PlanAhead Software Tutorial

Debugging with ChipScope

UG 677 (v 12.3) September 21, 2010

The PlanAhead™ software source code includes the source code for the following programs:

Centerpoint XML

- The initial developer of the original code is CenterPoint – Connective Software
- Software Engineering GmbH. portions created by CenterPoint – Connective Software
- Software Engineering GmbH. are Copyright© 1998-2000 CenterPoint - Connective Software Engineering GmbH. All Rights Reserved. Source code for CenterPoint is available at http://www.cpointc.com/XML/

NLView Schematic Engine

- Copyright© Concept Engineering.

Static Timing Engine by Parallax Software Inc.

- Copyright© Parallax Software Inc.

Java Two Standard Edition

- Includes portions of software from RSA Security, Inc. and some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/
- Powered By JIDE – http://www.jidesoft.com

Free IP Core License

This is the Entire License for all of our Free IP Cores.

Table of Contents

PlanAhead Software Tutorial

Debugging with ChipScope

Introduction

This tutorial:

- Shows you how to use the Xilinx® PlanAhead™ software to debug designs using the ChipScope™ debugging tool.
- Provides a quick walk through using PlanAhead to insert ChipScope debug cores into your designs.
- Guides you through the selection of nets for debug, and automates the generation of debug cores, instantiation, connectivity, and synthesis operations with the CORE Generator™ tool.
- Shows you how to implement a design with ChipScope cores, import placement and timing reports, run BitGen, and launch ChipScope Analyzer.
- Covers only a subset of the features contained in the PlanAhead software bundled with ISE® Design Suite Release 12.

Other tutorials cover many other PlanAhead features in detail, and not every command or command option is covered. The tutorial uses the features contained in the PlanAhead software product, which is bundled as a part of the ISE Design Suite. If you have any questions or comments about this tutorial, contact Xilinx Technical Support.

Sample Design Data

This tutorial uses sample design data included with the PlanAhead software. The design data is located in the following directory:

`<ISE_install_Dir>/PlanAhead/testcases/PlanAhead_Tutorial.zip`

Save and extract the zip file into any write-accessible location. The location of the unzipped `PlanAhead_Tutorial` data is referred to as the `<Extract_Dir>` throughout this document.

The tutorial sample design data is modified while performing this tutorial. A new copy of the original `PlanAhead_Tutorial` data is required each time you run the tutorial. For more information about the example design, see the *Tutorial Description* section.

Xilinx ISE Design Suite and PlanAhead Software

PlanAhead software is installed with ISE Design Suite by default. Before beginning this tutorial, ensure that PlanAhead is operational, and that the sample design data has been installed. For installation instructions and information, see the *ISE Design Suite 12: Installation, Licensing, and Release Notes* on the Xilinx website:

http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/irn.pdf

Required Hardware

Xilinx recommends 2 GB or more of RAM for use with PlanAhead on larger devices. For this tutorial, a smaller design was used, with a limited number of designs open at any one time. 1 GB of RAM should be sufficient, but it could impact performance.

PlanAhead Documentation and Information

For information about the PlanAhead software, see the following documents, which are available with your software:

- *PlanAhead User Guide* (UG632) - Provides detailed information about the PlanAhead software. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/PlanAhead_UserGuide.pdf
- *Floorplanning Methodology Guide* (UG633) - Provides floorplanning hints. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/Floorplanning_Methodology_Guide.pdf
- *Hierarchical Design Methodology Guide* (UG748) - Provides an overview of the PlanAhead hierarchical design capabilities. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/Hierarchical_Design_Methodology_Guide.pdf
- For additional information about PlanAhead, including video demonstrations, go to http://www.xilinx.com/planahead.

Tutorial Description

The sample design used in this tutorial consists of a typical system on a chip design with a RISC CPU core connected to several peripheral cores using a Wishbone bus arbiter. There are Verilog and VHDL source files.

The design targets an xc6vlx75Tff784 device. This tutorial uses a project file that has already synthesized the HDL and is ready to be implemented.

The tutorial design data is modified during the course of this tutorial. Xilinx recommends that you use a new copy of the original PlanAhead_Tutorial data each time the tutorial is performed. For more information on the tutorial data, see the *Sample Design Data* section above.

If you have any questions or comments about the tutorial, contact Xilinx Technical Support.

Tutorial Objectives

After completing this tutorial, you will have:

- Inserted ChipScope debug cores into designs using the Set Up ChipScope wizard
- Added additional debug nets and re-created ChipScope debug cores
- Changed default options on ChipScope debug cores
- Implemented ChipScope debug cores
- Implemented designs with ChipScope debug cores in the PlanAhead software

Tutorial Steps

This tutorial is separated into steps, followed by general instructions and supplementary detailed steps allowing you to make choices based on your skill level as you progress through the tutorial.

If you need help completing a general instruction, go to the detailed steps below it, or if you are ready, simply skip the step-by-step directions and move on to the next general instruction.

This tutorial has seven primary steps:

Step 1: Open a project

Step 2: Run the Set up ChipScope Wizard

Step 3: Add Additional Debug Nets

Step 4: Change Debug Core Attributes.

Step 5 Implement ChipScope Debug Cores

Step 6: Implement the Design

Step 7: Generate a Bitstream - Launch ChipScope Analyzer (optional)

Step 1: Open a Project Step 1

PlanAhead enables several types of projects to be created depending on the location in the design flow where the software is being used. RTL sources or synthesized netlists can be used to create a Project for development, analysis, or to take all the way through implementation and bit file creation. This tutorial uses a synthesized netlist project which has not yet implemented.

1-1. Open the software.

1-1-1. On Windows, double-click the Xilinx PlanAhead 12 desktop icon, or select **Start > Programs > Xilinx ISE Design Suite 12.3 > PlanAhead > PlanAhead**.

1-1-2. On Linux, change the directory to `<Extract_Dir>/PlanAhead_Tutorial/Tutorial_Created_Data`, and type **`planAhead`**.

The PlanAhead Getting Started Help page opens.

1-2. Open the example project_cpu_netlist Project.

The example design Project netlist is modified during this tutorial. You will open the example design and then save the project to a different name so the original example design project can be re-used.

1-2-1. In the Getting Started page, select **Open Project** and browse to select:

`<Extract_Dir>/PlanAhead_Tutorial/Projects/project_cpu_netlist/project_cpu_netlist.ppr`

1-2-2. Click **Open**.

PlanAhead opens with the selected project design.

1-2-3. In the Flow Navigator on the left side of the PlanAhead environment, select **Netlist Design**.

The Netlist Design is now open and ready to insert debug logic (Figure 1).

