ECSE-325
Digital Systems

Lab #1 – *VHDL Design and Simulation* **Winter 2025**

Introduction .

In this lab you will be introduced to the Intel *Quartus* program and learn the basics of digital simulation using the *Questa* simulation program.

OPTIONAL: If you wish, you can also download the Quartus and ModelSim/Questa software from the Intel website and run them on your home computer (Windows and Linux only, there is no Mac version, nor is there any version that can run on mobile devices).

Choose the most recent version (v23.1.1), the labs will work just the same, although some of the screenshots may look different.

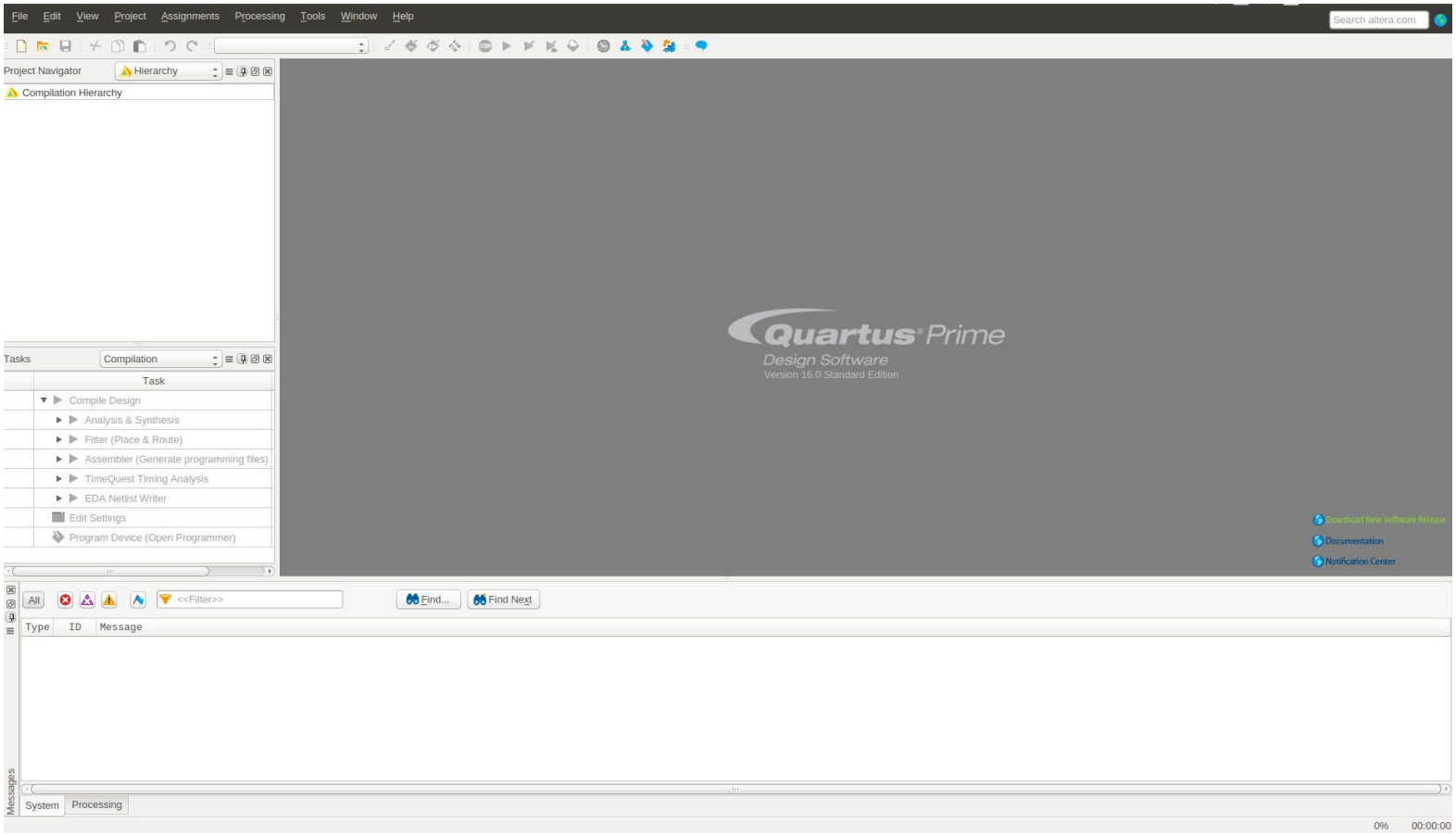
The version that is on the lab computers is v22.1.2, so you could use that version instead.
https://www.intel.com/content/www/us/en/software-kit/825278/intel-quartus-prime-lite-edition-design-software-version-23-1-1-for-windows.html

You only need to install device support for the Cyclone V device.

Tool Startup and Project Creation

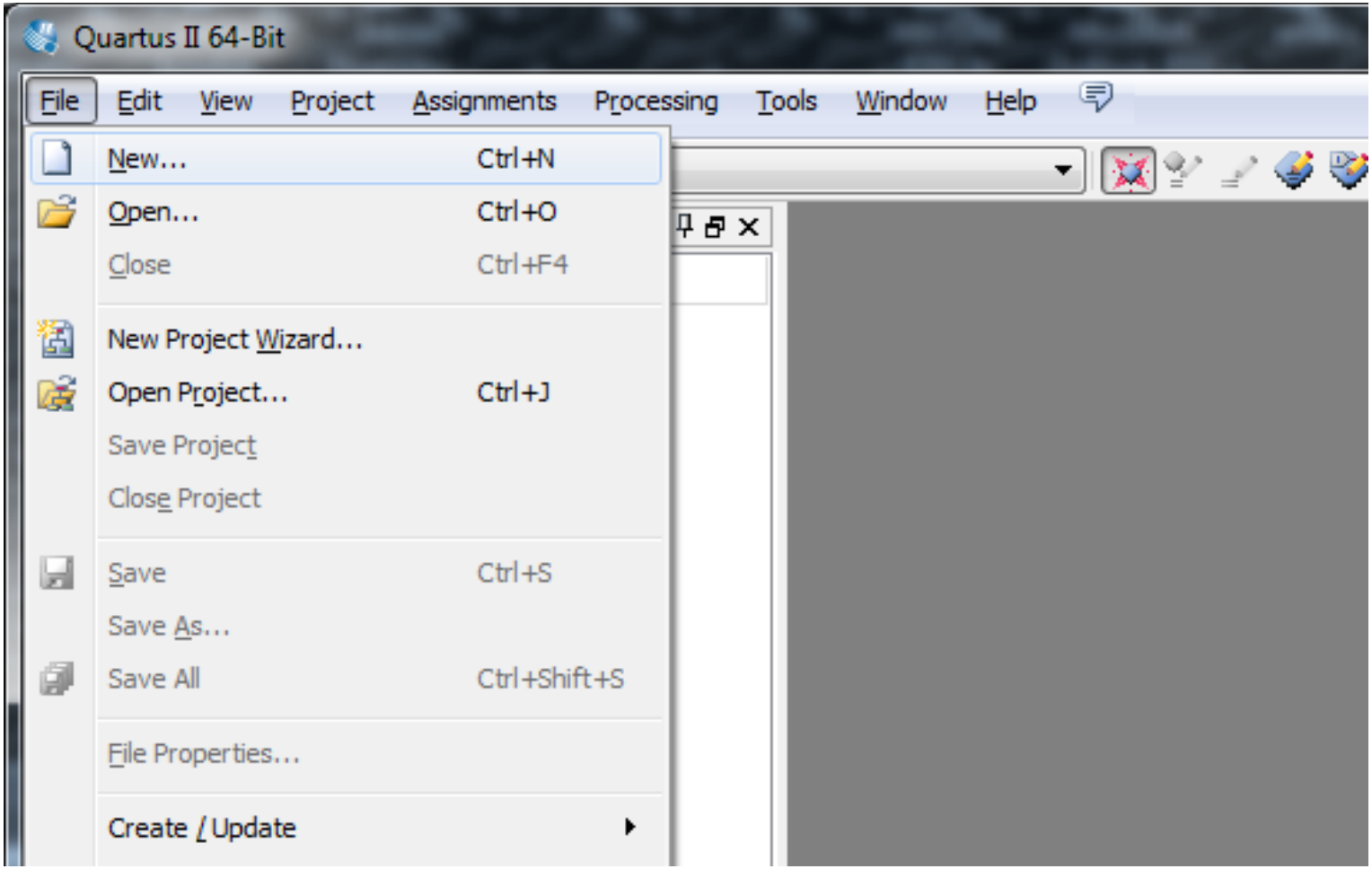
You can start the Quartus tool by selecting it from the Windows start menu.

On startup, you should see the following window (it might slightly differ in your case) for an empty project.



Project Creation

You have to first create a project by selecting **File->New Project Wizard**.



New Project Wizard

Introduction

The New Project Wizard helps you create a new project and preliminary project settings, including the following:

- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

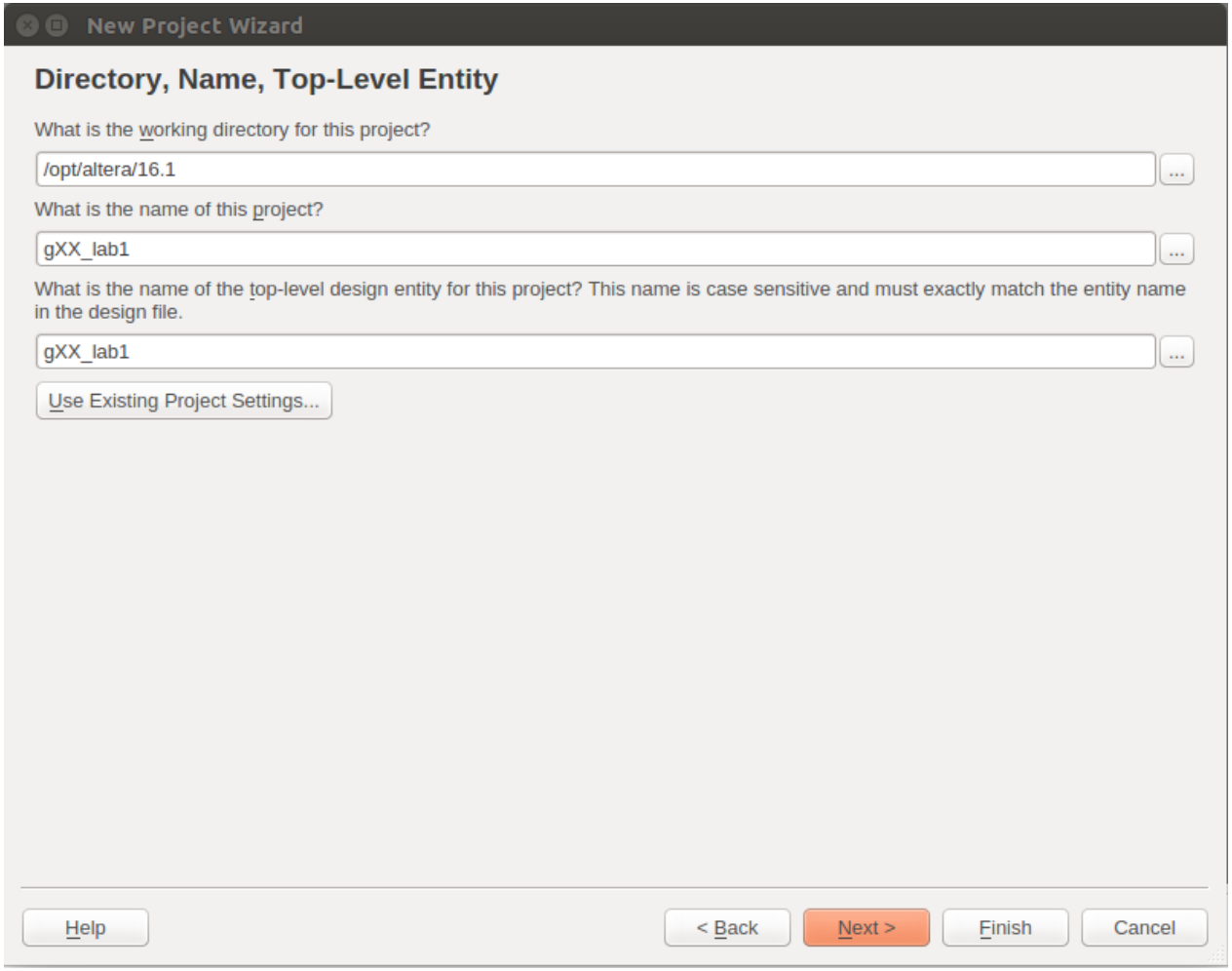
You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

