Single Chip Crypto Lab Using PR/ISO Flow with the Virtex-5 Family

XAPP1135 (v1.1.3) June 19, 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.

© 2009–2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, 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
06/15/09	1.0	Initial Xilinx release.
04/15/10	1.1	Updated for ISE 11.4 and Trusted Routing throughout. Removed "Synthesize Modules for Use in the Partial Reconfiguration Flow" section from Chapter 1 and put into new Chapter 2. Removed "Using the PlanAhead Tool to Place the Trusted Bus Macros" section in Chapter 3. Removed Appendix A: Trusted Bus Macro Rules.
09/21/10	1.1.1	Added EAR banner to first page.
08/29/11	1.1.2	Removed EAR banner. Updated disclaimer.
06/19/13	1.1.3	In Reference Design Files, added URL to download reference design.

Table of Contents

About This Guide

This lab covers the creation and implementation of a single-chip crypto (SCC) system, utilizing an isolated, redundant AES module. Complete step-by-step instructions are given for the entire process.

Guide Contents

This manual contains the following chapters:

- Chapter 1, "Design Challenge," describes the SCC design and the goals of the lab.
- Chapter 2, "Synthesizing Modules for Partial Reconfiguration Flow," describes the steps in the bottom-up synthesis flow.
- Chapter 3, "Implementing the System," gives step-by-step instructions for implementing the SCC design.
- Chapter 4, "Running the Isolation Verification Tool Against the UCF," covers running IVT against the pre-placed-and-routed design.
- Chapter 5, "Implementing the Design with the PlanAhead Tool," details placing and routing the SCC design.
- Chapter 6, "Verifying the Design with the NCD Isolation Verification Tool," describes running the verification tool on the placed-and-routed design.

Additional Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/support/documentation/index.htm.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support/mysupport.htm.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*
Helvetica bold	Commands that you select from a menu	**File** → **Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *Command Line Tools User Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

Design Challenge

From the instructions in this chapter, a Single Chip Crypto system can be created and implemented utilizing an isolated, redundant AES module targeting a Virtex®-5 or Virtex-5Q FPGA utilizing the ISE® 11.4 and PlanAhead™ 11.4 development tools. It is necessary to use the ISE 11.4 software for this lab. Figure 1-1 is a hierarchical diagram of the various VHDL sub-blocks used in the implementation of the design. This Single Chip Crypto lab shows how to develop a dual-AES design. The user can expand upon this to create their own custom design based on the design methodology.

Type 1 Virtex-5 FPGA Crypto applications require defense-grade (XQ) devices for mask control. The design example used in this application note was created using a non defense-grade Virtex-5 XC5VLX85-FF676-2 device.

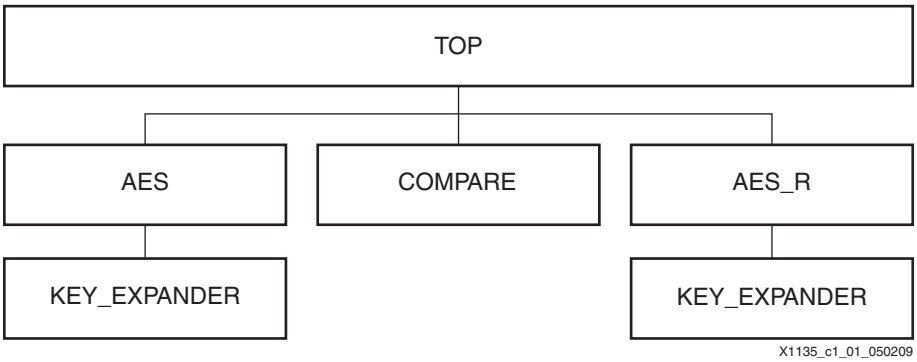


Figure 1-1: **VHDL Design Hierarchical Block Diagram**

Figure 1-2 shows the design flow used for the Single Chip Crypto lab.

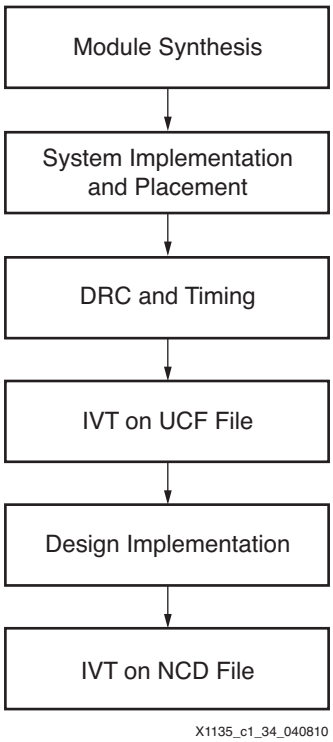


Figure 1-2: **SCC System Design Flow**

Figure 1-3 shows the partitioning and I/O mapping for an XC5VLX85-2FF676 device.

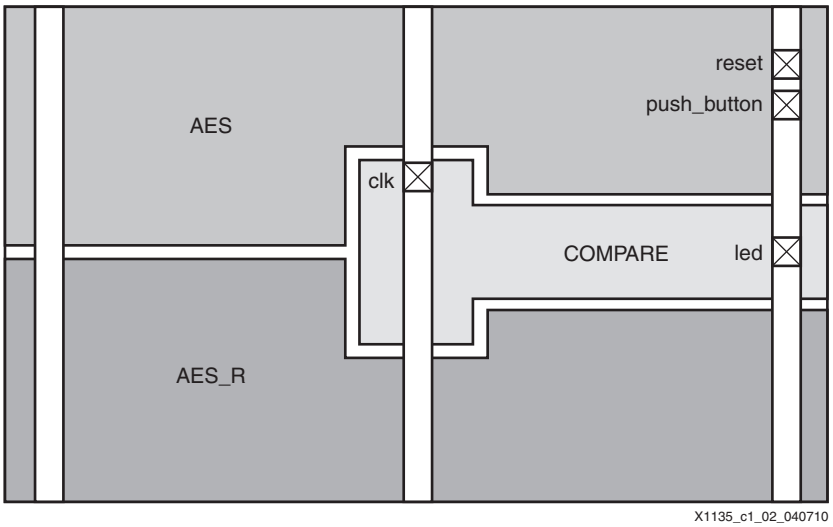
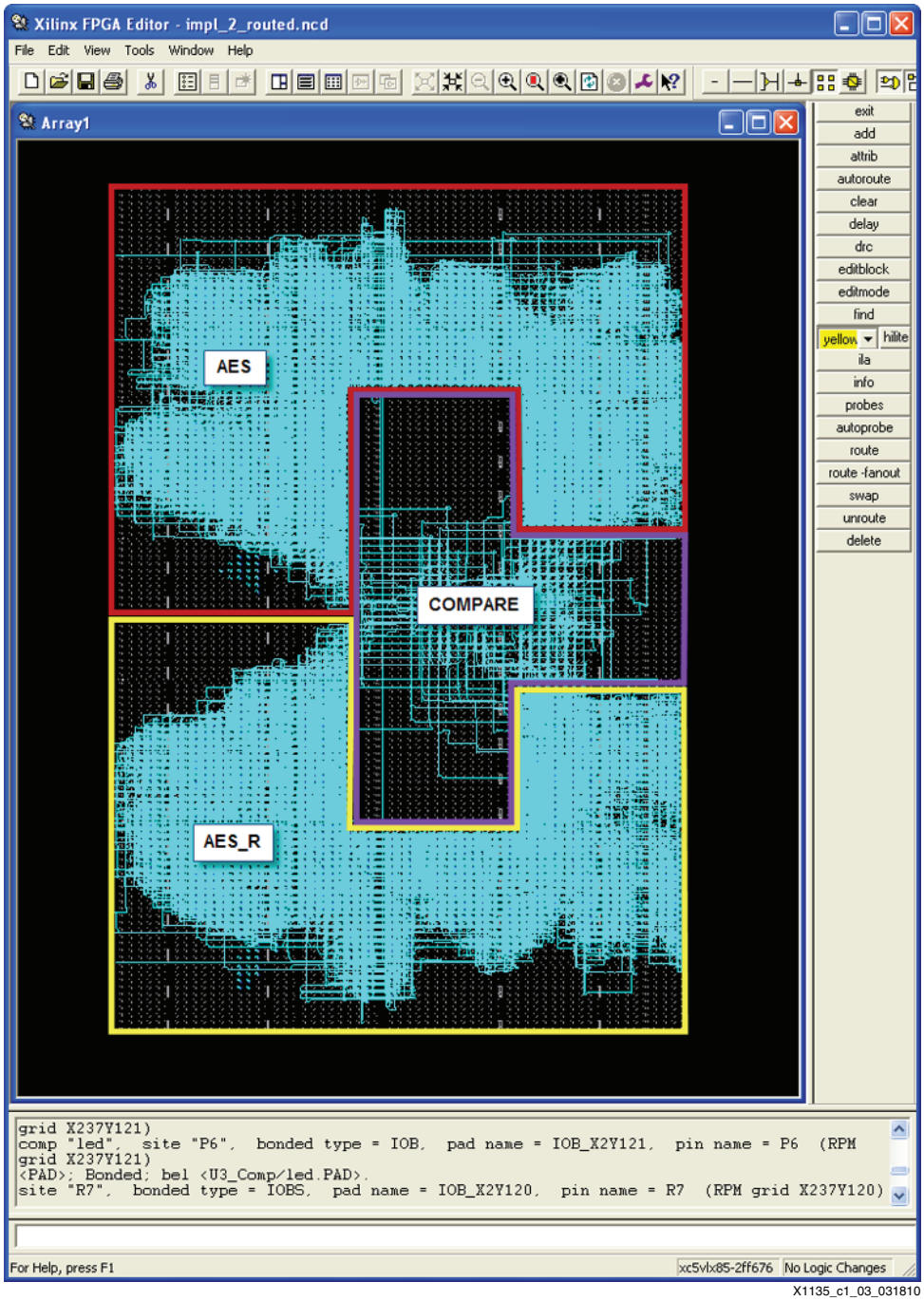


Figure 1-3: **Die View: Partition and I/O Map of XC5VLX85-2FF676**

Figure 1-4 is a view of the completed XSCC Reference design being displayed in the Xilinx FPGA Editor tool. The design incorporates two AES algorithm blocks: AES and AES_r. Both of the AES blocks are driven by the same input data and key. In Figure 1-4, the two AES blocks are the sideways C-shaped regions. Both of the AES blocks are tied to the

compare block, which compares the outputs of the AES blocks. If the outputs of the AES blocks do not match, the compare block sends an alert to the user indicated by an LED. Both AES blocks have an input that allows the user to inject an error. However, only the AES block has the error injection input tied to an external pushbutton.



X1135_c1_03_031810

Figure 1-4: **FPGA Editor View: Implementation of the Single Chip Crypto Flow**

Reference Design Files

Reference Design Checklist

The design files for this application note can be downloaded from:

https://secure.xilinx.com/webreg/clickthrough.do?cid=343393

The design checklist in Table 1-1 includes simulation, implementation, and hardware details for the reference design.

Table 1-1: **Design Checklist**

Parameter	Description
General	
Developer Name	Xilinx
Target devices	Virtex-5 LX85 FPGA
Source code provided	Y
Source code format	VHDL
Design uses code/IP from existing Xilinx application note/ reference designs, CORE Generator™ software, or 3rd party	N
Simulation	
Functional simulation performed	Y
Timing simulation performed	Y
Testbench used for functional and timing simulations	Y
Testbench format	VHDL
Simulator software/version used	ISE 11.4 software
SPICE/IBIS simulations	N
Implementation	
Synthesis software tools/version used	XST
Implementation software tools/versions used	ISE 11.4 software
Static timing analysis performed	Y
Hardware Verification	
Hardware verified	N
Hardware platform used for verification	N/A

Install Reference Design Files into Target Directories

These steps describe the process for installing the reference design files:

Note: It is best if the design files are unzipped onto a directory without any spaces in the path.

1. Copy `XAPP1135.zip` to the Windows desktop.
2. Double-click on `XAPP1135.zip` and unzip the contents to the desired location.
3. The project files are placed in the following directories:

```
\Xilinx_Design\source\
\Xilinx_Design\ivt
\Xilinx_Design\synthesis\
\Xilinx_Design\PlanAhead
\Xilinx_Design\BuildScripts
\Xilinx_Design\results
```

Chapter 2

Synthesizing Modules for Partial Reconfiguration Flow

This chapter describes the steps that take the user through a bottom-up synthesis flow, which is the flow used for SCC designs to maintain isolation. The top module is synthesized first followed by each of the lower-level modules (AES, AES_R, COMPARE).

Synthesize the `top` Module

These steps describe how to synthesize the `top` module:

1. Start the ISE 11.4 software:

   Start → **All Programs** → **Xilinx ISE Design Suite 11** → **ISE** → **Project Navigator**

2. Create a new ISE 11.4 software project:

   File → **New Project**

3. As shown in Figure 2-1, set the Project Location to **`\Xilinx_Design\synthesis`**, set the Project Name to **top**, and click **Next**.

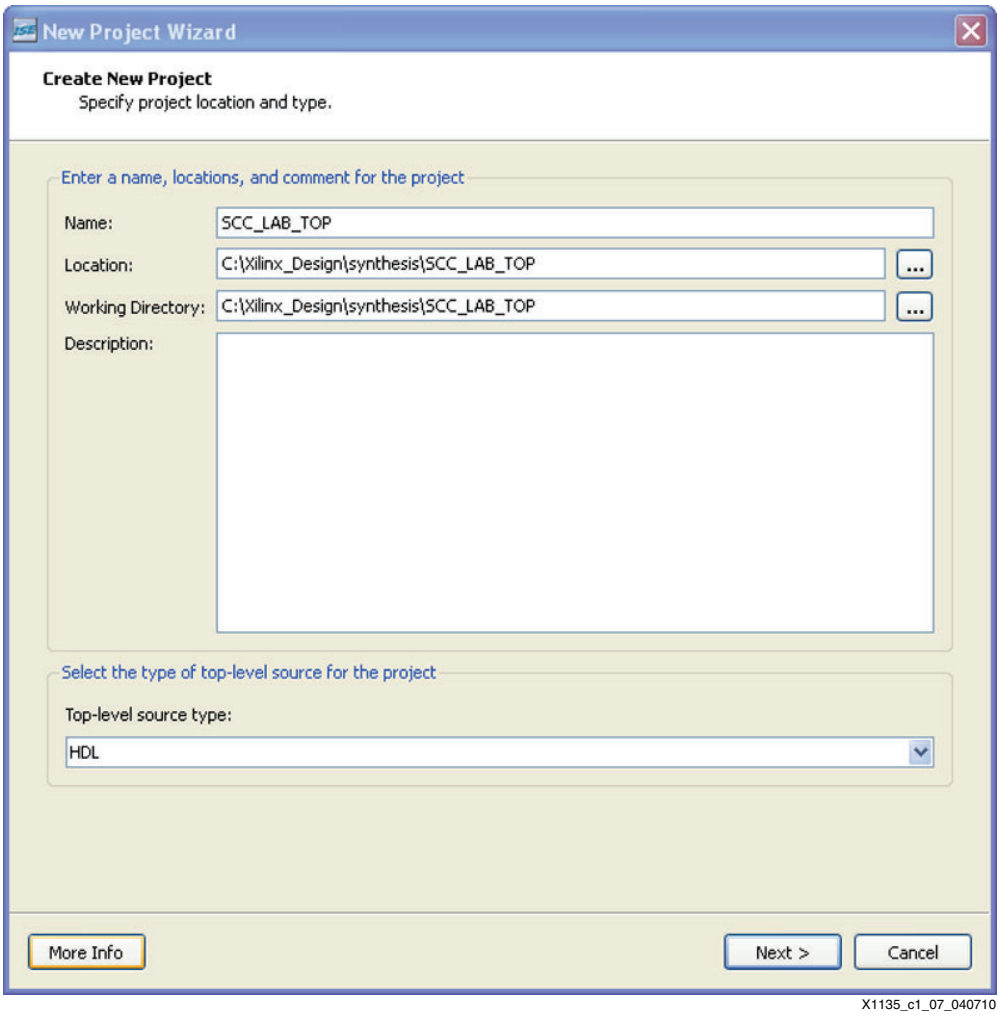
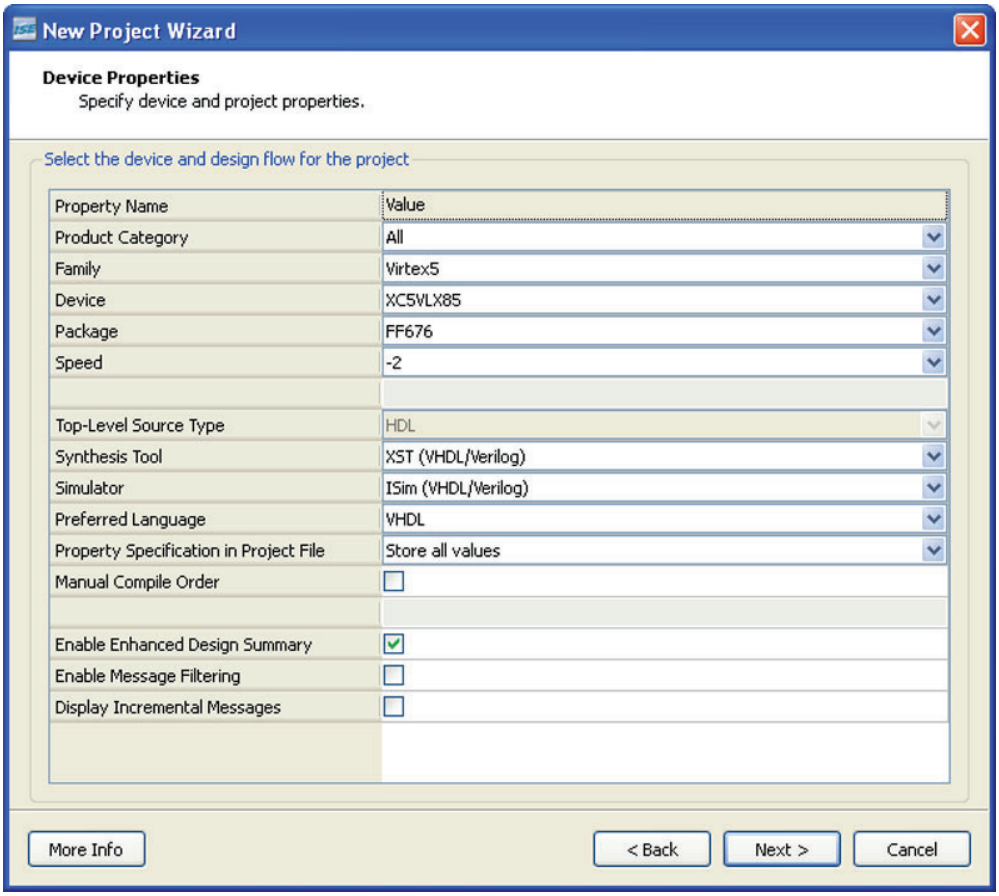


Figure 2-1: **New Project Wizard (Create New Project)**

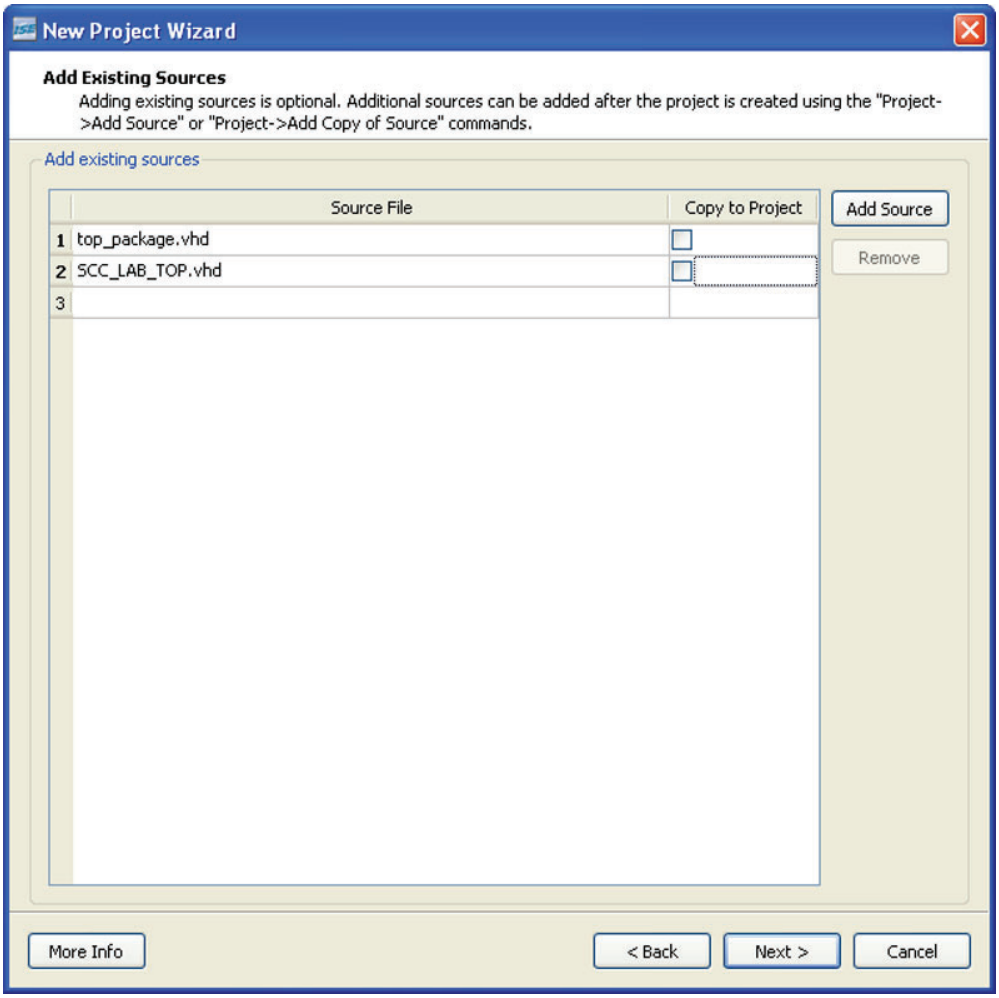
4. Choose an XC5VLX85-FF676-2 FPGA as the target device (see Figure 2-2).



X1135_c1_08_040710

Figure 2-2: **New Project Wizard (Device Properties)**

5. Click **Next** twice, click the **Add Source** button, navigate to the `source\design` directory, and add the `top_package.vhd` and `SCC_LAB_TOP.vhd` files to the project.
6. Uncheck **Copy to Project** for all sources and then click **Next** (see Figure 2-3).



X1135_c1_09_040710

Figure 2-3: **New Project Wizard (Add Existing Sources)**

7. Click **Finish** (see Figure 2-4).

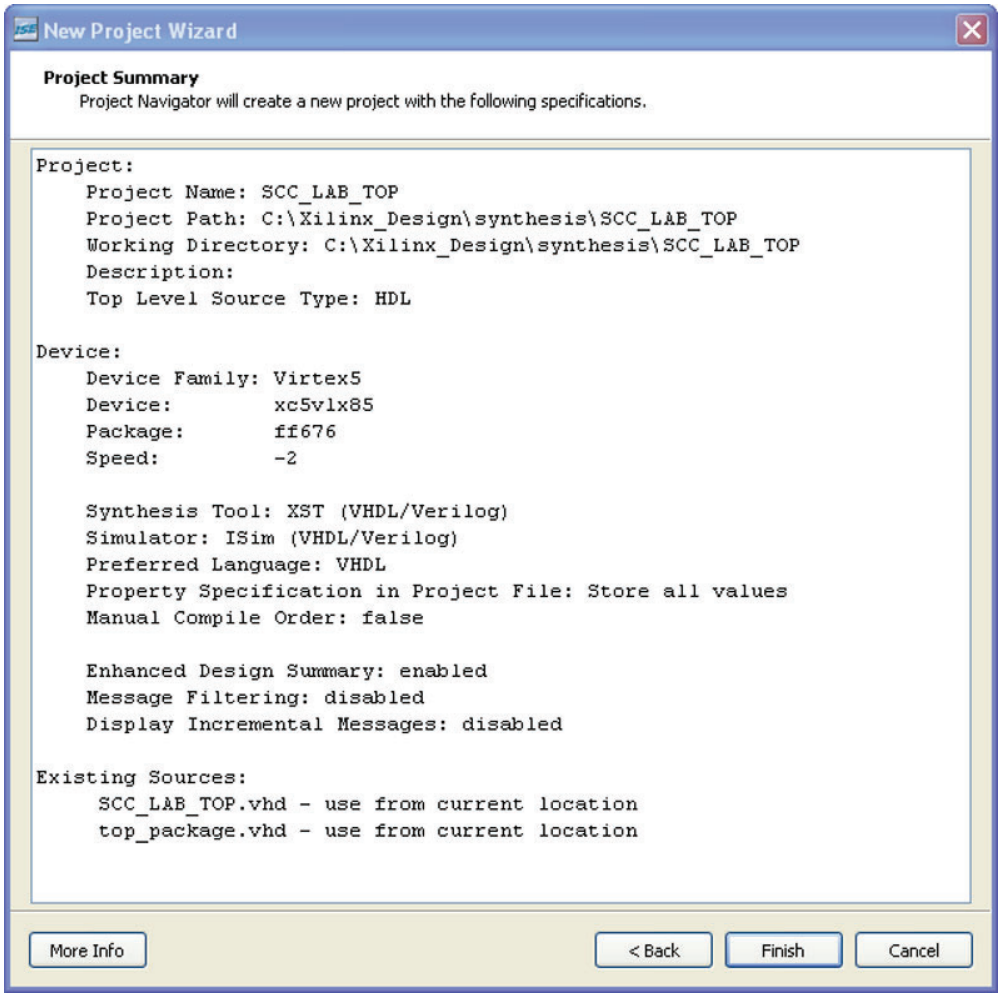
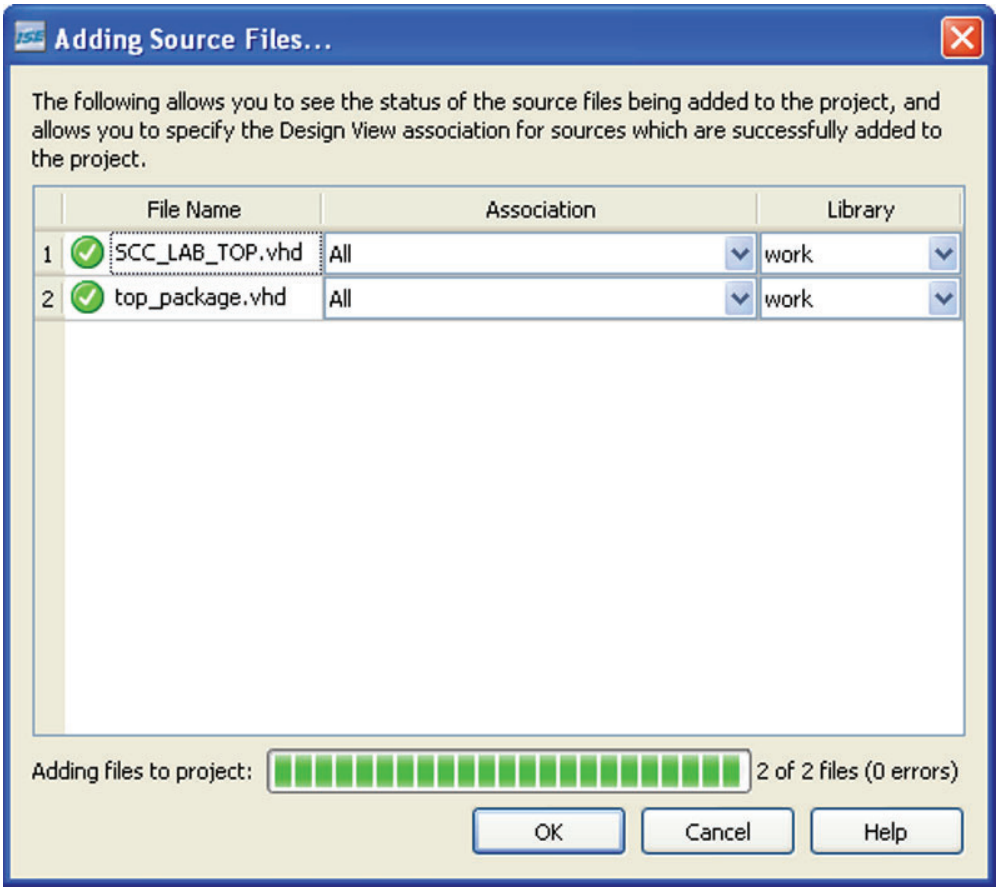


Figure 2-4: **New Project Wizard (Project Summary)**

8. Click **OK** (see Figure 2-5).

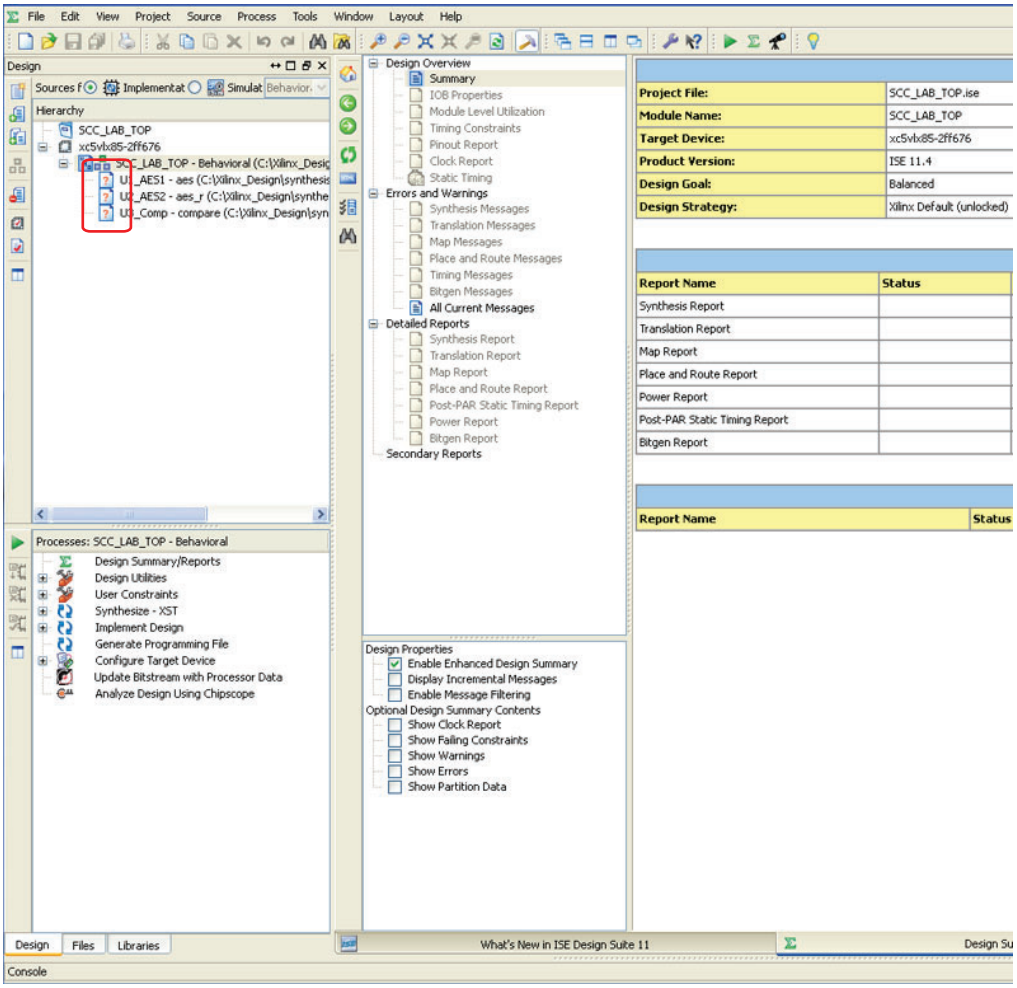


X1135_c1_11_040710

Figure 2-5: **Adding Source Files**

The ISE Project Navigator Window should look like the window shown in Figure 2-6.

