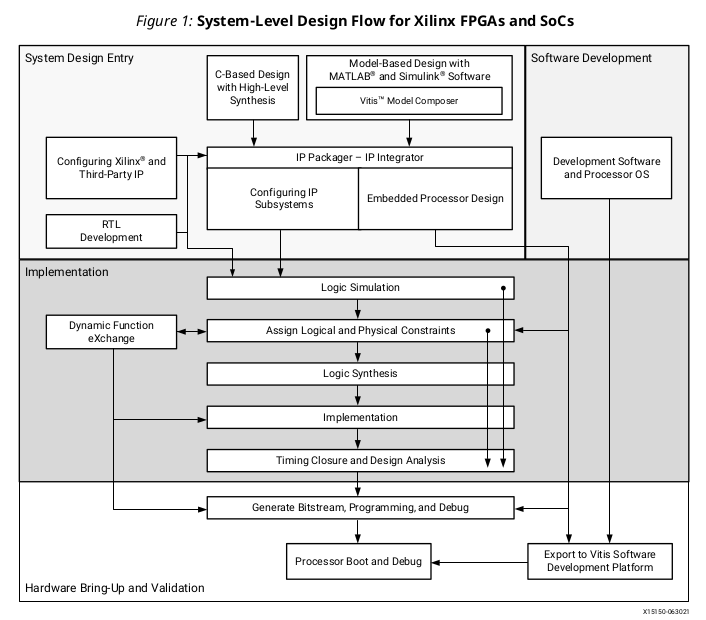
Vivado Design Flows Overview

**1. Vivado System-Level Design Flows**



（1）System Design Entry

（2）Implementation

（3）Hardware Bring-up and Validation

（4）Software Development

a. RTL-to-Bitstream Design Flow

* RTL Design

Vivado synthesis and implementation support multiple source file types, including Verilog, VHDL, SystemVerilog, and XDC.

* 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. Xilinx IP utilizes the AXI4 interconnect standard to enable faster system-level

integration.

* 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. For more information, see the Vivado

Design Suite User Guide: Designing IP Subsystems Using IP Integrator (UG994).

* I/O and Clock Planning
* Xilinx Platform Board Support
* Synthesis
* Design Analysis and Simulation
* Placement and Routing
* Hardware Debug and Validation

Debug signals can be identified in the RTL design, or inserted after synthesis and are processed throughout the flow. You can add debug cores to the RTL source files, to the synthesized netlist, or in an implemented design using the using the Engineering Change Order (ECO) flow.

b. Alternate RTL-to-Bitstream Design Flows

* Accelerated Kernel Flows

The Xilinx ® Vitis™ unified software platform introduces acceleration use cases into Vivado ®

flows. In this design methodology, Vivado is used to create a platform which is consumed by the

Vitis software platform to add accelerated kernels.For more information on platform creation, see Vitis Unified Software Platform Documentation: Application Acceleration Development

(UG1393).

* Embedded Processor Design

Data hand-off between the hardware and software flows, and validation across these two domains is critical for success. The embedded processor design flow is described in the following resources:

Vivado Design Suite User Guide: Embedded Processor Hardware Design (UG898)

Vivado Design Suite Tutorial: Embedded Processor Hardware Design (UG940)

UltraFast Embedded Design Methodology Guide (UG1046)

* Model-Based Design Using Model Composer

Model Composer is a model-based graphical design tool that enables rapid design exploration

within the MathWorks MATLAB ® and Simulink ® products and accelerates the path to production

for Xilinx devices through automatic code generation. For information, see the Model Composer

User Guide (UG1262).

* Model-Based DSP Design Using Xilinx System Generator

You create the DSP functions using System Generator as a standalone tool, and then package your System Generator design into an IP module that can be included in the Vivado IP catalog. From there, the generated IP can be instantiated into your Vivado design as a submodule. For more information, see the Vivado Design Suite User Guide: Model-Based DSP Design Using System Generator (UG897).

* High-Level Synthesis C-Based Design

The C-based High-Level Synthesis (HLS) tools within the Vivado Design Suite enable you to

describe various DSP functions in the design using C, C++, and SystemC. HLS lets you simulate the generated RTL directly from its design environment using C-based test benches and simulation. C-to-RTL synthesis transforms the C-based design into an RTL module that can be packaged and implemented as part of a larger RTL design, or instantiated into an IP integrator block design.

