

# ECSE-325

# Digital Systems

**Lab #1** – *Simulation and Timing Analysis*    Winter 2023

# Introduction

---

In this lab you will learn the basics of digital simulation using the *ModelSim* simulation program as well as the use of the Timequest Timing Analyzer.

# 1. Design of a ARCCOS Circuit

For the course project you will need a circuit that will take in an unsigned 8-bit value,  $X$ , and provide a 10-bit unsigned output, *ANGLE*, which represents  $10 \cdot (\arccosine(X))$ , in units of  $10^{\text{th}}$  of a degree.

To do this, you will use an approximation, based on the Taylor's Series expansion:

$$\cos^{-1}(x) \sim \frac{\pi}{2} - x - \frac{x^3}{6} - \frac{x^5}{40}; \text{ in radians, for } x \in (0,1)$$

We need to convert this to a form where the input  $X$  is scaled to range from 0-255 (8 bits) and the output is scaled to be in  $10^{\text{th}}$  of a degree. Doing this we get an expression that we can implement using unsigned integer operations:

$$\cos^{-1}(X) \sim 900 - \frac{X}{2^9} \left[ 1144 + \frac{X^2}{2^{16}} \left[ 191 + 86 \frac{X^2}{2^{16}} \right] \right]$$

(you should try to derive this formula yourself...)

The approximation is very poor as  $X$  approaches 256 but is better for lower values. If we take  $X=128$  and truncate the results of all operations to integer values, the above formula gives 601, which is close to the exact value of 600.

You should break the computation into a series of steps:

1.  $X2 = X * X$
2.  $P1 = [86 * X2] / 2^{16}$
3.  $S1 = 191 + P1$
4.  $P2 = [S1 * X2] / 2^{16}$
5.  $S2 = 1144 + P2$
6.  $P3 = [S2 * X] / 2^9$
7.  $ANGLE = 900 - P3$

Perform the division by powers of 2 with right shifts, which can be done easily just by discarding the appropriate number of LSBs.

Implement these operations in a single clocked process block.

## VHDL Description of the ARCCOS 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 ARCCOS operation.

Name this file gNN\_ARCCOS.vhd, where NN is your group number.

```
--  
-- entity name: gNN_ARCCOS  (replace "NN" by your group's number)  
--  
-- Version 1.0  
-- Authors: (list the group member names here)  
-- Date: March ??, 2023 (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;  
use ieee.numeric_std.all; -- needed since you are using unsigned numbers  
  
entity gNN_ARCCOS is  
  port ( X : in std_logic_vector(7 downto 0);  
         CLOCK : in std_logic;  
         ANGLE : out std_logic_vector(9 downto 0);  
end gNN_ARCCOS;
```

## Simulation of the ARCCOS circuit using ModelSim

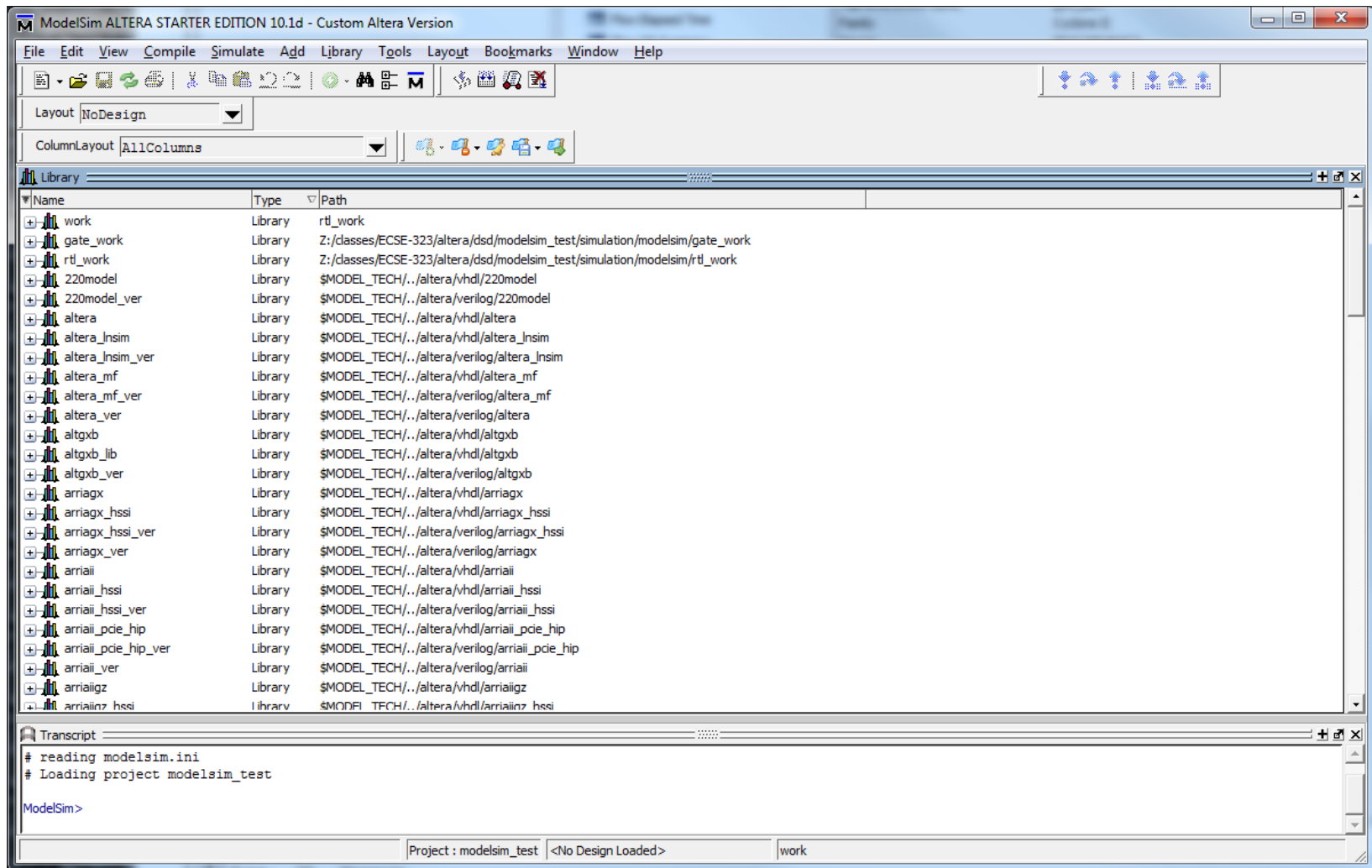
In this course, we will be using the *Modelsim* simulation software, created by the company Mentor Graphics (we will use a version of it specific to Quartus, called Modelsim-Altera).

