Vivado Design Suite User Guide

Design Flows Overview

UG892 (v2015.4) November 18, 2015

XILINX
ALL PROGRAMMABLE™

Revision History

11/18/2015: Released with Vivado Design Suite 2015.4 without changes from the previous version.

Date	Version	Revision
09/30/2015	2015.3	Added new tips in Chapter 2, Understanding Use Models and Chapter 4, Using Non-Project Mode.
04/01/2015	2015.1	Added new Hierarchical Design section and updated Figure 3-16. Added content updates and new QuickTake Video links in Chapter 1, Introduction, Chapter 2, Understanding Use Models, Chapter 3, Using Project Mode, and Chapter 4, Using Non-Project Mode. Updated Figure 3-1, Figure 3-2, Figure 3-4, Figure 3-5, Figure 3-6, Figure 3-14, Figure 3-15, Figure 3-16, and Figure 4-1.
06/19/2015	2015.1	Updated Figure 1-1 and three destination links in Chapter 3, Using Project Mode (page 32, 39, and 56).

Table of Contents

Chapter 1: Introduction

Overview 5
Vivado Design Suite Features 5
System-Level Design Flow 6
Out-of-Context Design Flow 9
Industry Standards-Based Design 10
Xilinx Evaluation Board Interfaces 11
I/O Pin Planning and Floorplanning 11
Design Analysis and Verification 11
Device Programming and Hardware Validation 12
Partial Reconfiguration 12
Hierarchical Design 12

Chapter 2: Understanding Use Models

Understanding Project Mode and Non-Project Mode 14
Working with Tcl 18
Working with the Vivado IDE 19
Interfacing with Revision Control Systems 20
Interfacing with PCB Designers 21
Using Third-Party Design Software Tools 21

Chapter 3: Using Project Mode

Overview 23
Project Mode Advantages 25
Creating Projects 25
Understanding the Flow Navigator 28
Performing System-Level Design Entry 30
Working with IP 33
Running Logic Simulation 38
I/O Pin Planning 38
Running Logic Synthesis and Implementation 39
Viewing Log Files, Messages, Reports, and Properties 42
Opening Designs to Perform Design Analysis and Constraints Definition 46

Device Programming, Hardware Verification, and Debugging 56
Using Project Mode Tcl Commands 57

Chapter 4: Using Non-Project Mode

Overview 60
Non-Project Mode Advantages 61
Reading Design Sources 61
Working with IP 62
Running Logic Simulation 63
Running Logic Synthesis and Implementation 63
Generating Reports 64
Using Design Checkpoints 64
Performing Design Analysis Using the Vivado IDE 65
Using Non-Project Mode Tcl Commands 68

Appendix A: Additional Resources and Legal Notices

Xilinx Resources 71
Solution Centers 71
References 71
Training Resources 72
Please Read: Important Legal Notices 73

Chapter 1

Introduction

Overview

The Vivado® Design Suite offers multiple ways to accomplish the tasks involved in Xilinx® FPGA design and verification. In addition to the traditional register transfer level (RTL)-to-bitstream FPGA design flow, the Vivado Design Suite provides new system-level integration flows that focus on intellectual property (IP)-centric design. Various IP can be instantiated, configured, and interactively connected into IP subsystem block designs within the Vivado IP integrator environment. Custom IP and IP block designs can be configured and packaged and made available from the Vivado IP catalog. 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, constraint definition and timing analysis, design rule checks (DRC), visualization of design logic, analysis and modification of 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 Tcl commands. Tcl commands can be scripted or entered interactively using the Vivado Design Suite Tcl shell or using the Tcl Console in the Vivado IDE. You can use Tcl scripts to run the entire design flow, including design analysis, or to run only parts of the flow.

Vivado Design Suite Features

You can use the Vivado Design Suite for different types of designs. The tool flow and the features differ depending on the type of design. The System-Level Design Flow section highlights the main features enabled by the Vivado Design Suite.

System-Level Design Flow

Figure 1-1 shows the high-level design flow in the Vivado Design Suite.

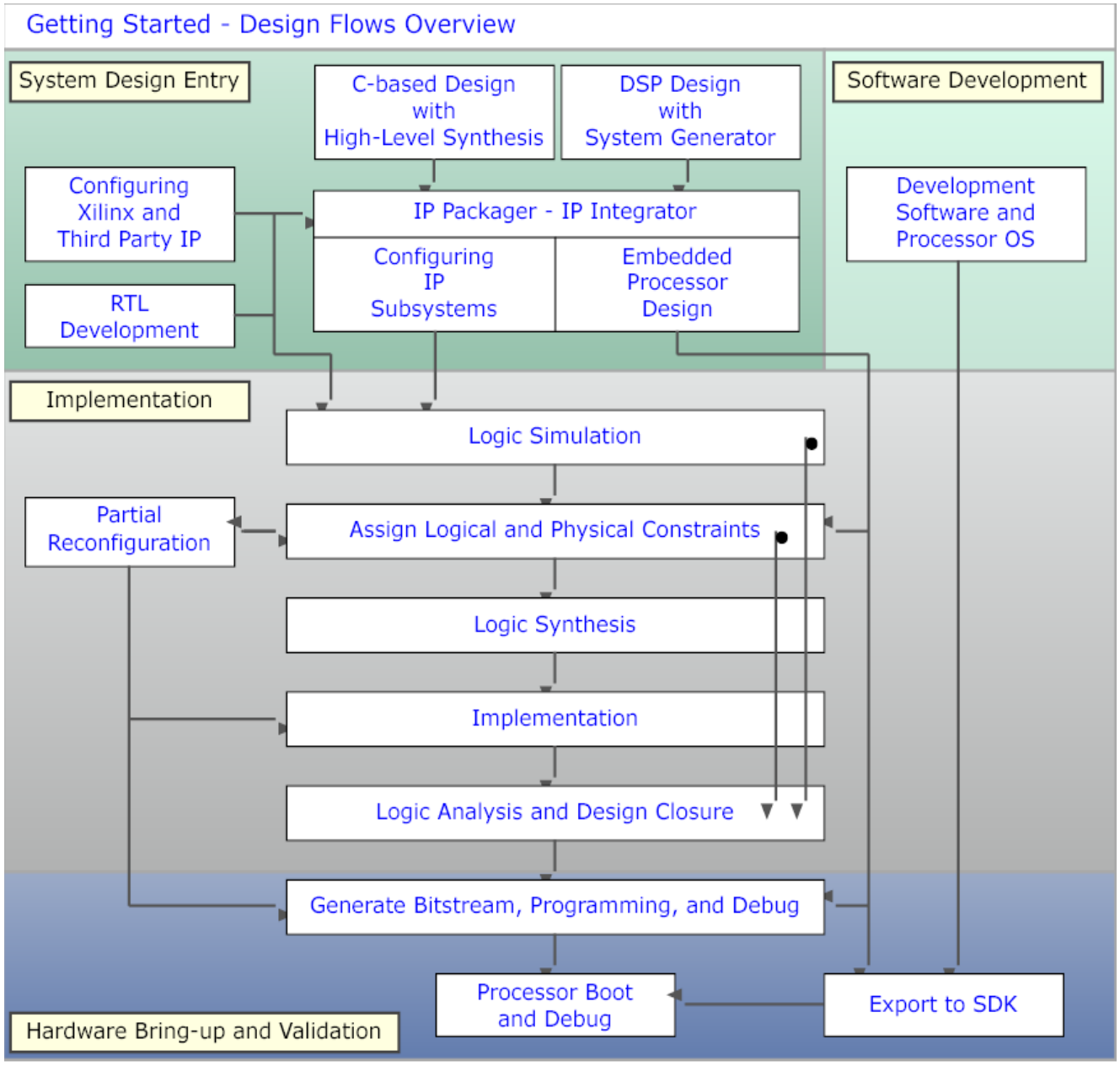

Figure 1-1: **Vivado Design Suite High-Level Design Flow**

IP Design and System-Level Design Integration

The Vivado Design Suite provides an environment to configure, implement, verify, and integrate IP as a standalone module or within the context of the system-level design. IP can include logic, embedded processors, digital signal processing (DSP) modules, or C-based DSP algorithm designs. Custom IP is packaged following IP-XACT protocol and then made available through the Vivado IP catalog. The IP catalog provides quick access to the IP for configuration, instantiation, and validation of IP. Xilinx IP utilizes the AXI4 interconnect standard to enable faster system-level integration. Existing IP can be used in the design either in RTL or netlist format. For more information, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Note: The ISE format IP (ngc) is no longer supported with UltraScale designs.

RTL or Netlist to Device Programming Design Flows

The Vivado Design Suite has different design entry points to support various design flows:

- RTL flow

  Vivado synthesis and implementation support multiple source file types, including Verilog, VHDL, SystemVerilog, and XDC. You can also use Vivado HLS to compile parts of the design using C-based sources.

- Third-party synthesis flow

  Vivado synthesis supports third-party synthesis sources, including EDIF or structural Verilog. SDC sources are also supported. However, it is recommended that you adhere to and take advantage of the XDC constructs. Vivado IP is synthesized using Vivado synthesis. In general, you must not synthesize Vivado Design Suite IP sources with third-party synthesis tools. However, there are a few exceptions, such as memory interface generator (MIG) cores for 7 series devices.

Following are the main design flow features:

- Vivado synthesis
- Vivado implementation
- Vivado timing analysis
- Vivado power analysis
- Bitstream generation

These features are designed to provide larger design capacity and increased design performance with decreased runtimes. The Vivado synthesis and implementation features are timing driven and use SDC or XDC format constraints. Various reports and analysis features are available at each stage of the design process. You can run the design through the entire flow by using the Vivado IDE, using batch Tcl scripts, or entering Tcl commands at the Vivado Design Suite Tcl shell or the Vivado IDE Tcl Console. To help improve design results, you can create multiple runs to experiment with different synthesis or implementation options, timing and physical constraints, or design configuration.

The Vivado Design Suite uses design projects to configure and manage the entire design process. Sources, design configuration, and run results are stored and managed within the Vivado Design Suite project. The design status notifies you of status changes, such as when source files have been updated and run results are out-of-date. The Vivado IDE generates and displays a standard set of reports, tool messages, and logs during synthesis and implementation. Some advanced options are available for implementation, such as Vivado power optimization, Vivado physical optimizer, and run strategies, that assist you with design closure. For more information, see the *Vivado Design Suite User Guide: Synthesis* (UG901) [Ref 2] and *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

IP Subsystem Design

The Vivado IP Integrator environment enables you to stitch together various IP into IP subsystems using the AMBA AXI4 interconnect protocol. You can interactively configure and connect IP using a block design style interface and easily connect entire interfaces by drawing DRC-correct connections similar to a schematic. Connecting the IP using standard interfaces saves time over traditional RTL-based connectivity. Connection automation is provided as well as a set of DRCs to ensure proper IP configuration and connectivity. These IP block designs are then validated, packaged, and treated as a single design source. Block designs can be used in a design project or shared among other projects. The IP Integrator environment is the main interface for embedded design and the Xilinx evaluation board interface. For more information, see the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 4].

Embedded Processor Hardware Design

Because an embedded processor requires software to boot and run effectively, the software design flow must work in unison with the hardware design flow. Different data handoff points and validation across the two domains is critical for success. Creating an embedded processor hardware design involves the IP integrator feature of the Vivado Design Suite. In a Vivado IP integrator block design, you instantiate, configure, and assemble the processor core and its interfaces. The tool enforces rules-based connectivity and provides design assistance. After the design is compiled through implementation, it is exported to the Xilinx Software Development Kit (SDK) for use in the software development and validation flows. Simulation and debug features allow you to simulate and validate the design across the two domains. For more information on the embedded processor design flow, see the *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898) [Ref 5] and *Vivado Design Suite Tutorial: Embedded Processor Hardware Design* (UG940) [Ref 6].

IMPORTANT: *The Vivado IP integrator is the replacement for Xilinx Platform Studio (XPS) for new embedded processor designs, including designs targeting Zynq®-7000 All Programmable devices and MicroBlaze™ processors. XPS is no longer integrated with the Vivado Design Suite.*

VIDEO: *For training videos on the Vivado IP integrator and the embedded processor design flow, see the* Vivado Design Suite QuickTake Video: Designing with Vivado IP Integrator *and* Vivado Design Suite QuickTake Video: Targeting Zynq Devices Using Vivado IP Integrator.

Model-Based and High-Level Synthesis-Based DSP Design

Model-Based DSP Design Using Xilinx System Generator

The Vivado Design Suite is also integrated directly with the Xilinx System Generator tool to provide a solution for implementing DSP functions. DSP modules are integrated and managed within the Vivado IDE. When you select a DSP source for edit, the System

Generator launches automatically. You can also use System Generator as a standalone tool and use the resulting output files as source files in the Vivado IDE. For more information, see the *Vivado Design Suite User Guide: Model-Based DSP Design Using System Generator* (UG897) [Ref 7].

DSP Design Using High-Level Synthesis

The C-based High-Level Synthesis (HLS) tool within the Vivado Design Suite enables you to describe various DSP functions in the design using C, C++, and System C languages. You create and validate the C code with the Vivado HLS tool. You can then perform multiple experiments using various parameters to optimize performance and area. You can quickly validate the design and create RTL simulation test benches using C-based simulation. C-to-RTL synthesis transforms the C-based design into an RTL module that can be packaged and implemented with the rest of the design. This module can then be instantiated into the RTL design or within Vivado IP integrator. For more information on the HLS tool flow and features, see the *Vivado Design Suite User Guide: High-Level Synthesis* (UG902) [Ref 8] and *Vivado Design Suite Tutorial: High-Level Synthesis* (UG871)[Ref 9].

VIDEO: *For various training videos on Vivado HLS, see the Vivado High-Level Synthesis video tutorials available from the* Vivado Design Suite QuickTake Video Tutorials *page on the Xilinx website.*

Out-of-Context Design Flow

By default, the Vivado Design Suite uses an out-of-context (OOC) design flow to synthesize IP cores from the Xilinx IP catalog, and block designs from the Vivado IP integrator. This OOC flow lets you synthesize, implement, and analyze design modules in a hierarchical design, IP cores, or block designs, independent of the top-level design. The OOC flow reduces design cycle time, and eliminates design iterations, letting you save synthesis and implementation results for completed modules in design checkpoint (DCP) files. IP cores that are added to a design from the Xilinx IP catalog default to use the out-of-context flow as described in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]. Block designs created in the Vivado IP integrator also default to the OOC flow when generating output products as described in the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 4].

The Vivado Design Suite supports global synthesis and implementation of a design, in which all modules, block designs, and IP cores, are handled as part of the integrated top-level design. However, with the OOC flow you can mark specific modules for out-of-context synthesis, and other modules for inclusion in the global synthesis of the top-level design. In the case of block design from Vivado IP integrator, the entire block design can be specified as an OOC module, or all of the individual IP used in the block design can be marked as out-of-context. The Vivado synthesis tool also uses a cache, or repository, in which synthesis results for OOC modules can be stored for reuse so multiple

instances of the same IP customization can use the same results to speed synthesis of large complex designs.

OOC modules are seen as black boxes in the top-level design until the synthesized design is open and all the elements are assembled. Before the top-level synthesized design is opened, resource utilization and analysis of the top-level design may not include information from the OOC modules, or black boxes, and so will not provide a complete view of the design. To obtain more accurate reports, you should open and analyze the top-level synthesized design, which will include all the integrated OOC modules.

The Out-of-Context flow is supported in Vivado synthesis, implementation, and analysis. For more information refer to this link in the *Vivado Design Suite User Guide: Synthesis* (UG901) [Ref 2]. It can be used to define a hierarchical design methodology and a team design approach as defined in the *Vivado Design Suite User Guide: Hierarchical Design* (UG905)[Ref 10].

Industry Standards-Based Design

The Vivado Design Suite supports the following established industry design standards:

- Tcl
- AXI4, IP-XACT
- Synopsys design constraints (SDC)
- Verilog, VHDL, SystemVerilog
- SystemC, C, C++

The Vivado Design Suite solution is native Tcl based with support for SDC and Xilinx design constraints (XDC) formats. Broad Verilog, VHDL, and SystemVerilog support for synthesis enables easier FPGA adoption. Vivado High-Level Synthesis (HLS) enables the use of native C, C++, or SystemC languages to define logic. Using standard IP interconnect protocol, such as AXI4 and IP-XACT, enables faster and easier system-level design integration. Support for these industry standards also enables the electronic design automation (EDA) ecosystem to better support the Vivado Design Suite. In addition, many new third-party tools are integrated with the Vivado Design Suite.

Xilinx Platform Board Flow

In the Vivado Design Suite, you can select an existing Xilinx evaluation platform board as a target for your design. In the platform board flow, all of the IP interfaces implemented on the target board are exposed to enable quick selection and configuration of the IP used in your design. The resulting IP configuration parameters and physical board constraints, such

as I/O standard and package pin constraints, are automatically assigned and proliferated throughout the flow. Connection automation enables quick connections to the selected IP. For more information see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 12].