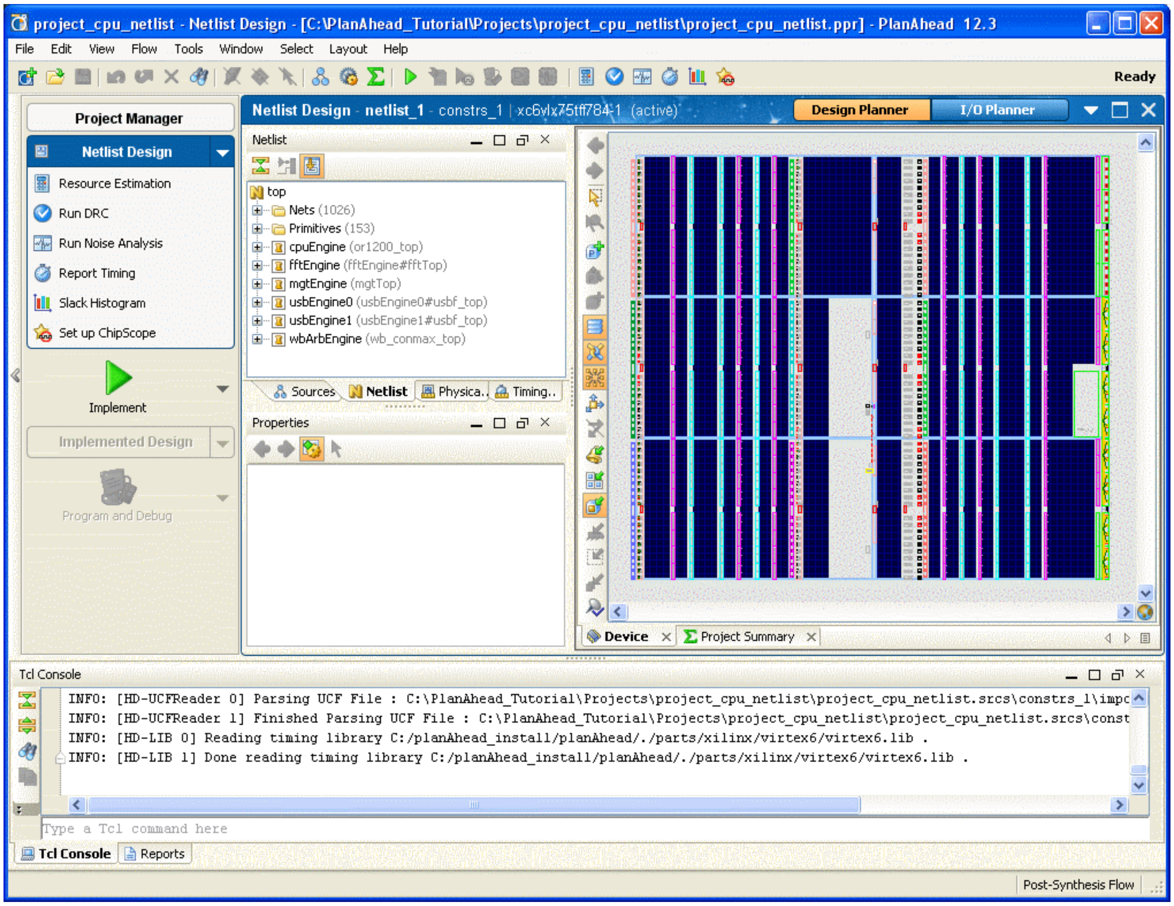

Figure 1: The Project in the Netlist Design Environment

Step 2: Run the Set Up ChipScope Wizard Step 2

Use the ChipScope debug core insertion flow to configure and instantiate the Integrated Controller (ICON) and Integrated Logic Analyzer (ILA) cores into a synthesized netlist project.

You can use the Set Up ChipScope wizard to:

- Select nets for debugging
- Determine how many cores are needed
- Instantiate and connect the core trigger ports to these nets

2-1. Select the wbArbEngine/m0/wb* nets to connect to ChipScope cores.

2-1-1. In the Flow Navigator menu, select **Set Up ChipScope**.

The Set up ChipScope wizard opens.

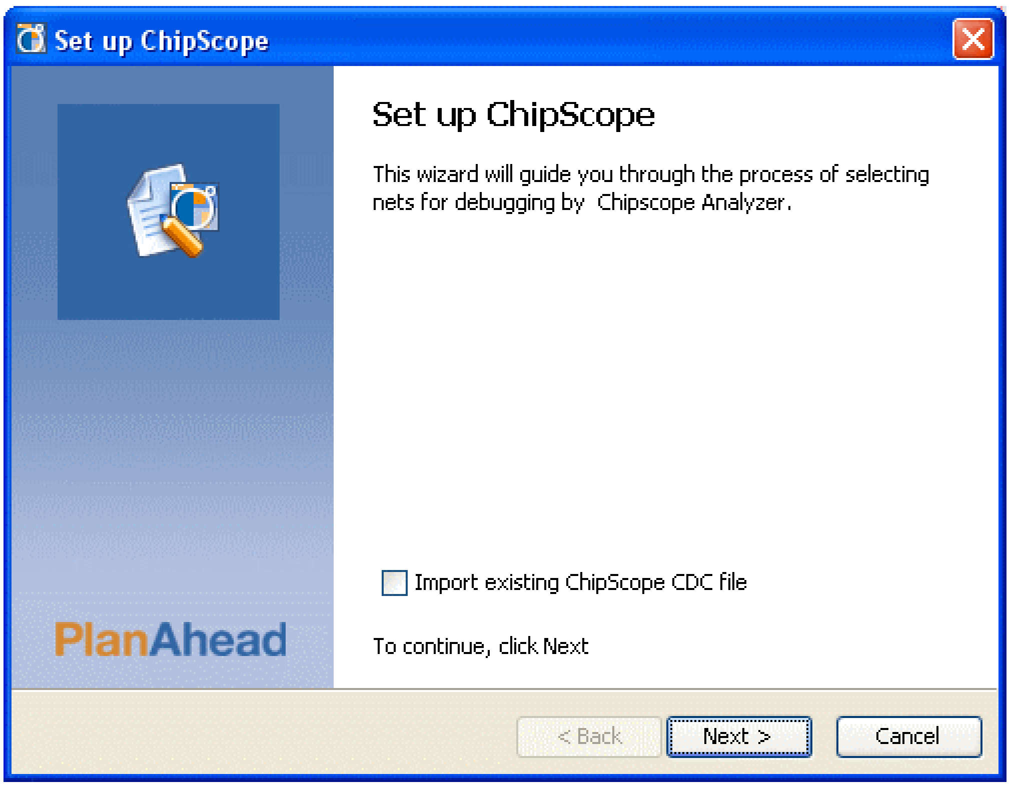

Figure 2: Set Up ChipScope

2-1-2. Click **Next**.

The Specify Nets to Debug page displays.

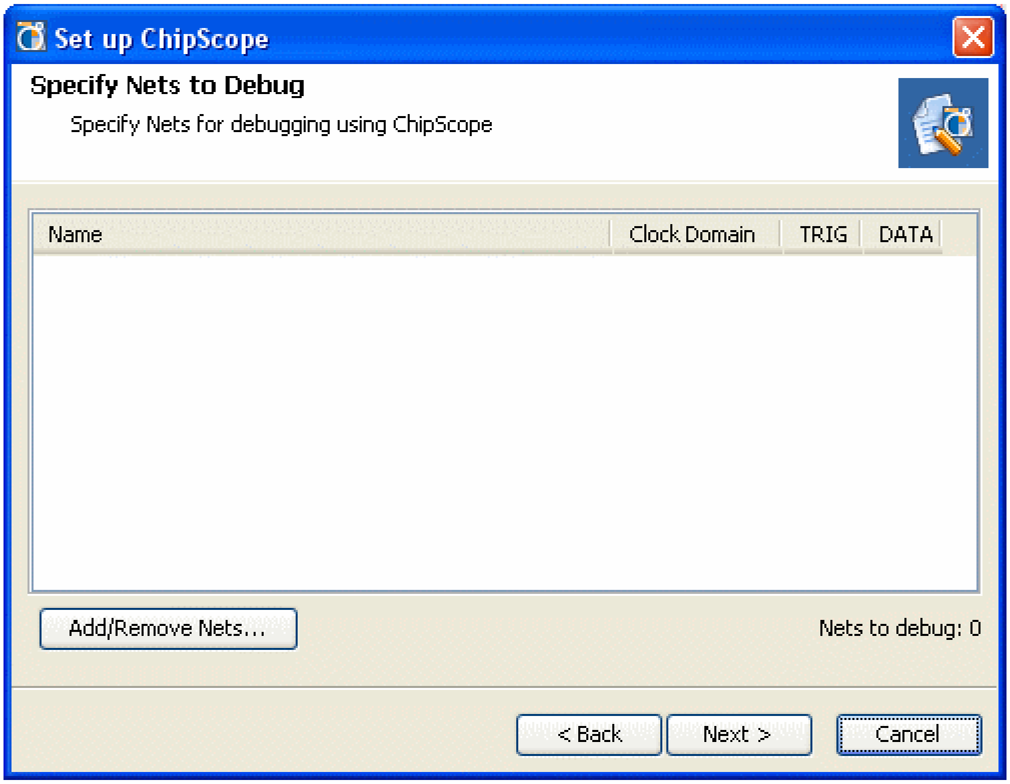

Figure 3: Specify Nets to Debug Page

2-1-3. Click **Add/Remove Nets**.

