Xilinx is disclosing this user guide, manual, release note, and/or specification (the "Documentation") to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU “AS-IS” WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© 2002–2008 Xilinx, Inc. All rights reserved.

XILINX, the Xilinx logo, the Brand Window, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.
About This Tutorial

The ISE 10.1 Quick Start Tutorial is a hands-on learning tool for new users of the ISE software and for users who wish to refresh their knowledge of the software. The tutorial demonstrates basic set-up and design methods available in the PC version of the ISE software. By the end of the tutorial, you will have a greater understanding of how to implement your own design flow using the ISE 10.1 software.

Additional Resources

To find additional documentation, see the Xilinx website at:


To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

# Preface: About This Tutorial

Additional Resources ......................................................... 3

# ISE 10.1 Quick Start Tutorial

**Getting Started** ................................................................. 7
- Software Requirements ................................................. 7
- Hardware Requirements ............................................... 7
- Starting the ISE Software ............................................. 8
- Accessing Help ............................................................ 8

**Create a New Project** ......................................................... 9

**Create an HDL Source** ....................................................... 10
- Creating a VHDL Source .............................................. 10
- Using Language Templates (VHDL) ................................ 11
- Final Editing of the VHDL Source .................................. 12
- Creating a Verilog Source ............................................ 13
- Using Language Templates (Verilog) .............................. 14
- Final Editing of the Verilog Source ............................... 15
- Checking the Syntax of the New Counter Module ............. 15

**Design Simulation** ............................................................ 16
- Verifying Functionality using Behavioral Simulation .......... 16
- Simulating Design Functionality ................................. 18

**Create Timing Constraints** ................................................. 19
- Entering Timing Constraints ........................................ 20

**Implement Design and Verify Constraints** ......................... 22
- Implementing the Design ............................................. 22
- Assigning Pin Location Constraints .............................. 23

**Reimplement Design and Verify Pin Locations** ..................... 24

**Download Design to the Spartan™-3 Demo Board** ............... 25
ISE 10.1 Quick Start Tutorial

The ISE 10.1 Quick Start Tutorial provides Xilinx PLD designers with a quick overview of the basic design process using ISE 10.1. After you have completed the tutorial, you will have an understanding of how to create, verify, and implement a design.

Note: This tutorial is designed for ISE 10.1 on Windows.

This tutorial contains the following sections:

- “Getting Started”
- “Create a New Project”
- “Create an HDL Source”
- “Design Simulation”
- “Create Timing Constraints”
- “Implement Design and Verify Constraints”
- “Reimplement Design and Verify Pin Locations”
- “Download Design to the Spartan™-3 Demo Board”

For an in-depth explanation of the ISE design tools, see the ISE In-Depth Tutorial on the Xilinx® web site at: http://www.xilinx.com/support/techsup/tutorials/

Getting Started

Software Requirements

To use this tutorial, you must install the following software:

- ISE 10.1


Hardware Requirements

To use this tutorial, you must have the following hardware:

- Spartan-3 Startup Kit, containing the Spartan-3 Startup Kit Demo Board
Starting the ISE Software

To start ISE, double-click the desktop icon,

or start ISE from the Start menu by selecting:

Start → All Programs → Xilinx ISE 10.1 → Project Navigator

Note: Your start-up path is set during the installation process and may differ from the one above.

Accessing Help

At any time during the tutorial, you can access online help for additional information about the ISE software and related tools.

To open Help, do either of the following:

• Press F1 to view Help for the specific tool or function that you have selected or highlighted.

• Launch the ISE Help Contents from the Help menu. It contains information about creating and maintaining your complete design flow in ISE.

Figure 1: ISE Help Topics
Create a New Project

Create a new ISE project which will target the FPGA device on the Spartan-3 Startup Kit demo board.

To create a new project:

1. Select **File > New Project...** The New Project Wizard appears.
2. Type **tutorial** in the Project Name field.
3. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically.
4. Verify that **HDL** is selected from the Top-Level Source Type list.
5. Click **Next** to move to the device properties page.
6. Fill in the properties in the table as shown below:
   - Product Category: **All**
   - Family: **Spartan3**
   - Device: **XC3S200**
   - Package: **FT256**
   - Speed Grade: **-4**
   - Top-Level Source Type: **HDL**
   - Synthesis Tool: **XST (VHDL/Verilog)**
   - Simulator: **ISE Simulator (VHDL/Verilog)**
   - Preferred Language: **Verilog** (or **VHDL**)
   - Verify that **Enable Enhanced Design Summary** is selected.

