ModelSim Simulation projects with IP units with and without Quartus Prime Lite

Contents

Introduction

Design simulations are able to include wide range of analyses that virtually test behavior of a product under various operating and environmental conditions.

As opposed to trial-and-error, a smart simulation process allows targeted implementation of design choices in various stages of the development cycle. Proper functional and timing simulation is important to ensure design functionality and success.

This drastically reduces the need for recurrent, time-consuming testing on expensive physical prototypes, and subsequently shortens the total development time.

ModelSim is a multi-language HDL simulation environment offered by Mentor Graphics. *It can be used* ***independently or with Intel Quartus Prime****, which can help create libraries and link the designs to ModelSim.* Using the ModelSim Intel FPGA Started Edition software simplifies the debugging with help of simulations.

Prerequisites

You are expected to:
- be familiar with basics of logic and digital design,
- be familiar with Verilog constructs,
- have Quartus Prime Lite installed.

Lab2_1

PART1 “Launching ModelSim from Quartus Prime”

Objectives

You will use this part to familiarize yourself with the following:

1 How to run ModelSim for simulation purposes from Quartus Prime.
2 How to use IP functions from IP Catalog of Quartus Prime with ModelSim.
3 Some helpful shortcuts for ModelSim.

The project overviews

This lab utilizes a simple digital logic design shown on Figure 1.

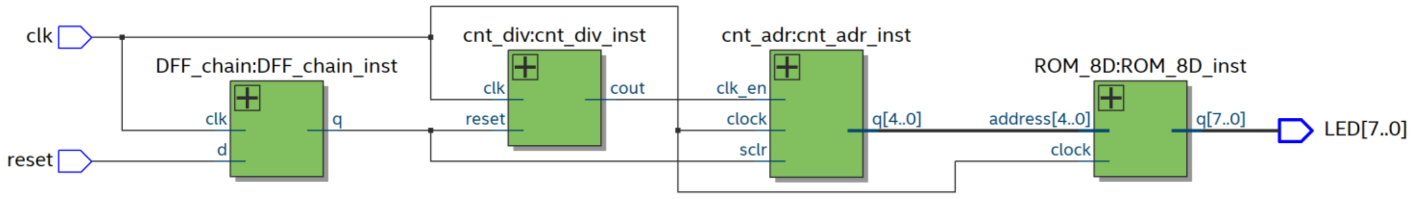


Figure 1

Inputs for the top-level module are:

- clk – clock signal (frequency is 25MHz for MiniDilab-CIV board).
- reset – reset signal coming from Push Button.

Outputs for the top-level module are:

- LED[7:0] – FPGA pins connected with Leds on a development board.

An algorithm of the project is:

- Input clock is divided by **cnt_div** unit. Division ratio is a parameter for the unit.
- Unit **cnt_adr** provides addresses for the **ROM_8D** unit.
- Unit **ROM_8D** keeps data and, in accordance of the current address, displays the word (8 bits) of data on the LED[7..0] outputs.
- Reset signal synchronously resets all counters having active value “1”.
- Unit **DFF_chain** synchronizes external reset signal. It contains two DFF flip-flops connected in a chain.

Create a project in Quartus Prime

1 Start **Quartus Prime Lite** development tool

2 Create a project:

a. Working directory: **C:/Intel_trn/Q_MS/lab2_1**

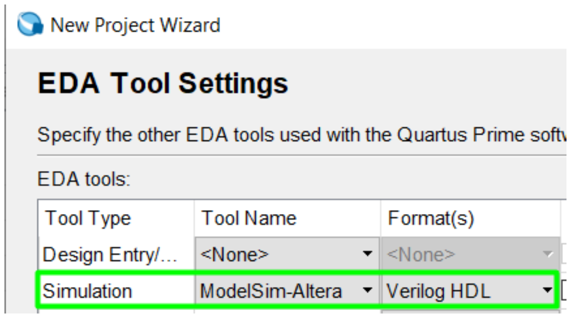
b. Project name: lab2_1

c. Top-Level design entity: lab2_1

d. Project Type: Empty project

e. Device: EP4CE6E22C8

f. EDA tools: ModelSim Altera; Verilog HDL (see Figure 2)



New Project Wizard

EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime softw

EDA tools:

Tool Type	Tool Name	Format(s)
Design Entry/...	<None>	<None>
Simulation	ModelSim-Altera	Verilog HDL

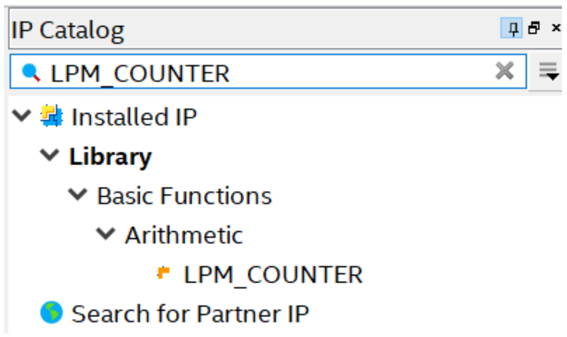
Figure 2

Add IP Components to the design

1 Create **cnt_adr** unit by using LPM_COUNTER.

Unit **cnt_adr** provides addresses for ROM_8D unit. You need to create **cnt_adr** unit during the Lab by using **LPM_COUNTER** from IP Library.

a. Find LPM_COUNTER function by typing LPM_COUNTER in the find field of IP catalog. If you do not have the IP Catalog already open, go to View menu => Utility Windows => IP Catalog to view the window.



IP Catalog

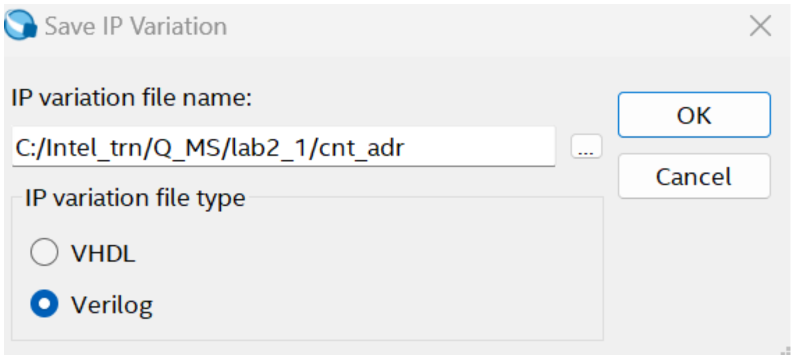
LPM_COUNTER

Installed IP
  Library
    Basic Functions
      Arithmetic
        LPM_COUNTER
Search for Partner IP

Figure 3

b. Double click LPM_COUNTER.

c. Name the file as **cnt_adr** (*Please make sure to name it as **cnt_adr***) in the **Save IP Variation** box that came up and click **OK**.

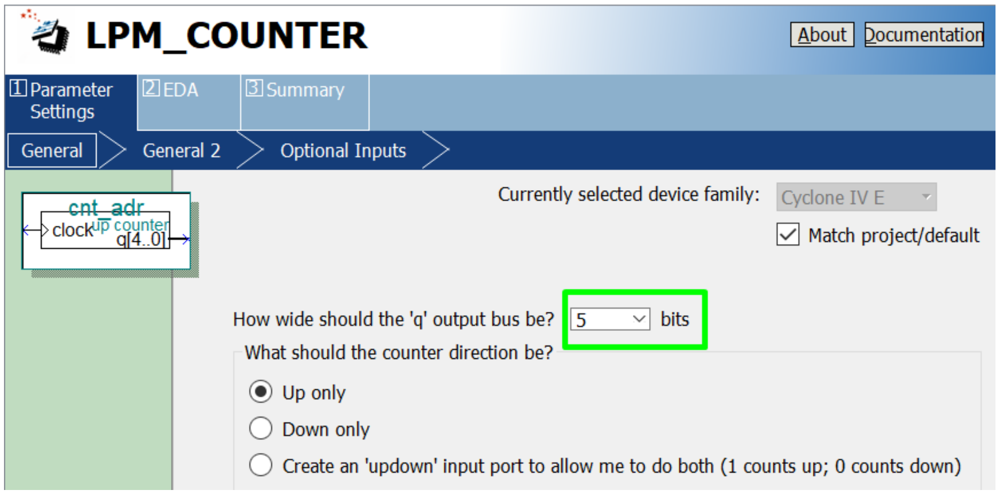

Save IP Variation

IP variation file name: C:/Intel_trn/Q_MS/lab2_1/cnt_adr

IP variation file type: VHDL [✓ Verilog]

Figure 4

d. In the MegaWizard, choose **5 bits** for bus "q" and leave remaining settings to default.


