XAPP338 (v2.0) October 30, 2000

Using Xilinx WebPACK and ModelTech ModelSim Xilinx Edition (MXE)

Summary

Xilinx WebPACK™ software is now more powerful than ever with the addition of Model Technology, Inc. (MTI) to this popular EDA tool suite. This application note is designed to quickly show WebPACK users, who are not familiar with MTI, how to utilize this powerful tool within the WebPACK environment.

Introduction

The Xilinx WebPOWERED™ solution provides a complete CPLD implementation environment for today's digital designer. In addition to software, the WebPOWERED solution includes CPLD device evaluation and design conversion tools with proven application notes for Xilinx CPLD devices.

WebPOWERED refers to both WebPACK and WebFITTER™. WebPACK consists of a series of downloadable software modules and WebFITTER is an on-line tool that allows designers to upload their design for device fitting and evaluation. WebPACK and WebFITTER are available at http://www.xilinx.com/products/software/webpowered.htm.

With the addition of MTI ModelSim to WebPACK, designers are now able to functionally simulate their VHDL and Verilog designs as well as perform post-route simulations to verify device timing. The advantage of using an HDL simulator such as MTI ModelSim over other types of simulators is the ability to use HDL testbenches in the simulation and verification processes. HDL testbenches allow designers to behaviorally describe the stimulus for their design, which provides a great deal of flexibility and allows the stimulus generated in the simulation to more accurately represent the system conditions. In addition, this allows the stimulus to vary based on the response received by the device and also allows for concurrent stimulation to different parts of the design. Testbenches can also be defined to compare the results from the device with expected results, eliminating having to pour over waveforms to insure the device is behaving correctly.

This application note will not thoroughly demonstrate the many capabilities of ModelSim, but will focus on the steps necessary to invoke the MTI ModelSim simulator from within WebPACK. More information for running ModelSim should be obtained from MTI and the ModelSim tutorials. The QuickStart Tutorial from ModelSim provides a good introduction to ModelSim for designers.

Please note that this application note will focus on the VHDL flow through Project Navigator and MXE, however, Verilog is also fully supported by these tools. This application note assumes that the reader is experienced in VHDL and/or Verilog and that Verilog users can substitute the appropriate Verilog file and/or terms in this flow where appropriate.

MTI ModelSim Xilinx Edition

MTI ModelSim's inclusion in WebPACK is through an exclusive OEM arrangement between MTI and Xilinx. MTI licenses to Xilinx a special edition of ModelSim called ModelSim Xilinx Edition (MXE). MXE Starter is a free trial version available to Xilinx WebPACK customers that allows up to 500 debuggable lines of code to be run without performance reductions. MTI requests that the user register for a license; which can be applied for at the end of the installation process. Upgrades to ModelSim Personal Edition (PE) and Elite Edition (EE) are available from MTI and both upgrades are compatible with WebPACK.

WebPACK Software Modules

Xilinx WebPACK consists of several software modules for design entry, device fitting, and device programming. Figure 1 shows the WebPACK web page accessed after entering a user name and password. If a single module download is desired, select *Individual Modules* and pick the module to be downloaded. However, to ease the process of obtaining the software modules required, the *Design Configurations* menu automatically selects the software modules needed based on the type of design to be completed.

Figure 1: **WebPACK Software Download Web Page**

Selecting *Design Configurations* brings up the web page shown in Figure 2 which allows the choice of either *CPLD Design* or *FPGA Design*. After choosing *CPLD Design*, the user can choose to download the software modules for all Xilinx CPLDs or select the particular family of CPLDs desired and only download those modules. This application note addresses both 9500 and CoolRunner CPLDs so the *All Xilinx CPLD Devices* menu choice is selected.

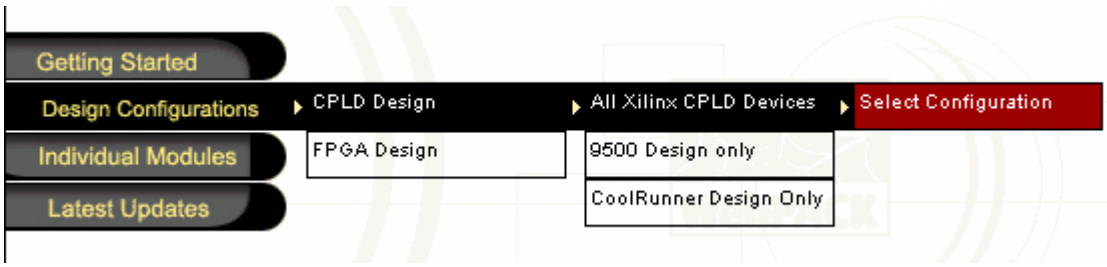

Figure 2: **WebPACK Design Configurations**

The next choice is *Select Configuration* which allows the user to see the modules that will be downloaded based on the design configuration chosen (as shown in Figure 3). Insure that the

MXE Simulator software module has a green check next to it so that it is included in the software modules to be downloaded.

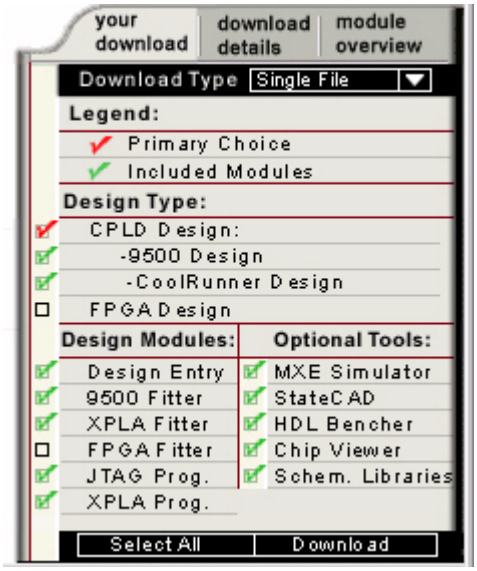

Figure 3: **WebPACK Download Configuration Menu**

Software module download may be accomplished by choosing either *Single File* or *Web Install* by clicking on the arrow near the *Download Type* field as shown in Figure 4.

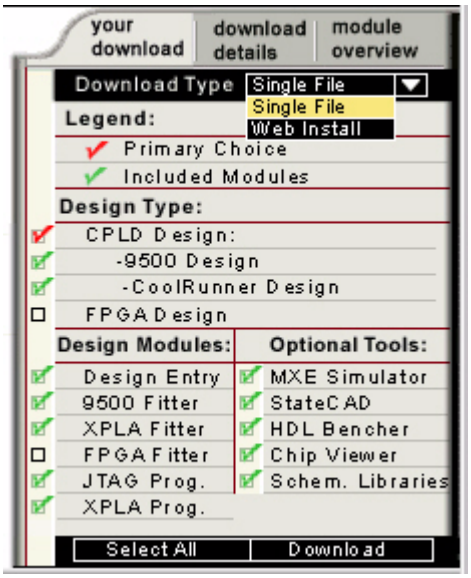

Figure 4: **Single File or Web Install Menu**

Choosing *Single File* as the download type downloads a single compressed installation file to a location specified by the user. This installation file may be run from the saved location at any time. Choosing *Web Install* as the download type downloads 100 Kb partitions while installing the module at the same time. The benefit of *Web Install* is that the install script remembers its location during the download and installation process. In the event data transmission is interrupted, the *Web Install* process will resume from that location once service is restored.

Once the desired software modules and method of download have been selected, clicking the *Download* button invokes the *Download Manager* shown in Figure 5. Each software module should then be downloaded from the web and installed on the user's computer.

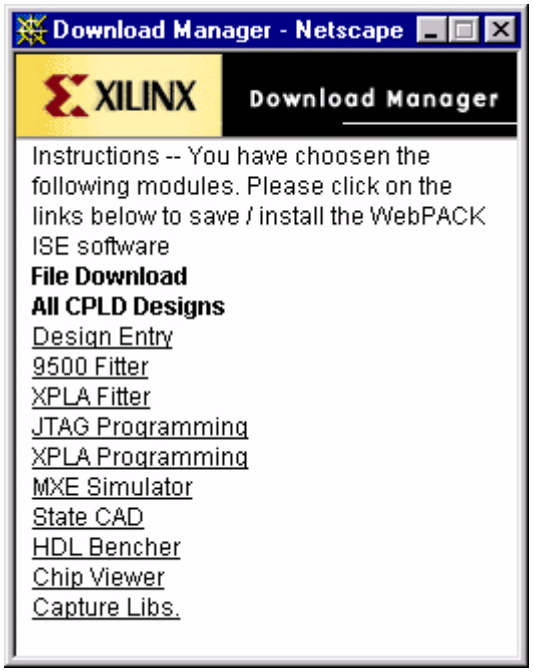

Figure 5: **WebPACK Download Manager**

Obtaining an MXE License

Licenses for MXE can be obtained at the end of the MXE installation process or after the product has been installed. If the user wishes to obtain a license at the end of the installation, the user should select *Yes* when asked whether to complete the license request process following setup as shown in Figure 6.

Figure 6: **Prompt to Request License after MXE Installation Process Completes**

When the ModelSim Setup is complete, the user's web browser will be directed to an online license request form shown in Figure 7.

Figure 7: **Online License Request Form for MXE**

If the user has already installed ModelSim and wishes to request a license, selecting *Start* / *Programs* / *ModelSim XE 5.3a* will bring up the menu shown in Figure 8. Selecting *Submit License Request* will cause the user's web browser to be directed to the online registration form shown in Figure 7.

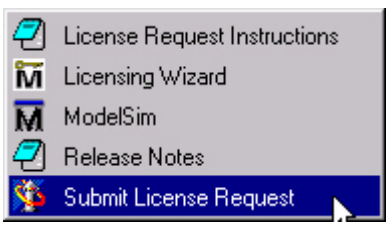

Figure 8: **Accessing the License Request Form Via the Program Menu**

The license for the MXE product will be emailed to the user in a short period of time. This email contains instructions for installing the license. Please note that the menu shown in Figure 8 also provides access to the Licensing Wizard. This valuable tool aids the user in troubleshooting any license issues and is shown in Figure 9.

Figure 9: **MTI License Wizard**

Getting Started

This application note assumes that the user has already downloaded and installed the following WebPACK modules:

- Design Entry Module
- MXE Simulator Module
- XC9500 Fitter Module (for XC9500 CPLD designs)
- CoolRunner XPLA Fitter Module (for CoolRunner CPLD designs)
- JTAG Programmer Module (for XC9500 CPLD designs)
- XPLA Programming Module (for CoolRunner CPLD designs)

The user should have also received via email the registration license from MTI and successfully installed the license per included instructions. The user should also be familiar with the use of WebPACK. The VHDL design code used in this application note can be downloaded from the Xilinx web site at **http://www.xilinx.com/products/xaw/coolvhdlq.htm** by selecting *I²C for XPLA3™*. Unzip this file to a directory and note the location. This application note will assume that the file downloaded from the web site, *xapp333.zip*, has been extracted to a folder named *design_example.*

NOTE:THIRD PARTIES INCLUDING PHILIPS MAY HAVE PATENTS ON THE INTER-INTEGRATED CIRCUIT ("I²C") BUS. BY PROVIDING THIS HDL CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THE I²C BUS IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED

WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGNS "AS IS" AS A COURTESY TO YOU.

Simple Design Example

As with any development of a complex, hierarchical design, most designers start with a simple design module first. This application note will therefore start with the source file addition and simulation of the 4-bit counter used in the I^2C Controller before implementing the entire design.

Start WebPACK Project Navigator by double clicking on the WebPACK icon shown in Figure 10 located on your desktop. (Alternatively, selecting the Xilinx WebPACK Project Navigator from the *Start* | *Programs* | *Xilinx CPLD WebPACK* listing will also launch the WebPACK Project Navigator.)

Figure 10: **Starting Project Navigator from the Desktop Icon**

Once Project Navigator opens, select *File* | *New Project*. Enter *upcnt4* in the *Project Name* text box as shown in Figure 11.This will create a project for simulating the 4-bit counter. Note that in the field *Project Location*, the directory *upcnt4* is specified. WebPACK automatically creates the project directory structure. Therefore, the directory *upcnt4* is created and the project *upcnt4* resides in this directory.

