PlanAhead Tutorial:

Debugging with ChipScope

UG677 (v 14.5) March 20, 2013

XILINX®

- **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.
-

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/24/2012	14.1	Updates for KC705 board.
07/25/2012	14.2	Editorial updates only; no technical content updates.
10/16/2012	14.3	Editorial updates only; no technical content updates.
12/18/2012	14.4	Editorial updates only; no technical content updates.
03/20/2013	14.5	Editorial updates only; no technical content updates.

Table of Contents

Tutorial: Debugging with ChipScope Pro

Prerequisites

A basic knowledge of Xilinx® ISE® Design Suite tool flows.

Objectives

This tutorial:

- Shows you how to take advantage of enhanced ChipScope™ Pro Analyzer features in the PlanAhead™ design environment that make the debug process faster and more simple.
- Provides specifics on how to use the PlanAhead design environment and the ChipScope Analyzer to debug some common problems in FPGA logic designs.

After completing this tutorial, you will be able to:

- Validate and debug your design using the PlanAhead design environment and the integrated ChipScope Analyzer with an ILA (Integrated Logic Analyzer) core.
- Understand how to create an RTL project, probe your design using an ILA core, and implement the design in the PlanAhead design environment.
- Generate and customize an IP core netlist in the PlanAhead design environment.
- Debug the design using ChipScope Pro Analyzer and iterate the design using the PlanAhead design environment and a Spartan®-6 FPGA SP601 Evaluation Kit Base Board (SP601 Platform). You can also use the SP605, the ML605 or the KC705 board for this tutorial.

Getting Started

Before you start this tutorial, make sure you have and understand the hardware and software components needed to perform the steps. The following subsection lists the requirements.

Set Up Requirements

The following software and hardware are required to follow the steps in this tutorial:

- Xilinx ISE® Design Suite 14.5 (Logic, DSP, Embedded, or System Edition).
- Spartan-6 FPGA SP601 Evaluation Kit Base Board. For a link to information about the SP601 board, see Additional Resources. You can also use the SP605 or ML605 board with this tutorial. See Board Support and Pinout Information for more information.

- USB and power cables that come with the SP601 Evaluation Kit.

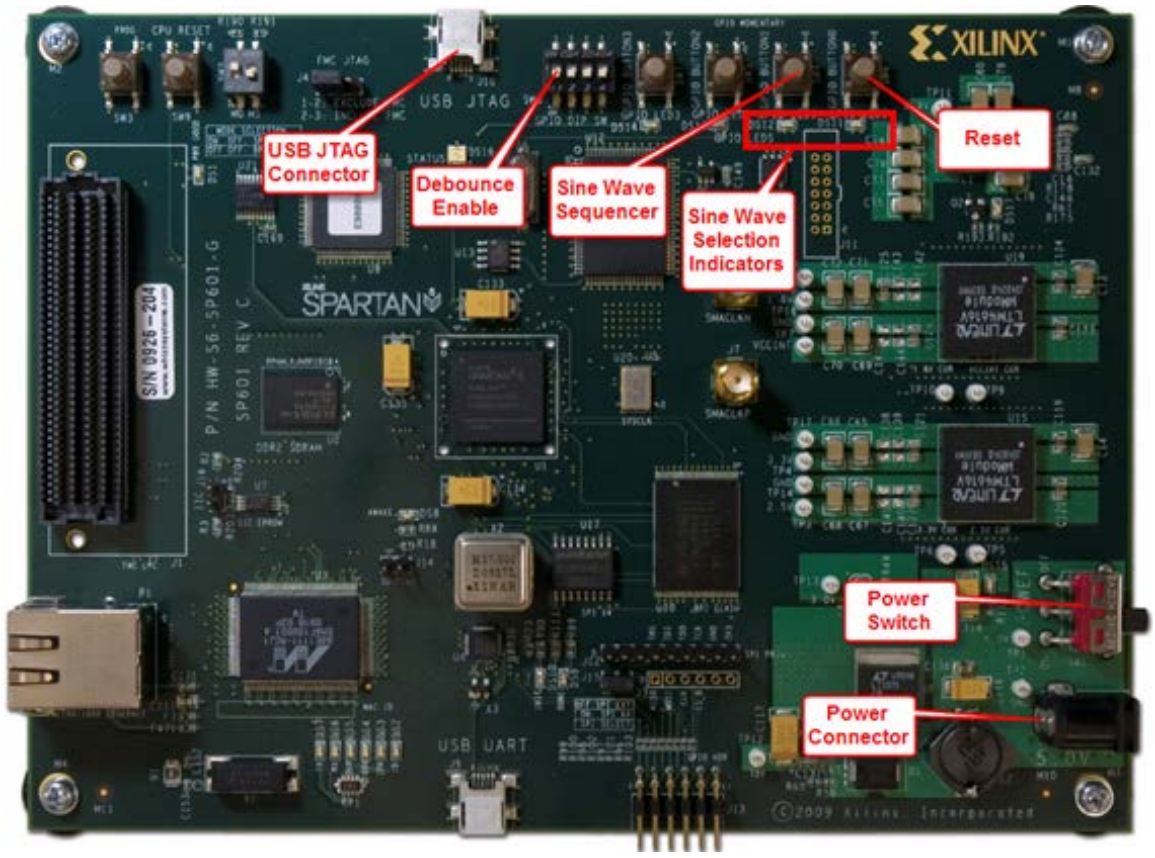

Figure 1: SP601 Board Showing Key Components

Tutorial Design Components

The design includes:

- A simple control state machine
- Multiple sine wave generators
- Common push buttons (GPIO_BUTTON)
- DIP switches (GPIO_SWITCH)
- LED displays (GPIO_LED)
- Push Button Switches: Serve as inputs to the debounce and control state machine circuits. Pushing a button generates a high-to-low transition pulse. Each generated output pulse is then used as an input into the state machine.
- DIP Switch: Enables or disables a debounce circuit.
- Debounce Circuit: In this example, when enabled, provides a clean pulse or transition from high to low. Eliminates a series of spikes or glitches when a button is pressed and released.

- Sine Wave Sequencer State Machine: Captures and decodes input pulses from the two push button switches. Provides sine wave selection and indicator circuits, sequencing between 00, 01, 10, and 11 (zero to three).
- LED Displays: GPIO_LED_0 and GPIO_LED_1 display selection status from the state machine outputs, each of which represents a different sine wave frequency: high, medium, and low.
- Tutorial design files: For details on locating design files, see Getting Started.