LPM_COUNTER

Currently selected device family: Cyclone IV E

✓ Match project/default

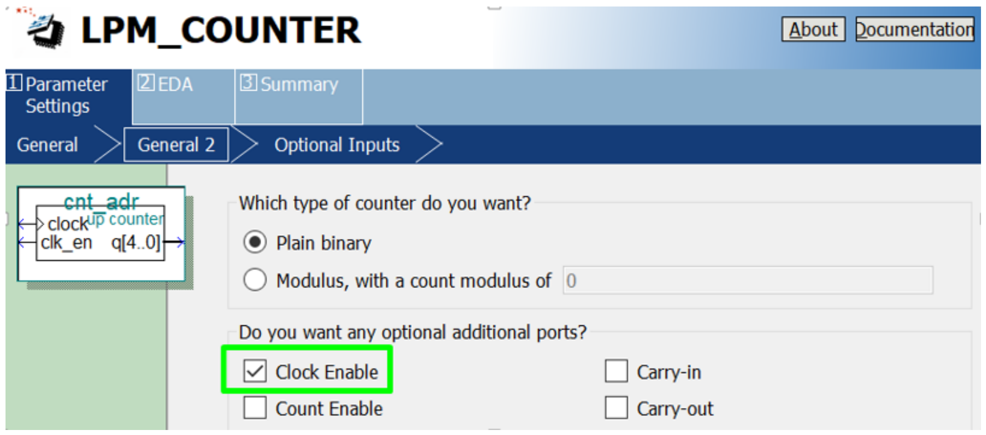
How wide should the 'q' output bus be? 5 bits

What should the counter direction be? [✓ Up only] Down only / Create an 'updown' input port to allow me to do both (1 counts up; 0 counts down)

Figure 5

Click **Next**.

e. In the window appeared select Clock enable check box and leave remaining settings to default.


LPM_COUNTER

Which type of counter do you want? [✓ Plain binary] Modulus, with a count modulus of 0

Do you want any optional additional ports?

✓ Clock Enable

Count Enable

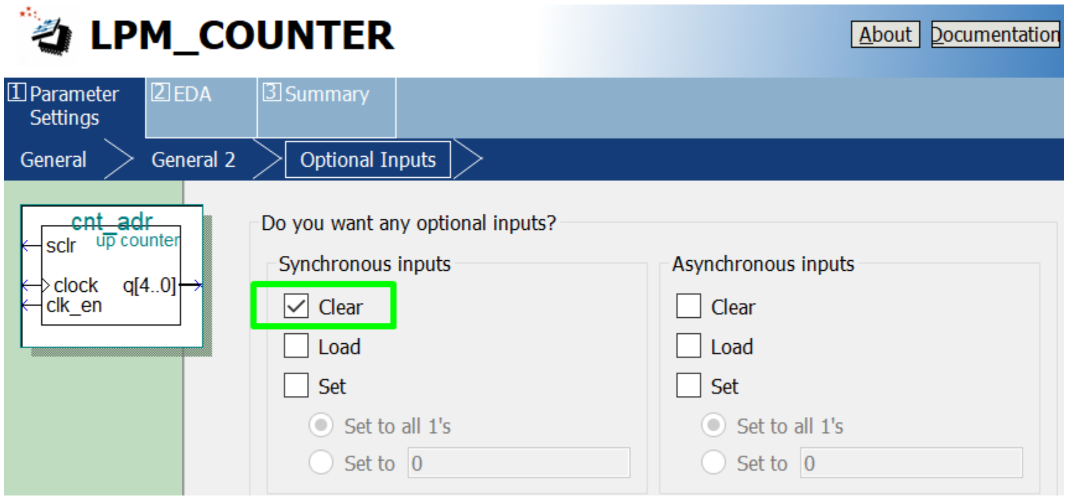
Carry-in

Carry-out

Figure 6

Click **Next**.

f. In the window appeared select synchronous **Clear** check box and leave remaining settings to default.



Do you want any optional inputs?

Synchronous inputs: [✓ Clear] Load Set; Set to all 1's / Set to 0

Asynchronous inputs: Clear Load Set; Set to all 1's / Set to 0

Figure 7

Click **Next**.

g. In the window appeared pay attention to the simulation library pointed. It is **lpm** library. When using Native link of Quartus Prime you have all libraries referenced correctly automatically. Leave all settings to default and click **Next**.

h. In the window appeared select **cnt_adr_inst.v** and **cnt_adr_bb.v** check boxes and in the working folder you will have templates for using the generated unit in a top level module. Leave the remaining settings to default and click **Finish**.

i. After you click **Finish** you will get a dialog box asking you if you want to add the IP files generated to the project. Click **Yes**. You will add cnt_adr.qip file to the project. The file contains information about generated unit, which is for Quartus Prime.

2 Create **ROM_8D** unit by using **ROM: 1-PORT**.

Unit **ROM_8D** keeps data that are stored in the file **C:\Intel_trn\Q_MS\lab2_1\ ROM_8D.hex**. You need to create **ROM_8D** unit during the Lab by using **ROM: 1-PORT** from IP Library.

a. Find **ROM: 1-PORT** function by typing **ROM: 1-PORT** in the find field of IP catalog. If you do not have the IP Catalog already open, go to View menu => Utility Windows => IP Catalog to view the window.

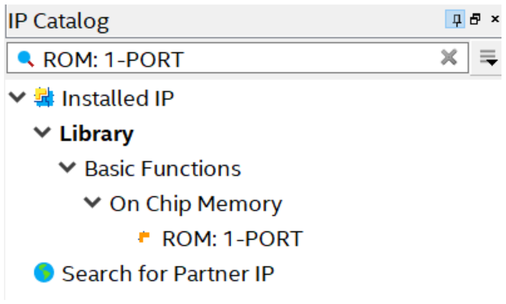
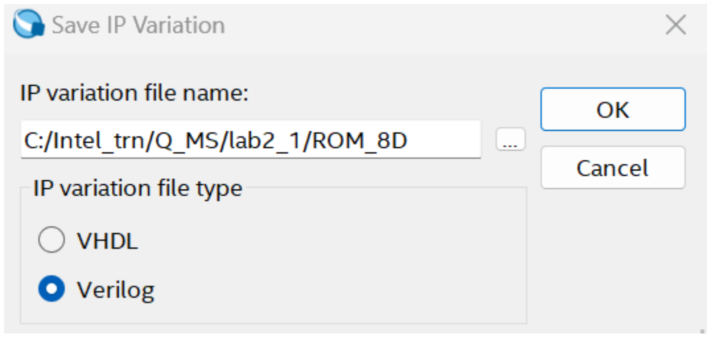


Figure 8

b. Double click ROM: 1-PORT.

c. Name the file as **ROM_8D** (*Please make sure to name it as **ROM_8D***) in the **Save IP Variation** box that came up and click **OK**.


Save IP Variation

IP variation file name:

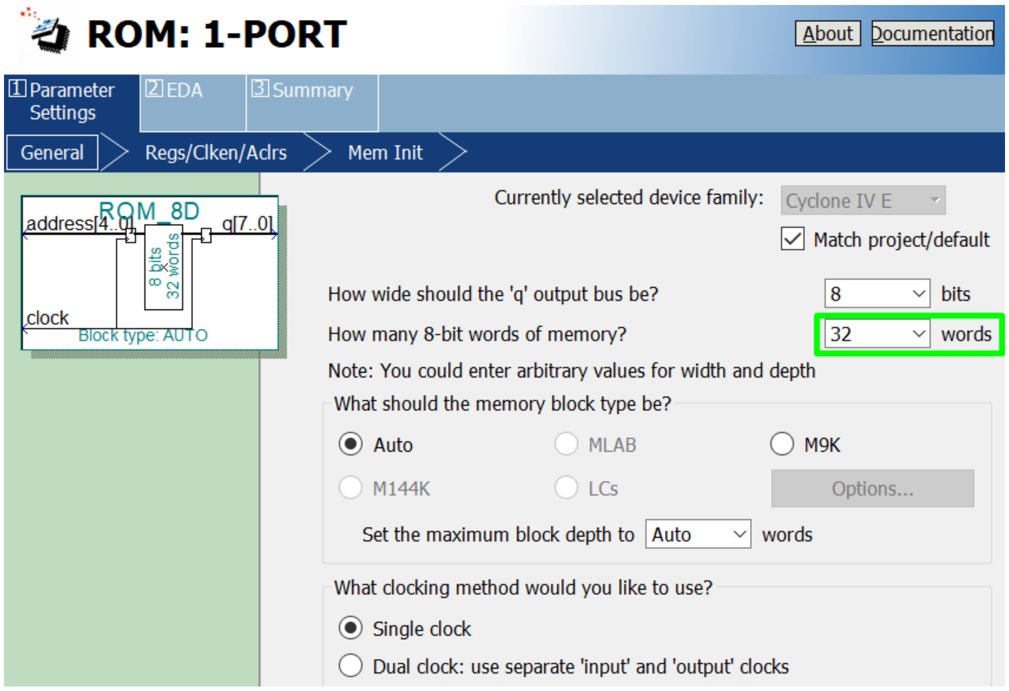
C:/Intel_trn/Q_MS/lab2_1/ROM_8D

IP variation file type

VHDL [✓ Verilog]

Figure 9

d. In the window appeared choose 32 words for the memory. Be sure that the bus ”**q**” is 8 bits wide. Leave the remaining settings to default.


