ISE Simulator (ISim)

In-Depth Tutorial

UG682 (v1.0) April 27, 2009

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/27/09	1.0	Initial Xilinx release.

Table of Contents

Chapter 4: Using ISE Simulator (ISim) Graphical User Interface

About This Tutorial

About the ISE Simulator (ISim) In-Depth Tutorial

The *ISim In-Depth Tutorial* provides Xilinx PLD designers with a detailed introduction of the ISE Simulator (ISim) software. After you have completed the tutorial, you will have a thorough understanding of how to analyze and debug your design via HDL simulation using ISim.

Note: This tutorial is designed for running the ISim software on a Windows environment. Some modifications may be required to run certain steps successfully in other operating systems.

Tutorial Contents

This tutorial covers the following topics:

Chapter 1, "Overview of the ISE Simulator (ISim)," introduces the ISim software environment, including the ISim compilers, linker, simulation executable and Graphical User Interface.

Chapter 2, "Using ISE Simulator from ISE Project Navigator," explains how to launch a functional simulation through the ISE Project Navigator software.

Chapter 3, "Running ISE Simulator (ISim) Standalone," guides you through a typical procedure for launching a functional simulation using the ISim compiler, linker and simulation executable outside of the ISE Project Navigator environment.

Chapter 4, "Using ISE Simulator (ISim) Graphical User Interface," introduces you to the ISim GUI by examining, debugging, and verifying a functional simulation.

Tutorial Flows

This tutorial presents two flows in which ISim can be used for performing a functional (Behavioral) simulation.

- Using ISim from ISE® Project Navigator
- Using ISim Standalone

Using ISim from ISE Project Navigator

In this flow you will launch ISim via one of the simulation processes available in the ISE Project Navigator. This flow works best when an ISE Project Navigator project is created in order to implement the design in a Xilinx® FPGA or CPLD. This flow is useful when your design involves sources that are not HDL (schematics, cores, etc.) and requires Project Navigator to properly convert these sources to HDL source files which ISim can compile.

Follow these chapters if you are interested in this flow:

- Chapter 1, "Overview of the ISE Simulator (ISim),"
- Chapter 2, "Using ISE Simulator from ISE Project Navigator,"
- Chapter 4, "Using ISE Simulator (ISim) Graphical User Interface,"

Using ISim Standalone

In this mode you will primarily simulate your design by creating your own ISim project files and running the HDL linker and simulation executable in a command line or batch file mode. This flow is useful for users not using Project Navigator to manage their HDL design.

The following chapters will help you understand this flow:

- Chapter 1, "Overview of the ISE Simulator (ISim),"
- Chapter 3, "Running ISE Simulator (ISim) Standalone,"
- Chapter 4, "Using ISE Simulator (ISim) Graphical User Interface,"

Additional Resources

To find more detailed information and discussions on ISE Simulator (ISim) topics covered in this tutorial, refer to the following documents:

- ISim Help is available from the ISim software.
- *ISim User Guide*, accessible from the Software Manuals page on the Xilinx website:

http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/plugin_ism.pdf

- Software Manuals:

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/literature

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:

http://www.xilinx.com/support

To discuss topics of interest with other Xilinx users, see the Xilinx User Community Forum at:

http://forums.xilinx.com/xlnx/

Chapter 1

Overview of the ISE Simulator (ISim)

Overview of ISim

The Xilinx® ISE Simulator (ISim) is a Hardware Description Language (HDL) simulator that enables you to perform functional and timing simulations for VHDL, Verilog and mixed language designs.

This ISE Simulator environment is comprised of the following key elements:

- Vhpcomp (VHDL compiler)
- Vlogcomp (Verilog compiler)
- fuse (HDL elaborator and linker)
- Simulation Executable
- isimgui (ISim Graphical User Interface)

vhpcomp, vlogcomp

vhpcomp and vlogcomp parse and compile VHDL and Verilog source files respectively. The object code generated by the compilers is used by HDL linker (fuse) to create a simulation executable.

fuse

The fuse command is the Hardware Description Language (HDL) elaborator and linker used by ISim. fuse effects static elaboration on the design given the top design units and then compiles the design units to object code. The design unit object files are then linked together to create a simulation executable.

fuse can link design units compiled previously with vhpcomp or vlogcomp. Alternatively, fuse can automatically invoke vlogcomp and vhpcomp for each VHDL or Verilog source code listed in a project file (.prj). This method allows for compilation of sources “on-the-fly”.

Simulation Executable

The Simulation Executable is generated by the fuse command. To run the simulation of a design in ISim, the generated simulation executable needs to be invoked. When ISim is run inside the ISE Project Navigator interface, ISE takes care of invoking the generated simulation executable. A command-line user needs to explicitly invoke the generated simulation executable to effect simulation. The simulation executable effects event-driven simulation and has rich support for driving and probing simulation using Tcl.

Note: The ISE Simulation Executable has a .exe extension in both Linux and Windows. The default executable naming format is **x.exe**.

isimgui.exe

isimgui.exe (isimgui on Linux) is the ISim Graphical User Interface. It contains the wave window, toolbars, panels, and the status bar. In the main window, you can view the simulation-visible parts of the design, add and view signals in the wave window, utilize ISim commands to run simulation, examine the design, and debug as necessary.

Using ISE Simulator from ISE Project Navigator

Overview of ISim ISE Integrated Flow

The Xilinx® ISE software provides an integrated flow with the Xilinx ISE Simulator (ISim) that allows simulations to be launched directly from the Xilinx Project Navigator (ISE). All simulation commands that prepare the ISim simulation are generated by ISE Project Navigator and automatically run in the background when simulating a design using this flow.

Getting Started

Software Requirements

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

1. ISE WebPACK™ 11, or
2. One of the ISE Design Suite 11 Editions (Logic, DSP, Embedded, System)

For more information about installing Xilinx software, see the *ISE Release Notes and Installation Guide* at: http://www.xilinx.com/support/software_manuals.htm

Installing the Tutorial Design Files

Design files for this tutorial can be downloaded from:

http://www.xilinx.com/support/techsup/tutorials/tutorials11.htm

After you have downloaded the tutorial project files from the Web, unzip them into an easily accessible directory with full read and write permissions.

The contents of the tutorial project files are as follows:

- **sources**: Folder containing all the HDL files necessary for a functional simulation of the design.
- **scripts**: Folder containing incomplete script files to run the simulation. These script files will be completed as you go through the tutorial.
- **completed**: Folder containing completed script, simulation and wave configuration files, as well as a completed ISE 11 project of the tutorial design, for comparison purposes.

Design Description

The ISim In-Depth Tutorial provides a design which the reader can use to become familiar performing some basic simulation steps while using the ISim software.

The tutorial design is a simple demonstration of the Dynamic Reconfiguration feature of the Virtex®-5 Digital Clock Manager (DCM).

Using the Virtex-5 DCM, the design generates an output clock using the following relationship:

Output Clock = Input Clock * (Multiplier / Divider)

Using the Dynamic Reconfiguration Ports (DRP) in the DCM, the design allows the user to re-define the Multiplier and Divider parameters to generate different output frequencies.

Functional Blocks

The tutorial design consists of the following functional blocks.

- **drp_dcm (drp_dcm.vhd)**

  Virtex-5 DCM macro with internal feedback, frequency controlled output, duty-cycle correction, and Dynamic Reconfiguration ability.

  The CLKFX_OUT output provides a clock that is defined by the following relationship:

  CLKFX_OUT = CLKIN_IN * (Multiplier / Divider)

  For example, using a 100 MHz input clock, setting the Multiplier factor to 6, and Divider factor to 5, produces a 120 MHz CLKFX_OUT output clock.

  Using the DRP ports of the DCM, the Multiplier (M) and Divider (D) parameters can be dynamically redefined to produce different CLKFX_OUT frequencies. For the purposes of this tutorial, it suffices to show how the Multiply and Divide parameters are provided to the DCM via the 16-bit wide DI_IN port:

  DI_IN[15:8] = M – 1

  DI_IN[7:0] = D – 1

  For example, for an M / D factor of 6 / 5, DI_IN = 0504h.

- **drp_stmach (drp_stmach.vhd)**

  This module describes a Dynamic Reconfiguration Controller. The DRP controller asserts and monitors the DCM DRP signals in order to perform a dynamic reconfiguration cycle.

  A dynamic reconfiguration cycle is started by asserting the **drp_start** signal. Following this step, the DRP Controller asserts the appropriate DCM DRP pins in order to complete a full Dynamic Reconfiguration cycle.

  Signal **drp_done** indicates a successful completion of a dynamic reconfiguration cycle.

- **drp_demo (drp_demo.vhd)**

  This is the top module of the tutorial design which connects the DCM macro and the DRP controller modules to the external I / O ports.

- **drp_demo_tb (drp_demo_tb.vhd)**

  Self-checking HDL test bench. Refer to Design Self-Checking Test Bench for more information.

Design Self-Checking Test Bench

To test the functionality of this design, a self-checking test bench has been provided. (Refer to source file drp_demo_tb.vhd in the sources folder.) A self-checking test bench contains a validation routine or function that compares sampled values from the simulation against expected results. The self-checking test bench provided for this design performs the following functions.

- Generates a 100 MHz input clock for the design system clock (clk_in).
- Performs four different tests in order to dynamically change the output frequency of the design. In each test, a DRP cycle is started (using the **drp_start** signal) to set the output clock to a different frequency. The following table shows the desired output frequency and Multiplier/Divider parameters used for each test.

Table 2-1: **Desired Output Frequency and Multiplier/Divider Parameters Used For Each Test**

Test	Freq. (MHz)	Period (ps)	Multiplier (M)	Divider (D)
1	75	13,332	3	4
2	120	8,332	6	5
3	250	4000	5	2
4	400	2,500	4	1

- In each test, the test bench will compare the expected clock period and the clock period measured during simulation. Based on the comparison results, messages to the simulator will be written indicating success or failure.
- Upon completion of the simulation, a summary report is provided, listing which tests passed or failed.

Note: For more details on the functionality of this design, refer to the in-line comments included in the sources of the design.

Simulating the Design

Thanks to an intuitive integrated flow, you can easily and quickly perform behavioral and timing simulations of your design in the ISE Project Navigator software. Using the integrated flow, you can quickly set up simulation properties and launch the ISim software with a few clicks of the mouse.

We shall demonstrate how ISim can be launched using the ISE Project Navigator by first creating an ISE project for the tutorial design. We will then set some behavioral simulation properties and launch the ISim simulator to perform a behavioral simulation of the design.

Creating a Project in ISE Project Navigator

We will use the New Project Wizard in ISE Project Navigator to quickly create an ISE project for the tutorial design.

Note: Read Installing the Tutorial Design Files to obtain the files required for this design.

Using New Project Wizard

Follow these steps to create an ISE project using the New Project Wizard.

1. Launch the ISE Project Navigator by double-clicking on the Xilinx ISE 11 desktop icon.

Figure 2-1: **Xilinx ISE 11**

2. Click the **New Project** button to launch the New Project Wizard.
3. Provide a name and an appropriate location for the project (Refer to Figure 2-2).
4. Click **Next** to continue.

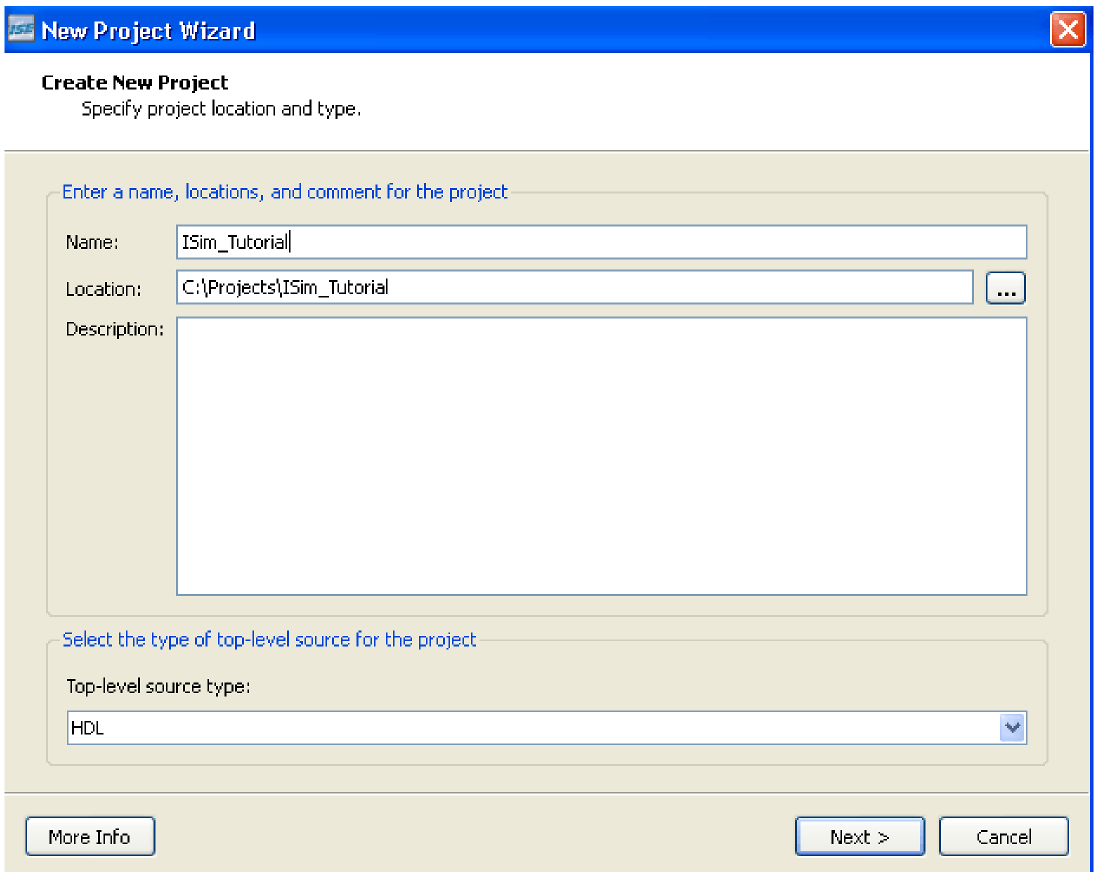


Figure 2-2: **New Project Wizard**

5. In the window, select the device and project properties.
6. Change the settings to match the settings shown inFigure 2-3.
7. Click **Next** to continue.

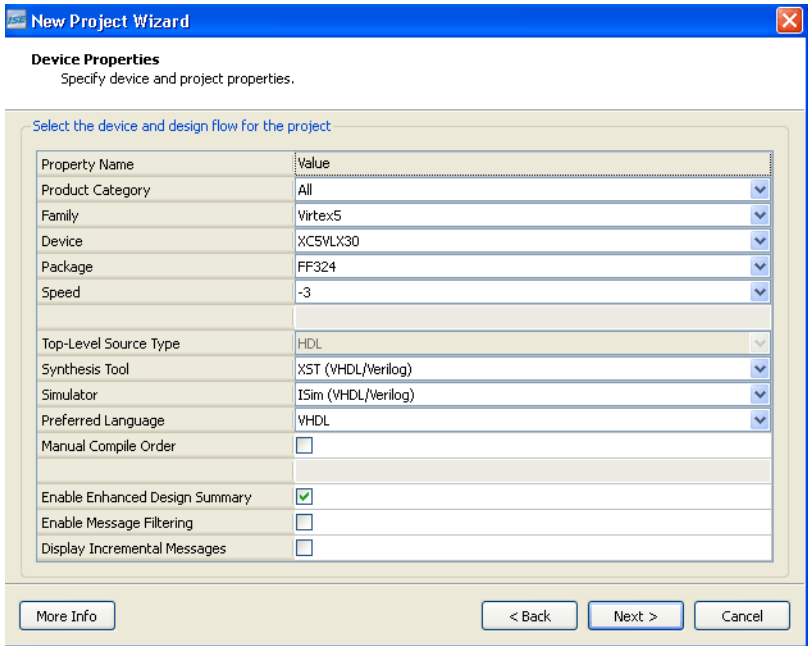


Figure 2-3: **Change the Settings**

8. Click **Next** in the next window. No new sources will be created for the tutorial design.

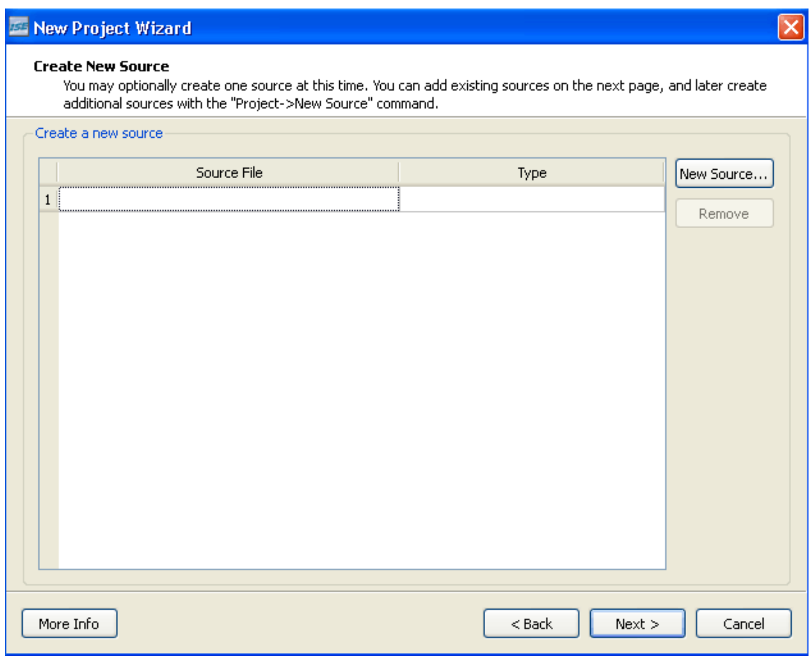


Figure 2-4: **Add Sources**

9. In the next window, point to the sources for the tutorial design. Click the **Add Source** button to select the sources provided for the tutorial design.