The HLS tool flow and features are described in the following resources:

Vivado Design Suite User Guide: High-Level Synthesis (UG902)

Vivado Design Suite Tutorial: High-Level Synthesis (UG871)

* Dynamic Function Exchange Design

Dynamic function exchange (DFx) allows portions of a running Xilinx device to be reconfigured in

real-time with a partial bitstream, changing the features and functions of the running design.

The DFx tool flow and features are described in the following resources:

Vivado Design Suite User Guide: Dynamic Function eXchange (UG909)

Vivado Design Suite Tutorial: Dynamic Function eXchange (UG947)

* Hierarchical Design

Hierarchical Design (HD) flows enable you to partition a design into smaller, more manageable

modules to be processed independently. For more information, see the Vivado Design Suite User Guide: Hierarchical Design (UG905).

**2. Vivado Design Suite Use Models**

（1）Working with the Vivado Integrated Design Environment (IDE)

（2）Working with Tcl

（3）Understanding Project Mode and Non-Project Mode

（4）Using Third-Party Design Software Tools

（5）Interfacing with PCB Designers

a. Working with the Vivado Integrated Design Environment (IDE)

For more information on the Vivado IDE, see the Vivado Design Suite User Guide: Using the Vivado IDE (UG893).

Command line: vivado -mode gui

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.

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

b. Working with Tcl

For more information about using Tcl and Tcl scripting, see the Vivado Design Suite User Guide: Using Tcl Scripting (UG894) and Vivado Design Suite Tcl Command Reference Guide (UG835). 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).

Launching the Vivado Design Suite Tcl Shell:

vivado -mode tcl

Launching the Vivado Tools Using a Batch Tcl Script:

vivado -mode batch -source <tcl\_script>

Using Xilinx Vivado Store

please refer to the documentation on the GitHub for the following repositories:

Xilinx/XilinxTclStore: tcl scripts

Xilinx/XilinxBoardStore: board files

Xilinx/XilinxCEDStore: example files?

c. Working in Project Mode and Non-Project Mode

Some users prefer the design tool for automatically managing their design flow process and design data, while others prefer to manage sources and process themselves. The Vivado Design Suite uses a project file (.xpr) and directory structure to manage the design source files, store the results of different synthesis and implementation runs, and track the project status through the design flow. This automated management of the design data, process, and status requires a project infrastructure. For this reason, Xilinx refers to this flow as the Project Mode.

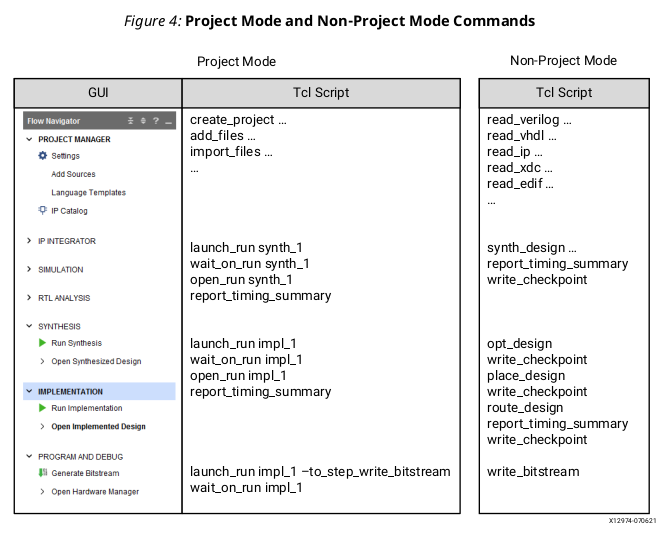
Other users prefer to run the FPGA design process more like a source file compilation, to simply

compile the sources, implement the design, and report the results. This compilation style flow is

referred to as the Non-Project mode.

The main distinctions are that Non-Project mode processes the entire design in memory. No files are written to disk. While Project mode creates and maintains a project directory structure on disk to manage design sources, results, and project settings and status.

Usually the Vivado IDE offers many benefits for the Project Mode, such as the Flow Navigator graphical workflow interface. Tcl commands are the simplest way to run the Non-Project Mode.



d. Interfacing with PCB Designers

For more information see:

Vivado Design Suite User Guide: I/O and Clock Planning (UG899)

Vivado Design Suite QuickTake Video: I/O Planning Overview

Vivado Design Hub: I/O and Clock Planning

3. Using Project Mode

Design Runs are launched with wrapper Tcl scripts that consolidate the various implementation commands and automatically generates standard reports. Run strategies only apply to Project Mode.

