MicroBlaze Tutorial on EDK 10.1 using Sparatan III E
Behavioural Simulation of MicroBlaze System

Ahmed Elhossini

January 24, 2010

1 Introduction

1.1 Objectives

This tutorial will demonstrate process of simulating a MicroBlaze system using the Embedded Development Kit (EDK) and ModelSim. ModelSim is an HDL simulation tool that is able to perform several types of simulations. In this tutorial we will use ModelSim to perform behavioral and timing simulations for an embedded system based on MircoBlaze. Behavioral simulation is used to test the functionality of the system using an abstracted model of the different component of the system. This simulation allows the designer to check the functionality of the system before moving to other stages of the design process. Timing simulation is performed on an HDL model of the system after completing the place and route phase. This model include a detailed information about the design. This simulation shows the behavior of the system when it is physically implemented on the FPGA.

1.2 System Requirements

You must have the following software installed on your PC to complete this tutorial:

- Windows 2000 SP2/Windows XP
- EDK 10.1i.
- ISE 10.1i.
- ModelSim SE 6.3c.
- Familiarity with Xilinx ISE 10.1 design flow.
- Complete the tutorial: "MicroBlaze Tutorial on EDK 10.1 using Sparatan III E". We will use the same MicroBlaze system for simulation in this tutorial.

Note: The design is based on the Spartan III E Starter Kit. However the board itself is not required to complete this tutorial.

1.3 Tutorial Steps

In this tutorial, the following steps will be performed:

- Preparing the MicroBlaze system for simulation using EDK.

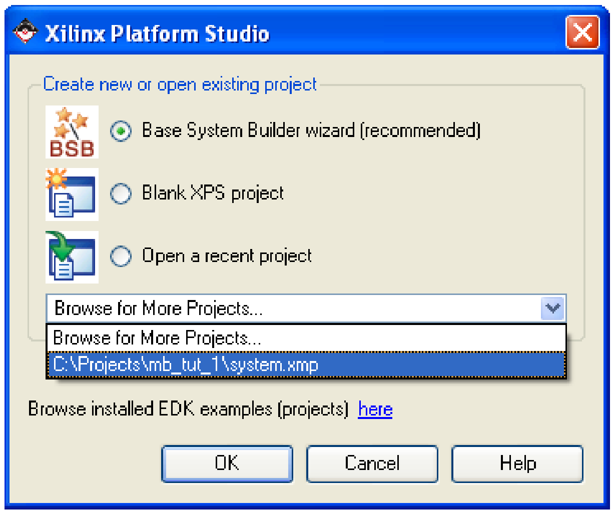

Xilinx Platform Studio
Create new or open existing project
BSB
Base System Builder wizard (recommended)
Blank XPS project
Open a recent project
Browse for More Projects...
Browse for More Projects...
C:\Projects\mb_tut_1\system.xmp
Browse installed EDK examples (projects) here
OK
Cancel
Help

Figure 1: Open an Existing Design

- Building the behavioral simulation model for the MicroBlaze system.
- Performing behavioral simulation using ModelSim.
- Building the timing simulation model for the MicroBlaze system.
- Performing timing simulation using ModelSim.

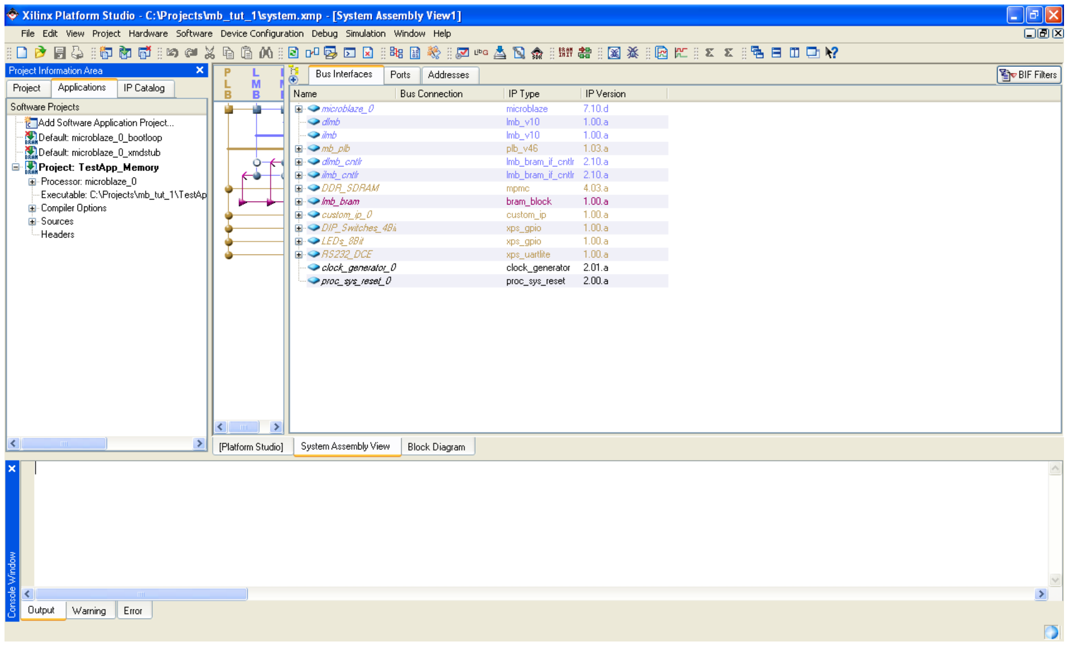
By the end of this tutorial you will be able to simulate the EDK design using ModelSim.

2 Preparing the Design for Simulation

The first step in this tutorial is to prepare the MicoBlaze system for simulation. This involves building the simulation library required by ModelSim to simulate EDK systems. Follow the following steps:

- Open XPS, by **selecting** Start → All Programs → Development → Xilinx ISE Design Suite 10.1 → EDK → Xilinx Platform Studio
- Open the design you created in the tutorial "MicroBlaze Tutorial on EDK 10.1 using Sparatan III E" as shown in Figure 1.

  The design will be loaded as shown in Figure 2.
- Now we will start building the simulation library. **Select** Simulation → Compile Simulation Libraries. This step is required only once as the simulation library can be used later to simulate any other system.This will start the "Simulation Library Compilation Wizard" that will be used to setup the compilation process of the simulation library as shown in Figure 3. **Click Next**.
- The second screen of the wizard will be used to select the simulation software as shown in Figure 4. EDK simulation supports only two types of simulators: i) ModelSim 6.3c SE and ii) NCSim which is not supported on windows platform. For this reason ModelSim 6.3c is used in this tutorial to simulate MicoBlaze system. It is important to know that if ModelSim 6.3c is not installed in your system, the wizard will not continue. **Select ModelSim 6.3c** and then **Click Next**.

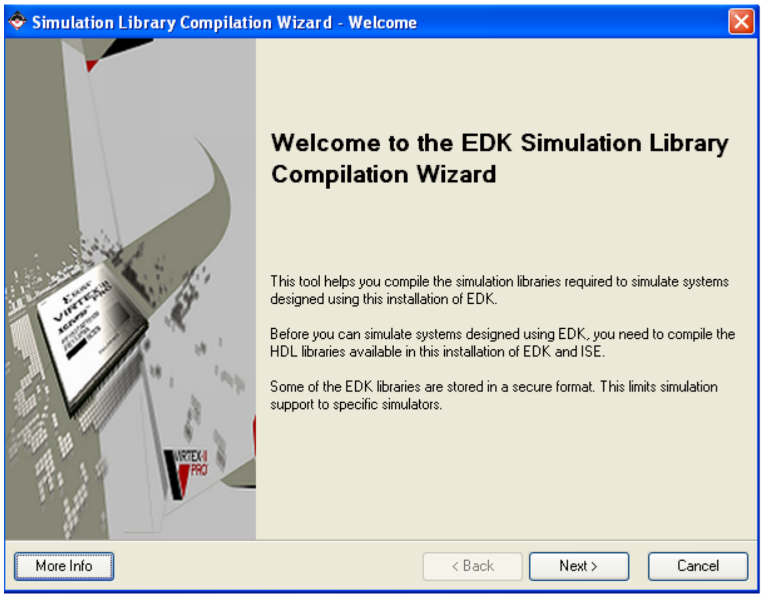


Xilinx Platform Studio - C:\Projects\mb_tut_1\system.xmp - [System Assembly View1]

Software Projects
- Add Software Application Project...
- Default: microblaze_0_bootloop
- Default: microblaze_0_xmdstub
- **Project: TestApp_Memory**
  - Processor: microblaze_0
  - Executable: C:\Projects\mb_tut_1\TestAp
  - Compiler Options
  - Sources
  - Headers

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.d
dlmb		lmb_v10	1.00.a
ilmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.03.a
dlmb_cntlr		lmb_bram_if_cntlr	2.10.a
ilmb_cntlr		lmb_bram_if_cntlr	2.10.a
DDR_SDRAM		mpmc	4.03.a
lmb_bram		bram_block	1.00.a
custom_ip_0		custom_ip	1.00.a
DIP_Switches_4Bi		xps_gpio	1.00.a
LEDs_8Bit		xps_gpio	1.00.a
RS232_DCE		xps_uartlite	1.00.a
clock_generator_0		clock_generator	2.01.a
proc_sys_reset_0		proc_sys_reset	2.00.a