Board Support and Pinout Information

> **Note**: This tutorial also supports two other Xilinx platforms: SP605 and ML605. Use the pin-out information in Table 1 to retarget this tutorial to the SP605 or ML605 board.

Table 1: Pinout Information for the SP601, SP605, ML605, or KC705 Board

	Pin-Out Locations				Function
	SP601	**SP605**	**ML605**	**KC705**	
CLK_N	K16	K22	H9	AD11	Clock
CLK_P	K15	K21	J9	AD12	Clock
GPIO_BUTTONS[0]	P4	F3	A19	AA12	Reset
GOIP_BUTTONS[1]	F6	G6	G26	G12	Sine Wave Sequencer
GPIO_SWITCH	D14	C18	D22	AA27	Debounce circuit Selector
LEDs n[0]	E13	D17	AC22	AB8	Sine Wave Selector[0]
LEDs n[1]	C14	AB4	AC24	AA8	Sine Wave Selector[1]
LEDs n[2]	C4	D21	AE22	AC9	Reserved
LEDs n[3]	A4	W15	AE23	AB9	Reserved

Step 1: Creating and Implementing an RTL Project in the PlanAhead Design Environment

To create and implement an RTL project you will:

- Get started by unzipping the tutorial source files and opening the PlanAhead design environment.

- Create a New Project with the New Project Wizard.
- Synthesize the design.

Getting Started

1. In your `C:/drive, create a folder called /ChipScope_PlanAhead`.
2. Find the tutorial design source files.

   This tutorial uses the sample design data included in the supplied example projects in the PlanAhead design environment or in a downloadable compressed ZIP file.

   Design data location:
   `http://www.xilinx.com/support/documentation/dt_planahead_planahead14-5_tutorials.htm`

   The tutorial and design files might be updated or modified in between software releases on the Xilinx website, where you can download the latest version of the materials.

3. Unzip the tutorial source file to the `/ChipScope_PlanAhead` folder.

4. When unzipped, look in `ChipScope_PlanAhead/src` for the files and folder shown in Figure 2.

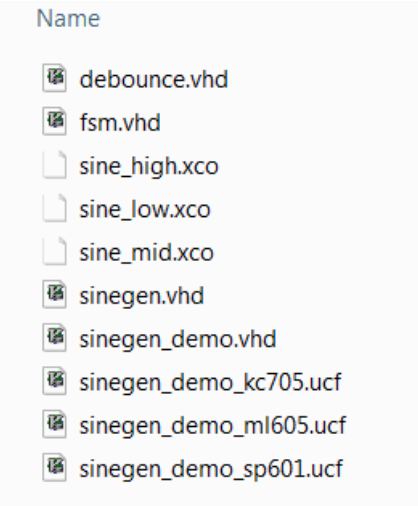

Figure 2: Tutorial Design File Set

Creating a Project with the PlanAhead New Project Wizard

To create a project, you will use the New Project wizard to name the project, to add RTL source files and constraints, and to specify the target device.

1. Invoke the PlanAhead design environment.
2. In the Getting Started screen, click **Create New Project** to start the New Project wizard.

3. In the **Project Name** screen, name the new project pa_step1 and provide the project location (`C:\ChipScope_PlanAhead`).
4. In the **Project Type** dialog box, select **RTL Project**.
5. In the **Add Sources** dialog box:
   a. Click the **Add Files** button.
   b. In the **Add Source Files** dialog box, navigate to the `/src` directory where you saved your project design files.
   c. Select all VHD source files, and click **OK**.
   d. Verify that the files are added, and click **Next**.
6. In the **Add Existing IP** dialog box:
   a. Click the **Add Files** button.
   b. In the **Add Configurable IP** dialog box, navigate to the `/src` directory.
   c. Select all XCO source files, and click **OK**.
   d. Verify that the files are added, and click **Next**.
7. In the **Add Constraints (optional)** dialog box:
   a. Click the **Add Files** button.
   b. In the **Add Constraints File** dialog box, navigate to the `/src` directory.
   c. Select the `sinegen_demo_sp601.ucf` file and click **OK**.
   d. Verify that the file is added and click **Next**.
8. In the **Default Part** dialog box, specify the `xc6slx16csg324-2` part for the SP601 platform. It is easiest to use the search tool just above the parts list to find the correct item.
9. Review the **New Project Summary** screen. Verify that the data appears as expected, per the steps above.

   Note: It might take a moment for the project to initialize.

After you exit the **New Project** wizard, you use the **Project Manager** in the PlanAhead design environment main window to add IP and to synthesize the design.

Synthesizing the Design

1. In the left panel, expand the Synthesis folder, and click the **Run Synthesis** button.

   Note: When synthesis runs, a progress indicator appears, showing that synthesis is occurring. This could take a few minutes.
2. In the **Synthesis Completed** dialog box, click **Cancel**. You will implement the design later.

3. Select **File > Save Project As** and save the project as `pa_step2`. (Saving the project to a new file allows you to more easily resume the tutorial if you do not wish to complete all the steps in one sitting.)

Step 2: Using ChipScope Tools to Debug the PlanAhead Design

To add a ChipScope Analyzer ILA core to the design, you will take advantage of integration flows between the PlanAhead tool and ChipScope Analyzer.

With the simplified workflow, you probe the design without modifying the original RTL source code. You will accomplish the following tasks:

- Add debug nets to the project
- Run the Set Up ChipScope Wizard
- Implement and open the design
- Generate the Bitstream

Adding Debug Nets to the Project

Working in the pa_step2 project:

1. From the Synthesis folder, click **Open Synthesized Design**.
2. Click the **Netlist** tab in the Synthesis pane if it is not already selected.
3. Expand **Nets**. Select the following nets for debugging (also shown in Figure 3):
   - `GPIO_BUTTONS_db(2)`
   - `GPIO_BUTTONS_dly(2)`
   - `GPIO_BUTTONS_re(2)`
   - `sine(20)`
   - `sineSel(2)`
   - `GPIO_BUTTONS_0_IBUF`
   - `GPIO_BUTTONS_1_IBUF`

   Note: These signals represent the significant behavior of this design and will be used to verify and debug the design in subsequent steps.
4. Right-click the selected nets and select **Mark Debug**. If a confirmation dialog box opens, click **OK** to close it.

   Note: With the ChipScope tab selected, you can see the unassigned nets you just selected.

The steps above are illustrated in Figure 3.

