Notice of Disclaimer

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 the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; 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 Critical Applications: http://www.xilinx.com/warranty.htm#critapps.

©Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, 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...
Table of Contents

Vivado Design Flows Tutorial ............................................................................................................................... 5
Design Flows Overview ........................................................................................................................................ 5
Software Requirements ...................................................................................................................................... 7
Hardware Requirements .................................................................................................................................. 7
Tutorial Design Description .............................................................................................................................. 7
Preparing the Tutorial Design Files .................................................................................................................. 7

LAB 1: Using the Non-Project Design Flow ........................................................................................................ 8
Step 1: Examine the Example Script .................................................................................................................. 8
Step 2: Start Vivado with the Example Design .................................................................................................. 9
Step 3: Synthesize the Design and Examine Reports ....................................................................................... 10
Step 4: Open the Vivado IDE ........................................................................................................................... 10
Step 5: Define Timing Constraints and I/O Planning ....................................................................................... 11
Step 6: Export the Modified Constraints and Close the Vivado IDE .............................................................. 15
Step 7: Implement the Design and Examine Reports ....................................................................................... 16
Step 8: Open the Implemented Design Checkpoint ....................................................................................... 16
Step 9: Examine Implementation Results and Timing ................................................................................... 17
Step 10: Exit Vivado and Examine the Log File ............................................................................................. 19

LAB 2: Using the Project Design Flow ............................................................................................................... 21
Step 1: Create a Project .................................................................................................................................. 21
Step 2: Use the Sources Window and the Text Editor .................................................................................... 26
Step 3: RTL Design Elaboration and Exploration ........................................................................................... 28
Step 4: Use the IP Catalog ............................................................................................................................... 29
Step 5: Run Behavioral Logic Simulation ....................................................................................................... 29
Step 6: Configure Synthesis and Implementation ........................................................................................... 30
Step 7: Synthesize and Implement the Design ............................................................................................... 31
Step 8: Analyze the Synthesized Design........................................................................................................... 32
Step 9: Analyze the Implemented Design........................................................................................................ 34
Step 11: Generate a Bit file................................................................................................................................... 37
Step 12: Exit Vivado and Examine Log File .................................................................................................... 37
Step 13: Create a Tcl Script from the Journal File ....................................................................................... 38
Step 14: Run the Project based Batch Script................................................................................................. 40
Step 15: Open the Vivado IDE to Check Design Status ............................................................................ 41

Summary............................................................................................................................................................................................... 41
Vivado Design Flows Tutorial

Design Flows Overview

This tutorial introduces the use models and design flows recommended for use with the Xilinx Vivado Integrated Design Environment (IDE). This tutorial describes the basic steps involved in taking a small example design from RTL to bitstream, using two different design flows as explained below. Both flows can take advantage of the Vivado IDE, or be run through batch Tcl scripts. The Vivado Tcl API provides considerable flexibility and power to help set up and run your designs, as well as perform analysis and debug.

Flexible Use Models

Some users prefer the design tool to automatically manage their design flow process and design data, while some 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 the FPGA design process to act more like a compile engine, to simply implement and report on the design. This compilation style flow is referred to as the Non-Project Mode. The Vivado Design Suite easily accommodates both of these use models.

Non-Project Mode

This use model is for script-based users that do not want Vivado tools to manage their design data or track design state. The Vivado tools simply read the various source files and compile the design through the entire flow in-memory. At any stage of the implementation process, you can generate a variety of reports, run design rule checks (DRCs), and write design checkpoints. Throughout the entire flow, you can open the design in-memory, or any saved design checkpoint, in the Vivado IDE for design analysis or netlist/constraint modification. Source files, however, are not available for modification in the IDE when running the Non-Project Mode. It is also important to note that this mode does not enable project-based features such as source file and run management, cross-probing back to source files, design state reporting, etc. Essentially, each time a source file is updated on the disk; you must know about it and reload the design.

There are no default reports or intermediate files created within the Non-Project Mode. You must direct the creation of reports or design checkpoints with Tcl commands.
Project Mode

This use model is for users that want the Vivado tools to manage the entire design process. This includes features like source file, constraint and results management, integrated IP design, and cross probing back to sources, to name a few. In Project Mode, the Vivado tools create a directory structure in order to manage the design source files, IP data, synthesis and implementation run results and related reports. The Vivado Design Suite manages and reports the status of the source files, configuration and the state of the design. You can create and configure multiple runs to explore constraint or command options. In the Vivado IDE, you can cross-probe implementation results back to the RTL source files. You can also script the entire flow with Tcl commands, and open Vivado IDE as needed.

Tcl Commands

The Tcl commands and scripting approach vary depending on the design flow used. When using the Non-Project Mode, the design is compiled using `read_verilog`, `read_vhdl`, `read_edif`, `read_ip`, and `read_xdc` commands. The Vivado Design Suite creates an in-memory design database, to pass to synthesis, simulation, and implementation. When using Project Mode, you can use the `add_files`, `import_files`, and `add_directories` commands to create the project infrastructure needed to manage sources and track changes. Replace the individual “atomic” commands, `synth_design`, `opt_design`, `place_design`, and `route_design`, and `write_bitstream` in the Batch flow, with an all-inclusive command called `launch_runs`. The `launch_runs` command groups the atomic commands together with other commands to generate default reports and track the run status. The resulting Tcl run scripts for the Project Mode are different from the Non-Project Mode. This tutorial covers both the Project Mode and Non-Project Mode, as well as the Vivado IDE.