Note: The question marks (?) by all blocks that are not coded at the top level are expected because only the top level is being synthesized at this time. All other blocks are out of context and are treated as black boxes.



X1135_c1_12_031810

Figure 2-6: **Project Navigator Window**

9. Open the `SCC_LAB_TOP.vhd` file and locate the buffer_type attributes:

```
attribute buffer_type: string;
attribute buffer_type of push_button : signal is "none";
attribute buffer_type of reset       : signal is "none";
attribute buffer_type of led         : signal is "none";
```

   The buffer_type attribute directs XST to disable I/O insertion on these ports. The buffer_type attribute is necessary to guarantee I/Os are included in the lower-level modules, and therefore are part of the isolated regions. However, in `SCC_LAB_TOP.vhd`, clk is driven to all modules. Clock networks are not required to be isolated. As such, do not apply this attribute to the CLK pin of the top-level code.

10. To run XST synthesis, either right-click on the Synthesize-XST icon in the Processes window and click **Run** or simply double-click **Synthesize-XST**. After synthesis is complete, close the top project.

Synthesize the `aes` Module

These steps describe how to synthesize the `aes` module:

1. Start the ISE 11.4 software.

   Start → **All Programs** → **Xilinx ISE Design Suite 11** → **ISE** → **Project Navigator**

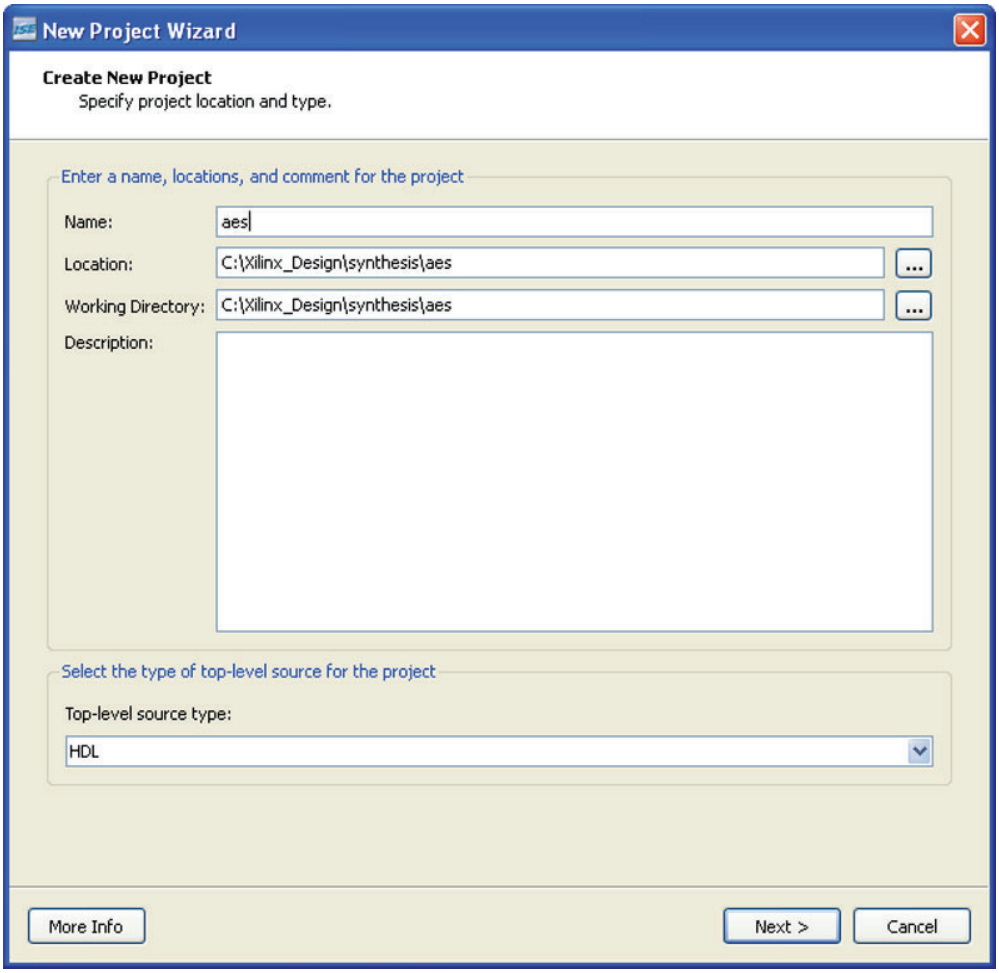
2. Create a new ISE 11.4 software project:

   File → **New Project**

3. Set the Project Location to:

   `\Xilinx_Design\synthesis`

4. Set the Project Name to **aes**.
5. Click **Next** (see Figure 2-7).



X1135_c1_13_040710

Figure 2-7: **New Project Wizard (Create New Project)**

6. Click **Next** (see Figure 2-8).

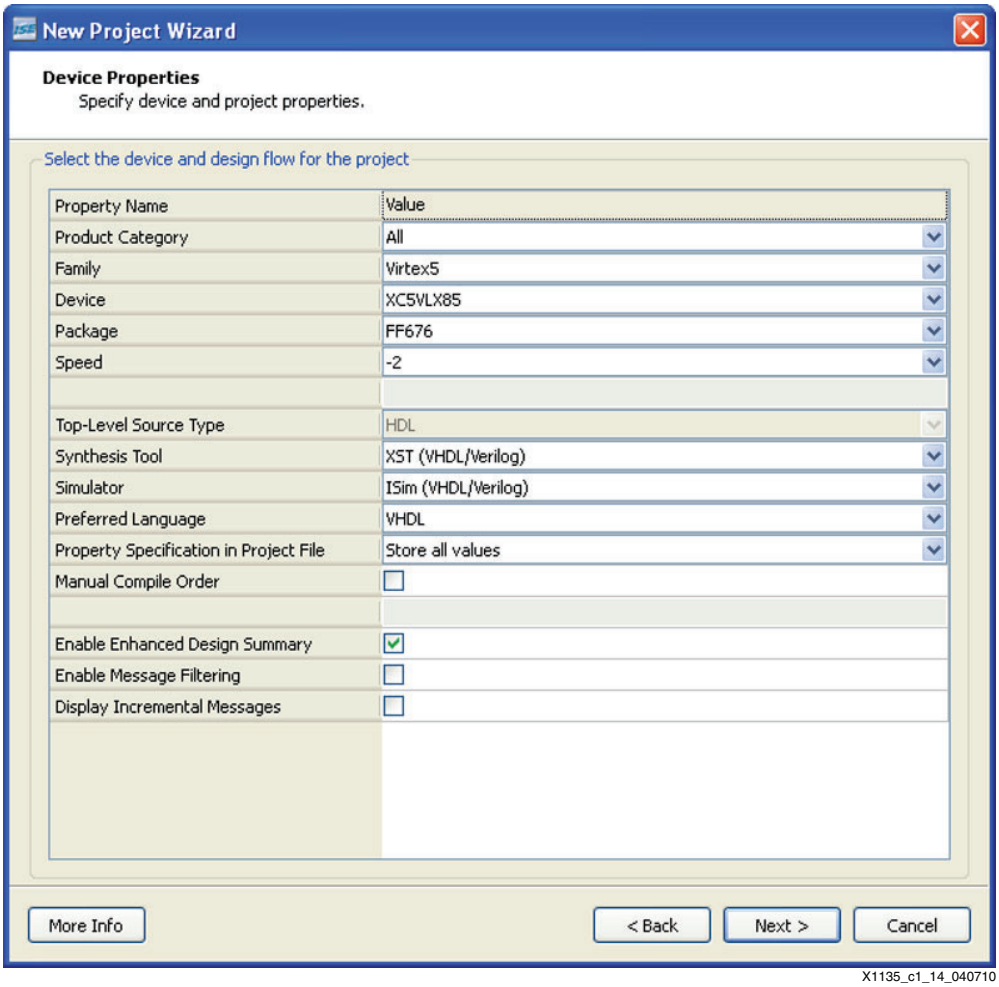
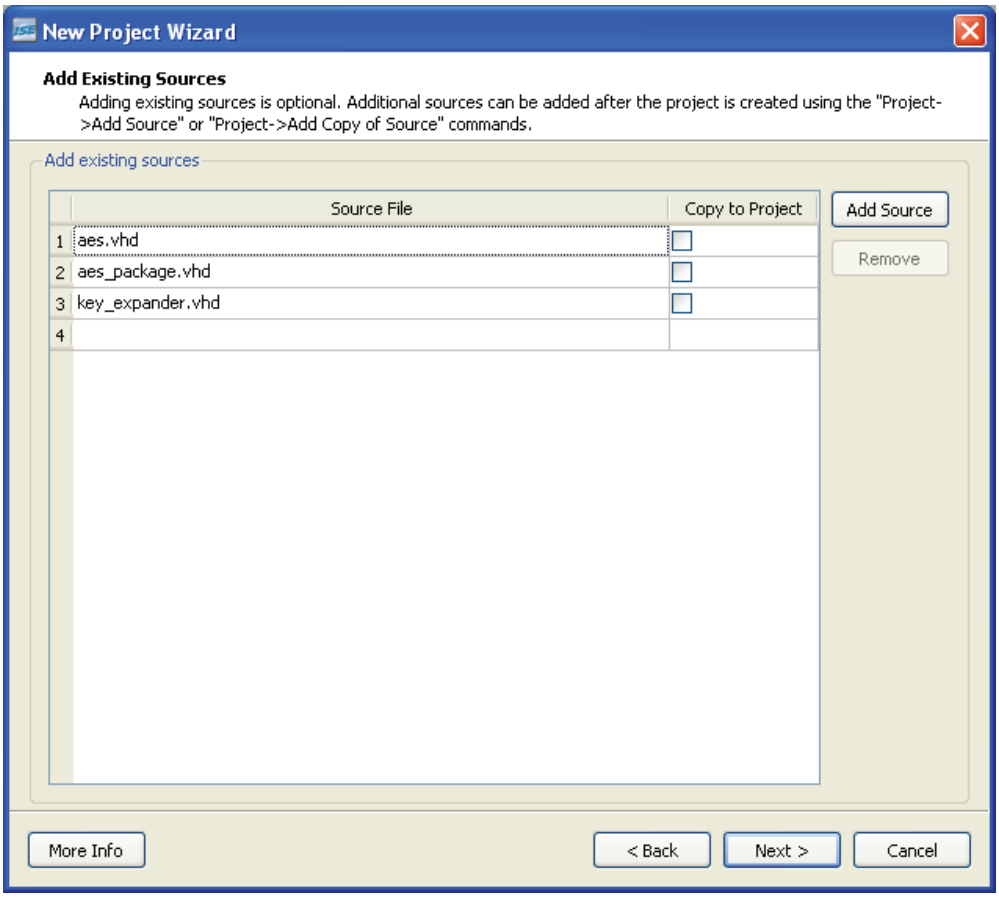


Figure 2-8: **New Project Wizard (Device Properties)**

7. Click **Next** twice.
8. To add the `key_expander.vhd`, `aes.vhd`, and `aes_package.vhd` files to the project, click the **Add Source** button and navigate to the `source\design` directory.
9. Uncheck **Copy to Project** for all sources and then click **Next** (see Figure 2-9).



X1135_c1_15_040710

Figure 2-9: **New Project Wizard (Add Existing Sources)**

10. Click **Finish** (see Figure 2-10).

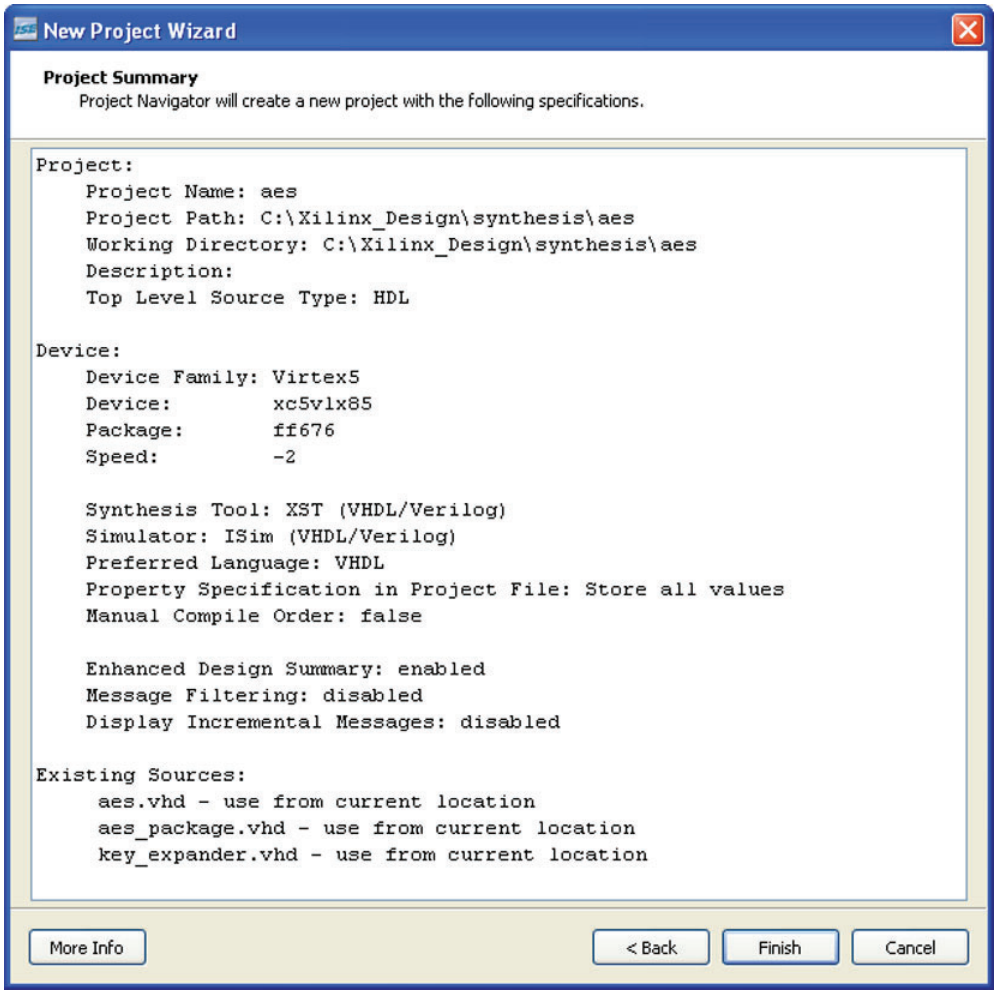


Figure 2-10: **New Project Wizard (Project Summary)**

11. Click **OK** (see Figure 2-11).

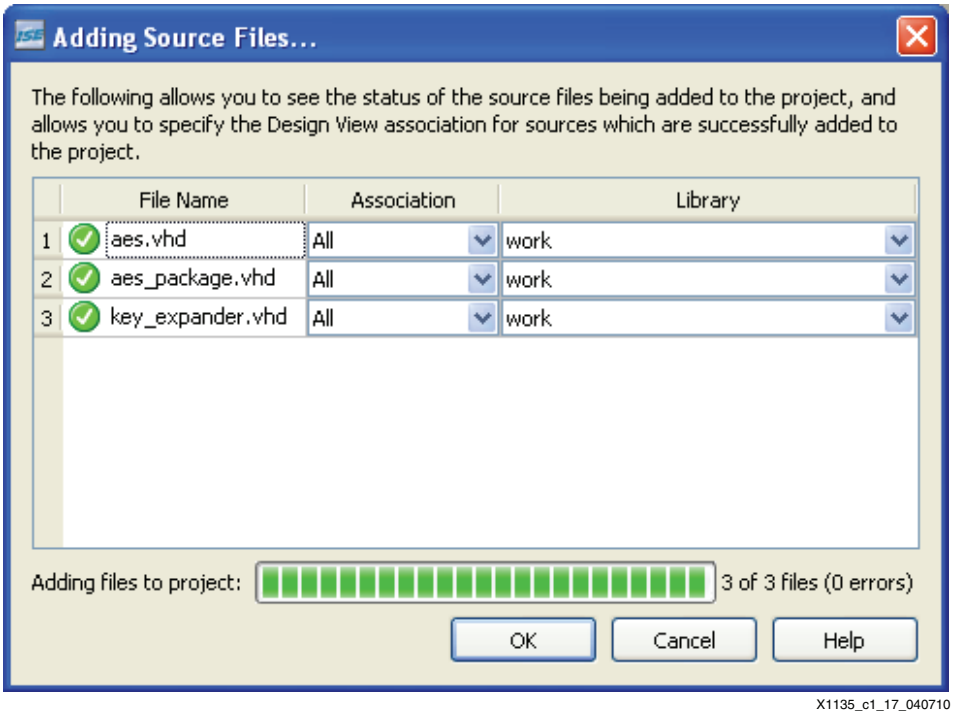


Figure 2-11: **Adding Source Files**

The ISE Project Navigator Window should look like the window shown in Figure 2-12.

Note: All blocks are defined. There should be no question marks (?) by any module. All modules at this level and below are in context.

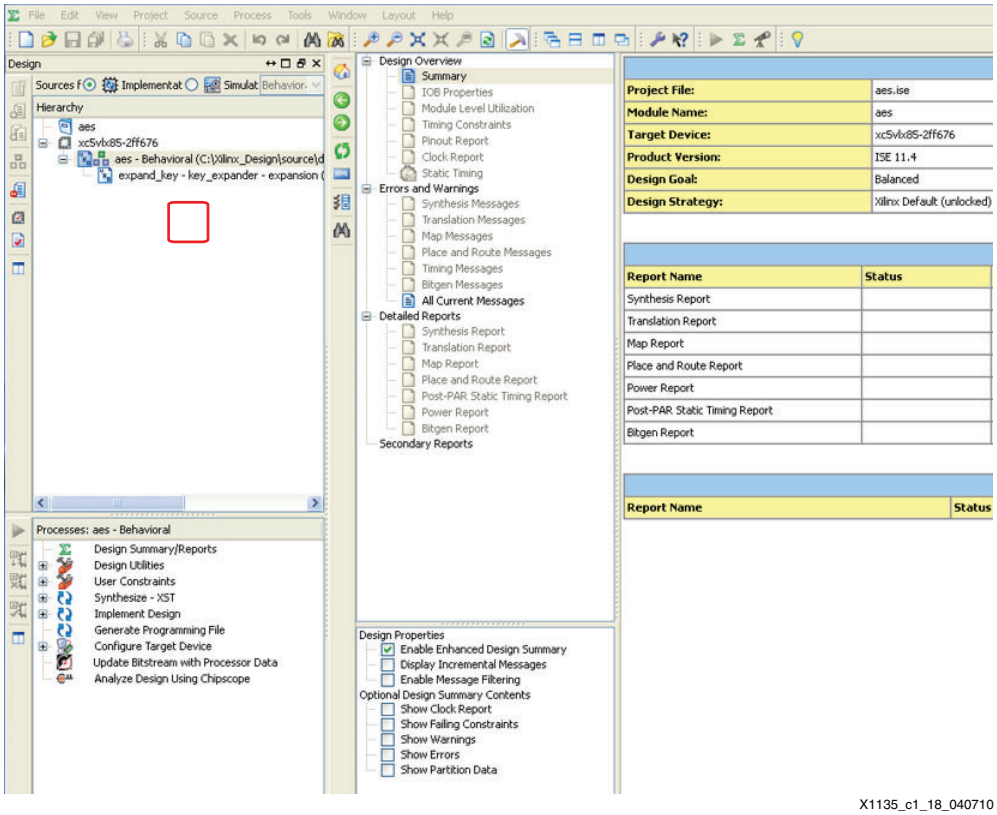


Figure 2-12: **Project Navigator Window**

12. Open the `aes.vhd` file and locate the section containing the buffer_type attributes:

```
attribute buffer_type: string;
attribute buffer_type of clk          : signal is "none";
attribute buffer_type of start        : signal is "none";
attribute buffer_type of mode         : signal is "none";
attribute buffer_type of load         : signal is "none";
attribute buffer_type of key          : signal is "none";
attribute buffer_type of data_in      : signal is "none";
attribute buffer_type of reset_out    : signal is "none";
attribute buffer_type of data_out     : signal is "none";
attribute buffer_type of done         : signal is "none";
```

The buffer_type attribute directs the XST tool to disable I/O insertion on the specified ports. By default, XST inserts an I/O on all ports. The buffer_type attribute is necessary to guarantee I/Os are not placed at this level in the hierarchy either because the I/Os are placed at the top level (such as CLK) or because the port is a direct connection to a port of another instance.

Note: The reset and push_button signals do not have an attribute associated with them because they are "owned" by AES and therefore require an I/O to be inferred within the `aes` module rather than the `top`. The clk I/O was inferred when the `top` was synthesized. The other signals are internal to the FPGA; therefore they have the attribute **`signal is "none"`** applied.

13. In the `aes.vhd` source code file, the LUT instantiation between reset and reset_out is necessary for Trusted Routing rules. Refer to XAPP1134, *Developing Secure Designs*

Using the Virtex-5 Family, for more details on Trusted Routing rules. A section of the VHDL code for the LUT instantiation is shown here:

```
-- Instantiate LUT buffers on nets that either drive two different regions
-- or are feedthrough's from one region to the next. This prevents a single
-- net being placed "shorting" three regions together. Trusted Bus Macros
-- fulfilled this requirement automatically.

  lut_reset_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => reset, O => reset_out );
```

14. Right-click on the Synthesize-XST icon in the Processes window and select **Process Properties…**. Set Optimization Goal to **Area**, set Optimization Effort to **High**, and click **OK** (see Figure 2-13).

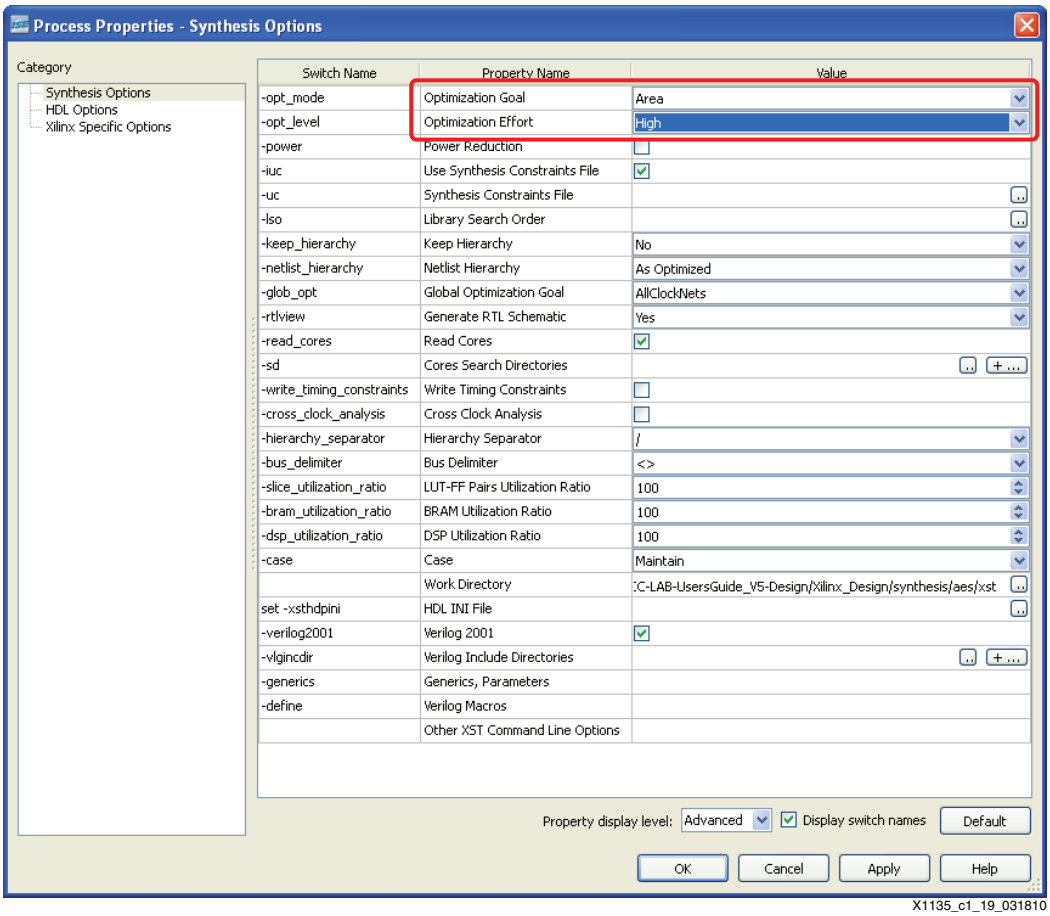


Figure 2-13: **Process Properties**

15. To run XST synthesis, either right-click on the Synthesize-XST icon in the Processes window and select **Run** or simply double-click **Synthesize-XST**.
16. After synthesis is complete, close the aes project.