☐ Don't show me this introduction again

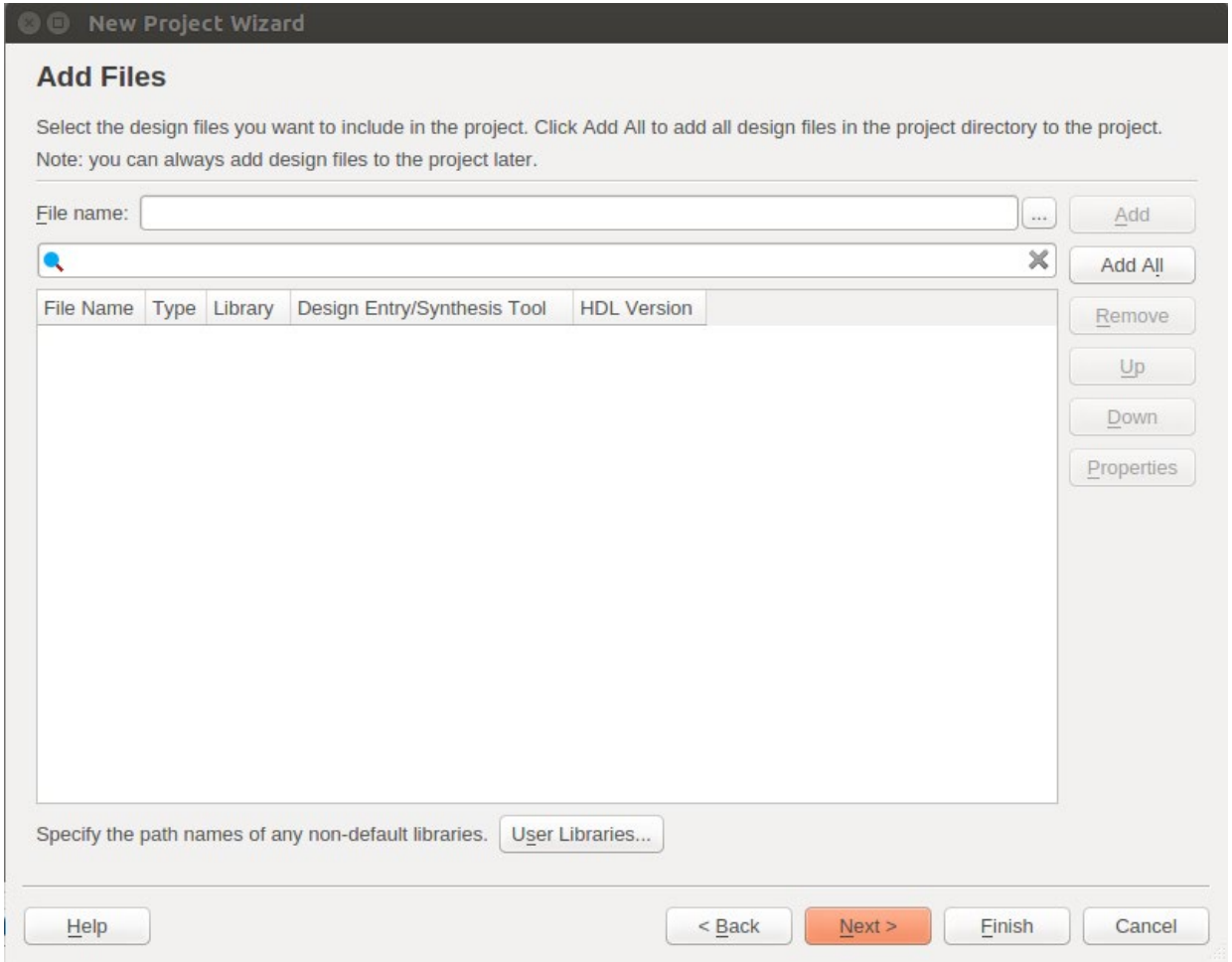
Help | < Back | Next > | Finish | Cancel

In the second window you should give the project the name: gXX_lab1 where XX is replaced with your 2-digit group number. The working directory for your project will be different than that shown here since you should use your network drive for all project files.

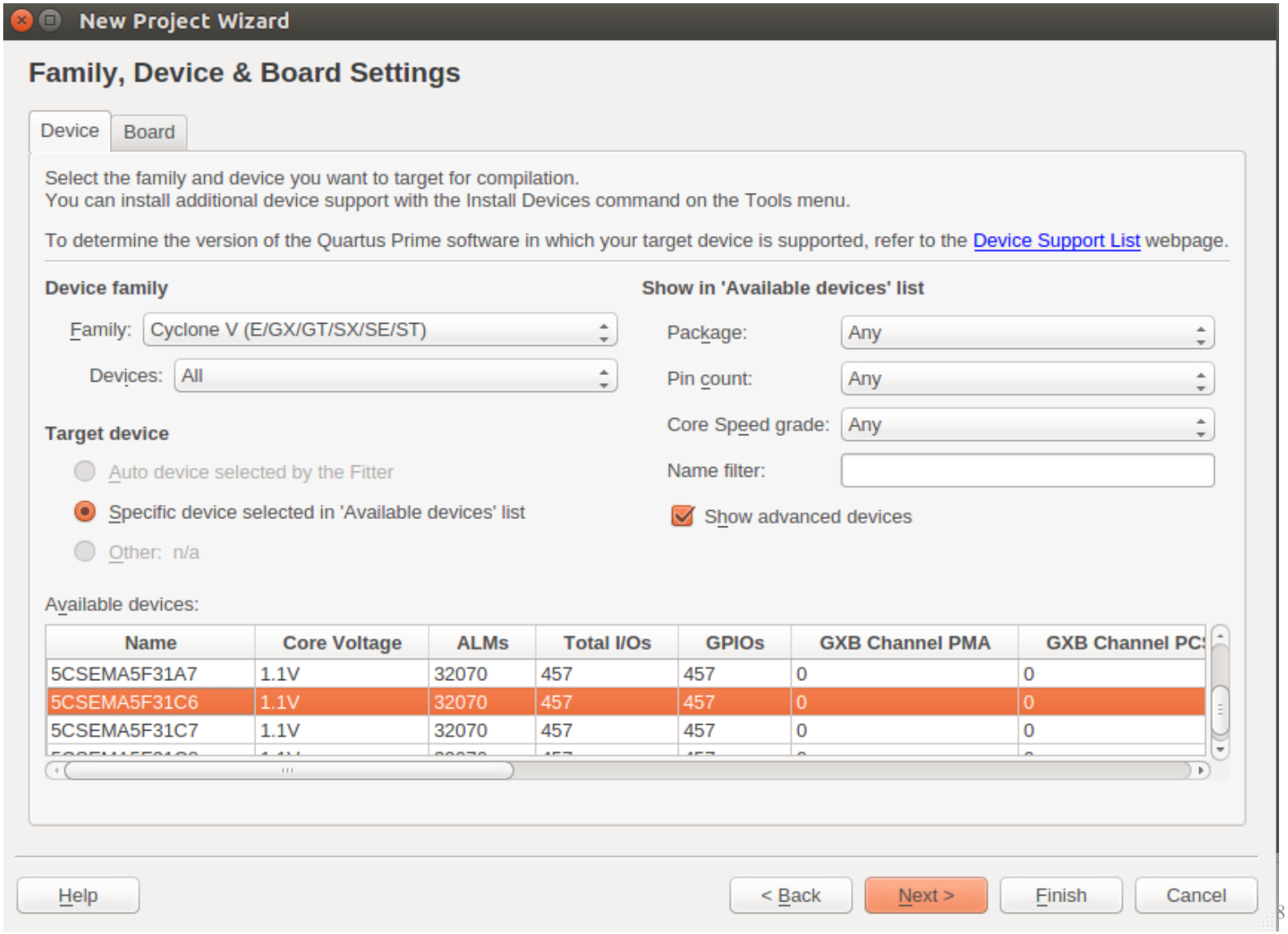
Since you do not have a project template, you can proceed with the Empty Project selection



You will add files later, so you should just select “Next >”



Although not important for the first lab, select the Cyclone V device 5CSEMA5F31C6. This is the type of FPGA that is on the Terasic DE1-SoC board, which will be used in later labs.



New Project Wizard

EDA Tool Settings

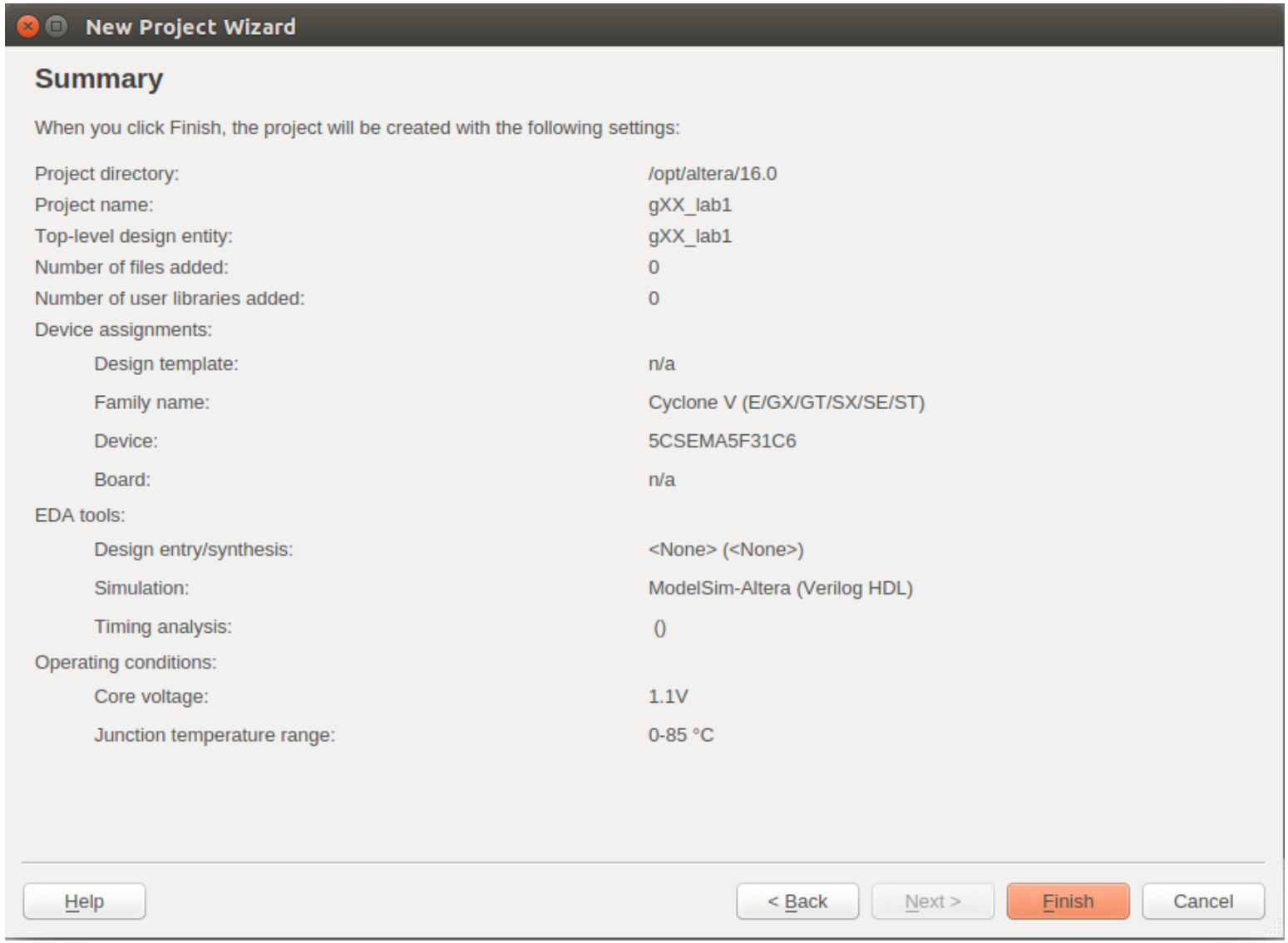
Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Sy…	<None>	<None>	Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	Verilog HDL	Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

Help < Back Next > Finish Cancel

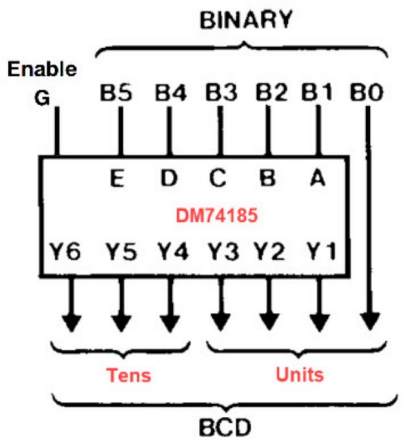
On the last page, you should check whether the project was created properly



1. Design of a 6-bit Binary-to-BCD Converter Circuit

In this lab you will design a 16-bit binary-to-bcd converter circuit in VHDL. This will be done by first designing a 6-bit binary-to-bcd converter module and using 16 of these modules to implement the full 16-bit version.

We will emulate the operation of the DM74185 binary-to-bcd converter chip from National Semiconductor.

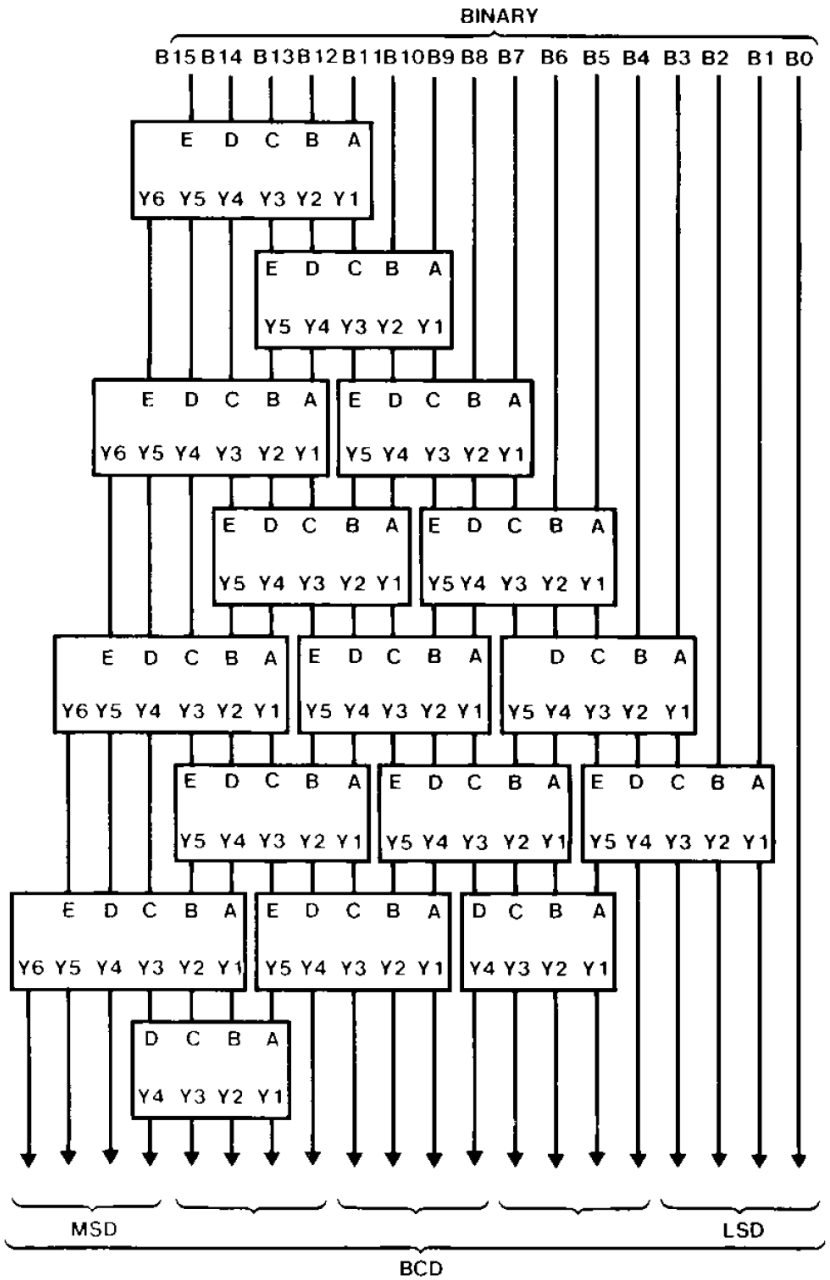


This module takes the 5 most-significant-bits of a 6-bit input and generates the 3 least-significant-bits of the higher digit, and the 3 most-significant-bits of the lower digit. The least-significant-bit of the lower digit is just taken directly from the least-significant-bit of the 6-bit input.

