Vivado Design Suite User Guide

Design Flows Overview

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/25/12	2012.2	Initial Xilinx release.

Table of Contents

Chapter 1: Introduction

Chapter 2: Design Flow Features

Chapter 3: Basic Design Flow

Appendix A: Additional Resources

Introduction

High-Level Design Flow

The Vivado™ Design Suite offers many ways to accomplish the tasks involved in Xilinx® FPGA design and verification. In addition to the traditional RTL to bitstream FPGA design flow, the Vivado Design Suite provides new system-level integration flows that focus on IP-centric design. Design analysis and verification is enabled at each stage of the flow. Design analysis features include logic simulation, I/O and clock planning, power analysis, timing analysis, design rule checking (DRC), visualization of design logic and implementation results, and programming and debugging.

The entire solution is integrated within a graphical user interface (GUI) known as the Vivado Integrated Design Environment (IDE). The Vivado IDE provides an interface to assemble, implement, and validate the design and the IP. In addition, all flows can be run using the Tcl application programming interface (API). Tcl commands can be interactively entered using the Tcl prompt or saved in a Tcl script. You can use Tcl scripts to run the entire design flow, including design analysis, or to run just part of the flow.

Figure 1-1 shows the high-level design flow in the Vivado Design Suite.
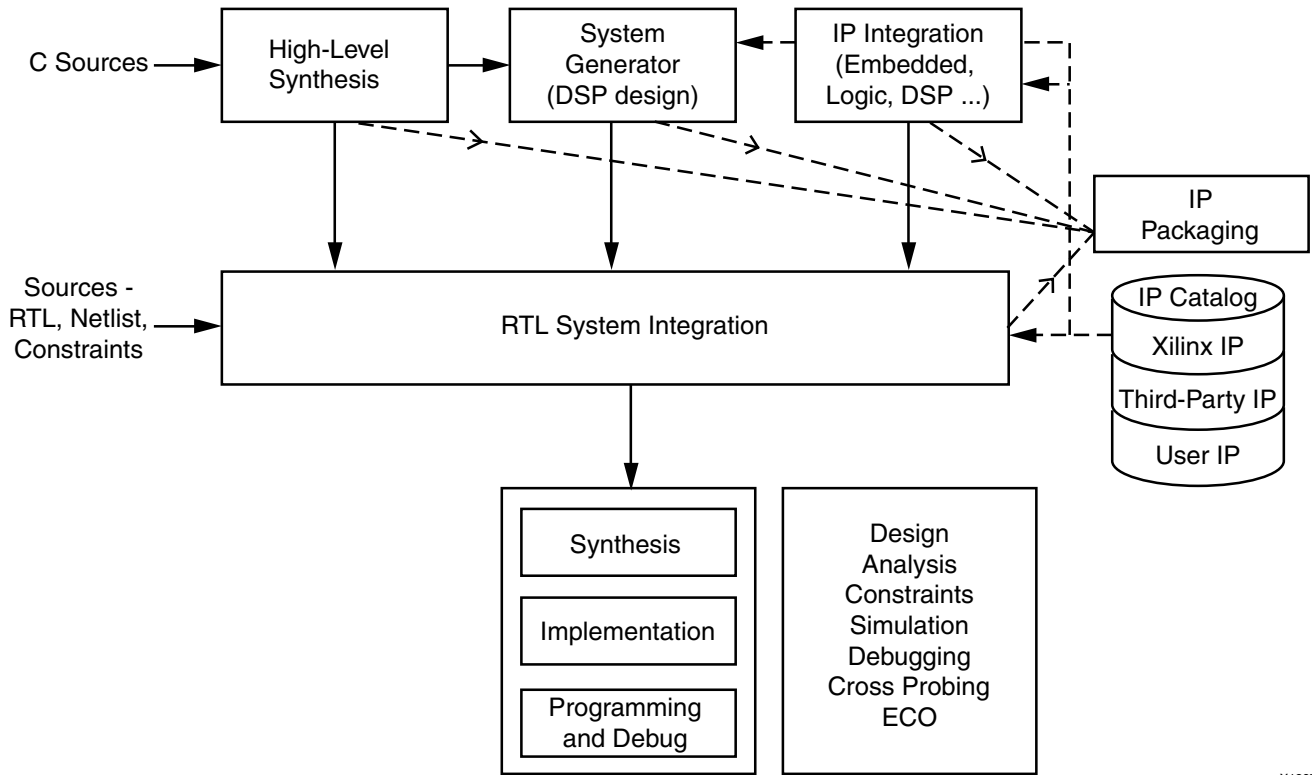
Figure 1-1: **Vivado Design Suite High-Level Design Flow**

Design Flow Features

RTL Development and Analysis

The Vivado™ IDE includes helpful features to assist with RTL development:

- Integrated Text Editor to create or modify source files

- Language templates you can use to copy example logic constructs

- Find in Files feature that lets you search template libraries using a variety of search criteria

When you open an Elaborated RTL Design, the Vivado IDE elaborates the RTL source files and loads the RTL netlist automatically. You can 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 (DRC)

- Behavioral simulation

- Early I/O pin planning using an RTL port list

- Ability to select an object in one view and cross probe to the object in other views, including instantiations and logic definitions within the RTL source files

For more information, see the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

IP Configuration and Implementation

The Vivado Design Suite provides an environment in which you can configure, implement, verify, and integrate IP. The IP can be configured and verified as a standalone module or within the context of the system-level design. The IP can include logic, embedded processors, digital signal processing (DSP) modules, or C-based DSP algorithm designs. Custom IP follows IP-XACT protocol, and is packaged and made available through the Vivado IP Catalog. The IP Catalog enables quick access to the IP for configuration, instantiation, and validation of the IP. Xilinx IP uses the AMBA AXI4 interconnect standard to enable faster system-level integration. Existing IP can be used in the design either as RTL or a netlist. In addition, the Vivado IDE accepts previously created CORE Generator™ tool cores (.xco file extension). For more information, see the *Vivado Design Suite User Guide: Designing with IP (UG896)*.

Logic Simulation

The Vivado simulator, integrated into the Vivado IDE, allows you to simulate the design, add and view signals in the waveform viewer, and examine and debug the design as needed. The Vivado simulator can perform behavioral simulation of designs, and full timing simulation of implemented designs. For more information, see the *Vivado Design Suite User Guide: Logic Simulation (UG900)*.

I/O Pin Planning

The Vivado IDE 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 Vivado IDE provides display windows and tables in which to analyze and design package, and design I/O-related data. For more information, see the *Vivado Design Suite User Guide: I/O and Clock Planning (UG899)*.

Logic Synthesis

The Vivado Design Suite lets you configure, launch, and monitor synthesis runs using Vivado synthesis. The Vivado Design Suite displays the synthesis results and creates report files that you can access. You can select synthesis warnings and errors from the Log window 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. With multiple synthesis runs, Vivado synthesis creates multiple netlists that are stored with the Vivado Design Suite project. You can import any version of the netlist into the Vivado Design Suite environment to perform device and design analysis. You can also create constraints for I/O pin planning, floorplanning, and implementation. The most comprehensive list of Design Rule Checks (DRC) is available after a synthesized netlist is produced, when clock and clock logic are available for analysis and placement.

For more information, see the *Vivado Design Suite User Guide: Synthesis (UG901)*.

Netlist Analysis and Constraints Definition

The Vivado Design Suite enables you to perform design analysis and assign constraints. To identify design issues early, you can perform timing analysis, which includes timing simulation, resource estimation, connectivity analysis and DRCs, prior to implementation. The design data is presented in different forms in different windows, and you can cross probe and coordinate data between windows.

The Vivado IDE, you can analyze device resources in the interactive, graphical windows of the internal die and the external package. You can also apply and analyze timing and physical constraints. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

Implementation

The Vivado Design Suite allows you to configure, launch, and monitor implementation runs. You can experiment with different implementation options and create reusable strategies for implementation runs. For example, you can create strategies for quick runtimes, performance, or area optimization. The implementation run results display interactively, and report files are accessible.

You can launch multiple implementation runs either simultaneously or serially. In Linux, you can use remote servers. You can create constraint sets to experiment with various logical constraints, physical constraints, or alternate devices. For more information, see the *Vivado Design Suite User Guide: Implementation (UG904)*.

Implementation Results Analysis and Floorplanning

You can open the various run results for analysis, constraints assignment and floorplanning. When you open an implemented design, the Vivado IDE displays the netlist, constraints, and implementation results. You can open multiple runs simultaneously. You can also launch the Vivado Simulator to run timing simulation. You can perform timing analysis, power estimation and power analysis, and examine or modify implementation results and design configuration.

In addition, you can visually analyze the placement, routing and timing results, and perform minor placement and routing modifications in the Device window. You can also modify design configuration, such as LUT equations, RAM initialization, and PLL configuration. You can also floorplan the design to provide the implementation tools with options for better or more consistent performance.

For more information, see the *Vivado Design Suite User Guide: Implementation (UG904)* and *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

Device Programming, Verification, and Debugging

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

You can configure and implement IP debug cores, such as the Integrated Logic Analyzer (ILA) and Integrated Controller (ICON) in either an RTL or synthesized netlist. Opening the synthesized design in the Vivado IDE allows you to select and configure the required probe signals into the cores. You can launch the Vivado logic analyzer on any run that has a completed bitstream file.

You can launch the Vivado logic analyzer directly from the Vivado IDE for further analysis of the routing or device resources. For more information, see the *Vivado Design Suite User Guide: Programming and Debugging (UG908)*.

Basic Design Flow

Understanding Project and Non-Project Modes

The Vivado™ Design Suite provides flexibility to enable you to interact with the tools in a way you prefer. Not all designers use the tools the same way. Some users prefer to interact with a GUI and to have the software tool automatically manage their design process and design data. Others prefer more of a script-based compilation style flow where they elect to manage sources and the design process themselves. The Vivado IDE supports both very well. In order to manage the design process, a Vivado project is used. When using a project, a directory structure is created on disk in order to manage design source files, run results, and view project status. This automated management of the design data, process, and status requires a project infrastructure. For this reason, we refer to that flow as the project-based flow, or *Project Mode*.

The compilation style flow is referred to as the non-project batch flow, or *Non-Project Mode*. Sources are read in from their current locations and the design is compiled through the flow in memory. You must run each design step individually and set design parameters and implementation options using Tcl commands. You can save design checkpoints and create reports at any stage of the design process using Tcl. You can open the Vivado IDE at each design stage for design analysis and constraints assignment. You are viewing the active design in memory, so any changes are automatically passed forward in the flow. You can save updates to new constraint files or design checkpoints. Some of the features of the Project Mode such as cross probing, design status, and IP integration are not available in Non-Project Mode.

You can execute the design flow using either of the two modes, which are described in further detail below.

Project Mode Overview

Project Mode is the easiest way to get acquainted with the tool behavior and Xilinx® recommendations. In this mode, you use the Vivado IDE to automatically manage your design and design process. You can use the Flow Navigator within the Vivado IDE (Figure 3-1) to launch predefined design flow steps, such as synthesis and implementation. The Vivado IDE creates a directory structure and automatically manages your design, including management of source files, constraints, intellectual property (IP) data, synthesis and implementation run results, and reports. The Vivado IDE also manages and reports on the status of the source files, configuration, and the state of the design. In addition, this mode includes the following features:

- Integrated IP design.

- Creation of multiple runs to configure and explore available constraint or command options.

- Cross probing from implementation results to RTL source files.

- Non-blocking GUI operation; that is, after launching design flow steps, you can use the GUI for visualization, analysis, or launching other jobs in parallel.