The Add/Remove Nets dialog box opens (Figure 4).

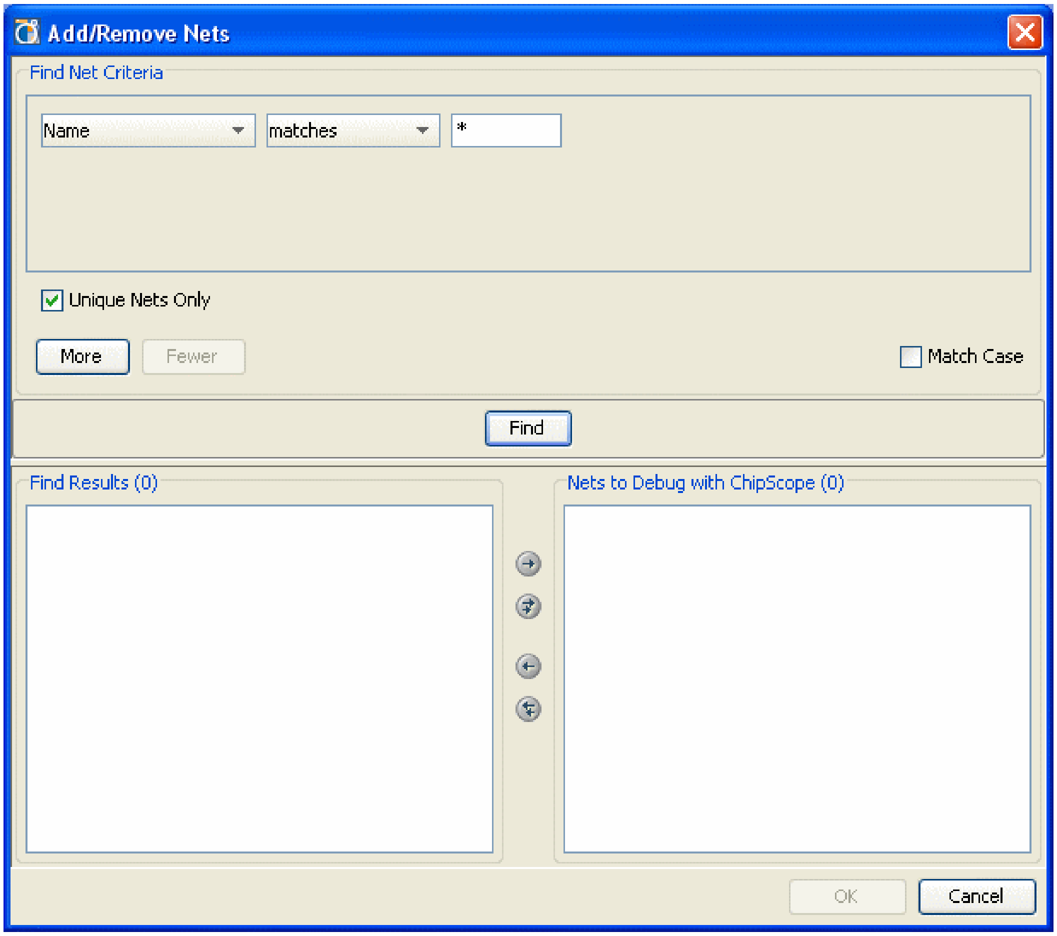

Figure 4: Add/Remove Nets Dialog Box

2-1-4. In the Find Net Criteria Matches text box, type **`wbArbEngine/m0/wb_*`**.

2-1-5. Click **Find**.

The Add/Remove Nets dialog box opens with the Find Results box populated (Figure 5).

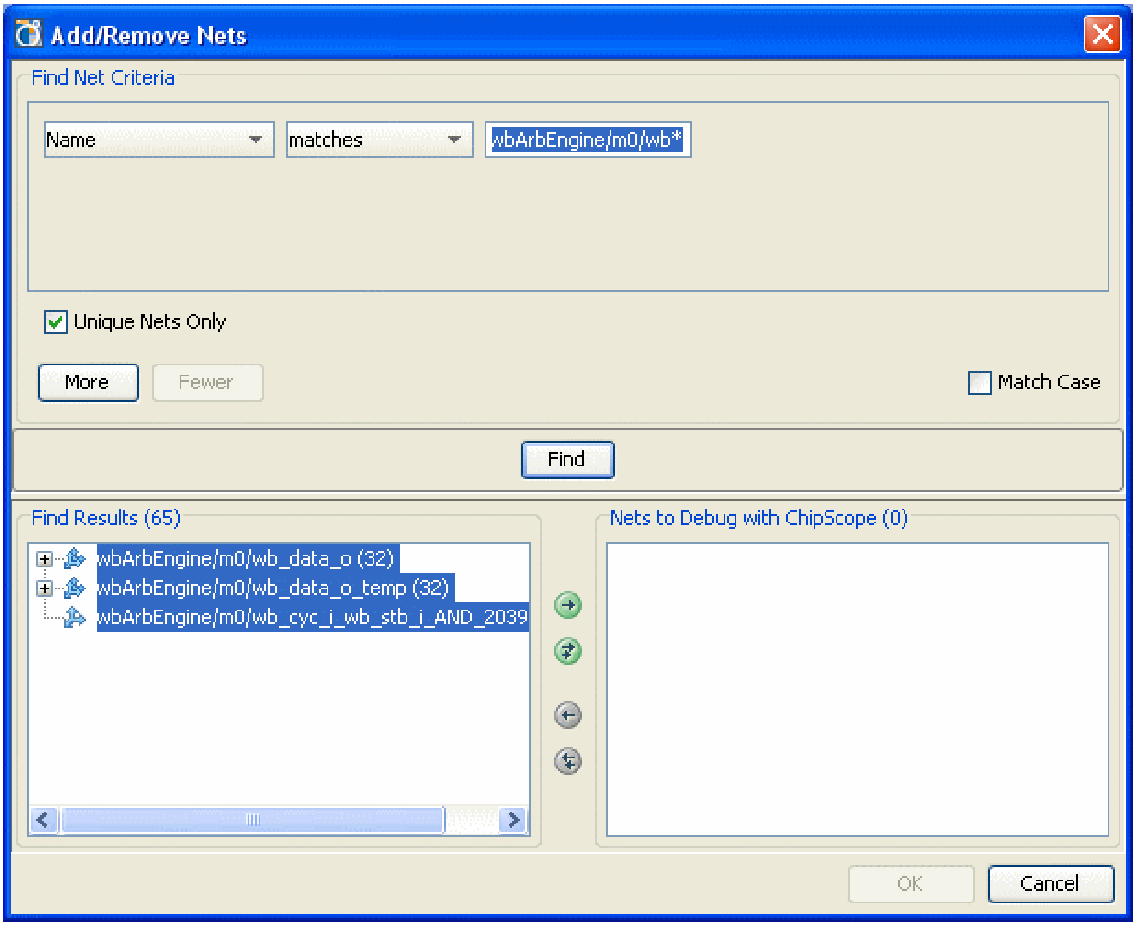

Figure 5: Add/Remove Nets Dialog Box with Find Results Populated

2-1-6. Click the Move all Nets to the Right  button to add all nets to the Nets to Debug with ChipScope list.