Synthesize the aes_r Module

These steps describe how to synthesize the aes_r module:

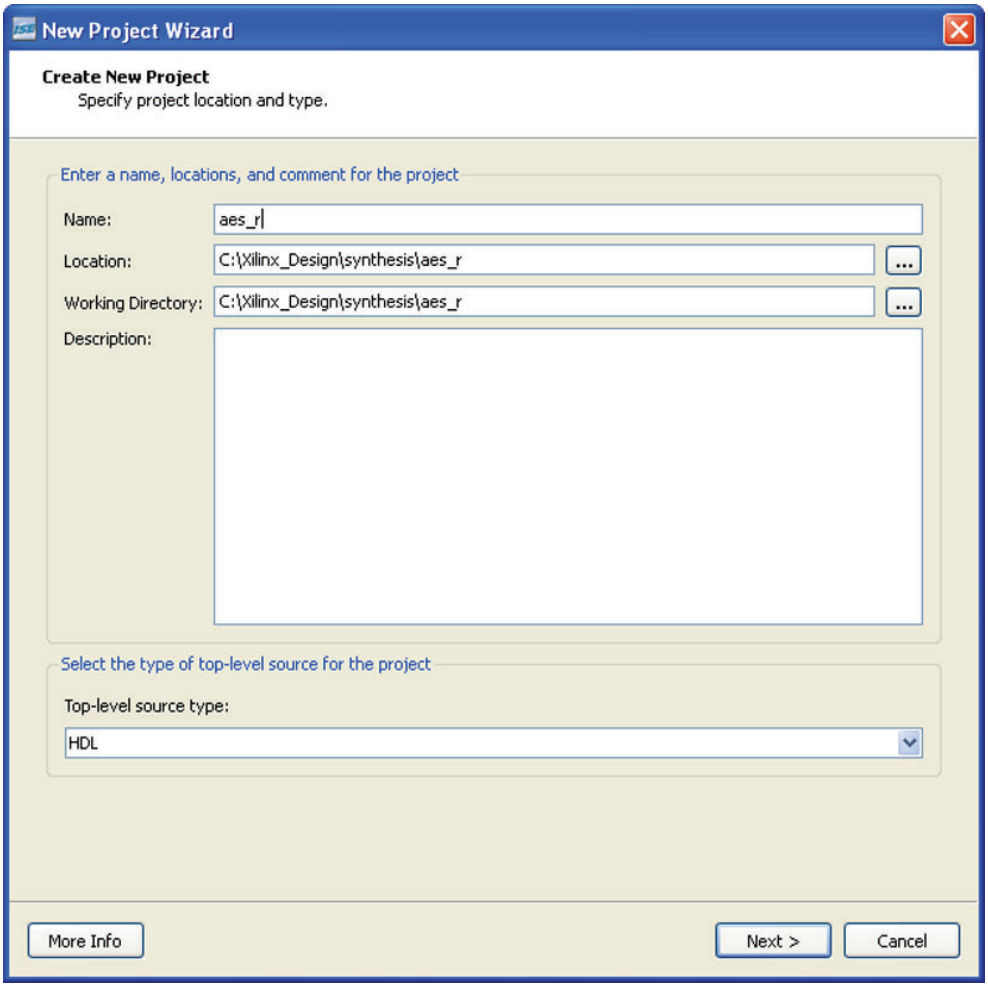
1. Start the ISE 11.4 software:

   Start → **All Programs** → **Xilinx ISE Design Suite 11** → **ISE** → **Project Navigator**

2. Create a new ISE 11.4 project:

   File → **New Project**

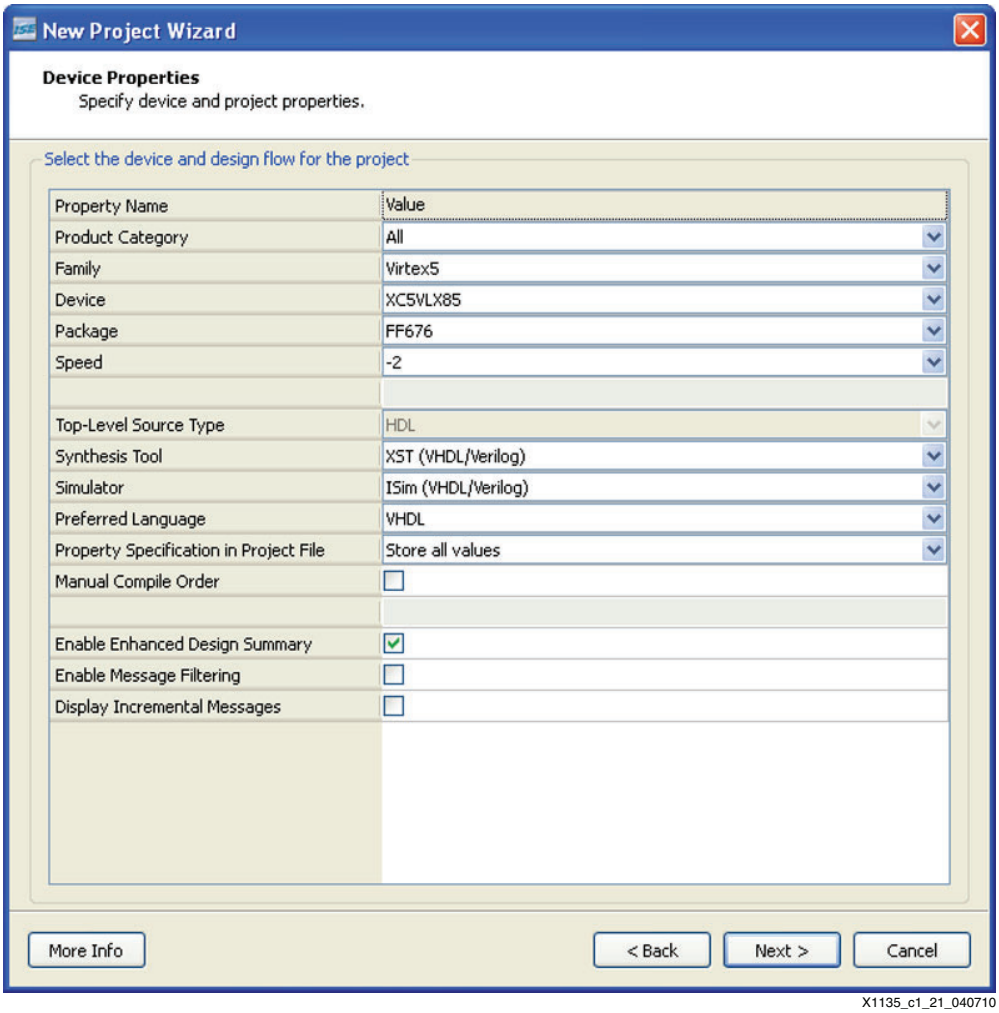
3. Set the Project Location to **\Xilinx_Design\synthesis**.
4. Set the Project Name to **aes_r**.
5. Click **Next** (see Figure 2-14).



X1135_c1_20_040710

Figure 2-14: **New Project Wizard (Create New Project)**

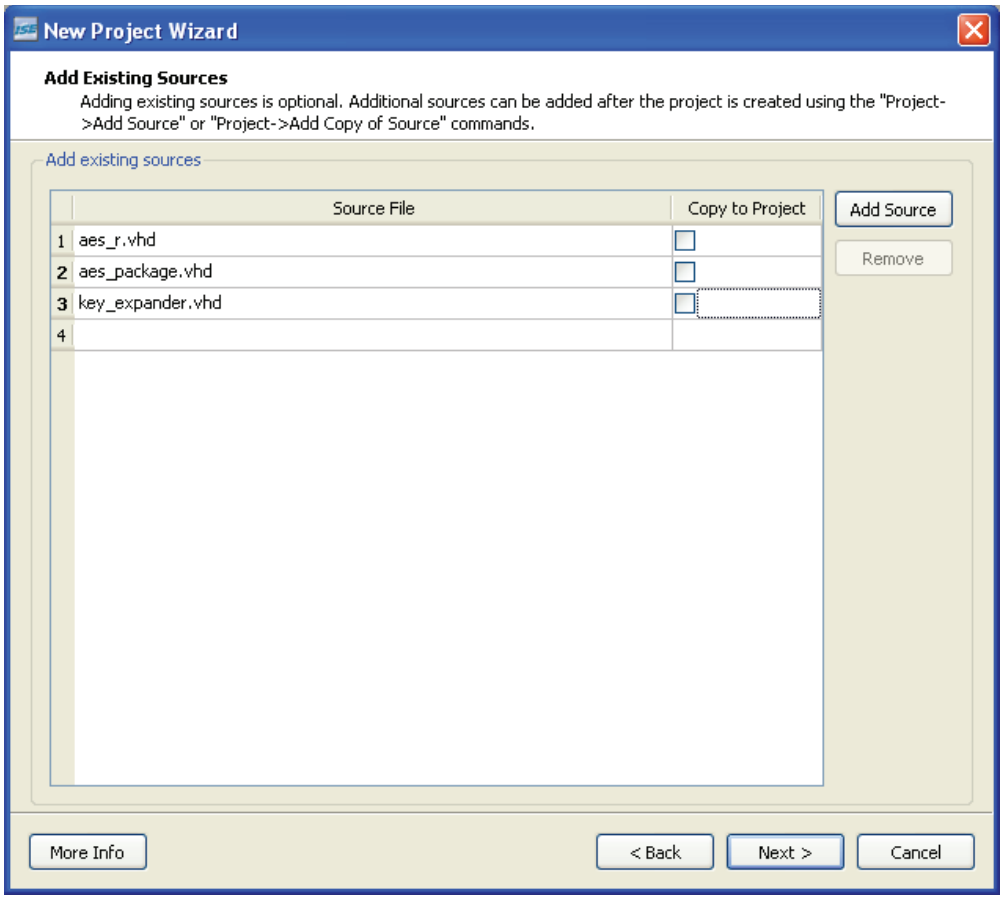
6. Choose an XC5VLX85-FF676-2 as the target device (see Figure 2-15).



X1135_c1_21_040710

Figure 2-15: **New Project Wizard (Device Properties)**

7. Click **Next** twice. Add the `key_expander.vhd`, `aes_r.vhd`, and `aes_package.vhd` files to the project by clicking the **Add Source** button and navigating to the `source\design` directory.
8. Uncheck **Copy to Project** for all sources and then click **Next** (see Figure 2-16).



X1135_c1_22_040710

Figure 2-16: **New Project Wizard (Add Existing Sources)**

9. Click **Finish** (see Figure 2-17).

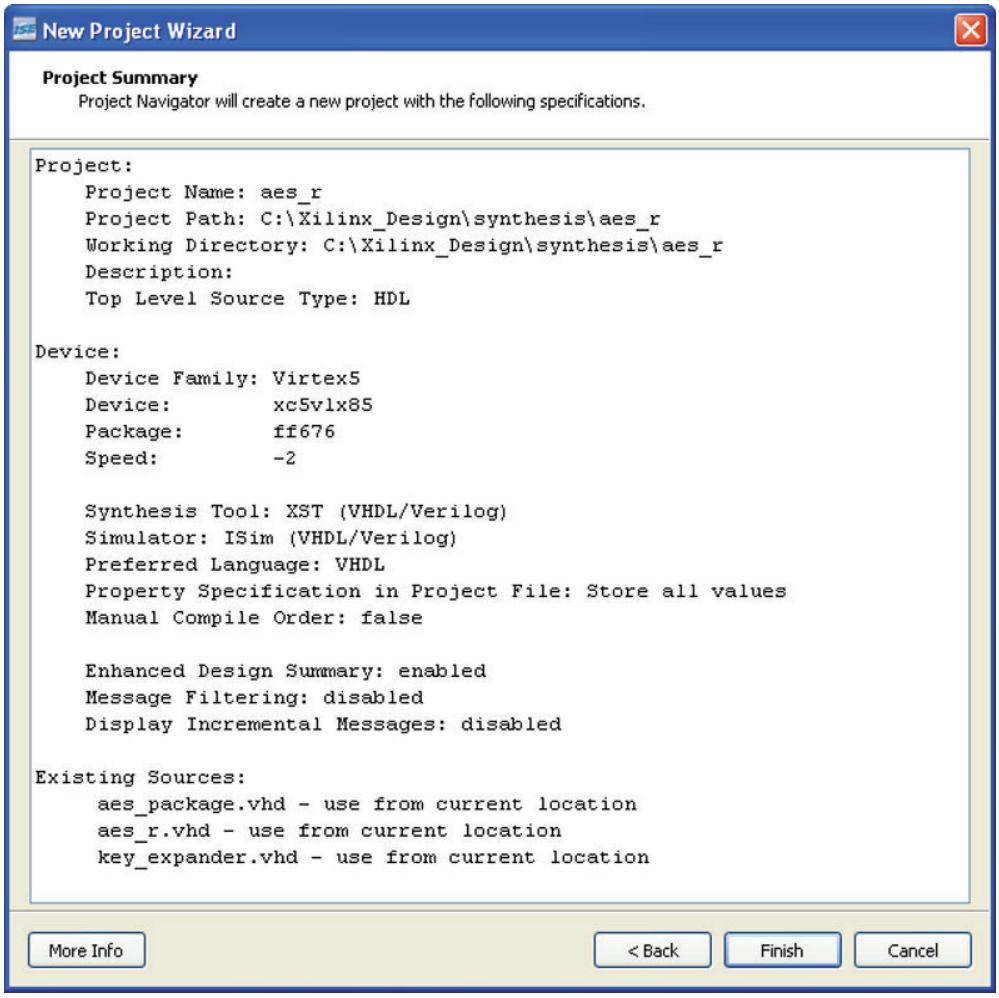
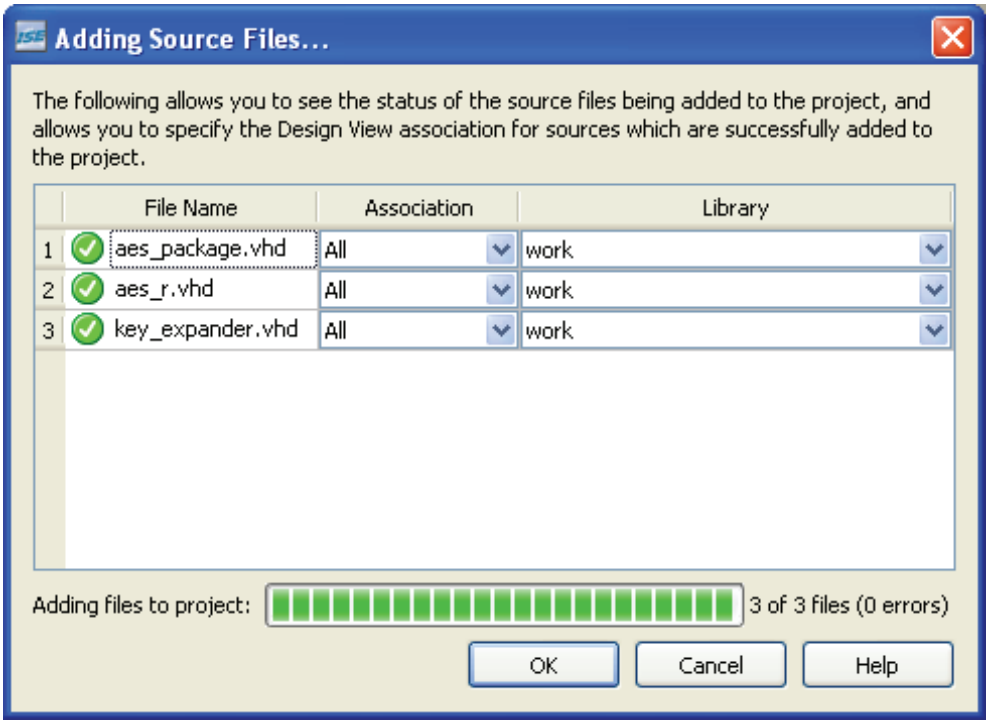


Figure 2-17: **New Project Wizard (Project Summary)**

10. Click **OK** (see Figure 2-18).



X1135_c1_24_040710

Figure 2-18: **Adding Source Files**

The ISE Project Navigator Window should look like the window shown in Figure 2-19.

Note: All modules are defined. There should be no question marks (?) by any module. All modules at this level and below are in context.

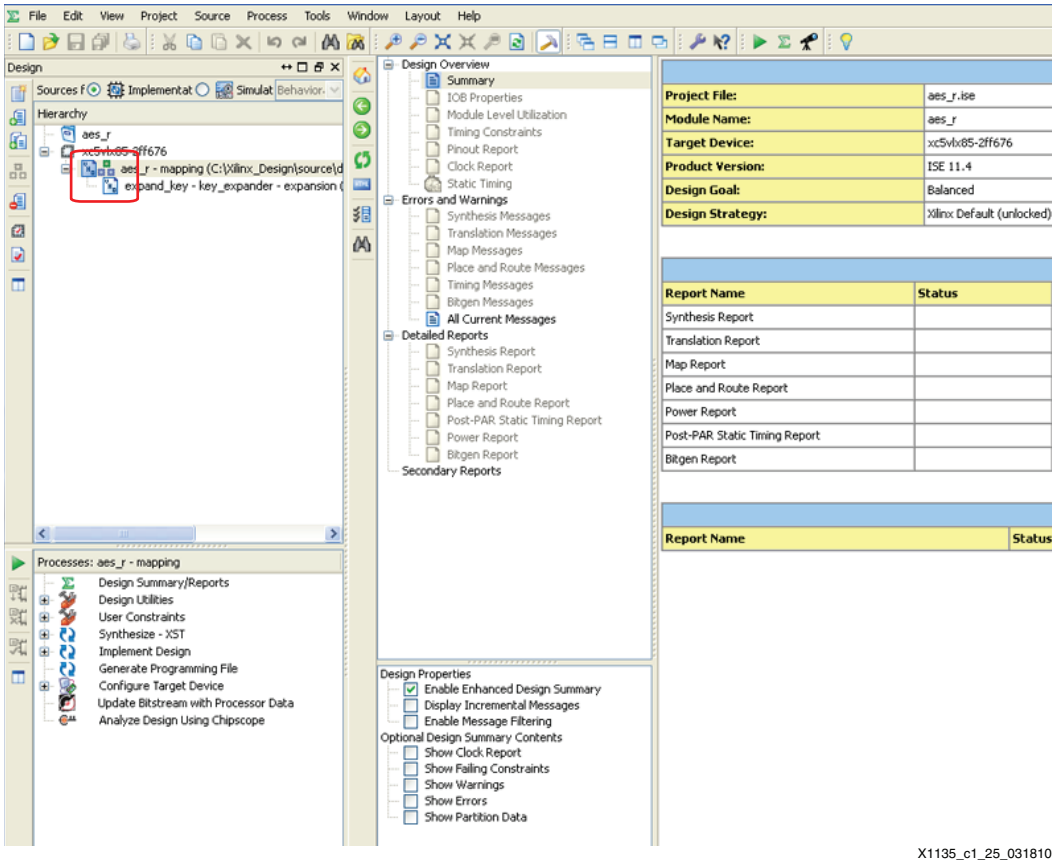


Figure 2-19: **Project Navigator Window**

11. Open the `aes_r.vhd` file and locate the section containing the buffer_type attributes:

```
attribute buffer_type: string;
attribute buffer_type of clk         : signal is "none";
attribute buffer_type of reset       : signal is "none";
attribute buffer_type of push_button : signal is "none";
attribute buffer_type of start       : signal is "none";
attribute buffer_type of mode        : signal is "none";
attribute buffer_type of load        : signal is "none";
attribute buffer_type of key         : signal is "none";
attribute buffer_type of data_in     : signal is "none";
attribute buffer_type of data_out    : signal is "none";
attribute buffer_type of done        : signal is "none";
```

The buffer_type attribute directs the XST tool to disable I/O insertion on the specified ports. By default, XST inserts an I/O on all ports. The buffer_type attribute is necessary to prevent this insertion at this level in the hierarchy either because the I/Os are placed at the top level (such as CLK) or the port is a direct connection to a port of another instance.

Note: The clk I/O is inferred when `top` is synthesized. All other ports are internal to the FPGA; therefore they have the attribute **`signal is "none"`** applied.

12. Right-click the Synthesize-XST icon in the Processes window, select **Process Properties…**.
13. Set Optimization Goal to **Area**, set Optimization Effort to **High**, and click **OK** (see Figure 2-20).

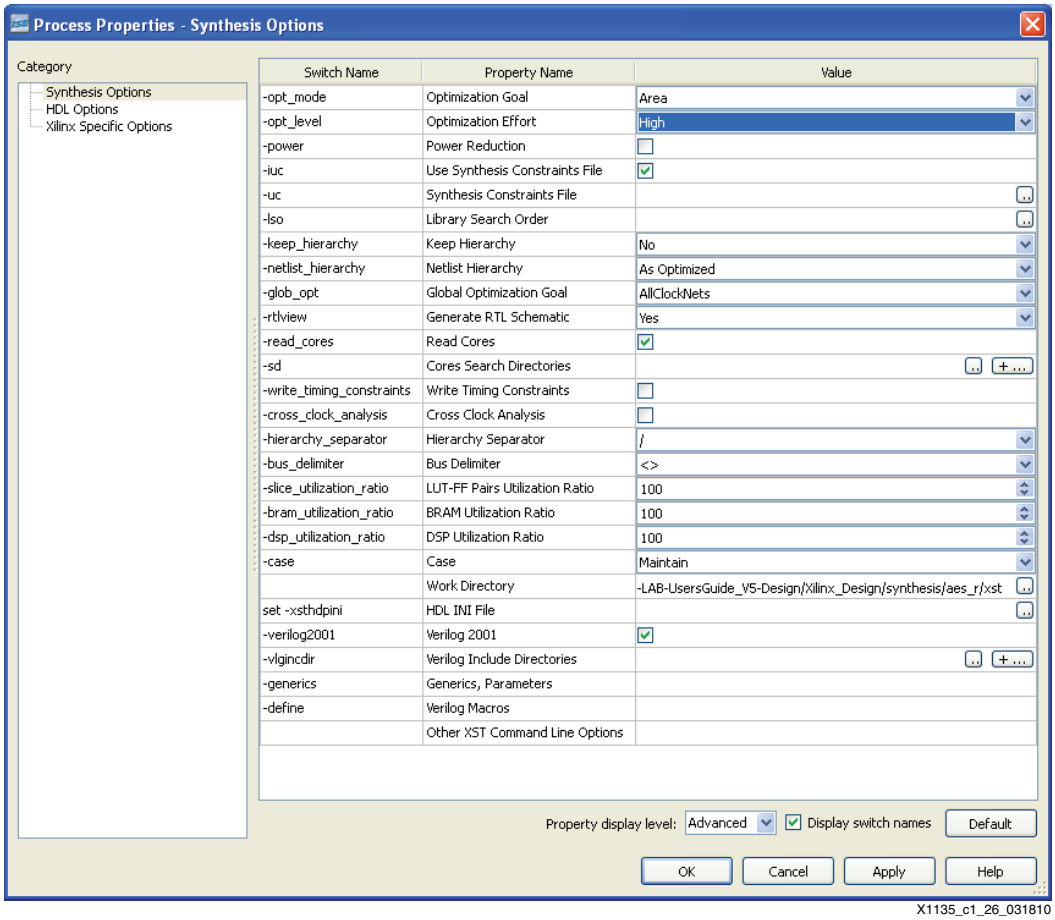


Figure 2-20: **Process Properties**

14. To run XST synthesis, either right-click on the **Synthesize-XST** icon in the Processes window and select **Run** or simply double-click **Synthesize-XST**.
15. After synthesis is complete, close the aes_r project.

Synthesize the `compare` Module

These steps describe how to synthesize the `compare` module:

1. Start the ISE 11.4 software:

   Start → **All Programs** → **Xilinx ISE Design Suite 11** → **ISE** → **Project Navigator**

2. Create a new ISE 11.4 project:

   File → **New Project**

3. Set the Project Location to **`\Xilinx_Design\synthesis`**, set the Project Name to **compare**, and click **Next** (see Figure 2-21).

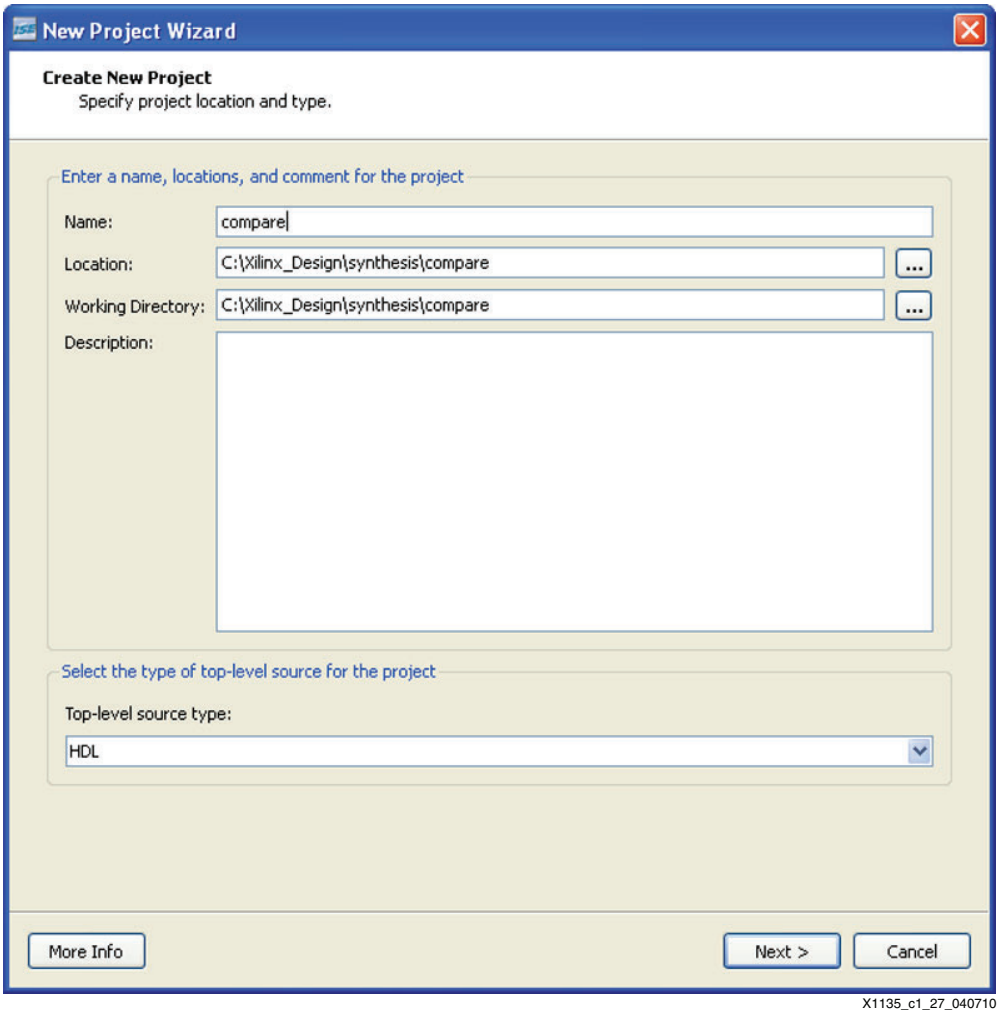


Figure 2-21: **New Project Wizard (Create New Project)**

4. Choose an XC5VLX85-FF676-2 as the target device (see Figure 2-22).

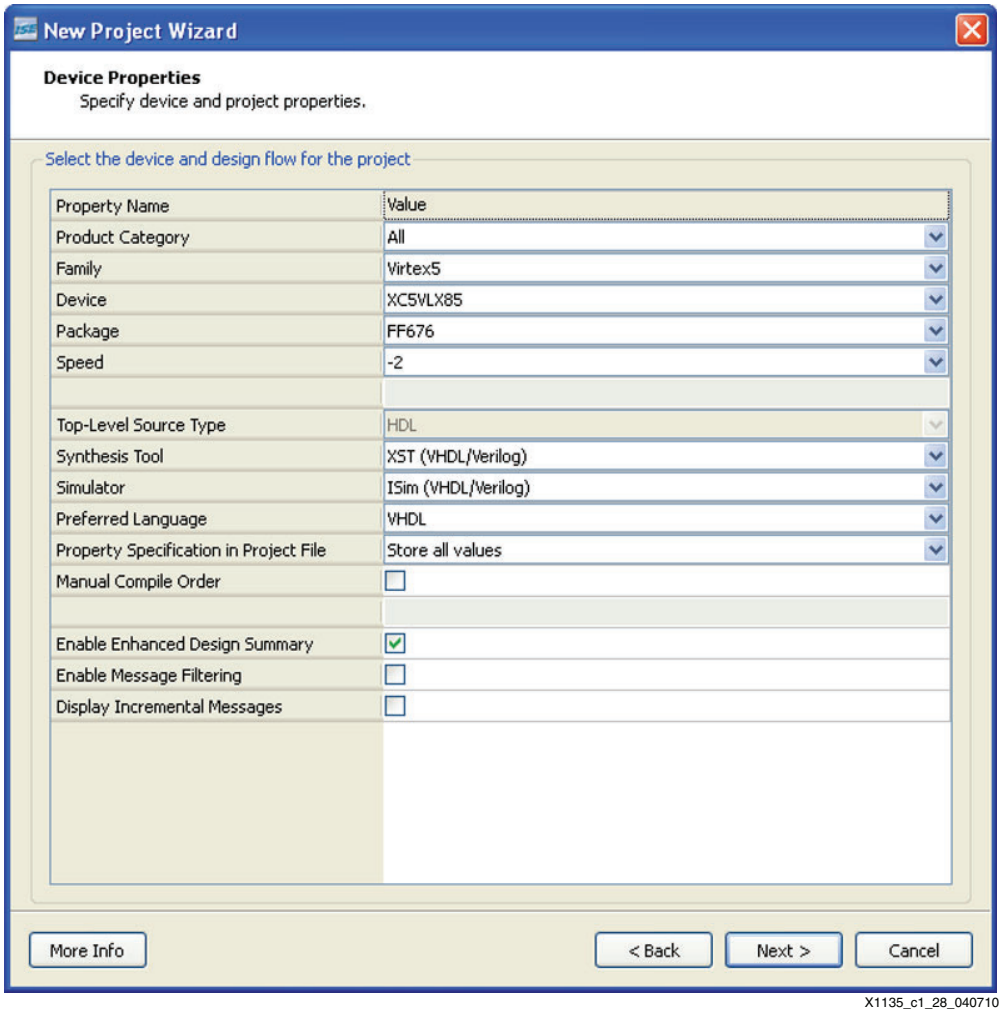
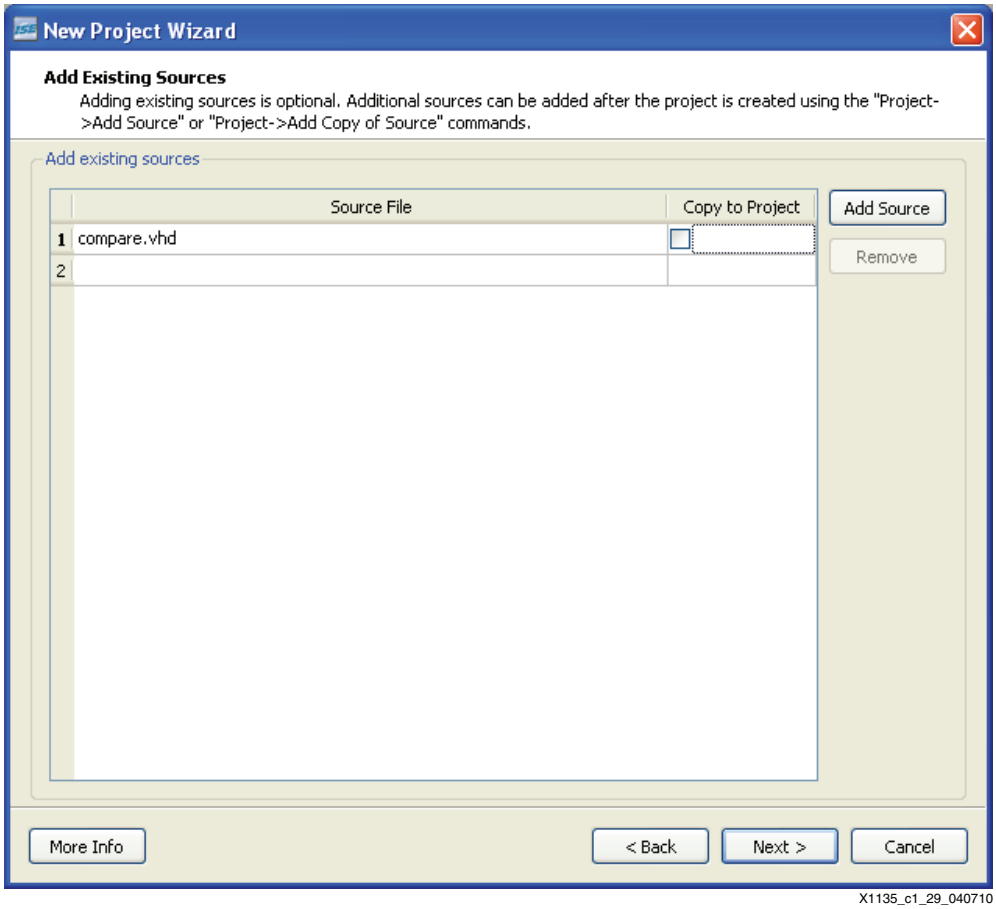


Figure 2-22: **New Project Wizard (Device Properties)**

5. Click **Next** twice. Add the `compare.vhd` file to the project by clicking the **Add Source** button and navigating to the `source\design` directory.
6. Uncheck **Copy to Project** and click **Next** (see Figure 2-23).