Under *Project Device Options*, the device family, device, and synthesis tool are set for the project. Set the *Device Family* to *Xilinx XPLA3 CPLDs* and the *Device* to *Auto XPLA3.* Choosing *Auto XPLA3* allows the fitter to pick the optimal device density for the design. The *Synthesis Tool* field should be set to *XST VHDL.* Note that if the design source is Verilog, the synthesis tool should be set to *XST Verilog*.

Click OK to close this window.

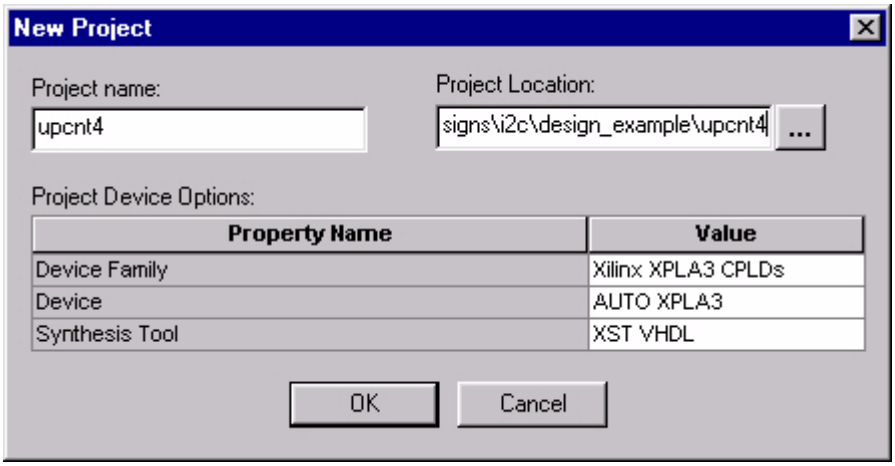

Figure 11: **Creating the Upcnt4 Project**

Having unzipped XAPP333.zip into the directory *design_example* earlier, the file, *upcnt4.vhd* may be imported to this project by clicking *Project* on the Project Navigator menu bar and selecting *Add Source.* In the Add Existing Sources window, navigate to the *design_example*

folder, select *upcnt4.vhd* and click *Open* (Figure 12). Choose *VHDL Module* as the Source File type and the file import into the project will be complete.

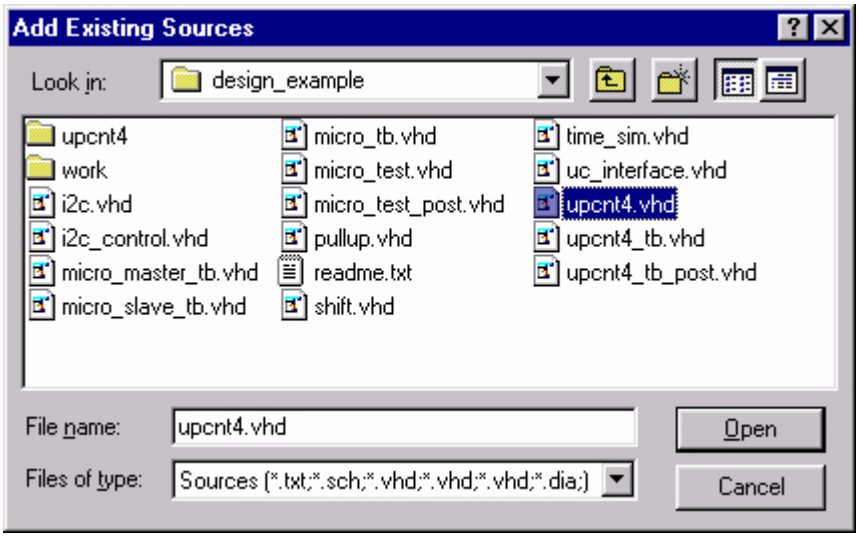

Figure 12: **Adding the *upcnt4.vhd* File to the Project**

Functional Simulation of the 4-bit Counter

To verify the functionality of the 4-bit counter, the MTI simulator can be used to simulate the design. A VHDL testbench is necessary to define the stimulus for the design and in some cases the expected output. The integration of the MTI simulator into the Project Navigator allows VHDL testbenches to be imported into the project. In this design example, the testbench for the 4-bit counter (*upcnt4_tb.vhd*) has been created using HDLBencher. HDLBencher from Visual Software Solutions, Inc. helps verify design functionality by creating a testbench from the specified stimulus and expected results. Information about HDLBencher can be obtained from http://www.testbench.com.

> **NOTE:** It is very important to view the online help information titled *ModelSim Considerations*. This document contains information to aid in the development of designs and testbenches for use with MXE for XC9500 and CoolRunner CPLD designs. This file can be found by choosing *Start* | *Programs* | *Xilinx WebPACK* | *Help and Technical Support* from the desktop task bar. Once the Help system is running, select the *Contents* tab and expand the selections under *CPLD WebPACK ISE*. Expand *Tutorials* and select *Creating a New VHDL Testbench File*. Follow the steps in this tutorial and take the link under step 4 titled *ModelSim Considerations*.

Using the same steps described to import the VHDL source file, import the testbench file, *upcnt4_tb.vhd*, into the project. This time, however, choose *VHDL TestBench* as the source file type as shown in Figure 13 and click *OK*.

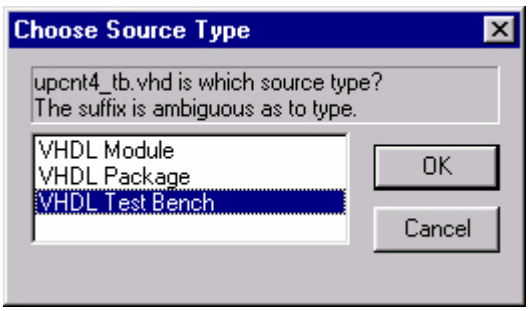

Figure 13: **Specifying UPCNT4_TB.VHD as VHDL TestBench**

Note that for Verilog designs, Project Navigator does not display this window. The Verilog testbench or test fixture file **must** have a ".tf" extension to be imported into Project Navigator.

Now that the testbench has been imported into the project, the properties of the MTI Simulator need to be set. Select *upcnt4_tb.vhd* in Source window of Project Navigator. The Process window now shows the functional and post-route simulation processes of the MTI Simulator as shown in Figure 14.

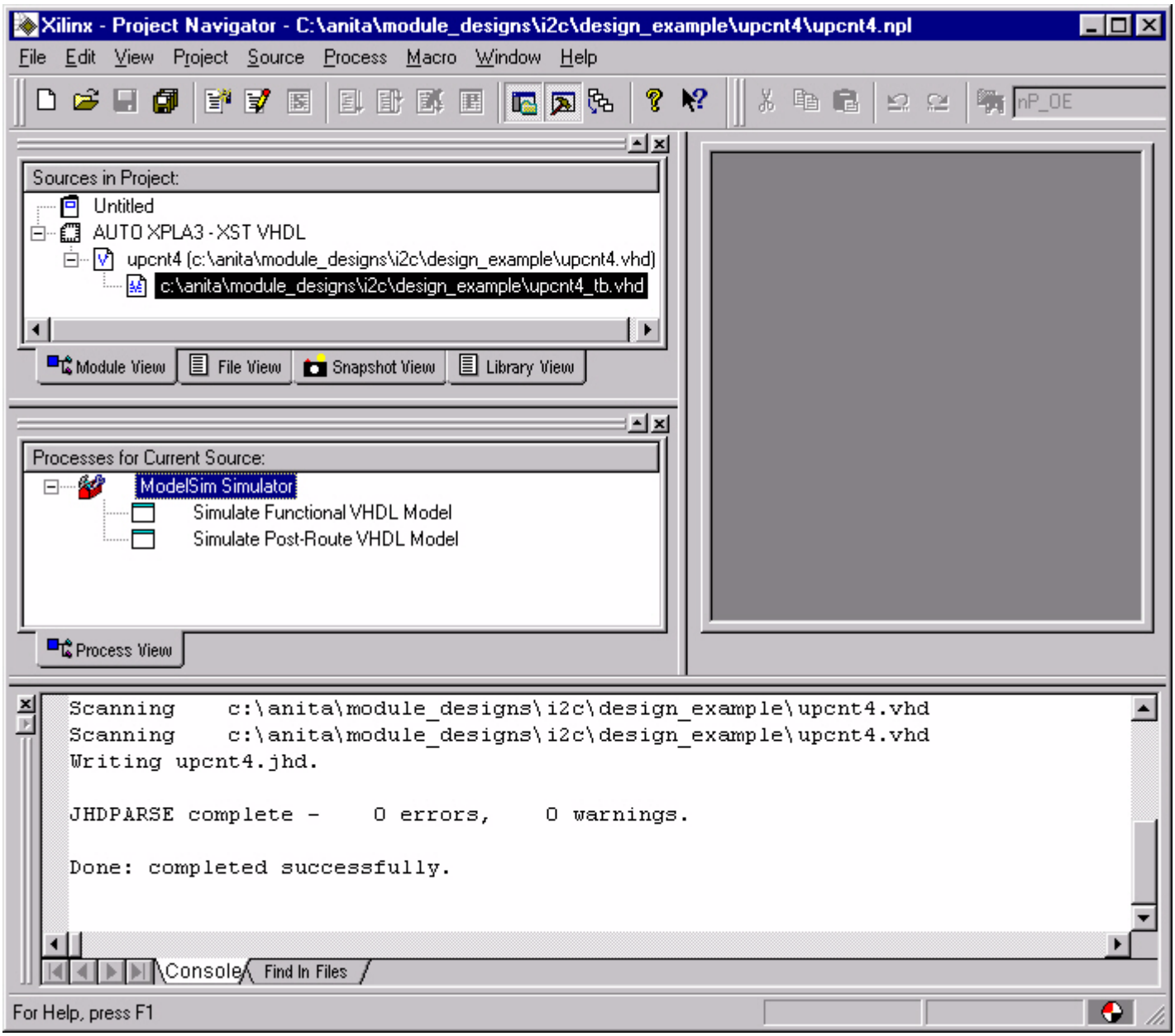

Figure 14: **ModelSim Simulator Processes**

Right click on *Simulate Functional VHDL Model*. Select the *Properties* option. A Process Properties options box should appear as shown in Figure 15. In the *VHDL Functional Simulations Options* tab, ensure the following:

- *Pre VSim Do File* value is empty
- *Use Automatic Do File* box is checked
- *Simulation Run Time* value is changed to 10,000 ns (10 µs)
- *Design Unit Name* is set to *testbench*. This corresponds to the top level entity in the test bench file.

Next, select the *Display Options* tab to select the ModelSim Simulator windows to be displayed during the simulation. For this design example, check the *Signal*, *Structure* and *Wave* windows. When these fields have been modified, select *OK*.