The truth table of this module is shown on the next page.

Function Tables

Binary Words		Inputs					Outputs					
		Binary Select										
		E	D	C	B	A	Y6	Y5	Y4	Y3	Y2	Y1
0	1	L	L	L	L	L	L	L	L	L	L	L
2	3	L	L	L	L	H	L	L	L	L	L	H
4	5	L	L	L	H	L	L	L	L	L	H	L
6	7	L	L	L	H	H	L	L	L	L	H	H
8	9	L	L	H	L	L	L	L	L	H	L	L
10	11	L	L	H	L	H	L	L	H	L	L	L
12	13	L	L	H	H	L	L	L	H	L	L	H
14	15	L	L	H	H	H	L	L	H	L	H	L
16	17	L	H	L	L	L	L	L	H	L	H	H
18	19	L	H	L	L	H	L	L	H	H	L	L
20	21	L	H	L	H	L	L	H	L	L	L	L
22	23	L	H	L	H	H	L	H	L	L	L	H
24	25	L	H	H	L	L	L	H	L	L	H	L
26	27	L	H	H	L	H	L	H	L	L	H	H
28	29	L	H	H	H	L	L	H	L	H	L	L
30	31	L	H	H	H	H	L	H	H	L	L	L
32	33	H	L	L	L	L	L	H	H	L	L	H
34	35	H	L	L	L	H	L	H	H	L	H	L
36	37	H	L	L	H	L	L	H	H	L	H	H
38	39	H	L	L	H	H	L	H	H	H	L	L
40	41	H	L	H	L	L	H	L	L	L	L	L
42	43	H	L	H	L	H	H	L	L	L	L	H
44	45	H	L	H	H	L	H	L	L	L	H	L
46	47	H	L	H	H	H	H	L	L	L	H	H
48	49	H	H	L	L	L	H	L	L	H	L	L
50	51	H	H	L	L	H	H	L	H	L	L	L
52	53	H	H	L	H	L	H	L	H	L	L	H
54	55	H	H	L	H	H	H	L	H	L	H	L
56	57	H	H	H	L	L	H	L	H	L	H	H
58	59	H	H	H	L	H	H	L	H	H	L	L
60	61	H	H	H	H	L	H	H	L	L	L	L
62	63	H	H	H	H	H	H	H	L	L	L	H



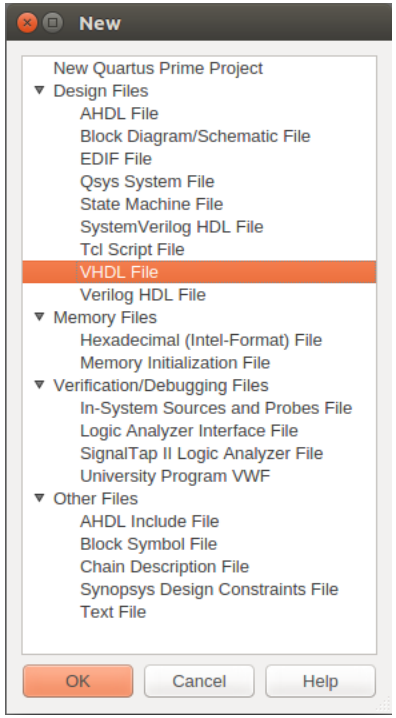
TL/F/6561–14

FIGURE 8. 16-Bit Binary-to-BCD Converter (See Note B)

We can use the DM74185 module to implement a full 16-bit binary-to-BCD converter circuit, which takes in a 16-bit binary unsigned value, and generates the corresponding 5 BCD digits.

For the blocks where the input E is not shown, assume that E is connected to 0.

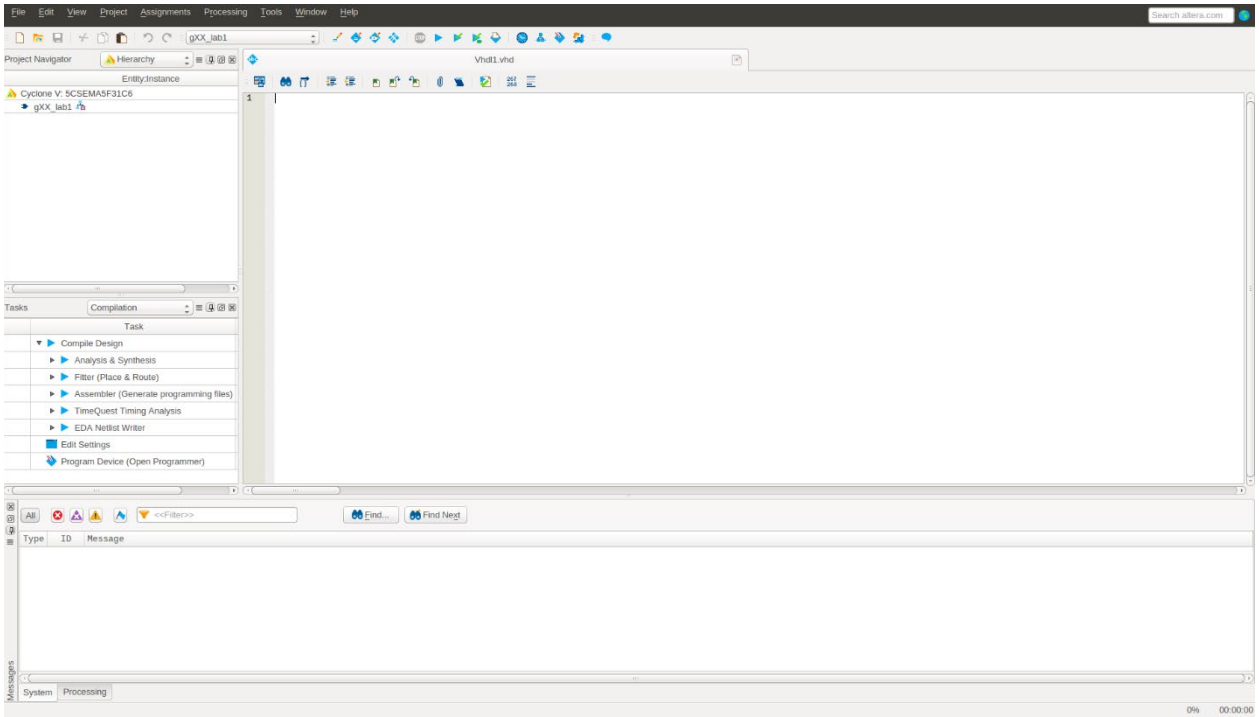
For example, the binary input: [1000111000000010] corresponds to the BCD digits: [0011][0110][0011][0101][0100] or (36354) in decimal.



We can use Quartus to enter our VHDL description of both the DM74185 module, and the implementation of the full 16-bit binary-to-BCD converter circuit.

In the Quartus file menu select new and select VHDL File then click OK.

This will open up an editing window where you can type in the VHDL description of the circuits.



VHDL Description of the DM74185 Binary-BCD circuit.

Using the following form for the entity declaration (replace the values in the header with your own information), in Quartus ***write the complete VHDL entity description*** for a circuit that implements the DM74185 Binary-BCD operation.

Use a CASE statement in a process block to implement the truth table on page 12 (make the last case a "when others" case to handle all of the other possible input patterns of the std_logic vectors, such as 'Z', 'X', etc).

Name this file gXX_DM74185.vhd, where XX is your group number.

```
--
-- entity name: gXX_DM74185  (replace "XX" by your group's number)
--
-- Version 1.0
-- Authors: (list the group member names here)
-- Date: March ??, 2025 (enter the date of the latest edit to the file)

library ieee; -- allows use of the std_logic_vector type
use ieee.std_logic_1164.all;

entity gXX_DM74185 is
 port ( EDCBA : in std_logic_vector(4 downto 0);
        Y : out std_logic_vector(5 downto 0) );
end gXX_DM74185;
```

VHDL Description of the 16-bit Binary-BCD circuit.

Using the following form for the entity declaration (replace the values in the header with your own information), ***write the complete VHDL entity description*** for a circuit that implements the 16-bit Binary-BCD operation. Use component instantiation statements to insert the 6-bit binary-bcd modules as indicated by the schematic on page 13.

Name this file gXX_Binary_BCD16.vhd, where XX is your group number.

```
--
-- entity name: gXX_Binary_BCD16  (replace "XX" by your group's number)
--
-- Version 1.0
-- Authors: (list the group member names here)
-- Date: March ??, 2025 (enter the date of the latest edit to the file)

library ieee; -- allows use of the std_logic_vector type
use ieee.std_logic_1164.all;

entity gXX_Binary_BCD16 is
 port ( bin : in std_logic_vector(15 downto 0);
         BCD5 : out std_logic_vector(3 downto 0); -- Most significant digit
         BCD4 : out std_logic_vector(3 downto 0);
         BCD3 : out std_logic_vector(3 downto 0);
         BCD2 : out std_logic_vector(3 downto 0);
         BCD1 : out std_logic_vector(3 downto 0) ); -- Least significant digit
end gXX_Binary_BCD16;
```

Simulation of the `gNN_binary_BCD16` circuit using QuestaSim

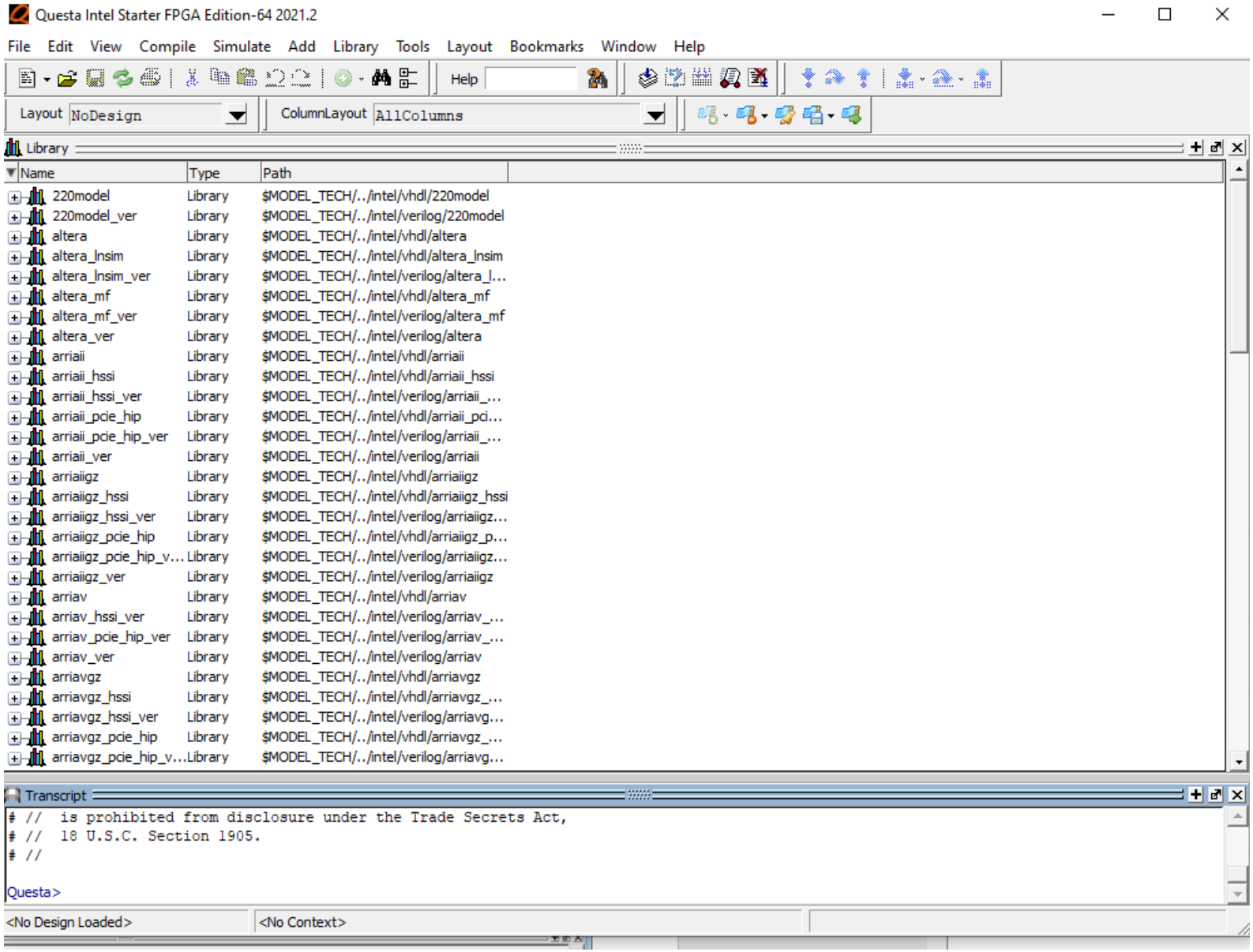
In this course, we will be using the ***QuestaSim*** simulation software, created by the company Mentor Graphics (we will use a version of it specific to Quartus, called QuestaSim-Altera).

The QuestaSim software operates on a Hardware Description Language (HDL) description of the circuit to be simulated, written either in VHDL, Verilog, or System-Verilog. **You will use VHDL**.

In Quartus, run "Processing/Start/Start Analysis and Elaboration" on the two VHDL files you wrote earlier (we don't need to do the synthesis/fitting steps) and correct any errors you might have.

