PlanAhead Software Tutorial

Debugging with ChipScope

UG677 (v 13.1) March 1, 2011

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Revision
03/01/2011	Updated for a new debug flow in PlanAhead™ software.

Table of Contents

Tutorial: Debugging with ChipScope Pro

Prerequisites

A basic knowledge of Xilinx® ISE® design tool flows.

Objectives

This tutorial:

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

After completing this tutorial, you will be able to:

- Validate and debug your design using PlanAhead and ChipScope tools with an ILA (Integrated Logic Analyzer) core and ChipScope Pro Analyzer.
- Understand how to create an RTL project, probe your design using an ILA core, and implement the design in the PlanAhead software.
- Generate and customize an IP core netlist in the PlanAhead software 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).

Getting Started

Before you start this tutorial, make sure you have and understand the hardware and software components needed to perform the steps. These are described below.

Set Up Requirements

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

- Xilinx ISE® Design Suite 13.1 (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, page 7 for more information.
- USB and power cables that come with the SP601 Evaluation Kit.

Figure 1, below, shows the SP601 board, with key components labeled.

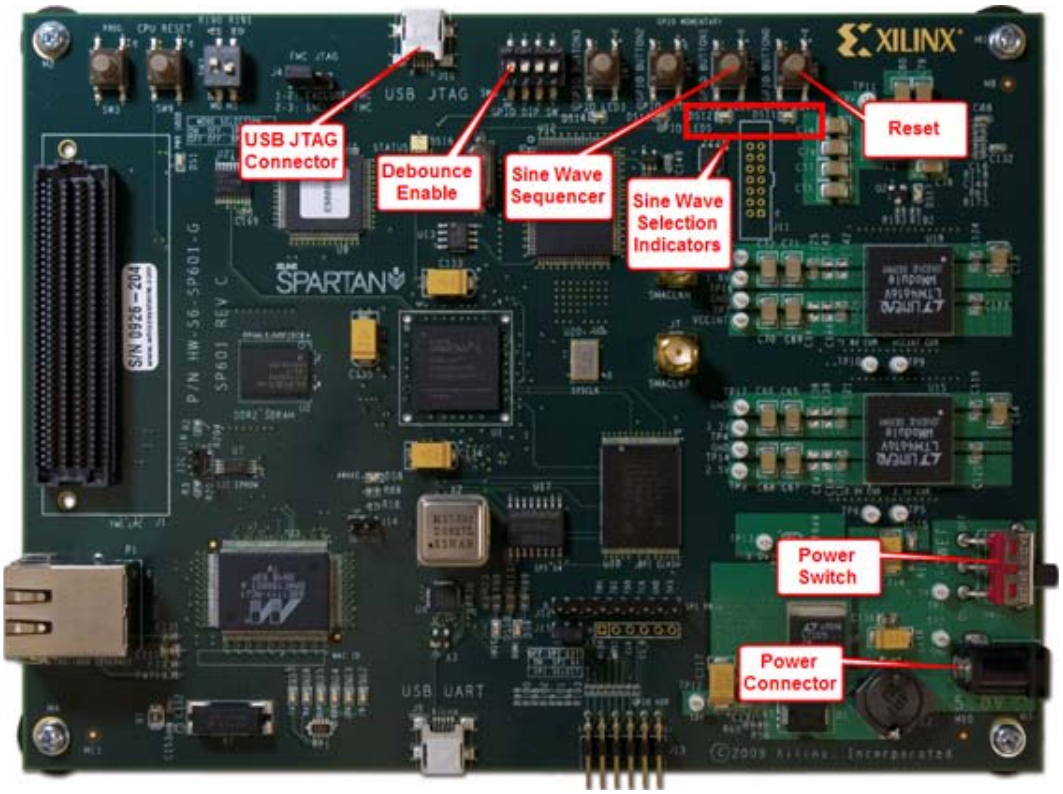

Figure 1: **SP601 Board Showing Key Components**

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_0 display selection status from the state machine outputs, each of which represents a different sine wave frequency: high, medium, and low.
- Tutorial design files: Instructions on locating the design files are provided in Getting Started, page 8.

