Chapter 1

# Watch Design - Exemplar Tutorial

This tutorial describes how to use the UNIX workstation and PC versions of Exemplar Leonardo / Galileo Extreme (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 version 4.2.2 or later, or Galileo Extreme version 4.2.2 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 or Galileo Extreme, 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 or Galileo Extreme 4.2.2 or later
   - 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:\Exemplar\leonardo\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**

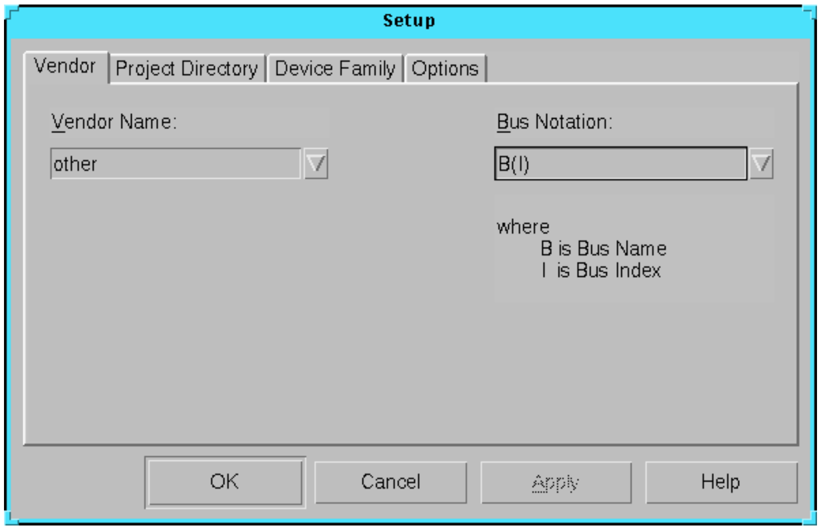
- Simulation Netilst: Behavioral Verilog netlist
- Component Declaration: Verilog Template
- Implementation Netlist: NGC File (A1.5, NGO for A1.4)
- LogiBLOX DRC: Stop Process on Warning

**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.



Vendor Name: other

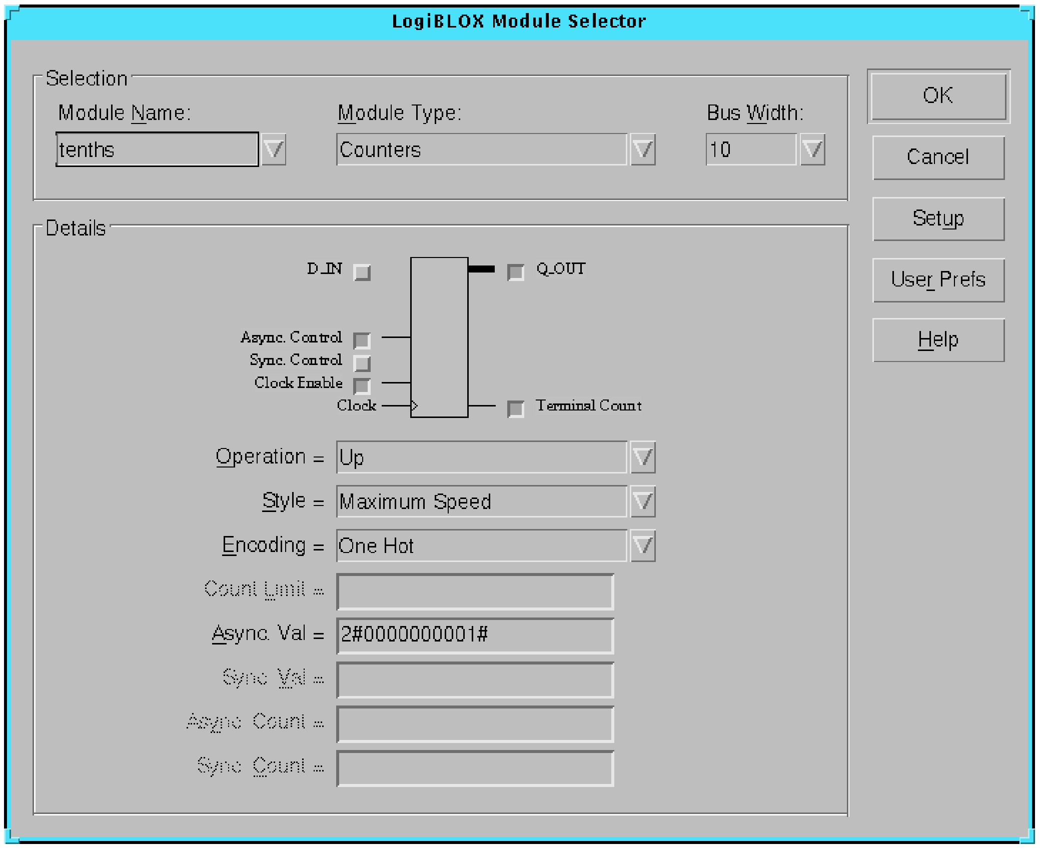
Bus Notation: B(I)

where
B is Bus Name
I is Bus Index

**Figure 1-1 LogiBLOX Setup Dialog Box**

7. In the LogiBLOX Module Selector dialog box (shown in “Logi-BLOX 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#



Module Name: tenths
Module Type: Counters
Bus Width: 10
Operation = Up
Style = Maximum Speed
Encoding = One Hot
Count Limit =
Async. Val = 2#0000000001#
Sync. Val =
Async. Count =
Sync. Count =

**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

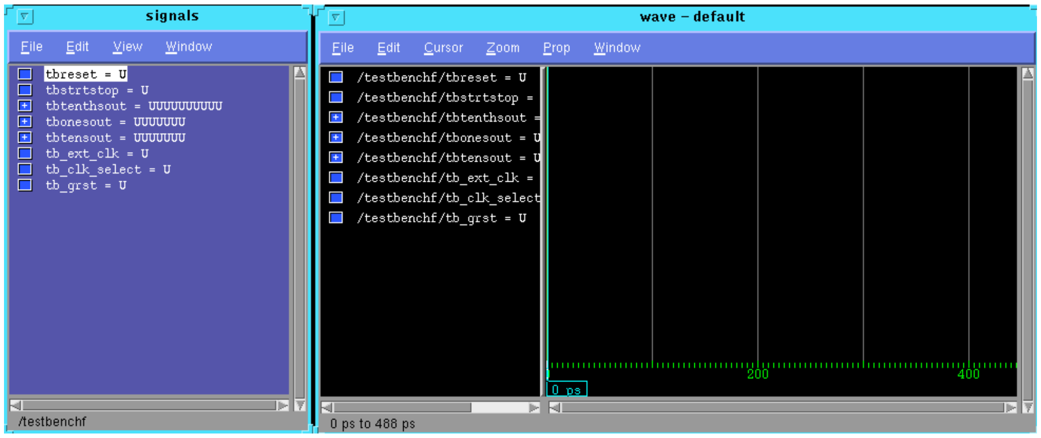
To perform the simulation do the following steps.

