ModelSim VHDL Simulation Tutorial

Watch Design

UG102 (v1.1) July 21, 2000

XILINX®

ModelSim VHDL Simulation Tutorial

UG102 (v1.1) July 21, 2000

The following table shows the revision history for this document.

Date	Version	Revision
06/01/00	1.0	Initial Xilinx release.
07/21/00	1.1	Accumulated miscellaneous updates and bug fixes.

Contents

ModelSim VHDL Simulation Tutorial

ModelSim VHDL Simulation Tutorial

This tutorial shows you how the VHDL simulation flow works for Xilinx FPGA designs using MTI's Modelsim simulator. The design, watch, targets a Virtex device and implements the functionality of a typical runner's stopwatch. This tutorial contains the following sections.

- "Design Description"
- "Before Beginning the Tutorial"
- "Tutorial Installation"
- "Including CoreGen Components"
- "Creating the Tenths CORE Generator Component"
- "Synthesizing/Implementing the design"
- "Timing Simulation"

Design Description

Throughout this tutorial, the design is referred to as Watch which is a design for a runner's stop watch. The tutorial assumes that you have a working knowledge of VHDL.

The Watch design is a counter that counts up from 0 to 59, then resets to zero, and starts over. There are three external inputs and three external outputs in the completed design.

The Watch design inputs, outputs, and macros are summarized below.

Inputs

- **STRTSTOP**—The start/stop button of the stopwatch. This is an active-low signal that must be depressed then released to start or stop the counting.
- **RESET**—Forces the signals TENSOUT and ONESOUT to be "00" after it has been stopped.
- **CLOCK**—System clock provided externally by the designer.

Outputs

- **TENSOUT[6:0]**—7-bit bus which represents the tens-digit of the stopwatch value. This is viewable on the 7-segment LED display of the Xilinx demo board.
- **ONESOUT[6:0]**—Similar to TENSOUT bus above, but represents the one-digit of the stopwatch value.
- **TENTHSOUT[9:0]**—10-bit bus which represents the tenths-digit of the stopwatch value. This bus is one-hot encoded. The output is displayed to the LED bar.

Macros

The top level of the Watch design consists of the following functional blocks.

- **STATMACH**—A statemachine that controls starting, stopping, and clearing the counters (One-hot encoded).

- **TENTHS**—A Coregen 10-bit one-hot counter macro which outputs the Tenths digit as 10-bit one-hot value.
- **CNT60**—A 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.
- **DECODE**—Decodes binary values to one-hot.

Before Beginning the Tutorial

Before you begin this tutorial, set up your system to use the Model Technology and Xilinx software as follows.

1. Install the following software.
   - Xilinx Development System 3.1i
   - Model Technology ModelSim EE/PE 5.3 or later
2. Verify that your system is properly configured. Consult the release notes and installation notes that came with your software package for more information.

Tutorial Installation

The Watch tutorial file is available for download from the Xilinx Web site at **http://www.xilinx.com/support/techsup/tutorials**.

Tutorial Directory and Files

The tutorial directory and tutorial files needed to complete the design are provided for you. Some files are not present since you will create them in later steps. The following table lists the contents of the tutorial directories.

Directory	Description
mtivhdl_tut/src	VHDL source and script files
mtivhdl_tut/soln	VHDL solutions directory
mtivhdl_tut/watch	VHDL Tutorial Directory

VHDL Design Files

Watch is the top level design. The tutorial uses the following VHDL files.

- stopwatch.vhd
- statmach.vhd
- smallcntr.vhd
- cnt60.vhd
- hex2led.vhd
- decode.vhd
- testbench.vhd (VHDL testbench for simulation)

Note: The Tenths one-hot counter is a Coregen macro. The tenths.vho file is indirectly used in RTL simulation. To get familiarized with how the Coregen RTL simulation flow works, please consult the Coregen User Guide at **http://support.xilinx.com/support/techsup/journals/coregen/2.1i/docs.htm**

Script Files

The following script files are provided to automate the steps in this tutorial.