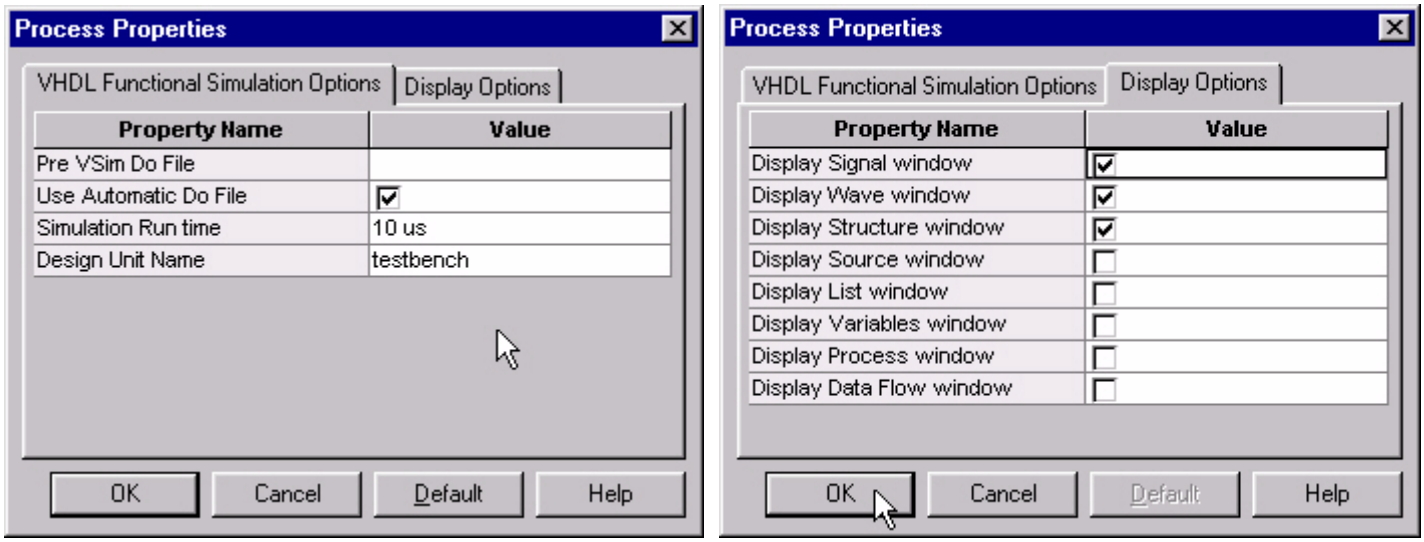

Figure 15: **ModelSim Simulator Properties**

The ModelSim Simulator is now configured and ready for simulation. In the Project Navigator Process Window, double click on *Simulate Functional VHDL Model*. Choose *Run ModelSim Simulator* if the Model Technology menu appears. The ModelSim command window shown in Figure 16 will open and show the progress of the design simulation.

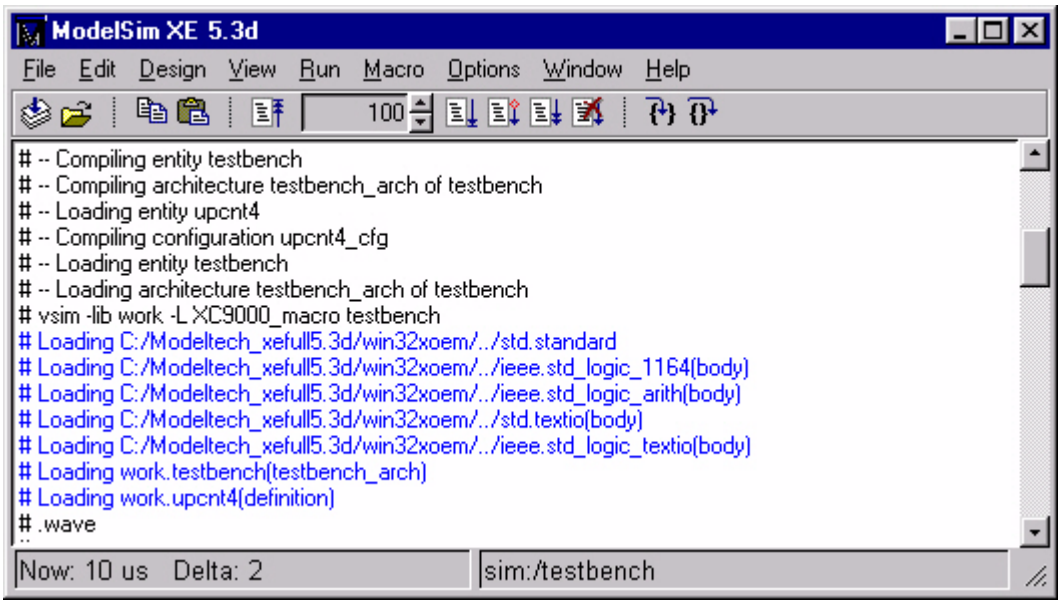

Figure 16: **ModelSim Command Window**

The other selected ModelSim windows will appear as well. Verify the proper operation of the design by selecting between the windows and zooming in on active sections in the Wave

window. Figure 17 shows the windows resulting from the simulation. Close ModelSim by selecting *File* / *Quit* in the ModelSim Command window.

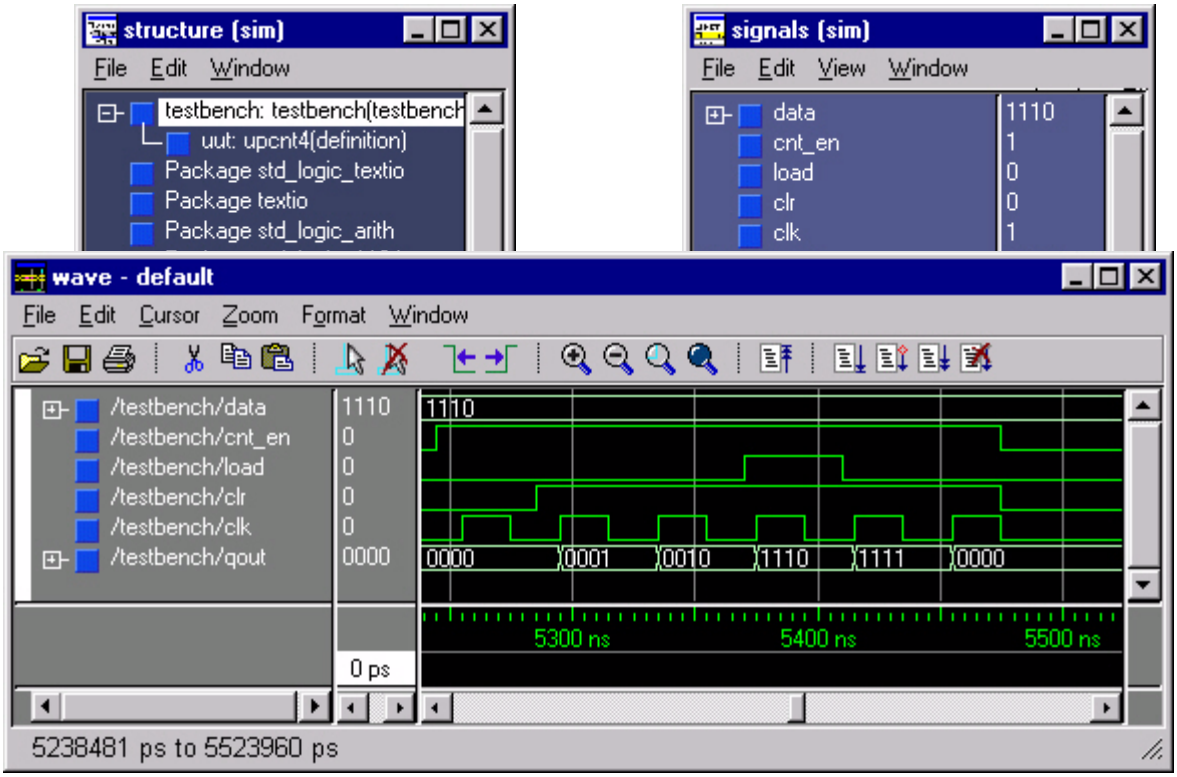

Figure 17: **ModelSim Structure, Signal, and Wave Windows**

Post-Route Simulation of 4-bit Counter

The post-route simulation processes for XC9500 CPLDs and CoolRunner CPLDs vary in the way the post-route model is created. For XC9500 CPLDs, a delay file with the target device timing (*time_sim.sdf*) and a VHDL file that calls primitives within the Xilinx library (*time_sim.vhd*) are created. These two files together form a complete post-route device model of the design that is simulated in ModelSim. The user can choose to have a post-route Verilog model created instead of the VHDL model. The Verilog file will be named *time_sim.v.*

For CoolRunner CPLDs, a VHDL model is created that contains models for all primitives and delay timing. Also named *time_sim.vhd,* this file is self-contained and provides a complete post-route device model for simulation in Modelsim. Again, the user can choose to have a post-route Verilog model (*time_sim.v)* created instead of the VHDL model.

The following sections detail the post-route simulations of the 4-bit counter design for a XC9500 implementation and a CoolRunner CPLD implementation.

> **NOTE:** It is very important to view the online help information titled *ModelSim Considerations.* This document contains information to aid in the development of designs and testbenches for use with MXE for XC9500 and CoolRunner CPLD designs. This file can be found by choosing *Start* | *Programs* | *Xilinx WebPACK* | *Help and Technical Support* from the desktop task bar. Once the Help system is running, select the *Contents* tab and expand the selections under *CPLD WebPACK ISE*. Expand *Tutorials* and select *Creating a New VHDL Testbench File*. Follow the steps in this tutorial and take the link under step 4 titled *ModelSim Considerations*.

XC9500 CPLD Post-route Simulation

In order to accomplish a XC9500 post-route simulation, the design must first be fit into a XC9500 CPLD. Double clicking on *Auto XPLA3 - XST VHDL* (or right clicking and selecting *Properties*) allows the target device to be modified. In the Project Properties window, select *Xilinx 9500XL CPLDs* for the *Device Family* and *Auto 9500XL* as the *Device* as shown in

Figure 18. Choosing *Auto 9500XL* allows the fitter to pick the optimal device density for the design. Click *OK* to close the window.

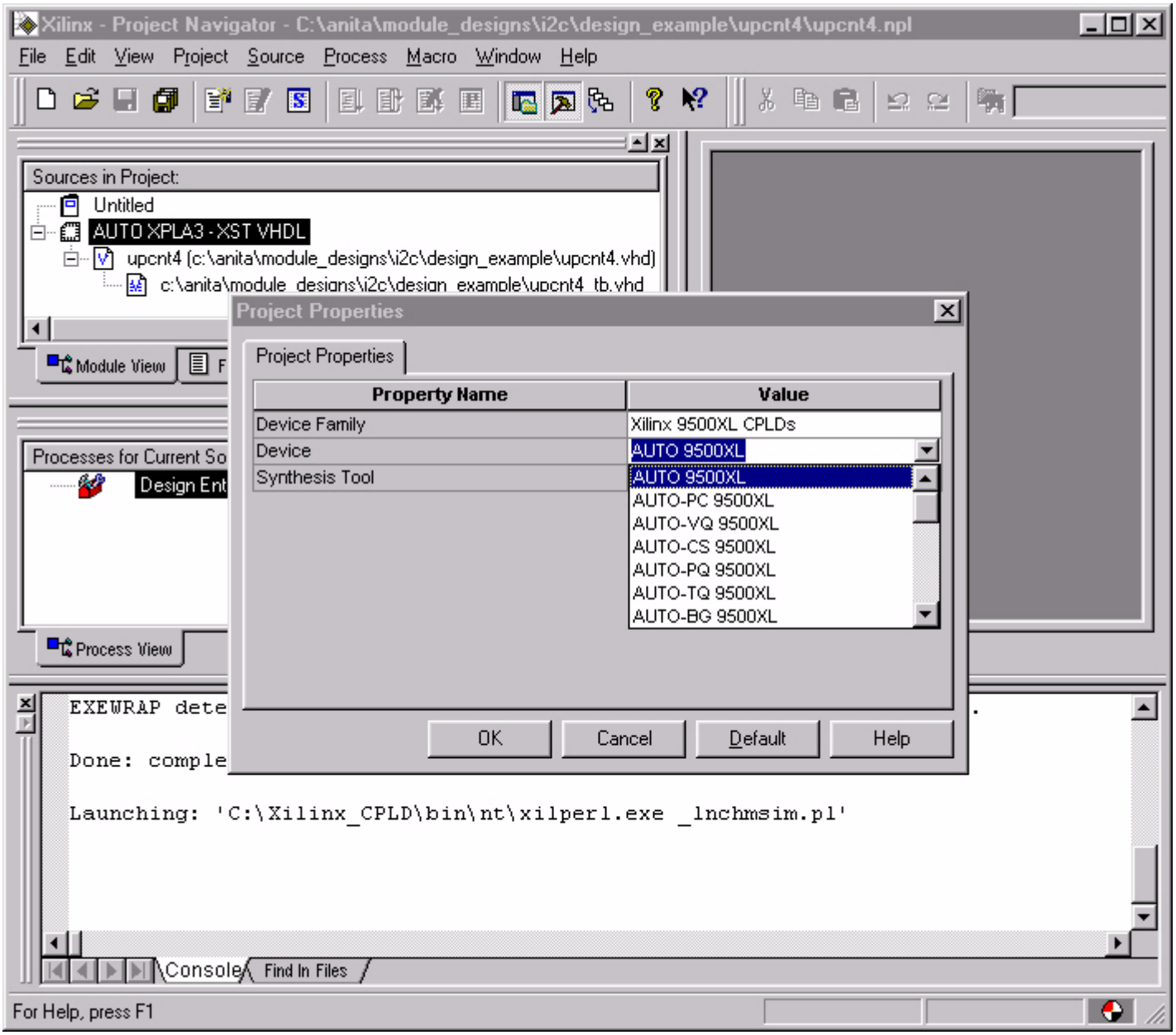

Figure 18: **Selecting a XC9500 Family CPLD**

The fitter must be instructed to produce a VHDL timing model of the design by setting fitter properties. Select *upcnt4.vhd* in the Project Navigator Source window. This displays various processes and utilities in the Project Navigator Process window. Right-click on the *Implement Design* process and select *Properties.* This will open the Process Properties window for the fitter process. Click the *Timing Simulation* tab and select *VHDL* in the *Output File Format* field as shown in Figure 19. Note that there is a menu choice to have the post-route model written in Verilog. The VHDL timing model will be named *time_sim.vhd* which is the file name expected by

the ModelSim simulator. The Verilog timing model will be named *time_sim.v* which is the file name expected by the ModelSim simulator. Click *OK* to close the window.

