***System-Level Design Entry***

**Chapter 1: Introduction**

System-level design entry consists of setting up your design, including creating a project (if applicable), creating and adding source files, adding block design and IP cores, elaborating the RTL design, and inserting and configuring debug information. You can enter your design using the graphical user interface (GUI), known as the Vivado Integrated Design Environment (IDE), or using Tcl commands and scripts.

**Chapter 2: Working with Projects**

When working in Project Mode, you can enter your design using various project types. This

chapter describes each project type and explains how to create and manage projects. It also

covers the Project Summary, Project Settings, and how to create a project using a Tcl script.

**2.1 Project Types**

• RTL Projects

You can add RTL source files, IP from the AMD IP catalog, block designs created in the Vivado IP integrator, digital signal processing (DSP) sources, and EDIF netlists for hierarchical modules. IP can include XCI or XCIX files generated by the Vivado tools, legacy XCO files generated by the CORE Generator tool, and precompiled EDIF or NGC-format netlists.

• Post-Synthesis Projects

You can create projects using synthesized netlists created using Vivado synthesis, XST, or any

supported third-party synthesis tool.

RECOMMENDED: Always reference the Vivado IP using the XCI or XCIX file. AMD does not recommend reading just the IP DCP file. While the DCP does contain constraints, it does not provide other output products that an IP could deliver and that could be needed, such as ELF, COE, and Tcl scripts.

• I/O Planning Projects

You can perform clock resource and I/O planning early in the design cycle by creating an empty I/O planning project. You can define I/O ports within the Vivado IDE or import them with either comma separated value (CSV) or XDC input files. You can also create empty I/O planning projects to explore the logic resources available on the different device architectures.

• Imported Projects

You can import RTL project data from Synopsys Synplify into the Vivado tools. The project source files and compilation order are imported, but implementation results and settings are not.

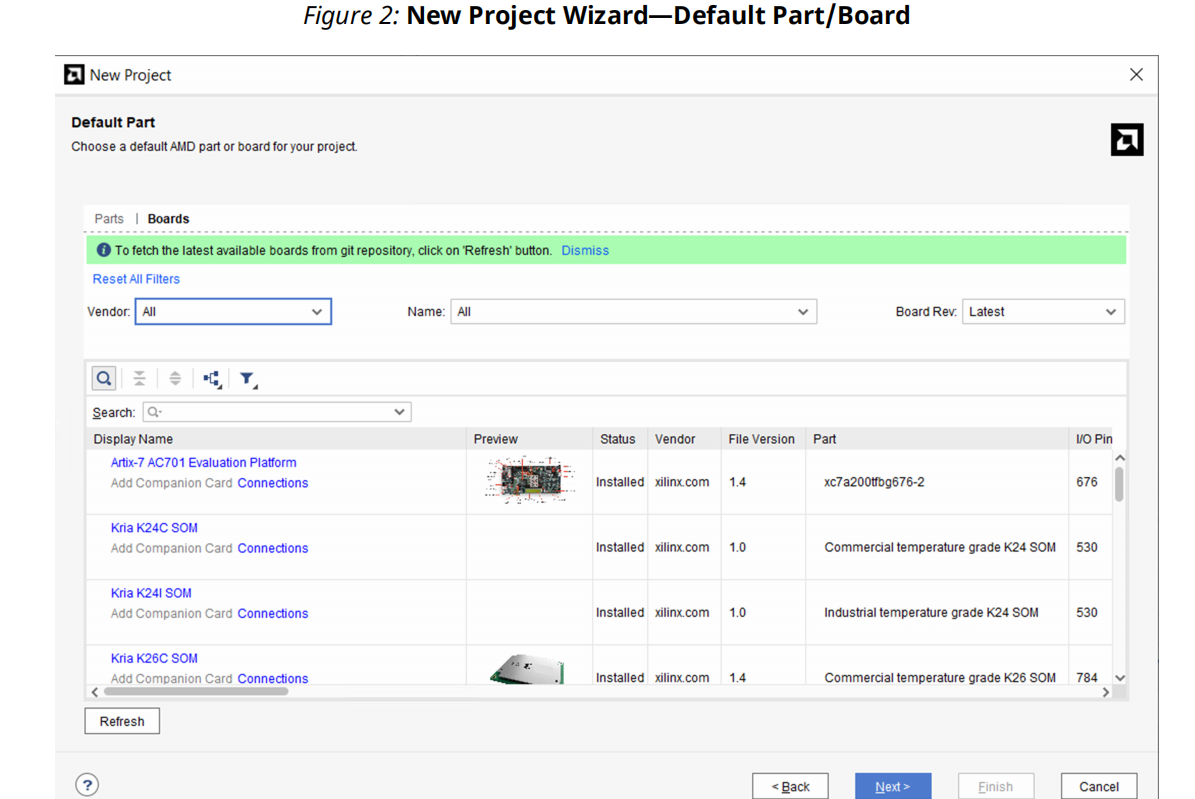
**2.2 Creating a Project**

VHDL 的一个强大功能是用库来组织 RTL 的不同部分。 通过使用库，不同的设计人员可以做这个工程中自己负责的那部分工作，而不必担心会在命名方面与其他设计师发生冲突。在VHDL中，名为“work”的库在 VHDL 中的用法比较特殊，实际上它指的是“当前库”，如果没有指定当前库的库名，编译器会创建“work”库。不同的项目若都使用“work”库名会产生混淆，因此不同项目需要指定独立的当前库名而避免使用“work”库名。