Figure 2: Loaded Design



Simulation Library Compilation Wizard - Welcome

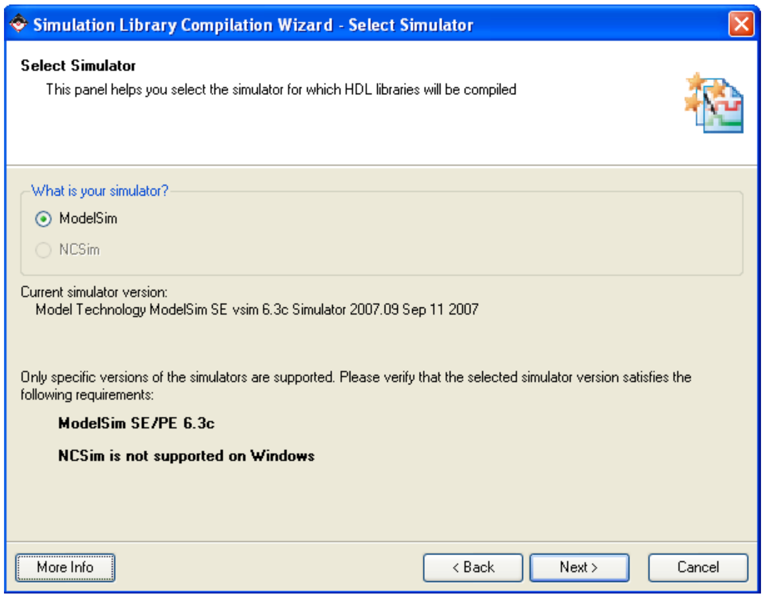
Welcome to the EDK Simulation Library Compilation Wizard

This tool helps you compile the simulation libraries required to simulate systems designed using this installation of EDK.

Before you can simulate systems designed using EDK, you need to compile the HDL libraries available in this installation of EDK and ISE.

Some of the EDK libraries are stored in a secure format. This limits simulation support to specific simulators.

Figure 3: Simulation Library Compilation Wizard - Welcome


Simulation Library Compilation Wizard - Select Simulator

Select Simulator

This panel helps you select the simulator for which HDL libraries will be compiled

What is your simulator?

(●) ModelSim

( ) NCSim

Current simulator version:
Model Technology ModelSim SE vsim 6.3c Simulator 2007.09 Sep 11 2007

Only specific versions of the simulators are supported. Please verify that the selected simulator version satisfies the following requirements:

ModelSim SE/PE 6.3c

NCSim is not supported on Windows

Figure 4: Simulation Library Compilation Wizard - Select Simulator

- The third screen of the wizard is used to select the HDL language used by the simulator as shown in Figure 5. ModelSim 6.3c supports both VHDL and Verilog. It is up to you to select which language, however, selecting both language will allow to use any of them if required by it will require more disk space. **Select Both VHDL and Verilog** and then **Click Next**.

- The fourth screen of the wizard is used to select the directory in which ISE simulation models are compiled as shown in Figure 6. You can select any location to store the library files. As mentioned earlier the library could be used by any design for simulation, so selecting the library location is important. **Select a suitable location and Click Next**.

- The next screen of the wizard is used to select the directory in which EDL simulation models are compiled as shown in Figure 7. **Select a suitable location and Click Next**.

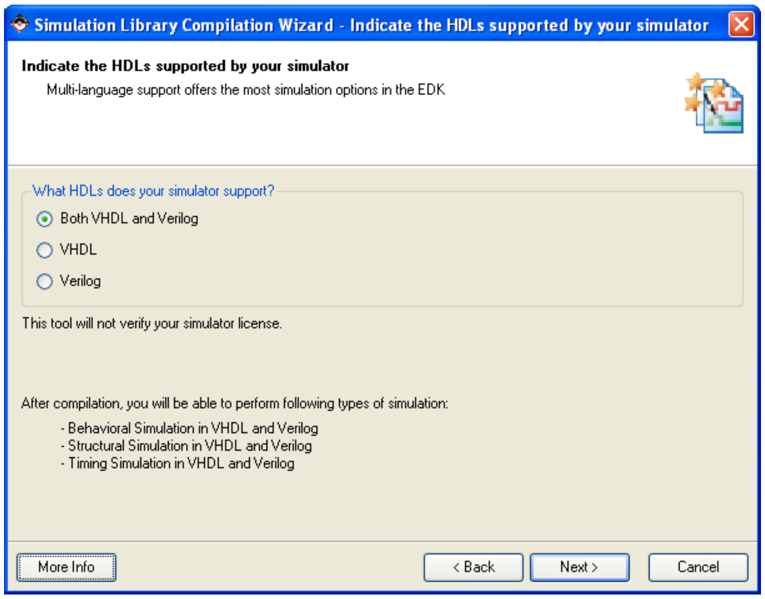
- The next screen is used to setup some options related to the EDK library as shown in Figure 8. Nothing needs to be changed in this screen. **Click Next**.

- The next screen of the wizard shown in Figure 9 is used for the compilation options for third party/user peripherals that need to be compiled with the library. We do not have such peripherals in our design so we can move to the next screen. **Click Next**.

- Now the compilation process will start as shown in Figure 10. The compilation process will take some time. When the compilation process complete the wizard window will be as shown in Figure 11. When the compilation process complete, the library files will be stored in the directory specified in the previous steps. The files can be used later for the simulation processes by selecting these directories with other designs. **Click Next** to continue.

- The next screen of the wizard shows some information about the simulation process in the EDK. **Click Next**.

- The wizard now will show the summary screen (Figure 13). This window shows the details of the compilation process. Now the simulation library building is complete and we are ready to simulate


Simulation Library Compilation Wizard - Indicate the HDLs supported by your simulator
More Info
< Back
Next >
Cancel

Indicate the HDLs supported by your simulator

Multi-language support offers the most simulation options in the EDK

What HDLs does your simulator support?

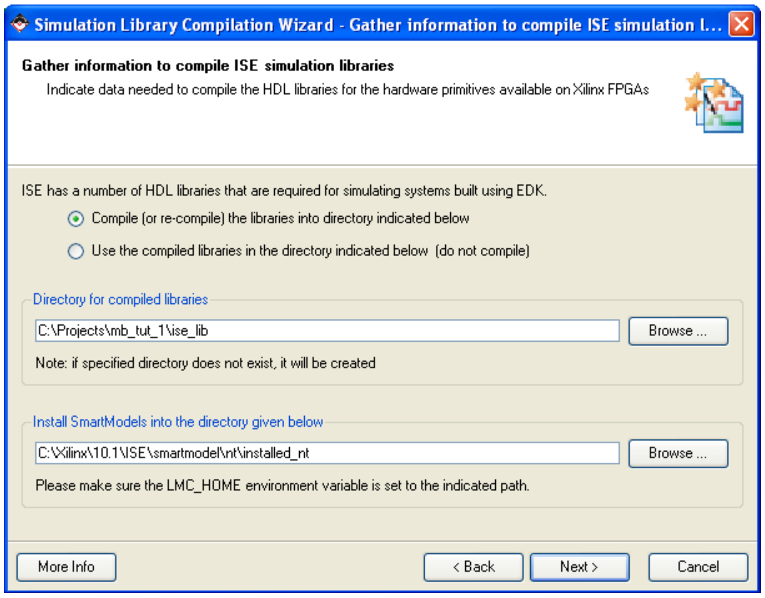
- [✓ Both VHDL and Verilog]
- VHDL
- Verilog

This tool will not verify your simulator license.

After compilation, you will be able to perform following types of simulation:

- Behavioral Simulation in VHDL and Verilog
- Structural Simulation in VHDL and Verilog
- Timing Simulation in VHDL and Verilog

Figure 5: Simulation Library Compilation Wizard - HDL Support


Simulation Library Compilation Wizard - Gather information to compile ISE simulation l...
Browse ...
Browse ...
More Info
< Back
Next >
Cancel

Gather information to compile ISE simulation libraries

Indicate data needed to compile the HDL libraries for the hardware primitives available on Xilinx FPGAs

ISE has a number of HDL libraries that are required for simulating systems built using EDK.

- [✓ Compile (or re-compile) the libraries into directory indicated below]
- Use the compiled libraries in the directory indicated below (do not compile)

Directory for compiled libraries

C:\Projects\mb_tut_1\ise_lib

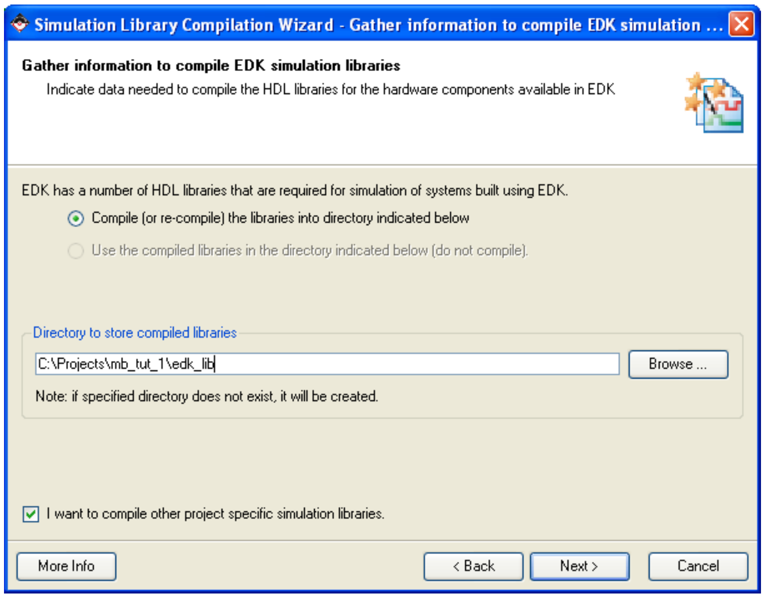
Note: if specified directory does not exist, it will be created

Install SmartModels into the directory given below

C:\Xilinx\10.1\ISE\smartmodel\nt\installed_nt

Please make sure the LMC_HOME environment variable is set to the indicated path.

