Model Technology
A MENTOR GRAPHICS COMPANY

Using Model Technology Model*Sim* PE version 5.2 with Altera Max Plus II Software

Model Technology Inc. Modelsim is a single kernal, dual language simulator. You are able to run either Verilog or VHDL separately or mixed in the same design. You can have a Verilog module instanciate VHDL entities or VHDL instanciate Verilog! One simulator, one interface, two languages. Altera Max Plus II can output both Verilog or VHDL, there are no restrictions on which language you choose. Model*Sim* simulator uses compiled HDL libraries. You must first compile all needed files into libraries for the top level netlist. The Altera ModelSim flow for each language is slightly different and is described below. Each language flow will be described with command line and User Interface examples. Links to the source files are listed below.

Methods of executing ModelSim

ModelSim can be used in batch mode, interactive command line, or with the User Interface (UI). Batch mode is the typical method when running regression tests. Interactive Command line mode is very similar to batch mode in the fact that the UI is not displayed, the only interface is a command line console. The UI can accept both command line, and menu input.

ModelSim Altera RTL Flow

The following commands can be used in any mode. Note that the view * (view all windows) and the add wave /* (add all signals at top level to wave form window) will show results only when in UI mode. By saving these commands in a file ("altera_rtl.do" is a common macro file naming convention) the file can be used in UI mode, command, or batch modes.

```
cd <design directory>
vlib work
vcom counter.vhd
vsim counter
view *
add wave /*
add list  /*
do run.do
```

In the above example the synthesizable RTL counter.vhd is VHDL, but could just as easly been Verilog. If the counter was a verilog file simply use "vlog counter.v" instead of "vcom counter.vhd".

RTL to Gate Level Flow

The RTL netlist will be input to your synthesis tools. The output of the synthesis tool will most likely be an EDIF netlist which will then be the source file for the Altera Max Plus II software. The Altera software can output a Verilog and/or VHDL netlist. The use of each language is different when using Altera Max Plus II software. In the case of VHDL the VITAL source files are located in the Altera release tree. The Verilog library is created by the Altera Max+Plus II software when and where the gate level netlist is output.

The following sections describe how to set up the Altera VHDL libraries and Gate Level source files for use in ModelSim simulations. The first section describes the command line sequence to setup and run an automated script to accomplish the task. Followed by the step-by-step instructions for compiling the Altera VHDL simulation model libraries (VITAL) and Gate Level source file(s) with the User Interface.

VITAL and VHDL Source Compile and Simulation With Command Line

Input files to the ModelSim VHDL Gate Level Flow that are from Altera place and route tools(these are default file extensions for VHDL output generation):

counter.vho	(VHDL Gate level file based on the original RTL)
counter.sdo	(SDF file with timing information)

Input files for ModelSim from Altera installation directory (Altera VITAL library):

alt_vtl.cmp
alt_vtl.vhd

The following ModelSim PE command sequence will build the Altera Max PlusII VITAL library and simulate the counter gate level design with SDF back-annotation timing. These commands will be equivalent to the UI commands in the following VITAL and VHDL sections. They can be entered from the ModelSim main **Transcript** window

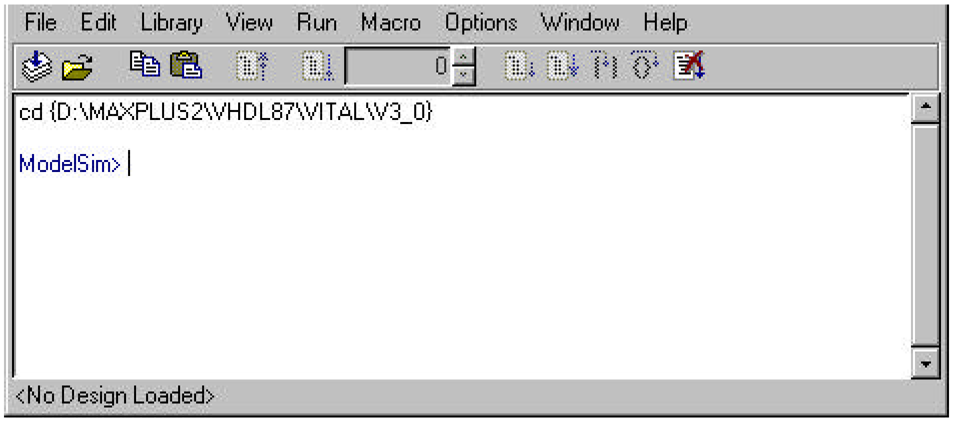
```
cd maxplus2\vhdl87\vital\v3 _0
vlib alt_vtl
vcom -work alt_vtl -87 alt_vtl.vhd
vcom -work alt_vtl -87 alt_vtl.cmp
cd your_design_directory
vmap c:/maxplus2/vhdl87/vital/v3_0/alt_vtl alt_vtl
vlib work
vcom  counter.vho
vsim –sdftyp c:/mti_vhdl/counter.sdo counter epf10k20rc208_a3
view *; add wave /*; add list /*; do run.do
```

Compiling the Altera VITAL library with the ModelSim UI

The User Interface Sequence to build the VITAL library and simulate the counter gate level design with SDF back-annotation timing will follow the exact same steps as the command line.

Using the Windows Explorer, make a working directory. This working directory will be where you will compile your VHDL gate level source files output of the Altera Max Plus II software (counter.vho and counter.sdo). In this example we will use the working directory: c:\mti_vhdl. Before you can compile your source files copied to this working directory, you must first compile the Altera VITAL Library. In this example the directory where you will create the ModelSim Altera Vital library will be in the Altera Max Plus II install tree. Once this library is created you will be able to reference it in all future designs.

After starting ModelSim from the new working directory, select **File** -> **Change Directory**, browse over (change directory) to the Altera VITAL Source files in the Altera install directory. In this example the install directory is d:\maxplus2. The full path to the VITAL source files is



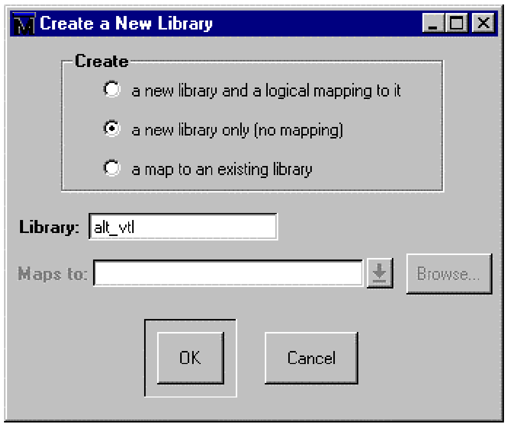
```
cd {D:\MAXPLUS2\VHDL87\VITAL\V3_0}

ModelSim> |
```

<No Design Loaded>

d:\maxplus2\vhdl87\vital\v3_0\alt_vtl. Once you have browsed to the directory the following command should be echoed in the main window:

```
cd {D:\MAXPLUS2\VHDL87\VITAL\V3_0}
```

To scroll through commands simply use the up and down arrows on the keyboard. In ModelSim, select **Library** -> Create **New Library,** press **a new library only (no mapping).** Type alt_vtl, press **OK**.

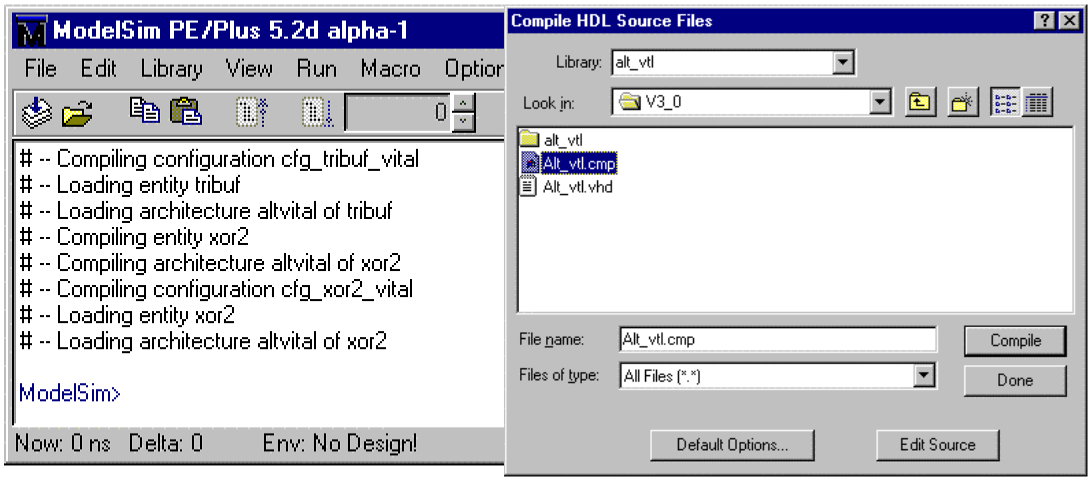


Create: ( ) a new library and a logical mapping to it [✓ a new library only (no mapping)] ( ) a map to an existing library

Library: alt_vtl

Maps to:

From the ModelSim application window, press the **Compile** button on the toolbar.
In the **Compile VHDL Source** dialog window, set the **Library** to alt_vtl (default is work).



```
# -- Compiling configuration cfg_tribuf_vital
# -- Loading entity tribuf
# -- Loading architecture altvital of tribuf
# -- Compiling entity xor2
# -- Compiling architecture altvital of xor2
# -- Compiling configuration cfg_xor2_vital
# -- Loading entity xor2
# -- Loading architecture altvital of xor2

ModelSim>
```

Now: 0 ns Delta: 0 Env: No Design!