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 SP605 or ML605 Board**

	Pinout Locations			Function
	SP601	**SP605**	**ML605**	
CLK_N	K16	K22	H9	Clock
CLK_P	K15	K21	J9	Clock
GPIO_BUTTONS[0]	P4	F3	A19	Reset
GPIO_BUTTONS[1]	F6	G6	G26	Sine Wave Sequencer
GPIO_SWITCH	D14	C18	D22	Debounce Circuit Selector
LEDS_n[0]	E13	D17	AC22	Sine Wave Selection[0]
LEDS_n[1]	C14	AB4	AC24	Sine Wave Selection[1]
LEDS_n[2]	C4	D21	AE22	Reserved
LEDS_n[3]	A4	W15	AE23	Reserved

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

To create and implement an RTL project you will:

- Get started by unzipping the tutorial source files and opening the PlanAhead software.
- 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 software or in a downloadable compressed ZIP file.

   Design data locations:

   - `http://www.xilinx.com/support/documentation/dt_planahead_planahead13-1_tutorials.htm`.
   - *`<ISE_install_area>`*`/PlanAhead/testcases/PlanAhead_Tutorial.zip`.

   The tutorial and design files might be updated or modified in between software releases via the web, so 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/PlanAhead_Tutorial/Sources/chipscope/pa_cs_tutorial/src` for the files and folder shown in Figure 2.

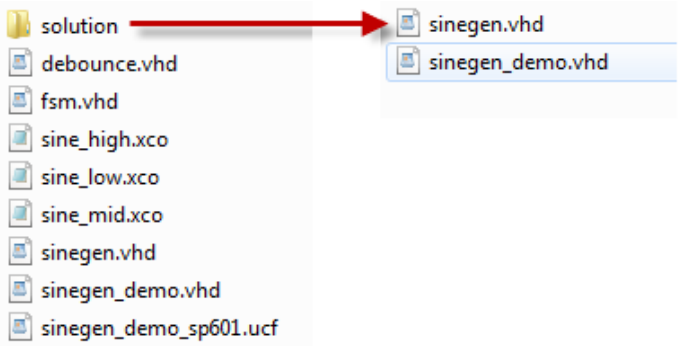

Figure 2: **Tutorial Design File Set**

Creating a Project with the PlanAhead Software 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. Start the PlanAhead software.

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 **Design Source** screen, select **Specify RTL Sources**.
5. In the **Add Sources** screen:
   a. Click the **Add Files** button.
   b. In the **Add Source Files** screen, 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 have been added and click **Next**.
6. In the **Add Existing IP** screen:
   a. Click the **Add Files** button.
   b. In the **Add Configurable IP** screen, navigate to the **src** directory.
   c. Select all **XCO** source files and click **OK**.
   d. Verify that the files have been added and click **Next**.
7. In the **Add Constraints (optional)** screen, the UCF file automatically appears in the main window. Keep the default selection and click **Next** to continue.
8. In the **Default Part** screen, 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 minute or two for the project to initialize.

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

Synthesizing the Design

1. In the left panel, click the **Synthesize** button.
2. In the **Select Top Module** screen, browse to and select **sinegen_demo**.

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

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

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 **Netlist Design** dropdown, click **Open Netlist Design.**
2. Accept the defaults in the **Open Netlist Design** pop-up menu.
3. Click the **Netlist** tab.
4. 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.
5. Right-click the selected nets and select **Add to ChipScope Unassigned nets**.

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

The steps above are illustrated in Figure 3.