The Add/Remove Nets dialog box now has the Nets to Debug with ChipScope list populated.

2-1-7. Click **OK**.

The Specify Nets to Debug pane of the Set Up ChipScope wizard opens (Figure 6).

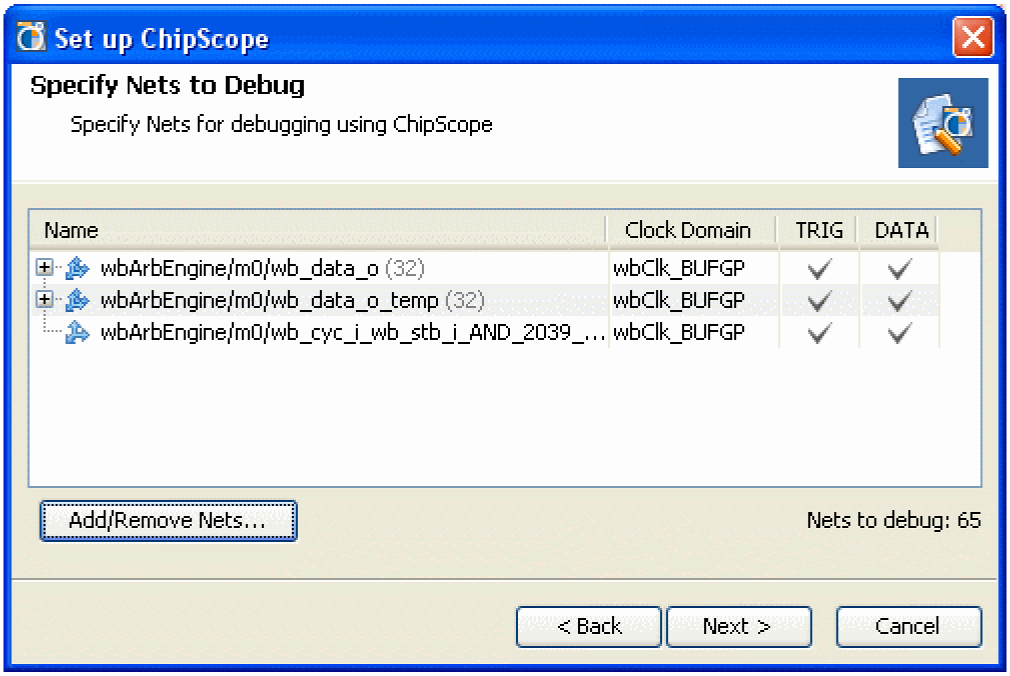

Figure 6: Set Up ChipScope Wizard with Nets to Debug Specified

Notice that the nets are all on the same Clock Domain, so only one ILA core should be required.

2-1-8. Click **Next**.

The Summary page displays.

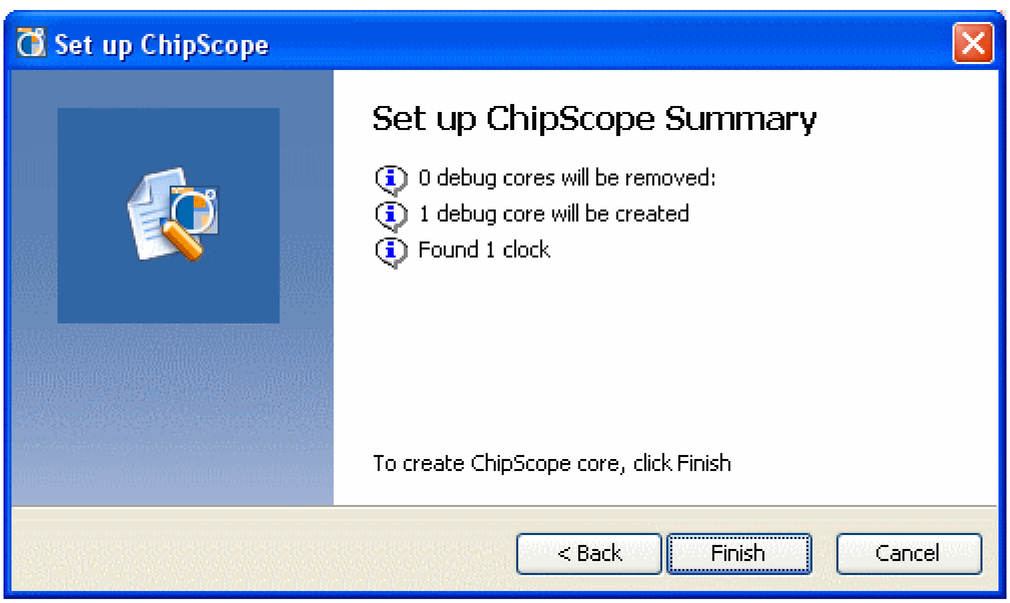

Figure 7: ChipScope Summary Page

2-1-9. Click **Finish**.

This returns you to the Netlist Design environment, with the ChipScope view displayed.

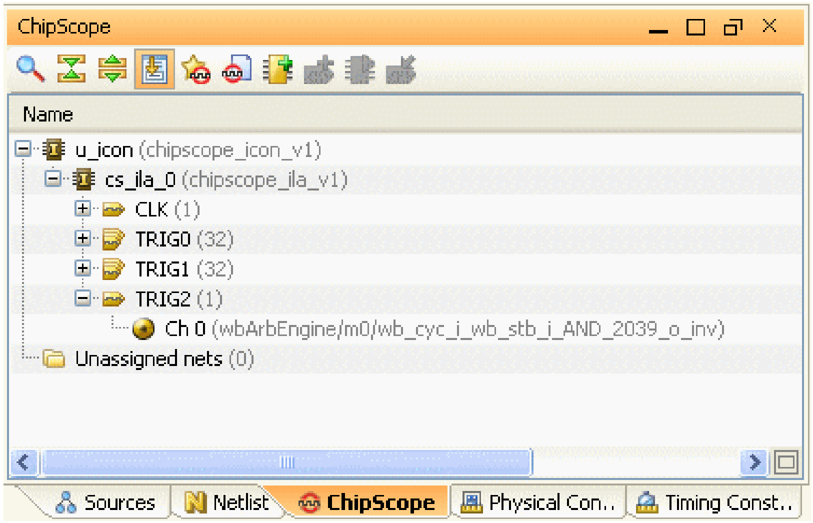

Figure 8: ChipScope View with ChipScope Cores Inserted

The ChipScope view contains information about the generated cores as well as any Unassigned Nets. Core configuration and management is performed in this view. You use this view in some of the later steps of this tutorial.

2-2. View the ILA and ICON debug cores in the Netlist view.

2-2-1. Select the **Netlist** tab.