Library: alt_vtl

Look in: V3_0

File name: Alt_vtl.cmp

Files of type: All Files (*.*)

Select the alt_vtl.vhd file and press **COMPILE**. The command and results will be echo in the transcript window. Note the **Default Options…** for VHDL and Verilog compile options.

In the **Files of Type** window, change to All Files (*.*) Verify that you compiled the VHDL files in the order they are shown:

```
alt_vtl.vhd
alt_vtl.cmp.vhd
```

You have created the Altera VITAL library.

VHDL Source Compile and SDF Timing Simulation with UI

Now that the Vital Library is compiled you must create a reference to it. Examination of the vhdl source file shows the use of the library alt_vhdl, this is the name of the Altera Vital Library.

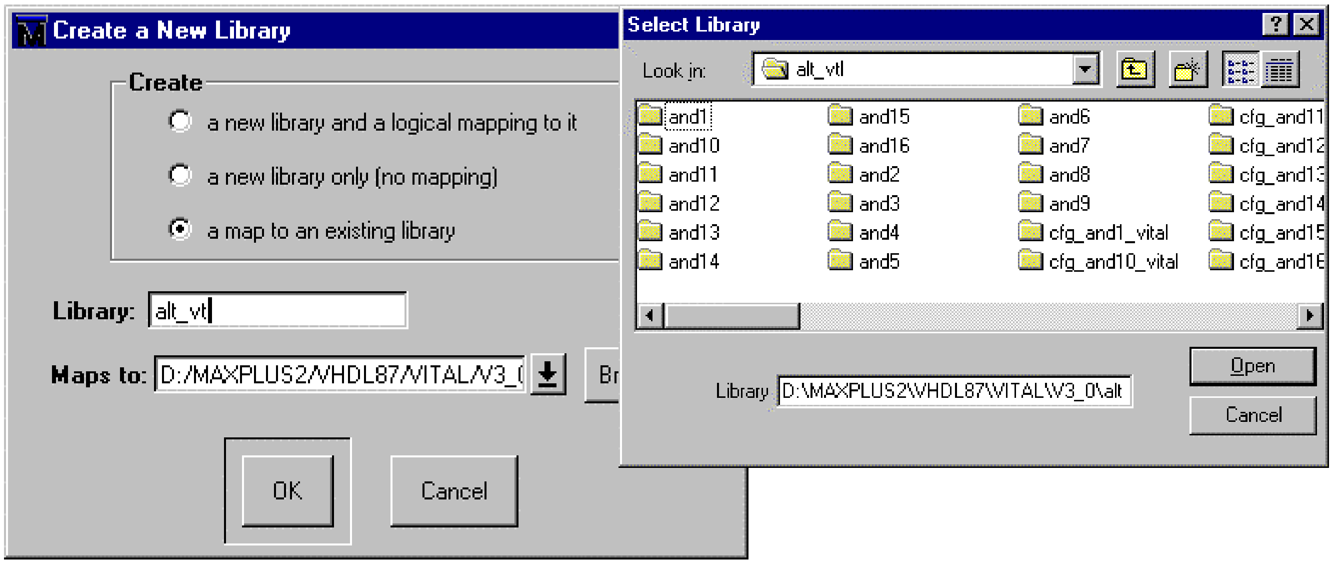
```
LIBRARY alt_vtl;
USE alt_vtl.VCOMPONENTS.all;
```

This mapping is accomplished with the vmap command. This command will modify a ModelSim initialization file "modelsim.ini". The modelsim.ini file can contain design specific information to be shared among users of the same library.

Move to the previously created working directory c:\mti_vhdl using **FILE** -> **Change Directory** browser. To create a library mapping select **Library** -> **Create a New Library** -> **a map to an existing library**.   Type alt_vtl for **Library**. Use the **browse** to **Select Library** alt_vtl that you just created (d:/maxplus2/vhdl87/vital/3_0/alt_vtl). The **Maps to** menu item should now include the full path D:/MAXPLUS2/VHDL87/VITAL/V3_0/alt_vtl. Select **OK**, you should see the following command in the transcript window:

```
exec vmap alt_vtl D:/MAXPLUS2/VHDL87/VITAL/V3_0/alt_vtl
# Modifying modelsim.ini
```

A local copy of the modelsim.ini is now in the design directory.



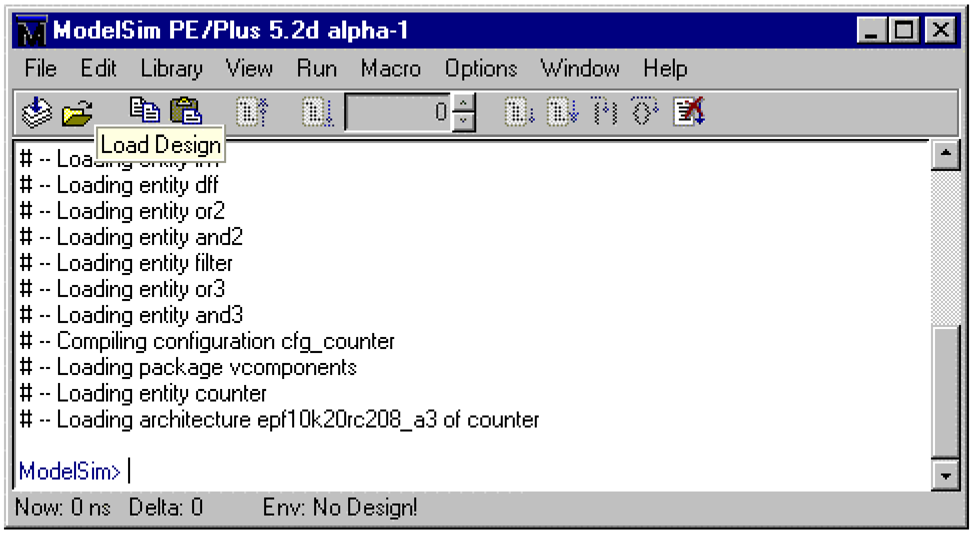
To compile the source files that were output from the Altera Maxplus II software select **Compile** to browse to the source, select **File** -> Change **Directory**, browse over (change directory) to C:\mti_vhdl. Create a library to contain the VHDL Source, In the **Library** -> **Create a New Library**, select **a new library only (no mapping)**, type **work** in the **Library** box, and Press **OK**. Note the following command in the transcript window:

```
exec vlib work
```

