Using Model Technology Model*Sim* with Xilinx Foundation Series Software

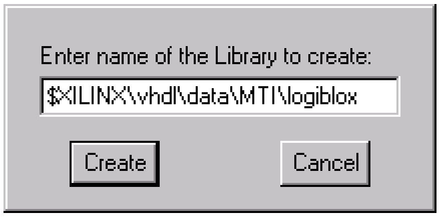
Xilinx Library Setup

Because the Model*Sim* VHDL and Verilog simulators use compiled HDL libraries, each of the Xilinx HDL simulation libraries must be compiled before simulating any HDL descriptions containing Xilinx specific parts. This is a task that should only be done once, and then updated only if new Xilinx libraries are required.

The following sections describe how to set up Xilinx libraries for use in Model*Sim* simulations. The first three sections describe the step-by-step instructions for compiling the Xilinx simulation model libraries. The forth section describes how to setup and run an automated script to accomplish the same task.

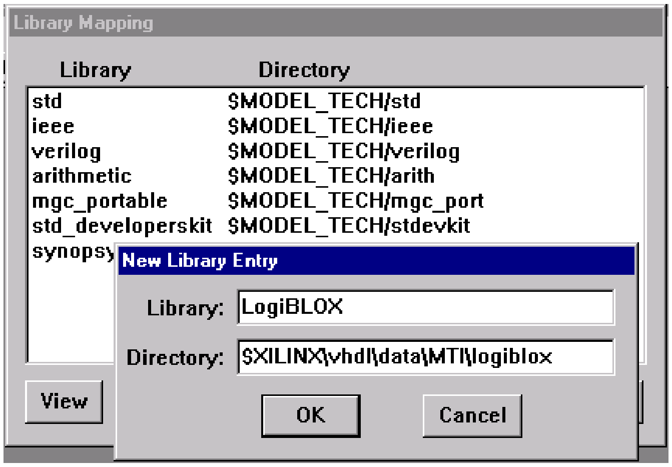
Section 1. Compiling the LogiBLOX library

1. Using the Windows Explorer, make an MTI directory in **`$XILINX\vhdl\data`**. If you have installed the Foundation Series software in the default location, this will resolve to **`C:\FNDTN\vhdl\data\MTI`**
2. After starting Model*Sim*, select File -> Directory and browse over to $MODELSIM.
3. In Model*Sim*, select Library -> New.
4. Type **`$XILINX/vhdl/data/MTI/logiblox`**.



Enter name of the Library to create: $XILINX\vhdl\data\MTI\logiblox

5. Select Library -> Mapping and press New.
6. For Library, enter LogiBLOX. For Directory, enter **`$XILINX/vhdl/data/MTI/logiblox`**.



Library Mapping

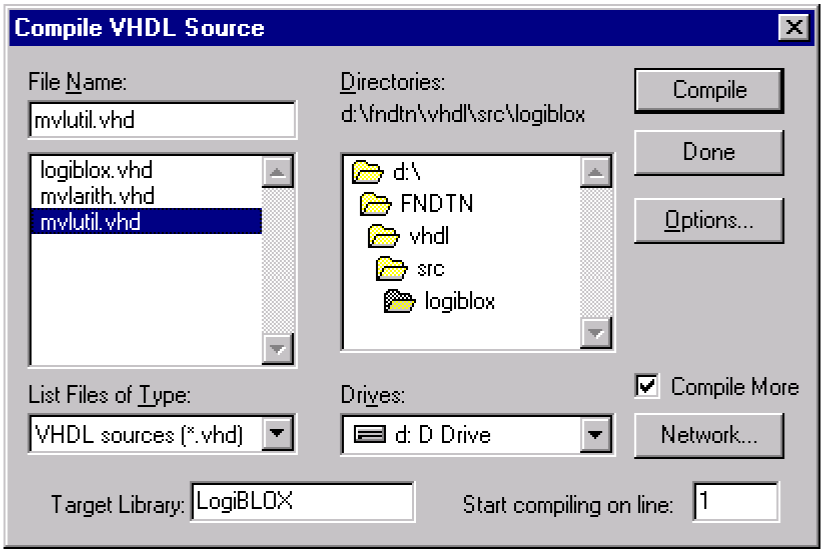
Library	Directory
std	$MODEL_TECH/std
ieee	$MODEL_TECH/ieee
verilog	$MODEL_TECH/verilog
arithmetic	$MODEL_TECH/arith
mgc_portable	$MODEL_TECH/mgc_port
std_developerskit	$MODEL_TECH/stdevkit
synopsy	

New Library Entry

Library: LogiBLOX

Directory: $XILINX\vhdl\data\MTI\logiblox

7. From the Model*Sim* application window, press the VCOM button on the toolbar.
8. In the **Compile VHDL Source** dialog window, set the Target Library to LogiBLOX.
9. In the **Directories** window, change to **`$XILINX\vhdl\src\logiblox`**.



