# Importing a *Warp*™ Post-fit Netlist Into Mentor Graphics' *ModelSim*™

## Introduction

This application note is intended to assist *Warp*™ users in importing and simulating post-fit models into Mentor Graphics's *ModelSim*™ product. This note contains detailed steps to create and compile the design in *Warp*. It also contains steps to compile and load the resulting post-fit model in *ModelSim*.

## Creating Post-fit Models in *Warp*

The following describes how to create a post-fit model in *Warp*. The model is the output simulation model, resulting from synthesizing and fitting the design in *Warp.*

1. Create a project in *Warp* with all the files required to compile the design.
2. Go to the "Project -> Compiler Options..." window as displayed in *Figure 1* below.

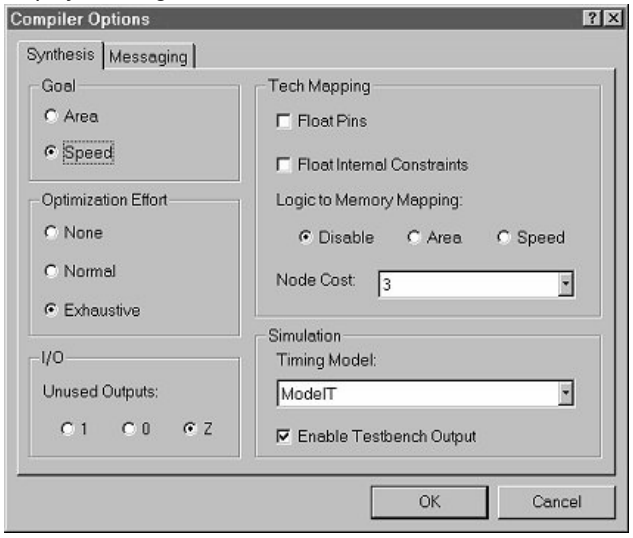


**Figure 1. Setting Testbench Output for *ModelSim***

For the above figure, set the Timing Model box to '**ModelT**', and select '**Enable Testbench Output**'.

3. Compile the project

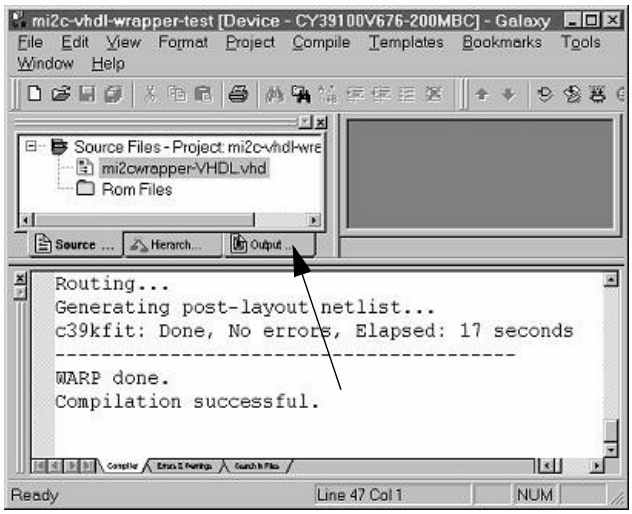


**Figure 2. Compiling the Project to a Post-fit Model**

When the project has compiled and fitting is complete, the console window on the bottom of the screen will show that compilation has been successful. As a result, a structural model will be created with timing information which can be imported into *ModelSim*.

Select the 3rd tab in the left hand window, shown by the arrow in Figure 2. This displays the output files from the compile operation. Click on the '**VHD**' folder icon to expand it. In the folder is the post-fit model created by *Warp*.

4. View the report file by selecting the 3rd tab in the left hand window and expanding the '**r**' icon in this window. This allows you to view device utilization, timing, pinout assignments, etc.

