Chapter 1

Watch Design - Exemplar Tutorial

This tutorial describes how to use the UNIX workstation and PC versions of Exemplar Leonardo Spectrum (Verilog/VHDL) for XC4000E/EX/XL/XV designs using MTI for simulation. It is based on the Watch design, and is a flow based tutorial. You can goto the following WEB address to see all the A1.5 tutorials that are available: http://www.xilinx.com/support/techsup/tutorials/.

To download the Watch Design - Implementation Tools Tutorial you can goto: ftp://ftp.xilinx.com/pub/documentation/M1.5_tutorials/wd_imp_15.pdf

To download the Watch Design - Hardware Verification Tutorial you can goto: ftp://ftp.xilinx.com/pub/documentation/M1.5_tutorials/wd_hwd_15.pdf

This chapter contains the following sections.

- "Design Description"
- "Required Software"
- "Before Beginning the Tutorial"
- "Installing the Tutorial Files"
- "Creating the Tenths LogiBLOX Component"
- "RTL Simulation"
- "Synthesizing the Design Using Exemplar"
- "Implementing the Watch Design"
- "Timing Simulation"

Design Description

The Watch design is a counter that counts up from 0 to 59, resets to zero, and starts over. The only user inputs are a start/stop button and a clear switch. The Watch design utilizes the OSC4 internal oscillator in the 4000E/EX/XL/XV parts. The design outputs hexadecimal values to a seven-segment display. The tenths count output is displayed on the bar LED.

The Watch Design consists of the following elements.

- **WATCH**—the top-level design
- **OSC4**—internal oscillator macro; used to generate the clock signal
- **STMCHINE**—statemachine that controls starting, stopping, and clearing the counters; one-hot encoded
- **TENTHS**—LogiBLOX 10-bit one-hot counter; outputs the tenths digit as 10-bit one-hot value
- **CNT60**—counter that outputs ones and tens digits as 4-bit binary values; counts 0 to 59 (decimal)
- **HEX2LED**—converts 4-bit values of ones and tens to 7-segment LED format
- **DEBUG_CKT**—circuit to allow synchronous debugging

Required Software

You must have the following software to use this tutorial.

- Exemplar Leonardo Spectrum v1998.2 or later
- Model Technology ModelSim EE5.1 or later (UNIX), Modelsim PE 4.7 or later (PC)
- Xilinx Development System Version M1.5 or later

Before Beginning the Tutorial

Before you begin this tutorial, set up your workstation to use Exemplar Leonardo Spectrum, Model Technology, and the Xilinx Development System as follows.

1. Verify that your system is properly configured. Consult the release notes and installation notes that came with your software package for more information.
2. Install the following software.
   - Xilinx Development System
   - Exemplar Leonardo Spectrum
   - Model Technology ModelSim EE
3. When you finish the installation, verify that the setup file or the .cshrc file contains variables similar to the following.

   UNIX platform

   ```
   setenv XILINX location_of_Xilinx_software
   setenv MODEL_TECH location_of_Modelsim_software
   setenv EXEMPLAR location_of_exemplar_software
   set path=($XILINX/bin/<platform_name> \
   $EXEMPLAR/bin/<platform_name> \
   ${MODEL_TECH}/bin $path)
   ```

   PC platform

   Make sure in the autoexec.bat that the path points to the Xilinx and Exemplar install area.

   path=c:\Xilinx\bin\nt;c:\leo_spectrum\V1998.2\bin\win32

   This is done automatically by install if you have chosen to let the install modify the system files.

Note: You may also set up the license file LM_LICENSE_FILE. See the installation notes for the particular software packages to setup the license file.

Installing the Tutorial Files

If you don't already have the tutorial files, you can download a tar.Z file from the Xilinx Web site at the following URL.

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

When you uncompress and untar or unzip the file, there are two directories, one for Verilog and one for VHDL. The Verilog and the VHDL directories each contain the /src directory where all the HDL

files are located, a solution directory called /watch_4ke, and the /watch directory containing all the files used to perform the tutorial.

Tutorial Directory and Files

The tutorial directory which contains the tutorial files needed to complete the design are in the /watch directory. Some files are not present in this directory since you will create them as part of this tutorial. The following table lists the contents of the tutorial directories.

Directory	Description
xmplr_tut/verilog/src	Verilog source files
xmplr_tut/verilog/watch_4ke	Verilog solutions directory for XC4003E-PC84
xmplr_tut/verilog/watch	Verilog Tutorial Directory
xmplr_tut/vhdl/src	VHDL source files
xmplr_tut/vhdl/watch_4ke	VHDL solutions directory for XC4003E-PC84
xmplr_tut/vhdl/watch	VHDL Tutorial Directory

Verilog Design Files

Watch.v is the top level design file which instantiates the following lower level Verilog files.

- stmachine.v
- smallcntr.v
- cnt60.v
- hex2led.v
- debug_ckt.v
- tenths.v for functional RTL simulation only

Note: The tenths one-hot counter is a LogiBLOX macro that will be created.

VHDL Design Files

Watch.vhd is the top level design file which instantiates the following lower level VHDL files.

- stmachine.vhd
- smallcntr.vhd
- cnt60.vhd
- hex2led.vhd
- debug_ckt.vhd
- tenths.vhd for functional RTL simulation only

Note: The tenths one-hot counter is a LogiBLOX macro that will be created.

Verilog Testbench

The testbench.v file is included in the tutorial directory.

VHDL Testbench

The testbench.vhd file is included in the tutorial directory.

Script Files

The following script files automate the steps in this tutorial.

- behav_sim.do
- synthesis.tcl
- implment.scr
- time_sim.do
- mti_run_ee.do (UNIX platform)
- mti_run_pe.do (PC Platform)

Simulation Libraries for MTI

To simulate Xilinx designs with ModelSim, you need the following simulation Libraries which you must compile.

- **UNISIM Library**—The UNISIM library is used for Behavioral (RTL) simulation with instantiated components in the netlist, and for post synthesis (Pre-M1) simulation. The UNISIM VHDL library is VITAL compliant, and it also adds support for new device start-up components ROC, ROCBUF, TOC, TOCBUF, and STARTBUF for simulation. The Verilog library contains separate libraries for each of the UNI3000, UNI4000E, UNI4000X, UNI5200, and UNI9000 device families.
- **LogiBLOX Library**—The LogiBLOX library is used for designs containing LogiBLOX components during pre-synthesis (RTL) and post-synthesis (Pre-M1) simulation. These LogiBLOX libraries are used for VITAL VHDL simulation only. Verilog uses SIMPRIM libraries.
- **SIMPRIM Library**—The SIMPRIM library is used for post NGDBUILD (gate level functional), post MAP (partial timing), and post-place and route (full timing) simulations. This library is architecture independent and supports VHDL and Verilog.

For detailed instructions on compiling these simulation libraries, see the instructions in Xilinx Solution # 1923 which is available at http://www.xilinx.com/techdocs/1923.htm.

After compiling the libraries, notice a file that is created by MTI called modelsim.ini. If you view this file you will notice that the upper portion defines where the compiled libraries are located. When doing a simulation, this modelsim.ini file must be copied into the directory where the HDL files are being compiled and where the simulation is being run. This can be done manually, or if the enviornment variable MODELSIM is set to point to the modelsim.ini then it will be copied over automatically when running MTI commands such as ‘vcom’.

setenv MODELSIM *path*/modelsim.ini

Copying the Tutorial Files

You can either perform the tutorial in the /xmplr_tut/(verilog or vhdl)/watch directory or copy it contents over to a directory in which to perform the tutorial.

Copy the tutorial files as follows.

UNIX platform

1. Create a project directory that you can write to when performing the tutorial. Normally you can name it whatever you want, but in this tutorial it is called tutor.

   `mkdir tutor`

2. On a UNIX workstation, use the **`cp`** command to copy all the files from the /xmplr_tut/vhdl/watch directory to the destination directory where the tutorial will be performed. Copy the tutorial files from the *untar_dir*/vhdl or verilog/ watch/ directory to the /tutor directory.