Many of the analysis features discussed in this tutorial are covered in more detail in other tutorials. Not every command or command option is represented here. To view the entire list of Tcl commands provided in the tools, consult the Vivado Design Suite Tcl Command Reference Guide (UG835).

This Tutorial contains two labs that can be performed independently.

**Lab 1: Using the Non-Project Design Flow**

- Walk through a sample run script to implement the bft design.
- View various reports at each step.
- Review the vivado.log file.
- Write design checkpoints.
- Open the Vivado IDE after synthesis to review timing constraint definition and I/O planning and demonstrate methods to update constraints.
- Open the implemented Design Checkpoint to analyze timing, power, utilization and routing.
Lab 2: Using the Project Based Design Flow

- Create a new Project.
- Walk through implementing the bft design using the Vivado IDE.
- View various reports at each step.
- Open the synthesized design and review timing constraint definition, I/O planning and design analysis.
- Open the implemented design to analyze timing, power, resource utilization, routing, and cross-probing.
- Exit and create a Tcl script from vivado.jou file (with launch_runs).
- Run the design again using the newly created Tcl script.
- Open the Project in the Vivado IDE, and check that the design status is correct from the batch run.

Software Requirements

This tutorial requires that the 2012.2 Vivado Design Suite software release or later is installed.

Hardware Requirements

The supported Operating Systems include Redhat 5.6 Linux 64 and Windows 64 and 32 bit.
Xilinx recommends a minimum of 2 GB of RAM when using the Vivado tool.

Tutorial Design Description

The sample design used throughout this tutorial consists of a small design called bft. There are several VHDL and Verilog source files in the bft design, as well as a XDC constraints file.

The design targets an xc7k70T device. A small design is used to allow the tutorial to be run with minimal hardware requirements and to enable timely completion of the tutorial, as well as to minimize the data size.

Preparing the Tutorial Design Files

You can find the files for this tutorial in the Vivado Design Suite examples directory at the following location:

- `<Xilinx_2012.2_install_area>/Vivado/<version>/examples/Vivado_Tutorial`
You can also extract the provided zip file, at any time, to write the tutorial files to your local directory, or to restore the files to their starting condition.

Extract the zip file contents from the software installation into any write-accessible location.

- `<Xilinx_2012.2_install_area>/Vivado/<version>/examples/Vivado_Tutorial.zip`

The extracted Vivado_Tutorial directory is referred to as the `<Extract_Dir>` in this Tutorial.

**Note:** You will modify the tutorial design data while working through this tutorial. You should use a new copy of the original Vivado_Tutorial directory each time you start this tutorial.

---

### LAB 1: Using the Non-Project Design Flow

This lab focuses on the Non-Project design flow and the associated Tcl commands.

---

### Step 1: Examine the Example Script

1. Open the example script, `<Extract_Dir>/Vivado_Tutorial/create_bft_batch.tcl`, in a text editor and review the different steps.

   **STEP#0:** Define output directory location.
   **STEP#1:** Setup design sources and constraints.
   **STEP#2:** Run synthesis, write design checkpoint and generate reports.
   **STEP#3:** Run placement and optimization commands, write design checkpoint and generate reports.
   **STEP#4:** Run routing command, write design checkpoint and generate reports.
   **STEP#5:** Generate a bitstream.

   Notice that many of the Tcl commands are commented out. You will run them manually, one at a time.

2. Leave the example script open, as you will copy and paste commands from it later in this tutorial.
Step 2: Start Vivado with the Example Design

- On Linux,
  a. Change the directory to `<Extract_Dir>/Vivado_Tutorial`
  b. Enter:

    ```
    >vivado –mode tcl –source create_bft_batch.tcl.
    ```

- On Windows: open a Xilinx Command Prompt window:
  a. Start > All Programs > Xilinx Design Tools > ISE Design Suite 14.2 > Accessories > ISE Design Suite 64 Bit Command Prompt:
  b. Change to the directory where the lab material is stored.

    ```
    >cd `<Extract_Dir>/Vivado_Tutorial`
    ```
  c. Enter:

    ```
    >vivado –mode tcl –source create_bft_batch.tcl
    ```

After the script execution has completed you will be presented with the Vivado Tcl prompt: `Vivado%`.

You can enter additional Tcl commands from the Vivado tool prompt.

---

1 Your Vivado Design Suite installation may called something different than Xilinx Design Tools on the Start menu. You can also use the 32 Bit command prompt as needed.
Step 3: Synthesize the Design and Examine Reports

1. Copy and paste the `synth_design...` command from the example run script into the Vivado Tcl Prompt and wait for synthesis to complete. You can paste into the Command Prompt window using the popup menu, by clicking the right mouse button.

   **Note:** The command in the example script is a comment. Do not copy the leading ‘#’ character, or your command will also be interpreted as a comment.

2. Examine the synthesis report as it scrolls by.
3. Copy and paste the `write_checkpoint`, the `report_utilization` and the `report_timing` commands into the Vivado Tcl Prompt (without the comment character ‘#’).
4. Open another window to look at the files created in the output directory. On Windows, it may be easier to use the file browser.
   
   `<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data/bft_output`

5. Use a text editor to open the various report (*.rpt) files that were created.

---

Step 4: Open the Vivado IDE

Even though a Vivado project has not been created on disk, the in memory design is available in the tool, so from the Tcl shell you can open the Vivado IDE to view the design.