## Importing the Post-Fit Model into a *ModelSim* Testbench

After compiling the design in *Warp*, the top-level component of the design must be defined and instantiated in the testbench for the design. This is accomplished by defining the top-level of the design as a component, just after the 'architecture definition' section of the testbench. An example of this follows:

*Warp*, *Ultra37K, and Delta39K* are trademarks of Cypress Semiconductor Corporation.

```
architecture arch_mi2ctb of mi2ctb is
-- architecture section begins here
-- component declaration of the design top-level
component mi2c
  port (
   CLK, NRST: in std_logic;
   A: in std_logic_vector (2 downto 0);
   DI: in std_logic_vector (7 downto 0);
   WR, SEL, ISCL, ISDA: in std_logic;
   DA: inout std_logic_vector (7 downto 0);
   NOE, INTR, OSCL, OSDA: inout std_logic
 );
end component;
```

Before the 'begin' in the architecture section, define signals to map the top-level variables to the IO signals of the imported model. In the architecture body, where the user has instantiated the Unit Under Test (UUT), map the IOs of the UUT to the top-level signals. An example of this follows:

```
-- architecture section begins here
begin
-- the following is defined after the begin reserved word.
UUT: mi2c port map (
    CLK=> TOP_CLK,
    NRST=>TOP_NRST,
    A=> TOP_A,
    DI=> TOP_DI,
    WR=>TOP_WR,
    SEL=> TOP_SEL,
    ISCL=> TOP_ISCL,
    ISDA=> TOP_ISDA,
    DA=> TOP_DA,
    NOE=> TOP_NOE,
    INTR=> TOP_INTR,
    OSCL=> TOP_OSCL,
    OSDA=> TOP_OSDA
);
end arch_mi2ctb;
-- end of architecture section
```

Where top_clk, top_nrst, top_a and all the signals on the right hand side of the '=>' sign are the signals defined in the top level of the testbench.

## Preparing *ModelSim* for Post-Fit Simulation

Create a library in *ModelSim* into which the post-fit model will be compiled. Before the post-fit model can be used, *Warp* primitives and libraries must be compiled into *ModelSim*. To compile these into *ModelSim*, copy the script in Appendix B into the *ModelSim* console and press return until all lines are executed.

The "$env(CYPRESS_DIR)" is an environment variable set automatically upon the installation of *Warp*. In Win95 or 98, the value of the variable CYPRESS_DIR is defined in the autoexec.bat. In WinNT, the definition is located in ControlPanel -> System->Environment. In Unix, the variable is an environment variable.

## Compile the *Warp* post-fit model in *ModelSim*

1. Copy the post-fit model file from the '***Warp* project / vhd**' directory into the project directory in *ModelSim*. Compile this file into the library you've created in *ModelSim*.

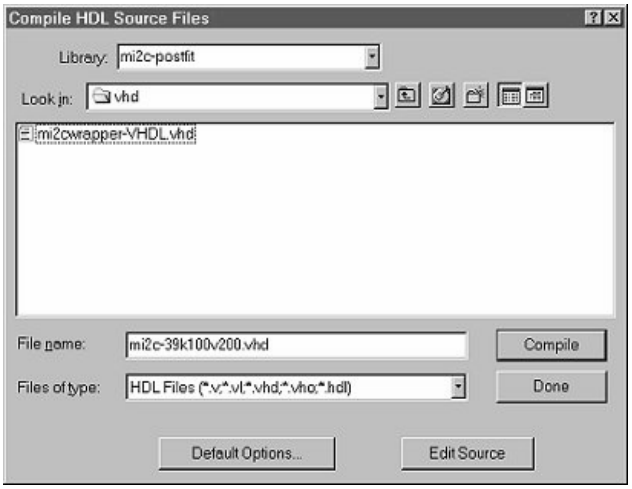


**Figure 3. Compiling the Post-Fit Model Located in the vhd File.**

In the above, the library created is called 'mi2c-postfit' and the post-fit model located in the vhd file folder is 'mi2cwrapper-VHDL.vhd'.

2. Sometimes, when the user compiles the top-level file the following error may occur in ModelSim:

*"WARNING[1]: D:/CCFolder/MI2C/testbench-source/mi2c.vhd(440): Incompatible modes for port DA"*

If this occurs, change all the OUTS to INOUTS in the user's top-level component declaration located in the testbench.

