

FPGA simulation: ModelSim /Quest

1. Program configuration

Any changes to the simulation system parameters are made in **the Tools -> Edit Preferences menu** ... The setting results should be saved in the system startup script (*modelsim.tcl*). Saving settings:

command line: write preferences <filename>

To make the settings visible every time you start the system, set the system variable **MODELSIM_TCL=< path >\ modelsim.tcl**

< path > - path to the modelsim.tcl file

The MODELSIM system variable points to the default environment configuration file, *modelsim.ini*. The value of this variable is particularly important when installing different versions of the simulator in one system.

2. Project management

Creating a new project: File -> New -> Project...

Create Project ×
Project Name
84210
Project Location
D:/CADHDL/84210 Browse
Default Library Name
Copy Settings From
ltech64_10.7/modelsim.ini Browse
Copy Library Mappings C Reference Library Mappings
OK Cancel

When a new project is created, a working library is created with the name given in the window above (in this case **work**). To create a new library or change the mapping (alias) to an existing library, use the commands described in point 4 of this manual or in [1].

[!] Exercise: create a new project called <index_no> in the working directory on the disk indicated by the teacher (if the directory does not exist, create it); leave the default name for the working library;

Once you have created a design space, you can add existing HDL models to it or create new design elements. For this purpose, you can use the wizard that starts after creating the project (window below) or create a new source using the command **Project -> Add to Project -> New File...**

	$\overline{\mathbf{M}}$ Add items to the Project $\qquad \qquad \qquad$
	Click on the icon to add items of that type:
	Create New File Add Existing File
	Create Simulation Create New Folder
	Close
M Crea	te Project File X
File N pusty	ame Browse
Add f	Folder Top Level
	OK Cancel

[!] Exercise: create a new design source called 'empty'; source type 'VHDL'; leave the default file location;

The effect of each command that modifies the structure and state of the project is visible in the workspace:

Project - D:/CADHDL/8421	10/84210 :				
▼ Name	Status	Туре	Order	Modified	
pusty.vhd	?	VHDL	0	09/30/2019 08:19:52	
	•				
4					
👖 Library 🛛 🛗 Project 🗧	C				

At the project creation stage, there are two tabs in the workspace: 'Project' for source files and their status, and 'Library' containing data on libraries seen from the project level. At the current stage of the project, an empty ' work ' library should be visible in the 'Library' tab.

👖 Library 🚃		
▼ Name	Туре	Path
work (empty)	Library	D:/CADHDL/84210/work
ahblite_axi_bridge_v1_00_a	Library	C:\Xilinx\14.6\ISE_DS\ISE\mti_se\10.7\nt64/edk/ahblite_axi_bridg
	Library	C:\Xilinx\14.6\ISE_DS\ISE\mti_se\10.7\nt64/edk/apu_fpu_v3_10_
apu_fpu_virtex5_v1_01_a	Library	C:\Xilinx\14.6\ISE_DS\ISE\mti_se\10.7\nt64/edk/apu_fpu_virtex5
axi2axi_connector_v1_00_a	Library	C: \Xilinx\14.6\ISE_DS\ISE\mti_se\10.7\nt64/edk/axi2axi_connect
	Library	C:\Xilinx\14.6\ISE_DS\ISE\mti_se\10.7\nt64/edk/axi_ahblite_bridg
4		
👖 Library 🗙 🔛 Project 🙁		

In addition to creating new models, you can add existing HDL files to your project to reuse design units once defined. Existing files can be added using **the Project -> Add to Project -> Existing File... command** or from the context menu of the 'Project' window.

M Add file to Project	×
File Name <path file="" to="" vhd=""></path>	Browse
Add file as type	Folder Top Level
C Reference from current location	Copy to project directory OK Cancel

[!] Exercise: download the 'gates.vhd ' file from the course server and save it in any temporary directory; add this file to the project with the ' Copy to project directory '; source type 'VHDL'; (If the file was saved in the project directory, it should be added to the project with the 'Reference from current location '.)

