PlanAhead User Guide

This document is applicable to the following software versions: ISE Design Suite 14.6 and 14.7

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/26/2013	14.6	Added information on running ModelSim in Simulation Settings in Chapter 3.
		Add information on IP Settings in Chapter 3.
		Changed Compilation view to Using the Log view in Chapter 4.
		Removed Slack Histogram command from Chapter 7, Synthesized Design Constraints and Analysis.
		Updated creating new implementation runs in Defining Implementation Runs in Chapter 9.
		Minor updates throughout.

Table of Contents

Chapter 5: Elaborated RTL Design

Chapter 6: Synthesizing the Design

Chapter 7: Synthesized Design Constraints and Analysis

Chapter 8: I/O Pin Planning

Chapter 9: Implementing the Design

Chapter 10: Floorplanning the Design

Chapter 11: Analyzing Implementation Results

Chapter 12: Programming and Debugging the Design

Chapter 13: Using Hierarchical Design Techniques

Chapter 14: Tcl and Batch Scripting

Chapter 15: Using PlanAhead With Project Navigator

Appendix A: PlanAhead Input and Output Files

Appendix B: PlanAhead DRCs

Appendix C: Installing Releases with XilinxNotify

Appendix D: Configuring SSH Without Password Prompting

Appendix E: Additional Resources

☰ XILINX®

Chapter 1

About the PlanAhead Tool

The Xilinx® PlanAhead™ tool is a design tool for the entire FPGA design and implementation cycle.

For information about Xilinx tool installation and new features, see the following documents cited in Appendix E, Additional Resources:

- *Xilinx Design Tools: Installation and Licensing Guide (UG798)*
- *Xilinx Design Tools: Release Notes Guide (UG631)*
- *Known Issues for PlanAhead (AR40512)*

About PlanAhead Software

The PlanAhead tool is a design and analysis product that provides an environment for the entire FPGA design and implementation process. The PlanAhead tool is integrated with:

- Xilinx ISE® Design Suite synthesis and implementation tools
- System Generator for DSP design
- Xilinx Embedded Development Kit (EDK)
- Xilinx Synthesis Technology (XST) tool
- CORE Generator™ tool
- ChipScope™ Pro debugging tool
- ISE Simulator (ISim) tool
- XPower Analyzer tool
- FPGA Editor tool
- iMPACT device programming tool

The PlanAhead tool lets you improve circuit performance by defining and analyzing the Register Transfer Level (RTL) sources in the design, synthesized netlists, and implementation results. You can experiment with different implementation options, refine timing constraints, and apply physical constraints with floorplanning techniques to help improve design results. Early estimates of resource utilization, interconnect delay, power consumption, and routing connectivity can assist with appropriate logic design, device selection and floorplanning.

The PlanAhead tool includes a hierarchical data model that enables an incremental design capability referred to as Design Preservation. By partitioning the design, unchanged modules can be preserved, providing consistent results and in some cases reduced runtimes.

With appropriate licensing, the PlanAhead tool also provides access to the Partial Reconfiguration design application. See Chapter 13, Using Hierarchical Design Techniques, for an overview of these advanced design techniques.

You can use the PlanAhead tool as a stand-alone application, or launch it for specific purposes from the ISE software. A subset of features is available when launched from Project Navigator; this subset is called ISE Integration Mode. Refer to Chapter 15, Using PlanAhead With Project Navigator, for more information about integration with Project Navigator.

Using the PlanAhead Tool

Use the PlanAhead tool to:

- Manage the design data flow with a push button run process from RTL development through bitstream generation.

- Perform RTL design and analysis using an Elaborated Design.

- Perform behavioral and timing simulation using the integrated Xilinx ISim tool.

- Export I/O Buffer Information Specification (IBIS) models.

- Customize and implement IP using the integrated CORE Generator tool.

- Configure and launch multiple synthesis and implementation runs.

- Perform I/O pin planning.

- Manage constraints and perform floorplanning.

- Estimate the resource utilization, timing, and power consumption.

- Perform Design Rule Checks (DRCs).

- Debug core insertion and implementation with the ChipScope debugging tool.

- Perform device configuration and file generation using the iMPACT tool.

- Analyze implementation results.

- Launch programming and design verification tools.

- Work with third party tools such as:

  - Synplify Pro® FPGA synthesis software from Synopsys®

  - Mentor Graphics® ModelSim or Questa® Advanced Simulator tool

  - Precision® synthesis from Mentor Graphics

Project Creation and Management

The PlanAhead tool provides a variety of options for creating and managing FPGA design projects, and creating different constraint sets to explore design variations. See Chapter 3, Working with Projects.

RTL and IP Design

The PlanAhead tool Elaborated Design environment lets you create and manage RTL source files. You can customize and implement IP through integration with the CORE Generator tool. In addition to creating and managing design source files, the PlanAhead tool provides RTL source file elaboration enabling RTL logic exploration, an RTL Schematic Viewer, a set of RTL DRCs, and RTL-based resource and power estimation. For more information, see Chapter 5, Elaborated RTL Design.

Design Simulation

The PlanAhead tool also lets you launch Xilinx ISim to perform behavioral simulation of designs, and full timing simulation of implemented designs.

The PlanAhead tool prepares the required simulation executables and launches the ISim Graphical User Interface (GUI) to allow you to simulate the design, add and view signals in the waveform viewer, and examine and debug the design as needed. For more information, see:

- Chapter 5, Elaborated RTL Design for information about Behavioral simulation.
- Chapter 11, Analyzing Implementation Results for Timing simulation.

Synthesis and Implementation

The PlanAhead tool includes a synthesis and implementation environment supporting multiple synthesis and implementation runs using different physical or timing constraints, and different tool options to help complete routing and achieve timing closure. The synthesis and implementation runs can be queued to launch sequentially or simultaneously with multi-processor machines using the Xilinx ISE synthesis and implementation software.

- Chapter 6, Synthesizing the Design describes how to use synthesis.
- Chapter 9, Implementing the Design describes how to use implementation.

Design Analysis and Constraints Definition

The PlanAhead tool enables analyzing the design at each stage of the design process. Resource, timing, and power estimations along with DRCs let you experiment with various devices, constraints, and synthesis and implementation options to achieve desired results.

After implementation you can launch the FPGA Editor and XPower Analyzer tools directly from the PlanAhead tool for a detailed look at the implemented FPGA

These features are available both before and after implementation by analyzing the synthesized design or the implemented design results. For more information, see:

- Chapter 7, Synthesized Design Constraints and Analysis
- Chapter 11, Analyzing Implementation Results

Pin Planning

The PlanAhead tool provides an I/O Planning view to define and analyze the device I/O requirements. It supports creating and assigning I/O ports to physical package pins. This lets you define an I/O pinout configuration that satisfies the requirements of both the Printed Circuit Board (PCB) and the Field Programmable Gate Array (FPGA) designers. Chapter 8, I/O Pin Planning contains more information about I/O pin planning,

Floorplanning

The PlanAhead tool supports a floorplanning methodology that lets you group and constrain related logic to ensure shorter interconnect lengths with less delay, and to ensure more predictable implementation results. You can floorplan a design by creating physical block (Pblock) locations to constrain logic placement, or by locking individual logic objects to specific device sites. See Chapter 10, Floorplanning the Design and *Floorplanning Methodology Guide (UG633),* cited in Appendix E, Additional Resources, for more information about floorplanning.

Programming and Debugging Designs and ChipScope Integration

The PlanAhead tool is integrated with the Xilinx ChipScope debugging tool, which lets you add debugging cores to your design.

After implementation, you can access the ISE tools to generate bitstream files, and launch the iMPACT and ChipScope Analyzer tools directly from within the PlanAhead tool. See Chapter 12, Programming and Debugging the Design.

Hierarchical Design, Design Preservation, and Partial Configuration

The PlanAhead tool provides some advanced hierarchical design features to support team design, design preservation, and partial reconfiguration methodologies. See Chapter 13, Using Hierarchical Design Techniques.

The *Hierarchical Design Methodology Guide (UG748)*, cited in Appendix E, Additional Resources, more completely describes the hierarchical design process.

Tcl Commands and Batch Scripting

For information about the syntax of Tool Command Language (Tcl) commands and batch options in the PlanAhead tool, see Chapter 14, Tcl and Batch Scripting.

For a reference to the Tcl commands supported by the PlanAhead tool, see *PlanAhead Tcl Commands Reference Guide (UG789),* cited in Appendix E, Additional Resources.

Using PlanAhead Software with the ISE Project Navigator Environment

The PlanAhead tool is integrated with the Project Navigator software tool to provide an environment for improving your design results throughout the design flow.

Project Navigator automatically launches the PlanAhead tool at the following design steps:

- Pre-Synthesis:
  - I/O pin planning
- Post-Synthesis:
  - I/O pin planning
  - Floorplan Area/IO/logic
- Post-Implementation:
  - Analyze Timing and Floorplan design

See Chapter 15, Using PlanAhead With Project Navigator for information about ISE Integration mode.

Input and Output Files

The PlanAhead tool accepts a variety of input files and creates a variety of output file types and formats. See Appendix A, PlanAhead Input and Output Files for descriptions of the input and output files.

Design Rule Checks

The PlanAhead tool Design Rule Checks (DRCs) are listed in Appendix B, PlanAhead DRCs.

PlanAhead Software Terminology

For definitions of the terminology used in this Guide, see *Xilinx Glossary*, cited in Appendix E, Additional Resources.

Accessing Updates

Xilinx uses a `XilinxNotify` utility that notifies you when there are updates available. See Appendix C, Installing Releases with XilinxNotify for more information.

Configuring Multiple Linux Hosts

The multiple host capabilities for executing PlanAhead synthesis and implementation runs use the Linux Secure Shell (SSH). Before configuring multiple hosts in the PlanAhead tool, configure SSH so you are not prompted for a password each time you log in to a remote computer.

For more information, see Appendix D, Configuring SSH Without Password Prompting.

Additional Resources

Appendix E, Additional Resources contains links to all documents referenced in this guide. If you are using at a printed copy of this guide, the full URL to the referenced document is provided.

Launching the PlanAhead Tool

You can invoke the PlanAhead tool from any directory; however, running it from a project directory is advantageous because the PlanAhead log and journal files are written to the launch directory, making the project log files easy to find.

You can launch the PlanAhead tool and run the tools using different methods, or modes of operation, depending on your design preference. For example, you can choose to run the PlanAhead tool interactively from the command line using Tcl shell mode, or interactively with a graphical editor using the PlanAhead GUI. You can also run the PlanAhead tool automatically in batch mode, specifying a Tcl script to complete the design.

For help on the various modes and options for running the PlanAhead tool, you can enter the following from the Linux or Windows command line:

```
planAhead -help
```

Launching the PlanAhead tool Tcl Shell

Use the following command to invoke the interactive Tcl shell either at the Linux command prompt or within a Windows Command Prompt window:

```
planAhead –mode tcl
```

Launching the PlanAhead tool Using a Batch Tcl Script

You can run the PlanAhead tool in batch mode by supplying a Tcl script when invoking the tool. Use the following command either at the Linux command prompt or within a Windows Command Prompt window:

```
planAhead –mode batch –source <your_Tcl_script>
```

Note: When working in batch mode, the PlanAhead tool exits after running the specified script.

Launching the PlanAhead GUI

To launch the PlanAhead from the Windows Start menu, select the following command:

> **Start > All Programs > Xilinx Design Tools > ISE Design Suite 14.x >
> PlanAhead > PlanAhead**

Note: You can also double-click the PlanAhead tool shortcut icon on your desktop, or double-click the PlanAhead project file (.PPR) from the directory where the file resides.

The PlanAhead tool opens to the Getting Started Page as shown in Figure 1-1, page 13.

The Getting Started page assists you with creating or opening projects as well as viewing the documentation. To display the Getting Started page, close all open projects.

Launching the PlanAhead GUI from the Command Line

From the Windows or Linux command prompt, enter the following command:

> planAhead -mode gui

> -or-

> planAhead

Note: These two commands are the same. If you need help, type planAhead -help.

Starting the PlanAhead GUI from the Tcl Shell

To launch the PlanAhead tool GUI when running from the Tcl shell, enter the following command at the Tcl command prompt:

> start_gui

You can exit the GUI and return to the Tcl shell using the following command:

> stop_gui
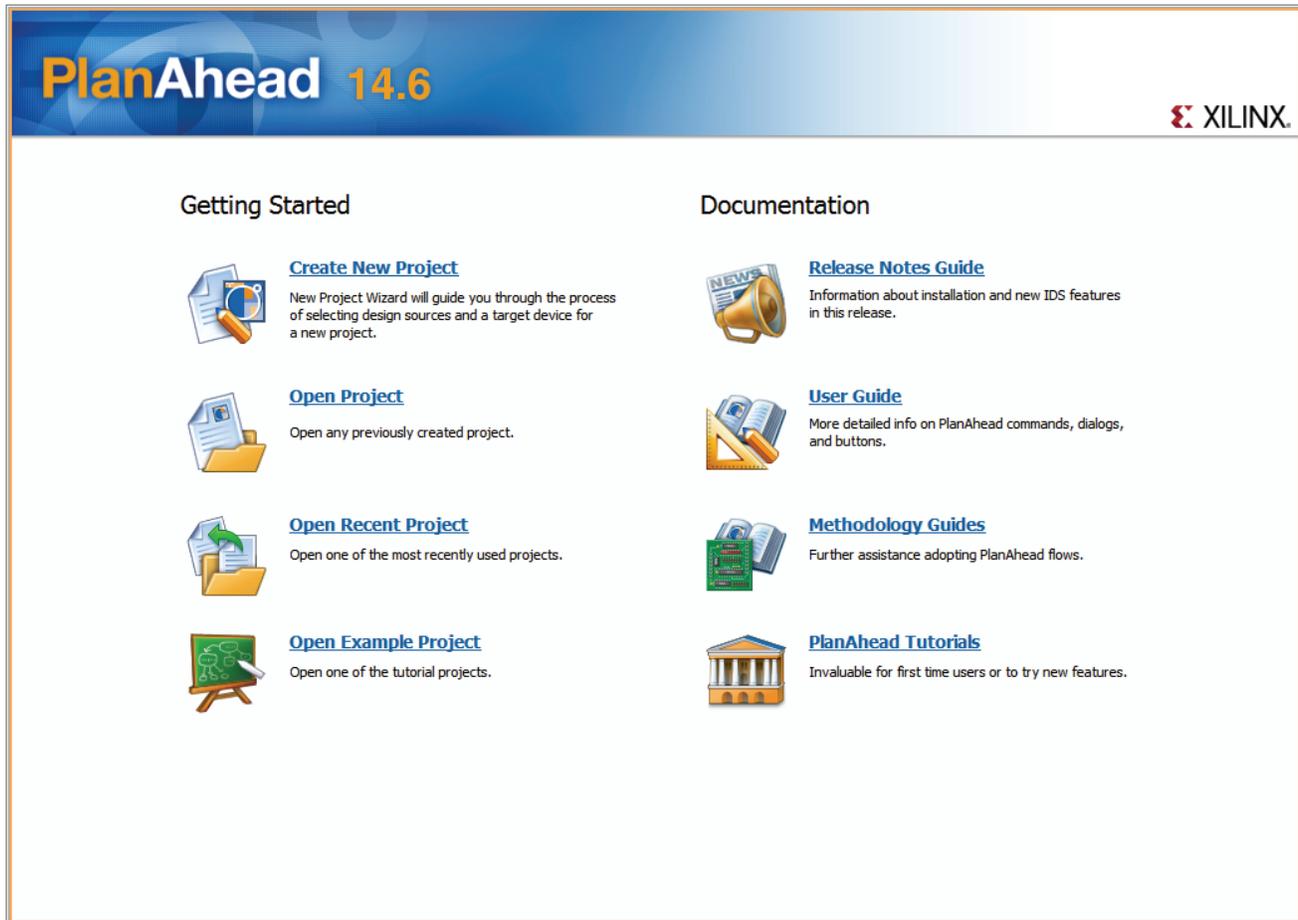
Figure 1-1: **PlanAhead Software Getting Started Page**

Using the Getting Started Page

The Getting Started page displays with the following options:

- **Create New Project:** Invokes the New Project wizard which lets you create any type of PlanAhead design project.

- **Open Project:** Invokes a browser enabling you to open any PlanAhead design project file (`.ppr`), or ISE Design Suite project file (`.xise`).

- **Open Recent Project:** Displays the last 10 previously-opened projects. Ten is the default; to change this number, use **Tools > Options > General**). The PlanAhead tool checks to ensure the project data is available before displaying the projects.

- **Open Example Project:** Provides four sample design projects:

  - **BFT Core:** A small RTL project BFT Core (`bft.ppr`)

  - **CPU (HDL):** A larger RTL project CPU (HDL)

  - **CPU (Synthesized):** A netlist-based project CPU (synthesized)

  - **Wave (HDL):** An IP example with three embedded IP cores from CORE Generator, Wave (HDL). You can use this design as a reference project to see how you can use IP cores with PlanAhead design projects.

To open or download the PlanAhead tool documentation, click the documentation links to launch a PDF viewer or website download location.

Note: The PlanAhead tool installs with a placeholder PDF file for the User Guide. In that file, you are guided to a web URL to download the latest document. There are simple installation instructions to enable the link to then invoke the full local copy of the User Guide when selected.

- **Release Notes Guide:** A web link to the Release Notes for a given release.

- **User Guide:** A link to this manual.

- **Methodology Guides:** A web link to Xilinx methodology guides.

- **PlanAhead Tutorials:** A web link to Xilinx tutorials and supporting design data.

Command Line Options

The PlanAhead tool has several command line options. To view the PlanAhead tool command line options, type the following command at the command prompt:

```
# planAhead -help
```

A help menu displays in the shell window.

Using a Startup Tcl Script

Use the **Tools > Run Tcl Script** command to run a script.

You can copy the PlanAhead tool Tcl commands from the `planAhead.jou` file, or from the Tcl console, to create startup scripts. Figure 1-2 shows a code snippet from a Tcl script.

```
..
#----------------------------------------------------------
create_project project_1 {C:\Data\PlanAhead_Designs\PlanAhead_Tutorial\Tutorial_Created_Data\project_1}
set_property design_mode RTL [get_property srcset [current_run]]
import_files -force -norecurse {C:\Data\PlanAhead_Designs\12_demo\Sources\Therm}
set_property library work [get_files -of_objects [get_property srcset [current_run]] {{C:\Data\PlanAhea
import_files -fileset [get_property constrset [current_run]] -force -norecurse {C:\Data\PlanAhead_Desig
set_property top therm [get_property srcset [current_run]]
set_property verilog_2001 true [get_property srcset [current_run]]
set_property verilog_uppercase false [get_property srcset [current_run]]
set_property loop_count 1000 [get_property srcset [current_run]]
launch_runs -runs synth_1 -jobs 1
launch_runs -runs impl_1 -jobs 1
close_project
```

Figure 1-2: **Example Tcl Script**

For more information about the PlanAhead tool journal file, see Journal File (planAhead.jou and planAhead_<id>.backup.jou), page 408. For more information about scripting and using the PlanAhead tool with Tcl, see Chapter 14, Tcl and Batch Scripting.

The PlanAhead Tool Flow

You can use the PlanAhead™ tool at many places in the design flow. The flows described in this chapter correspond to the project types that can be created. Project types are described in more detail in Chapter 3, Working with Projects.

You have access to design analysis and constraint definition capabilities at each stage of the design flow, including elaborated Register Transfer Level (RTL) design, synthesized design, and any of the implemented design runs.

Design flows supported by the PlanAhead tool include:

*   RTL to Bitstream - Use the PlanAhead tool to manage the entire FPGA design process from RTL development, IP customization, synthesis and implementation through to programming the device. You can add Verilog and VHDL sources, previously defined and configured intellectual property (IP) cores, DSP modules from System Generator, embedded processor designs from EDK, and physical and timing constraints to a project. You can experiment with synthesis, simulation, and implementation options and constraints to help meet your design objectives.

*   Synthesized Netlist to Bitstream - The PlanAhead tool manages netlist designs from implementation to device programming. You can add synthesized netlists, netlist-based IP cores, and constraints to a project. You can experiment with implementation options and constraints to help meet your design objectives.

*   Device Exploration and I/O Pin Planning - The PlanAhead tool provides an I/O Planning environment to analyze the device resources and visualize the relationship between the FPGA and system-level designs. Proper clocking and I/O planning can improve device performance and routability. Early I/O planning can also improve PCB routing, signal integrity, and the performance of the overall system. You can also migrate an I/O Pin Planning Project into an RTL Project, using the pin plan as the starting point for the top-level of the design.

*   Analyze Implemented Design Results - Use the PlanAhead tool to analyze implementation results generated outside of the PlanAhead tool, such as in Xilinx ISE® Design Suite. You can examine placement and timing results to explore design changes, timing adjustments, or floorplanning to achieve timing closure.

*   Partial Reconfiguration[1] - The PlanAhead tool provides an environment to set up and manage partial reconfiguration projects. These designs require special features and project structure to manage reconfigurable modules. For more information, see the *Partial Reconfiguration User Guide (UG702)*, cited in Appendix E, Additional Resources.

---

1. Partial Reconfiguration is only available under special licensing.

Design Flow

The PlanAhead design flow depends on the type of input file that you start with. This section describes the design flow and design tasks. Figure 2-1 shows the common design flows, and the input and outputs of the PlanAhead tool.
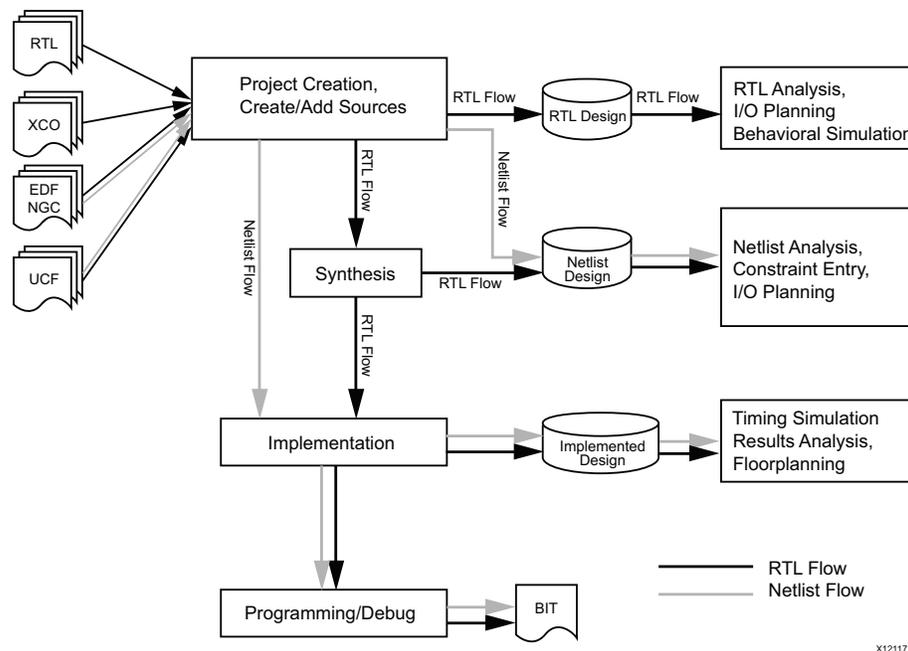
Figure 2-1: **PlanAhead Design Flows**

Project Creation and Management of Project Sources

The PlanAhead tool provides a wizard flow to facilitate project creation and the creation or addition of source files to the project. You can:

- Create projects specific to the type of flow and sources you are using

- Create new source files or add existing source files to the project

- Reference remote, write-protected files, or copy files into the local project folder

- Disable or enable source files within a project

- Create constraint sets to experiment with various constraints options or devices

- Create simulation source sets for behavioral and timing simulation

- Archive projects to create a backup or make a portable copy of the design

The PlanAhead tool project environment lets you create and store multiple variations of the design constraints within a single project. This allows for the creation of multiple RTL source versions, constraints sets, target devices, synthesized netlists, and implementation run results using various implementation strategies. As you modify the source files or launch design tools, the software monitors and displays design flow status.

RTL Development and Analysis

The PlanAhead tool includes an integrated text editor to create or modify source files. You can copy example logic constructs directly from the supplied Xilinx template library. The PlanAhead tool contains a Find in Files feature that lets you search these libraries using a variety of search criteria.

Opening an Elaborated Design elaborates the RTL source files and loads the RTL netlist automatically. The open Elaborated Design lets you check RTL structure, syntax, and logic definitions. Analysis and reporting capabilities include:

- RTL compilation validation and syntax checking
- Netlist and schematic exploration
- Design Rule Checks (DRCs)
- Behavioral simulation
- Resource utilization and power consumption estimation
- Early I/O pin planning is enabled using an RTL port list.

With a design open, selecting an object in one view will cross-select the object in other views, including instantiations and logic definitions within the RTL source files. Refer to Chapter 5, Elaborated RTL Design, for more information.

IP Customization and Implementation

Run the integrated CORE Generator™ tool to browse, customize, instantiate, implement, and automatically update IP.

Logic Synthesis

The PlanAhead tool lets you configure, launch, and monitor synthesis runs using the Xilinx® Synthesis Technology (XST) tool. Refer to Chapter 6, Synthesizing the Design, for more information.

You can experiment with different synthesis options and create reusable *strategies* for Synthesis runs. For example, you could create strategies for power, performance, or area optimization.

The Synthesis run results display interactively, and the PlanAhead tool creates report files that you can access. Select synthesis Warnings and Errors from the Compilation Messages view to highlight the logic in the source files.

- You can launch multiple Synthesis runs simultaneously or serially.
- On a Linux system, you can launch runs locally or on remote servers.

When you have multiple Synthesis runs, those runs create multiple netlists that are stored within the PlanAhead tool project. The PlanAhead tool then lets you load the various versions of the netlist into the environment for analysis. After the netlist import, you can perform device and design analysis, and create constraints for I/O pin planning, floorplanning, and implementation.

The most comprehensive list of Design Rule Checks (DRCs) is available after a synthesized netlist is produced, when clock and clock logic are available for analysis and placement.

I/O Pin Planning

The PlanAhead tool provides an I/O pin planning environment that enables correct "by-construction" I/O port assignment either onto specific device package pins or onto internal die pads. The PlanAhead tool offers a variety of display views and tables in which to analyze and design

package and design I/O related data. Refer to Chapter 8, I/O Pin Planning, for more information.

You can:

- Examine the internal I/O connectivity to ensure proper data flow through the device as well as optimal access to internal device resources.

- Improve system performance by examining both the external and internal connectivity requirements and then making informed decisions.

- Use DRCs and Simultaneous Switching Noise (SSN) analysis to ensure compliance to connectivity requirements.

- Use a variety of source input formats to begin I/O pin planning including: CSV, UCF, RTL, or synthesized netlists.

  When you use a synthesized netlist as the source, the DRC coverage improves substantially because often the clock logic dictates proper I/O assignment. The final I/O verification step is to run the complete design through the implementation tools.

Netlist Analysis and Constraints Definition

The PlanAhead tool has design analysis and constraints assignment capabilities. The design data is presented in different forms using cross-selecting and coordinating views.

The PlanAhead tool provides interactive graphical views of the internal die and external package with which you can analyze device resources and apply constraints. You can also apply and analyze timing and physical constraints.

Early timing analysis, including timing simulation, resource estimation, connectivity analysis, and DRCs help identify design issues prior to Implementation.

Implementation

The PlanAhead tool lets you configure, launch, and monitor implementation *runs* using the ISE® Design Suite.

You can experiment with different implementation options and create reusable *strategies* for implementation runs. As an example, you can create strategies for quick runtimes, performance, or area optimization.

The implementation run results display interactively, and report files are accessible.

Also, you can launch multiple implementation runs either simultaneously or serially: when using the Linux operating system you can use remote servers. You can create *Constraint Sets* so you can experiment with various logical constraints, physical constraints, or alternate devices.

Results Analysis and Floorplanning

Load the various run results interactively for analysis and floorplanning. The capabilities in the Implemented Design are described in:

- Chapter 7, Synthesized Design Constraints and Analysis
- Chapter 11, Analyzing Implementation Results

You can import results from any run that is launched from the PlanAhead tool.

When you open an Implemented Design, the original netlist, constraints, and implementation results are loaded into the Results Viewer. You can open multiple designs simultaneously. You can also launch the ISE Simulator (ISim) for timing simulation, XPower Analyzer, and FPGA Editor tools directly from the PlanAhead tool for further design analysis.

Device Programming

You can create programming bitstream files for any completed implementation run. Bit file generation options are configurable. Launch the iMPACT tool to configure and program the part.

Design Verification and Debug

You can configure and implement ChipScope™ Pro Analyzer tools and IP cores, such as the Integrated Logic Analyzer (ILA) and Integrated Controller (ICON), in the Synthesized Design, and select and configure the required probe signals into the cores. You can launch the ChipScope Analyzer tool on any run that has a completed bitstream file.

You can launch the ChipScope Analyzer tool directly from the PlanAhead tool for further analysis of the routing or device resources.

User Models

The PlanAhead tool provides a Graphical User Interface (GUI) with "layered complexity." It provides an intuitive environment for new or casual users and also provides access to the more advanced features. By default, the PlanAhead tool opens with a push button flow suitable for users who do not require more advanced analysis and floorplanning features. The flow is controlled by a view called the *Flow Navigator*, described in Understanding the Flow Navigator, page 19.

Basic User Flow

The PlanAhead tool lets you execute the entire development cycle using the run buttons in the PlanAhead tool Flow Navigator. You can traverse the FPGA development process front-to-back, beginning with importing source files, synthesizing design logic, implementing synthesized netlists, analyzing the results, generating bitstreams, and launching programming and verification tools.

Advanced User Features

The PlanAhead tool provides a series of analysis environments at each stage of the design flow for advanced design configuration and analysis. You can load the Elaborated Design, Synthesized Design, and Implemented Design for analysis and constraint definition. These environments are described in Working with Designs, page 24.

The PlanAhead tool lets you create and store multiple variations of the design within a single project. This allows for the creation of multiple RTL source versions, constraints sets, target devices, synthesized netlists, and implementation run results using various implementation strategies. As you modify the source files or launch design tools, the software provides a design status.

You can configure, launch, and monitor multiple synthesis and implementation runs locally or on remote Linux servers. You can experiment with different command options, constraints, and devices.

Understanding the Flow Navigator

The Flow Navigator provides control over the major design process tasks, such as project configuration, synthesis, implementation, and bitstream creation. As these tasks are completed, you can open the resulting designs to analyze results and apply constraints by clicking **Open Elaborated Design**, **Open Synthesized Design**, or **Open Implemented Design** in the Flow Navigator. Each of these Designs displays a set of commonly used commands for the applicable phase of the design flow.

Available options depend on the status of the design. Inapplicable steps are greyed out until the appropriate design tasks are completed.

Figure 2-2 and Figure 2-3, page 21 illustrate how the Flow Navigator view is used to perform design tasks and to open the analysis environments at various stages of the design process.

Using the Flow Navigator with an RTL Project

Figure 2-2 illustrates the design flow using RTL sources as input to the PlanAhead tool.

Configure Project Sources

- Create and Manage Source Files
- Browse, Customize, and Implement IP
- Create and Manage Constraint Sets
- Manage Project Settings

Load Elaborated Designs

- View Compilation issues
- Analyze Logic Hierarchy and Connectivity
- Behavioral Simulation
- Early Resource and Power Estimation
- Assign I/Os using RTL Ports
- DRCs and SSN Analysis

Load Synthesized Design

- Analyze Logic Hierarchy and Connectivity
- View Resource Estimates
- Define Constraints (I/O, Timing, Placement)
- Configure and Run Timing Analysis
- DRC and SSN Analysis
- Insert ChipScope Debug Cores

Load Implemented Design

- Analyze Timing and Placement Results
- View Connectivity
- Launch ISim Timing Simulation
- Refine Constraints (Timing, Placement...)

Run XST Synthesis

- Configure and Launch Runs
- Assign Run Strategies
- View Results

Run ISE Implementation

- Configure and Launch Runs
- Assign Run Strategies
- Analyze Results
- Launch FPGA Editor

Launch Programming and Debug

- Run BitGen
- Launch ChipScope Analyzer
- Launch Impact

Figure 2-2: **PlanAhead Software Flow Navigator - RTL Project**

Using the Flow Navigator with a Synthesized Netlist Project

Figure 2-3 illustrates the design flow for synthesized netlist-based projects.
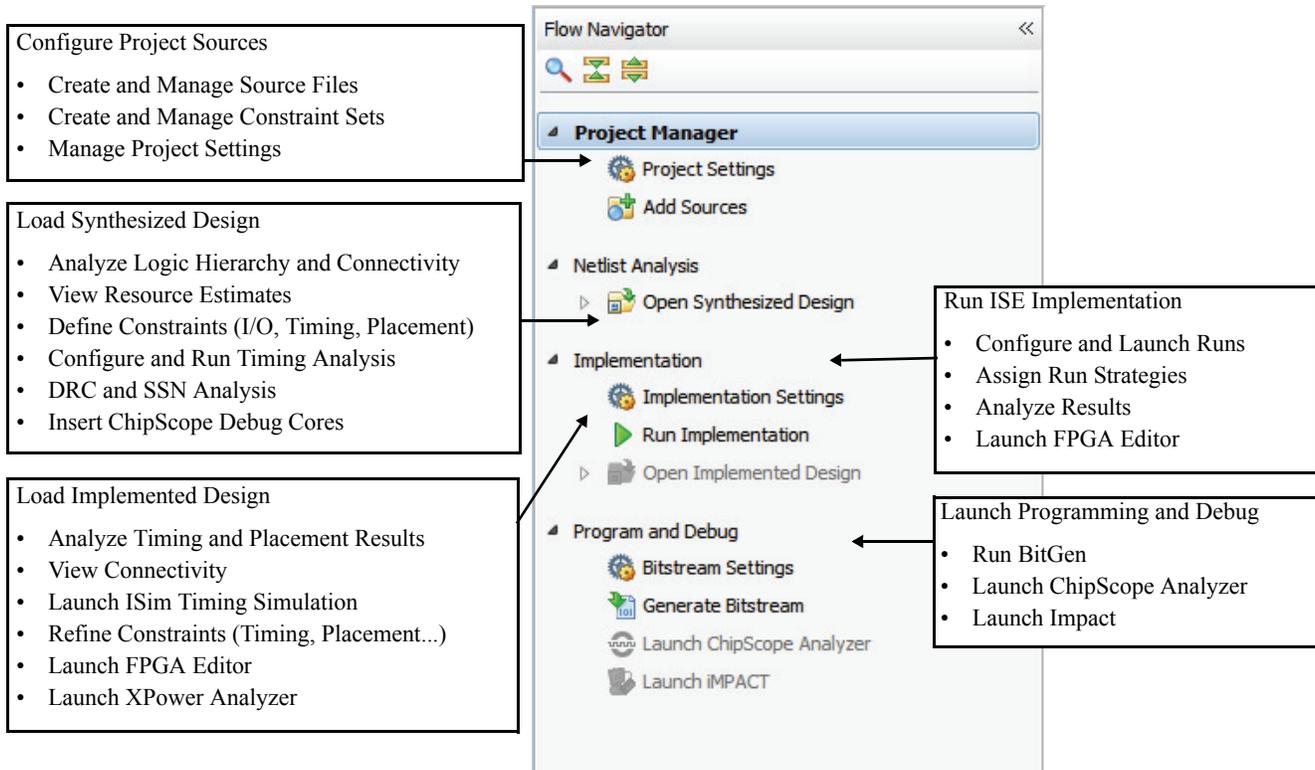
Figure 2-3:   **PlanAhead Software Flow Navigator - Synthesized Netlist Project**

Launching Commands from the Flow Navigator

The Flow Navigator facilitates a push button flow by enabling synthesis and implementation to be run immediately after you add source files to a project. There is no need to open any of the Design environments to complete the design. The following subsections describe how to complete a design using the Flow Navigator.

Project Manager

When you open a project, the Project Manager opens by default. When you open the Project Manager, no design compilation is performed and no design data is loaded into memory. The Project Manager displays the sources, source properties, and a Project Summary view by default. This environment enables creating, importing, and managing source files and constraint sets. You can also use the Project Manager to browse, customize, and create IP from the Xilinx IP Catalog.

The Project Manager menu of the Flow Navigator contains the following commands:

- **Project Manager:** Opens the Project Summary view of the current project. See Using the Project Summary view, page 84 for more information. Click the arrow icon to expand and collapse the menu of Project Manager commands.

- **Project Settings:** Opens the Project Settings dialog box. Refer to Configuring Project Settings, page 88 for more information.

- **Add Sources:** Invokes the Add Sources dialog box. Refer to Managing Project Sources, page 52.

- **IP Catalog:** Opens the IP Catalog view. Refer to Managing IP Cores, page 66.

- **Run Behavioral Simulation:** Launches ISim to perform a behavioral simulation on the Elaborated Design. Refer to Performing Behavioral Simulation, page 188.

RTL Analysis

Open the Elaborated Design to see an un optimized view of the RTL, which is useful for understanding how the tools are interpreting the HDL.The RTL Analysis menu of the Flow Navigator provides the following commands to elaborate and analyze RTL source files:
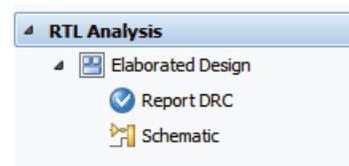
- **RTL Analysis:** Click the arrow icon to expand and collapse the menu of RTL Analysis commands.

- **Elaborated Design/Open Elaborated Design:** The state of this command depends on whether an Elaborated Design is open or not.

  - **Elaborated Design:** Make an open Elaborated Design the active design.

  - **Open Elaborated Design:** If an Elaborated Design is not open, elaborate the RTL source files according to the current top module, with the associated design constraints, and create and open a schematic view. Refer to Elaborating and Analyzing the RTL Source Files, page 184.

- **Report DRC:** Run RTL-based design rule checks on the active Elaborated Design, as explained in Running RTL DRCs, page 195.

- **Schematic**: Create and open a schematic view of the current top module of the Elaborated Design. See Using the Schematic View, page 146.

Synthesis

Open the Synthesized Design to analyze the design for clock and pin-planning, floorplanning, and timing constraint creation.The Synthesis menu of the Flow Navigator provides the following commands to manage and analyze Synthesized Designs:
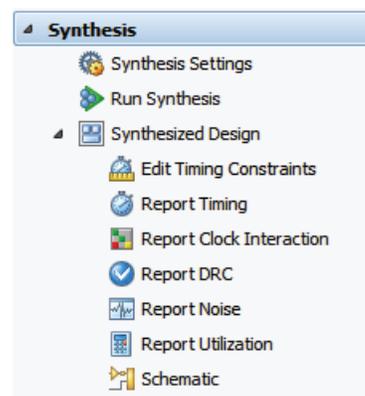
- **Synthesis:** Click the arrow icon to expand and collapse the menu of Synthesis commands.

- **Synthesis Settings:** Open the Project Settings dialog, to modify the Synthesis Settings. See Synthesis Settings, page 92 for more information.

- **Run Synthesis:** Launch the active synthesis run. Refer to Launching a Synthesis Run, page 204.

- **Synthesized Design/Open Synthesized Design:** The state of this command depends on whether a Synthesized Design is open or not.

  - **Synthesized Design:** Switch the tool focus to a currently open Synthesized Design.

  - **Open Synthesized Design:** If a Synthesized Design is not open, the tool reads the synthesized netlist, maps it to the target part, and applies the specified constraints file. Refer to Using the Synthesized Design Environment, page 209.

- **Edit Timing Constraints:** Open the Timing Constraints view to display, edit, and create timing constraints for the Synthesized Design. See Defining Timing Constraints, page 219 for more information.

- **Report Timing:** Report timing analysis of the Synthesized Design to ensure that paths are covered by needed constraints. See Running Timing Analysis, page 222.

- **Report Clock Interaction:** The Clock Interaction Report offers information on clock interactions and signals that cross clock domains. See Analyzing Clock Interactions, page 233.

- **Report DRC:** Run design rule checks on the current Synthesized Design, as explained in Running the Design Rule Checker (DRC), page 242.

- **Report Noise:** Run SSO/SSN analysis as appropriate for the target part. Refer to Using Noise Analysis Predictors, page 278 for more information.

- **Report Utilization:** Report the utilization statistics for the Synthesized Design in the context of the target part. Refer to Viewing and Reporting Resource Statistics, page 211 for more information.

- **Schematic**: Create and open a schematic view of the current top module of the Synthesized Design. See Using the Schematic View, page 146.

Implementation

After Synthesis has completed, you can run the implementation tools in the Flow Navigator to place and route the design. The Implementation menu of the Flow Navigator provides the following commands to manage and analyze Implemented Designs:

- **Implementation:** Click the arrow icon to expand and collapse the menu of Implementation commands.

- **Implementation Settings:** Open the Project Settings dialog, to modify the Implementation Settings. See Implementation Settings, page 93 for more information.

- **Run Implementation:** Launch the active implementation run. If you have not yet run synthesis, you can still use **Run Implementation** and the tool completes the parent synthesis run as a prerequisite to the active implementation run. See Chapter 9, Implementing the Design, for more information.

- **Implemented Design/Open Implemented Design:** The state of this command depends on whether an Implemented Design is open or not.

  - **Implemented Design:** Switch the tool focus to a currently open Implemented Design.

  - **Open Implemented Design:** If an Implemented Design is not open, the tool reads the synthesized netlist, processes the design constraints, reads the placement and routing results, and loads the timing results to open the Implemented Design in memory. Refer to Opening the Implemented Design, page 335.

- **Edit Timing Constraints:** Open the Timing Constraints view to display, edit, and create timing constraints for the Implemented Design. See Defining Timing Constraints, page 219 for more information.

- **Run TRCE:** The TRCE software verifies that the design meets timing constraints, and generates a report file that lists compliance of the design against the input constraints. Refer to Analyzing Timing Results, page 338.
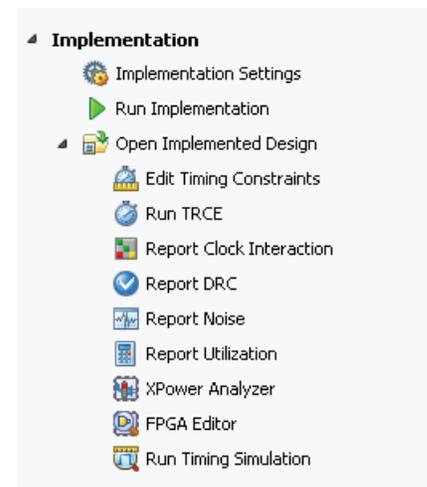
- **Report Clock Interaction:** The Clock Interaction Report offers information on clock interactions and signals that cross clock domains. See Analyzing Clock Interactions, page 233.

- **Report DRC:** Run design rule checks on the current Implemented Design, as explained in Running the Design Rule Checker (DRC), page 242.

- **Report Noise:** Run SSO/SSN analysis as appropriate for the target part. Refer to Using Noise Analysis Predictors, page 278 for more information.

- **Report Utilization:** Report the utilization statistics for the Synthesized Design in the context of the target part. Refer to Viewing and Reporting Resource Statistics, page 211 for more information.

- **XPower Analyzer:** Run Power Analysis on the Implemented Design. See Analyzing Power Distribution with XPower Analyzer, page 362.

- **FPGA Editor:** Open FPGA Editor on the Implemented Design. See Launching FPGA Editor, page 363.

- **Run Timing Simulation:** Perform timing simulation on the Implemented Design. See Performing Timing Simulation, page 354.

Program and Debug

After Implementation has completed, you can generate bitstream files and launch the debugging and programming tools. The Program and Debug menu in the Flow Navigator provides the following commands:
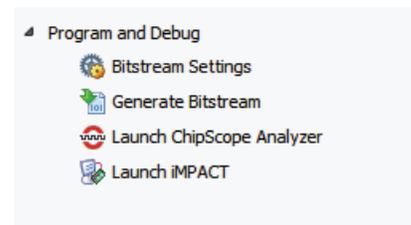
- **Program and Debug:** Click the arrow icon to expand and collapse the menu of commands under the Program and Debug heading.

- **Bitstream Settings**: Open the Project Settings dialog, to modify the Bitstream Settings. See Bitstream Settings, page 94 for more information.

- **Generate Bitstream**: Run the BitGen command with the options specified by the Bitstream Settings dialog box.

- **Launch ChipScope Analyzer:** Use for launching the ChipScope Pro Analyzer product on Implemented Designs that have implemented ChipScope ILA cores. See Launching ChipScope Pro Analyzer, page 375 for more information.

- **Launch iMPACT**: Launch iMPACT on the Implemented Design. See Launching iMPACT, page 375 for more information.

The ChipScope Pro Analyzer and iMPACT require a BIT file, and are available only after you run the **Generate Bitstream** command. Refer to Chapter 12, Programming and Debugging the Design, for more information.

Working with Designs

The PlanAhead tool lets you open designs at various stages of the design process. A design is defined as a netlist (elaborated RTL or synthesized), a constraint set, and a target device. The PlanAhead tool loads the design into memory for analysis, constraint definition, or ChipScope debug core insertion.

The Flow Navigator has buttons for Elaborated Design, Synthesized Design, and Implemented Design. The Implemented Design loads the specific design data used to launch the run. You can open the Elaborated and Synthesized Designs with different target devices and/or constraint sets allowing experimentation and constraints version control.

You can analyze the design at various stages of completion, including elaborated RTL, synthesized netlist, or implemented results. You can make design changes by editing constraints such as timing or floorplanning and placing logic at any phase by opening the appropriate design.

Constraint change management is made possible by the ability to create and manage multiple constraints files. The available levels of design abstraction that you can open in the PlanAhead tool environment are:

- Elaborated Design: Elaborated RTL source files, constraint set, and target device.
- Synthesized Design: Synthesized netlist, constraints, and target device.
- Implemented Design: Synthesized netlist, constraints, and results from any implemented run.

For all design views, the default view layout in the GUI includes a graphical view of the design, such as the Schematic or Device view, the Netlist view, the Properties view, and the Messages and Reports views at the bottom of the workspace. Select a different view layout in the menu toolbar to toggle between different arrangements of the various views. See Using View Layouts, page 102.

Opening an Elaborated Design

An elaborated RTL design consists of multiple RTL source files, for synthesis and simulation, can include synthesized netlists for modules and IP cores to be used in the design, constraints files to hold timing and physical constraints, and a Xilinx part which is the target device for the design.

When you click **Open Elaborated Design** in the Flow Navigator, the PlanAhead tool automatically elaborates the RTL source files and loads the elaborated netlist into memory along with the active constraint set and the target part. Elaboration messages display in the Messages view.

To open a Elaborated Design, select either:

- The **Open Elaborated Design** command in the RTL Analysis menu of the Flow Navigator to load the elaborated netlist, the active constraint set, and the target device into memory.
- **Flow > Open Elaborated Design** from the main menu.
- **New Elaborated Design** accessed from the RTL Analysis popup menu in the Flow Navigator.
- **Flow > New Elaborated Design** from the main menu.

You can specify the name for a new Elaborated Design, the constraint set to apply, and the target part to load into the design, as shown in Figure 2-4, page 25.

Note: If you select a constraint set that is not the active constraint set, you can also choose to make it active. See Managing Constraints, page 53 for more information.

Figure 2-4: **New Elaborated Design**

When you click OK, the RTL design is elaborated to create an expanded netlist for the design. This is not a fully synthesized netlist, but permits some analysis of the design beyond what is possible from the RTL files.

Moving Processes to the Background

As the PlanAhead tool spawns the process to perform elaboration, reading the various design files and constraint files, the Open Elaborated Design dialog box opens, to let you put the elaboration process into the background, as shown in Figure 2-5.

When you put a process into the background, it releases the PlanAhead tool to perform certain other functions, like viewing reports, or opening design files, while it completes the background task. You can make use of this time to review reports for instance. However, the Tcl Console is blocked, and you will not be able to use Tcl commands, or perform tasks that require Tcl commands, such as switching to another open design.

Figure 2-5: **Open Elaborated Design - Background Process**

The Elaborated Design view is opened with the default view layout, including the RTL Netlist view and Schematic view of the top-level schematic, as shown below.

The RTL Netlist view displays the elaborated logic hierarchy. See Chapter 5, Elaborated RTL Design for more information on analyzing the RTL logic design.
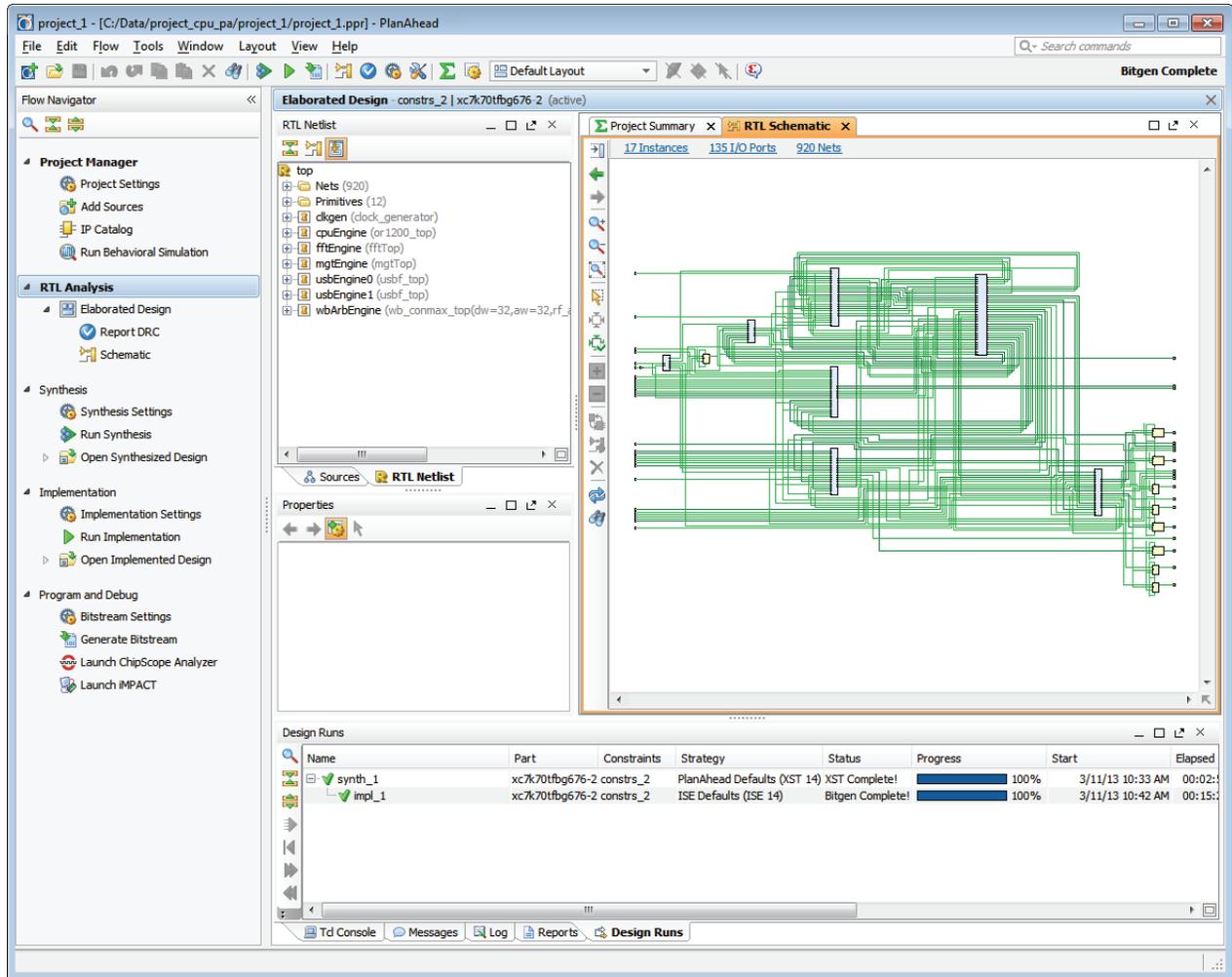
Figure 2-6: **RTL Analysis view**

Setting the Active Run

When opening a Synthesized design, or an Implemented Design, the PlanAhead tool default is to apply the netlist and the placement and routing data from the currently active run. The Flow Navigator menu is sensitive to the active run, with commands like Open Synthesized Design and Open Implemented Design dependant upon whether the active run has completed synthesis or implementation. The Project Summary, Log view, and Messages view also display information about the active run.

If multiple synthesis or implementation runs exist, the PlanAhead tool applies the netlist and design information from the currently *active run*. The active run displays in bold text in the Design Runs view, as shown in Figure 2-7, page 28.

To change the active run, select the **Synthesis Run** or **Implementation Run** in the Design Runs view, then use the **Make active** command in the popup menu. The PlanAhead tool then uses the netlist or placement and routing data of the new active run when opening designs or launching runs. See Using the Design Runs View, page 165 for more information.

Figure 2-7: **Design Runs view**

For more information on creating and managing multiple runs, see Defining Implementation Runs, page 287.

Opening a Synthesized Design

The PlanAhead tool creates a Synthesized Design using combinations of a synthesized netlist, physical and timing constraints, and a target part.

When it is opened, the different elements of the Synthesized Design are loaded into memory, where you can analyze them and modify design elements as needed to complete the design. You can save the Synthesized Design, updating the constraints file, or reload the design without saving to restore the original design.

To open a Synthesized Design, select either:

- Double-click on the run name in the Design Runs view.

- **Open Synthesized Design** command in the Synthesis menu of the Flow Navigator.

- **Flow > Open Synthesized Design** command from the main menu.

- **New Synthesized Design** from the popup menu of the Synthesis command of the Flow Navigator.

- **Flow > New Synthesized Design** command from the main menu.

  You can open the synthesized netlist with the active constraint set, and the target device, or specify an alternative constraint set and target device to open the Synthesized Design in memory. As shown in Figure 2-8, page 29, you can enter the following to define a new Synthesized Design:

  - **Design Name:** Enter a name to display in the view banner. The design is stored in memory during the session only.

  - **Constraint Set:** Select an existing constraint set to be opened against the netlist.

  - **Part:** Select a target part.
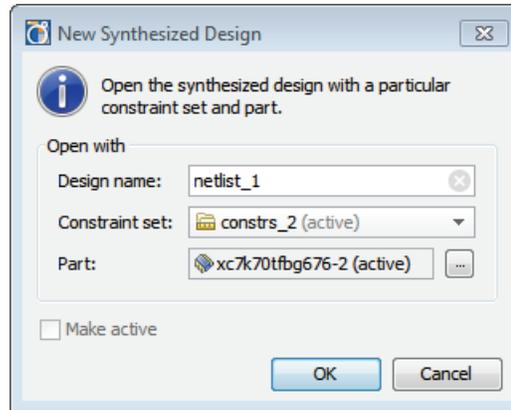
Figure 2-8: **New Synthesized Design**

Figure 2-9 shows the default view layout of the open Synthesized Design.
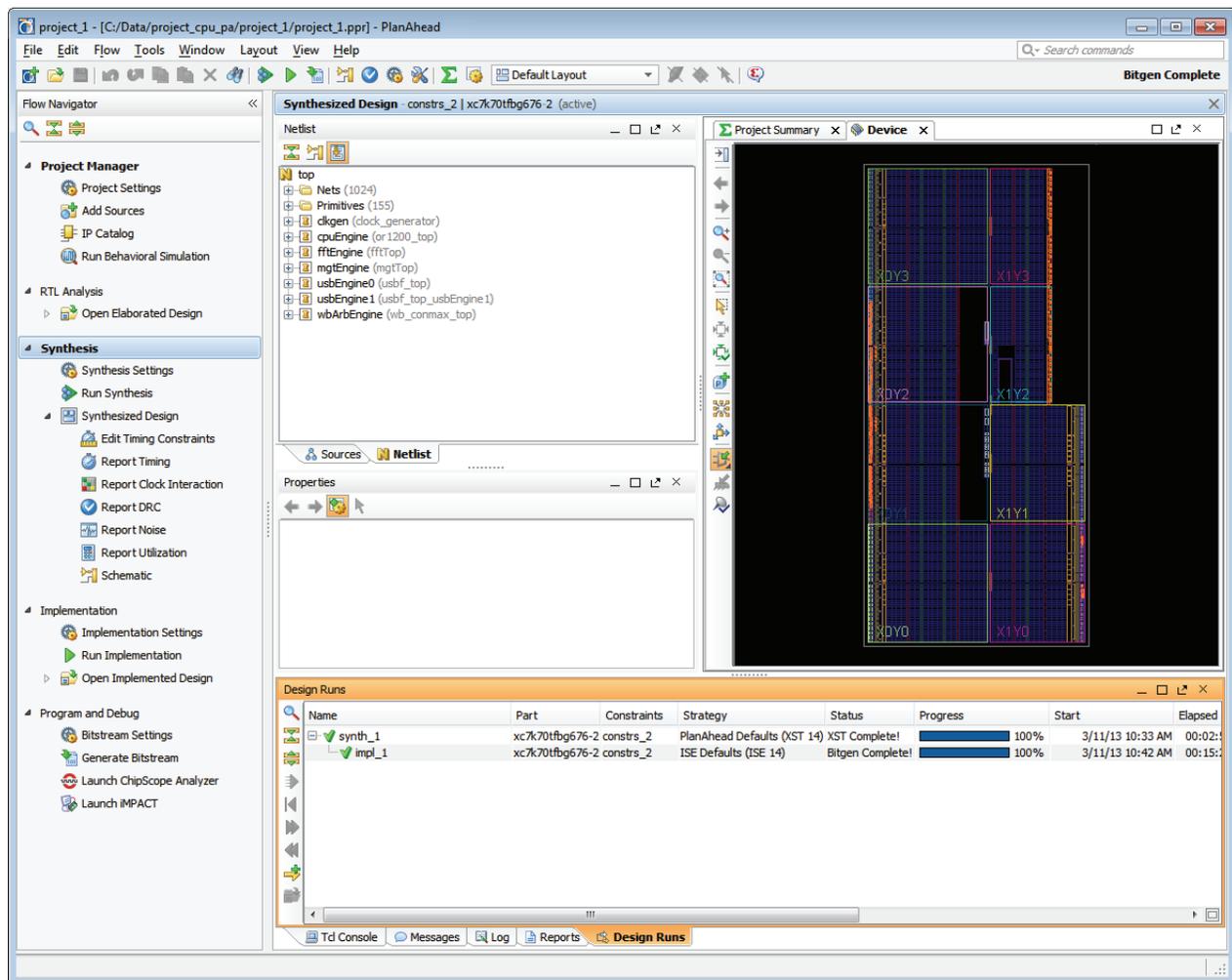
Figure 2-9: **Open Synthesized Design**

Moving Processes to the Background

As the PlanAhead tool spawns the process to open the Synthesized or Implemented Designs, reading design files and constraint files, the Open Synthesized Design dialog box or Open Implemented Design dialog box lets you put the process into the background, as shown in Figure 2-10.

When you put a process into the background, it releases the PlanAhead tool to perform certain other functions, like viewing reports, or opening design files, while it completes the background task. You can make use of this time to review previous runs for instance, or examine reports. However, the Tcl Console is blocked, and you will not be able to use Tcl commands, or perform tasks that require Tcl commands, such as switching to another open design.
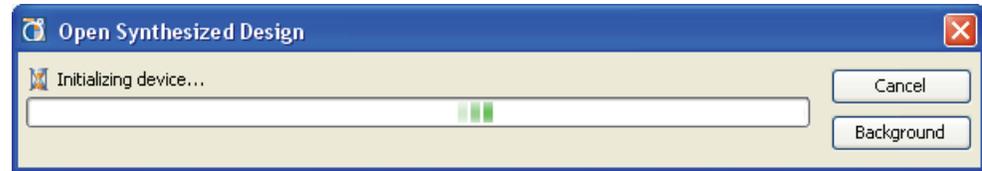
Figure 2-10: **Open Synthesized Design - Background Process**

Opening an Implemented Design

An Implemented Design consists of a synthesized netlist, physical and timing constraints, the target XIlinx part, and placement and routing data from a completed implementation run. Because the PlanAhead tool allows for multiple implementation runs, you can select any completed implementation run to open in the Implemented Design.

The Implemented Design imports the netlist, constraints, placement, and timing results from the implementation run directory. The PlanAhead tool loads the Implemented Design into memory where you can analyze it and make any needed design changes.

To open an Implemented Design, select one of the following methods:

• Double-click on the run name in the Design Runs view.

• **Flow > Open Implemented Design** in the main menu.

• **Open Implemented Design** from the Implementation menu in the Flow Navigator.

Since the Flow Navigator reflects the state of the active run, the Open Implemented Design command may be disabled, or greyed out, if the active run is not implemented. In this case, the Implementation popup menu in the Flow Navigator will allow you to open an Implemented Design from any completed implementation runs.

The Implemented Design default view opens. Typically, you do placement and timing analysis and floorplanning in this view. Figure 2-11, page 31 shows the default layout view of an open Implemented Design.
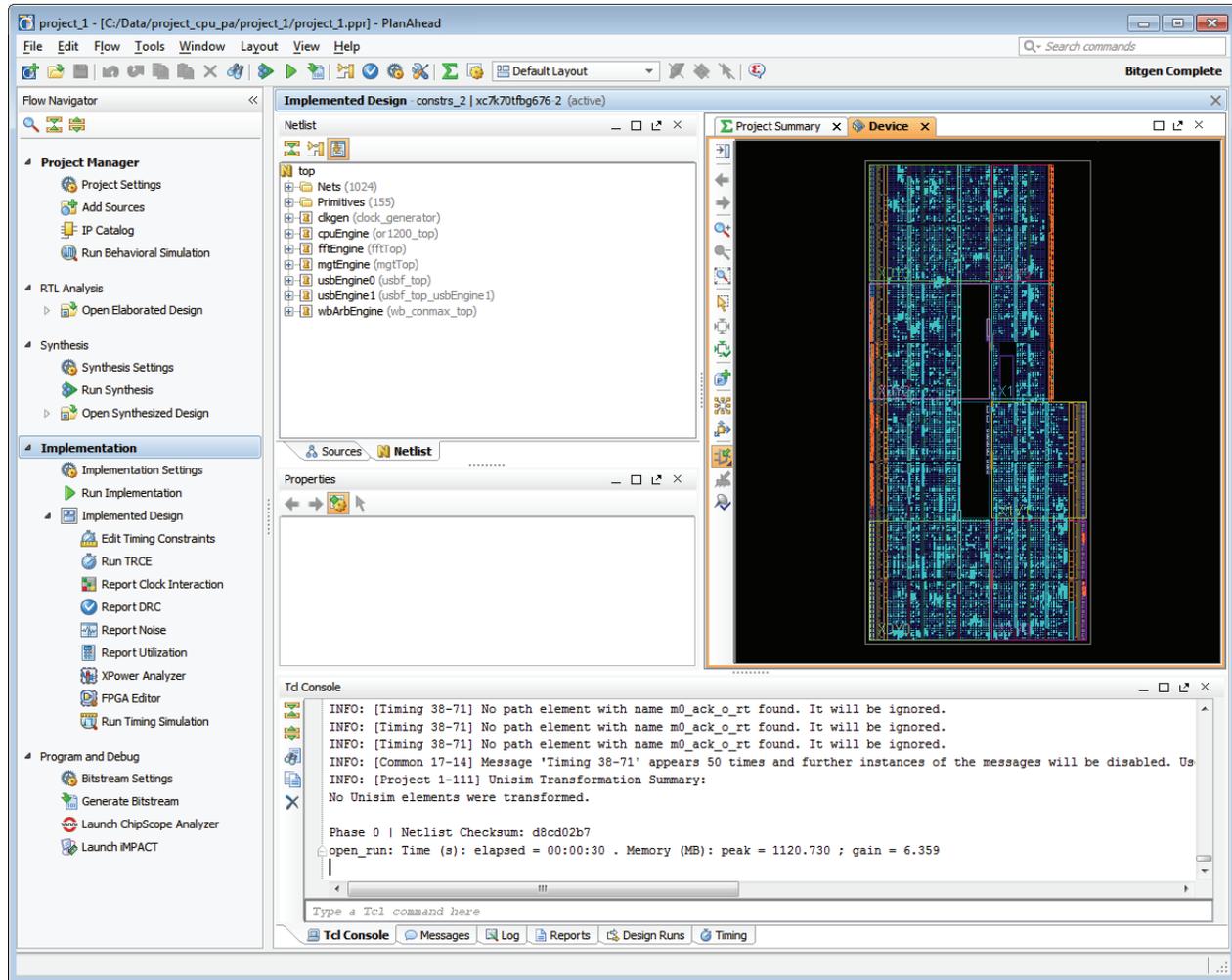
Figure 2-11: **Open Implemented Design**

Managing Open Designs

As you open designs and the PlanAhead tool loads the design into memory, the associated menu in the Flow Navigator is highlighted. This provides you with a visual reference of the current design. Figure 2-12, page 32 shows the highlighted RTL Analysis menu. This indicates that an Elaborated Design is both opened and active in the current session.

You can open multiple Elaborated Designs, Synthesized Designs, and Implemented Designs simultaneously to display the results of different design options and different runs.

When multiple designs are open, the Flow Navigator provides multiple indications of the open designs. In Figure 2-12 the Synthesized Design command under the Synthesis menu reflects the fact that a Synthesized Design is open, but is not the active design. Selecting the **Synthesized Design** command will make the open design the active design, and will move the highlighting from RTL Analysis to Synthesis. The grey Open Implemented Design command under the Implementation menu indicates that implementation has not been completed, and there is no available Implemented Design for the active implementation run.
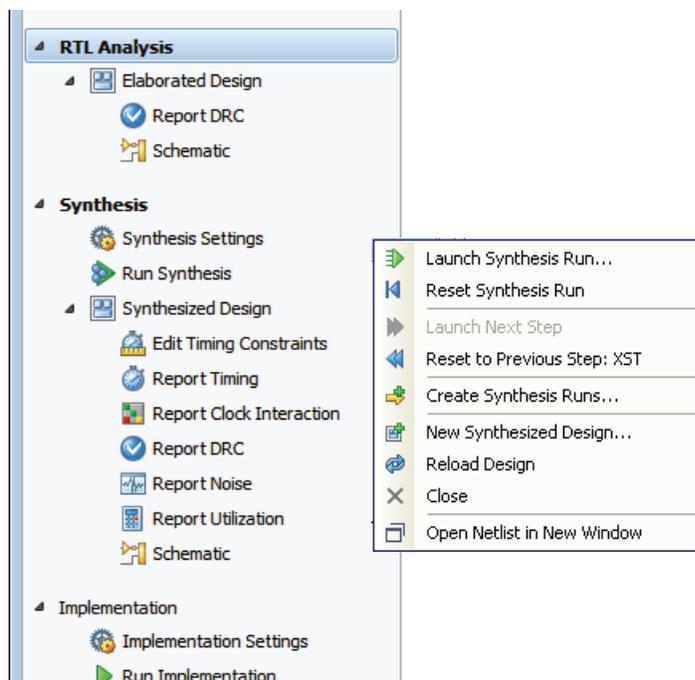
Figure 2-12: **Open Designs in Flow Navigator**

The Flow Navigator also supports a popup menu accessed from the right mouse button (RMB). Click the RMB over the Project Manager, RTL Analysis, Synthesis, or Implementation commands in the Flow Navigator, and a popup menu is displayed. Figure 2-12 shows the Synthesis popup menu.

The **Open Netlist in New Window** lets you open a new PlanAhead window with the Synthesized Design, while maintaining the current window with the currently open design. You can use this command to view the Elaborated Design in one window and open the Synthesized Design or Implemented Design in a second window. You can zoom and pan, and select different elements of the design, analyzing and developing both design views.

Using the Design View Banner

The Design View banner reflects the content of the current design, displaying the constraint set, and target part, as well as the synthesis or implementation run as appropriate. Figure 2-13, page 32 shows the Design View banners for each of the different design types.

Figure 2-13: **Design View Banners**

If multiple designs are open, tabs display in the Design View banner to let you toggle between the open designs, as shown in Figure 2-14.
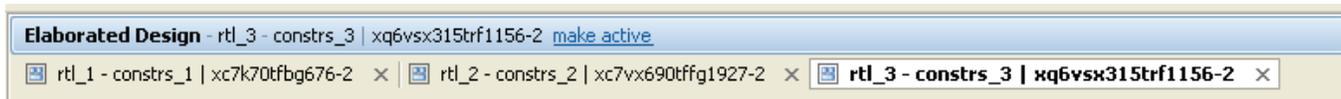
Figure 2-14: **Multiple Open Design Tabs**

Updating and Reloading Designs

In the course of any design process, source files or constraints often require modification. The PlanAhead tool manages the dependencies of these files, and indicates when the design data in the current design is out-of-date. Changing project settings, such as the target part or active constraint set, can also make a design out-of-date.

When the target part or constraint set of the open design are different from the constraint set or target part of the active synthesis or implementation runs, the Design View banner displays the **make active** link, as shown in Figure 2-14. Clicking on the **make active** link sets the constraint set and target part of the current design as the active constraint set and target part for the current project, and the active synthesis and implementation runs.

As source files, netlists, or implementation results are updated, an "out-of-date" message is displayed at the right side of the Design View banner of an open Synthesized Design or Implemented Design to indicate that the run is out of date, as shown in Figure 2-15. Click the **more info** link to display what aspects of the design are out of date.
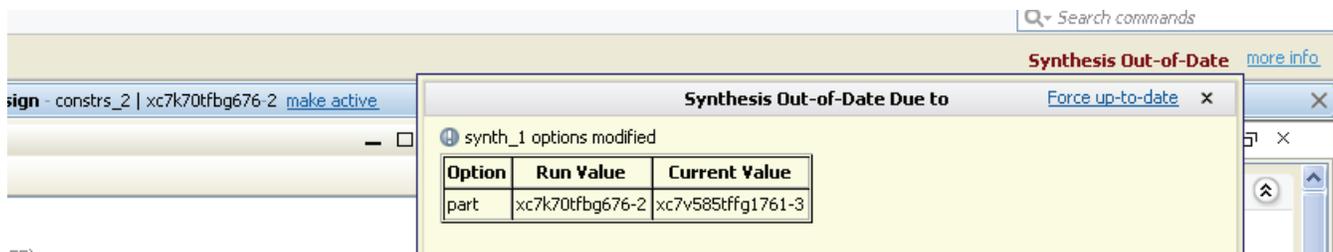
Figure 2-15: **Design Out-of-Date and Reload Banner**

There are three actions you can take to resolve an out-of-date design:

- **Force up-to-date**: Select the **Force up-to-date** link in the "Out-of-Date Due to" window that is opened when you click **more info** as shown in Figure 2-15.

  Force up-to-date will reset the NEEDS_REFRESH property on the active synthesis or implementation runs as needed to force them into an up-to-date state. The Tcl command for this is shown in the following sample code:

  ```
  set_property needs_refresh false [get_runs synth_2]
  ```

  Note: Use this command to force designs up-to-date when a minor design change was made, and you do not want to refresh the design because of it.

- **Reset and Rerun:** Select the **Reset Runs** command from the Design Runs view, or from the popup menu of the Synthesis or Implementation commands in the Flow Navigator. This will reset the run to allow it to be rerun and updated with the current project settings.

  You can also use the **Launch Runs** command directly. The tool will warn you that the run needs to be reset first, and prompt you to confirm before proceeding.

- **Reload the Design**: Use the **Reload Design** command, in the Flow Navigator popup menu as shown in Figure 2-16, to revert any changes and reload the original netlist, constraints, and target part of the design.

This Reload Design command will refresh the in-memory view of the current design, eliminating any changes that you made to the design data that have not been saved.
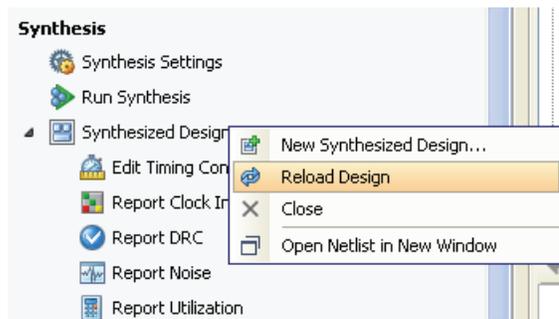
Figure 2-16: **Reload Design**

Saving Designs

To save any changes you have made to your design data, click **File > Save Constraints**.

The **Save Constraints** command saves any changes to the constraint files, design partitions, ChipScope cores and configuration added to the Synthesized Design, and project settings.

When an Implemented Design is open, **Save Constraints** saves any changed constraints to the constraint file associated with the implementation run, and used during implementation, rather than saving to the currently active constraint file. This ensures that the changes to the Implemented Design are saved with the proper constraint file for the implementation run.

However, this can cause unintended changes to the constraint files used for a specific implementation run, and may not be the intended result. The PlanAhead tool warns of this situation, and lets you choose the constraint file to target changes to before writing changes to disk.

You can also save changes to the design in a new constraints file, to preserve your original design, while still saving any changes. Use the **File > Save Constraints As** command to create a new constraint file, while preserving the original design.

Closing Designs

You can close designs to reduce the number of designs in memory and to prevent multiple locations where sources could be edited. In some cases, you are prompted to close a design prior to changing to another design representation. In some cases, such as for a Partial Reconfiguration design, you must close the design when leaving the design. You can close:

- Individual designs by clicking **Close** in the banner of the main viewing area.

- All designs by selecting the **Close All** command from the popup menu in the Flow Navigator.

Chapter 3

Working with Projects

This chapter discusses working with projects, including:

- Project Types
- Creating a New Project
- Managing Projects
- Managing Project Sources
- Managing Design Source Files
- Managing Constraints
- Managing Simulation Sources
- Managing IP Cores
- Managing DSP Sources
- Managing Embedded Processor Sources
- Using the Project Summary view
- Configuring Project Settings

Project Types

You can use the PlanAhead™ application at different points in the FGPA design flow. To accommodate this, you can create the following types of projects:

- Register Transfer Level (RTL) source-based
- Synthesized netlist-based
- Implemented design results-based
- I/O pin planning projects - Comma Separated Values (CSV), User Constraint File (UCF)-based
- Projects created in ISE® Project Navigator
- Partial Reconfiguration projects (with an enabled license)

These projects are differentiated by the input source types used to create the project. You can select the type of project during the Create New Project process.

With the exception of an I/O Pin Planning project, a project cannot be changed to a different project type after it has been created. The I/O Pin Planning project can be used as the foundation of an RTL-based design project, and migrated at any time.

RTL Source-Based Projects

You can use the PlanAhead tool to manage the entire FPGA design flow from RTL creation through bitstream generation. You can add RTL source files, EDIF netlists for blocks of the design, as well as IP generated by the CORE Generator™ tool and precompiled NGC/NGO-format IP netlists to the project.

You can elaborate and analyze the RTL to ensure proper constructs, launch, and manage various synthesis and implementation runs, and analyze the design and run results. You can also experiment with different constraints or implementation strategies.

Synthesized Netlist-Based Projects

You can create projects from designs that were synthesized outside of the PlanAhead tool using the Xilinx® Synthesis Technology (XST) tool or any supported third party synthesis tool. The PlanAhead tool can import EDIF and NGC/NGO-format netlists. The netlist can be all-inclusive in a single file or hierarchical in nature, consisting of multiple, module-level netlists.

You can analyze the logic netlist, launch and manage various Implementation runs, and analyze the design and run results. You can also experiment with different constraints or implementation strategies.

Implemented Design Results-Based Projects

You can create projects to allow analysis of implementation results created outside of the PlanAhead tool using the Xilinx command line tools. You can import a design netlist, implementation, and timing results to explore timing or placement related issues in the PlanAhead tool.

I/O Pin Planning Projects

You can perform I/O pin planning early in the design cycle by creating an empty I/O Pin Planning project. You can create I/O ports within the PlanAhead tool or import them with either CSV or User-Constraint Files (UCF) input files. You can also create pin planning projects to explore the logic resources available in the different device architectures.

After I/O pin assignment, the PlanAhead tool can create CSV, UCF, and RTL output files for use later in the design flow when RTL sources or netlists are available. The output files can also be used to create schematic symbols for use in the Printed Circuit Board (PCB) design process.

An I/O Pin Planning project can be used as the foundation of an RTL-based design project. See Migrating to an RTL Design, page 275 for more information.

Externally Created Projects

You can import project data from ISE Project Navigator, from XST, or from Synopsys® Synplify® to migrate an RTL-level project into PlanAhead. The PlanAhead tool uses the various project settings from the source project when creating this new project.

Note: You can also launch the PlanAhead tool from within Project Navigator for performing I/O Pin Planning and Floorplanning of ISE designs. For information about using PlanAhead tools from the ISE Project Navigator environment, refer to Chapter 15, Using PlanAhead With Project Navigator.

Creating a New Project

The New Project wizard takes you through the individual steps to define a project name and location, add source files and constraint files to the project, and select a target device.

To create a new project:

1. Select one of the following:

   - On the Getting Started page, click **Create a New Project**.
   - Select **File > New Project** or the **New Project** toolbar button.

   The first dialog box of the New Project wizard gives an overview of the wizard.

2. Click **Next** to continue.

   The Project Name page opens, as shown in Figure 3-1.
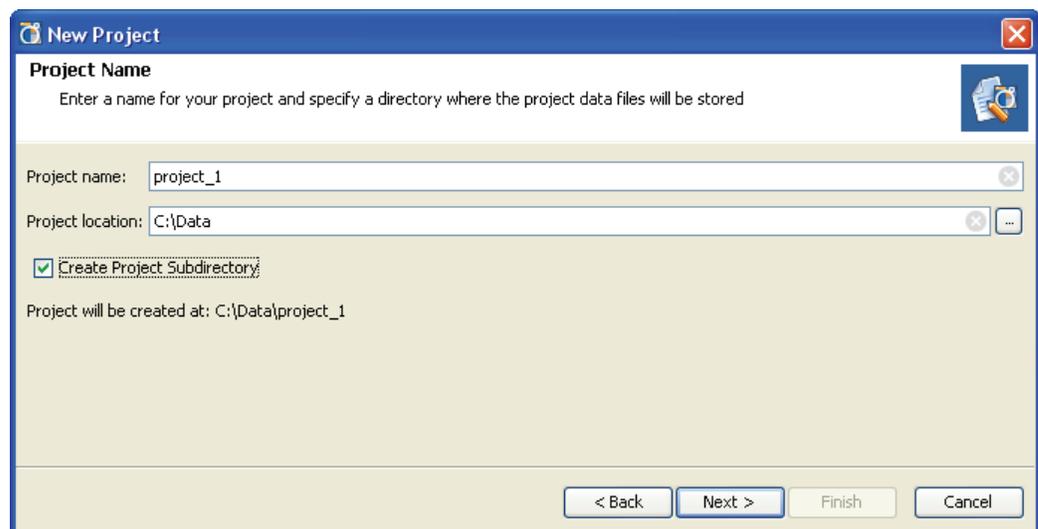
Figure 3-1: **Create Project**

3. In the Project Name page, specify a project name and disk storage location:

   - **Project name:** Enter a name to identify the project directory (for example, `project_3`).
   - **Project location:** Enter a location to create the new project directory.
   - **Create Project Subdirectory:** Use this checkbox to indicate whether the PlanAhead tool should add a subdirectory of the same name as the project within the specified project location.

     If you enable this checkbox, which is the default, the project file (`.ppr`) is created at *<project_location>/<project_name>*, and all folders and data files created for the project are stored in the *<project_name>* subdirectory.

     If you disable this checkbox, the project file (`.ppr`) is created at *<project_location>*, and all folders and data files created for the project are stored in that project location.

4. Click **Next**.

Selecting the Project Type

Designate the Project Type by selecting one of the options shown in Figure 3-2, page 38. This selection determines the type of source files that can be associated with the project.
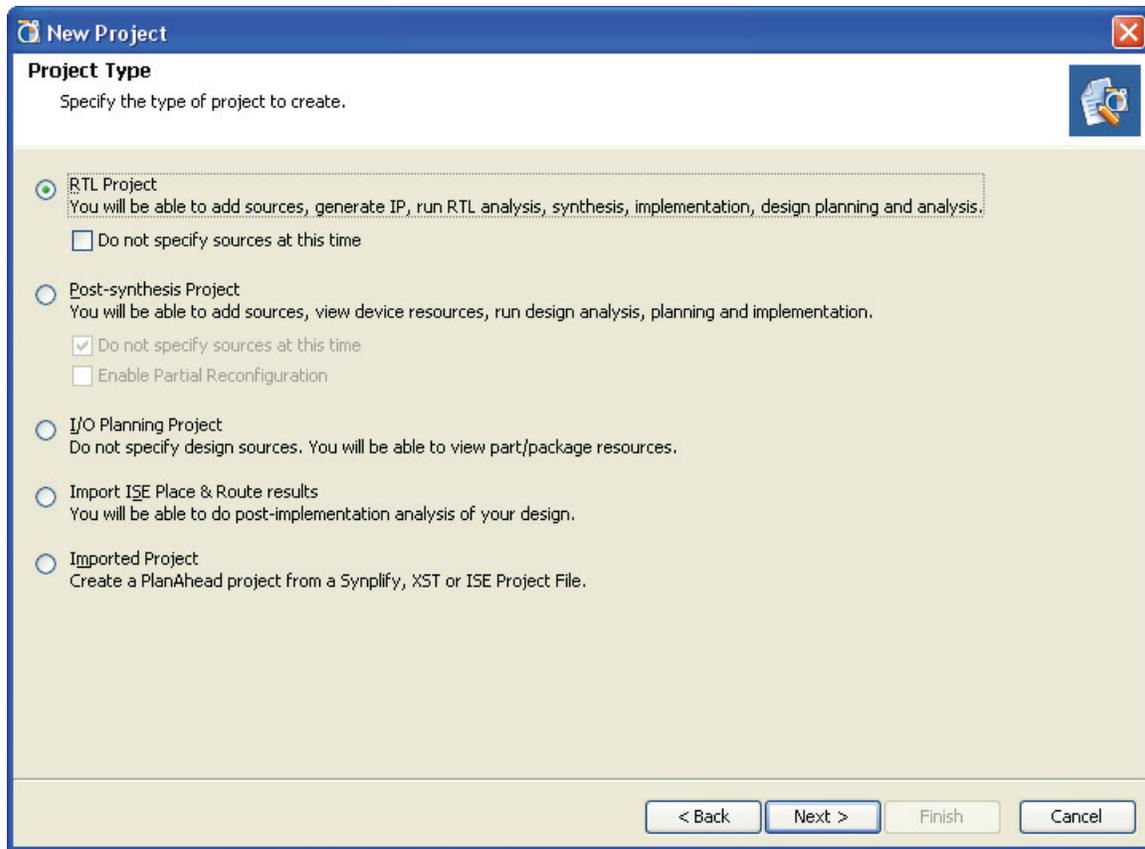
Figure 3-2: **New Project Wizard: Project Type**

5. Select the design source, and click **Next**.

6. Depending upon the type of project you are creating, continue with the instructions in one of the following sections:

    • Creating an RTL Project

    • Creating a Post-synthesis Project

    • Creating an I/O Planning Project

    • Creating a Project with ISE Placement and Timing Results

    • Importing an External Project

The next pages of the wizard guide you through adding appropriate sources to the project based on the project type you selected in the previous steps.

Creating an RTL Project

You can specify RTL source files to create a project. This can be used for RTL code development and analysis purposes as well as synthesis and implementation. See Chapter 5, Elaborated RTL Design for more information on RTL development and analysis.

1. Follow the project creation steps previously described in Creating a New Project, page 37.

2. In the Project Type page shown in Figure 3-2, select the **RTL Project** option.

a. If desired, select the **Do not specify sources at this time** checkbox, and go to Selecting a Default Part or Board, page 45. This will skip the following steps of adding sources files and constraints, and allow you to select the target part and create the project.

3. Click **Next**.

Adding Source Files or Directories

The Add Sources page in the New Project wizard lets you add source files and add directories that contain source files. Use the **Add Files** and **Add Directories** commands as shown in Figure 3-3.
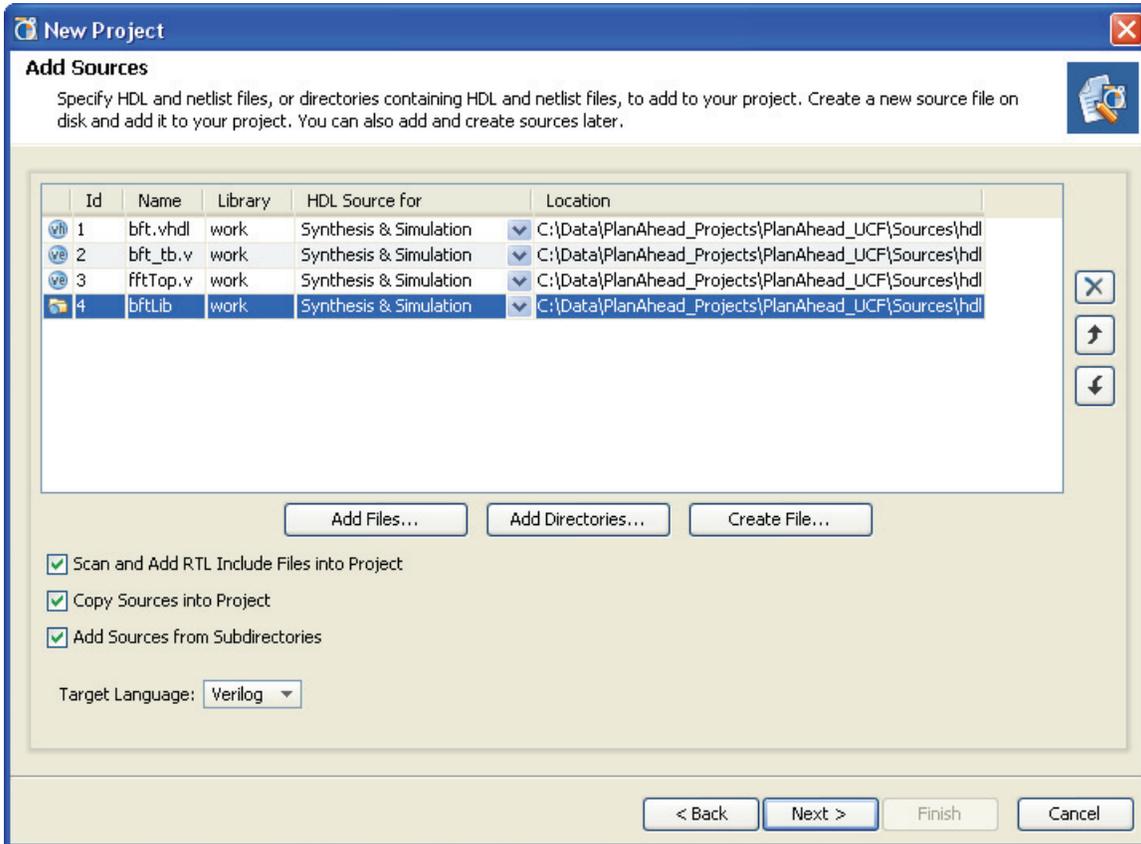
Figure 3-3: **New Project Wizard: Add Sources**

The available commands and options of the Add Sources dialog box are:

- **Add Files**: Invokes a file browser so you can select files to add to the project. The file types that can be added to an RTL project include HDL files, as well as EDIF and NGC files, BMM, ELF, and others.

  Note: Each file or directory is represented by an icon indicating it as a file or folder. A small red square indicates it is read only.

- **Add Directories**: Invokes a directory browser to add all source files from the selected directories. Files in the specified directory with valid source file extensions are added to the project.

  - **Library**: Specify the RTL library for a file or directory by selecting one from the currently defined library names, or specify a new library name by typing in the Library text field.

    The work library is the default library into which all HDL source files are placed. You can create or reference additional user VHDL libraries as needed.

- • **HDL Source for**: Specify if the source being loaded is an RTL source file for synthesis and simulation, or an RTL test bench for simulation only.

- • **Create File:** Invokes the **Create Source File** dialog box where you can create new VHDL, Verilog, or Verilog header files. See Defining New Modules, page 41 for more information.

- • **Remove Selected Files and Directories:** The 'X' icon removes the selected source files and directories from the list of files to be added to the project.

- • **Move Selected Files and Directories Up**: The upward pointing arrow icon moves the file or directory up in the list order. The order of the files affects the order of elaboration and compilation during downstream processes such as synthesis and simulation. See Controlling File Compilation Order in Chapter 6 for more information.

- • **Move Selected Files and Directories Down**: The downward pointing arrow icon moves the file or directory down in the list order.

- • **Scan and Add RTL Include Files into Project:** Scans all RTL source files and imports any referenced Verilog `include files into the local project directory structure.

- • **Copy Sources into Project:** Copies the original source files into the project, and uses the local copied version of the file in the project. If you added directories of source files using **Add Directories**, the directory structure is maintained when the files are copied locally into the project. For a complete discussion of this topic refer to Using Remote Sources or Copying Sources into Project, page 63.

- • **Add Sources from Subdirectories**: Adds source files from the subdirectories of directories specified with **Add Directories**.

- • **Target Language**: Specifies the target language for the design as either Verilog or VHDL. New RTL files will default to the specified target language. Output files will be generated from the design in the specified target language.

Creating RTL Sources

To create a new source file, select the **Add or Create Design Sources** option from the **Add Sources** command, and click **Create File**. The Create Source File dialog opens, in which you can define the type, name, and location of the new source file you are creating, as shown in Figure 3-4.
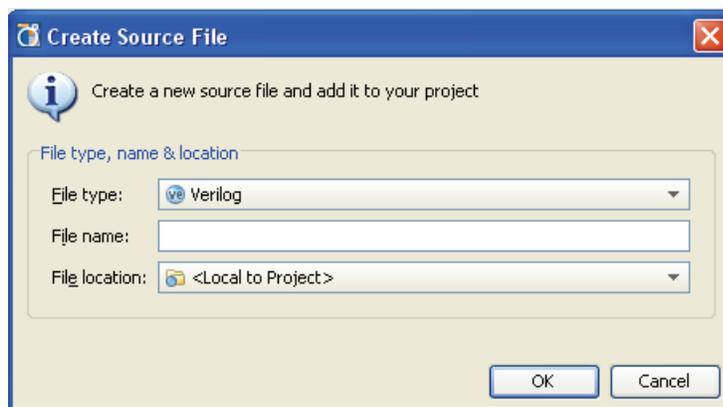
Figure 3-4: **Create Source File Dialog Box**

1. You can define the following information in the Create Source File dialog box:

   - • **File type**: Select one of the following file types:

     - - **Verilog**: Create a Verilog format file (.v).

     - - **Verilog Header**: Create a Verilog Header format file (.vh).

- **VHDL**: Create a VHDL format file (`.vhdl`).

- • **File name**: Enter a name for the new HDL source file.

- • **File location**: Designate a location to create the file.

2. Click **OK**.

   A placeholder for the file is added to the list of sources. The file itself is created when you click **Finish.**

3. You can repeat the **Create File** command multiple times to define a number of new modules to add to the project.

4. In the **Add Sources** dialog box, specify the appropriate Library for the source file. By default, sources are added to the `work` library.

5. Click **Finish** to add the specified sources to the project.

   - • If you have used Create Files, the Define Modules dialog opens as discussed in Defining New Modules, page 41.

   - • If you have not defined new modules, the process continues with Adding IP, page 42.

Defining New Modules

If you have specified new RTL source files to add to the project, you will need to define the module or architecture in the Verilog or VHDL code. The PlanAhead tool provides the Define Modules dialog box to facilitate creating new RTL code, as shown in Figure 3-5.

Note: The Define Module dialog box will open when you have defined the various source and constraint files for the project, selected the target part, and finish creating the project.
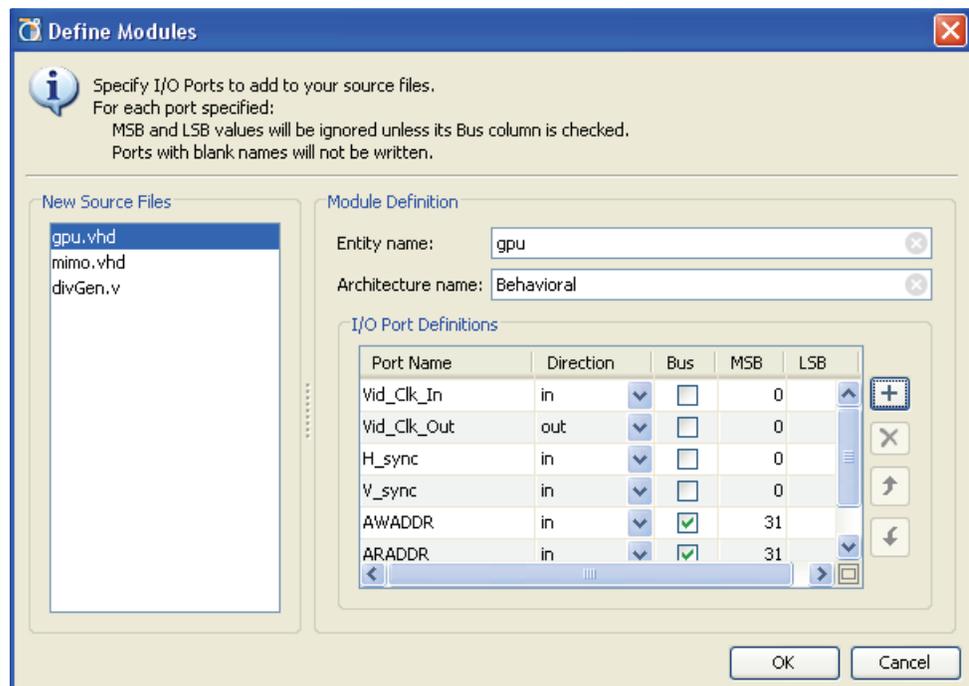
Figure 3-5: **Define Modules Dialog Box**

1. The Define Modules dialog box is populated with the new source files that were defined by the Create File command as described under Creating RTL Sources, page 40. Both Verilog and VHDL modules can be defined in this form.

2. Start by selecting a specific module to define from the list of **New Source Files**.

3. The following information can be provided in the Define Modules dialog box to speed RTL coding:

- **Entity name/Module name**: Specify the name for the entity construct in the VHDL code, or the module name in the Verilog code. Although the entity or module name defaults to the file name, it does not have to match file name.

- **Architecture name**: Specify the Architecture for the RTL source file. By default the name is **Behavioral**.

  Note: This field only applies to VHDL code, and does not appear when defining Verilog modules.

- **I/O Port Definitions**: Define the ports to be added to the module definition.

  - **Port Name**: Define the name of the port to appear in the RTL code.

  - **Direction**: Specify if the port is an Input, Output, or Bidirectional port.

  - **Bus**: Specify if this is a bus port. Define the width of the bus using the MSB and LSB fields as described below.

  - **MSB**: Define the number of the most significant bit. This combines with the LSB field to determine the width of the bus being defined.

  - **LSB**: Define the number of the least significant bit.

    Note: MSB and LSB are ignored if the port is not a bus port.

4. When you have finished defining the details of each of the new modules listed under **New Source Files**, click **OK** to create the RTL source files and add the modules to your project. The Sources view lists the newly defined modules.

5. Open the new source files in the Text Editor to edit as needed. Double-click the file, or select **Open File** from the popup menu, to open the file in the Text Editor for editing. See Using the Text Editor in Chapter 4 for information on editing the newly created file.

Adding IP

After specifying RTL source files for your project, you can add existing IP cores using existing CORE Generator core files (`.xco`) created outside of the PlanAhead tool. An XCO file is a log file that records all the customization parameters used to create the core and the project options in effect when the core was generated. The XCO file is used by the PlanAhead tool to recreate the core in the new project.

Note: XCO core files can only be added to RTL projects.

Figure 3-6, page 43 shows IP cores being added to the project.
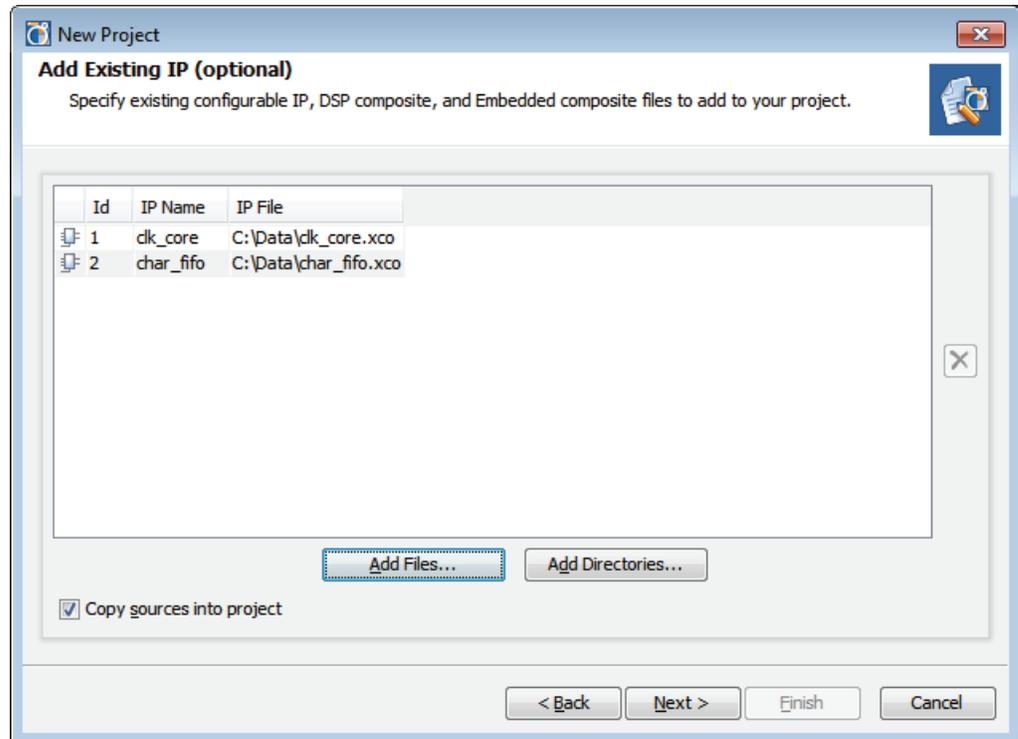
Figure 3-6: **Add Existing IP Dialog Box**

1. Use the **Add Files** button to navigate to the CORE Generator core files (`.xco`) to add to the project.

2. Use the **Add Directories** button to specify a directory to add all Core Generator core files found in the directory to the project.

3. Select the **Copy Sources into Project** checkbox to copy files into the local project directory instead of referencing the original files.

4. Click **Next**.

Parameterized cores can also be loaded into the project by running the CORE Generator software from within the PlanAhead tool by using the IP Catalog command. See Managing IP Cores, page 66 for more information.

In some cases, third-party providers offer IP as synthesized NGC or EDIF netlists. To load these files into a design, use the **Add Sources** command, and specify **Add or Create Design Sources**.

Adding Constraints

The New Project wizard prompts you with the optional Add Constraints dialog box, shown in Figure 3-7, page 44, to add or create top-level UCF files or module-level Netlist Constraint Files (NCF) files.
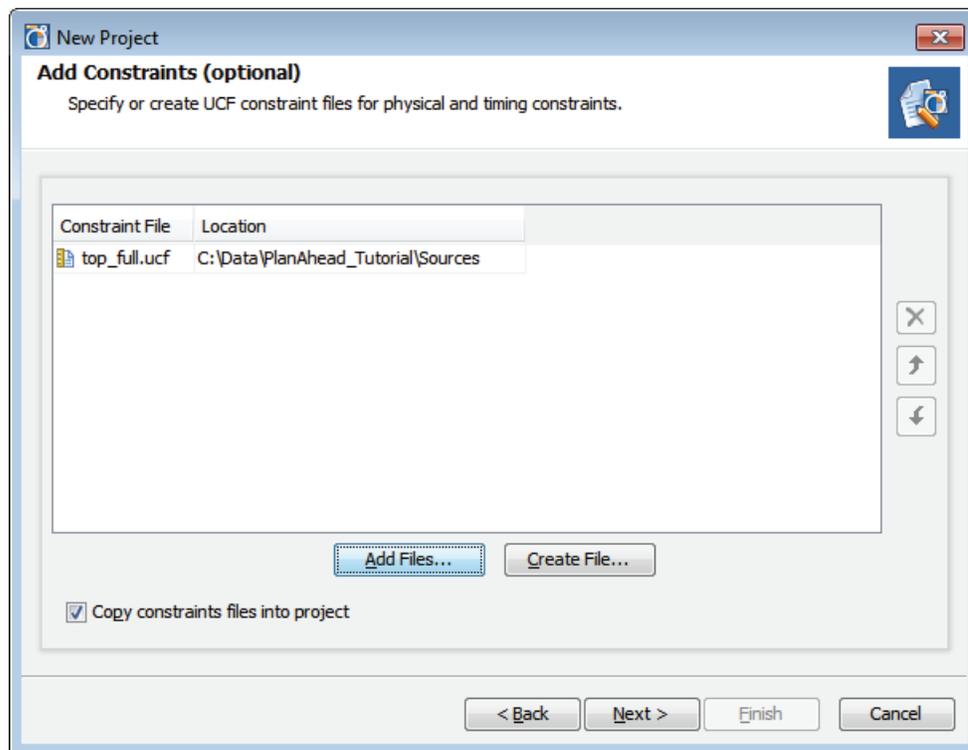
Figure 3-7: **Add Constraints Dialog Box**

You can add or create constraint files as needed for your project. Refer to Managing Constraints, page 53 for a complete description of this process.

Any UCF, NCF, or XST Constraint File (XCF) found in the same directories as the RTL, or Netlist source files you have already added to the project, are listed automatically as constraint files to be added to the project.

The Add Constraints dialog box has the following options:

- **Add Files**: Browse and select a UCF, NCF, or XCF to add to the project.

- **Create File**: Create a new top-level UCF for the project.

- **Remove**: Remove the selected UCF from the constraint files list.

- **Up / Down**: Move a constraint file up or down in the listed order of the UCF. Constraints are order-dependent; the last read definition of a constraint over-writes earlier definitions.

- **Copy Constraints into Project**: Check box copies constraint files into the local project directory instead of referencing the original files.

After adding the constraint files, and optionally setting the target UCF, click **Next**.

Selecting a Default Part or Board

The New Project wizard prompts you to select a Part or Board, as shown in Figure 3-8.
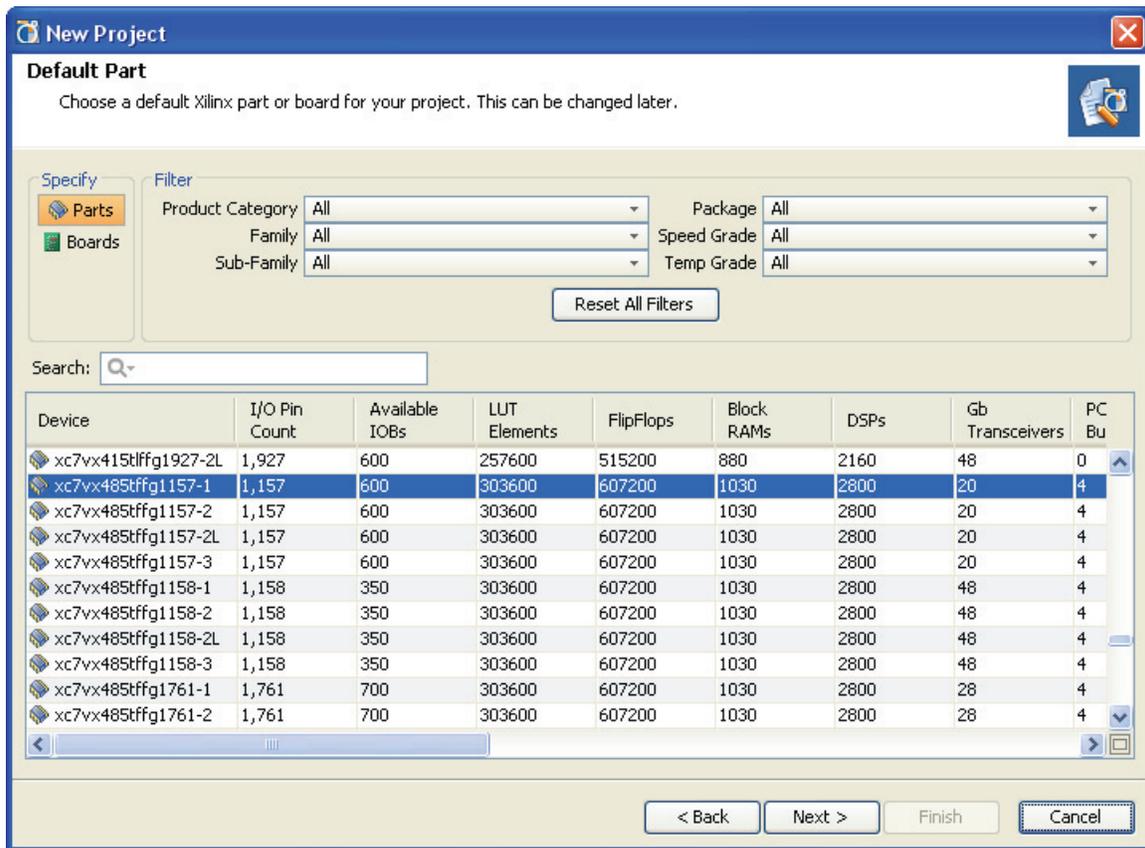
Figure 3-8: **Specifying the Default Part**

The Default Part dialog lets you specify the target Xilinx device for the design, or choose a TDB board as the target for the design:

- **Parts**: Available devices display in a scrolling list. Information about the device resources displays in a table view. You can filter the list to help you select a target device using the Product, Family, Sub-Family, Package, Speed Grade, and Temp Grade filters at the top of the dialog box.

    Note: Zynq™ devices include processor sub-system IO pins (PSSIO) that are read-only. These pins are not configurable as programmable logic (PL) pins on the Zynq device, but are configured as processor system (PS) pins, and imported from XPS. Refer XPS Help for more information.

- **Boards:** Available Targeted Design Platform (TDP) boards, and the Xilinx part used on the board, display in a scrolling list. Information about device resources displays in a table view, such as I/O pin count, the number of LUTs and FFs, available Block RAMs. You can also filter the list according to device family, package, and speed grade.

You can also use the **Search** command to limit the listed devices to those matching the search criteria you specify.

1. Select a target part or board from the list, and click **Next**.

   Note: You can change the target part when opening an Elaborated or Synthesized Design and during synthesis and implementation. See Configuring Project Settings, page 88 for more information.

   The New Project Summary page opens.

2. To build the project, click **Finish** in the Summary page.

   The project environment then displays with the Project Manager-related views available.

Creating a Post-synthesis Project

A second project type of begins with a synthesized netlist and corresponding constraints. You can then analyze, floorplan, and implement the design using the Floorplanning and Implementation environment.

1. Follow the project creation steps described in Creating a New Project, page 37.

2. Select the **Post-synthesis Project** option in the Project Type page.

   a. If desired, select the **Do not specify sources at this time** checkbox, and go to Selecting a Default Part or Board, page 45. This will skip the following steps of adding sources files and constraints, and allow you to select the target part and create the project.

   b. Select the **Enable Partial Reconfiguration** checkbox to define the project as a partial reconfiguration project.

      Note: The PlanAhead tool uses a special type of project to support Partial Reconfiguration designs. This capability is available only through special licensing, and is covered in the *Partial Reconfiguration User Guide (UG702)*, cited in Appendix E, Additional Resources.

3. Click **Next**.

Selecting a Top-Level Netlist and Module Search Path

The next page in the New Project wizard, shown in Figure 3-9, lets you specify netlist files to read, identify the file containing the top module, and define directories to search for lower-level module netlist files.
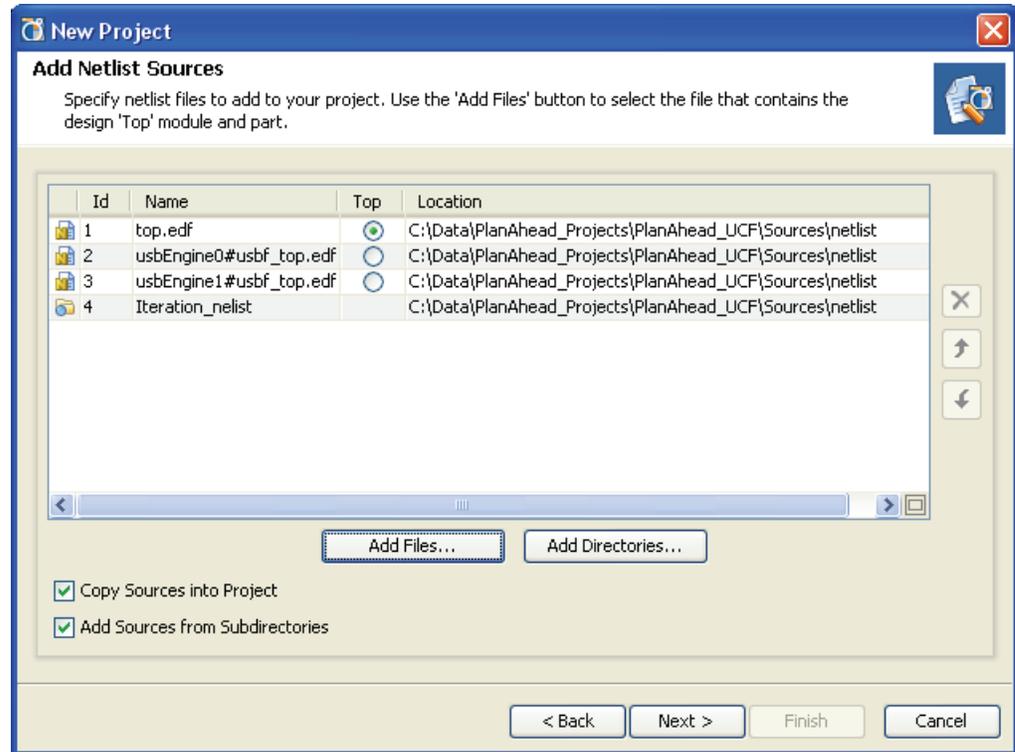
Figure 3-9: **New Project Wizard: Specify Netlist Files**

1. Edit the definable options:

   • **Add Files**: Invokes a file browser so you can select netlist files (NGC or EDIF) to add to the project.

      - **Top**: Enable the radio button for the file containing the top-level netlist.

   • **Add Directories**: Select directories to search for modules and cores.

   • **Remove Selected Files and Directories**: The 'X' icon removes the selected source files and directories from the list.

   • **Move Selected Files and Directories Up**: The upward pointing arrow icon moves the file or directory up in the list order.

   • **Move Selected Files and Directories Down**: The downward pointing arrow icon moves the file or directory down in the list order.

   • **Copy Sources into Project**: Copies the original source files into the project, and uses the local copied version of the file in the project. If you added directories of source files using **Add Directories**, the directory structure is maintained when the files are copied locally into the project. For a complete discussion of this topic refer to Using Remote Sources or Copying Sources into Project, page 63.

   • **Add Sources from Subdirectories**: Look for netlist files in the subdirectories of directories specified with **Add Directories**.

2. Click **Next**.

3. Add constraints as described in Adding and Creating Constraint Files, page 54.

4. Define a project part as discussed under Selecting a Default Part or Board, page 45.

   Note: In the case of a Netlist-based project, the PlanAhead tool will attempt to extract the target part from the top-level netlist and suggest that as the default part.

The New Project Summary page displays the selected options that define the project.

5. Click **Finish** to create and open the project.

Creating an I/O Planning Project

Another project type is an I/O Pin Planning project used specifically for planning the device pinout for an in-progress system-level design. You can create such a project prior completing the HDL or the synthesized EDIF. This allows you, for example, to exchange design information with the system-level or PCB designer. For more information about I/O pin planning, refer to Chapter 8, I/O Pin Planning.

In the Project Type page, select the **Create an I/O Planning Project** option.

Importing I/O Ports

The next page in the New Project wizard, shown in Figure 3-10, page 48, prompts you to select a file for importing I/O Port definitions and constraints.
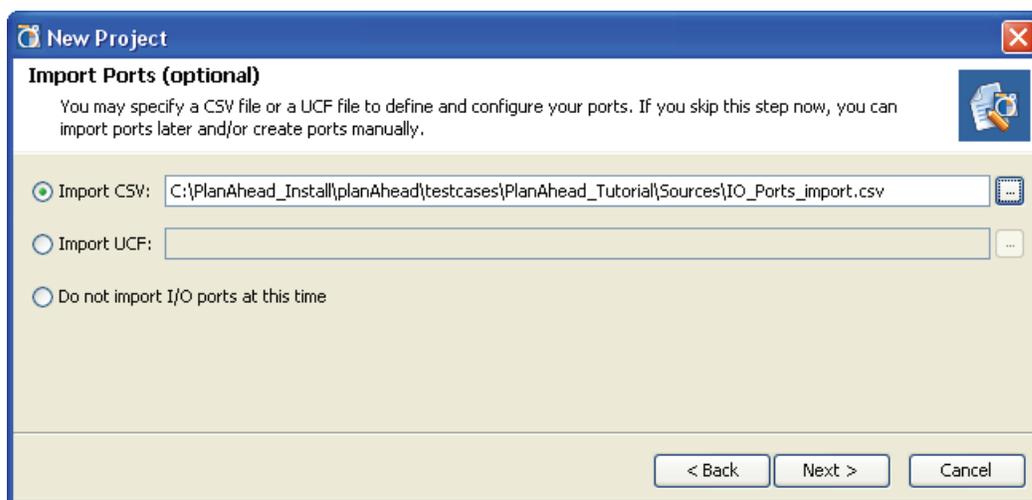
Figure 3-10: **New Project Wizard: IO Pin Planning**

1. Set the following options:

   • **Import CSV**: Select a CSV format file with I/O Ports definitions in the defined PlanAhead file format. See I/O Port Lists (CSV) File Format, page 404 for a specification of this file.

   • **Import UCF**: Select a UCF with I/O Port-related constraints only.

   • **Do not import I/O ports at this time**: Create an empty project. You can create or import I/Os later.

   Note: Use an RTL Source project to perform I/O pin planning on a design using RTL header or source files.

2. Click **Next**.

3. Define a project part as discussed under Selecting a Default Part or Board, page 45.

   The New Project Summary page displays the options you selected to define the project.

4. Click **Finish** to create and open the project.

Creating a Project with ISE Placement and Timing Results

Another type of project lets you import netlists with ISE® Placement and Routing (PAR) results along with corresponding constraints and project settings. This is used to analyze the PAR results using the implementation and analysis environment in the PlanAhead tool.

1. Follow the project creation steps described in Creating a New Project, page 37.

2. Select the **Import ISE Place & Route results** option in the Project Type page.

   The steps for creating this project type are the same as creating a synthesized netlist project, except this project also asks for ISE placement and timing files.

3. Specify the top-level netlist as described in Selecting a Top-Level Netlist and Module Search Path, page 46.

4. Define a project part as discussed under Selecting a Default Part or Board, page 45.

Figure 3-11, page 49 shows the dialog box that prompts for the ISE implementation results. You can import the placement and routing results and timing files generated in ISE. The PlanAhead tool then uses these files to create an implemented design for viewing and analysis.
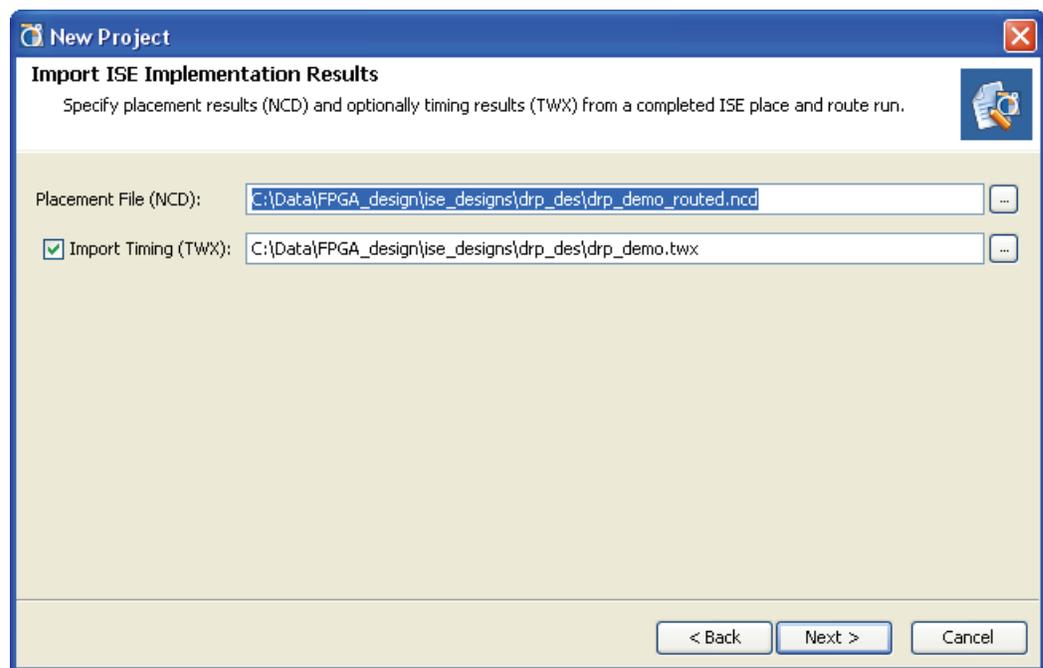
Figure 3-11: **New Project Wizard: ISE Implementation Results**

5. In the **Import ISE Implementation Results** page, edit the following options:

   • **Placement File (NCD)**: Select an implemented NCD file to import, and the PlanAhead tool automatically converts the file to read the placement information. NCD is the output file from the ISE software place and route application (PAR).

   • **Import Timing (TWX)**: Check this option, then locate and select a TRCE command output file from the ISE Implementation, such as a TWX format file.

6. Click **Next**.

   The New Project Summary page displays the various options selected to define the project.

7. Click **Finish** to create and open the project.

You may be prompted to select the top module of the design. The PlanAhead tool identifies the top module, and asks you to confirm the selection. When the implemented design is complete, the PlanAhead tool prompts you to
Open Implemented Design, Generate Bitstream, or View Reports.

The PlanAhead tool opens the Implemented design environment with design placement and timing results loaded and related views available.

Importing an External Project

You can import an existing RTL-level project file created outside of PlanAhead in ISE, XST, or Synplify tools. The PlanAhead tool detects the source files in the specified project and automatically adds the files to the new project. Settings such as top module, target device, and VHDL library assignment are also imported from these existing projects.

1. Follow the steps described in Creating a New Project, page 37.

2. In the Project Type page, shown in Figure 3-2, page 38, select **Imported Project**.

   The Import Project dialog box opens as shown in Figure 3-12, page 50.
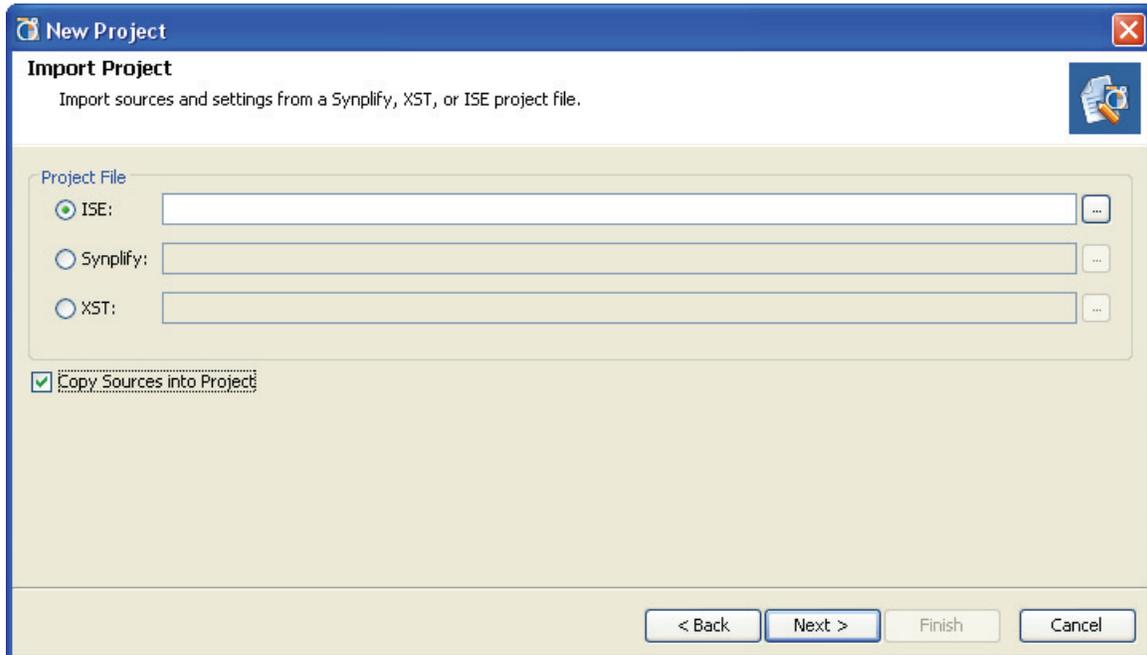
Figure 3-12: **Import Projects**

3. Specify the **Project File** to import.

   • **ISE**: Import the specified Xilinx ISE (`.xise`) project file.

   • **Synplify**: Import the specified Synplify (`.prj`) project file.

   • **XST**: Import the specified XST (`.xst`) project file.

4. Select the **Copy Sources into Project** checkbox to have the design sources copied locally into the PlanAhead tool project instead of being referenced from their current location.

5. Click **Next**.

   The New Project Summary page displays the options that are selected to define the project.

6. Click **Finish** to create and open the project.

The PlanAhead tool imports the RTL source files, constraint files, and run settings from the specified project, and creates a project file in the specified directory.

The PlanAhead tool writes a summary of the import process to the *Import Summary Report* log file in the new project directory. You can review the steps used in creating the project, and any errors or warnings encountered, in this summary file.

Managing Projects

Opening Existing Projects

You can open existing projects in the PlanAhead tool. As a project is opened, the PlanAhead tool restores the *state* of the project from the time the project was closed. The project state includes the current source file order, disabled and enabled source files, active and target constraint files, and the state of synthesis, simulation, and implementation runs.

To open a project, use one of the following methods:

- In the Getting Started page, click **Open Recent Project** or **Open Project**.
- Select **File > Open Project**.
- Click **Open Project** on the toolbar menu.

The **Open Project** dialog box lets you select a project file (`.ppr`). The File Preview window in the **Open Project** dialog box displays information about the currently selected file.

To open a project from within Windows Explorer, double-click a PlanAhead tool project file (.ppr).

Opening Multiple Projects

To open multiple projects in a single session, use any of the methods previously described to open an additional project while a project is already open. The PlanAhead tool asks if you want to close the current project.

If you do not close the first project, both projects are opened. Each open project has a separate main window.

When opening multiple projects from the same PlanAhead application process, the commands used in all open projects are written to the Tcl Console. This can be confusing, when reviewing the transcript of commands, as it is not clear which project they were associated with. In addition, there is only a single `planAhead.jou/.log` file for the application for all projects, which can also lead to some confusion.

Note: System memory requirements can hinder performance when opening multiple projects.

Saving a Project

To save projects, select **File > Save Project** or **File > Save Project As**. When you save a project, the PlanAhead tool prompts you to save unsaved changes to designs and source files. **Save Project As** copies the entire project directory structure to a new specified location, and maintains the status of the existing runs.

Archiving Projects

To create a project archive to store as backup, or to encapsulate the design to send to a remote site, click **File > Archive Project**. The PlanAhead tool:

- • Parses the hierarchy of the design
- • Copies the required source files, include files, and remote files from the library directories
- • Copies the constraints
- • Copies the results of the various synthesis, simulation, and implementation runs
- • Creates a ZIP file of the project

Figure 3-13, page 52 shows the Archive Project dialog box.
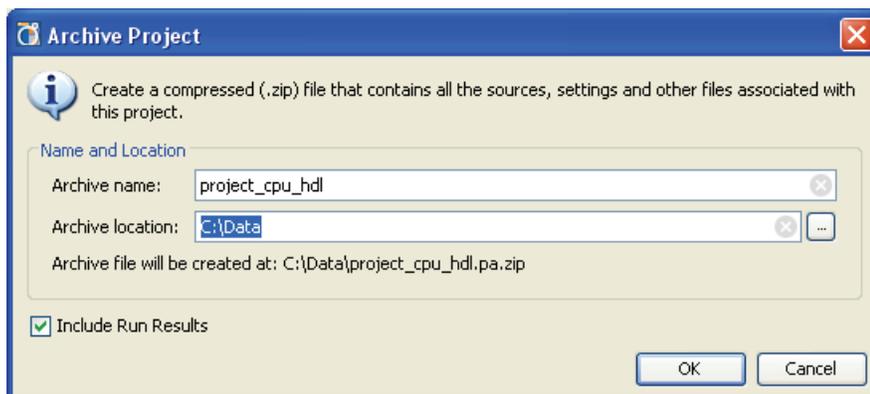
Figure 3-13: **Archive Project**

To archive a project:

1. Select **File > Archive Project**.
2. Specify an **Archive Name**.
3. Specify an **Archive Location** to write the archive file.
4. Enable or disable **Include Run Results** to include the settings and results of the runs performed on the project, then click **OK** to create the archive.

The PlanAhead tool creates a ZIP file archive of the project containing the required source and include files, and the run files, if specified, and creates an `archive.log` file of the archival process that is included in the archive ZIP file. You can review the creation of the archive in the `archive.log` file.

Closing a Project

To close projects, select **File > Close Project**. When you close a project, you are prompted to save any unsaved changes to the design or source files.

Managing Project Sources

The PlanAhead tool can create new source files and manage existing source files that are either local to the current project or remotely referenced from a library. You can add Verilog and VHDL source files to an existing project at any time in the design flow. You can also create and add constraint files, add a simulation test bench, and add existing IP cores to your design.

To create new source files or add existing source files to your project, select **File > Add Sources** from the main menu, or **Add Sources** from the popup menu or from the Flow Navigator. The Add Sources wizard opens as shown in Figure 3-14, page 53.
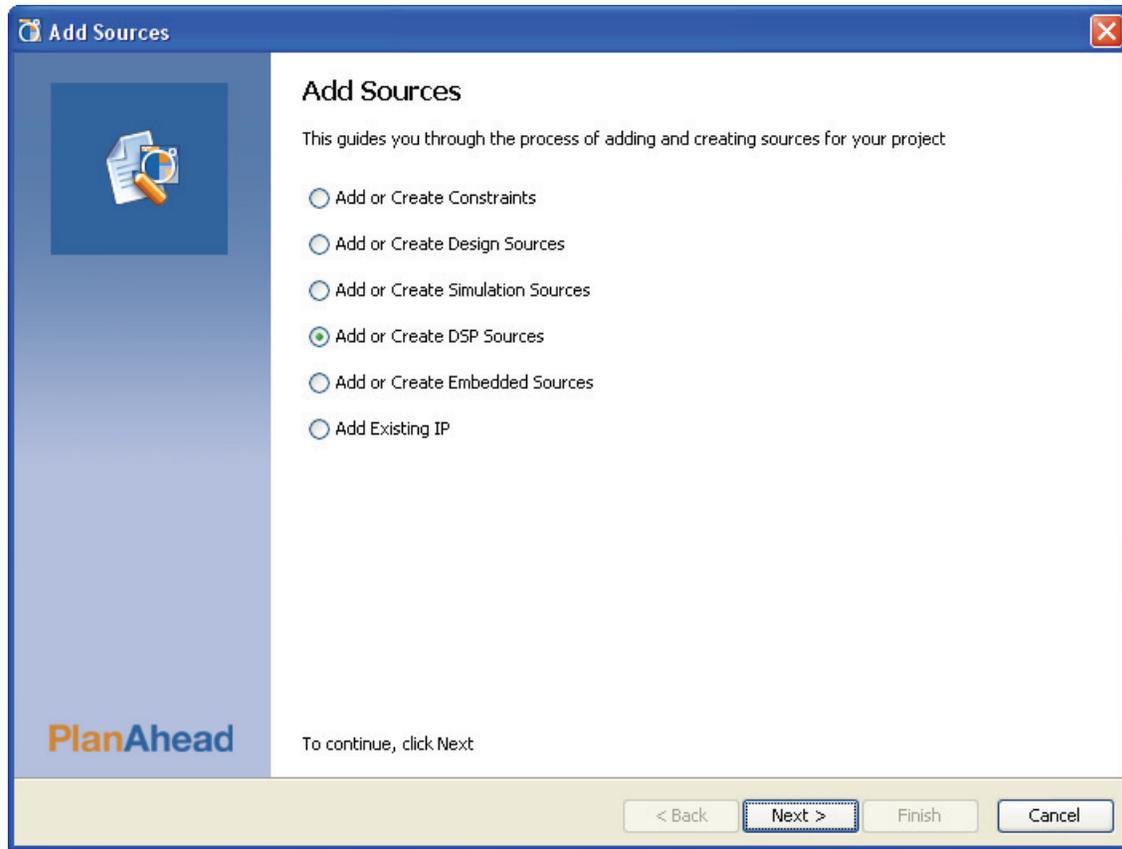
Figure 3-14:   **Add Sources Wizard**

Refer to the following sections for discussion on adding the various types of source files:

- Managing Constraints, page 53
- Managing Design Source Files, page 58
- Managing Simulation Sources, page 64
- Managing DSP Sources, page 72
- Managing Embedded Processor Sources, page 75
- Managing IP Cores, page 66

Managing Constraints

The PlanAhead tool provides flexibility for defining and using constraints in a project. You can use a single UCF to add and maintain the design constraints, or you can use multiple UCF files to organize the constraints into separate files. Add the UCF while creating the project or later by using **Add Sources**.

You can create multiple constraint sets to experiment with various types of constraints, or store multiple versions of constraints. Each constraint set can contain one or more constraint files.

You can open multiple designs referencing a single constraint set. However, you must be careful to manage changes made to multiple designs that reference the same constraint set. If the PlanAhead tool detects unsaved changes in multiple designs, it prompts you to select which design to save to the referenced constraint file.

Note: This could overwrite any unsaved constraint definitions in an unsaved design.

An Implemented Design saves a snapshot of the constraint set used during the implementation run. This constraint set can have the same name as the active constraint set in the open project.

When opening an Implemented Design, the constraint set loaded from the implementation run could be older than the constraint set currently in the project memory. This would cause the loss of newly-defined constraints when you save the design. Generally, the PlanAhead tool manages these revision issues, and prompts you to take the appropriate action as needed. However, you should keep in mind the potential conflict between the current constraint set in memory and any existing constraints associated with an Implemented Design.

Adding and Creating Constraint Files

The PlanAhead tool supports a variety of constraint file formats. This lets you add top-level constraints in UCF or XCF, or module-level constraints in NCF. Constraints can include placement, timing, and I/O restrictions.

To add constraint files to the project:

1. Select **File > Add Sources** from the main menu, or **Add Sources** from the popup menu or from the Flow Navigator.

   The Add Sources wizard opens as shown in .

2. Select **Add or Create Constraints**, and click **Next**.

   The Add or Create Constraints dialog box opens as shown in .
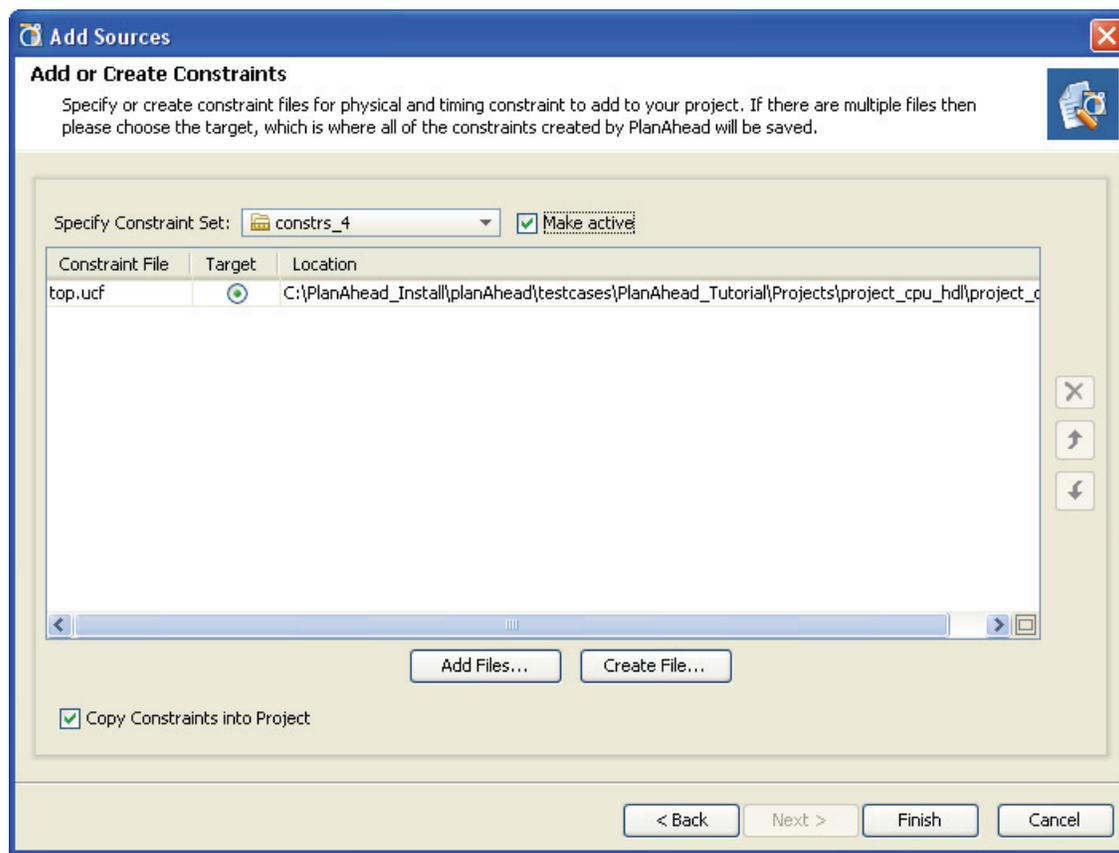
Figure 3-15: **Add or Create Constraints**

The Add or Create Constraints dialog box has the following options:

- **Specify Constraint Set**: Defines the constraint set into which the constraint files are placed. By default, the currently active constraint set is selected, but you can specify a different constraint set or define a new constraint set using the drop-down menu.

- **Add Files**: Browse to, and select the UCF, NCF, or XCF files to add to the project.

- **Create File**: Create a new top-level UCF for the project.

- **Remove**: Remove the selected UCF from the Constraint files list.

- **Up / Down**: Move a constraint file up or down in the listed order of UCF files. Constraints are order-dependent, with the last read definition of a constraint over-writing earlier definitions. If there are multiple files in a constraint set, the order in which they appear in the Sources view corresponds to the order that the PlanAhead tool processes the files. The first file in the list is the first file processed. If the same constraint appears in more than one constraint file, the last file read has precedence in defining the constraint.

- **Copy Constraints into Project**: Select the checkbox to copy constraint files into the local project directory instead of referencing the original files.

Setting the Target UCF

When you add multiple UCF files to a project, you must define the *target* UCF. The PlanAhead tool writes newly created constraints to the target UCF. Existing constraints, that are modified during design are written back to the UCF from which they originated, *not* the target UCF.

You can change the target UCF at any time using the **Set Target UCF** command from the Sources view.

Note: NCF and XCF files cannot act as a target for new constraints. New constraints must be written to a UCF.

Referencing Original UCF Files or Copying Files

As with other source files, you can add UCF files to reference from a remote location or copy the files locally into the project directory when they are added. When you add remote files, the PlanAhead tool automatically detects the latest file version and prompts you to **Reload the design** with the latest files.

To copy files into a project, select **Copy Constraints into Project** from the Add Constraints dialog box.

See for more information.

Using Constraint Sets

A constraint set is one or more constraint files that are maintained independently and concatenated into a single UCF for analysis and implementation. A constraint set defines the constraint files to be used at specific moments, or under specific conditions, in the design process. By defining multiple constraints sets, you can, for example, specify different active constraints to resolve floorplanning and timing problems.

A constraint set must include at least one UCF in order to save any design constraints. If the constraint set does not include a UCF, the PlanAhead tool will create one for the project, named `<design>.ucf`, and will write constraints to that file.

Creating Constraint Sets

In the PlanAhead tool, you can define constraints at various stages of the design flow, including Elaborated Design, Synthesized Design, Implementation, and Analysis.

Figure 3-16 shows the Create Constraint Set Name dialog box from the Add Constraints command.

Figure 3-16: **Create Constraint Set Name dialog box**

To create constraint sets:

1. In the **Add Constraints** dialog box, select the **Create Constraint Set** command in the **Specify Constraint Set** field, and enter a name for the constraint set, as shown in Figure 3-16.

   When you select **Create Constraint Set**, the PlanAhead tool prompts you to enter a name for the new constraint set, and provides a checkbox to make it the active constraint set, as seen in Figure 3-15, page 54.

2. Click **Add Files** to select UCFs, NCFs, or XCFs to add to the constraint set to the design sources.

3. Use the **Create File** command to select a location and name for a new UCF file.

Using the Save Constraints As Command

You can also create a new constraint set by saving changes made to constraints during the design and analysis process. **Save Constraints As** lets you enter a new constraint set name into which to save all constraints. Figure 3-17 shows the Save Constraints As dialog box.

Figure 3-17: **Save Constraints As Command**

With multiple places to make constraint changes, it is convenient to save changes to a new constraint set to manage changes or support "what-if" analysis.

The Save Constraints As dialog box:

- Creates a new constraint set.
- Copies the active constraint files into the new constraint set in the local project directory.
- Writes any modifications to the constraints to the copied constraint files, leaving the original UCF files unchanged.

- Saves the design partition information and changes to ChipScope™ cores with the new design.
- Provides an option to make the new constraint set active in the project.

Defining the Active Constraint Set

If more than one constraint set exists, you must designate an *active* constraint set. The PlanAhead tool uses the active constraint set by default when you launch the next implementation run or when you open an Elaborated or Synthesized Design.

To set the active constraint set, select the constraint set, and click **Make active** from the popup menu. The active constraint set appears **bold** and has the (active) designation in the Sources view as shown in Figure 3-18.
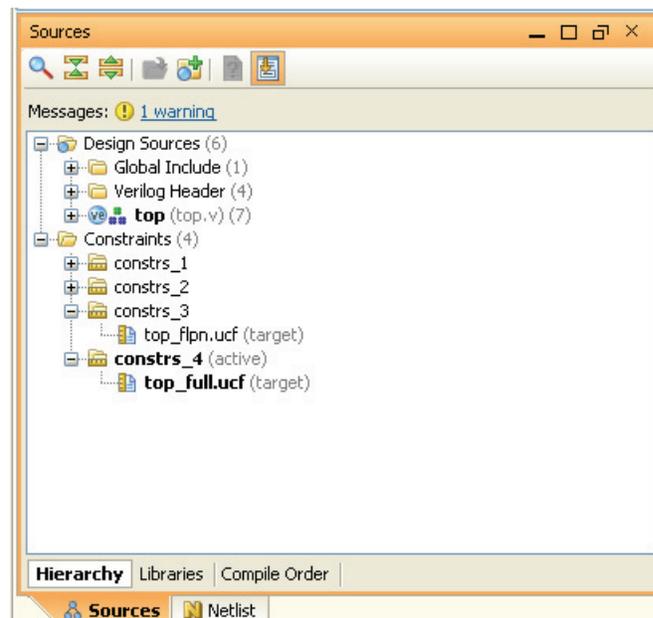
Figure 3-18: **Setting the Active Constraint Set**

Using Module-level Constraint Files

The PlanAhead tool supports adding module-level NCF files into a project. These files are typically associated with an IP core. The module constraint file must have the same name as the module to which it applies to be recognized and applied. For example, xyz.ngc must have a netlist constraint file named xyz.ncf. When you add the NCF to the project, the PlanAhead tool automatically makes the association to the proper netlist module.

Note: NCFs are processed only if the name matches the name of the module-level netlist.

In projects that have constraints defined in both top-level UCF files and module-level NCF files, the following rules apply:

1. If the constraints overlap, UCF overrides NCF and any embedded netlist constraints.
2. The NCF overrides any embedded constraints.

You can add NCF files to the project in the New Project wizard, or use the **Add Sources** option. An NCF that is in the same directory as an RTL source file loaded into a project is automatically added to the Add Constraints dialog box.

The PlanAhead tool also loads constraints automatically when an NGC format core has embedded timing constraints, as is often the case with EDK and some CORE Generator software cores.

The `ngc2edif` command extracts timing constraints before passing the EDIF netlist to the PlanAhead tool. This allows the PlanAhead tool to recognize the constraints and to expose them during design analysis. These files are treated as read-only, and does not enable modification to them directly; however, you can define new values for module-level constraints.

New constraint values are written into the target UCF. Because the PlanAhead tool passes the top-level UCF to implementation after the module-level NCF, the new constraint values are given higher precedence, and are used during implementation. You cannot remove constraints from the NCF because it is read-only.

Note: When modifying module-level constraints, it is a "best practice" to edit the files in the original source using the IP creation method.

Exporting Constraints

Designers often use the PlanAhead tool to create constraint files for use in scripting command line design flows. To export constraints for a command-line flow, select **File > Export > Export Constraints**.

You can export the I/O STANDARD constraints for I/O Ports and Banks, both user-specified and the default values assigned by the PlanAhead tool, to a UCF by selecting **File > Export > Export I/O Ports**, and generating the UCF output.

Managing Design Source Files

The following subsections describe how to manage design source files.

Adding and Creating Design Sources

To add design source files, select the **Add or Create Design Sources** option from the **Add Sources** command, and click **Next**.