Note: Alternatively, you can use Tcl commands to run Project Mode and use the Vivado IDE only when desired.
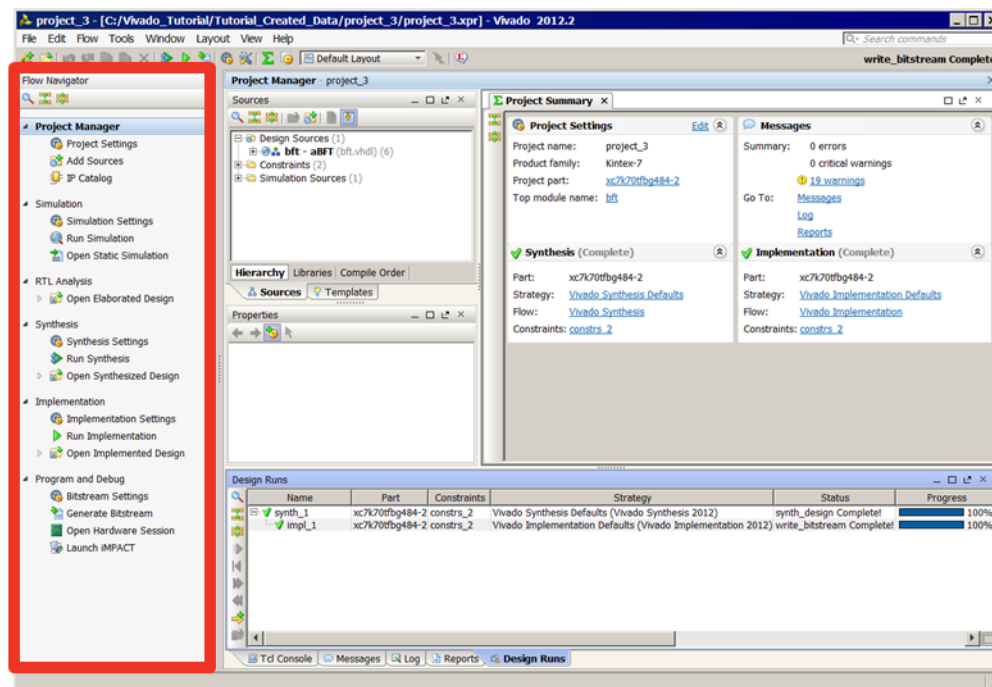
Figure 3-1: **Flow Navigator in the Vivado IDE**

Project Mode Advantages

Using Project Mode has a number of advantages:

- Vivado automatically manages your design. Specifically, it does the following:
  - ◦ Automatically manages Project status, HDL sources, constraint files, and IPs
  - ◦ Generates and stores synthesis and implementation results
  - ◦ Has advanced design analysis capabilities including cross-probing from implementation results to RTL source files
  - ◦ Enables you to easily create multiple runs to configure and explore available constraint or command options.
- You can equitably use the GUI, Tcl commands, or a mix of the two to control design implementation and analysis features, depending on your preferences.
- In GUI mode, you can use Flow Navigator (Figure 3-1) to launch predefined design flow steps, such as synthesis and implementation, and use various design analysis features. The GUI mode is the easiest way to get acquainted with the tool behavior and Xilinx recommendations.

Non-Project Mode Overview

In this mode, you use Tcl commands to compile a design through the entire flow in memory. Tcl commands provide the flexibility and power to set up and run your designs as well as perform analysis and debugging. Tcl commands can be run in batch mode, from the Tcl Prompt, or through the Vivado IDE Tcl Console.

This mode enables you to have full control over each design flow step, but you must manually manage source files, reports, and intermediate results known as design checkpoints. You can generate a variety of reports, perform design rule checks (DRCs), and write design checkpoints at any stage of the implementation process.

Unlike Project Mode, this mode does not include features such as runs infrastructure, source file management, cross probing, or design state reporting. Each time a source file is updated, you must rerun the design manually. There are no default reports or intermediate files created in this mode. However, you can create reports and design checkpoints using Tcl commands if desired.

In Non-Project Mode, you can open the design saved in memory at any stage in the Vivado IDE to analyze the design or update the netlist or constraints. The Vivado IDE does not include Project Mode features such as the Flow Navigator, IP Catalog, Sources window, Message and Reports, and so forth. You cannot access or modify source files or runs in the Vivado IDE. Note that Tcl commands are blocking; that is, you must wait until the commands or scripts complete before you can use the Vivado IDE.

Non-Project Mode Advantages

The main advantage of using Non-Project Mode is that it enables you to have full control over each design flow step.

Along with full freedom to manage your design, you also have the responsibility to manage your design, including:

- Managing HDL Source files, constraints, and IPs
- Managing dependencies
- Generating and storing synthesis and implementation results

In this mode, you can do the following:

- Use Tcl commands to compile a design through the entire flow.
- Use Tcl commands or the GUI for design analysis and report generation.

Note: Cross probing from implementation results to RTL source files is not available in this mode.

Unlike Project Mode, this mode does not include features such as runs infrastructure, source file management, or design state reporting. Each time a source file is updated, you must manually rerun the design.

In addition, there are no default reports or intermediate files created in this mode. You must manually generate them.

Project and Non-Project Mode Differences

There are substantial feature and command differences between the two modes.

Feature Differences

The project infrastructure enables the Vivado IDE to track the history of the design as well as store pertinent design information. However, the tool flow is somewhat fixed, in that launching a run only generates a factory standard set of report files. Some of the Vivado Design Suite features that are only available when using Project Mode are listed below.

- Source file management and status

- Flow Navigator and Project Summary

- Consolidated Messages and automatically generated standard reports

- Cross probing back to RTL

- Storage of tool settings and design configuration

- Experimentation with multiple synthesis and implementation runs

- Use and management of Constraint Sets

- Run results management and status

- IP configuration and integration with the IP Catalog

When using Non-Project Mode, everything is individually done using Tcl commands. All of the processing is done in memory, so no files or reports are generated automatically. Each time you compile the design, it is your responsibility to define all of the sources, set all tool and design configuration parameters, launch all Tcl commands, and generate your desired report files. Because no project is created on disk, source files remain in their original locations and run output is only created where you specify. This flow provides you with all of the power of the Tcl API and full control over the entire design process.

Table 3-1 summarizes the feature differences between Project and Non-Project Modes.

Table 3-1: **Project Mode versus Non-Project Mode**

Flow Element	Project Mode	Non-Project Mode
Design Source File Management	Automatic	Manual
Flow Navigation	Guided	Manual
Flow Customization	Limited	Unlimited
Reporting	Automatic	Manual
GUI Operations	Non-blocking	Blocking
Analysis Stages	Designs only	Designs and checkpoints

Command Differences

The Project Mode Tcl commands are substantially different than the individual commands used in the Non-Project Mode. Some major differences are:

- You add sources to the project for management using the `add_files` Tcl commands. They can be copied into the project to maintain a separate version of the sources within the project directory structure.

- You use a runs infrastructure to manage the automated synthesis and implementation process, and to track run status. Wrapper commands around the individual synthesis and implementation commands are used for this purpose. They are referred to as the `launch_runs` commands.

Figure 3-2 shows the difference between Project Mode and Non-Project Mode Tcl commands. Project Mode includes GUI operations, which result in a Tcl command being executed in most cases. The Tcl commands appear in the Vivado IDE Tcl Console and are also captured in the `vivado.jou` file. You can use this file to develop scripts for use with either mode.
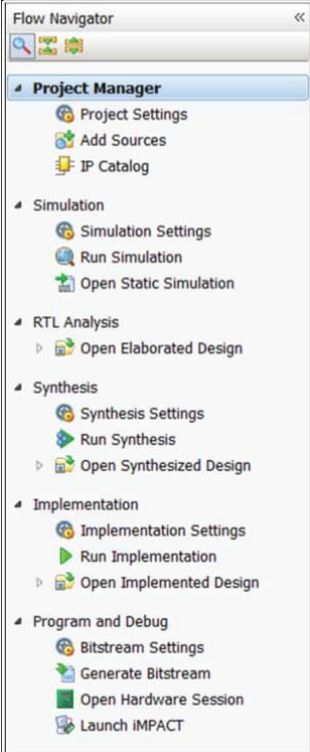

Figure 3-2: **Project and Non-Project Mode Commands**

Using Project Mode

Project Mode is fully supported for both GUI and script-based users. You can alternate between using the IDE and the Tcl Console for any given project. However, the features of the Project Mode work very well when used with the Vivado IDE. As projects are opened or created in the Vivado IDE, you are presented with the current state of the design, run results, and previously generated reports and messages. You can create or modify sources, apply constraints and debug information, configure tool settings, and perform design tasks.

Using the Flow Navigator, you can walk through the entire design flow. When you click **Generate Bitstream**, the Vivado IDE synthesizes and implements the design and generates a bitstream file. Designs can be opened after RTL elaboration, synthesis and implementation for analysis, and constraints application. Opening a design loads it into memory by compiling the netlist and constraints against the target device. After the design is loaded in memory, a wide range of analysis and reporting features of the die are available.

The design can be analyzed from a variety of different criteria and viewpoints. Constraint and design changes can be applied and saved. Refer to Opening Designs or to the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

Creating Projects

The Create Project wizard walks you through the process of creating your project. You are presented with various dialog boxes to define the project. They include the project name and location to store the project, project type (for example, RTL, Netlist, and so forth), and the target part. Various types of sources can be added such as RTL, IP, XDC/SDC constraints, simulation test benches, DSP modules from System Generator (XMP) or Vivado High-Level Synthesis (HLS), processor modules from Xilinx Platform Studio (XPS), memory initialization files from XPS (.bmm extension files), and design documentation. When you select sources, you can determine whether to reference the original source in its location or to copy them into the project directory. The Vivado IDE tracks the time/date stamp of each file and report status. If files are modified, you are alerted to the source and/or design status being out-of-date. For more information, see the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

Understanding the Flow Navigator

The Flow Navigator provides control over the major design process tasks, such as project configuration, synthesis, implementation, and bitstream generation. The commands and options available in the Flow Navigator depend on the status of the design. Inapplicable steps are grayed out until the appropriate design tasks are completed, as shown in Figure 3-3.
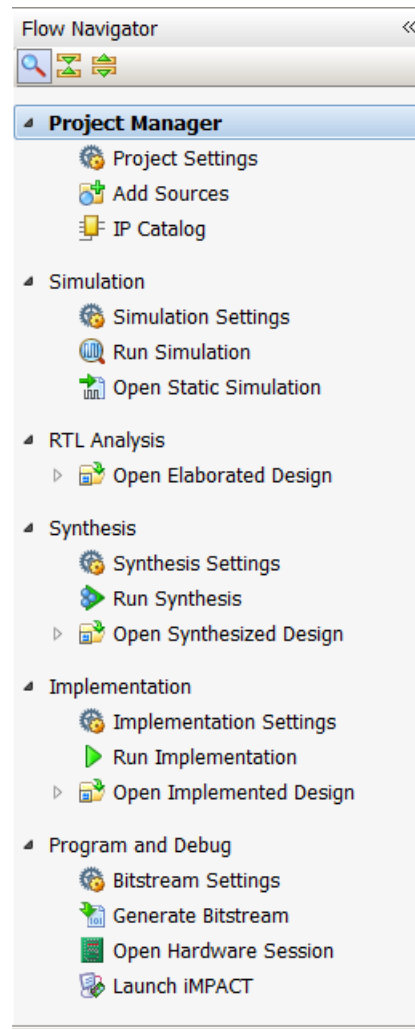
Figure 3-3: **Flow Navigator**

As these design tasks are completed, you can open the resulting designs to analyze results and apply constraints. To do so, in the Flow Navigator, click **Open Elaborated Design**, **Open Synthesized Design**, or **Open Implemented Design**. For more information, refer to Opening Designs.

Each of these designs displays a set of commonly used commands for the applicable phase of the design flow. Selecting any of these commands in the Flow Navigator opens the design, if it is not already opened, and performs the operation.

For example, Figure 3-4 shows the commands related to synthesis.
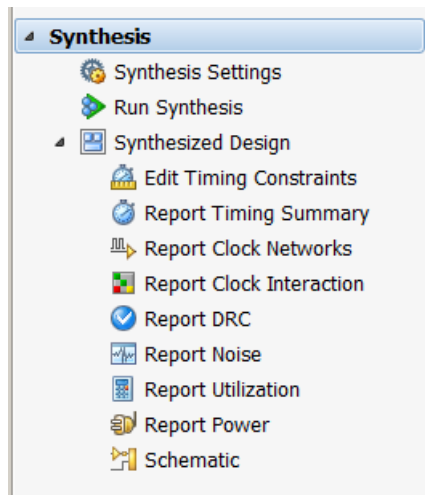
Figure 3-4: **Synthesis Section in the Flow Navigator**