- rtl_sim.do
- stim.do
- time_sim.do

Simulation Models for MTI

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

- **UNISIMS Library**—The Unisim library is used for behavioral (RTL) simulation with instantiated components in the netlist, and for post-synthesis simulation. The library is VITAL compliant, and it also adds the device start-up components ROC, ROCBUF, TOC, TOCBUF, and STARTBUF, for simulation.
- **LogiBLOX Library**—The LogiBLOX library is used for designs containing LogiBLOX components, during pre-synthesis (RTL), and post-synthesis simulation. Since LogiBLOX models are not supported in Virtex, this library will not be used in this tutorial.
- **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.
- **COREGEN Library** —CORE Generator is a graphical interactive design tool you use to create high-level modules such as counters, shift registers, RAM and multiplexers. The CORE Generator HDL library models are used for RTL simulation, and the models do not use library components for global signals.The CHDL CORE Generator source files are found in $XILINX/vhdl/src/XilinxCoreLib.

For detailed instructions on compiling these libraries, see Xilinx Solution # 2561 which is available on the Internet at **http://www.xilinx.com/techdocs/2561.htm**.

After compiling the libraries, notice that ModelSim creates a file called modelsim.ini. View this file and notice that the upper portion defines the locations of the compiled libraries. When doing a simulation, the modelsim.ini file must be provided either by copying the file directly to the directory where the HDL files are to be compiled and the simulation is to be run, or by setting the MODELSIM environment variable to the location of your master .ini file. You must set this variable since the ModelSim installation does not initially declare the path for you. For UNIX, type the following.

```
setenv MODELSIM /path/to/the/modelsim.ini
```

Including CoreGen Components

Creating the Tenths CORE Generator Component

Since the Watch design contains a CORE Generator macro, you must create it before performing RTL simulation or implementation. When creating the CORE Generator component, you will create a behavioral simulation netlist for RTL simulation, as well as the implementation netlist.

In this section, you create a CORE Generator module called Tenths. Tenths is a 4-bit binary encoded counter. The 4-bit number is decoded to count the tenths digit of the stopwatch's time value.

Creating the Core Generator module

You select the type of module you want in the CORE Generator dialog box as well as the specific features of the module.

To invoke the CORE Generator GUI, type coregen at the UNIX prompt, or if you are using a PC, click on the CORE Generator icon in the Xilinx Program group.

From Project -> Project Options link, select “VHDL” and your synthesis vendor of choice for your design entry.

In the Target Architecture field, choose the Virtex family.

CORE Generator Setup Dialog

Click OK to close the dialog box.

A list of possible CORE s are available. Double Click on Basic Elements - Counters.

Double Click on Binary Counter to open the Binary Counter dialog. This dialog allows the user to customize the counter to the design specifications.

Fill in the Binary Counter dialog with the following settings.

Component Name: tenths

Defines the name of the module.

Output Width: 4

Defines the width of the output bus.

Operation: Up

Defines how the counter will operate. This field is dependent on the type of module you select.

Count Restrictions: Restrict Count to A.

This dictates the maximum count value.

Output Options: Threshold0 set to A

Signal goes high when the value specified has been reached.

Output Options: Registered

CORE Generator Module Selection

Click on the Register Options button to open the Register Options dialog. Enter the following settings.

Clock Enable: Selected

Asynchronous Settings: Init with a value of 1.

Synchronous Settings: None

Click Ok.

Check that only the following pins are used.

AINIT

CE

Q

Q_Thresh0

CLK

Click Generate. The module is created.

Select Cancel and close Core Generator.

CORE Generator generates the following output Files.

tenths.edn

This file is the netlist that is used during the Translate phase of implementation.

tenths.vho

This is the instantiation template that is used to incorporate the CORE Generator module in your source VHDL.

tenths.xco

This file stores the configuration information for the Tenths module.

coregen.prj

This file stores the CORE Generator configuration for the project.

Instantiating the CORE Generator Module

Open stopwatch.vhd in a text editor.