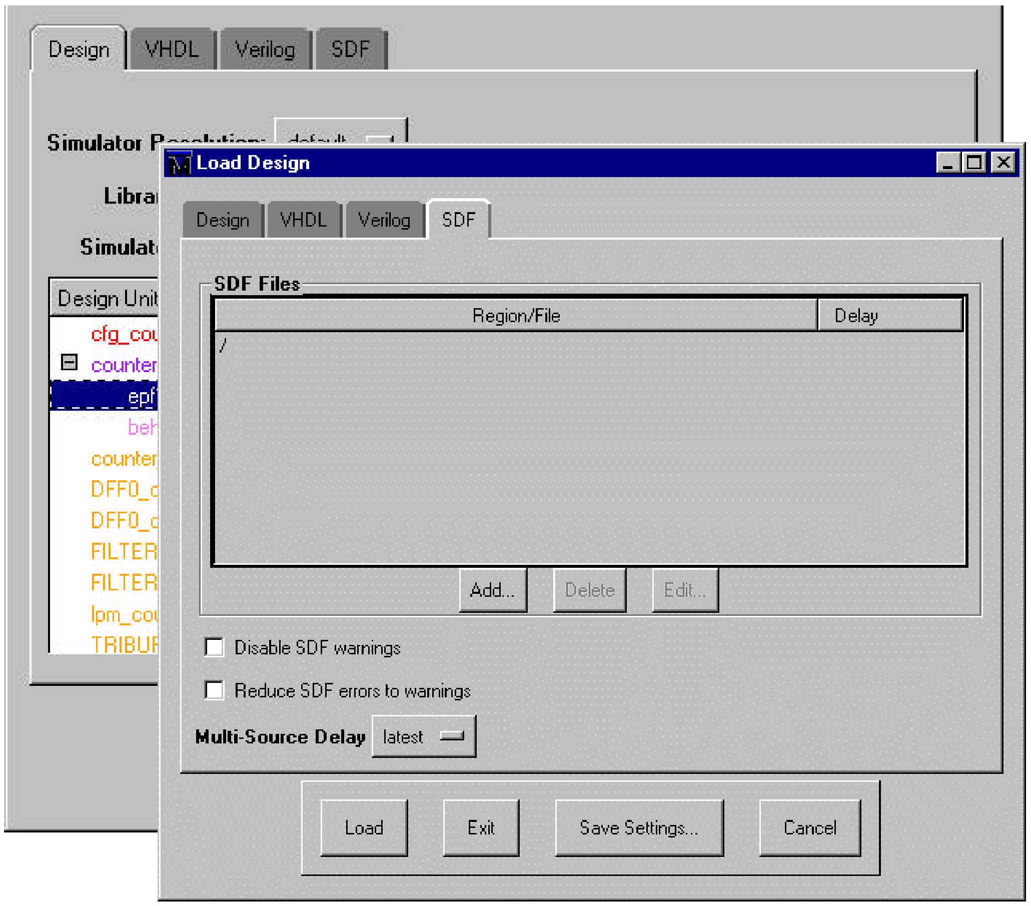
In the **Compile,** list **All Files** in the **List Files of Type**. Select the source file that was output from the Altera Max PlusII Place and Route software counter.vhd, press **Compile**. Notice the messages in the **Transcript** window. Press **Done** in **Compile**. This example only contains one file. The top level entity is counter. Note the following command in the **Transcript** window:

```
exec vcom counter.vhd
```

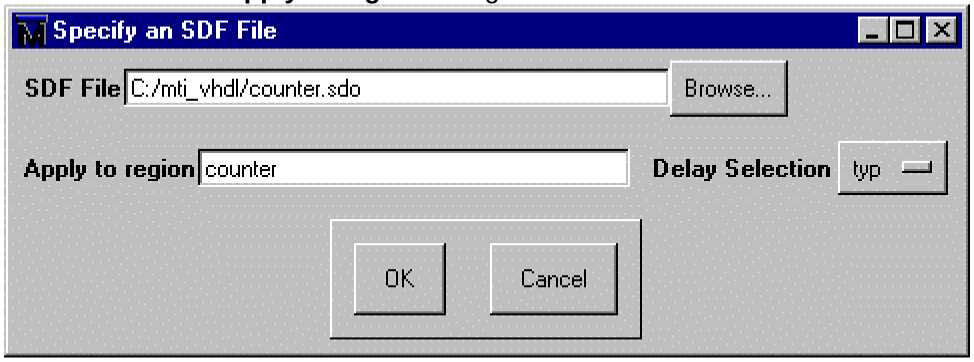
We will now simulate the design with the SDF file containing timing information. This file is output from the Altera Max PlusII software. We will apply the SDF timing information to the top level. Press **LOAD DESIGN** -> **SDF** tab



```
# -- Loading entity inv
# -- Loading entity dff
# -- Loading entity or2
# -- Loading entity and2
# -- Loading entity filter
# -- Loading entity or3
# -- Loading entity and3
# -- Compiling configuration cfg_counter
# -- Loading package vcomponents
# -- Loading entity counter
# -- Loading architecture epf10k20rc208_a3 of counter

ModelSim> |
Now: 0 ns   Delta: 0        Env: No Design!
```



Press **Add**…, in the **Specify an SDF File** dialog box enter C:/mti_vhdl/counter.sdo for **SDF File**, and counter in the **Apply to region** dialog box. Press **OK**.

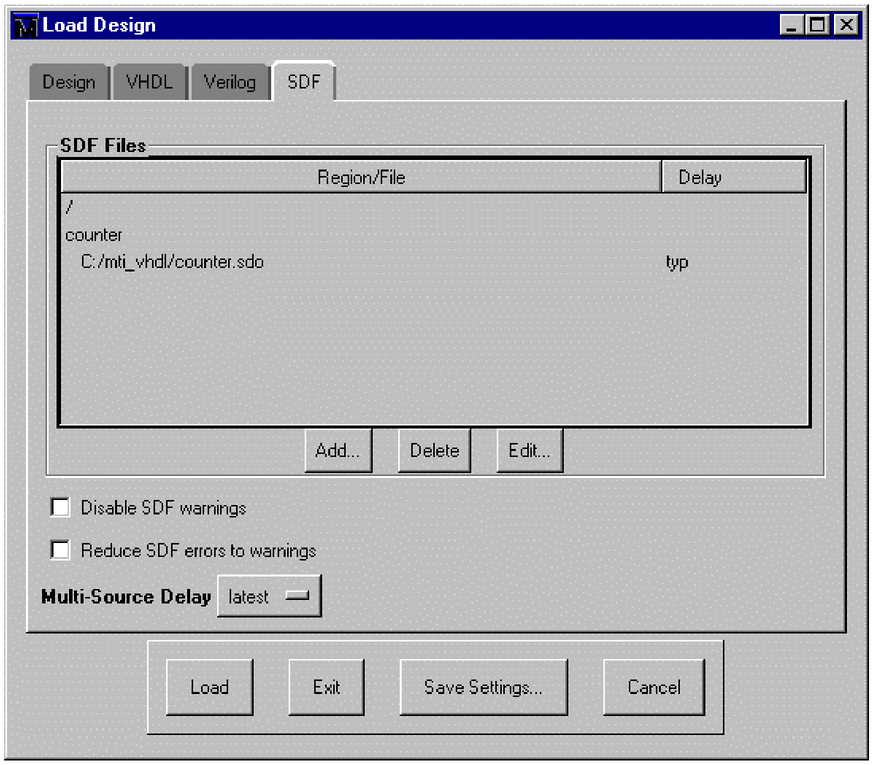

Specify an SDF File

SDF File: C:/mti_vhdl/counter.sdo

Apply to region: counter

Delay Selection: typ

The **Load Design** -> **SDF** tab will look as below
.



Region/File	Delay
/	
counter	
C:/mti_vhdl/counter.sdo	typ

☐ Disable SDF warnings

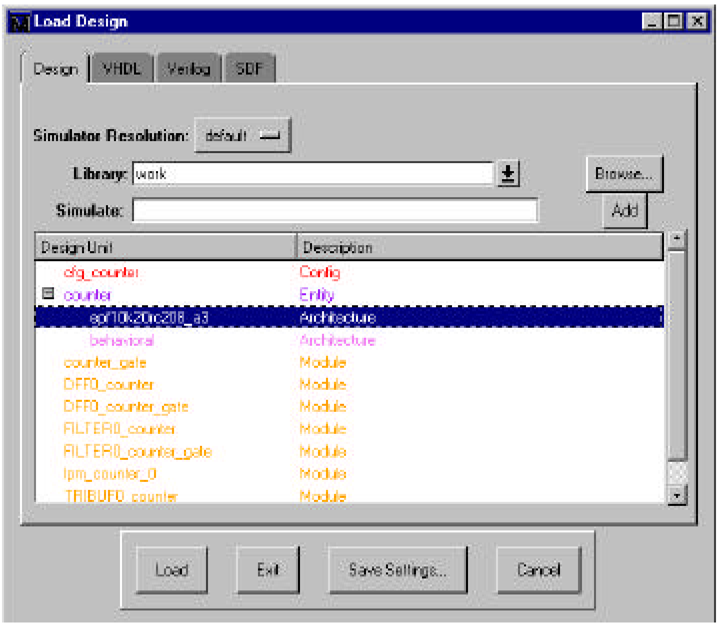
☐ Reduce SDF errors to warnings

Multi-Source Delay latest

In **Load Design** expand the counter **Design Unit** by pressing the **+** next to counter. Select the epf10k20rc208_a3 **Architecture**. Press **Load**. Note the following command echoed in the transcript:

```
vsim -sdftyp counter=C:/mti_vhdl/counter.sdo -multisource_delay latest
work.counter(epf10k20rc208_a3)
```

You are now ready to execute simulation commands.