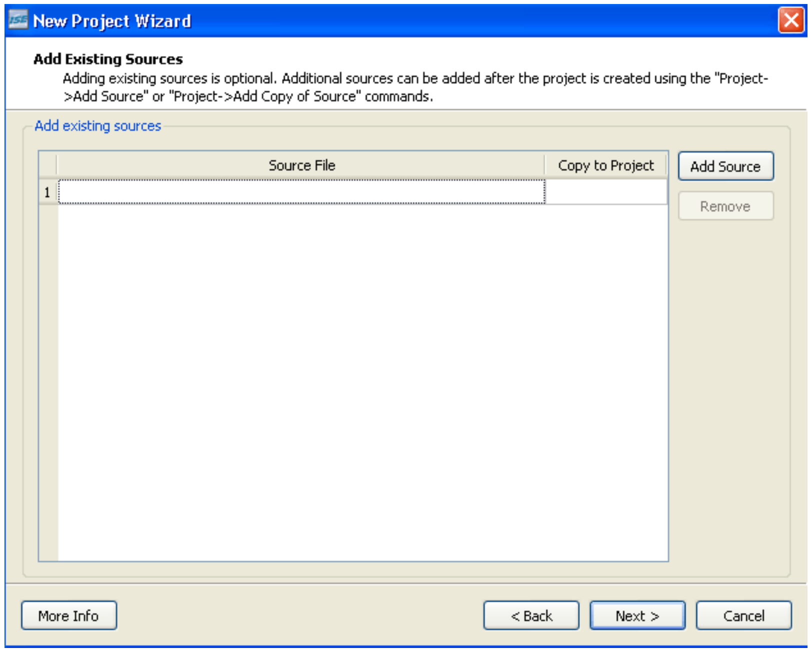


Figure 2-5: **Add Source Button**

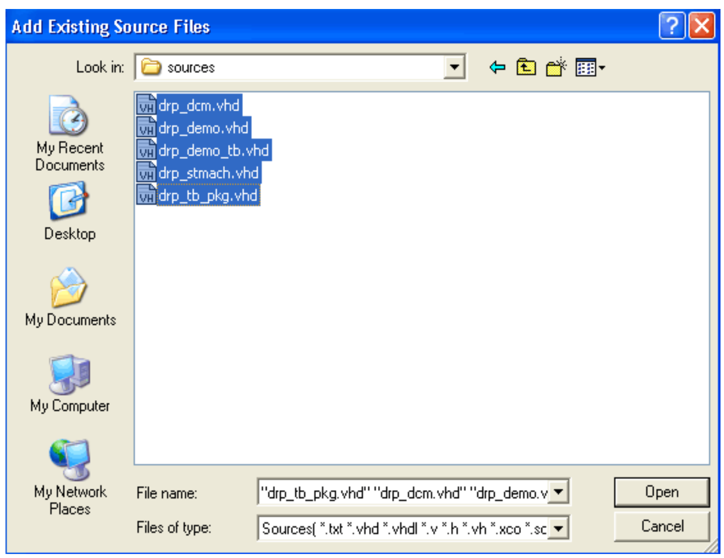


Figure 2-6: **Add Existing Source Files**

10. Remove the check boxes under the column Copy to Project so the source files are not copied into the project directory.
11. Click **Next** to continue.

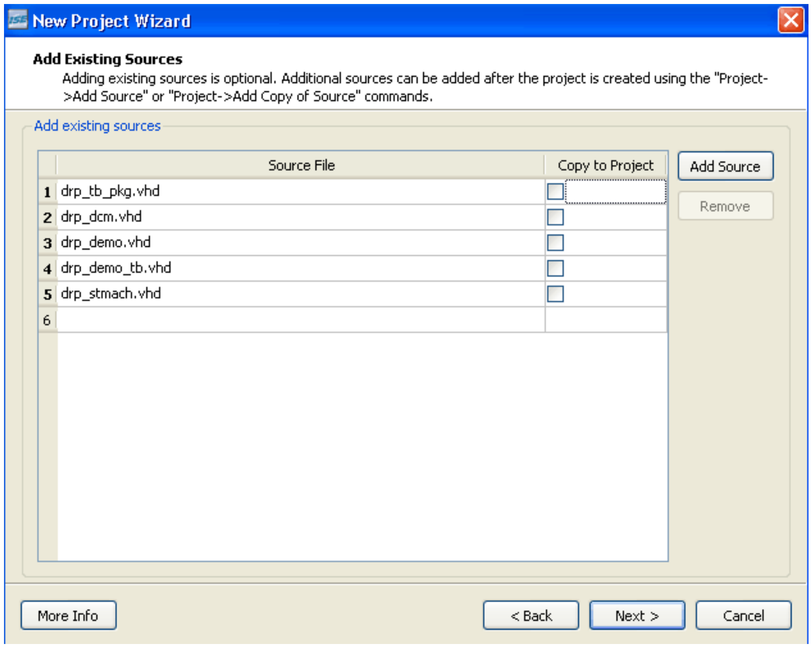


Figure 2-7: **Remove the Check Boxes**

12. Review the Project Summary page and make sure that the settings match those shown in Figure 2-8.
13. Click **Next** to continue.

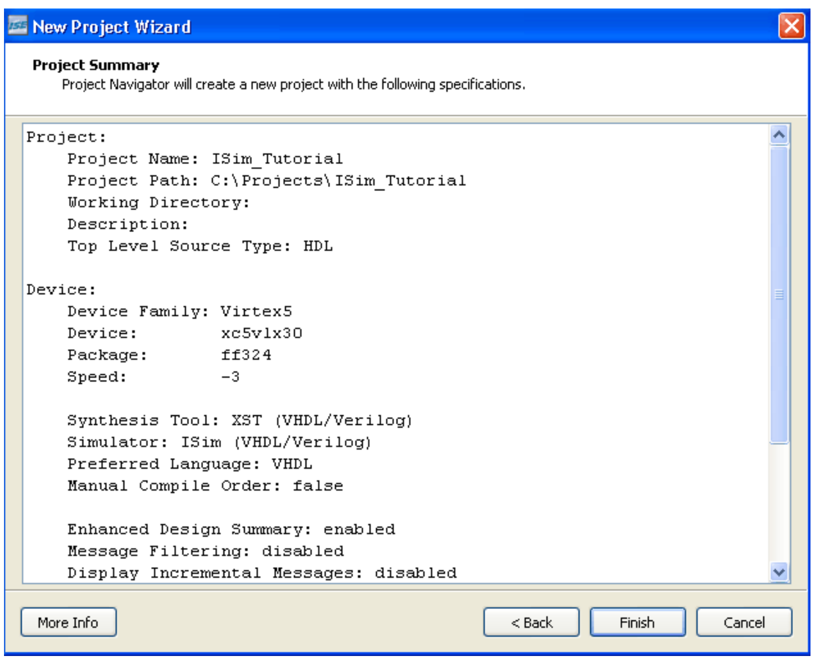


Figure 2-8: **Project Summary**

14. In the next window, make sure that the association and libraries have been properly specified for the tutorial sources. Compare your settings with the settings shown inFigure 2-9.

15. Click **OK** to finalize the New Project Wizard and start using ISE with the tutorial design files.

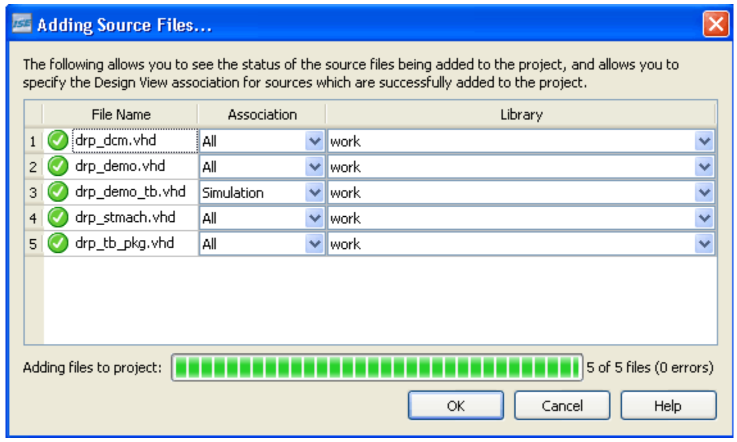


Figure 2-9: **Status of Source Files and Associations**

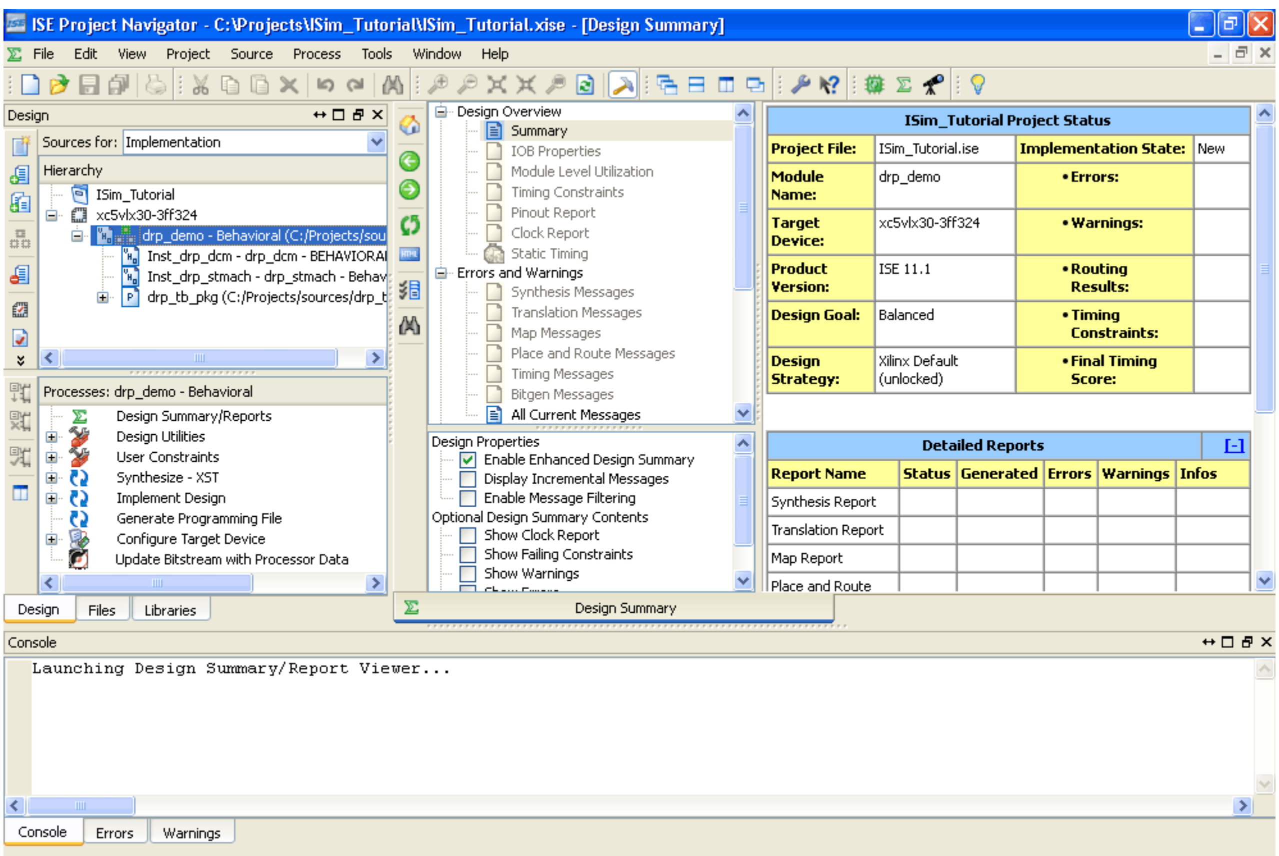


Figure 2-10: **ISE Project Navigator Design Summary**

Next, you need to create a user VHDL library for a VHDL package (drp_tb_pkg.vhd) used by the test bench of this design. The VHDL package contains VHDL functions used by the test bench to perform verification routines. Once the VHDL library is created, move the VHDL package file from the work library to the newly-created VHDL library.

Creating VHDL Library

Follow these steps to create a VHDL library.

1. In Project Navigator, select **Project** > **New Source**. The New Source Wizard opens.
2. Select **VHDL Library** as a source type.
3. Type drp_tb_lib for the VHDL library name. (Refer to Figure 2-11).
4. Click **Next** to continue.

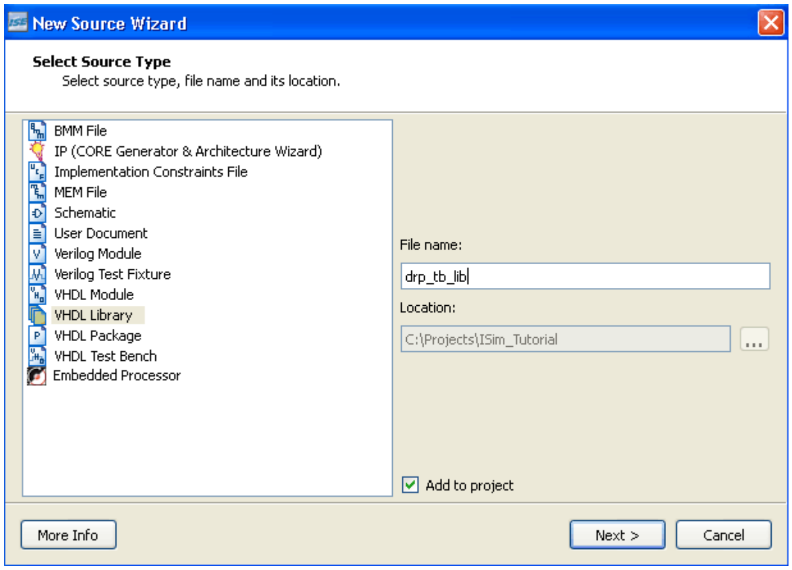


Figure 2-11: **Select Source Type**

5. Click **Finish** to complete the New Source Wizard.

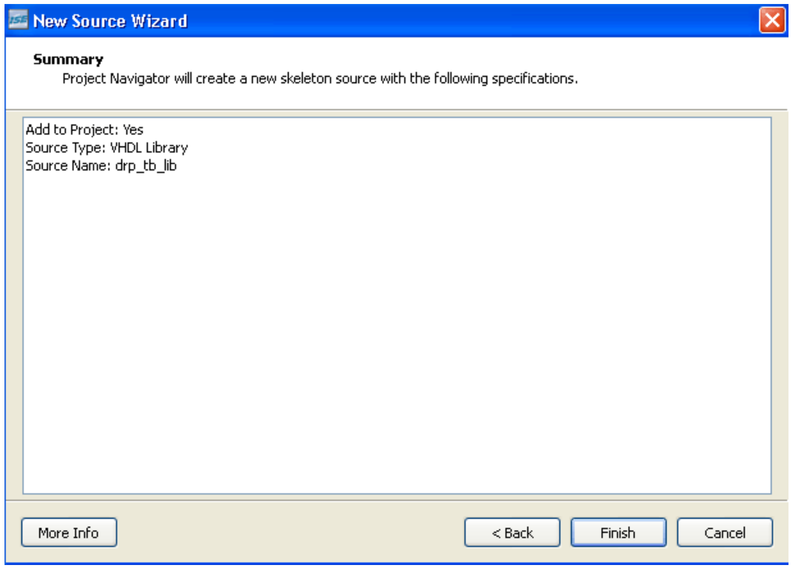


Figure 2-12: **New Source Wizard**

Moving VHDL files to a Library

Follow these steps to move the VHDL package file to the drp_tb_lib VHDL library.

1. In the Sources Pane, select the Libraries tab to switch to the Libraries Pane. (Refer to Figure 2-13.)

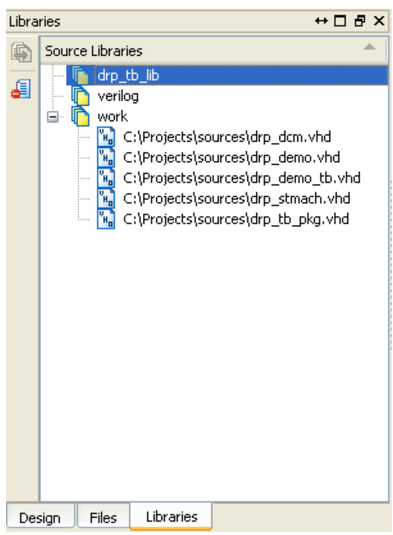


Figure 2-13: **Select the Libraries Tab**

2. Expand the work library by clicking once on the hierarchy separator (Refer to Figure 2-13.)
3. Right-click on the VHDL file drp_tb_pkg.vhd, and select **Move to Library**.
4. In the Move to Library dialog box, select drp_tb_lib as the library into which you will move the VHDL package drp_tb_pkg.vhd file.
5. Click **OK**. (Refer to Figure 2-14.)

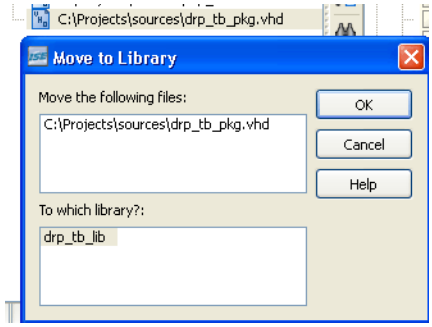


Figure 2-14: **Move to Library Window**

You can now observe that a new VHDL library, drp_tb_lib, contains a VHDL package file, drp_tb_pkg.vhd. (Refer to Figure 2-15.)