1. Open the IDE using the `start_gui` command.

   *Vivado% start_gui*
Step 5: Define Timing Constraints and I/O Planning

Prior to implementation, you must often define timing and physical constraints for the design. The Vivado tools let you load constraints from constraints file(s), or enter constraints interactively using the IDE.

Since the design does not have a project in Non-Project Mode, the Vivado IDE does not enable source file or run management. You are effectively analyzing the current in memory design. The Vivado Flow Navigator and other Project based commands are also not available in Non-Project Mode.

Define Timing Constraints

1. Open the Timing Constraints window.

Window > Timing Constraints
Step 5: Define Timing Constraints and I/O Planning

Figure 3: Define Timing Constraints

Notice the two clock constraints currently defined in the design, displayed in the table on the right side of the Timing Constraints window.

The values of currently defined constraints can be modified by directly selecting them in the table. You can click on the values, but do not modify them at this time.

A tree view of the different types of constraints displays on the left side of the Timing Constraints window.

1. Double-click on Create Clock(2) at the top of the Constraint type list.

   The Create Clock dialog box opens to help you define clock constraints. Notice the Tcl Command on the bottom displays the XDC command that will be executed.

   Do not create or modify any timing constraints at this time.
Step 5: Define Timing Constraints and I/O Planning

2. Click **Cancel**

3. **Close** the Timing Constraints window by clicking the **X** in the window tab.

The Vivado Design Suite offers a variety of features for design analysis and constraint assignment. Other tutorials cover these features in detail, and they are only mentioned here. Feel free to examine some of the features under the Tools menu.

**I/O Planning**

Vivado has a comprehensive set of capabilities for performing and validating I/O pin assignments. These are covered in greater detail in the I/O Planning Tutorial.

1. Open the I/O Planning view layout by selecting **I/O Planning** from the Layout Selector pull down located in the main Toolbar.
Step 5: Define Timing Constraints and I/O Planning

2. In the Package window, select a placed I/O Port (shown as an orange block inside a pin).
3. Drag the selected I/O Port onto another pin site in the same I/O bank.
4. Look in the I/O Ports window at the bottom for the device site and Port signal name.
5. Examine the data displayed in the I/O Port Properties window. Click each of the tabs at the bottom of the window.
6. Remember the Port signal and I/O Site names of the port you moved (write them down if necessary) because you will look for the LOC constraint of the placed port in the output XDC file, after implementation.

Non-Project Mode enables the use of the Vivado IDE at various stages of the design process. The current netlist and constraints are loaded into memory in the IDE, enabling analysis and modification. Any changes to the logic or the constraints are live in memory and are passed to the downstream tools. This is quite a different concept than with the current ISE tools that require saving and reloading files.
Step 6: Export the Modified Constraints and Close the Vivado IDE

The modified constraints can be output for use later in the flow. You can also save design checkpoints that include the latest changes. You will explore design checkpoints later in this tutorial.

1. Use the Export Constraints command to output a modified XDC constraints file with the new IO Loc constraint value.
   
   **File > Export > Export Constraints**

2. Enter a name and location for the file and click **OK**.

   Notice the checkbox for **Export fixed location constraints only**. When this is enabled, only the LOC constraints of fixed cells are exported, rather than of all placed cells. For a more detailed description of fixed versus unfixed cells, refer to the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)*.

3. Open the exported constraints file in the Text Editor.
   
   **File > Open File**

4. Set the **File of type** filter at the bottom of the Open File dialog to **All Files**.

5. Browse to select the newly exported constraints file and click **OK**.

6. Notice the file reflects the I/O Port placement change you made earlier.

   You can open any ASCII file in the Text Editor. This is helpful for editing Tcl scripts, constraints files, and viewing reports. The Text Editor is context sensitive when displaying some file types, such as Verilog, VHDL, XDC, and Tcl, and highlights keywords and comments.

7. Close the Text Editor when you are finished examining the constraints file.

8. Select the **Tcl Console** tab at the bottom of the IDE and click in the Tcl Console command line to activate it. Enter the following command:

   ```
   > stop_gui
   ```

   The Vivado IDE closes, and you are returned to the Tcl prompt in the Command Prompt window.
Step 7: Implement the Design and Examine Reports

1. Open the `create_bft_batch.tcl` script, or bring it to the front.
   
   Individually copy and paste the Tcl commands in the script, in order from `opt_design` to `write_bitstream`.

2. Examine each command and notice the various messages produced as the commands are run.

3. Examine the files created in the output directory.
   
   `<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data/bft_output`

4. Use a text editor to open the various report (*.rpt) files that were created.

5. Open the `bft_impl.xdc` file.

6. Validate that the design has been implemented with the I/O Port constraint that you modified earlier.

Step 8: Open the Implemented Design Checkpoint

The Vivado IDE can open any saved design checkpoint. This “snapshot” of the design can be opened in the Vivado IDE or Tcl shell for synthesis, implementation, and analysis.

1. Open the Vivado IDE again.

   `Vivado% start_gui`

   This loads the active design in-memory into the IDE. You will now load the implemented design checkpoint.