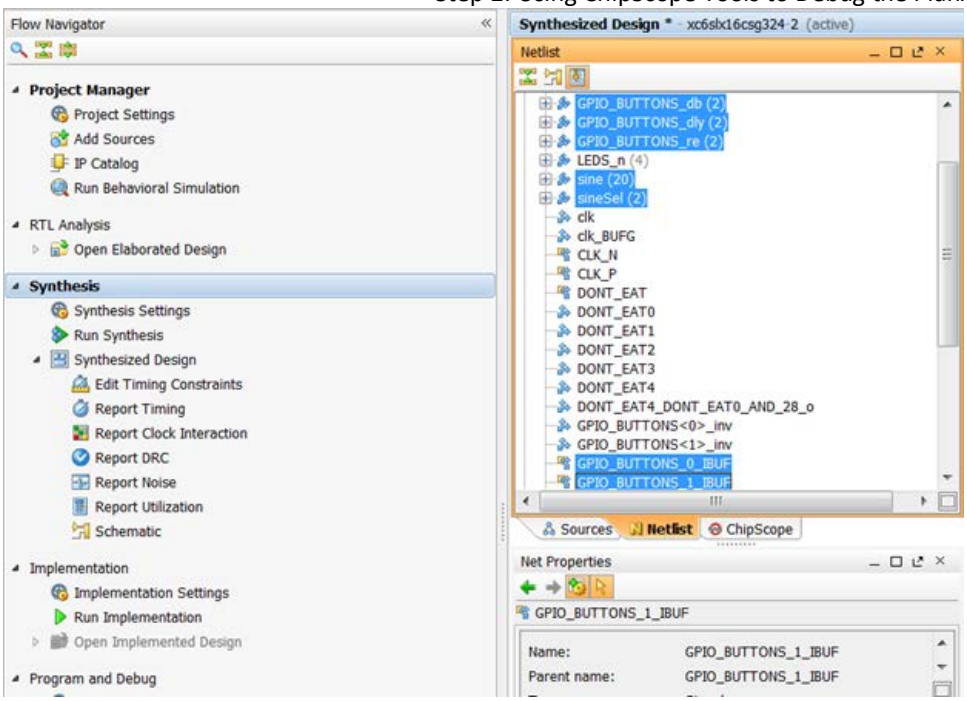

Figure 3: Adding Nets from the Netlist Tab

Running the Set Up ChipScope Wizard

1. With the **ChipScope** tab selected, select all the Unassigned Debug Nets, right click on them, and select **Set Up ChipScope**. The **Set Up Chipscope** wizard opens.
2. Click through the wizard to create ChipScope Analyzer debug cores, keeping the default settings.

Implementing the Design and Generating the Bitstream

1. In the **Flow Navigator** of the PlanAhead tool, open the **Implement** folder, and click on **Run Implementation**.
2. In the Save Project pop-up menu, select **Save**.

When the implementation process ends, an **Implementation Completed** dialog box opens. Click **Cancel**.

Note: Implementation could take a few minutes.

3. Select the **Generate Bitstream** option and click **OK**.
4. In the Generate Bitstream dialog box, click **OK** to start generating the bitstream.

A **Bitstream Generation Completed** pop-up appears to let you know the process is finished. Click **OK**.

If another pop-up appears saying **Do you want to close 'Synthesized Design' before opening 'Implemented Design'** click **OK**.

Step 3: Using ChipScope Tools to Debug the Hardware

In this step, you learn:

- How to debug the design using ChipScope tools.
- How to discover and correct a circuit problem by making a small adjustment to the design.
- Some useful techniques for triggering and capturing design data.

Verifying Operation of the Sine Wave Generator

After doing some setup work, you will use ChipScope Pro Analyzer to verify that the sine wave generator is working correctly. The two primary objectives are to verify that:

- All sine wave selections are correct.
- The selection logic works correctly.

Setting Up

1. Ensure that your SP601 board is correctly set up:
   - Digilent USB JTAG cable should run from the USB JTAG connector on the board to the USB port on your system. (*Do not use the USB UART* connector on the board.)
   - The board is plugged in and powered on.
   - Turn DIP switch positions on SW8 (Debounce Enable) to the OFF position.
2. In the PlanAhead design environment, from the **Program and Debug** drop-down list, select **Launch ChipScope Analyzer**.
3. In **ChipScope Pro Analyzer**, configure the JTAG Chain to the Platform USB JTAG cable and communication parameters:
   a. Select **JTAG Chain > USB Platform Cable**.
   b. The USB Platform JTAG Cable Parameters dialog box opens. Verify that the speed is set to **3 MHz** and Device is connected to **JTAGSmt/SN:xxxxxxxxxxxx**.

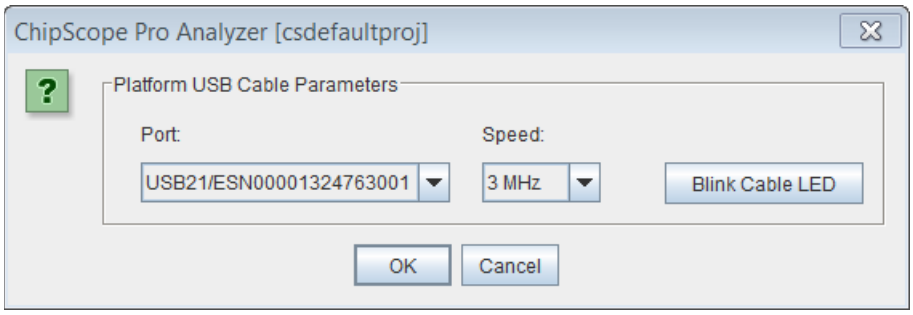

Figure 4: ChipScope Pro Analyzer Settings

4. Confirm the connection to the JTAG chain by verifying the information that appears in the bottom pane of the ChipScope Pro Analyzer window, as illustrated in Figure 4.
5. In the ChipScope Pro Analyzer dialog box, click **OK**.

   Note: The ESN option number varies from what is shown. The number is unique to each board.