Leave the default values in the remaining fields.

When the table is complete, your project properties will look like the following:

![New Project Wizard - Device Properties](image)

*Figure 2: Project Device Properties*
7. Click **Next** to proceed to the Create New Source window in the New Project Wizard. At the end of the next section, your new project will be complete.

**Create an HDL Source**

In this section, you will create the top-level HDL file for your design. Determine the language that you wish to use for the tutorial. Then, continue either to the “Creating a VHDL Source” section below, or skip to the “Creating a Verilog Source” section.

**Creating a VHDL Source**

Create a VHDL source file for the project as follows:

1. Click the **New Source** button in the New Project Wizard.
2. Select **VHDL Module** as the source type.
3. Type in the file name **counter**.
4. Verify that the **Add to project** checkbox is selected.
5. Click **Next**.
6. Declare the ports for the counter design by filling in the port information as shown below:

7. Click **Next**, then **Finish** in the New Source Wizard - Summary dialog box to complete the new source file template.
8. Click **Next**, then **Next**, then **Finish**.

The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown below:
Using Language Templates (VHDL)

The next step in creating the new source is to add the behavioral description for the counter. To do this you will use a simple counter code example from the ISE Language Templates and customize it for the counter design.

1. Place the cursor just below the `begin` statement within the counter architecture.
2. Open the Language Templates by selecting `Edit → Language Templates...`
   
   **Note:** You can tile the Language Templates and the counter file by selecting `Window → Tile Vertically` to make them both visible.
3. Using the “+” symbol, browse to the following code example:
   
   
   **VHDL → Synthesis Constructs → Coding Examples → Counters → Binary → Up/Down Counters → Simple Counter**
4. With Simple Counter selected, select `Edit → Use in File`, or select the `Use Template in File` toolbar button. This step copies the template into the counter source file.
5. Close the Language Templates.

### Final Editing of the VHDL Source

1. Add the following signal declaration to handle the feedback of the counter output below the architecture declaration and above the first `begin` statement:

   ```vhdl
   signal count_int : std_logic_vector(3 downto 0) := "0000";
   ```

2. Customize the source file for the counter design by replacing the port and signal name placeholders with the actual ones as follows:
   - replace all occurrences of `<clock>` with `CLOCK`
   - replace all occurrences of `<count_direction>` with `DIRECTION`
   - replace all occurrences of `<count>` with `count_int`

3. Add the following line below the `end process;` statement:

   ```vhdl
   COUNT_OUT <= count_int;
   ```

4. Save the file by selecting **File → Save**.

When you are finished, the counter source file will look like the following:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitive in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity counter is
    Port ( CLOCK : in  STD_LOGIC;
           DIRECTION : in  STD_LOGIC;
           COUNT_OUT : out  STD_LOGIC_VECTOR (3 downto 0));
end counter;