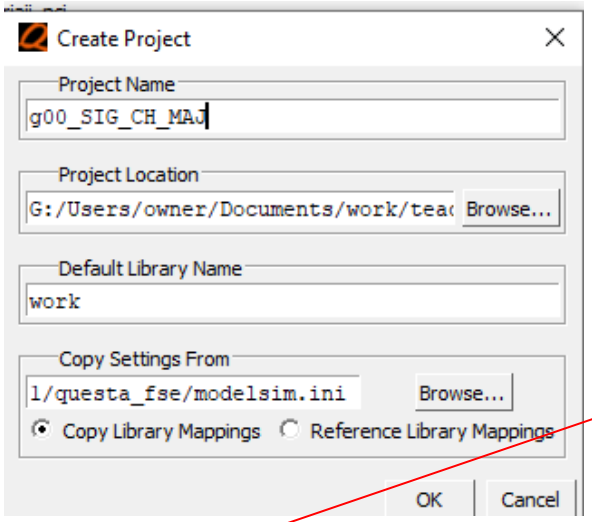
For those who want to run Questa on their home computers, you can download the "Starter" edition from https://www.intel.com/content/www/us/en/software-kit/845641/questa-intel-fpgas-pro-edition-software-version-24-3-1.html
You will need to get a (free) one-year license to run it from https://www.intel.com/content/www/us/en/support/programmable/licensing/support-center.html

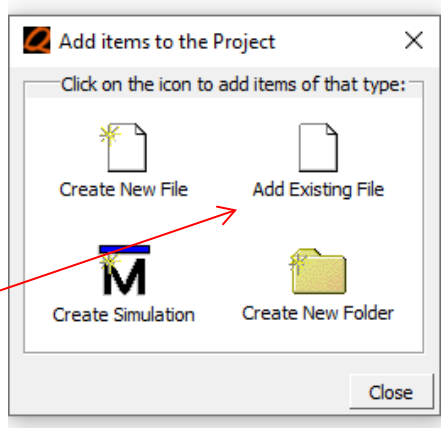
Next, select the QuestaSim program from the Windows Start menu.
A window like the one shown below will appear.



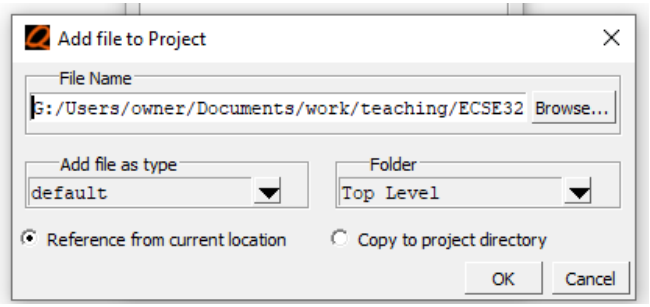
Select FILE/New/Project and, in the window that pops up, give the project the name “gNN_lab1”

Click OK.

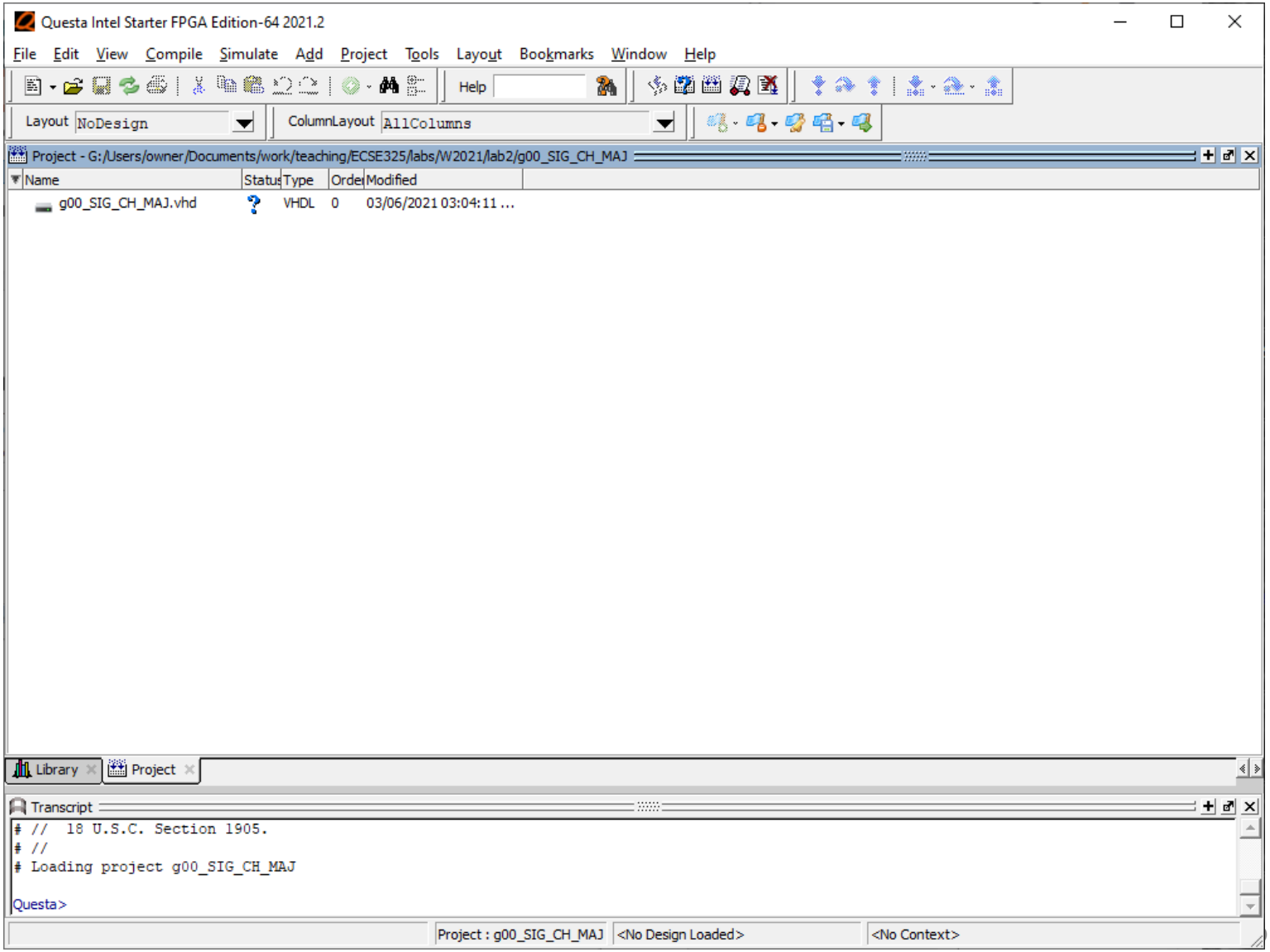




Another dialog box will appear, allowing you to add files to the project. Click on “Add Existing File” and select the two VHDL files that were generated earlier (gXX_DM74185.vhd and gXX_Binary_BCD16.vhd). You can also add files later.

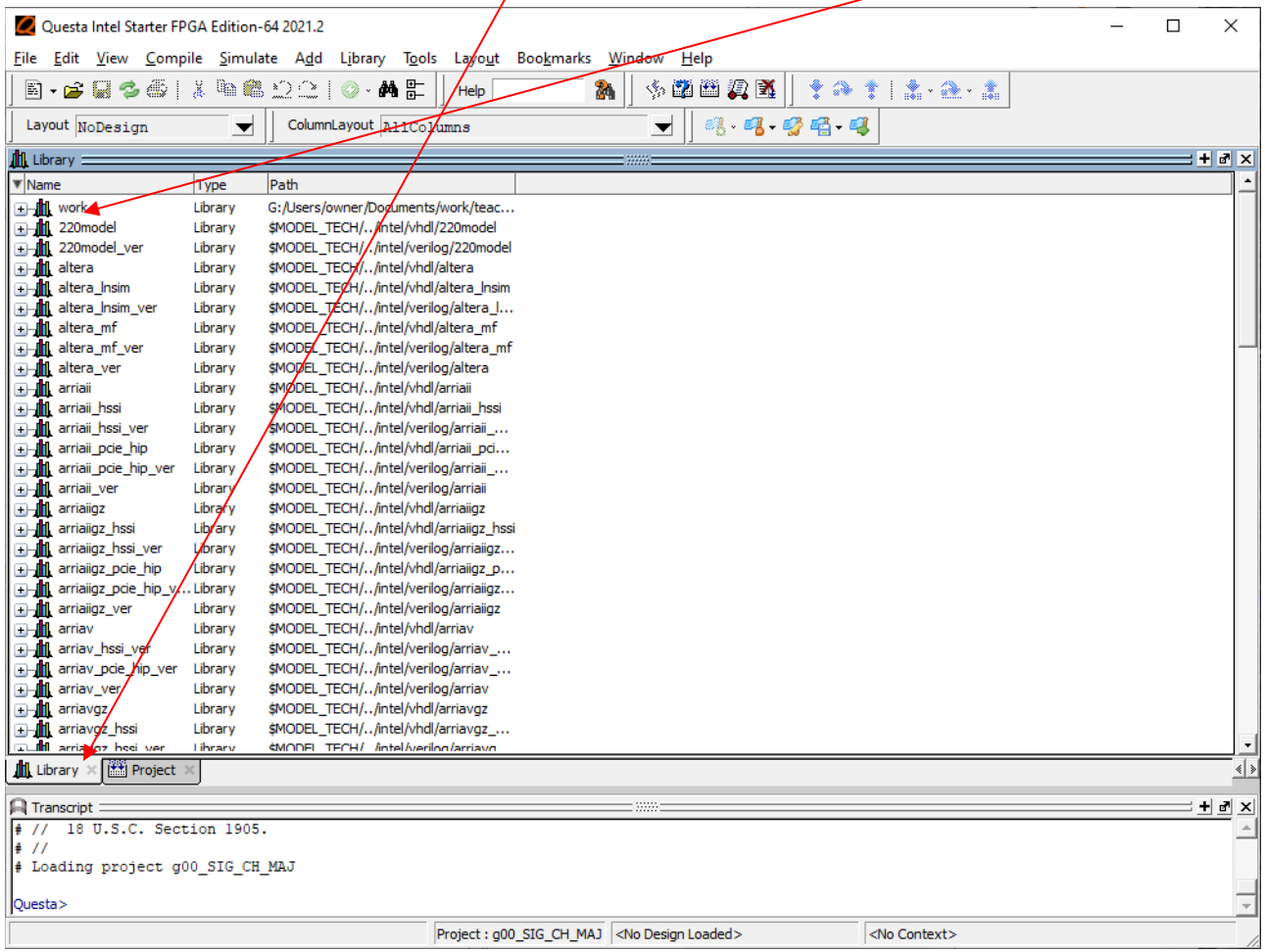


The QuestaSim window will now show the vhdl files in the Project pane.
(note: these screenshot images are for illustration only, as they refer to different files than what you will be using. But the flow is still the same.)

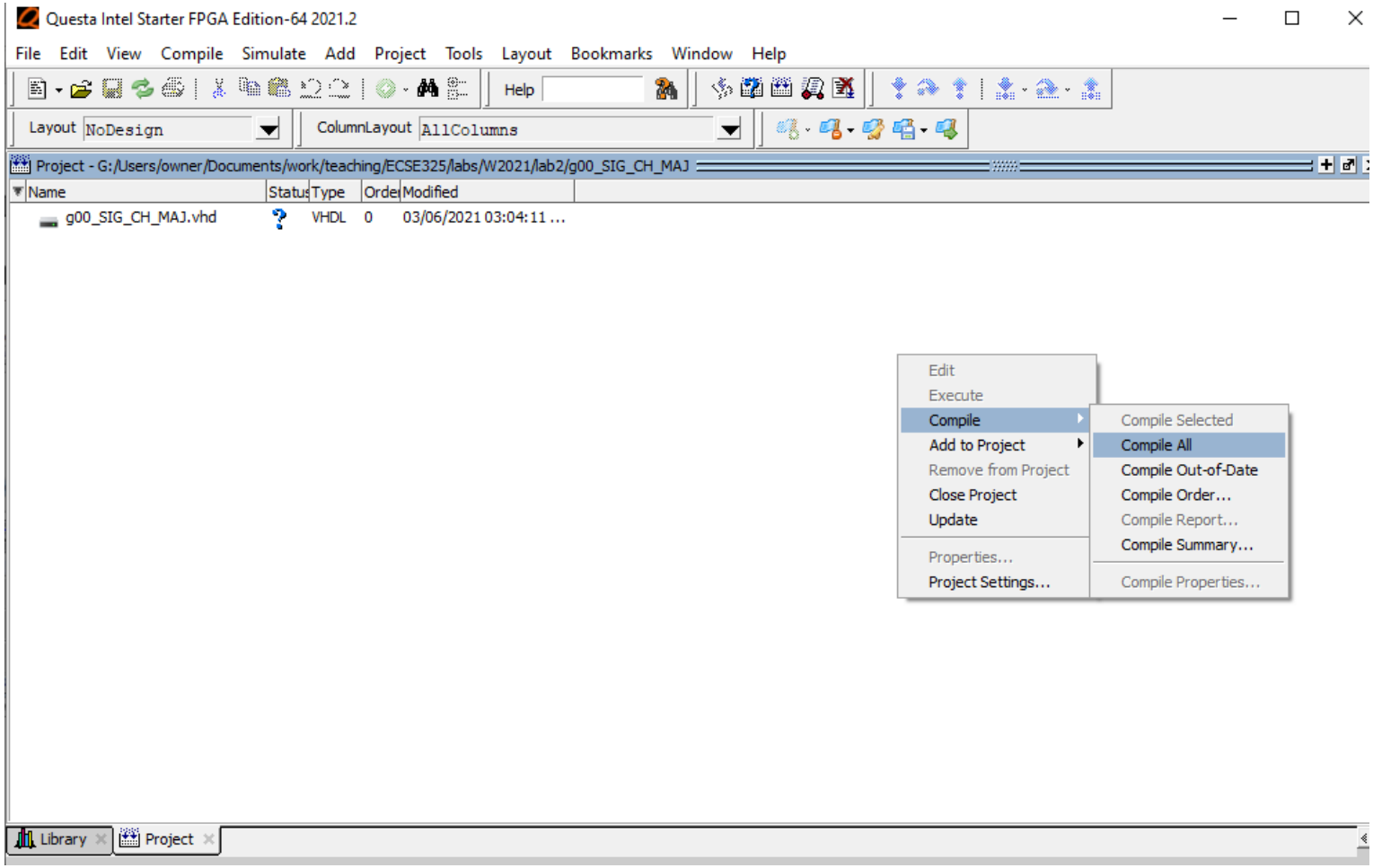


In order to simulate the design, QuestaSim must analyze the VHDL files, a process known as *compilation*.