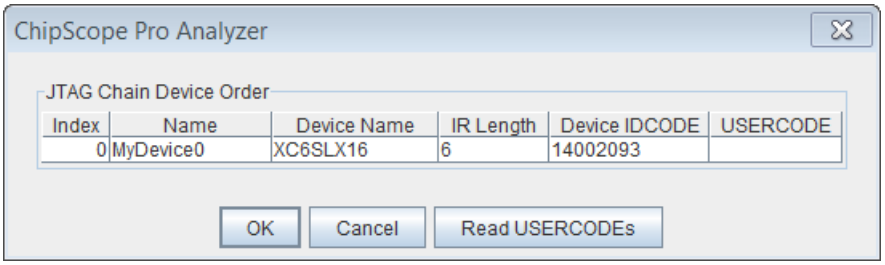

Figure 5: Chipscope Pro Analyzer Dialog Box

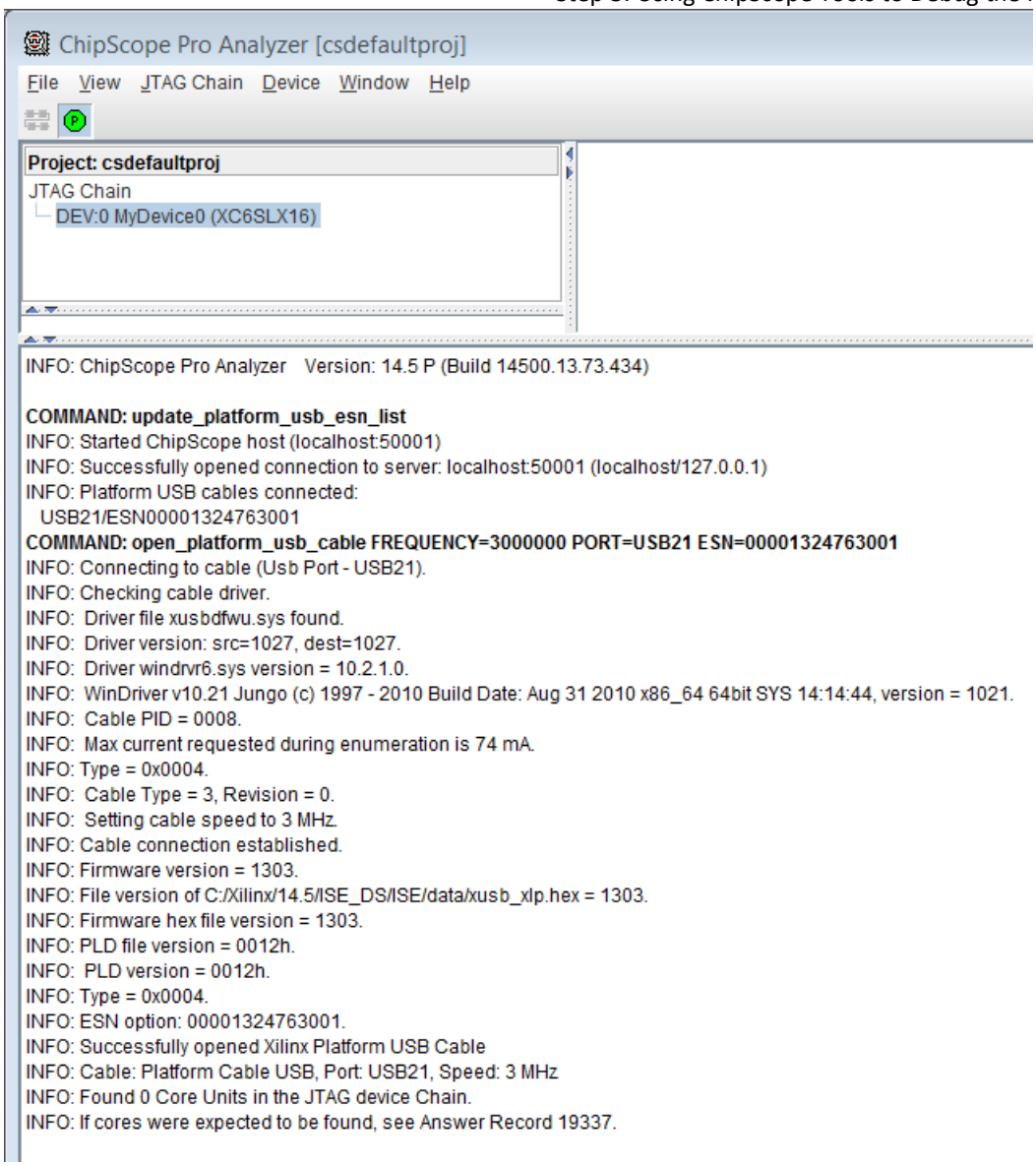

Figure 6: ChipScope Pro Analyzer Window

6. Figure 7: JTAG Chain Connection DataConfigure the device.
   a. In the upper left pane of ChipScope Pro Analyzer, right-click the JTAG Chain device listed under **Project: csdefaultproj** and select **Configure**.

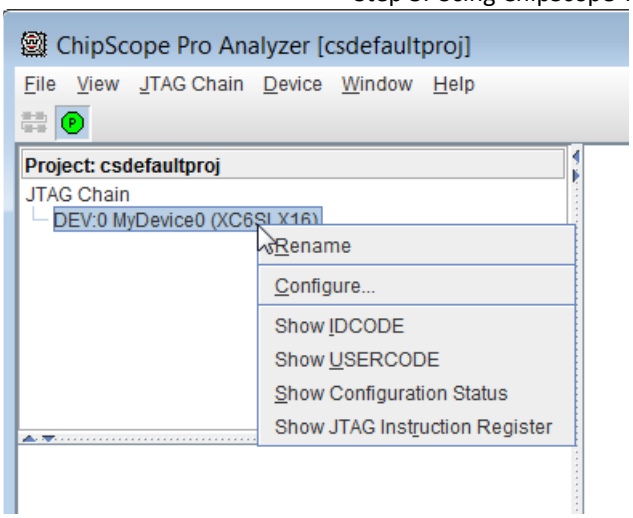

Figure 7: Configuring the Device

The JTAG Configuration dialog box opens. This dialog box lets you program the device with the BIT file you created earlier. The PlanAhead tools provide the location of the BIT and CDC files to the ChipScope Pro Analyzer; consequently, leave all settings at their default values.

b. You can now verify the device configuration and ILA core in the ChipScope Pro Analyzer main window, as shown in Figure 8.