I/O Pin Planning and Floorplanning

The Vivado IDE provides an I/O pin planning environment that enables I/O port assignment either onto specific device package pins or onto internal die pads. You can analyze the device and design-related I/O data using the views and tables available in the Vivado pin planner. For more information, see the *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 11].

The Vivado IDE provides advanced floorplanning capabilities to help drive improved implementation results. These include the ability to force specified logic inside of a particular area or by interactively locking specific placement or routing for subsequent runs. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

Design Analysis and Verification

The Vivado IDE enables you to analyze, verify, and modify the design at each stage of the design process. You can improve circuit performance by analyzing the interim results in the design process. This analysis can be run after RTL elaboration, synthesis, and implementation.

The Vivado simulator enables you to run behavioral and structural logic simulation at each stage of the design. The simulator supports Verilog and VHDL mixed-mode simulation, and results are displayed in a waveform viewer integrated with the Vivado IDE. Third-party simulators can also be used. For more information, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 14].

Results can be interactively analyzed in the Vivado IDE at each stage of the design process. Some of the design and analysis features include timing analysis, power estimation and analysis, device utilization statistics, DRCs, I/O planning, floorplanning, and interactive placement and routing analysis. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

Device Programming and Hardware Validation

After implementation, the device can be programmed and then analyzed within the Vivado lab tools environment. Debug signals can be identified in RTL or after synthesis and are processed throughout the flow. Debug cores can be configured and inserted either in RTL, in the synthesized netlist, or in the implemented design. The Vivado logic analyzer also enables hardware validation. The interface is designed to be consistent with the Vivado simulator, and both share a common waveform viewer. For more information, see the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 15].

Partial Reconfiguration

Partial Reconfiguration allows a portion or portions of the design to be reconfigured while the device is up and running. This flow requires a rather strict design process to ensure that the reconfigurable modules are designed properly to enable glitchless operation during partial bitstream updates. The reconfigurable modules need to be properly planned to ensure they function as expected and for maximum performance. This includes reducing the number of interface signals into the module, proper floorplanning, module pin placement, as well as adhering to special partial reconfiguration DRCs. The method you use to program the device must also be properly planned to ensure the I/O pins are assigned accordingly. For more information, see the *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909) [Ref 16] and *Vivado Design Suite Tutorial: Partial Reconfiguration* (UG947) [Ref 17].

VIDEO: *For an overview of partial reconfiguration, see the* Vivado Design Suite QuickTake Video: Partial Reconfiguration in Vivado Design Suite.

Hierarchical Design

There are several Vivado features that enable a hierarchical design approach. The term **out-of-context** or **OOC** is used throughout the IDE and the documentation to describe either synthesizing or implementing a logic module standalone and not in the context of the top-level design. Users can select levels of design hierarchy and synthesize them OOC. Module-level constraints can be applied to optimize and validate module performance. The module netlist will then be applied during implementation. This method can help reduce top-level synthesis run time. Modules can also be implemented OOC and the results reused during top-level implementation. This involves proper module interface design, constraints definition, floorplanning, and some special commands and design techniques. For more information, see the *Vivado Design Suite User Guide: Hierarchical Design* (UG905) [Ref 10].

Understanding Use Models

Understanding Project Mode and Non-Project Mode

The Vivado® Design Suite enables you to run the tools using different methods depending on your preference. The Vivado Design Suite takes advantage of a project based architecture to a assemble, implement, and track the state of any given design. This is referred to as Project Mode. You can elect to use a project-based method to automatically manage your design process and design data. When working in Project Mode, a project file and a project directory structure is created on disk to manage design source files, store synthesis and implementation run results, and track project status.

A runs infrastructure is used to manage the automated synthesis and implementation process and to track run status. The entire design flow can be run with a single click within the Vivado IDE. The entire flow can also be scripted using Tcl commands. For detailed information on working with projects, see Chapter 3, Using Project Mode.

Alternatively, you can choose a Tcl script-based compilation flow in which you manage sources and the design process yourself, known as *Non-Project Mode*. Non-Project Mode discards the in-memory project after each session and only writes data to disk that you instruct it to with Tcl commands. That is the fundamental difference between Project and Non-Project Mode. When working in Non-Project Mode, sources are accessed from their current locations and the design is compiled through the flow in memory. Each design step is run individually using Tcl commands, and design parameters and implementation options are set using Tcl commands. You can save design checkpoints and create reports at any stage of the design process using Tcl. In addition, 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. For example, you can save updates to new constraint files or design checkpoints. For more information on Non-Project Mode, see Chapter 4, Using Non-Project Mode.

Note: Some of the features of Project Mode, such as source file and run results management, saving of design and tool configuration, design status, and IP integration, are not available in Non-Project Mode.

TIP: A project is always created in memory regardless of the use model.

TIP: *Either of these modes can be run using a Tcl scripted batch mode or run interactively in the Vivado IDE.*

Feature Differences

In Project Mode, the Vivado IDE tracks the history of the design and stores pertinent design information. However, because many features are automated, you have less control in this mode. For example, only a standard set of report files is generated with each run. The following automated features are only available when using Project Mode:

- Source file management and status
- Consolidated messages and automatically generated standard reports
- Storage and reuse of tool settings and design configuration
- Experimentation with multiple synthesis and implementation runs
- Use and management of constraint sets
- Run results management and status
- Flow Navigator
- Project Summary

Non-Project Mode, is more of a compile style methodology where each action is executed using a Tcl command. All of the processing is done in memory, so no files or reports are generated automatically. Each time you compile the design, you must define all of the sources, set all tool and design configuration parameters, launch all implementation commands, and generate report files. This can be accomplished using a Tcl run script, because a project is not created on disk, source files remain in their original locations and design output is only created when and where you specify. This method provides you with all of the power of Tcl commands and full control over the entire design process. Many users prefer this batch compilation style interaction with the tools and the design data.

Table 2-1 summarizes the feature differences between Project Mode and Non-Project Mode.

Table 2-1: **Project Mode versus Non-Project Mode Features**

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

Command Differences

Tcl commands vary depending on the mode you use, and the resulting Tcl run scripts for each mode are different. In Non-Project Mode, all operations and tool settings require individual Tcl commands, including setting tool options, running implementation commands, generating reports, and writing design checkpoints. In Project Mode, wrapper commands are used around the individual synthesis, implementation, and reporting commands.

For example, in Project Mode, you add sources to the project for management using the `add_files` Tcl commands. Sources can be copied into the project to maintain a separate version within the project directory structure or can be referenced remotely. In Non-Project Mode, you use the `read_verilog`, `read_vhdl`, `read_xdc`, and `read_*` Tcl commands to read the various types of sources from their current location.

In Project Mode, the `launch_runs` command launches the tools with pre-configured run strategies and generates standard reports. This enables consolidation of implementation commands, standard reporting, use of run strategies, and run status tracking. However, you can also run custom Tcl commands before or after each step of the design process. Run results are automatically stored and managed within the project. In Non-Project Mode, individual commands must be run, such as `opt_design`, `place_design`, and `route_design`.

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 must not be mixed when creating scripts. For example, if you are using the Project Mode you must 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.

TIP: *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 2-1 shows the difference between Project Mode and Non-Project Mode Tcl commands.

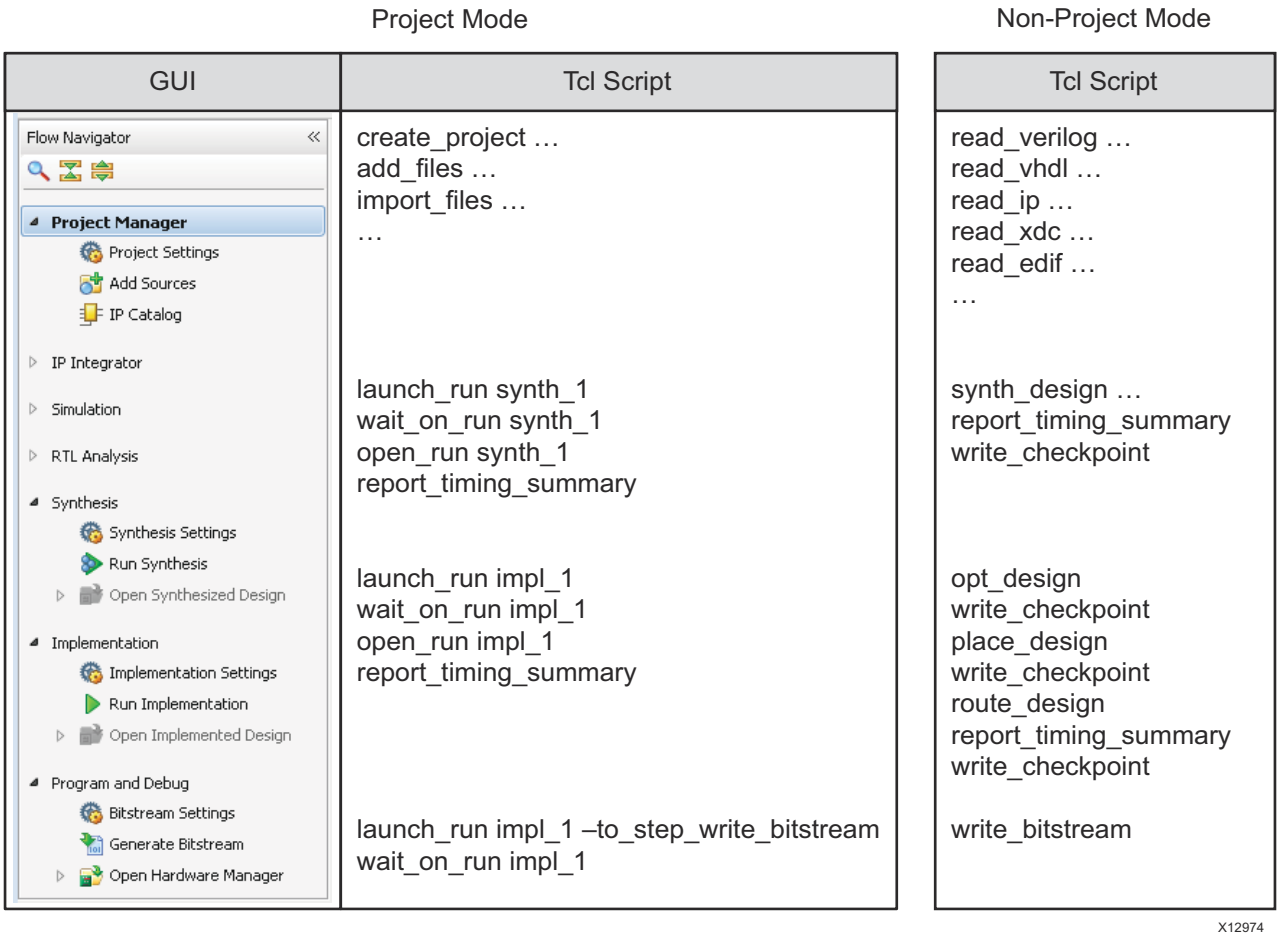

Figure 2-1: **Project Mode and Non-Project Mode Commands**

Working with Tcl

All flows can be run using Tcl commands. You can use Tcl scripts to run the entire design flow, including design analysis reporting, or to run only parts of the flow. If you prefer to work directly with Tcl, you can interact with your design using Tcl commands using either of the following methods:

- Enter individual Tcl commands in the Vivado Design Suite Tcl shell outside of the Vivado IDE.
- Enter individual Tcl commands in the Tcl Console at the bottom of the Vivado IDE.
- Run Tcl scripts from the Vivado Design Suite Tcl shell.
- Run Tcl scripts from the Vivado IDE.

For more information about using Tcl and Tcl scripting, see the *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894) [Ref 18] and *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 19]. For a step-by-step tutorial that shows how to use Tcl in the Vivado tools, see the *Vivado Design Suite Tutorial: Design Flows Overview* (UG888) [Ref 20]. For more information on using a Tcl-based approach using either the Project Mode or Non-Project Mode, see Chapter 3, Using Project Mode or Chapter 4, Using Non-Project Mode.

Launching the Vivado Design Suite Tcl Shell

Use the following command to invoke the Vivado Design Suite Tcl Shell either at the Linux command prompt or within a Windows Command Prompt window:

```
vivado -mode tcl
```

Note: On Windows, you can also select **Start > All Programs > Xilinx Design Tools > Vivado 2015.x > Vivado 2015.x Tcl Shell**.

Launching the Vivado Tools Using a Batch Tcl Script

You can use the Vivado tools 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:

```
vivado -mode batch -source <your_Tcl_script>
```

Note: When working in batch mode, the Vivado tools exit after running the specified script.

Using the Vivado IDE with a Tcl Flow

When working with Tcl, you can still take advantage of the interactive GUI-based analysis and constraint definition capabilities in the Vivado IDE. You can open designs in the Vivado IDE at any stage of the design cycle, as described in Performing Design Analysis Using the Vivado IDE in Chapter 4. You can also save design checkpoints at any time and open the checkpoints later in the Vivado IDE, as described in Using Design Checkpoints in Chapter 4.

Using the Xilinx Tcl Store

The Xilinx Tcl Store is an open source repository of Tcl code designed primarily for use in FPGA designs with the Vivado Design Suite. The Tcl Store provides access to multiple scripts and utilities contributed from different sources, which solve various issues and improve productivity. You can install Tcl scripts and also contribute Tcl scripts to share your expertise with others. For more information on working with Tcl scripts and the Xilinx Tcl Store, see the *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894) [Ref 18].

Working with the Vivado IDE

The Vivado IDE provides an interface to assemble, implement, and validate your design and IP. In Project Mode, the Vivado IDE supports a push-button design flow that manages all design sources, configuration, and results. The Vivado IDE enables constraints assignment and design analysis throughout the design process by introducing the concept of opening designs in memory. Opening a design loads the design netlist at that particular stage of the design flow, assigns the constraints to the design, and applies the design to the target device. This allows you to visualize and interact with the design at each design stage. You can open designs after RTL elaboration, synthesis, or implementation and make changes to constraints, logic or device configuration, and implementation results. You can also use design checkpoints to save the current state of any design. For more information on the Vivado IDE, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 21] and the *Vivado Design Suite Tutorial: Design Flows Overview* (UG888) [Ref 20].

RECOMMENDED: *Launch the Vivado IDE from your project working directory. This makes it easier to locate the project file, log files, and journal files, which are written to the launch directory.*

Launching the Vivado IDE on Windows

Select **Start > All Programs > Xilinx Design Tools > Vivado 2015.x > Vivado 2015.x**.

Note: You can also double-click the Vivado IDE shortcut icon on your desktop.

Figure 2-2: **Vivado IDE Desktop Icon**

TIP: *You can right-click the Vivado IDE shortcut icon, and select **Properties** to update the Start In field. This makes it easier to locate the project file, log files, and journal files, which are written to the launch directory.*

Launching the Vivado IDE from the Command Line on Windows or Linux

Enter the following command at the command prompt:

```
vivado
```

When you enter this command, it automatically runs `vivado -mode gui` to launch the Vivado IDE. If you need help, type `vivado -help`.

TIP: *To add the Vivado tools path to your current shell/command prompt, run settings64.bat or settings64.sh from the <install_path>/Vivado/<version> directory.*

Launching the Vivado IDE from the Vivado Design Suite Tcl Shell

When the Vivado Design Suite is running in Tcl mode, enter the following command at the Tcl command prompt to launch the Vivado IDE:

```
start_gui
```

Interfacing with Revision Control Systems

Many design teams use source management systems to store various design configurations and revisions. There are multiple commercially available systems, such as Revision Control System (RCS), Concurrent Versions System (CVS), Subversion (SVN), ClearCase, Perforce, Git, BitKeeper, and many others. The Vivado tools can interact with all such systems. The Vivado tools input and output files that are consumed and produced in the flow are the ones that most often need revision control.For more information on working with revision control

software, see this link in the *UltraFast™ Design Methodology Guide for the Vivado Design Suite* (UG949) [Ref 22].

VIDEO: *For information on best practices when using revision control systems with the Vivado tools, see the* Vivado Design Suite QuickTake Video: Version Control Overview.

RECOMMENDED: *The run script and tool settings should also be checked in for revision control. This information can be extracted into a Tcl script using the* `write_project_tcl` *command. Because* `init.tcl` *is not included in* `write_project_tcl`, `init.tcl` *must also be checked in. Checking in all these files enables the design to be recreated using the current sources and tool configuration settings.*

Interfacing with PCB Designers

The I/O planning process is critical to high-performing systems. Printed circuit board (PCB) designers are often concerned about the relationship and orientation of the FPGA device on the PCB. These large ball grid array (BGA) devices are often the most difficult routing challenge a PCB designer faces. Additional concerns include critical interface routing, location of power rails, and signal integrity. A close collaboration between FPGA and PCB designers can help address these design challenges. The Vivado IDE enables the designer to visualize the relationship between the physical package pins and the internal die pads to optimize the system-level interconnect.