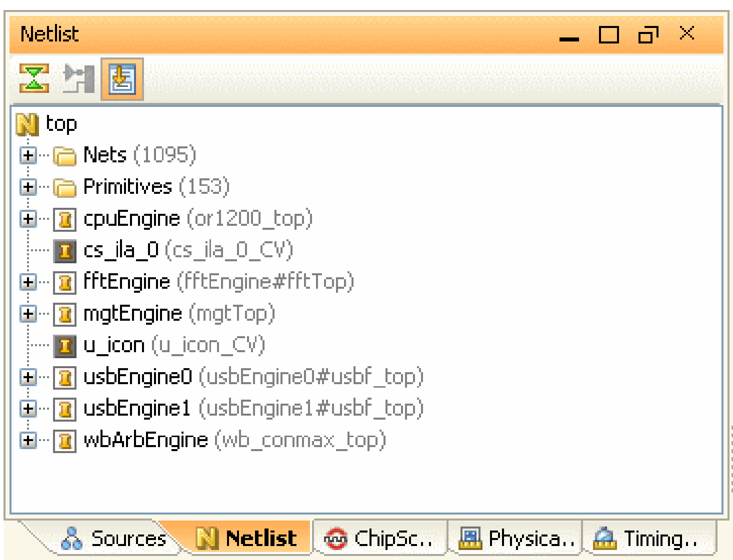

Figure 9: Debug Core Black Box Instance in Netlist Design

The `u_icon` and `cs_ila_0` are new instances inserted into the netlist. Both are marked with black box icons in the Netlist view.

Step 3: Add Additional Debug Nets Step 3

You have now added a set of nets connected to ChipScope debug cores and are ready to implement your design. This portion of the tutorial demonstrates alternate methods to add additional nets to be debugged and re-generate the cores. If you were satisfied with the list of nets already being debugged, this step is not necessary.

3-1. Use the Set Up ChipScope wizard to re-configure the Debug cores.

3-1-1. Click the ⊞ icon next to the `cpuEngine` hierarchy in the **Netlist** view to expand the hierarchy below the CPU module.

3-1-2. Click the ⊞ icon next to the `iwb_biu` hierarchy below the `cpuEngine` module to expand the hierarchy below the wishbone bus for the instruction fetch module of the CPU.

3-1-3. Click the ⊞ icon next to the `/Nets` folder under the `iwb_biu` hierarchy to expand the list of nets within the wishbone interface.

3-1-4. Select the `wb_adr_o` bus under the `/Nets` folder to select all 32 bits of the wishbone address output vector.

The Netlist view of the PlanAhead environment should look like it does in Figure 10 below.

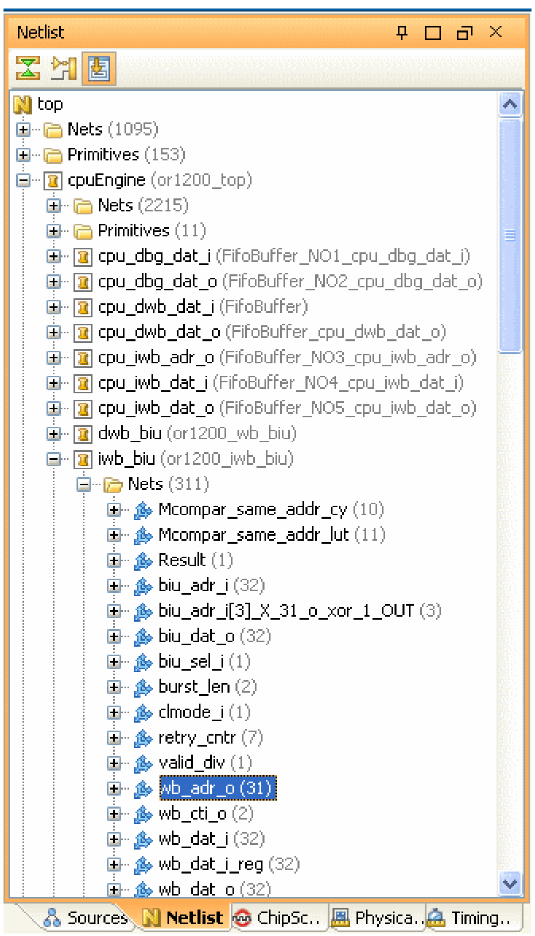

Figure 10: Netlist View with wb_adr_o Bus Selected

3-1-5. In the Flow Navigator, click **Set up ChipScope** to restart the Set Up ChipScope wizard.

The Set up ChipScope wizard opens.

3-1-6. Click **Next**.

The Existing Debug Nets page displays.

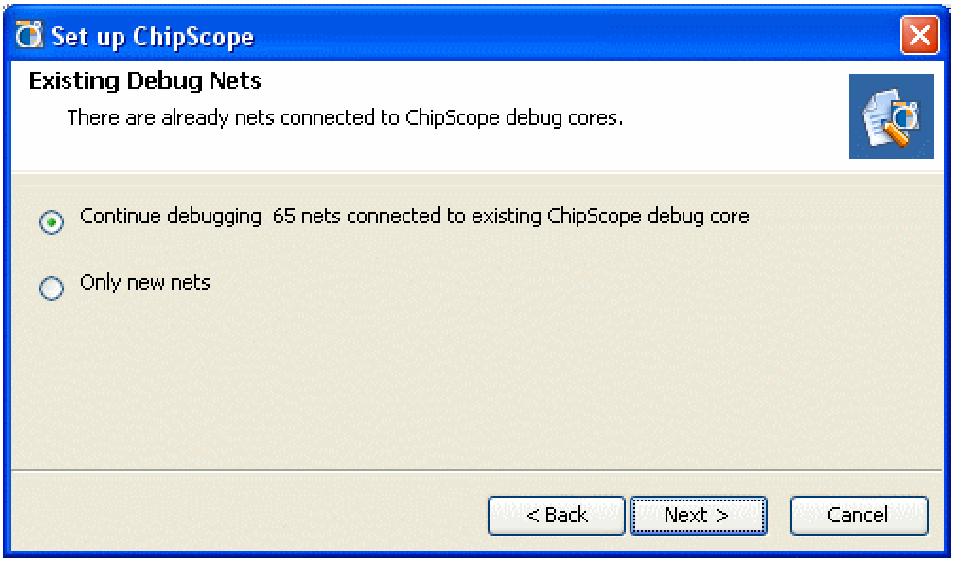

Figure 11: Existing Debug Nets Page of the Set Up ChipScope Wizard

3-1-7. Click **Next** to reconfigure the cores attached to the previous debug nets along with the new nets.

The Specify Nets to Debug page displays, showing an additional 32 nets to debug.

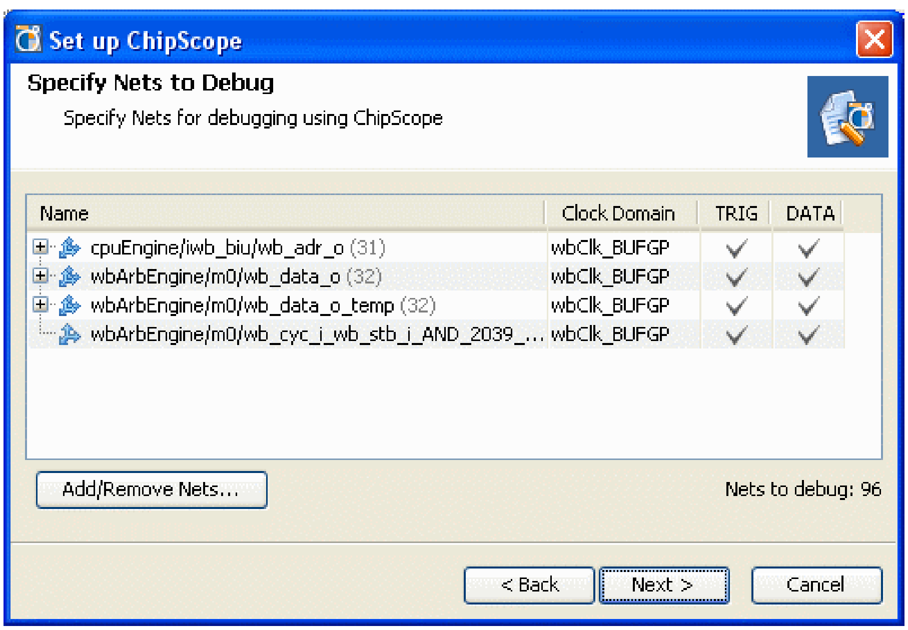

Figure 12: Specify Nets to Debug Page of the Set Up ChipScope Wizard

Notice again that the Clock Domain signals are all `wbClk_BUFGP`.