Compile VHDL Source

File Name: mvlutil.vhd

logiblox.vhd
mvlarith.vhd
mvlutil.vhd

Directories: d:\fndtn\vhdl\src\logiblox

d:\
FNDTN
vhdl
src
logiblox

List Files of Type: VHDL sources (*.vhd)

Drives: d: D Drive

☑ Compile More

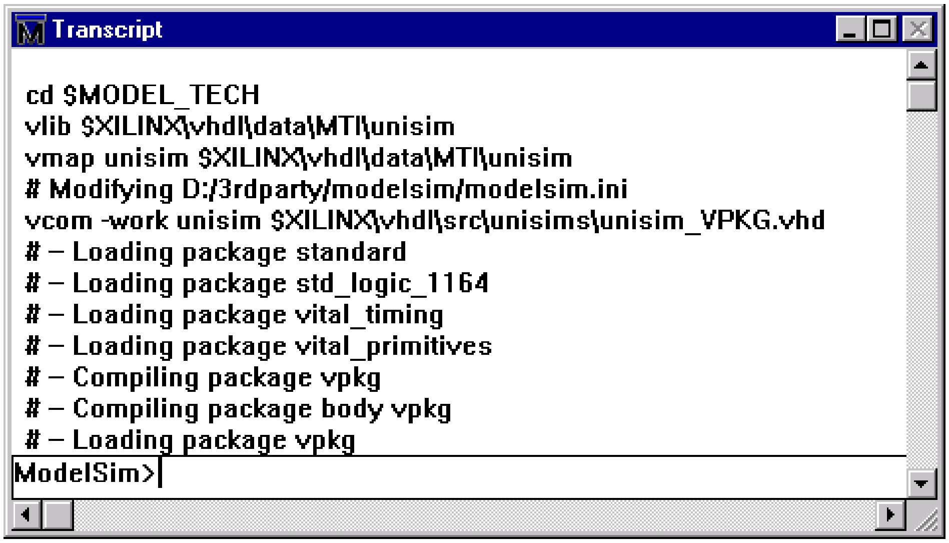
Target Library: LogiBLOX

Start compiling on line: 1

10. Use the **Compile** to compile the VHDL files in the order they are shown:
    mvlutil.vhd
    mvlarith.vhd
    logiblox.vhd

Section 2. Compiling the Unisim library:

As you may have noticed, a transcript of the work you have been doing has appeared in the Model*Sim* Transcript window. The transcript window records the commands used to accomplish the same tasks. To accomplish the remainder of the Xilinx library setup, type the following commands into the Model*Sim* Transcript window:



Transcript

```
cd $MODEL_TECH
vlib $XILINX\vhdl\data\MTI\unisim
vmap unisim $XILINX\vhdl\data\MTI\unisim
# Modifying D:/3rdparty/modelsim/modelsim.ini
vcom -work unisim $XILINX\vhdl\src\unisims\unisim_VPKG.vhd
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Loading package vital_timing
# -- Loading package vital_primitives
# -- Compiling package vpkg
# -- Compiling package body vpkg
# -- Loading package vpkg
ModelSim>
```

```
cd $MODEL_TECH
vlib $XILINX/vhdl/data/mti/unisim
vmap unisim $XILINX/vhdl/data/mti/unisim
vcom -work unisim $XILINX/vhdl/src/unisims/unisim_VPKG.vhd
vcom -work unisim $XILINX/vhdl/src/unisims/unisim_VCOMP.vhd
vcom -work unisim $XILINX/vhdl/src/unisims/unisim_VITAL.vhd
vcom -work unisim $XILINX/vhdl/src/unisims/unisim_VCFG4K.vhd
```

Section 3. Compiling the Simprims library:

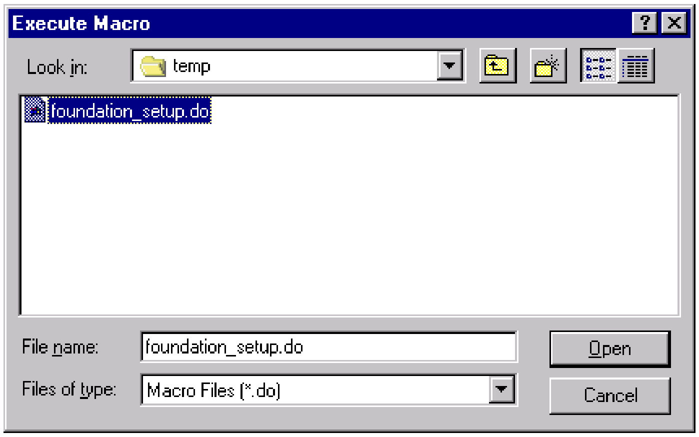
The following commands are those used to compile the Xilinx simprim VHDL libraries.

```
cd $MODEL_TECH
vlib $XILINX/vhdl/data/MTI/simprim
vmap simprim $XILINX/vhdl/data/mti/simprim
vcom -work simprim $XILINX/vhdl/src/simprims/simprim_Vpackage.vhd
vcom -work simprim $XILINX/vhdl/src/simprims/simprim_Vcomponents.vhd
vcom -work simprim $XILINX/vhdl/src/simprims/simprim_VITAL.vhd
```

Section 4. Using a macro file

If you place all the above ModelSim commands into a file (i.e. foundation_setup.do), it can then be used to automatically setup and compile each of the libraries as demonstrated above. This script could then be used to recompile the libraries should any of them be updated. To run this script, perform the following:

1. Using the Windows Explorer, make an MTI directory in **`$XILINX\vhdl\data`**. If you have installed the Foundation Series software in the default location, this will resolve to **`C:\FNDTN\vhdl\data\MTI`**
2. From the Model*Sim* main window, chose **File** >> **Execute Macro…** and browse to the ***foundation_setup.do*** file.



Look in: temp

File name: foundation_setup.do

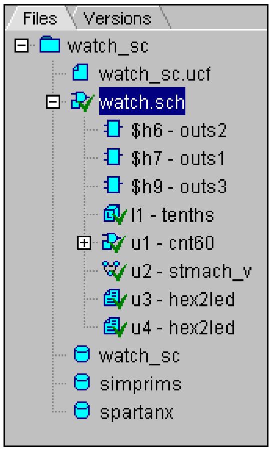
Files of type: Macro Files (*.do)

Exporting Foundation Schematics for VHDL Simulation

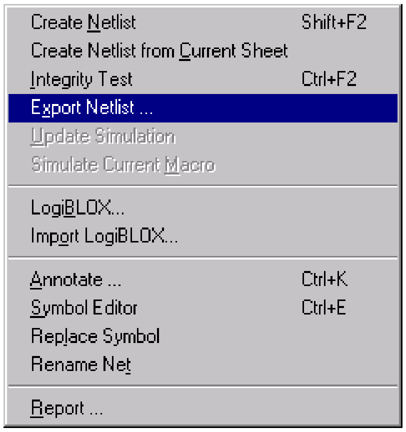
The following section describes the steps necessary for exporting and simulating a Foundation Schematic.

Exporting the Foundation Project

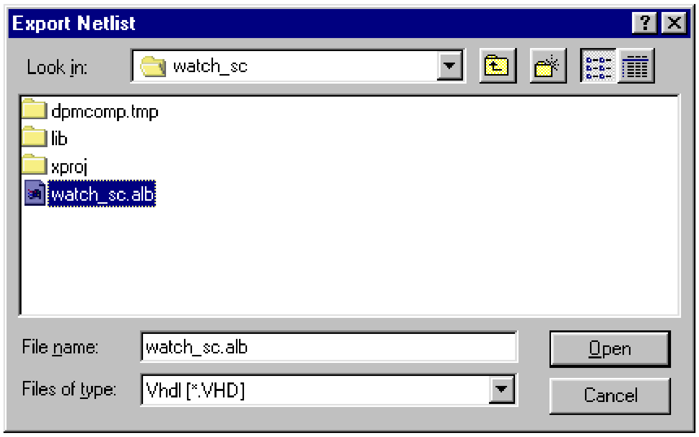
1. Start Xilinx Foundation Project Manager
2. From the Program Manger **Files** tab, double click on the top level schematic file. This will open the top level of the design in the schematic editor.