ROM: 1-PORT

Currently selected device family: Cyclone IV E

[✓] Match project/default

How wide should the 'q' output bus be? 8 bits

How many 8-bit words of memory? 32 words

Note: You could enter arbitrary values for width and depth

What should the memory block type be? [✓ Auto] MLAB M9K M144K LCs

Set the maximum block depth to Auto words

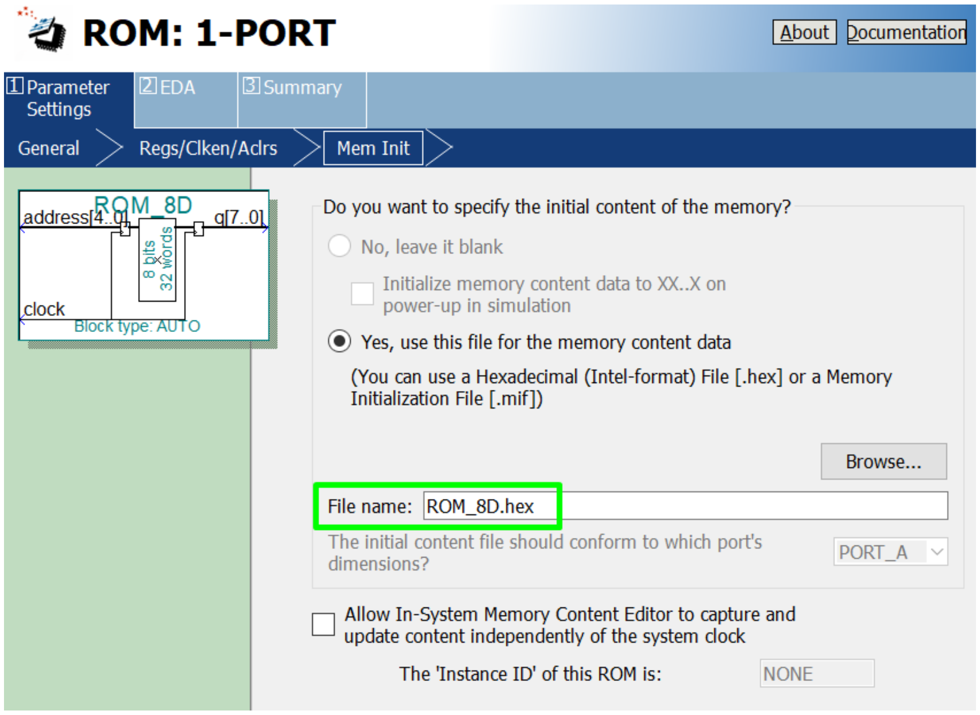
What clocking method would you like to use? [✓ Single clock] Dual clock: use separate 'input' and 'output' clocks

Figure 10

Click **Next**.

e. In the window appeared leave all settings to default. Click **Next**.

f. In the window appeared point name **ROM_8D.hex** as the name of the file with memory data. You can type the name directly to the field or use Browse button to select the file from working folder. Leave the remaining settings to default.



Do you want to specify the initial content of the memory?

No, leave it blank

Initialize memory content data to XX..X on power-up in simulation

[✓ Yes, use this file for the memory content data]

(You can use a Hexadecimal (Intel-format) File [.hex] or a Memory Initialization File [.mif])

File name: ROM_8D.hex

The initial content file should conform to which port's dimensions? PORT_A

Allow In-System Memory Content Editor to capture and update content independently of the system clock

The 'Instance ID' of this ROM is: NONE

Figure 11

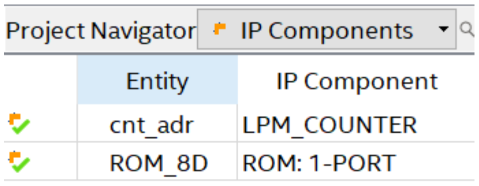
Click **Next**.

g. In the window appeared pay attention to the simulation library pointed. It is **altera_mf** library. When using Native link of Quartus Prime you have all libraries referenced correctly automatically. Leave all settings to default and click **Next**.

h. In the window appeared select **ROM_8D_inst.v** and **ROM_8D_bb.v** check boxes and in the working folder you will have templates for using the generated unit in a top level module. Leave the remaining settings to default and click **Finish**.

i. After you click **Finish** you will get a dialog box asking you if you want to add the IP files generated to the project. Click **Yes**.

3 To check if the IPs are added in the design: select **Project navigator** window => **IP Components** in the drop down menu.

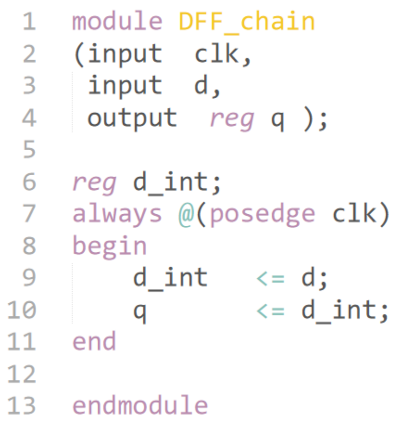


	Entity	IP Component
✓	cnt_adr	LPM_COUNTER
✓	ROM_8D	ROM: 1-PORT

Figure 12

Create units and a top-level design file

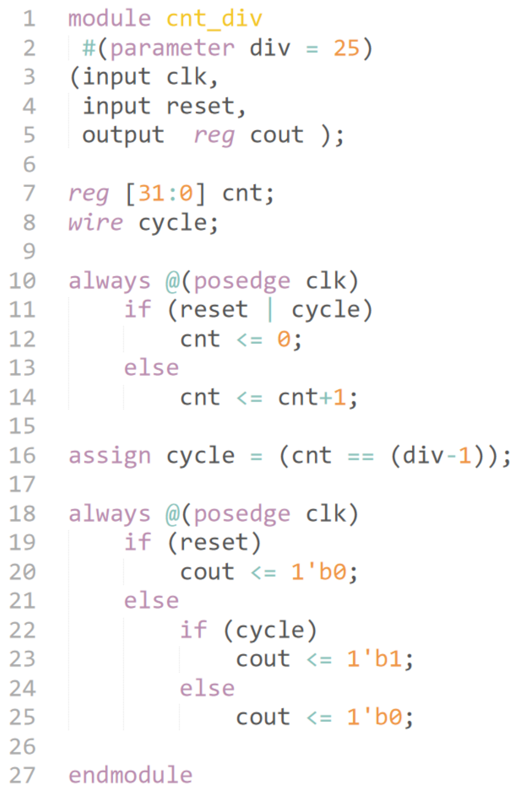
1 You need to create a source code for **DFF_chain** unit (file name: **DFF_chain.v)**. An example of a code is on Figure 13.



```verilog
module DFF_chain
(input  clk,
 input  d,
 output  reg q );

reg d_int;
always @(posedge clk)
begin
    d_int   <= d;
    q       <= d_int;
end

endmodule
```

Figure 13

2 You need to create a source code for **cnt_div** unit (file name: **DFF_chain.v)** . An example of a code is on Figure 14.

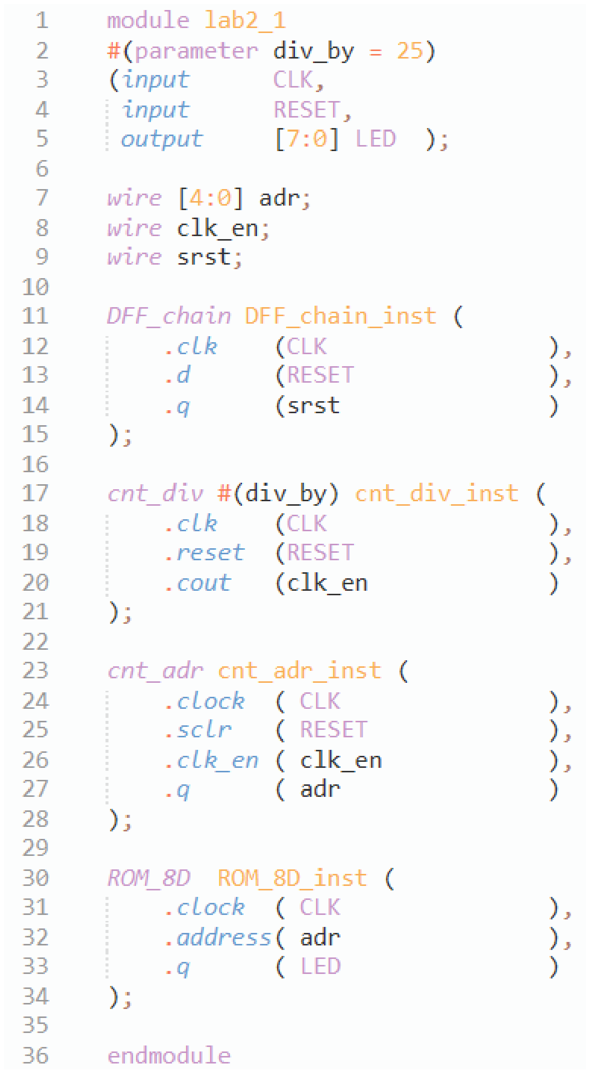


```verilog
module cnt_div
 #(parameter div = 25)
(input clk,
 input reset,
 output  reg cout );

reg [31:0] cnt;
wire cycle;

always @(posedge clk)
    if (reset | cycle)
        cnt <= 0;
    else
        cnt <= cnt+1;

assign cycle = (cnt == (div-1));

always @(posedge clk)
    if (reset)
        cout <= 1'b0;
    else
        if (cycle)
            cout <= 1'b1;
        else
            cout <= 1'b0;

endmodule
```