Using the Flow Navigator with an RTL Project

Figure 3-5 illustrates the design flow using RTL sources as input to the Vivado Design Suite.
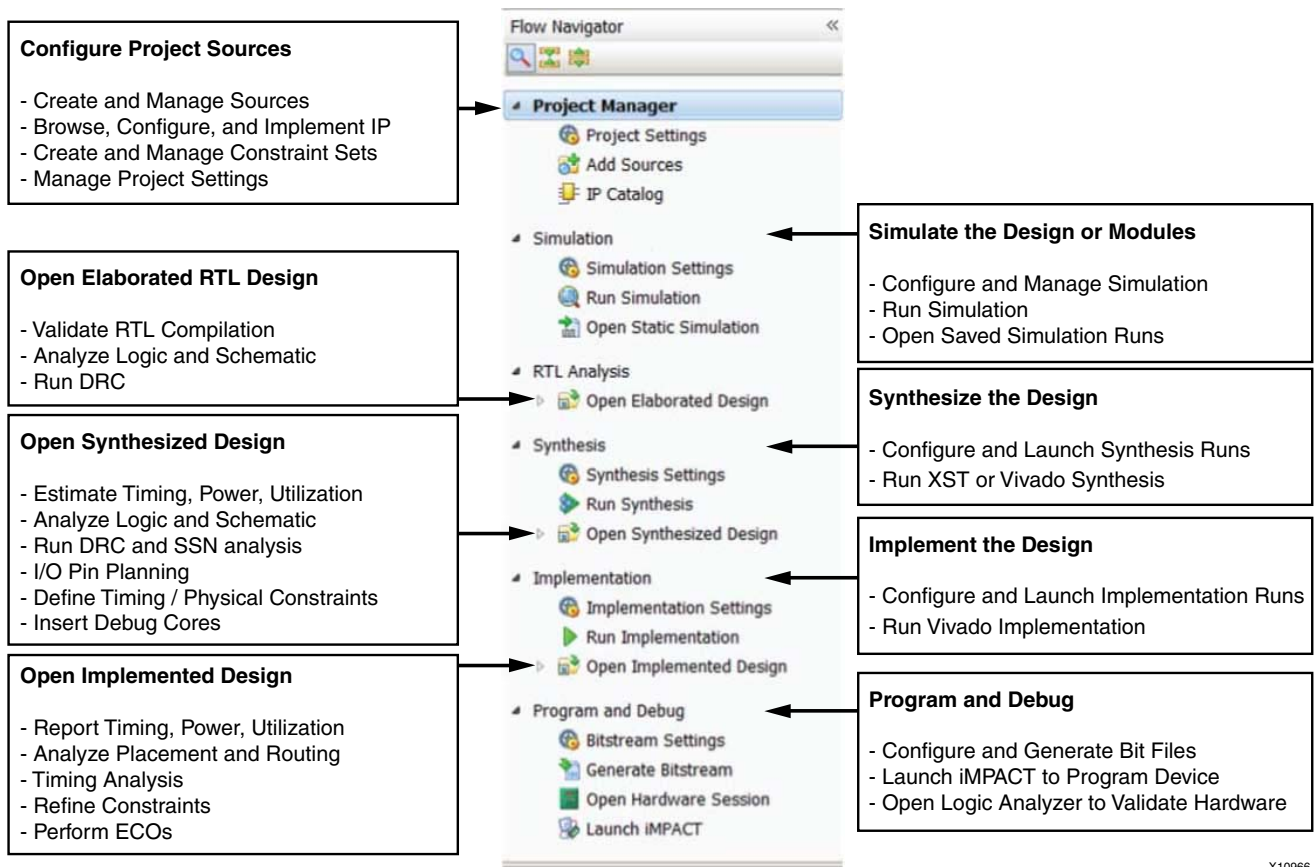
Figure 3-5: **Flow Navigator for RTL Project**

www.xilinx.com

Using the Flow Navigator with a Synthesized Netlist Project

Figure 3-6 illustrates the design flow for synthesized netlist-based projects.

Open Synthesized Design

- Estimate Timing, Power, Utilization
- Analyze Logic and Schematic
- Run DRC and SSN analysis
- I/O Pin Planning
- Define Timing / Physical Constraints
- Insert Debug Cores

Open Implemented Design

- Report Timing, Power, Utilization
- Analyze Placement and Routing
- Timing Analysis
- Refine Constraints
- Perform ECOs

Manage the Project

- Create and Manage Sources
- Browse, Configure, and Implement IP
- Create and Manage Constraint Sets
- Manage Project Settings

Implement the Design

- Configure and Launch Implementation Runs
- Run Vivado Implementation

Program and Debug

- Configure and Generate Bit Files
- Launch iMPACT to Program Device
- Open Logic Analyzer to Validate Hardware

X12975

Figure 3-6: **Flow Navigator for a Synthesized Netlist Project**

IP Configuration and Verification

Using the IP Catalog

The IP Catalog displays all available Xilinx LogiCORE™ IP, as well as any user IP or third-party IP added using the **Update IP Catalog** command. The catalog is categorized and displays useful IP type, version and license information. You can do the following in the IP Catalog:

- Double-click an IP to invoke the Customize IP dialog box.

- Select **Generate IP** to create a new IP source in the Source window. An instantiation template is created to enable integration into the system-level design. Vivado Design Suite IP is created as RTL source files, not netlists.

- Select **Synthesis** and **Implementation** to implement the IP along with the rest of the design.

Validating and Reusing IP

In some cases, you might want to validate the performance of standalone IP. Running synthesis, simulation, implementation, and analysis on the standalone IP ensures that IP performance targets are hit. You can also ensure that the validated version of the IP netlist is not re-synthesized with the rest of the design. To do this, set the Top Module Name in the project to the module of the IP or you can create a separate project with the IP only and run it through the flow. The Synthesis out-of-context parameter can be used to ensure I/O buffers do not get instantiated in synthesis. Use commands in the Flow Navigator to implement and validate the IP.

You can add sources for validated standalone IP to a new project. If the standalone IP has been synthesized, the netlist is used as the design source, thus ensuring that the validated IP netlist is preserved. If an existing standalone IP modules, without Synthesis results (NGC or EDIF netlist), is added to a project, the IP is synthesized along with the rest of the design during Run Synthesis.

An example design is optionally created as a part of the IP core generation. Setting the Vivado IDE top module name to match the example IP module enables you to verify the standalone IP within the context of the design project. If the synthesized netlist exists in the example design directory, it is used during synthesis of the top-level design. That is, the IP netlist that you validated standalone is not re-synthesized along with the rest of the design.

Creating and Managing Runs

Configuring Synthesis and Implementation Runs

Various settings are available to control the behavior of synthesis and implementation. When using Project Mode, these settings are passed to runs using run strategies. A run strategy is simply a saved set of run configuration parameters. Xilinx supplies several run strategies for running synthesis and implementation. Custom run settings can also be applied.

Configuring Synthesis

To modify Synthesis Project Settings, do one of the following:

- In the Flow Navigator, select **Synthesis Settings**.
- In the Flow Navigator, from the Project Manager section, select **Project Settings > Synthesis**.
- Select **Tools > Project Settings > Synthesis**.
- Select the Synthesis related links in the Project Summary.
- In the Design Runs window, select a synthesis run, right-click, and select **Change Run Settings**.
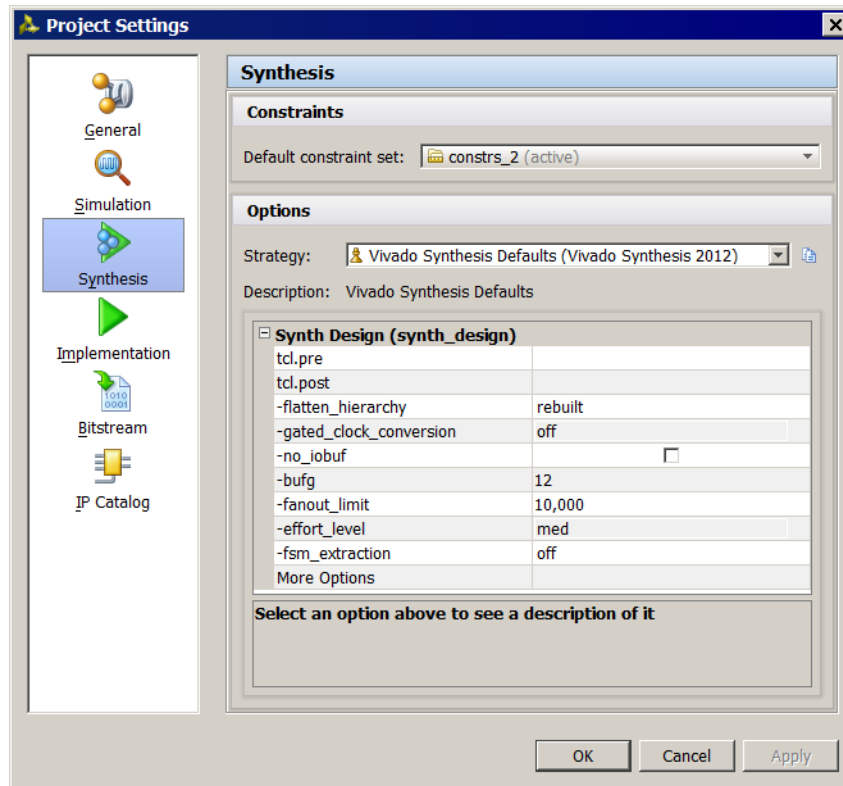
Figure 3-7: **Setting Synthesis Project Settings**

To create custom strategies, select the desired settings, right-click, and select **Save Strategy As**.

You can select constraint sets for Vivado synthesis runs. Vivado synthesis is timing driven and will utilize the timing constraints to maximize performance. Separate constraint sets can be used for synthesis and implementation. For more information, see *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* and *Vivado Design Suite User Guide: Using Constraints (UG903)*.

Running XST

The Vivado IDE allows you to use the ISE Design Suite XST synthesis tool or Vivado synthesis. The XST strategy launches the runs using XST synthesis. Either result uses Vivado implementation.
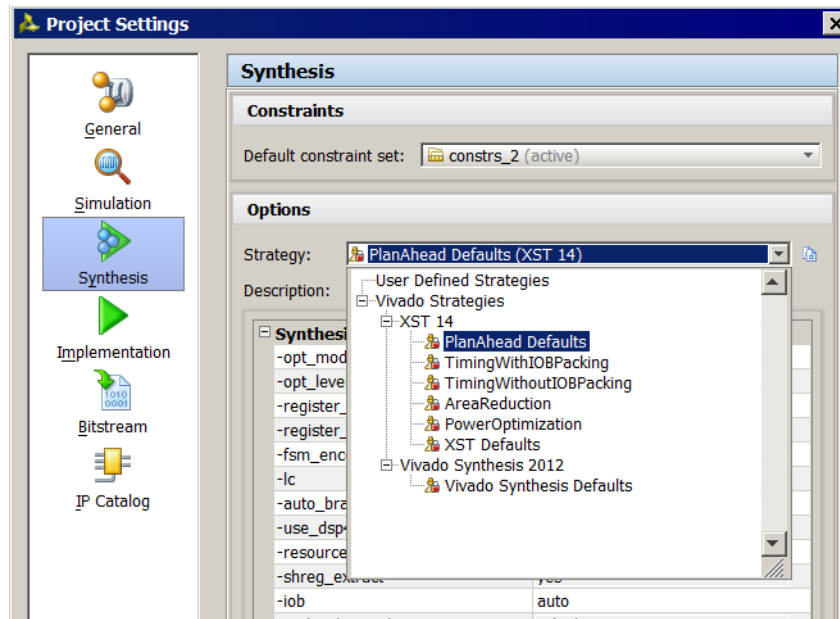
Figure 3-8: **Selecting XST Synthesis Strategies**

Configuring Implementation

To modify Implementation Project Settings, do one of the following:

• In the Flow Navigator, select **Implementation Settings**.

• In the Flow Navigator, from the Project Manager section, select **Project Settings > Implementation**.

• Select **Tools > Project Settings > Implementation**.

• Select the Implementation related links in the Project Summary.

• In the Design Runs window, select a run, right-click, and select **Change Run Settings**.
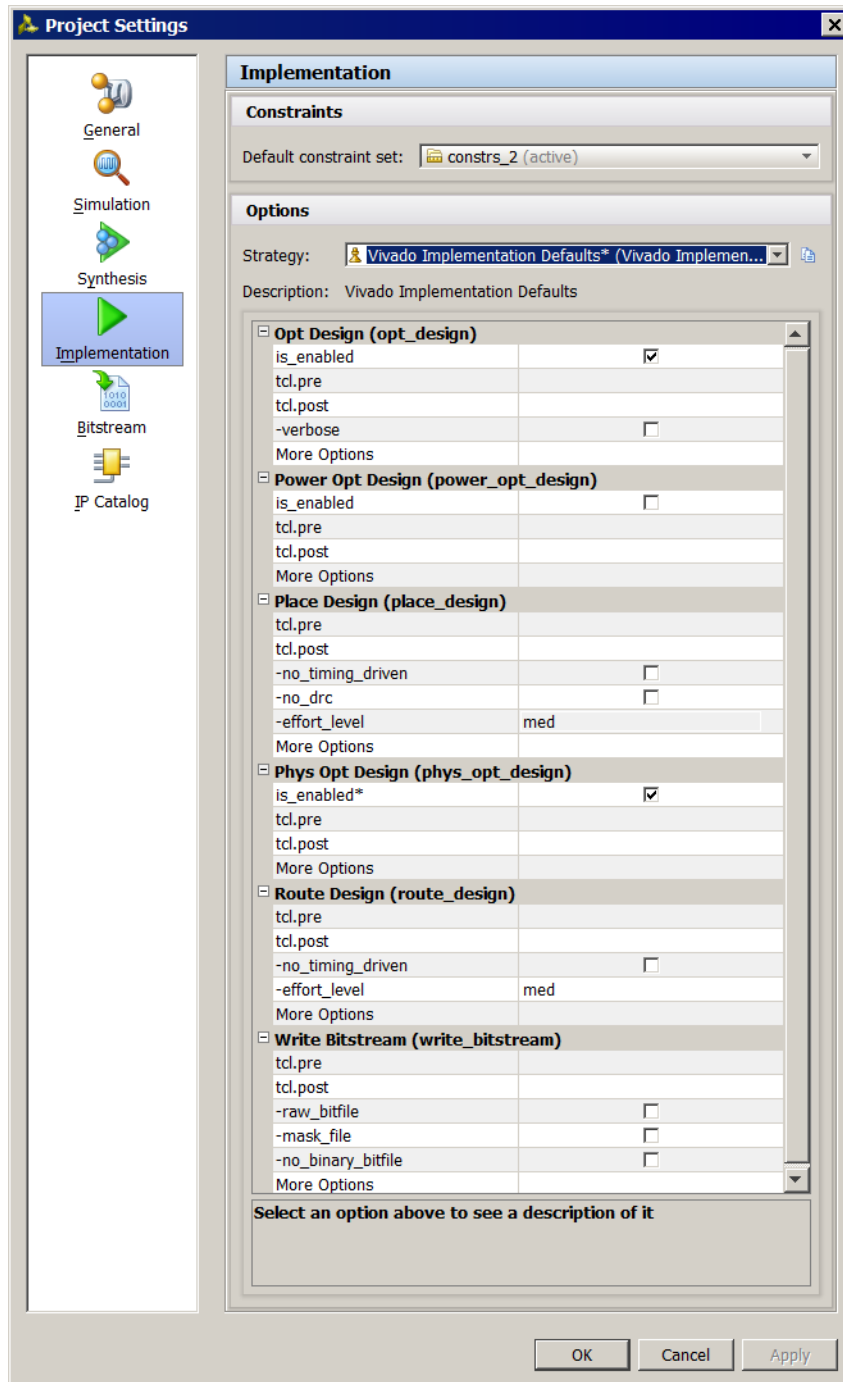
*Figure 3-9:*   **Setting Implementation Project Settings**