The Vivado Design Suite has several methods to pass design information between the FPGA, PCB, and system design domains. I/O pin configuration can be passed back and forth using a comma separated value (CSV) spreadsheet, RTL header, or XDC file. The CSV spreadsheet contains additional package and I/O information that can be used for a variety of PCB design tasks, such as matched length connections and power connections. An I/O Buffer Information Specification (IBIS) model can also be exported from the Vivado IDE for use in signal integrity analysis on the PCB.

For more information, see the *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 11].

Using Third-Party Design Software Tools

Xilinx has strategic partnerships with several third-party design tool suppliers. The following software solutions include synthesis and simulation tools only.

Running Logic Synthesis

The Xilinx FPGA logic synthesis tools supplied by Synopsys and Mentor Graphics are supported for use with the Vivado Design Suite. In the Vivado Design Suite, you can import the synthesized netlists in structural Verilog or EDIF format for use during implementation. In addition, you can use the constraints (SDC or XDC) output by the logic synthesis tools in the Vivado Design Suite.

Running Logic Simulation

Logic simulation tools supplied by Mentor Graphics, Cadence, and Synopsys are integrated and can be launched directly from the Vivado IDE. Netlists can also be produced for all supported third-party logic simulators. From the Vivado Design Suite, you can export complete Verilog or VHDL netlists at any stage of the design flow for use with third-party simulators. In addition, you can export post-implementation delays in standard delay format (SDF) for use in third-party timing simulation.

VIDEO: *For more information see the* Vivado Design Suite QuickTake Video: Simulating with Cadence IES in Vivado *and* Vivado Design Suite QuickTake Video: Simulating with Synopsys VCS in Vivado

Note: Some Xilinx IP provides RTL sources in only Verilog or VHDL format. After synthesis, structural netlists can be created in either language.

Using Project Mode

Overview

In Project Mode, the Vivado® Design Suite creates a project directory structure and automatically manages your source files, constraints, IP data, synthesis and implementation run results, and reports. In this mode, the Vivado Design Suite also manages and reports on the status of the source files, configuration, and the state of the design.

You can create RTL-based projects or synthesized, netlist-based projects. Netlist projects are primarily used with third-party synthesis tools, and the design process is managed from a post-synthesis perspective. You can analyze the netlist design, assign and manage constraints, implement and analyze the design, program and debug the device, and manage the sources and outputs for the entire flow.

In the Vivado IDE, you can use the Flow Navigator (Figure 3-1, page 24) to launch predefined design flow steps, such as synthesis and implementation. When you click **Generate Bitstream**, the Vivado IDE ensures that the design is synthesized and implemented and generates a bitstream file. The environment provides an intuitive pushbutton design flow and also offers advanced design management and analysis features. Runs are launched with wrapper Tcl scripts that consolidate the various implementation commands and automatically generates standard reports. You can use various run strategies to address different design challenges, such as routing density and timing closure. You can also simultaneously launch multiple implementation runs to see which will achieve the best results.

Note: Run strategies only apply to Project Mode. In Non-Project Mode, all directives and command options must be set manually.

You can run Project Mode using the Vivado IDE or using Tcl commands or scripts. In addition, you can alternate between using the Vivado IDE and Tcl within a project. When you open or create projects 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.

RECOMMENDED: *Project Mode is the easiest way to get acquainted with features of the Vivado tools and Xilinx® recommendations.*

You can open designs for analysis and constraints definition after RTL elaboration, synthesis, and implementation. When you open a design, the Vivado tools compile the netlist and constraints against the target device and show the design in the Vivado IDE. After you open the design, you can use a variety of analysis and reporting features to analyze the design using different criteria and viewpoints. You can also apply and save constraint and design changes. For more information, see *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

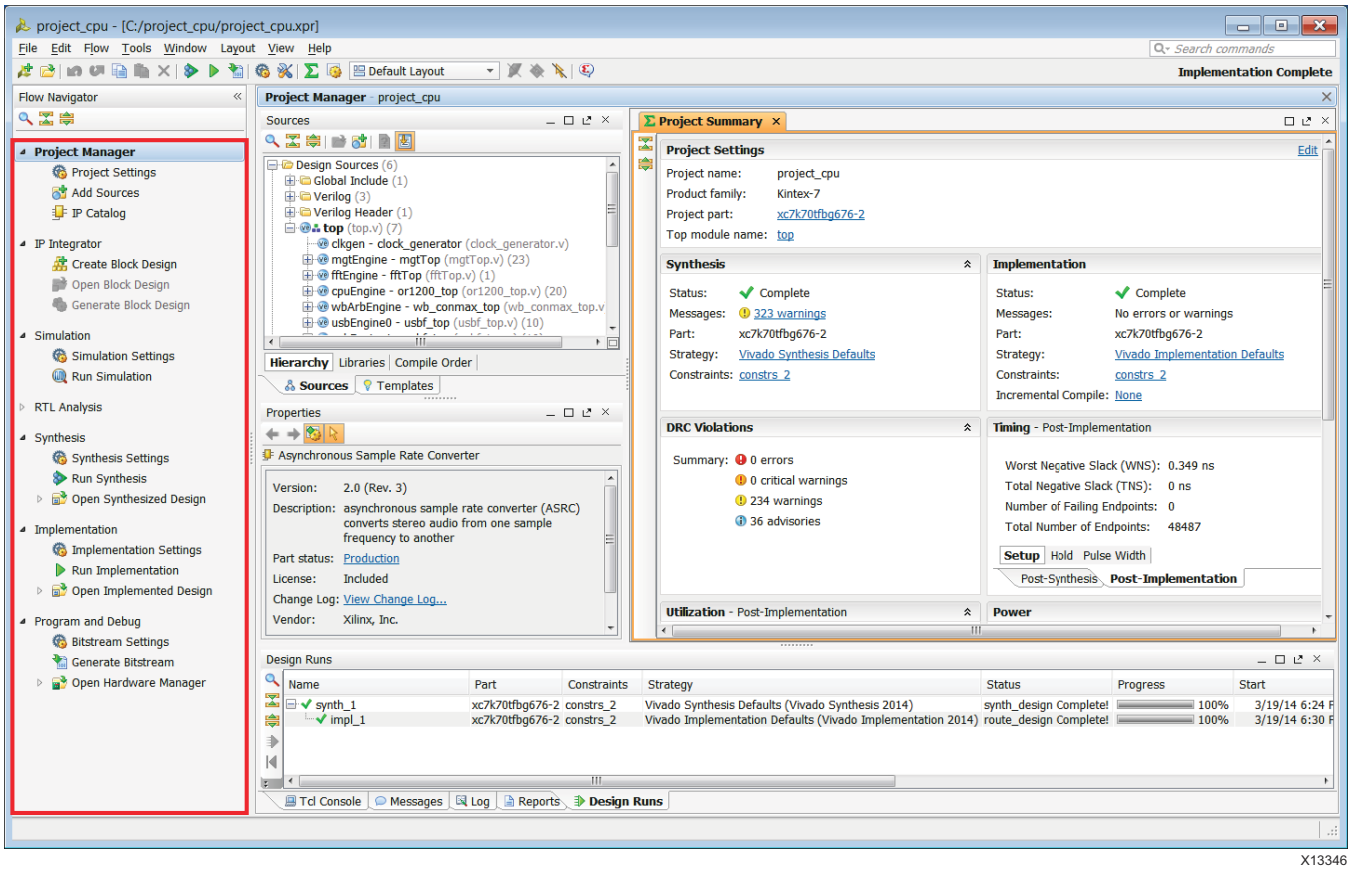

Figure 3-1: **Flow Navigator in the Vivado IDE**

Project Mode Advantages

Project Mode has the following advantages:

- Automatically manages project status, HDL sources, constraint files, IP cores and block designs.
- Generates and stores synthesis and implementation results
- Includes advanced design analysis capabilities, including cross probing from implementation results to RTL source files
- Automates setting command options using run strategies and generates standard reports
- Supports the creation of multiple runs to configure and explore available constraint or command options

Creating Projects

The Vivado Design Suite supports different types of projects for different design purposes. For example, you can create a project with RTL sources or synthesized netlists from third-party synthesis providers. You can also create empty I/O planning projects to enable device exploration and early pin planning. The Vivado IDE only displays commands relevant to the selected project type.

In the Vivado IDE, the Create Project wizard walks you through the process of creating a project. The wizard enables you to define the project, including the project name, the location in which to store the project, the project type (for example, RTL, netlist, and so forth), and the target part. You can add different types of sources, such as RTL, IP, XDC or SDC constraints, simulation test benches, DSP modules from System Generator (XMP) or Vivado High-Level Synthesis (HLS), and design documentation. When you select sources, you can determine whether to reference the source in its original location or to copy the source into the project directory. The Vivado Design Suite tracks the time and date stamp of each file and report status. If files are modified, you are alerted to out-of-date source or design status. For more information, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 12].

CAUTION! *The Windows operating system has a 260 character limit for path lengths which can affect the Vivado tools. To avoid this issue, use the shortest possible names and directory locations when creating projects, defining IP or managed IP projects, or creating block designs.*

Different Types of Projects

The Vivado Design Suite allows for different design entry points depending on your source file types and design tasks. Following are the different types of projects you can use to facilitate those tasks:

- **RTL Project**: You can add RTL source files and constraints, configure IP with the Vivado IP catalog, create IP subsystems with the Vivado IP integrator, synthesize and implement the design, and perform design planning and analysis.
- **Post-Synthesis Project**: You can import third-party netlists, implement the design, and perform design planning and analysis.
- **I/O Planning Project**: You can create an empty project for use with early I/O planning and device exploration prior to having RTL sources.
- **Imported Project**: You can import existing project sources from the ISE Design Suite, Xilinx Synthesis Technology (XST), or Synopsys Synplify.
- **Configure an Example Embedded Evaluation Board Design**: Enables you to target the example Zynq®-7000 or MicroBlaze™ embedded designs to the available Xilinx evaluation boards.
- **Partial Reconfiguration Project**: If you have a license for partial reconfiguration, RTL projects include an option that enables the partial reconfiguration design flow and commands.

Managing Source Files in Project Mode

In Project Mode, source management is performed by the project infrastructure. The Vivado IDE manages different types of sources independently, including RTL design sources, IP, simulation sources, and constraint sources. It uses the concept of a source set to enable multiple versions of simulation or design constraints sets. This enables you to manage and experiment with different sets of design constraints in one design project. The Vivado IDE also uses the same approach for simulation, enabling management of module-level simulation sets for simulating different parts of the design.

When adding sources, you can reference sources from remote locations or copy sources locally into the project directory structure. Sources can be read from any network accessible location. With either approach, the Vivado IDE tracks the time and date stamps on the files to check for updates. If source files are modified, the Vivado IDE changes the project status to indicate whether synthesis or implementation runs are out of date. Sources with read-only permissions are processed accordingly.

When adding sources in the Vivado IDE, RTL files can optionally be scanned to look for include files or other global source files that might be in the source directory. All source file types within a specified directory or directory tree can be added with the **File > Add Sources** command. The Vivado IDE scans directories and subdirectories and imports any file with an extension matching the set of known sources types.

After sources are added to a project, the compilation order is derived and displayed in the Sources window. This can help you to identify malformed RTL or missing modules. The Messages window shows messages related to the RTL compilation, and you can cross probe from the messages to the RTL sources. In addition, source files can be enabled and disabled to allow for control over configuration.

Using Remote, Read-Only Sources

The Vivado Design Suite can utilize remote source files when creating projects or when read in Non-Project Mode. Source files can be read-only, which compiles the files in memory but does not allow changes to be saved to the original files. Source files can be saved to a different location if required.

Archiving Projects

In the Vivado IDE, the **File > Archive Project** command creates a ZIP file for the entire project, including the source files, IP, design configuration, and optionally the run result data. If the project uses remote sources, the files are copied into the project locally to ensure that the archived project includes all files.

Creating a Tcl Script to Recreate the Project

In the Vivado IDE, the **File > Write Project Tcl** command creates a Tcl script you can run to recreate the entire project, including the source files, IP, and design configuration. You can check this script into a source control system in place of the project directory structure.

Working with a Revision Control System

Many design teams use source management systems to store various design configurations and revisions. There are multiple commercially available systems, such as Revision Control System (RCS), Concurrent Versions System (CVS), Subversion (SVN), ClearCase, Perforce, Git, BitKeeper, and many others. The Vivado tools can interact with all such systems. The Vivado Design Suite uses and produces files throughout the design flow that you may want to manage under revision control.

Using Project mode in the Vivado Design Suite can complicate the interaction with a source control system. The project folder maintains its own copy of design sources and provides its own design management. However, you can manage a project with a revision control system. For more information on working with revision control software, see this link in the *UltraFast™ Design Methodology Guide for the Vivado Design Suite* (UG949) [Ref 22].

VIDEO: *For information on best practices when using revision control systems with the Vivado tools, see the* Vivado Design Suite QuickTake Video: Version Control Overview.

Understanding the Flow Navigator

The Flow Navigator (Figure 3-2) 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. Unavailable steps are grayed out until required design tasks are completed.

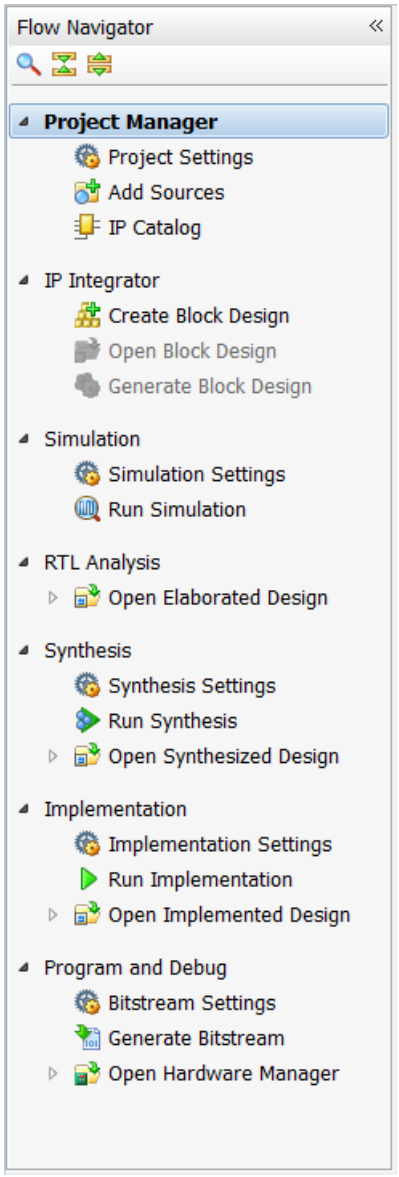

Figure 3-2: **Flow Navigator**

The Flow Navigator (Figure 3-3) differs when working with projects created with third-party netlists. For example, system-level design entry, IP, and synthesis options are not available.

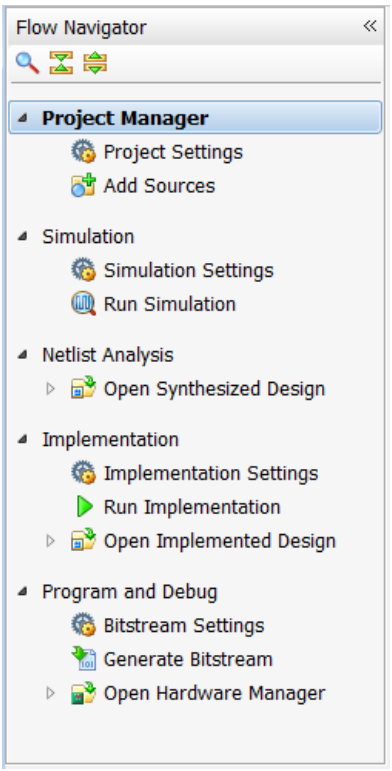

Figure 3-3: **Flow Navigator for Third-Party Netlist Project**

As the design tasks complete, you can open the resulting designs to analyze results and apply constraints. In the Flow Navigator, click **Open Elaborated Design**, **Open Synthesized Design**, or **Open Implemented Design**. For more information, see Opening Designs to Perform Design Analysis and Constraints Definition.

When you open a design, the Flow Navigator shows 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, page 30 shows the commands related to synthesis.

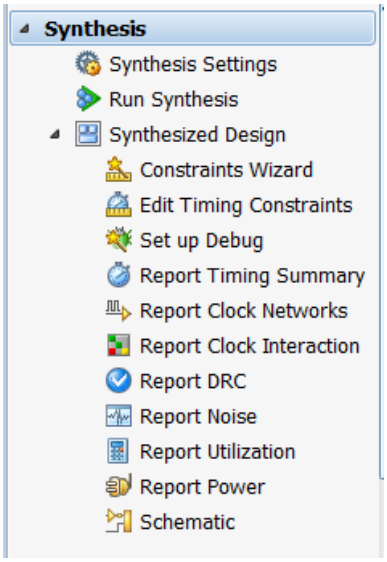

Figure 3-4: **Synthesis Section in the Flow Navigator**

Performing System-Level Design Entry

Automated Hierarchical Source File Compilation and Management

The Vivado IDE Sources window (Figure 3-5) provides automated source file management. The window has several views to display the sources using different methods. When you open or modify a project, the Sources window updates the status of the project sources. A quick compilation of the design source files is performed and the sources appear in the Compile Order view of the Sources window in the order they will be compiled by the downstream tools. Any potential issues with the compilation of the RTL hierarchy are shown as well as reported in the Message window. For more information on sources, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 12].