The Modelsim 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.**

Run the Quartus program and create a new project called gNN\_ARCCOS (where “NN” is your group number). As in lab #0 set the device to Cyclone V 5CSEMA5F31C6

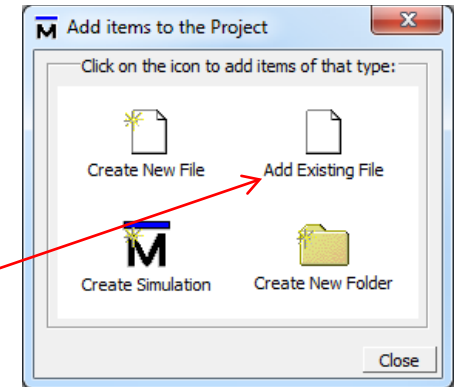
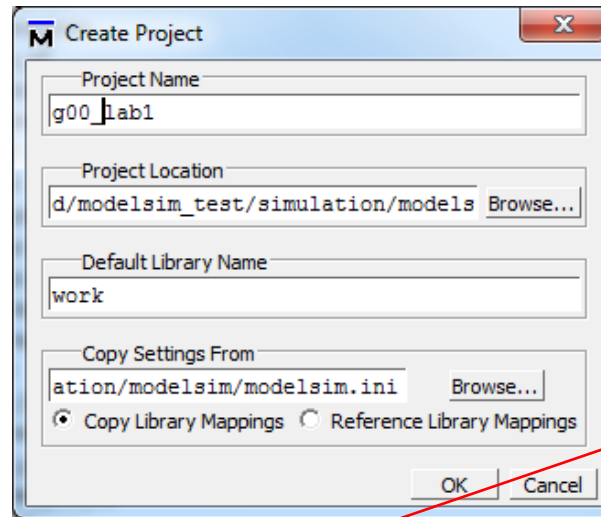
In Quartus, add your VHDL file to this new project and run “Processing/Start/Start Analysis and Elaboration” on the VHDL file (we don’t need to do the synthesis/fitting steps for now) and correct any errors you might have.

Next, double-click on the ModelSim desktop icon to startup the ModelSim program. A window somewhat like the one shown below will appear.



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

Click OK.

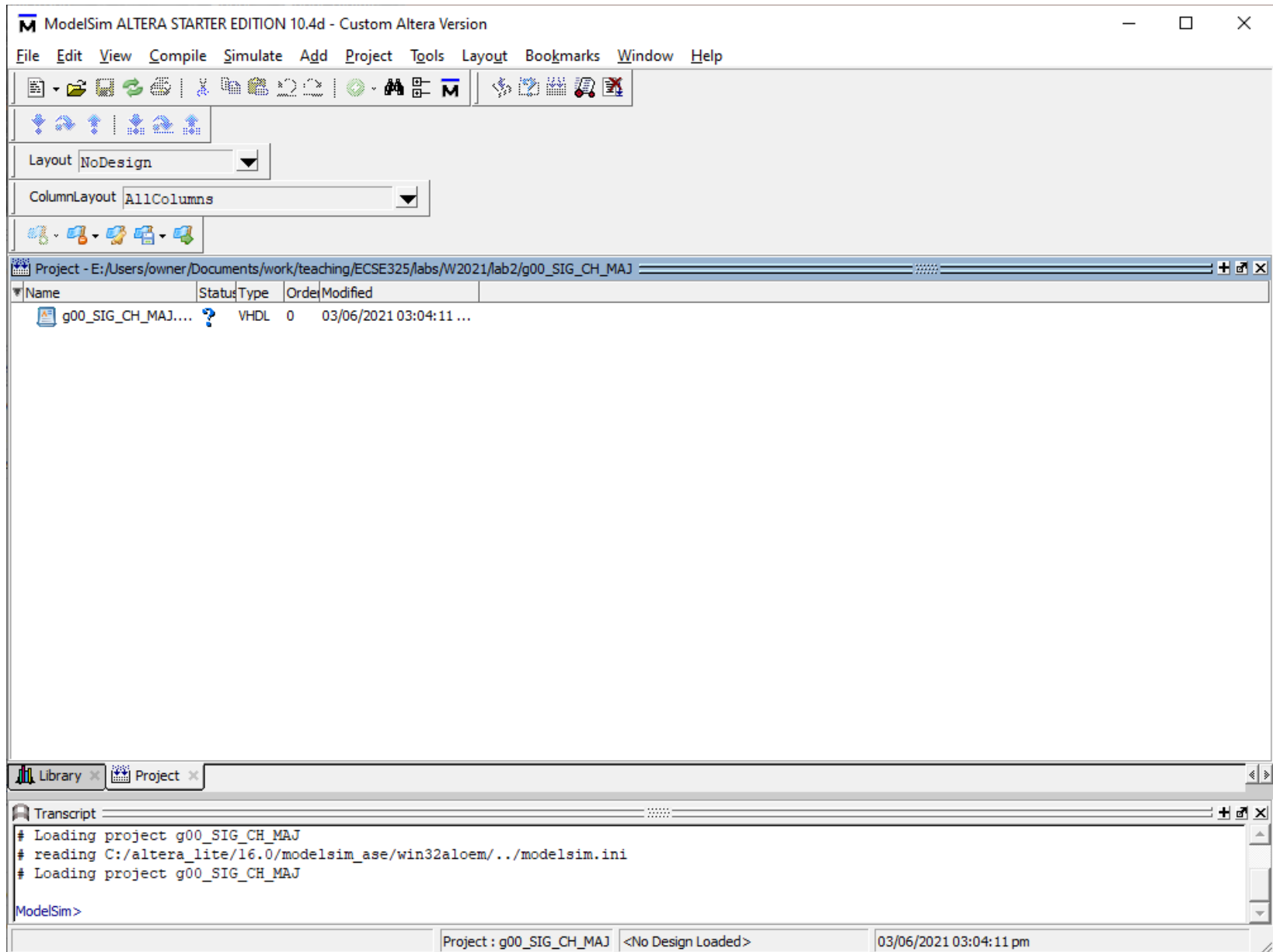


Another dialog box will appear, allowing you to add files to the project. Click on “Add Existing File” and select the VHDL file that was generated earlier (gNN\_ARCCOS.vhd). You can also add files later.