   `cp -r /xmplr_tut/(veriog or vhdl)/watch/*` *path*/ **`tutor`**

PC platform

Use the Windows Explorer and create the tutorial directory, then copy the contents of the /xmplr_tut/(verilog or vhdl)/watch directory over to this new directory.

Creating the Tenths LogiBLOX Component

Because the Watch design contains a LogiBLOX macro, you must create it before performing RTL simulation or implementation. When creating the LogiBLOX component, you will also create a behavioral simulation netlist for RTL simulation, an implementation netlist, and an instantiation netlist if the option is chosen. To create the LogiBLOX component, follow these steps.

1. To invoke the LogiBLOX GUI, type **`lbgui`** at the UNIX prompt.

   The LogiBLOX GUI and the Setup dialog box open. See the “LogiBLOX Setup Dialog Box” figure 1-1, and the “LogiBLOX Module Selector” figure 1-2.

2. In the Vendor tab of the Setup dialog box, select Mentor or other, and pick the bus notation for parenthesis B(I). See the “LogiBLOX Setup Dialog Box” figure 1-1.

3. For the Project Directory, specify the directory you wish to write the files to. You can use the Browse button, or type in the path to the project directory.

4. For the Device Family, select the xc4000e family since you are going to download to the demoboard. You can pick a different device if you do not plan to download to the demoboard.
5. In the Options tab of the Setup dialog box set the following options.

   Verilog tutorial

   - Simulation Netilst: Behavioral Verilog netlist
   - Component Declaration: Verilog Template
   - Implementation Netlist: NGC File (A1.5, NGO for A1.4)
   - LogiBLOX DRC: Stop Process on Warning

   VHDL tutorial

   - Simulation Netlist: Behavioral VHDL netlist
   - Component Declaration: VHDL Template
   - Implementation Netlist: NGC File (A1.5, NGO for A1.4)
   - LogiBLOX DRC: Stop Process on Warning
6. Click OK to close the Setup dialog box.

Note: If you are familiar with LogiBLOX, notice that the implementation netlist extension is now .ngc. This is new in the Xilinx A1.5. For more details read Xilinx Solution #3904, which is available at http://www.xilinx.com/techdocs/3904.htm.

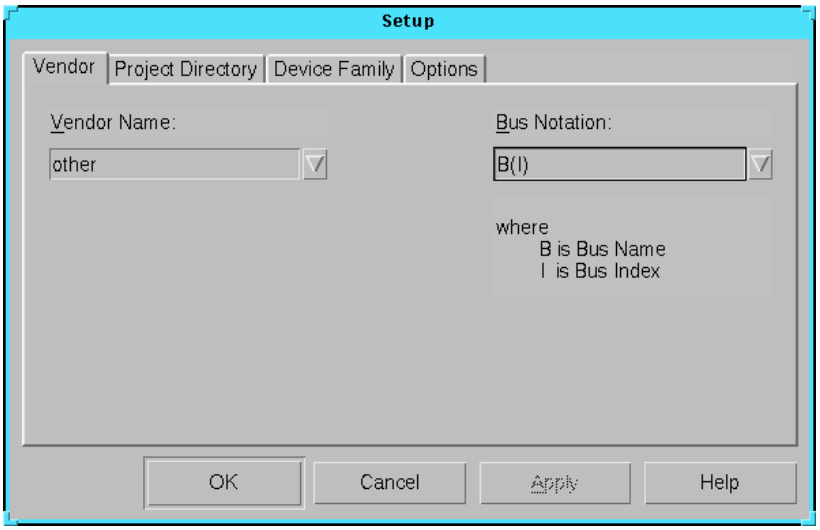

Figure 1-1 LogiBLOX Setup Dialog Box

7. In the LogiBLOX Module Selector dialog box (shown in “LogiBLOX Module Selector” figure 1-2), set the following options.
   - Module Type = Counters
   - Module Name = tenths
   - Bus Width = 10 (This width is typed in by user.)
   - Deselect D_IN
   - Select the following: Async. Control, Terminal Count
   - By default, the following is selected: Clock Enable, Q_OUT
   - Operation = Up
   - Style = Maximum Speed
   - Encoding = One Hot
   - Async. Val = 2#0000000001#

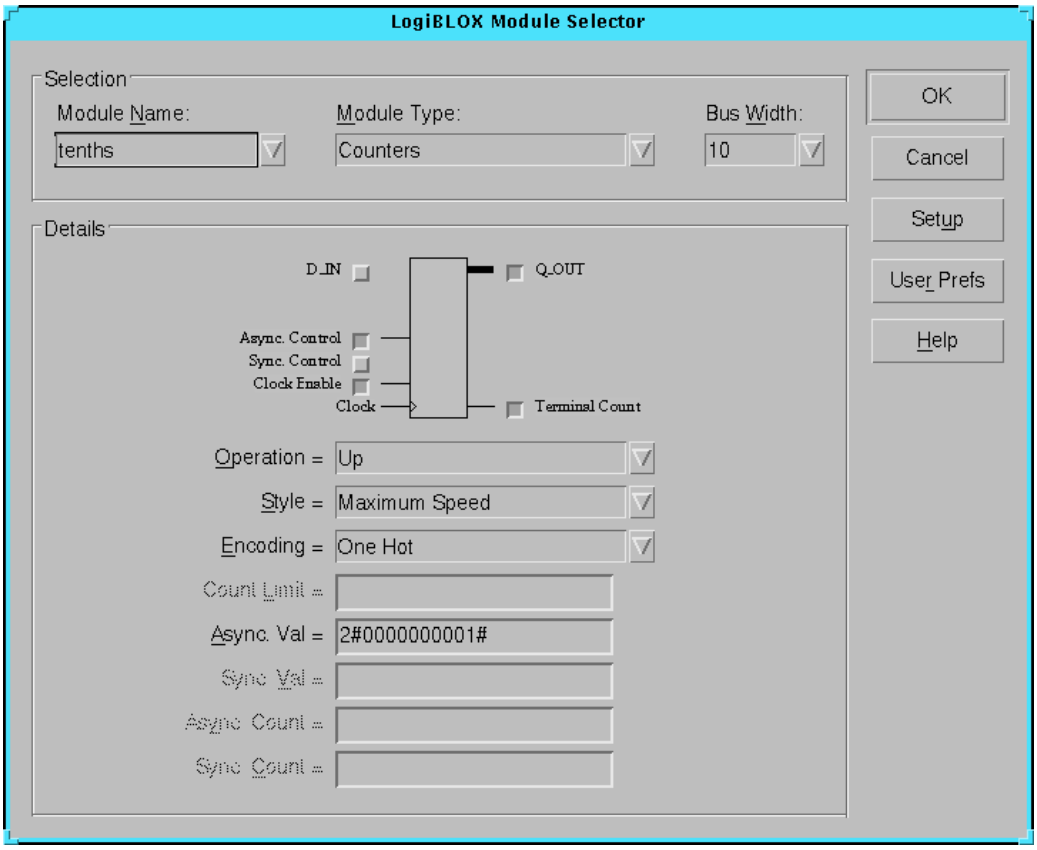

Figure 1-2 LogiBLOX Module Selector

8. Click OK.

   LogiBLOX generates the following output files.

   - logiblox.ini—shows the LogiBLOX options used
   - logiblox.log—log file of the LogiBLOX GUI messages window
   - tenths.mod—LogiBLOX Modules options file
   - tenths.ngc—implementation netlist
   - tenths.vei—Verilog declaration / instantiation template

- tenths.v—Verilog behavioral simulation netlist
- tenths.vhi—VHDL declaration / instantiation template
- tenths.vhd—VHDL behavioral simulation netlist

RTL Simulation

For simulation, the testbench.vhd file is provided for simulation stimuli, as well as the configuration statement to simulate the OSC4 symbol. For simulation, the only required files are the VHDL source files, the testbench, and the LogiBLOX tenths.vhd behavioral netlist. The simulation can be done in the project directory or in a directory of choice, such as in a directory called func.