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.



```
tbreset = U
tbstrtstop = U
tbtenthsout = UUUUUUUUUU
tbonesout = UUUUUUU
tbtensout = UUUUUUU
tb_ext_clk = U
tb_clk_select = U
tb_grst = U
/testbenchf
```

```
/testbenchf/tbreset = U
/testbenchf/tbstrtstop =
/testbenchf/tbtenthsout =
/testbenchf/tbonesout = U
/testbenchf/tbtensout = U
/testbenchf/tb_ext_clk =
/testbenchf/tb_clk_select
/testbenchf/tb_grst = U
0 ps to 488 ps
```

**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.

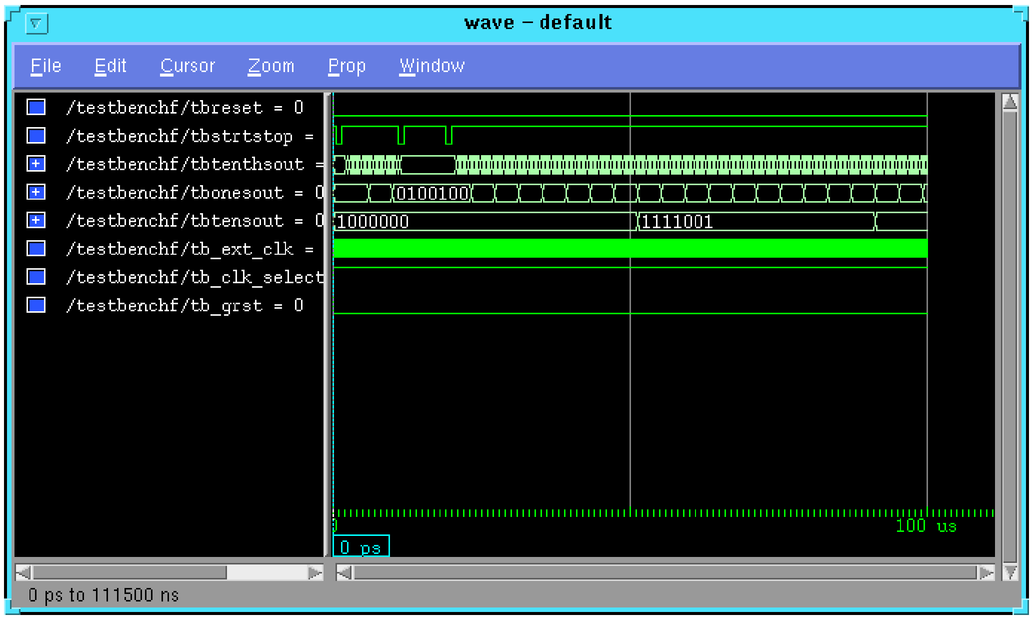


**Figure 1-4    Simulation Output in Wave Window**

## Synthesizing the Design Using Exemplar

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

- Leonardo
- Galileo Extreme
- Leonardo in Batch Mode

### Verilog tutorial

You will need to either add or make sure the Tenths module declaration within the file watch.v exists since 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 GUI

The following steps apply for UNIX and PC Leonardo GUI users unless otherwise specified.

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

   #### UNIX platform

   ```
   At the UNIX prompt type the following:
   ```

   ```
   leonardo &
   ```

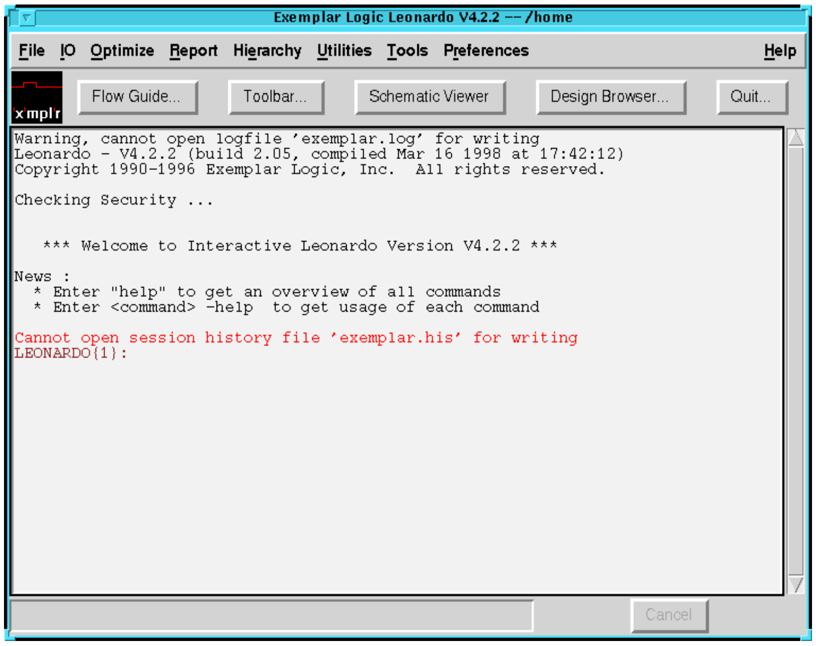
### PC platform

On a PC choose **Programs** → **Leonardo 4.2.2** → **Leonardo**

or at the Win 95 / NT prompt change to the directory where synthesis is going to be run and type the following:

**start leonardo**

The Exemplar Leonardo main window opens as shown in the following figure 1-5. The Toolbar is similar to the Flow Guide, because the ordering and executables run underneath, but the Flow Guide is more explanatory since each step has a description of what each step does. Usually when users become more familiar with the Leonardo tool, they use the Toolbar, or use a tcl script file to run the designs.



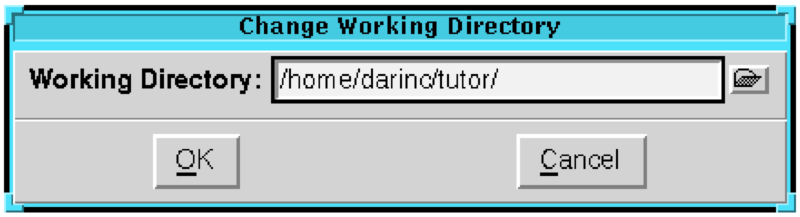
```
Warning, cannot open logfile 'exemplar.log' for writing
Leonardo - V4.2.2 (build 2.05, compiled Mar 16 1998 at 17:42:12)
Copyright 1990-1996 Exemplar Logic, Inc.  All rights reserved.

Checking Security ...


   *** Welcome to Interactive Leonardo Version V4.2.2 ***

News :
  * Enter "help" to get an overview of all commands
  * Enter <command> -help  to get usage of each command

Cannot open session history file 'exemplar.his' for writing
LEONARDO{1}:
```

**Figure 1-5 Leonardo Main Window**

2. Check the working directory setting. By default the working directory is set to the directory in which Leonardo was invoked in. To check the current work directory setting, look at the path which is displayed at the top of the Leonardo Main Window.

3. If necessary, change the working directory using one of the following methods.
   - At the Leonardo Main Window prompt type the following.

     **`cd`** *path_to_project_directory*
   - Select **`File`** → **`Change Working Directory`** to open the Change Working Directory dialog box as shown in the following figure 1-6. In this dialog box, browse to the directory or type in the full path.



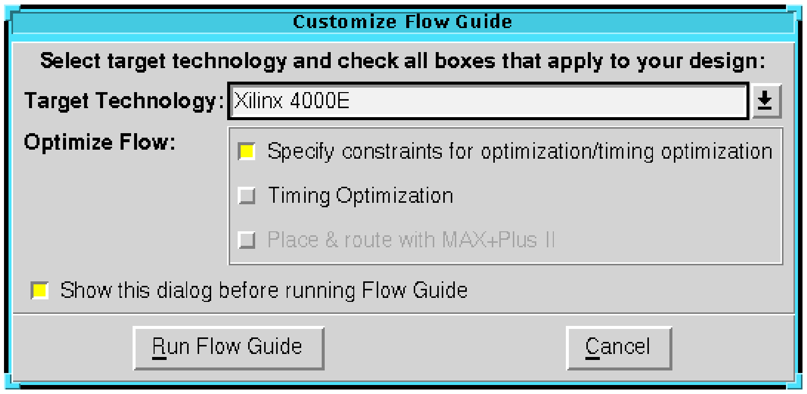
Change Working Directory

Working Directory: /home/darinc/tutor/

**Figure 1-6 Change Working Directory Dialog Box**

4. To run the design through using the Flow Guide, click on the Flow Guide button.

   The Customize Flow Guide dialog box opens as shown in the following figure 1-7. It contains options for customizing the flow. This is where you select the target technology.