Figure 14

3 You need to create a source code for top-level unit **lab2_1** file (file: **lab2_1.v**).

Unit lab2_1 is the top level unit, highlighted on Figure 1. It connects all low-level units together. An example of a code is on Figure 15.



```
module lab2_1
#(parameter div_by = 25)
(input      CLK,
 input      RESET,
 output     [7:0] LED  );

wire [4:0] adr;
wire clk_en;
wire srst;

DFF_chain DFF_chain_inst (
    .clk    (CLK                ),
    .d      (RESET              ),
    .q      (srst               )
);

cnt_div #(div_by) cnt_div_inst (
    .clk    (CLK                ),
    .reset  (RESET              ),
    .cout   (clk_en             )
);

cnt_adr cnt_adr_inst (
    .clock  ( CLK               ),
    .sclr   ( RESET             ),
    .clk_en ( clk_en            ),
    .q      ( adr               )
);

ROM_8D  ROM_8D_inst (
    .clock  ( CLK               ),
    .address( adr               ),
    .q      ( LED               )
);

endmodule
```

Figure 15

4 To check if the project (design units description and connections) is correct:

a. Select **Project navigator** window => **Hierarchy** in the drop-down menu.

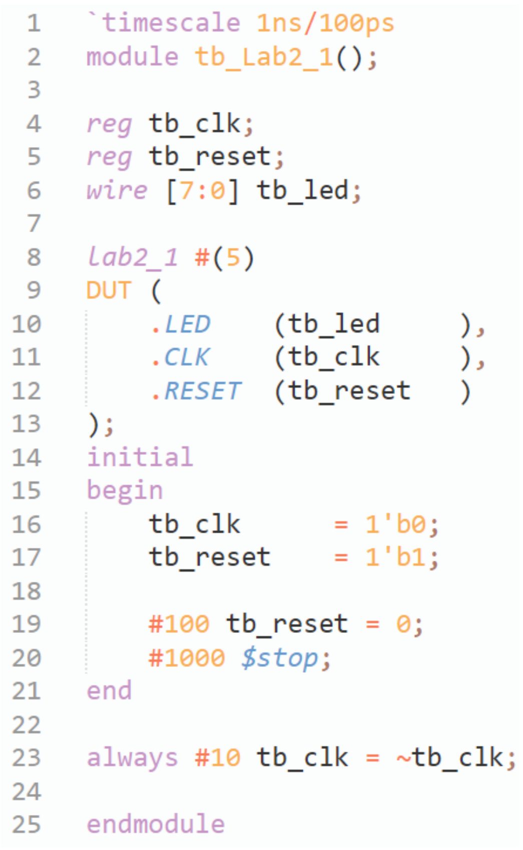
b. Select **lab2_1** top level unit

c. Select **Processing** menu => **Start => Start Analysis and Elaboration**

d. Be sure that you don’t have Errors and Critical Warnings.

Setting up ModelSim from Quartus

1 You need to create a source code for simple testbench **tb_Lab2_1 (**file: **tb_Lab2_1.v)**.



```
`timescale 1ns/100ps
module tb_Lab2_1();

reg tb_clk;
reg tb_reset;
wire [7:0] tb_led;

Lab2_1 #(5)
DUT (
    .LED    (tb_led      ),
    .CLK    (tb_clk      ),
    .RESET  (tb_reset    )
);
initial
begin
    tb_clk       = 1'b0;
    tb_reset     = 1'b1;

    #100 tb_reset = 0;
    #1000 $stop;
end

always #10 tb_clk = ~tb_clk;

endmodule
```

Figure 16

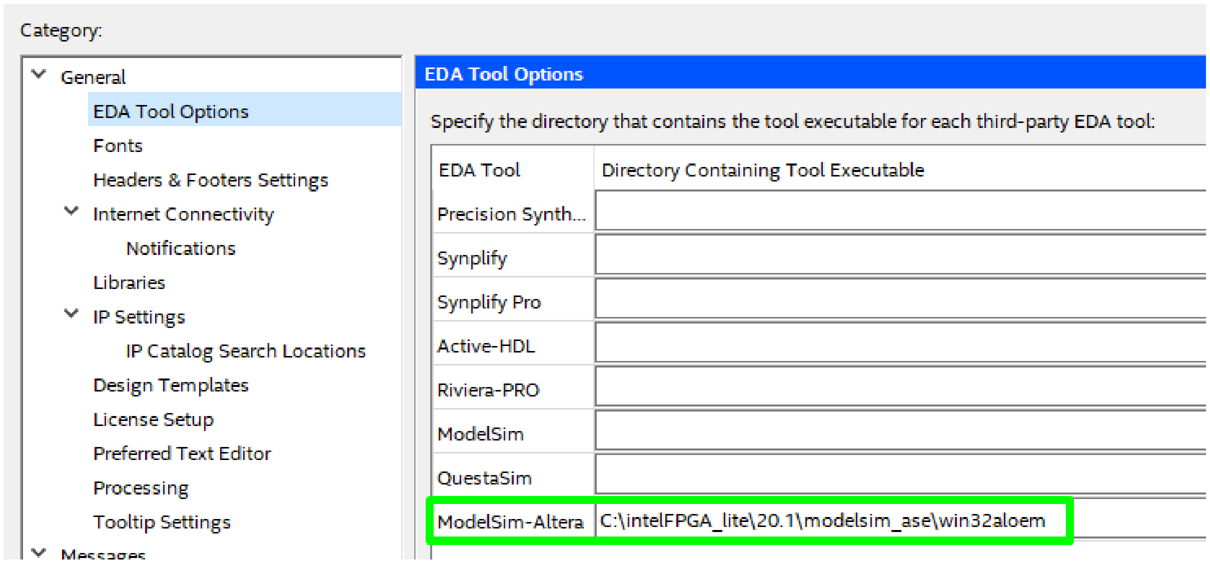
This is a simple testbench. Take a look at how the connections are made between the module and the IP.

- First, notice all the inputs of the **lab2_1** module (**tb_clk, tb_reset**) are declared as type **reg** in the **tb_Lab2_1.v** file**.**
- Next, all the outputs of the **lab2_1** are declared as type **wire** i.e. **[7:0] tb_led**.
- In the **initial block** (The initial block in Verilog is used generally in test benches. It is a block that only runs once unlike the always block):
  - all inputs are initialized
  - Then, after 100 ns, signal **tb_reset** switched to inactive state, i.e. "0".
  - Then after 1000 ns, the simulation process will be stopped by command **$stop**.
- **always** block creates a clock of **50 MHz** for **CLK** input.

Once you have completed the above sections you need to launch ModelSim for simulations. To do so you need to check (and may be to change) settings so that ModelSim can open through Quartus Prime Lite.

1 Go to **Tools → Options → EDA Tool Options**.
2 In ModelSim-Altera, enter the executable pathway.

For finding the executable pathway, locate where the Quartus was installed by you (For example: C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem).


Category:

- General
  - EDA Tool Options
  - Fonts
  - Headers & Footers Settings
  - Internet Connectivity
    - Notifications
  - Libraries
  - IP Settings
    - IP Catalog Search Locations
  - Design Templates
  - License Setup
  - Preferred Text Editor
  - Processing
  - Tooltip Settings
- Messages

EDA Tool Options

Specify the directory that contains the tool executable for each third-party EDA tool:

EDA Tool	Directory Containing Tool Executable
Precision Synth...	
Synplify	
Synplify Pro	
Active-HDL	
Riviera-PRO	
ModelSim	
QuestaSim	
ModelSim-Altera	C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem

Figure 17

Select **OK** when finished.

3 Next step is to make sure that the test bench can be compiled without errors.

a. Go to **Assignments** => **Settings** => **EDA Tool Settings** => **Simulation**.