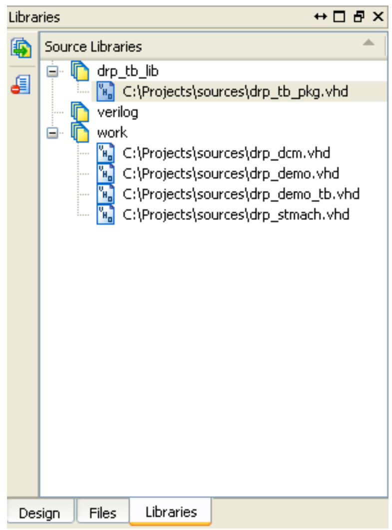


Figure 2-15: **Source Libraries**

Launching a Behavioral Simulation

Now that the ISE project has been created for the tutorial design, we can proceed to setup and launch a behavioral simulation using ISim.

Setting Behavioral Simulation Properties

Follow these steps to set behavioral simulation properties in ISE:

1. In the Sources for drop-down menu, select **Behavioral Simulation**. Highlight the tutorial design test bench, drp_demo_tb by clicking the file name. You should now see the simulation processes available for the design in the Processes pane. (Refer to Figure 2-16)

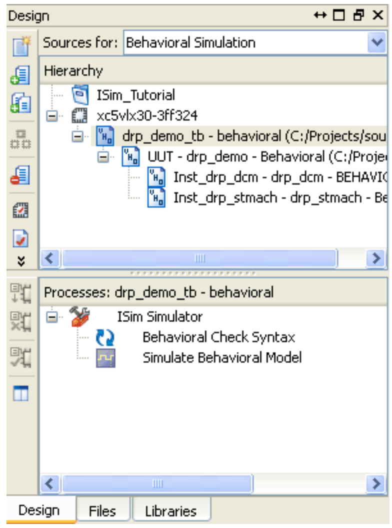


Figure 2-16: **Process Pane**

2. Right-click on Simulate Behavioral Model under the ISim Simulator process and click on **Properties**. The ISim Properties window comes up (Refer toFigure 2-17).

In this window you can set different simulation properties such as simulation runtime, waveform database file location, and whether you would like to use a customer simulation command file to launch the simulation.

3. For the purposes of this tutorial, we will disable the feature that runs the simulation for a specified amount of time right from the start of the simulation. Uncheck the property Run for Specified Time, and click OK (Refer toFigure 2-17.)

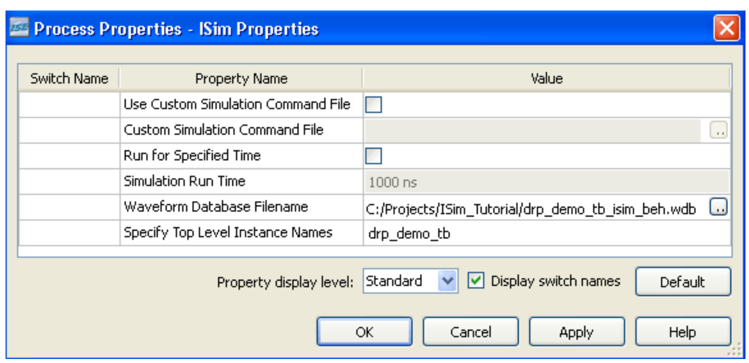


Figure 2-17: **ISim Properties Window**

Launching Behavioral Simulation

You are now ready to launch the ISE Simulator to perform a behavioral simulation of the tutorial design. To launch the simulator, double-click on Simulate Behavioral Model. The ISim Graphical User Interface (GUI) (Figure 2-18) will appear shortly after the design is successfully parsed and compiled.

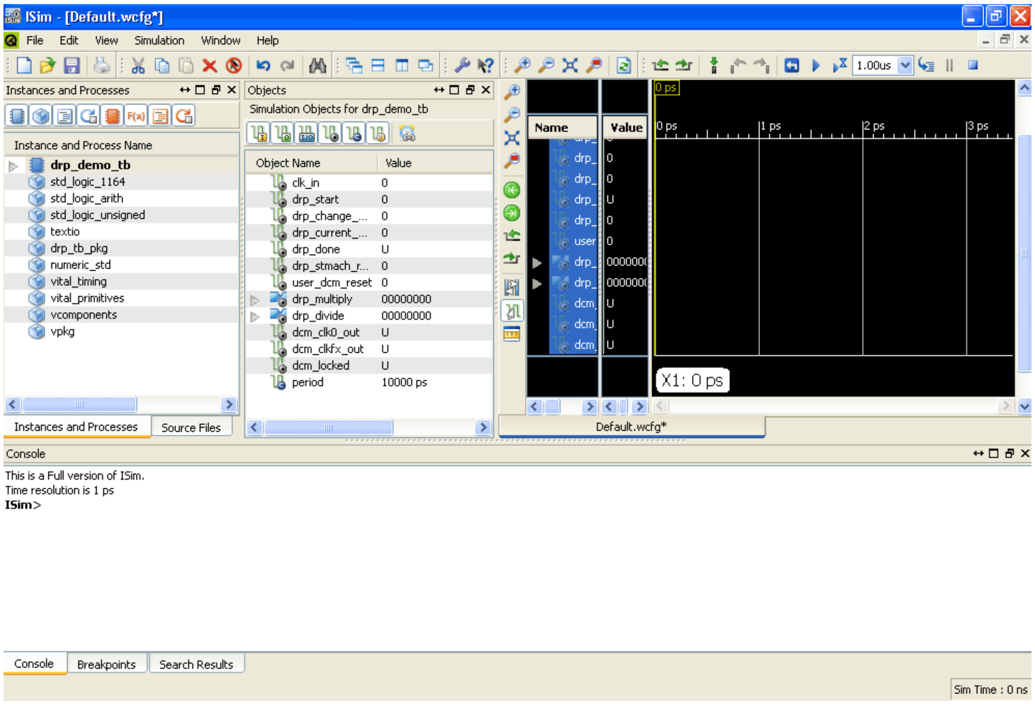


Figure 2-18: **ISim GUI**

What's Next?

Continue on to Chapter 4, "Using ISE Simulator (ISim) Graphical User Interface" to learn more about the ISim GUI features, and tools for analyzing and debugging HDL designs.

Running ISE Simulator (ISim) Standalone

Overview of ISim Standalone Flow

ISim offers a standalone flow which you can use to simulate your design without setting up a project in ISE® Project Navigator. In this flow, you:

1. **Prepare the simulation project** by manually creating an ISim project file in order to create a simulation executable using fuse.
2. **Start the ISim Graphical User Interface** by running the simulation executable generated by **fuse**.

Getting Started

Software Requirements

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

- ISE WebPACK™ 11, or
- One of the ISE Design Suite 11 Editions (Logic, DSP, Embedded, System)

For more information about installing Xilinx® software, see the *ISE Design Suite 11: Installation, Licensing, and Release Notes*:

http://www.xilinx.com/support/sw_manuals/xilinx11/irn.pdf

Installing the Tutorial Design Files

Design files for this tutorial can be downloaded from:

http://www.xilinx.com/support/techsup/tutorials/tutorials11.htm

After you have downloaded the tutorial project files from the Web, unzip them into an easily accessible directory with full read and write permissions.

The contents of the tutorial project files are as follows:

- **sources**: Folder containing all the HDL files necessary for a functional simulation of the design.
- **scripts**: Folder containing incomplete script files to run the simulation. These script files will be completed as you go through the tutorial.

- **completed**: Folder containing completed script, simulation and wave configuration files, as well as a completed ISE 11 project of the tutorial design, for comparison purposes.

Design Description

This tutorial provides a design which the reader can use to become familiar with performing some basic simulation steps while using the ISim software.

The tutorial design is a simple demonstration of the Dynamic Reconfiguration feature of the Virtex®-5 Digital Clock Manager (DCM).

Using the Virtex-5 DCM, the design generates an output clock using the following relationship:

Output Clock = Input Clock * (Multiplier / Divider)

Using the Dynamic Reconfiguration Ports (DRP) in the DCM, the design allows the user to re-define the Multiplier and Divider parameters to generate different output frequencies.

Functional Blocks

The tutorial design consists of the following functional blocks.:

- **drp_dcm (drp_dcm.vhd)**

  Virtex-5 DCM macro with internal feedback, frequency controlled output, duty-cycle correction, and Dynamic Reconfiguration ability.

  The CLKFX_OUT output provides a clock that is defined by the following relationship:

  CLKFX_OUT = CLKIN_IN * (Multiplier / Divider)

  For example, using a 100 MHz input clock, setting the Multiplier factor to 6, and Divider factor to 5, produces a 120 MHz CLKFX_OUT output clock.

  Using the DRP ports of the DCM, the Multiplier (M) and Divider (D) parameters can be dynamically redefined to produce different CLKFX_OUT frequencies. For the purposes of this tutorial, it suffices to show how the Multiply and Divide parameters are provided to the DCM via the 16-bit wide DI_IN port:

  DI_IN[15:8] = M – 1

  DI_IN[7:0] = D – 1

  For example, for an M / D factor of 6 / 5, DI_IN = 0504h.

- **drp_stmach (drp_stmach.vhd)**

  This module describes a Dynamic Reconfiguration Controller. The DRP controller asserts and monitors the DCM DRP signals in order to perform a dynamic reconfiguration cycle.

  A dynamic reconfiguration cycle is started by asserting the **drp_start** signal. Following this step, the DRP Controller asserts the appropriate DCM DRP pins in order to complete a full Dynamic Reconfiguration cycle.

  Signal **drp_done** indicates a successful completion of a dynamic reconfiguration cycle.

- **drp_demo (drp_demo.vhd)**

This is the top module of the tutorial design which connects the DCM macro and the DRP controller modules to the external I/O ports.

- **drp_demo_tb (drp_demo_tb.vhd)**

  Self-checking HDL test bench. Refer to "Design Self-Checking Test Bench" for more information

Design Self-Check Test Bench

To test the functionality of this design, a self-checking test bench has been provided. (Refer to source file drp_demo_tb.vhd in the sources folder.) A self-checking test bench contains a validation routine or function that compares sampled values from the simulation against expected results.

The self-checking test bench provided for this design performs the following functions:

- Generates a 100 MHz input clock for the design system clock (clk_in).
- Performs four different tests in order to dynamically change the output frequency of the design. In each test, a DRP cycle is started (using the **drp_start** signal) to set the output clock to a different frequency. The following table shows the desired output frequency and Multiplier/Divider parameters used for each test:

Table 3-1: **Desired Output Frequency and Multiplier/Divider Parameters Used For Each Test**

Test	Freq. (MHz)	Period (ps)	Multiplier (M)	Divider (D)
1	75	13,332	3	4
2	120	8,332	6	5
3	250	4000	5	2
4	400	2,500	4	1

- In each test, the test bench will compare the expected clock period and the clock period measured during simulation. Based on the comparison results, messages to the simulator will be written indicating success or failure.
- Upon completion of the simulation, a summary report is provided, listing which tests passed or failed.

Note: For more details on the functionality of this design, refer to the in-line comments included in the sources of the design.

Preparing the Simulation

ISim offers a standalone flow which you can use to simulate your design without setting up a project in ISE Project Navigator. In contrast to the ISE Integrated Flow, you will manually create an ISim project file which **fuse** will use to create a simulation executable. Following completion of this step, the ISim Graphical User Interface (GUI) can be launched by running the simulation executable.

Creating an ISim Project File

The typical syntax for an ISim project file is as follows:

```
verilog|vhdl <library_name> {<file_name_1>.v|.vhd}
```

where:

- ***verilog|vhdl*** indicates that the source is a Verilog or VHDL file. Include either verilog or vhdl.
- ***<library_name>*** indicates the library that a particular source on the given line should be compiled. `work` is the default library.
- ***<file_name>*** is the source file or files associated with the library.

Note: While more than one Verilog source file can be specified on a given line, only one VHDL source can be specified on a given line.

Complete the following steps to build an ISim project file for the tutorial design:

1. Browse to the folder "**scripts**" from the downloaded files. Open the project file **simulate_isim.prj** with a text editor.
2. The project file is incomplete. List the missing sources using the syntax guidelines shown above.

   Missing sources:

   - **drp_dcm.vhd**: VHDL source file. It should be compiled to 'work' library.
   - **drp_tb_pkg.vhd**: VHDL package file. It should be compiled to 'drp_tb_lib' library.
3. Save and close the file.

Note: You need not list the sources based on their order of dependency. **fuse** automatically resolves the order of dependencies, and processes the files in the appropriate order.

Note: You can browse to the "**completed**" folder of the tutorial files for a completed version of the project file, for comparison purposes.

Building the Simulation Executable

In this simulation step, **fuse** will use the project file created in the previous section to parse, compile and link all the sources for the design. Following completion of these steps, a simulation executable will be created which will allow you to run the simulation in the ISim GUI.

Using fuse

The typical fuse syntax is as follows:

```
fuse -incremental -prj <project file> -o <simulation executable>
<library.top_unit>
```

where:

- ***-incremental***: requests fuse to compile only the files that have changed since the last compile
- ***-prj***: specifies an ISim project file to use for input
- ***-o***: specifies the name of the simulation executable output file
- ***<library.top_unit>***: specifies the top design unit

Complete the following steps to parse, compile and elaborate the tutorial design using **fuse**:

1. Browse to the folder **scripts** from the downloaded files.

2. Open the batch file **fuse_batch.bat** using a text editor.
3. This fuse command is incomplete. Using the syntax information provided above, edit the command line so it includes the following options:
   a. Use **incremental compilation**
   b. Use **simulate_isim.prj** as the project file
   c. Use **simulate_isim.exe** as the simulation executable
   d. Use **work.drp_demo_tb** as the top design unit for simulation.
4. Save and close the batch file.
5. Double-click on the **fuse_batch.bat** file to run **fuse**.

Once **fuse** completes compiling source code, elaborating design units, and linking the object code, a simulation executable (**simulate_isim.exe**) should be present in the **scripts** folder.

Note: You can browse to the "**completed**" folder for a completed version of the fuse batch file, for comparison purposes.

Simulating the Design

In this simulation step you will launch the ISim Graphical User Interface by running the simulation executable which was generated by the **fuse** tool in the previous section, "Building the Simulation Executable". After this step is complete, you will be able to use the ISim GUI to explore the design in more detail.

Running the Simulation Executable

The typical syntax used when launching the simulation executable is as follows:

```
Simulation_executable -gui -wcfg <wave_configuration_file> -wdb
<waveform_database_file>
```

where:

- **-gui**: launches ISim in Graphical User Interface mode
- **-wcfg**: specifies the Wave Configuration file for setting up the waveform
- **-wdb**: specifies the file name of the simulation database output file. Default simulation executable name is "x.exe"

Complete the following steps to launch the simulation:

1. Browse to the folder **scripts** from the downloaded files.
2. Open the batch file **simulate_isim.bat** using a text editor. The batch file is intentionally blank.
3. Using the syntax information provided above, edit the batch file so it includes the following settings:
   a. Simulation Executable name: **simulate_isim.exe**
   b. Launch in **GUI mode**
   c. Set simulation database output name to **simulate_isim.wdb**

   Note: A wave configuration file is not provided in the tutorial files. This file will be created
4. Save and close the file.
5. Double-click on the **simulate_isim.bat** file to run the simulator.

The ISim GUI will now open and load the design. The simulator time will remain at 0 ns until you specify a run time.

Note: You can browse to the **completed** folder for a completed version of the **simulate_isim.bat** batch file, for comparison purposes.

Using ISE Simulator (ISim) Graphical User Interface

Overview of ISim Graphical User Interface

The ISim Graphical User Interface (GUI) contains the wave window, toolbars, panels, and the status bar. In the main window, you can view the simulation-visible parts of the design, add and view signals in the wave window, utilize ISim commands to run simulation, examine the design, and debug as necessary.

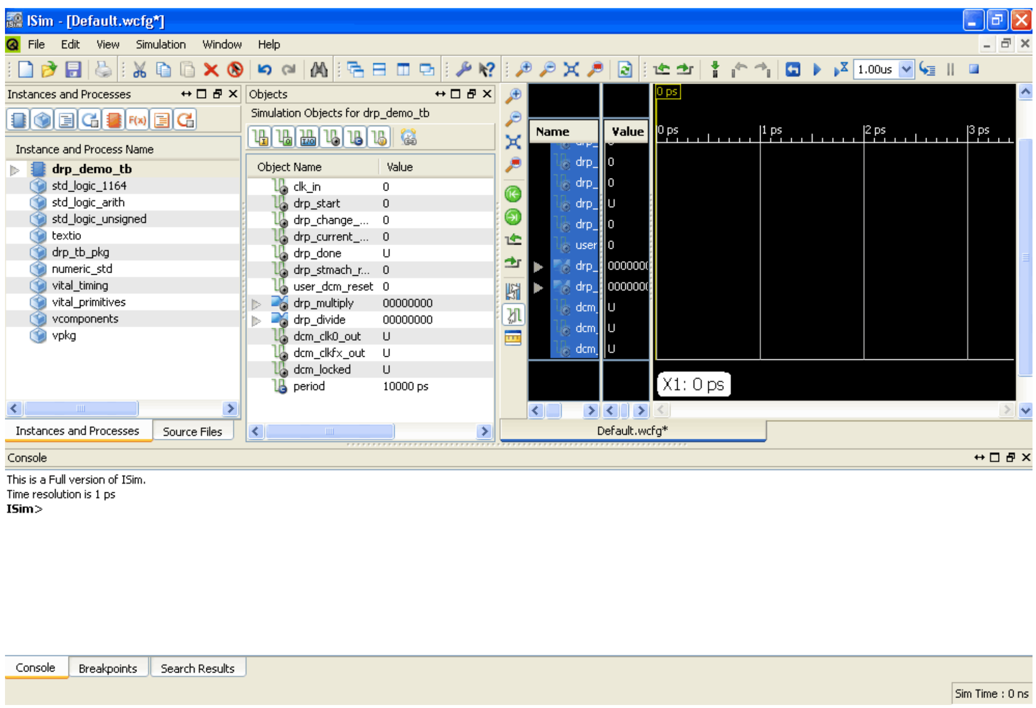


Figure 4-1: **ISim GUI**

Exploring the User Interface

Main Toolbar

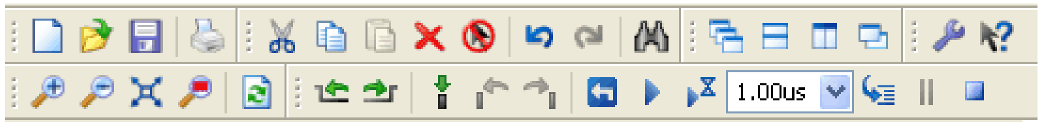


Figure 4-2: **Main Toolbar**

The toolbars available in the ISim main window consists of many functionally different toolbars. Each of these toolbars offers access to frequently used commands:

- File and Edit menu commands
- Window and View menu commands
- Simulation menu commands

The main window toolbar icons are located near the top of the user interface.