3-1-8. Click **Next** to accept the list of debug nets.

The Existing ChipScope Cores page displays, prompting you to remove the previous debug cores and create a new set.

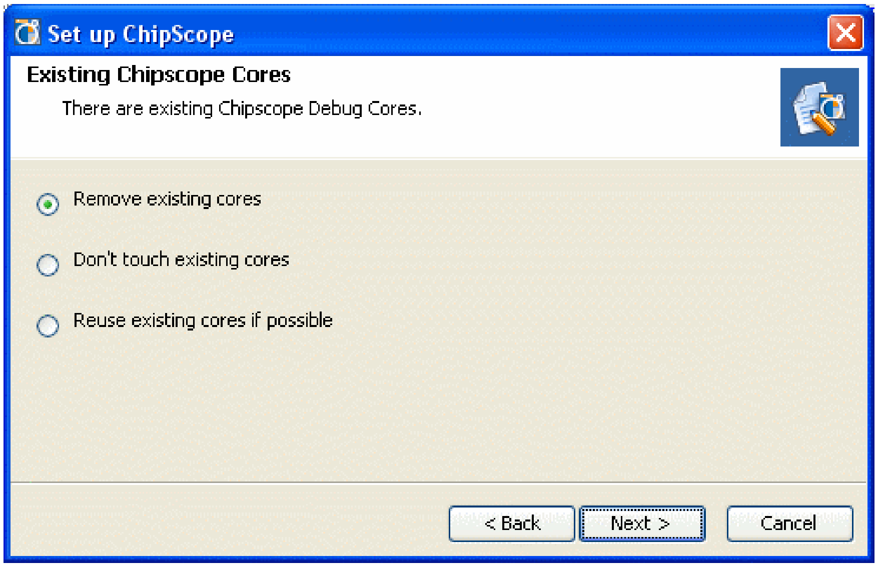

Figure 13: Existing ChipScope Cores Page of the Set Up ChipScope Wizard

3-1-9. Click **Next** to remove the existing cores and create a new set based on the new list of nets and clock domains.

The Summary page displays.

Figure 14: Summary Page of the Set Up ChipScope Wizard

3-1-10. Click **Finish** to remove the previously defined core and create the new one.

3-1-11. Examine the ChipScope view to view the newly added nets and trigger.

Step 4: Change Debug Core Attributes Step 4

Now you change the default attributes for the ChipScope ILA cores.

4-1. Change the csdebugcore_1_0 trigger port TRIG0 port property match_type.

4-1-1. Select the ChipScope view tab to bring it to the front of the screen.

4-1-2. Select the `cs_ila_v1` instance.

4-1-3. Click the TRIG0 debug port of `cs_ila_v1`.

4-1-4. In the Debug Port Properties view, click the Options tab.

4-1-5. In the Debug Port Properties view, click in the right column next to match_type to view a drop down list of the options.

4-1-6. Select **extended_with_edges** to change the trigger port match type.

The Debug Port Properties view is displayed.

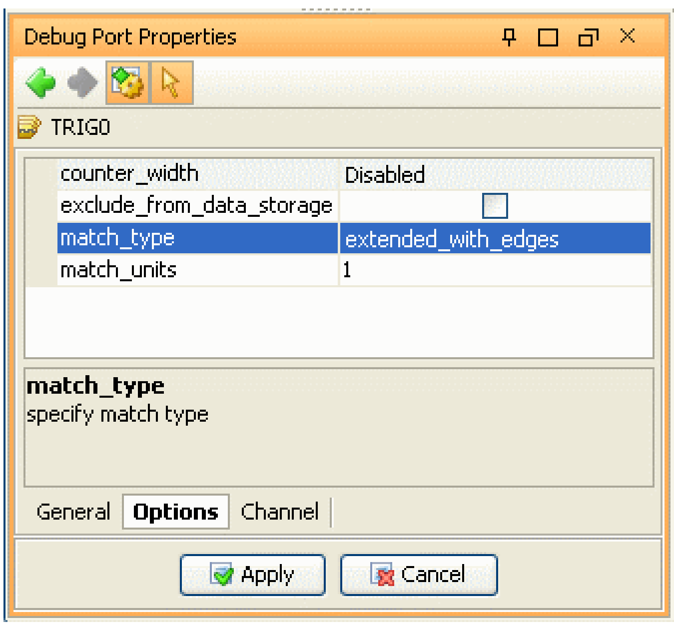

Figure 15: Debug Port Properties View with match_type Set

4-1-7. Click **Apply** to accept the change.

4-2. Change the data sampling depth.

You can change the data sampling depth from the default depth of 1024 to 2048 to take advantage of additional block RAM capacity in the design as follows:

4-2-1. In the ChipScope view, click **cs_ila_v1**.

4-2-2. In the Debug Core Properties view, click **sample_data_depth**.

4-2-3. Click the right column next to sample_data_depth and select **2048** from the drop down list.

The Debug Core Properties view opens.

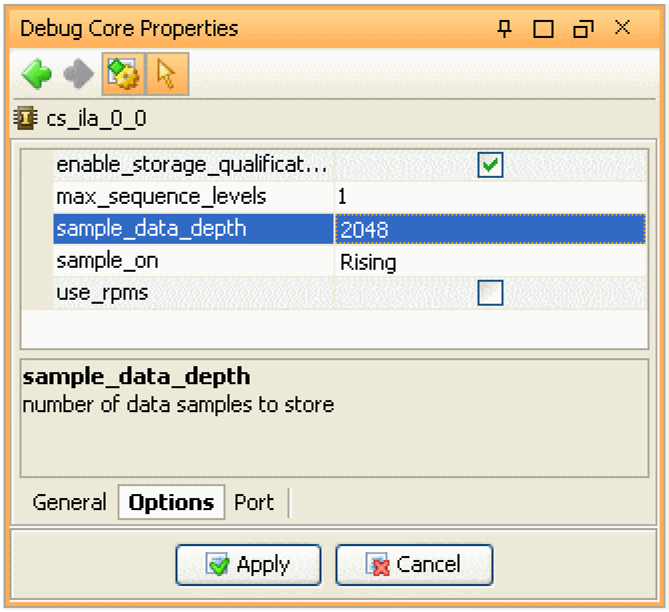

Figure 16: Debug Core Properties with sample_data_depth set to 2048

4-2-4. Click **Apply** to confirm the property change.

Step 5: Implement ChipScope Debug Cores Step 5

You have now inserted black box models for the ChipScope debug cores connected to the debug nets you selected. When you implement the FPGA design using Run Implementation, the PlanAhead software converts these black box debug cores automatically to synthesized cores by calling the CORE Generator software prior to implementing with NGDBuild, Map, and PAR.

However, if you want to floorplan the debug cores along with any critical logic you might be debugging, you must first implement them.

5-1. Implement the ChipScope debug cores.

5-1-1. Click the ChipScope view tab to ensure that it is the active view.

5-1-2. Select the **chipscope_ila_v1** core.

5-1-3. Click the Implement ChipScope Debug Cores button.

5-1-4. Click **OK** to save the Project, if prompted.

This invokes CORE Generator and configures each of the ChipScope Debug cores created in the previous step. This might take a few minutes.

5-1-5. Select the Netlist view tab.