默认情况下，将 VHDL 文件输入 Vivado 工程时，该工具会将这些文件放入一个名为“xil\_defaultlib”的库中。xil\_defaultlib是当前库的库名，对应于VHDL语句中的“work”。使用“xil\_defaultlib”作为当前库的库名是为了区别其他（引入）项目时，可能会使用“work”作为当前库名，这样以示区别，避免混淆。因此，开发独立库时应使用对应的库名，不要使用“work”库名。

**2.3 Using the Vivado Design Suite Platform Board Flow**

The Vivado Design Suite lets you create projects using AMD target design platform boards (TDP), or user-specified boards that have been added to a board repository. When you select a specific board, the Vivado design tools show information about the board, and enable additional designer assistance as part of IP customization, and for IP integrator designs.



**2.3.1 Adding User-Boards to a Repository**

set\_param board.repoPaths [list "<path1>" “<path2>” “...”]

These paths can also be added in the GUI using **Tools → Settings → Vivado Store → Board**

**Repository**

For more information about the Vivado\_init.tcl file refer to the Scripting in Tcl section in

the Vivado Design Suite Tcl Command Reference Guide (UG835).

**TIP:** The Vivado Design Suite board repository at <Vivado\_install\_location>/data/boards is

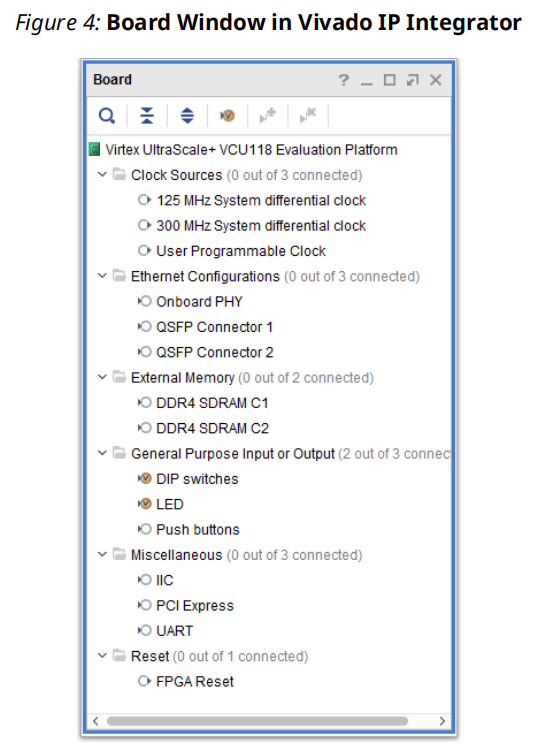
always read regardless of the value of this parameter.

**2.3.2 Using the IP Catalog with the Platform Board Flow**

A Board tab is available in the Customize IP dialog box when you are working with IP from the AMD IP catalog that supports the platform board flow.

**2.3.3 Using the Vivado IP Integrator with the Platform Board Flow**

Optionally, you can use the Vivado IP integrator to add IP to your block design. If you selected a board for the project, the Board window is available in IP integrator in the toolbar by selecting **Window → Boards**.



**2.4 Managing Projects**

**2.4.1 Opening a Project**

Tcl Command for Opening a Project

• Tcl Command: open\_project

• Tcl Command Example: open\_project c:/projects/project\_1.xpr

**2.4.2 Opening Multiple Projects**

**2.4.3 Saving a Project**

Projects are automatically saved. However, changes to the design constraints are not automatically saved as part of the project. You must use the Save Constraints command, or Save Constraints As..., to write constraint changes to disk.

Tcl Command for Saving a Project

• Tcl Command: save\_project\_as

• Tcl Command Example: save\_project\_as new\_project c:/projects/project\_1.xpr

**2.4.3 Closing a Project**

Tcl Command for Closing a Project

Tcl command: close\_project

**2.4.3 Archiving a Project**

1. Select File → Project → Archive.

2. In the Archive Project dialog box, set the following options, and click OK.

• Archive name: Specifies the name of the project archive.

• Temporary location: (Windows only) Specifies a temporary directory to copy files to when

creating the project archive.

• Include configuration settings: Includes the Vivado\_init.tcl file, which contains Tcl

initialization commands that are helpful in debugging your design.

• Include run results: Includes the settings and results of the runs performed on the project.

Including the results of synthesis and implementation runs can significantly increase the

size of the project archive.

• Include local IP cache results: Includes the information included in the local IP cache (if

any) to speed up IP generation times using version of the tools.