Figure 6: Simulation Library Compilation Wizard - Information to compile ISE simulation models


Simulation Library Compilation Wizard - Gather information to compile EDK simulation ...

Gather information to compile EDK simulation libraries

Indicate data needed to compile the HDL libraries for the hardware components available in EDK

EDK has a number of HDL libraries that are required for simulation of systems built using EDK.

[✓ Compile (or re-compile) the libraries into directory indicated below]

Use the compiled libraries in the directory indicated below (do not compile).

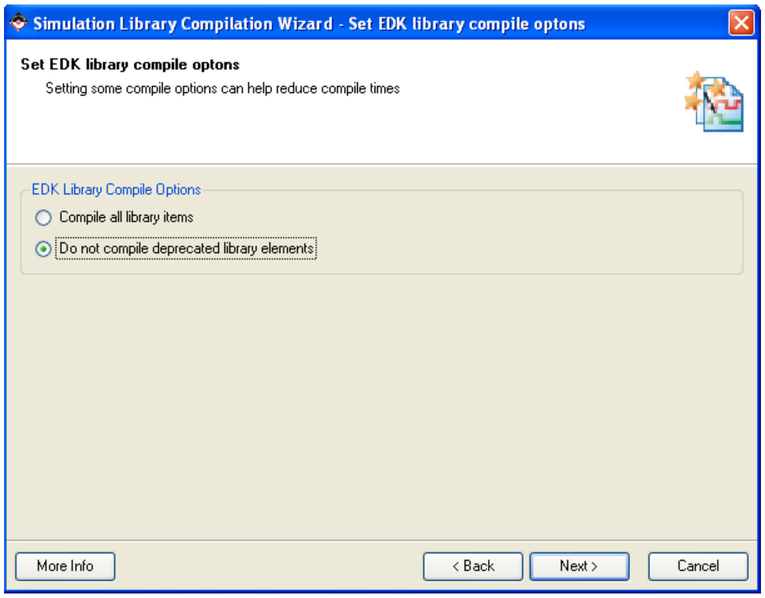
Directory to store compiled libraries

C:\Projects\mb_tut_1\edk_lib

Note: if specified directory does not exist, it will be created.

[✓ I want to compile other project specific simulation libraries.]

Figure 7: Simulation Library Compilation Wizard - Information to compile EDK simulation models


Simulation Library Compilation Wizard - Set EDK library compile optons

Set EDK library compile optons

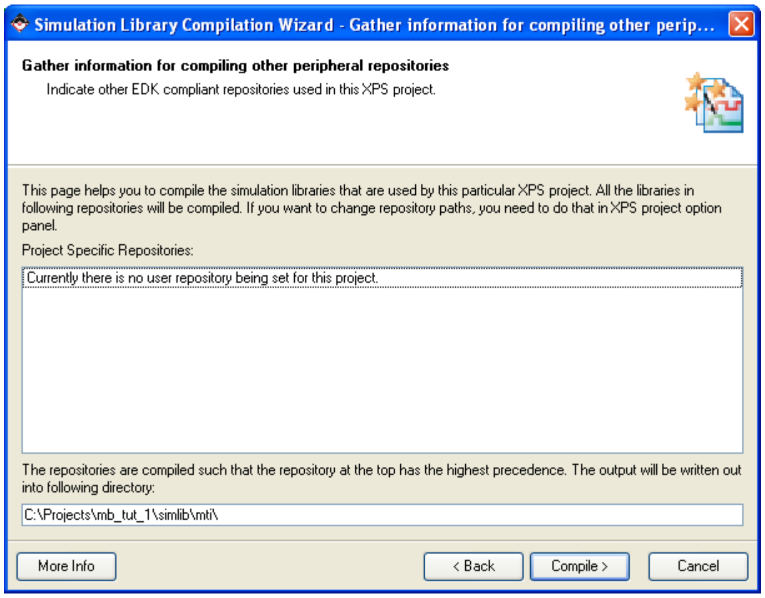
Setting some compile options can help reduce compile times

EDK Library Compile Options

Compile all library items

[✓ Do not compile deprecated library elements]

Figure 8: Simulation Library Compilation Wizard - EDK library options


Simulation Library Compilation Wizard - Gather information for compiling other perip...

Gather information for compiling other peripheral repositories

Indicate other EDK compliant repositories used in this XPS project.

This page helps you to compile the simulation libraries that are used by this particular XPS project. All the libraries in following repositories will be compiled. If you want to change repository paths, you need to do that in XPS project option panel.

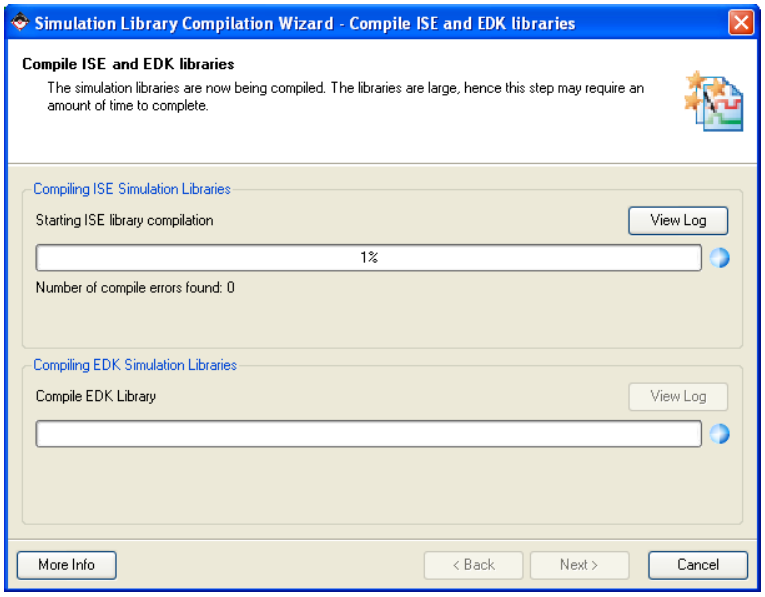
Project Specific Repositories:

Currently there is no user repository being set for this project.

The repositories are compiled such that the repository at the top has the highest precedence. The output will be written out into following directory:

C:\Projects\mb_tut_1\simlib\mti\

Figure 9: Simulation Library Compilation Wizard - Compiling other peripherals


Simulation Library Compilation Wizard - Compile ISE and EDK libraries

Compile ISE and EDK libraries

The simulation libraries are now being compiled. The libraries are large, hence this step may require an amount of time to complete.

Compiling ISE Simulation Libraries

Starting ISE library compilation

1%

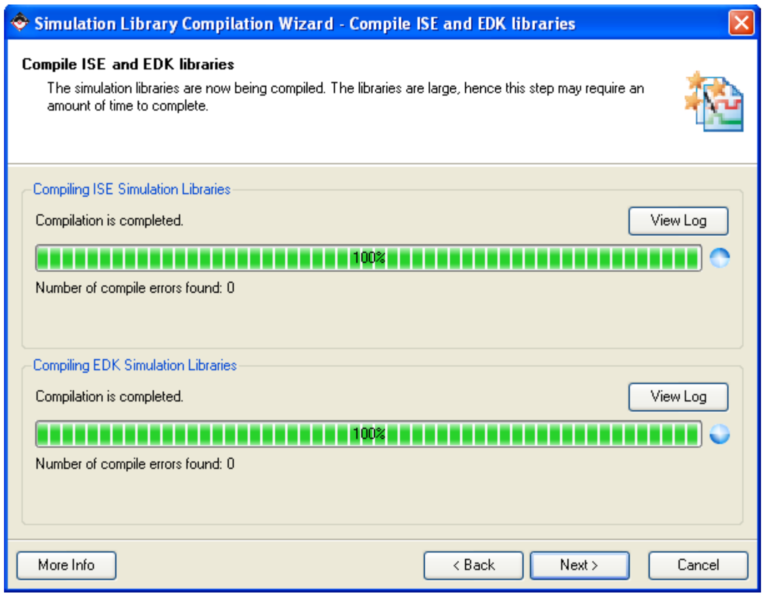
Number of compile errors found: 0

Compiling EDK Simulation Libraries

Compile EDK Library

View Log
View Log
More Info
< Back
Next >
Cancel

Figure 10: Simulation Library Compilation Wizard - Compiling ISE and EDK library (start)


Simulation Library Compilation Wizard - Compile ISE and EDK libraries

Compile ISE and EDK libraries

The simulation libraries are now being compiled. The libraries are large, hence this step may require an amount of time to complete.

Compiling ISE Simulation Libraries

Compilation is completed.

100%

Number of compile errors found: 0

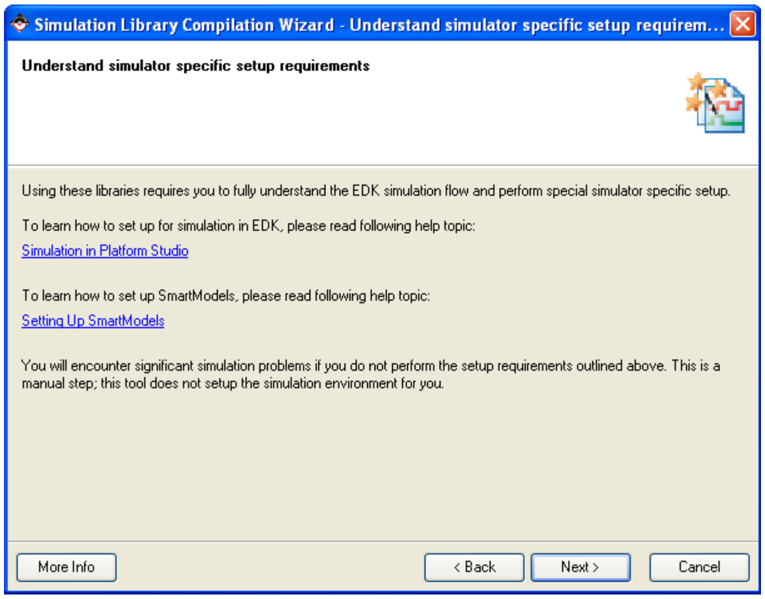
Compiling EDK Simulation Libraries

Compilation is completed.