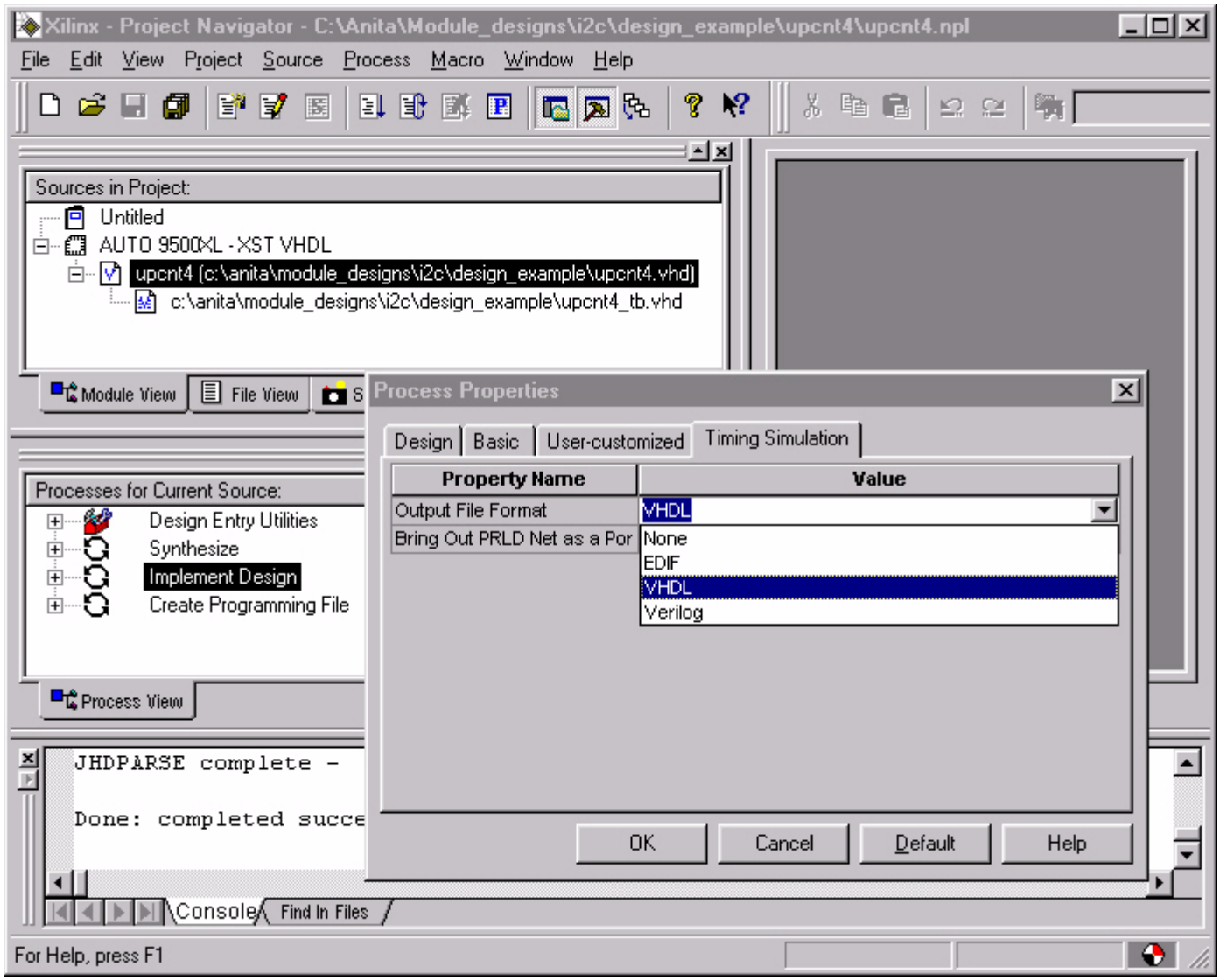

Figure 19: **Specifying VHDL Output Model for XC9500 Fitter**

To start the fitting process, double-click on *Implement Design* in the Project Navigator Process Window. As the design is fit into the device, the progress is displayed in the bottom portion of the Project Navigator Window. When the fitter process has successfully completed, a green check mark appears in front of *Implement Design* in the Project Navigator Process window.

The files *time_sim.sdf* and *time_sim.vhd* have now been created and a post-route simulation can be performed. Select *upcnt4_tb.vhd* in the Project Navigator Source window. Right-click on *Simulate Post-route VHDL Model* and select *Properties.* In the Process Properties window, select the *VHDL Timing Simulations Options* tab to set the parameters for the post-route

simulation as shown in Figure 20. The *Display Options* will be left at the default values which are the same settings as in the functional simulation. Click *OK* to close the window.

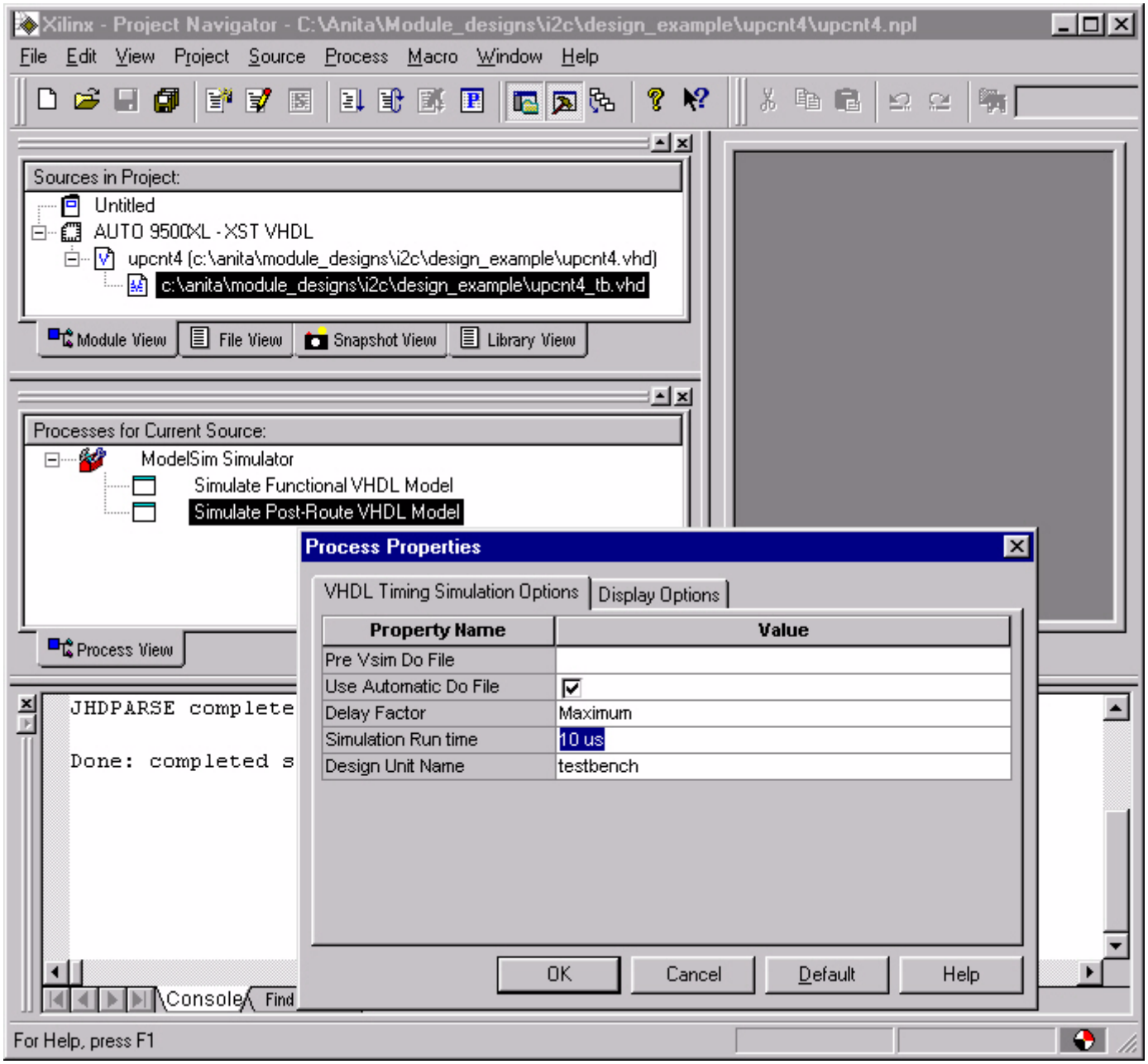

Figure 20: **Setting Post-Route Simulation Parameters**

Start the post-route simulation by double-clicking on *Simulate Post-Route VHDL Model.* Select *Run Model Sim* if the Model Technology menu is displayed. The ModelSim Command window is displayed showing the progress as the design is loaded into the simulator. The specified windows are opened and the simulation runs for 10 µs. As shown in Figure 21, the cursors in the ModelSim Wave window can be used to measure the delay from the clock to one of the

counter register outputs. Select *File* / *Quit* from the ModelSim Command window to exit ModelSim.