2. Open the implemented checkpoint.

   Use `File > Open Checkpoint` and browse to select the checkpoint file:

   `<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data/bft_output/post_route.dcp`

3. Select Yes to close the in-memory design.

4. If prompted, select Don’t Save to save the constraints in the open design.

Now you can use the visualization and analysis capabilities of the IDE, working from a design checkpoint.
Step 9: Examine Implementation Results and Timing

Vivado has an extensive set of features to examine the design and device data from a number of perspectives. You can generate standard reports for power, timing, utilization, clocks, etc. With the Tcl API, the custom reporting capabilities in the Vivado tools are extensive.

1. Run report_timing_summary to analyze timing data.
   
   **Tools > Timing > Report Timing Summary**

2. In the Report Timing Summary dialog, click **OK** to accept the default run options.

3. Examine the information available in the timing Summary window. Select the various categories from the tree on the left side of the Timing Summary window and examine the data displayed.

4. Run report_timing to perform timing analysis.
   
   **Tools > Timing > Report Timing**

5. In the Report Timing dialog, click **OK** to accept the default run options.

6. Select the first Path listed in the Timing results window.

7. Maximize or float the Path Properties window to look at the path details.

![Figure 6: Maximize the Path Properties Window](image-url)
8. Restore the Path Properties window by clicking the **Restore** button, or the **Dock** button, in the window banner.

9. Select the **Schematic** command in the popup menu of the Timing results window to open the Schematic window with the selected path displayed.

   **Note:** Alternatively, you can press the **F4** function key to open the Schematic window.

10. Double-click on an instance pin, instance or wire in the Schematic window to expand the connectivity, and traverse the design hierarchy.

11. Close the Schematic window or click the Device window tab to bring it to the front.

12. In the Device window, select the **Routing Resources** button to display the device routing.

   ![Figure 7: Displaying the Device Routing](image)

   Notice the Device window displays and highlights the routing for the selected path.

13. Select the **Auto Fit Selection** button in the Device window toolbar menu to enable the Vivado IDE to automatically zoom into selected objects.
Step 10: Exit Vivado and Examine the Log File

14. Select some additional paths from the Timing results window.
15. Examine the routing for the selected paths in the Device window.
16. Expand the Tools main menu and examine the available analysis features.
   Many of these Design Analysis features are covered in other PlanAhead and Vivado tutorials.

**Step 10: Exit Vivado and Examine the Log File**

Exit the Vivado IDE:

1. Select the Tcl Console window tab and click in Tcl command line and type the following:
   >stop_gui
2. Exit Vivado:
   Vivado% exit
3. Examine the Vivado log (vivado.log) file. On Windows, it may be easier to use the file browser.
   <Extract_Dir>/Vivado_Tutorial/vivado.log
Notice the log file contains the history and results of all Tcl commands executed during the Vivado session.

4. Examine the Vivado journal (vivado.jou) file. On Windows, it may be easier to use the file browser.

   `<Extract_Dir>/Vivado_Tutorial/vivado.jou`

Notice the journal file contains only the Tcl commands executed during the Vivado session, without the added details recorded in the log file. The journal file is often helpful when creating Tcl scripts from prior design sessions, as you will see in the next lab.
LAB 2: Using the Project Design Flow

In this lab, you will learn about the Project Mode features for project creation, source file management, design analysis, constraint definition, and run management. You will walk through the entire flow using the example design starting with the Vivado IDE. Then you will examine some of the major features in the IDE. Most of these features are covered in detail in other tutorials. Finally, you will create a batch run script to implement the design project and see how easy it is to switch between running Tcl scripts and working in the Vivado IDE.

**Step 1: Create a Project**

Start Vivado

1. On Linux:
   a. Change to the directory where the lab material is stored.
      
      \texttt{>cd <Extract\_Dir>/Vivado\_Tutorial}
   b. Launch Vivado
      
      \texttt{>vivado}

1. On Windows:
   a. Before clicking the Desktop icon, you will first define where to write the Vivado log files.
   b. Right-click the Vivado 2012.2 Desktop icon and select \textit{Properties}.
   c. Under the \textit{Shortcut} tab, set the \textit{Start in} value to the extracted Tutorial directory:
      
      \texttt{<Extract\_Dir>/Vivado\_Tutorial/}
   d. Click \textit{OK}.  

Step 1: Create a Project

1. After Vivado opens, select **Create New Project** on the Getting Started page.
2. Click **Next** in the New Project page.
3. Enter **project_bft** for the Project name and select **<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data** for the Project location.
4. Click **Next**.
5. Select **RTL Project** as the **Project Type** and click **Next**.

---

**Figure 9: Setting the location to write Vivado log files on Windows**

- d. Click the **Vivado 2012.2** Desktop icon to start Vivado.
7. Click **Add Files...**
   a. Browse to `<Extract_Dir>/Vivado_Tutorial/Sources/hdl/`
   b. Use the Ctrl key to select `async_fifo.v, bft.vhdl, bft_tb.v, FifoBuffer.v`.

8. Press **OK**.

![Add RTL Source Files to the Project](image)

**Figure 11: Add RTL Source Files to the Project**

9. Click **Add Directories...**
   a. Select the `<Extract_Dir>/Vivado_Tutorial/Sources/hdl/bftLib` directory
   b. Press **Select**.

10. In the Add Sources list, click on the **HDL Sources for** field for the `bft_tb.v` and change Synthesis and Simulation to **Simulation only**, as shown below.