The Watch design contains an XC4000E library part, OSC4. This component represents the on-chip oscillator that generates nominal clock frequencies of 8 MHz, 500 KHz, 16 KHz, 490 Hz, and 15 Hz. The Watch design uses the 15-Hz output from this component when targeted for XC4000E family designs. The clock output from OSC4 is buffered through a BUFG global clock buffer to minimize clock skew. XC4000E family devices have eight on-chip clock buffers, one BUFGP (primary global buffer), and one BUFGS (secondary global buffer) in each corner of the device. Although it is possible to use them for other purposes, BUFGPs are best used to route externally-generated clock signals. BUFGSs have more flexibility, and can be used to route any large fan-out net, even if it is internally sourced. A BUFG symbol can represent either type of buffer, and allows the implementation software to choose the type of global buffer that is best in each situation. BUFG also facilitates design retargeting to other Xilinx device families, since it can represent any type of global buffer in any family. The BUFG in the Watch design is substituted for a BUFGS during design implementation, because the clock is generated internally by the on-chip oscillator. See the Xilinx Libraries Guide and the Xilinx Programmable Logic Data Book for more information on global clock buffers for Xilinx devices.

It is not necessary to create a clock for the WATCH testbench, since the design already contains the OSC4 component which generates the 8MHz and 15 Hz signals. However, for simulation purposes, it would take an enourmous amount of CPU time to simulate the 8MHz and 15Hz signals. Therefore, the testbench will create the clock signal and bring this clock in on the external clock signal ‘ext_clk’, to speed up

the simulation. When downloading to the demoboard the internal clock will be used unless you are performing the Debugging Tutorial.

Note: For Verilog simulation, the OSC4 model has a timescale precision of 100ps. To make a transition to the first edge of the 15Hz clock, which is at 3.33E10 ps (.0333 seconds), requires 3.33E10 / 100 = 333 million simulation events. The OSC4.v UNISIM model is located at $XILINX/verilog/src/UNI4000E. Therefore, a clock is defined in the testbench/testfixture that clocks much faster, and this clock is selected through a multiplexer to force its values onto the CLK signal, bypassing the OSC4 F15 clock.

Note: Xilinx Solution # 3767 contains further information on the use of the OSC4 with VHDL simulation for ModelSim. This is available at http://www.xilinx.com/techdocs/3767.htm for review.

Copying Source Files to the Functional Simulation Directory

Copy the following files into the tutorial directory.

Verilog tutorial

Copy the following files from the xmplr_tut/verilog/watch/ directory to the /tutor/func directory:

- smallcntr.v
- cnt60.v
- hex2led.v
- tenths.v
- debug_ckt.v
- watch.v
- stmachine.v
- testbench.v
- behav_sim.do
- mti_run_ee.do (UNIX)
- mti_run_pe.do (PC)

VHDL tutorial

Copy the following files from the xmplr_tut/vhdl/watch/ directory to the /tutor/func directory:

- smallcntr.vhd
- cnt60.vhd
- hex2led.vhd
- tenths.vhd
- debug_ckt.vhd
- watch.vhd
- stmachine.vhd
- testbench.vhd
- behav_sim.do
- mti_run_ee.do (UNIX)
- mti_run_pe.do (PC)

Starting ModelSim

Unix platform

If you are using ModelSim EE, invoke the simulator by typing the following at the UNIX prompt in the /tutor/func/ directory:

vsim -i &

PC platform

If you are using a PC, invoke the simulator by selecting Programs → Model Tech → ModelSim from the Startmenu. Then set the project directory using the File → Directory menu command and select tutor/func/ directory.

Creating the Work Directory

Before compiling the HDL files, you must create a work directory, for use as a library, as follows.

1. At the ModelSim prompt type the following:.

   `vlib work`

Note: You must use the vlib command to create the work directory. MTI creates a file inside work so it can recognize this as a work directory.

Compiling the HDL Source Files

Verilog Tutorial

You will need to comment out the Tenths module declaration within the file watch.v since we will be providing the simulation model for this component in later steps. You can comment out the lines by adding a slash-slash '//' at the beginning of the following lines:

```
module tenths (CLK_EN, CLOCK, ASYNC_CTRL, Q_OUT,
TERM_CNT)
/* synthesis black_box */;
input CLK_EN, CLOCK, ASYNC_CTRL;
output [9:0] Q_OUT;
output TERM_CNT;
endmodule
```

The Vlog command compiles Verilog code for use with Vsim RTL simulation. Type the following at the ModelSim prompt:

vlog testbench.v watch.v stmchine.v hex2led.v debug_ckt.v cnt60.v smallcntr.v tenths.v

VHDL tutorial

Since Xilinx Unified library components are instantiated within the VHDL source code, the UNISIM simulation models must be provided for the OSC4, BUFG, MD0, MD1, IBUF, OBUF, RDBK, and STARTUP components. The following lines have already been added in the files watch.vhd and debug_ckt.vhd.

```
library UNISIM;
use UNISIM.vcomponents.all;
```

To compile the VHDL files for RTL simulation type the following commands at the ModelSim prompt.

```
vcom tenths.vhd
vcom -explicit smallcntr.vhd
vcom cnt60.vhd
vcom hex2led.vhd
vcom debug_ckt.vhd
vcom stmchine.vhd
vcom watch.vhd
vcom testbench.vhd
```

The -explicit option resolves resolution conflicts in favor of explicit functions.

Note: The above command along with invoking the simulator commands below have been combined into a macro 'do' file. See the Note in the 'Invoke the Simulator' section below on how to run the macro file.

Invoke the Simulator

Verilog tutorial

For the verilog tutorial type the following at the ModelSim prompt to invoke the ModelSim simulator:

vsim -L simprim_ver -L UNI4000E test

Since Xilinx Unified library components are instantiated within the Verilog source code, the UNISIM simulation models must be provided for the OSC4, BUFG, MD0, MD1, IBUF, OBUF, RDBK, and STARTUP component. Also notice that the library simprim_ver is listed as well, which is the name of the compiled verilog simprim library name. For LogiBLOX generated components, NGD2ver is used to generate a structural Verilog netlist to facilitate functional simulation. The structural netlist contains SIMPRIM library components which is mapped to the library simprim_ver.

VHDL

For the VHDL tutorial type the following at the ModelSim prompt to invoke the ModelSim simulator, and to load 'overall'

vsim overall

Note: The above sections for compling the HDL source and Invoking the simuator commands have been combined into a 'do' file that can be run after creating the 'work' library. The file is called behav_sim.do. To execute the file type the following at the ModelSim prompt:

do behav_sim.do

ModelSim EE users can run the macro file by choosing: macro → Execute Macro, and choosing the do file. ModelSim PE users can choose: File → Execute Macro, then choosing the do file

Running the Simulation

To perform the simulation do the following steps.

1. To view all the ModelSim debugging windows, type the following at the ModelSim prompt.

   ```
   view *
   ```

   The Signals window, Wave window, and various other MTI windows open. See the "MTI Signals and Wave Windows" figure 1-3.

2. In the Signals window, highlight the signals to be simulated. Highlight all of them for this tutorial.

3. Drag and drop the signals into the Wave window to display the simulation waveforms as shown in the following figure 1-3.

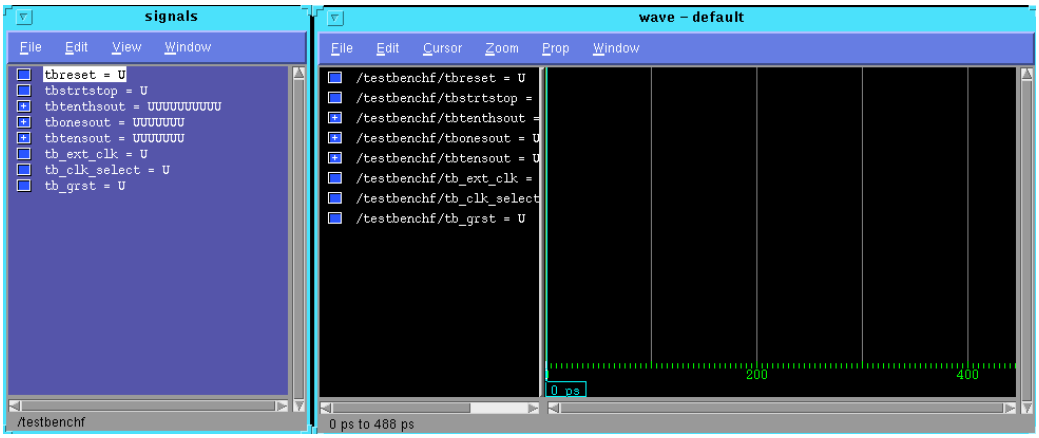

Figure 1-3 MTI Signals and Wave Windows

The equivalent way of doing this at the ModelSim prompt is to type the following.