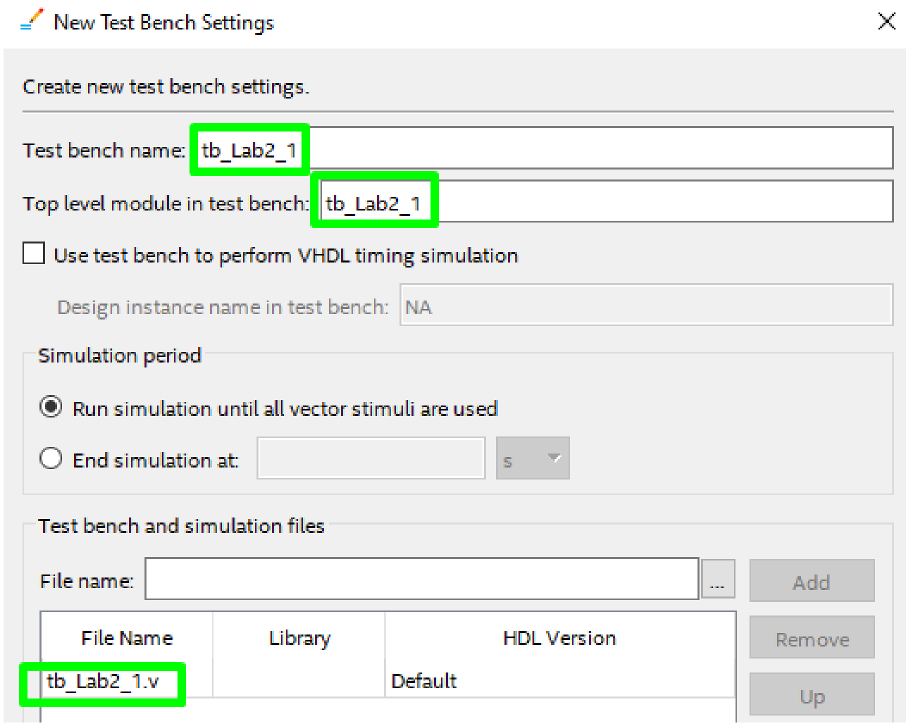
b. Select **ModelSim-Altera** in the **Tool Name** field.

c. Under **NativeLink Settings** select **Compile Test Benches** and click on **Test Benches...**

d. In the **Test Benches** Dialog Box click on **New...**

e. In the field Test bench name, print the name **tb_Lab2_1** (It is the top module of the testbench too).

f. In the field **File name** select the Testbench file tb_Lab2_1.v to be added. Click **Add**.


New Test Bench Settings

Create new test bench settings.

Test bench name: tb_Lab2_1

Top level module in test bench: tb_Lab2_1

☐ Use test bench to perform VHDL timing simulation

Design instance name in test bench: NA

Simulation period

● Run simulation until all vector stimuli are used

○ End simulation at: s

Test bench and simulation files

File name:

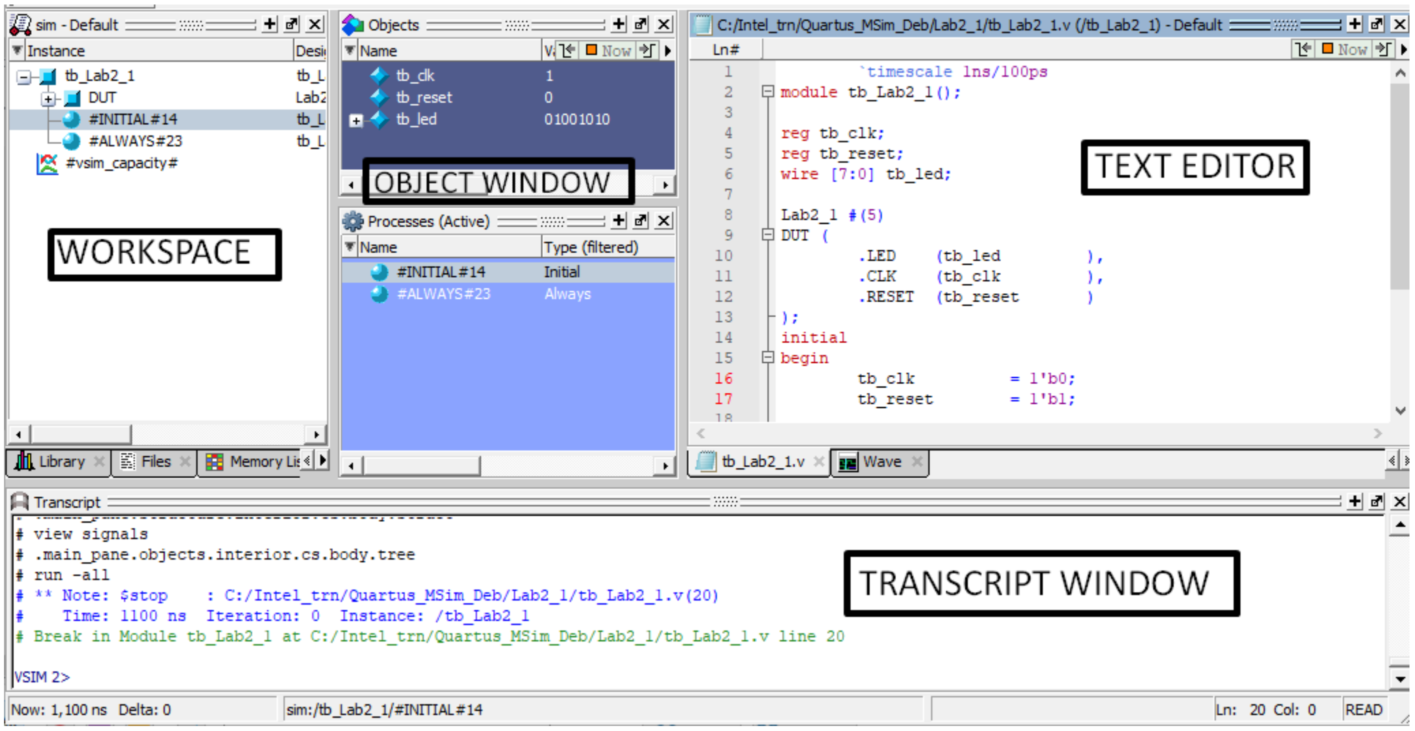
File Name	Library	HDL Version
tb_Lab2_1.v		Default

Figure 18

g. Click **OK**.

h. Click **OK** again**.**
i. Click **OK** again**.**

4 Once all the settings are in place you can go to in the toolbar to **Start Analysis & Synthesis** of the design.
5 Next, from Quartus: **Tools=> Run Simulation=> RTL Simulation**
6 Once this step is completed. ModelSim should have opened up on the screen. The GUI for ModelSim is shown below.



Name	Value
tb_clk	1
tb_reset	0
tb_led	01001010

```
        `timescale 1ns/100ps
module tb_Lab2_1();

reg tb_clk;
reg tb_reset;
wire [7:0] tb_led;

Lab2_1 #(5)
DUT (
        .LED    (tb_led         ),
        .CLK    (tb_clk         ),
        .RESET  (tb_reset       )
);
initial
begin
        tb_clk          = 1'b0;
        tb_reset        = 1'b1;
```

```
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: $stop    : C:/Intel_trn/Quartus_MSim_Deb/Lab2_1/tb_Lab2_1.v(20)
#    Time: 1100 ns  Iteration: 0  Instance: /tb_Lab2_1
# Break in Module tb_Lab2_1 at C:/Intel_trn/Quartus_MSim_Deb/Lab2_1/tb_Lab2_1.v line 20

VSIM 2>
```

Figure 19

Note: If you get error loading the design related to source files you might have to go to your working folder, where the project is stored, and check and correct the files with errors.

If you have used ModelSim before and know how to move around the various tabs, you may not see the same layout as shown in Figure 19. You can go to Layout => Reset to get the same layout of the GUI.

Workspace:

The workspace consists of various tabs, each tab having a different functionality.

Library tab represents various modules and files present in the design.

Memory List tab will represent the values in the memory units used in the design.

Once the design is simulated, a tab called sim is created which displays the hierarchical structure of the design. You can navigate within the hierarchy by clicking on any line with a '+' (expand) or '-' (contract) icon.

Objects Window Pane:

The Objects pane shows the names and current values of data objects in the current region (selected in the Workspace).

Data objects include signals, nets, registers, constants and variables not declared in a process, generics, and parameters.

Transcript Window:

The Transcript pane provides a command-line interface and serves as an activity log including status and error messages.

Text Editor:

This pane allows you to edit and read design files in ModelSim.