3. From the **Options** menu of the schematic editor, select **Export Netlist…**



4. In the Export Netlist dialog box, change the *Files of type* selection to VHDL
5. Browse to the schematic project file (.alb extension), and select Open



Look in: watch_sc

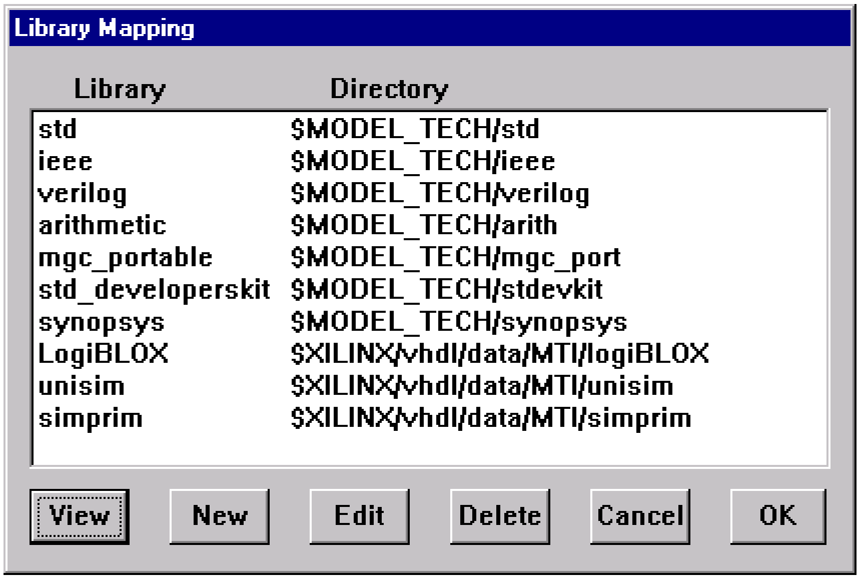
File name: watch_sc.alb

Files of type: Vhdl [*.VHD]

Each sheet of schematic will be exported as a VHDL structural netlist to the project directory. Each structural netlist will be given a name identical to that of the schematic block from which it was created.

Simulating the design

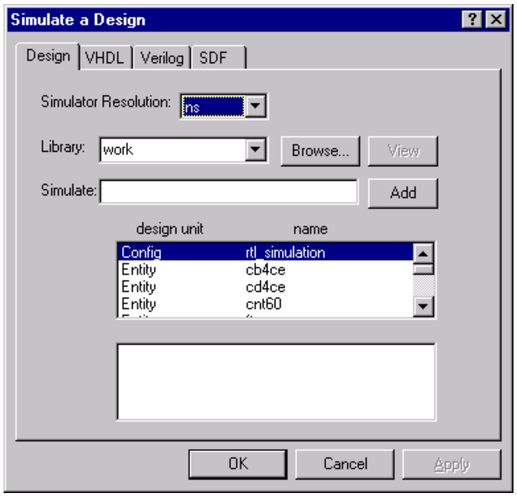
1. Open Model*Sim*
2. Use **File** >> **Directory…** to change to the project directory where the VHDL structural netslist files were exported
3. Create a new local modelsim.ini file by selecting **Project** >> **New** and entering modelsim.ini as the name of the project to create.
4. If you have correctly set up the Xilinx libraries previously, selecting **Libraries** >> **Mapping…** should reveal that the Xilinx libraries are mapped to the previously compiled libraries. This would look similar to the following:



Library Mapping

Library	Directory
std	$MODEL_TECH/std
ieee	$MODEL_TECH/ieee
verilog	$MODEL_TECH/verilog
arithmetic	$MODEL_TECH/arith
mgc_portable	$MODEL_TECH/mgc_port
std_developerskit	$MODEL_TECH/stdevkit
synopsys	$MODEL_TECH/synopsys
LogiBLOX	$XILINX/vhdl/data/MTI/logiBLOX
unisim	$XILINX/vhdl/data/MTI/unisim
simprim	$XILINX/vhdl/data/MTI/simprim

5. Create a working library using **Library** >> **New…** and typing WORK as the library name.
6. (If your design has a global signals) Compile the global signal support package ***pack.vhd*** (automatically generated by the Foundation tools) using **File** >> **Compile VHDL…** It is necessary to first compile this package into the working library to support any global signals that may have been used in the schematic.
7. Beginning from the leaf models of the design, compile each of the modules until the entire design has been compiled into library work.
8. If there are LogiBLOX modules referenced in the schematic, the VHDL simulation models may not have been copied to the Export VHDL command. Use Explorer to copy each of the simulation models to the local directory. If you did not generate VHDL simulation models for your logiBLOX modules, run LogiBLOX again to generate the simulation models for each module.
9. From the **File** >> **Simulate**… dialog window, choose the top level entity/architecture or module for simulation. If you have included a VHDL testbench, this will be the top level module.