100%

Number of compile errors found: 0

Figure 11: Simulation Library Compilation Wizard - Compiling ISE and EDK library (completed)


Simulation Library Compilation Wizard - Understand simulator specific setup requirem...

Understand simulator specific setup requirements

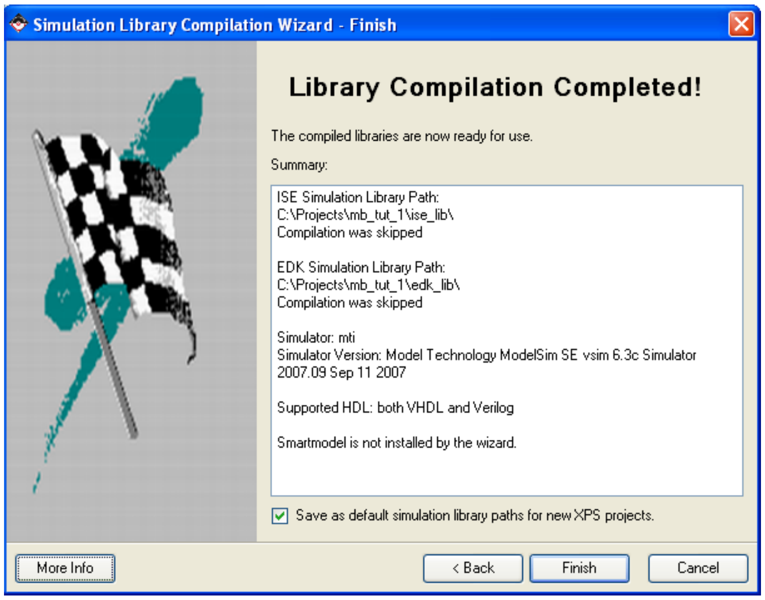
Using these libraries requires you to fully understand the EDK simulation flow and perform special simulator specific setup.

To learn how to set up for simulation in EDK, please read following help topic:
Simulation in Platform Studio

To learn how to set up SmartModels, please read following help topic:
Setting Up SmartModels

You will encounter significant simulation problems if you do not perform the setup requirements outlined above. This is a manual step; this tool does not setup the simulation environment for you.

Figure 12: Simulation Library Compilation Wizard - Simulation Requirements


Simulation Library Compilation Wizard - Finish
Library Compilation Completed!
More Info
< Back
Finish
Cancel

The compiled libraries are now ready for use.

Summary:

ISE Simulation Library Path:
C:\Projects\mb_tut_1\ise_lib\
Compilation was skipped

EDK Simulation Library Path:
C:\Projects\mb_tut_1\edk_lib\
Compilation was skipped

Simulator: mti
Simulator Version: Model Technology ModelSim SE vsim 6.3c Simulator 2007.09 Sep 11 2007

Supported HDL: both VHDL and Verilog

Smartmodel is not installed by the wizard.

☑ Save as default simulation library paths for new XPS projects.

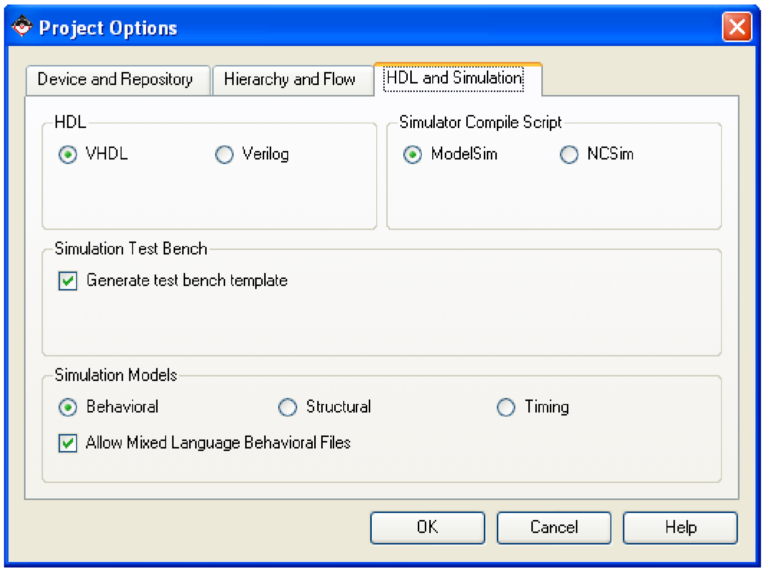
Figure 13: Simulation Library Compilation Wizard - Summary

the design.

3 Building the Behavioral Simulation Model

In the previous section we used the "Simulation Library Compilation Wizard" to build the simulation library required by ModelSim to simulate EDK designs. In this section we will setup EDK for the behavioral simulation for our MicroBlaze system and then build a simulation model for the system. Behavioral simulation is performed to verify the functionality of the system. the model being simulated does not include any information about the underlying technology and the timing information associated with it. Follow the following steps:

- In the EDK main window **Select Project** → **Project Options**. The project options dialog box will be displayed, select the **HDL and Simulation** tab as shown in Figure 14.
- In the **HDL and Simulation** tab do the following selections as shown in Figure 14:
  - In the **HDL** section select **VHDL**.
  - In the **Simulator Compile Script** section select **ModelSim**.
  - In the **Simulation Test Bench** section select **Generate test bench template**.
  - In the **Simulation Models** section select **Behavioral** and select **Allow Mixed Language Behavioral files**.
  - **Click Ok**.
- Now we are ready to build the simulation model of the system. Select **Simulation** → **Generate Simulation HDL Files** in the EDK main window. This will build the simulation files required to simulate your system. The simulation files will be stored under the directory *Simulation\Behavioral* in your project directory. Check the content of this directory. You will notice that several VHDL



HDL: [✓ VHDL] Verilog

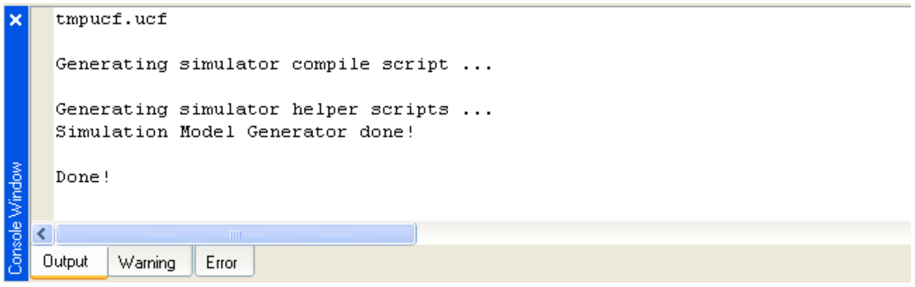
Simulator Compile Script: [✓ ModelSim] NCSim

Simulation Test Bench: [✓] Generate test bench template

Simulation Models: [✓ Behavioral] Structural Timing

[✓] Allow Mixed Language Behavioral Files

Figure 14: Change Simulation Setting for Behavioral Simulation



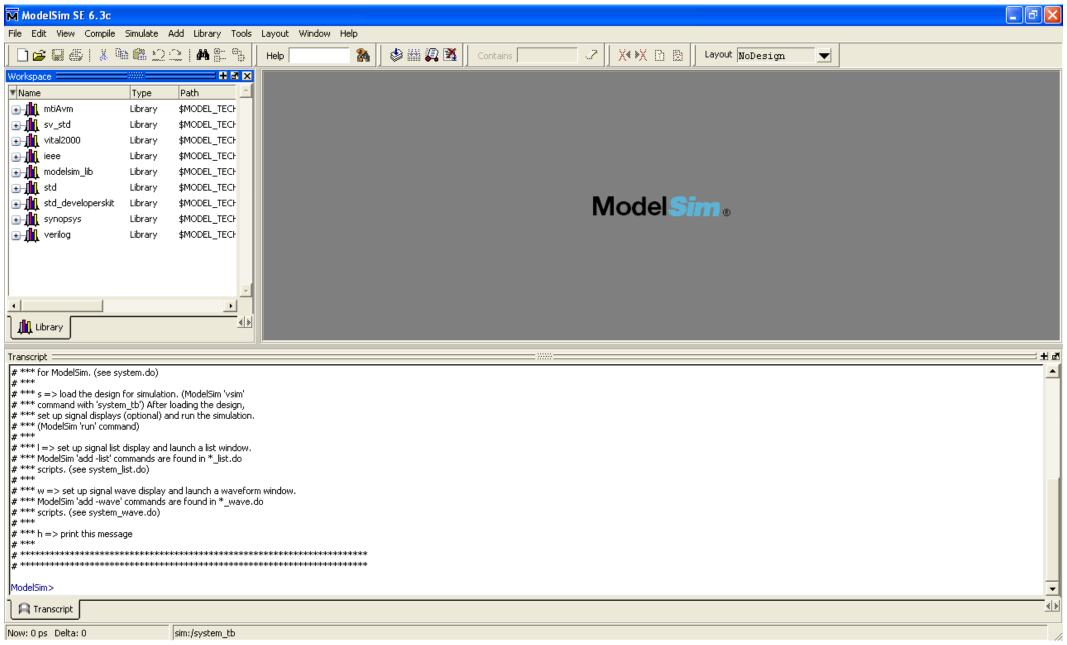
```
tmpucf.ucf

Generating simulator compile script ...

Generating simulator helper scripts ...
Simulation Model Generator done!

Done!
```

Figure 15: Building Simulation Model Completed

files are created for each component of the system. The file **system_tb.vhdl** is a test bench VHDL file create using EDK to control the IO signals of the system during the simulation.