The Wave window

7 Select the tab Wave to see the wave window. You can now observe the waveform and verify the functionality of the design.

a. **Undock** the wave window .

b. **Zoom Full** the wave window .

c. Set radix for tb_led bus as ASCII

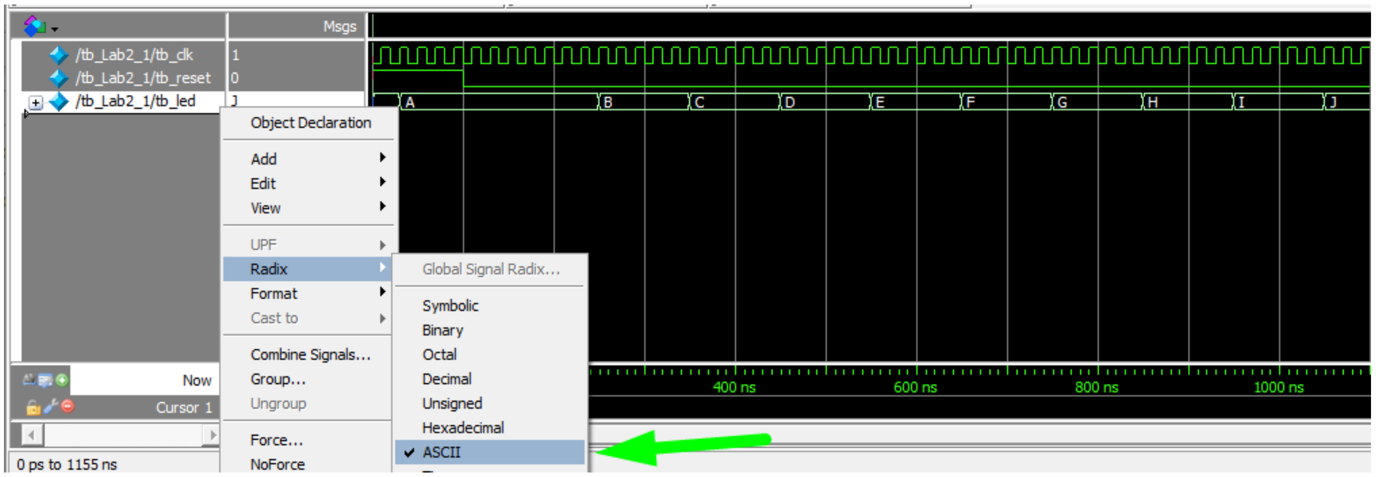


Figure 20

You should see A, B, C, D … letters displayed as on Figure 20.

Some tips:

- If you want to get to the next rising or falling edge of a particular signal. Select that signal in

the waveform and click on in the toolbar for respective actions.

If you want to change the default timescale (for example from ns to us), navigate the cursor on the lower part near the timescale as shown on Figure 21, then right click and select **Grid, Timeline & Cursor Control..**

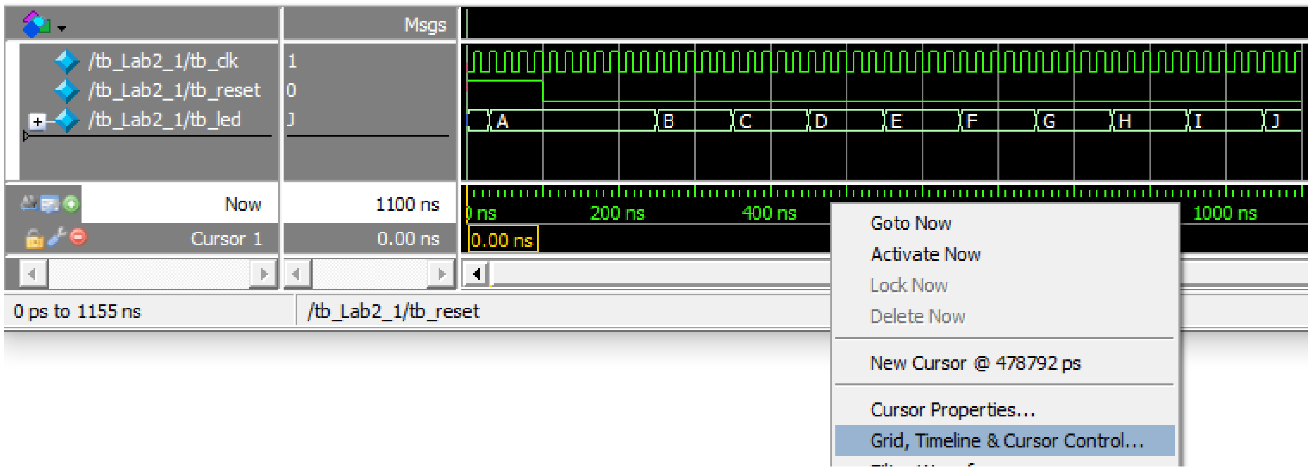


Figure 21

- In the Grid, Timeline & Cursor Control dialogue box, select us as the Time Units as shown on the Figure 22.

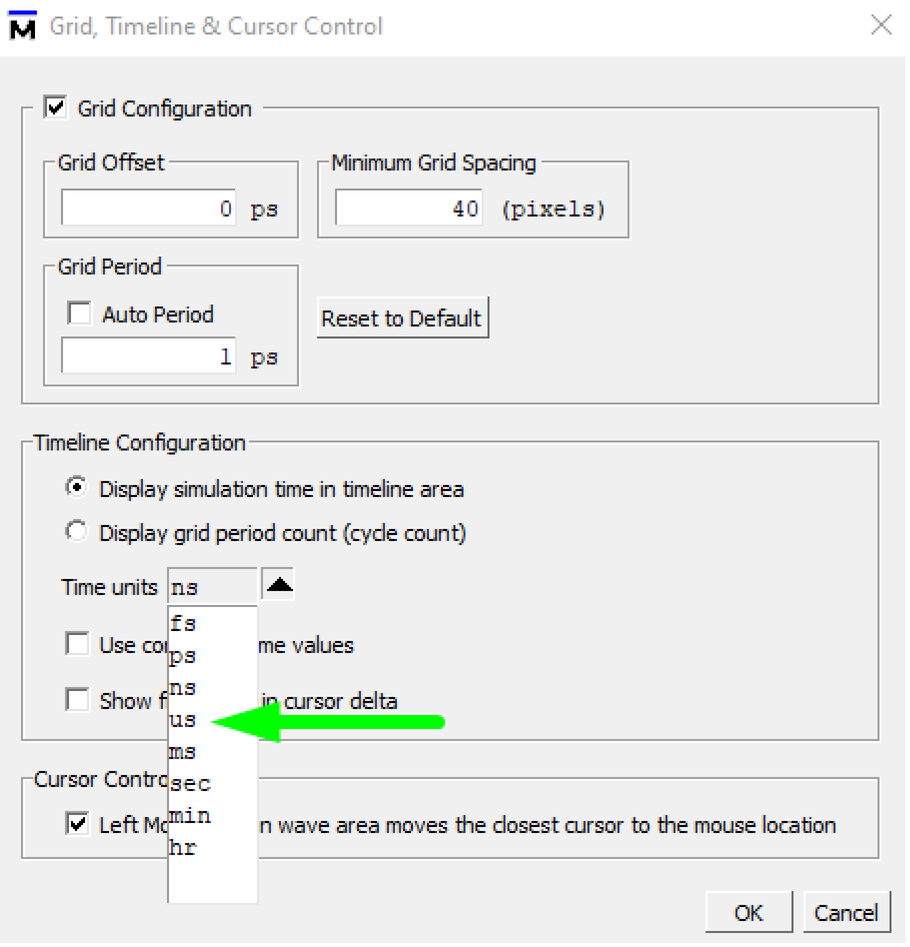
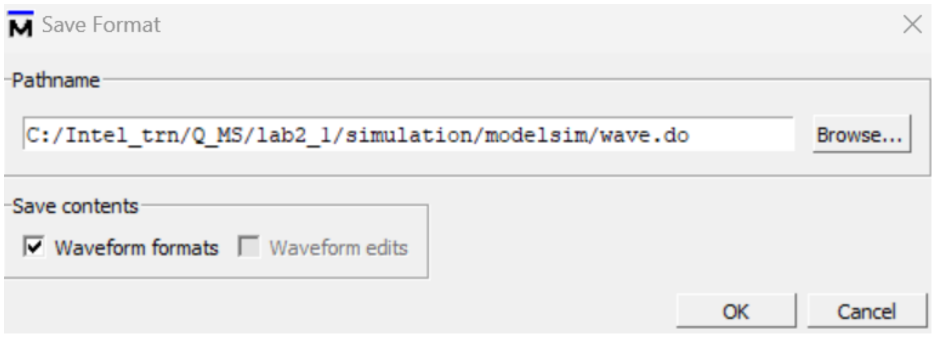

Grid, Timeline & Cursor Control
Grid Configuration
Grid Offset
0 ps
Minimum Grid Spacing
40 (pixels)
Grid Period
Auto Period
1 ps
Reset to Default
Timeline Configuration
Display simulation time in timeline area
Display grid period count (cycle count)
Time units ns
fs
ps
ns
us
ms
sec
min
hr
Use co... me values
Show f... in cursor delta
Cursor Contro...
Left Mo... n wave area moves the closest cursor to the mouse location
OK
Cancel

Figure 22

Click **OK** and find differences in the waveform window.

