Device 0 (XC3S500E)** will expand, allowing you to access various utilities associated with the ILA. In the current tutorial we are interested in the **Trigger Setup** and **Waveform** items. Double-click each to bring up their respective windows.



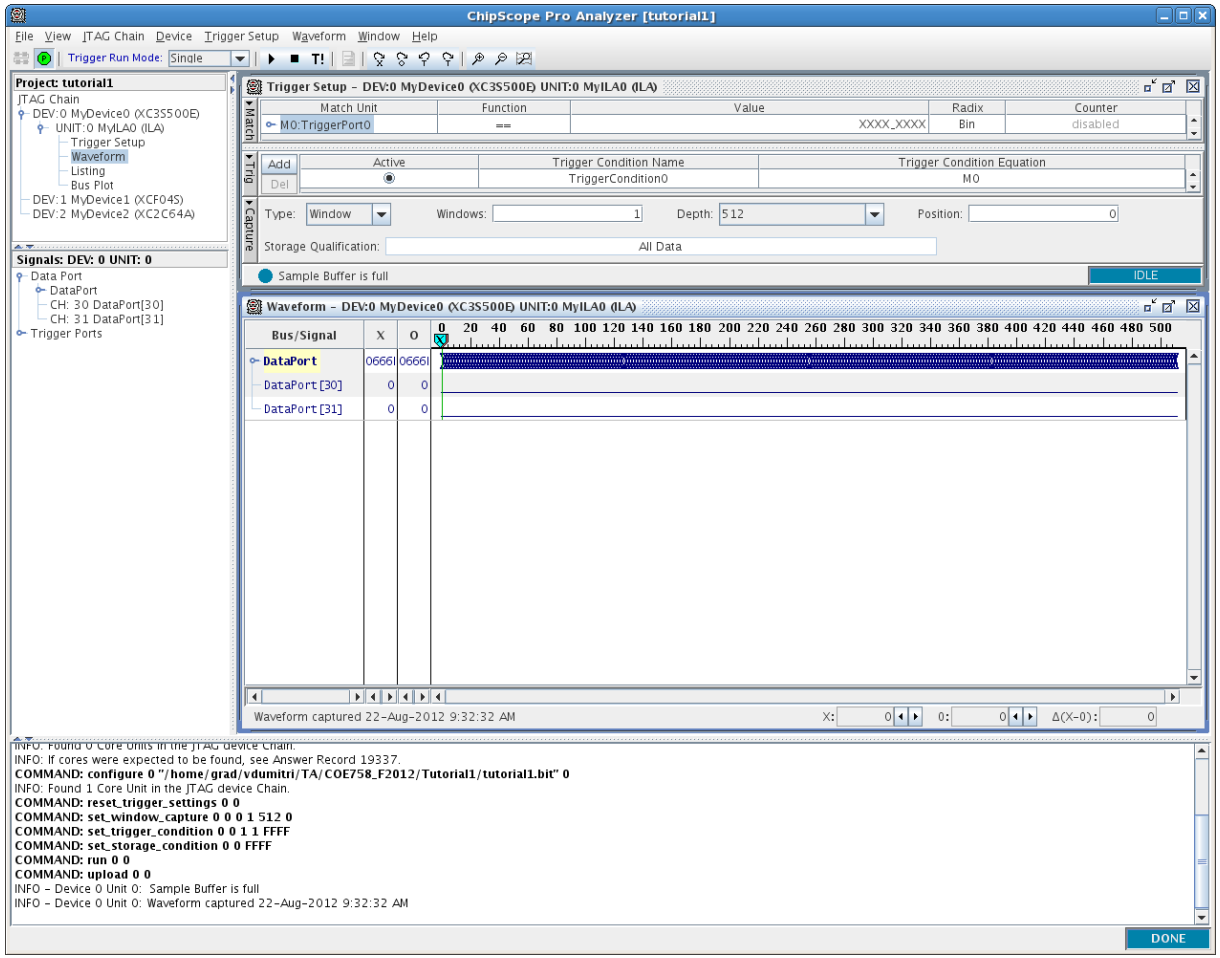
The figure above shows the **Waveform** and **Trigger Setup** windows. The waveform window is used to display all captured results in the form of timing diagrams. The trigger setup window is used to configure trigger conditions for the ILA core. At any time, the ILA core can be triggered to capture data from the design by clicking on the T! Button.