5-1-6. Click Collapse All.

Since the cores are now implemented, the `u_icon` and `chipscope_ila_v1` instances in the Netlist view change from black boxes to cores.

5-1-7. Expand and select the **cs_ila_0_0** instance.

5-1-8. In the Instance Properties view, select the **Statistics** tab.

The Primitive Statistics displays the logic used to implement the core.

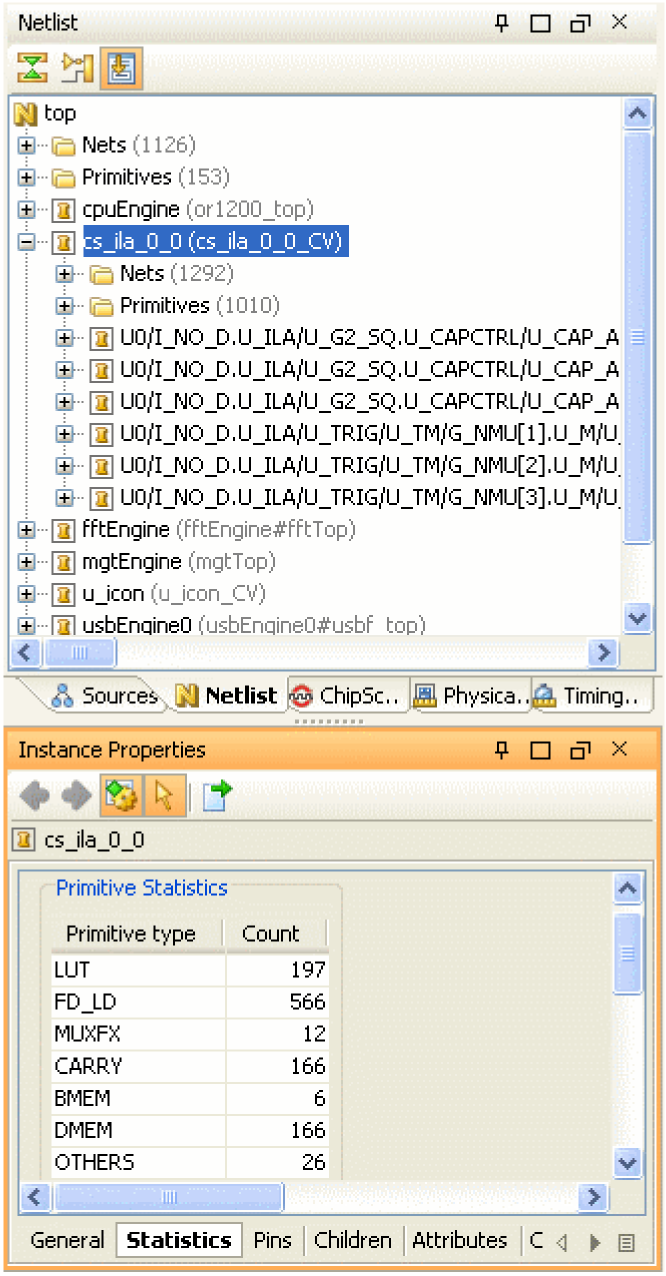

Figure 17: Displaying Debug Core Resource Usage

5-1-9. In the Netlist view, make sure the `cs_ila_0_0` instance is still selected.

5-1-10. In the Netlist view, click Schematic.

The schematic for the ILA core instantiation `cs_ila_0_0` opens.

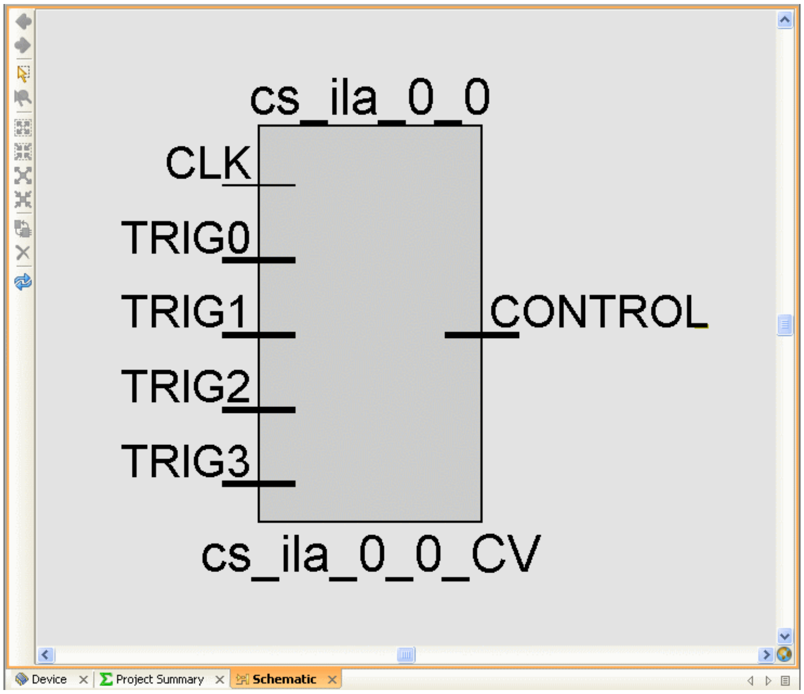

Figure 18: Schematic for the cs_ila_0_0 debug core

5-1-11. Double-click the **CLK** pin and the **TRIG1** pins on the outer side of the instance to expand these nets in the schematic.

5-1-12. In the Schematic window, click the Regenerate Schematic button.

5-1-13. Zoom in to view the expanded `cs_ila_0_0` schematic (Figure 19). To Zoom in, click on the schematic diagram in the upper-left corner and drag the mouse to the lower right.