The Vivado IDE creates a project archive in ZIP file format that contains the required source

files, include files, and run files (if specified) as well as an archive.log file of the archival

process. You can review the creation of the archive in the archive.log file.

You can also use the write\_project\_tcl command to generate a Tcl script that will recreate the current project. The script will keep the project settings and sources, but might not retain output products or design state.

Tcl Command for Archiving a Project

• Tcl command: archive\_project

• Tcl command example: archive\_project -exclude\_run\_results proj3.zip

Note: To avoid the 256 character limit on Windows, use the -temp\_dir option to specify a temporary directory to copy files to when creating the project archive.

**2.4.4 Working with Source Control Systems**

**2.5 Using the Project Summary**

To open the Project Summary, do either of the following:

• Select Windows → Project Summary.

• Select the Project Summary toolbar button ∑

Note: The Overview tab in the Project Summary appears by default.

**2.6 Configuring Project Settings**

To open the Settings dialog box, use any of the following methods:

• Select Tools → Settings.

• Click the Settings toolbar button ✿

• In the Flow Navigator, click Settings in the Project Manager section, or right click one of the

following:

SIMULATION: Opens Simulation Settings

RTL ANALYSIS: Opens Elaboration Settings

SYNTHESIS: Opens Synthesis Settings

IMPLEMENTATION: Opens Implementation Settings

PROGRAM AND DEBUG: Opens Bitstream Settings

• In the Project Summary, click the Edit link next to the Settings header, or click the strategy or flow in either the Synthesis or Implementation section.

**2.7 Creating a Project Using a Tcl Script**

You can use the **write\_project\_tcl** command to generate a Tcl script that will re-create the

current project. The script keeps the project settings and sources, but might not retain output products or design state.

As an alternative to creating a project in the Vivado IDE, you can create a project using a Tcl

script. Most actions run in the Vivado IDE result in a Tcl command being executed. The Tcl

commands appear in the Vivado IDE Tcl console and are also captured in the vivado.jou and

vivado.log files. The vivado.jou file contains only the commands, and the vivado.log

file contains both commands and any returned messages. You can use these files to develop

scripts for use with Project Mode. Refer to Output Files in Appendix A of the Vivado Design Suite User Guide: Using the Vivado IDE (UG893) for information on where the vivado.jou and log files are written.

**Chapter 3: Working with Source Files**

1. RTL design sources
2. intellectual property (IP) sources added from the AMD IP catalog
3. IP subsystems, also known as block designs, created by the IP integrator
4. C-based Vitis HLS sources，support C、C++、SystemC Language
5. model-based Vitis Model Composer graphical design sources （UG1262）
6. digital signal processing (DSP) sources added from the System Generator
7. simulation source files
8. constraints files that specify timing requirements and physical constraints

**3.1、 Creating and Adding Design Sources**

The Sources window provides different ways of viewing the source files associated with a project, including the following views:

**• Hierarchy**

**• IP Sources**: Displays all of the files defined by an IP core, a block design added from the Vivado IP integrator, or a DSP module added from System Generator.

**• Libraries**: Displays design sources sorted into their various libraries.

**• Compile Order**: Displays source files in the order in that they will be compiled, first to last, and shows the processing order for constraints.

**3.1.1、Creating New Source Files**

• Library: Specifies the RTL library for a file or directory. You can select a library name, or

specify a new library name by typing in the Library text field.

Note: This option applies to VHDL files only. By default, HDL sources are added to the

xil\_defaultlib library. You can create or reference additional user VHDL libraries as needed.

For Verilog and SystemVerilog files, leave the library set to xil\_defaultlib.

**3.1.2、Adding Design Sources**

**Note**: In the Add Source Files dialog box, each file or directory is represented by an icon indicating it as a file or folder. A small red square indicates it is read only.

**3.1.3、Specifying the Top Module and Reordering Source Files**

Select **Refresh Hierarchy** from the right-click menu in the Sources window to reorder files based on changes to the source files.

You can specify the manual compile order mode by selecting

**Hierarchy Update → Automatic Update, Manual Compile Order** or

**Hierarchy Update → No Update, Manual Compile Order** in the right-click menu of the Sources window.

To manually order source files, select a file and drag it up or down in the file list order in the Compile Order view of Sources window. Alternatively, after selecting the file, use **Move Up, Move Down, Move to Top or Move to Bottom** from the Sources window right-click menu.

To see a full list of the compile or evaluation order for all sources, use the

**report\_compile\_order** command in the Tcl Console. RTL compile order is listed for synthesis and simulation. Constraints evaluation order is listed for synthesis and implementation.

**3.1.4、Enabling or Disabling Source Files**

**3.1.5、Using Remote Sources or Copying Sources into Project**

**3.1.6、Updating Local Source Files**

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

• In the Sources window, select the file, and select Replace File from the right-click menu.

• In the Sources window, select Add Sources from the right-click menu to add the newly

updated source files to the project.