architecture Behavioral of counter is
    signal count_int : std_logic_vector(3 downto 0) := "0000";
begin
    process (CLOCK)
    begin
        if CLOCK='1' and CLOCK'event then
            if DIRECTION='1' then
                count_int <= count_int + 1;
            else
                count_int <= count_int - 1;
            end if;
        end if;
    end process;
    COUNT_OUT <= count_int;
end Behavioral;
```
You have now created the VHDL source for the tutorial project. Skip past the Verilog sections below, and proceed to the “Checking the Syntax of the New Counter Module” section.

Creating a Verilog Source

Create the top-level Verilog source file for the project as follows:

1. Click **New Source** in the New Project dialog box.
2. Select **Verilog Module** as the source type in the New Source dialog box.
3. Type in the file name **counter**.
4. Verify that the **Add to Project** checkbox is selected.
5. Click **Next**.
6. Declare the ports for the counter design by filling in the port information as shown below:

![New Source Wizard - Define Module](image)

**Figure 5: Define Module**

7. Click **Next**, then **Finish** in the New Source Information dialog box to complete the new source file template.
8. Click **Next**, then **Next**, then **Finish**.
The source file containing the counter module displays in the Workspace, and the counter displays in the Sources tab, as shown below:

![New Project in ISE](image)

**Figure 6: New Project in ISE**

Using Language Templates (Verilog)

The next step in creating the new source is to add the behavioral description for counter. Use a simple counter code example from the ISE Language Templates and customize it for the counter design.

1. Place the cursor on the line below the `output [3:0] COUNT_OUT;` statement.
2. Open the Language Templates by selecting **Edit** → **Language Templates**…
   
   **Note:** You can tile the Language Templates and the counter file by selecting **Window** → **Tile Vertically** to make them both visible.
3. Using the “+” symbol, browse to the following code example:

   **Verilog** → **Synthesis Constructs** → **Coding Examples** → **Counters** → **Binary** → **Up/Down Counters** → **Simple Counter**
4. With Simple Counter selected, select **Edit → Use in File**, or select the **Use Template in File** toolbar button. This step copies the template into the counter source file.

5. Close the Language Templates.

**Final Editing of the Verilog Source**

1. To declare and initialize the register that stores the counter value, modify the declaration statement in the first line of the template as follows:

   replace: `reg [<upper>:0] <reg_name>;`

   with: `reg [3:0] count_int = 0;`

2. Customize the template for the counter design by replacing the port and signal name placeholders with the actual ones as follows:

   ♦ replace all occurrences of `<clock>` with `CLOCK`
   ♦ replace all occurrences of `<up_down>` with `DIRECTION`
   ♦ replace all occurrences of `<reg_name>` with `count_int`

3. Add the following line just above the `endmodule` statement to assign the register value to the output port:

   ```
   assign COUNT_OUT = count_int;
   ```

4. Save the file by selecting **File → Save**.

   When you are finished, the code for the counter will look like the following:

   ```
   module counter(CLOCK, DIRECTION, COUNT_OUT);
   input CLOCK;
   input DIRECTION;
   output [3:0] COUNT_OUT;
   );