Figure 3-19, page 59 shows the Add Sources dialog box.
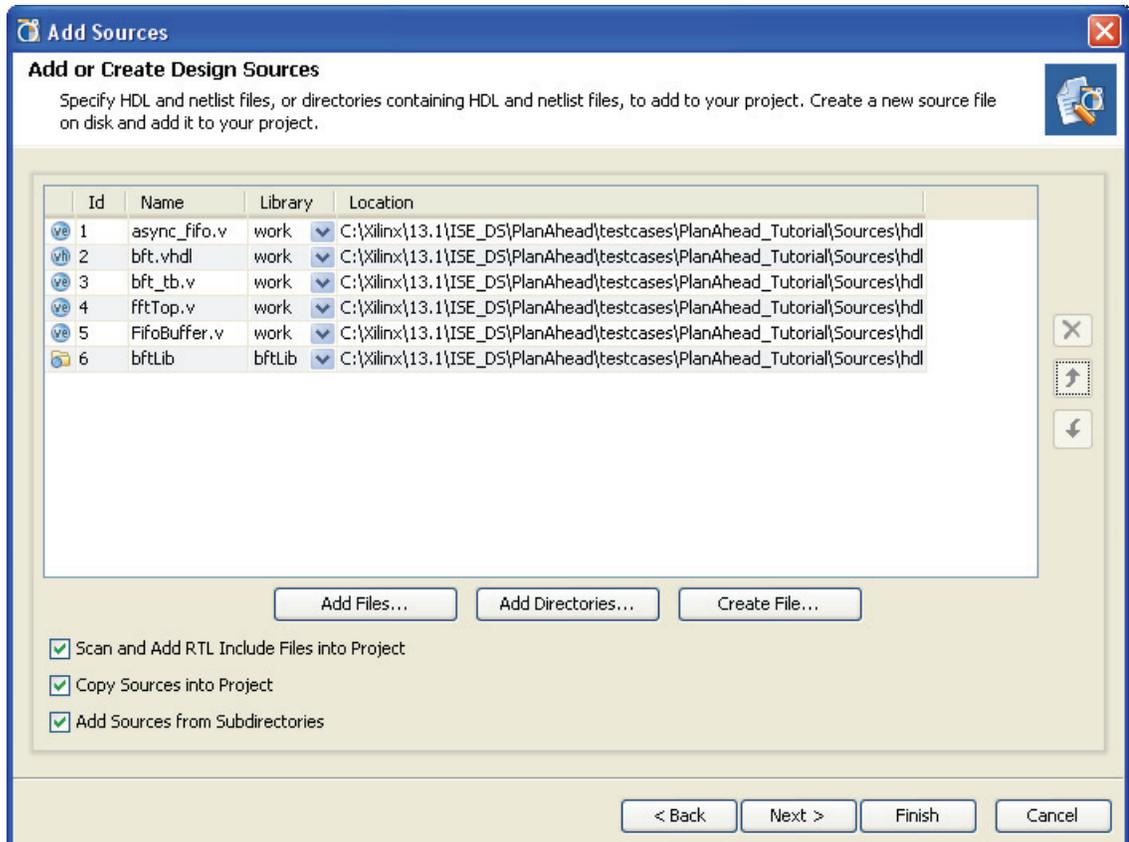
Figure 3-19: **Add Sources Dialog Box**

Adding RTL Sources

The available commands and options of the Add Sources dialog box are:

- **Add Files**: Invokes a file browser so you can select files to add to the project. The file types that can be added to a project include HDL files, as well as EDIF and NGC files, BMM, ELF, and others.

  Note: Each file or directory is represented by an icon indicating it as a file or folder. A small red square indicates it is read only.

- **Add Directories**: Opens a directory browser to add source files from the selected directories. Files in the specified directory with valid source file extensions are added to the project.

  - **Library**: Specify the RTL library for a file or directory by selecting one from the currently defined library names, or specify a new library name by typing in the Library text field.

  - **HDL Source for**: Specify if the source being loaded is an RTL source file for synthesis and simulation, or an RTL test bench for simulation only.

- **Create File**: Invokes the Create Source File dialog box where you can create new VHDL, Verilog, or Verilog Header files. See Creating RTL Sources, page 60 for more information.

- **Remove Selected Files and Directories**: The 'X' icon removes the selected source files and directories from the list.

- **Move Selected Files and Directories Up**: The upward pointing arrow icon moves the file or directory up in the list order. The order of the files affects the order of elaboration and compilation during downstream processes such as synthesis and simulation.

- **Move Selected Files and Directories Down**: The downward pointing arrow icon moves the file or directory down in the list order.

- **Scan and Add RTL Include Files into Project**: Scans the added RTL files and adds any referenced Verilog `include` files.

- **Copy Sources into Project**: Copies the original source files into the project and uses the local copied version of the file in the project.

- If you added directories or source files with **Add Directories**, the directory structure is maintained when the files are copied locally into the project. For a complete discussion of this topic, refer to Using Remote Sources or Copying Sources into Project, page 63.

- **Add Sources from Subdirectories**: Adds source files from the subdirectories of directories specified using the **Add Directories** option.

Creating RTL Sources

To create a new RTL source file:

1. Select the **Add or Create Design Sources** option from the **Add Sources** command.

2. Click **Create File**.

    The Create Source File dialog box opens, in which you can define the type, name, and location of the new source file you are creating, as shown in Figure 3-20.
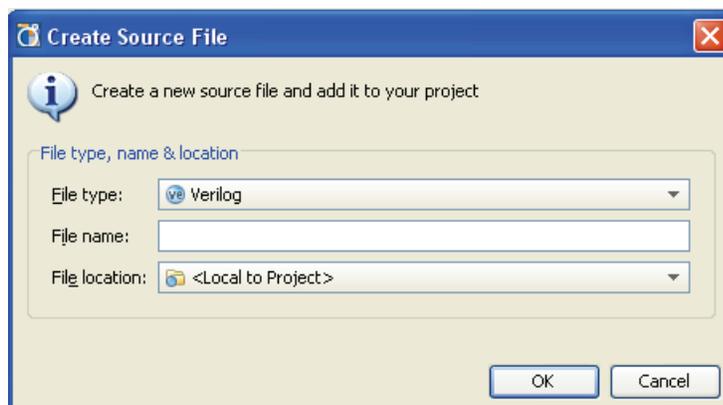
Figure 3-20: **Create Source File**

3. You can define the following information in the Create Source File dialog box:

    - **File type**: Select one of the following file types:

        - **Verilog**: Create a Verilog format file (`.v`).

        - **Verilog Header**: Create a Verilog Header format file (`.vh`).

        - **VHDL**: Create a VHDL format file (`.vhdl`).

    - **File name**: Enter a name for the new HDL source file.

    - **File location**: Designate a location to create the file.

4. Click **OK**.

    A placeholder for the file is added to the list of sources; the file is created when you click **Finish**.

5. You can repeat the **Create File** command multiple times to define a number of new modules to add to the project.

6. In the **Add Sources** dialog box, specify the appropriate Library for the source file. By default, sources are added to the `work` library.

7. Click **Finish** to add the specified sources to the project. If you have used Create Files, the Define Modules dialog box opens as discussed in Defining New Modules, page 61.

Defining New Modules

If you have specified new RTL source files to add to the project, you will need to write the Verilog or VHDL code to define the module. The PlanAhead tool provides the Define Modules dialog box to facilitate creating new RTL code, as shown in Figure 3-21.
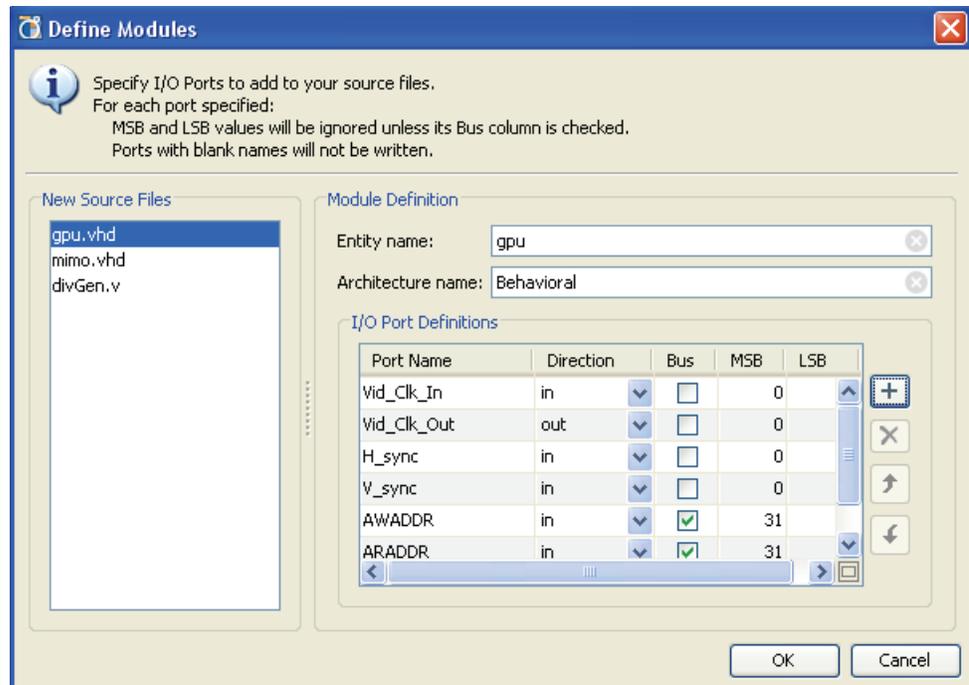
Figure 3-21: **Define Modules Dialog Box**

1. The Define Modules dialog is populated with the new source files that were defined by the Create File command as described under Creating RTL Sources, page 60. Both Verilog and VHDL modules can be defined in this form.

2. Start by selecting a specific module to define from the list of **New Source Files**.

3. The following information can be provided in the Define Modules dialog box to speed RTL coding:

   • **Entity name/Module name**: Specify the name for the entity construct in the VHDL code, or the module name in the Verilog code. Although the entity or module name defaults to the file name, it does not have to match file name.

   • **Architecture name**: Specify the Architecture for the RTL source file. By default the name is **Behavioral**.

   Note: This field only applies to VHDL code, and does not appear when defining Verilog modules.

   • **I/O Port Definitions**: Define the ports to be added to the module definition.

     - **Port Name**: Define the name of the port to appear in the RTL code.

     - **Direction**: Specify if the port is an Input, Output, or Bidirectional port.

- **Bus**: Specify if this is a bus port. Define the width of the bus using the MSB and LSB fields as described below.

- **MSB**: Define the number of the most significant bit. This combines with the LSB field to determine the width of the bus being defined.

- **LSB**: Define the number of the least significant bit.

Note: MSB and LSB are ignored if the port is not a bus port.

4. When you have finished defining the details of each of the new modules listed under **New Source Files**, you can click **OK** to create the RTL source files and add the modules to your project. You will see the newly defined modules listed in the Sources view.

5. Open the new source files in the Text Editor to edit as needed. Double-click the file, or select **Open File** from the popup menu, to open the file in the text editor for editing. See Using the Text Editor in Chapter 4 for information on editing the newly created file.

Specifying the Top Module and Reordering Source Files

The PlanAhead tool automatically determines the top-level of the design hierarchy and the order of elaboration, synthesis, and simulation for source files added to the project. The hierarchy of the design is displayed in the Hierarchy tab of the Source view. The file order is displayed in the Compile Order tab of the Sources view. See Using the Sources View, page 128 for more information. The automatic assignment of top module and compilation order are controlled using the Hierarchy Update command in the Sources view.

However, you can override the automatic determination of the top module, by manually specifying the top of the design hierarchy. To define the top module, from within the Sources view you can use the **Set as Top** popup menu command.

Note: The selected top is automatically reset to the best candidate if the specified top module cannot be found in the design source files, and the hierarchy update mode is set to automatic.

The PlanAhead tool automatically reorders files according to the requirements of the new top module. You can also manually order files according to your own requirements. Use the **Refresh Hierarchy** command in the Sources view to automatically reorder files based on updates to the source files.

To manually re-order source files, from within the Sources view, select a file and drag it up or down in the file list order. Alternatively, after selecting the file, execute **Move Up**, **Move Down**, **Move to Top**, or **Move to Bottom** from the Sources view popup menu.

Enabling or Disabling Source Files

When you add or create source files, those files are enabled in the Sources view by default. You can disable source files to prevent them from being elaborated, synthesized, or used in simulation.

You can load different versions of a source file, then enable or disable the appropriate source file in the Sources view to control the current configuration of the design.

- To disable source files, select the files in the Sources view, then select the **Disable File** popup command.

- To enable disabled files, select the files in the Sources view, then select the **Enable File** popup command.

Using Remote Sources or Copying Sources into Project

To provide project management flexibility, you can reference source files from a remote location or copy the source files into the project directory. Copy the files into the project if you plan to move or archive the project because the files are contained within the project.

When you add remote files, the PlanAhead tool automatically detects the latest file version, then prompts you to **Refresh your open Designs**, or to **Synthesize with the latest updates** made to the file.

Note: When you copy files into a project, the project is easier to port to another system; however, the PlanAhead tool does not automatically recognize external file changes. When remote files change, you must remove and re-add, or update the files using commands in the Sources view.

When you add sources to the project using the Add Sources command, you can copy the sources to the local project directory by selecting the **Copy Sources into Project** option.

If you initially add the sources as remote sources, but later wish to copy them into the project directory, use **Copy into Project** or **Copy All Files into Project** in the popup menu in the Sources view to copy some or all individual remote source files into the project directory.

- Local files that you copy into the project directory have a green dot next to the file name in the Sources view.

- Remote sources, that are not copied into the local project directory, have a green circle next to the file name, rather than a dot.

- A red dot indicates an RTL file that could not be located, either local or remote.

```
Design Sources (5)
    Verilog (3)
        work (3)
            async_fifo.v
            FifoBuffer.v
            bft_tb.v
    VHDL (2)
        work (2)
            bft_package.vhdl
            bft.vhdl
Constraints (1)
    constrs_1
```
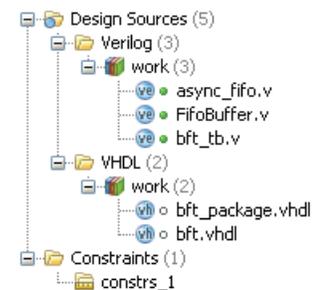
Updating Local Source Files

When referencing remote sources, the PlanAhead tool automatically detects the updated source file changes. However, with source files that are copied to the local project, any changes to the original source file are not recognized. You must manually update local source files, if necessary.

You can update source files that are copied into the local project directory using the following options:

- Select the source file in the Sources view and select the **Update File Contents** command from the popup menu

  A file browser opens with the original source file referenced. Click **OK** to reload the original source file and update the project file with any changes to the source file.

  You can also specify a different file, and the PlanAhead tool copies the contents of the new file into the local project file.

  For instance, if the original file is `File_1.v`, and you select `File_2.v`, the contents of `File_2.v` will be copied into and overwrite the original `File_1.v` contents.

- Click **Add Sources** from the popup menu to add the newly updated source files to the project. The PlanAhead tool then imports the added file into the project.

  However, because there is already a local source with the same name, the **Import Source Conflicts** dialog box, shown in Figure 3-22, page 64, prompts you to resolve the conflict by overwriting the existing file, or by not loading the newly added file.

Figure 3-22: **Import Source Conflicts**

Managing Simulation Sources

The PlanAhead tool lets you add simulation sources to the project for behavioral simulation of an RTL Project, or timing simulation of an implemented design. Simulation source files include Hardware Definition Language (HDL)-based test bench files to use as a stimulus for simulation. Simulation sources are used for behavioral and timing simulation in ISim. See Performing Behavioral Simulation, page 188 and Performing Timing Simulation, page 354 for more information.

The PlanAhead tool stores simulation source files in simulation file sets that display in folders in the Sources view, and remotely referenced or stored in the local project directory.

The simulation set lets you define different sources for different stages of the design. For example, one simulation source to provide stimulus for behavioral simulation of the Elaborated Design or a module of the design, and a different test bench to provide stimulus for timing simulation of the implemented design.

When adding simulation sources to the project, you can specify which simulation source set into which to add files.

Adding and Creating Simulation Source Files

To add simulation sources to a project:

1. Select either:

   • **File > Add Sources**.

   • **Add Sources** from the popup menu or from the Flow Navigator.

   The Add Sources wizard opens.

2. Select **Add or Create Simulation Sources**, and click **Next**. The Add or Create Simulation Sources dialog box opens as shown in Figure 3-23, page 65.
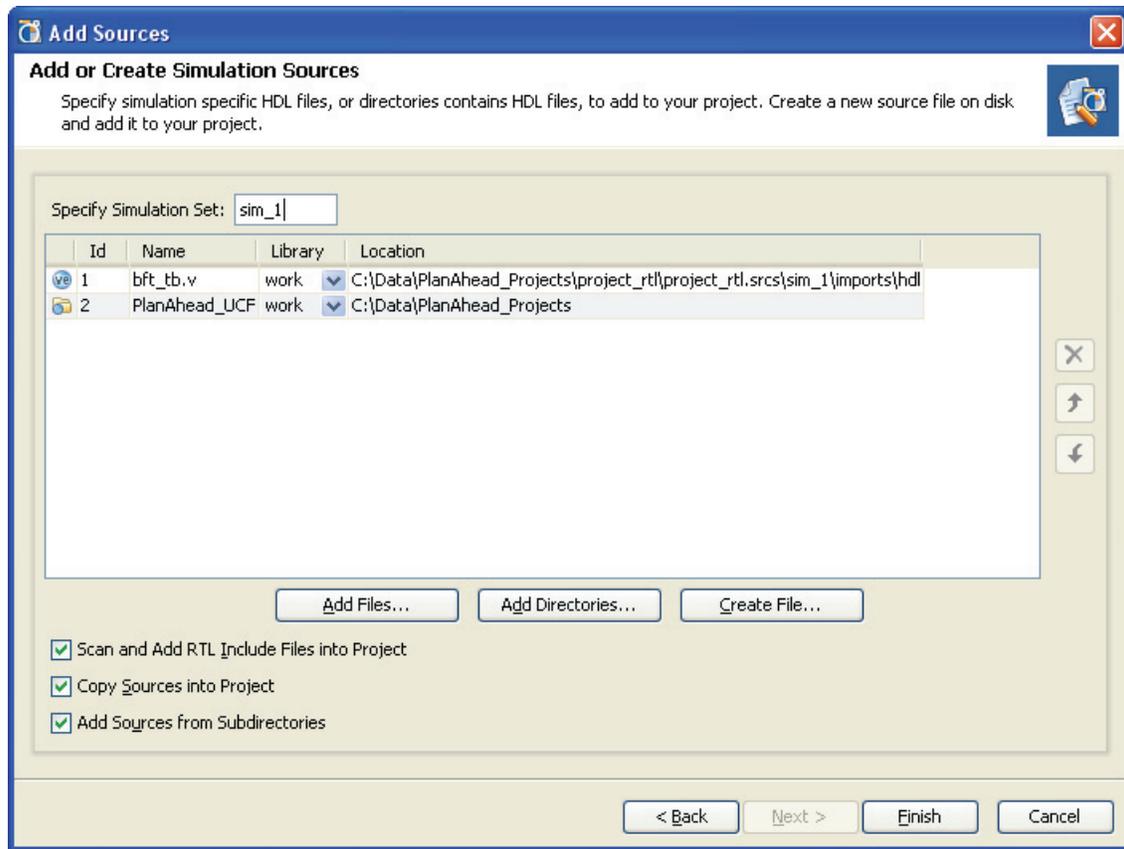
Figure 3-23: **Add or Create Simulation Sources**

The options are:

- **Specify Simulation Set**: Enter the name of the simulation set to test bench files and directories. If there is one or more simulation sets already defined, select the **Create Simulation Set** command from the pulldown menu to define a new simulation set.

- **Add Files**: Invokes a file browser so you can select simulation source files to add to the project.

- **Add Directories**: Invokes directory browser to add all simulation source files from the selected directories. Files in the specified directory with valid source file extensions are added to the project.

    - **Library**: Specify the library for an added file or directory by selecting one from the currently defined library names, or specify a new library name by typing in the Library text field.

       Note: The Verilog library is the /work directory.

- **Create File**: Invokes the Create Source File dialog box where you can create new simulation source files. See Creating RTL Sources, page 60.

- **Remove Selected Files and Directories**: The 'X' icon removes the selected source files and directories from the list.

- **Move Selected Files and Directories Up**: The upward pointing arrow icon moves the file or directory up in the list order.

- **Move Selected Files and Directories Down**: The downward pointing arrow icon moves the file or directory down in the list order.

- **Scan and Add RTL Include Files into Project**: Scans the added RTL file and adds any referenced `include` files.

- **Copy Sources into Project**: Copies the original source files into the project and uses the local copied version of the file in the project.

  If you selected to add directories of source files using the Add Directories command, the directory structure is maintained when the files are copied locally into the project.

- **Add Sources from Subdirectories**: Adds source files from the subdirectories of directories specified in the Add Directories option.

Managing IP Cores

The PlanAhead tool lets you add IP cores to a project from the Xilinx IP Catalog, existing IP core files created by the CORE Generator tool, and IP cores from third party IP providers.

Note: The PlanAhead tool reads IP from the ChipScope tool. It does not support reading Vivado native IP (XCI) files.

In RTL-based projects, you can add and manage IP cores in your project using the following methods:

- Adding Existing IP Cores, page 66
- Adding IP from the Xilinx Catalog, page 67
- Generating Output Products, page 70
- Managing DSP Sources, page 72

DSP modules can be imported like IP from SysGen, and Embedded Processor modules can be imported from XPS. These tools are integrated into PlanAhead to allow design modules to be added and managed from within the PlanAhead tool. See Managing DSP Sources, page 72 and Managing Embedded Processor Sources, page 75 for more information.

Note: In some cases, third-party providers offer IP as synthesized NGC or EDIF netlists. You can load these files into a design using the **Add Sources** command. See Managing Design Source Files, page 58.

Adding Existing IP Cores

To add existing IP cores (`.xco`) to a project:

1. Select **File > Add Sources** from the main menu, or **Add Sources** from the popup menu, or from the Flow Navigator.

   The Add Sources wizard opens, as shown in Figure 3-14, page 53.

2. Select the **Add Existing IP** option, and click **Next**.

   The Add Existing IP dialog box opens, as shown in Figure 3-24, page 67.
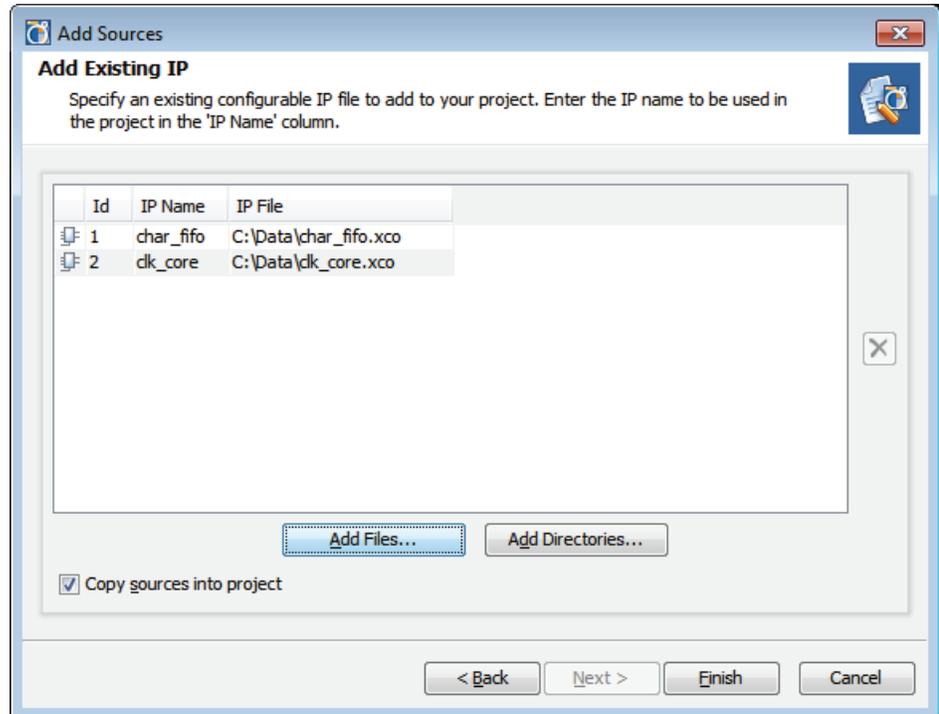
Figure 3-24: **Add Existing IP**

The Add Existing IP options are:

- **Add Files**: Invokes a file browser so you can select XCO files to add existing IP cores to the project.

- **Add Directories**: Invokes a file browser to select a directory of XCO files to add to the project.

- **Remove**: Removes the selected source files from the list of files to be added.

- **Copy Sources into Project**: Copies the original IP core files into the project and uses the local copied version of the file in the project.

When you have specified the existing IP core files to add to the project, click **Finish** to add the cores.

The added IP cores display separately in the IP Sources tab of the Sources view, as well as with other source files in the Hierarchy, Libraries, and Compile Order tabs. You can select these cores in the Sources view to see the files that make up the core, and to view the properties in the Source File Properties view. See Using the Sources View, page 128.

You can also add EDIF netlists or NGC files for IP Cores into either RTL or netlist-based projects. See Creating a Post-synthesis Project, page 46 for more information.

Adding IP from the Xilinx Catalog

To open the IP Catalog, select **IP Catalog** under the Project Manager menu in Flow Navigator.

The IP cores display by category in an expandable tree table, that provides the IP version, Advanced eXtensible Interface (AXI) protocol compliance, status of the IP, and license requirements. When you select a specific IP core, a description displays in the lower pane of the view. Figure 3-25, page 68 shows an example of the IP Catalog view.

You can select an IP core from the catalog and view a variety of information regarding that core. To bring up a PDF datasheet for a selected IP, you can:

- Click **Data Sheet** from the popup menu.

- Click the **View Information** button on the IP Catalog toolbar and select the **Data Sheet** command from that popup menu.
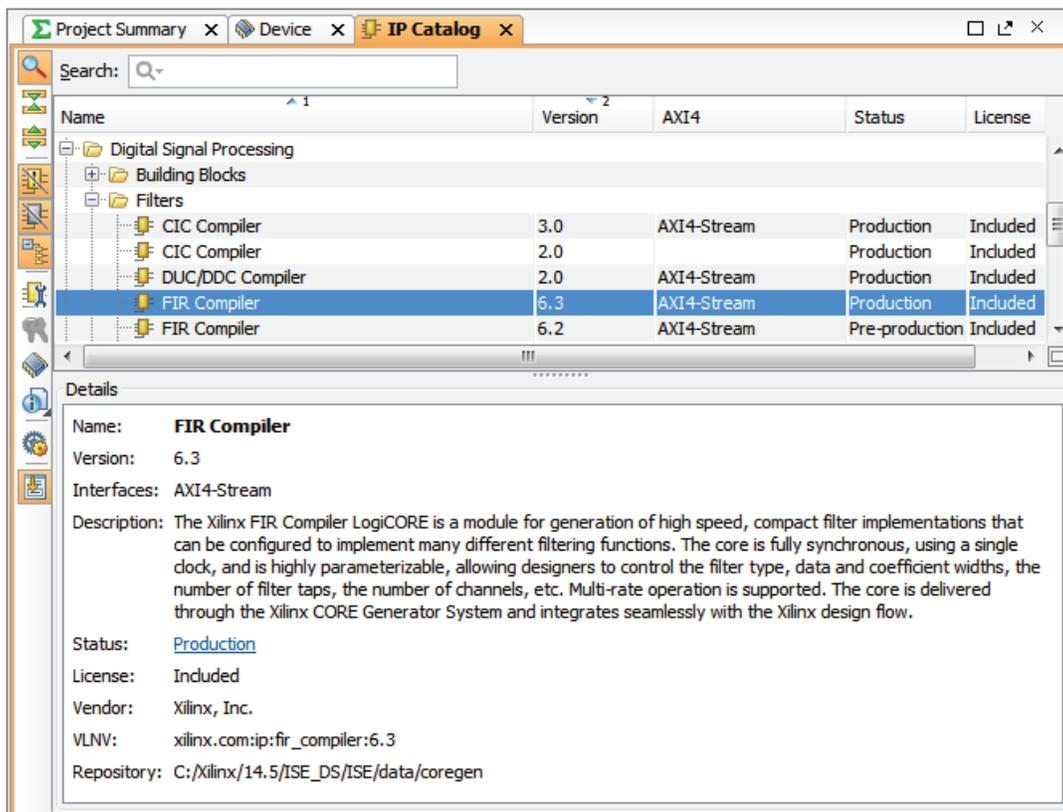
Figure 3-25: **Xilinx IP Catalog**

Command options available in the IP Catalog toolbar and/or popup menu include:

- **Show Search**: Displays a Search field to search the catalog for any text string.

- **Collapse/Expand All**: Collapses or Expands the IP Catalog tree.

- **Hide Superseded and Discontinued IPs**: Filters the list to show current IP only.

- **Hide incompatible IP**: Filters the list to show only the IP that is compatible with the selected device family.

- **Group by Category**: Groups or flattens the list for better sorting and searching.

- **Customize IP**: Opens the customization GUI for the selected IP. See Customizing IP for the Project, page 69 for more information.

- **License Status**: Displays license requirements and status for the selected IP.

- **Compatible Families**: Displays a list of all device families, and specific Xilinx parts, that are compatible with the selected IP.

- **View Information:** Displays a menu of available information resources for the selected IP. Available information includes Data Sheet, Version Information, Webpage, and Answer Records.

- **IP Settings**: Opens the Project Settings dialog box to the IP Settings tab to let you configure various options related to the IP catalog. See IP Settings, page 95.

- **Automatically Scroll to Selected Objects**: Toggles the display to jump to the selected object in the open view.
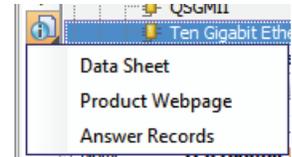
Customizing IP for the Project

You can select a core from the IP Catalog, and customize the IP for use in your design by specifying values for the various parameters associated with the IP core. The PlanAhead tool customizes IP using the integrated CORE Generator tool.

1. Select the IP in the catalog to customize.

2. Select the **Customize IP** command from the toolbar or popup menu, or double-click the selected IP.

The PlanAhead tool invokes the CORE Generator tool interface to enable core generation. The IP type you select determines what type of interface displays. The interfaces are:

- Memory Interface Generator (MIG) wizard

- CORE Generator software wizard, as seen in Figure 3-26.
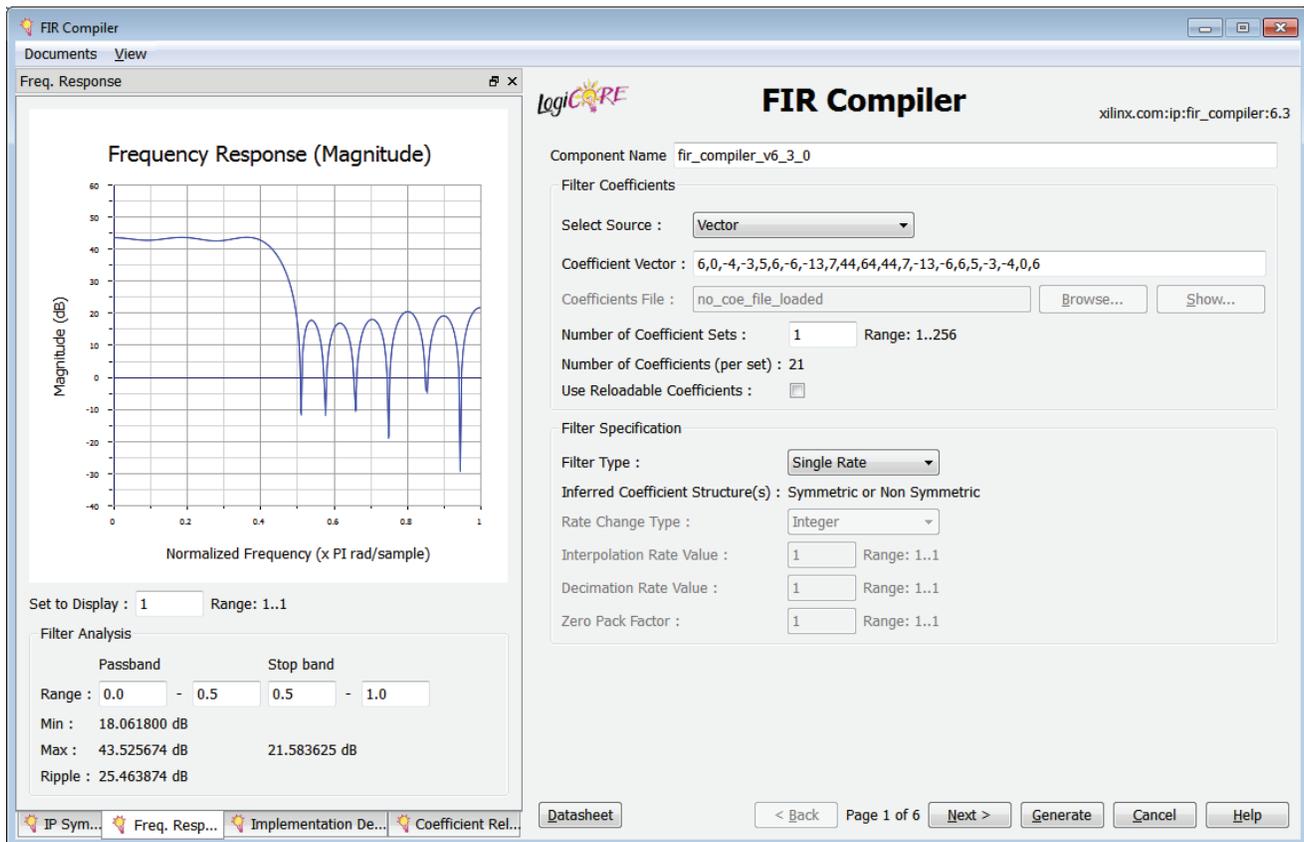

Figure 3-26: **CORE Generator Interface**

The interface fields define the parameters of the IP that allow you to customize the IP for use in your design. The CORE Generator interface varies, depending on the type of core you have selected, and can include one or more pages of parameters to define.

The CORE Generator interface also includes the IP symbol, the Frequency Response graph, and Implementation Details related to resource consumption. You can toggle through these different view by selecting the appropriate tab the bottom left of the interface.

When you have completed defining the various parameters of the interface, click **Generate** to create the customized IP core and add it as a source into the project. The core is not synthesized at this time.

The added IP cores display separately in the IP Sources tab of the Sources view. IP cores display with other source files in the Hierarchy, Libraries, and Compile Order tabs of the Sources view. You can select these cores in the Sources view to see the files that make up the core, and to view the properties in the Source File Properties view. See Using the Sources View, page 128.

- For information on IP, refer to the IP Documents in Appendix E

- For information on using the CORE Generator tool to generate IP, refer to: http://www.xilinx.com/tools/coregen.htm.

- For information on using MIG refer to: http://www.xilinx.com/products/intellectual-property/MIG.htm.

Generating Output Products

IP cores added to the project display in the Sources view. Click the **IP Sources** tab to see the IP and collected output products displayed. Expanding an IP core in the Sources view displays the various files associated with the core.

Output products are the different design elements of the IP required to support the core in the current project. These include the instantiation template, the synthesized netlist, simulation files, and any supporting documents such as log files and datasheets. The source files for the IP (`.xco`) are also displayed in the Hierarchy tab of the Sources view.
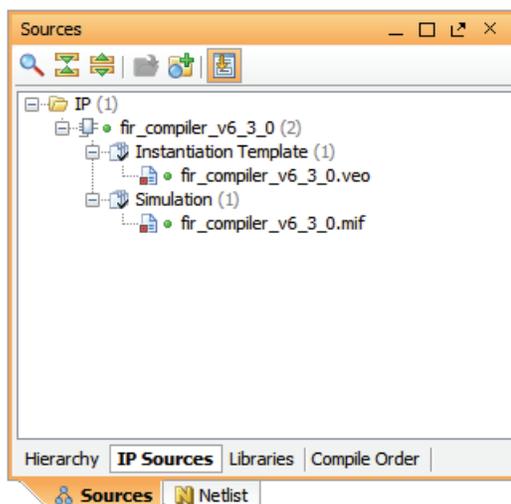
Figure 3-27: **IP Sources**

The output products of an IP core are generated automatically as it is needed in the design flow. By default, when you import IP into the design, the PlanAhead tool will deliver an instantiation template only.

Instantiating IP

The Instantiation Template is the Verilog (`.veo`) or VHDL (`.vho`) module definition that you can copy and paste into your RTL design to create instances of the IP module as needed. Figure 3-28, page 71 shows an instantiation template for an IP core.

Figure 3-28: **IP Instantiation Template**

To instantiate the IP into the design:

1. Open the Verilog or VHDL template file for the IP core and the RTL design file in the Sources view by double-clicking on the file, or by selecting the files and using **Open Files**.

2. Select the Instantiation Template at the **"Begin Cut Here..."** line, and copy it to the appropriate RTL file to create an instance of the module in the design.

3. Edit the RTL as needed to integrate the IP module into your design.

4. With the IP core properly instantiated into the design, you are ready to synthesize the IP core along with the rest of your design.

Synthesizing IP

The generation of required synthesis files is deferred until needed by the tool. With IP cores added to your project, when you run synthesis, the PlanAhead tool automatically synthesizes the IP cores in the project first, then synthesizes the top-level design. This lets you instantiate multiple IP cores without having to synthesize each one as it is added to the project, and also bundles synthesis to save time.

You can also synthesize a specific IP core at any time by selecting the IP in the Sources view, and running **Generate Output Products** from the popup menu. This opens the Manage Output Products dialog box as shown in Figure 3-29, page 72, and lets you specify which output products you would like to generate.
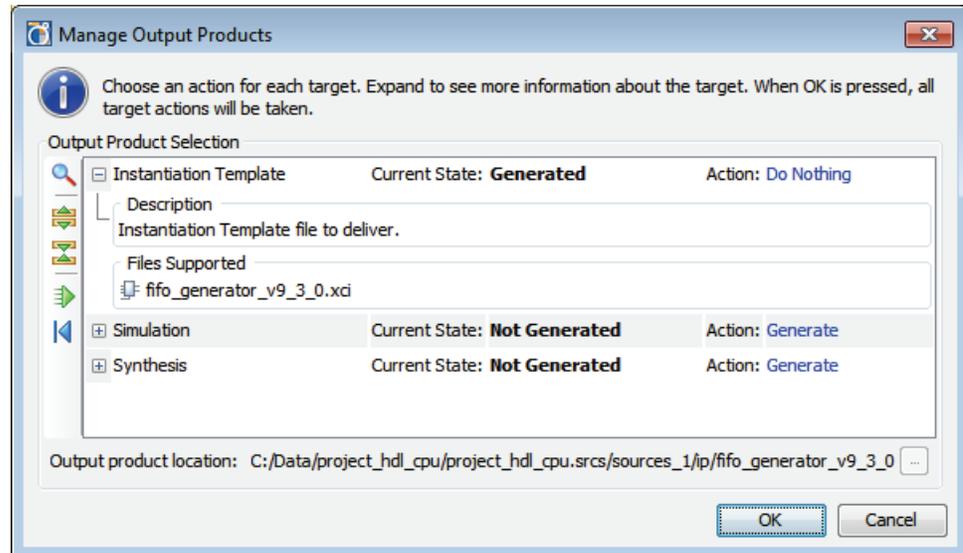
Figure 3-29: **Generate Output Products**

When you generate the synthesis output products, the Xilinx® Synthesis Tool (XST) runs on the core and creates the logic content based on the customization settings.

After the IP is generated, a check mark appears on the IP source icon in the Sources view, and the PlanAhead tool adds the synthesized netlist files (NGC) to the Sources view.
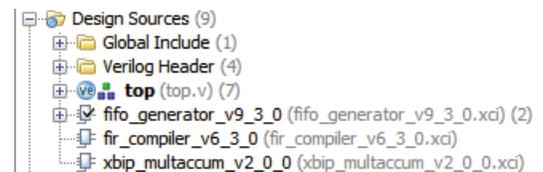
Resetting IP

You can reset an IP at any time to eliminate the output products from that IP, or re-customize the IP to change its definition in the current design.

To delete the current data for an IP core, select it in the IP sources tab of the Sources view, and select the **Reset Output Products** command from the popup menu. The Reset Output Products command lets you select the outputs to reset, or reset all outputs. This will eliminate the current output data, and require you to generate the output products again.

To change the definition of the IP in the current design, select the IP in the Sources view and use the **Re-customize IP** command from the popup menu. This will reopen the CORE Generator tool interface and let you change any of the parameters associated with the core for this project. When you have made the necessary changes, select the **Generate** command in the CORE Generator dialog to add the IP back into the design, and generate the Instantiation Template.

You can also upgrade IP in the current project to the latest version of the IP in the catalog. Use this feature to bring any changes in the catalog into the current project. Select the IP in the Sources view, and select the **Upgrade** command. This will update the customized IP in the design to the latest version from the Xilinx IP Catalog, and reapply any customization from the current project.

Managing DSP Sources

The PlanAhead tool lets you import an existing System Generator design model file (`.mdl`) as a DSP module. The model can be added to any hierarchy level as a sub-module or imported at the

top-level of the design. You can also define a new DSP module from within the PlanAhead tool and launch System Generator to complete the design.

System Generator (Sysgen) is a DSP design tool from Xilinx that allows the RTL source files, Simulink® and MATLAB® software models, and C/C++ components of a DSP system to come together in a single simulation and implementation environment. Refer to System Generator for DSP Documentation in Appendix E for more information on these tools.

A System Generator design is often a sub-design that is incorporated into a larger HDL design. While Sysgen supports creating and implementing standalone FPGA designs, the recommended flow is to begin with a project in the PlanAhead tool, and develop a DSP module source for the project using Sysgen. This allows the PlanAhead tool to manage the project for the FPGA design, while handling the DSP module as a single source file, that is developed and managed within Sysgen.

Adding DSP Modules

PlanAhead allows you to import existing Model files (`.mdl`) files from Sysgen, or to define new DSP modules within the PlanAhead tool and open Sysgen to create and manage the project.

To add or create DSP modules:

1. Select **File > Add Sources** from the main menu, or **Add Sources** from the popup menu, or from the Flow Navigator.

    The Add Sources wizard opens, as shown in Figure 3-14, page 53.

2. Select the **Add or Create DSP Sources** option, and click **Next**.

The Add or Create DSP Sources dialog box opens, as shown in Figure 3-30.
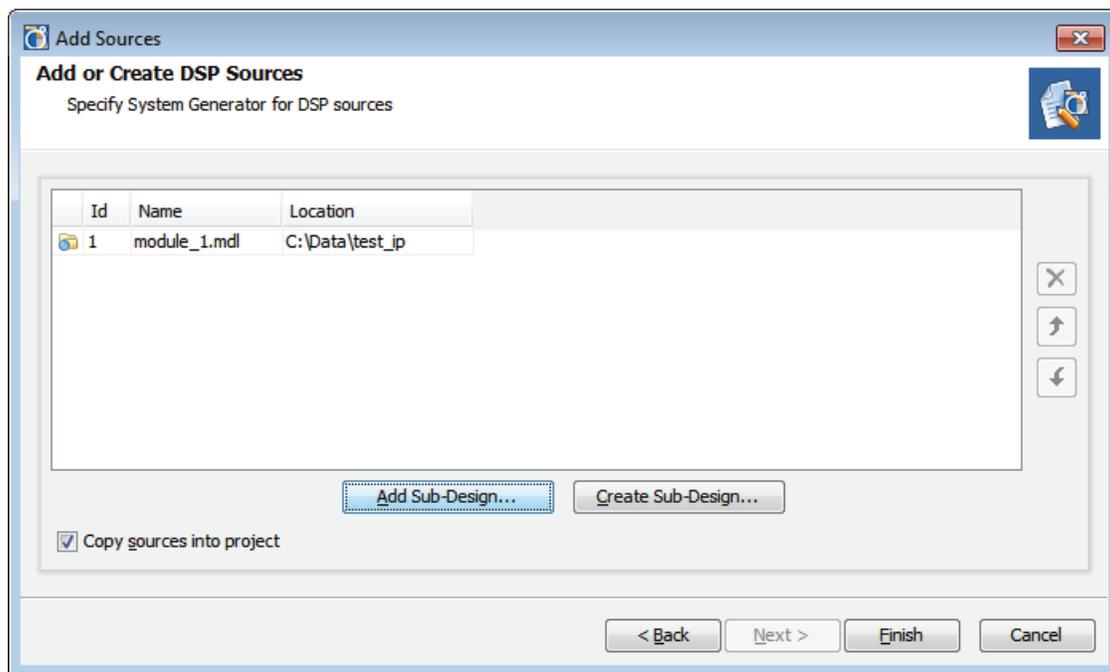
Figure 3-30:   **Add DSP Sources**

- **Add Sub-Design:** Add an existing DSP source into the current project. You will be presented with a file browser to navigate to existing Sysgen model file (.mdl) files to add.

- **Create Sub-Design:** Create a new Sysgen model to add to the PlanAhead tool project. This will launch Sysgen to allow you to define the new DSP module.

- **Remove**: The 'X' icon removes the selected DSP source from the list of files to be added.

- **Move Up:** The upward pointing arrow icon moves the selected source up in the list order.

- **Move Down:** The downward pointing arrow icon moves the selected source down in the list order.

- **Copy Sources into Project**: Copies the Sysgen model file (.mdl) file into the local project directory.

When you have specified the DSP sources to add or to create, click **Finish** to add the files. Sysgen and MATLAB are started to create and manage the DSP source files, as shown in Figure 3-31.
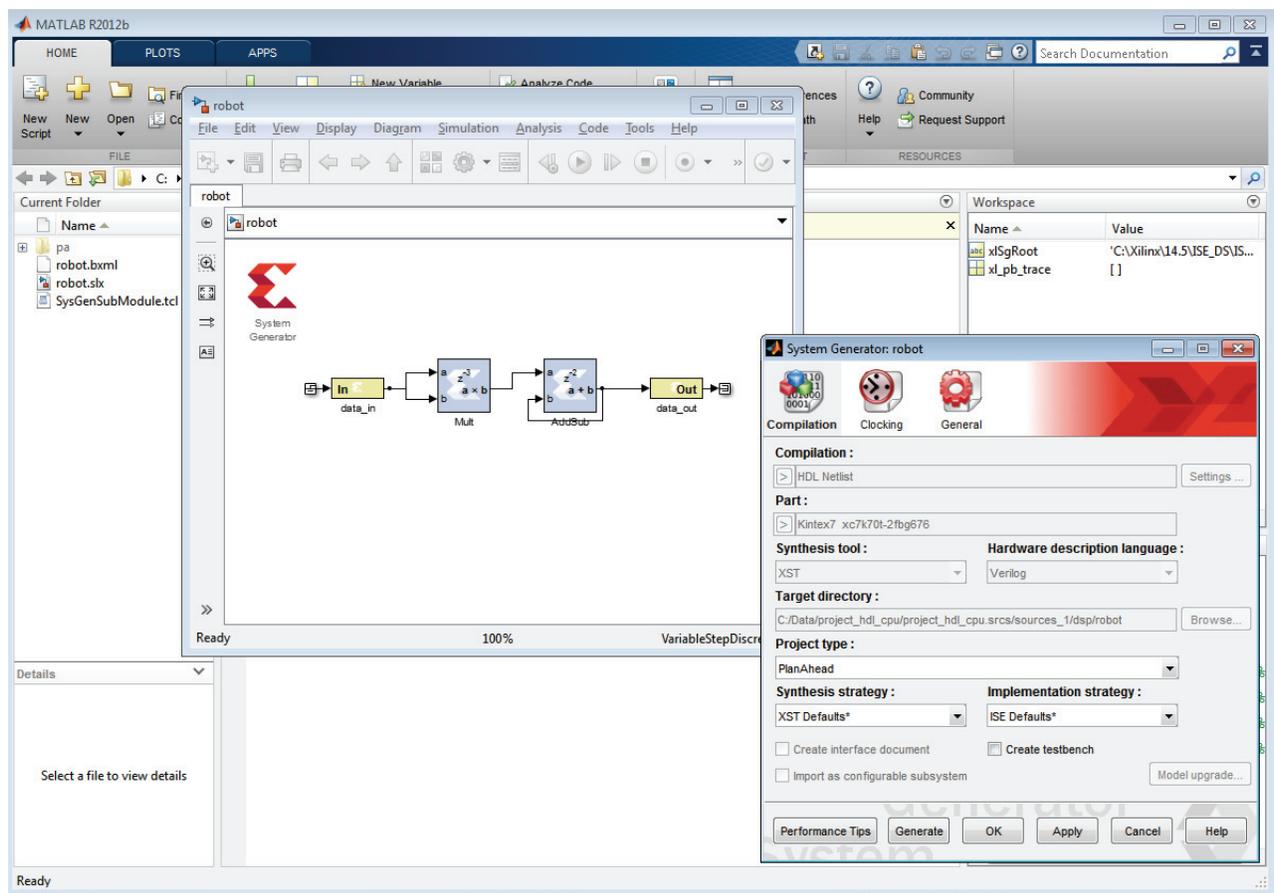
Figure 3-31: **Generating DSP Sources in Sysgen**

The added DSP design sources display separately in the IP Sources tab of the Sources view. DSP sources display with other source files in the Hierarchy, Libraries, and Compile Order tabs of the Sources view. You can select the DSP modules in the Sources view to see the associated files, and to view the properties in the Source File Properties view. See Using the Sources View, page 128.

Generating Targets

Once the Sysgen design is completed, the FPGA target files can be generated in the PlanAhead tool using the DSP module commands in the Sources view popup menu. The DSP module commands are available in the Sources view popup menu when a DSP source has been selected.

Targets are the different design elements of the DSP module that are required to support synthesis, simulation, and implementation of the current project. These include the a top module definition, an Instantiation Template, the synthesized netlist, and any supporting documents.
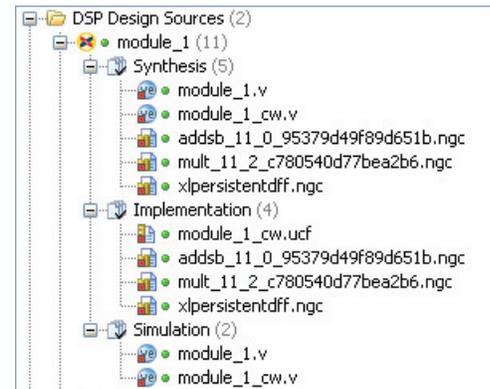
- • **Create Top HDL:** Create a top-level wrapper for the DSP module, and import it into the project. Use this command when the Sysgen design is the top level of the current project.

- • **View Instantiation Template:** Create an instantiation template to use for instantiating the DSP module into the RTL design. The Instantiation Template can be cut and paste into another RTL file to create an instance of the DSP module in the hierarchy.

- • **Create Testbench:** Sysgen will write test vector files that are extracted from the Simulink simulation and generate an HDL testbench and script files for simulation. The test bench will be added to the Sources view in a simulation fileset.

- • **Generate Output Products**: Generate the synthesis, implementation, and simulation target data from the Sysgen model. This invokes Sysgen and MATLAB to create the necessary data.

- • **Reset Output Products**: Remove the specified target data from the current project and from the local project repository so that it can be regenerated as needed.

Managing Embedded Processor Sources

The Embedded Development Kit (EDK) is a suite of tools and IP that you can use to integrate your hardware and software system components. EDK includes Xilinx Platform Studio (XPS) and Software Development Kit (SDK).

You can use the Xilinx Platform Studio (XPS) to design the hardware portion of your embedded processor system. Specification of the microprocessor, peripherals, and the interconnection of these components, along with their respective detailed configuration, takes place in Xilinx Platform Studio (XPS). For more information on using specific features of XPS refer to EDK Documentation in Appendix E.

While the EDK environment supports creating and implementing designs, the recommended flow is to begin with a project in the PlanAhead tool, and develop an embedded processor source for the project using XPS. This allows the PlanAhead tool to manage the project for the FPGA design, while handling the embedded processor design as a single source file, that is developed and managed within XPS. Figure 3-32, page 76 shows the integrated embedded design flow.

Typical Embedded Flow
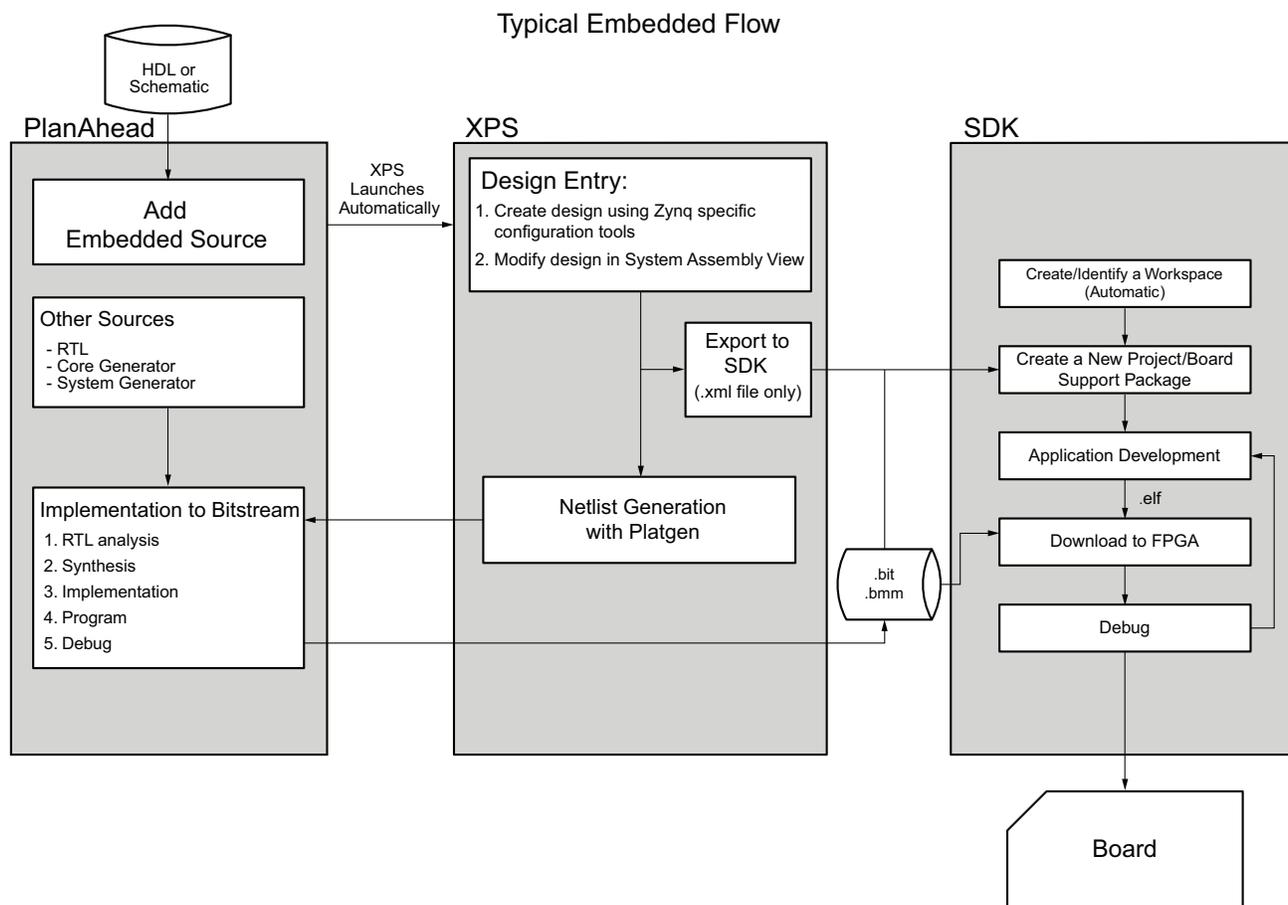
X10254

Figure 3-32: **Embedded Design Flow**

Adding Embedded Processors

PlanAhead allows you to import existing Xilinx Microprocessor Project (.xmp) files from XPS, or to define new embedded processor sub-designs within the PlanAhead tool and open XPS to create and manage the project.

To add or create XPS projects:

1. Select **File > Add Sources** from the main menu, or **Add Sources** from the popup menu, or from the Flow Navigator.

    The Add Sources wizard opens, as shown in Figure 3-14, page 53.

2. Select the **Add or Create Embedded Sources** option, and click **Next**.

The Add or Create Embedded Sources dialog box opens, as shown in Figure 3-33, page 77.
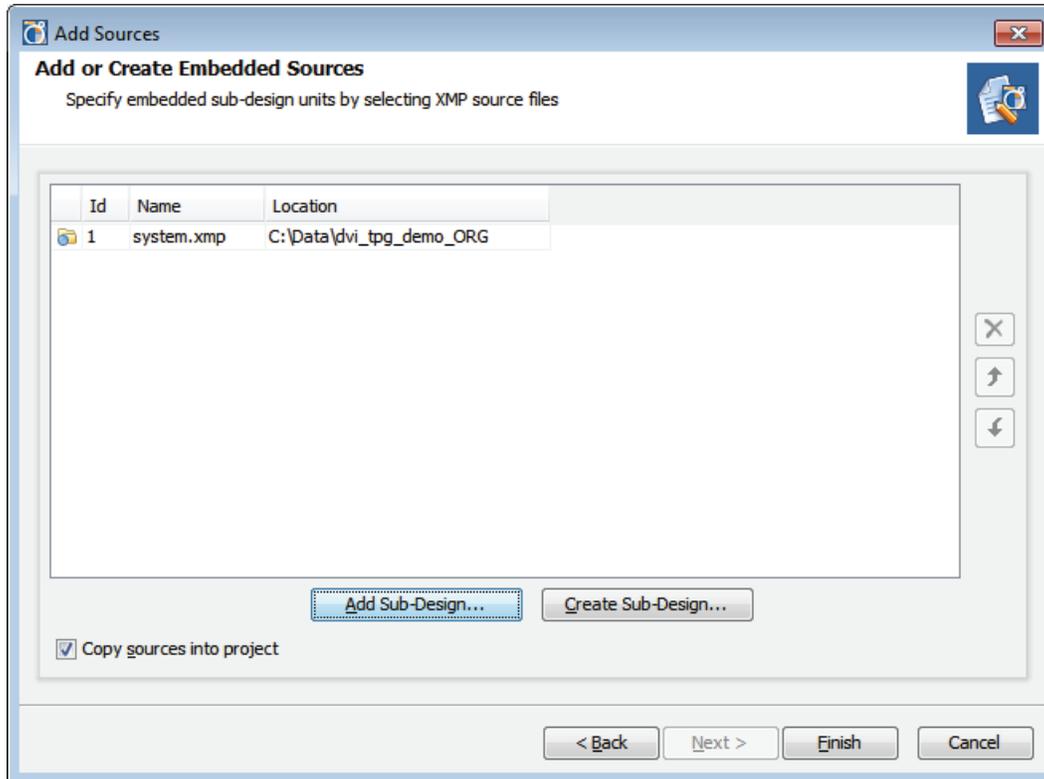
Figure 3-33: **Add Embedded Sources**

- **Add Sub-Design**: Add an existing XPS project into the current project. You will be presented with a file browser to navigate to existing Xilinx Microprocessor Project (.xmp) files to add.

- **Create Sub-Design:** Create a new XPS project to add to the PlanAhead tool project. This will launch XPS to allow you to define the new sub-design.

- **Remove**: The 'X' icon removes the selected sub-design from the list of files to be added.

- **Move Up:** The upward pointing arrow icon moves the selected sub-design up in the list order.

- **Move Down**: The downward pointing arrow icon moves the selected sub-design down in the list order.

- **Copy Sources into Project**: Copies the embedded sources (.xmp) file into the local project directory.

When you have specified the XPS projects to add or to create, click **Finish** to add the files.

The added sub-designs display separately in the IP Sources tab of the Sources view, as well as with other source files in the Hierarchy, Libraries, and Compile Order tabs. You can select these sub-designs in the Sources view to see the files that are associated with the module, and to view the properties in the Source File Properties view. See .

Creating a Sub-Design

If you selected **Create Sub-Design** in the Add or Create Embedded Sources dialog box, XPS will be launched to let you define the new embedded sub-design. Attributes of the project in the PlanAhead tool, such as target part or TDP, are automatically passed to XPS as it opens.

The following is a brief overview of the process of defining the embedded design. For more information, refer to *EDK Concepts, Tools, and Techniques (UG683)* and *Embedded System Tools*

Reference Manual (UG111) as referenced in EDK Documentation in Appendix E.

XPS recognizes that this is a new sub-design and prompts you to use the Base System Builder wizard to help design the board for the design.

1. Click **Yes** and continue.

   The Base System Builder (BSB) wizard opens as shown in Figure 3-34. The BSB wizard helps you quickly build a working system. Some embedded design projects can be completed using the BSB wizard alone. For more complex projects, the BSB wizard provides a baseline system that you can then customize to complete your embedded design.
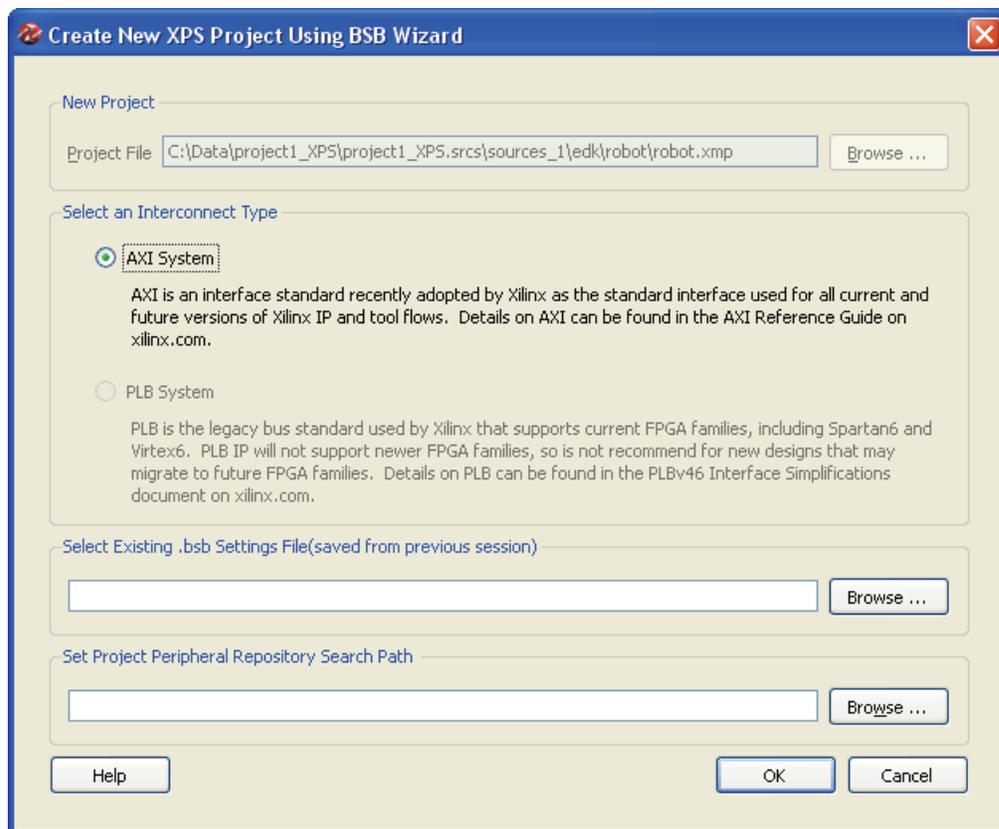
Figure 3-34: **Base System Builder**

Much of the form is filled in with data from the current project, and cannot be modified in order to protect the integration of the PlanAhead project with the XPS project.

- **Project File:** Indicates the name of the sub-design specified in the Create Sub-Design dialog box. This name comes from the PlanAhead tool.

- **Select an Interconnect Type:** Specifies the AXI System. This is a hard coded selection, to avoid the legacy system.

- **Select Existing .bsb Settings File:** Specify a previous BSB settings file to automatically apply the same selections into to this BSB session.

- **Set Project Peripheral Repository Search Path:** Specify user repositories containing custom pcores, Board Support Packages (BSPs), and Software Services. If more than one repository search path is specified, they must be separated by a semi-colon (;).

2. Click **OK** to continue.

The Board and System Selection dialog box opens as shown in Figure 3-35. This dialog box defines the TDP or platform for the embedded processor design.

The definition of the TDPs offered is limited to those containing the target part that you selected for the project in the PlanAhead tool. From the target development board, the BSB is able to determine what devices are on the target board, such as the specific FPGA device, external memories, I/O devices, clock resources, and reset polarity. The following wizard pages are customized based on the information that can be retrieved for the selected board, minimizing the amount of input that is required.
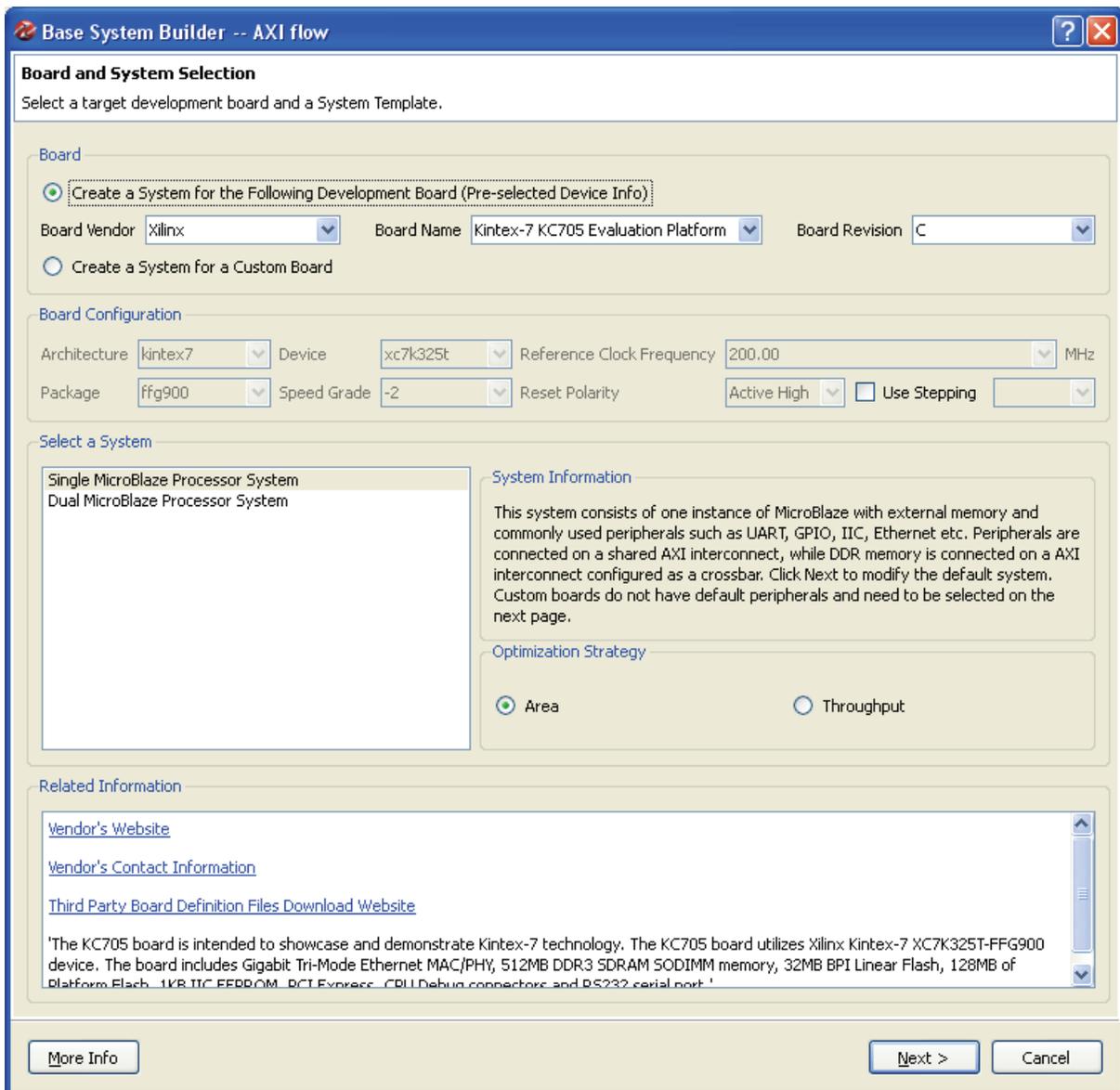
Figure 3-35: **Board and System Selection**

If the selected FPGA is not featured on any of the supported boards, then you will receive a message indicating that no boards for the device were found, and you must select **Create a System for a Custom Board**.

3. Click **Next** to continue.

The Processor, Cache, and Peripheral Configuration dialog box opens as shown in Figure 3-36. This dialog lists the peripheral that are available on the specified TDP. You can select which of the available peripherals you want to include in your embedded design.
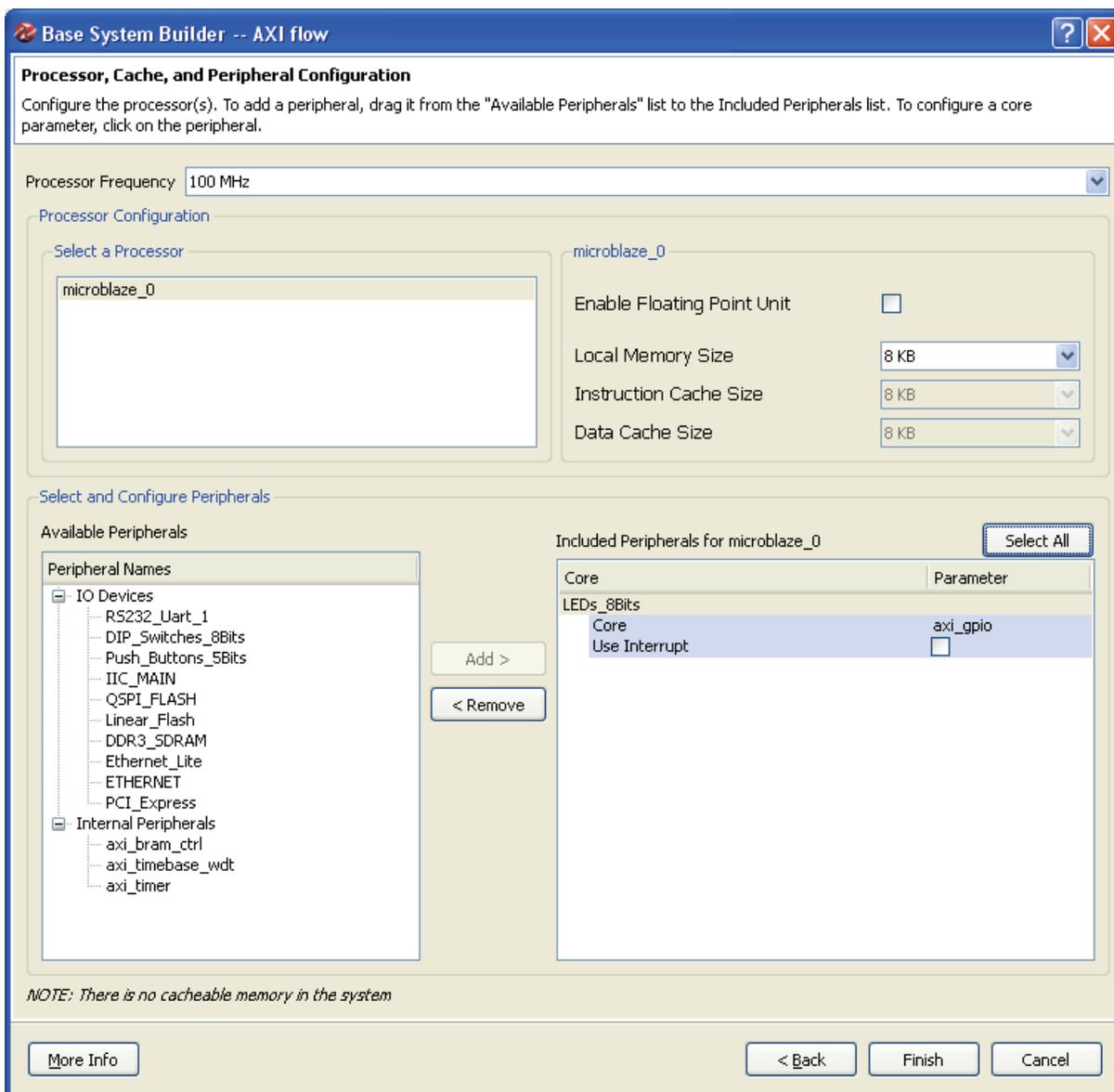
Figure 3-36: **Processor, Cache, and Peripheral Configuration**

4. Click **Finish** to create the embedded design and open the project in XPS.

    You can edit and manage the embedded processor sub-design in the XPS tool.

Generating Output Products

When you exit the XPS tool, the top-level project design file
(`.xmp`) and the Microprocessor Hardware Specification file
(`.mhs`) are added in the Sources view in the PlanAhead tool.
Expanding the Embedded Design Sources in the Sources view
displays the various output files associated with the sub-design.

```
☐ 🗁 Embedded Design Sources (2)
  ☐ 🗹 ● robot1 (1)
      ☐ 🗁 Non-target Files (1)
          └── 📄 ● robot1.mhs
  ☐ 🗹 ● chicken (1)
      ☐ 🗁 Non-target Files (1)
          └── 📄 ● chicken.mhs
```
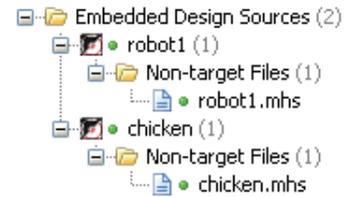
Targets are the different design elements of the XPS sub-design
that are required to support the object in the current project.
These include the a top module definition, an Instantiation Template, the synthesized netlist, and
any supporting documents such as log files and datasheets. The project file for the embedded design
(`.xmp`) is displayed in the Hierarchy tab of the Sources view.

The popup menu commands for Embedded Sources in the Sources view are:

- **Create Top HDL:** Create a top-level wrapper for the embedded design, and import it into the
  project. Use this command when the embedded design is the top level of the current design.

- **View Instantiation Template:** Create an instantiation template to use for instantiating the
  embedded design module into the RTL design. The Instantiation Template can be cut and paste
  into another RTL file to create an instance of the sub-design in the hierarchy.

  Note: The template file will not be added to the project.

- **Create Testbench:** Create a testbench for the embedded design. The test bench will be added
  to the Sources view in a simulation fileset.

- **Generate Output Products:** Create the specified output data for Synthesis, Implementation,
  or Simulation. The target data includes the Verilog or VHDL files, the wrapper files, the BMM
  model, and the top-level simulation model for the sub-design.

- **Reset Output Products:** Remove the specified output data from the current project. This also
  removes the generated output data from the local project repository so that it can be
  regenerated as needed.
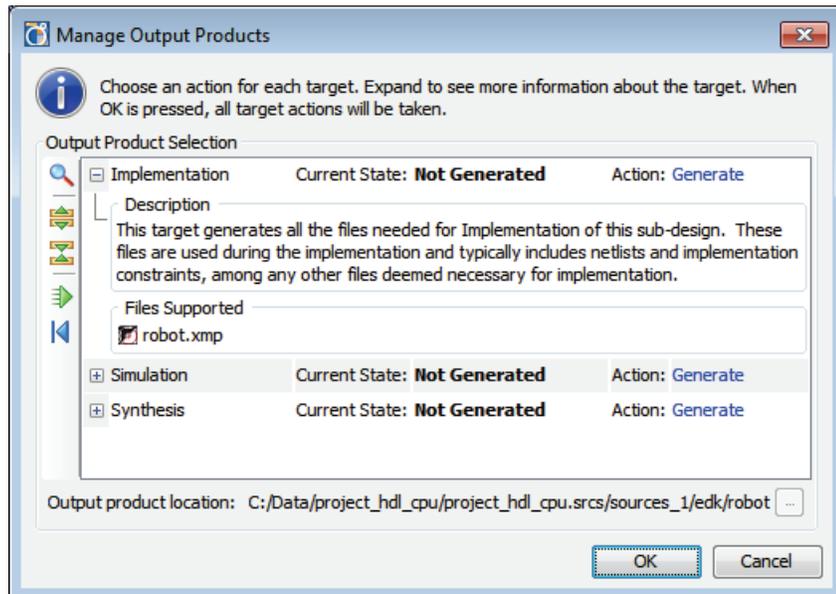
Figure 3-37: **Generate Embedded Source Output Products**

When you generate the output products data, the `/synthesis` and `/implementation` subdirectories are created for the Embedded sub-design. This also occurs when you use either the **Hardware > Generate Netlist** or **Hardware > Generate Bitstream** commands in XPS.

If you added an existing XPS embedded design source, `/synthesis` and `/implementation` are subdirectories of the embedded design, external from the current project. However, if you used the **Create Sub-Design** command to add the embedded processor design to the current project, the `/synthesis` and `/implementation` subdirectories will be local to the project directory in: `<project>`.srcs\sources_1\edk\`<subdesign_name>`

- The `/synthesis` subdirectory contains all the XST synthesis scripts (`.scr`), project files (`.prj`) and report (`.srp`) files that create the netlists used for implementation.

- The `/implementation` subdirectory contains a copy of the User Constraints file (UCF), Block RAM Memory Map (BMM) files for configuring the block RAMs on the device, and the implementation results including the BIT file.

Implementing the Embedded Processor

The following MAP options should be set when running implementation on the top-level design:

- `-pr b`: pack the internal flops into input (i), output (o), or both (b) types of IOBs.
- `-ol high`: overall effort level is set to high.
- `-timing`: perform a timing-driven packing.

Set these options in the Implementation Settings for the implementation run to ensure Processor IP cores will function correctly. Refer to Setting Implementation Options in Chapter 9 for more information.

Exporting Hardware

The PlanAhead tool is also integrated with SDK to support the design of software for the embedded processor sources in your project. To develop software for your project using SDK, with the embedded processor in your design, select **File > Export > Export Hardware**.

The Export Hardware dialog box opens, as shown in Figure 3-38.
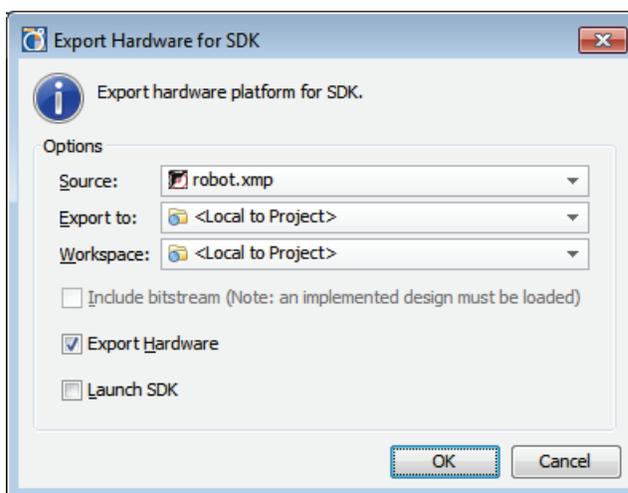
Figure 3-38:   **Export Hardware**

Specify the source XPS project file to export, and the location to export the hardware to. By default the hardware files will be written to the local project directory, under:

`<project>`.sdk/SDK/SDK_Export/hw

Select the **Export Hardware** check box. This will generate the files necessary to support software development for the embedded processor design.

Select the **Launch SDK** check box to optionally launch the SDK tool after the hardware files are generated.

Click **OK**.

The PlanAhead design tool exports the Hardware Platform Specification (`system.xml`) for your design, and opens the file in SDK, if you selected Launch SDK. Refer to SDK Help for more information.

Creating the Bitstream File

In order to boot up an embedded processor system, both hardware and software components of the system must be downloaded to the FPGA and program memory respectively. This requires creating a bitstream file in BitGen that contains the software application targeted for the block RAM. You can download the hardware bitstream created by BitGen, and the software Executable and Linkable Format (ELF) file associated with the embedded processor. You can then use the iMPACT tool from the Flow Navigator to program the FPGA with the bitstream.

You can add or update the ELF files associated with a processor instance using the **Tools > Associate ELF Files** command, from the main menu.

The Associate ELF Files dialog box opens as shown in Figure 3-39. This lets you specify the ELF file to use when generating the bitstream file.
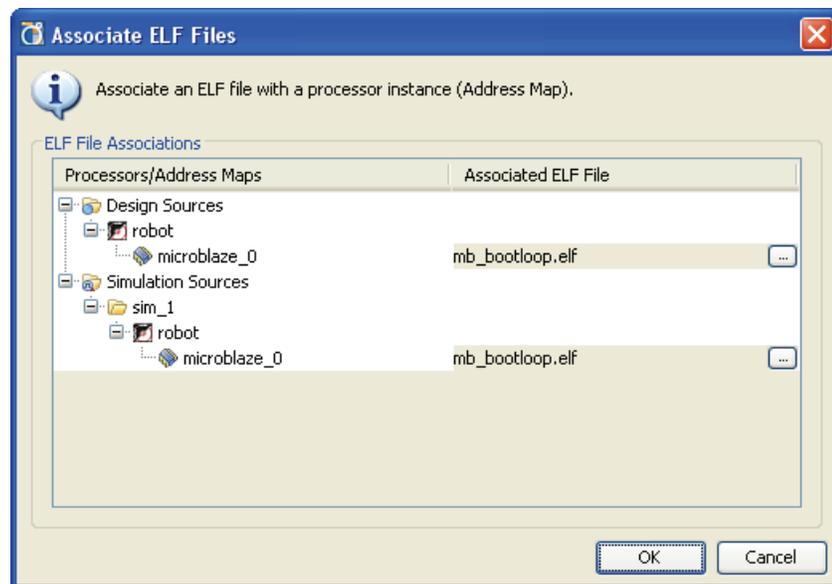
Figure 3-39: **Associate ELF Files**

The ELF file populates the Block RAMs specified in the `.bmm` file. The bit file created by PlanAhead now has the block RAM initialized with the selected executable code.

The Bitinit tool initializes the on-chip block RAM memory connected to a processor with its software information. This utility reads hardware-only bitstream produced by the ISE tools (`system.bit`), and outputs a new bitstream (`download.bit`) which includes the embedded

application executable (ELF) for each processor. Refer to EDK Documentation in Appendix E for more information.

Using the Project Summary view

The PlanAhead tool provides an interactive Project Summary view that details information about the design and the project. The Project Summary view updates dynamically as design commands are run and as the design progresses through the design flow.

The Project Summary displays design information like project part, project status, state of synthesis and implementation, resource estimates, and timing results. The Project Summary view also provides links to launch commands and to display more detailed information. You can use the scroll bar or the **Collapse and Expand** buttons to view or hide the different data categories.

You can open the Project Summary using one of these methods:

- Select **Project Summary** from the Project Manager menu.

- Select the **Project Summary** toolbar button.

- Select **Windows > Project Summary**.

Figure 3-40 shows the Project Summary view.

Project Settings

The Project Settings displays the Project Name, Device Family, Project Part, and Top Module Name. Selecting the **Edit** link invokes the **Project Settings** dialog box. Refer to Configuring Project Settings, page 88.
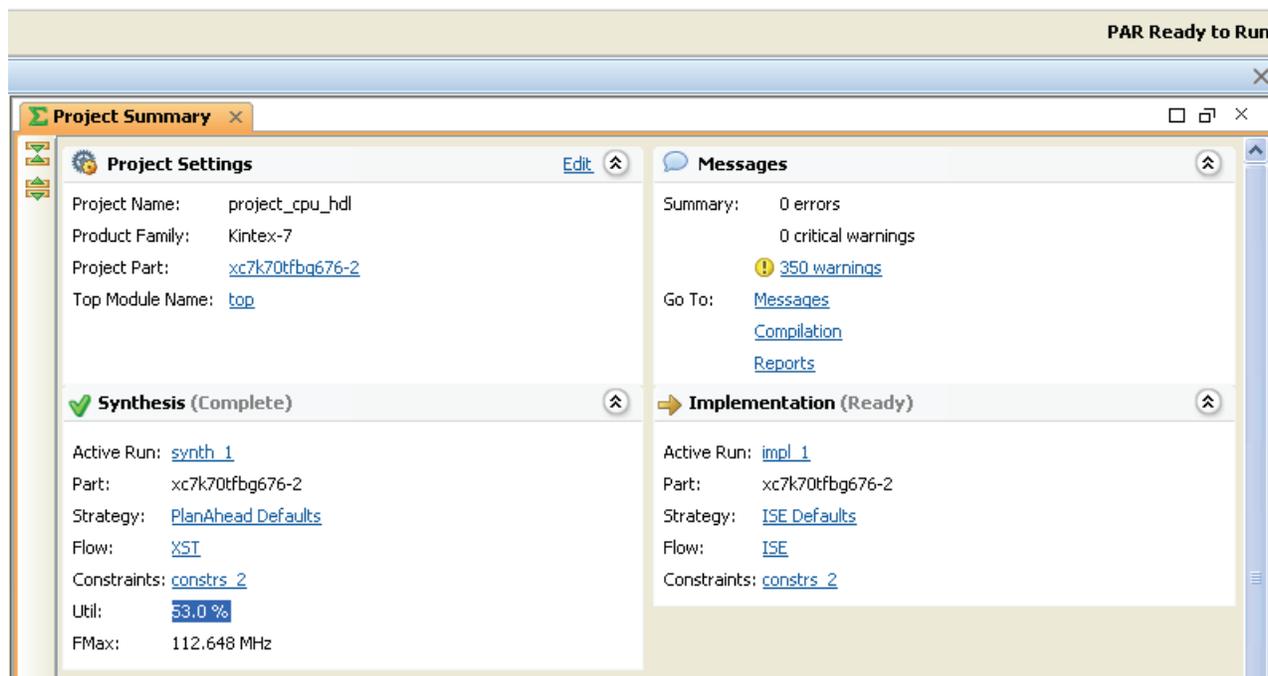
Figure 3-40: **Project Summary**

Messages

The Messages panel contains the following information:

- **Summary:** Summarizes the number of errors and warnings encountered during the design process. There is also a link to open the Messages view that is filtered to show the warnings or errors.

- **Go To**: Provides links to open the Messages, Compilation, or Reports views. Refer to Using the Tcl Console and Messages Area in Chapter 4 for more information.

Synthesis

Summarizes the state of synthesis in the active run. Synthesis shows the target part, the strategy applied in the run, the resource utilization, and the max frequency (Fmax) observed on the device.

Implementation

Summarizes the state of implementation in the active run. Implementation shows the target part, the strategy applied in the run, the tool flow used, and the Constraints set.

Click the target part or strategy links to open the **Project Settings** dialog box for either synthesis or implementation. See Configuring Project Settings, page 88 for more information.

Resources

The resource utilization for the target device displays in either a graphical form as shown in Figure 3-41, or tabular form as shown in Figure 3-42, page 87. Click the link in the upper right of the Resources section of the Project Summary to toggle between the graph and table views.
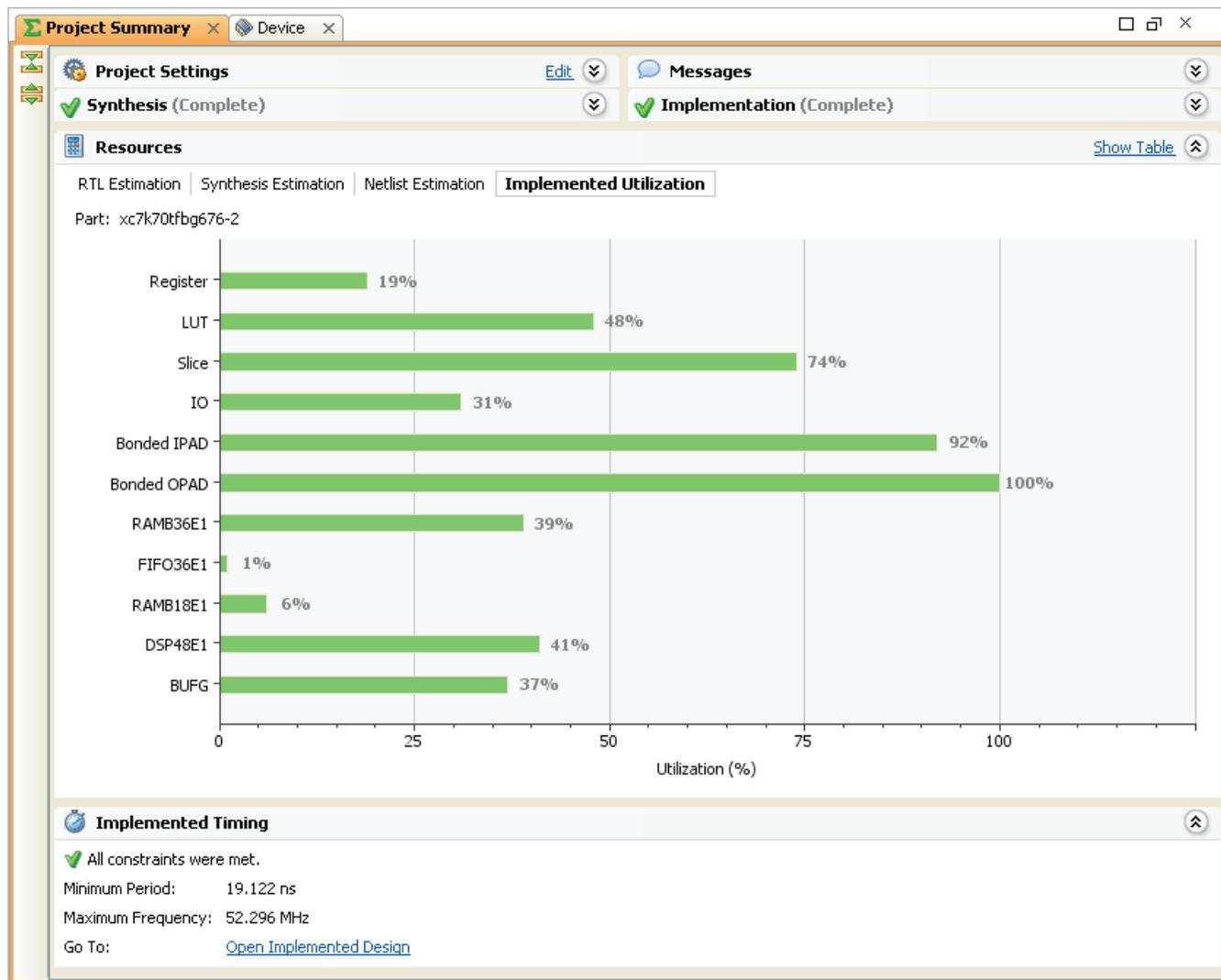
Figure 3-41: **Graphical Resource Estimates**

The PlanAhead tool populates the Project Summary Resources at each stage of the design process. The types of displayed logic objects varies as the design progresses through the design stages. As the information becomes available, the tabs at the top of the view panel become selectable.

- **RTL Estimation**: Provides estimates from the Elaborated Design after the **Estimate Resources** command has run.

- **Synthesis Estimates**: Extracts resource estimates from the XST Synthesis report.

- **Netlist Estimation**: Provides estimates from the Synthesized Design after the **Estimate Resources** command has run.

- **Implemented Utilization**: Extracts actual resource utilization from the Implemented Design.

Figure 3-42: **Tabular Resource Estimates**

In some cases, links display in the Resources panel to guide you through the commands required for populating the Resources chart. For Netlist Estimation a Synthesized Design must be open. For Implemented Utilization an Implemented Design must be open.

Implemented Timing

Implemented Timing summarizes the overall timing results for an implementation run as shown in Figure 3-43. The Timing Score, Minimum Period, Maximum Frequency, and worst Failing Constraint for the active Implementation run display, as well as a link to open the Implemented Design and the Timing Results view. For more information about Timing Analysis, see Chapter 11, Analyzing Implementation Results.

Figure 3-43: **Timing Results in Project Summary View**

Partition Summary

For Implemented Designs that have partitions defined the Project Summary view also provides a Partition Summary to allow you to quickly view the status of all partitions. The Partition Summary provides a post implementation report of the attributes of each partition and values such as State, Preservation Level, and BoundaryOpt.

Refer to Chapter 13, Using Hierarchical Design Techniques, and *Partial Reconfiguration User Guide (UG702)*, cited in Appendix E, Additional Resources for more information on defining and using partitions.

Configuring Project Settings

Each project has specific settings that can be configured to meet specific needs. These project settings include general settings related to the top module definition and language options, simulation settings, synthesis settings, implementation settings, and IP Catalog settings.

The PlanAhead tool provides access to the project settings from a variety of views and menus. Based upon how you invoke the project settings, the dialog box displays the settings. Select **IP Catalog Settings** from the IP Catalog view to open the Project Settings dialog box with the IP Catalog settings displayed.

To open the **Project Settings** dialog box, use the one of the following methods:

- **Project Settings** toolbar button.
- **Project Settings** from the Project Manager menu in the Flow Navigator.
- **Tools > Project Settings** from the main menu.
- Click the **Edit** link in the Project Summary view.

The Project Settings dialog box opens, and displays the following menu on the left-hand side, as shown in Figure 3-44, page 89:

- **General:** Displays a dialog box where you can view the project Name, specify the Top Module Name, and set language options. See General Project Settings, page 89.

- **Simulation**: Displays the Simulation Set, the Simulation Top Module Name, Top Module (Design Under Test), Simulation Runtime, and a tabbed listing of Launch Options, Language Options, and Netlist Options. The currently selected options have a green check mark. See Simulation Settings, page 91.

- **Synthesis**: Shows the target Part and Constraints Set, and provides an Options area for selecting a synthesis strategy and for setting synthesis command-line options. The command-line options are defined by the selected synthesis strategy, but you can override these with your own selections. A description of the selected command-line option displays at the bottom of the dialog box. See Synthesis Settings, page 92.

- **Implementation**: Shows the target Part and Constraints Set, and provides an Options area for selecting an implementation strategy; and for setting command-line options for the translate, MAP, PAR, and timing analysis steps that occur during Implementation. The command-line options are defined by the selected implementation strategy. You can override the setting with your own selections. A description of the selected command-line option displays at the bottom of the dialog box. See Implementation Settings, page 93.

- **Bitstream**: Specify the Bitstream options to use. See Bitstream Settings, page 94.

- **IP:** Configure various options related to the Xilinx IP Catalog. See IP Settings, page 95.
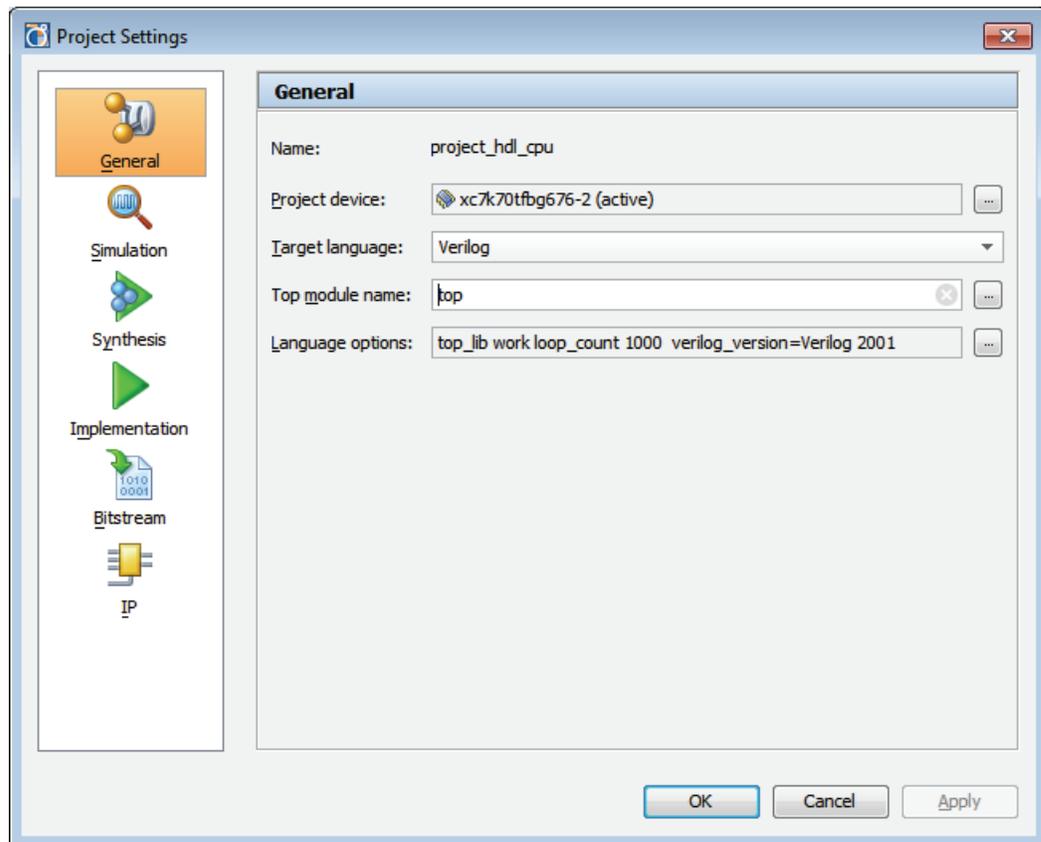
Figure 3-44: **General Project Settings**

General Project Settings

Figure 3-44 shows the General Project Settings with the Language options dialog box:

- **Name**: Displays the project name.

- **Project Part**: Defines the target part to be used as a default for both synthesis and implementation. Select the browser button to invoke the **Part Selector** dialog box to choose another part. See Selecting a Default Part or Board, page 45 for more information on setting the target part.

  The Project Part allows you to quickly define or change the target device for the whole project. However, when there are multiple synthesis or implementation runs, you can also change the part used for a specific run by changing the run settings from the Run Properties view. See Using the Run Properties View in Chapter 4 for more information on setting design run attributes.

- **Target Language**: Specifies the target output language for the design as either Verilog or VHDL. The tool generates RTL output from the design in the specified target language. Specific examples of output controlled by the target language are synthesis, simulation, top-level wrappers, test benches, and IP instantiation templates.

- **Top Module Name**: Enter the top RTL module name of the design. You can enter a lower-level module name to experiment with synthesis on a specific module also. Selecting the **Browse (...)** button will allow the tool to automatically search for the top module, or provide a list of possible top modules when more than one potential top module is found.
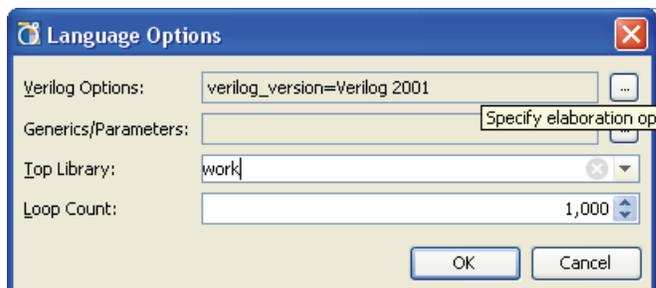
Figure 3-45: **Language Options**

- **Language Options**: Enter specific Verilog and VHDL options. These fields are shown in Figure 3-45.

  - **Verilog Options**: Specify Verilog options as defined below. These options are defined on the Verilog Options dialog box, are shown in Figure 3-46, page 90.

    - **Verilog Search Paths**: Specify the paths to search for files referenced by `include` statements in the source Verilog files.

    - **Defines**: Specify Verilog macro definitions for the project.

    - **Uppercase all identifiers**: Specify that all Verilog identifiers should be uppercase.
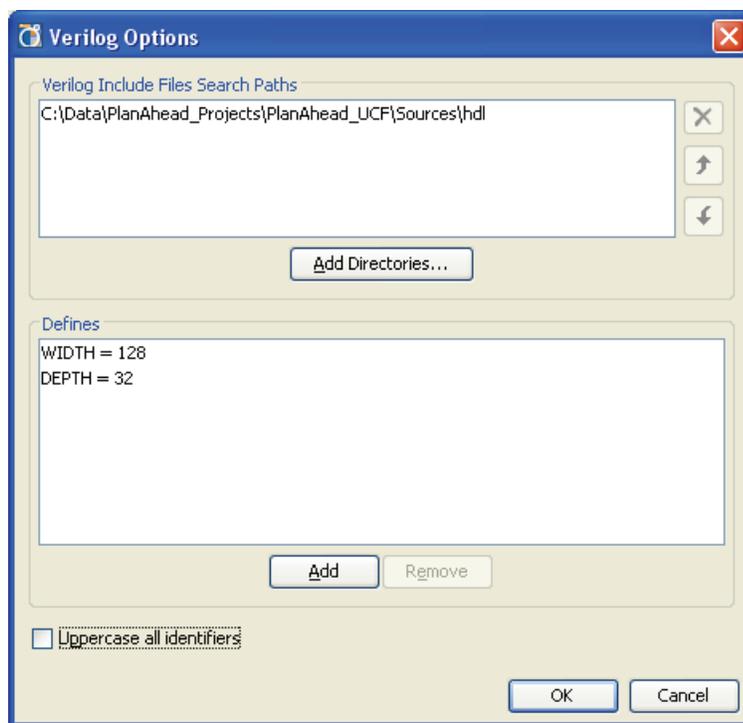
Figure 3-46: **Verilog Options**

  - **Generics/Parameters**: VHDL supports generics, while Verilog supports defining parameters for constant values. Both of these techniques allow parameterized designs that can be re-used in different situations. This option allows you to define generic and parameter values to override defaults defined in the source files.

  - **Top Library**: Specify the top-level module library name.

  - **Loop Count**: Specify the maximum loop iteration value. The default is 1000.

The Loop Count option is used during RTL elaboration, but does not apply to synthesis. For synthesis, you must specify the `-loop_iteration_limit` switch in the **More Options** field of the **Synthesis Settings** dialog box.

Simulation Settings

Figure 3-47 shows the Simulation Project Settings dialog box.
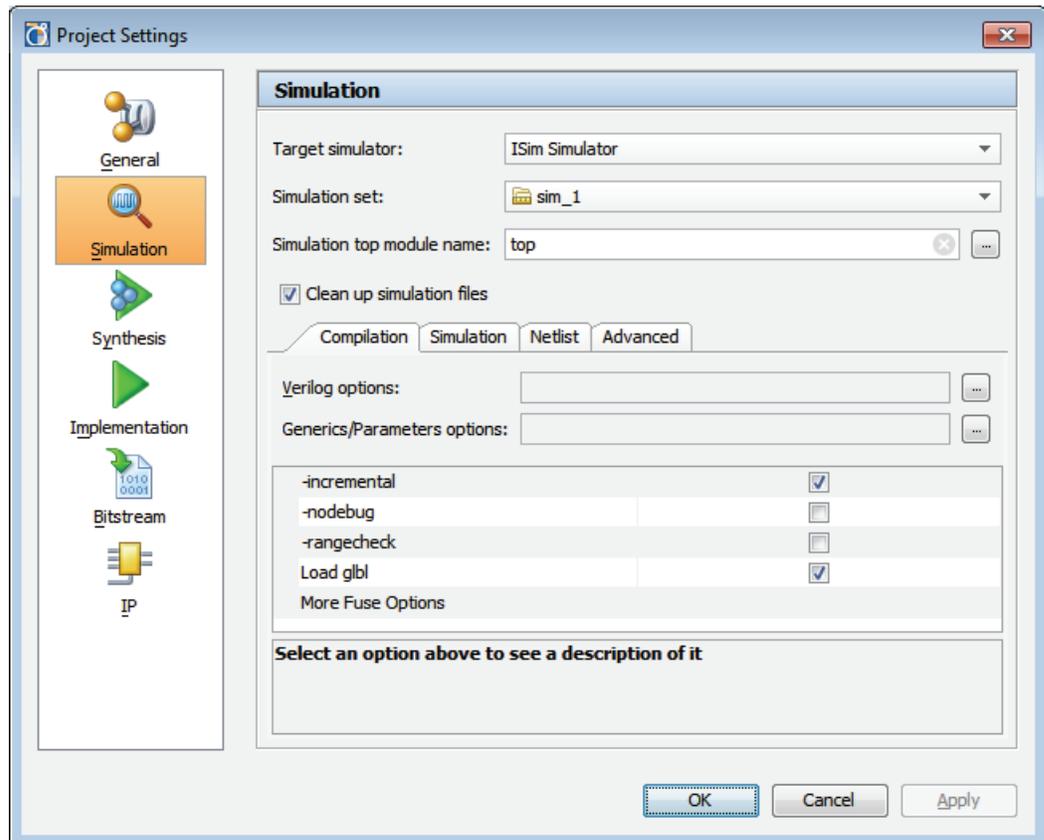
Figure 3-47:   **Simulation Project Settings**

- **Target Simulator**: Specifies the simulator to be launched for behavioral or timing simulation. The available options are:

  - **ISim Simulator**: Specifies the target simulator is Xilinx ISim.

  - **QuestaSim/ModelSim**: Specifies the target simulator is Mentor Graphics® ModelSim or Questa® Advanced Simulator tool. The specified simulator must be installed, and appear in your $PATH in order to be properly invoked when launching simulation.

    In order to support the use of ModelSim/Questa you must compile the Xilinx simulation libraries for use with the target simulator. Type **compxlib** from the Tcl Console inside the PlanAhead tool, or from an ISE Design Suite command prompt to open the *Xilinx Simulation Library Compilation* wizard. Select the **More Info** button on the dialog box to open the ISE Help system for instructions on using compxlib.

    After the libraries are compiled, the simulator will reference these compiled libraries using the `modelsim.ini` file. The modelsim.ini file is the default initialization file and contains control variables that specify reference library paths, optimization, compiler and simulator settings.

The `modelsim.ini` is located as follows:

- The path specified by `compxlib.compiled_library_dir` variable set at the time compxlib is run.

- The path defined by MODELSIM environment variable.

- The path defined by MGC_WD environment variable.

If the `modelsim.ini` file is not found at any of these locations a warning message is returned by the simulator.

Refer to Performing Timing Simulation, page 354 for a description of the simulation project settings.

Synthesis Settings

Figure 3-48 shows the Synthesis Project Settings dialog box.

The Synthesis Settings can be accessed directly by selecting the **Synthesis Settings** command in the Flow Navigator.
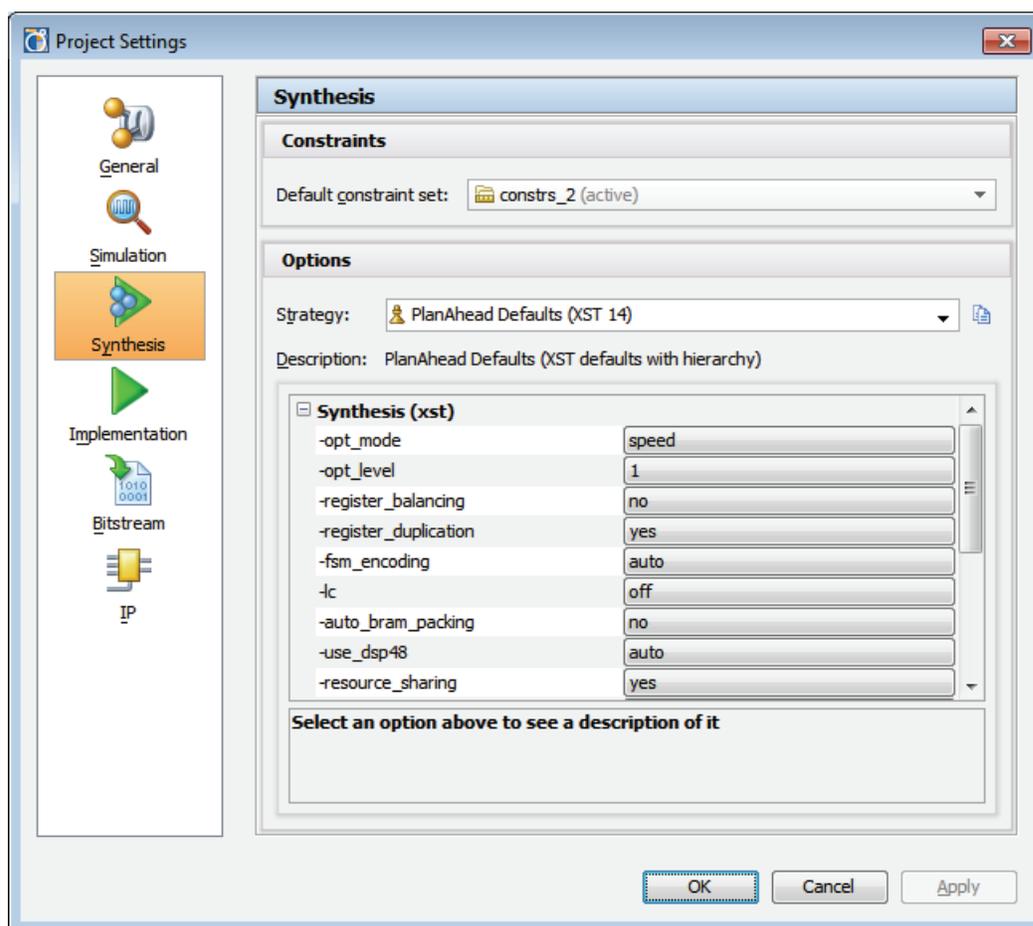
Figure 3-48: **Synthesis Project Settings**

The Synthesis Project Settings dialog box contains the following information:

- **Default Constraint Set**: Defines the active constraint set. This is the constraint set that will be used for new runs, but is also the constraint set that is targeted for any design changes. See

Managing Constraints, page 53 for more information on constraint sets.

- **Strategy**: Select the strategy to use for the synthesis run. The PlanAhead tool includes a set of pre-defined synthesis strategies, or you can define your own. For more information see Defining Strategies for Synthesis and Implementation in Chapter 4.

  When you select a synthesis strategy, the command-line options for XST display in the lower part of the dialog box as shown in Figure 3-48. You can override synthesis strategy settings by changing the values of command-line options.

- **Description**: Displays a text description of the selected strategy.

For more information on setting Synthesis options, see Chapter 6, Synthesizing the Design.

Implementation Settings

Figure 3-49 shows the Implementation Project Settings dialog box.

The Implementation Settings can be accessed directly by selecting the **Implementation Settings** command in the Flow Navigator.
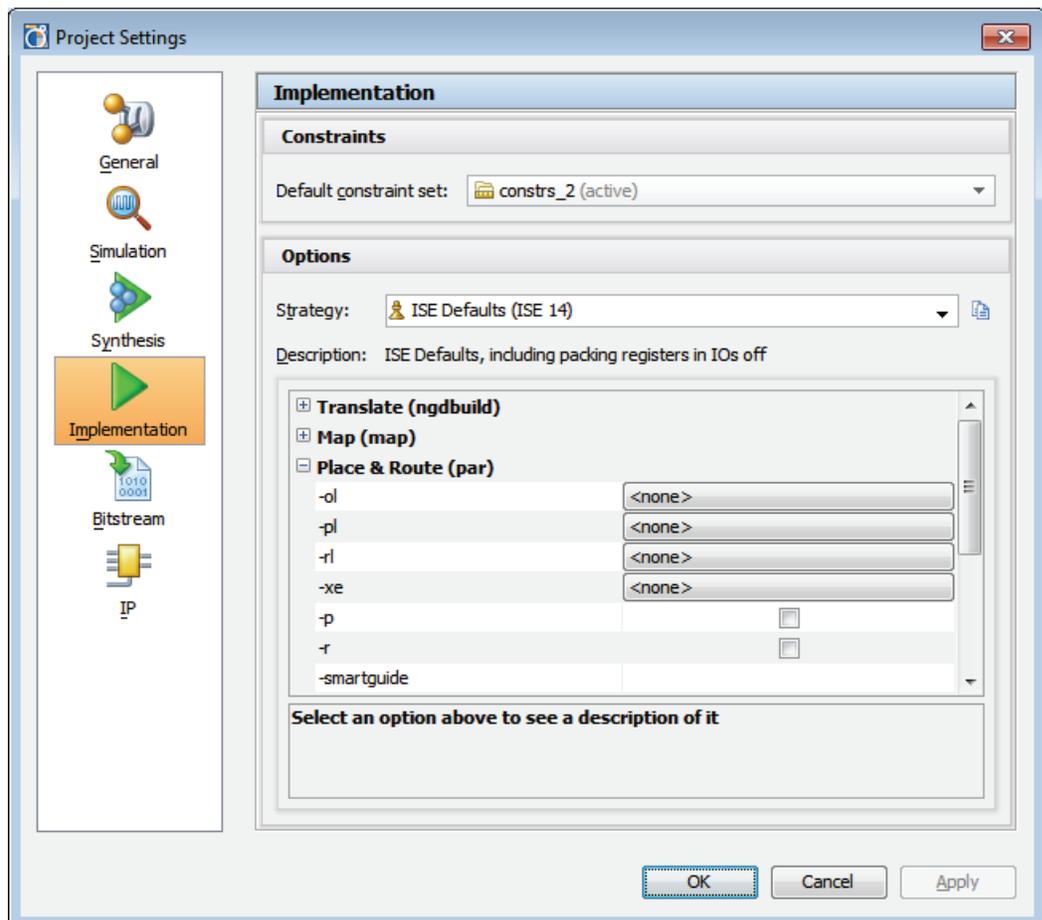
Figure 3-49:   **Implementation Project Settings**

The Implementation Project Settings dialog box contains the following information:

- **Default Constraint Set**: Select the constraint set to be used for the implementation run.

- **Strategy**: Select the strategy to use for the implementation run. There is a set of pre-defined strategies to select from, or you can create your own. For more information see Defining

Strategies for Synthesis and Implementation in Chapter 4.

When you select a strategy, the command-line options for the ISE implementation tools (NGDBuild, MAP, PAR, and TRCE) display in the lower part of the dialog box as shown in Figure 3-49. You can override implementation strategy options by changing the command-line options.

- **Description**: Displays a textual description of the selected implementation strategy.

For more information on setting Implementation options, see Chapter 9, Implementing the Design.

Bitstream Settings

After Implementation is successful, you can run the ISE® **BitGen** command on the Implemented Design to create the bitstream data to program the Xilinx device.

The Bitstream Project Settings dialog box lets you define BitGen options prior to generating the bitstream. When you select an option, a description of the option displays in the dialog box. For more information about BitGen options, see the *Command Line Tools User Guide (UG628)*, cited in Appendix E, Additional Resources.
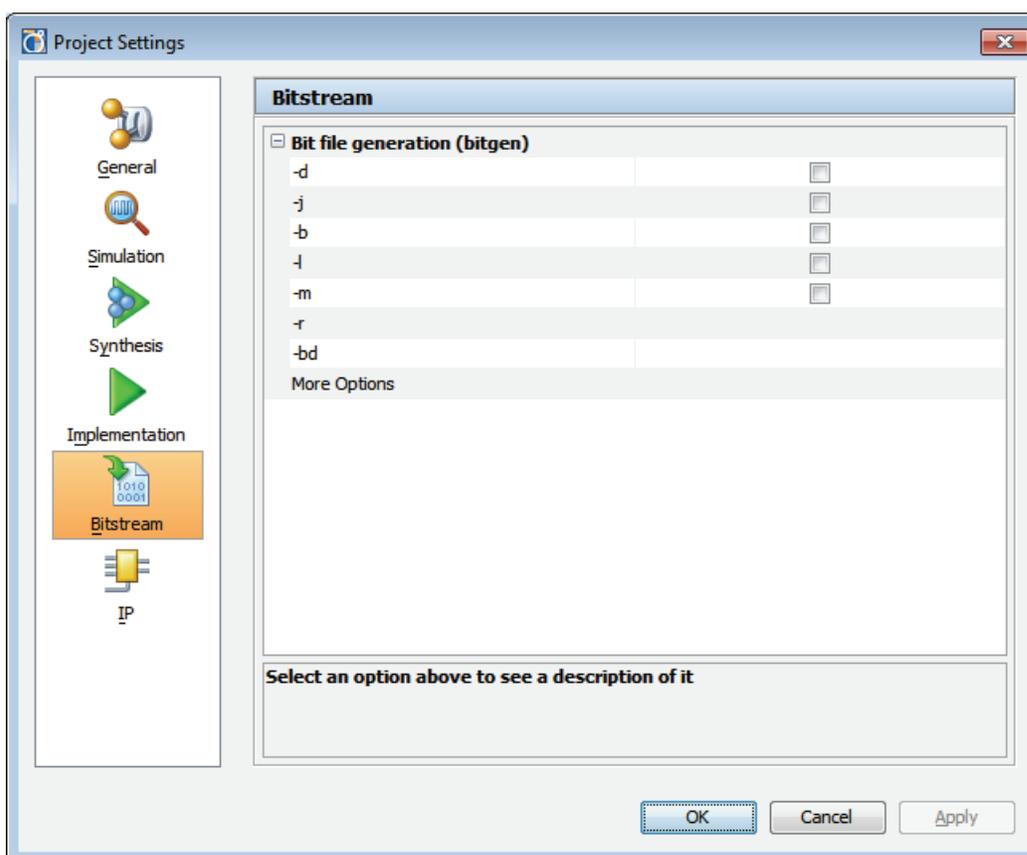
Figure 3-50: **Bitstream Project Settings**

IP Settings

shows the IP Catalog Settings dialog box.
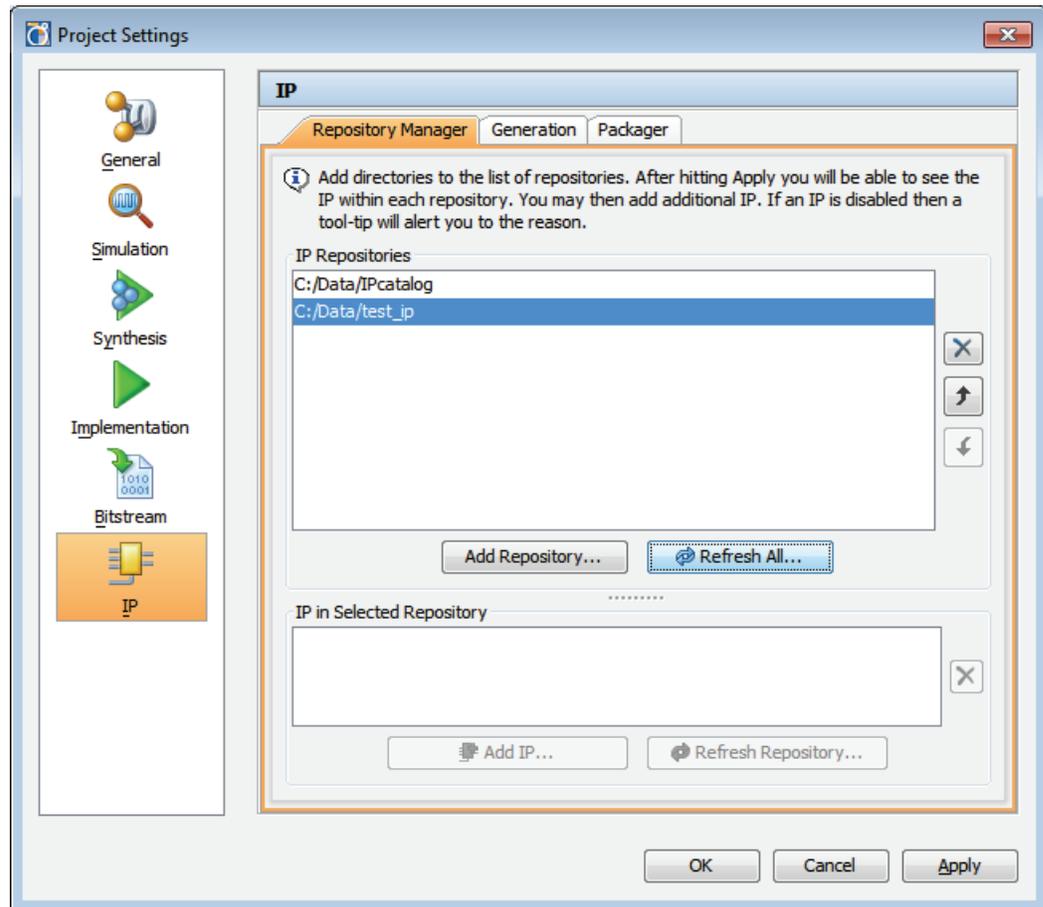
Figure 3-51: **IP Catalog Project Settings**

The IP Catalog settings are:

- **Repository Manager**: Manage the IP repositories used by the PlanAHead tool to locate IP cores to add to a design.

- **Generation:** Specify the default output products to generate when IP is added to the design.

- **Packager:** Specify Vendor, Library, Taxonomy settings to be used by the IP Packager when creating new IP cores.

Repository Manager

With the Repository Manager, the PlanAhead tool allows you to add local directories to the IP Catalog, to add new IP to the repositories, and to manage the local IP repository as part of the catalog.

IP Repositories: Lists the optional IP repositories currently added to the IP Catalog.

- **Add Repository:** Specify directories to add to the IP repository search path. IP files in the form of a `component.xml` or a ZIP file will be added to the IP Catalog.

- **Refresh All**: This option forces the index files in each of the IP repositories to be rebuilt, and then it updates the in-memory IP catalog from the rebuilt index files. If you have added new IP

to a repository then you need to use this option to force the index file to be rebuilt and the new IP to be picked up.

By default, Update IP Catalog will update the in-memory IP catalog from the existing index files. It will not regenerate the index files that already exist for the repositories, and will not pick up any newly added IP.

- **Remove:** The 'X' icon removes the selected IP repository path from the IP catalog.

- **Move Up**: The upward pointing arrow icon moves the selected IP repository up in the list order.

  The order of the IP repositories is important because it affects the order in which the IP cores are read from the IP catalog. The first IP repository is always the Xilinx IP Catalog, then each user-defined IP repository as it is listed in the search path. In the case of duplicate IP cores, the later definition of an IP core overwrites the prior definition. In this case, the IP core found in the last IP repository would be the one imported into the design.

- **Move Down**: The downward pointing arrow icon moves the file or directory down in the list order.

IP in Selected Repository: Lists the specific IP cores found in a selected IP repository.

- **Add IP:** Select IP files in the form of a `component.xml` or a ZIP file to add to the selected IP repository.

- **Refresh Repository:** Rebuild the index files of the selected IP repositories, and update the in-memory IP catalog from the rebuilt index files. Use this after adding IP to the repository to update the in-memory catalog.

For more information on managing IP catalogs, see Chapter 3, Managing IP Cores.

Chapter 4

Using the Viewing Environment

The PlanAhead™ tool has a dynamic viewing environment comprised of view layouts that display the design and device information for the design current task. It provides unique capabilities and viewing perspectives for design information.

Most views cross-select. Information selected in one view is also selected in other open views, enabling efficient methods to examine the design and device information. Select a port in the I/O Ports View, then see the highlighted port in the Device View or Package View for example. Cross-selection also applies to other open designs in a single project. Selecting an object in an implemented design of a project also cross-selects the object in an open Register Transfer Level (RTL) design.

The type of project determines the initial appearance of the viewing environment. See Chapter 3, Working with Projects.

The PlanAhead tool lets you control each major step of the FPGA design process, including RTL development and analysis, logic synthesis, constraints definition, physical design analysis, floorplanning, and implementation control with the Xilinx® ISE® Design Suite software.

The Viewing Environment

The main elements of the viewing environment are shown by number in Figure 4-1, page 98:

1. Main Menu

2. Main Toolbar

3. Flow Navigator

4. View Layout Selector

5. Main Viewing Area

6. Project Status Bar

7. Tcl Console and Messages Area

8. Status Bar
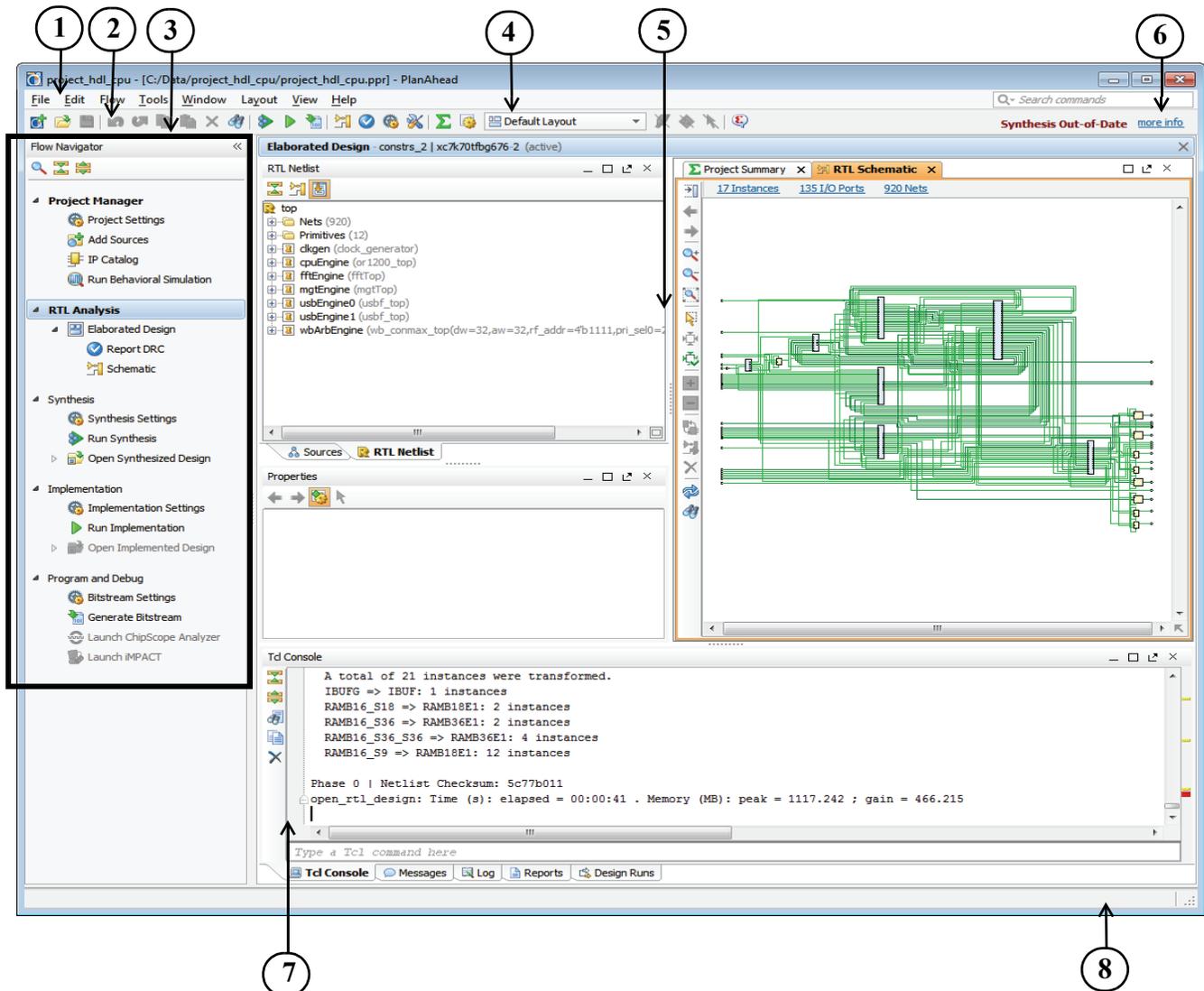
Figure 4-1: **PlanAhead Tool Viewing Environment**

1. Main Menu

Contains the available commands. Certain commands in the menu might not be available, or might be greyed out, if the command is not effective for the current state of the design or for the specific selected object.

2. Main Toolbar

Contains commonly-used commands, and extends to include Design specific commands.

3. Flow Navigator

Enables workflow-like control over the design process. The defined flow lets you manage project data, view and edit RTL source and constraint files, launch synthesis and implementation, and generate bitstream files. These commands are also located under the **Flow** menu of the main menu.

4. View Layout Selector

Provides access to predefined and user-defined view layout configurations. See Using View Layouts, page 102 for more information.

5. Main Viewing Area

Displays the Project Summary and graphical views of the design. Figure 4-2 shows that the main viewing area is split between data views and graphical views.

- Data views present tree and text views of design data such as the Sources, Netlist, and Property views.

- Graphical views, such as the Schematic, Device, and Package views, are displayed in the workspace area and are for interacting with the design and device resources.

See Working with Views, page 108 for information on using these different views.
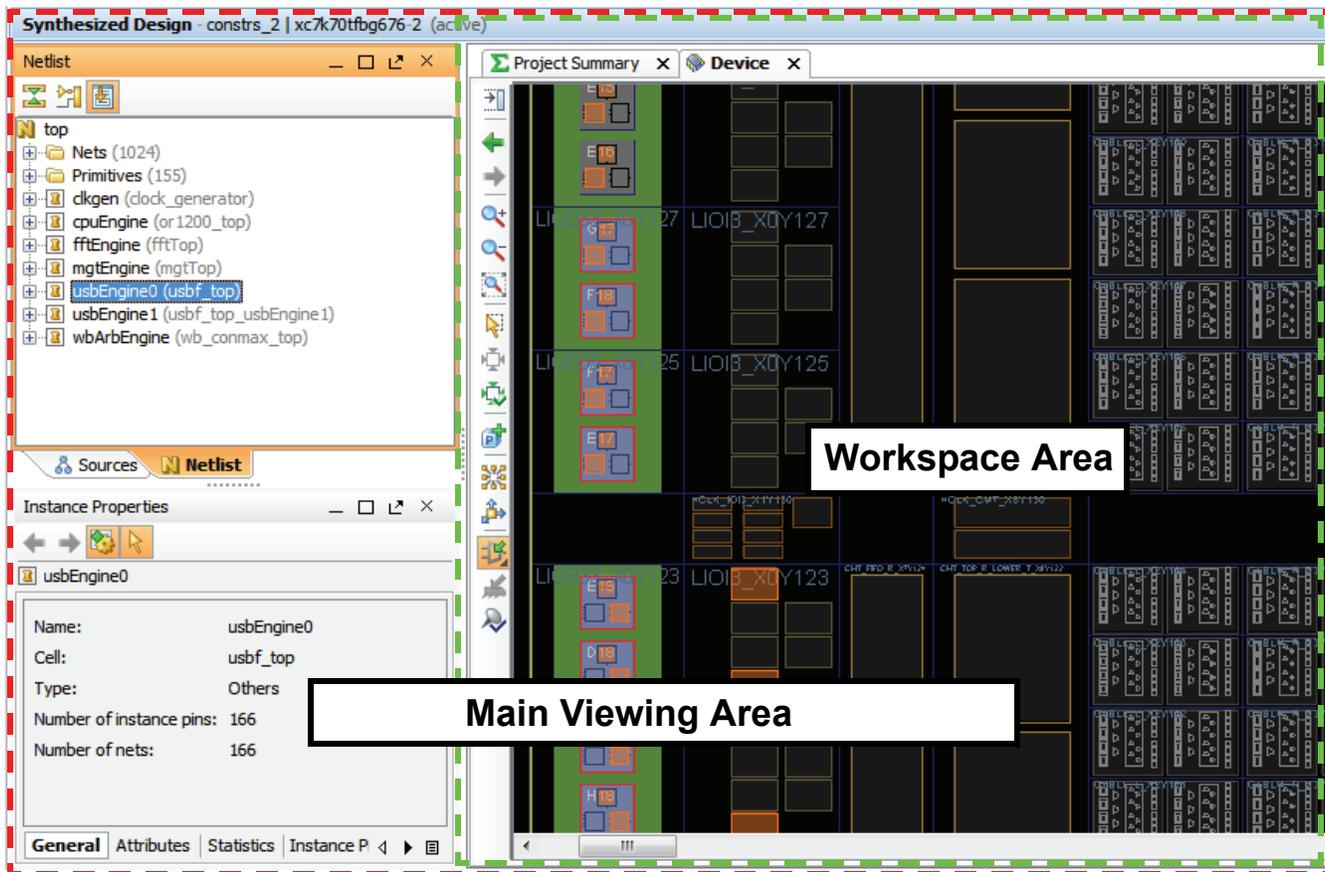
Figure 4-2: **The Main Viewing Area**

6. Project Status Bar

Shows project status and the actively-running commands. You can cancel Synthesis, Implementation, and Generate Bitstream commands from this area.

7. Tcl Console and Messages Area

Shows Tcl command status, application messages, compilation results, reports, and access to design runs. See Using the Tcl Console and Messages Area, page 103 for more information

8. Status Bar

Displays information about the open project and any objects currently beneath the cursor.

Using the Main Viewing Area

Each window or view has a set of banner controls to maximize or restore the view, float the view as separate from other views, or close the view altogether. These controls are located in the upper right corner of the view, as shown in Figure 4-3.
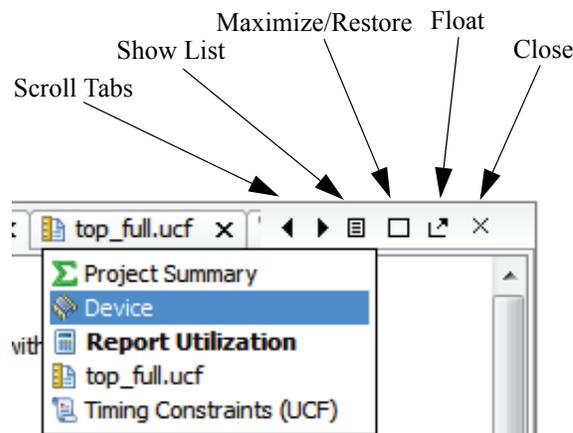
Figure 4-3: **Window Controls**

The following subsections describe these commands in greater detail.

Scroll Tabs

In the Workspace Area, when you have multiple views open, such as the Project Summary, Device view, Package view, Timing Constraints view, and possibly more, the **Scroll Backwards** and **Scroll Forwards** icons appear as arrowheads in the window banner. Use these icons to scroll the displayed tabs either forward or backwards to make any hidden tabs visible.

Show List

As an alternative to scrolling the tabs forward or backward, the **Show List** command displays a list of all open view, and lets you choose which one to make the active view.

Maximize/Restore

You can expand a view to use the all of the available viewing environment by clicking **Maximize** in the upper-right of the view. The PlanAhead tool minimizes all other open views, except the Flow Navigator, and expands the selected view to fill all available screen area. After maximizing the view, the icon changes to the **Restore** command, which can be used to restore the view to its original size and position. Double clicking in the title bar of any window will also maximize the window, or restore it to its original size.

When one view is maximized, all other open views are minimized and placed temporarily at the side or at the bottom of the maximized view. You can also minimize a specific view at any time by selecting **Restore** in the upper-right corner of the view.

In Figure 4-4 the Properties, Netlist/Sources views are minimized at the side of the maximized Device view. Clicking on any of these minimized views restores it to its original location.
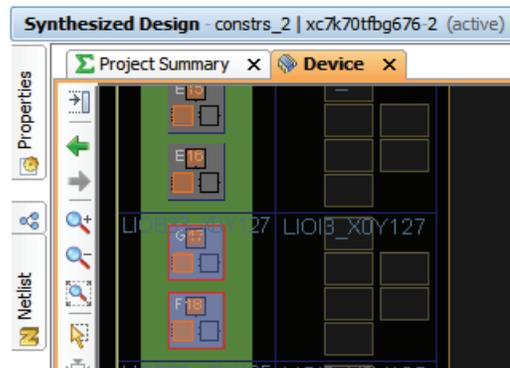
Figure 4-4:   **Minimized views make room for Maximized views**

Float

You can un-dock views, including the workspace, from the display docking area so that it "floats" and can be moved and sized independently. To float a window:

- Click **Float Frame**, or
- Select **Float** from the popup menu

When you activate the floating option, the view appears in a separate floating window.

In the case where windows overlap, you can move it by dragging the view banner after you float the window.

You can move a floating window outside of the main window. The PlanAhead tool stores the default locations and sizes in which to display floating views.

Close

You close the views by clicking on the **X** icon in a view tab.

Hiding and Showing the Flow Navigator

To make more screen space available for other views you can hide the Flow Navigator by:

- Selecting **View > Hide Flow Navigator** command in the main menu, or
- Clicking the **<<** button on the right side of the Flow Navigator.

To re-display the Flow Navigator:

- Select **View > Show Flow Navigator**, or
- Use the window slider to re-size the hidden Flow Navigator as show in Figure 4-5.

Figure 4-5: **Hiding and Showing Flow Navigator**

Using View Layouts

The PlanAhead tool provides predefined view layouts as shown in Figure 4-6, page 102 to facilitate various tasks in the design process. The layout of the views generally supports the required tasks for the selected design process.

After you open a design, you can switch the view layout between available preconfigured layouts using:

- **Layout** menu on the main menu, or
- Layout Selector on the toolbar menu
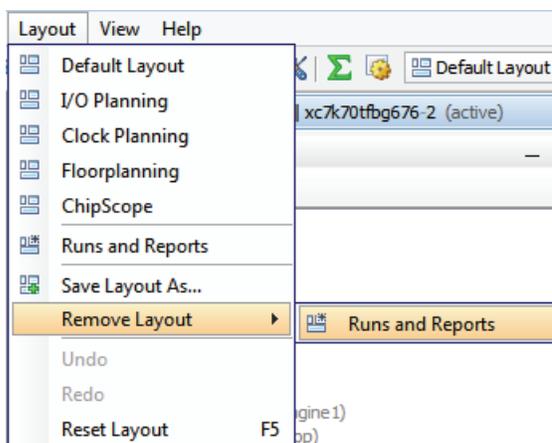
Figure 4-6: **Layout Pulldown Menu**

The predefined view layouts are:

- **Default Layout:** After a design is opened and elaborated into memory, the design data displays in the Design Analysis view layout. You can analyze elaborated and synthesized designs, apply timing and physical constraints.
- **I/O Planning**: Define I/O placement constraints, place ports.
- **Clock Planning**: Cross-select between the Clock Resources view and the Device view and I/O Port view to plan and place clock resources in the design.
- **Floorplanning**: Define Pblocks, manage partitions, and perform hierarchical floorplanning.
- **ChipScope**: Insert ChipScope™ analyzer tool debug cores.

You can also create custom view layouts that meet your specific requirements.

Creating and Removing View Layouts

The PlanAhead tool provides predefined view layout configurations to complete specific design tasks, such as the I/O Planning view layout or the Design Analysis view layout. These view layouts predefine the location and size of views commonly used for a specific design task. The PlanAhead tool also lets you save user-defined view layouts after moving and resizing views to suit your own needs.

- Save a custom view layout using the **Layout > Save Layout As** command. The user-defined view is available in your installation for use in all your design projects. User-defined layouts are listed in the **Layout** pulldown menu from the main menu, and are saved to a layout file by the PlanAhead tool (see Outputs for Environment Defaults in Appendix A).

- Remove user-defined view layouts using the **Layout > Remove Layout** command. This command has a submenu of any user-defined view layouts that can be removed, as seen in Figure 4-6.

- Reset to the original layout view if you have resized or moved views, using the **Layout > Reset Layout** command.

- Undo any view manipulation commands by selecting **Layout > Undo**. To repeat a command, use the **Layout > Redo**.

Using the Tcl Console and Messages Area

The status and results of the commands that run from within the PlanAhead tool display in a set of views grouped at the bottom of the viewing environment. As messages are generated, they appear in the appropriate view within this area.

The views in this area include the Tcl Console, the Messages view, the Log view, Reports view, and Design runs. In addition, some of the other views such as the Find Results view, Package Pins view, and Timing Results display in this area when opened.

Using the Messages view

Messages are grouped under specific headings to enable quick location of messages from different tools or processes. For instance, when you open the Synthesized Design, the PlanAhead tool displays the design and reports messages in the Message view, as shown in Figure 4-7.
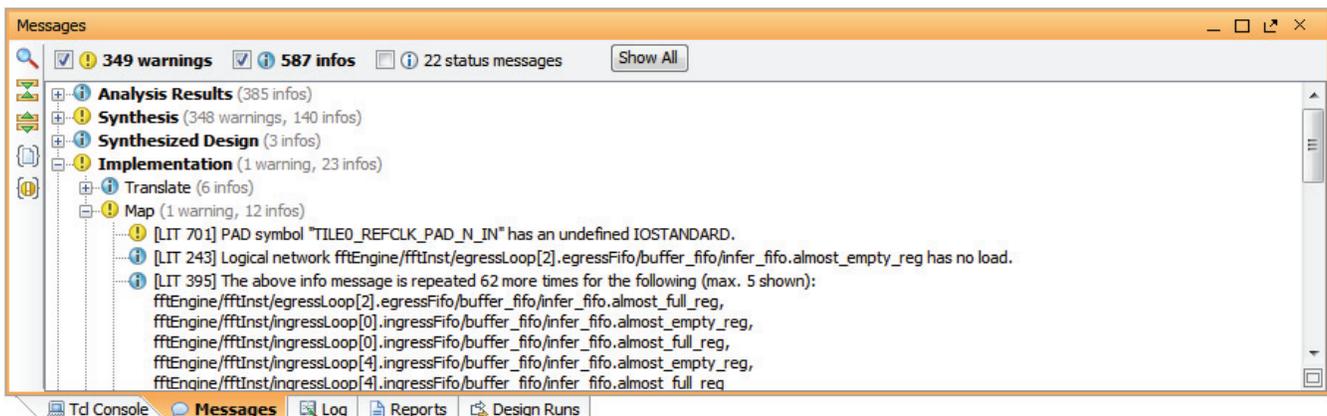
Figure 4-7: **Messages View**

To open the Messages view, select **Windows > Messages**.

- Messages displays with a link to the relevant source file. Click on the link to open the RTL source file in the Text Editor with the appropriate line of code highlighted.