**3.2、 Working with IP Sources**

In the Vivado IDE, you can add and manage the following types of IP cores in an RTL project:

• Vivado Design Suite AMD Core Instance files (XCI) 原厂IP 的XCI文件

The XCI file stores the configuration and constraint options for an IP core that you specify, or customize, when you add the IP to a design.

**IMPORTANT!** When using IP in either Project Mode or Non-Project Mode, always add the XCI file to the design; not a synthesized DCP file. The use of the XCI file ensures that the output products of the IP core that are needed by the tool are generated and used consistently throughout the design flow.

• Vivado Design Suite Core Container files (XCIX) 原厂IP 的XCIX文件

The Core Container feature simplifies working with revision control systems by providing a

single file representation of an IP. The IP configuration and all generated output files are

contained in one compressed binary file with an extension of XCIX. This extension is similar to the XCI file used for the IP customization file and works in a similar way.

• User IP packaged with Vivado IP packager (XCI) 用户IP 的XCIX文件及库

The Vivado Design Suite User Guide: Creating and Packaging Custom IP (UG1118) describes how to package user-defined IP for use in the AMD IP catalog. User IP repositories can be added to the catalog using the **IP\_REPO\_PATHS** property, which defines the path for one or more directories containing third-party or user-defined IP.

• CORE Generator IP cores (XCO) 早期版本的IP文件

Legacy IP from the CORE Generator tool are supported by the Vivado Design Suite.

• Third-party IP Netlists

In some cases, third-party providers offer IP as synthesized NGC or EDIF netlists. You can load these files into a project or design as hierarchical design sources using the Add Sources

command. For information, see Creating and Adding Design Sources.

**3.2.1、Adding IP from the IP Catalog**

TIP: By default, the IP catalog only displays IP cores that are compatible with, or supported by the target part (or board) for the current project. You can change the default setting to show all IP in the catalog by deselecting the Hide toolbar button in the Vivado IP catalog.

**3.2.2、Adding Existing IP Files**

you can directly add XCI or XCIX files into your project or design. To add existing XCI or XCIX files directly into your design or project, select File → Add Sources. See Adding Design Sources for detailed information.

If the XCI or XCIX file included any needed support files, referred to collectively as output

products, those files are added when the design source is added to the design. If the XCI or XCIX file does not include these associated files, you must generate the output products required to support the IP in your design, such as the instantiation template, XDC constraints, and simulation sources. See Generating Output Products for IP Cores for more information.

You can run Reports → Report IP Status and review the state of the newly added IP. The Tcl command for reporting IP status is report\_ip\_status.

**3.2.3、Generating Output Products for IP Cores**

The IP core includes, or requires, specific files to support the IP in the overall design flow.

When an IP is customized from the IP catalog, the Generate Output Products dialog box is opened. However, you can also open this dialog box at any time by right-clicking the IP in the

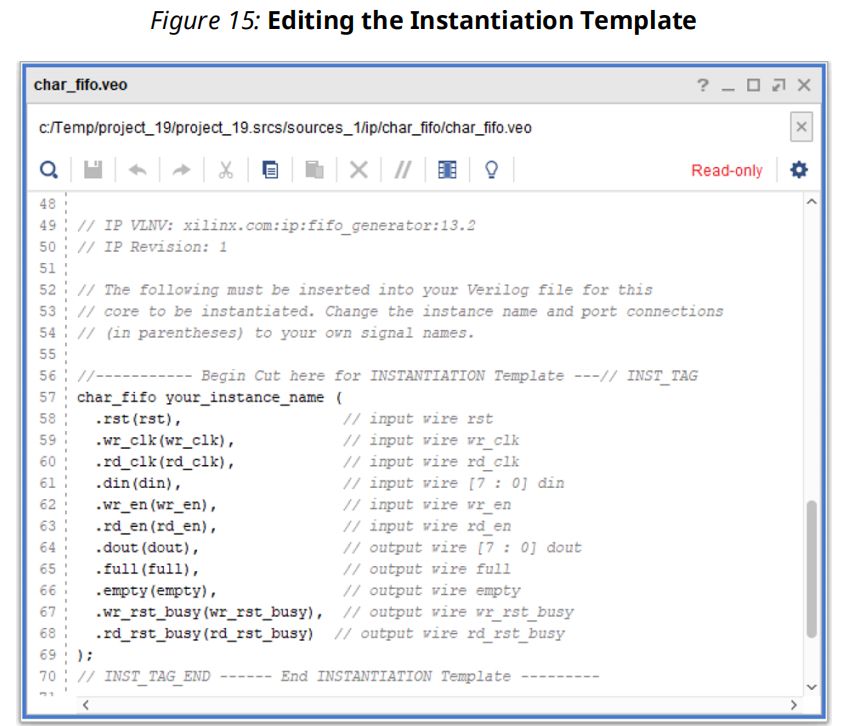
Sources window and selecting the Generate Output Products command.