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

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.

a. Creating Projects

Different Types of Projects:

**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.

**Example Project**: You can explore several example projects, including example Zynq ® -7000

SoC or MicroBlaze™ embedded designs with available Xilinx evaluation boards.

**DFx**: You can dynamically reconfigure an operating FPGA design by loading a partial bitstream

file to modify reconfigurable regions of the device.

Manage source files in projects:

**Using Remote, Read-Only Sources:**

Remote 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 → Project → Archive command creates a ZIP file for the entire project.

**Creating a Tcl Script to Recreate the Project :**

In the Vivado IDE, the File → Project → Write Tcl command creates a Tcl script you can run to

recreate the entire project, including the source files, IP, and design configuration.

**Working with a Revision Control System:**

The Vivado Design Suite uses and produces files throughout the design flow that you can manage with a revision control system. Such as Git.

b. Performing System-Level Design Entry

Vivado Design Suite User Guide: System-Level Design Entry (UG895)

Vivado Design Suite User Guide: I/O and Clock Planning (UG899)

Vivado Design Suite User Guide:Using Constraints (UG903)

ISE to Vivado Design Suite Migration Guide (UG911)

c. Working with IP

The Vivado Design Suite provides an IP-centric design flow that lets you configure, implement,

verify, and integrate IP modules to your design from various design sources. Reference to Vivado Design Suite User Guide: Designing with IP (UG896).

You can also package custom IP using the IP-XACT protocol and make it available through the

Vivado IP catalog. Xilinx IP uses the AMBA ® AXI4 interconnect standard to enable faster system-

level integration.

**Configuring IP:**

The available methods to work with IP in a design are as follows:

• Use the managed IP flow to customize IP and generate output products.

• Use IP in either Project or Non-Project modes by importing or reading the created Xilinx core

instance (XCI, Xilinx Core Instance) file.(in the sources directory to the project)

• Access the IP catalog from a project to customize and add IP to a design.

This license status information is available for IP cores used in a project using Report IP Status by

selecting Reports → Report IP Status.

**The Vivado Design Suite generates the following IP output products:**

• Instantiation template

• RTL source files and XDC constraints

• Synthesized design checkpoint (default)

• Third-party simulation sources

• Third-party synthesis sources

• Example design (for applicable IP)

• Test bench (for applicable IP)

• C Model (for applicable IP)

**Out-of-Context Design Flow:**

IP cores that are added to a design from the Vivado IP catalog default to use the out-of-context

flow. Block designs created in the Vivado IP integrator also default to the OOC flow when

generating output products.

The Vivado Design Suite also supports global synthesis and implementation of a design, in which

all modules, block designs, and IP cores, are synthesized as part of the integrated top-level design.

What the difference?

When run in global mode, Vivado synthesis has full visibility of design constraints. When run in OOC mode, estimated constraints are used during synthesis.

Vivado Design Suite User Guide: Synthesis (UG901)

Vivado Design Suite User Guide: Hierarchical Design (UG905)

**IP Constraints :**

Many IP cores contain XDC constraint files that are used during Vivado synthesis and

implementation.

**Packaging Custom IP and IP Subsystems**

The Vivado Design Suite lets you package custom IP or block designs into IP to list in the Vivado IP catalog for use in designs or in the Vivado IP integrator.

Vivado Design Suite User Guide: Creating and Packaging Custom IP (UG1118)

and Vivado Design Suite Tutorial: Creating, Packaging Custom IP (UG1119).

**d、Creating IP Subsystems with IP Integrator:**

The Vivado IP integrator enables the creation of Block Designs (.bd), or IP subsystems with multiple IP stitched together using the AXI4 interconnect protocol. including ——

embedded processor-based designs using Zynq® UltraScale+™ MPSoC, Zynq®-7000 SoC, and MicroBlaze™ processors.

It can instantiate High-Level Synthesis modules from Vivado HLS,

DSP modules from System Generator,

and custom user-defined IP as described in Packaging Custom IP and IP Subsystems

Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator (UG994)

Vivado Design Suite User Guide: Embedded Processor Hardware Design (UG898).

**Building IP Subsystems**

The interactive block design capabilities of the Vivado IP integrator make the job of configuring

and assembling groups of IP easy.

