Implementation of a Fail-Safe Design in the Spartan-6 Family Using ISE Design Suite 12.4

XAPP1104 (v1.0.1) June 19, 2013

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/23/11	1.0	Initial Xilinx release.
06/19/13	1.0.1	Replaced SCC lounge with IDF page throughout. Added URL to download reference design in Reference Design Files.

Table of Contents

About This Guide

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

Guide Contents

This manual contains the following chapters:

- Chapter 1, Single Chip Crypto Design Overview, describes the SCC design and the goals of the lab.
- Chapter 2, Synthesizing Modules for the Isolation Design Flow, describes the steps in the bottom-up synthesis flow.
- Chapter 3, Floorplanning the System, gives step-by-step instructions for implementing the SCC design.
- Chapter 4, Running the Isolation Verification Tool Against the UCF, covers running the Isolation Verification Tool (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.
- Appendix A, Tactical Patch Needed for ISE Tools 12.3 and 12.4, describes the tactical patches needed for ISE tools 12.3 and 12.4 for the Windows PC and Linux server.

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.

Single Chip Crypto Design Overview

This chapter describes how a Single Chip Crypto (SCC) system can be created using a Spartan®-6 FPGA, and the Xilinx® ISE® Design Suite 12.4 and PlanAhead™ 12.4 development tools.

The isolation design flow (IDF) allows for applications that include redundant type 1 encryptors, resident red and black data, and functionality that operates on multiple levels of security within the same Xilinx FPGA.

At the core of the IDF is physical and electrical isolation (ISO) of chosen logic from other areas of the design. This is accomplished via:

- Conventional, best practices, modular Xilinx design entry where each function isolated by the user must be at its own level of hierarchy
- Modular synthesis of each isolated region of the design
- (Mandatory) Manual floorplanning to isolate the desired regions of the design on the FPGA while providing user-specified intra-region communication through the use of trusted routing
- Automated verification of isolation via the Xilinx Isolation Verification Tool (IVT) software

Lab Design Overview

The SCC rules for the Spartan-6 FPGA are outlined in XAPP1145, *Developing Fail-Safe Designs in the Spartan-6 Family Using the Isolation Design Flow*. This application note gives details on how functions are to be isolated, specific differences between a normal partition flow and an SCC partition flow, information on SCC-specific HDL code mnemonics, and trusted routing rules.

To illustrate the IDF and capabilities, this design implements isolated, redundant AES modules. Figure 1-1 is a hierarchical diagram of the various VHDL sub-blocks used in the implementation of the design. Type 1 Spartan-6 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 Spartan-6 XC6SLX150T-3FGG676 device.

Note: This lab requires the ISE software, version 12.2 or later.

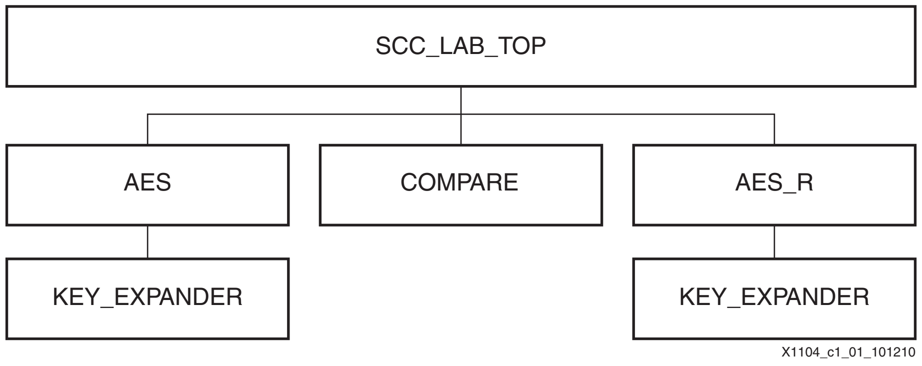


Figure 1-1: **VHDL Design Hierarchical Block Diagram**

Figure 1-2 shows the IDF used for the SCC lab.

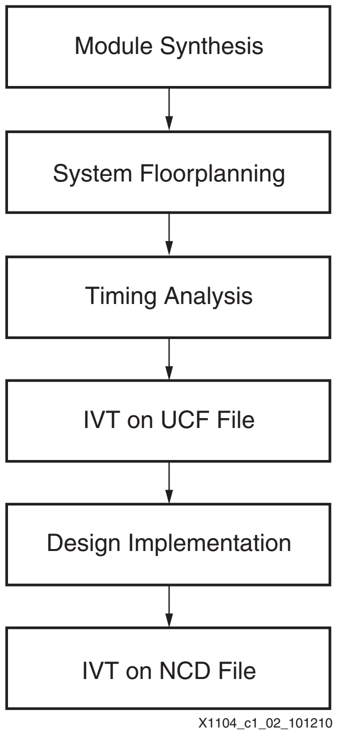


Figure 1-2: **Isolation Design Flow**

Figure 1-3 shows the partitioning and I/O mapping for an XC6SLX150T-3FGG676 device.

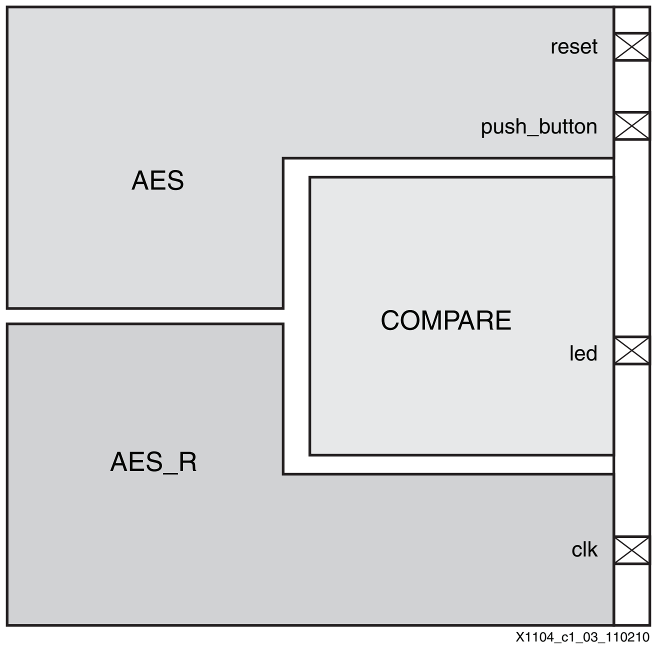
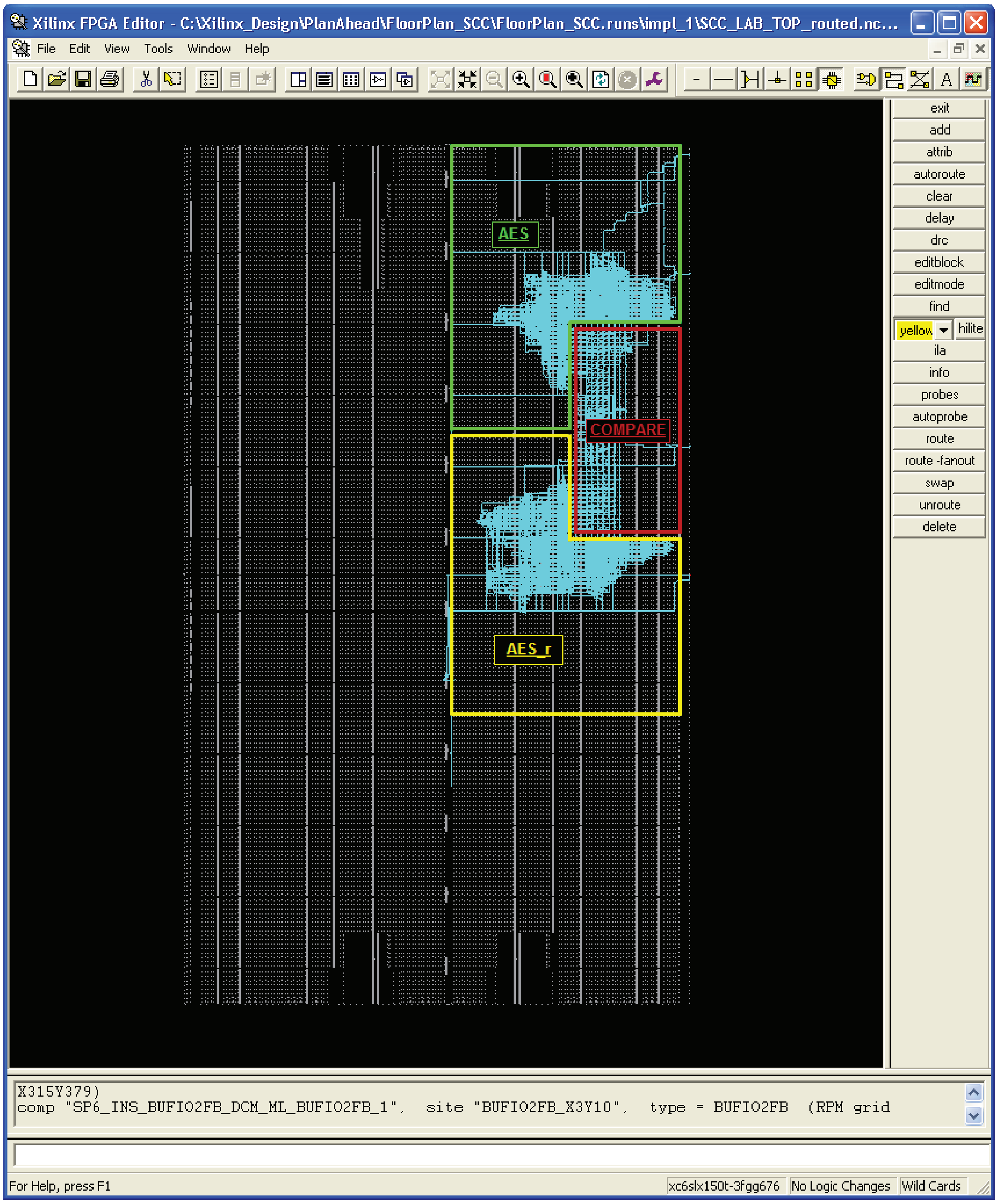


Figure 1-3: **Die View: Partition and I/O Map of XC6SLX150T-3FGG676**

Figure 1-4 is a view of the completed SCC reference design 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 angle-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.



```
X315Y379)
comp "SP6_INS_BUFIO2FB_DCM_ML_BUFIO2FB_1",  site "BUFIO2FB_X3Y10",  type = BUFIO2FB  (RPM grid
```

Figure 1-4: **FPGA Editor View: Implementation of the SCC Design**

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=343344

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	Spartan-6 SLX150T FPGA
Source code provided	Y
Source code format	VHDL
Design uses code/IP from existing Xilinx application note/ reference designs, CORE Generator™ software, or third 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 design suite 12.4
SPICE/IBIS simulations	N
Implementation	
Synthesis software tools/version used	XST
Implementation software tools/versions used	ISE design suite 12.4
Static timing analysis performed	Y
Hardware Verification	
Hardware verified	N
Hardware platform used for verification	N/A

Installing 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 `XAPP1104.zip` to the Windows desktop.
2. Double-click `XAPP1104.zip` and unzip the contents to the desired location.
3. The project files are placed in these directories:

   ```
   \Xilinx_Design\buildscripts
   \Xilinx_Design\ivt
   \Xilinx_Design\planahead
   \Xilinx_Design\results\
   \Xilinx_Design\simulation\
   \Xilinx_Design\source\
   \Xilinx_Design\synthesis\
   ```

4. The `readme.txt` file describes how to use the files that come with `XAPP1104.zip` and the process for using the build scripts.

Chapter 2

Synthesizing Modules for the Isolation Design 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 (`SCC_LAB_TOP`) is synthesized first, followed by each of the lower-level modules (`aes`, `aes_r`, and `compare`) creating four separate ISE® tools HDL projects.

Synthesize the `SCC_LAB_TOP` Module

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

1. Start the ISE design suite 12.4:

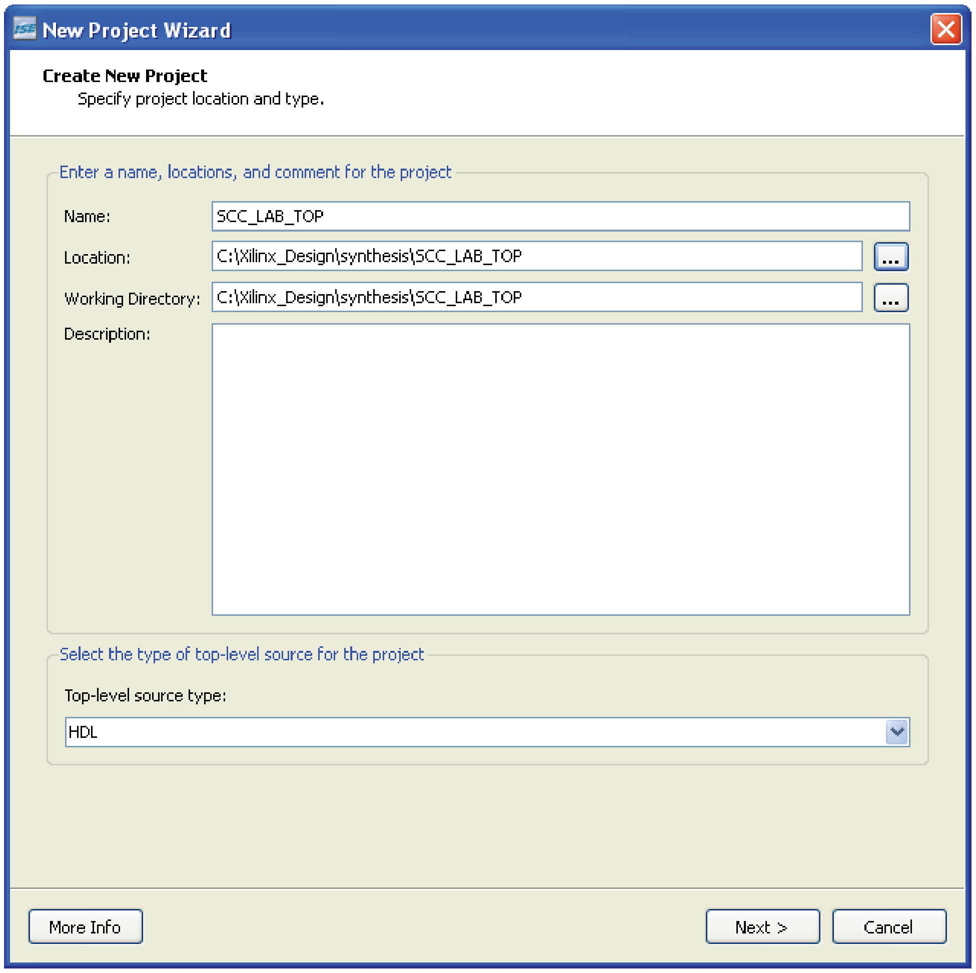
   Start → **All Programs** → **Xilinx ISE Design Suite 12.4** → **ISE Design Tools** → **Project Navigator**

2. Create a new ISE design suite 12.4 project:

   File → **New Project**

3. Set the project location to **`..\Xilinx_Design\synthesis`**.
4. Set the project name to **`SCC_LAB_TOP`**.
5. Click **Next** (see Figure 2-1).

   Note: All requisite files were copied to a user-defined location by the instructions in Installing Reference Design Files into Target Directories, page 12.



Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name:	SCC_LAB_TOP
Location:	C:\Xilinx_Design\synthesis\SCC_LAB_TOP
Working Directory:	C:\Xilinx_Design\synthesis\SCC_LAB_TOP
Description:	

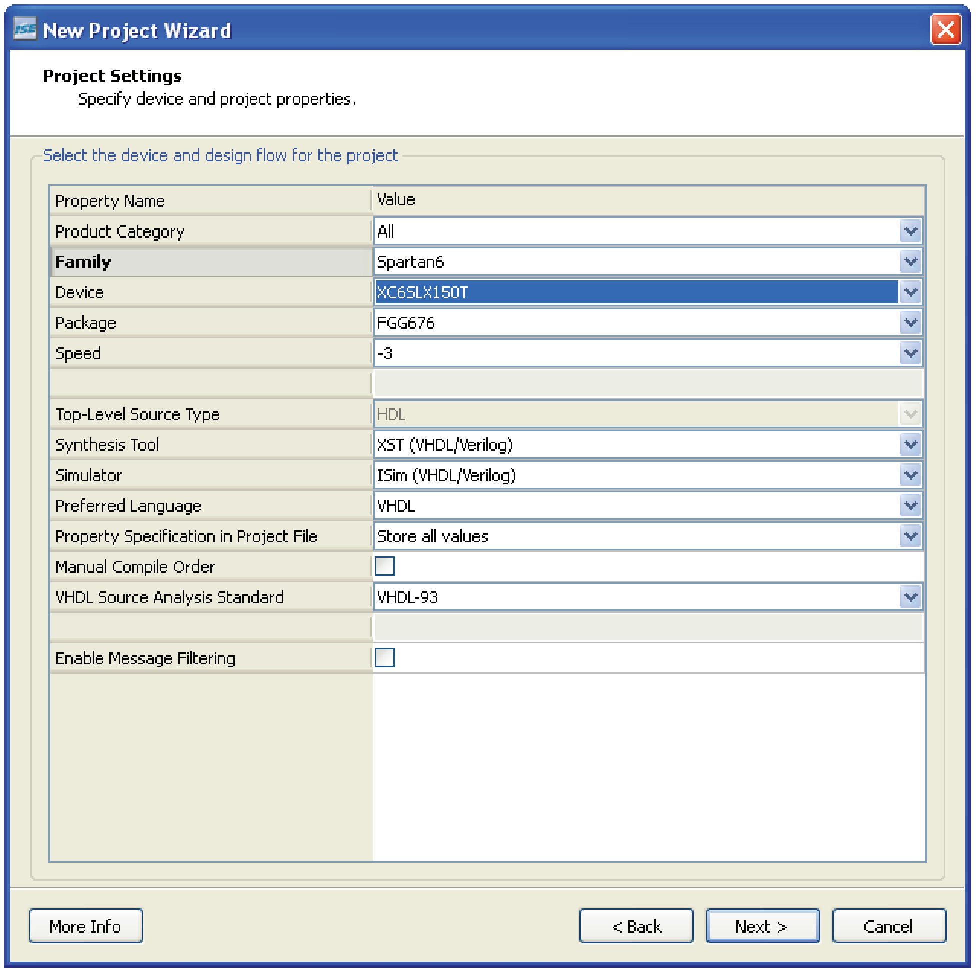
Select the type of top-level source for the project

Top-level source type: HDL

X1104_c2_01_102510

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

6. Make the following selections for the target device (see Figure 2-2):
   - Family: **Spartan6**
   - Device: **XC6SLX150T**
   - Package: **FGG676**
   - Speed: **-3**
   - Preferred Language: **VHDL**


New Project Wizard

Project Settings
Specify device and project properties.

Select the device and design flow for the project

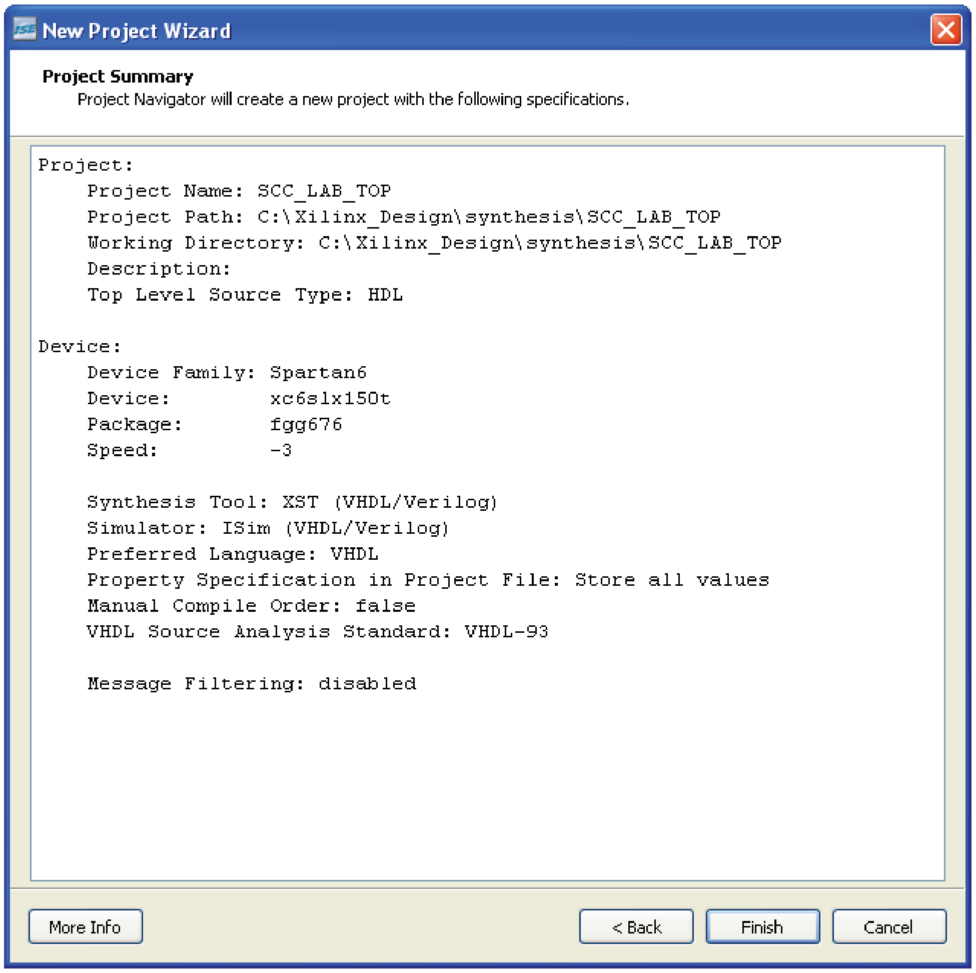
Property Name	Value
Product Category	All
Family	Spartan6
Device	XC6SLX150T
Package	FGG676
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

X1104_c2_02_102510

Figure 2-2: **New Project Wizard (Project Settings)**

7. Click **Next**.

8. Click **Finish** (see Figure 2-3).



Project Summary

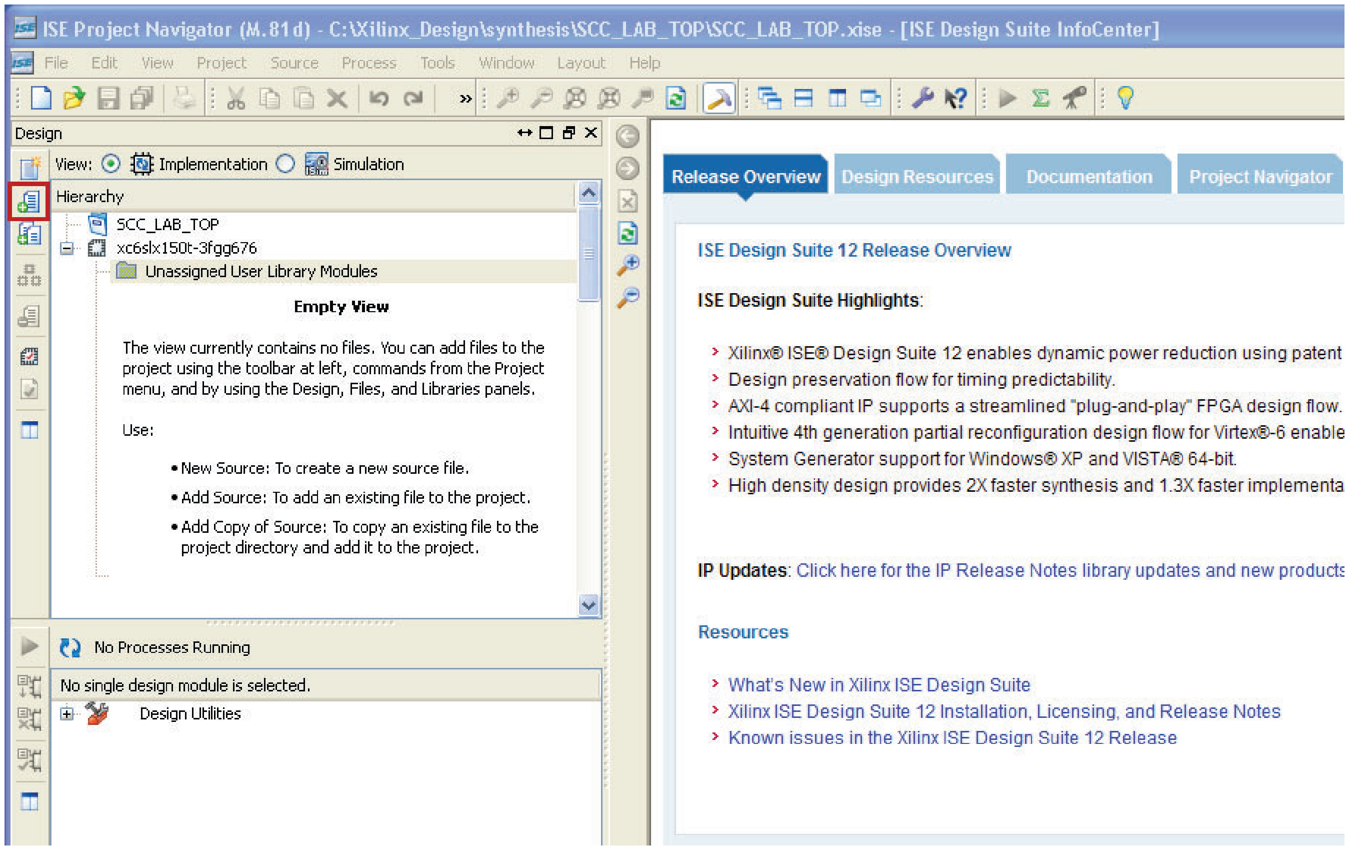
Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: SCC_LAB_TOP
    Project Path: C:\Xilinx_Design\synthesis\SCC_LAB_TOP
    Working Directory: C:\Xilinx_Design\synthesis\SCC_LAB_TOP
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan6
    Device:        xc6slx150t
    Package:       fgg676
    Speed:         -3

    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

X1104_c2_03_102510

Figure 2-3: **New Project Wizard (Project Summary)**

9. Add source files to the project. There are two ways of doing this:
   a. Select **Project → Add Source**.
   b. Click the **Add Source** button on the left side of the Design window in the Project Navigator of the ISE tool (see Figure 2-4).


ISE Project Navigator (M.81d) - C:\Xilinx_Design\synthesis\SCC_LAB_TOP\SCC_LAB_TOP.xise - [ISE Design Suite InfoCenter]

Empty View

The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.

Use:

- New Source: To create a new source file.
- Add Source: To add an existing file to the project.
- Add Copy of Source: To copy an existing file to the project directory and add it to the project.

No single design module is selected.

ISE Design Suite 12 Release Overview

ISE Design Suite Highlights:

- Xilinx® ISE® Design Suite 12 enables dynamic power reduction using patent
- Design preservation flow for timing predictability.
- AXI-4 compliant IP supports a streamlined "plug-and-play" FPGA design flow.
- Intuitive 4th generation partial reconfiguration design flow for Virtex®-6 enable
- System Generator support for Windows® XP and VISTA® 64-bit.
- High density design provides 2X faster synthesis and 1.3X faster implementa

IP Updates: Click here for the IP Release Notes library updates and new products

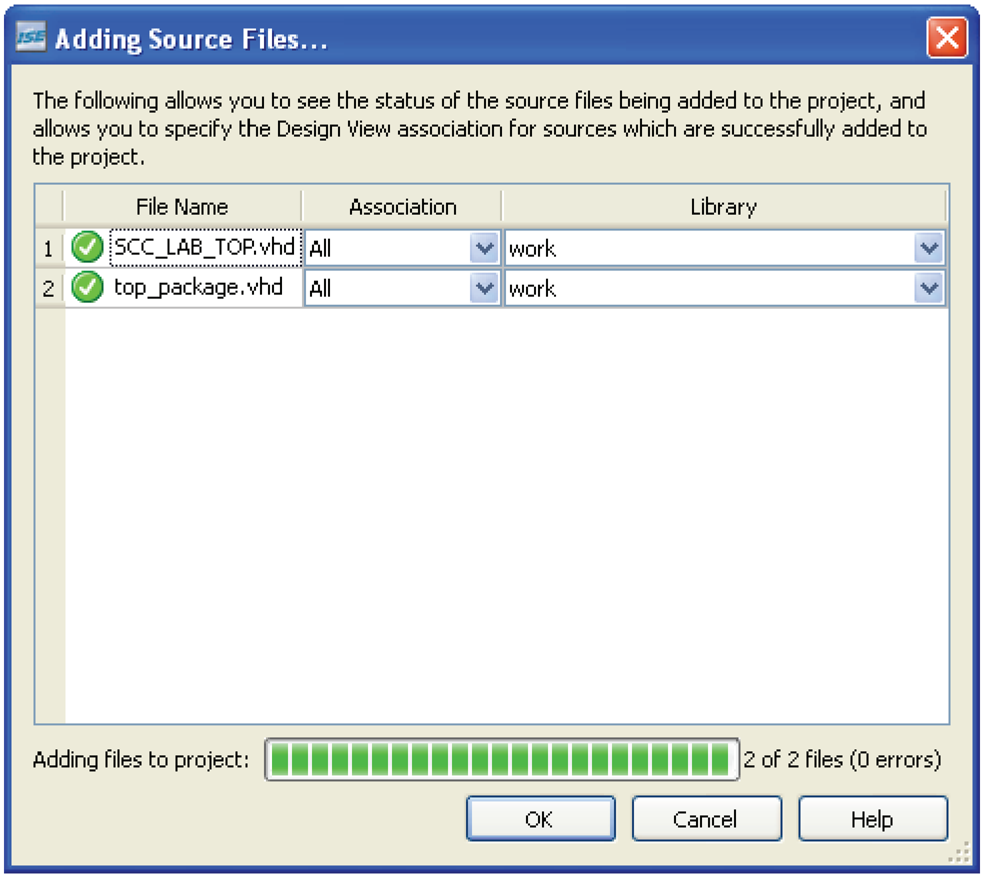
Resources

- What's New in Xilinx ISE Design Suite
- Xilinx ISE Design Suite 12 Installation, Licensing, and Release Notes
- Known issues in the Xilinx ISE Design Suite 12 Release

X1104_c2_04_062411

Figure 2-4: **Project Navigator Window (Add Existing Sources)**

10. Navigate to the `..\Xilinx_Design\source\design` directory, and add the `top_package.vhd` and `SCC_LAB_TOP.vhd` files to the project by selecting a box with the mouse such that both filenames are selected, and then click **Open**. Click **OK** (see Figure 2-5).



Adding Source Files...

The following allows you to see the status of the source files being added to the project, and allows you to specify the Design View association for sources which are successfully added to the project.

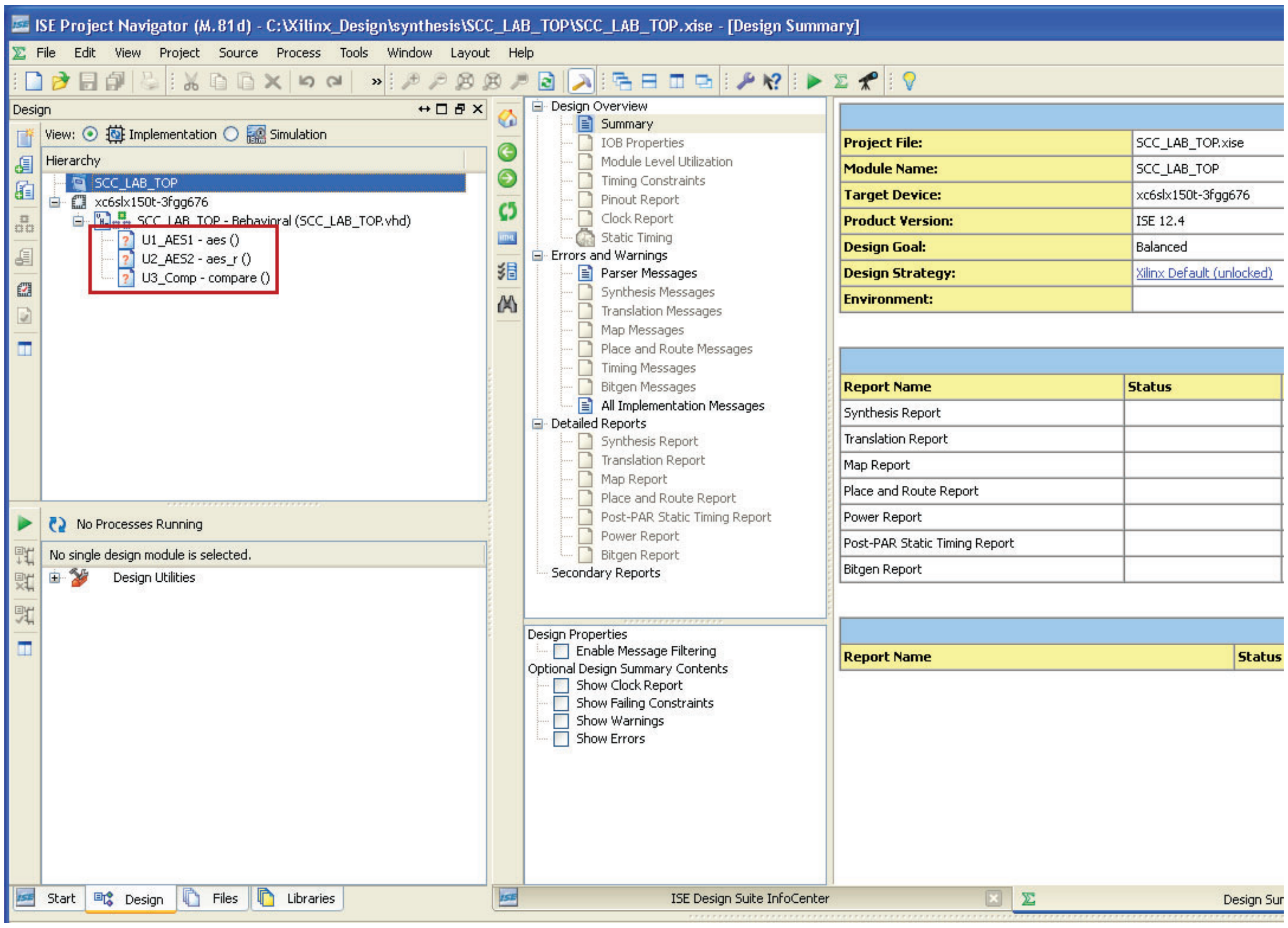
	File Name	Association	Library
1	✓ SCC_LAB_TOP.vhd	All	work
2	✓ top_package.vhd	All	work

Adding files to project: 2 of 2 files (0 errors)

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

The ISE Project Navigator window should look like 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.



Hierarchy:
- SCC_LAB_TOP
- xc6slx150t-3fgg676
  - SCC_LAB_TOP - Behavioral (SCC_LAB_TOP.vhd)
    - U1_AES1 - aes ()
    - U2_AES2 - aes_r ()
    - U3_Comp - compare ()

No single design module is selected.

- Design Overview
  - Summary
  - IOB Properties
  - Module Level Utilization
  - Timing Constraints
  - Pinout Report
  - Clock Report
  - Static Timing
- Errors and Warnings
  - Parser Messages
  - Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - All Implementation Messages
- Detailed Reports
  - Synthesis Report
  - Translation Report
  - Map Report
  - Place and Route Report
  - Post-PAR Static Timing Report
  - Power Report
  - Bitgen Report
- Secondary Reports

Design Properties
- [ ] Enable Message Filtering

Optional Design Summary Contents
- [ ] Show Clock Report
- [ ] Show Failing Constraints
- [ ] Show Warnings
- [ ] Show Errors

Project File:	SCC_LAB_TOP.xise
Module Name:	SCC_LAB_TOP
Target Device:	xc6slx150t-3fgg676
Product Version:	ISE 12.4
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)
Environment:	

Report Name	Status
Synthesis Report	
Translation Report	
Map Report	
Place and Route Report	
Power Report	
Post-PAR Static Timing Report	
Bitgen Report	

Report Name	Status

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

11. Apply attributes to control I/O insertion:

    To include I/O within isolated areas, the I/O buffers must be included at lower levels of the design's hierarchy. This is counter to the normal Xilinx flow that has all device I/O at the top level. To fulfill the needs of the IDF, these two steps must be taken:

    a. Force instantiation of I/O buffers at lower levels

    b. Avoid creation of I/O buffers at the top level

    Both are accomplished by attaching the buffer_type attribute to the signals in the lower level modules. One exception to this rule is that clock networks are not required to be isolated. As such, this attribute should not be applied to the CLK pin of the top-level code.

    For this design, open the `SCC_LAB_TOP.vhd` file and locate the buffer_type attributes:

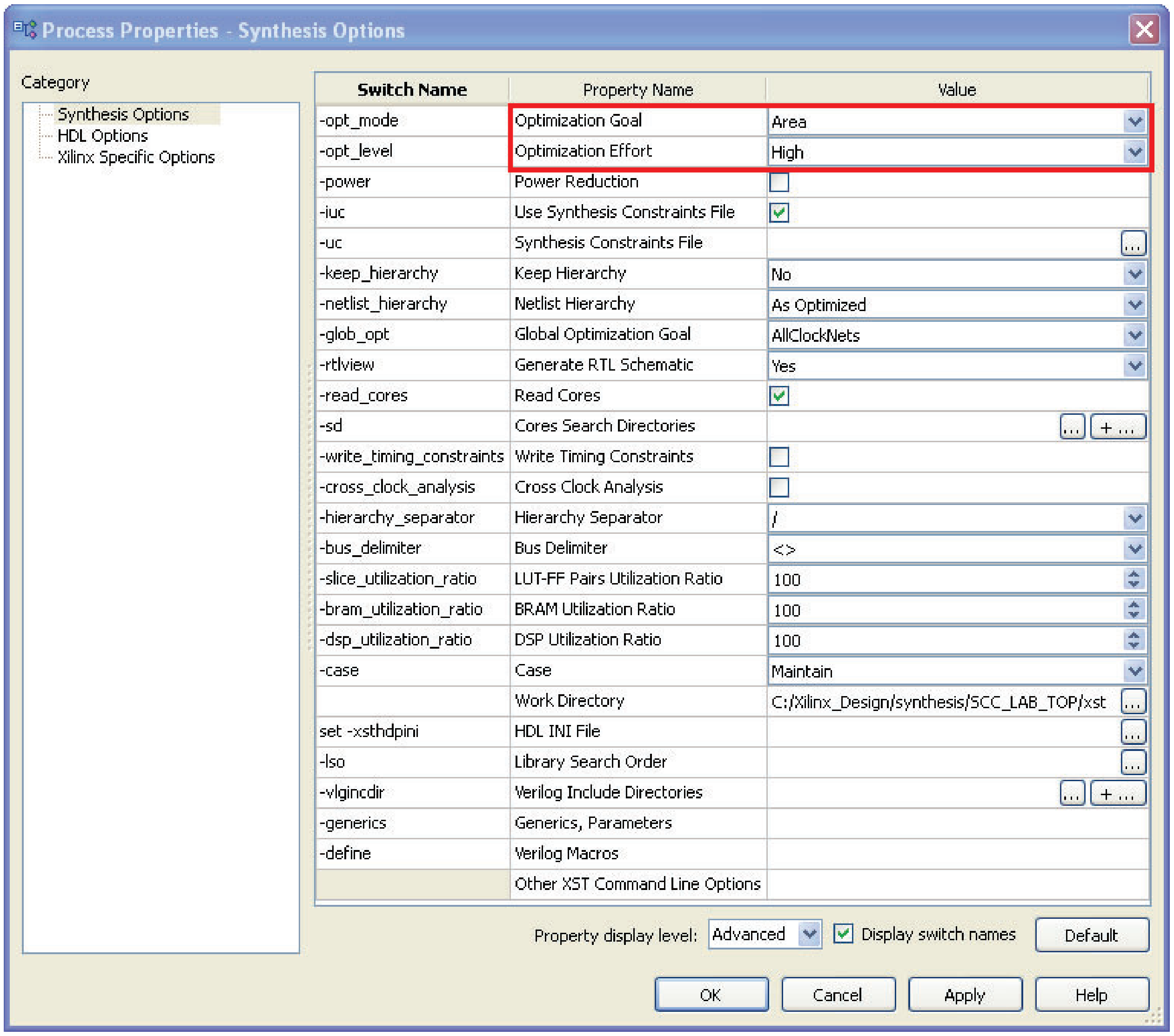
```
attribute buffer_type: string;
attribute buffer_type of push_button    : signal is "none"; -- Pin in AES1
attribute buffer_type of reset        : signal is "none"; -- Pin inside AES1
--attribute buffer_type of clk        : signal is "none"; -- Top Level Pin
attribute buffer_type of led          : signal is "none"; -- Pin in Compare
```

All three signals go off-chip, and must therefore have the buffer_type attribute assigned to direct the Xilinx Synthesis Tool (XST) to disable I/O insertion on these ports. The buffer_type attribute is also needed to guarantee that I/Os are included in the lower level modules, and therefore are part of the isolated regions. In `SCC_LAB_TOP.vhd`, clk is driven to all modules, so no attribute is added and it is therefore commented out. Simple rules for applying the buffer_type attribute are:

- Clock networks are not required to be isolated, and thus do not need this attribute.
- Signals that are present within a single isolated region and get routed off-chip do not get the attribute. (During synthesis, this appears to XST to be a top-level design, and creating top-level I/O buffers is normal operation for XST.)
- Signals on the region's top level that connect to other regions or to the upper level of the design (and are thus not routed off-chip) are assigned the attribute. The designer must manually make this assignment at the top level of the given design region's source code.