At the line that states:

"-- Insert Coregen Counter Component Declaration"

Open the tenths.vho file, and cut-and-paste the component declaration to stopwatch.vhd.

The template component declaration for the Coregen instantiation is inserted.

At the line that states:

"--Place the CoreGen Counter Instantiation"

Open the tenths.vho file, cut-and-paste from the line "your_instance_name : tenths" to "AINIT => AINIT);" to stopwatch.vhd.

Change "your_instance_name" to "XCOUNTER".

Edit this code to connect the signals in the Stopwatch design to the ports of the CoreGen module. The completed code is shown below.

XCOUNTER : tenths port map (Q => Q, CLK => CLK, Q_THRESH0 => xtermcnt, CE => clkenable, AINIT => rstint);

From the tenths.vho file, also cut and paste the section beginning from line "for all : tenths use entity" to "end for;" and paste it to the bottom of the stopwatch.vhd file, after the "end inside;" line.

Save stopwatch.vhd.

Using the configuration declaration

Configurations are a primary design unit used to bind component instances to entities. For component instances, the configuration specifies from many architectures for an entity which architecture to use for a specific instance. When the configuration for an entity-architecture combination is compiled into the library, a simulatable object is created.

Xilinx Coregen VHDL simulation methodology uses the configuration statement to bind the Coregen model to the top level design unit. The configuration declaration is provided below:

```
library XilinxCoreLib;

CONFIGURATION stopwatch_cfg OF testbench IS

FOR testbench_arch

FOR ALL : stopwatch use configuration work.cfg_tenths;

END FOR;

END FOR;

END stopwatch_cfg;
```

Copy and paste this code to the bottom of the testbench.vhd file, below the "end testbench_arch;" line. Save testbench.vhd and exit the editor.

RTL Simulation

In RTL simulation, the testbench.vhd file instantiates the top level file, stopwatch.vhd. The testbench also passes stimulus for the system clock and other inputs. Additionally, the testbench.vhd file contains a configuration statement. The configuration statement links the Coregen tenths component to the testbench. Without this configuration statement, the Coregen component will not simulate. To see further details on why the configuration statement is needed for Coregen components, please refer to the Coregen user guide at **http://support.xilinx.com/support/techsup/journals/coregen/2.1i/docs.htm**.

Copying Source Files to the Functional Simulation Directory

Copy the following files from the /mtivhdl_tut/src directory into the **/mtivhdl_tut/watch/func** directory.

- smallcntr.vhd
- cnt60.vhd
- hex2led.vhd
- stopwatch.vhd
- statmach.vhd
- testbench.vhd
- decode.vhd
- rtl_sim.do

Starting ModelSim

If you are using the PC, invoke the simulator by selecting **Programs** → **Model Tech** → **ModelSim** from the Start menu. For UNIX workstations, type the following at the prompt.

```
vsim -i &
```

Set the project directory using the **File** → **Change Directory** menu command and select **watch/func**.

Creating the Work Directory

Before compiling the VHDL/Verilog source files, you must create a directory for use as a library. Type the following at the ModelSim prompt.

```
vlib work
```

This action is echoed in the Transcript window as shown in the following figure.