* Block Design Containers

Block design containers allow a block diagram to reference a secondary block diagram. This

enables a design to be partitioned into several block diagrams. Block design containers also

support several instances of child block diagrams in a parent block diagram.

* Referencing RTL Modules in Block Designs

The Module Reference feature of the Vivado IP Integrator lets you quickly add a module or entity

definition from a Verilog or VHDL source file directly into your block design.

* Designer Assistance

To expedite the creation of a subsystem or a design, the IP integrator offers Block Automation

and Connection Automation.

* Using the Platform Board Flow

You can download support files for partner developed boards from the partner websites or from the Xilinx Vivado Store.

**Validating IP Subsystems**

IP integrator runs basic design rule checks in real time as the design is being assembled. However, there is still a potential for design errors.You can run design validation by selecting Tools → Validate Design or through the Tcl command validate\_bd\_design.

The Validate Design command applies design rule checks on the block design and reports warnings and/or errors found in the design. Xilinx recommends validating a block design to catch errors that would otherwise be found later in the design flow. Running design validation also runs Parameter Propagation on the block design. Parameter Propagation enables IP integrator to automatically update the parameters associated with a given IP based on its context and its connections in the design.

**Generating Block Design Output Products**

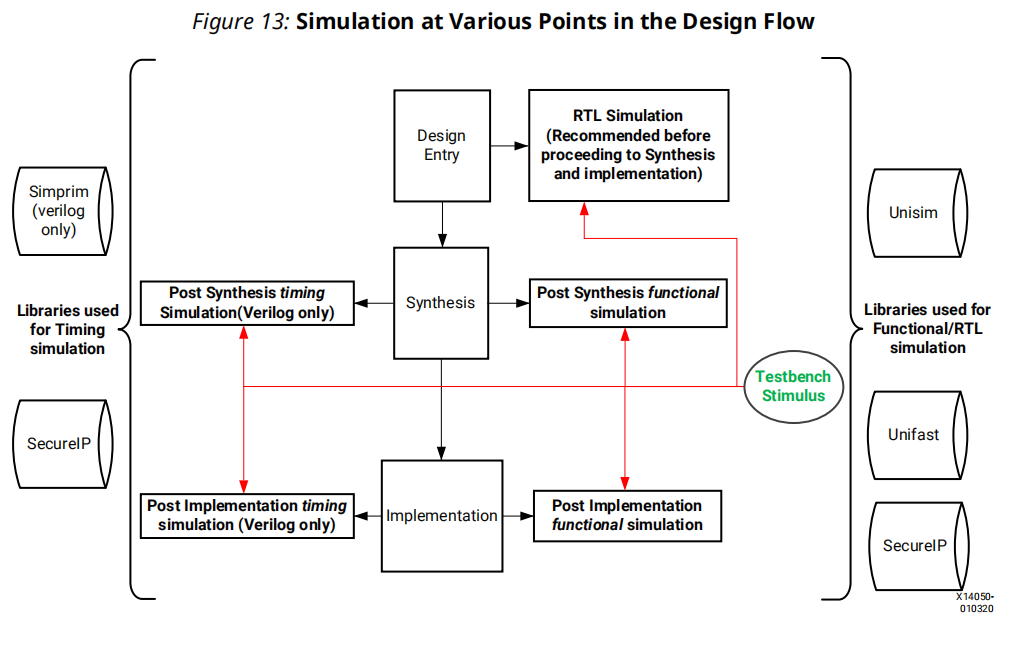
You can generate the block design by right-clicking on the block design (in the Sources window) and selecting Generate Output Products from the pop-up menu. In the Vivado Design Suite Flow Navigator you can also select IP Integrator → Generate Block Design.

**Integrating the Block Design into a Top-Level Design**

An IP integrator block design can be integrated into a higher-level design or it can be the highest

level in the design hierarchy. You can perform a higher-level instantiation of the block design by selecting the block design in the Vivado IDE Sources window and selecting Create HDL Wrapper.

**e、Logic Simulation：**



Vivado Design Suite User Guide: Logic Simulation (UG900)

**Simulation Flow Overview**

The following are some key suggestions related to simulating in the Vivado Design Suite.

1. Run behavioral simulation before proceeding with synthesis and implementation. Issues identified early will save time and money.

