

ISE Tutorial:

Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Applications

UG750 (v 14.5) Month March 20, 2013





Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

©Copyright 2012-2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/18/2012	13.4	Revalidated for the 13.4 release. Editorial updates only; no technical content updates.
04/24/2012	14.1	Revalidated for the 14.1 release. Editorial updates only; no technical content updates.
07/25/2012	14.2	Revalidated for the 14.2 release. Editorial updates only; no technical content updates.
10/16/2012	14.3	Revalidated for the 14.3 release. Updated Figure 7 for CR 657985. No other technical content updates.
12/18/2012	14.5	Revalidated for the 14.4 release. Updated graphics to reflect latest release of software.

Table of Contents

Revision History.....	2
Introduction.....	4
Objectives.....	4
Lab Exercise Setup.....	4
Design Description.....	5
Push Button Switches.....	5
Debounce Circuit.....	6
Control State Machine.....	7
LED Displays.....	7
Files Provided for this Lab Exercise.....	7
Procedure.....	8
Creating and Implementing Your RTL Design.....	9
Adding a ChipScope ILA Core.....	21
Questions.....	30
Debugging Your Design.....	31
Questions.....	41
Conclusion.....	42
Question Answers.....	42

Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Applications

Introduction

In this lab exercise, you will explore how an Integrated Logic Analyzer (ILA) core can be inserted within the Project Navigator design environment to debug your FPGA designs. You will take advantage of ChipScope™ Pro Analyzer functions to debug and discover some potential root causes of your design, thereby allowing you to address issues quickly as will be shown by this tutorial.

Example RTL designs will be used to illustrate overall integration flows between ChipScope and Xilinx® ISE® Project Navigator. In order to be successful using this tutorial, you should have some basic knowledge of ISE tool flows.

Objectives

After completing this lab exercise, you will be able to:

- Validate and debug your design using Project Navigator and ChipScope with ILA core and Analyzer
- Understand how to create an ISE project, probe an ILA core, and implement the design in Project Navigator
- Debug the design using ChipScope Analyzer and iterate the design using Project Navigator design environment and an SP601Platform

Lab Exercise Setup

The following software and hardware are required for this lab:

- Xilinx ISE Design Suite 14.5 (Logic, DSP, Embedded, or System Edition)
- SP601Platform
- JTAG Cables that come with the SP601 Platform

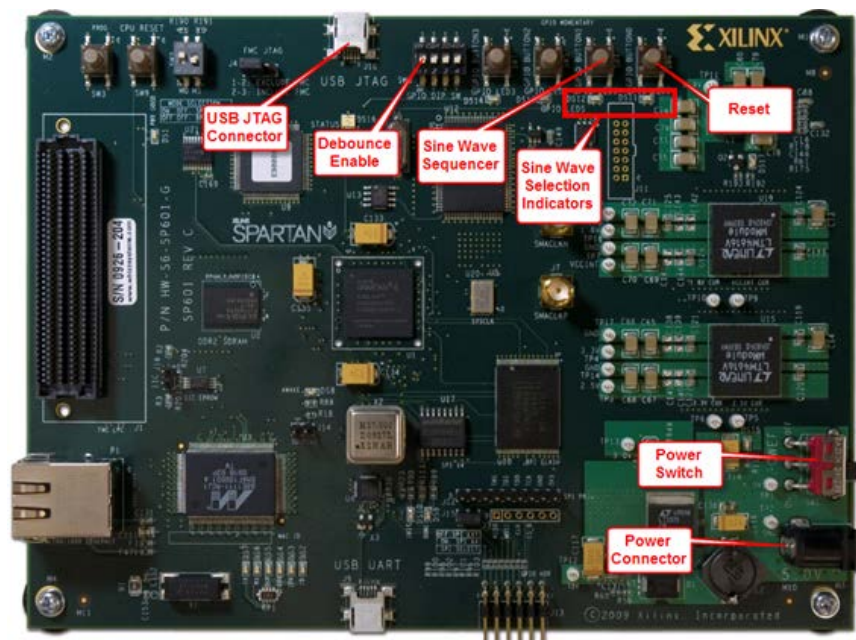


Figure 1: SP601-Platform

Design Description

The top-level block diagram for the RTL example design is shown in Figure 2. The design is comprised of a simple control state machine, multiple sine wave generators, common Push Button (GPIO_BUTTON), Dip Switch (GPIO_SWITCH), and LED displays (GPIO_LED).

Push Button Switches

Push Button Switches are used as inputs into the debounce and / or control state machine circuits. A high to low transition pulse is generated when a switch is pushed. Each generated output pulse is then used as an input into the state machine.

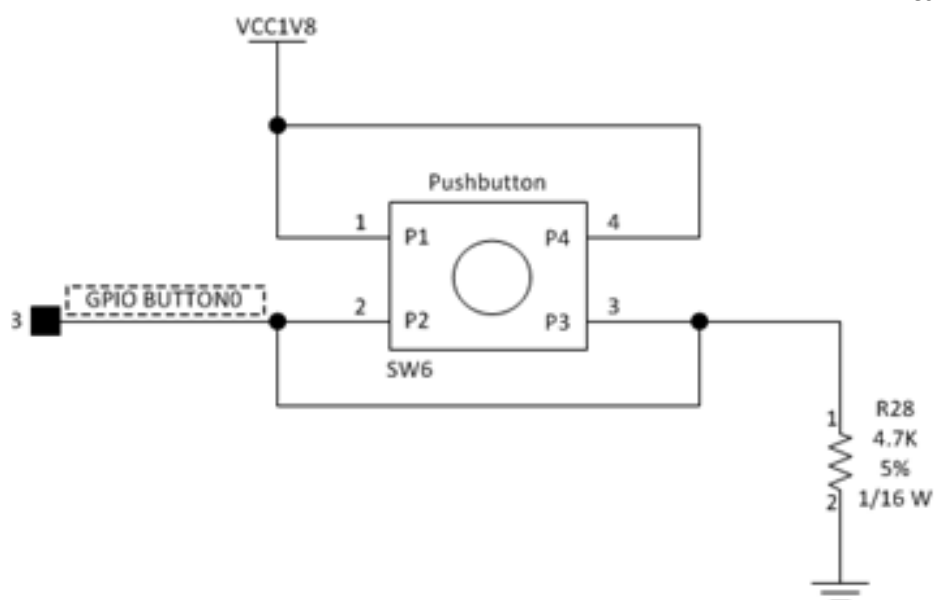


Figure 2: Push Button Used to enable or disable a debounce circuit

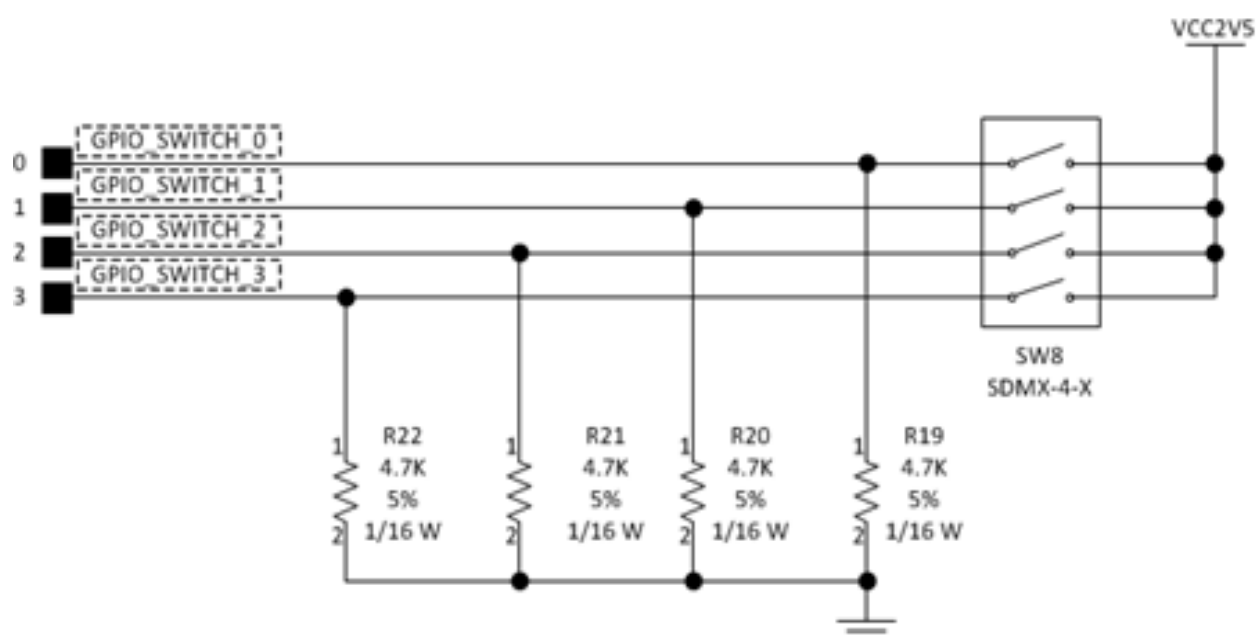


Figure 3: Debounce circuit

Debounce Circuit

When enabled, the debounce circuit provides a clean pulse or transition from a high to low on this particular example. It eliminates series of spikes or glitches when a button is pressed and released.

Control State Machine

The control state machine is used to capture and decode input pulses from the two Push Button switches.. It provides sine wave selection and indicator circuits, sequencing between 00, 01, 10, and 11 (zero to three).

LED Displays

GPIO_LED_0 through GPIO_LED_3 are used to display selection status from the state machine outputs, each of which represents a different sine wave frequency - high, medium, and low.

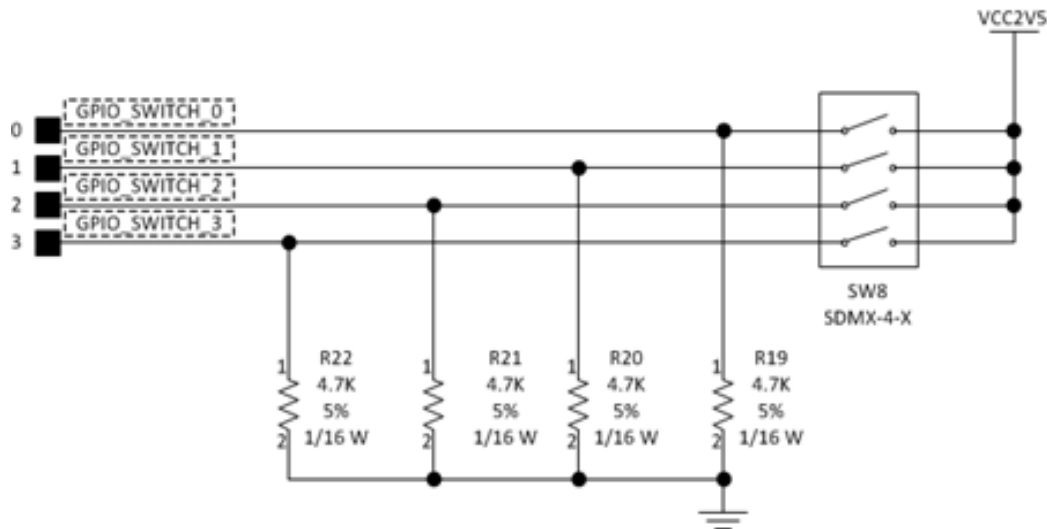


Figure 4: LED Displays

Files Provided for this Lab Exercise

The following files and subfolders are provided for this lab exercise:

- debounce.vhd - Debounce circuit
- fsm.vhd - Control state machine
- sinegen_demo.vhd - Wrapper for Sine wave generators
- sine_high.xco, sine_mid.xco, sine_low.xco - Xilinx Core Generator files
- sinegen_demo_sp601.ucf - UCF constraint file

Note: This lab also supports two other Xilinx platforms - SP605 and ML605. Use pin-out information provided by the table below to retarget this tutorial to either SP605 or ML605.

The design files can be downloaded from the following link:

http://www.xilinx.com/support/documentation/dt_isel4-5_tutorials.htm

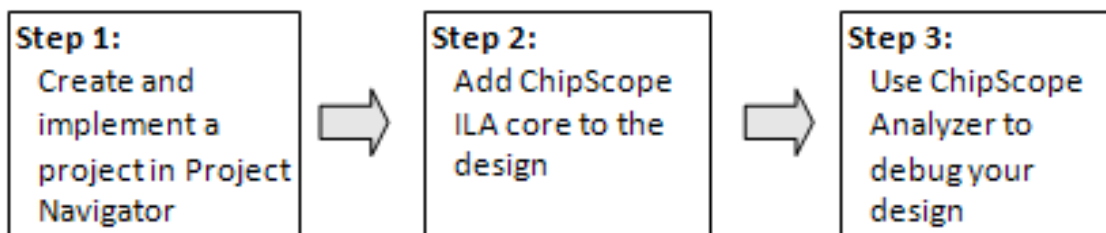
	Pin-Out Locations			Function
	SP601	SP605	ML605	
CLK_N	K16	K22	H9	Clock
CLK_P	K15	K21	J9	Clock
GPIO_BUTTONS0	P4	F3	A19	Reset
GOIP_BUTTONS1	F6	G6	G26	Sequencer
GPIO_SWITCH	D14	C18	D22	Debounce circuit selector
LEDs n[0]	E13	D17	AC22	Selector[0]
LEDs n[1]	C14	AB4	AC24	Selector[1]
LEDs n[2]	C4	D21	AE22	Selector[2]
LEDs n[3]	A4	W15	AE23	Selector[3]

Table 1: Pin-out information for Xilinx Platforms

Procedure

In this tutorial, you will complete three tasks:

1. Creating and Implementing a Project in Project Navigator.
2. Adding a ChipScope ILA Core to Your Design.
3. Debugging Your Design using ChipScope Pro Analyzer.



Chapter 2

Creating and Implementing a Project in Project Navigator

Creating and Implementing Your RTL Design

In this tutorial, you will explore how Project Navigator can be used to quickly implement your RTL design. You will learn how to create an ISE® project in Project Navigator targeting the SP601 Platform.

1. Unzip the provided source files in C:\ChipScope_ProjNav\
2. Start **Project Navigator** and select **File > New Project** to create a new project. Provide the name, location, and project type, and then click **Next**.