Instances and Processes Panel

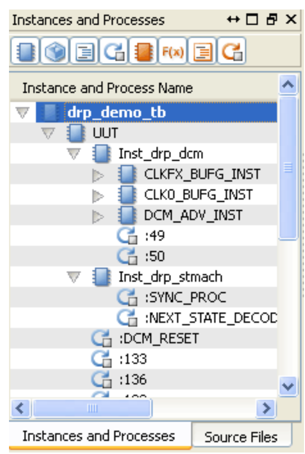


Figure 4-3: **Instances and Processes Panel**

The Instances and Processes panel displays the block (instance and process) hierarchy associated with the wave configuration open in the Wave window. Instantiated and elaborated entities / modules are displayed in a tree structure, with entity components being ports, signals and other entities / modules.

Source Files Panel

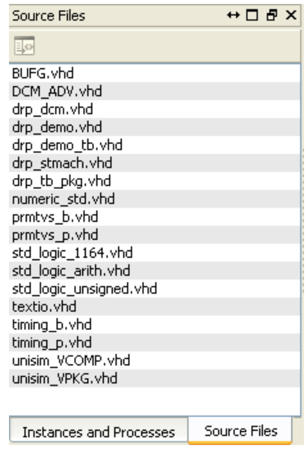


Figure 4-4: **Sources Files Panel**

The Source Files panel displays the list of all the files associated with the design. The list of files is provided by the **fuse** command during design parsing and elaboration, which is run in the background for GUI users. The HDL source files are available for quick access to the read-only source code.

Objects Panel

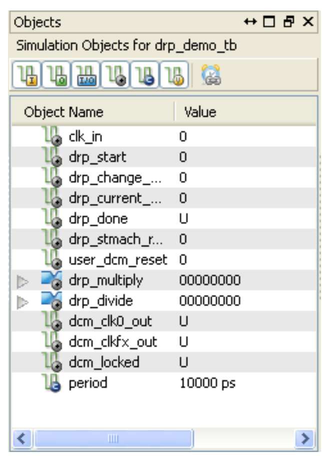


Figure 4-5: **Objects Panel**

The Objects panel displays all ports and signals associated with the selected instances and processes in the Instances and Processes panel.

At the top of the panel, the Simulation Objects displays which instance / process is selected in the Instances and Processes panel whose objects and their values are listed in the Objects panel.

The table columns are defined as follows:

- **Object Name -** Displays the name of the signal, accompanied by the symbol which represents the type of object it is.

- **Value -** The value of the signals at the current simulation time or at the main cursor, as determined by the Sync Time toolbar icon.
- **Data Type -** Displays the data type of the corresponding simulation object, logic or an array.

Wave Window

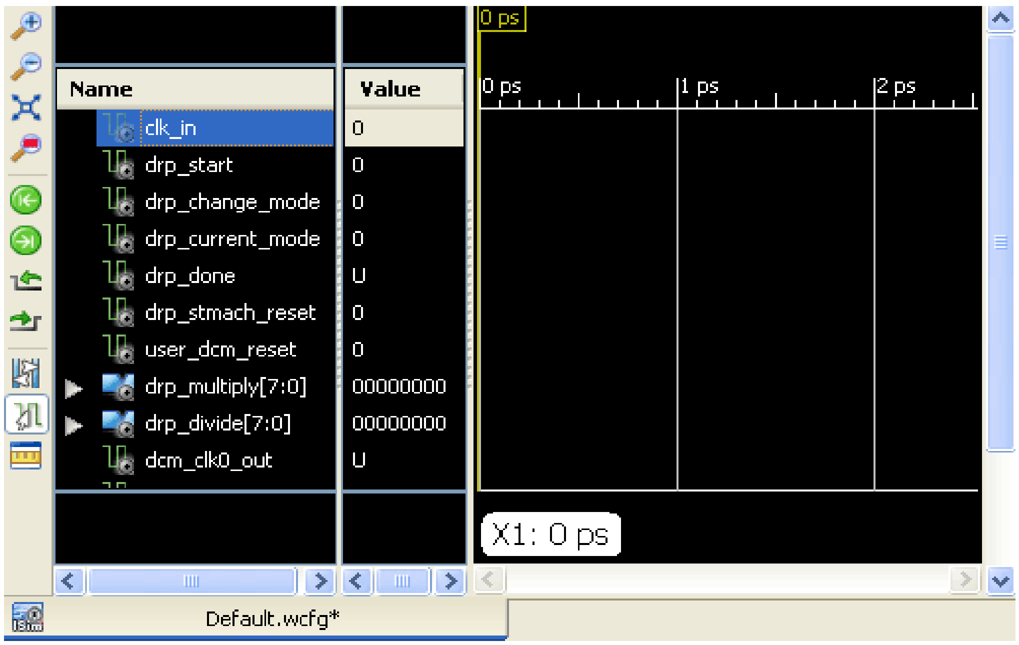


Figure 4-6: **Wave Window**

The Wave window displays signals, buses and their waveforms. Each tab in the Wave window represents a wave configuration, which consists of a list of signals and buses, their properties, and any added wave objects, such as dividers, cursors, and markers.

In the user interface, the signals and buses in the wave configuration are being traced during simulation, and therefore, the wave configuration is used to drive the simulation, and to then examine the simulation results. Since design and simulation data are contained in a database, simulation data is not affected when adding signals to or removing signals from the wave configuration.

Text Editor

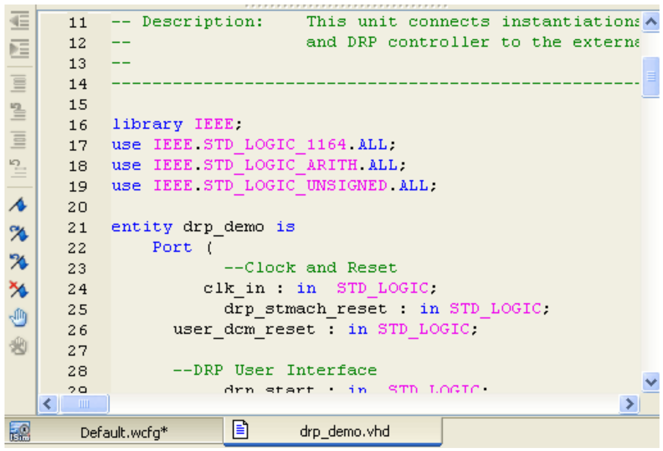


Figure 4-7: **Text Editor**

The text editor window is available for easy access to the HDL source files used in the simulation. Basic steps available are:

- Opening HDL source files (read mode only)
- Viewing HDL source files
- Setting breakpoints to source files for debugging.
- Step through the source code using stepping

Breakpoints Panel

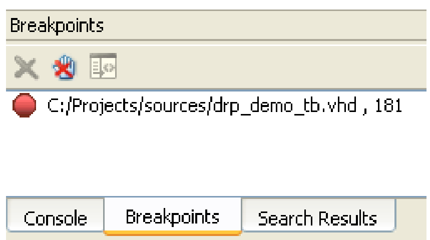


Figure 4-8: **Breakpoints Panel**

The Breakpoints panel displays a list of all breakpoints currently set in the design. For each breakpoint set in your source files, the list in the breakpoints panel identifies the file location, file name and line number. You can delete a selection, delete all breakpoints, and go to the source code from the Breakpoint panel toolbar icons or context menu.

For more information, see Chapter 4, “Debugging the Design” in the *ISim User Guide*:

http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/plugin_ism.pdf

Console Panel

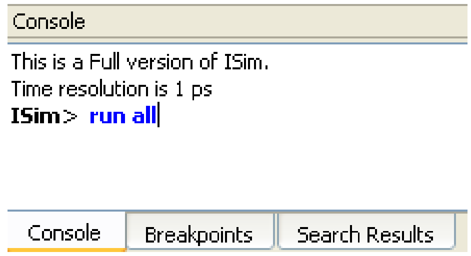


Figure 4-9: **Console Panel**

The Console panel enables you to view a log of messages generated by ISim, and to enter Tcl standard and ISim-specific commands at the command prompt.

Examining the Design

In this section, you will perform several steps to further analyze the functional behavior of the tutorial design. These include:

- **Running and restarting** the simulation to review the design functionality, using signals in the wave window and messages from the test bench shown in the Console Panel.
- **Adding signals** from the test bench and other design units to the wave window so their status can be monitored.
- **Adding groups and dividers** in order to better identify signals in the wave window
- **Changing signal and wave window properties** to better interpret and review the signals in the wave window.
- **Using markers and cursors** to highlight key events in the simulation and to perform zoom and time measurement features.
- **Using multiple wave window configurations** to further enhance your ability of reviewing multiple signals in one simulation session.

Adding Signals

Note: Skip this step if you completed the “Running ISE Simulator (ISim) from the ISE Project Navigator”. All visible simulation objects from the test bench should have been automatically added to the wave window.

Prior to running for a specified time in the simulator, you will need to add signals to the wave window so you can observe the signal status.

You will add all available simulation objects from the testbench to the wave window, which include:

- **Input Clock (clk_in)**: This is a 100 MHz clock generated by the test bench and will be the input clock into the Digital Clock Manager (DCM).
- **Dynamic Reconfiguration Ports (DRP) (drp_*)**: These are signals associated with the DCM DRP feature. The test bench asserts and monitors these signals to control and review the DCM DRP functionality.
- **DCM Output signals (dcm_*)**: These are output clocks from the DCM.

To add these signals to the wave window:

1. Right-click on the **drp_demo_tb** instance unit, in the Instances and Processes panel. (Refer to Figure 4-10.)
2. Select **Add to Wave Configuration**.

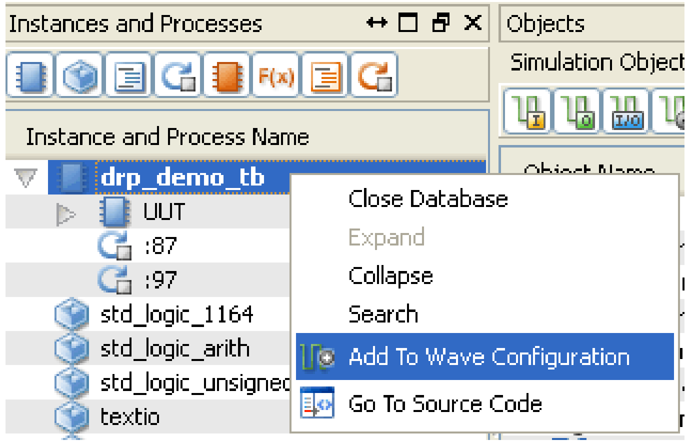


Figure 4-10: **Add to Wave Configuration**

All visible simulation objects from the drp_demo_tb test bench will now show up in the wave configuration. (Refer to Figure 4-11.)

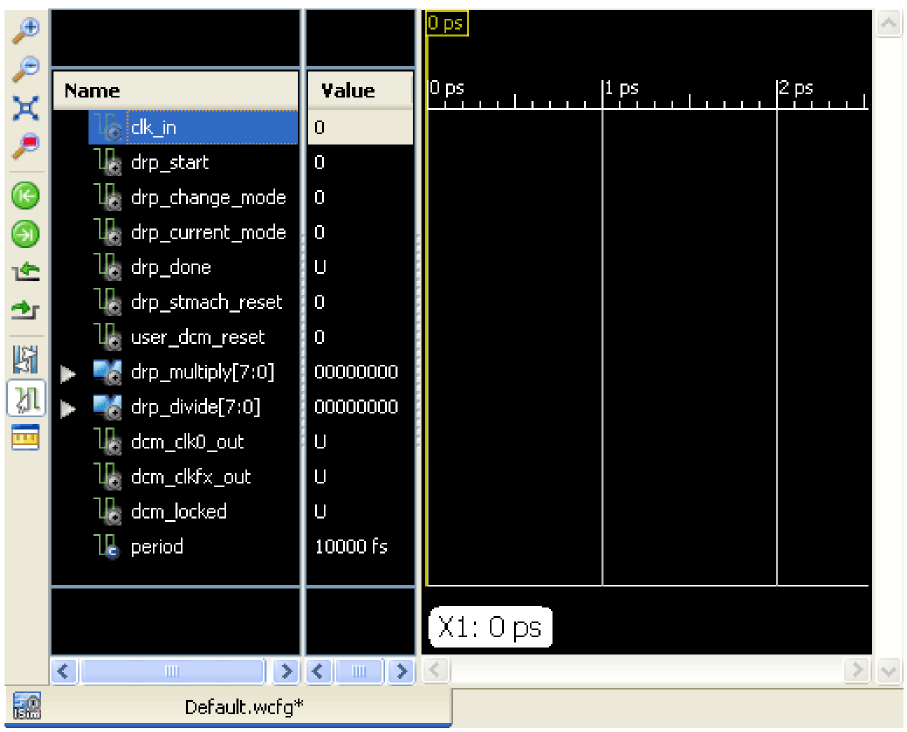


Figure 4-11: **Wave Configuration**

Running the Simulation for a Specified Time

You can now run the simulator for a specified time. Run the simulation for 5 microseconds (us). You can do so by either:

- Typing "5 us" in the Simulation Time field on the menu toolbar (refer to Figure 4-12); then, either
  - ♦ Pressing the "enter" key.
  - ♦ Clicking the Run For toolbar button, or
  - ♦ Select the menu command **Simulation** > **Run for.**

Figure 4-12: **Simulation Time Field**

- Typing "run 5 us" in the Tcl prompt (refer to Figure 4-13), then pressing **Enter**.

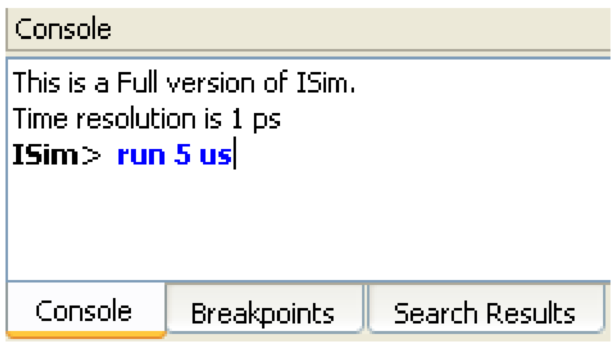


Figure 4-13: **ISim Tcl Prompt**

The wave window now shows traces of the signals up to 5 microseconds in simulation time. (Refer to Figure 4-14.)

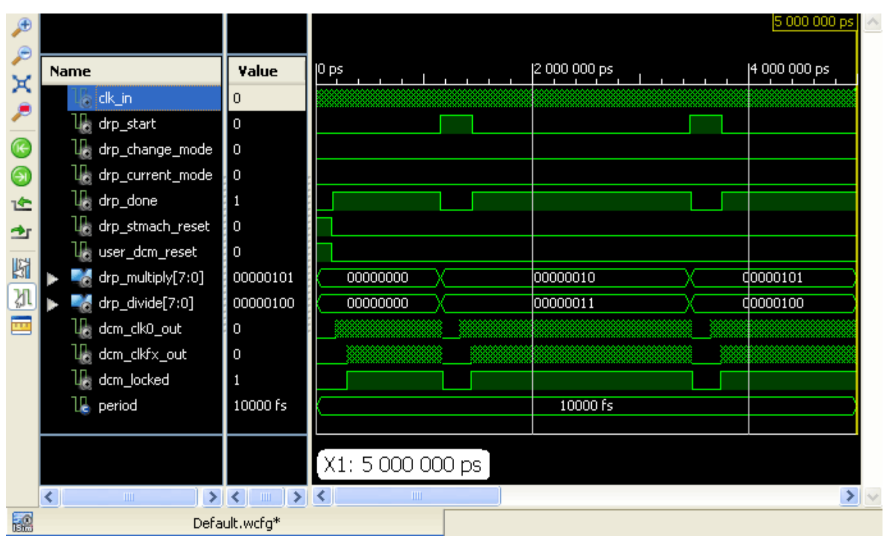


Figure 4-14: **Wave Window**

Note:

- Use menu command **Edit** > **Zoom** > **Zoom Full View** or click the Zoom Full View icon to view the full time spectrum.
- You can use the horizontal and vertical sliders to view the full wave configuration.
- There are assertions from the test bench during the time of simulation. Review the Console panel for messages from the test bench. (Refer to Figure 4-15.)