*(note that the various entries in the popup windows will be different for your lab – these are just example images from previous years)*

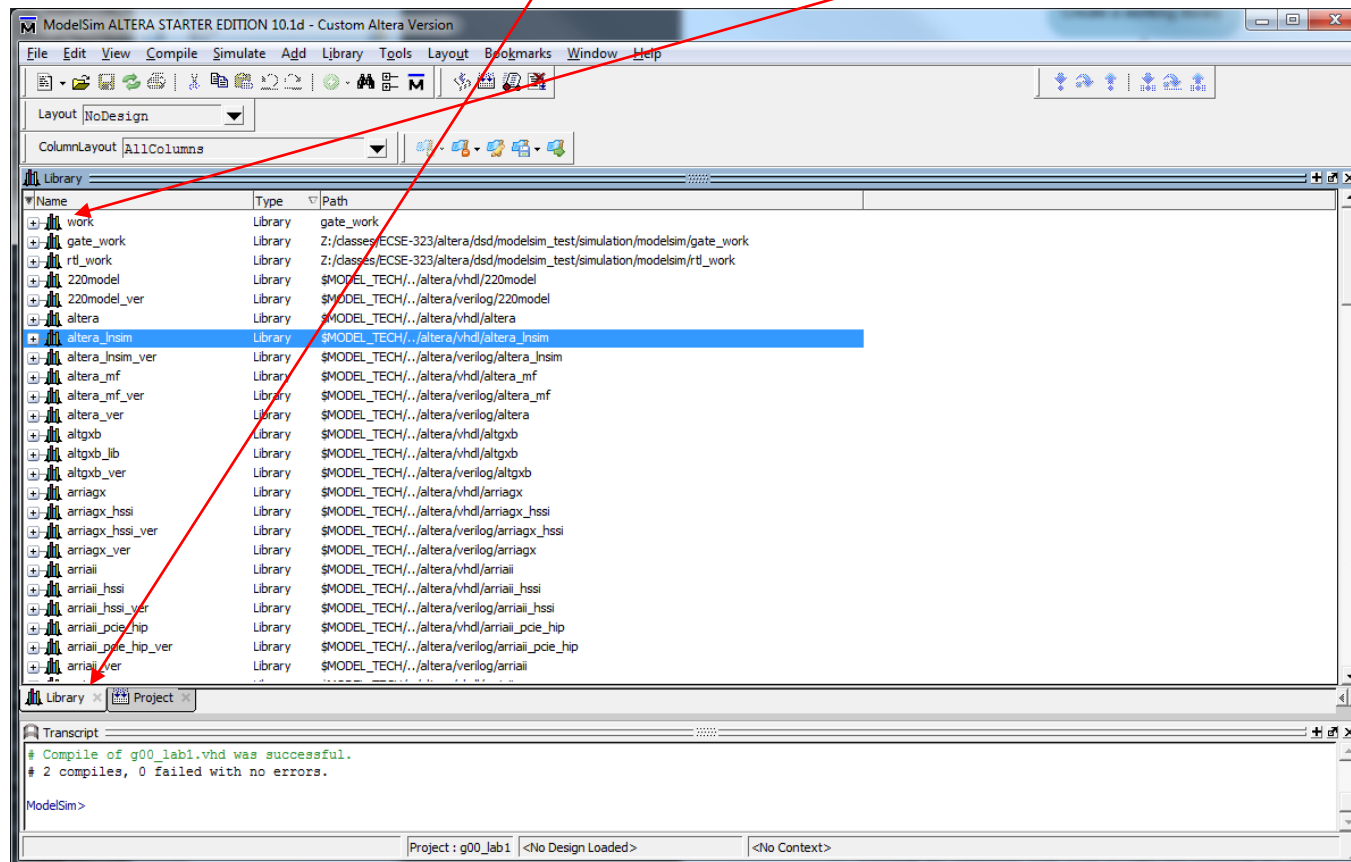


The ModelSim window will now show this vhd1 file in the Project pane.

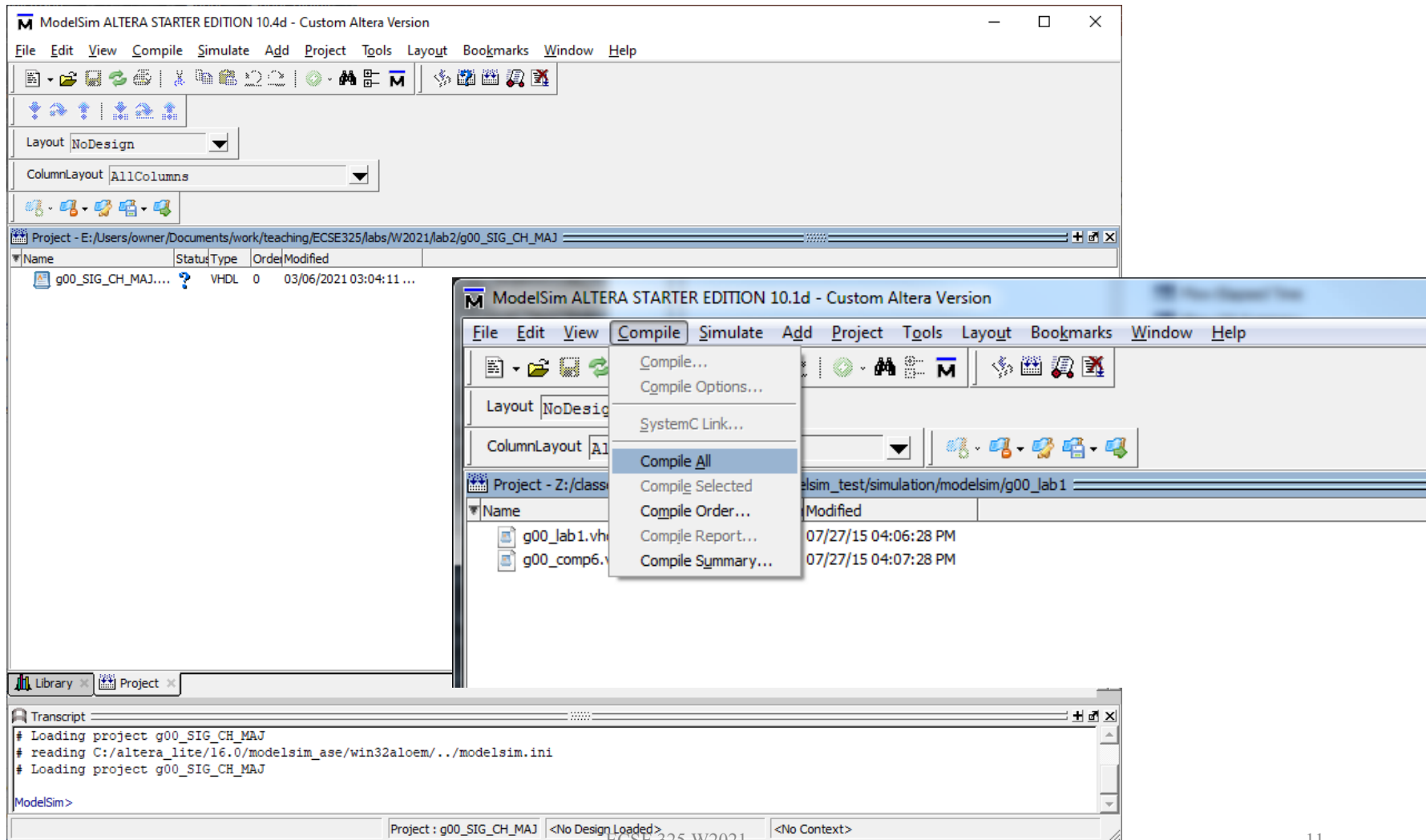


In order to simulate the design, ModelSim must analyze the VHDL files, a process known as *compilation*.