In the Implementation Project Settings dialog box, select the constraint set and strategy to apply to the run. Custom strategies are created when you select the desired settings and then right-click and select **Save Strategy As**.

www.xilinx.com

You can use separate constraint sets for synthesis and implementation. To determine what constraints files are used for synthesis or implementation, select the constraint file in the Sources window and view the Source File Properties window. The Used In section can be adjusted to control constraints assignment to Runs. For more information, see *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* and *Vivado Design Suite User Guide: Using Constraints (UG903)*.
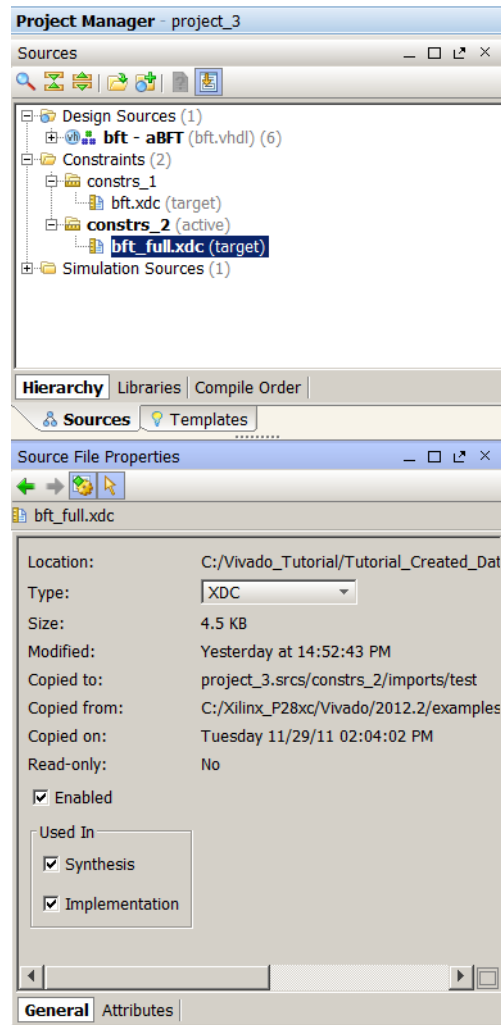
Figure 3-10: **Selecting Constraints for Synthesis and Implementation**

The active constraint set is displayed in bold in the Sources window. This designates the constraint set as the default one used for open runs or designs. A constraint set can be made active using the **Make Active** popup menu command.

Launching Synthesis and Implementation Runs

After the synthesis and implementation settings are configured in the Project Settings dialog box, you can launch synthesis or implementation runs as follows:

1. In the Flow Navigator, select **Run Synthesis**, **Run Implementation**, or **Generate Bitstream**.

2. In the Design Runs window, select a run, right-click, and select **Launch Selected Run**. Alternatively, you can click the **Launch Selected Run** button.

3. Select **Flow > Run Synthesis**, **Flow > Run Implementation**, or **Flow > Generate Bitstream**.

Creating and Managing Multiple Runs

You can create multiple synthesis or implementation runs in order to experiment with constraints or tool settings. To create multiple runs:

1. In the Flow Navigator, right-click **Synthesis** or **Implementation**.

2. Select **Create Synthesis Runs** or **Create Implementation Runs**.

3. In the Create New Runs wizard, select the constraint set and target part.

If more than one synthesis run exists, the netlist can also be selected when creating implementation runs. You can then create one or more runs with varying strategies.
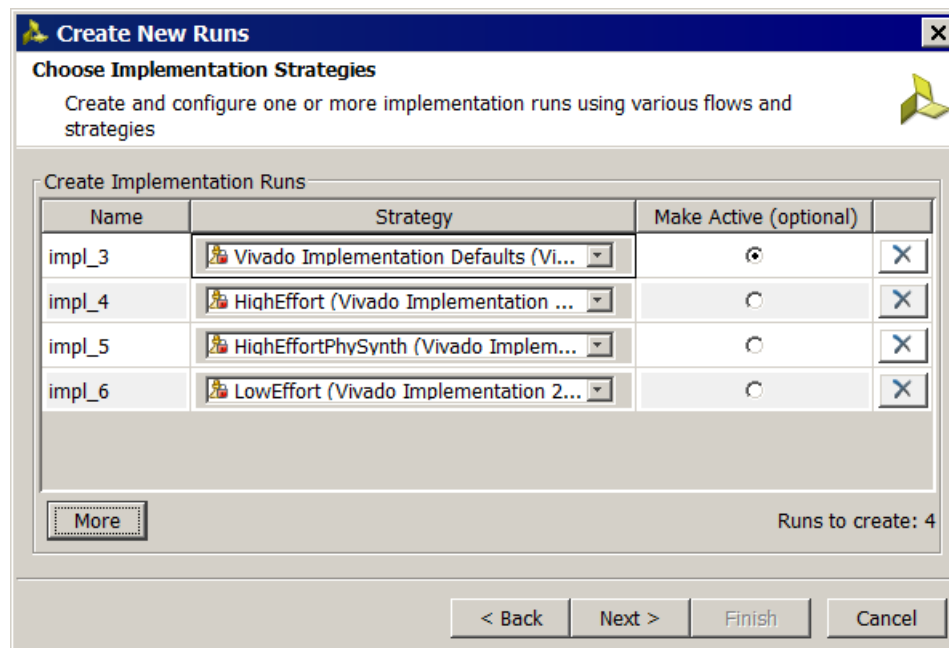

Figure 3-11:   **Creating Multiple Synthesis or Implementation Runs**

There are several launch options available when multiple runs exist. Selected runs can be launched sequentially or in parallel on multiple local processors. Remote hosts can be configured and used for Linux only.

Managing Runs with the Design Runs Window

Run status and information is displayed in the Design Runs window. Many run management commands are available in the popup menu.

You can manage multiple runs from the Design Runs window. When multiple runs exist, the active run is displayed in bold. The Vivado IDE displays the design information for the active run. The Project Summary, reports, and messages all reflect the results of the active run.

The Vivado IDE opens the active design by default when you select **Open Synthesized Design** or **Open Implemented Design** in the Flow Navigator. You can make a run the active one using the **Make Active** popup menu command. The Vivado IDE updates results to reflect a change in the designated active run. Double-click on any synthesized or implemented run to open the design in the Vivado IDE.
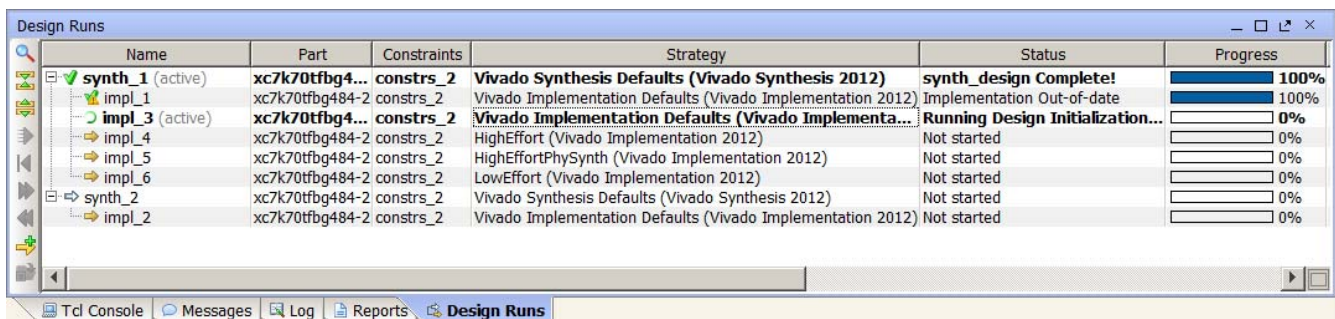
Figure 3-12:   **Managing Runs with the Design Runs Window**

Viewing Messages

When using Project Mode, all of the Vivado IDE messages are presented in the Messages window and are categorized according to severity level: Errors, Critical Warnings, Warnings, and Info. To filter messages, select the appropriate check boxes in the window header. You can expand the Message categories to view specific messages. Many messages include links that take you to logic lines in the RTL files.

Viewing Reports

Several standard reports are generated as a part of the `launch_runs` Tcl commands. They are presented in the Reports window. Double-click any report to display it in the Text Editor. You can create custom reports using Tcl commands in the Tcl Console.
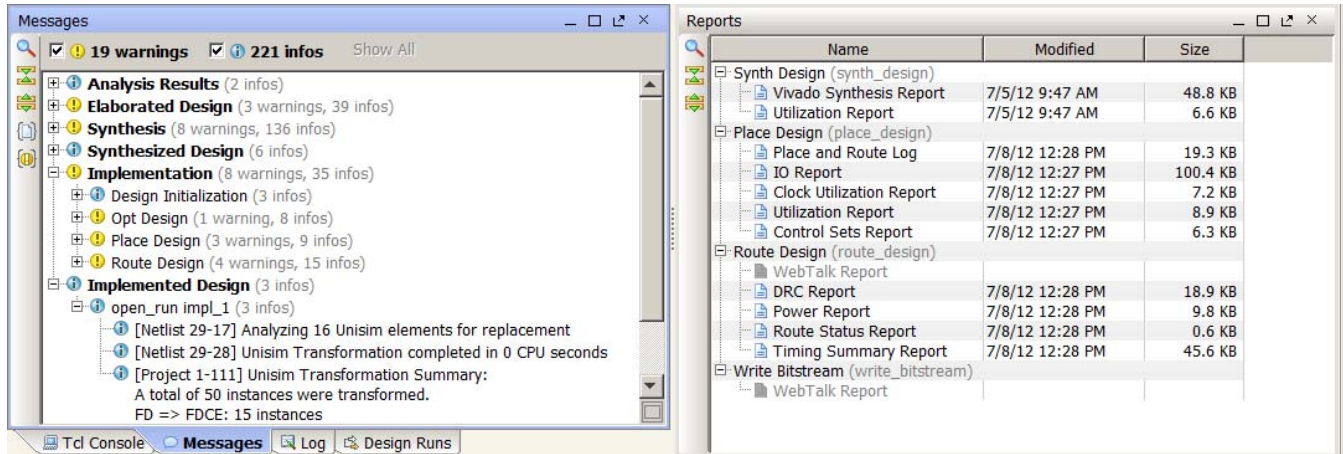
Figure 3-13: **Viewing Messages and Reports**

Opening Designs

You can open the design after RTL elaboration, synthesis, or implementation to analyze the design or apply constraints. Opening a design loads it into memory after compiling the netlist and constraints against the target device. After that, a wide range of analysis and reporting features are available in the Vivado IDE.

The Vivado Design Suite uses a *design* that you can open, save and close. When you open a new design, you are prompted to close any previously opened designs in order to preserve memory. However, you are not required to close the designs, because multiple designs can be opened simultaneously. Note that if constraint changes are made while the design is open, you are prompted to save the changes to the original XDC source files or to create a new constraint set. For more information, see *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* and *Vivado Design Suite User Guide: Using Constraints (UG903)*.