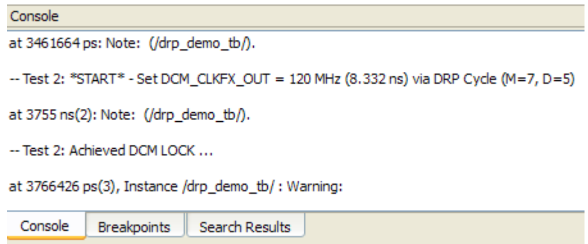


Figure 4-15: **Console Panel**

In the next tutorial steps, you will be analyzing the simulation of the tutorial design in more detail using features from the wave window, such as dividers, groups, cursors and markers.

Before you continue, restart the simulation to clear the wave window and set the simulation time to 0 picoseconds (ps).

Restarting the Simulation

To restart the simulation, either:

- Click the Restart icon in the menu toolbar
- Run menu command **Simulation** > **Restart**.
- Type “restart” in the Tcl prompt.

The wave window should look like the one shown in Figure 4-16:

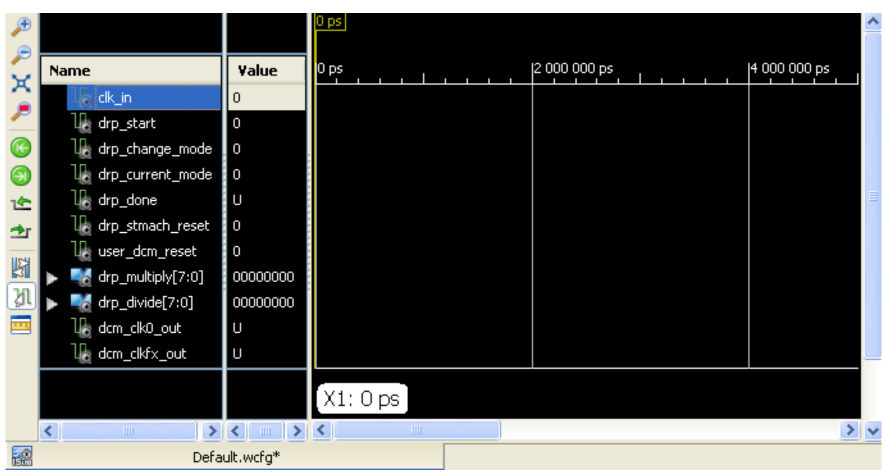


Figure 4-16: **Wave Window**

Adding Groups

In the next steps, you will be adding signals from other design units in order to better analyze the functionality of this design. However, soon after you add additional signals to the wave window, the size of the wave window will not be large enough to display all signals in the same view. Reviewing all signals would require the use of the vertical scroll bar in the wave window repeatedly, making the review process rather tedious.

We can remedy this situation by collecting signals into a **group.** With a group, you can collectively show or hide signals of similar purpose.

To group signals in the wave configuration:

1. While holding down the Ctrl key, select signals on the wave window of similar purpose.
2. Right-click on either of the selected signals. Select **New Group**.
3. Enter a name for the group (i.e., "DRP Test Signals").
4. A collapsed group will be created in the wave window. To expand the group, click once to the left of the group name.

Use the instructions above to make groups for the following signals:

1. All signals in the drp_demo_tb design unit that start with "drp_". Name the group "DRP Test Signals".
2. All signals in the drp_demo_tb design unit that start with "dcm_". Name the group "DCM Test Signals".

Expand all the created groups. Your wave window should be similar to the one shown in Figure 4-17.

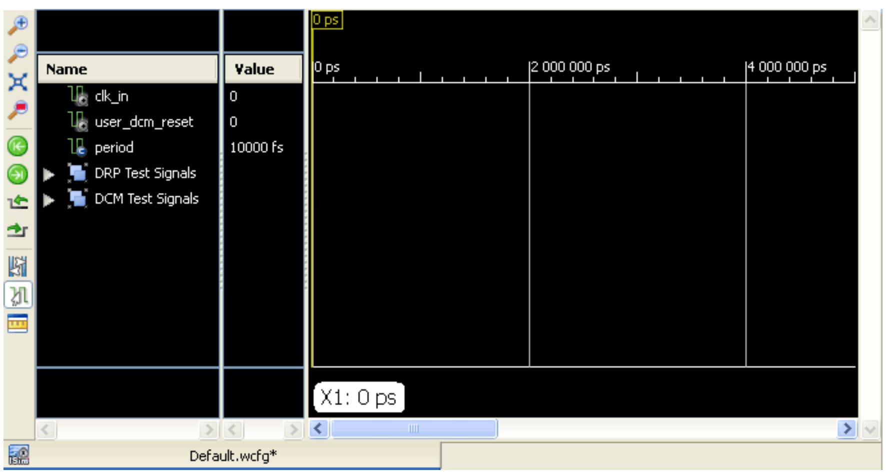


Figure 4-17: **Wave Window**

Note: If your signal groups do not match the figure shown above, you can use the following techniques to fix them:

- If you included an unrelated signal, you can **cut** it from the group and **paste** it into the main list.
- If you created the group but missed a signal in the main list, simply **drag and drop** the signal into the group. The signal will then be placed inside the group
- You can **undo** the group by using the **Edit** > **Undo** menu command.

- You can start over by **ungrouping** a group. Right-click on the group you wish to ungroup, then select **Ungroup.**

Adding Dividers

Soon you will be adding signals from other design units in order to better analyze the functionality of this design. To better visualize which signals belong to which design units, we can add dividers to separate the signals by design unit.

To add dividers to the wave window:

1. Right-click anywhere on the wave window, select **New Divider**.
2. Enter a name for the divider.

Use the instructions above to add three dividers named:

- TEST BENCH
- DCM
- DRP CONTROLLER

Move the **TEST BENCH** divider to the top of the list by clicking the divider name and holding the mouse button down while moving the cursor to the top of the list. Move the other dividers to the bottom of the list.

Note: Divider names can be changed at any time by double-clicking on the divider name or pressing the F2 function key, and entering a new name.

Your wave window should be similar to the one shown in Figure 4-18 (with groups collapsed).

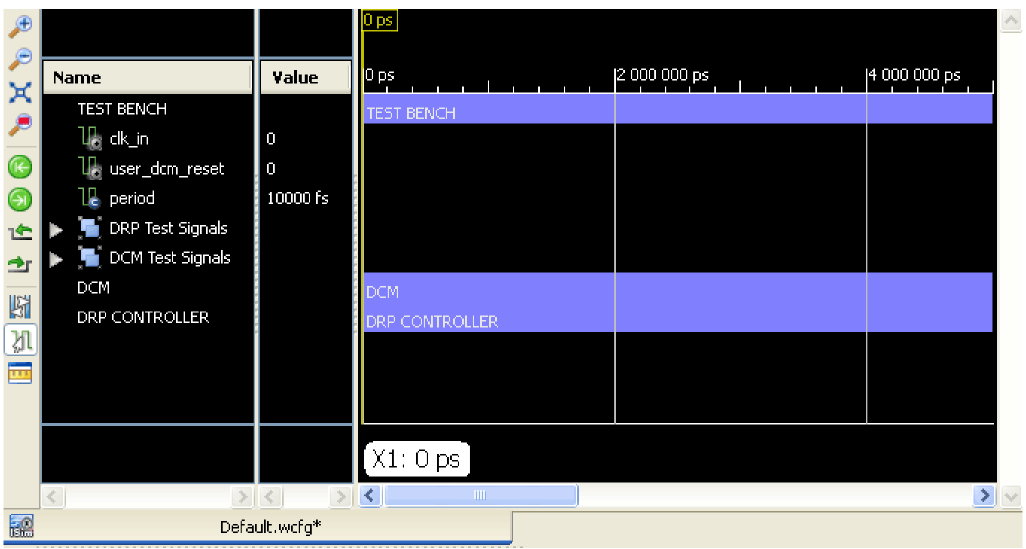


Figure 4-18: **Wave Window**

Adding Signals from Sub-Modules

You will now add signals from the instantiated DCM module (**Inst_drp_dcm**) and the instantiated DRP controller module (**Inst_drp_statmach**) in order to study the interactions between these sub-modules and the test bench test signals.

Follow these steps to add the necessary signals:

1. In the Instance and Process panel, expand the hierarchy by clicking once to the left of each child module (refer to Figure 4-19).
2. Simulation objects associated with the currently highlighted design unit will appear in the Objects panel (refer to Figure 4-20).

You shall first add all input and output ports from the **Inst_drp_dcm** design unit instantiation onto the wave window.

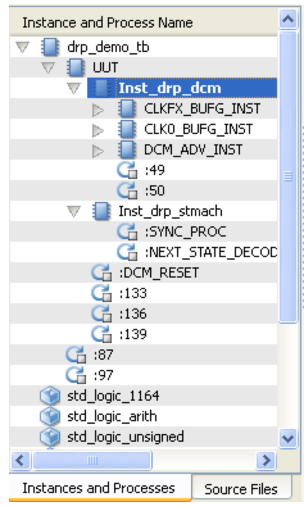


Figure 4-19: **Instances and Process Panel**

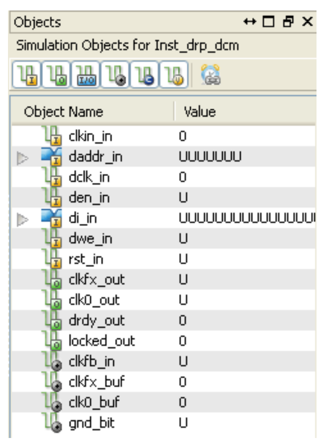


Figure 4-20: **Simulation Objects Panel**

To add the input/output ports of instance Inst_dcm_drp to the wave window, either:

- Highlight the "Inst_drp_dcm" design unit in the Instance and Process panel, then right-click on the input/output ports in the Objects panel. Select **Add to Wave Configuration** from the context menu. (Refer to Figure 4-21.)

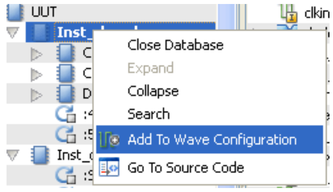


Figure 4-21: **Add to Wave Configuration**

- Select the input/output ports of the Inst_drp_dcm design unit while holding the **Ctrl** key. Then, drag and drop the signals to the wave window.
- Enter the "wave add" Tcl command in the ISim Tcl prompt. For example:

wave add /drp_dcm_tb/uut/drp_dcm/

Note: By default, all types of simulation objects (variables, constants, etc.) are displayed in the Objects panel. You can filter the type of simulation objects shown in this panel. Use the Objects panel toolbar to filter by inputs, outputs, bi-directional, internal, constants and variables. Toggle the desired object type by clicking on the corresponding icon.

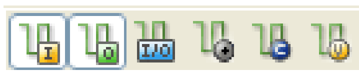


Figure 4-22: **Inputs, outputs, bi-directional, internal, constants and variables**

3. You can move the recently added signals if they do not appear directly under the DCM divider.
   - While holding Ctrl+Shift key, click once on the first added DCM signal (clk_in) and the last added DCM signal (gnd_bit).
   - Once all signals are selected, move the signals under the DCM divider by holding the mouse button and placing the mouse cursor right under the divider name.

Repeat the steps above for input/output ports of **Inst_drp_statmach** instantiated design unit.

Additionally, you can also create groups for the signals recently added. Using the instructions provided for adding groups, define groups "Inputs", "Internal", and "Outputs" for each set of signals recently added.

Note: Use the object icon to the left of the signal name to determine the type of the simulation object (Figure 4-23):

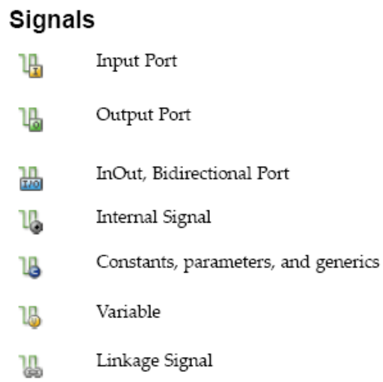


Figure 4-23: **Signals and Icons**

Your wave window should be similar to the one shown in Figure 4-24 (with groups collapsed).

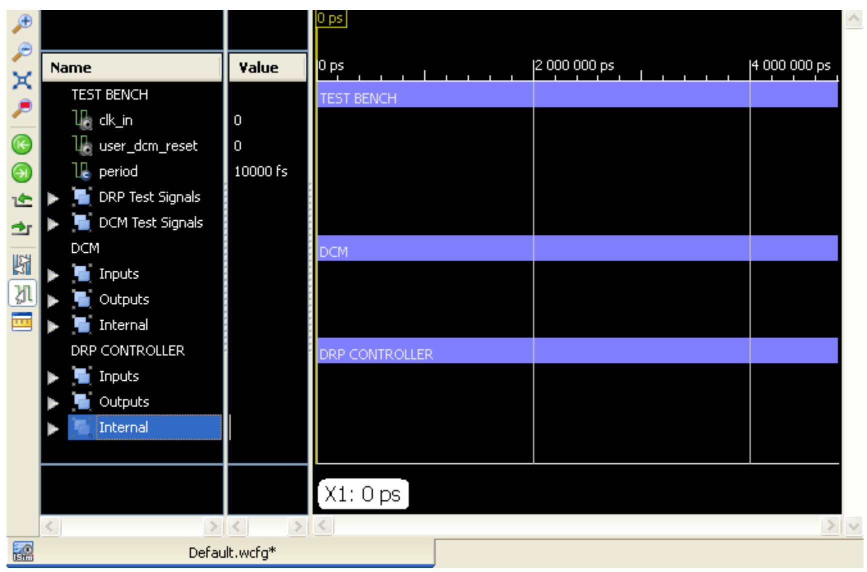


Figure 4-24: **Wave Window**

Changing Signal and Wave Window Properties

Next you will change the properties of some of the signals currently shown in the wave window in order to better visualize the behavioral simulation.

Changing the Signal Name Format

By default, ISim adds signals to the waveform using the short name (hierarchy reference removed). For some signals, it is important to know which module they belong to.

You will change the format of the following bus signals from "Short" to "Long", listed under the DRP Test Signals group:

- drp_multiply
- drp_divide

To change the signal name format:

1. In the wave window, right-click on the signal name, listed under the Name column.
2. Select Name > **Long**. (Refer to Figure 4-25.)

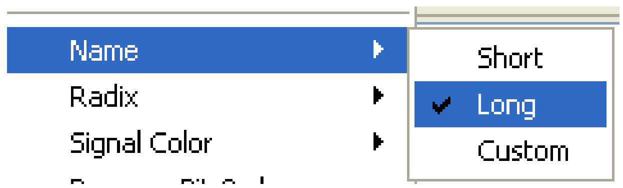


Figure 4-25: **Change the Signal Name Format**

Note: You can perform a format change on multiple signals with fewer clicks by:

- Selecting multiple signals using Ctrl+Shift.
- Applying the format change via the right-click context menu.

Changing the Signal Radix Format

Some signals are better interpreted if seen in hexadecimal rather than in binary. For example, the signals **drp_multiply** and **drp_divide** are bus signals that are best interpreted in hexadecimal format, rather than binary.

You will change the format of the following signals from “Binary” to “Hexadecimal”:

- drp_demo_tb / drp_multiply
- drp_demo_tb / drp_divide

To change the radix of a signal:

1. In the wave window, right-click on the signal name, listed under the Name column.
2. Select **Radix**, then the radix type you wish to interpret the signal in. (Refer to Figure 4-26.)

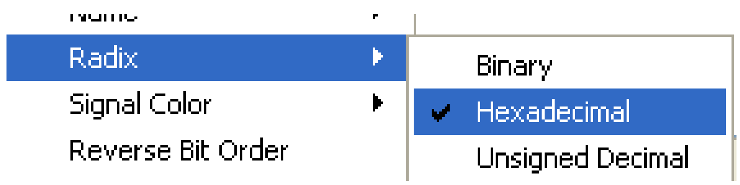


Figure 4-26: **Changing the Radix of a Signal**

Changing the Signal Color

ISim allows you to change the signal color in the wave window to help you quickly identify similar signals from each other.

You will change the format of the following signals from their default color to a color of your choice:

- drp_demo_tb / drp_multiply
- drp_demo_tb / drp_divide

To change the color of a signal:

1. In the wave window, right-click on the signal name, listed under the Name column.
2. Select **Signal Color**, then pick a color from the color palette, or a custom color by clicking on the ellipsis (…) button. (Refer to Figure 4-27.)

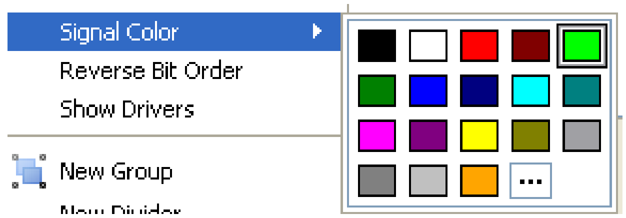


Figure 4-27: **Changing the Signal Color**

Floating the Wave Window

Depending on your screen resolution, you may notice that the wave window has been populated with more signals than the screen can view at one time. To alleviate this problem, we can increase the viewable area by **floating** the wave window. Following this step will open a new window with just the waveform contents.

To **float** a window, either:

- While highlighting an object in the wave window, select **View** >**Float**.
- Click once on the Float Window main toolbar icon:

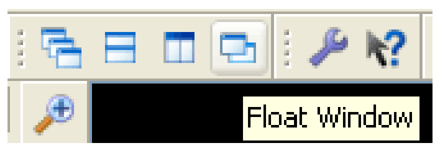


Figure 4-28: **Selecting Float from the View Menu**

- Right-click on the wave configuration name tab and select **Float.**

Figure 4-29: **Selecting Float from the Wave Configuration Name Tab**

You are done making modifications to the wave window. The wave window should now look similar to Figure 4-30. (Test bench groups are expanded.)