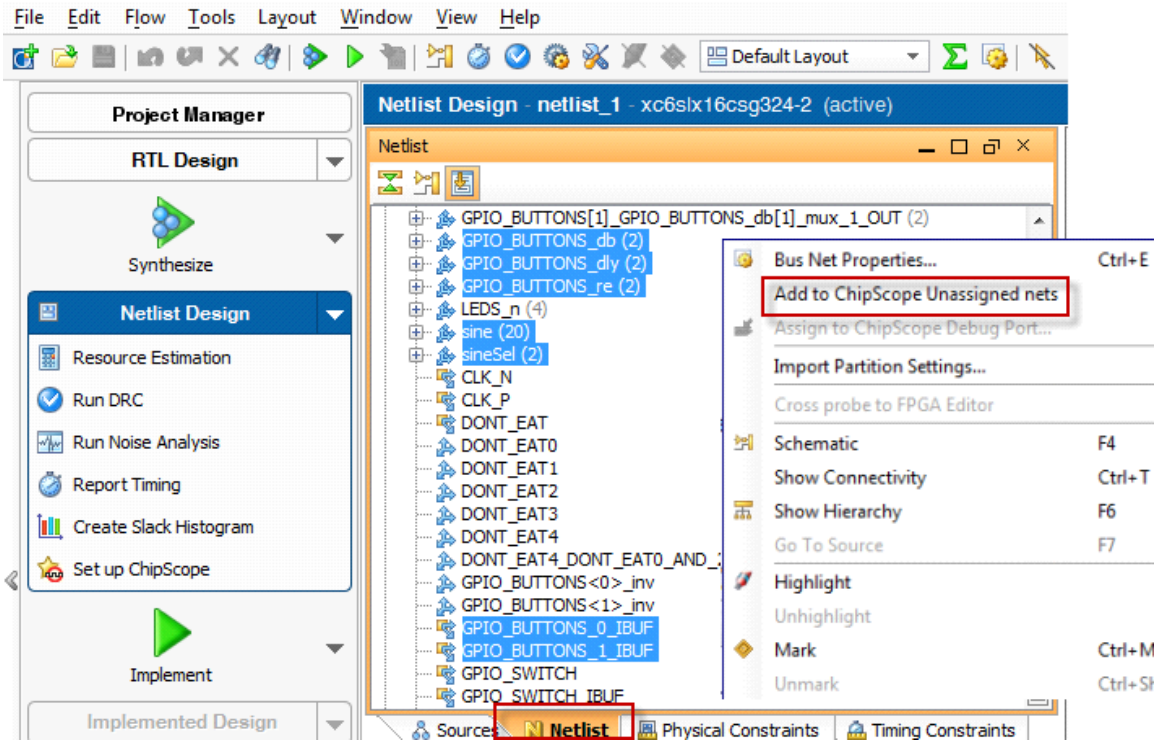

Figure 3: **Adding Nets from the Netlist Tab**

Running the Set Up ChipScope Wizard

1. From the **Netlist Design** dropdown, select **Set Up ChipScope**.
2. The **Set Up ChipScope Wizard** opens. Click through the wizard to create ChipScope Analyzer debug cores. Use the default settings.

Implementing and Opening the Design

1. Click the **Implement** button in the left panel, main window of the PlanAhead software.
2. In the **Save Project** pop-up menu, select **Save**.
3. When the implementation process ends, an **Implementation Completed** dialog box appears. Select the **Open Implemented Design** radio button and click **OK**.

   Note: Implementation could take a few minutes.
4. A prompt appears, asking if you would like to close the netlist design before opening the implemented design. Select **Yes**.

Figure 4 illustrates the steps above.