For more information about the types of design activities that can be performed on open designs, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

Opening an Elaborated RTL Design

The Vivado Design Suite opens an elaborated RTL Design by compiling the RTL netlist and applying physical and timing constraints against a target part. When opened, the different elements of the elaborated RTL Design are loaded into memory, and you can analyze and modify them as needed to complete the design.

When you select **Open Elaborated Design** in the Flow Navigator, the Vivado IDE compiles the RTL design using the top module name defined. A basic RTL elaboration is performed, the constraints are applied, and the design is loaded into memory. Compilation messages and errors are reported in the Messages window, and links in the messages open RTL source files at specific lines of code. For more information, see the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The RTL design enables you to analyze your design for logic correctness. You can check that the logic compiles properly, no modules are missing, and no interface mismatches can be found. Links in messages in the Messages window allow you to display the problem lines in the RTL files in the Text Editor. The Schematic window displays RTL interconnects using RTL-based logic constructs. There is no FPGA technology mapping during RTL elaboration. The Schematic window allows you to explore the logic interconnects and hierarchy in a variety of ways. You can select logic in the Schematic window and see specific lines in the RTL files in the Text Editor. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

Constraints that are defined on specific logic instances within the logic hierarchy, such as registers, might not be resolvable during RTL elaboration. The logic names and hierarchy generated during elaboration might not match those generated during synthesis. For this reason, you might see constraint mapping warnings or errors when elaborating the RTL Design, if you have these types of constraints defined. Running Synthesis on the design should not produce those same warnings.

You can interactively configure and assign I/O Ports in the elaborated RTL Design. The I/O planning capabilities of the Vivado IDE can be utilized to interactively assign I/O Ports and run DRCs. When possible, I/O Planning should be performed after synthesis to ensure proper clock and logic constraint resolution. The DRCs performed after synthesis are much more extensive. For more information, see *Vivado Design Suite User Guide: I/O and Clock Planning (UG899)*.

When you select the **Run DRC** command, the Vivado IDE invokes a set of base level RTL and I/O DRCs to identify logic issues such as asynchronous clocks, latches, and so forth.

Opening a Synthesized Design

The Vivado Design Suite opens a Synthesized Design by opening the synthesized netlist and applying physical and timing constraints against 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 them as needed to complete the design. You can save updates to the constraints files, netlist, debug cores, and configuration.

There are many design tasks that you can perform in a Synthesized Design. Early timing, power, and utilization estimates can help you determine if your design is converging on desired targets. The design can be explored in a variety of ways using the windows of the Vivado IDE. Objects are always cross selected in all other windows. Timing constraints can be applied and further timing analysis can be performed. Physical constraints for I/O ports, floorplanning, or design configuration can be interactively defined.

You can cross probe problem lines in the RTL files from various windows such as Messages, Schematic, Device, Package, Find, and so forth. The Schematic window allows you to interactively explore the logic interconnect and hierarchy in a variety of ways. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

You can interactively configure and assign I/O Ports in the Synthesized Design. The I/O planning capabilities of the Vivado IDE can be utilized to interactively assign I/O Ports and run DRCs. For more information, see the *Vivado Design Suite User Guide: I/O and Clock Planning (UG899)*.

Select the **Run DRC** command to invoke a comprehensive set of DRCs to identify logic issues. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

You can configure and implement debug core logic in the Synthesized Design. Signals can be interactively selected in the Schematic or Netlist views for debug. Debug cores are then configured and inserted into the design. The core logic and interconnect is preserved through synthesis updates of the design when possible.

To open a Synthesized Design, use one of the following methods:

• In the Flow Navigator, from the Synthesis section, select **Open Synthesized Design**.

• In the Design Runs view, double-click on the run name.

• Select **Flow > Open Synthesized Design**.

• In the Flow Navigator, from the Synthesis section, select **New Synthesized Design**.

• Select **Flow > New Synthesized Design**.

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 3-14, you can enter the following to define a new Synthesized Design:

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

- **Synthesis Run**: Use the netlist from the specified completed Synthesis run. This field only appears if you have more than one synthesis run defined in the Project.

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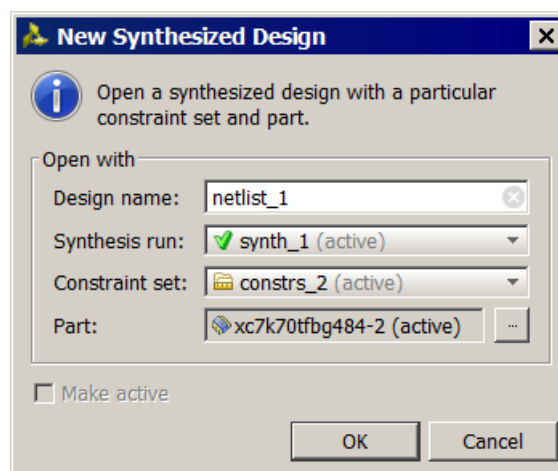
- **Part**: Select a target part.

Figure 3-14: **New Synthesized Design**

Figure 3-15 shows the default view layout of the open Synthesized Design.
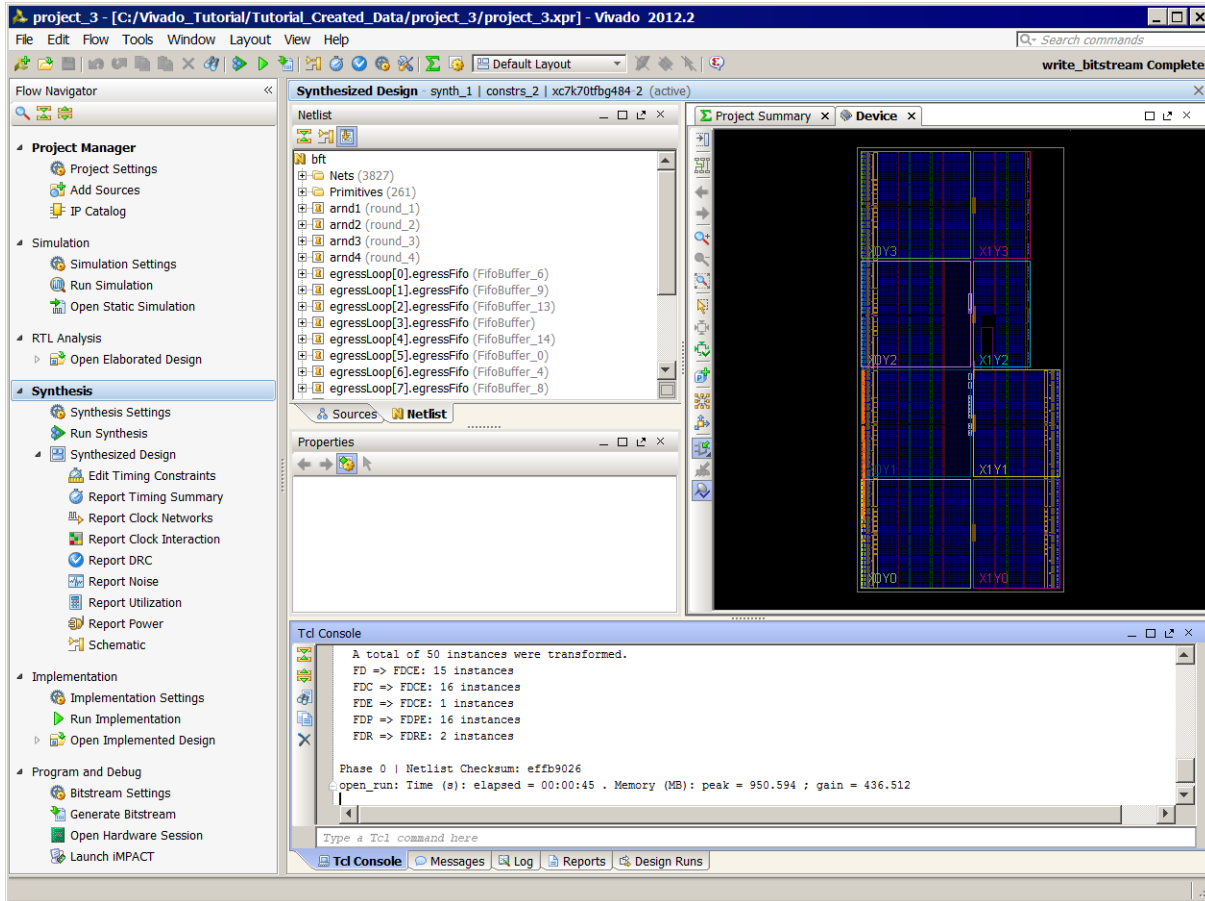
Figure 3-15: **Open Synthesized Design**

Moving Processes to the Background

As the Vivado Design Suite launches the process to open the Synthesized or Implemented Designs, reading design files and constraint files, the Open Synthesized Design dialog box and Open Implemented Design dialog boxes let you put the process into the background, as shown in Figure 3-16.

When you put a process into the background, it releases the Vivado Design Suite to perform certain other functions, like viewing reports or opening design files, while it completes the background task. You can use 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 3-16: **Open Synthesized Design - Background Process**

Using View Layouts

Several default view layouts are supplied to better enable specific design tasks such as I/O Planning, Floorplanning, Debug configuration, and so forth. Changing view layouts simply alters the windows that are presented to better enable that particular design task. Custom view layouts can be created.
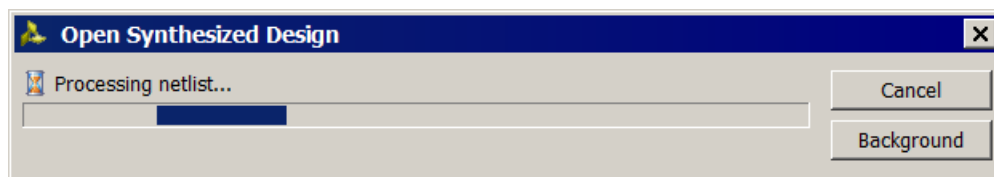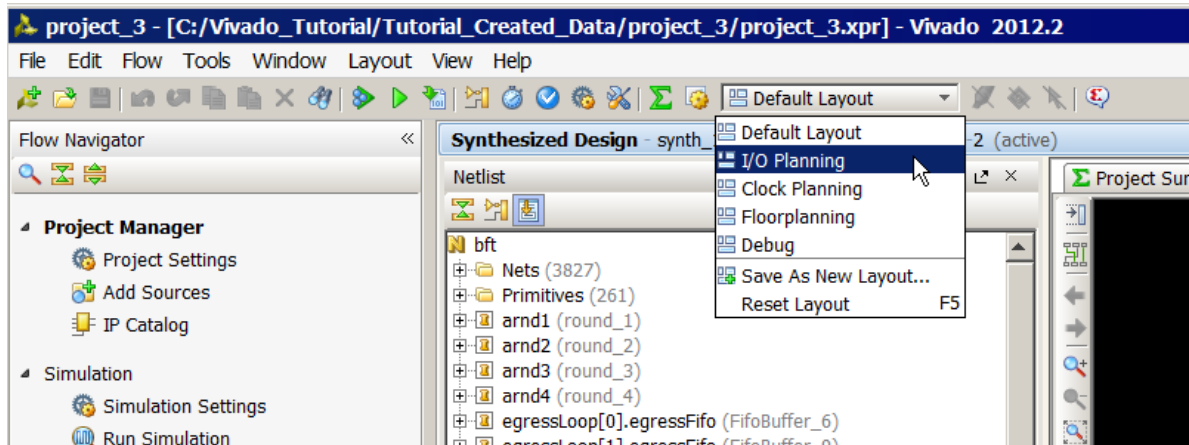

Figure 3-17: **Selecting View Layout**

Opening an Implemented Design

The Vivado Design Suite opens an Implemented Design by opening the implemented netlist, and then applying the physical and timing constraints used during implementation, placement, and routing results against the implemented part. When it is opened, the different elements of the Implemented Design are loaded into memory, where you can analyze them and modify them as needed to complete the design. You can save updates to the constraints files, netlist, implementation results, and design configuration. Because the Vivado Design Suite allows for multiple implementation runs, you can select any completed implementation run to open in the Implemented Design.

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

• In the Flow Navigator, click **Open Implemented Design** in the Implementation section.

• Select **Flow > Open Implemented Design**.

• In the Design Runs view, double-click on the run name.

You can perform many design tasks in an Implemented Design. Actual timing, power, and utilization statistics can help you determine if your design has converged on desired targets. The design can be explored in a variety of ways using the windows of the Vivado IDE. Objects are always cross selected in all other windows. Timing constraints can be manipulated and further timing analysis can be performed. Floorplanning or design configuration constraints can be interactively applied and saved for future runs.