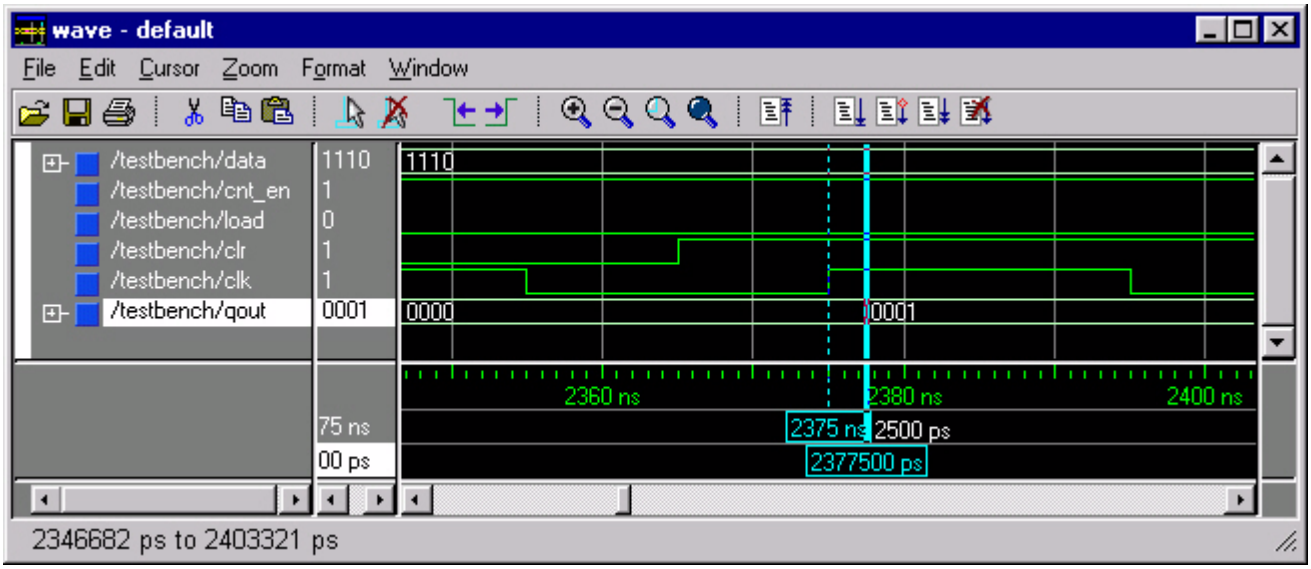

Figure 21: **XC9500 CPLD Post-Route Simulation Results**

Post-Route Simulation for CoolRunner CPLDs

To perform a post-route simulation for CoolRunner CPLDs, the design must be targeted to a CoolRunner device. Double clicking on *Auto 9500XL - XST VHDL* (or right clicking and

selecting *Properties*) opens the Project Properties window for device selection. Select *Xilinx XPLA3 CPLDs* for the *Device Family* and *Auto XPLA3* for the *Device* as shown in Figure 22.

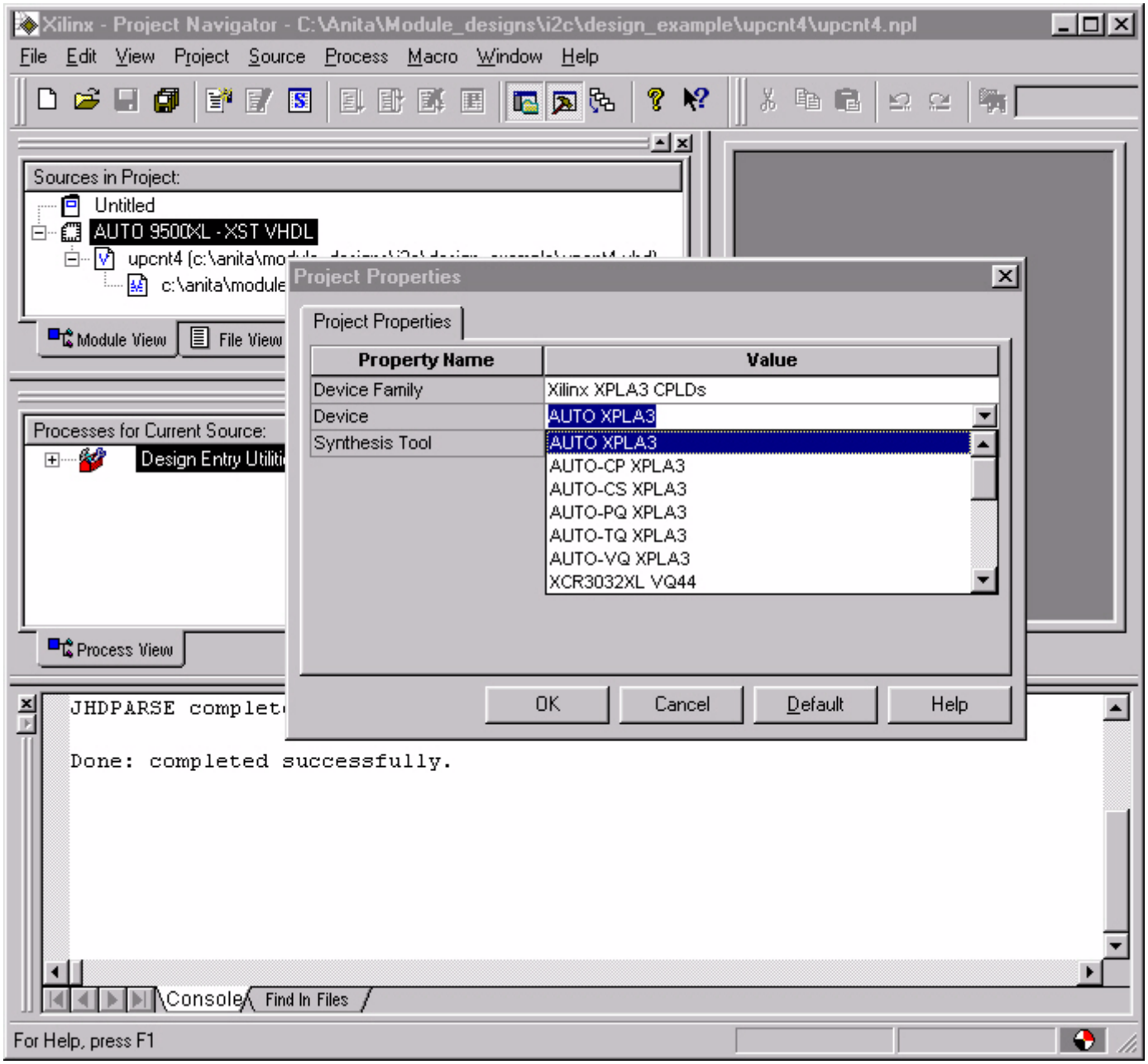

Figure 22: **Selecting a CoolRunner XPLA3 CPLD**

The fitter must be instructed to produce a VHDL timing model of the design by setting fitter properties. Select *upcnt4.vhd* in the Project Navigator Source window. This displays various processes and utilities in the Project Navigator Process window. Right-click on the *Implement Design* process and select *Properties.* This will open the Process Properties window for the fitter process. Click the *Timing Simulation* tab and select *VHDL* as the *Timing Model Output Type* as shown in Figure 23. The timing VHDL model will be named *time_sim.vhd* which is the file name expected by the ModelSim simulator. For a Verilog design flow, the timing Verilog

model will be named *time_sim.v* which is the file name expected by the ModelSim simulator. Click *OK* to close the window.

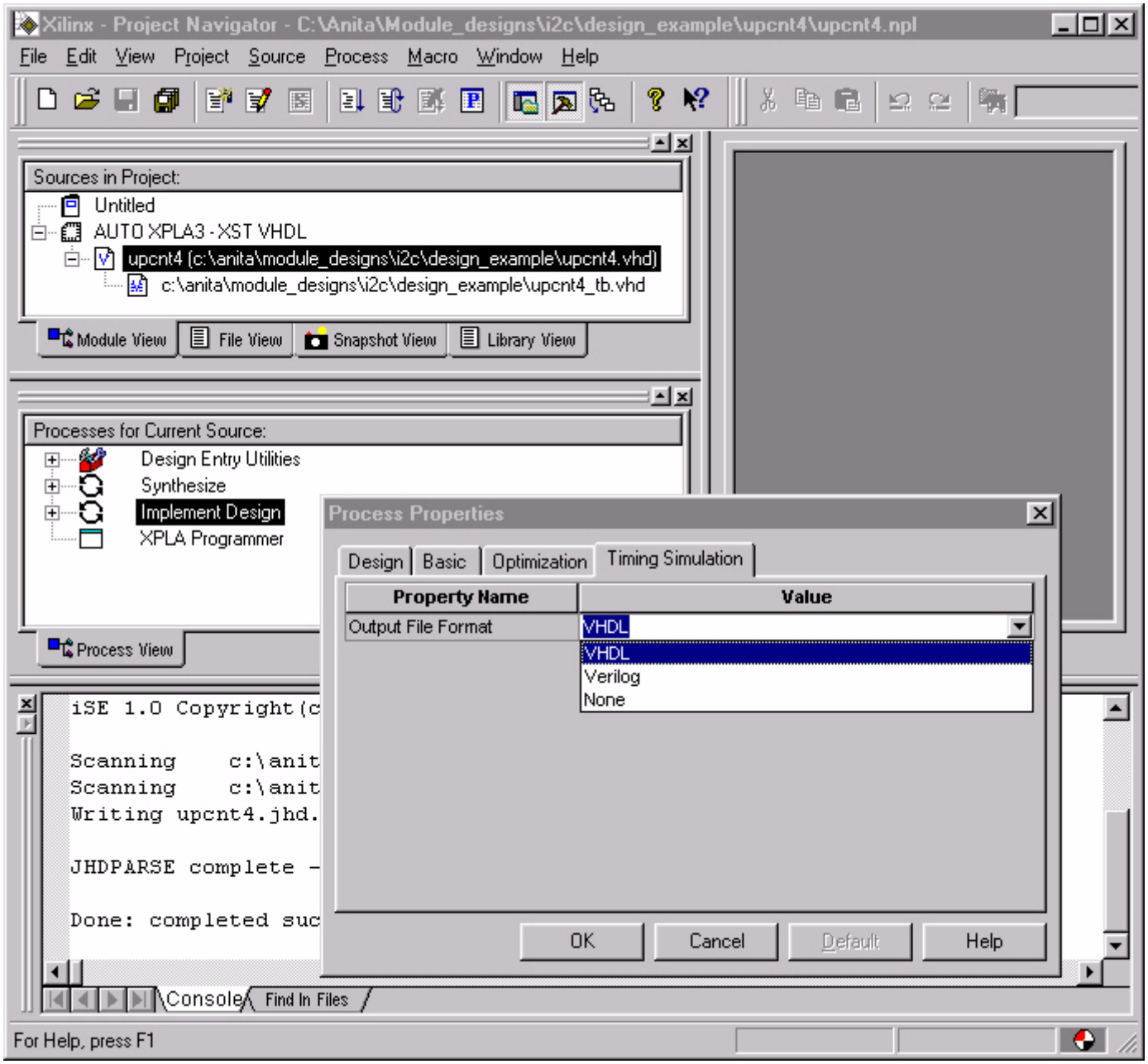

Figure 23: **Specifying VHDL Output Model for CoolRunner Fitter**

To start the fitting process, double-click on *Implement Design* in the Project Navigator Process Window. As the design is fit into the device, the progress is displayed in the bottom portion of the Project Navigator Window. When the fitter process has successfully completed, a green check mark appears in front of *Implement Design* in the Project Navigator Process window. The file *time_sim.vhd* has now been created.

Select *upcnt4_tb.vhd* in the Project Navigator Source window. Then right-click on *Simulate Post-route VHDL Model* and select *Properties.* In the Process Properties window, select the *VHDL Timing Simulations Options* tab to set the parameters for the post-route simulation as

shown in Figure 24. The *Display Options* will be left at the default values which are the same settings as in the functional simulation. Click *OK* to close the window.