Simulate a Design

Simulator Resolution: ns

Library: work

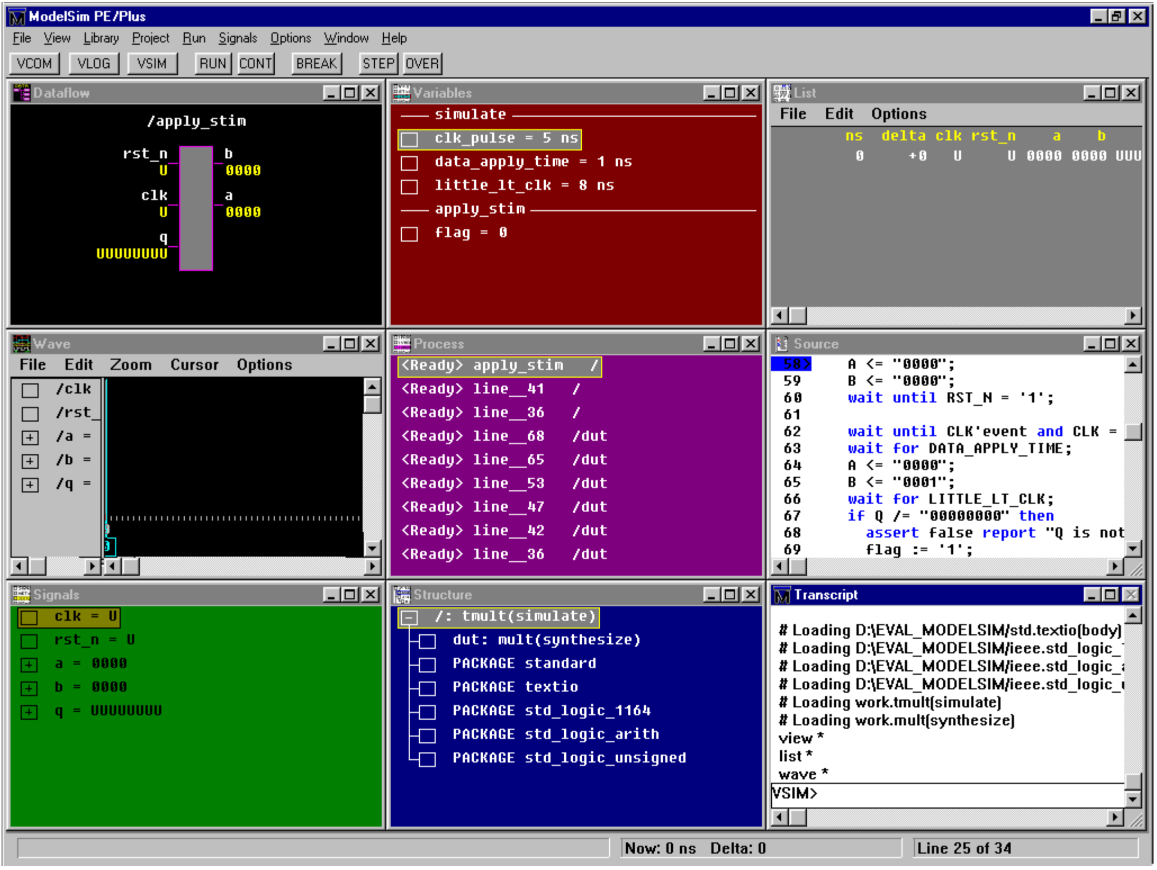
Simulate:

design unit	name
Config	rtl_simulation
Entity	cb4ce
Entity	cd4ce
Entity	cnt60

10. Set up the simulation debugging windows by first resizing the Model*Sim* window to take up most (if not all) of your screen and then selecting View->All and Window->Tile Horizontally. Find the Transcript window and type in the following commands to set up the Wave and List windows:

```
VSIM> list /*
VSIM> wave /*
```