X1135_c1_29_040710

Figure 2-23: **New Project Wizard (Add Existing Sources)**

7. Click **Finish** (see Figure 2-24).

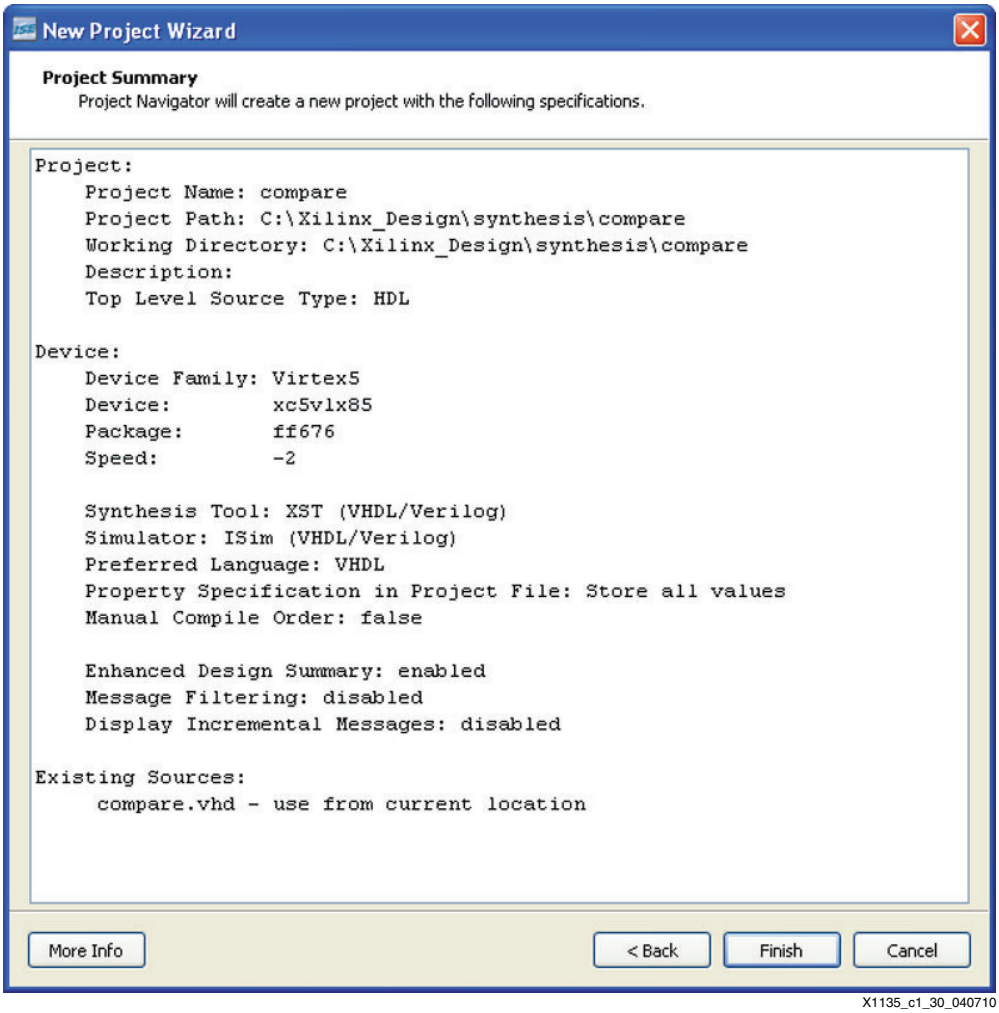


Figure 2-24: **New Project Wizard (Project Summary)**

8. Click **OK** (see Figure 2-25).

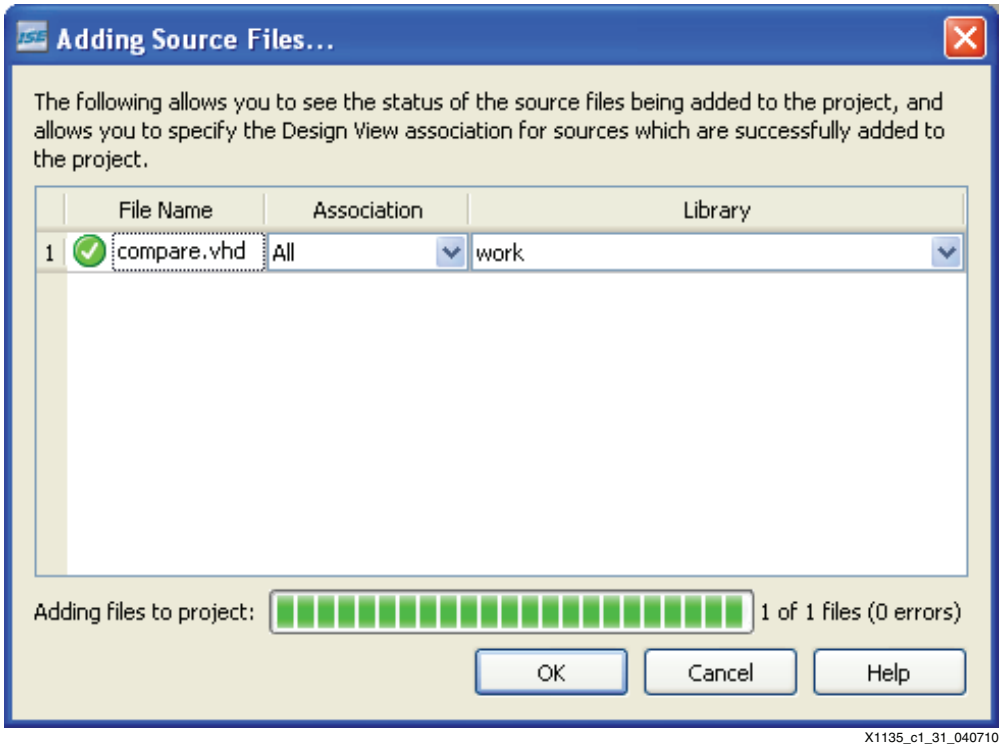
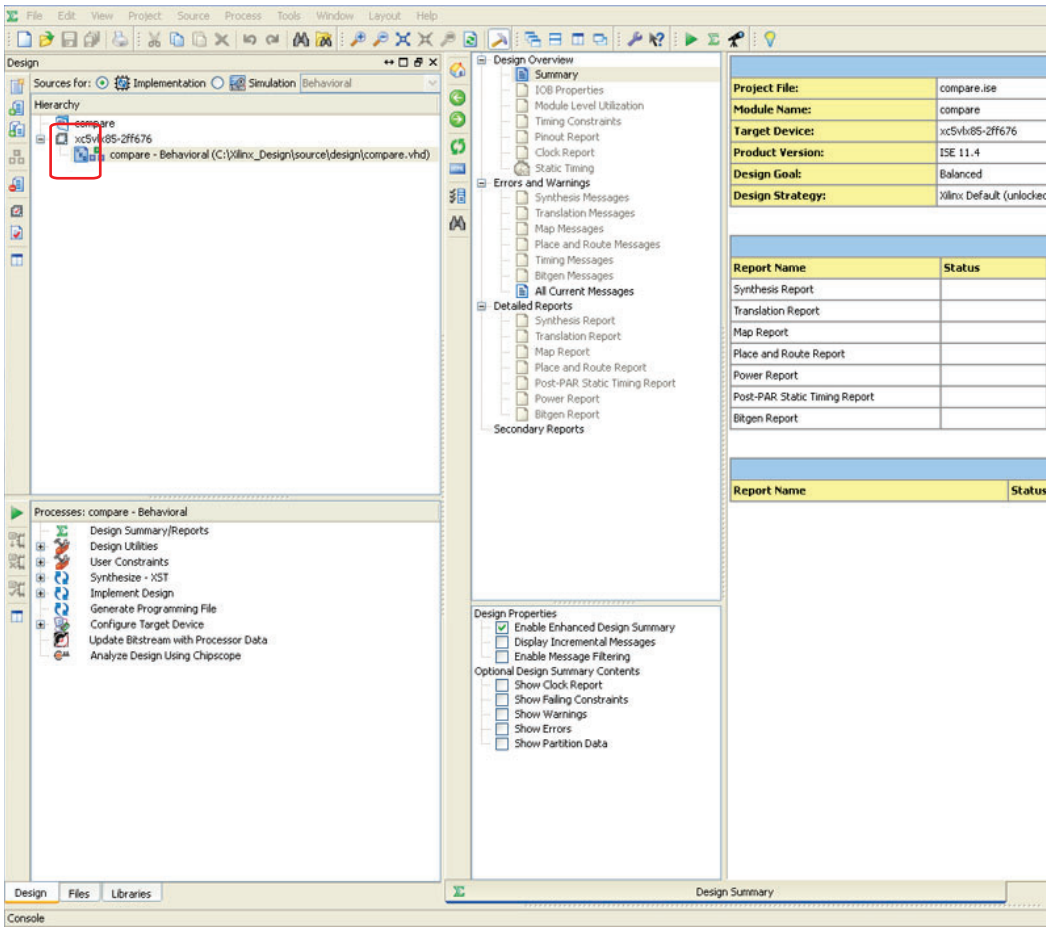


Figure 2-25: **Adding Source Files**

The ISE Project Navigator Window should look like the window shown in Figure 2-26.

Note: All modules are defined. There should be no question marks (?) by any module. All modules at this level and below are in context.



X1135_c1_32_031810

Figure 2-26: **Project Navigator Window**

9. Open the `compare.vhd` file and locate the buffer attributes:

```
attribute buffer_type: string;
attribute buffer_type of clk          : signal is "none";
attribute buffer_type of reset        : signal is "none";
attribute buffer_type of reset_out    : signal is "none";
attribute buffer_type of done1        : signal is "none";
attribute buffer_type of done2        : signal is "none";
attribute buffer_type of start        : signal is "none";
attribute buffer_type of load         : signal is "none";
attribute buffer_type of data_in1     : signal is "none";
attribute buffer_type of data_in2     : signal is "none";
```

The buffer_type attribute directs the XST tool to disable I/O insertion on the specified ports. By default, XST inserts an I/O on all ports. The buffer_type attribute is necessary to prevent this insertion at this level in the hierarchy either because the I/Os are placed at the top level (such as CLK) or because the port is a direct connection to a port of another instance.

Note: The led signal does not have an attribute associated with it because the led signal is driven directly by an output buffer from within the compare code. All other ports are either ports within the FPGA or had their I/Os inferred at a different level; therefore they have the attribute **`signal is "none"`** applied.

10. In the `compare.vhd` source code file, the LUT instantiation between certain inputs and outputs is necessary for Trusted Routing rules. Refer to XAPP1134, *Developing Secure Designs Using the Virtex-5 Family,* for more details on Trusted Routing rules. A section of the VHDL code for the LUT instantiation is shown here:

```
-- Instantiate LUT buffers on nets that either drive two different regions
-- or are feedthrough's from one region to the next. This prevents a single
-- net being placed "shorting" three regions together. Trusted Bus Macros
-- fulfilled this requirement automatically.

  lut_reset_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => reset_i, O => reset_out );

  lut_start_aes1_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => start_i, O => start_aes1 );

  lut_start_aes2_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => start_i, O => start_aes2 );

  lut_load_aes1_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => load_i, O => load_aes1 );

  lut_load_aes2_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => load_i, O => load_aes2 );
```

11. Right-click on the Synthesize-XST icon in the Processes window and select **Process Properties** (see Figure 2-27).

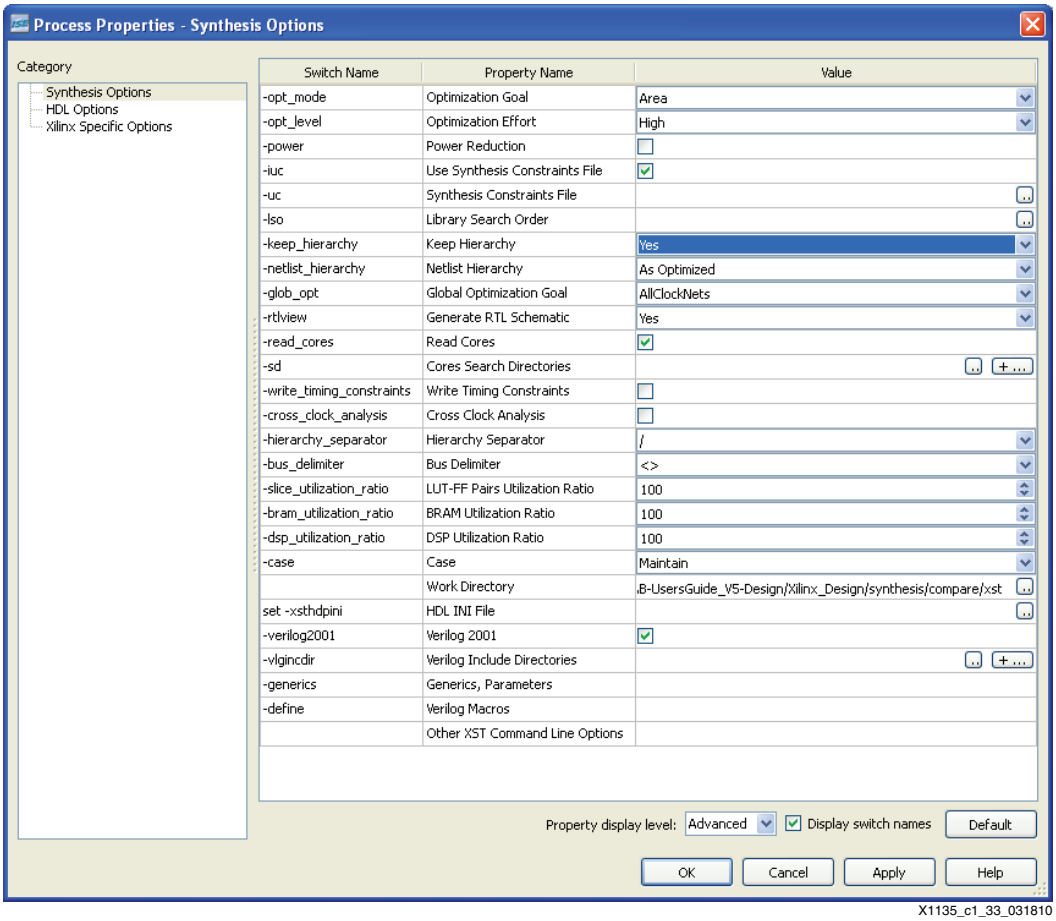


Figure 2-27: **Process Properties**

12. Set the Optimization Goal to **Area** and the Optimization Effort to **High**.
13. Set the Keep Hierarchy option to **Yes**.

    Note: This extra step is different than the `SCC_LAB_TOP`, `aes`, and `aes_r` blocks. Compare, in this lab, is an isolated module, not a PR module. Keeping its hierarchy prevents it from being flattened with the top level or other isolated blocks. This only keeps its top-level instance, not any sub-hierarchy. As such, it has no additional impact to optimization.
14. Click **OK**.
15. To run XST synthesis, either right-click on the Synthesis-XST icon in the Processes window and select **Run** or simply double-click **Synthesize-XST**.
16. After synthesis is complete, close the `compare` project.

Chapter 3

Implementing the System

PlanAhead Project Entry

Launch the PlanAhead Tool

To launch the 11.4 version of the PlanAhead™ tool, select:

Start → **All Programs** → **Xilinx ISE Design Suite 11** → **PlanAhead** → **Xilinx PlanAhead**

PlanAhead Project Creation

The PlanAhead tool works with any synthesized netlist (XST, Synplify, etc.). The regular guidelines are followed to generate a new project and import the netlist into the PlanAhead tool to create a floorplan for the design.

1. Set up a new PlanAhead project:

   File → **New Project**

2. Click **Next** and enter:
   - Project name: For this lab, the **Floorplan_SCC** project name is used.
   - Project location: `\Xilinx_Design\PlanAhead`

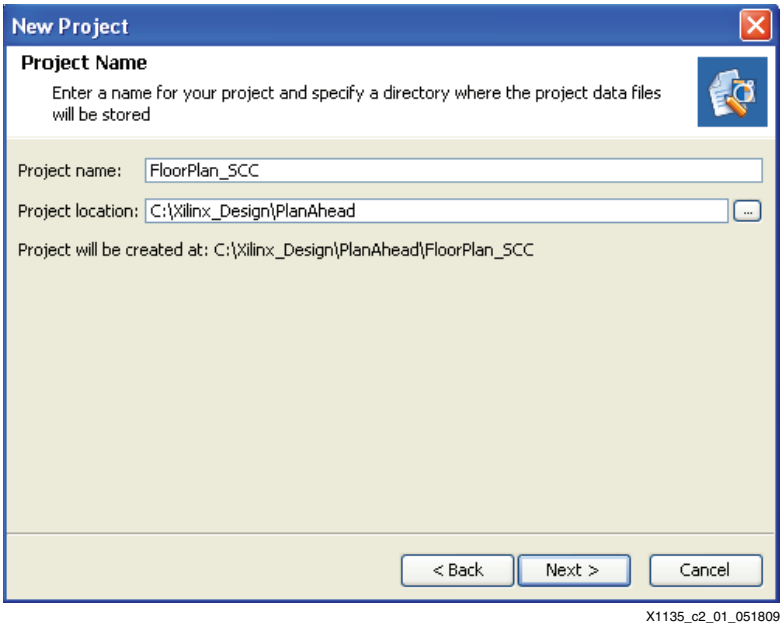


Figure 3-1: **New Project**

3. Click **Next**.
4. Select **Import a synthesized (EDIF or NGC) netlist** (see Figure 3-2) because the modules have already been synthesized and click **Next**.

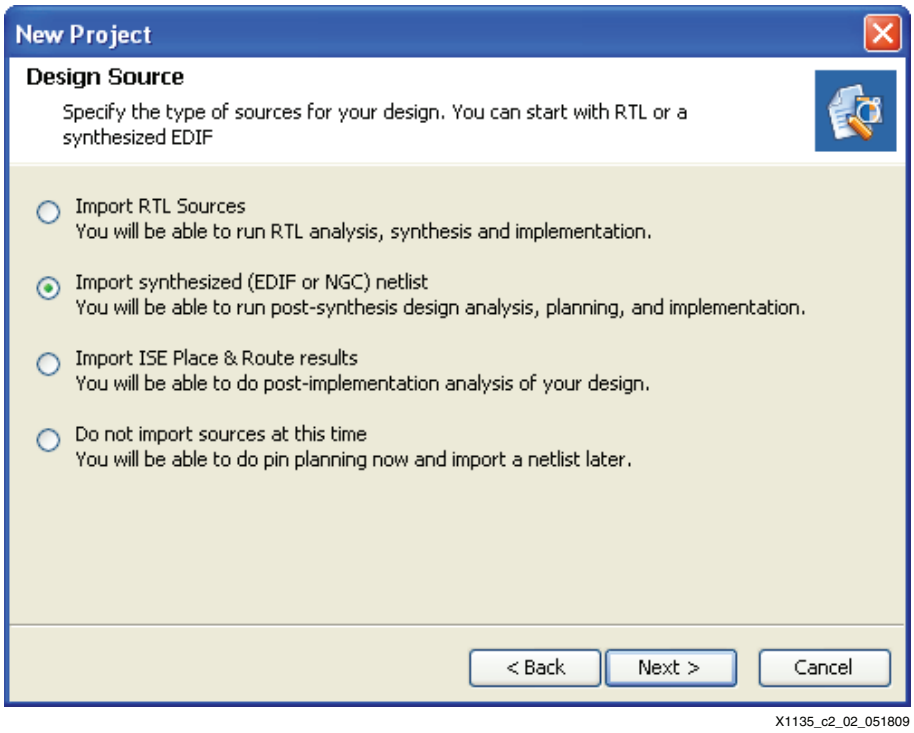


Figure 3-2: **New Project (Design Source)**

5. Import the previously generated top level Netlist (see Figure 3-3).

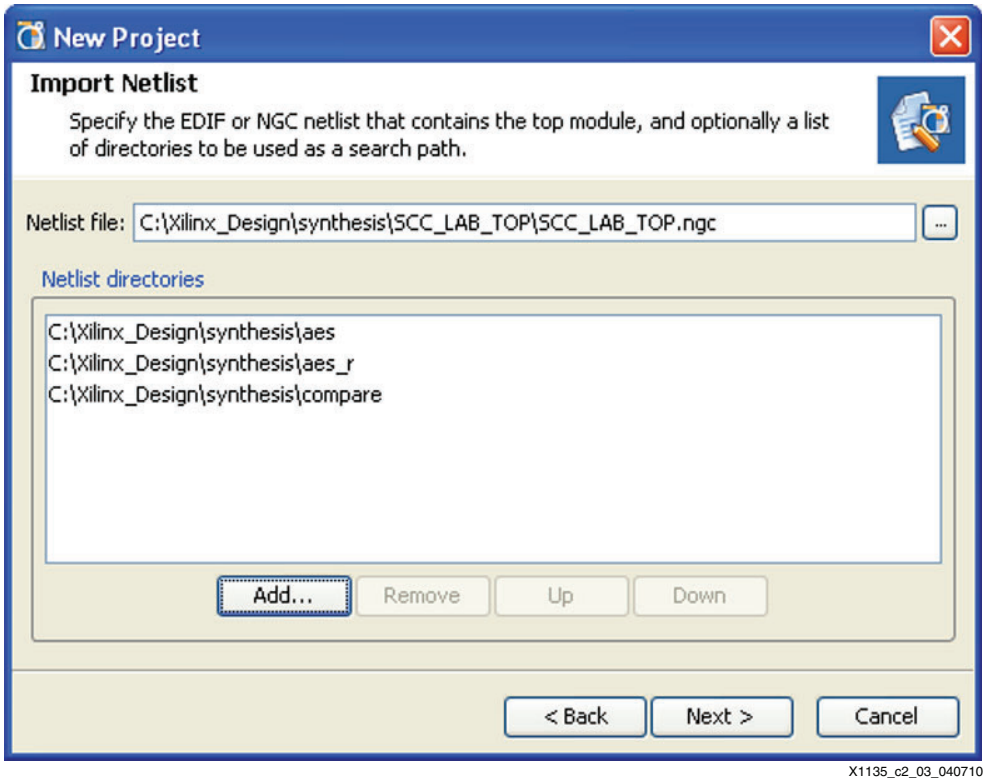


Figure 3-3: **New Project (Import Netlist)**

6. Designate the top-level netlist and library directories. The netlist directories should include the static logic `top.ngc` and only one "version" of each partially reconfigurable (PR) or isolated (ISO) module.

   Netlist File:

   ```
   ..\Xilinx_Design\synthesis\top\top.ngc
   ```

   Netlist Directories:

   ```
   ..\Xilinx_Design\synthesis\aes
   ..\Xilinx_Design\synthesis\aes_r
   ..\Xilinx_Design\synthesis\compare
   ```

7. Click **Next**.

8. Choose the Product Family (see Figure 3-4). For this lab, **Virtex5** is used.

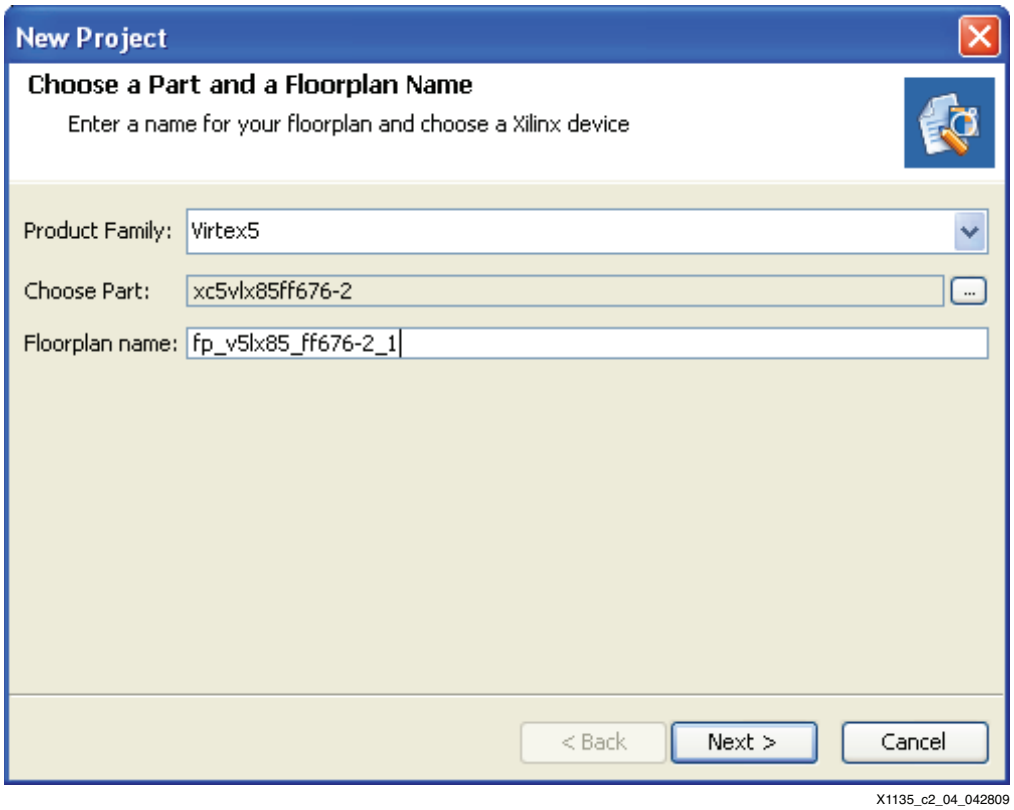
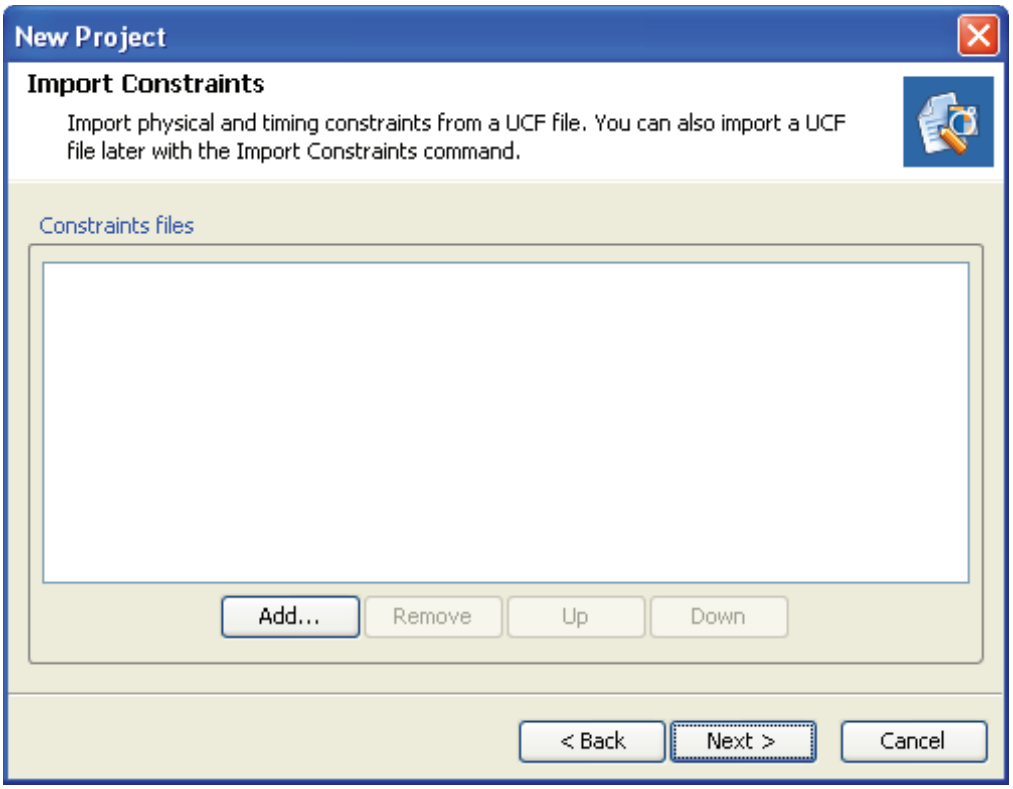


Figure 3-4: **New Project (Product Family and Floorplan)**

9. Choose the device:
   a. Click the '**...**' icon to the right of the Choose Part field.
   b. Navigate to the **LX85** device.
   c. Select the **ff676** package
   d. Select the **-2** speed grade
10. Set the Floorplan name to **fp_v5lx85_ff676-2_1**.
11. Click **Next**.
12. This lab creates the final UCF from scratch. Thus no UCF files need to be imported.