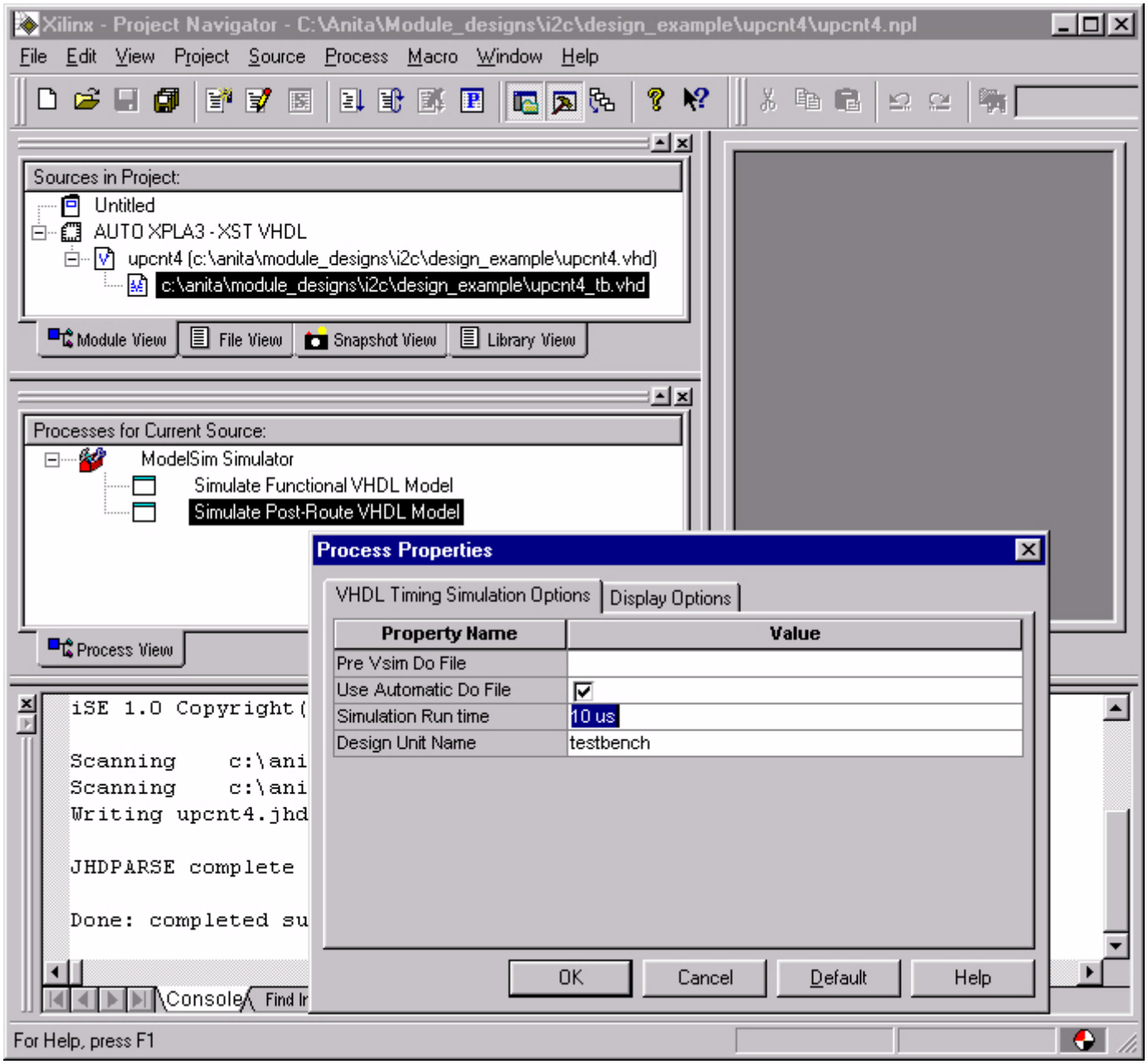

Figure 24: **Setting CoolRunner CPLD Post-Route Simulation Parameters**

Start the post-route simulation by double-clicking on *Simulate Post-Route VHDL Model.* Select *Run Model Sim* if the Model Technology menu is displayed. The ModelSim Command window is displayed showing the progress as the design is loaded into the simulator. The specified windows are opened and the simulation runs for 10 µs. As shown in Figure 25, the cursors in the ModelSim Wave window can be used to measure the delay from the clock to one of the

counter register outputs. Select *File* / *Quit* in the ModelSim Command window to close ModelSim.

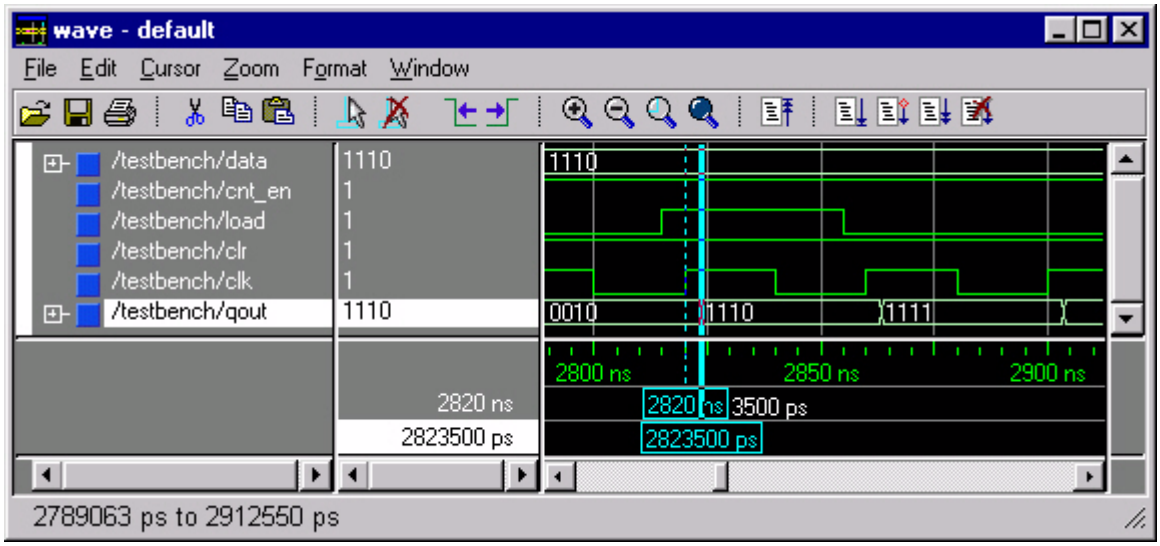

Figure 25: **CoolRunner CPLD Post-Route Simulation Results**

Complex Design Example

The strength of the WebPOWERED tool suite can be seen through the creation of the I²C Bus Controller design for a CoolRunner CPLD. The design of the I²C Bus Controller using a CoolRunner CPLD is described in Xilinx Application Note XAPP333 and can be downloaded from http://www.xilinx.com/xapp/xapp333.pdf.

THIRD PARTIES INCLUDING PHILIPS MAY HAVE PATENTS ON THE INTER-INTEGRATED CIRCUIT ("I²C") BUS. BY PROVIDING THIS HDL CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THE I²C BUS IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGNS "AS IS" AS A COURTESY TO YOU.

Note: The I²C design contains more than 500 lines of executable VHDL code, therefore, the full MXE product (not MXE Starter) is required to simulate this design.

Opening the I²C Project

To simulate the full I²C design, the I²C project contained in *XAPP333.zip* must be opened in Project Navigator. This application note will assume that *XAPP333.zip* has been extracted to a folder named *design_example*. Select *File* / *Open Project* from the Project Navigator menu bar

and select *i2c.npl*. This project contains all of the VHDL source code necessary for implementing the design in a CoolRunner CPLD.

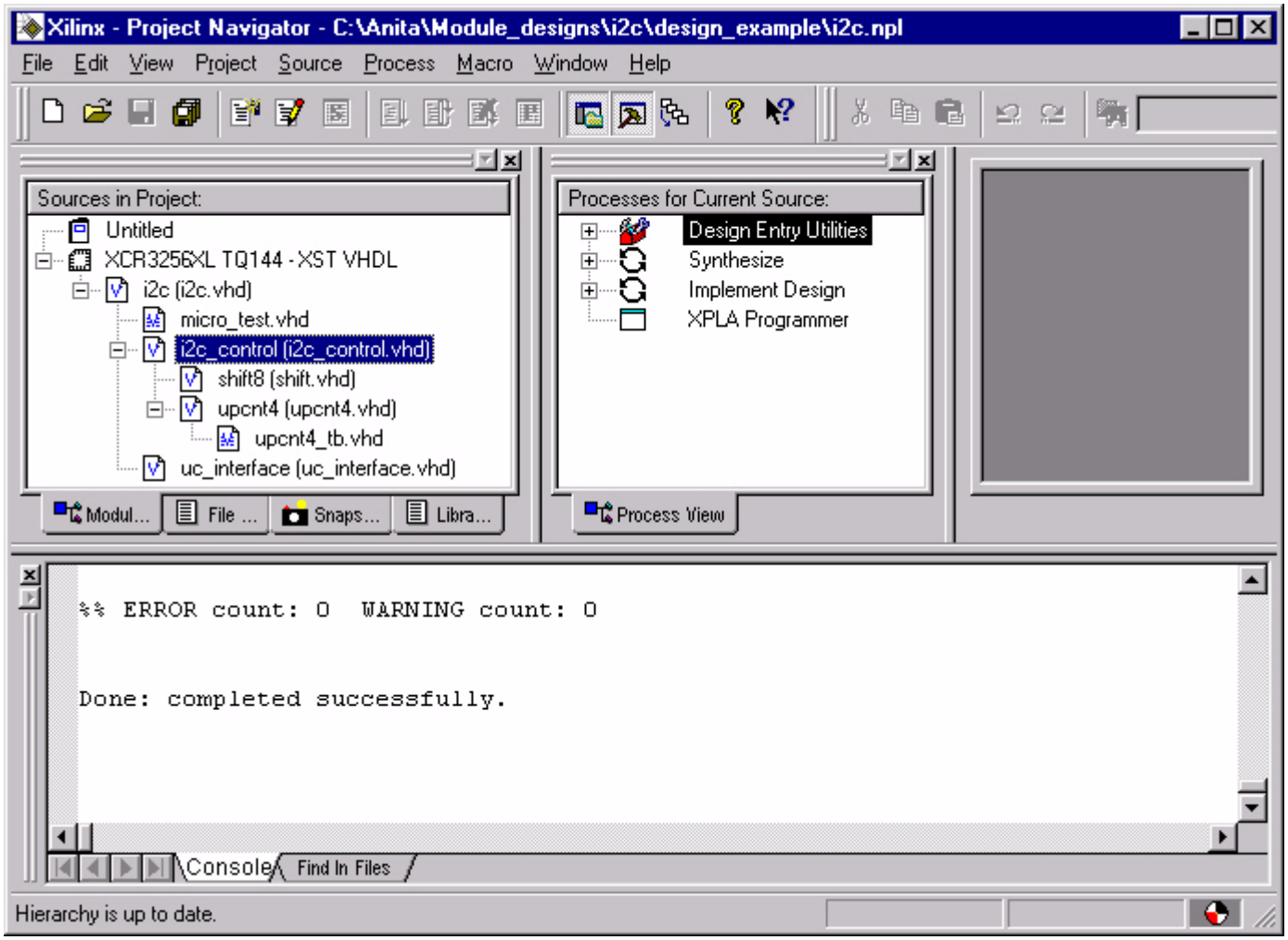

Figure 26: **I²C Project**

I²C Design Functional Simulation

Note that as shown in Figure 26, the functional simulation testbench for the I²C design (micro_test.vhd) has already been imported into the project. Also note that testbenches can be imported into the project for simulating VHDL modules throughout the design hierarchy. The testbench for the 4-bit counter can be imported in the project and the simulations performed in the previous sections of this application note can be executed from this project.

Also note that the WebPACK Process window can be moved within the WebPACK GUI by dragging the double bar at the top of the window to the desired location. In Figure 26, this window has moved to the center of the GUI for better visibility.

The testbench, *micro_test.vhd*, instantiates a file that emulates the microcontroller interface to the I²C design and instantiates two of the I²C designs. The microcontroller model configures one of the I²C designs as the I²C master and configures the other I²C design as the I²C slave.The microcontroller then writes data (DE, AD, BE, EF, FA, DE) to the I²C master for transmission to the I²C slave. It then writes the same data to the slave and instructs the master to read this data. Note that the underlying VHDL models and components used in the functional testbench, *micro_test.vhd*, have been previously compiled in the ModelSim simulator. Please refer to the ModelSim documentation and tutorials for information on how to compile designs.

ModelSim allows the settings and commands of a simulation to be saved in a script file, typically called a ".do" file. This allows the user to configure the wave window, for example, with the signals to be viewed and set colors for these signals. It also allows for a sequence of commands from the ModelSim command windows to be executed in a particular order. Simulation of the I^{2}C design will be done with predefined DO files. Please refer to the ModelSim documentation and help files for more information on DO files.