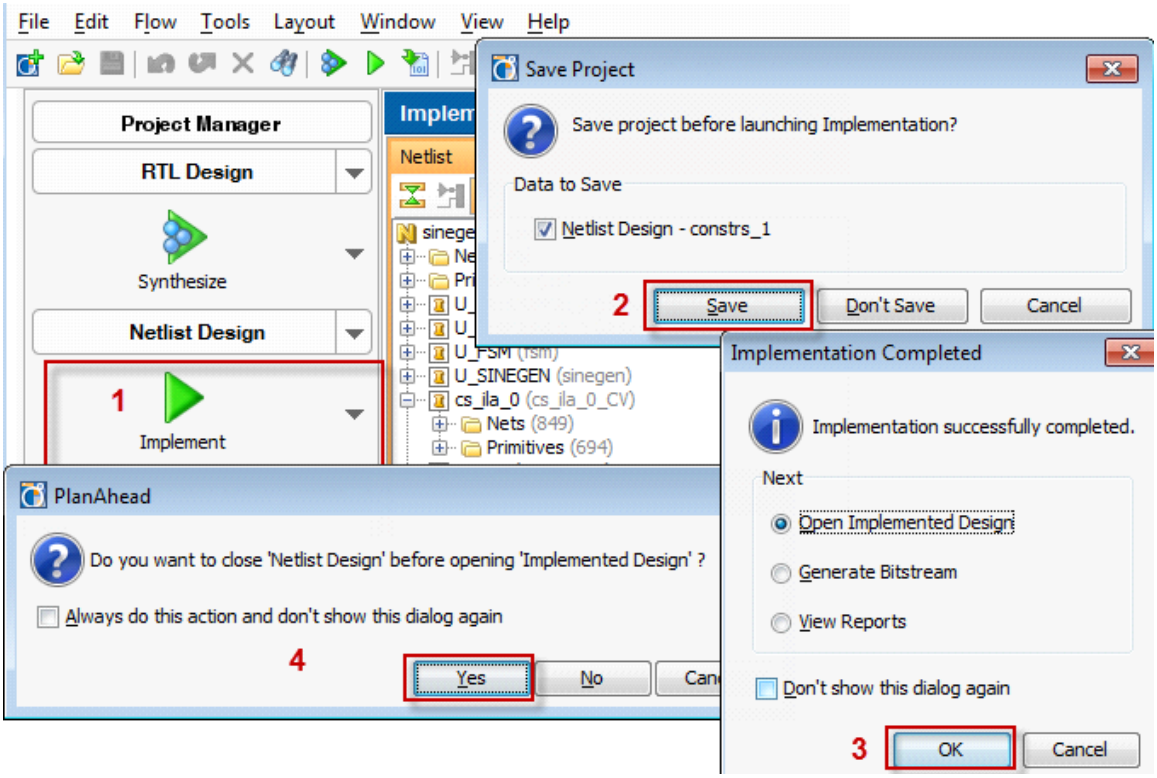

Figure 4: **Steps to Implement and Open the Design**

Generating the Bitstream

1. In the left pane, under **Program and Debug,** click **Generate Bitstream**.
2. The **Generate Bitstream** dialog box appears. Click **OK** to start generating the bitstream.
3. A **Bitstream Generation Completed** pop-up appears to let you know the process is finished.

Step 3: Using ChipScope Tools to Debug the Hardware

Following the steps below, you will:

- Debug the design using ChipScope tools.
- Discover and correct a circuit problem by making a small adjustment to the design.
- Learn 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 will be 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:
   - USB 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.
   - All DIP switch positions are set to the *OFF* position.
2. In the PlanAhead software, from the **Program and Debug** drop-down list, select **ChipScope Analyzer**.
3. In ChipScope Analyzer, configure the JTAG Chain to the USB cable and communication parameters:
   a. Select **JTAG Chain > Xilinx Platform USB Cable**.
   b. The Platform **USB Cable Parameters** dialog box opens. Verify that the speed is set to **3 MHz** and the Port to **USB21**.
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 5.

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

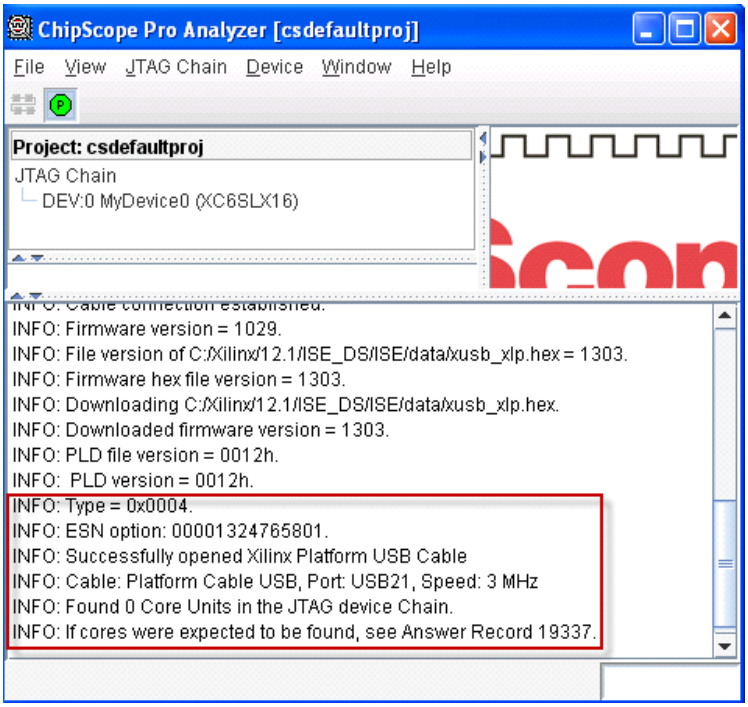

Figure 5: **JTAG Chain Connection Data**

5. Configure the device.
   a. Right-click the JTAG Chain device listed under **Project: csdefaultproj** in the upper left pane of ChipScope Pro Analyzer and select **Configure**.
   b. The JTAG Configuration dialog box appears. This dialog box allows you to program the device with the BIT file you created earlier. PlanAhead tools provided the location of the BIT and CDC files to the ChipScope Pro Analyzer. You should therefore leave all settings at their default values.

c. You can now verify the device configuration and ILA core in the ChipScope Pro Analyzer main window, as shown in Figure 6.