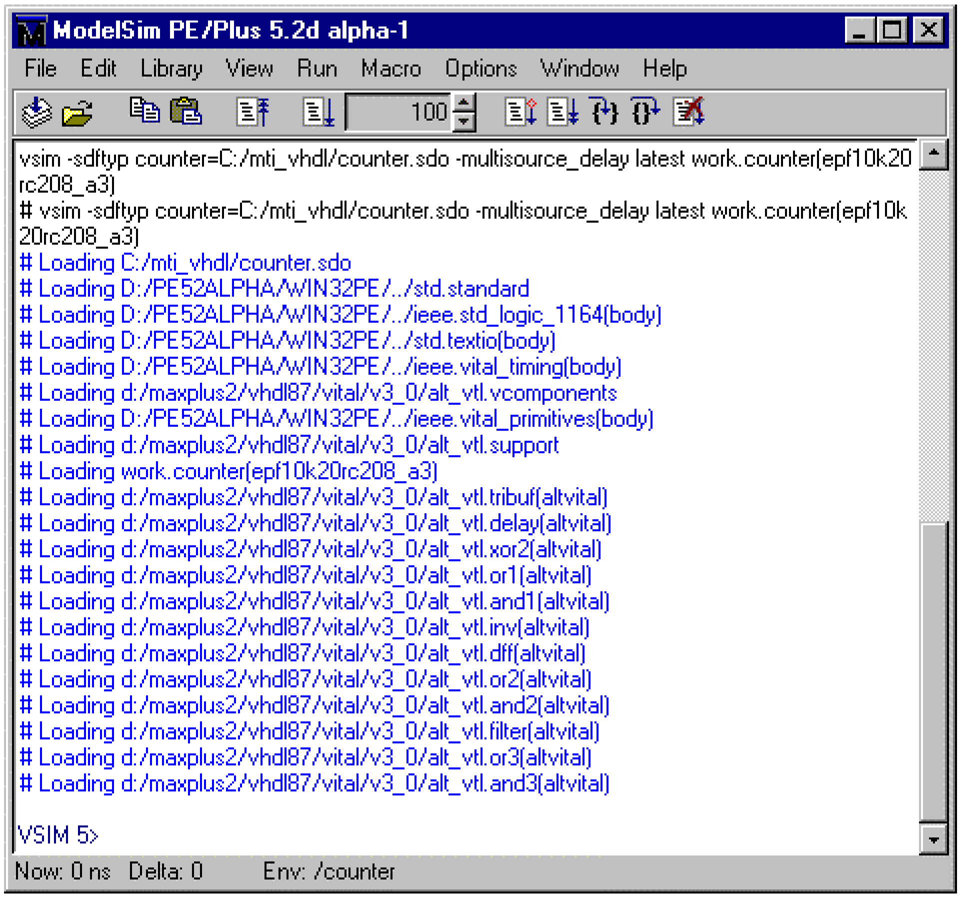


Simulator Resolution: default

Library: work

Simulate:

Design Unit	Description
cfg_counter	Config
⊟ counter	Entity
epf10k20rc208_a3	Architecture
behavioral	Architecture
counter_gate	Module
DFF0_counter	Module
DFF0_counter_gate	Module
FILTER0_counter	Module
FILTER0_counter_gate	Module
lpm_counter_0	Module
TRIBUF0_counter	Module


ModelSim PE/Plus 5.2d alpha-1
File Edit Library View Run Macro Options Window Help
100

```
vsim -sdftyp counter=C:/mti_vhdl/counter.sdo -multisource_delay latest work.counter(epf10k20
rc208_a3)
# vsim -sdftyp counter=C:/mti_vhdl/counter.sdo -multisource_delay latest work.counter(epf10k
20rc208_a3)
# Loading C:/mti_vhdl/counter.sdo
# Loading D:/PE52ALPHA/WIN32PE/../std.standard
# Loading D:/PE52ALPHA/WIN32PE/../ieee.std_logic_1164(body)
# Loading D:/PE52ALPHA/WIN32PE/../std.textio(body)
# Loading D:/PE52ALPHA/WIN32PE/../ieee.vital_timing(body)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.vcomponents
# Loading D:/PE52ALPHA/WIN32PE/../ieee.vital_primitives(body)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.support
# Loading work.counter(epf10k20rc208_a3)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.tribuf(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.delay(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.xor2(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.or1(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.and1(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.inv(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.dff(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.or2(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.and2(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.filter(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.or3(altvital)
# Loading d:/maxplus2/vhdl87/vital/v3_0/alt_vtl.and3(altvital)

VSIM 5>
Now: 0 ns   Delta: 0        Env: /counter
```

Using a macro file

If you place all the commands echoed in the transcript window above into a file (i.e. altera.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.

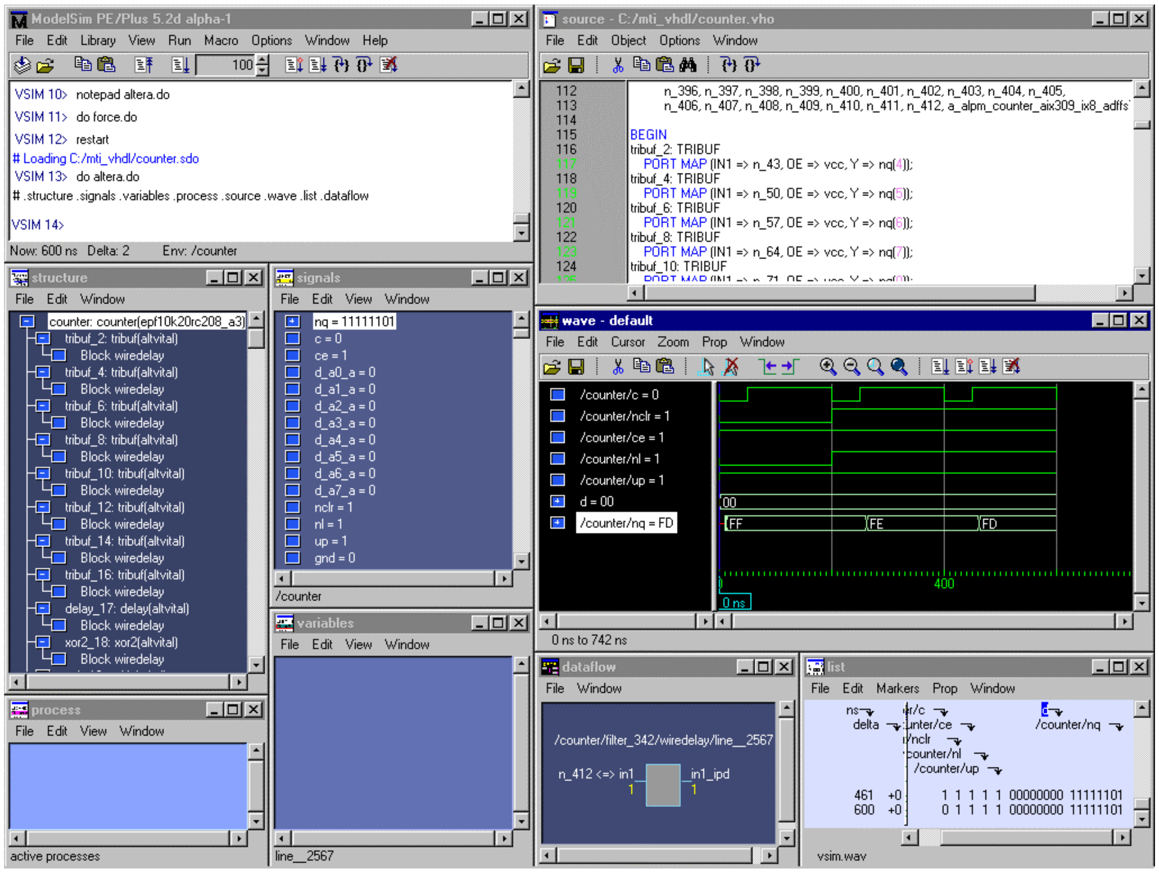
From the Model*Sim* main window, chose **Macro -> Execute Macro** and browse to the altera.do file in the **Execute Do File** dialog box. This will execute the macro file. Please note the command echoed in the **Transcript** window.

```
do C:/mti_vhdl/altera.do
```