**3.2.4、Instantiating IP Into the Design**

An instantiation template is created when you customize and IP and add it to your design or

project, regardless of whether you generated output products. The instantiation template

provides a Verilog or VHDL instance declaration (.veo or .vho) that you can copy and paste into your RTL design hierarchy.



**3.3、 Working with IP Integrator Sources**

In the Vivado Design Suite, you can add and manage IP subsystem block designs (.bd) in an RTL project or design.

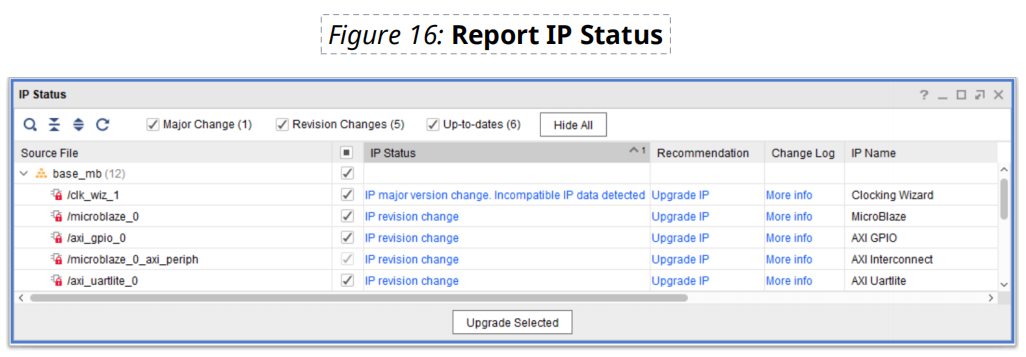
* Using the Vivado IP integrator, you can create an IP subsystem block design.
* The IP integrator enables you to create complex system designs by instantiating and interconnecting multiple IP cores from the Vivado IP catalog.

**3.3.1、Creating a New Block Design**

**3.3.2、Adding existing Block Design sources**

When an existng block design is added to the current project, the project might target a diferent AMD part. This will result in all of the IP used in the block design being locked, and needing to be updated. Run Tools - report\_ip\_status to determine the state of the IP imported with the Creatng a New Block Design.

Tcl Command Example: report\_ip\_status -name ip\_status



**3.3.3、Generating Output Products for Block Designs**

These output products include fles such as a Verilog or VHDL instantaton template, or HDL wrapper fles, to facilitate integratng the block design into the current project, design constraints fles (XDC) that are included to provide tming or physical constraints for the block design, and synthesized netlists or design checkpoints to support the block design.

Once the block design is created and generated you need to then instantate it into your design by selectng either the block design RMB → Create Wrapper, or by instantatng the block design in your own RTL.

If you want to modify the wrapper, select the Copy Generated Wrapper to allow user edits,otherwise select Let Vivado Manage Wrapper to auto-update.

**3.3.4、Instantiating Block Designs into the Current Project**

The HDL wrapper provides a Verilog module declaraton, or VHDL enttydeclaraton for the block design, and creates an instance of the block design module in the wrapper. You can edit the instance defniton in the HDL wrapper and cut and paste it into the design hierarchy as needed.

**3.4、 Working with Vitis HLS Sources**

The AMD Vits™ High-Level Synthesis (HLS) tool transforms a C specifcaton into a register

transfer level (RTL) implementaton that you can synthesize into an AMD device. You can write C specifcatons in C, C++, or SystemC, and the AMD device provides a massively parallel

architecture with benefts in performance, cost, and power over traditonal processors.See Vits High-Level Synthesis User Guide (UG1399).

**3.5、Working with Model Composer Sources**

Model Composer is a model-based design tool that enables rapid design exploration and accelerates the path to production for AMD devices through automatic code generation. Model Composer is built as an AMD toolbox that fits into the MathWorks Simulink® software, which is an add-on product to the MATLAB® software that provides an interactive, graphical environment for modeling, simulating, analyzing and verifying system-level designs.

You can express your algorithms in Model Composer using blocks from the Model Composer library as well as user-imported custom blocks. Model Composer transforms your algorithmic specifications to packaged IP blocks using automatic optimizations and leveraging the high-level synthesis technology of Vitis HLS. See Vitis Model Composer User Guide (UG1483).

**3.6、Working with System Generator Sources**

AMD System Generator for DSP is a design tool that combines RTL source files, Simulink and MATLAB software models, and C/C++ components of a DSP system into a single simulation and implementation environment. Refer to the Vitis Model Composer User Guide (UG1483).

A System Generator design is often a sub-design that is incorporated into a larger HDL design. The recommended flow is to package the DSP module as an IP core in the Vivado Design Suite, to be added to the AMD IP catalog and integrated into any level of the design hierarchy as a sub module as described in Working with IP Sources,or imported into the top-level of the design.

**3.7、Editing Source Files**

**3.7.1、Using the Find/Replace in Files Commands**

• Enter any text string, including wildcards (\*, ?, #, +), or regular expression as search criteria.

• Use the filtering options to search source files, constraint files, and report files.