Refer to XAPP1145, *Developing Fail-Safe Designs in the Spartan-6 Family Using the Isolation Design Flow*, for more details on IDF off-chip communication concepts.

12. Select **SCC_LAB_TOP - Behavioral** in the Design Hierarchy window, right-click the **Synthesize-XST** icon in the Processes window, and select **Process Properties** (see Figure 2-7).



Switch Name	Property Name	Value
-opt_mode	Optimization Goal	Area
-opt_level	Optimization Effort	High
-power	Power Reduction	☐
-iuc	Use Synthesis Constraints File	✓
-uc	Synthesis Constraints File	
-keep_hierarchy	Keep Hierarchy	No
-netlist_hierarchy	Netlist Hierarchy	As Optimized
-glob_opt	Global Optimization Goal	AllClockNets
-rtlview	Generate RTL Schematic	Yes
-read_cores	Read Cores	✓
-sd	Cores Search Directories	
-write_timing_constraints	Write Timing Constraints	☐
-cross_clock_analysis	Cross Clock Analysis	☐
-hierarchy_separator	Hierarchy Separator	/
-bus_delimiter	Bus Delimiter	<>
-slice_utilization_ratio	LUT-FF Pairs Utilization Ratio	100
-bram_utilization_ratio	BRAM Utilization Ratio	100
-dsp_utilization_ratio	DSP Utilization Ratio	100
-case	Case	Maintain
	Work Directory	C:/Xilinx_Design/synthesis/SCC_LAB_TOP/xst
set -xsthdpini	HDL INI File	
-lso	Library Search Order	
-vlgincdir	Verilog Include Directories	
-generics	Generics, Parameters	
-define	Verilog Macros	
	Other XST Command Line Options	

Property display level: Advanced ✓ Display switch names

Figure 2-7: **Process Properties (Synthesis Options)**

13. Set the Property display level to **Advanced**, set Optimization Goal to **Area**, set Optimization Effort to **High**, and click **OK**. **Keep Hierarchy** does not have to be set on this module for this flow.

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

Synthesize the `aes` Module

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

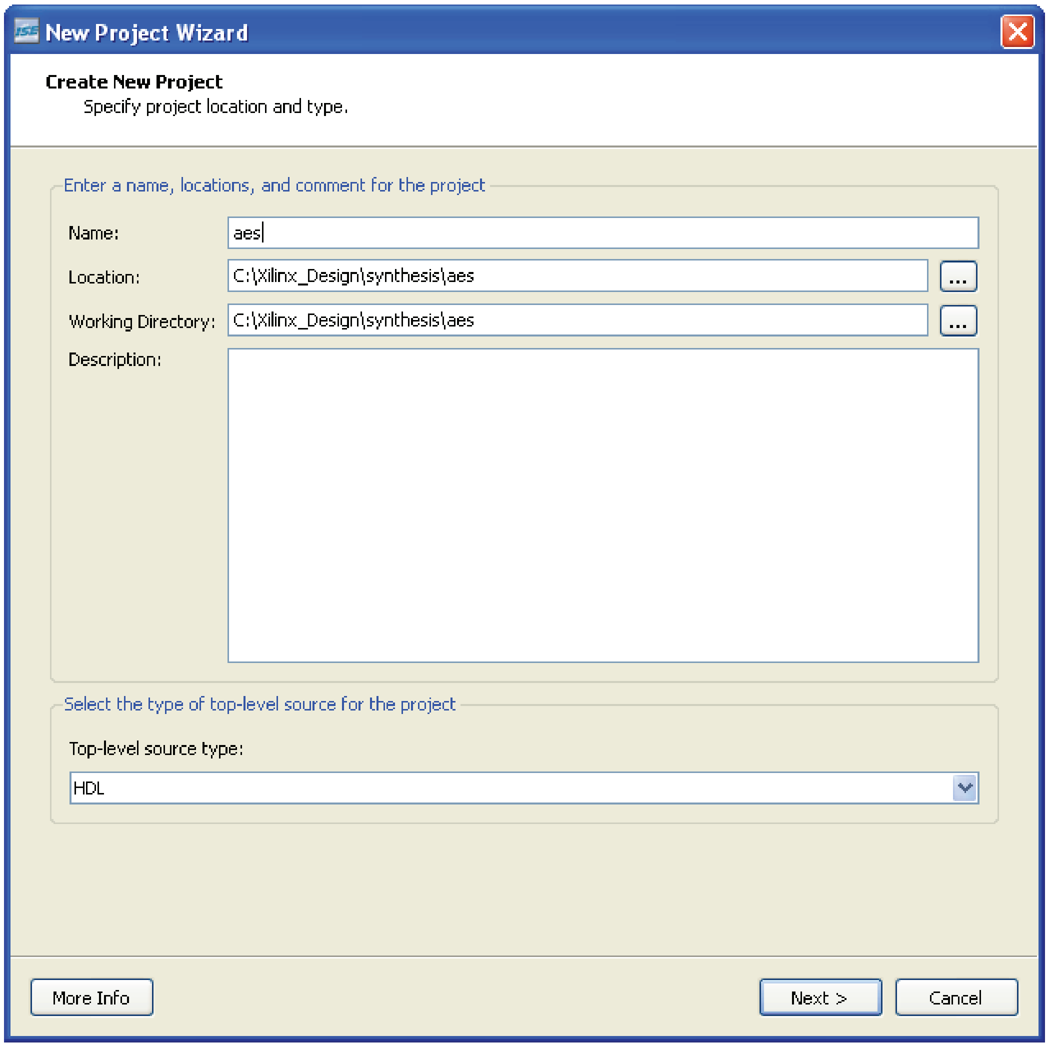
1. Start the ISE design suite 12.4:

   Start → All Programs → Xilinx ISE Design Suite 12.4 → ISE Design Tools → Project Navigator

2. Create a new ISE design suite 12.4 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 2-8).



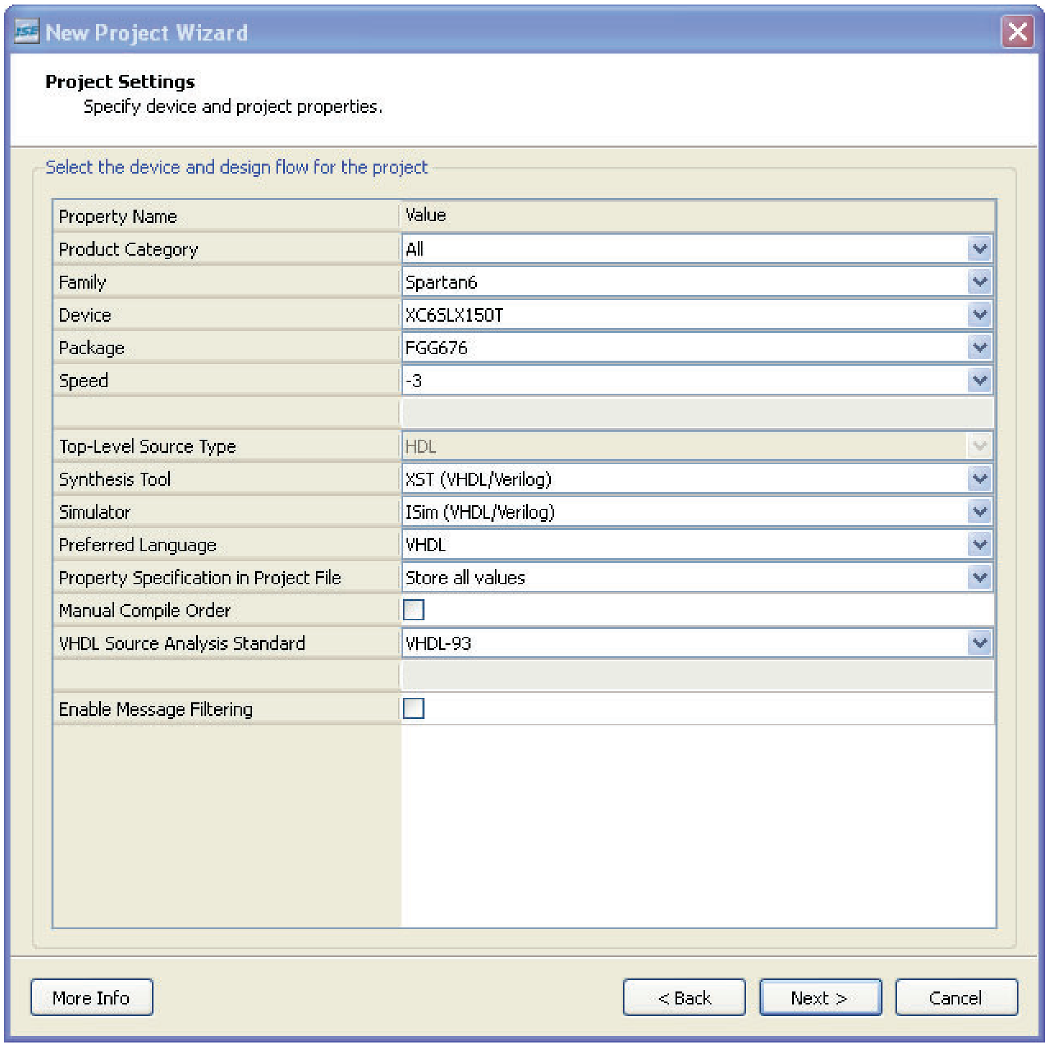
New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project
Select the type of top-level source for the project
More Info
Next >
Cancel

Name: aes
Location: C:\Xilinx_Design\synthesis\aes
Working Directory: C:\Xilinx_Design\synthesis\aes
Description:
Top-level source type: HDL

X1104_c2_08_102510

Figure 2-8: **New Project Wizard (Create New Project)**

6. Make the following selections for the target device (see Figure 2-9):
   - Family: **Spartan6**
   - Device: **XC6SLX150T**
   - Package: **FGG676**
   - Speed: **-3**
   - Preferred Language: **VHDL**


New Project Wizard
Project Settings
Specify device and project properties.
Select the device and design flow for the project
More Info
< Back
Next >
Cancel

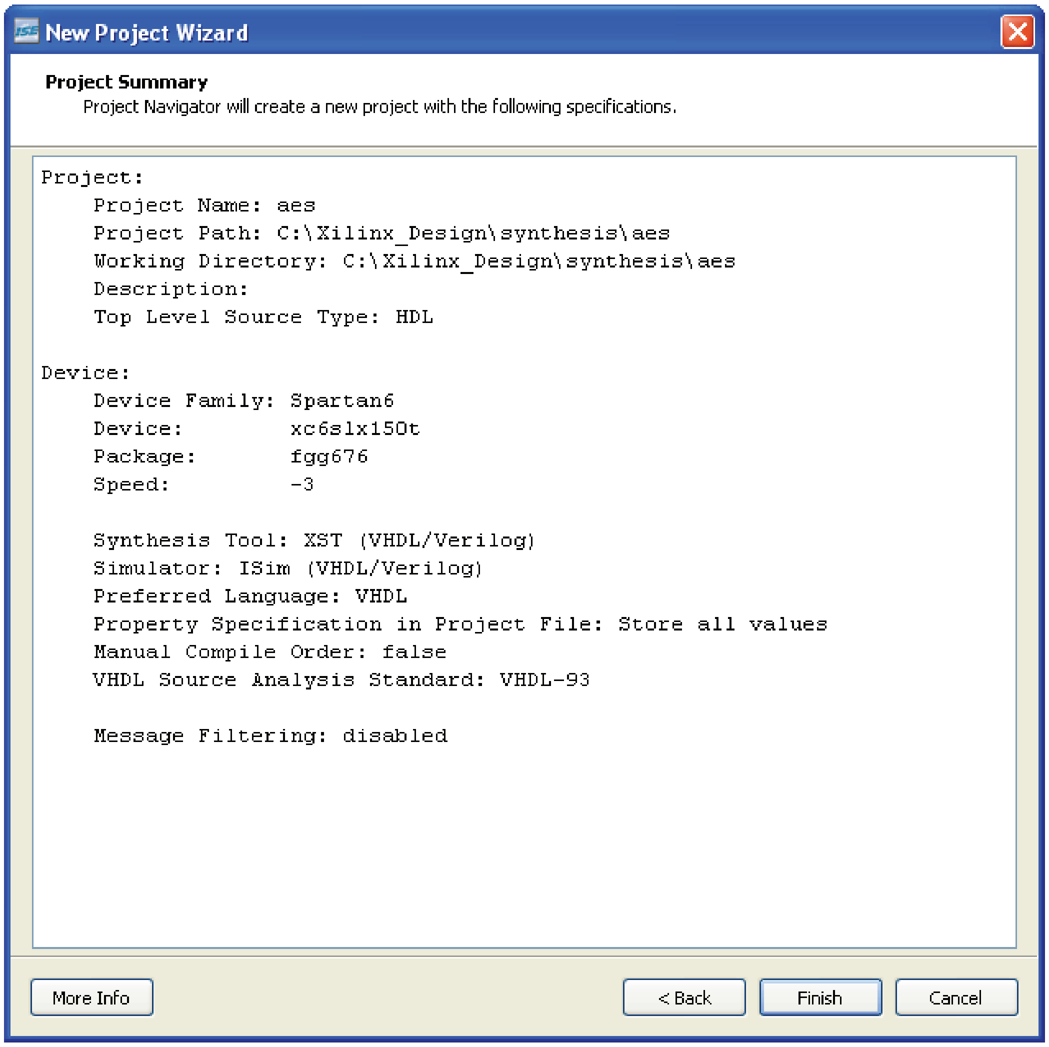
Property Name	Value
Product Category	All
Family	Spartan6
Device	XC6SLX150T
Package	FGG676
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

X1104_c2_09_102510

Figure 2-9: **New Project Wizard (Device Properties)**

7. Click **Next**.

8. Click **Finish** in the Project Summary window (see Figure 2-10).

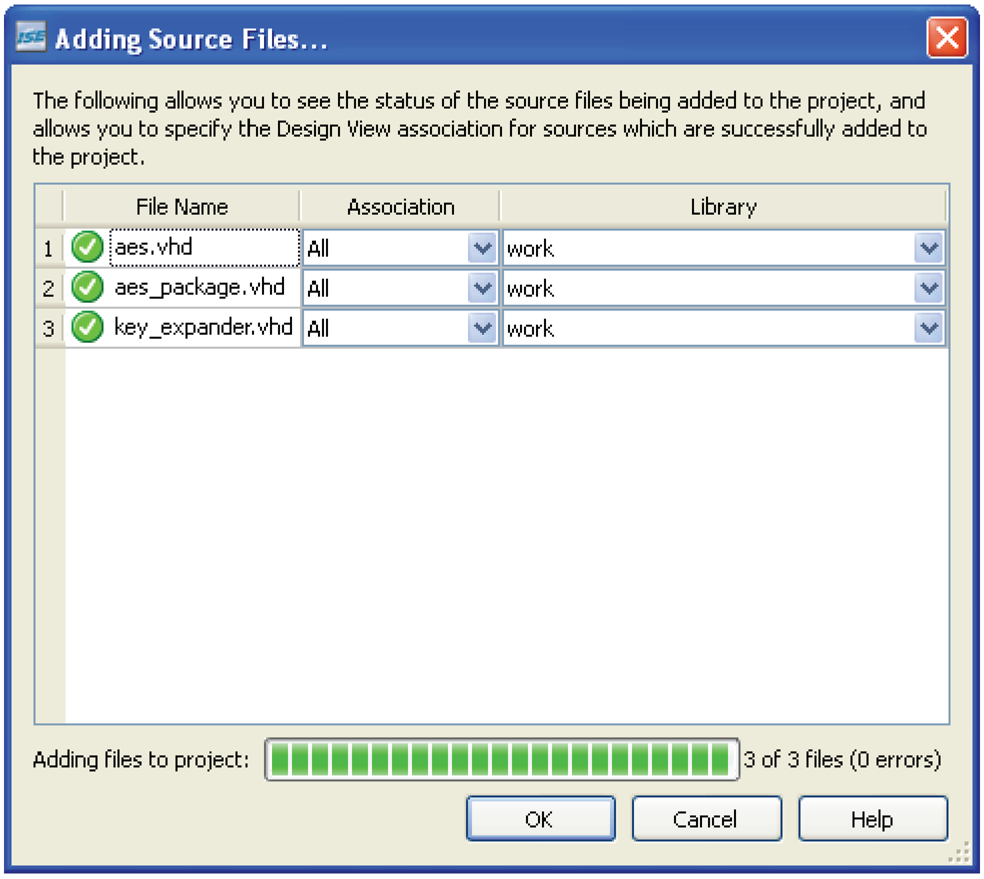


Project Summary
Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: aes
    Project Path: C:\Xilinx_Design\synthesis\aes
    Working Directory: C:\Xilinx_Design\synthesis\aes
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan6
    Device:        xc6slx150t
    Package:       fgg676
    Speed:         -3

    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

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

9. Add source files to the project by selecting **Project** → **Add Source**. Navigate to the `source\design` directory, and add the `aes.vhd`, `aes_package.vhd`, and `key_expander.vhd` files to the project. This is done by holding down the Ctrl key on the keyboard and left-clicking each of the filenames once such that all filenames are selected, then clicking **Open**. Click **OK** (see Figure 2-11).



The following allows you to see the status of the source files being added to the project, and allows you to specify the Design View association for sources which are successfully added to the project.

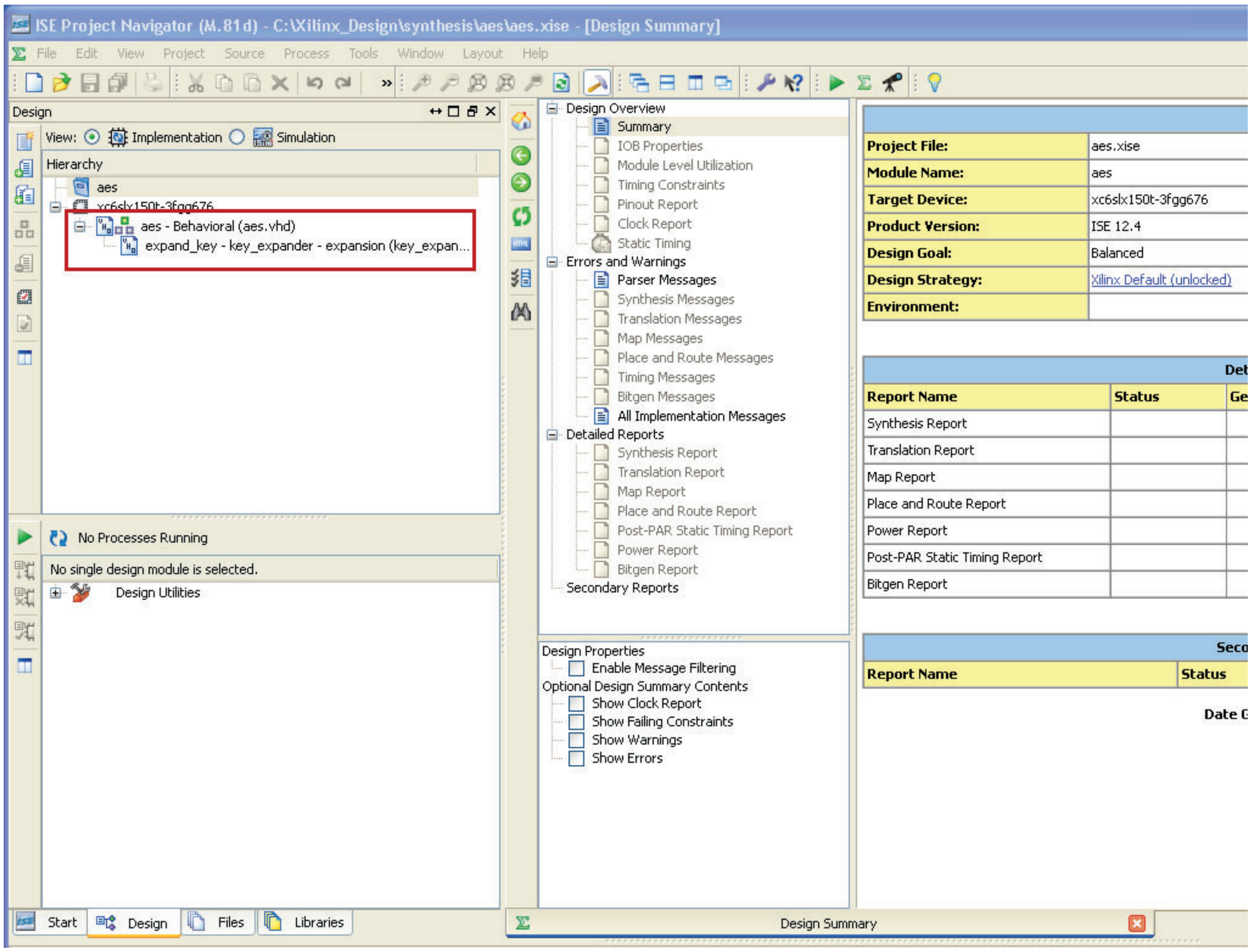
	File Name	Association	Library
1	✓ aes.vhd	All	work
2	✓ aes_package.vhd	All	work
3	✓ key_expander.vhd	All	work

Adding files to project: 3 of 3 files (0 errors)

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

The ISE Project Navigator Window should look as 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.



aes
xc6slx150t-3fgg676
aes - Behavioral (aes.vhd)
expand_key - key_expander - expansion (key_expan...

Project File:	aes.xise
Module Name:	aes
Target Device:	xc6slx150t-3fgg676
Product Version:	ISE 12.4
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)
Environment:	

Report Name	Status	Ge
Synthesis Report		
Translation Report		
Map Report		
Place and Route Report		
Power Report		
Post-PAR Static Timing Report		
Bitgen Report		

Report Name	Status

X1104_c2_12_062411

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

10. Apply attributes to control I/O insertion:

    As in Synthesize the SCC_LAB_TOP Module, page 13, insertion of I/O buffers at this level of hierarchy must be prevented for the top-level signals (such as CLK) within the `aes` block that connect to the upper level of the design. Also, insertion of an I/O buffer on a port (such as DONE) that is a direct connection to a port of another instance must be prevented because no I/O buffer is needed if not going off-chip.

    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";   -- Top level PIN
--attribute buffer_type of reset        : signal is "none";   -- Hooked up to IBUF
--attribute buffer_type of push_button  : signal is "none";   -- Hooked up to IBUF
attribute buffer_type of start          : signal is "none";   -- Port
attribute buffer_type of mode           : signal is "none";   -- Port
attribute buffer_type of load           : signal is "none";   -- Port
attribute buffer_type of key            : signal is "none";   -- Port
attribute buffer_type of data_in        : signal is "none";   -- Port
attribute buffer_type of reset_out      : signal is "none";   -- Port
attribute buffer_type of data_out       : signal is "none";   -- Port
attribute buffer_type of done           : signal is "none";   -- Port
```

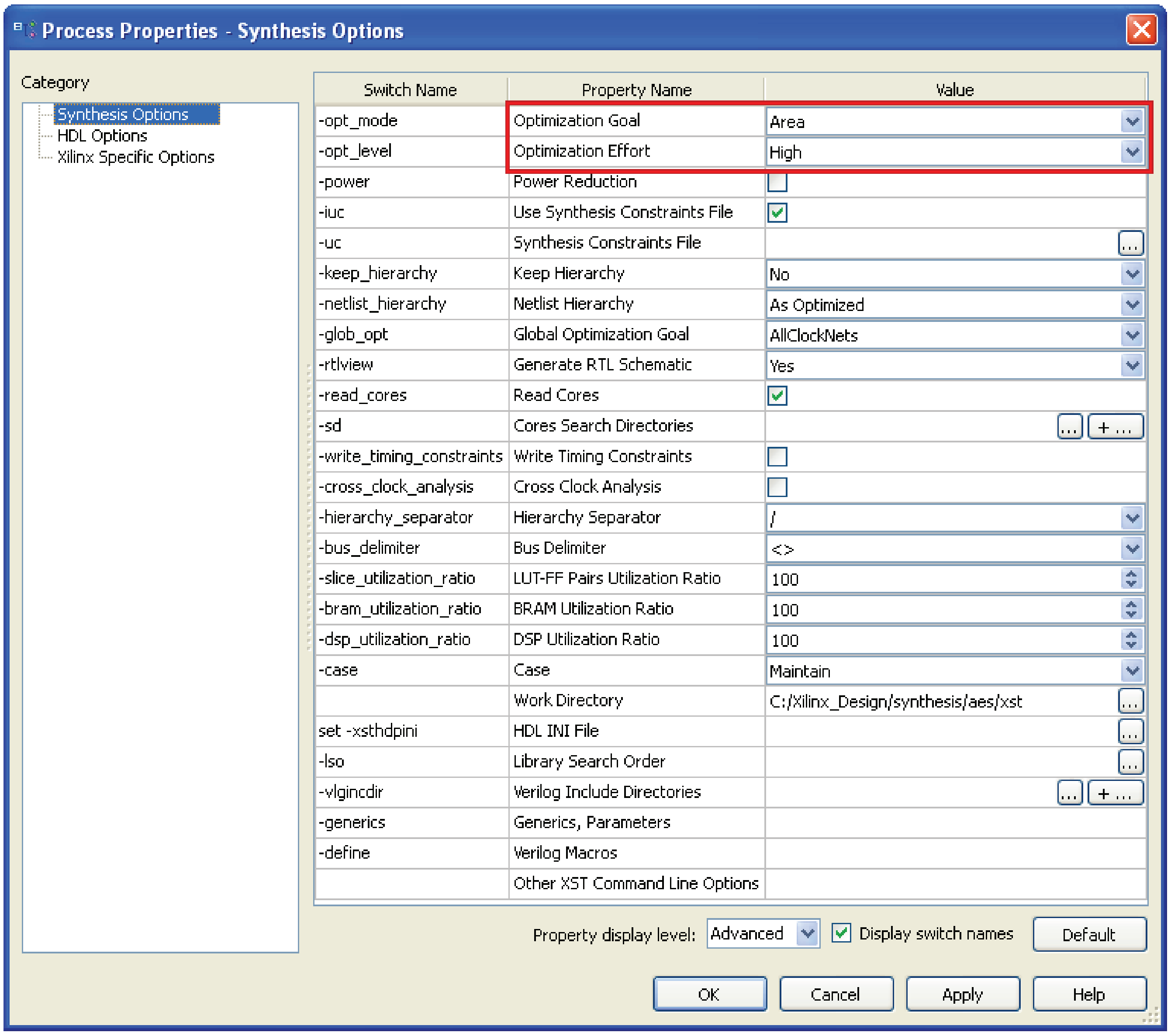
Note: The reset and push_button signals do not have an attribute associated with them and are therefore commented out. They are a connection between the off-chip world and the `aes` module only, i.e., they are "owned" by `aes` and therefore require an I/O to be inferred within the `aes` module rather than within `SCC_LAB_TOP`. The clk I/O was inferred when `SCC_LAB_TOP` was synthesized. The other signals cross the isolation barrier but remain internal to the FPGA; therefore, they have the attribute **`signal is "none"`** applied.

11. Instantiate LUTs per the trusted routing rules:

    In the `aes.vhd` source code file, LUT instantiation between reset and reset_out and on done_aes1 are necessary for trusted routing rules. Trusted routing is a set of resources that is selected automatically by the tools to facilitate communication between isolated regions. These resources serve to prevent "shorted" nets between isolated regions and also preserve one-to-one mapping of input and output ports of isolated regions. Refer to XAPP1145, *Developing Fail-Safe Designs in the Spartan-6 Family Using the Isolation Design Flow,* 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.

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

  lut_done_aes1_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => done_d2, O => done );
```

12. Select **aes - Behavioral** in the Design Hierarchy window, right-click the **Synthesize-XST** icon in the Processes window, and select **Process Properties** (see Figure 2-13).



Switch Name	Property Name	Value
-opt_mode	Optimization Goal	Area
-opt_level	Optimization Effort	High
-power	Power Reduction	☐
-iuc	Use Synthesis Constraints File	✓
-uc	Synthesis Constraints File	
-keep_hierarchy	Keep Hierarchy	No
-netlist_hierarchy	Netlist Hierarchy	As Optimized
-glob_opt	Global Optimization Goal	AllClockNets
-rtlview	Generate RTL Schematic	Yes
-read_cores	Read Cores	✓
-sd	Cores Search Directories	
-write_timing_constraints	Write Timing Constraints	☐
-cross_clock_analysis	Cross Clock Analysis	☐
-hierarchy_separator	Hierarchy Separator	/
-bus_delimiter	Bus Delimiter	<>
-slice_utilization_ratio	LUT-FF Pairs Utilization Ratio	100
-bram_utilization_ratio	BRAM Utilization Ratio	100
-dsp_utilization_ratio	DSP Utilization Ratio	100
-case	Case	Maintain
	Work Directory	C:/Xilinx_Design/synthesis/aes/xst
set -xsthdpini	HDL INI File	
-lso	Library Search Order	
-vlgincdir	Verilog Include Directories	
-generics	Generics, Parameters	
-define	Verilog Macros	
	Other XST Command Line Options	

Property display level: Advanced ✓ Display switch names Default

Figure 2-13: **Process Properties (Synthesis Options)**

13. Set Optimization Goal to **Area**, and set Optimization Effort to **High**. **Keep Hierarchy** does not have to be set.
14. Within the Process Properties window, click **HDL Options** in the column on the left, and uncheck **Resource Sharing**. Click **Xilinx Specific Options** in the column on the left, and uncheck **Equivalent Register Removal**.
15. Click **OK**.
16. To run XST synthesis, either right-click the **Synthesize-XST** icon in the Processes window and select **Run** or double-click **Synthesize-XST**.
17. After synthesis is complete, close the `aes` project.

Synthesize the `aes_r` Module

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

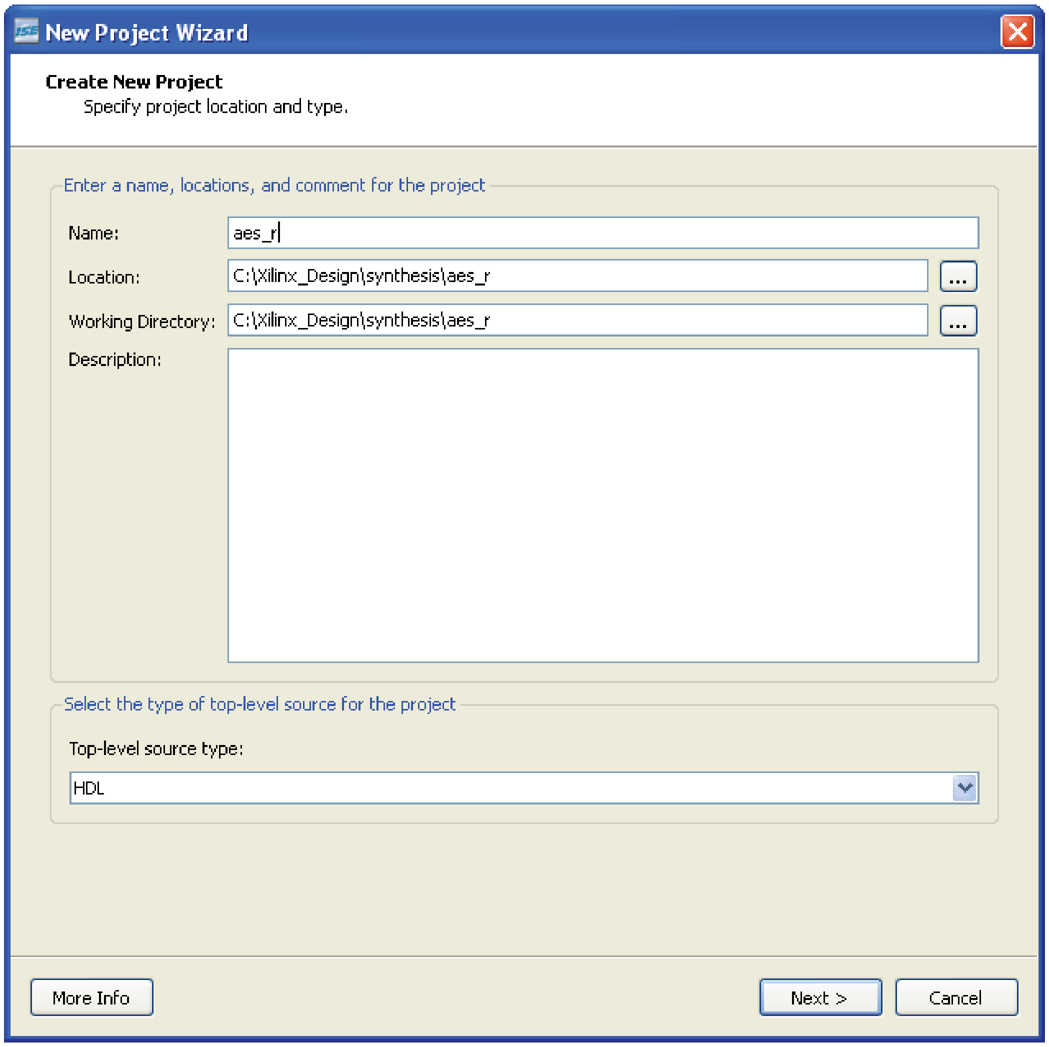
1. Start the ISE design suite 12.4:

   Start → All Programs → Xilinx ISE Design Suite 12.4 → ISE Design Tools → Project Navigator

2. Create a new ISE design suite 12.4 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).


New Project Wizard

Create New Project
Specify project location and type.