The added file may contain one or more design unit models. To make them available in the working library, the file must be compiled using the **Compile -> Compile Selected** (menu or right mouse button).

me	Status	Туре	Order	Modified	
gates.v'	Edit Execute	L	1 0	09/30/2019 06:1: 09/30/2019 08:19	
	Compile		Compile 9	Selected	
	Add to Project		Compile /	All	
	Remove from Proje	ct	Compile (Out-of-Date	
	Close Project		Compile (Order	
	Update		Compile F	Report	
	Properties		Compile S	Summary	
	Project Settings		Compile F	Properties	

The effect of the compilation command is visible in the transcription window: # Compile of gates.vhd was successful.

Detailed information about the operation of the command (syntax, error numbers, etc.) can be obtained by double-clicking on the transcription window message. The circled line in the window below shows the syntax of the command invoked by the **Compile -> Compile Selected**.

MDL/84210/gates.vhd Successful Compile	×
vcom -work work -2002 -explicit -vopt -stats=none D:/CADHDL/84210/gates.vhd Model Technology ModelSim SE-64 vcom 10.7 Compiler 2017.12 Dec 7 2017 Loading package STANDARD Loading package TEXTIO	÷
Loading package std_logic_1164	
Compiling package gates_pkg	_
Compiling package body gates_pkg	
Loading package gates_pkg	
Compiling entity and2	
Compiling architecture dataflow of and2	
Compiling entity and3	
Compiling architecture dataflow of and3	
Compiling entity and4	
Compiling architecture dataflow of and4	
Compiling entity and5	
Compiling architecture dataflow of and5	
Comprising activities and a second se	

[!] Exercise: run compilation of 'gates.vhd ' file; observe the build results in the transcription window. Note the compilation command (vcom) syntax shown in the first line. Check the meaning of the **-work** parameter in the manual

After successful compilation, the working library shows the design units defined in the VHDL file being compiled.

👖 Library 🚃		
▼ Name	Туре	Path
- M work	Library	D:/CADHDL/84210/work
- E and2	Entity	D:/CADHDL/84210/gates.vhd
A dataflow	Architecture	
-E and3	Entity	D:/CADHDL/84210/gates.vhd
A dataflow	Architecture	
<u>+</u> -E and4	Entity	D:/CADHDL/84210/gates.vhd
<u> </u>	Entity	D:/CADHDL/84210/gates.vhd
— P gates_pkg	Package	D:/CADHDL/84210/gates.vhd
<u> <u> </u> </u>	Entity	D:/CADHDL/84210/gates.vhd
<u> </u>	Entity	D:/CADHDL/84210/gates.vhd
	Entity	D:/CADHDL/84210/gates.vhd
<u> </u>	Entity	D:/CADHDL/84210/gates.vhd
⊕-E nand5	Entity	D:/CADHDL/84210/gates.vhd
⊕-E nor2	Entity	D:/CADHDL/84210/gates.vhd
⊕-E nor3	Entity	D:/CADHDL/84210/gates.vhd
	Eatity	Du/CADHDL/94210/astos.vbd
•		
Library 🗙 🛗 Project 💥		

3. Simulation

The way to verify the operation of models described in hardware description languages is their simulation. The VHDL model can be simulated using forces defined in the test environment, the so-called testbench or using the ' force ' command of the simulator . the **Simulate -> Start Simulation ...** menu.

Design VHDL Verilog Libraries S		
▼ Name	Туре	Path
Ė-Ē and2	Entity	D:/CADHDL/84210/ -
A dataflow	Architecture	2
E and3	Entity	D:/CADHDL/84210/
E and4	Entity	D:/CADHDL/84210/
E and5	Entity	D:/CADHDL/84210/
P gates_pkg	Package	D:/CADHDL/84210/
🔁 🔁 inverter	Entity	D:/CADHDL/84210/
<u> </u>	Entity	D:/CADHDL/84210/
🔁 🔁 nand3	Entity	D:/CADHDL/84210/
⊞- <mark>E</mark>] nand4	Entity	D:/CADHDL/84210/
4		Þ
Design Unit(s)		Resolution
work.and2(dataflow)		default 💌
Optimization		
Enable optimization		Optimization Options

The simulation results can be observed in text form (transcription window) or graphically (time waveform window). To observe time waveforms, create a graphical debugging window using the **View -> Wave** command and add waveforms from the 'Objects' window to it. Select the desired signals in the 'Objects' window and then run the **Add -> To Wave -> Selected Signals** from the context menu.