3. Compile the testbench which instantiates the post-fit model in the same library as Step 1.

## Loading the testbench into *ModelSim*

1. Load the testbench by going to "Design->Load New Design".
2. Select the library, which contains the compiled post-fit netlist. In Figure 4, the library is called 'mi2c-postfit' as mentioned in the previous section.
3. Select the design unit of the top level of the testbench to load. In Figure 4, it is 'mi2c_tb'. The design is now ready to be simulated.

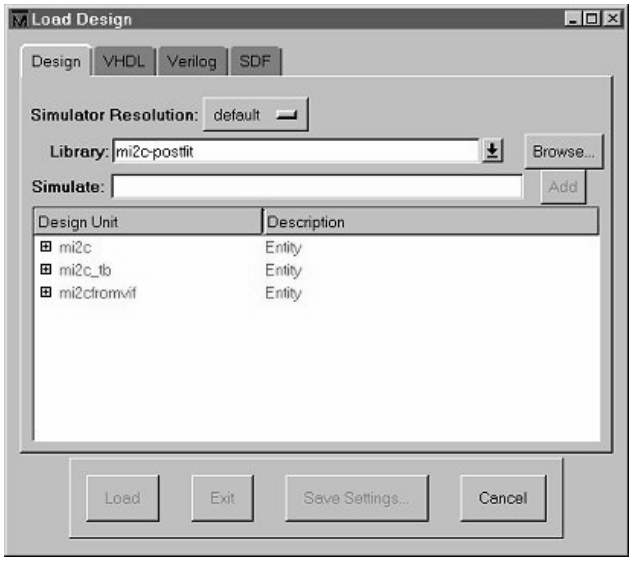


**Figure 4. Loading a Testbench in *ModelSim.***

## Conclusion

The steps required to use *ModelSim* for post-fit simulation of a design created with *Warp* software have been described.

## Appendix A: Loading and Simulating a *ModelSim* Compiled Library

When a user purchases a core from Inventra, the core package includes a compiled *ModelSim* library. This library contains a compiled testbench and simulation model of the core, fit to 39K100-200. The user can open this in *ModelSim* and run it to see how the core behaves when fit to Delta39K. The following steps describe how the user can use the *ModelSim* compiled library in *ModelSim*:

1. Unzip the zip file containing the *ModelSim* library folder.
2. Create a new folder in the *ModelSim* working directory and place this library folder in this new folder.
3. Open *ModelSim* and change directory to select the working directory from Step 2.
4. Compile *Warp* primitives and libraries into this directory by following the instructions in the section "Preparing ModelSim for Post-fit Simulation".
5. Click on the '**Load**' design button as shown below. Select the new *ModelSim* library along with the top-level testbench file of this library and load it.

 This is the '**Load**' design button in *ModelSim*.

6. Click on the "Run -> Run -All" to run the entire design.

# Appendix - B: Script Lines to Compile Primitives and Libraries into *ModelSim*™

```
vlib cypress
vmap cypress cypress
vcom -93 -quiet -work cypress $env(CYPRESS_DIR)/lib/prim/presynth/std/cypress.vhd
vcom -93 -quiet -work cypress $env(CYPRESS_DIR)/lib/prim/presynth/std/rtlpkg.vhd
vcom -93 -quiet -work cypress $env(CYPRESS_DIR)/lib/prim/presynth/std/lpmpkg.vhd
vcom -93 -nowarn 5 -quiet -work cypress $env(CYPRESS_DIR)/lib/prim/presynth/std/sim.vhd
vcom -93 -quiet -work cypress $env(CYPRESS_DIR)/lib/prim/presynth/std/ttfctn.vhd
vcom -93 -nowarn 3 -quiet -work cypress $env(CYPRESS_DIR)/lib/prim/presynth/std/std_arth.vhd

vlib cyprim
vmap primitive cyprim
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c37kp.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c37kinp.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c37km.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c37kmux.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c37koreg.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c39kp.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c39kcf.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c39kck.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c39kcm.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c39kcr.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c39kio.vhd
vcom -93 -quiet -work primitive $env(CYPRESS_DIR)/lib/prim/vhdl/c39kmc.vhd
```

© Cypress Semiconductor Corporation, 2000. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress Semiconductor product. Nor does it convey or imply any license under patent or other rights. Cypress Semiconductor does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress Semiconductor products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress Semiconductor against all charges.