Enter a name, locations, and comment for the project

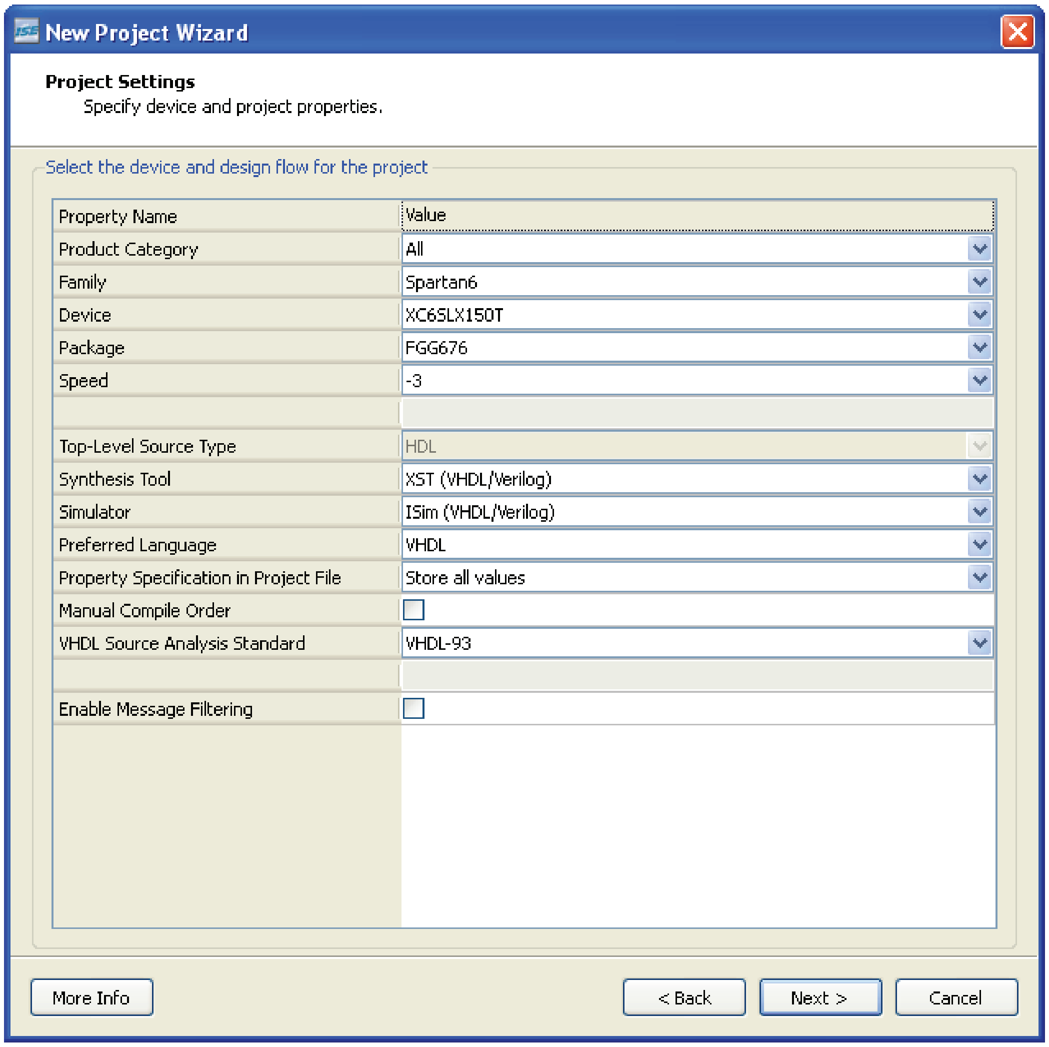
Name: aes_r
Location: C:\Xilinx_Design\synthesis\aes_r
Working Directory: C:\Xilinx_Design\synthesis\aes_r
Description:

Select the type of top-level source for the project

Top-level source type:
HDL

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

6. Make the following selections for the target device (see Figure 2-15):
   - Family: **Spartan6**
   - Device: **XC6SLX150T**
   - Package: **FGG676**
   - Speed: **-3**
   - Preferred Language: **VHDL**


New Project Wizard
Project Settings
Specify device and project properties.
Select the device and design flow for the project
More Info
< Back
Next >
Cancel

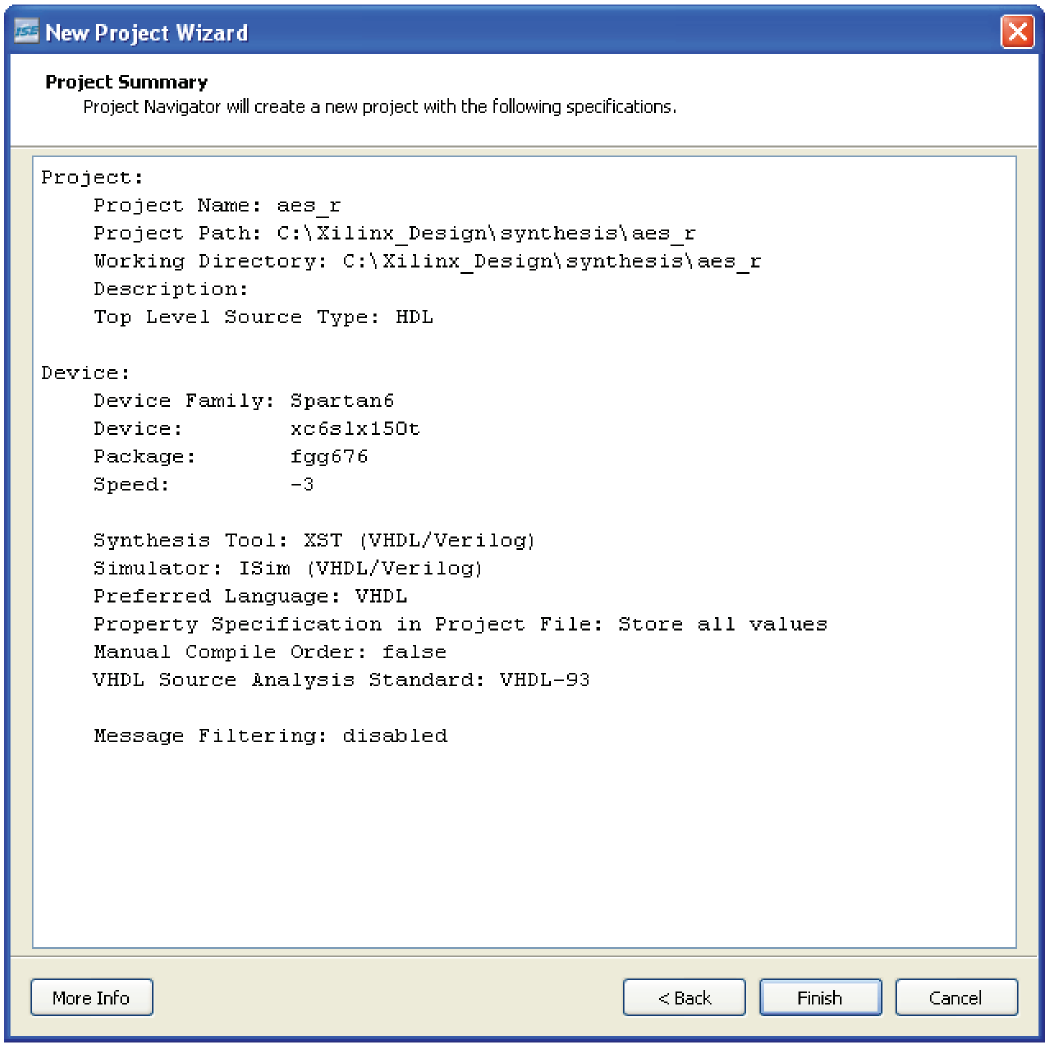
Property Name	Value
Product Category	All
Family	Spartan6
Device	XC6SLX150T
Package	FGG676
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

X1104_c2_15_102510

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

7. Click **Next**.

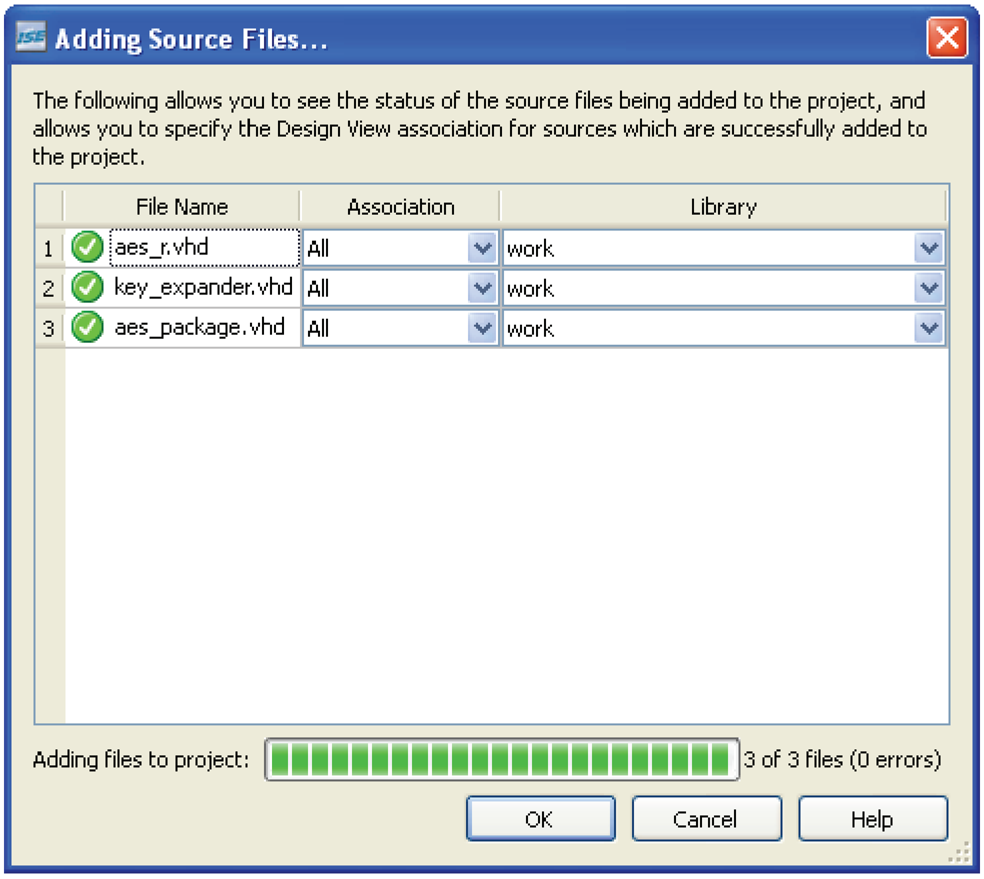
8. Click **Finish** in the Project Summary window (see Figure 2-16).


New Project Wizard
Project Summary
Project Navigator will create a new project with the following specifications.
More Info
< Back
Finish
Cancel
X1104_c2_16_102510

```
Project:
    Project Name: aes_r
    Project Path: C:\Xilinx_Design\synthesis\aes_r
    Working Directory: C:\Xilinx_Design\synthesis\aes_r
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan6
    Device:        xc6slx150t
    Package:       fgg676
    Speed:         -3

    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

Figure 2-16: **New Project Wizard (Project Summary)**

9. Add source files to the project by selecting **Project** → **Add Source**. Navigate to the `source\design` directory and add the `aes_r.vhd`, `aes_package.vhd`, and `key_expander.vhd` files to the project. This is done by holding down the Ctrl key on the keyboard and left-clicking each of the filenames once such that all filenames are selected, then clicking **Open**. Click **OK** (see Figure 2-17).



The following allows you to see the status of the source files being added to the project, and allows you to specify the Design View association for sources which are successfully added to the project.

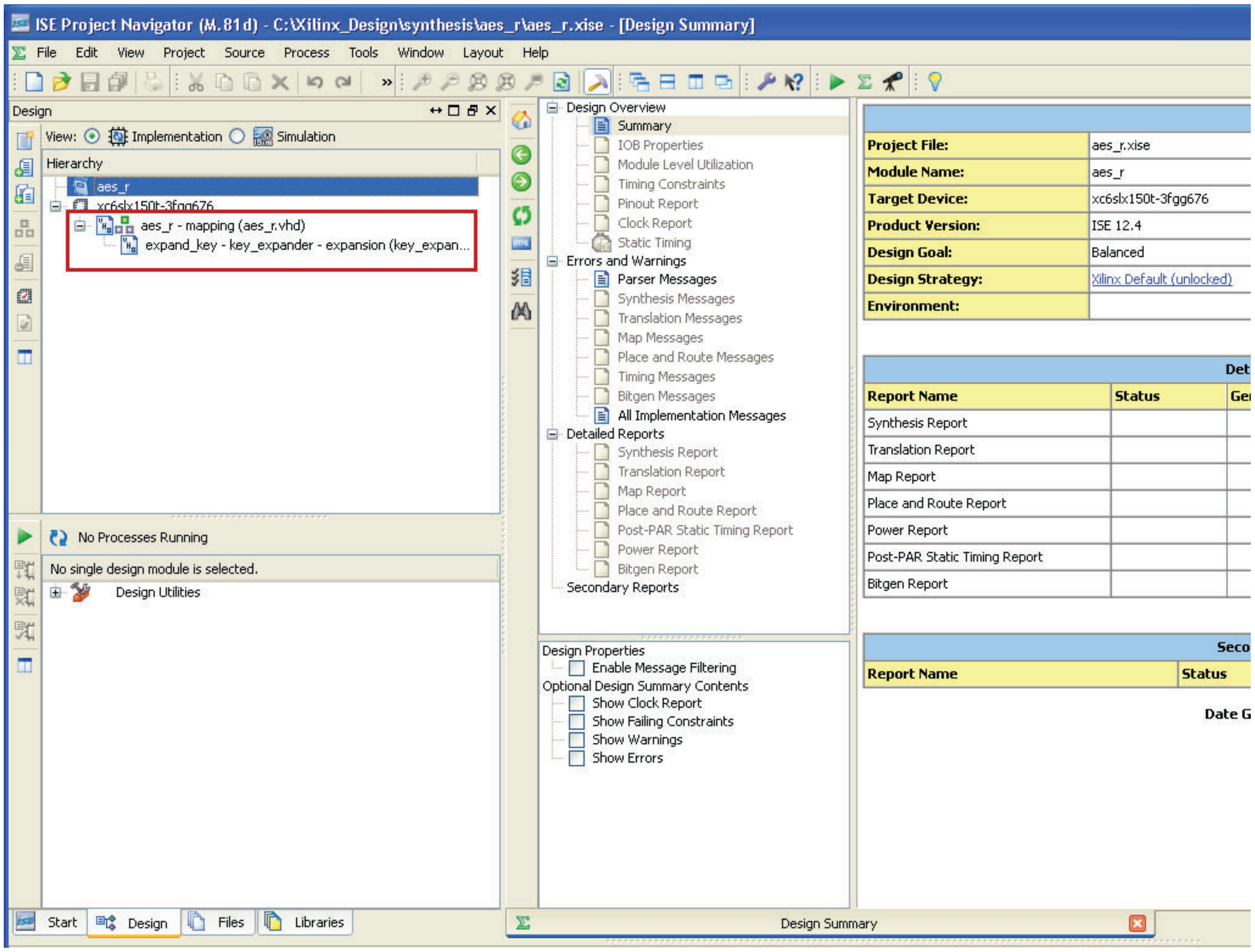
	File Name	Association	Library
1	✓ aes_r.vhd	All	work
2	✓ key_expander.vhd	All	work
3	✓ aes_package.vhd	All	work

Adding files to project: 3 of 3 files (0 errors)

Figure 2-17: **Adding Source Files**

The ISE Project Navigator Window should look like Figure 2-18.

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



ISE Project Navigator (M.81d) - C:\Xilinx_Design\synthesis\aes_r\aes_r.xise - [Design Summary]

Hierarchy
- aes_r
  - xc6slx150t-3fgg676
    - aes_r - mapping (aes_r.vhd)
      - expand_key - key_expander - expansion (key_expan...

No Processes Running

No single design module is selected.

Design Utilities

Project File:	aes_r.xise
Module Name:	aes_r
Target Device:	xc6slx150t-3fgg676
Product Version:	ISE 12.4
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)
Environment:	

Report Name	Status	Ge
Synthesis Report		
Translation Report		
Map Report		
Place and Route Report		
Power Report		
Post-PAR Static Timing Report		
Bitgen Report		

Report Name	Status

Date G

Figure 2-18: **Project Navigator Window**

10. Apply attributes to control I/O insertion:

    As in Synthesize the SCC_LAB_TOP Module, page 13, insertion of I/O buffers at this level of hierarchy must be prevented for the top-level signals (such as CLK) within the aes_r block that connect to the upper level of the design, or because the port is a direct connection to a port of another instance.

    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";
```

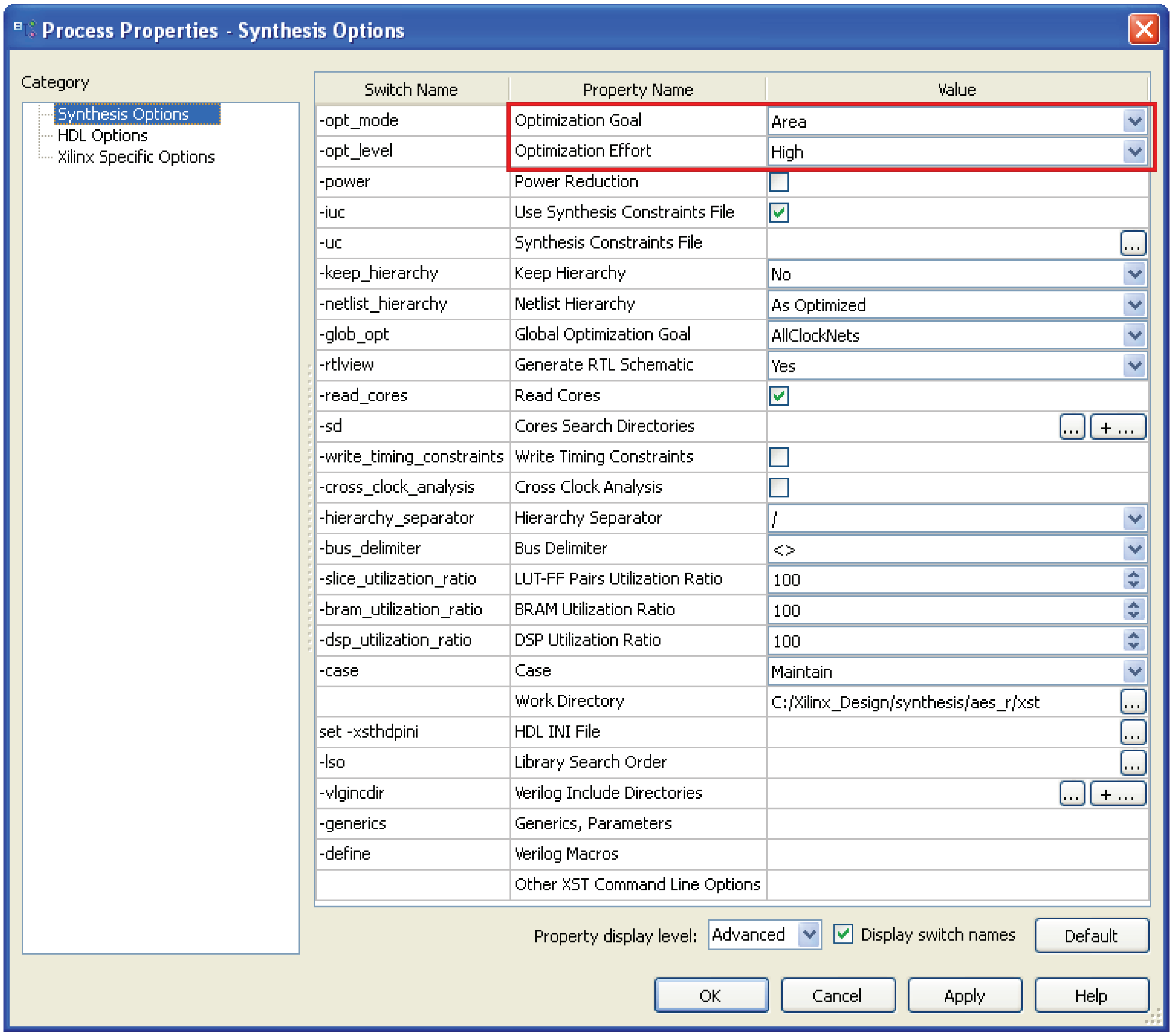
Note: The clk I/O is inferred when `SCC_LAB_TOP` is synthesized. All other ports are internal to the FPGA; therefore, they have the attribute **`signal is "none"`** applied.

11. Instantiate LUTs per the Trusted Routing rules:

    In the `aes_r.vhd` source code file, LUT instantiation on done_aes2 is necessary for trusted routing rules. Trusted routing is a set of resources that is selected automatically by the tools to facilitate communication between isolated regions. These resources serve to prevent "shorted" nets between isolated regions and also preserve one-to-one mapping of input and output ports of isolated regions. Refer to XAPP1145, *Developing Fail-Safe Designs in the Spartan-6 Family Using the Isolation Design Flow,* 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.

  lut_done_aes2_out : LUT1
  GENERIC MAP (INIT => X"2")
  PORT MAP (I0 => done_d2, O => done );
```

12. Right-click the **Synthesize-XST** icon in the Processes window and select **Process Properties** (see Figure 2-19).



Process Properties - Synthesis Options

Category
- Synthesis Options
- HDL Options
- Xilinx Specific Options

Switch Name	Property Name	Value
-opt_mode	Optimization Goal	Area
-opt_level	Optimization Effort	High
-power	Power Reduction	☐
-iuc	Use Synthesis Constraints File	✓
-uc	Synthesis Constraints File	
-keep_hierarchy	Keep Hierarchy	No
-netlist_hierarchy	Netlist Hierarchy	As Optimized
-glob_opt	Global Optimization Goal	AllClockNets
-rtlview	Generate RTL Schematic	Yes
-read_cores	Read Cores	✓
-sd	Cores Search Directories	
-write_timing_constraints	Write Timing Constraints	☐
-cross_clock_analysis	Cross Clock Analysis	☐
-hierarchy_separator	Hierarchy Separator	/
-bus_delimiter	Bus Delimiter	<>
-slice_utilization_ratio	LUT-FF Pairs Utilization Ratio	100
-bram_utilization_ratio	BRAM Utilization Ratio	100
-dsp_utilization_ratio	DSP Utilization Ratio	100
-case	Case	Maintain
	Work Directory	C:/Xilinx_Design/synthesis/aes_r/xst
set -xsthdpini	HDL INI File	
-lso	Library Search Order	
-vlgincdir	Verilog Include Directories	
-generics	Generics, Parameters	
-define	Verilog Macros	
	Other XST Command Line Options	

Property display level: Advanced ✓ Display switch names

Figure 2-19: **Process Properties (Synthesis Options)**

13. Set Optimization Goal to **Area**, and set Optimization Effort to **High**. Keep Hierarchy does not have to be set.
14. Within the Process Properties window, click **HDL Options** in the column on the left, and uncheck **Resource Sharing**. Click **Xilinx Specific Options** in the column on the left, and uncheck **Equivalent Register Removal**.
15. Click **OK**.
16. To run XST synthesis, either right-click the **Synthesize-XST** icon in the Processes window and select **Run**, or double-click **Synthesize-XST**.
17. After synthesis is complete, close the `aes_r` project.

Synthesize the `compare` Module

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

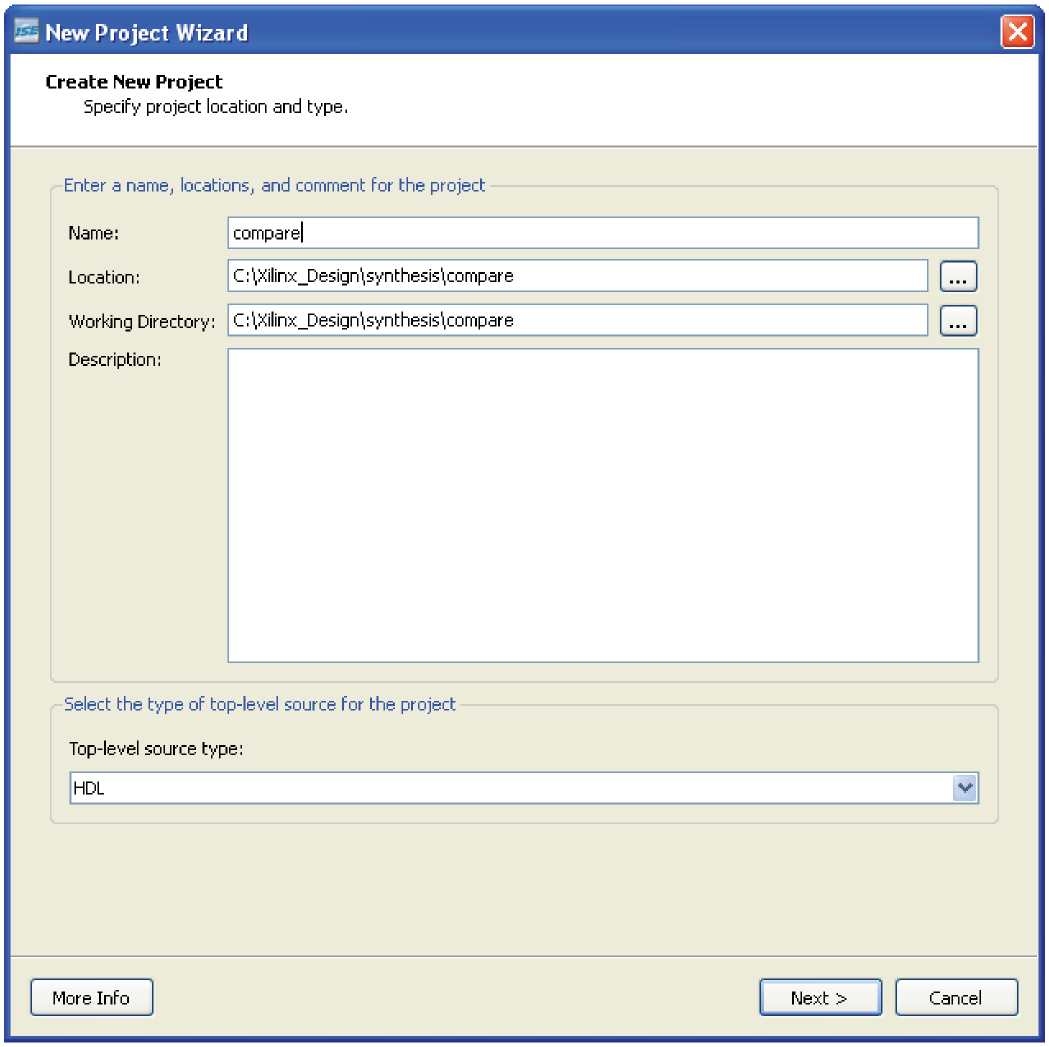
1. Start the ISE design suite 12.4:

   Start → **All Programs** → **Xilinx ISE Design Suite 12.4** → **ISE Design Tools** → **Project Navigator**

2. Create a new ISE design suite 12.4 project:

   File → **New Project**

3. Set the project location to **`\Xilinx_Design\synthesis`**.
4. Set the project name to **compare**.
5. Click **Next** (see Figure 2-20).



Create New Project
Specify project location and type.

Enter a name, locations, and comment for the project

Name: compare

Location: C:\Xilinx_Design\synthesis\compare

Working Directory: C:\Xilinx_Design\synthesis\compare

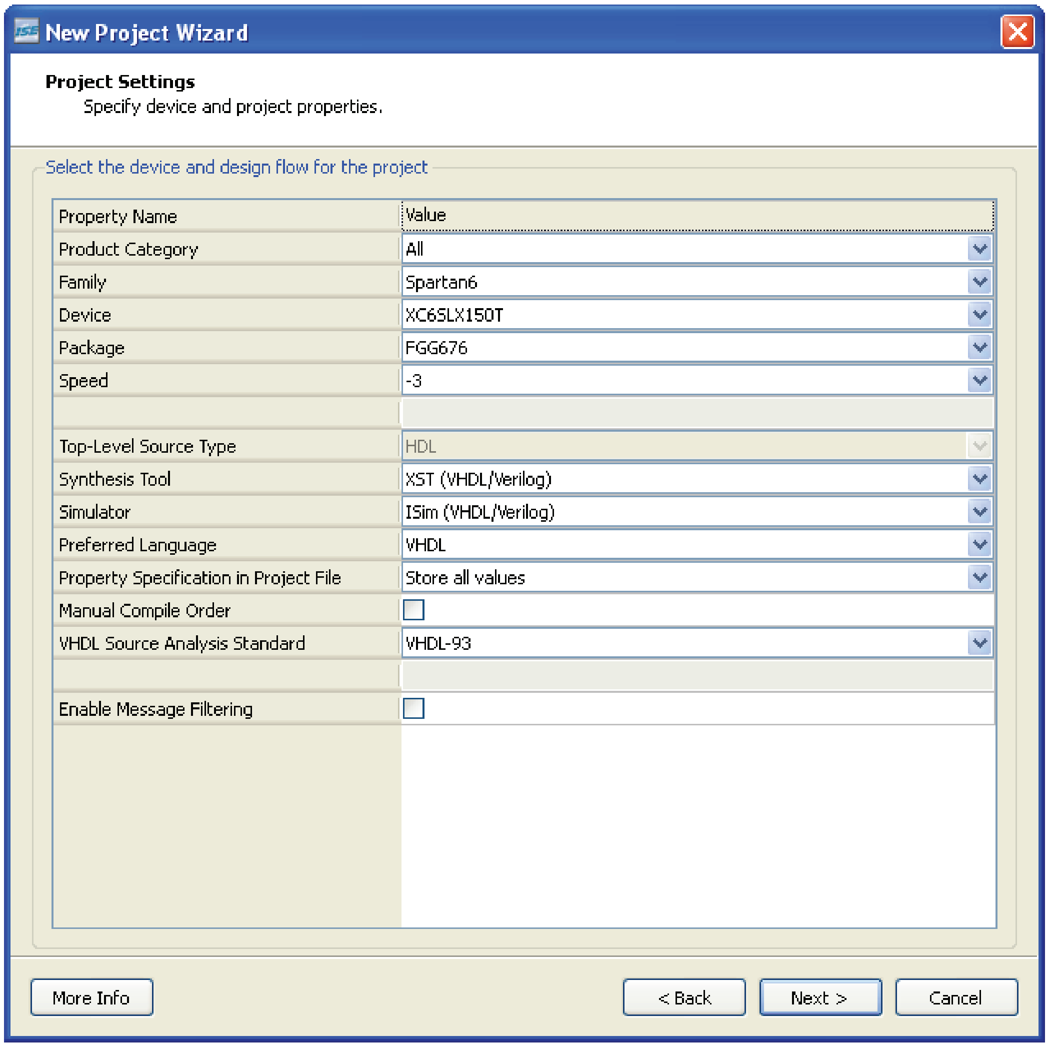
Description:

Select the type of top-level source for the project

Top-level source type: HDL

Figure 2-20: **New Project Wizard (Create New Project)**

6. Make the following selections for the target device (see Figure 2-21):
   - Family: **Spartan6**
   - Device: **XC6SLX150T**
   - Package: **FGG676**
   - Speed: **-3**
   - Preferred Language: **VHDL**



Project Settings
Specify device and project properties.

Select the device and design flow for the project

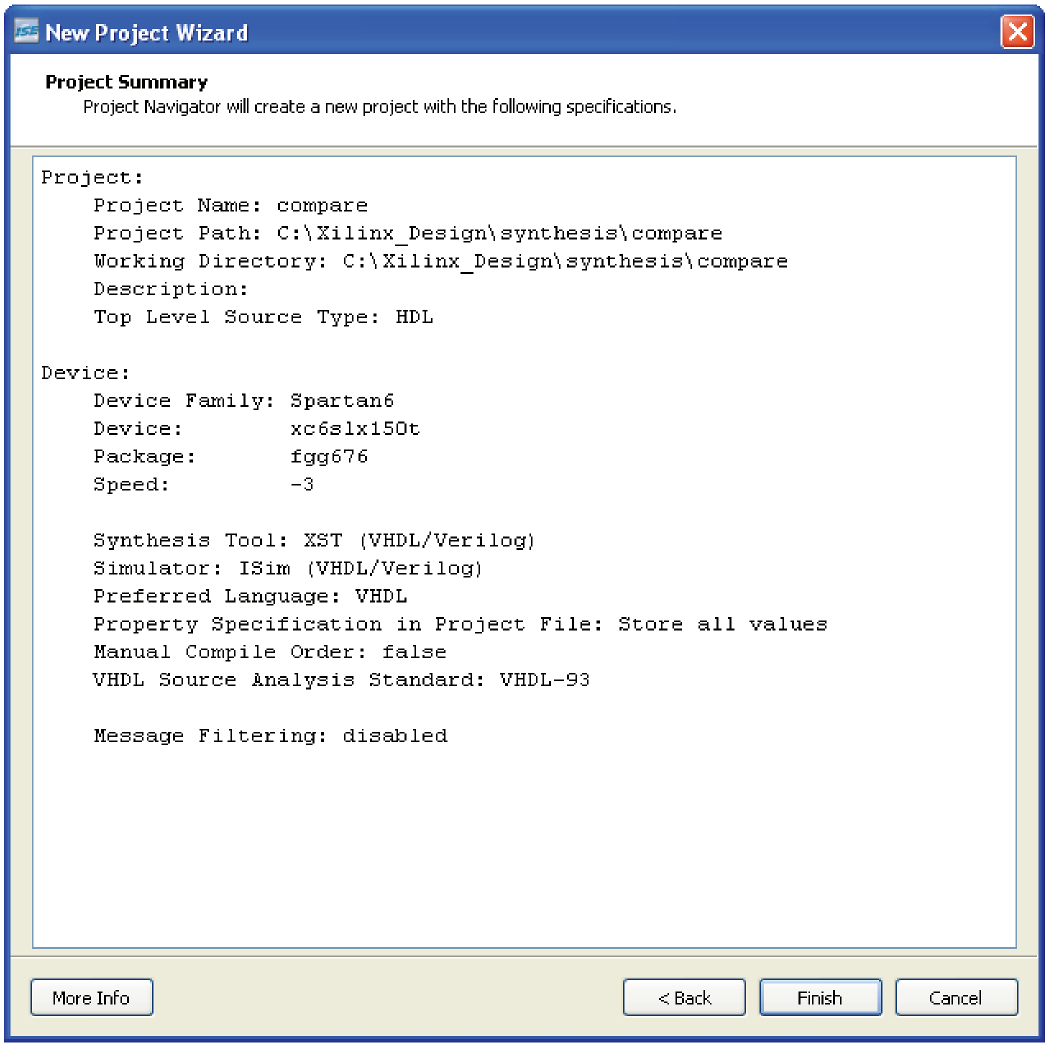
Property Name	Value
Product Category	All
Family	Spartan6
Device	XC6SLX150T
Package	FGG676
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

X1104_c2_21_102510

Figure 2-21: **New Project Wizard (Device Properties)**

7. Click **Next**.

8. Click **Finish** in the Project Summary window (see Figure 2-22).

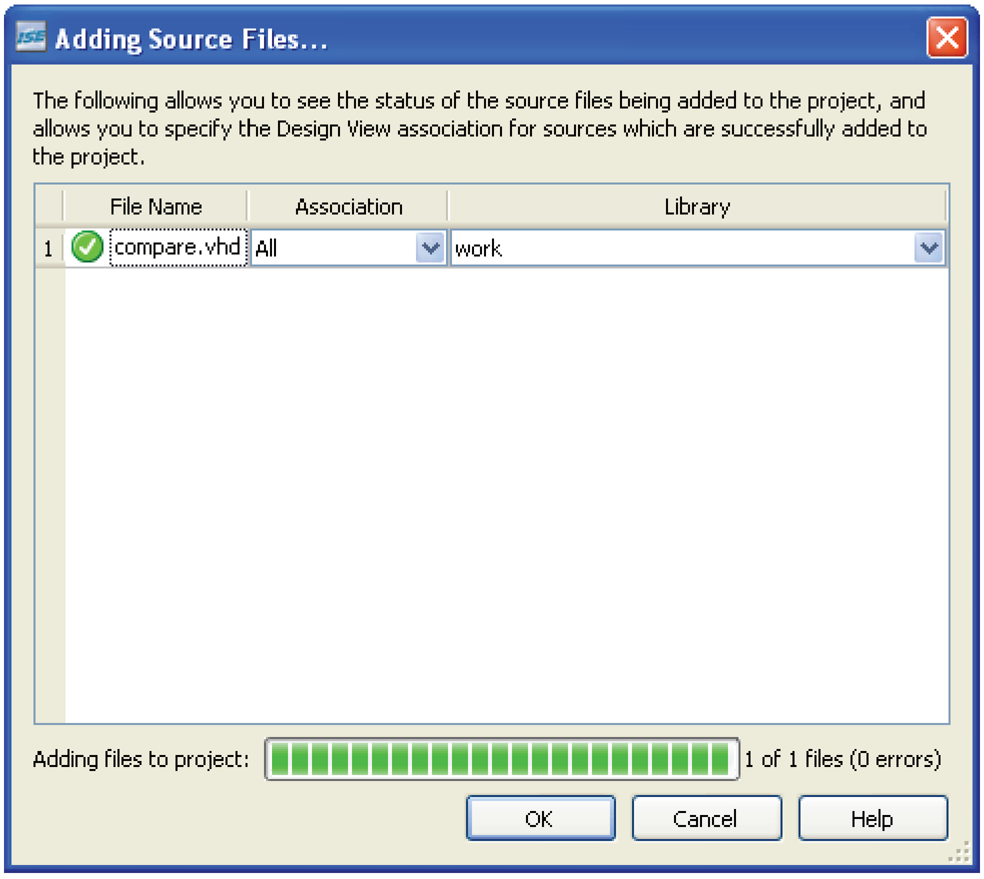


Project Summary
Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: compare
    Project Path: C:\Xilinx_Design\synthesis\compare
    Working Directory: C:\Xilinx_Design\synthesis\compare
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan6
    Device:        xc6slx150t
    Package:       fgg676
    Speed:         -3

    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

Figure 2-22: **New Project Wizard (Project Summary)**

9. Add source files to the project by selecting **Project** → **Add Source**. Navigate to the `source\design` directory, and add the `compare.vhd` by left-clicking once on the filename such that it is selected, then clicking **Open**. Click **OK** (see Figure 2-23).



The following allows you to see the status of the source files being added to the project, and allows you to specify the Design View association for sources which are successfully added to the project.

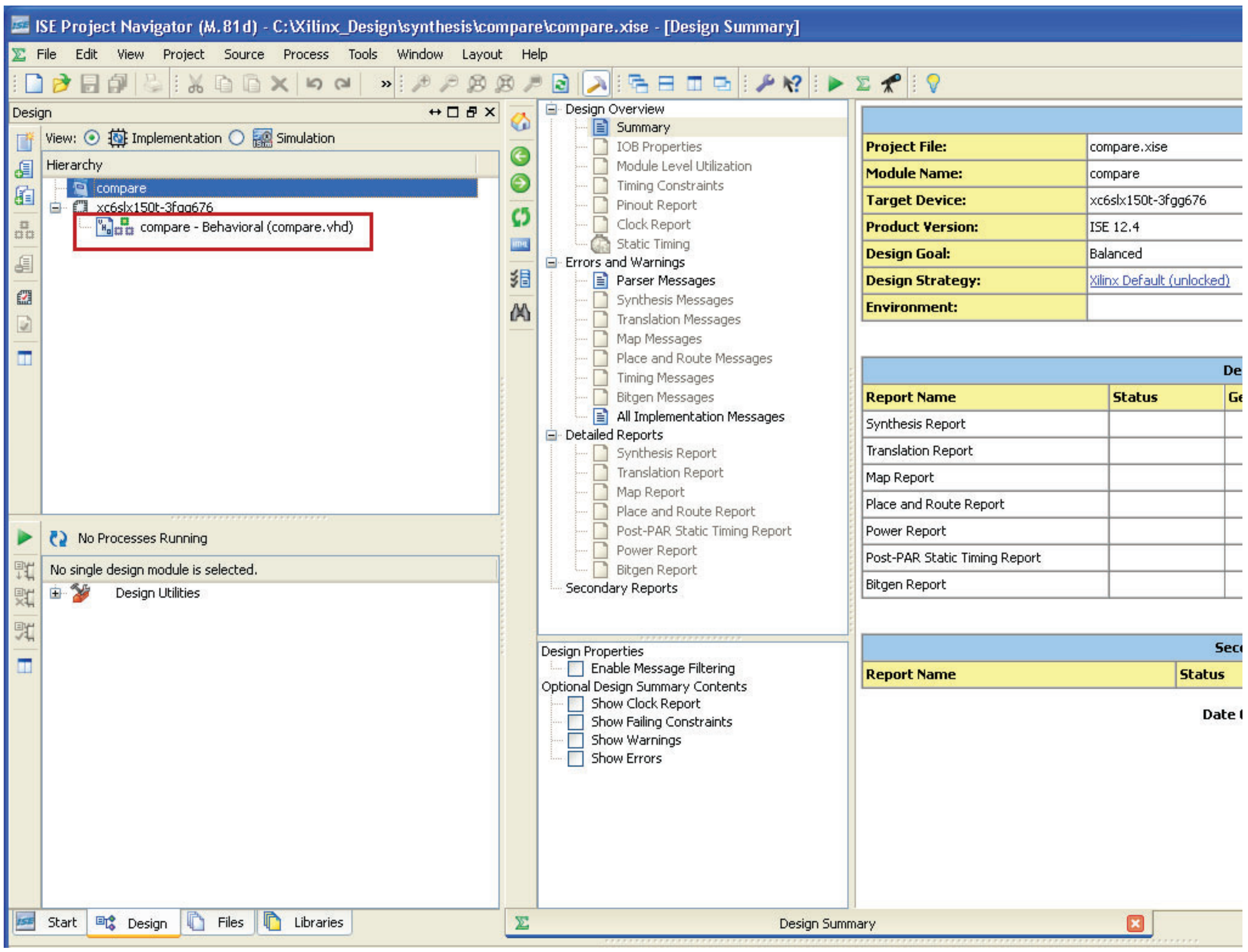
	File Name	Association	Library
1	✓ compare.vhd	All	work

Adding files to project: 1 of 1 files (0 errors)

Figure 2-23: **Adding Source Files**

The ISE Project Navigator Window should look like Figure 2-24.

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



Project File:	compare.xise
Module Name:	compare
Target Device:	xc6slx150t-3fgg676
Product Version:	ISE 12.4
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)
Environment:	

Report Name	**Status**	**Ge**
Synthesis Report		
Translation Report		
Map Report		
Place and Route Report		
Power Report		
Post-PAR Static Timing Report		
Bitgen Report		

Report Name	**Status**

X1104_c2_24_062411

Figure 2-24: **Project Navigator Window**

10. Apply attributes to control I/O insertion:

    As in Synthesize the SCC_LAB_TOP Module, page 13, insertion of I/O buffers at this level of hierarchy must be prevented for the top-level signals (such as CLK) within the `compare` block that connect to the upper level of the design, or because the port is a direct connection to a port of another instance.

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

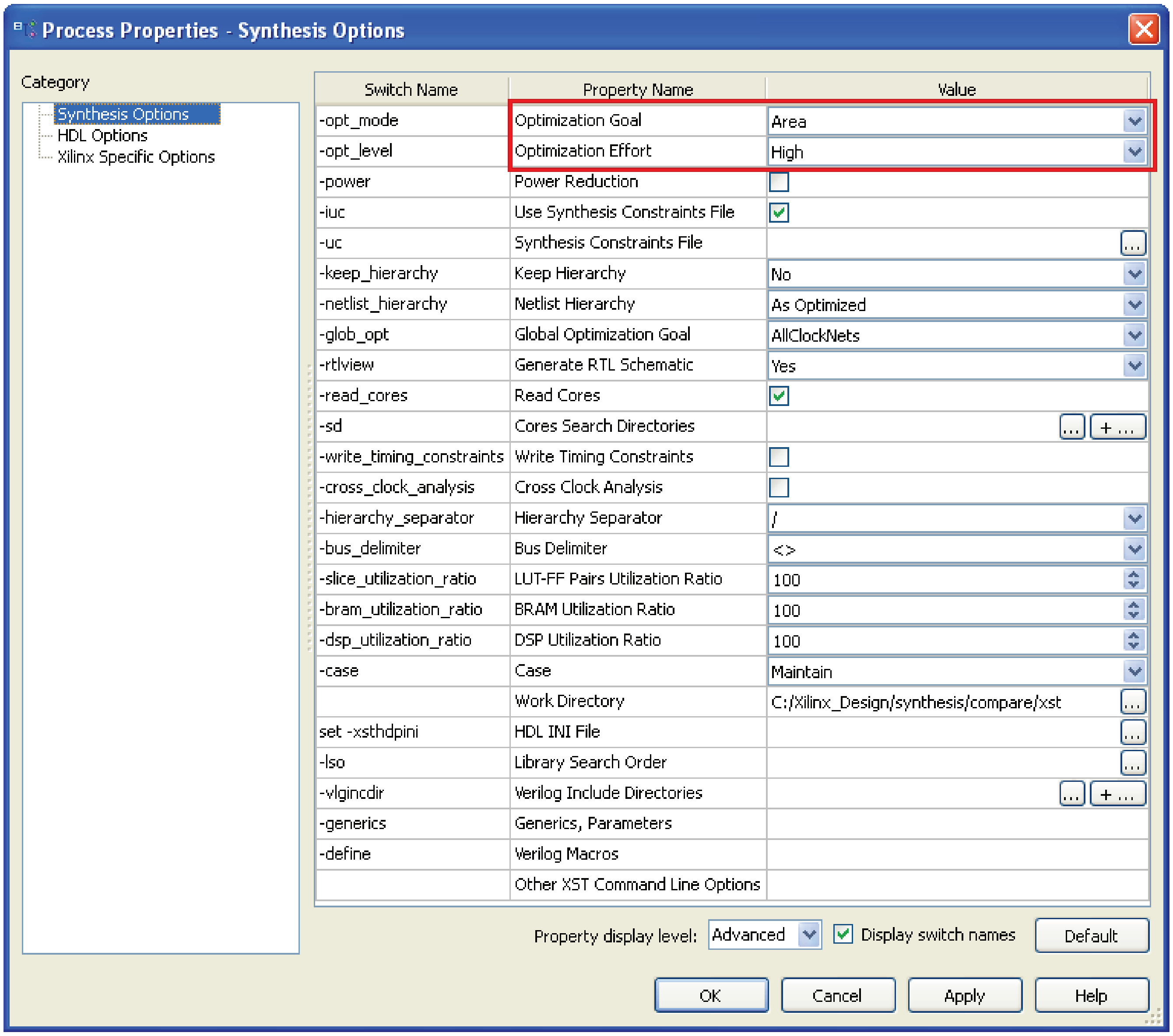
```
attribute buffer_type: string;
--attribute buffer_type of led         : signal is "none"; -- HOOKED UP TO OBUF
attribute buffer_type of clk         : signal is "none"; -- Port
attribute buffer_type of reset       : signal is "none"; -- Port
attribute buffer_type of reset_out   : signal is "none"; -- Port
attribute buffer_type of done1       : signal is "none"; -- Port
attribute buffer_type of done2       : signal is "none"; -- Port
attribute buffer_type of start_aes1    : signal is "none"; -- Port
attribute buffer_type of start_aes2    : signal is "none"; -- Port
attribute buffer_type of load_aes1     : signal is "none"; -- Port
attribute buffer_type of load_aes2     : signal is "none"; -- Port
attribute buffer_type of data_in1      : signal is "none"; -- Port
attribute buffer_type of data_in2      : signal is "none"; -- Port
```

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

11. In the `compare.vhd` source code file, LUT instantiation between certain inputs and outputs is necessary for trusted routing rules. Refer to XAPP1145, *Developing Fail-Safe Designs in the Spartan-6 Family Using the Isolation Design Flow,* 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.

  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 );
```