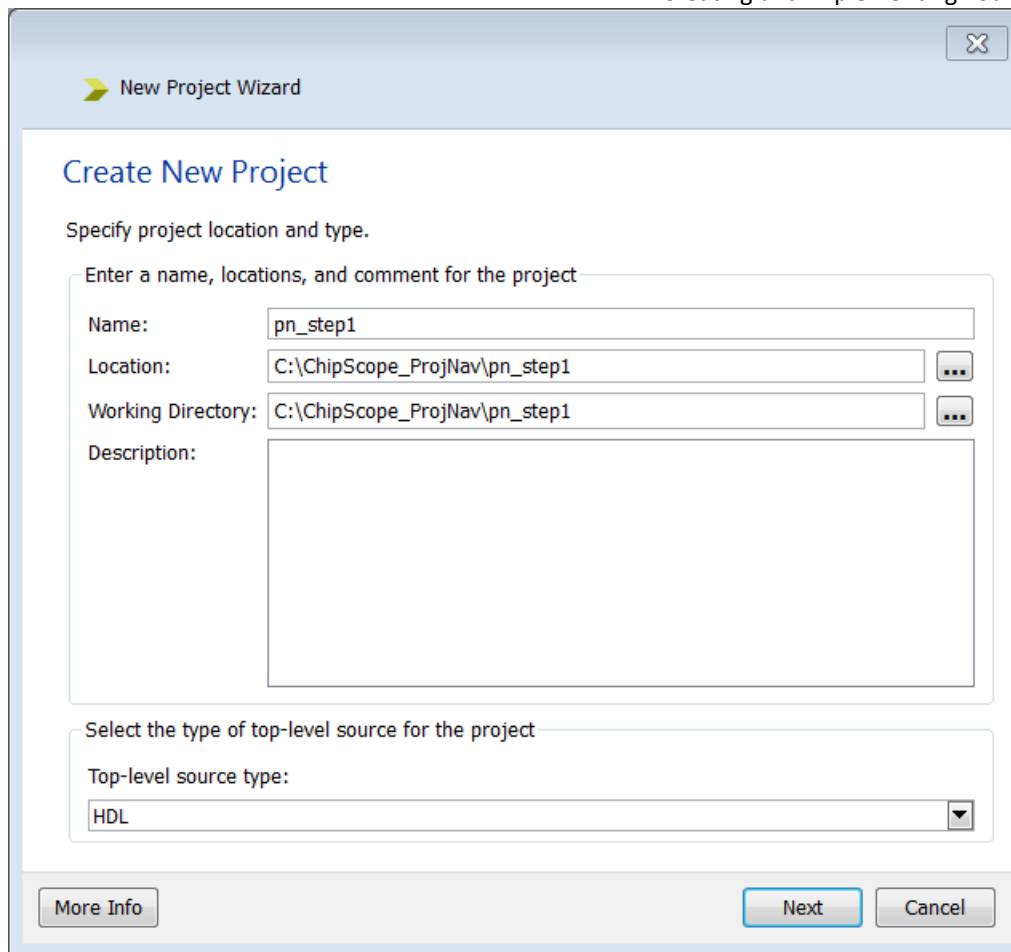


Figure 5: Create New Project Window

3. Specify device and project properties as shown here, and click **Next**.

New Project Wizard

Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX16
Package	CSG324
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

[More Info](#) [Next](#) [Cancel](#)

Figure 6: Project Settings Window

4. Verify that the Project Settings are set correctly as shown here.

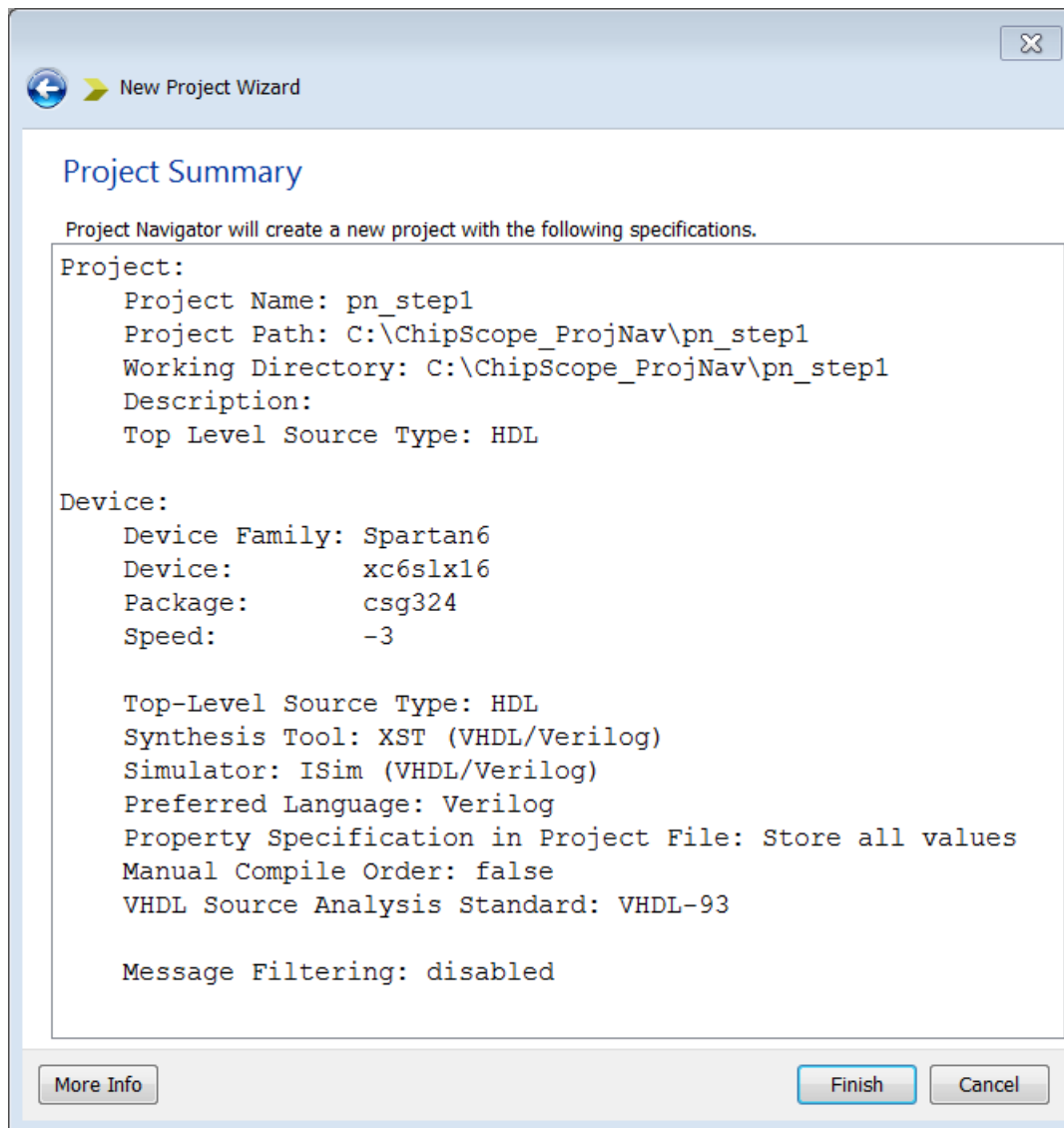


Figure 7: Project Summary Window

5. Click **Finish** on the Project Summary page.
6. Right-click the **pn_step1** project under the Design folder and select **Add Source** to add VHDL source files to the new project.

Locate the source files in the `src` folder, select all VHDL source files, all the xco files and the `sinegen_demo_sp601.ucf` file. Click on **Open**.

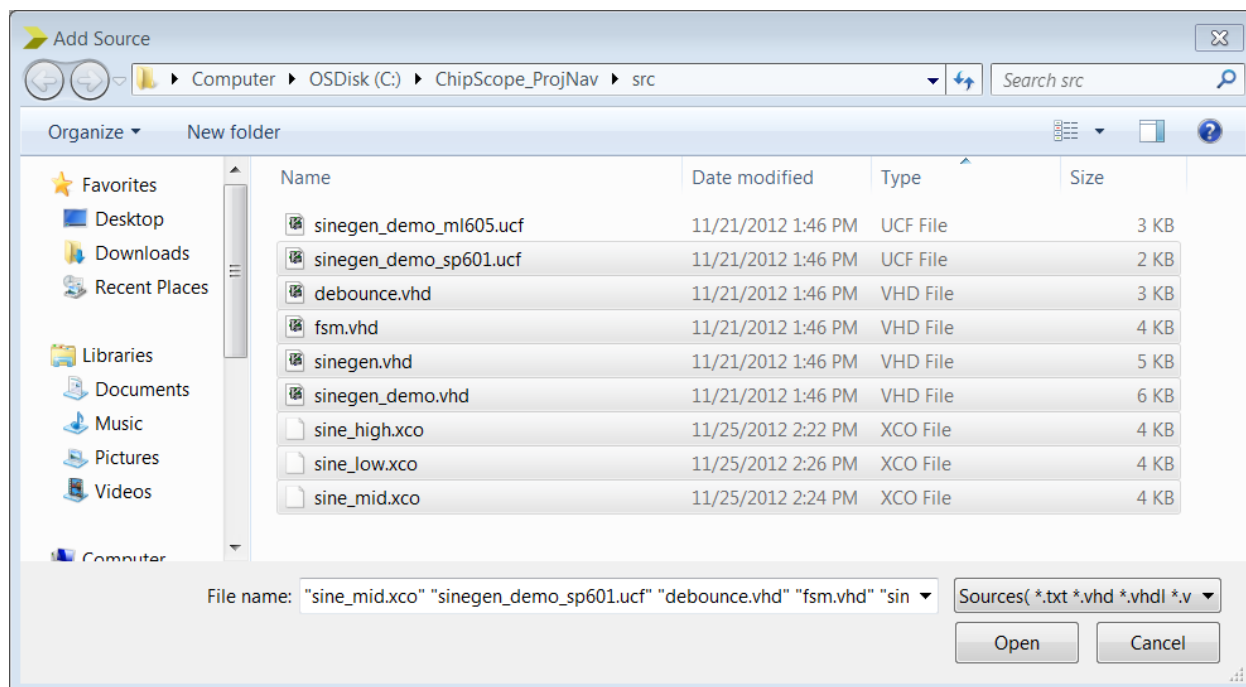


Figure 8: Source Files

- Review the files listed in the Adding Source Files window and click **OK**.

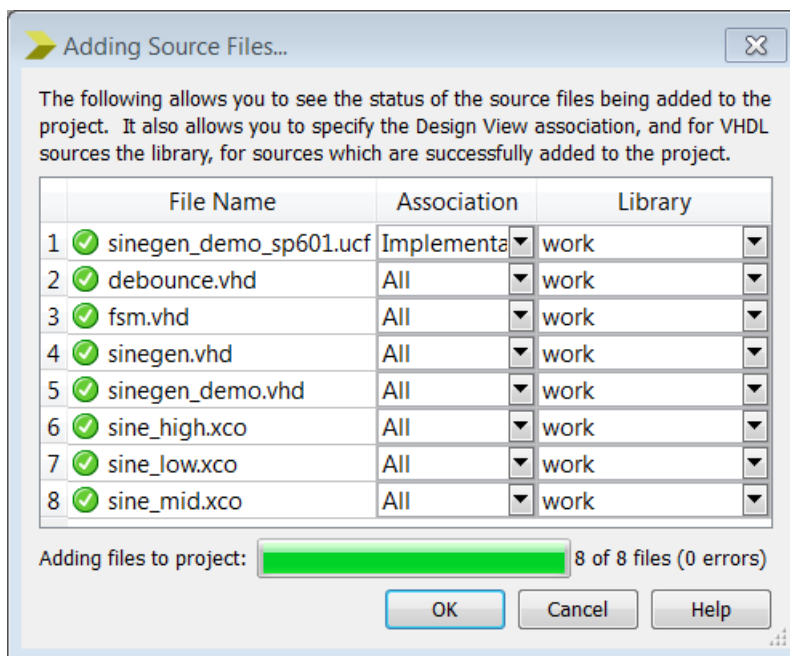


Figure 9: Adding Source Files Window

8. Right-click **sinegen_demo** and select **New Source** to add a new definition and connection (.cds) file to the project.

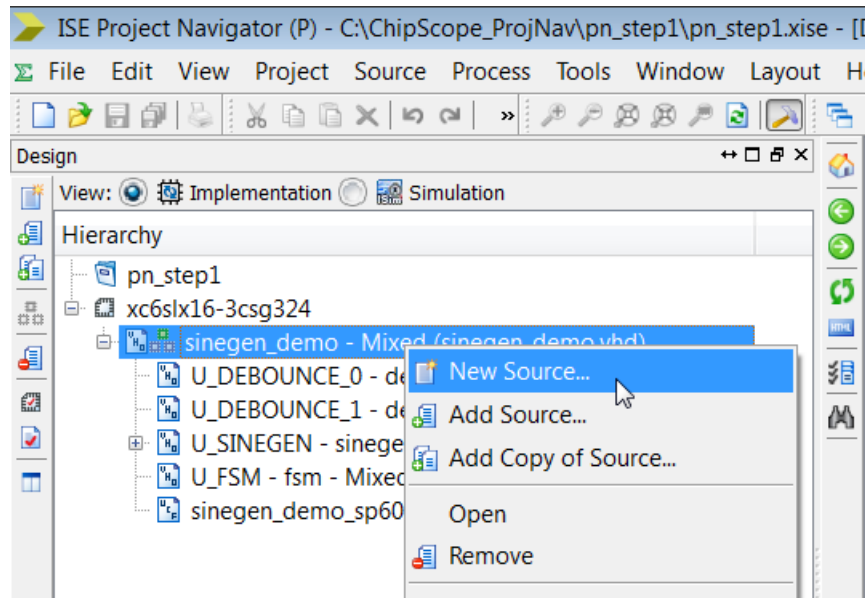


Figure 10: New Source Selection

9. In the New Source Wizard, select **ChipScope Definition and Connection File**, and then type `pn_step1.cdc` for the file name and click **Next**.