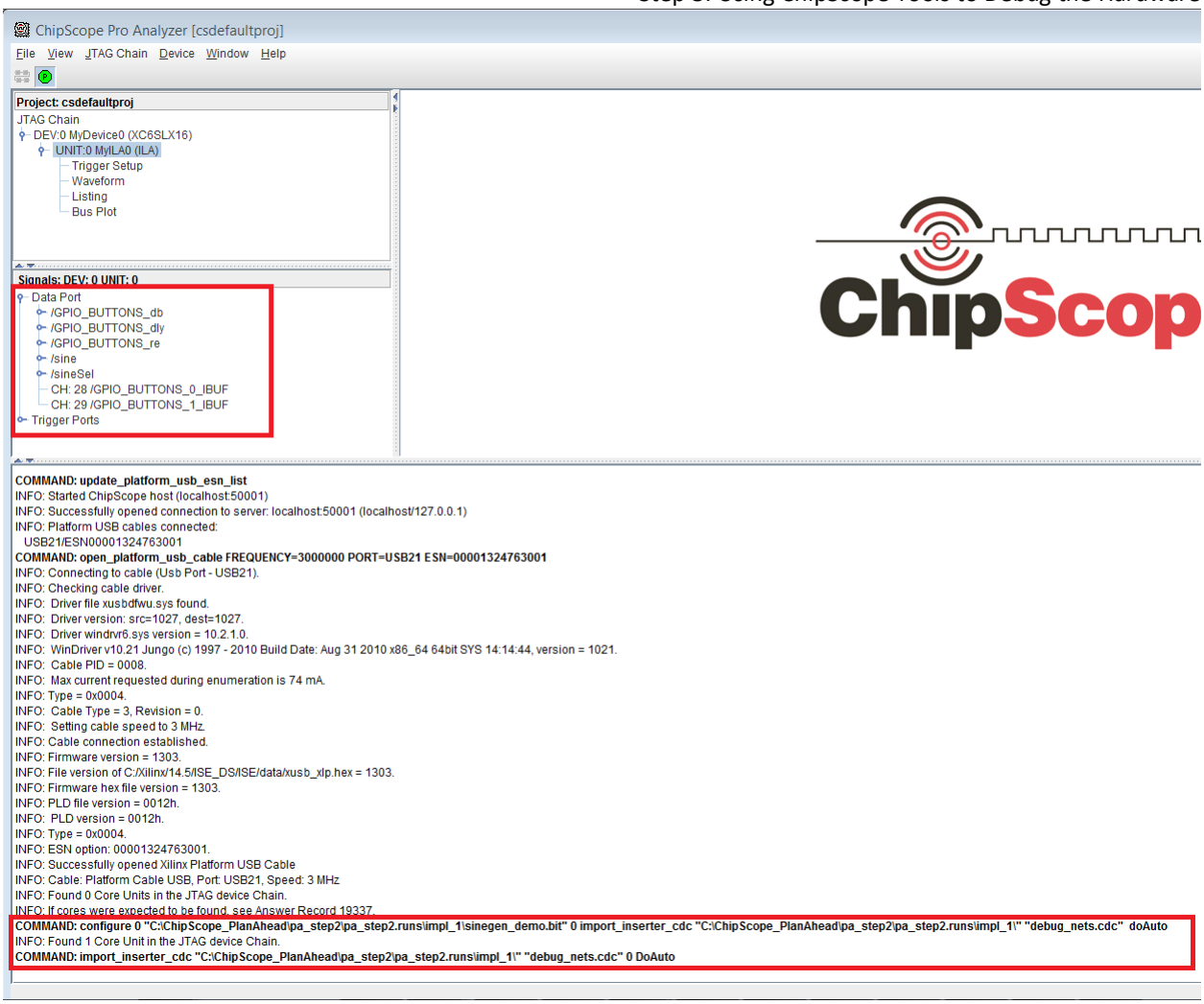

Figure 8: Device Configuration and ILA Core Information

Verifying Sine Wave Activity

1. With the **item DEV:0 MyDevice0 (XC6SLX16)** expanded, as shown in Figure 9, double-click **Trigger Setup**. The Trigger Setup display appears in the upper right pane.
2. Double-click **Waveform** to add the Waveform display to the upper right pane.
3. In the tool bar menu, click **T!** to trigger immediately and capture data.
4. In the Waveform window, verify that there is activity on the sine signal.

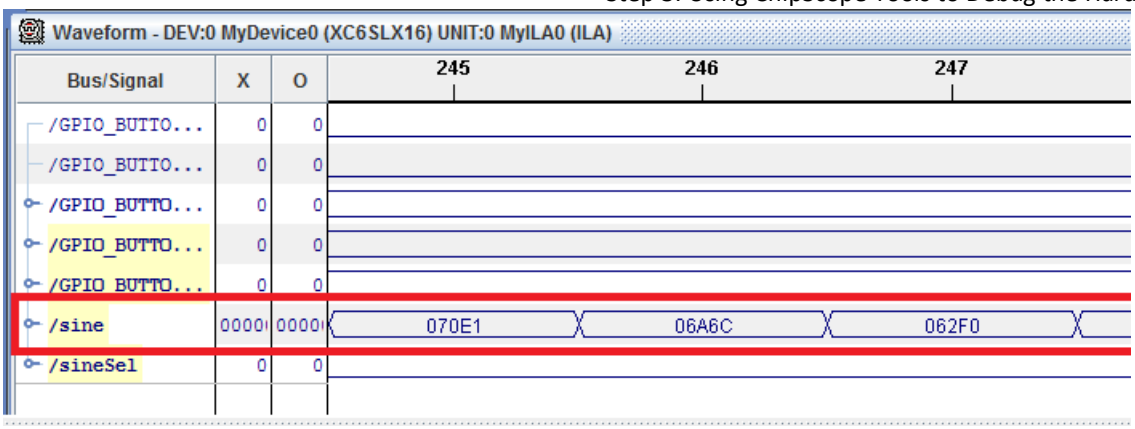

Figure 9: Verifying activity on the sine signal

Displaying the Sine Wave

1. With the item **DEV:0 MyDevice0 (SP601)** expanded, double-click **Bus Plot** to open the Bus Plot window.
2. In the Bus Plot window, select the **/sine** checkbox to display sine wave.

   Notice that the waveform does not look like a sine wave. This is because you must change the radix setting from Hex to Signed Decimal, as described in the following subsection.

Correcting Display of the Sine Wave

To change the radix setting and correct the display, use the Trigger Immediate function

(**T!** button) to view the high, mid, and low frequency sine wave bus plots and change the setting for each.

1. Select the waveform in the project panel.
2. In the Signals window, on the left side of the screen, select **/sine** and right-click to select **Bus Radix > Signed Decimal**. Review the information in the resulting Decimal Values dialog box, and then click **OK** to close it because scaling is not required in this case.