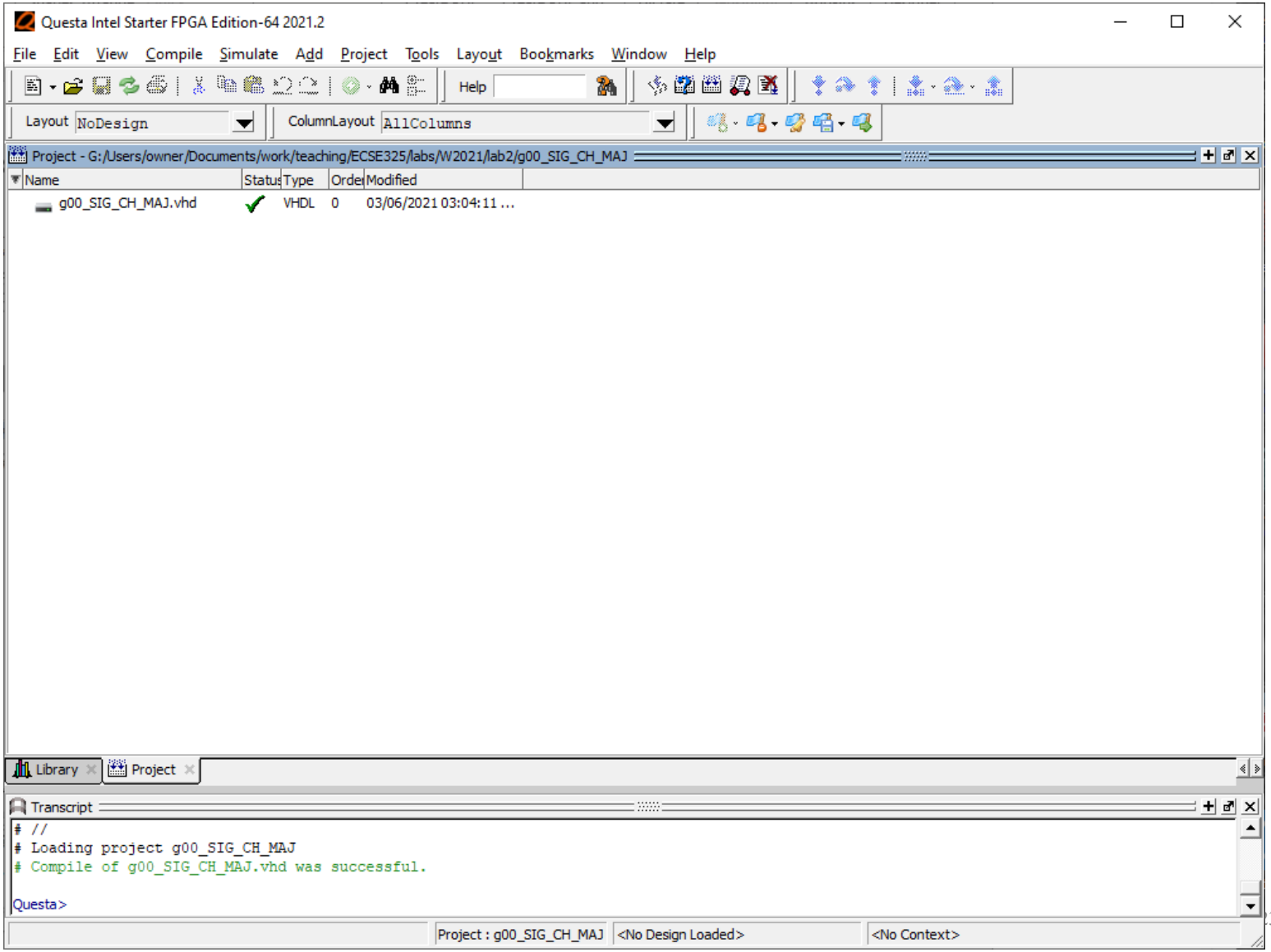
The compiled files are stored in a *library*. By default, this is named “work”. You can see this library in the “library” pane of the QuestaSim window.



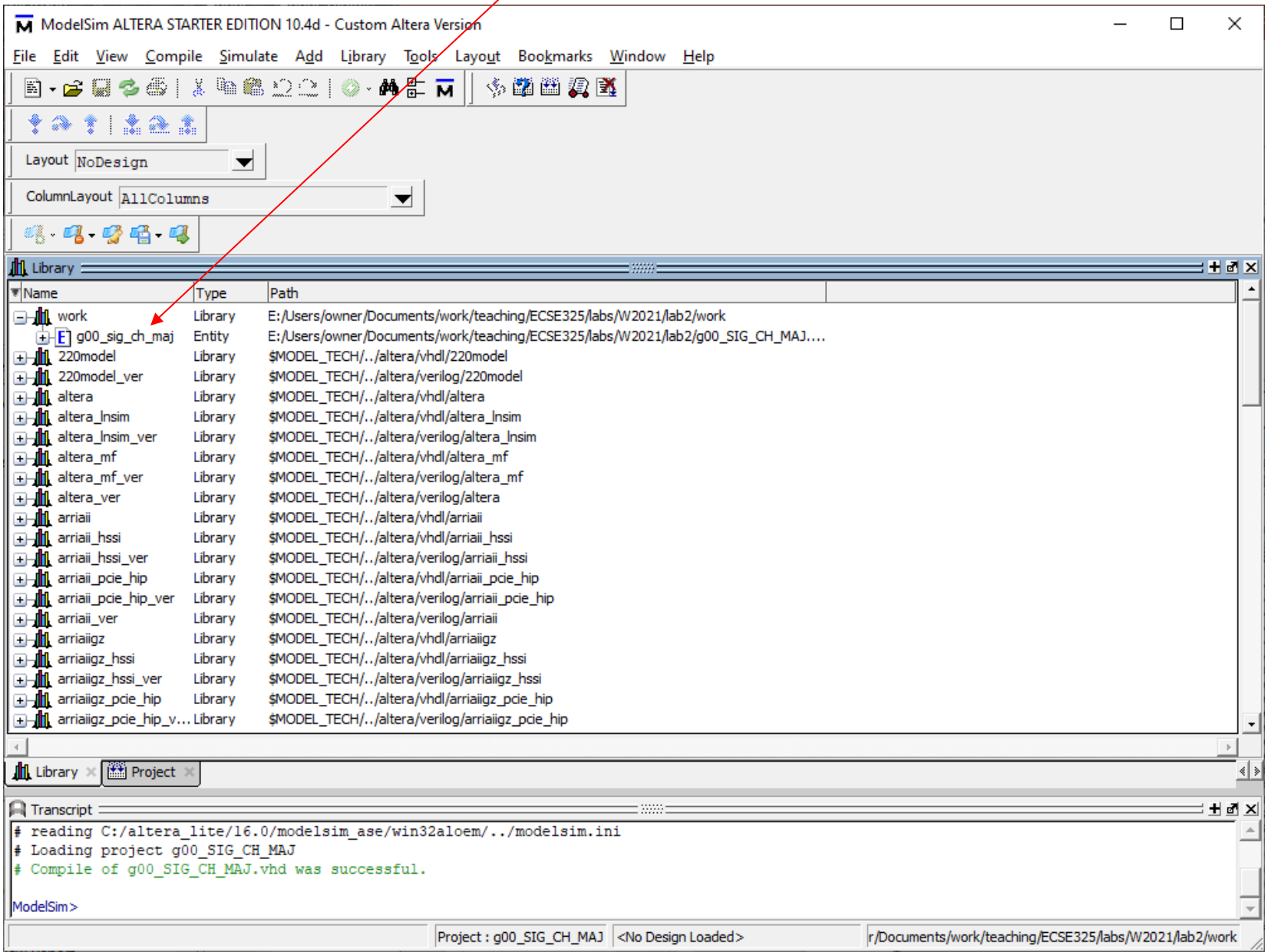
The question marks in the Status column in the Project tab indicate that either the files haven't been compiled into the project or the source file has changed since the last compilation. To compile the files, select **Compile > Compile All** or right click in the Project window and select **Compile > Compile All**.



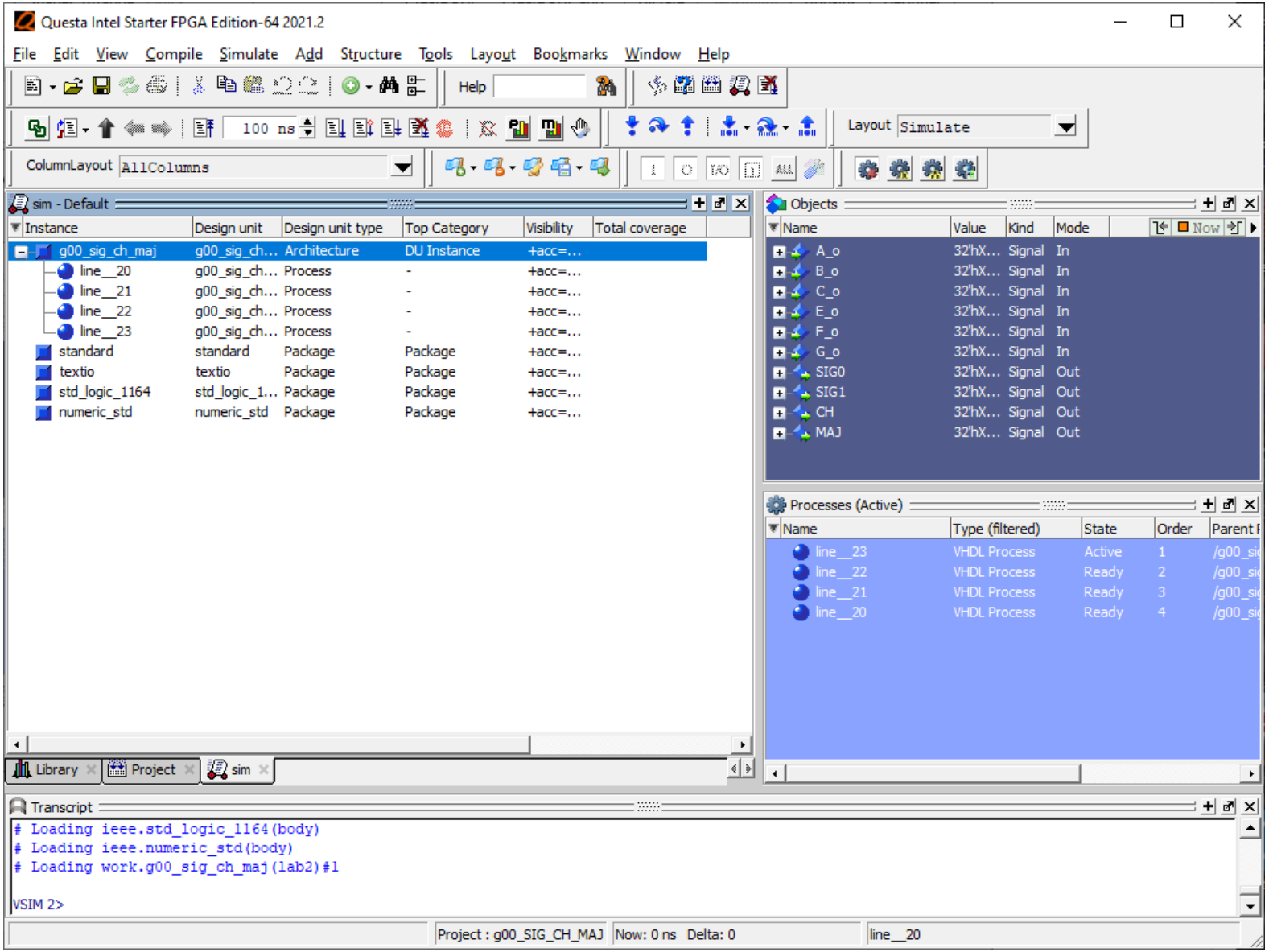
If the compilation is successful, the question marks in the Status column will turn to check marks, and a success message will appear in the Transcript pane.



The compiled vhdl files will now appear in the library “work”.



In the library window, double-click on gXX_Binary_BCD16. This will open some windows, which will be used in doing the simulation of the circuit.



You are not quite ready to start the simulation yet!

In the “Objects” window, the signals may have the value “UUUUUU”. This means that all the inputs are *undefined*. If you ran the simulation now, the outputs would also be undefined.

So, you need to have a means of setting the inputs to certain patterns, and of observing the outputs' responses to these inputs.

In QuestaSim, this is done by using a special VHDL entity called a ***Testbench***.

A testbench is some VHDL code that generates different inputs that will be applied to your circuit so that you can automate the simulation of your circuit and see how its outputs respond to different inputs.

It is important to realize that the testbench is only used in QuestaSim for the purposes of simulating your circuit. You will NOT synthesize the testbench into real hardware.