This will add all the signals or nets at the top level of the design to the Wave and List windows.



```
—— simulate ——
□ clk_pulse = 5 ns
□ data_apply_time = 1 ns
□ little_lt_clk = 8 ns
—— apply_stim ——
□ flag = 0
```

```
ns  delta clk rst_n    a    b
 0     +0   U     U 0000 0000 UUU
```

```
<Ready> apply_stim   /
<Ready> line__41   /
<Ready> line__36   /
<Ready> line__68   /dut
<Ready> line__65   /dut
<Ready> line__53   /dut
<Ready> line__47   /dut
<Ready> line__42   /dut
<Ready> line__36   /dut
```

```
      A <= "0000";
      B <= "0000";
      wait until RST_N = '1';

      wait until CLK'event and CLK =
      wait for DATA_APPLY_TIME;
      A <= "0000";
      B <= "0001";
      wait for LITTLE_LT_CLK;
      if Q /= "00000000" then
        assert false report "Q is not
        flag := '1';
```

```
□ clk = U
□ rst_n = U
⊞ a = 0000
⊞ b = 0000
⊞ q = UUUUUUUU
```

```
⊟ /: tmult(simulate)
  □ dut: mult(synthesize)
  □ PACKAGE standard
  □ PACKAGE textio
  □ PACKAGE std_logic_1164
  □ PACKAGE std_logic_arith
  □ PACKAGE std_logic_unsigned
```

```
# Loading D:\EVAL_MODELSIM/std.textio(body)
# Loading D:\EVAL_MODELSIM/ieee.std_logic_
# Loading D:\EVAL_MODELSIM/ieee.std_logic_
# Loading D:\EVAL_MODELSIM/ieee.std_logic_
# Loading work.tmult(simulate)
# Loading work.mult(synthesize)
view *
list *
wave *
VSIM>
```

Now: 0 ns Delta: 0

Line 25 of 34

Take a look at each of the 9 windows within Model*Sim* to familiarize yourself with them.

Transcript – The command-line window; displays a transcript of all command activity.

Source – Displays the HDL source code for the design.

Structure – Displays the hierarchy of structural elements such as VHDL component instances, packages, blocks, generate statements, and Verilog model instances, names blocks, tasks and functions.

Wave – Displays waveforms and current values for the VHDL signals, and Verilog nets and register variables you have selected.

List – Shows the simulation values of selected VHDL signals, and Verilog nets and register variables in tabular format.

Dataflow – Allows you to trace VHDL signals or Verilog nets through your design.

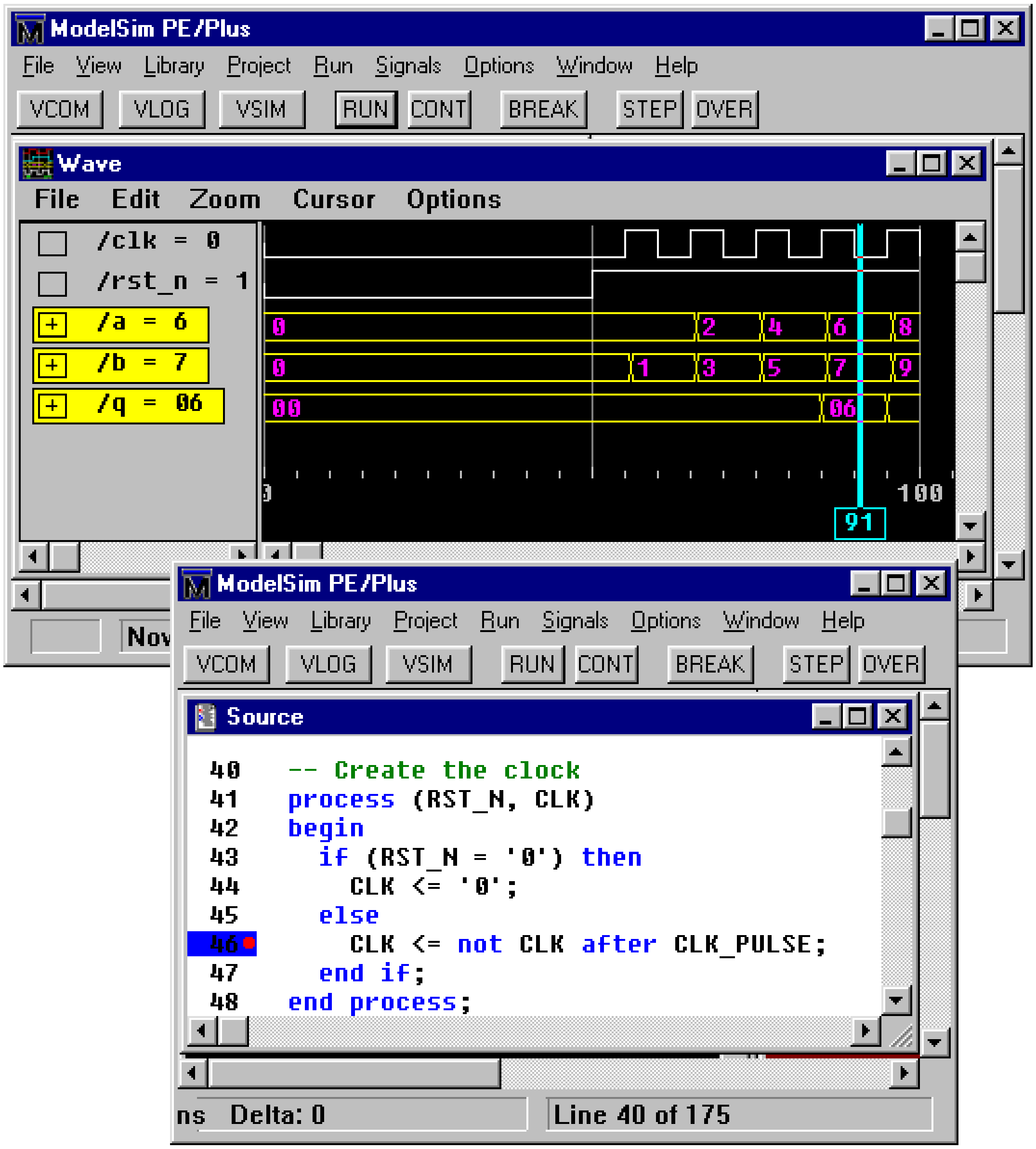
Signals – Shows the names and current values of VHDL signals, and Verilog nets and register variables in the region currently selected in the Structure window.