Open the file **system_tb.vhdl** in the EDK window by selecting **File→Open** and then browse to the file location as stated above. The contains the required VHDL code required to instantiate the system inside ModelSim, and two VHDL processes that provides both the clock and reset signals for the system as shown in the following code:

```
——————————————————————————————————-
– system_tb.vhd
——————————————————————————————————-
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity system_tb is
end system_tb;


architecture STRUCTURE of system_tb is


    constant sys_clk_pin_PERIOD : time := 20000.000000 ps;
    constant sys_rst_pin_LENGTH : time := 320000 ps;


    component system is
        port (
            fpga_0_RS232_DCE_RX_pin : in std_logic;
            fpga_0_RS232_DCE_TX_pin : out std_logic;
            fpga_0_LEDs_8Bit_GPIO_d_out_pin : out std_logic_vector(0 to 7);
            fpga_0_DIP_Switches_4Bit_GPIO_in_pin : in std_logic_vector(0 to 3);
            fpga_0_DDR_SDRAM_DDR_DQS_Div_I_DDR_SDRAM_DDR_DQS_Div_O : inout std_logic;
            fpga_0_DDR_SDRAM_DDR_Clk_pin : out std_logic;
            fpga_0_DDR_SDRAM_DDR_Clk_n_pin : out std_logic;
            fpga_0_DDR_SDRAM_DDR_Addr_pin : out std_logic_vector(12 downto 0);
            fpga_0_DDR_SDRAM_DDR_BankAddr_pin : out std_logic_vector(1 downto 0);
            fpga_0_DDR_SDRAM_DDR_CAS_n_pin : out std_logic;
            fpga_0_DDR_SDRAM_DDR_CE_pin : out std_logic;
            fpga_0_DDR_SDRAM_DDR_CS_n_pin : out std_logic;
            fpga_0_DDR_SDRAM_DDR_RAS_n_pin : out std_logic;
            fpga_0_DDR_SDRAM_DDR_WE_n_pin : out std_logic;
            fpga_0_DDR_SDRAM_DDR_DM_pin : out std_logic_vector(1 downto 0);
            fpga_0_DDR_SDRAM_DDR_DQS : inout std_logic_vector(1 downto 0);
            fpga_0_DDR_SDRAM_DDR_DQ : inout std_logic_vector(15 downto 0);
            sys_clk_pin : in std_logic;
            sys_rst_pin : in std_logic
        );
    end component;

    – Internal signals

    signal fpga_0_DDR_SDRAM_DDR_Addr_pin : std_logic_vector(12 downto 0);
    signal fpga_0_DDR_SDRAM_DDR_BankAddr_pin : std_logic_vector(1 downto 0);
    signal fpga_0_DDR_SDRAM_DDR_CAS_n_pin : std_logic;
    signal fpga_0_DDR_SDRAM_DDR_CE_pin : std_logic;
    signal fpga_0_DDR_SDRAM_DDR_CS_n_pin : std_logic;
    signal fpga_0_DDR_SDRAM_DDR_Clk_n_pin : std_logic;
    signal fpga_0_DDR_SDRAM_DDR_Clk_pin : std_logic;
    signal fpga_0_DDR_SDRAM_DDR_DM_pin : std_logic_vector(1 downto 0);
    signal fpga_0_DDR_SDRAM_DDR_DQ : std_logic_vector(15 downto 0);
    signal fpga_0_DDR_SDRAM_DDR_DQS : std_logic_vector(1 downto 0);
    signal fpga_0_DDR_SDRAM_DDR_DQS_Div_I_DDR_SDRAM_DDR_DQS_Div_O : std_logic;
    signal fpga_0_DDR_SDRAM_DDR_RAS_n_pin : std_logic;
    signal fpga_0_DDR_SDRAM_DDR_WE_n_pin : std_logic;
    signal fpga_0_DIP_Switches_4Bit_GPIO_in_pin : std_logic_vector(0 to 3);
    signal fpga_0_LEDs_8Bit_GPIO_d_out_pin : std_logic_vector(0 to 7);
    signal fpga_0_RS232_DCE_RX_pin : std_logic;
    signal fpga_0_RS232_DCE_TX_pin : std_logic;
    signal sys_clk_pin : std_logic;
    signal sys_rst_pin : std_logic;

begin

    dut : system
        port map (
            fpga_0_RS232_DCE_RX_pin ⇒ fpga_0_RS232_DCE_RX_pin,
            fpga_0_RS232_DCE_TX_pin ⇒ fpga_0_RS232_DCE_TX_pin,
            fpga_0_LEDs_8Bit_GPIO_d_out_pin ⇒ fpga_0_LEDs_8Bit_GPIO_d_out_pin,
            fpga_0_DIP_Switches_4Bit_GPIO_in_pin ⇒ fpga_0_DIP_Switches_4Bit_GPIO_in_pin,
            fpga_0_DDR_SDRAM_DDR_DQS_Div_I_DDR_SDRAM_DDR_DQS_Div_O ⇒ fpga_0_DDR_SDRAM_DDR_DQS_Div_I_DDR_SDRAM
            fpga_0_DDR_SDRAM_DDR_Clk_pin ⇒ fpga_0_DDR_SDRAM_DDR_Clk_pin,
            fpga_0_DDR_SDRAM_DDR_Clk_n_pin ⇒ fpga_0_DDR_SDRAM_DDR_Clk_n_pin,
            fpga_0_DDR_SDRAM_DDR_Addr_pin ⇒ fpga_0_DDR_SDRAM_DDR_Addr_pin,
            fpga_0_DDR_SDRAM_DDR_BankAddr_pin ⇒ fpga_0_DDR_SDRAM_DDR_BankAddr_pin,
            fpga_0_DDR_SDRAM_DDR_CAS_n_pin ⇒ fpga_0_DDR_SDRAM_DDR_CAS_n_pin,
            fpga_0_DDR_SDRAM_DDR_CE_pin ⇒ fpga_0_DDR_SDRAM_DDR_CE_pin,
            fpga_0_DDR_SDRAM_DDR_CS_n_pin ⇒ fpga_0_DDR_SDRAM_DDR_CS_n_pin,
            fpga_0_DDR_SDRAM_DDR_RAS_n_pin ⇒ fpga_0_DDR_SDRAM_DDR_RAS_n_pin,
            fpga_0_DDR_SDRAM_DDR_WE_n_pin ⇒ fpga_0_DDR_SDRAM_DDR_WE_n_pin,
            fpga_0_DDR_SDRAM_DDR_DM_pin ⇒ fpga_0_DDR_SDRAM_DDR_DM_pin,
            fpga_0_DDR_SDRAM_DDR_DQS ⇒ fpga_0_DDR_SDRAM_DDR_DQS,
            fpga_0_DDR_SDRAM_DDR_DQ ⇒ fpga_0_DDR_SDRAM_DDR_DQ,
            sys_clk_pin ⇒ sys_clk_pin,
            sys_rst_pin ⇒ sys_rst_pin
        );

    – Clock generator for sys_clk_pin


    process
    begin
        sys_clk_pin ⇐ '0';
        loop
            wait for (sys_clk_pin_PERIOD/2);
            sys_clk_pin ⇐ not sys_clk_pin;
        end loop;
    end process;

    – Reset Generator for sys_rst_pin


    process
    begin
        sys_rst_pin ⇐ '1';
        wait for (sys_rst_pin_LENGTH);
        sys_rst_pin ⇐ not sys_rst_pin;
        wait;
    end process;

    – START USER CODE (Do not remove this line)

    – User: Put your stimulus here. Code in this
    – section will not be overwritten.
```



Name	Type	Path
mtiAvm	Library	$MODEL_TECH
sv_std	Library	$MODEL_TECH
vital2000	Library	$MODEL_TECH
ieee	Library	$MODEL_TECH
modelsim_lib	Library	$MODEL_TECH
std	Library	$MODEL_TECH
std_developerskit	Library	$MODEL_TECH
synopsys	Library	$MODEL_TECH
verilog	Library	$MODEL_TECH

```
# *** for ModelSim. (see system.do)
# ***
# *** s => load the design for simulation. (ModelSim 'vsim'
# *** command with 'system_tb') After loading the design,
# *** set up signal displays (optional) and run the simulation.
# *** (ModelSim 'run' command)
# ***
# *** l => set up signal list display and launch a list window.
# *** ModelSim 'add -list' commands are found in *_list.do
# *** scripts. (see system_list.do)
# ***
# *** w => set up signal wave display and launch a waveform window.
# *** ModelSim 'add -wave' commands are found in *_wave.do
# *** scripts. (see system_wave.do)
# ***
# *** h => print this message
# ***
# *****************************************************************
# *****************************************************************

ModelSim>
```

Now: 0 ps Delta: 0 sim:/system_tb

Figure 16: Modelsim Started

```
 – END USER CODE (Do not remove this line)

end architecture STRUCTURE;

configuration system_tb_conf of system_tb is
    for STRUCTURE
        for all : system
            use configuration work.system_conf;
        end for;
    end for;
end system_tb_conf;
```

This code can be modified to add customized simulation scenarios for the system. We do not need to modify the code as it generates all the signals required to run the system (clock and reset signals).