2. Infer logic wherever possible. Instantiating primitives adds significant simulation runtime cost.

3. Always set the Target Language to Mixed unless you do not have a mixed mode license for

your simulator.

4. Turn off the waveform viewer when not in use to improve simulation performance.

5. In the Vivado simulator, turn off debug during xelab for a performance boost.

6. In the Vivado simulator, turn on multi-threading to speed up compile time.

7. When using third-party simulators, always target supported versions.

8. Make sure incremental compile is turned on when using third-party simulators.

9. Use the Xilinx Tcl command export\_simulation to generate batch scripts for selected simulators.

10. Generate simulation scripts for individual IP, BDs, and hierarchical modules as well as for the

top-level design.

11. If you are targeting a 7 series device, use UNIFAST libraries to improve simulation performance.

Note: The UNIFAST libraries are not supported for UltraScale device primitives.

**Compiling Simulation Libraries**

Vivado delivers precompiled simulation libraries for use with the Vivado simulator, as well as

precompiled libraries for all the static files required by Xilinx IP. When simulation scripts are

created, they reference these precompiled libraries.

When using third-party simulators, you must compile Xilinx simulation libraries prior to running

Simulation.You can run the compile\_simlib Tcl command to compile the Xilinx simulation libraries for the target simulator. You can also issue this command from the Vivado IDE by selecting Tools →  Compile Simulation Libraries.

**Simulation Time Resolution**

Xilinx recommends that you run simulations using a resolution of 1 ps. Because most of the simulation time is spent in delta cycles, there is no significant simulator performance gain by using coarser resolution with the Xilinx simulation models. However There is no need to use a finer resolution.

**Functional Simulation Early in the Design Flow**

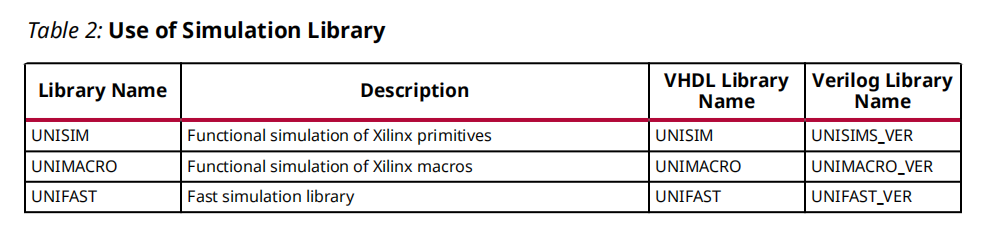
Use functional or register transfer level (RTL) simulation to verify syntax and functionality. This

first pass simulation is typically performed to verify the RTL or behavioral code and to confirm

that the design is functioning as intended.

**Using Structural Netlists for Simulation**

After synthesis or implementation, you can perform netlist simulation in functional or timing mode.



**Timing Simulation**

Xilinx supports timing simulation in Verilog only. You can export a netlist for timing simulation

from an open synthesized or implemented design using the File → Export → Export Netlist

command in the Vivado IDE, or by using the write\_verilog Tcl command.

If you decide to skip timing simulation, you should make sure of the following:

• Ensure that your STA constraints are absolutely correct. Pay special attention to exceptions.

• Ensure that your netlist is exactly equivalent to what you intended through your RTL. Pay

special attention to any inference-related information provided by the synthesis tool.

**Simulation Flow**

* Integrated Simulation——launch\_simulation Tcl command.
* Batch Simulation——export\_simulation Tcl command

**f、Running Logic Synthesis and Implementation：**

Vivado Design Suite User Guide: Synthesis (UG901)

Vivado Design Suite User Guide: Implementation (UG904)

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).

**Configuring Synthesis and Implementation Runs**

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.

• **Resetting Runs**

In the Flow Navigator, you can right-click Synthesis or Implementation, and use the following

popup menu commands to reset runs.

**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.

**Performing Implementation with Incremental Compile**

You can specify the incremental compile flow when running Vivado implementation to facilitate

small design changes.

You can specify the Set Incremental Compile option in the Implementation Settings dialog box in

the Vivado IDE, or by using the Set Incremental Compile command from the right-click menu of

the Design Runs window. You can also use the read\_checkpoint Tcl command with the -incremental option, and point to a routed design checkpoint to use as a reference.