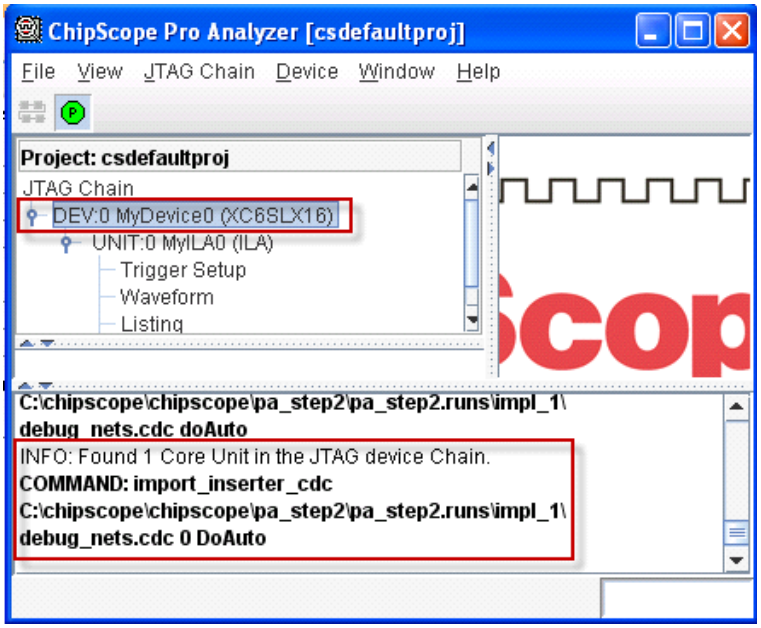

Figure 6: **Device Configuration and ILA Core Information**

Verifying Sine Wave Activity

1. With the item **DEV:0 MyDevice0 (XC6SLX16)** expanded, as shown in Figure 6, 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.

Displaying the Sine Wave

1. With the item **DEV:0 MyDevice0 (XC6SLX16)** 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 much like a sine wave. This is because you must change the radix setting from Hex to Signed Decimal, as described below.

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. In the **Signals** window, on the left side of the screen, select **/sine** and right-click to select **Bus Radix > Signed Decimal**.

2. To verify that the sine wave selection state machine is working correctly, perform the steps shown in Table 2, below, on your SP601 board. Refer to Figure 1, page 6 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 LEDs are ***off,on*** (0,1)**.**	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.
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, view the root cause for the glitch, and repair it.

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 7 shows the schematic elements of the state machine. You will probably find it helpful to refer to this diagram as you read the description below 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, page 6). The output signal `Y` connects to the top-level signal `sineSel`, which selects the sine wave.

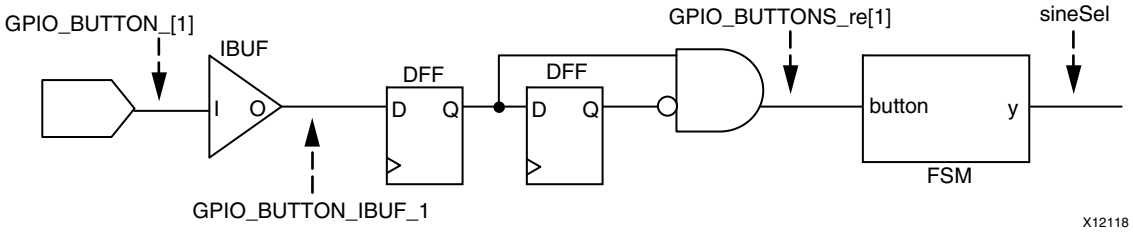

Figure 7: **Sine Wave Sequencer Button Schematic**

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 7) 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.

   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. You can tell that the correct match unit is enabled if it is highlighted in blue, as shown in Figure 8.