Customize Flow Guide

Select target technology and check all boxes that apply to your design:

Target Technology: Xilinx 4000E

Optimize Flow:

[✓] Specify constraints for optimization/timing optimization

[ ] Timing Optimization

[ ] Place & route with MAX+Plus II

[✓] Show this dialog before running Flow Guide

**Figure 1-7 Customize Flow Guide Dialog Box**

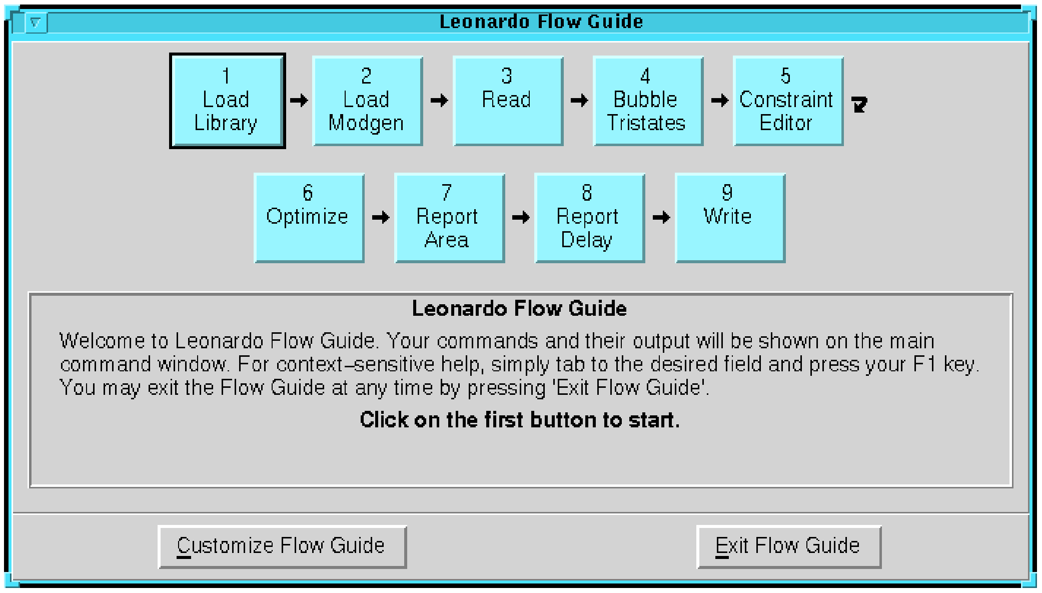
5. Since you are going to download this design to the demoboard, choose the following.

   Target Technology: Xilinx 4000E

6. Enable the Specify constraints for optimization / timing optimization option.

7. Click on the Run Flow Guide button.

   The Leonardo Flow Guide window appears as shown in the following figure 1-8.



Leonardo Flow Guide

Welcome to Leonardo Flow Guide. Your commands and their output will be shown on the main command window. For context–sensitive help, simply tab to the desired field and press your F1 key. You may exit the Flow Guide at any time by pressing 'Exit Flow Guide'.

**Click on the first button to start.**

**Figure 1-8 Leonardo Flow Guide Dialog Box**

The Flow Guide walks you through the steps, in order, to process the design starting in the upper left with the Load Library button and ending with Write. 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) **Load Library**—Reads a compiled Technology library file, then creates a library in Leonardo’s design database.

b) **Load Modgen**—Loads a technology specific module generator library description into the HDL database.

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

d) **Bubble Tristates**—Tcl script used to move tristates up in the design hierarchy, allowing you to optimize designs with buried tristate I/O's without flattening. Also allows you to transform internal tristates to MUX logic without flattening. This is not necessary to run the flow

e) **Constraint Editor**—Allows you to specify user-defined constraints in the design. This option only appears in the Flow Guide if the Specify constraints for optimizing/timing optimization option is selected, or generate_timespec is applicable.

f) **Optimize**—Performs technology-specific logic optimization and technology mapping.

g) **Optimize Timing**—Performs extensive timing optimization on the design. This only appears if Timing Optimization is selected.

h) **Report Area**—Reports the accumulated area of the present design.

i) **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.

j) **Write**—Writes the output netlist in the user specified format.

**Note:** Running the Optimize Timing command requires that you separately purchase the Leonardo Timing Module.

8. After the Flow Guide opens, load the library by clicking on the Load Library button.

   The Load Library dialog box opens as shown in the following figure 1-9. The Technology should already be set to Xilinx 4000E.

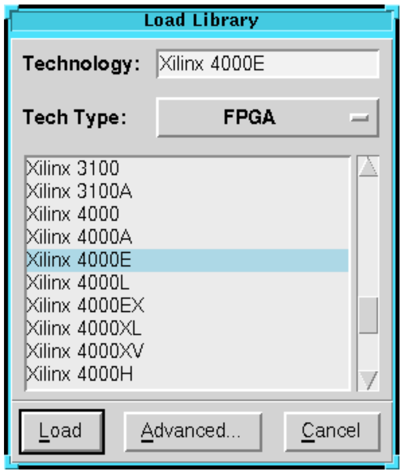


**Figure 1-9 Load Library Dialog Box**

9. Click on the Load button.

   Notice that as each button or step finishes in the Flow Guide (figure 1-8), the button turns a different color.

10. In the Flow Guide, click on the Load Modgen button.

    The Load Modgen window opens. The Modgen Library should already be set to Xilinx 4e. This window is very similar to the Load Library window.

11. Click on the Load button.

12. Click on the Read button.

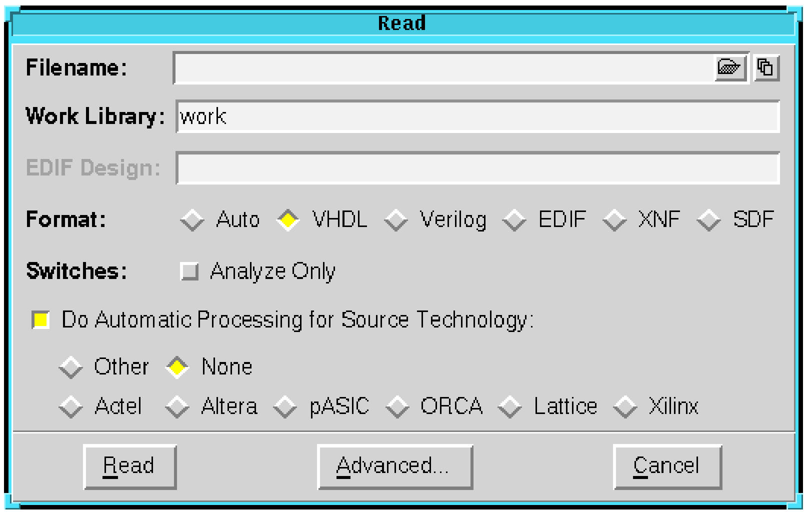
    The Read dialog box opens as shown in the following figure 1-10.

### Verilog tutorial

As a general rule the files should be read in from the bottom up, such that the top level files is read in last. The files can be read in individually or all at once.

### VHDL tutorial

The VHDL file must be read in from the bottom up, so the top level VHDL file will be read in last. In the Read window, the files can be read in individually or all at once. If the files are all read in at once, the files must be in order from the bottom most level to the top level.



Read

Filename:

Work Library: work

EDIF Design:

Format: Auto [✓ VHDL] Verilog EDIF XNF SDF

Switches: Analyze Only

[✓] Do Automatic Processing for Source Technology:

Other [✓ None]

Actel Altera pASIC ORCA Lattice Xilinx

**Figure 1-10 Leonardo Read Dialog Box**

13. In the Read dialog box Format option, select the appropriate option for Verilog or VHDL.

14. At this point you can either read-in individual files or multiple files. The two methods are described following.

**To Read-In Individual Files**

a) In the Read dialog box on the right side of the Filename field is a single file icon, and a multi sheet icon. To read in an individual file click on the single file icon.

b) In the Select a File dialog box that opens, select the file you wish to read-in by highlighting the file and clicking the OK button.

c) In the Read dialog box, click on the Read button.