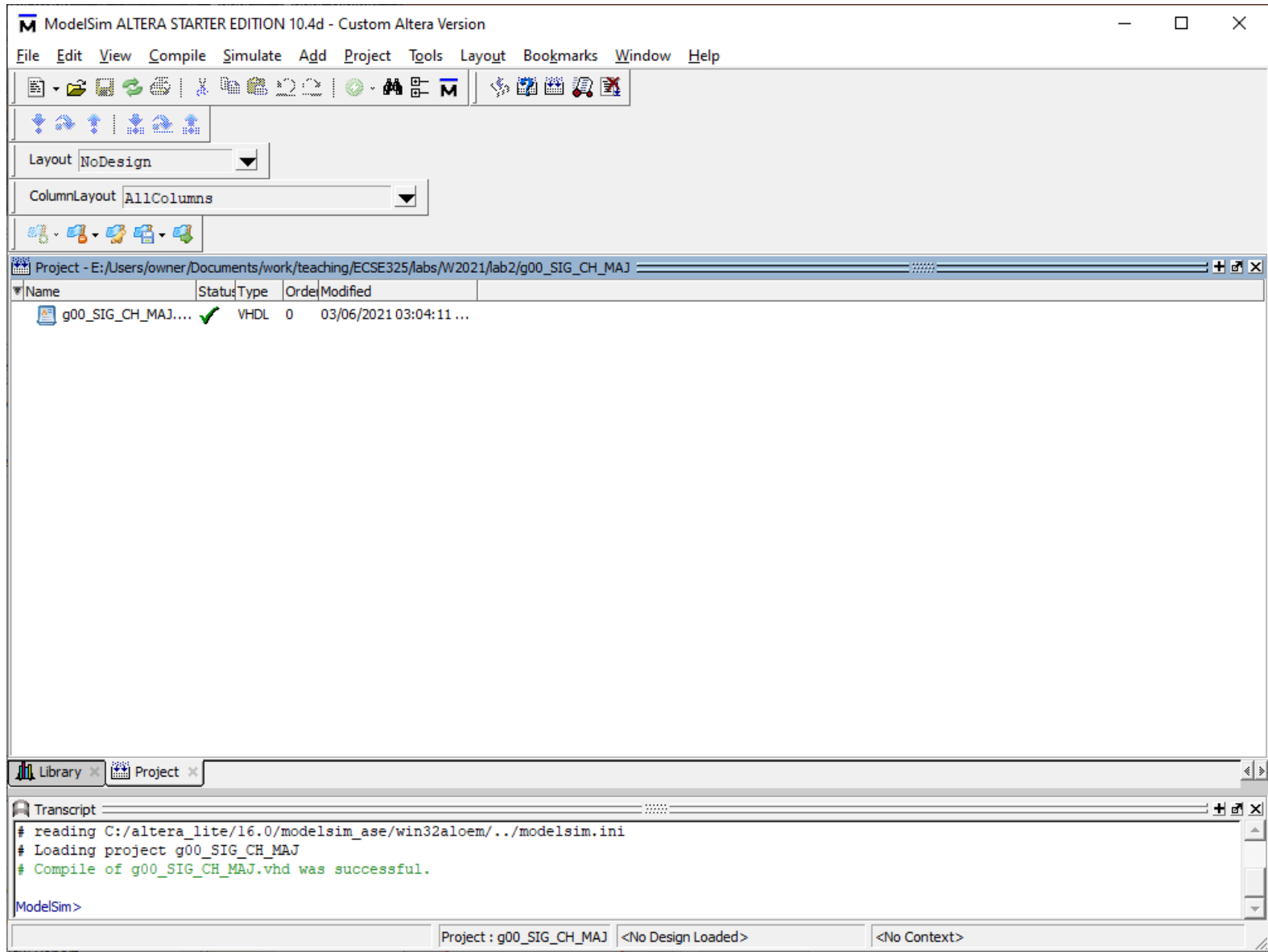
The compiled files are stored in a *library*. By default, this is named “work”. You can see this library in the “library” pane of the ModelSim window.