4 Behavioral Simulation

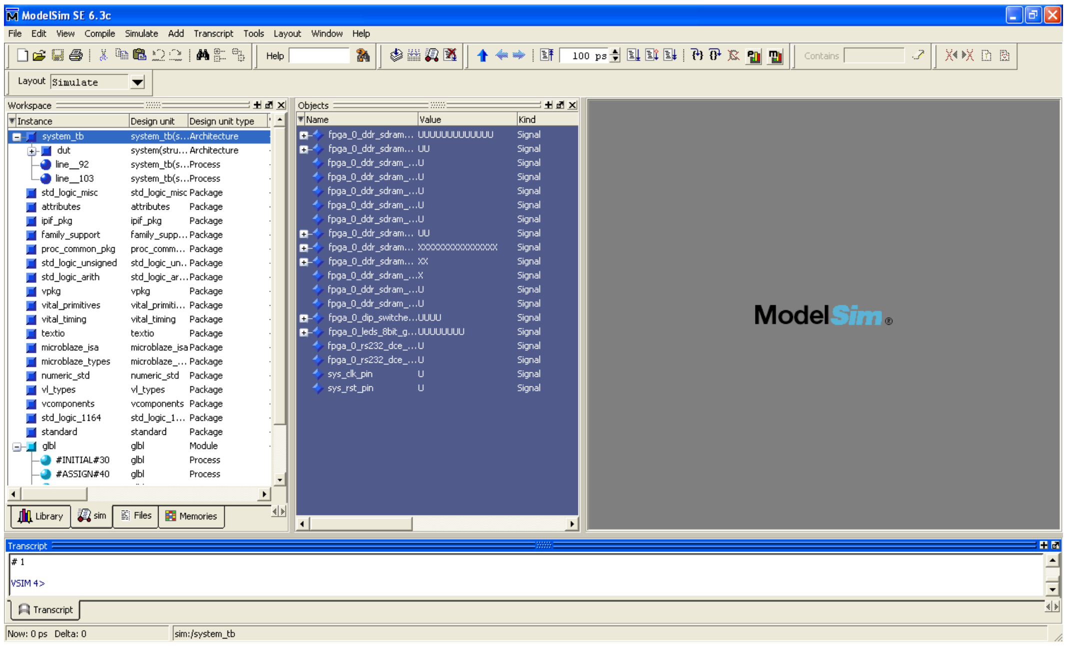
In the previous section we used EDK to generate a simulation model for our design. Now we are ready to use ModelSim to perform behavioral simulation. **Select Simulation→ Launch HDL Simulator**. This will start ModelSim as shown in Figure 16. The ModelSim main window contains several windows at start up.

The **transcript** window is used to enter ModelSim commands that are used to control the simulation. EDK simulator script define five commands to simulate EDK systems and a help window is displayed in the **transcript** window as shown below:

```
# *********************************************************************
# *********************************************************************
# ***
# *** Simulation Setup Macros (system_setup.do)
# ***
# *** c ⇒ compile the design by running the EDK compile script.
# *** Assumes ISE and EDK libraries were compiled earlier
# *** for ModelSim. (see system.do)
# ***
# *** s ⇒ load the design for simulation. (ModelSim 'vsim'
# *** command with 'system_tb') After loading the design,
# *** set up signal displays (optional) and run the simulation.
# *** (ModelSim 'run' command)
# ***
# *** l ⇒ set up signal list display and launch a list window.
# *** ModelSim 'add -list' commands are found in *_list.do
# *** scripts. (see system_list.do)
# ***
# *** w ⇒ set up signal wave display and launch a waveform window.
# *** ModelSim 'add -wave' commands are found in *_wave.do
# *** scripts. (see system_wave.do)
# ***
# *** h ⇒ print this message
# ***
# *********************************************************************
# *********************************************************************
```

- In the **transcript** window enter the command **"c"** at the command line. This will compile the design which will take sometime. Then enter the command **"s"** which loads the design in ModelSim. After loading the design, the different component of the design are shown in the **Workspace** window as shown in Figure 17 and the **Objects** windows show the internal signals of any selected component.
- Enter the command **"w"** to load the waveform that will display some signals defined by the simulation script generated by EDK. In behavioral simulation, the simulation scripts displays all input/output signals of each component in the system and the designer can add any other signal if required.
- Enter the command **run 1 ms**. This will start the simulation of the MicroBlaze system for 1 ms. After the simulation complete the waveform window will show the changes in all the selected signals of the system. **Clock the Plus button** at the top right corner of the waveform window to maximize the view. The results should look like Figure 19.

  You can go through different signals and watch activity on each signal. For example the signal **fpga_0_RS232_DCE_TX_pin** is the serial output of the system. The activities on this signal is corresponding the output of the software running on MicroBlaze as the standard output of the system is configured to use the RS232 interface (refer to the tutorial "MicroBlaze Tutorial on EDK 10.1 using Sparatan III E").



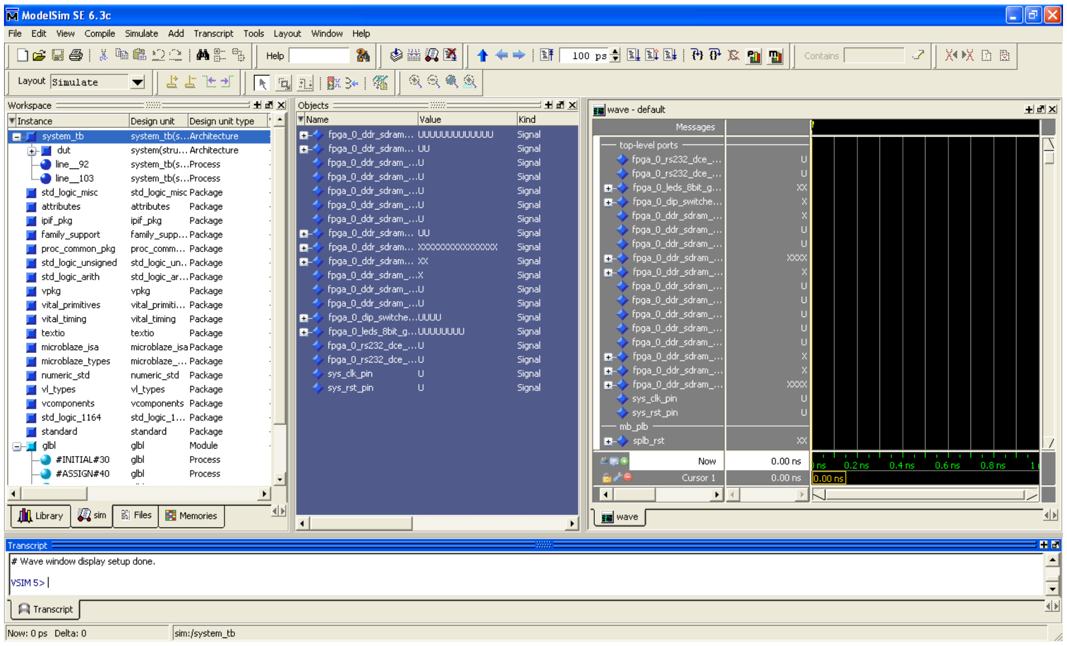
Instance	Design unit	Design unit type
system_tb	system_tb(s…	Architecture
dut	system(stru…	Architecture
line__92	system_tb(s…	Process
line__103	system_tb(s…	Process
std_logic_misc	std_logic_misc	Package
attributes	attributes	Package
ipif_pkg	ipif_pkg	Package
family_support	family_supp…	Package
proc_common_pkg	proc_comm…	Package
std_logic_unsigned	std_logic_un…	Package
std_logic_arith	std_logic_ar…	Package
vpkg	vpkg	Package
vital_primitives	vital_primiti…	Package
vital_timing	vital_timing	Package
textio	textio	Package
microblaze_isa	microblaze_isa	Package
microblaze_types	microblaze_…	Package
numeric_std	numeric_std	Package
vl_types	vl_types	Package
vcomponents	vcomponents	Package
std_logic_1164	std_logic_1…	Package
standard	standard	Package
glbl	glbl	Module
#INITIAL#30	glbl	Process
#ASSIGN#40	glbl	Process

Name	Value	Kind
fpga_0_ddr_sdram…	UUUUUUUUUUUUU	Signal
fpga_0_ddr_sdram…	UU	Signal
fpga_0_ddr_sdram_…	U	Signal
fpga_0_ddr_sdram_…	U	Signal
fpga_0_ddr_sdram_…	U	Signal
fpga_0_ddr_sdram_…	U	Signal
fpga_0_ddr_sdram_…	U	Signal
fpga_0_ddr_sdram…	UU	Signal
fpga_0_ddr_sdram…	XXXXXXXXXXXXXXXX	Signal
fpga_0_ddr_sdram…	XX	Signal
fpga_0_ddr_sdram_…	X	Signal
fpga_0_ddr_sdram_…	U	Signal
fpga_0_ddr_sdram_…	U	Signal
fpga_0_dip_switche…	UUUU	Signal
fpga_0_leds_8bit_g…	UUUUUUUU	Signal
fpga_0_rs232_dce_…	U	Signal
fpga_0_rs232_dce_…	U	Signal
sys_clk_pin	U	Signal
sys_rst_pin	U	Signal

```
# 1

VSIM 4>
```

```
Now: 0 ps   Delta: 0        sim:/system_tb
```

Figure 17: Modelsim - Design Compiled and Loaded



(Workspace and Objects panes as in Figure 17.)