12. Right-click the **Synthesize-XST** icon in the Processes window and select **Process Properties** (see Figure 2-25).



Process Properties - Synthesis Options

Category
- Synthesis Options
- HDL Options
- Xilinx Specific Options

Switch Name	Property Name	Value
-opt_mode	Optimization Goal	Area
-opt_level	Optimization Effort	High
-power	Power Reduction	☐
-iuc	Use Synthesis Constraints File	✓
-uc	Synthesis Constraints File	
-keep_hierarchy	Keep Hierarchy	No
-netlist_hierarchy	Netlist Hierarchy	As Optimized
-glob_opt	Global Optimization Goal	AllClockNets
-rtlview	Generate RTL Schematic	Yes
-read_cores	Read Cores	✓
-sd	Cores Search Directories	
-write_timing_constraints	Write Timing Constraints	☐
-cross_clock_analysis	Cross Clock Analysis	☐
-hierarchy_separator	Hierarchy Separator	/
-bus_delimiter	Bus Delimiter	<>
-slice_utilization_ratio	LUT-FF Pairs Utilization Ratio	100
-bram_utilization_ratio	BRAM Utilization Ratio	100
-dsp_utilization_ratio	DSP Utilization Ratio	100
-case	Case	Maintain
	Work Directory	C:/Xilinx_Design/synthesis/compare/xst
set -xsthdpini	HDL INI File	
-lso	Library Search Order	
-vlgincdir	Verilog Include Directories	
-generics	Generics, Parameters	
-define	Verilog Macros	
	Other XST Command Line Options	

Property display level: Advanced ✓ Display switch names

X1104_c2_25_102510

Figure 2-25: **Process Properties (Synthesis Options)**

13. Set the Optimization Goal to **Area** and the Optimization Effort to **High**. Keep Hierarchy does not have to be set.
14. Within the Process Properties window, click **HDL Options** in the column on the left and uncheck **Resource Sharing**. Click **Xilinx Specific Options** in the column on the left and uncheck **Equivalent Register Removal**.
15. Click **OK**.
16. To run XST synthesis, either right-click the **Synthesize-XST** icon in the Processes window and select **Run**, or double-click **Synthesize-XST**.
17. After synthesis is complete, close the `compare` project.

Isolation Design Flow Progress

The Module Synthesis block of the isolation design flow diagram is complete, as shown in Figure 2-26.

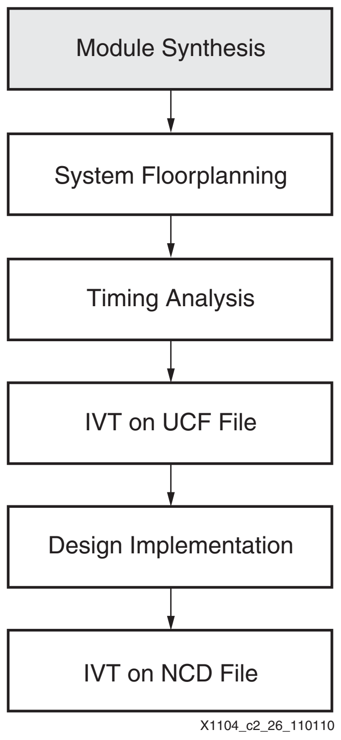


Figure 2-26: **Isolation Design Flow with Module Synthesis Block Complete**

Chapter 3

Floorplanning the System

Project Entry in PlanAhead Tool

Launch the PlanAhead Tool

To launch the 12.4 version of the PlanAhead™ tool, select:

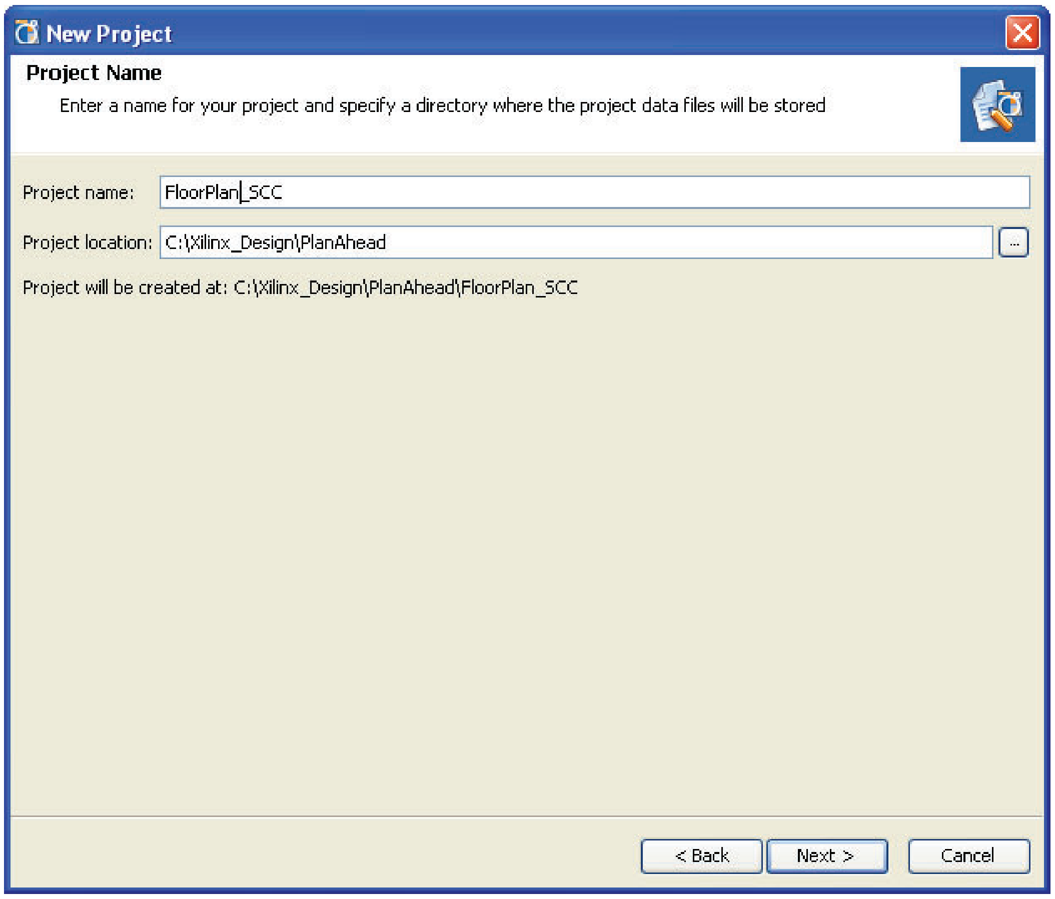
Start → **All Programs** → **Xilinx ISE Design Suite 12.4** → **PlanAhead** → **PlanAhead**

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 tool project (see Figure 3-1):

   File → New Project



New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored

Project name: FloorPlan_SCC

Project location: C:\Xilinx_Design\PlanAhead

Project will be created at: C:\Xilinx_Design\PlanAhead\FloorPlan_SCC

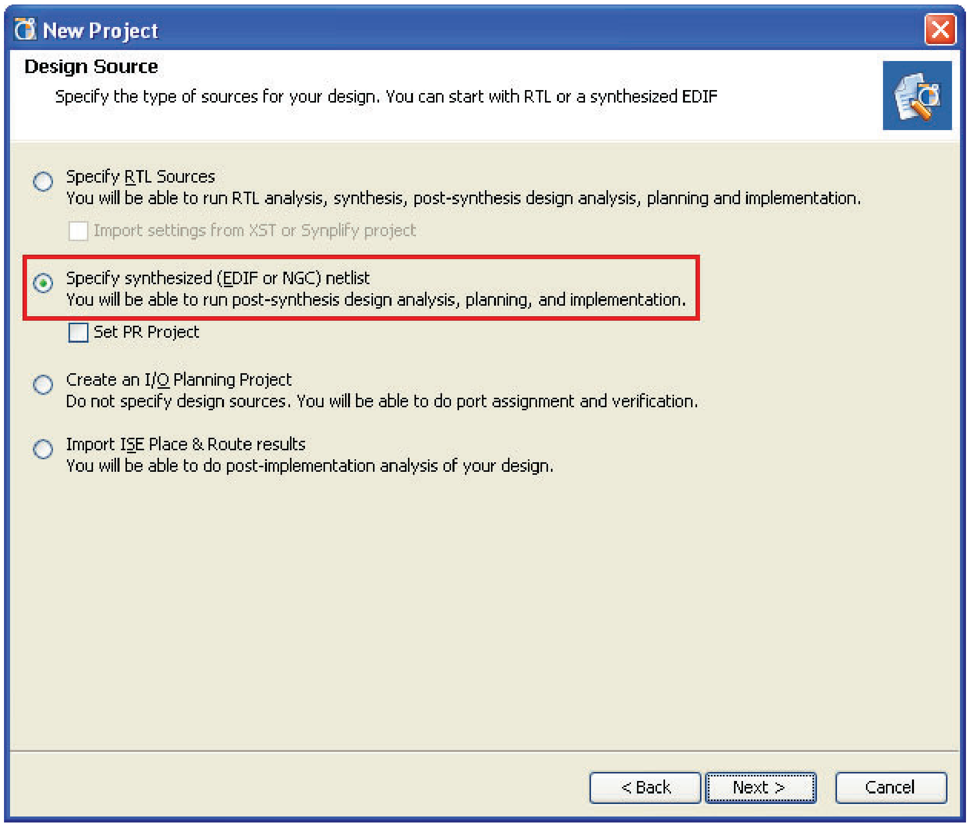
X1104_c3_01_102510

Figure 3-1: **New Project**

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

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

   Note: The Spartan-6 FPGA does not support partial reconfiguration, so do not select the **Set PR Project** option.



Design Source

Specify the type of sources for your design. You can start with RTL or a synthesized EDIF

( ) Specify RTL Sources
You will be able to run RTL analysis, synthesis, post-synthesis design analysis, planning and implementation.

☐ Import settings from XST or Synplify project

[✓ Specify synthesized (EDIF or NGC) netlist]
You will be able to run post-synthesis design analysis, planning, and implementation.

☐ Set PR Project

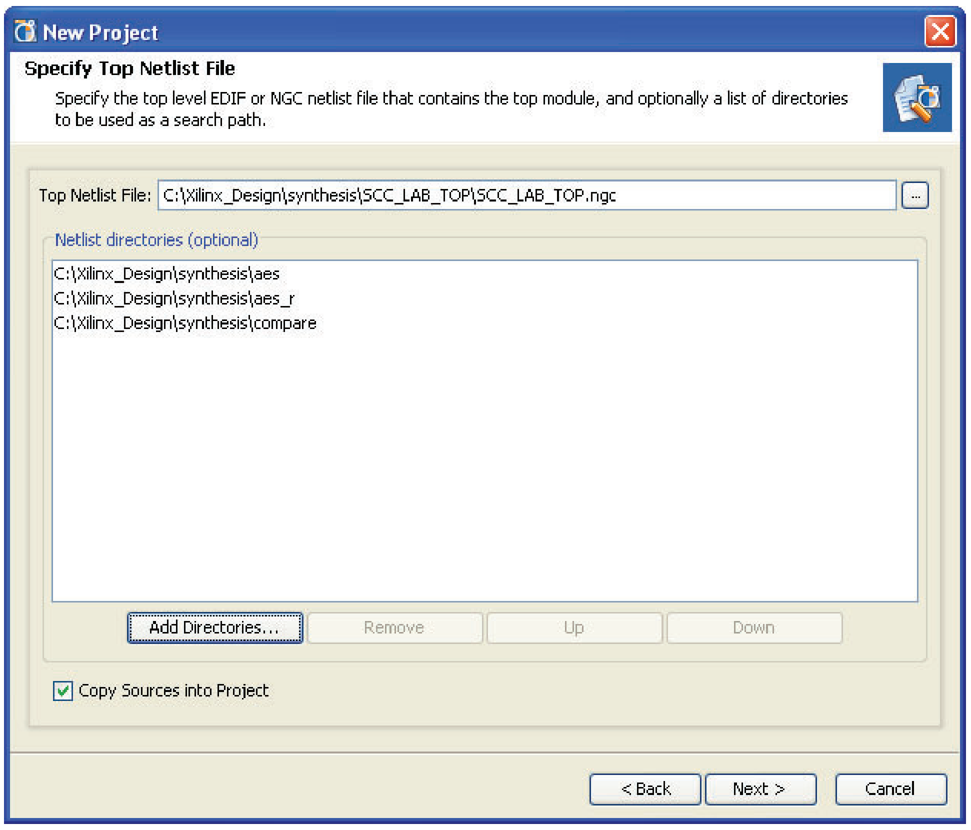
( ) Create an I/O Planning Project
Do not specify design sources. You will be able to do port assignment and verification.

( ) Import ISE Place & Route results
You will be able to do post-implementation analysis of your design.

X1104_c3_02_102510

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

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



Specify Top Netlist File

Specify the top level EDIF or NGC netlist file that contains the top module, and optionally a list of directories to be used as a search path.

Top Netlist File: C:\Xilinx_Design\synthesis\SCC_LAB_TOP\SCC_LAB_TOP.ngc

Netlist directories (optional)

```
C:\Xilinx_Design\synthesis\aes
C:\Xilinx_Design\synthesis\aes_r
C:\Xilinx_Design\synthesis\compare
```

[✓] Copy Sources into Project

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

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

   Netlist File:

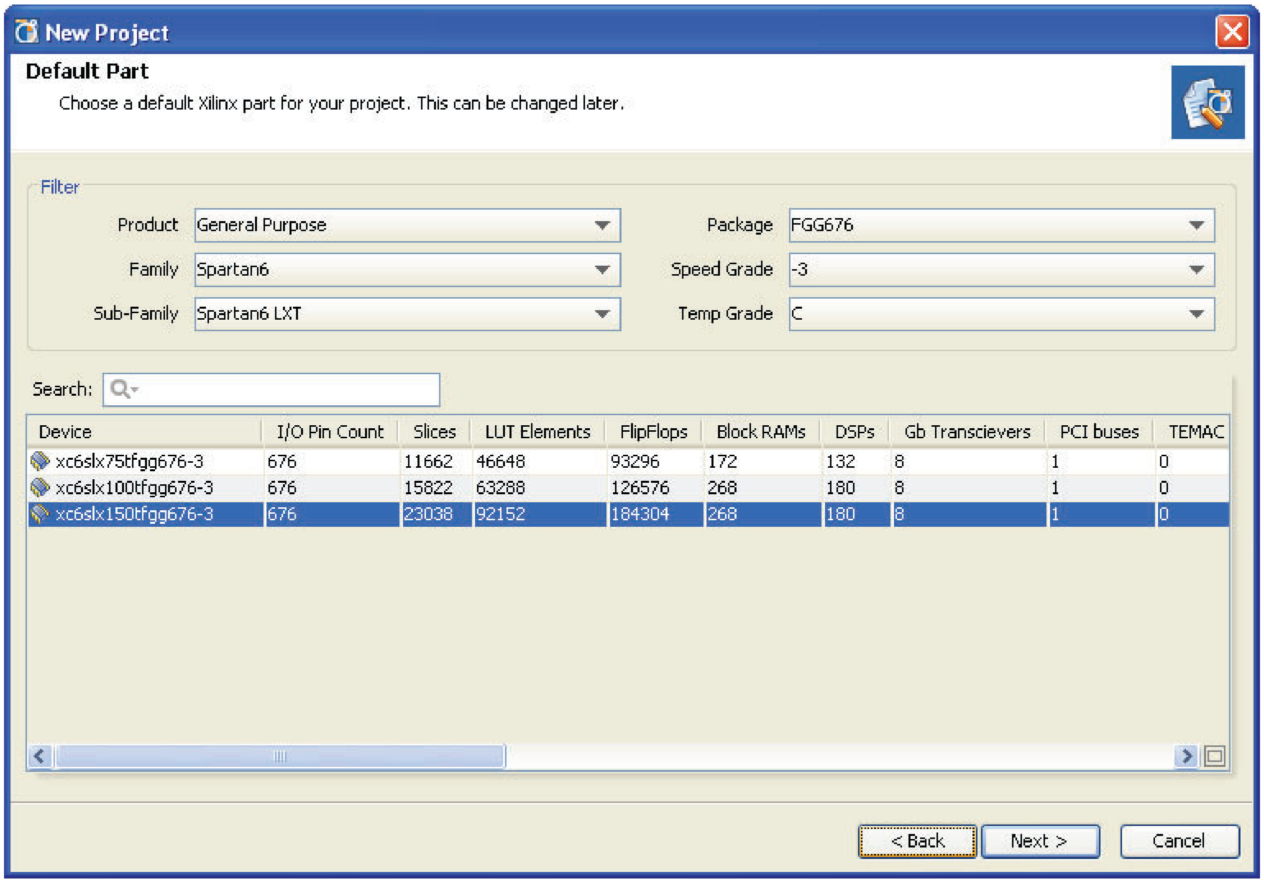
   ```
   ..\Xilinx_Design\synthesis\SCC_LAB_TOP\SCC_LAB_TOP.ngc
   ```

   Netlist Directories:

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

7. Click **Next**.
8. This lab creates the final user constraints file (UCF) from the beginning. Thus, no UCF files needs to be imported. Click **Next**.

9. Select the product, family, and sub-family for the default part (see Figure 3-4). For this lab, make the following selections:
   a. Product: **General Purpose**
   b. Family: **Spartan6**
   c. Sub-Family: **Spartan6 LXT**



Default Part

Choose a default Xilinx part for your project. This can be changed later.

Filter

Product: General Purpose
Family: Spartan6
Sub-Family: Spartan6 LXT
Package: FGG676
Speed Grade: -3
Temp Grade: C

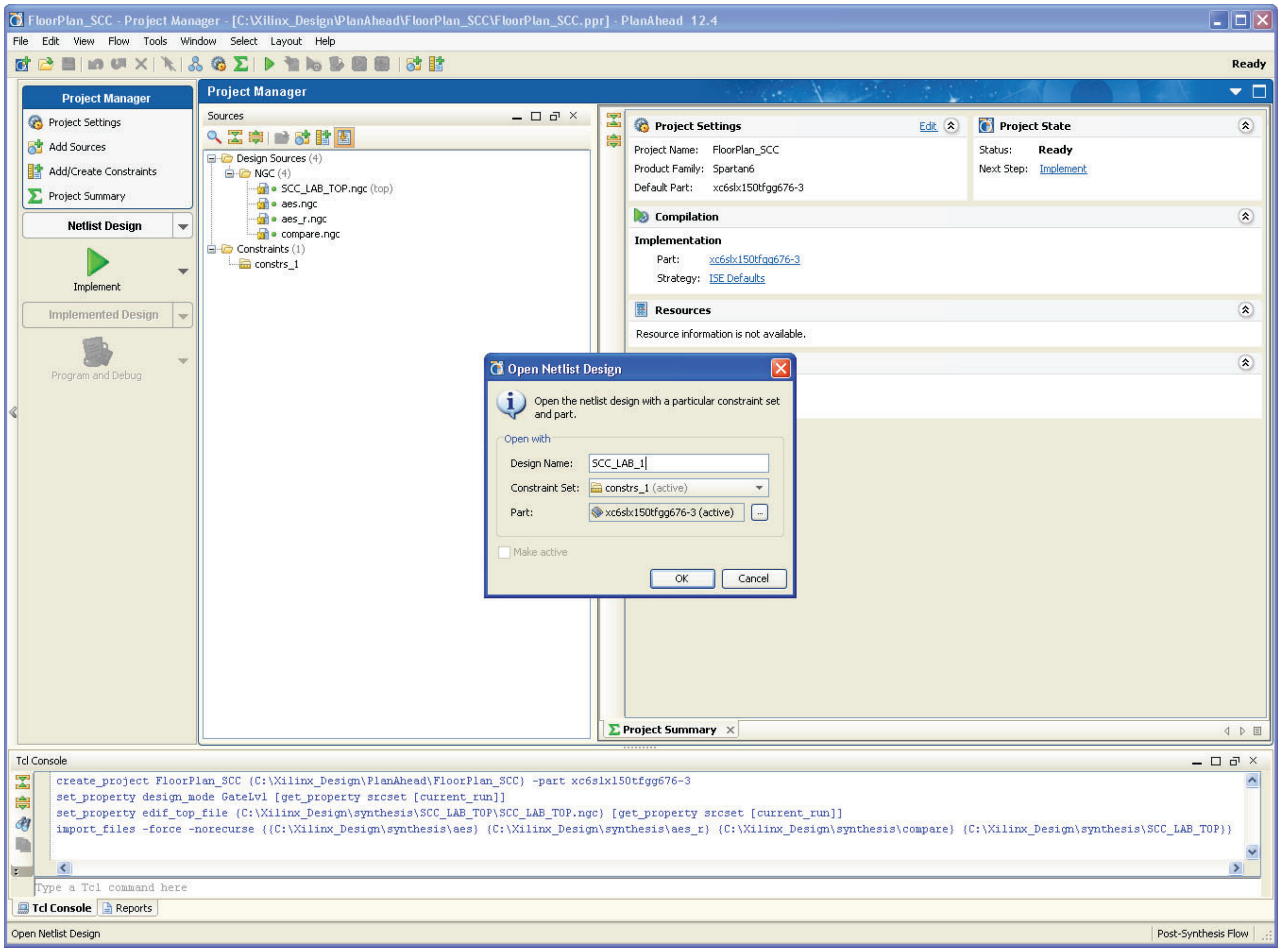
Device	I/O Pin Count	Slices	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Transcievers	PCI buses	TEMAC
xc6slx75tfgg676-3	676	11662	46648	93296	172	132	8	1	0
xc6slx100tfgg676-3	676	15822	63288	126576	268	180	8	1	0
xc6slx150tfgg676-3	676	23038	92152	184304	268	180	8	1	0

X1104_c3_04_102510

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

10. Select the device specifics:
    a. Package: **FGG676**
    b. Speed grade: **-3**
    c. Temperature grade: **C**
    d. Device: **xc6slx150tfgg676-3**
11. Click **Next** and **Finish**.

12. The PlanAhead tool project is created. Figure 3-5 shows the PlanAhead tool Project Manager window and the Open Netlist Manager window for the FloorPlan_SCC project.



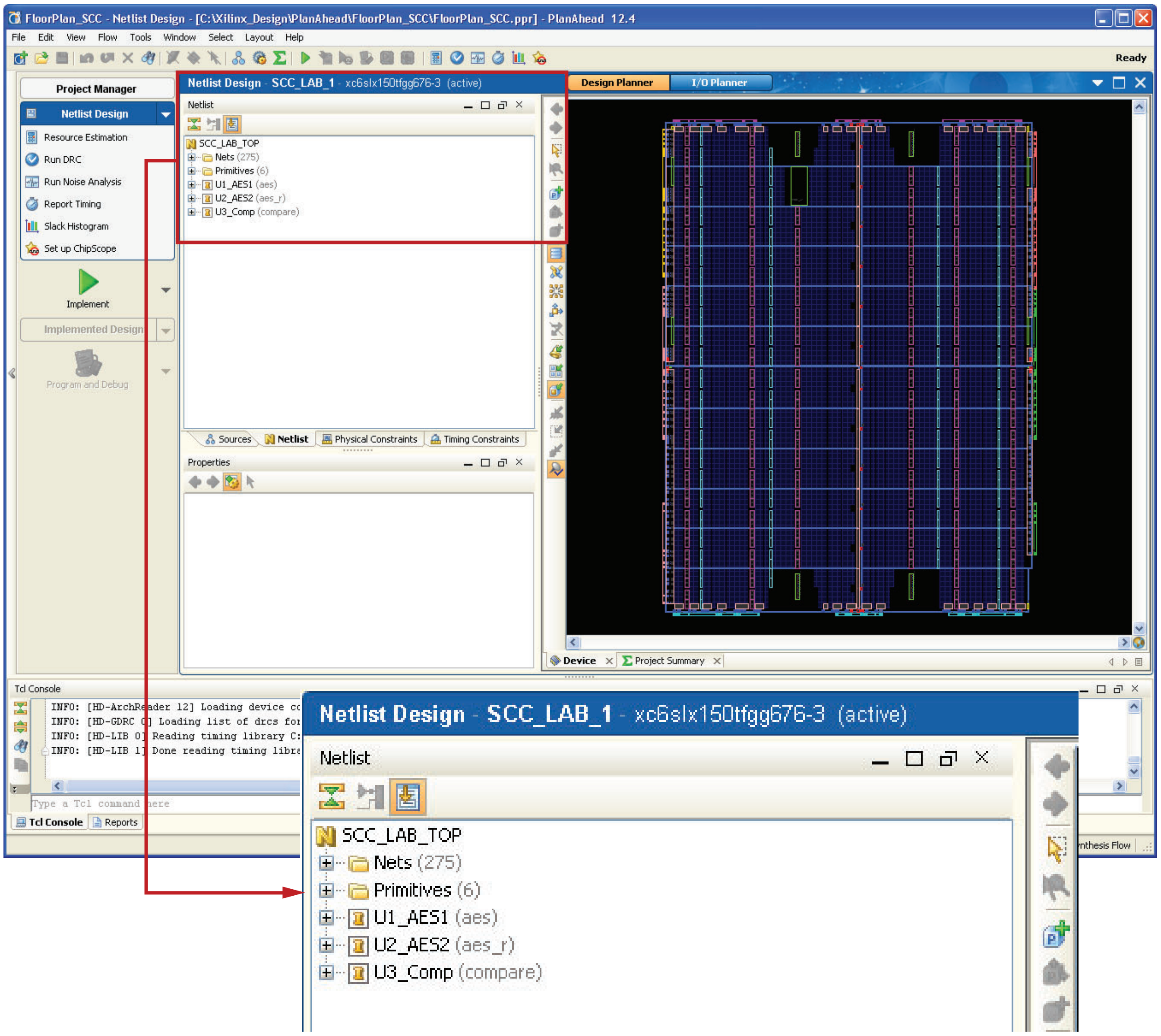
```
create_project FloorPlan_SCC {C:\Xilinx_Design\PlanAhead\FloorPlan_SCC} -part xc6slx150tfgg676-3
set_property design_mode GateLvl [get_property srcset [current_run]]
set_property edif_top_file {C:\Xilinx_Design\synthesis\SCC_LAB_TOP\SCC_LAB_TOP.ngc} [get_property srcset [current_run]]
import_files -force -norecurse {{C:\Xilinx_Design\synthesis\aes} {C:\Xilinx_Design\synthesis\aes_r} {C:\Xilinx_Design\synthesis\compare} {C:\Xilinx_Design\synthesis\SCC_LAB_TOP}}
```

X1104_c3_05_062411

Figure 3-5: **PlanAhead Tool Project Manger (Open Netlist Design)**

13. To open the netlist design in the PlanAhead tool, under the Project Manager pane on the left, select **Netlist Design** → **Open Netlist Design...**.
14. Set the design name to **SCC_LAB_1**. Click **OK**.

15. The floor plan of the FPGA device appears, as shown in Figure 3-6.



```
INFO: [HD-ArchReader 12] Loading device co
INFO: [HD-GDRC 0] Loading list of drcs for
INFO: [HD-LIB 0] Reading timing library C:
INFO: [HD-LIB 1] Done reading timing libra
Type a Tcl command here
```

Figure 3-6: **PlanAhead Design Planner View**

As stated in step 4, the Spartan-6 FPGA does not support partial reconfiguration. Therefore, the **File** → **Set PR Project** option is grayed out and not selectable.

Placing I/Os and DCMs with the PlanAhead Tool

When placing I/O buffers or pins, it is imperative to consider the physical location of I/Os in relation to the logic regions they interface with. While the clock input can be placed on any clock-capable pin because it does not have to be isolated, the led pin is part of the compare logic, so it does need to be physically placed within that region. Similarly, the reset and push_button pins are owned by the aes logic, so they need to be physically placed inside the aes region. All available resources should be included, even if the logic is not used, because excluding them also excludes using their respective routing resources. This includes clock components such as DCMs, PLLs, and BUFGs, even though such components are instantiated (logically owned) at the top level.

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 Design window and the Device Planner window.

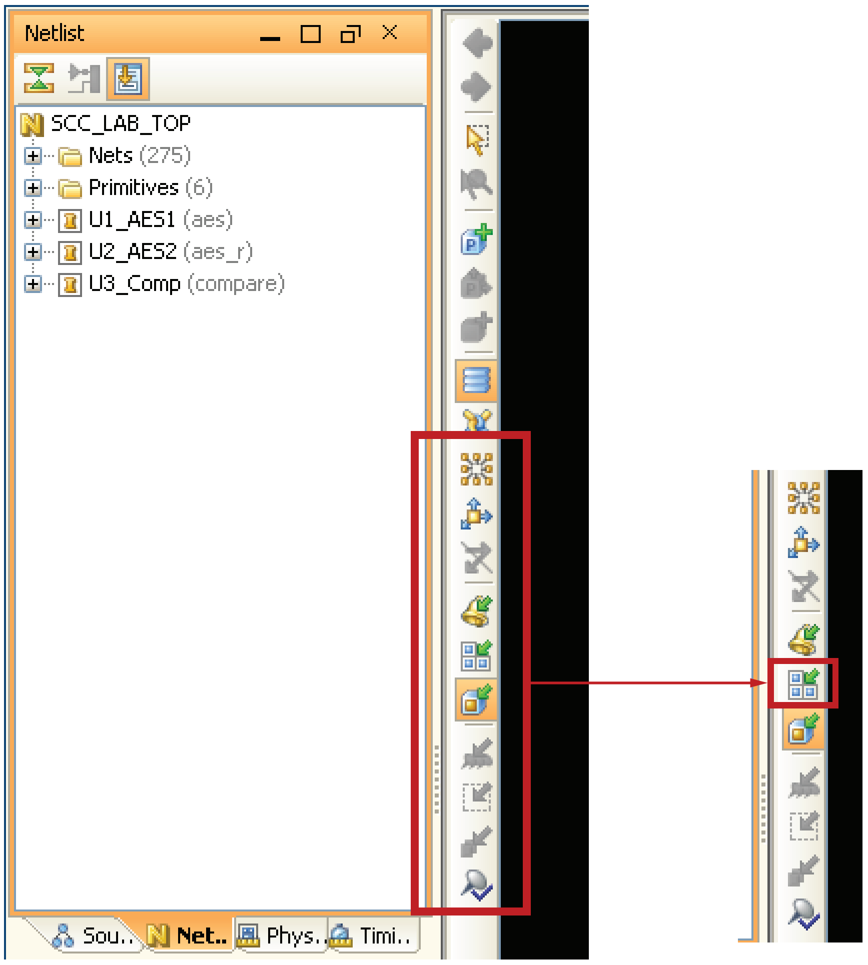
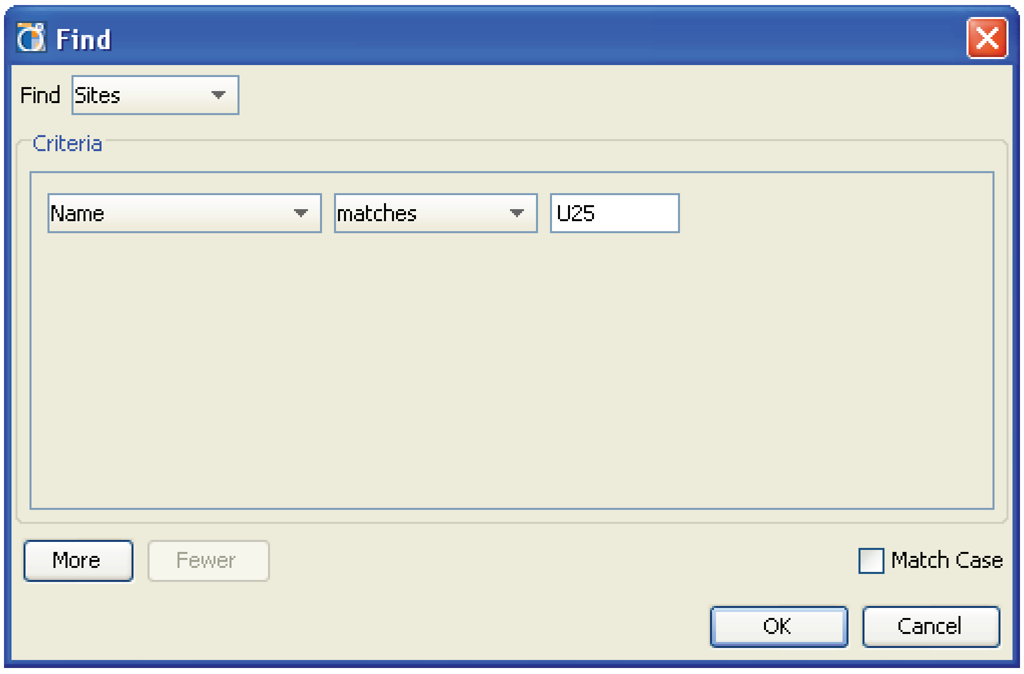


Figure 3-7: **Select Site Constraint Mode Symbol**

2. To specify the I/O pin for the clk, select **Edit** → **Find** (shortcut: **Ctrl-F**). A Find window appears, as shown in Figure 3-8.