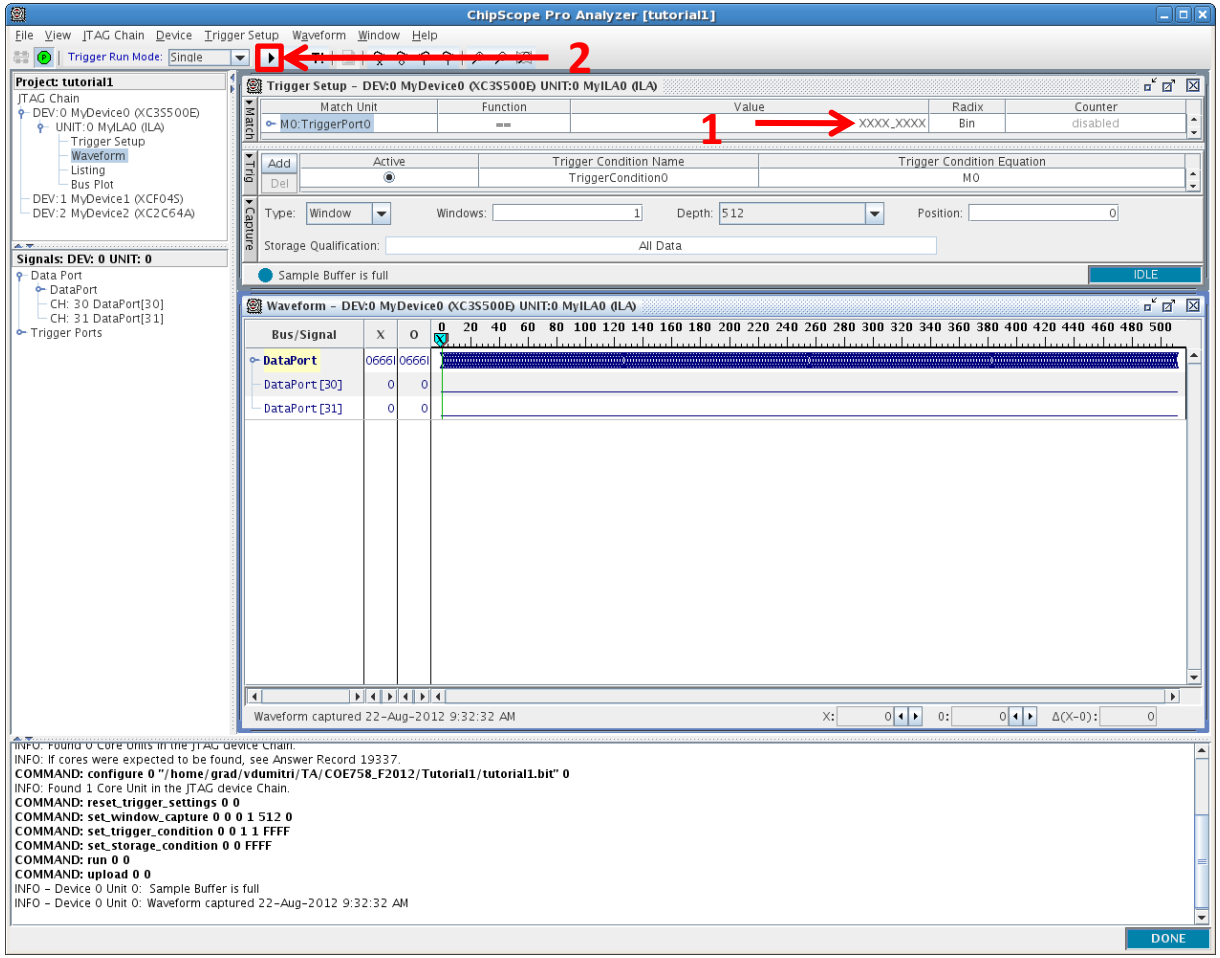
The above figure shows an example of data captured from the counter present in the design.



To facilitate analysis, individual signals in the waveform window can be grouped into buses, to reflect their organization inside your design. As an example, signal DataPort[29] down to DataPort[0] form a single bus, connected to the system counter. To group these signals together select all of them, right-click them, select the option **Move to Bus ➔ New Bus**.