- Expand or collapse the messages by toggling the tree widgets in the view or by clicking the **Expand all** or **Collapse all** icons on the sidebar menu.

- Use the checkboxes in the banner of the Messages view to hide or display the **Errors, Critical Warnings, Warnings, Info** and **Status** messages.

- Use the **Show Search** icon in the sidebar menu to search for and display specific messages. This command is also available through the `Alt+/` keyboard shortcut.

- Use **Group messages by file** to sort the messages for each source file, as shown in Figure 4-7, page 103.

- Use **Group messages by ID** to sort the messages by message ID.

By default, messages written to the Messages view are line-wrapped to allow the display of the whole message within the message window. When line wrap is enabled, the lines adjust to fit the width of the Messages view. If the Messages view is resized, line wrapping is adjusted accordingly. If line wrap is disabled, the messages are written to a single line, regardless of the width of the Messages view.

You can enable or disable the line wrap function using the right mouse popup menu **Wrap Messages** command as shown in Figure 4-8, page 104.
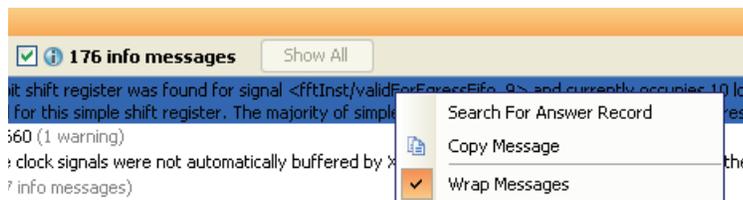
Figure 4-8: **Wrap Messages**

Critical Warnings and Errors are also displayed in a popup dialog when loading a project, opening a design, or creating or launching runs, as shown in Figure 4-9. This is to ensure that you are aware of any issues that might require your attention. These messages can also be found in the main Messages view.
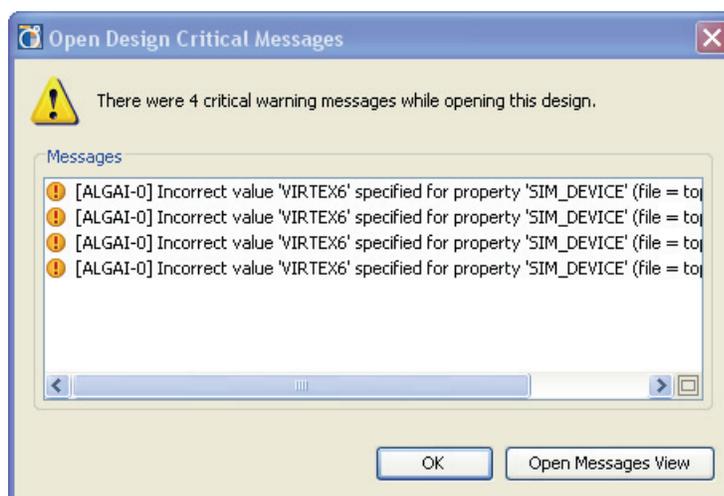
Figure 4-9: **Critical Messages Window**

Using the Log view

The Log view displays the active output status of commands that compile the design such as `ngc2edif`, Xilinx Synthesis Technology (XST), MAP, and Place and Route (PAR), under three separate tabs for Synthesis, Implementation, and Simulation, as seen in Figure 4-10, page 105.

The Log view opens automatically when you launch a command on an active run. The output displays in a continuous scrolling format and is overwritten when new commands are run.

To open the Log view, select the:

- **Log** tab, or

- **Window > Log**
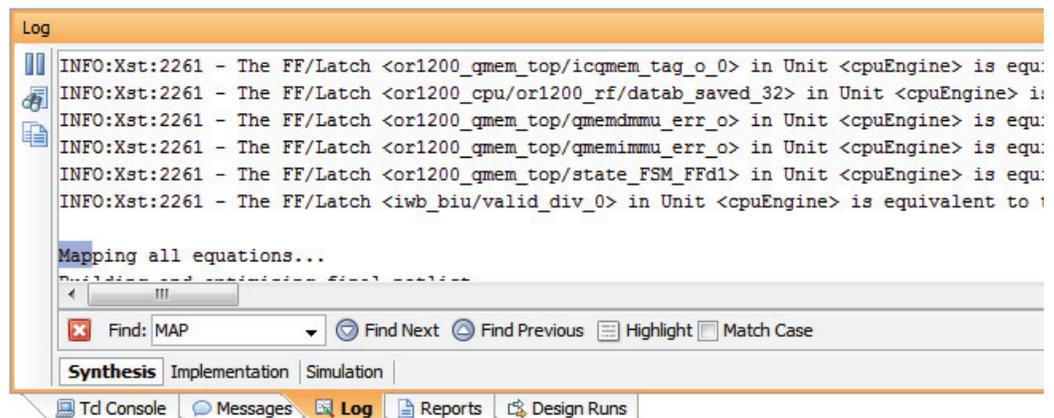
Figure 4-10: **Log view**

- Use the **Pause output** button to scroll back or read the Log view while commands are still running, and information is being added to the view.

- Use **Show Find** to open a search field that lets you locate, and highlight, specific words or phrases in the Log view text.

- Use **Copy** to copy selected text from the Log view into a buffer for pasting into the Text Editor, or other locations or applications.

Using the Tcl Console

The Tcl Console view displays messages from previously executed Tcl commands. The PlanAhead tool writes messages to the `planAhead.log` file. Command errors, warnings, and successful completion echo to this window also.

The status of design netlists and constraints that open in the Netlist Planner and Results Viewer display also.

To invoke the Tcl Console view, select **Window > Tcl Console**. Figure 4-11, page 106 shows the Tcl Console.
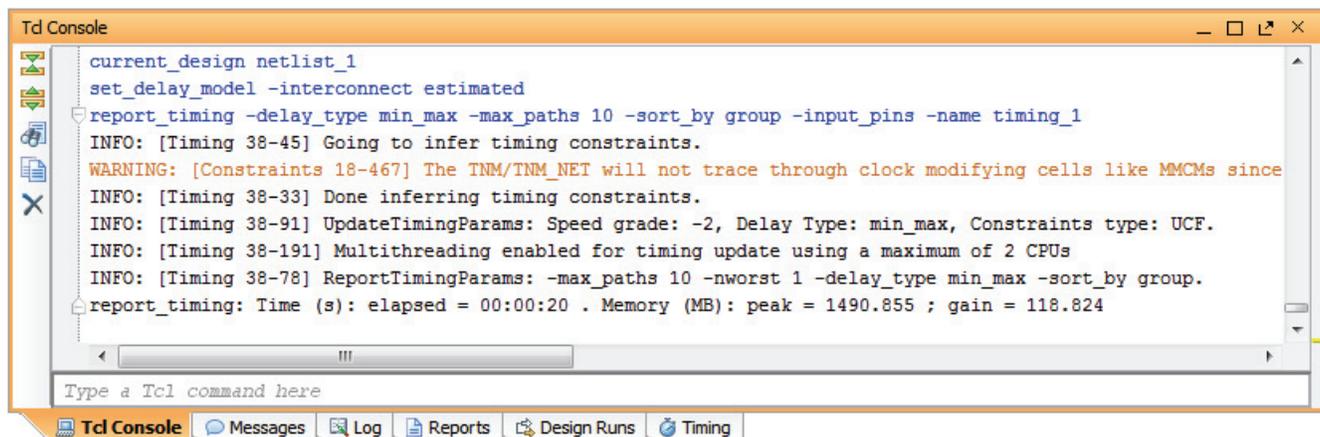
Figure 4-11: **Tcl Console**

In the Tcl Console view, you can:

- Expand or collapse the messages reported by each Tcl command in the Tcl Console view by toggling the tree widgets or by clicking the **Expand all** or **Collapse all** icons on the sidebar menu.

- Find text strings in the displayed messages using the **Show Find** icon.

- Use **Copy** to cut and paste commands within the Tcl Console.

- Use **Clear all output** or **Clear All Output** popup menu command to clear the Tcl console report.

Locating Warnings and Errors in the Tcl Console

Warnings and Errors in the Tcl Console are marked with a yellow or red indicator to the right of the view.

- Hold the mouse over one of the yellow or red indicators to display the warning or error in a tool tip.

- Click on the color indicator to jump to the associated message in the Tcl Console.

Entering Tcl Commands[1]

You can use Tcl format commands on the command line entry box at the bottom of the Tcl Console view (shown in Figure 4-11, page 106). To enter commands, click on the command line and type Tcl commands.

The Tcl Console auto-complete feature attempts to complete the name of the command, or command parameters, as you type on the command line. In Figure 4-12, page 107 the Tcl Console provides a list of commands that match "**creat**". Click on a command from the list to select it, or use the up or down arrow key or scroll bar, to scroll to the command and hit Enter to select it.

You can also continue to type until the auto-complete feature has narrowed the matching commands to one choice, and press the **Tab** key to select that command.

After a command is selected, the Tcl Console also attempts to auto-complete any arguments of the command based on your typing. You can also select from a list of parameters, or press the **Tab** key when the list is narrowed to one choice.
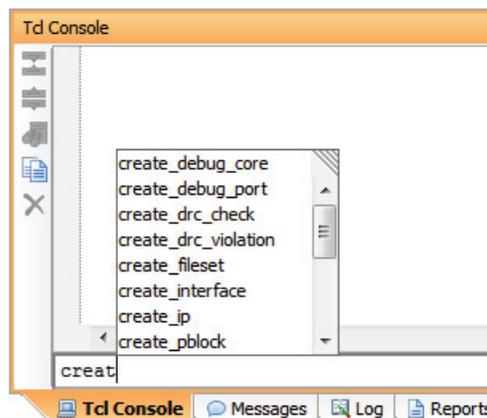
Figure 4-12: **Auto Complete Tcl Commands**

The PlanAhead tool writes a Tcl command equivalent to the Tcl Console and to the journal file, `planAhead.jou`, for all actions performed using a menu command or through direct manipulation, such as drag and drop. The `planahead.jou` file is written each time the PlanAhead tool is launched.

Note: Backup versions of the journal file, named `planahead_<id>.backup.jou`, are written to save the details of prior runs whenever the PlanAhead tool is launched. The *<id>* is a unique identifier that allow the tool to create and store multiple backup versions of the log and journal files.

Use the **history** command to access the command history in the Tcl Console. Press the **Up** or **Down** arrow keys to scroll through the command history one command at a time.

---

1. On Linux, the function of specific keystrokes or key combinations can be configured with an `.editrc` file in your home directory. See Chapter 14, Mapping Keystrokes in the Tcl Console for more information.

Using Tcl Help

Command line help is available for commands by using the following syntax at the command line:

```
Command> help
```

You can retrieve more detailed information about the commands by extending the help query with the following command:

```
Command> help get_cells
```

The Tcl Console displays the list of available commands or command options based on the help topic that you enter.

For explicit command syntax, perform the command once, then view the `planAhead.jou` file in the invocation directory. See Chapter 14, Tcl and Batch Scripting, or refer to the *PlanAhead Tcl Command Reference Guide (UG789)* as referenced in Appendix E, Additional Resources, for more information about the Tcl command formats and the available help.

Hiding the Tcl Console and Messages view

To hide the entire set of Tcl Console and Messages views click **Minimize** in the banner of any of the Messages views.

When the Tcl Console and Messages views are minimized, they temporarily display as tabs at the bottom of the viewing environment. Click any of the minimized tabs to restore the Messages view to its original size and location.

Figure 4-13, page 108 shows the **Minimize** button in the upper right of the window, as well as the minimized Messages tabs displayed in the lower left of the viewing environment.

Figure 4-13: **Minimize View command with Minimized Message View tabs**

Working with Views

The PlanAhead tool has different types of views for displaying different types of information. You can independently control views with respect to size, visibility, and location.

Opening Views

To open most view types use the **Window** menu from the main menu. Select a window that is already open to make it the active window.

As commands run, new views open in the interface with which to interact, or to display results.

- The Schematic view requires at least one design object to be selected in another view, such as in the Sources view. Use the popup menu **Schematic** command or the **Schematic** toolbar button to open the Schematic view. See Using the Sources View, page 128.

  Note: An exception to this general rule is when opening the Elaborated Design a Schematic view of the top-level of the hierarchy is automatically opened without selecting anything.

- The Properties view requires one object to be selected in another view, such as in the Netlist view. If multiple items are elected, the properties for the last selected item are displayed. Use

the popup menu `Object_type` **Properties** command to open the Properties view. See Using the Properties View, page 152.

Navigating Views

Each opened view in the viewing environment has a tab to select and make that view active. Some views have a tab at the bottom of the view window, such as the Tcl Console and Messages view. Some views have a tab at the top of the view window, such as the Project Management and Device views. You can activate a view by selecting the appropriate tab.

In views, the following actions are available:

- Double-click the view tab to display the view on the full screen.

- Restore workspace views by double-clicking the view tab again.

- Stretch the overall size of these viewing areas by sliding the view borders. The cursor changes to a slider symbol you can use to stretch the view borders.

- Move the views by selecting a view tab and dragging it to a new location. See Configuring the Viewing Environment, page 172 for more information.

Some view types support multiple tabs. For instance, Figure 4-14 shows the Noise Analysis Results view with two result sets: `results_1` and `results_2`, displayed in tabs within the Results view.
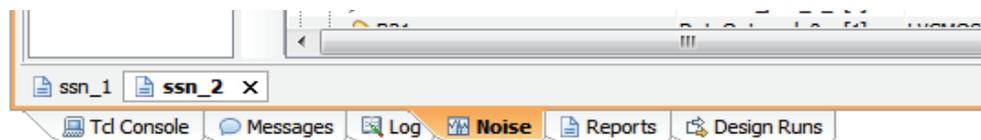
Figure 4-14: **SSN Results Tab**

Moving Views

To move a view from its current location to another, select the tab and drag the view. This produces a rectangular box which indicates where the view is located after the move. Dropping one view onto an existing view places the two tabs in the same region.

To move a view to share the space in a viewing area:

1. Click a tab; for example, the Constraints tab.

2. Drag the tab to a location. The gray outline serves as a guide.

3. Release the tab at the location you want.

Note: You can not move views into, or out of, the graphical workspace.

Using Graphical Workspace Views

The views with a graphical interface and those that require more screen space like the Text Editor or the Report Viewer display in an area called the *workspace*. These views are different than other views because more than one can be opened simultaneously to view or compare different information. These workspace views can be made full size, floated or split by using the popup menu commands on the view tab.

The PlanAhead tool workspace is the graphic viewing area that displays the graphical views and some report and log information. The views that display in the workspace are:

- Project Summary
- Text Editor
- Device view
- Package view
- Clock Resource Planner
- Schematic view
- Hierarchy view

To open a new Device or Package view, select **Window > Device** or **Window > Package**. You can open multiple views of the same view type within the workspace area. For example, two Device views can display different areas of the device.

While most graphical views can be opened from the Window menu, the Schematic and Hierarchy views must be opened after selecting a logic element from another view. To open a Schematic view:

1. Select at least one object to display in schematic format in the Netlist view for instance.
2. Select the **Schematic** command from the popup menu,
   - Press **F4**, or
   - Click the **Schematic** toolbar button

Understanding the Context Sensitive Cursor

The cursor changes based on the available command mode. When the cursor changes to a:

- Horizontal, vertical, or diagonal stretch bar symbol, you can stretch Pblock edges and Windows view borders.
- Hand symbol, you can move Pblocks or instances.
- Cross symbol, you can draw rectangles for zooming in, defining pin assignment areas, or drawing Pblock rectangles.
- Slashed circle symbol, you are dragging objects are over illegal placement sites.
- Move point-to-point symbol, you are dragging objects over legal placement sites.

Using Mouse Strokes to Zoom

You can hold down the left mouse button, and drag the cursor in the Device view to zoom into an area, or to zoom out. The mouse stroke commands are:

- **Zoom Area**: Press and hold the left mouse button, while drawing a rectangle from top-left to bottom right to define the area to zoom into.
- **Zoom In:** Press and hold the left mouse button while drawing a diagonal line from upper-right to lower-left. This zooms in the Device view by a variable amount. The length of the line drawn determines the zoom factor applied.
- **Zoom Out**: Press and hold the left mouse button while drawing a diagonal line from lower-left to upper-right. This zooms out the Device view by a variable amount. The length of the line drawn determines the zoom factor applied.
- **Zoom Fit**: Press and hold the left mouse button while drawing a diagonal line from lower-right to upper-left. The Device view is zoomed out to display the entire device.

Using the World View

When zoomed-in on a graphical workspace view, such as the Device view, you can open the World view to navigate around the overall design area. The World view displays a less detailed overview of the active graphical workspace to enable a quick pan of the viewed area. This capability is available in the Device view, Schematic view, Package view, and Hierarchy view when you are zoomed into a small region of the device or design.

To invoke the World view, click the World view icon in the lower right-hand corner of a graphical workspace view, such as the Device view, as shown in Figure 4-15.
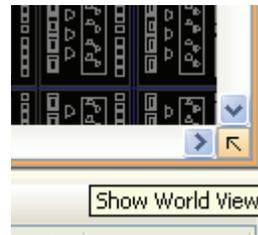
Figure 4-15: **World View**

The World view reflects the zoom area and the selected objects for the active view. In Figure 4-16, page 111, the world view shows the overall Device view which is currently zoomed into the area identified by the navigation rectangle in the World view. You can select and drag the navigation rectangle to reposition the displayed area in the graphical workspace view.
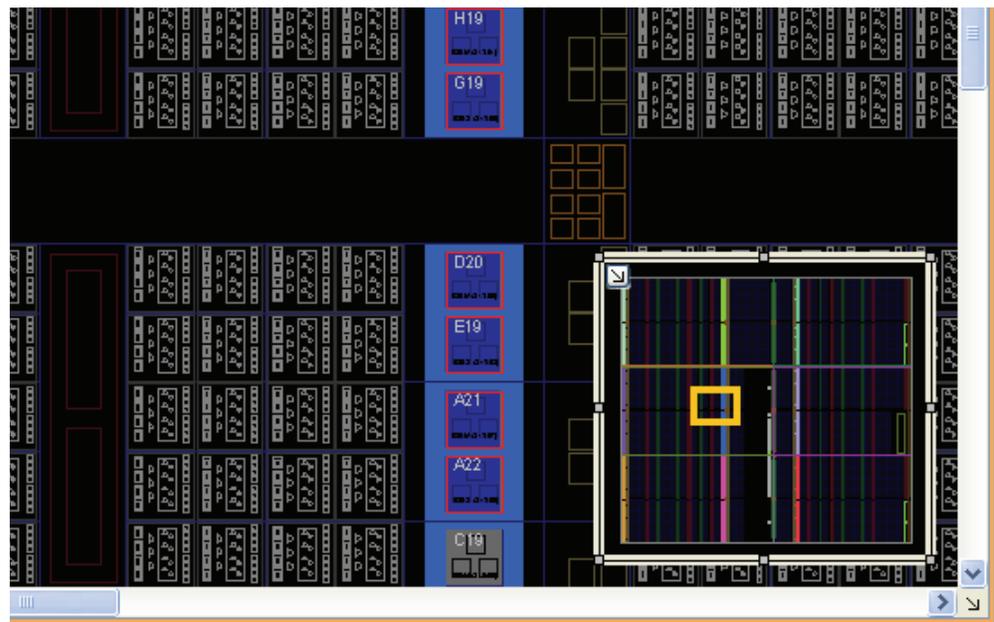
Figure 4-16: **World View**

The World view opens to a default size. To resize the World view to make it larger or smaller, left-click and hold and drag any of the drag handles on the edge of the World view as shown in the figure above.

To reposition the World view, left-click and hold and drag anywhere on the perimeter of the view, except on the drag handles. Use this feature to reposition the World view anywhere within the limits of the graphical workspace view.
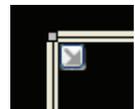
To close the World view, click on the downward pointing arrow icon in the view.

When you resize or reposition the World view, it remains at the size and position you have established whenever you close and reopen the view.

Printing the Workspace View

You can print the view that is active in the workspace: Device view, Package view, Schematic view, and Instance Hierarchy view. Select **File > Print** to print the current viewable area.

Splitting the Workspace

You can split the workspace viewing area horizontally or vertically to enable multiple simultaneously displayed views. Each panel acts independently, allowing multiple views to be docked for viewing.

You can open two views of the same type, such as two Device views for viewing different areas of the device, or at different zoom levels. You can also open two different views, such as the Device and Package view, as shown in Figure 4-17, page 112, or the Device and Clock Resources view to permit better interaction between the two views.
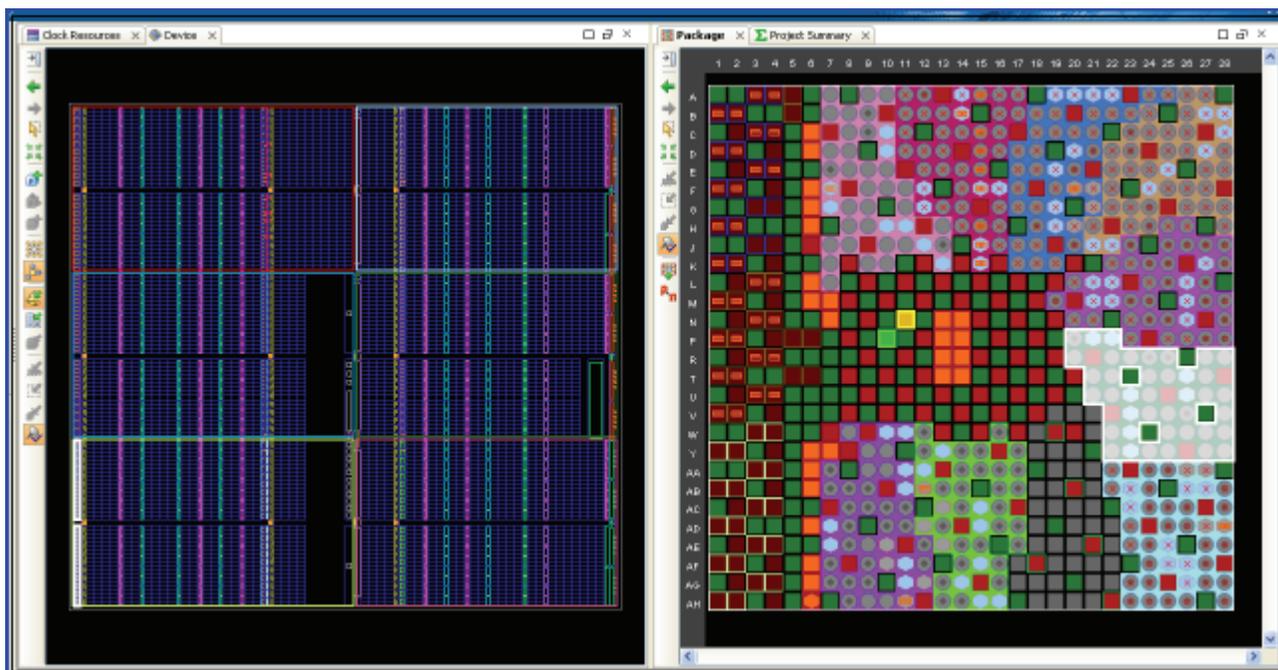
Figure 4-17: **Splitting the Workspace to Display Device and Package Views**

You can split the workspace by either of the two following methods:

* Use the **New Horizontal Group** or **New Vertical Group** commands from the popup menu. These commands are available in the workspace views only.

* Select the tab of an open view and drag it to the very far right scroll banner of the workspace. A grey rectangle displays where the view can be placed. Position the cursor so that the workspace view arrangement is as desired, and release the mouse to move the view and split the workspace. This same technique works with many of the views.

Merging Split Views

If you split the workspace views, you might need to merge the views to better utilize the available viewing area. To collapse the views back into one tabbed area, do one of the following:

- Select **Move to Previous Tab Group**, or **Move to Next Tab Group** from the popup menu in a workspace view.

- Select any view and drag it onto the view tab of another. The dragged grey rectangle displays around the entire view, showing how the windows will merge.

Using Tree Table Style Views

The PlanAhead tool contains views that display as expandable spreadsheet tables. These views all share some common characteristics and features described in the following subsections, and illustrated in Figure 4-18, page 113.

Figure 4-18:   **Tree Table Style View**

Expanding and Collapsing the Table

Toggle the expand and collapse widgets in the Name column to expand or collapse the tree independently.

The **Expand all** and **Collapse all** buttons expand or collapse the entire tree.

Display Entries in a Flat List or in a Group

Most of these style views have a **Group by Type** button or equivalent which either display the entries grouped by an expandable type or as a single flat list of entries.

Flattening the list is helpful to search and filter the overall list of entries, as shown in Figure 4-19.
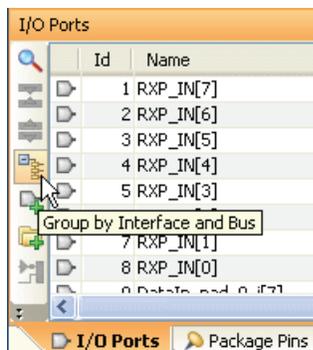
Figure 4-19: **Group by Type or Flat List Toggle Button**

Using the Show Search Capability to Filter the List

Click the **Show Search** button to display a search field in the banner of the view enabling any text string to be entered to filter the list displayed in the table view. Use the keyboard shortcut **Alt**+/ to quickly access this command.

For best results flatten the list first by using the **Group by Type** toggle button described in the previous section.

Any column in the table can be used as search filter criteria. Select the pulldown menu in the Search field to select a column header to search. Figure 4-20, page 114 shows the search pulldown menu.
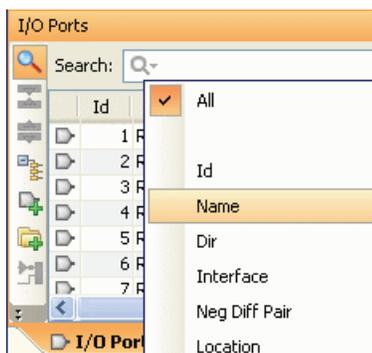
Figure 4-20: **Search Pull down Menu**

Enter any text string and the list adjusts dynamically to list only those entries that contain the string. Select the **Show Search** button again to remove the Search field and sorting.

Sorting Columns

Sort any table by clicking in a column header. This sorts the data in the table in an increasing order according to the sort criteria of the selected column. Click the column header again to sort the data in the table in a decreasing order according to the sort criteria of the selected column. A visual indication of the sort order and direction displays in the column header, as shown in Figure 4-21.
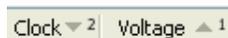
Figure 4-21: **Up/Down Sort Arrows**

To add a second sort criteria from a second column, press **Ctrl** and click the header of a second column. In the example in Figure 4-21, the Voltage column is the primary sort criteria, and the Clock column is the secondary sort criteria.

Add additional columns as needed to refine the sort. Press **Ctrl** and click the column header again to remove a sort from a column from the sort criteria.

Organizing Columns

You can move, hide, and restore columns:

- To move a column, select it and drag it into a new location.
- To hide a column, select it and use the popup menu in the column header to select **Hide This Column**.

The popup menu also enables quick view configuration for each column:

- Adjust the width of the columns per the displayed data using the **Auto Resize Column** command.
- Restore the defaults using the **Reset to Default** command.

Using View-Specific Toolbar Commands

Most views provide toolbar buttons within the view to operate commands specific to that view, as shown in Figure 4-22.
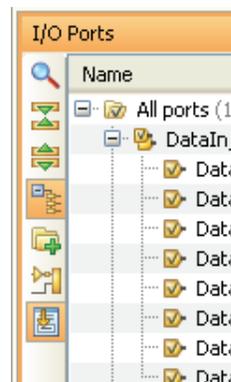
Figure 4-22: **View Toolbar Menus**

These buttons become enabled only when certain data is selected or commands are active. The PlanAhead tool features are made available through these view-specific toolbar buttons so it is beneficial to become familiar with them. These view-specific commands are covered in more detail in the specific view sections of this document.

Using the Main Menu Command Search

The PlanAhead tool provides a **Command Search** field on the main menu bar at the top right corner of the viewing environment to quickly locate and execute a command from the main menu. Enter a few letters of a command name, and a list of commands that match your search criteria displays.

The search mechanism employs a wildcard search of the form *value*, where *value* is the string of characters you typed in the search field. For example, in Figure 4-23, typing the letters `cl` in the Command Search field found the following commands: Clear List, Clock Regions, Clock Resources, Close Project, Run Tcl Script, Tcl Console, Clear Prohibit, and Clear Placement. These last two are inactive due to the current state of the example project.
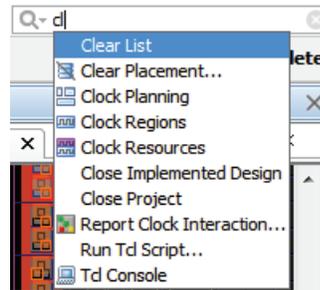
Figure 4-23: **Main Menu Command Search**

The list of commands presented is based on the current design context in the project. An open Elaborated Design offers a different set of commands than an open implemented design.

In addition to commands, the command search field reports project names and files that are listed under the **Open Recent Project** and **Open Example Project** commands on the main menu.

Select a command from the results list of the command search to execute that command.

Using the Status Bar

The application status bar at the bottom of the main window displays useful information such as the item currently under the mouse cursor, the currently selected object, the coordinates of the cursor, and the current design mode. Figure 4-24 shows both the left-hand end and the right-hand end of the status bar, separated to provide a closer look.

Figure 4-24: **Information Banner**

The status bar displays:

- **Status Message**: The first field in the status bar displays context-sensitive information. For example, when the cursor is in the device view or the schematic view, this field contains the name of the instance or site directly under the cursor. The status message field also contains detailed description of commands when the mouse is over its corresponding toolbar button or menu item.

- **Coordinates**: To the right of the status message field is the coordinates field. As the cursor moves over block RAM, DSP48, and other parts in the Device view, this field displays the name and coordinates as shown in Figure 4-24. If the cursor is over a pin in the Package view, this field displays pin information, such as coordinates, type and name.

- **Mode**: Indicates the type of project, such as RTL Flow or Post-Synthesis Flow. When you invoke the PlanAhead tool from ISE Project Navigator, the ISE Integration mode displays.

Selecting, Marking, and Moving Objects

The PlanAhead tool provides a few methods for selecting objects:

- You can select a single object. Click the primary object to select it in the current view. When selected in any view, objects also become cross-selected as appropriate in other open views, and in other open designs of the same project.

- When you have selected a primary object, the PlanAhead tool can also select any connected or associated secondary objects. This feature can be configured through selection rules. See Setting Selection Rule Options, page 175 for more information.

- You can select multiple objects. Click to select the first object, then press and hold the **Ctrl** key and click to select multiple additional objects.

- You can select a range of objects, from a list of objects for instance. Click on and select the primary object, then press and hold the **Shift** key and select the last object in a range of elements from a tree or table view.

- You can select all the objects in an area of a graphical view. See Using the Select Area Command, page 118 for more information.

When objects overlap, a priority scheme is used where the smaller size objects are selected. If objects become difficult to select in the Device view, use the Physical Constraints or Netlist views. You can select objects from either of these two views regardless of the Selection Rule in the Options dialog box (see Setting Selection Rule Options, page 175).

If you experience difficulty selecting the desired object, use the **Select** command in the right-click popup menu, as shown in Figure 4-25, page 117, to select a specific object from a list of objects at the location of the currently selected item.
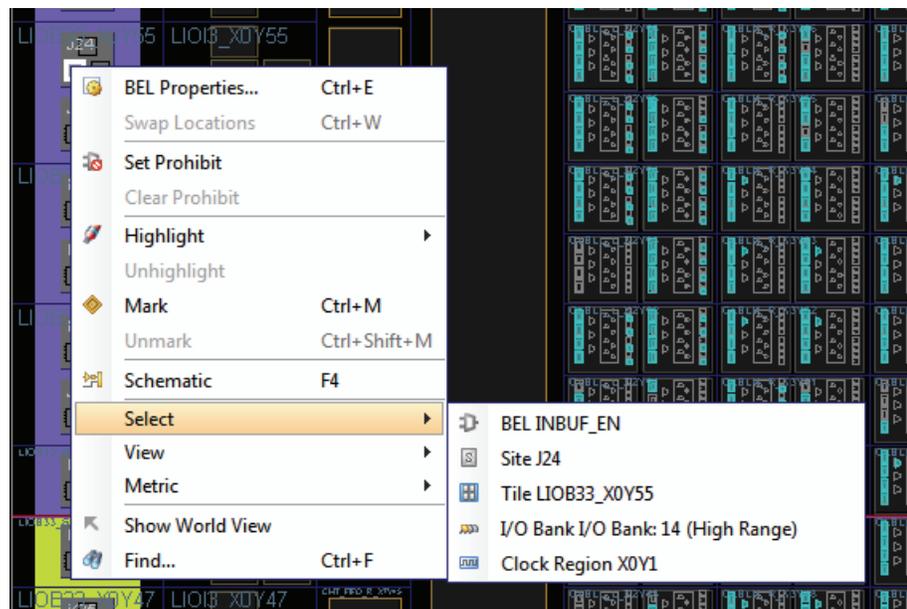
Figure 4-25: **Select > Objects**

Using the Select Area Command

You can draw a rectangle in any of the workspace views to select objects:

1. Use one of the following:

   • **View > Select Area** from the main menu, or

   • Click the **Select Area** toolbar button.

   Objects that the rectangle surrounds or touches are listed in the Select Area dialog box, with which you can filter selection by type, shown in Figure 4-26.
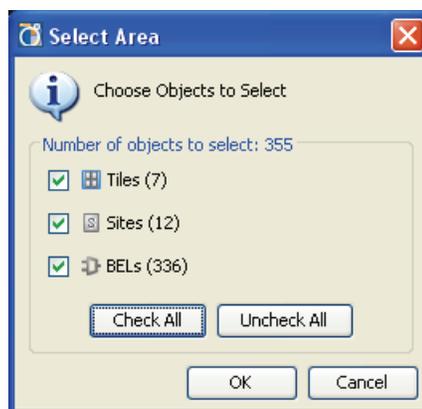
Figure 4-26: **Select Area Dialog Box**

2. Turn off the check box to filter Object types from selection.

3. Click **OK** to select all of the checked objects.

Selecting Primitive Parent Modules

The Select Primitive Parents command lets you select the parent modules for selected primitive logic. To access the command, select **Select Primitive Parents** from the popup menu (available in most views).

Floorplans are easier to maintain when logic modules are assigned to Pblocks rather than primitive logic instances.

Select a group of timing paths to select the primitive logic instances contained on the paths. The Select Primitive Parents command selects the parent modules automatically for selected primitive logic.

The originally selected primitive logic is no longer selected. The command interpolates what is selected and returns with only modules selected, unless ROOT level logic was originally selected.

If you select modules, the command does not select parent modules; the pre-selected modules remain selected.

Setting Selection Rules

When selecting an object, associated or connected objects might also become selected. For example, selecting a Pblock can also select the netlist instances assigned to the Pblock. Selecting a Port object may also select the Pin object of the port.

Use the **Tools > Options > Selection Rules** command to control what secondary elements are selected when you select a primary object. Figure 4-27 shows the Selection Rules dialog box.
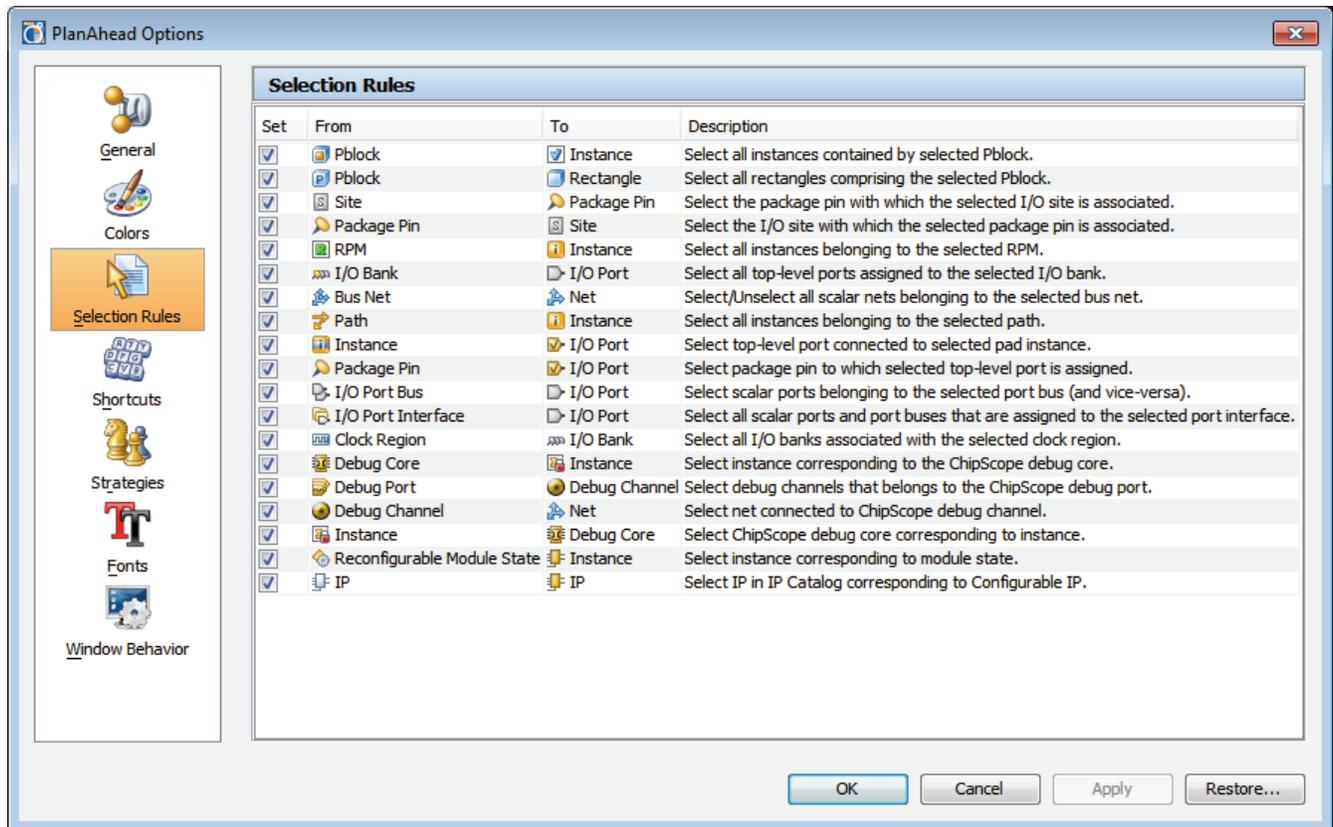
Figure 4-27: **PlanAhead Options: Selection Rules**

The default selection rules enable the PlanAhead tool to operate in the most efficient manner. You can change these selection rules when you are having trouble selecting a specific object.

To enable or disable automatic selections, click the **Set** column heading.

- Enabling a selection rule causes the PlanAhead tool to select the secondary "To" object types that are associated with the "From" object that is the primary selection.

- Disabling the selection rule allows the PlanAhead tool to only select the primary "From" object when it is selected.

Using the Selection View

The Selection view, as shown in , displays the list of currently selected objects. You can sort, de-select, or mark objects from this view. The list is updated dynamically as you manipulate objects.

Use **Window > Selection** to open the Selection view.
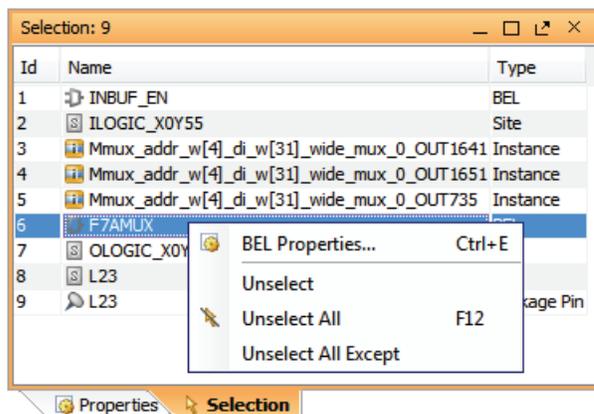
Figure 4-28: **Selection View**

- To sort objects by **Name**, **ID**, or **Type**, click the banner of the sort column. In the figure you can see items are sorted in descending order by Type.

- To remove selected items from the list, use the **Unselect**, **Unselect All,** or **Unselect All Except** commands from the popup menu.

Select multiple of objects using the **Ctrl** and **Shift** keys, or the **Select Area** command. The total number of objects selected displays in the view banner.

Fitting the Display to Show Selected Objects

When selecting objects directly in the workspace view, or indirectly by cross-selecting from other views, you might want to have the display updated to focus on the selected object. Various workspace views offer different methods for doing this, as described in the following sections.

Automatically Displaying Selected Objects

When selecting objects in the Sources view or Netlist view, the objects become cross-selected in the graphical workspace view, such as the Device view or the Schematic view. However, these objects can be small and difficult to see because of the volume of information that may be displayed, or the current zoom-level of the view.

The PlanAhead tool provides a **Auto Fit selection** command on the toolbar menu of graphical workspace views to help display selected objects. Enable this command so that the workspace view is always redrawn to display the selected object. Disable the command so the view is not redrawn every time an object is selected.

The PlanAhead tool automatically corrects the zoom factor to allow multiple objects to be displayed when more than one object is selected.

Fit Selection

Views in the workspace also have an interactive zoom command, **Fit Selection**, to fit all selected objects. Use this command when the **Auto Fit Selection** mode is disabled. To zoom fit the selected objects, use one of the following methods:

- Select **View > Fit Selection**.

- Press **F9**.

- Click the **Fit Selection** toolbar button.

Automatically Scrolling to Selected Objects

Some views in the workspace, such as the Clock Resources view, have an option to automatically scroll to a selected object. This allows the displayed view to be updated to focus on an object that is cross-selected from a different view.

To enable this mode, use the **Automatically scroll to selected objects** toolbar button in the views where it is available: the Sources view, the Netlist view, and the Clock Resources view. Enable or disable this command as needed.

Moving Selected Objects

You can move selected objects in graphical workspace views:

- To move objects:
  - Hold the left mouse down to select an object.
  - Drag the object to move it.
  - Release the object to drop it on a new location.

  The cursor changes to a hand symbol when the move mode is activated.

- To move multiple instances at the same time:
  - Press and hold the **Ctrl** key to enable multiple selections.
  - Left-click multiple times to select multiple objects.
  - Hold down the left mouse button and drag and drop the selected objects to a new location.

Highlighting Selected Objects

The PlanAhead tool has a highlight mechanism that lets you selectively highlight objects. Highlighting lets you display multiple placement groups at the same time using different colors.

Highlighted objects remain highlighted even when you unselect them. They are highlighted in all applicable views, including the Schematic.

You can highlight any number of selected objects, as follows:

- Select objects in a graphical view, such as the Device view, and use the **View > Highlight > Highlight Color** command from the main menu.
- Select objects in a graphical view, such as the Device view, and use the **Highlight > Highlight Color** command from the popup menu.
- Select a Pblock in the Device view and use the **Highlight > Highlight Color** command from the popup menu.
- Select a module, or primitives, in the Netlist view and use the **Highlight > Highlight Color** command from the popup menu. For more information, see Using Select Primitives and Highlight Primitives Commands, page 348.

Marking and Unmarking Selected Objects

The PlanAhead tool lets you place a Mark symbol and remove the Mark in all applicable views for selected objects.

Marking Objects

Marking a selected object is helpful when displaying small objects that you want to see in the Device view.

To mark selected objects, select the object and then use the **View > Mark** command from the main menu, the **Mark** command from the popup menu, or use the **Ctrl+M** keyboard shortcut.

The **Mark** command is also available in other views, including the Netlist view, Physical Hierarchy view, and Timing Report view. Figure 4-29 shows a timing path marked from the Timing Report view. The start point of the timing path is marked in green, the end point in red, and the through points are marked in yellow.

Figure 4-29: **Marked Timing Path Symbols in Device View**

Unmarking Objects

You can remove marks on selected object or all objects, using one of the following methods:

- Choose **Unmark** to unmark the selected instance.
- Click **Unmark All** to unmark all instance.

Using the Find Commands

The PlanAhead tool lets you selectively search for design objects, such as instances or nets in open designs, using the Find command. To invoke the **Find** command you must open an RTL, Netlist, or Implemented Design.

1. Select one of the following:
   - **Edit > Find** from the main menu.
   - **Ctrl+F** keyboard shortcut.
   - The **Find** command on the main toolbar menu.

   The Find dialog box opens, as shown in Figure 4-30, page 123.
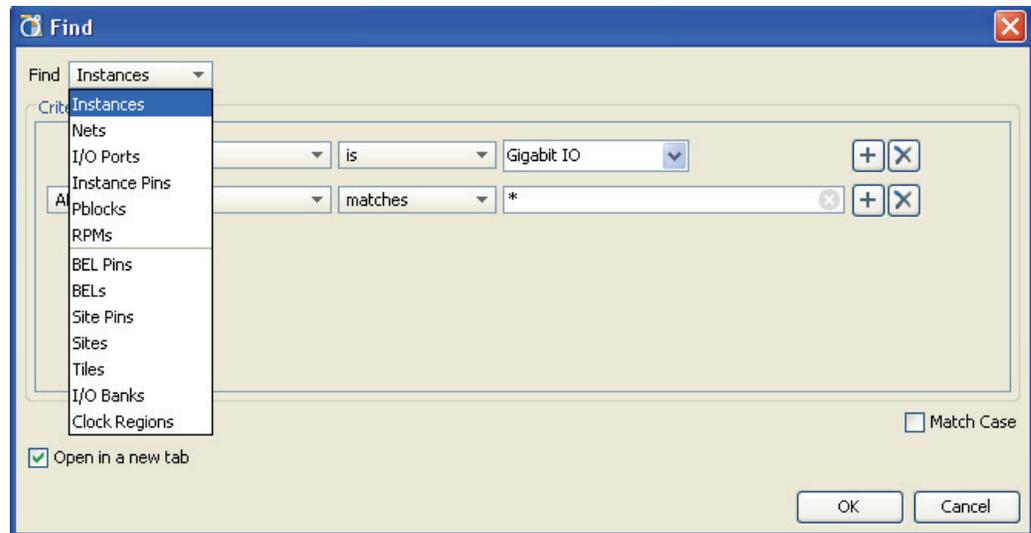
Figure 4-30: **Find Dialog Box**

2. Edit the search fields:

- **Find**: Select the physical or logical object type (Instances, Nets, Pblocks, Sites, etc.) to find.

- **Criteria**: For each object type, a different set of search parameters are available in the dialog box.

  - In the first field, select the attribute to use to search for objects. For example, Name, Status, Type, Parent Pblock, Module, or Primitive count. The values in this field depend on the object type in the Find field.

  - In the second field, you can define search criteria such as: matches, does not match, contains, and does not contain.

  - Use the third field to define the value of the attribute based on the search criteria. You can use the asterisk (*) as wildcards in search strings.

3. Optionally, click the + button to define additional search criteria.

A new row of search criteria fields display in the **Find** dialog box. An **AND/OR** field displays where you can define the additional search criteria as shown in .

New search criteria are added parenthetically with prior search criteria to define the Find query. The first two search criteria are grouped together, and nested as each new search criteria is added. Adding multiple search criteria results in grouping them as follows:

```
((((crit1 OP crit2) OP crit3) ...) OP critN)
```

Where:

- `crit1` is the first search criteria,

- `critN` is the last search criteria,

- `OP` is the boolean AND or OR operation defining the relationship between the search criteria.
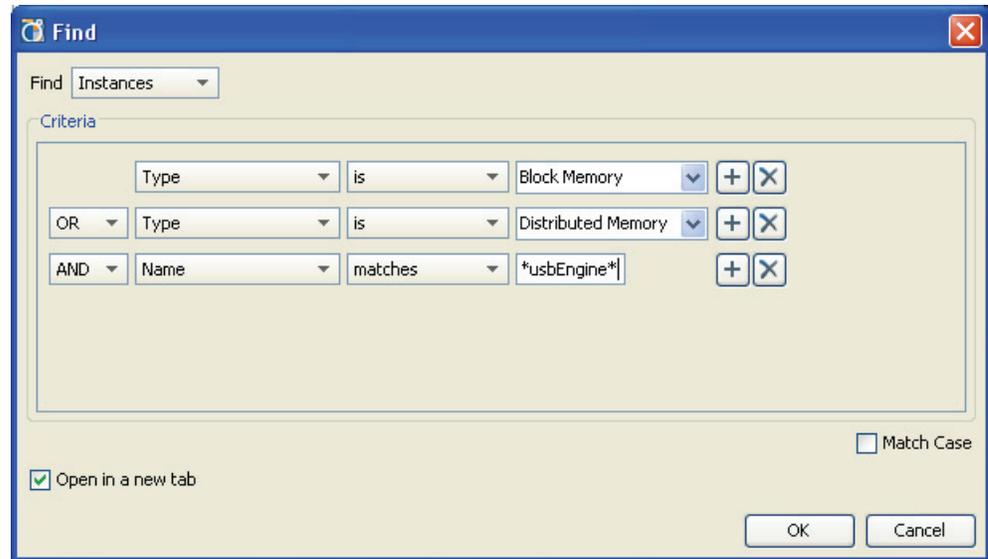
Figure 4-31: **Searching for Objects with Additional Search Criteria**

The query defined in Figure 4-31 should be read as:

```
Find Instances where
((Type is Block Memory OR Type is Distributed Memory)
AND Name matches *usbEngine*)
```

This query finds all block memories and distributed memories in the design, and then filters the list to return only those named *usbEngine*.

4. Click the + or **X** button to add or remove search criteria rows.

5. Click the **Match Case** checkbox to make all search criteria case sensitive.

6. Enable or disable the **Open in a new tab** checkbox to have the results open a new tab in the Find Results view, or replace existing results.

7. Click **OK** to perform the search.

The combined search results are discussed in Using the Find Results View.

Using the Find Results View

The objects matching the search criteria defined in the Find dialog box display in the Find Results view after you click **OK**. Figure 4-32, page 125 shows the Find Results view.
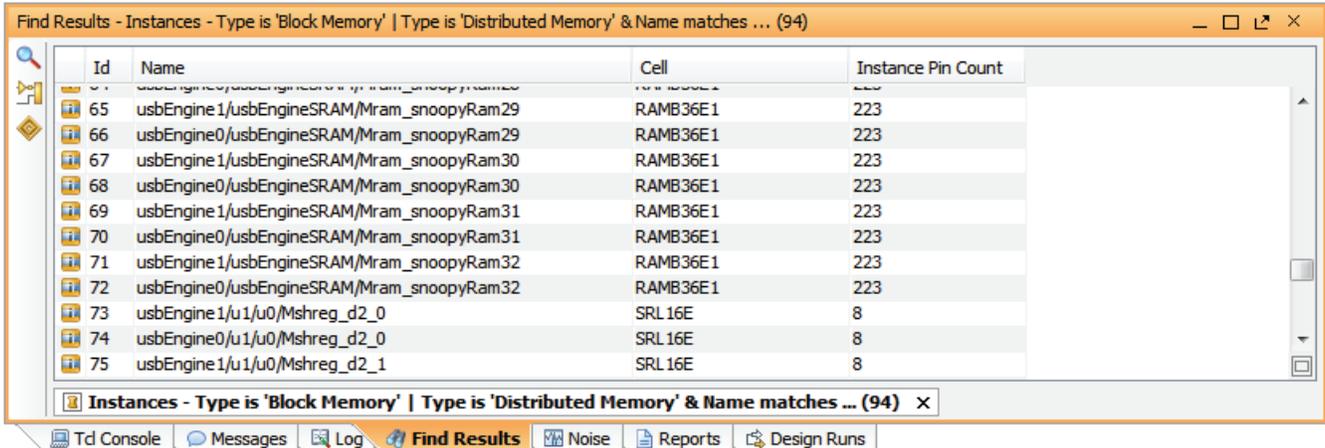
Figure 4-32: **Find Results View**

The PlanAhead tool creates a new Find Results tab each time you run the Find command, which is named according to the Search criteria and number of objects found.

You can select objects directly from the Find Results dialog box, and those objects are selected in other views as well. You can select multiple elements by using the **Shift** or **Ctrl** keys. Additional commands are available from the popup menu.

To sort columns:

* Click any of the column headers.

* Press the **Ctrl** key and clicking a second column header.

To close the Find Results views, click the **X** icon in a **Find Results** tab.

Searching and Replacing in Source Files

While the Find command searches for design objects, such as nets or instances, in open designs, you can use the Find, Find in Files, and Replace in Files commands to search for text strings in source files.

Use the **Find** command to search within a single open source file. Select the **Find** command from within the Text Editor popup menu, or from the toolbar icon.

The Find toolbar opens at the bottom of the Text Editor. Enter a text string to find in the source file. You can also optionally choose to highlight all occurrences of the string in the open file. Figure 4-33 shows an example of the Find command toolbar.
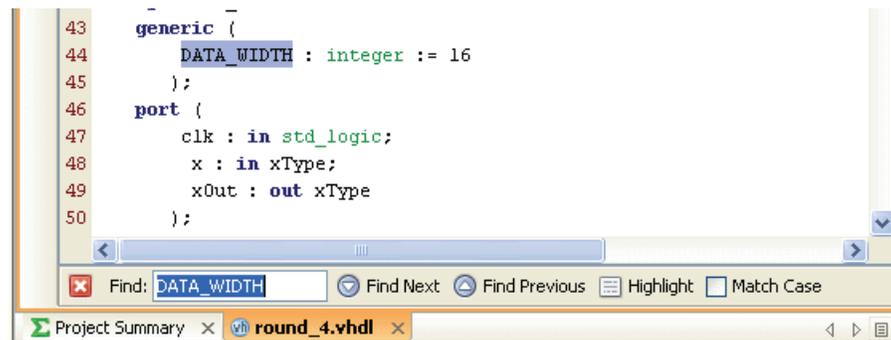
Figure 4-33: **Using Find Feature in an Open Source File**

You can also use Find in Files to search for a given text string in an open source file, or a specified set of source files. Select the **Find in Files** command from the Text Editor popup menu, or from the toolbar icon, or select **Edit > Find in Files** from the main menu.

You can:

- Enter any text string, including wildcards (*), as search criteria.
- Use the filtering options to search source files, constraint files, and report files.

Figure 4-34 shows the Find in Files dialog box.
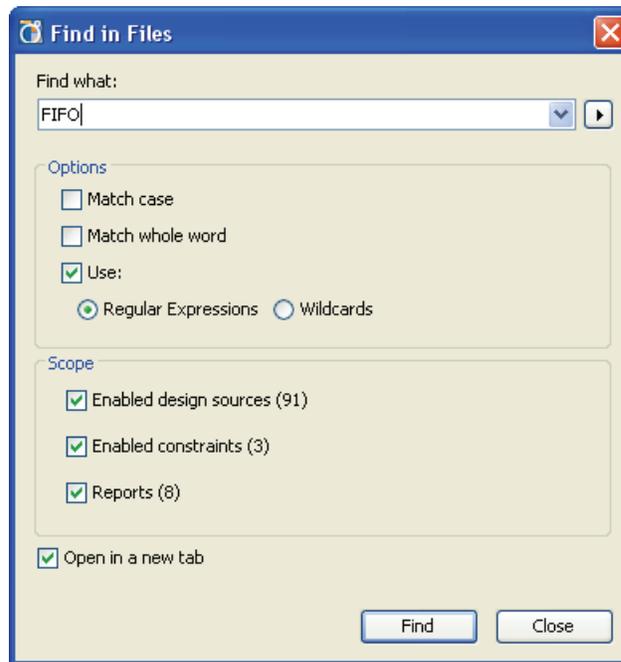
Figure 4-34: **Find in Files Dialog Box**

Use the Find in Files command to search for a specified text string as follows:

1. Enter the text string to locate in the Find What field.
2. Enable options as needed:
   - **Match case**: Makes the search case sensitive.
   - **Match whole word**: Do not search for substrings.
   - **Use**: Regular expressions or Wildcard based search.
   - **Scope**: Search enabled design sources, constraint files, open reports.
   - **Open in a new tab**: Open the results in a new view, or replace an existing view.
3. Click **Find**.

The search results display in the Find in Files Results view as a list of files that contain the search string and the number of occurrences in each file.

Click a file name in the Results view to load the source file into the Text Editor and highlight the string. Figure 4-35, page 127 shows the Results view of a sample search.
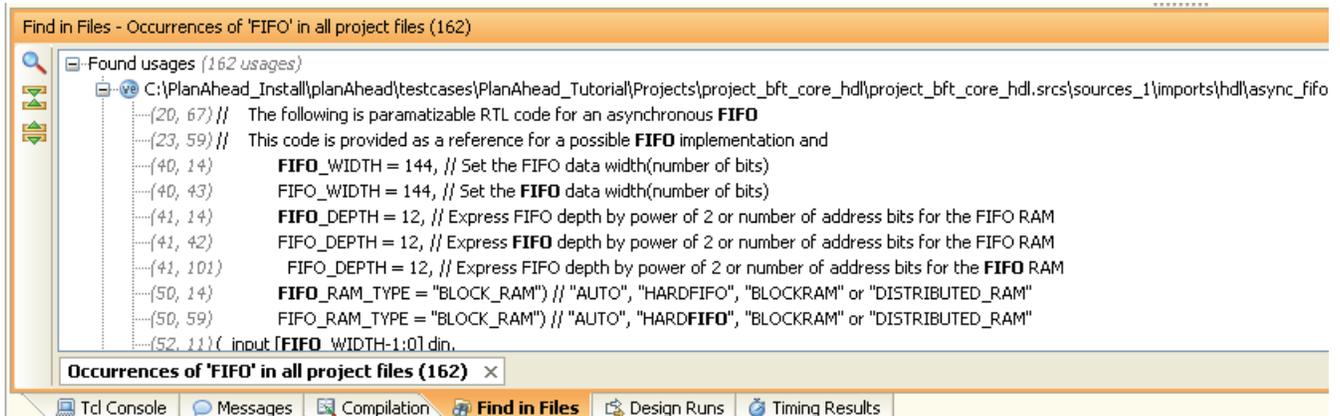
Figure 4-35: **Find Results View**

Use the Replace in Files command to search for any given text string in a selected set of source files, and replace it with a different text string. Select the **Replace in Files** command from the Text Editor popup menu, or select **Edit > Replace in Files** from the main menu.

Figure 4-36, page 127 shows the Replace in Files dialog box.
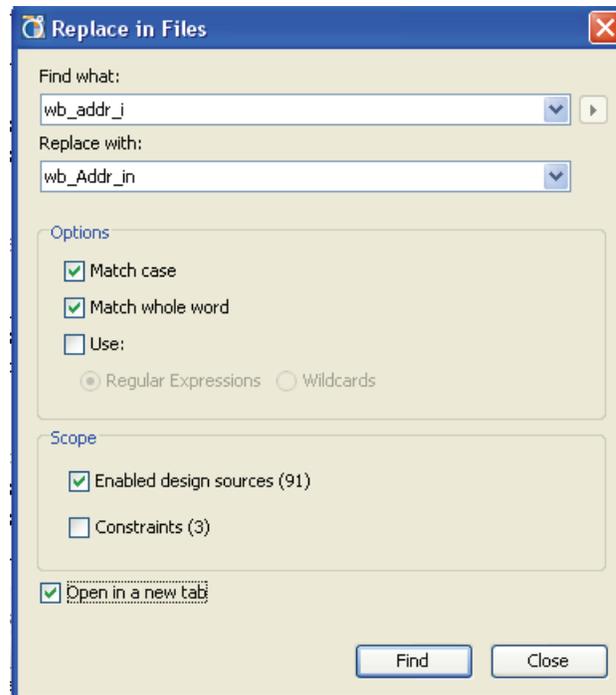
Figure 4-36: **Replace in Files Dialog Box**

Use the Replace in Files command to search and replace a specified text string as follows:

1. Enter the text string to locate in the Find What field.

2. Enter replacement text in the Replace with field.

3. Enable options as needed:
    - **Match case**: Makes the search case sensitive.
    - **Match whole word**: Do not search for substrings.
    - **Use**: Regular expressions or Wildcard based search.
    - **Scope**: Search enabled design sources, and constraint files.
    - **Open in a new tab**: Open the results in a new view, or replace an existing view.
4. Click **Find**.

The search results display in the Find in Files Results view as a list of files that contain the search string and the number of occurrences in each file.

Figure 4-37, page 128 shows the results of an example search.

- You can select any occurrence of the text in the list to **Replace Selected**.
- You can use **Replace All** occurrences of the specified text.
- You can **Cancel** the search.
- Clicking on an occurrence of the text in the Results view will highlight that occurrence in the Text Editor.
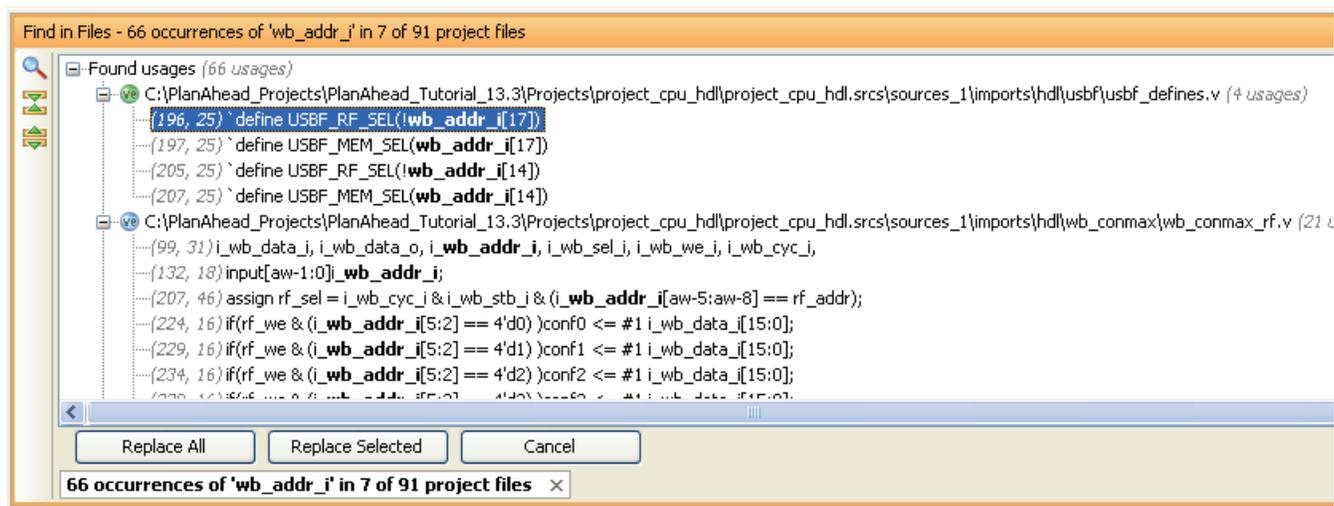
Figure 4-37: **Replace in Files Results**

Using Common Views

The following subsections describe the common views.

Using the Sources View

When design sources, constraint files, simulation sources, and IP cores are added to a project, they appear in the Sources view of the project. You can use the Sources view to manage project source files; adding, removing and reordering the sources to meet specific design requirements.

The *Design Sources* folder contains source file types which include: Verilog, VHDL, NGC/NGO, EDIF, as well as IP cores, DSP modules, and Embedded Processors. Constraint files are assigned to constraint sets and are displayed under the *Constraints* folder. Refer to Managing Project Sources, page 52 for more information.

Note: Module-level Netlist Constraints File (NCF) show as design sources adjacent to their associated cores, and are read-only.

Generally, the Sources view is open in the PlanAhead tool whenever a project is open. However, to open the Sources view select **Windows > Sources**. Figure 4-38 shows an example of the Sources view.
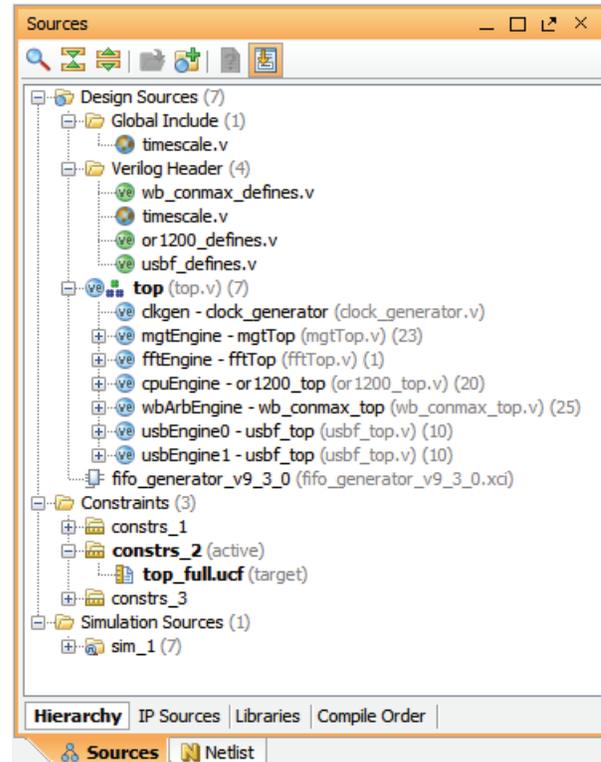
Figure 4-38: **Sources View**

The Sources view has four tabs to display the source files in different ways:

- The Hierarchy tab displays the hierarchical view of the design sources, starting with the top module. The top module defines the hierarchy of the design for compilation, synthesis and implementation. It is identified with a special icon in the Sources view as shown in Figure 4-38. The PlanAhead tool automatically detects the top module, but you can also manually define the top module using the **Set as Top** command (see Sources View Popup Menu, page 131).

    Note: The **Filter Sources by missing files or instantiations** command, on the Sources view toolbar menu, becomes enabled when a file, module definition, or instantiation of a module is missing in the design hierarchy. This can be helpful in debugging missing information.

- The IP Sources tab displays:

    - IP cores imported from the Xilinx IP catalog. See Managing IP Cores, page 66 for more information.

    - DSP modules imported from Sysgen. See Managing DSP Sources, page 72 for more information.

    - Embedded processor designs imported from XPS. See Managing Embedded Processor Sources, page 75 for more information.

- The Libraries tab displays the sources sorted into the various libraries. A green dot next to the source file name indicates that the source file is copied into the local project directory. When source files display in red, the software could not find the required files.

- The Compile Order tab displays source files in the order in which they will be compiled, first to last. In this case the top module is usually the last. You can allow the PlanAhead tool to automatically determine the compile order based on the defined top module and the elaborated design. You can also manually control the compile order of the design by setting the **Hierarchy Update** command and reordering the source files (see Sources View Popup Menu, page 131).

Using the Sources View Commands

The Sources view has a toolbar menu, and a popup menu that displays when you press the right mouse button. You can add, view, or modify source files using the Sources view toolbar and popup menu commands.

Sources View Toolbar Menu

The toolbar menu, as shown in Figure 4-38, page 129, contains the following commands:

- **Show Search**: Opens the Search bar to allow you to quickly locate objects in the Sources view. This command is also accessed through the **Alt+/** keyboard shortcut.

- **Expand all**: Expands all hierarchical tree objects to see all elements of the Sources view.

- **Collapse all**: Collapses the hierarchical tree objects to display only the top-level objects.

- **Open Selected Source Files**: Opens selected RTL source files, or constraint files, in the text editor. Selected IP cores will open in CORE Generator wizard, DSP modules will open in Sysgen, and embedded processors will open in XPS. For more information, refer to Chapter 3, Working with Projects, or Using the Text Editor in Chapter 5.

- **Add Sources**: Lets you add or create RTL source files, simulation source files, Constraint files, add existing IP, add DSP modules, or embedded processors to the project.

- **Filter Sources by missing files or instantiations:** Filter sources to display missing files or missing instances. This command is enabled when a file, module definition, or instantiation is missing in the design hierarchy. When you select the command, the Sources view is filtered to display the missing files or modules, making it much easier to locate and fix the problem. When the command icon is grey, or disabled, the PlanAhead tool has not identified a problem with the design hierarchy.

- **Automatically Scroll to Selected Object**: Causes the Sources view to update to focus on the currently selected object. This can be useful on large designs with many source files. This feature is on by default.

Sources View Popup Menu

The Sources view popup menu commands are:

	Source Node Properties...	Ctrl+E
	Open File	Alt+O
	Go To Instantiation	F7
	Launch Timing Simulation	
	Replace File...	
	Copy File Into Project	
	Copy All Files Into Project	Alt+I
✕	Remove File from Project...	Delete
	Enable File	Alt+Equals
	Disable File	Alt+Minus
	Move to Simulation Sources	
	Hierarchy Update	▶
	Refresh Hierarchy	
	Set as Top	
	Set Global Include	
	Set Library...	Alt+L
	Set File Type...	
	Set Used In...	
	Edit Constraints Sets...	
	Edit Simulation Sets...	
	Add Sources...	Alt+A

- **Source Node Properties**: Invokes the Source Node Properties view. This is also called Source File Properties on the Libraries and Compile Order tab in the Sources view. See Viewing Source File Properties, page 134.

- **Open File**: Opens the selected RTL or constraint file(s) in the Text Editor view. Selected IP cores will open in CORE Generator wizard, DSP modules will open in Sysgen, and embedded processors will open in XPS.

- **Go To Instantiation**: Opens the RTL source file for the selected cell or module.

- **Launch Timing Simulation**: Run timing simulation on the selected simulation sources. See Performing Timing Simulation in Chapter 11.

- **Replace File**: Replaces the specified source file with another file.

- **Copy File into Project**: Copies selected source files and directories into the project directory. This command is only enabled when the selected source file is not currently local to the project. See Using Remote Sources or Copying Sources into Project in Chapter 3.

- **Copy All Files Into Project**: Copies remotely referenced source files into the local project directory. This command is only available when the source files are not local to the project.

- **Remove File from Project**: Deletes the selected source files from the project, and optionally removes the files from the local project disk location.

- **Enable File**: Sets the source file status to active for the project. You can toggle source files between *enabled* and *disabled* to define different design configurations.

  Note: The Enabled attribute can also be set from within the Source File Properties view (see Viewing Source File Properties, page 134).

  - **Disable File**: Sets the source file status to inactive for the project. You can toggle source files between *enabled* and *disabled* to define different design configurations. Disabled source files appear in a shaded grey color in the Sources view.

- **Move to Simulation Sources**: Relocates currently selected source files into the simulation set. If there are more than one simulation sets, the application prompts you to select the simulation set to use.

- **Move to Top**: In the Compile Order tab of the Sources view, this relocates the currently selected source file to the top of the source file list. The compilation and synthesis of source files is handled in the order listed in Compile Order tab, from top to bottom, so the order of files will affect the elaboration, synthesis, and simulation results.

  The file order shown in the Compile Order tab will be automatically updated or can be manually defined depending on the setting of the **Hierarchy Update** command as discussed below.

  Note: Except for Move to Simulation Sources, the *Move...* commands are not available from the Hierarchy tab of the Sources view.

  - **Move Up**: Moves the currently selected source file up in the source file list.
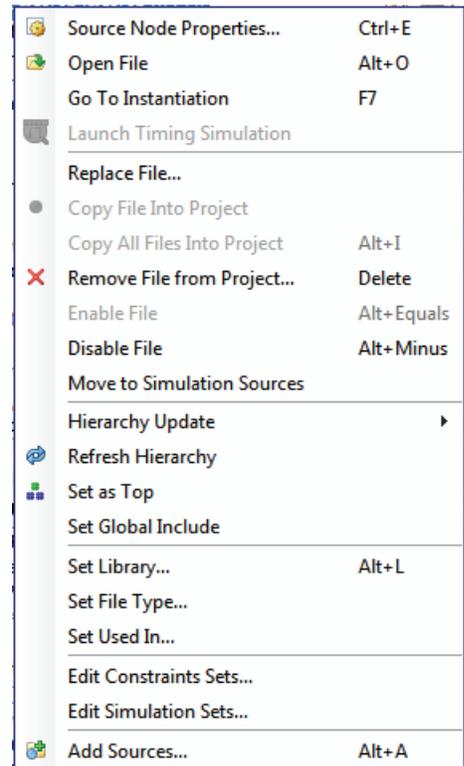
- **Move Down**: Moves the currently selected source file down in the source file list.
- **Move to Bottom**: Relocates the currently selected source file to the bottom of the source file list.

- **Hierarchy Update**: In the Hierarchy tab of the Sources view, this determines how the PlanAhead tool responds to changes of the source files such as redefined top module, added or removed files, or changed file order. The three possible settings for this command are:
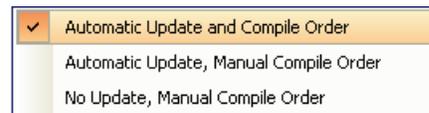
✓	Automatic Update and Compile Order
	Automatic Update, Manual Compile Order
	No Update, Manual Compile Order

- **Automatic Update and Compile Order**: Specifies that the hierarchy view of the design and the compilation order should be automatically updated as source files are changed. The PlanAhead tool automatically identifies and sets the best top module candidate. The compile order is also automatically managed, as the top module file and all sources that are under the active hierarchy are passed to synthesis and simulation in the correct order. The files that are outside of the hierarchy defined by the top module are not sent.

- **Automatic Update, Manual Compile Order**: Specifies that the hierarchy view of the design should be automatically updated as source files are changed, but that the compilation order is determined manually. All files in the project are passed to synthesis and simulation. The compilation order is manually defined by ordering the files using the **Move to Top**, **Move Up**, **Move Down**, and **Move to Bottom** commands from the Compile Order tab.

- **No Update, Manual Compile Order**: Specifies that the hierarchy view should not be automatically updated, and that the compilation order is determined manually. To update the design hierarchy in this mode, use the **Refresh Hierarchy** command.

- **Refresh Hierarchy**: Updates the design hierarchy to reflect the latest source file changes and top module definition. Use this command to manually refresh the hierarchy as needed.

- **Set as Top**: Specifies the Top Module to define the starting point for elaboration of the design hierarchy for synthesis and simulation purposes. See Specifying the Top Module and Reordering Source Files, page 62.

  The current top module is identified by a special icon in the Hierarchy tab of the Sources view.

  Note: The selected top is automatically reset to the best candidate if the specified top module cannot be found in the design source files, and the hierarchy update mode is set to automatic.

- **Set Global Include**: Defines the specified file as a global include file. This command is only available for use with Verilog source files. See Defining Global Include Files, page 198 for more information.

  Note: The Global Include attribute can also be set from within the Source File Properties view (see Viewing Source File Properties, page 134).

  - **Clear Global Include**: Clears the Global Include attribute from the selected Verilog source file.

- **Make active**: This makes the selected Constraint Set the active constraints for synthesis or implementation.

- **Set as Target Constraint File:** For Constraint Sets with multiple constraint files, this command lets you specify the constraint file that the PlanAhead tool targets to write newly-created constraints. See Managing Constraints in Chapter 3.

- **Set Library**: Lets you specify a library for the selected RTL source file(s). You can choose from a list of libraries that are currently defined in the project, or type a new library in the text entry field. Entering a new library adds it to the list of currently defined libraries.

- **Set File Type**: Define the type of the currently selected file or files. The PlanAhead tool will automatically recognize the type of a file as it is added to the project based on appropriate file extensions. However, you can use the Set Type command to redefine the file type in cases of non-standard file extensions.
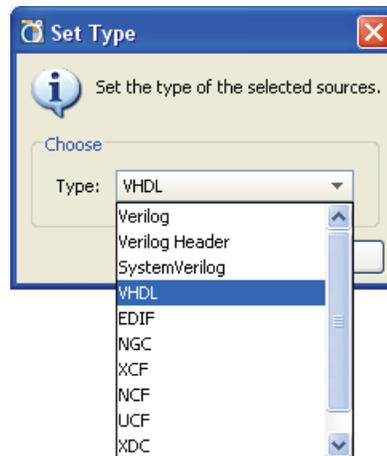
Figure 4-39:   **Set Type**

- **Set Used In**: Specify what tools the file is used for. You can specify a source file to be used, or not used, during synthesis, simulation, or implementation. Disabling a source file for a particular tool will prevent that file from being used by that tool.

    For example, if you set a source file as not used in Synthesis, and then open the Elaborated Design, you will see a black box for that source file. Disabling an EDIF or NGC source file from implementation will prevent it from being used during implementation.

    However, a Verilog or VHDL source file enabled for synthesis but disabled for implementation will cause the source to be synthesized into the output netlist, and thus used during implementation. To prevent RTL source files from being used during implementation, you must disable the source file from synthesis.

    Note: The Library, Type, and Used In attributes can be set from within the Source File Properties view (see Viewing Source File Properties).
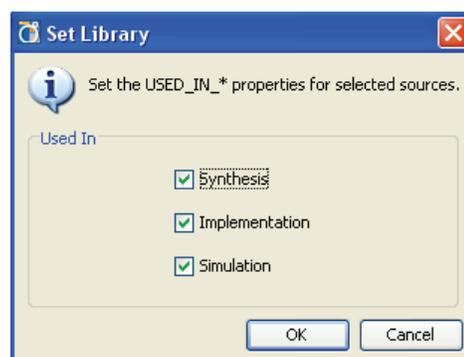
Figure 4-40:   **Set Used In**

- **IP Sources Commands**: The following commands are available in the Sources view popup menu when an IP core is selected in the Sources view. See Managing IP Cores, page 66 for more information.

    - **Re-customize IP**: Opens the CORE Generator interface for the IP core to allow attributes of the IP to be modified.

- • **Generate Output Products:** Generate target data for the IP core as needed for the design.

  - • **Reset Output Products:** Remove the current target data to allow it to be regenerated as needed.

  - • **Upgrade IP**: Upgrade the IP from an older version to the latest version if one is available.

  - • **Open IP Example Design**: Open an example project for the IP core if example target data is provided by the core, and has been generated.

  - • **IP Documentation**: Provides access to IP documentation sub-menu.

    - **View Product Guide**: Open the PDF datasheet from the IP catalog.

    - **View Version Information**: Open the HTML version information from the IP catalog.

    - **View Product Web Page**: View the web page for the IP core if one is available.

    - **View Answer Record**: Search the Support database for Answer Records associated with the IP core.

- • **DSP Sources and Embedded Design Sources Commands:** The following commands are available in the Sources view popup menu when a DSP module or an embedded processor sub-design is selected in the Sources view. See Managing DSP Sources, page 72 or Managing Embedded Processor Sources, page 75 for more information.

  - • **Create Top HDL**: Create a top-level Verilog or VHDL module containing the selected DSP or Embedded Processor.

  - • **View Instantiation Template**: Open the Instantiation Template for the DSP or Embedded Processor Design to instantiate it into another RTL file.

  - • **Create Testbench**: Create the simulation test bench for the selected DSP module or Embedded Processor Design.

  - • **Generate Output Products:** Generate target data for the DSP module or Embedded Processor Design as needed for the design.

  - • **Reset Output Products:** Remove the current target data to allow it to be regenerated as needed.

  - • **Export Hardware for SDK** - See Exporting Hardware in Chapter 3.

- • **Edit Constraint Sets:** Lets you create and modify constraint sets.

- • **Edit Simulation Sets:** Lets you create and modify simulation sets.

- • **Add Sources**: Lets you add or create RTL source files, simulation source files, Constraint files, existing IP, DSP modules, or EMbedded Processor Sub-Designs to the project. See Managing Project Sources in Chapter 3.

Viewing Source File Properties

Selecting an RTL source file in the Sources view displays information in the Source File Properties view. To view source file properties:

1. In the Sources view, select a source file and select the **Source File Properties** view.
   Figure 4-41, page 135 shows the Source File Properties view for a selected Verilog file.
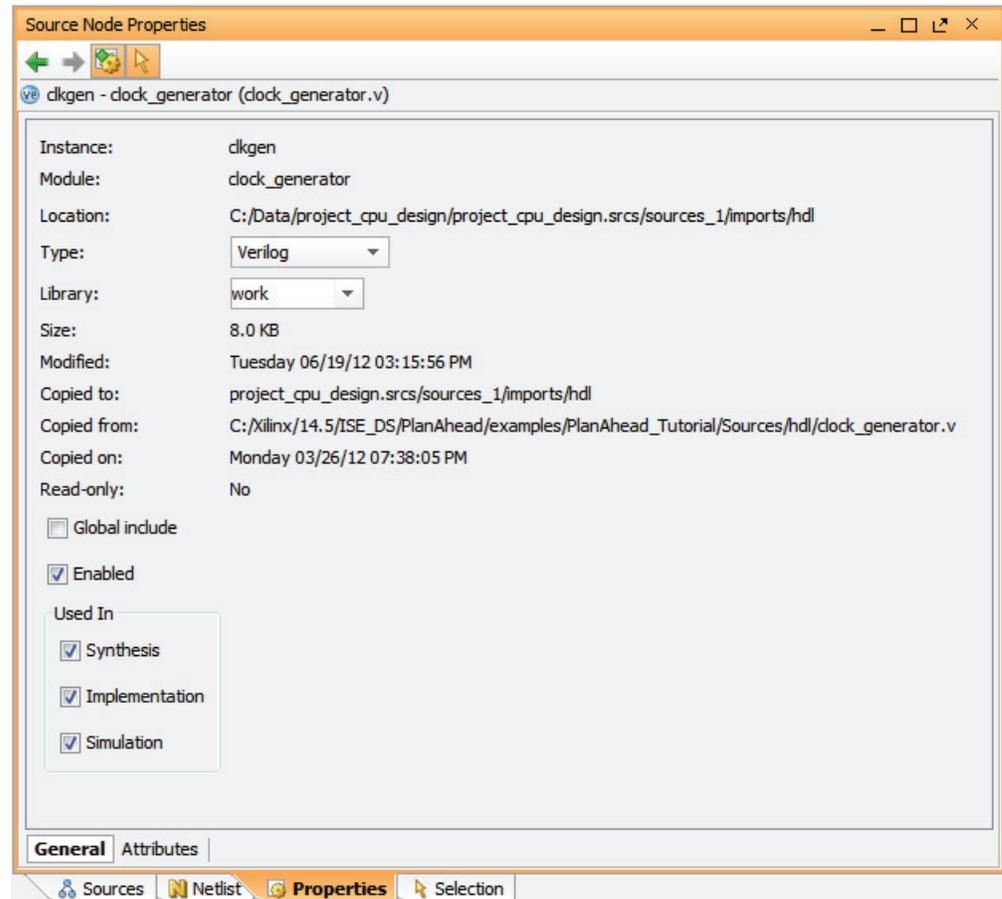
Figure 4-41: **Viewing Modified Source File Properties**

The file information includes location, type, library, size, modified timestamp date, location copied from, copy date, and parent module.

- You can change the file type using the **Type** option. This is useful in cases where files could have non-standard extensions and the file type is not properly detected.

- Select a new target library for a source file by clicking the pulldown **Library** menu to select from the list of defined libraries, or simply type a new name in the **Library** name field.

- Use the **Global Include** checkbox to enable or disable Verilog source files as global include files. The **Global Include** switch forces the selected file to be compiled at the start of the compile order for elaboration and synthesis. See Defining Global Include Files in Chapter 6 for more information.

- Deselect the **Enabled** checkbox to disable a specific source file from the design. Disable files are still listed in the source files, but are not considered part of the design for elaboration or compilation.

- Specify a source file to be **Used In**, or not used, during synthesis, simulation, or implementation. Disabling a source file for a particular tool will prevent that file from being used by that tool. See the description of the Used In command in Using the Sources View Commands, page 130 for more information.

2. Click **Apply** or **Cancel** to implement or discard any changes to the Source File Properties.

Using the Device View

The Device view is the main graphical interface used for multiple purposes related to design analysis and floorplanning. For more information, see:

- Chapter 7, Synthesized Design Constraints and Analysis
- Chapter 8, I/O Pin Planning
- Chapter 11, Analyzing Implementation Results
- Chapter 10, Floorplanning the Design
- The Device view displays the FPGA device resources including the FPGA logic, clock regions, I/O pads, BUFGs, DCMs, Pblocks, instance locations, and net connectivity. The locations on the device where specific logic can be assigned are called *Sites*. Figure 4-42 shows a Device view.
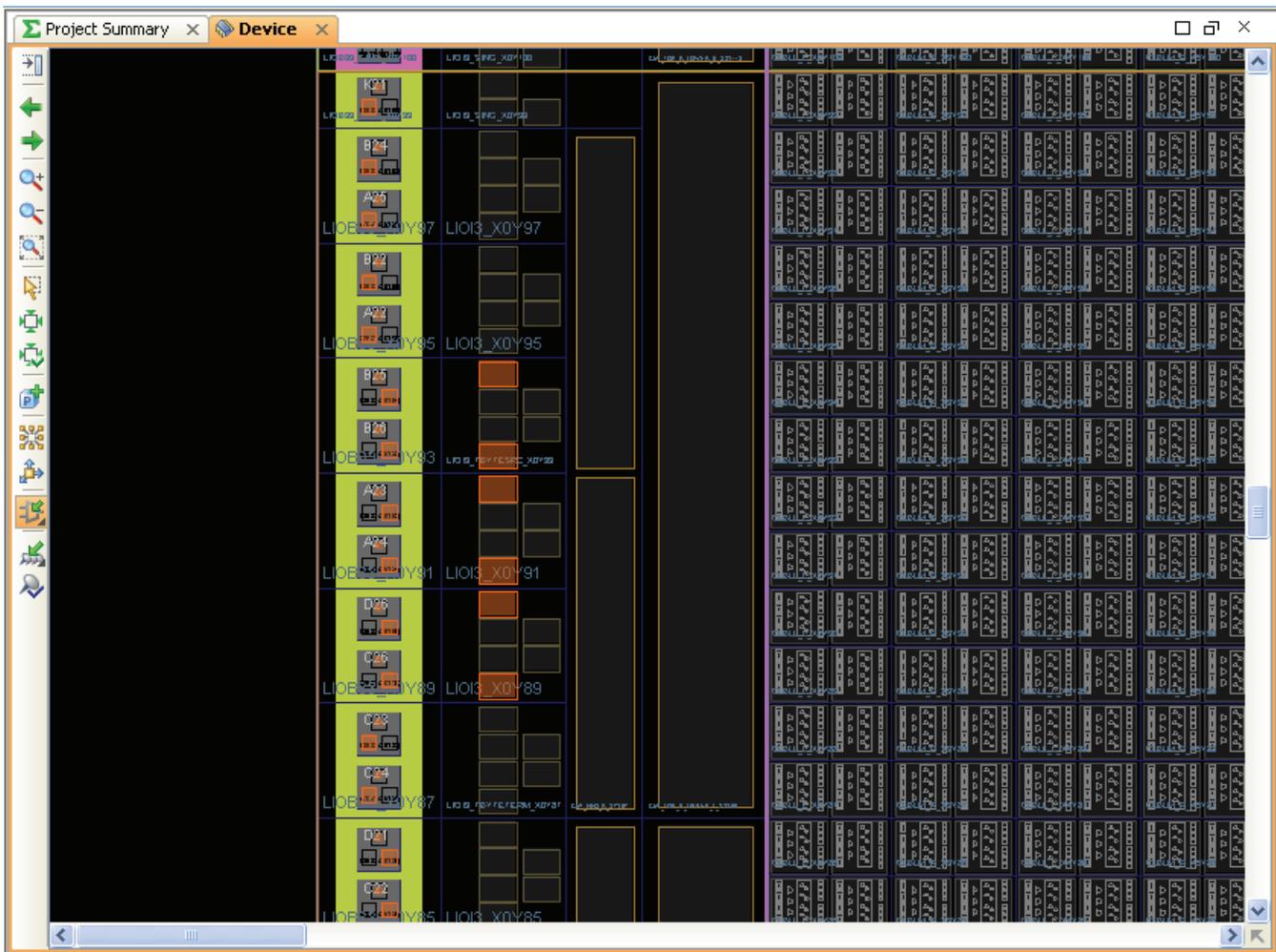
Figure 4-42: **Device View**

The amount of logic object detail displayed is determined by the selected zoom level: the more you increase the zoom level, the more logic object detail displays. The Device view popup and toolbar menus contain self-explanatory zoom level commands.

You can also hold down the left mouse button, and drag the cursor in the Device view to zoom into an area or to zoom out. See Using Mouse Strokes to Zoom, page 110 for more information.

The Device view has scroll bars and dynamic pan capabilities to pan the viewable area of the device also.

When you place the cursor over an object in the Device view, a tool tip identifies the object. The Properties view displays object properties for selected sites or logic objects.
Use **Edit > Find** to search for specific device resource sites.

For example, if you attempt a logic resource assignment that is illegal, the dynamic cursor changes so you can make adjustments. For more information, see Understanding the Context Sensitive Cursor, page 110.

Using Device View Commands

Toolbar buttons on the left side of the Device view include the following commands:

- **Device View Options:** Configure the display, color, and fill of specific layers in the Device view. See Setting Device View Display Options, page 139 for more information.

- **Previous Position:** Reset the Device view to display the prior zoom and coordinates.

- **Next Position:** Return the Device view to display the original zoom and coordinates after Previous Position has been used.

- **Zoom In:** Zoom into the Device view by a factor of two.

- **Zoom Out:** Zoom out the Device view by a factor of two.

- **Zoom Fit:** Zoom out to fit the whole device into the display area of the Device view.

- **Select Area:** Select the objects in the specified rectangular area.

- **Fit Selection:** Redraw the Device view to display the currently selected objects. This is useful when selecting objects in another view, such as the Netlist view, and redrawing the display around those selected objects.

- **Autofit Selection:** Automatically redraw the Device view around newly selected objects. This mode can be enabled or disabled.

- **Draw Pblock:** Create a new Pblock rectangle to place instances. For more information see Working with Pblocks, page 303.

- **Show I/O Nets:** To toggle the display of I/O connectivity to placed LOCs or Pblocks.

- **Show Instance Connections:** With this mode on, connectivity displays automatically for newly-selected objects. This button toggles the mode on and off.

- **Instance Drag & Drop Modes:** Determines the manner in which instances placed onto the device will be assigned placement constraints.

    - **Create BEL Constraint Mode:** Assign a LOC and BEL constraint to the instance being placed. This fixes the instance to the specified BEL within the Slice.

    - **Create Site Constraint Mode:** Assign a LOC placement constraint to the instance being placed. This fixes the instance to the specified Slice, but allows it to float to available BELs within the Slice.

    - **Assign Instance to Pblock Mode:** Assign logic instances to Pblocks. This is the default mode; use this mode whenever possible to ensure proper command behavior.

- **Place Ports Modes:** Determines how I/O ports are placed onto the device from the I/O Ports view or Netlist view.

    - **Place I/O Ports in an I/O Bank:** Assign the currently selected ports onto pins on the specified I/O Bank.

- • **Place I/O Ports in Area:** Assign the currently selected ports onto pins in the specified area.

- • **Place I/O Ports Sequentially:** Assign the currently selected ports individually onto pins.

- • **Autocheck I/O Placement:** Toggle Automatic enforcement of interactive I/O placement DRCs. When this mode is enabled, interactive I/O port placement will be checked against enabled design rules.

Understanding the Device Resource Display

The PlanAhead tool displays the various resources contained in a selected device in the Device view. Graphical sites display and are available for all of the device-specific FPGA resources. The level of detail for displaying the device resources depends on the zoom level within the Device view. Some resources, such as specific slice resources, are not visible until zoomed in quite close onto the FPGA logic. Some resources, such as Clock Regions and I/O Banks are displayed even when viewing the whole device. In addition, the Device View Options command lets you turn on or off the display of specific objects or resources in the Device view. For more information see Setting Device View Display Options, page 139.

The PlanAhead tool Device view resource placement process is:

- • The I/O pads and clock objects display around the periphery and/or down the center of the device.

  - • I/O banks display as thin color-shaded rectangles just outside the row of I/O pads.

  - • The Xilinx 7 series FPGAs offer both high-performance (HP) and high-range (HR) I/O banks which display in the Device view as right or left angled lines.

  - • Available I/O bank sites display as color-filled I/O bank rectangles.

  - • Some devices have unbonded I/O banks that display rectangles with a white X.

  - • The I/O clock pads display as filled-in rectangles.

- • All clock resources, such as `BUFGs`, `BUFGCTRLs`, `BUFRs`, and `BUFHCEs`, show in the Device view. When you select an I/O bank or clock region, the available device resources display in the Properties view.

- • The interior of the device is broken up into smaller rectangles called *Tiles*. Tiles are placement sites for the different types of logic primitives for the architecture.
  A tool tip identifies each site in the Device view when you hover the cursor over a logic site.

See Viewing and Reporting Resource Statistics, page 211 for more information about viewing device resources. You must zoom the view to see the Configurable Logic Blocks (CLBs) or Tiles, which consist of logic resources such as SLICEs and BELs.

You can:

- • Assign primitive logic instances to the appropriate displayed sites.

- • Import ISE placement results to display the logic assignments.

At the zoom level, the placed instances appear as rectangles within a `SLICE`. When you increase the zoom level, the logic symbols display. You can assign logic to specific sites that generate `LOC` placement constraints. Using `BEL` constraints you can assign instances to specific device resources in a `SLICE`.

Logic imported from ISE displays as `BEL`-level constraints. For more information about placement constraints, see Working with Placement LOC and BEL Constraints, page 324.

Setting Device View Display Options

For Xilinx 7 series FPGAs, Virtex®-6 family, and Spartan®-6 family, the Device view toolbar includes the **Device View Options** command with the following features:

- **Layers:** Define what device and design objects are displayed in the Device view. This command allows you to control the level of detail displayed in the Device view, and is especially useful when the display seems over-crowded with information. Figure 4-43 shows the Layers tab of the Device View Options command.
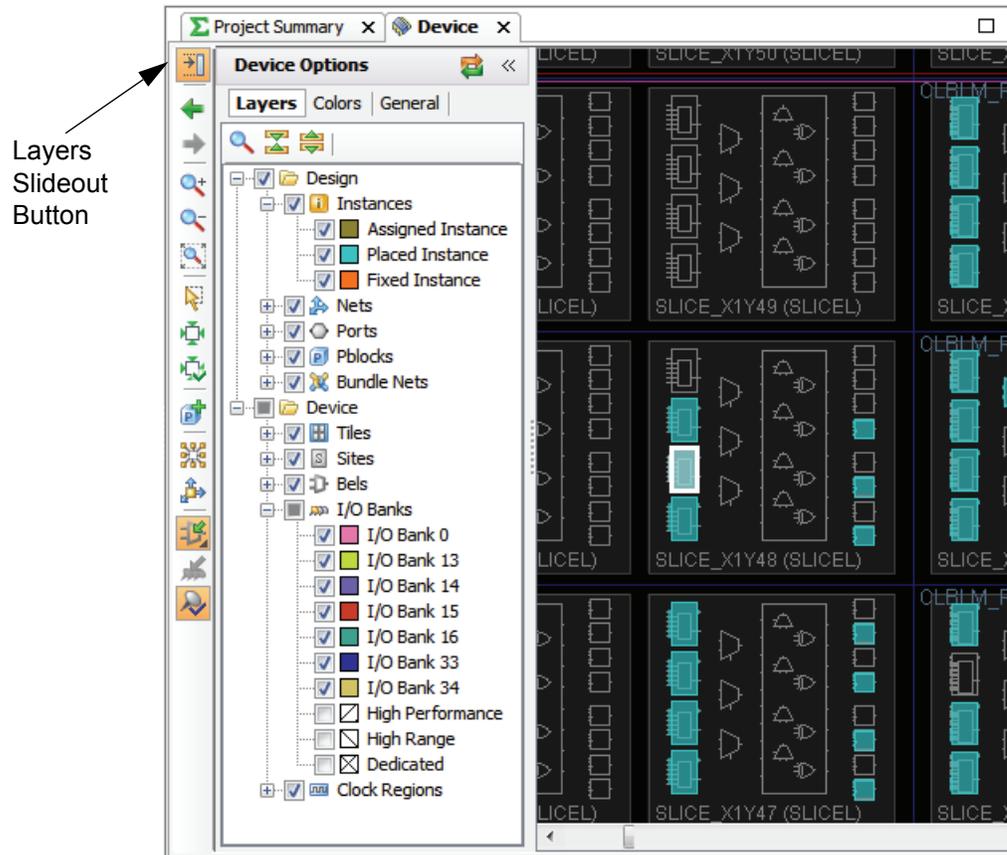
Figure 4-43: **Device View Layers**

The two primary branches are Design objects and Device objects:

- Design objects are elements from the design sources, such as instances, nets, and ports, that are placed on the device.
- Device objects are resources on the device such as I/O banks, clock regions, and tiles on which design objects can be placed.

Click the **+** sign to expand or the **–** sign to collapse the levels of the tree view so you can see the layer or object.

A check mark indicates a currently displayed layer. Click the checkbox to enable or disable the layer or object for display in the Device view. You can display or hide groups of objects or layers by clicking the category of the layers.

Note: If you cannot see a specific object or layer of the Device view, check the configuration of the Device View Options command to see if the design object or device resource is currently hidden.

- **Colors:** The color and fill values for elements of the Device view can be changed by:

  - Clicking on a color box to expose a pulldown menu and select from a list of available colors.

  - Choosing **More Colors** to display even more colors to choose from.

  - Entering a specific RGB value directly in the text field for the color.

Object Type	Frame Color	Fill Color
Background		0, 0, 0
Foreground		255, 255, 255
Selection		255, 255, 255
Markers		255, 255, 0
Pblock 1st Level	204, 102, 255	
Pblock 2nd Level	153, 51, 255	170, 151, 189
Pblock 3rd+ Levels	102, 0, 204	225, 196, 255
Assigned Instance	143, 131, 1	
I/O Net	0, 153, 0	
Placed Port	0, 255, 255	0, 255, 255

- **General:** Define the display characteristics of the Device view.

  - **Nets:** Draw nets as a mesh or tree, as shown in Figure 4-44, page 141.

    - **Mesh**: Pin to pin connections making a mesh of all connected pins.

    - **Tree**: A branching structure, grouping nearby pin connections.

  - **Connections**: Displayor hide connections when moving cells.

    - **Show connections while dragging instances**:Toggles the display of nets connected to the selected instance while dragging and placing the instance in the device view.

    - **Show/Hide connections for selected instances:** Toggles the display of nets connected to the selected instance.

Device Options

Layers | Colors | **General**

Nets
- ⦿ Draw nets as mesh
- ○ Draw nets as tree
- *Note: Floorplanning View Only*

Connections
- ☑ Show connections while dragging instances
- ☐ Show/Hide connections for selected instances

Routing
- ⦿ Leave existing routing
- ○ Unroute branches
- ○ Unroute and reroute branches
- *Note: There will be additional runtime moving placement when 'Unroute branches' or 'Unroute and reroute branches' is selected.*

High fanout net limit: 200 ⤢ nets

- **Routing:** Determines the action on existing routes when moving a cell in the Device view

  - **Leave existing routing:** Leave the existing routing in place, even though the pins may no longer be connected. The design will need to be rerouted at a later time.

  - **Unroute branches:** Remove the branches of the route that are no longer connected as a result of moving a cell. The design will need to be rerouted at a later time.

  - **Unroute and reroute branches:** Unroute and reroute the branches of the route that connect to the pins of a moved cell.

- **High Fanout Net Limit:** Limits the number of connections a pin can have in order to be displayed. The nets on a pin with a fanout greater than the specified number of connections will not be displayed.

- **Reset:** Reset the Device View Options for the Layers, Colors, or Connectivity Display, to the default settings provided by the PlanAhead tool.

Tree   Mesh

Figure 4-44: **Draw Nets As**

The Device View Options has a toolbar menu featuring a **Show Search** command to search for specific layers to display, and **Expand all**/ **Collapse all** commands.

Click the << icon to hide the Device View Options when you have finished with the command. The PlanAhead tool stores your settings for the Device View Options, to reload them each time the tool is launched.

Selecting Clock Regions

The clock regions display as large rectangles indicating the periphery of the various device clock regions. These outlines can help guide floorplanning for critical circuitry.

In the Device view, you can:

• Select the clock regions in the Clock Regions view.

• Select and specify that Clock regions display their resource statistical properties.

• View the clock placement statistics after importing the Implementation results.

When you select the clock region, the PlanAhead tool also selects the associated I/O banks and clock related logic sites.

See Clock Region Resources and Statistics, page 249 for more information about displaying Clock region statistics.

Opening Multiple Device Views

You can open multiple Device views for the same floorplan. This enables you to work on different areas of the device. See Splitting the Workspace, page 112 to open multiple Device views. Figure 4-45 shows an example of a split view.
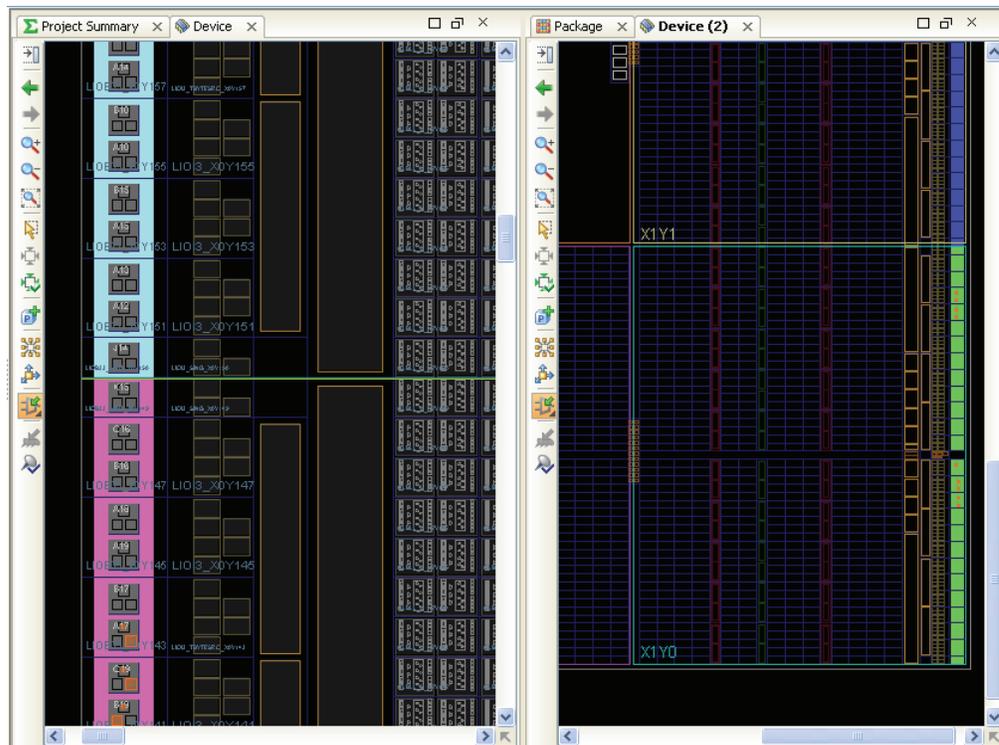
Figure 4-45: **Displaying Multiple Device Views**

Using the Package View

The Package view displays the physical characteristics of the target Xilinx part, and is used primarily during the I/O pin planning process, or during port placement. Pin types display in different colors and shapes for better visualization. The Package view opens automatically when the I/O Planning view layout is selected. For information about using the Package view for I/O pin planning, refer to Chapter 8, I/O Pin Planning.

You can manually open the Package view using the **Window > Package**. command from the main menu. The Package view is shown in Figure 4-46, page 143.

Using the Package view you can:

- Drag ports into the Package view for assignment and reassign placed instances to other I/O pins within the Package view.

- View pins and I/O banks as follows:

  - VCC and GND pins show as red VCC symbols and green GND symbols.

  - Clock-capable pins display as hexagon pins.

  - Colored regions display the different I/O banks on the device.

- Moving the cursor within the Package view shows the I/O pin coordinates actively on the top and left sides of the view.

- Select I/O pins or banks to cross-select between the Device and Package views, and see pin information in the Pins Properties view.

- Hold the cursor over a pin to display a tool tip that displays the pin information. Additional I/O pin and bank information displays in the Information bar located at the bottom of the environment.
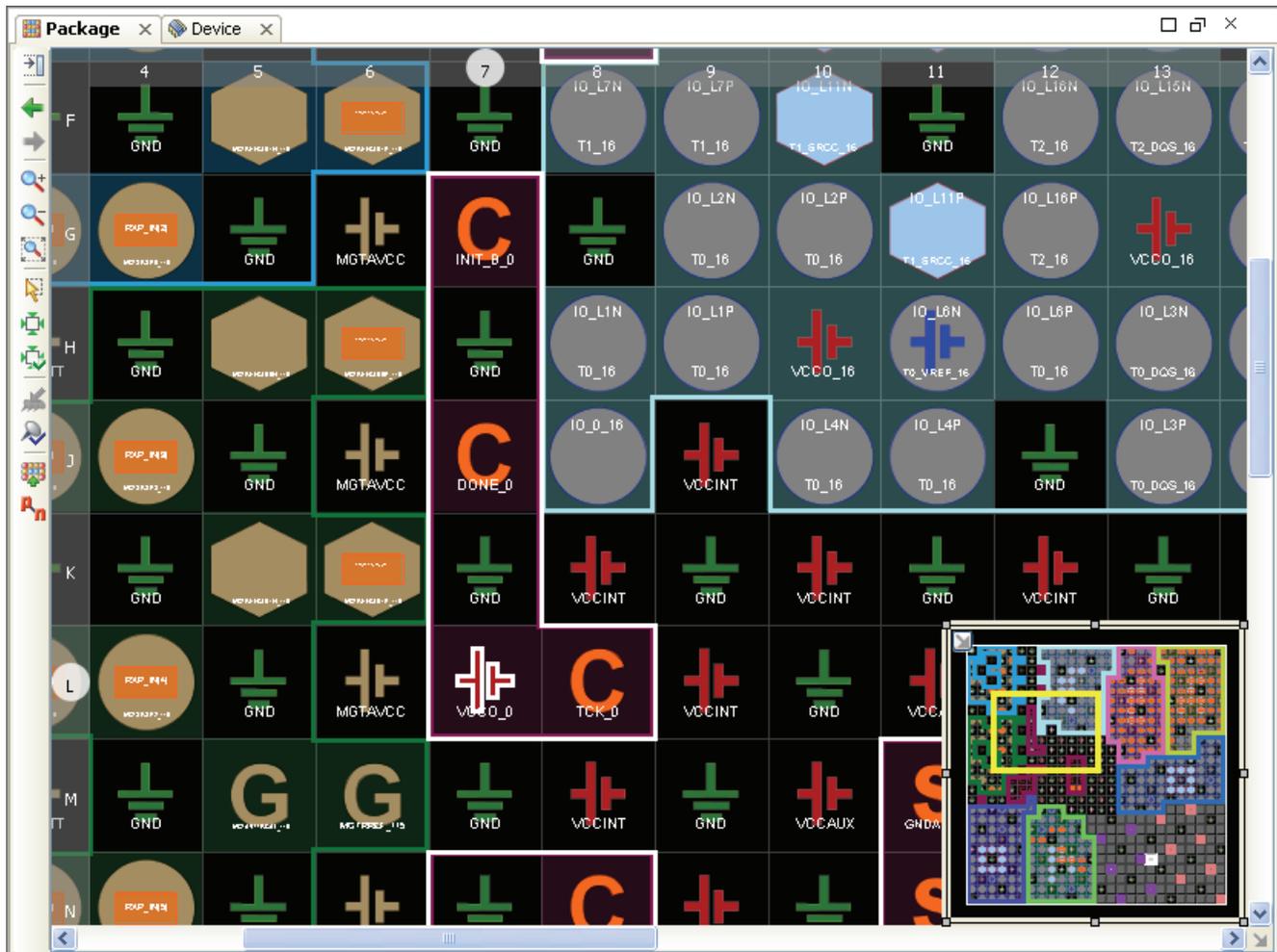
Figure 4-46: **Package View**

Using Package View Commands

Toolbar buttons on the left side of the Package view include the following commands:

- **Package View Options:** Configure the display of specific objects in the Package view. See Setting Package View Options, page 144 for more information.

- **Previous Position:** Reset the Package view to display the prior zoom and coordinates.

- **Next Position:** Return the Package view to display the original zoom and coordinates after Previous Position has been used.

- **Zoom In:** Zoom into the Package view by a factor of two.

- **Zoom Out:** Zoom out the Package view by a factor of two.

- **Zoom Fit:** Zoom out to fit the whole package into the display area.

- **Select Area:** Select the objects in the specified rectangular area.

- **Fit Selection:** Redraw the Package view to display the currently selected objects. This is useful when selecting objects in another view, such as the Netlist view, and redrawing the display around those selected objects.

- **Autofit Selection:** Automatically redraw the Package view around newly selected objects. This mode can be enabled or disabled.

- **Place Ports:** Determines how I/O ports are placed onto the package from the I/O Ports view or Netlist view.

  - **Place I/O Ports in an I/O Bank:** Assign the currently selected ports onto pins on the specified I/O Bank.

  - **Place I/O Ports in Area:** Assign the currently selected ports onto pins in the specified area.

  - **Place I/O Ports Sequentially:** Assign the currently selected ports individually onto pins.

- **Autocheck I/O Placement:** Toggle Automatic enforcement of interactive I/O placement DRCs. When this mode is enabled, interactive I/O port placement will be checked against enabled design rules.

- **Show Bottom/Top View:** Toggle the Package view to display the package pins as viewed from the top, or as viewed from the bottom.

- **Show Differential I/O Pairs:** Display the differential pair pins in the Package view.
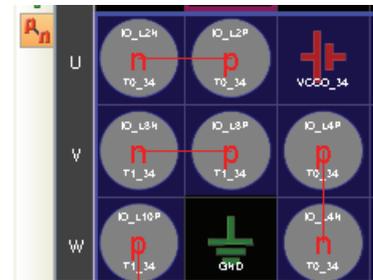
Setting Package View Options

The Package view toolbar includes a Package View Options command that lets you define what layers and objects are visible in the Package view. Figure 4-47, page 145 shows the Package View Options command with the Package view layers you can display or hide.
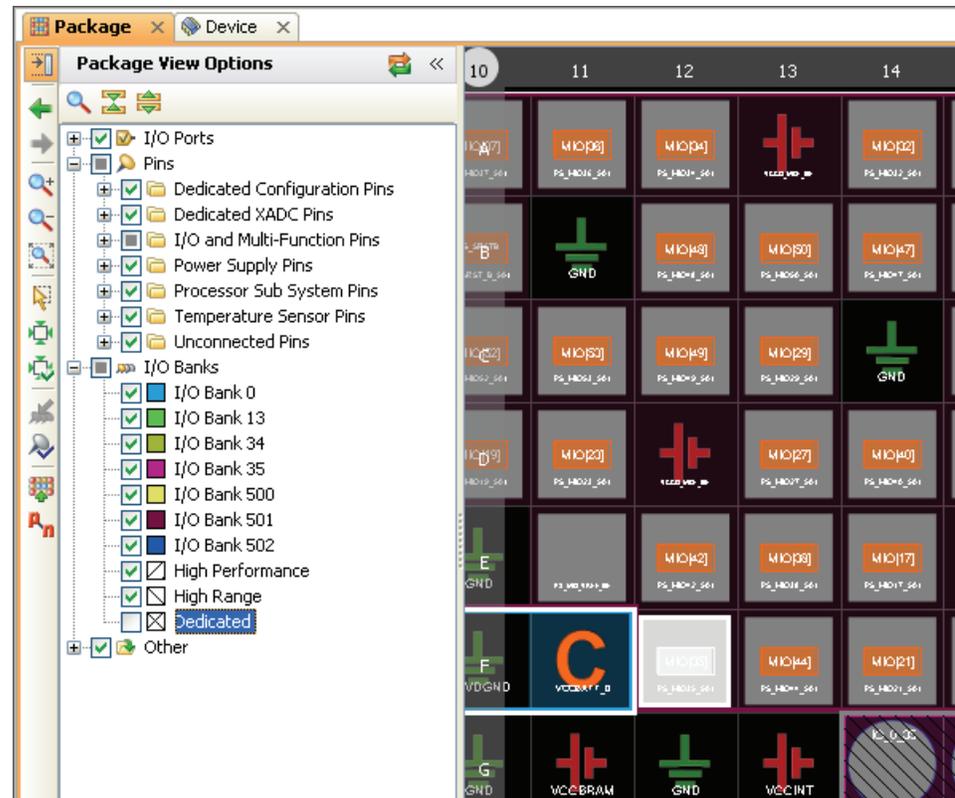
Figure 4-47: **Package View Layers**

The available layers are listed hierarchically in a tree view which can be expanded or collapsed. In the Package view the three primary categories of layers are I/O Ports, Pins, and I/O Banks:

- I/O Ports shows the ports in the design that are currently placed in either a fixed or unfixed state. The design might have currently unplaced ports that are not displayed in the Package view.

- Pins includes the available package pins grouped into specific categories; such as multifunction pins, power pins, and unconnected pins.

- Power pins display separately from the I/O banks.

- Multifunction pins display as part of the I/O bank they are contained in, and display with symbols representing their available functions. For example:

  - Clock capable pins display as blue hexagrams

  - Vref pins display with a small power icon

  The Package View Options form provides a legend of the icons used for multifunction pins.

- Processor Sub System Pins (PSSIO) are found on Zynq™ devices, and are read-only pins in the PlanAhead tool, as can be seen in the Package Pin Properties view. These pins are not configurable as programmable logic (PL) pins on the Zynq device, but are configured as processor system (PS) pins, and imported from XPS. Refer XPS Help for more information.

- I/O Banks show the sites of the pins for each of the banks on the device as well as the sites of the GTX pins. Each I/O bank and GT bank is color-coded to allow you to easily differentiate the different banks of pins.

- The Xilinx 7 series FPGAs typically offer both high-performance (HP) and high-range (HR) I/O banks. HP and HR I/O banks are displayed as a feature of the I/O banks.

The display of a specific pin in the Package view can depend on a combination of layers that represent the pin in the Package View Options form. If the I/O banks are not displayed, then power pins display but the multifunction pins do not display even though both are selected under the Pins heading of the Layers Slideout.

Click the **+** sign to expand or the **-** sign to collapse the levels of the tree view to see the layer of interest. Click in the box to enable or disable the layer for display in the Package view. A check mark indicates the currently displayed layer.

You can display or hide:

- Groups of objects by clicking the category of the objects.
- Individual objects by selecting the item directly.

Note: If a specific pin is not visible in the Package view you cannot assign a port to it. Check that both the pin and the I/O block it is contained in are selected for display in the Package View Options form.

Opening Multiple Package Views

You can open multiple Package views for the same design. This enables you to display and work on different areas of the package. See Splitting the Workspace, page 112 to open multiple Package views. Figure 4-45, page 142 shows an example of a split view.

Using the Schematic View

You can generate a Schematic view for any level of the logical or physical hierarchy. You can select a logic element in an open view, such as a primitive or net in the Netlist view, and use the **Schematic** command in the popup menu to crate a schematic view for the selected object. The Elaborated Design always opens with a schematic view of the top-level of the design, as shown in Figure 4-48, page 147.

In the Schematic view you can view design interconnect, hierarchy structure or trace signal paths for either the Elaborated Design, Synthesized Design, or Implemented Design.

- See Analyzing Synthesis Results in Chapter 6 for more information about analyzing RTL netlist.
- For more information about synthesized netlist analysis see Chapter 7, Synthesized Design Constraints and Analysis.

To create a Schematic view:

1. Select one or more logic elements in an open view, such as the Netlist view.

2. Right-click and select **Schematic** from the popup menu, or select the **Schematic** toolbar button.

The Schematic view displays the selected logic instances or nets. If only one instance is selected, a schematic symbol for that module is displayed as shown in Figure 4-49, page 148.

The links at the top of the schematic sheet, labeled *Instances*, *I/O Ports*, and *Nets*, open a searchable list in the Find Results view, making it easier to find specific items in the schematic.

When you select objects in the Schematic view those objects are also selected in all other views. If you have opened an implemented design, the instances and nets display in the Device view.
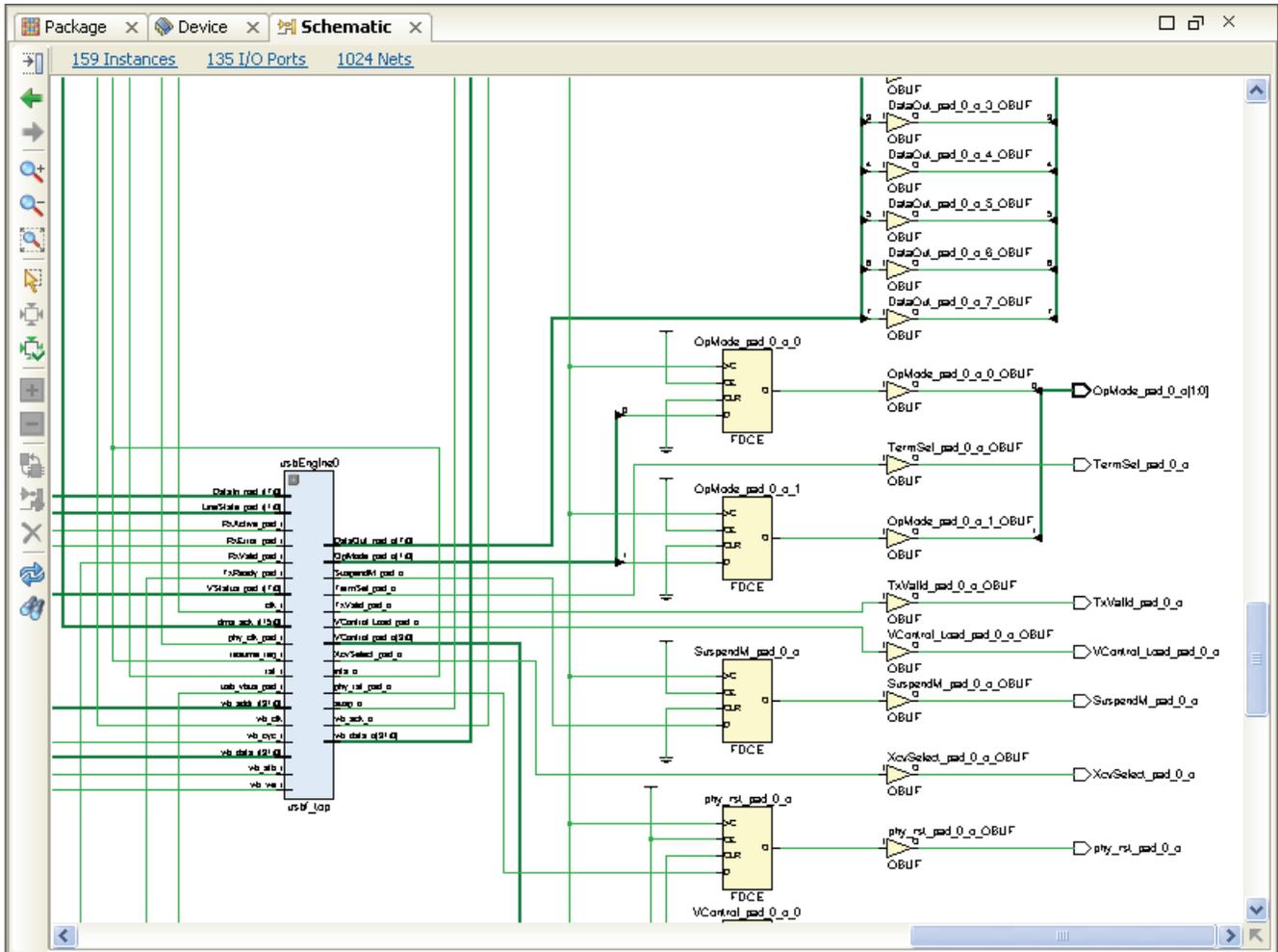
Figure 4-48: **Schematic View**

Using Schematic View Commands

Toolbar buttons on the left side of the Schematic view include the following commands:

- **Schematic View Options:** Configure the display of specific objects in the Schematic view. See Selecting Objects in the Schematic View, page 150 for more information.

- **Previous Position:** Reset the Schematic view to display the prior zoom and coordinates.

- **Next Position:** Return the Schematic view to display the original zoom and coordinates after Previous Position has been used.

- **Zoom In:** Zoom into the Schematic view.

- **Zoom Out:** Zoom out the Schematic view.

- **Zoom Fit:** Zoom out to fit the whole schematic into the display area.

- **Select Area:** Select the objects in the specified rectangular area.

- **Fit Selection:** Redraw the Schematic view to display the currently selected objects. This is useful when selecting objects in another view, such as the Netlist view, and redrawing the display around the currently selected objects.

- **Autofit Selection:** Automatically redraw the Schematic view around newly selected objects. This mode can be toggled on or off.

- **Expand all logic inside selected instance:** Expands a hierarchical module from the symbol view to the logic view. Hierarchical modules can also be expanded directly from the schematic by clicking on the '+' icon on the schematic symbol, as shown in as shown in Figure 4-49.

- **Collapse all logic inside selected instance:** Collapses a hierarchical module from the logic view to the symbol view. An expanded hierarchical block can also be collapsed directly from the schematic by clicking on the '-' icon on the hierarchical block, as shown in Figure 4-49.
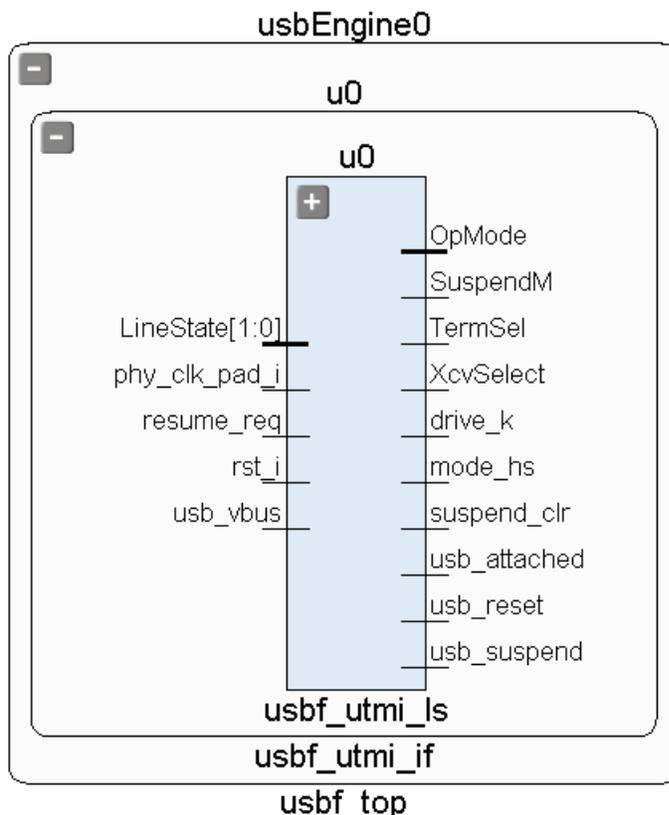
Figure 4-49: **Viewing Hierarchy in the Schematic**

- **Toggle autohide pins for selected instance**: Toggle the pin display on hierarchical modules. Higher levels of the hierarchy display as concentric rectangles without pins, when a Schematic view is generated, as shown in Figure 4-49. In most cases, the lack of pins makes the Schematic view more readable, however, you can display the pins for selected instances as needed.

- **Add selected elements to schematic:** Recreate the Schematic view with the newly selected elements added to the existing schematic.

- **Remove selected elements from the schematic:** Recreate the Schematic view with the currently selected elements removed from the existing schematic.

- **Regenerate Schematic:** Redraw the active Schematic view.

- **Find in Schematic:** Open the Find dialog box. See Using the Find Commands, page 122 for more information.

Expanding Logic from Selected Instances and Pins

With a selected schematic instance, or a selected pin on a schematic instance, you can:

- Individually expand or collapse module pins and logic.

- Selectively expand the logic either from individual pins, instances, or the entire logic content inside or outside the module.

You can expand or collapse logic contained either inside a selected module or outside in the next level of hierarchy. You can expand a single module or multiple selected modules. The commands to expand schematic logic are:

- Use the **Expand/Collapse > Expand Inside** from the popup menu to display the schematic hierarchy inside of a selected instance. The PlanAhead tool regenerates the Schematic view to expand the contents of the selected instance.

  - This command is not available if the selected instance is a primitive within the design hierarchy.

  - Use **Expand/Collapse > Collapse Inside** to hide the expanded contents of a selected hierarchical block.

- Use the **Expand/Collapse > Expand Outside** from the popup menu to display the hierarchy upward from a selected instance. The PlanAhead tool regenerates the Schematic view to expand the hierarchy up from the selected instance.

  - This command has no effect if the selected instance is at the top-level of the design hierarchy.

  - Use **Expand/Collapse > Collapse Outside** to hide the expanded hierarchy outside the selected instance.

- Double-click a pin of an instance to trace the net down into, or up out of the hierarchy. A pin is displayed on the schematic symbol with a stub inside and outside of the symbol as shown in Figure 4-50. This reflects the ability to expand inside or outside the symbol.

  - Double-click a pin inside a schematic symbol to trace the net downward, into the hierarchy.

  - Double-click the pin outside a schematic symbol to trace the net up the hierarchy.

Note: Net expansion has a different result than expanding the hierarchical module using the Expand Inside/Expand Outside commands. Double-clicking a pin expands the hierarchy to follow the net, and does not display the full contents of the hierarchy.
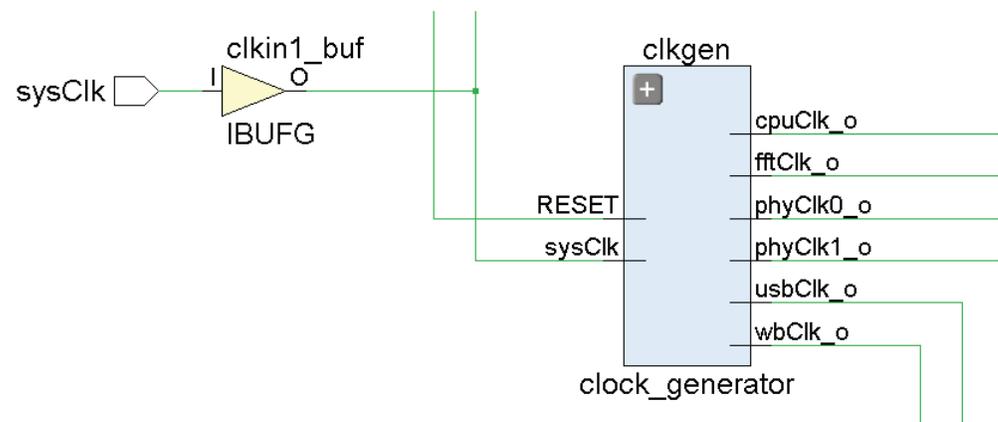
Figure 4-50: **Double-click Schematic Pins**

- Expand buses to include all bits of the bus. Buses show as thick wires.

- Use the **Expand Cone** command from the popup menu to expand the cone of logic from a selected pin or instance, or between two selected instances. Expansion of logic can go beyond hierarchical boundaries. The available **Expand Cone** options, as shown in Figure 4-51, are:

  - **To Flops or I/Os:** Appends the view to display the entire cone of logic to the first flops or I/Os, or to any sequential element, such as block RAMs, FIFOs, and embedded processors.

  - **To Primitives**: Appends the view to display the entire cone of output logic to the first primitives. This is also the default behavior when you double-click a pin.

  - **Between selected instances**: Appends the view to display the entire cone of logic between two selected instances.
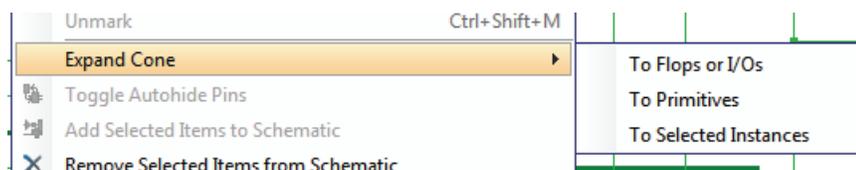
Figure 4-51: **Expand Logic Cone**

Selecting Objects in the Schematic View

The following are object selection options for the Schematic view:

- Left-click an object in the Schematic view

- Use the **Ctrl** key to select multiple objects

- Click **Select Area**, and draw a rectangle around multiple instances, ports and nets

When you select instances in the Schematic view, they are also selected in all other views. The cross-selection works also when you select or highlight objects in other views so that they are highlighted in the Schematic view.

When you select objects in the Schematic view, the Properties view for the selected object opens as well.

The Connectivity tab traverses the hierarchy to report all primitive instances connected to the net. This is different from the Pins tab which reports the pins of all instances connected to the net, reporting both primitive and hierarchical instances. Select a net that is connected to a hierarchical instance to see the difference between these tabs.

Setting Schematic View Options

The Schematic View Options command, as shown in Figure 4-52, page 151, allows you to define which attributes to display on schematic symbols and pins, and also allows you to configure the colors to use when creating the Schematic view.
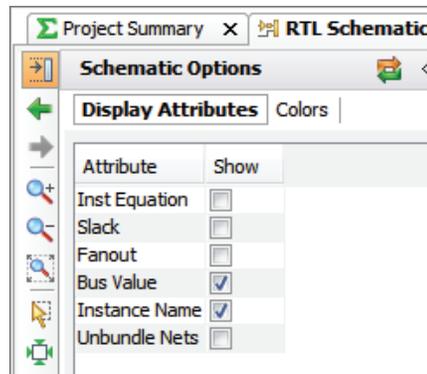
Figure 4-52: **Schematic View Options**

- **Display Attributes**: Allows you to enable or disable the following features to be displayed on the Schematic view:

  - **Inst Equation**: Label instances with truth table equations.

  - **Slack**: Label destination pins with *Slack* values. Slack values do not display until after timing analysis.

  - **Fanout**: Label source pins with *Fanout* values.

  - **Bus Value**: Label bus pins with bus values.

  - **Instance Name**: Label instances with Instance names.

  - **Unbundle Nets**: Display bus nets as bundled, or unbundled nets.

- **Colors:** The color of elements of the Schematic view can be changed by:

  - Clicking on a color box to expose a pulldown menu and select from a list of available colors.

  - Choosing **More Colors** to display even more colors to choose from.

  - Entering a specific RGB value directly in the text field for the color.

- **Reset:** Reset the Device View Options for the Layers, Colors, or Connectivity Display, to the default settings provided by the PlanAhead tool.

Viewing Timing Path Logic in the Schematic View

You can select Timing paths from the Timing Results view and use the Schematic command in the popup menu to display the full timing path in the Schematic view. All of the objects on the selected path or group of paths display with the logic hierarchy boundaries and the interconnect wires, shown in Figure 4-53.
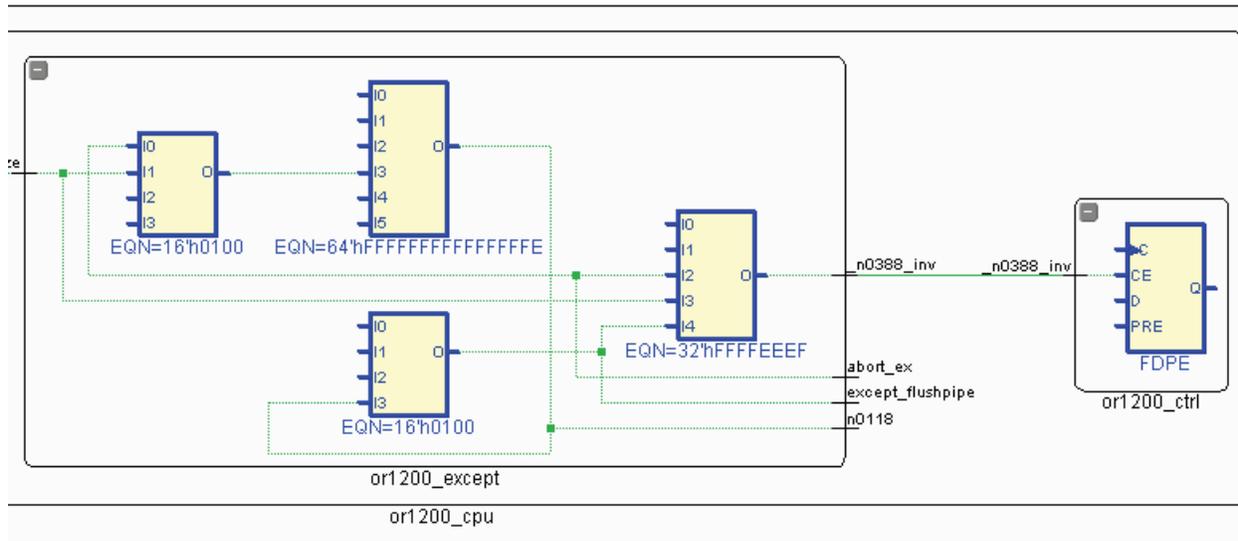
Figure 4-53: **Timing Path in Schematic View**

For more information about setting the timing path logic, see: Chapter 7, Synthesized Design Constraints and Analysis, and Chapter 11, Analyzing Implementation Results.

Note: Occasionally, paths displayed from the Timing Reporter and Circuit Evaluation (TRCE) TWX (an XML file) or TWR (a Text file) format timing reports are missing interconnect wires. This is because the logic was optimized out of the path during ISE Implementation. The objects that display in the Schematic view are all of the actual objects contained in the selected paths; however, the PlanAhead tool cannot interpolate the connectivity after objects have been optimized away and no longer exist. You can use the Schematic view in conjunction with the Path Properties to trace the path connectivity. Usually the schematic is drawn is such a way that it is easy to see the path direction. For more information, see Analyzing Timing Results, page 230.

Using the Properties View

The Properties view displays information about any selected logic object or device resource. The Properties view is dynamic by default; as you select an object, its properties are automatically displayed in the Properties view.

The displayed name of the Properties view changes to reflect the selected object. For example, the view will be called BEL Properties view when a BEL is selected, or Clock Region Properties view when a clock region is selected.

Note: When selecting multiple objects, the Properties view will display the properties of the last object selected.

To open the Properties view:

- Use the **Windows > Properties** command from the main menu,
- Click the **Properties** toolbar icon,
- Select an object and use the *Object* **Properties** command in the popup menu.

The Properties view displays the various properties associated with a selected object, using tabs to organize information under different categories. The specific tabs that are available and the information they display is dependant on the type of object currently selected. Figure 4-54, page 153, shows the various tabs of the Instance Properties view, currently displaying the Attributes tab of the selected instance.
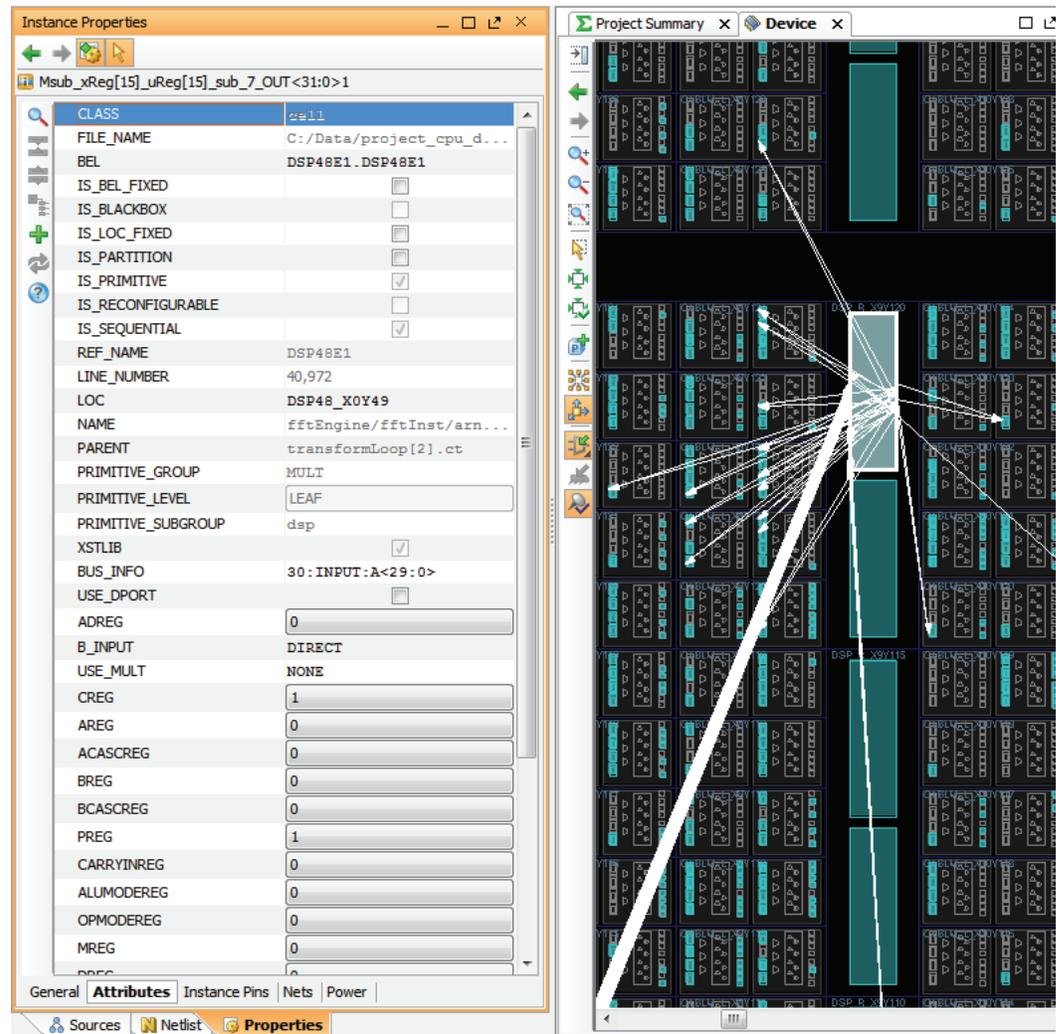
Figure 4-54: **Properties View**

Properties View Commands

The Properties view toolbar contains a variety of commands depending on the selected object and the specific tabs displayed. Some of the common commands are:

Previous object: Displays the properties of the previously selected object as opposed to the currently selected object. You can use this command iteratively to scroll backward through the selected objects.

Next object: Scrolls forward through the selected objects to display the object properties. This command is only available after using the **Previous object** command.

Automatically update the view when new objects are selected: By default, the Properties view is updated to display the properties of the latest object as new objects are selected. This command toggles the Properties view to auto-update as new objects are selected or to remain static displaying the properties of the currently selected object.

Select/Unselect object: Select or de-select the object reported in the Properties view.

Expand all: Expands all hierarchical tree objects to see all elements of the Sources view.

Collapse all: Collapses the hierarchical tree objects to display only the top-level objects.

Group by type: Groups the selected items by type.

Delete: Deletes an attribute or object from within one of the tabs of the Properties view. This command is only available for certain object types and in specific view panes.

Export statistics to file: Saves data to the specified file for later use. Available from the Statistics tab of the Pblock, Clock Region and Instance Properties view only.

Show unsaved attributes only: Displays only the unsaved attributes on the Attributes tab of the Properties view. These are the attributes that have been updated and will be saved when the Apply command is used.

Add pre-defined attributes: Adds a new attribute to the selected object. This command is only available for certain object types and in specific view panes.

Using the Run Properties View

The Run Properties view, which is one form of the Properties view, displays information about a selected synthesis or implementation run. Accordingly, it is labeled the Synthesis Run Properties view or the Implementation Run Properties view. Figure 4-55 shows the Implementation Run Properties view for a selected run.
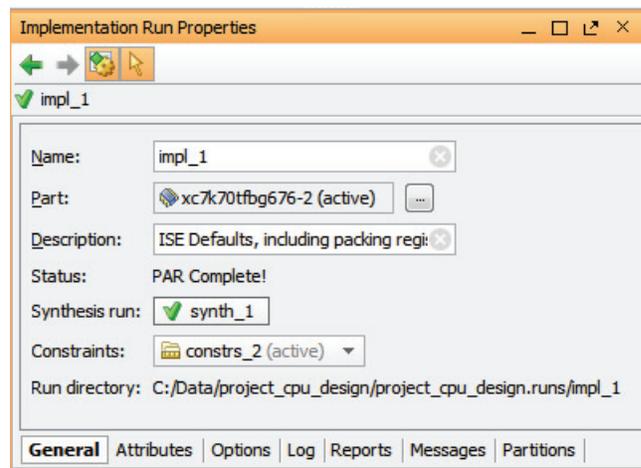
Figure 4-55: **Implementation Run Properties View**

As you select runs in the Design Runs view, the properties of the run appear in the Run Properties view under one of the tabs. The General tab reports the configuration of the run, including the target part, the constraint set, the parent run, the run strategy, and the run directory where output files for the run are or will be written. Table 4-1, page 155 provides details of each of the tabs of the Run Properties view.

The properties reported in the Run Properties view can be changed directly in the view. However, after a run has been synthesized or implemented, the run will become out-of-date if changes are made to the run properties. In this case you may need to reset the run, and re-launch the run to update the results. To reset a run use the **Reset Runs** command from the Design Runs view popup menu. See Launching Runs on Remote Linux Hosts in Chapter 9 for more information.

Table 4-1: **Tabs of the Run Properties View**

Tab	Options
General	• **Name**: Defines the run name. • **Part**: Displays the target part for the current run, and allows you to change the project part for the run. The target part is defined under Project Settings, but can be changed in the Run Properties form. See Configuring Project Settings in Chapter 3 for information on setting the target part for the entire project. • **Description**: Provides a brief description of the current run strategy. • **Status**: Displays the status of the run. • **Synthesis Run**: Displays the parent synthesis run of a selected implementation run. ***Note:*** This is a property of the implementation run, and does not appear on synthesis runs. • **Constraints**: Accept or change the constraint set for the run. • **Run Directory**: Displays the location of the run data.
Attributes	Lists the various properties of the selected Synthesis or Implementation run.