- To save this particular set of signals in the waveform:
  a. Go to **File** => **Save Format** in the wave window.
  b. Name it "**wave_my_lab2_1.do**"



C:/Intel_trn/Q_MS/lab2_1/simulation/modelsim/wave.do

[✓ Waveform formats] Waveform edits

Figure 23

c. Click **Save.**

*Next time you, running commands from the Modelsim Transcript, can type **"do wave_my_lab2_1.do"** before a simulation.*

d. The last step: **Stop simulation and close ModelSim window**.

Additional steps and comments

You can use opened ModelSim as usual ModelSim unless you need to change any IP unit. Therefore, you can change source files and test benches, then recompile ones, restart simulation, debug the source codes. If you need to change IP you need to close ModelSim, switch back to Quartus Prime and change what you want. Then you need to run from Quartus Prime: **Tools=> Run Simulation=> RTL Simulation**.

Pay attention:

When you run RTL Simulation from Quartus for the first time, a folder called **simulation** was generated in the Project folder. The folder has sub-folder **Modelsim** – it is the working folder for ModelSim**.**

- Under **Modelsim** folder look for a .do extension file.
  There is a file **"Lab2_1_run_msim_rtl_verilog.do"** which was created.
- You can run this do file in Modelsim independently from Quartus Prime:
  - Open ModelSim using the start menu.

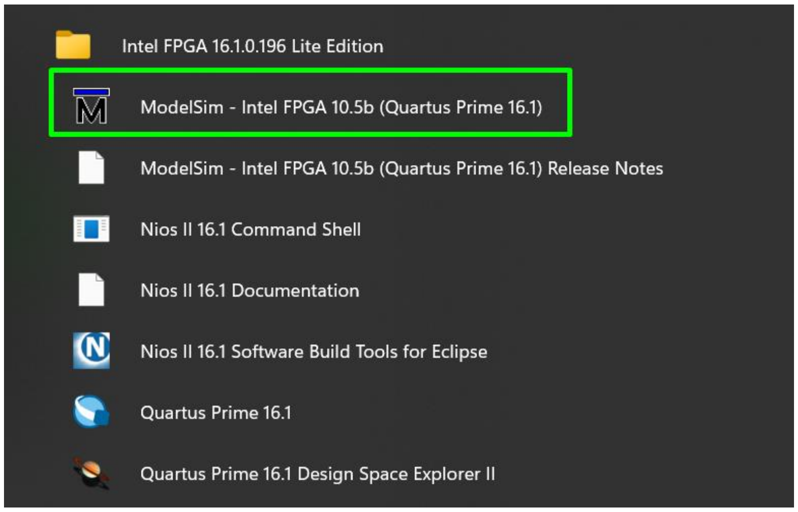
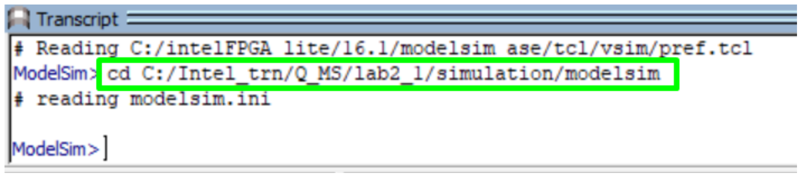

Intel FPGA 16.1.0.196 Lite Edition
ModelSim - Intel FPGA 10.5b (Quartus Prime 16.1)
ModelSim - Intel FPGA 10.5b (Quartus Prime 16.1) Release Notes
Nios II 16.1 Command Shell
Nios II 16.1 Documentation
Nios II 16.1 Software Build Tools for Eclipse
Quartus Prime 16.1
Quartus Prime 16.1 Design Space Explorer II

Figure 24

  - Once the software loads:
    - Go to the working folder location, typing in the transcript window the command:

cd C:/Intel_trn/Q_MS/lab2_1/simulation/modelsim



```
# Reading C:/intelFPGA lite/16.1/modelsim ase/tcl/vsim/pref.tcl
ModelSim> cd C:/Intel_trn/Q_MS/lab2_1/simulation/modelsim
# reading modelsim.ini

ModelSim>
```

Figure 25

- Once in the folder, type the command in the transcript window:

do Lab2_1_run_msim_rtl_verilog.do

The design should load without using the Quartus Software.

- Close the waveform window, if one is open.
- Type in the transcript window the command

do wave_my_lab2_1.do

- Be sure that you have the wave as it was saved early.
- **Stop simulation and close ModelSim window**.

For more help and shortcuts on Modelsim, you can go to Help → Documentation → Tutorial.

Conclusions

With this part you learnt how to use IP modules in simulation and to launch ModelSim simulation from Quartus Prime by using Native link.

PART2 “Launching ModelSim independently from Quartus Prime”

Objectives

You will use this part to familiarize yourself with the following:

How to launching ModelSim independently from Quartus Prime and use IP functions from IP Catalog of Quartus Prime with ModelSim.

Launching ModelSim independently from Quartus Prime

1 Start ModelSim Intel FPGA Starter Edition.

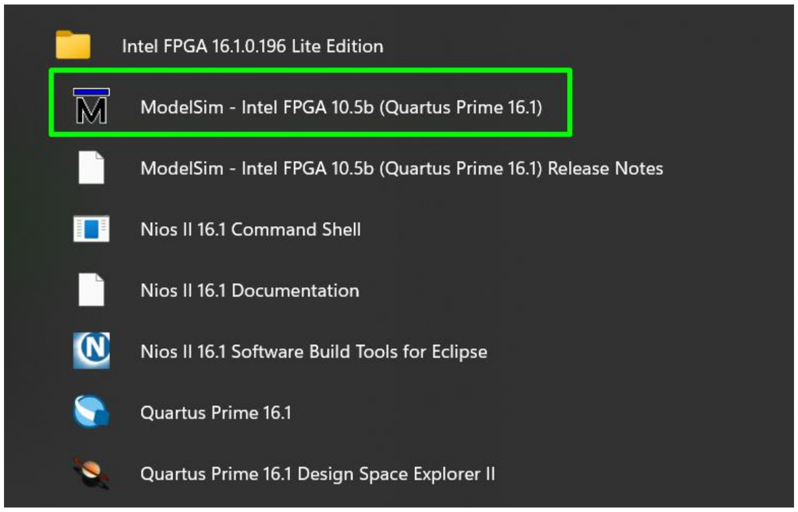
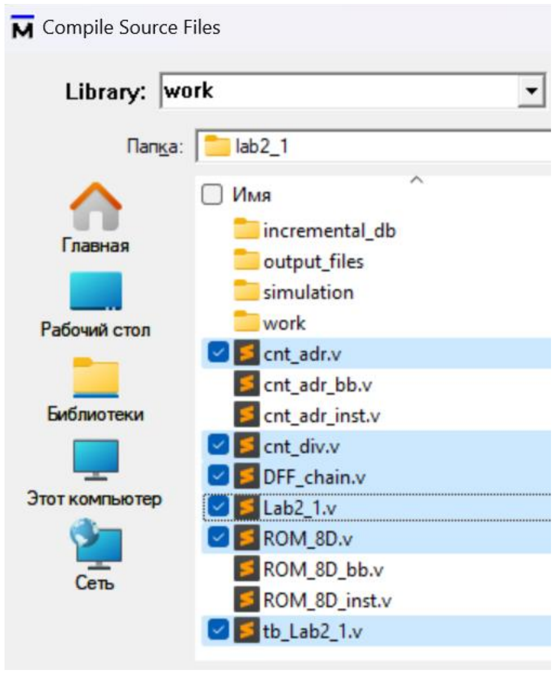

Intel FPGA 16.1.0.196 Lite Edition
ModelSim - Intel FPGA 10.5b (Quartus Prime 16.1)
ModelSim - Intel FPGA 10.5b (Quartus Prime 16.1) Release Notes
Nios II 16.1 Command Shell
Nios II 16.1 Documentation
Nios II 16.1 Software Build Tools for Eclipse
Quartus Prime 16.1
Quartus Prime 16.1 Design Space Explorer II

Figure 26

2 Change working Directory.
   a. Select **File => Change Directory** and change to the directory **C:\Intel_trn\Q_MS\Lab2_1**.
   b. OR by typing in the Transcript window: **cd C:/Intel_trn/Q_MS/Lab2_1**

3 Create the working library.
   a. Select **File => New => Library =>work**.
   b. OR by typing in the Transcript window: **vlib work**