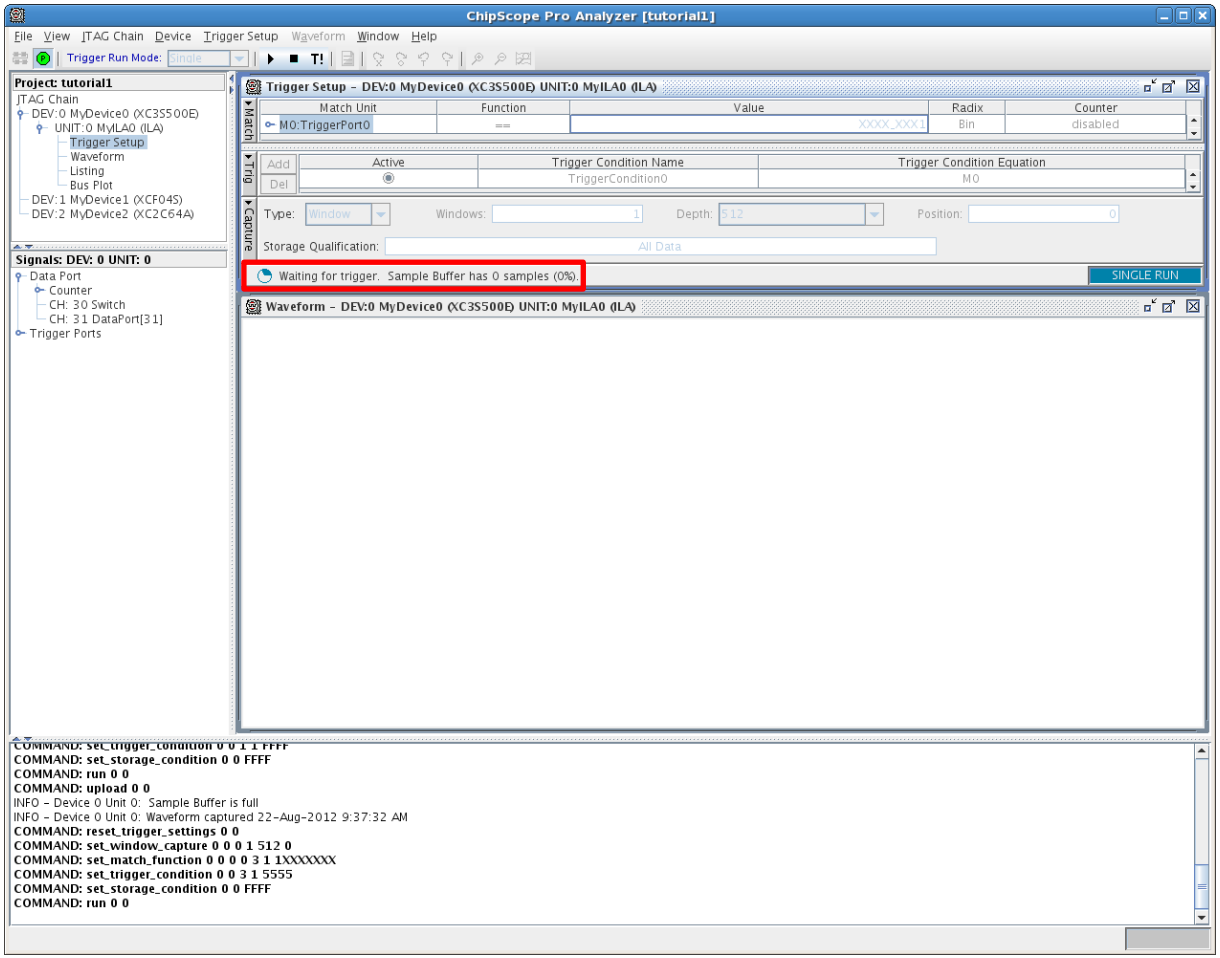
Once created, the new bus can be renamed to something more suitable by right-clicking it and selecting the option **Rename**. The way bus data is represented in the waveform window can be changed to a more convenient format by right-clicking the bus, selecting the **Bus Radix** option, and selecting a different bus radix (such as binary, unsigned decimal or hexadecimal).