   reg [3:0] count_int = 0;
   always @(posedge CLOCK)
     if (DIRECTION)
       count_int <= count_int + 1;
   else
     count_int <= count_int - 1;
   assign COUNT_OUT = count_int;
   endmodule
   ```

   You have now created the Verilog source for the tutorial project.

**Checking the Syntax of the New Counter Module**

When the source files are complete, check the syntax of the design to find errors and typos.

1. Verify that **Implementation** is selected from the drop-down list in the Sources window.

2. Select the **counter** design source in the Sources window to display the related processes in the Processes window.
3. Click the “+” next to the Synthesize-XST process to expand the process group.
4. Double-click the Check Syntax process.  
   **Note:** You must correct any errors found in your source files. You can check for errors in the Console tab of the Transcript window. If you continue without valid syntax, you will not be able to simulate or synthesize your design.
5. Close the HDL file.

### Design Simulation

#### Verifying Functionality using Behavioral Simulation

Create a test bench waveform containing input stimulus you can use to verify the functionality of the counter module. The test bench waveform is a graphical view of a test bench.

Create the test bench waveform as follows:

1. Select the **counter** HDL file in the Sources window.
2. Create a new test bench source by selecting **Project** → **New Source**.
3. In the New Source Wizard, select **Test Bench WaveForm** as the source type, and type **counter_tbw** in the File Name field.
4. Click **Next**.
5. The Associated Source page shows that you are associating the test bench waveform with the source file **counter**. Click **Next**.
6. The Summary page shows that the source will be added to the project, and it displays the source directory, type, and name. Click **Finish**.
7. You need to set the clock frequency, setup time and output delay times in the Initialize Timing dialog box before the test bench waveform editing window opens.

   The requirements for this design are the following:
   - The counter must operate correctly with an input clock frequency = 25 MHz.
   - The DIRECTION input will be valid 10 ns before the rising edge of CLOCK.
   - The output (COUNT_OUT) must be valid 10 ns after the rising edge of CLOCK.

   The design requirements correspond with the values below.

   Fill in the fields in the Initialize Timing dialog box with the following information:
   - Clock High Time: **20** ns.
   - Clock Low Time: **20** ns.
   - Input Setup Time: **10** ns.
   - Output Valid Delay: **10** ns.
   - Offset: **0** ns.
   - Global Signals: **GSR (FPGA)**

   **Note:** When GSR(FPGA) is enabled, 100 ns. is added to the Offset value automatically.
   - Initial Length of Test Bench: **1500** ns.
Leave the default values in the remaining fields.

8. Click **Finish** to complete the timing initialization.

9. The blue shaded areas that precede the rising edge of the CLOCK correspond to the Input Setup Time in the Initialize Timing dialog box. Toggle the DIRECTION port to define the input stimulus for the counter design as follows:
   - Click on the blue cell at approximately the 300 ns to assert DIRECTION high so that the counter will count up.
   - Click on the blue cell at approximately the 900 ns to assert DIRECTION low so that the counter will count down.

*Figure 7: Initialize Timing*
Note: For more accurate alignment, you can use the Zoom In and Zoom Out toolbar buttons.

10. Save the waveform.

11. In the Sources window, select the Behavioral Simulation view to see that the test bench waveform file is automatically added to your project.

12. Close the test bench waveform.

Simulating Design Functionality

Verify that the counter design functions as you expect by performing behavior simulation as follows:

1. Verify that Behavioral Simulation and counter_tbw are selected in the Sources window.

2. In the Processes tab, click the “+” to expand the Xilinx ISE Simulator process and double-click the Simulate Behavioral Model process.

   The ISE Simulator opens and runs the simulation to the end of the test bench.

3. To view your simulation results, select the Simulation tab and zoom in on the transitions.
Create Timing Constraints

The simulation waveform results will look like the following:

![Simulation Results](image)

**Figure 10: Simulation Results**

*Note:* You can ignore any rows that start with TX.

4. Verify that the counter is counting up and down as expected.

5. Close the simulation view. If you are prompted with the following message, “You have an active simulation open. Are you sure you want to close it?”, click **Yes** to continue.

You have now completed simulation of your design using the ISE Simulator.

Create Timing Constraints

Specify the timing between the FPGA and its surrounding logic as well as the frequency the design must operate at internal to the FPGA. The timing is specified by entering constraints that guide the placement and routing of the design. It is recommended that you enter global constraints. The clock period constraint specifies the clock frequency at which your design must operate inside the FPGA. The offset constraints specify when to expect valid data at the FPGA inputs and when valid data will be available at the FPGA outputs.
Entering Timing Constraints

To constrain the design do the following:

1. Select **Implementation** from the drop-down list in the Sources window.
2. Select the **counter** HDL source file.
3. Click the “+” sign next to the User Constraints processes group, and double-click the **Create Timing Constraints** process.

ISE runs the Synthesis and Translate steps and automatically creates a User Constraints File (UCF). You will be prompted with the following message:

![Project Navigator](image)

**Figure 11**: Prompt to Add UCF File to Project

4. Click **Yes** to add the UCF file to your project.

   The `counter.ucf` file is added to your project and is visible in the Sources window. The Xilinx Constraints Editor opens automatically.