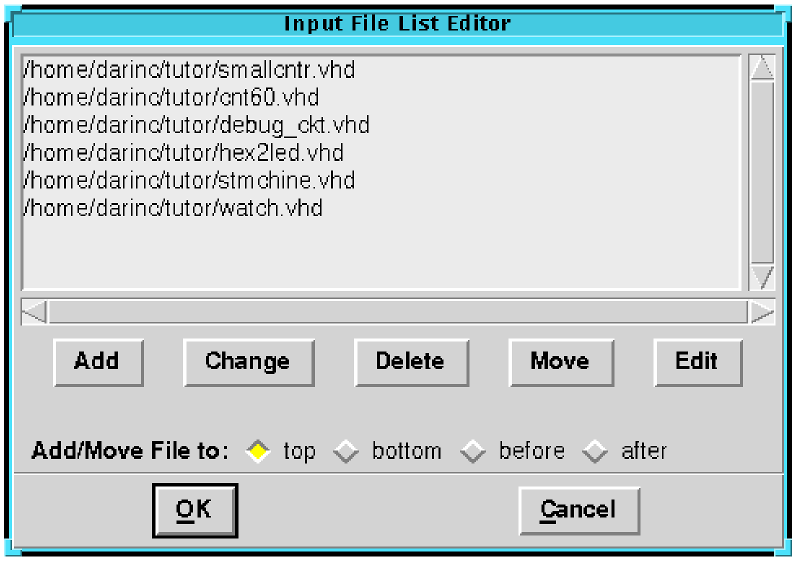
d) If you wish to read in the files individually for this tutorial read them in following order for Verilog or VHDL:

smallcntr, cnt60, debug_ckt, hex2led, stmachine, watch.

**To Read-In Multiple Files**

a) In the Read dialog box on the right side of the Filename field, choose the multiple sheets icon.

b) In the Input File List Editor dialog box that opens as shown in the following figure 1-11, select the Add button.


Input File List Editor

```
/home/darinc/tutor/smallcntr.vhd
/home/darinc/tutor/cnt60.vhd
/home/darinc/tutor/debug_ckt.vhd
/home/darinc/tutor/hex2led.vhd
/home/darinc/tutor/stmchine.vhd
/home/darinc/tutor/watch.vhd
```

Add/Move File to: [✓ top] bottom before after

**Figure 1-11 Input File List Editor Dialog Box**

c) To add the HDL design files do the following:

### Verilog tutorial

To add all the Verilog files, select the file cnt60.v by highlighting it. Next hold down the keyboard shift-key, then left mouse click and select the watch.v file. Once all the Verilog files are highlighted click on the OK button.

All the selected files should now show in the Input File List Editor dialog box.

### VHDL tutorial

To add all the VHDL files, select the file cnt60.vhd by highlighting it. Next hold down the keyboard shift-key, then left mouse click and select the watch.vhd file. Once all the VHDL files are highlighted click on the OK button.

All the selected files should now show in the Input File List Editor dialog box.

d) Now you must change the order to reflect the order in Reading In Individual Files, step d). To move a file, highlight the file to be moved, select the appropriate Add / Move File to option, and click on Move.

e) When the order is correct click on OK in the Input File List Editor dialog box.

In the Read dialog box, the Filename box contains the files added, but you have to scroll right to see all of them.

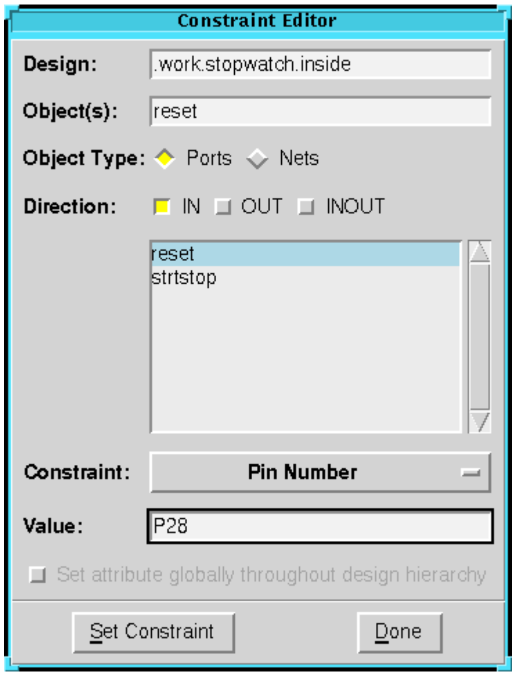
f) In the Read dialog box, click on the Read to read all the files into Leonardo's design database.

**Note:** You may get some warnings that can be ignored during the READ operation.

15. Optionally after reading-in the files, you can run the Schematic Viewer to view how the HDL was read in. This could be useful to try and determine if Optimize is removing some logic you want to keep. See step 22 in this procedure for more information on using the Schematic Viewer.

16. In the Flow Guide, click on the Constraints Editor button to open the Constraints Editor dialog box which is shown in the following figure 1-12.

    In this tutorial you will use the Constraint Editor to lock some signals to pins. To save time, the tutorial will only have you lock a couple of pins, and use a .ucf file to lock the rest of the pins.



Design: .work.stopwatch.inside

Object(s): reset

Object Type: [✓ Ports] Nets

Direction: [✓ IN] OUT INOUT

reset
strtstop

Constraint: Pin Number

Value: P28

**Figure 1-12 Constraint Editor**

17. To lock the reset signal to pin 28 and the strtstop signal to pin 18, follow these steps in the Constraint Editor.

    a) Specify the Direction IN.

    b) Select the signal reset.

    c) In the Constraint field select Pin Number.

    d) In the Value field enter P28

    e) Click on the Set Constraint button.

    f) Repeat steps a through e for the signal strtstop and lock it to P18.

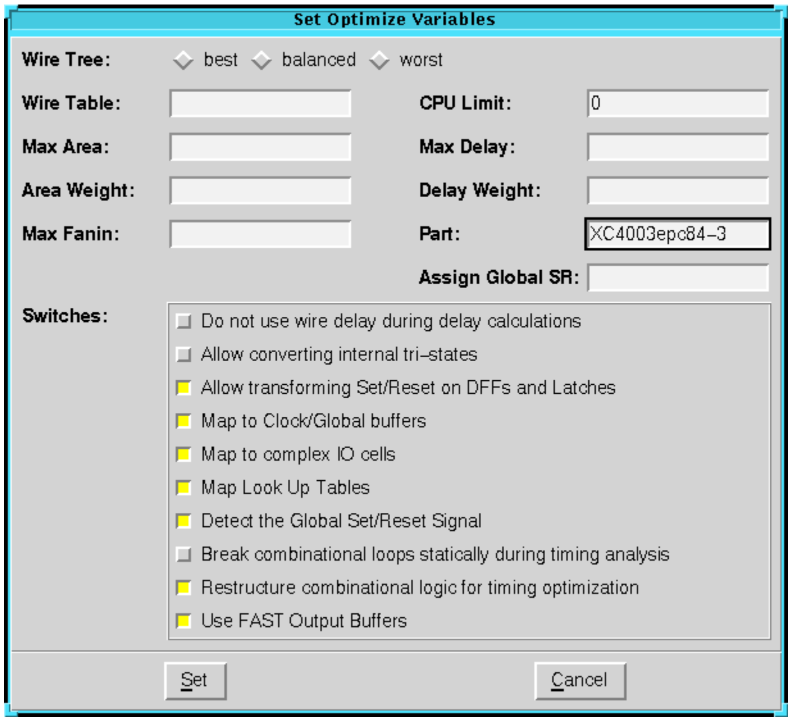
    g) After setting the pin locking constraints, click on the Done button to write these constraints to the EDIF file.