11. In the Add Sources list, click on the Library field for the bftLib and change the Library from **work** to **bftLib**, as shown below.
Step 1: Create a Project

12. Enable the check box for **Copy Sources into Project**.
13. Press **Next**.
14. Press **Next** to skip the Add Existing IP page, since you will not be adding IP at this time.
15. Press **Add Files...** on the Add Constraints page.
17. Press **OK**.
18. Enable the check box for **Copy Constraints into Project**.
19. Click **Next**.
Step 1: Create a Project

20. On the Default Part page, click the Family filter and select the Kintex-7 family.
21. Scroll to the top of the list and select the xc7k70tfbg484-2 part.
22. Click **Next**.

23. Click **Finish** on the New Project Summary page.

---

### Step 2: Use the Sources Window and the Text Editor

The Vivado tool allows you to add different file types as design sources, including Verilog, VHDL, EDIF, NGC format cores, SDC and XDC constraints files, and specific simulation sources. These files display by category in the Sources window. The tabs at the bottom of the Sources window enable the files to be displayed by Hierarchy, Library or Compile Order.

**Note:** Support for NCF/UCF constraints has been deprecated in the 2012.2 release. You should migrate existing UCF constraints to XDC format. Refer to the *Vivado Design Suite Migration Methodology Guide (UG912)* for more information.

The Vivado IDE includes a context sensitive text editor to create or develop RTL sources. Third party text editors can also be configured for use with the Vivado tools.

### Explore the Sources Window and Project Summary

1. Examine the information in the Project Summary. More detailed information is presented as the design progresses through the design flow.

2. Examine the Sources window and expand the **Design Sources**, **Constraints** and **Simulation Sources** folders.

---

![Figure 15: Viewing Sources](image)
The Design Sources folder helps keep track of VHDL and Verilog source files and libraries. Notice the **Hierarchy** tab is displayed by default. The **Libraries** tab groups source files by file type, while the **Compile Order** tab shows the file order used for synthesis.

3. Select the **Libraries** tab and the **Compile Order** tabs in the Sources window and notice the different ways that Sources are listed.

4. Select the **Hierarchy** tab.

5. Right-click in the Sources to review the commands available in the Sources window popup menu.

**Explore the Sources Window Commands and Text Editor**

1. Select one of the VHDL sources in the Sources window.

2. Right-click to review the commands available for source file management in the Sources window popup menu.

3. Select **Open File**, and use the scroll bar to browse the text in the Text Editor.

**Note:** Alternatively, you can double-click on source files in the Sources window to open them in the Text Editor.

5. With the cursor in the Text Editor, right-click and select **Find in Files**. Notice also, the **Replace in Files** command.

   The Find in Files dialog box opens with various search options.

![Find in Files dialog box](image)

**Figure 16: Using the Find in Files Command**
5. Enter `clk` in the Find what: field, and click **Find**.

The Find in Files window displays in the messaging area at the bottom of the Vivado IDE.

![Figure 17: Viewing the Find in Files Results](image)

6. In the Find in Files window, expand one of the displayed files, and select an occurrence of `clk` in the file.

Notice that the Text Editor opens the selected file and displays the selected occurrence of `clk` in the file.

7. Close the **Find in Files – Occurrences** window.

8. Close the open Text Editor windows.

The next few steps highlight some of the design configuration and analysis features available prior to running synthesis. Skip ahead to Step 6 to begin Synthesis.

---

### Step 3: RTL Design Elaboration and Exploration

The Vivado IDE includes an RTL analysis and IP customizing environment. There are also several RTL Design Rule Checks (DRCs) to examine ways to improve performance or power on the RTL design.

1. Select **Open Elaborated Design** in the Flow Navigator to elaborate the design.

2. Ensure that the Layout Selector pull down menu in the main Toolbar has **Default Layout** selected.

   The Elaborated Design enables various analysis views including an RTL Netlist, Schematic, and Graphical Hierarchy. The views have a “cross-select” feature, which helps you to debug and optimize the RTL.

3. Explore the logic hierarchy in the RTL Netlist window and use the Schematic traversal commands to examine the Schematic.

4. Select any logic instance in the Schematic and right-click to select the **Go to Instantiation** or **Go to Definition** commands.
Notice the Text Editor opens the RTL source file for the selected cell with the logic instance highlighted. With **Go to Instantiation**, the RTL source containing the instantiation of the selected module is opened. In the case of the **Go to Definition** command, the RTL source file containing the module definition is opened.

5. Close the Text Editor windows.
6. Click on the **Messages** window at the bottom of the Vivado IDE, and examine the messages.
7. Click the **Collapse All** button in the Messages toolbar.
8. Expand the **Elaborated Design – synth_design** messages.
   Notice there are links in the messages to open the RTL source files associated with a message.
9. Click one of the links and the Text Editor opens the RTL source file with the relevant line highlighted.
10. Close the Text Editor windows.
11. Close the Elaborated Design by clicking on the X on the right side of the Elaborated Design window banner and click **OK** to confirm.

---

**Step 4: Use the IP Catalog**

The Xilinx IP Catalog provides access to the Vivado IP configuration and generation features. You can sort and search the Catalog in a variety of ways. IP can be customized, generated, and instantiated.

1. Click the **IP Catalog** button in the Flow Navigator.
2. Browse the IP Catalog and examine the information available and filtering capabilities.
3. Expand the **Basic Elements** and **Counters** Folders.
4. Double-click on **DSP48 Macro** and notice the Customize IP dialog is opened directly within Vivado.

   To learn more about IP configuration and implementation see the *Vivado Design Suite User Guide: IP Design (UG896)* and the *Vivado Design Suite Tutorial: IP Design (UG939)*.
5. Click **Cancel** to close the Customize IP dialog
6. Close the IP Catalog by clicking on the X on the right side of the window banner.