**3.7.2、Using HDL Language Templates**

The Vivado IDE provides templates for many Verilog, VHDL, and XDC structures, including AMD Parameterized Macros (XPMs) and library primitives. To view the templates:

1. In the Vivado IDE Text Editor, select the Language Templates toolbar button 

2. Select Tools → Language Templates.

**Using Xilinx Parameterized Macros**

XPMs are simple customizable solutions for common use cases in an HDL flow, such as RAM or ROM, clock domain crossings, and FIFOs. XPMs are SystemVerilog HDL code delivered with the Vivado Design Suite, and can be found in the ./data/ip/xpm folder of the software

installation. They can be thought of as parameterized IP, with default values for parameters that can be changed to meet design requirements.

The types of XPMs include:

• XPM\_MEMORY with various RAM and ROM memory structures

• XPM\_CDC with various safe Clock Domain Crossing (CDC) logic implementations

• XPM\_FIFO for synchronous and asynchronous FIFO structures

**Enabling Xilinx Parameterized Macros**

When using the Project Mode, the Vivado tool will parse the files added to the project and automatically recognizes the XPMs. However, when using XPMs in the Non-Project mode you must issue the auto\_detect\_xpm command prior to the synth\_design command.

**Using XPMs**

You can use any XPM language templates in your design. The parameters available for a specific XPM are explained in the instantiation template displayed in the Language Templates window. Select and copy the contents of the instantiation template and paste it into your own source file, or use the Insert Template command from the popup menu in the Text Editor. You do not need to copy the comments for the instantiation template into your design source file. You can change the instance name and wire ports as needed to fit the XPM instance into your design, and modify parameters/generics according to the documentation provided as comments in the language template.

Some XPMs deliver constraints that are defined in Tcl files located in the ./data/ip/xpm/

<xpm>/tcl folder of a specific XPM. The constraints are applied during synthesis and appear in the synthesis log file along with other constraints that are processed.

**3.7.3、Cross Probing to Source Files**

The Vivado IDE provides cross probing to RTL source files from the following windows:

• Schematic window (RTL elaborated, synthesis, or implementation)

• Netlist window (after synthesis or implementation)

• Device window (for an implemented design)

**Tcl Commands for Cross Probing to Source Files**

You can use the FILE\_NAME and LINE\_NUMBER properties on a cell to get information about

where the cell is located in the RTL source. You can then open the RTL source in a text editor and navigate to the appropriate line number. Following is the associated Tcl command:

• Tcl Command: get\_cells

• Tcl Command Example: The following example shows how to use the get\_cells Tcl command in an open design to get a specific instance of a cell, query the properties of that

cell object, and report the file and line of interest:

**set** cellName dac\_spi\_i0

**set** fileName [**get\_property** FILE\_NAME [**get\_cells** $cellName]]

**set** lineNum [**get\_property** LINE\_NUMBER [**get\_cells** $cellName]]

**puts *"Cell: $cellName is instanced in file: $fileName \***

**at line number $lineNum*"***

**3.7.4、Using Alternate Text Editors**

**3.8、Working with Simulation Sources**

The Vivado IDE stores simulation source files in simulation sets that display in folders in the

Sources window, and are remotely referenced or stored in the local project directory. See Adding or Creating Simulation Source Files in the Vivado Design Suite User Guide: Logic Simulation (UG900).

**3.8.1、Adding and Creating Simulation Source Files**

**3.9、Working with Constraints**

The Vivado IDE supports the Xilinx design constraint (XDC) and Synopsys design constraint

(SDC) file formats. The SDC format is for timing constraints while the XDC format is for both

timing and physical constraints. Constraints can include placement, timing, and I/O restrictions. You can create multiple constraint sets to experiment with various types of constraints, or store multiple versions of constraints. Each constraint set can contain one or more constraint files.

An implemented design saves a snapshot of the constraints used during the implementation run along with a reference to the original constraint file lines. When opening an implemented design, the constraints loaded from the implementation run might be older than the implementation constraints from the constraints set in the project. This can cause the loss of newer constraints in the project constraint files when you save the design from an implemented run after adding or editing the constraints in memory. Generally, the Vivado IDE manages these revision issues and prompts you to take the appropriate action as needed. However, you should be aware of the potential conflict between the constraints in memory and the constraints files in the constraints set associated with the implementation run.

In the Vivado IDE, the following windows enable you to create and work with constraints:

• Timing Constraints Window: Shows all XDC file timing constraints for the project in a table

format. You can an interactively edit existing constraints, which are saved back to the source

file, or create new constraints.

• Device Constraints Window: Enables you to set various SelectIO interface constraints on

displayed banks.

• Physical Constraints Window: Enables you to create and manage Pblocks.

**TIP: Select Tools → Timing → Constraints Wizard on a synthesized design to create a top-level XDC file based on design methodologies recommended by . The wizard guides you through specifying clocks, setting up input and output constraints, and properly constraining cross-clock domain clock groups.**

**3.9.1、Adding and Creating Constraint Files**

**IMPORTANT!** Constraints are read in the order they appear in a constraint set. If the same

constraint is defined more than once in a constraint file, or in more than one constraint file, the last definition of the constraint overwrites earlier constraints.