18. Click on the Optimize button in the Flow Guide window and verify that the following options are selected.

- Target: Xilinx 4000E
- Effort: Quick
- Mode: Chip
- Optimize: Area

19. Click on the Advanced button in the Flow Guide window to open the Set Optimize Variables dialog box which is shown in the following figure 1-13.

    In this dialog box, you specify the part you are targeting, such that the Xilinx tools will read the part number in. Specify 4003EPC84-3 for this tutorial.


Set Optimize Variables

Wire Tree: best balanced worst

| | | | |
|---|---|---|---|
| Wire Table: | | CPU Limit: | 0 |
| Max Area: | | Max Delay: | |
| Area Weight: | | Delay Weight: | |
| Max Fanin: | | Part: | XC4003epc84–3 |
| | | Assign Global SR: | |

Switches:

- [ ] Do not use wire delay during delay calculations
- [ ] Allow converting internal tri–states
- [x] Allow transforming Set/Reset on DFFs and Latches
- [x] Map to Clock/Global buffers
- [x] Map to complex IO cells
- [x] Map Look Up Tables
- [x] Detect the Global Set/Reset Signal
- [ ] Break combinational loops statically during timing analysis
- [x] Restructure combinational logic for timing optimization
- [x] Use FAST Output Buffers

**Figure 1-13 Set Optimize Variables Dialog Box**

Normally a Wire Table would be inserted, but there is no wire load table for the XC4003e-3 so let Exemplar use the default wire table for the 4000E.

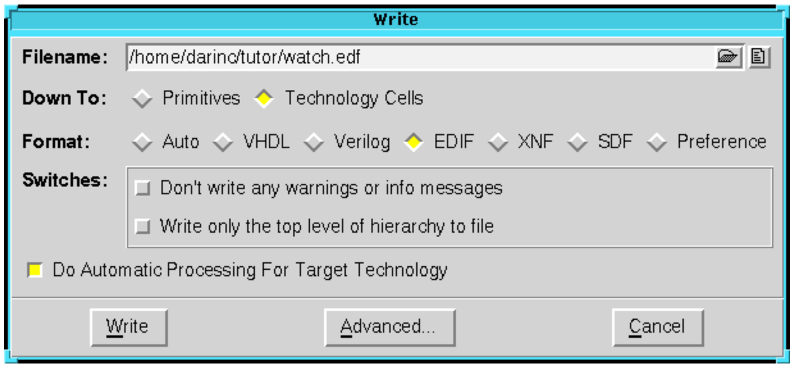
20. Click on the Set button.

21. Click on the Optimize button in the Optimize window.

    The design will now be optimized and mapped for the Xilinx 4000E.

22. Optionally after Optimize, you can run the Schematic viewer to verify what synthesis has done to the design. To start the Schematic Viewer either click on the Schematic Viewer icon, or go Tools → Schematic Viewer. There may be more than one page. Use the buttons near the top center of the Viewer to scroll to other pages. In the case of this tutorial there are macros instantiated in a top level. The schematic for the macros can be viewed by double clicking on the macro. This opens the schematic for the macro. To go back to the top level use the left command buttons, Show Top or Go up.

23. In the Leonardo Flow Guide main window, click on the Write button to open the Write dialog box which is shown in the following figure 1-14.



Write

Filename: /home/darino/tutor/watch.edf

Down To: Primitives [✓ Technology Cells]

Format: Auto VHDL Verilog [✓ EDIF] XNF SDF Preference

Switches: ☐ Don't write any warnings or info messages

☐ Write only the top level of hierarchy to file

[✓] Do Automatic Processing For Target Technology

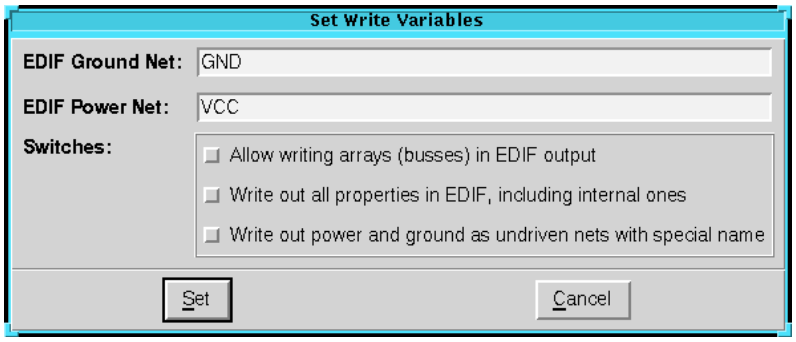
**Figure 1-14 Leonardo Write Dialog Box**

24. Now you can write out an EDIF netlist to be read into the Xilinx tools. A user defined filename will be written to. The file extension name must be given as well. Set the following options in the Write dialog box.

    - Filename: watch.edf
    - Down To: Technology Cells

- Format: EDIF

25. Click on the Advanced button in the Write dialog box to open the Set Write Variables dialog box shown in the following figure 1-15.



EDIF Ground Net: GND

EDIF Power Net: VCC

Switches:

- ☐ Allow writing arrays (busses) in EDIF output
- ☐ Write out all properties in EDIF, including internal ones
- ☐ Write out power and ground as undriven nets with special name

**Figure 1-15 Set Write Variables Dialog Box**

26. In the Set Write Variables dialog box, de-select the Allow writing arrays (busses) in EDIF output option.

    Failing to do this causes an unexpanded error in the Xilinx tools.

27. Click on the Set button in the Set Write Variables window.

28. Click on the Write button in the Write window to write the file watch.edf

**Note:** When an HDL design has LogiBLOX or Coregen modules with busses, de-select the Advanced Write option for allowing writing of arrays (busses) in the EDIF output.

## Galileo Extreme GUI

1. Start-up the Galileo Extreme GUI using the Leonardo license by typing the following:

   **UNIX platform**

   At the UNIX prompt:

   ```
   galileo -product leonardo &
   ```

   The Galileo Extreme Main Window opens. figure 1-16.

### PC platform

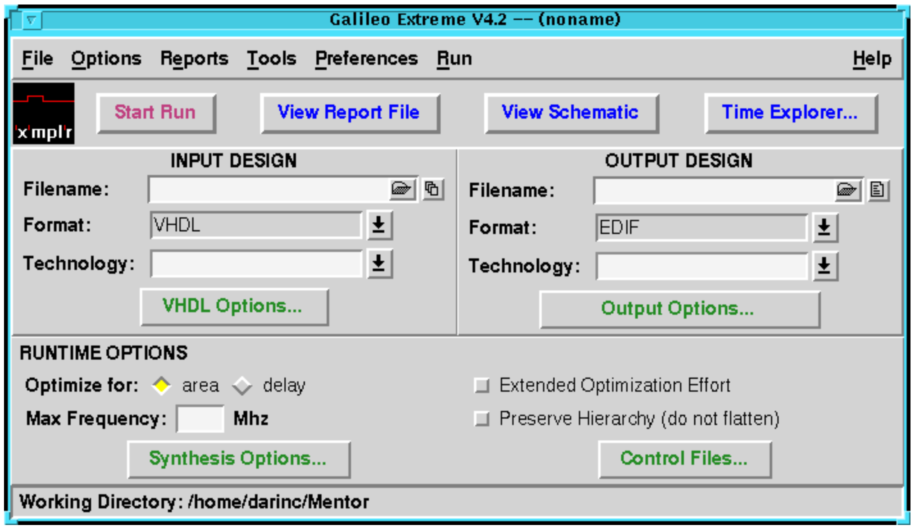
At the Win 95 / NT prompt:

**galileo -product leonardo**

or Choose the following:

**Start** → **Programs** → **Leonardo 4.2.2** → **Galileo**

The Galileo Extreme Main Window opens, figure 1-16.