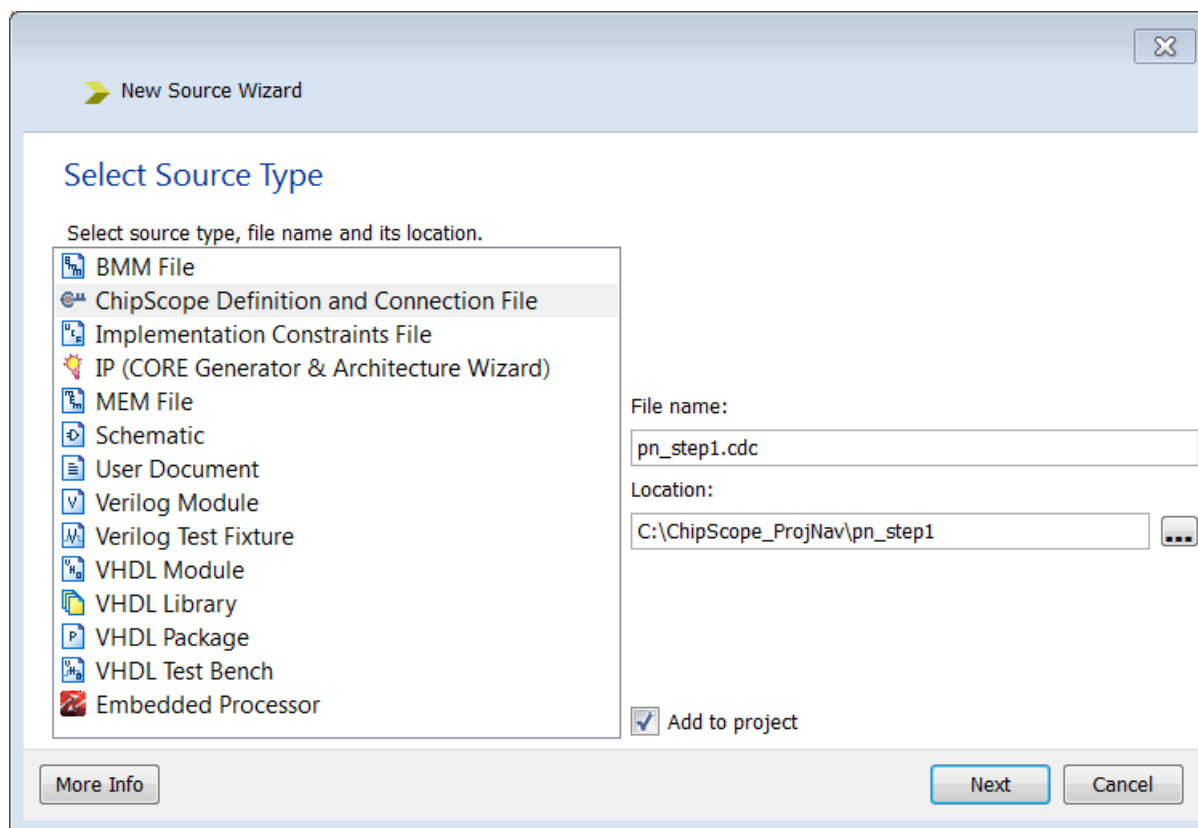


Figure 11: New Source Wizard

10. Review the information on the **Summary** page in the New Source Wizard and then click **Finish**.

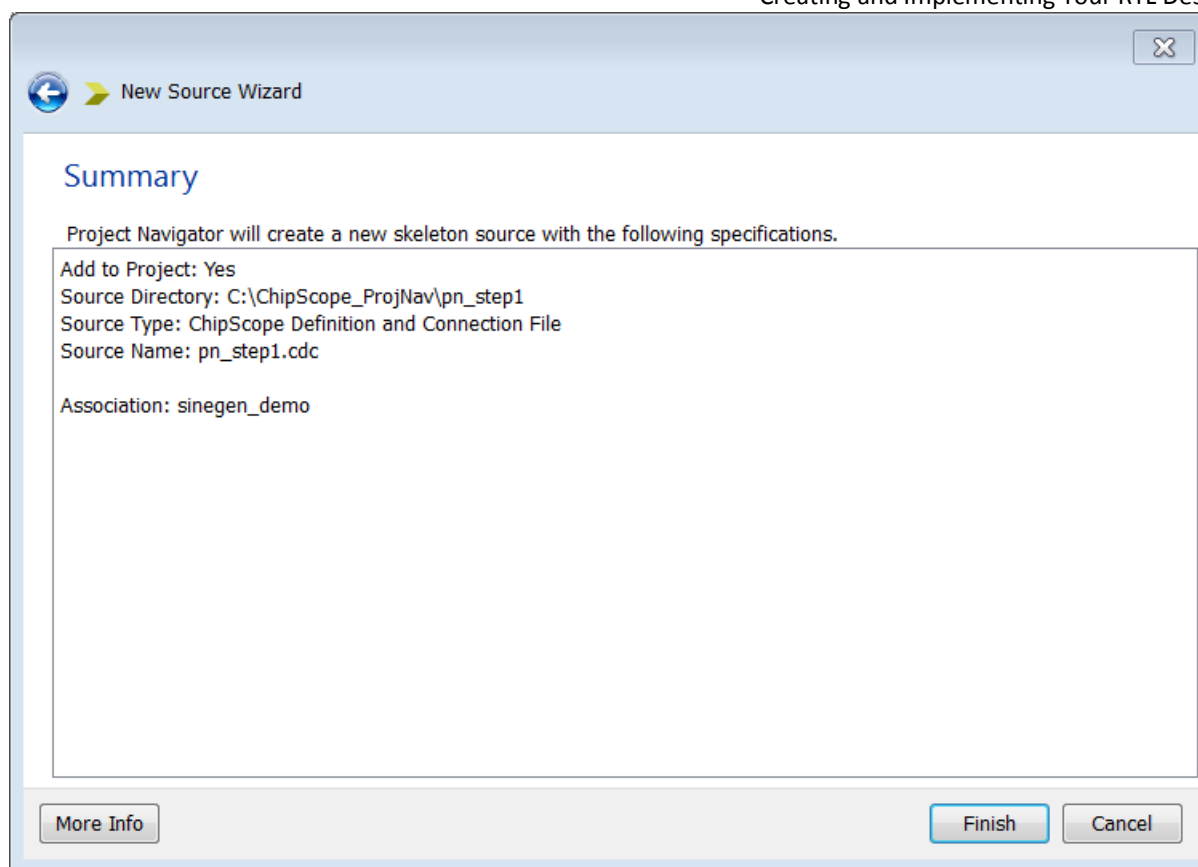


Figure 12: New Source Wizard Summary

The next step is to generate all three sine wave generator cores.

11. In the Design window, select all three .xco files listed under U_SINEGEN, and then double-click **Regenerate All Cores**.

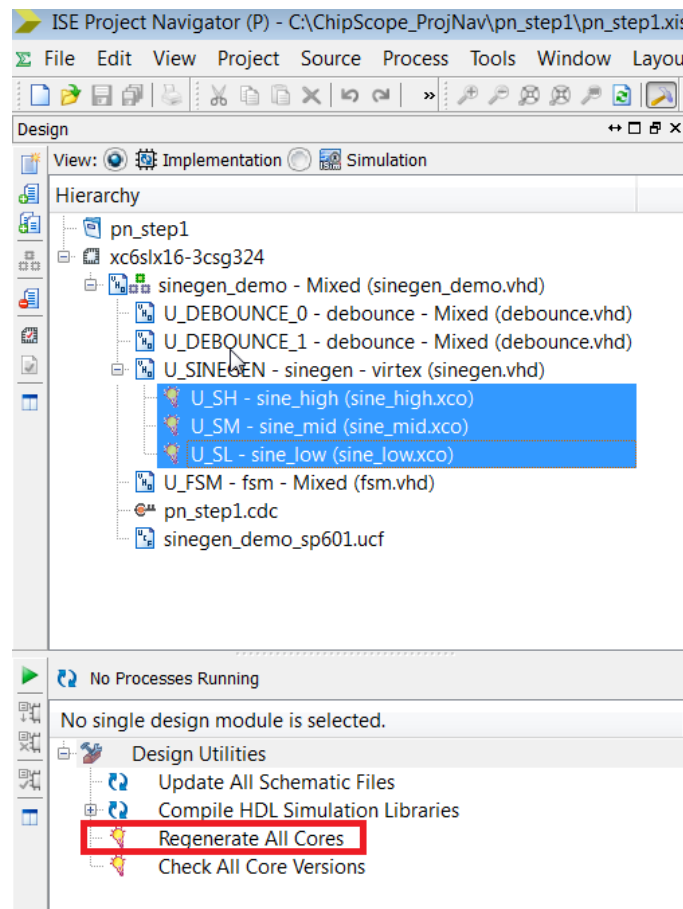


Figure 13: Sine Wave Generator Files in the Design Window

12. Prior to synthesizing the design, set the **Keep Hierarchy** option to **Soft** to preserve the design hierarchy and prevent the XST from performing hierarchical optimizations. To do this, right-click the **Synthesize** process and set the **-Keep_hierarchy** switch to **Soft**.

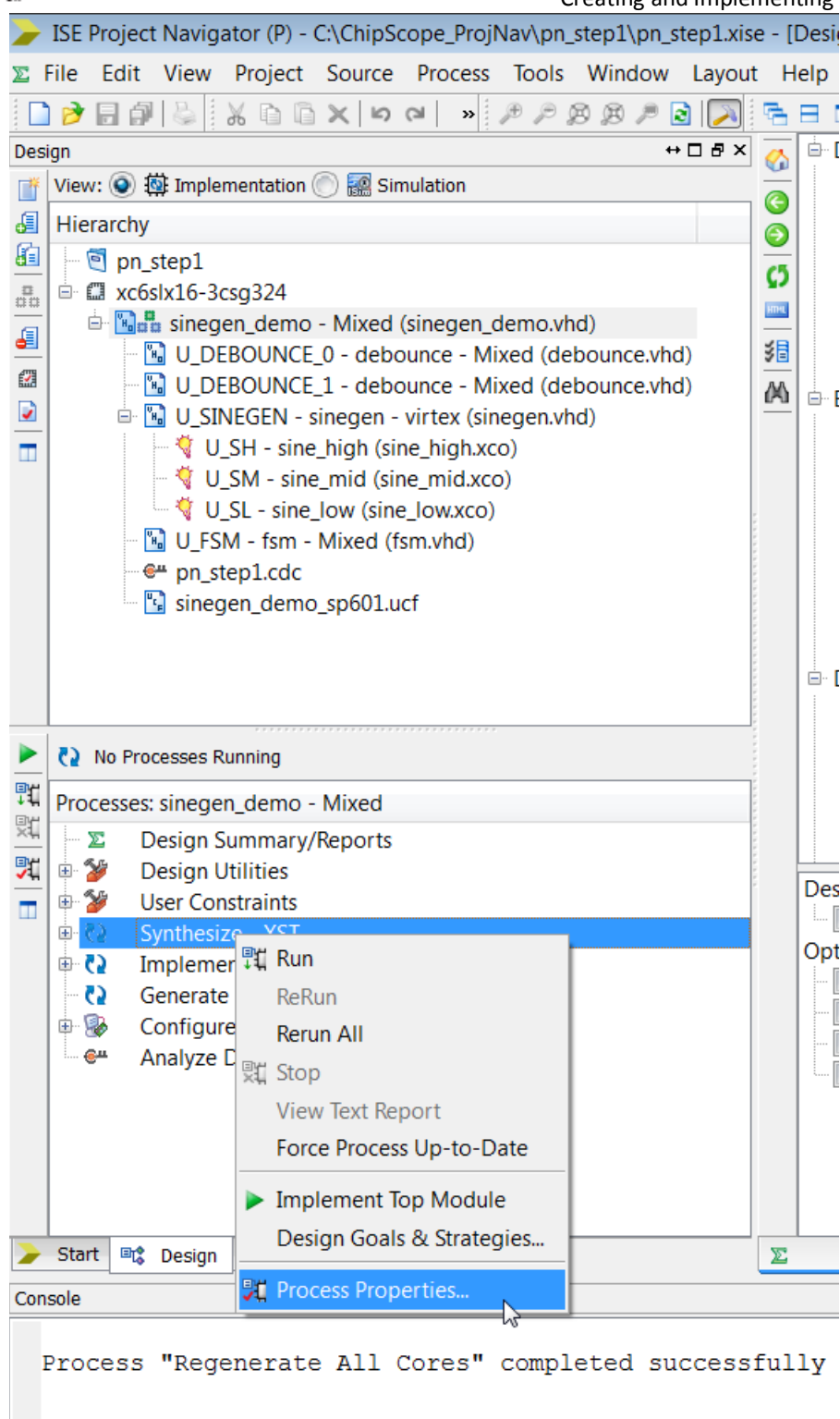


Figure 14: Changing Synthesis Process Properties

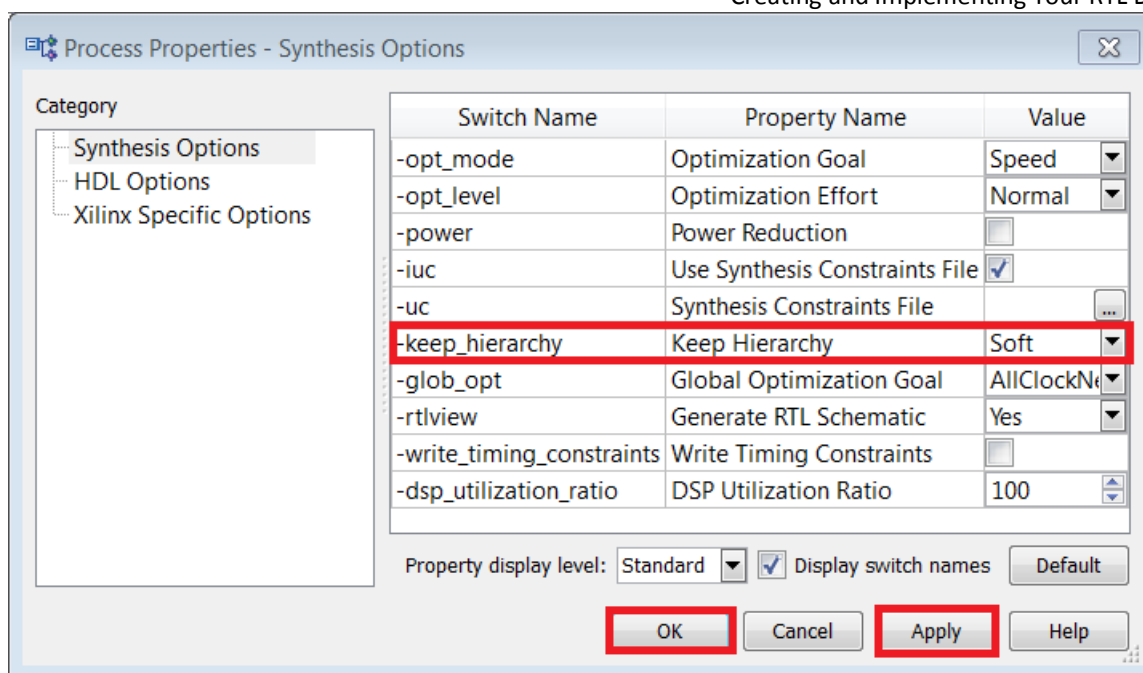


Figure 15: Synthesis Options Dialog Box

13. Click **Apply** or **OK**.

Now you are ready to synthesize the design.