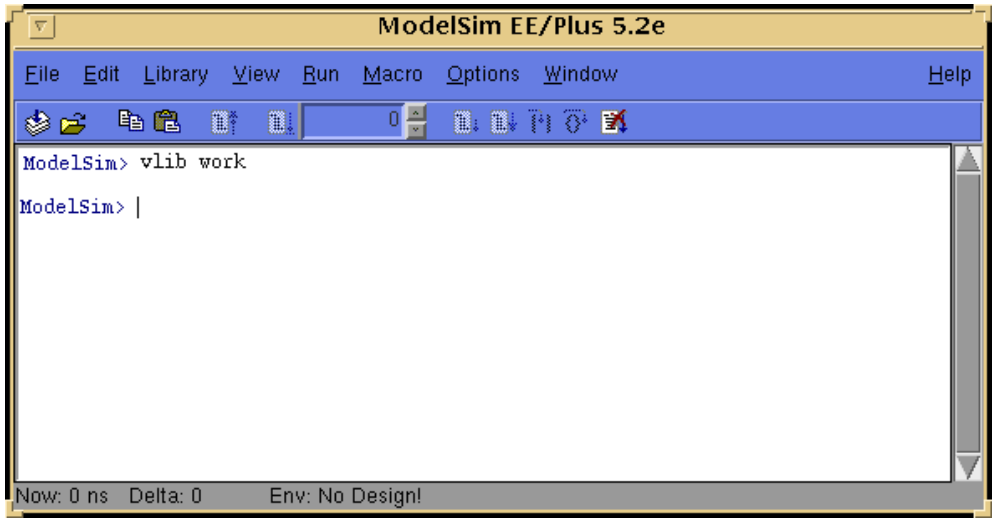

Figure 1: **MTI Transcript Window**

Compiling the Source Files

Since Xilinx Unified library components are instantiated within the VHDL source code, the UNISIM simulation models must be provided. The following lines must be added in the files watch.vhd and .

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

As a key point, the unisim library is for simulation only, and so these lines should be commented out for synthesis.

The Vcom command compiles VHDL code for use with Vsim RTL simulation. Also, to enhance simulation, ModelSim supports VHDL '93. The -93 switch is used to enable support for 1076-93. Type the following at the ModelSim prompt.

```
vcom -93 -explicit smallcntr.vhd
vcom -93 cnt60.vhd
vcom -93 decode.vhd hex2led.vhd statmach.vhd
vcom -93 stopwatch.vhd testbench.vhd
```

The -explicit is used to compile smallcntr.vhd since there is a definition of "=" in the std_logic_1164 and std_logic_unsigned libraries that are declared for the entity. The option resolves resolution conflicts in favor of explicit function.

Invoke the Simulator

Type the following at the ModelSim prompt to invoke the ModelSim simulator.

```
vsim stopwatch_cfg
```

Note: The file, rtl_sim_xilinx.do, runs the above commands; you can run it instead of executing each command. The file is located in the src directory and you can copy it into the watch/func directory. To execute the file, type the following at the ModelSim prompt.

```
do rtl_sim.do
```

Optionally, you may launch the macro via the **Macro** → **Execute Macro** menu command.

Running the Simulation

To perform simulation using ModelSim, follow these steps.

1. To view all the ModelSim debug windows, type the following.
   ```
   view *
   ```
2. Add the signals from the selected region in the Structure window to the Wave and List windows by issuing the following commands at the ModelSim prompt.
   ```
   add wave *
   add list *
   ```
3. In the Structure window, notice that VHDL design units are indicated by squares. You can expand and collapse regions of hierarchy by clicking on the (+) and (-) notations.
4. The signals window lists the signals in the VHDL design unit currently
5. To run the simulation for a specified amount of time at the ModelSim prompt, type the following.
   ```
   run 20000 ns
   ```
   The simulation output is displayed in the Wave window. You may have to zoom in / out to view the waveforms.
6. In the Wave window, try adding or removing cursors with the **Cursor → Add | Remove** menu command. When multiple cursors are drawn, ModelSim adds a delta measurement showing the time difference between the cursors. The selected cursor is drawn as a solid line and the values at the cursor location are shown to the right of the signal name. All other cursors are drawn as dotted lines. If you cannot see the signal value next to the signal name, select the bar separating the signal names from the waveforms and drag it to the right.

   Note: The above commands have been combined into a macro file called stim.do. You can execute them at the ModelSim prompt.