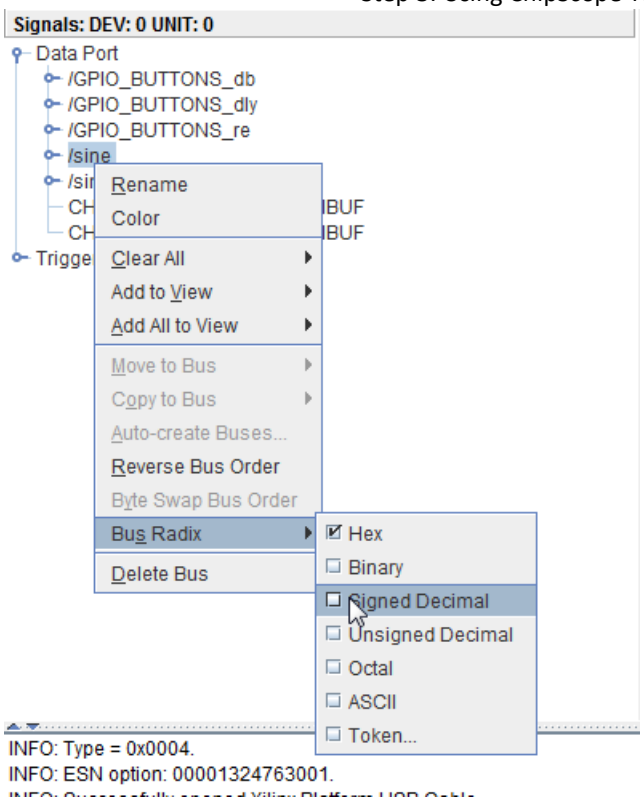

Figure 10: Changing the radix of the sine signal

3. To verify that the sine wave selection state machine is working correctly, perform the steps shown in Table 2, on your SP601 board. Refer to Figure 1, to identify the board components cited in the table.

   Note: As you sequence through the sine wave selections, you might notice that the LEDs do not light up in the expected order. You will debug this in the next section of this tutorial. For now, you will verify, for each LED selection, that the correct sine wave is displayed.

Table 2: Sequencing Through the Sine Wave Selections

Test Setup on the SP601 Board	Trigger and Capture Data in ChipScope Pro Analyzer	Verify Test Results
1. Verify that the sine wave selection indicators (LEDs) are both *off* (0,0). If they are not, push the *Sine Wave Sequencer* button until they are off.	Click **T!** (Trigger Immediately) to view the **high frequency** sine wave bus plot.	Verify that you see the **high frequency** sine wave in the Bus Plot viewer.
2. Push *the Sine Wave Sequencer* button on the board until the two *Sine Wave Selection* indicator	Click **T!** to view the **mid frequency** sine wave bus plot.	Verify that you see the **mid frequency** sine wave in the Bus Plot viewer.

Test Setup on the SP601 Board	Trigger and Capture Data in ChipScope Pro Analyzer	Verify Test Results
LEDs are *off,on* (0,1).		
3. Push the *Sine Wave Sequencer* until the until the two *Sine Wave Selection* indicator LEDs are *on/off* (1,0).	Click **T!** and view the **low frequency** sine wave bus plot.	Verify that you see the **low frequency** sine wave in the Bus Plot viewer.
4. Push the *Sine Wave Sequencer* button on board until Sine Wave Selection indicator LEDs display *on, on* (1,1).	Click **T!** to view the **combined** sine wave bus plot.	Verify that you see the **combined** sine wave in the Bus Plot viewer.

Debugging the Sine Wave Sequencer State Machine

As you were correcting the sine wave display, the LEDs might not have lit up in sequence as you pressed the Sine Wave Sequencer button.

With each push of the button, there should be a single, cycle-wide pulse on the `GPIO_BUTTONS_re[1]` signal. If there is more than one, the behavior of the LEDs becomes irregular. In this section of the tutorial, you will use ChipScope to probe the sine wave sequencer state machine, and view and repair the root cause of the glitch.

Before starting the actual debug process, it is important to understand more about the sine wave sequencer state machine.

Sine Wave Sequencer State Machine Overview

The sine wave sequencer state machine selects one of the four sine waves to be driven onto the sine signal at the top-level of the design. The state machine has one input and one output. Figure 11, shows the schematic elements of the state machine.

Refer to this diagram as you read the following description and as you perform the steps to view and repair the state machine glitch.

- The input is a scalar signal called button. When the button input equals '1', the state machine advances from one state to the next.
- The output is a two-bit signal vector called Y, and it indicates which of the four sine wave generators is selected.

The input signal button connects to the top-level signal `GPIO_BUTTONS_re[1]`, which is a low-to-high transition indicator on the *Sine Wave Sequencer* button (shown in Figure 1). The output signal Y connects to the top-level signal, sineSel, which selects the sine wave.

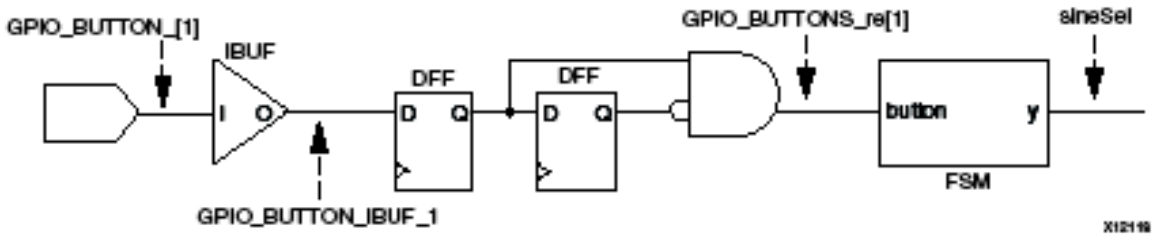

Figure 11: Sine Wave Sequencer Button Schematic

Viewing the State Machine Glitch

Viewing the Input to the State Machine

In this step, you will examine the input to the state machine (`GPIO_BUTTONS_re[1]`), shown in Figure 12) to verify that this signal is causing the glitch. First, you must configure the trigger setup to take a measurement:

1. In the Match area of the Trigger Setup window, locate and expand the **Match Unit** row that contains `GPIO_BUTTONS_re[1]` and `GPIO_BUTTONS_re[0]`.
2. Select the **Value** column in the **GPIO_BUTTONS_re[1]** row, type **R** (rising edge), and press **Enter**. This sets the trigger port match unit to look for a single cycle-wide pulse on the `GPIO_BUTTONS_re[1]` signal.