---

**Step 5: Run Behavioral Logic Simulation**

The Vivado IDE integrates the Vivado Simulator, which enables you to add and manage simulation sources in the project. You can configure simulation options, and create and manage simulation source sets. You can run behavioral simulation on RTL sources, prior to synthesis.
1. Click the **Simulation Settings** command in the Flow Navigator in the Simulation section.

2. Examine the settings available under each tab and click **Cancel**.

3. Click the **Run Simulation** command in the Flow Navigator

4. Examine the Simulation environment.
   
   Simulation is covered in detail in the *Vivado Design Suite User Guide: Logic Simulation (UG900)* and the *Vivado Design Suite Tutorial: Logic Simulation (UG937)*.

5. Close the Simulation Design by clicking the X icon on the Simulation view banner.

6. Click **No** when prompted to save changes.

---

**Step 6: Configure Synthesis and Implementation**

Before launching the tools, you will review and modify some of the synthesis options.

1. In the Flow Navigator, select **Synthesis Settings** under Synthesis.

---

**Figure 18: Open Synthesis Options**
The Synthesis Project Settings dialog is displayed as shown above.

Notice the Strategy is set to a Vivado Synthesis strategy. The Vivado Design Suite can also use XST to synthesize a run prior to Vivado implementation by selecting an XST 14 strategy.

2. After reviewing the synthesis options, select the **Implementation** button on the left side of the Project Settings dialog box.

   The Project Settings change to reflect the Implementation settings. You will not modify the Implementation options in this lab, you can view the currently available options in the Implementation settings.

3. Click **Cancel** to close the Project Settings dialog box.

   You are now ready to launch Vivado Synthesis and Implementation.

### Step 7: Synthesize and Implement the Design

After configuring the Synthesis and Implementation runs, you can:

- Use the **Run Synthesis** command to run only synthesis.
- Use the **Run Implementation** command, which will first run synthesis if it has not been run, then run implementation as directed.
- Use the **Generate Bitstream** command, which will first run synthesis and then run implementation if they have not been run, then write the bitstream for programming the Xilinx device.

For this tutorial, we will run these steps one at a time.

1. In the Flow Navigator, click on the **Run Synthesis** button, and wait for this task to complete. Notice the progress bar in the upper-right corner of the Vivado IDE.

   Meanwhile, you can continue browsing Vivado IDE windows as the synthesis task is running in the background. You will notice that the Log window displays the synthesis log at the bottom of the IDE. This is also available through the Reports window.

   After synthesis has completed, the Vivado IDE prompts you to choose the next step.

2. Select **Run Implementation**.

3. Click **OK**.

   The next step in this tutorial shows you how you can explore some of the design analysis and constraints definition capabilities in the open Synthesized Design, while you wait for implementation to complete.
Step 8: Analyze the Synthesized Design

Opening the Synthesized Design enables design analysis, timing constraint definition, I/O planning, floorplanning and debug core insertion. These features are covered in other tutorials, but you can take a quick look in this step.

1. Select **Open Synthesized Design** in the Flow Navigator and wait for the design to load.
   Notice that as the Vivado IDE opens the Synthesized Design, the implementation is still running in the background.
   At some point while you are in the Synthesized Design, implementation will complete. The Vivado IDE prompts you to choose the next step.

2. Click **Cancel** to close the dialog.
   You will open the Implemented Design after you are finished examining the Synthesized Design features.

3. Ensure that the Layout Selector pull down menu in the main Toolbar has **Default Layout** selected.

4. Click the **Reports** window tab at the bottom of the Vivado IDE.
   If this tab does not show by default, you can open it via the menu: **Windows > Reports**.

5. Double click the **Vivado Synthesis Report** to examine the report.

6. Double click the **Utilization Report** to examine the report.

7. Close all reports when you have finished examining them.

8. Click the **Messages** window tab at the bottom of the Vivado IDE.
   If this tab does not show by default, you can open it via the menu: **Windows > Messages**.

9. Click the **Collapse All** button to condense all of the Messages.

10. Expand the **Synthesis** messages.

11. Scroll through the Synthesis messages and notice the links to specific lines within source files. Click some of the links and notice the source file opens in the Text Editor with the appropriate line highlighted.

12. **Close** the open files.
13. Experiment with the **Warnings** and **Infos** filters on the top of the Messages window to display or hide the different types of messages.

   If the design had Critical Warnings or Errors, you would see filters for those messages as well.


   Examine the Timing Summary Results window showing timing estimates prior to implementation. Click on some of the reporting categories in the tree on the left side of the Timing Summary Results window.

15. Select **Report Power** in the Flow Navigator and click **Run** to run with default options.

   Examine the Power Results window showing power estimates prior to implementation. Click on some of the reporting categories in the tree on the left side of the Power Results window.

The Vivado IDE is interactive. It enables live editing of design constraints and netlists. When you save the design, constraint changes are written back to the original source XDC files. Alternatively, you can save the changes to a new constraints file to preserve the original constraints. This flexibility supports exploration of alternate timing and physical constraints, including floorplanning, while keeping the original source files intact.
Feel free to explore the different analysis and constraints capabilities of the Vivado tools. Do not save the design, if prompted.