Figure 4-30: **Wave Window**

Saving the Wave Window Configuration

You can save the current state of the wave window (wave configuration) so it is available for use in future ISim simulation sessions of your design.

To save the wave configuration:

1. Use File > **Save As** to assign a name to the current wave configuration. (Refer to Figure 4-31.)

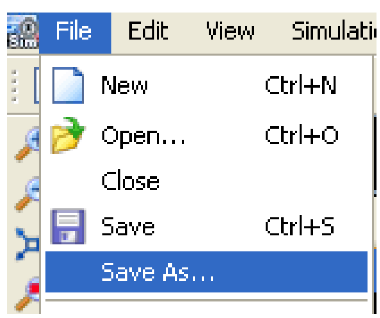


Figure 4-31: **Saving the Wave Window Configuration**

2. Save the current wave configuration as “tutorial_1.wcfg”.

The wave configuration is now saved for future use.

Note: You can load the saved wave window configuration using the menu command File > **Open**. This feature is useful when you have se tup a wave configuration that you will reuse in future simulation sessions of the design.

You are ready to simulate the design again with the updated wave configuration. Re-run the simulation by either:

- Use Run All  from the main toolbar.
- Use the menu command **Simulation** >**Run All**.
- Type "run all" on the Tcl prompt.

The simulation will run for about 13 microseconds (us).

After the simulation is complete, use the menu toolbar icon  to zoom to full view.

The wave configuration should look similar to Figure 4-32.

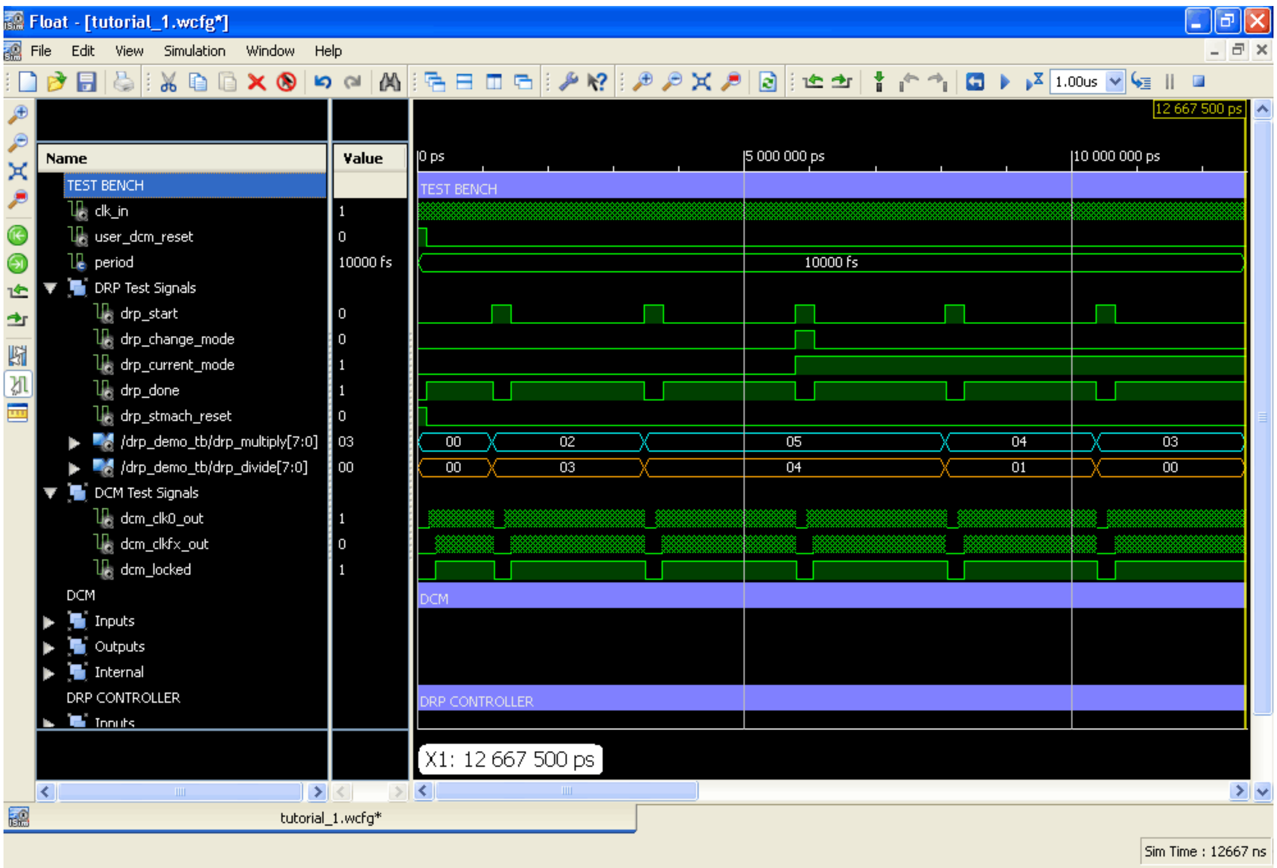


Figure 4-32: **Wave Configuration**

Using Markers

The self-checking test bench used in this design performs 4 different tests to showcase the functionality of the DCM Dynamic Reconfiguration feature. Follow the next steps to mark each time a new test has started with markers in the wave window.

1. In the Console panel, identify the simulation times when each test has started. For example, Test 2 starts at about 3.46 microseconds (3,461,664 ps) as shown by this segment of the ISim console:

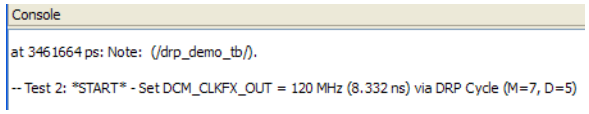


Figure 4-33: **Console Window**

2. From the menu, select **Edit** > **Go To…** to move the main (yellow) cursor when the first test bench test is performed. (It should be about 1,150 ns.)

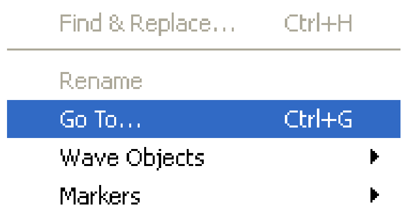


Figure 4-34: **Edit > Go To...**

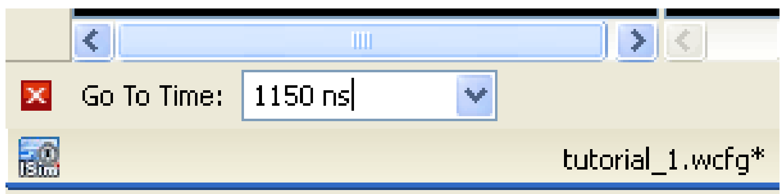


Figure 4-35: **Go To Time**

3. Add a marker at this time. To add a marker, either:
   - Use the Add Marker  icon in the main toolbar.
   - Use the menu command **Edit** > **Markers** > **Add Marker.**
4. Repeat these steps for all 4 tests performed by the test bench. The wave window should look similar to Figure 4-36.

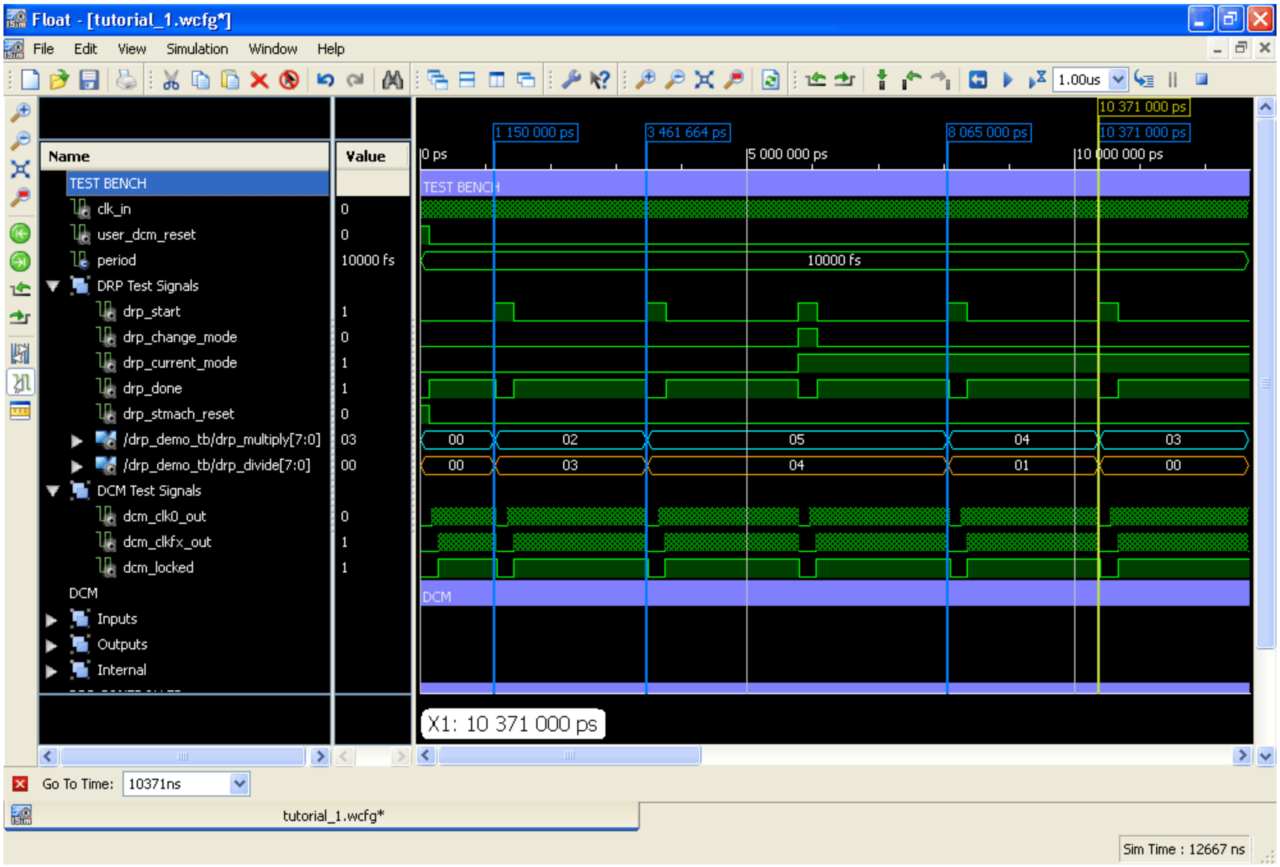


Figure 4-36: **Wave Window**

Using Cursors

The ISim Console reports that Test 2 and Test 4 failed (Figure 4-37).

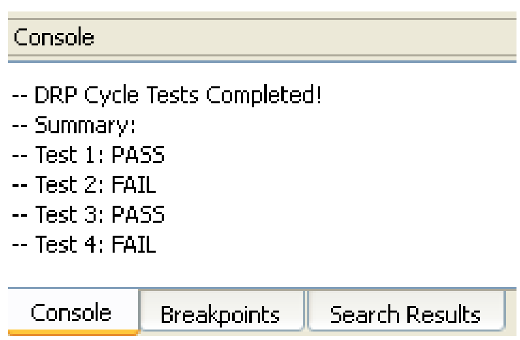
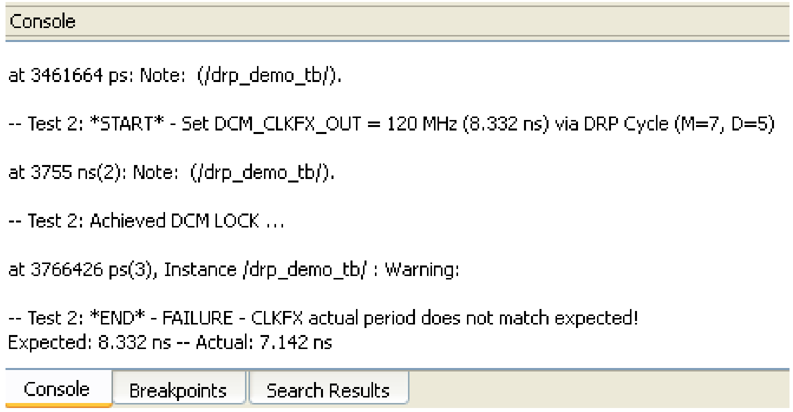


Figure 4-37: **Console Report Test 2 and Test 4 Failed**

In Test 2 and 4, a Dynamic Reconfiguration (DRP) write cycle is performed in order to change the multiply and divide factors of the Digital Frequency Synthesizer and set new clock output (CLKFX) frequencies (120 MHz and 400 MHz respectively). However, at the end of the DRP cycle, the test bench measured a period that did not match the expected period. Tests 2 and 4 fail due to the period discrepancy (Figure 4-38, Figure 4-39).



Console

at 3461664 ps: Note: (/drp_demo_tb/).

-- Test 2: *START* - Set DCM_CLKFX_OUT = 120 MHz (8.332 ns) via DRP Cycle (M=7, D=5)

at 3755 ns(2): Note: (/drp_demo_tb/).

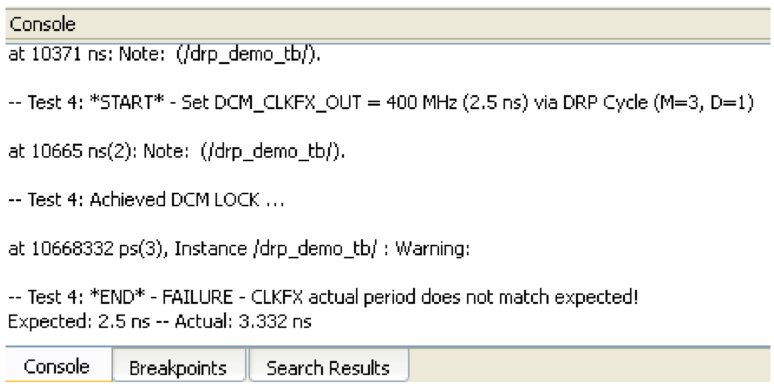
-- Test 2: Achieved DCM LOCK ...

at 3766426 ps(3), Instance /drp_demo_tb/ : Warning:

-- Test 2: *END* - FAILURE - CLKFX actual period does not match expected!
Expected: 8.332 ns -- Actual: 7.142 ns

Console | Breakpoints | Search Results

Figure 4-38: **Test 2 Fails Due To Period Discrepancy**



Console

at 10371 ns: Note: (/drp_demo_tb/).

-- Test 4: *START* - Set DCM_CLKFX_OUT = 400 MHz (2.5 ns) via DRP Cycle (M=3, D=1)

at 10665 ns(2): Note: (/drp_demo_tb/).

-- Test 4: Achieved DCM LOCK ...

at 10668332 ps(3), Instance /drp_demo_tb/ : Warning:

-- Test 4: *END* - FAILURE - CLKFX actual period does not match expected!
Expected: 2.5 ns -- Actual: 3.332 ns

Console | Breakpoints | Search Results

Figure 4-39: **Test 4 Fails Due To Period Discrepancy**

In the next few steps, you will use the ISim main cursor (yellow cursor) to zoom in the wave window when one of the failing tests takes place. You will also use the cursor to measure the period of signal **dcm_clkfx_out** and verify that the test bench is making accurate measurements.

Zooming In

Let us first zoom in when Test 2 starts to review the status of output clock **dcm_clkfx_out**.

To use a cursor for zooming in on a specific area:

1. Place the cursor on the desired area. You can do so by:
   - Dragging the main cursor (yellow cursor) close to the marker that represents the start of Test 2 (marker at time 3,461,664 ps). The cursor will snap onto the marker.
   - Click the Previous Marker  or Next Marker  toolbar icons to quickly move the main cursor from marker to marker.
   - Select **Edit** > **Go To** and specify the time when Test 2 starts (time 3,461,664 ps). The main cursor will now move to this time location.

2. Zoom in by either:
   - Clicking the Zoom In toolbar icon.
   - Selecting the menu command **View** > **Zoom** > **Zoom In**.
   - Press F8 function key.

The wave window will zoom in around the area specified by the cursor. Use step 2 above repeatedly until you can clearly see DCM test signals **dcm_clk0_out** and **dcm_clkfx_out** toggle.