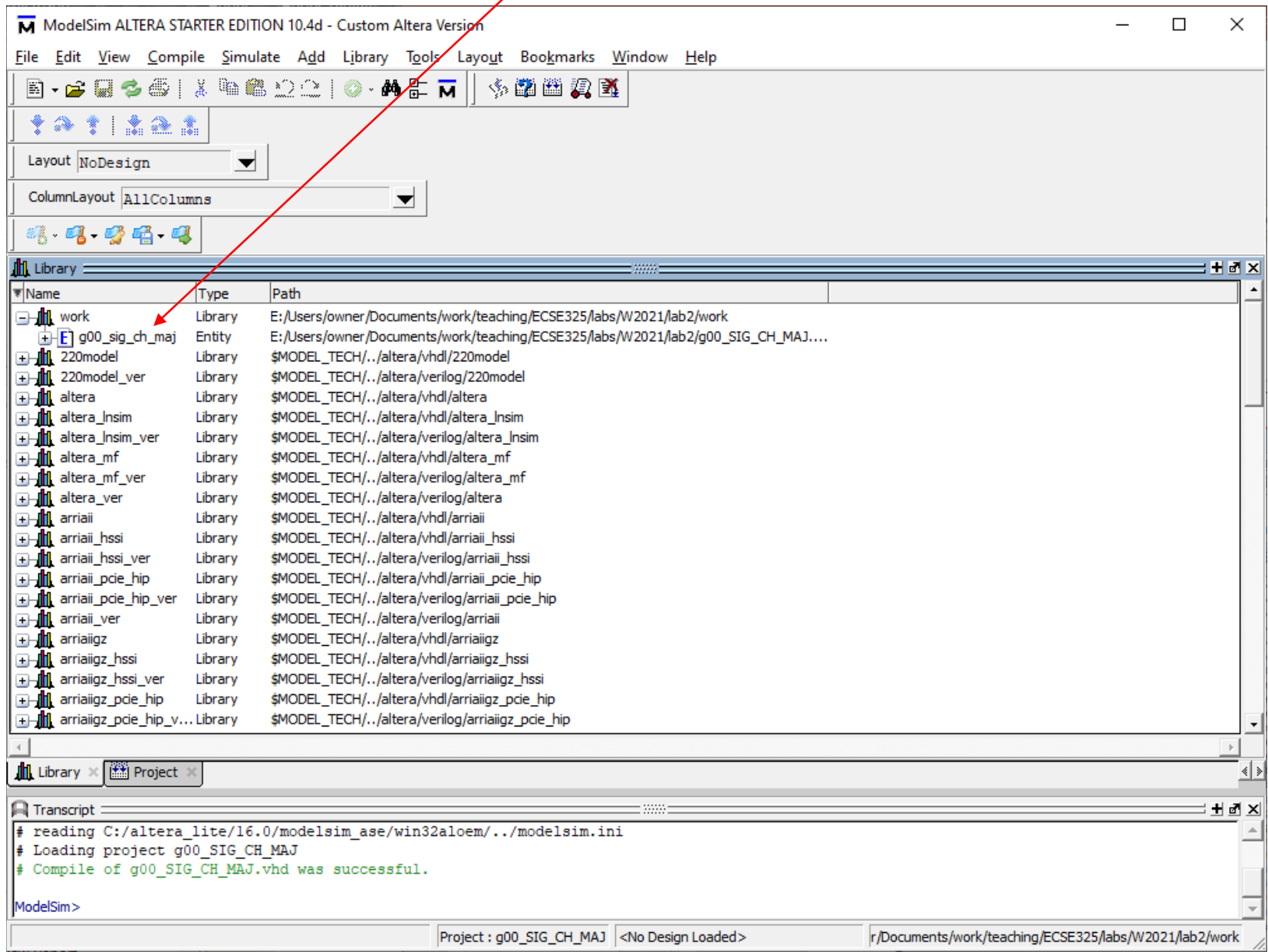
The question marks in the Status column in the Project tab indicate that either the files haven't been compiled into the project or the source file has changed since the last compilation. To compile the files, select **Compile > Compile All** or right click in the Project window and select **Compile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



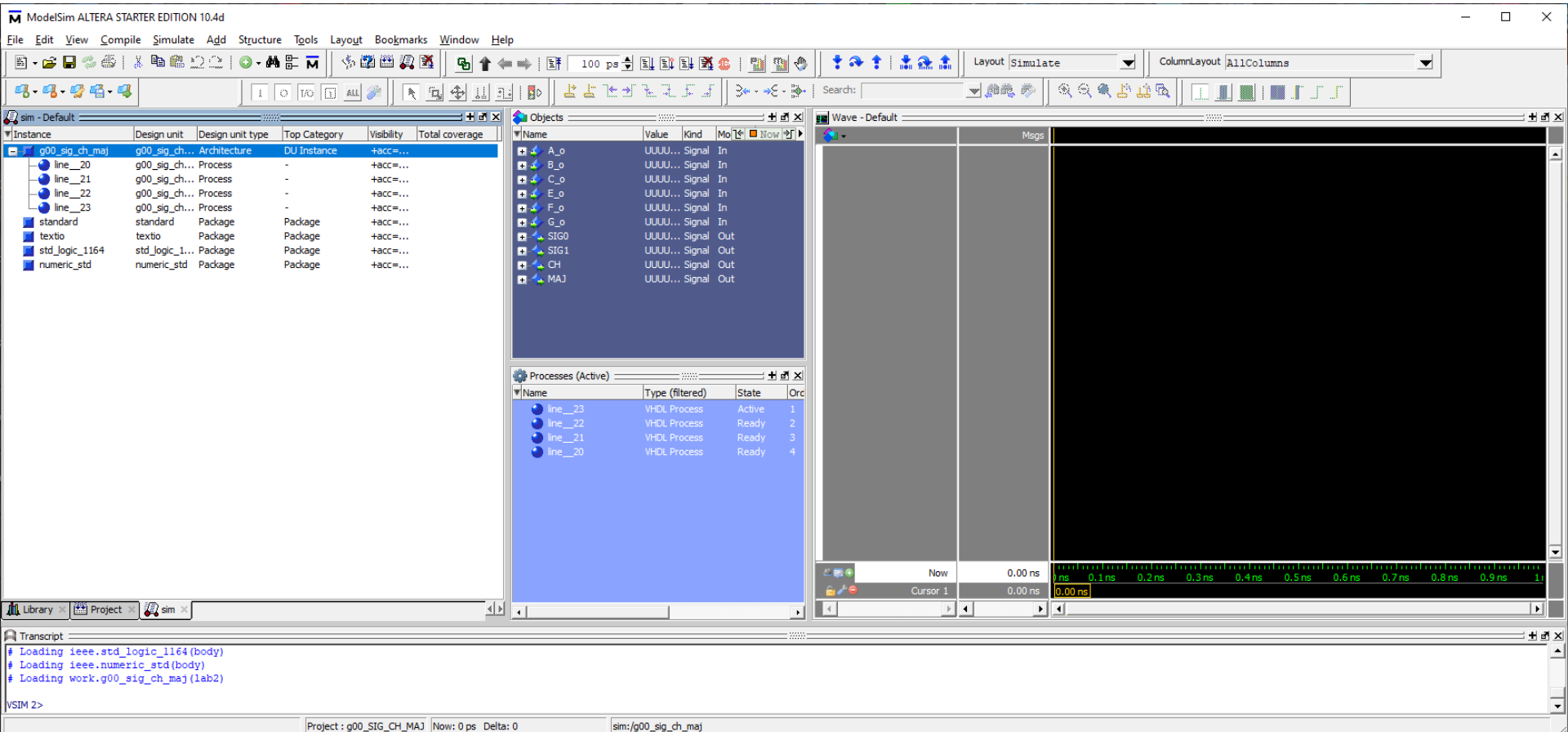
If the compilation is successful, the question marks in the Status column will turn to check marks, and a success message will appear in the Transcript pane.



The compiled vhd1 files will now appear in the library “work”.



In the library window, double-click on gNN\_ARCCOS. This will open a bunch of windows which will be used in doing the simulation of the circuit.



# You are not quite ready to start the simulation yet!

Notice that, in the “Objects” window, the signals all have the value “UUUUUU”. This means that all of 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 Modelsim, 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 Modelsim 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.