Explore the placement or the routing results by toggling the **Routing Resources** button in the Device Window. Increasing levels of detail are exposed as zoom levels increase. Placement and Routing as well as design configuration such as LUT equations and RAM initialization can be altered interactively. You can cross probe back to problem lines in the RTL files by selecting results in the Device or Schematic windows. The Schematic window allows you to interactively explore the logic interconnect and hierarchy in a variety of ways. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906).*

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

The Implemented Design default view opens. Figure 3-18 shows the default layout view of an open Implemented Design. The Device view might display placement only or routing depending on the state the window was in when it was last closed, as shown below. Click the **Routing Resources** button in the Device window to toggle the view to display only placement or routing.
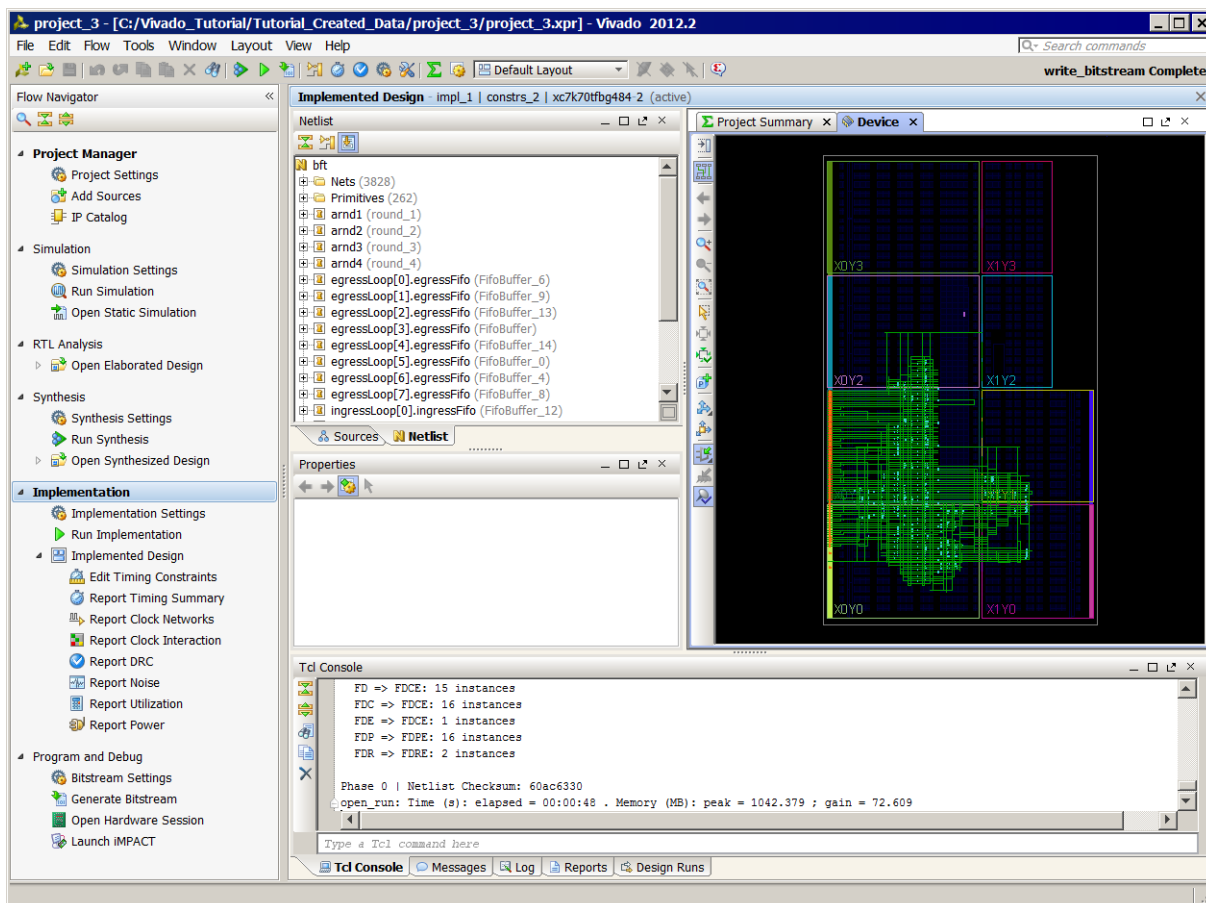

Figure 3-18: **Open Implemented Design**

Managing Multiple Open Designs

As you open designs and the Vivado Design Suite loads the design into memory, you can open multiple Elaborated Designs, Synthesized Designs, and Implemented Designs simultaneously to display the results of different design options and different runs. The only limiting factor is the memory capacity of your machine. When opening a new Design, you are prompted to close the current Design in memory, if you have one open in order to conserve memory. You can also close any open Design by clicking the large Close (**X**) icon in the upper-right corner of the active Design window banner.

The active design being analyzed is identified by highlighting the associated menu in the Flow Navigator. This provides you with a visual reference of the current design. Figure 3-19 shows the highlighted Synthesized Design menu. This indicates that a Synthesized Design is both opened and active in the current session. The Design window banner is also a good indication of which design is open.

The Flow Navigator displays some subtle changes to indicate multiple open Designs. The command name text changes from **Open Synthesized Design** to **Synthesized Design** and the icon changes.

When multiple designs are open, the Flow Navigator provides multiple indications of the open designs. In Figure 3-19 the Implemented Design command under the Implementation menu reflects the fact that an Implemented Design is open, but is not the active design. Selecting the Implemented Design command makes the open design the active design, and moves the highlighting from **Synthesis** to **Implementation**.
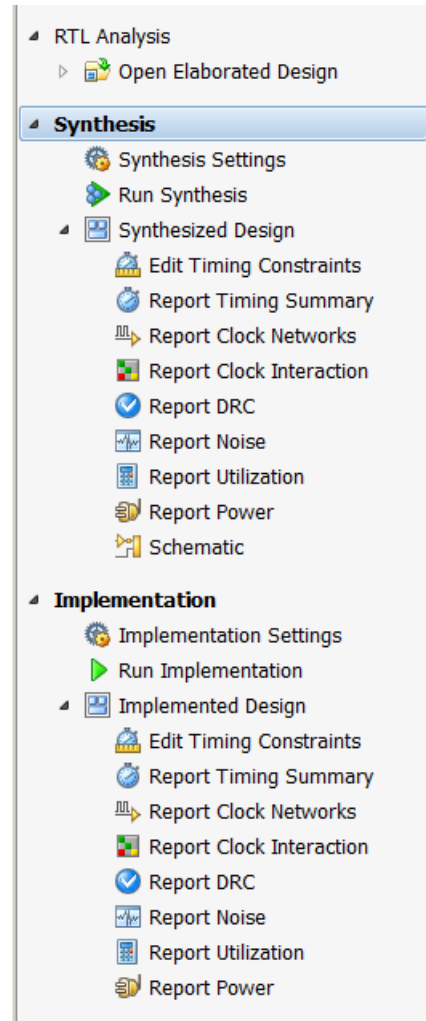
Figure 3-19:   **Open Designs in Flow Navigator**

The Flow Navigator also supports a popup menu accessed with the right mouse button. Right-click on the RTL Analysis, Synthesis, or Implementation commands in the Flow Navigator, and a popup menu is displayed. Figure 3-20 shows the Synthesis popup menu.
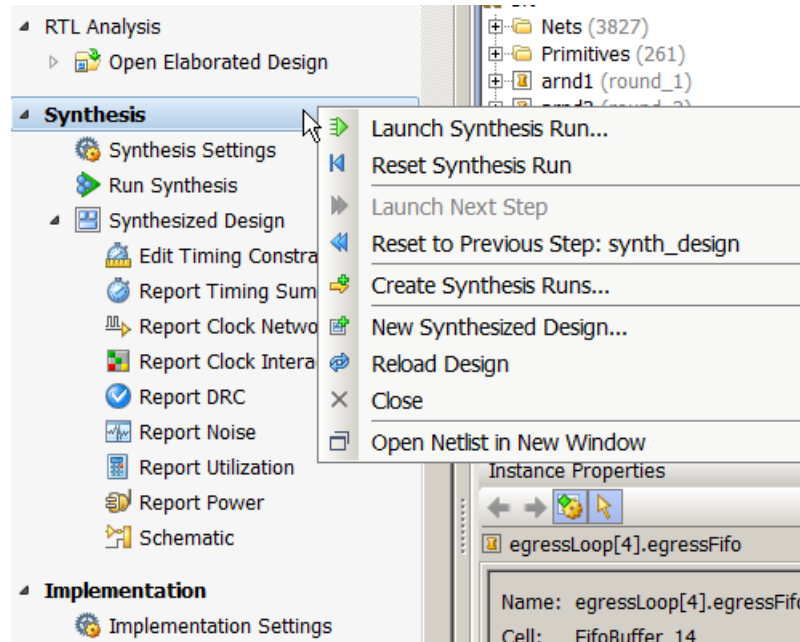
Figure 3-20:   **Advanced Flow Navigator Control Commands**

You can select **Open Netlist in New Window** to open a new 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 3-21 shows the Design View banners for each of the different design types.

Figure 3-21:   **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 3-22.

Figure 3-22:   **Multiple Open Design Tabs**

Updating Out-of-Date Designs

In the course of any design process, source files or constraints often require modification. The Vivado Design Suite 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.
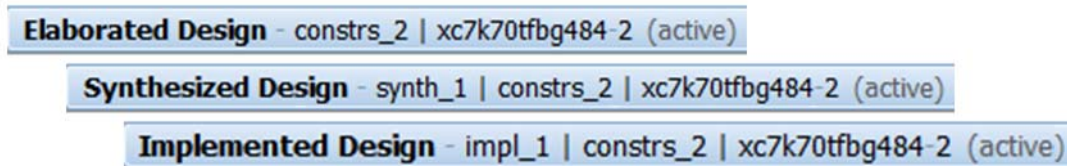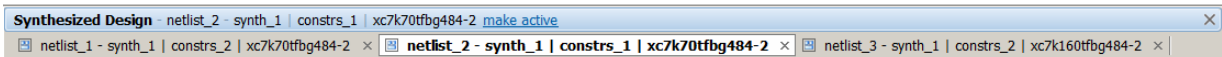
As source files, netlists, or implementation results are updated, an out-of-date message is displayed at the right side of the Design View banner of an open Synthesized Design or Implemented Design to indicate that the run is out of date, as shown in Figure 3-23. Click the associated link to view which aspects of the design are out of date.
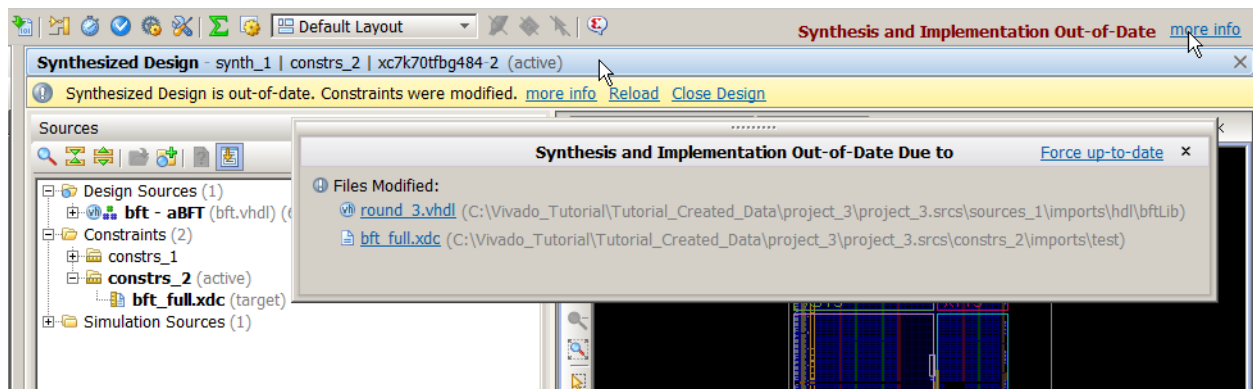
Figure 3-23:   **Design Out-of-Date and Reload Banner**

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 opens when you click more info as shown in Figure 3-23.

   Force up-to-date resets 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.

- **Close the Design**: Use the Close Design command to close the out-of-date Design.

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

Saving Design Changes

The Vivado IDE enables you to interactively edit the live design in memory. Possible interactions that might require a design save are constraints changes, netlist changes, design parameters such as power analysis characteristics, hardware configuration mode parameters, debug configuration, and so forth. Changes made while interactively editing any open design can be saved in one of two ways. They can be written back to your original XDC constraint files or into a new constraint set. The new constraint set includes all design constraints with the changes. This is one way to maintain your original XDC source files. The Vivado IDE attempts to maintain the original file format as much as possible. Additional constraints are added at the end of the file. Changes to existing constraints remain in their original file locations.

Saving Changes to Original XDC Constraint Files

Changes can be written back to your original XDC constraint files. The Vivado IDE attempts to maintain the original file format as much as possible. Additional constraints are added at the end of the file. Changes to existing constraints remain in their original file locations.

To save any changes you have made to your design data back to your original XDC constraint files, select **File > Save Constraints** or click the **Save Constraints** button ![save icon] .

The Save Constraints command saves any changes made to the constraints, debug cores and configuration, and design configuration settings made in the open design.