12 Wave	_	
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>A</u> dd F <u>o</u> r	rmat <u>T</u> ools Boo <u>k</u> marks <u>W</u> indow <u>H</u> elp	
📰 Wave - Default 🚞		
📙 🖻 🕶 🕞 🔛 🍩 🕴 👗	- "■ 🛍 🗅 😂 🔕 - 🛤 🖺 🖕 🟙 📽 🖉 🛛 💾 🚣 "Ҽ 📲 🦉	⊾∓⊥∓ ⊒
🔄 🔁 - 👚 🦇 🕸 🗄	🕴 🚺 🔍 🧠 🕵 👔 👔 👔 👔 👔 👘 👘 🕸	à
🏤 🗸	Msgs	
📣 /and2/in1	U	_
4 /and2/in2	U	
/and2/out1		
		•
A R O Now	0.00 ns) ns 1 ns 2 ns 3 ns 4 ns 5 ns 6 ns 7 ns	8 ns
🔓 🌽 🤤 🛛 Cursor 1	0.00 ns 0.00 ns	
I →		
0 ps to 8523 ps	/and2/in1	

[!] Exercise: run simulation for design unit 'and2' from the work library ; display the ' Wave ' window and add all signals from the 'Objects' window to the window;

The 'force' command will be used to build forces in ModelSim . It allows you to define simple forces, but when simulating complex systems, issuing commands via a graphical interface is too time-consuming - Tcl scripts or force models defined in HDL are used for this purpose .

In the graphical interface of the simulator (context menu after selecting the appropriate time course), there are two commands for defining forces:

Force (defining single values)

M Force Selected Signal	×
Signal Name: sim:/and2/in1	-1
Value: 1	
Kind	
• Freeze O Drive O Deposit	
Delay For: 0	
Cancel After:	
OK Cancel	

Clock (defining extortion periodic)

M Define Clock ×
Clock Name
sim:/and2/in2
0 Duty 50
Period Cancel
Logic Values
High: 1 Low: 0
First Edge
OK Cancel

Please note that the above windows are dialog windows - you cannot see previously defined enforcements in them.

[!] Exercise: build forces for both inputs of gate 'and2'; use the 'force' command to build the sequence:

in1 -> '1' lasting 20ns, '0' lasting 20ns, '1' lasting 20ns, '0' lasting 20ns

in2 -> '1' lasting 40ns, '0' lasting 40ns

Run the simulation with the **run 80 ns** command entered in the transcription window; observe the simulation results in the 'Wave 'window;

🔷 /and2/in1	1								
\land 🔶 /and2/in2	1								
🔷 /and2/out1	1								
Now	80 ns	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	0 2	111111111 0 2	0 A	liiiiliii 0 5	n e	0 7	liiiiliiiilii 0 90
		, i i	0 2	U 3	0 4	0 0	0 0	0 /	0 00

4. Commands in the transcription window

The Tcl scripting language command interpreter is built into the ModelSim program interface . This makes it easier to automate design through Tcl scripts (macros have a standard extension of .do). Each operation described in the previous point can be performed by issuing a command in the transcription window.