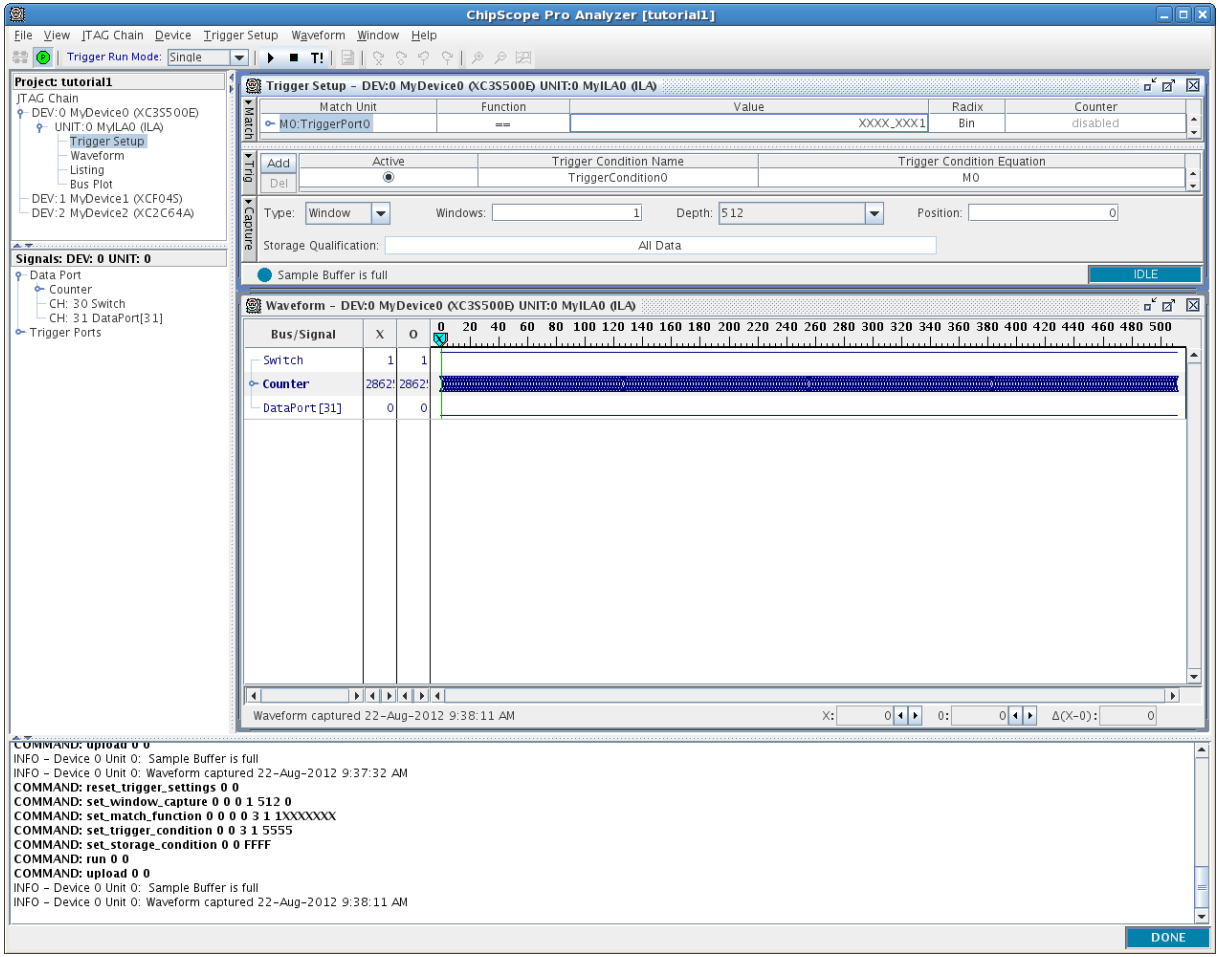
To arm the ILA so that it captures data only when signal ***switches[0]*** is 1, do the following:

1. Replace the least-significant X in the trigger setup with 1.
2. Press the Play (arm core) button.

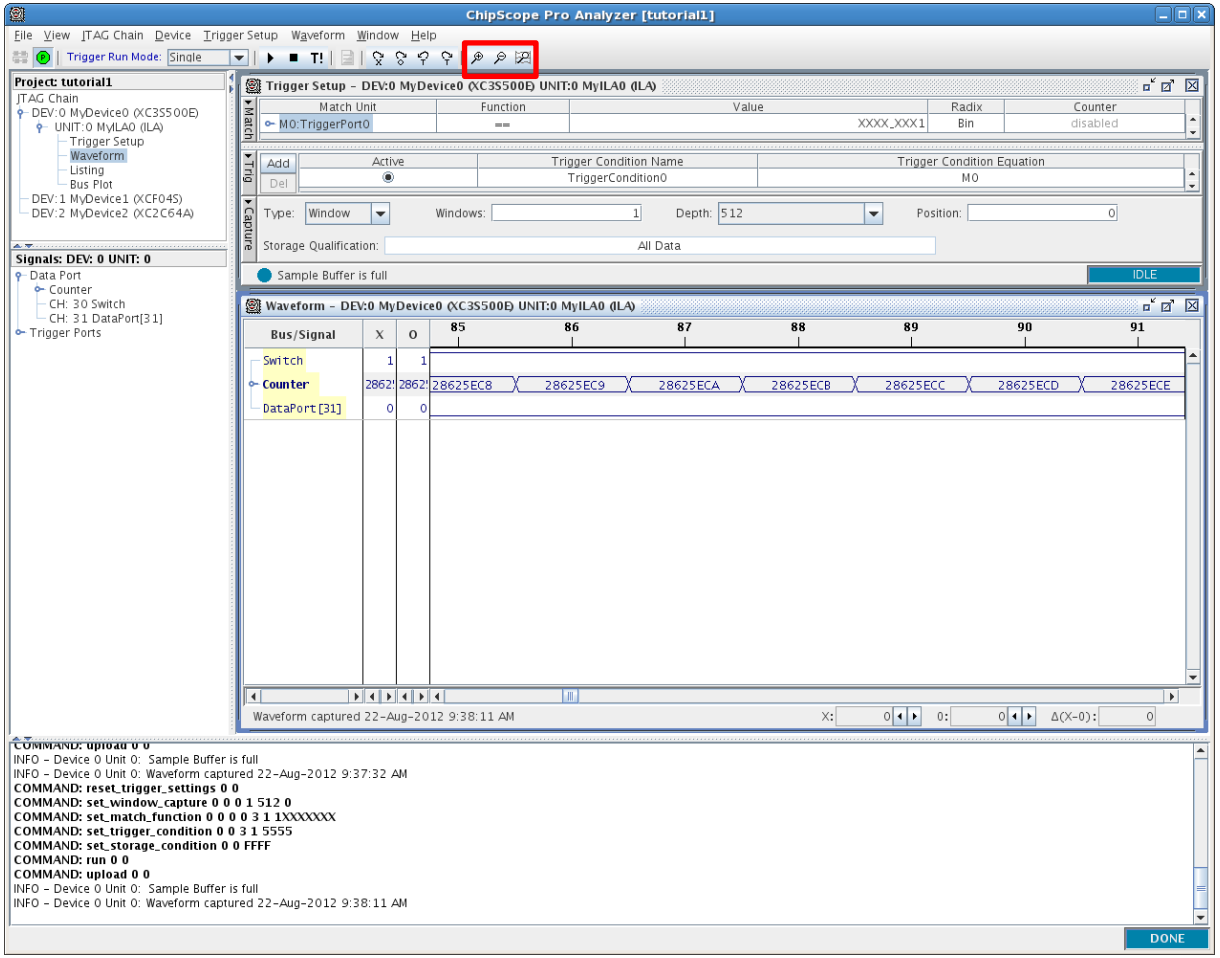
It is often beneficial to capture data only when certain conditions are met. Specifically, the ILA core can capture data when signals connected to the trigger port have certain values, or when certain edges are detected. For the current tutorial, we will configure the ILA so that it captures data once signal trig0[0] (which is connected to port ***switches[0]*** in your design) has a value of 1. This is accomplished by editing the trigger setup so that trigger port 0 is set to 1. Once the trigger condition is set, the core can be armed by pressing the “play” button, as highlighted above.



Once armed, the core will wait for the trigger condition before capturing data and sending it to the host computer. The image above shows the waiting for trigger message, which indicates that the trigger condition has not occurred yet. To trigger the core, push Switch 0 on the laboratory development board from the 0 position to the 1 position.



Once the capture process is triggered and completes, the new data is visible in the waveform window.



To be able to see individual values of the counter, the waveform view can be zoomed in. To zoom in, either right-click inside the waveform window and select **Zoom ➔ Zoom In**, or use the zoom buttons in the top tool-bar (highlighted above).

Conclusion

- This completes Tutorial 2. This tutorial has covered the following topics:
  - An overview of the ChipScope Pro Integrated Logic Analyzer.
  - Generation of ChipScope IP cores.
  - Insertion of ChipScope IP cores into the design created in Tutorial 1.
  - Use of the ChipScope Logic Analyzer to configure an FPGA and capture signal values of interest.
- You may now begin work on Tutorial 3, where you will learn how to use the ChipScope Virtual I/O core (VIO) to inject I/O signals directly into your design. You will also learn how to generate, instantiate and interact with local memory inside the FPGA, in the form of a single-ported SRAM unit.