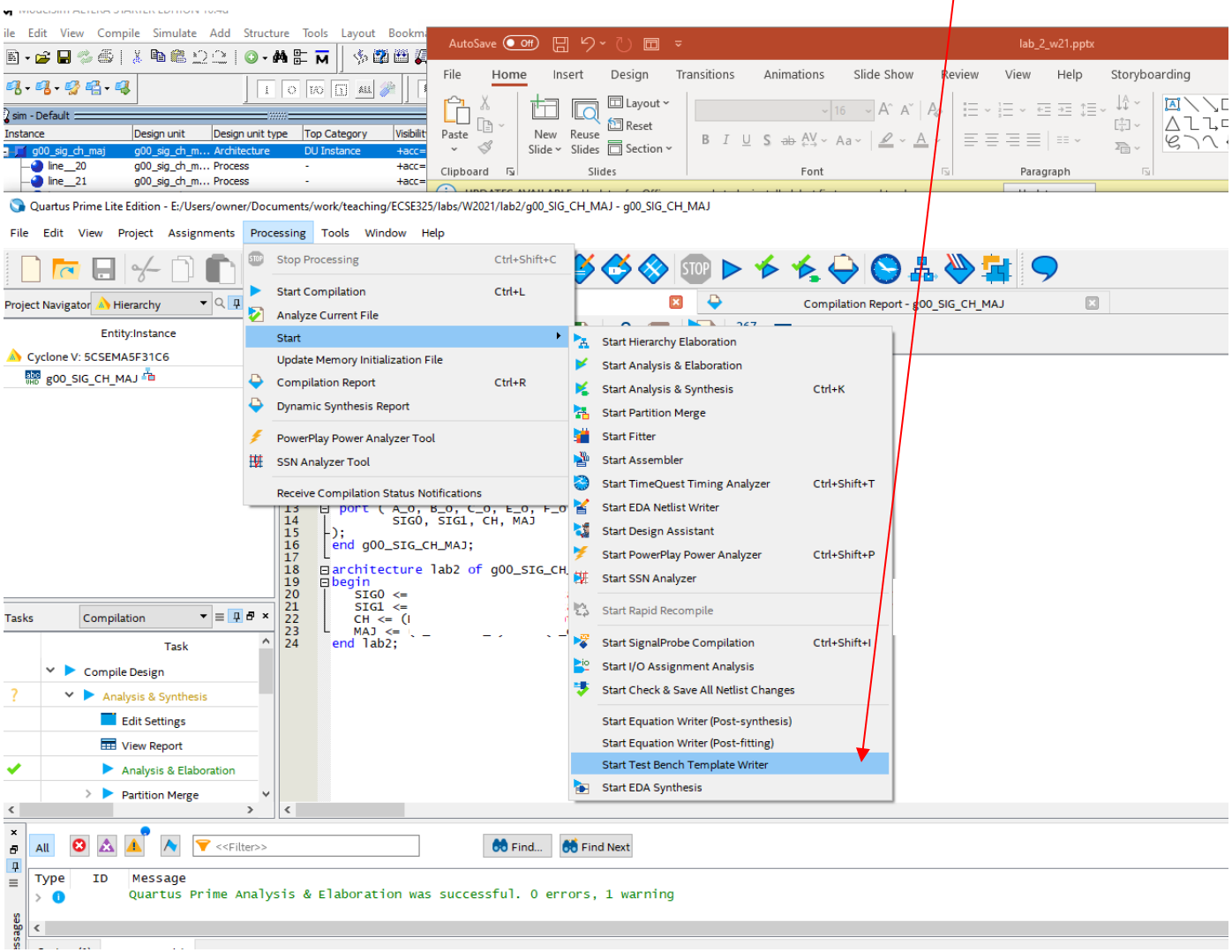
Because of its special purpose (and that it will not be synthesized), the testbench entity is unique in that it has NO inputs or outputs and uses some special statements that are only used in test benches. These special statements are not used when describing circuits that you will later *synthesize* to a FPGA.

The testbench contains a single *component instantiation statement* that inserts the module to be tested (in this case the gXX_Binary_BCD16 module), as well as some statements that describe how the test inputs are generated.

After you gain more experience, you will be able to write VHDL testbenches from scratch. However, Quartus has a convenient built-in process, called the ***Test Bench Writer***, which produces a VHDL template from your design that will get you started.

Go back to the Quartus program, making sure that you have the gXX_Binary_BCD16 project loaded.

Then, in the **Processing** toolbar item, select **Start/Start Test Bench Template Writer**



This will generate a VHDL file named gXX_Binary_BCD16.vht and place it in the simulation/QuestaSim directory. Open it up in Quartus. It will look something like this:

Quartus Prime Lite Edition - E:/Users/owner/Documents/work/teaching/ECSE325/labs/W2021/lab2/g00_SIG_CH_MAJ - g00_SIG_CH_MAJ

File Edit View Project Assignments Processing Tools Window Help

Project Navigator | Hierarchy

Entity:Instance

Cyclone V: 5CSEMA5F31C6

g00_SIG_CH_MAJ

Tasks | Compilation

Task

- Compile Design
  - Analysis & Synthesis
    - Edit Settings
    - View Report
  - Analysis & Elaboration
  - Partition Merge
  - Netlist Viewers
  - Design Assistant (Post-Map
  - I/O Assignment Analysis
  - Fitter (Place & Route)

g00_SIG_CH_MAJ.vhd | Compilation Report - g00_SIG_CH_MAJ

```
1   -- Copyright (C) 2016  Intel Corporation. All rights reserved.
2   -- Your use of Intel Corporation's design tools, logic functions
3   -- and other software and tools, and its AMPP partner logic
4   -- functions, and any output files from any of the foregoing
5   -- (including device programming or simulation files), and any
6   -- associated documentation or information are expressly subject
7   -- to the terms and conditions of the Intel Program License
8   -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9   -- the Intel MegaCore Function License Agreement, or other
10  -- applicable license agreement, including, without limitation,
11  -- that your use is for the sole purpose of programming logic
12  -- devices manufactured by Intel and sold by Intel or its
13  -- authorized distributors.  Please refer to the applicable
14  -- agreement for further details.
15
16  -- ***************************************************************************
17  -- This file contains a Vhdl test bench template that is freely editable to
18  -- suit user's needs .Comments are provided in each section to help the user
19  -- fill out necessary details.
20  -- ***************************************************************************
21  -- Generated on "03/06/2021 15:31:41"
22
23  -- Vhdl Test Bench template for design  :  g00_SIG_CH_MAJ
24  --
25  -- Simulation tool : ModelSim-Altera (VHDL)
26  --
27
28   LIBRARY ieee;
29   USE ieee.std_logic_1164.all;
30
31  ENTITY g00_SIG_CH_MAJ_vhd_tst IS
32  END g00_SIG_CH_MAJ_vhd_tst;
33  ARCHITECTURE g00_SIG_CH_MAJ_arch OF g00_SIG_CH_MAJ_vhd_tst IS
34  -- constants
35  -- signals
36   SIGNAL A_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
37   SIGNAL B_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
38   SIGNAL C_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
39   SIGNAL CH : STD_LOGIC_VECTOR(31 DOWNTO 0);
40   SIGNAL E_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
41   SIGNAL F_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
42   SIGNAL G_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
43   SIGNAL MAJ : STD_LOGIC_VECTOR(31 DOWNTO 0);
44   SIGNAL SIG0 : STD_LOGIC_VECTOR(31 DOWNTO 0);
45   SIGNAL SIG1 : STD_LOGIC_VECTOR(31 DOWNTO 0);
46  COMPONENT g00_SIG_CH_MAJ
47      PORT (
48      A_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
49      B_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
50      C_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
51      CH : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
52      E_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
53      F_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
54      G_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
55      MAJ : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
```

```
37   SIGNAL B_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
38   SIGNAL C_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
39   SIGNAL CH : STD_LOGIC_VECTOR(31 DOWNTO 0);
40   SIGNAL E_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
41   SIGNAL F_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
42   SIGNAL G_o : STD_LOGIC_VECTOR(31 DOWNTO 0);
43   SIGNAL MAJ : STD_LOGIC_VECTOR(31 DOWNTO 0);
44   SIGNAL SIG0 : STD_LOGIC_VECTOR(31 DOWNTO 0);
45   SIGNAL SIG1 : STD_LOGIC_VECTOR(31 DOWNTO 0);
46  COMPONENT g00_SIG_CH_MAJ
47      PORT (
48      A_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
49      B_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
50      C_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
51      CH : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
52      E_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
53      F_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
54      G_o : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
55      MAJ : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
56      SIG0 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
57      SIG1 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
58      );
59  END COMPONENT;
60   BEGIN
61      i1 : g00_SIG_CH_MAJ
62      PORT MAP (
63  -- list connections between master ports and signals
64      A_o => A_o,
65      B_o => B_o,
66      C_o => C_o,
67      CH => CH,
68      E_o => E_o,
69      F_o => F_o,
70      G_o => G_o,
71      MAJ => MAJ,
72      SIG0 => SIG0,
73      SIG1 => SIG1
74      );
75  init : PROCESS
76  -- variable declarations
77   BEGIN
78          -- code that executes only once
79   WAIT;
80  END PROCESS init;
81  always : PROCESS
82  -- optional sensitivity list
83  -- (        )
84  -- variable declarations
85   BEGIN
86          -- code executes for every event on sensitivity list
87   WAIT;
88  END PROCESS always;
89  END g00_SIG_CH_MAJ_arch;
90
```

All | <<Filter>> | Find... | Find Next

Type	ID	Message
		Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning

Messages | System (1) | Processing (6)

Note that the template will have already included the instantiation of the gXX_Binary_BCD16 component.

It also includes the skeletons of two “process” blocks, one labeled “*init*” and the other labeled “*always*”.

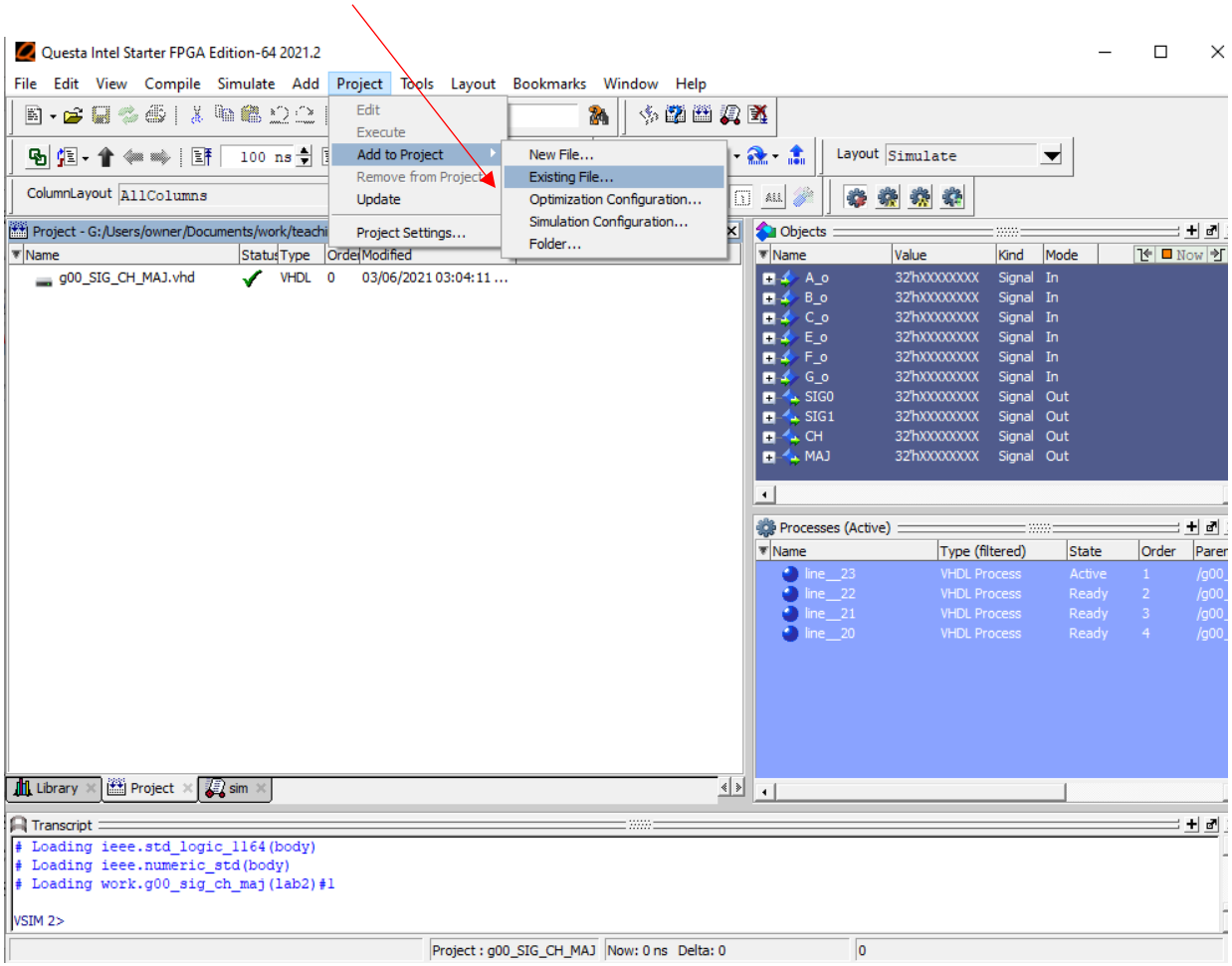
The init process block can be left blank for this lab but is usually used to specify initial signal values and other initial conditions for the simulation.

You should edit the “always” process block to suit your needs, so in this case it will be used to generate the input signal waveforms.

As a first test, enter the following code into the testbench file:

```
-- delete the init process block


always : PROCESS
-- optional sensitivity list
-- (        )
-- variable declarations
BEGIN
        -- code executes for every event on sensitivity list
                        bin <= “0000000000000000";

                        WAIT FOR 10 ns;

                        bin <= “1111111111111111";

                        WAIT FOR 5 ns;

                        bin <= “1110011001010011";

                        WAIT;
END PROCESS always;
```

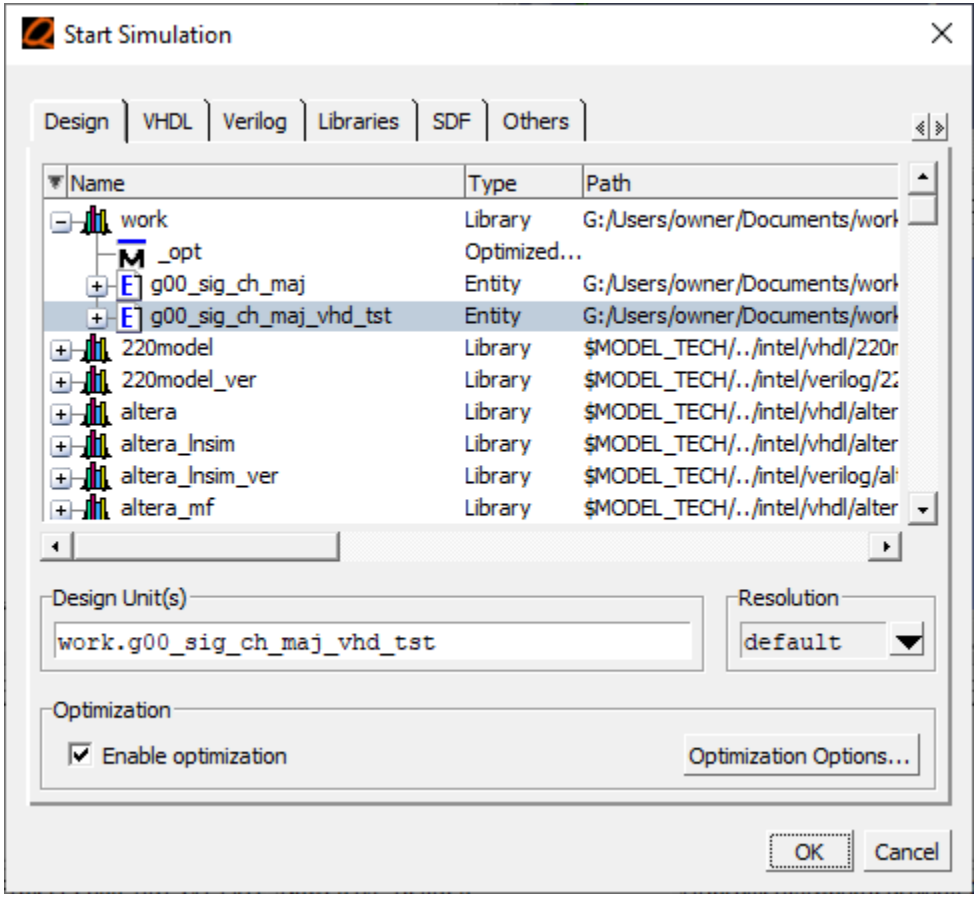
Once you have finished editing the testbench file, you need to add it to the project in QuestaSim (make sure you are in the “Project” pane):



Once the testbench file has been added to the project, you should select the testbench file in the Project pane and click on Compile Selected from the Compile toolbar item. This will compile the testbench file.