14. Double click the **Synthesize** process.

15. From the menu, select **File > Copy Project** to copy the project as pn_step2 as shown here.

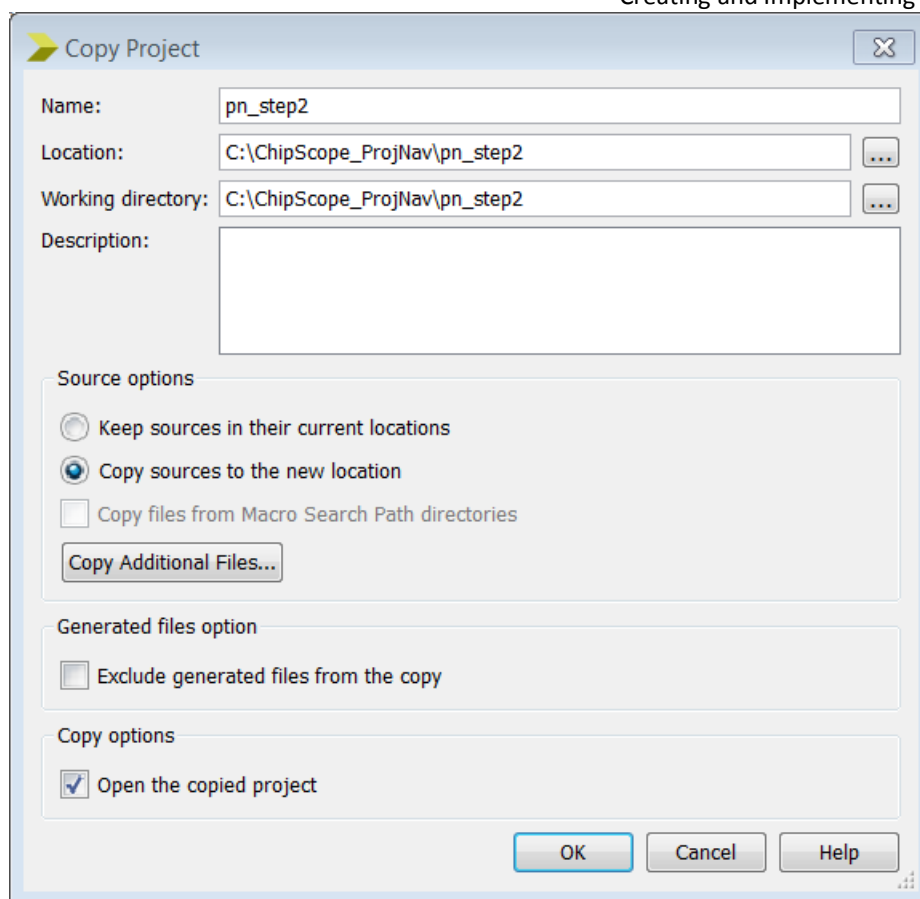


Figure 16: Copy Project Dialog Box

16. Type the project name and location. Ensure that the **Copy sources to new location** is selected and also that the **Open the copied project** box is checked. Click **OK**.

At this point, you have successfully created the ISE project and synthesized the design using Project Navigator.

Questions

1. Describe briefly of what you did in Step1: _____
2. What are some major circuits used in this lab? _____
3. Which source file would you have to modify if you were to target other Xilinx® boards?

Adding a ChipScope ILA Core to Your Design

Adding a ChipScope ILA Core

In this section of the tutorial, you will add a ChipScope™ ILA core to your design by taking advantage of integration flows between the Project Navigator and ChipScope Pro Core Inserter tools. Traditionally, you had to manually instantiate the ILA core instances into the RTL designs. However, that method requires you to modify your design source files, which can be cumbersome and can increase the chance of potentially making mistakes.

Instead, you will learn here how to add the core, complete the signal connections, and probe the design without modifying the original RTL source files.

1. In the Hierarchy window, double-click the `pn_step1.cdc` file to open the ChipScope Pro Core Inserter tool.

The first window displays Device Options. Because the target device is already set in Project Navigator, the device options are already set for you. This eliminates some potential conflicts in device selection between ChipScope Pro Core Inserter and Project Navigator.

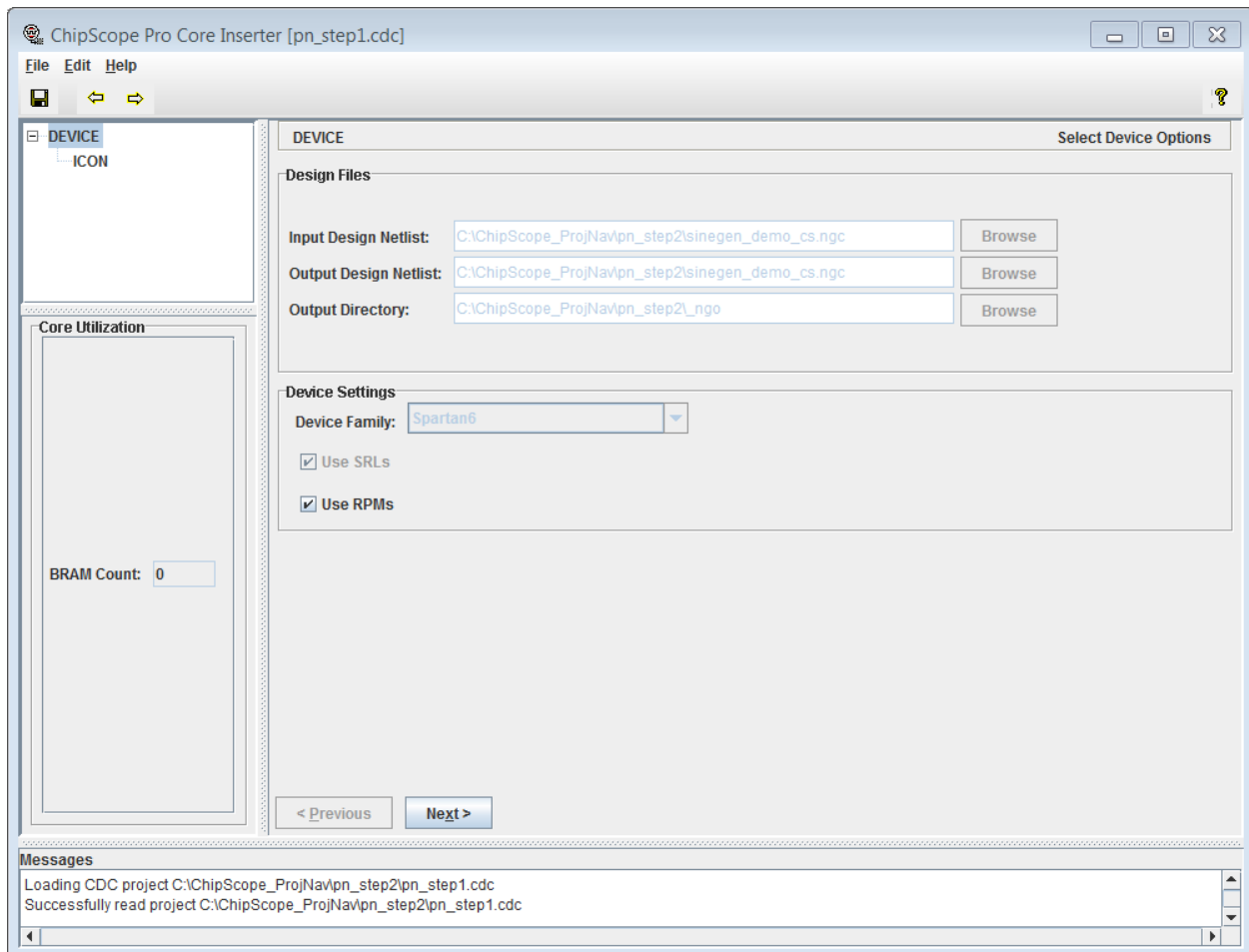


Figure 17: Chipscope Pro Core Inserter

2. Click **Next**. The window displays the ICON options.

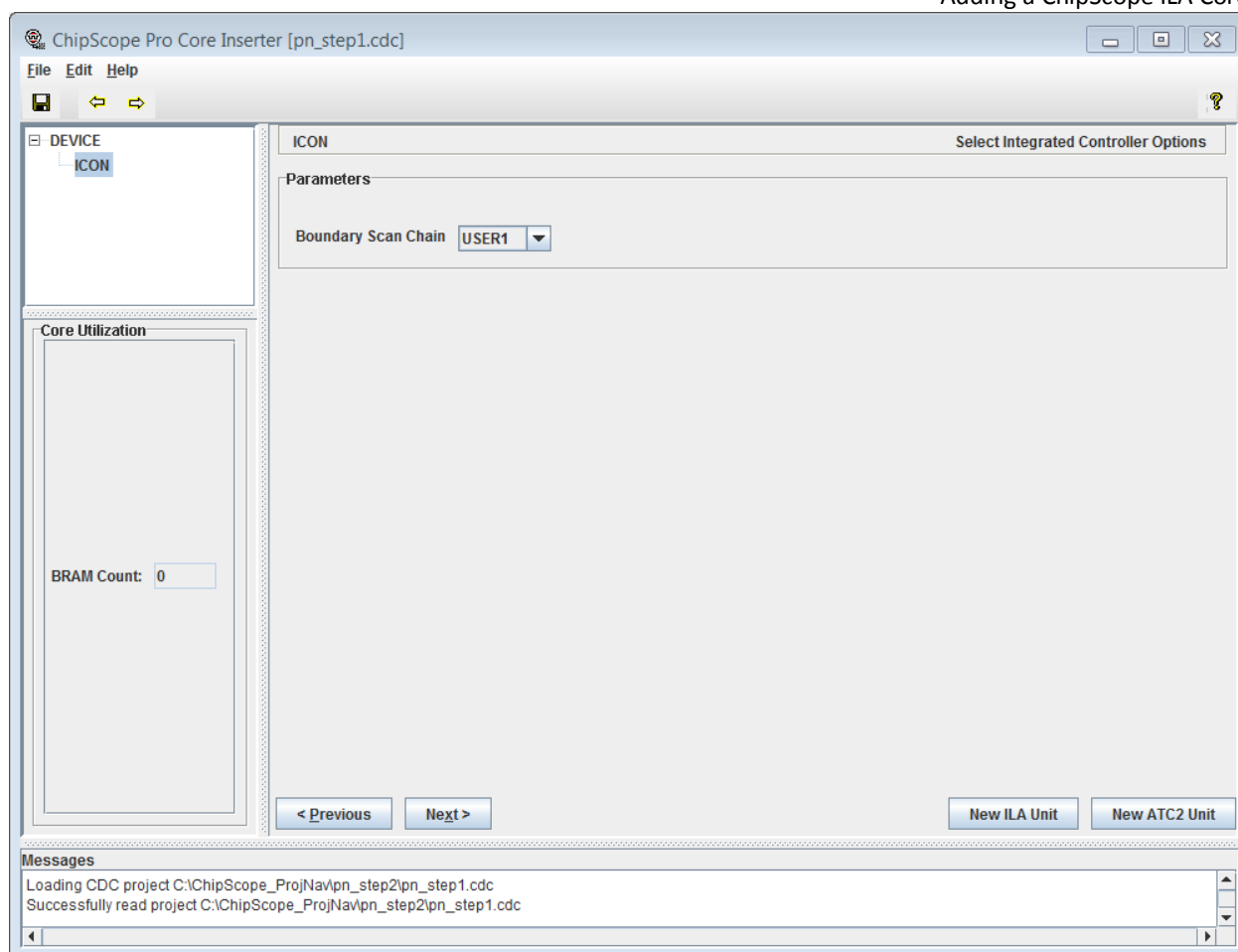


Figure 18: ICON Options

3. Click **Next** to add an ILA core.
4. In the Trigger Parameters tab of the ILA Options window, set the trigger parameters.

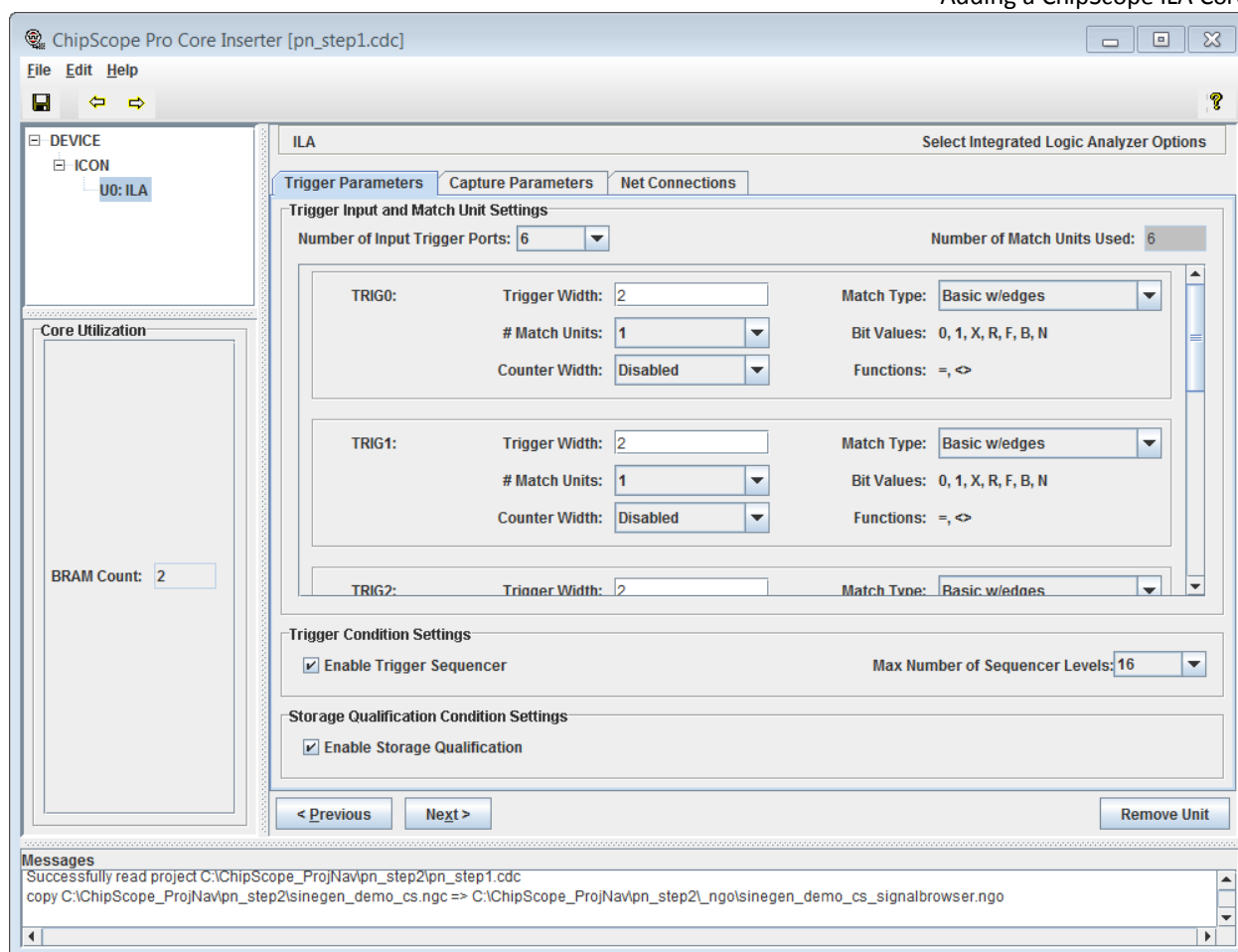


Figure 19: Trigger Parameters Tab of the ILA Options Window

For this example, set the Number of Trigger ports to 6 and the rest of the parameters as follows.