Find

Find Sites

Criteria

Name | matches | U25

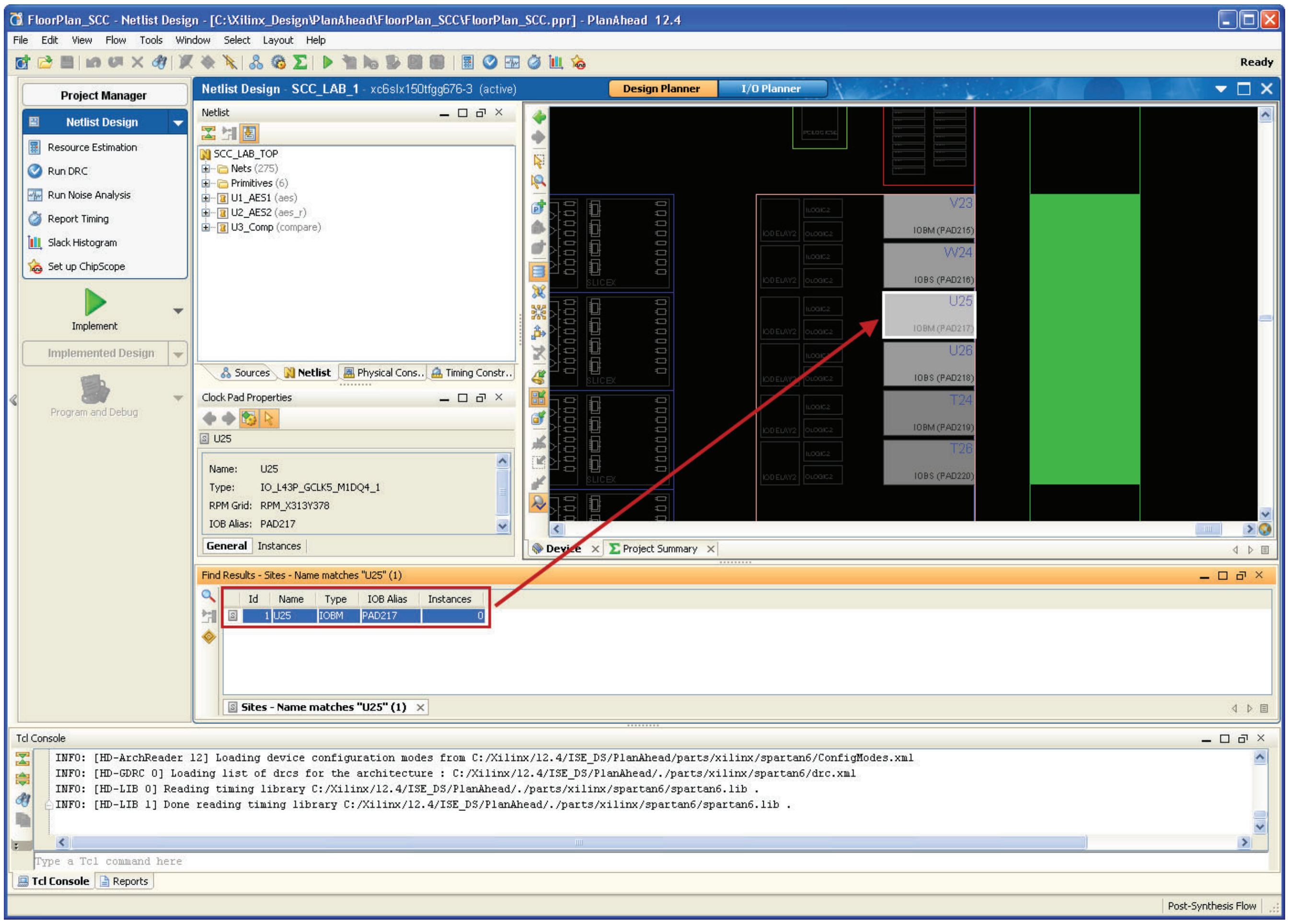
X1104_c3_08_102510

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 **U25** 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 find 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.



Name: U25
Type: IO_L43P_GCLK5_M1DQ4_1
RPM Grid: RPM_X313Y378
IOB Alias: PAD217

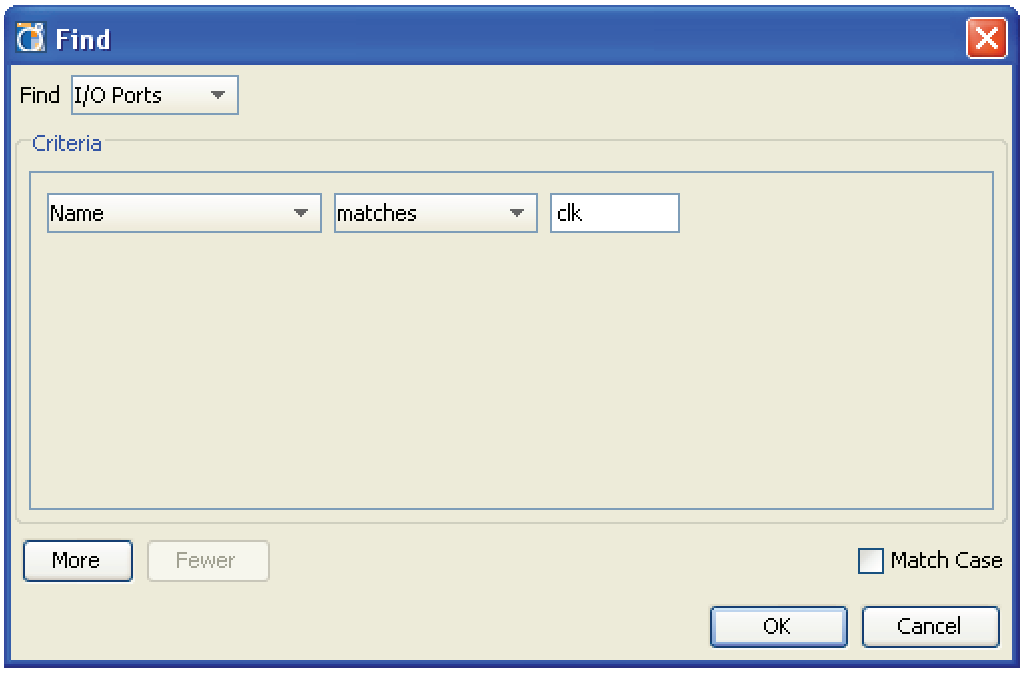
	Id	Name	Type	IOB Alias	Instances
	1	U25	IOBM	PAD217	0

```
INFO: [HD-ArchReader 12] Loading device configuration modes from C:/Xilinx/12.4/ISE_DS/PlanAhead/parts/xilinx/spartan6/ConfigModes.xml
INFO: [HD-GDRC 0] Loading list of drcs for the architecture : C:/Xilinx/12.4/ISE_DS/PlanAhead/./parts/xilinx/spartan6/drc.xml
INFO: [HD-LIB 0] Reading timing library C:/Xilinx/12.4/ISE_DS/PlanAhead/./parts/xilinx/spartan6/spartan6.lib .
INFO: [HD-LIB 1] Done reading timing library C:/Xilinx/12.4/ISE_DS/PlanAhead/./parts/xilinx/spartan6/spartan6.lib .
```

X1104_c3_09_062411

Figure 3-9: **Find Results**

7. To place clk on pin **U25**, select **Edit** → **Find** (shortcut: **Ctrl-F**). A Find window appears, as shown in Figure 3-10.



Find: I/O Ports

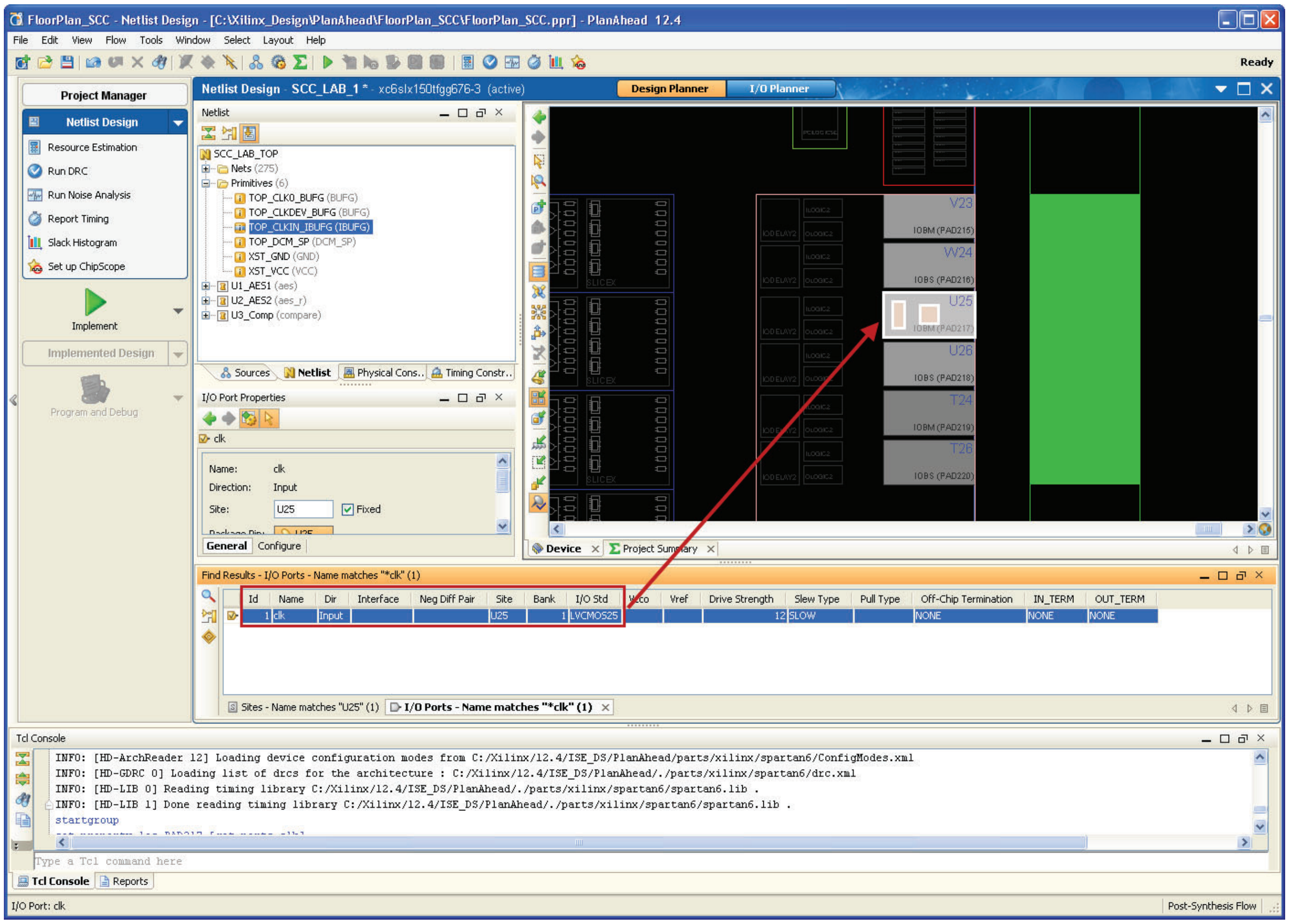
Criteria: Name matches clk

☐ Match Case

Figure 3-10: **Find**

8. Select **I/O 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.



Find Results - I/O Ports - Name matches "*clk" (1)

Id	Name	Dir	Interface	Neg Diff Pair	Site	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM	OUT_TERM
1	clk	Input			U25	1	LVCMOS25			12	SLOW		NONE	NONE	NONE

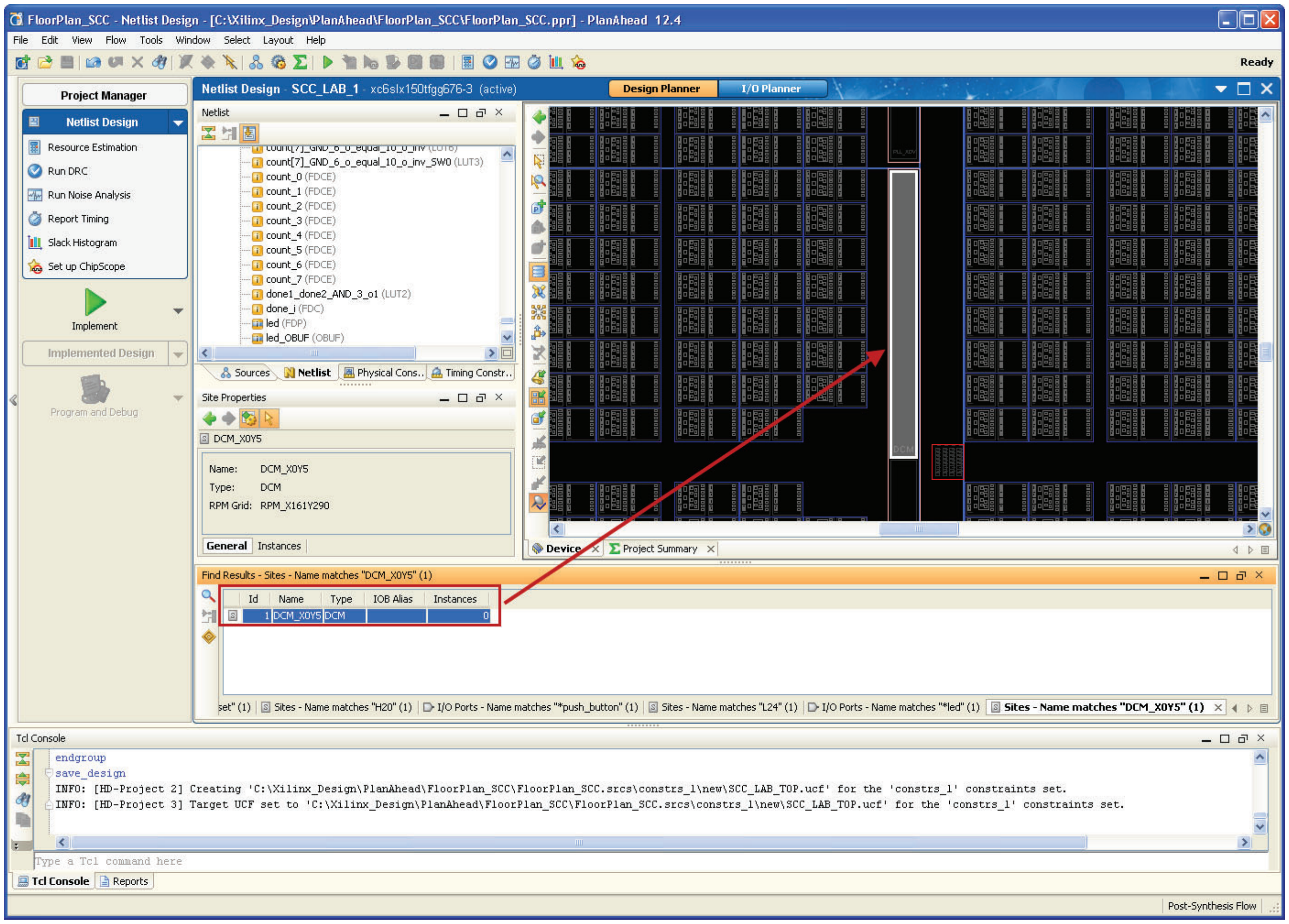
```
INFO: [HD-ArchReader 12] Loading device configuration modes from C:/Xilinx/12.4/ISE_DS/PlanAhead/parts/xilinx/spartan6/ConfigModes.xml
INFO: [HD-GDRC 0] Loading list of drcs for the architecture : C:/Xilinx/12.4/ISE_DS/PlanAhead/./parts/xilinx/spartan6/drc.xml
INFO: [HD-LIB 0] Reading timing library C:/Xilinx/12.4/ISE_DS/PlanAhead/./parts/xilinx/spartan6/spartan6.lib .
INFO: [HD-LIB 1] Done reading timing library C:/Xilinx/12.4/ISE_DS/PlanAhead/./parts/xilinx/spartan6/spartan6.lib .
startgroup
```

X1104_c3_11_062411

Figure 3-11: **Drag clk**

12. Repeat step 2 to step 11 to place reset at M19.
13. Repeat step 2 to step 11 to place push_button at H20.
14. Repeat step 2 to step 11 to place led at L24.

15. Select **Edit** → **Find** (shortcut: **Ctrl-F**).
16. Select **Sites** from the Find pull-down menu.
17. Select **Name** and **matches** from the two pull-down menus under Criteria.
18. Type **`DCM_X0Y5`** 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. Select the result and press **F9** to zoom to the current selection, as shown in Figure 3-12.



Name: DCM_X0Y5
Type: DCM
RPM Grid: RPM_X161Y290

Find Results - Sites - Name matches "DCM_X0Y5" (1)

Id	Name	Type	IOB Alias	Instances
1	DCM_X0Y5	DCM		0

set" (1) | Sites - Name matches "H20" (1) | I/O Ports - Name matches "*push_button" (1) | Sites - Name matches "L24" (1) | I/O Ports - Name matches "*led" (1) | Sites - Name matches "DCM_X0Y5" (1)

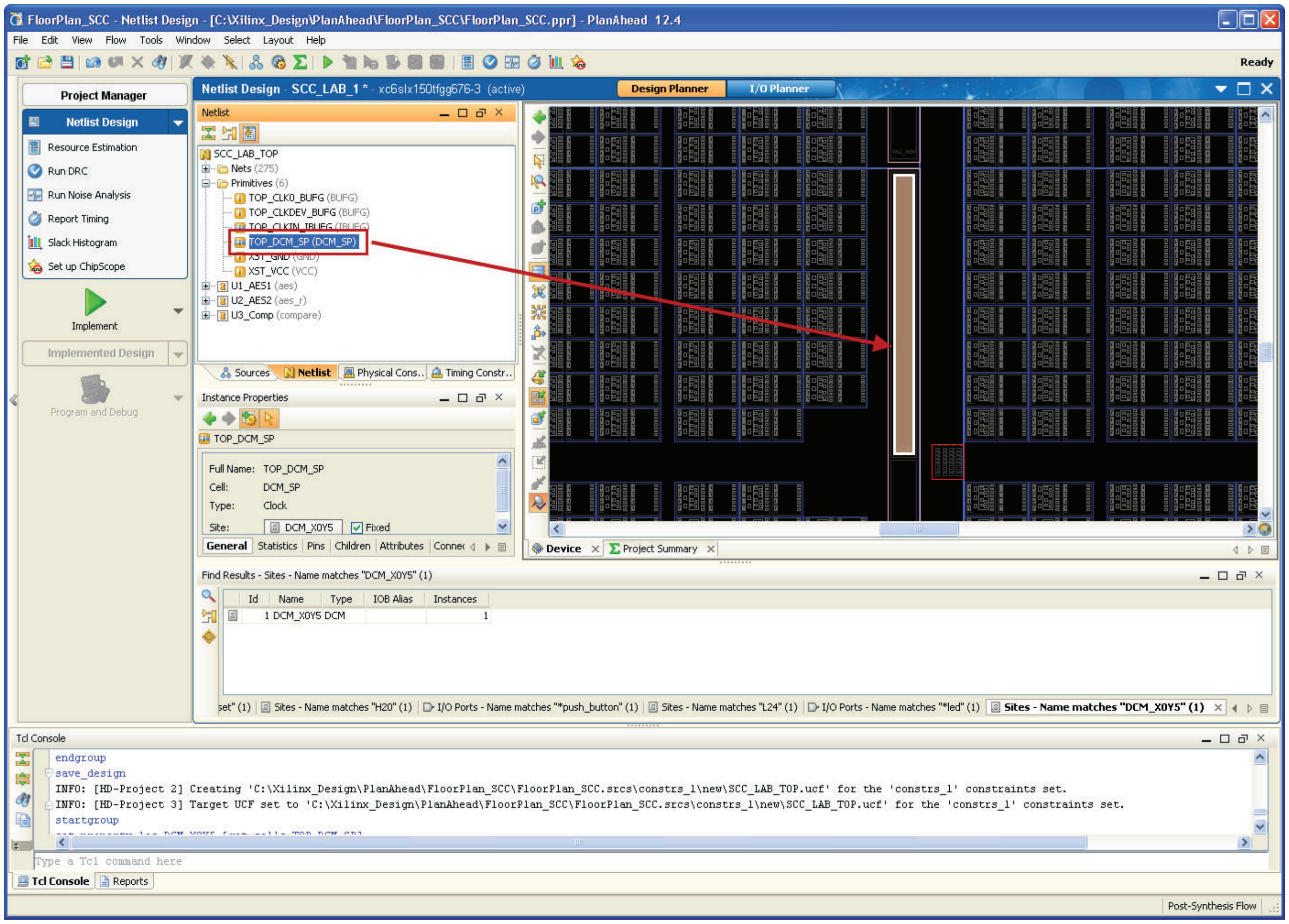
```
endgroup
save_design
INFO: [HD-Project 2] Creating 'C:\Xilinx_Design\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.srcs\constrs_1\new\SCC_LAB_TOP.ucf' for the 'constrs_1' constraints set.
INFO: [HD-Project 3] Target UCF set to 'C:\Xilinx_Design\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.srcs\constrs_1\new\SCC_LAB_TOP.ucf' for the 'constrs_1' constraints set.
```

Type a Tcl command here

X1104_c3_12_062411

Figure 3-12: **Search Results for DCM_X0Y5**

19. In the Netlist window, expand Primitives, and then drag TOP_DCM_SP to the DCM_X0Y5 box, as shown in Figure 3-13.



Full Name: TOP_DCM_SP
Cell: DCM_SP
Type: Clock
Site: DCM_X0Y5 [✓ Fixed]

Find Results - Sites - Name matches "DCM_X0Y5" (1)

Id	Name	Type	IOB Alias	Instances
1	DCM_X0Y5	DCM		1

set" (1) | Sites - Name matches "H20" (1) | I/O Ports - Name matches "*push_button" (1) | Sites - Name matches "L24" (1) | I/O Ports - Name matches "*led" (1) | **Sites - Name matches "DCM_X0Y5" (1)**

```
endgroup
save_design
INFO: [HD-Project 2] Creating 'C:\Xilinx_Design\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.srcs\constrs_1\new\SCC_LAB_TOP.ucf' for the 'constrs_1' constraints set.
INFO: [HD-Project 3] Target UCF set to 'C:\Xilinx_Design\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.srcs\constrs_1\new\SCC_LAB_TOP.ucf' for the 'constrs_1' constraints set.
startgroup
```

Type a Tcl command here

X1104_c3_13_062411

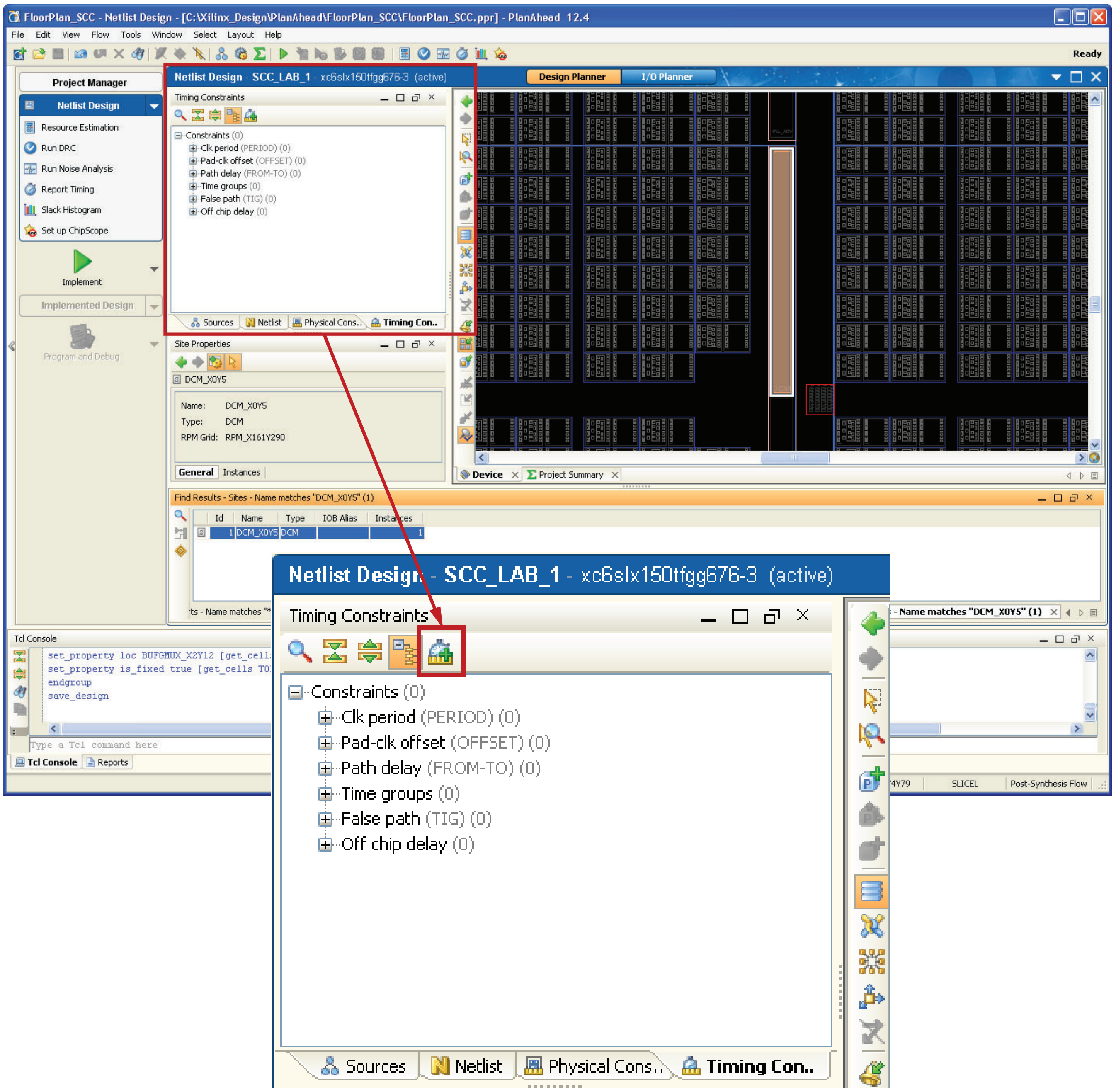
Figure 3-13: **Placing DCM_X0Y5**

20. Repeat step 15 to step 19 to place TOP_CLK0_BUFG at BUFGMUX_X3Y16.
21. Repeat step 15 to step 19 to place TOP_CLKDEV_BUFG at BUFGMUX_X2Y12.

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.

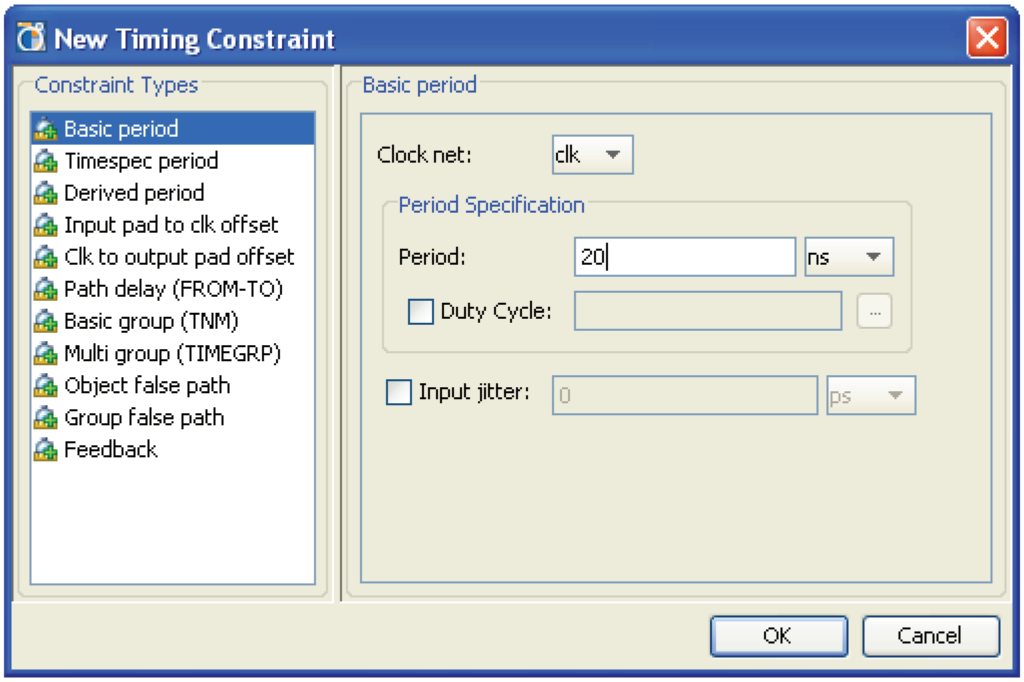
1. Select the **Timing Constraints** tab in the Netlist Design window, and click the **Create New Constraint** button located at the top of the netlist/constraints window (see Figure 3-14).


FloorPlan_SCC - Netlist Design - [C:\Xilinx_Design\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.ppr] - PlanAhead 12.4
Netlist Design - SCC_LAB_1 - xc6slx150tfgg676-3 (active)
Timing Constraints
Constraints (0)
Clk period (PERIOD) (0)
Pad-clk offset (OFFSET) (0)
Path delay (FROM-TO) (0)
Time groups (0)
False path (TIG) (0)
Off chip delay (0)
Sources
Netlist
Physical Cons..
Timing Con..
X1104_c3_14_062411

```
set_property loc BUFGMUX_X2Y12 [get_cell
set_property is_fixed true [get_cells TO
endgroup
save_design
```

Figure 3-14: **Create New Constraint**

2. Add a **Basic period** constraint of 20 ns to the design by entering the value, as shown in Figure 3-15.

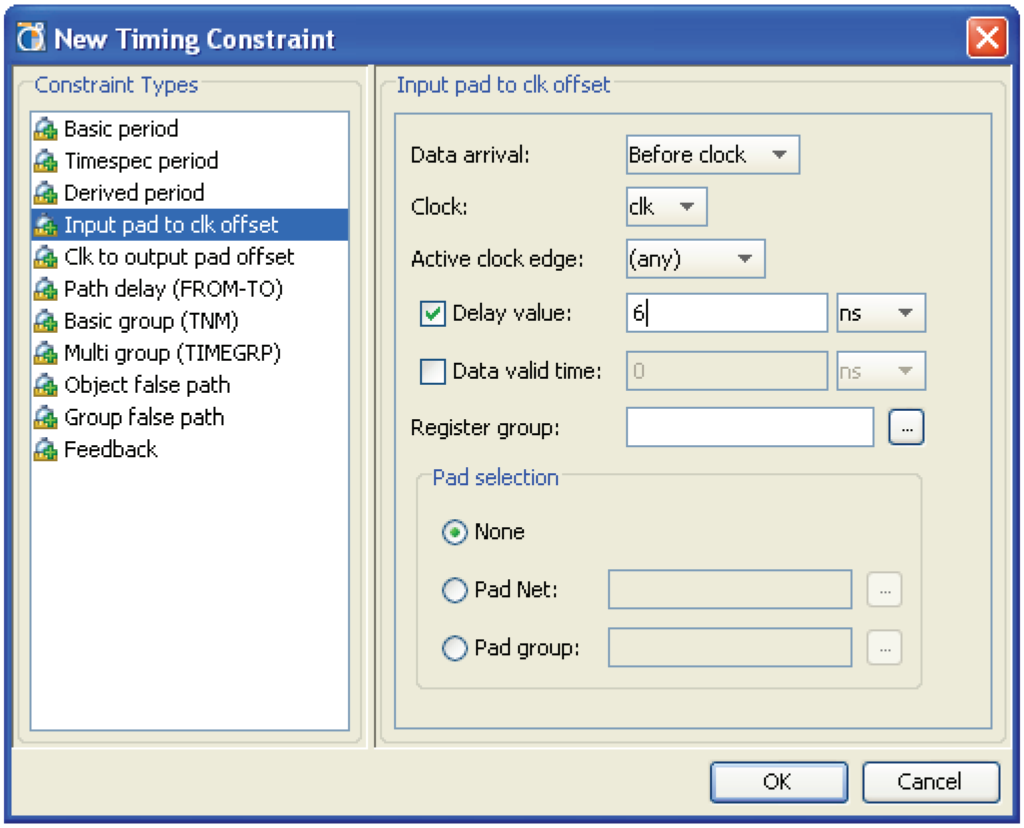


Clock net: clk
Period: 20 ns
Duty Cycle:
Input jitter: 0 ps

Figure 3-15: **New Timing Constraint - Basic Period**

3. Click **OK**.

4. Create a new timing constraint for **Input pad to clk offset**. For Data arrival, select **Before clock**, 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 3-16.

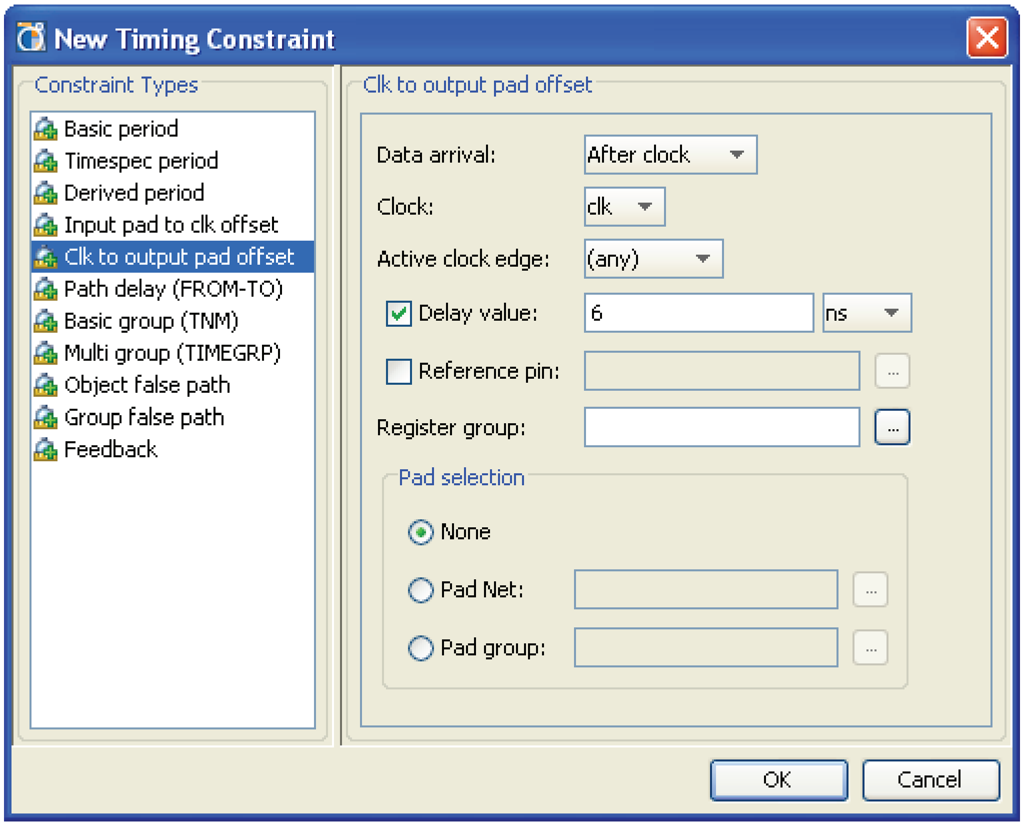


Data arrival: Before clock
Clock: clk
Active clock edge: (any)
[✓] Delay value: 6 ns
[ ] Data valid time: 0 ns
Register group:
Pad selection: [✓ None] Pad Net: Pad group:

Figure 3-16: **New Timing Constraint - Input Pad to Clk Offset**

5. Click **OK**.

6. Create a new timing constraint for **Clk to output pad** offset. For Data arrival, select **After clock**, 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 3-17.



Data arrival: After clock
Clock: clk
Active clock edge: (any)
[✓] Delay value: 6 ns
[ ] Reference pin:
Register group:
Pad selection: [✓ None] Pad Net: Pad group:

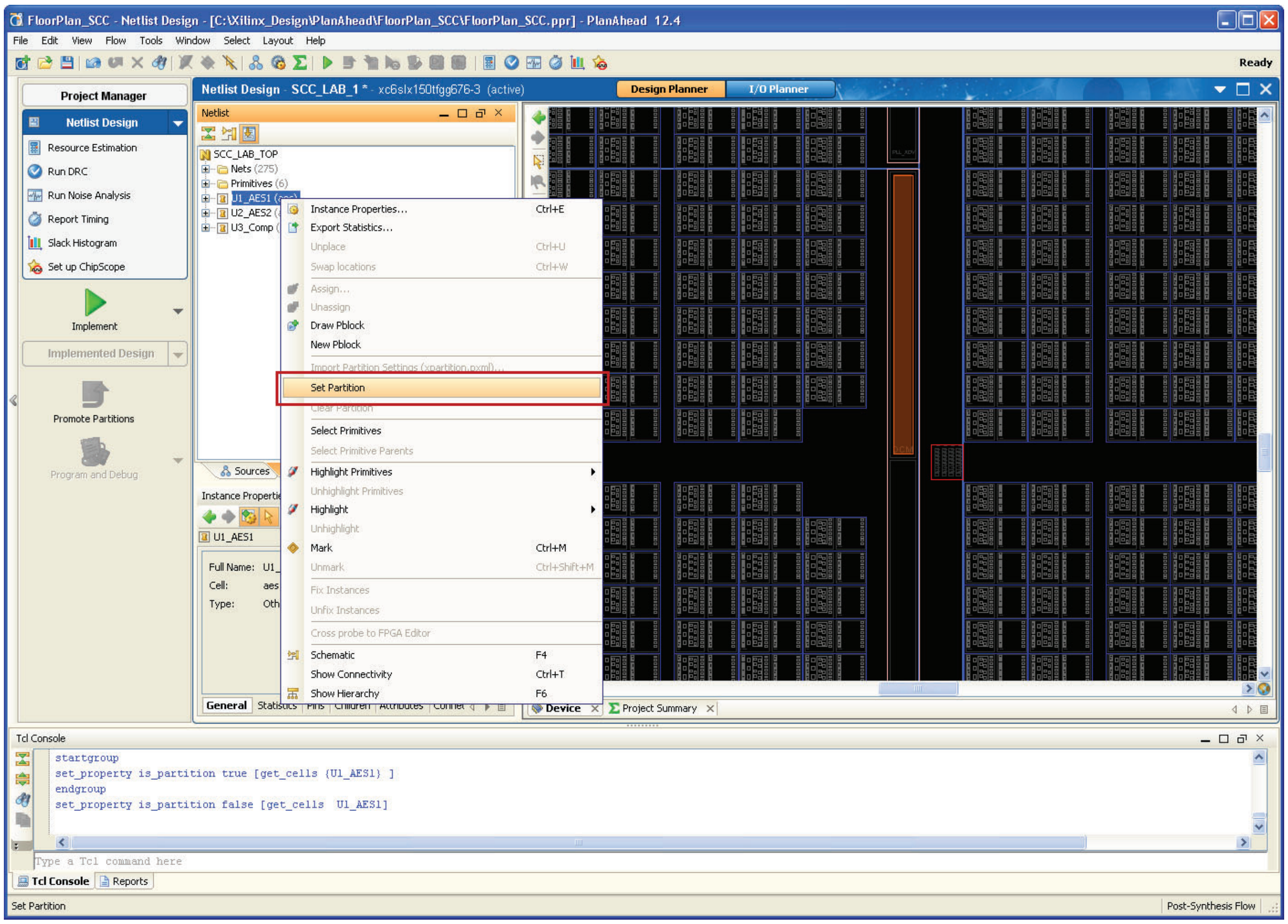
Figure 3-17: **New Timing Constraint - Clk to Output Pad Offset**

7. Click **OK**.

Defining ISO Partitions with the PlanAhead Tool

Each partition must be turned into an ISO partition:

1. Select and right-click **U1_AES (aes)** in the Netlist tab and select **Set Partition** from the pull-down menu (see Figure 3-18).