X1135_c2_05_042809

Figure 3-5: **New Project (Import Constraints)**

13. Click **Next** and **Finish**.

14. The PlanAhead project now looks as shown in Figure 3-6.

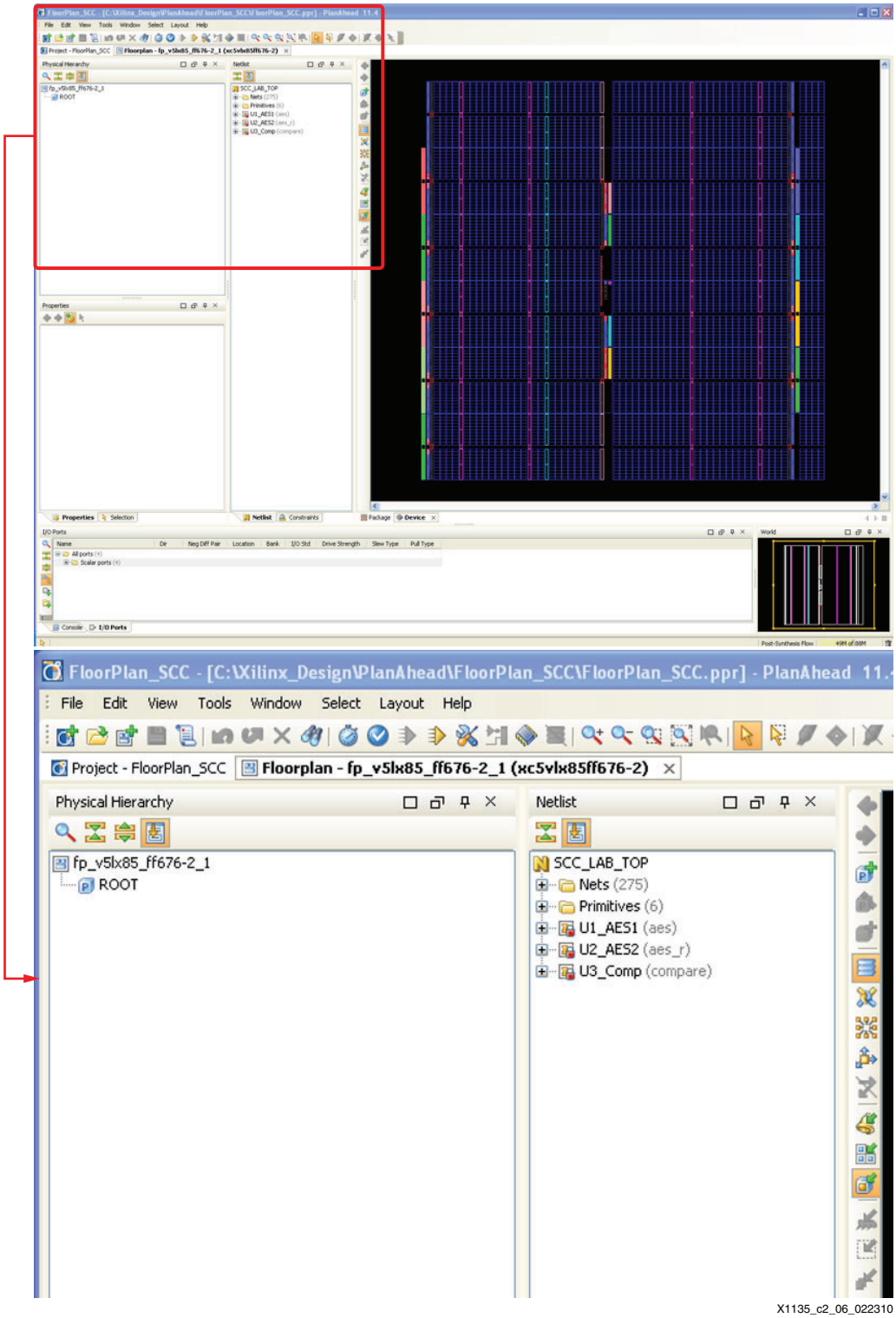


Figure 3-6: **PlanAhead Project View**

15. To turn the project into a PR project, select **File** → **Set PR Project**. When the **Set PR Project** option is selected, it is grayed out to indicate this step has been completed.

Place I/Os with the PlanAhead Tool

When placing I/Os it is imperative to consider the physical location of I/Os in relation to the logic regions they interface with. For example, the clock input can be placed anywhere because it does not have to be isolated. Because the led pin is part of the COMPARE logic, it needs to be physically placed in that region. Similarly, the reset and push_button pins are owned by the AES logic, so they need to be physically placed inside that region.

To place the PlanAhead tool in Site Constraint Mode:

1. Select the symbol shown in Figure 3-7 that is on the vertical shortcut bar between the netlist window and the device window.

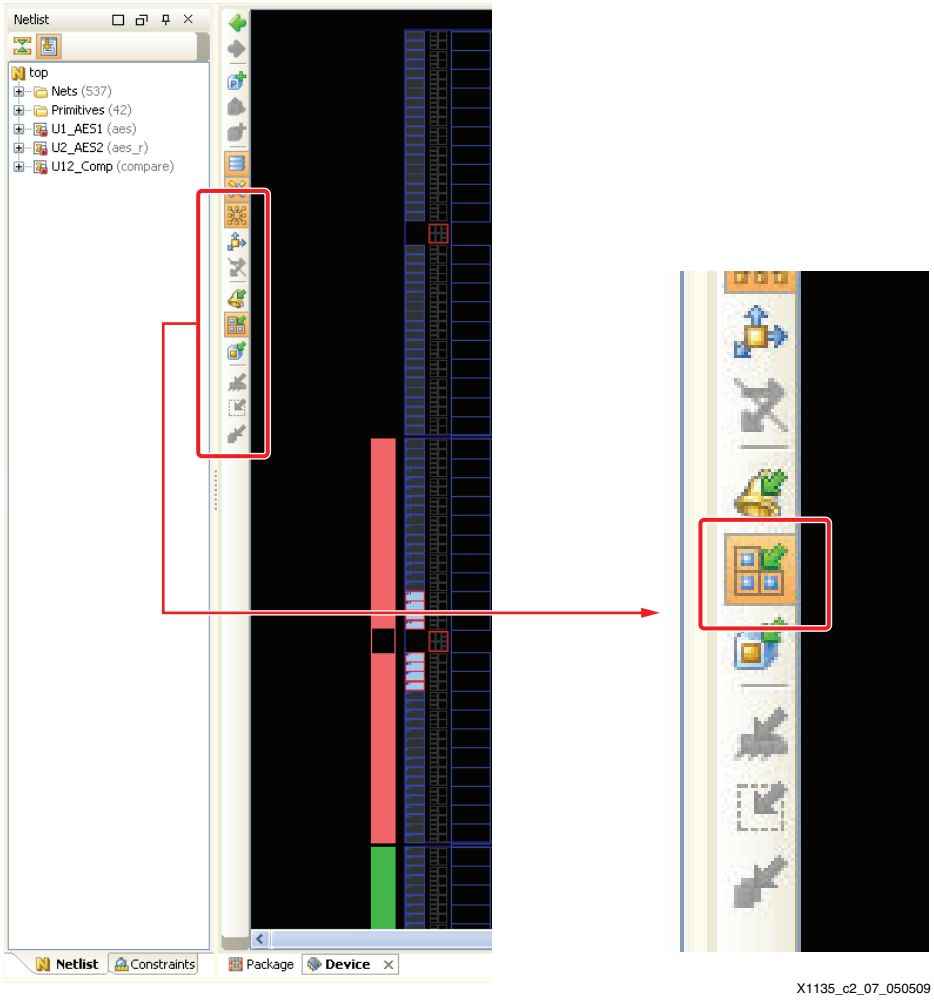


Figure 3-7: **Select Site Constraint Mode Symbol**

2. Select **Edit** → **Find** (shortcut: **Ctrl-F**).

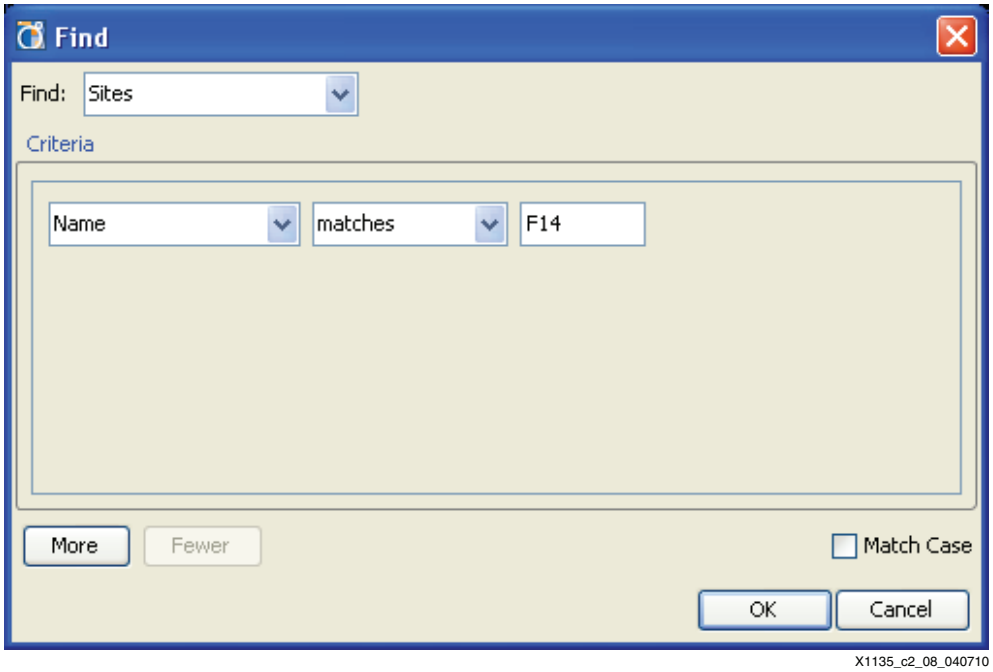
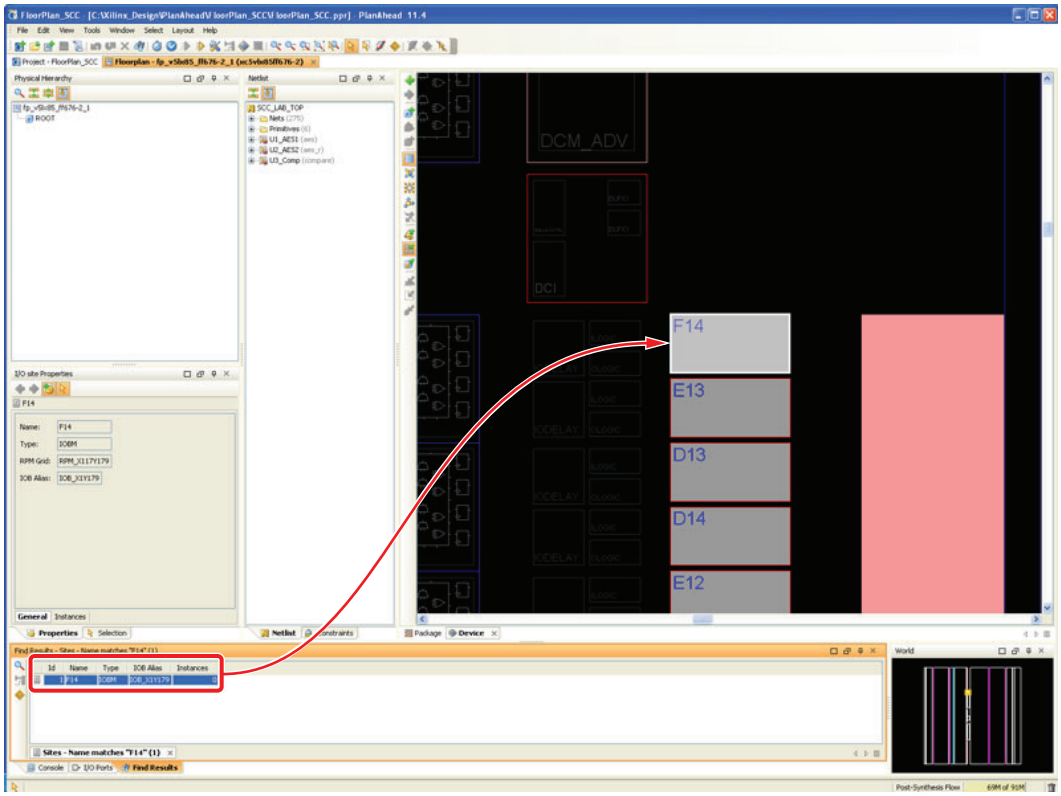


Figure 3-8: **Find**

3. Select **Sites** from the Find pull-down menu.
4. Select **Name** and **matches** from the two pull-down menus under Criteria.
5. Type **F14** in the remaining field box and click **OK**.

   The search results appear on a tab in the Find Results section at the bottom left of the PlanAhead tool window.

6. Select the result and press **F9** to zoom to the current selection, as shown in Figure 3-9. Alternatively, select **View** → **Fit Selection** from the top menu bar to perform the same task.



X1135_c2_09_031810

Figure 3-9: **Find Results**

7. Select **Edit** → **Find** (shortcut: **Ctrl-F**).

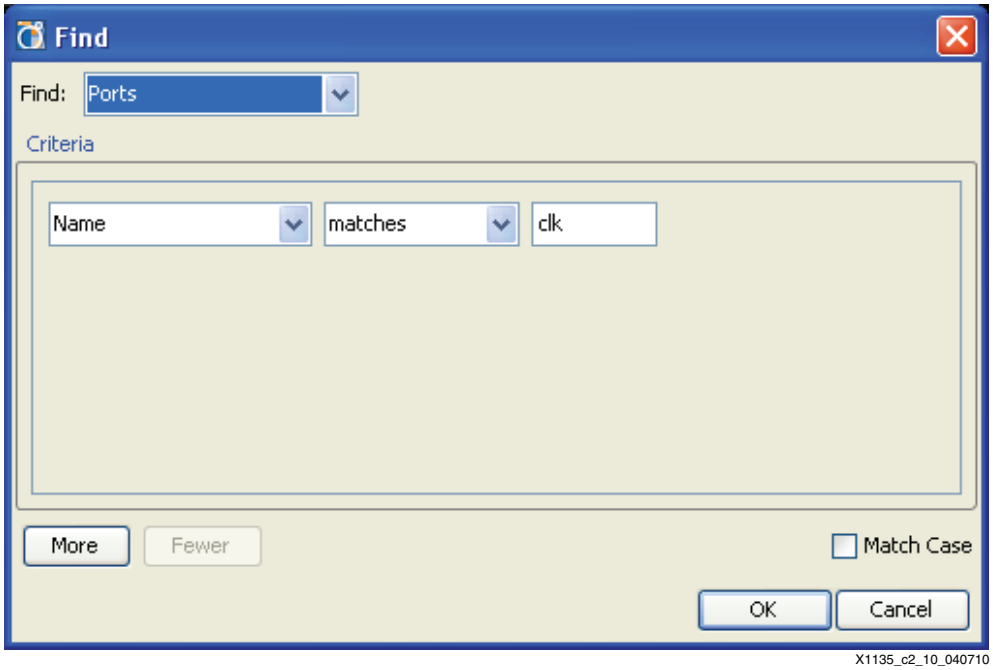
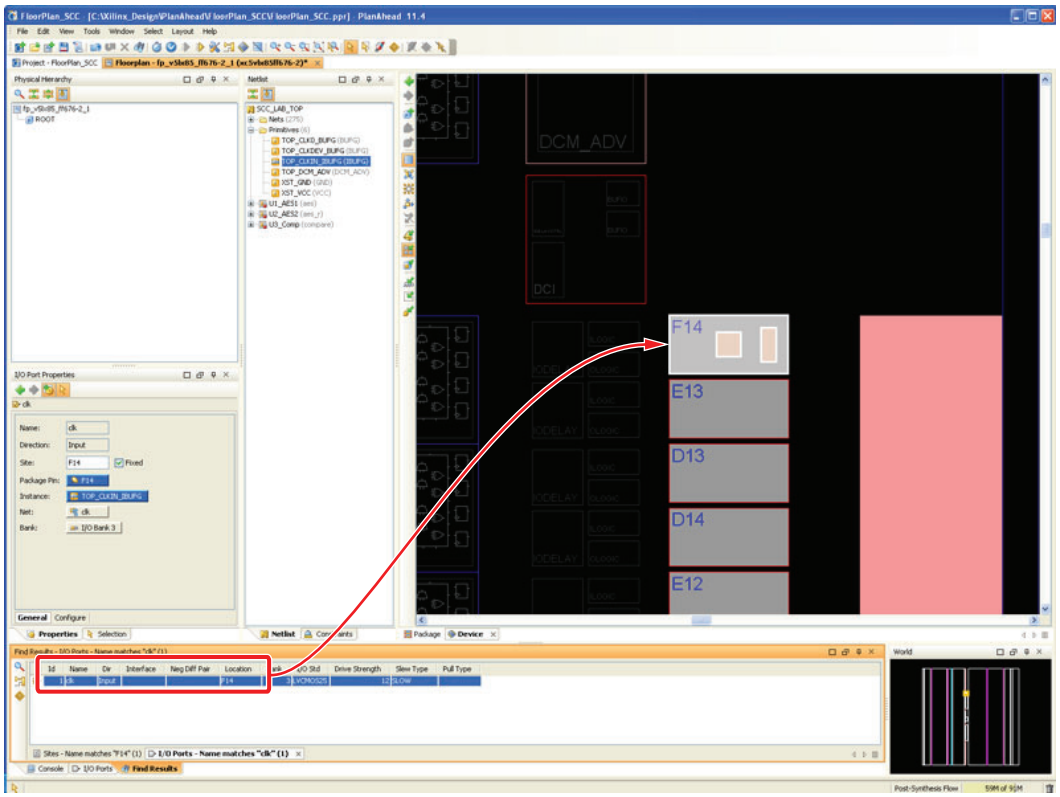


Figure 3-10: **Find**

8. Select **Ports** from the Find: pull-down menu.
9. Select **Name** and **matches** from the two pull-down menus under Criteria.
10. Type **clk** in the field box, and click **OK**. The search results appear on a tab in the Find Results section at the bottom left of the PlanAhead tool window.
11. Click and drag the result **clk** to the site identified in step 2 through step 6, as shown in Figure 3-11.



X1135_c2_11_031810

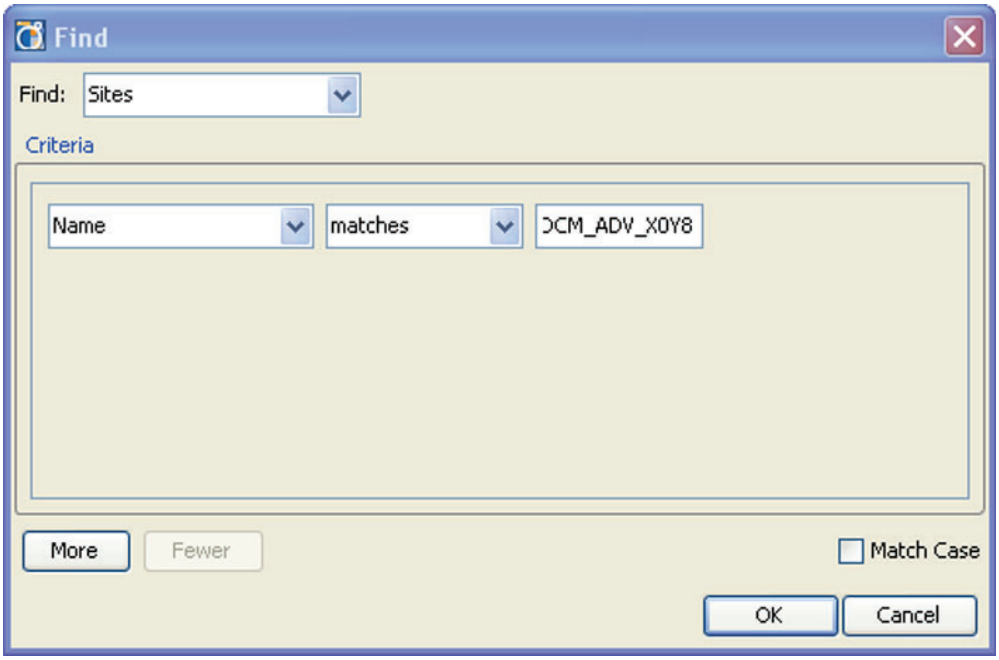
Figure 3-11: **Drag clk**

12. Repeat step 1 to step 10 to place “reset” at D11.
13. Repeat step 1 to step 10 to place “push_button” at D10.
14. Repeat step 1 to step 10 to place “led” at P6.

Place DCM with the PlanAhead Tool

These steps describe how to place the DCM using the PlanAhead tool:

1. Select **Edit** → **Find** (shortcut: **Ctrl-F**) and select **Sites** from the Find: pull-down menu.
2. Select **Name** and **matches** from the two pull-down menus under Criteria.



X1135_c2_12_040710

Figure 3-12: **Find**

3. Type **DCM_ADV_X0Y8** in the field box, and click **OK**.

   The search results appear on a tab in the Find Results section at the bottom left of the PlanAhead tool window.

4. Select this result and press **F9** to zoom to the current selection. The PlanAhead tool then zooms to the selected location (see Figure 3-13). Alternatively, select **View** → **Fit Selection** from the top menu bar to perform the same task.

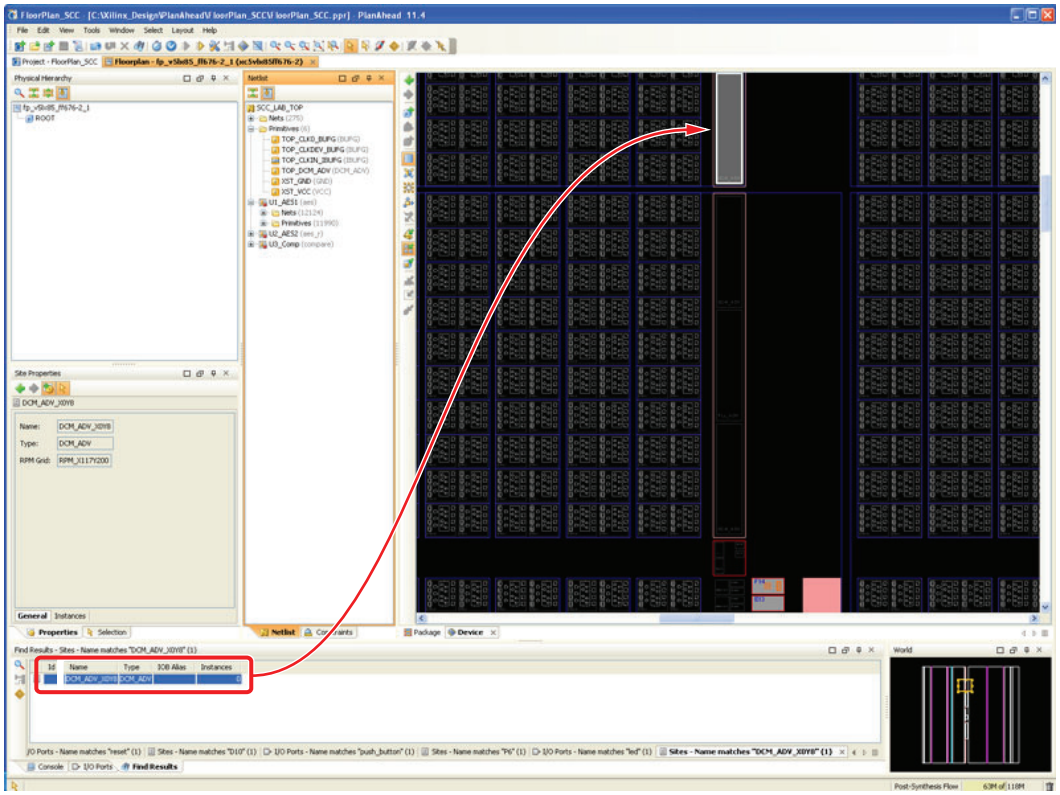


Figure 3-13: **Search Results**

5. In the Netlist window, expand **Primitives**, then drag **TOP_DCM_ADV** to the DCM_ADV_X0Y8 box as shown in Figure 3-14.

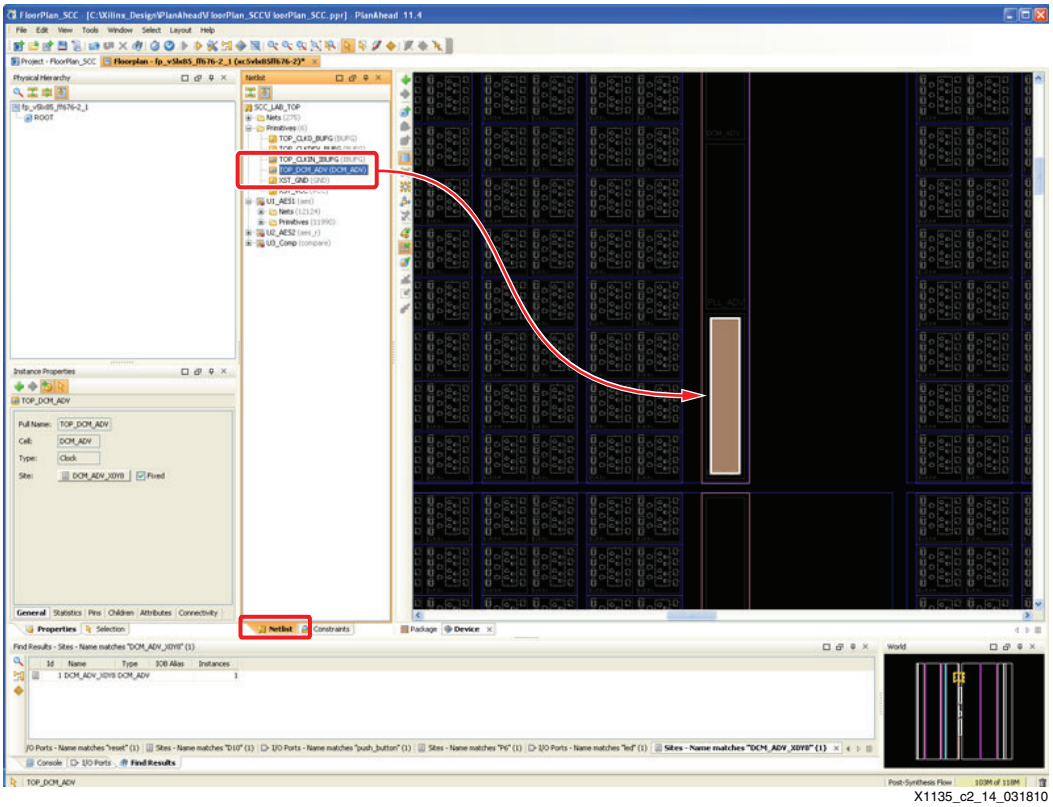


Figure 3-14: **DCM_ADV_X0Y8**

6. Repeat step 1 through step 5 to place **TOP_CLK0_BUFG** at BUFGCTRL_X0Y31.
7. Repeat step 1 through step 5 to place the **TOP_CLKDEV_BUFG** at BUFGCTRL_X0Y30.

Setting Up Timing Constraints with the PlanAhead Tool

This section shows that timing constraints can be applied to each module and the top-level design easily with the PlanAhead tool.

1. Select the **Constraints** tab in the NETLIST window, and click the **Create New Constraint** button located at the top of the Netlist / Constraints window (see Figure 3-15).