Options	Displays the command-line options and their currently set values. Select a command option to see a description of the command.
	• If you have not yet launched a run, you can modify the values of the command options. Select a command option to edit, and click the checkbox to enable or disable the option, enter a value, or select the value from the pulldown menu, and click **Apply**.
	• Modified values show an asterisk (*) next to the option to indicate that default Strategy values have changed.
	• With the right mouse popup menu:
	  • **Save Strategy As** to save the new option settings as a strategy for later use in other runs.
	  • Refresh or restore the command options to their preset values.
	• After you launch the run (which is described in Launching Implementation Runs in Chapter 9), if you modify the run strategy, the run will become out-of-date. You should cancel and reset the run instead. See Canceling the Run in Chapter 9.

Log	Displays the same `STDOUT` command status logs that display in the Log view. Figure 4-10, page 105 shows an example of the Compilation log. The Log continues to update as commands run. You can use the scroll bar to browse through the command log reports. Click **Automatically update the contents of this view** to stop the active reporting. This lets you scroll easier and read results while the command is running.
Reports	View report files generated by the ISE tool from within the PlanAhead tool. In the Implementation Run Properties view, select the run, and then select the **Reports** tab to display the list of available report files, and to open a report in the workspace.

Table 4-1: **Tabs of the Run Properties View**

Tab	Options
Messages	View run messages. An example of the Messages view is shown in Figure 4-7, page 103.
Partitions	Lists any partitions defined in the design. See Chapter 13, Using Hierarchical Design Techniques for more information on using partitions.

Using the Netlist View

The Netlist view provides a hierarchical view of the elaborated or synthesized logic design including the nets, logic primitives, and hierarchical modules of the design, starting with the currently defined top module.

Figure 4-56 shows the Netlist view.
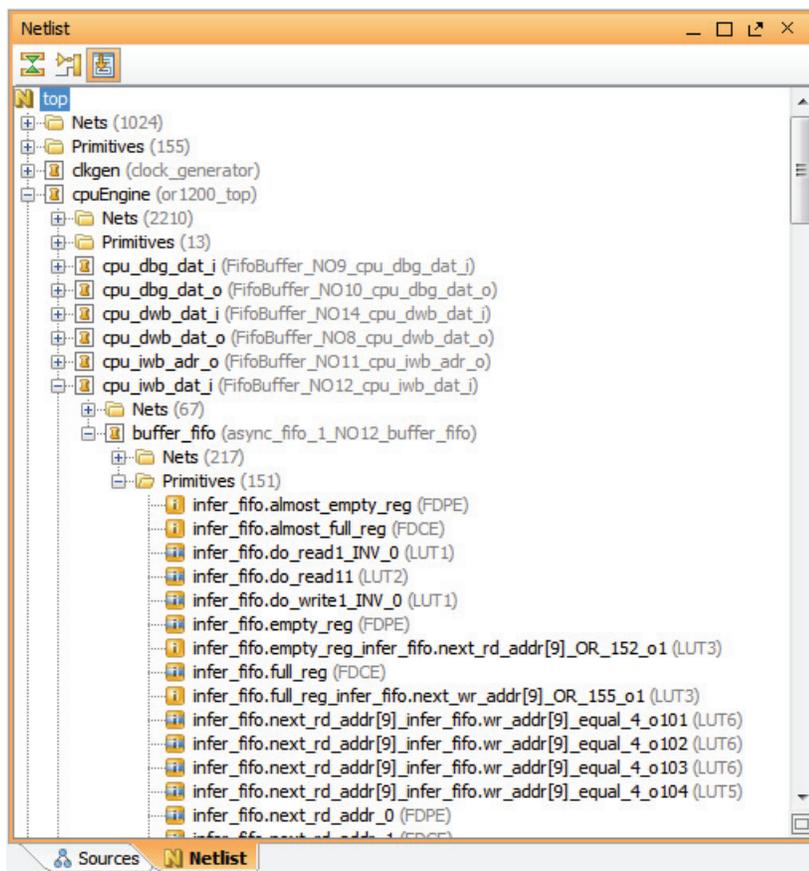
Figure 4-56: **Netlist View**

The Netlist view displays the logic instances and nets contained in the design. The netlist can be navigated by expanding and collapsing the logic tree. The default netlist tree setting is to expand and scroll the netlist object dynamically when they are selected in other views. To disable this feature, select the **Automatically scroll to selected objects** toolbar button in the Netlist view.

The entire netlist tree can be collapsed by selecting the **Collapse All** toolbar button in the Netlist view. For more information, see Using View-Specific Toolbar Commands, page 115. The Netlist tree collapses to display only the top-level logic modules.

You can select instances and apply commands using the main menu, the toolbar, or the popup menu.

Use the **Shift** key or the **Ctrl** key to select multiple elements in the Netlist view for use with most commands. Selected logic is highlighted in the Netlist view.

Logic selected in a different view, such as the Schematic or Device views, is cross-selected in the Netlist view. The Netlist tree expands automatically to display all selected logic. You might need to scroll the tree to view all selected logic. Collapsing the Netlist tree does not unselect logic.

Primitive logic is placed in a /Primitives folder for each level of the hierarchy. This condenses the display of logic content and hierarchical modules in the Netlist view, as shown in Figure 4-57, page 157.
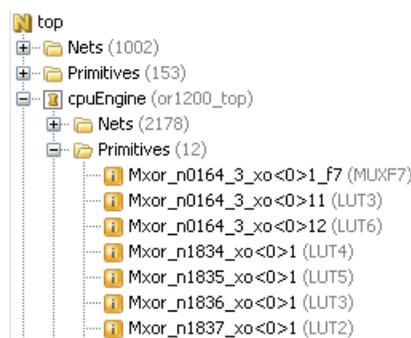
Figure 4-57: **Primitives Folder in the Netlist View**

You can assign all the primitive logic of a module to a Pblock by assigning the /Primitives folder directly to the Pblock.

Note: Netlist updates might require reassignment of the Primitives folder to the Pblock because logic names might change during re-Synthesis.

Nets, or wires, are placed in a /Nets folder for each level of the hierarchy. All of the bits of a bus are collapsed under the bus by default, but you can expand buses to show each individual bit, as shown in Figure 4-58.
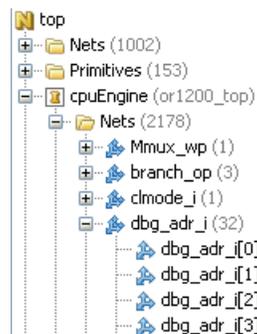
Figure 4-58: **Nets Folder in Netlist View**

When you select nets, they highlight in the Device view. Selecting a bus highlights all nets contained within that bus. You can also view nets in the Schematic view.

You can select nets for ChipScope tool debug testing by using **Add to ChipScope Unassigned Nets**. See Connecting and Disconnecting Nets to Debug Cores in Chapter 12.

Understanding the Netlist View Icons

Various icons are used to represent the state of netlist logic as shown in Figure 4-59 and described in the following subsections.

- Hierarchical instance (logic)
- Hierarchical instance (black box)
- Hierarchical instance (black box + Pblock)
- Hierarchical instance (Pblock)
- Partitioned instance (logic)
- Partitioned instance (black box)
- Partitioned instance (Pblock)
- Partitioned instance (black box + Pblock)
- Bus
- Net
- Primitive instance
- Primitive instance (fixed with LOC)

Figure 4-59: **Netlist View Icons**

Hierarchical Instances

Hierarchical netlist modules or "instances" are displayed with a yellow **I** icons, as shown in the following figure.

LED (decode7SegDispCK_49_123)
receiver (RCV_802_11)
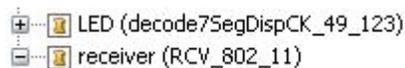
Hierarchical Instances Assigned to Pblocks

Hierarchical netlist modules or "instances" are assigned to Pblocks are displayed with blue check mark icons, as shown in the following figure.

busMuxWrapInst (busMuxWrap)
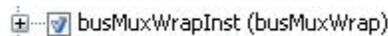
Black Box Modules

Hierarchical instances that do not have netlists or logic content, are viewed by the PlanAhead tool as black boxes. These are identified with a hierarchical instance icon with a dark background as shown in the following figure. A hierarchical instance may be a black box by design, or may be the result of bad a search path or missing files.

CONVDECOD (viterbi)
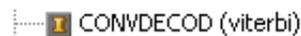
Partition Modules

Modules that have been set as Partitions using the **Set Partition** popup menu command display as yellow boxes within a white box. See Chapter 13, Using Hierarchical Design Techniques for more information on using design partitions.

wbArbEngine (wb_conmax_top)
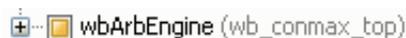
Partial Reconfiguration Partition Modules

Modules that have been set as Reconfigurable Partitions using the **Set Partition** menu command in a Partial Reconfiguration project are displayed as a yellow diamond in a white box.

Primitive Logic Instances

Primitive logic instances display as follows:

- *Without* placement constraints assigned display as an "**i**" inside a yellow rectangle.
- *With* placement constraints assigned display a yellow rectangle with a blue stripe.
- Assigned to a Pblock display a blue check mark inside a yellow rectangle.
- Placed and assigned to a Pblock display a check mark and blue stripe in a yellow rectangle.

Notice the logic types display also, as shown in the following figure.
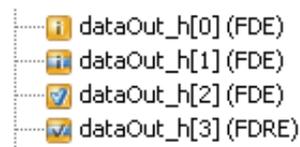
Using the Hierarchy View

The Hierarchy view lets you visualize the logic hierarchy reflected in the Netlist view, based on the current top module. You can see the relationship between hierarchical modules as well as their relative sizes.

In the popup menu of an open view, such as the Netlist view, select the **Show Hierarchy** command to invoke the Hierarchy view, shown in Figure 4-60, page 160.

The Hierarchy view displays a graphical representation of the logic hierarchy for the current design. Viewing the design from top to bottom, you can identify module sizes and location within the design.
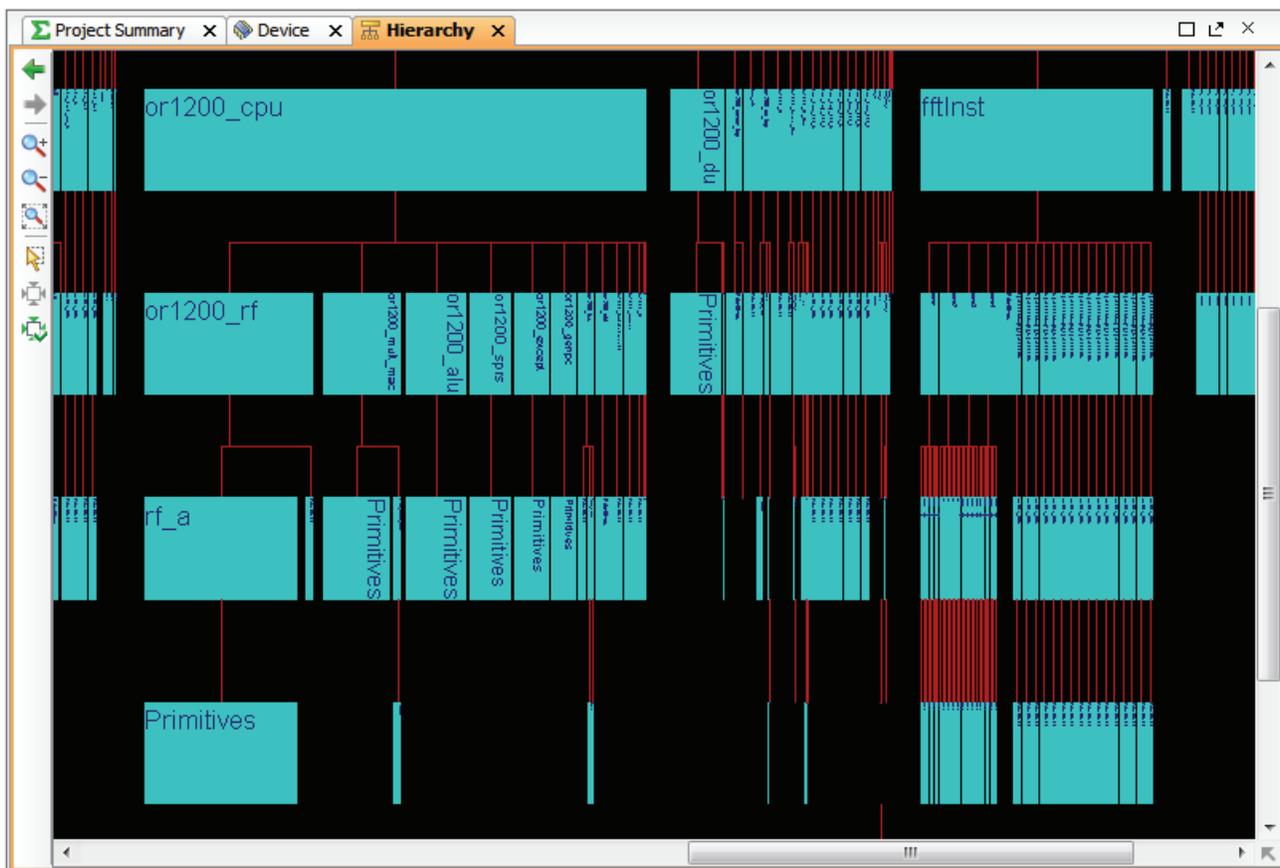
Figure 4-60: **Hierarchy View**

The Hierarchy view is used primarily during design analysis and floorplanning. It lets you see how a timing path traverses the logic hierarchy, or gauge how big a module is before you floorplan the module. The widths of the blocks in the Hierarchy view are based on the relative FPGA resources consumed by the block.

Each hierarchical instance displays in the Hierarchy view. Primitive logic is grouped into folders that are represented as sub-modules. Refer to Using the Netlist View, page 156 for more information about primitive logic folders.

To select logic parent modules for Pblock assignment in this view, use the **Select Primitive Parents** command.

Using the I/O Ports View

The I/O Ports view is used to create, configure, and place I/O ports onto I/O sites in either the Package view or Device view. The I/O Ports view shows the I/O signal ports defined in the design. To open the I/O Ports view, select **Window > I/O Ports**. Figure 4-61 shows the I/O Ports view.

In an I/O Planning project you can import a port list from a CSV file or a UCF, and create ports manually for the project. See Chapter 8, I/O Pin Planning for more information.

Creating RTL source or Netlist projects populates the I/O Ports view with the I/O ports defined in the design source files.

Note: Many of the commands for defining and configuring ports in the I/O Ports view are not available in an RTL or Netlist project, as the ports are defined by the design sources.

Figure 4-61: **I/O Ports View**

The I/O Ports view:

- Lists the port signal names, direction, package pin, I/O bank, I/O Standard, Drive strength, Diff pair partner, Slew type, voltage requirements and other signal information for each I/O port.

- Sorts the I/O Ports based on column values. See Using Tree Table Style Views, page 113 for more information about working with table views.

- Shows table values as follows:

  - Blank if they are default values

  - An asterisk (*) for non-default values

  - Red when they are illegal or undefined values

- Shows cells with editable values in the I/O Ports view. Enter text or select text from drop-down menus to change current values.

- Buses are in expandable folders that can be selected as one object for analysis, configuration and assignment.

Using I/O Ports View Commands

In the I/O Ports view, you can:

- **Create I/O Port:** Manually define I/O Ports in an I/O Planning project.

- **Search:** Search the I/O Ports view for ports by name, or by keywords or values within the various port attributes.

- **Collapse All:** Display buses by name, and do not display individual bits of the bus.

- **Expand All:** Show all pins of a bus expanded.

- **Group by Interface and Bus:** Display the ports by interface, or alphabetically by name.

- **Create I/O Port Interface:** Define a new port interface to group ports into. You can select and place port interfaces as one object within the I/O planning environment.

- **Schematic**: Open a Schematic view for selected I/O ports.

- **Automatically scroll to selected objects:** Scroll the I/O Ports view to display objects selected in other views like the Netlist or Device views.

You can select ports and interfaces from the I/O Ports view and assign them to Package pins, or Device resources, using the I/O Planning view layout. Using the popup menu of the I/O Ports view you can:

- **Create Interface**: Define an Interface to group related ports.

- **Assign to/Unassign from Interface**: Assign a port or multiple ports to a previously defined Interface. Or, unassign ports from an Interface.

- **Configure I/O Ports:** Assign various properties of the selected I/O ports.

- **Reset Invalid Port Properties:** Reset any invalid properties on the specified port to the default value.

- **Reset Port Properties:** Reset all properties on the specified port to the default values.

- **Set Direction**: Define the direction for the port as Input, Output, or Bidirectional.

- **Split Diff Pair**: Split a differential pair of ports into two separate ports.

- **Place I/O Ports in an I/O Bank:** Assign the currently selected ports onto pins on the specified I/O Bank.

- **Place I/O Ports in Area:** Assign the currently selected ports onto pins in the specified area.

- **Place I/O Ports Sequentially:** Assign the currently selected ports individually onto pins.

- **Fix/Unfix:** Fixes or Unfixes the selected placed I/O ports. See Understanding Fixed and Unfixed Placement Constraints, page 324.

- **Unplace**: Unplace the selected I/O ports.

- **Swap Locations**: Lets you swap the pin assignments of two selected ports.

- **Export I/O Ports**: Write the contents of the I/O Ports view to a CSV, UCF, Verilog, or VHDL file.

- **Export to Spreadsheet**: Write the contents of the I/O Ports view to a Microsoft® Excel spreadsheet.
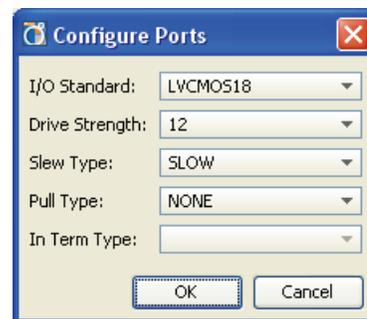
Using the Package Pins View

The Package Pins view displays I/O related package information. You can sort and filter the table to analyze the I/O pins and I/O ports information.

To invoke the Package Pins view, select **Window > Package Pins**. Figure 4-62 shows the Package Pins view.

Name	Prohibit	Port	I/O Std	Dir	Vcco	Bank	Bank Type	Type	Diff Pair
⊟ I/O Bank 1 (Standard) (69)					1.8		Standard		
E22		DataOut_pad_0_o[3]	LVCMOS18*	Output	1.8	1	Standard	User IO	L01P
D22	☐					1	Standard	User IO	L01N
C23		LineState_pad_0_i[0]	LVCMOS18*	Input	1.8	1	Standard	User IO	L04P
B23	☐					1	Standard	User IO	L04N
A23		VControl_pad_0_o[0]	LVCMOS18*	Output	1.8	1	Standard	User IO	
F21	☐					1	Standard	User IO	L05P
E21		DataOut_pad_0_o[2]	LVCMOS18*	Output	1.8	1	Standard	User IO	L05N
C22		DataOut_pad_0_o[7]	LVCMOS18*	Output	1.8	1	Standard	User IO	L06P
B22		VStatus_pad_0_i[0]	LVCMOS18*	Input	1.8	1	Standard	User IO	L06N
A22		VStatus_pad_0_i[1]	LVCMOS18*	Input	1.8	1	Standard	User IO	
D21		DataOut_pad_0_o[6]	LVCMOS18*	Output	1.8	1	Standard	User IO	L07P
C21		DataOut_pad_0_o[0]	LVCMOS18*	Output	1.8	1	Standard	User IO	L07N
B21		DataOut_pad_0_o[1]	LVCMOS18*	Output	1.8	1	Standard	User IO	L08P

Tcl Console   Messages   Compilation   Reports   Design Runs   **Package Pins**   I/O Ports   Timing

Figure 4-62: **Package Pins View**

Device pin information such as I/O Bank number, Bank Type, Differential pair partners, Site Types, and Min/Max package delay are listed for each package pin. The Bank Type column identifies the HP/HR banks of the Virtex-7, Kintex™-7, and Artix™-7 devices.

Note: The unit of measurement for the Min/Max package trace delay in the Package Pins view is in picoseconds (ps).

Table values appear as follows:

- Black or default values.
- Non-default values are marked with an asterisk.
- Red for illegal values.

You can sort the information in the Package Pins view by clicking any of the column headers. Clicking again reverses the sort order. To sort by a second column, press the **Ctrl** key and click another column. You can add as many sort criteria as necessary to refine the list order. Refer to the Using Tree Table Style Views, page 113 for more information on sorting the information in the Package Pins view.

Cells with editable values can be directly edited in the Package Pins view, either by entering text or by selecting from drop-down menus.

Using Package Pins View Commands

- **Search**: Search the Package Pins view for ports by name, or by keywords or values within the various pin attributes.
- **Collapse All**: Display I/O Banks by name, and do not display individual pins of the bank.
- **Expand All**: Show all pins of an I/O Bank expanded.

- **Group by I/O Bank**: Group the pins by I/O Bank, or list them alphabetically by name.

- **Automatically scroll to selected objects**: Scroll the Package Pins view to display objects selected in other views like the Netlist or Device views.

The commands available from the popup menu of the Package Pins view are:

- **Package Pin Properties**: Opens the Package Pin Properties view for the selected pin, if it is not open, and switches focus to that view.

- **Copy Rows**: Copies the selected rows of the Package Pins view, to let you paste them into a text editor.

  Note: To copy into a spreadsheet, use the Export I/O Ports command to export a CSV file.

- **Unplace:** Unplace the I/O ports assigned to the selected pins..

- **Swap Locations**: Lets you swap the pin assignments of two selected pins.

- **Set Prohibit:** Prohibit the assignment of a port to the selected pin or pins.

- **Clear Prohibit**: Remove the PROHIBIT property from the selected pin or pins.

- **Set User Column Values**: Specify the Column Name and Value of a custom column for assignment to the selected pins.

  Note: If the Column Name already exists the specified value is assigned selected pins.

- **Configure I/O Ports:** Assign various properties for the I/O ports assigned to the selected pins.

🖼	Package Pin Properties...	Ctrl+E
🖹	Copy Rows	
	Unplace	Ctrl+U
	Swap Locations	Ctrl+W
🔖	Set Prohibit	
	Clear Prohibit	
	Set User Column Values	
	Configure I/O Ports...	
	Create a DCI Cascade	
🖊	Highlight	▸
	Unhighlight	
◆	Mark	Ctrl+M
	Unmark	Ctrl+Shift+M
🖹	Export I/O Ports...	

- **Create DCI Cascade:** Opens the DCI Cascade Editor to tie I/O Banks together for DCI reference voltage purposes. See Configuring DCI_CASCADE Constraints, page 259 for more information.

- **Highlight/Unhighlight:** Highlight or unhighlight the selected rows in the Package Pins view. Highlighted objects are also highlighted in other open views, such as the Package view, or the I/O Ports view.

- **Mark/Unmark:** Mark or unmark the selected rows in the Package Pins view. Marked objects are also marked in other open views, such as the Device and Package views.

- **Export I/O Ports**: Write the contents of the I/O Ports view to a CSV, UCF, Verilog, or VHDL file.
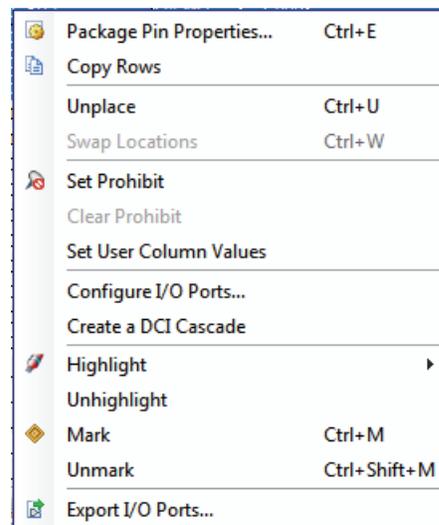
Using the Design Runs View

You can use the Design Runs view to view, configure, launch, and analyze Synthesis and Implementation runs. The Design Runs view offers commands to manage, launch, and reset runs for synthesis and implementation. Select **Window > Design Runs** from the main menu to open the Design Runs view. Figure 4-63 shows the Design Runs view.
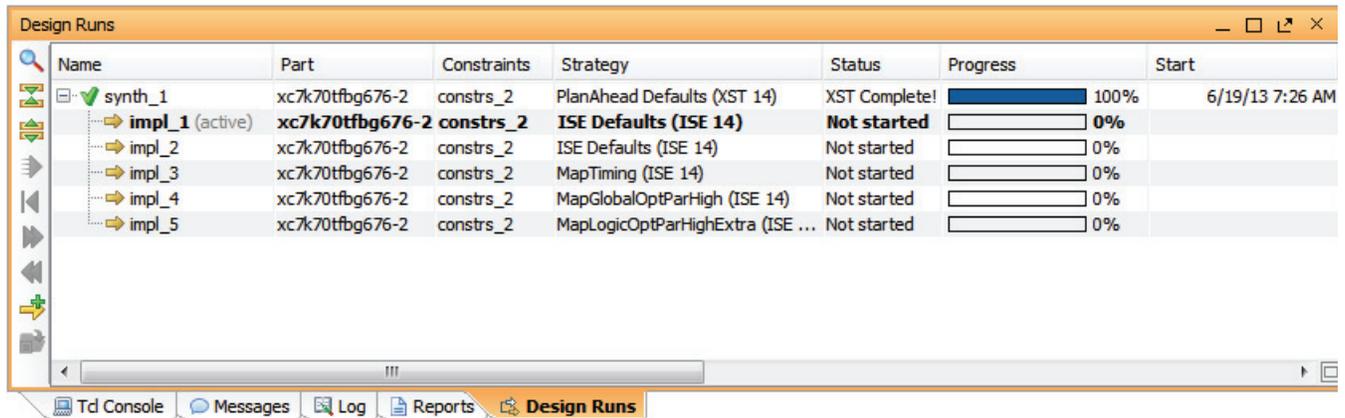
Figure 4-63: **Design Runs View**

As runs are created, launched, or imported, the run status shows in the Design Runs view. The view displays the status and results of the design runs defined, and provides commands to modify, import, launch, and manage the design runs.

The view indicates the runs as follows:

- Currently running with a spinning blue vortex.

- Completed runs have a blue check mark icon.

Run information displays as the commands are being run. You can close the PlanAhead tool without affecting in-progress runs. When you re-open a project, the PlanAhead tool updates the run status to reflect the latest status, which displays in the Design Runs chart.

The columns used for tracking information are:

- **Name**: Displays run name.

- **Part**: Indicates the target part selected for the run.

- **Constraint**: Displays the constraint set used for the run.

- **Strategy**: Displays the strategy assigned to the run. Strategies appearing with an asterisk (*) indicate that some command options for the strategy have been overridden. See Defining Synthesis Runs, page 200 or Setting Implementation Options, page 291 for more information.

- **Status**: Indicates run status as not started, running, or complete.

- **Progress**: Provides an indication of overall completion of the run. This encompasses the entire ISE implementation tool sequence (`ngdbuild` to `XDL`).

- **Start**: Reports the start time for the run.

- **Elapsed**: Reports the elapsed time for the run.

- **Device Utilization** (for synthesis runs only): Indicates the LUT utilization for the run.

- **Fmax** (for synthesis runs only): Indicates the expected clock frequency for the run from the XST Synthesis report.

- **Timing Score** (for Implementation runs only): Indicates the current timing score on the run in progress or after completion.

- **Unrouted Nets** (for Implementation runs only): Indicates the current number of unrouted nets on the run in progress or after completion.

- **Description**: Displays the description associated with the run. This description is set initially to a strategy description when that strategy is applied to the run; however, the description can be modified later.

The table is updated dynamically as the run commands progress. Runs that are launched outside of the PlanAhead tool using generated scripts cause the table to update upon invoking the software.

Using the Design Runs View Popup Menu Commands

The Design Runs view popup menu contains the following commands:

- **Run Properties**: Displays the Run Properties view. See Using the Run Properties View, page 154.

- **Delete**: Deletes the selected runs and removes the associated run data from disk. You are prompted to confirm before the runs are deleted.

   Note: You cannot delete the active runs.

- **Make Active**: Sets the selected run as the active run. The active run launches automatically when the Run Synthesis or Run Implementation command is used. The results for the active run display in the Messages, Log, Reports, and Project Summary views.

- **Change Run Settings**: Lets you change the strategy and command-line options for the selected synthesis or implementation run. See Defining Synthesis Runs, page 200 or Defining Implementation Runs, page 287 for more information.

- **Save as Strategy**: Lets you save the current strategy to a new strategy file for future use and modification.

🖼	Implementation Run Properties...	Ctrl+E
✖	Delete	Delete
	Make Active	
	Change Run Settings...	
	Save As Strategy...	
⇒	Launch Runs...	
◄	Reset Runs	
⇒⇒	Launch Next Step: Bitgen	
	Launch Step To	▶
◄◄	Reset to Previous Step: XDL	
📲	Open Elaborated Design	
📲	Open Synthesized Design	
📲	Open Implemented Design	
📲	Generate Bitstream	
📋	Copy Run...	
🔧	Create Runs...	
🗂	Open Run Directory...	
◎	Launch ChipScope Analyzer...	
🔧	Launch iMPACT...	
🖥	Launch FPGA Editor...	
🖥	Launch XPower Analyzer...	
	Export to Spreadsheet...	

- **Launch Runs**: Invokes the Launch Selected Runs dialog box to launch the selected runs.

- **Reset Runs**: Invokes the Reset Runs dialog box to remove previous run results and to set the run status back to Not Started for the selected runs.

- **Launch Next Step: <*Step*>:** Launches the next step of the selected run for incremental runs. For implementation runs the available steps are NGDBuild, MAP, PAR, TRCE, XDL, BitGen. Synthesis only has one step: XST.

- **Reset to Previous Step: <*Step*>:** Resets the selected run to the prior incremental step. This allows you to step backward through a run, to make any needed changes, then step forward to incrementally complete the run.

- **Open Elaborated Design**: Creates an Elaborated Design for the RTL sources files, with the constraint set and target part from the selected run.

- **Open Synthesized Design**: Opens a Synthesized Design for the selected run.
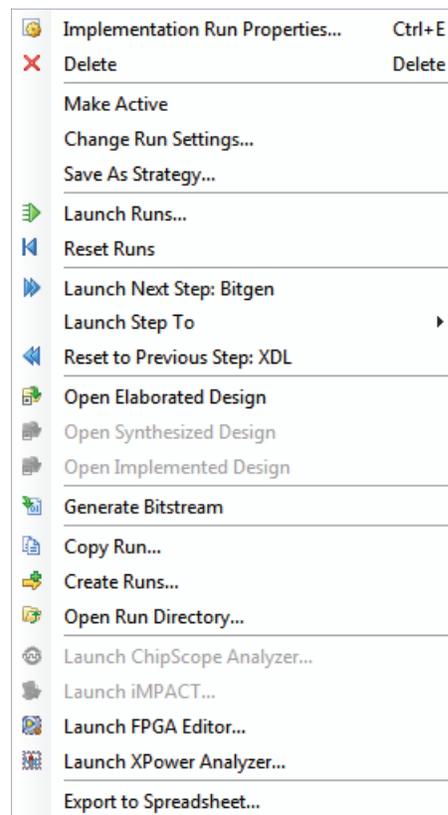
- **Open Implemented Design**: Opens an Implemented Design for the selected run.

- **Generate Bitstream**: Invokes the Generate Bitstream dialog box to create a bitstream. This command is available for completed Implementation runs only. See Generating Bitstream Files, page 365.

- **Copy Run**: Creates a new run using the same Strategy as the selected run.

- **Create Runs**: Invokes the Create New Runs wizard to create and configure new Synthesis or Implementation runs. See Chapter 6, Synthesizing the Design or Chapter 9, Implementing the Design for more information.

- **Open Run Directory**: Opens a file browser in the selected run directory on disk.

- **Launch ChipScope Analyzer**: Invokes ChipScope analyzer using the BIT file of the selected Implemented Design.

- **Launch iMPACT**: Invokes iMPACT using the BIT file of the selected Implemented Design.

- **Launch FPGA Editor**: Invokes FPGA Editor on the selected Implemented Design.

- **Launch XPower Analyzer**: Invokes XPower Analyzer with the selected Implemented Design.

- **Export to Spreadsheet**: Export information in the Design Runs view into a spreadsheet file.

- **Promote Partitions**: Invokes the Promote Partitions dialog box to promote implemented partitions. This option is only available when partitions are defined in the design. See Chapter 13, Using Hierarchical Design Techniques for more information.

Using the Text Editor

The Text Editor in PlanAhead provides the ability to edit a variety of text files in a context sensitive editor. PlanAhead supports a variety of formats: Verilog and Verilog header files, VHDL files, Constraint files, TCL scripts, PlanAhead Journal and Log files, and simple text files. Some of these files are supported with context sensitive editing to make it easy to identify keywords and comment lines within the file.

Refer to Setting Fonts for Text Editor, page 179 for more information on configuring the Text Editor.

Opening a Text File

You can open a file for editing in the Text Editor by selecting the file in the Sources view, and selecting the **Open File** command from the right mouse popup menu.

You can also open a file using the **File > Open File** command from the main menu. This command opens a file browser for you to navigate to the file and open it for editing.

You can double-click the file name from a warning or error message in the Messages view and the PlanAhead tool opens the selected file in the Text Editor.

You can also open the PlanAhead tool Journal and Log files in the Text Editor by using the **File > Open Log File** and **File > Open Journal File** as appropriate.

Creating a New Text File

You can use the **File** > **New File** command from the main menu to create a new file and open it for editing in the Text Editor. This is useful for creating text files that capture portions of the Tcl Console, Compilation logs, or errors or warnings from the Messages view.

This command opens a file browser for you to navigate to the desired folder and to specify the name of a new file to be created.

Text Editor Commands

The Text Editor commands are available from the popup menu or from the view-specific toolbar buttons. The commands are:

	Save File	Ctrl+S
	Save File As...	
	Undo	Ctrl+Z
	Redo	Ctrl+Shift+Z
	Cut	Ctrl+X
	Copy	Ctrl+C
	Paste	Ctrl+V
	Duplicate Selection	Ctrl+D
	Show Find	Ctrl+F
	Replace...	Ctrl+R
	Indent Selection	Tab
	Unindent Selection	Shift+Tab
//	Toggle Line Comments	Ctrl+Slash
	Toggle Block Comments	Ctrl+Shift+Slash
	Select All	Ctrl+A
	Toggle Column Selection	
	Tabs	
	Find in Files...	Ctrl+Shift+F
	Replace in Files...	Ctrl+Shift+R
	Change Font Style	Ctrl+Shift+T
	Insert Template	
	Select Source File	
	Define Module...	

- **Save File**: Saves the individually displayed file.

- **Save File As**: Lets you rename and save a file.

- **Undo/Redo**: Reverses changes in sequential order.

- **Cut, Copy, Paste:** Cuts or copies selected text to the clipboard. Pastes the contents of the clipboard to the cursor location.

- **Duplicate Selection:** Copies the selected text and pastes it to the cursor location, immediately below or above the selected text.

- **Show Find:** Invokes the Find field to enter a text string to search for.

- **Replace**: Invokes the Replace dialog box to enter a text string to search and replace in the open file.

- **Indent Selection/Unindent Selection**: Inserts or removes a tab space on the selected line or lines.

- **Toggle Line Comments**: Lets you select a line of text, or group of lines, and insert a line comment symbol at the start of the line. This command removes the line comment symbol if the selected lines are already commented.

  Note: The comment symbol inserted is contextually dependent on the type of file being edited.

- **Toggle Block Comments**: Adds or removes a block comment (/*...*/) at the start and end of a selected block of text. This command is useful for commenting out a section of text in a single command.

- **Select All**: Selects all the text in the Text Editor.

- **Toggle Column Selection:** Lets you select a block of text characters as a grid of rows and columns, rather than selecting lines of text. This command can be toggled ON or OFF.

- **Tabs**: Specifies that the Text Editor should use the tab character ($\backslash t$), or use a specified number of spaces when a tab is inserted. This allows the text file to be portable to third party applications that might not properly handle tab characters.

- **Find in Files/Replace in Files:** Invokes the Find in Files dialog box for you to enter text strings for searching the selected files. The Find in Files view displays at the bottom of the

PlanAhead environment with the results of the search. You can also replace the search string with a new string using the Replace in Files command.

- **Change Font Style**: Lets you edit the fonts and colors associated with the Text Editor. See Setting Fonts for Text Editor, page 179 for more information.

- **Insert Template**: Inserts the currently selected Language Template into your text file at the location of your cursor. See Instantiating Language or Constraint Templates, page 169 for more information.

- **Select Source File:** Selects the current file in the sources view.

- **Define Module:** Opens the Define Module dialog box to specify a new Verilog or VHDL module definition to add to the open text file. See Defining New Modules, page 41 for more information. When you close the Define Module dialog box, the new module is inserted into the open text file at the location of the cursor.

Instantiating Language or Constraint Templates

The PlanAhead tool provides standard Verilog and VHDL language templates and UCF constraint templates for use with the Text Editor to quickly define certain logic constructs or design constraints. Selected templates can be instantiated into any file that is open in the Text Editor. Figure 4-64 shows the Template view with a selected language template.
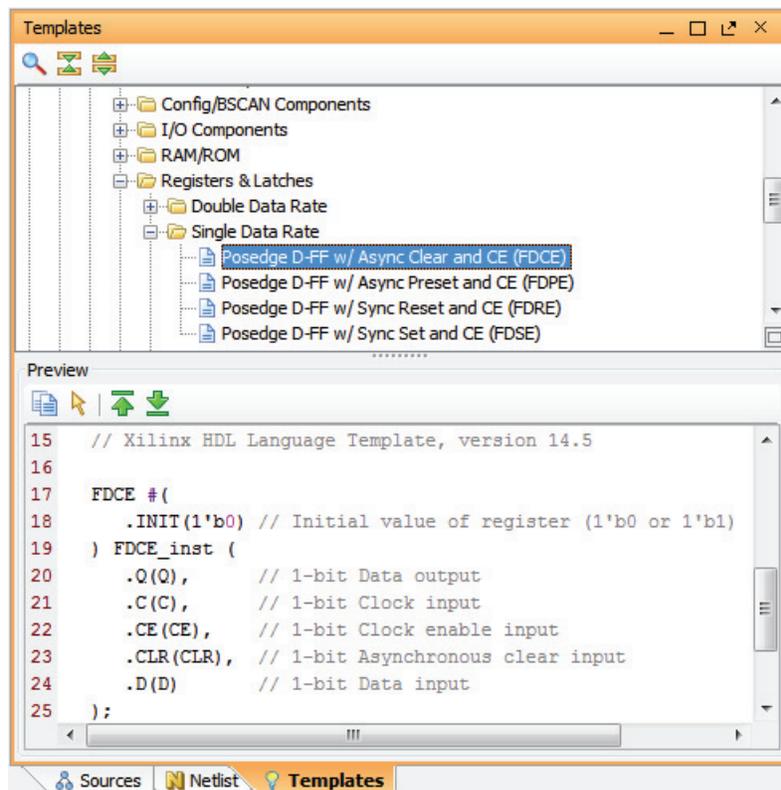

Figure 4-64: **Xilinx Language Templates**

To instantiate a language or constraint template:

1. Open the Template view to browse the available language templates:

   • Select the **Window > Language Templates** command from the main menu, or

   • Click the **Language Template** icon on the Text Editor tool bar.

2. Browse and select a template from the VHDL, Verilog, or UCF hierarchy in the Template view. Figure 4-64 shows an example of a Xilinx-supplied templates.

   The Preview window shows the text that will be inserted into the Text Editor window. The text cannot be edited in the Preview window prior to inserting it into the Text Editor.

3. In the Text Editor, place the cursor at the location in the file where the Xilinx language template should be inserted. The cursor should be placed at the location where you want to insert the start of the template text.

   Note: The PlanAhead tool does not overwrite existing text in the Text Editor when inserting the template. If a block of text is selected in the Text Editor window, the template text is inserted at the end of the block of text.

4. Insert the selected template into the text file by using the **Insert Template** command from the Text Editor toolbar or popup menu to add the template text at the cursor location.

5. Modify the inserted template text as needed to suit your design.

Configuring PlanAhead

The PlanAhead tool configuration options include: selection rule options, shortcut keys, general settings options, and window settings options. The following subsections describe the configuration options.

Setting General PlanAhead Options

To set the PlanAhead tool options, select **Tools > Options**. The PlanAhead Options dialog box opens for you to set options of how PlanAhead operates as an application.

Some of these options overlap with the Project Settings that can be specified for a specific project within PlanAhead rather than for the tool in general. See Configuring Project Settings, page 88, for more information.

Figure 4-65, page 171 shows the General options in the PlanAhead Options dialog box.
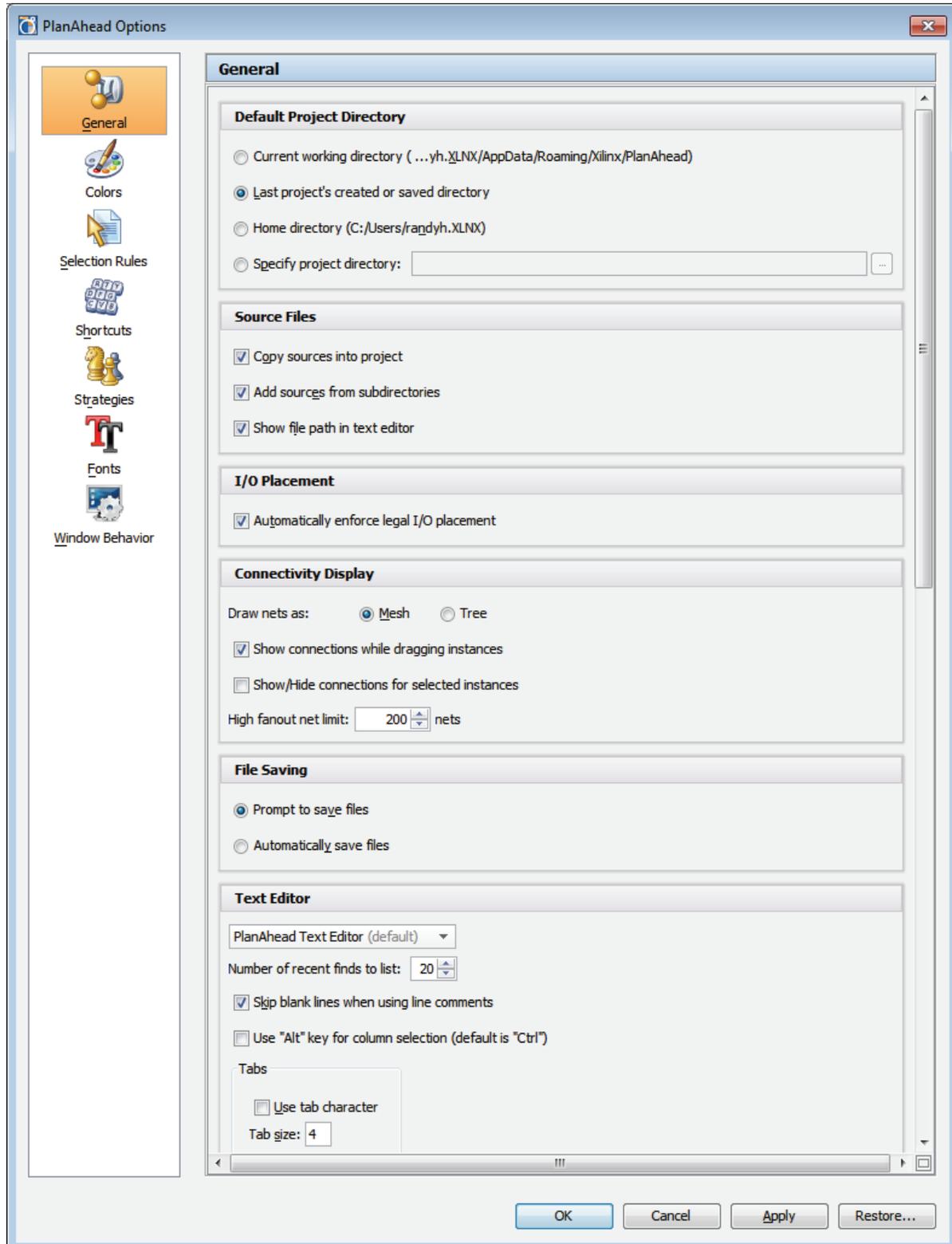
*Figure 4-65:* **General Options Dialog Box**

The General options are:

- **Default Project Directory**: Specifies where PlanAhead searches for existing projects, and where it writes newly created projects.

- **Source Files**: Contains checkboxes to copy sources, recurse through the subdirectories to locate source files, and to show the file path in the Text Editor.

- **I/O Placement**: Toggles ON or OFF the interactive I/O placement DRCs.

- **Connectivity Display**: Controls how connectivity displays in the Device view. See Setting Device View Display Options, page 139 for more information.

- **File Saving**: Defines whether PlanAhead should save project files automatically when closing, or prompt to save any changes.

- **Text Editor**: Defines the preferred text editor to use when opening RTL sources or UCF files for modification. There are a number of preconfigured Text Editors to choose from, or you can specify the command line to launch a third-party text editor. The native text editor is the default selection, which also opens if the PlanAhead tool detects issues with the specified text editor.

  - **Recent Finds, Skip Blank Lines, Use Alt Key:** Configure features of the Text Editor.

  - **Tabs**: Specifies that the Text Editor should use the Tab character `\t` or use a specified number of spaces when a tab is inserted. This allows the text file to be portable to third party applications that might not properly handle tab characters.

  Note: Some features of PlanAhead are not supported in third-party text editors.

- **Language & Tooltips:** Specify the language and behavior of the tooltips that appear when you move your mouse over a command in the PlanAhead tool. The supported languages are English, Chinese, and Japanese.

- **WebTalk**: Controls whether WebTalk is able to send Xilinx® usage information.

- **IP Catalog**: Lets you configure and update the IP catalog used in your projects. See Managing DSP Sources, page 72 for more information.

- **CoreGen**: Specifies the amount of memory allocated to Java when generating an IP core. You might need to increase the memory allocated to Java when generating very large IP cores. Conversely, you might need to reduce the allocated memory if you encounter an "Initialization Failed" error.

- **Miscellaneous**: Controls whether the software automatically searches the Xilinx website for new software updates, and defines how many previously opened projects and directories to list in the Getting Started page.

- **3rd Party Tools**: Lets you define the installation path to launch Mentor Graphics® ModelSim or QuestaSim.

Configuring the Viewing Environment

The PlanAhead tool has numerous user-configurable viewing options. The tool ships with default settings that you can customize and save for use in subsequent sessions.

The PlanAhead tool:

- Separately saves each available view layout.

- Creates a layout file to restore the overall window size and location.

- Stores the saved view configurations in your home directory upon exiting the software (see Outputs for Environment Defaults in Appendix A).

Specifying Colors

You can adjust view display options to control the appearance and behavior of the environment.

To view or edit the display options available in the PlanAhead Options dialog box, select **Tools > Options** and select **Colors** as shown in Figure 4-66, page 173. Changes to the various settings only take effect after you modify options and click **OK** or **Apply**.

There are default view settings for both light and dark background themes. To use either, select the **PlanAhead Light Theme** or **PlanAhead Default Theme** options in the Theme pulldown menu.

These default options are defined in the `planahead.ini` file. For more information, see Outputs for Environment Defaults in Appendix A.
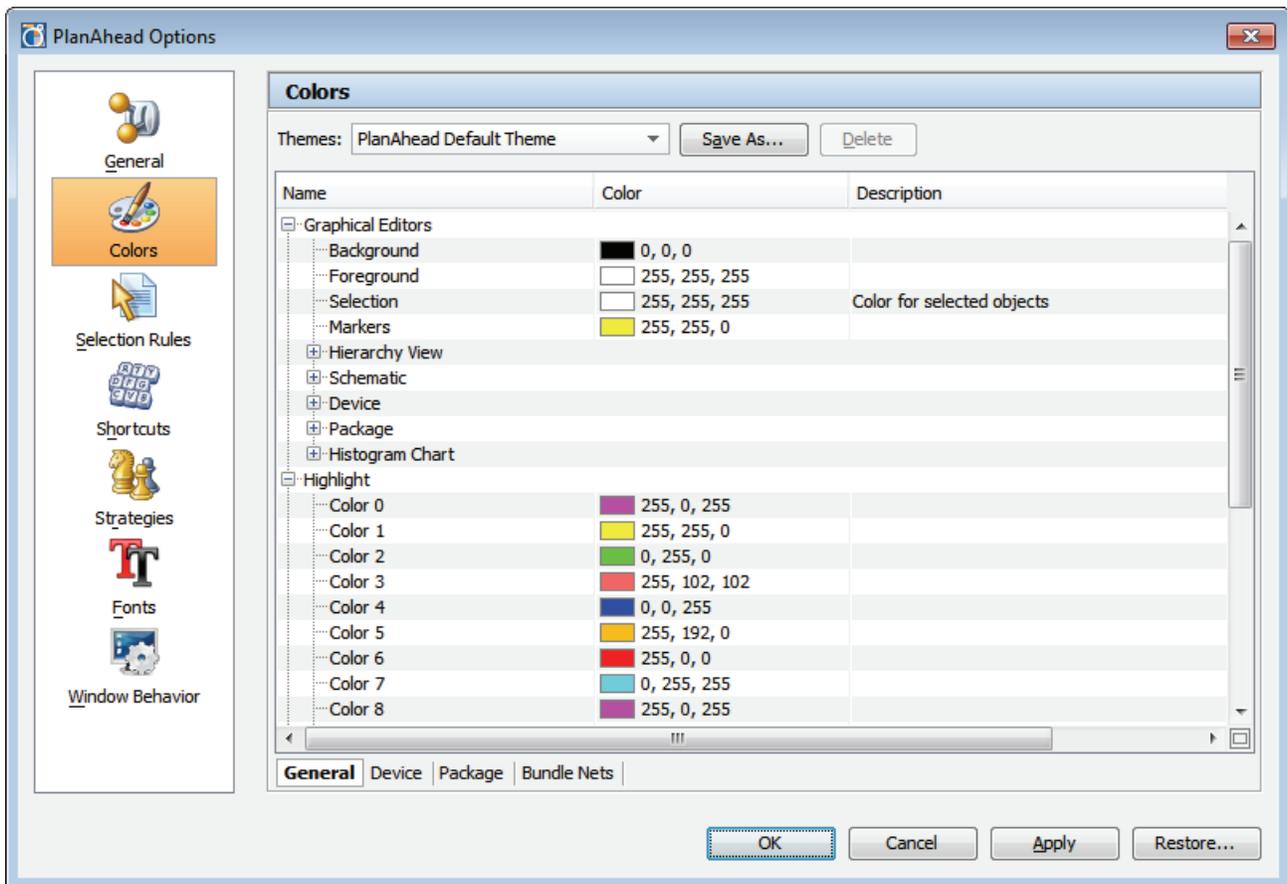

*Figure 4-66:* **Theme Options**

Customizing Display Themes

You can adjust the options that control the appearance of the PlanAhead tool viewing environment in the Themes options. The tabs at the bottom of the Themes options let you modify different elements of views: General, Device, I/Os, and Bundle Nets.

General Tab

Define colors for the PlanAhead tool to use when displaying different elements of the viewing environment, such as background and highlight colors.

The color values for specific elements can be changed by:

- Clicking on a color box to expose a pulldown menu and select from a list of available colors.

- Choosing **More Colors** to display even more colors to choose from.

- Entering a specific RGB value directly in the text field for the color.

Device Tab.

Adjust the default color and selectability for objects in the Device view

- In the Select column, toggle the check boxes off to make the object types unselectable.

- The display of items in the Device view is controlled through the layer slideout in the Device view menu bar. See Using the Device View, page 136 for more information.

Note: Some elements of the Device view are specific to a target device and so changes to the display color or selectability will not have effect except on that device.

Package Tab

Adjust the default color for each object type in the Package view.

- The display of elements in the Package view can be controlled through the layer slideout in the Package view menu bar. See Using the Package View, page 142 for more information.

Bundle Nets Tab

Configure the characteristics of displayed bundle nets.

- Use the **From/To** columns to adjust the signal count ranges for bundles. Each column represents a bundle net range which can be independently configured.
- Use the **Width** column to define the line width for the bundle in the Device view.
- In the **Display** column, toggle the check boxes off to hide the bundle.
- In the **Select** column, toggle the check boxes off to make the object types unselectable.

Saving a Custom Theme

After you have configured the various color settings in the Color Options dialog, you can save your own custom view settings for use in future sessions. Click the **Save As** button next to the Theme pulldown menu.

If you create your own theme, it is prudent to back-up the initialization file that contains the custom settings. For detailed information about the default and custom initialization files for the PlanAhead tool, see Outputs for Environment Defaults in Appendix A.

Setting Selection Rule Options

Selection rule options control the object selection settings for all views. When you select an object, other objects can become selected also (for example, selecting a Pblock also selects the assigned netlist instances).

For more information on setting selection rules, see Setting Selection Rules, page 118.

Configuring Shortcut Keys

Most commonly used commands have pre-defined shortcuts using combinations of key strokes. The shortcuts defined display next to the command in the popup menu. For example, press **F9** to access the **Fit Selection** command.

You can modify the default shortcut values by using the **Shortcuts** command from the Options dialog box, which is shown in Figure 4-67, page 176.

The Shortcuts dialog box lets you create new, custom shortcut settings.

At the top of the Options dialog box, the available shortcut schemas lets you manage shortcuts. You cannot modify the default shortcuts provided by the PlanAhead tool. You must create a new shortcut schema to customize.

1. Click **Copy** to create a new schema from the Default schema.

   Activate any schema in the list by selecting it from the pulldown menu of available schemas. You can change shortcuts in the schema in the bottom portion of the Options dialog box.

2. Search through the list of menus and views and select the command to enter a new shortcut.

   You can filter the commands listed for shortcut assignment using the **Filter** field. Enter any text string to filter the list of available commands. Also, you can use different shortcuts for the same command in different views.

3. Select **Add Shortcut**.

4. Type in the new shortcut in the Add Shortcut dialog box, then click **OK** to accept the new shortcut.

   User-specific shortcuts are saved to the shortcuts.xml file in the PlanAhead tool configuration directories. Refer to Outputs for Environment Defaults in Appendix A for more information.

5. To delete shortcuts, click the **Remove** button.
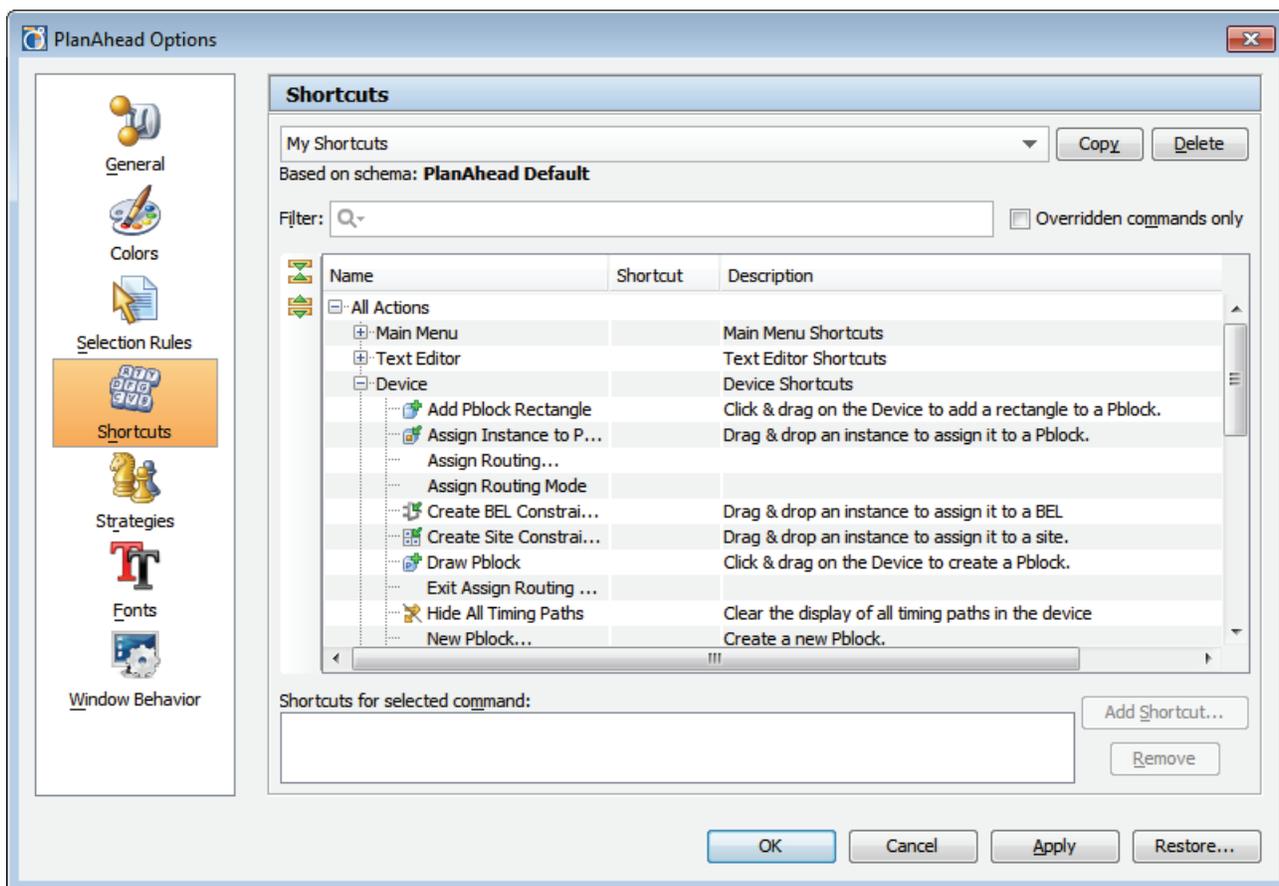
Figure 4-67: **Shortcuts Options**

Defining Strategies for Synthesis and Implementation

A strategy is a defined approach for resolving the synthesis or place and route challenges of the design. The strategy is defined in a pre-configured set of command-line options for the synthesis application, or the various utilities and programs run during implementation. Strategies are tool- and version- specific. Each major release has version-specific command line options. See Synthesis Settings, page 92 and Implementation Settings, page 93 for more information on setting strategies.
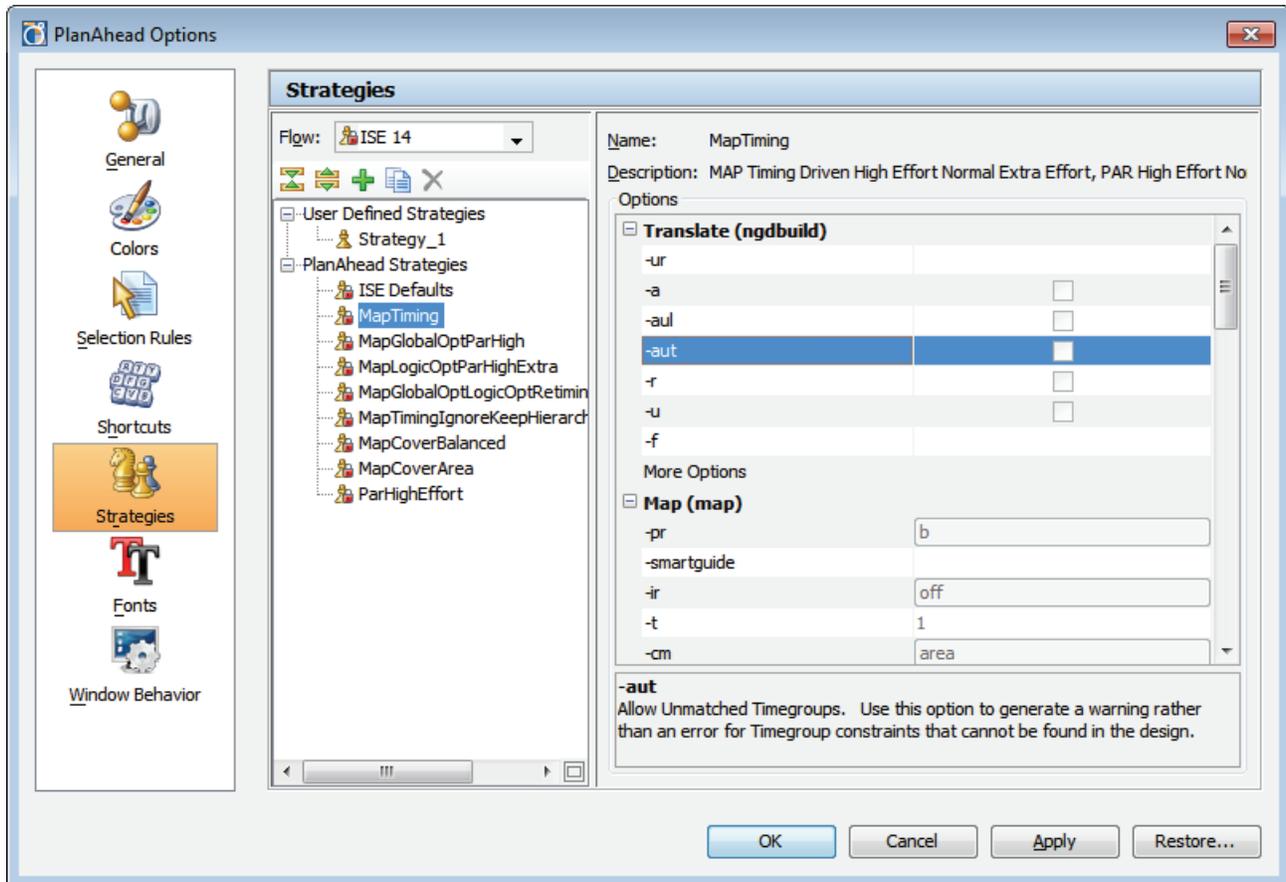
Figure 4-68: **Strategies Options**

The PlanAhead tool provides several commonly used strategies that are tested against internal benchmarks. The command-line settings for predefined PlanAhead synthesis and implementation strategies cannot be changed. However, you can copy and modify supplied strategies to create your own.

The PlanAhead tool writes the copied strategies to:

- Windows 7 - C:\Users\username\AppData\Roaming\Xilinx\PlanAhead\strategies

- Windows XP - C:\Documents and Settings\username\Application Data\Xilinx\PlanAhead\strategies

To review, copy, and modify strategies:

1. Select **Tools > Options** from the main menu.

2. Select **Strategies** in the left-side panel. The Strategies dialog box, shown in Figure 4-68, page 177, contains a list of pre-defined strategies for each ISE and Xilinx Synthesis Technology (XST) version.

3. In the **Flow** pulldown select the appropriate XST version for synthesis, or ISE version for implementation. A list of provided strategies and their various command-line options displays

   For more information about specific XST and ISE software options, see the *Command Line Tools User Guide (UG628)*, cited in Appendix E, Additional Resources.

4. To create a new strategy, use the Add Strategy command on the toolbar or from the popup menu.

    Alternatively, you can copy an existing strategy by using the **Create a copy of this strategy command** on the toolbar or from the popup menu. The PlanAhead tool creates a copy of the strategy in the User Defined Strategies list, and displays the command-line options on the right side of the dialog box for you to modify.

5. Provide a name and description for the new strategy as follows:

    • **Name**: Enter a strategy name to assign to a run.

    • **Type:** Specify Synthesis or Implement.

    • **Tool Version:** Specify the XST or ISE tool version.

    • **Description**: Enter the strategy description which is displayed in the Design Run results table.

6. Edit the options for the command-line tools used during the synthesis or implementation run.

    Click a specific command option to view a description of the option at the bottom of the dialog box.

7. Modify command options by clicking in the command option area (to the right), and selecting an option from the pull down menu, or typing a new value. Available command option settings display in the popup menu, as shown in Figure 4-68, page 177. Notice that you cannot change the default options for predefined PlanAhead tool strategies.

8. Click **Apply** and **OK** to save the new strategy.

    The new strategy is listed under **User Defined Strategy** and can be used for synthesis or implementation, as shown in Figure 4-69.
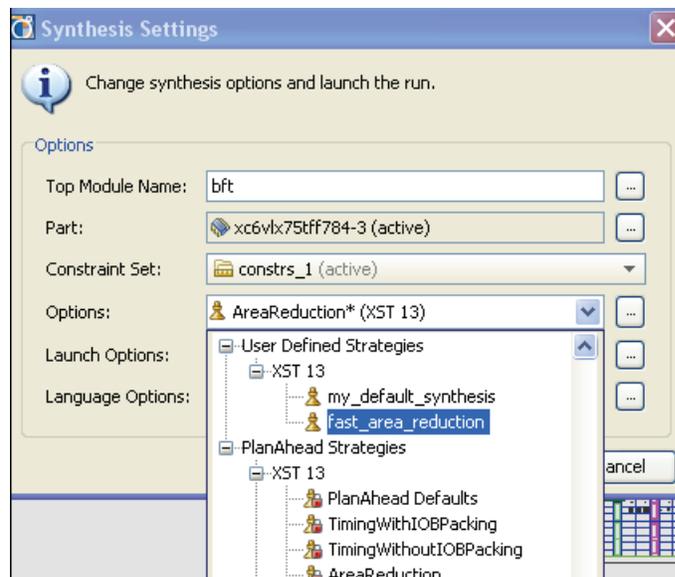
Figure 4-69: **Selecting User Defined Strategies**

Design teams that want to create and share strategies can copy any user-defined strategy to the *<InstallDir>*/strategies directory, where *<InstallDir>* is the installation directory.

Setting Fonts for Text Editor

You can define the fonts and colors used by the Text Editor. You can specify a different font, size, and color, for different elements of a text file such as comments and keywords. Figure 4-70 shows the effect of changing the comment font to blue text.
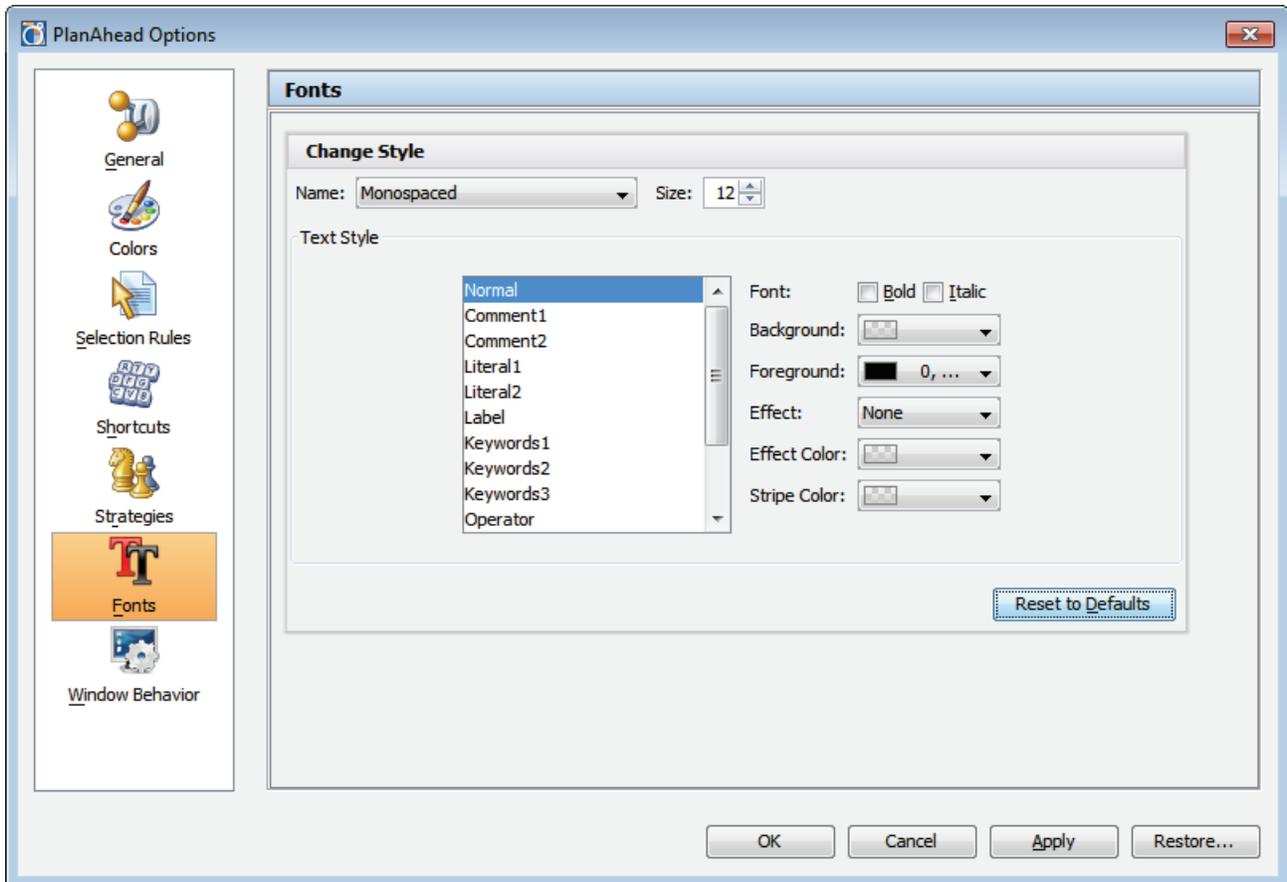
Figure 4-70:   **Font Options for PlanAhead Text Editor**

Use the **Reset to Defaults** command to reset the default font definitions.

Setting Window Behavior Options

To set the window behavior options, select **Tools > Options > Window Behavior**. Figure 4-71, page 180 shows the Window Behavior dialog box.

- **Warnings & Confirmation**: Defines how the PlanAhead tool shows warning and confirmation dialog boxes.
- **Notifications**: Define which notifications the tool will display.
- **Alerts**: Define alerts for success or failure of non-active runs.
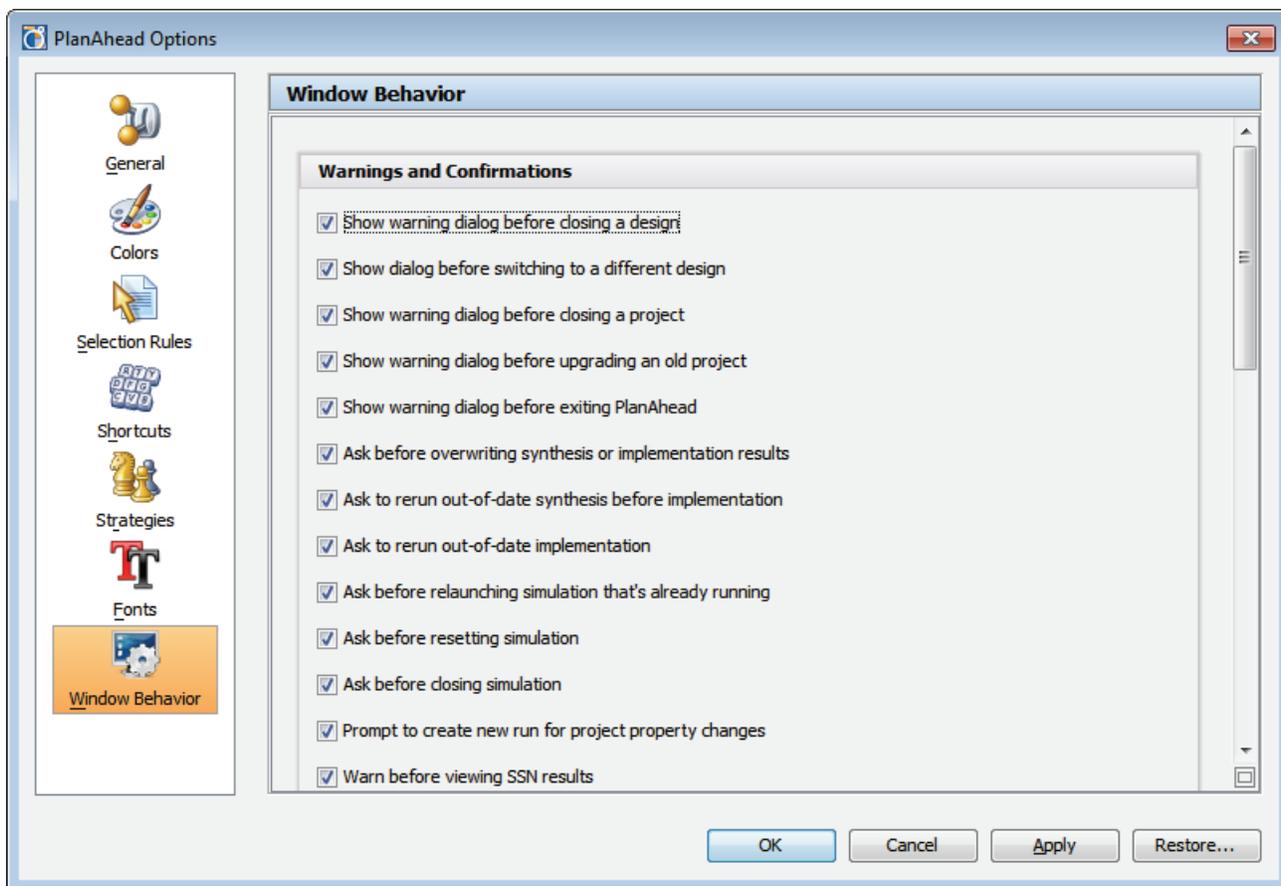
Figure 4-71: **Window Behavior Options**

Adding Custom Menu Commands

You can add system or user-defined Tcl commands to the main menu, and to the main toolbar menu with **Tools > Custom Commands > Customize Commands** from the main menu. This allows you to add frequently used Tcl commands to the GUI for use within the graphical environment.

The Customize Commands dialog box is opened when you run Customize Commands, as shown in .
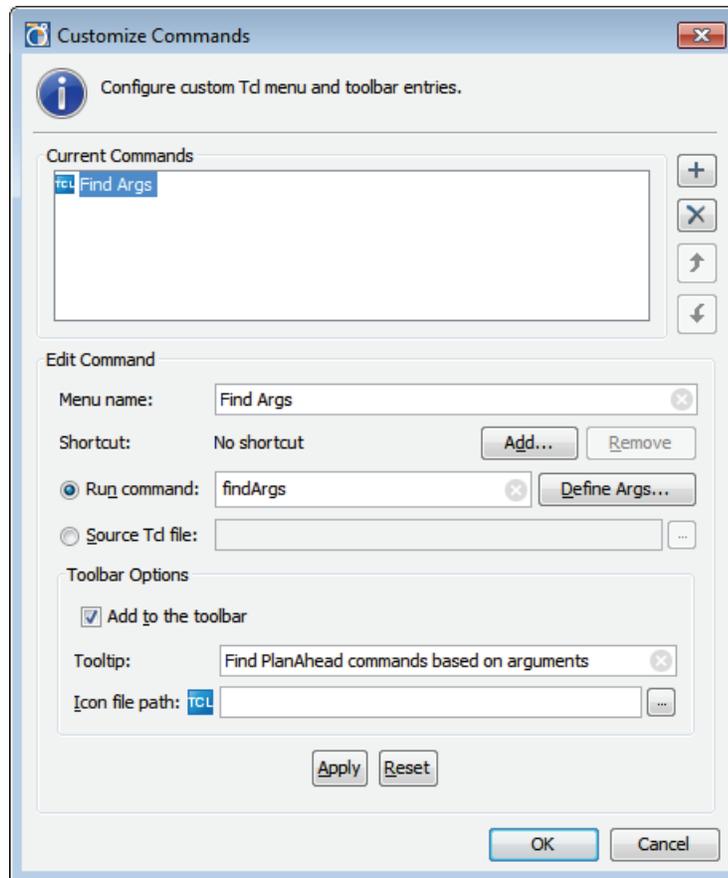
Figure 4-72: **Customizing Menu Commands**

The options of the Customize Commands dialog box are:

- **Current Commands**: Displays the list of currently defined custom commands.

  - **Add a New Command**: Displayed as a '+' icon, the Add command lets you specify new commands to add to the custom menu. Click **Add a New Command**, type the command name, and press **Enter** to add the command to the list of custom commands.

  - **Remove Selected Commands**: Displayed as an 'X' icon, the Remove command lets you select one or more commands and remove them from the custom menu.

- **Edit Command**: Lets you specify the attributes of the selected command in the Current Commands list.

  - **Menu Name**: Defines the name of the command as it is displayed on the Custom Command menu.

  - **Shortcut**: Defines a keyboard shortcut to use to run the Tcl command. Click **Add** to open the Add Shortcut dialog. Then you can enter a keystroke combination to run the selected command. The dialog box reports any commands that already use the specified shortcut.

  - **Run Command**: This radio button causes the specified Tcl command or procedure to be run from the custom menu.
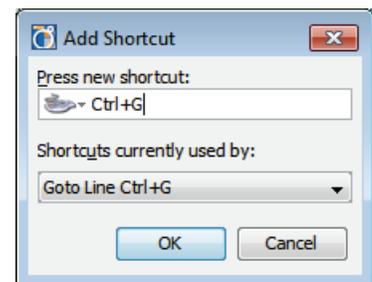
- **Define Args:** Lets you define arguments for the specified command, as seen in Figure 4-73.

  - **Name**: The name of an argument to supply to the Tcl command.

  - **Comments:** A brief description of the argument.

  - **Default Value:** A default value to use when the command is run.

  - **Optional:** Specifies that the argument is optional to the command.

- **Source Tcl File**: This radio button causes the specified Tcl script file to be sourced rather than a single Tcl command or procedure to be run from the custom menu.
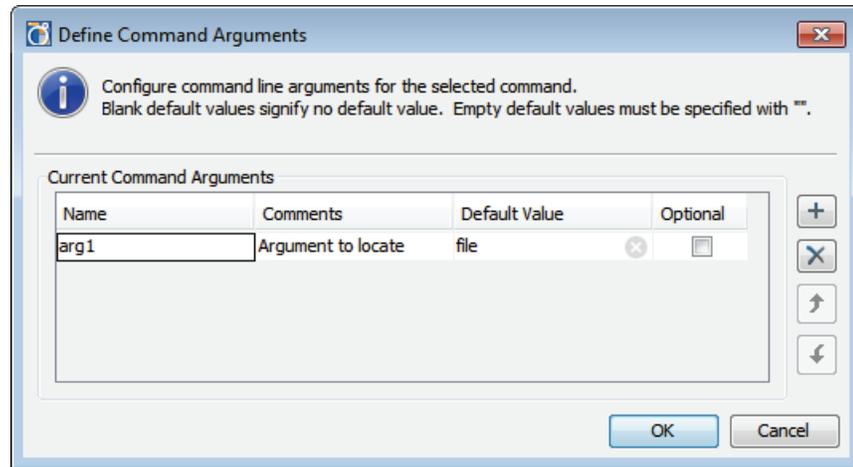
Figure 4-73: **Define Arguments**

- **Toolbar Options**: Specifies whether an icon for the command should be added to the main toolbar menu.

  - **Add to the toolbar**: Checkbox to enable the toolbar icon. When this checkbox is disabled, the command will not appear on the main toolbar menu.

  - **Tooltip**: Specifies the text to display as a tooltip when the mouse hovers over the command icon.

  - **Icon file path**: Specify the file path to the toolbar icon.

    Note: An icon file should be a PNG, JPG, or GIF file of approximately 20x20 pixels. The PlanAhead tool will resize larger images to fit onto the toolbar.

Currently defined custom commands are available from the **Tools > Custom Commands** menu item, with each command listed on the submenu.

The Custom Command menu is persistent with the PlanAhead tool, and is restored each time the tool is launched. Custom commands are saved to the commands.paini file output by the PlanAhead tool. Refer to Outputs for Environment Defaults in Appendix A for more information.

Elaborated RTL Design

The Project Manager in the PlanAhead tool lets you create and manage individual Register Transfer Level (RTL) files, and elaborate and analyze them in the context of an overall FPGA design. The PlanAhead tool includes:

- Source file management
- Context sensitive text editor for Verilog and VHDL files
- RTL schematic viewer
- RTL design rule checks (DRCs)
- Behavioral simulator

Managing Design Source Files

You can add and manage the following design source file types in a project:

- Verilog and VHDL RTL source files
- NGC, EDIF, and Xilinx® CORE Generator™ XCO files
- UCF constraint files
- Simulation test benches
- Sysgen Model (MDL) for DSP Modules
- XPS Projects (XMP) for Embedded Sources
- Block Memory Map (BMM) files

Most of the interaction with source files is from within the Sources view. See Using the Sources View, page 128.

Editing RTL Source Files

The PlanAhead RTL environment provides a Text Editor in which to create or modify RTL sources. The Text Editor is syntax sensitive, and uses color-coding to distinguish RTL keywords. You can open multiple files simultaneously, and each open file displays a view tab in the workspace to provide access to all open files.

When you modify a file, the PlanAhead tool appends an asterisk (*) to the file name in the view tab until the file is saved.

To save the file, use one of the following methods:

- Right-click in the Text Editor and select **Save File**
- Use the **Save File** toolbar button in the Text Editor, or
- Select **File > Save File**.

If you attempt to close a file with unsaved changes, the PlanAhead tool prompts you to save the changes. You can also use the **Save As** command to save the source file to a new location.

Using the Text Editor

The tool enables cross-probing to and from the Text Editor with other views, such as the Schematic, Messages, RTL Netlist, and Hierarchy views. For more information on the Text Editor, see Using the Text Editor, page 167.

Using Find Commands to Search Source Files

You can use **Find** or **Find in Files** to search for any given text string in an open source file or a selected set of source files. You can perform the following actions:

- Enter any text string, including wildcards (*), as search criteria.
- Use the filtering options to search source files, constraint files, and report files.

For more information, see Using the Find Commands, page 122.

Elaborating and Analyzing the RTL Source Files

The PlanAhead tool lets you elaborate and analyze the RTL source files without running synthesis. When an Elaborated Design is opened, the RTL source files and constraints files are elaborated into memory, and the RTL Netlist view, Schematic view, and Messages and Log view are opened. An Elaborated Design is shown in Figure 5-1, page 185.

You can open the Elaborated Design at any time in the design process to review and re-analyze the design; or to make any necessary changes.

Elaborating the RTL Source

Enabled RTL source files in the project are elaborated regardless of whether they are compiled as a part of the design during synthesis. The Messages view shows the messages from elaboration and compilation.

You can select the HDL language options used during elaboration in the PlanAhead Project Settings. See General Project Settings, page 89.

Elaboration results are not saved with the design. Every time you open the Elaborated Design, it is re-elaborated. After you synthesize the Elaborated Design, the PlanAhead tool saves it as a Synthesized Design.

After design source files are imported into the project, you can elaborate the design using one of the following commands:

- The **Open Elaborated Design** command in the RTL Analysis menu of the Flow Navigator to load the elaborated netlist, the active constraint set, and the target device into memory.
- **Flow > Open Elaborated Design** from the main menu.
- **New Elaborated Design** accessed from the RTL Analysis popup menu in the Flow Navigator.
- **Flow > New Elaborated Design** from the main menu.

When you open an Elaborated Design, the PlanAhead tool automatically elaborates the RTL source files, generates the top schematic view, and displays the design in the default view layout. Figure 5-1 shows the Elaborated Design default view layout.

The PlanAhead tool automatically identifies the top module for the design in most cases. In some case, where there may be multiple candidates, the tool will prompt you to choose the top module for the design. The current top module is identified by a special icon in the Hierarchy tab of the Sources view.

You can also manually define the top module using the **Set as Top** command from the popup menu of the sources view.
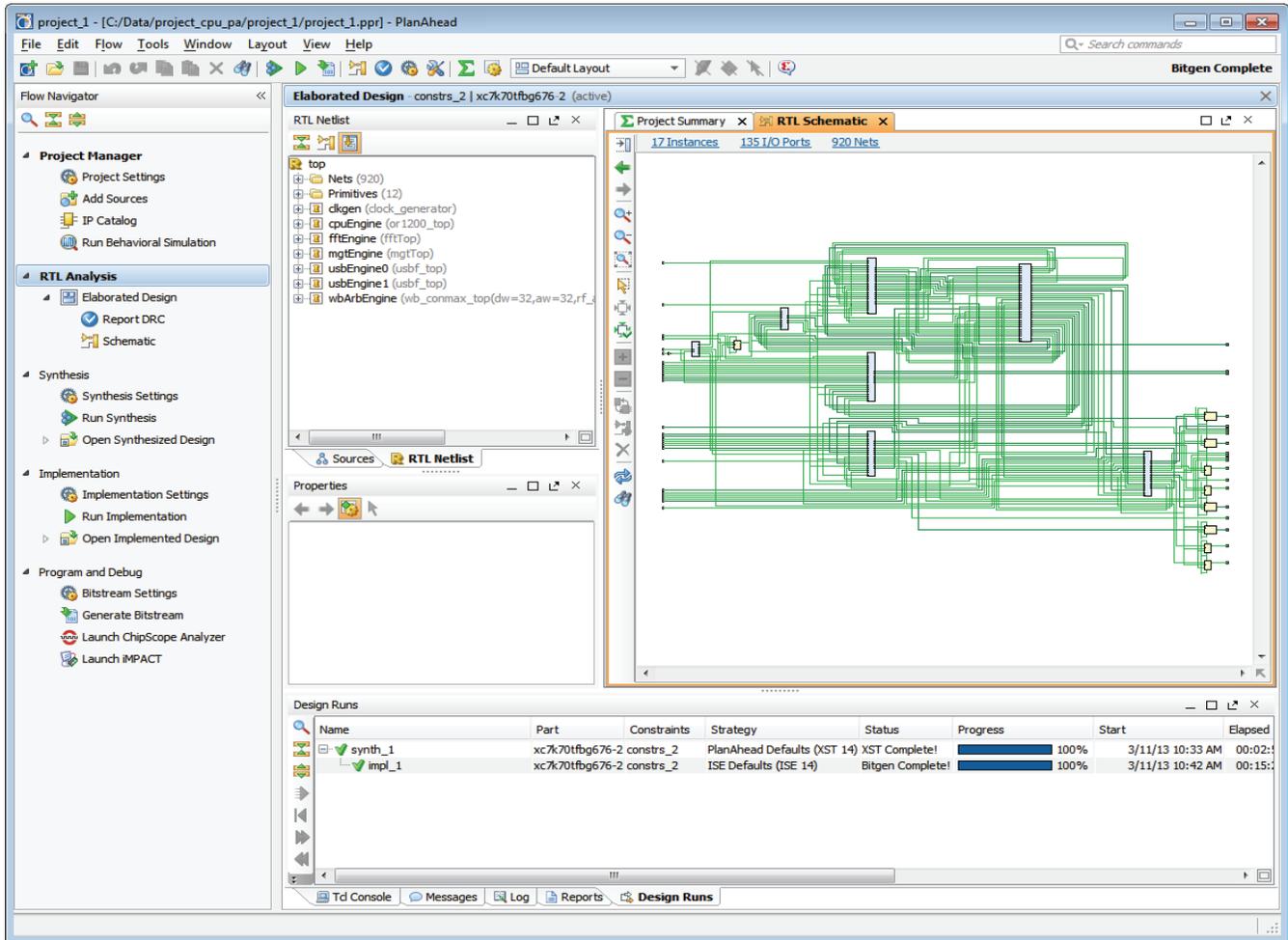
Figure 5-1: **Elaborated Design view**

The Messages view displays the results of the compilation and flags irregularities in the RTL source files under the heading of **Analysis**.

You can filter the Messages view to display errors or warnings or informational messages from the results of RTL elaboration. To enable or disable the display of Errors, Critical Warnings, Warnings, Informational, or Status messages select a checkbox in the banner of the Messages view.

You can select any of the warning or error messages in the Messages view to load the appropriate RTL source file with the selected source code highlighted in the Text Editor.

Resource Utilization of the Elaborated Design

The PlanAhead tool also applies the active constraint set to the elaborated design. This enables:

- I/O pin planning, based on the RTL port list, and

- Module-level floorplanning from the RTL logic hierarchy.

For information on creating and managing constraint sets, see Managing Constraints, page 53.

The PlanAhead tool provides device resource estimations based on the Elaborated Design by selecting **Run RTL Report Utilization** in the Project Summary. The RTL Resource Estimation displays in the Project Summary as shown in Figure 5-2.
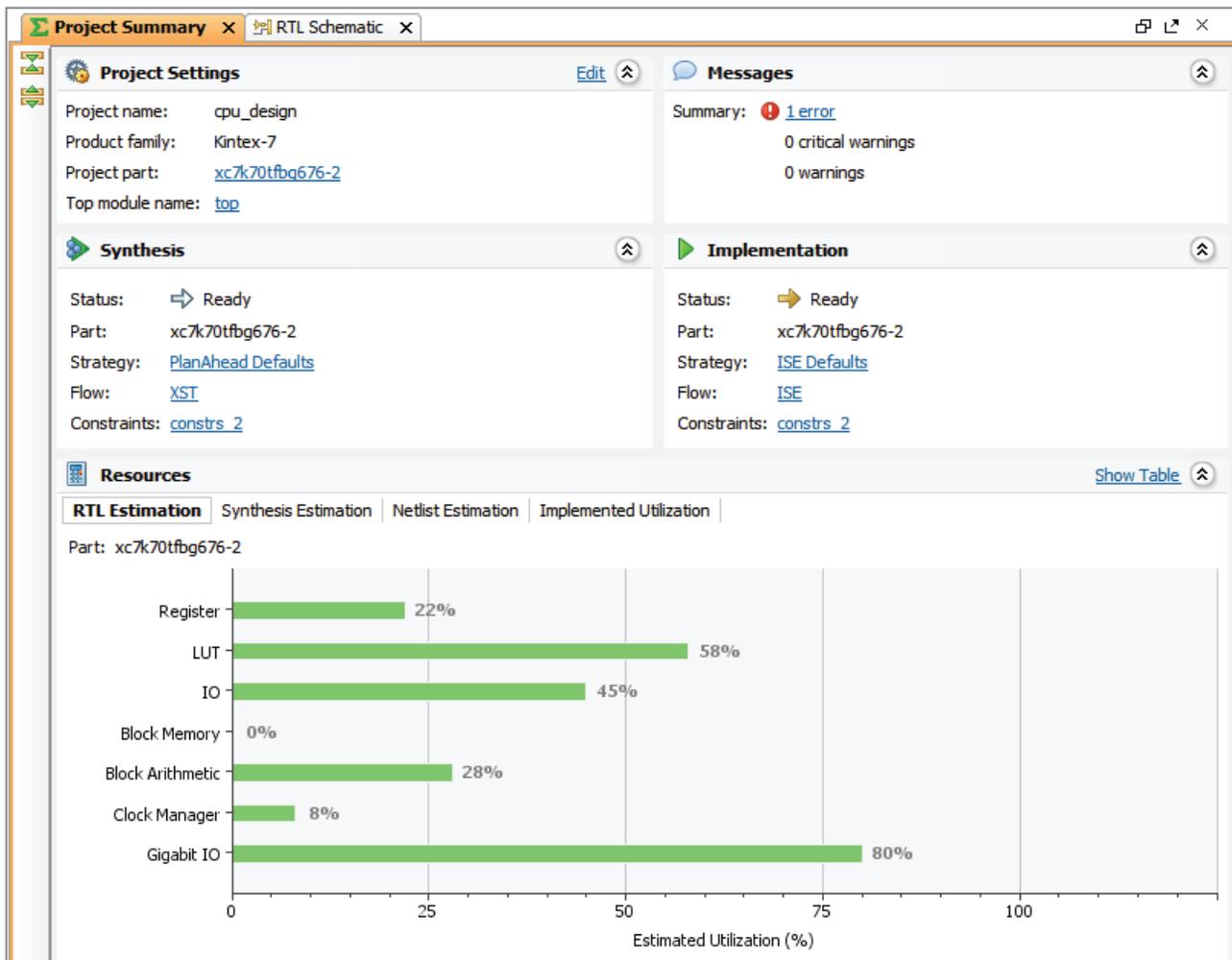
Figure 5-2: **Resource Utilization of an Elaborated Design**

The Resource Utilization view shows the resource utilization of device resources such as registers, LUTs, and block memory. The resources displayed in the Resource Utilization view is based on the logic hierarchy of the design. If a specific type of device resource is not used by the Elaborated Design, it is not displayed. A note at the bottom of the report indicates why a device resource is not displayed.

The PlanAhead tool determines the estimation of the hardware resources required for an Elaborated Design from pre-synthesis design data. These statistics are an early estimation and can change after synthesis or implementation.

Analyzing Resource Statistics in the Pblock Properties view

You can also select the ROOT design in the Physical Constraints view to see a more detailed estimation of resources in the Statistics tab of the Pblock Properties view. Use the **Windows > Physical Constraints** command to open the view.

The **Statistics** tab of the Pblock Properties view displays the estimated device resource requirements for the top module in the elaborated design, as seen in Figure 5-3.

Figure 5-3: **Viewing Physical Resource Estimates**

To save the resource estimation report to a text file, an XML file, or a CSV file, click the **Export Statistics** toolbar button in the Pblock Properties view.

Analyzing the RTL Logic Hierarchy

The PlanAhead tool provides the following views into the logical design hierarchy:

- The RTL Netlist view provides an expandable logic tree.

- The Hierarchy view provides a graphical representation of the logic hierarchy.

- The Schematic view provides a view in which to explore the logic and hierarchy in a schematic representation.

All views cross-select offering a unique set of capabilities to explore and analyze the logical design. For more information, see Using the Netlist View and Using the Hierarchy View in Chapter 4.

Exploring the Elaborated Design Schematic

You can select any level of logic hierarchy in the RTL Netlist view and display it in the RTL Schematic view.

To invoke the RTL Schematic view for any selected logic, select one of the following:

- **Tools > Schematic** from the main menu.

- **Schematic** from the popup menu in the RTL Netlist view.

For more information on traversing, expanding and exploring the RTL Schematic, refer to Using the Schematic View in Chapter 4.

After you have elaborated the Elaborated Design, you can use the **Find** command to search for logic objects using a range of filtering techniques. Refer to Using Find Commands to Search Source Files, page 184.

Exploring the RTL Source Files

You can also select any logic element in the RTL Netlist view and open the instantiation of that object in the RTL source file it is instantiated in, or open the definition of the logic in the RTL file it is defined in.

To open the instantiation or definition of any selected logic in the RTL source file, select the object and use the **Go To Instantiation** command. The PlanAhead tool opens the appropriate source file with the specific instance highlighted.

Performing Behavioral Simulation

The PlanAhead tool is integrated with the ISE Simulation tool, ISim. Xilinx ISim is a Hardware Description Language (HDL) simulator that lets you perform behavioral and timing simulations for VHDL, Verilog, and mixed VHDL/Verilog designs.

The PlanAhead tool also supports integration with Mentor Graphics® ModelSim or Questa® Advanced Simulator tool for behavioral or timing simulation. See Configuring Project Settings in Chapter 3 for instructions on specifying the target simulator.

You can launch behavioral simulation by selecting:

- **Tools > Simulation > Run Behavioral Simulation** from either from the main menu bar on an Elaborated Design, or

- **Run Behavioral Simulation** from the Project Manager menu of the Flow Navigator.

The PlanAhead tool opens the Launch Behavioral Simulation dialog box, as shown in Figure 5-4, page 189.
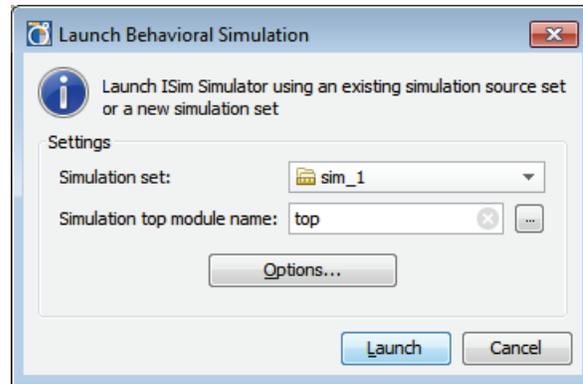
Figure 5-4: **Launch Behavioral Simulation**

The Launch Behavioral Simulation dialog box contains the following fields:

- **Simulation Set**: Specify the name of the simulation run. This lets you create different simulation runs with different design hierarchies and different options.

- **Simulation Top Module Name**: Specify the top of the design. This field is automatically populated with the defined Simulation Top Module Name, but you can specify a different top module to allow simulation from different levels of the hierarchy, or to elaborate different variations of the design. Click the file browse button to view top modules found in the design.

- **Options**: Opens the Simulation Options dialog box as shown in Figure 5-5, page 190 and Figure 5-6, page 191.

- **Launch**: Runs the ISim compile and elaboration steps and launches ISim in GUI mode.

- **Cancel**: Closes the dialog without launching ISim.

When you select the **Options** button, the PlanAhead tool opens the Simulation Options dialog box, which includes two option tabs: Launch Options and Language Options. The following subsections describe these options.

Specifying Simulation Options

When you select the Options button, the PlanAhead tool opens the **Simulation Options** form. This form includes the previously specified top module, and two option tabs:

Compilation tab: Specify command line options for the compiler used to prepare the design for simulation.

Simulation tab: Provides run-time directives to be used when the simulation is launched.

Advanced tab: Include all design sources for simulation.

The following subsections describe these options.

Compilation Tab

The **Compilation Option** tab of the Simulation Options dialog box contains the options to control the execution of fuse and ISim, as shown in Figure 5-5, page 190.
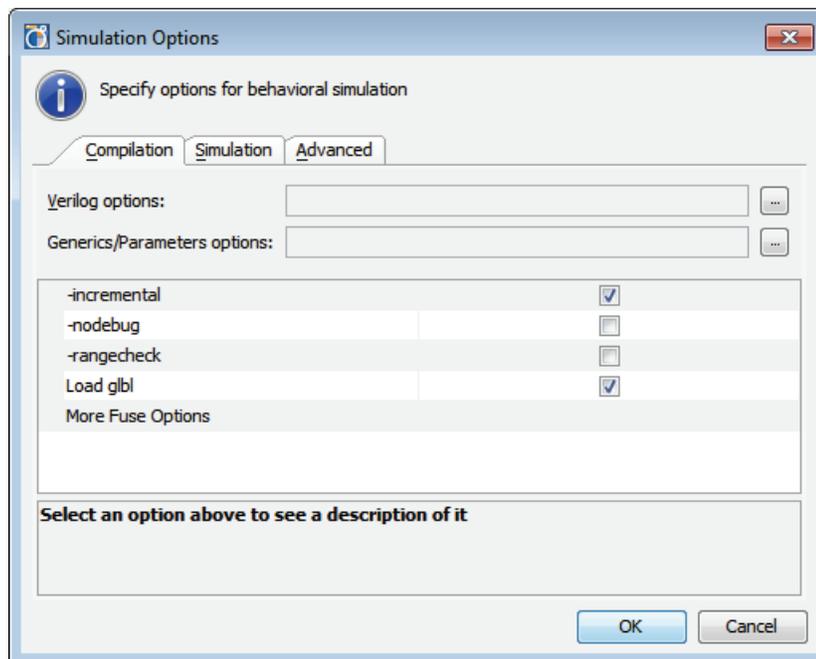
Figure 5-5: **Simulation Options Dialog Box**

The Compilation Options are:

- **Verilog Options**: Specify Verilog Search Paths, Macro definitions, and Uppercase identifiers. See Configuring Project Settings in Chapter 3 for more information.

- **Generics/Parameters Options**: Define values for VHDL generics or Verilog parameters.

- **-incremental**: Switch to indicate that fuse linker and compiler should compile only the files that have changed from the last compilation.

- **-nodebug**: Switch to indicate that fuse should create a simulation executable (.exe) that has no information for debugging your HDL code during simulation, resulting in faster simulation run times.

- **-rangecheck**: Switch to specify that fuse linker and compiler should perform a value range check on VHDL assignments during compilation. This option applies to VHDL code only.

  Note: This does not affect index range checking for arrays which ISim always checks.

- **Load glbl**: Switch to specify that the `glbl` module should be loaded during compilation. If the design uses a Verilog UniSim or a SimPrim library, you must enable this switch.

- **More Fuse Options**: Specify additional command line options for fuse. These commands should be typed in a single string with the command value pair. For instance:

  `-maxdelay -init_file <filename> -notimingcheck`

  You can also add the fuse options into a command file, and reference this file in the **More Fuse Options** field with the **-f** command, as follows:

  `-f <command_file>`

Simulation Tab

Figure 5-6 shows the **Simulation** tab of the Simulation Options dialog box.
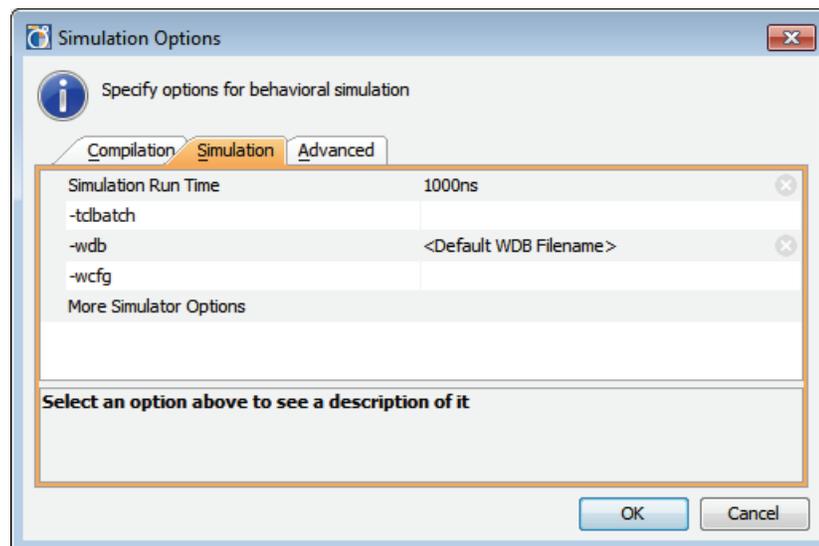
Figure 5-6: **Behavioral Simulation Language Options**

- **Simulation Run Time**: Specify length of simulation time with time unit. You can specify time in units of fs, ps, ns, us, ms, and sec. The default unit is ps. You can also specify `all` as a keyword to indicate that ISim should run until there are no more events to simulate.

- **-tclbatch**: Specify filename of Tcl commands listed in a batch file for execution by the simulator at run time. Commands in the specified batch file are executed sequentially until completion. ISim ignores any commands entered from the command prompt until batch file execution has completed.

  PlanAhead uses a **tclbatch** command to pass three required commands to ISim in a file called `isim.cmd`. The contents of this file are:

  ```
  onerror {resume}
  wave add /
  run <value>
  ```

  If you create a Tcl command to control the execution of the simulator at launch time, you must include these three commands in your `tclbatch` file. It is recommended that `onerror` be the first command listed, and that `wave add` and `run` be the last commands listed. You can add any other ISim command-line commands between `onerror` and `wave add`.

  Note: The `tclbatch` command file must have a file extension of either TCL or CMD to be properly handled by ISim.

- **-wdb**: Specify a filename to save the simulation waveform data. The simulation results of the signals being traced are saved to the specified filename in the working directory. The PlanAhead tool creates a `<top_module_name>.wdb` file by default.

- **-wcfg**: Specify a waveform configuration filename to use when opening the waveform data in the ISim GUI. The wave configuration file specifies settings such as the signal order, name style, radix and color.

- **More Simulator Options**: Specify additional command line options for ISim. The commands must be in a single string with the command value pair. For example:

  ```
  -log <filename> -transport_int_delays
  ```

  You can also add the ISim options into a command file, and reference this file in the **More Simulator Options** field with the **-f** command, as follows:

  ```
  -f <command_file>
  ```

Advanced Tab

Figure 5-7 shows the **Advanced** tab of the Simulation Options dialog box.
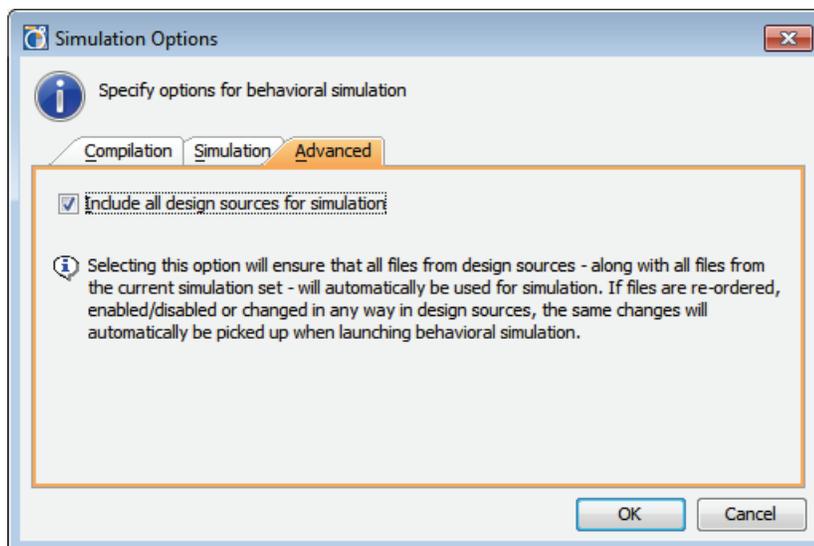
Figure 5-7: **Behavioral Simulation Advanced Options**

- **Include all design sources for simulation:** Ensures all design sources are available for use by the simulator.

ModelSim Options

When the target simulator has been set to QuestaSim/ModelSim, the compilation and simulation options presented by the tool will be different. The following are the options for ModelSim:

Compilation

- **VHDL Syntax**: Specifies the VHDL language version used by the source files and test bench. Valid values are 93 (1993), 87 (1987), 2002, and 2008.

- **Explicit Declarations**: Resolve ambiguous function overloading by using the "explicit" function declaration versus the function declaration automatically created by the compiler.

- **More VLOG Options**: Provide any added Verilog compiler (vlog) command line options to pass to the tool.

- **More VCOM Options**: Provide any added VHDL compiler (vcom) command line options to pass to the tool.

Simulation

- **Simulation Run Time**: Specify length of simulation time with time unit.

- **Log All Signals**: Save all signal values during simulation to help trace the source of an unknown signal state.

- **More VSIM Options**: Provide any added ModelSim (vsim) command line options to pass to the tool.

Advanced

- **Include all design sources for simulation:** Ensures all design sources are available for use by the simulator.

Launching the Simulator

To invoke ISim for behavioral simulation click **Launch**. The PlanAhead tool runs fuse, the ISim object compiler and linker, to compile and elaborate the Verilog and VHDL code.

The compiled object codes are then linked into a simulation executable named after the top module specified in the ISim Launch dialog box. When the ISim executable is complete, the PlanAhead tool launches the simulator. The following code snippet is an example of the command execution:

```
INFO: [Runs-8] Fuse completed.
INFO: [Runs-10] Launching ISim...
INFO: [Runs-11] Running '"C:/project_cpu_hdl/project_cpu_hdl.sim/
sim_1/top.exe"
-intstyle pa -gui -tclbatch ISim.cmd
-wdb "wdb_test1.wdb" -view "wcfg_test1.wcfg"'
```

The simulation executable runs with options specified in the Launch Options tab.

The PlanAhead tool launches ISim with the -gui option, which opens the ISim user-interface for you to interactively simulate the design. For more information about running ISim through the GUI, see the *ISim User Guide (UG660)*, cited in Appendix E, Additional Resources.

Figure 5-8, page 194 shows the ISim GUI.
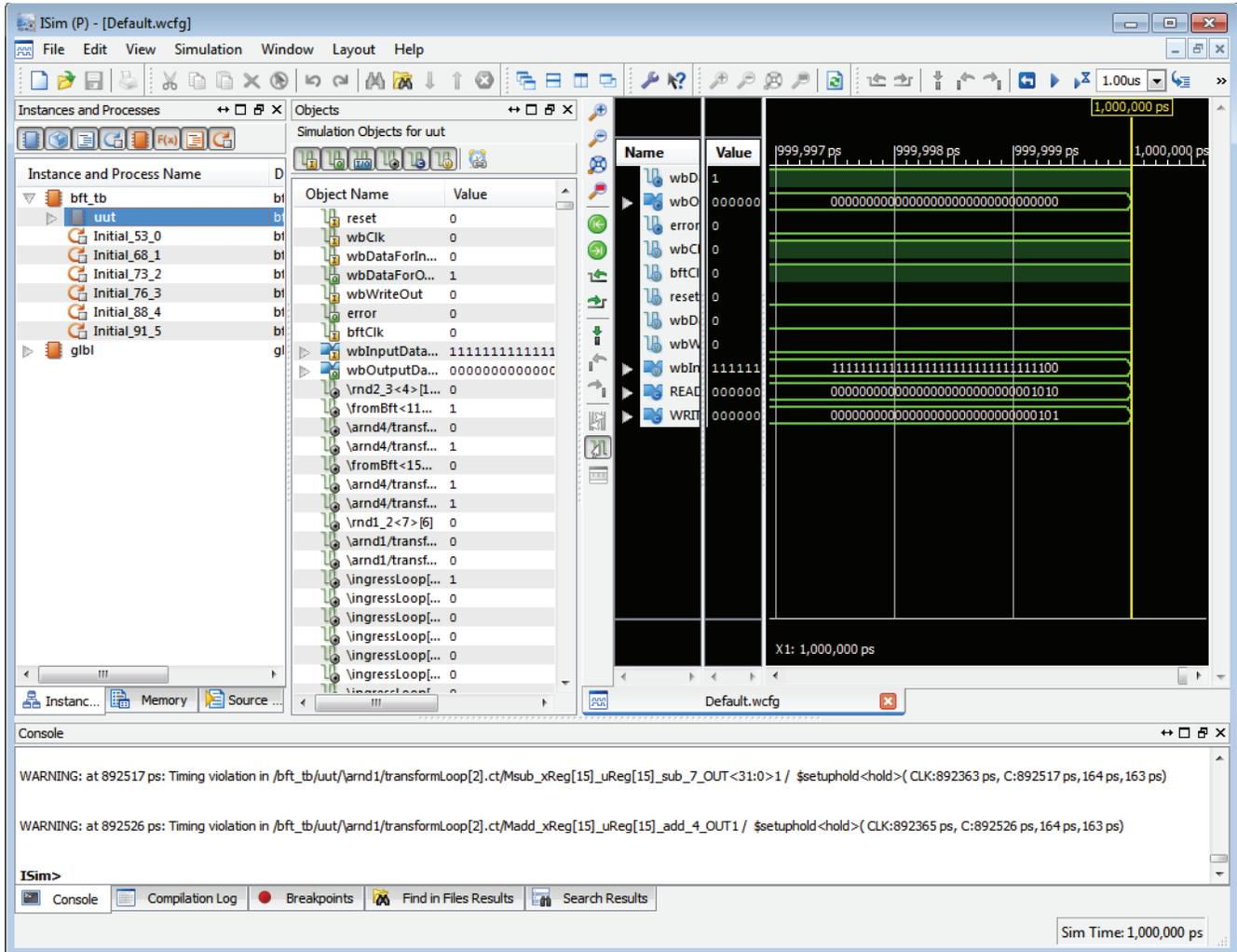
Figure 5-8: **ISim User Interface**

Running RTL DRCs

The following subsections describe selecting DRC rules and analyzing DRC violations in the PlanAhead tool.

Selecting DRC Rules

You can run Design Rule Checks (DRCs) on an Elaborated Design. The available rules focus on power reduction and performance improvement opportunities.

1. To run the DRC checks after elaborating the design, select one of the following commands:

    *   **Tools > Report DRC** from the main menu.

    *   From the RTL Analysis menu of the Flow Navigator, select **Report DRC**.

    The Run DRC dialog box opens, as shown in Figure 5-9, to enable rule selection.
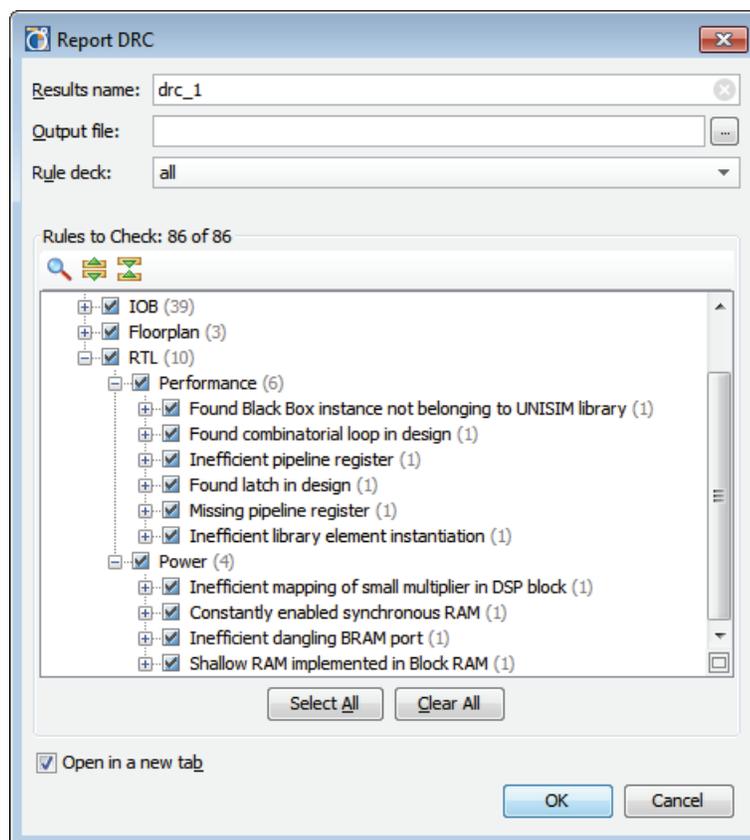
Figure 5-9:   **Run Design Rule Checker Dialog Box**

2. In the **Run DRC** dialog box, select the required rules, and click **OK**.

    For rule descriptions, see Appendix B, PlanAhead DRCs.

Analyzing DRC Violations

If violations are found, the DRC Results view opens, as shown in Figure 5-10, page 196. The DRC Results view displays the rule violations found, grouped under the various rule categories defined in the Run DRC dialog.

The rule violations are also categorized by severity. A violation can be informational only, to make you aware of a possible issue; can be a warning to suggest an issue that might need some resolution; can be an error to highlight issues that prevent proper implementation of your design.

The DRC Results view also color codes violations for quick review of errors, warnings, and informational messages.

- Errors have a red marker.

- Warnings have an orange marker.

- Informational messages have a yellow marker.

You can toggle the **Hide warnings and informational messages** button in the toolbar menu to turn off warnings and informational messages and see only the errors reported.

You can also click the header of the **Severity** column of the DRC Results view to sort violations by severity.

- Click *once* on the column header to sort in an increasing order.

- Click *twice* to sort in a decreasing order.

See Using Tree Table Style Views in Chapter 4 for more information.
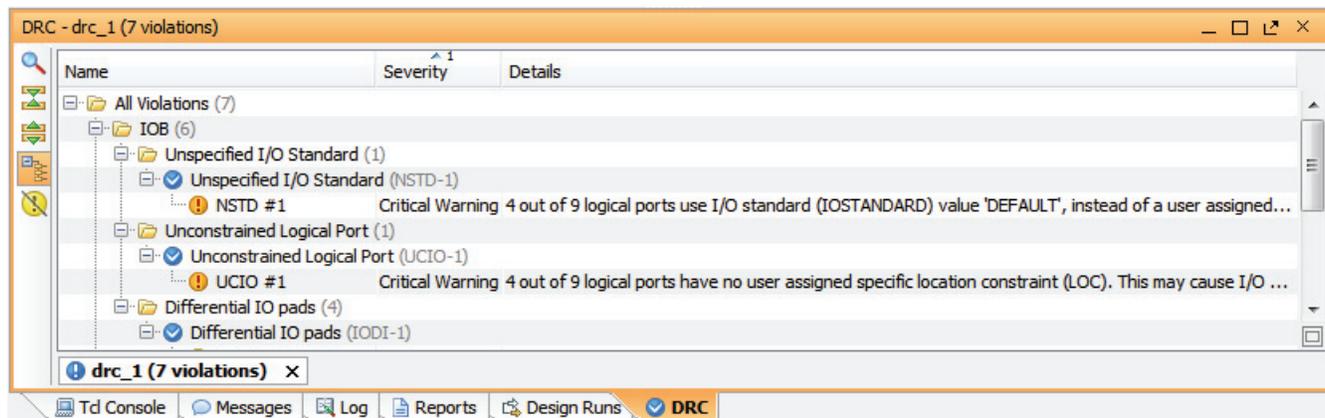
Figure 5-10: **Elaborated Design with Objects that Violate DRCs**

When you select a violation message in the DRC Results view, you can use the **Violation Properties** command in the popup menu to open the Violation Properties view. This view lets you see both a general review of the DRC rule violation, and specific details of the design elements that violate the rule.

The **Details** tab of the Violations Properties can have links to specific design objects that violate the DRC. Click the links to view the design object in the RTL Netlist view, the Device view, the Schematic, or the source RTL file.

Synthesizing the Design

The PlanAhead™ application includes a synthesis and implementation environment that facilitates a push button flow with synthesis and implementation *runs*. The PlanAhead tool manages the run data automatically, allowing repeated run attempts with varying Register Transfer Level (RTL) source versions, synthesis or implementation options, or constraints.

Also, the PlanAhead tool allows multiple synthesis and implementation runs using different command options, target devices, timing constraints, and physical constraints. You can queue the synthesis and implementation runs to launch sequentially or simultaneously with multi-processor machines. The PlanAhead tool uses Xilinx® Synthesis Technology (XST) for synthesis runs.

You can create and save *strategies*, which are configurations of command options that are applied to runs for synthesis or implementation. See Defining Strategies for Synthesis and Implementation in Chapter 4.

You can monitor synthesis or implementation progress, view log reports, and cancel runs.

Synthesis Methodology

The following are suggestions for the logic synthesis methodology to use the PlanAhead tool optimally for design analysis, floorplanning and hierarchical design.

- Register the module outputs to limit the number of modules involved in a critical path.

  Note: Long paths in single large hierarchical blocks can make floorplanning a difficult task. Consider dividing large hierarchical blocks in the RTL.

- The PlanAhead tool lets you define block-level partitions in the Elaborated Design. You might want to partition the design at the RTL level to confine critical timing paths to individual modules. Defining partitions invokes the XST incremental flow which generates individual NGC files for each partition. See Chapter 13, Using Hierarchical Design Techniques for more information.

  Note: Critical paths that span large numbers of hierarchical modules can be difficult to floorplan or partition.

- If you expect to change the design often, consider an incremental approach to synthesis. Most synthesis tools enable a top-down approach for incremental synthesis and implementation. This capability, coupled with Xilinx partitions, can preserve unchanged modules of the implemented design. Refer to Chapter 13, Using Hierarchical Design Techniques for more information.

  Design Preservation helps create an incremental flow but can hurt performance because global optimizations across the hierarchy are disabled. Consider this trade-off before you embark on an incremental methodology.

- Constrain the synthesis engine to rebuild or to otherwise preserve the hierarchy in the synthesized netlist. Flattened netlists may be optimal from a synthesis perspective, but they make it difficult to floorplan and constrain placement reliably.

  The `-netlist_hierarchy=rebuilt` option instructs XST to flatten the netlist to perform logic optimization but then to rebuild the netlist hierarchy in the Synthesized Design, based upon logic names. This facilitates design analysis and floorplanning within the PlanAhead tool. All synthesis strategies in the PlanAhead tool, except the XST default strategy, are defined using the `-netlist_hierarchy=rebuilt` option.

  If you encounter problems with this option in XST, set the `-netlist_hierarchy` option to **as_optimized** in the Synthesis Settings dialog box, as shown in Figure 6-5, page 203.

For more information about optimizing synthesis results, see the following documents cited in Appendix E, Additional Resources:

- *Xilinx Synthesis and Simulation Design Guide (UG626)*
- *XST User Guide for Virtex-4, Virtex-5, Spartan-3, and Newer CPLD Devices (UG627)*
- *XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices (UG687)*

Controlling File Compilation Order

RTL source files are compiled by the synthesis engine in the order they are displayed in the Compile Order tab of the Sources view. The file at the top of the list is compiled first and the file at the bottom of the list is compiled last. A specific compile order is necessary in cases where declarations are made in RTL source files that must be compiled before those entities can be used in other source files.

The PlanAhead tool automatically identifies and sets the best top module candidate. The compile order is also automatically managed, as the top module file and all sources that are under the active hierarchy are passed to synthesis and simulation in the correct order.

The Hierarchy Update command in the Sources view controls how the PlanAhead tool handles changes to the top module, and to changes to the source files in the design. The default setting, **Automatic Update and Compile Order**, specifies that the hierarchy view of the design and the compilation order should be automatically updated as source files are changed.

To modify the compile order before synthesis:

1. Set **Hierarchy Update > Automatic Update, Manual Compile Order** to allow the PlanAhead tool to automatically determine the best top module for the design, but allow you to manually specify the compilation order.

2. Drag and drop files in the Compile Order tab of the Sources view to arrange the compilation order, or

3. Use the **Move Up** or **Move Down** commands in the Sources view popup menu.

See Using the Sources View in Chapter 4 for more information.

Defining Global Include Files

In the case of Verilog and Verilog header files, you can also define the files as Global Include files. The PlanAhead tool supports designating one of more Verilog or Verilog Header source files as Global Include files. Files that are marked as global include, are processed first before any other sources for the following operations: specify top module, scan for include files, auto re-order source, RTL elaboration, synthesis and simulation.

Because the global include files are always processed before any other files in the design, this lets you more easily specify common Verilog header files and other sources in the design.

Verilog typically requires an `` `include `` statement to be placed at the top of any Verilog source file that references content from a another Verilog file or header file. Designs that use common header files might require multiple `` `include `` statements to be repeated across multiple Verilog sources used in the design. Marking the header files as global include files can eliminate the need for these repeated `` `include `` statements since the global include files are processed before any other source files.

To designate a Verilog or Verilog header file as a global include file:

- Select the file in the Sources view, and use the right mouse popup menu **Set Global Include** command, or

- Use the **Global Include** checkbox in the Source File Properties view, as shown in Figure 6-1.
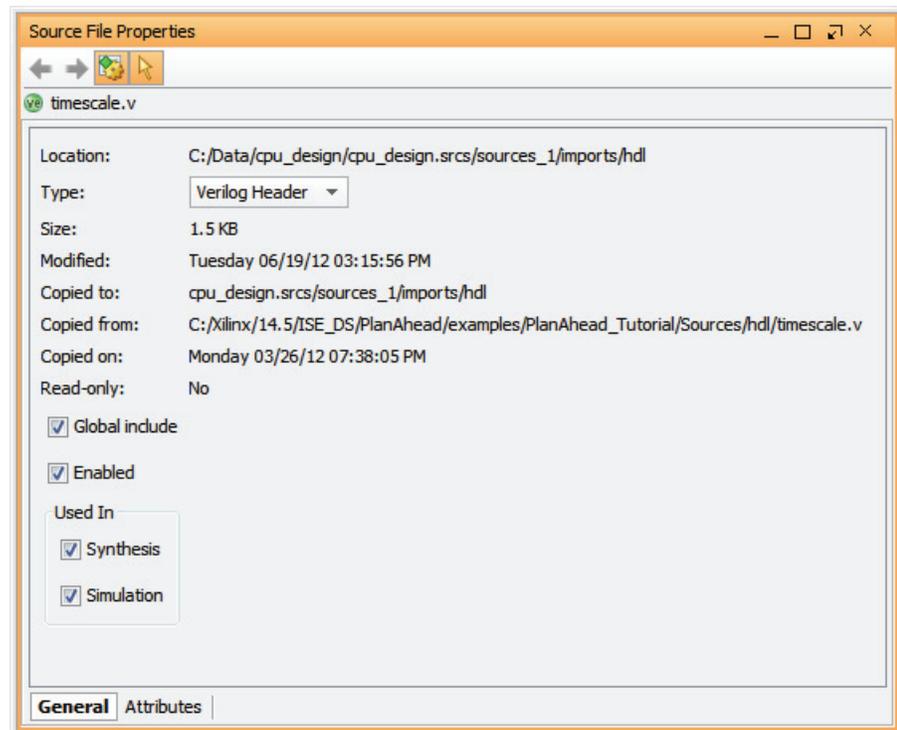
Figure 6-1: **Global Include**

See Using the Sources View and Viewing Source File Properties in Chapter 4 for more information.

When a file is designated as a global include, it displays with a different icon and is placed in a Global Include folder in the Sources view. Multiple files can be designated as global include files and are evaluated in the order listed in the Sources view. In this case, all global include files are processed before the other source files in the project.

Because the content of all global include files is visible to all the other source files in the project, any Verilog header files that should be specifically applied to a single Verilog source (for example a particular `` `define `` macro), should be referenced by an `` `include `` statement rather than marked as a global include file.

Running Synthesis

To synthesize the design for the Xilinx targeted part, you must run the RTL source files and the design constraints through XST using the following process:

- Defining Synthesis Runs
- Setting Synthesis Options
- Launching a Synthesis Run

Defining Synthesis Runs

A synthesis run defines and configures aspects of the design which are used during synthesis. A synthesis run defines the Xilinx device to target during synthesis, the constraint set to apply, and command line options to control the results of the synthesis engine.

A synthesis run can be defined by:

- Select **Flow > Create Runs** from the main menu.
- In the Flow Navigator, select **Create Synthesis Runs** from the Synthesis popup menu.

1. The **Create New Runs** wizard opens. The first page of the wizard is a summary of the command.

2. Click **Next** to proceed.

   The **Configure Synthesis Runs** dialog box opens, as shown in Figure 6-2. A new synthesis run is available by default, but additional runs can be added with the **More** button.
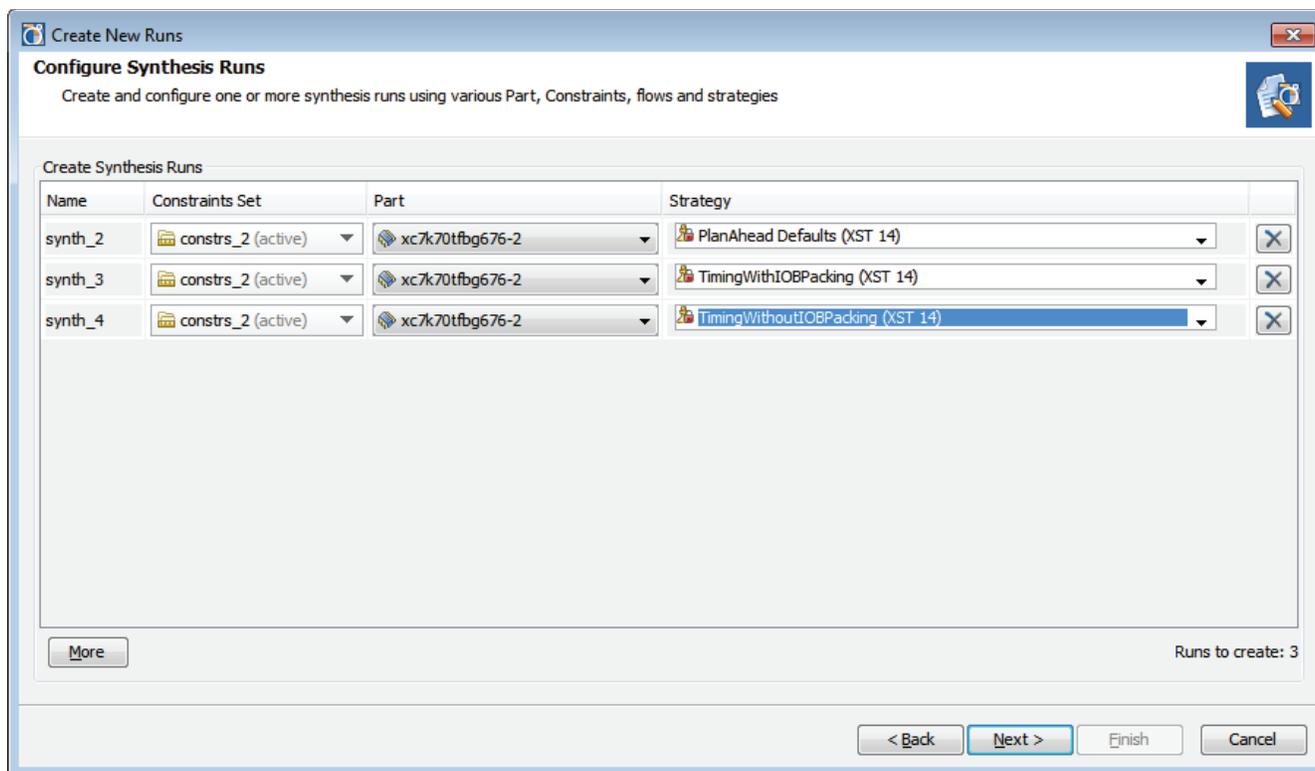
Figure 6-2: **Create New Runs: Set Up Synthesis Runs**

3. Enter a **Name** for the run, or accept the default name.

4. Select a **Constraint Set**, a target **Part**, and a **Strategy** for the run.

   The default values are defined by the synthesis or implementation Project Settings at the time the **Create New Runs** command is run. See Configuring Project Settings in Chapter 3.

   The strategies are a defined set of XST run-time options controlling the synthesis results. See Defining Strategies for Synthesis and Implementation in Chapter 4 for more information.

5. Click the **More** button to define additional runs.

6. Click **Next**.

   The Launch Options dialog box opens as shown in Figure 6-3, page 201.
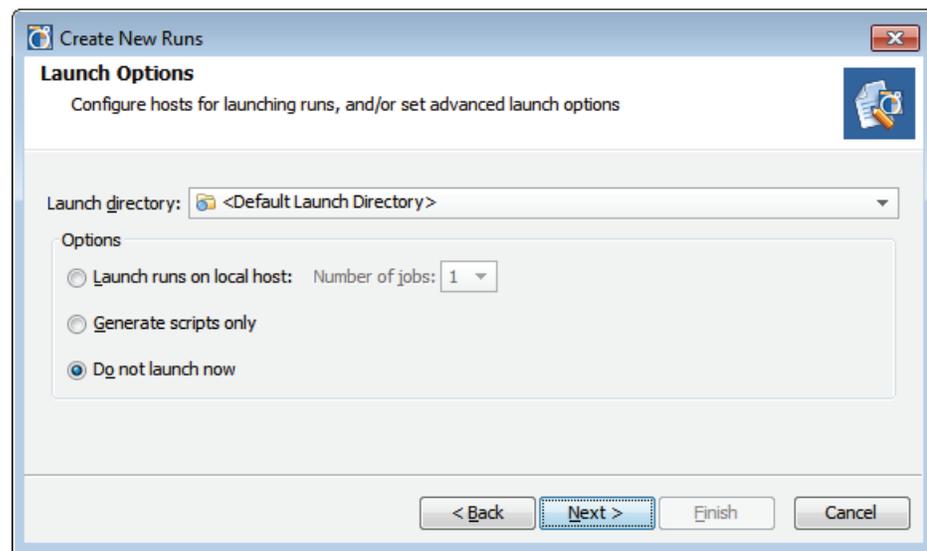
Figure 6-3: **Synthesis Launch Options**

7. Specify the **Launch Directory** to create and store the synthesis run data.

   The default launch directory is contained within `<project>.runs` directory structure.

   Note: Defining any non-default location outside of the project directory structure makes the project non-portable because absolute paths are written into the project files.

8. Specify the **Options**:

   • **Launch Runs on Local Host**: Launch the run on the local machine processor.

      - **Number of Jobs**: Define the number of local processors to use for runs. This option is used only when you are launching multiple runs simultaneously. Individual runs are launched on each processor. No multi-threaded processors are used with this option.

   • **Launch Runs on Remote Hosts** (Linux only): Use remote hosts to launch one or more jobs. See Launching Runs on Remote Linux Hosts in Chapter 9.

      - **Configure Hosts**: Select this option to configure remote hosts.

   • **Generate scripts only**: Export and create the run directory and run script, but do not launch the run at this time. The script can be run at a later time outside of the PlanAhead tool.

   • **Do not launch now** - Save the new runs, but do not launch or create run scripts at this time.

9. Click **Next** and review the **Create New Runs Summary**.

10. Click **Finish** to create the defined runs and execute the specified launch options.

New runs are added to the Design Runs view.

Using the Design Runs View

The Design Runs view displays all of the synthesis and implementation runs created in a project, and provides commands to configure, manage, and launch the runs.

Select **Window > Design Runs** to open the Design Runs view if it is not already displayed. Figure 6-4 shows the Design Runs view.

A synthesis run can have multiple implementation runs. You can expand and collapse synthesis runs using the tree widgets in the view. The Design Runs view is a tree table view. Refer to Using Tree Table Style Views, page 113, for more information on working with the columns to sort the data in this view.

Figure 6-4: **Design Runs View**

The Design Runs view reports the run status, including when the run has not been started, is in progress, is complete, or is out-of-date. Runs can become out-of-date when source files, constraints or project settings are modified. You can reset and delete stale run data in the Design Runs view.

Setting the Active Run

Only one synthesis run and one implementation run can be "active" in the PlanAhead tool at any time. The Compilation and Messages views, Status Bar, and Project Summary display the information for the active run. The Project Summary view only displays compilation, resource, and summary information for the active run.

When there are multiple synthesis or implementation runs in the Design Runs view, to make a specific run active, select the run and use the **Make Active** command from the popup menu.

Setting Synthesis Options

You can change many aspects of a synthesis run prior to launching the run.

When you select a run in the Design Runs view, the Run Properties view displays the current configuration of the selected run. In the Run Properties view you can change the name of the run, the Xilinx part targeted by the run, the run description, and the constraints file that drives the synthesis and is the repository of new constraints for the design. See Using the Run Properties View in Chapter 4 for more information.

You can also change the run-time options used by XST during synthesis by using the **Change Run Settings** command from the Design Runs view popup menu, to open the Design Run Settings dialog box, as shown in Figure 6-5.
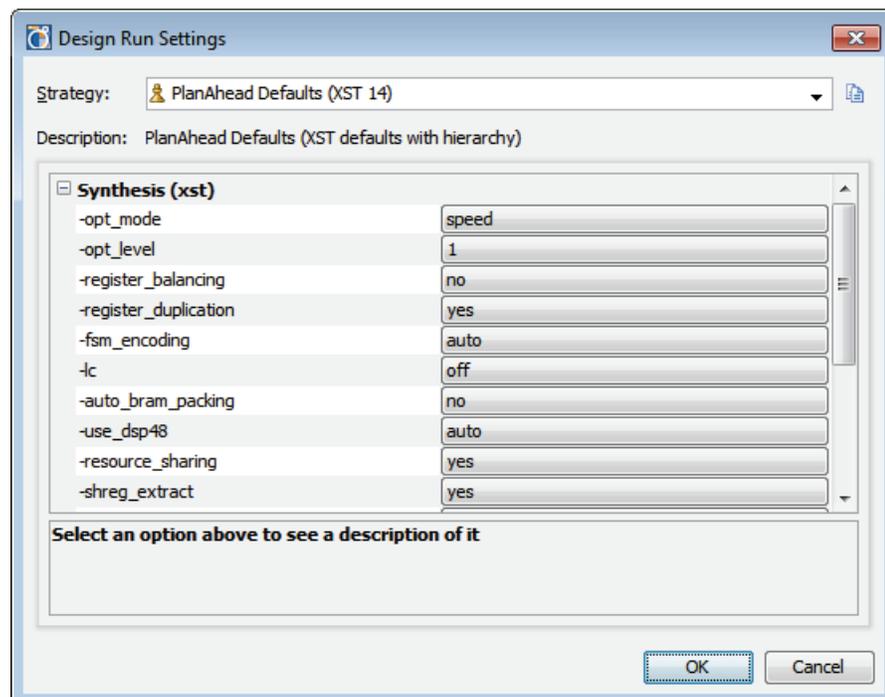
Figure 6-5: **Change Synthesis Settings**

The Design Run Settings dialog box lets you specify:

- **Strategy**: The general strategy to be used by the run.

- **XST options**: Change any XST command-line option to refine the general run strategy to meet the specific needs of the current design.

  A brief description of each option is displayed in the lower dialog box pane. An asterisk (*) next to an option name indicates that the value is changed from the default used by the strategy.

  For more information about specific XST options, see the following documents cited in Appendix E, Additional Resources:

  - *XST User Guide for Virtex-4, Virtex-5, Spartan-3, and Newer CPLD Devices (UG627)*

  - *XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices (UG687)*

- **Save Strategy As**: Save the changes to the strategy as a new strategy for use in other projects.

Launching a Synthesis Run

To launch Synthesis, you can:

- Click the **Run Synthesis** button on the Flow Navigator.

- Use the **Flow > Run Synthesis** command from the main menu.

- Use the **Run Synthesis** command from the toolbar menu.

  Note: The preceding commands will launch the active run in the Design Runs view. See Setting the Active Run, page 202.

You can also launch a run other than the active run, or launch multiple runs at the same time. Select one or more runs in the Design Runs view. Use **Shift+click** or **Ctrl+click** to select multiple runs. Use the **Launch Runs** command from the popup menu, or from the Design Runs view toolbar menu, to open the Launch Selected Runs dialog box as shown in Figure 6-6.

Note: You can choose both synthesis and implementation runs when selecting multiple runs in the Design Runs view. The PlanAhead tool will manage run dependencies, and launch runs in the correct order.
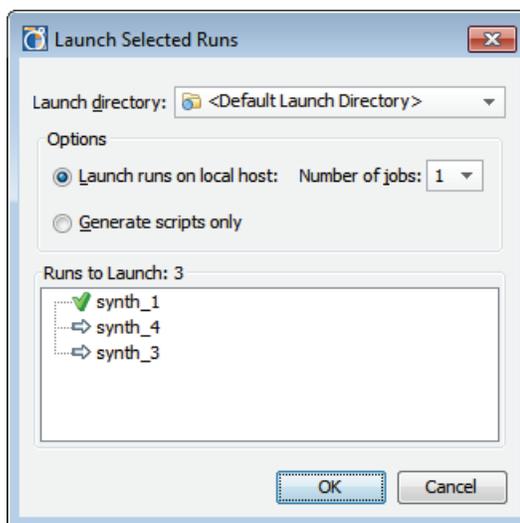
Figure 6-6: **Launch Selected Synthesis Runs**

- **Launch Directory**: The default launch directory is contained within the local project directory structure, stored at: `<project_name>.runs/<run_name>.`

- **Options**:

  - **Launch Runs on Local Host**: Launch the run on the local machine processor.

    - **Number of Jobs**: Define the number of local processors to use for runs. This option is used only when you are launching multiple runs simultaneously. Individual runs are launched on each processor. No multi-threaded processors are used with this option.

  - **Launch Runs on Remote Hosts** (Linux only): Use remote hosts to launch one or more jobs. See Launching Runs on Remote Linux Hosts in Chapter 9.

    - **Configure Hosts**: Select this option to configure remote hosts.

  - **Generate scripts only**: Export and create the run directory and run script, but do not launch the run at this time. The script can be run at a later time outside of the PlanAhead tool.

The PlanAhead tool processes the run, and launches synthesis, depending on the status of the run. The status is displayed in the Design Runs view, as shown in Figure 6-4, page 202.

- If the status of the run is *Not Started*, the run will begin immediately.
- If the run is in an *Error* state, the PlanAhead tool will first reset the run to remove any incomplete run data, and then will restart the run.
- If the run is *Complete* or *Out-of-Date*, the PlanAhead tool will prompt you to confirm that the run should be reset prior to proceeding with the run.

Moving Processes to the Background

As the PlanAhead tool spawns the process to run synthesis or implementation, reading design files and constraint files, the Starting Run dialog box lets you put the process into the background, as shown in Figure 6-7.

When you put a process into the background, it releases the PlanAhead tool to perform certain other functions, like viewing reports, or opening design files, while it completes the background task. You can make use of this time to review previous runs for instance, or examine reports. However, the Tcl Console is blocked, and you will not be able to use Tcl commands, or perform tasks that require Tcl commands, such as switching to another open design.
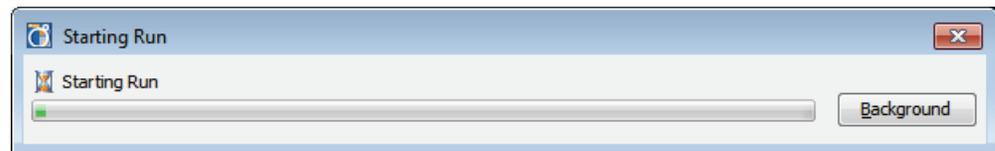
Figure 6-7: **Start Run - Background Process**

Monitoring the Synthesis Run

You can monitor the status of a Synthesis run by reading the compilation log in the Log view, or viewing the synthesis information, warnings, and errors in the Messages view, or viewing the run in the Design Runs view. See Monitoring the Implementation Run, page 294 for more information.