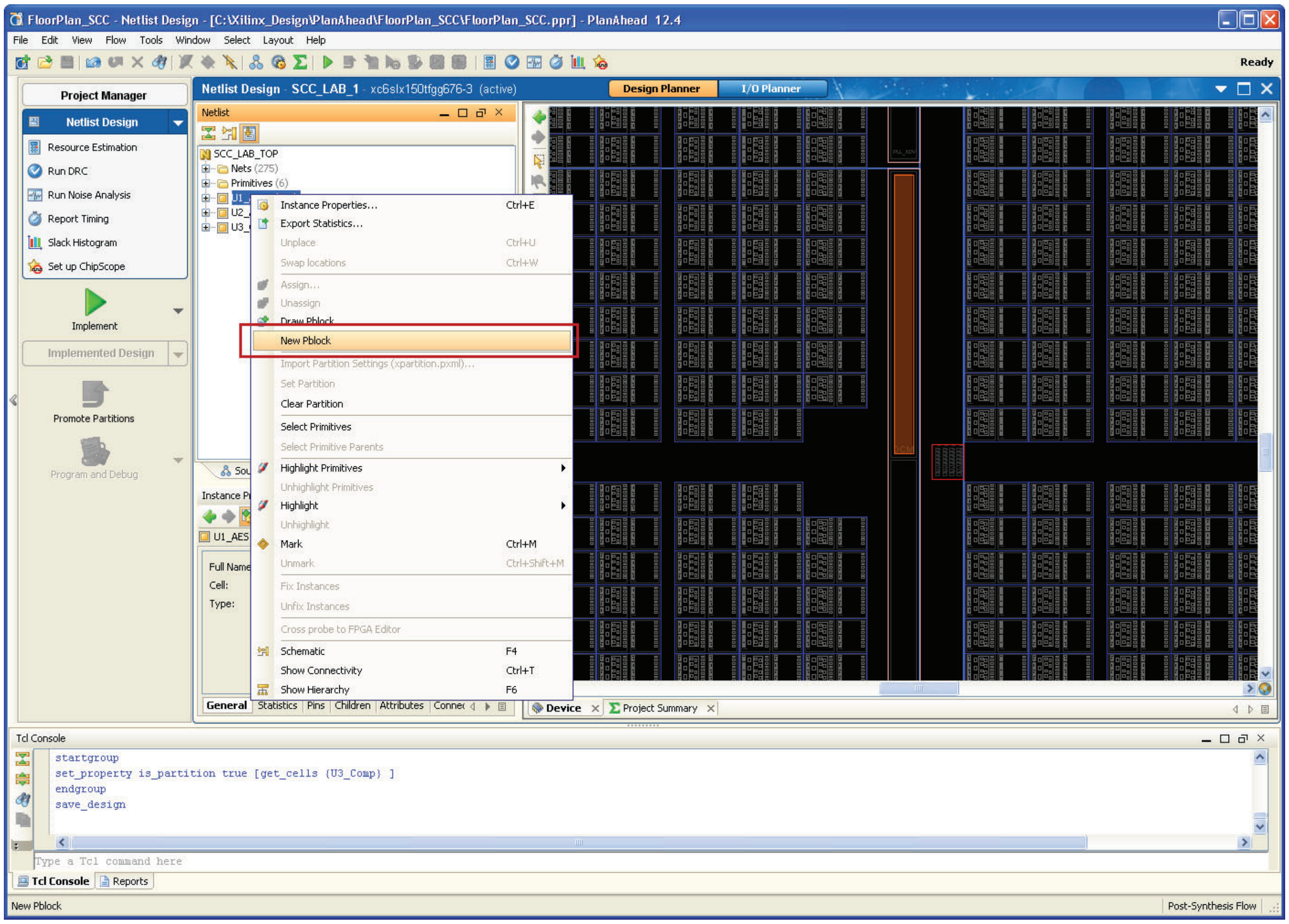
```
startgroup
set_property is_partition true [get_cells {U1_AES1} ]
endgroup
set_property is_partition false [get_cells  U1_AES1]
```

X1104_c3_18_062411

Figure 3-18: **Set Partition**

2. Repeat step 1 for the **U2_AES2 (aes_r)** partition.
3. Repeat step 1 for the **U3_Comp (compare)** partition.

4. A new Pblock must be created for each block. In the Netlist tab, select the **U1_AES1 (aes)** partition, and right-click to select **New Pblock** from the menu (see Figure 3-19). Keep the default name, pblock_U1_AES1, and click **OK**.



```
startgroup
set_property is_partition true [get_cells {U3_Comp} ]
endgroup
save_design
```

X1104_c3_19_062411

Figure 3-19: **New Pblock**

5. Repeat step 4 for the **U2_AES2 (aes_r)** partition while keeping the default name, pblock_U2_AES2.

6. Repeat step 4 for the **U3_Comp (compare)** partition while keeping the default name, pblock_U3_Comp.

Defining Attributes for Each ISO Partition with the PlanAhead Tool

IDF 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 Netlist tab, select the **U1_AES1 (aes)** partition, right-click the selection, and click **Instance Properties** in the resulting menu.
2. Click the **Attributes** tab of the Instance Properties window, and click the **Add Pre-defined Attributes** button (a green "+" symbol).
3. Click the **SCC_ ISOLATED** general attribute to select it, and click **OK** (see Figure 3-20).

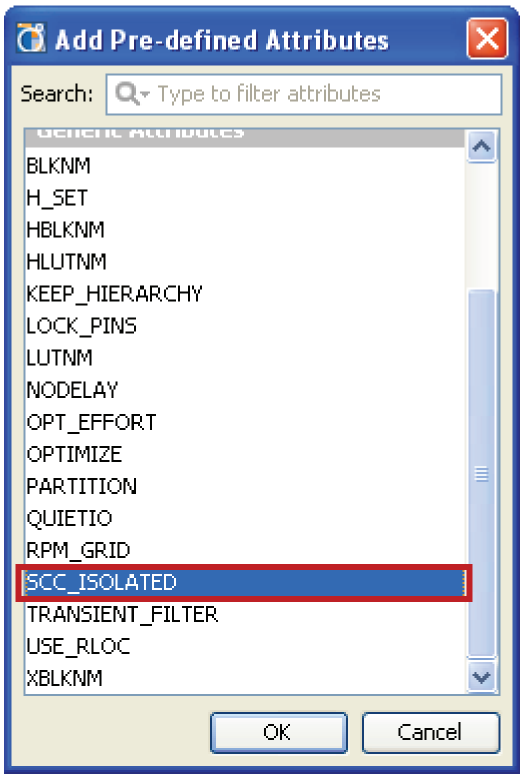
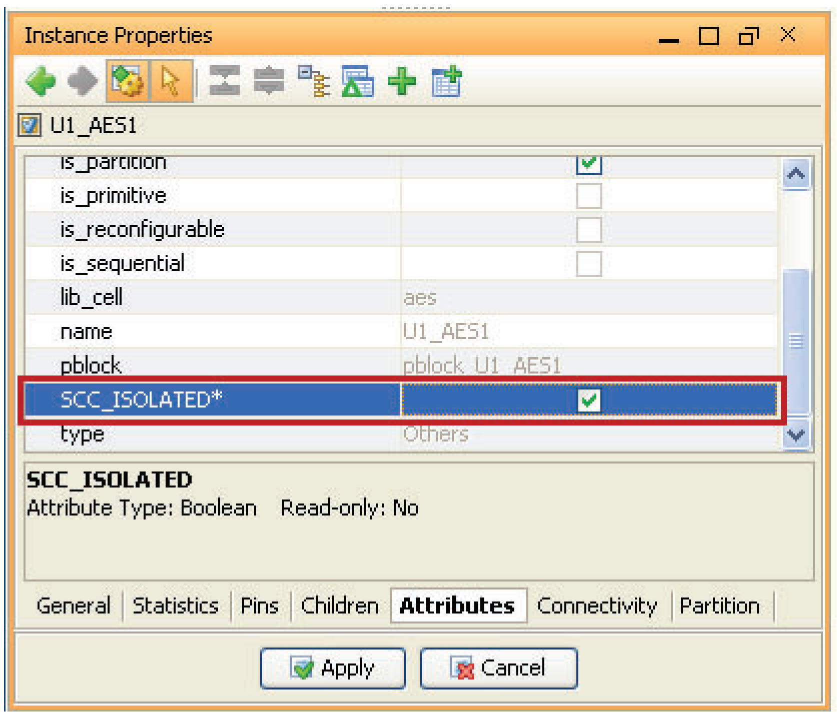


Figure 3-20: **Add Pre-Defined Attribute**

4. Select the checkbox for **SCC_ ISOLATED** in the general attributes list in the Instance Properties window, and click **Apply** (see Figure 3-21).



is_partition		✓
is_primitive		☐
is_reconfigurable		☐
is_sequential		☐
lib_cell	aes	
name	U1_AES1	
pblock	pblock_U1_AES1	
SCC_ISOLATED*		✓
type	Others	

SCC_ISOLATED
Attribute Type: Boolean   Read-only: No

X1104_c3_21_102510

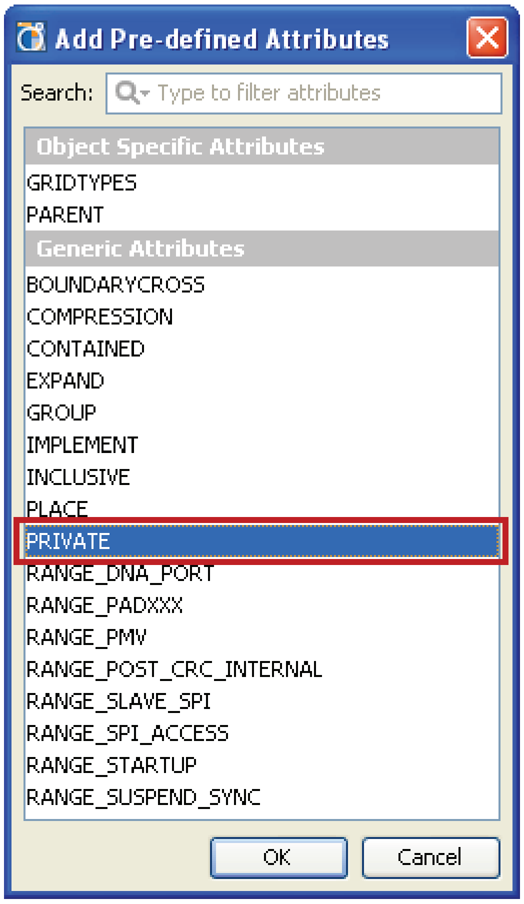
Figure 3-21: **Instance Properties**

5. Repeat step 1 to step 4 for the **U2_AES2 (aes_r)** partition.
6. Repeat step 1 to step 4 for the **U3_Comp (compare)** partition.

The PRIVATE attribute needs to be set for each pblock in the PlanAhead tool GUI as follows:

7. Under the Physical Constraints tab, select the **pblock_U1_AES** pblock, right-click the selection, and click **Pblock Properties** in the resulting menu.
8. Click the Attributes tab of the Pblock Properties window, and click the **Add Pre-defined Attributes** button (a green "+" symbol).

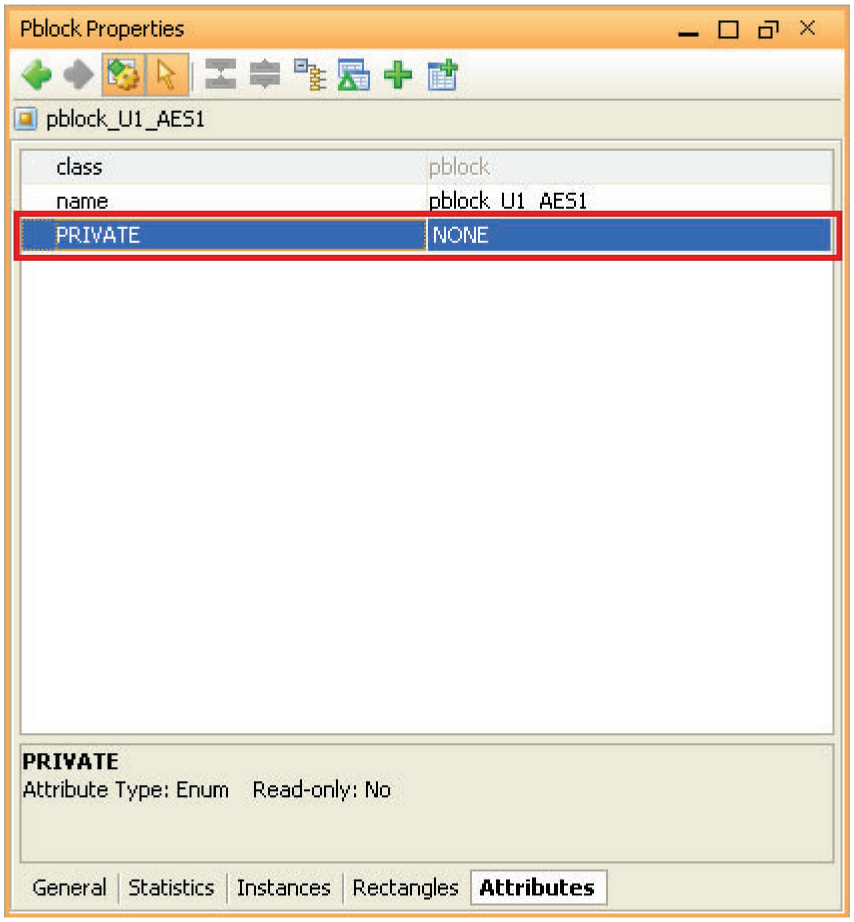
9. Click the **PRIVATE** attribute to select it, and click **OK** (see Figure 3-22).



X1104_c3_22_102510

Figure 3-22: **Adding PRIVATE Attribute**

10. Select the attribute **PRIVATE** in the general attributes list in the Pblock Properties window, and select **NONE** (see Figure 3-23).



class	pblock
name	pblock_U1_AES1
PRIVATE	NONE

PRIVATE
Attribute Type: Enum   Read-only: No

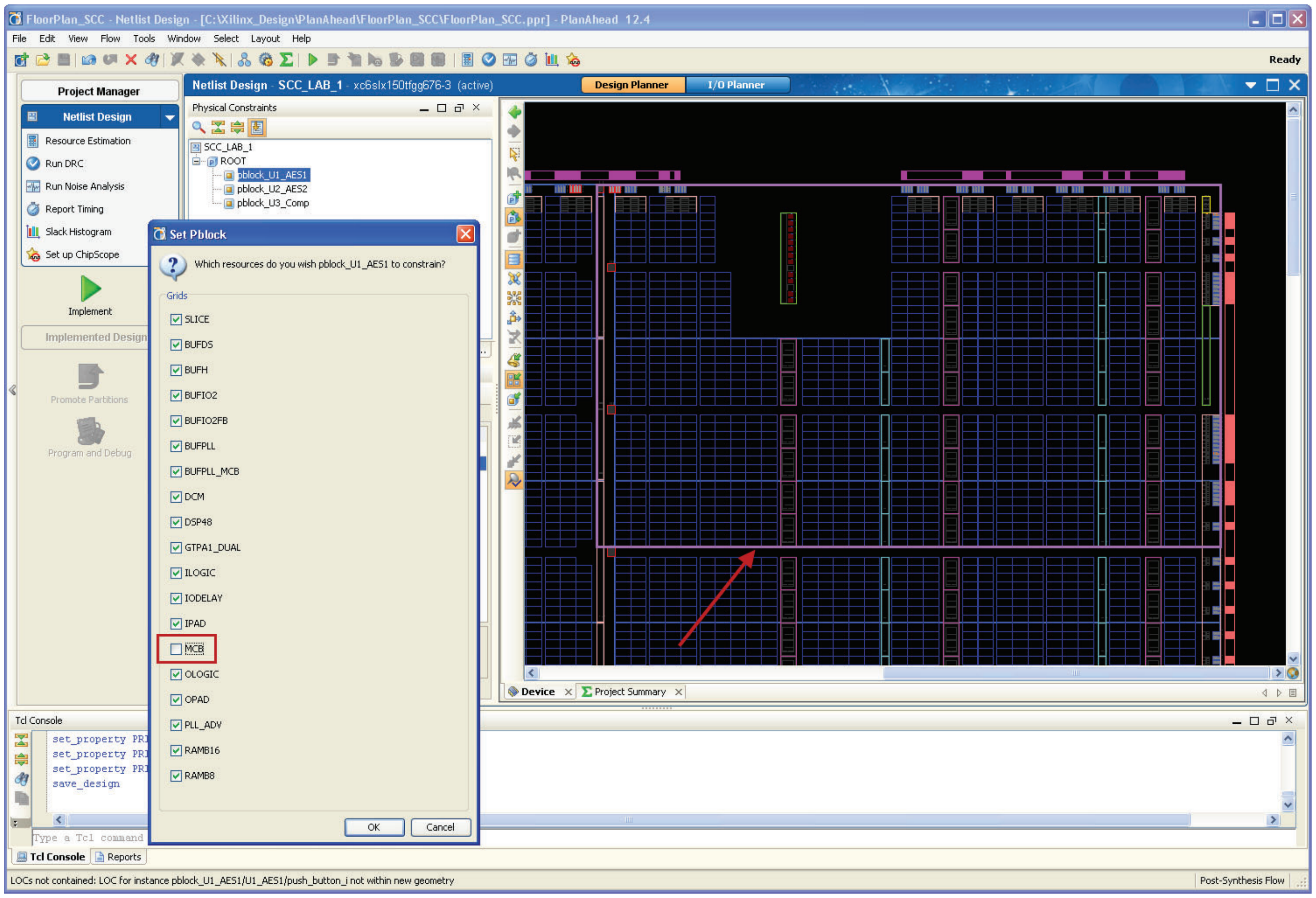
Figure 3-23: **Setting PRIVATE Attribute to NONE**

11. Click **Apply**.
12. Repeat step 7 to step 11 for pblock_U2_AES2, and set the attribute **PRIVATE** to **NONE**.
13. Repeat step 7 to step 11 for pblock_U3_Comp, and set the attribute **PRIVATE** to **NONE**.

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

Area groups constrain FPGA resources. Their size can be changed by the user with the PlanAhead tool. For the purpose of this exercise, the dimensions for the area groups are made known and are presented. However, a general method of configuring an area group for an arbitrary ISO region is demonstrated in this section:

1. Under the Physical Constraints tab, select the block **pblock_U1_AES1 (aes)**.
2. Right-click **pblock_U1_AES1 (aes)** and select **Set Pblock Size** from the pull-down menu.
3. Draw a rectangle in the upper-right area of the Design Planner window, as shown in Figure 3-24. (The rectangle does not have to be 100% accurate—it will be resized shortly.)



Set Pblock

Which resources do you wish pblock_U1_AES1 to constrain?

Grids

- [✓] SLICE
- [✓] BUFDS
- [✓] BUFH
- [✓] BUFIO2
- [✓] BUFIO2FB
- [✓] BUFPLL
- [✓] BUFPLL_MCB
- [✓] DCM
- [✓] DSP48
- [✓] GTPA1_DUAL
- [✓] ILOGIC
- [✓] IODELAY
- [✓] IPAD
- [ ] MCB
- [✓] OLOGIC
- [✓] OPAD
- [✓] PLL_ADV
- [✓] RAMB16
- [✓] RAMB8

OK Cancel

```
set_property PRI
set_property PRI
set_property PRI
save_design
```

Type a Tcl command

LOCs not contained: LOC for instance pblock_U1_AES1/U1_AES1/push_button_i not within new geometry

X1104_c3_24_062411

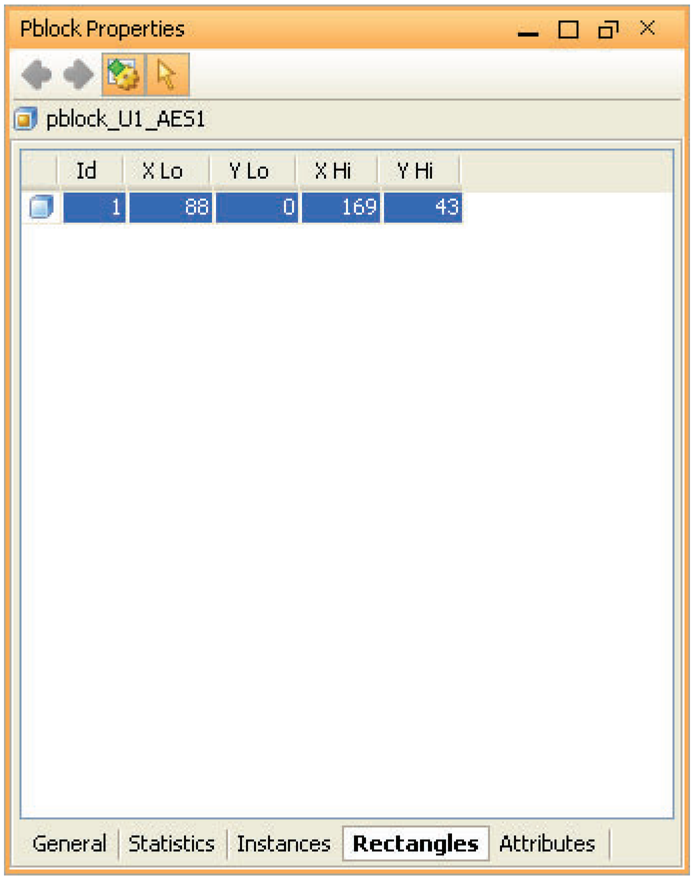
Figure 3-24: **pblock_U1_AES1 Pblock Layout (Set Pblock)**

4. A dialog box (shown in Figure 3-24) appears 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 used in the design, it is important to select them to take advantage of their routing resources. Failure to do so, while not an error, might produce designs that are unnecessarily

difficult to route. Naturally, all used components must be included. The only exception is the Memory Controller Block (MCB) box, which should not be checked for this lab.

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.

5. Click **OK**.
6. In the Choose LOC mode dialog box, select **Leave all location constraints in their current position**.
7. Click **OK**.
8. Ensure that **pblock_U1_AES1 (aes)** is selected in the Physical Constraints pane.
9. Select the **Rectangles** tab in the Pblock Properties window (see Figure 3-25).



Id	X Lo	Y Lo	X Hi	Y Hi
1	88	0	169	43

X1104_c3_25_102510

Figure 3-25: **Pblock Properties**

10. After the preliminary U1_AES1 (aes) block is drawn, adjust the rectangle graphically as necessary to match these coordinates:
    - X Lo = 88
    - Y Lo = 0
    - X Hi = 169
    - Y Hi = 43

    The final Pblock should look like Figure 3-26.

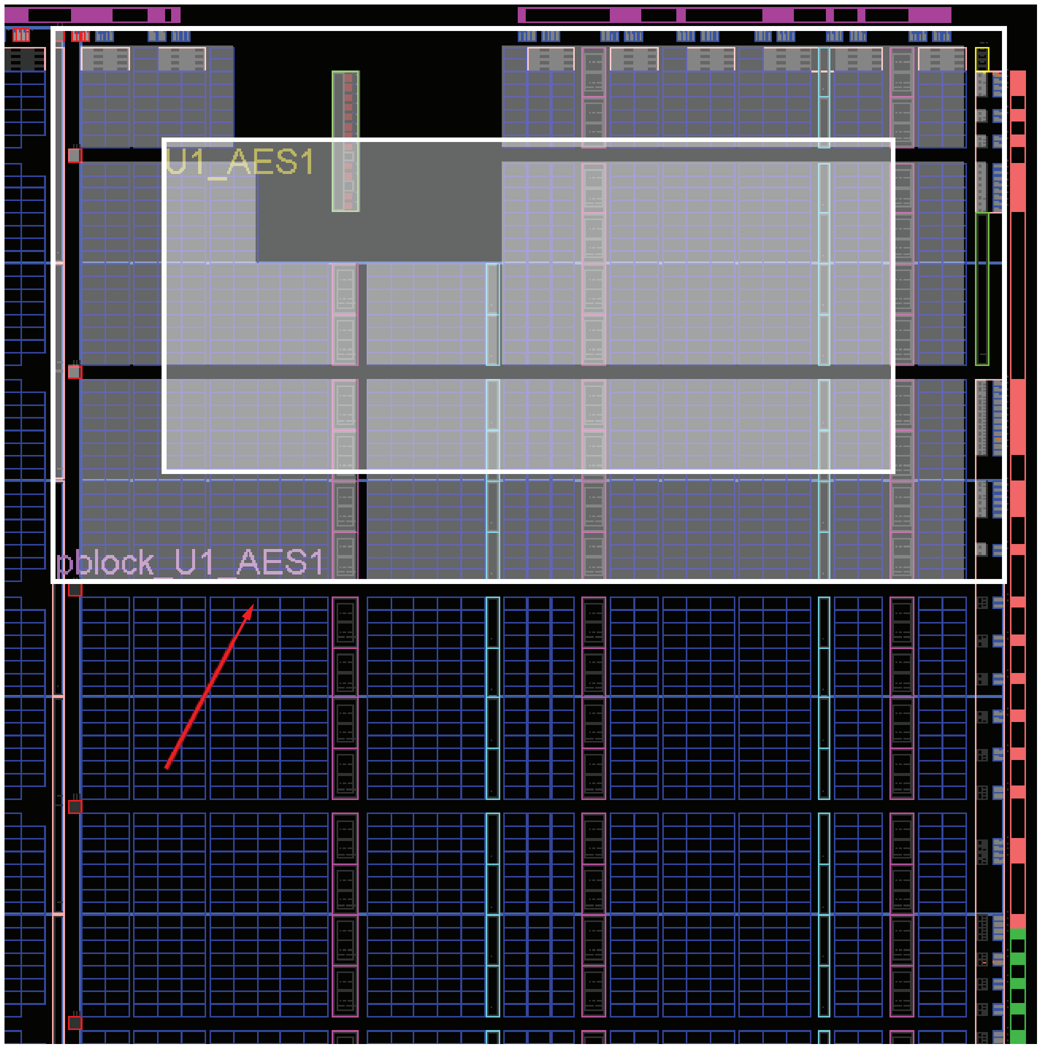


Figure 3-26: **pblock_U1_AES1 Pblock Layout**

11. Alternately, the Pblock rectangle (of the size specified in step 10) can be added to the design by entering this command string into the Tcl command line in the PlanAhead tool:

```
resize_pblock pblock_U1_AES1 -add {SLICE_X68Y152:SLICE_X127Y191
BUFDS_X2Y4:BUFDS_X2Y5 BUFH_X0Y320:BUFH_X3Y383
BUFIO2_X1Y26:BUFIO2_X4Y29 BUFIO2FB_X1Y26:BUFIO2FB_X4Y29
BUFPLL_X1Y4:BUFPLL_X2Y5 BUFPLL_MCB_X1Y9:BUFPLL_MCB_X2Y9
DCM_X0Y10:DCM_X0Y11 DSP48_X2Y38:DSP48_X3Y47
GTPA1_DUAL_X1Y1:GTPA1_DUAL_X1Y1 ILOGIC_X19Y136:ILOGIC_X35Y175
IODELAY_X19Y136:IODELAY_X35Y175 IPAD_X1Y8:IPAD_X1Y15
OLOGIC_X19Y136:OLOGIC_X35Y175 OPAD_X1Y4:OPAD_X1Y7
PLL_ADV_X0Y5:PLL_ADV_X0Y5 RAMB16_X3Y76:RAMB16_X5Y94
RAMB8_X3Y76:RAMB8_X5Y95} -locs keep_all -replace
```

12. The second rectangle is drawn to complete pblock_U1_AES1. Under the Physical Constraints tab, select the block **pblock_U1_AES1**.

13. Right-click **pblock_U1_AES1** and select **Add Pblock Rectangle** from the pull-down menu.

14. Draw a new rectangle below and to the far left side of the first rectangle to make a resultant L-shaped area, as shown in Figure 3-27. (The second rectangle does not have to be 100% accurate—it will be resized shortly.)

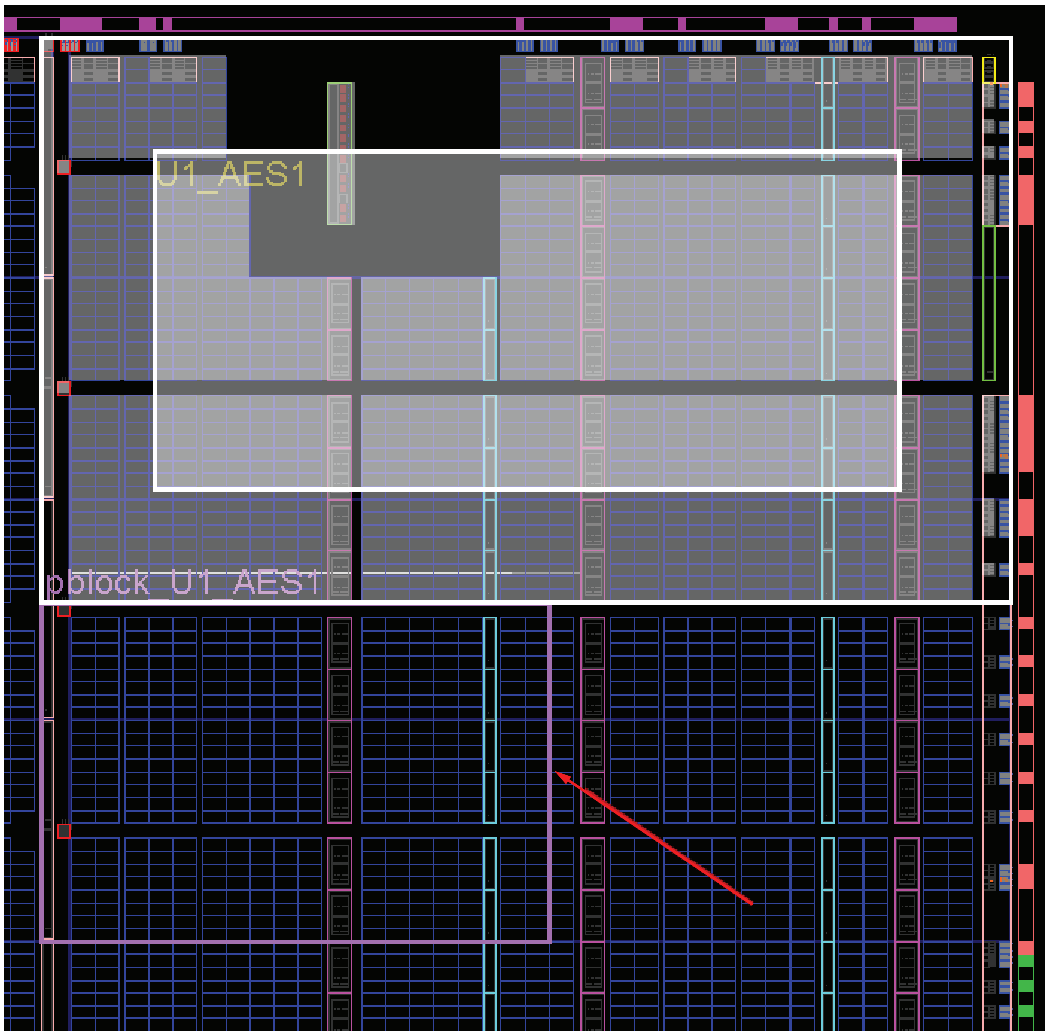


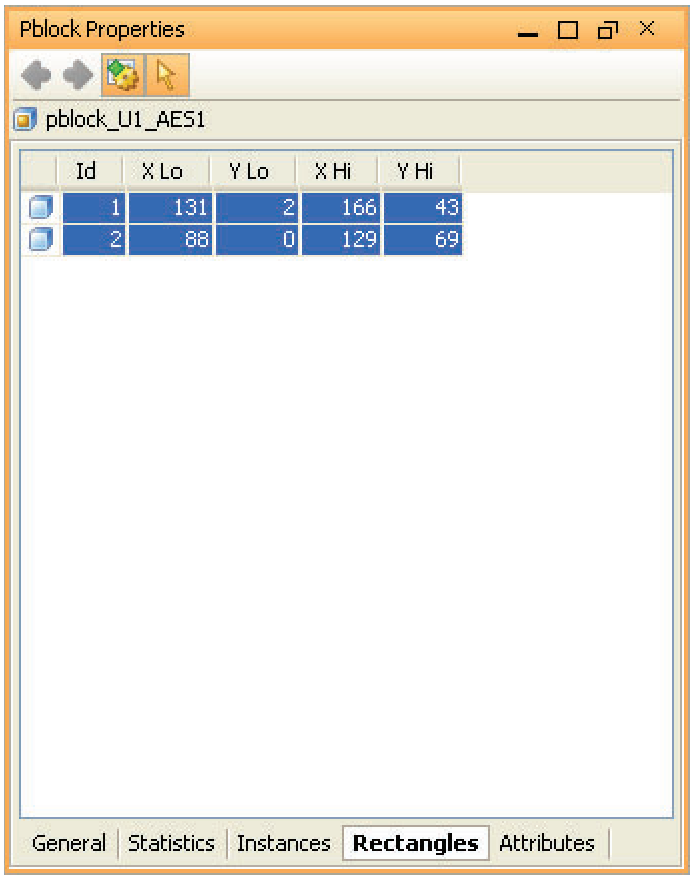
Figure 3-27: **pblock_U1_AES1 Pblock Layout - Add Pblock Rectangle**

15. A dialog box 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 used in the design, it is important to select them to take advantage of their routing resources. Failure to do so, while not an error, might produce designs that are unnecessarily difficult to route. Naturally, all used components must be included. The only exception is the MCB box, which should not be checked for this lab.

    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.

16. Click **OK**.
17. Ensure that **pblock_U1_AES1** is selected in the Physical Constraints pane.

18. Select the **Rectangles** tab in the Pblock Properties window to view all of the rectangles created for this Pblock (see Figure 3-28).



Id	X Lo	Y Lo	X Hi	Y Hi
1	131	2	166	43
2	88	0	129	69

X1104_c3_28_102510

Figure 3-28: **Pblock Properties**

Note: The PlanAhead tool might resize the rectangle to many different combinations of smaller rectangles. Make sure that all of the resources (such as block RAMs and DCMs) inside the area group are fully enclosed.

19. The format of multiple rectangles makes it more difficult to resize the second rectangle. Alternately, the second Pblock rectangle can be added to the design by entering the following command strings into the Tcl command line in the PlanAhead tool:

```
resize_pblock pblock_U1_AES1 -add {SLICE_X68Y128:SLICE_X101Y151
BUFH_X0Y256:BUFH_X3Y319 DCM_X0Y8:DCM_X0Y9 DSP48_X2Y32:DSP48_X2Y37
RAMB16_X3Y64:RAMB16_X3Y74 RAMB8_X3Y64:RAMB8_X3Y75} -locs keep_all

resize_pblock pblock_U1_AES1 -add {DCM_X0Y10:DCM_X0Y11
PLL_ADV_X0Y4:PLL_ADV_X0Y5} -remove {DCM_X0Y10:DCM_X0Y11
PLL_ADV_X0Y5:PLL_ADV_X0Y5} -locs keep_all
```

20. The completed Pblock for pblock_U1_AES1 is shown in Figure 3-29.

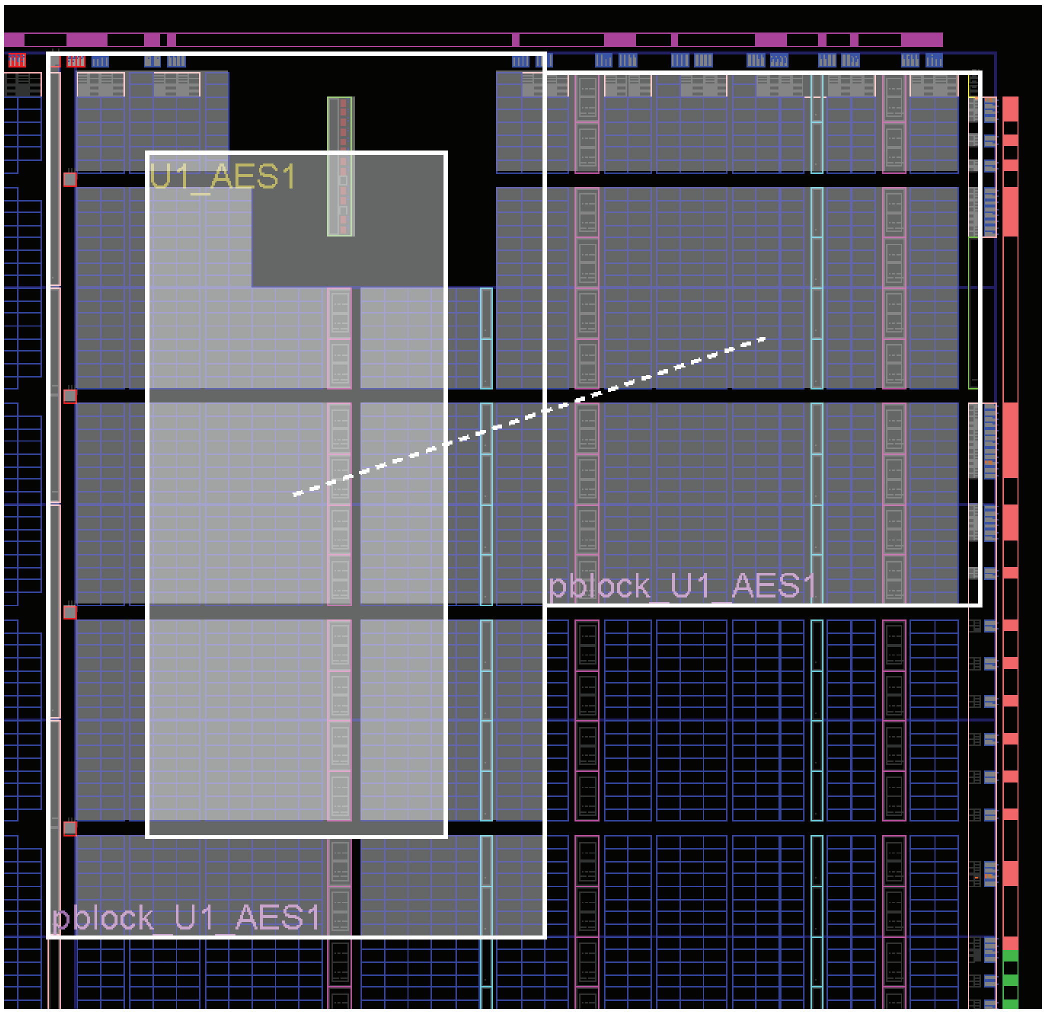
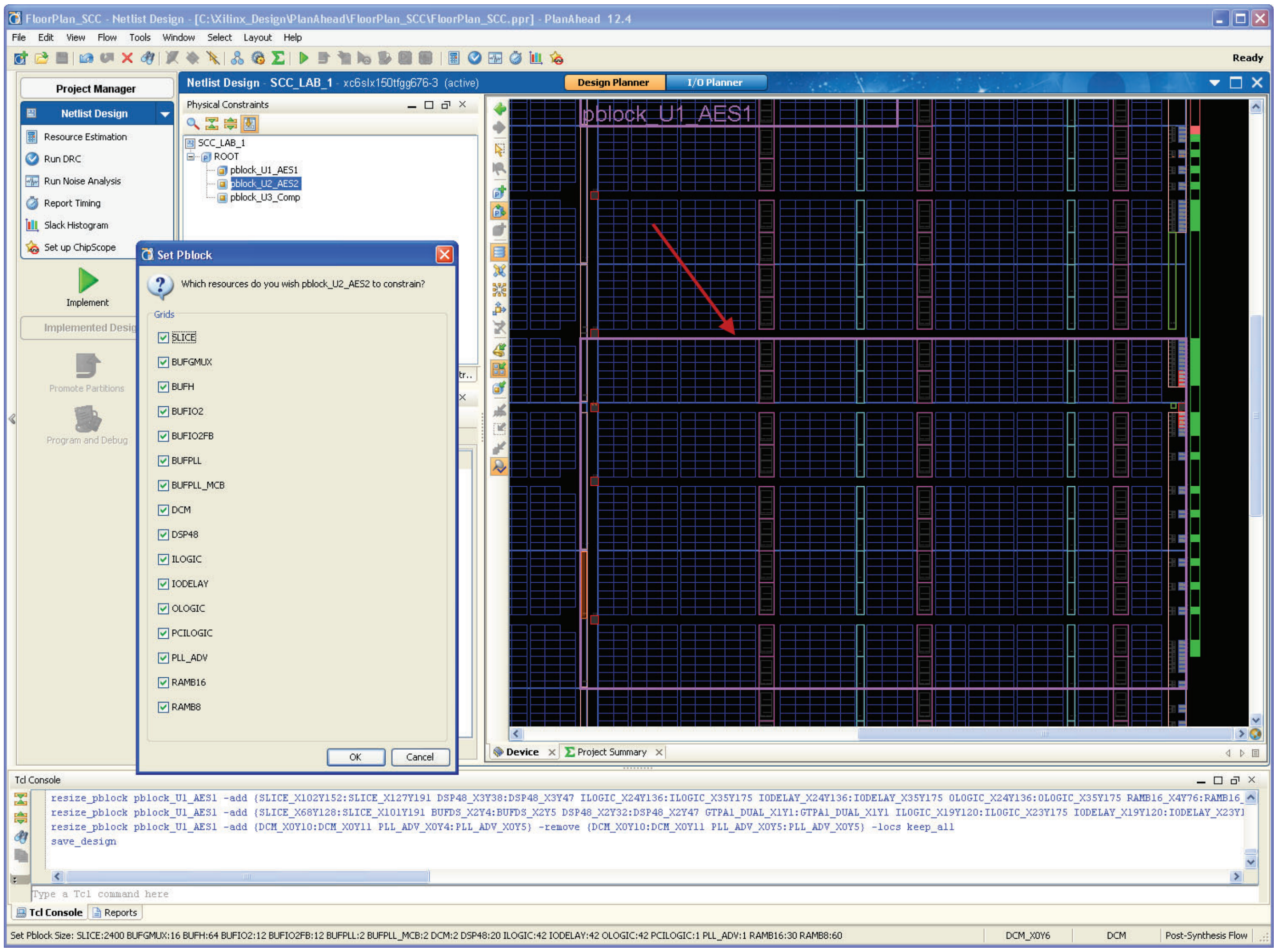


Figure 3-29: **Completed pblock_U1_AES1 Pblock Layout**

21. Under the Physical Constraints tab, select the block **pblock_U2_AES2**.
22. Right-click **pblock_U2_AES2** to select **Set Pblock Size** from the pull-down menu.