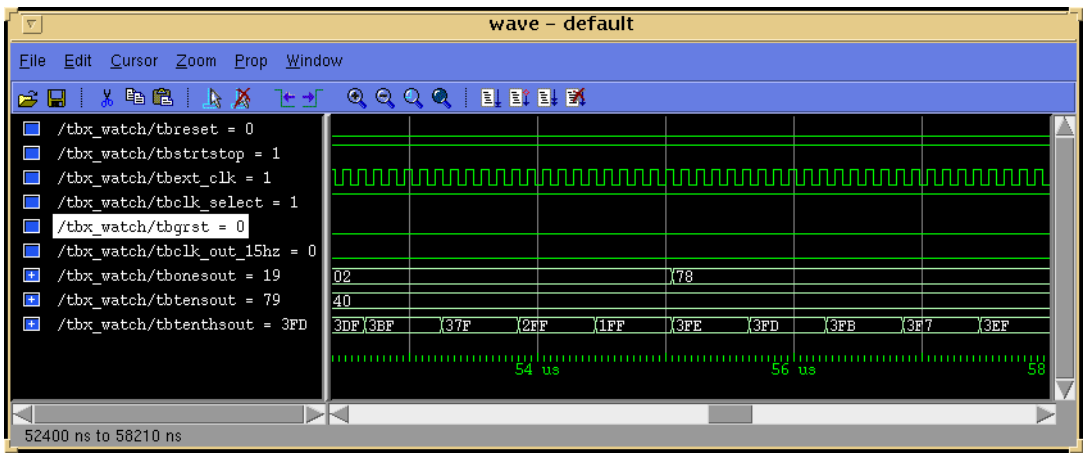

Figure 2: **Simulation Output in Wave Window**

Synthesizing/Implementing the Watch Design

This tutorial only covers VHDL RTL and Back-annotated simulation. Please go to **http://support.xilinx.com/support/techsup/tutorials/31i_tutorials_ph.htm** and select tutorials for synthesis and implementation to obtain the VHDL files needed for back-annotated simulation.

While Implementing the design, make sure that the “MODELSIM VHDL” option is selected for “Simulation Options” so that a back-annotated VHDL file compatible with Modelsim is created by Xilinx.

Timing Simulation

For VHDL tutorial, you need two files from the Xilinx core tools.

- time_sim.vhd
- time_sim.sdf

These files are created by the Xilinx Implementation tools, and are copied into the implementation directory where all the place and route files are stored.

Now that the HDL netlist has been resolved into primitives, you need to modify the testbench configuration. The Unisim library was referenced since the pre-synthesis netlist contained instantiated Xilinx macros.

To perform timing simulation, follow these steps.

1. Copy time_sim.vhd, time_sim.sdf, and testbench.vhd to the following directory.
   ```
   /mtivhdl_tut/watch/time
   ```
2. Launch ModelSim, and navigate to the following directory.
   ```
   /mtivhdl_tut/watch/time
   ```
3. Create the work directory.
   ```
   vlib work
   ```
4. View the testbench.vhd file and notice that there are two sections at the bottom.

   The configuration statement is for RTL simulation. Comment out the configuration statement by inserting "--" at the start of all the lines after the "end testbench_arch;" line.
5. After editing the testbench.vhd, save the changes and exit.
6. Compile the VHDL source files and the testbench.
   ```
   vcom time_sim.vhd testbench.vhd
   ```
7. Read in the SDF file for timing simulation.
   ```
   vsim -sdftyp uut=time_sim.sdf work.testbench
   ```

   Alternatively, select **File** → **Load New Design**. Highlight "testbench" in the Design Unit window. Click the Add button. To apply the timing data, click on the SDF tab on the Load Design window. Click the Add button. Browse and select for the time_sim.sdf file. Type **uut** in the Apply to Region field and click the Load button.
8. View the necessary debugging windows by typing the following command at the ModelSim prompt.
   ```
   view wave signals source
   ```
9. View and add the signals of the design to the waveform window.
10. At the ModelSim prompt type.
    ```
    run 20000 ns
    ```
11. Right click in the waveform window and zoom in. Another way to zoom in, press and hold the middle mouse button and draw a square around the area to zoom in on. After simulating, you can then zoom in and view the delay from the clock edge to the TENSOUT, ONESOUT, and TENTHSOUT output change.

    Note: The above commands have been combined into a macro file, time_sim.do, and can be executed at the ModelSim prompt.

The MTI-VHDL Tutorial is now complete