UNIX platform

```
add wave *
```

PC platform

```
wave *
```

In the Structure window notice that Verilog design units are indicated by circles, and VHDL design units are indicated by squares. You can expand and collapse the regions of hierarchy by clicking on the (+) and (-) notations.

4. To run the simulation for a user specified amount of time at the ModelSim prompt, type the following.

   ```
   run 100 us
   ```

   The simulation runs for the specified amount of time, and the simulation output shows up in the Wave window. See “Simulation Output in Wave Window” figure 1-4.

5. You may have to zoom in or out to view the waveforms. To do this right mouse click in the Wave window and choose the Zoom Full option to see the entire simulation up to this point.

Note: The above commands have been combined into a macro file called mti_run_ee.do or mti_run_pe.do, that can be executed in ModelSim. After invoking vsim and loading the design select **`Macro`** → **`Execute Macro`** from the MTI main window then select mti_run_*.do.

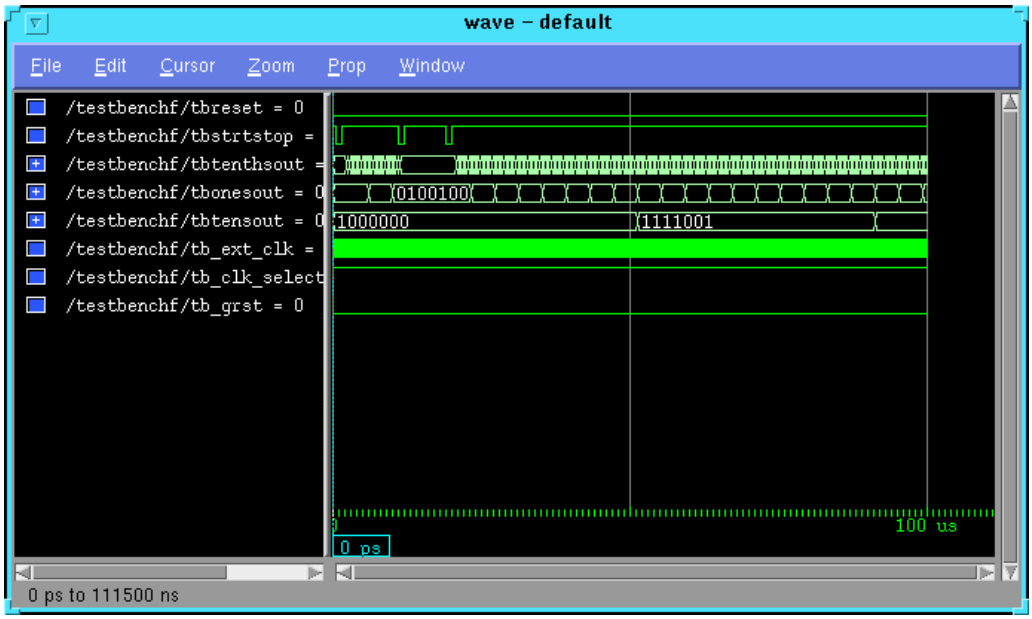

Figure 1-4 Simulation Output in Wave Window

Synthesizing the Design Using Exemplar

In this section you will synthesize your design using three different methods.

- Leonardo Spectrum Level 1
- Leonardo Spectrum Level 2
- Leonardo Spectrum Level 3

Currently Leonardo Spectrum Level 1 is not included with the release software, but will be introduced at a later time. Level 2 is basically equivalent to the previous Galileo version, and has a Wizard to automate the synthesis step. Level 3 is equivalent to the previous

Leonardo 4.2.2 version, and also has the Wizard to automate the synthesis step, but also has an interactive capability to give the user more control over synthesis.

Verilog tutorial

You will need to either add or make sure the Tenths module declaration within the file watch.v exists, as this is needed for a black box instantiation. You can un-comment the lines by removing the slash-slash '//' at the beginning of the following lines if they exist:

```
module tenths (CLK_EN, CLOCK, ASYNC_CTRL, Q_OUT,
TERM_CNT)
/* synthesis black_box */;
input CLK_EN, CLOCK, ASYNC_CTRL;
output [9:0] Q_OUT;
output TERM_CNT;
endmodule
```

VHDL tutorial

Note: Since the VHDL files contain instantiated Xilinx components, the UNISIM library must be used. For synthesis, references in the VHDL files to the UNISIM libraries must be removed. To do this, use a text editor such as vi (UNIX) to edit the file watch.vhd and debug_ckt.vhd. Either comment out the following lines by putting "--" in front of each line, or just remove the lines.

```
-- library unisim;
-- use unisim.vcomponents.all;
```

For synthesis you may want to create a directory in which to process the design through Exemplar. You will need to copy the follwing files into the directory for synthesis: smallcntr.vhd, cnt60.vhd, debug_ckt.vhd, hex2led.vhd, stmchine.vhd, watch.vhd, and synthesis.tcl.

Leonardo Spectrum Level 1

Leonardo Spectrum Level 1 is for a single technology and allows a new designer to quickly produce a high-qulaity netlist. Level 1 has an

easy upgrade path to Leonardo Spectrum Level 2. A logic designer will select the input design and technology and the click on the Run button. Currently Level 1 is not included in the initial release, but will be introduced at a later date. For more information about Leonardo Spectrum Level 1 you can goto the Exemplar web site at: http://www.exemplar.com

Leonardo Spectrum Level 2

Leonardo Spectrum Level 2 is an easy-to-use FPGA synthesis, timing analysis, and back-annotation tool. It is designed to maximize the productivity of FPGA designers. The designer selects the input design, output design, sets the constraints and target technology, and runs the tool. A netlist is quickly produced. Level 2 has an easy upgrade path to Level 3. For this tutorial we will run the Spectrum Synthesis Wizard to process the design. With Level 2 it is possible to run through each step of the design using the tabs.

The following apply for Unix and PC users unless otherwise specified.

1. To start-up the Leonardo Spectrum Graphical User Interface, do the following:

   UNIX platform

   At the UNIX prompt type the following:

   Leonardo &

   PC platform

   On a PC double-click on the Leonardo Spectrum icon on the desktop, or choose **Programs** → **Leonardo Spectrum V1998.2**→ **Leonardo Spectrum**

The Exemplar Leonardo Spectrum license checkout window opens as shown in the following figure 1-5.

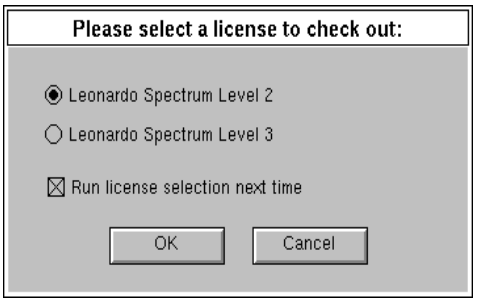

Figure 1-5 Leonardo Spectrum license checkout Window

2. Select Leonardo Spectrum Level 2, and click the OK button
3. The Leonardo Spectrum Synthesis Wizard Input Files window will open as shown in the following figure 1-6

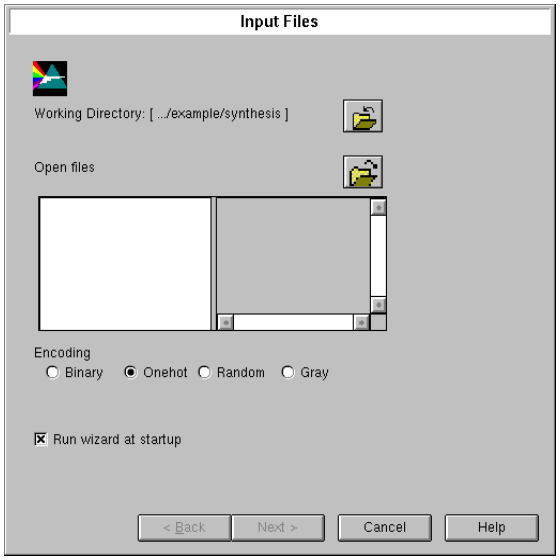

Figure 1-6 Leonardo Spectrum Synthesis Wizard Input Files

If this window does not come up choose **Flows** → **Synthesis Wizard**.