Creating a new library

command line : vlib <library name>
by default, a new directory called libraries is created

Mapping a logical name to a library directory

command line: vmap <logical name> <library name>
specifying only the logical name displays the current mapping

Compilation of the selected source: Compile -> Compile Selected <u>command line</u>: vcom - work <library name> -2002 - explicit <file name> *default work library name = work*

Loading the simulation: Simulate -> Simulate ... <u>command line</u>: vsim <library name>.< entity name >(<architecture>)

Opening windows signals : View -> Objects line commands : view objects / view signals

Opening the Time Diagrams window: View -> Wave line commands: view wave / view -new wave

Adding the selected signal to the results window: Add -> Wave -> Selected Signals <u>command line</u>: add wave - label <label> <signal name>

Adding all signals to the results window: Add -> Wave -> Signals in Design <u>command line</u>: add wave -r /* (all issued commands and messages regarding their results appear in the transcription window; detailed description of commands in item [1])

[!] Exercise: using only the command line, perform the following operations:

- create library 'gates '
- compile the gates.vhd file into the 'gates 'library
- map the 'gates 'library to the logical name 'gates'

In the Library tab, display the contents of the 'gates' library, is it identical to the contents of the 'gates' library?

Using the force command

Simple device models can be tested using **force** commands issued from the transcription window or saved in a macro file (.do). When simulating complex devices, this design approach is ineffective - a more advanced test environment (testbench) should be used.

```
force < signal_name > <v1> - cancel < tc >
example:force clk 1 - cancel 100ns
```

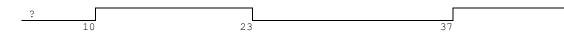
clk signal at the current hierarchy level takes the value '1' when the command is run; the - cancel switch allows you to cancel the force after a set time



```
force < signal_name > <v1> <t1>, <v2> <t2>
```

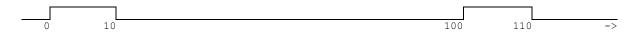
```
example:force / uut /reset 1 10ns, 0 23ns, 1 37ns
```

the reset signal at the / uut level of the hierarchy takes the value '1' at 10ns from the time the command is run, then '0' at 23ns and '1' at 37ns (relative to the time the command is run)



force < signal_name > <v1> <t1>, <v2> <t2> - repeat < tp >
example: force clk2 1 Ons, 0 1Ons - r 10Ons

-repeat (-r) switch allows you to create periodic forces with infinite duration and repetition period tp ; the example shows the clk2 signal with a frequency of 10MHz and a duty cycle of 10%



force

force command without parameters allows you to control currently defined forces, e.g.:

```
force - drive / cntrx / ce 0 {@0 ps } , 1 {@700 ns}
force - drive / cntrx / rst 0 {@1002 us} , 1 {@1002125 ns} , 0 {@1004400 ns}
force - drive / cntrx / clk 0 {@0 ps } , 1 {@50 ns} - repeat {@100 ns}
```

5. Simulation automation

.do script files (macros) are used to automate the operation of the program They consist of Questa [1] commands and Tcl commands .

macro example for the testowy.vhd file :

```
# compile test model
    vcom - work work -2008 ./testowy.vhd
# load the model into the simulator
    vsim test.work
# display the signals and simulation results window
    view waves -title test_wave
    view signals
# adding all signals to the results window
    add waves *
# extortion construction
    force c 1 0, 0 {20 ns} -r 40ns
    force b 1 0, 0 40ns -r 80ns
    force a 1 0, 0 {80 ns} -r 160ns
# run the simulation
    run 350 ns
```

Download the test.vhd file from the course server and add it to the project, do not compile the file.

[!] Exercise: write a macro file that performs the following operations:

- creating a library with a logical name <student index number>
- compiling the test.vhd source into a library called <student index number>
- starting the simulator for the test model
- display of waveforms
- construction of forces and model simulation
- scaling of the results window for the entire simulation time
- use command line parameters to pass the filename and simulation time as macro command line arguments
- ! upload all sources needed to run the macro (ZIP) as well as a screenshot of the simulation results and macro listing (PDF) to the course server

[1] Siemens EDA, Questa® SIM Command Reference Manual Including Support for Questa Base, Software Version 2023.3, Document Revision 8.3, © 2023 Siemens.