INPUT DESIGN
Filename:
Format: VHDL
Technology:

OUTPUT DESIGN
Filename:
Format: EDIF
Technology:

RUNTIME OPTIONS
Optimize for: [✓ area] delay
Max Frequency: Mhz
Extended Optimization Effort
Preserve Hierarchy (do not flatten)

Working Directory: /home/darinc/Mentor

**Figure 1-16 Galileo Extreme Main Window**

2. Set the working directory. By default the working directory is the directory in which you invoke Galileo. The current work directory location is displayed in the lower left part of the Galileo Extreme window. To change the work directory select **`File`** → **`Change Working Directory`** and either browse to the directory, or type in the full path to the project directory.

3. Look at the Input Design portion of the Galileo Main Window. This is where you specify input design files, Format, technology, and VHDL input design options.

4. Specify the Input Design Filename(s). You can either specify an individual file or multiple files as described below.

To read in an individual file either type a filename in the File-name box, or click on the File Selector button to open the Select A File dialog box in which you can browse and select a file to read in.

To Read In Multiple Files, click on the multiple sheet button on the right side of the Filename box to open the Input File List Editor dialog box (Shown in the “Input File List Editor” figure 1-17). Select the Add button to open the Select a File dialog box.

### Verilog tutorial

To add all the Verilog files, select the file cnt60.v by highlighting it, hold down the keyboard shift-key, then left mouse click and select the watch.v file. When all the Verilog files are highlighted click on the OK button. All the selected files shold now appear in the Input File List Editor window.

Now you can change the order of the files to the following order:

```
smallcntr.v
cnt60.v
debug_ckt.v
hex2led.v
stmachine.v
watch.v
```

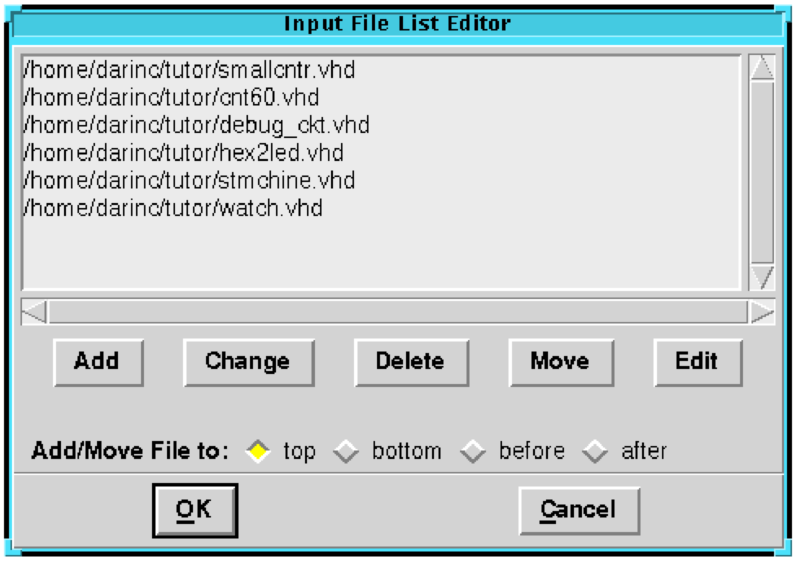
### VHDL tutorial

To add all the VHDL files, select the file cnt60.vhd by high-lighting it, hold down the keyboard shift-key, then left mouse click and select the watch.vhd file. When all the VHDL files are highlighted click on the OK button. All the selected files should now appear in the Input File List Editor window.

Now you must change the order of the files to the following order.

```
smallcntr.vhd
cnt60.vhd
debug_ckt.vhd
hex2led.vhd
stmachine.vhd
watch.vhd
```

To move a file, highlight the file to be moved, select the appropriate Add / Move File to option, and click on Move.

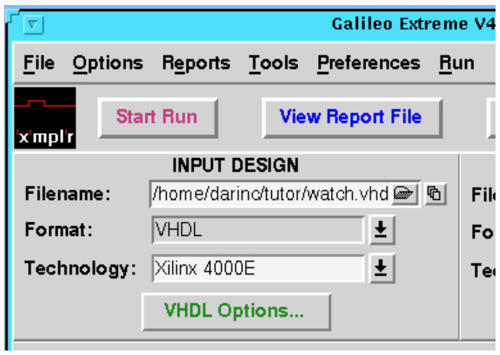

/home/darinc/tutor/smallcntr.vhd
/home/darinc/tutor/cnt60.vhd
/home/darinc/tutor/debug_ckt.vhd
/home/darinc/tutor/hex2led.vhd
/home/darinc/tutor/stmchine.vhd
/home/darinc/tutor/watch.vhd

Add/Move File to: [✓ top] bottom before after

**Figure 1-17 Input File List Editor**

5. When the order is correct, click on OK in the Input File List Editor dialog box.

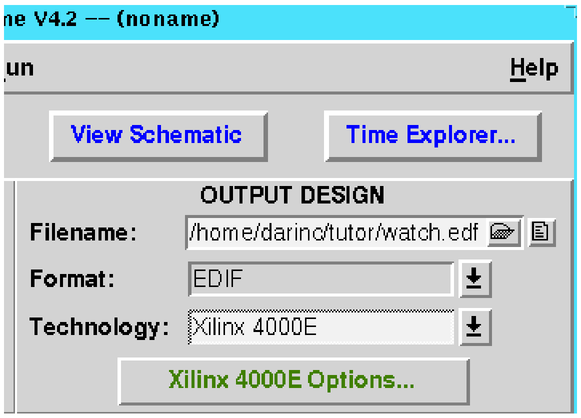
   The Input Design Filename field in the Input Design section of the Galileo Extreme window now contains all the .v or .vhd files selected, but you have to scroll to see all of the files. This is shown in the following figure 1-18.



Filename: /home/darino/tutor/watch.vhd
Format: VHDL
Technology: Xilinx 4000E

**Figure 1-18 Specifying Input Design Settings**

6. For the Format, select the appropriate Veriog or VHDL.
7. For the Technology, select Xilinx 4000E.
8. Look at the Output Design portion of the Galileo Main Window. This is where you specify the output filename, the format of the output file, the technology, and output options.



Filename: /home/darino/tutor/watch.edf
Format: EDIF
Technology: Xilinx 4000E

**Figure 1-19 Galileo Main Window with Output Design Settings**

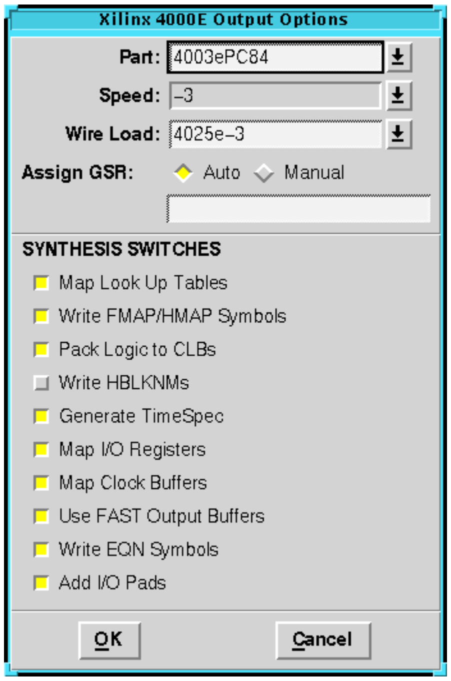
9. Change the filename to watch.edf as shown in the preceding figure 1-19. By default, the Filename box is set to the path to the directory where the .vhd files reside, and will write to the file of the last, or top level, .vhd file, with the extension of the format, EDIF in this case.

10. Set the Output Design options as follows.

    - Format: EDIF
    - Technology: Xilinx 4000E
    - In the Xilinx 4000E options icon the Part type can be specified:

11. In the Output Design Area of the Galileo Main Window, click on the Xilinx 4000E Options button and make the following settings in the Xilinx 4000E Output Options dialog box that opens (see the following figure 1-20).