Below is an example script (altera.do) and the results of a run:

```
add wave c
add wave nclr
add wave ce
add wave nl
add wave up
add wave -hex {d {d_a0_a d_a1_a d_a2_a d_a3_a d_a4_a d_a5_a d_a6_a d_a7_a }}
add wave -hex nq
add list c
add list nclr
add list ce
add list nl
add list up
add list -hex {d {d_a0_a d_a1_a d_a2_a d_a3_a d_a4_a d_a5_a d_a6_a d_a7_a }}
add list -hex nq
view *
force c 0 0 ns , 1 { 50 ns } -repeat 200 ns
force ce 1 ns
force nl 0 ns
force nclr 0 ns
force up 1 ns
force d_a0_a 0 ns
force d_a1_a 0 ns
force d_a2_a 0 ns
force d_a3_a 0 ns
force d_a4_a 0 ns
force d_a5_a 0 ns
force d_a6_a 0 ns
force d_a7_a 0 ns
run 200 ns
force nl 1 ns
force nclr 1 ns
run 400 ns
write list new.lst
```

Below is an example of all available UI windows with a brief description of each window.



ModelSim PE/Plus 5.2d alpha-1
File Edit Library View Run Macro Options Window Help
source - C:/mti_vhdl/counter.vho
File Edit Object Options Window
structure
File Edit Window
signals
File Edit View Window
variables
process
wave - default
File Edit Cursor Zoom Prop Window
dataflow
File Window
list
File Edit Markers Prop Window

```
VSIM 10>  notepad altera.do
VSIM 11>  do force.do
VSIM 12>  restart
# Loading C:/mti_vhdl/counter.sdo
VSIM 13>  do altera.do
# .structure .signals .variables .process .source .wave .list .dataflow

VSIM 14>
Now: 600 ns   Delta: 2        Env: /counter
```

```
        n_396, n_397, n_398, n_399, n_400, n_401, n_402, n_403, n_404, n_405,
        n_406, n_407, n_408, n_409, n_410, n_411, n_412, a_alpm_counter_aix309_ix8_adffs

BEGIN
tribuf_2: TRIBUF
   PORT MAP (IN1 => n_43, OE => vcc, Y => nq(4));
tribuf_4: TRIBUF
   PORT MAP (IN1 => n_50, OE => vcc, Y => nq(5));
tribuf_6: TRIBUF
   PORT MAP (IN1 => n_57, OE => vcc, Y => nq(6));
tribuf_8: TRIBUF
   PORT MAP (IN1 => n_64, OE => vcc, Y => nq(7));
tribuf_10: TRIBUF
```

```
counter: counter(epf10k20rc208_a3)
  tribuf_2: tribuf(altvital)
    Block wiredelay
  tribuf_4: tribuf(altvital)
    Block wiredelay
  tribuf_6: tribuf(altvital)
    Block wiredelay
  tribuf_8: tribuf(altvital)
    Block wiredelay
  tribuf_10: tribuf(altvital)
    Block wiredelay
  tribuf_12: tribuf(altvital)
    Block wiredelay
  tribuf_14: tribuf(altvital)
    Block wiredelay
  tribuf_16: tribuf(altvital)
    Block wiredelay
  delay_17: delay(altvital)
    Block wiredelay
  xor2_18: xor2(altvital)
    Block wiredelay
```

```
nq = 11111101
c = 0
ce = 1
d_a0_a = 0
d_a1_a = 0
d_a2_a = 0
d_a3_a = 0
d_a4_a = 0
d_a5_a = 0
d_a6_a = 0
d_a7_a = 0
nclr = 1
nl = 1
up = 1
gnd = 0
/counter
```

```
/counter/c = 0
/counter/nclr = 1
/counter/ce = 1
/counter/nl = 1
/counter/up = 1
d = 00
/counter/nq = FD
00
FF    FE    FD
0     400
0 ns
0 ns to 742 ns
```

```
active processes
line__2567
```

```
/counter/filter_342/wiredelay/line__2567
n_412 <=> in1        in1_ipd
          1          1
```

```
   ns     er/c                  d
  delta   unter/ce              /counter/nq
          r/nclr
          counter/nl
           /counter/up

   461  +0    1 1 1 1 1  00000000 11111101
   600  +0    0 1 1 1 1  00000000 11111101
vsim.wav
```

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.

Variables – Lists the names of the HDL items within the current process, followed by the current value(s) associated with each name. The HDL items for VHDL are constants, generics and variables. The HDL items for Verilog are register variables.

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.

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

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

Each window is capable of cross-highliging. For example you may select an item in the structure window and the, variables, signals, process, source window will reflect its contents. You may also select, drag and drop a from one window to the wave or list window. For example select a signal and drag and drop into the wave or list window. You may also select a variable in the source window and do similar process. Please refer to the ModelSim refrence manual for a more complete understanding of all the UI features.

Gate Level Verilog Library and Source Compile and Simulation With ModelSim Command Line

The following section describes how to set up the Altera Verilog libraries and Altera Gate Level source files for using the Model*Sim* command line. Following section will explain how to use the ModelSim User Interface to accomplish the same tasks.

Input files for ModelSim from Altera place and route tools:

counter.vo	(Verilog Gate Level file based on the original counter.vhd RTL which was synthesized, and Place and Routed)
alt_max2.vo	(Altera Verilog Gate Level cells generated when synthesized counter.edf was Place and Routed)
counter.sdo	(SDF file with timing information)

Command Sequence to build the Verilog library and simulate the counter gate level design with SDF back-annotation timing:

```
vlib altera
vlog -work altera  alt_max2.vo
vlib work
vlog counter.vo
vsim -sdftyp C:/mti_verilog/counter.sdo -L C:/mti_verilog/altera  counter
view *; add wave /*; do run.do
```

Compiling the Altera Verilog library with the ModelSim UI

Using the Windows Explorer, make a working directory. This working directory will be where you will compile your Verilog gate level source files output of the Altera Max Plus II software. In this example we will use the working directory: c:\mti_verilog. As you prepare to compile the source files you must also copy the Altera Verilog Library to your working directory. The default name for this file is alt_max2.vo. This file is generated by the Altera Max Plus II place and route Verilog netlist generation tool. It is generated in the same directory as the counter.vho output netlist. The alt_max2.vo file is design specific, we will compile the Altera Verilog Library in the c:\mti_verilog directory as a library with name arbitrary name altera.

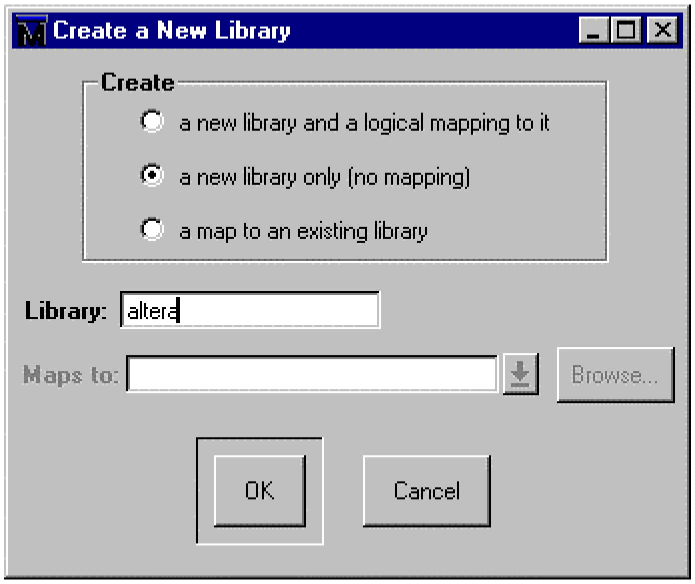
After starting ModelSim select **File** -> **Change Directory**, browse to (change directory) the c:\mti_verilog directory where you copied the Verilog gate level file(s) and the alt_max2.vo file. As commands are entered in the pulldown menues or command line, you can scroll through them with the up and down arrows on the keyboard. Once you have browsed to the directory the following command should be echoed in the **Transcript** window:

```
cd { C:\MTI_VERILOG }
```