Selecting any message in the Messages view opens the associated RTL source file and cross-selects the appropriate line of code. Figure 6-8 shows this interaction between the Messages view and the Text Editor. This lets you find and resolve issues that occur during Synthesis.
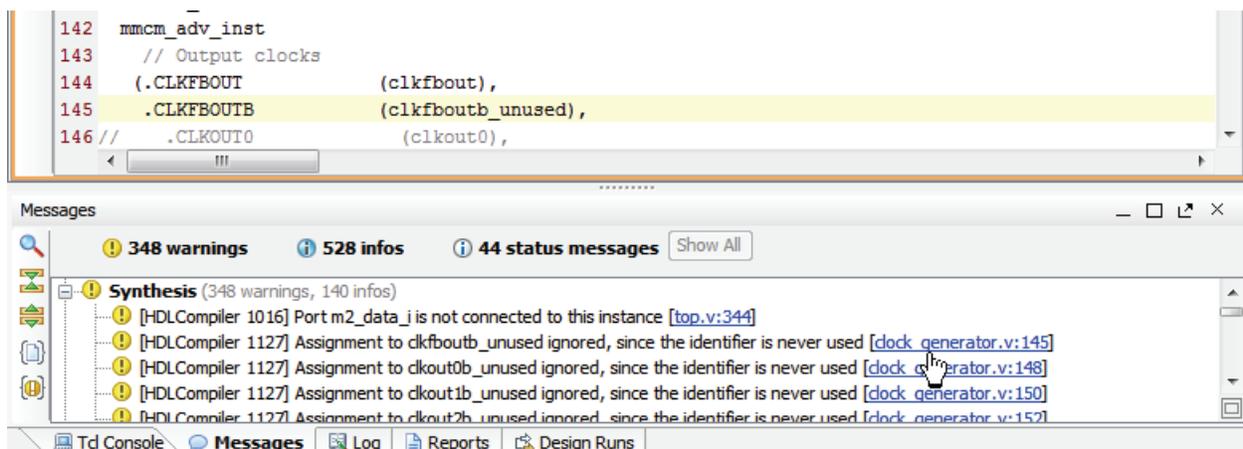
Figure 6-8: **Synthesis Message and RTL Source File**

Following Synthesis

After the run is complete, the Synthesis Completed dialog box opens as shown in Figure 6-9.
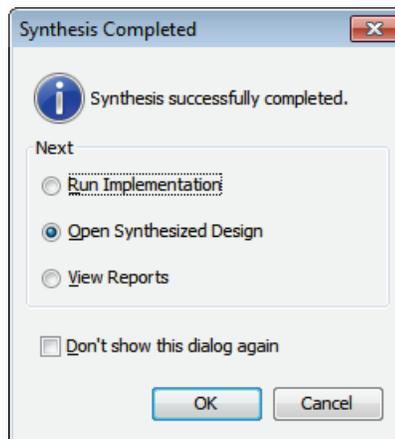
Figure 6-9: **Synthesis Completed Dialog Box**

1. In the Synthesis Completed dialog box, select the option that matches how you want to proceed.

   The options are:

   • **Run Implementation**: Launches Implementation with the current Implementation Project Settings. For more information on the Implementation process see Chapter 9, Implementing the Design.

   • **Open Synthesized Design**: Imports the netlist, active constraint set, and target part into the Design Analysis and Floorplanning environment so you can perform I/O pin planning, design analysis, and floorplanning. See Chapter 7, Synthesized Design Constraints and Analysis for more information.

   • **View Reports**: Opens the Reports view so you can select and view the XST Report file. For more information, see Viewing Report Files, page 297.

2. Click **OK** or **Cancel**.

Analyzing Synthesis Results

After Synthesis completes, you can view the reports, and open, analyze, and use the Synthesized Design to apply constraints to the design before Implementation. The Reports view contains a list of reports provided by the various synthesis and implementation tools in the PlanAhead tool. Open the Reports view and select a report for a specific run to see details of the run. Figure 6-10, page 207 shows an example report from a Synthesis run.

Figure 6-10: **XST Synthesis Report**

The Table of Contents for this report shows the type of information that can be found in the synthesis report. For more information, see:

- Using the Project Summary view in Chapter 3
- Using the Synthesized Design Environment in Chapter 7
- Analyzing Implementation Run Results in Chapter 9

Synthesized Design Constraints and Analysis

The design analysis and constraint definition features available in the PlanAhead™ tool are typically performed by opening a Synthesized Design before running Implementation. However, many of the analysis and constraints features described in this chapter are available for implemented designs as well.

In the Synthesized Design environment, you can:

- Perform I/O Planning

- Analyze aspects of the design

- Validate resource and timing estimates

- Run Design Rule Checks (DRCs), listed in Appendix B, PlanAhead DRCs

- Define physical and timing constraints for the Xilinx® ISE® Design Suite

The PlanAhead tool design tasks performed in the Synthesized Design environment are:

- Inserting ChipScope™ Pro Analyzer debug cores (see Debugging the Design with ChipScope, page 366.)

- Defining partitions for Design Preservation and Partial Reconfiguration on netlist-based projects. (see Using Hierarchical Design Techniques in Chapter 13.)

Using the Synthesized Design Environment

The PlanAhead tool provides an environment to analyze the design from several different perspectives, and to apply constraints to the design prior to implementation.

When you open a Synthesized Design, the software loads the synthesized netlist, the active constraint set, and the target device. See Using the Netlist View, page 156 for more information.

Create or open a Synthesized Design using one of the following:

- **Open Synthesized Design** command in the Synthesis menu of the Flow Navigator.

- **Flow > Open Synthesized Design** command from the main menu.

- **New Synthesized Design** from the popup menu of the Synthesis command of the Flow Navigator.

- **Flow > New Synthesized Design** command from the main menu. See Opening a Synthesized Design, page 28.

With a Synthesized Design open, the PlanAhead tool opens the Floorplanning view layout for you to examine design logic and hierarchy, view the resource utilization and timing estimates, run DRCs, and apply timing and physical constraints.

The default views presented in the Floorplanning view layout include: the Project Summary, Device, Schematic, Netlist, Sources, I/O Ports, and Physical Constraints views, as shown in Figure 7-1.
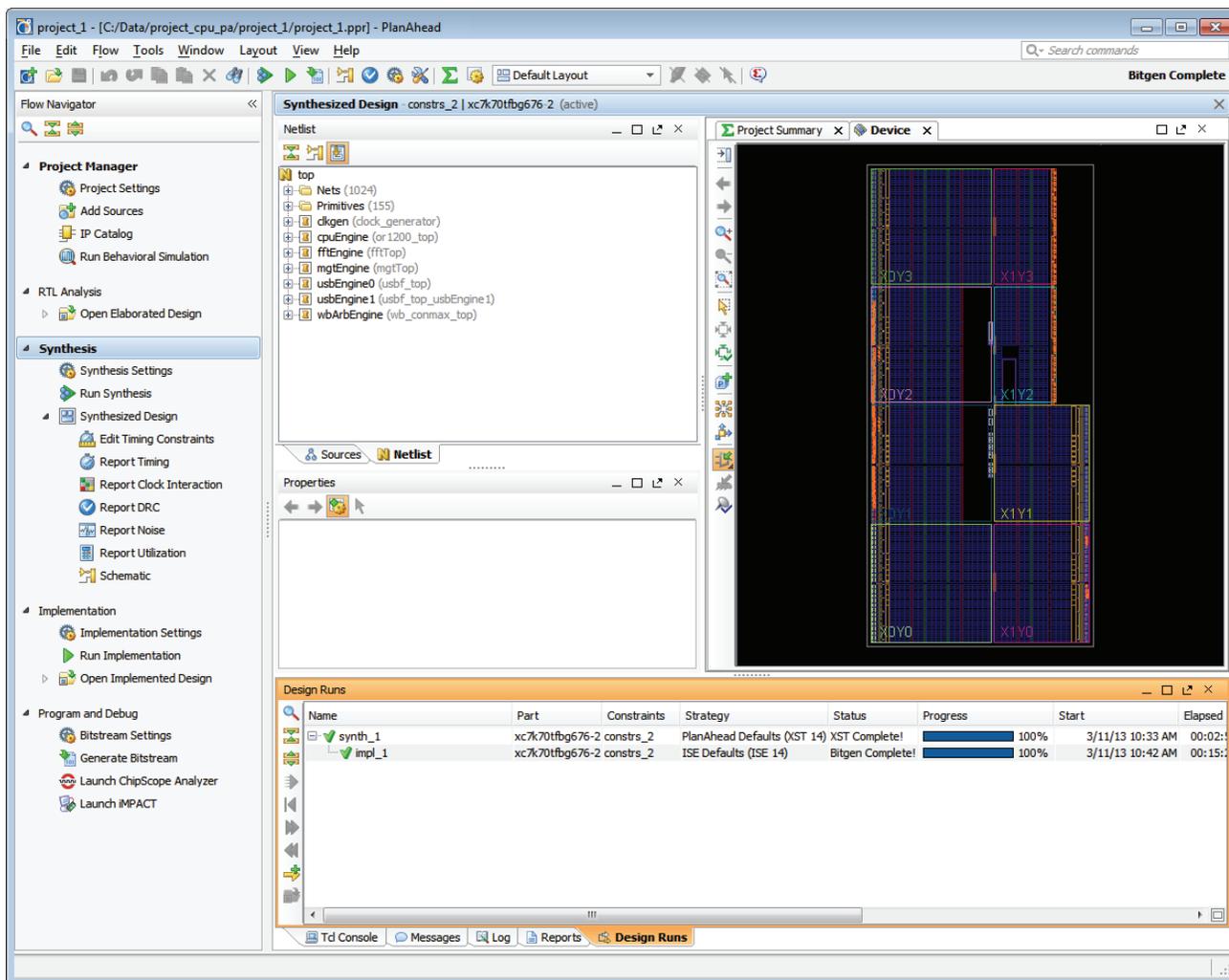
Figure 7-1:   **Synthesized Design Floorplanning View Layout**

The PlanAhead tool provides default configurations of many of the design views and presents them as *View Layouts.* You can open other views, if needed, including the Package view. For more information about using specific views, see Using Common Views, page 128.

You can save the arrangement of any opened views under a user-defined view layout so you can restore frequently-used view configurations. See Using View Layouts in Chapter 4 for more information.

Viewing and Reporting Resource Statistics

In an opened Synthesized Design statistical information about the design logical content and device utilization is available in the Project Summary view. This includes a:

- Resources pane with resource utilization estimates for the elaborated RTL and synthesized netlists, and

- Compilation pane with summary information from synthesis and implementation reports.

You can select any netlist instance or Pblock and examine the resource statistics in the Instance or Pblock Properties view, including selecting the top design. The information includes: logic object type counts, percentage of device resources utilized, carry chain information, and clock reports. You can export the information into an Excel spreadsheet.

Generating Hierarchical Resource Estimates

You can display resource estimates graphically as an expandable hierarchical tree with the Report Utilization command, as shown in Figure 7-2. As each Resources type displays, you can expand it to view each level of logic hierarchy.

1. Open a Synthesized Design.
2. Click either:
   - **Flow Navigator > Report Utilization**
   - **Tools > Report Utilization**
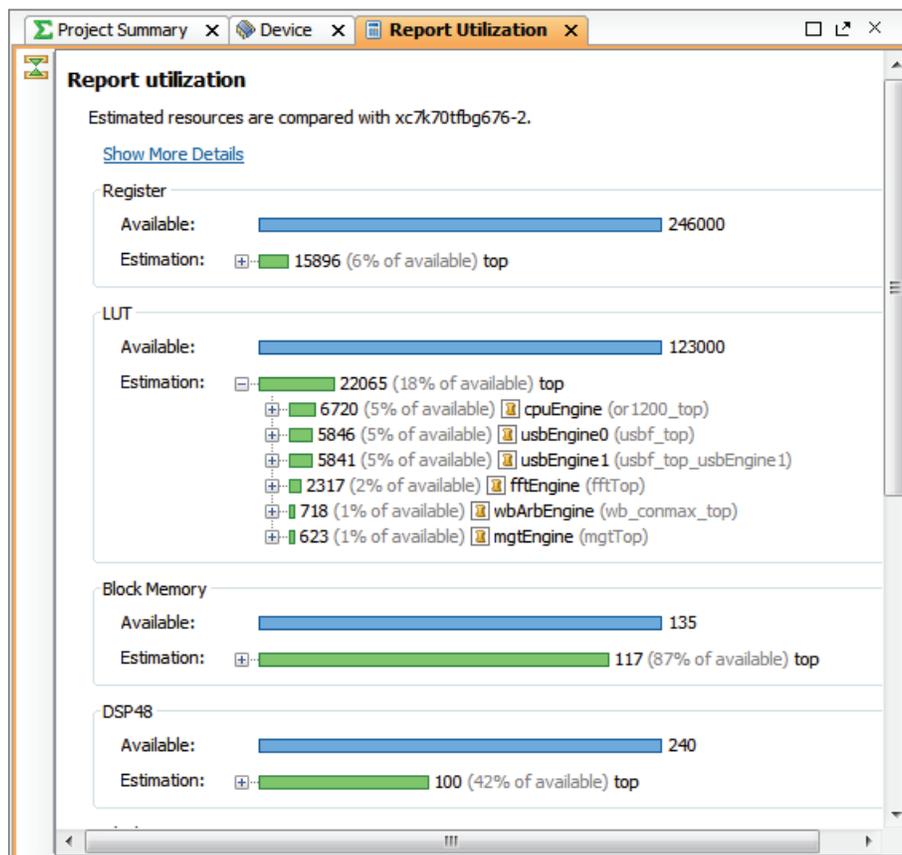
Figure 7-2:   **Resource Utilization Report**

Viewing Resource Statistics for Logic Instances

The PlanAhead tool provides estimates of the number of device resources contained in the design. You can display resource statistics for any logic instance, including the top-level, in the Instance Properties view.

To display design resource statistics, select a top module or any instance module in the Netlist view. Figure 7-3 shows the Netlist view with a top module selected.
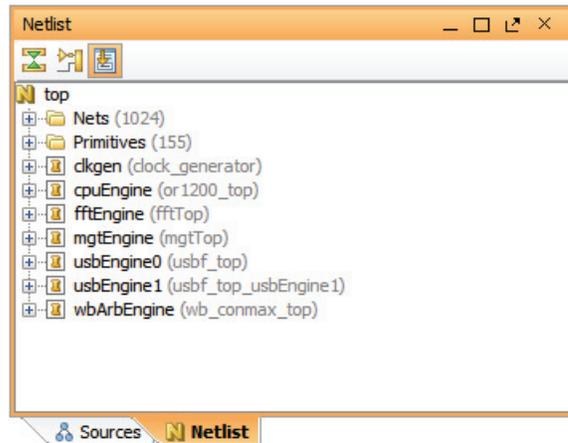
Figure 7-3: **Netlist View with Top Module Selected**

The Netlist or Instance properties open in the Properties view.

If the Netlist or Instance Properties do not display, right-click on the module, and select **Netlist Properties** or **Instance Properties** from the popup menu.

The Netlist Properties view contains five tabs. The Instance Properties dialog box contains seven tabs. In the Netlist or Instance Properties view, click the **Statistics** tab.

The Statistics tab displays valuable design information including: Primitive Instance Counts, Interface Signal Counts, Clock Names and Clocked Instance Count, Carry Chain Count, and Max Length.

Figure 7-4, page 213 shows an example of the Netlist Resource Statistics.
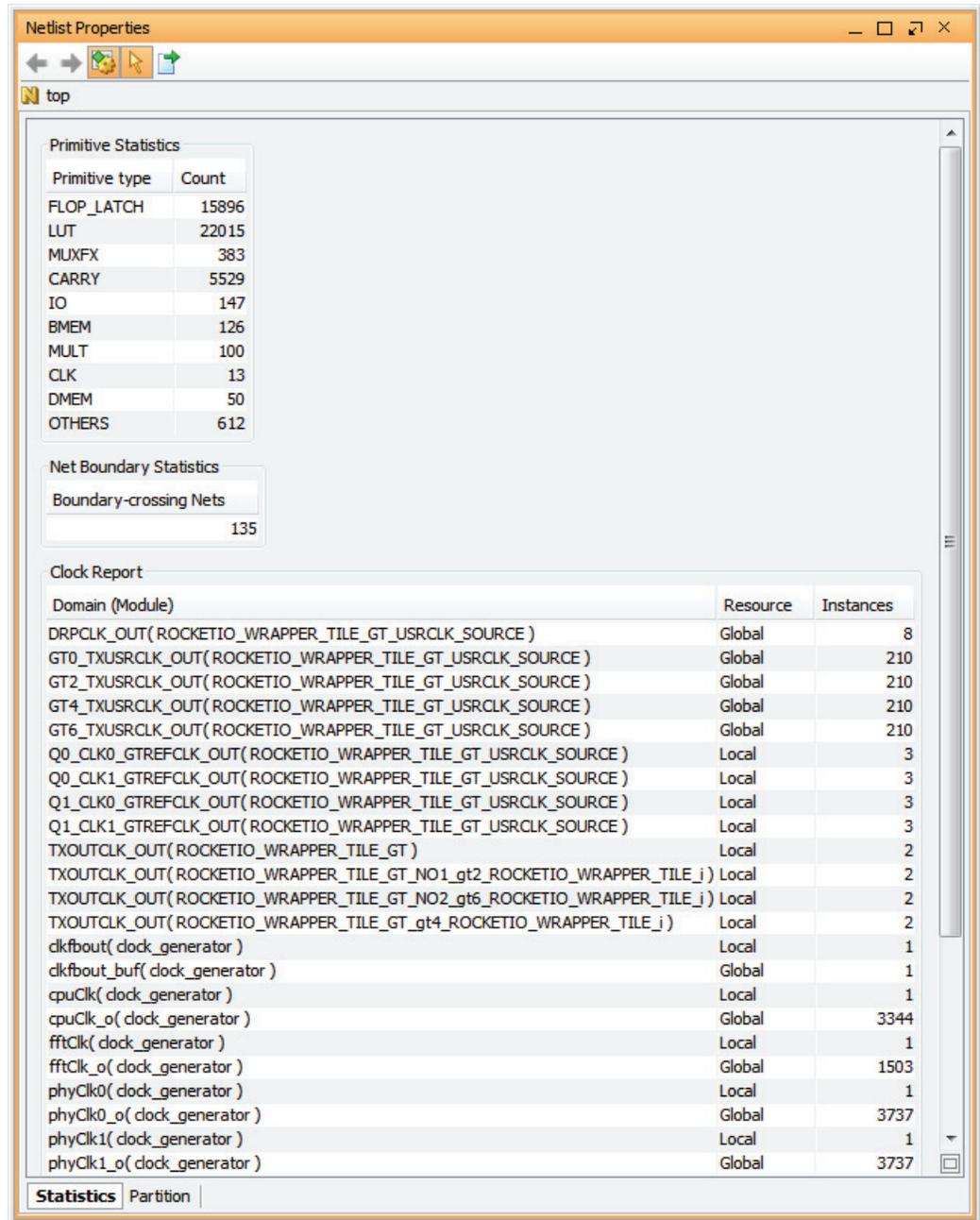
Figure 7-4: **Netlist Resource Statistics**

Viewing Resource Statistics for Pblocks

The PlanAhead tool provides logic utilization statistics for Pblocks that can help determine if enough device resources are contained in the Pblock area to satisfy the assigned logic. Also, the ROOT Pblock is considered the top of the design and can provide utilization statistics for the entire design.

To display utilization statistics for a Pblock:

1. Select either the ROOT Pblock or any Pblock in the Physical Constraints view. Figure 7-5, page 214 shows a selected ROOT Pblock.
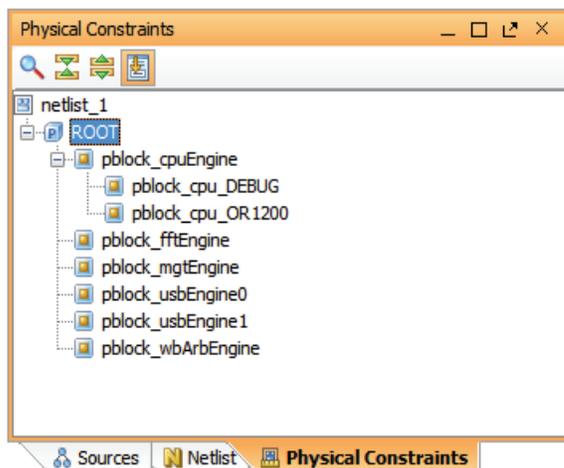
Figure 7-5: **Physical Constraints View with ROOT Selected**

The Pblock Properties display in the Properties view.

2. If the Pblock Properties do not display, right-click ROOT or Pblock, and select **Pblock Properties** from the popup menu.

For more information, refer to Viewing Pblock Properties in Chapter 10.

Using the Statistics Tab

The Statistics tab displays design information that includes: overall device utilization for the various device resources; carry chain count and max length, RPM count, maximum sizes; clock names and clocked instance count; I/O utilization; signal and primitive instance counts. Figure 7-6 provides an example of the Statistics tab for a Pblock.
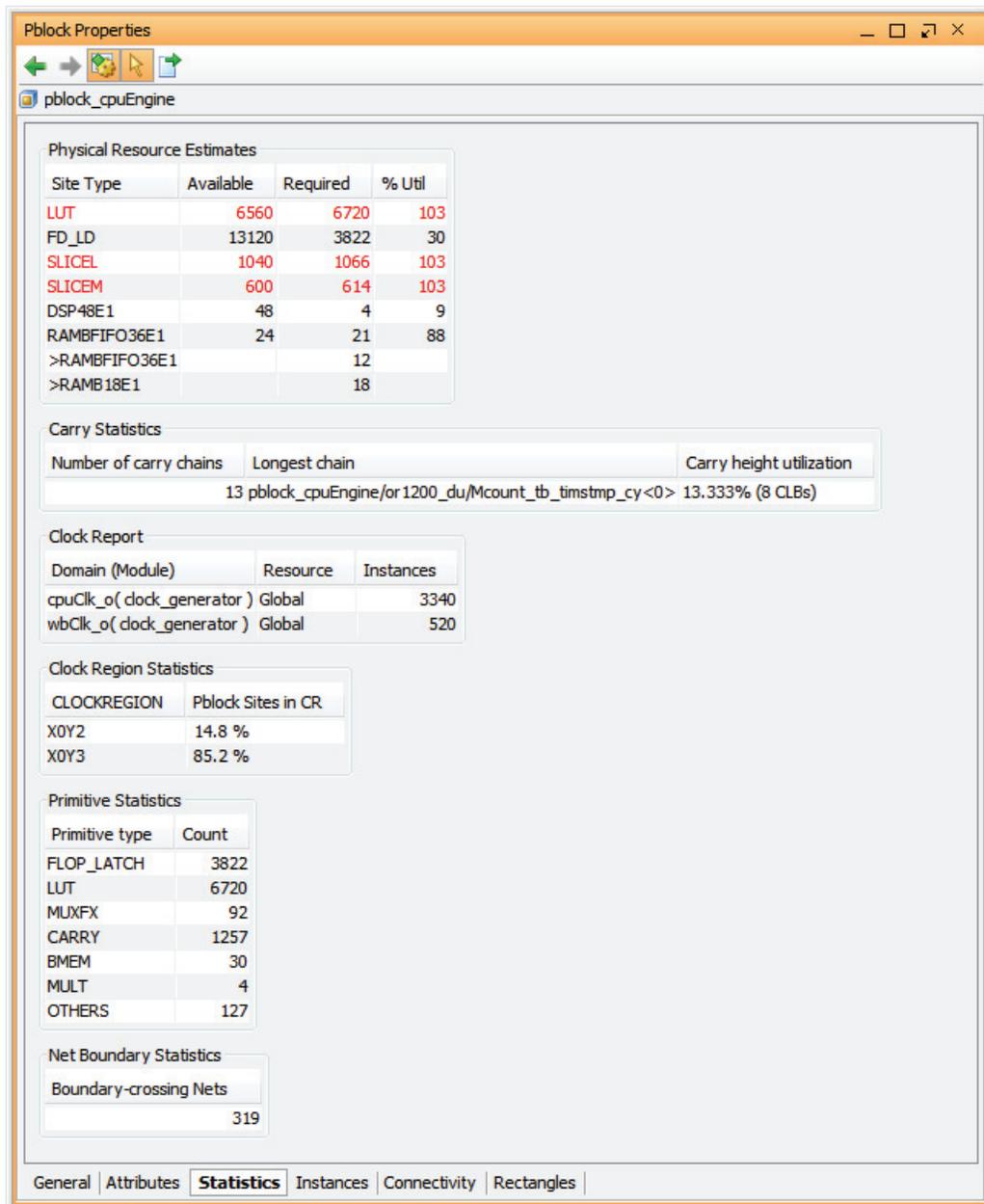
Figure 7-6:   **Pblock Properties: Statistics Tab**

Exporting Resource Statistics Reports

You can save displayed resource statistic data to a spreadsheet file. The PlanAhead tool generates a hierarchical-style report, in which you can define how many levels of hierarchy to report with estimates listed for each module at each level.

To export a resource statistics report:

1. Select the **Export Statistics** button, to export the data to a spreadsheet file.

   The Export Netlist Statistics dialog box opens, as shown in Figure 7-7.
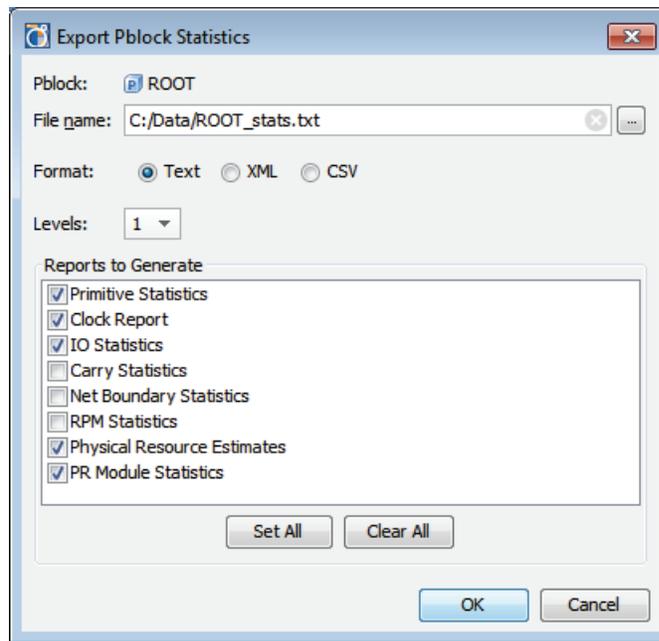
Figure 7-7: **Export Netlist Resource Statistics**

The Export Netlist Statistics dialog box options are:

- **File Name**: Enter the name and location of the spreadsheet file to be created.

- **Format**: Select the format of the output file: Text, XML, or CSV.

- **Levels**: Indicate the number of levels of hierarchy to traverse and include in the report as separated modules.

- **Reports to Generate**: Define the types of netlist statistics to include in the output report file.

2. Select the options for the exported file, and click **OK**.

Exploring the Logic

The PlanAhead tool provides several perspectives in which to analyze design logic:

- The Netlist and Hierarchy views contain a navigable hierarchical tree-style view.

- The Schematic view allows selective logic expansion and hierarchical display.

- The Device view provides a graphical view of the device, placed logic objects, and connectivity. All views cross-select and present the most useful information.

- The Implemented Design contains additional logic analysis capabilities. The capabilities are more powerful after placement and timing results are imported. Refer to Chapter 11, Analyzing Implementation Results, for more information.

The following subsections describe the available logic exploration methods.

Exploring the Logic Hierarchy

The Netlist view displays the logic hierarchy of the RTL. You can expand and select any logic instance or net within the netlist. As you select logic objects in other views, the Netlist view expands automatically to display the selected logic objects. For more information, refer to Using the Netlist View, page 156.

Information about instances or nets displays in the Instance or Net Properties views.

The Hierarchy view displays a graphical representation of the RTL logic hierarchy. Each module is sized in relative proportion to the others, so you can determine the size and location of any selected module. For more information, refer to Using the Hierarchy View, page 160.

Exploring the Logical Schematic

The Schematic view allows selective expansion and exploration of the logical design. You must select at least one logic object before the Schematic view is available. You can view and select any logic in the Schematic view.

You can display groups of timing paths to show all of the instances on the paths. This aids floorplanning because it helps you visualize where the timing critical modules are in the design.

To open the Schematic view:

1. Select one or more instances, nets, or timing paths.

2. Select **Schematic** from the view toolbar or the popup menu, or press the **F4** key.

   The view opens with the selected logic displayed.

3. You can then select and expand the logic for any pin, instance or hierarchical module.

For more information, see Using the Schematic View, page 146.

Analyzing Hierarchical Connectivity

Sometimes, it is helpful to create a top floorplan to help visualize the connectivity flow of the design, as shown in Figure 7-8, page 218.
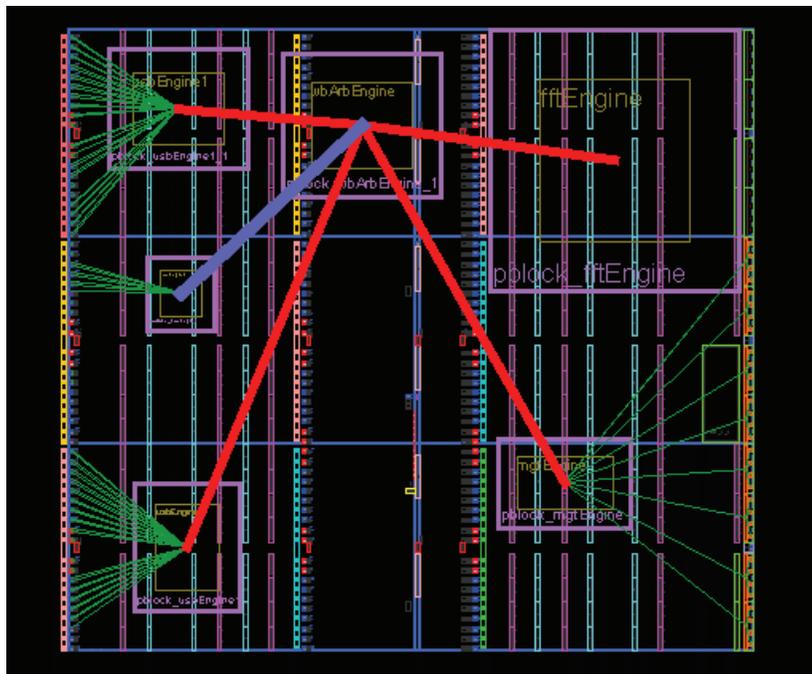
Figure 7-8: **Viewing Top-level Design Connectivity**

You can create a top-level floorplan by creating Pblocks for key levels of the design hierarchy. See Chapter 10, Floorplanning the Design for more information.

The Net bundles indicate the heaviest connectivity requirements between the modules. When you select a Net, bundle information about the net content displays in the Net Bundle Properties view.

The color and line thickness of the Net bundles can be configured depending on the number of signals they contain. The **Tools > Options > General** dialog box contains connectivity display options. One option is to display the Net Bundles as a **Mesh** or in a **Tree** pattern.

You can also traverse the hierarchy and create sub-modules for the larger top module to gain more detailed granularity.

This top-level floorplan can be an indicator of the I/O pinout configuration quality, and can help identify potential routing congestion issues.

Examining resource statistics and clock requirements for each module can aid in understanding potential placement issues.

For more information, see the *Floorplanning Methodology Guide (UG633)*, cited in Appendix E, Additional Resources.

Inserting ChipScope Debug Cores

The PlanAhead tool lets you insert and configure ChipScope Integrated Logic Analyzer (ILA) and Integrated ChipScope Onboard Netlist (ICON) debug cores into the Synthesized Design. You can select debug nets and configure cores using the Set Up ChipScope wizard.

The added ChipScope cores are preserved though through netlist iterations. The ChipScope netlist overlay reconnects with the selected debug nets after you add a new netlist and open it in a project. The PlanAhead tool sends a warning if it finds any discrepancies. For more information on inserting debug logic and debugging with ChipScope, Debugging the Design with ChipScope in Chapter 12.

Defining Timing Constraints

PlanAhead provides the ability to define and modify timing constraints for the design; however, you must ensure that constraints are written to the correct constraints set and target UCF so that constraints are applied as expected.

Editing Constraints

You can view and modify constraints in the UCF in which they are defined. This makes it easy to cut and paste constraints, and to modify values of existing constraints.

To open a UCF in the Text Editor, double-click the appropriate constraints file name in the Sources view. Figure 7-9 shows an open UCF. You can open multiple files at one time.

Figure 7-9: **Text Editor**

For more information about the commands and features available in the Text Editor, see Using the Text Editor, page 167.

Using Constraint Templates

Common UCF templates are available for use in the Text Editor to assist with defining new constraints. The PlanAhead tool also provides standard Verilog and VHDL language templates, as well as predefined UCF templates. Selected templates can be instantiated into any file that is open in the Text Editor. See Instantiating Language or Constraint Templates, page 169.

Using the Timing Constraints View

You can use the Timing Constraints view to display, edit, and create timing constraints for the design.

PlanAhead provides a view of the timing constraints defined in the design. Constraints are constraint set-specific, and can vary between open designs within the same project. You can experiment with different constraints, devices, I/O pins, and so forth.

You can modify defined values and create new constraints in the Timing Constraints view. To view the timing constraints defined in the design, click the **Timing Constraints** tab, or select **Window > Timing Constraints**.

The Timing Constraints view opens as shown in Figure 7-10.
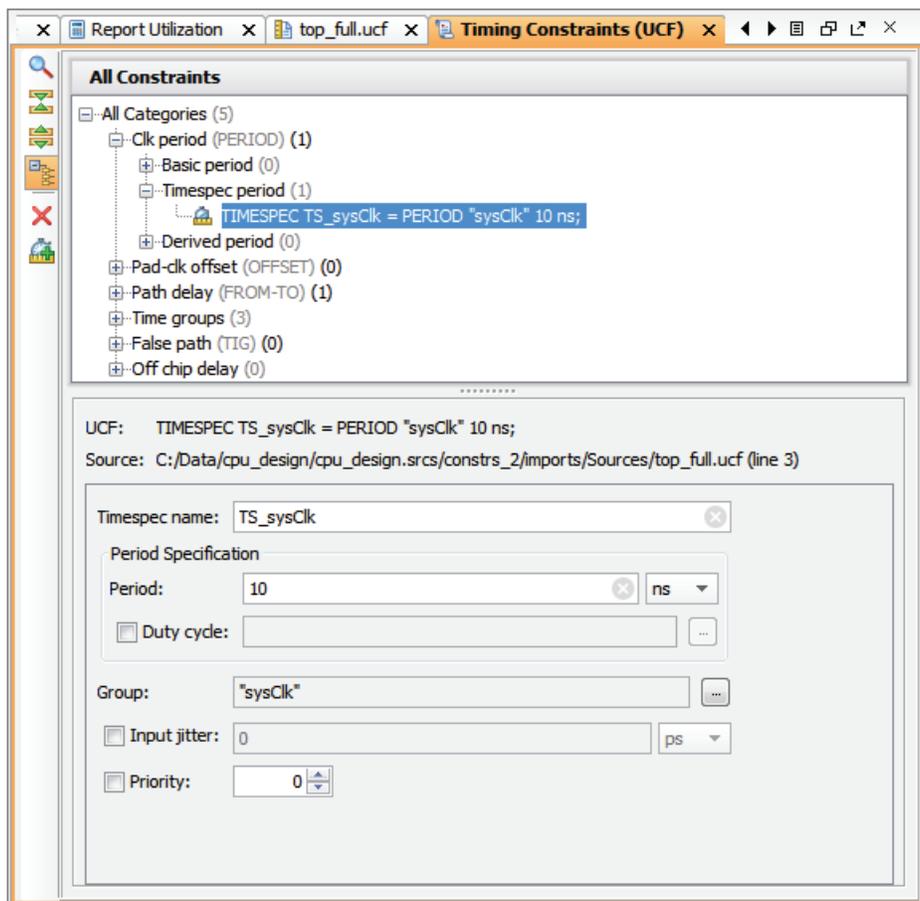
Figure 7-10: **Timing Constraints view**

The constraints display in two different ways: by type, or as a list.

As shown in Figure 7-10, page 220, constraints are sorted by type, allowing expansion and collapsing of the constraint types. Notice that the number of defined constraints of each type displays in parenthesis.

To view all currently defined timing constraints as a list, disable the **Group by type** button in the Timing Constraints view.

You can modify the values of defined constraints by selecting a constraint in the upper half of the Timing Constraints view, and editing the values of the constraint in the lower half of the view as shown in Figure 7-10. The attributes of the constraint that can be changed are displayed in editable fields.

Click **Apply** to save changes to modified constraints values.

Adding New Timing Constraints

1. Open the New Timing Constraints dialog box using one of the following methods:

    •   From the main menu, click **Tools > Timing > New Timing Constraint**

    •   In the Timing Constraints view, click the **New Timing Constraint** button.

    The New Timing Constraint dialog box opens as shown in Figure 7-11.
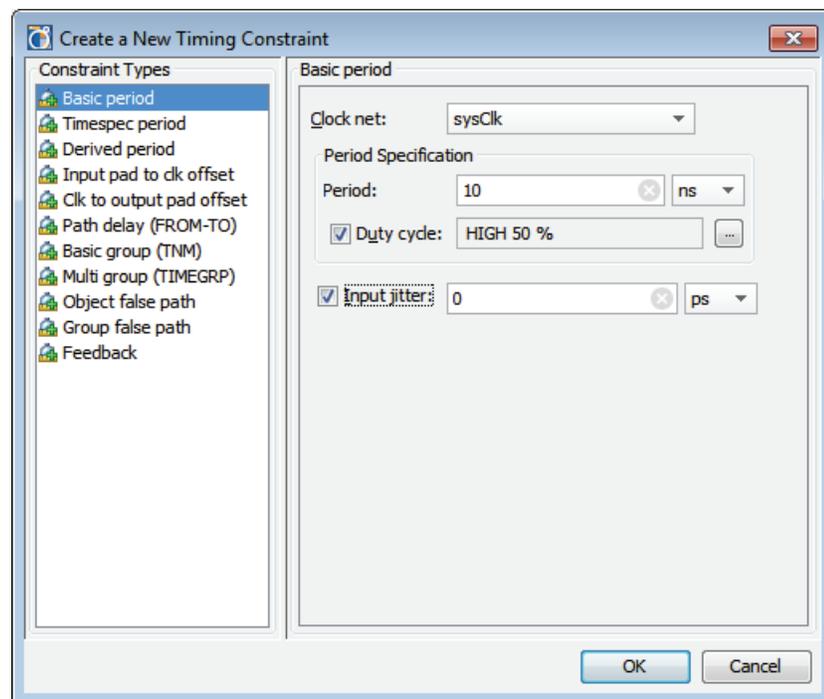
Figure 7-11: **Create New Timing Constraint Dialog Box**

2. On the left, select the type of constraint you want to create.

    The appropriate fields display on the right.

3. Define the constraint values.

4. Click **OK**.

For more information about timing constraints and timing constraints syntax, see the *Constraints Guide (UG625)*, cited in Appendix E, Additional Resources.

Removing Timing Constraints

To remove the constraint from the design, select a constraint or group of constraints in the Timing Constraints view, and select **Delete** from the popup menu. You are prompted to remove the selected constraint(s) prior to the removal of the constraint.

Note: Due to the interdependence between timing constraints, removing one constraint could result in the removal of several other related constraints. Adding, editing, and deleting timing constraints cannot be undone.

Running Timing Analysis

Timing analysis of the Synthesized Design is useful for ensuring that paths are covered by needed constraints to help ensure effective implementation. As more physical constraints, such as Pblocks and LOC constraints, are assigned in the design, the results of the timing analysis becomes more accurate, although still containing some estimation of path delay. As the implementation of the design is completed, the timing analysis includes the actual routed path delays from the implemented design.

Caution! The static timing analysis engine in PlanAhead is intended for timing estimation only, and should not be used for timing sign-off. For sign-off timing, you can run TRCE on an Implemented Design incorporating both the circuit delays and the path delays of the placed and routed design. See Running TRCE on an Implemented Design, page 338 for more information.

Using the Report Timing Command

Perform a timing analysis using one of the following methods:

- From the main menu, click **Tools > Timing > Report Timing**
- From the Flow Navigator, click **Report Timing** from the Synthesized Design menu

The **Report Timing** dialog box opens, as shown in Figure 7-12, and lets you customize the timing report.
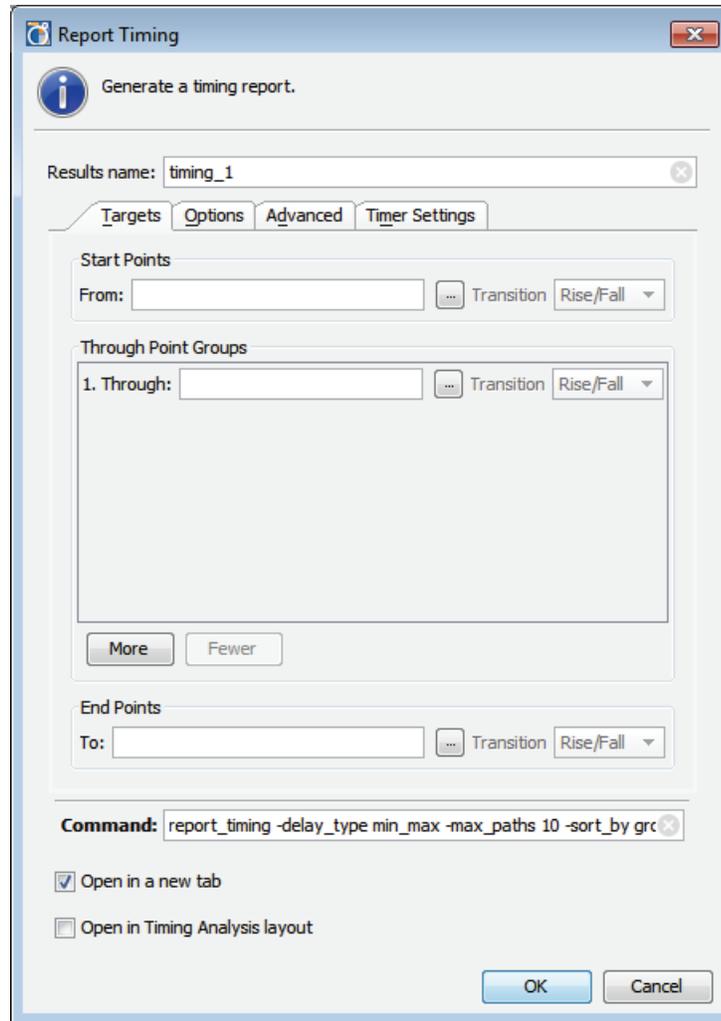
Figure 7-12: **Report Timing Dialog Box**

The **Report Timing** dialog box options are listed as follows:

- **Results Name**: Specifies the name of the timing report.

- **Targets** tab: Allows the filtering of reported paths based on Start Points, Through Points, and Endpoints.

- **Options** tab: Specify the options used in generating the report.

- **Advanced** tab: Contains advanced options used for generating and storing the timing report.

- **Timer Settings** tab: Specify timing engine and delay options.

- **Command**: Contains the text of the Tcl command generated by the Report Timing options.

  Note: This field can be edited, and modifies the Tcl command being executed, but does not change the options specified in the dialog box.

- **Open in New Tab**: Specifies that the timing analysis results open a new tab in the Timing Results view, or replaces an existing tab.

- **Open in Timing Analysis layout**: Opens the timing report in the Timing Analysis view layout.

- Click **OK** to run the timing report.

The following subsections describe the various tabs of the Report Timing dialog box.

Understanding the Targets Tab

The Targets tab of the Report Timing dialog box, shown in Figure 7-12, page 223, contains fields where you can specify the starting points, through points, and end points of the paths to include in the timing report.

By default, the dialog box opens with blank fields indicating all points included in the report up to the maximum number of specified paths. By adding values to the various fields, you can generate a report focusing on the paths of interest.

The available fields are:

- **Start Points**: Lets you choose the synchronous elements to begin the paths for analysis. The Start Point fields are:

  - **From**: Contains an expression used to filter the starting points. Enter text in this field to manually create the filter, or view the filter text created by the Choose Start Points dialog box.

  - **Choose Start Points**: Opens a dialog box used to build expressions that filter the starting points. This dialog box is described in further detail in Understanding the Choose Points Dialog Box.

  - **Transition**: Further filters the paths according to the active clock edge of the starting point synchronous elements. This field contains the following values:

    - **Rise**: Filters starting point synchronous elements to those triggered by a rising (positive) clock edge.

    - **Fall**: Filters starting point synchronous elements to those triggered by a falling (negative) clock edge.

    - **Rise/Fall**: Includes both rising and/or falling clock edges.

- **Through Point Groups**: Enter paths that travel through a set of points for analysis. The following fields are available:

  - **Through**: Contains an expression used to filter the paths based on the points the path travels through.

  - **Choose Through Points**: Opens a dialog box to build the expression that filters the paths based on the through points.

  - **Transition**: Further filters the paths according to the active clock edge of the through point synchronous elements. This field contains the following values:

    - **Rise**: Filters through point synchronous elements to those triggered by a rising (positive) clock edge.

    - **Fall**: Filters through point synchronous elements to those triggered by a falling (negative) clock edge.

    - **Rise/Fall**: Includes both rising and/or falling clock edges.

  - **More**: Adds additional through point filter expressions

  - **Fewer**: Removes existing through point filter expressions

- **End Points**: Enter paths that end in a set of synchronous elements. This section contains the following fields:

  - **To**: Contains an expression used to filter the paths based on the points the path travels to, or end points.

- • **Choose End Points**: Opens a dialog box used to build the expression that filters the paths based on the end points.

- • **Transition**: Further filters the paths according to the active clock edge of the starting point synchronous elements. This field contains the following values:

  - - **Rise**: Filters ending point synchronous elements to those triggered by a rising (positive) clock edge.

  - - **Fall**: Filters ending point synchronous elements to those triggered by a falling (negative) clock edge.

  - - **Rise/Fall**: Includes both rising and/or falling clock edges.

Understanding the Choose Points Dialog Box

The Choose Points dialog box lets you choose the design elements for which you require timing analysis based on element type and a pattern matching string and lets you enter filter strings for the **Start Points**, **Through Points**, and **End Point**. Figure 7-13 shows the Choose Start Points dialog box.
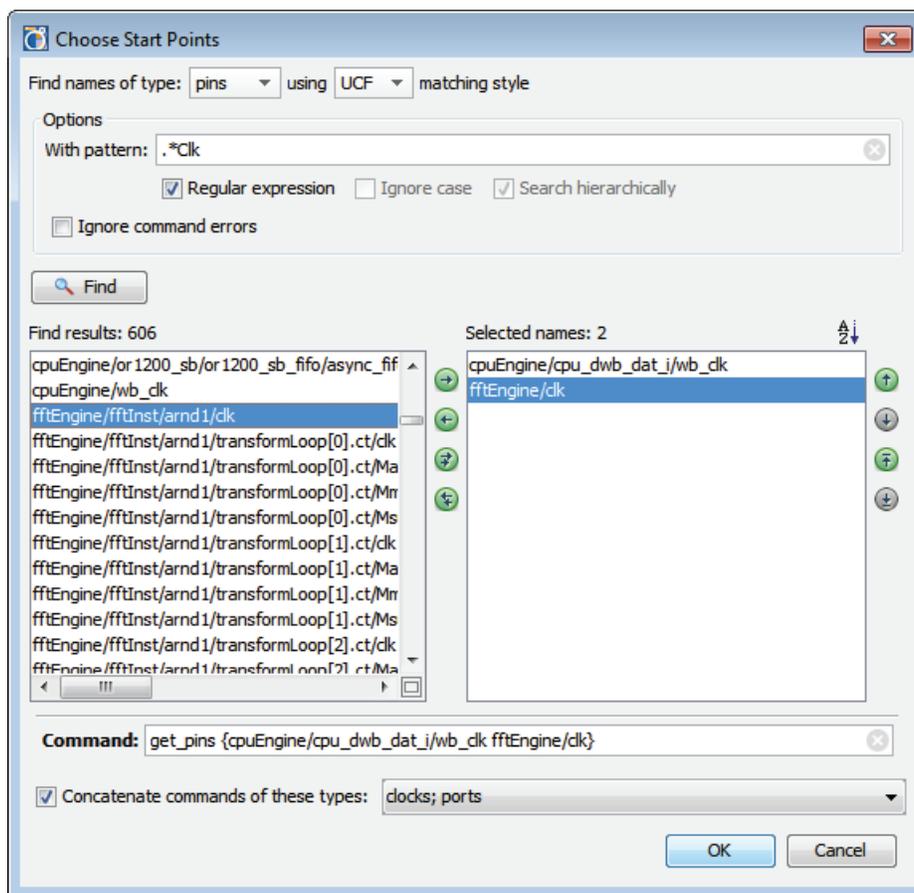
Figure 7-13: **Choose Start Points Dialog Box**

The options are:

- **Find Names of Type**: Filters the points based on the type of design element. The field contains the following options:

  - **Cells**: Choose design elements based on cell name.

  - **Clocks**: Choose design elements based on clock name.

  - **Pins**: Choose design elements based on pin name.

  - **Ports**: Design elements based on port name.

  - **Nets**: Specify nets for Through Points.

- **Matching Style**: Select the type of pattern matching used for filtering the design elements. The field contains the following options:

  - **UCF**: Choose UCF-based syntax for pattern matching.

  - **SDC**: Choose an SDC-based syntax for pattern matching.

- **With Pattern**: The pattern expression used to filter the design elements. This field is modified with the following options:

  - **Regular Expression**: Specifies that the search string uses regular expression syntax.

  - **Ignore Case**: Specifies the search string is case insensitive.

  - **Search Hierarchically**: This option is available for SDC pattern matching, and specifies that the SDC-based search pattern is applied to every level of hierarchy.

- **Of These Objects**: Select objects based on dialog box selection. The following options are available:

  Note: This field is only available when matching style is set to SDC.

  - **Include Leaf Pins**: An SDC syntax option that specifies that the search string should only match pins components, and not match pins across hierarchical boundaries.

  - **Select Object Dialog Box**: An SDC syntax option that launches an additional object selection dialog box where you can generate recursive search expressions.

  - **Filter Matching Name with Expression**: An SDC syntax option that specifies the `-filter` command.

- **Ignore Command Errors**: Suppresses warning messages generated during Tcl command processing of the timing report.

- **Find**: Command button for launching a search based on the defined expression.

- **Find Results**: Contains the results of the object search.

  - **Move Item to the Right**: Moves the currently selected object from the Find Results column to the Selected Names column.

  - **Move Item to the Left**: Moves the currently selected object from the Selected Names column to the Find Results column.

  - **Move All Items to the Right**: Moves items from the Find Results column to the Selected Names column.

  - **Move All Items to the Left**: Moves items from the Selected Names column back to the Find Results column.

- **Selected Name**: Contains the subset of the search result objects that are chosen for Start Points, Through Points, or End Points.

- **Command**: Contains the Tcl command used to represent the selected objects.

- **Concatenate commands of these types**: Combine different objects (pins, cells, ports, clocks) as start points for the timing report.

Understanding the Options Tab

The Options tab of the Report Timing dialog box contains fields to specify the type of timing report to be generated as well as the number of paths reported. The Options tab is shown in Figure 7-14.
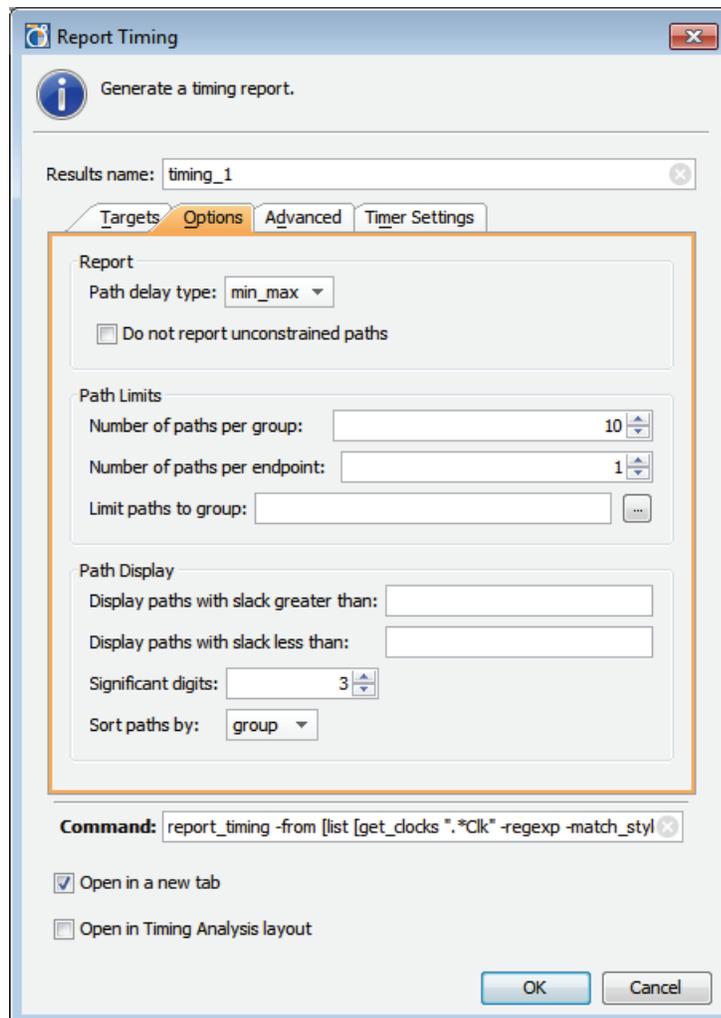
Figure 7-14: **Report Timing - Options tab**

The fields available in the Report Timing Options tab are:

- **Path Delay Type**: Specifies the type of delays that are used in the timing report path analysis. Accepted values are min, min_rise, min_fall, max, max_rise, max_fall, and min_max. A "rising" or "falling" path delay refers to the transition at the timing path endpoint.

- **Do Not Report Unconstrained Path**: Specifies that the report contain details for constrained paths only.

- **Number of Paths Per Group**: Specifies the number of timing paths reported per group.

- **Number of Paths Per Endpoint**: Specifies the maximum number of timing paths reported per endpoint.

- **Limit Paths Per Group**: Limits the paths to a group or set of groups. The group identifier can be entered directly, or by using the Choose Path Groups dialog box.

- **Display Paths With Slack Greater Than**: Filters the displayed paths based on a minimum slack value. Only paths with slack greater than this value are shown.

- **Display Paths With Slack Less Than**: Filters the displayed paths based on a maximum slack value. Only paths with slack less than this value are shown.

- **Significant Digits**: Specifies the significant digits of the timing report delay values. The default value is 3.

- **Sort Paths By**: Selects that parameter that are used to sort the timing report. This field contains the following values:

    - **Group**: Sorts the timing report based on group name.

    - **Slack**: Sorts the timing report based on path slack.

Understanding the Advanced Tab

The Advanced Options tab of the Report Timing dialog box contains fields that let you specify pin and net details, report output locations, and command error processing. The Advanced Options tab is shown in Figure 7-15.
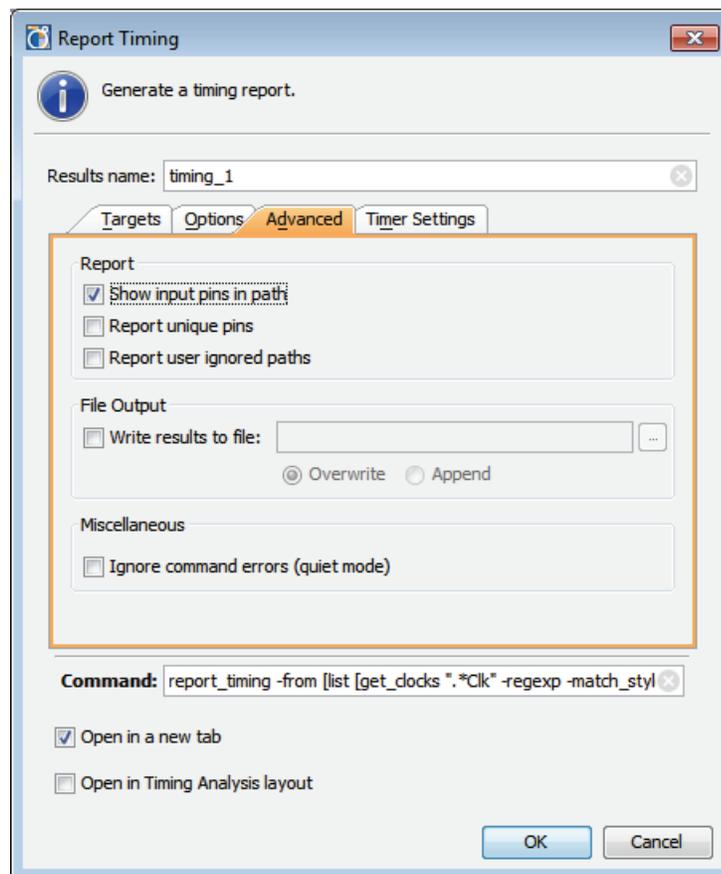
Figure 7-15: **Report Timing - Advanced Tab**

The Advanced tab options are:

- **Show input pins in path**: Shows the input pins that start each path.

- **Report unique pins:** Show only one timing path for each unique set of pins.

- **Report user ignored paths:** Report only the paths that are usually ignored by timing due to the being defined as false paths.

- **Write results to file**: Write the results of the timing report to a file.

  - **Overwrite**: Overwrites a file with the same name as the specified file.

  - **Append**: Appends the timing report details to the specified file.

- **Quiet Mode**: Suppresses messages in the timing report regarding errors in command options.

Understanding the Timer Settings Tab

The Timer Settings tab of the Report Timing dialog box contains fields that allow the designer to specify the delays parameters used by the timing engine in generating the timing report. Figure 7-16, page 229 shows the Timer Settings tab.
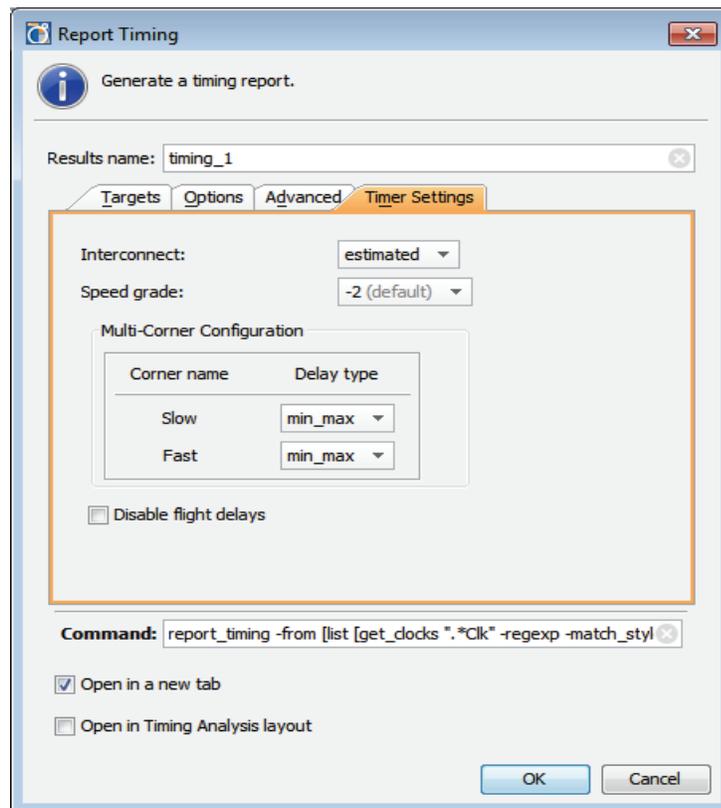
Figure 7-16: **Report Timing - Timer Settings Tab**

The options are:

- **Interconnect**: Selects the type of delay values used for the interconnect delay. The different delay values are as follows:

  - **Estimated**: Uses estimated delays for the interconnect values.

  - **None**: Sets the interconnect delays to 0.

- **Speed Grade**: Selects the speed grade of the device used in the timing analysis. This field allows the estimation of design timing using different device speed grades.

- **Multi-corner analysis**: Multi-corner analysis simultaneously uses different process and operating condition corners to perform a worst-case setup and hold analysis. This results in a more accurate, but pessimistic, analysis than minimum or maximum delays alone.

- **Slow corner**: Selects the delay types used for the slow corner analysis. The available values are:
  - **None**: Specifies that no delays are used.
  - **Max**: Specifies that maximum delays are used for the clock and data paths during setup and hold analysis.
  - **Min**: Specifies that minimum delays are used for the clock and data paths during setup and hold analysis.
  - **Min_Max**: Uses a combination of minimum and maximum delays for the clock and data paths during setup and hold analysis.
- **Fast Corner**: Selects the delay types used for the fast corner analysis. The values are:
  - **None**: Specifies to use no delays.
  - **Max**: Use maximum delays for the clock and data paths during setup and hold analysis.
  - **Min**: Use minimum delays for the clock and data paths during setup and hold analysis.
  - **Min_Max**: Use a combination of minimum and maximum delays for the clock and data paths during setup and hold analysis.
- **Disable flight delays:** This configures timing analysis to not add package delays to I/O calculations when this option is selected.

Analyzing Timing Results

The Timing Results view opens when the PlanAhead Timing Analysis completes. Figure 7-17 shows an example of timing results. The Timing Results view is also available when you open an Implemented Design and run TRCE for sign-off timing analysis. See Analyzing Timing Results, page 338 for more information.
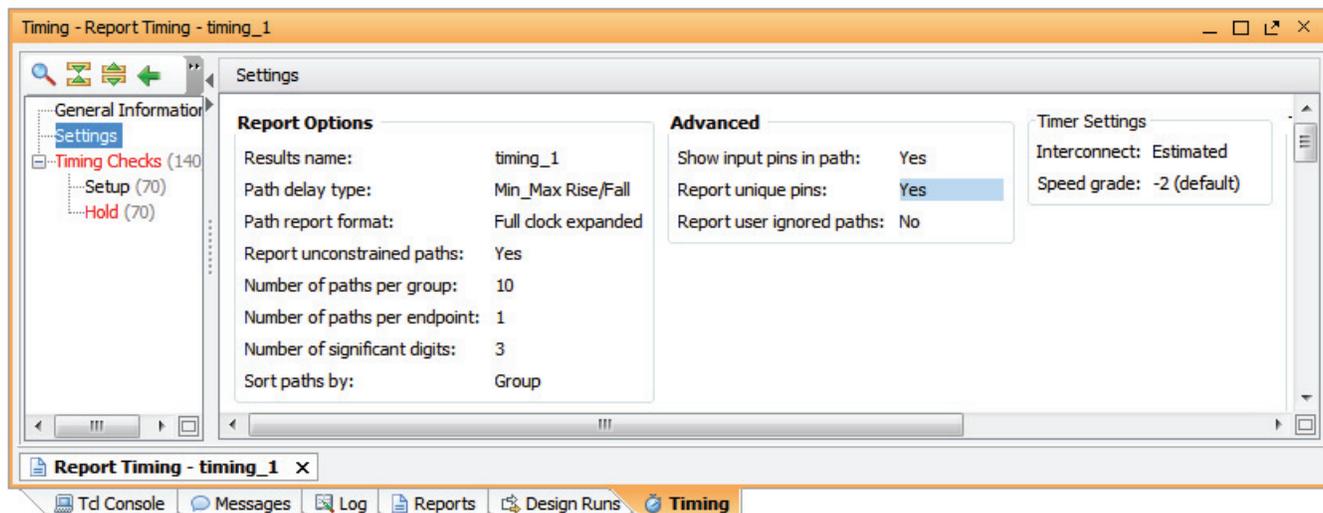
Figure 7-17: **Timing Results**

Either the PlanAhead Timing Analysis or the ISE TRCE tool must be run to populate the Timing Results view with paths. The Timing Results view contains the paths that meet the criteria defined in the Run Timing Analysis dialog box, as described in Running Timing Analysis, page 222.

You can examine, sort, and select specific paths and instances in the Timing Results view. The following information is displayed:

- **General Information**: Reflects specific information about the design and the report_timing command.
- **Settings**: Displays a summary of the Report Timing command, and the settings used when the timing analysis was performed.
- **Timing Checks**: Sorted into Setup and Hold checks, the Timing Checks provide a table view for the timing results, initially sorted by the source clock for the timing paths.
  - **Name**: Displays the source clock, and a sequential name for the timing paths reported.
  - **Slack**: Displays the total positive or negative slack on the path.
  - **From**: Displays the path source pin.
  - **To**: Displays the paths destination pin.
  - **Total Delay**: Lists the total estimated delay on the path.
  - **Logic Delay**: Lists the delay attributed to logic elements on the path.
  - **Net Delay**: Lists the delay attributed to the interconnect of the path.
  - **Logic%**: Displays the percentage of the delay attributed to logic elements.
  - **Net%**: Displays the percentage of the delay attributed to interconnect.
  - **Stages**: Displays the total number of instances on the path including the source and destination which contribute to the overall delay.

    Note: In the PlanAhead tool, carry chain interconnect is counted as individual stages of logic, so the stages reported might be different than levels of logic reported in ISE.
  - **Source Clock**: Displays the source clock name.
  - **Destination Clock**: Displays the destination clock name.

Sorting the Timing Report

You can sort the Timing Results by clicking any of the column headers. For example, click on the **Stages** column header to sort the list by stages of logic. Click the column header a second time to reverse the sort order.

You can sort by a second column by pressing the **Ctrl** key and clicking a second column header. You can sort by as many columns as necessary to refine the sort results. Press **Ctrl** and click the column header again to remove a sort from a column.

Refer to the Using Tree Table Style Views, page 113 for information regarding working with tree table style views.

Flattening the List of Paths

By default, the paths are categorized by constraint. You can flatten the list and view all paths by clicking **Group by Constraint** button in the Timing Results view toolbar, as shown in Figure 7-17, page 230. The Group by Constraint toolbar button toggles between a list of paths grouped by constraint, and a flattened list of paths.

Displaying Path Details

When you select a path from the list, the Path Properties view populates with information about the path. Logic elements are listed with detailed delay information, shown in Figure 7-18, page 232.

Figure 7-18: **Path Properties View: Report Tab**

The report has a similar format to the TRCE report.

- By default, selecting a path also selects all instances contained within the path.

- Selecting any of the objects in the report cross-selects the object in other open views such as the Netlist and Device views.

- Select multiple paths using the **Shift** or **Ctrl** key while selecting objects.

- All instances contained in the selected paths are selected, but the Path Properties displays information only about the first path selected.

Displaying Timing Path Reports

A Timing Path report can also be displayed in the workspace for easier viewing. This displays the same information as the Path Properties view, but lets you display multiple paths at the same time. The Path Properties view only displays the properties for the last selected path.

To view the Timing Path report:

1. Select a timing path.

2. Select the **View Path Report** popup menu command.

Analyzing Clock Interactions

In large complex FPGA designs, data is frequently transferred from one clock domain to another. To identify potential problems such a metastability or data loss or incoherency some visibility into the paths that cross clock domains is beneficial. The Clock Interaction Report in PlanAhead offers insight into clock interactions and signals that cross clock domains.

The path between a flip-flop in one clock domain to logic in a second clock domain is not automatically analyzed by the ISE or PlanAhead timing tools unless there is a constraint that specifies that the two clock domains are related. Therefore, unconstrained paths might go unanalyzed. The purpose of the Clock Interaction Report is to look at the circuit topology and the design constraints and inform you of any paths that cross clock domains, whether they have defined constraints or not. In this way you can identify paths that are properly constrained, unconstrained, or are false paths that should simply be ignored. The Clock Interaction Report can also help you find signals that are implied in the RTL code and inadvertently cross clock domains, and so might lack required constraints.

Reporting Clock Interactions

With a Synthesized Design open in PlanAhead, select:

- **Tools > Timing > Report Clock Interaction** command from the main menu.

- **Report Clock Interaction** from the Synthesis menu of the Flow Navigator.

This brings up the Report Clock Interaction form. which has a number of fields and switches on it, as shown in Figure 7-19, page 234.

- **Results Name**: Specifies the output results set.

- **Options Tab**: Allows the customization of the Clock Interaction report. The detailed options of this tab are discussed in Setting the Clock Interaction Options.

- **Timer Settings Tab**: Specifies timing engine and delay options used to generate the timing report. The detailed options of this tab are discussed in Using the Timer Settings tab, page 235.

- **Command**: Contains the text of the Tcl command generated by the Report Clock Interaction options.

Note: This field can be edited to modify the Tcl command being executed, but does not change the options specified in the dialog box.

- **Open in a new tab**: When enabled, this switch opens the results of the Report Clock Interaction command in a new window without closing any older open windows. However, when not selected, the older window is closed before the new results are displayed.

Setting the Clock Interaction Options

The Options tab has three fields or switches as shown in Figure 7-19:
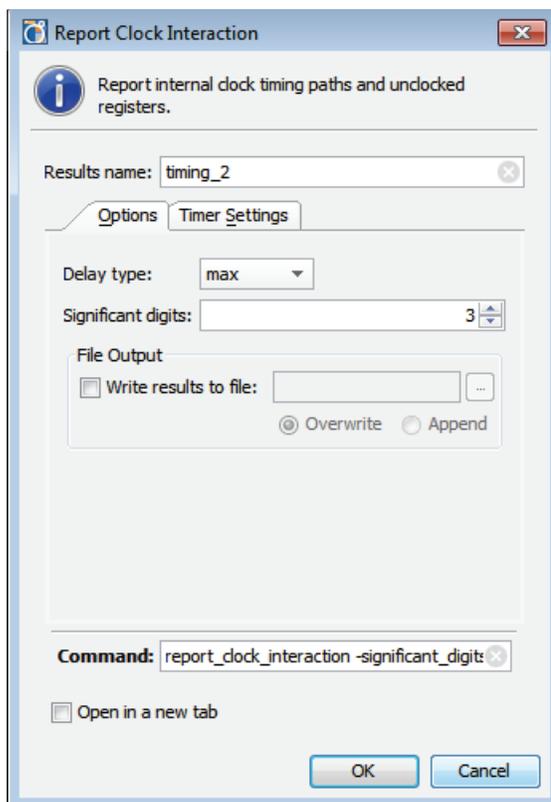
Figure 7-19: **Report Clock Interaction - Options Tab**

- **Delay Type**: Specifies the type of delays that are used in the Clock Interaction analysis. This field contains the following options:
    - **Max**: Uses maximum delays for the clock and data paths during setup and hold analysis.
    - **Min**: Uses minimum delays for the clock and data paths during setup and hold analysis.
    - **Min_Max**: Uses a combination of minimum and maximum delays for the clock and data paths during setup and hold analysis.

- **Significant Digits**: Determines the significant digits reported in the Clock Interaction report. Specified values from 0 to 13.

- **Write Results to File**: Specify a file to save the results to.
    - **Overwrite:** Overwrite the named file, if one exists, with the new file.
    - **Append:** Append the results of the report to the named file.

Using the Timer Settings tab

The Timer Settings tab has three fields or switches as shown in Figure 7-20:
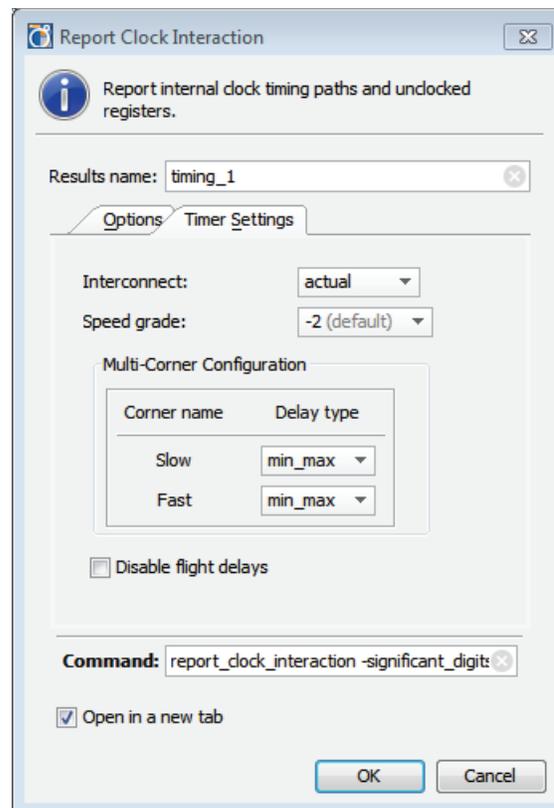
Figure 7-20: **Report Clock Interaction - Timer Settings Tab**

- **Interconnect**: Selects the type of delay values used for the interconnect delay. The different delay values are as follows:
  - **Estimated**: Uses estimated delays for the interconnect values.
  - **None**: Sets the interconnect delays to 0.
- **Speed Grade**: Selects the speed grade of the device used in the timing analysis. This field allows the estimation of design timing using different device speed grades.
- **Multi-corner analysis**: Multi-corner analysis simultaneously uses different process and operating condition corners to perform a worst-case setup and hold analysis. This results in a more accurate, but pessimistic, analysis than minimum or maximum delays alone.
  - **Slow corner**: Selects the delay types used for the slow corner analysis. The available values are:
    - **None**: Specifies that no delays are used.
    - **Max**: Specifies that maximum delays are used for the clock and data paths during setup and hold analysis.
    - **Min**: Specifies that minimum delays are used for the clock and data paths during setup and hold analysis.
    - **Min_Max**: Uses a combination of minimum and maximum delays for the clock and data paths during setup and hold analysis.

- **Fast Corner**: Selects the delay types used for the fast corner analysis. The values are:

  - **None**: Specifies that no delays are used.

  - **Max**: Uses maximum delays for the clock and data paths during setup and hold analysis.

  - **Min**: Uses minimum delays for the clock and data paths during setup and hold analysis.

  - **Min_Max**: Uses a combination of minimum and maximum delays for the clock and data paths during setup and hold analysis.

- **Disable flight delays**: This configures the timing engine to not add package delays to I/O calculations when this option is selected.

Viewing the Clock Interactions Matrix
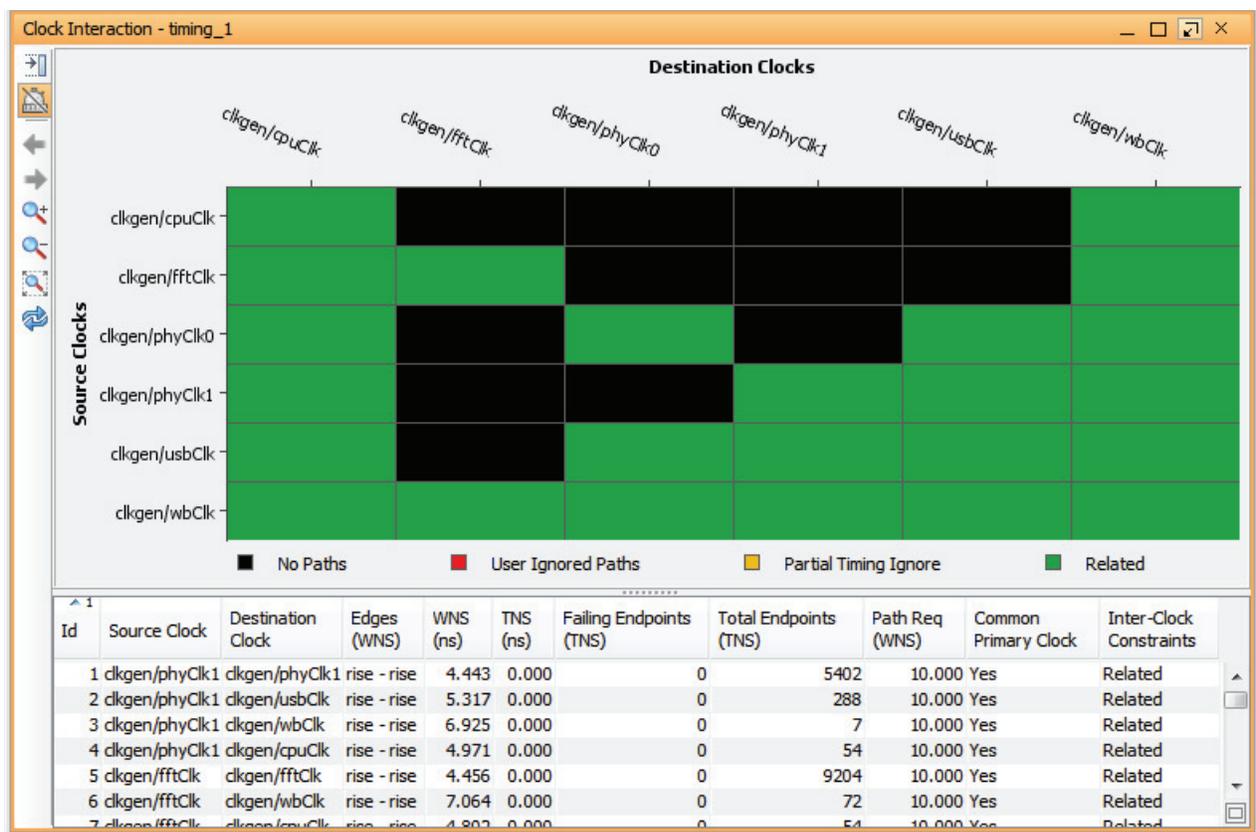
Figure 7-21: **Clock Interaction Report**

After running the Report Clock Interaction command, the results open in the Clock Interaction view. The Clock Interaction Report displays as a matrix of clock domains with the launching clocks in the vertical axis and the receiving clocks in the horizontal axis. A table showing the worst case slack calculated for each cross clock domain is displayed below the matrix. Figure 7-21 shows an example of the Clock Interaction Report and its table.

In Figure 7-21 there are six separate clock domains found by the PlanAhead tool, so each clock is analyzed against all the other clock domains to determine if there are timing paths that cross those domains. The state of constraints across any two clock domains is reported in the Clock Interaction

Report. The results display in a six-by-six matrix which uses the following color scheme to display the state of cross domain signals:

- A white (or black[1]) cell indicates that no paths were found crossing between the two clock domains. In this case there is no clock interaction and nothing to report.

- A green cell indicates properly constrained paths cross from the launching clock to the receiving clock. All the paths have been constrained with the use of the `FROM:TO` constraint.

  Note: The diagonal line in which the launching clock and the receiving clock are the same is always green.

- An orange or yellow cell indicates partially constrained clock domain crossing. Some of the paths through the logic are properly constrained, but some paths are not constrained. This could be due to paths that are implied in the RTL and so are not properly constrained in the Synthesized Design.

- A red cell indicates an unconstrained clock interaction. None of the paths crossing between clock domains have the required `FROM:TO` constraint.

Green or Black is the desired result in this matrix. A red or yellow cell highlights a potential timing problem. Note that the color of a cell in the matrix reflects the state of the constraints between clock domains, not the state of timing of the paths between the domains. A green cell does not indicate that the timing is good, only that timing paths are constrained across clock domains. The table below the matrix shows the state of the timing.

Select the **Clock Interaction View Layers** command in the toolbar to filter the clocks to display the specific clocks of interest. This command reduces the matrix complexity by limiting the number of clocks, as shown in Figure 7-22, but does not reduce the number of clock interactions reported in the table below the matrix.
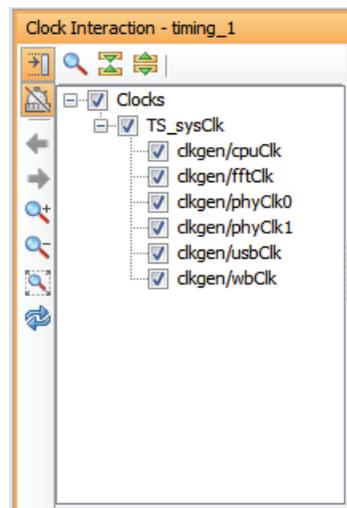
Figure 7-22: **Report Clock Interaction - Filter Clocks**

The table below the matrix displays the path with the worst case negative slack for each clock interaction. This provides details not initially displayed in the matrix above. The table does not include paths from cells in which there are no cross-clock interactions; the cells displayed in black on the matrix. Therefore, a six by six matrix might result in a table with only a few results.

---

1. The color of the cell is determined by the background color of the Graphical Editors as defined under the **Tools > Options** command. See Configuring the Viewing Environment in Chapter 4 for more information.

The data in the table can be sorted by selecting one of the column headers as a key for sorting in increasing or decreasing amounts. Select once for up, once for down, and once again turns it off restoring the table to its original state. Use the **CTRL** key to select additional columns to add secondary sorting criteria. See Using Tree Table Style Views in Chapter 4 for more information on working with tables.

Selecting a cell in the matrix cross-selects a specific row of the table below, and selecting a row from the table highlights a cell in the matrix above.

The specific columns from the table are:

- **ID**: Indicates a numeric ID for the path being displayed.

- **Source Clock**: Displays the clock domain from which the path originates.

- **Destination Clock**: Displays the clock domain within which the path terminates.

- **Edge (WNS):** The clock edges used to calculate the worst negative slack.

- **WNS** (Worst Negative Slack): Displays the worst slack calculated for various paths crossing the specified clock domains. A negative slack indicates a problem in which the path violates a required setup time.

- **TNS** (Total Negative Slack): The sum of the worst slack violation for all the endpoints that belong to paths crossing the specified clock domains.

- **Failing Endpoints:** The number of endpoints in the crossing paths that fail to meet timing. The sum of the violations corresponds to TNS.

- **Total Endpoints:** The total number of endpoints in the crossing paths.

- **Path Requirements (WNS):** The timing path requirement corresponding to the path reported in the WNS column.

- **Common Primary Clock:** Displays whether the source and destination clocks of the timing path are defined by a common primary clock.

- **Inter-Clock Constraints:** Displays whether the clock domains are related, or are defined as false paths, or constrained as asynchronous groups.

A popup menu available from the right mouse button from within the table holds the **Export to Spreadsheet** command. This command exports the table to an XLS file for use in a spreadsheet application.

Note: The cursor must be located in the field of the table, and not in the headers of the column to see the **Export to Spreadsheet** command.

Defining Physical Constraints

The PlanAhead tool provides a variety of ways to apply physical constraints. Physical constraints consist of `LOC/BEL` instance placement constraints, `AREA_GROUP` placement constraints, `DCI_CASCADE` constraints and Device Configuration Mode constraints. For more information on assigning physical constraints, refer to Chapter 8, I/O Pin Planning, and Chapter 10, Floorplanning the Design.

The following subsections describe the Physical Constraints view and the available options.

Using the Physical Constraints View

The Physical Constraints view is used to display and interact with a variety of physical constraint types. The Physical Constraint view is opened when you switch to the Floorplanning view layout using **Layout > Floorplanning**, or by selecting the Floorplanning layout in the Layout Selector on

the toolbar. You can also open the Physical Constraints view at any time by selecting **Window > Physical Constraints**.

The Physical Constraints view displays the hierarchical structure of the design relative to the created Pblocks. The physical hierarchy is dynamic; expanding and changing automatically as you manipulate the physical hierarchy. As objects are selected in other views, the appropriate elements are highlighted in the Physical Constraints view.

The objects that display in the Physical Constraints view are Relatively Placed Macros (RPMs) and Physical Block (Pblocks), and DCI Cascade Constraints. You can select these objects in the Physical Constraints view for manipulation in other views.

Any physical constraints that you add to the design are saved when you select:

- **File > Save Constraints**, or
- **File > Save Constraints As**

You are prompted to save any unsaved design changes if you close the project without saving. If you have a design with multiple constraint sets, or multiple constraint files, be aware that any added constraints are written to the target constraint file of the active constraint set.

Using the ROOT Design Pblock

The physical hierarchy begins with the design name of the Synthesized Design, netlist_1 as a default, followed by the top-level Pblock called "ROOT."

As you create lower-level Pblocks they display in a hierarchical fashion, with the child Pblock nested under the parent. Figure 7-23 shows different levels of Pblock nesting.
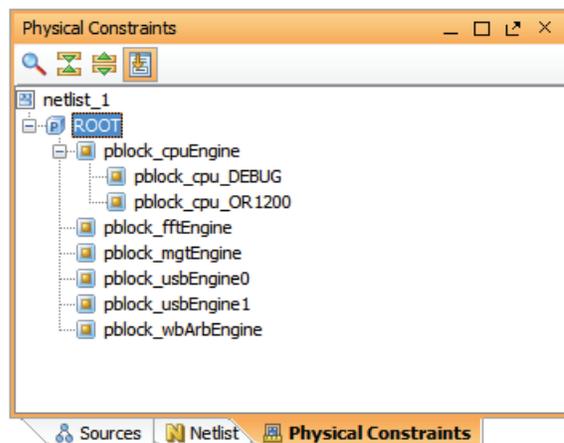
Figure 7-23:   **Physical Constraints View**

Selecting a Pblock selects all of the assigned logic for that Pblock.

Understanding the Physical Constraints View Icons

The Physical hierarchy Pblock tree uses several icons that can help you identify the state of the various objects. This display updates automatically as you manipulate the physical hierarchy.

As you create Pblocks. they appear in a hierarchical fashion in the Physical Constraints view. Each folder type in the Physical Constraints view displays a number in parenthesis that details the number of objects in that folder.

Each instantiation of an RPM displays in the Physical Constraints view. If RPMs are assigned to Pblocks, they appear in an RPM folder under the Pblock. Selecting an RPM in the Physical Constraints view selects all of the logic contained in the RPM.

Pblocks With Assigned Instances

Pblocks with instances assigned and *with* rectangles defined in the Device view appear as blue three-dimensional cubes with a yellow center as shown in the following snippet.
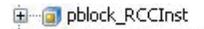
Pblocks with instances assigned and *with no* rectangles defined in the Device view appear as blue two-dimensional squares with a yellow center as shown in the following snippet.

Pblocks Without Assigned Instances

Pblocks with no instances and *with* rectangles defined assigned appear as blue three-dimensional cubes with a blue P in the center as shown in the following snippet.

Pblocks with no instances assigned with *no* rectangles defined appear as blue two-dimensional squares with a blue P in the center as shown in the following snippet.

Partially Reconfigurable PBlocks

Partially reconfigurable partition Pblocks appear as yellow diamonds as shown in the following snippet.

Working with Relatively Placed Macro (RPMs)

Relatively Placed Macros (RPMs) folders list the RPMs that exist in a Design. RPMs can be assigned to Pblocks. If RPMs are assigned to Pblocks, they appear in an RPM folder under the Pblock. Each instantiation of an RPM displays in the Physical Constraints View, as shown in Figure 7-24.
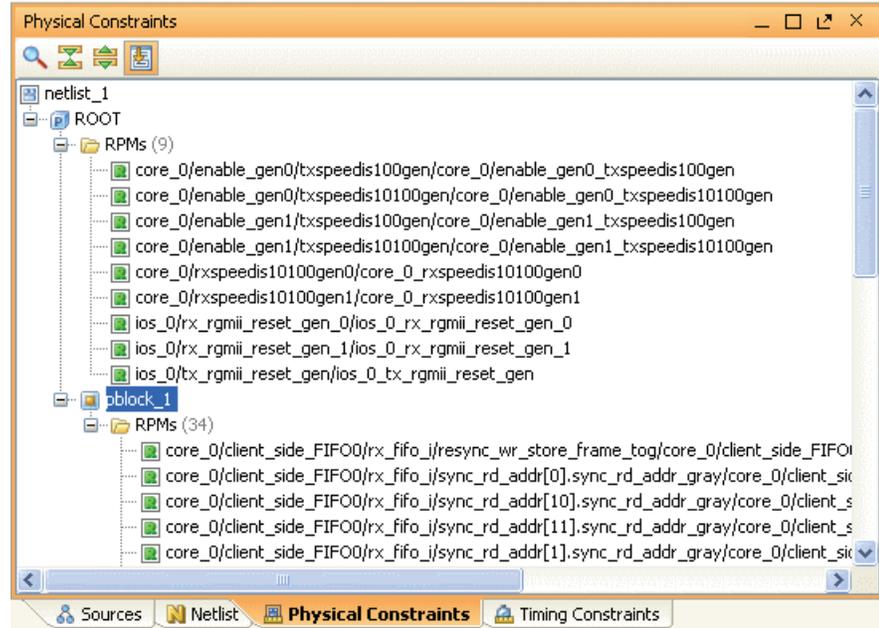
Figure 7-24:   **Displaying Relatively Placed Macros "RPMs"**

RPM properties and statistics display in the RPM Properties view. When RPMs are assigned to Pblocks, the Pblock Properties view displays RPM size and utilization statistics information, as shown in Figure 7-25.

Figure 7-25:   **RPM Utilization Statistics for Pblocks**

Running the Design Rule Checker (DRC)

The PlanAhead tool contains a set of batch DRC commands that can help verify design integrity prior to running the ISE software. The rules are categorized by type of logic checks being performed. Many different types of checks are available.

These checks are designed to provide an early indication about potential design implementation issues. The final validation step to ensure the design is DRCs compliant is to run the design through the ISE implementation tools.

Running I/O Port and Clock Logic DRCs

Many of the DRC rules are related to I/O pin assignment and clock logic. For information on running I/O Port and clock logic related DRCs, see Chapter 8, I/O Pin Planning, and Appendix B, PlanAhead DRCs.

Running Netlist and Constraint DRCs

To run DRCs on the Synthesized Design to validate netlist and constraint condition:

1. From the Flow Navigator or Tools menu, select the **Report DRC** command.

    The Run DRC dialog box opens, as shown in Figure 7-26, page 243.

    Note: The number of checks for any category may vary from release to release.

2. View or edit the Results Name field. Enter a name for the results for a particular run for easier identification during debug in the DRC Violations browser.

3. Enter an **Output File** name to create a report text file. This is an optional field.

4. In the **Rules to Check** group box, use the check boxes to select the design rules to check for each design object. For a description of each rule, see Appendix B, "PlanAhead DRCs,".

    •   Expand the hierarchy using the **Expand All** toolbar buttons, or click the + next to each category or design object.

    •   Click the check box next to the design object to run all DRCs.

    •   Click individual DRCs to run individual ones.

    •   Click **Select All** to run a complete DRC.

5. Click **OK** to invoke the selected DRC checks.

The DRC Results view displays the rule violations found, grouped under the various rule categories defined in the Run DRC dialog box, shown in Figure 7-26, page 243.

The rule violations are also categorized by severity and color-code. A violation can be:

•   Informational only, to make you aware of a possible issue; shown with a yellow marker.

•   A warning or a critical warning to suggest an issue that might need some resolution; shown with an orange marker.

•   An error to highlight issues that prevent proper implementation of your design; shown with a red marker.
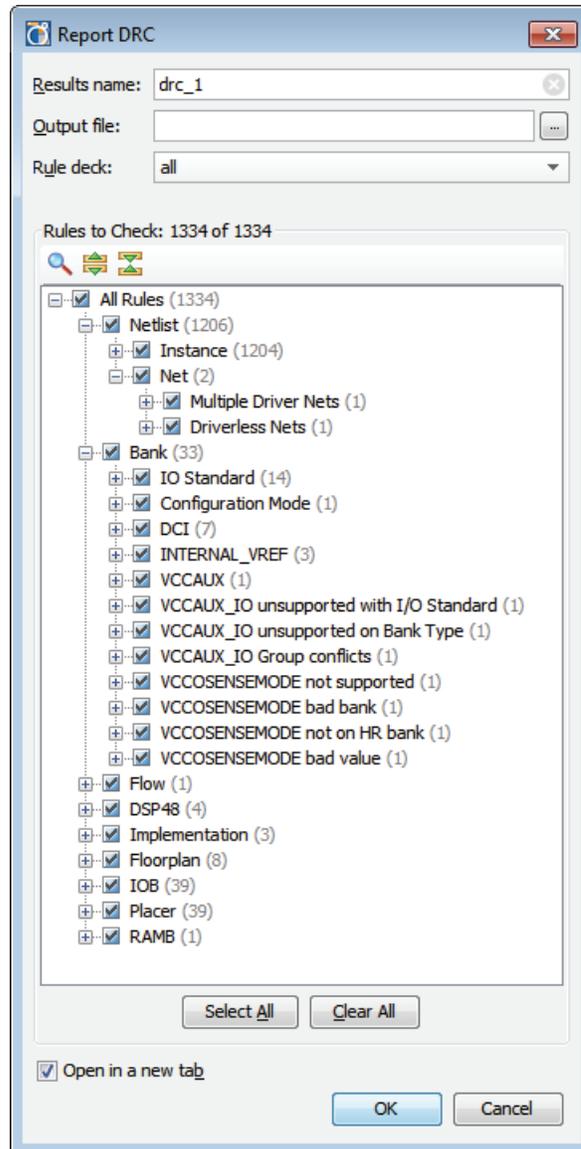
Figure 7-26: **Run DRC Dialog Box**

Viewing DRC Violations

After the DRCs are completed, the DRC Results view opens, as shown in Figure 7-27.
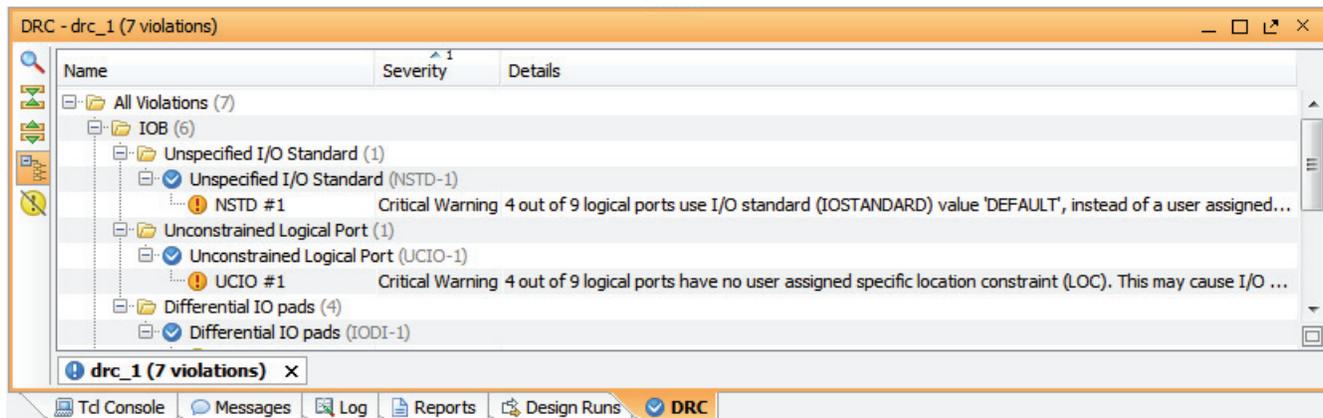
Figure 7-27: **DRC Results**

You can toggle the **Hide warnings and informational messages** button in the toolbar menu to turn off warnings and informational messages to see only the errors reported.

You can also click on the header of the **Severity** column of the DRC Results view to sort violations by severity:

- Click once on the column header to sort in an increasing order
- Click twice to sort in a decreasing order

See Using Tree Table Style Views, page 113 for more information. When you select a violation message in the DRC Results view, you can select **Violations Properties** in the popup menu to open the Violations Properties view. This view shows both a general review of the DRC rule violation, and specific details of the design elements that violate the rule.

The **Details** tab of the Violations Properties can have links to specific design objects that violate the DRC. These links can be clicked to view the design object in the Netlist view, the Device view, the Schematic or the source RTL file. Appendix B, PlanAhead DRCs lists the PlanAhead DRCs.

I/O Pin Planning

The I/O Planning view layout provides an interface to analyze the design and device I/O requirements, and to define an I/O pinout configuration or *pinout* that satisfies the requirements of both the PCB and the FPGA designers.

The PlanAhead™ application lets you create I/O port signals (I/O Planning project only), and import I/O port lists from Comma Separated Value (CSV) files, User Constraint Files (UCF), or Register Transfer Level (RTL) files. This allows for early pinout definition to eliminate pinout-related changes that typically happen late in the design cycle.

Often, designers are hindered by a non-optimal pinout that causes further delays when trying to meet timing and signal integrity requirements. By considering the data flow from Printed Circuit Board (PCB) to FPGA die, you can achieve optimal pinout configurations quickly, thus reducing internal and external trace lengths as well as routing congestion.

I/O Pin Planning Methodology

I/O pin planning is a complex process involving design groups that include PCB designers, FPGA designers, and the System designer; each with their own specific set of concerns and requirements.This chapter focuses on using the PlanAhead tool environment to perform device exploration and I/O pin planning and related tasks.

For more information about I/O pin planning methodology, see the *Pin Planning Methodology Guide (UG792)*, cited in Appendix E, Additional Resources.

I/O Planning Stages

The PlanAhead tool facilitates I/O planning at different stages of the design process. As the design progresses, more information becomes available, enabling more complex rule checking as the design is synthesized and implemented.

Proper I/O assignment can depend on how the clocks are configured; assigning I/Os and clock logic often go together. For the I/O placement DRCs to account for clocks, you must have a Synthesized Design. Whenever possible, it is optimal to perform I/O assignment with a Synthesized Design.

The final validation of an I/O pin and clock configuration is to implement the design. Proper clock resource validation can require full implementation of all clocks.

The PlanAhead tool lets you do I/O Planning starting with an empty project, moving to RTL source files, synthesized netlist, and finally working in an Implemented Design.

The kinds of work you can do in each step of the design process varies because early in the process, some data is missing; consequently, analysis is an estimate only, and later in the process, some data is decide, consequently, design changes can be restricted.

The following briefly describes the design stages:

1. **Creating I/O Pin Planning Projects**

   You can create an empty project to enable early device exploration and I/O port configuration. Create I/O ports manually or import them from CSV or UCF inputs. Export device and I/O port assignments for use later in the design process. You can also migrate an I/O Pin Planning project into an RTL sources project when the port definitions and pin assignments are resolved. See Migrating to an RTL Design, page 275.

2. **Elaborating and Checking RTL Source Files**

   You can perform I/O planning in a PlanAhead application RTL sources project. In an Elaborated Design the tool provides basic DRC checks. To check clock logic, validation with a Synthesized Design is recommended.

3. **Synthesizing Designs**

   You can perform I/O planning after synthesis in the Synthesized Design. Because all clocks are determined and the tool has visibility into all clocks, the PlanAhead tool performs a more thorough validation. Whenever possible, perform I/O assignment using a Synthesized Design.

4. **Implementing the Design and Final I/O Validation**

   The Design must be fully implemented to ensure a legal I/O pinout. Examine the NGDBuild and MAP reports for I/O and clock-related messages. Only ISE® Implementation tools have sign-off DRCs.

Configuring and Placing I/O Ports

You can create or import ports into the I/O Pin Planning project. Configure ports by setting I/O standard, drive strength, and slew type for ports. See Configuring I/O Ports, page 257.

You can place a `prohibit` property on individual package pins or I/O banks to prevent I/O assignment to them. See Prohibiting I/O Pins and I/O Banks, page 260.

To perform I/O port placement use one of the following:

- Group I/Os together into interfaces to easily identify or select them for placement. See Creating I/O Port Interfaces, page 261.

- Drag groups of I/O ports and assign them to placement sites in either the Package view or Device view using one of three I/O placement modes. See Placing I/O Ports, page 263.

The cursor tool tip provides information about the number of ports being placed, and whether the placement site under the cursor is a legal site for the port. See Disabling or Enabling Interactive Design Rule Checking, page 262 for more information.

To place unplaced I/Os or any selected group of I/Os automatically, select **Tools > I/O Planning > Autoplace I/O Ports**. The command obeys I/O bank rules, differential pair rules, and global clock pins, and places as many of the I/O Ports as is possible. This functionality is available for certain device architectures only, and you must have a synthesized netlist for the available rules to be applied. See Automatically Placing I/O Ports, page 267.

The PlanAhead tool attempts to maintain correct assignment rules by:

- Placing differential pair ports into proper pin pairs

- Grouping and placing GTXs together with their I/O buffers to ensure proper resource assignment on the device

The PlanAhead tool uses interactive and batch DRCs to ensure legal I/O placement. See Validating I/O and Clock Logic Placement, page 273.
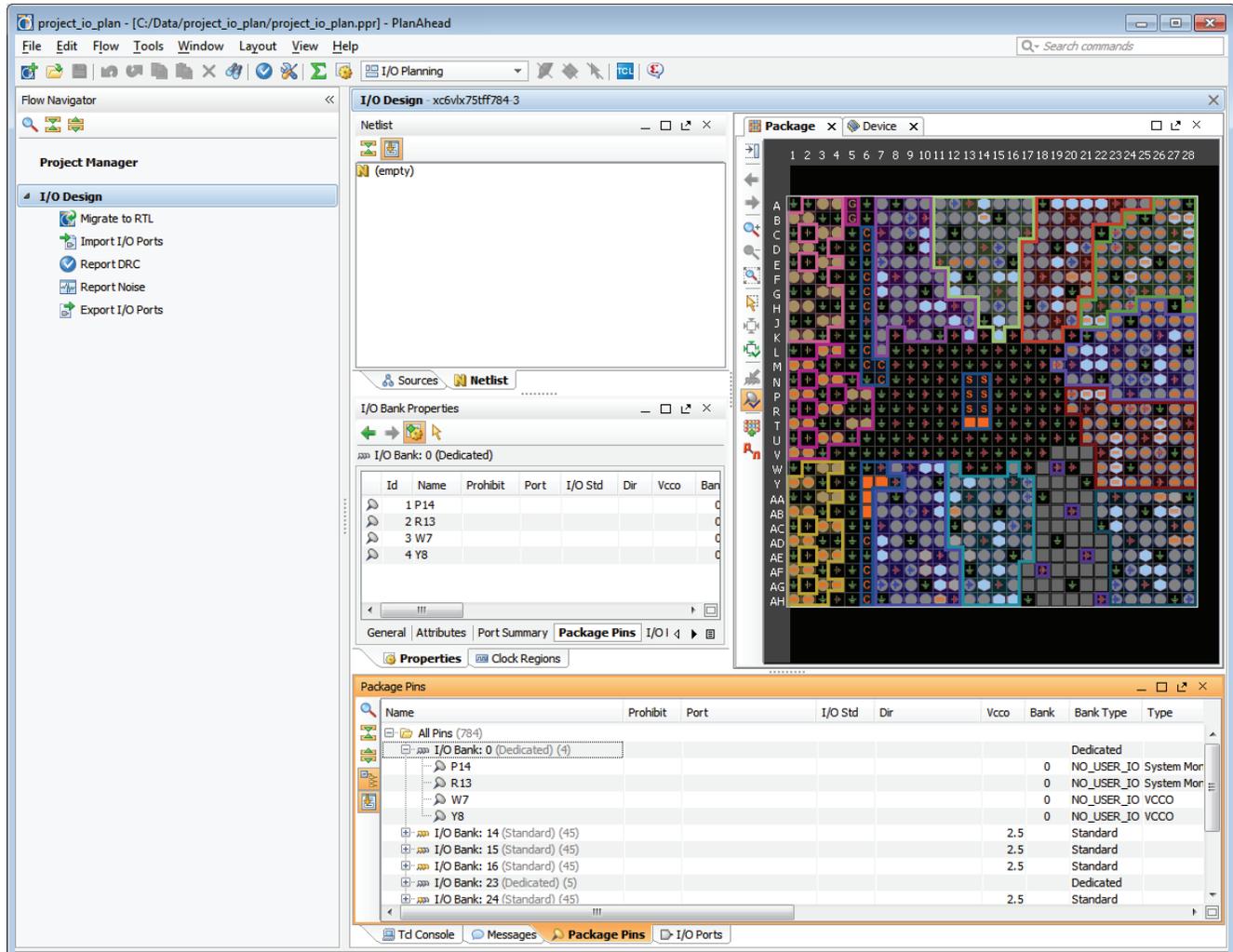
Figure 8-1: **I/O Planning Environment**

Using the I/O Planning view layout

You can use the I/O Planning view layout in the Elaborated Design, Synthesized Design, and Implemented Design environments. The view layout uses both Device and Package views. Other views provide additional I/O information: the Clock Resources and Clock Regions view, the Package Pins view, the I/O Ports view, and the Properties view.

There are two ways to launch the I/O Planning view layout:

- Select the I/O Planning layout in the Layout menu or from the Layout selector.

- Create a new Pin Planning project using the New Project wizard.

The following references in Chapter 4, Using the Viewing Environment apply to the I/O Planning environment:

- Using the Device View, page 136

- Using the Package View, page 142

- Using the I/O Ports View, page 161

- Using the Package Pins View, page 163

Viewing Device Resources

The Device and Package views display a graphical representation of the device and placed logic resources. When any logic object or device site is selected in any view, information displays in the Properties view. The Properties view often has tabbed panes to display various types of information as shown in Figure 8-2.

To view properties for a selected object, display the Properties view. If the Properties view is not visible, select **Window > Properties** from the main menu.

You can search for specific objects or device sites by using the **Find** command. There are a variety of searchable object types and robust filtering capabilities to search the device or design for specific objects. You can select objects directly from the Find Results view; for more information, refer to Using the Find Results View in Chapter 4.

Package Pin Properties

You can select pins or I/O banks in the Package view and see the corresponding details in the Properties view, as shown in the Figure 8-2.
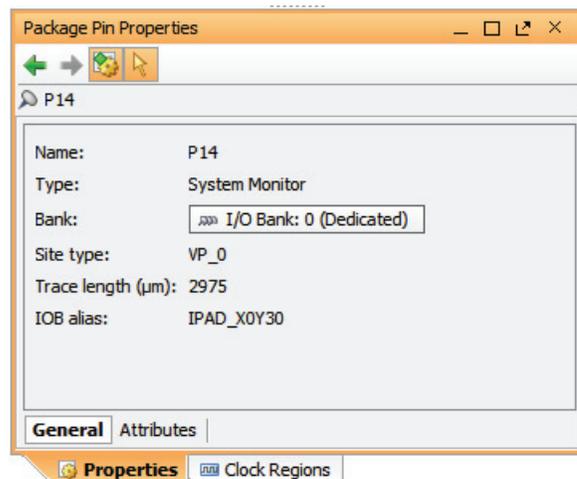
Figure 8-2: **Package Pin Properties**

Clock Region Resources and Statistics

The Clock Regions view allows easy selection of the clock regions. Selecting a clock region highlights the related I/O banks and regional clock resources as shown in Figure 8-3.
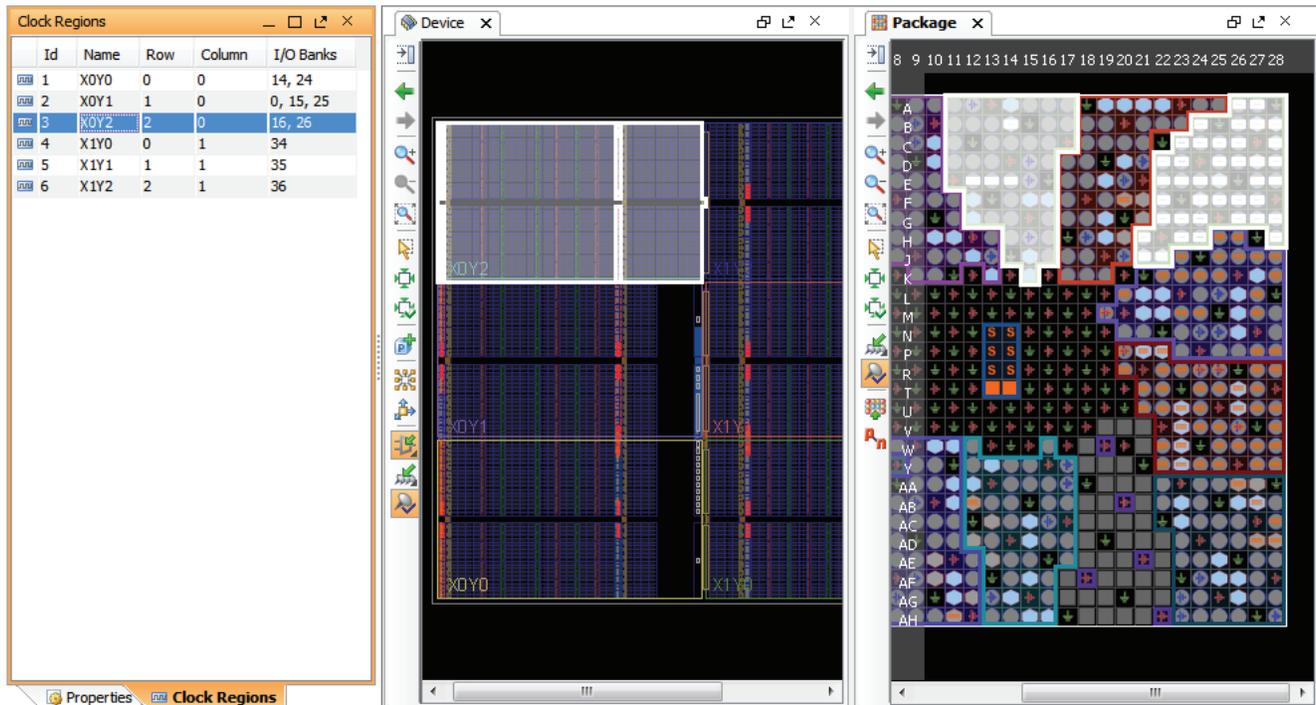
Figure 8-3: **I/O Planning Clock Region Sources**

The Properties view displays the properties for the selected clock region. Selecting the **Statistics** tab in the Clock Region Properties view displays the resource statistics available within the clock region as well as the logic content of the selected clock region.

Click the **Resources** tab to locate device clock resources to for logic assignment, as shown in Figure 8-4, page 250.

When you selecting an object in the Clock Regions Properties view, that object is cross-selected in another open view, such as the Device view.

The Clock Resources view also provides a view of available clock resources to aid in planning and placing elements of global and regional clock trees; see Using the Clock Resources View, page 269.
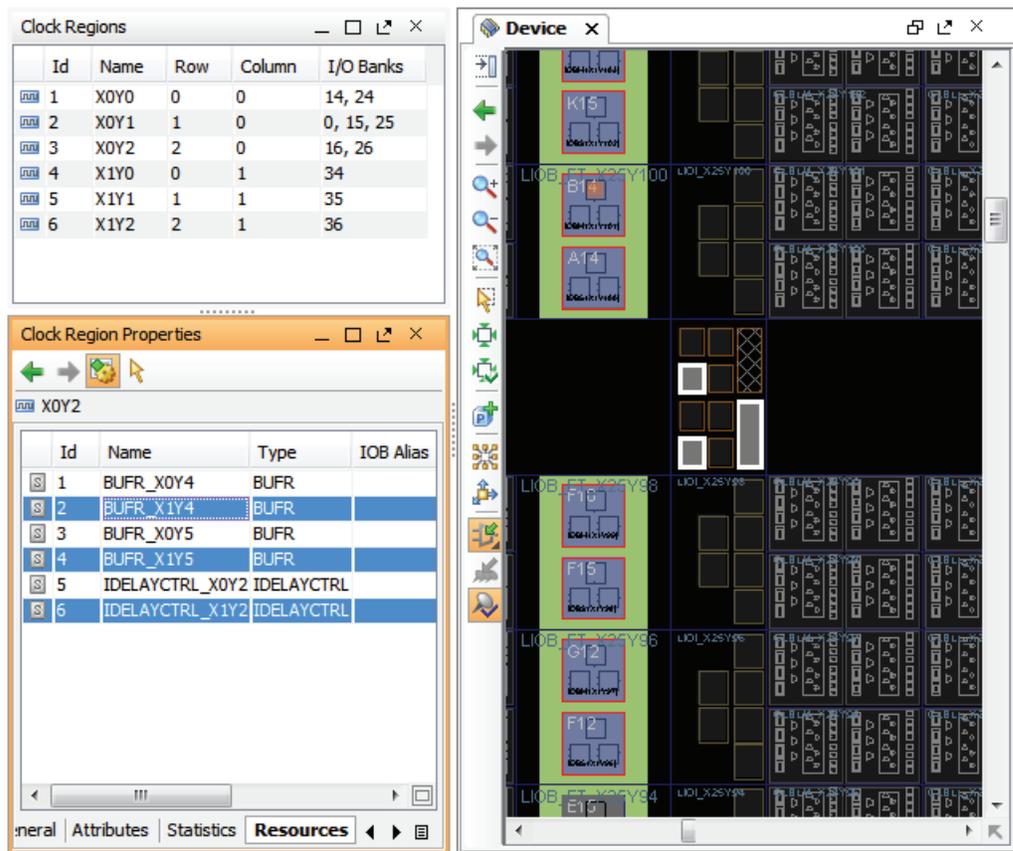
Figure 8-4: **Viewing Clock Region Resources**

I/O Bank Resources

You can select I/O resources in any of the I/O Planning views and their corresponding data is highlighted in all other views as shown in Figure 8-3, page 249. This provides a visual indication of the relationship between the physical package and the internal die.

Use the following steps to access information about a specific I/O bank:

1. In the Package Pins view, select one of the I/O banks.

   The I/O Bank Properties view will be opened.

2. In the I/O Bank Properties view, click the various tabs at the bottom to see the different types of information available.

Figure 8-5, page 251 shows the I/O bank Properties view.
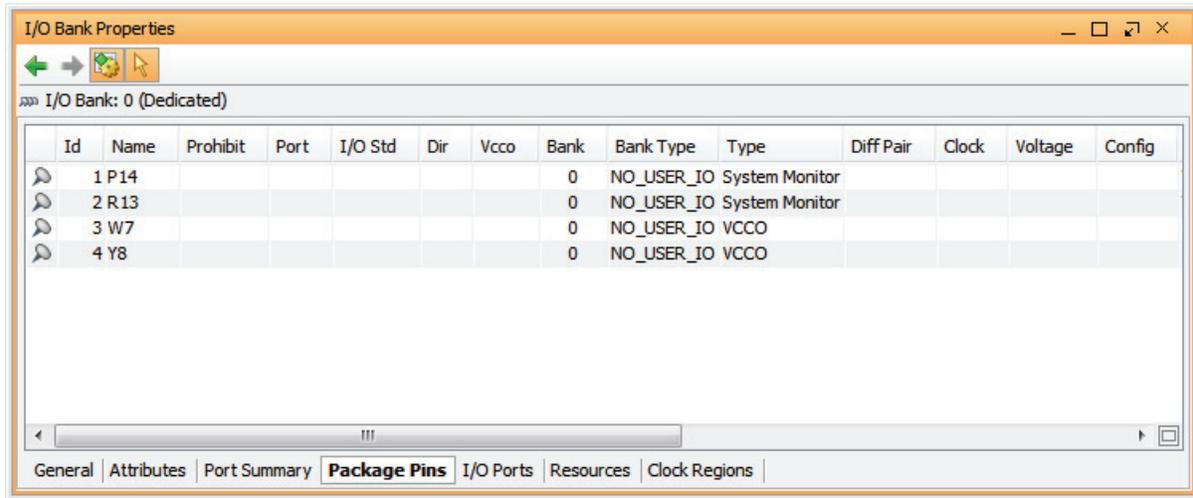
Figure 8-5: **Displaying I/O Bank Properties**

Multifunction Pins

The Package Pins view contains a variety of data types that display in spreadsheet-like columns. The view can be flattened, filtered, and sorted. You can move, hide, and configure columns to display and compare the various multifunction pins. You can edit cells which have editable values directly by entering text or selecting a value from drop-down menus. Figure 8-6 shows an example of the Package Pins view.
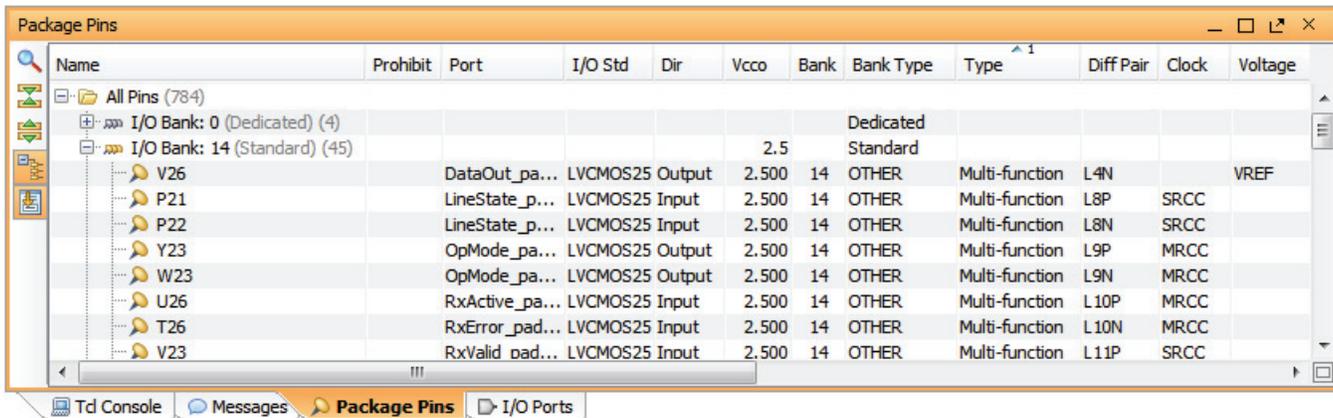
Figure 8-6: **Package Pin View**

The Type column identifies multifunction pin types. Other columns contain information about logic or configuration modes involving multifunction type pins.

The Package view identifies multifunction pins using specific symbols representing their available function:

- Clock capable pins are hexagon shaped
- Vref pins display with a small power icon

The Package View options command provides a legend of the icons used for multifunction pins. See Setting Package View Options, page 144 for more information.

Designs that contain Gigabit Transceivers (GTs), memory controllers, or PCI™ logic, have information in the Package Pin table that identifies conflicting multifunction pins.

Many device configuration modes use multifunction pins. Select **Tools > I/O Planning > Set Configuration Modes** to set the required device configuration modes. For more information, see Setting Device Configuration Modes, page 253.

When you set the device configuration mode, the pin definition of the multifunction pin displays in the Config column of the Package Pins view.

Defining Alternate Compatible Parts

The PlanAhead tool lets you select compatible devices for the design, to allow you to retarget the design to alternate Xilinx FPGAs if necessary.

Set Part Compatibility checks that I/O pin assignments are defined to work across selected alternate devices. Compatible Xilinx devices are selected in the same package as the current target part to preserve as much of the I/O assignment as possible.

To define an alternate compatible part:

1. Select **Tools > IO Planning > Set Part Compatibility**.

   The command locates alternate parts available in the same package as the current project part. Compatible devices available in the same package display in a list, as shown in Figure 8-7.
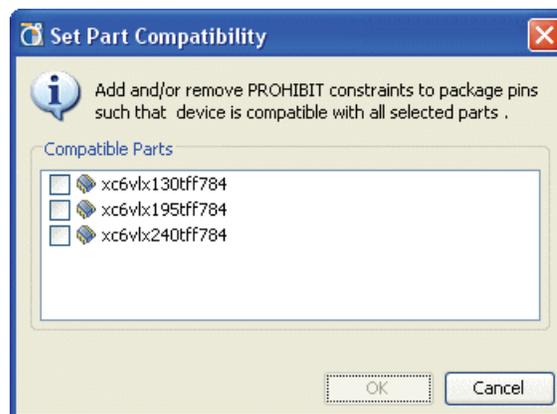
Figure 8-7: **Select Alternate Compatible Parts**

2. Select the alternate parts.

   The PlanAhead tool identifies the pins that are common to all selected alternate parts, and assigns `Prohibit` constraints to pins that are not common to all devices. The number of pins available for placement might be reduced as you select additional alternate parts.

The PlanAhead tool automatically prohibits signals from being assigned to any unbonded pins in the selected alternate devices. A dialog box displays the number of prohibited package pins. You can view prohibits in the Package, Package Pins, and Device views.

Note: If you are defining an alternate compatible part for a Spartan®-6 LX25 or LX25T device, pins that are bonded are prohibited because of differences in the clocking topology between this device and alternate compatible parts in the same package. Refer to AR 34885 for more information.

Setting Device Configuration Modes

The PlanAhead tool provides information about device configuration modes, and you can set any number of them, as shown in Figure 8-8, page 253.
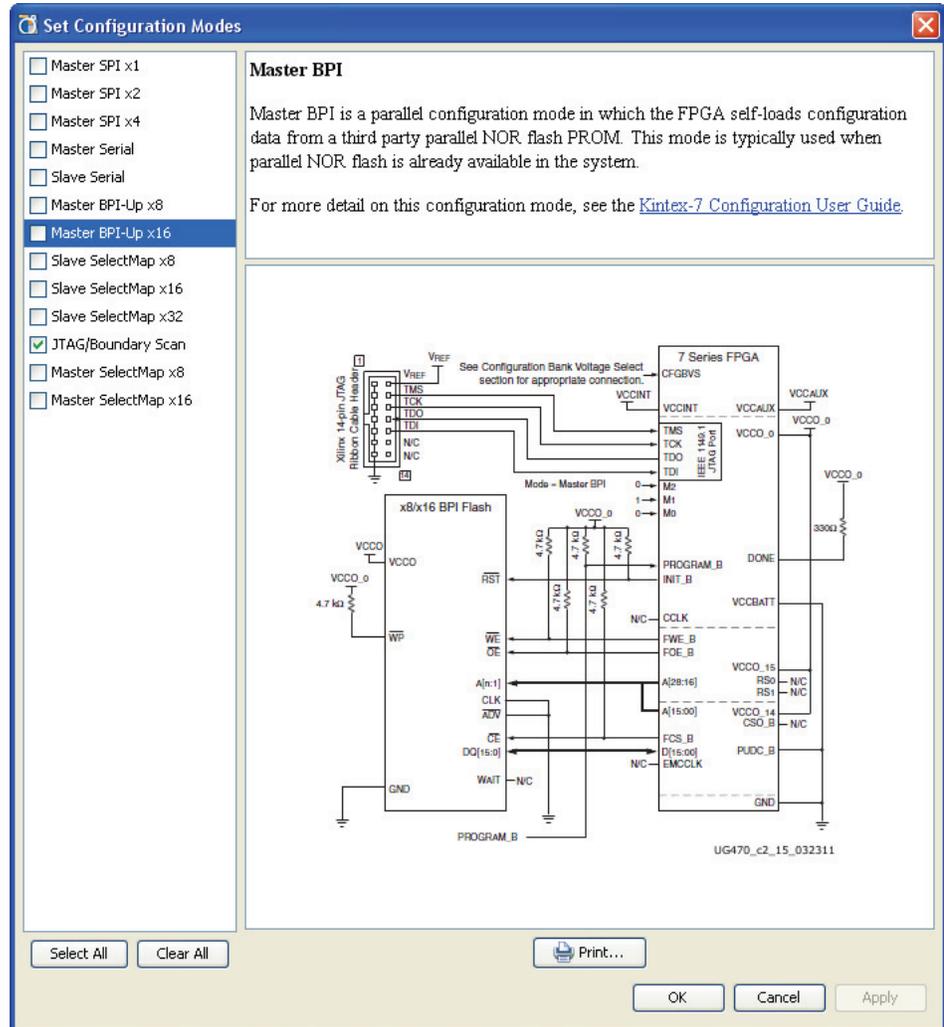
Figure 8-8: **Set Configuration Modes Dialog Box**

To set the device configuration modes:

1. Select **Tools > IO Planning > Set Configuration Modes**.

    •   Click any configuration mode to view information, including the schematic.

    •   Click **Print** to print a copy of the configuration diagram.

2. Select the configuration modes you want, and click **OK**.

    When you set the configuration modes the associated I/O pins display in the **Config** column of the Package Pins view. The PlanAhead tool prompts you to automatically sort the Package Pins view using the **Config** column header to sort the pins as shown in Figure 8-9, page 254.

    In addition, any package pins that are not common to the various configuration modes you selected will be disabled with a PROHIBIT property, as shown in Figure 8-9.
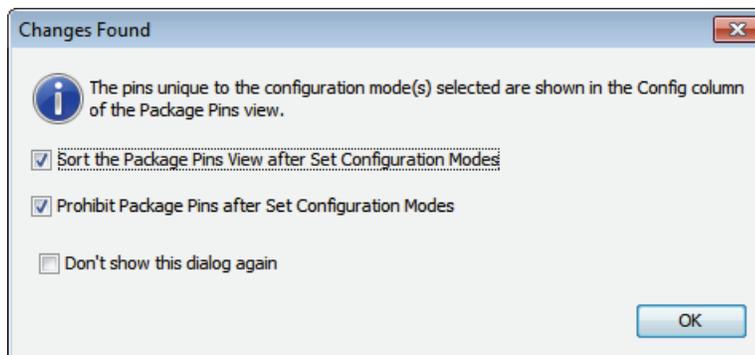
Figure 8-9: **Sort Config Pins**

For more information about analyzing how the configuration modes might conflict with other multi-function pins, refer to Multifunction Pins, page 251.

Defining and Configuring I/O Ports

To create and configure I/O ports in an empty Pin Planning project use the I/O Planning environment.

Importing I/O Ports

The PlanAhead tool supports importing UCF or CSV format files into an empty Pin Planning project at the time you create the project, or later using the file import capability.

Use RTL files or headers to create an RTL source project for I/O pin planning, then add more complete RTL source files to the project later as the design progresses. When you create an RTL-based or synthesized Netlist-based project, the I/O Ports view populates automatically with the I/O ports defined in the design.

You an also convert an I/O Pin Planning project into an RTL design project, turning the I/O ports into a top-level Verilog or VHDL module definition for the design. See Migrating to an RTL Design, page 275 for more information.
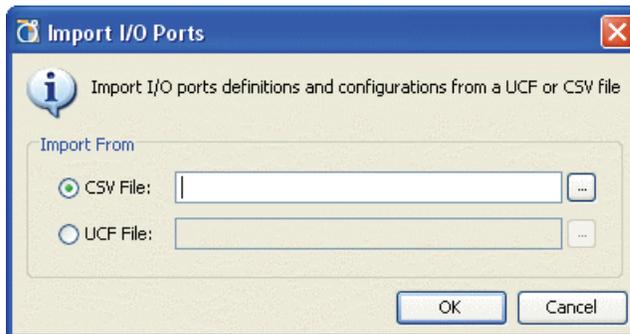
Figure 8-10: **Import I/O Ports Dialog Box**

Importing a CSV Format File

To import an I/O ports list from a CSV:

1. In Flow Navigator, select **Import I/O Ports**, or use the **File > Import > Import I/O Ports** command in the main menu.

   The Import I/O Ports dialog box opens, as shown in Figure 8-10.

2. Select the **CSV File** option, and browse to select the CSV file to import.

   The CSV file format is shown in Figure 8-11.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Top: top	Floorplan: floorplan_1	Part: xc5vsx35tff665-1										
	Generated by: brianj	on: Fri Feb 06 17:28:39 2009											
	Build: PlanAhead v11.1.LR0	by: ECloudInternalUser4	on: Thu Feb 5 20:04:57 PST 2009										
IO Bank	**Pin Number**	**IOB Alias**	**Site Type**	**Min Trace**	**Max Trace**	**Prohibit**	**Interface**	**Signal Name**	**Direction**	**DiffPair Type**	**DiffPair Sign**	**IO Standard**
	P2	OPAD_X0Y5	MGTTXP0_114	34878	40691			TXP_OUT[4]	OUT	P	TXN_OUT[4]	LVDS_25
	W2	OPAD_X0Y7	MGTTXP1_114	41406	48307			TXP_OUT[5]	OUT	P	TXN_OUT[5]	LVDS_25
	B2	OPAD_X0Y13	MGTTXP0_116	63540	74130			TXP_OUT[6]	OUT	P	TXN_OUT[6]	LVDS_25
	G2	OPAD_X0Y15	MGTTXP1_116	55620	64890			TXP_OUT[7]	OUT	P	TXN_OUT[7]	LVDS_25
17	AD16	IOB_X0Y8	IO_L15N_17	80604	94038			DataOut_USB	OUT			LVCMOS25
17	AE15	IOB_X0Y6	IO_L16N_17	89010	103845			DataOut_USB	OUT			LVCMOS25
17	AC21	IOB_X0Y17	IO_L11P_CC_17	47370	55265			DataOut_USB	OUT			LVCMOS25
17	AE16	IOB_X0Y9	IO_L15P_17	78702	91819			DataOut_USB	OUT			LVCMOS25
17	AE21	IOB_X0Y22	IO_L8N_CC_17	63150	73675			DataOut_USB	OUT			LVCMOS25
17	AD20	IOB_X0Y18	IO_L10N_CC_17	62046	72387			DataOut_USB	OUT			LVCMOS25
17	AC23	IOB_X0Y26	IO_L6N_17	58710	68495			DataOut_USB	OUT			LVCMOS25
17	AF17	IOB_X0Y10	IO_L14N_VREF_17	80994	94493			DataOut_USB	OUT			LVCMOS25
17	AD24	IOB_X0Y36	IO_L1N_17	64386	75117			DataIn_USB_0	IN			LVCMOS25

Figure 8-11: **I/O Port List CSV Format**

CSV is a standard file format used by FPGA and board designers to exchange information about device pins and pinout. The PlanAhead tool requires a specific CSV file format for importing I/O pin-related data, as described in I/O Port Lists (CSV), page 403.

You can define differential pairs in the CSV in a number of ways. The PlanAhead tool recognizes differential pairs directly defined with DiffPair Signal and DiffPair Type attributes, and it can infer diff pairs when only one port of the pair is defined in the CSV, or two named nets imply a differential pair. See Defining Differential Pairs in the CSV in Appendix A for more information.

When the PlanAhead tool infers differential pairs, it displays a prompt to confirm the assignment of the pairs, as shown in Figure 8-12.
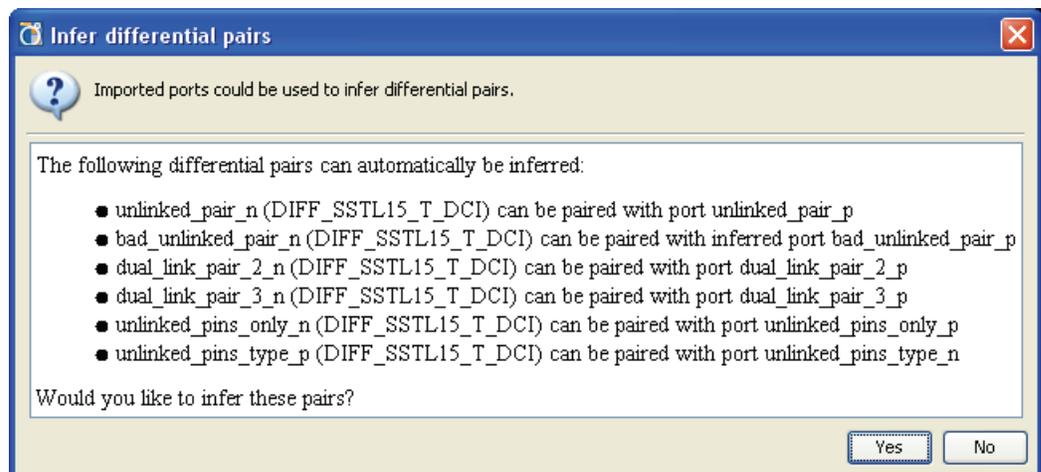
Figure 8-12: **Inferring Differential Pairs**

CSV files can also contain additional information not recognized by the PlanAhead tool. If unrecognized information is found in the imported CSV file, it displays that information in user columns of the Package Pins view for your review and use.

To modify or define values in the user CSV fields, select the **Set User Column Values** from the popup menu in the Package Pins view.

Use the **File > Export > Export I/O Ports** to export a CSV file. Added columns and user-defined values are preserved and exported to the output file.

Importing a UCF Format File

You can import UCF format files as a way to populate the I/O Ports view.

To import I/O port definitions from a UCF, select **File > Import > Import I/O Ports** and select the UCF option to browse to a UCF, as shown in Figure 8-10, page 254.

Because the UCF format does not define port direction, the direction is undefined. To define the I/O port direction, select **Set Direction** from the I/O Ports view. You can also directly modify the direction of a specific I/O port in the I/O Ports view.

See Setting I/O Port Direction, page 258 for more information.

Note: If the imported UCF defines ports uses wildcard (*) syntax, the PlanAhead tool does not expand the wildcards and does *not* import those ports.

Creating New I/O Ports

You can manually define new ports in an I/O Planning project. Refer to Xilinx® device documentation for information regarding voltage capabilities of the device.

To create I/O ports:

1. In the I/O Ports view, select **Create I/O Ports**.

    The Create I/O Ports dialog box opens, as shown in Figure 8-13.
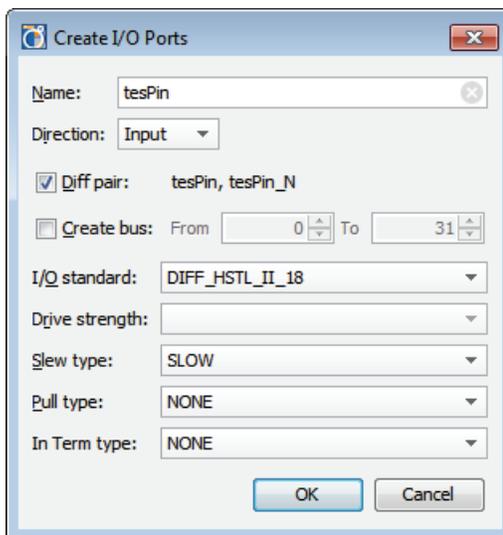
Figure 8-13: **Create I/O Ports Dialog Box**

2. Edit the options:

   • **Name**: Enter the port or bus name to create.

   • **Direction**: Select the port direction.

   • **Diff Pair**: Define differential pair signals or buses. This will create two ports: one with the specified name for the positive side of the differential pair, and one with '_N' appended to the specified name for the negative side.

   • **Create Bus**: Enter bus range for bus creation.

   • **I/O Standard**: Select the I/O Standard constraint.

   • **Drive Strength**: Select the Drive Strength value.

   • **Slew Type**: Select the Slew Type value.

   • **Pull Type**: Select the Pull Type value.

   • Depending on the target part you will see one of the following:

      - **Phase**: The Phase option applies to Virtex®-6 devices only. Enter a phase group or select an existing phase group. A phase group is a logical grouping of ports used in Simultaneous Switching Noise (SSN) calculations to indicate that the set of ports share the same frequency and phase. For more information, see Defining I/O Port Switching Phase Groups in Virtex-6 Devices, page 282.

      - **In Term Type:** Defines the parallel termination attributes of the input signal.

3. Click **OK**.

Configuring I/O Ports

You can configure one or more I/O ports to define I/O Standard, Drive Strength, Slew Type, Pull Type, and In Term. This is useful to configure ports that were imported from CSV or UCF files without the appropriate characteristics. Refer to Xilinx device documentation for information regarding voltage capabilities of the device.

To configure a port or a group of ports:

1. In the I/O Ports view, select the ports.

2. From the popup menu, select **Configure I/O Ports**.

   The Configure Ports dialog box opens, as shown in Figure 8-14.

Figure 8-14:   **Configure Ports Dialog** B**ox**

3. Edit the options:

   • **I/O Standard**: Select the I/O Standard constraint. The tool does not check the I/O Standard when it is assigned. You can assign any I/O Standard to any port, but this could result in errors when running DRC.

- **Drive Strength**: Select the Drive Strength value.

- **Slew Type**: Select the Slew Type value.

- **Pull Type**: Select the Pull Type value.

- Depending on the target part you will see one of the following:

    - **Phase**: The Phase option applies to Virtex®-6 devices only. Enter a phase group or select an existing phase group. For more information, see Defining I/O Port Switching Phase Groups in Virtex-6 Devices, page 282.

    - **In Term Type:** Defines the parallel termination attributes of the input signal.

4. Click **OK**.

Setting I/O Port Direction

To set the I/O port direction, select the I/O ports, buses, or interfaces to be configured, and select **Set Direction** in the I/O Ports view. You can use this command to define a port direction of Input, Output, or In/Out. This command is only available in I/O Pin Planning projects. In RTL source projects, you define the port direction in the RTL source.

You can also edit I/O Ports directly in the I/O Ports view table by clicking in the **Dir** column on a port and changing the direction using the drop-down menu.

Making and Splitting Differential Pairs

To define a differential pin pair in an I/O Planning project:

1. Select any two I/O ports, then select **Make Diff Pair** in the popup menu of the I/O Ports view, as shown in Figure 8-15.

   Note: The Make Diff Pair option is not applicable in RTL-based projects. In RTL-based projects, differential ports must be defined in the source code using appropriate I/O buffer instantiations.
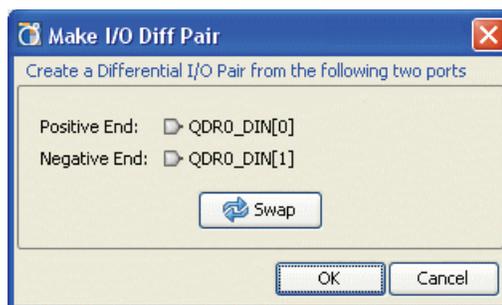
Figure 8-15: **Make I/O Diff Pair**

The two I/O Ports display in the dialog box with Positive End and Negative End assignments made by the tool.

2. To reverse the Positive End and Negative End signals, click **Swap**.

3. Click **OK**.

Use the **Split Diff Pair** command from the popup menu to separate a diff pair into two ports.

Configuring DCI_CASCADE Constraints

The DCI_CASCADE constraint links two or more adjacent I/O Banks together for DCI reference voltage purposes. The I/O bank with the DCI reference voltage is the *master*. All other I/O banks in the cascade are *slaves*. All banks in a cascade must be in the same I/O column of the device.

You can configure the DCI_CASCADE constraint for Xilinx 7 series FPGAs, Virtex®-6, and Virtex-5 devices.

To set the constraint:

- Pre-select the I/O banks before running the command.

  or

- Select the I/O banks in the command dialog box.

For more information about the DCI_CASCADE constraint, see the *Constraints Guide (UG625)*, cited in Appendix E, Additional Resources.

Creating a DCI_CASCADE Constraint

To create a DCI_CASCADE constraint:

1. Select the I/O banks to configure.

2. Select **Create a DCI Cascade** from the popup menu of the Package Pins view, or the Package view.

   The DCI Cascade Editor opens. Preselected I/O banks appear in the dialog box.
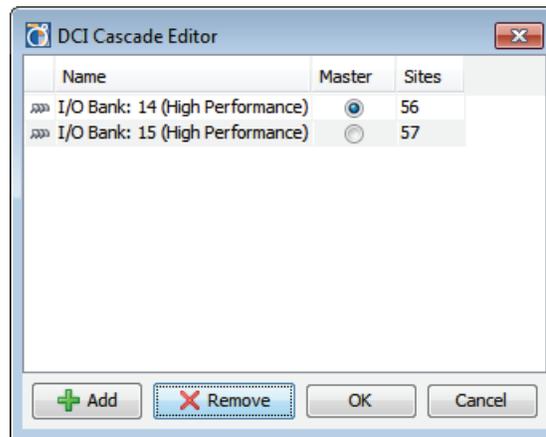
Figure 8-16:   **Creating a DCI Cascade**

3. To include additional I/O banks, click **Add**.

   Select I/O banks from the same column on the device. The software does not check this interactively when you create the DCI_CASCADE constraint, but can check for errors when you run DRC.

4. Select an I/O bank to be the Master.

5. Click **OK**.

As I/O banks are selected, they are highlighted in the other views.

The DCI Cascades are displayed in the Physical Constraints View. See Figure 8-17, page 260. If you select a DCI cascade in the Physical Constraints view, the PlanAhead tool selects the associated I/O banks.
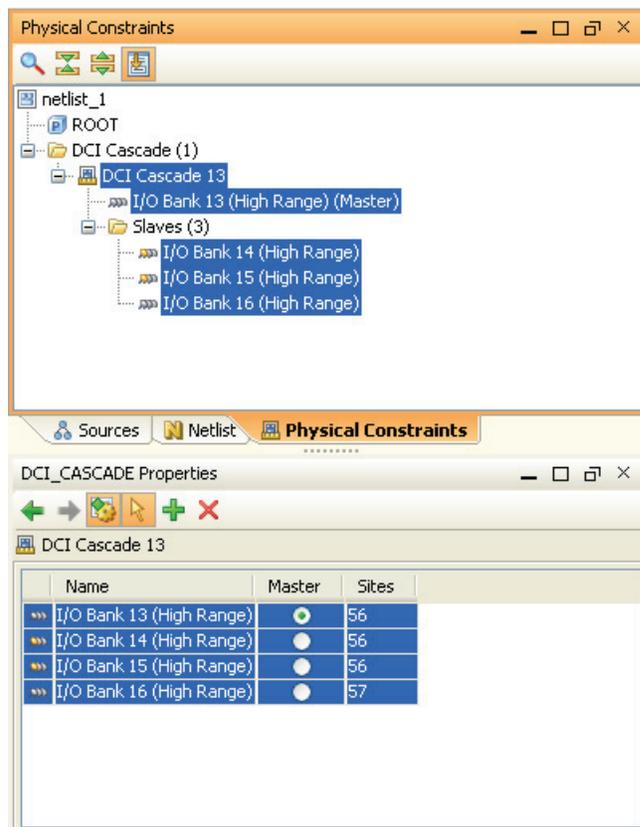
Figure 8-17: **Viewing DCI Cascades**

Modifying or Removing DCI Cascade Constraints

You can modify DCI Cascades by selecting the DCI Cascade in the Physical Constraints view, and using the DCI_CASCADE Properties view. The actions are:

- To change the Master, select a different I/O Bank and assign it as the Master.

- To remove I/O banks from the DCI Cascade, select the I/O banks in the DCI_CASCADE Properties view, then click **Delete I/O Banks**.

- To include additional I/O banks in the DCI Cascade, select them in the DCI_CASCADE Properties view and click **Add I/O Banks**. The Add I/O Banks dialog box displays and you can select new I/O banks. The newly-selected I/O banks also highlight in the other views.

- To remove DCI Cascade constraints, in the Physical Constraints view, select the constraint, then click **Delete**.

- To save all changes, in the DCI_CASCADE Properties view, click **Apply**.

Prohibiting I/O Pins and I/O Banks

The I/O Planning view layout provides an interface to selectively prohibit port placement onto individual I/O pins, groups of I/O pins, or I/O banks. You can select and prohibit pins in the Device, Package, and Package Pins views.

To prohibit I/O pins or I/O banks:

1. Select the I/O pins or I/O banks in the Device, Package, or Package Pins view.

2. Select **Set Prohibit** from the popup menu.

The PlanAhead tool places an slashed circle on the prohibited pins in the Package view, as shown in Figure 8-18, and places a checkmark in the Prohibit column of the Package Pins view.

Figure 8-18: **Setting Prohibits on Package Pins**

Creating I/O Port Interfaces

To group multiple ports or buses together, you can create an interface. This aids in pin assignment by treating all of the interface ports as one group. Assigning all of the pins simultaneously helps condense and isolate the interface for clock region or PCB routing. Also, it is easier to visualize and manage the signals associated with a particular logic interface.