**3.9.2、Setting the Target XDC File**

By default, in a new constraint set, there is no target XDC file. When you create new constraints, you must set a target XDC file when you save the constraints.

If an XDC file is set as a target, the word "(target)" appears next to the file name in the Sources window. You can change the target XDC file at any time using the Set as Target Constraint File right-click menu command in the Sources window.

当新增约束条件时，需要指定这些条件写入哪一个约束文件（约束文件可以有多个）。指定写入的那个文件就是目标文件。

**3.9.3、Referencing Original XDC Files or Copying Files**

As with other source files, you can reference XDC files from a remote location or copy the files locally into the project directory. If your project references remote files, the Vivado IDE

automatically detects changes to the referenced source file and prompts you to Reload the

design with the latest files.

**3.9.4、Using Constraint Sets**

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

**Creating and Editing Constraint Sets**

If the same constraint appears in more than one constraint file, the last file read has precedence in defining the constraint.

**Defining the Active Constraint Set**

If more than one constraint set exists, you must designate the active constraint set. The Vivado IDE uses the active constraint set by default when you launch the synthesis or implementation runs or when you open an elaborated, synthesized or implemented design.

**Creating Constraints Sets Using the Save Constraints As Command**

At any time in the design flow you can also create a copy of the active constraint set using the Save Constraints As command. Select File → Constraints → Save As to open the Save Constraints As dialog box, and enter a new constraint set name in which to save all constraints.

**3.9.5、Exporting Constraints**

To export all the constraints applied to the in-memory design to a single constraints file, select File → Export → Export Constraints.

To export I/O constraints, select File → Export → Export I/O Ports, and generate an XDC file.

**3.9.6、Enabling or Disabling Constraint Files**

**3.9.7、Changing the Constraint Evaluation Order**

You can reorder user constraints within the associated constraint set. In the Sources window, drag and drop the XDC files to rearrange the order.

To get an ordered list of all XDC files that the Vivado IDE processes, use the following command in the Tcl Console: report\_compile\_order -constraints. This lists all the constraints in the design, including user constraints and IP.

**3.10、Working with Sources in Non-Project Mode**

**Chapter 4： RTL Analysis （Elaboration）**

The AMD Vivado™ Design Suite has three views of the design represented by the source files: the elaborated RTL design, the synthesized design, and the placed and routed design.

The Vivado Design Suite offers many analysis capabilities for an RTL design.

• RTL linting

• Visualize design details with Schematic and Hierarchy windows

• Cross probe between windows

• Run design rule checks (DRCs)

• Check messaging

• Search the RTL netlist produced with the Find command

• Create and apply constraints at the RTL level

Note: You cannot run timing analysis at this stage.

**4.1、Running the RTL Linter（短棉绒，此处为代码检查工具）**

The RTL linter is an important tool that analyzes RTL code, catching issues ranging from syntax errors to QoR issues. The RTL linter is a Vivado built-in feature that analyzes the RTL design code and provides a detailed report for the violations. You can activate the linter functionality by clicking **RTL Analysis  →  Run Linter** in the Flow Navigator. （然而并没找到）

When you click Run Linter, Vivado analyzes the RTL source files and provides a detailed report by opening the linter window. When you right-click anywhere on the violations report, you will have the following options.

• Create Waiver（弃权，放弃）: Opens Create Waiver window with the Tcl command preview to let you waive specific violations in the current design. When you click OK, the waived check box displays an increased number of total waivers in the design.

• Export To Spreadsheet: Opens Export Table to a spreadsheet format to export the violations report to a spreadsheet to be opened in Microsoft Office or on a Open Office platform.

**4.2、Elaborating the Design in Project Mode**

Enabled RTL source files in the project are elaborated automatically during synthesis. You can also elaborate source files manually for constraint development and RTL netlist exploration.

**4.2.1、Tcl Command for Elaborating the Design in Project Mode**

Following is the associated Tcl command:

• Tcl Command: synth\_design -rtl -name <project\_name>

• Tcl Command Example: synth\_design -rtl -name rtl\_1

**4.2.2、Viewing Elaboration Messages**

The Vivado IDE provides the following views into the logical design hierarchy:

• Netlist View: Shows an expandable logic tree of the RTL hierarchy and primitives.

• Hierarchy View: Shows a graphical representation of the logic hierarchy.

• Schematic View: Shows the logic and hierarchy in an explorable schematic representation.

To invoke the RTL Schematic window for any selected logic, do one of the following:

• Select **Tools → Schematic** or **F4**.

• In the Netlist window, select **Schematic** from the right-click menu.

To invoke the Hierarchy view for any selected logic, do one of the following:

• Select **Tools → Show Hierarchy** or **F6**.