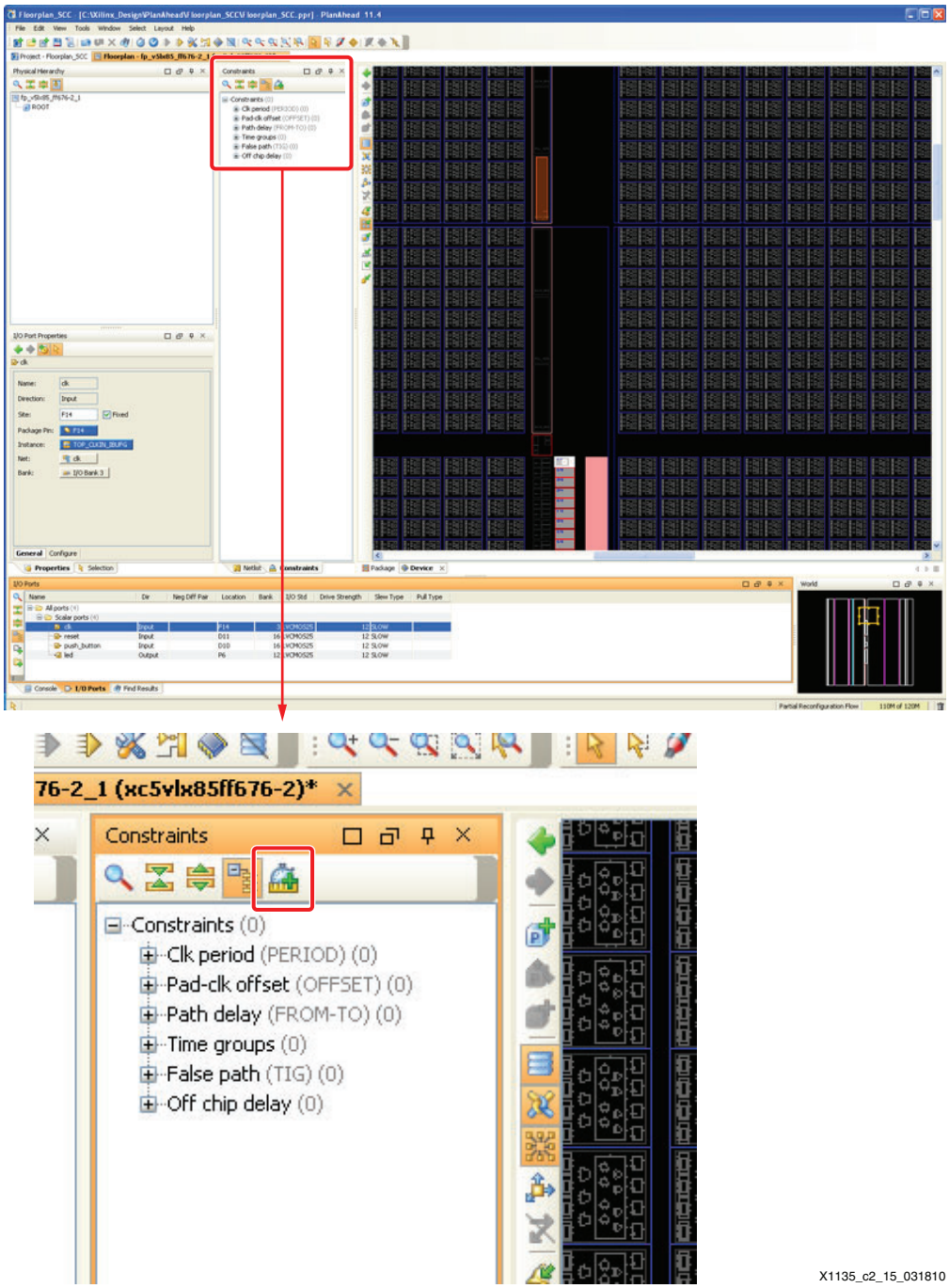
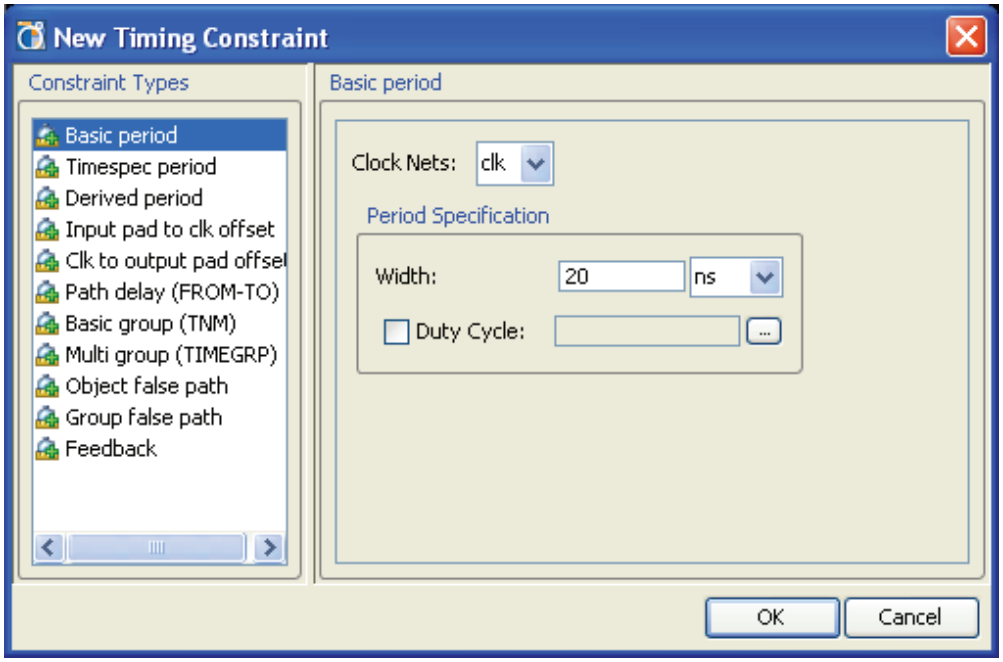


Figure 3-15: **Create New Constraint**

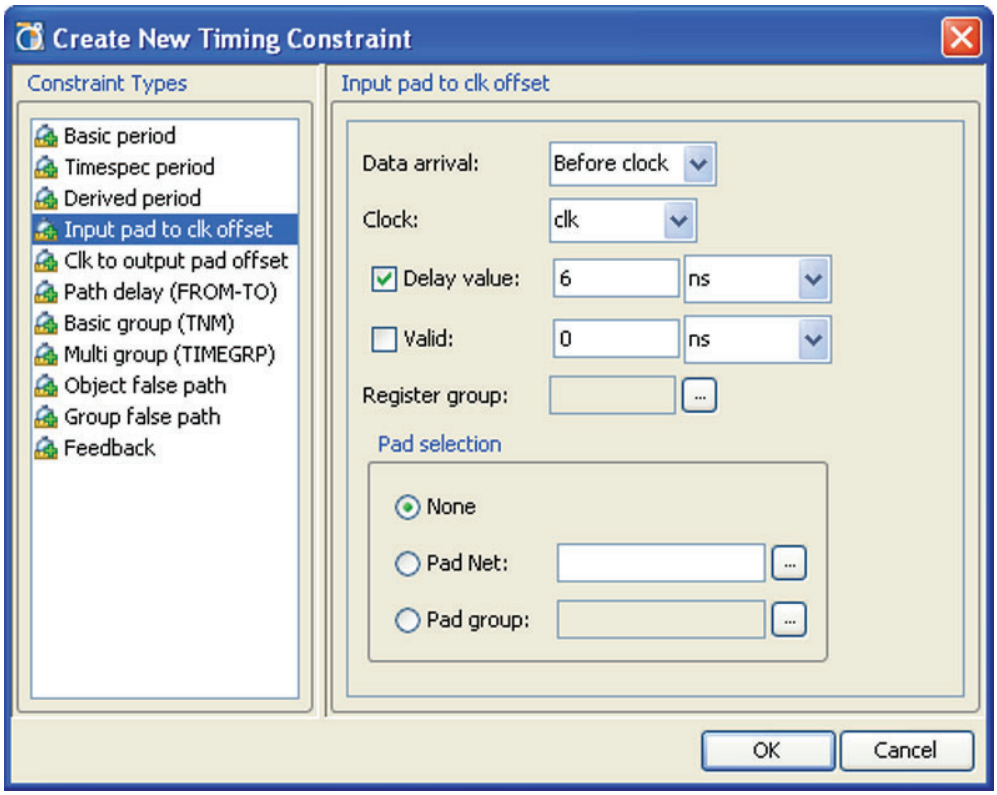
2. Add a basic period constraint of 20 ns to the design by entering the value as shown in Figure 3-16.



X1135_c2_16_040710

Figure 3-16: **New Timing Constraint**

3. Click **OK**.
4. Check the Delay Value box and add a Global Input constraint of 6 ns to the design by entering the value as shown in Figure 3-17.

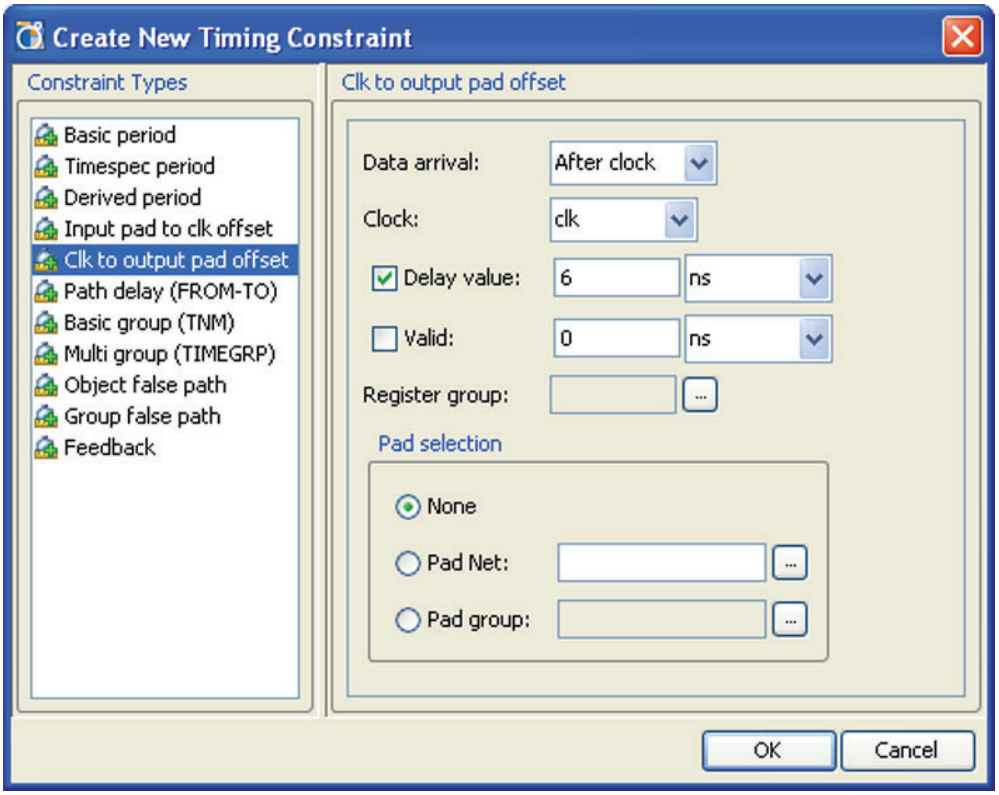


X1135_c2_17_040710

Figure 3-17: **New Timing Constraint**

5. Click **OK**.

6. Check the Delay Value box and add a Global Output constraint of 6 ns to the design by entering the value as shown in Figure 3-18.



X1135_c2_18_040710

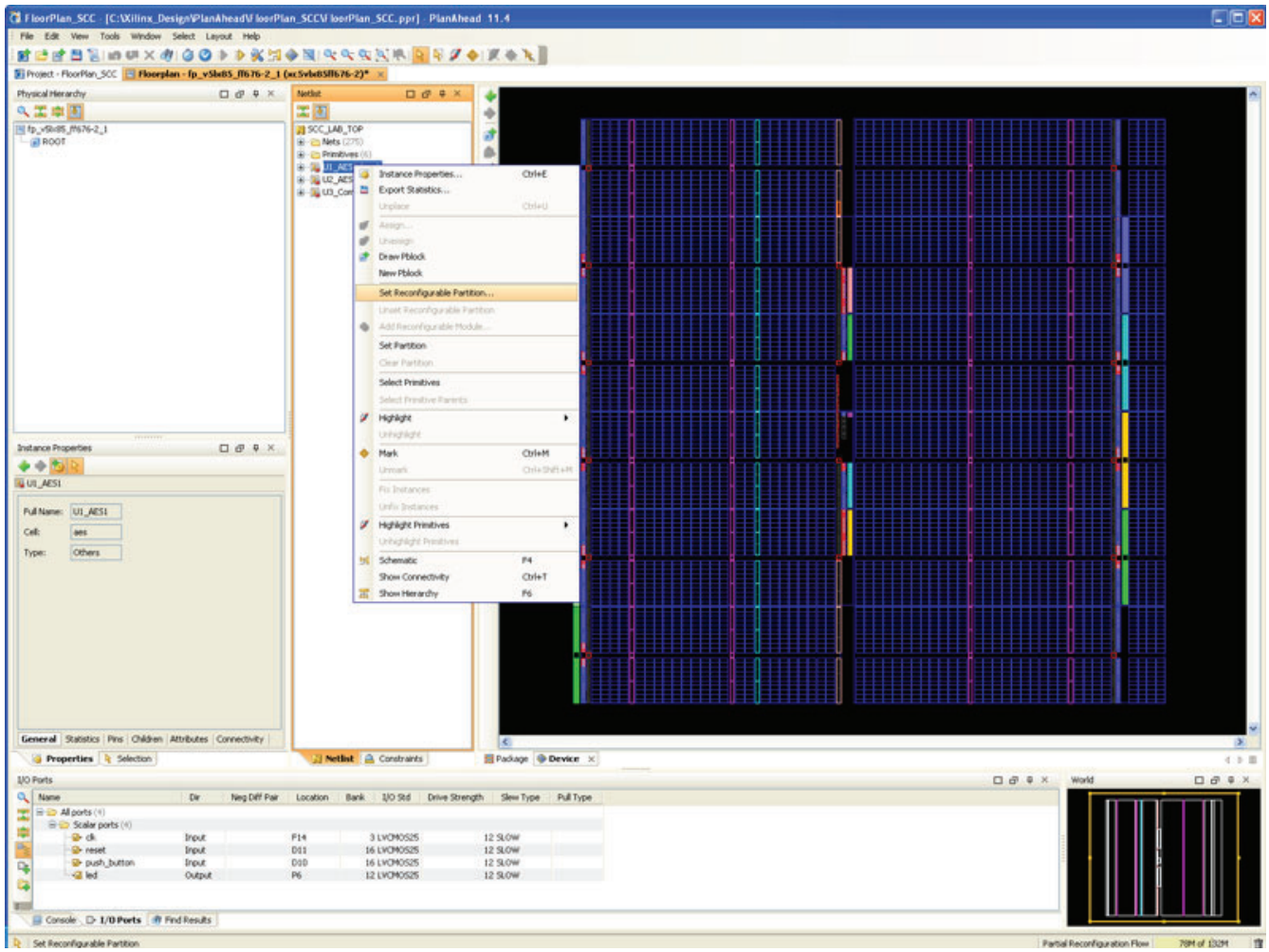
Figure 3-18: **New Timing Constraint (Delay Value)**

7. Click **OK**.

Defining RP or ISO Partitions with the PlanAhead Tool

Next each partition must be turned into either a Reconfigurable Partition (RP) or an Isolated Partition (ISO) as follows:

1. Right-click on **U1_AES (aes)** in the Netlist tab and select **Set Reconfigurable Partition** from the pull-down menu (see Figure 3-19).



X1135_c2_19_022610

Figure 3-19: **Set Reconfigurable Partition**

2. Keep the default name at **module_1**. If there were to be multiple configurations for this block, then the user would change the field to a more descriptive name for that specific configuration.
3. Repeat step 1 through step 2 for the **U2_AES2 (aes_r)** partition.
4. Repeat step 1 through step 2 for the **U3_Comp (compare)** partition with the following exception:

   Select **Set Partition** instead of **Set Reconfigurable Partition**. The compare block is an ISO block, not an RP block. This menu option is three lines below the **Set Reconfigurable Partition** option.

Defining Attributes for each RP or ISO Partition with the PlanAhead Tool

SCC rules have many additional constraint rules as opposed to a traditional Partial Reconfiguration flow. It is necessary to define several attributes for each partition as follows:

1. Under the Physical Hierarchy pane, select **pblock_U1_AES1**.
2. Under the Pblock Properties window, select the **Attributes** tab for *pblock_U1_AES1*.
3. Select the **Define New Attribute** button (a green + symbol).
4. Select the **BOUNDARYCROSS**, **CONTAINED**, and **PRIVATE** attributes by clicking each attribute while holding the **Ctrl** button (see Figure 3-20).

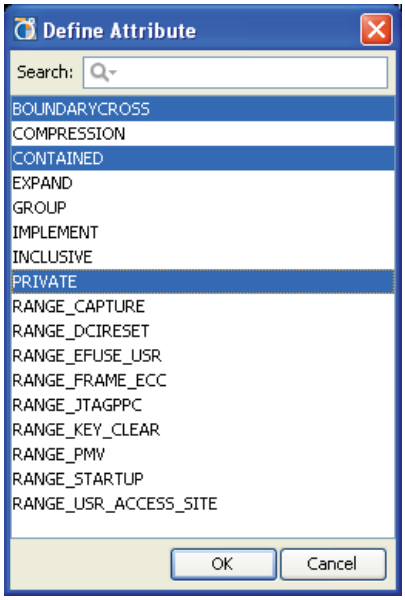


Figure 3-20: **Define Attribute**

5. Click **OK.**

6. In the Pblock Properties window (see Figure 3-21), uncheck **BOUNDARYCROSS**, set CONTAINED to **ROUTE** and set PRIVATE to **NONE**.

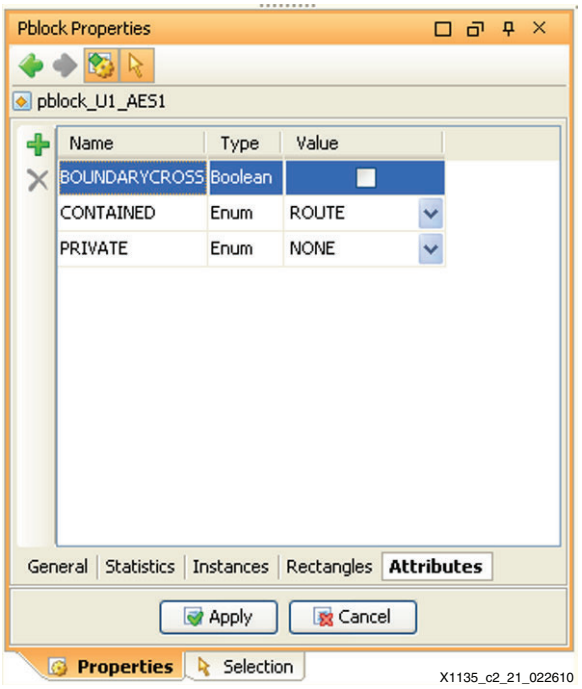


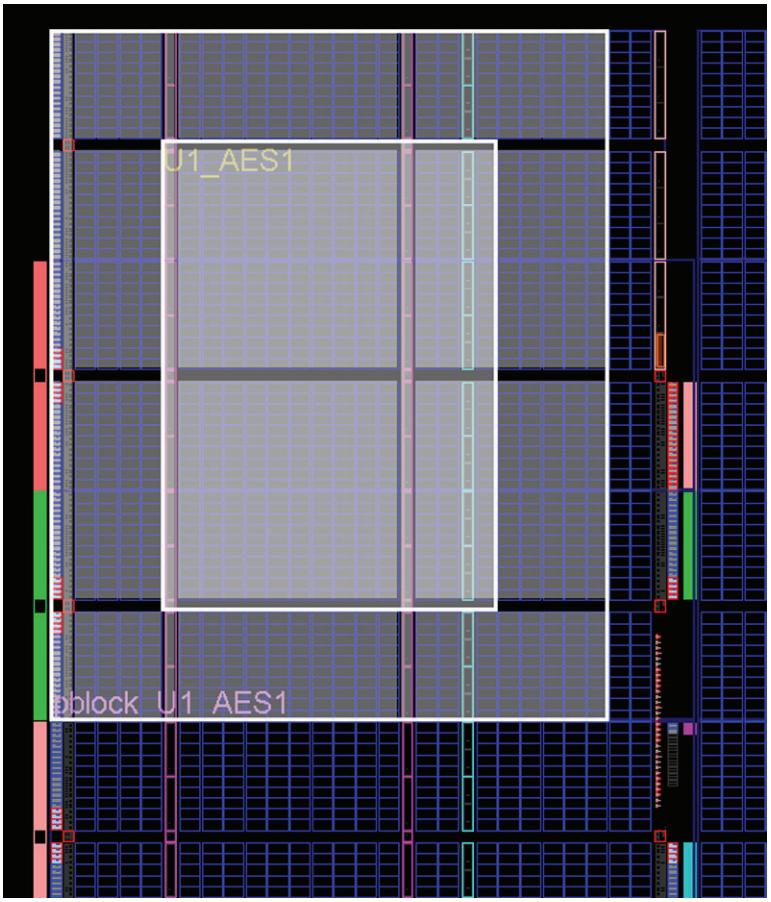
Figure 3-21: **Pblock Properties**

7. Click **Apply**.
8. Repeat step 1 through step 7 for **pblock_U2_AES2**.
9. Repeat step 1 through step 7 for **pblock_U3_Comp**.

Set Up the Area Groups for the RP Regions with the PlanAhead Tool

These steps configure the area groups for the RP regions:

1. Under the Physical Hierarchy pane, select the block **pblock_U1_AES1**.
2. Right-click on **pblock_U1_AES1** and select **Set Pblock Size** from the pull-down menu.
3. Draw a rectangle as shown in Figure 3-22 in the upper left-hand corner of the device window. (It does not have to be 100% accurate – it will be resized shortly.)



X1135_c2_22_022610

Figure 3-22: **Pblock Layout**

Note: The PlanAhead tool might resize the rectangles to many different combinations of smaller rectangles. The user needs only to ensure that there is one CLB of separation between the Pblocks AES1, AES2, and COMPARE.

4. A dialog box (see Figure 3-23) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

   Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component, even though the component is not logically instantiated in the HDL of that module.

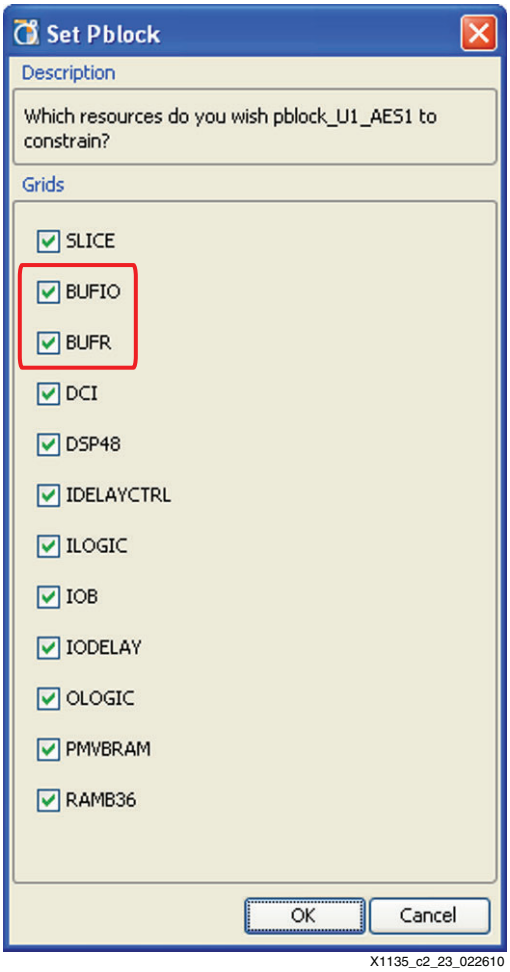


Figure 3-23: **Set Pblock**

5. Click **OK**.
6. In the Choose LOC mode dialog box, select the action **Leave all location constraints in their current position.**
7. Click **OK**.
8. Ensure that **pblock_U1_AES1** is selected in the Physical Hierarchy pane.
9. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.

10. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo = 0

    Y Lo = 1

    X Hi = 63

    Y Hi = 65

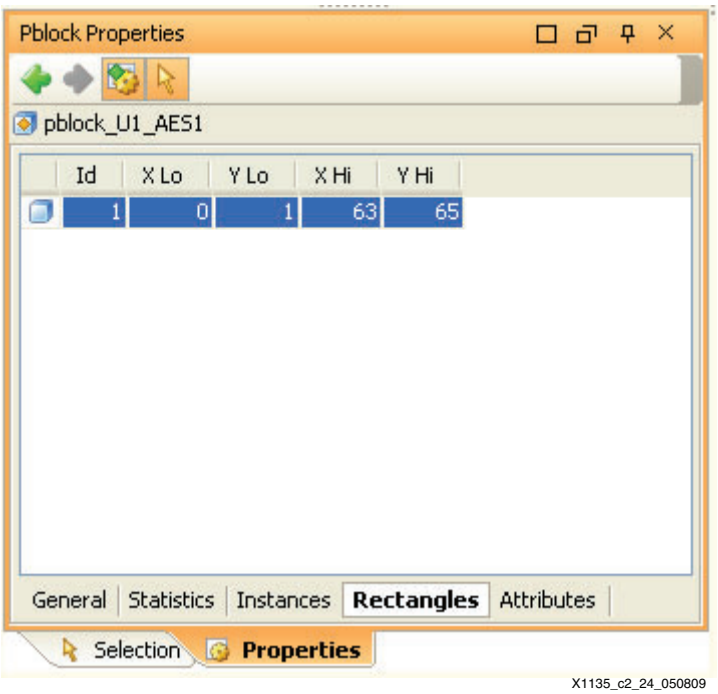
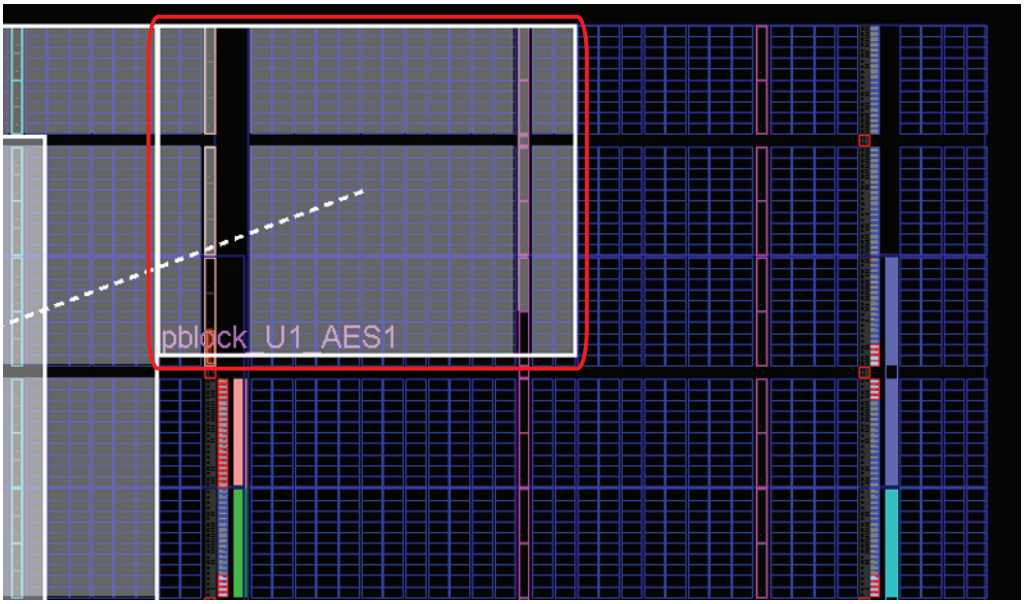


Figure 3-24: **Pblock Properties**

11. Under the Physical Hierarchy tab, select the block **pblock_U1_AES1**.
12. Right-click on **pblock_U1_AES1** and select **Add Pblock Rectangle** from the pull-down menu.

13. Draw a rectangle in the upper center of the device window. (It does not have to be 100% accurate—it will be resized shortly.)



X1135_c2_25_022610

Figure 3-25: **Pblock Layout**

14. A dialog box (see Figure 3-26) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

    Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component even though the component is not logically instantiated in the HDL of that module.

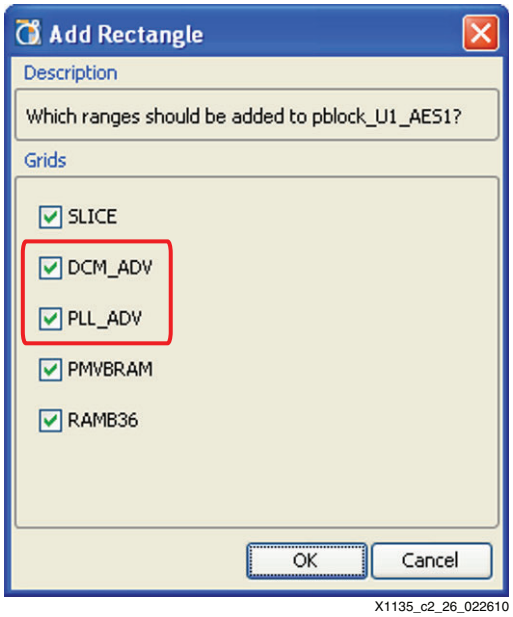


Figure 3-26: **Add Rectangle**

15. Click **OK**.
16. Ensure that **pblock_U1_AES1** is selected in the Physical Hierarchy pane.
17. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.
18. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo = 65

    Y Lo = 1

    X Hi = 109

    Y Hi = 31

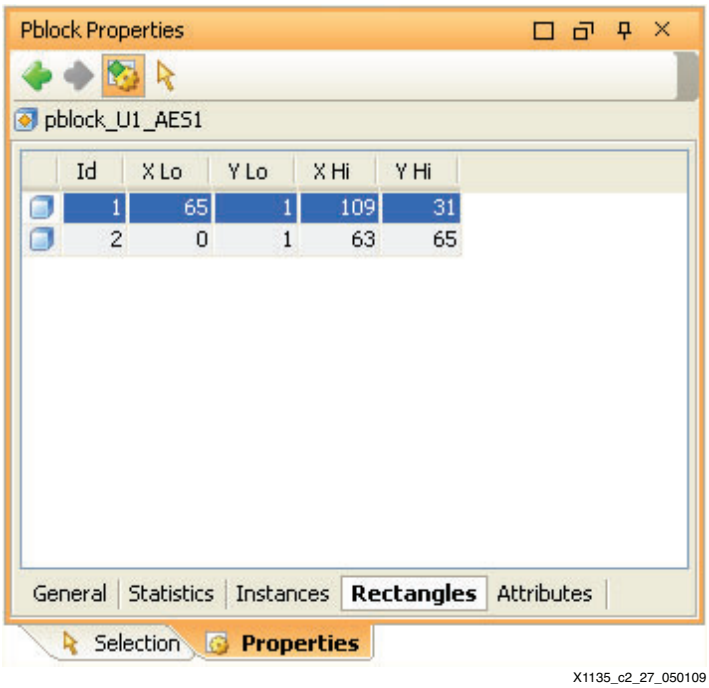


Figure 3-27: **Pblock Properties**

19. Under the Physical Hierarchy tab, select the block **pblock_U1_AES1**.
20. Right-click on **pblock_U1_AES1** and select **Add Pblock Rectangle** from the pull-down menu.

21. Draw a rectangle in the upper right of the device window. (It does not have to be 100% accurate—it will be resized shortly.)