To create an interface:

1. In the I/O Ports view, select the signals to group together.

2. From the popup menu, select **Create I/O Port Interface,** as shown in Figure 8-19.
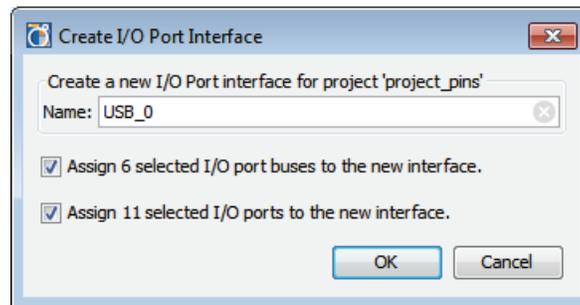
Figure 8-19: **Create I/O Ports Interface**

3. Enter a name for the interface and adjust assignment selection, then click **OK**.

The interfaces appear as expandable folders in the I/O Ports view. To add additional I/O ports to the interface, select them in the I/O Ports view, and drag them into the Interface folder, as shown in Figure 8-20.

Figure 8-20: **Manage I/O Port Interfaces**

To include additional I/O ports in an interface:

1. Select a port or bus.

2. From the popup menu, select **Assign to Interface**.

3. Select the target interface to which to add the I/O ports.

To remove I/O ports and interfaces:

1. Select a port or interface.

2. From the popup menu, click **Unassign from Interface**.

To delete interfaces, select an interface, and select **Delete** from the popup menu, or press the **Delete** key.

Disabling or Enabling Interactive Design Rule Checking

The PlanAhead tool checks to ensure a legal pin out; however, only ISE® implementation tools provide complete signoff DRCs. Therefore, you need to run your design through the ISE pinning process to ensure final legal pinouts.

The interactive I/O placement routines check common error cases. You can toggle this capability on and off using:

*   In the Device or Package view toolbar menu, click **Autocheck I/O Placement** button, or

*   Use the **Tools > Options** command from the main menu, and click the **General** button on the left hand side of the dialog box. This allows you to set the **Automatically Enforce Legal I/O Placement** option on the right side of the dialog box.

When you enable automatic checking, the tool does not allow placement of I/O ports on pins that cause a design issue. In **Place I/O Ports Sequentially mode**, if you attempt to place an I/O Port on a problematic pin, a tool tip display that describes why the I/O Port is not able to be placed. The interactive DRC checks are enabled by default.

Note: Many of these checks can run only when a synthesized netlist of the full design is loaded.

The interactive I/O placement rules include:

*   Prohibiting:

    *   Placement on noise sensitive pins associated with Gigabit Transceivers (GTs). I/O package pins that are potentially noise-sensitive.

- • I/O standard violations.
- • Ensuring:
  - • I/O standards are not used in banks that do not support them.
  - • Banks do not have incompatible VCC ports assigned.
  - • Banks that need VREF ports have free VREF pins.
  - • Proper assignment of global clocks and regional clocks (only with an imported EDIF/NGC netlist and UCF).
  - • Input and High Drive outputs only go to capable pins (Spartan®-3 devices only).
  - • Differential I/O ports are set to the proper sense pin.
  - • No output pins are placed on input-only pins.

It is recommended that you begin your I/O port placement with DRCs enabled. I/O Port and Clock Logic and Placement DRC Rule Descriptions, page 419, lists the I/O-related DRCs.

Placing I/O Ports

The I/O Planning view layout provides a variety of ways to assign I/O Ports to package pins. You can select individual I/O ports, groups of I/O ports, or interfaces in the I/O Ports view, and assign them to package pins in the Package view or I/O pads in the Device view.

Note: Zynq™ devices include processor sub-system IO pins (PSSIO) that are read-only. These pins are not configurable as programmable logic (PL) pins on the Zynq device, and therefore cannot be assigned ports in an I/O Pin Planning project. PSSIO pins are configured as processor system (PS) pins, and imported from XPS. Refer XPS Help for more information.

Placing I/O Ports Sequentially
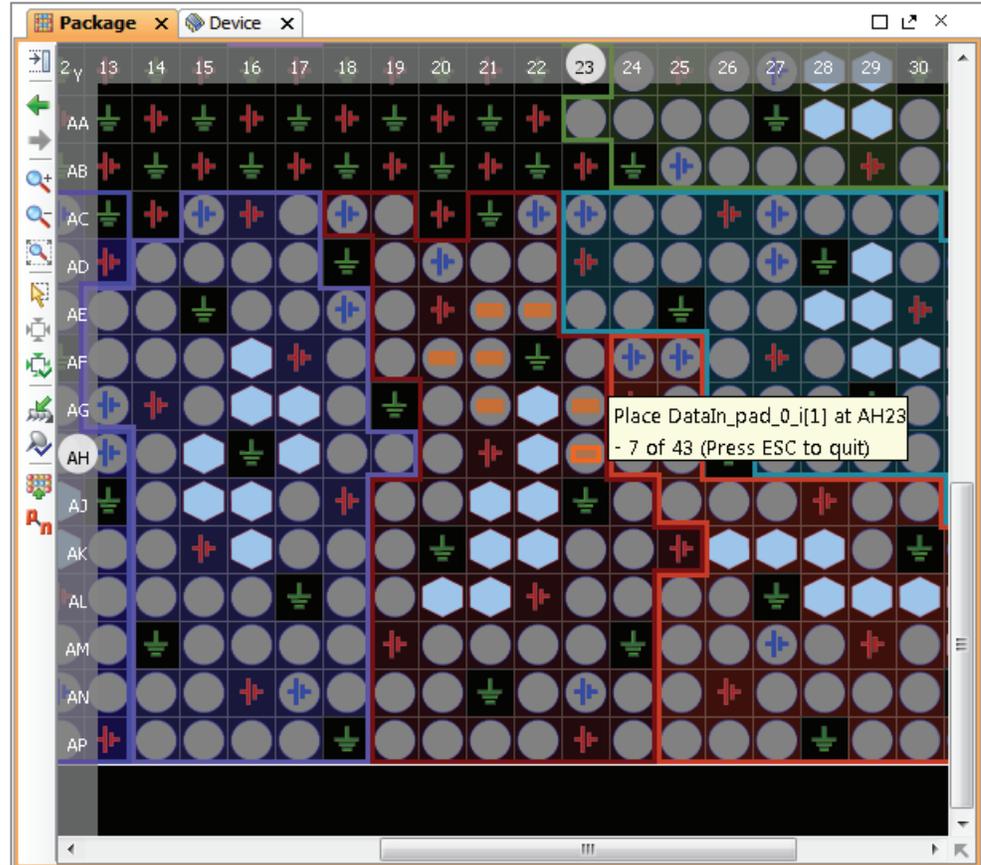
Figure 8-21: **Placing I/O Ports Sequentially**

To place I/O ports sequentially:

1. In the I/O Ports view, select an individual I/O port, a group of I/O ports or interfaces.

2. Use one of the following commands:
   - In the I/O Ports view, from the popup menu, select **Place I/O Ports Sequentially**.
   - In either the Package view or the Device view, click the **Place Ports** button on the toolbar menu, and select the **Place I/O Ports Sequentially** mode.

   The first I/O port in the group is attached to the cursor when you move it over a package pin or I/O pad. A tool tip displays the I/O port and package pin names.

3. To assign an I/O port, click a pin or a pad.

   Figure 8-21, page 264 shows a sequential I/O port placement.

4. If more I/O ports are selected, the command is continued. The cursor drags the next I/O ports, and so on until all of the I/O ports are placed or until you press **Esc**.

The PlanAhead tool assigns ports in the order that they appear in the I/O Ports view. You can adjust the assignment order by applying sorting techniques in the I/O Ports view prior to assignment.

Placing I/O Ports into I/O Banks
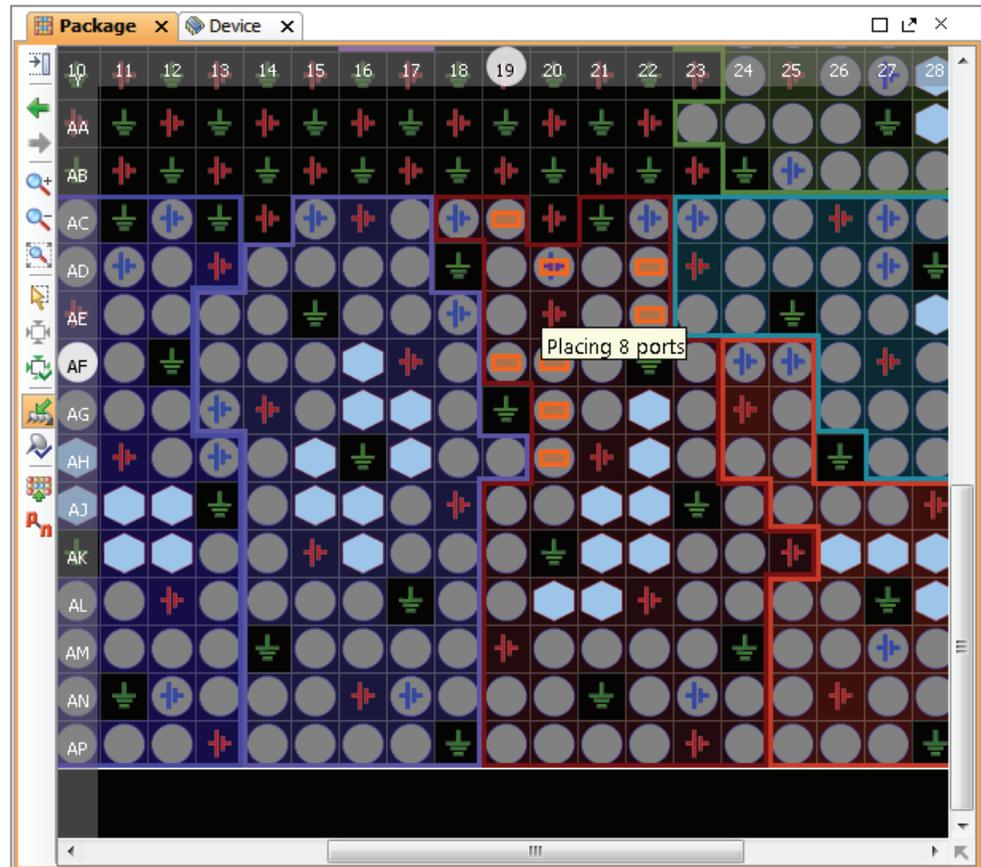
Figure 8-22:   **Placing I/O Ports in I/O Banks**

To place I/O ports into I/O banks:

1.   In the I/O Ports view, select an individual I/O port, a group of I/O ports, or Interfaces.

2.   Use one of the following commands:

   •   In the I/O Ports view, from the popup menu, select **Place I/O Ports in an I/O Bank**.

   •   In either the Package view or the Device view, click the **Place Ports** button on the toolbar menu, and select the **Place I/O Ports in an I/O Bank** mode.

   The group of I/O ports is attached to the cursor when it is dragged over a package pin or I/O pad. A tool tip displays how many pins can be placed in the selected I/O bank.

3.   Click on a pin or pad to assign the selected I/O ports. Figure 8-22, page 265 shows an I/O pad.

4.   If more I/O ports are selected than fit in the I/O bank, the PlanAhead tool places as many as possible in the selected I/O bank, then lets you select another I/O bank into which to place the remaining ports. The cursor drags the remaining I/O ports to the next selected I/O bank, and so on until all of the I/O ports are placed, or you press **Esc**.

   •   Moving the cursor within the Package view actively displays the I/O pin coordinates on the top and left sides of the view.

   •   Additional I/O pin and bank information displays in the Status Bar located at the bottom of the environment.

   •   The active object being reported is highlighted in the Package view.

- Holding the cursor over the Package view invokes a tool tip that displays the pin information.

Ports are assigned in the order they appear in the I/O Ports view. The assignment order can be adjusted by applying sorting techniques in the I/O Ports view prior to assignment.

Port assignment to device resources is also driven from the initial selection from the I/O bank. Selecting a pin at one end of an I/O Bank results in a continuous bus assignment across the I/O bank.

The PlanAhead tool also keeps track of PCB routing concerns for buses. Pin ordering during assignment attempts to keep the bus bits vectored within the assignment area. You can customize assignment patterns to address other bus routing concerns.

Placing I/O Ports in a Defined Area
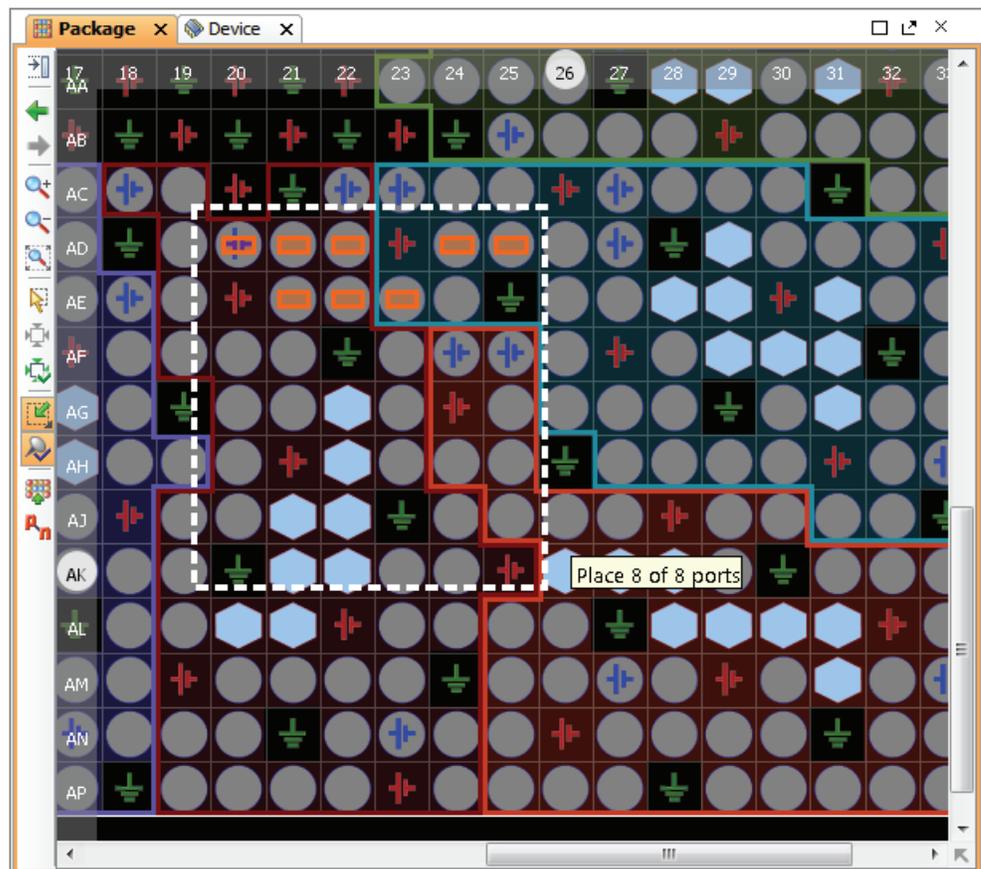
Figure 8-23: **Placing I/O Ports in an Area**

To place I/O Ports into a defined area:

1. In the I/O Ports view, select individual I/O ports, groups of I/O ports or interfaces.

2. Use one of the following commands:
   - In the I/O Ports view, from the popup menu, select **Place I/O Ports in Area**.

3. In either the Package view or the Device view, click the **Place Ports** button on the toolbar menu, and select the **Place I/O Ports in Area** button.

   The cursor turns into a cross symbol which indicates that you can define a rectangle for port placement.

4. In either the Package view or the Device view, draw a rectangle to define the assignment area, as shown in Figure 8-23.

5. If you select more I/O Ports than fit in the defined area, the command is continued. The cursor continues to display as a cross to draw another area to place the remaining I/O ports until all of the I/O ports are placed, or until you press **Esc**.

Ports are assigned in the order that they appear in the I/O Ports view. You can adjust the assignment order by applying sorting techniques in the I/O Ports view prior to assignment.

The direction in which you draw the rectangle dictates the I/O ports assignment order. I/O ports are assigned from the inside pin of the first rectangle coordinate selected. Creative definition of the area rectangles can provide useful pinout configurations from a PCB routing perspective.

Swapping and Moving Previously Placed I/O Ports

Working interactively, you find occasions where you need to move or swap I/O ports that have already been assigned. You can select two placed I/O ports and swap their locations with each other as follows:

1. Select two I/O ports from any of the available views.

   Hold down the **Ctrl** key while clicking to select multiple ports.

2. Right-click and select **Swap Locations.**

If you swap two ports which are not yet fixed (if you are viewing the placement in the Implemented Design), the act of swapping them also causes them to be fixed and the resultant LOC constraint are written to the design constraint file.

You can also move a port, or groups of ports, by selecting them and dragging them from one location to another. When you move a group of ports from one I/O bank to another, the PlanAhead tool automatically finds suitable locations for the selected ports. This is similar to using the **Place I/O Ports in an I/O Bank** command.

Automatically Placing I/O Ports

Note: The Auto-place I/Os function is only available for Xilinx Virtex-6 devices.

The Auto-place I/O Ports command can automatically assign all I/O ports to package pins, or any unplaced or selected I/O ports. The autoplacer obeys I/O standard and differential pair rules, and places global clock pins appropriately.

To automatically assign I/O ports:

1. In the I/O Ports view, choose the I/O ports, to place.

2. Select:

   • **Tools > I/O Planning > Auto-place I/O Ports**, or

   • In the I/O Ports view, **Auto-place I/O Ports**

   The Autoplace I/O Ports wizard opens, as shown in Figure 8-24, page 268.

3. Select the group of I/O ports to place, and click **Next**.
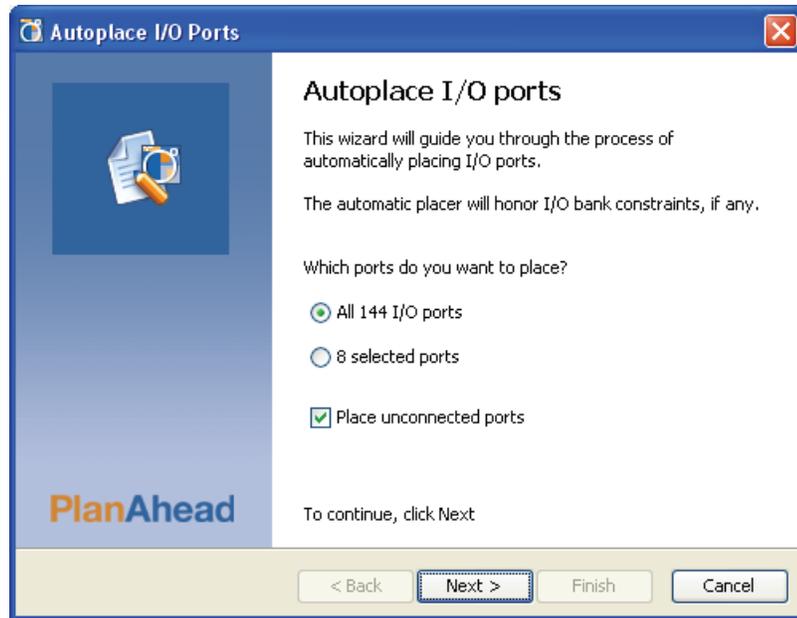
Figure 8-24: **Autoplace I/O Ports Wizard**

4. If you select I/O ports that have already been assigned to package pins, the dialog box opens, as shown in Figure 8-25.

5. Select the I/O ports to place, click **Next**, and then **Finish** in the Summary page.
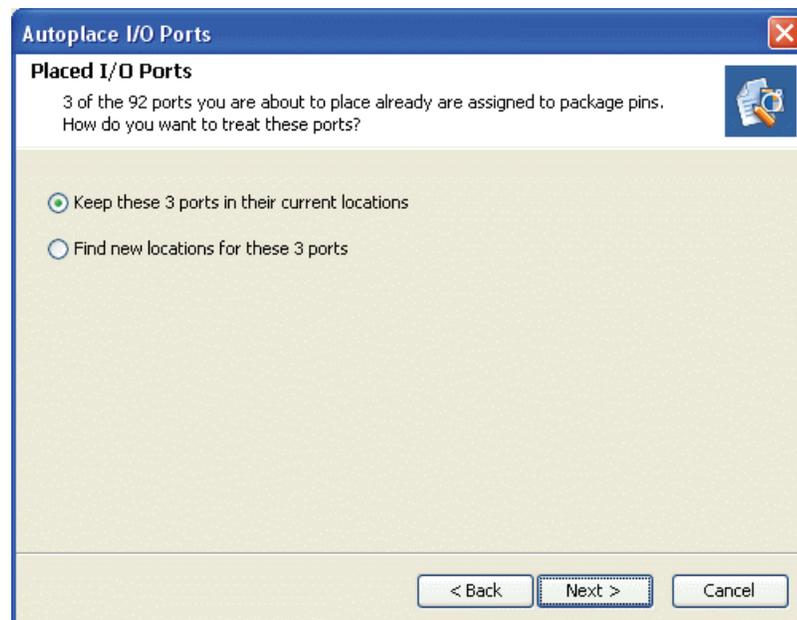
Figure 8-25: **Autoplace I/O Ports Wizard**

Placing Gigabit Transceiver I/O Ports

To better manage Gigabit Transceivers (GTXs), the I/O Planning views group the two related I/O diff pairs and the GTX logic object automatically during selection, placement, and moving.

When using **Place I/O Ports Sequentially**, the GTX objects are selected as one object and move together, which prohibits illegal assignment of the GTX resources.

If the online DRCs are enabled, the noise sensitive I/O pins surrounding the GTXs are prohibited automatically during port placement. Refer to Disabling or Enabling Interactive Design Rule Checking, page 262.

Removing I/O Placement Constraints

You can remove placement constraints by selecting placed logic and then selecting the **Unplace** command from the right mouse popup menu.

Refer to Working with Placement LOC and BEL Constraints in Chapter 10, for information on selectively removing placement constraints.

Placing Clock Logic

You can manually place global and regional clock-related logic, such as `BUFGs`, `DCMs` or `MMCMs`, `BUFRs`, and `IDELAYCTRLs`, using the Clock Resources view as discussed in Using the Clock Resources View, page 269. You can also manually place clock logic in the Device view. Appropriate logic sites are displayed in the Device view for all device-specific resources.

To locate logic instances for placing onto the device, such as `BUFGs`:

1. Select **Edit > Find**.

2. Specify Instances in the **Find** field, and define the criteria as needed to locate the specific logic instance or instances of interest.

3. Use the Find Results view to drag logic instances onto the Clock Resources or the Device view to assign to the appropriate device resource.

   Refer to Using the Find Results View in Chapter 4 for more information on using **Find**.

You can also locate physical resources on the device, such as Global Clock Buffers (`BUFGCTRL`), for placing logic instances. Specify **Sites** in the **Find** field, and define the criteria as needed. Use the results in the **Find Results** field to highlight the physical device resource in the Clock Resources view or the Device view.

Using the Clock Resources View

The Clock Resources view supports the Xilinx 7 series FPGAs, and Virtex-6 devices, showing the relationship and interactions between regional and global clocking resources: `BUFRs`, `BUFIOs`, `BUFGs`, `MMCMs` and `GTs`. The spreadsheet-like interface of the Clock Resources view lets the PlanAhead tool display a simplified view of the device resources, while maintaining proper relative positioning between these resources. Most of the details of the FPGA device shown in the Device view are not shown in the Clock Resources view.

Note: The resources displayed in the Clock Resources represent the target part for the current project, and vary from device to device.

Figure 8-26, page 271 displays the Clock Resources of a Virtex-6 LX75 device. For a complete description of the clock resources of this and other Virtex-6 devices, see the *Virtex-6 FPGA Clocking Resources User Guide (UG362)*, cited in Appendix E, Additional Resources.

The Clock Resources view for this device shows that:

- The device is divided into a top "half" containing four clock regions, and a bottom "half" containing two clock regions.

- The device has six clock regions in total, arranged in a 2x3 matrix, numbered from X0Y0 in the lower-left of the device to X1Y2 in the upper-right.

- Each of these clock regions also has I/O banks containing multi-region clock-capable pins (MRCCs), single-region clock-capable pins (SRCCs), BUFIOs, and BUFRs that display in the Clock Resources view for each clock region.

- In the center column of the device, as displayed in the Clock Resources view, are the Clock Management Tiles (CMTs) with MMCMs and BUFGs for managing global clocks on the device.

- The clock regions on the left side of the device contain both an outer I/O bank (IOOL) and a center I/O bank (IOCL), while the clock regions on the right side of the device have only a center I/O bank (IOCR) with a column of Gigabit Transfers (GTs) located on the right-hand edge of the device.

You can expand or collapse the Clock Resources view levels to display only the information of interest.

- Click **Collapse All** or **Expand All** to hide or view levels of the Clock Resources view.

- Click the '+' or '-' sign on the label of a specific level to expand or collapse that level.

To cross-select the object in other views such as the Device view, in the Clock Resources view, click the name of a specific clock region or I/O bank. Use this to quickly locate a specific object in the Clock Resources view on the device, on the package, or in the netlist.

Use **Automatically Scroll to Selected Object** on the toolbar menu to scroll the Clock Resources view to display an object that is cross-selected when you have selected an instance or resource in another view. Use this to quickly locate a specific resource on the device in the Clock Resources view. You can toggle the automatic scroll off to prevent the PlanAhead tool from changing the displayed resources every time an object is selected in a different view.

Placing Design Instances

The Clock Resources view displays two columns, Site and Instance, under each I/O bank, CMT, or GTX bank to report both the device resource and the design instance to which it is assigned.

You can select logic instances from the design to place into the device resources from the Find Results, Schematic, Netlist, or I/O Ports views.

Select logic instances and drag them onto the instance column of the appropriate device resource in the Clock Resources view.

As you drag the instance around in the Clock Resources view, the PlanAhead tool indicates sites where the instance cannot be placed with a slashed circle symbol and sites where the instance can be placed with a rectangle.

The PlanAhead tool enforces specific rules and limitations regarding global and regional clock tree structures while you are placing instances from the design. For specific information on these rules and limitations refer to the device-specific clocking resources guide.

xc6vlx75tff784-3

Top Half

Clock Region X0Y2

I/O Bank: 16 (Standard)

Site	Instance
BUFR_X0Y5	
BUFR_X0Y4	
BUFIODQS_X0Y11:SR	
BUFIODQS_X0Y10:MR	
BUFIODQS_X0Y9:MR	
BUFIODQS_X0Y8:SR	
IDELAYCTRL_X0Y2	
J21:SRCC	
J22:SRCC	
B27:MRCC	
B28:MRCC	
E25:MRCC	
F24:MRCC	
D27:SRCC	
C28:SRCC	

I/O Bank: 26 (Standard)

Site	Instance
BUFR_X1Y5	
BUFR_X1Y4	
BUFIODQS_X1Y11:SR	
BUFIODQS_X1Y10:MR	
BUFIODQS_X1Y9:MR	
BUFIODQS_X1Y8:SR	
IDELAYCTRL_X1Y2	
J15:SRCC	
K15:SRCC	
B14:MRCC	
A14:MRCC	
F16:MRCC	
F15:MRCC	
G12:SRCC	
F12:SRCC	

CMT @ Y=2

Site	Instance
MMCM_ADV_X0Y5	
MMCM_ADV_X0Y4	

Clock Region X1Y2

I/O Bank: 36 (Standard)

Site	Instance
BUFR_X2Y5	
BUFR_X2Y4	
BUFIODQS_X2Y11:SR	
BUFIODQS_X2Y10:MR	
BUFIODQS_X2Y9:MR	
BUFIODQS_X2Y8:SR	
IDELAYCTRL_X2Y2	
D10:SRCC	
C10:SRCC	
F7:MRCC	
G7:MRCC	
H11:MRCC	
H10:MRCC	
J12:SRCC	
K13:SRCC	

GT Bank 116

Site	Instance
GTXE1_X0Y11	
GTXE1_X0Y10	
GTXE1_X0Y9	
GTXE1_X0Y8	
D2:TXN3	
A4:RXN3	
D1:TXP3	
A3:RXP3	
F2:TXN2	
B2:RXN2	
F1:TXP2	
G4:REFCLK1P	
G3:REFCLK1N	
B1:RXP2	
J4:REFCLK0P	
J3:REFCLK0N	
H2:TXN1	
C4:RXN1	
H1:TXP1	
C3:RXP1	
K2:TXN0	
E4:RXN0	
K1:TXP0	
E3:RXP0	

Clock Region X0Y1

I/O Bank: 15 (Standard)

Site	Instance
BUFR_X0Y3	
BUFR_X0Y2	
BUFIODQS_X0Y7:SR	
BUFIODQS_X0Y6:MR	
BUFIODQS_X0Y5:MR	
BUFIODQS_X0Y4:SR	
IDELAYCTRL_X0Y1	
L21:SRCC	
L22:SRCC	
K27:MRCC	
L26:MRCC	
M22:MRCC	
M21:MRCC	
M26:SRCC	
N26:SRCC	

I/O Bank: 25 (Standard)

Site	Instance
BUFR_X1Y3	
BUFR_X1Y2	
BUFIODQS_X1Y7:SR	
BUFIODQS_X1Y6:MR	
BUFIODQS_X1Y5:MR	
BUFIODQS_X1Y4:SR	
IDELAYCTRL_X1Y1	
A20:SRCC	
A19:SRCC	
H19:MRCC	
G19:MRCC	
D20:MRCC	
E19:MRCC	
A21:SRCC	
A22:SRCC	
H20:GCC	
G21:GCC	
F20:GCC	
F21:GCC	

CMT @ Y=1

Site	Instance
MMCM_ADV_X0Y3	
MMCM_ADV_X0Y2	
BUFGCTRL_X0Y31	
BUFGCTRL_X0Y30	
BUFGCTRL_X0Y29	
BUFGCTRL_X0Y28	
BUFGCTRL_X0Y27	
BUFGCTRL_X0Y26	
BUFGCTRL_X0Y25	
BUFGCTRL_X0Y24	
BUFGCTRL_X0Y23	
BUFGCTRL_X0Y22	
BUFGCTRL_X0Y21	
BUFGCTRL_X0Y20	
BUFGCTRL_X0Y19	
BUFGCTRL_X0Y18	
BUFGCTRL_X0Y17	
BUFGCTRL_X0Y16	

Clock Region X1Y1

I/O Bank: 35 (Standard)

Site	Instance
BUFR_X2Y3	
BUFR_X2Y2	
BUFIODQS_X2Y7:SR	
BUFIODQS_X2Y6:MR	
BUFIODQS_X2Y5:MR	
BUFIODQS_X2Y4:SR	
IDELAYCTRL_X2Y1	
W11:SRCC	
W10:SRCC	
AB11:MRCC	
AA11:MRCC	
AD7:MRCC	
AE7:MRCC	
AF11:SRCC	
AG11:SRCC	
AF10:GCC	
AE10:GCC	
AH11:GCC	
AG12:GCC	

GT Bank 115

Site	Instance
GTXE1_X0Y7	
GTXE1_X0Y6	
GTXE1_X0Y5	
GTXE1_X0Y4	
M2:TXN3	
L4:RXN3	
M1:TXP3	
L3:RXP3	
P2:TXN2	
N4:RXN2	
P1:TXP2	
P6:REFCLK1P	
P5:REFCLK1N	
N3:RXP2	
T6:REFCLK0P	
T5:REFCLK0N	
T2:TXN1	
R4:RXN1	
T1:TXP1	
R3:RXP1	
V2:TXN0	
U4:RXN0	
V1:TXP0	
U3:RXP0	

Bottom Half

Clock Region X0Y0

I/O Bank: 14 (Standard)

Site	Instance
BUFR_X0Y1	
BUFR_X0Y0	
BUFIODQS_X0Y3:SR	
BUFIODQS_X0Y2:MR	
BUFIODQS_X0Y1:MR	
BUFIODQS_X0Y0:SR	
IDELAYCTRL_X0Y0	
P21:SRCC	
P22:SRCC	
Y23:MRCC	
W23:MRCC	
U26:MRCC	
T26:MRCC	
V23:SRCC	
U23:SRCC	

I/O Bank: 24 (Standard)

Site	Instance
BUFR_X1Y1	
BUFR_X1Y0	
BUFIODQS_X1Y3:SR	
BUFIODQS_X1Y2:MR	
BUFIODQS_X1Y1:MR	
BUFIODQS_X1Y0:SR	
IDELAYCTRL_X1Y0	
AA26:GCC	
AA27:GCC	
AD28:GCC	
AD27:GCC	
AB27:SRCC	
AB26:SRCC	
AF26:MRCC	
AG26:MRCC	
AE25:MRCC	
AF25:MRCC	
AB23:SRCC	
AC23:SRCC	

CMT @ Y=0

Site	Instance
BUFGCTRL_X0Y15	
BUFGCTRL_X0Y14	
BUFGCTRL_X0Y13	
BUFGCTRL_X0Y12	
BUFGCTRL_X0Y11	
BUFGCTRL_X0Y10	
BUFGCTRL_X0Y9	
BUFGCTRL_X0Y8	
BUFGCTRL_X0Y7	
BUFGCTRL_X0Y6	
BUFGCTRL_X0Y5	
BUFGCTRL_X0Y4	
BUFGCTRL_X0Y3	
BUFGCTRL_X0Y2	
BUFGCTRL_X0Y1	
BUFGCTRL_X0Y0	
MMCM_ADV_X0Y1	
MMCM_ADV_X0Y0	

Clock Region X1Y0

I/O Bank: 34 (Standard)

Site	Instance
BUFR_X2Y1	
BUFR_X2Y0	
BUFIODQS_X2Y3:SR	
BUFIODQS_X2Y2:MR	
BUFIODQS_X2Y1:MR	
BUFIODQS_X2Y0:SR	
IDELAYCTRL_X2Y0	
AE13:GCC	
AD12:GCC	
AB12:GCC	
AC13:GCC	
Y13:SRCC	
AA12:SRCC	
AH15:MRCC	
AH16:MRCC	
AG14:MRCC	
AF15:MRCC	
AD15:SRCC	
AC15:SRCC	

GT Bank 114

Site	Instance
GTXE1_X0Y3	
GTXE1_X0Y2	
GTXE1_X0Y1	
GTXE1_X0Y0	
Y2:TXN3	
AC4:RXN3	
Y1:TXP3	
AC3:RXP3	
AB2:TXN2	
AE4:RXN2	
AB1:TXP2	
W4:REFCLK1P	
W3:REFCLK1N	
AE3:RXP2	
AA4:REFCLK0P	
AA3:REFCLK0N	
AD2:TXN1	
AG4:RXN1	
AD1:TXP1	
AG3:RXP1	

Figure 8-26: **Clock Resources View**

Placing Clock Logic in the Device View

To place clock logic manually:

1. In the Device view zoom to locate the appropriate device site to place the logic.

2. Select the **Instance Drag & Drop Modes** button on the toolbar menu, and select the **Create Site Constraint Mode**.

3. Select the logic instance to place from the Find Results, Schematic, Netlist, or I/O Ports views, and drag it onto the appropriate device resource in the Device view.

Figure 8-27 shows an example of manual clock placement.
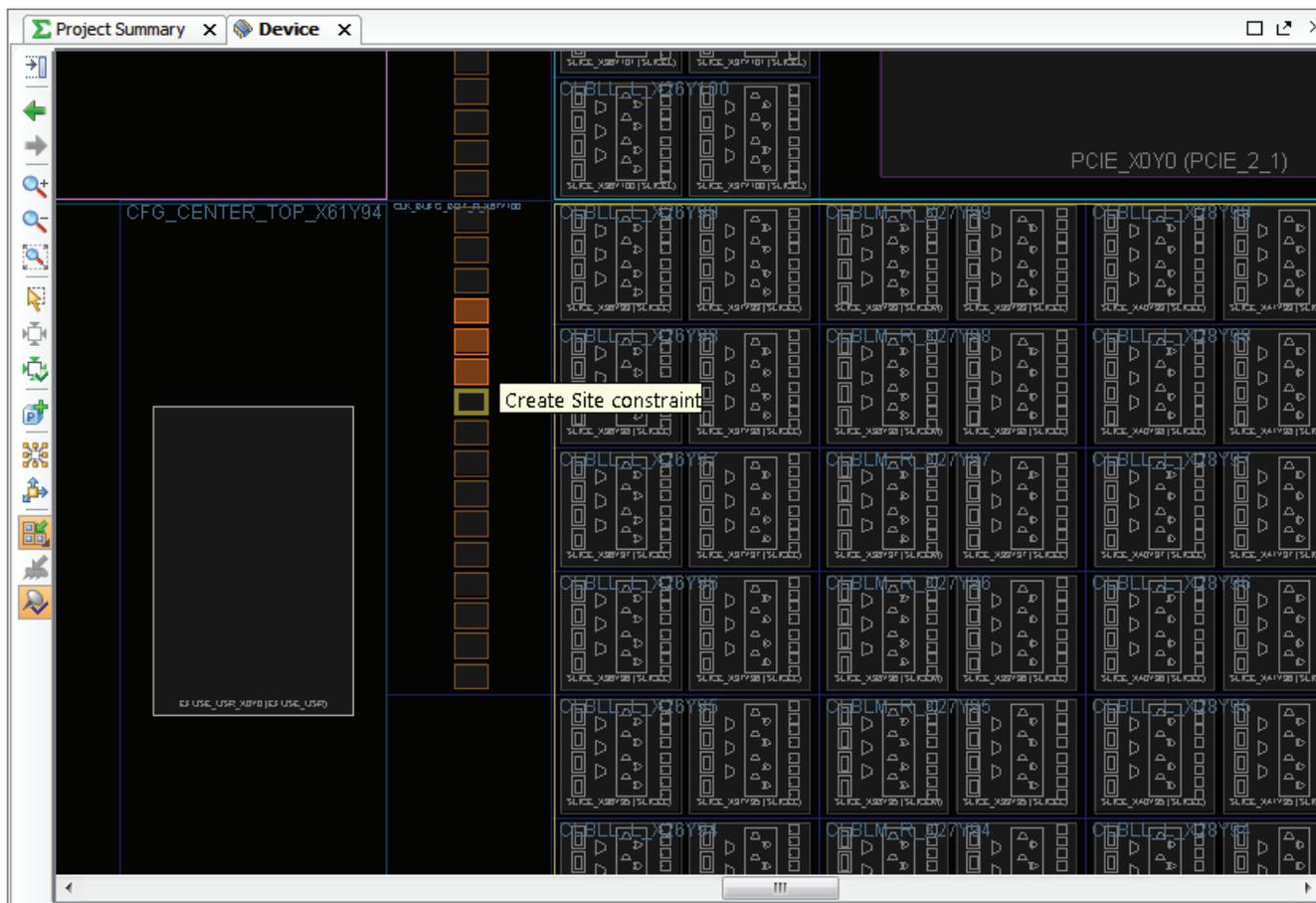
Figure 8-27: **Manually Placing Clock Logic**

Validating I/O and Clock Logic Placement

This section describes the required steps for running I/O Port and clock-related DRCs. See I/O Port and Clock Logic and Placement DRC Rule Descriptions, page 419 for information on running netlist and floorplan related DRCs.

Running I/O Port and Clock Logic Related DRCs

To select and run individual rules:

1. Select:

   • **Tools > Report DRC** from the main menu, or

   • **Report DRC** from the Flow Navigator.

   The Run DRC dialog box opens, as shown in Figure 8-28.
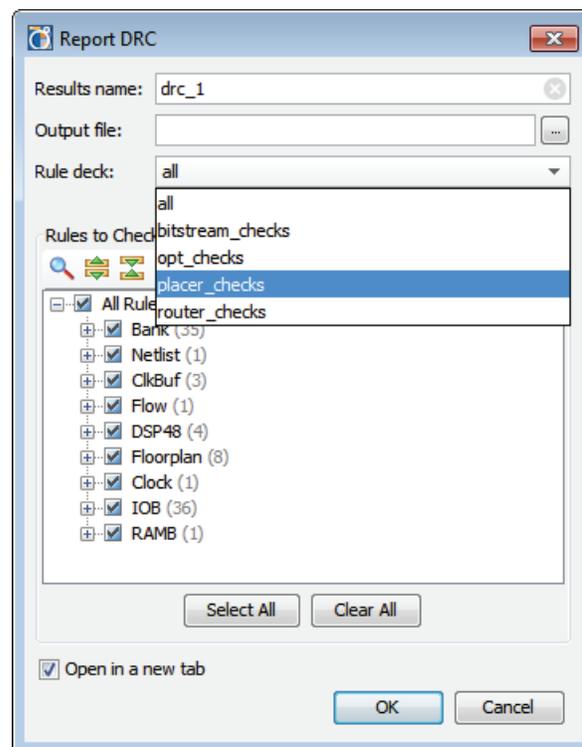
Figure 8-28:   **Run DRC Dialog Box: I/O Pin and Clock DRC Rules**

2. View or edit the Results Name field. Enter a name for the results for a particular run for easier identification during debug in the DRC Violations browser. The output file name matches the entered name.

3. In the **Rule deck** field, you can select the collection of rules to run, or use the **Rules to Check** check boxes to select specifiec design rules to check for each design object. For information about rule decks and rules, see I/O Port and Clock Logic and Placement DRC Rule Descriptions, page 419.

4. Click **OK** to invoke the selected DRC checks.

Viewing DRC Violations

If violations are found, the DRC Results view opens, as shown in Figure 8-29. The DRC Results view displays the rule violations found, grouped under the various rule categories defined in the Run DRC dialog box. The rule violations are also categorized by severity, and display color codes for quick review of errors, warnings, and informational messages.
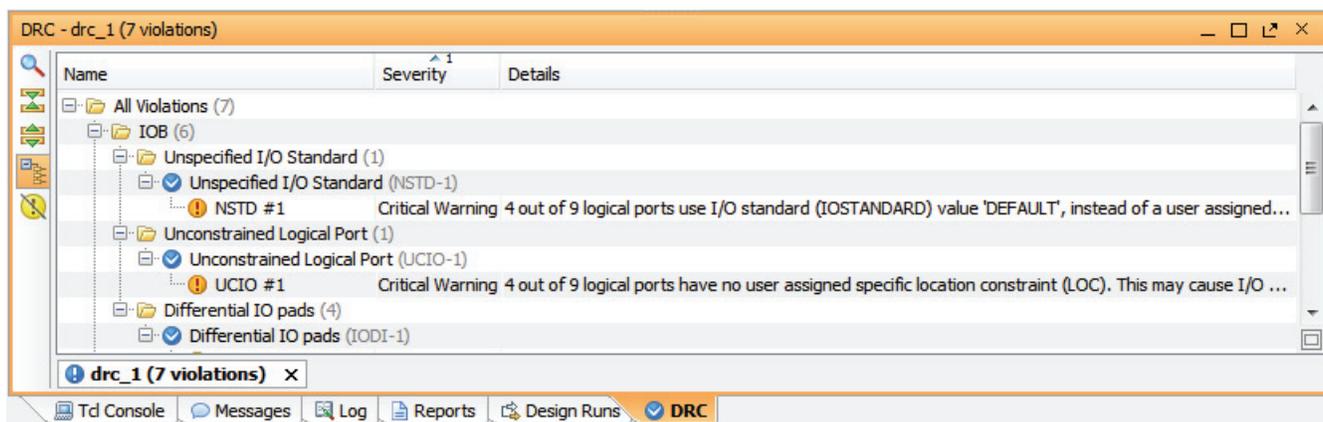
Figure 8-29: **Highlighting DRC Violations**

A violation can be:

- Informational only, to make you aware of a possible issue; shown in yellow.
- A warning to suggest an issue that might need some resolution; shown in orange.
- A critical warning to highlight an issue that you must resolve; shown in orange.
- An error to highlight issues that would prevent proper implementation of your design. Errors show with a red marker.

You can toggle the **Hide Warnings and Informational Messages** button in the toolbar menu to turn off warnings and informational messages to see only the errors reported.

You can also click on the header of the **Severity** column of the DRC Results view to sort violations by severity.

- Click once on the column header to sort in an increasing order
- Click twice to sort in a decreasing order

See Using Tree Table Style Views in Chapter 4 for more information.

Select a violation message in the DRC Results view, then select **Violations Properties** to open the Violations Properties view. This view provides a general review of the DRC rule violation, and specific details of the design elements that violate the rule.

The **Details** tab of the Violations Properties might have links to specific design objects that violate the DRC. Click these links to view the design object in the RTL Netlist view, the Device view, the Schematic view, or the source RTL file.

Migrating to an RTL Design

The I/O Pin Planning project can be used as the foundation, or starting point of an RTL source-based project.

When the I/O ports have been defined and placed onto the package pins, you can transform the I/O Planning project into an RTL source project. The current port definitions are used to create a top module for the RTL design in either Verilog or VHDL, as specified. Differential pair buffers are added to the top module, and bus definitions are also included in the RTL. The project attributes are changed to reflect the RTL source project type.

Note: The RTL project cannot be converted back into an I/O Planning project.

To convert the project select one of the following commands:

- **File > Migrate to RTL** from the main menu.

- **Migrate to RTL** from the Flow Navigator menu.

The Migrate to RTL dialog box opens as shown in Figure 8-30.
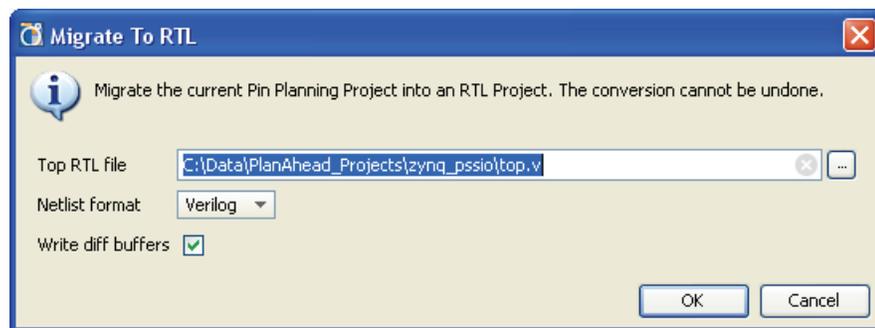
Figure 8-30: **Migrate to RTL Project**

The options are:

Top RTL file - The Verilog (`.v`) or VHDL (`.vhd`) file to create for the top module of the design. The HDL file will include the module definition, with port definitions, direction, and width for bus pins.

Netlist format - Specify Verilog or VHDL format for the top module.

Write diff buffers - Write the diff pair buffers as part of the top module definition. This preserves any differential pairs defined in the I/O Pin Planning project.

When you click OK, the current I/O Pin Planning project is converted, in place, into an RTL source-based project. You may now begin adding sources to the project, and working with it as the foundation of your design. See Chapter 5, Elaborated RTL Design for details.

Exporting I/O Pin and Package Data

You can export I/O pin and pin package information as described in the following subsections.

Exporting Package Pin Information

You can export the device package pin information to a CSV format file. The exported information includes information about all of the package pins in the device as well as design-specific I/O Port assignments and their configuration.

The package pin section of the exported list is a good starting point for defining I/O port definitions in a spreadsheet format. Refer to Defining and Configuring I/O Ports, page 254, for information on the exported CSV file format.

Exporting an I/O Port List

You can export the I/O port list to a file for use in RTL coding or PCB schematic symbol creation.

1. To export the I/O Ports list information, select **File > Export > Export I/O Ports**, as shown in Figure 8-31.
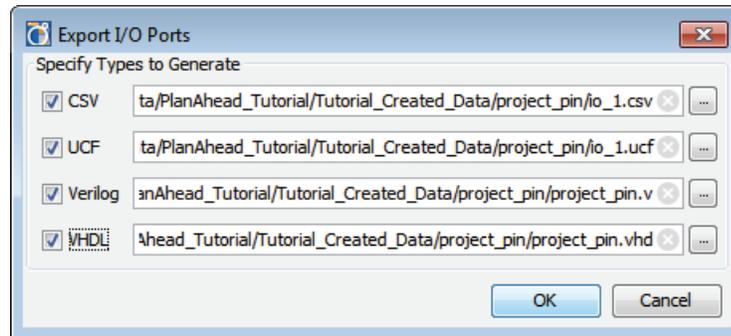
Figure 8-31: **Exporting I/O Port Lists**

You can specify the type of I/O port to generate and the path. The I/O port type options are: CSV, UCF, Verilog, and VHDL.

Exporting IBIS Models

To better understand the signal integrity at the system level, PCB designers often need to simulate the design with I/O Buffer Information Specification (IBIS) models. Designers must consider signal integrity issues such cross talk, ground bounce, and Simultaneous Switching Noise (SSN). IBIS models help characterize I/V curves and parasitic information of the packaged device.

Xilinx provides generic IBIS models for complete device families that can be downloaded from the website: http://www.xilinx.com/support/download.

For Xilinx 7 series FPGAs, the PlanAhead software can generate IBIS models from the design and per-pin package data through the Export IBIS Model command. The PlanAhead tool uses the netlist and implementation details from the design, and combines that information with the available per-pin parasitic package information to create a custom IBIS model for the design.

With an Elaborated Design, Synthesized Design, or Implemented Design open, you can export an IBIS file for use in analyzing the design.

Select **File > Export > Export IBIS Model**. The Export IBIS Model dialog box opens, as shown in Figure 8-32, page 277.
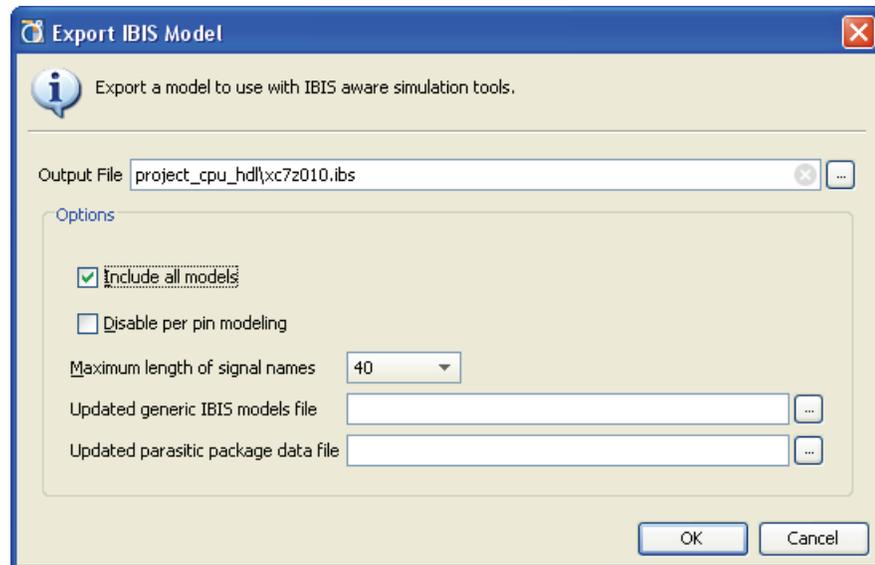
Figure 8-32: **Export IBIS Model Dialog Box**

The Export IBIS Model options are:

- **Output File:** Specify the filename and path for the output IBIS file.

- **Include all models:** Checkbox to include all available I/O buffer models for this device. By default, only buffer models used in the design are included.

- **Disable per pin modeling:** Checkbox to disable inclusion of the per-pin modeling of the package. This is the path from the die pad of the device to the pin of the package. With per-pin modeling disabled, the package is reduced to a single RLC transmission line model applied to all pins and defined in the `[Package]` section of the IBIS file.

- **Maximum length of signal names:** Truncate signal names to the specified limit.

    - 40: Truncate signal names to 40 characters, supported by IBIS version 4.2 as the default.

    - 20: Truncate signal names to 20 characters.

    - Unlimited: Do not truncate signal names.

- **Updated generic IBIS model file:** Optionally provide an IBIS model file for the device. This is used to override the IBIS models found in the installation under the `parts` directory.

    Note: The IBIS model file is required for devices that do not have IBIS models included with software installation.

- **Updated parasitic package data file**: Optionally provide a parasitic package file (`.pkg`) file to use for per-pin extractions. This is used to override the parasitic package file found in the installation hierarchy under the `parts` directory.

    Note: The parasitic package file is required for devices that do not have IBIS models included with software installation.

Using Noise Analysis Predictors

The PlanAhead tool provides analysis of the switching noise levels associated with the I/O of different devices. This analysis can be accessed by clicking the **Run Noise Analysis** command from Flow Navigator or from the **Tools > Run Noise Analysis** command from the main menu. Depending on the Xilinx device targeted by the design, the PlanAhead tool performs either a Simultaneous Switching Noise (SSN) or a Simultaneous Switching Output (SSO) analysis.

- For XIlinx 7 series FPGAs, Virtex-6, and Spartan-6 devices, the PlanAhead tool performs SSN analysis. See Running SSN Analysis for more information on this type of analysis.

- For Spartan-3, Virtex-4, and Virtex-5 devices, the PlanAhead tool performs SSO analysis. See Running WASSO Analysis, page 283 for more information.

Running SSN Analysis

The PlanAhead tool uses the SSN predictor to provide detailed analysis of simultaneous switching output noise in Spartan-6, Virtex-6, Virtex-7, Kintex™-7, and Artix™-7 devices[1]. SSN analysis provides estimates of the disruption that simultaneously switching outputs can cause on other output ports in the I/O bank, as well as input ports in the case of Spartan-6 devices. The SSN predictor incorporates I/O bank-specific electrical characteristics into the prediction to better model package effects on SSN.

I/Os are grouped into separate isolated I/O banks, each with its own unique power distribution networks, and each with unique responses to switching activity. Because power distribution networks within a packaged FPGA have different responses to noise, it is important to understand not only the I/O standards and number of I/O in a design, but also the response of the device power system to this switching.

Xilinx characterizes all banks through three-dimensional extraction and simulation. This information is incorporated into the SSN predictor such that it can take the expected switching profile of a device and predict how the switching affects the power network of the system, and in turn, how other outputs in the I/O bank are affected.

The SSN predictor is the most accurate method available for predicting how output switching affects interface noise margins. The calculation and results are based on a range of variables. These estimates are intended to identify potential noise-related issues in your design and should not be used as final design "sign off" criteria.

---

1. For Virtex-7, Kintex-7, and Artix-7 devices the PlanAhead tool performs a preliminary SSN analysis.

To run the SSN predictor:

1. In the Flow Navigator, or from the Tools menu, select **Run Noise Analysis**. The Run SSN Analysis dialog box opens, as shown in Figure 8-33.
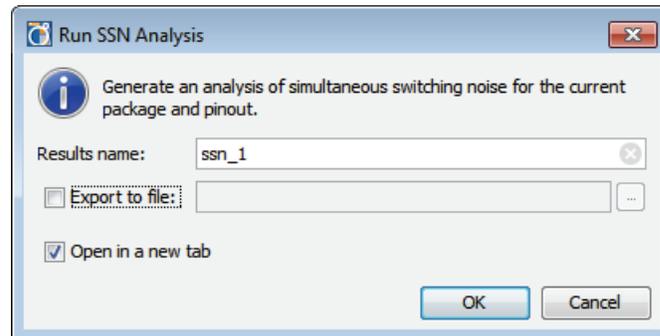
Figure 8-33: **Run SSN Analysis Dialog Box**

Note: The Run Noise Analysis command runs either SSN or SSO analysis based on the target Xilinx device. The dialog box that opens depends on the type of analysis the PlanAhead tool performs. See Running WASSO Analysis, page 283 for more information.

2. Enter a name in the **Results Name** field to identify the results in the SSN Results view.

3. Click the **Export to File** checkbox, and enter an output file name in the **Output File** field, or browse and select a location to write an external report file,

    Specify the **Output Format** for the file as either CSV, or HTML.

4. Click **OK**.
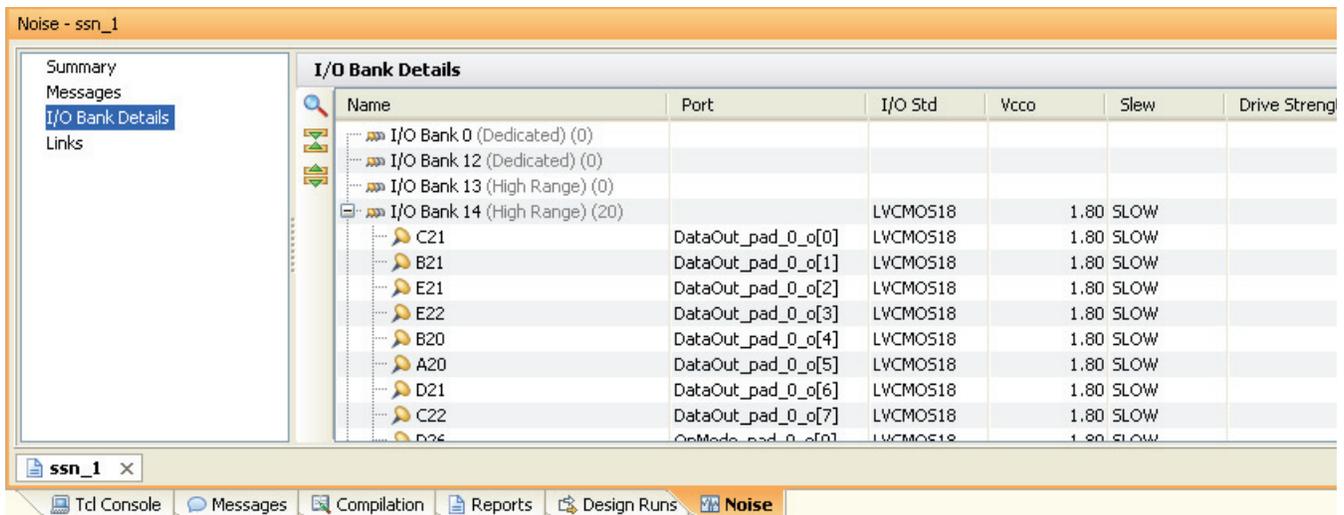
Figure 8-34: **SSN Results View**

Viewing the SSN Results

After the analysis is complete, the SSN Results view opens as shown in Figure 8-34, displaying the following information:

- **Summary**: A summary of the Report Noise command, the current design, and the results.

- **Messages**: Specific messages returned by the Report Noise command.

- **I/O Bank Details**: Specific findings for the Report Noise command. The results are returned to a table style view on the right-hand side of the Noise Report, as described below, begginning with **Name**.

- **Links**: Links to additional information related to the current device and the Report Noise command.

- **Name**: Displays the I/O Banks available in the device. Each I/O bank has pin icons indicating how full the bank is, and puts a check mark for a passing predictor or a red circle indicating failure.

  - **Group (Virtex-6 only)**: Displays the pin groups with like I/O standards assigned within the bank, and displays their status. Groups are determined automatically; based on the I/O standards, drive strength, slew rate, and phase assigned.

- **I/O Std, Vcco, Slew, Drive Strength**: Displays the appropriate values for the port or bank.

- **Noise (V)**

  - **Contributed (Virtex-6 only)**: Contains the SSN aggregate of each group, generated by the I/O standard, drive strength, and slew type of that group.

  - **Bank Total (Virtex-6 only)**: Defines aggregate SSN predicted for a bank or group. If multiple phases are specified for the groups of a given bank, the SSN contributions of groups with different phases are accumulated separately, and the maximum of these is reported. Because the SSN calculation is isolated to the output of that bank, one SSN bank total does not affect another bank total.
    This column identifies which I/O groups are creating the most SSN, and how much margin they use.

- **Off-Chip Termination**: The Off-Chip Termination field automatically populates with the default terminations for each I/O standard, if one exists. For example, for LVTTL (at 2mA, 4mA, 6mA, and 8mA) no termination is assumed. However, for LVTTL (at 12mA, 16mA, and 24mA) a far-end parallel termination of 50 Ohms to V_{TT} is assumed. As a result of this termination, the available noise margin is less for signals with drive strength of 12mA, or more, when compared to 2mA to 8mA.

  Virtex-4, Virtex-5,Virtex-6, Spartan-6, and all Xilinx 7 series FPGA devices use this assumption.

  Displays either **None** or a short description of the expected or defined off-chip termination style; for example, `FP_VTT_50` describes a **Far-end Parallel 50 Ω termination to V_{TT}** termination style.

  The full list of termination styles is available in the device-specific *SelectIO™ Resources User Guide*, cited in Appendix E, Additional Resources

  To change the settings, use either the CSV file import feature described in Importing a CSV Format File, or the pulldown selection in the I/O Ports table.

- **OUT_TERM (Spartan-6 only)**: OUT_TERM (Spartan-6 Devices Only) displays the OUT_TERM attribute setting for the port, if defined. NONE is most common.

  For more information about OUT_TERM, see the *Spartan-6 FPGA SelectIO Resources User Guide (UG381)*, cited in Appendix E, Additional Resources.

- **Margin (V)**
  - **Available (Virtex-6 only)**: Defines the allowable noise margin for that particular I/O standard on the high side of the signal as it switches to a 1. It is strictly based on the DC logic levels implicit in the I/O standard (no quantity information is taken into account) and represents the margin that the weakest drive of a high signal is above the JEDEC input thresholds. These margin values assume the weakest drive conditions, JEDEC specification termination, and standard receiver requirements for the standard. This is one place where conservative assumptions are made in the analysis, providing some guard band.
  - **Remaining**: Displays the amount of noise margin that is left over after accounting for all SSN in the bank.
- **Result**: Displays a Pass or Fail condition with failures in red.
- **Notes**: Displays information about the I/O bank or groups.

The SSN results are relative to the state of the design when the SSN Analysis is run. It is not a dynamic report.

Improving SSN Results

To improve SSN results when a violation occurs:

- Use I/O standards that have a lower SSN impact for the failing group. Changing to a lower drive strength, a parallel-terminated DCI I/O standard, or a lower class of driver can improve SSN; for example, changing the SSTL Class II to an SSTL Class I.
- Spread the failing pins across multiple banks. This reduces the number of aggressive outputs on the power system of one bank.
- Spread the failing group(s) across multiple synchronous phases. (See the following note regarding phase groups).
- If the Result is a Fail condition, assign phase groups to ports that are switching concurrently.

  Note: Phase groups are only supported in the software and SSN calculations for Virtex-6 devices. While phase shifting improves the Spartan-6 device performance on SSN, the improved performance cannot be seen in the software calculations. For Virtex-6, see Defining I/O Port Switching Phase Groups in Virtex-6 Devices, page 282.

- Phase-shift the offending group by 90 degrees if at a DDR rate, or by 180 degrees if at an SDR rate. This puts half the aggressive output switching out-of-phase with the other half, and instead of interfering constructively, it interferes destructively.
- If a Spartan-6 design fails SSN, certain failures can be ignored. See AR 36141 for details. For more information, see "Pin Planning to Mitigate SSO Sensitivity" in the *Spartan-6 FPGA SelectIO Resources User Guide (UG381)*, cited in Appendix E, Additional Resources.

Viewing I/O Bank Properties in SSN Results

You can select an I/O bank in the SSN Results view to display information about the I/O ports, pins, and groups assigned to the I/O bank in the I/O Bank Properties view.

- Select the **General** tab to view information about the number and types of Ports assigned to the I/O Bank.
- Select the **Package Pins** or **I/O Ports** tab to view the detailed information about the Pins or Ports within the bank, as shown in Figure 8-35, page 282.

Figure 8-35: **I/O Bank Properties: Package Pins**

Defining I/O Port Switching Phase Groups in Virtex-6 Devices

Note: Phase groups are supported in Virtex-6 devices only.

Sometimes, different groups of I/O within a bank have a synchronous phase offset from one another, meaning it is not possible for them to switch simultaneously. This is true of data and strobe signals in many memory interfaces. In this case, proper SSN accounting must be informed by phase information.

A *phase group* is a logical grouping of ports that are all in phase with each other from a timing perspective (their clocks have the same frequency and phase). When you create a grouping phase, not only the group created, but I/Os with a different phase are separated.

The noise produced by the groups within a bank is summed to get the total noise for that bank, and if all outputs are either in phase with each other, or do not have a synchronous relationship, the output can be expected to switch (change values) at the same time.

If a bank is failing in the SSN analysis, phase groups can be used to group ports that are at separate synchronous phases, thus lowering the total noise for that bank when the SSN predictor is run again.

To set the switching phase for a single, or a set of I/O ports:

1. In any of the I/O Planning views, select one or more I/O port(s).

2. In the I/O Ports view, Package Pins view, or SSN view, select **Configure I/O Ports**, as shown in Figure 8-36, page 283.
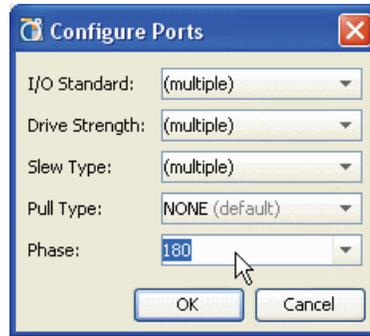
Figure 8-36: **Configure Ports Dialog Box**

3. In the **Configure Ports** dialog box, ensure the I/O Standard is correct.

4. If the port(s) are in-phase, leave the **Phase** as default, or enter a unique phase, such as `180`, and click **OK**.

5. After the appropriate phase groups are assigned, rerun the SSN analysis.

Note: Asynchronous groups should not be treated as separate synchronous phases, as it is possible for them to switch simultaneously.

Running WASSO Analysis

The PlanAhead tool contains a set of Weighted Average Simultaneous Switching Output (WASSO) checks to validate signal integrity of the device based on the I/O pin and bank assignments made in the design. This analysis provides less detail than the SSN analysis offered for the Spartan-6, Virtex-6, Virtex-7, Kintex-7, and Artix-7devices, but provides some understanding of the switching noise for Spartan-3, Virtex-4, and Virtex-5 devices.

1. To run a WASSO analysis, select **Run Noise Analysis** from either the Flow Navigator or from the Tools menu. The **Run WASSO Analysis** dialog box opens, as shown in Figure 8-37, page 284.

    Note: Run Noise Analysis runs either SSN or SSO analysis based on the target Xilinx device. The dialog box that opens depends on the type of analysis the PlanAhead tool performs. See Using Noise Analysis Predictors, page 278 for more information.

2. Use the **Output File** field to specify a report file name and a location to write to disk.

3. Tool tips appear when you drag the cursor over each of the entry fields that indicate what values to enter. You can modify the device and board parameter values to reflect your specific design.
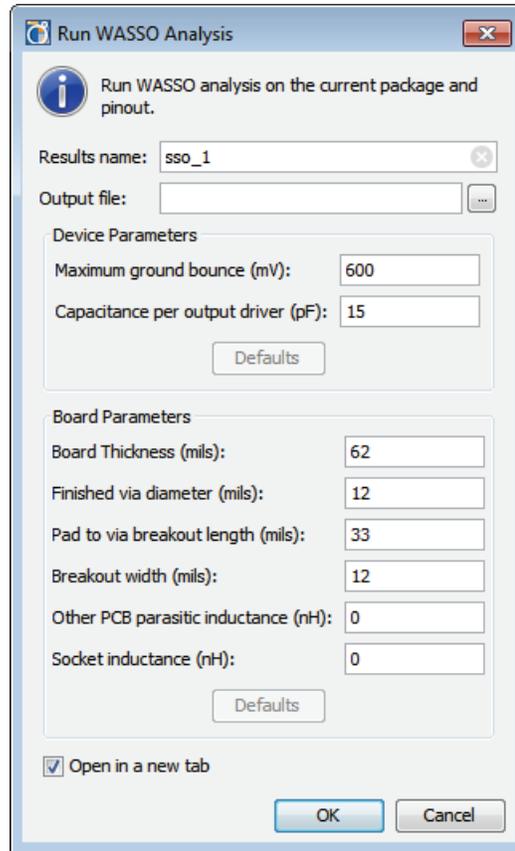
Figure 8-37: **Run WASSO Analysis Dialog Box**

Viewing the WASSO Analysis Results

The analysis is performed across the entire device first and then within each I/O bank as they relate to their neighboring I/O banks. The WASSO Results view displays in the workspace, as shown in Figure 8-38, page 285.
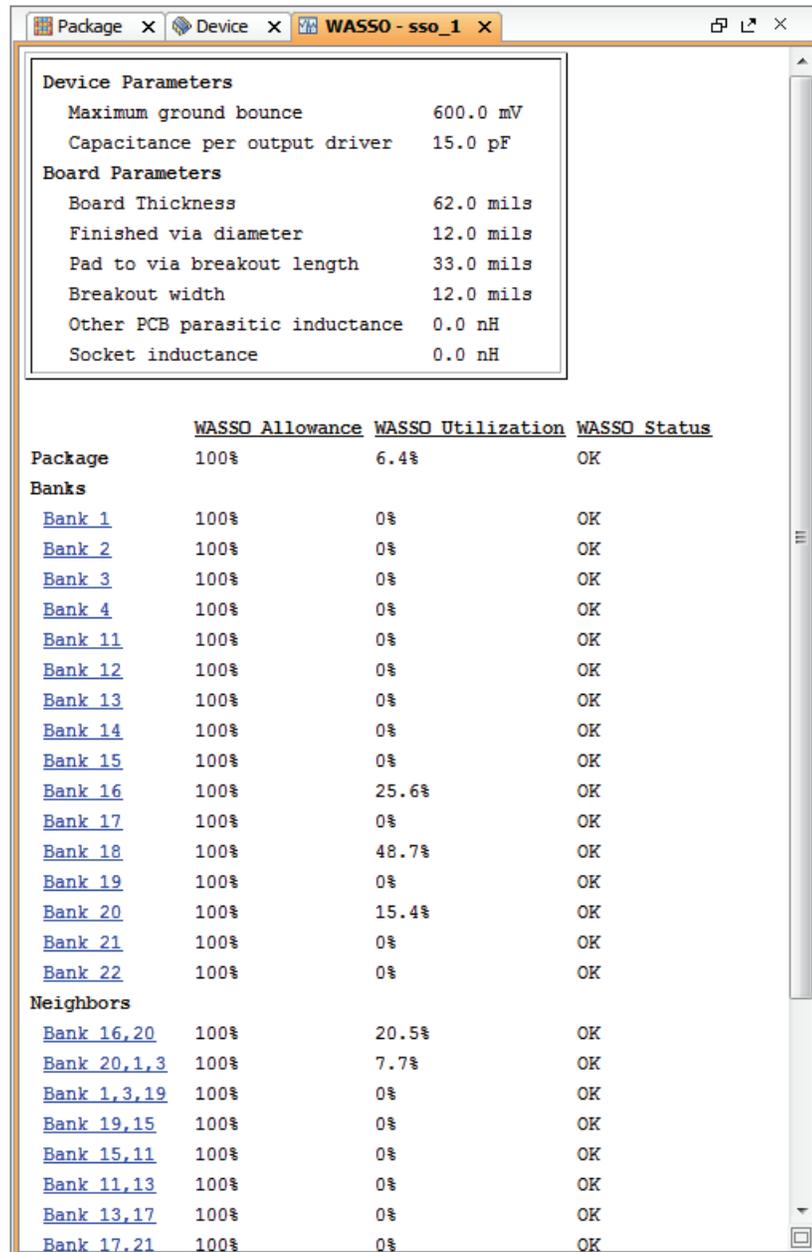
Figure 8-38: **WASSO Results**

Notice that the report lists load allowance, utilization, and status for I/O banks and neighboring pairs.

Implementing the Design

The PlanAhead™ tool features a push-button flow with single synthesis and implementation attempts or *runs*.

Run data is managed automatically, allowing repeated run attempts with varying:

- Synthesis options
- Implementation options
- Constraints

The PlanAhead tool also allows multiple synthesis and implementation runs using varied:

- Software command options
- Timing constraints
- Physical constraints

You can queue the synthesis and implementation runs to launch sequentially or simultaneously with multi-processor computers. Implementation runs use Xilinx® ISE® Design Suite tools.

Running Implementation

To implement the synthesized netlist onto the targeted Xilinx part, you must run the source files and the design constraints through a series of steps known as implementation. Do this using the following process:

- Defining Implementation Runs
- Setting Implementation Options
- Launching Implementation Runs

Defining Implementation Runs

You can create and launch new implementation runs to explore design alternatives and find the best results. You can queue and launch the runs serially, or in parallel using multiple local CPUs.

On Linux systems, you can use remote servers. See Launching Runs on Remote Linux Hosts, page 300.

An implementation run can be defined by:

- Select **Flow > Create Runs** from the main menu.
- In the Flow Navigator, click **Create Implementation Runs** from the Implementation popup menu, as shown in Figure 9-1, page 288.
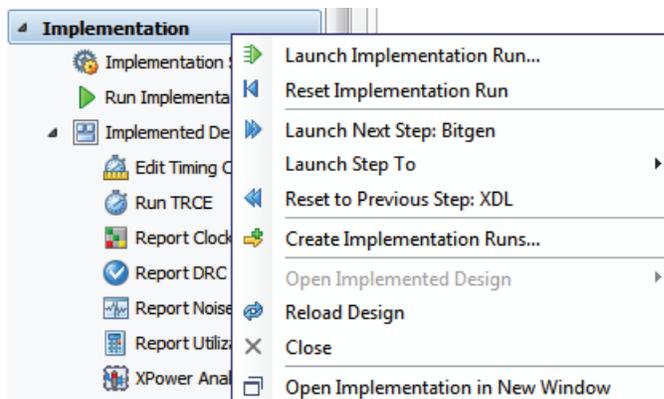
Figure 9-1: **Flow Navigator: Create Implementation Runs**

The **Create New Runs** wizard opens. The first page of the wizard is a summary of the command.

1. Click **Next** to proceed.

    The **Configure Implementation Runs** dialog box opens, as shown in Figure 9-2.
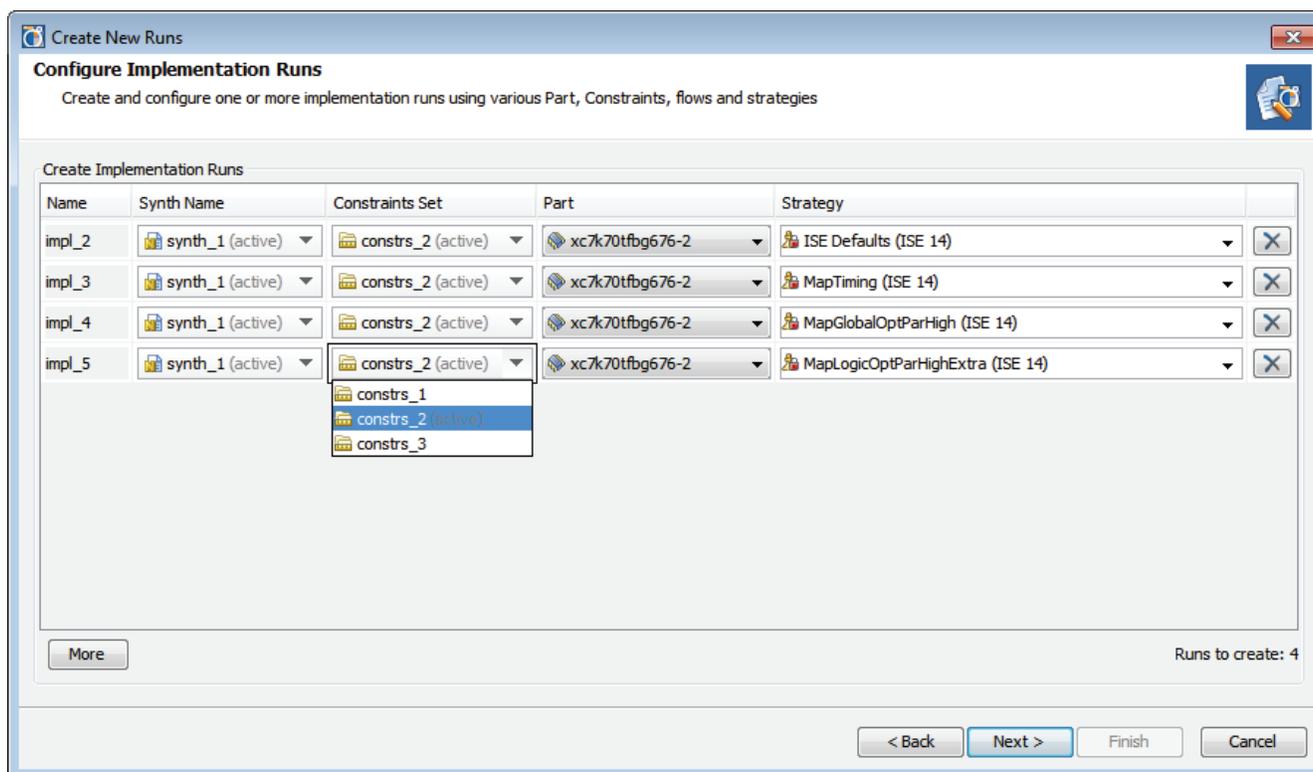
Figure 9-2: **Design Runs View**

2. Enter a **Name** for the run in the Configure Implementation Runs dialog box, or accept the default name.

3. Select a **Synth Name** to choose the synthesis run that will be used to generate (or that has already generated) the synthesized netlist to be implemented.

The default is the currently active synthesis run in the Design Runs window. The default values are defined by the synthesis or implementation Project Settings at the time the **Create New Runs** command is run. See Configuring Project Settings, page 88.

Alternatively, you can select a synthesized netlist that was imported into the project from a third party synthesis tool.

4. Select a **Constraints Set** to specify the constraint set to apply during implementation.

   The optimization, placement, and routing are largely directed by the physical and timing constraints in the specified constraint set.

5. Select a target Part.

   The default values for Constraints Set and Part are defined by the Project Settings when the **Create New Runs** command is executed.

6. Select a **Strategy** for the new run.

   The strategies are a defined set of ISE tools run-time options controlling the implementation results. See Defining Strategies for Synthesis and Implementation in Chapter 4 for more information.

7. Click the **More** button to define additional runs. Specify names and strategies for the added runs as shown in Figure 9-2, page 288.

8. Click the **Remove Run** button to remove a selected run.

9. Click **Next**.

   The Launch Options dialog box opens as shown in Figure 9-3.

10. Specify the **Launch Directory** to create and store the implementation run data.

    The default launch directory is contained within the local project directory structure. Files for implementation runs are stored at:

    `<project_name>/<project_name>.runs/<run_name>`

    Note: Defining any non-default location outside of the project directory structure makes the project non-portable because absolute paths are written into the project files.
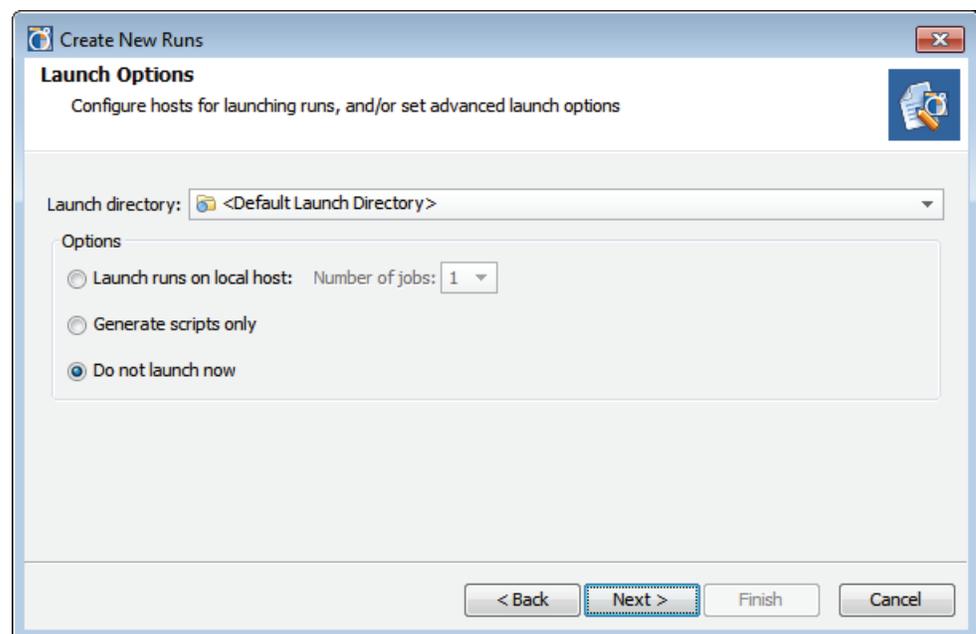

Figure 9-3: **Implementation Launch Options**

11. Specify the **Options**:

- **Launch Runs on Local Host**: Launch the run on the local machine processor.

  - **Number of Jobs**: Define the number of local processors to use for runs. This option is used only when you are launching multiple runs simultaneously. Individual runs are launched on each processor. No multi-threaded processors are used with this option.

- **Generate scripts only**: Export and create the run directory and run script, but do not launch the run at this time. The script can be run at a later time outside of the PlanAhead tool.

- **Do not launch now** - Save the new runs, but do not launch or create run scripts at this time.

12. Click **Next** and review the **Create New Runs Summary**.

13. Click **Finish** to create the defined runs and execute the specified launch options.

New runs are added to the Design Runs view.

Using the Design Runs View

The Design Runs view displays all of the synthesis and implementation runs created in a project, and provides commands to configure, manage, and launch the runs.

Select **Window > Design Runs** to open the Design Runs view if it is not already displayed. Figure 9-4 shows the Design Runs view.

Each implementation run appears indented beneath the synthesis run it is a child of. A synthesis run can have multiple implementation runs. You can expand and collapse synthesis runs using the tree widgets in the view. The Design Runs view is a tree table view. Refer to Using Tree Table Style Views, page 113, for more information on working with the columns to sort the data in this view.

Figure 9-4: **Design Runs View**

The Design Runs view reports the run status, including when the run has not been started, is in progress, is complete, or is out-of-date. Runs can become out-of-date when source files, constraints or project settings are modified. You can reset and delete stale run data in the Design Runs view.

Setting the Active Run

Only one synthesis run and one implementation run can be "active" in the PlanAhead tool at any time. The Compilation and Messages views, Status Bar, and Project Summary display the information for the active run. The Project Summary view only displays compilation, resource, and summary information for the active run.

To make a run active, select the run in the Design Runs view and use the **Make Active** command from the popup menu to set it as the active run.

Setting Implementation Options

You can change many aspects of an implementation run prior to launching the run.
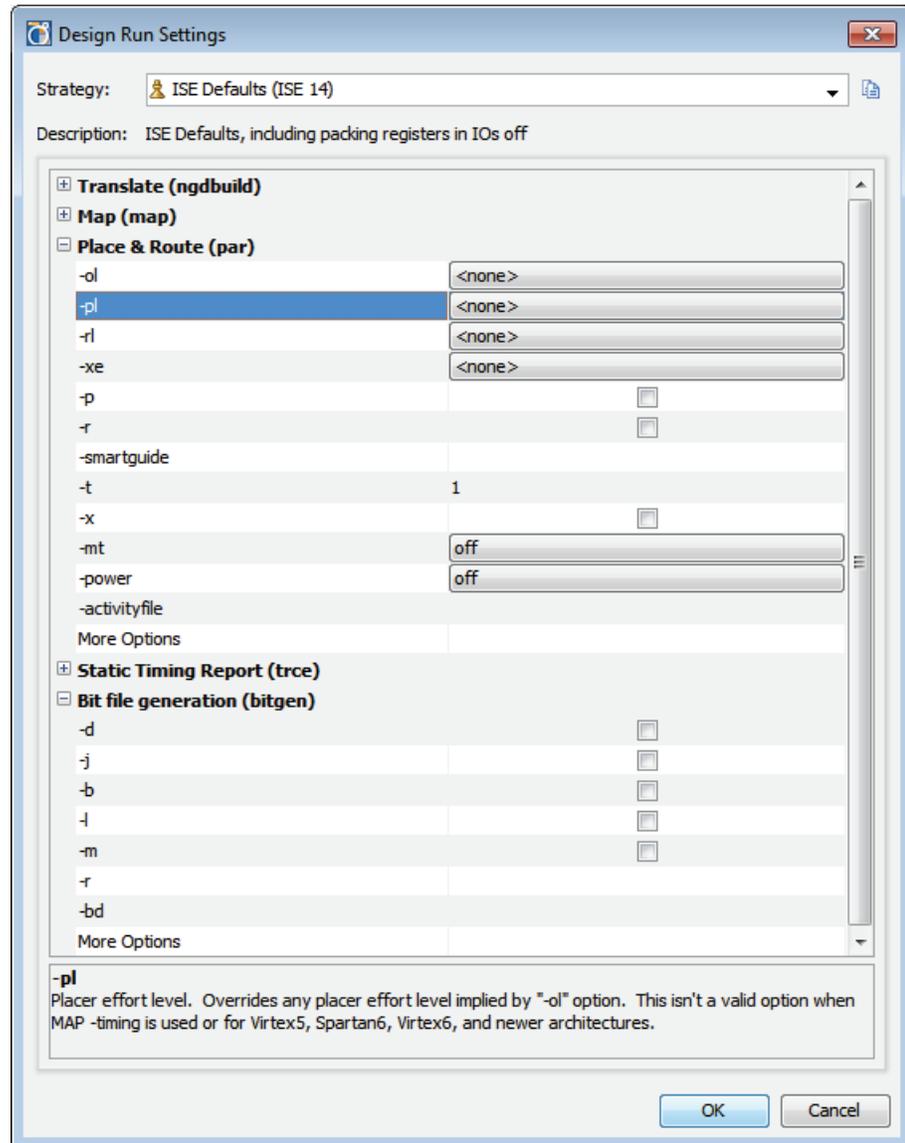
Figure 9-5:   **Change Implementation Settings**

When you select a run in the Design Runs view, the Run Properties view displays the current configuration of the selected run. In the Run Properties view you can change the name of the run, the Xilinx part targeted by the run, the run description, and the constraints set that both drives the implementation and is the target of new constraints. See Using the Run Properties View in Chapter 4 for more information.

You can also change the run-time options used by implementation tools by using the **Change Run Settings** command from the Design Runs view popup menu, to open the Design Run Settings dialog box, as shown in Figure 9-5, page 291.

The Design Run Settings dialog box lets you specify:

- **Strategy**: The general strategy to be used by the run.

- **ISE command-line options**: Configure command-line options for the ISE implementation tools (NGDBuild, MAP, PAR, and TRCE). Use the **More Options** field to specify options that are not listed.

  A brief description of each option is displayed in the lower portion of the dialog box. An asterisk (*) next to an option indicates that the value is changed from the default used by the strategy.

  For more information about specific options, see the *Command Line Tools User Guide (UG628)* cited in Appendix E, Additional Resources.

- **Save Strategy As**: Save the changes to the strategy as a new strategy for use in other projects.

Launching Implementation Runs

To launch Implementation, you can:

- Click the **Run Implementation** button on the Flow Navigator.

- Use the **Flow > Run Implementation** command from the main menu.

- Use the **Run Implementation** command from the toolbar menu.

  Note: The preceding commands will launch the active run in the Design Runs view. See Setting the Active Run, page 290.

  You can also launch a run other than the active run, or launch multiple implementation runs at the same time. Select one or more runs in the Design Runs view using **Shift+click** or **Ctrl+click** to select multiple runs. You can choose both synthesis and implementation runs when selecting multiple runs, and the PlanAhead tool will manage run dependencies, and launch runs in the correct order.

  Use the **Launch Runs** command from the popup menu, or from the Design Runs view toolbar menu, to open the Launch Selected Runs dialog box as shown in Figure 9-6.

- **Launch Directory**: The default launch directory is contained within the local project directory structure. Files for implementation runs are stored at:
  `<project_name>/<project_name>.runs/<run_name>`

  Defining any non-default location outside of the project directory structure makes the project non-portable because absolute paths are written into the project files.

- **Options**:
  - **Launch Runs on Local Host**: Launch the run on the local machine processor.
    - **Number of Jobs**: Define the number of local processors to use when launching multiple runs simultaneously. Individual runs are launched on each processor.
  - **Launch Runs on Remote Hosts** (Linux only): Use remote hosts to launch one or more jobs. See Launching Runs on Remote Linux Hosts in Chapter 9.
    - **Configure Hosts**: Select this option to configure remote hosts.
  - **Generate scripts only**: Export and create the run directory and run script, but do not launch the run at this time. The script can be run at a later time outside of the PlanAhead tool.
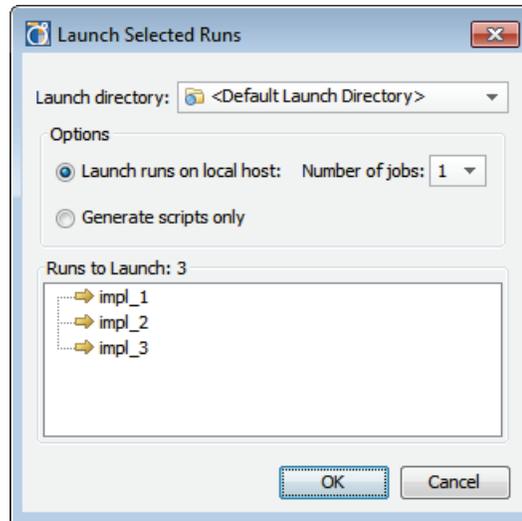
Figure 9-6: **Launch Selected Implementation Runs**

The PlanAhead tool processes the run, and launches implementation, depending on the status of the run. The status is displayed in the Design Runs view, as shown in Figure 9-4, page 290.

- If the status of the run is *Not Started*, the run will begin immediately.

- If the run is in an *Error* state, the PlanAhead tool will first reset the run to remove any incomplete run data, and then will restart the run.

- If the run is *Complete* or *Out-of-Date*, the PlanAhead tool will prompt you to confirm that the run should be reset prior to proceeding with the run.

Running Incremental Implementation

The implementation run consists of a number of smaller processes based on ISE Design Suite tools: NGDBuild, MAP, PAR, TRCE. The PlanAhead tool supports the ability to run implementation as a series of incremental steps, rather than as a single process.

Select a run in the Design Runs view, and use the **Launch Next Step: <*Step*>** command from the popup menu. Where valid <*Step*> values are:

- **NGDBuild**: Reads a netlist file in EDIF or NGC format and creates a logical description of the design as a Xilinx® Native Generic Database (NGD) file.

- **MAP**: Maps a logical design to a Xilinx® FPGA.

- **PAR**: Place and Route the design onto the target Xilinx device.

- **TRCE**: Performs static timing analysis of an FPGA design based on input timing constraints.

- **XDL**: Runs NCD2XDL to read the NCD netlist and write an XDL file.

- **BitGen**: Generates a bitstream for Xilinx device configuration. Although not technically part of an implementation run, the BitGen step is available as an incremental step.

To back up from a completed step, you can use the **Reset to Previous Step: <*Step*>** command from the Design Runs view popup menu. This command resets the selected run from its current state to the prior incremental step. This allows you to step backward through a run, to make any needed changes, then step forward again to incrementally complete the run.

Note: At any time you can use the **Launch Runs** command to complete the implementation run through all remaining steps, or use the **Reset Runs** command to reset to the beginning.

Moving Processes to the Background

As the PlanAhead tool spawns the process to run synthesis or implementation, reading design files and constraint files, the Starting Run dialog box lets you put the process into the background, as shown in Figure 9-7.

When you put a process into the background, it releases the PlanAhead tool to perform certain other functions, like viewing reports, or opening design files, while it completes the background task. You can make use of this time to review previous runs for instance, or examine reports. However, the Tcl Console is blocked, and you will not be able to use Tcl commands, or perform tasks that require Tcl commands, such as switching to another open design.
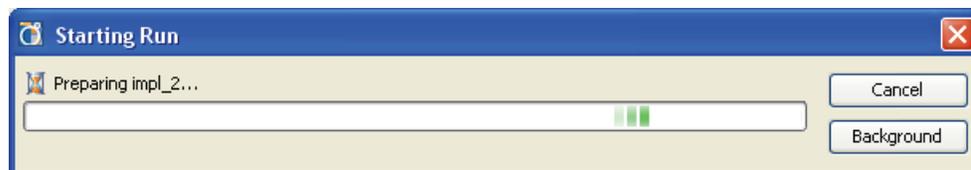
Figure 9-7: **Start Run - Background Process**

Monitoring the Implementation Run

You can monitor the status of a Synthesis or Implementation run in the Log view, reading the compilation information, reviewing warnings and errors in the Messages view, viewing the Project Summary, or opening the Design Runs view.

The following subsections describe the run status monitoring options.

Using the Run Status Display

The status of a run that is in-progress can be displayed in two ways for synthesis and implementation runs. These status displays, as shown in Figure 9-8, indicate that a run is in progress, and provide an opportunity to cancel the run if needed.

1. The Run Status indicator in the upper right corner of the PlanAhead tool environment displays a scrolling bar to indicate the run is in process, and a Cancel button to end the run.

2. The Run Status indicator in the Design Runs view, as shown at the bottom of Figure 9-8, displays a circular arrow to indicate the run is in process. Select the run and use the **Reset Run** command from the popup menu to cancel the run.

Figure 9-8: **Run Status**

Canceling the Run

If you cancel a run that is in-progress, either through the **Cancel** button, or through the **Reset Run** command, the PlanAhead tool will prompt you to delete any run files created during the cancelled run.
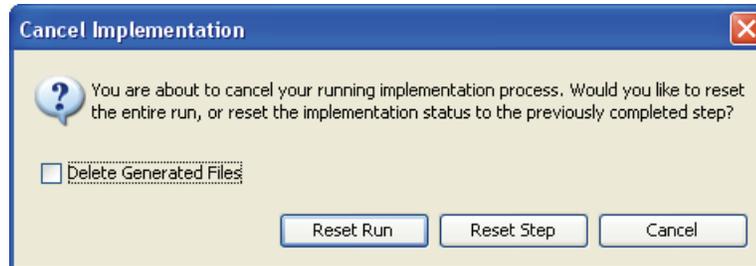
Figure 9-9:   **Cancel Synthesis**

Delete Generated Files will clear the run data from the local project directories. It is a good idea to delete any data created as a result of a cancelled run to avoid any problems with future runs.

Viewing the Log

The Log view opens after you launch a run, and shows the standard output messages. Figure 9-10, page 295 is an example of the log.
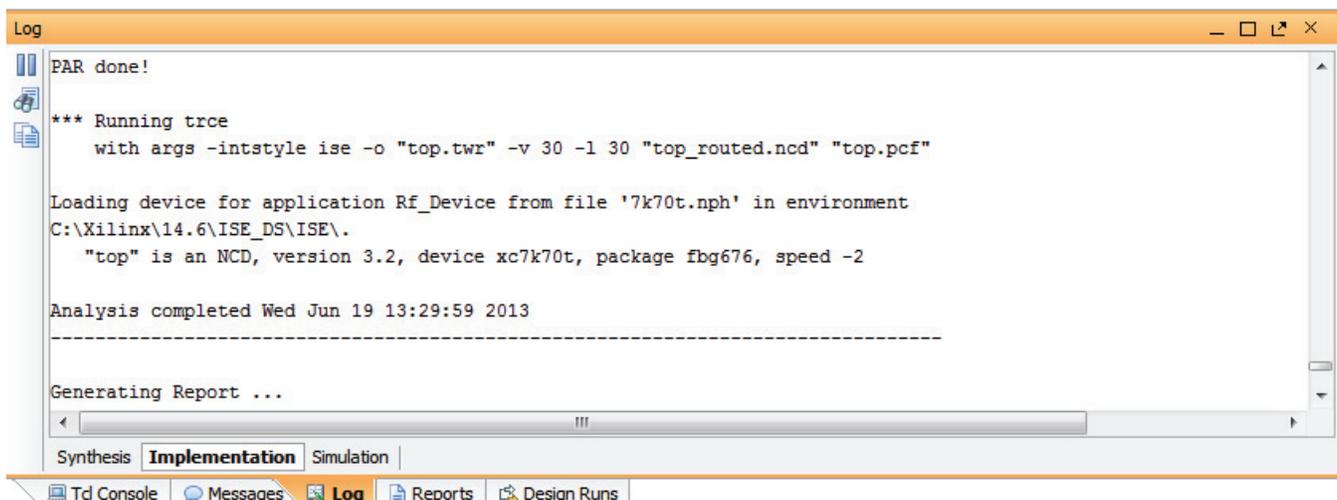
Figure 9-10:   **Log view**

The **Pause** button lets you pause the output to the Log view so that you can scroll and read the log while a command continues running.

The **Find** button opens the Find field at the bottom of the Log view to let you search through the Log text for a specific string.

The **Copy** button lets you copy the selected text in the Log view to paste into a separate file such as a design report or Email.

Determining the Project Status

The PlanAhead tool provides several visual indicators about the overall status of a project as well as methods to take in the next step of the process. Only the results of the major design tasks in the design process are reported in the project status.

The overall project status displays in the Project Summary and in the Status Bar so you can quickly visualize the status of a project upon opening the project, or while you are running the design flow commands. These include RTL Elaboration, Synthesis, Implementation, and Bitstream Generation.

Project Status Bar

The overall project status displays in the upper-right corner of the viewing environment, as shown in Figure 9-11.

Figure 9-11: **Project Status Bar**

As you run the Elaborate, Synthesize, Implement, and Generate Bitstream commands, the Project Status Bar changes to indicate either a successful or failed attempt. Failures are displayed with red text.

If source files change, the project can be marked **Out-of-Date** if either synthesis or implementation was previously completed, as shown in Figure 9-12, page 296. The Status Bar indicates an Out-of-Date Status. Click the **more info** link to display what aspects of the design are out of date. Refer to Updating and Reloading Designs, page 33 for more information.
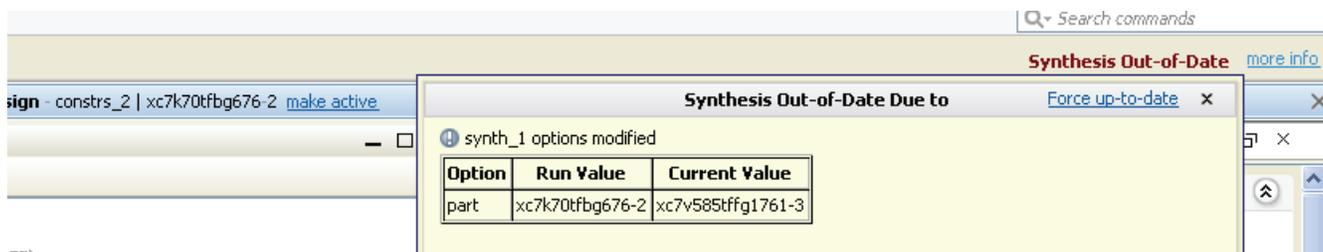
Figure 9-12: **Design Out-of-Date and Reload Banner**

Design Data Out-of-Date Banner

As source files, netlists, constraints, or implementation results update, a banner appears in the top of the open design indicating that a newer version of the design is available than what is loaded in memory. You are prompted to reload what is loaded in memory, as shown in Figure 9-13.

Figure 9-13: **Out-of-Date Banner and Reload button**

Analyzing Implementation Run Results

After Synthesis or Implementation completes, you can view the ISE reports, and open the Netlist or Implemented design to apply timing or physical constraints, analyze the design, and re-implement the run as needed. See Chapter 11, Analyzing Implementation Results, for details.

Viewing Report Files

You can view report files generated by the ISE tools from within the Reports view. The view is usually opened automatically after commands are run. if it is not available, select the Reports link in the Project Summary. The Reports view opens, as shown in Figure 9-14.
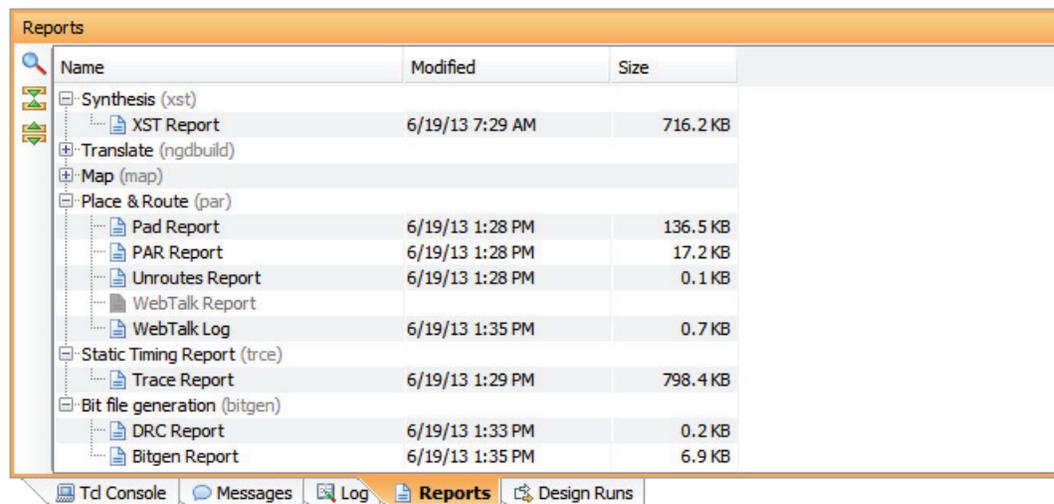
Figure 9-14:   **Selecting Report Files to View**

Select and view any available report files in the workspace, shown in Figure 9-15, page 298.

When viewing reports, you can:

- Browse the report file using the scroll bar.

- Select the **Find** or **Find in Files** buttons to search for specific text.

- Use the **Go to the beginning** or **Go to the End** toolbar buttons to scroll to the beginning or end of the file.

Figure 9-15: **Viewing Report Files**

Viewing Messages

The Messages view provides a filtered list of the Compilation log that includes only the main messages, warnings, and errors. The view has further filtering capabilities using view toolbar buttons to show only errors or warnings. Figure 9-16, page 299 shows an example Messages view. The implementation messages are organized by ISE command and severity.
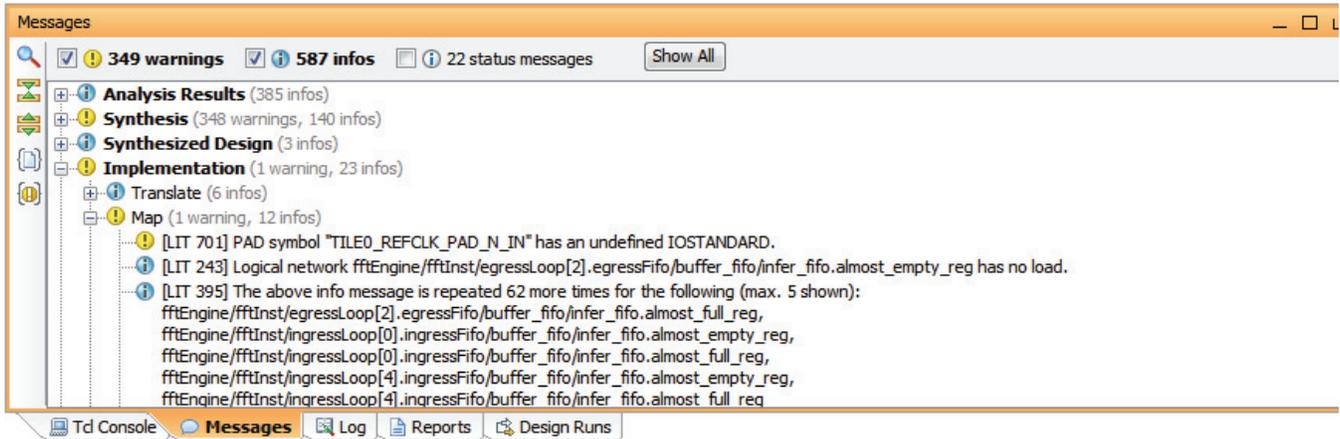
Figure 9-16: **Messages View**

Click the expand and collapse tree widgets to view the individual messages.

You can use select to display Errors, Critical Warnings, Warnings, and Informational Messages in the Messages view by selecting the appropriate checkbox in the banner as shown in Figure 9-17.
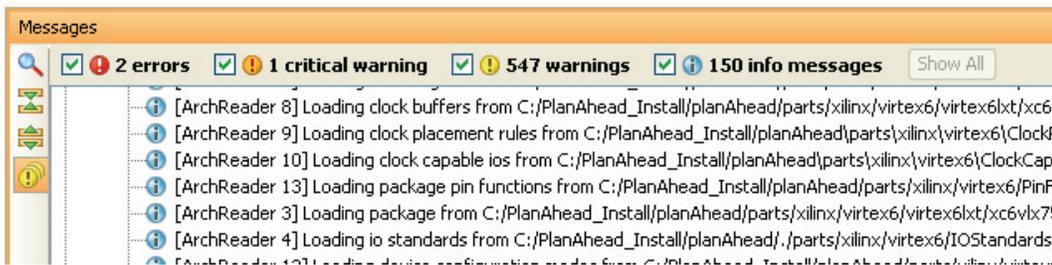
Figure 9-17: **Messages View Banner**

Selecting any of the messages in the Messages view that have a referenced line number automatically opens the RTL file and highlights that line of source code. You can also use the **Search for Answer Record** command from the popup menu to search the Xilinx Customer Support database for related answer records.

Following Implementation

After the run is complete, a dialog box opens that prompts you to take the next step, shown in Figure 9-18.
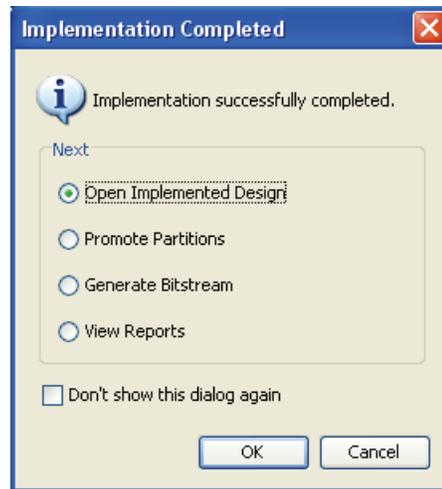
Figure 9-18: **Implementation Completed Successfully Dialog Box**

In the **Implementation Completed** dialog box, select the option, then click **OK**:

- **Open Implemented Design**: Imports the netlist, active constraint set, ISE placement and timing results, and the target part into the design analysis and floorplanning environment so you can perform design analysis and floorplanning. See Chapter 11, Analyzing Implementation Results, for more information.

- **Promote Partitions**: This option is only available when the design has defined partitions. This lets you choose partitions from the design to promote for use in future design iterations. See Chapter 13, Using Hierarchical Design Techniques for more information.

- **Generate Bitstream**: Launches the Generate Bitstream dialog box. See Chapter 12, Generating Bitstream Files, for more information.

- **View Reports**: Opens the Reports view for you to select and view ISE Report files. See Chapter 9, Viewing Report Files, for more information.

Launching Runs on Remote Linux Hosts

The PlanAhead tool ships functionality to allow parallel execution of Runs on multiple Linux hosts. This is accomplished with simplified versions of more robust load-sharing software, such as Grid Engine by Oracle and LSF®.

Job submission algorithms are implemented using a "greedy," round-robin style with Tcl pipes within Secure Shell (SSH).

Limitations on Launching Runs on Remote Linux Hosts

The limitations on launching Runs on remote Linux hosts are:

- Host execution is performed with SSH, a service provided by Linux operating system, and not PlanAhead. For this to work, you must configure SSH so that you are not prompted for a password each time you log in to a remote machine. If you have not configured key-agent forwarding for password-less SSH, or if you have configured SSH and you are prompted for a

password, see Appendix D, Configuring SSH Without Password Prompting.

- Linux-to-Linux hosting is the only supported operating system because of security and lack of remote-shell capabilities on windows systems.

- ISE tool installation is assumed to be available from any login shell, which means that $XILINX and $PATH are configured correctly in your `.cshrc`/`.bashrc` setup scripts. If you can log into a remote machine and enter **map -help** without sourcing any other scripts, this flow works. If you do not have ISE set up upon login (CSHRC or BASHRC), you can use the **Run pre-launch script** option to pass an environment setup script to be run prior to all jobs.

- PlanAhead installation must be visible from the mounted file systems on remote machines. If the PlanAhead installation is stored on a local disk on your own machine, it is not be visible from remote machines.

- PlanAhead project files (`.ppr`) and directories (`.data and.runs`) must be visible from the mounted file systems on remote machines. If the design data is saved to a local disk, it is not visible from remote machines.

Configuring Remote Hosts (Linux Only)

After you have configured SSH, as described in Setting Up SSH Key Agent Forward in Appendix D, the PlanAhead tool enables launching Runs using remote servers. To do so, they must first be configured.

1. To configure a remote host, select one of the following commands:
   - **Tools > Options > Remote Hosts**
   - **Synthesis > Launch Runs > Configure Hosts**
   - **Implementation > Launch Runs > Configure Hosts**
   - **Configure Hosts** in the Launch Selected Runs options dialog box

   The Remote Hosts dialog box displays as shown in Figure 9-19, page 302.

2. Click **Add** to enter the names of remote servers.

3. Toggle the **Jobs** option to specify how many processors on the remote machine to use. Individual Runs are launched on each processor. No multi-threading of processors is used.

4. Toggle the **Enable** option to specify whether to use the server. You can use this field when launching Runs to specify which servers to use for the selected Runs to be launched.

5. Optionally, modify the launch jobs in the field to change the remote access command. The default is `ssh`.

   Note: Be careful when modifying this field. For example, removing 'BatchMode =yes' could cause the process to hang because the shell incorrectly prompts for an interactive password.

6. Optionally, click the **Run pre-launch script** button and define a script to run prior to launching the Runs. Use this option to pass an environment setup script if you do not have ISE set up upon login (CSHRC or BASHRC).

7. Optionally, click the **Run post-completion script** button and define a custom script to run after the run completes.

8. Optionally, click **Send email to**, and enter an email address to send a notification when the run completes.

9. Select one or more hosts, and click **Test** to verify that the server is available and the configuration is properly set.

   Note: It is strongly recommended that you test each host to ensure proper set up.

10. Click **Remove** to delete selected remote hosts, or click **OK** to accept the Remote Host configuration settings.
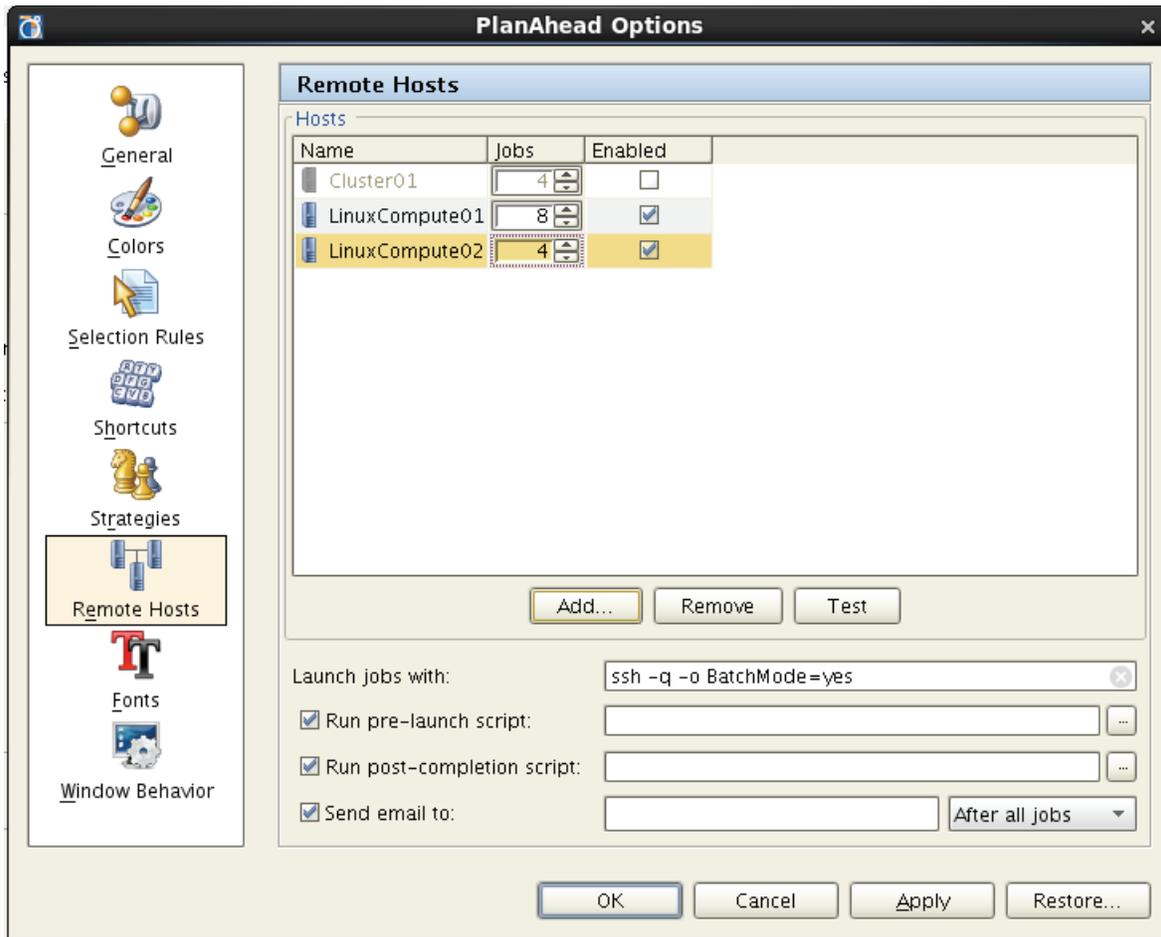
Figure 9-19: **Configure Remote Hosts**

Floorplanning the Design

The PlanAhead™ tool supports a floorplanning methodology that allows you to constrain critical logic to ensure shorter interconnect lengths with less delay.

This methodology requires user interaction with the physical design. It is not a push-button design flow. You can use the analysis capabilities in the PlanAhead tool, coupled with your own knowledge of the design, to define constraints and set tool options for improving performance.

Floorplanning can be accomplished by:

- Creating physical block (Pblock) locations to constrain logic placement.

  or

- Locking individual logic objects to specific device sites.

The Floorplanning View shows the more common views used during floorplanning. To open the Floorplanning View, select:

- **Layout > Floorplanning**.

  or

- Select **Floorplanning** from the Layout Selector on the toolbar menu.

The complexities of floorplanning require more explanation than is practical to provide in this user guide. For detailed information, see the *Floorplanning Methodology Guide (UG633),* cited in Appendix E, Additional Resources.

Working with Pblocks

The process of floorplanning begins by dividing some or all of the logic in the design into groups and constraining that logic. The PlanAhead tool provides the ability to hierarchically divide the design into smaller, more manageable physical blocks (Pblocks). These Pblocks can include logic modules and primitive logic from anywhere in the logic hierarchy. You can group critical or associated logic together into a single Pblock, which prevents logic migration, limits interconnect lengths, and reduces delays.

The following subsections describe how to work with Pblocks.

Creating Pblocks

Creating a Pblock results in an `AREA_GROUP` constraint that is written into the exported UCF constraint file. The constraints in the PlanAhead tool reflect the assigned logic, specified ranges, and defined attributes.

Using the Draw Pblock Command

The Draw Pblock commands assigns pre-selected logic to a new Pblock in the Device view. You can select the logic to assign to the Pblock before you invoke the command.

To create a Pblock:

1. Select the logic in any view, such as the Netlist view, to assign to the Pblock.
2. Click **Draw Pblock**, in either the Device view popup menu or toolbar.
3. Move your cursor to the location in the Device view to draw a Pblock.
4. Press and hold the left mouse button, move the mouse cursor to the opposite corner of the Pblock, and release the mouse button.

   The New Pblock dialog box opens as shown in Figure 10-1.
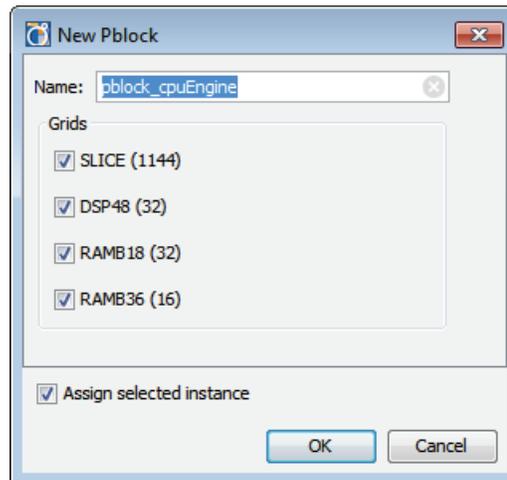
Figure 10-1: **New Pblock Dialog Box**

5. Edit the options, and click **OK** when done. Options are:
   - **Name**: Enter a name for the Pblock. If no name is entered, a default name of `pblock_n` or `Pblock_instancename` is used.
   - **Grids**: Select the different types of devices within the specified area to be constrained by the Pblock.
   - **Assign selected instances**: Assign the selected instances to the new Pblock.

     Note: Occasionally, logic is inadvertently selected, which is not intended for assignment. Take care to avoid this error.

The Pblock displays, and then you can select it in the Device view and the Physical Constraints view. The Physical Constraints view opens in the Floorplanning view layout, or you can open it using **Window > Physical Constraints**.

The initial Pblock size and location are not critical during manual creation. Pblocks can be appropriately sized by viewing the Physical Resource Estimates in the **Statistics** tab of the Pblock Properties view. The location of a Pblock's rectangles is displayed under the **Rectangles** tab of the Pblock Properties view. You can also determine how to reposition Pblocks relative to other Pblocks by viewing the connectivity in the Device view.

Sometimes, it is helpful to initially create all of the Pblocks with small rectangles to visualize the logic connectivity flow between the Pblocks prior to attempting sizing as shown in Figure 10-2, page 305.
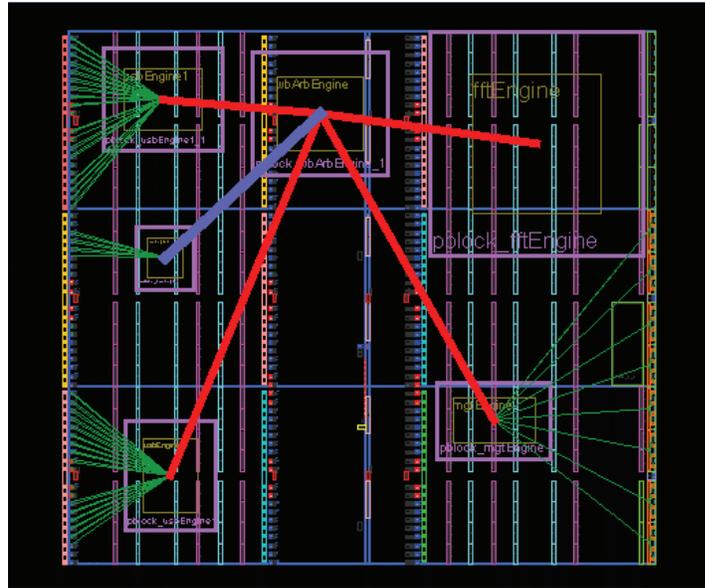
Figure 10-2: **Pblock Connectivity Analysis in the Device View**

See Analyzing Hierarchical Connectivity, page 217 and Using the Show Connectivity Command, page 346 for more information about this view.

Using the New Pblock Command

The New Pblock command creates a new Pblock in the Physical Constraints view, but does not create a rectangle in the Device view.

You must pre-select logic for assignment to the new Pblock. If no logic is pre-selected, the command creates an empty Pblock.

To create new Pblocks with or without pre-selected logic, select **New Pblock** from the popup menu.

Creating Multiple Pblocks with the Create Pblocks Command

You can create multiple Pblocks in a semi-automated way by using the **Create Pblocks** wizard. The wizard creates a separate, unplaced Pblock for each selected netlist instance.

To use the wizard, pre-select a set of instances for inclusion into individual Pblocks.

To create multiple Pblocks for specific netlist instances:

1. Select the instances to include in the Pblocks.

2. Select **Tools > Floorplanning > Create Pblocks**.

   The Create Pblocks wizard opens with a list of the selected instances as shown in Figure 10-3, page 306.
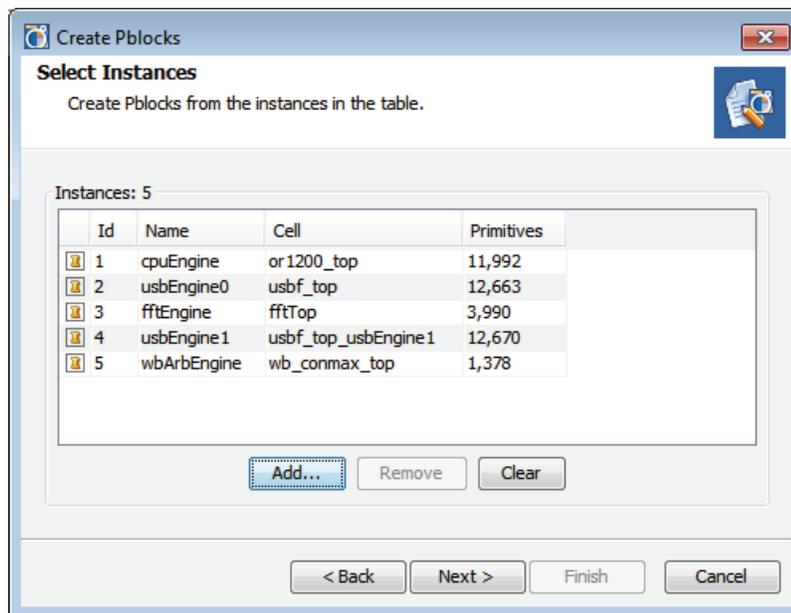
Figure 10-3: **Create Pblocks Wizard: Create Pblocks from Instances**

- To add additional netlist instances to this list, click **Add**, which invokes a browser in which you can select other instances.

- To remove any netlist instances from the list, click **Remove**.

- To clear netlist instances from the list, click **Clear**.

3. Click **Next**.

   The Create Pblocks wizard lets you specify a naming scheme.

4. In the Create Pblocks wizard, edit the naming scheme fields:

   - **Prefix**: Defines a name prefix used for naming Pblocks. The default is pblock_.

   - **Suffix**: Appends selected characters to the specified prefix.

     - **Instance Name**: Appends the instance name to the pblock name.

     - **Number**: Numbers pblocks sequentially from pblock_1.

5. Click **Next**.

6. Verify the contents in the Summary page.

7. Click **Finish** to create the Pblocks with these settings.
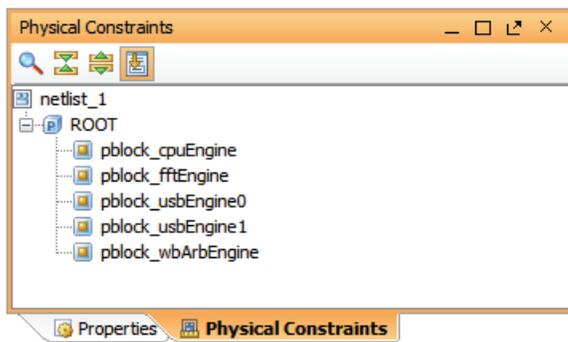
Figure 10-4: **Pblocks in Physical Constraints**

The Pblocks show in the Physical Constraints view, as represented in Figure 10-4, page 306.

To create rectangles for the newly-created Pblocks:

1. Select each of the new Pblocks (one at a time) in the Physical Constraints view.

2. In the Device view, select **Set Pblock Size** from the popup menu.

3. Draw a rectangle in the Device view.

Creating Nested Pblocks

You can create Pblocks within Pblocks (nested Pblocks) to provide further control for constraining logic. This can be helpful when trying to improve performance of critical modules. The top-level Pblock contains all the lower-level Pblocks during utilization estimates.

Note: The ISE® implementation software does not support extensive use of this feature. Occasionally, map and placement errors result when creating nested Pblocks.

Creating Pblocks with Multiple Rectangles

You can also create Pblocks with multiple rectangles to create non-rectangular Pblocks, or to cover distant device resources without creating a single monolithic Pblock.

With an existing Pblock selected, use the **Add Pblock Rectangle** command from the popup menu to add a rectangle to an existing Pblock.

Pblocks consisting of multiple rectangles display with dashed lines connecting the rectangles to indicate that they are part of the same Pblock. The assigned instance rectangles and connectivity display in the largest rectangle, as shown in Figure 10-5.
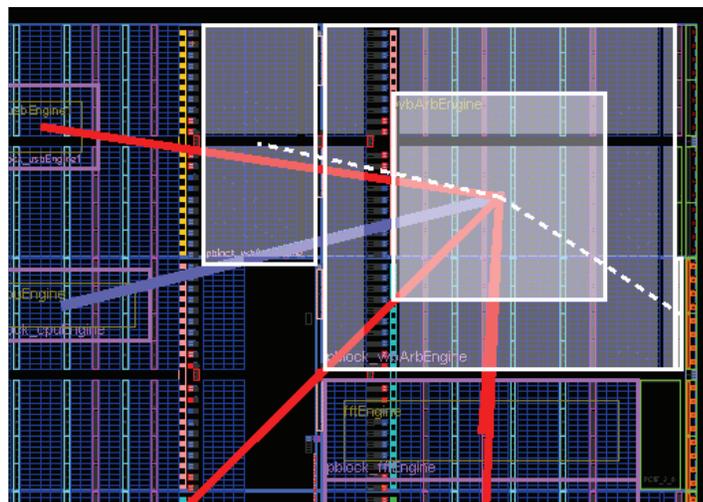
Figure 10-5: **Creating Pblocks with Non-Rectangular Shapes**

Creating Clock Region Pblocks

A Pblock is defined to contain the specified device resources for the region bounded by the Pblock boundary. You can also define a Pblock to include all resources within a specific clock region or clock regions.

To define a Pblock as a clock region in the Device view:

1. Draw a Pblock with a rectangle that encompasses the boundary of the clock region.

The PlanAhead tool displays the clock region boundaries. To change the color or display characteristics of the clock region boundaries, refer to the Customizing Display Themes in Chapter 4.

Figure 10-6 shows a clock region Pblock. The tool tip indicates the Pblock range covers a clock region.
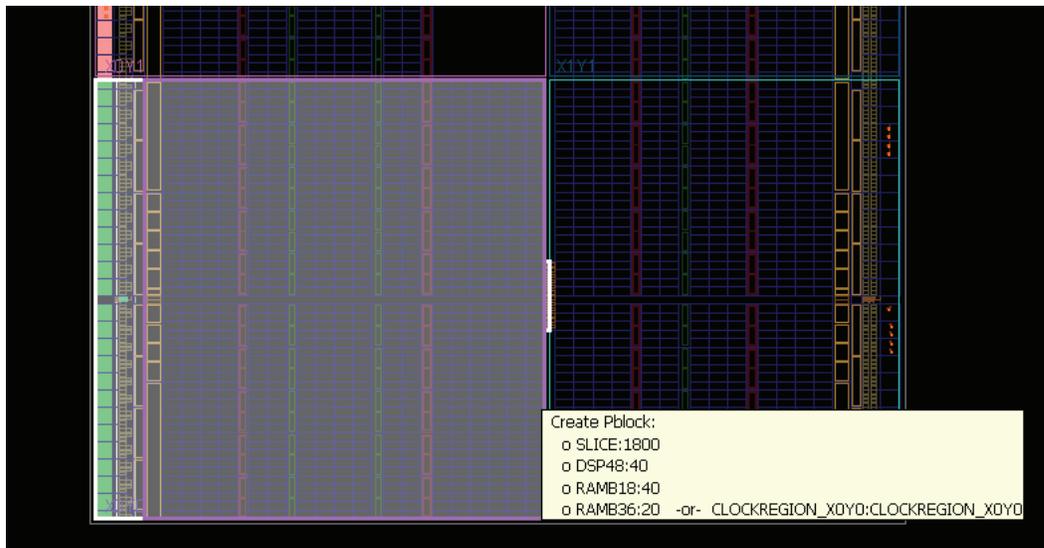
Figure 10-6: **Creating Clock Region Pblocks**

2. In the **Set Pblock** dialog box, as shown in Figure 10-7, page 308, select **OK** to define the Pblock range as the clock region (`CLOCKREGION_X`).

The Pblock rectangle must encompass the clock region boundary to enable the `CLOCKREGION` option in the **Set Pblock** dialog box. When you de-select `CLOCKREGION_X` on the dialog box, you can define the Pblock using traditional logic-based device resources.

Note: You can toggle between the clock region and logic-based Pblocks by selecting or deselecting `CLOCKREGION` in the Set Pblock dialog box or in the Pblock Properties view.

The Pblock clock region coordinates display in the Pblock Properties view.
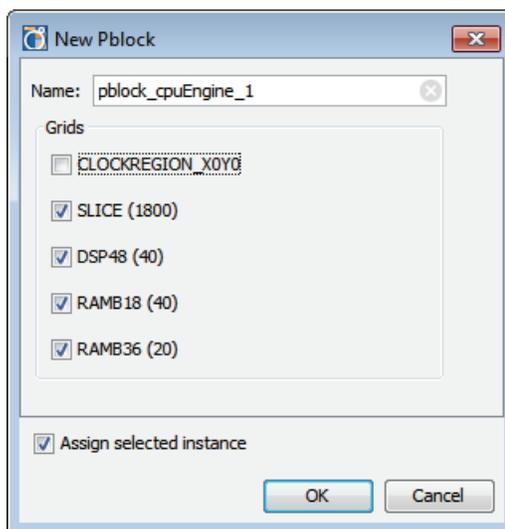
Figure 10-7: **Set Pblock Dialog Box to Confirm Pblock as Clock Region**

Understanding Pblock Graphics

The default display options show the Pblock and the instances assigned to the Pblock.

- The outer rectangle is the Pblock border, showing the area on the FPGA covered by the Pblock.

- The inner rectangle reflects the ratio of device resources assigned to the Pblock.

You can place multiple instances into a Pblock, and instance rectangles displayed inside the Pblock are sized based on the amount of logic they contain, relative to the other instances in the same Pblock.

If many instances are assigned to a Pblock, they might appear as lines instead of rectangles, as shown in Figure 10-8.
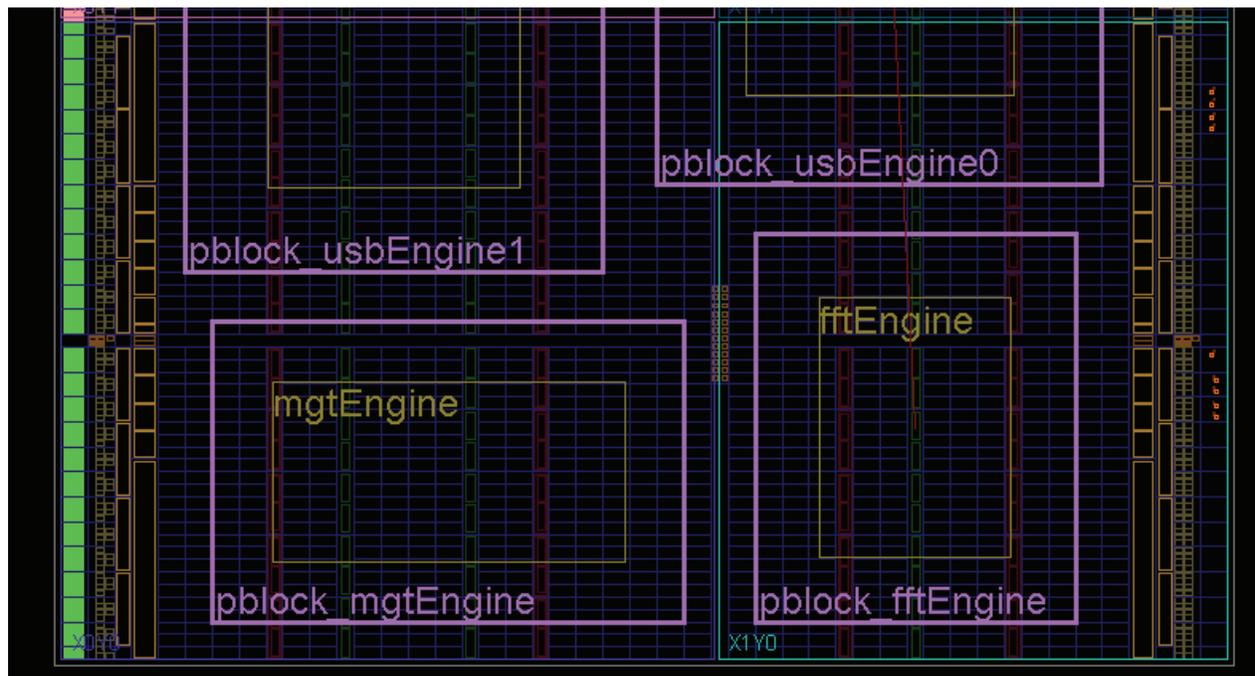
Figure 10-8: **Pblocks With Assigned Instances Displayed Graphically**

Using the default selection rules, selecting the Pblock rectangle also selects all of the netlist instances assigned to it. You can drag instances from one Pblock and assign them into another Pblock. To modify Selection Rules for the tool, select **Tools > Options**, then choose **Selection Rules** from the menu on the left of the Options dialog box.

Note: When manipulating Pblocks take care to ensure the Pblock rectangle is selected and not the assigned instances in the Pblock. It is helpful when manipulating Pblocks to turn off the selection ability for instances. This ensures Pblocks are selected in the Device view and not the instances assigned to them.

PlanAhead draws the I/O nets connected to the center of the instance inside the Pblock rather than in the Pblock center, as shown in Figure 10-9, page 310.
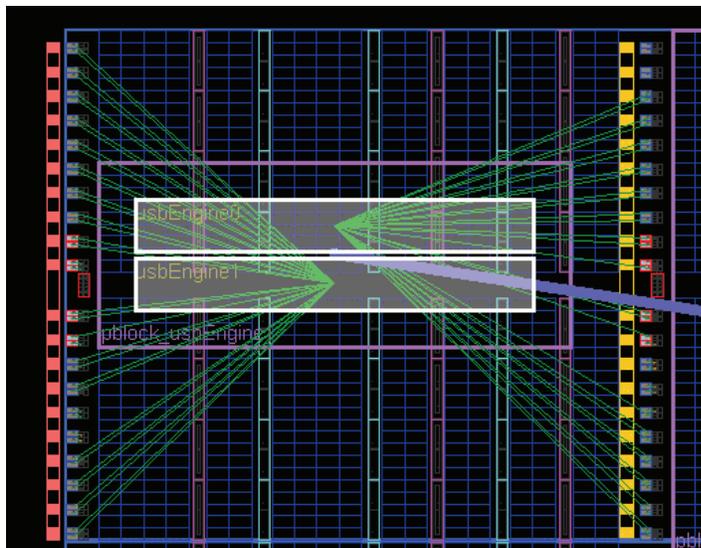
Figure 10-9: **I/O Connectivity Displays to Center of Instance Rectangles**

Child Pblocks appear in a different color to differentiate the rectangles. You can configure the color configuration of objects like Pblocks and children Pblocks using:

- **Tools > Options**
- **Themes** from the Options dialog box
- The **Device** tab of the Themes, and defining the settings for Pblocks

Viewing Pblock Properties

To display or edit Pblock properties, select the Pblock and examine the Pblock Properties view, as shown in Figure 10-10.
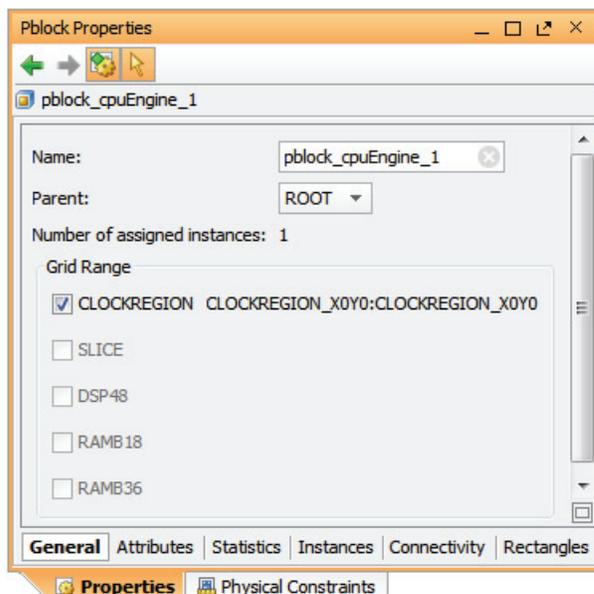
Figure 10-10: **PBlock Properties**

Table 10-1, page 311 lists the tabs in the Pblock Properties view, and information presented.

Table 10-1: **Pblock Properties Tabs and Options**

Tab	Options
General	• **Name**: Displays the Pblock name. • **Parent**: Displays the Parent Pblock. This field is a non-editable field for some Pblocks. If a Pblock has multiple potential parent Pblocks, the field becomes active allowing definition of the Parent Pblock. • **Number of assigned instances**: Indicates the number of logic elements that are assigned to the Pblock. • **Grid Range**: Enables the Pblocks to be specified with specific AREA_GROUP RANGE properties. Selecting specific ranges constrains only the selected logic types within the Pblock area. The grid range coordinates display for each logic type after the Pblock is created. • **CLOCKREGION**: Defines the Pblock range to be an entire clock region. The Pblock rectangle is drawn to match the clock region boundary.
Attributes	Lets you define Attribute properties to a Pblock. See Setting Attributes for Pblocks, page 320.

Statistics	Pblock statistics will probably differ from the final implemented design. The Pblock properties are for estimation only. For instance, Pblock statistics indicate that under ideal packing conditions the tool could pack a certain design into 995 of the 32700 available SLICEMs on the device. Therefore, a device with less than 995 SLICEMs will not fit the design. However, during implementation the logic may be distributed around the available device resources, and the minimum required 995 SLICEMs could expand to 1200, as an example, in the implemented design. Some of the utilized SLICEMs might have 2 LUTs used, while others have 3 or 4 LUTs used.
	• **Physical Resources Estimates**: A chart of each resource type in the device.
	  • **Site Type**: The site types defined within the Pblock rectangle.
	  • **Available**: The number of sites contained in the Pblock.
	  • **Required**: The number of sites required for the logic assigned to the Pblock.
	  • **% Util**: The estimated percentage of the sites utilized in the Pblock.
	• **Carry Statistics**: The number of vertical carry chain logic objects assigned to the Pblock. It also displays the tallest carry chain assigned to the Pblock and the percentage of its height in relation to the Pblock height. Carry height utilization values over 100% can cause PlanAhead DRC errors and ISE map errors.
	• **Clock Report**: Clock signals (Local, Global, and Resource) contained in the Pblock as well as the number of clocked instances on each clock.
	• **RPM Statistics**: The number of Relatively Placed Macros (RPM) objects assigned to the Pblock. It also displays the tallest and widest RPM assigned to the Pblock and the percentage of its size in relation to the Pblock size.
	Note: RPM utilization values over 100% causes PlanAhead DRC errors and ISE map errors. PlanAhead does not indicate whether multiple RPMs can fit inside the Pblock rectangle.
	• **Clock Region Statistics**: The utilization percentage of each clock region that the Pblock overlaps.
	• **Primitive Statistics**: The number of each type of logical resource assigned to the Pblock.
	Note: You can save the contents to a text file by clicking on the **Export Statistics** icon.

Instances	Displays information about the instances contained in the Pblock. You can select the instance fields and use them to seed many popup menu commands.
Connectivity	Lists the nets connecting to cells inside the Pblock.
Rectangles	Displays information about the various rectangles created for the Pblock. The Rectangle tab is used for selecting rectangles of a Pblock. Refer to the Using Non-Rectangular Pblocks, page 319.

Note: To accept changes, click **Apply,** to cancel changes click **Cancel**. Selecting another item or closing the Properties view does not initiate any changes unless you click **Apply**.

Configuring Pblocks

The following subsections describe configuring Pblocks.

Setting Pblock Logic Type Ranges

To set Pblock `AREA_GROUP` range types in the Pblock Properties view, modify the Grid Range options in the General tab. Adjusting these options controls which type of logic is constrained within the Pblock rectangle, as shown in Figure 10-11.
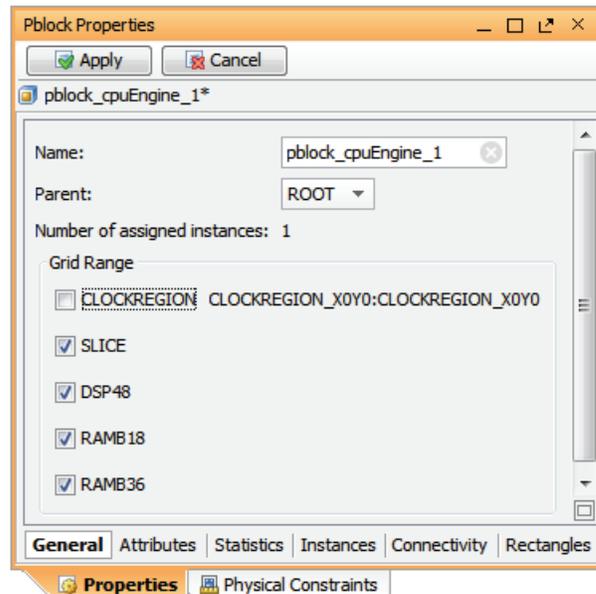
Figure 10-11: **Setting AREA_GROUP Range Types**

If you resize a Pblock or move it to a location that includes new device logic types, such as block RAM and DSP, a dialog box displays prompting you to add the new range types to the Pblock definition. The contents of this dialog box varies, based upon the location of the Pblock. Toggling the ranges *off* results in the Pblock showing differently in the Device view.

Note: The Set Pblock dialog box will only report the Clock Region range if the Pblock encompasses a clock region on the device.