• In the Netlist or Schematic window, select **Show Hierarchy** from the right-click menu.

To open the instantiation or definition of any selected logic in the RTL source file, select the

object and select Go To Source or Go to Definition from the right-click menu. The Vivado IDE

opens the appropriate source file with the specific instance highlighted.

**4.2.3、Running Methodology Checks**

The Vivado Design Suite provides automated methodology checks based on the UltraFast Design Methodology Guide for FPGAs and SoCs (UG949) using the Report Methodology command.

You can generate a methodology report on an opened, elaborated, synthesized, or implemented design. For an elaborated design, the methodology report checks the XDC and RTL files.

**RECOMMENDED:** Running the methodology report allows you to find design issues early during the elaboration stage prior to synthesis, which saves time in the design process. It is highly recommended that you run these checks on your design and address any issues identified.

**Running Report Methodology**

1. From the Flow Navigator under RTL Analysis, select Open Elaborated Design.

2. Once the design has been elaborated, select Report Methodology from the Flow Navigator under RTL Analysis. Alternatively, select Reports → Report Methodology.

Note: Alternatively, you can enter this command in the Tcl Console: report\_methodology -name <results\_name>.

3. In the Report Methodology dialog box, set the options, and click OK.

**Analyzing the Methodology Report**

If violations are found, the Methodology window opens. The window displays the violations,

grouped under the various rule categories.

**4.2.4、Reporting DRCs**

**RECOMMENDED:** Running RTL DRCs enables you to find design issues early, during the elaboration stage prior to synthesis, which saves time over the course of your design.

**Selecting DRC Rules**

1. From the Flow Navigator under RTL Analysis, select Open Elaborated Design.

2. Once the design has been elaborated, select Report DRC from the Flow Navigator under RTL Analysis. Alternatively, select **Reports → Report DRC**.

Note: Alternatively, you can enter this command in the Tcl Console: report\_drc -name

<results\_name>.

3.In the Report DRC dialog box, set the following options, and click OK.

**Analyzing DRC Violations**

If violations are found, the DRC window opens. The DRC window displays the rule violations

found, grouped under the various rule categories defined in the Run DRC dialog box.

In addition, the violation properties are shown in the Violation Properties window by default. In the DRC window, you can also select Violation Properties from the right-click menu to open the Violation Properties window. The Violation Properties window shows both a General view of the DRC rule violation and specific Details of the design elements that violate the rule. The Details view includes links to specific design objects that violate the DRC. Click the links to view the design object in the Netlist window, the Device window, the Schematic window, or the source RTL file.

**Tcl Command for Running RTL DRCs**

Following is the associated Tcl command:

• Tcl Command: report\_drc

• Tcl Command Example: report\_drc -name drc\_1

Note: By default, a text-based report is produced. You can use the -name option to create an

interactive tab for the report.

**4.3、Elaborating the Design in Non-Project Mode**

In Non-Project Mode, you can perform elaboration of the RTL. You can also cross probe back to the RTL and run DRCs. Cross probing requires that you load the Vivado IDE using the

start\_gui Tcl command. You can perform DRCs with or without the Vivado IDE.

Following is a script that sources various files and elaborates the RTL using the synth\_design Tcl command with the -rtl option. The script also loads the Vivado IDE so you can cross probe back to the RTL source from the schematic or netlist.

*# create\_bft\_batch.tcl*

*# bft sample design*

*# A Vivado script that demonstrates a very simple RTL-to-bitstream batch\*

*flow*

*#*

*# NOTE: typical usage would be "vivado -mode tcl -source \*

*create\_bft\_batch.tcl"*

*#*

*# STEP#0: define output directory area.*

*#*

**set** outputDir ./Tutorial\_Created\_Data/bft\_output

file mkdir $outputDir

*#*

*# STEP#1: setup design sources and constraints*

*#*

**read\_vhdl** -library bftLib [ glob ./Sources/hdl/bftLib/\*.vhdl ]

**read\_vhdl** ./Sources/hdl/bft.vhdl

**read\_verilog** [ glob ./Sources/hdl/\*.v ]

**read\_xdc** ./Sources/bft\_full.xdc

*#*

*# STEP #2 Elaborate the RTL and start the GUI for interaction*

*#*

**synth\_design** -top bft -part xc7k70tfbg484-2 -rtl

**start\_gui**

*# Use stop\_gui to quit the GUI and return back to the Vivado IDE Tcl\*

*command line*

**Chapter 5： Debugging the Design**

Debugging an FPGA design is a multi-step, iterative process. Like most complex problems, it is best to break the FPGA design debugging process down into smaller parts, for example, by

focusing on making a smaller section of the design work rather than trying to make the whole design work at one time.

You can use this design and debug methodology in any combination of the following design flow stages:

• RTL-level design simulation

• In-system debugging

In addition to using the Set up Debug wizard, you can also use Tcl commands to create, connect, and insert debug cores into your synthesized design netlist. For more information on debugging, see the Vivado Design Suite User Guide: Programming and Debugging (UG908).