Variables – Displays VHDL constants, generics, variables, and Verilog register variables in the current process and their current values.

Process – Displays a list of processes that are scheduled to run during the current simulation cycle.

Many of these windows are dynamically linked with each other. If you click on a signal or process in one window, the other windows will update to display or highlight that object.

11. Run the simulation. Press the “RUN” button or type “run 100” at the VSIM> prompt. Below is an example of setting a cursor in the Wave window and of hitting a breakpoint in the Source window.


ModelSim PE/Plus
File View Library Project Run Signals Options Window Help
VCOM VLOG VSIM RUN CONT BREAK STEP OVER
Wave
File Edit Zoom Cursor Options
0
2
4
6
8
0
1
3
5
7
9
00
06
0
100
91
Nov
ModelSim PE/Plus
File View Library Project Run Signals Options Window Help
VCOM VLOG VSIM RUN CONT BREAK STEP OVER
Source
ns Delta: 0
Line 40 of 175

/clk = 0
/rst_n = 1
/a = 6
/b = 7
/q = 06

```
-- Create the clock
process (RST_N, CLK)
begin
  if (RST_N = '0') then
    CLK <= '0';
  else
    CLK <= not CLK after CLK_PULSE;
  end if;
end process;
```

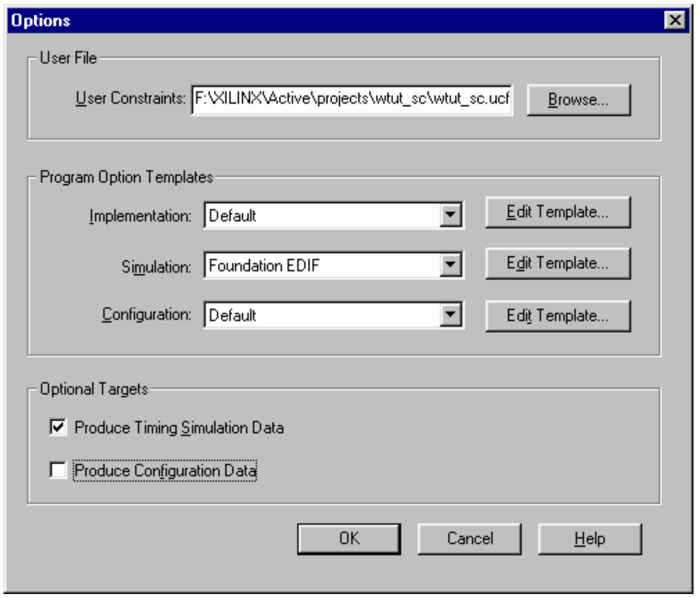
12. Quit ModelSim. Select “File->Exit ModelSim” from the Model*Sim* menus.