Note: If you explicitly set a module as the top module, the module is retained and passed to synthesis. However, if you do not explicitly set a top module, the Vivado tools select the best possible top module from the available source files in the project. If a file includes syntax errors and does not elaborate, this file is *not* selected as the top module by the Vivado tools.

Constraints and simulation sources are organized into sets. You can use constraint sets to experiment with and manage constraints. You can launch different simulation sessions using different simulation source sets. You can add, remove, disable, or update any of the sources. For more information on constraints, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 23]. For more information on simulation, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 14].

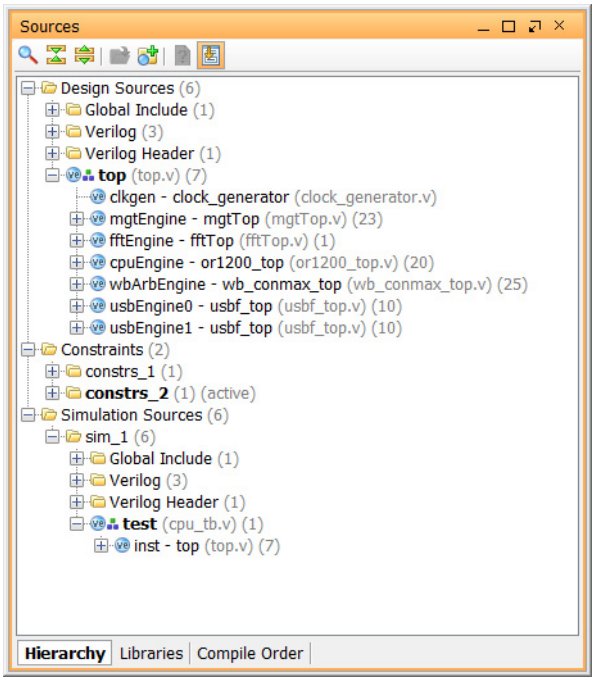

Figure 3-5: **Hierarchical Design Source File Compilation in the Sources Window**

RTL Development

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

- Integrated Vivado IDE Text Editor to create or modify source files
- Language templates for copying example logic constructs
- Find in Files feature for searching template libraries using a variety of search criteria
- RTL elaboration and interactive analysis
- RTL design rule checks
- RTL constraints assignment and I/O planning

RTL Elaboration and Analysis

When you open an elaborated RTL design, the Vivado IDE compiles the RTL source files and loads the RTL netlist for interactive analysis. 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
- 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 on RTL development and analysis features, see the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 23]. For more information on RTL-based I/O planning, see the *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 11].

Timing Constraint Development and Verification

The Vivado IDE provides a Timing Constraints wizard to walk you through the process of creating and validating timing constraints for the design. The wizard identifies clocks and logic constructs in the design and provides an interface to enter and validate the timing constraints in the design. It is only available in synthesized and implemented designs, because the in-memory design must be clock aware post-synthesis. For more information, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 23].

Note: The Vivado Design Suite only supports Synopsys design constraints (SDC) and Xilinx design constraints (XDC). It does *not* support Xilinx user constraints files (UCF) used with the ISE Design Suite nor does it directly support Synplicity design constraints. For information on migrating from UCF format to XDC format, see this link in the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 24].

Working with IP

In the Vivado IDE, 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 can be packaged following IP-XACT protocol 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 extension). For more information, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Creating IP Management Locations

You can configure and manage Vivado Design Suite IP in a remote IP location. To create or open an IP location, select **Manage IP** in the Vivado IDE Getting Started page. When you create an IP location, the following is created:

- A directory structure that separates and maintains the various IP sources and output products
- A project named `manage_ip_project`

When using this type of project, you can:

- Configure IP using the Vivado IP catalog.
- Manage the output product generation and IP validation processes.
- Generate IP output products for individual IP or for multiple IP concurrently.
- View and manage IP output products in the Sources window of the Vivado IDE.
- Interactively perform IP version upgrades.

For more information, see this link in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Configuring IP with the Vivado IP Catalog

The Vivado IP catalog (Figure 3-6) enables you to browse the available Xilinx LogiCORE™ IP for the target device selected in the project. The IP catalog shows version and licensing information about each IP and provides the applicable data sheet. You can double-click any IP to launch the Configuration wizard and begin the IP configuration and instantiation process for your design.

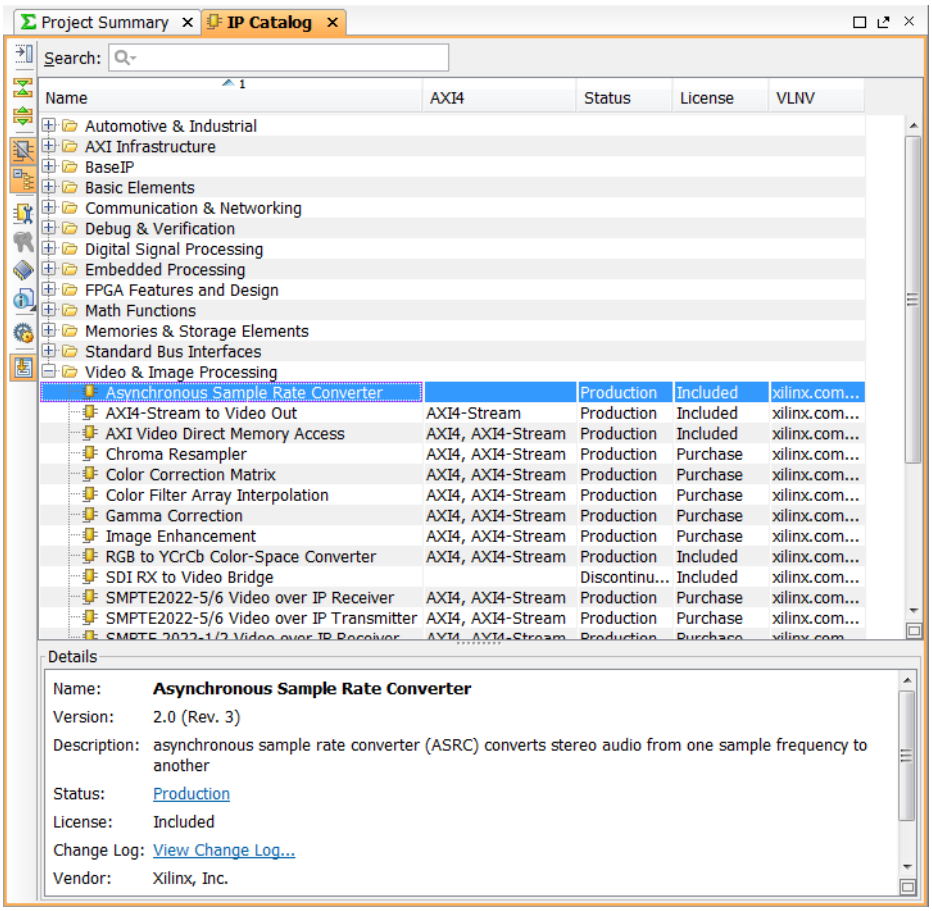

Figure 3-6: **Vivado IP Catalog**

Generating Output Products

Vivado Design Suite IP includes a variety of output products for use in synthesis, implementation, and validation of the IP. You can generate these output products immediately following IP configuration, or you can generate the output products at a later time. In Project Mode, missing output products are automatically generated during synthesis, including a synthesized design checkpoint (DCP) file for the out-of-context flow. In Non-Project Mode, the output products must be generated prior to execution of the `synth_design` Tcl command.

Vivado Design Suite IP includes the following output products:

- Instantiation template
- RTL source files and XDC constraints
- Synthesized design checkpoint (optional,default)
- Third-party simulation sources
- Third-party synthesis sources
- Example design (for applicable IP)
- Test bench (for applicable IP)
- C Model (for applicable IP)

Synthesizing IP Standalone for an Out-of-Context Design

By default, a design checkpoint is created for each IP, which contains the synthesized netlist and constraints for the IP module. The IP module is synthesized with Vivado synthesis, and the design checkpoint is used during implementation. Vivado synthesis inserts a black box stub file for each IP that includes a design checkpoint. This allows you to optimize and validate IP separately and reduce synthesis runtimes. This is referred to as an out-of-context design.

Disabling design checkpoint generation forces a global top-down synthesis approach where the IP and the top-level design are synthesized together. In some cases, you can perform additional logic optimization during synthesis when this approach is used.

TIP: *In the Vivado IDE, the Generate Output Products dialog box contains the Generate Design Checkpoint (.dcp) option, which you can use to control the default behavior.*

Verifying the IP

You can verify Vivado IP by running either behavioral or structural logic simulation and by implementing the IP module to validate timing, power, area, and so forth. Typically, a smaller top-level design project containing an RTL wrapper and test bench are created to validate the standalone IP. The Vivado Design Suite also enables validating the IP module within the context of the top-level design project. Because the IP creates synthesized design checkpoints, this bottom-up verification strategy works well either standalone or within a project.

An example design is optionally created as a part of IP core generation. To create an example design, select the IP in the Sources > IP Sources window and use the **Open IP Example Design** popup menu command. The example IP module enables you to verify the standalone IP within the context of the example design project.

For more information, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 14], *Vivado Design Suite User Guide: Synthesis* (UG901) [Ref 2], and *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Creating IP Subsystem Block Designs

The Vivado IP integrator (Figure 3-7, page 37) enables multiple IP to be stitched together using AXI4 interconnect protocol. You can select compliant IP from the Vivado IP catalog and instantiate the IP onto the design canvas. You can then double-click the IP to invoke the Configuration wizard. Drag and drop interconnect is DRC-correct and provides visual assistance to locate compatible pins. You can connect entire AXI interfaces with one wire and place ports and interface ports to connect the IP subsystem to the rest of the design. These IP block designs can then be packaged as sources (.bd) and reused in other designs. For more information, see the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 4] or *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898) [Ref 5].

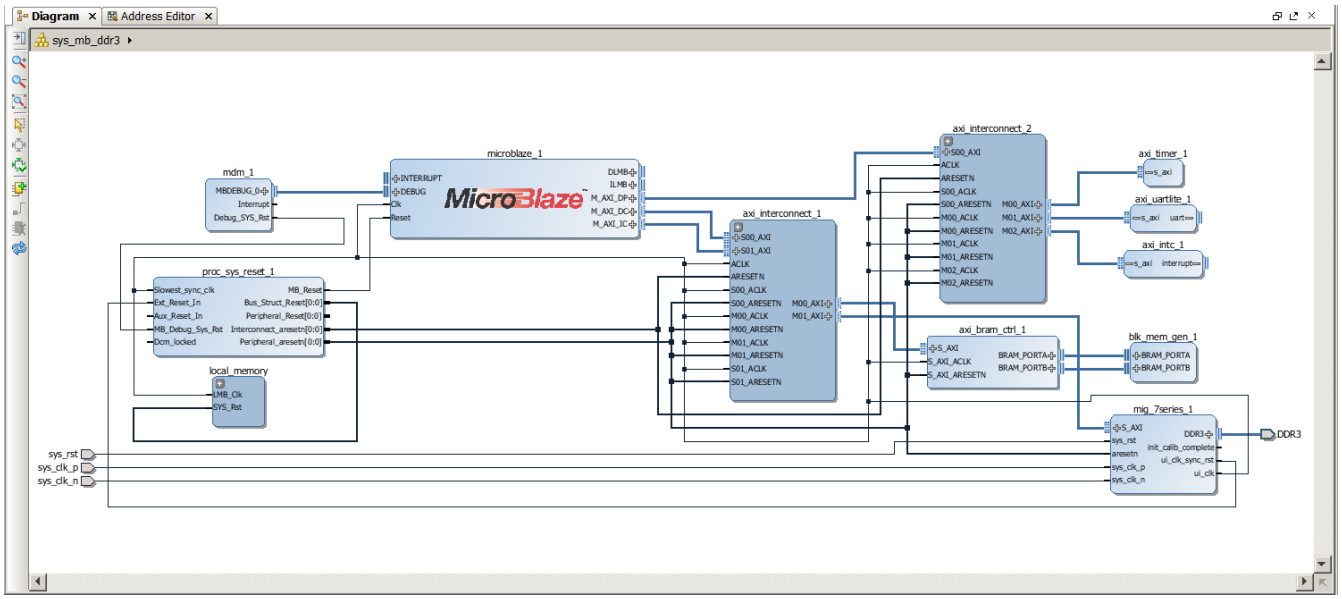

Figure 3-7: **Vivado IP Integrator**

Creating and Packaging Custom IP

The Vivado IDE enables you to package custom IP or IP block designs into IP. You can then access the IP from the Vivado IP catalog for use in designs or in the Vivado IP integrator. You can package IP from a variety of sources, such as from a collection of RTL source files, a Vivado IP integrator block design, or an entire Vivado Design Suite project. There are multiple ways to configure the IP and make it available for use within the Vivado IP catalog and IP integrator. For example, the Package IP wizard walks you through the process of inputting information and gathering data to ensure the IP is complete, AXI4 compliant, configurable, and is applicable for use in the IP catalog. You can also create custom interface peripherals for use in embedded processor designs using the **Create and Package IP** command. For more information, see the *Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118) [Ref 25] and *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119) [Ref 26].

Upgrading IP

With each release of the Vivado Design Suite, new IP versions are introduced. You can either use the static version of the IP that is already configured or upgrade the IP. To report on the current status of your IP, you can run the `report_ip_status` Tcl command. If needed, you can selectively upgrade the IP to the latest version. A change log details the changes made and lists any design updates that are required. For example, top-level port changes are occasionally made in newer IP versions, so some design modification might be required. You can also use the `report_ip_status` Tcl command to change the target device for an existing IP.

Note: To use the static version of the IP, all output products must be created using the same version of software that was used for the IP. Reconfiguration of out-of-date IP is not allowed in newer versions of the Vivado Design Suite. If an upgrade is not performed, the IP appears with a lock icon indicating that it cannot be reconfigured.

Running Logic Simulation

The Vivado Design suite has several logic simulation options for verifying designs or IP. 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. You can use the Vivado simulator to perform behavioral and structural simulation of designs as well as full timing simulation of implemented designs. Alternatively, you can use third-party simulators by writing the Verilog, VHDL netlists, and SDF format files from the open design. From the Vivado IDE, you can launch simulators from Mentor Graphics, Synopsys, and Cadence. For more information, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 14].

I/O Pin Planning

The Vivado IDE provides an I/O pin planning environment that enables I/O port assignment either onto specific package pins or onto internal die pads. The Vivado IDE provides display windows and tables in which you can analyze and design package and design I/O-related data. The tool also provides I/O DRC and simultaneous switching noise (SSN) analysis commands to validate your I/O assignments. For more information, see the *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 11].

Running Logic Synthesis and Implementation

Logic Synthesis

Vivado synthesis enables you to configure, launch, and monitor synthesis runs. The Vivado IDE displays the synthesis results and creates report files. You can select synthesis warnings and errors from the Log window to highlight the logic in the RTL source files.

You can launch multiple synthesis runs concurrently 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 open different versions of the synthesized netlist in the Vivado IDE to perform device and design analysis. You can also create constraints for I/O pin planning, timing, floorplanning, and implementation. The most comprehensive list of DRCs 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) [Ref 2].

Implementation

Vivado implementation enables you to configure, launch, and monitor implementation runs. You can experiment with different implementation options and create your own reusable strategies for implementation runs. For example, you can create strategies for quick runtimes, improved system performance, or area optimization. As the runs complete, implementation run results display and report files are available.

You can launch multiple implementation runs either simultaneously or serially. On a Linux system, you can use remote servers. You can create constraint sets to experiment with various timing constraints, physical constraints, or alternate devices. For more information, see the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3] and *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 23],

TIP: *You can add Tcl scripts to be sourced before and after synthesis, any stage of implementation, or bitstream generation using the* `tcl.pre` *and* `tcl.post` *files. For more information, see the Vivado Design Suite User Guide: Using Tcl Scripting (UG894) [Ref 18].*

Configuring Synthesis and Implementation Runs

When using Project Mode, various settings are available to control the features of synthesis and implementation. These settings are passed to runs using run strategies, which you set in the Project Settings dialog box. A run strategy is simply a saved set of run configuration parameters. Xilinx supplies several pre-defined run strategies for running synthesis and implementation, or you can apply custom run settings. In addition, you can use separate constraint sets for synthesis and implementation.