**Closing Timing Using Intelligent Design Runs**

Intelligent design runs (IDR) uses a multi-stage run approach to automatically close timing on a

design. This flow can be invoked in the GUI by right clicking on implementation run in the design

runs window and selecting "Close Timing Using Intelligent Design Runs" or in Tcl, by creating a

new run using the IDR flow and properly setting the reference run. For information on Intelligent

design runs, see Chapter 8 of Vivado Design Suite User Guide: Design Analysis and Closure

Techniques (UG906)

**Implementing Engineering Change Orders (ECOs)**

Engineering change orders (ECOs) are modifications to an implemented design, with the intent to

minimize impact to the original design. The advantage of the ECO flow is fast turn-around time by taking advantage of the incremental place and route features of the Vivado tool.

**g、Viewing Log Files, Messages, Reports, and Properties：**

**Viewing or Editing 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 in which you can view and set device configuration and bitstream-related properties.

**h、Opening Designs to Perform Design Analysis and Constraints Definition：**

**Opening an Elaborated RTL Design**

There is no FPGA technology mapping during RTL elaboration. 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.

**Opening a Synthesized Design**

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 interactively configure and assign I/O ports in the synthesized design and run DRCs.

You can configure and implement debug core logic in the synthesized design to support test and

debug of the programmed device.

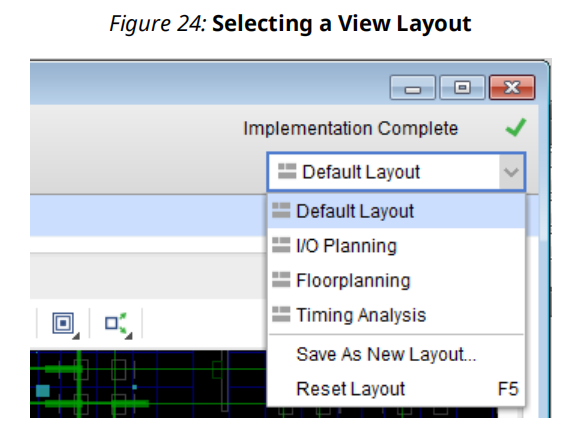
**Opening an Implemented Design**

**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.

**Using View Layouts to Perform Design Tasks**



**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).

**Running Reports: DRC, Power, Utilization Analysis**

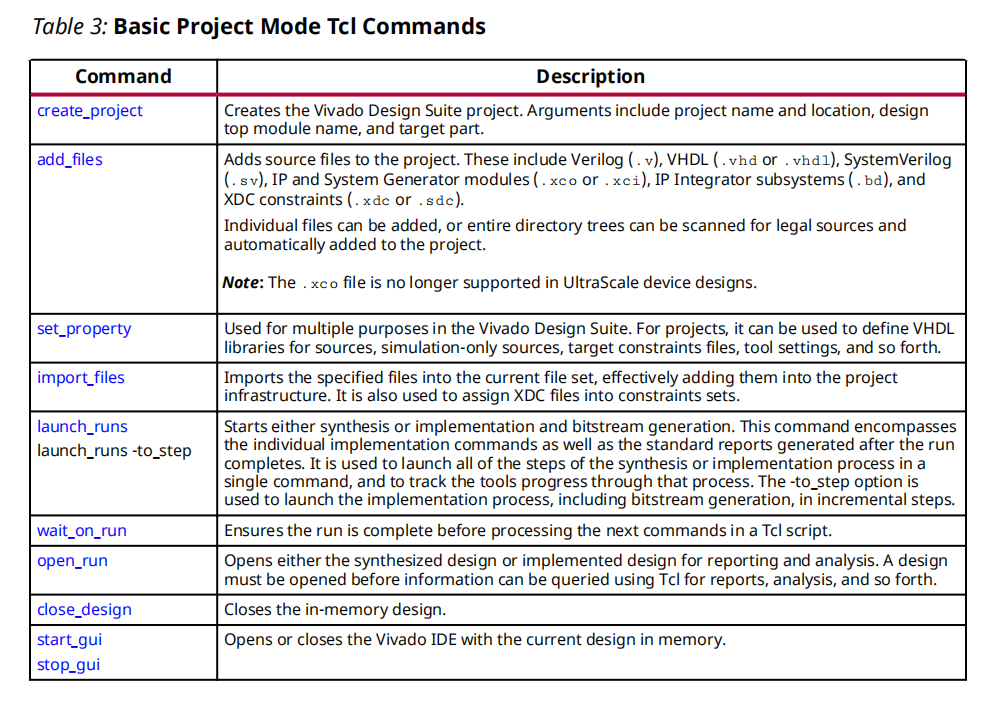
Report strategy is a property of a run. Setting report strategy on the run generates all specified reports when the run is launched.