Trigger Name	Trigger Width	Match Type
TRIG0	2	Basic w/ edges
TRIG1	2	Basic w/ edges
TRIG2	2	Basic w/ edges
TRIG3	20	Basic
TRIG4	2	Basic w/ edges
TRIG5	3	Basic w/ edges

5. Select the **Capture Parameters** tab and examine the trigger parameters you just set.

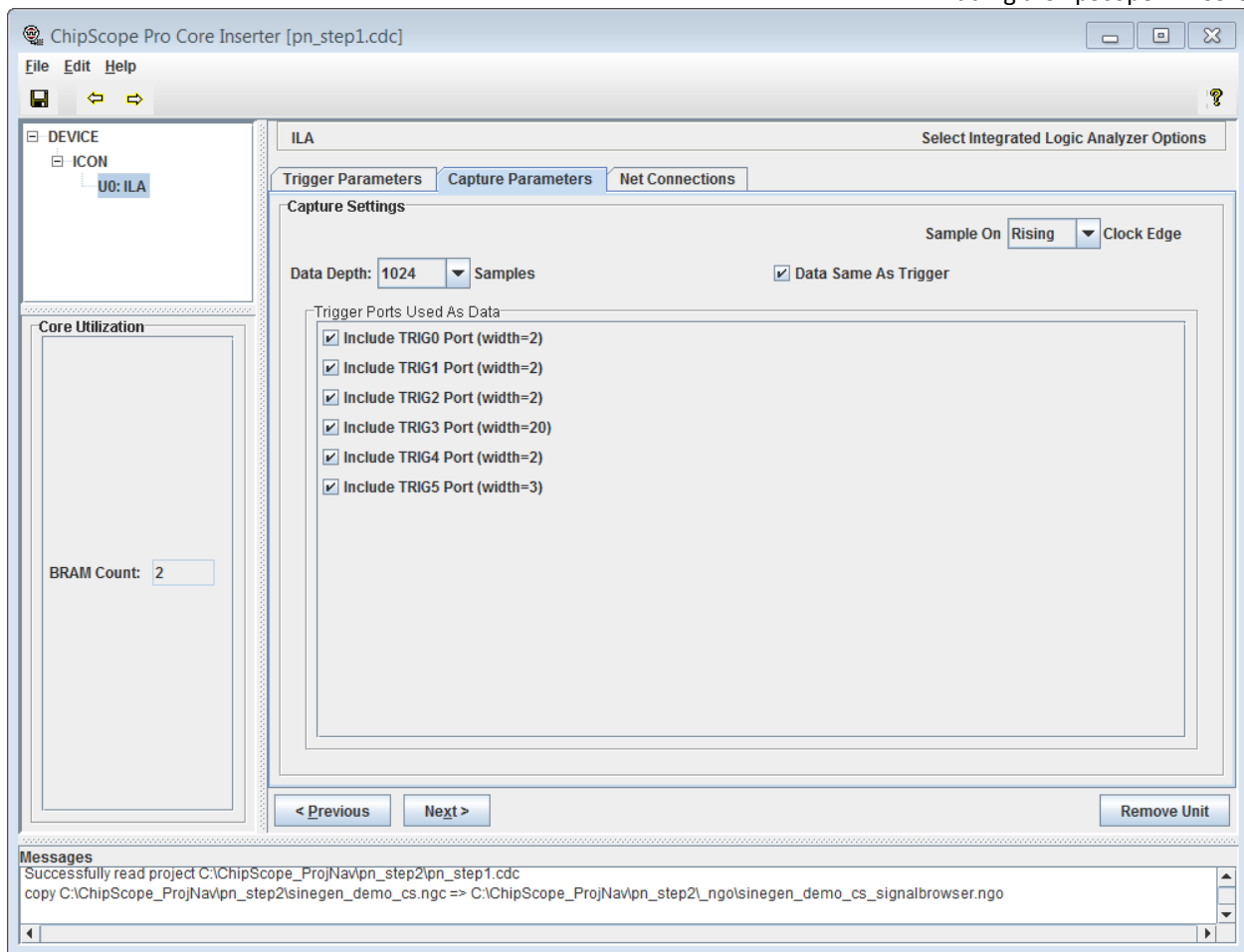


Figure 20: Capture Parameter Tab

6. Connect each port to debugging nets. To do this:
 - a. Select the **Net Connections** tab as shown here.

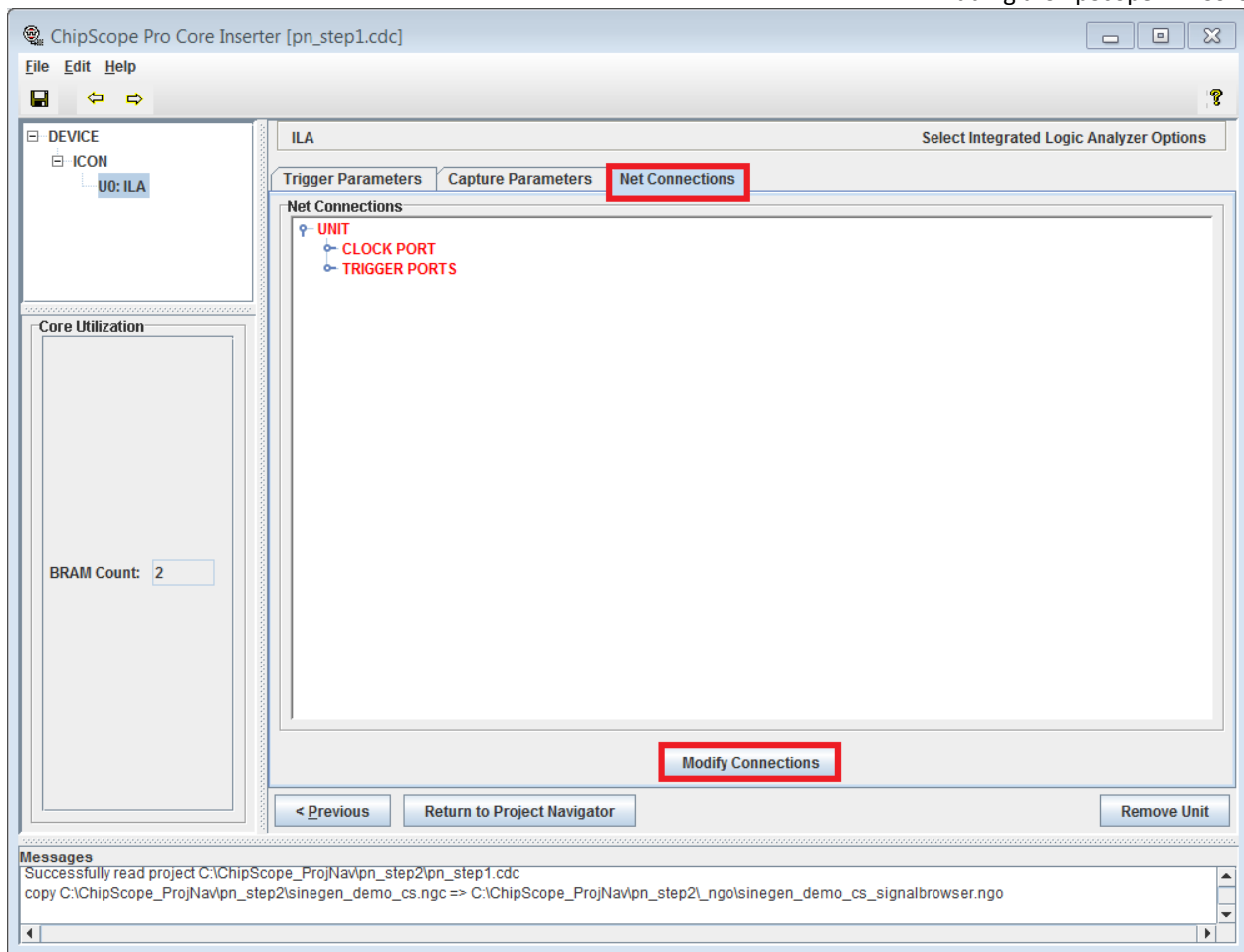


Figure 21: ChipScope Core Inserter Dialog Box

- b. Click **Modify Connections**. The Select Net dialog window appears.
- c. From the Select Net window, search for the clk_BUFG net from the sinegen_demo hierarchy. To search for the "clk_BUFG" net, type the "clk_BUFG" string in the Pattern field and click Filter.
- d. Select the **clk_BUFG** net from search results and click **Make Connections**.

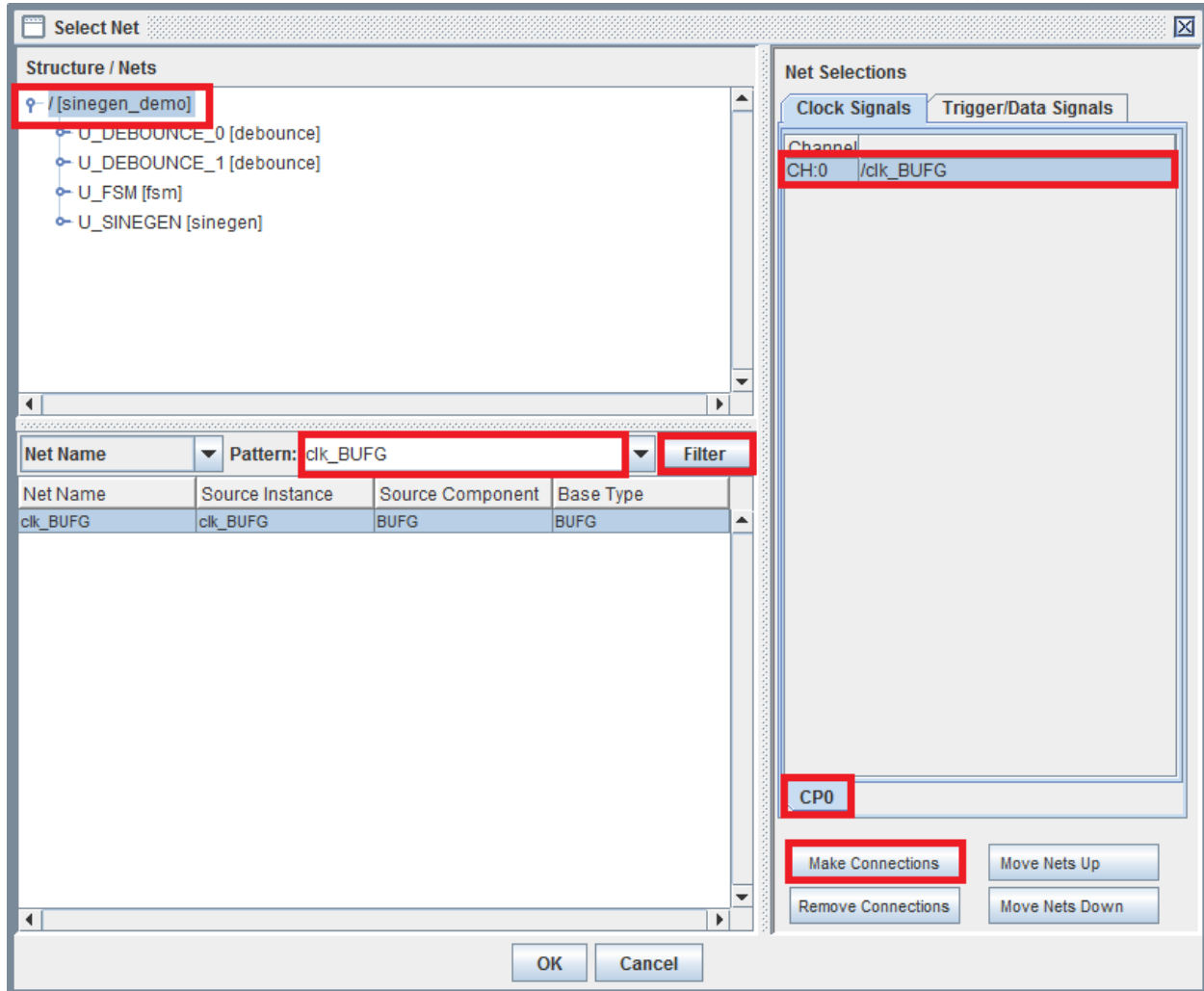


Figure 22: Select Net Window

7. Repeat the previous step to connect the rest of trigger ports:

- TP0: search for *sel* from the U_SINEGEN hierarchy
- TP1: search for *GPIO_BUTTONS_re* from the sinegen_demo hierarchy
- TP2: search for *GPIO_BUTTONS_dly* from the sinegen_demo hierarchy
- TP3: search for *SINE* from the sinegen_demo hierarchy
- TP4: search for *GPIO_BUTTONS_db<0>* from the sinegen_demo hierarchy
- TP4: search for *GPIO_BUTTONS_db<1>* from the sinegen_demo hierarchy
- TP5: search for *GPIO_BUTTONS_0_IBUF* from the sinegen_demo hierarchy
- TP5: search for *GPIO_BUTTONS_1_IBUF* from the sinegen_demo hierarchy
- TP5: search for *GPIO_SWITCH_IBUF* from the sinegen_demo hierarchy