Messages	
top-level ports	
fpga_0_rs232_dce_…	U
fpga_0_rs232_dce_…	U
fpga_0_leds_8bit_g…	XX
fpga_0_dip_switche…	X
fpga_0_ddr_sdram_…	X
fpga_0_ddr_sdram_…	U
fpga_0_ddr_sdram_…	U
fpga_0_ddr_sdram_…	XXXX
fpga_0_ddr_sdram_…	X
fpga_0_ddr_sdram_…	U
fpga_0_ddr_sdram_…	U
fpga_0_ddr_sdram_…	U
fpga_0_ddr_sdram_…	U
fpga_0_ddr_sdram_…	U
fpga_0_ddr_sdram_…	X
fpga_0_ddr_sdram_…	X
fpga_0_ddr_sdram_…	XXXX
sys_clk_pin	U
sys_rst_pin	U
mb_plb	
splb_rst	XX
Now	0.00 ns
Cursor 1	0.00 ns

```
# Wave window display setup done.

VSIM 5> |
```

```
Now: 0 ps   Delta: 0        sim:/system_tb
```

Figure 18: Modelsim - Loading the Waveform Window

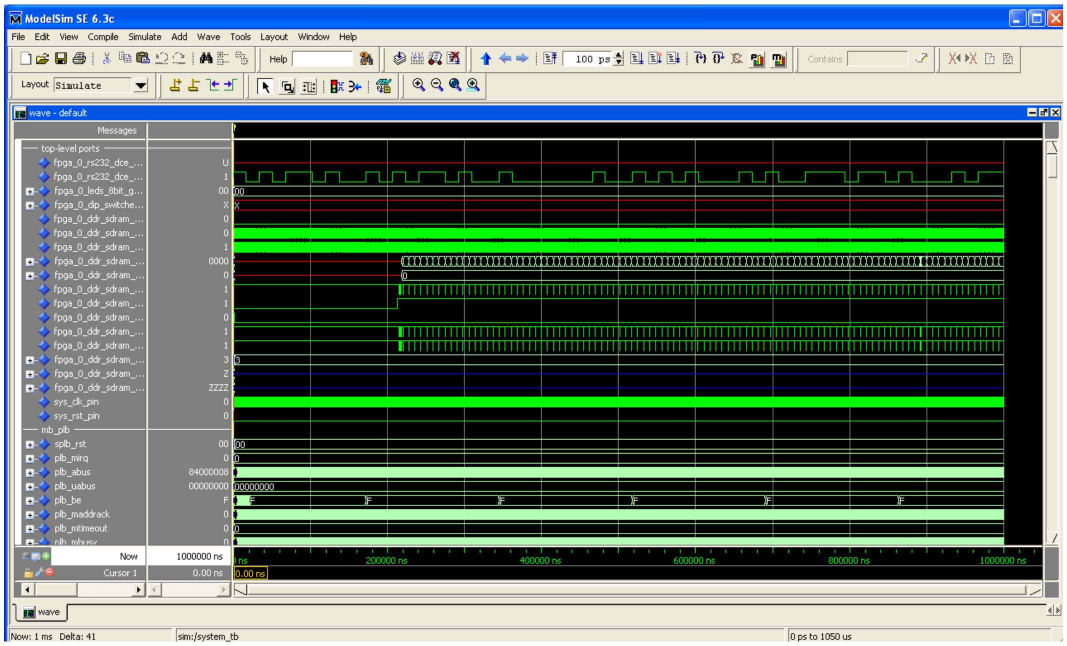


Figure 19: Modelsim - Simulation Output

5 Building the Timing Simulation Model

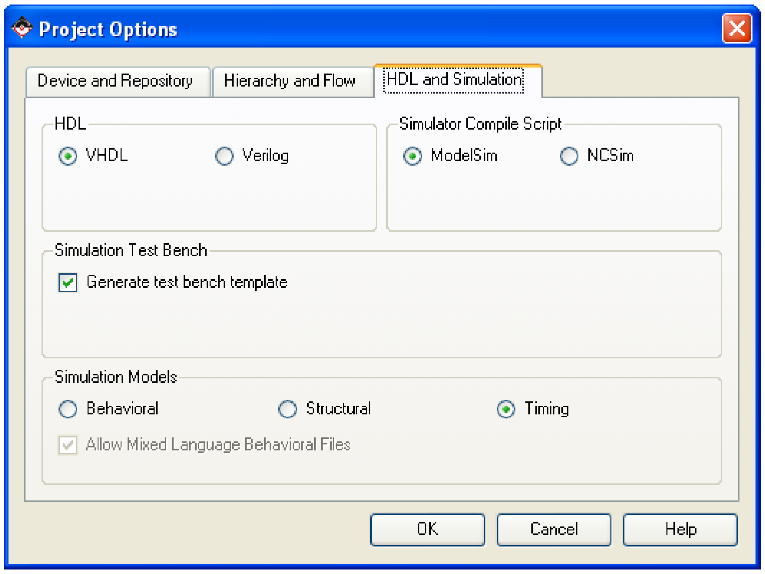
In the previous sections we performed behavioral simulation for a MicroBlaze system using ModelSim. We can use this simulation to verify the functionality of the system specially in the case of a custom hardware being implemented. In this section we are going to perform timing simulation which is performed using a simulation model generated from the post place and route net-list. This include all the timing information about the underlying technology (in our case it will include the timing information of Spartan III E FPGA).

- The first step is to change the simulation setting for timing simulation by **selecting Project→Project Options**. Select **HDL and Simulation** tab and in the **Simulation Models** section select **Timing** as shown in Figure 20 and then **Click Ok**.
- Build the timing simulation model by selecting **Simulation→ Generate Simulation HDL Files**. The process will take sometime and then the EDK console window will show a message to indicate the completion of the process as shown in Figure 21.

6 Timing Simulation

The simulation scripts generated for ModelSim define the same menu discussed earlier to control the simulation process. Execute the commands **"c"** and **"s"** to compile and load the design. When the design is loaded as shown in Figure 22 all the components in the simulation model are displayed in the **Workspace Window**. Compare the components tree of the timing simulation model to the behavioral simulation model (see Figure 17).

Execute the command **"w"** to load the waveform. The timing simulation model is extracted from the post place and route net-list. For this reason the information about the internal signals of each component are different and so are not included in the waveform (compare this to the behavioral model waveform).



HDL: [✓ VHDL] Verilog

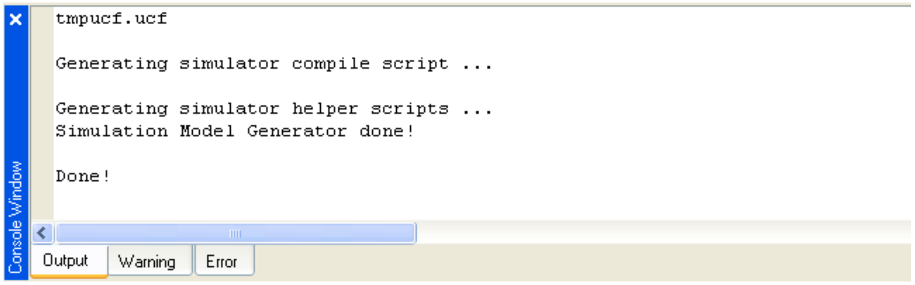
Simulator Compile Script: [✓ ModelSim] NCSim

Simulation Test Bench: [✓ Generate test bench template]

Simulation Models: Behavioral Structural [✓ Timing]

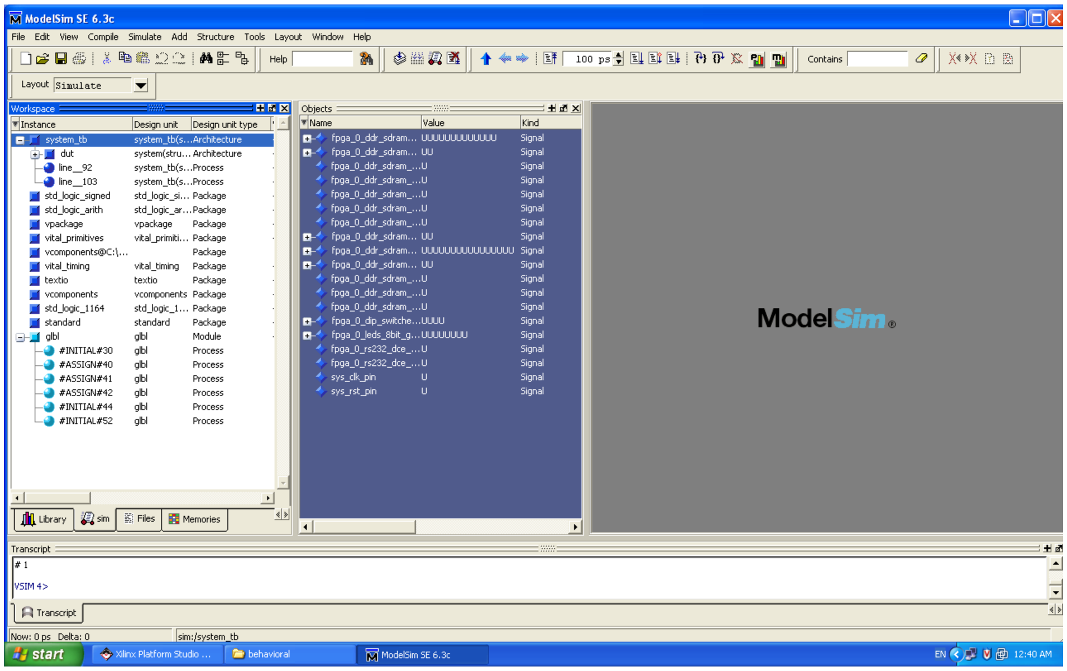
[✓ Allow Mixed Language Behavioral Files]