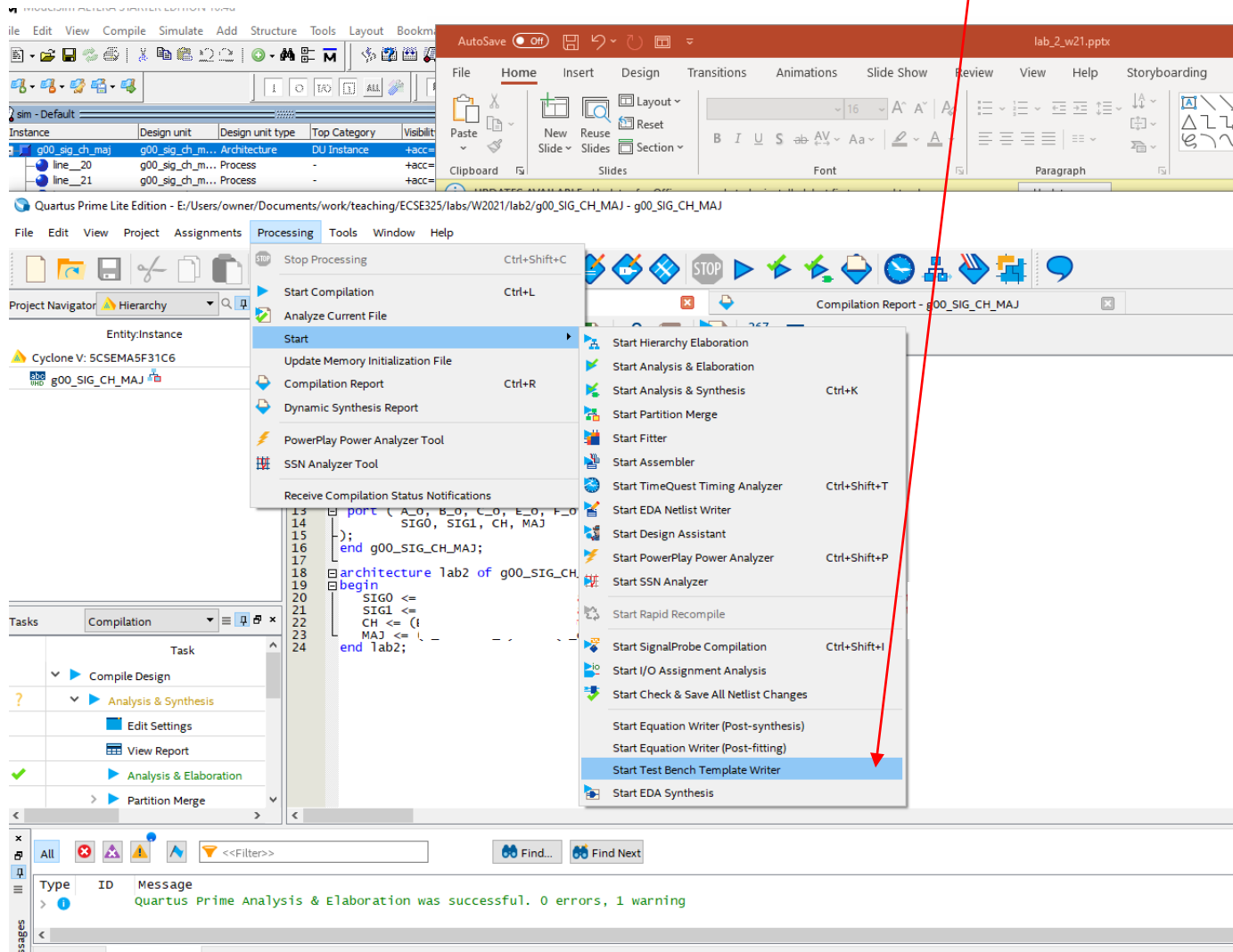
The testbench contains a single *component instantiation statement* that inserts the module to be tested (in this case the gNN\_ARCCOS module), as well as some statements that describe how the test inputs are generated.

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 gNN\_ARCCOS project loaded.

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



This will generate a VHDL file named gNN\_ARCCOS.vht and place it in the simulation/modelsim 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

g00\_SIG\_CH\_MAJ

Project Navigator Hierarchy

Entity:Instance

Cyclone V: 5CSEMA5F31C6

g00\_SIG\_CH\_MAJ

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 vhd 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);
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

```

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)

Messages

All

Find...

Find Next

Type ID Message

Quartus Prime EDA Netlist writer was successful. 0 errors, 1 warning

System (1) Processing (6)

Note that the template already includes the instantiation of the gNN\_ARCCOS 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:

```
signal clock : std_logic := '0'; -- make sure you initialise the clock signal to get it started

-- delete the init process block

always : PROCESS
-- optional sensitivity list
-- (          )
-- variable declarations
BEGIN
    -- code executes for every event on sensitivity list
    X <= x"10"; -- value is 0.5 (128/256) (exact 10arccos is 600)

    WAIT FOR 100 ns;

    X <= x"00"; -- value is 0 (exact 10arccos is 900)

    WAIT FOR 100 ns;