To simulate the I^{2}C design with the predefined file, *micro_test.do*, select the testbench, *micro_test.vhd* in the Project Navigator source window. Right-click on *Simulate Functional VHDL Model* and select *Properties.* With the *VHDL Functional Simulation Options* tab selected, enter *micro_test.do* in the *Pre VSIM Do File* field. Un-check the *Use Automatic Do File* box as shown in Figure 27. Since the simulation entity, run time, and viewable windows are all defined in the file, *micro_test.do*, the *Simulation Run Time* field and the fields under the *Display Options* tab can be left with the default entries. This testbench has the entity name of *micro_test,* therefore, *micro_test* is entered as the *Design Unit Name.* Select *OK* to close the window.

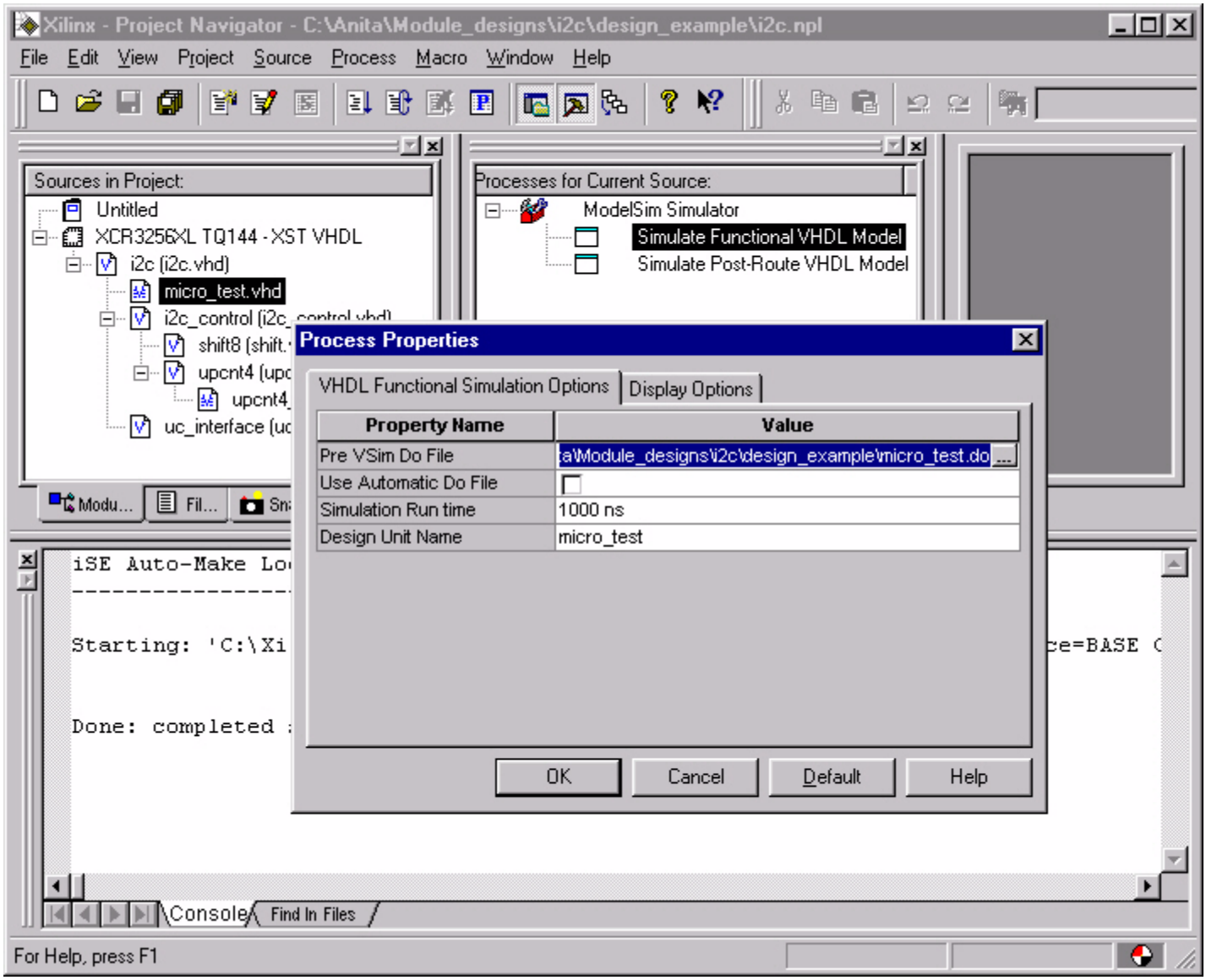

Figure 27: **Functional Simulation Properties Using Predefined DO file and Testbench Entity Name**

The I^{2}C design is now simulated by double-clicking *Simulate Functional VHDL Model* in the Project Navigator Process window. Choose *Run ModelSim* if the Model Technologies menu

appears. Again note that the ModelSim Command window displays the status of loading and simulating the design. The file, *micro_test.do*, only displays the ModelSim Wave window, but has placed the signals in a desired order and has applied formatting (color, radix) to the signals for easier debugging. This is shown in Figure 28.

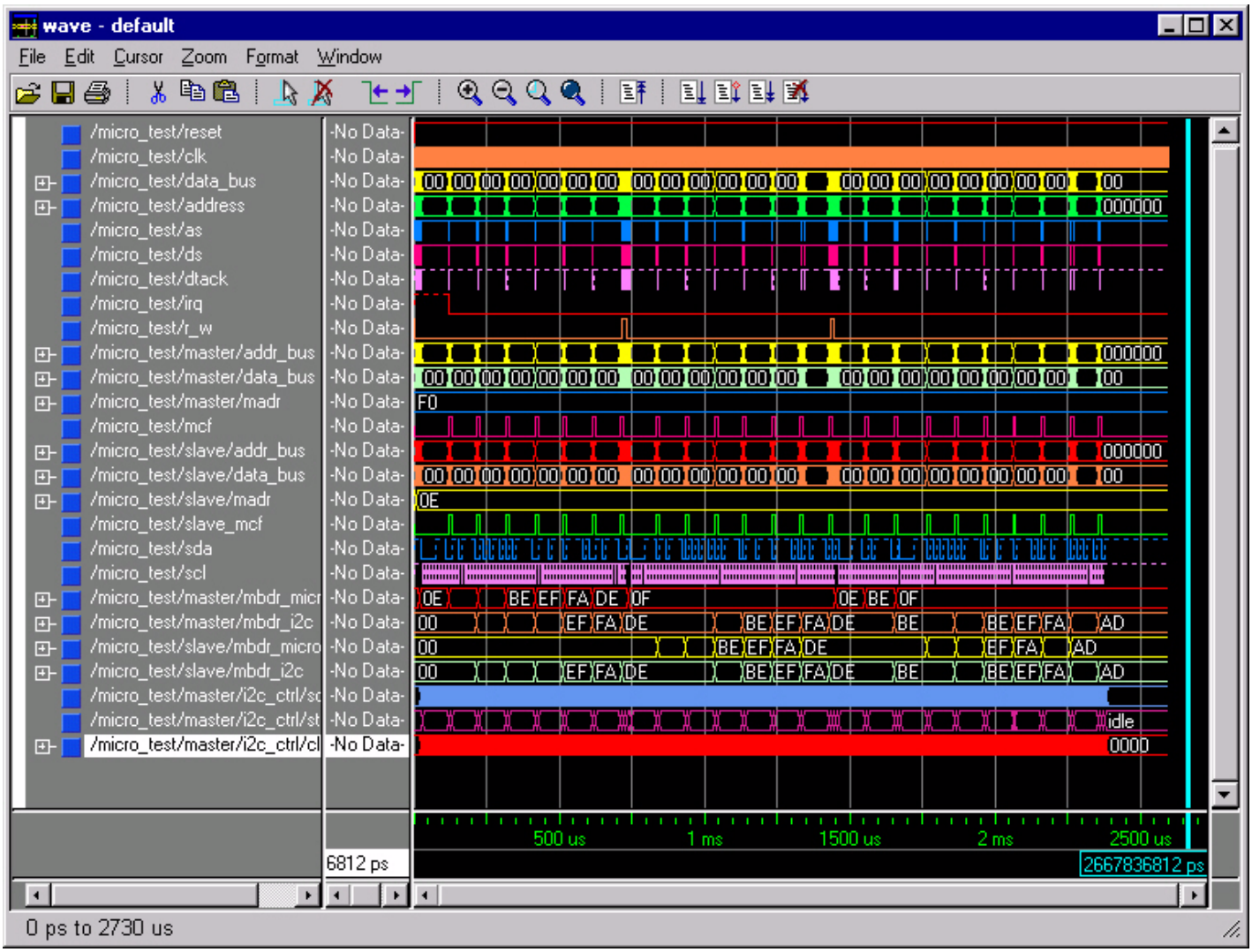

Figure 28: **ModelSim Wave Window for I²C Design**

As seen in Figure 28, the microcontroller writes data to the master for transmission on the I²C bus after it has configured the I²C Master and Slave. This is seen in the Master's MBDR_MICRO signal. As this data is transmitted across the I²C bus, it is captured by both the Master and the Slave and is shown in both the Master's and the Slave's MBDR_I2C signals. When the simulation changes so that the Master reads data from the Slave, the microcontroller writes the data to be read in the Slave's MBDR_MICRO register. The Slave's MBDR_MICRO waveforms reflect the data written by the microcontroller and this data is captured by both the Master and Slave's MBDR_I2C signals as this data is transferred across the I²C bus.

Select *File* / *Quit* in the ModelSim Command window to close ModelSim.

I²C Design Post-Route Simulation

After fitting the I²C design into a CoolRunner CPLD, a post-route VHDL model is created for use in post-route simulations for final verification of the design with device timing. This file is named *time_sim.vhd* and contains all primitive models and timing.

The testbench file, *micro_test_post.vhd*, is similar to the testbench file, *micro_test.vhd*, except that it instantiates two versions of the microcontroller model. In the functional simulation, generics were used to specify the address on the microcontroller bus for the I^2C master and I^2C slave CPLDs. In the post-route version of the I^2C VHDL model, however, the code is written in a very structural sense and doesn't allow for the passing of generics. Therefore, one microcontroller is used to configure one instantiation of the CPLD code as the I^2C Master and the other is used to configure the other instantiation of CPLD code as the I^2C Slave.

This testbench still writes the same data pattern (DE, AD, BE, EF, FA, DE) to the I^2C Master for transmission to the Slave and then writes this data pattern to the Slave to be read by the Master. If this file is not in the project, import this file to the project as described in previous sections, associating it with the source file *i2c* as shown in Figure 29.