23. Draw a rectangle in the center right area of the Design Planner window, as shown in Figure 3-30. (The rectangle does not have to be 100% accurate—it will be resized shortly.)



```
resize_pblock pblock_U1_AES1 -add {SLICE_X102Y152:SLICE_X127Y191 DSP48_X3Y38:DSP48_X3Y47 ILOGIC_X24Y136:ILOGIC_X35Y175 IODELAY_X24Y136:IODELAY_X35Y175 OLOGIC_X24Y136:OLOGIC_X35Y175 RAMB16_X4Y76:RAMB16_
resize_pblock pblock_U1_AES1 -add {SLICE_X68Y128:SLICE_X101Y191 BUFDS_X2Y4:BUFDS_X2Y5 DSP48_X2Y32:DSP48_X2Y47 GTPA1_DUAL_X1Y1:GTPA1_DUAL_X1Y1 ILOGIC_X19Y120:ILOGIC_X23Y175 IODELAY_X19Y120:IODELAY_X23Y1
resize_pblock pblock_U1_AES1 -add {DCM_X0Y10:DCM_X0Y11 PLL_ADV_X0Y4:PLL_ADV_X0Y5} -remove {DCM_X0Y10:DCM_X0Y11 PLL_ADV_X0Y5:PLL_ADV_X0Y5} -locs keep_all
save_design
```

Set Pblock Size: SLICE:2400 BUFGMUX:16 BUFH:64 BUFIO2:12 BUFIO2FB:12 BUFPLL:2 BUFPLL_MCB:2 DCM:2 DSP48:20 ILOGIC:42 IODELAY:42 OLOGIC:42 PCILOGIC:1 PLL_ADV:1 RAMB16:30 RAMB8:60

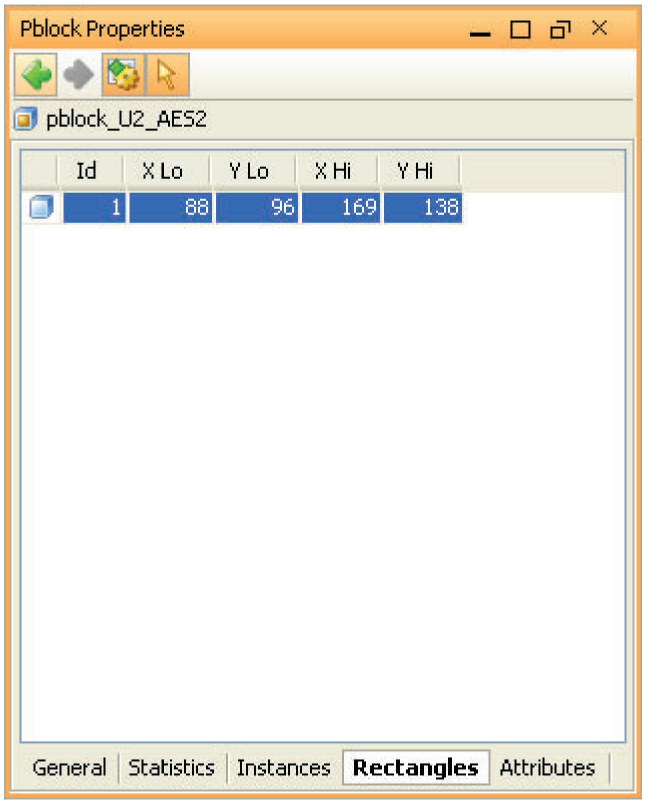
Figure 3-30: **pblock_U2_AES2 Pblock Layout (Set Pblock)**

24. A dialog box (shown in Figure 3-30) appears 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. Failure to do so, while not an error, might produce designs that are unnecessarily difficult to route. Naturally, all used components must be included. The only exception is the MCB box, which should not be checked for this lab.

    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.

25. Click **OK**.
26. Ensure that **pblock_U2_AES2** is selected in the Physical Constraints pane.

27. Select the **Rectangles** tab in the Pblock Properties window (see Figure 3-31).



Id	X Lo	Y Lo	X Hi	Y Hi
1	88	96	169	138

Figure 3-31: **Pblock Properties**

28. After the preliminary U2_AES2 block is drawn, adjust the rectangle graphically as necessary to match these coordinates:
    - X Lo = 88
    - Y Lo = 96
    - X Hi = 169
    - Y Hi = 138

    The Pblock area should look like Figure 3-32.

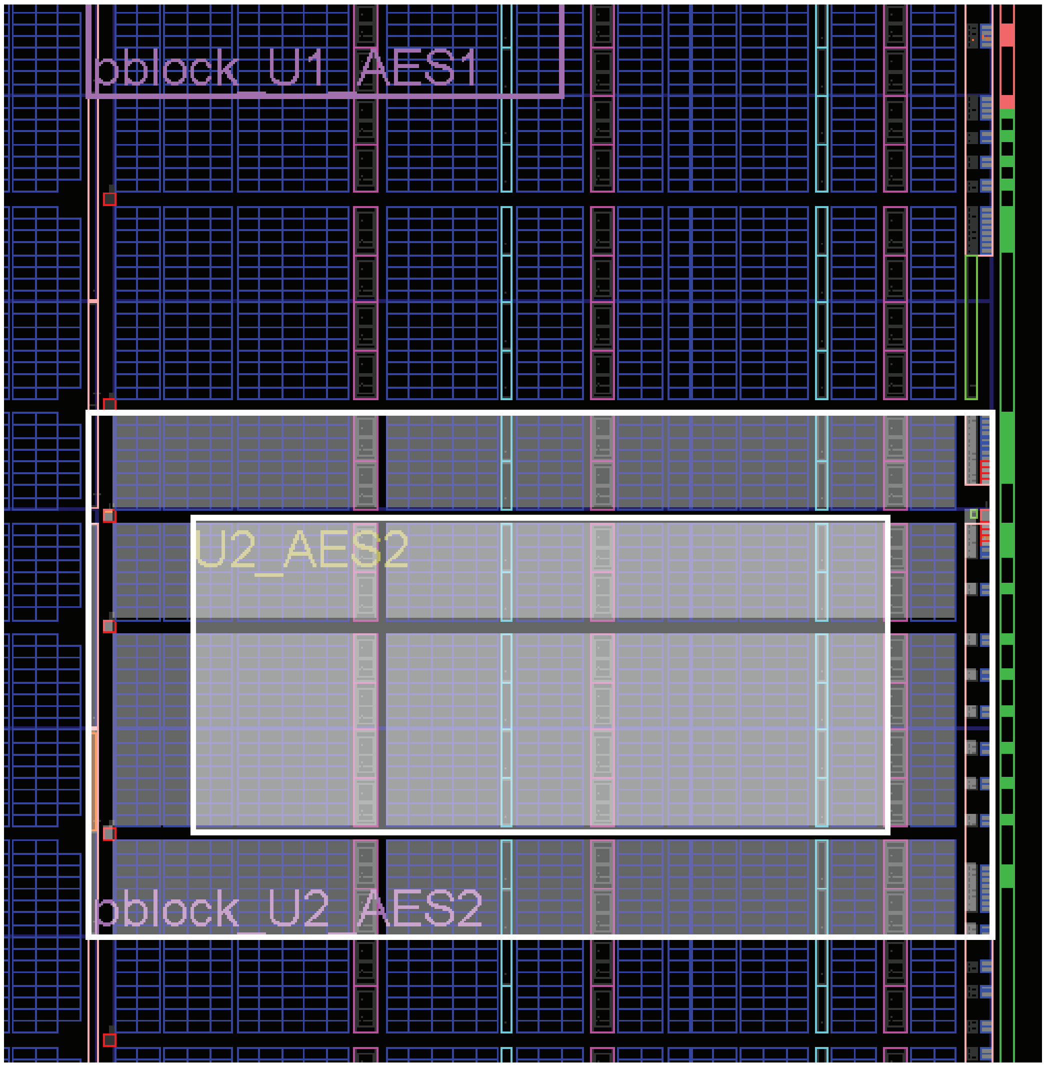


Figure 3-32: **pblock_U2_AES2 Pblock Layout**

29. Alternately, the Pblock rectangle (of the size specified in step 28) can be added to the design by entering the following command string into the Tcl command line in the PlanAhead tools:

```
resize_pblock pblock_U2_AES2 -add {SLICE_X68Y64:SLICE_X127Y103
BUFGMUX_X2Y1:BUFGMUX_X3Y16 BUFH_X0Y128:BUFH_X3Y191
BUFIO2_X1Y8:BUFIO2_X4Y23 BUFIO2FB_X1Y8:BUFIO2FB_X4Y23
BUFPLL_X1Y2:BUFPLL_X2Y3 BUFPLL_MCB_X1Y5:BUFPLL_MCB_X2Y5
DCM_X0Y4:DCM_X0Y5 DSP48_X2Y16:DSP48_X3Y25 ILOGIC_X19Y60:ILOGIC_X35Y101
IODELAY_X19Y60:IODELAY_X35Y101 OLOGIC_X19Y60:OLOGIC_X35Y101
PCILOGIC_X1Y0:PCILOGIC_X1Y0 PLL_ADV_X0Y2:PLL_ADV_X0Y2
RAMB16_X3Y32:RAMB16_X5Y50 RAMB8_X3Y32:RAMB8_X5Y51} -locs keep_all -
replace
```

Note: The fence between the AES1 and AES2 isolated partitions contains DSP tiles. Fence rules dictate that there must be two adjacent vertical DSP tiles in a horizontal fence to provide the required isolation.

30. The next rectangle for pblock_U2_AES2 is drawn up to, but not including, the DSP tile column. To complete the area needed for pblock_U2_AES2, multiple rectangles should be drawn to exclude the two DSP fence tiles. Under the Physical Constraints tab, select the block **pblock_U2_AES2**.
31. Right-click **pblock_U2_AES2** and select **Add Pblock Rectangle** from the pull-down menu.
32. Draw a new rectangle above and to the far left side of the first rectangle to make a resultant L-shape area, as shown in Figure 3-33. (The second rectangle does not have to be 100% accurate—it will be resized shortly.)

    Note: Remember to leave one fence tile of isolation between the AES1 and AES2 partitions.

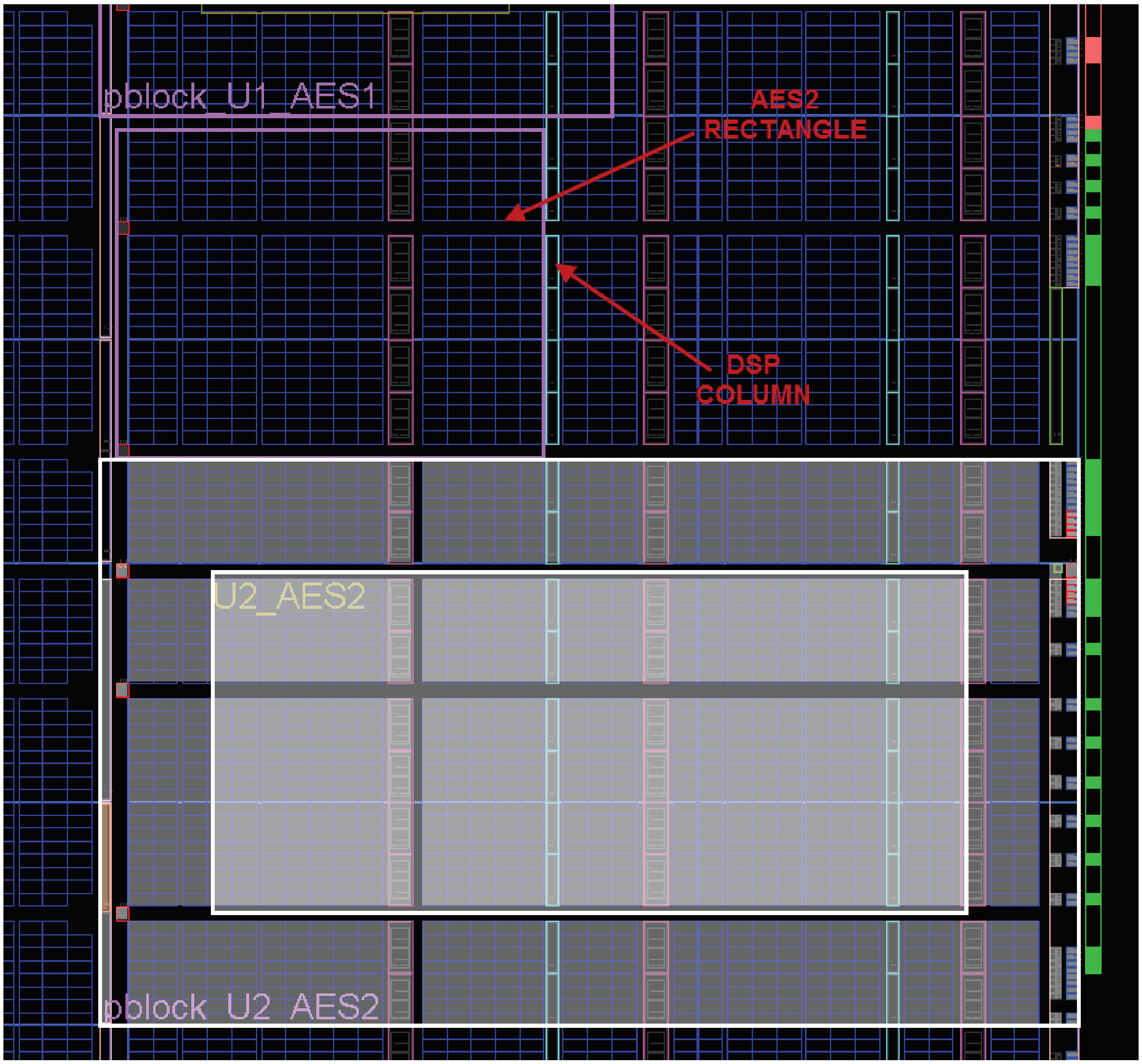
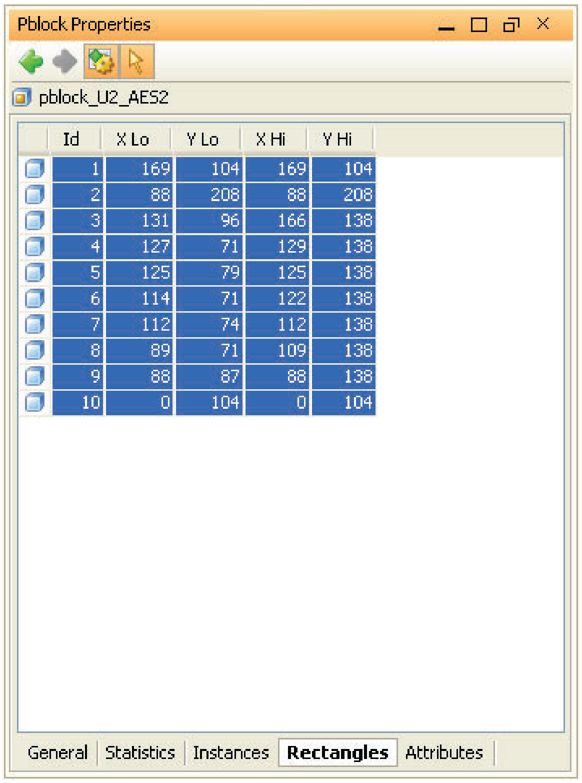


Figure 3-33: **pblock_U2_AES2 Pblock Layout**

33. A dialog box appears 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. Failure to do so, while not an error, might produce designs that are unnecessarily difficult to route. Naturally, all used components must be included. The only exception is the MCB box, which should not be checked for this lab.

    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.

34. Click **OK**.
35. Use the **Add Pblock Rectangle** option to draw the remaining two rectangles needed to complete the pblock_U2_AES2 area group. One rectangle should include the DSP tiles in this area group, and the last rectangle should include the remaining CLB tiles needed for this area group.
36. Ensure that **pblock_U2_AES2** is selected in the Physical Constraints pane.
37. Select the **Rectangles** tab in the Pblock Properties window to view all of the rectangles created for this pblock (see Figure 3-34).



Id	X Lo	Y Lo	X Hi	Y Hi
1	169	104	169	104
2	88	208	88	208
3	131	96	166	138
4	127	71	129	138
5	125	79	125	138
6	114	71	122	138
7	112	74	112	138
8	89	71	109	138
9	88	87	88	138
10	0	104	0	104

X1104_c3_34_062711

Figure 3-34: **Pblock Properties**

Note: The PlanAhead tool might resize the rectangle to many different combinations of smaller rectangles. Make sure that all of the resources (such as block RAMs and DCMs) inside the area group are fully enclosed. The user should ensure that there is a fence of one configurable logic block (CLB) tile, one BRAM tile, and two DSP tiles separation between the pblocks AES1, AES2, and Comp.

38. The completed pblock for pblock_U2_AES2 (aes_r) is shown in Figure 3-35.

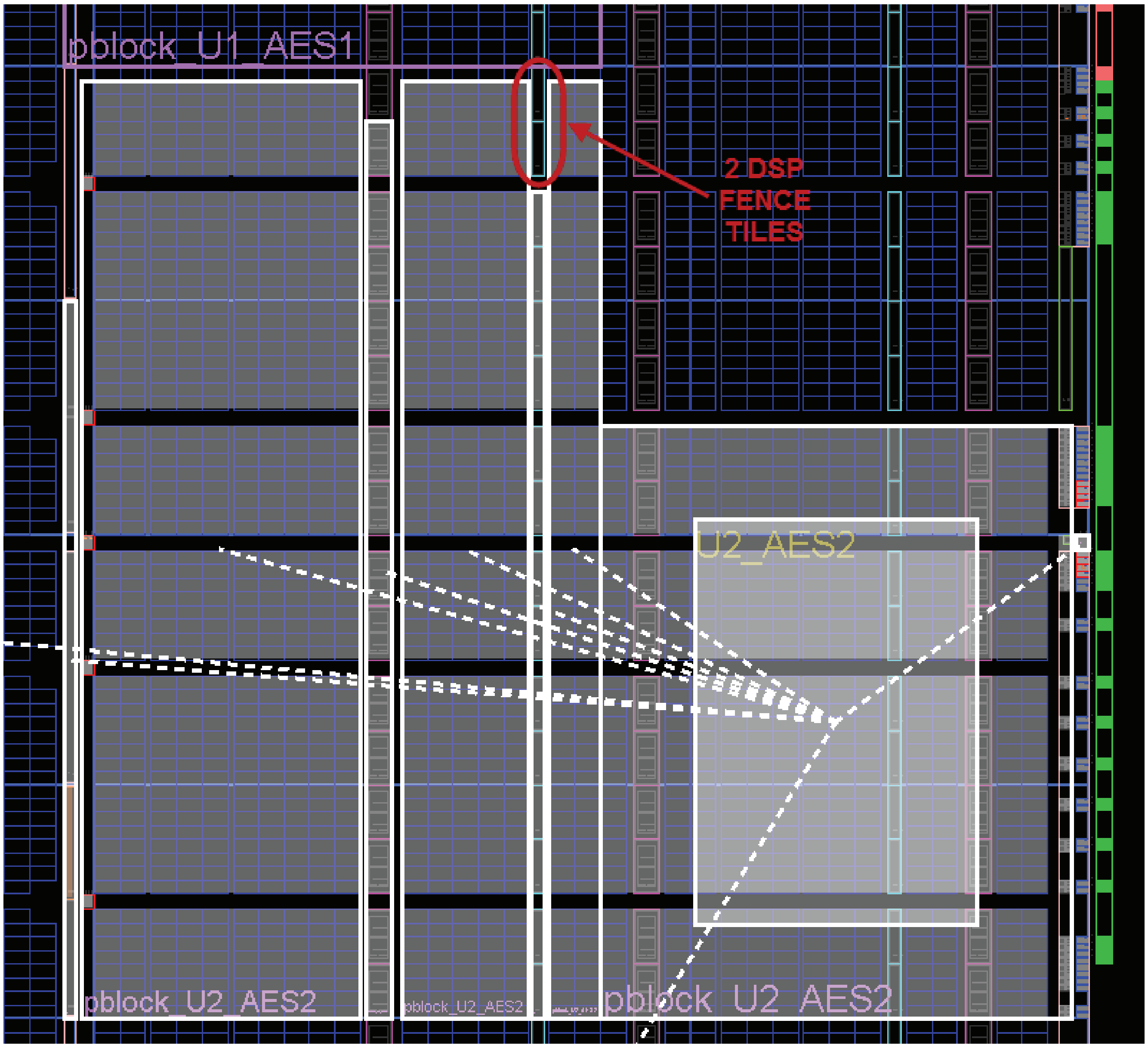
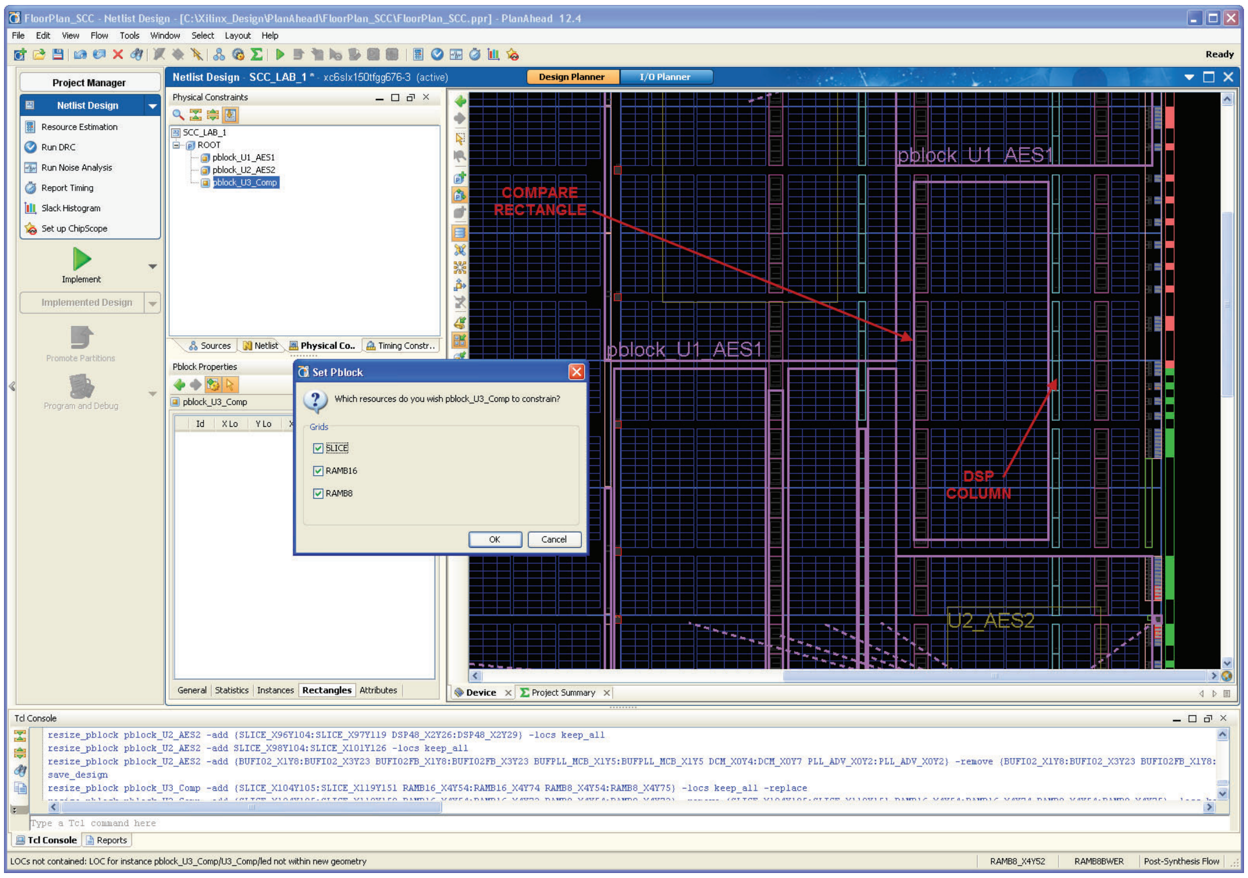


Figure 3-35: **Completed pblock_U2_AES2 Pblock Layout**

Note: The fence between the COMPARE isolated partition and AES1 and AES2 also contains DSP tiles. There must also be two adjacent vertical DSP tiles in these horizontal fences to provide the required isolation. Therefore, multiple rectangles are needed to define this pblock region to accommodate the DSP fence tiles.

39. Under the Physical Constraints tab, select the block **pblock_U3_Comp**.
40. Right-click **pblock_U3_Comp** to select **Set Pblock Size** from the pull-down menu.
41. Draw the first rectangle between the AES1 and AES2 blocks up to, but not including, the DSP column, as shown in Figure 3-36. (The rectangle does not have to be 100% accurate—it will be resized shortly.)



```
resize_pblock pblock_U2_AES2 -add {SLICE_X96Y104:SLICE_X97Y119 DSP48_X2Y26:DSP48_X2Y29} -locs keep_all
resize_pblock pblock_U2_AES2 -add SLICE_X98Y104:SLICE_X101Y126 -locs keep_all
resize_pblock pblock_U2_AES2 -add {BUFIO2_X1Y8:BUFIO2_X3Y23 BUFIO2FB_X1Y8:BUFIO2FB_X3Y23 BUFPLL_MCB_X1Y5:BUFPLL_MCB_X1Y5 DCM_X0Y4:DCM_X0Y7 PLL_ADV_X0Y2:PLL_ADV_X0Y2} -remove {BUFIO2_X1Y8:BUFIO2_X3Y23 BUFIO2FB_X1Y8:
save_design
resize_pblock pblock_U3_Comp -add {SLICE_X104Y105:SLICE_X119Y151 RAMB16_X4Y54:RAMB16_X4Y74 RAMB8_X4Y54:RAMB8_X4Y75} -locs keep_all -replace
```

LOCs not contained: LOC for instance pblock_U3_Comp/U3_Comp/led not within new geometry

X1104_c3_36_062411

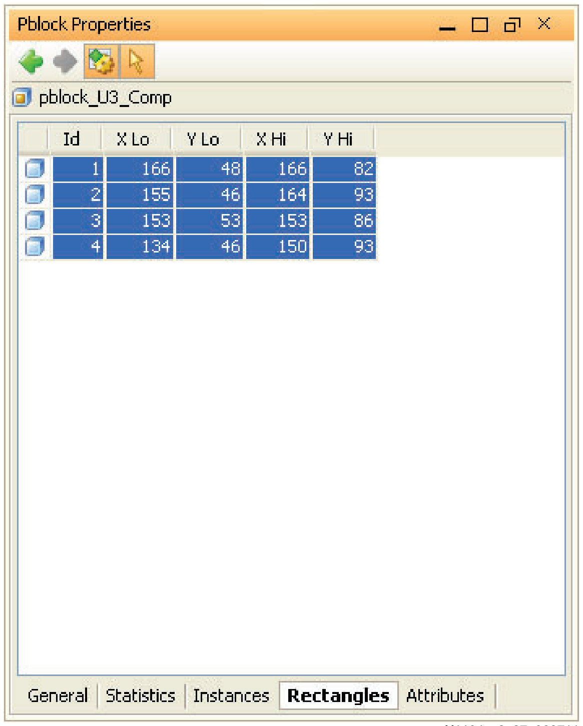
Figure 3-36: **pblock_U3_Comp Pblock Layout (Set Pblock)**

42. A dialog box (shown in Figure 3-36) appears 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 used in the design, it is important to select them to take advantage of their routing resources. Failure to do so, while not an error, might produce designs that are unnecessarily difficult to route. Naturally, all used components must be included. The only exception is the MCB box, which should not be checked for this lab.

    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.

43. Click **OK**.

44. In the Choose LOC Mode dialog box, select **Leave all location constraints in their current position**.
45. Click **OK**.
46. Use the **Add Pblock Rectangle** option to draw the remaining two rectangles needed to complete the pblock_U3_Comp area group. One rectangle should include the DSP tiles in this area group, and the last rectangle should include the remaining CLB tiles needed for this area group.
47. Ensure that **pblock_U3_Comp** is selected in the Physical Constraints pane.
48. Select the **Rectangles** tab in the Pblock Properties window (see Figure 3-37).



Id	X Lo	Y Lo	X Hi	Y Hi
1	166	48	166	82
2	155	46	164	93
3	153	53	153	86
4	134	46	150	93

Figure 3-37: **Pblock Properties**

Note: The PlanAhead tool might resize the rectangle to many different combinations of smaller rectangles. The user should ensure that there is a fence of one CLB tile, one BRAM tile, and two DSP tiles separation between the pblocks AES1, AES2, and Comp.

49. After the preliminary U3_Comp block is drawn, adjust the rectangle graphically as necessary to match these coordinates:
    - X Lo = 134
    - Y Lo = 46
    - X Hi = 166
    - Y Hi = 93

50. The completed pblock for pblock_U3_Comp (aes_r) is shown in Figure 3-38.

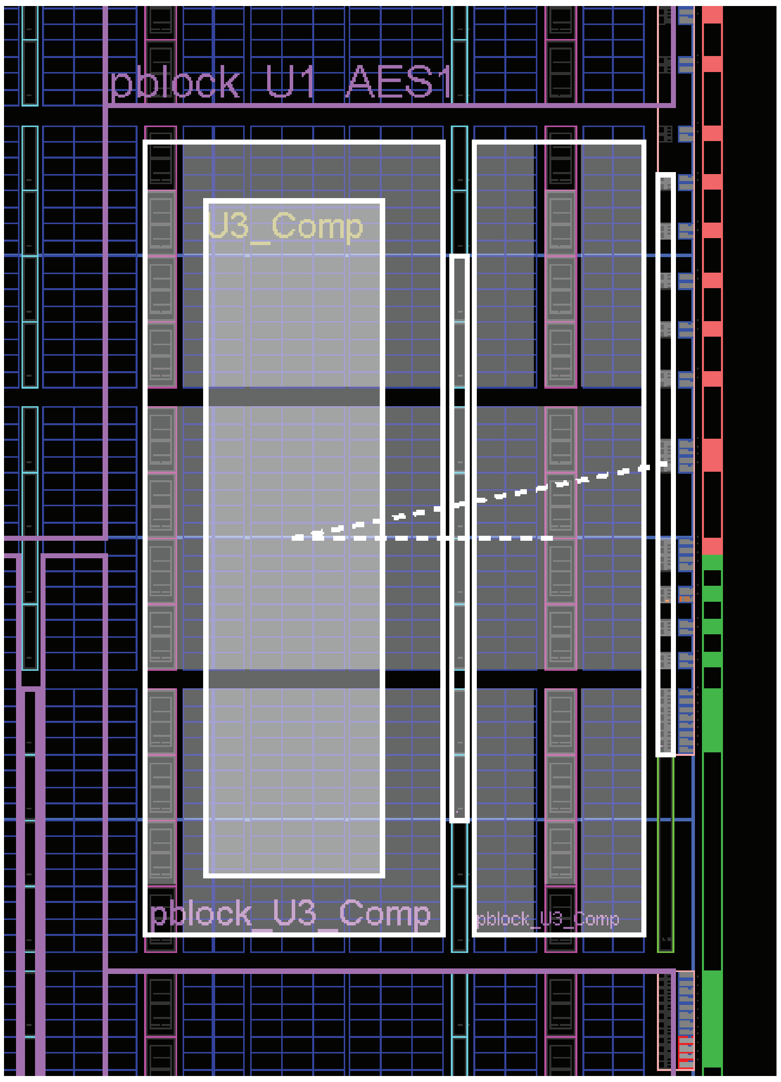
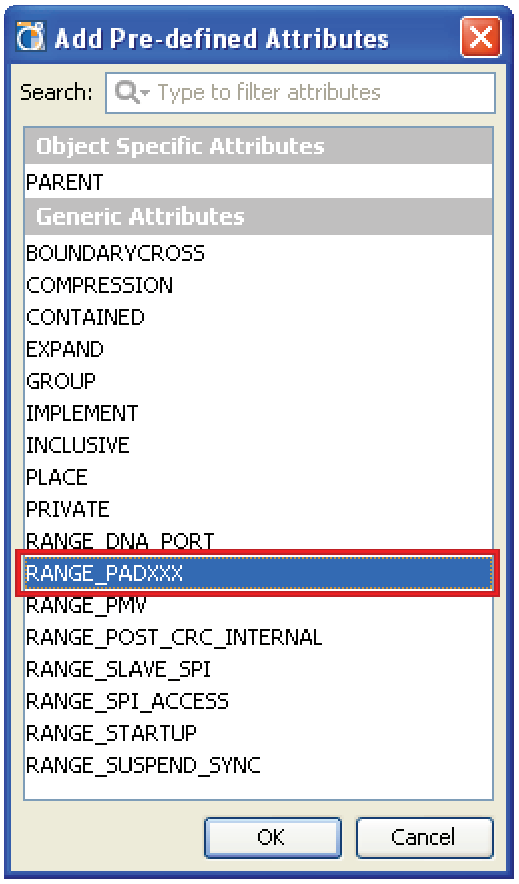


Figure 3-38: **pblock_U3_Comp Pblock Layout**

For the Spartan-6 FPGA, the pad ranges need to be set manually for the I/O blocks (IOBs) or pads used in the design to include them in each ISO partition with I/O pins. The pad range attribute (RANGE_PADXXX) can be set for each pblock in the PlanAhead tool GUI as follows:

51. Under the **Physical Constraints** tab, select the pblock **pblock_U1_AES**.
52. Right-click **pblock_U1_AES** and click **Pblock Properties** in the pull-down menu.
53. Click the **Attributes** tab of the Pblock Properties window, and click the **Add Pre-defined Attributes** button (a green "+" symbol).
54. Click the **RANGE_PADXXX** general attribute to select it, and click **OK** (see Figure 3-39).



Object Specific Attributes
PARENT
Generic Attributes
BOUNDARYCROSS
COMPRESSION
CONTAINED
EXPAND
GROUP
IMPLEMENT
INCLUSIVE
PLACE
PRIVATE
RANGE_DNA_PORT
RANGE_PADXXX
RANGE_PMV
RANGE_POST_CRC_INTERNAL
RANGE_SLAVE_SPI
RANGE_SPI_ACCESS
RANGE_STARTUP
RANGE_SUSPEND_SYNC

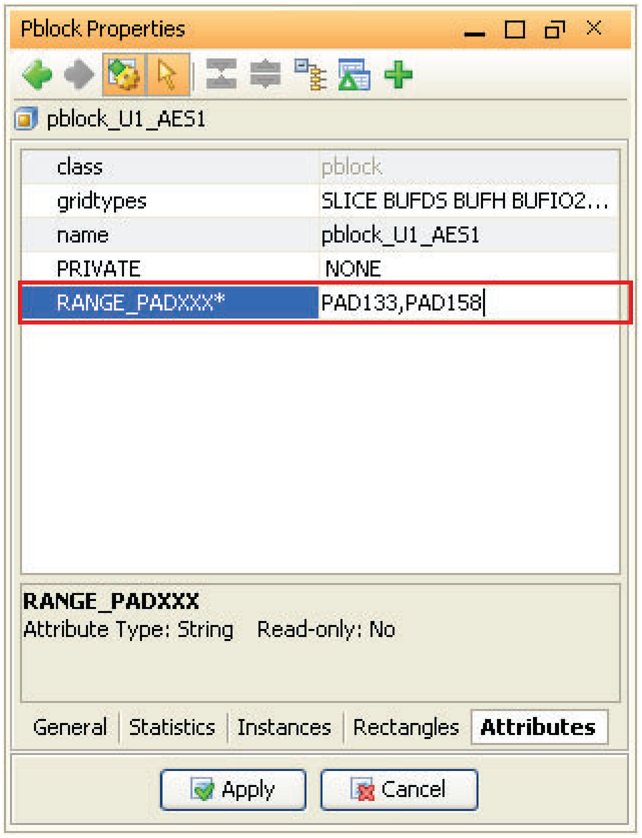
X1104_c3_39_102510

Figure 3-39: **Adding New RANGE_PADXXX Attribute**

55. Select the attribute **RANGE_PADXXX** in the general attributes list in the Pblock Properties window, and enter the pad range of **PAD133,PAD158** for pblock_U1_AES.

    Note: Do not insert a space between pads when entering the pad ranges into the Pblock Properties window.

56. Click **Apply** (see Figure 3-40).



class	pblock
gridtypes	SLICE BUFDS BUFH BUFIO2...
name	pblock_U1_AES1
PRIVATE	NONE
RANGE_PADXXX*	PAD133,PAD158

RANGE_PADXXX
Attribute Type: String Read-only: No

X1104_c3_40_102510

Figure 3-40: **Setting New RANGE_PADXXX Attribute**

57. Repeat step 51 to step 56 for **pblock_U2_AES2** and set the attribute **RANGE_PADXXX** to **PAD217**.

58. Repeat step 51 to step 56 for **pblock_U3_Comp** and set the attribute **RANGE_PADXXX** to **PAD190**.

59. The final layout is shown in Figure 3-41. Each block is separated by one user tile (except the DSP tile which is two tiles for a horizontal fence) to ensure IDF isolation. A CLB, block RAM, DSP, IOB, or any other site type that contains a global switch (GSM), can be used for this isolation.