Note: When an Implemented Design is open, the Save Constraints command 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 might not be the intended result. The Vivado Design Suite warns of this situation, and lets you choose the constraint file to target changes to before writing changes to disk.

Saving Changes to a New Constraint Set

You can also save changes to the Design to a new constraint set. This preserves your original constraints source files, while still saving any changes. The new constraint set includes all design constraints including the changes. This is one way to maintain your original XDC source files. You also have the ability to make it the active Constraint set so that it is automatically applied to the next run or when opening Designs.

Select **File > Save Constraints As** to create a new constraint file, while preserving the original constraint source files.

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. You can close:

*   Individual designs by clicking Close (**X**) in the open Design banner of the main viewing area.

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

Performing Design Analysis, Verification, and Debugging

Logic Simulation

The Vivado Design suite has several logic simulation options for verifying designs or IP. The Vivado Simulator is a fully integrated mixed mode simulator with analog waveform display capabilities. The same simulation environment is used for behavior and structural simulation, as well as hardware validation with Vivado logic analyzer.

Note: In 2012.2, the Vivado IDE is only configured to launch behavioral simulation. Structural and timing simulation must be invoked from the Tcl Console.

Third-party simulators can also be used by writing the Verilog, VHDL netlists, and SDF format files from the open design. You can launch the Mentor Graphics ModelSim and Questa simulators from the Vivado IDE.

For more information, see the *Vivado Design Suite User Guide: Logic Simulation (UG900)* or *Vivado Design Suite User Guide: Programming and Debugging (UG908)*.

Timing Analysis

The Vivado Design Suite has many timing analysis options available through the Tcl API and SDC constraint options. A large list of standard report Tcl commands are available to provide information about the clock structure, logic relationships, and constraints applied to your design. For more information, see the *Vivado Design Suite Tcl Command Reference Guide (UG835)*, or use the build in Help capability in the Tcl Console.

The Vivado IDE provides a graphical means to configure and view timing analysis results. Experiment with various types of timing analysis parameters in the Report Timing commands. The Clock Networks and Clock Interaction Report windows can be used to view clock topology and relationships. The Slack Histogram provides an overall view of the design timing performance.

For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

DRC, Power, and Utilization Analysis

The Vivado Design Suite has many design analysis options available through the Tcl API and Vivado IDE. The Vivado IDE provides many different ways to visually examine the design and design data. The interactive nature of the environment coupled with window cross selection and logic exploration features enable powerful design analysis capabilities.

The Vivado IDE provides a graphical means to configure and view power, utilization, and DRC analysis results. Experiment with power parameters and quickly estimate power at any stage of the design. Various types of device resource utilization statistics can be analyzed. A comprehensive set of DRCs are available and can be configured and run. Results are reported with links to offending objects.

For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

A large list of standard report Tcl commands are available to provide information about the design. For more information, see the *Vivado Design Suite Tcl Command Reference Guide (UG835)*, or use the built-in Help capability in the Tcl Console.

Implementation Results Analysis

The implementation results for placement and routing are displayed when Implemented Designs are opened. Timing paths can be selected in the Timing Results window to display the placement and routing in the Device window. Placement and routing can be manipulated to achieve design goals. Designs configuration such as LUT equations, RAM initialization, PLL configuration, and so forth can be performed. Realize that these changes are being made on a version of the implemented design. Resetting the run causes changes to be lost.

For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

Hardware Validation and Debugging

The Vivado logic analyzer environment within the Vivado IDE has many features to enable verification and debugging of the design. You can configure and implement debug cores, such as the Integrated Logic Analyzer (ILA) and Integrated Controller (ICON) in either RTL or synthesized netlist. Opening the Synthesized Design in the Vivado IDE enables you to select and configure the required probe signals into the cores. You can launch the Vivado logic analyzer on any run that has a completed bitstream file for performing interactive hardware verification.

For more information, see the *Vivado Design Suite User Guide: Programming and Debugging (UG908)*.

Using Project Mode Tcl Commands

Tcl commands and scripting vary depending on the mode you use, and the resulting Tcl run scripts for each mode are different. Refer to the Project and Non-Project Mode Differences. The basic Project Mode Tcl commands are broken out below to show project creation and implementation and reporting. For a complete description of all project-related Tcl commands, refer to the *Vivado Design Suite Tcl Command Reference Guide (UG835)*.

The best way to understand the Tcl commands involved in a design task is to run the desired command interactively in the Vivado IDE and inspect the syntax in the Tcl Console or the `vivado.jou` file.

Note: This document is not a complete reference for the available Tcl commands. Instead, refer to the *Vivado Design Suite Tcl Command Reference Guide (UG835)* and *Vivado Design Suite User Guide: Using the Tcl Scripting Capabilities (UG894)*.

Basic Project Mode Tcl Commands

create_project	Creates the Vivado Project. Arguments include project name and location, design Top Module name and the target part
add_files	Adds source files to the Project. These include Verilog (.v), VHDL (.vhd or .vhdl), System Verilog (.sv), IP (.xco or .xci), XDC constraints (.xdc or .sdc), Embedded processor sub systems from XPS (.xmp), System Generator modules (.mdl). Individual files or entire directory trees can be scanned for legal sources and automatically added to the Project.
set_property	Used for a variety of purposes in the Vivado IDE. For Projects, it can be used to define VHDL Libraries for sources, simulation only sources, target constraints files, tool settings, etc.
import_files	Imports the specified files into the current file set, effectively adding them into the Project infrastructure. It is also used to define XDC files into Constraints sets.
launch_run launch_run –to_step	Starts either synthesis (synth_1), implementation (impl_1) and bitstream generation (impl_1). This command encompasses the individual implementation commands as well as the standard reports generated after the run completes. It is used to track run status and to bundle all implementation commands together into a single command. The –to_step command option can be used to run individual implementation commands and for bitstream generation.
wait_on_run	Ensures the run is complete before processing the next steps in the flow.
open_run	Opens either the Synthesized Design or Implemented Design for reporting or analysis. A design must be opened before information can be queried using Tcl for reports, analysis, etc.
close_design	Closes the design in memory.
start_gui stop_gui	Invokes or closes the Vivado IDE with the current design in memory.

Figure 3-24: **Basic Project Mode Tcl Commands**

Other Project Mode Tcl Commands

Table 3-2: **Project Mode-Specific Tcl Commands**

add_files	current_project	get_ips	list_targets	reset_run
archive_project	current_run	get_projects	make_wrapper	reset_target
close_design	delete_fileset	get_runs	open_example_project	save_design
close_project	delete_run	import_as_run	open_io_design	save_design_as
convert_ip	find_top	import_files	open_project	save_project_as
create_fileset	generate_target	import_ip	open_run	set_speed_grade
create_project	get_boards	import_synplify	refresh_design	update_design
create_run	get_files	import_xise	reimport_files	update_files
create_violation	get_filesets	import_xst	remove_files	upgrade_ip
current_fileset	get_ipdefs	launch_runs	reorder_files	wait_on_run

Project Mode Tcl Script Example

Following is a Project Mode Tcl script for the BFT sample design included with the Vivado Design Suite. In this example, many of the base-level commands, such as synth_design, are encapsulated in the launch_runs command.

```
#
# STEP#1: Create Project, add and configure sources and configure design
#
create_project project_bft ./project_bft -part xc7k70tfbg484-2
add_files -norecurse {./Sources/hdl/async_fifo.v ./Sources/hdl/bft.vhdl
./Sources/hdl/FifoBuffer.v}
add_files -norecurse ./Sources/hdl/bftLib
set_property library bftLib [get_files -of_objects sources_1 [glob
./Sources/hdl/bftLib/*.vhdl]]
import_files -force -norecurse
import_files -fileset constrs_1 -force -norecurse ./Sources/bft_full.xdc
set_property target_constrs_file
./project_bft/project_bft.srcs/constrs_1/imports/Sources/bft_full.xdc\
[cur-rent_fileset -constrset]
#
# STEP#2: Configure and launch Synthesis and Implementation and generate reports
#
set_property args.synth_design.flatten_hierarchy rebuilt [get_runs synth_1]
launch_runs synth_1 -jobs 4
wait_on_run synth_1
launch_runs impl_1 -jobs 4
wait_on_run impl_1
launch_runs impl_1 -to_step bitgen
wait_on_run impl_1
#
# STEP#3: Start IDE for design analysis
#
start_gui
stop_gui
```

Using Non-Project Mode

Using Non-Project Mode Tcl Commands

When using Non-Project Mode, the design is compiled using `read_verilog`, `read_vhdl`, `read_edif`, `read_ip`, and `read_xdc` type commands. The sources are ordered for compilation and passed to synthesis.

Note: This document is not a complete reference for the available Tcl commands. Instead, refer to the *Vivado Design Suite Tcl Command Reference Guide (UG835)* and *Vivado Design Suite User Guide: Using the Tcl Scripting Capabilities (UG894)*.

Basic Non-Project Mode Tcl Commands

read_verilog	Reads the Verilog (.v) and System Verilog (.sv), source files for the Non-Project session.
read_vhdl	Reads the VHDL (.vhd or .vhdl) source files for the Non-Project session.
read_ip	Reads existing IP (.xco or .xci) project files or the Non-Project session. The .ngc netlist is used from the .xco IP project. For .xci IP, the RTL is used for compilation or the netlist is used , if it exists.
read_xdc	Reads the .sdc or .xdc format constraints source files for the Non-Project session.
set_parameter set_property	Used for a variety of purposes to define design configuration, tool settings, etc.
link_design	Compiles the design for synthesis, if netlists sources are used for the session.
synth_design	Launches Vivado synthesis with the design Top Module name and target part as arguments.
opt_design	Performs some high level design optimization to better prepare the design for synthesis.
power_opt_design	Optional step to perform intelligent clock gating in order to reduce overall system power.
place_design	Places the design
phys_opt_design	Optional step to perform physical logic optimization to improve timing or routability.
route_design	Routes the design
report_*	Wide range of standard reports that can be run at any stage of the design process
write_bitstream	Generates bit file and runs DRCs.
write_checkpoint read_checkpoint	Saves the design at any point in the flow. A Design Checkpoint consists of the netlist and constraints with any optimizations at that point in the flow, as well as the implementation results
start_gui stop_gui	Invokes or closes the Vivado IDE with the current design in memory.

Figure 3-25: **Basic Non-Project Mode Tcl Commands**

Other Non-Project Mode Tcl Commands

Many Tcl commands can be used in either mode, such as the reporting commands. In some cases, Tcl commands are specific to either Project Mode or Non-Project Mode. Commands that are specific to one mode should not be mixed when creating scripts. For example, if you are using the Project Mode you should not use base-level commands such as `synth_design` because these are specific to Non-Project Mode. If you use Non-Project Mode commands in Project Mode, the database is not updated with status information and reports are not automatically generated.

Table 3-3:   **Non-Project Mode Specific Tcl Commands**

opt_design	power_opt_design	read_verilog	route_design
phys_opt_design	read_checkpoint	read_vhdl	write_checkpoint
place_design	read_edif	read_xdc	

Non-Project Mode Tcl Script Example

Following is a Non-Project Mode Tcl script for the BFT sample design included with the Vivado Design Suite. This example shows how to use the design checkpoints for saving the database state at various stages of the flow and how to manually generate various reports.

```
# create_bft_batch.tcl
# bft sample design
# A Vivado script that demonstrates a very simple RTL-to-bitstream batch flow
#
# NOTE:  typical usage would be "vivado -mode tcl -source create_bft_batch.tcl"
#
# STEP#0: define output directory area.
#
set outputDir ./Tutorial_Created_Data/bft_output
file mkdir $outputDir
#
# STEP#1: setup design sources and constraints
#
read_vhdl -library bftLib [ glob ./Sources/hdl/bftLib/*.vhdl ]
read_vhdl ./Sources/hdl/bft.vhdl
read_verilog  [ glob ./Sources/hdl/*.v ]
read_xdc ./Sources/bft_full.xdc
#
# STEP#2: run synthesis, report utilization and timing estimates, write checkpoint
design
#
synth_design -top bft -part xc7k70tfbg484-2 -flatten rebuilt
write_checkpoint -force $outputDir/post_synth
report_timing_summary -file $outputDir/post_synth_timing_summary.rpt
report_power -file $outputDir/post_synth_power.rpt
#
# STEP#3: run placement and logic optimization, report utilization and timing
estimates, write checkpoint design
#
opt_design
power_opt_design
```

```
place_design
phys_opt_design
write_checkpoint -force $outputDir/post_place
report_timing_summary -file $outputDir/post_place_timing_summary.rpt
#
# STEP#4: run router, report actual utilization and timing, write checkpoint design,
run drc, write verilog and xdc out
#
route_design
write_checkpoint -force $outputDir/post_route
report_timing_summary -file $outputDir/post_route_timing_summary.rpt
report_timing -sort_by group -max_paths 100 -path_type summary -file
$output-Dir/post_route_timing.rpt
report_clock_utilization -file $outputDir/clock_util.rpt
report_utilization -file $outputDir/post_route_util.rpt
report_power -file $outputDir/post_route_power.rpt
report_drc -file $outputDir/post_imp_drc.rpt
write_verilog -force $outputDir/bft_impl_netlist.v
write_xdc -no_fixed_only -force $outputDir/bft_impl.xdc
#
# STEP#5: generate a bitstream
#
write_bitstream -force $outputDir/bft.bit
```

Using Design Checkpoints

The Vivado IDE provides design checkpoints as a mechanism to save and restore the database. Checkpoints are a merely a snapshot of the design at that point in the flow. The current netlist, constraints and implementation results are stored in the design checkpoint. You can elect to save a design checkpoint at any point in the Non-Project Mode design flow. Design checkpoint designs can be run through the remainder of the design flow using Tcl commands, but cannot accept updates from new sources.