### Step 9: Analyze the Implemented Design

**Open the Implemented Design**

1. Select **Open Implemented Design** in the Flow Navigator.
2. Select **Yes** to close the Synthesized Design and **Don’t Save**, if prompted.
   
   After the Implemented Design has loaded, you can see the implementation results in the Device window.

3. Click on the **Reports** window tab at the bottom of the Vivado IDE.
   
   If this tab does not show by default, you can open it via the menu: **Windows > Reports**. Select and examine some of the reports. Close each of the reports when you are done.

4. Select the **Messages** window tab at the bottom of the IDE.

   If this tab does not show by default, you can open it via the menu: **Windows > Messages**.

5. Click the **Collapse All** button to condense all of the Messages.

6. Expand the **Implementation** folder

   View the messages from the **Design Initialization**, **Opt_Design**, **Place_Design**, and **Route_Design** processes.

**Analyze Routing**

7. In the Device window, select the **Routing Resources** button to display the device routing.

   Notice Device window now displays the routing resources.

![Figure 20: Turn on Routing Resource Display](image)

8. Select the **Auto Fit Selection** button in the Device window toolbar menu to enable the Vivado IDE to zoom automatically into selected objects.
Create the Timing Report Summary

After a design has been loaded into the Vivado tools, you can generate a timing report to verify that all the timing constraints are met. If there are timing problems, you can revisit the RTL source files or design constraints to resolve any problems.


10. Click **OK** in the Report Summary Timing dialog to generate the default report.

11. On the left side pane of the Timing Summary Results window, select: **Intra-Clock Paths > bftClk > SETUP…**

12. Click on any path in the table view on the right side of the Timing Summary Results window to select it and highlight it in the Device window. Select various paths in the Timing Summary window and notice the path routing.

13. On the left side pane of the Timing Summary Results window, select: **Intra-Clock Paths > bftClk > HOLD…**

14. Click on any path in the table view on the right side of the Timing Summary Results window to select it and highlight it in the Device window. Select various paths in the Timing Summary Results window and examine the path routing.

15. Select the **Schematic** command in the popup menu of the Timing results window to open the Schematic window with the selected path displayed.

**Note:** Alternatively, you can press the **F4** function key to open the Schematic window.

16. **Close** the Schematic window.
Step 9: Analyze the Implemented Design

17. In the Device window, select the **Routing Resources** toolbar icon again to turn off routing resources.

   Notice the Device window now displays the placed instances.

18. Expand the **Tools** main menu and examine the available analysis features.

**Create the Power Report**


20. Examine the Power Results window.

21. In the **Utilization Details** area of the Power results window, click a few of the reporting categories and examine the graphical report.

22. **Close** the Power Results window.
Step 11: Generate a Bit file

Since the XDC constraints file has LOC and IOSTANDARDs constraints set for all of the I/O ports, you can generate a bitstream.

1. Click on the Generate Bitstream button in the Flow Navigator, under the Program and Debug section.

![Bitstream Generation Completed](image)

**Figure 20: Turn on Routing Resource Display**

2. After the bitstream has been generated, click Cancel in the Bitstream Generation Completed dialog box.

Step 12: Exit Vivado and Examine Log File

Vivado creates two files as it runs:

- The Vivado tools log (vivado.log) file contains the history and results of all Tcl commands executed during the Vivado session.

- The Vivado tools journal (vivado.jou) file contains only the Tcl commands executed during the Vivado session, without the added details recorded in the log file.

These files are a great way to learn the Tcl commands used by Vivado tools to perform different design tasks. It is also a great help when creating a new Tcl script for a design.

Running in batch mode is faster and takes less memory than running in the Vivado IDE. When you need multiple runs to complete a design, it is a good idea to use a Tcl script to automate the flow. You can also add report generation commands into the script after key steps, and redirect the output to specific files and directories.

1. Select **File > Exit**, or type **exit** in the Tcl command line.
2. Click **OK** to close the Vivado tool.

3. Examine the Vivado log (*vivado.log*) file. On Windows, it may be easier to use the file browser.

   `<Extract_Dir>/Vivado_Tutorial/vivado.log`

   **Note:** This is the location you entered for the **Start-in** property in Step 1 of this Lab.

2. Examine the contents and close the file.

---

**Step 13: Create a Tcl Script from the Journal File**

You will now manually create a Tcl script from the journal file that the Vivado tools automatically created as you worked from Step 1 through Step 12 of this lab. When you execute the new script, it will create a project file (.xpr) and directory structure just like when you used the Vivado IDE. If you load this project into the Vivado IDE, you will see all the results and project status displayed, as you would expect.

1. Open up **vivado.jou** file in a text editor.

2. Examine the Vivado journal (**vivado.jou**) file. On Windows, it may be easier to use the file browser.

   `<Extract_Dir>/Vivado_Tutorial/vivado.jou`

   You should see something similar to what is shown below.

   ![Figure 23: Create Run Script from Journal file](image)

**Note:** If you are using Linux, you will see different path references.
3. Remove the comment header as unnecessary (text lines starting with #).

4. Since you do not want to open the IDE in the Tcl script, remove the `start_gui` line.

5. Use the Save As command to save the file to `<Extract_Dir>/Vivado_Tutorial/run_bft.tcl`.

6. Examine the script and notice the differences from PART 1 of this Tutorial.
   
   Notice the `add_files` and `set_property` commands used for project creation, as well as the commands to set up the constraints sets. Notice that `launch_runs` is used instead of `synth_design`, etc.