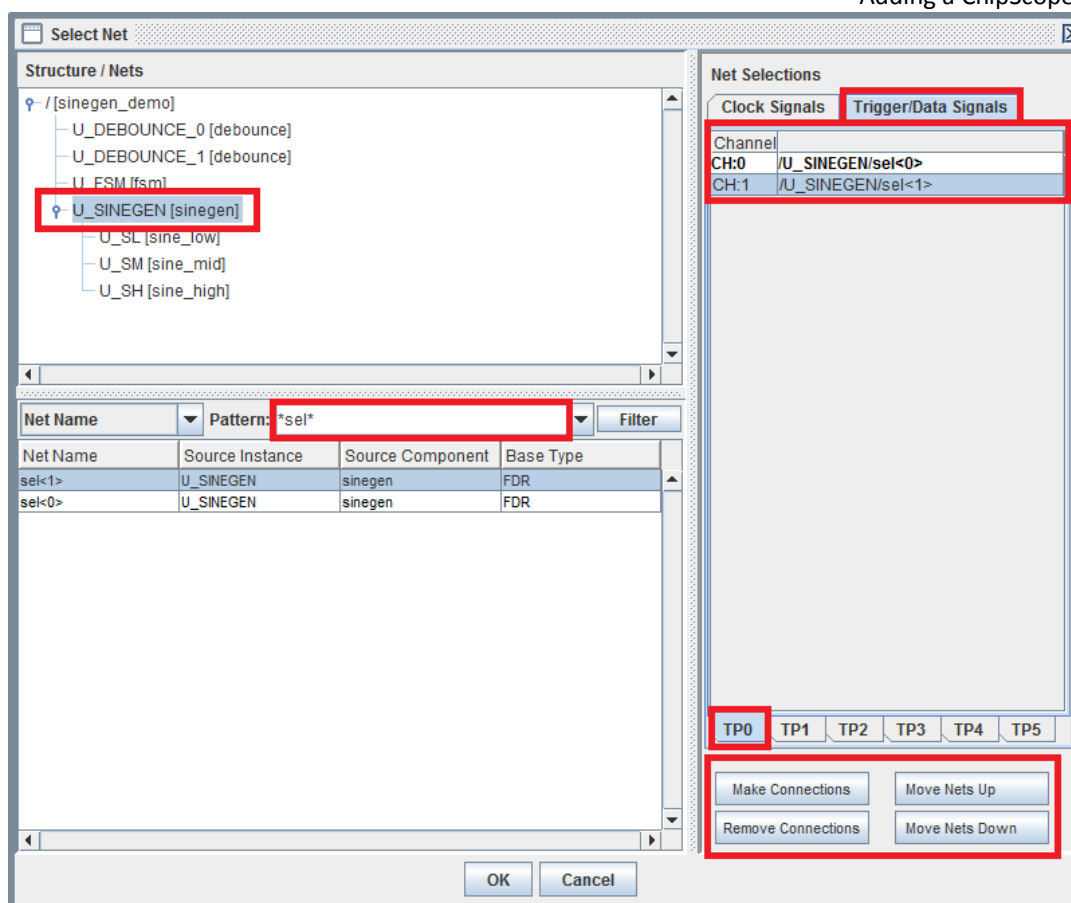


Figure 23: Trigger/Data Signals

- Once complete, all ports turn from red to black, indicating that you have finished connecting all the clock and trigger ports to debugging nets. Click OK. Back in the ChipScope Pro Core Inserter dialog box you can expand the **Net Connections** to verify that all appropriate connections were made.

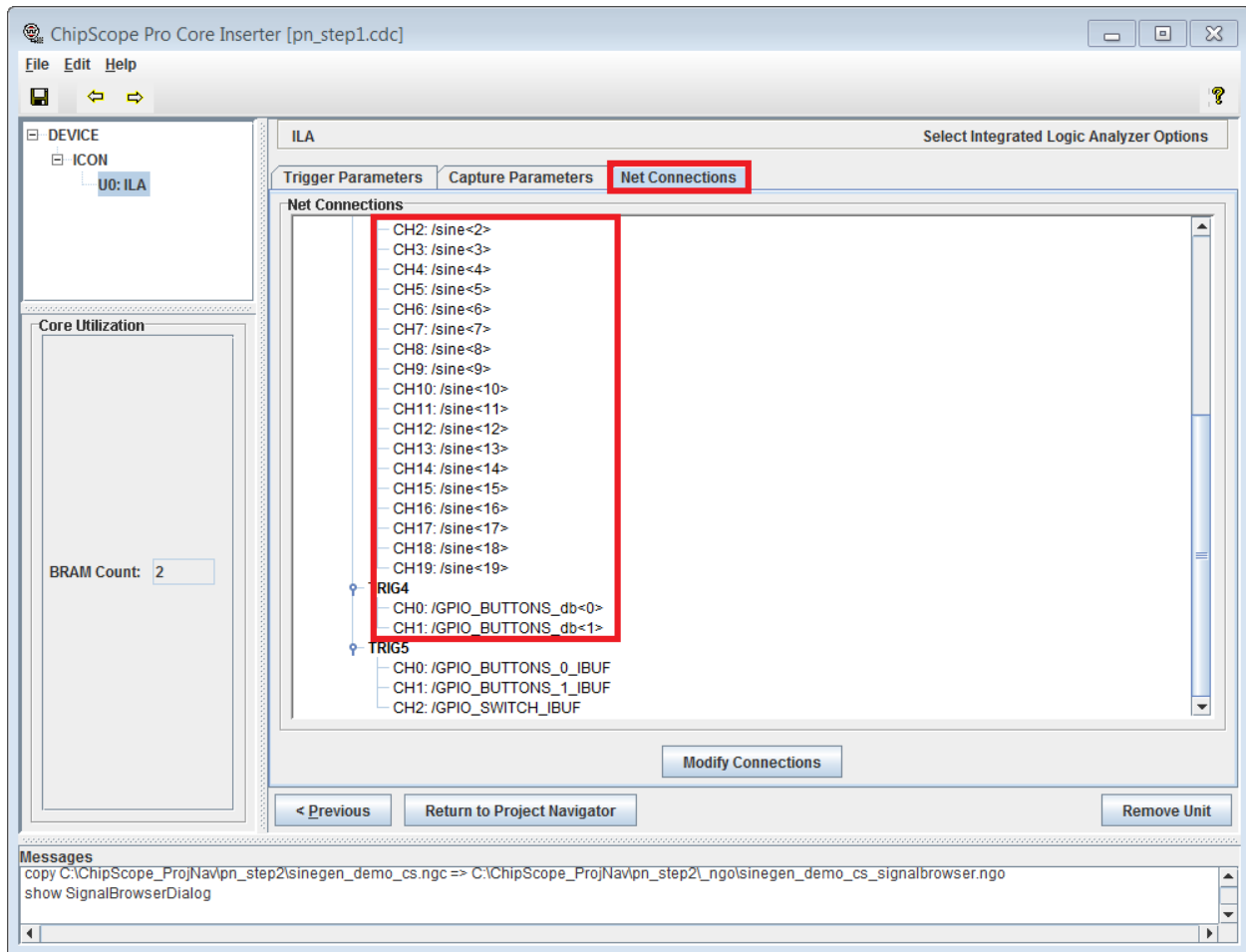


Figure 24: Complete Net Connections

9. Save and close the `pn_step1.cdc` file by clicking on **File** and selecting **Save**. Click on **Return to Project Navigator** Before you use the ChipScope Pro Analyzer tool to download your bitstream into your device, make sure the bitstream generation options are set properly.
10. With **sinegen_demo** selected in the Design browser, right-click on the **Generate Programming File** process and select **Process Properties**.

11. In the **Startup Options** category, set the **-g StartUpClk** switch to the **JTAG Clock** option. Click **Apply**, then **OK**.

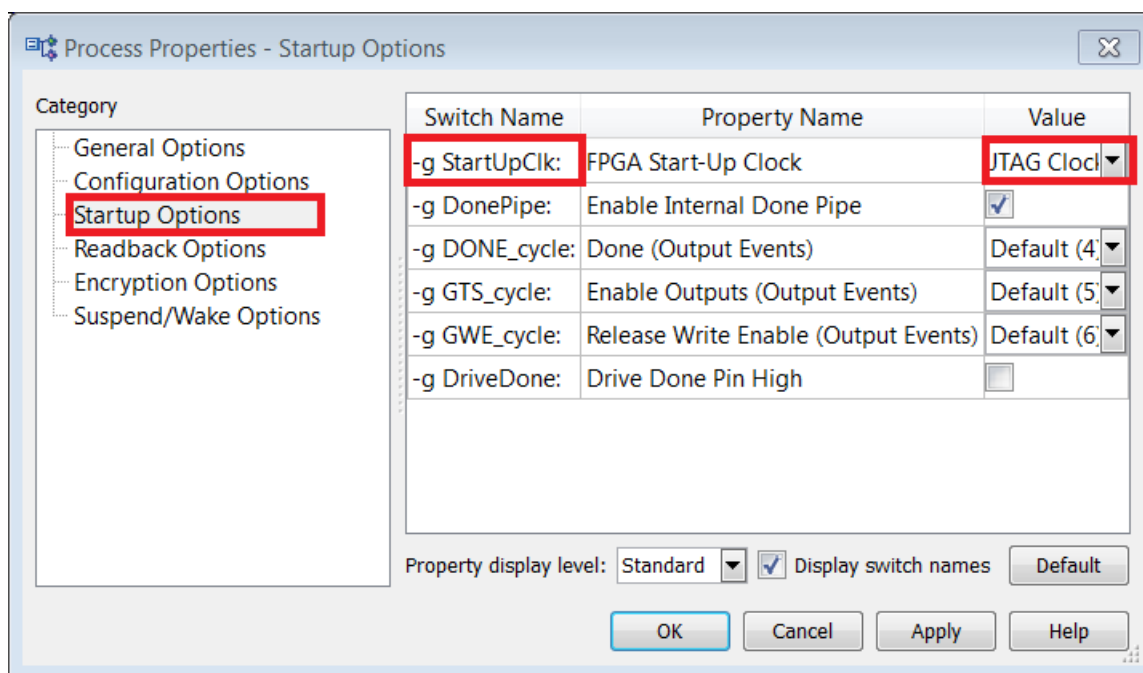


Figure 25: Startup Options Window

Now you can start generating a programming file.

12. Double-click the Analyze Design Using ChipScope process.

When the process completes, the ChipScope Pro Analyzer tool opens.

Questions

4. What is the main advantage of inserting debug probes onto your post-synthesis netlist instead of adding them onto HDL design files? _____

You have just finished with inserting a ChipScope ILA core and now you are ready to debug the design using ChipScope Pro Analyzer.

Debugging Your Design using ChipScope Pro Analyzer

Debugging Your Design

This lab exercise shows how to debug a design using Xilinx® ChipScope™ Pro Analyzer and how to iterate the design once you have discovered and fixed errors. This step will also show some useful techniques of how to trigger and capture certain data from your design.

You will be using ChipScope Pro Analyzer to verify that Sine wave generator is working correctly. The two primary objectives will be to verify and check off the following two items.

- Verify that all sine wave selections look correct
- Verify that selection logic is working correctly

Do the following:

1. Configure JTAG Chain to USB cable and communication parameters by doing the following:
 - a. Select **JTAG Chain > Xilinx Platform USB Cable**.
 - b. In the ChipScope Pro Analyzer dialog box that opens, set the speed and port parameters.

For this tutorial, Speed is 3 MHz and Port is USB21.

- c. Click **OK**.
- d. In the ChipScope Pro Analyzer dialog box that opens, verify the device details and then click **OK**.

2. Confirm the connection to the JTAG chain.

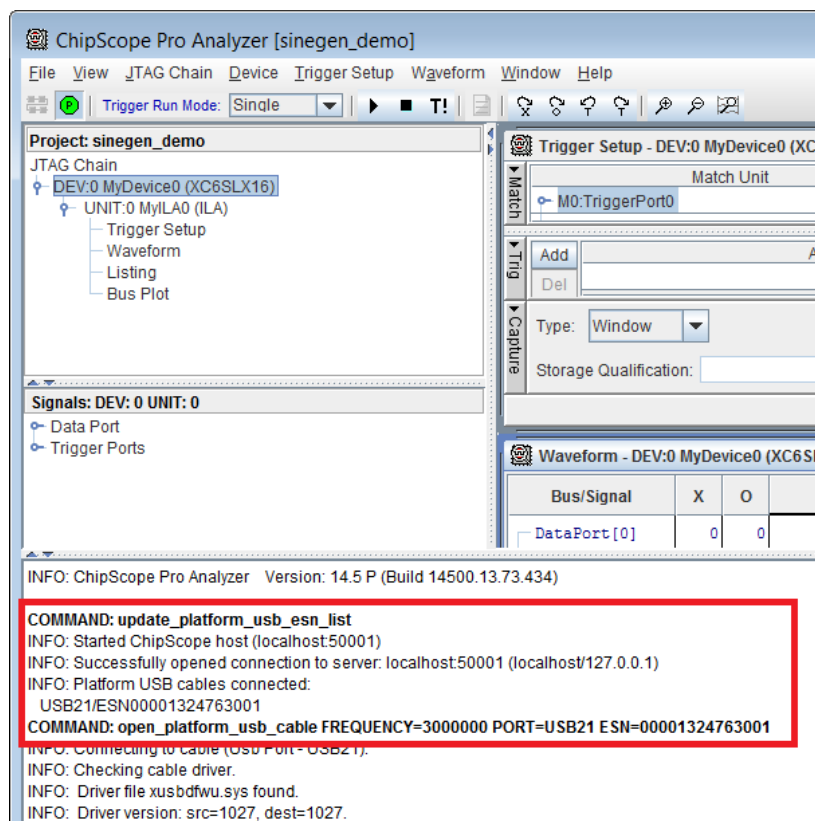


Figure 26: JTAG Chain Connection Details

3. Right-click the device name in the Project list and select **Configure** to configure the device.
4. In the Configuration window, select the default BIT and CDC files from Project Navigator.