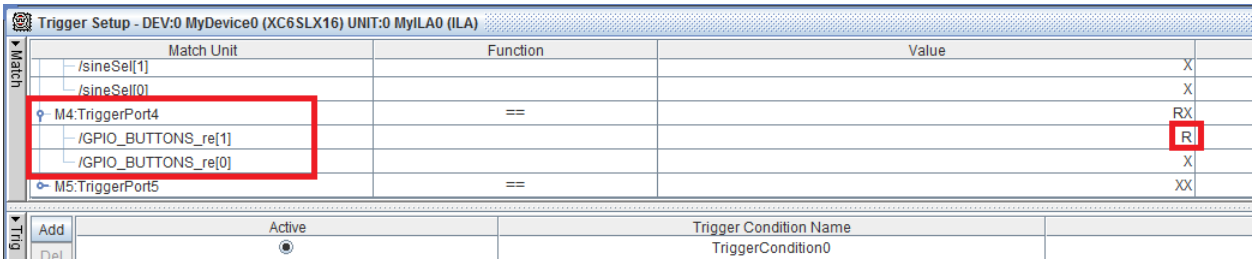

Figure 12: Changing the trigger condition of GPIO_BUTTONS_re[1]

Note: When the `GPIO_BUTTONS_re[1]` signal is assigned a value of '1', the sine wave sequencer state machine transitions from state to state.

3. In the Trigger Conditions area of the Trigger Setup window, click inside the **Trigger Condition Equation** field. In the resulting dialog box, ensure that the appropriate match unit is enabled.

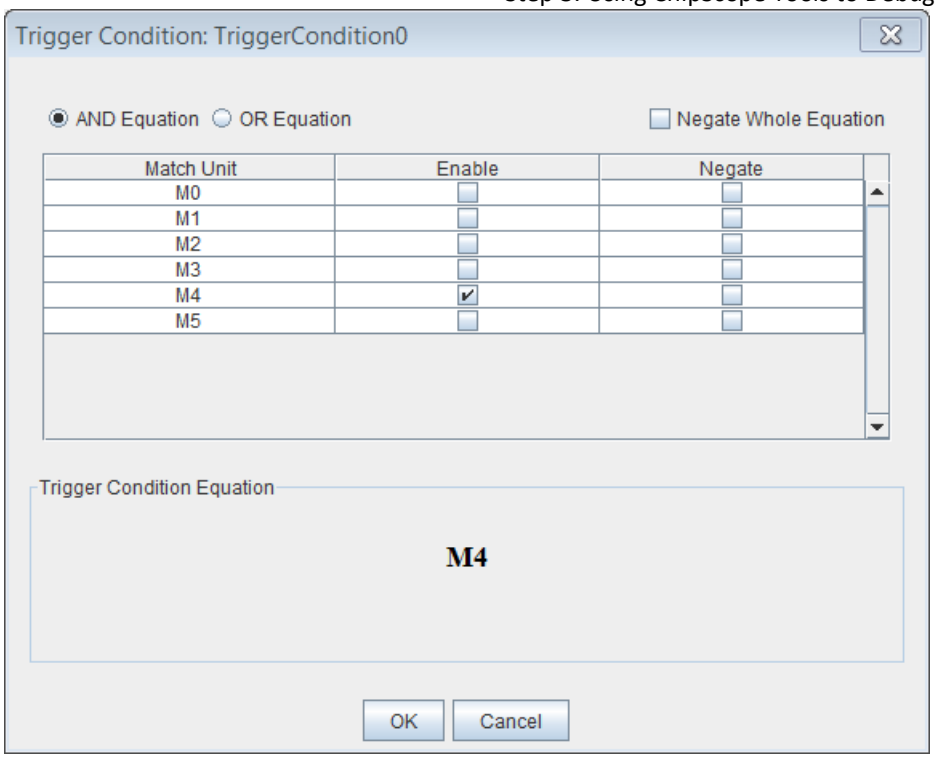

Figure 13: Enabling the correct Match Unit

4. You can tell that the correct match unit is enabled if it is highlighted in blue, as shown in Figure 14.

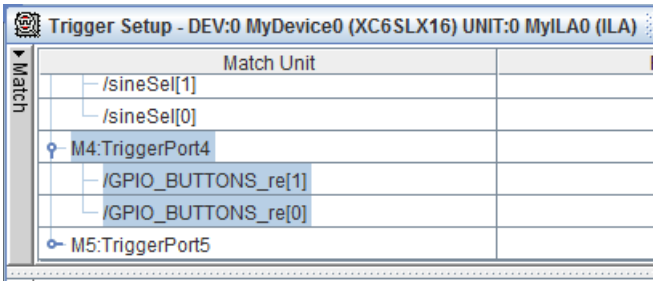

Figure 14: Sample View of Display Setting and State Machine Glitch

5. In the Capture area of the Trigger Setup window, , change the settings as follows:
   - **Windows** = 10
   - **Depth** = 4

   Note: The Depth setting controls the number of samples per window.

Capturing and Viewing the Data

Note the toolbar buttons and names, shown in Figure 15, when following the steps below.

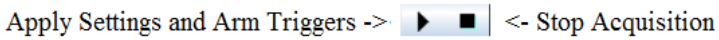

Figure 15: Toolbar Buttons Apply Settings and Arm Trigger, Stop Acquisition

1. In the toolbar, click the **Apply Settings and the Arm Trigger** button.
2. Ensure you have the **Trigger Setup and Waveform** windows displayed.
3. Push the *Sine Wave Sequencer* button on the SP601 board. Each button push generates one or more "windows" that show segments of time on the time axis of the waveform. Observe the number of windows captured in the status bar that appears just below the capture settings. (The status bar is shown in Figure 16.) If more than one window is captured with a single push of the button, you have determined that there is an issue on the `GPIO_BUTTON[1]` input. If only one window is captured, try again. You might have to push the button several times to reproduce this intermittent glitch.

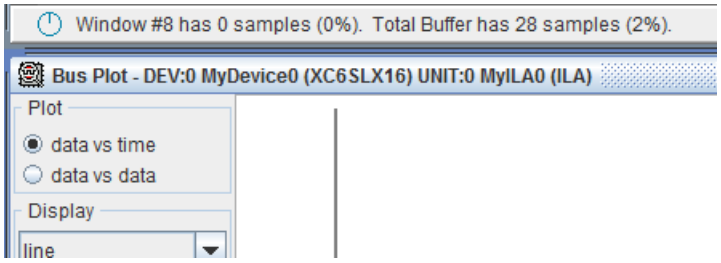

Figure 16: The Status Bar

As soon as a single button push generates multiple windows, go on to the next step.