For information on modifying project settings, see this link to the *Vivado Design Suite User Guide: Synthesis* (UG901) [Ref 2] and see this link in the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

TIP: *You can create an out-of-context module run to synthesize the Vivado Design Suite IP in the project. If you generate a design checkpoint for the IP, the default behavior is to create an out-of-context run for each IP in the design.*

Creating and Managing Runs

After the synthesis and implementation settings are configured in the Project Settings dialog box, you can launch synthesis or implementation runs using any of the following methods:

- In the Flow Navigator, select **Run Synthesis**, **Run Implementation**, or **Generate Bitstream**.
- In the Design Runs window, select a run, right-click, and select **Launch Runs**. Alternatively, you can click the **Launch Selected Runs** button.
- Select **Flow > Run Synthesis**, **Flow > Run Implementation**, or **Flow > Generate Bitstream**.

You can create multiple synthesis or implementation runs to experiment with constraints or tool settings. To create additional 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 (Figure 3-8, page 41), select the constraint set and target part.

If more than one synthesis run exists, you can also select the netlist when creating implementation runs. You can then create one or more runs with varying strategies, constraint sets, or devices. There are several launch options available when multiple runs exist. You can launch selected runs sequentially or in parallel on multiple local processors.

Note: You can configure and use remote hosts on Linux systems only.

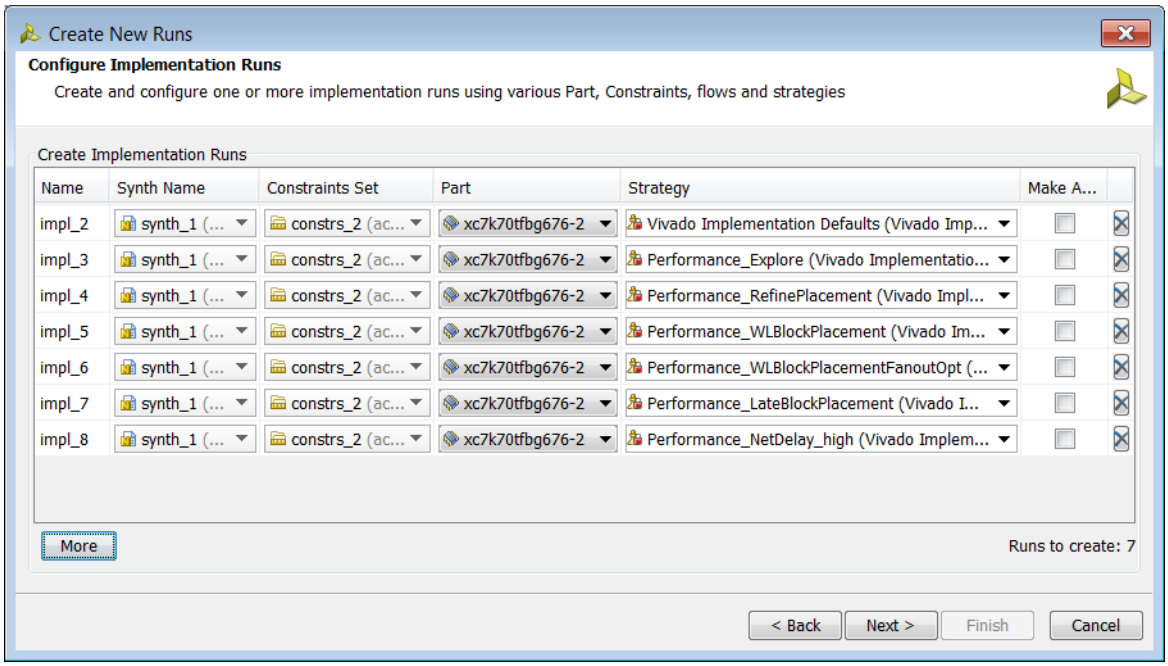

Figure 3-8: **Creating Multiple Synthesis and Implementation Runs**

Managing Runs with the Design Runs Window

The Design Runs windows (Figure 3-9) displays run status and information and provides access to run management commands 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 run using the **Make Active** popup menu command. The Vivado IDE updates results to reflect the information about the newly designated active run. Double-click any synthesized or implemented run to open the design in the Vivado IDE.

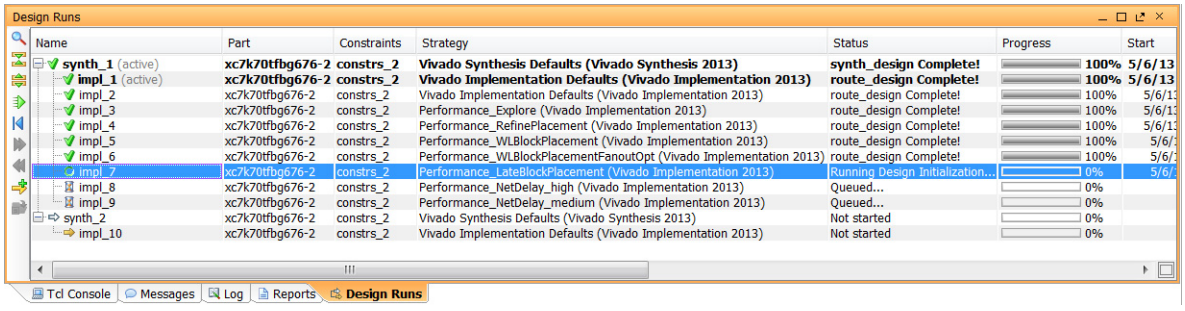

Figure 3-9: **Design Runs Window**

Resetting Runs

In the Flow Navigator, you can right-click **Synthesis** or **Implementation**, and use the following popup menu commands to reset runs. For more information, see this link in the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

- **Reset Runs** resets the run to its original state and optionally deletes generated files from the run directory.
- **Reset to Previous Step** resets the run to the listed step.

TIP: *To stop an in-process run, click the **Cancel** button in the upper right corner of the Vivado IDE.*

Accessing a Remote Server Farm Through LSF

To launch runs on remote Linux hosts, you can directly access a load sharing facility (LSF) server farm. For information, see this link in the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

Performing Implementation with Incremental Compile

You can perform Vivado implementation using incremental compile to facilitate small design changes. Incremental compile can reduce runtime and preserve existing results depending on the scope of the change and the amount of timing-critical logic that is modified. Use the `read_checkpoint` Tcl command with the `-incremental` option, and point to a routed design checkpoint to use as a reference. For more information, see this link in the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

Viewing Log Files, Messages, Reports, and Properties

Viewing Log Files

In the Log window (Figure 3-10, page 43), you can click the different tabs to view the standard output for **Synthesis**, **Implementation**, and **Simulation**. This output is also included in the `vivado.log` file that is written to the Vivado IDE launch directory.

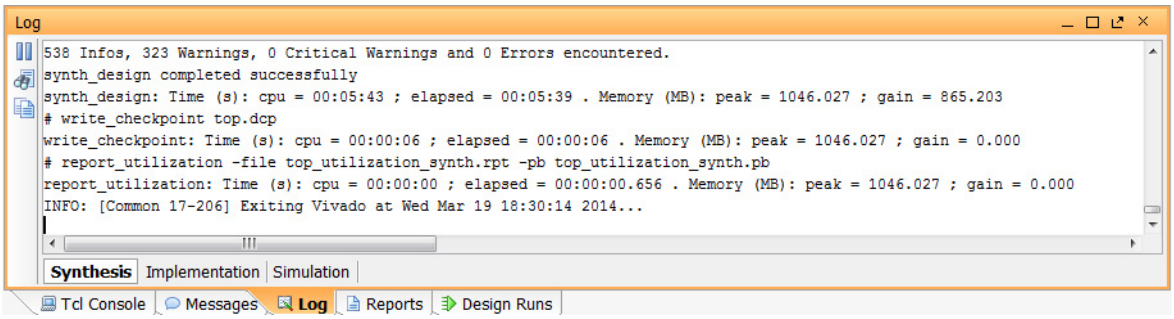

Figure 3-10: **Viewing Log Files**

Viewing Messages

In the Messages window (Figure 3-11), messages are categorized according to design step and severity level: Errors, Critical Warnings, Warnings, Info, and Status. To filter messages, select the appropriate check boxes in the window header. You can expand the message categories to view specific messages. You can click the **Collapse All** icon to show only the main design steps. This enables better navigation to specific messages. Many messages include links that take you to logic lines in the RTL files. For more information, including advanced filtering techniques, see this link in the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 21].

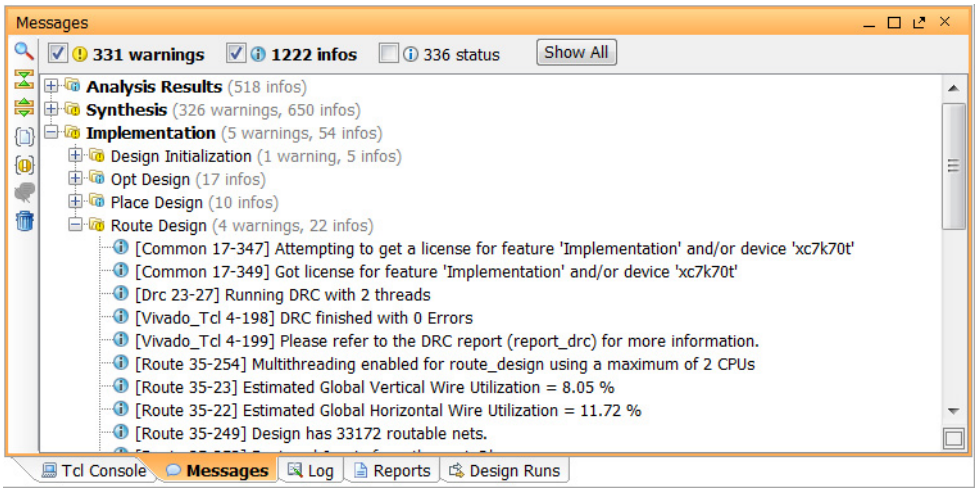

Figure 3-11: **Viewing Messages**

Viewing Reports

In the Reports window (Figure 3-12), several standard reports are generated using the `launch_runs` Tcl commands. You can double-click any report to display it in the Vivado IDE Text Editor. You can also create custom reports using Tcl commands in the Tcl Console. For more information, see this link in the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 21] and see this link in *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

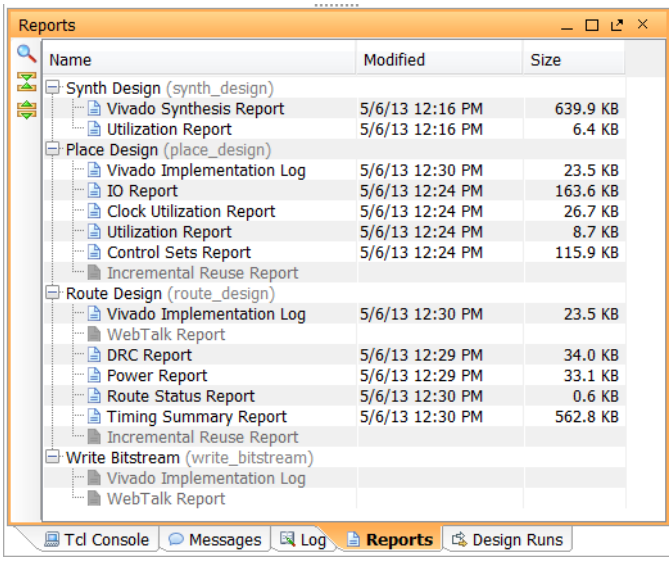

Figure 3-12: **Viewing Reports**

Viewing Device Properties

With the elaborated, synthesized, or implemented design open, you can use the **Tools > Edit Device Properties** command to open the Edit Device Properties dialog box (Figure 3-13) in which you can view and set device configuration and bitstream-related properties. This command is available only when a design is open. For information on each property, see the link in the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 15]. For information on setting device configuration modes, see this link in the *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 11].

Note: The **Edit > Device Properties** is only available when a design is open.

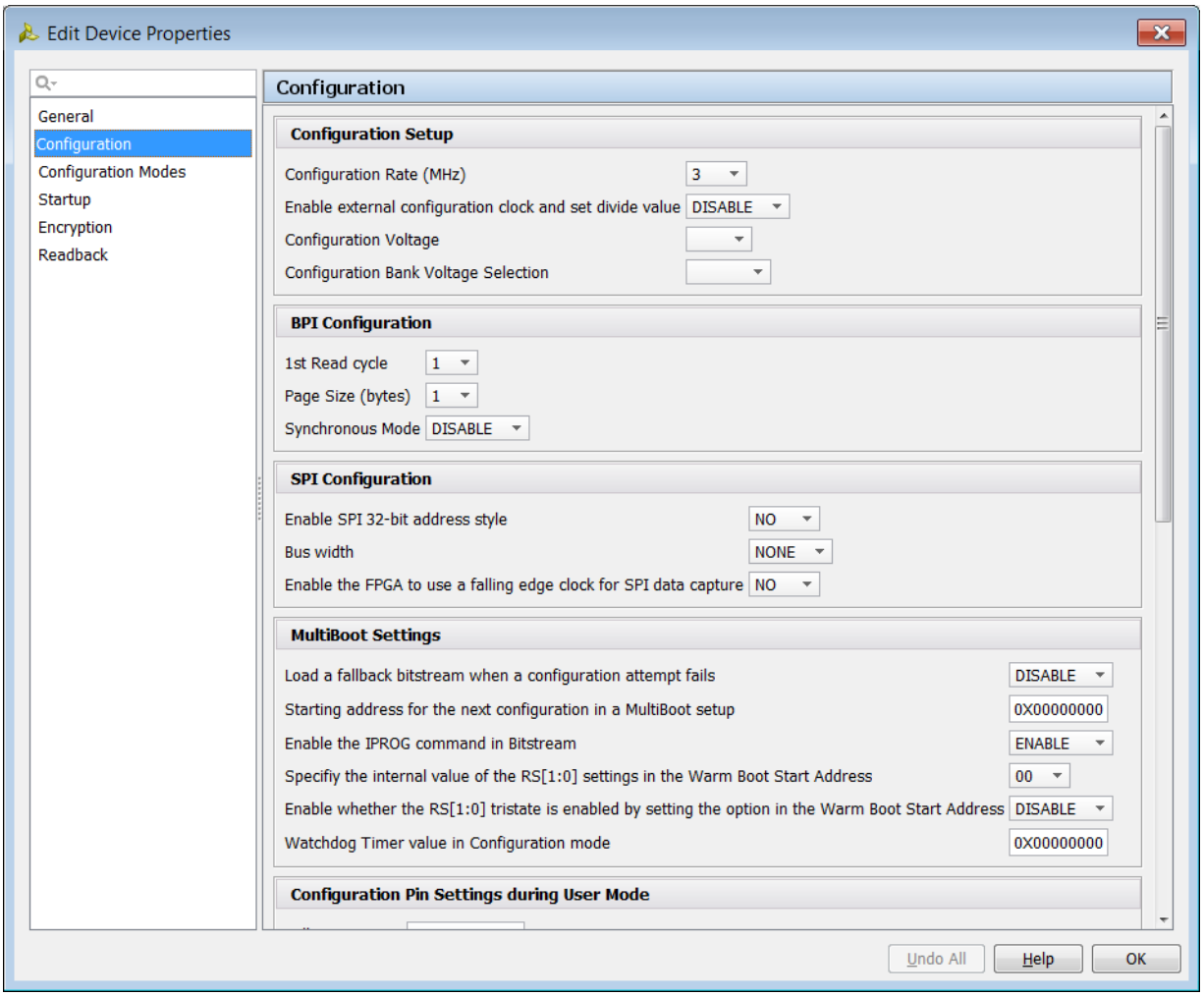

Figure 3-13: **Viewing Device Properties**

Opening Designs to Perform Design Analysis and Constraints Definition

You can perform design analysis and assign constraints after RTL elaboration, after synthesis, or after implementation. To identify design issues early, you can perform design analysis prior to implementation, including timing simulation, resource estimation, connectivity analysis, and DRCs. You can open the various synthesis or implementation run results for analysis and constraints assignment. This is known as opening the design.

When you open the design, the Vivado IDE compiles the netlist and applies physical and timing constraints against a target part. You can open, save, and close designs. 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. When you open a synthesized design, the Vivado IDE displays the netlist and constraints. When you open an implemented design, the Vivado IDE displays the netlist, constraints, and implementation results. The design data is presented in different forms in different windows, and you can cross probe and coordinate data between windows.

After opening a design, many analysis and reporting features are available in the Vivado IDE. For example, you can analyze device resources in the graphical windows of the internal device and the external physical package. You can also apply and analyze timing and physical constraints in the design using the Netlist, Device, Schematic, or Hierarchy windows. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13] and *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 23].

Note: If you make constraint changes 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 this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 12].

Opening an Elaborated RTL Design

When you open an elaborated design, the Vivado Design Suite expands and compiles the RTL netlist and applies physical and timing constraints against a target part. The different elements of the elaborated design are loaded into memory, and you can analyze and modify the elements as needed to complete the design. For more information, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 12].