4. Check the working directory which is listed in the Synthesis Wizard window. The working directory is also listed in lower right hand corner of the Leonardo Spectrum Main Window. By default the working directory is set to the directory which the working directory was last set to. To change the working directory click on the folder icon just to the right of the listed working directory and browse to the proper directory. Select the directory and click on the Set button.

5. Add the files to the Open files box by clicking on the open files icon to the right. The Set Input File(s) window will be shown as in the following figure 1-7. By default Verilog and VHDL files will be displayed. You can select the files by left mouse clicking and using a combination of either holding the left mouse button down and highlighting all files at once, using the left mouse button along with the shift-key and / or ctrl-key.

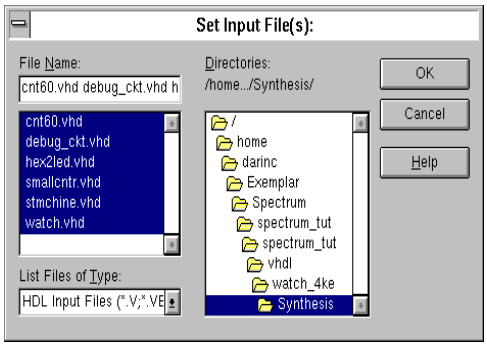

Figure 1-7 Set Input File(s) window

After the appropriate files are selected click the OK button.

6. The order of the files read in must be from the bottom up. To arrange the files into the proper order highlight the file and drag and drop it into the appropriate place to reflect the following figure 1-8

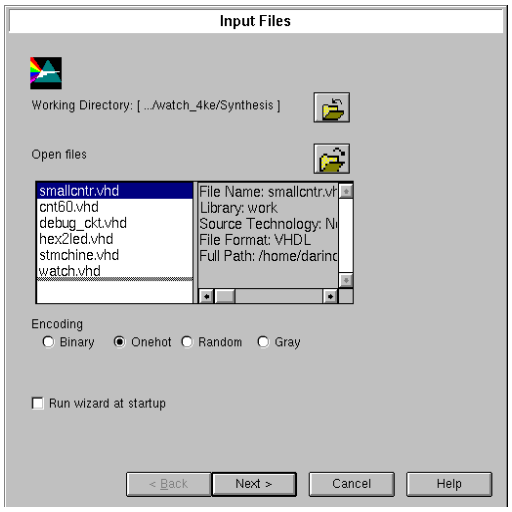

Figure 1-8 Input Files order window

7. Once the files are listed the Technology must be set for the HDL files. This must be done since Level 2 runs everything at once and the Target Technology library end up getting loaded after reading the files in. So when there are instantiated components in the HDL from a particular technology, then it must be set on the files. To do this do the following:

   a) Right mouse click in the Open files window where the files are listed.

   b) Goto **Set Technology All → Xilinx → 4000E**

8. Click on the Next > button

9. In the Device Settings window you can set the technology. If you are targeting the demo board make the following selections:

   Xilinx 4000E

   Part 4003ePC84

   Speed -3

   Wire Load 4003e-3_avg

   The wire load wire table gives an estimate for wire length as a function of fanout. Average in the instance "_avg".

After the selections are chosen click on the Next > button

10. In the Global window you can Specify Clock Frequency of 40 MHz, although for the tutorial this actually is not needed as on the demo board the clock is going to be very slow. Click the Next > button

11. In the Output File window the Filename: should already be set to the top level name.edf, watch.edf in this tutorial, by default. It will also give the path to the directory it is writing it to. If you would like to change where it writes the otuput file to click on the folder icon and select the destination.

    For the Format choose EDIF.

    Check the box for Write vendor constraints file (.ncf file)

    Click on the Next > button

12. The Review window will come up showing all the options you have set and should look something like the following figure 1-9

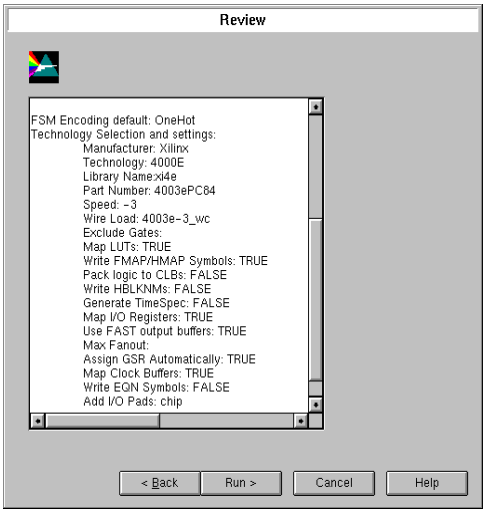

Figure 1-9 Spectrum Level 2 Review window

Click on the Run > button. The Review window will disappear and the Spectrum Main Window will remain open and in the right hand side of the Main window information will scroll by as the design is processed. The following files are written out to the working directory:

exemplar.log - text file containing all the information that scrolls by in the Main Window

exemplar.his - text file of the command and options run

watch.sum - Summary of the area and device utilization

watch.edf - Edif netlist to go into the Xilinx core tools

watch.ncf - constraints file for timing to go into the Xilinx core tools.

Optionally you can now view the schematics of the RTL and Optimized netlists by selecting: **Tools → View RTL Schematic** or by selecting **Tools → View Gate Level Schematic** respectively. With Spectrum Level 2 you will only be able to view the Schematics after completing the flow.

Using the Exemplar Design Wizard, the option for entering in constraints for pin locks is not available. Before Implementing the design in the Xilinx tools you will need to add pin locks for two signals into the supplied .ucf file. Using Spectrum Level 3 will explain how to enter in the pin lock from the GUI. To add the pin locks to the .ucf add the following two lines to the file watch.ucf:

NET reset LOC = P28;

NET strtstop LOC = P18;

Optionally you can now implement the design through the Xilinx tools from the Exemplar main window, given that the Xilinx environment had been setup properly. This can be done by clicking on the P&R tab in the main window choosing the Execute Place_Route option. If you are going to be doing a Timing Simulation you will also need to select Generate files for timing simulation, as well as Generate bit file if you are going to download to the demoboard. For more specific usage of the Xilinx Design Manager see the ‘Implementing the Watch Design” section on Page 1-34 of this tutorial, which will refer you to the ‘Watch Design mplementations Tools Tutorial’.

Leonardo Spectrum Level 3

Leonardo Spectrum Level 3 has all the capabilities as described for Level 2, plus interactivity capabilities. Level 3 supports bottom-up and top-down design methodologies. A designer can efficiently and economically consolidate multiple designs into one design, and preserve and manipulate the design hierarchy. A designer may set constraints on any level of hierarchy, and then synthesize it separately with a different constraint. The design can be verified after synthesis is complete by generating HDL RTL from the tool and running it through a HDL simulator. Post place-and-route functionality with timing information can be verified.