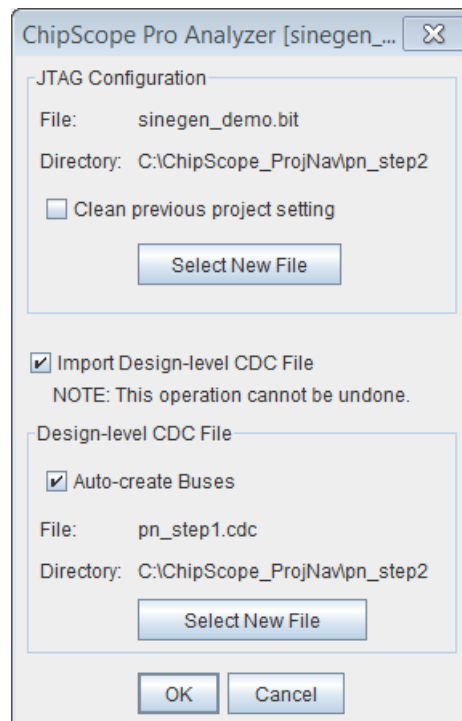


Figure 27: JTAG Chain Connection Details

5. Verify the device configuration and ILA core.

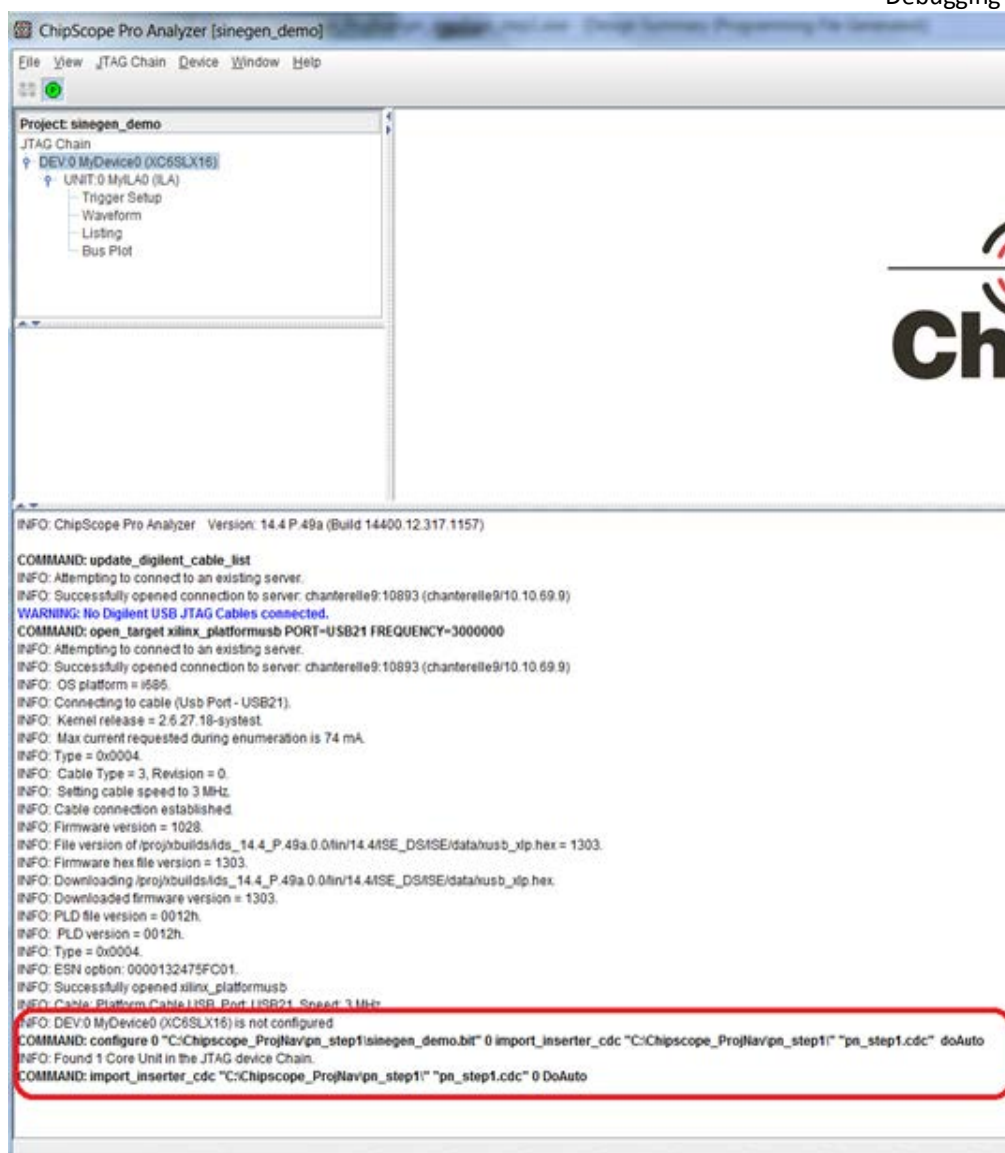


Figure 28: Device Configuration Details

- Open the Trigger Setup and Waveform windows by double-clicking on each item: **Trigger Setup** and **Waveform**.

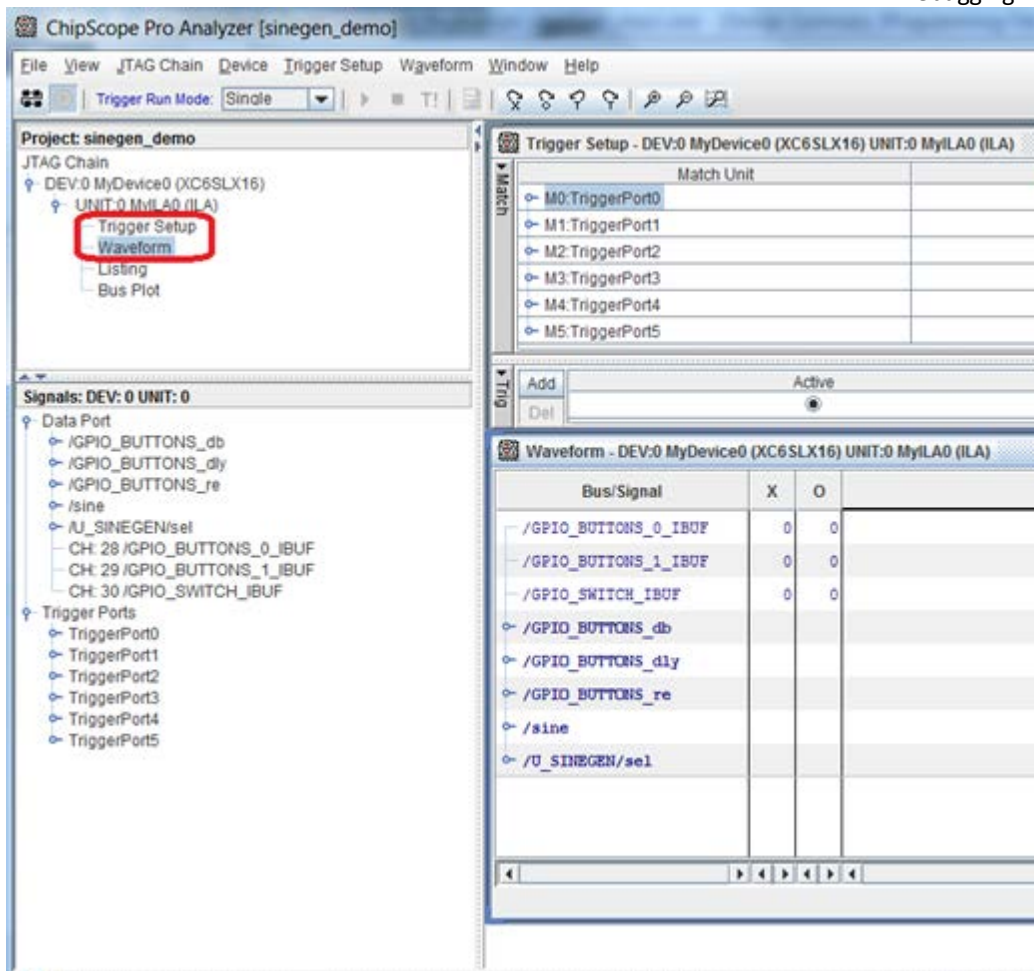


Figure 29: Location of Trigger Setup and Waveform

7. Select **Trigger Setup > Trigger Immediate**.
8. Verify that there is activity on the sine wave.

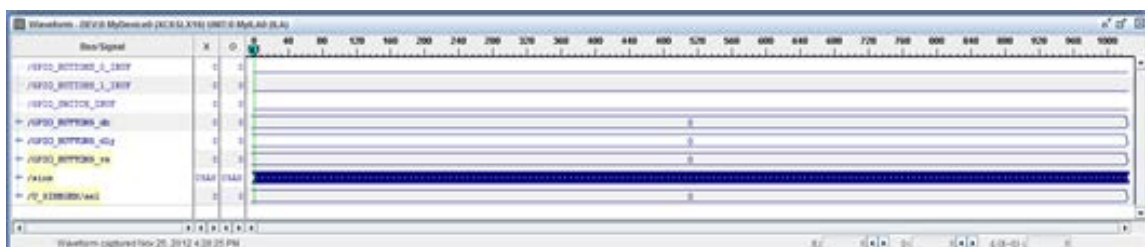


Figure 30: Sine Wave Activity

9. Double-click **Bus Plot** to open the Bus Plot viewer.
10. On the Bus Plot window, select **/sine** to display the sine wave.

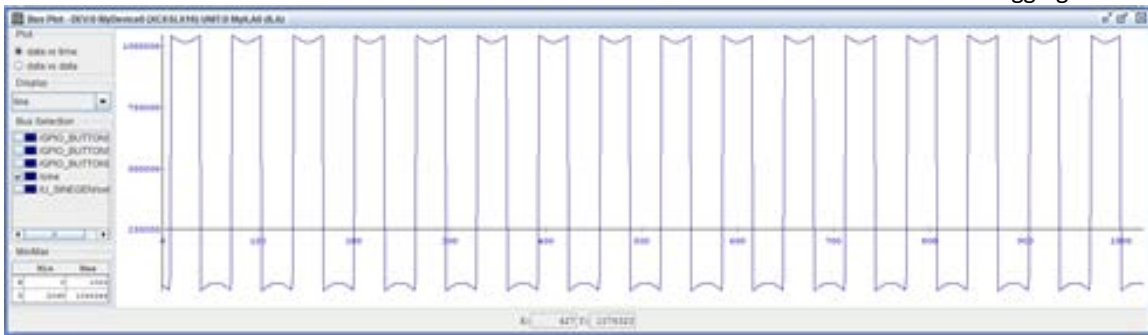


Figure 31: Sine Wave Display

Notice that this waveform does not look much like a sine wave. This is because the radix setting needs to be changed to from Hex to Signed Decimal.

11. In the Signal window, right-click **/sine** and select **Bus Radix**. Click to select the **Signed Decimal** check box.
12. Click the **Trigger Immediately** button, and view the high-frequency sine wave bus plot.

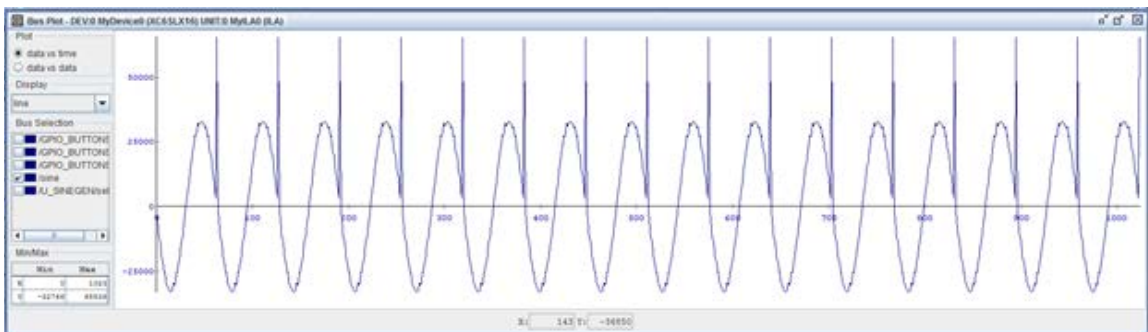


Figure 32: High Frequency Sine Wave Bus Plot

13. On your board, push the Sine Wave Sequencer button (shown in Figure 1-1) until the Sine Wave Selection indicator LEDs on the board display "off,on, 01."

Note: Sequencer is not working correctly. The expected behavior is a simple 2-bit counter that counts for each press (00, 01, 10, 11...). You will debug the root cause for this later in this tutorial.

14. Click the **Trigger Immediately** button again, and view the mid-frequency bus plot.

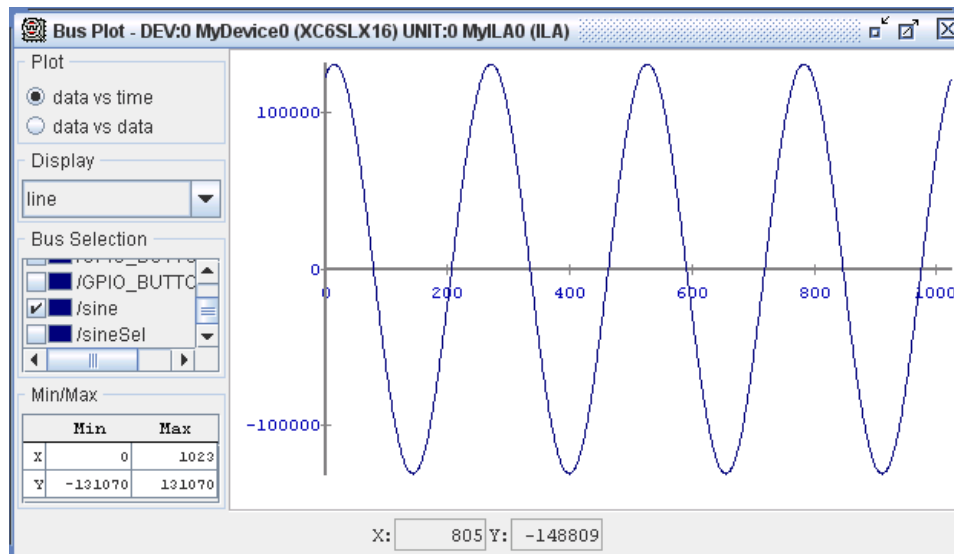


Figure 33: Mid Frequency Sine Wave Bus Plot

15. Push the Sine Wave Sequencer button on the board until the Sine Wave Selection indicator LEDs on the board display "on, off, 10."