Simulation of Post Place and Route Designs

The Xilinx implementation tools will produce a timing simulation netlist after the design has been placed and routed. The Model*Sim* simulator HDL can perform a fully back annotated timing simulation using this netlist and SDF timing information.

Creating a back annotated timing netlist

1. Select **Implement** >> **Implementation Options** from the Foundation Project Manager
2. Ensure that the *Produce Timing Simulation Data* option is selected in the Options dialog, and that the *Program Option Template* has the *Simulation* template set to Modelsim VHDL or Modelsim Verilog (depending on the desired HDL language)



User Constraints: F:\XILINX\Active\projects\wtut_sc\wtut_sc.ucf

Implementation: Default

Simulation: Foundation EDIF

Configuration: Default

☑ Produce Timing Simulation Data

☐ Produce Configuration Data

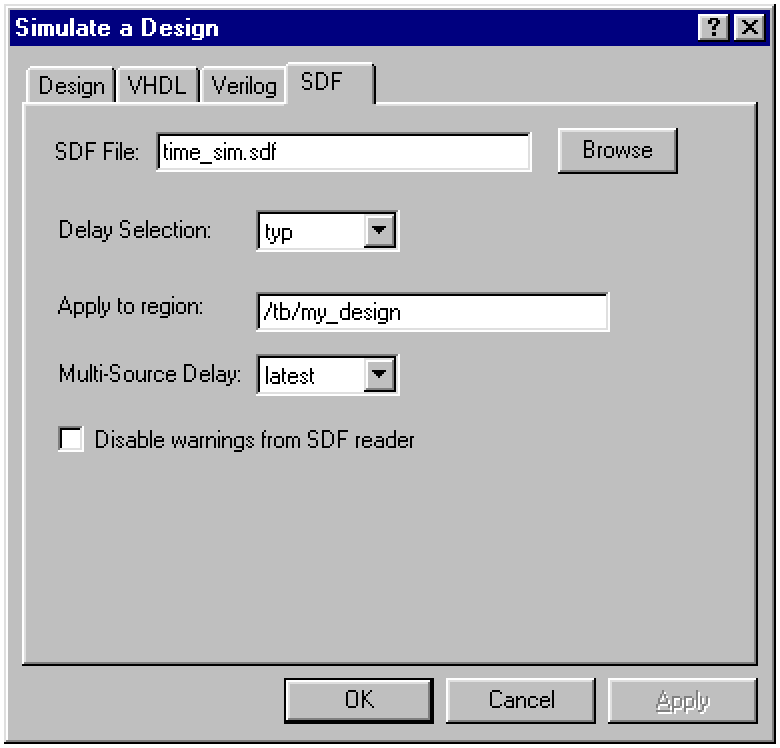
3. Click OK and complete the implementation of the design.

Simulating the full timing netlist

1. From the Model*Sim* window, browse to the specific design revision in the project directory
2. Using **File** >> **Compile VHDL …** (or Verilog…) compile the timing simulation netlist produced by the Xilinx implementation tools. This file will be named either *time_sim.vhd* or *time_sim.v*

   Note: If you have symbols which require configuration before simulation, such as OSC, ROC, or TOC, configure those before compiling the timing simulation netlist.

3. Compile the HDL test bench. There is a template test bench file created by the Xilinx implementation tools. The file will be named either *time_sim.tvhd* or *time_sim.tv*
4. From the **File** >> **Simulate** dialog, choose the top level of the design (including any test bench you have created).
5. From the **Simulate a Design** dialog, under the **SDF** tab, include the SDF back annotation timing file created by the Xilinx implementation tools. This file will be named *time_sim.sdf*. In the **Apply to region** field, enter the hierarchical path to the region that the time_sim.sdf file applies to. For example, for a testbench called "tb" that instantiates a design called "my_design" that the time_sim.sdf was created for, the region to apply the SDF file to would be "/tb/my_design". Press "OK".



SDF File: time_sim.sdf

Delay Selection: typ

Apply to region: /tb/my_design

Multi-Source Delay: latest

□ Disable warnings from SDF reader

6. After the simulation is loaded, you can use the ModelSim debugging windows described above to simulate and debug the full timing netlist.