Each step is explained below in the order you would run them. You do not necessarily need to run all the steps to write out the EDIF file.

a) **(Quick Setup Tab)** —Define all input files, output files, target technology, target frequency, and effort to Run Flow to get an output netlist for implementation. Simliar to a consolidated Synthesis Wizard. This Step takes the place of running the following steps b through i, not including constraints nor report options.

b) **(Technology Tab) Load Library** —Reads a compiled Technology library file, then creates a library in Leonardo's design database. Modgen is automatically loaded with Load Library.

c) **(Input Tab) Read** —Loads a design from a file into the Leonardo design database.

d) **(Constraint Tab) Apply** —Allows you to specify user-defined constraints in the design.

e) **(Optimize Tab) Optimize** —Performs technology-specific logic optimization and technology mapping.

f) **(Timing Opt Tab) Optimize for Timing** —Performs extensive timing optimization on the design. This only appears if Timing Optimization is selected.

g) **(Report Tab) Report Area** —Reports the accumulated area of the present design.

h) **(Report Tab) Report Delay** —This option does critical path reporting. This appears twice if Timing Optimization is

selected, otherwise it appears once. This allows comparing of results before and after doing timing optimization.

i) **(Output Tab) Write** —Writes the output netlist in the user specified format.

j) **(P&R Tab) Run PR**—Exemplar template that uses standard setting to run the Xilinx core tools and to write out Timing Simulation netlists and bit file for download to the chip. There is also options for netlist for functional simulation pre-Place & Route delay estimate, and running Xilinx Design Manager only.

1. To start-up the Leonardo Spectrum Graphical User Interface, do the following:

UNIX platform

At the UNIX prompt type the following:

Leonardo &

PC platform

On a PC double-click on the Leonardo Spectrum icon on the desktop, or choose **Programs** → **Leonardo Spectrum V1998.2** → **Leonardo Spectrum**

The Exemplar Leonardo Spectrum license checkout window opens as shown in the following figure 1-10.

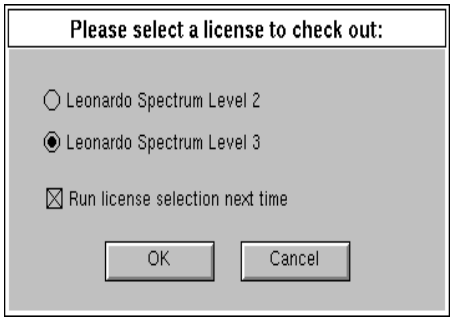

Figure 1-10 Leonardo Spectrum license checkout Window

2. Select Leonardo Spectrum Level 3 and click the OK button

3. The Leonardo Spectrum Main Window will now show up similiar to the following figure 1-11.

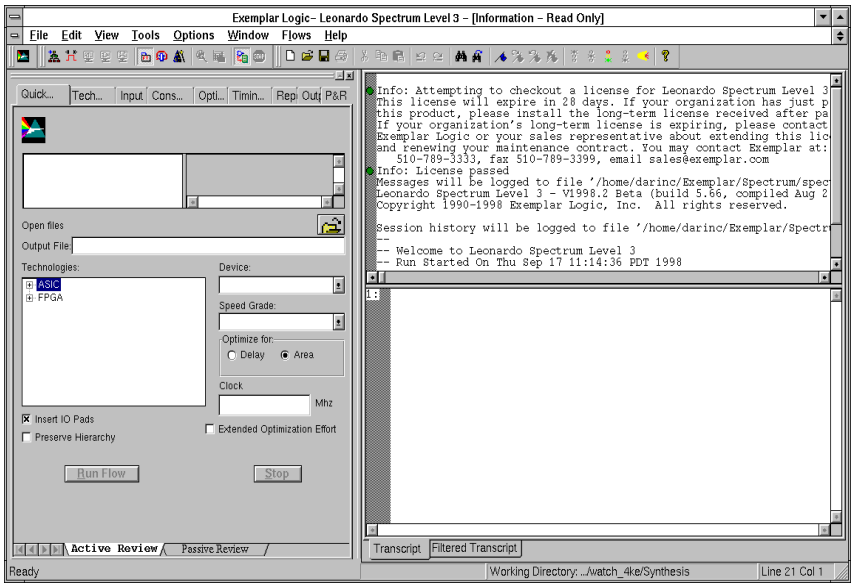

Figure 1-11 Leonardo Spectrum Main Window

4. Click on the technology tab and choose **FPGA** → **Xilinx** → **4000E** and then choose the following options as seen in figure 1-12:

   Part: 4003ePC84

   Speed: -3

   Wire Load: 4003e-3_wc

   The wire load wire table gives an estimate for wire length as a function of fanout. Worst case in the instance "_wc".

   Click the Load Library button at the bottom of the window

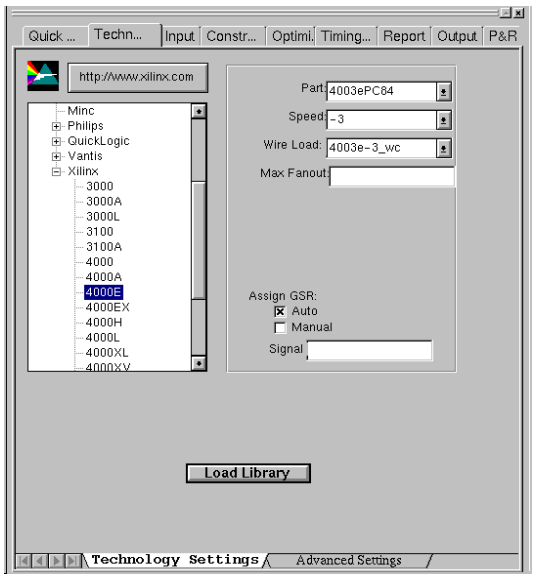

Figure 1-12 Spectrum Level 3 Technology Settings window

5. In the lower right hand portion of the Main Window will show the working directory. You can change the working directory by going **File** → **Change Working Directory**, then browsing to the directory and hitting the set button. You can also change the working directory in the Input tab window, by clicking on the open folder icon and browsing to the appropriate directory. See figure 1-13.

6. Under the Input tab add the HDL files to be read in by clicking on the open file icon and browsing, or right-mouse click in the empty box under the Open files text and choose the Add Input Files. Select the files you wich to add and click on the OK button.

7. Next the files must be read in from the bottom up. To change the order of the listing just drag and drop the file in the appropriate location. The order should reflect the following in figure 1-13.

   Click on the Read button in the lower portion of the Input window.

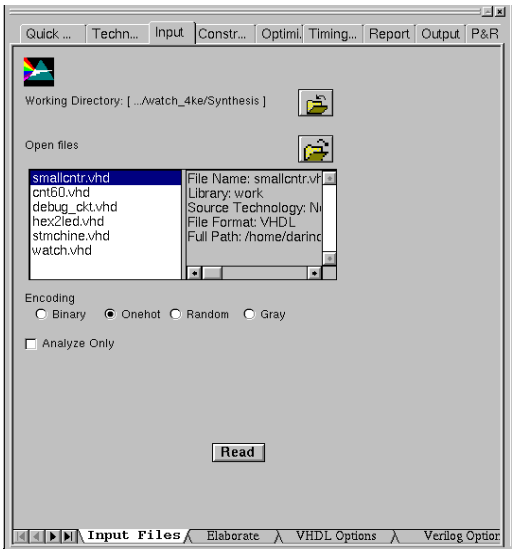

Figure 1-13 Spectrum Level 3 Input Files window

8. You will notice after the ‘READ’ operation that the ‘View RTL Schematic’ icon is now selectable, on the toolbar just below the pulldown menus.. Optionally you can now view the RTL Schematic by choosing **Tools** → **View RTL Schematic** or by clicking on the toolbar icon. The Schematic Viewer will come up with the design loaded and the schematic will be shown. You will notice as you select components in the schematic, that Spectrum automatically opens the corresponding HDL code and cross-probes to the code which created the selected logic.

9. Click on the Constraints tab. Since this design runs at quite a slow frequency it is not necessary to enter in a Clock frequency. You can enter in a Global timing constraint for the clock, of 40 MHz to see the resulting .ncf file file timing constraints that are written out. Click on the Apply button. We will be using the Input sub-tab, found at the bottom of this particular window, to lock two signals to two pins, then use a .UCF to lock all the rest in order to save time. See figure 1-14.

   In the Input window highlight the signal reset, then give the Pin Location of P28, then click the Apply button.

Next highlight the signal strtstop and lock it down to pin 18 using the same process as above.