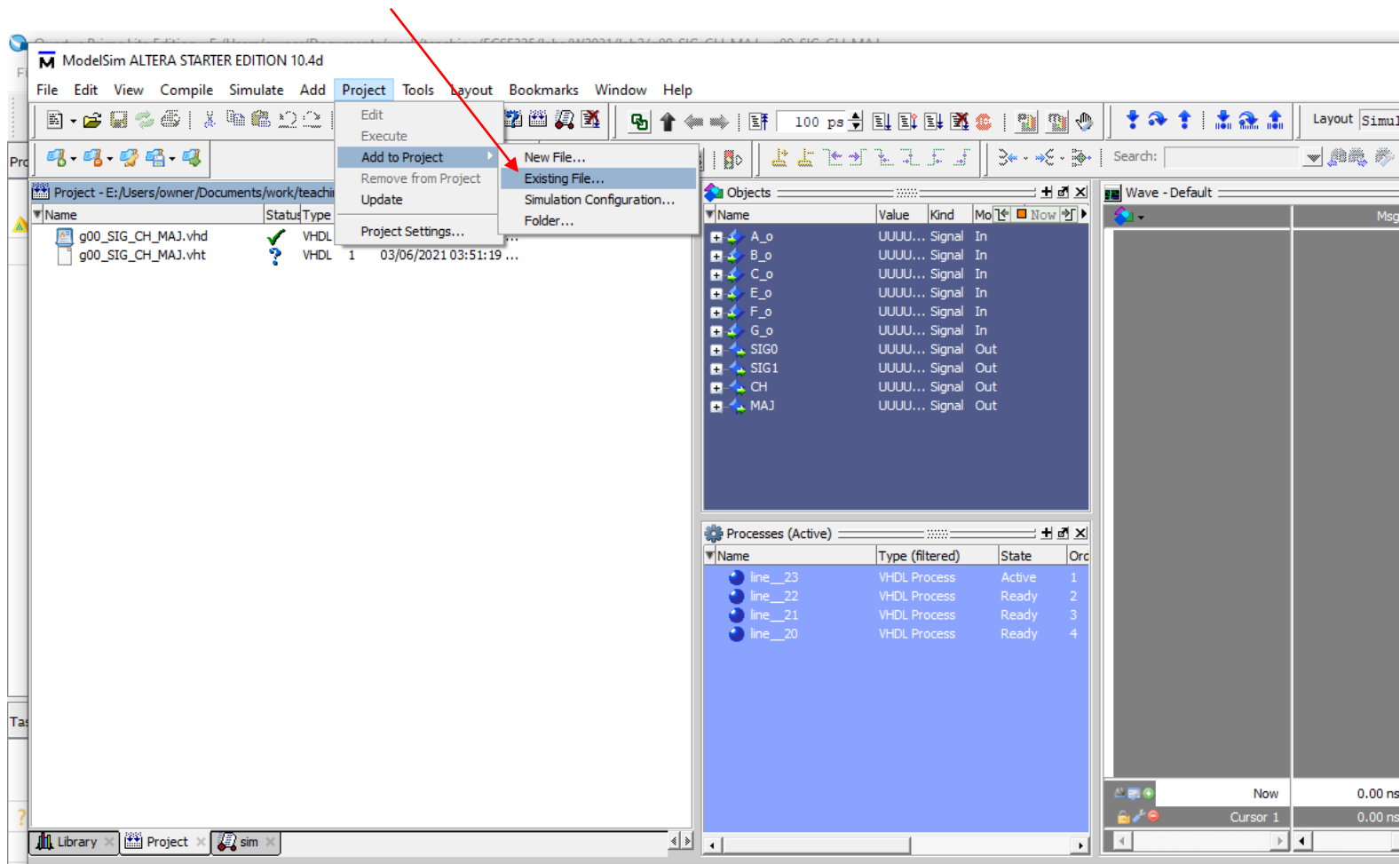
    X <= x"B5"; -- value is 0.707 (181/256) (exact 10arccos is 450)

    WAIT;
END PROCESS always;