4. In the toolbar, press the **Stop Acquisition** button and view the captured data.

Notice that each of the multiple windows shows a low-to-high transition on the `GPIO_BUTTON_re[1]` signal. A single button push should only result in a single low-to-high transition. Again, this indicates that something is wrong with the `GPIO_BUTTON[1]` input signal.

Viewing the Button Input to the Design

You cannot troubleshoot the issue you identified above by connecting a debug probe to the `GPIO_BUTTON[1]` input signal itself. The `GPIO_BUTTON[1]` input signal is a PAD signal that is not directly accessible from the FPGA fabric. Instead, you must trigger on low-to-high transitions (rising edges) on the `GPIO_BUTTON_IBUF` signal (shown in Figure 11), which is connected to the output of the input buffer of the `GPIO_BUTTON[1]` input signal.

1. In the **Match** area of the Trigger Setup window, locate and expand the Match Unit row that contains **GPIO_BUTTONS_1_IBUF** and **GPIO_BUTTONS_0_IBUF**.
2. Select the **Value** column in the **GPIO_BUTTONS_1_IBUF** row, type **R**, and press **Enter** to set the trigger port match unit to look for a single cycle-wide pulse on the `GPIO_BUTTONS_IBUF` signal.
3. In the Trigger Conditions area of the Trigger Setup window, click inside the **Trigger Condition Equation** field. In the resulting dialog box, ensure that the appropriate match unit is enabled.

   As described earlier, the glitch reveals itself as multiple low-to-high transitions on the `GPIO_BUTTONS_1_IBUF` signal, but it occurs intermittently. Because it could take several button presses to detect it, you will now set up the ChipScope Pro Analyzer tool to `Repetitive Trigger Run Mode`. This setting makes it easier to repeat the button presses and look for the event in the Waveform viewer.
4. In the toolbar drop-down menu, set the **Trigger Run Mode** to **Repetitive**.
5. In the Capture area of the Trigger Setup window, change the settings as follows:
   - **Windows** = **1**
   - **Depth** = **1024**
   - **Position** = **512**
6. Click the **Apply Settings and Arm Trigger** button.
7. On the board, press the *Sine Wave Sequencer* button until you see multiple transitions on the `GPIO_BUTTONS_1_IBUF` signal (this could take 20 or more tries). This is a visualization of the glitch that occurs on the input. An example of the glitch is shown in Figure 17.

   Note: You might not observe signal glitches at exactly the same location as shown in the figure.

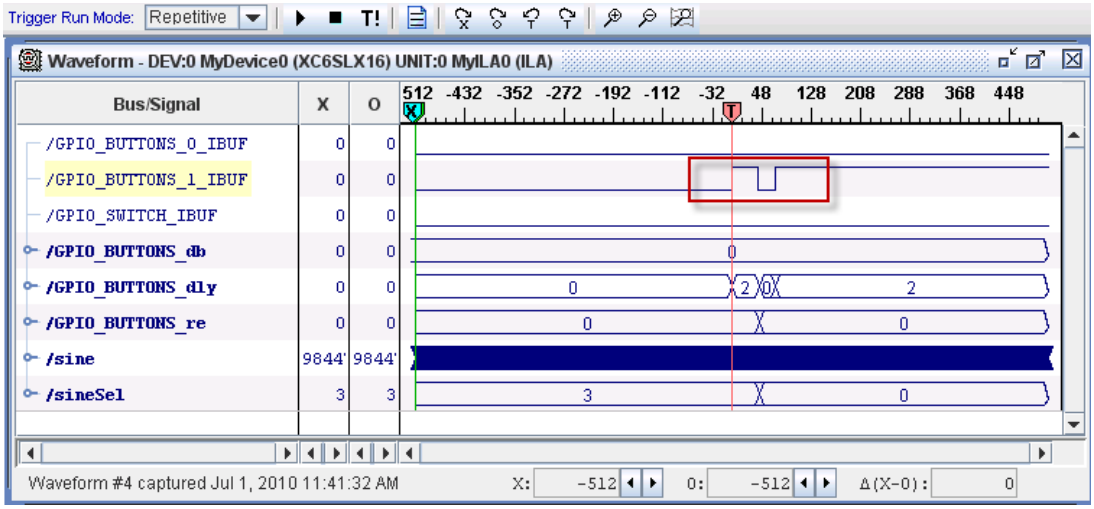

Figure 17: GPIO Buttons_1_IBUF Signal Glitch

Fixing the Signal Glitch and Verifying the Correct State Machine Behavior

The multiple transition glitch or "bounce" occurs because the mechanical button is making and breaking electrical contact just as you press it. To eliminate this signal bounce, a "debouncer" circuit is required.

1. Enable the debouncer circuit by setting *DIP* switch position 1 on the SP601 board (labeled *Debounce Enable* in Figure 1) to the *ON* position.
2. Repeat steps 6 and 7, above, to:
   - Ensure that you no longer see multiple transitions on the `GPIO_BUTTON_re[1]` signal on a single press of the *Sine Wave Sequencer* button.
   - Verify that the state machine is working correctly by ensuring that the *sineSel* signal transitions from 00 to 01 to 10 to 11 and back to 00 with each successive button press.

Additional Resources

Xilinx Resources

- Spartan®-6 PCB Design Guide (UG393): http://www.xilinx.com/support/documentation/user_guides/ug393.pdf
- Xilinx® Glossary: http://www.xilinx.com/company/terms.htm
- Product Support and Documentation: http://www.xilinx.com/support

ChipScope Documentation

- ChipScope™ Pro Software and Cores User Guide (UG029): http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/chipscope_pro_sw_cores_ug029.pdf
- Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Applications (UG750): http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/ug750.pdf

PlanAhead Documentation

- PlanAhead™ User Guide (UG632): http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/PlanAhead_UserGuide.pdf

PlanAhead Tutorials:
http://www.xilinx.com/support/documentation/dt_planahead_planahead14-5_tutorials.htm

- Quick Front-to-Back Flow Overview (UG673): http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/PlanAhead_Tutorial_Quick_Front-to-Back_Overview.pdf

Board Documentation

- Spartan-6 FPGA SP601 Evaluation Kit information: http://www.xilinx.com/products/devkits/EK-S6-SP601-G.htm