The Vivado Design Suite includes linting DRCs and checking tools that enable you to analyze your design for logic correctness. You can make sure that there are no logic compilation issues, no missing modules, and no interface mismatches. In the Messages window, you can click links in the messages to display the problem lines in the RTL files in the Vivado IDE Text Editor. In the Schematic window, you can explore the logic interconnects and hierarchy in a variety of ways. The Schematic window displays RTL interconnects using RTL-based logic constructs. You can select logic in the Schematic window and see specific lines in the RTL files in the Vivado IDE Text Editor. For more information, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 12].

Note: There is no FPGA technology mapping during RTL elaboration.

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. However, when you run synthesis on the design, these issues are resolved.

Using the I/O planning capabilities of the Vivado IDE, you can interactively configure and assign I/O Ports in the elaborated RTL design and run DRCs. When possible, it is recommended that you perform I/O planning after synthesis. This ensures proper clock and logic constraint resolution, and the DRCs performed after synthesis are more extensive. For more information, see *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 11].

TIP: *When you select the* ***Report DRC*** *command, the Vivado IDE invokes a set of RTL and I/O DRCs to identify logic issues such as asynchronous clocks, latches, and so forth. For more information, see this link in the Vivado Design Suite User Guide: System-Level Design Entry (UG895) [Ref 12].*

To open an elaborated design, use one of the following methods:

- In the RTL Analysis section of the Flow Navigator, select **Open Elaborated Design**.
- In the Flow Navigator, right-click **RTL Analysis**, and select **New Elaborated Design** from the popup menu.
- Select **Flow > Open Elaborated Design**.

Figure 3-14 shows the default view layout for an open elaborated RTL design. Notice the logic instance that was cross-selected from the schematic to the specific instance in the RTL source file and within the elaborated RTL netlist.

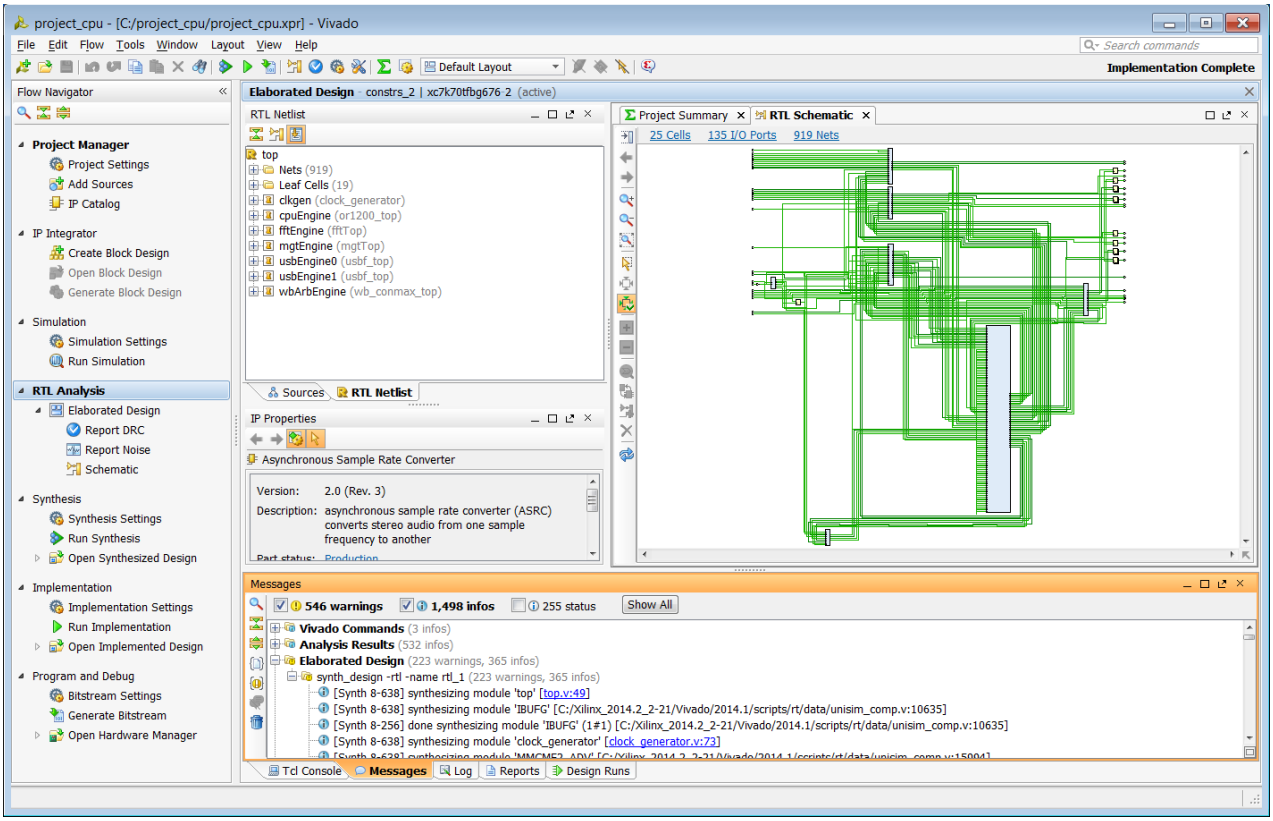

Figure 3-14: **Open Elaborated RTL Design**

Opening a Synthesized Design

When you open a synthesized design, the Vivado Design Suite opens the synthesized netlist and applies physical and timing constraints against a target part. The different elements of the synthesized design are loaded into memory, and you can analyze and modify these elements as needed to complete the design. You can save updates to the constraints files, netlist, debug cores, and configuration.

In a synthesized design, you can perform many design tasks, including early timing, power, and utilization estimates that can help you determine if your design is converging on desired targets. You can explore the design in a variety of ways using the windows in the Vivado IDE. Objects are always cross-selected in all other windows. You can cross probe to problem lines in the RTL files from various windows, including the Messages, Schematic, Device, Package, and Find windows. The Schematic window allows you to interactively explore the logic interconnect and hierarchy. You can also apply timing constraints and perform further timing analysis. In addition, you can interactively define physical constraints for I/O ports, floorplanning, or design configuration. For more information, see

the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

Using the I/O planning capabilities of the Vivado IDE, you can interactively configure and assign I/O ports in the synthesized design and run DRCs. Select the **Run DRC** command to invoke a comprehensive set of DRCs to identify logic issues. For more information, see this link in the *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 11] and see this link in *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

You can configure and implement debug core logic in the synthesized design to support test and debug of the programmed FPGA device. In the Schematic or Netlist windows, interactively select signals 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. For more information, see this link in the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 15].

To open a synthesized design, use one of the following methods:

- In the Synthesis section of the Flow Navigator, select **Open Synthesized Design**.
- In the Flow Navigator, right-click **Synthesis**, and select **New Synthesized Design** from the popup menu.
- Select **Flow > Open Synthesized Design**.
- In the Design Runs view, double-click the run name.

Figure 3-15 shows the default view layout for an open synthesized design.

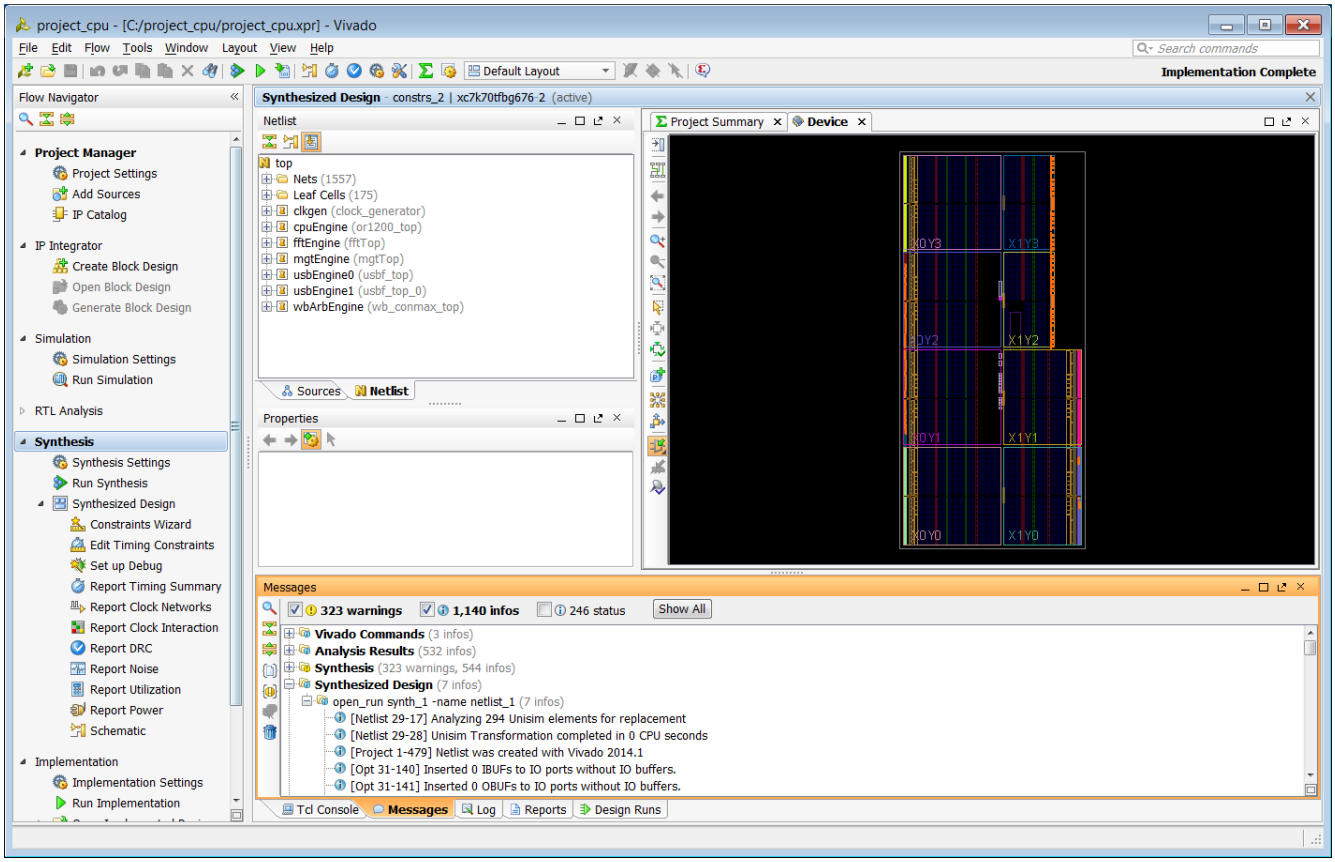

Figure 3-15: **Open Synthesized Design**

Opening an Implemented Design

When you open an implemented design in the Flow Navigator, the Vivado IDE opens the implemented netlist and applies the physical and timing constraints used during implementation, placement, and routing results against the implemented part. The placed logic and routed connections of the implemented design are loaded into memory, and you can analyze and modify the elements as needed to complete the design. You can save updates to the constraints files, netlist, implementation results, and design configuration. Because the Vivado IDE allows for multiple implementation runs, you can select any completed implementation run to open the implemented design.

In an implemented design, you can perform many design tasks, including timing analysis, power analysis, and generation of utilization statistics, which can help you determine if your design converged on desired performance targets. You can explore the design in a variety of ways using the windows in the Vivado IDE. Selected objects are always cross-selected in all related windows. You can cross probe to lines in the source RTL files from various windows, including the Messages, Schematic, Device, Package, and Find windows. The Schematic window allows you to interactively explore the logic interconnect and hierarchy. You can also apply timing constraints and perform further timing analysis. In addition, you can interactively apply floorplanning or design configuration constraints and save the constraints for future runs. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

In the Device window, you can explore the placement or the routing results by toggling the **Routing Resources** button . As you zoom, the amount of detail shown in the Device window increases. You can interactively alter placement and routing as well as design configuration, such as look-up table (LUT) equations and random access memory (RAM) initialization. You can also select results in the Device or Schematic windows to cross probe back to problem lines in the RTL files. In the Schematic window, you can interactively explore the logic interconnect and hierarchy. For more information, see the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

To open an implemented design, use one of the following methods:

- In the Implementation section of the Flow Navigator, click **Open Implemented Design**.
- Select **Flow > Open Implemented Design**.
- In the Design Runs view, double-click the run name.

TIP: *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, use the **Implementation** popup menu in the Flow Navigator to open an implemented design from any of the completed implementation runs.*

Figure 3-16 shows the default layout view for an open implemented design.

Note: The Device window might display placement only or routing depending on the state the window was in when it was last closed. In the Device window, click the **Routing Resources** button to toggle the view to display only placement or routing.

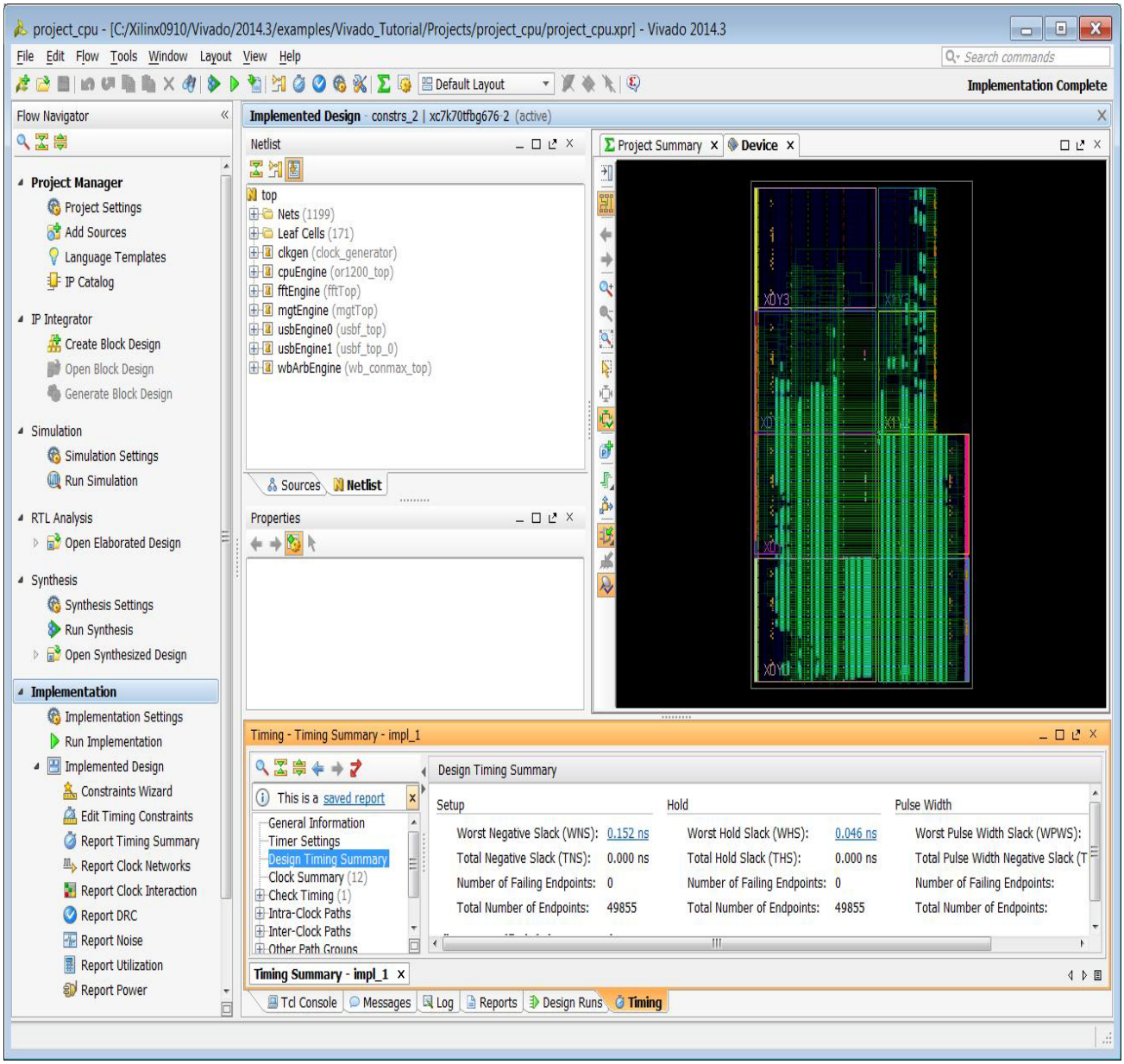

Figure 3-16: **Open Implemented Design**

Updating Out-of-Date Designs

During the design process, source files or constraints often require modification. The Vivado IDE manages the dependencies of these files and indicates when the design data in the current design is out of date. For example, changing project settings, such as the target part or active constraint set, can make a design out of date. As source files, netlists, or implementation results are updated, an out-of-date message is displayed in the design window banner of an open synthesized or implemented design to indicate that the run is out of date (Figure 3-17). Click the associated more info link to view which aspects of the design are out of date.