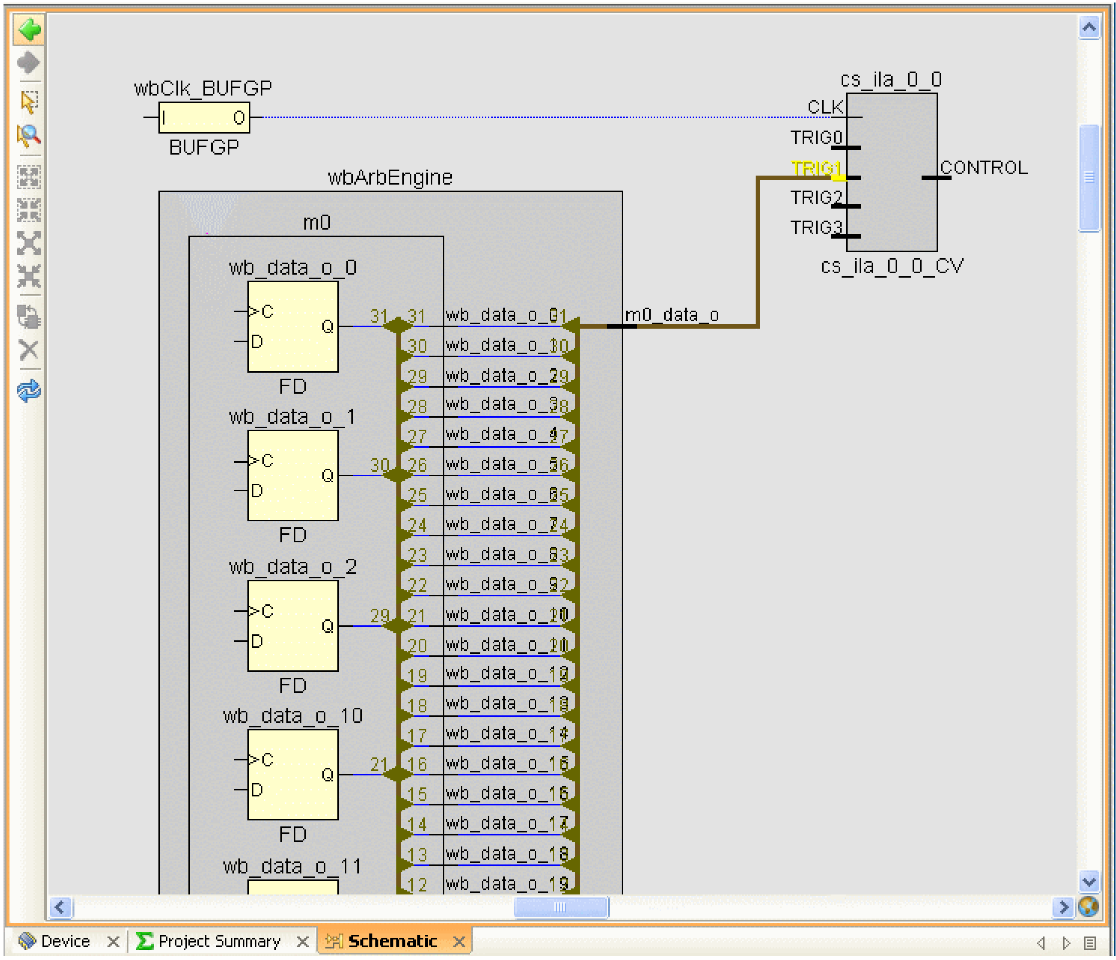

Figure 19: PlanAhead Expanded Schematic of TRIG1 Net

The trigger ports are connected to wishbone interface address control signals generated from the wbArbEngine. These signals are clocked by the clock signal `wbClk`.

Floorplanning is outside the scope of this tutorial, but it should be noted that inserting debug cores on critical logic could impact quality of results. Xilinx recommends that nets driven directly by the output pin of a flop be chosen for debug. It may also be necessary to use AREA_GROUP constraints (Pblocks in PlanAhead) to group the debug cores with critical logic being debugged to keep them placed close together and to minimize the timing impact of core insertion.

Other PlanAhead tutorials provide an introduction to floorplanning and assigning logic hierarchies to AREA_GROUP constraints for placement.

5-1-14. Close the Schematic view.

Step 6: Implement the Design Step 6

At this point, you have:

- Created ChipScope debug cores
- Connected ChipScope debug cores to debug nets
- Changed debug core options from the defaults
- Generated the core netlists

You are now ready to implement the design.

More detailed information about implementing FPGA devices with PlanAhead is provided in other tutorials.

6-1. Implement the design.

6-1-1. In the Flow Navigator, click **Implement**.

The ISE Design Suite implementation is launched using the active constraint set and run strategy.

The PlanAhead environment reports the active results while implementation is running in the Compilation Log and Compilation Messages views.

This design takes approximately 20 minutes to implement. The Status Bar shows when implementation is complete. You could elect to read through the remaining steps rather than running implementation.

6-1-2. After implementation completes, select the **Open Implemented Design** option in the Implementation Complete dialog box.

6-1-3. If prompted, click **Yes** to close the Netlist Design.

The implemented design opens.

6-2. Highlight debug core logic placement.

6-2-1. In the Netlist view, press the **Ctrl** key to select both the cs_ila_0_0 and u_icon instances.

6-2-2. Right-click the instances, and select **Highlight Primitives > Cycle Colors**.

The logic in the debug cores is highlighted using different colors.

6-2-3. On the main toolbar, click Unhighlight All .

Step 7: Generate a Bitstream - Launch ChipScope Analyzer Step 7

You have now implemented the design and closed timing. Although this sample design might have timing failures, you can still generate a bitstream that can be used to program a device.

7-1. Generate a Bitstream for the implemented run.

7-1-1. In the Flow Navigator, select **Program and Debug > Generate Bitstream**.

The Generate Bitstream dialog box opens.

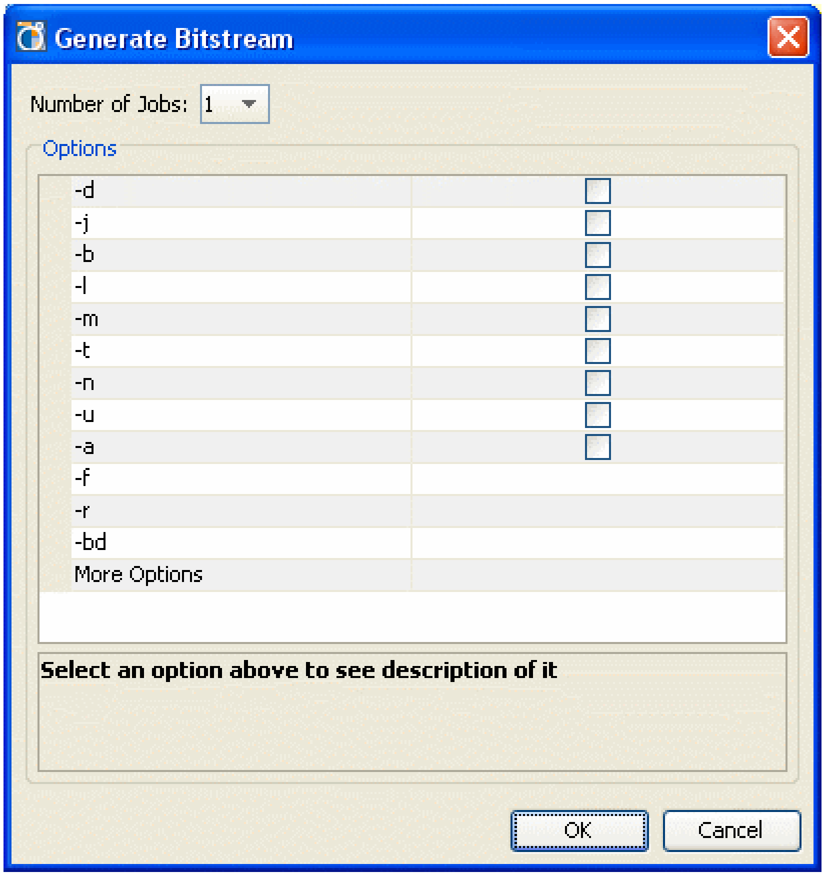

Figure 20: Generate Bitstream Dialog Box

7-1-2. Click **OK**.

7-1-3. In the BitGen Completed dialog box, click **OK**.

Once the Generate Bitstream command completes, the bit file can be downloaded to the FPGA device through the normal programming mechanisms.

You can now run ChipScope Analyzer.

7-2. Launch ChipScope Analyzer on the impl_1 bitstream.

7-2-1. In the Flow Navigator, select **Program and Debug > Launch ChipScope Analyzer**.

ChipScope Analyzer opens.

This step requires a functioning board connected to your computer. Because this is a demonstration design, this step does not work unless you have a FPGA device connected with which the Analyzer software can communicate.

7-2-2. In the ChipScope Analyzer view, select File > Exit. Click No when prompted to save.

ChipScope Analyzer closes.

7-2-3. Select **File > Exit**. If prompted, click **No** to save and **OK** to close PlanAhead.

Conclusion

In this tutorial, you:

- Used a sample design to set up and configure ChipScope debug cores.
- Selected nets to attach to trigger ports of the debug cores.
- Added selected additional nets to add to the debug cores.
- Changed the default settings for the implementation of debug cores.
- Implemented these Debug cores by calling CORE Generator to synthesize the configured debug cores and convert them from black boxes to implemented cores.
- Implemented the design and back-annotated placement and timing reports into the design.
- Ran BitGen.
- Launched the ChipScope Analyzer on the completed design.