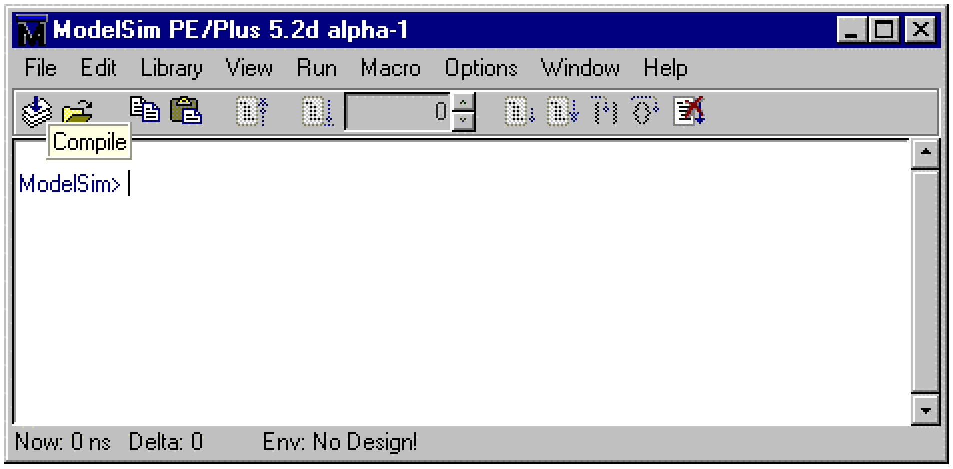
Compile Altera Verilog reference library

In ModelSim select **Library** -> Create **New Library,** press **a new library only (no mapping).** Type altera, press **OK**. Note the following command echoed in the **Transcript**:

```
exec vlib altera
```


Create a New Library
Create
a new library and a logical mapping to it
a new library only (no mapping)
a map to an existing library
Library: altera
Maps to:
Browse...
OK
Cancel

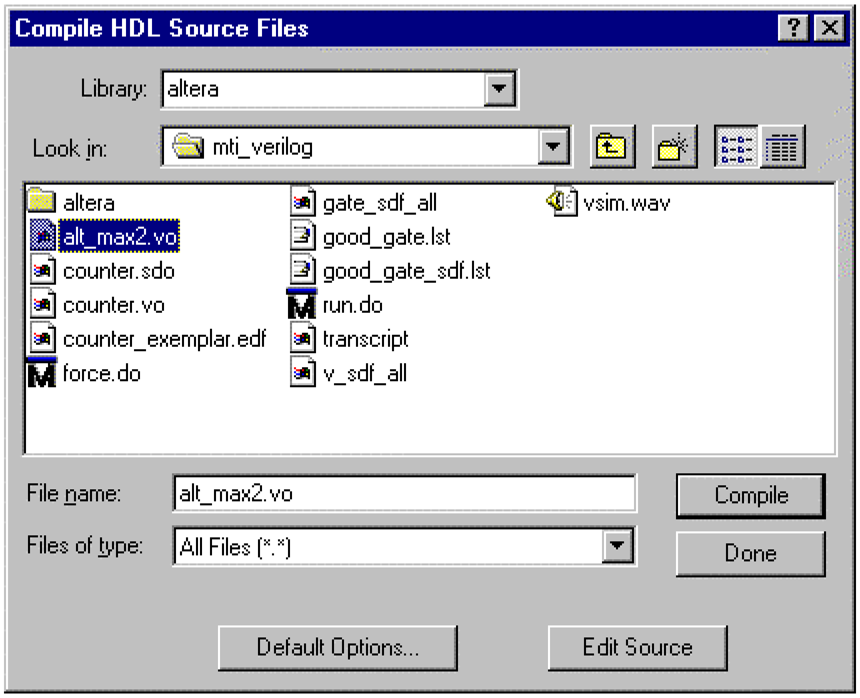
From the Model*Sim* application window, press the **Compile** button on the toolbar.



In the **Compile HDL Source Files** dialog window, set the **Library** to altera (default is work). In the **Files of Type** window, select All Files (*.*). Select the alt_max2.vo and press Compile. You will see the results of the compile in the transcript window. The following command will echo:

```
vlog -work altera {C:/mti_verilog/alt_max2.vo}
```

Press **Done** in the **Compile HDL Source Files**. You have compiled the Altera verilog library for this design.



Verilog Source Compile and SDF Timing Simulation

Now that you have a complete Altera Verilog reference library you can compile your Verilog Gate Level source file(s).

In Model*Sim*, select **File** -> **Change Directory**, browse to (change directory) the c:\mti_verilog directory where you copied the Verilog gate level file(s) and the counter.vo file. As commands are entered in the pulldown menues or command line, you can scroll through them with the up and down arrows on the keyboard. Once you have browsed to the directory the following command should be echoed in the main window:

```
cd { C:\MTI_VERILOG }
```

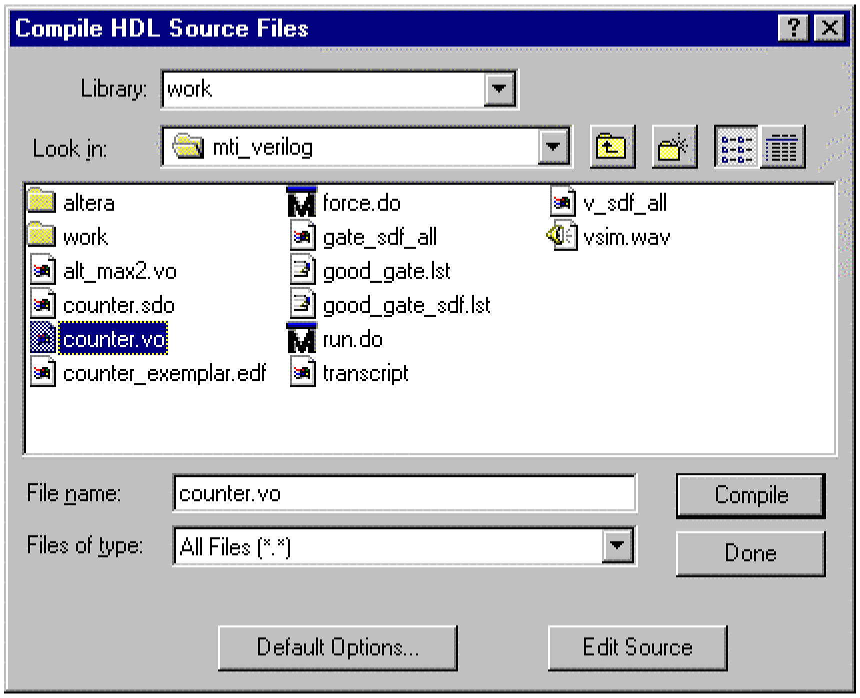
In Model*Sim*, select **Library** -> Create **New Library,** press **a new library only (no mapping).** Type work, press **OK**. Note the following command echoed in the transcript:

```
exec vlib work
```

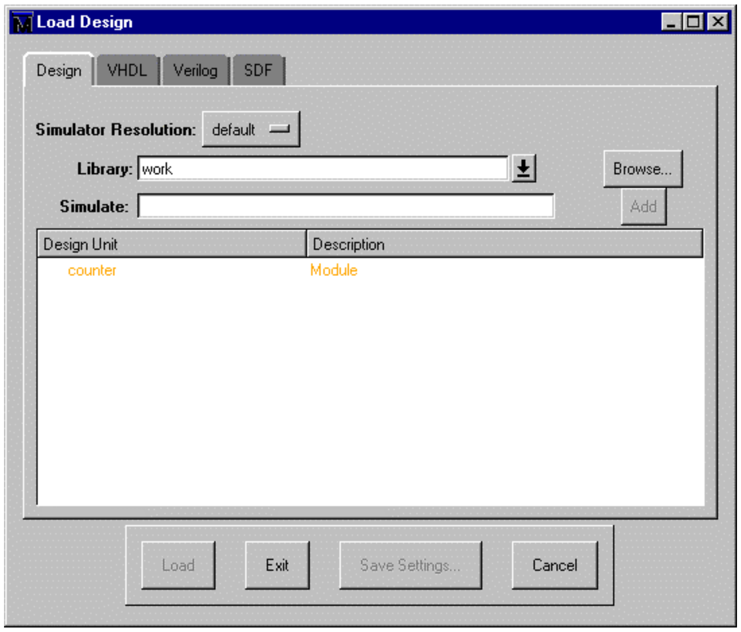
In the **Compile HDL Source Files** dialog window, set the **Library** to work (default is work). In the **Files of Type** window, select All Files (*.*). Select the counter.vo and press Compile. You will see the results of the compile in the transcript window. The following command will echo:

```
vlog -work work {C:/mti_verilog/counter.vo}
```