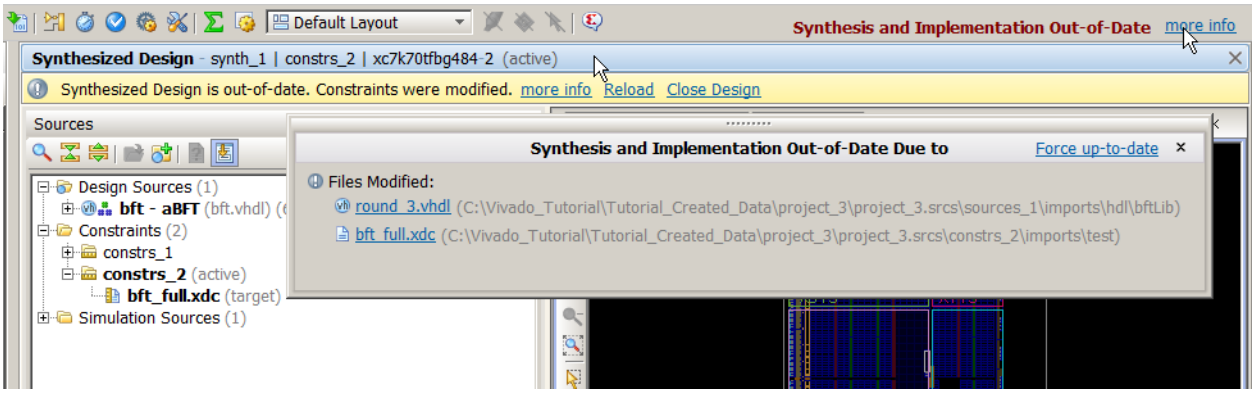

Figure 3-17: **Design Out-of-Date and Reload Banner**

From the design window banner, use any of the following actions to resolve an out-of-date design:

- Click **More Info**, and click the **Force up-to-date** link in the Out-of-Date Due to window that appears.

  Force up-to-date resets the `NEEDS_REFRESH` property on the active synthesis or implementation runs as needed to force the runs into an up-to-date state. The associated Tcl command 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.

- Click **Reload** to refresh the in-memory view of the current design, eliminating any unsaved changes you made to the design data.
- Click **Close Design** to close the out-of-date design.

Using View Layouts to Perform Design Tasks

When a design is open, several default view layouts (Figure 3-18) are provided to enable you to more easily work on specific design tasks, such as I/O planning, floorplanning, and debug configuration. Changing view layouts simply alters the windows that are displayed, which enables you to focus on a particular design task. You can also create custom view layouts using the **Save Layout As** command.

Note: Default view layouts are available only when a design is open.

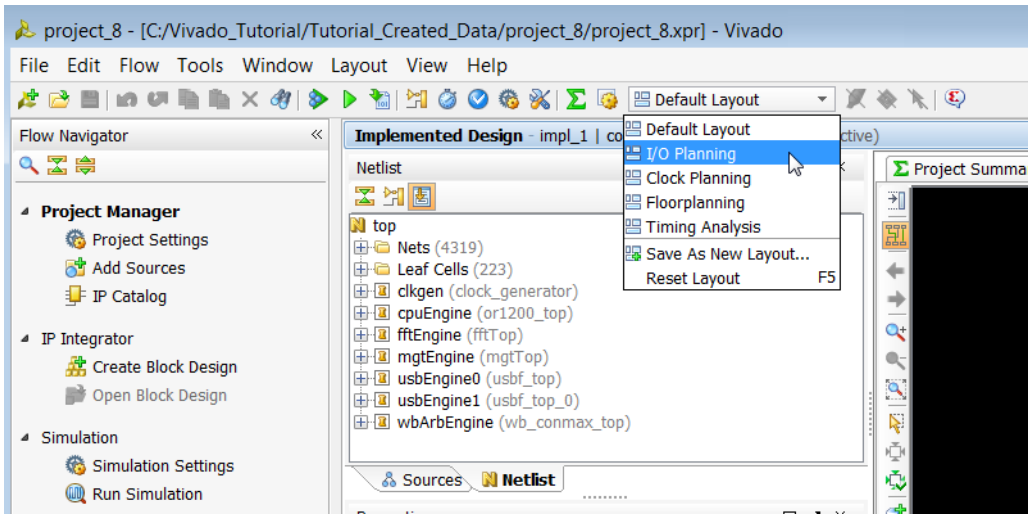

Figure 3-18: **Selecting a View Layout**

Saving Design Changes

In the Vivado IDE, you interactively edit the active design in memory. It is important to save the design when you make changes to constraints, netlists, and design parameters, such as power analysis characteristics, hardware configuration mode parameters, and debug configuration. For changes made while interactively editing an open design, you can save the changes either back to your original XDC constraint files or to a new constraint set as described in the following sections.

Saving Changes to Original XDC Constraint Files

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

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

To save changes to the design to a new constraint set, select **File > Save Constraints As** to create a new constraint file.

This saves any changes while preserving your original constraints source files. The new constraint set includes all design constraints, including all changes. This is one way to maintain your original XDC source files. You can also make the new constraint set the active constraint set, so that it is automatically applied to the next run or when opening designs.

Closing Designs

You can close designs to reduce the number of designs in memory and to prevent multiple locations where sources can be edited. In some cases, you are prompted to close a design prior to changing to another design representation. To close individual designs, do either of the following:

- In the design title bar, click the close button (**X**).
- In the Flow Navigator, right-click the design, and select **Close**.

Analyzing Implementation Results

When you open an implemented design, placement and routing results are displayed in the Device window. In the Timing Results window, you can select timing paths to highlight the placement and routing for the selected path in the Device window. You can also interactively edit placement and routing to achieve design goals and change design characteristics, such as LUT equations, RAM initialization, and phase-locked loop (PLL) configuration. For more information, see this link in the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

IMPORTANT: *Changes are made on the in-memory version of the implemented design only. Resetting the run causes changes to be lost. To save the changes, use the* ***Save Checkpoint*** *command, as described in Saving Design Changes to Design Checkpoints in Chapter 4.*

Running Timing Analysis

The Vivado IDE provides a graphical way to configure and view timing analysis results. You can experiment with various types of timing analysis parameters using **Tools > Timing** commands. You can use the Clock Networks and Clock Interaction report windows to view clock topology and relationships. You can also use the Slack Histogram window to see an overall view of the design timing performance. For more information, see this link in the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906) [Ref 13].

In addition, the Vivado IDE has many timing analysis options available through the Tcl Console and SDC constraint options. Many 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) [Ref 19], or type `help report_*`.

Running DRC, Power, and Utilization Analysis

The Vivado IDE provides a graphical way to configure and view power, utilization, and DRC analysis results. You can experiment with power parameters and quickly estimate power at any stage of the design. You can also analyze various types of device resource utilization statistics. A comprehensive set of DRCs is available that you can configure 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) [Ref 13].

TIP: *Many 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) [Ref 19].*

Device Programming, Hardware Verification, and Debugging

In the Vivado IDE, the Vivado logic analyzer includes many features to enable verification and debugging of the design. You can configure and implement IP debug cores, such as the Integrated Logic Analyzer (ILA) and Debug Hub core, in either an RTL or synthesized netlist. Opening the synthesized or implemented 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. In addition, you can create programming bitstream files for any completed implementation run. Bitstream file generation options are configurable. Launch the Vivado device programmer to configure and program the part. 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 this link in the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 15].

Using Project Mode Tcl Commands

Table 3-1 shows the basic Project Mode Tcl commands that control project creation, implementation, and reporting.

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

Table 3-1: **Basic Project Mode Tcl Commands**

Command	Description
create_project	Creates the Vivado Design Suite project. Arguments include project name and location, design top module name, and 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), and System Generator modules (.mdl). Individual files or entire directory trees can be scanned for legal sources and automatically added to the project. ***Note:*** The .xco file is no longer supported in UltraScale.
set_property	Used for multiple purposes in the Vivado Design Suite. For projects, it can be used to define VHDL libraries for sources, simulation-only sources, target constraints files, tool settings, and so forth.
import_files	Imports the specified files into the current file set, effectively adding them into the project infrastructure. It is also used to assign XDC files into constraints sets.
launch_runs launch_runs -to_step	Starts either synthesis or implementation and bitstream generation. This command encompasses the individual implementation commands as well as the standard reports generated after the run completes. It is used to launch all of the steps of the synthesis or implementation process in a single command, and to track the tools progress through that process. The -to_step option is used to launch the implementation process, including bitstream generation, in incremental steps.
wait_on_run	Ensures the run is complete before processing the next commands in a Tcl script.
open_run	Opens either the synthesized design or implemented design for reporting and analysis. A design must be opened before information can be queried using Tcl for reports, analysis, and so forth.
close_design	Closes the design in memory.
start_gui stop_gui	Invokes or closes the Vivado IDE with the current design in memory.

Note: This document is not a complete reference for the available Tcl commands. Instead, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 19].

Project Mode Tcl Script Examples

The following examples show a Tcl script for an RTL project and a netlist project. The first example script, `run_bft_kintex7_project.tcl`, is available in the Vivado Design Suite installation at: `<install_dir>/Vivado/2015.1/examples/Vivado_Tutorial`

The scripts can be source from the Vivado Tcl shell, or the Tcl Console inside of the Vivado IDE.

RTL Project Tcl Script

```
# run_bft_kintex7_project.tcl
# BFT sample design
#
# NOTE:  -Typical usage would be "vivado -mode tcl -source
run_bft_kintex7_project.tcl"
#       -To use -mode batch comment out the "start_gui" and "open_run impl_1" to save
time
#
create_project project_bft ./Tutorial_Created_Data/project_bft -part
xc7k70tfbg484-2
add_files {./Sources/hdl/FifoBuffer.v ./Sources/hdl/async_fifo.v
./Sources/hdl/bft.vhdl}
add_files -fileset sim_1 ./Sources/hdl/bft_tb.v
add_files ./Sources/hdl/bftLib
set_property library bftLib [get_files  {./Sources/hdl/bftLib/round_4.vhdl \
./Sources/hdl/bftLib/round_3.vhdl ./Sources/hdl/bftLib/round_2.vhdl
./Sources/hdl/bftLib/round_1.vhdl \
./Sources/hdl/bftLib/core_transform.vhdl ./Sources/hdl/bftLib/bft_package.vhdl}]
import_files -force
import_files -fileset constrs_1 -force -norecurse ./Sources/bft_full_kintex7.xdc
# Mimic GUI behavior of automatically setting top and file compile order
update_compile_order -fileset sources_1
update_compile_order -fileset sim_1
# Launch Synthesis
launch_runs synth_1
wait_on_run synth_1
open_run synth_1 -name netlist_1
# Generate a timing and power reports and write to disk
report_timing_summary -delay_type max -report_unconstrained -check_timing_verbose \
-max_paths 10 -input_pins -file
./Tutorial_Created_Data/project_bft_batch/syn_timing.rpt
report_power -file ./Tutorial_Created_Data/project_bft_batch/syn_power.rpt
# Launch Implementation
launch_runs impl_1 -to_step write_bitstream
wait_on_run impl_1
# Generate a timing and power reports and write to disk
# comment out the open_run for batch mode
open_run impl_1
report_timing_summary -delay_type min_max -report_unconstrained
-check_timing_verbose \
-max_paths 10 -input_pins -file
./Tutorial_Created_Data/project_bft_batch/imp_timing.rpt
report_power -file ./Tutorial_Created_Data/project_bft_batch/imp_power.rpt
# comment out the for batch mode
start_gui
```

Netlist Project Tcl Script

```
# Kintex-7 Netlist Example Design
#
# STEP#1: Create Netlist Project, add EDIF sources, and add constraints
#
create_project -force project_K7_netlist ./Tutorial_Created_Data/project_K7_netlist
-part xc7k70tfbg676-2
# Property required to define Netlist project
set_property design_mode GateLvl [current_fileset]
add_files {./Sources/netlist/top.edif}
import_files -force
import_files -fileset constrs_1 -force ./Sources/top_full.xdc

#
# STEP#2: Configure and Implementation, write bitstream, and generate reports
#
launch_runs impl_1
wait_on_run impl_1
launch_runs impl_1 -to_step write_bitstream
wait_on_run impl_1
open_run impl_1
report_timing_summary -delay_type min_max -report_unconstrained
-check_timing_verbose \
-max_paths 10 -input_pins -file
./Tutorial_Created_Data/project_K7_netlist/imp_timing.rpt
report_power -file ./Tutorial_Created_Data/project_K7_netlist/imp_power.rpt
#
# STEP#3: Start IDE for design analysis
#
start_gui
```

Using Non-Project Mode

Overview

In Non-Project Mode, you use Tcl commands to compile a design through the entire flow. In this flow, an in-memory project is created, to manage various properties of the design, but no project file is written to disk. Tcl commands provide the flexibility and power to set up and run your designs and perform analysis and debugging. Tcl commands can be run in batch mode, from the Vivado® Design Suite Tcl shell, or through the Vivado IDE Tcl Console. Non-Project 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 DRCs, and write design checkpoints at any stage of the implementation process.

Unlike Project Mode, Non-Project 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 rerun the design manually. Default reports and intermediate files are not created automatically in this mode. However, you can create a wide variety of reports and design checkpoints as needed using Tcl commands. In addition, you can still access the GUI-based design analysis and constraints assignment features of the Vivado IDE. You can open either the current design in memory or any saved design checkpoint in the Vivado IDE.

When you launch the Vivado IDE in Non-Project Mode, the Vivado IDE does not include Project Mode features such as the Flow Navigator, Project Summary, or Vivado IP catalog. In Non-Project Mode, you cannot access or modify synthesis or implementation runs in the Vivado IDE. However, if the design source files reside in their original locations, you can cross probe to design objects in the different windows of the Vivado IDE. For example, you can select design objects and then use the **Go To Instantiation**, **Go To Definition**, or **Go To Source** commands to open the associated RTL source file and highlight the appropriate line.

Non-Project Mode Advantages

Non-Project Mode enables you to have full control over each design flow step. You can take advantage of a compile-style design flow.

In this mode, you manage your design manually, including:

- Manage HDL Source files, constraints, and IP
- Manage dependencies
- Generate and store synthesis and implementation results

The Vivado Design Suite includes an entire suite of Vivado Tcl commands to create, configure, implement, analyze, and manage designs as well as IP. In Non-Project Mode, you can use Tcl commands to do the following:

- Compile a design through the entire flow
- Analyze the design and generate reports

Reading Design Sources

When using Non-Project Mode, the various design sources are read into the in-memory design for processing by the implementation tools. Each type of Vivado Design Suite source file has a `read_*` Tcl command to read the files, such as `read_verilog`, `read_vhdl`, `read_ip`, `read_edif`, or `read_xdc`. Sources must be read each time the Tcl script or interactive flow is started.

TIP: *Because there is no project structure to add the files or import the files into, you should not use the add_files or import_files Tcl commands to add files to a non-project based design.*

Managing Source Files

In Non-Project Mode, you manage source files manually by reading the files into the in-memory design in a specific order. This gives you full control over how to manage the files and where files are located. Sources can be read from any network accessible location. Sources with read-only permissions are processed accordingly.

Working with a Revision Control System

Many design teams use source management systems to store various design configurations and revisions. There are multiple commercially available systems, such as Revision Control System (RCS), Concurrent Versions System (CVS), Subversion (SVN), ClearCase, Perforce, Git, BitKeeper, and many others. The Vivado tools can interact with all such systems. The Vivado

Design Suite uses and produces files throughout the design flow that you may want to manage under revision control.

Working with revision control software is simple when using the Non-Project mode. The designer checks out the needed source files into a local directory structure. The sources are then instantiated into a top-level design to create the design. New source files might also need to be created and read into the design using various read_* Tcl commands. The design files are passed to the Vivado synthesis and implementation tools. However, the source files remain in their original locations. The checked-out sources can be modified interactively, or with Tcl commands during the design session using appropriate code editors. Source files are then checked back into the source control system as needed. Design results, such as design checkpoints, analysis reports, and bitstream files, can also be checked in for revision management. For more information on working with revision control software, see this link in the *UltraFast™ Design Methodology Guide for the Vivado Design Suite* (UG949) [Ref 22].

VIDEO: *For information on best practices when using revision control systems with the Vivado tools, see the* Vivado Design Suite QuickTake Video: Version Control Overview.

Using Third-Party Synthesized Netlists

The Vivado Design Suite supports implementation of synthesized netlists, such as when using a third-party synthesis tool. 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.

Working with IP

In Non-Project mode, output products must be generated for the IP prior to launching synthesis. You can configure IP to use RTL sources and constraints or use a synthesized design checkpoint as the source in the top-level design. The default behavior is to generate an out-of-context design checkpoint for each IP.

In Non-Project Mode, you can implement the IP in your design using any of the following methods:

- IP generated using the Vivado IP catalog (.xci format)

  If the out-of-context design checkpoint file exists in the IP directory, it is used for implementation and a black box is inserted for synthesis. If a design checkpoint file does not exist in the IP directory, the RTL and constraints sources are used for global synthesis and implementation.

- Synthesized netlist for the IP (.dcp format)

Using a synthesized netlist enables you to structurally verify the IP standalone to provide a stable known netlist. Adding a Vivado IP file also uses the design checkpoint file if it exists in the IP directory.