1. **Device Programming, Hardware Verification, and Debugging：**

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.

**j、Using Project Mode Tcl Commands：**

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.



**4、Using Non-Project Mode**

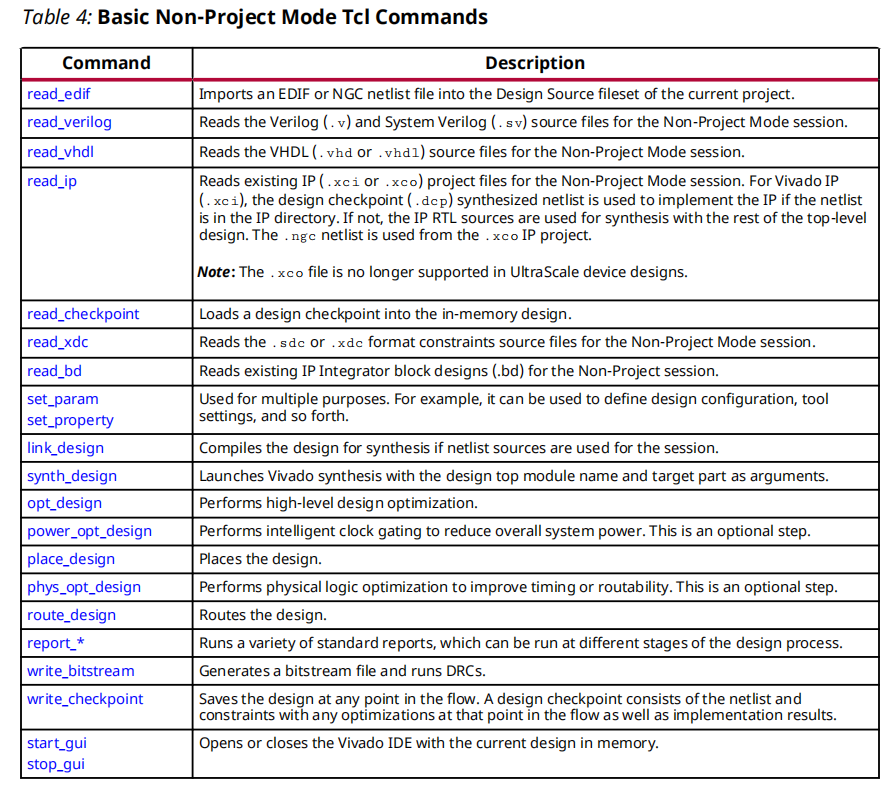
Non-Project Mode enables you to have full control over each design flow step. You can take

advantage of a compile-style design flow.

In Non-Project Mode, you use Tcl commands to compile a design through the entire flow. In this

mode, an in-memory project is created to let the Vivado® tools manage various properties of a

design, but the project file is not written to disk, and the project status is not preserved.



**5、Source Management and Revision Control Recommendations**

**a、Generating a Script to Recreate a Design**

For a project flow, a script to recreate your design can be generated manually or by using the

write\_project\_tcl command.

**b、Revision Controlling Projects with Only RTL Sources**

**c、Revision Controlling Xilinx IP**

**d、Managing Custom IP Repositories**

**e、Revision Controlling Block Diagrams**

**f、Other Files to Revision Control**

• XDC files containing design constraints

• Simulation test benches

• HLS IP

• Pre/post Tcl hook scripts used for synthesis or implementation

• Incremental compile DCPs

• ELF files

These files reside in the project.srcs directory or the project.util directory.

**g、Output Files to Optionally Revision Control**

• Simulation scripts for third-party simulators generated by export\_simulation. Because

these are typically hand-off files between design and verification, you might want to snapshot

them at different stages of the design process.

• XSA （Xilinx Solution Archive）files. These are hardware hand-off files between Vivado and Vitis™ software platform.

• Bitsteams/PDIs.

• LTX files for hardware debug

• Intermediate DCP files created during the flow

• IP output products.

• Archiving Designs. The archive\_design command can compress your entire project into a zip file.

**h、Managing Hardware Manager Projects and Sources**