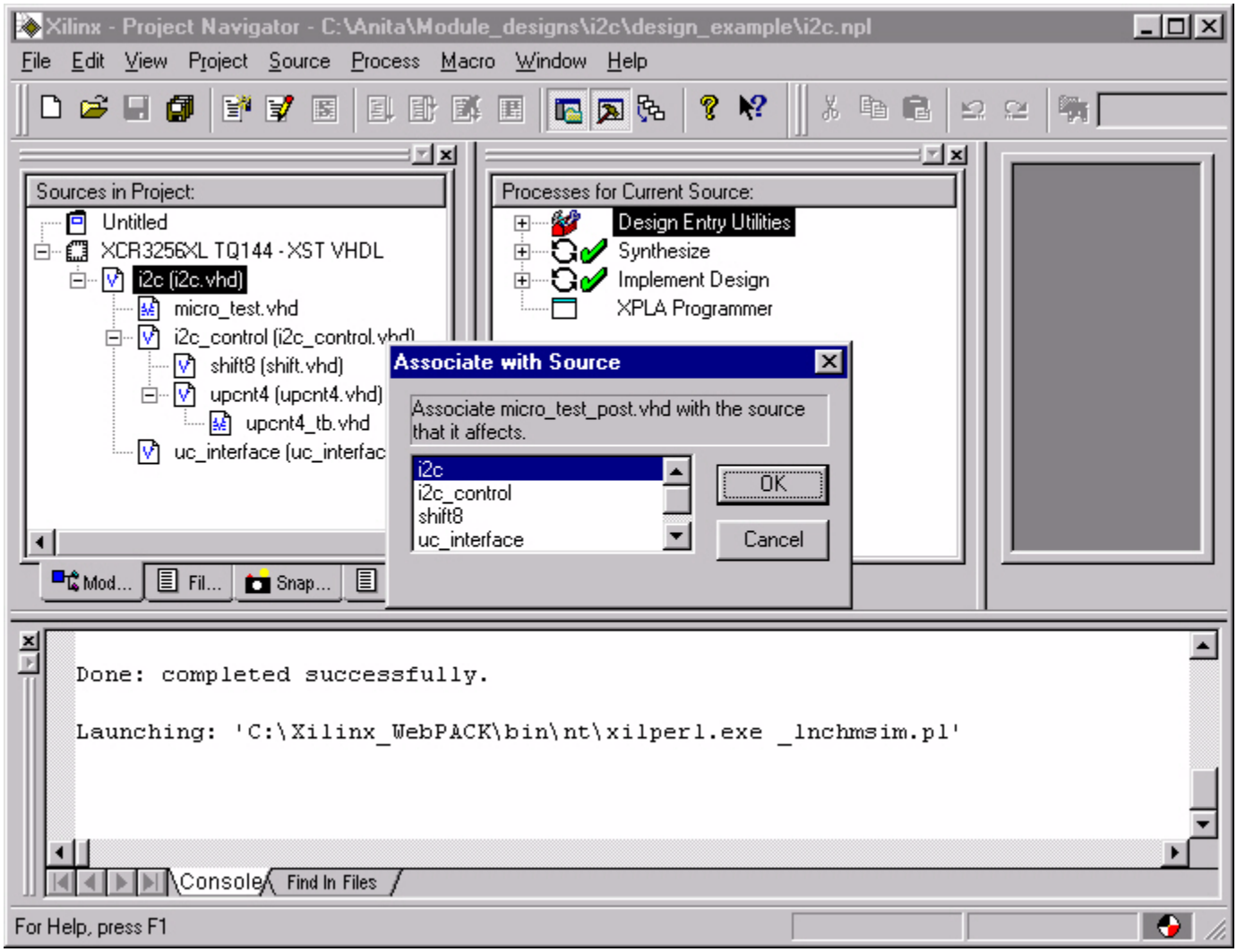

Figure 29: **Adding *micro_test_post.vhd* Testbench**

As in the functional simulation, predefined DO files are used in simulating the post-route VHDL file. To set the simulator properties, select *micro_test_post.vhd* in the Project Navigator Source window. The same two ModelSim Simulator processes, *Simulate Functional VHDL Model* and *Simulate Post-route VHDL Model* appear in the Project Navigator Process window. Right-click on *Simulate Post-route VHDL Model* to open the Simulator Process Properties window. Again, since a predefined DO file will be used, enter *micro_test_post.do* in the *Pre Vsim Do File* field and un-check the *Use Automatic Do File* box. The entity name in the file *micro_test_post.vhd* is

micro_test_post, therefore, *micro_test_post* is entered as the *Design Unit Name* as shown in Figure 30.

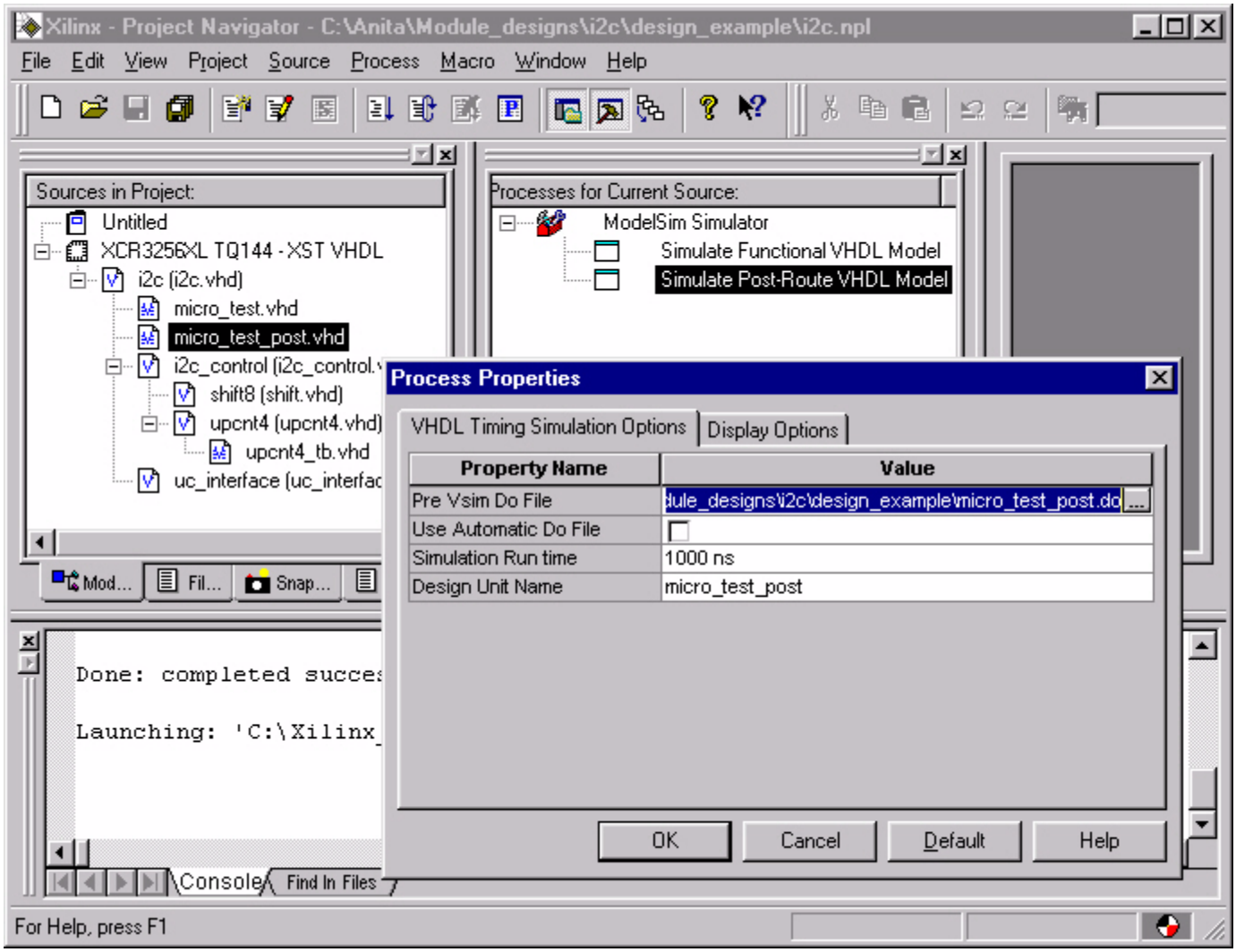

Figure 30: **Specifying the Post-Route DO File and Testbench Entity Name**

Note that to actually run the simulation, the underlying testbench files must have been previously compiled in ModelSim. Please refer to ModelSim tutorials on how to compile designs.

To run the post-route simulation, double-click on *Simulate Post-Route VHDL Model.* Choose *Run ModelSim* if the Model Technologies menu appears. The specified DO file instructs

ModelSim to display the Wave window, and formats and colors the data for easy debugging. The simulation results are shown in Figure 31.

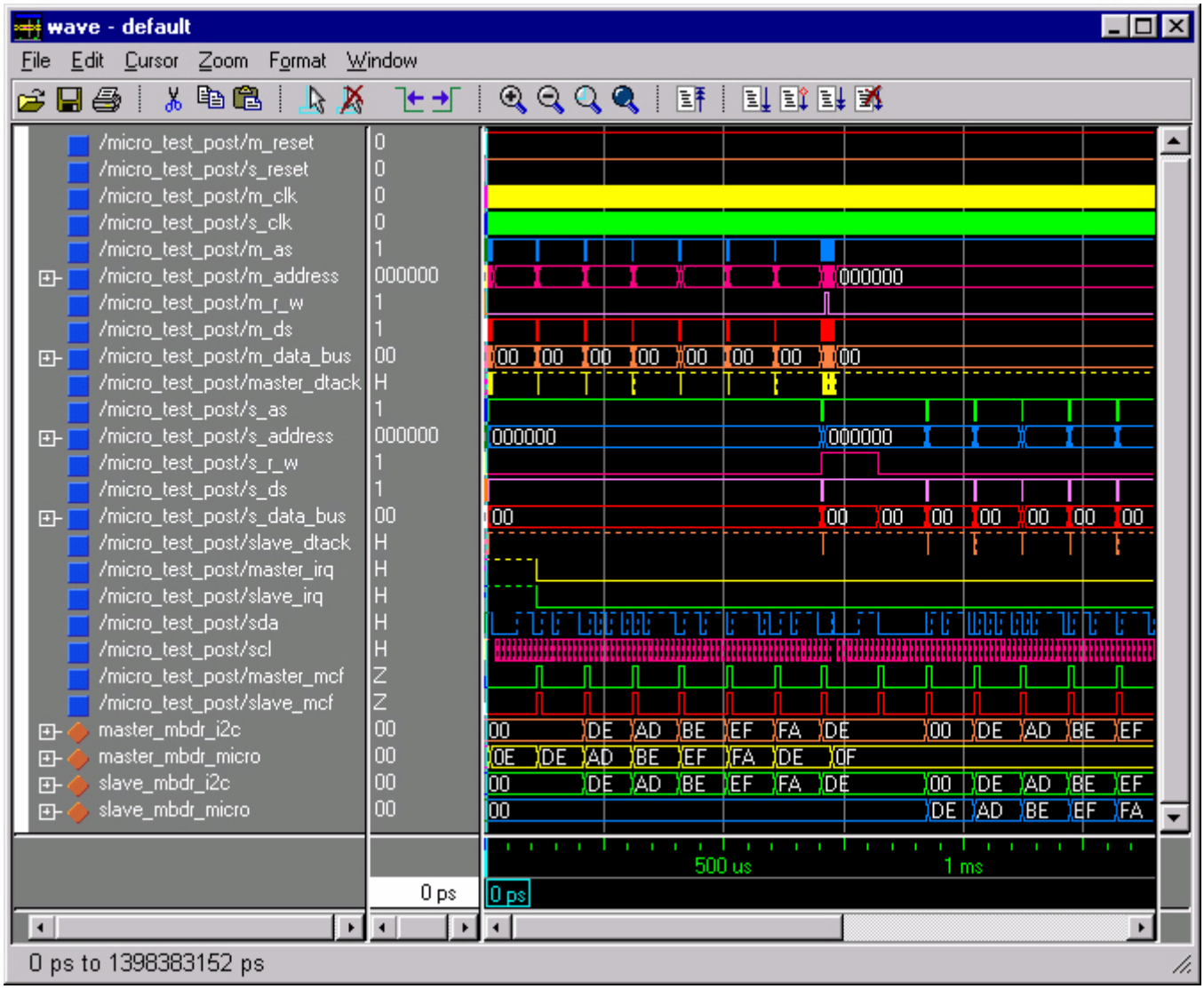

Figure 31: **I²C Post-Route Simulation Results**

As seen in Figure 31, the microcontroller writes data to the master for transmission on the I²C bus after it has configured both I²C designs. This is seen in the MASTER_MBDR_MICRO signal. As this data is transmitted across the I²C bus, it is captured by both the Master and the Slave and is shown in both the MASTER_MBDR_I2C and the SLAVE_MBDR_I2C signals. When the simulation changes so that the Master reads data from the Slave, the microcontroller writes the data to be read in the SLAVE_MBDR_MICRO register. This data is captured by both the MASTER_MBDR_I2C and the SLAVE_MBDR_I2C signals as this data is transferred across the I²C bus.

Select *File* / *Quit* in the ModelSim Command window to close ModelSim.

Conclusion

This application note describes the integration between Xilinx WebPACK and Model Technology's ModelSim simulator. This powerful combination can be used for both functional and timing verification of a design before the design is actually implemented in a real device. This allows a designer the capability of working through design issues and ideas while a board

is being fabricated, increasing the flexibility of the design and shortening the end product time-to-market.

Revision History

The following table shows the revision history for this document.

Date	Version #	Revision
04/12/00	1.0	Initial Xilinx release.
10/30/00	2.0	Updated for WebPACK ISE.