Figure 20: Change Simulation Setting for Timing Simulation



```
tmpucf.ucf

Generating simulator compile script ...

Generating simulator helper scripts ...
Simulation Model Generator done!

Done!
```

Figure 21: Building Simulation Model Completed



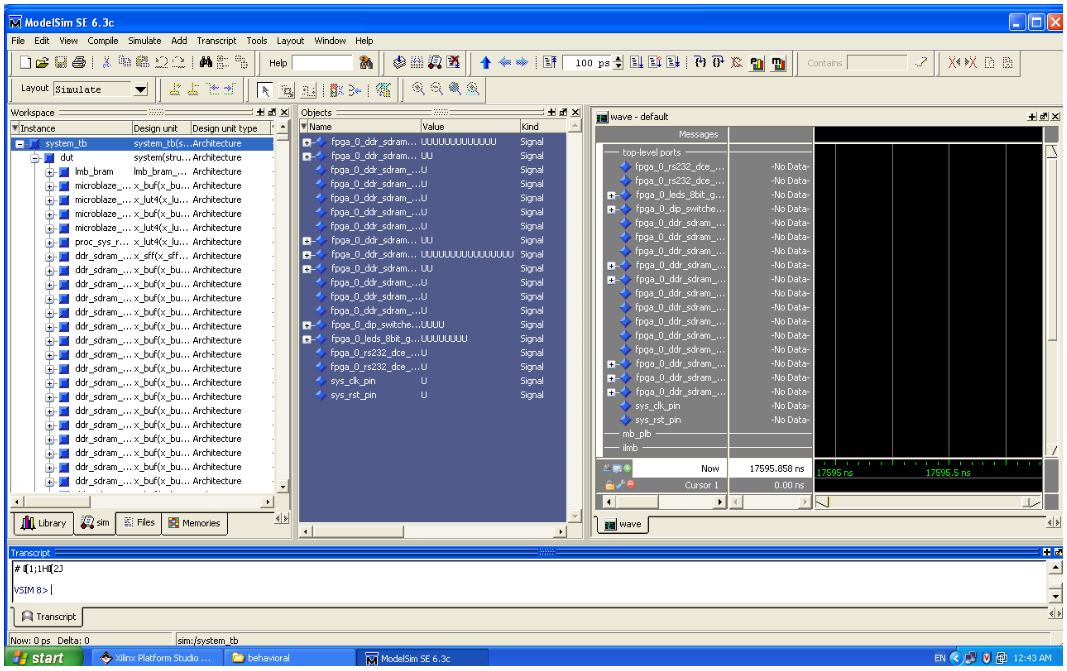
Instance	Design unit	Design unit type
system_tb	system_tb(s...	Architecture
dut	system(stru...	Architecture
line__92	system_tb(s...	Process
line__103	system_tb(s...	Process
std_logic_signed	std_logic_si...	Package
std_logic_arith	std_logic_ar...	Package
vpackage	vpackage	Package
vital_primitives	vital_primiti...	Package
vcomponents@C:\...		Package
vital_timing	vital_timing	Package
textio	textio	Package
vcomponents	vcomponents	Package
std_logic_1164	std_logic_1...	Package
standard	standard	Package
glbl	glbl	Module
#INITIAL#30	glbl	Process
#ASSIGN#40	glbl	Process
#ASSIGN#41	glbl	Process
#ASSIGN#42	glbl	Process
#INITIAL#44	glbl	Process
#INITIAL#52	glbl	Process

Name	Value	Kind
fpga_0_ddr_sdram...	UUUUUUUUUUUUU	Signal
fpga_0_ddr_sdram...	UU	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram...	UU	Signal
fpga_0_ddr_sdram...	UUUUUUUUUUUUUUUU	Signal
fpga_0_ddr_sdram...	UU	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_dip_switche...	UUUU	Signal
fpga_0_leds_8bit_g...	UUUUUUUU	Signal
fpga_0_rs232_dce_...	U	Signal
fpga_0_rs232_dce_...	U	Signal
sys_clk_pin	U	Signal
sys_rst_pin	U	Signal

```
# 1

VSIM 4>
```

Now: 0 ps Delta: 0 | sim:/system_tb

Figure 22: Modelsim - Design Compiled and Loaded



Instance	Design unit	Design unit type
system_tb	system_tb(s...	Architecture
dut	system(stru...	Architecture
lmb_bram	lmb_bram_...	Architecture
microblaze_...	x_buf(x_bu...	Architecture
microblaze_...	x_lut4(x_lu...	Architecture
microblaze_...	x_buf(x_bu...	Architecture
microblaze_...	x_lut4(x_lu...	Architecture
proc_sys_r...	x_lut4(x_lu...	Architecture
ddr_sdram_...	x_sff(x_sff...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture
ddr_sdram_...	x_buf(x_bu...	Architecture

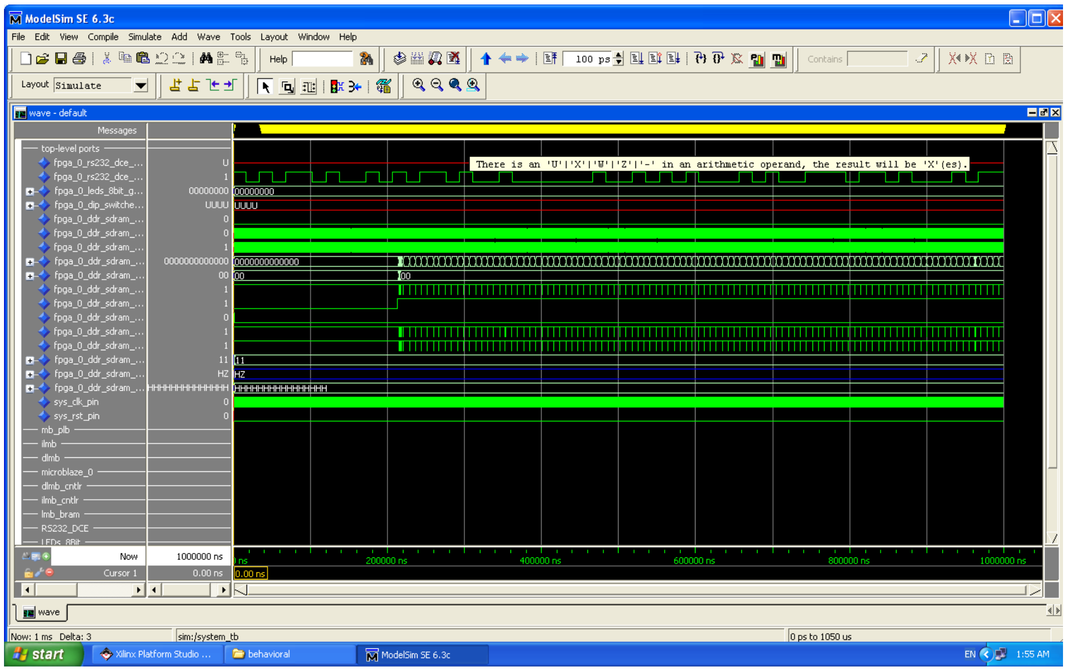
Name	Value	Kind
fpga_0_ddr_sdram...	UUUUUUUUUUUUU	Signal
fpga_0_ddr_sdram...	UU	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram...	UU	Signal
fpga_0_ddr_sdram...	UUUUUUUUUUUUUUUU	Signal
fpga_0_ddr_sdram...	UU	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_ddr_sdram_...	U	Signal
fpga_0_dip_switche...	UUUU	Signal
fpga_0_leds_8bit_g...	UUUUUUUU	Signal
fpga_0_rs232_dce_...	U	Signal
fpga_0_rs232_dce_...	U	Signal
sys_clk_pin	U	Signal
sys_rst_pin	U	Signal

Messages	
top-level ports	
fpga_0_rs232_dce_...	-No Data-
fpga_0_rs232_dce_...	-No Data-
fpga_0_leds_8bit_g...	-No Data-
fpga_0_dip_switche...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
fpga_0_ddr_sdram_...	-No Data-
sys_clk_pin	-No Data-
sys_rst_pin	-No Data-
mb_plb	
ilmb	
Now	17595.858 ns
Cursor 1	0.00 ns

```
# [1;1H[2J

VSIM 8>
```

Now: 0 ps Delta: 0 | sim:/system_tb

Figure 23: Modelsim - Loading the Waveform Window



```
There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
```

Figure 24: Modelsim - Simulation Output

Start the simulation by executing the command **run 1 ms**. The timing simulation will start. Notice that the timing simulation will take much longer time compared to the behavioral simulation. That is because the timing simulation model contains more information about the internal structure of the system and technology being used to implement the system (Spartan III E FPGA). The resulting waveform should look like Figure 24.

Compare the resulting waveform of the timing simulation to waveform of the behavioral simulation (Figure 19). Notice the timing differences in each input/output signals of the system. The results of the timing simulation is the closest to the actual implementation on board. If simulation results show something wrong in the system performance, this will be the same when the system is implemented on the board.

7 Summary

In this tutorial we performed both behavioral and timing simulation for a MicroBlaze system. We used ModelSim 6.3c SE to perform both types of simulation after creating a simulation library using EDK. Behavioral simulation allows the designer to verify the functionality of the system while timing simulation allows the designer to investigate timing related issues and verify the functionality of the system on a specific technology.