    - Part: 4003ePC84
    - Speed: -3
    - Wire Load: 4025e-3 (this is default)
    - The Synthesis Switches can be left at default.



Xilinx 4000E Output Options

Part: 4003ePC84

Speed: -3

Wire Load: 4025e-3

Assign GSR: [✓ Auto] Manual

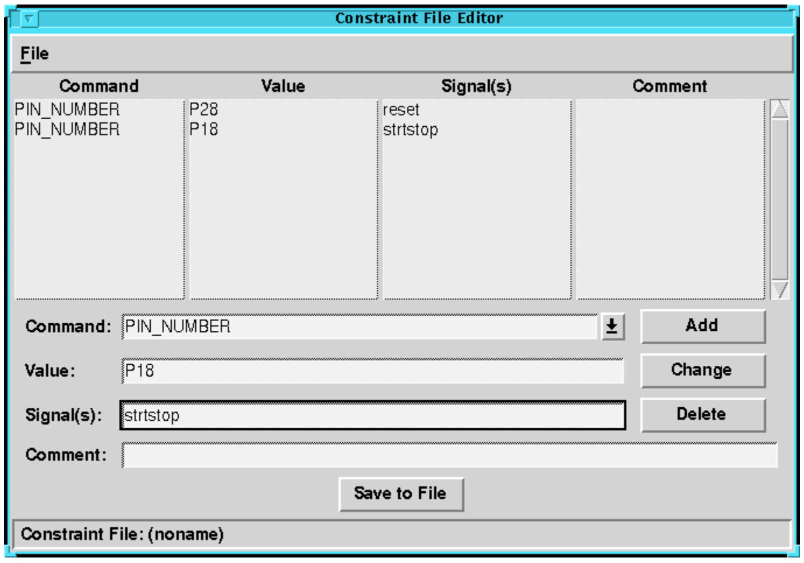
SYNTHESIS SWITCHES

- [✓] Map Look Up Tables
- [✓] Write FMAP/HMAP Symbols
- [✓] Pack Logic to CLBs
- [ ] Write HBLKNMs
- [✓] Generate TimeSpec
- [✓] Map I/O Registers
- [✓] Map Clock Buffers
- [✓] Use FAST Output Buffers
- [✓] Write EQN Symbols
- [✓] Add I/O Pads

**Figure 1-20    Xilinx 4000E Output Options**

12. Click on the OK button.
13. Next, you will use the Constraints editor to lock a couple of signals down to some pins. You can also use this for timing constraints to control synthesis. To save time the rest of the pins are locked in a UCF file, which is read in by the Xilinx tools.
14. To lock the signal reset to pin 28 and signal strtstop to pin 18, Select **`Tools`** → **`Constraint File Editor`**.

    The Constraints File Editor opens as shown in the following figure 1-21.



| Command | Value | Signal(s) | Comment |
|---|---|---|---|
| PIN_NUMBER | P28 | reset | |
| PIN_NUMBER | P18 | strtstop | |

Command: PIN_NUMBER
Value: P18
Signal(s): strtstop
Comment:
Constraint File: (noname)

**Figure 1-21 Constraint File Editor**

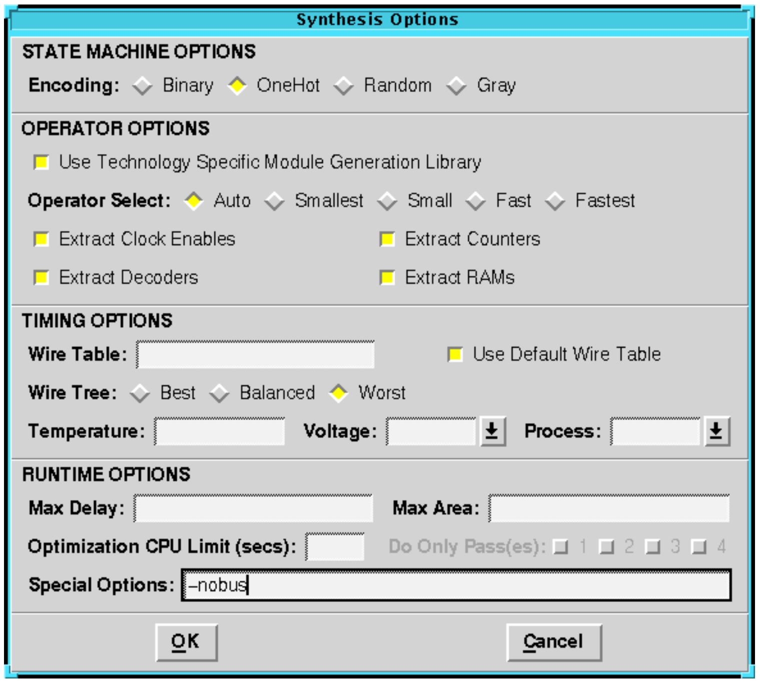
15. In the Command field in the Constraint File Editor dialog box, use the down arrow to scroll down to PIN_NUMBER and make the following settings.
    - Value: P28
    - In the Signal(s) type in reset
16. Click the Add button and the constraint appears in the top portion of the Constraint File Editor window.
17. Repeat the procedure for the signal strtstop and lock it to pin 18.

18. Click on the Save to File button and enter the name watch.ctr.

    This is now saved as a control file, and actually will be read in automatically if it has the same name as the input filename. If the file is saved as a different name this can be specified and will be shown in step 24.

19. To exit the Constraint File Editor, select **`File`** → **`Close`**.

20. In the Galileo Extreme window, click on the Synthesis Options button to open the Synthesis Options dialog box which is shown in the following figure 1-22.



Synthesis Options

STATE MACHINE OPTIONS

Encoding: Binary [✓ OneHot] Random Gray

OPERATOR OPTIONS

[✓] Use Technology Specific Module Generation Library

Operator Select: [✓ Auto] Smallest Small Fast Fastest

[✓] Extract Clock Enables [✓] Extract Counters

[✓] Extract Decoders [✓] Extract RAMs

TIMING OPTIONS

Wire Table: [✓] Use Default Wire Table

Wire Tree: Best Balanced [✓ Worst]

Temperature: Voltage: Process:

RUNTIME OPTIONS

Max Delay: Max Area:

Optimization CPU Limit (secs): Do Only Pass(es): 1 2 3 4

Special Options: –nobus

**Figure 1-22 Synthesis Options Dialog Box**

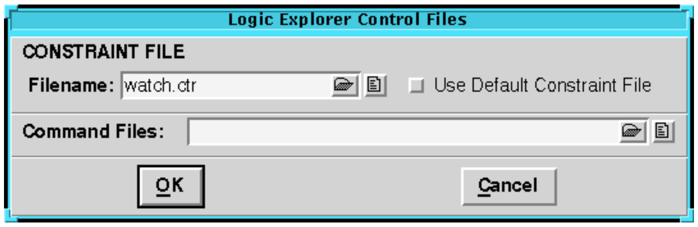
21. Look at the Runtime Options area. This is where you can specify optimization options, target frequency, synthesis options, and control files, if any.

22. In the Runtime Options area, keep the default option settings, but in the Special Options field, type.

    **-nobus**

This prevents writing of arrays (busses) in the EDIF file, writing the busses in expanded form. Failing to do this will cause unexpanded errors in the Xilinx tools. This can happen anytime there is a black-box instantiations netlist that is written bit-blasted, each bit of a bus, and the top level netlist is written in bus arrays.

23. Click OK to close the Synthesis Options dialog box.
24. In the Galileo Extreme window, click on the Control Files button, to open the Logic Explorer Control Files dialog box which is shown in the following figure 1-23.