As you select the Pblock, the shading affects only the logic types for the ranges set on the Pblock, as shown in Figure 10-12, page 313.
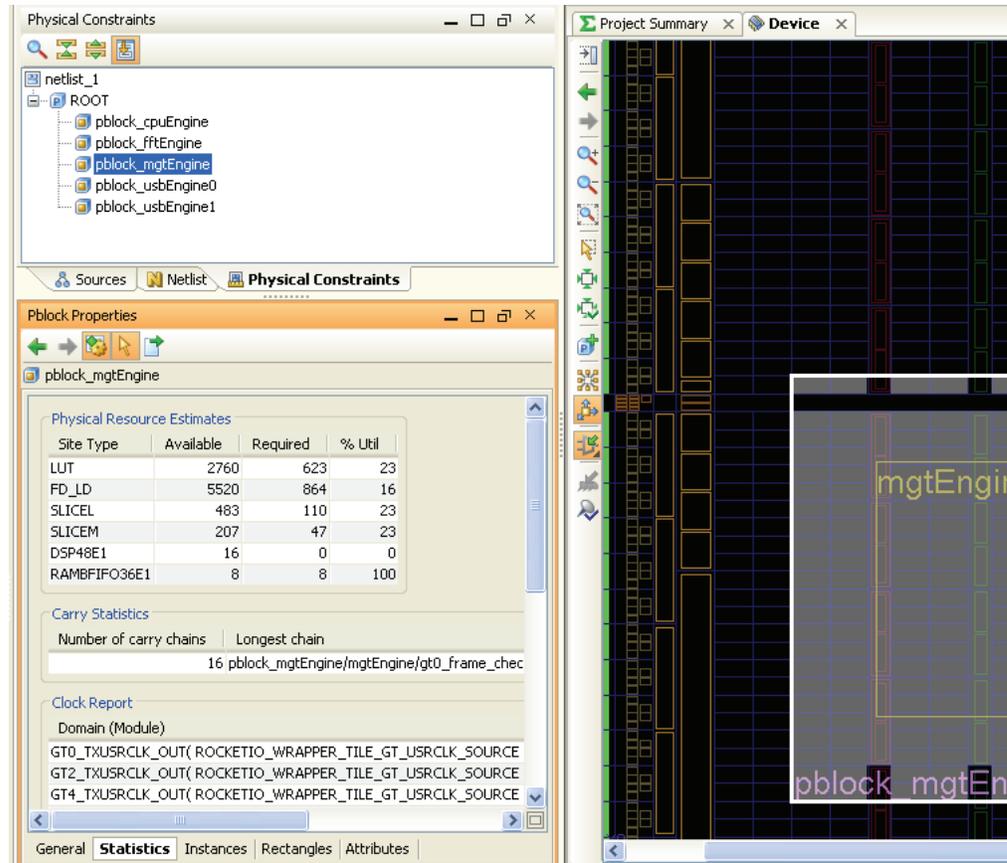
Figure 10-12: **Pblock Shading Reflects Logic Contained in Pblock**

Assigning Logic to Pblocks

After you create a Pblock, you can assign netlist instances by either dragging and dropping logic, or by using the **Assign** popup command.

To use the drag and drop method:

1. Click and drag logic instances from the Netlist, Schematic, Hierarchy, or Find Results views.

2. Drop the instances into the Pblock rectangle area.

To use the Assign command method to assign logic to existing Pblocks:

1. Select logic instances in the Netlist view.

2. Select **Assign**.

   The Select Pblock dialog box displays the allowable selections of the Pblock assignment, as shown in Figure 10-13, page 314.
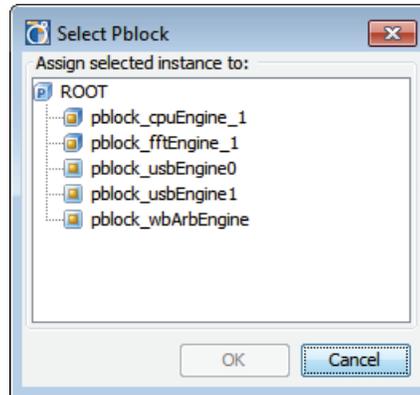
Figure 10-13: **Select Pblock Dialog Box**

Unassigning Logic from Pblocks

To remove instances from Pblocks:

1. Select the instances.

2. Select the **Unassign** popup command.

A confirmation dialog box opens asking you to confirm the removal of instances from the Pblock.

Moving and Resizing Pblocks

The following subsections describe how to move and resize Pblocks.

Moving a Pblock

To move a Pblock, select and drag the Pblock within the Device view, and drop it in the new location. The dynamic, hand-shaped cursor indicates that the Pblock is selected for moving. Ensure that you have selected the outer Pblock rectangle, and not one of the assigned instances.

If you move the Pblock to a location that includes new device logic types, such as a block RAM or a DSP, a dialog box displays prompting you to add the new range types to the Pblock definition.

Pblocks behave differently when assigned logic has a BEL or LOC placement constraint inside the Pblock borders. The target location must contain adequate resources to accommodate the placement constraints. As you move the Pblock, the cursor indicates which sites are legal for a move, based upon placement requirements. If you attempt to move a Pblock to an inadequate location, the Choose LOC Mode dialog box displays as shown in Figure 10-14, page 315, prompting you to either remove the location constraints or leave them intact.

The available actions are:

- **Move location constraints with the Pblock site ranges**

  This option moves all currently placed instances, whether fixed or unfixed, reassigning their LOC and BEL constraints to new locations relative to the movement of the Pblock. See Understanding Fixed and Unfixed Placement Constraints, page 324 for more information.

  This mode is useful for preserving the placement of instances within the Pblock relative to each other.

- **Move unfixed location constraints with the Pblock site ranges**

  This option moves currently placed but unfixed instances, reassigning their LOC and BEL constraints to new locations relative to the movement of the Pblock. This leaves placed and fixed instances in their current location.

  This mode is useful for preserving the placement of fixed instances in their current location, to avoid disrupting placement, while moving any unfixed locations with the Pblock.

- **Leave all location constraints in their current position**

  This leaves all placed logic instances in their current location, removing the logic from the Pblock, and moving the Pblock and all remaining logic to the newly selected site range.

  This mode is useful to preserve placement of all placed instances when many instances have been placed, while still allowing the Pblock to be relocated to a new region of the device.
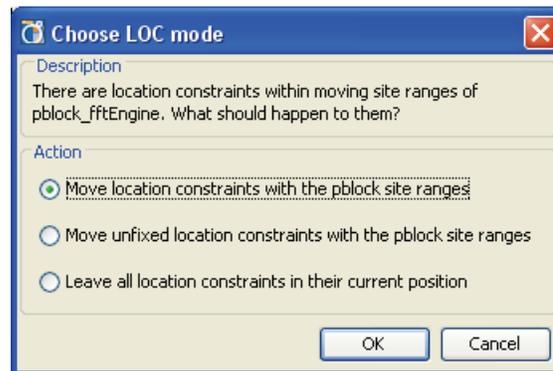
Figure 10-14: **Moving Pblocks with LOCs**

Note: If you have difficulty moving Pblocks, click **Set Pblock Size** to redraw the rectangle elsewhere. You might need to remove placement constraints prior to moving Pblocks.

Resizing a Pblock

To stretch the edges of a Pblock, select the Pblock, then move the cursor near one of its edges or corners.

- When the cursor changes to a drag symbol, click and drag to reshape the Pblock.
- To cancel an active stretch operation, press **Esc**.

When you have resized the Pblock to make it smaller, some of the placed instances assigned to the Pblock may now fall outside the boundary. In this case, the Choose LOC Mode dialog box will open with the following options, as shown in Figure 10-15, page 316:

- **Leave all location constraints in their current position**

  This leaves all placed logic instances in their current location, removing any logic that falls outside the Pblock boundary from the Pblock.

- **Delete location constraints on sites trimmed from Pblock**

  This unplaces any logic, either fixed or unfixed, that falls outside of the resized Pblock, but leaves the instances assigned to the Pblock. Unplaced instances can now be placed relative to the new Pblock.

- **Delete unfixed location constraints on sites trimmed from Pblock**

  This unplaces any unfixed logic that falls outside of the resized Pblock, but leaves fixed instances placed in their current location. This preserves the placement of fixed logic, but unassigns the instances from the Pblock.
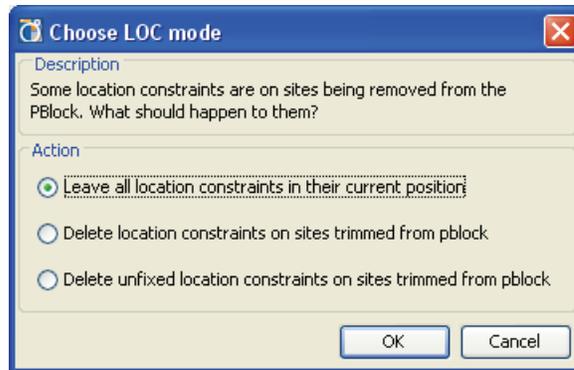
Figure 10-15: **Resizing Pblocks with LOCs**

Using the Set Pblock Size Command

You can size or resize an existing Pblock using the **Set Pblock Size** command to create a new rectangle rather than resize the current one. You can also use this command to draw a rectangle for an existing Pblock that has no rectangle defined, such as one created with the **New Pblock**(s) commands. For more information, see Creating Multiple Pblocks with the Create Pblocks Command, page 305.

To create a new rectangle for an existing Pblock:

1. In the Physical Constraints view or Device view, select the Pblock.

2. Click the **Set Pblock Size** command from the popup menu.

   The cursor changes to enable you to draw a new rectangle in the desired location in the Device view.

   If the selected Pblock has multiple rectangles, this command regenerates the Pblock with a single rectangle. This can be useful when a Pblock gets fragmented into multiple rectangles as the floorplan is developed.

3. Draw a new rectangle.

   Note: Press the **Esc** key to cancel the command without creating a new Pblock.

When you have resized the Pblock with a new rectangle, some of the placed instances assigned to the Pblock may now fall outside the boundary of the new Pblock. In this case, the Choose LOC Mode dialog box will open, as shown in Figure 10-16, with the following options:
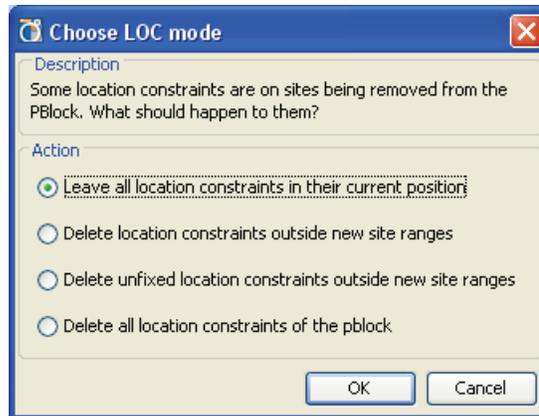
Figure 10-16: **Set Pblock Size with LOCs**

- **Leave all location constraints in their current position**

  This leaves all placed logic instances in their current location, removing any logic that falls outside the Pblock boundary from the Pblock.

- **Delete location constraints outside new site ranges**

  This unplaces any logic, either fixed or unfixed, that falls outside of the new Pblock boundary, but leaves the instances assigned to the Pblock. Unplaced instances can now be placed relative to the new Pblock.

- **Delete unfixed location constraints outside new site ranges**

  This unplaces any unfixed logic that falls outside of the new Pblock boundary, but leaves fixed instances placed in their current location. This preserves the placement of fixed logic, but unassigns the instances from the Pblock.

- **Delete all location constraints of the Pblock**

  This unplaces all placed instances in the Pblock, whether they fall inside or outside of the new boundary. This allows the Pblock to be placed over again without working around prior placed instances.

Using Resource Utilization Statistics to Shape Pblocks

You can use the utilization estimates in the Pblock Properties view to size and place Pblocks. The tool estimates resources required by the logic assigned to the Pblock and compares that against available device resources to compute utilization estimates.

To display the utilization estimates for a Pblock:

1. Select the Pblock, and view the Pblock Properties.
2. Click the **Statistics** tab, shown in .

Figure 10-17: **Pblock Properties View: Statistics Tab**

3. In the Statistics tab, view the utilization estimates columns.

   • **Available**: Displays the number of available sites in the Pblock.

   • **Required**: Displays the number of sites required by the assigned logic.

   • **% Util**: Displays the estimated utilization percentage for each logic type. The appropriate utilization can be met by resizing the Pblock. If utilization for logic objects is over 100%, the text appears in red, as shown in Figure 10-17.

4. Scroll down to view the required RAM sites for the Pblock.

   The Pblock Properties view is dynamic and updates each time you modify the Pblock.

   If the Pblock does not contain a site for a specific logic device element, the dialog box shows the following values:

   • **Available**: 0.

   • **Required**: The required number.

   • **Utilization**: Percentage of available resources used by the logic in the Pblock.

      - A value of **Disabled** means that the Site Type is disabled (on the General tab) and is not available for use. If there are sites of this type required by the Pblock, it is an error condition.

      - A value of **No Sites** means that the Pblock range on the device does not include any sites of that type.

Note: The Pblock SLICE utilization calculation assumes maximum site utilization. Realistically, the maximum site utilization is rarely achieved in placement and routing tools. Designers should optimize for a target utilization of approximately 80% or higher. This number is a function of the device used and the characteristics of the design and its constraints.

Note: Pblock utilization is affected by carry chains, RPM macros, and the geometry of the Pblock rectangle. These statistics are estimates to help guide you to a successful ISE implementation. All Pblock Statistics must be taken into account when sizing Pblocks. Occasionally, Pblocks must be enlarged for design tools to place them successfully.

Placing Pblocks Based on Connectivity

The PlanAhead tool provides dynamic connectivity feedback to help guide placement of Pblocks. Figure 10-5, page 307 shows an example of a connectivity display.

The combined connectivity between Pblocks show as bundled nets. PlanAhead sizes and colors each bundle, based upon the number of connections between Pblocks so that the heavily connected Pblocks are easy to identify.

A reasonable strategy is to define the Pblocks with the largest net bundles close together. Typically, the Pblocks are placed to achieve the shortest net lengths, and to avoid routing conflicts or congestion.

Displaying Bundle Net Properties

You can view connectivity information by displaying properties for net bundles or for individual nets. To view connectivity information:

1. Select the net or bundle net.

2. View the Net Properties or Bundle Net Properties view.

   The Nets tab in the **Bundle Net Properties** view displays the nets contained in the bundle as shown in Figure 10-18.

	Id	Name	Instance Pins	Flat Pins
	1	usbEngine0/wb_ack_o/Q	1	3
	2	usbEngine0/susp_o/Q	1	3
	3	usbEngine0/u4/inta/Q	1	4
	4	wbArbEngine/s0/s0_we_o	1	2
	5	wbArbEngine/s0/s0_cyc_o	1	2
	6	wbArbEngine/s0/s0_stb_o	1	2
	7	usbEngine0/wb_data_o_31/Q	1	3
	8	usbEngine0/wb_data_o_30/Q	1	3
	9	usbEngine0/wb_data_o_29/Q	1	3
	10	usbEngine0/wb_data_o_28/Q	1	3
	11	usbEngine0/wb_data_o_27/Q	1	3
	12	usbEngine0/wb_data_o_26/Q	1	3

Figure 10-18: **Bundle Net Properties: Nets Tab**

Adjusting Bundle Net Defaults

You can define the color, line width, and signal count range in the view-specific settings of the Themes dialog box (**Tools > Options > Themes > Bundle Nets.**)

Using Non-Rectangular Pblocks

PlanAhead supports creating, modifying, and deleting non-rectangular Pblock shapes with multiple rectangles per Pblock. It is recommended that you use this approach sparingly and only when necessary because non-rectangular Pblock shapes can increase the possibility of error in downstream tools.

Creating Non-Rectangular Pblocks

To add additional rectangles to existing Pblocks with rectangles, use **Add Pblock Rectangle**.

Pblocks with multiple rectangles appear as separate rectangles with a dashed line connecting them as shown in Figure 10-5, page 307.

Modifying Non-Rectangular Pblocks

The PlanAhead tool is not optimized to handle too many ranges per AREA_GROUP constraint. If a single rectangular Pblock will not work, it is best to use simple shape configurations, such as L or T shapes, made up of two rectangles.

When you select a Pblock with multiple rectangles defined, all of the rectangles are selected. They can be moved individually, or together as a group.

To reshape one of the rectangles of a multiple rectangle Pblock, select a rectangle and use the **Set Pblock Size** command, or resize it manually.

To select or resize a rectangle individually, use one of the following methods:

• For a single Pblock rectangle, click **Select** in the popup command.

• To select the rectangle, then select the **Pblock Properties Rectangles** tab.

Defined Pblocks that span PowerPC® (405 and 440) processors, serial transceiver sites, or configuration blocks might receive multiple rectangle regions automatically. This is done to enable the correct rectangle ranges to be defined for implementation.

Removing a Pblock Rectangle

To remove the Pblock rectangle, select the Pblock and click **Clear Rectangle**. You can clear:

• Individual Pblock rectangles one at a time

• Multiple rectangles and Pblocks simultaneously

Clearing Pblock rectangles does not delete the Pblock from the physical constraints.

Setting Attributes for Pblocks

You can assign attributes to Pblocks in the **Attributes** tab of the **Pblock Properties** view as shown in Figure 10-19. Pblock attributes define options for ISE which can affect the implementation results and cause failures.
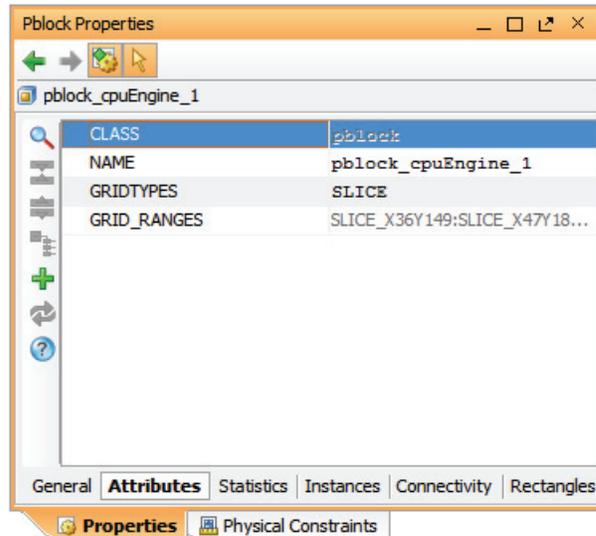
Figure 10-19: **Pblock Properties View: Attributes Tab**

To define new attributes for the Pblock:

1. In the **Attributes** tab of the Pblock Properties View, select **Add pre-defined attributes** from the popup menu, or the toolbar button.

    The Add Pre-defined Attributes dialog box opens, as shown in Figure 10-20.
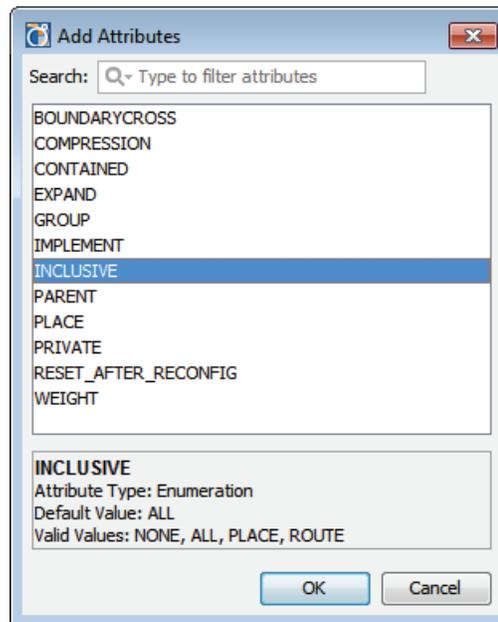
Figure 10-20: **Add Predefined Attributes**

2. Select the attribute to assign, then click **OK**.

    The specified attribute type is added into the Attributes tab.

3. You can then specify an attribute value, and click **Apply** to accept the changes.

Renaming a Pblock

You can rename Pblocks using the General tab of the Pblock Properties view. Enter the new Pblock name in the Name field and click **Apply**. Alternatively, you can modify the **Name** attribute on the Attributes tab of the Pblock Properties view.

Deleting a Pblock

You can delete selected Pblocks as follows:

1. In the Physical Constraints view, select one or more Pblocks.

2. Press **Delete**.

3. In the **Confirm Delete** dialog box, select the **Remove Pblock children** option to remove any nested Pblocks along with their partitions. Otherwise, when left unselected, you delete the selected Pblock only, and move any nested Pblocks up one layer of hierarchy.

4. Click **OK** to remove the Pblock partition from the Physical Constraints view.

Creating Pblocks Automatically

To use automatic Pblock creation, select **Tools > Floorplanning > Auto-create Pblocks**. Use this command to create top-level Pblocks to view the data flow of the design, and to understand the relative size and relationship between the various logic modules in the design. Figure 10-21, page 322 shows the Auto-create Pblocks dialog box.
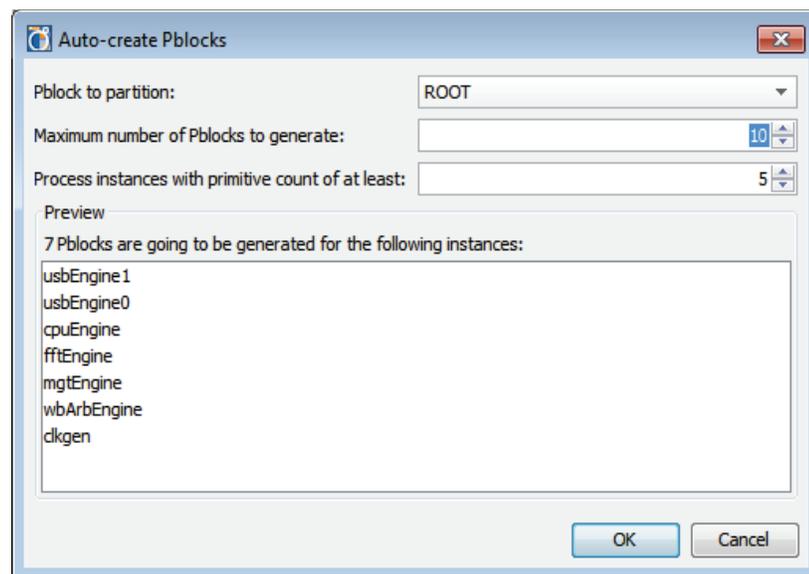
Figure 10-21: **Auto-Create Pblocks**

Specify the following options:

- **Pblock to partition**: Enter the name of an existing Pblock to partition, or specify ROOT to partition the top of the design.

- **Maximum number of Pblocks to generate**: Specify the number of Pblocks to generate based on the hierarchy of the design or selected Pblock. Use this to increase or decrease the number of Pblocks created.

- **Process instances with primitive count of at least**: Limits the examination of instances within the design hierarchy to modules with more than the specified number of primitives. Use this to reduce the number of instances for which to create Pblocks.

- **Preview**: Displays the number and names of Pblocks to create, based upon the defined parameters.

Click **OK** to create the specified Pblocks. The PlanAhead tool creates the Pblocks, and adds them to the list of defined Pblocks in the Physical Constraints view. You can select a Pblock to examine its contents and attributes in the Pblocks Properties view.

The Auto-Create Pblocks command does not add Pblocks to the Device view. It creates the Pblock constraint, and assigns logic elements to the Pblock. After creating the Pblocks, you must place them onto the device.

Running the Automatic Pblock Placer

When you create Pblocks in an open Synthesized Design or an Implemented Design, you can place the Pblocks onto the FPGA logic in the Device view using **Tools > Floorplanning > Place Pblocks**.

The Pblock placer automatically sizes and places Pblocks, based on `SLICE` content only.

Other logic elements are ignored by the **Place Pblocks** command while the PlanAhead tool sizes and places Pblocks. Results are non-deterministic, and may vary from one run to another. The command provides a quick placement of the specified Pblocks to let you view the data flow through the design modules.

Note: The resulting Pblocks from the **Place Pblocks** command might not be suitable for implementation. You might need to resize the Pblocks manually to account for non-SLICE based logic elements.

To run the Pblock placer:

1. Select **Tools > Floorplanning > Place Pblocks**.

   The Place Pblocks dialog box opens as shown in Figure 10-22.

2. Edit the options as follows:

   - **Parent Pblock**: Select the level of hierarchy to place Pblocks. You can place Pblocks at:

     - The top module

     - ROOT

     - Any partitioned Pblock-level of hierarchy

   - **Pblocks to place**: Displays the list of Pblocks that exist under the specified parent Pblock or ROOT.

     - **Place**: Checkbox to specify the Pblocks to be placed. Deselecting the checkbox causes the PlanAhead tool to ignore that Pblock, and preserve existing Pblock locations.

       Pblocks in the list, that have been previously sized and placed, are reset by PlanAhead and resized and replaced unless the Place checkbox is deselected.

     - **Pblock**: Lists the name of the Pblocks within the specified level of the hierarchy.

     - **Utilization**: Lets you set specific SLICE utilization targets for each Pblock. PlanAhead sizes and places the Pblock according to the specified Utilization percentage.
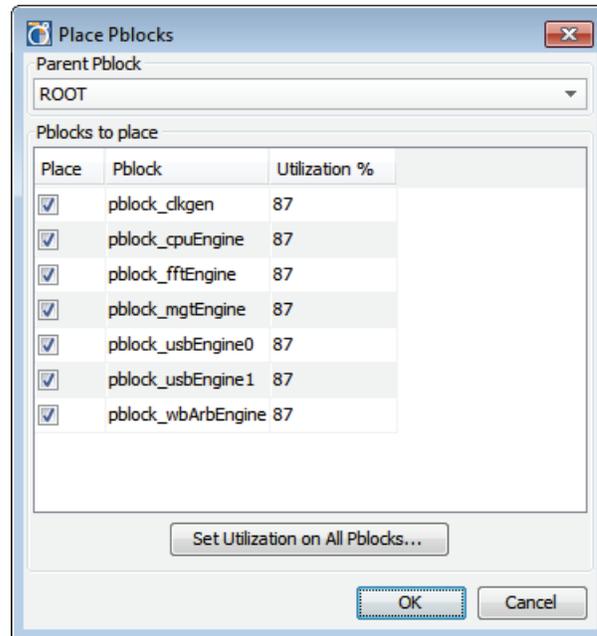
Figure 10-22: **Place Pblocks Dialog Box**

- • **Set Utilization on all Pblocks**: Specifies a new SLICE utilization target for listed Pblocks. Setting this option overrides any individual utilization values you might have modified previously. Use this option to specify the preferred value for all Pblocks, then modify the Utilization of any specific Pblock as needed.

3. Click **OK** to place Pblocks in the design.

A Place Pblocks progress meter displays while the **Place Pblocks** command is running. The Pblocks are sized and placed automatically based on SLICE utilization only.

For instructions on creating top level floorplans using **Create Pblocks** and **Place Pblocks**, see the *Design Analysis and Floorplanning Tutorial: PlanAhead Software (UG676)*, cited in Appendix E, Additional Resources.

Working with Placement LOC and BEL Constraints

You can assign primitive logic elements to specific logic sites using either **Create Site Constraint Mode** or **Create BEL Constraint Mode**. PlanAhead uses LOC constraints to assign logic elements to a specific SLICE and to restrict placement to the available resources within the SLICE; and uses BEL constraints to place a logic instance on a specific site in the SLICE. LOC applies generally to a SLICE, while BEL targets a specific site.

Understanding Fixed and Unfixed Placement Constraints

The PlanAhead tool differentiates between logic placement assigned interactively by the user and automatically by the tool.

- • User-assigned placement is either defined within an imported constraint file or assigned by interactive placement within the PlanAhead tool. User-assigned placement constraints are considered *fixed* and display in one color.

- • Placement constraints defined by PlanAhead or ISE during implementation are considered *unfixed*, and display in a different color.

To fix placement constraints for unfixed instances use the **Fix Instances** or **Fix Ports** command in the Device, Package, or Schematic view.

The PlanAhead tool exports fixed constraints by default to the ISE implementation tools to lock the placement.

The **File > Export > Export Constraints** and **File > Export > Export Pblocks** dialog boxes have a switch that enables exporting both fixed and unfixed placement constraints.

Figure 10-23 shows an example of how Export Pblock and Export IP can be used to export different parts of the design, based on either logical (Export IP) or physical (Export Pblock) hierarchy:
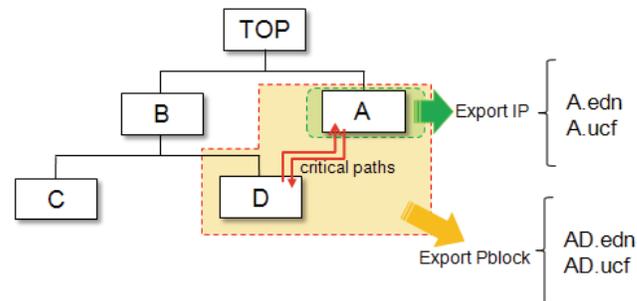
Figure 10-23: **Example of Export Pblock and Export IP**

Understanding Site and BEL Level Constraints

When you place instances into sites, the PlanAhead tool adds LOC constraints, or LOCs and BELs, to the UCF. The locations are assigned as fixed and locked during subsequent implementation runs.

Site constraints result in a LOC constraint being assigned to the instance. The logic element is assigned to the CLB SLICE only, and not to any specific site or resource. The following is an example code snippet:

```
INST "receiver/uartInst/G_98_1" LOC = SLICE_X49Y69;
```

BEL constraints assign a logic element to a specific site within the CLB SLICE, and result in a LOC constraint and a BEL constraint being assigned to the instance in the saved and exported UCF files. The following example shows the BEL and LOC constraints for an instance:

```
INST "channel/receiveRE[8]" BEL = FFX;
INST "channel/receiveRE[8]" LOC = SLICE_X59Y2;
```

Select the **Instance Drag & Drop Mode** in the Device view to determine the manner in which instances placed onto the device will be assigned placement constraints:

- **Create BEL Constraint Mode:** Assign a LOC and BEL constraint to the instance being placed. This fixes the instance to a specific site within the Slice.

- **Create Site Constraint Mode:** Assign a LOC placement constraint to the instance being placed. This fixes the instance to a specific Slice, but does not assign it to a specific site within the Slice.

- **Assign Instance to Pblock Mode:** Assign logic instances to Pblocks. This is the default mode; use this mode whenever possible to ensure proper floorplanning methodology.

The PlanAhead tool does not allow placement to an illegal site, or to a site that is already occupied. The instances being placed snaps between legal placement sites, with a slashed circle indicating illegal sites when they are encountered.

Placing Instances onto SLICEs (LOCs)

You can place a *leaf-level* primitive instance such as a LUT, block RAM, or flip-flop into a specific device resource by dragging it from the netlist tree and dropping it into a specific site. Set the **Instance Drag & Drop Mode** in the Device view to **Create Site Constraint Mode**.

Figure 10-24 shows the placement of a logic element into a specific SLICE. The logic element is dragged into the center of the SLICE, and is only assigned to a specific device resource when you click to assign the element to the SLICE.
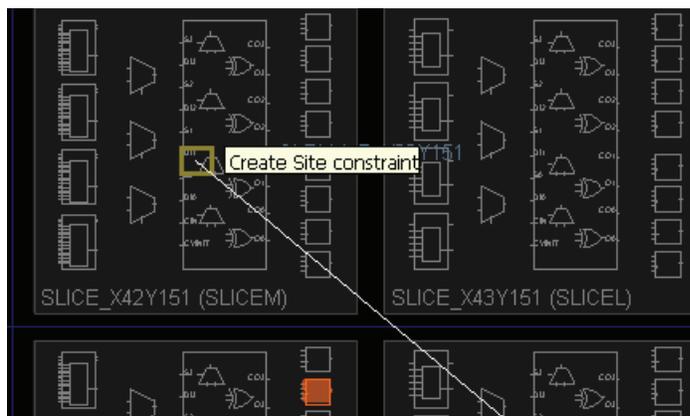
Figure 10-24: **Create Site Constraint**

The dynamic cursor does not allow instances to be placed if the SLICE would be over-packed with logic. Certain logic groups, such as carry-chain logic, move as a single object, which requires open placement sites for all logic on the carry chain.

The location constraints for the placed instance are displayed in the Instance Property view.

Placing Instances onto Sites (BELs)

You can place a *leaf-level* primitive instance into a specific device resource site by dragging it from the netlist tree and dropping it onto a specific device resource. Set the **Instance Drag & Drop Mode** in the Device view to **Create BEL Constraint Mode**.

The BEL constraint assigns a logic element to both a SLICE and a specific device resource within that SLICE. Figure 10-25 shows a logic element placed into a specific device resource. The logic element is dragged onto a specific device resource and directly placed.
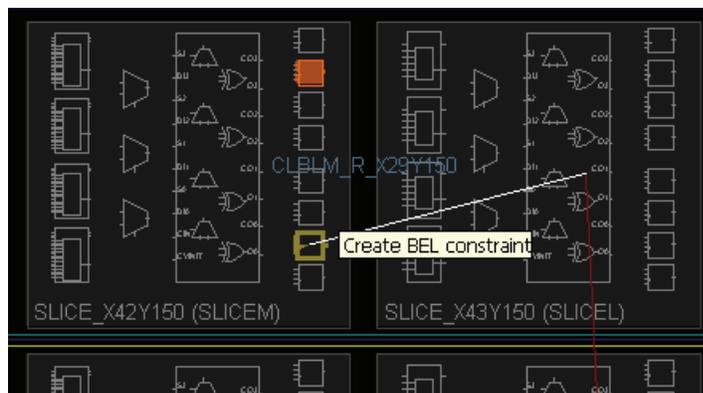
Figure 10-25: **Create BEL Constraint**

The dynamic cursor does not allow instance placement to an illegal or occupied gate site. PlanAhead indicates a legal placement site when the dynamic cursor changes from a slashed circle to an arrow.

Adjusting the Visibility of Placement Constraints

To change how assigned placement constraints display, adjust the zoom level:

- From a zoomed-out view, the LOCs and BELs display as a filled in rectangle inside the assigned site.

- When the zoom level increases, the logic displays as being assigned to specific logic gates within the site.

Figure 10-26, page 327 shows the Device view zoomed-out and zoomed-in and the different details displayed at different zoom factors.
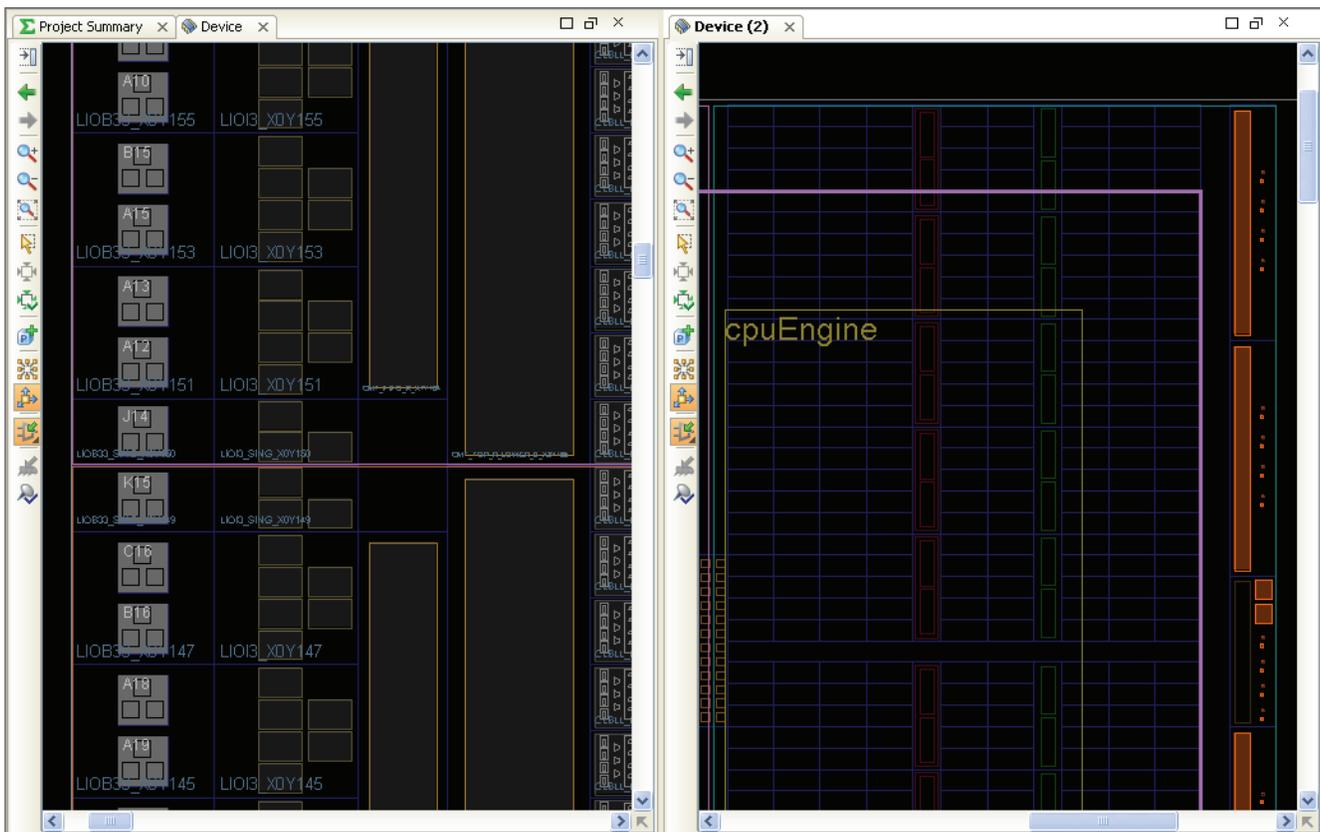
Figure 10-26:   **Dual Zoomed Device Views**

You can use the Device view Layers control to display or hide the location constraints. See Setting Device View Display Options, page 139 for more information.

To adjust other display characteristics for the LOC and BEL constraints:

1. Select **Tools > Options** and select **Colors** from the menu on the left, then select the **Device** tab.

    The **Device** tab displays various configurable elements of the Device view.

2. Change settings in the **Select** column or adjust the colors in the **Frame Color** and **Fill Color** columns.

Fixed and unfixed placement instances have individual color and selection controls.

Moving Placed Instances

To move placed instances:

1. Select a placed instance in the Device view, Netlist view, or Schematic view. You can also select multiple instances and drag and drop them to new locations at the same time.

2. Drag and drop the selected instance or instances to another legal site.

   The primitive instance is assigned to the new site. You can display net flight lines from the instance to connected placed logic or Pblocks.

   Moving a combinational logic object such as a MUX or a carry chain results in the selection of the entire group of instances to move together. The cursor indicates legal placement sites for the entire group and the PlanAhead tool moves all objects to new relative locations.

You can examine location constraint properties in the Instance Property view for selected instances.

Swapping Placement Locations

You can select two placed components and swap their locations. To swap locations:

1. Select two component instances from any of the available views. Hold the **Ctrl** key while clicking to select multiple ports.

2. Right-click and select **Swap Locations.**

Swapping unfixed instances causes them to become fixed.

Clearing Placement Constraints and Unplacing Instances

Placed instances can be unplaced by selecting one or more instances, and using one of the following methods:

- Click the **Unplace** command in the popup menu of the Device view or another view.

  This will simply unplace the selected objects from sites on the FPGA device, clearing LOC and BEL constraints in the process.

- Select **Tools > Floorplanning > Clear Placement**.

  The Clear Placement command opens the Clear Placement Constraints wizard, as shown in Figure 10-27, allowing you to unplace objects, removing LOC and BEL constraints from instances, I/O ports, or both instances and ports.
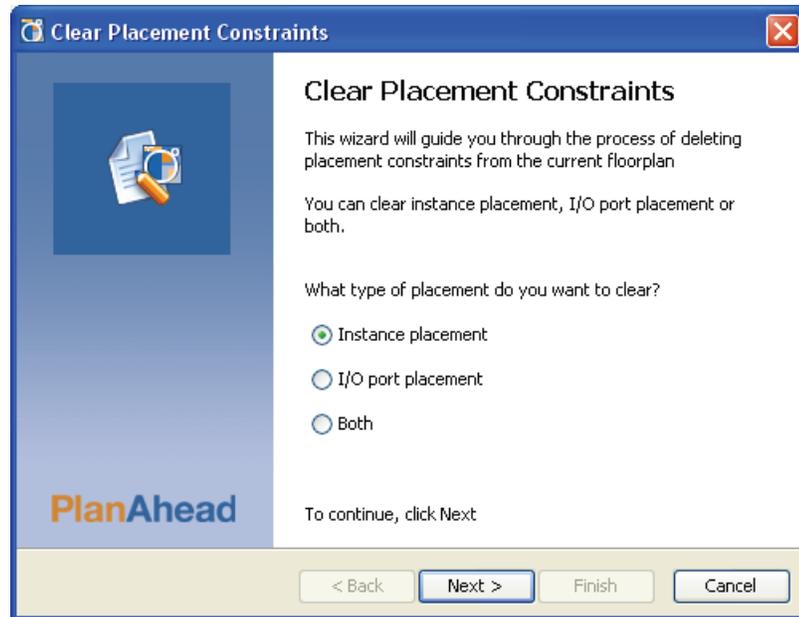
Figure 10-27: **Clear Placement Constraints**

1. From the Clear Placement Constraints wizard, specify the type of placement constraints to remove: Instance placement, I/O port placement, or Both.

2. Click **Next** to continue.

   The Unplace Instances dialog is opened as shown in Figure 10-28.
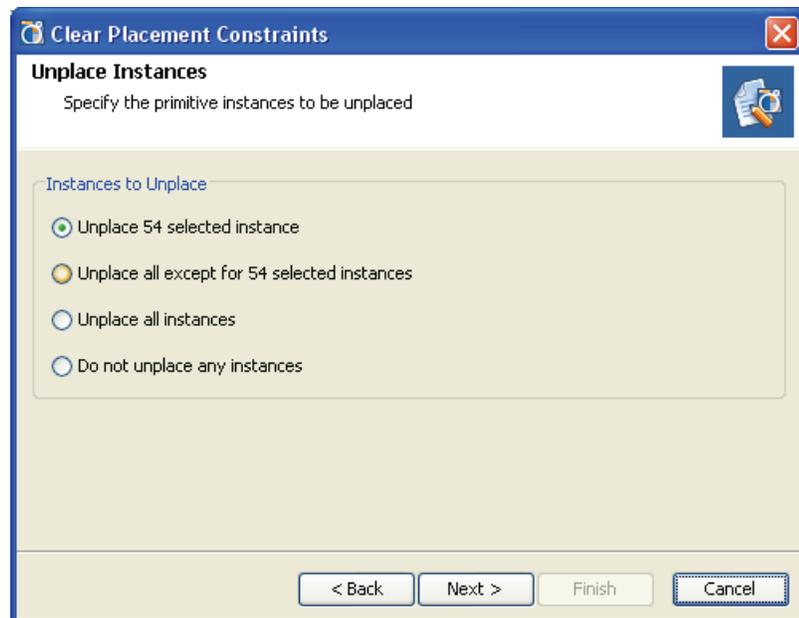
Figure 10-28: **Unplace Instances**

3. If you have selected instances prior to using the Clear Placement command, you are prompted to specify which instances to unplace:

   • Unplace the selected instances.

   • Unplace all but the selected instances.

- Unplace all instances, regardless of the selected objects.

- Do not unplace any instances, if you no longer want to unplace instances.

   Note: If you have not previously selected instances, step 2 is skipped, and you are presented with a list of instance types in the design to unplace as described in step 4, and shown in Figure 10-29, page 330.

4. Click **Next** to continue.

   You are presented with a list of instance types to unplace as shown in Figure 10-29, page 330. This lets you selected from the various types of instances within the group of instances specified in Step 2.
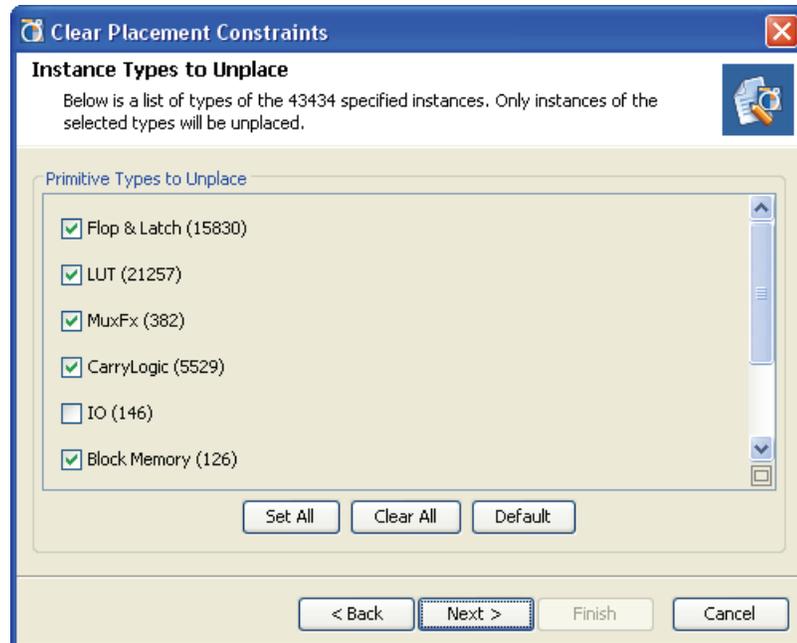
Figure 10-29: **Unplace Instance Types**

5. Choose the types of instances to unplace, and click **Next**.

   If some of the instances or ports you have selected to unplace are fixed, the Fixed Placement dialog box opens as shown in Figure 10-30.
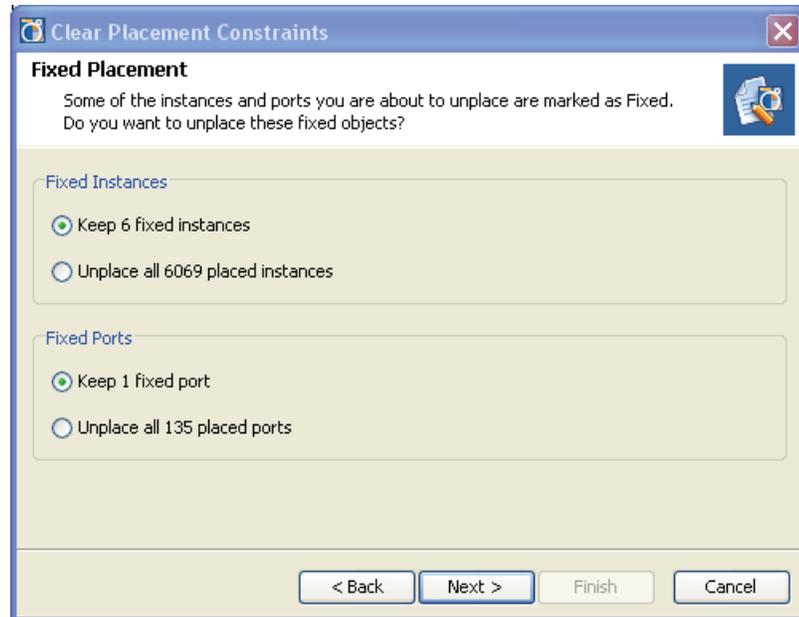
Figure 10-30: **Unplace Fixed Instances and Ports**

6. Specify whether you want to preserve the current placement of fixed instances, or you want to unplace all instances. See Understanding Fixed and Unfixed Placement Constraints, page 324 for more information.

    If you chose to unplace ports, you might also be presented the choice of preserving fixed ports or unplacing all ports as shown.

7. Click **Next** to continue.

    The Clear Placement Summary opens to show a summary of the instances to be unplaced.

8. Verify the contents of the Clear Placement Summary, and click **Finish**.

    •   The specified I/O ports and instances are unplaced from the device.

    •   Previously assigned ports are not cleared until a new UCF is read into the PlanAhead tool.

    •   New port assignments write over previous assignments.

Note: A best practice is to first clear all port assignments prior to importing new port assignment constraints.

Setting Placement Prohibit Constraints

PROHIBIT constraints allow you to mark Sites on the device as prohibited for placement purposes. That is you cannot place, and the software cannot place, instances or ports on the prohibited Sites. The PROHIBIT constraint can be used to ensure device compatibility between multiple parts. By prohibiting resources not common to compatible parts you can ensure the design easily maps from one device to another.

You can create a PROHIBIT constraint for any Site on the device. To do so:

1. Select the sites in the Device view.

2. Use the **Select Area** command from the side toolbar menu to select multiple sites. See Using the Select Area Command in Chapter 4 for more information.

3. Select the **Set Prohibit** command from the right-click pop-up menu. The prohibited sites display a red warning, as shown in Figure 10-31.
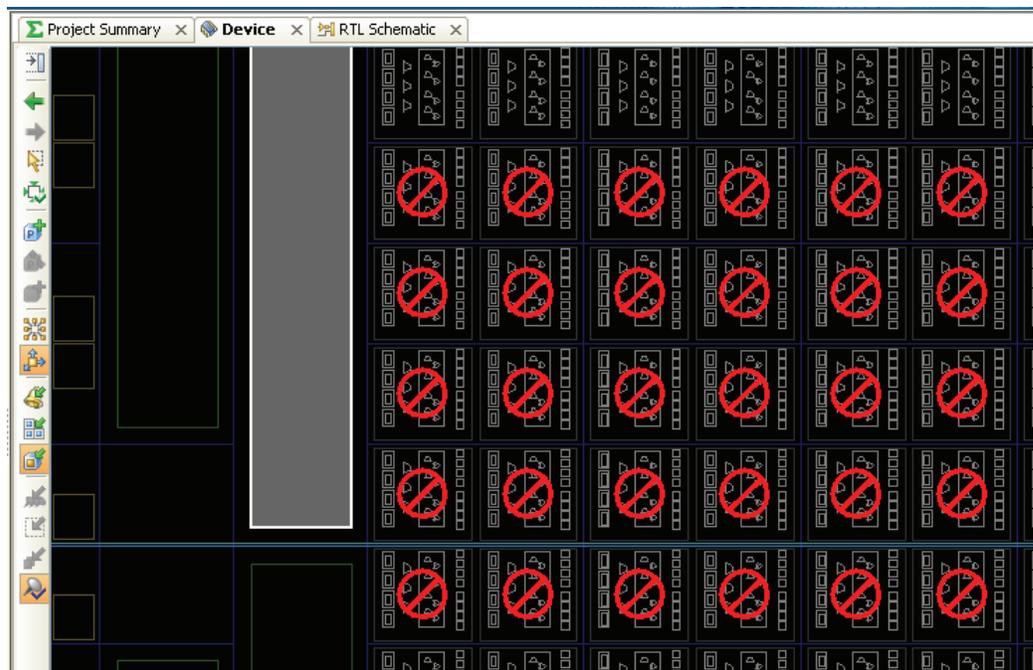
Figure 10-31: **Prohibited Sites in the Device View**

Exporting Data from PlanAhead

The PlanAhead tool lets you selectively export files required by external software, for synthesis, simulation, or implementation.

Exporting Constraints

Exporting constraints to ISE consists of exporting a UCF constraints file for the entire design or for individual Pblocks.

To export the constraints:

1. In the Sources view, select the constraint set to export, and click **Make Active** to set it as the active constraint set.

2. Select **File > Export > Export Constraints**.

    The Export Constraints dialog box opens.

3. Edit the options in the **Export Constraints** dialog box:

    •   **File name**: Enter the file name and location to create the UCF.

    •   **Export fixed location constraints only**: Select this option to export only user-assigned "fixed" placement `LOC` constraints or uncheck it to export the fixed and unfixed placement constraints imported from ISE, then click **OK** to export the constraints.

The PlanAhead tool creates the designated top-level UCF format constraint file in the export directory. You can use this file as input for custom ISE implementation scripts.

For more information about the exported files, see Appendix A, Outputs for Reports.

Exporting Netlist

Exporting the PlanAhead Netlist to ISE consists of exporting a single EDIF format netlist file for the entire design or for individual Pblocks. This requires an open Synthesized Design.

To export the netlist, from an open Synthesized Design:

1. Select **File > Export > Export Netlist**.

   The **Export Netlist** dialog box opens.

2. Edit the File name and location to create the EDIF format netlist file in the **Export Netlist** dialog box, then click **OK** to export the netlist.

For more information about the exported files, see Appendix A, Outputs for Reports.

Exporting Pblocks for ISE Implementation

The PlanAhead tool can export Pblock-level files for Implementation. These Pblocks consist of logic from anywhere in the logic hierarchy.

Exporting a Pblock creates an EDIF netlist and a UCF physical constraints file for each Pblock selected for export.

To export Pblocks to EDIF and UCF:

1. Select one or more Pblocks.

2. Select **File > Export > Export Pblocks**.

   The **Export Pblocks** wizard opens as shown in Figure 10-32, page 333.
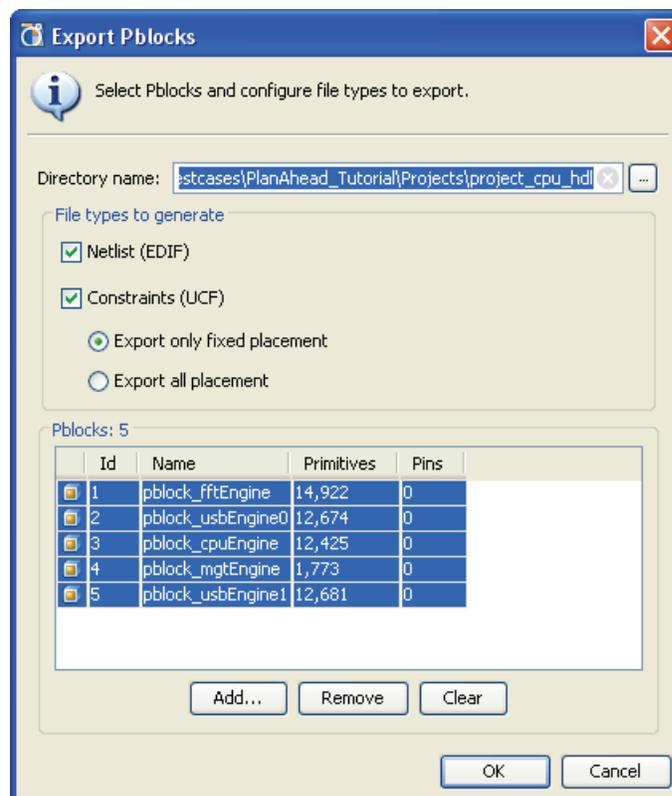
Figure 10-32:   **Export Pblocks Wizard**

3. Edit the options:

- **Directory name**: Enter the directory name or use the file browser to select a directory to export the files for the specified Pblocks. The PlanAhead tool creates a subdirectory called *pblockname*_CV for each exported Pblock.

   To keep track of EDIF and UCFs associated with each Pblock, it is a good practice to specify a unique directory name for each ISE run. The exported Pblock directory is used to seed the **Import Placement** and **Import Timing** command file browsers.

- **File types to generate:**
   - **Netlist (EDIF)**: Export the netlist.
   - **Constraints (UCF)**: Export all, or only fixed, placement constraints.

- **Pblocks**: Lists the Pblocks selected for export.

4. Click **Add** and **Clear** to select and remove Pblocks from the export list, respectively, and click **Next** or **Finish** to continue.

   When you chose **Next**, the **Export Pblocks Summary** dialog box displays Pblock export selections.

5. Click **Finish** to perform the export.

The PlanAhead tool creates separate EDIF and UCF files for each of the exported Pblocks named *pblockname*_CV.edn and *pblockname*_CV.ucf.

A /*pblockname*_CV directory is also created for each exported Pblock containing the Pblock-specific files.

Analyzing Implementation Results

The PlanAhead™ tool has many features to help analyze the placement and routing results of an implemented design. The design can be implemented in the PlanAhead tool, or you can import implementation results from the Xilinx® ISE® Design Suite of tools.

The PlanAhead tool can perform the following on placed and routed designs:

- Design Rules Check (DRC)
- SSN analysis
- SSO analysis
- Timing analysis
- Power analysis

In addition, the PlanAhead tool allows you to:

- Open the FPGA Editor to view and modify routing resources in the implemented design.
- Launch the Xilinx ISim tool for timing simulation of the implemented design.

Opening the Implemented Design

To open an Implemented Design, select one of the following methods:

- **Flow > Open Implemented Design** in the main menu.
- **Open Implemented Design** from the Implementation menu in the Flow Navigator.
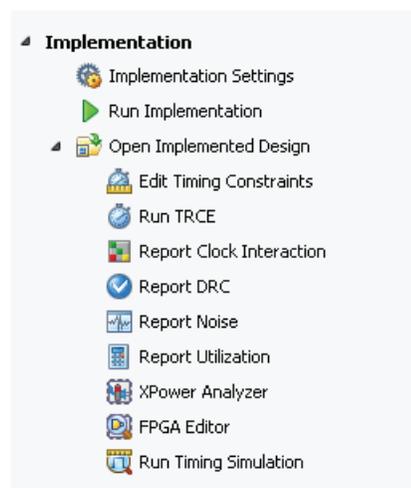
Figure 11-1: **Flow Navigator - Implemented Design**

The Implemented Design menu, as shown in Figure 11-1, page 335, lists some of the available analysis tools when an Implemented Design is opened.

Figure 11-2 shows the default view layout for an Implemented Design.
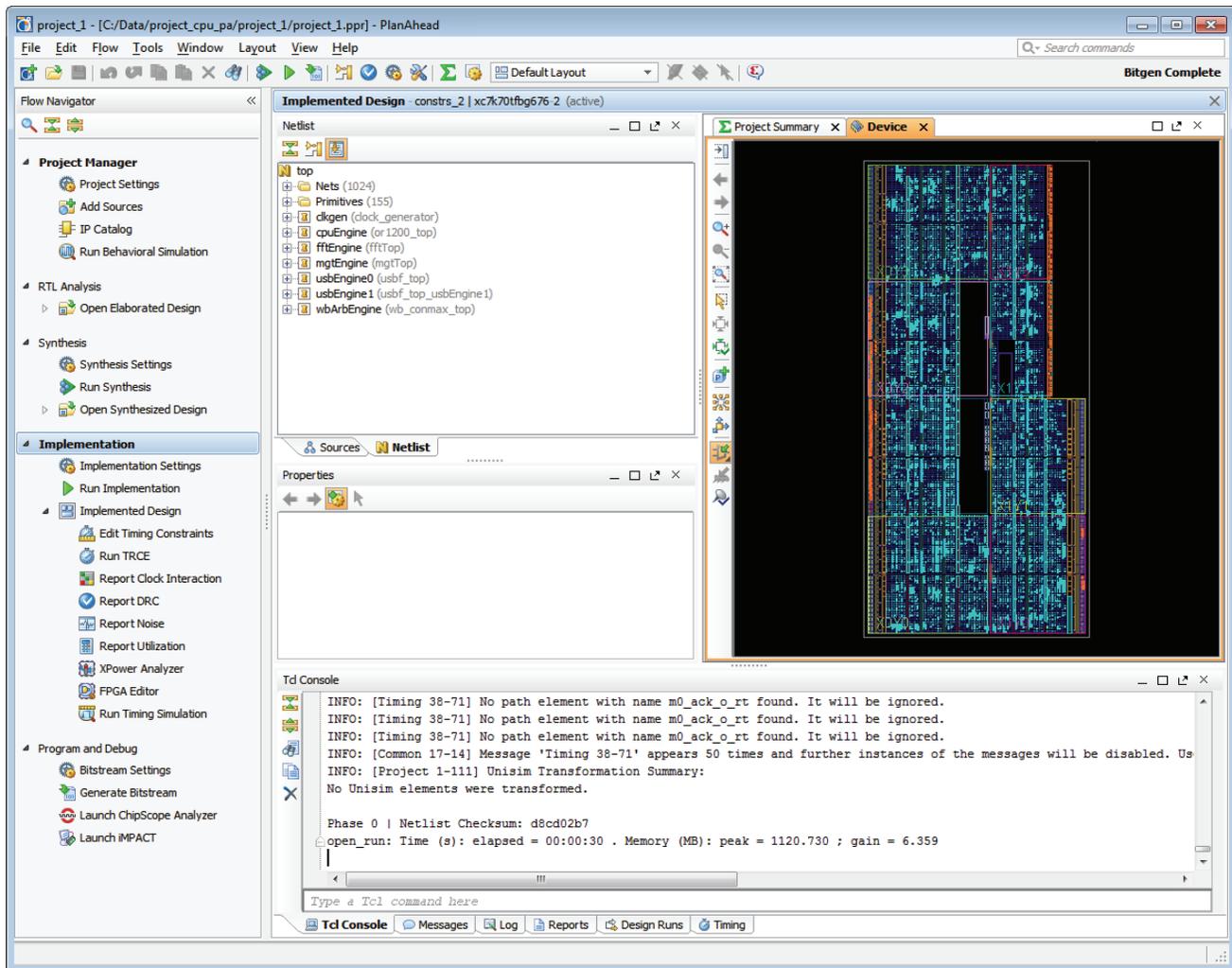
Figure 11-2: **Implemented Design Environment**

The placed design shows in the Device view, with fixed and unfixed instances displaying in different colors. Fixed instances are logic instances that were interactively placed by a user. Unfixed instances are logic instances that were automatically placed by the software. You can fix the placement of any unfixed instances to prevent reassignment during subsequent implementation runs by using the **Fix Instances** or **Fix Ports** command in the Device, Package, or Schematic view. Refer to Working with Placement LOC and BEL Constraints, page 324 for more information.

Note: ISE can optimize and change logic to improve placement and routing results. When this happens, logic in the original netlist is removed or replaced. This results in a mismatch between the pre-Implementation netlist opened in the PlanAhead tool and the Implementation results. The Tcl console reports these discrepancies as the Implemented Design is opened. It does not pose any problems other than the logic in the netlist does not match the viewed results exactly.

Opening Multiple Implemented Designs

You can open any implementation run by one of the following options:

- Select an implementation run in the Design Runs view, and use **Open Implemented Design** from the popup menu, or double-click the implementation run.

- Set the run as the active run, then click **Open Implemented Design** from the Flow Navigator. Refer to Setting the Active Run, page 290.

When multiple Implemented Designs are opened, tabs display in the upper-left corner of the viewing environment to enable switching between the open designs, as shown in Figure 11-3.

Figure 11-3: **Multiple Implemented Designs Open Tabs**

You can switch between open designs using the tabs in the display banner, or by using the **Implemented Design** popup menu in Flow Navigator.

You can open and close available Implemented designs using the pulldown menu in the Flow Navigator. Each Implementation run shows under the Open Implemented Design menu enabling any completed run to be opened, as shown in Figure 11-4.
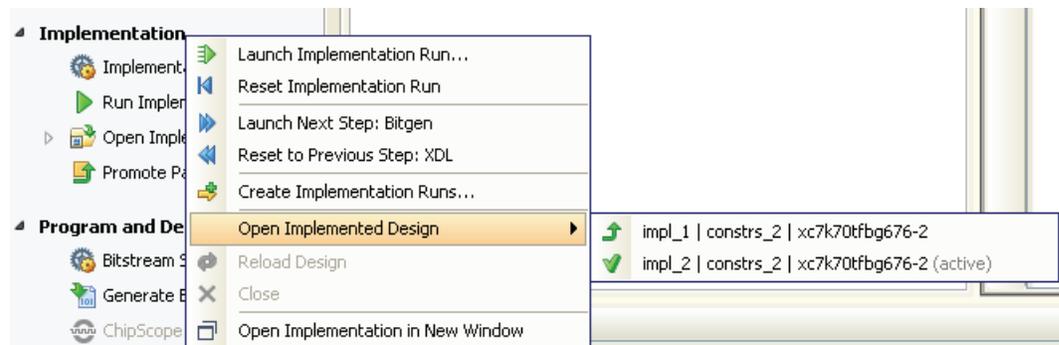
Figure 11-4: **Opening Implemented Designs**

When multiple implemented designs are open, the Open Implemented Design button displays the number of open designs.

Building an Implemented Design with Imported Placement

You can import existing ISE results using the New Project wizard, or by import timing results. The following subsections describe the options for importing ISE results.

Using the New Project Wizard to Import Placement

You can create a new project and populate the project with a netlist, and placement and timing results from an ISE software implementation run. For more information about creating a new project that imports ISE implementation results, see Creating a Project with ISE Placement and Timing Results, page 49.

Importing Placement Results into an Existing Project

You can also import placement results generated outside the PlanAhead tool into an existing project. Placement constraints are assigned for all placed logic objects.

You can select an implemented Native Circuit Description (NCD) file and the PlanAhead tool takes the data in the file and converts it to Xilinx Definition List (XDL) format to import the placement information into the project. NCD is the output file from PAR, the ISE software place and route application. The PlanAhead tool uses the data in the NCD file, but does not add the file into the project.

To import placement results:

1. Select **File > Import > Import Placement**. The **Import Placement** dialog box opens.

    You must have an open RTL, Netlist, or Implemented Design for the Import Placement command to be available. See Working with Designs, page 24, for more information.

2. Select an NCD file in the **Import Placement** dialog box.

    The PlanAhead tool automatically runs the XDL utility to convert the NCD file to XDL format, and imports the results.

    Note: You can also import an XDL file directly if one exists.

3. Click **OK** to import the placement results.

The PlanAhead tool imports the placement file and build an Implemented Design with the file. With the Implemented Design open, you can analyze the FPGA placement results performing timing analysis, power analysis, and timing simulation as described in the following sections.

Analyzing Timing Results

Timing analysis of the implemented design, being placed and routed, includes the actual routed path delays between logic elements or device resources. To complete timing analysis of the implemented design you can:

- Run the **Tools > Timing > Run TRCE** command to access the Xilinx Timing Reporter And Circuit Evaluator (TRCE) tool for sign-off timing analysis. See Running TRCE on an Implemented Design for more information.

- Click the **Run TRCE** command on the Flow Navigator.

- Run the **Tools > Timing > Run Timing** command to access the internal timing engine of the PlanAhead tool for timing estimation. See Running Timing Analysis in Chapter 7 for more information.

    Note: The static timing analysis engine in PlanAhead should not be used for timing sign-off.

Running TRCE on an Implemented Design

The TRCE tool provides static timing analysis of an FPGA design based on input timing constraints. The TRCE software verifies that the design meets timing constraints, and generates a report file that lists compliance of the design against the input constraints.

As a standalone ISE application, TRCE can be run on unplaced designs or placed designs, partially routed design, and completely placed and routed designs. In the PlanAhead tool, TRCE can be run on an Implemented Design. When an Implemented Design is open in PlanAhead, you can launch the TRCE software by:

- Run the **Tools > Timing > Run TRCE** command.

- Click the **Run TRCE** command on the Flow Navigator.

The PlanAhead tool opens the Run TRCE dialog box as shown in Figure 11-5, page 339.
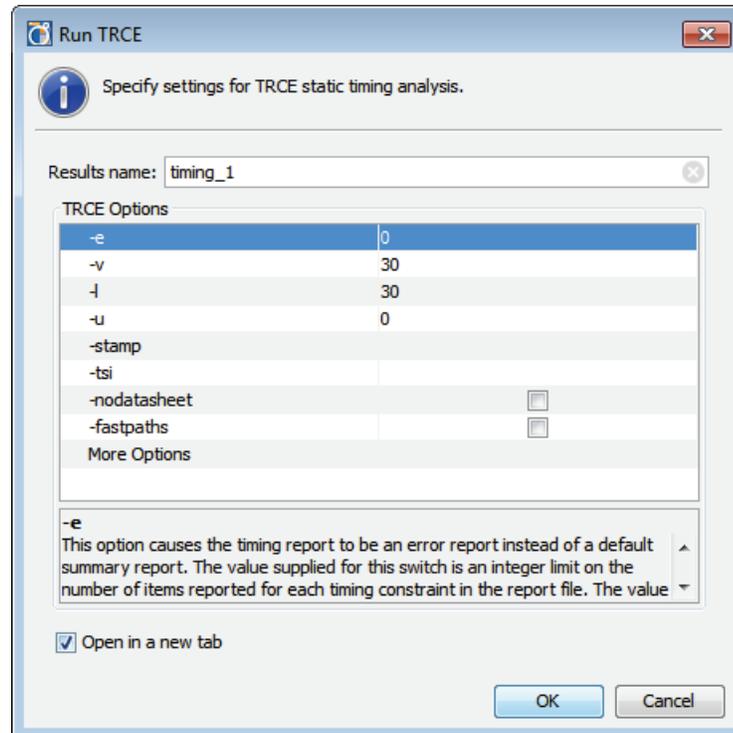
Figure 11-5: **Run TRCE Dialog Box**

- **Results Name**: Specifies the name of the results for the output timing results. As a default, the results are written to a file in the implementation directory with the name of the top module and the specified Results Name combined. The output file name can also be changed by specifying the **-o** option in the **More Options** field.

  The results are posted to the Timing Results view under the specified results name. See .

TRCE Options:

- **-e** *<limit>*: Causes the timing report to be an error report instead of the default summary report. *<limit>* is an integer, from 0 to 32,000 inclusive, that restricts the number of items reported for the different violations of each timing constraint.

- **-v** *<limit>*: Generates a verbose report. The optional limit is used to limit the number of items reported for each timing constraint in the report file. The value of limit must be an integer from 1 to 32,000 inclusive. If a limit is not specified, the default value is 3.

- **-l** *<limit>*: Limits the number of items reported for each timing constraint in the report file. *<limit>* must be an integer from zero to 2 billion inclusive. The default value is 3.

- **-u** *<limit>*: Reports delays for unconstrained paths. *<limit>* is an integer from one to 2 billion inclusive.

  The -u option adds a constraint to all unconstrained paths in the design to include them in the timing analysis. This constraint performs a default path enumeration on any paths for which no other constraints apply. The default path enumeration includes circuit paths to data and clock pins on sequential components and data pins on primary outputs.

- **-stamp** *<filename>*: Generates a pair of STAMP timing model files (`<filename>.mod` and `<filename>.data`) that characterize the timing of a design.

- **-tsi *<filename*.tsi>**: Generates a Timing Specification Interaction (TSI) report (also known as the Constraint Interaction report). The TSI report can be used to understand which timing constraints take precedence when multiple timing constraints cover the same paths.

- **-nodatasheet**: Excludes the datasheet section of the standard timing report.

- **-fastpaths**: Reports the fastest paths of a design

- **More Options**: Applies the specified TRCE options. For more information about specific TRCE commands see the *Command Line Tools User Guide, (UG628)* as cited in Appendix E, Additional Resources.

  Two examples of additional commands are:

  - **-o *<file_name>***: Specifies the name of the output timing report. The `.twr` extension is optional.

  - **-s *<speed_grade>***: Overrides the device speed contained in the design and instead performs an analysis for the device speed specified. The option lets you see if faster or slower speed grades might meet your timing requirements. *<speed_grade>* can be specified as values of 1, 2, or 3, entered with or without the leading dash.

- **Open in a New Tab**: Specifies that the timing report as a result of the run should be opened in a new tab in the Timing Results view. If this option is not specified the last timing results are closed when the new report is opened.

The PlanAhead tool displays the results from TRCE timing analysis by extracting information from the TWX timing report files. After the TWX files are imported, the timing results display in the Timing Results view. If no timing constraints are returned, then no timing results display.

Importing ISE TRCE Timing Results into an Existing Project

You can import timing results from the TWR or TWX format timing report files generated by the TRCE command run outside of the PlanAhead tool. The TRCE software outputs the following timing reports based on options specified on the command line:

- **TWR**: default timing report.

  The `-e` (error report) and `-v` (verbose report) options can be used to specify the type of timing report you want to produce: a summary report (the default), an error-only report, or a verbose report.

  Note: PlanAhead cannot import the TWR file format.

- **TWX**: an XML timing report output by using the -xml option.

  This report is viewable with the Timing Analyzer GUI tool. The `-e` (error report) and `-v` (verbose report) options apply to the TWX file as well as the TWR file. See the `-xml` (XML Output File Name) section for details.

To import TRCE timing results into the PlanAhead tool:

1. Select **File > Import > Import Timing** from the main menu.

   You must have an open RTL, Netlist, or Implemented Design for the Import Timing command to be available. See Working with Designs, page 24, for more information.

2. Edit the dialog box fields:

   - **File Name**: Select a TWX file to import.

   - **Results Name**: Define a name for the results reported in the Timing Results view.

3. Click **OK** to import the timing results.

   The timing results display within the PlanAhead environment.

Using the Timing Results View

PlanAhead imports the timing results from the ISE TRCE program and displays the results in the Timing Results view.

The Timing Results view displays the TRCE timing path information sorted by clock constraint. Timing paths can be expanded and collapsed using the tree widgets in the view. Slack numbers for failing paths are displayed in red. Figure 11-6 shows the TRCE results.

Figure 11-6:   **ISE TRCE Timing Results**

Timing results from the TRCE software are different from the results of the **Report Timing** command available for use with Synthesized Designs as described in Analyzing Timing Results, page 230

The TRCE results report the exact timing information of the placed and routed design. The PlanAhead tool only estimates the routing delays in its timing analysis.

When you read in the TRCE timing report, the nets and primitives from the timing report are correlated to the netlist database in PlanAhead for cross-probing and analysis. If primitives (BELs) from the timing report cannot be directly correlated to primitives in the netlist, they are not displayed in the timing report and the delays are associated with primitives that PlanAhead does find. The timing delays from TRCE are preserved in the accumulated delay of the timing report in PlanAhead.

Note: ISE can change logic to optimize and improve placement and routing results. When this happens, logic in the original netlist might be removed or replaced. This causes a mismatch between the pre-Implementation netlist opened in the PlanAhead tool and the timing results returned by TRCE. These discrepancies do not pose any problems in the FPGA implementation, but are reported by the Tcl console when the Implemented Design is opened.

You can examine, sort, and select specific paths and instances in the Timing Results view. Selected paths are displayed in the Device view and view properties of the path displayed in the Path Properties view.

You can examine, sort, and select specific paths and instances in the Timing Results view. The following information is displayed:

- **Settings:** Displays a summary of the TRCE command line, and the options used when the timing analysis was performed.

- **Statistics:** Displays an overview of the timing results.

- **Timing Checks:** Grouped into the different types of checks performed, this provides a table view for the timing results, initially sorting timing paths by the calculated slack.

  - **Name**: Shows a sequential number with which to sort back to the original order.

  - **Slack**: Displays the total positive or negative slack on the path.

  - **From**: Displays the path source pin.

  - **To**: Displays the paths destination pin.

  - **Total Delay**: Lists the total estimated delay on the path.

  - **Logic Delay**: Lists the delay attributed to logic elements on the path.

  - **Net Delay**: Lists the delay attributed to the interconnect of the path. Not displayed by default.

  - **Logic%**: Displays the percentage of the delay attributed to logic elements. Not displayed by default.

  - **Net%**: Displays the percentage of the delay attributed to interconnect.

  - **Stages**: Displays the total number of instances on the path including the source and destination which contribute to the overall delay.

  - The stages reported might be different than levels of logic reported in ISE.

  - **Source Clock**: Displays the source clock name.

  - **Destination Clock**: Displays the destination clock name.

Sorting the Timing Report

You can sort the Timing Results by clicking any of the column headers. For example, click on the **Stages** column header to sort the list by stages of logic. Click the column header a second time to reverse the sort order.

You can sort by a second column by pressing the **Ctrl** key and clicking a second column header. You can sort by as many columns as necessary to refine the sort results. Press **Ctrl** and click the column header again to remove a sort from a column.

Refer to the Using Tree Table Style Views, page 113 for information regarding working with tree table style views.

Figure 11-7 shows a timing result sorted by stages of logic.
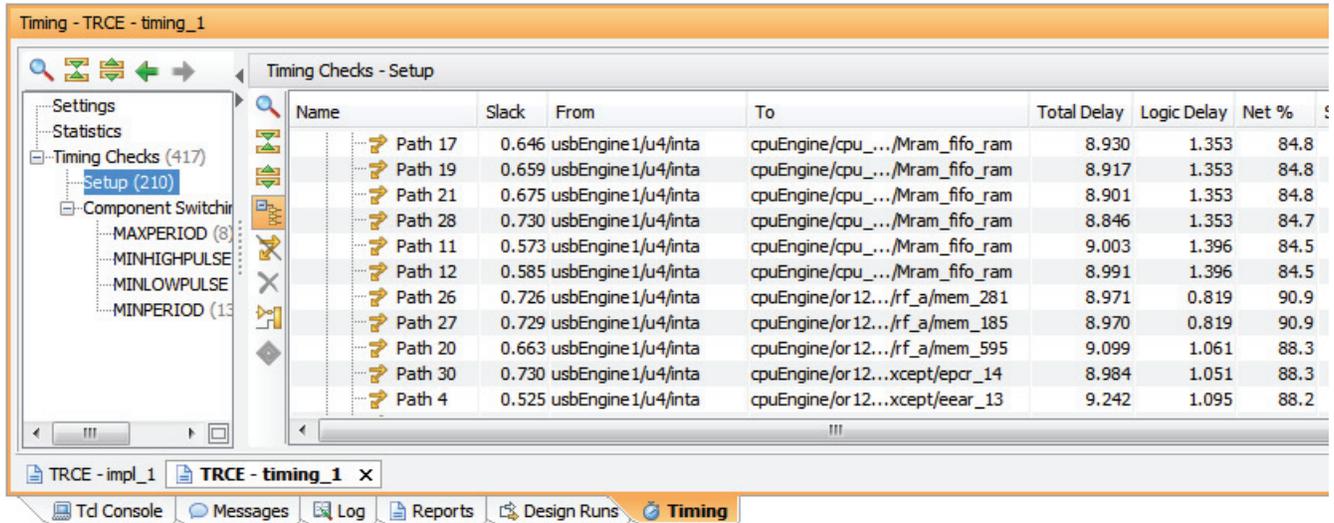
Figure 11-7: **Sorted Timing Results and Group by Constraint**

Flattening the List of Paths

By default, the paths are categorized by constraint. You can flatten the list and view all paths by clicking **Group by Constraint** button in the Timing Results view toolbar, as shown in Figure 11-7, page 343. The Group by Constraint toolbar button toggles between a list of paths grouped by constraint, and a flattened list of paths.

Using the Path Properties View

To display information about the logic and delay on a selected path in the Path Properties view, as shown in Figure 11-8, select a path in the Timing Results view.

Figure 11-8: **Timing Path Property View**

- By default, selecting a path also selects all instances contained within the path, but the Path Properties view only displays information about the first path selected.

- Selecting any of the objects in the report cross-selects the object in other open views such as the Netlist and Device views.

- Select multiple paths using the **Shift** or **Ctrl** keys.

The PlanAhead tool provides links under the **Delay Type** column, which can be clicked to open the FPGA device data sheet, and automatically search the PDF file for the selected logic site object.

You can also view the path data in the larger workspace area by selecting a path in the Timing Results view, and using the **View Path Report** command in the right mouse popup menu, or by double-clicking on the selected path in the Timing Results view. For more information about analyzing timing results using the Timing Results and Path Properties views, see Chapter 7, Synthesized Design Constraints and Analysis.

Viewing Timing Paths in the Device View

You can view timing paths in the Device view when you select a path row or rows in the Timing Results view. The path is highlighted in the Device view. You can select multiple paths and all instances found in the path are selected and highlighted.

Figure 11-9, shows a highlighted timing path in the device view.
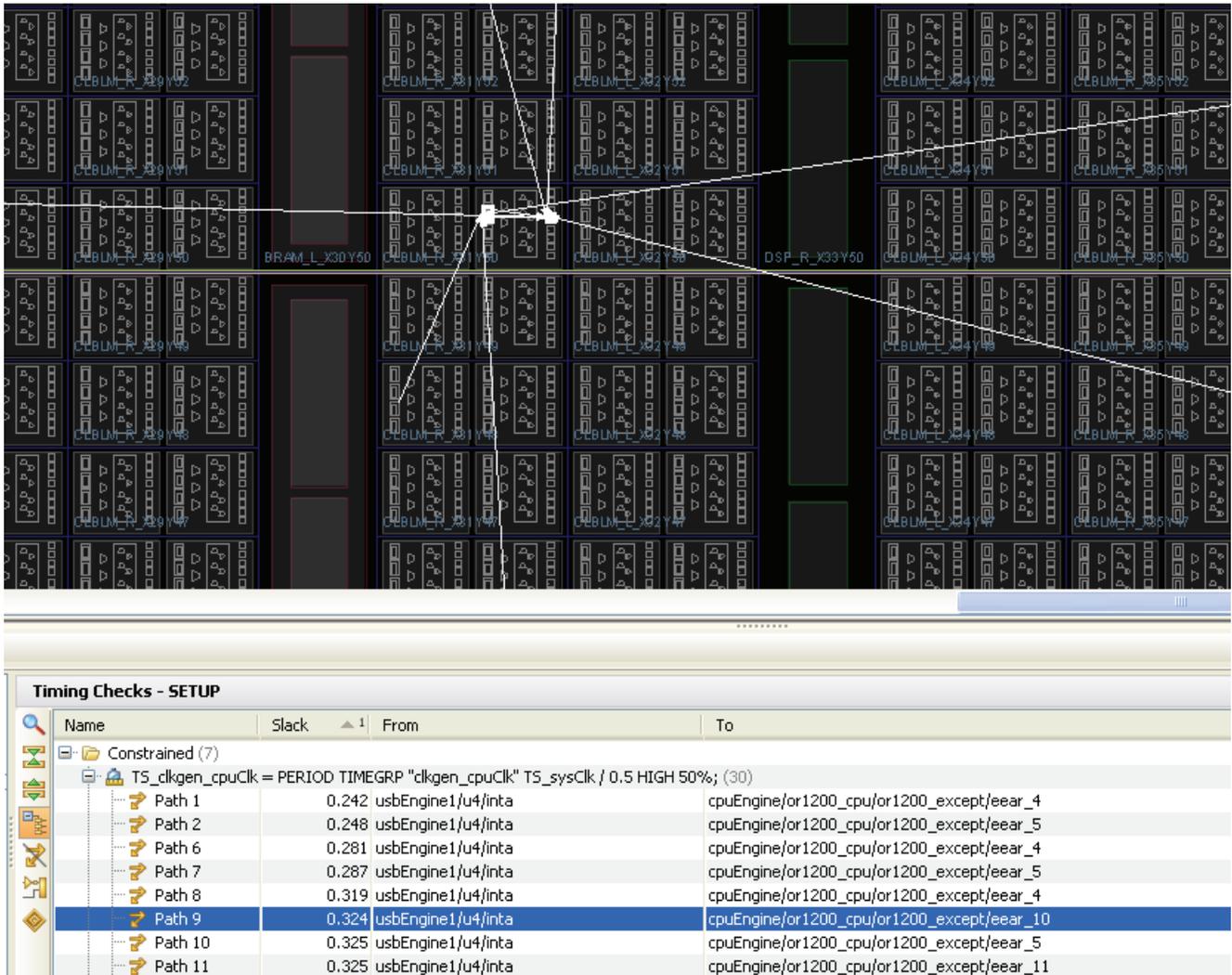
Figure 11-9: **Highlighted Timing Paths in the Device View**

Viewing Timing Paths in Schematic View

When you select Schematic from the Timing Results toolbar or popup menu, the PlanAhead tool generates a Schematic view that displays the instances found in the selected paths. The Schematic view displays the instances clearly along with the hierarchical modules as shown in Figure 11-10.
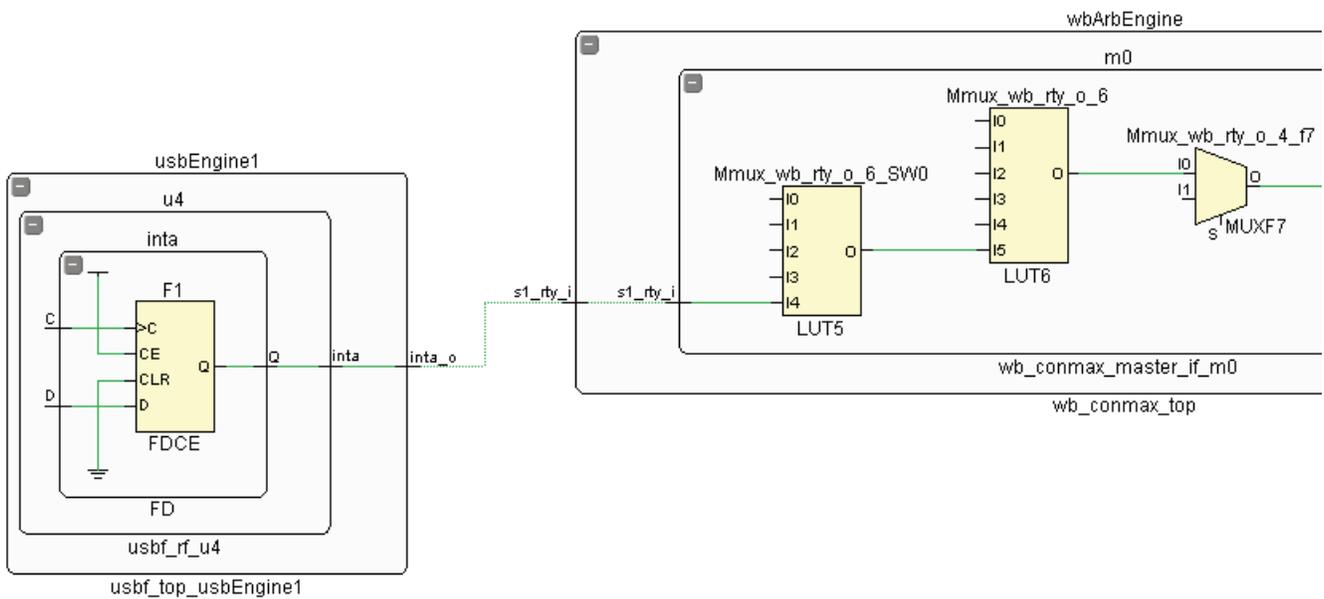
Figure 11-10: **Timing Paths Displayed in Schematic View**

When the PlanAhead tool generates the Schematic view for a timing path, it displays all of the objects. When you select a Schematic view for individual logic instances to be generated, only the selected instances display.

You can display the instances from a group of paths in this manner making it easy to identify what modules should be grouped together for floorplanning. Pblock creation popup commands in the Schematic view let you make direct assignment to Pblocks in the device view. For more information about Schematic expansion and traversal commands, see Using the Schematic View, page 146.

Exploring Logic Connectivity

The following sections describe logic connectivity options in the PlanAhead tool.

Using the Show Connectivity Command

The Show Connectivity command highlights all of the nets connected to the selected elements. To use this command, perform one of the following:

1. Select one or more primitives from the Netlist view or the Schematic view, a placed logic element in the Device view, or a Pblock instance from the Device view or the Physical Properties view. You can also select these objects in some combination.

2. From the popup menu, select **Show Connectivity**.

For example, if you select an instance or Pblock in the Schematic view, all of the nets connecting to that element are highlighted in the Device view, as shown in Figure 11-11, page 347.
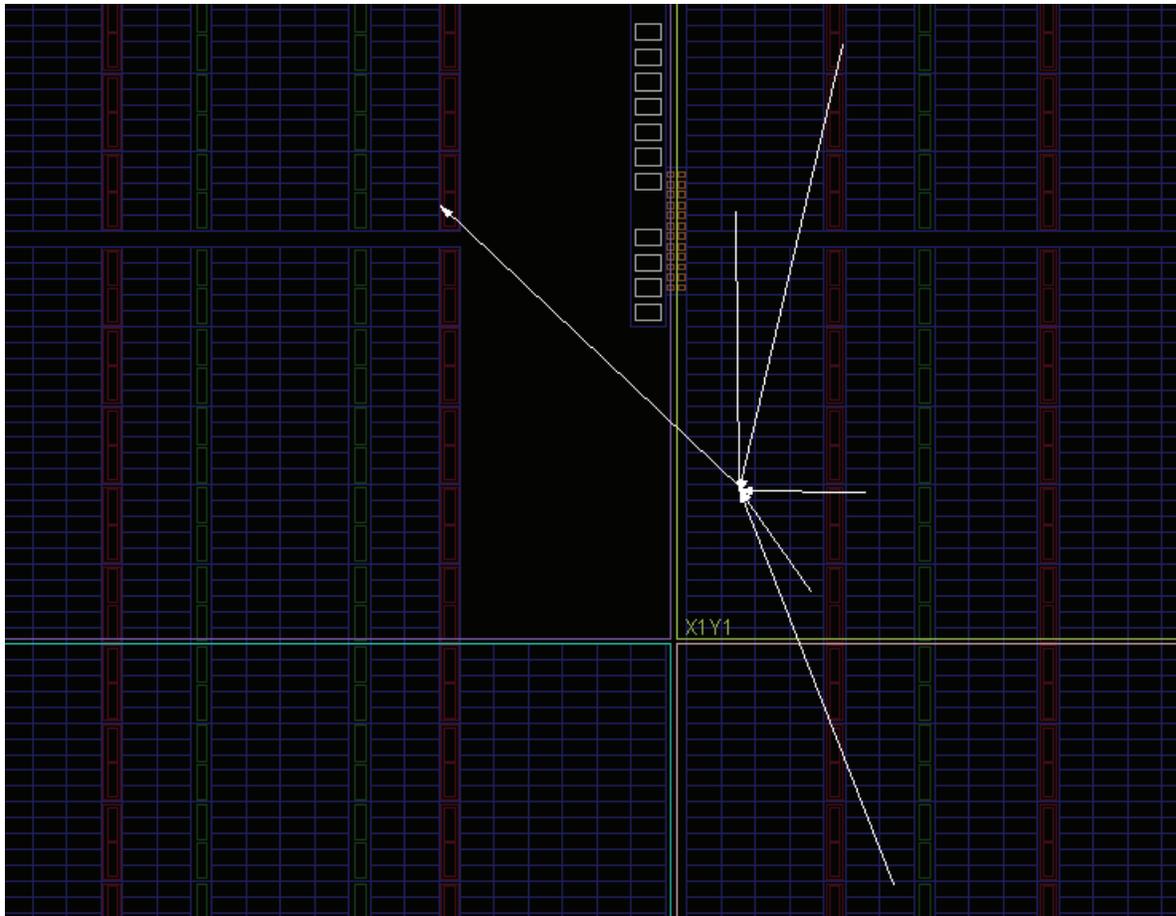
Figure 11-11: **Net Connectivity in the Device View**

Viewing Logic Connectivity using Show Connectivity Mode

You can show connections on selected objects by toggling the **Show Instance Connections** toolbar button on and off.

The **Show Instance Connections** mode remains active, which lets you select additional logic objects for viewing connectivity. Toggle the toolbar button to turn off that mode.

Expanding and Selecting a Logic Fanout

You can run **Show Connectivity** sequentially to continue to select and expand a logic fanout.

1. Select a Netlist, a Pblock instance, or a combination thereof.

2. From the popup menu, select **Show Connectivity**. The command highlights all nets connected to the selected element.

3. From the popup menu, select **Show Connectivity** a *second* time. The command selects the set of connected instances to those nets. You can use the **Ctrl+T** shortcut key also.

4. From the popup menu, select **Show Connectivity** a *third* time. The command highlights the next level of nets connected to the selected instances, and so on.

This is an easy way to select a logic fanout starting at a particular instance or I/O port.

Expanding Logic in the Schematic View

You can trace logic throughout the design hierarchy using the Schematic view. Anything selected in the Schematic view is highlighted in the Device view also.

You can expand signals interactively by double-clicking on the pins of the instance to be traced. Figure 11-12 shows an example of expanded logic in the Schematic view.
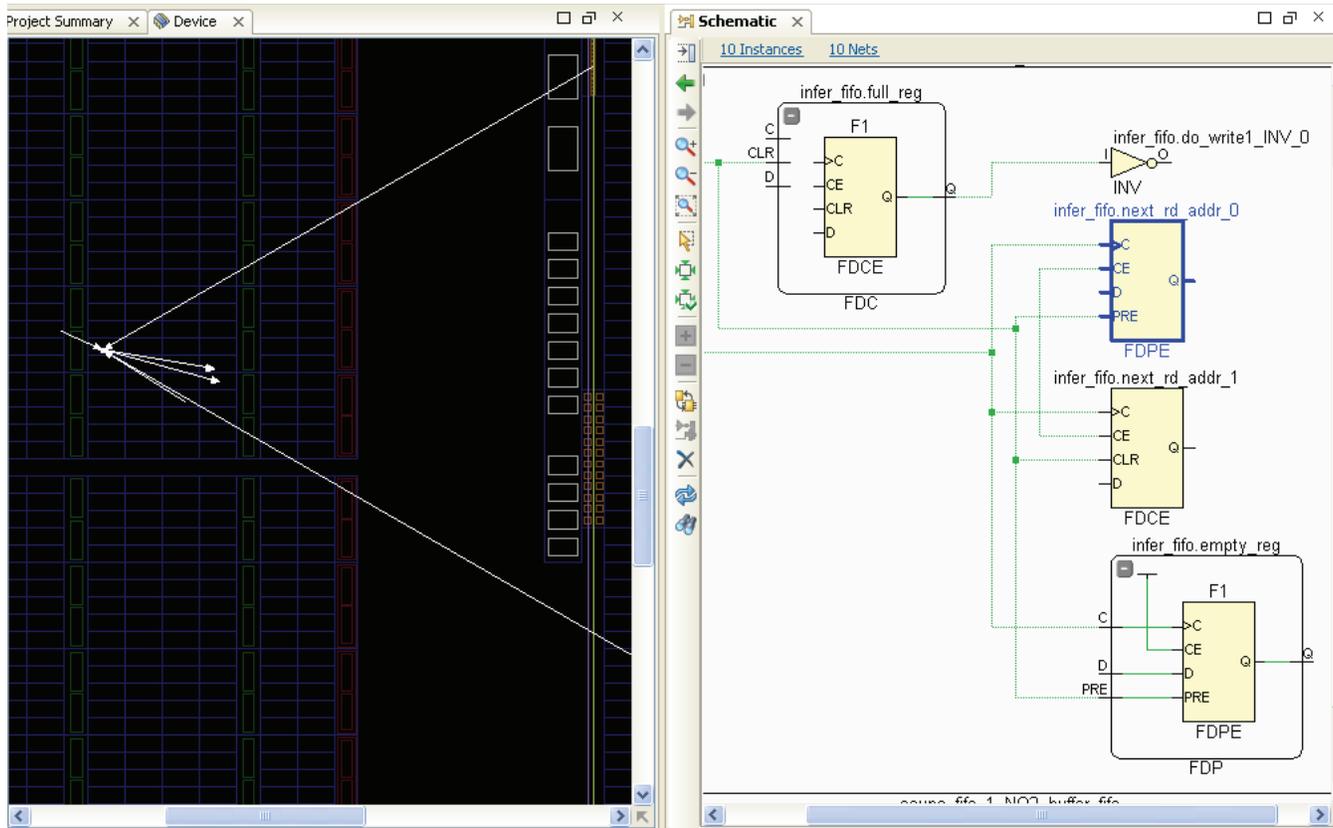
Figure 11-12:   **Logic Expanded in the Schematic View**

Also, you can expand and display instance and module connectivity, and content interactively.

For more information about exploring logic in the schematic, see Using the Schematic View, page 146.

Searching for Objects using the Find Command

After the placement displays in the Device view, you can use **Find** to search for and locate any type of logic. The **Edit > Find** dialog box provides flexibility for filtering the search criteria in many different ways.

For more information about searching for logic objects, see Searching and Replacing in Source Files, page 125.

Using Select Primitives and Highlight Primitives Commands

You can select Pblocks, or logic modules, and use the **Select Primitives** command from the popup menu to select the underlying primitive logic elements in an Implemented Design. This command is often used in conjunction with **Show Connectivity**, **Fix Instances**, or **Clear Placement** to operate

on the placed instances. You can also highlight the selected instances with the **Highlight** command from the popup menu.

You can select the Pblocks or logic modules, and highlight the primitives within those blocks by using the **Highlight Primitives** command from the popup menu, then choose a color to highlight the placed instances. If you have selected multiple Pblocks, or selected multiple modules, you can use the **Highlight Primitives > Cycle Colors** command to use a different color for each module or Pblock. This lets you quickly distinguish different blocks of logic with different colors.

The PlanAhead tool marks modules and primitives in the Netlist view with the matching highlight color in the Device, Schematic, and Package views, as shown in Figure 11-13.
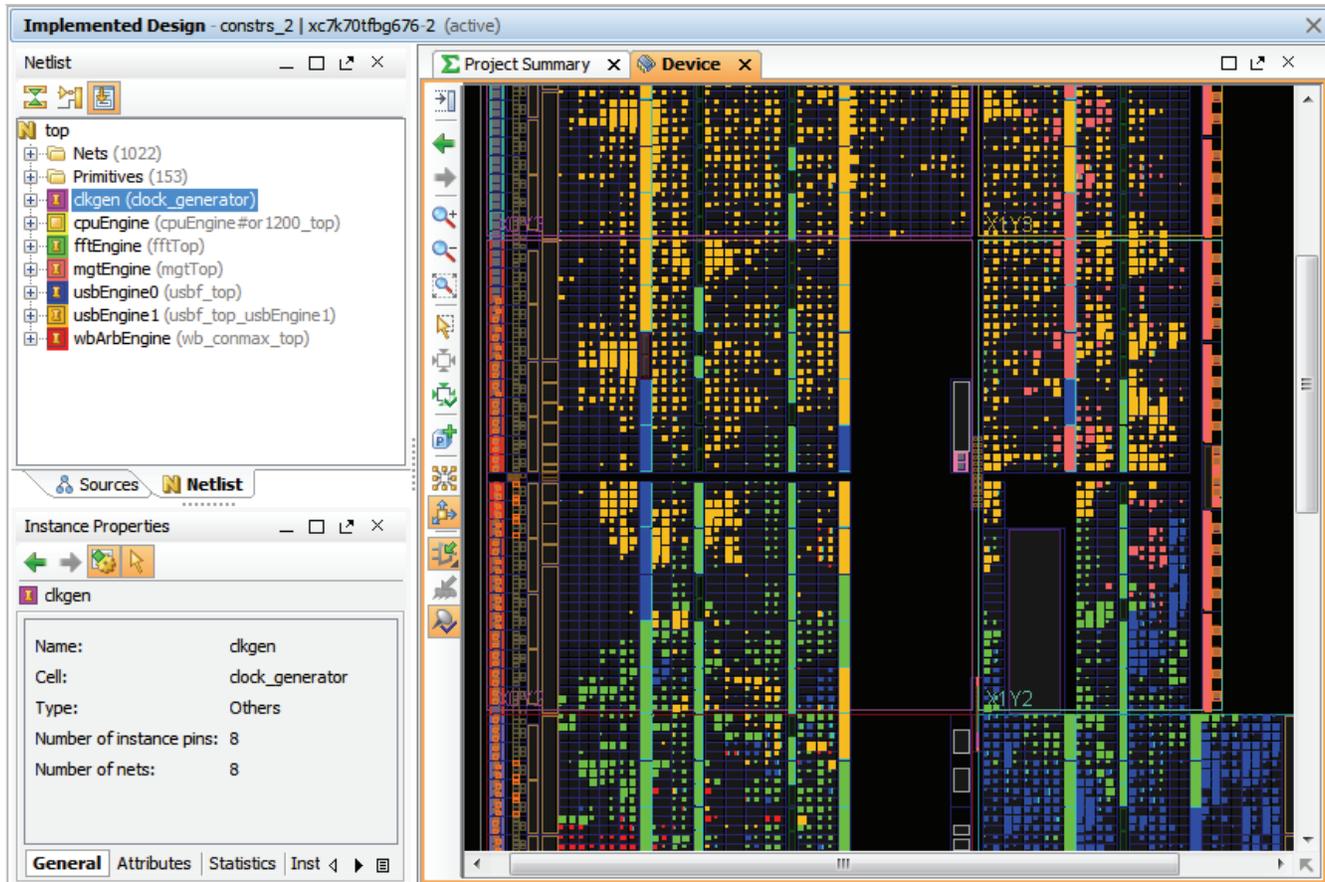
Figure 11-13: **Matching Highlighting Color in the Netlist and Device Views**

Unhighlighting Objects

To un-highlight objects, use one of the following commands:

- **View > Unhighlight All** to unhighlight all objects.
- **View > Unhighlight Color** to unhighlight based on color.
- **Unhighlight All** toolbar button.

Setting Object Selections in the Workspace Views

Object selection is set in the PlanAhead Options dialog box, available when you go to
Tools > Options. For more information about setting object selections, see Customizing Display
Themes, page 173.

Highlighting Selected Objects

You can highlight objects with color for display purposes. Highlighting remains until you clear all
highlights for the floorplan. For more information on highlighting, see Highlighting Selected
Objects, page 121.

Marking and UnMarking Selected Objects

You can place a Mark symbol for all selected objects. Select the objects to mark, and click either:

- **View > Mark**

- **Mark** in the popup menu.

Select the **UnMark All** toolbar button to remove all marks.

Locking Placement for Future Implementation Runs

Placed instances in PlanAhead can be fixed or unfixed. Placed instances that were placed by a
designer are referred to as *fixed*, while placed instances placed by the tool during implementation are
referred to as *unfixed*. Fixed and unfixed instances display with a different color. Figure
Figure 11-14 shows fixed instances in the orange color, while unfixed instances are shown as blue.
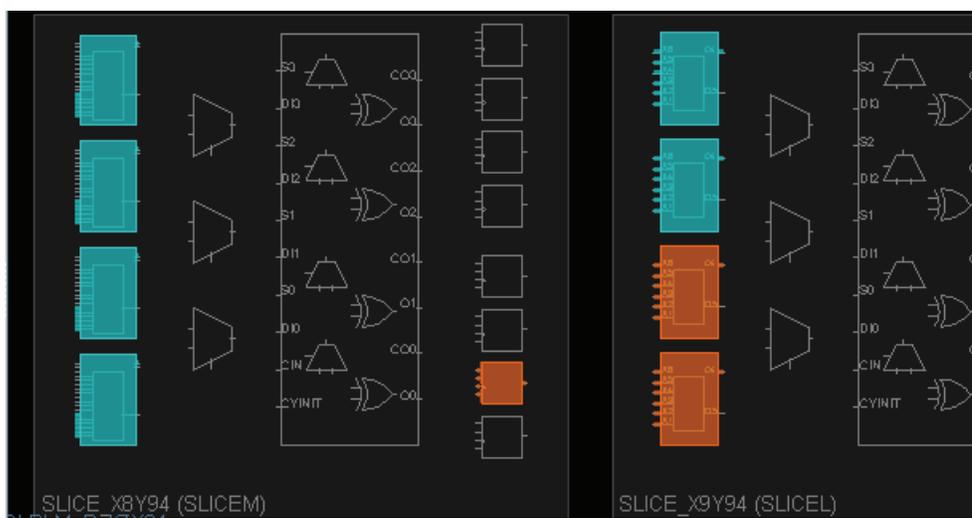
Figure 11-14:   **Fixed and Unfixed Instances**

To lock placement in place for subsequent implementation runs, select the logic, then select **Fix
Instances** from the popup menu.

You can use this feature to help ensure consistent implementation results. After you have saved the
implemented design, the fixed logic receives `LOC` and `BEL` constraints in the UCF, and can be
passed to the subsequent implementation runs. Fixed logic will not be moved during the
implementation process. Unfixed logic, may be moved by the tool as needed to optimize and iterate

on the design. See Understanding Fixed and Unfixed Placement Constraints, page 324 for more information.

Fixing Specific Types of Logic

A method for improving the consistency of implementation results is to lock some or all of the block macro logic such as block RAMs and DSPs. You can do this manually in the PlanAhead tool using your own knowledge of the design, or by leveraging successful ISE implementation results. If your design has many block RAMs or DSPs, this method can help produce more consistent results and improve run time.

To *Fix* (place) specific types of logic, use **Find** to select specific types of logic such as block RAMs and DSPs. Then select them in the Find Results view, and use **Fix Instances**.

Fixing Logic Modules

A method for improving consistency of implementation result is to lock critical logic in place. This can involve locking specific logic, timing paths, or entire logic modules.

To fix (place) all logic in a particular module, select the module or modules and use **Select Primitives** to select the primitive logic instances associated with the logic module. To lock the logic, use **Fix Instances**.

Displaying Design Metrics

The following subsections describe the design metrics options.

Using the Metrics View

The PlanAhead Metrics view displays a list of design metrics that you can display in the Device view, using a colored graph to identify potential trouble spots in the design. The metrics include resource utilization, routing congestion, and timing slack checks in Pblocks, Clock Regions, and Tiles (or CLBs).

To open the Metrics view, select **Window > Metrics**. Figure 11-15, page 351 shows the Metrics view, with a selected metric.
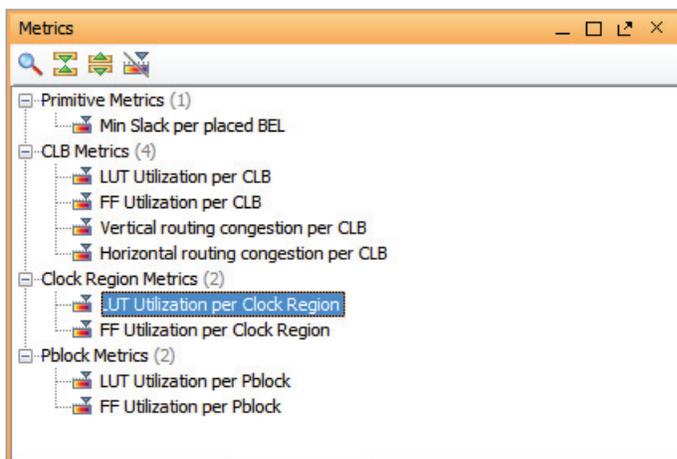
Figure 11-15: **Metrics View**

Displaying the Metric Maps in the Device View

To display a Metric map in the Device view, select the desired metric(s) in the Metrics view, and from the popup menu, select **Show**. The PlanAhead tool displays a color-based metric map. You can also display a specific metric by right clicking in the Device view and selecting from the Metric popup menu. Figure 11-16 displays a metric map for LUT utilization by Clock Region.

You can select multiple metric maps to display simultaneously, though the metrics displayed in the Device view may be difficult to read.

Figure 11-16: **Metric Map in the Device View**

Configuring the Metric Ranges

When a metric is selected in the Metrics view, the Metric Properties view provides a description of the selected Metric, along with the bins defined to highlight potential problems.
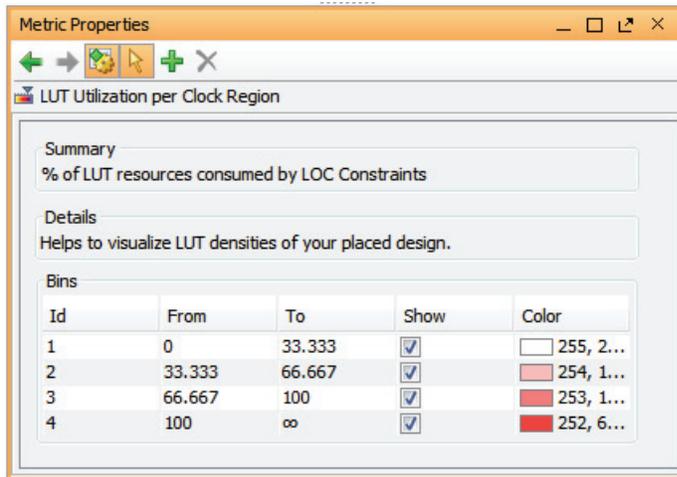
Figure 11-17: **Metric Properties View**

Within the Metric Properties view you can configure the value ranges and display colors used for each metric map, as displayed in Figure 11-17. You cannot edit the ranges of existing bins, but you can add or remove bins, and the range values of remaining bins are adjusted automatically.

Select **Insert Bin,** from the toolbar or popup menu, to add a new value range, and provide greater granularity in the metrics displayed. Figure 11-18 shows a new bin being defined. You can specify the value range for the bin, and specify the color to use. The ranges of existing bins are adjusted to accommodate the newly defined bin.

Adjust the color used for a specific bin by clicking in the **Color** column. You can select from a common color, choose **More Colors** to pick from a greater range of colors, or type the RGB value of a specific color.
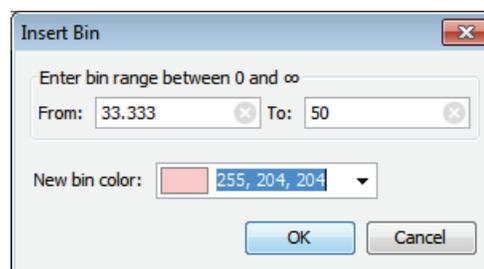
Figure 11-18: **Insert Bin Dialog Box**

Enable or disable the display of the bin by selecting the **Show** checkbox.

You can **Delete** bins to reduce the number of ranges displayed. The range of values is merged into the bin above it in the Metrics Properties view.

Note: The PlanAhead tool automatically adjusts the outside bin with a range from the last user-defined value to infinity or negative infinity.

Select **Apply changes** in the Metric Properties view toolbar menu, or from the popup menu to apply any changes you have made to the metrics properties.

Using the Metric Results View

After you select **Show**, the metric results display in the bottom view. In the Metric Results view, you can:

- Sort the information by clicking any of the column headers.
- Sort by a second column by pressing **Ctrl**, and clicking a second column header.
- Add as many sort criteria as necessary to refine the list order.

The Pblock metric results update automatically as you modify the Pblocks. The different types of metrics such as for Pblocks, CLBs, and primitives, display in different charts. Each type has a tab along the bottom of the Metric Results view, as shown in Figure 11-19.
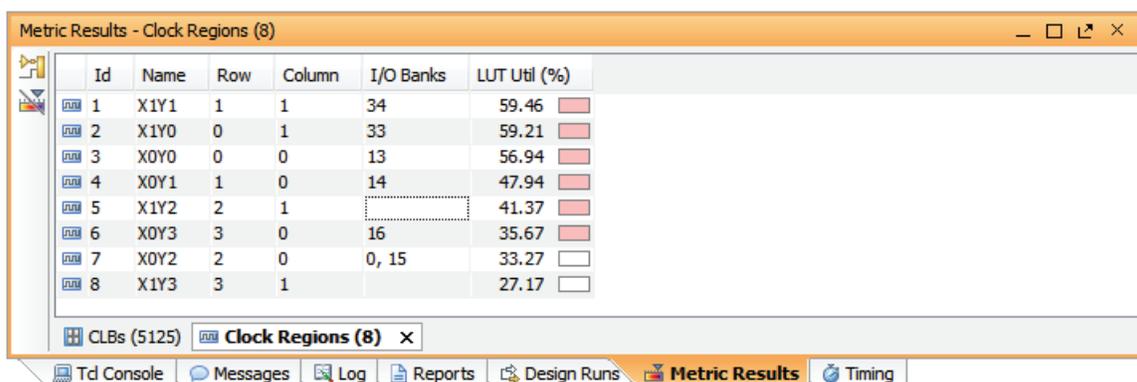
Figure 11-19:   **Metric Results View**

Removing the Metric Map Display

To hide a metric map in the Device view, right-click to open the popup menu and select **Metric > Hide Metrics**. Or, in the Metrics view, select a metric and right-click to open the popup menu and select **Hide Metric** or **Hide All Metrics**.

Performing Timing Simulation

To perform timing simulation on the implemented FPGA design, the PlanAhead tool first invokes the NetGen utility to convert the internal database to a Verilog or VHDL netlist. NetGen is a command line executable that reads Xilinx design files as input and generates a Verilog or VHDL netlist that can be used with third-party simulators and timing analysis tools. For more information on running NetGen, see the *Command Line Tools User Guide, (UG628)* as cited in Appendix E, Additional Resources.

The PlanAhead tool is integrated with the ISE Simulation tool, ISim. Xilinx ISim is a Hardware Description Language (HDL) simulator that lets you perform behavioral and timing simulations for VHDL, Verilog, and mixed VHDL/Verilog designs.

The PlanAhead tool also supports integration with Mentor Graphics® ModelSim or Questa® Advanced Simulator tool for behavioral or timing simulation. See Configuring Project Settings in Chapter 3 for instructions on setting the target simulator.

To launch timing simulation, select:

- **Tools > Simulation > Run Timing Simulation** on an Implemented Design, or
- **Run Timing Simulation** from the Implementation menu of the Flow Navigator.

The PlanAhead tool opens the Launch Timing Simulation dialog box as shown in Figure 11-20.
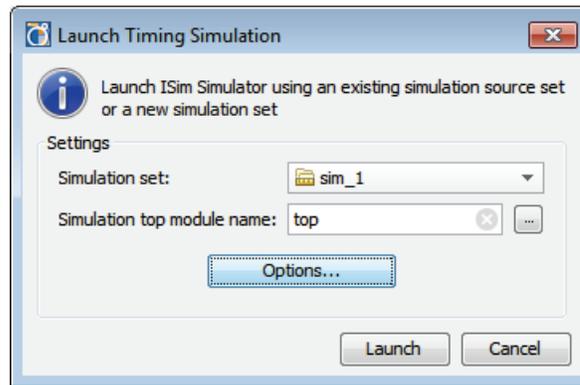
Figure 11-20: **Launch Timing Simulation**

The fields are:

- **Simulation Set**: Specify the name of the simulation run. This lets you create different simulation runs with different design hierarchies and different options.

- **Simulation Top Module Name**: Specify the top-level of the design. The PlanAhead tool automatically populates this field with the defined top module, but you can specify a different top module simulate from different levels of the hierarchy, or to elaborate different variations of the design.

- **Options**: Opens the Simulation Options dialog to define runtime options for the ISim tool. See Specifying Simulation Options for more information.

- **Launch**: Runs the ISim compile and elaboration steps and launches ISim in GUI mode.

- **Cancel**: Closes the dialog box without launching ISim.

Specifying Simulation Options

When you select the Options button, the PlanAhead tool opens the **Simulation Options** form. This form includes the target simulator and clean-up switch, and four option tabs.

- **Target Simulator**: Select either the ISim simulator, or the ModelSim/Questa simulator. The default option is defined in the Simulation Settings. Refer to Configuring Project Settings in Chapter 3 for more information.

    Note: The compilation and simulation options for ModelSim are different from the options for ISim. Refer to ModelSim Options, page 360 for more information.

- **Clean-up simulation files**: Select this checkbox to remove simulation files that are not stored in the /sim directory.

- The four option tabs are:

    - **Compilation Options**: Specify command line options for the compiler used to prepare the design for simulation.

    - **Simulation Options**: Provides run-time directives to be used when the simulation is launched.

    - **Netlist Options:** Define options for the NetGen program used to create the Verilog or VHDL netlist for use in simulation.

    - **Advanced Options:** Include all sources with simulation fileset.

Compilation Options

Click the **Compilation Options** tab of the Simulation Options dialog box, shown in Figure 11-21, to control the execution of fuse and ISim tools.
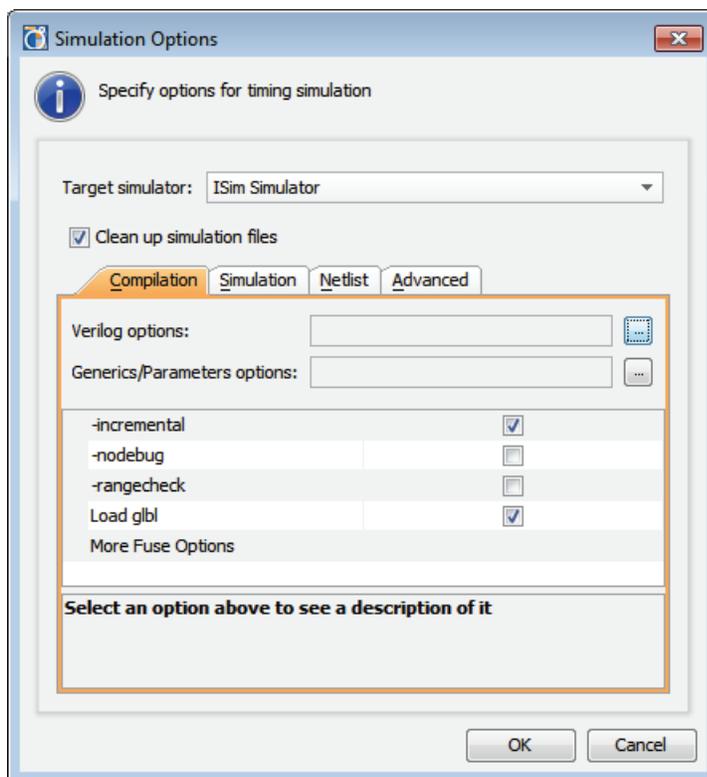
Figure 11-21: **Timing Simulation - Compilation Options**

- **Verilog Options**: Specify Verilog Search Paths, Macro definitions, and Uppercase identifiers. See Configuring Project Settings in Chapter 3 for more information.

- **Generics/Parameters Options**: Specify keywords and values for Generics or Parameters.

- **-incremental**: Switch to indicate that fuse linker and compiler should compile only the files that have changed from the last compilation.

- **-nodebug**: Switch to indicate that fuse should create a simulation executable (.exe) that has no information for debugging your HDL code during simulation, resulting in faster simulation run times.

- **-rangecheck**: Switch to specify that fuse linker and compiler should perform a value range check on VHDL assignments during compilation. This option applies to VHDL code only.

  Note: This does not affect index range checking for arrays which ISim always checks.

- **Load glbl**: Switch to specify that the glbl module should be loaded during compilation. If the design uses a Verilog UniSim or a SimPrim library, you must enable this switch.

- **More Fuse Option**s: Specify additional command line options for fuse. These commands should be typed in a single string with the command value pair. For instance:

  ```
  -maxdelay -init_file <filename> -notimingcheck
  ```

  You can also add the fuse options into a command file, and reference this file in the **More Fuse Options** field with the **-f** command, as follows:

  ```
  -f <command_file>
  ```

Simulation Options

Figure 11-22 shows the **Simulation** tab of the Simulation Options dialog box.[1]
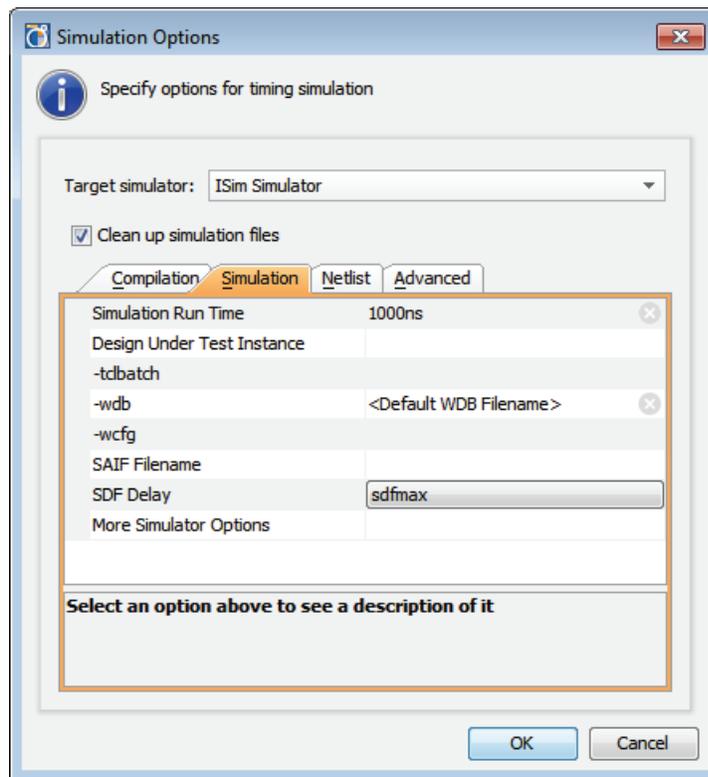
Figure 11-22: **Timing Simulation - Simulation Options**

- **Simulation Run Time**: Specify length of simulation time with time unit. You can specify time in units of fs, ps, ns, us, ms, and sec. The default unit is ps. You can also specify `all` as a keyword to indicate that ISim should run until there are no more events to simulate.

- **Design Under Test**: Specify the name of a unit to test. Typically, this is the same name as the top module; however, in cases where a test bench contains multiple possibilities for the top module, the software prompts you to select the unit to test.

- **-tclbatch**: Specify filename of Tcl commands listed in a batch file for execution by the simulator at run time. Commands in the specified batch file are executed sequentially until completion. ISim ignores any commands entered from the command prompt until batch file execution has completed.

  PlanAhead uses a **tclbatch** command to pass three required commands to ISim in a file called `isim.cmd`. The contents of this file are:

  ```
  onerror {resume}
  .
wave add /
  run <value>
  ```

  If you create a Tcl command to control the execution of the simulator at launch time, you must include these three commands in your `tclbatch` file. It is recommended that `onerror` be

---

1. Simulation Options will be different when ModelSim or Questa is the target simulator.

the first command listed, and that `wave add` and `run` be the last commands listed. You can add any other ISim command-line commands between `onerror` and `wave add`.

Note: The `tclbatch` command file must have a file extension of either TCL or CMD to be properly handled by ISim.

- **-wdb**: Specify a filename to save the simulation waveform data. The simulation results of the signals being traced are saved to the specified filename in the working directory. The PlanAhead tool creates a `<top_module_name>.wdb` file by default.

- **-wcfg**: Specify a waveform configuration filename to use when opening the waveform data in the ISim GUI. The wave configuration file specifies settings such as the signal order, name style, radix and color.

- **SAIF Filename:** Specify the filename of a Switching Activity Interchange Format (SAIF) file recording port and signal switching rates. The default filename is `xpower.saif`.

- **SDF Delay**: Specify the type of delays for ISim to use. Values are:

  - `sdfmin` to annotate minimum delays

  - `sdfmax` to annotate maximum delays

  Xilinx recommends running separate simulations to check for setup violations by specifying `sdfmax` and for hold violations by specifying `sdfmin`.

- **More Simulator Options**: Specify additional command line options for ISim. The commands must be in a single string with the command value pair. For example:

  `-log <filename> -transport_int_delays`

  You can also add the ISim options into a command file, and reference this file in the **More Simulator Options** field with the **-f** command, as follows:

  `-f <command_file>`

Netlist Options

The **Netlist Options** tab, as shown in Figure 11-23, page 359, provides options to control the NetGen program for generating a Verilog or VHDL netlist from the implemented FPGA design.

The commands and options on this tab specify how NetGen writes the simulation netlist:

- **-insert_pp_buffers**: Switch to control whether path pulse buffers are inserted into the output netlist to eliminate pulse swallowing. Pulse swallowing is seen in timing simulations when the pulse width is shorter than the delay on the input port of the component.

- **-fn**: Switch to output a flattened netlist versus a hierarchical netlist.

- **-a**: Switch to generate only architectures in the VHDL output. This option suppresses generation of VHDL entities in the output.

- **-sdf_anno**: Specify whether the sdf_annotate statement should be added to the netlist output by Netgen for delay annotation.

- **-tm**: Specify a new name for the top module to be used in the output from NetGen.

- **-extid**: Switch to output extended identifiers in VHDL output.

- **-sdf_path**: Specify the path to output the SDF file created by NetGen. By default the SDF file is output to the simulation run directory.

- **-ism**: Switch to include SimPrim modules from the SimPrim library in the output Verilog file. This option lets you avoid specifying the library path during simulation, but increases compile time and the size of the output netlist.

- **More NetGen Options**: Specify additional command line options for NetGen. These commands must be typed in a single string with the command value pair. For example:

  ```
  -aka -gp <port_name> -s 3
  ```

  You can also add all of the NetGen options into a command file, and reference this file in the **More Netgen Options** field with the **-f** command, as follows:

  ```
  -f <command_file>
  ```
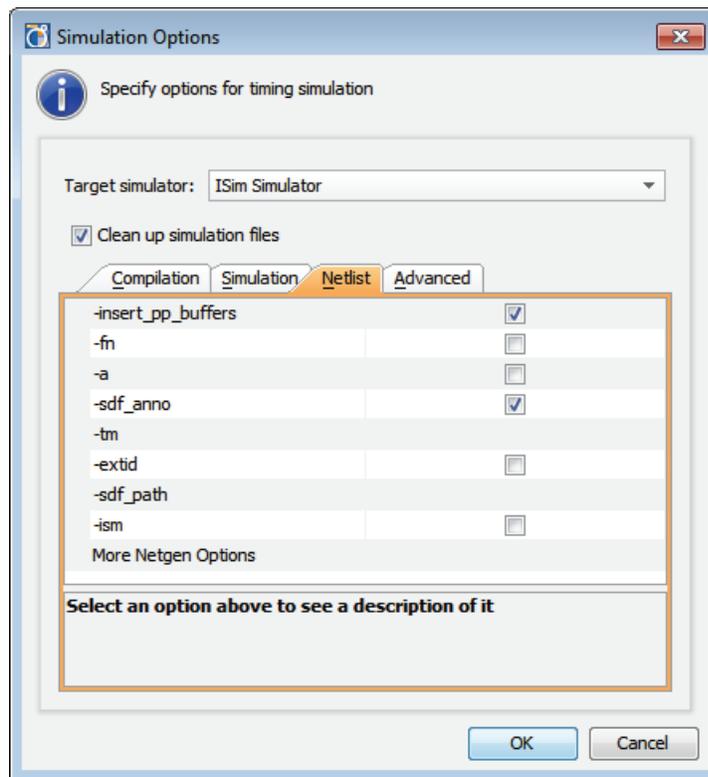

Figure 11-23: **Timing Simulation - Netlist Options**

Advanced Options

The **Advanced Options** tab, as shown in Figure 11-24, page 359, provides an option to include all design sources in the simulation fileset.

Unselecting this option gives you the flexibility to include only the files you want to simulate.
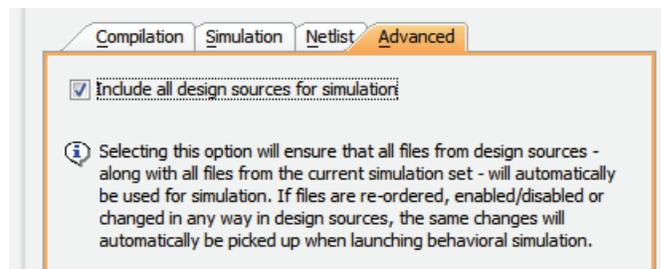
Figure 11-24: **Timing Simulation - Advanced Options**

ModelSim Options

When the target simulator has been set to QuestaSim/ModelSim, the compilation and simulation options presented by the tool will be different. The following are the options for ModelSim:

Compilation

- **VHDL Syntax**: Specifies the VHDL language version used by the source files and test bench. Valid values are 93 (1993) and 87 (1987).

- **Explicit Declarations**: Resolve ambiguous function overloading by using the explicit function declaration versus the function declaration automatically created by the compiler.

- **More VLOG Options**: Provide any added Verilog compiler (vlog) command line options to pass to the tool.

- **More VCOM Options**: Provide any added VHDL compiler (vcom) command line options to pass to the tool.

Simulation

- **Simulation Run Time**: Specify length of simulation time with time unit.

- **Log All Signals**: Save all signal values during simulation to help trace the source of an unknown signal state.

- **SDF Delay**: Specify the type of delays for ModelSim to use. Values are:

  - `sdfmin` to annotate minimum delays

  - `sdfmax` to annotate maximum delays

  Xilinx recommends running separate simulations to check for setup violations by specifying `sdfmax` and for hold violations by specifying `sdfmin`.

- **Design Under Test Instance:** The name of the DUT module defined in the test bench.

- **SAIF Filename:** Specify the filename of a Switching Activity Interchange Format (SAIF) file that records port and signal switching rates.

- **More VSIM Options**: Provide any added ModelSim (vsim) command line options to pass to the tool.

The ISim Tool

Click the **Launch** button on the Launch Timing Simulation dialog box to invoke ISim (shown in Figure 11-20, page 355). The PlanAhead tool runs NetGen, which reads the NCD file from MAP or PAR and creates a partial or full timing Standard Delay Format (SDF) netlist based on the results.

Before launching a timing simulation, a timing simulation model and delay file for back-annotation are required. PlanAhead uses the NetGen tool to generate these files. For more information on running NetGen, see the *Command Line Tools User Guide, (UG628)* as cited in Appendix E, Additional Resources.

NetGen creates the flattened timing delays in the output SDF file, and creates a verilog (or VHDL) netlist for simulation purposes (`top_timing_sim.v`).

Note: The output Verilog or VHDL file is for simulation purposes only, and cannot be synthesized.

The Verilog or VHDL netlist created by NetGen references the Xilinx simulation primitive library (SimPrim) and so must be used with SimPrim during simulation. When NetGen completes, The ISim object compiler and linker, fuse, is run to compile and elaborate the Verilog and VHDL code.

Then the compiled object codes are linked into a simulation executable named after the top module specified in the ISim Launch dialog. When the ISim executable is complete, the PlanAhead tool launches the simulator:

```
INFO: [Runs-8] Fuse completed.
INFO: [Runs-10] Launching ISim...
INFO: [Runs-11] Running '"C:/project_cpu_hdl/project_cpu_hdl.sim/
sim_1/top.exe"
-intstyle pa -gui -tclbatch ISim.cmd
-wdb "wdb_test1.wdb" -view "wcfg_test1.wcfg"'
```

The simulation executable is run with the various options specified in the Launch Options form.

The PlanAhead tool launches ISim with the `-gui` option. This opens the ISim GUI so you can interactively simulate the design. For information about running ISim through the GUI, see the *ISim User Guide, (UG660)* as cited in Appendix E, Additional Resources.

Figure 11-25 shows the ISim GUI.
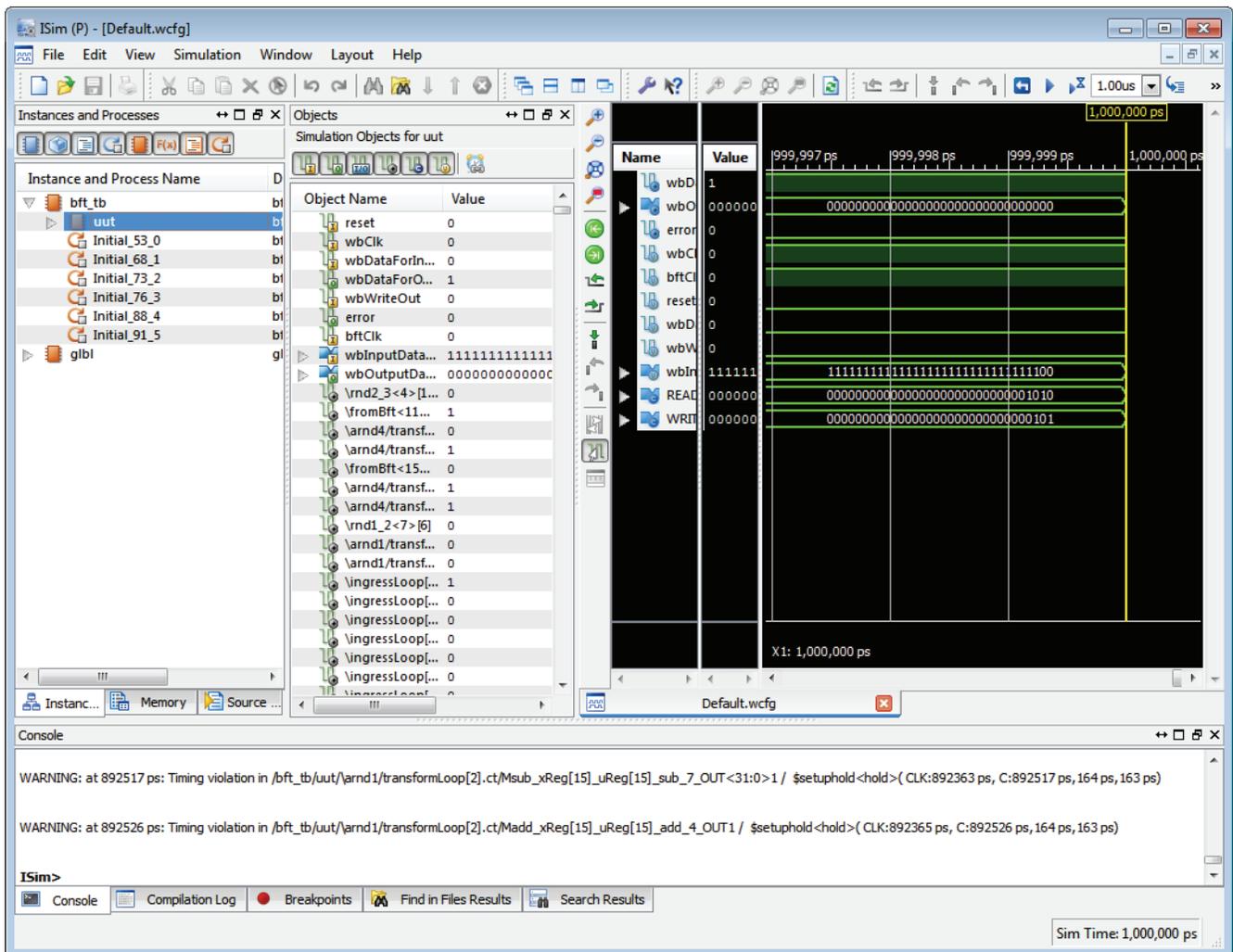
Figure 11-25:   **ISim User Interface**

Analyzing Power Distribution with XPower Analyzer

You can launch the XPower Analyzer (XPA) tool directly from the PlanAhead tool to perform power analysis on an implemented design.

- Select the **XPower Analyzer** command from the Flow Navigator.

- Select **Tools > Analysis > Xpower Analyzer** from the main menu.

The Launch XPower Analyzer dialog box is open as shown in Figure 11-26, page 362.
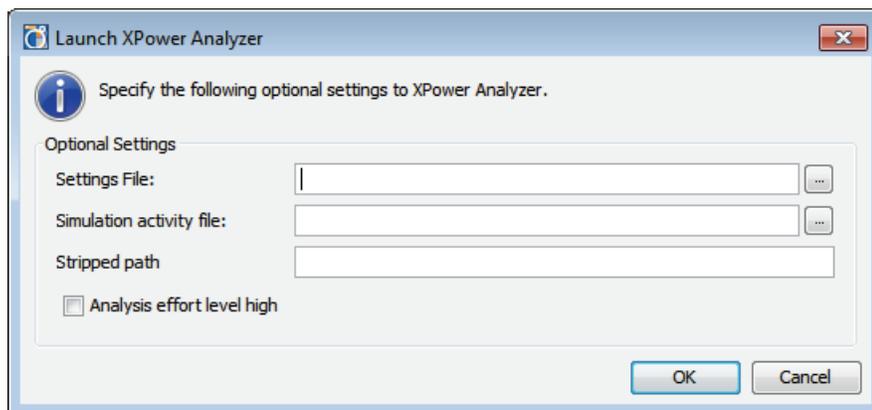
Figure 11-26: **Launch XPower Analyzer**

Specify the options for the XPA command:

Settings File: Specifies an existing XPower Analyzer file (`.xpa`) to let you import the settings from a previous run for use in this run.

Simulation Activity File: Reads a Switching Activity Interchange Format (SAIF) file, or Read a Value Change Dump (VCD) file, for use during power analysis.

Stripped Path: Strip the specified instance path prefix from elements in the SAIF or VCD file to allow them to be mapped properly to instances in the current design.

Analysis Effort High: A high effort level will increase accuracy as well as runtime.

The routed NCD file and timing constraints (PCF file) are passed to XPower Analyzer when it launches. Figure 11-27, page 363 shows the XPower Analyzer tool. For more information on using XPower Analyzer, see the ISE Help.
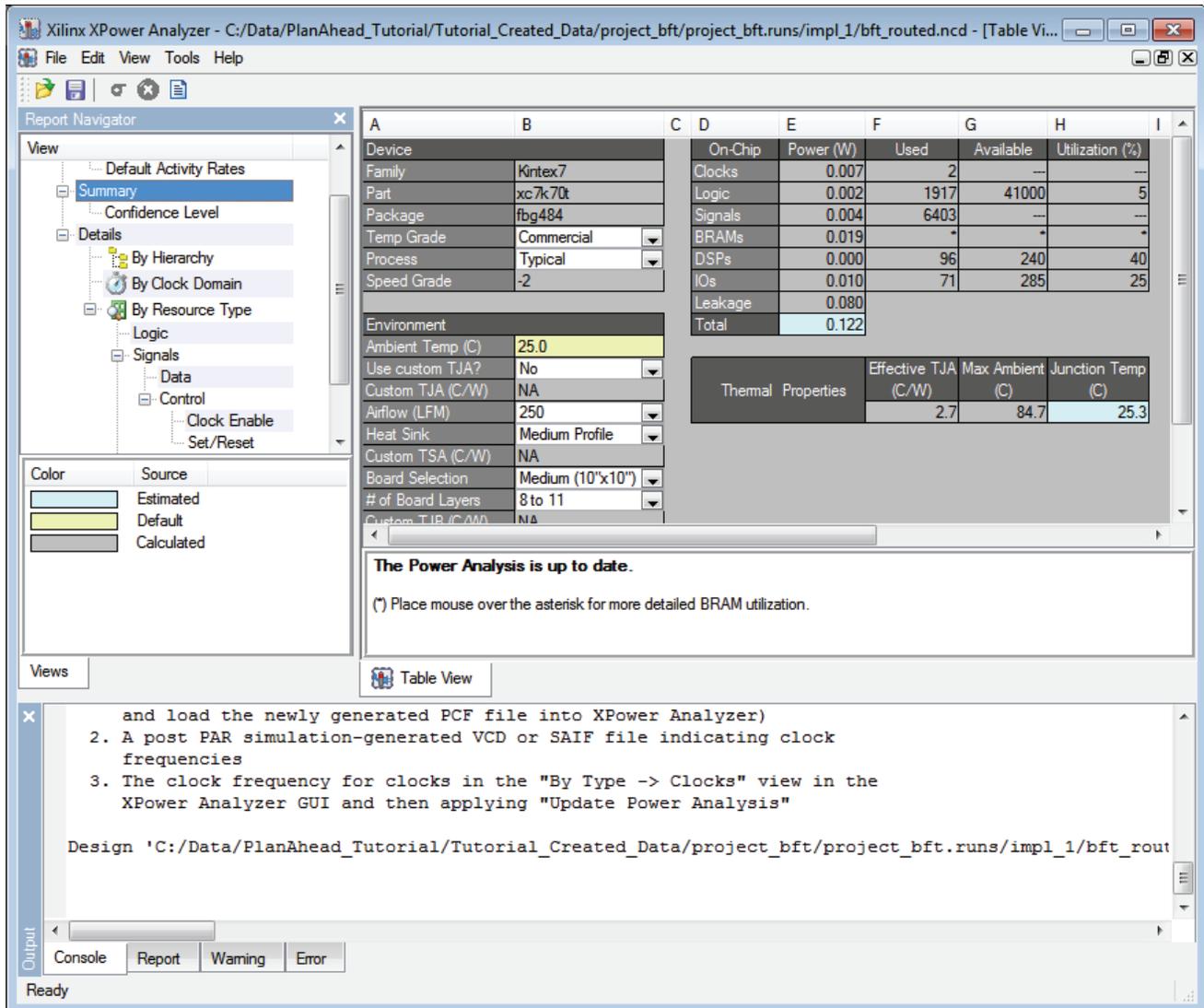
Figure 11-27: **XPower Analyzer**

Launching FPGA Editor

You can launch the FPGA Editor directly from the PlanAhead tool on any implemented design. To invoke the FPGA Editor:

- Select **FPGA Editor** from the Implementation menu of the Flow Navigator.

- Select **Tools > Analysis > FPGA Editor** from the main menu.

The routed NCD file passes automatically to the FPGA Editor, and the implemented design is opened. For more information on using FPGA Editor, see the ISE Help.

Cross Probing Timing Paths to FPGA Editor

To cross-probe from a timing path in the PlanAhead tool to FPGA Editor, select a timing path from the Timing Results view or the Device view, and select **Cross probe to FPGA Editor**. You can also select individual logic instances to cross probe to FPGA Editor.

FPGA Editor opens with the selected path or instance highlighted, as shown in Figure 11-28.
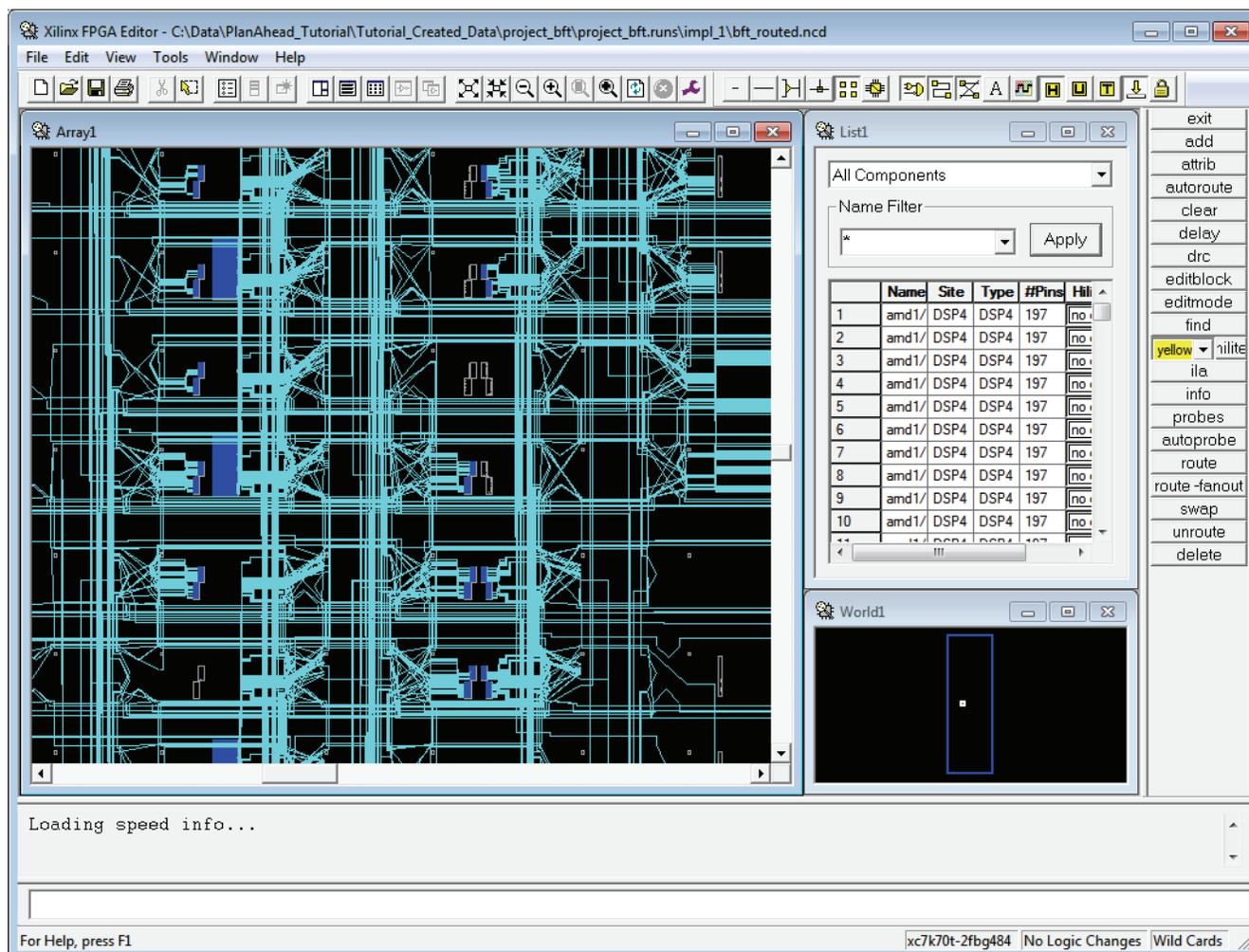
Figure 11-28: **FPGA Editor**

Chapter 12

Programming and Debugging the Design

This chapter discusses programming and debugging the design, and includes:

- Generating Bitstream Files
- Debugging the Design with ChipScope
- Launching ChipScope Pro Analyzer
- Launching iMPACT

Generating Bitstream Files

After Implementation is successful, you can run the ISE® BitGen command on the Implemented Design to create the bitstream data to program the Xilinx device.