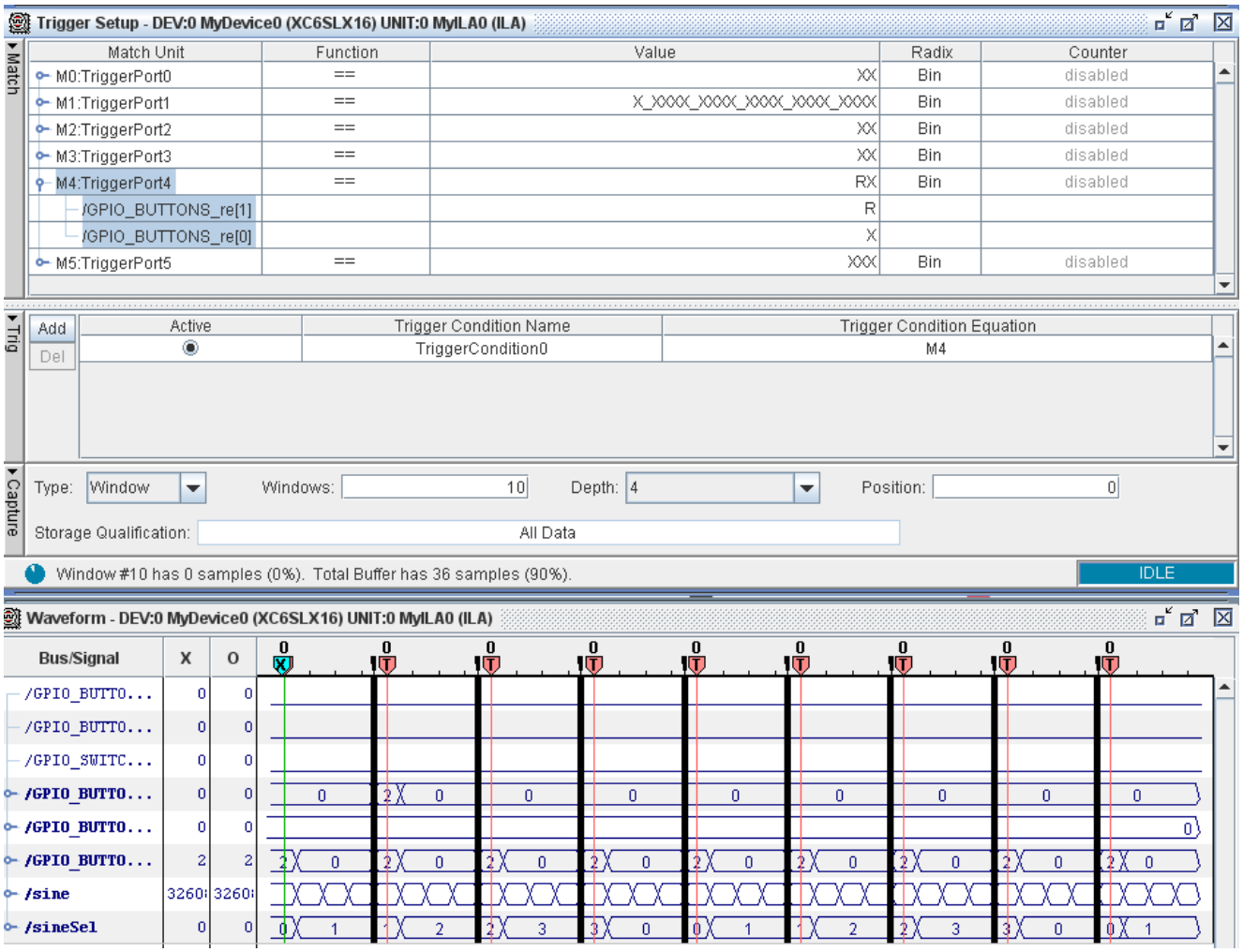

Figure 8: **Sample View of Display Setting and State Machine Glitch**

4. In the **Capture** area of the **Trigger Setup** window, shown in Figure 8, 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 9 when following the steps below.

Figure 9: **Toolbar Buttons: Apply Settings and Arm Trigger, Stop Acquisition**

1. In the toolbar, click the **Apply Settings and Arm Trigger** button.
2. Be sure you have the **Trigger Setup** and **Waveform** windows displayed.
3. Push the *Sine Wave Sequencer* button on the SP601 board. Observe the number of windows being captured in the status bar that appears just below the capture settings. (The status bar is shown in Figure 8.) If more than one window is captured with a single push of the button, you have determined that there is a problem 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.