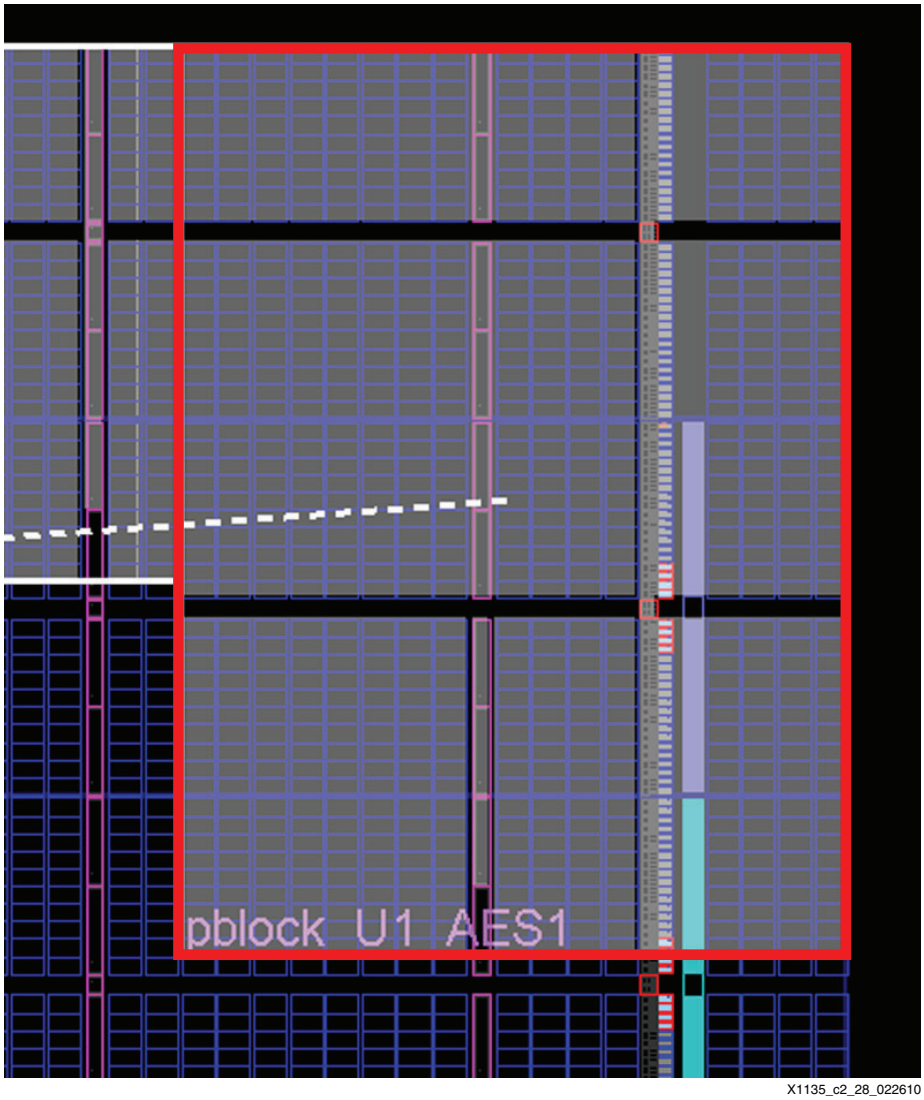
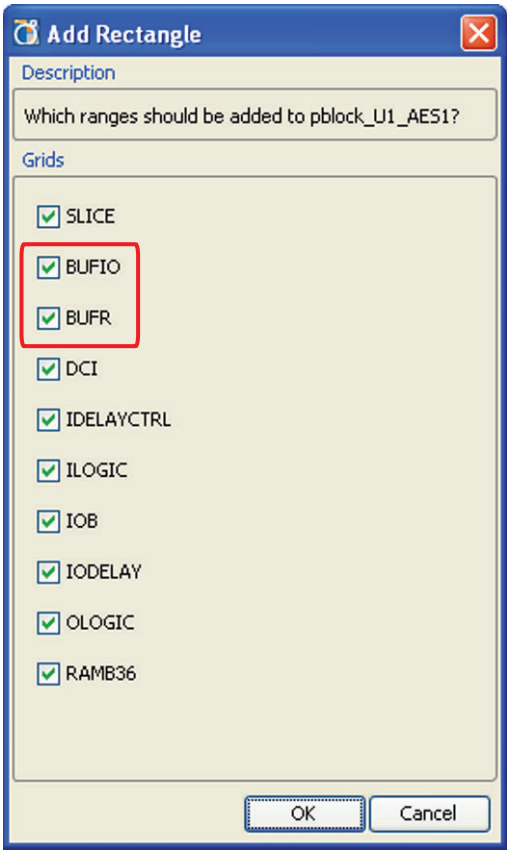


Figure 3-28: **Pblock Layout**

22. A dialog box (see Figure 3-29) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

    Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component even though the component is not logically instantiated in the HDL of that module.



X1135_c2_29_022610

Figure 3-29: **Add Rectangle**

23. Click **OK**.
24. Ensure that **pblock_U1_AES1** is selected in the Physical Hierarchy pane.
25. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.

26. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo = 112

    Y Lo = 1

    X Hi = 155

    Y Hi = 53

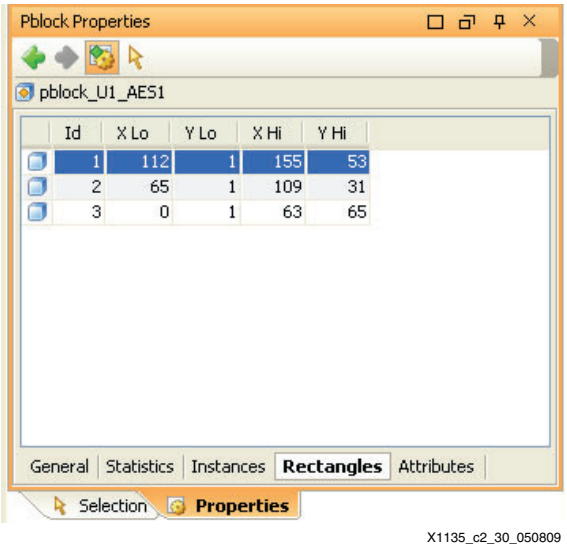
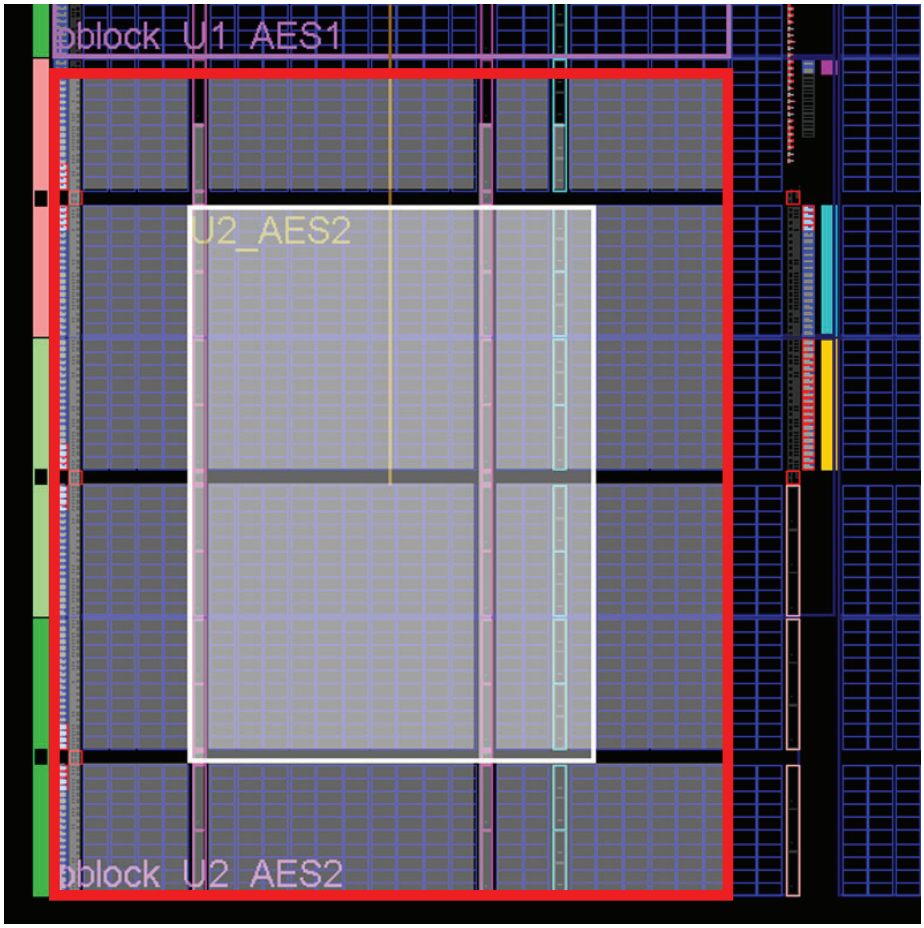


Figure 3-30: **Pblock Properties**

27. Under the Physical Hierarchy tab, select the block **pblock_U2_AES2**.
28. Right-click on **pblock_U2_AES2** and select **Set Pblock Size** from the pull-down menu.

29. Draw a rectangle in the lower left-hand corner of the device window. (It does not have to be 100% accurate—it will be resized shortly.)



X1135_c2_31_022610

Figure 3-31: **Pblock Layout**

30. A dialog box (see Figure 3-32) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

    Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component even though the component is not logically instantiated in the HDL of that module.

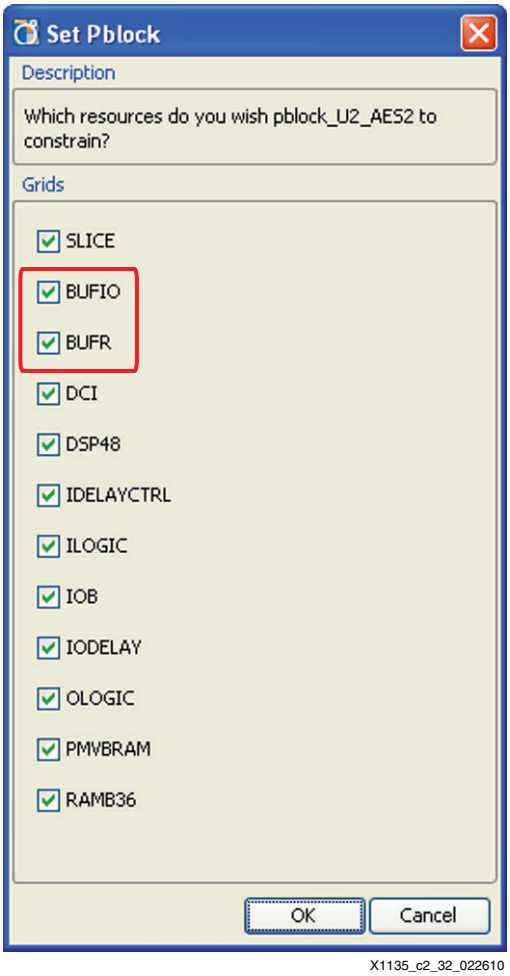


Figure 3-32: **Set Pblock**

31. Click **OK**.
32. Ensure that **pblock_U2_AES2** is selected in the Physical Hierarchy pane.
33. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.

34. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo = 0

    Y Lo = 68

    X Hi = 63

    Y Hi = 131

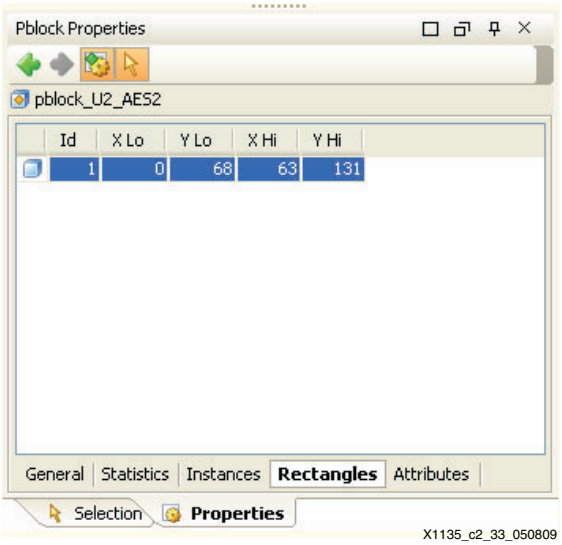


Figure 3-33: **Pblock Properties**

35. Under the Physical Hierarchy tab, select the block **pblock_U2_AES2**.
36. Right-click on **pblock_U2_AES2** and select **Add Pblock Rectangle** from the pull-down menu.
37. Draw a rectangle in the lower center of the device window. (It does not have to be 100% accurate—it will be resized shortly.)

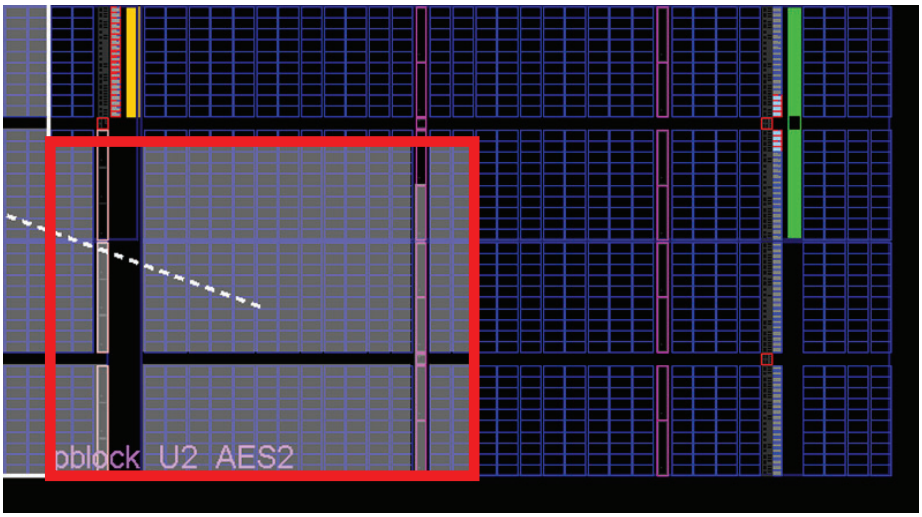
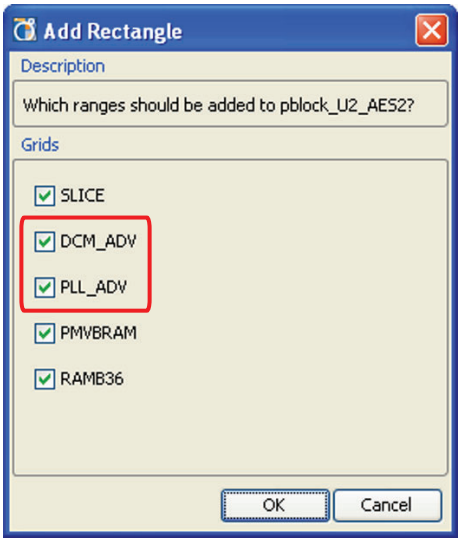


Figure 3-34: **Pblock Layout**

38. A dialog box (see Figure 3-35) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

    Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component even though the component is not logically instantiated in the HDL of that module.



X1135_c2_35_022610

Figure 3-35: **Add Rectangle**

39. Click **OK**.
40. Ensure that **pblock_U2_AES2** is selected in the Physical Hierarchy pane.
41. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.

42. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo = 65

    Y Lo = 101

    X Hi = 109

    Y Hi = 131

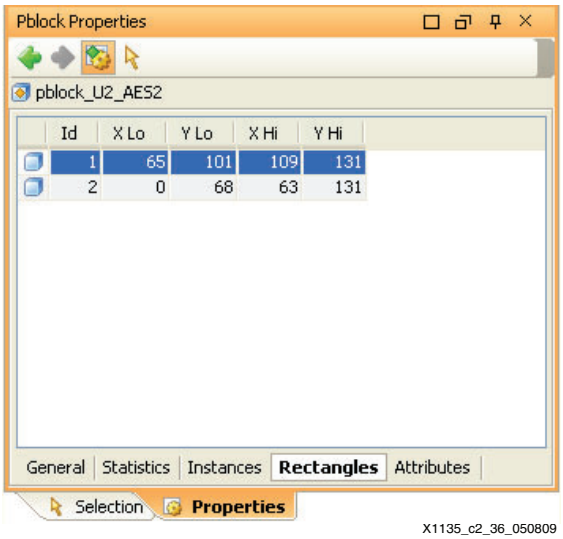
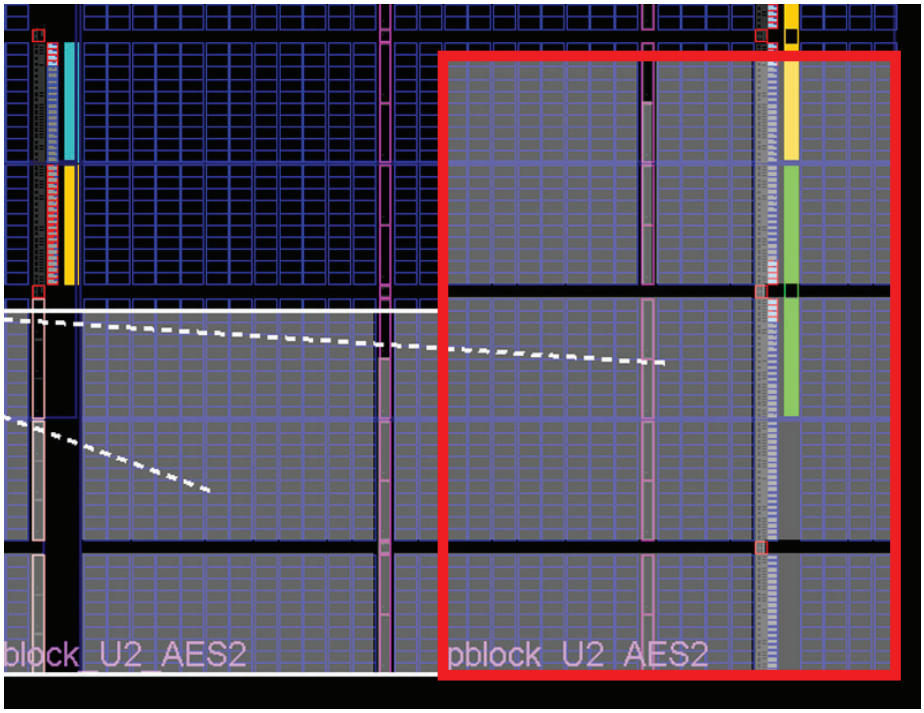


Figure 3-36: **Pblock Properties**

43. Under the Physical Hierarchy tab, select the block **pblock_U2_AES2**.
44. Right-click on **pblock_U2_AES2** and select **Add Pblock Rectangle** from the pull-down menu.

45. Draw a rectangle in the lower right of the device window. (It does not have to be 100% accurate—it will be resized shortly.)

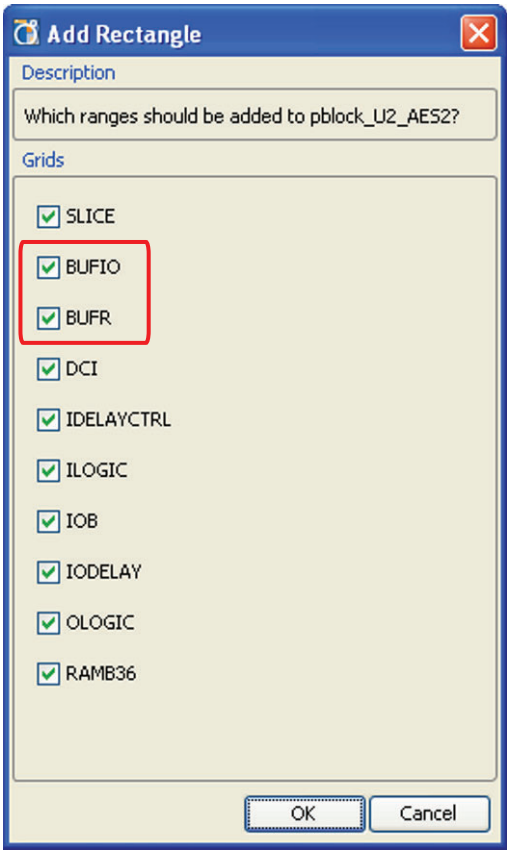


X1135_c2_37_022610

Figure 3-37: **Pblock Layout**

46. A dialog box (see Figure 3-38) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

    Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component even though the component is not logically instantiated in the HDL of that module.



X1135_c2_38_022610

Figure 3-38: **Add Rectangle**

47. Click **OK**.
48. Ensure that **pblock_U2_AES2** is selected in the Physical Hierarchy pane.
49. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.

50. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo =112

    Y Lo = 79

    X Hi = 155

    Y Hi = 131

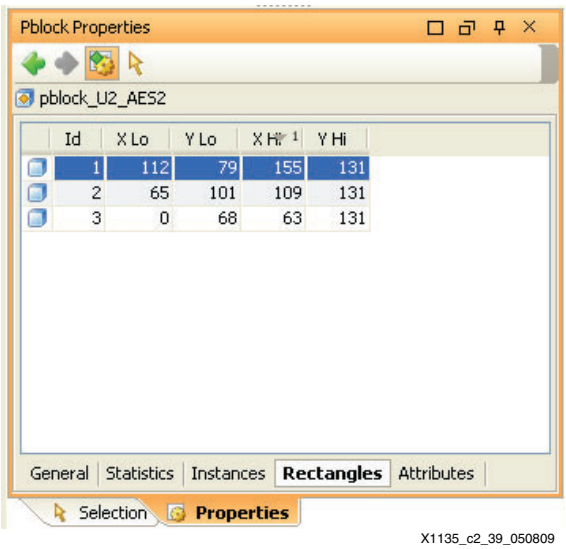
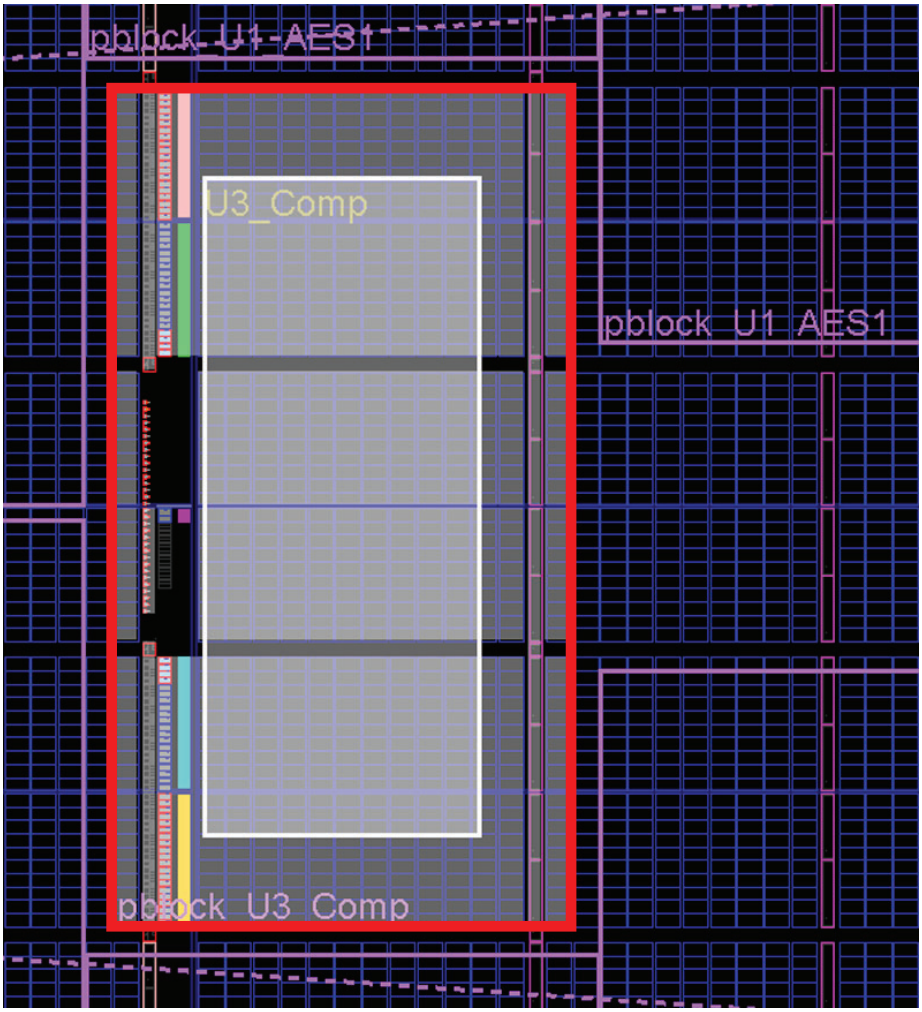


Figure 3-39: **Pblock Properties**

51. Under the Physical Hierarchy tab, select the block **pblock_U3_Comp**.
52. Right-click on **pblock_U3_Comp** and select **Set Pblock Size** from the pull-down menu.

53. Draw a rectangle in the center of the device window. (It does not have to be 100% accurate—it will be resized shortly.)



X1135_c2_40_022610

Figure 3-40: **Pblock Layout**

54. A dialog box (see Figure 3-41) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

    Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component even though the component is not logically instantiated in the HDL of that module.

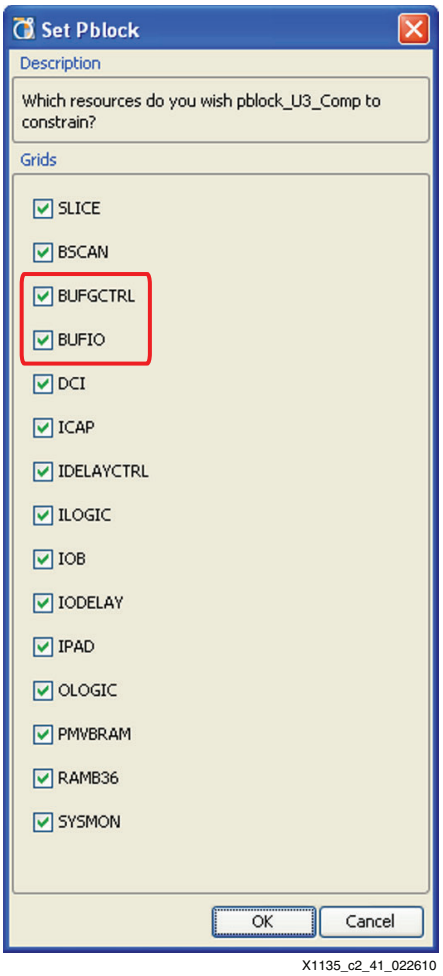


Figure 3-41: **Set Pblock**

55. Click **OK**.
56. In the Choose LOC mode dialog box, select the action **Leave all location constraints in their current position.**
57. Click **OK**.
58. Ensure that **pblock_U3_Comp** is selected in the Physical Hierarchy pane.
59. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.

60. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo = 67

    Y Lo = 34

    X Hi = 107

    Y Hi = 98

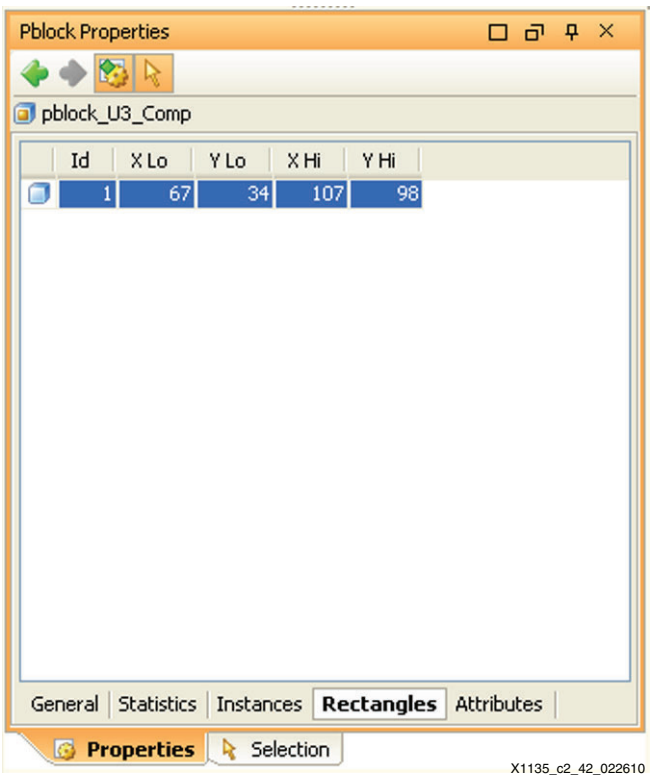
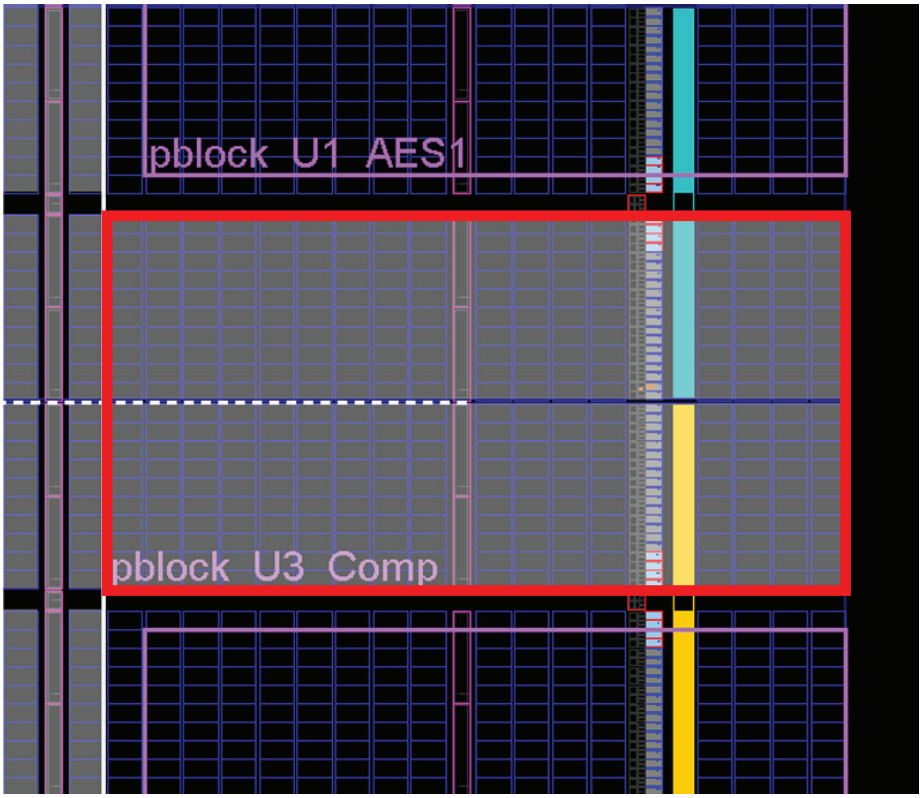


Figure 3-42: **Pblock Properties**

61. Under the Physical Hierarchy tab, select the block **pblock_U3_Comp**.
62. Right-click on **pblock_U3_Comp** and select **Add Pblock Rectangle** from the pull-down menu.

63. Draw a rectangle in the center right of the device window. (It does not have to be 100% accurate—it will be resized shortly.)