1. Specify BitGen command line options:
   - Click **Bitstream Settings** in the Flow Navigator, or
   - Select **the Flow > Bitstream Settings** command from the main menu.

   The Bitstream Project Settings dialog box opens as shown in Figure 12-1, page 366.

   You can set ISE BitGen options prior to running the command.When you select an option, a description of the option displays in the dialog box. For more information about BitGen options, see the *Command Line Tools User Guide (UG628)*, cited in Appendix E, Additional Resources.

2. Click **OK** or **Apply** to set the selected options.
3. Run BitGen:
   - Click **Generate Bitstream** in the Flow Navigator, or
   - Select **Flow > Generate Bitstream** from the main menu.

   You can view the command status in the Log view and Messages view and the BitGen report file in the Reports view after it completes.

   PlanAhead generates the resulting bit file in the project Run directory.
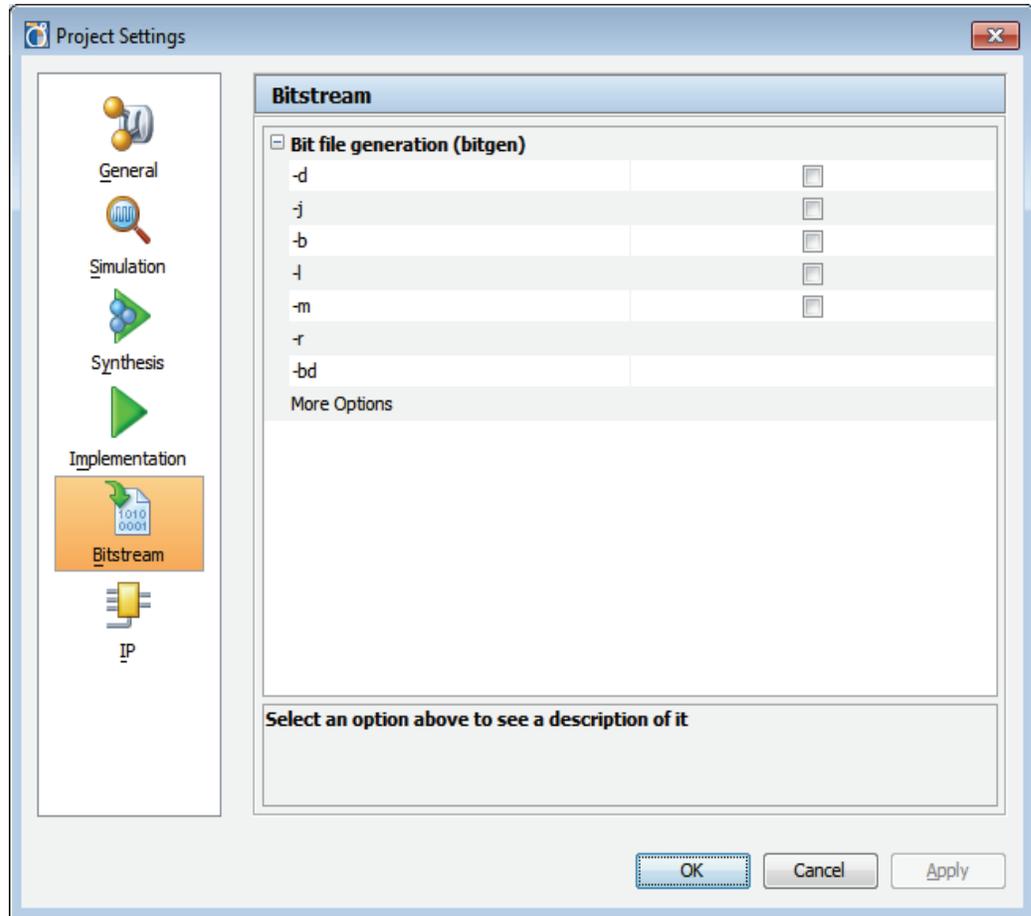
Figure 12-1:   **Generate Bitstream Dialog Box**

Debugging the Design with ChipScope

The PlanAhead™ application is integrated with the ChipScope™ Pro debug software.

The ChipScope Pro integration provides simplified post-Synthesis insertion and connection of the ChipScope Pro Integrated Logic Analyzer (ILA) debug cores in the PlanAhead tool.

Overview of ChipScope Integration in PlanAhead

PlanAhead provides a GUI wizard for quick and easy design debug for most situations. A non-wizard GUI and Tcl command flow are available for precision debug core and net connection control. This flow provides a robust ILA core connection solution without leaving the PlanAhead tool. Figure 12-2, page 367 illustrates the debug core integration.
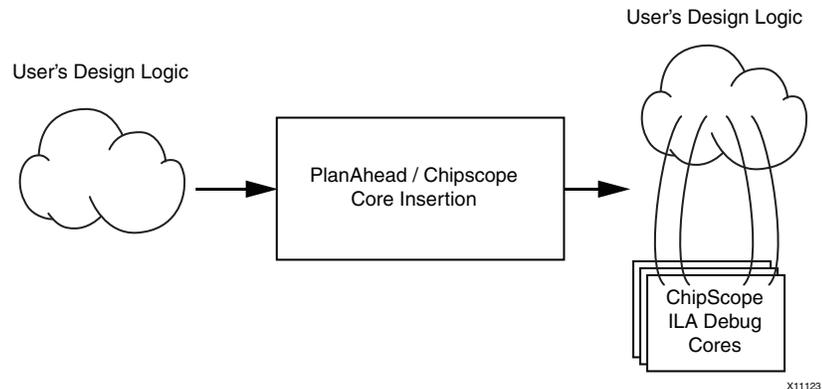
Figure 12-2:   **Block Diagram of PlanAhead/ChipScope Integration**

Requirements and Limitations When Using Core Insertion Flow

To use the ChipScope integration features in PlanAhead, you must have Xilinx® ISE Design Suite 14.x tools, including ChipScope Pro and PlanAhead, installed.

To perform run time design debugging you must also have the Xilinx Targeted Design Platform (TDP) USB cable and board. For more information about ChipScope Pro, see http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/design_tools/ chipscope_pro.html, or refer to the *ChipScope Pro Software and Cores User Guide* (UG029) as referenced in Appendix E, Additional Resources.

The ChipScope integration in the PlanAhead tool has the following limitations:

- The same software versions of PlanAhead, ISE, and ChipScope Pro must be used with this flow. Mixing and matching versions is not supported.

- This flow is not available with the Project Navigator or the ChipScope Pro Core Inserter flow. However, you can import ChipScope Debug Cores (CDC) into PlanAhead.

- This flow is not available when PlanAhead is launched from ISE Project Navigator.

- You can view, but not change, pre-existing debug cores connected to a ChipScope Pro Integrated Controller (ICON) core.

- This flow is not compatible with a pre-existing ICON core that was generated without a BSCAN primitive and that requires connection to a BSCAN primitive instantiated outside of the core.

- Because the PlanAhead tool adds debug cores to the post-synthesis design netlist, some nets might be unavailable for debugging due to trimming or other optimization that takes place during the synthesis process.

- Only ChipScope Pro ILA cores can be created and connected using this flow.

Using the Core Insertion Flow

Insertion of ChipScope debug core in the PlanAhead tool is presented in a layered approach to address different needs of the diverse group of PlanAhead users:

- The highest level is a simple GUI wizard that creates and configures ILA cores automatically based on the selected set of nets to debug.

- The next level is the main ChipScope view allowing control over individual cores, ports and their parameters. The ChipScope view can be displayed by selecting the ChipScope view

layout from the Layout Selector or the Layers menu, or can be opened directly using
Window > ChipScope.

* The lowest level is the set of Tcl debug commands that you can enter manually or replay as a script.

You can use a combination of the modes to insert and customize debug cores also.

Deciding Which Debug Core Insertion Mode to Use

The following table summarizes how to decide what insertion mode(s) to use based on the debugging goal.

Table 12-1: **Debugging Goals and Core Insertion Modes**

Debugging Goal	Core Insertion Mode
Quickly create ILA debug core(s) with default settings for the selected nets.	ChipScope wizard
Change parameters on existing debug cores.	ChipScope view
Manually create or delete existing debug cores.	ChipScope view
Manually create, delete, or configure trigger or data ports on an ILA core.	ChipScope view
Manually assign nets to the trigger/data/clock channels.	ChipScope view
Play back a recorded script of debug commands later.	Tcl Commands

Marking Nets for Debug

The first step in the debug flow is to identify the set of nets to debug.

ChipScope debug core insertion and configuration operations must be performed in the Synthesized Design because you must add the cores to the netlist prior to Implementation. See Using the Synthesized Design Environment, page 209 for more information.
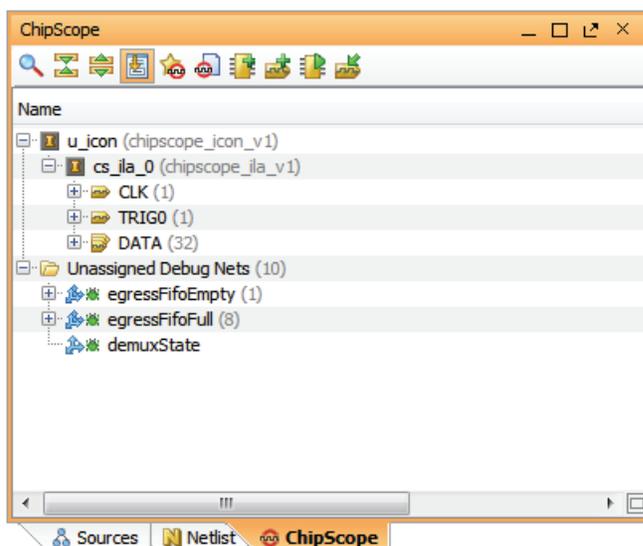
Figure 12-3: **Unassigned Nets List in ChipScope Window**

The PlanAhead tool lets you make debug net selection by picking a set of nets or buses in the Netlist view or the Schematic view, and either selecting **Mark Debug**, or dragging and dropping the net from the Netlist view in the /Unassigned nets folder. You can perform net selection by selecting nets or buses in any view including the Schematic view. The ChipScope view is shown in Figure 12-3, page 368.

There is also a net selector built into the Setup ChipScope wizard.

You can also identify the nets to debug prior to Synthesis. Nets marked for debug in HDL or constraint files are automatically listed in the ChipScope view under the /Unassigned nets folder, and in the Set up ChipScope wizard.

The procedure for marking nets for debug depends on whether you are working with an RTL source-based project or a synthesized netlist-based project.

For an RTL netlist-based project:

- Using Xilinx Synthesis Technology (XST) you can optionally mark nets for debug using the mark_debug constraint in VHDL and Verilog sources, or in a Xilinx Constraint File (XCF) source. For more information about the Mark Debug constraint, see the *Constraints Guide (UG625)*, cited in Appendix E, Additional Resources.

  In addition to the boolean string values of, "true" or "false," a value of "soft" allows the software to optimize the specified net, if possible.

  Note: XST automatically supports this constraint for Spartan-6, Virtex-6, and newer devices.

For a synthesized netlist-based project:

- Using the Synopsys® Synplify® synthesis tool, you can optionally mark nets for debug using the mark_debug and syn_keep constraints in VHDL or Verilog, or using the mark_debug constraint alone in the Synopsys Design Constraints (SDC) file. Synplify does not support the "soft" value, as this behavior is controlled by the syn_keep attribute.

- Using the Mentor Graphics® Precision® synthesis tool, you can optionally mark nets for debug using the mark_debug constraint in VHDL or Verilog.

The following subsections provide syntactical examples for XST, Synplify, and Precision source files.

XST Syntax Examples

The following are examples of VHDL, Verilog, and XCF syntax when using XST.

VHDL Syntax Example

```
attribute mark_debug : string;
attribute mark_debug of char_fifo_dout: signal is "true";
```

Verilog Syntax Example

```
(* mark_debug = "true" *) wire [7:0] char_fifo_dout;
```

XCF Syntax Example

```
BEGIN MODEL "wave_gen"
NET "char_fifo_dout" mark_debug= "true";
END;
```

Synplify Syntax Examples

The following are examples of Synplify syntax for VHDL, Verilog, and SDC.

VHDL Syntax Example

```
attribute syn_keep : boolean;
attribute mark_debug : string;
attribute syn_keep of char_fifo_dout: signal is true;
attribute mark_debug of char_fifo_dout: signal is "true";
```

Verilog Syntax Example

```
(* syn_keep = "true" *) (* mark_debug = "true" *) wire [7:0]
char_fifo_dout;
```

SDC Syntax Example

```
define_attribute {n:char_fifo_din[*]} {mark_debug} {"true"}
```

Note: Net names in an SDC source must be prefixed with the "n:" qualifier. See About SDC, page 396 for more information.

Precision Syntax Examples

The following are examples of VHDL and Verilog syntax when using Precision.

VHDL Syntax Example

```
attribute mark_debug : string;
attribute mark_debug of char_fifo_dout: signal is "true";
```

Verilog Syntax Example

```
(* mark_debug = "true" *) wire [7:0] char_fifo_dout;
```

Using the ChipScope Wizard for Debug Core Insertion

The Set up ChipScope debug wizard is the easiest and fastest way to add debug cores in the PlanAhead tool.

To use the Set Up ChipScope wizard to insert debug cores:

1. Optionally, select a set of nets for debug either using the unassigned nets list or direct net selection.

2. From the main menu, click **Tools > Set Up ChipScope**.

3. Follow the instructions on the Set up ChipScope wizard screen-by-screen to connect and configure the debug cores.

Importing a ChipScope CDC File

The Set up ChipScope wizard lets you add an existing ChipScope Debug Core (CDC) file to the project.

When you click the **Set up ChipScope** box, the first screen of the wizard provides a checkbox to Import Existing ChipScope CDC file.

Select the checkbox, and click **Next**.

Select the CDC file, and click **Next**.

Note: Not all ChipScope cores are importable. CDC files originating from the ChipScope Core Inserter or PlanAhead contain the required core information required for import.

Selecting or Confirming Debug Nets

If you have added nets to the Unassigned Nets list, you are prompted to use them or to select new nets. The PlanAhead tool invokes an Add/Remove Nets dialog box to search for and select nets to debug.

Add or remove nets as needed, and click **Next**.

Specifying Debug Nets and Clock Domains

The ChipScope wizard attempts to detect the correct clock automatically for each selected net or bus as shown in Figure 12-4.
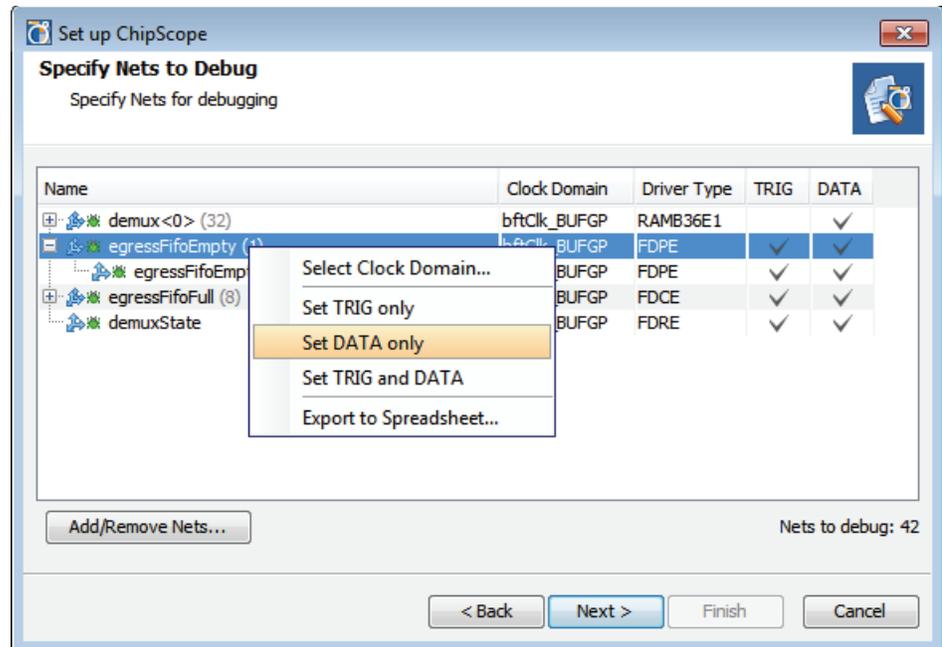
Figure 12-4: **Specifying Debug Nets and Clock Domains**

If multiple clocks are detected for a given net, a drop-down list allows the selection of different clocks for the net or bus.

1. To modify the debug net selection further, click **Add/Remove Nets**.

2. Configure each net or bus for use as a trigger, data storage, or both.

3. When the net and clock configuration is correct, click **Next** to proceed to the summary screen.

    If ILA cores exist in the design, you are prompted to remove them and regenerate based on the new information or to keep them intact and generate new cores.

Inserting ILA Cores

The ChipScope wizard inserts one ILA core per clock domain. The nets that were selected for debug are assigned automatically to the trigger and data ports of the instantiated ILA cores.

The last wizard screen shows the core creation summary displaying the number of clocks found and ILA cores to be created and/or removed.

If you are satisfied with the results, click **Finish** to instantiate and connect the ILA cores in the design.

Using the ChipScope Window to Add and Customize Debug Cores

The main ChipScope view provides more fine-grained control over ILA core insertion than what is available in the ChipScope wizard.

The controls available in this window allow core creation, core deletion, debug net connection, and core parameter changes.

The main ChipScope view:

- Shows the list of debug cores that are connected to the ICON controller core.

- Maintains the list of unassigned nets at the bottom of the window.

You can manipulate debug cores and ports from the popup menu or the toolbar buttons on the top of the view.

Creating and Removing Debug Cores

To create ChipScope debug cores in the ChipScope view, click **Create Debug Core**.

Using this interface, you can change the parent instance, debug core name, and set parameters for the core.

To remove an existing debug core, in the ChipScope view, select the core and select **Delete**.

Adding, Removing, and Customizing Debug Core Ports

In addition to adding and removing debug cores, you can add, remove, and customize ports of each debug core to suit your debugging needs. To add a new port:

1. Select the core.

2. Click **Create Debug Port**.

   The Create Debug Port dialog box opens, as shown in Figure 12-5.
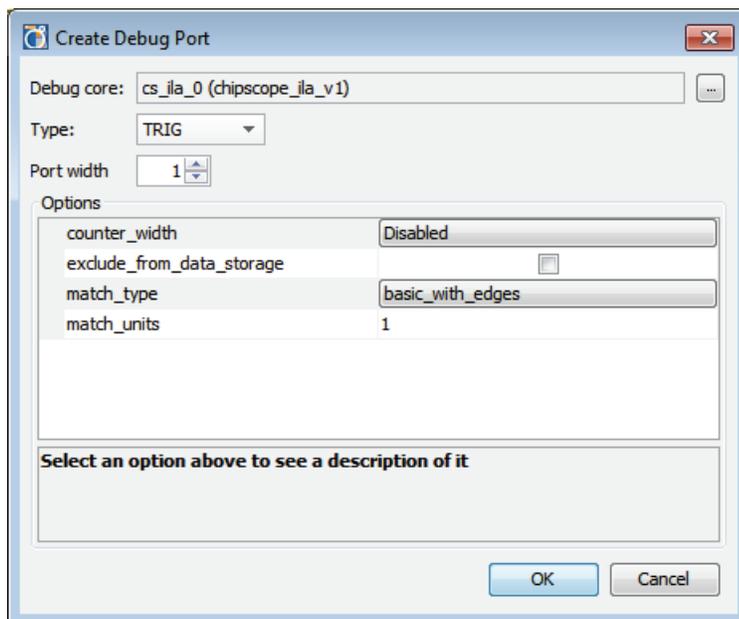
Figure 12-5: **Customizing Ports and Options of Debug Cores**

3. Select the port type in the drop-down.

Any configurable options for the port display in the Options area. The port width starts as a default, but expands or contracts as you add or remove nets from the port.

4. Click **OK**.

To remove a debug port, in the **ChipScope** tab select the port, and select **Delete**.

Connecting and Disconnecting Nets to Debug Cores

You can select, and then drag and drop nets and buses (vectors of nets) from the Schematic view or the Netlist view onto the debug core ports (shown in the following figure). This expands the port as needed to accommodate the net selection.

Also, you can right-click on any net or bus, and select **Assign to ChipScope Debug Port**.

To disconnect nets from the debug core port, select the nets that are connected to the debug core port, and click **Disconnect Net**. Figure 12-6 illustrates these activities.
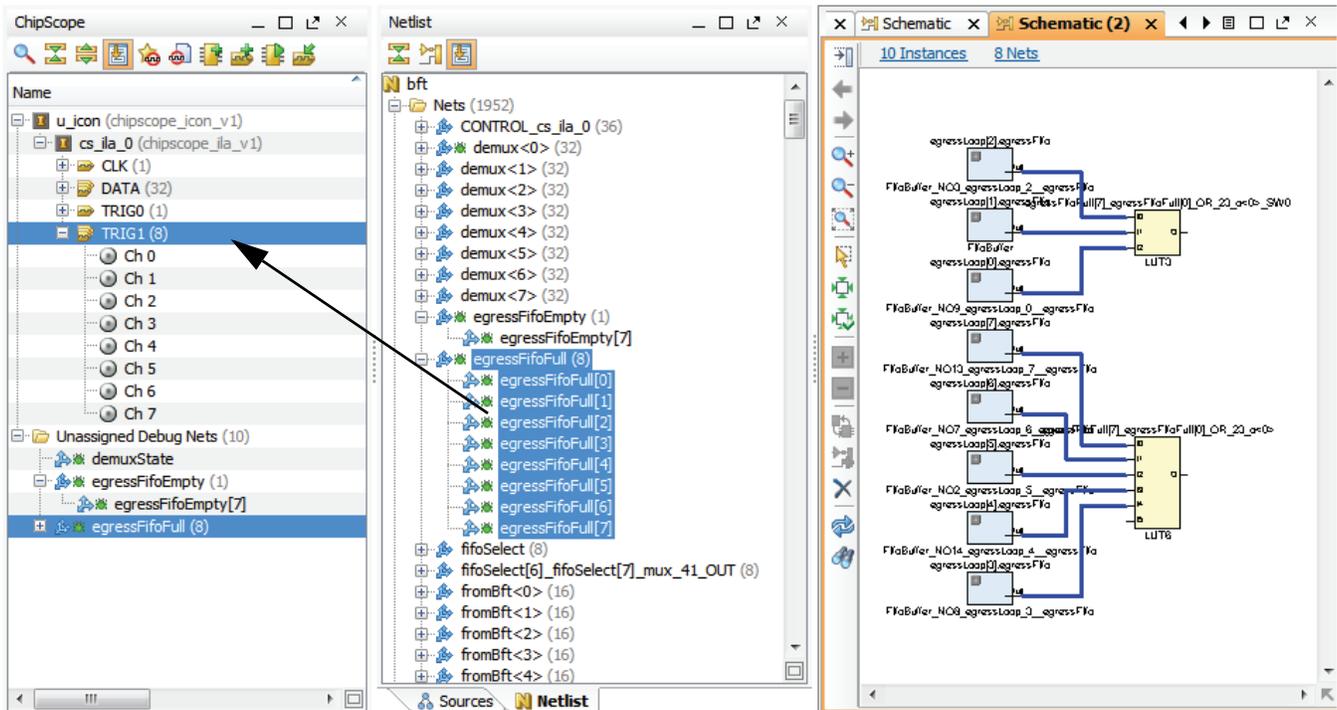
Figure 12-6: **Dragging and Dropping Nets onto Debug Core Ports**

Customizing Debug Core and Port Parameters

ChipScope Debug cores have parameters that can be customized.

To access these core parameters:

1. In the ChipScope view, select one of the ChipScope debug cores.

This opens the Instance Properties view with the selected debug core.

2. In the Instance Properties view, select **Debug Core Options** to view and configure the debug core parameters as shown in Figure 12-7, page 374.
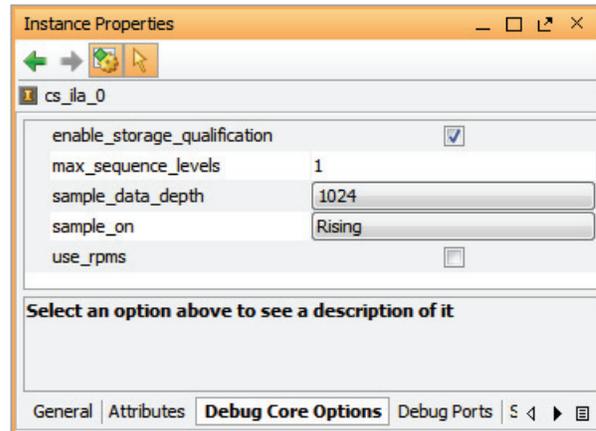
Figure 12-7: **Instance Properties - Debug Core**

Implementing Debug Cores

The PlanAhead tool creates ChipScope Pro ICON and ILA cores initially as black boxes. These cores must be implemented prior to running through MAP and PAR.

ChipScope debug core implementation is automatic when running the implementation using **Run Implementation** from the Flow Navigator or Tools menu; however, you can also force debug core implementation manually for floorplanning or timing analysis. To manually implement the cores:

- Click on the **Implement ChipScope Debug Cores** icon on the toolbar menu of the ChipScope view, or

- Select the **Implement ChipScope Debug Cores** command from the popup menu of the ChipScope view.

You will be prompted to save the design prior to the debug core being implemented, if there are unsaved changes.

The Xilinx CORE Generator™ tool invokes in batch mode for each black box debug core. This operation can take some time. A progress indicator shows that the operation is running. When the debug core implementation is complete, the debug core black boxes are resolved and you can access the generated instances.

Moving Processes to the Background

As the PlanAhead tool spawns the CORE Generator process to implement the debug core, the Implement Debug Core dialog box lets you put the process into the background, as shown in Figure 12-8, page 375.

When you put a process into the background, it releases the PlanAhead tool to perform certain other functions, like viewing reports, or opening design files, while it completes the background task. You can make use of this time to review previous runs for instance, or examine reports. However, the Tcl Console is blocked, and you will not be able to use Tcl commands, or perform tasks that require Tcl commands, such as switching to another open design.
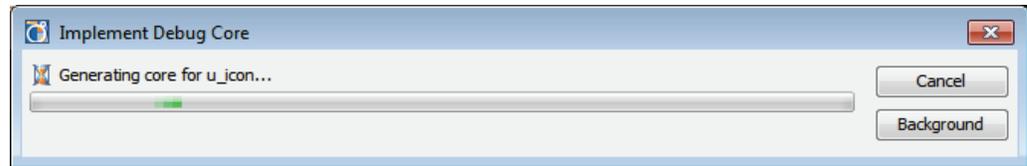
Figure 12-8:   **Implement Debug Core - Background Process**

Exporting Net Connections CDC File for ChipScope Analyzer Tool

A ChipScope Analyzer CDC file is generated automatically when design implementation is complete. Also, you can export a CDC file manually using **Export Debug Net Names** in the ChipScope view. You can import this CDC file into the ChipScope Analyzer to automatically set up the net names on the ILA core data and trigger ports.

Implementing the Design with the Debug Cores

After ChipScope debug cores are created and connected, you can run the standard PlanAhead design implementation flow to create a bitstream for the device.

Start the Implementation flow by selecting the **Implement** in the Flow Navigator, or the Tools menu.

Launching ChipScope Pro Analyzer

When the ChipScope Pro Analyzer software is installed, you can launch it directly from the PlanAhead tool on any implemented design on which **Generate Bitstream** has been run.

To launch ChipScope Pro Analyzer, do one of the following:

* In the Flow Navigator, select **ChipScope Analyzer** from the Program and Debug menu.

* Use the **Tools > Analysis > ChipScope Analyzer** command from the main menu.

* In the ChipScope view, right-click and select **Launch ChipScope Analyzer** from the popup menu.

The PlanAhead tool passes the BIT bitstream and CDC netlist name files automatically to the ChipScope Pro Analyzer. For more information about ChipScope Pro Analyzer see the Xilinx website, http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/design_tools/ chipscope_pro.html. Or refer to the manuals cited in ChipScope Documentation in Appendix E.

Launching iMPACT

The iMPACT tool lets you perform device configuration and file generation.

* Device Configuration lets you directly configure Xilinx FPGAs and PROMs with the Xilinx cables (Parallel Cable IV, Platform Cable USB, or Platform Cable USB II).

* Operating in Boundary-Scan mode, iMPACT can configure or program Xilinx FPGAs, CPLDs, and PROMs.

* File generation enables you to create the following programming file types: System ACE™ interface files, CF, PROM, SVF, STAPL, and XSVF files.

iMPACT also lets you:

* Readback and verify design configuration data

- Debug configuration problems
- Execute SVF and XSVF files

You can launch the iMPACT software tool directly from the PlanAhead tool on any implemented design on which the Generate Bitstream command has been run. To invoke iMPACT, in the Flow Navigator, select **iMPACT**.

The BIT bitstream file is passed automatically to iMPACT when launched from the PlanAhead tool. For more information on using iMPACT, see the iMPACT Help.

Using Hierarchical Design Techniques

The PlanAhead™ tool supports hierarchical design.

Xilinx® strongly recommends that you decide whether or not to use hierarchical design *before* you begin your project.

- The decision to use hierarchical design requires some forethought about design partitioning and Register Transfer Logic (RTL) coding decisions.

- Your results could be less than optimal if you try to adapt a design to this methodology late in the design cycle in an attempt to close timing or reduce runtime.

For more information, see the following documents, cited in Appendix E, Additional Resources:

- *Hierarchical Design Methodology Guide (UG748)*
- *Partial Reconfiguration User Guide (UG702)*

Using Partitions

The hierarchical capabilities that are central to setting and managing hierarchical boundaries in the design are called *partitions*.

Defining partitions prevents the Synthesis and Implementation tools from optimizing logic across partition boundaries, making it possible to isolate the logic for reuse.

Effective partitioning relies on good logic design practices and knowledge of the design.

The PlanAhead tool supports the following partitioning capabilities:

- An incremental Xilinx® Synthesis Technology (XST) flow using RTL projects
- Importing a partition into a different hierarchy than in which a partition was created
- `AREA_GROUPS` within partitions
- Black Box support in Synthesis and Implementation
- Boundary optimization for constants and unconnected inputs and outputs on partition ports.

    Note: Nested partitions, which are available in the ISE® command line flow, are not support in PlanAhead.

After you have implemented designs with partitions, you can export the results for use in future runs. The partition definition and behavior is defined in an XML file called `xpartitions.pxml`. The ISE Design Suite tools search the run directory for that file and act accordingly. Partitions are defined as well as the specified partition "action" such as Implement or Import. See the Hierarchical design documentation for a description of the `xpartitions.pxml` file usage and syntax.

Setting Partitions

You can set partitions on any hierarchical design instance in a Design. The hierarchical design flows require that each partition be synthesized separately, resulting in individual netlists per partition. This ensures that partition netlists can be isolated and reused.

To set a partition:

1. In the Netlist view, select the module instances on which to set partitions.

2. Select the **Set Partition** command from the popup menu.

3. The icon associated with the instance in the Netlist view changes to highlight that the module is defined as a partition, and a Partition tab appears in the Instance Properties view, as shown in Figure 13-1.

The Partition tab of the Properties view reports the dates of promoted Partitions, and the location the partition was promoted to. For more information on the Partition Properties view, and the BoundaryOpt and ImportTag properties, refer to the *Hierarchical Design Methodology Guide (UG748)*.
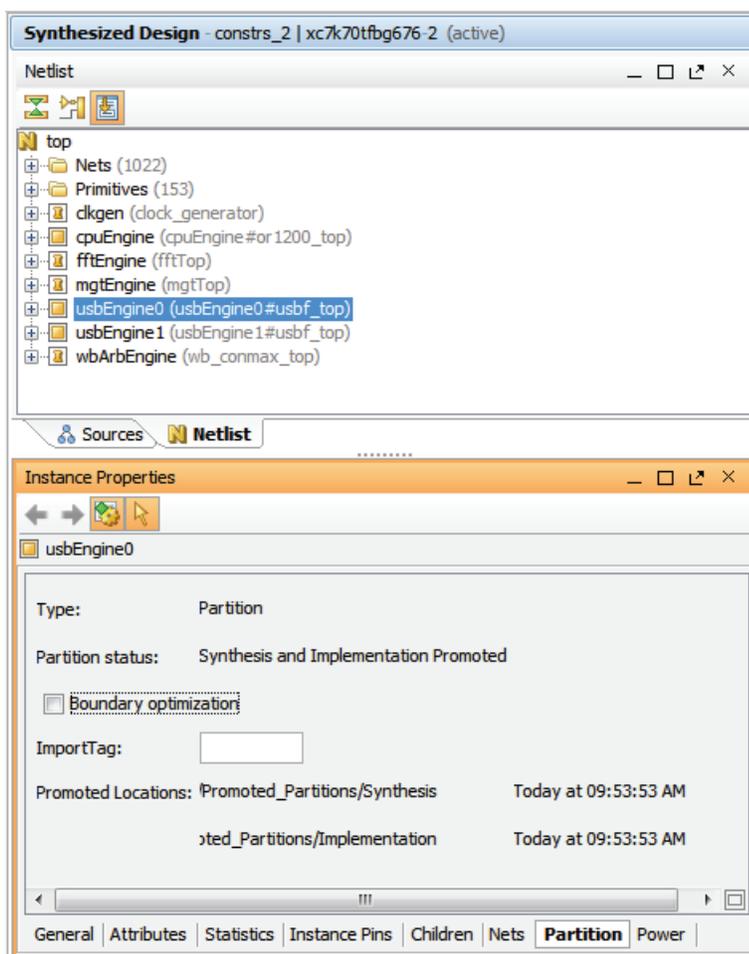
Figure 13-1: **Setting Partitions in the Netlist View**

After a partition is defined in a project, the top-level of the design automatically becomes a partition as well. You can treat this top partition like any other partition; implement or import it as needed during design iterations.

Configuring Synthesis and Implementation Runs with Partitions

Defining partitions in a design automatically enables the XST incremental flow. This incremental flow implements individual netlist files (NGC) for each partition, including the top partition, and results are then imported into the Synthesized Design. For more information on synthesis, see Chapter 6, Synthesizing the Design.

On subsequent design iterations, you can configure the action the PlanAhead tool takes for each partition during synthesis and implementation. Partitions can either be resynthesized and re-implemented if changed, or imported from a specified location if unchanged.

Prior to synthesis or implementation of a design containing partitions, you must specify how the PlanAhead tool will handle the partitions in the design.

1. Select **Specify Partitions** in the Flow Navigator, under Project Manager, as shown in Figure 13-2. This will open the Specify Partitions dialog box.

Figure 13-2: **Synthesis and Implementation Settings**

The Specify Partition dialog box, as shown in Figure 13-3, lets you specify the actions for each partition in the design for the active synthesis and implementation runs. If there are multiple synthesis and implementation runs defined in the project, you should specify the active run prior to using the **Specify Partitions** command.

You can import partition data from a specified location, or you can rebuild the partition through synthesis or implementation.

2. Set the partition **Action** as follows:

   • **Implement -** To synthesize and implement a partition for the first time, or re-implement a partition due to design changes.

   Note: The first time a design is implemented, all partitions must be set to **Implement**, because there are no promoted locations to import the implemented partitions.

   • **Import** - Import the partition data.

     - **Import From** - Specify the source location for the partition being imported. This is useful for specifying which results to import when there are multiple implementations of a specific partition.

     - **Preservation** - Indicates the level of partition data to preserve when importing the partition. Lets you import the **Synthesis** results for the partition, and re-implement the block; or preserve the **Placement** but not the routing; or preserve the **Routing** and timing of the implemented partition.

3. Select **OK** or **Cancel** to close the form.
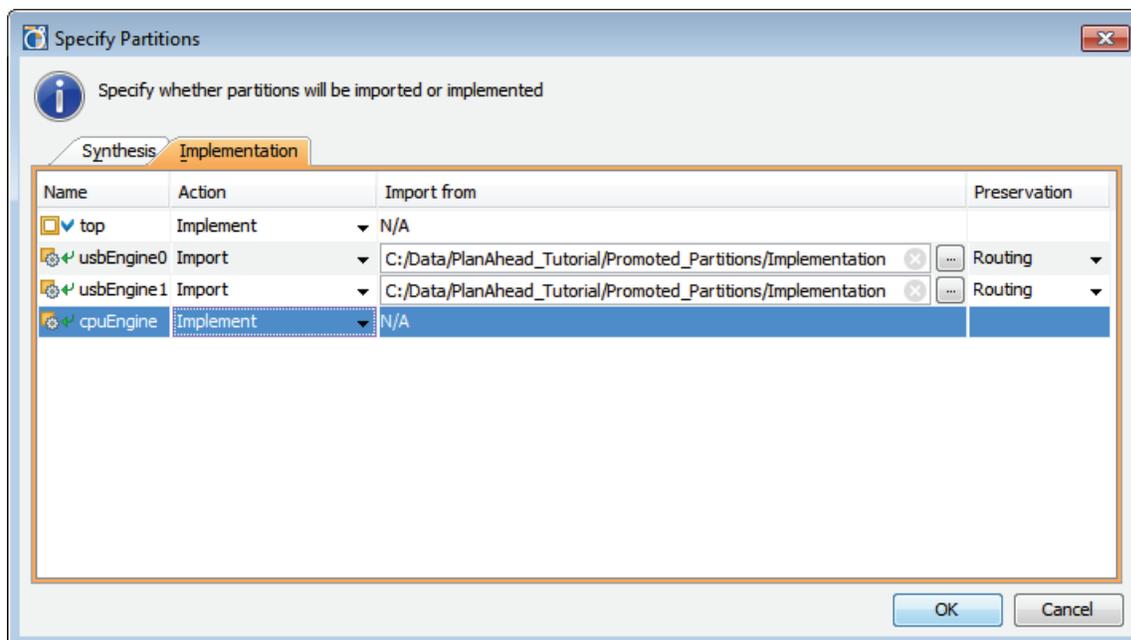
Figure 13-3: **Specify Partitions**

You can also specify partition settings from the Partitions tab of the Run Properties view for the synthesis or implementation runs, as shown in Figure 13-4.
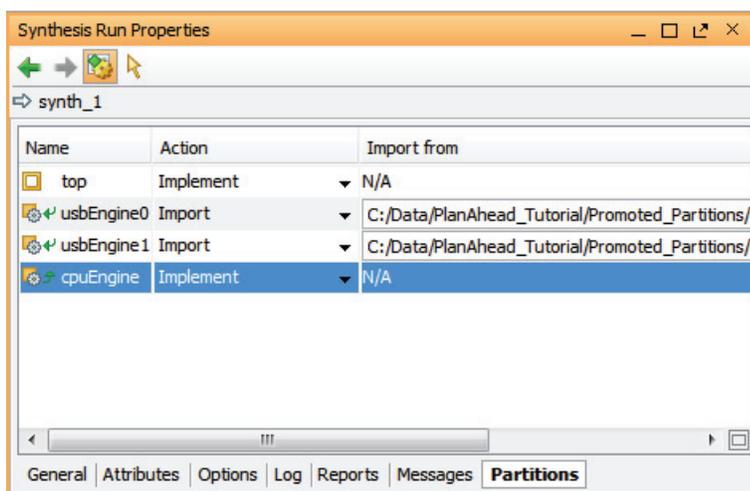
Figure 13-4: **Run Properties - Partitions**

When you launch synthesis or implementation for the design, you can view the results in the Reports view. The actions taken for the partitions in the run display in the report. Figure 13-5, page 381 shows the NGDBuild report for an implementation with partitions.

Figure 13-5: **Viewing Partition Actions in NGDBuild Report**

Promoting Partitions

When synthesis or implementation have completed to your satisfaction, you can copy the data to a directory or repository to preserve the results to import into future design iterations. Preserving the partition results is called *promoting* the partition. You can only promote partitions from successfully completed synthesis or implementation runs.

To promote partitions:

1. In the Flow Navigator, select **Promote Partitions**.

   You can also use the **Flow > Promote Partitions** command from the main menu.
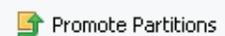
   The Promote Partition dialog box opens, as shown in Figure 13-6, page 382.
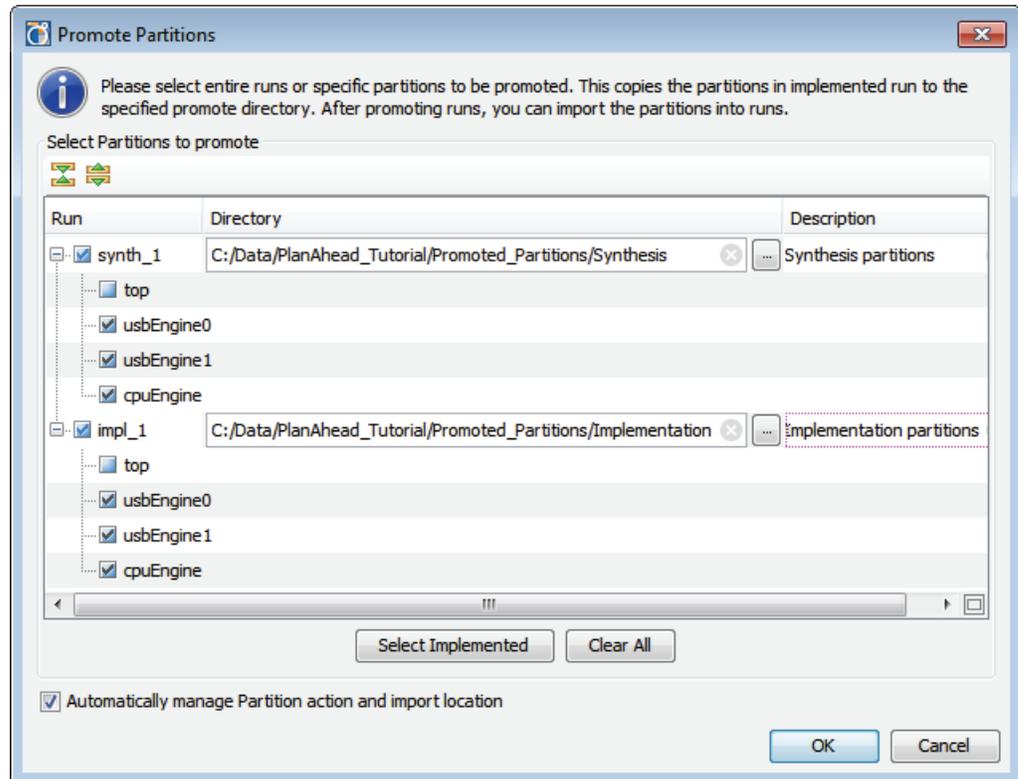
Figure 13-6: **Promoting Partitions**

2. Specify the directory to store the promoted partition.

   The directory or data repository is where the partition and all of its required files are stored. You will reference this location when importing partitions rather than re-implementing them in future design iterations.

   Note: You should promote all partitions to a single location. The PlanAhead tool prompts you to overwrite any previously promoted partitions in an effort to enforce this methodology. Refer to the *Hierarchical Design Methodology Guide, (UG748)* as cited in Appendix E, Additional Resources.

3. Select the partitions to promote.

   By default, all partitions in the design are selected to promote except for the top partition. Use the **Select Implemented** button to select all implemented partitions, including the top partition.

4. Optionally enter a **Description** for the partition.

5. Check the **Automatically manage Partition action and import location** to have the PlanAhead tool set the Action to Import for all promoted partitions in the design. After you promote the partition, the default action for that partition in the Specify Partitions dialog box is set to **Import** for subsequent synthesis or implementation runs. See Configuring Synthesis and Implementation Runs with Partitions, page 379.

   If this checkbox is disabled, you will need to manually set the Action for partitions in the design.

6. Click **OK**.

The Promoted Partitions view opens to report the results.

Using the Promoted Partitions View

The Promoted Partitions view displays each time you select **Promote Partitions**.

To open the Promoted Partitions view, select **Window > Promoted Partitions**. Figure 13-7 shows the Promoted Partitions view.
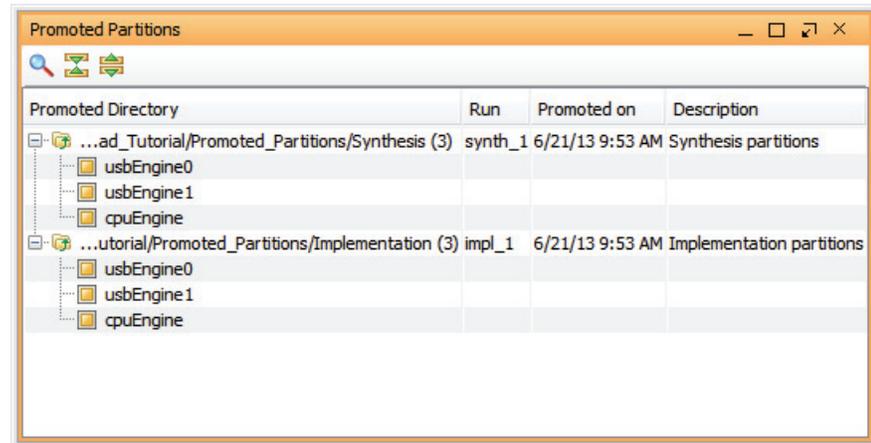
Figure 13-7: **Promoted Partitions View**

The Promoted Partitions view displays a tree table detailing the directory for the promoted partitions, the source run, the date and time of promotion, and description.

Selecting a partition opens the Promoted Run Properties view, where the description field can be edited if needed.

Deleting Promoted Partitions

You can remove promoted partitions in the Promoted Partitions view. The entire directory containing the promoted partition is also removed, including all run data for the partition.

To delete promoted partitions, select the promoted directory in the Promoted Partitions view and click the **Delete** popup menu command.

After you delete a partition, the default action for that partition in the Specify Partitions dialog box is set to **Implement** for subsequent synthesis or implementation runs.

Importing Partitions

After you have promoted partitions, you can import those partitions for subsequent runs.

- For Synthesis, importing partitions results in the imported NGC file being copied into the current results directory. This ensures the identical synthesis results are used for each iteration.

- For Implementation, importing partitions causes the placement and routing of the partition to be imported into the top-level design prior to implementing the rest of the design, ensuring identical results.

Updating Sources

Updating Sources

When changes are made to RTL sources, the RTL and Netlist views as well as any completed Synthesis or Implementation Runs, go out-of-date. You can open or reload the RTL and Netlist view to apply the logic changes.

As design modifications occur, the updated netlists are either picked up automatically, if you are using remote sources, or updated in the project to update the existing netlists. After source files are updated, you can open or reload the Synthesized Design to apply the logic changes. Refer to Managing Design Source Files, page 183 for more information.

Setting Partition Actions Based on Logic Updates

You must set the Partition action for each run as described in Configuring Synthesis and Implementation Runs with Partitions, page 379.

- Partitions with updated netlists MUST be set to **Implement** to ensure that the logic is re-implemented with the changes.

- Unchanged partitions can be set to **Import** to preserve the partition, or can be set to **Implement** to re-implement as needed.

Related Methodologies

Team Design

The PlanAhead tool provides an environment to create and manage multiple projects for independent and parallel development. This Team Design flow supports Xilinx Synthesis Technology (XST) incremental synthesis and black-box support throughout Implementation.

See *Increased Productivity Using Team Design, (WP388)* as cited in Appendix E, Additional Resources.

Design Preservation

The Design Preservation methodology, coupled with the partition features, lets you lock and preserve the placement and routing of a partition from one run and use it in subsequent runs. This incremental design approach produces more consistent results, reduces verification time, and reduces design closure time.

See *Leveraging Design Preservation for Predictable Results, (UG747)* as cited in Appendix E, Additional Resources.

Partial Reconfiguration

The PlanAhead tool provides an environment to configure, implement and manage Partial Reconfiguration projects. The PlanAhead tool uses the partitions and Partial Reconfiguration capabilities that are built into the ISE Design Suite implementation tools.

See *Overview of Partial Reconfiguration Flow, (UG743)* as cited in Appendix E, Additional Resources.

Tcl and Batch Scripting

This chapter is not a comprehensive reference to Tcl commands, but does provide references to Tcl resources, and describes the general capabilities of Tcl in the PlanAhead environment.

The Tool Command Language (Tcl) is the scripting language integrated in the PlanAhead™ application environment. Tcl is a standard language in the semiconductor industry for design constraints.

Tcl lets you perform interactive queries to design tools in addition to executing automated scripts. Tcl offers the ability to "ask" questions interactively of design databases, particularly around tool and design settings and state.

The following sections describe some of the basic capabilities of Tcl with PlanAhead. For specific information about each command, consult the online help for the individual command. The *Tcl Command Reference Guide (UG789)*, as cited in Appendix E, Additional Resources, lists and describes the PlanAhead Tcl commands.

Tcl Journal Files

When you invoke the PlanAhead tool, it writes the `PlanAhead.log` file to record the various command and operations performed during the design session. The PlanAhead tool also writes a journal file, called `PlanAhead.jou`, which is a journal of just the Tcl commands run during the session that can be used as a source to create new Tcl scripts.

Note: Backup versions of the journal file, named `planahead_<id>.backup.jou`, are written to save the details of prior runs whenever the PlanAhead tool is launched. The *<id>* is a unique identifier that allow the tool to create and store multiple backup versions of the log and journal files.

Refer to Appendix A, PlanAhead Input and Output Files for information regarding the location of these files.

Tcl Help

The Tcl `help` command provides information related to the supported Tcl commands.

- `help`: Returns a list of Tcl command categories.

  help

  Command categories are groups of commands performing a specific function, like File I/O for instance.

- `help -category <category>`: Returns a list of commands found in the specified category.

  help -category object

  This example returns the list of Tcl commands for handling objects.

- `help <pattern>`: Returns a list of commands that match the specified search pattern. This form can be used to quickly locate a specific command from a group of commands.

  `help get_*`

  This example returns the list of Tcl commands beginning with get_.

- `help <command>`: Provides detailed information related to the specified command.

  `help get_cells`

  This example returns specific information of the get_cells command.

- `help -args <command>`: Provides an abbreviated help text for the specified command, including the command syntax and a brief description of each argument.

  `help -args get_cells`

- `help -syntax <command>`: Reports the command syntax for the specified command.

  `help -syntax get_cells`

You can also specify help as an argument to the command being used. This is useful when typing a command with arguments, and needing some help to recall the command syntax or arguments. In this case you can use either:

```
report_power -help syntax
report_power -help args
```

Tcl Console

The PlanAhead tool GUI environment contains an area that echoes the Tcl commands as operations are performed, and provides information, warnings, and error messages that result from tasks performed. The Tcl console is located along the bottom of the PlanAhead environment and fixed to the width of the GUI. Along the right side of the Tcl console, just to the right of the scrollbar is an area with color coded indicators for warnings and errors. Any warnings issued to the console are colored yellow and errors are red. This is a useful feature to scroll back through message history and navigate to view warnings and errors in the context of the command that was performed. Figure 14-1 shows the Tcl console within the environment:

```
Tcl Console                                                              _ □ ⤢ ×
current_design netlist_1
set_delay_model -interconnect estimated
report_timing -delay_type min_max -max_paths 10 -sort_by group -input_pins -name timing_1
INFO: [Timing 38-45] Going to infer timing constraints.
WARNING: [Constraints 18-467] The TNM/TNM_NET will not trace through clock modifying cells like MMCMs since
INFO: [Timing 38-33] Done inferring timing constraints.
INFO: [Timing 38-91] UpdateTimingParams: Speed grade: -2, Delay Type: min_max, Constraints type: UCF.
INFO: [Timing 38-191] Multithreading enabled for timing update using a maximum of 2 CPUs
INFO: [Timing 38-78] ReportTimingParams: -max_paths 10 -nworst 1 -delay_type min_max -sort_by group.
report_timing: Time (s): elapsed = 00:00:20 . Memory (MB): peak = 1490.855 ; gain = 118.824

Type a Tcl command here
Tcl Console   Messages   Log   Reports   Design Runs   Timing
```

Figure 14-1: **Tcl Console**

Mapping Keystrokes in the Tcl Console[1]

The Tcl Console supports native Tcl, so key bindings correspond to the standard tclsh, which uses editline, with keystroke mappings defined in an `.editrc` file located in your home directory ($HOME). The `.editrc` file defines various settings to be used by the editline library.

Note: No `.editrc` file is provided in the software installation by default. You will need to create one to customize the key bindings for the Tcl Console.

You can customize keystroke bindings to specific editor commands by editing the `.editrc` file in your home directory. The format of the `.editrc` file is as follows:

```
bind <key> <command>
```

Where:

- `bind` - The keyword to bind the key to the command.
- *<key>* - The key or key combination to bind.
- *<command>* - The editor command or macro to bind to the key.

For example, the following `.editrc` file maps the tilda key (~) to the Emacs delete previous character command, and the Ctrl-R key combination (^r) to the Emacs incremental reverse search command:

```
bind "~" em-delete-prev-char
bind "^r" em-inc-search-prev
```

Refer to the following URL for more information on the `.editrc` file:

http://www.freebsd.org/

Invoking the PlanAhead Software

The PlanAhead tool provides three primary modes of operation:

- The GUI mode (default)
- Invocation of the PlanAhead tool executable by a Tcl command-line option (Batch mode)
- Tcl shell mode

The following subsections describe Batch and Tcl shell modes.

Batch Mode

Batch mode executes a script and then shuts down the tool. To invoke the PlanAhead tool in batch mode:

```
planAhead -mode batch -source script_name.tcl
```

Tcl Shell Mode

Tcl mode invokes a shell similar to windows command shell or a linux shell. This is an interactive shell session that does not invoke the GUI.

---

1. The `.editrc` file is only available to map keystrokes on the Linux operating system.

Optionally, you can pass a script with the `-source` switch, which executes the script and then passes the control to the interactive shell where you can manually enter Tcl commands. To launch PlanAhead in Tcl shell mode:

```
planAhead -mode tcl -source script_name.tcl
```

Tcl Init File

When the PlanAhead tool is launched it looks for the existence of a Tcl initialization script in two different locations:

1. `<installdir>/planAhead/scripts/init.tcl`

2. `<userdir>/Xilinx/PlanAhead/init.tcl`

Where:

- `<installdir>` is the installation directory where the PlanAhead tool is installed, and

- `<userdir>` is your home directory.

  - For Windows: `%APPDATA%/Xilinx/PlanAhead/init.tcl`

  - For Linux: `$HOME/.Xilinx/PlanAhead/init.tcl`

If the `init.tcl` script exists, in one or both of those locations, the PlanAhead tool sources this file; first from the installation directory and second from your home directory.

The `init.tcl` file in the installation directory allows a company or design group to support a common initialization script for all users. Anyone invoking the PlanAhead tool from that software installation sources that `init.tcl` script.

The `init.tcl` in the home directory allows each user to specify additional commands, or to overwrite commands from the software installation to meet their specific design requirements.

The `init.tcl` script is a standard Tcl command file that can contain any legal Tcl command supported by the PlanAhead tool. You can also source another Tcl script file from within the `init.tcl` file by adding the following statement:

```
source <path_to_file>/<file_name>.tcl
```

General Tcl Syntax Guidelines

Tcl uses the Linux file separator (/) convention regardless of the OS on which you are operating.

The following subsections describe the general syntax guidelines for using Tcl in the PlanAhead tool.

Sourcing a Tcl Script

A Tcl script can be sourced from either one of the command-line options or from the GUI. Within the PlanAhead GUI you can source a Tcl script from **Tools > Run Tcl Script**.

You can source a Tcl script from a command-line option:

```
source file_name
```

When you invoke a Tcl script from the GUI, a progress bar is displayed and all operations in the GUI are blocked until the scripts completes.

There is no way to interrupt script execution during run time; consequently, standard OS methods of killing a process must be used to force interruption of the tool. If the process is killed, you lose any work done since your last save.

Typing **help source** in the Tcl console will provide additional information regarding the `source` command.

Using Tcl eval

When executing Tcl commands, you can use variable substitution to replace some of the command line arguments accepted or required by the Tcl command. However, you must use the Tcl eval command to evaluate the command line with the Tcl variable as part of the command.

For instance, the `help` command can take the `-category` argument, with one of a number of command categories as options.

```
help -category ipflow
```

You can define a variable to hold the command category:

```
set cat "ipflow"
```

You can then evaluate the variable in the context of the Tcl command:

```
eval help -category $cat
```

or,

```
set cat "-category ipflow"
eval help $cat
```

You can also use braces {} in place of quotation marks "" to achieve the same result.

```
set runblocksOptDesignOpts { -sweep -retarget -propconst -remap }
eval opt_design $runblocksOptDesignOpts
```

Typing **help eval** in the Tcl console will provide additional information regarding the `eval` command.

General Syntax Structure

The general structure of a PlanAhead tool Tcl commands is:

```
command [optional_parameters] required_parameters
```

Command syntax is of the verb-noun and verb-adjective-noun structure separated by the underscore ("_") character.

Commands are grouped together with common prefixes when they are related.

- Commands that query things are generally prefixed with `get_`.
- Commands that set a value or a parameter are prefixed with `set_`.
- Commands that generate reports are prefixed with `report_`.

The commands are exposed in the global namespace. Commands are "flattened," meaning there are no "sub-commands" for a command.

Example Syntax

The following is an example of the return format on the **help get_cells** command:

```
help get_cells
get_cells

Description:
Get a list of cells in the current design
```

```
Syntax:
get_cells [-hsc <arg>] [-hierarchical] [-regexp] [-nocase]
[-filter <arg>] [-of_objects <args>] [-match_style <arg>]
[-quiet] [-verbose] [<patterns>]

Returns:
list of cell objects

Usage:
```

Name	Optional	Default	Description
-hsc	yes	/	Hierarchy separator
-hierarchical	yes	false	Search level-by-level in current instance
-regexp	yes	false	Patterns are full regular expressions
-nocase	yes	false	Perform case-insensitive matching
-filter	yes		Filter list with expression
-of_objects	yes		Get cells of these pins or nets
-match_style	yes	sdc	Style of pattern matching
-quiet	yes	false	Ignore command errors
-verbose	yes		Suspend message limits during command execution
patterns	yes	*	Match cell names against patterns

Unknown Commands

Tcl contains a list of built-in commands that are generally supported by the language, PlanAhead-specific commands which are exposed to the Tcl interpreter, and user-defined procedures.

Commands that do not match any of these known commands are sent to the OS for execution in the shell from the `exec` command. This lets users execute shell commands that might be OS-specific. If there is no shell command, then an error message is issued to indicate that no command was found.

Return Codes

Some Tcl commands are expected to provide a return value, such as a list of objects on which to operate. Other commands perform an action but do not necessarily return a value that can be used directly by the user. Some tools that integrate Tcl interfaces return a 0 or a 1 to indicate success or error conditions when the command is run.

To properly handle errors in Tcl commands or scripts, you should use the Tcl built-in command `catch`. Generally, the `catch` command and the presence of numbered info, warning, or error messages should be relied upon to assess issues in Tcl scripted flows.

All PlanAhead application commands return either `TCL_OK` or `TCL_ERROR` upon completion., In addition, the PlanAhead tool sets the global variable `$errorinfo` through standard Tcl mechanisms.

To take advantage of the `$errorinfo` variable, use the following line to report the variable after an error occurs in the Tcl console:

```
puts $errorinfo
```

This reports specific information to the standard display about the error. For example, the following code example shows a Tcl script (`procs.tcl`) being sourced, and a user-defined procedure (`loads`) being run. There are a few transcript messages, and then an error is encountered at line 5.

```
Line 1: PlanAhead % source procs.tcl
Line 2: PlanAhead% loads
Line 3: Found 180 driving FFs
Line 4: Processing pin a_reg_reg[1]/Q...
Line 5: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of_objects'.
Line 6: PlanAhead% puts $errorInfo
Line 7: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of_objects'. While executing
"get_ports -of objects $pin" (procedure "my_report" line 6) invoked from within procs.tcl
```

You can add the `puts $errorInfo` into `catch` clauses in your Tcl script files to report the details of an error when it is caught, or use the command interactively in the Tcl console immediately after an error is encountered to get the specific details of the error.

In the example code above, typing the **`puts $errorInfo`** command in line 6, reports detailed information about the command and its failure in line 7.

First Class Tcl Objects and Relationships

The Tcl commands in the PlanAhead tool provide direct access to the object models for netlist, devices, and projects. These objects are *first-class* which means they are more than just a string representation, and they can be operated on and queried. There are a few exceptions to this rule, but generally "things" can be queried as objects, and these objects have properties that can be queried and they have relationships that allow you to get to other objects.

Object Types and Definitions

There are many object types in the PlanAhead tool; this chapter provides definitions and explanations of the basic types. The most basic and important object types are associated with entities in a design netlist, and these types are listed in the following subsections:

Cell

A cell is an instance of either a library primitive or a hierarchical module/entity inside a netlist. Examples of cells include flip-flops, LUTs, I/O buffers, RAM and DSPs, as well as hierarchical instances which are wrappers for other lists of cells.

Pin

A pin is a point of logical connectivity on a cell. A pin allows the internal logic of a cell to be abstracted away and simplified for easier use, and can either be on hierarchical or leaf cells. Examples of pins include clock, data, reset, and output pins of a flop.

Port

A port is a point of logical connectivity on the interface of the top-level module or entity of the netlist. Ports are normally attached to I/O pads and connect externally to the FPGA device.

Net

A net is a collection of hierarchical pins, leaf pins and ports. In case of a hierarchical design, a net can have several segments, each at a different level of the hierarchy.

Clock

A clock is a periodic signal that propagates to sequential logic within a design. Clocks can be primary, i.e. created on an input port, or generated by Clock Modifying Block primitives such as DCM, PLL or MMCM. A clock is the rough equivalent to a TIMESPEC PERIOD constraint in UCF and forms the basis of static timing analysis algorithms.

Querying Objects

All first class objects can be queried by a get_ Tcl command that generally has the following syntax:

get_object_type pattern

Where pattern is a search pattern, which includes if applicable a hierarchy separator to get a fully qualified name. Objects are generally queried by a string pattern match applied at each level of the hierarchy, and the search pattern also supports wildcard style search patterns to make it easier to find objects, for example:

get_cells */inst_1

This command searches for a cell named inst_1 within the first level of hierarchy under the top-level of hierarchy. To recursively search for a pattern at every level of hierarchy, use the following syntax:

get_cells -hierarchical inst_1

This command searches every level of hierarchy for any instances that match inst_1.

For complete coverage of syntax, see the specific online help for the individual command:

```
help get_cells
get_cells -help
```

Object Properties

Objects have properties that can be queried. Property names are unique for any given object type. To query a specific property for an object, the following command is provided:

get_property <property_name> <object>

An example would be the ref_name property on cell objects, which tells you what UniSim component a given instance is mapped to:

get_property ref_name [get_cell inst_1]

To discover all of the available properties for a given object type, use the **report_property** command:

report_property [get_cells inst_1]

Table 14-1 shows the properties returned for a specific object.

Some properties are read-only and some are user-definable. Properties that map to attributes that can be annotated in UCF or in HDL are generally user-definable through Tcl with the `set_property` command:

```
set_property loc OLOGIC_X1Y27 [get_cell inst_1]
```

Table 14-1: **Reported Properties for Specified Object**

Key	Value	Type
bel	OLOGICE1.OUTFF	string
class	cell	string
iob	TRUE	string
is_blackbox	0	bool
is_partition	0	bool
is_primitive	1	bool
is_reconfigurable	0	bool
is_sequential	1	bool
ref_name	FD	string
loc	OLOGIC_X1Y27	string
name	error	string
primitive_group	FD_LD	string
primitive_subgroup	flop	string

Filtering Based on Properties

The object query `get_*` commands have a common option to filter the query based on any property value attached to the object. This is a powerful capability for the object query commands. For example, to query all cells of primitive type FD do the following:

```
get_cells * -hierarchical -filter {ref_name == FD}
```

To do more elaborate string filtering, utilize the =~ operator to do string pattern matching. For example, to query all flip-flop types in the design, do the following:

```
get_cells * -hierarchical -filter {ref_name =~ FD*}
```

Multiple filter properties can be combined with other property filters with logical OR (‖) and AND (&&) operators to make very powerful searches. To query every cell in the design that if of any flop type and has a placed location constraint:

```
get_cells * -hierarchical -filter {ref_name =~ FD* && loc != ""}
```

Note: In the examples, the filter option value was wrapped with curly braces {} instead of double quotes. This is normal Tcl syntax that prevents command substitution by the interpreter and allows users to pass the empty string (`""`) to the `loc` property.

Large Lists of Objects

Commands that return more than one object generally return a container that looks and behaves like a native Tcl list. This is a feature of the PlanAhead tool that allows dramatic optimization of

handling large lists of Tcl objects without the need for special iteration commands like the `foreach_in_collection` which is handled with the Tcl built-in `foreach`.

There are a few nuances with respect to large lists, particularly in the log files and the GUI Tcl console. Typically, when you set a Tcl variable to the result of a `get_*` command, the entire list is echoed to the console and to the log file. For large lists, this is truncated when printed to the console and log to prevent memory overloading of the buffers in the tool.

What is echoed is the list printed to the log and console is truncated and the last element appears to be "…" in the log and console, however the actual list in the variable assignment is still correct and the last element is not an error. An example of this is querying a single cell versus every cell in the design, which can be large:

```
%get_cells inst_1
inst_1
%get_cells * -hierarchical
XST_VCC XST_GND error readIngressFifo wbDataForInputReg fifoSelect_0
fifoSelect_1 fifoSelect_2 fifoSelect_3 ...
%set x [get_cells * -hierarchical]
XST_VCC XST_GND error readIngressFifo wbDataForInputReg fifoSelect_0
fifoSelect_1 fifoSelect_2 fifoSelect_3 ...
%lindex $x end
bftClk_BUFGP/bufg
%llength $x
4454
```

In this example, all four thousand cells were not printed to the console and the list was truncated with a "…" but the actual last element of the list is still correct in the Tcl variable.

Object Relationships

Related objects can be queried using the `-of_objects` option to the relevant `get_*` command. For example, to get a list of pins connected to a cell object, do the following:

`get_pins -of_objects [get_cells inst_1]`

Figure 14-2, page 395 is a diagram of the object types in the PlanAhead tool and their relationship, where an arrow from one object to another object indicates that you can use the `-of_objects` option to the `get_*` command to traverse logical connectivity and get Tcl references to any connected object.
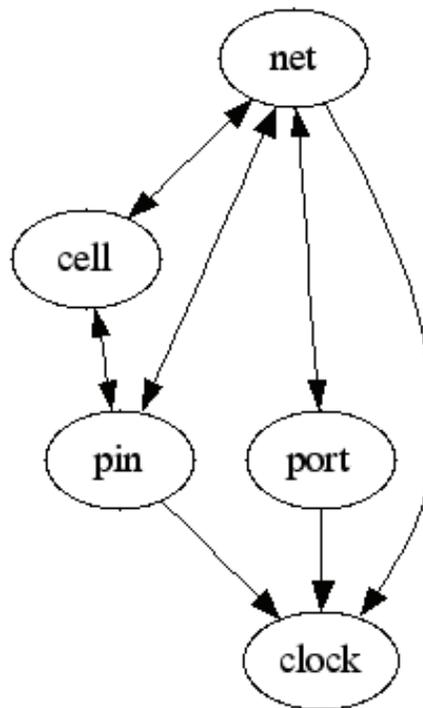
Figure 14-2: **PlanAhead Object Relationships**

Errors, Warnings, Critical Warnings, and Info Messages

Messages that result from individual commands appear in the log file as well as in the GUI console if it is active. These messages are generally numbered to identify specific issues and are prefixed in the log file with "INFO", "WARNING", "CRITICAL_WARNING", "ERROR" followed by a subsystem identifier and a unique number.

The following is an example of an INFO message that appears after reading the timing library.

INFO: [HD-LIB 1] Done reading timing library

These messages make it easier to search for specific issues in the log file to help to understand the context of operations during command execution.

Generally, when an error occurs in a Tcl command sourced from a Tcl script, further execution of subsequent commands is halted. This is to prevent unrecoverable error conditions. There are Tcl built-ins that allow users to intercept these error conditions, and to choose to continue. Consult any Tcl reference for the catch command for a description of how to handle errors using general Tcl mechanisms.

Tcl References

The following subsections provide recommended Tcl references.

Tcl Developer Xchange

Tcl reference material is available on the Internet. Xilinx recommends the Tcl Developer Xchange, which maintains the open source code base for Tcl, and is located at:

http://www.tcl.tk

An introductory tutorial is available at:

http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html

About SDC

Synopsys Design Constraints (SDC) is an accepted industry standard for communicating design intent to tools, particularly for timing analysis. A reference copy of the SDC specification is available from Synopsys by registering for the TAP-in program at:

http://www.synopsys.com/Community/Interoperability/Pages/TapinSDC.aspx

Available Tcl Documents

Tcl reference documents are available in book stores or online retailers.

Using PlanAhead With Project Navigator

The PlanAhead™ tool is integrated with the ISE® design tools, letting you perform specific tasks of the FPGA design flow in the PlanAhead tool. When you invoke the PlanAhead tool from Project Navigator in the ISE Design Suite, the features of the PlanAhead tool are restricted to I/O pin planning, floorplanning, and timing analysis.

The four processes in the Project Navigator Processes pane from which you can invoke the PlanAhead tool are:

- Pre-Synthesis
  - I/O Pin Planning
- Post-Synthesis
  - I/O Pin Planning
  - Floorplan Area/IO/Logic
- Post-Implementation
  - Analyze Timing/Floorplan Design

The Project Navigator creates and manages the PlanAhead tool project data automatically. The data passed between the two tools and the view layout presented in the PlanAhead tool depends upon which step you invoke. Refer to the PlanAhead Processes within Project Navigator, page 397, for more information on the mechanics of the integration including data passing and processes.

The PlanAhead tool has two default view layouts for the many design tasks:

- The I/O pin planning environment, called the I/O Planning view layout, which contains views pertinent to I/O pin planning and assignment.
- The Design Analysis environment, called the Default Layout view layout, which contains views pertinent to design analysis and floorplanning.

For more information about using the PlanAhead tool viewing environment, see Chapter 4, Using the Viewing Environment. For more information about configuring and loading view layouts, refer to Using View Layouts in Chapter 4.

PlanAhead Processes within Project Navigator

Project Navigator and the PlanAhead tool are two independent environments operating under a separate system process. The two processes are integrated to ensure that data is passed effectively between the two tools. Changes to design data in one tool are not recognized automatically in the other in real time, and you should not to edit logic or constraints simultaneously in both tools. Invoke PlanAhead for the intended purpose and close it before updating the Project Navigator design data. The Project Navigator process steps are synchronized to recognize edits to the User Constraint File (UCF) made in the PlanAhead tool after saving the data. The following sections describe the steps involved and the data transactions that enable the integration.

Passing Logic and Constraints

The PlanAhead tool in ISE Integration mode enables only physical constraint modification for I/O pins, logic LOC, and AREA_GROUP constraints. Logic connectivity in the form of Register Transfer Level (RTL) sources or synthesized netlists are passed into the PlanAhead tool for analysis purposes only, and not passed back to Project Navigator. The PlanAhead tool features that enable logic or timing constraint modification are disabled in ISE Integration mode.

You must perform logic modifications in Project Navigator, an external RTL, or in synthesis tools. The PlanAhead tool passes only the UCF constraint files to Project Navigator.

The PlanAhead tool maintains the original content and format of the UCF files, including comments, incomplete constraints, and so forth. The legality of constraints in the design is checked upon opening or closing the PlanAhead tool, and critical warnings are issued to the Tcl Console. Running DRC can provide more detailed information.

When you invoke the PlanAhead tool, the UCF source files in the Project Navigator project are passed to the PlanAhead tool project where you can add or modify physical constraints.

When you use the Save Constraints command in the PlanAhead tool, the software writes the modified UCF files back to the original Project Navigator source location. If you make constraint changes in the PlanAhead tool and select the Exit command, you are prompted to save changes back to the Project Navigator project before the tool closes.

If the PlanAhead tool is invoked and no UCF exists in the Project Navigator project, you are prompted to create one. This empty UCF is then passed to the PlanAhead tool.

The PlanAhead tool supports Project Navigator projects with more than one UCF source file. Before the PlanAhead tool is invoked, a dialog box prompts you to select one of the UCF files.

- New constraints defined in the PlanAhead tool are written to the selected UCF.

- Any physical constraints that exist in a non-chosen UCF remain in that file, even if the value of that constraint is modified.

The Project Navigator design flow does not pass core-level NCF files to the PlanAhead tool. To use or view any physical constraints in these files, you must merge them manually into a top-level UCF prior to invoking the PlanAhead tool.

Project Navigator creates a temporary PlanAhead design project in the ISE project directory, and removes and replaces this project every time you invoke the PlanAhead tool from Project Navigator.

I/O Pin Planning (Pre-Synthesis)

You can elect to perform early I/O pin planning prior to having a synthesized netlist either by using the stand-alone PlanAhead application or by selecting this process step in Project Navigator.

Note: At this stage of the design process, logic Synthesis has not been run. The tool has no concept of clock ports, clock related logic, differential pairs, or GTs. You must ensure these types of ports are placed appropriately to avoid Implementation errors. Whenever possible, you should perform the I/O pin planning after logic Synthesis. The presence of a netlist ensures that the clocks, clock logic, differential pairs, GTs, and so forth are recognized and considered automatically during pin assignment in PlanAhead. Also, there are many Design Rule Checks (DRCs) that are performed based on logic connectivity and clocks to ensure a legal placement prior to Implementation.

To perform I/O pin planning in Project Navigator prior to running Synthesis:

- In the Processes pane, expand **User Constraints** and double-click **IO Pin Planning (PlanAhead) - Pre-Synthesis**, as shown in Figure 15-1, page 399.

- Select the **Tools > PlanAhead > I/O Pin Planning (PlanAhead) - Pre-Synthesis** command.

Figure 15-1: **Project Navigator Processes Pane**

When you invoke the PlanAhead tool, Project Navigator passes all of the RTL source files, the top module name and the UCF(s) to PlanAhead. The PlanAhead tool opens with a display of the default I/O Planning view layout. The PlanAhead tool performs an RTL elaboration to extract and display the top-level I/O ports in the I/O Ports view.

When you save or close the PlanAhead project it updates the original Project Navigator source UCF(s). This resets the Project Navigator design process state, if appropriate.

Refer to Passing Logic and Constraints, page 398 for more information about the integration mechanics and process.

Refer to Chapter 8, I/O Pin Planning, for more information about using the I/O Planning view layout.

I/O Pin Planning (Post-Synthesis)

Note: Whenever possible, I/O pin planning should be performed after logic synthesis. The presence of a netlist ensures that the clocks, clock logic, differential pairs, GTs, and so forth are recognized and considered automatically during pin assignment in PlanAhead. There are also many Design Rule Checks (DRCs) that are performed based on logic connectivity and clocks to ensure a legal placement prior to Implementation.

To perform I/O pin planning in Project Navigator after running logic Synthesis:

- In the Processes pane, expand **User Constraints** and double-click **I/O Planning (PlanAhead) - Post-Synthesis**.

- Select the **Tools > PlanAhead > I/O Pin Planning (PlanAhead) - Post-Synthesis** command.

When you invoke the PlanAhead tool, Project Navigator passes the synthesized NGC or EDIF format netlist and the UCF(s) to the project. The PlanAhead tool opens with a display of the default I/O Planning view layout, and the I/O ports display in the I/O Ports view.

When you save or close the PlanAhead tool project, it updates the original Project Navigator source UCF(s), and this resets the Project Navigator design process state, if appropriate.

Refer to Passing Logic and Constraints, page 398 for more information about the integration mechanics and process. Refer to Chapter 8, I/O Pin Planning, for more information about using the I/O pin planning environment.

Floorplan Area/IO/Logic (Post Synthesis)

You can use the PlanAhead tool design analysis and floorplanning environment prior to or after Implementation. To analyze the design or to perform floorplanning from Project Navigator after running logic Synthesis and prior to Implementation:

- In the Processes pane, expand **User Constraints**, and select **Floorplan Area/IO/Logic (PlanAhead)**.

- Select the **Tools > PlanAhead > Floorplan Area/IO/Logic (PlanAhead)** command.

When PlanAhead is invoked, Project Navigator passes the synthesized NGC or EDIF format netlist and the UCF(s) to the PlanAhead tool. It opens with the default design analysis and floorplanning environment displaying.

Note: In Project Navigator, set the Translate process property Macro Search Path (-sd) to the appropriate directory if lower-level NGC format core files are used in the design and are not added as sources.

When you save or close the PlanAhead tool project, it updates the original Project Navigator source UCF(s), and resets the Project Navigator design process state, if appropriate.

Refer to Passing Logic and Constraints, page 398 for more information about the integration mechanics and process. Refer to Chapter 5, Elaborated RTL Design, for more information about using the PlanAhead tool environment prior to Implementation. For more information about using it after Implementation, see Chapter 11, Analyzing Implementation Results and Chapter 10, Floorplanning the Design

Analyze Timing/Floorplan Design (Post Implementation)

You can use the PlanAhead tool design analysis and floorplanning environment after Implementation. When you analyze the design after Implementation you can view the placement and timing results to catch potential design issues.

Often, physical `LOC` or `AREA_GROUP` floorplanning constraints can help drive the Implementation tools toward better and more consistent results, and reduce Implementation runtimes.

To analyze the design or to perform floorplanning from Project Navigator after Implementation:

- In the Process pane, expand **Implement Design**, expand **Place & Route**, and double-click **Analyze Timing/Floorplan Design (PlanAhead).**

- Select **Tools > PlanAhead > Analyze Timing / Floorplan Design (PlanAhead).**

Project Navigator passes the following files to the PlanAhead tool when it opens:

- Synthesized NGC or EDIF format netlist

- UCF (s)

- ISE placement data

- Timing results

- Block RAM Memory Map (BMM) files

The PlanAhead tool is invoked with the default design analysis and floorplanning environment displayed. For the PlanAhead tool to extract the ISE placement data, it must convert the Native Circuit Description (NCD) file from ISE to a Xilinx® Definition List (XDL) format file.

A progress bar displays in the PlanAhead tool while this command is running. To expedite re-invoking the PlanAhead tool, the interface first checks for the existence of the XDL file and does not regenerate the file if it is still current.

When you select **Tools > PlanAhead > Analyze Timing / Floorplan Design (PlanAhead)** with the Implementation process out-of-date, you are prompted to either re- implement the design, or launch the PlanAhead tool on the existing result data without rerunning the Implementation tools.

When you save or exit, the PlanAhead tool updates the original Project Navigator source UCF(s), and this resets the Project Navigator design process state also, if appropriate.

XILINX

Appendix A

PlanAhead Input and Output Files

Input Files

While reading the input files, the PlanAhead™ application writes out any information, errors, warnings, critical warnings, and messages into the `planAhead.log` file. These messages display in the PlanAhead tool Tcl Console also.

The PlanAhead tool lets you specify the location of files used as input.

Note: Backup versions of the log file, named `planahead_<id>.backup.log`, are written to save the details of prior runs whenever the PlanAhead tool is launched. The *<id>* is a unique identifier that allow the tool to create and store multiple backup versions of the log and journal files. Backups of the journal file are also saved in the same manner.

Table A-1 lists the input file names and descriptions.

Table A-1: **Input Files**

File Name	Description
Design Text Files (Verilog, VHDL)	You can import and elaborate Verilog and/or VHDL files to analyze the logic or modify the source. The original source files can be referenced and left in place or they can be copied into the project for portability. You can specify directories when importing RTL source files. All recognized files and file types contained in the directories are imported into the project.
I/O Port Lists (CSV)	You can import a Comma Separated Values (CSV) format file to populate the I/O Ports view within the I/O Planning view layout. This functionality is intended for use in an I/O pin planning project only. You can then assign these I/O ports to physical package pins to define the device pin configuration. CSV is a standard file format used by FPGA and board designers to exchange information about device pins and pinout. The CSV columns are listed in I/O Port Lists (CSV) File Format, page 404.
Module-Level Netlists and Cores (EDIF, NGC, NGO, BMM)	The PlanAhead tool can construct a design using multiple EDIF or NGC netlists supporting a Hierarchical Design methodology. When you select the top-level logic, lower-level modules are imported automatically. The advantage to this process is more flexibility when updating the design. The PlanAhead tool has an incremental netlist import capability that allows netlist updates at any level of the design hierarchy.
Top-Level Netlists (EDIF, NGC)	The PlanAhead tool supports importing EDIF or NGC netlists. The netlist should be synthesized for a Virtex®-4, Virtex-5, Virtex-6, Virtex-7, Kintex™-7, Artix™-7, Spartan®-3 or Spartan-6 device. The PlanAhead tool can construct the design using multiple netlists supporting a Hierarchical Design methodology. When you select the top-level logic, lower-level modules are imported automatically. Incremental netlist import capabilities allow netlist updates at any level of design hierarchy. In-process floorplanning constraints are maintained through iterations.

Table A-1: **Input Files** *(Cont'd)*

File Name	Description
Constraint Files (UCF / NCF / XCF)	The PlanAhead tool supports importing UCF, NCF, and XCF format files for timing and physical constraints. The PlanAhead tool can import multiple UCF files which allows for separation of physical constraints, I/Os, and timing constraints. NCF files are module-level constraints that are specific to blocks of design or IP cores. XCF files, or XST Constraint Files, can also be imported into projects. The PlanAhead tool supports all of the UCF constraints supported by Xilinx®. Refer to the *Constraints Guide, (UG625)* as cited in Appendix E, Additional Resources for more information about UCF constraints and the supported syntax.
ISE Placement Results (NCD / XDL)	The PlanAhead tool can import ISE placement results using XDL format data. XDL data is created automatically when Implementation runs are launched. After ISE commands complete, you can create an XDL format file from the `placed_design_name`.ncd file. You can create XDL files and import placement for individual blocks or for the entire design. The PlanAhead tool runs the XDL command automatically when you select a `placed_design_name`.ncd file in the Import Placement dialog box.
TRCE Timing Results (TWX/ TWR)	The PlanAhead tool can import the timing report generated by the Xilinx `TRCE` command; including TWX and TWR files. After the files are imported, signal tracing and selection is available through the Timing Results View. If both files exist, TWX is the preferred format to use to import timing results.

I/O Port Lists (CSV) File Format

You can import a Comma Separated Values (CSV) format file to populate the I/O Ports view within the I/O Planning views. This functionality is available in an Empty I/O pin planning project only.

You can then assign these I/O ports to physical package pins to define the device pin configuration. CSV is a standard file format used by FPGA and board designers to exchange information about device pins and pinout.

You can also export a CSV file from an Elaborated Design, a Synthesized Design, or an Implemented Design by using the **File > Export > Export I/O Ports** command.

The CSV columns are:

- **I/O Bank**: The I/O Bank in which the pin is located. The software fills in this field for all pins in the device. Values are a number or blank. This is not required in the input CSV file.

- **Pin Number**: The name (or location) of the package pin. The software writes this out for all pins in the device. This is not required in the input file. If used for input, it is used to define placement. Values are legal pins in the device.

- **IOB Alias**: An alternate part name for the package pin. This field is specified by the software, and is unused if specified in the input CSV file.

- **Site Type**: The pin name from the device data sheet. This field is specified by the software, and is unused if specified in the input CSV file.

- **Min/Max Trace Delay (ps)**: The distance between the pad site of the die and the ball on the package, in picoseconds. This is specified by the tool to help the board engineer match trace delays. The Trace Delay fields are in the output file only. They are not expected in the input file.

- **Trace Length (um)**: Specifies the length of the internal trace between the package pin and the die pad.

- **Prohibit**: Certain sites can be prohibited for many reasons to prevent user I/O from being added to the site. Prohibits ease board layout issues, reduce cross-talk between signals, and ensure that a pinout works between multiple FPGAs in the same package. In the UCF this is represented by a `CONFIG PROHIBIT` constraint. Values are `TRUE` or a blank field; leave this field blank when the Pin Number is left blank.

- **Interface**: An optional user-specified grouping for an arbitrary set of user I/O. As an example, this field provides a means to specify a relationship for the data, address, and enable signals for a memory interface. Values are a text string or blank.

- **Signal Name**: The name of the User I/O in the FPGA design. Values are a string or blank for an unassigned Package Pin.

- **Direction**: The direction of the signal. Values are IN, OUT, INOUT, or blank when a user I/O is not assigned to the site.

- **DiffPair Type**: Instructs the software about which pin is the N side of a differential pair, and which pin is the P side. This is used for differential signals only. The software uses this column instead of a naming convention to determine which pin is the N side of the pair, and which pin is the P side.

  Values are P, N, or blank when a user I/O is not assigned to the site.

- **DiffPair Signal**: Specifies the name of the other pin in the differential pair. Values are the name of the user I/O or blank when unused.

- **I/O Standard**: I/O standard for a specific user I/O. When this field is blank for a user I/O, the software uses the appropriate device defaults. Values are a legal I/O standard for the user I/O in the device or blank.

- **Drive**: Drive strength of the I/O standard for a specific user I/O. Not all I/O standards accept a drive strength. If this field is blank, the tools use the default. Values are a number or blank.

- **Slew Rate**: Slew rate of the I/O standard for a specific user I/O. Not all I/O standards accept a slew rate. If this field is blank, the tools use the default. Values are FAST and SLOW.

- **Pull Type**: Specifies the pull type for the selected port. When using 3-state output (OBUFT) or bidirectional (IOBUF) buffers, the output can have a weak pull-up resistor, a weak pull-down resistor, or a weak "keeper" circuit. For input (IBUF) buffers, the input can have either a weak pull-up resistor or a weak pull-down resistor.

- **Phase**: Specifies the phase of an I/O relative to the phase of other I/O in the bank in cases of a synchronous phase offset.

- **Board Signal**: The name of the signal coming into the I/O from the board-level design.

- **Board Voltage**: Specifies the voltage level of the signal coming into the I/O from the board-level design.

- **BUFIO2_REGION**: Defines the BUFIO2 clocking region a port is related to.

- **IN_TERM/OUT_TERM**: Defines the optional `IN_TERM` or `OUT_TERM` driver impedance attributes for Spartan-6. This is most commonly left blank and is not yet supported for production devices. Using this terminal definition overrides the SLEW and DRIVE STRENGTH attributes and is not supported in SSN calculations.

- **OFFCHIP_TERM**: Specifies the external board level termination of the I/O. This is used for SSN calculations. If the field is left blank, PlanAhead uses the expected terminations in the SSN calculations and shows this expected termination by default in the SSN report and I/O Ports table. The expected terminations, as well as the corresponding shortened names that display in PlanAhead, can be found in the *Spartan-6 FPGA SelectIO Resources User Guide, (UG381)* as cited in Appendix E, Additional Resources.

Any column values that are not defined are retained by the PlanAhead tool when reading the CSV, and reported as user-defined columns in the I/O Ports view.

Defining Differential Pairs in the CSV

There are several properties used to define differential pairs in the CSV file: Signal Name, DiffPair Signal, DiffPair Type, and I/O Standard. The other values in the CSV are used to validate a diff pair, to ensure they are fully compatible, but they are not used to define the pair. You can define differential pairs in the CSV file in a number of ways.

The following outlines the different approaches to defining diff pairs:

- **Diff Pair**: This is a direct definition of the two signals which make up a differential pair. Two port entries, each have DiffPair Signal values linking to the Signal Name of the other, and have complementary DiffPair Type values, one N and one P. The tool checks to ensure that the other attributes such as I/O Standard are compatible when forming the diff pair.

- **Single-link Diff Pair**: Two port entries, with complementary DiffPair Type values (one N, one P), but only one port has a DiffPair Signal linking to the other Signal Name. The tool will create the differential pair if all other attributes are compatible.

- **Single Port Diff Pair**: A single port entry with a differential I/O Standard, a DiffPair Type value, and a DiffPair Signal that does not otherwise appear in the CSV. The tool will create the opposite side of the differential pair (the N or P side), with all properties matching those on the original port.

- **Inferred Diff Pair**: Two ports entries, with differential pair I/O standards (e.g. DIFF_HSTL, DIFF_SSTL), with Signal Names that infer the N and P side. The tool will infer a differential pair if all other attributes are compatible.

When the tool encounters any of these port entries in the CSV file, it either directly defines the differential pair, or it infers the differential pair if all the attributes are compatible. When differential pairs are inferred, the PlanAhead tool returns a prompt to confirm the assignment of the pairs. See Importing I/O Ports, page 254 for more information.

Outputs for Reports

Figure A-1 shows a the directory structure used for reporting output on Microsoft Windows 7. When the file location is not user-specified, the PlanAhead tool places report output files as follows:

- Outputs from Tcl commands will be written to the start-in directory, where the PlanAhead tool was launched.

- Outputs from the GUI will be written to the project directory by default.

- When running synthesis or implementation, the output files will be written to the run directories for the project.

- The location of the journal and log files for the PlanAhead tool depends on the operating system:

  - On Linux, the directory is the PlanAhead invocation directory.

  - Windows 7:`%APPDATA%/Roaming/Xilinx/PlanAhead/`

  - Windows XP:`%APPDATA%/Xilinx/PlanAhead/`
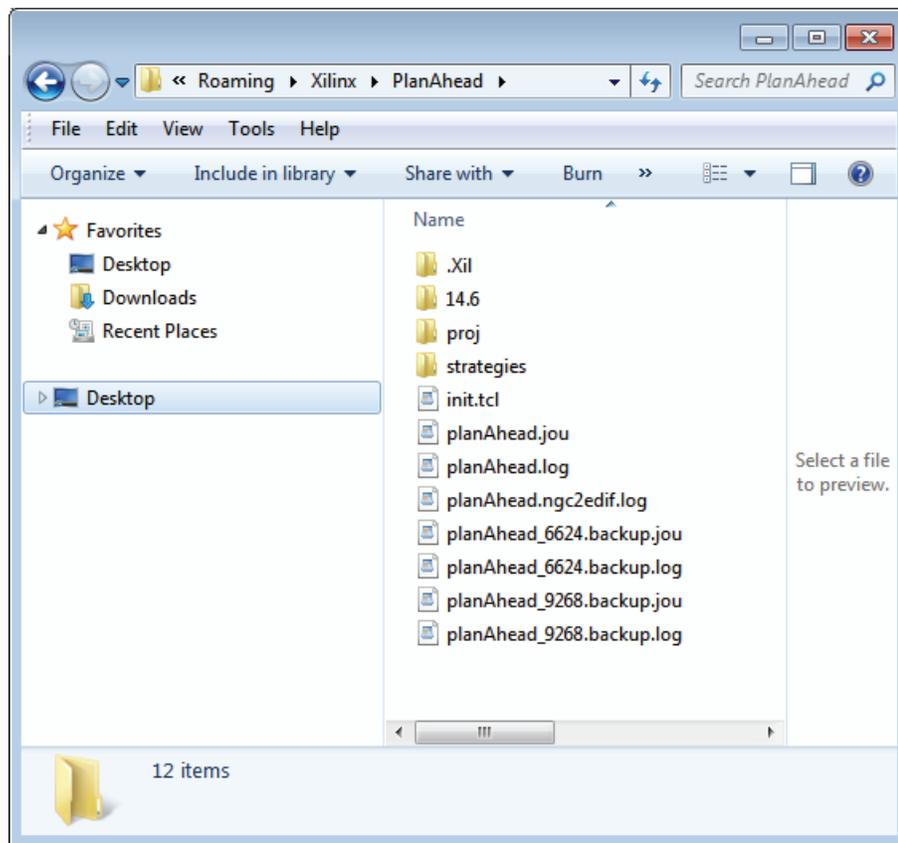
Figure A-1: **PlanAhead Directories for Reporting Output**

Table A-2 lists the PlanAhead output files and a description.

Table A-2: **Reporting Outputs**

File Name	Description
I/O Pin Assignment (CSV)	I/O pin assignments are stored in a CSV format file that contains the I/O port assignment and relative package pin information. This file used for RTL port header definition and PCB schematic symbol generation.
I/O Pin Assignment (RTL - Verilog, VHDL)	Verilog or VHDL format file contains the I/O port assignments defined as ports in the file header in a legal language format. This file is used for RTL port header definition.
Log File (planAhead.log, planAhead_*<id>*.backup.log)	The log file, `planAhead.log`, captures the contents of the messages created from running PlanAhead commands. View the file by selecting **File > Open Log File** from the main menu.
Journal File (planAhead.jou and planAhead_*<id>*.backup.jou)	The journal file, `planAhead.jou`, captures the Tcl commands from a session. View the file by selecting **File > Open Journal File** from the main menu.
	You can replay the journal file to reproduce the commands of the previous session. You can create Tcl scripts by copying commands from the journal file for later replay. It might be necessary to edit this file to remove any erroneous commands or commands from multiple sessions prior to replay.
	Note: Not every action from the GUI logs a Tcl command into the journal file.
Error Log Files (planAhead_pid*xxxx*.debug & hs_err_pid*xxxx*.log)	The error files can provide valuable information for debugging the PlanAhead tool in the event of an unexpected interrupt. If the PlanAhead tool issues a dialog box that warns of an internal exception error, the error files are stored When you open a case with Xilinx Technical Support, include any generated error log files, the journal file (`planAhead.jou`), and the log file (`planAhead.log`).
	These files contain no design data.
DRC Results	Each time Design Rule Checks (DRC) is run, the results can be written to a file.
Timing Analysis Results (Excel file)	The results from timing analysis can be exported into a text file. To export the data, select Export Statistics in the Timing Results view.
Netlist Module, Pblock, and Clock Region Statistics Reports	Resource statistics displayed in the Instance Properties, Clock Region, and Pblock Properties View can be exported to an Microsoft Excel format file. Information includes resource utilization, RPM and carry chain sizes, clocks and clocked instances, and other relevant resource data.
	To export the data from the Statistics tab of the Instance, Clock Region, or Pblock Properties View, select Export Statistics. The dialog box lets you define the information to include in the report as well as how many levels of hierarchy to report
SSN Analysis Report	The results from Simultaneous Switching Noise (SSN) analysis can be exported to a CSV or HTML report file by specifying a file name and location in the Run SSN Analysis dialog box
WASSO Analysis Reports	The results from the Weighted Average Simultaneous Switching Output (WASSO) analysis can be exported to a text report file by specifying a file name and location in the Run WASSO Analysis dialog box.
Strategy Files	The `/Strategy` directory contains files with your specified default command line options for ISE® implementation commands. You can apply a strategy to any given ISE attempt. You can either create strategies or copy a supplied strategy.

Outputs for Environment Defaults

The PlanAhead tool saves the current configuration of window layouts and themes to configuration and initialization files that are loaded when the tool is launched. You can also save custom themes, window layouts, and run strategies, to be loaded when you need them. The files defining these themes and layouts are written to the PlanAhead tool environment folders in the following locations:

- On Windows, the PlanAhead tool environment settings are stored in files located under the AppData folder (`%APPDATA%`) for a specific user:
  - Windows 7:`%APPDATA%/Roaming/Xilinx/PlanAhead/`
  - Windows XP:`%APPDATA%/Xilinx/PlanAhead/`
- On Linux, the PlanAhead tool environment settings are stored in files located under the user's Home directory:
 - `~/.Xilinx/PlanAhead/<version>`

Table A-3 lists the files, locations, and descriptions for the PlanAhead environment configuration files:

Table A-3: **PlanAhead Environment Default Outputs**

File Name	Description
View Display Options File (planAhead.ini)	The `../PlanAhead/<version>/planAhead.ini` file captures the settings in **Tools > Options** that include display color and other viewing options for the environment. For more information, see Configuring PlanAhead in Chapter 4.
	The PlanAhead tool saves your settings to the `planAhead.ini` file when you exit the tool. Upon invocation, it imports the file automatically and applies the settings to initialize the tool.
PlanAhead Themes (*<theme_name>*.patheme)	The `../PlanAhead/<version>/themes` directory contains `*.patheme` files that are created when you customize color and fill pattern themes for displaying layers in the PlanAhead GUI. You can select a theme file to use during the active session from a pull-down selection menu. For more information, see Configuring the Viewing Environment in Chapter 4.
View Layout Files (*<layout_name>*.layout)	The `../PlanAhead/<version>/layouts` directory contains `*.layout` files that define the layout configuration of the PlanAhead tool views in the GUI. You can create custom view layouts using the **Layout > Save Layout As** command. See Using View Layouts in Chapter 4 for more information.
Keyboard Shortcuts (shortcuts.xml)	The `../PlanAhead/<version>/shortcuts` directory contains a `shortcuts.xml` file that maps keyboard shortcuts to tool commands. You can define and configure multiple keyboard shortcuts, which are stored in the shortcuts file. Refer to Configuring Shortcut Keys in Chapter 4 for more information.
Custom Commands (commands.paini)	The `../PlanAhead/<version>/commands` directory contains the commands.paini file which stores custom TCL commands added to the PlanAhead tool GUI. Refer to Adding Custom Menu Commands in Chapter 4 for more information.

Outputs for Project Data

Figure A-2 is a diagram of PlanAhead Project data directory structure for an example project.
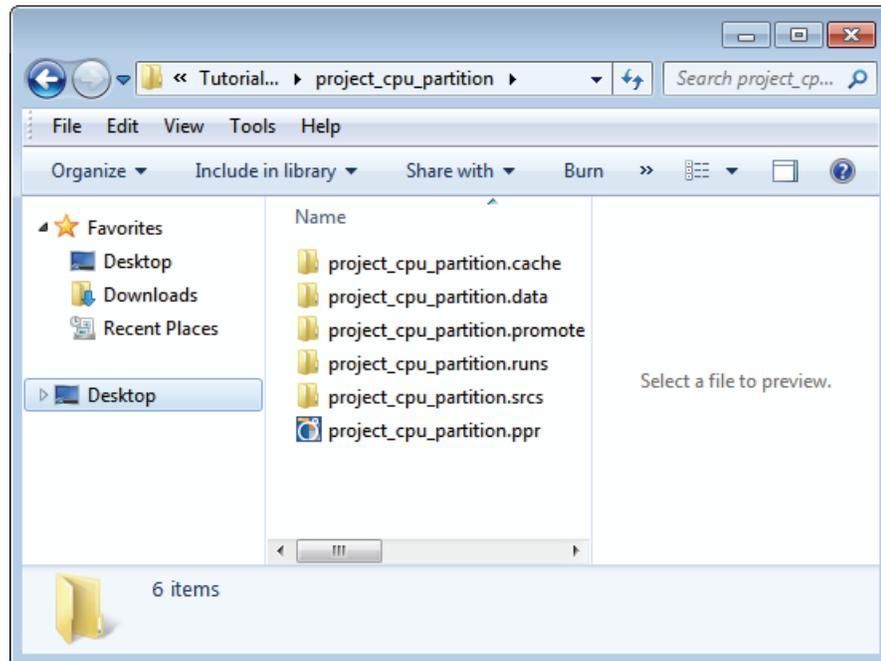
Figure A-2: **Project Data Directory Structure**

Table A-4 lists the PlanAhead Project Data outputs and descriptions.

Table A-4: **PlanAhead Project Data Outputs**

Output	Description
Project Directory `<project>`	When you create a new project, the PlanAhead tool creates a project directory in which to store the project file, the project data directory, and the ISE Implementation results. The project directory has the same name as the project name entered in the New Project wizard.
Project File (`project`.ppr)	The project file has the same name as the project name entered in the New Project wizard.
Cached Library Data (`project`.cache)	Contains a cached version of the Xilinx CoreLib used when running simulation.
Project Data Directory (`project`.data)	Contains project XML metadata files for HDL source files, design constraints, and simulation source files.
Project Data - Netlist Subdirectory (project.data/ netlist)	A /netlist subdirectory containing a copy of the netlist files for the design. For RTL-based projects, the PlanAhead tool creates a /Synthesis subdirectory for each run for the produced netlist, which refreshes each time the run is reset. For Netlist-based projects, the PlanAhead tool creates a single netlist directory that contains the imported netlist, including copies of all NGC core files used in the design.
Project RTL Directory (`projectname`.srcs)	The project sources directory stores the HDL source files, and UCF constraint files, that are imported into a project. These folders are maintained by the PlanAhead tool and do not require your attention. ***Caution!*** Modifying any of these files could result in project data corruption

Project Data-Simulation (*projectname*.sim)

The project simulation directory structure for both behavioral and timing simulation Runs is the same:

project_name/*project_name*.sim/*sim_run_name*/sim_#

Table A-5 lists the created directories and files for behavioral and timing simulation Runs:

Table A-5: **Behavioral and Timing Simulation Files and Directories**

File/Directory Name	Description	Simulation Type
fuse.log	Fuse execution log file.	Both
fuse.xmsgs	Fuse execution log file in XML format.	Both
fuseRelaunch.cmd and ISim.cmd	Batch commands for ISim, passed under the banner of tclbatch when -tclbatch is *NOT* specified.	Both
ISim.log	ISim execution log file.	Both
top.exe	ISim simulation executable created by fuse from the top module specified on the Simulation Launch dialog box (name varies based on the name of the top module.)	Both
top.prj	PlanAhead project file containing the top-level design.	Both
top.wdb	Waveform database file created by ISim.	Both
top_timing_sim.nlf	NetGen execution log file.	Timing
top_timing_sim.sdf	DF delay file output from netgen for use in timing simulation.	Timing
top_timing_sim.v	Verilog netlist output by Netgen used in Timing simulation (could be a VHDL file based on the specified -ofmt option.)	Timing
xilinxsim.ini	File containing logical-to-physical mappings of libraries.	Both
/ISim	Directory containing the isim_usage_statistics.htm file.	Both
/ISim/work	Directory containing the glbl.sdb and top.sdb files.	Both
ISim/top.exe.sim	Directory generated by fuse to store object code and data files for each design unit comprising the design. Contains top.exe, the simulation executable created by fuse to run ISim on the design.	Both
../top.exe.sim/ secureip	Directory containing design data and object code compiled by fuse.	Both
../top.exe.sim/ simprims_ver	Directory containing design data files and object code compiled by fuse.	Both
../top.exe.sim/work	Directory containing design data files and object code compiled by fuse (top.exe_main.c and top.exe_main.*os_type*.obj.)	Both

Outputs for ISE Implementation

Table A-6 provides a brief description of the files PlanAhead creates during ISE Implementation design operations. These files are maintained by PlanAhead and must not be modified manually.

Table A-6: **Outputs For ISE Implementation**

Output	Description
Run Directory (`projectname.runs`)	The PlanAhead tool lets you launch and queue multiple ISE Implementation attempts or *runs*. You are prompted to enter a location to create the run directory. The default location is in the project directory.
	Each run directory contains a complete EDIF netlist and UCF constraint file for that run. The PlanAhead tool creates a run script to launch the ISE commands with your specified options in each run directory.
	Each run directory contains the implementation design data including the netlist and the constraints files. When a satisfactory implementation result is achieved, you can copy and archive the entire run directory because it is self-contained.
EDIF Netlists (`.edf`)	PlanAhead exports EDIF format ASCII netlist files that are created during the following commands:
	• **Implement and Launch Runs (PlanAhead)**
	• **File > Export > Export Netlist**
	• **File > Export > Export Pblocks**
	• **File > Export > Export IP**
Run Implementation and Launch Runs	Launching PlanAhead implementation runs automatically exports the files required to implement the Runs and to launch the ISE commands with the options specified in the Strategy.
	When you launch a run, the PlanAhead tool exports the EDIF and UCF data automatically. When you launch a run, the PlanAhead tool creates a run directory containing a single EDIF-format netlist file and a UCF-format constraint file for the entire top-level design. The file names correspond to the original top-level netlist name contained in the originally imported EDIF file.
	If NGC/NGO format module netlist files are used, they are copied to each run directory.
	The Synthesis Run Properties view and the Implementation Run Properties view indicate where the actual run directory resides on disk.
Exported Netlists	Exporting a Netlist supplies the design EDIF file for ISE Implementation outside of the PlanAhead environment. When a Netlist is exported, the original logical netlist hierarchy is maintained in the output netlist. You can specify an output file name in the Export Netlist dialog box.
ChipScope Core Netlists (`.ngc`)	The PlanAhead tool is integrated with ChipScope™ Pro Analyzer, which lets you insert and configure Integrated Logic Analyzer (ILA) cores. An NGC format netlist for the core is compiled when the core is implemented. The PlanAhead tool places the core in the project `/netlist` directory and copies it to each Implementation run directory as Runs are launched. Refer to Chapter 12, Programming and Debugging the Design, for more information.
Constraint Files (`.ucf`)	The PlanAhead tool writes UCF ASCII files containing timing and physical constraints that are used for ISE. These files are created during the following commands:
	• **Implement and Launch Runs (PlanAhead)**
	• **File > Export > Export Constraints**
	• **File > Export > Export Pblocks**
	• **File > Export > Export IP**

Table A-6: **Outputs For ISE Implementation** *(Cont'd)*

Output	Description
Run Implementation and Launch PlanAhead Runs	When you launch runs, the PlanAhead tool exports the EDIF and UCF data automatically.
	When you launch a run, the PlanAhead tool creates a run directory that contains the original logic hierarchy in the output netlist. The exported files for the run consist of a single EDIF format netlist file and a UCF format constraint file for the entire top-level design. The file names correspond to the original top-level netlist name of the imported EDIF file.
Exported Constraints	When you export constraints the PlanAhead tool attempts to preserve the original UCF content and structure, including comments.
	You can specify the output constraints file in the Export Constraints dialog box.
Exported Pblocks	Exporting a Pblock writes the EDIF and UCF files for the specified Pblocks to use for ISE Implementation outside of the PlanAhead environment.
	When you export a Pblock, the PlanAhead tool derives the netlist hierarchy based on the Pblock assignments. The resulting UCF references the PlanAhead *physical hierarchy* structure to match the exported EDIF netlist names and provides flexibility when using a block-based Implementation strategy.
	The exported Pblock files consist of a single netlist file and constraint file. The PlanAhead tool automatically creates and maintains a block-level directory structure. When you export selected Pblocks, the PlanAhead tool creates `pblockname_CV` subdirectories containing `pblockname_CV.edn` and `pblockname_CV.ucf` files.
	Export Pblocks is typically used for a complex IP that contains the physical hierarchy, on which you can close timing, and then use those instances in other designs without restructuring your code to obtain a netlist.
Exported IP	Exporting IP writes the EDIFs and UCF s for specified netlist modules for use when creating reusable IP blocks.
	When you run the Export IP command on a selected module instance in the design it exports the Pblock *logical hierarchy* and placement constraints. The exported files include the EDIF netlist and UCF physical constraints in the original logical netlist format. This allows for easier implementation in the next design by keeping the interface identical.
	You can use the exported UCF to re-create the Pblock placement constraints. You can duplicate identical placement for multiple modules by moving the modules after they are imported.
	Figure 10-22, page 324 is an example of how Export Pblock and Export IP can be used to export different parts of the design, based on either logical (Export IP) or physical (Export Pblock) hierarchy.
ISE Launch Scripts (`jobx.bat/sh` & `runme.bat/sh` & `.ISE_command.rst`)	When you launch a run, the PlanAhead tool creates ISE launch scripts automatically. These scripts contain commands and command-line options specified in the PlanAhead Strategy.
	The `jobx.bat/sh` scripts are located under the project run directory in a `/jobs` subdirectory. These scripts sequentially launch each selected run. The script calls each run-specific `runme.bat|sh` script. You can launch these scripts independently also.

Appendix B

PlanAhead DRCs

RTL DRCs: Power and Performance

Table B-1 and Table B-2, page 416 list the RTL Power and the RTL Performance DRCs.

RTL DRCs: Power

Table B-1: **Power Rules**

Rule Name	Rule Abbrev	Rule Intent	Severity
Constantly enabled synchronous RAM	RPRC	A described RAM (either inferred or instantiated), which is constantly enabled, was found in one or both ports. If it can be determined that this RAM is not constantly accessed. Significant power reduction might be seen by describing the logic to disable the RAM unless it is being accessed.	Warning
Inefficient dangling block RAM port	RPRM	A RAM in which there is an unconnected output port has been detected, and the WRITE_MODE is set to a value other than NO_CHANGE. Modifying the description of the RAM to set unconnected output port (WRITE_MODE set to NO_CHANGE) could save up to 10% of the dissipated block RAM power.	Warning
Shallow RAM implemented in block RAM	RPRS	Virtex®-5 and Virtex-6 devices: For wide (over 18-bits) and shallow (64-bits or less) RAM, it is generally advantageous to choose **SelectRAM™** (LUT-based RAM referred to as distributed RAM) whenever possible unless the RAM is being used as a FIFO, in which case the crossover point becomes a depth of 32-bits or less. When building interfaces less than 18-bits wide, the LUT-based **SelectRAM** could be a better choice for depths up to 128-bits; however, generally past that, the dedicated block RAM is a better choice for power.	Warning
Inefficient mapping of small multiplier in DSP block	RPDS	Small multipliers mapped to DSP or to other hard multipliers IP, such as MULT18X18, should be pushed to MSBs. The rest of the LSBs should be mapped to ground. In this way, the carry propagation is reduced to its minimum. Usual Implementation, especially when inferring the multiplier, uses LSBs and sign extensions to map the MSBs.	Warning

RTL DRCs: Performance

Table B-2: **Performance Rules**

Rule Name	Rule Abbrev	Rule Intent	Severity
Inefficient library element instantiation	RPWL	Found *instance name* of type *library_component_name* that belongs to another FPGA family. This might result in suboptimal performance. The ISE software might remap this element automatically onto a similar element in the selected family. However, modifying the source code to infer or instantiate native elements takes advantage of any added or expanded functionality in the element. This could improve area utilization and performance.	Warning
Missing pipeline register	RPPR	Found multiplier with unregistered outputs. You can improve the multiplier clock-to-out performance by adding a level of registers. In addition, for best results, avoid using asynchronous control signals on these registers. Found RAM/ROM with unregistered outputs. You can improve the RAM/ROM clock-to-out performance by adding a level of registers. In addition, for best results, avoid using asynchronous control signals on these registers.	Warning
Inefficient pipeline register	RPIP	Found *register_name* (*file_name:line_number*) register with asynchronous control signals on input or output of multiply function. Dedicated DSP hardware resources do not have asynchronous control signals, such as preset or clear. The registers cannot be mapped into the dedicated hardware resources resulting in suboptimal use of the device.	Warning
Found Black Box instance not belonging to UniSim library	RPBX	Component/Module *component/module_name* description unavailable during Synthesis (*file_name:line*). Paths to and from this black box cannot be optimized. Synthesis tool utilization estimates and mapping decisions could be negatively affected.	Warning
Found latch in design	RPLD	Found latch description for signal *signal_name* (*file_name:line_num*). Latches creates difficult to analyze timing paths which require post Implementation simulations to ensure implemented design match expected behavior.	Warning
Found combinatorial loop in design	RPCL	Found combinatorial loop for signal *signal_name* (*file_name:line_number*). Combinatorial loops are generated when a cone of combinatorial logic uses its outputs to feedback as partial input to the same cone of logic. The total combinatorial delay from source to destination should be increased by the feedback path delay. This type of structure could be required from the design expected behavior or might be unintentional.	Warning

Floorplanning DRCs

Floorplan Pblock DRCs

Table B-3: **Floorplan Pblock DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Longest Carry Chain Height	LCCH	Checks that the Pblock height can accommodate longest carry chain assigned to Pblock.	Warning
Pblock overlap	FLBO	Checks for overlapping Pblock rectangles.	Information
Pblock Partition	FLBP	Checks that the LUT to MUXCY and MUXFx connection is not broken by a Pblock partition.	Error

Table B-3: **Floorplan Pblock DRCs** *(Cont'd)*

Rule Name	Rule Abbrev	Rule Intent	Severity
Resource Utilization	UTLZ	Checks that Pblocks have enough resources for logic assigned to them.	Warning for SLICE logic. Error for non-SLICE logic.
Area Group Tile Alignment	FLBA	Checks that the site ranges in `AREA_GROUP` constraints are aligned with the CLB grid.	Warning

Bank DCI DRCs

Table B-4: **DCI Cascade DRC**

Rule Name	Rule Abbrev	Rule Intent	Severity
DCI Cascade Checks	DCIC	Checks that DCI cascade constraint is legal.	Error
DCI Cascade with part compatibility	DCICPC	Warns the user to load the UCF into other compatible parts, and to run DRC manually to ensure the DCI cascades are valid.	Warning
DCI check for I/O standard legality	DCICIOSTD	Ensures that there are no conflicts related to Vcc and DCI termination of I/O standards used within the DCI Cascade.	Error

Table B-5: **IDelay Control DRC**

Rule Name	Rule Abbrev	Rule Intent	Severity
IDelayCtrl Checks	IDLYCTRL	Checks that IDelay placement is consistent with IDlyController LOCs	Error

For a full list of Bank IO standard rules see I/O Port and Clock Logic and Placement DRC Rule Descriptions, page 419.

ClkBuf DRC

Table B-6: **ClkBuf DRC**

Rule Name	Rule Abbrev	Rule Intent	Severity
BufR & BufIO Locations	BUFRIOC	Checks that BUFR and BUFIO driven by the same regional clock terminal are placed at mutually route-able locations.	Error

See Global Clock DRCs, page 420 for a full list of global clock rules.

DSP48 DRCs

Table B-7: **DSP48 DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
DSP48 output registers	DPOR	DSP48 has a register on the output side; to use this register the register should be synchronously controlled. (Virtex-4 only)	Information
DSP48 input registers	DPIR	DSP48 has a register on the input side; to use this register the register should be synchronously controlled. (Virtex-4 only)	Information
DSP48 output pipelining	DPOP	DSP48 has a register on the output side; using this pipeline mechanism improves performance. (Virtex-4 only)	Information
DSP48 multiplier output pipelining	DMOP	DSP48 output is not pipelined. Pipelined output improves performance.	Warning
DSP48 input pipelining	DPIP	DSP48 has a register on the input side; using this pipeline mechanism improves performance. (Virtex-4 only).	Information
DSP48 cascade	DPCA	DSP48 cascade check.	Warning
DSP48 asynchronous	DPREG	DSP48 asynchronous feedback.	Warning

RAMB16 DRC

Table B-8: **RAMB16 DRC**

Rule Name	Rule Abbrev	Rule Intent	Severity
RAMB16 output registers	RBOR	RAMB16 has a register on the output side; to use this register, the register should be synchronously controlled. (Virtex-4 only).	Information

RAMB DRC

Table B-9: **RAMB16 DRC**

Rule Name	Rule Abbrev	Rule Intent	Severity
RAMB Read first mode	RBRF	Clock restrictions for `READ_FIRST` mode.	Warning

FIFO DRC

Table B-10 lists the FIFO DRC.

Table B-10: **FIFO DRC**

Rule Name	Rule Abbrev	Rule Intent	Severity
FIFO Synchronous	FSYN	Check for synchronous FIFO.	Warning

Netlist DRC

Table B-11 lists the Netlist DRC.

Table B-11: **Netlist DRC**

Rule Name	Rule Abbrev	Rule Intent	Severity
Driverless Nets	NDRV	Checks that each net has a proper driver pin.	Warning

Instance DRC

Table B-12 lists the Instance DRCs.

Table B-12: **Instance DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Black Box Instances	INBB	Checks that there is no blackbox (undefined logics in the netlist).	Warning

Attribute DRCs

Table B-13 lists the Attribute DRCs.

Table B-13: **Attribute DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Invalid attribute	AVAL	Checks for invalid attribute values.	Warning
Undefined attribute	ADEF	Checks for undefined attribute values.	Warning

Required Pin DRC

Table B-14 lists the Required Pin DRC.

Table B-14: **Required Pin DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Unconnected pin	REQP	Required pin not connected.	Warning

I/O Port and Clock Logic and Placement DRC Rule Descriptions

The I/O Port and Clock Logic DRCs available in PlanAhead is not an exhaustive list of I/O-related DRCs. Consult the device documentation for more information about I/O ports and clock region specifications.

In some cases the Severity level issued by PlanAhead might differ from the severity level of the same condition reported by ISE Implementation tools.

Selecting DRC Rules, page 195 provides other DRC rule descriptions.

Global Clock DRCs

Table B-15 provides the Global Clock DRCs, abbreviation, intent, and severity.

Table B-15: **Global Clock Rules**

Rule Name	Rule Abbrev	Rule Intent	Severity
IBUFG to DCM connectivity	IDCM	IBUFGs have dedicated routing only to all DCMs on the same edge (top, bottom, left, right) of the device.	Warning
DCM to BUFG connectivity	DCMB	DCM can connect to a maximum of four BUFGs. There are pairs of buffers with shared dedicated routing resources such that if both are driven by the same DCM, one of the two is driven using non-dedicated routing resources; this causes the design to fail. If the buffers are numbered 1 through 8 from left to right, there are four pairs of exclusives: 1:5, 2:6, 3:7, 4:8. If a buffer is placed in Site 1, another driven by the same DCM should not be placed in Site 5.	Error
Number of BUFGs allowed for DCM	DCMN	DCM can connect only up to 4 BUFGs. This is related to DCMB.	Error
DCM and BUFG connectivity	DCME	BUFGs have dedicated routing only to all DCMs on the same edge (top, bottom, left, right) of the device.	Warning

Placer DRCs

Table B-16 lists the Placer DRCs.

Table B-16: **Placer DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Placement constraint	PLCR	Placement constraint check for clock regions.	Error
Clock placer	PLCK	Checks the clock placement for valid locations. Included in PLCK are: • IOBUFR, which checks that a regional clock terminal and the related BUFR are placed at mutually route-able locations • IOBUFIO, which checks that a regional clock terminal and the related BUFIO are placed at mutually route-able locations	Error
Design lock	PLDL	Checks for invalid LOC constraints.	Error
Valid placement	PLVP	Checks for constraints that would lead to an infeasible placement	Error
Valid placement	PLIO	Checks for valid placement of I/O	Error

IOB DRCs

Table B-17, page 421 lists the IOB DRCs, abbreviation, intent, and severity.

Table B-17: **IOB DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Port Properties	PORTPROP	Check for inconsistencies within the properties of a port	Error
Differential I/O pads	IODI	Differential I/O P and N signals should be LOCed in dedicated differential pair.	Error
I/O Standard Type	IOSTDTYPE	Ensures that a Diff pair I/O Standard has been applied to diff pair pins only.	Warning
Number of IOs	IOCNT	Indicates whether more I/O Ports are defined than there are pins in the target device.	Warning
I/Os placement	IOPL	I/Os placement on disallowed site. Included in the IOPL are: • IOPR, which checks that a port is not placed on a prohibited pin. • IOLVDSCC, which checks that differential output standards are not used on low-capacitance sites that cannot support them.	Error
IO Part Compatibility for Bank Type	IOPCBT	This message occurs in 7-series designs with multiple compatible parts, when the use of an I/O standard is legal in the bank it is placed in on one part, but not in legal in the corresponding bank of one of the compatible parts. This can occur when the bank on one part is a high performance (HP) I/O bank, but the corresponding bank on a compatible part is a high range (HR) I/O bank. The set of legal I/O standards is not the same for these two bank types. To resolve this issue: • Place the ports on an I/O bank that supports the I/O standard on all the compatible parts. • Change to an I/O standard that is legal in both HP and HR banks. • Change to different compatible parts.	Warning
Prohibit not specified for part compatibility	IOPCPR	For designs that use part compatibility, checks that if any package pin does not exist on at least one compatible part, it is marked as "prohibit" and nothing is placed on it.	Error
MGT not allowed for part compatibility	IOPCMGT	Indicates whether part compatibility is used with two parts that have different serial transceiver supply voltages, thereby disallowing the use of serial transceivers.	Warning
I/O Crosstalk to MGT	IOCTMGT	Check for possible crosstalk problems between I/Os and serial transceivers.	Warning

I/O Bus SLR Crossings	IOBUSSLRC	The following message occurs when bits of the same bus are placed on different Super Logic Regions.
		
		`Bus port <BUSPORT[LO:HI]> spans more than one Super Logic Region (SLR).`
		`Bits placed in SLR <SLR1>: 0-3`
		`Bits placed in SLR <SLR2>: 4-7`
		
		This is not recommended as it makes routing and timing closure more difficult.
		
		Eliminate the warning by moving all related bus ports into the same SLR.
			Warning

Table B-17: **IOB DRCs** *(Cont'd)*

Rule Name	Rule Abbrev	Rule Intent	Severity
Part compatibility	IOPCSLR	Check for part compatibility between monolithic and multi-die devices.	Information
IOB clock sharing	IOCS	IOB sites are divided into pairs for the purpose of sharing clock routing resources. These pairs are normally LVDS pairs as well. In some cases there could be routing issues based on how the flops are packed inside the IOB. To resolve this issue flops need to be assigned to specific BEL.	Warning
IOB set reset sharing	IOSR	IOB site has input, output, and tri-state registers; all of these registers share same set/reset signal. Cannnot pack registers with different reset signals.	Error

Bank I/O Standard DRCs

Table B-18 lists the Bank I/O standard DRCs, abbreviation, intent, and severity.

Table B-18: **Bank I/O Standard DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Bank I/O Standard Vcc	BIVC	`IOSTANDARD`-based VOUT voltage compatibility check for IOs in that bank.	Error
Bank I/O Standard Support	BIVB	Checks that the I/O Standard is supported in the I/O bank.	Error
Bank I/O standard Termination	BIVT	`IOSTANDARD`-based DCI Termination voltage compatibility check for IOs in that bank.	Error
Bank I/O Standard VREF	BIVR	`IOSTANDARD`-based VREF voltage compatibility check for I/Os in that bank.	Error
Bank I/O Standard VREF Utilization	BIVRU	Checks for an available VREF site in I/O banks which implement I/O Standards requiring a VREF.	Warning
Bank I/O Standard	BIIVRC	Checks for a conflict between the I/O standards of a Bank and an `INTERNAL_VREF` constraint on the bank. Standards in a bank cannot require a VREF voltage that differs from that specified by an `INTERNAL_VREF` constraint for the bank.	Warning
Bank I/O Simultaneous Switching Output Limits	BISLIM	Checks the I/O placed within an I/O bank against Simultaneous Switching Noise (SSN) Output.	Error
Bank I/O Standard VRN/VRP Occupied	DCIP	There are dedicated VRP and VRN I/O sites in I/O banks which can be used as regular IOs also. If a DCI I/O standard is used in that bank these IOs should be left unoccupied.	Error
Inconsistent Diff pair I/O Standards	DIFFISTD	Checks that the terminals of a differential pair have the same I/O standard.	Error
Inconsistent Diff pair I/O Standards	DIFFISTDDrv	Checks that the terminals of a differential pair have the same drive.	Error
Inconsistent Diff pair I/O Standards	DIFFISTDSlew	Checks that the terminals of a differential pair have the same slew.	Error

Table B-18: **Bank I/O Standard DRCs** *(Cont'd)*

Rule Name	Rule Abbrev	Rule Intent	Severity
Vccaux Voltage requirement	VCCAUX1	Warns of any requirements on for LVCMOS25.	Warning
Vccaux Voltage requirement	VCCAUX2	Warns of any requirements on LVPECL_33 and TMDS_33.	Error

ChipScope DRCs

Table B-19 lists the ChipScope DRCs, abbreviation, intent, and severity.

Table B-19: **ChipScope DRCs**

Rule Name	Rule Abbrev	Rule Intent	Severity
Unconnected channel	CSUC	Checks for unconnected channels.	Error
Clocked by non-clock net	CSCL	Checks for non-clock net connected to debug clock port.	Warning
Device block RAM	CSBR	Checks if device has sufficient block RAM resources to implement ChipScope debug cores.	Warning

Installing Releases with XilinxNotify

PlanAhead Release Strategy

The PlanAhead™ application is targeted to introduce new technology and respond to customer requests. New releases are periodically introduced. The version number reflects the release (for example: 14.1 or 14.3). The **Help > About PlanAhead** command displays the currently installed PlanAhead application version.

To check for new releases, run the **Help > Check for Updates** command, as described in the following subsections.

Refer to the Appendix E, Additional Resources for more information about Xilinx tool installation.

Running XilinxNotify

The XilinxNotify tool is the preferred method of obtaining software updates. It performs the following activities:

- Compares the latest version of Xilinx software updates available on http://www.xilinx.com/support/ with what you have installed, and notifies you if a newer version is available.

- Provides a **Download** button that launches a browser, allowing you to login to the Xilinx Download Center.

When you login, the download of your selected product starts. XilinxNotify can be run in any of the following ways:

- Automatic periodic checks upon software invocation.

- Select **Help > Check for Updates**.

- Type `xilinxnotify` in a Linux shell

Note: The **Edit** > **Preferences** menu selection lets you control the frequency of automatic checks at start-up.

XilinxNotify Network Installations

By default, the machine used to install the PlanAhead tool to a network location is enabled to automatically check for updates from Xilinx.com. All users pointing to the software on this network location have the option disabled by default.

You can enable the option using the **Tools > Options** command from the main menu, and selecting the **General** option on the left side menu, as shown in Figure C-1, page 426. The Miscellaneous category includes the option **Automatically check xilinx.com for software updates on startup**. You can also manually check using **Help > Check for Updates** from the main menu.
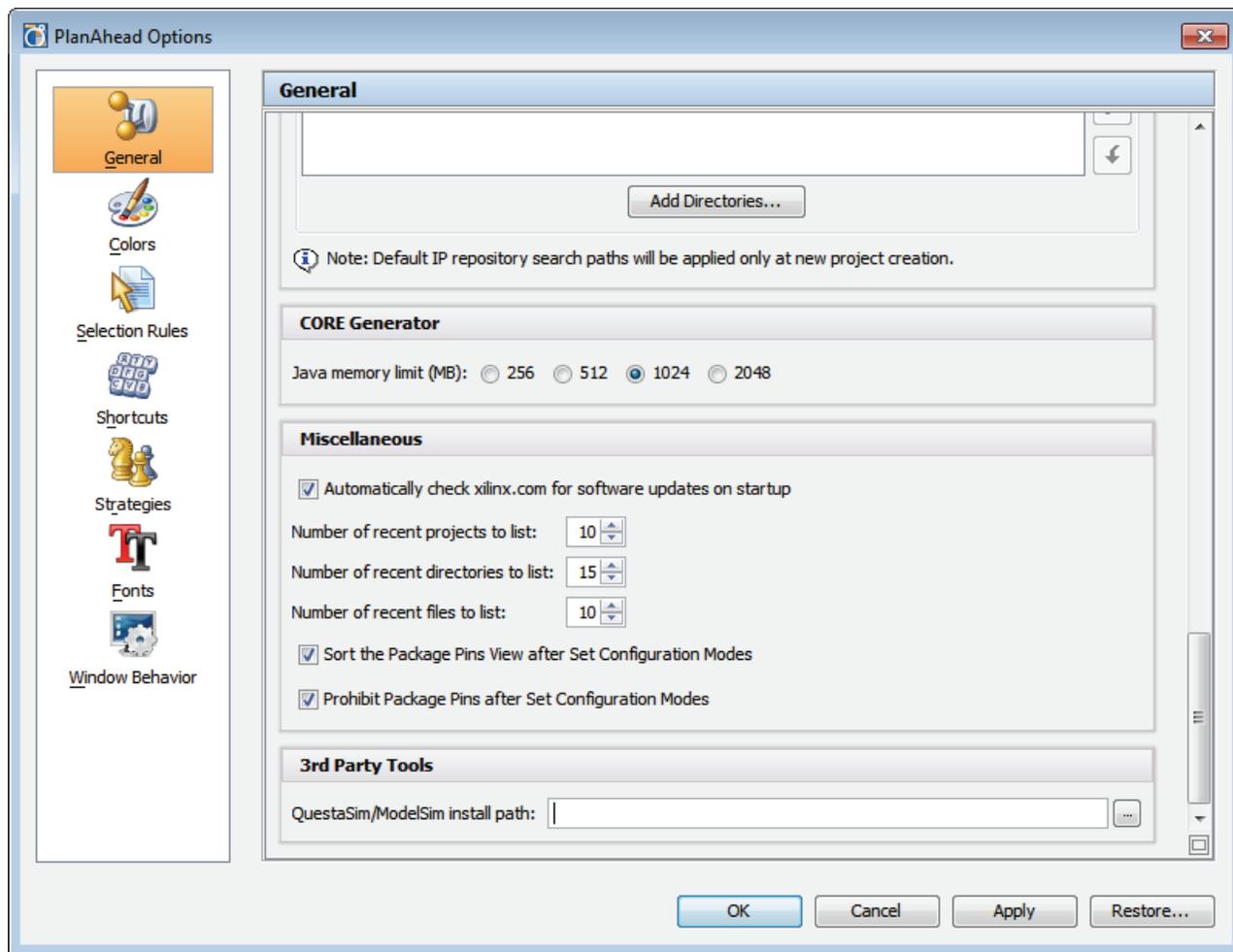
Figure C-1: **Automatically Check xilinx.com for Software Updates**

Note: To perform a software update installation, you must have write permissions for the $XILINX installation directory.

New releases are regularly made available on the Download Center at:
http://www.xilinx.com/support/download/index.htm.

Configuring SSH Without Password Prompting

The PlanAhead™ application allows you to run synthesis and implementation runs on remote hosts, or multiple hosts simultaneously. The multiple host capabilities, for executing synthesis and implementation runs in the PlanAhead tool, uses Secure Shell (SSH), a service provided by Linux operating system. Prior to configuring multiple hosts in the PlanAhead tool, you should configure SSH so that you are not prompted for a password each time you log in to a remote machine.

Setting Up SSH Key Agent Forward

SSH configuration is accomplished with the following commands at a Linux terminal or shell:

Note: This is a one-time step, and when successfully set-up, does not need to be repeated.

1. Run the following command at a Linux terminal or shell to generate a public key on your primary machine. Though not required, it is a good practice to enter (and remember) a private key phrase when prompted for maximum security.

   ssh -keygen -t rsa

2. Append the contents of your publish key to an `authorized_keys` file on the remote machine. Change `remote_server` to a valid host name:

   cat ~/.ssh/id_rsa.pub | ssh *remote_server* **"cat - >> ~/.ssh/ authorized_keys"**

3. Run the following command to prompt for your private key pass phrase, and enable key forwarding:

   ssh -add

You should now be able to `ssh` to any machine without typing a password. The first time you access a new machine, it prompts you for a password; upon subsequent access it does not prompt. If you are always prompted for a password, contact your System Administrator to have your Linux account set up for passwordless SSH.

After a passwordless SSH is set up, you can continue configuring the remote host. (Linux only.)

Additional Resources

The following are links to additional resources available for your use.

Xilinx Resources

- **Device User Guides**:
  http://www.xilinx.com/support/documentation/user_guides.htm

- **Xilinx Glossary**: http://www.xilinx.com/company/terms.htm

- *ISE Design Suite 14: Release Notes, Installation, and Licensing (UG631)*
  http://www.xilinx.com/cgi-bin/docs/rdoc?v=14.6;t=release+notes

- **Product Support and Documentation:** http://www.xilinx.com/support

Hardware Documentation

- 7 Series Device Documentation:
  http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/silicon_devices/fpga/num-7-series.html

- *7 Series FPGAs SelectIO Resources User Guide (UG471)*:
  http://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf

- *Virtex-6 FPGA Configuration User Guide (UG360)*:
  http://www.xilinx.com/support/documentation/user_guides/ug360.pdf

- *Virtex-6 FPGA SelectIO Resources User Guide (UG361)*:
  http://www.xilinx.com/support/documentation/user_guides/ug361.pdf

- *Virtex-6 FPGA Clocking Resources User Guide (UG362)*:
  http://www.xilinx.com/support/documentation/user_guides/ug362.pdf

- *Spartan-6 FPGA Configuration User Guide (UG380)*:
  http://www.xilinx.com/support/documentation/user_guides/ug380.pdf

- *Spartan-6 FPGA SelectIO Resources User Guide (UG381)*:
  http://www.xilinx.com/support/documentation/user_guides/ug381.pdf

- *Spartan-6 PCB Design Guide (UG393)*:
  http://www.xilinx.com/support/documentation/user_guides/ug393.pdf

- *Virtex-5 FPGA Configuration User Guide (UG191):*
  http://www.xilinx.com/support/documentation/user_guides/ug191.pdf

- *Virtex-4 FPGA Configuration User Guide (UG071)*:
  http://www.xilinx.com/support/documentation/user_guides/ug071.pdf

ChipScope Documentation

- *ChipScope Pro Software and Cores User Guide (UG029)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
  chipscope_pro_sw_cores_ug029.pdf

- *Debugging with ChipScope (UG677)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
  PlanAhead_Tutorial_Debugging_w_ChipScope.pdf

- *Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Applications (UG750)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug750.pdf

EDK Documentation

- **Platform Studio and EDK website**:
  http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm

- **EDK Tutorials website**:
  http://www.xilinx.com/support/documentation/dt_edk_edk14-1_tutorials.htm

- **XPS/EDK Supported IP website**:
  http://www.xilinx.com/ise/embedded/edk_ip.htm

- *EDK Concepts, Tools, and Techniques (UG683)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/edk_ctt.pdf

- **EDK Profiling Guide** (UG448):
  http://www.xilinx.com/support/documentation/xilinx14_6/edk_prof.pdf

- *Embedded System Tools Reference Manual (UG111)*:
  http://www.xilinx.com/support/documentation/xilinx14_6/est_rm.pdf

- *OS and Librarires Document Collection (UG643)*:
  http://www.xilinx.com/support/documentation/xilinx14_6/oslib_rm.pdf

- *MicroBlaze Processor User Guide (UG081)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/mb_ref_guide.pdf

- *Platform Specification Format Reference Manual (UG642)*:
  http://www.xilinx.com/support/documentation/xilinx14_6/psf_rm.pdf

- *PowerPC 405 Processor Block Reference Guide (UG018):*
  http://www.xilinx.com/support/documentation/user_guides/ug018.pdf

- *PowerPC 405 Processor Reference Guide (UG011):*
  http://www.xilinx.com/support/documentation/user_guides/ug011.pdf

- *PowerPC 440 Embedded Processor Block in Virtex-5 FPGAs (UG200)*:
  http://www.xilinx.com/support/documentation/user_guides/ug200.pdf

ISE Documentation

- **Libraries Guides**:
  http://www.xilinx.com/support/documentation/dt_ise14-5_librariesguides.htm

- **ISE Design Suite Documentation**:
  http://www.xilinx.com/support/documentation/dt_ise14-5.htm

  - *Command Line Tools User Guide (UG628):*
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/devref.pdf

- *Constraints Guide (UG625)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/cgd.pdf
- *Data2MEM User Guide (UG658)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/data2mem.pdf
- *ISim User Guide (UG660)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/plugin_ism.pdf
- *Synthesis and Simulation Design Guide (UG626)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/sim.pdf
- *Timing Closure User Guide (UG612):*
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug612.pdf
- *Xilinx/Cadence PCB Guide (UG629)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/cadence_pcb.pdf
- *Xilinx/Mentor Graphics PCB Guide (UG630)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/mentor_pcb.pdf
- *XPower Estimator User Guide (UG440)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug440.pdf
- *XST User Guide for Virtex-4, Virtex-5, Spartan-3, and Newer CPLD Devices (UG627)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/xst.pdf
- *XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices (UG687)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/xst_v6s6.pdf

- **ISE Methodology Guides:**
  - *Power Methodology Guide (UG786)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug786_PowerMethodology.pdf
  - *Large FPGA Methodology Guide (UG872)*: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug872_largefpga.pdf

- **ISE Tutorials:**
  - *ISE In-Depth Tutorial (UG695)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ise_tutorial_ug695.pdf
  - *ISE RTL Technology Viewer Tutorial (UG685):*
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug685.pdf
  - *ISim In-Depth Tutorial (UG682)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug682.pdf
  - *Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Applications (UG750)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug750.pdf
  - *Xilinx Power Tools Tutorial (UG733)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug733.pdf

System Generator for DSP Documentation

- *System Generator for DSP Reference Guide (UG638)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/sysgen_ref.pdf
- *System Generator for DSP Getting Started Guide (UG639)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/sysgen_gs.pdf
- *System Generator for DSP User Guide (UG640)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/sysgen_user.pdf

- **System Generator website:**
  http://www.xilinx.com/tools/sysgen.htm

Partial Reconfiguration Documentation

- **Partial Reconfiguration website**: www.xilinx.com/tools/partial-reconfiguration.htm
- *Partial Reconfiguration User Guide (UG702)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug702.pdf
- **Tutorials**:
  - *Design Preservation Tutorial (UG747)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
    PlanAhead_Tutorial_Design_Preservation.pdf
  - *Partial Reconfiguration Tutorial (UG743)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
    PlanAhead_Tutorial_Partial_Reconfiguration.pdf
  - *Partial Reconfiguration of a Processor Peripheral Tutorial (UG744)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
    PlanAhead_Tutorial_Reconfigurable_Processor.pdf
- **White Papers**:
  - *Repeatable Results with Design Preservation (WP362)*:
    http://www.xilinx.com/support/documentation/white_papers/wp362.pdf

Application Notes

- **AXI Multi-Ported Memory Controller** *(XAPP739)*:
  http://www.xilinx.com/support/documentation/application_notes/xapp739_axi_mpmc.pdf

- *Fast Configuration of PCI Express Technology through Partial Reconfiguration
  (XAPP883)*:
  http://www.xilinx.com/support/documentation/application_notes/
  xapp883_Fast_Config_PCIe.pdf

- *PRC/EPRC: Data Integrity and Security Controller for Partial Reconfiguration (XAPP887)*:
  http://www.xilinx.com/support/documentation/application_notes/xapp887_PRC_EPRC.pdf

PlanAhead Documentation

- **PlanAhead User Guides**:
  http://www.xilinx.com/support/documentation/dt_planahead_planahead/14-1_userguides.htm
  - *Floorplanning Methodology Guide (UG633)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
    Floorplanning_Methodology_Guide.pdf
  - *Hierarchical Design Methodology Guide (UG748)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
    Hierarchical_Design_Methodology_Guide.pdf
  - *Pin Planning Methodology Guide (UG792)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug792_pinplan.pdf
  - *PlanAhead Tcl Command Reference Guide (UG789)*:
    http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/
    ug789_pa_tcl_commands.pdf

- **PlanAhead Tutorials:**
  http://www.xilinx.com/support/documentation/dt_planahead_planahead14-1_tutorials.htm
  - *Quick Front- to-Back Flow Overview (UG673)*
  - *I/O Pin Planning (UG674*)
  - *RTL Design and IP Generation(UG675)*
  - *Design Analysis and Floorplanning (UG676)*
  - *Debugging with ChipScope (UG677)*
  - *Team Design (UG839)*
  - *Design Preservation (UG747)*
  - *Partial Reconfiguration (UG743)*
  - *Reconfiguration with Processor Peripheral (UG744)*

Zynq Documentation

- *Zynq-7000 All Programmable SoC Overview (DS190)*:
  http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf
- **Zynq-7000 Software Developers Guide** (UG821):
  http://www.xilinx.com/support/documentation/user_guides/ug821-zynq-7000-swdev.pdf
- **Zynq All Programmable SoC Concepts, Tools, and Techniques Guide**, *(UG873)*:
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_6/ug873_zynq_ctt.pdf
- **USB Cable Installation Guide** (UG344):
  http://www.xilinx.com/support/documentation/user_guides/ug344.pdf
- **Platform Cable USB II Data Sheet** (DS593):
  http://www.xilinx.com/support/documentation/data_sheets/ds593.pdf

IP Documents

- **Xilinx IP Center:**
  http://www.xilinx.com/ipcenter/
- **Xilinx AXI IP website**:
  http://www.xilinx.com/support/documentation/axi_ip_documentation.htm
- *AXI Reference Guide (UG761)*:
  http://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/v14_6/ug761_axi_reference_guide.pdf