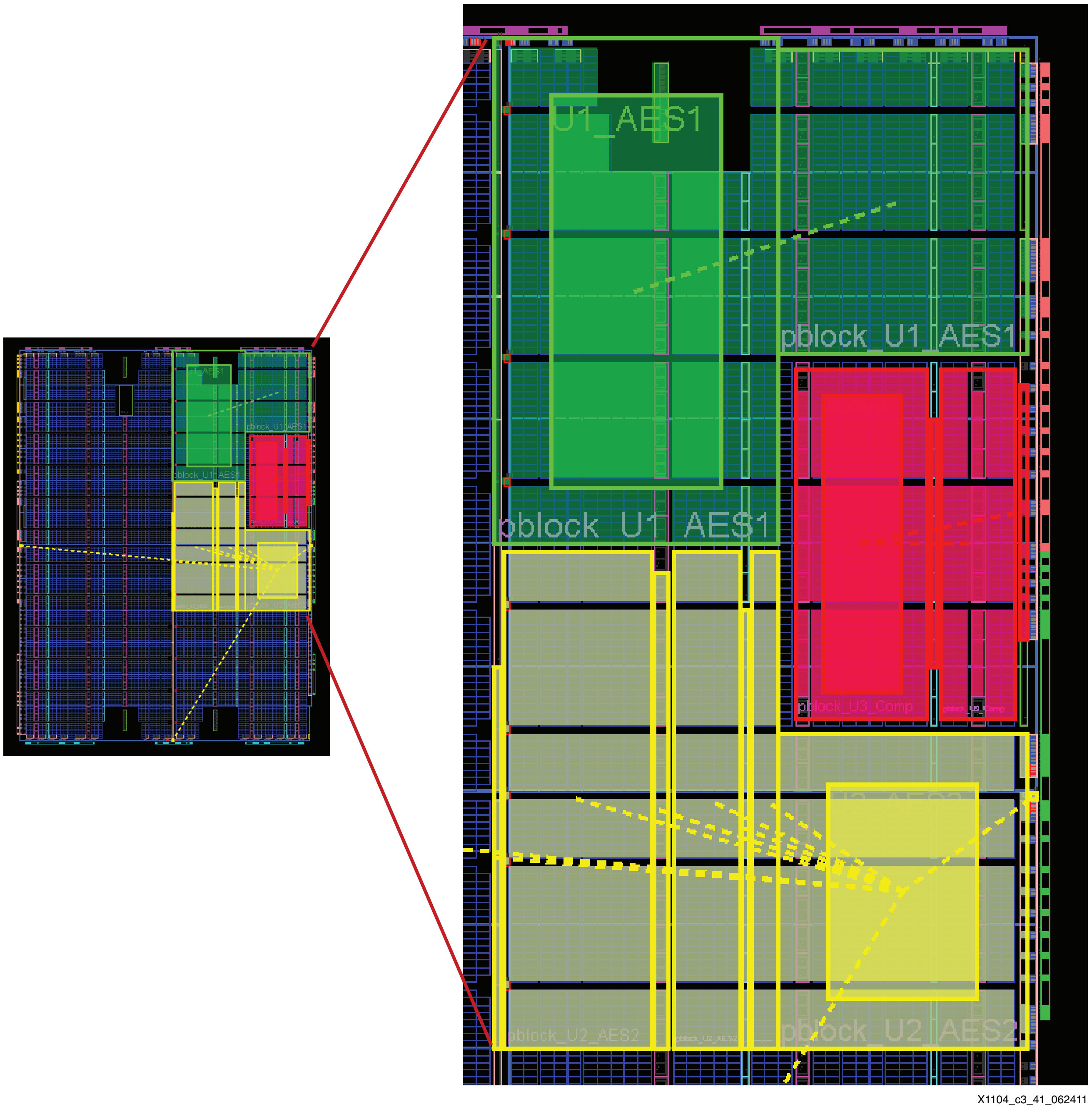


Figure 3-41: **Final Layout**

Isolation Design Flow Progress

The System Floorplanning block of the isolation design flow diagram is complete, as shown in Figure 3-42.

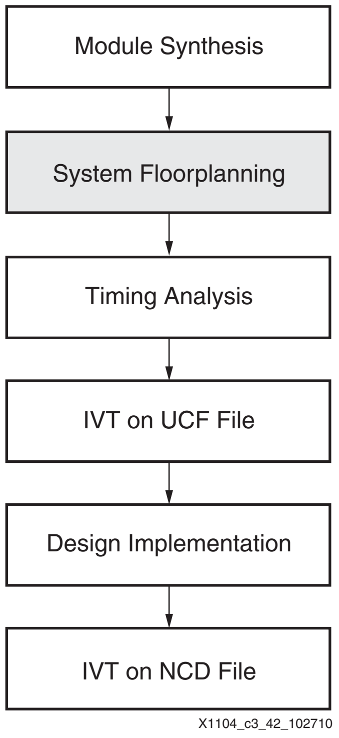
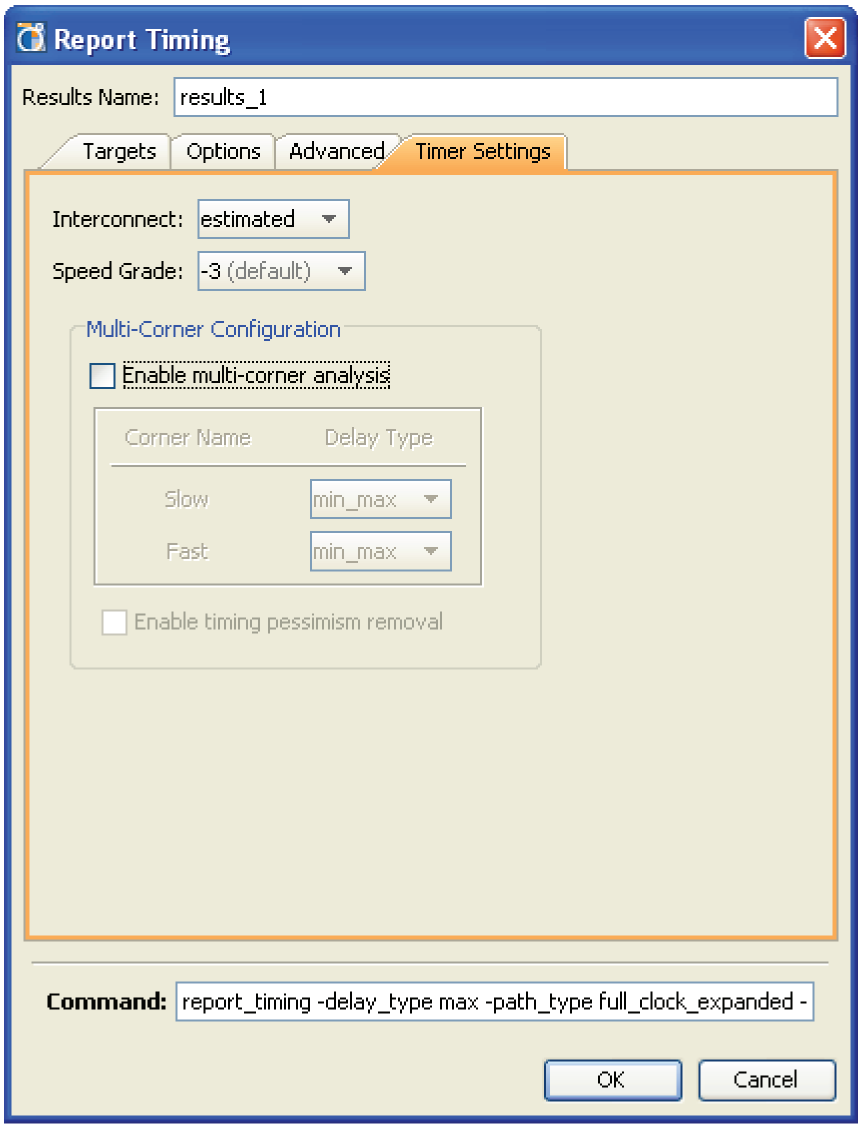


Figure 3-42: **Isolation Design Flow with System Floorplanning Block Complete**

Running Report Timing with the PlanAhead Tool

This section tests the timing constraints that were set up in Setting Up Timing Constraints with the PlanAhead Tool, page 59.

1. Select **Tools** → **Report Timing** and click **OK** to run report timing (see Figure 3-43).



Results Name: results_1

Interconnect: estimated

Speed Grade: -3 (default)

Enable multi-corner analysis [ ]

Corner Name	Delay Type
Slow	min_max
Fast	min_max

Enable timing pessimism removal [ ]

Command: report_timing -delay_type max -path_type full_clock_expanded -

Figure 3-43: **Report Timing Dialog Box**

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

   Note: The timing analysis performed through the PlanAhead tool is just a timing estimate and not a timing design sign-off step. The Xilinx® trce tool needs to be run after implementation to obtain the actual timing results.

Isolation Design Flow Progress

The Timing Analysis block of the isolation design flow diagram is complete, as shown in Figure 3-44.

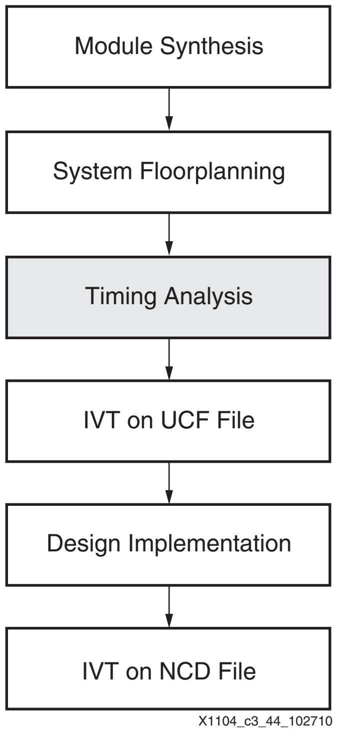


Figure 3-44: **Isolation Design Flow with Timing Analysis Block Complete**

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 UCF, or both, if desired. For this lab, only the UCF 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\PlanAhead` directory and keep the default filename `FloorPlan_SCC.edf`.
3. Click **OK**.
4. To export the constraints file, select **File** → **Export Constraints** and browse to the desired location or choose the default location and specify the UCF: `..\Xilinx_Design\PlanAhead\FloorPlan_SCC\SCC_LAB_TOP.ucf`.
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 native circuit description (NCD) test (required for IDF 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. The `ivt.zip` file posted on the Isolation Design Flow page contains the files needed for IVT:
   - `ivt.exe` – the IVT executables for these systems: nt, nt64, lin, and lin64
   - `IVT_End_User_License_Agreement.pdf` – the IVT user license
   - IVT User Guide and Release Note documentation (both in PDF format)
3. Create a new text file and name the file `SCC_LAB_TOP_ucf.ivt` with these contents:

```
-device xc6slx150t -package fgg676

# 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_TOP.pig

# User Constraint File
..\PlanAhead\Floorplan_SCC\SCC_LAB_TOP.ucf

# Output file
-output SCC_LAB_TOP_ucf.rpt
```

4. Note 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 points to the pin isolation group (PIG) file.
   e. IVT points to the UCF.
   f. IVT is told where to place the output file and its 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 or modify a text file and name the file `SCC_LAB_TOP.pig` with these contents:

```
# Place all Global (top level) signals here (each commented out)
# NET "clk" LOC = U25;

ISOLATION_GROUP AES BEGIN
  NET "reset" LOC = M19;
  NET "push_button" LOC = H20;
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 = L24;
END ISOLATION_GROUP
```

3. Note 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, page 91.

   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 as a starting point. 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. To set up the Xilinx environment variables for this command shell, run the system32 (`settings32.bat`) or system64 (`settings64.bat`) batch file from the ISE software installation directory (`...\Xilinx\12.4\ISE_DS`).
3. Navigate to the `..\Xilinx_Design\ivt` directory (where all of the IVT-related files should ultimately reside for the project).
4. Rename the IVT executable to `ivt.exe`.
5. Run the UCF test by typing the following at the command prompt:

   ```
   ivt -f SCC_LAB_TOP_ucf.ivt
   ```
6. The output for a successful UCF test run is "`SUCCESS!`".
7. 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. A sample of this file can be found in the `../ivt` directory.

Examining the Output from the IVT UCF Test

The output report has three key sections:

1. Area Range Constraints
2. Package Pins, I/O Buffers, and I/O Banks

   ```
   Pin(col, row)      Bank      I/O Buffer          Isolation Group     Net
   ---------------    ----      ---------------     ---------------     -------
   M19(  7,  14)      5         PAD158              AES                 reset
   H20(  6,  18)      5         PAD133              AES                 push_button
   L24(  2,  15)      1         PAD190              COMPARE             led
   ```
3. Isolation Verification Summary

   ```
   I/O Isolation
       I/O Buffer Isolation Violations: 0
       Package Pin Isolation Violations: 0
       Bank Isolation Violations: 0
   Area Group Isolation
       Area Group Isolation Violations: 0
   UCF Isolation Verification Summary
       Total Isolation Violations: 0
   Isolation analysis completed.
   Elapsed time: 0:00:12
   ```

IVT SVG Output File for UCF Mode

The SVG file `SCC_LAB_TOP_ucf.svg` created by IVT for UCF mode gives a graphical view of the Spartan-6 device with colored tiles denoting the ownership of the tiles by each isolated region. The SVG file also visually highlights that there is a proper fence isolating each of the regions (uncolored tiles). Figure 4-1 shows the SVG UCF mode output for the Spartan-6 FPGA design used in the SCC lab.

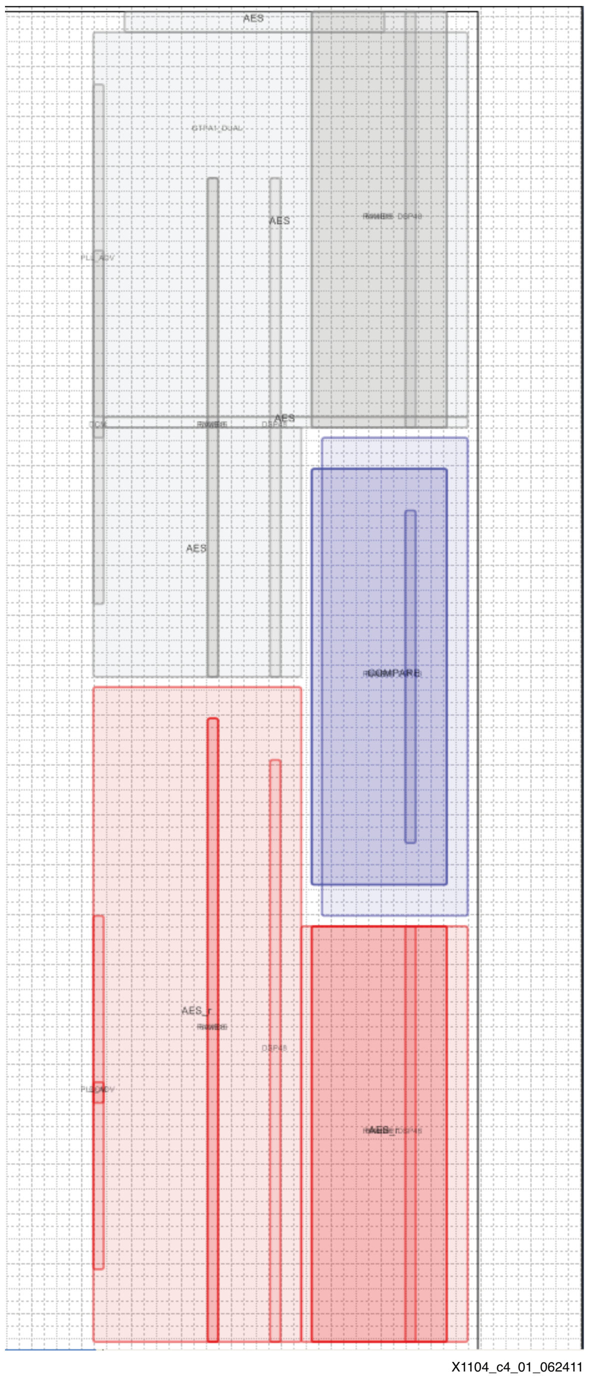


Figure 4-1: **SVG UCF Mode Output**

Isolation Design Flow Progress

The IVT on UCF File block of the isolation design flow diagram is complete, as shown in Figure 4-2.

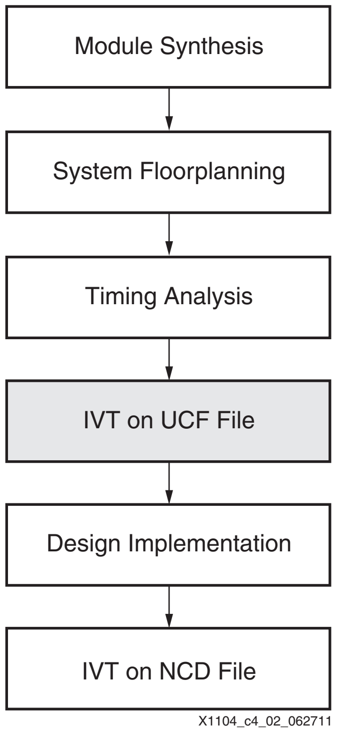


Figure 4-2: **Isolation Design Flow with IVT on UCF File Block Complete**

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. Click the bright green triangle on the Project Manager pane to implement the design.
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. Implement immediately starts the run: **NGDBUILD** → **MAP** → **Place and Route**.
3. The combined and routed design is placed in this directory:

```
..\Xilinx_Design\PlanAhead\FloorPlan_SCC\FloorPlan_SCC.runs\impl_1
```

4. Alternately, from the Project Manager pane in the PlanAhead™ tool:
   a. Select **Implement**, click the downward facing arrow to the right of the green Implement triangle, and select **Implementation Settings...** (see Figure 5-1).

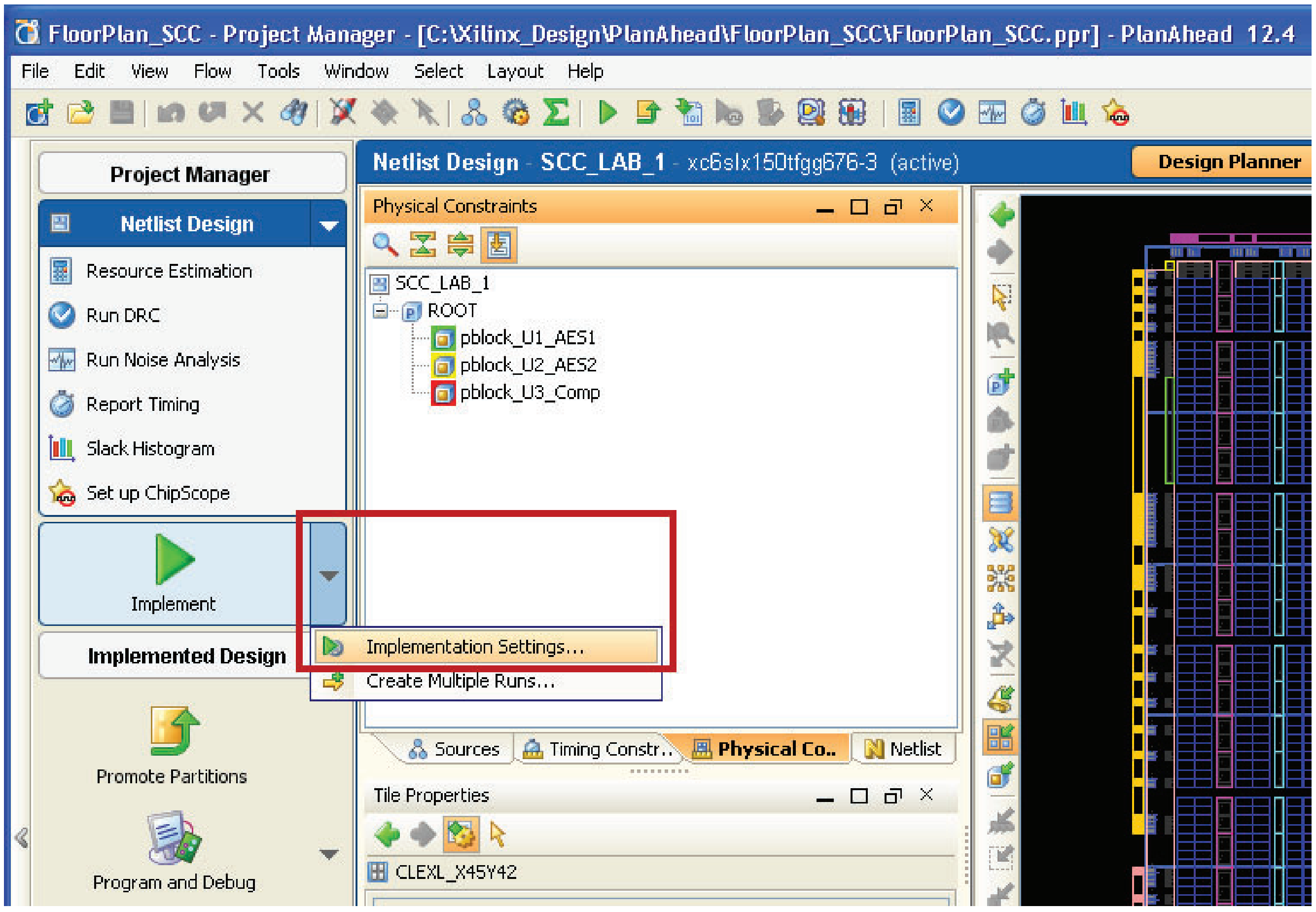
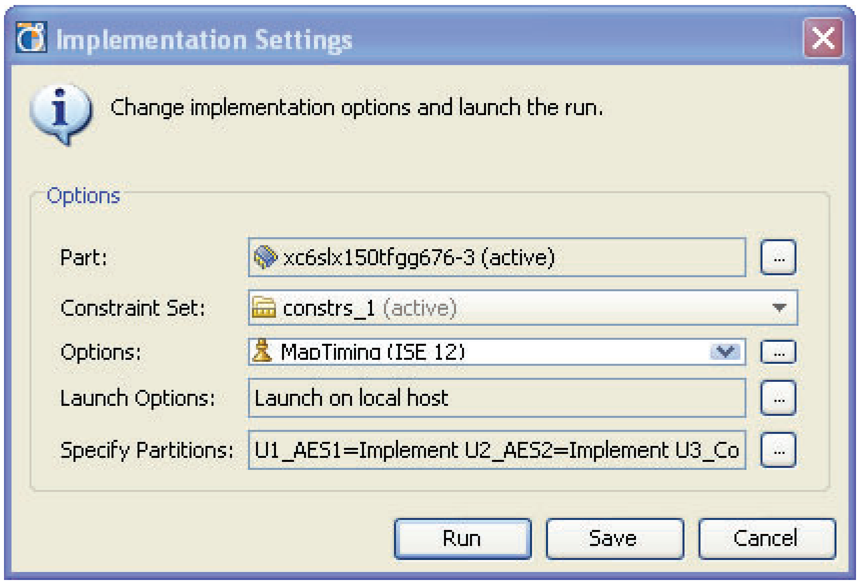


Figure 5-1: **Implement Design (Implementation Settings)**

b. An Implementation Settings dialog box appears (see Figure 5-2). To launch a run, change or accept the defaults and click **Run**.



Implementation Settings

Change implementation options and launch the run.

Options

Part:	xc6slx150tfgg676-3 (active)
Constraint Set:	constrs_1 (active)
Options:	MapTiming (ISE 12)
Launch Options:	Launch on local host
Specify Partitions:	U1_AES1=Implement U2_AES2=Implement U3_Co

Figure 5-2: **Implementation Settings**

5. A new tab named Design Runs is created, and a single entry named impl_1 is generated. It immediately starts the run: **NGDBUILD** → **MAP** → **Place and Route**.
6. An Implementation Completed dialog box appears after the design run is completed. Select the **Open Implemented Design** option and click **OK**.

7. The Device tab in the Design Planner pane shows the placed, routed, and partitioned design (see Figure 5-3).

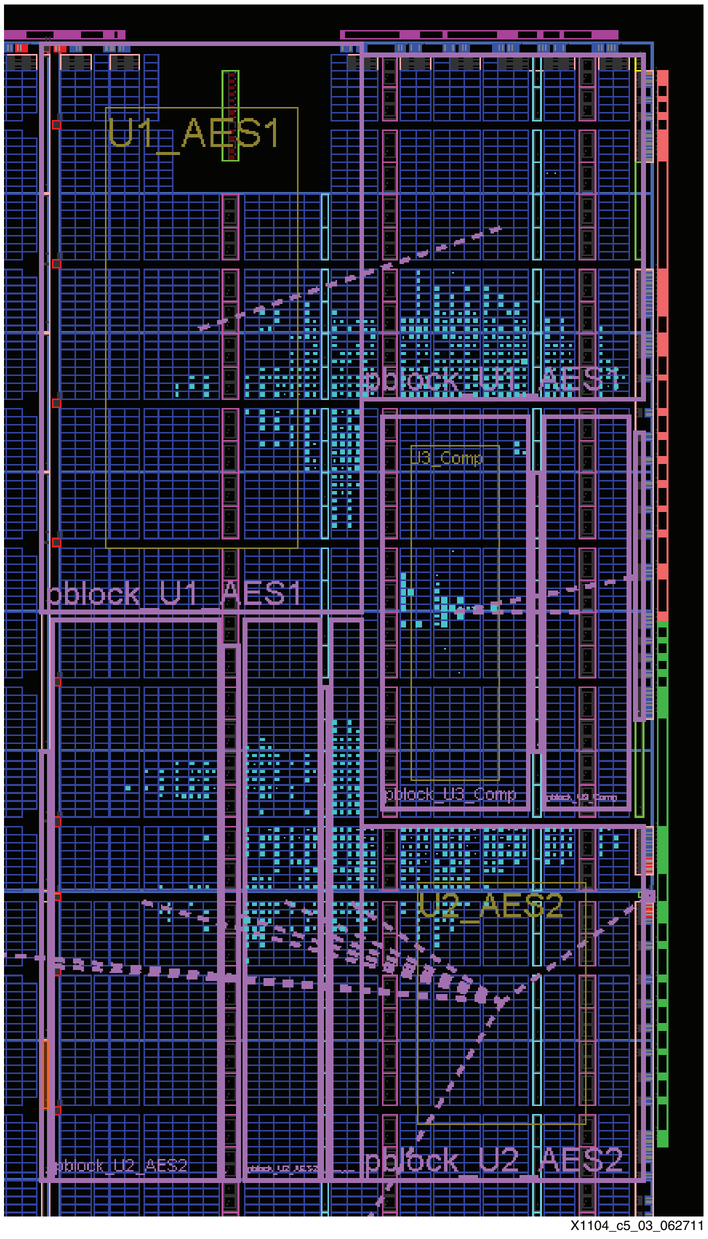


Figure 5-3: **Implemented and Floorplanned Design**

8. The file name for the combined and routed design is `SCC_LAB_TOP_routed.ncd`, and its view in FPGA Editor is shown in Figure 5-4.

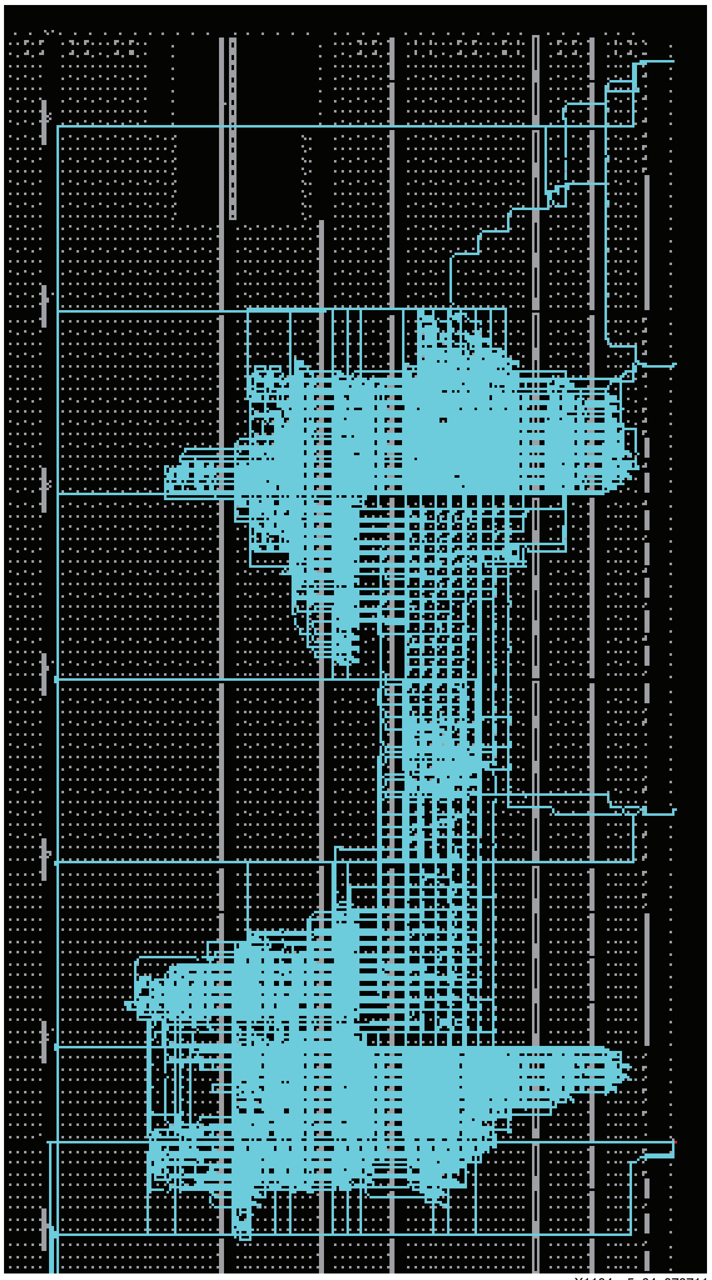


Figure 5-4: **FPGA Editor View**

Isolation Design Flow Progress

The Design Implementation block of the isolation design flow diagram is complete, as shown in Figure 5-5.

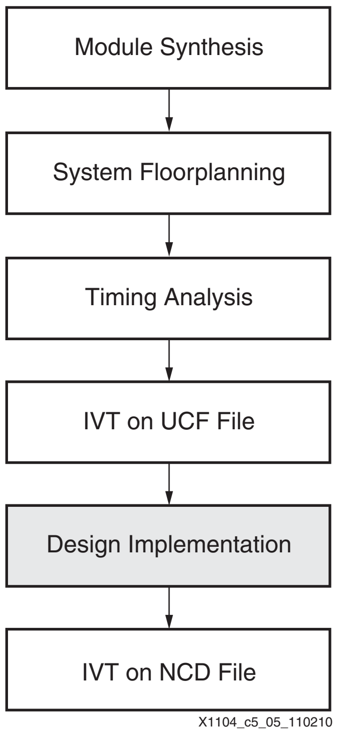


Figure 5-5: **Isolation Design Flow with Design Implementation Block Complete**

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 an example 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_TOP_ncd.ivt`
3. Open the text file `SCC_LAB_TOP_ncd.ivt` in a text editor.
4. Add these lines to the `SCC_LAB_TOP_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_1\
SCC_LAB_TOP_routed.ncd

# Output Report File
-output SCC_LAB_TOP_ncd.rpt
```

The NCD IVT command file sets these 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. Rename the IVT executable to `ivt.exe` or replace **`ivt`** in the command line with the name of the IVT executable.
4. To run the IVT NCD test, type the following at the command prompt:

   ```
   ivt -f SCC_LAB_TOP_ncd.ivt
   ```
5. The output for a successful NCD test run is "`SUCCESS!`".

Examining the Output from the IVT NCD Test

IVT creates two types of output files. The standard RPT file is a text report file. The SVG output file is a graphical view of the Spartan®-6 device with colored tiles denoting the ownership of the tiles by each isolated or PR area.

IVT RPT Output File

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_TOP_ncd.rpt
   ```

   All the groups are listed in the Isolated Modules section:

   ```
   Isolated Modules

   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 Uncategorized User Global Nets section. For SCC Trusted Routing designs, signals shared between isolated regions are expected and intended and appear here. An example of global signals are the following Global Clock signals, which are expected to be shared outside of an isolated region:

   ```
   Uncategorized User Global Nets

   The nets 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, global clocks, trusted
   inter-region signals, or explicitly permitted control signals may be
   global.

   clk_ibufg
   clk0_buf
   clkdev_buf
   ```
3. In the Categorized Nets section, ensure that all remaining Clocks are listed in the Nets Driven by Global Clock Sources section:

   ```
   The nets listed below present lesser risk than uncategorized nets due to
   their physical extents or signal sources.
   ```

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

clk_fb_i
clk_i
```

4. Ensure that only trusted bus macros are listed in the Trusted Bus Macro section.

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
-------------  ----  ----------  ---------------  -------
M19(  7,  14)   5      PAD158    AES              U1_AES1/reset_IBUF
L24(  5,  14)   1      PAD190    COMPARE          U3_Comp/led_OBUF
H20(  6,  18)   5      PAD133    AES              U1_AES1/push_button_IBUF
U25(  1,   9)   1      PAD217    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:

```
Isolation Verification Summary

  Tile Adjacency

    Net Adjacency Violations: 0
    Logic Adjacency Violations: 0

  Tile Content

    Net Content Violations: 0
    Logic Content Violations: 0

  Inter-region Signals

    Inter-region Net PIP Violations: 0
    Inter-region Load Violations: 0

  Special Fence Rules

    DSP Violations: 0

  I/O Isolation

    I/O Buffer Isolation Violations: 0
    Package Pin Isolation Violations: 0
    Bank Isolation Violations: 0

  NCD Isolation Verification Summary

    Total isolation violations: 0
    Unrouted nets: 0


  Isolation analysis completed.

  Elapsed time: 0:00:53
```

IVT SVG Output File for NCD Mode

The SVG file `SCC_LAB_TOP_ncd.svg` created by IVT gives a graphical view of the Spartan-6 device with colored tiles denoting the ownership of the tiles by each isolated region. The SVG file also visually highlights that there is a proper fence isolating each of the regions (uncolored tiles). Figure 6-1 shows the SVG output for the Spartan-6 FPGA design used in the SCC lab.

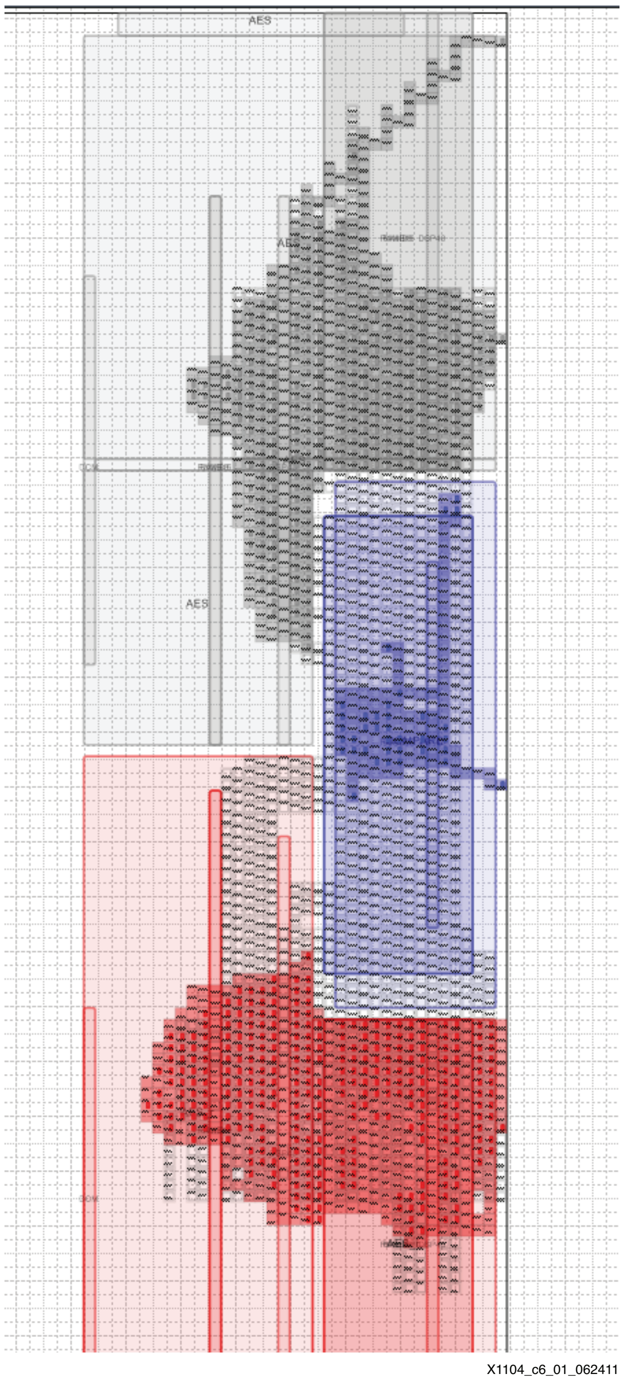


Figure 6-1: **IVT SVG File Graphical Output**

Isolation Design Flow Progress

The SCC lab is complete with the IVT on NCD File block of the flow diagram, as shown in Figure 6-2.

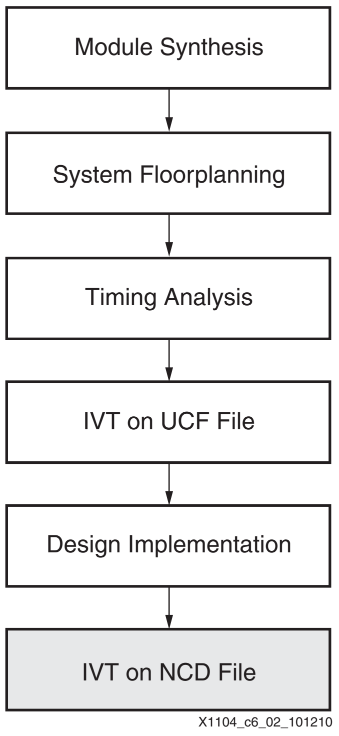


Figure 6-2: **Isolation Design Flow with IVT on NCD File Block Complete**

Appendix A

Tactical Patch Needed for ISE Tools 12.3 and 12.4

Due to a software issue in XST, a tactical patch must be used for ISE® Design Suite 12.3 and 12.4. The XST synthesis tool can intermittently remove the buffer on an I/O pin causing the backend implementation tools to optimize out (remove) the apparently unused pin.

Tactical Patch for ISE Tools 12.3 or 12.4 for Windows PC

This section describes how to apply the tactical patch for the ISE tools 12.3 or 12.4 for the Windows PC:

1. Obtain the tactical patch ZIP file for the ISE tools 12.3 or 12.4 at the Isolation Design Flow page on Xilinx.com.
2. Unzip and place the tactical patch in the ISE tools installation path location as shown in the following example (the example shows the installation on the C: drive of a Windows PC):

   ```
   C:\Xilinx\TacticalPatch_12.4
   ```
3. To point to the proper location for the patch, set the MYXILINX environment variable in the Systems Properties/Environment Variables window to point to the appropriate path where the patch resides, as in this example:

   ```
   Variable name:    MYXILINX
   Variable value:   C:\Xilinx\TacticalPatch_12.4\rtf
   ```
4. When the ISE software is invoked, the updated patch for XST runs automatically.

Tactical Patch for ISE Tools 12.3 or 12.4 for the Linux Server

This section describes how to apply the tactical patch for the ISE tools 12.3 or 12.4 for the Linux server:

1. Obtain the tactical patch ZIP file for the ISE tools 12.3 or 12.4 at the Isolation Design Flow page on Xilinx.com.
2. Unzip and place the tactical patch in a known location, as shown in the following example (the example shows the installation on a Linux server):

   ```
   $ setenv MYXILINX <install_path>/TacticalPatch_12.4/rtf
   ```
3. When the ISE software is invoked, the updated patch for XST runs automatically.