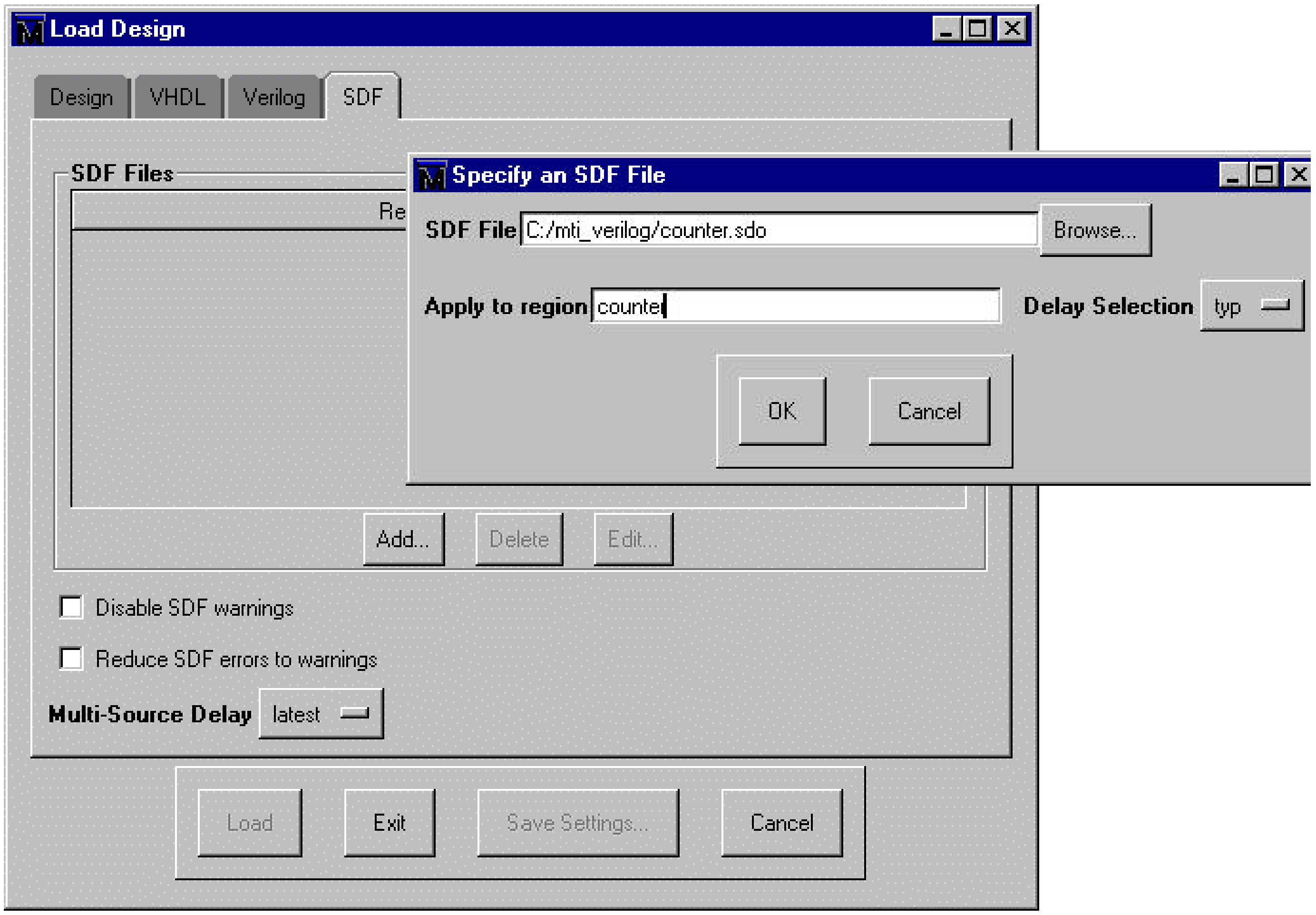
This example only contains one file, press done in the **Compile HDL Source Files**.



To Simulate the design press the **Load Design** icon in the transcript window. The top level entity is counter. We will simulate the design with timing information (SDF/SDO file) which contains timing information for counter. This file is output from the Altera Max PlusII software.



We will apply the SDF timing information to the top level. Press **Load Design -> SDF** tab, Press **Add**, use the browse button or enter the path to the SDF file. The name is counter.sdo.



SDF File C:/mti_verilog/counter.sdo

Apply to region counter

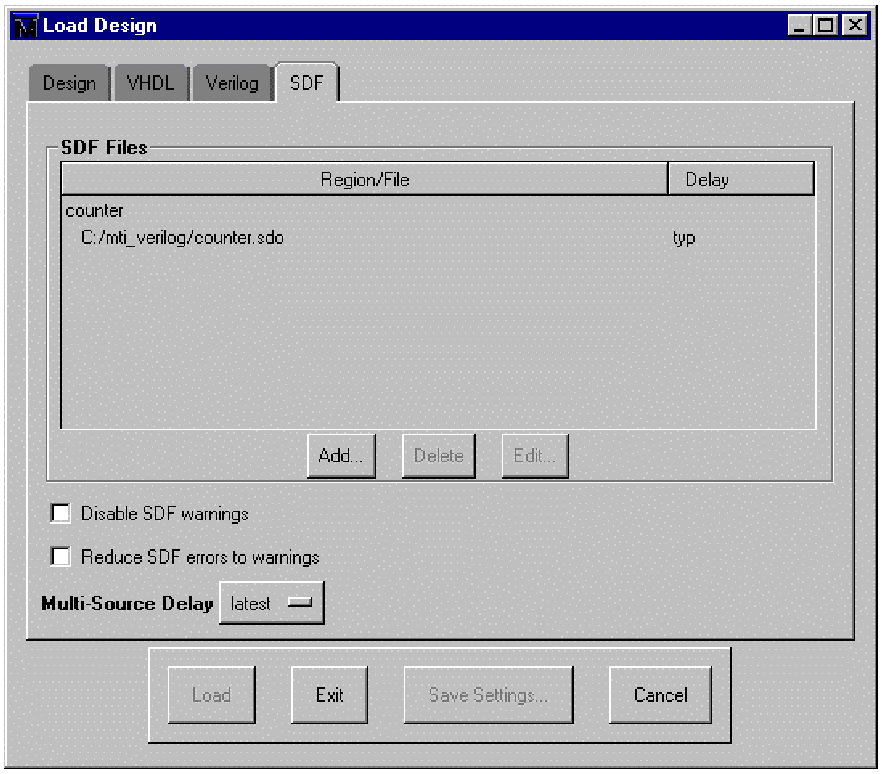
Delay Selection typ

☐ Disable SDF warnings

☐ Reduce SDF errors to warnings

Multi-Source Delay latest

In the **SDF File** dialog box, select **all files (*.*)**, select counter.sdo and press **Open**. In the **Specify and SDF File** type counter in **Apply to region** dialog box. Press **OK**. You will see the following in **Load Design**:



SDF Files

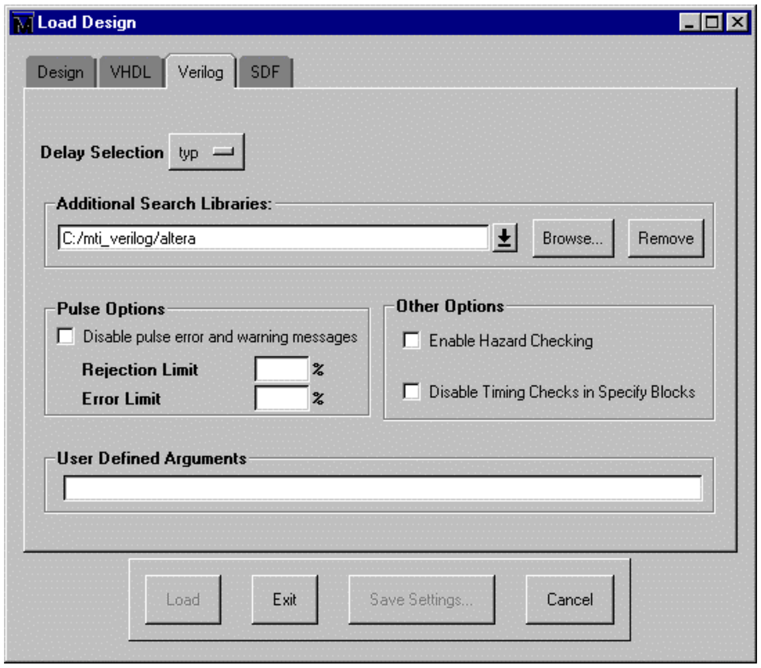
Region/File	Delay
counter	
C:/mti_verilog/counter.sdo	typ

☐ Disable SDF warnings

☐ Reduce SDF errors to warnings

Multi-Source Delay: latest

Press on the **Verilog** tab. Press **Browse** in the **Additional Search Libraries**. Select altera in the **Select Library dialog** box. Press **Open**. The dialog box will be as shown below.