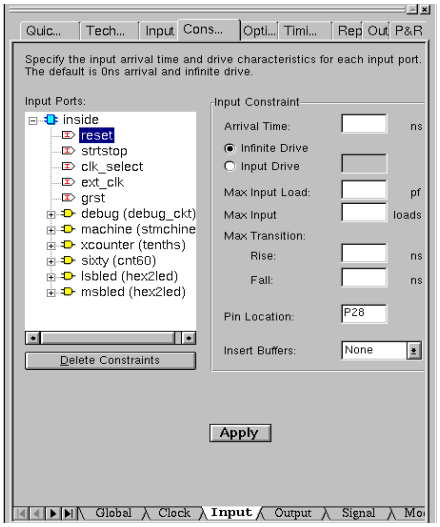

Figure 1-14 Spectrum Level 3 Constraints window

10. Click on the Optimize tab. By default the architecture named inside should already be highlighted. We will be using all default settings, so simply click on the Optimize button. See figure 1-15.

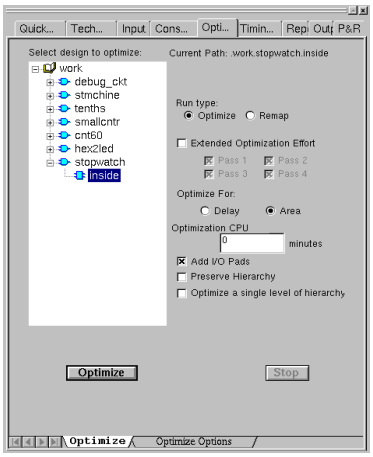

Figure 1-15 Spectrum Level 3 Optimize window

11. Click on the Output tab. By default the Filename will be the top level file .EDF, i.e. watch.edf in this tutorial. Select the Format to be EDIF. Here will will need to set an EDIF output option. Click on the EDIF Out Options tab in the lower portion of the Output window. Here you will need to turn off the option to Allow Writing Busses, by deslecting the checkbox. Next click on the Apply button. See figure 1-16.

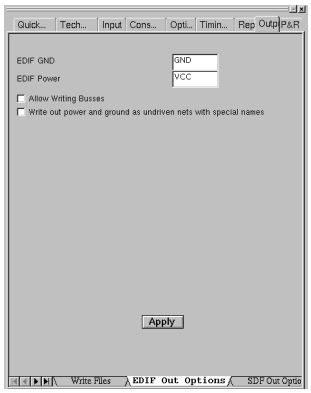

Figure 1-16 Spectrum Level 3 Edif Out Options

Click on the Write Files tab in the lower portion of the Output window, and click on the Write button to write out the EDIF netlist as in the following figure 1-17.

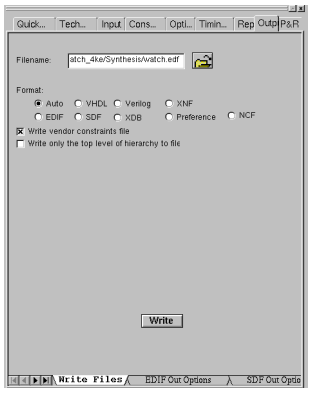

Figure 1-17 Spectrum Level 3 Write Files

The following files are written out to the working directory:

exemplar.log - text file containing all the information that scrolls by in the Main Window

exemplar.his - text file of the command and options run

watch.sum - Summary of the area and device utilization

watch.edf - Edif netlist to go into the Xilinx core tools

watch.ncf - constraints file for timing to go into the Xilinx core tools.

Optionally you can now view the schematics of the Optimized netlists by selecting: **Tools** → **View Gate Level Schematic**. With Spectrum Level 3 you will be able to view the RTL Schematic after doing the 'Read' operation, and can view the Synthesized gaet level netlist after the 'Optimize' operation.

Optionally you can now implement the design through the Xilinx tools from the Exemplar main window, given that the Xilinx environment had been setup properly. This can be done by clicking on the P&R tab in the main window choosing the Execute Place_Route option. If you are going to be doing a Timing Simulation you will also need to select Generate files for timing simulation, as well as

Generate bit file if you are going to download to the demoboard. For more specific usage of the Xilinx Design Manager see the 'Implementing the Watch Design" section on Page 1-34 of this tutorial, which will refer you to the 'Watch Design Implementations Tools Tutorial'.

Operating Leonardo in Batch Mode

As you were processing the Watch design in Leonardo you may have noticed that for each command that ran, such as Load Library, Read, and Optimize, that the exact command including the file names appeared in blue in the Leonardo Main Window. Each of these commands can be put into a file and run from a command line using the 'spectrum' command, which is equivalent to using the 4.2.2 'elsyn' command. This script file has already been created, called synthesis.tcl.

To run the Script from the Leoanrdo Spectrum GUI choose **File** → **Run Script** and either select the file synthesis.tcl or type the filename in. Click the OK button and the script will be executed.

To run Leonardo Spectrum in script mode you can also type the following from the UNIX prompt.

```
spectrum -file synthesis.tcl
```

This executes the Tcl script file and exits when finished. The file watch.edf as well as an exemplar.log file are created. The flow through Leonardo is fully defined by the commands in the script and not fixed as with Galileo compatibility mode. The script can use any command that Leonardo accepts including all Tcl and shell commands that can be found in the path.

Implementing the Watch Design

To implement the design, refer to the Watch Design - Implementation Tools Tutorial. You can download this file by from:

ftp://ftp.xilinx.com/pub/documentation/M1.5_tutorials/wd_imp_15.pdf

If you are going to download and do a readback from the demo board you can get the Watch Design - Hardware Verification Tutorial from:

ftp://ftp.xilinx.com/pub/documentation/M1.5_tutorials/wd_hwd_15.pdf

You need the following files for implementation.

- watch.edf
- watch.ucf (provided, not created)
- tenths.ngc (A1.5, tenths.ngo for A1.4)

You may want to create a directory in which to process the design thorugh the Xilinx core tools, such as m1_run. The above three files will need to be copied over to this directory.

When you implement the Watch design in the Design Manager, here are some specific options to use in the Xilinx Design Manager.

1. When you go to implement, in the Implement window, select the Options button to open the Options dialog box.
2. In the Program Option Template, set the Simulation to the appropriate ModelSim Verilog or ModelSim VHDL.
3. Just to the right of this, click on the Edit Template button to open the XC4000 Simulation Options ModelSim VHDL dialog box.
4. In the VHDL/Verilog Tab, de-select the option to Generate Test Fixture Testbench File and click OK.
5. In the Options window, select the Produce Timing Simulation Data option.
6. Proceed with the “Watch-Design Implementation Tools Tutorial.”
7. Also it is possible to run a post-NGDBuild and post-MAP simulation, which may be helpful for debugging the design. However, this tutorial does not include running these simulations.

Also provided is the file implment.scr which will run the necessary programs to create the files needed for Timing Simulation and for downloading to the demoboard. To use the script file first copy the file implement.scr to the *path*/m1_run directory. You will also need to copy the file bitgen.ut into this directory. To run the script file type the following at the UNIX prompt: ./implmement.scr. On a PC you can run each individual command contained in the implment.scr, from the MS-DOS prompt.

Timing Simulation

Verilog tutorial

For the Timing Simulation you will need the time_sim.v and the time_sim.sdf from the Xilinx core tools.

VHDL tutorial

For Timing simulation, you will need time_sim.vhd and the time_sim.sdf files from the Xilinx core tools.

Now that the HDL netlist has been resolved into primitives, the testbench configuration needs to be modified slightly. The UNISIM library was referenced in the RTL simulation since the pre-synthesis netlist contained instantiated Xilinx macros. Now for timing simulation the UNISIM library reference must be removed from the testbench.

Again, it is not necessary to create a clock for the WATCH testbench, since the design already contains the OSC4 component which generates the 8MHz and 15 Hz signals. However, for simulation purposes, it would take an enourmous amount of CPU time to simulate the 8MHz and 15Hz signals. Therefore, the testbench will create the clock signal and bring this clock in on the external clock signal 'ext_clk' to speed up the simulation. When downloading to the demoboard the internal clock will be used, unless you are performing the Debugging Tutorial

To perform timing simulation for your design, follow these steps.