Now everything is ready for you to actually run a simulation!

Select "Start Simulation" from the Simulate toolbar item in the QuestaSim program. The following window will popup:



Select the gNN_Binary_BCD16_tst entity (expand the "work" item to find this) and click on OK

The QuestaSim window should now look like this. Enter a value of 20ns into the simulation length window.

Questa Intel Starter FPGA Edition-64 2021.2

File Edit View Compile Simulate Add Process Tools Layout Bookmarks Window Help

Help

20 ns

Layout Simulate

ColumnLayout AllColumns

sim - Default

Instance	Design unit	Design unit type	Top Category	Visibility	Total coverage
g00_sig_ch_maj_v...	g00_sig_ch...	Architecture	DU Instance	+acc=...	
standard	standard	Package	Package	+acc=...	
textio	textio	Package	Package	+acc=...	
std_logic_1164	std_logic_1...	Package	Package	+acc=...	
numeric_std	numeric_std	Package	Package	+acc=...	

Objects

Name	Value	Kind	Mode
A_o	32'hXXXXXXXX	Signal	Internal
B_o	32'hXXXXXXXX	Signal	Internal
C_o	32'hXXXXXXXX	Signal	Internal
CH	32'hXXXXXXXX	Signal	Internal
E_o	32'hXXXXXXXX	Signal	Internal
F_o	32'hXXXXXXXX	Signal	Internal
G_o	32'hXXXXXXXX	Signal	Internal
MAJ	32'hXXXXXXXX	Signal	Internal
SIG0	32'hXXXXXXXX	Signal	Internal
SIG1	32'hXXXXXXXX	Signal	Internal

Processes (Active)

Name	Type (filtered)	State	Order	Parent Path
always	VHDL Process	Active	1	/g00_sig_ch_maj...
line__23	VHDL Process	Ready	2	/g00_sig_ch_maj...
line__22	VHDL Process	Ready	3	/g00_sig_ch_maj...
line__21	VHDL Process	Ready	4	/g00_sig_ch_maj...
line__20	VHDL Process	Ready	5	/g00_sig_ch_maj...

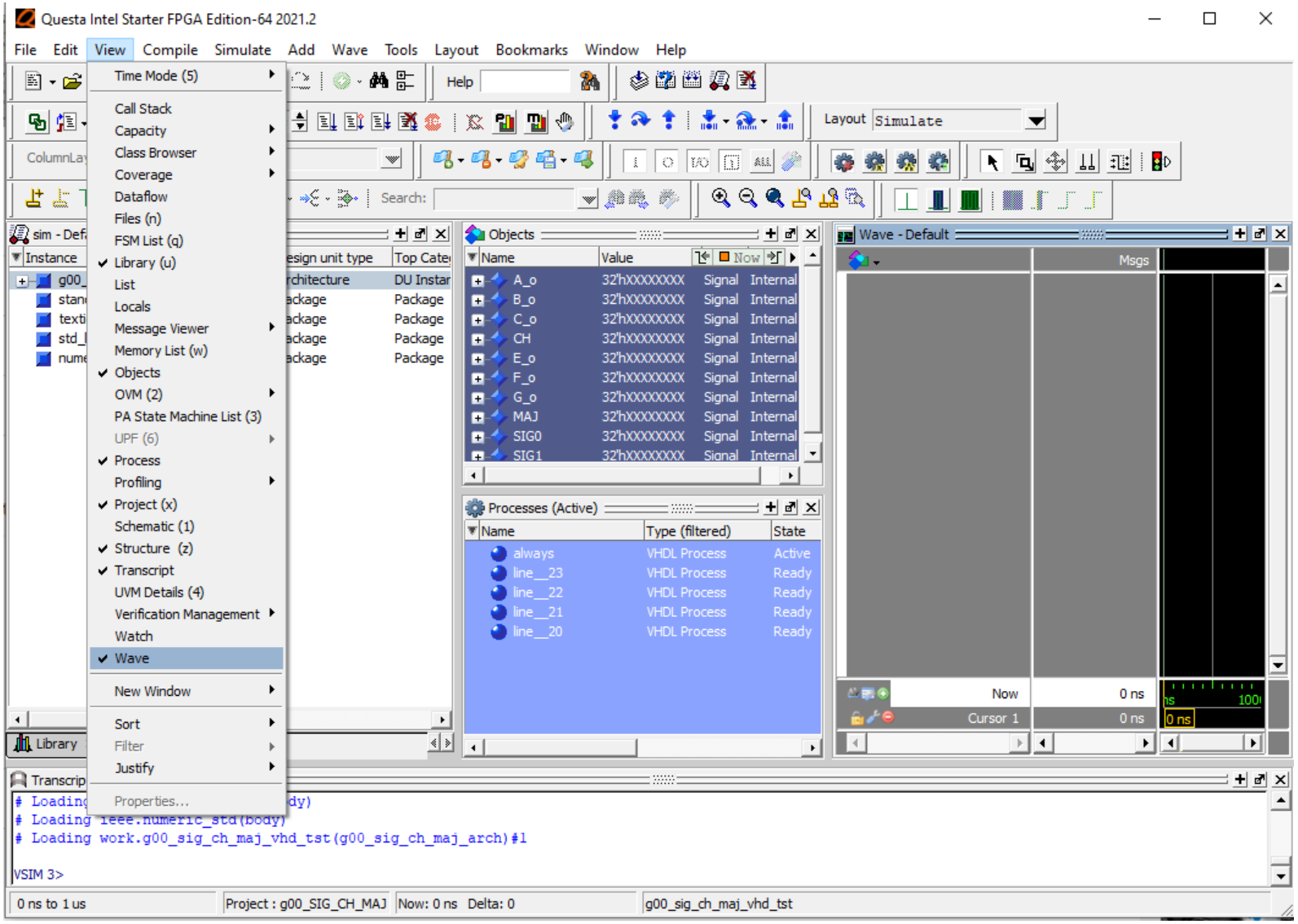
Library Project sim

Transcript

```
# Loading ieee.std_logic_1164(body)
# Loading ieee.numeric_std(body)
# Loading work.g00_sig_ch_maj_vhd_tst(g00_sig_ch_maj_arch)#1

VSIM 3>
```

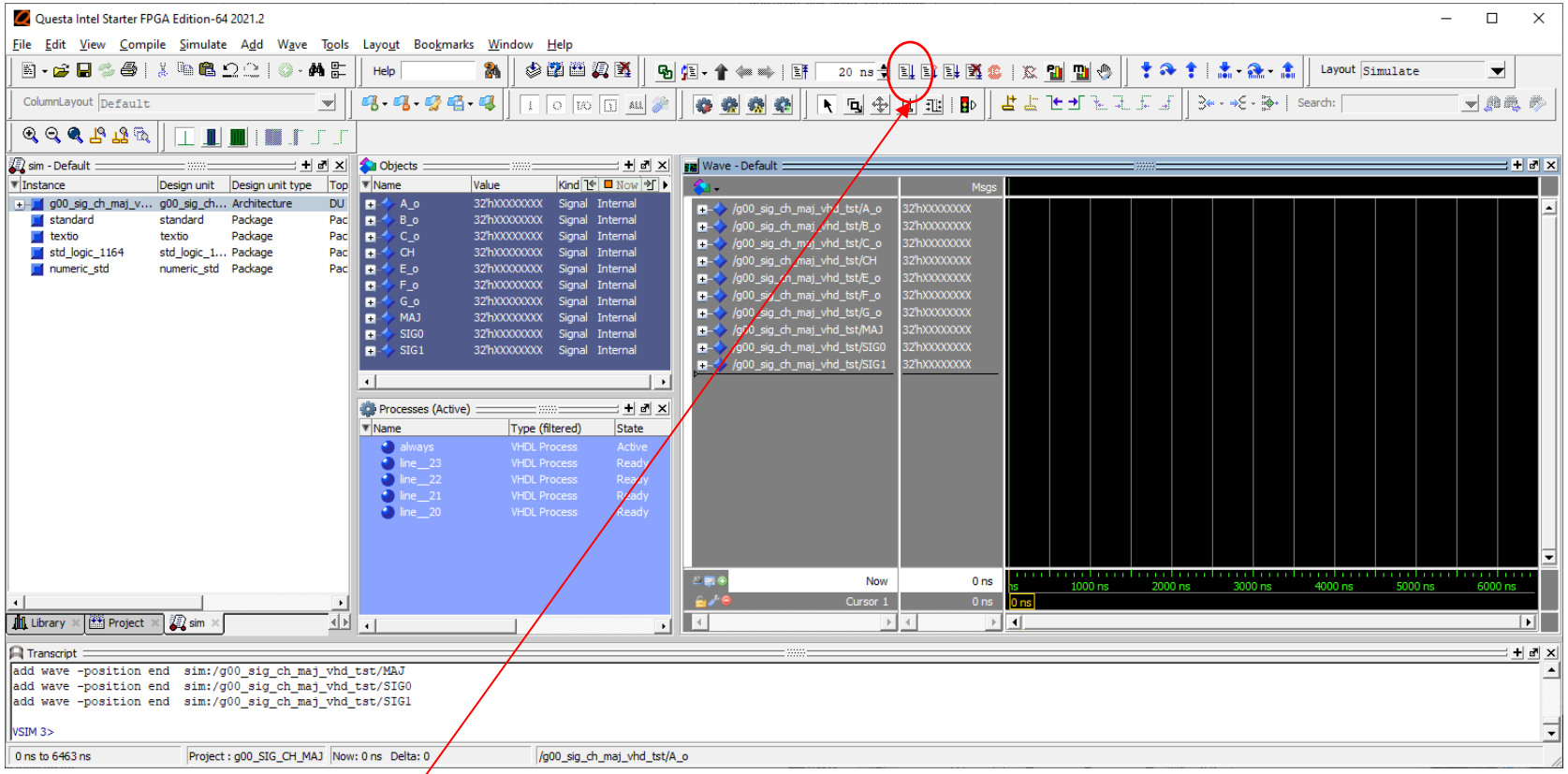
Project : g00_SIG_CH_MAJ | Now: 0 ns Delta: 0 | sim:/g00_sig_ch_maj_vhd_tst

To display waveforms, open the wave window by selecting it in the View menu.



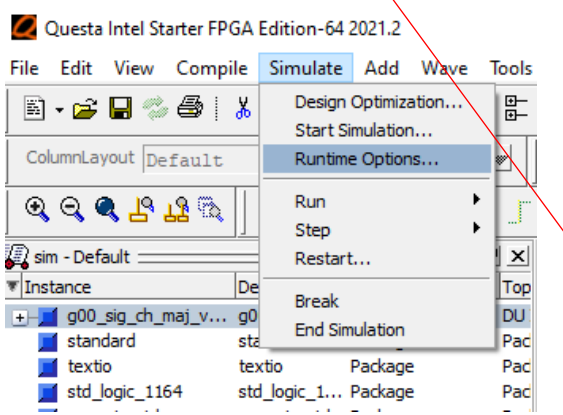
At first, the “Wave” window will not have any signals in it. You can drag signals from the “Objects” window by clicking on a signal, holding down the mouse button, and dragging the signal over to the Wave window. Do this for all the signals.

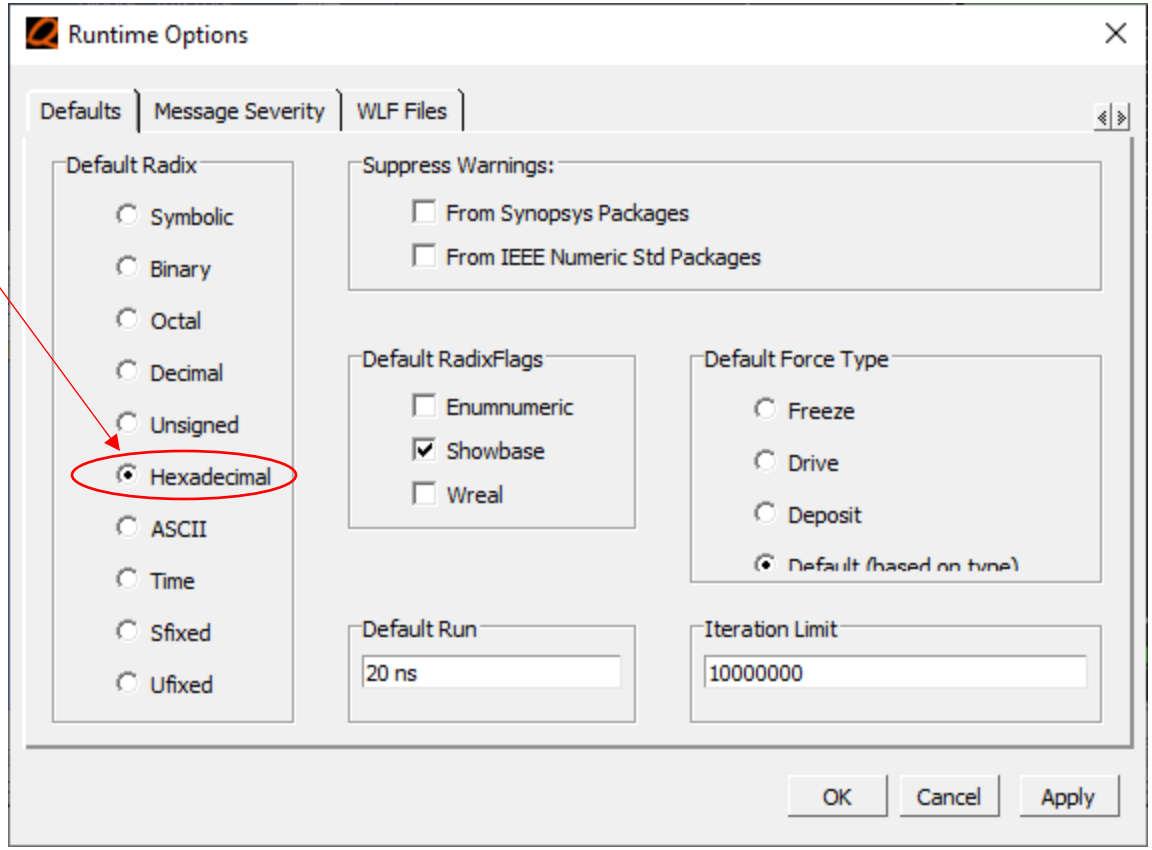
The Wave window will now look like this:



Now, to actually run the simulation, click on the “Run” icon in the toolbar (or press the F9 key).