The Vivado Design Suite Tcl flow provides the following commands for saving/restoring in Non-Project Mode:

- `write_checkpoint` (write design checkpoint) captures a snapshot of the design database at any stage. This creates a file with a `.dcp` file extension.

- `read_checkpoint` (read design checkpoint) restores the design snapshot, which might be helpful for debugging issues, because it represents a full save of the design (but not a project) in its current implementation state. It can be run through the remainder of the flow using Tcl commands.

For detailed syntax of these commands, enter the commands with the `-help` option or the *Vivado Design Suite Tcl Command Reference Guide (UG835)*.

Using the Vivado IDE in Non-Project Mode

Opening the Vivado IDE for the Active Design

The Vivado Design Suite Non-Project Tcl flow provides the ability to open the Vivado IDE after each stage of the design process. Use this to analyze and operate on the current design in memory. The start_gui and stop_gui Tcl commands are used to invoke and close the Vivado IDE. Some of the project features such as the Flow Navigator, Project Summary, Source access and management, and Runs are not available in Non-Project Mode. However, many of the analysis and constraint modification features are available in the Tools menu.

Be aware that any changes made in the Vivado IDE are made to the active design in memory and are automatically applied to downstream tools. There is no save function. If you want to save constraint changes for subsequent runs, select **File > Export > Export Constraints** command to write a new all-inclusive XDC constraint file.

When using Non-Project Mode, the following commands are used for opening and closing the Vivado IDE on the active design in memory:

* start_gui opens the Vivado IDE with the active design in memory.

* stop_gui closes the Vivado IDE and returns the Tcl prompt.

IMPORTANT: *Closing the Vivado IDE also closes the Tcl shell, and flushes the design in memory. Be sure to enter the* **stop_gui** *command in the Tcl Console to return to the Tcl shell with the active design intact.*
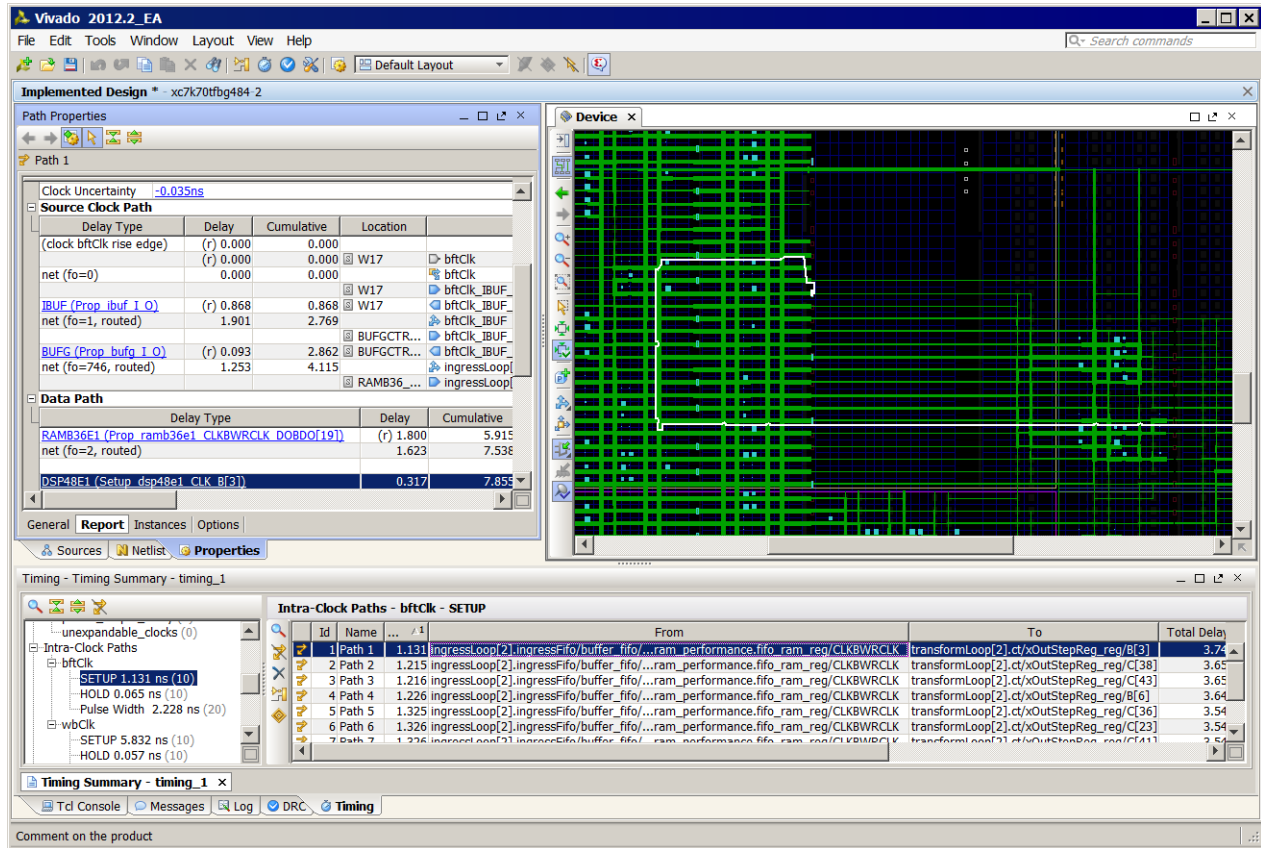
Figure 3-26:   **Opening Vivado IDE with the Active Design**

Saving Design Changes to the Active Design

Because you are actively editing the design in memory, changes made are automatically passed to downstream tools for the remainder of the Vivado IDE Tcl session. This is a new concept to those familiar with the ISE Design Suite tools. It provides a lot of power to affect the current run and to save the changes for future attempts.

Select **File > Export > Export Constraints** to save constraints changes for future use. You can use the command to write a new constraints file or override your original file.

Opening Design Checkpoints in the Vivado IDE

The Vivado IDE can also be used to analyze designs saved as design checkpoints. You can run a design in Non-Project Mode using Tcl commands (`synth_design`, `opt_design`, `power_opt_design`, `place_design`, `phys_opt_design`, and `route_design`), store the design at any stage, and read it in later in a Vivado IDE session. You can start with a routed design, analyze timing, adjust placement to address timing problems, and save your work for later, even if the design is not fully routed.
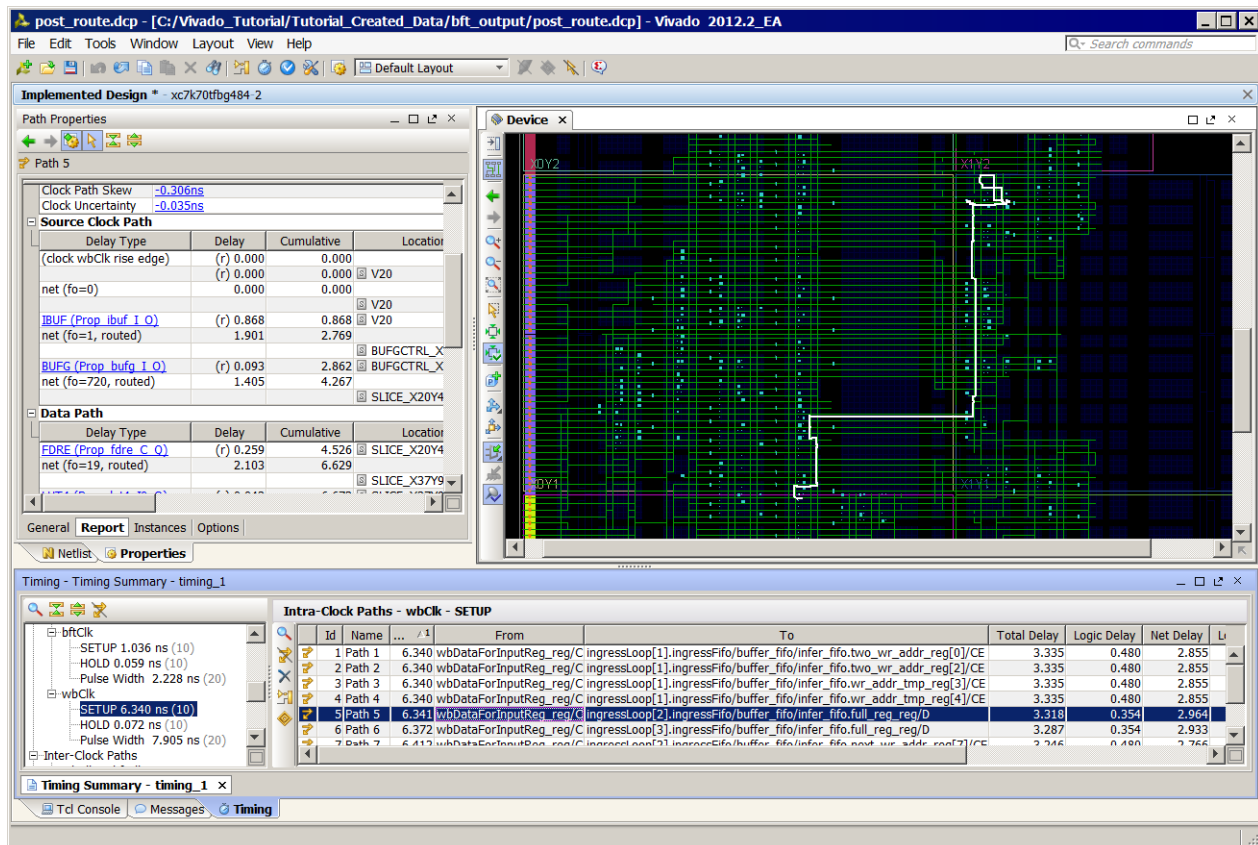

Figure 3-27: **Opening Design Checkpoints in the Vivado IDE**

The Vivado IDE view banner displays the open design checkpoint name.

Saving Design Changes to Design Checkpoints

You can open, analyze, and save design checkpoints designs. You can also save changes to a new design checkpoint:

- Select **File > Save Checkpoint** to save changes made to the current design checkpoint.

- Select **File > Write Checkpoint** to save the current state of the checkpoint to a new design checkpoint.

www.xilinx.com

Constraints changes can be exported to a new constraints file. Select **File > Export > Export Constraints** to save changes to a new constraints file or override the original file. For more information, see the *Vivado Design Suite User Guide: Implementation (UG904)*.

For detailed syntax of these commands, enter the commands with the `-help` option or the *Vivado Design Suite Tcl Command Reference Guide (UG835)*.

Using Third-Party Synthesized Netlists

The Vivado Design Suite supports implementation of synthesized netlists, such as when using a third-party synthesis tool or Xilinx Synthesis Technology (XST). The external synthesis tool generates a Verilog or EDIF netlist and a constraints file, if applicable. These netlists can be used standalone or mixed with RTL files in either Project Mode or Non-Project Mode.

Non-Project Mode Implementation

For a netlist-only Non-Project Mode, the netlist flow differs as follows:

- `read_edif` reads the EDIF or NGC netlist files. When using netlists, the file base name must match the name of the top-level design.

- `read_verilog` reads structural Verilog files.

- `read_xdc` reads constraints files.

- The top module and part is specified when opening the design for implementation.

  ```
  link_design -top my_module -part xc7vx485tffg1157-1
  ```

- For a mixed RTL/netlist design, the `synth_design` command links the netlist automatically, so there is no need for the `link_design` command.

Following is a simple example of a Non-Project Mode script for a single EDIF:

```
read_edif my_top.edf
read_xdc my_top.xdc
link_design -top my_top -part xc7vx485tffg1157-1
opt_design
# power_opt_design   ;# Optional
place_design
# phys_opt_design    ;# Optional
route_design
report_timing_summary
report_drc
write_checkpoint routed
write_bitstream
```

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx® Support website at:

www.xilinx.com/support

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

*   *Vivado™ Design Suite 2012.2 Documentation*
    (www.xilinx.com/support/documentation/dt_vivado_vivado2012-2.htm)