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

Unfortunately, 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_1` signal (shown in Figure 7, page 16), 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_IBUF_1** and **GPIO_BUTTONS_IBUF_1**.
2. Select the **Value** column in the **GPIO_BUTTONS_IBUF_1** 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_1` 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_IBUF_1` 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 is a visualization of the glitch that is occurring on the input. An example of the glitch is shown in Figure 10.

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

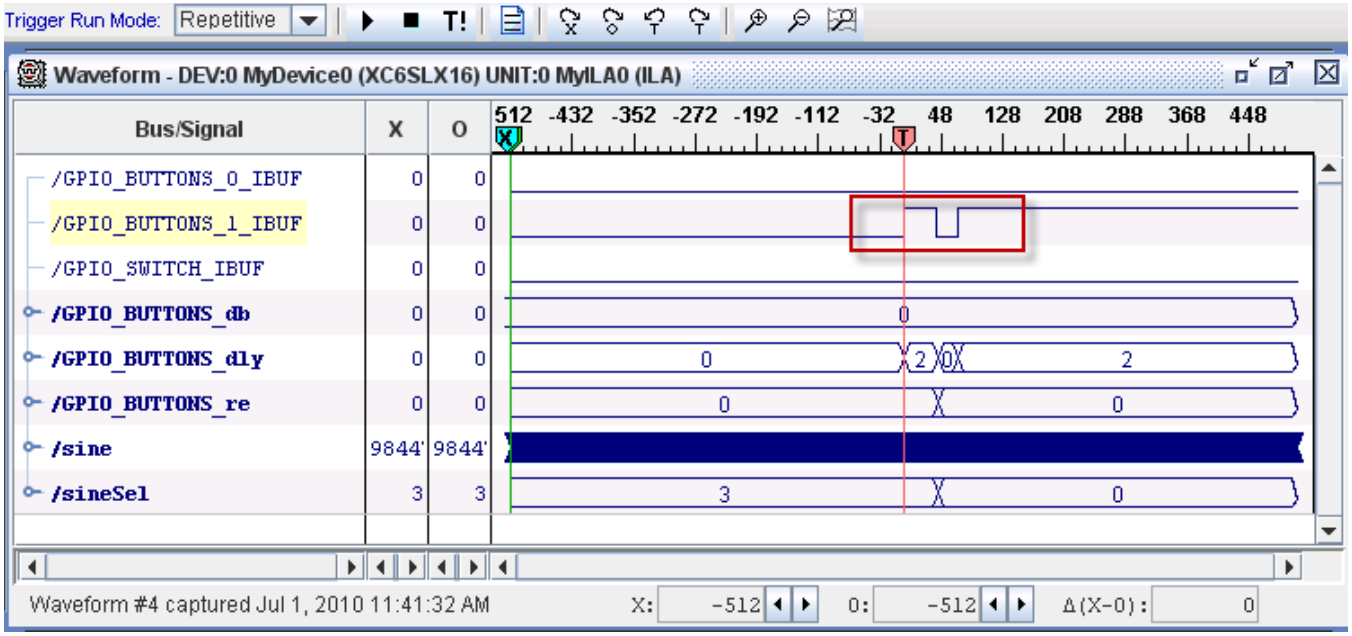

Figure 10: **GPIO Buttons_1_re Signal Glitch**

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, page 6) 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

- *ISE® Design Suite: Installation and Licensing Guide (UG798)*: http://www.xilinx.com/support/documentation/dt_ise13-1_releasenotes_knownissues.htm
- *ISE Design Suite 13: Release Notes Guide (UG631):* http://www.xilinx.com/support/documentation/dt_ise13-1_releasenotes_knownissues.htm
- *Spartan®-6 PCB Design Guide (UG393)*: http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug393.pdf
- **Xilinx® Documentation**: http://www.xilinx.com/support/documentation
- **Xilinx Global Glossary**: http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- **Xilinx Support**: http://www.xilinx.com/support

ChipScope Documentation

- *ChipScope™ Pro Software and Cores User Guide (UG029)*: http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/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/xilinx13_1/ug750.pdf

PlanAhead Documentation

- *PlanAhead™ User Guide (UG632):* http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/PlanAhead_UserGuide.pdf
- *Quick Front-to-Back Flow Overview (UG673)*: http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/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