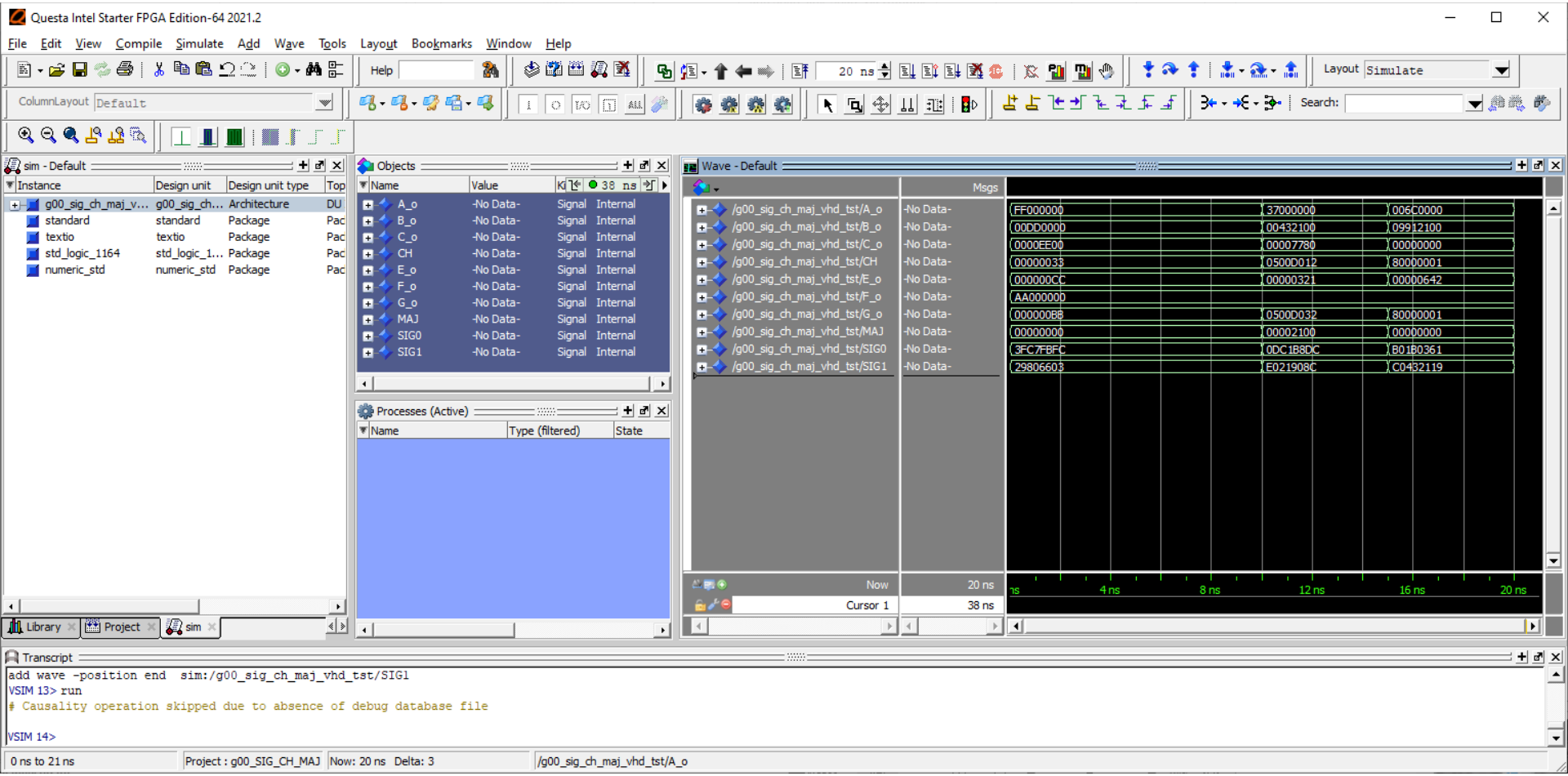
Since we are working with a 16-bit vector input signal and multiple 4-bit output signals, it is inconvenient to view their values as binary vectors. Instead, set the "Runtime Options" from the Simulate menu and set the "Default Radix" to Hexadecimal.



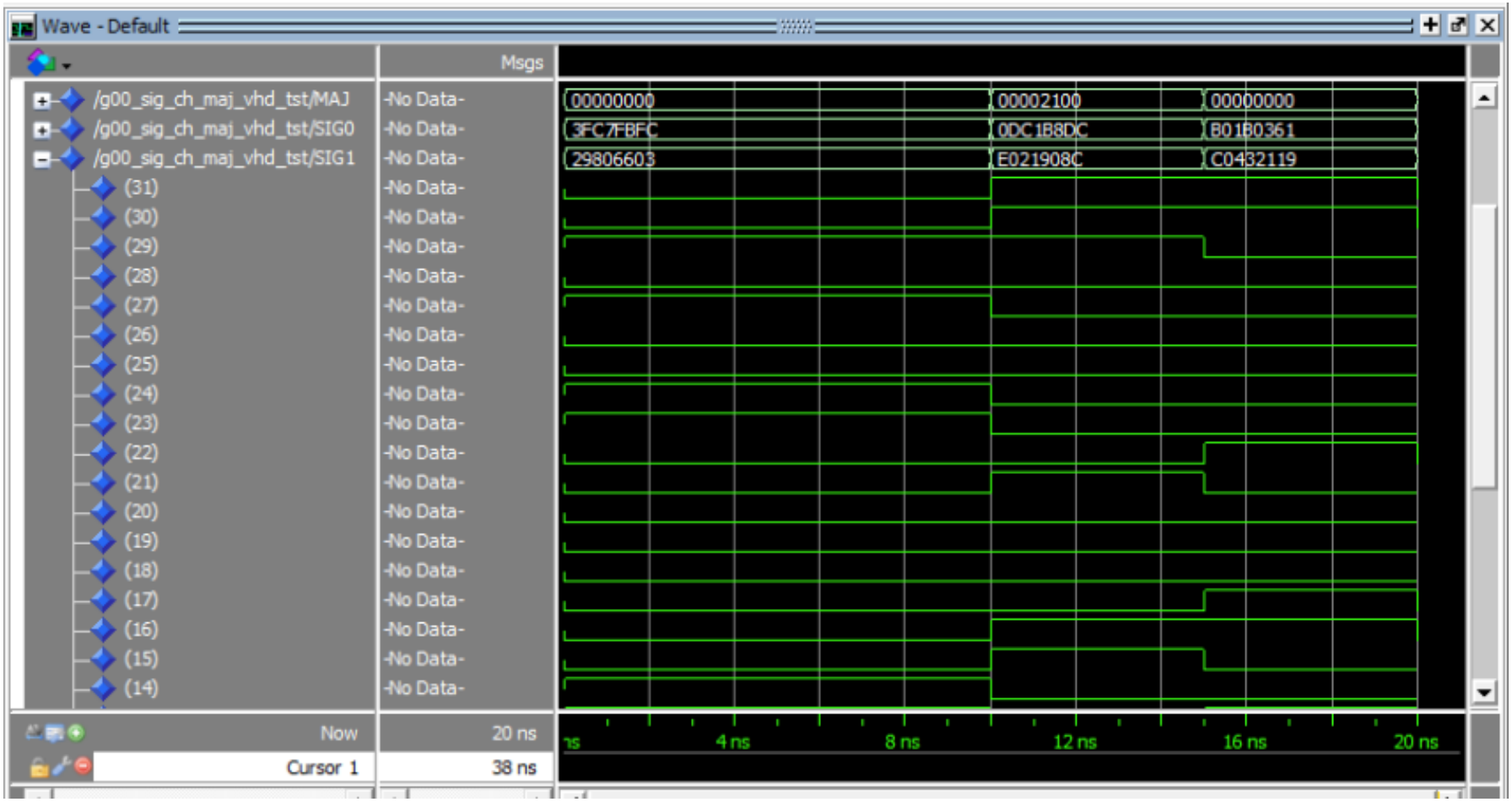


You should now see a set of waveforms with numeric (hex) values displayed (you can right-click in the right-hand pane and select “Zoom Full” to see the entire time range).

Your waveforms will look different than the ones shown here, as this is just an illustrative example.



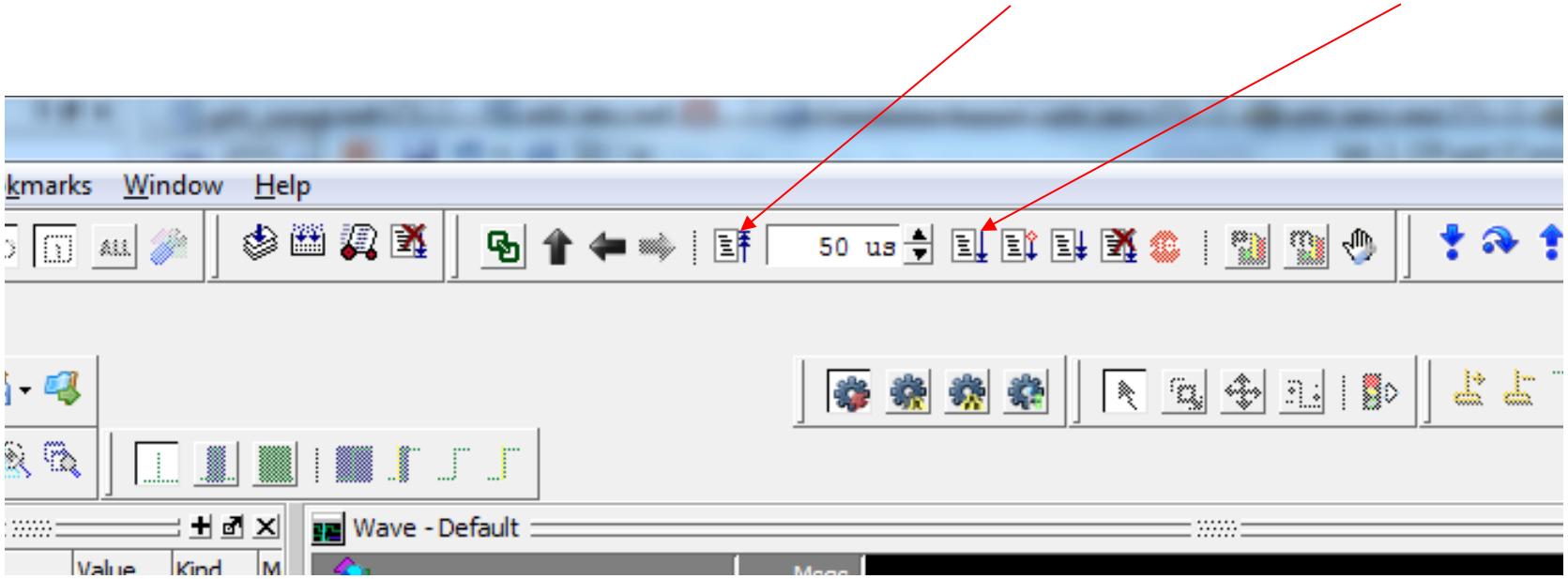
To see the individual bits in a vector, click on the "+" next to the signal name in the left-hand column.



If you get an incorrect output waveform, you will have to go back and look at your design and correct the errors.

Then you will have to re-run the compilation of the changed files in QuestaSim.

Finally, to rerun the simulation, first click on the “Restart” button, then click on the “Run” button.



Further Testing of the Project using QuestaSim

The simulation you ran in the previous part of the lab just had a couple of input signal transitions and did not test all possible input patterns.

There are 2^16 possible input patterns, so one could modify the testbench file to generate all possible test patterns. But you don't want to look at all 64K+ patterns.

Instead, generate a set of 32 different test patterns, evenly spaced in steps of 2048, by using a FOR LOOP, with a WAIT of 5 nsec per loop.

Rerun the simulation with this amended testbench. You will have to change the run time of the simulation, since you have more transitions.

Capture a screenshot of the resulting wave display for your report. Make sure that hexadecimal values are shown for the signal vector values.

Writeup the Lab Report .

Write up a short report describing the ***gXX_DM74185*** and ***gXX_Binary_BCD16*** circuits that you designed and simulated in this lab. This report should be submitted in pdf format.

The report must include the following items:

- A header listing the group number, the names and student numbers of each group member.
- A title, giving the name (e.g. ***gXX_Binary_BCD16***) of the main circuit.
- A description of the circuits' functions, listing the inputs and outputs.
- The VHDL descriptions of the circuits.
- The final version of the testbed file.
- A screenshot of the simulation results for the final simulation run. This should show clearly the values of the signals over the simulation time interval.

The report is due one weeks after the end of the lab period (i.e. on March 21), at 11:59 PM.

Submit the Lab Report to myCourses .

The lab report, and all associated design files (by design files, I mean the vhdl and testbench (.vhd and .vht) files) must be submitted, as an assignment to the myCourses site. Only one submission need be made per group (both students will receive the same grade).

Combine all of the files that you are submitting into one *zip* file and name the zip file gXX_LAB_1.zip (where XX is your group number).