- Set of Tcl commands to configure and generate the IP

  Using Tcl ensures that the IP is configured, generated, and synthesized with each run.

For more information, see this link in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Running Logic Simulation

The Vivado simulator, integrated with 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 is a fully integrated mixed-mode simulator with analog waveform display capabilities. Using the Vivado simulator, you can perform behavioral and structural simulation of designs and full timing simulation of implemented designs.

You can also use third-party simulators to write 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 this link in the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 14].

Running Logic Synthesis and Implementation

In Non-Project Mode, each implementation step is launched with a configurable Tcl command, and the design is compiled in memory. The implementation steps must be run in a specific order, as shown in the Non-Project Mode Tcl Script Example. Optionally, you can run steps such as `power_opt_design` or `phys_opt_design` as needed. Instead of run strategies, which are only supported in Project Mode, you can use various commands to control the tool behavior. For more information, see the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

It is important to save design checkpoints after critical design steps for design analysis and constraints definition. With the exception of generating a bitstream, design checkpoints are not intended to be used as starting points to continue the design process. They are merely snapshots of the design for analysis and constraint definition.

TIP: *After each design step, you can launch the Vivado IDE to enable interactive graphical design analysis and constraints definition on the active design, as described in Performing Design Analysis Using the Vivado IDE.*

Generating Reports

With the exception of the `vivado.log` and `vivado.jou` reports, reports must be generated manually with a Tcl command. You can generate various reports at any point in the design process. For more information, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 19] or *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 3].

Using Design Checkpoints

Design checkpoints enable you to take a snapshot of your design in its current state. The current netlist, constraints, and implementation results are stored in the design checkpoint. Using design checkpoints, you can:

- Restore your design if needed
- Perform design analysis
- Define constraints
- Proceed with the design flow

You can write design checkpoints at different points in the flow. It is important to write design checkpoints after critical design steps for design analysis and constraints definition. You can read design checkpoints to restore the design, which might be helpful for debugging issues. The design checkpoint represents a full save of the design in its current implementation state. You can run the design through the remainder of the flow using Tcl commands. However, you cannot add new sources to the design.

Note: You can also use the `write_checkpoint <file_name>.dcp` and `read_checkpoint <file_name>.dcp` Tcl commands to write and read design checkpoints. To view a checkpoint in the Vivado IDE, use the `open_checkpoint <file_name>.dcp` Tcl command. For more information, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 19].

IMPORTANT: *When using IP in Project Mode or Non-Project Mode, always use the XCI file not the DCP file. This ensures that IP output products are used consistently during all stages of the design flow. If the IP was synthesized out-of-context and already has an associated DCP file, the DCP file is automatically used and the IP is not re-synthesized. For more information, see this link in the Vivado Design Suite User Guide: Designing with IP (UG896) [Ref 1].*

Performing Design Analysis Using the Vivado IDE

In Non-Project Mode, you can launch the Vivado IDE after any design step to enable interactive graphical design analysis and constraints definition on the active design.

Opening the Vivado IDE for the Active Design

When working in Non-Project Mode, use the following commands to open and close 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 to the Vivado Design Suite Tcl shell.

CAUTION! *If you exit the Vivado Design Suite from the GUI, the Vivado Design Suite Tcl shell closes and does not save the design in memory. To return to the Vivado Design Suite Tcl shell with the active design intact, use the `stop_gui` Tcl command rather than the `exit` command.*

After each stage of the design process, you can open the Vivado IDE to analyze and operate on the current design in memory (Figure 4-1, page 66). In Non-Project Mode, some of the project features are not available in the Vivado IDE, such as the Flow Navigator, Project Summary, source file access and management, and runs. However, many of the analysis and constraint modification features are available in the **Tools** menu.

IMPORTANT: *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.*

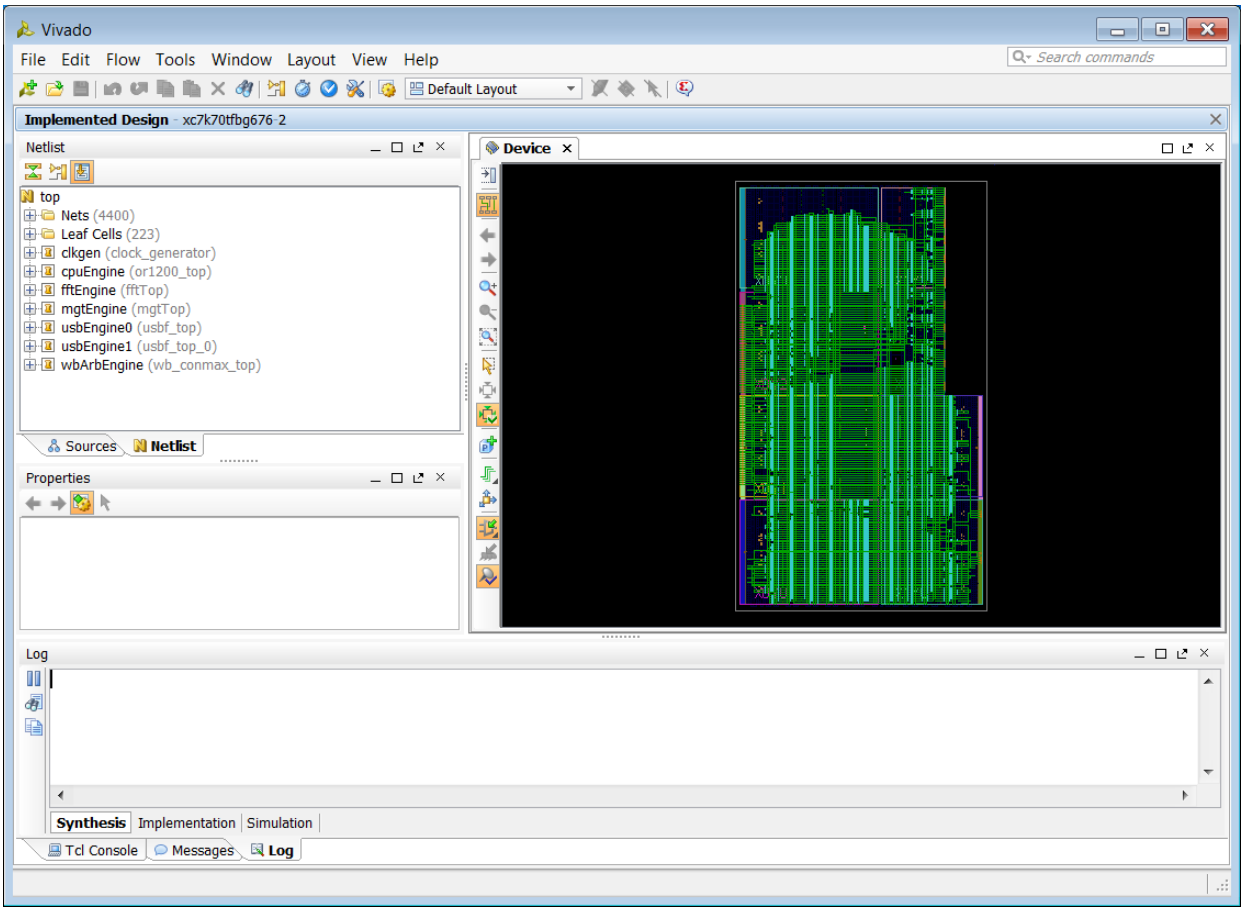

Figure 4-1: **Opening Vivado IDE with the Active Design**

Saving Design Changes to the Active Design

Because you are actively editing the design in memory, changes are automatically passed to downstream tools for the remainder of the Vivado IDE Tcl session. This enables you 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 this command to write a new constraints file or override your original file.

Note: When you export constraints, the `write_xdc` Tcl command is run. For more information, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 19].

Opening Design Checkpoints in the Vivado IDE

You can use the Vivado IDE to analyze designs saved as design checkpoints (Figure 4-2). 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 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. The Vivado IDE view banner displays the open design checkpoint name.

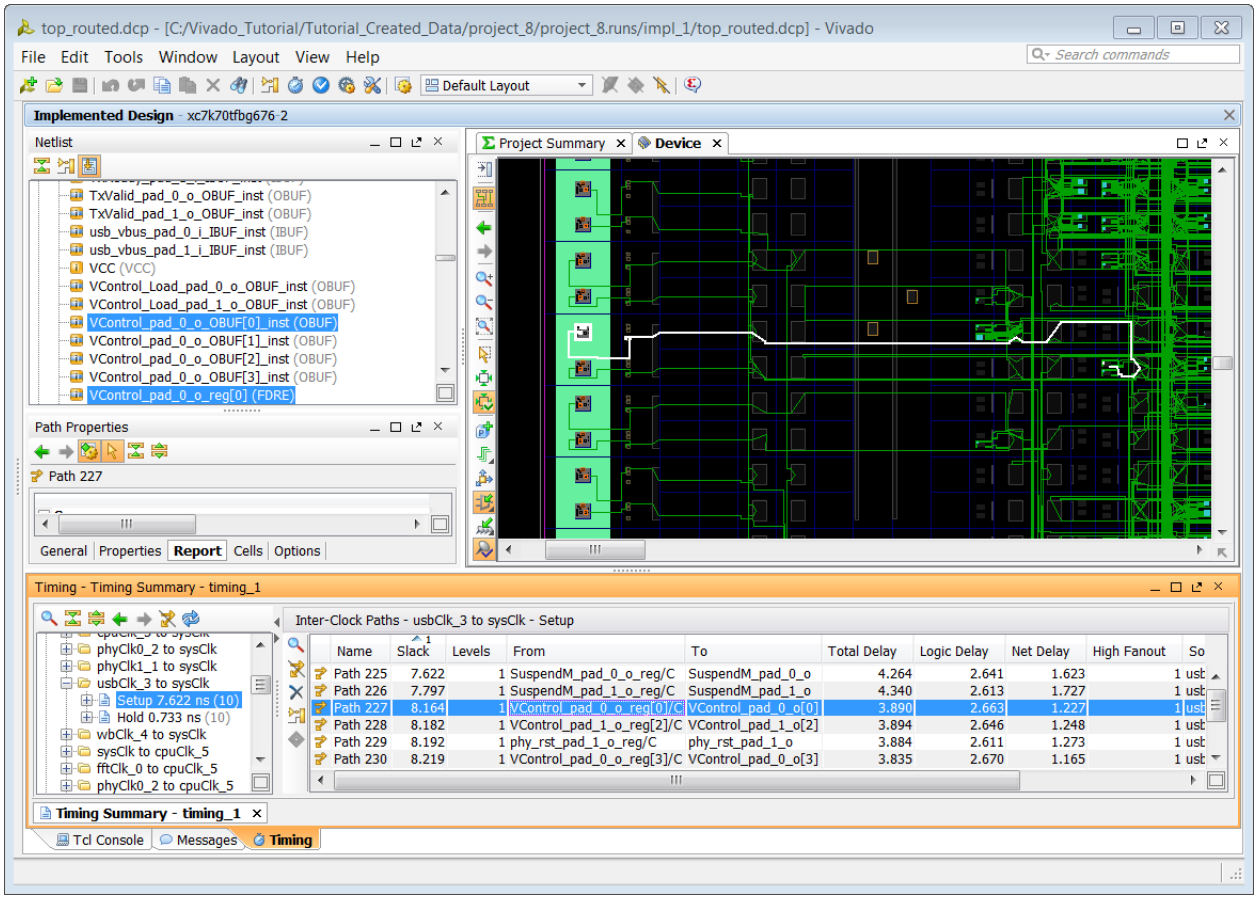

Figure 4-2: **Opening Design Checkpoints in the Vivado IDE**

Saving Design Changes to Design Checkpoints

You can open, analyze, and save design checkpoints. 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 design checkpoint to a new design checkpoint.

Using Non-Project Mode Tcl Commands

Table 4-1 shows the basic Non-Project Mode Tcl commands. When using Non-Project Mode, the design is compiled using `read_verilog`, `read_vhdl`, `read_edif`, `read_ip`, `read_bd`, and `read_xdc` type commands. The sources are ordered for compilation and passed to synthesis. For information on using the Vivado Design Suite Tcl shell or using batch Tcl scripts, see Working with Tcl in Chapter 2.

Note: This document is not a complete reference for the available Tcl commands. Instead, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 19] and *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894) [Ref 18].

Table 4-1: **Basic Non-Project Mode Tcl Commands**

Command	Description
read_edif	Imports an EDIF or NGC netlist file into the Design Source fileset of the current project.
read_verilog	Reads the Verilog (.v) and System Verilog (.sv) source files for the Non-Project Mode session.
read_vhdl	Reads the VHDL (.vhd or .vhdl) source files for the Non-Project Mode session.
read_ip	Reads existing IP (.xci or .xco) project files for the Non-Project Mode session. For Vivado IP (.xci), the design checkpoint (.dcp) synthesized netlist is used to implement the IP if the netlist is in the IP directory. If not, the IP RTL sources are used for synthesis with the rest of the top-level design. The .ngc netlist is used from the .xco IP project. ***Note:*** The .xco file is no longer supported in UltraScale.
read_checkpoint	Loads a design checkpoint into the in-memory design.
read_xdc	Reads the .sdc or .xdc format constraints source files for the Non-Project Mode session.
read_bd	Reads existing IP Integrator block designs (.bd) for the Non-Project session.
set_param set_property	Used for multiple purposes. For example, it can be used to define design configuration, tool settings, and so forth.
link_design	Compiles the design for synthesis if netlist 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 high-level design optimization.
power_opt_design	Performs intelligent clock gating to reduce overall system power. This is an optional step.
place_design	Places the design.
phys_opt_design	Performs physical logic optimization to improve timing or routability. This is an optional step.
route_design	Routes the design.
report_*	Runs a variety of standard reports, which can be run at different stages of the design process.

Table 4-1: **Basic Non-Project Mode Tcl Commands *(Cont'd)***

Command	Description
write_bitstream	Generates a bitstream file and runs DRCs.
write_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 implementation results.
start_gui stop_gui	Invokes or closes the Vivado IDE with the current design in memory.

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.

The script, `run_bft_kintex7_batch.tcl`, is available in the Vivado Design Suite installation at: `<install_dir>/Vivado/2015.1/examples/Vivado_Tutorial`

You can source this script from the Vivado Tcl shell, or the Tcl Console inside of the Vivado IDE.

```
# run_bft_kintex7_batch.tcl
# bft sample design
# A Vivado script that demonstrates a very simple RTL-to-bitstream non-project batch
flow
#
# NOTE:  typical usage would be "vivado -mode tcl -source run_bft_kintex7_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_kintex7.xdc
#
# STEP#2: run synthesis, report utilization and timing estimates, write checkpoint
design
#
synth_design -top bft -part xc7k70tfbg484-2
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 optimzation, report utilization and timing
estimates, write checkpoint design
#
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
$outputDir/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
```

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

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

1. *Vivado Design Suite User Guide: Designing with IP* (UG896)
2. *Vivado Design Suite User Guide: Synthesis* (UG901)
3. *Vivado Design Suite User Guide: Implementation* (UG904)
4. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898)
6. *Vivado Design Suite Tutorial: Embedded Processor Hardware Design* (UG940)
7. *Vivado Design Suite User Guide: Model-Based DSP Design Using System Generator* (UG897)
8. *Vivado Design Suite User Guide: High-Level Synthesis* (UG902)
9. *Vivado Design Suite Tutorial: High-Level Synthesis* (UG871)
10. *Vivado Design Suite User Guide: Hierarchical Design* (UG905)
11. *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899)
12. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)

13. *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906)
14. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
15. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
16. *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909)
17. *Vivado Design Suite Tutorial: Partial Reconfiguration* (UG947)
18. *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894)
19. *Vivado Design Suite Tcl Command Reference Guide* (UG835)
20. *Vivado Design Suite Tutorial: Design Flows Overview* (UG888)
21. *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893)
22. *UltraFast™ Design Methodology Guide for the Vivado Design Suite* (UG949)
23. *Vivado Design Suite User Guide: Using Constraints* (UG903)
24. *ISE to Vivado Design Suite Migration Guide* (UG911)
25. *Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118)
26. *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119)
27. Vivado Design Suite QuickTake Video: Designing with Vivado IP Integrator
28. Vivado Design Suite QuickTake Video: Targeting Zynq Devices Using Vivado IP Integrator
29. Vivado Design Suite QuickTake Video: Partial Reconfiguration in Vivado Design Suite
30. Vivado Design Suite QuickTake Video: Version Control Overview
31. Vivado Design Suite QuickTake Video Tutorials
32. Vivado Design Suite Documentation

Training Resources

Xilinx provides a variety of training courses and QuickTake videos to help you learn more about the concepts presented in this document. Use these links to explore related training resources:

1. Vivado Design Suite Hands-on Introductory Workshop
2. Vivado Design Suite Tool Flow
3. Essentials of FPGA Design

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

© Copyright 2012–2015 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.