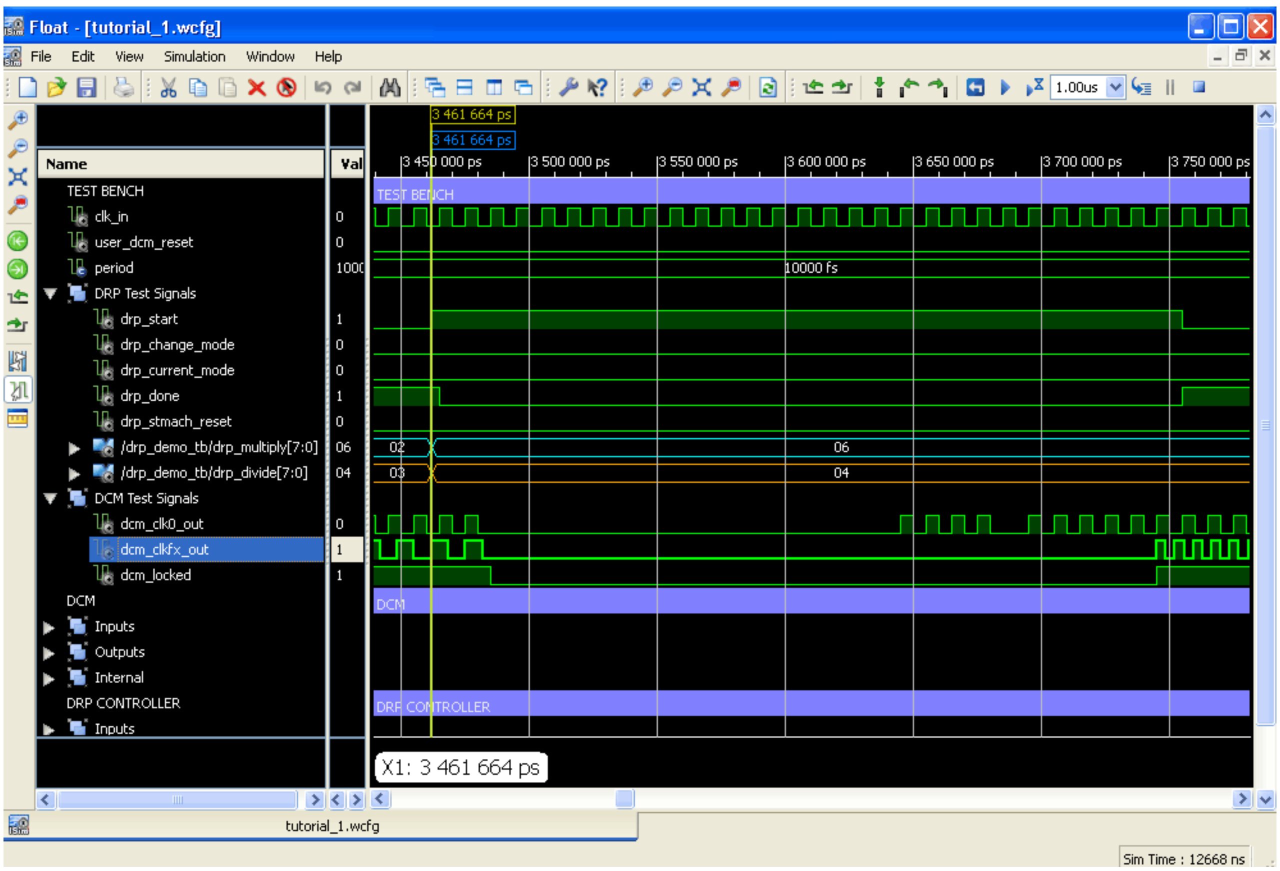


Figure 4-40: **Wave Window**

Measuring Time

You can use the main cursor to measure time between two endpoints. You will use this feature to confirm the test bench calculations reported in the console during Test 2 by measuring the period of **dcm_clkfx_out** after the DRP cycle has completed (signal **drp_done** is asserted).

To measure time using cursors:

1. Use the Snap to Transition toggle button to easily snap the cursor on to transition edges.

2. Press and hold the left mouse button in an area around the first clock rising edge following DRP cycle completion (drp_done signal asserted). The main cursor will snap to the rising edge of dcm_clkfx_out.
3. While holding the button, move the mouse over to the next clock rising edge. A second marker should appear.
4. The time between the two defined endpoints will appear at the bottom of the wave window as a time delta (refer to Figure 4-41).

Note: Use Zoom In  for better performance of the time measurement feature.

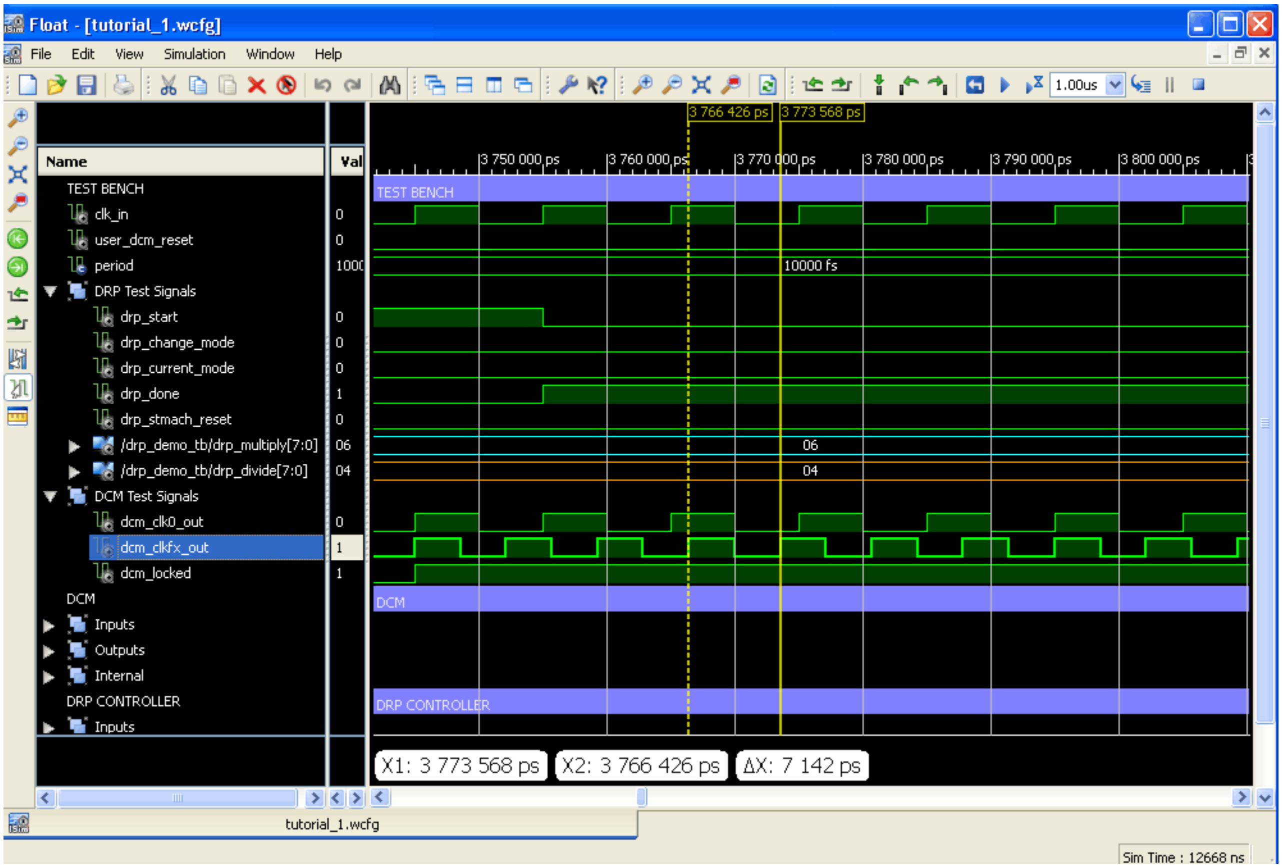


Figure 4-41: **Time Delta**

Using the cursors, we measure a 7,142 ps time difference between two rising edges of the **dcm_clkfx_out** output clock. This translates to a 140 MHz clock signal. Test 2 fails due to the frequency discrepancy (expected is 75 MHz).

Repeat the same steps above to analyze the Test 4 failure. You should observe that while the test bench expects a frequency of 400 MHz, the actual frequency measured is 300 MHz.

Note: Use the Floating Ruler  feature (available from the wave window toolbar) to display a hovering ruler over the wave configuration. This feature is available when performing a time measurement using cursors between two endpoints. The zero (0 ps) on the ruler is placed at the first time endpoint. This feature is useful when making multiple time measurements with respect to the first endpoint (Figure 4-42).

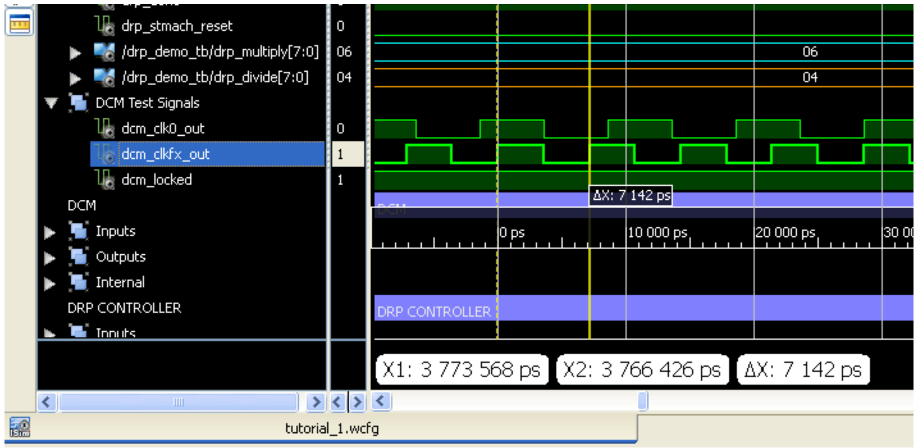


Figure 4-42: **Floating Ruler Feature**

Using Multiple Wave Configurations

Depending on the resolution of the screen, a single wave window may not display all the signals of interest at the same time. You can resolve this problem by opening multiple wave windows, each with their own set of signals and signal properties.

To open a new wave window:

- In ISim, select **File** > **New**. In the resulting pop-up window, select **Wave Configuration** and click **OK** (Figure 4-43).
- A blank wave configuration will be shown.

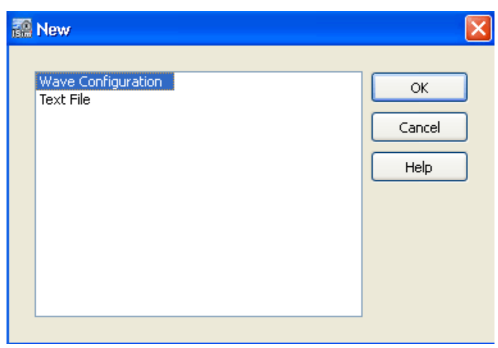


Figure 4-43: **New Wave Configuration**

To move dividers, groups and simulation objects to the new wave configuration:

1. While pressing the Ctrl key, highlight objects you want to move to the new wave window.
2. Right-click on either of the selected signals, and select **Cut**.
3. Enable the new wave configuration, **untitled 1**, by clicking on its corresponding window tab.
4. Right-click in the Name column area of the wave configuration, and select **Paste**.

Use the instructions above to move all the simulation objects associated with the DCM and DRP Controller units to a new wave window (dividers, groups, etc.). Upon completion of this task, select **File** > **Save As** to save this wave configuration as **tutorial_2.wcfg**.

You should now have two wave windows that should look similar to Figure 4-44 and Figure 4-45.

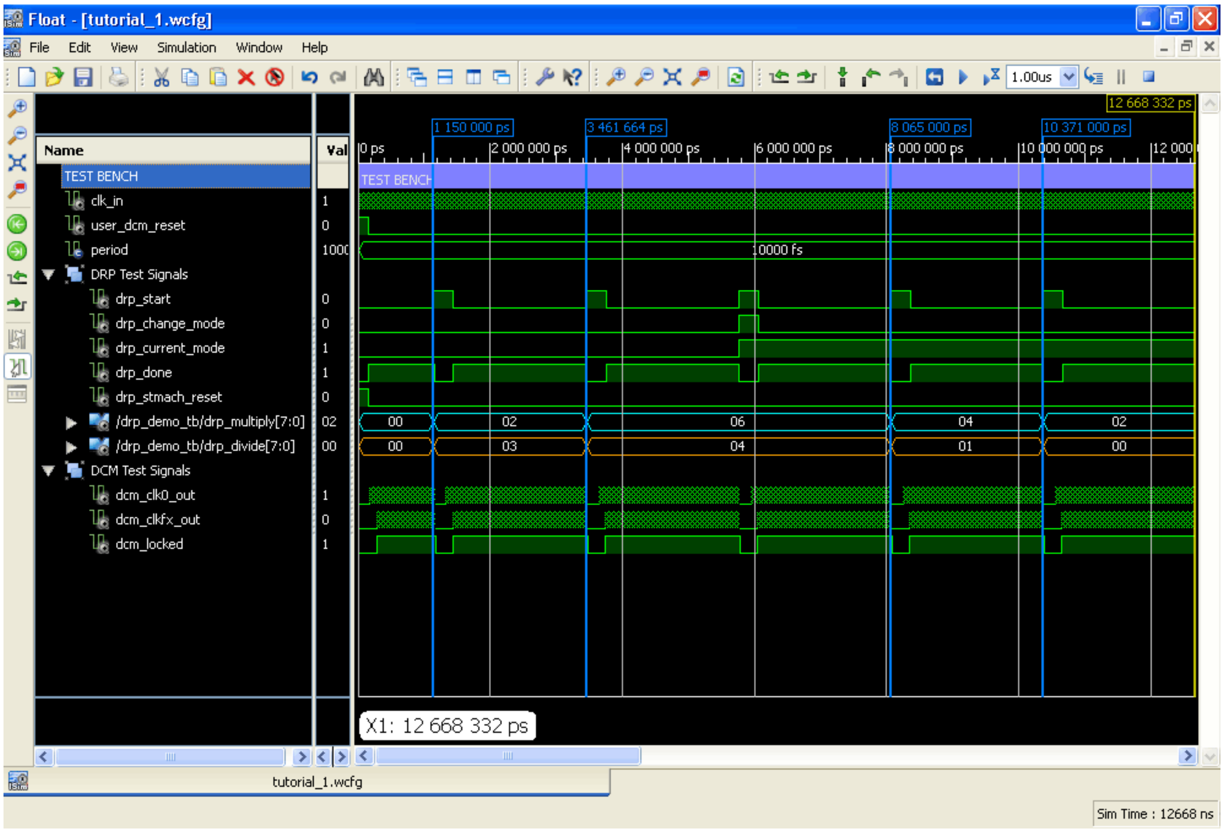


Figure 4-44: **Wave Window**

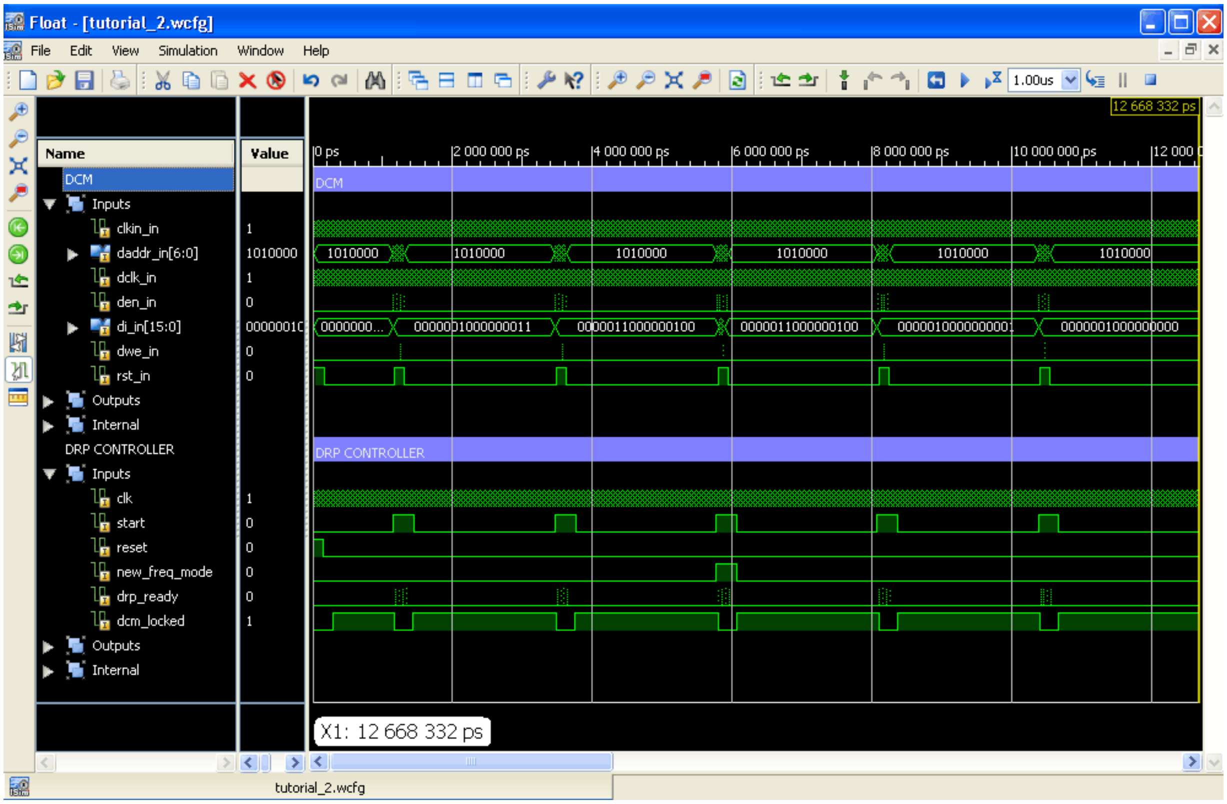


Figure 4-45: **Wave Window**

Debugging the Design

Now that you have examined the design using markers, cursors, and multiple wave configurations, you will now use ISim debugging features, such as setting breakpoints and stepping through source code, in order to debug the design and address the two failing DRP tests.

Viewing Source Code

First, take a look at the test bench for the tutorial design and learn how each test is performed.

To open a source code (read-only mode), either:

- Select **File** >**Open** to point to the file of choice.
- In the Instances and Processes Panel, right-click on the design unit described by the source file of interest, then select **Go to Source Code.**
- In the Objects Panel, right-click on any of the simulation objects declared in the source file of choice, then select **Go to Source Code**.
- In the **Source Files Panel** (viewable by clicking on the "Source Files" tab), double-click on the source file of choice.

Use the directions above to open the source code for the tutorial design test bench (**drp_demo_tb.vhd)**. The source file will be opened using the integrated text editor. (See Figure 4-46.)