16. Click the **Trigger Immediately** button and view the low-frequency bus plot.

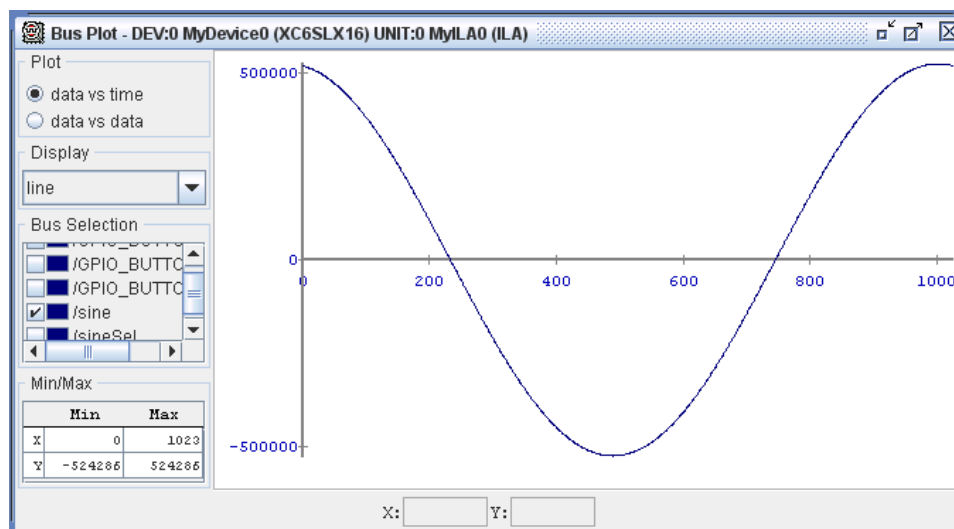


Figure 34: Low Frequency Sine Wave Bus Plot

17. Push the Sine Wave Sequencer button on the board until the Sine Wave Selection indicator LEDs on the board display "on, on, 11."

18. Click the **Trigger Immediately** button and view the combined sine wave bus plot.

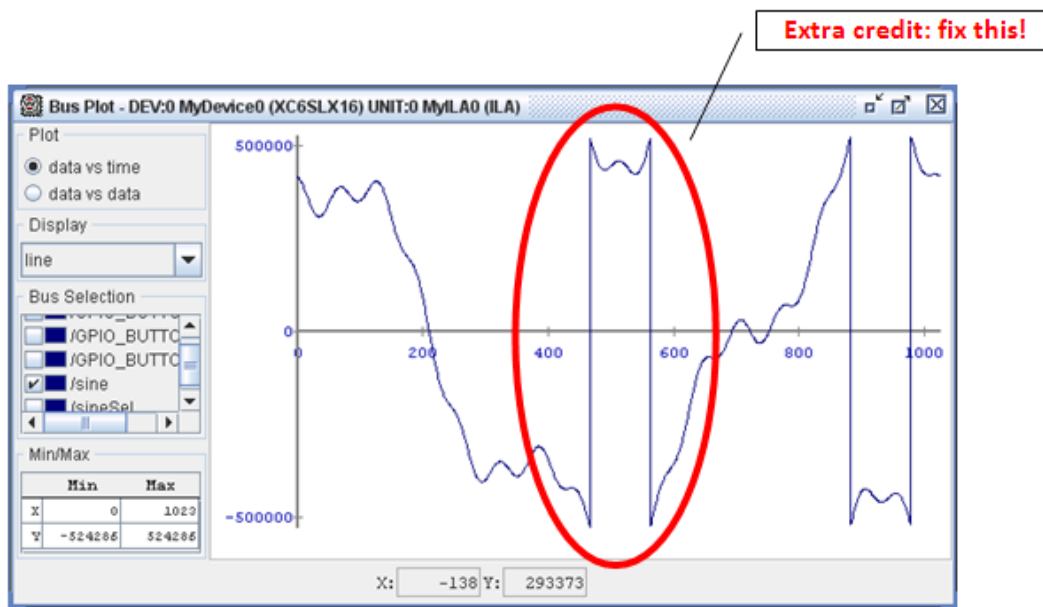


Figure 35: Combined Sine Wave Bus Plot

You just verified that all sine wave selections look correct. However, the selection logic circuit is still not working correctly.

- Verify that all sine wave selections look correct.
- Verify that selection logic is working correctly:
 - Verify that state machine is transitioning correctly and outputs are correct.
 - Verify that state machine inputs are correct.

Next, start debugging the selection logic circuit.

19. Set the following parameters:

- **Match:** set TriggerPort1 to **RX** -- to look for rising edge on GPIO_BUTTONS_re[1] because this is what causes FSM to transition.
- **Trigger Conditions:** M1 -- set up trigger equation to **M1**
- **Capture Settings:** Windows = 10; Depth = 2

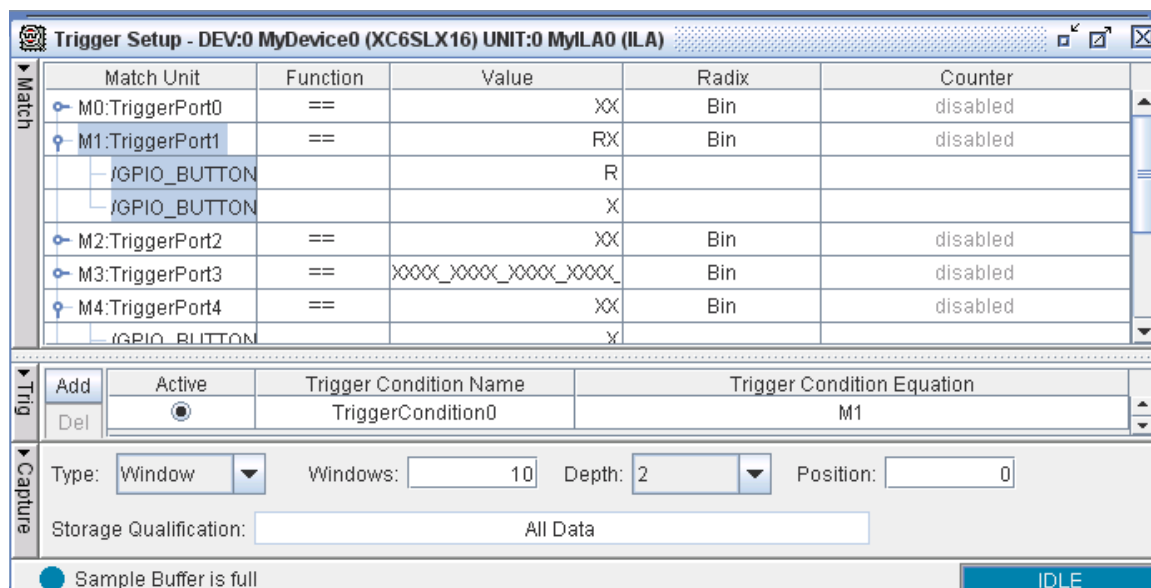



Figure 36: Trigger Setup Window

Note: The actual TriggerPort number for the GPIO_BUTTONS_re[1] net might not be the same as specified on this tutorial.

20. Click the **Run Trigger** button  to arm the trigger and wait for the trigger condition. Information about the run progress displays at the bottom of the window.
21. Push the Sine Wave Sequencer button on the board.
22. Observe the number of windows captured.
 - If only one window was captured, repeat steps 21 and 22.
 - If more than one window was captured, go to the next step.

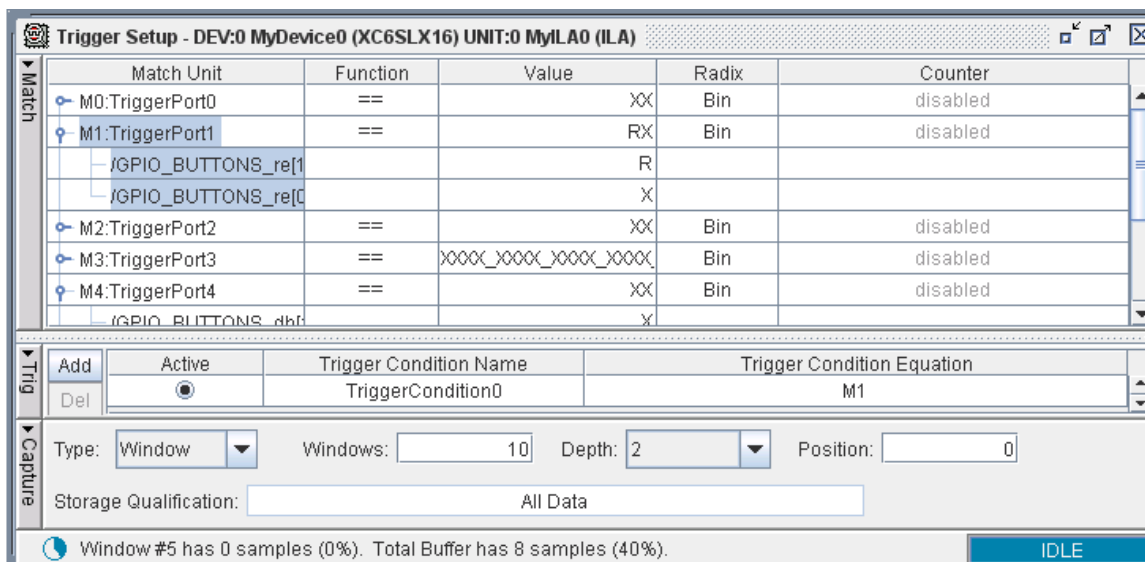


Figure 37: Observing the Windows Captured

23. Click the Stop Trigger button , and view the captured data.


Notice the multiple rising edges each time you pressed the Sine Wave Sequencer button. Also, note the correct transition of sineSel, which indicates that the state machine is working properly.

You just verified that sine wave generators are working correctly.

- Verify that all sine wave selections look correct.
- Verify that selection logic is working correctly:
- Verify that state machine is transitioning correctly and outputs are correct.
- Verify that state machine inputs are correct.

24. Set the following parameters for trigger modes and conditions:

- Trigger Run Mode: Repetitive
- Match: set TriggerPort5 to XRX -- to look for rising edge on GPIO_BUTTONS_1_IBUF, which is the input buffer for the Sine Wave Sequencer button on the board.
- Trigger Conditions: M5
- Capture Settings:
 - Windows = 1
 - Depth = 1024
 - Position = 512

25. Click the Run Trigger button  and press the Sine Wave Sequencer button on the board until you see multiple transitions on the GPIO_BUTTONS_1_IBUF signal.

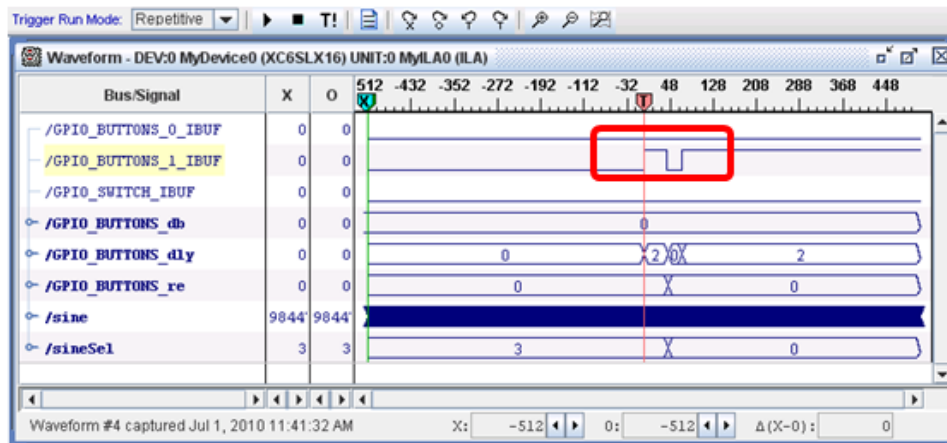


Figure 38: View Waveform

Note: Your waveform might not display signal glitches at exactly the same location as shown here. This is one of the advantages of the Repetitive triggering feature.

You just verified that sine wave generators are working correctly. However, the state machine inputs are not correct. These inputs are connected directly from Push Button switches.

- Verify that all sine wave selections look correct.
- Verify that selection logic is working correctly:
 - Verify that state machine is transitioning correctly and outputs are correct.
 - Verify that state machine inputs are correct.

As shown in Figure 32, the problem seems to point to Push Button switches, which generate glitches every time the Push Button switch is pressed and released. A debounce circuit is required for each push button switch to eliminate these glitches that result in multiple transitions.

The debounce circuits have already been integrated in the provided design. To enable debounce circuits, turn on Dip switch-1, repeat steps 24 and 25, and verify there is only one transition for each button push.

Questions

5. Did you have time to resolve the problem on Step 18 for extra credit? _____
6. Why is a debounce circuit required for this lab? _____

Tutorial Conclusion

Conclusion

This tutorial introduced you to the tightly integrated design flow between ChipScope Pro ILA Core Inserter and Project Navigator. It showed you how to generate IP core netlist in Project Navigator and synthesize the design. Secondly, it illustrated how to add ChipScope ILA core to the design using the ILA Core Inserter. More importantly, this tutorial guides you through a debugging process. It showed you how to validate and debug your design using ChipScope Pro Analyzer using various triggering setups.

You should now be familiar with some basic design flows and integration between Project Navigator and ChipScope Pro.

Question Answers

1. Briefly describe what you did in Step1.

You just created an RTL PlanAhead project, loaded a VHDL ChipScope™ design, and implemented the design.

2. What are some major circuits used in this lab?

- *debounce.vhdl - Debounce circuit*
- *fsm.vhdl - Control state machine*
- *sinegen_demo - Wrapper for Sine wave generators*

3. Which source file would you have to modify if you were to target other Xilinx® boards?

UCF constraint file

4. What is the main advantage of inserting debug probes onto your post-synthesis netlist instead of adding them onto HDL design files?

You do not have to directly touch and / or modify original HDL source files, thereby eliminating the risk of making any unintentional changes

5. Did you have time to resolve the problem on Step 18 for extra credit?

Hint: Judging by the waveform from Analyzer, it appeared that the output Sine wave got truncated possible due to insufficient bit vector specified in "sinegen_demo.vhd and sinegen.vhd" modules. It is currently specified as a 20-bit vector. It should be expanded to 22-bit. Feel free to modify these two modules and iterate the design.

6. Why is a debounce circuit required for this lab?

It provides a clean pulse or transition from a high to low on this particular example. It eliminates series of spikes or glitches when a button is pressed and released.