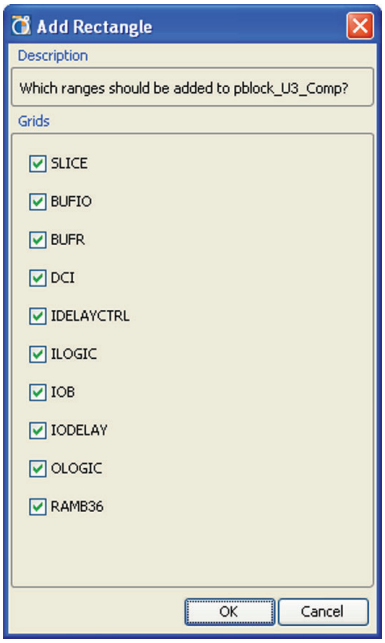


X1135_c2_43_022610

Figure 3-43: **Pblock Layout**

64. A dialog box (see Figure 3-44) pops up with various attributes and associated checkboxes. All boxes must be checked, including (if listed) **DCM_ADV**, **PLL_ADV**, **BUFGCTRL**, **BUFR**, and **BUFIO**. Even though most of these blocks are not in the design, it is important to select them to take advantage of their routing resources. All used components must be included.

    Note: Trusted Routing requires that all clocking components be assigned to the AREA GROUP that physically contains the component even though the component is not logically instantiated in the HDL of that module.



X1135_c2_44_022610

Figure 3-44: **Add Rectangle**

65. Click **OK**.
66. Ensure that **pblock_U3_Comp** is selected in the Physical Hierarchy pane.
67. Select the **Rectangles** tab in the Properties tab of the Pblock Properties window.

68. Adjust the rectangle graphically as necessary to match the following coordinates:

    X Lo = 109

    Y Lo = 56

    X Hi = 155

    Y Hi = 76

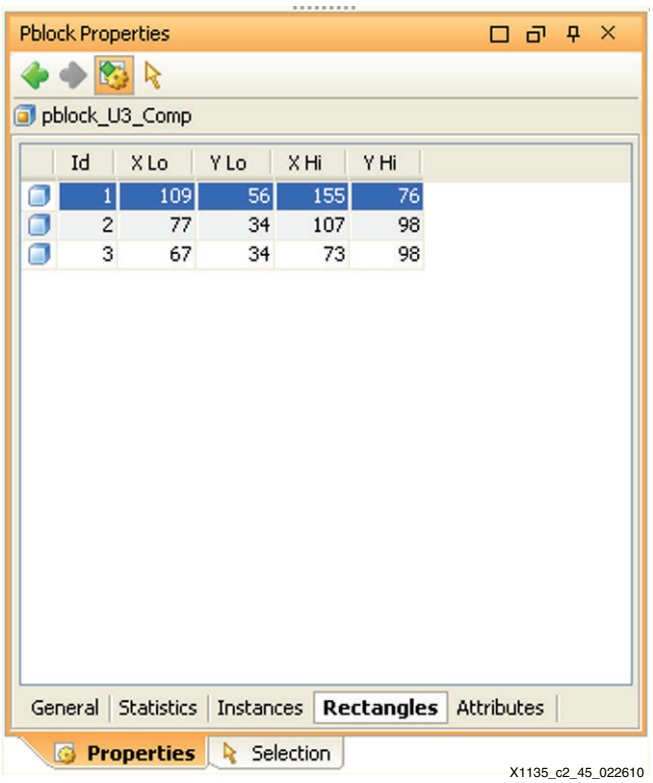


Figure 3-45: **Pblock Properties**

69. The final layout is shown in Figure 3-46. Each block is separated by one CLB to ensure SCC isolation. A block RAM, DSP, IOB, or any other site type that contains a Global Switch Matrix (GSM) can be used for this isolation.

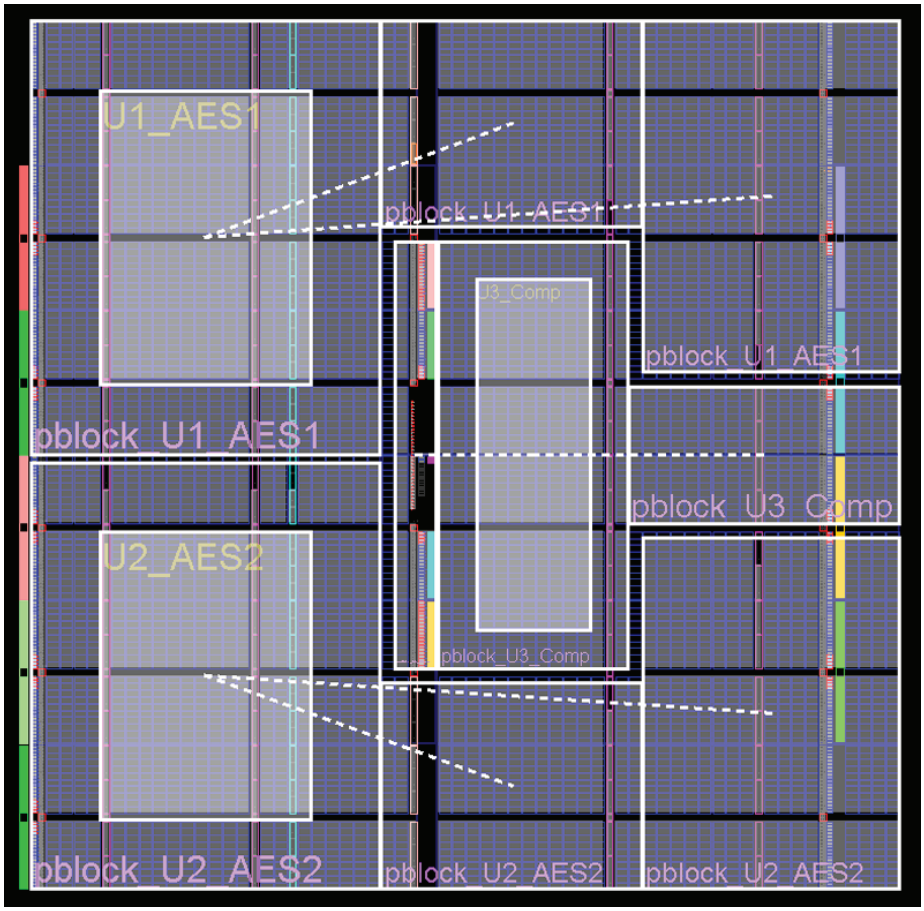


Figure 3-46: **Final Layout**

70. Ensure there is one CLB of spacing between each area group.

Running Design Rule Checks

Given the complexity of the SCC flow, it is highly recommended to run the built-in design rule checks (DRCs) of the PlanAhead tool. This step can be time saving because it can find many common mistakes early on in the design process.

1. From the Tools menu, select **Run DRC**. Alternatively, click the checkmark button on the top toolbar (see Figure 3-47).

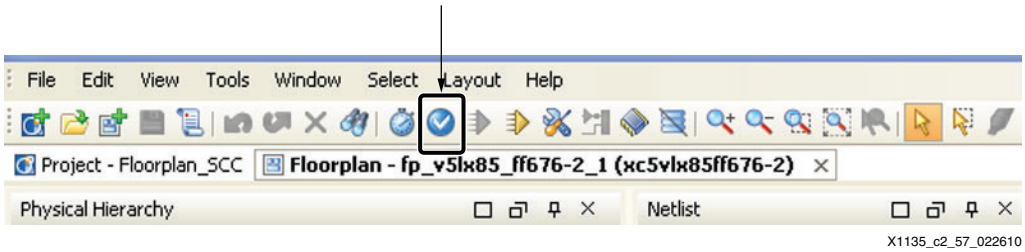


Figure 3-47: **Checkmark Button**

2. Due to incompatibilities between Commercial Partial Reconfiguration and SCC Trusted Routing, several DRC rules need to be disabled:
   a. Bus macro between static logic and isolated region (ISNM)
   b. PR static logic illegally placed (PRLL)
   c. Area group tile alignment (FLBA).

3. Click **OK** (see Figure 3-48).

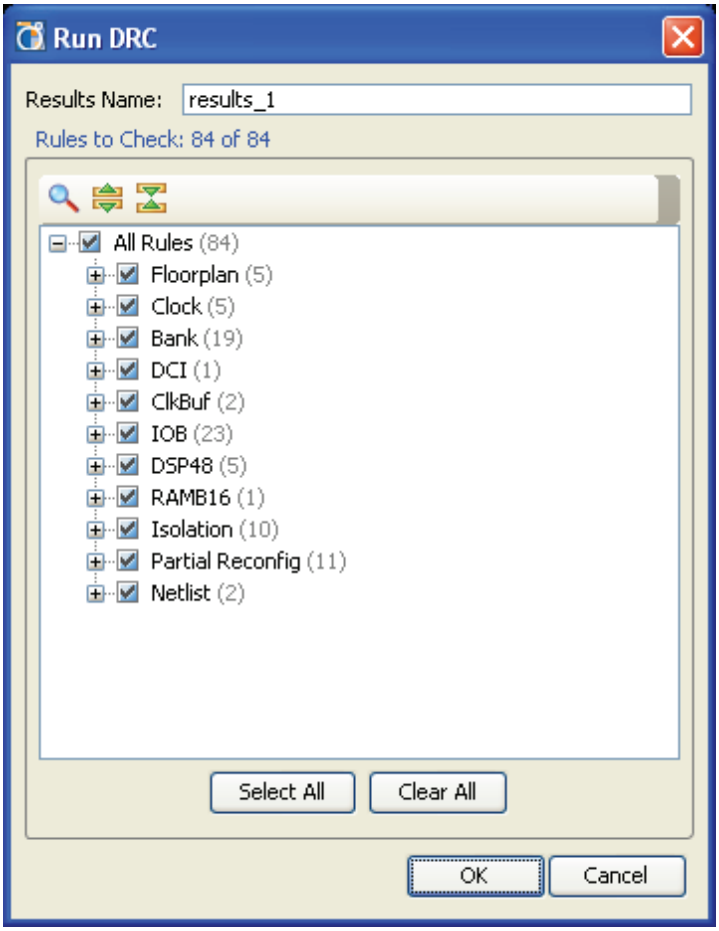


Figure 3-48: **Run DRC Dialog Box**

4. A new tab, DRC Results, appears at the bottom with a sub-tab named as specified in the Run DRC window (results_1 in this case). There should be two violations:
   a. PRSC #1: This warning indicates that a configuration has not been defined for the U1_AES1 module. This warning goes away when at least one implementation has been completed.
   b. PRSC #2: This warning indicates that a configuration has not been defined for the U2_AES2 module. This warning goes away when at least one implementation has been completed.

For a complete list of DRCs and their descriptions, consult the *PlanAhead User Guide* in the Help pull-down menu.

Running TimeAhead with the PlanAhead Tool

This section tests the timing constraints that were set up in "Setting Up Timing Constraints with the PlanAhead Tool," page 57.

1. Select **Tools** → **Run TimeAhead** and click **OK** (see Figure 3-49).

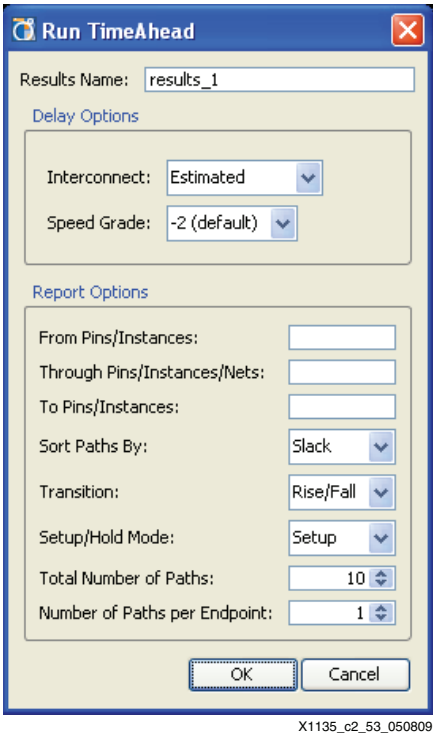


Figure 3-49: **Run TimeAhead Dialog Box**

2. A new tab, Timing Results, appears at the bottom with a sub-tab named as specified in the Run TimeAhead window (results_1 in this case). As specified when launched, TimeAhead reports the 10 paths closest to missing timing.

Design Flow Progress

The DRC and Timing block of the SCC system design flow diagram is complete, as shown in Figure 3-50.

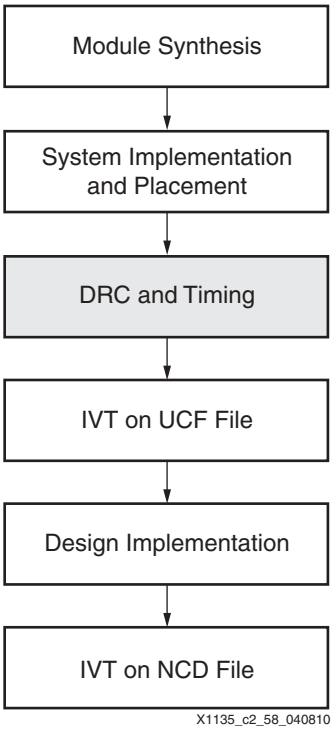


Figure 3-50: **SCC System Design Flow with DRC and Timing Block Complete**

Exporting the Design

At this stage, the design can be exported. The user can export either the combined netlist in EDIF form, the top-level netlist, or both, if desired. For this lab, only the combined netlist need be exported for checking by the Isolation Verification Tool.

1. To export the netlist, select **File** → **Export Netlist**.
2. Browse to the `..\Xilinx_Design` directory and select **OK**.
3. Keep the default file name `fp_v5lx85_ff676-2_1.ucf`
4. To export the constraints file, select **File** → **Export Constraints** and browse to the desired location.
5. Click **OK**.

Chapter 4

Running the Isolation Verification Tool Against the UCF

The Xilinx Isolation Verification Tool (IVT) software verifies that an FPGA design that has been partitioned into isolated modules meets the stringent standards for a fail-safe design. IVT is a batch application with a command line and file-based user interface. While there is a graphical output there is no graphical user interface.

While not required, it is highly recommended that the user run IVT on the UCF. This can catch pin and area group isolation faults early in the design when changes are more easily integrated. The steps in this chapter guide the reader through the process. After implementation, the IVT NCD test (required for SCC designs) is run against the routed design.

Creating the File Used to Run the IVT UCF Test

These steps describe how to create the file used to run the IVT UCF test:

1. Open the directory `..\Xilinx_Design\ivt`.
2. Notice the files:
   - `ivt.exe` – the IVT executable for IVT, version 5.0
   - `IVT_End_User_License_Agreement.pdf` – the IVT user license
3. Create a new text file and name the file `SCC-LAB_ucf.ivt` with the following contents:

```
-device xc5vlx85 -package ff676

# Groups                Isolation Group             Area Group
# -------             --------------------        ---------------------
-group                  AES                       pblock_U1_AES1
-group                  AES_r                     pblock_U2_AES2
-group                  COMPARE                   pblock_U3_Comp

# Pin Isolation Groups
-pig SCC-LAB.pig

# User Constraint File
..\PlanAhead\Floorplan_SCC\fp_v5lx85_ff676-2_1.ucf

# Output file
-output SCC-LAB_ucf.rpt
```

4. Notice the instructions placed in the IVT command file:
   a. The first line sets the target device and package for IVT to compare against

b. Three isolation groups are assigned: AES, AES_r, and COMPARE

Note: These names are arbitrary. If desired they can be named the same to denote RED and BLACK groups.

c. Three Area groups are assigned: pblock_U1_AES1, pblock_U2_AES2, and pblock_U3_Comp

Note: These names must match the area groups in the UCF.

d. IVT is pointed to the UCF

e. IVT is told where to place the output file and it's associated name

5. Save and close the IVT command file.

Creating the Pin Isolation Group File

The pin isolation group (PIG) file is an IVT command file that defines which pins are associated with what isolation groups. These steps describe the process for creating a PIG file:

1. Open the directory `..\Xilinx_Design\ivt`.
2. Create a new text file and name the file `SCC-LAB.pig` with the following contents:

```
# Place all Global (top level) signals here (each commented out)
# NET "clk" LOC = F14;

ISOLATION _GROUP AES BEGIN
  NET "reset"       LOC = D11;
  NET "push_button" LOC = D10;
END ISOLATION_GROUP

ISOLATION_GROUP AES_r BEGIN
# There are no pins in AES_r
END ISOLATION_GROUP

ISOLATION_GROUP COMPARE BEGIN
  NET "led" LOC = P6;
END ISOLATION_GROUP
```

3. Notice the instructions placed in the IVT PIG file:

   a. The clock pin, clk, is commented out because it is not required to be isolated.

   b. The three isolation groups that were created in the IVT UCF command file have their associated pins assigned to them.

4. The isolation group definitions must match the isolation group definitions from the IVT command created in "Creating the File Used to Run the IVT UCF Test."

   Hint: The IVT PIG file uses UCF syntax for each pin definition. It is useful to copy the pins from the UCF and place them in the PIG file for start. From there the user can either comment out the lines at the top level or add isolation group definitions around the remaining pins to assign them to their specific isolation group.

Running the IVT UCF Test

These steps describe how to run the IVT UCF test:

1. Open a DOS command prompt (**Start** → **Run** → **cmd**).
2. Navigate to the `..\Xilinx_Design\ivt` directory.
3. Run the UCF test by typing the following at the command prompt:

   ```
   ivt -f SCC-LAB_ucf.ivt
   ```

4. The output for a successful UCF test run will be "`SUCCESS!`".
5. For more detailed messages from IVT, add the **`-verbose`** switch to the IVT command file or at the command line.

   Hint: It is useful to add this command line to a file with the name `run_ivt_ucf.bat` so that it can be double-clicked from a Windows Explorer window.

Examining the Output from the IVT UCF Test

The output report has five key sections:

1. Isolation Groups and Area Groups
2. Pin Isolation Summary

   ```
   Die Pin Adjacency Violations: 0
   Package Pin Adjacency Violations: 0
   Bank Isolation Violations: 0
   ```

3. Area Fault Summary

   ```
   Area Group Faults: 0
   ```

4. Isolation Verification Summary

   ```
   UCF file contains 0 constraint violations.
   Isolation analysis completed.
   Elapsed time: 0:00:03
   ```

Chapter 5

Implementing the Design with the PlanAhead Tool

Generating and Running an Implementation

These steps describe how to generate and run a design implementation:

1. From the tools menu, select **Run Implementation**.
   a. Select **Tools** → **Run Implementation** (see Figure 5-1).

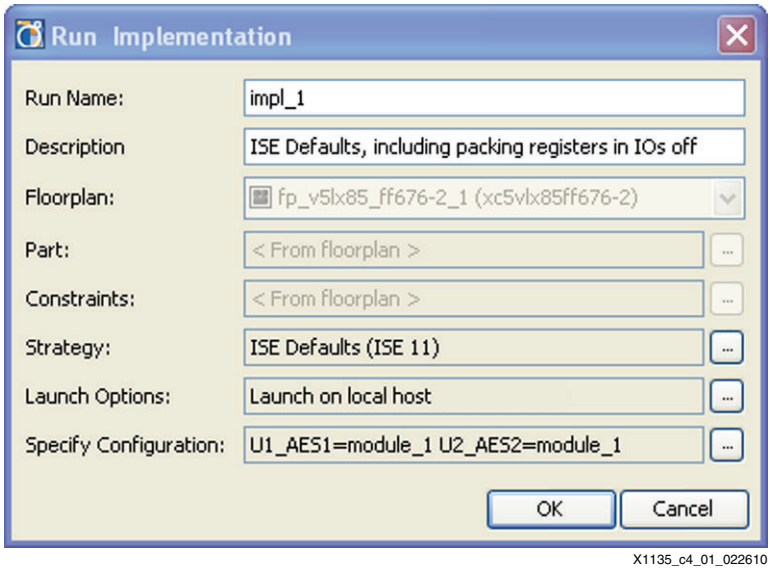
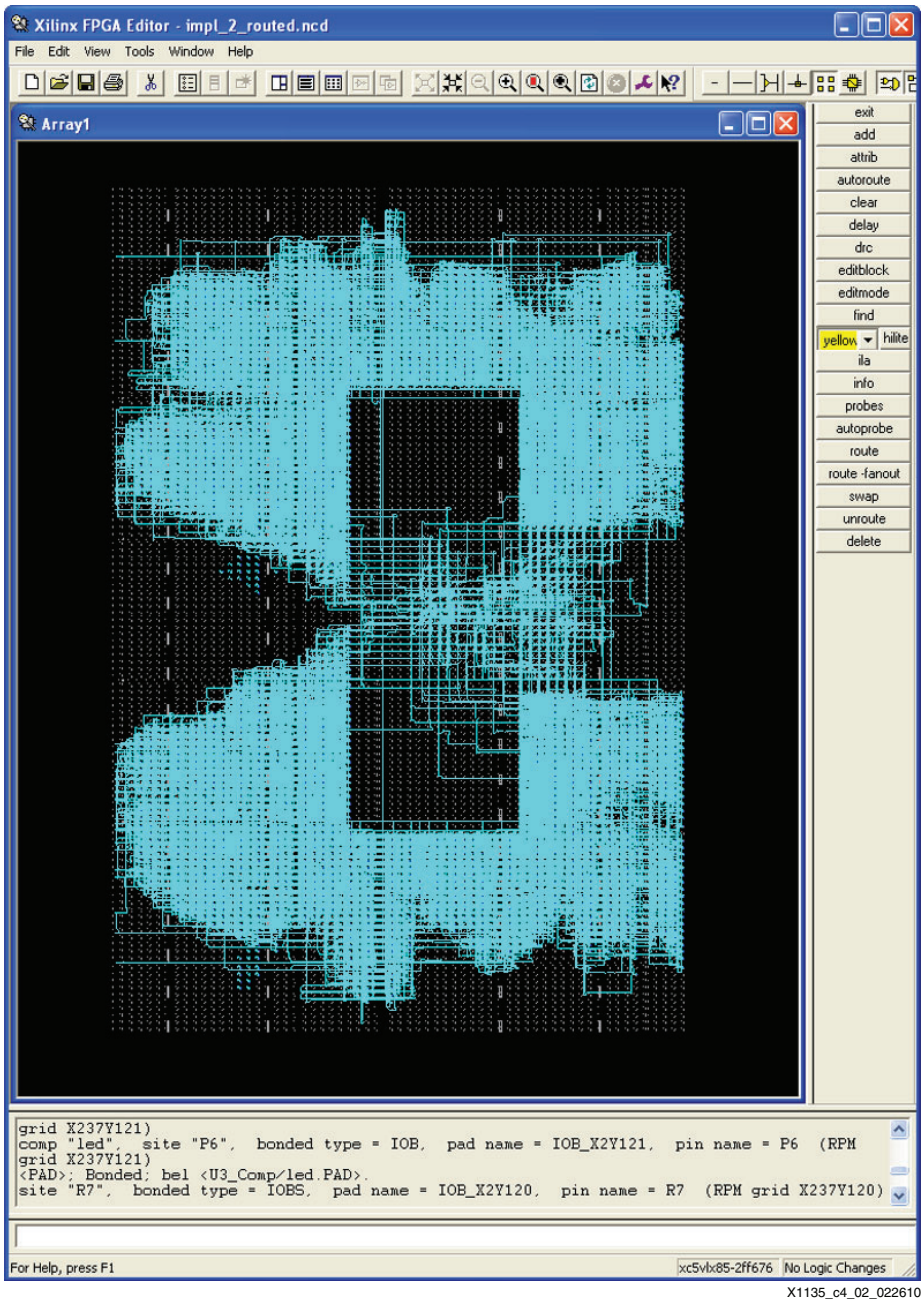


Figure 5-1: **Run Implementation**

   b. Accept the defaults and click **OK**.
2. A new tab named Design Runs is created and a single entry named "impl_1" is generated as specified in the Run Implementation window. It immediately starts the run: **NGDBUILD** → **MAP** → **Place and Route**.
3. The combined and routed design is placed in the following directory:

   `..\Xilinx_Design\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.runs\impl\floorplans\fp_v5lx85_ff676-2_1\impl_1`

   The file name is `impl_1_routed.ncd` and its view in FPGA Editor is shown in Figure 5-2.



X1135_c4_02_022610

Figure 5-2: **FPGA Editor View**

Chapter 6

Verifying the Design with the NCD Isolation Verification Tool

Creating the File Used to Run the IVT NCD Test

These steps describe how to create the file used to run the IVT NCD test:

1. Open the directory
   `\Xilinx_Design\ivt\`.
2. Create a new text file in the `\Xilinx_Design\ivt\` directory:
   `SCC-LAB_ncd.ivt`
3. Open text file `SCC-LAB_ncd.ivt` in a text editor.
4. Add the following to the `SCC-LAB_ncd.ivt` text test file.

```
# comment the next line for reduced detail in the report file.
-verbose
# Groups    Isolation Group     Instance Name in Final NCD
# --------  ----------------    --------------------------
-group       AES                  U1_AES1
-group       AES_r                U2_AES2
-group       COMPARE              U3_Comp

# Combined design
..\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.runs\impl\floorplans\
fp_v5lx85_ff676-2_1\impl_1\impl_1_routed.ncd

# Output Report File
-output SCC-LAB_ncd.rpt
```

The NCD IVT command file sets the following options:

- Enables the verbose IVT switch
- Assigns three area groups to the three NCD files making up the project
- Points the IVT tool to the combined NCD file
- Tells IVT what to name the output report file and where to put the file

Note: The Isolation Group names are arbitrary but the instance name must match the actual design.

Running the IVT NCD Test

These steps describe how to run the IVT NCD test:

1. Open a command prompt (**Start** → **Run** → **cmd**).
2. Navigate to the `\Xilinx_Design\ivt\` directory.
3. To run the IVT NCD test, type the following at the command prompt:

   ```
   ivt -f SCC-LAB_ncd.ivt
   ```

4. The output for a successful NCD test run will be "`SUCCESS!`".

Examining the Output from the IVT NCD Test

These steps describe how to read the output file from the IVT NCD test:

1. Open the IVT NCD test output file:

   ```
   \Xilinx_Design\ivt\SCC-LAB_ncd.rpt
   ```

   All the groups are listed in the input designs section:

   ```
   Input Designs

   Group AES module: U1_AES1
   Group AES_r module: U2_AES2
   Group COMPARE module: U3_Comp
   ```

2. Ensure that Clocks and Resets are listed in the unidentified shared networks section. For SCC Trusted Routing designs, signals shared between isolated regions are expected and intended and will show up here. An example of global signals are the following Global Clock signals, which are expected to be shared outside of an isolated region:

   ```
   Unidentified Shared Networks

   The networks below are found in multiple isolation groups, therefore it
   is incumbent upon the user to prove these signals do not violate data
   isolation requirements. Only power, ground, bus macros, global resets
   or explicitly permitted control signals may be shared.

   clk_ibufg
   clk0_buf
   clkdev_buf
   ```

3. Ensure all remaining Clocks are listed in the Networks Driven by Global Clock Sources section:

   ```
   Networks Driven by Global Clock Sources (BUFG, DCM, PLL, and PMCD)

   clk_fb_i
   clk_i
   ```

4. Notice that in the Identified Networks section, Shared Networks Attached to Bus Macros are listed. All networks passing through Bus macros are considered safe by IVT and are listed in this section.
5. The next section in the report contains Shared Networks Without General Routing (Including Clock Networks) information. As stated in the report file each of the following networks is either internal to a logic block (such as a CLB or an IOB) or is part of a global clock network, therefore they do not need to be considered for isolation analysis.

6. Ensure that only bus macros are listed in the Bus Macro Usage Summary.
7. Pay special attention to the Package Pins, I/O Buffers, and I/O Banks because the note states the user must verify that pins connected to ignored networks are correct.

Package Pins, I/O Buffers, and I/O Banks

Note: It is incumbent on the user to verify that pins connected to ignored networks are correct. For example, pins must not be directly connected to bus macros, but pins can be connected to clocks, power, and global resets.

```
Pin(col, row)   Bank   I/O Buffer    Isolation Group          Network
-------------   ----  ------------   ---------------   ------------------------
D10( 16,  22)    16     IOB_X2Y198       AES           U1_AES1/push_button_IBUF
D11( 15,  22)    16     IOB_X2Y199       AES           U1_AES1/reset_out
P6( 20,  12)     12     IOB_X2Y121       COMPARE       U3_Comp/led_OBUF
F14( 12,  20)     3     IOB_X1Y179       ignored       clk
```

The following output in the NCD report indicates that there are no faults in the NCD and lists the time it took to run the test:

```
Pin Isolation Summary

  Die Pin Adjacency Violations: 0
  Package Pin Adjacency Violations: 0
  Bank Isolation Violations: 0

Network Isolation Summary

  Net Isolation Faults Found: 0
  Net Isolation Faults Reported: 0
  Routes Examined: 389030

Isolation Verification Summary

  Design contains 0 constraint violations.

  Isolation analysis completed.

  Elapsed time: 0:03:35
```

Design Flow Progress

The Single Chip Crypto lab is complete with the IVT on NCD File block of the flow diagram, as shown in Figure 6-1.

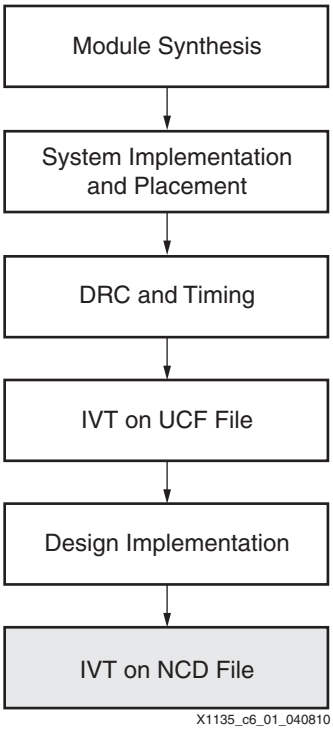


Figure 6-1: **SCC System Design Flow with IVT on NCD File Block Complete**