Logic Explorer Control Files

CONSTRAINT FILE

Filename: watch.ctr

Use Default Constraint File

Command Files:

**Figure 1-23 Logic Explorer Control Files Dialog Box**

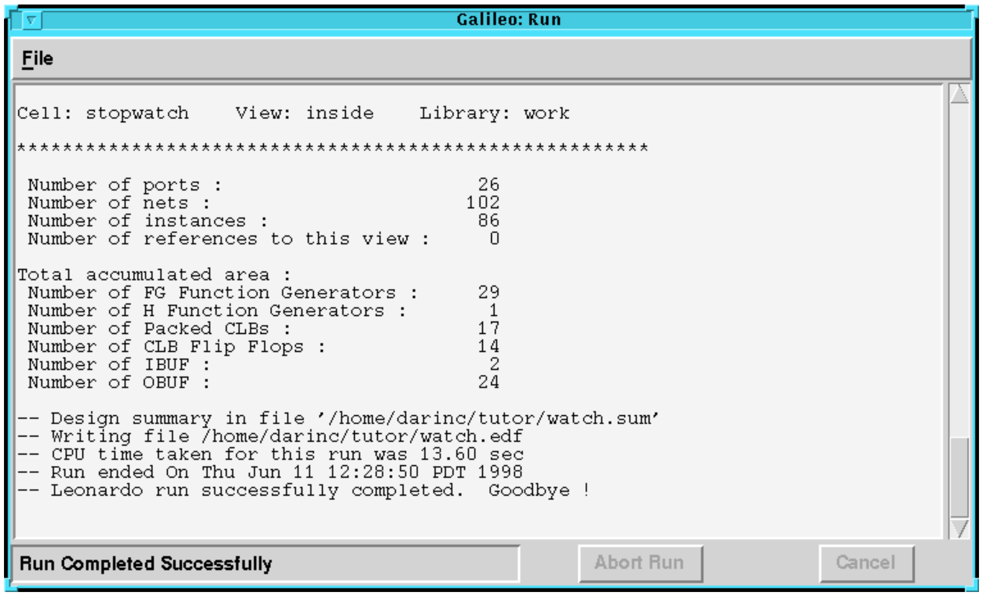
25. In the filename field, click on the single file button and in the Select A File dialog box that opens, highlight the file watch.ctr.
26. Click OK in the Select A File window and then OK in the Logic Explorer Control File dialog box.

    Now you have set all the necessary options and you are ready to synthesize the design.

27. To synthesize the design, click on the Start Run button.

    The Galileo Run window opens, and displays information as the synthesis runs. This screen output will be piped to a log file with the top level name, watch.log in the working directory.

    When Synthesis is done and the EDIF file has been written, the Galileo window shows something similar to the following figure 1-24.



```
Cell: stopwatch    View: inside    Library: work
*******************************************************

 Number of ports :                           26
 Number of nets :                           102
 Number of instances :                       86
 Number of references to this view :          0

Total accumulated area :
 Number of FG Function Generators :          29
 Number of H Function Generators :            1
 Number of Packed CLBs :                     17
 Number of CLB Flip Flops :                  14
 Number of IBUF :                             2
 Number of OBUF :                            24

-- Design summary in file '/home/darinc/tutor/watch.sum'
-- Writing file /home/darinc/tutor/watch.edf
-- CPU time taken for this run was 13.60 sec
-- Run ended On Thu Jun 11 12:28:50 PDT 1998
-- Leonardo run successfully completed.  Goodbye !
```

**Figure 1-24 Galileo Run Window**

28. If you wish to view how the design was synthesized, you can click on the View Schematic button.

    The Netscope window opens and displays the schematic. There will be several pages and you can change the pages with the controls at the top of the Netscope window.

**Note:** In Galileo the Schematic Viewer can only view a design that has already been synthesized. Leonardo can view a design after READ or after Optimize.

## 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 elsyn. This script file has already been created, called synthesis.tcl.

To run Leonardo in script mode, type the following from the MS-DOS prompt or from the UNIX prompt.

```
elsyn -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.

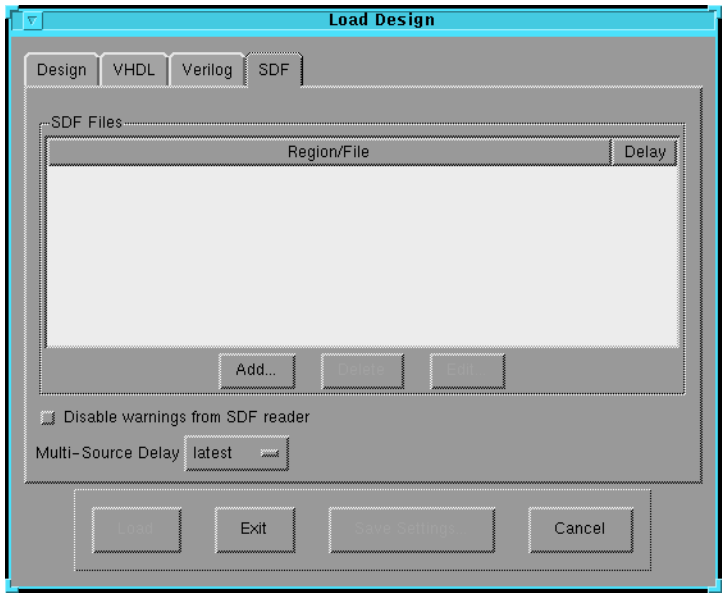
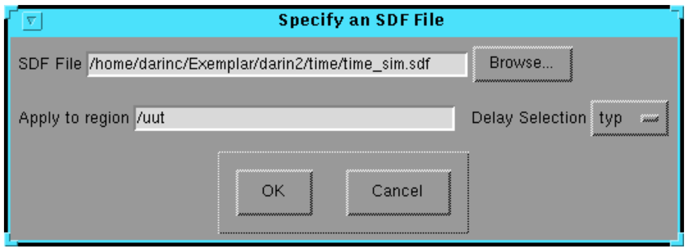


**Figure 1-25 Load Design Dialog Box**

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



SDF File /home/darinc/Exemplar/darin2/time/time_sim.sdf

Apply to region /uut

Delay Selection typ

**Figure 1-26 Specifying the SDF File**

b) Click the Add button.

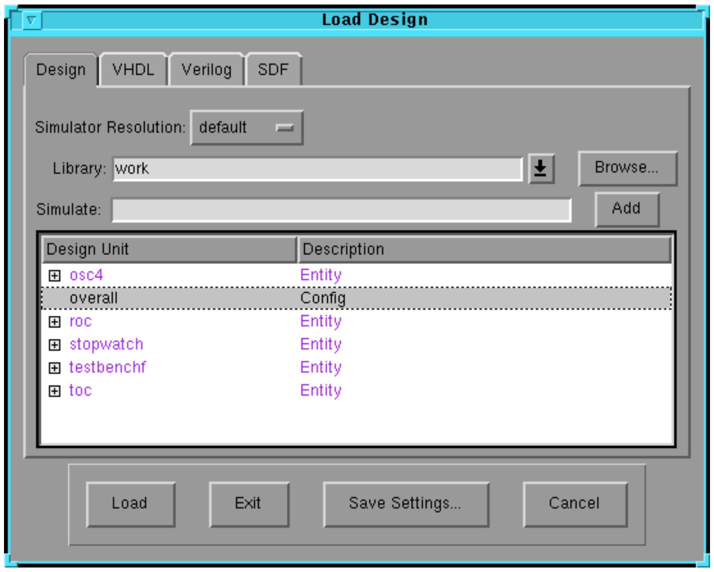
c) Browse for the time_sim.sdf file and select it.

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.



Simulator Resolution: default

Library: work

Simulate:

| Design Unit | Description |
|---|---|
| ⊞ osc4 | Entity |
| overall | Config |
| ⊞ roc | Entity |
| ⊞ stopwatch | Entity |
| ⊞ testbenchf | Entity |
| ⊞ toc | Entity |

**Figure 1-27 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!