Note: The following steps for the Veriog/VHDL timing simulation have been combined into macro 'do' files. The file time_sim.do can be run at the ModelSim prompt, followed by the mti_run_ee.do or mti_run_pe.do file. The time_sim.do file will compile the HDL file and then start the simulator. The mti_run_*.do file will bring up the necessary debugging windows and run the simulation for 100 us.

Create a directory in which to run the timing simulation.

```
mkdir /tutor/time
```

8. Copy the following files to the /tutor/time directory.

Verilog tutorial

```
cp time_sim.v path/tutor/time/
cp time_sim.sdf path/tutor/time/
cp testbench.v path/tutor/time/
```

VHDL tutorial

```
cp time_sim.vhd path/tutor/time/
cp time_sim.sdf path/tutor/time/
cp testbench.vhd path/tutor/time/
```

9. As long as the MODELSIM variable is set the modelsim.ini will be copied over automatically when using the MTI commands. To copy the modelsim.ini file into the /tutor/time directory manually use the following:.

```
cp /ModelLibs/modelsim.ini /tutor/time/
```

10. Another way to run ModelSim is to compile all the files outside of the GUI, then startup the GUI. To do this do the following:

    Create the work directory in the /tutor/time directory.

```
vlib work
```

11. Modify the testbench to use the configuration for Timing Simulation. The Verilog Testbench does not need to be modified since the 'vsim' -L option will point to which library to use. Edit the testbench.vhd file, and at the bottom there are two sections. The first section is for functional simulation and is currently being used. This has to be commented out by using the '--' at the beginning of each line starting with the line

```
configuration overall of testbenchf is
```

    and ending with the line

```
end overall
```

    in the Functional Sim Section.

12. In the time_sim.vhd Timing Sim Section, uncomment the lines by removing the '--' symbols, again for the line beginning with

```
configuration overall of testbenchf is
```

and ending with

```
end overall
```

13. Also in the testbench.vhd the reference to the UNISIM libraries must be removed. Use the '--' at the beginning of the following lines to comment them out:

    librarary UNISIM;

    use UNISIM.vcomponents.all;

14. Save the changes and exit the testbench.vhd file.
15. Compile the HDL files by doing the following from the UNIX prompt (UNIX), or from the ModelSim prompt (PC):

    Verilog tutorial

    vlog time_sim.v

    vlog testbench.v

    VHDL tutorial

    vcom tim_sim.vhd

    vcom testbench.vhd

16. Invoke ModelSim and read in the SDF file for timing simulation.

UNIX platform

From the UNIX prompt do the following:

Verilog

vsim -L simprim_ver test

VHDL

vsim -sdftyp uut=time_sim.sdf overall

PC platform

From the ModelSim prompt do the following:

Verilog

vsim -L simprim_ver test

VHDL

vsim -sdftyp uut=time_sim.sdf overall

OR

Reading the SDF in through the ModelSim GUI

Verilog

NGD2ver automatically writes out a directive, $sdf_annotate within the time_sim.v file. This directive specifies the appropriate SDF file to use in conjunction with the produced netlist. So, it is unnecessary for the user to specify an option for ModelSim to read the SDF. To load the design and to tell it to use the simprim_ver simulation models type the following at the ModelSim prompt:

vsim -L simprim_ver test

VHDL

Alternatively after starting up ModelSim ModelSim EE users can select File → Load New Design. ModelSim PE users can select File → Simulate.

The Load Design dialog box opens as shown in the following figure 1-25.

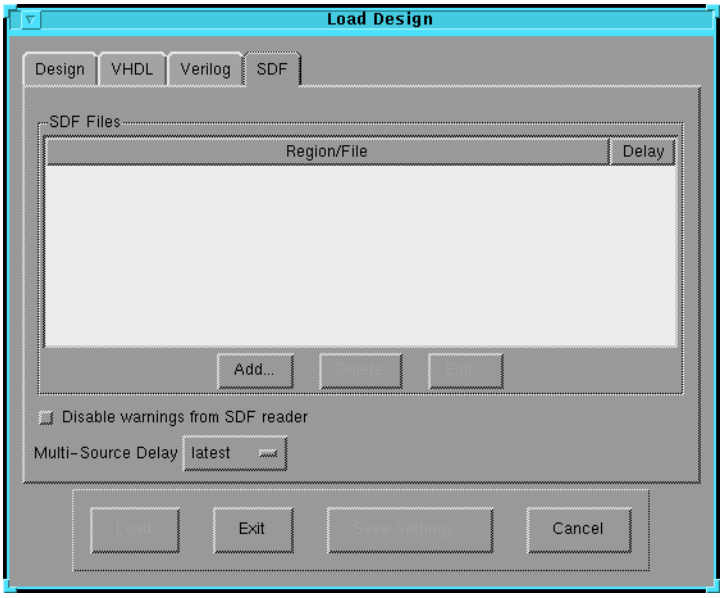

Figure 1-18 Load Design Dialog Box

a) Click on the SDF tab of the Load Design window. Refer to the following figure 1-26.

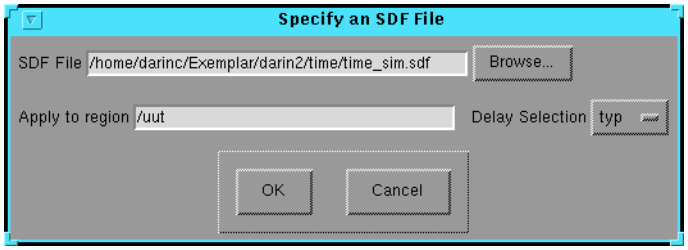

Figure 1-19 Specifying the SDF File

b) Click the Add button.

c) Browse for the time_sim.sdf file and select it.

d) In the Apply to Region field type the following.

```
/uut
```

e) Click on the Design tab and select the Design Unit overall (should be in red) and Description Config.

These should both be highlighted when either is selected.

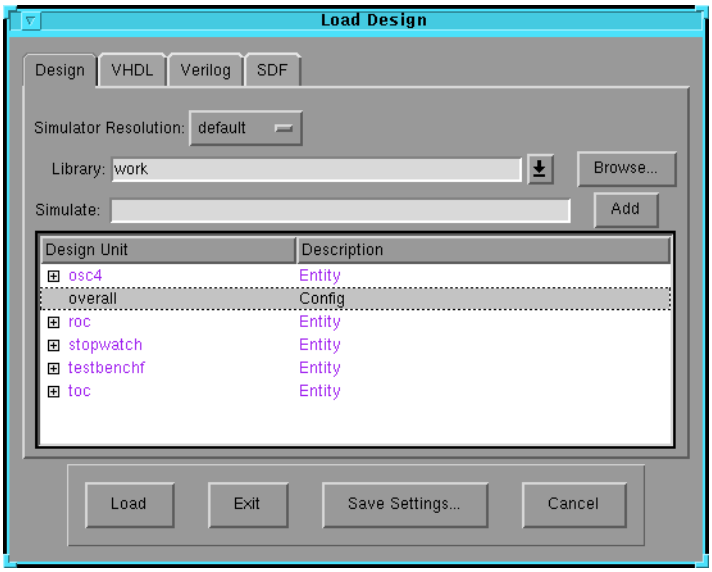

Figure 1-20 Loading the Design into MTI

f) Click the Load button.

17. View the necessary windows.

    `view wave signals source`

18. Highlight the signals you wish to view and add them to the waveform window using the drag and drop technique.

    OR

    type the following at the ModelSim prompt:

    UNIX platform

    add wave *

    PC platform

    wave *

19. At the ModelSim prompt, run for 100 us

    run 100 us

20. Right click in the waveform window and zoom in. Another way to zoom in is to press and hold the middle mouse button and draw a square around the area to zoom in on. If you click on the oscillator in the structure window you can then add the f15 signal from the signals window. After simulating you can zoom in and view the delay from the clock edge to the tenthsout, onesout, and tensout output change.

    The Exemplar Tutorial is now completed!