Delay Selection: typ

Additional Search Libraries: C:/mti_verilog/altera

Pulse Options
☐ Disable pulse error and warning messages
Rejection Limit %
Error Limit %

Other Options
☐ Enable Hazard Checking
☐ Disable Timing Checks in Specify Blocks

User Defined Arguments

Press on the **Design** tab. Select the **Design Unit** counter, press **Load**. You will see the following command echoed in the transcript as the design is being elaborated:

```
vsim +typdelays -L C:/mti_verilog/altera -sdftyp counter=C:/mti_verilog/counter.sdo -
multisource_delay latest work.counter
```

Note the –L option will reference the additional altera library.

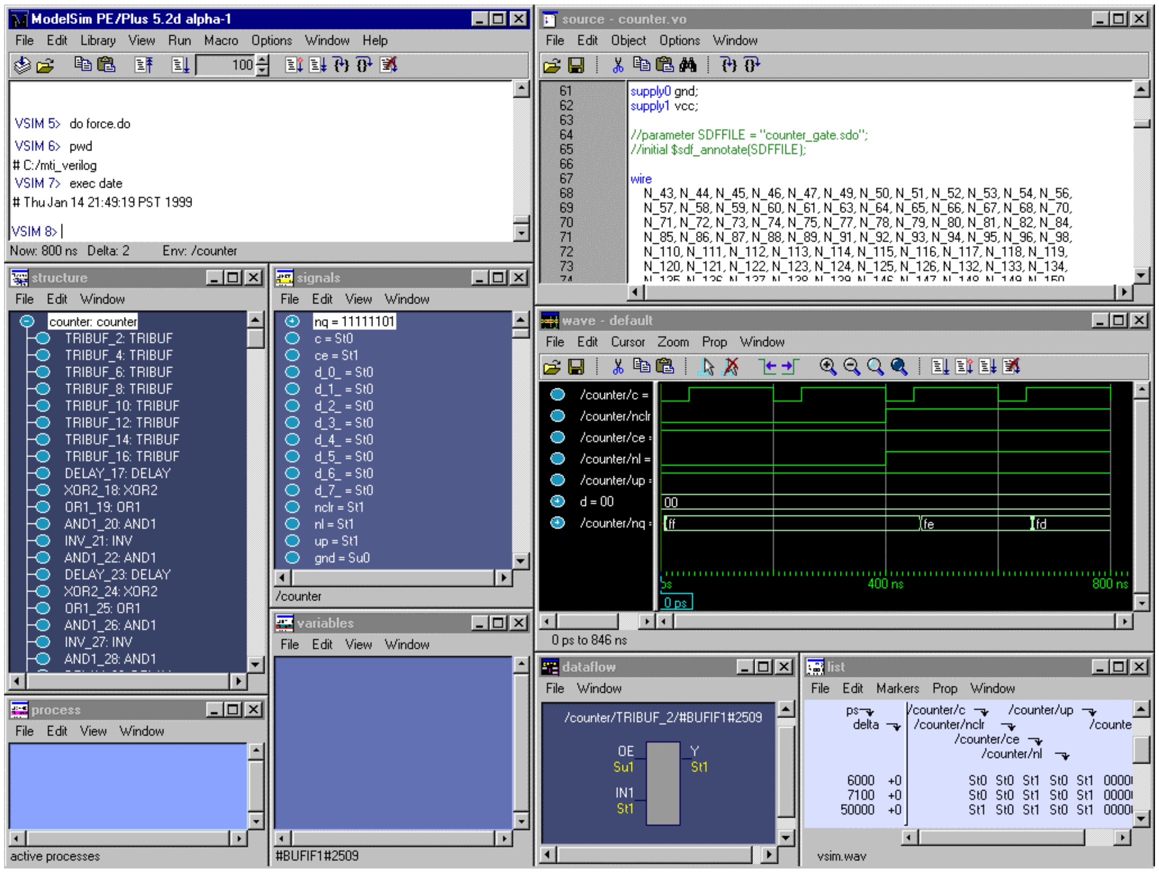
Simulating the Design Using a macro file

If you place all the above ModelSim commands into a file (i.e. altera.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:

From the Model*Sim* main window, chose **Macro -> Execute Macro…** and browse to the altera.do file.

Select the altera.do file and press **Open**. This will execute the macro file. Please note the command echoed in the **transcript** window:

```
do C:/mti_verilog/run.do
```



```
VSIM 5>  do force.do
VSIM 6>  pwd
# C:/mti_verilog
VSIM 7>  exec date
# Thu Jan 14 21:49:19 PST 1999

VSIM 8> |
Now: 800 ns   Delta: 2        Env: /counter
```

```
supply0 gnd;
supply1 vcc;

//parameter SDFFILE = "counter_gate.sdo";
//initial $sdf_annotate(SDFFILE);

wire
   N_43, N_44, N_45, N_46, N_47, N_49, N_50, N_51, N_52, N_53, N_54, N_56,
   N_57, N_58, N_59, N_60, N_61, N_63, N_64, N_65, N_66, N_67, N_68, N_70,
   N_71, N_72, N_73, N_74, N_75, N_77, N_78, N_79, N_80, N_81, N_82, N_84,
   N_85, N_86, N_87, N_88, N_89, N_91, N_92, N_93, N_94, N_95, N_96, N_98,
   N_110, N_111, N_112, N_113, N_114, N_115, N_116, N_117, N_118, N_119,
   N_120, N_121, N_122, N_123, N_124, N_125, N_126, N_132, N_133, N_134,
```

```
counter: counter
  TRIBUF_2: TRIBUF
  TRIBUF_4: TRIBUF
  TRIBUF_6: TRIBUF
  TRIBUF_8: TRIBUF
  TRIBUF_10: TRIBUF
  TRIBUF_12: TRIBUF
  TRIBUF_14: TRIBUF
  TRIBUF_16: TRIBUF
  DELAY_17: DELAY
  XOR2_18: XOR2
  OR1_19: OR1
  AND1_20: AND1
  INV_21: INV
  AND1_22: AND1
  DELAY_23: DELAY
  XOR2_24: XOR2
  OR1_25: OR1
  AND1_26: AND1
  INV_27: INV
  AND1_28: AND1
```

```
nq = 11111101
c = St0
ce = St1
d_0_ = St0
d_1_ = St0
d_2_ = St0
d_3_ = St0
d_4_ = St0
d_5_ = St0
d_6_ = St0
d_7_ = St0
nclr = St1
nl = St1
up = St1
gnd = Su0
/counter
```

```
      ps     /counter/c      /counter/up
   delta    /counter/nclr              /counte
                  /counter/ce
                      /counter/nl

    6000  +0         St0  St0  St1  St0  St1  0000
    7100  +0         St0  St0  St1  St0  St1  0000
   50000  +0         St1  St0  St1  St0  St1  0000
```

You can use a testbench or the force commands to stimulate the design.

All VHDL Source Files can be found at:

http://www.model.com/pdf/mti_altera_vhdl.zip

All Verilog Source Files can be found at:
http://www.model.com/pdf/mti_altera_verilog.zip

List of all source files:

RTL VHDL RTL source netlist:

counter.vhd	VHDL RTL source file
run.do	VHDL compile script
force.do	VHDL stimulus script

Altera VHDL Source netlists:

alt_vtl.cmp	VHDL Altera library
alt_vtl.vhd	VHDL Altera library
counter.sdo	VHDL gate level timing file
counter.vho	VHDL gate level file
run.do	VHDL compile script
force.do	VHDL stimulus script

Altera Verilog Source netlists:

alt_max2.vo	Verilog Altera library
counter.sdo	Verilog gate level timing file
counter.vo	Verilog gate level file
run.do	Verilog compile script
force.do	Verilog stimulus script