4 Compile source files:
   a. Select **Compile => Compile =>** "DFF_chain.v" "ROM_8D.v" "tb_Lab2_1.v" "cnt_adr.v" "cnt_div.v" "Lab2_1.v".
   b. OR by typing in the Transcript window: **vlog -work work "DFF_chain.v" "ROM_8D.v" "tb_Lab2_1.v" "cnt_adr.v" "cnt_div.v" "Lab2_1.v"**


Compile Source Files

Library: work

Папка: lab2_1

- [ ] Имя
- incremental_db
- output_files
- simulation
- work
- [✓] cnt_adr.v
- cnt_adr_bb.v
- cnt_adr_inst.v
- [✓] cnt_div.v
- [✓] DFF_chain.v
- [✓] Lab2_1.v
- [✓] ROM_8D.v
- ROM_8D_bb.v
- ROM_8D_inst.v
- [✓] tb_Lab2_1.v

Figure 27

5 Be sure that there are no any Errors. If there are ones you will need to correct source files.

6 Now you're ready to load the design into the simulator:

a. In the Library window, click the '+' sign next to the **work** library to show the files contained there and double-click **tb_Lab2_1** to load the design.

b. OR by typing in the Transcript window: **vsim work.tb_Lab2_1**

7 You will find two errors reported in the Transcript window

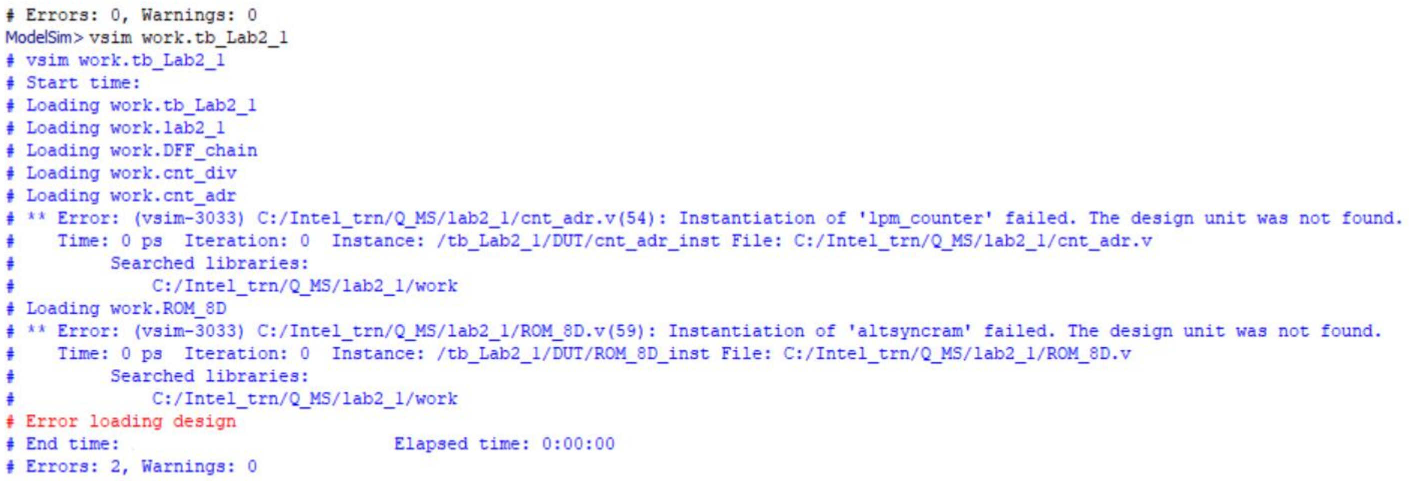

```
# Errors: 0, Warnings: 0
ModelSim> vsim work.tb_Lab2_1
# vsim work.tb_Lab2_1
# Start time:
# Loading work.tb_Lab2_1
# Loading work.lab2_1
# Loading work.DFF_chain
# Loading work.cnt_div
# Loading work.cnt_adr
# ** Error: (vsim-3033) C:/Intel_trn/Q_MS/lab2_1/cnt_adr.v(54): Instantiation of 'lpm_counter' failed. The design unit was not found.
#    Time: 0 ps  Iteration: 0  Instance: /tb_Lab2_1/DUT/cnt_adr_inst File: C:/Intel_trn/Q_MS/lab2_1/cnt_adr.v
#         Searched libraries:
#             C:/Intel_trn/Q_MS/lab2_1/work
# Loading work.ROM_8D
# ** Error: (vsim-3033) C:/Intel_trn/Q_MS/lab2_1/ROM_8D.v(59): Instantiation of 'altsyncram' failed. The design unit was not found.
#    Time: 0 ps  Iteration: 0  Instance: /tb_Lab2_1/DUT/ROM_8D_inst File: C:/Intel_trn/Q_MS/lab2_1/ROM_8D.v
#         Searched libraries:
#             C:/Intel_trn/Q_MS/lab2_1/work
# Error loading design
# End time:                          Elapsed time: 0:00:00
# Errors: 2, Warnings: 0
```

Figure 28

8 You need to reference **lpm** and **altera_mf** libraries to fix the error.

a. Select **Simulate => Start Simulation**

b. In the window appeared the '+' sign next to the work library to show the files contained there and click **tb_Lab2_1** to select it.

c. Select **Libraries** folder

d. In the field Search Libraries click **Add…**

e. In the window appeared select **altera_mf_ver** library

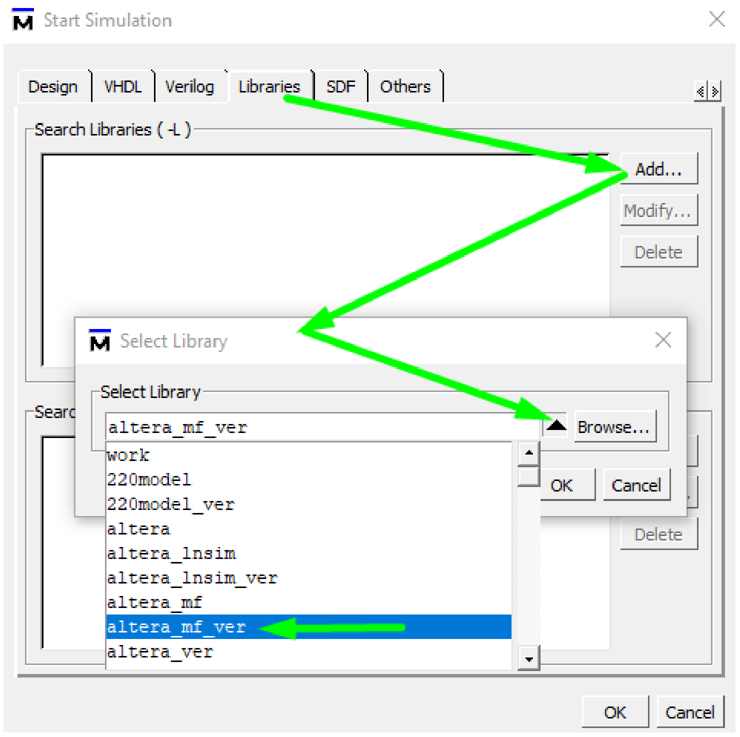


Figure 29

f. Click OK
g. Click **Add…** again
h. Select **lpm_ver** library.
i. Click OK
j. Click OK
k. The design will be loaded into the simulator without any errors. If you still have some errors you need to check all steps from the beginning.
l. Type in the transcript window the command:
do C:/Intel_trn/Q_MS/lab2_1/simulation/modelsim/ wave_my_lab2_1.do
Wave window with top-level signals will appear.
m. Type in the transcript window the command: **run 1000 ns**
n. Undock wave window
o. Zoom full wave window.
p. Finally, be sure that you see something like the Figure 30.

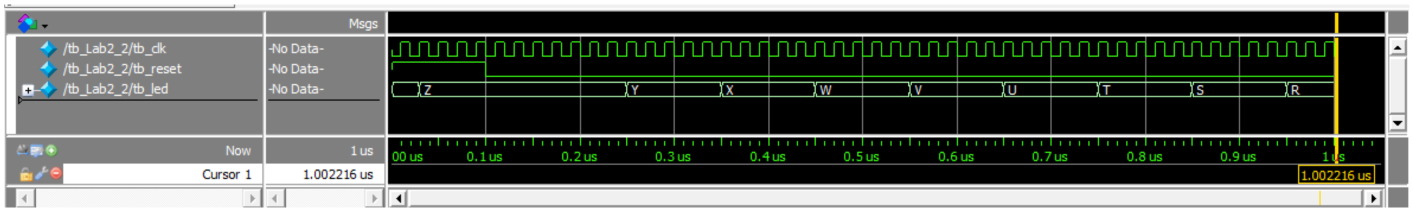


Figure 30

9 End simulation and close ModelSim window.

Conclusions

With this part you learnt how to use IP modules in simulation and ModelSim simulation independently from Quartus Prime.