   Use the `launch_runs` command when creating or running Project based designs.

   **CAUTION!** Mixing the atomic commands (`synth_design`, `opt_design` ...) with the Project based `launch_runs` command could damage the integrity of the Project, and should be avoided. The `launch_runs` command has Tcl options to run commands independently and to create custom reports. For more information on the Tcl commands, refer to the *Vivado Design Suite Tcl Command Reference Guide* (UG835).

7. With the `run_bft.tcl` script opened, search and replace the four occurrences of “project_bft” with “project_bft_batch”.

   **Note:** If you did not complete Lab #2 of the tutorial in one design session, your `vivado.jou` file will not reflect the complete tutorial design flow. In this case, you can use the `run_bft_project.tcl` script supplied in the Vivado_Tutorial directory of the software installation. That script will create a project-based design, and will run the entire design flow from RTL to bitstream.

   Notice that lines 1 through 9 of the script, create the project, bring in the source RTL and XDC files, and configure the target Xilinx part, the HDL library and the synthesis options. The script creates two runs, `synth_1` and `impl_1`, and defines the constraint set, `constrs_1`.

8. The next few lines elaborate the RTL design and simulate it. You don’t need to do that in the batch flow, so remove the following lines:

   ```
   #synth_design –rtl –name rtl_1
   #close_design
   #launch_xsim -simset sim_1 -mode behavioral
   #close_sim
   ```

9. Set up the script to report timing and power estimates after synthesis.

   The `open_run` command will open the synthesized netlist and apply the constraint set to it. You can then run the `report_timing_summary` and `report_power` commands. Remember that you ran those during the interactive session, and they are shown in the script.
10. Cut the following lines:

```
#open_run synth_1...
#set_delay_model...
#report_timing_summary...
#report_power...
```

11. Paste them after the `wait_on_run synth_1...` line.

12. Remove the `close_design` line if one is in the file.

13. Save your file. It should now look like the following:

```
create_project project_bft_batch C:/Vivado_Tutorial/Tutorial_Created_Data/project_bft_batch -part xc7k70tfbg484-2
create_files -simset sim_1
add_files -fileset sim_1 C:/Vivado_Tutorial/Sources/hdl/async_fifo.v C:/Vivado_Tutorial/Sources/hdl/bft_lib
set_property library bftLib [get_files C:/Vivado_Tutorial/Sources/hdl/bftLib/round_4.vhd]
import_files -force
import_files -fileset constrs_1 -force -noreuse C:/Vivado_Tutorial/Sources/bft_full.vhd
update_compile_order -fileset sources_1
update_compile_order -fileset sources_1
launch_runs synth_1
wait_on_run synth_1
open_run synth_1 -name netlist_1
set_delay_model -interconnect estimated
report_timing_summary -delay_type max -path_type full_clock_expanded -report_unconstrained -check_timing_verbose -report_power -results [power_1]
launch_runs impl_1
wait_on_run impl_1
open_run impl_1
current_design netlist_1
report_timing_summary -delay_type min_max -path_type full_clock_expanded -report_unconstrained -check_timing_verbose
report_power -results [power_1]
launch_runs impl_1 -to_step write_bitstream
wait_on_run impl_1
```

Figure 24: Complete the Run Script

---

**Step 14: Run the Project based Batch Script**

1. You can now execute your new TCL script, running Vivado tools in batch mode, which will run all the commands in your Tcl script and then quit Vivado when finished.

2. On Windows, open a Command Prompt window. Refer to Step 1 in Part 1 of this tutorial for instructions on how to set the PATH to find Vivado commands.

3. Change directory to `<Extract_Dir>/Vivado_Tutorial/` and type:

   ```bash
   > vivado -mode batch -source run_bft.tcl
   ```

4. Examine the Vivado log output, as it is transcripted to the Command Prompt window.

   **Note:** Since the `launch_runs` command is used, less information is echoed to the tool transcript. However, the Vivado tool is running, so please wait for it to complete. Reports and run status are also gathered in the Project and will be available after the run completes.
5. Because you ran it in batch mode, Vivado exits after it has completed running, and you are returned to the Windows command prompt.

---

**Step 15: Open the Vivado IDE to Check Design Status**

6. Launch Vivado and open the project_bft_batch.xpr file in the project_bft_batch directory that was created by your script.

   You can also do this from the command prompt:

   ```cmd
   > vivado Tutorial_Created_Data/project_bft_batch/project_bft_batch.xpr
   ```

   The Vivado IDE will launch and load the project you created with your Tcl script.

7. As you can see in the project status bar in the upper right of the Vivado IDE, the status reflects the fact that a bitstream has been generated (write_bitstream Complete). You can now visualize the implemented design by clicking the **Open Implemented Design** button in the Flow Navigator.

8. Quit Vivado when you are finished. This concludes the tutorial.

   File > Exit

---

**Summary**

After completing this tutorial, you should have learned the following:

- The differences between Project Mode and Non-Project Mode.
- How to create an RTL project in the Vivado IDE.
- Configuring the Vivado synthesis, simulation and implementation tools.
- Using the IP Catalog.
- Launching the Vivado simulator, synthesis and implementation.
- Applying constraints to the synthesized design.
- Generating timing and power reports.
- Examining routing results in the Device editor.
- Generating a bit file.
- Using a Journal file (`.jou`) to create a project based Tcl script.
- Launching a project based Tcl script from the command line.
- Switching between Vivado batch mode and the Vivado IDE.