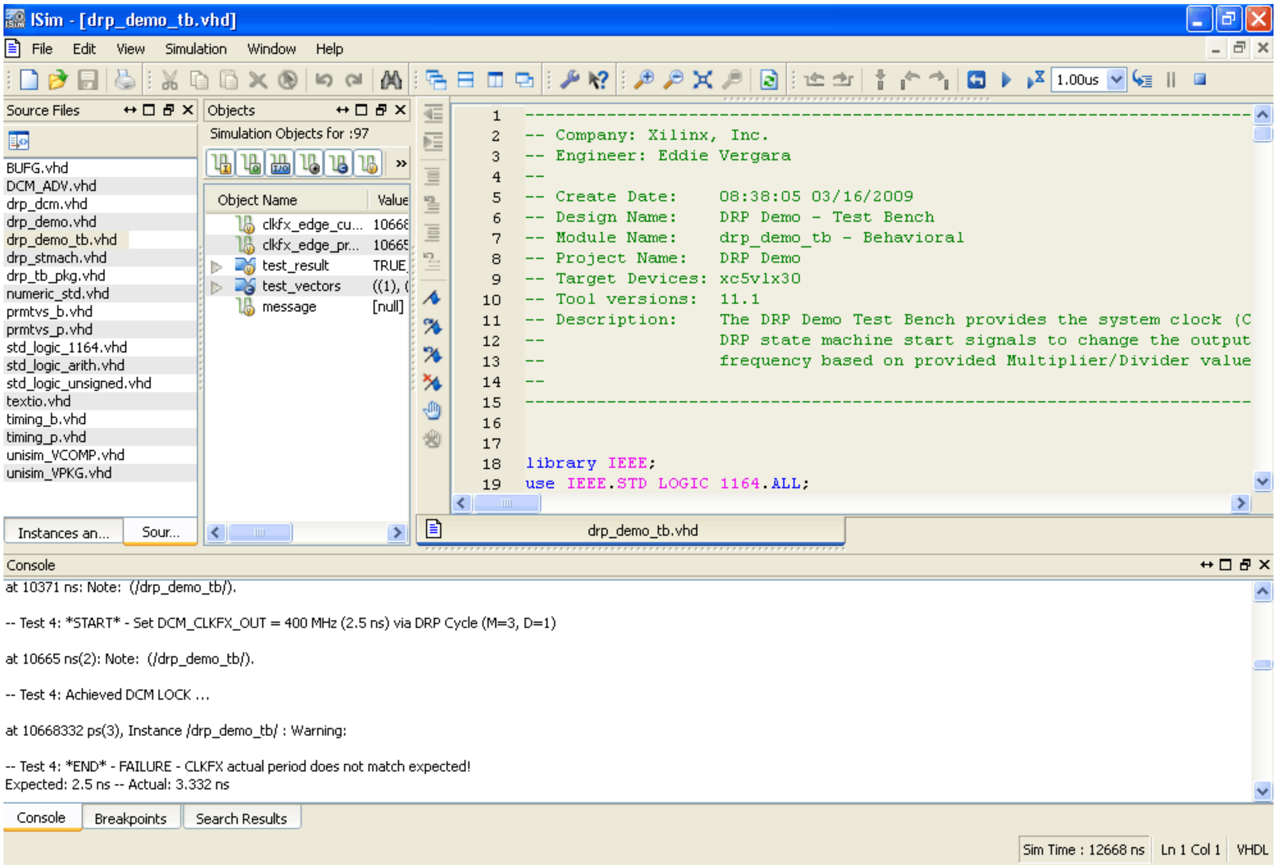


Figure 4-46: **Integrated Text Editor**

Using Breakpoints and Stepping

A breakpoint is a user-determined stopping point in the source code used for debugging the design with ISim. When simulating a design with set breakpoints, simulation of the design stops at each breakpoint in order to verify the design behavior. Once the simulation stops, an indicator is shown in the text editor next to the line of source code where the breakpoint was set, allowing you to compare the wave window results with a particular event in the source code.

Another useful ISim debugging tool is the Stepping feature. With stepping, you can run the simulator one simulation unit at the time. This is helpful if you are interested in learning how each line of your source code affects the results in simulation.

We can use both of these debugging features to learn how the DRP cycle is performed during Test 2 in an attempt to debug the failing test.

Setting Breakpoints

Begin by first setting a breakpoint around the first signal assignment performed during each of the DRP cycle tests.

To set a breakpoint:

1. Open the source code which will contain the breakpoint.
2. Go to an executable line in the source code which will contain the breakpoint.
3. Add a breakpoint by either:
    - Right-clicking anywhere on the executable line and selecting **Toggle Breakpoint**.
    - Highlighting the line by performing a left-click on the line number, then using the menu command **View** > **Breakpoint** > **Toggle Breakpoint**.
    - Clicking the text editor toolbar breakpoint icon .

Use the instructions above to set a breakpoint at line 185 in **drp_demo_tb.vhd** (see Figure 4-47). Doing so will cause the simulator to stop every time the signal **drp_multiply** is assigned a value.

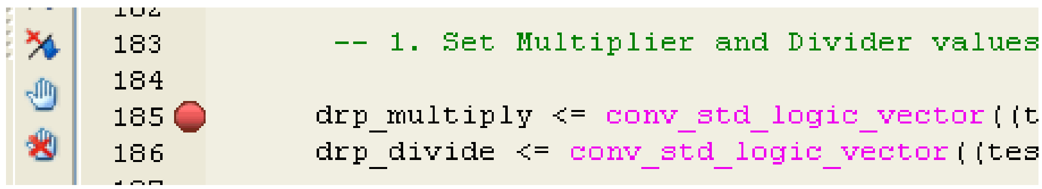


Figure 4-47: **Setting a Breakpoint at Line 185 in drp_demo_tb.vhd**

Note: You can manage breakpoints by clicking on the Breakpoints tab (next to the Console tab). All set breakpoints will appear in this list. From here, you can:

- Delete selected breakpoint
- Delete all breakpoints
- Go to the line of source code for selected breakpoint

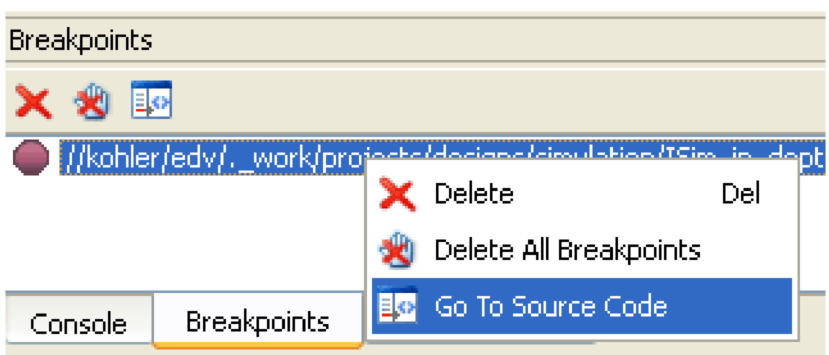


Figure 4-48: **Breakpoints Tab**

Re-run the simulation with the breakpoint enabled by following these steps:

1. Bring to focus the ISim main window.

Note: Debugging with the breakpoints and stepping feature works best when you are able to review the console output and the wave windows at the same time. Use the **float** feature of the ISim panels, or resize the windows of the simulator, to best accommodate the windows so they can be reviewed at the same time.

2. Restart the simulation by pressing the Restart  icon in the ISim menu toolbar.
3. Run the simulation by pressing the Run All  toolbar icon.

The simulation runs near the start of the first test.

Focus changes to the text editor while it shows, with a yellow indicator ( ), the last line of source code the simulator executed.

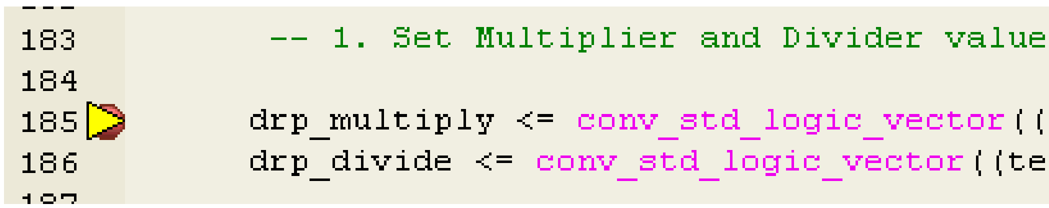


Figure 4-49: **Yellow Indicator Shows the Last Line of Source Code the Simulator Executed**

Additionally, a message will appear in the Console indicating that the simulator has stopped, including the line of source code last executed by the simulator.

4. We know Test 1 finishes successfully when we examined the design earlier. As such, we can skip debugging this test. Press the Run All toolbar icon to continue forward to Test 2.

The simulation now stops at the start of Test 2.

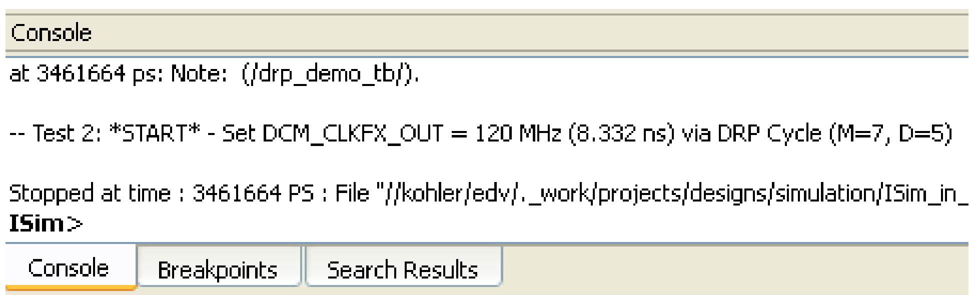


Figure 4-50: **Message in the Console Indicating That the Simulator Has Stopped**

Stepping through Source Code

You first need to verify that in Test 2, the appropriate Multiplier and Divider parameters are being set correctly via the **drp_multiply** and **drp_divide** bus signals. You will use **stepping** to step through the source code line by line and review how the **drp_multiply** and **drp_divide** bus signals are assigned to the DCM DRP ports.

To **step through a simulation**, either:

- Click on the Step toolbar icon.
- Select **Simulation** >**Step**.
- Type "**step**" in the Tcl prompt.

1. Use the instructions above to step through the design. As you step through the source code, pay close attention to each of these events:
   - **drp_multiply** and **drp_divide** bus signals are assigned values from a constant **test_vectors**.
   - **drp_start** asserts in order to start a DRP cycle.
   - **drp_multiply** bus signal is assigned to the 8 uppermost bits of bus signal DI_IN, while **drp_divide** bus signal is assigned to the 8 lowermost bits of the same bus.

- The DRP controller (**drp_stmach.vhd**) leaves idle mode and moves to the next DRP cycle step: clearing the DCM status registers.

2. In the “tutorial_2” wave window, expand the DCM Inputs bus.
3. Continue stepping through the simulation until the di_in bus signal is updated with a new value (you may need to zoom in considerably in order to observe the change). At around 3,465 ns, the bus should be updated from 0203h to 0604h.

Note: Change the radix of bus signal di_in to Hexadecimal to verify this value change.

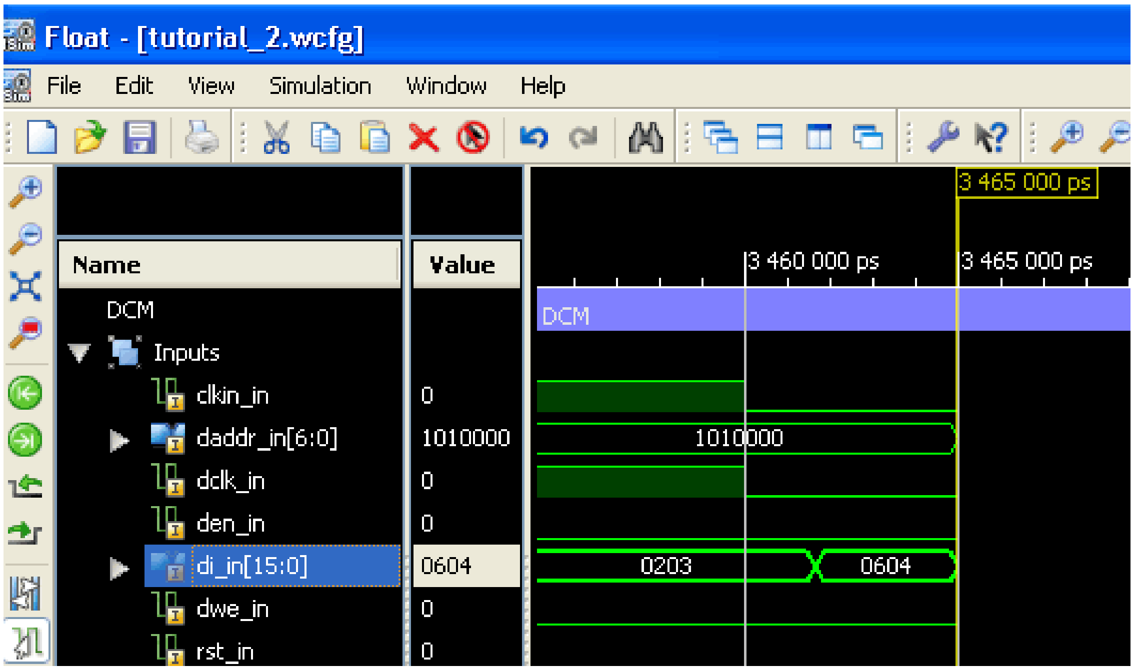


Figure 4-51: **Expanding the DCM Inputs Bus in the Tutorial 2 Wave Window**

4. The output clock frequency of this design (**dcm_clkfx_out)** is dependent on the multiply and divide factors provided by the user. For Test 2, we use the following parameters and expected output clock frequency:

Table 4-1: **Parameters and Expected Output Clock Frequency**

Test	Freq. (MHz)	Period (ps)	Multiplier (M)	Divider (D)
2	120	8,332	6	5

You may recall that for M=6 and D=5, di_in[15:0] bus value should be 0504h. Notice that the status of di_in in Test 2 is 0604h. Test 2 fails because an incorrect M/D factor is provided via the **drp_multiply** and **drp_divide** signals in the test bench.

5. You can repeat the steps above to determine the cause of failure for Test 4. You will determine that the failure is also due to incorrect assignments of the multiply and divide signals in the test bench.

Fixing Bugs in the Design

By using breakpoints and stepping, you have determined that the incorrect multiply and divide values are assigned to signals **drp_multiply** and **drp_divide** in the test bench.

In the next steps, revise the test bench test vectors to use the correct Multiplier and Divider parameters in tests 2 and 4.

1. Close the ISE Simulator by selecting **File** > **Close**.

   Note: If changes have been made to the wave configuration before the last save, ISim will remind you to save changes prior to closing the session.

2. Using a text editor, open the test bench source file, **drp_demo_tb.vhd**.
3. In lines 117 through 127, test vectors for the 4 DRP tests are defined. Revise the constant declaration to read (changes highlighted in **bold**):

```
------------------------------------------------
-- ** TEST VECTORS **
--  (Test, Frequency, Period, Multiplier, Divider)
------------------------------------------------

constant test_vectors : vector_array := (

  ( 1, 75, 13332 ps, 3, 4),
  ( 2, 120, 8332 ps, 6, 5),
  ( 3, 250, 4000 ps, 5, 2),
  ( 4, 400, 2500 ps, 4, 1));
```

4. Save and close the file.

Verifying Bug Fix

Now that the test bench source code has been fixed, you need to re-compile the source code and build a new simulation executable.

1. Re-launch the ISE Simulator.
   - **If you are using the ISim ISE Integrated flow**, in Project Navigator re-launch ISim by double-clicking on **Simulate Behavioral Model**.
   - **If you are using the ISim Standalone flow**, re-launch the ISE Simulator by running the **fuse** script, followed by the simulation executable (fuse_batch.bat and simulate_isim.bat).
2. Once ISim starts, load the wave configurations previously saved in "Examining the Design", **tutorial_1.wcfg** and **tutorial_2.wcfg**.

   To load a wave window configuration:
   - Select **File** > **Open**, and point to the wave configuration files (.wcfg).
3. We are ready to simulate the design again with the updated test bench. Re-run the simulation by either:
   - Click the Run All toolbar icon .
   - Use the menu command **Simulation** >**Run All** .
   - Type "run all" on the Tcl prompt

If the test vectors in the test bench were properly revised, the simulation should run to completion, showing that all tests passed (Figure 4-52):

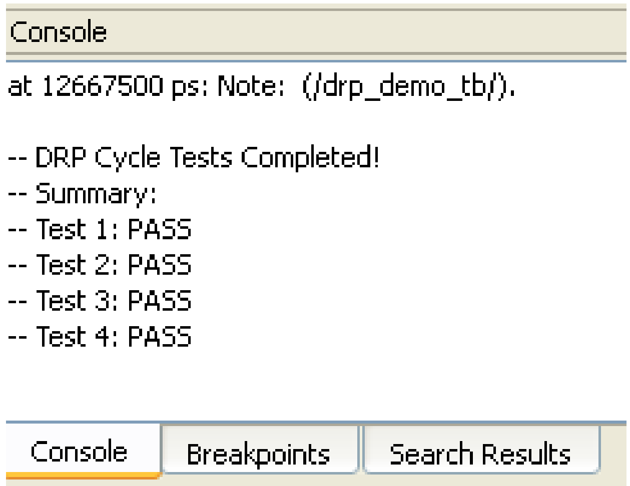


Figure 4-52: **Console Showing That All Tests Passed**

What's Next

This completes the *ISE Simulator (ISim) In-Depth Tutorial*. Refer to the Additional Resources section in the Preface for more detailed information and discussion on the ISE Simulator.