   **Note**: You can also create a UCF file for your project by selecting **Project → Create New Source**.

5. In the Timing Constraints dialog, enter the following in the Period, Pad to Setup, and Clock to Pad fields:
   - Period: 40
   - Pad to Setup: 10
   - Clock to Pad: 10

6. Press Enter.
After the information has been entered, the dialog should look like what is shown below.

![Creating Timing Constraints](image1.png)

**Figure 12: Creating Timing Constraints**

7. Select **Timing Constraints** under Constraint Type in the Timing Constraints tab and the newly created timing constraints are displayed as follows:

![Timing Constraints](image2.png)

**Figure 13: Timing Constraints**
8. Save the timing constraints. If you are prompted to rerun the TRANSLATE or XST step, click OK to continue.
9. Close the Constraints Editor.

Implement Design and Verify Constraints

Implement the design and verify that it meets the timing constraints specified in the previous section.

Implementing the Design

1. Select the counter source file in the Sources window.
2. Open the Design Summary by double-clicking the View Design Summary process in the Processes tab.
4. Notice that after Implementation is complete, the Implementation processes have a green check mark next to them indicating that they completed successfully without Errors or Warnings.

5. Locate the Performance Summary table near the bottom of the Design Summary.
6. Click the **All Constraints Met** link in the Timing Constraints field to view the Timing Constraints report. Verify that the design meets the specified timing requirements.

![Figure 15: All Constraints Met Report](image)

7. Close the Design Summary.

**Assigning Pin Location Constraints**

Specify the pin locations for the ports of the design so that they are connected correctly on the Spartan-3 Startup Kit demo board.

To constrain the design ports to package pins, do the following:

1. Verify that **counter** is selected in the Sources window.
2. Double-click the **Floorplan Area/IO/Logic - Post Synthesis** process found in the User Constraints process group. The Xilinx Pinout and Area Constraints Editor (PACE) opens.
3. Select the **Package View** tab.
4. In the Design Object List window, enter a pin location for each pin in the **Loc** column using the following information:
   - CLOCK input port connects to FPGA pin **T9** (GCK0 signal on board)
   - COUNT_OUT<0> output port connects to FPGA pin **K12** (LD0 signal on board)
   - COUNT_OUT<1> output port connects to FPGA pin **P14** (LD1 signal on board)
   - COUNT_OUT<2> output port connects to FPGA pin **L12** (LD2 signal on board)
   - COUNT_OUT<3> output port connects to FPGA pin **N14** (LD3 signal on board)
   - DIRECTION input port connects to FPGA pin **K13** (SW7 signal on board)
Notice that the assigned pin locations are shown in blue:

![Figure 16: Package Pin Locations](image)

5. Select File → Save. You are prompted to select the bus delimiter type based on the synthesis tool you are using. Select XST Default <> and click OK.

6. Close PACE.

Notice that the Implement Design processes have an orange question mark next to them, indicating they are out-of-date with one or more of the design files. This is because the UCF file has been modified.

**Reimplement Design and Verify Pin Locations**

Reimplement the design and verify that the ports of the counter design are routed to the package pins specified in the previous section.

First, review the Pinout Report from the previous implementation by doing the following:

1. Open the Design Summary by double-clicking the View Design Summary process in the Processes window.
2. Select the Pinout Report and select the Signal Name column header to sort the signal names. Notice the Pin Numbers assigned to the design ports in the absence of location constraints.

3. Reimplement the design by double-clicking the Implement Design process.

4. Select the Pinout Report again and select the Signal Name column header to sort the signal names.

5. Verify that signals are now being routed to the correct package pins.


**Download Design to the Spartan™-3 Demo Board**

This is the last step in the design verification process. This section provides simple instructions for downloading the counter design to the Spartan-3 Starter Kit demo board.

1. Connect the 5V DC power cable to the power input on the demo board (J4).
2. Connect the download cable between the PC and demo board (J7).
3. Select Implementation from the drop-down list in the Sources window.
4. Select **counter** in the Sources window.
5. In the Process window, double-click the **Configure Target Device** process.
6. The Xilinx WebTalk Dialog box may open during this process. Click **Decline**.

iMPACT opens and the Configure Devices dialog box is displayed.

![iMPACT Welcome Dialog Box](image)

**Figure 19:** iMPACT Welcome Dialog Box

7. In the Welcome dialog box, select **Configure devices using Boundary-Scan (JTAG)**.
8. Verify that **Automatically connect to a cable and identify Boundary-Scan chain** is selected.
9. Click **Finish**.
10. If you get a message saying that there are two devices found, click **OK** to continue.

The devices connected to the JTAG chain on the board will be detected and displayed in the iMPACT window.
11. The **Assign New Configuration File** dialog box appears. To assign a configuration file to the xc3s200 device in the JTAG chain, select the `counter.bit` file and click **Open**.

![Assign New Configuration File](image)

**Figure 20: Assign New Configuration File**

12. If you get a Warning message, click **OK**.

13. Select **Bypass** to skip any remaining devices.

14. Right-click on the xc3s200 device image, and select **Program...** The **Programming Properties** dialog box opens.

15. Click **OK** to program the device.

   When programming is complete, the Program Succeeded message is displayed.

   ![Program Succeeded](image)

   **Program Succeeded**

On the board, LEDs 0, 1, 2, and 3 are lit, indicating that the counter is running.


You have completed the ISE Quick Start Tutorial. For an in-depth explanation of the ISE design tools, see the ISE In-Depth Tutorial on the Xilinx® web site at: [http://www.xilinx.com/support/techsup/tutorials/](http://www.xilinx.com/support/techsup/tutorials/)