Clock : PROCESS
BEGIN
    clock <= not clock after 5 ns;
END PROCESS clock;
```

This just creates 3 different test patterns, each presented at different times.

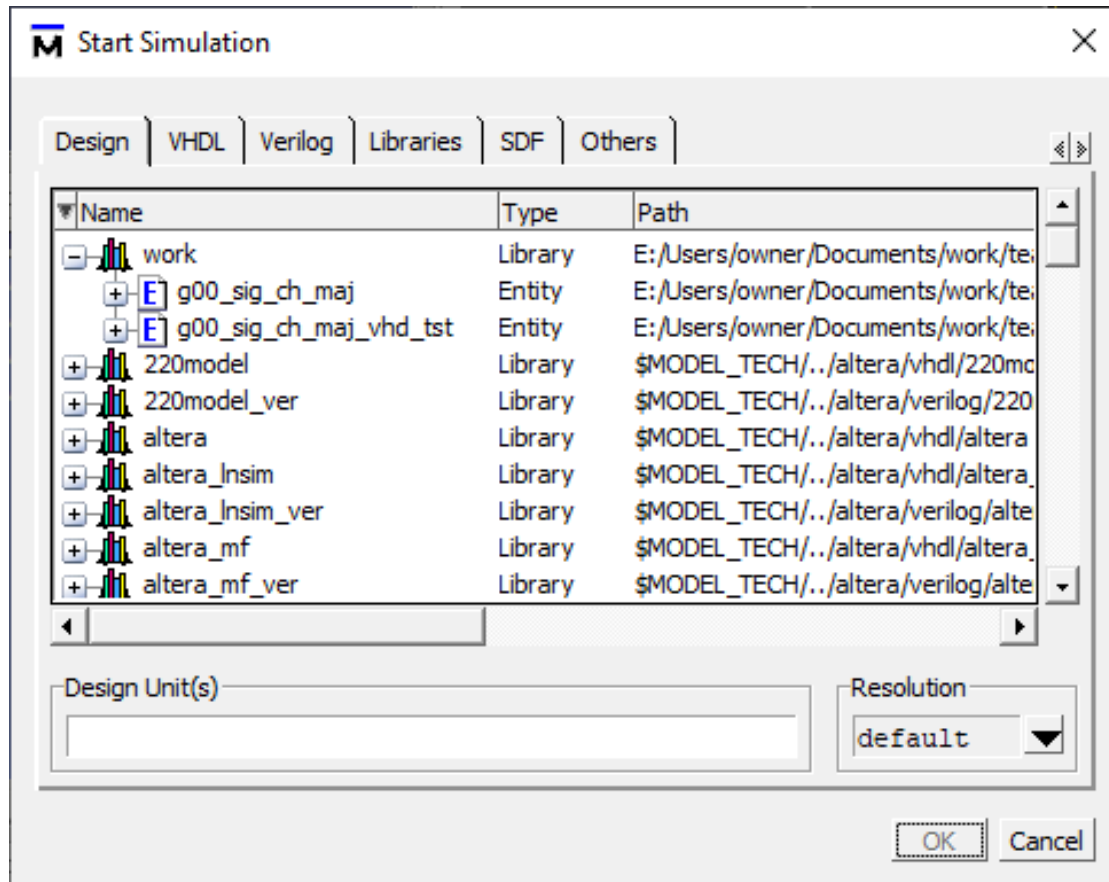
Once you have finished editing the testbench file, you need to add it to the project in ModelSim (make sure you are in the “Project” pane):



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.

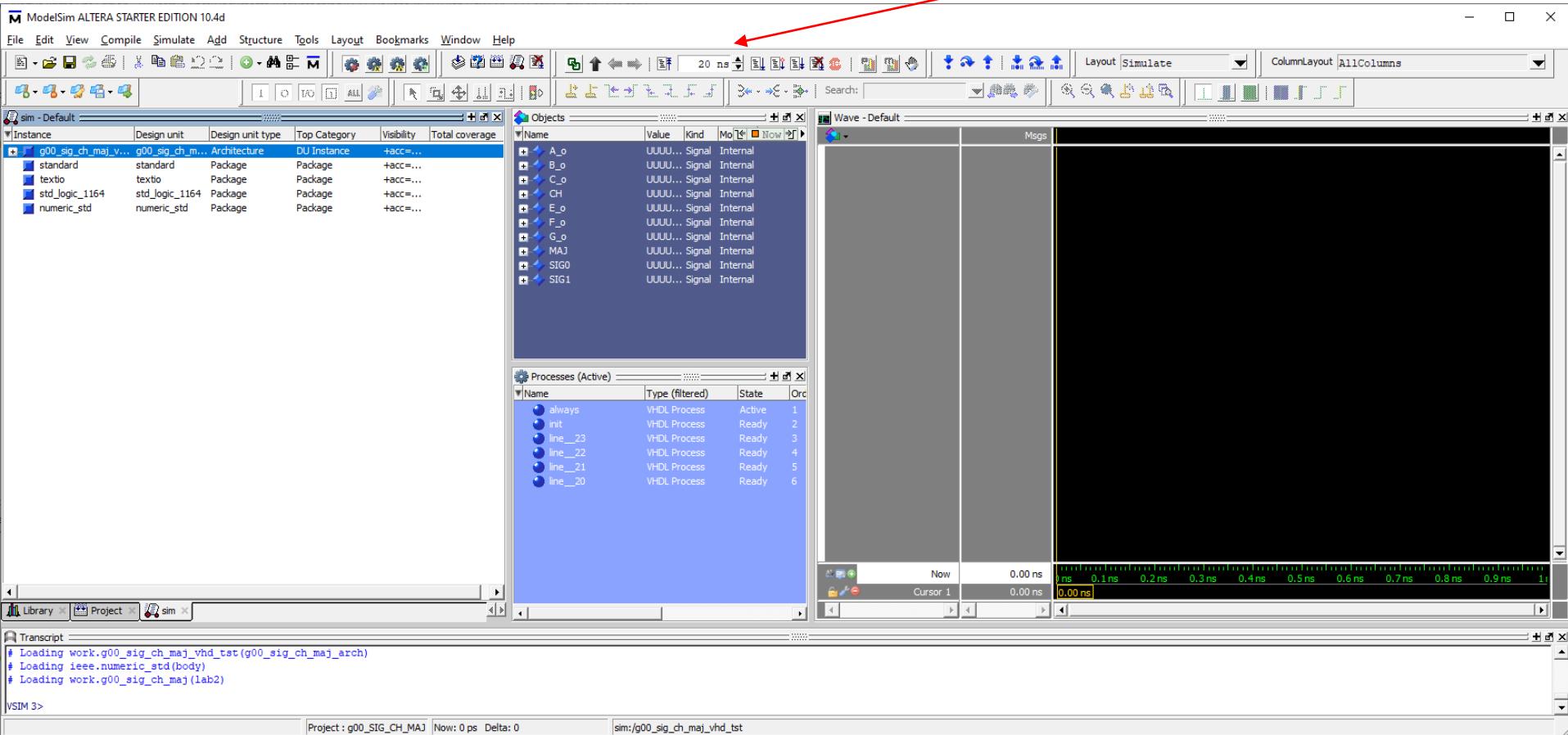
**Now everything is ready for you to actually run a simulation!**

Select “Start Simulation” from the Simulate toolbar item in the ModelSim program.  
The following window will popup:



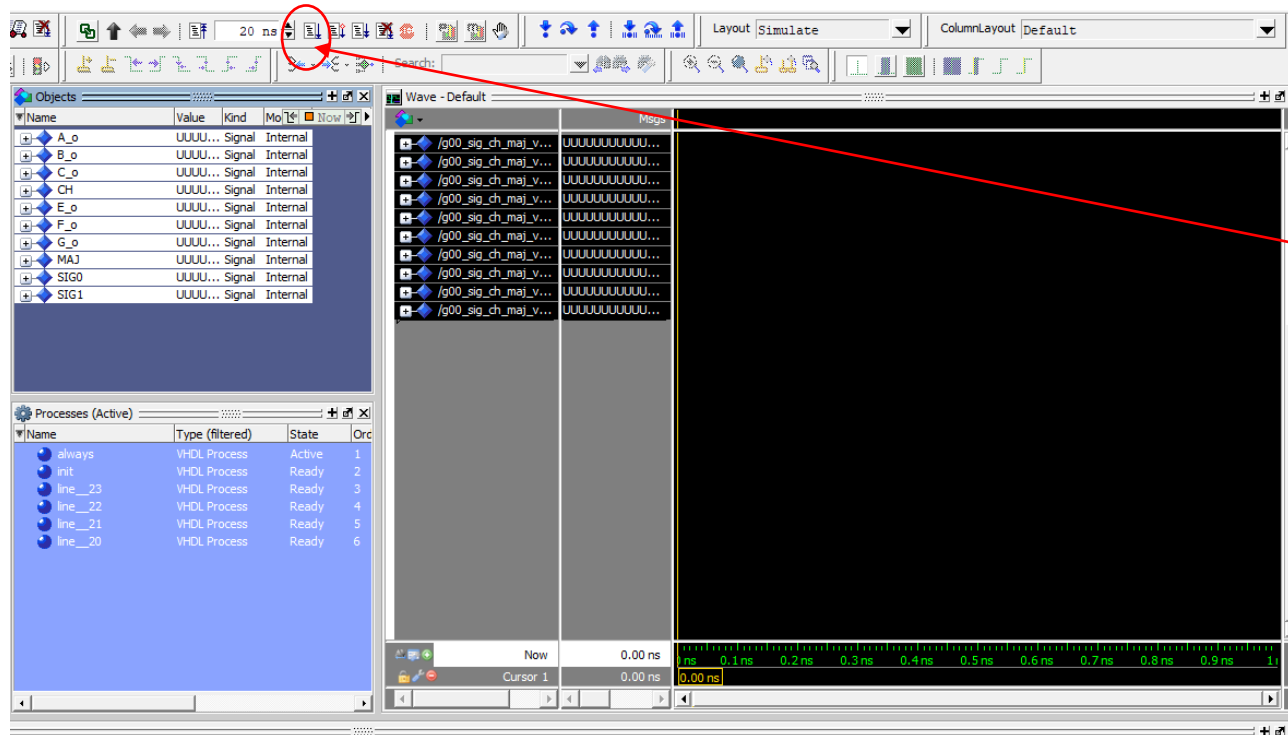
Select the gNN\_ARCCOS\_tst entity and click on OK

The ModelSim window should now look like this. Enter a value of 500ns into the simulation length window.



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.

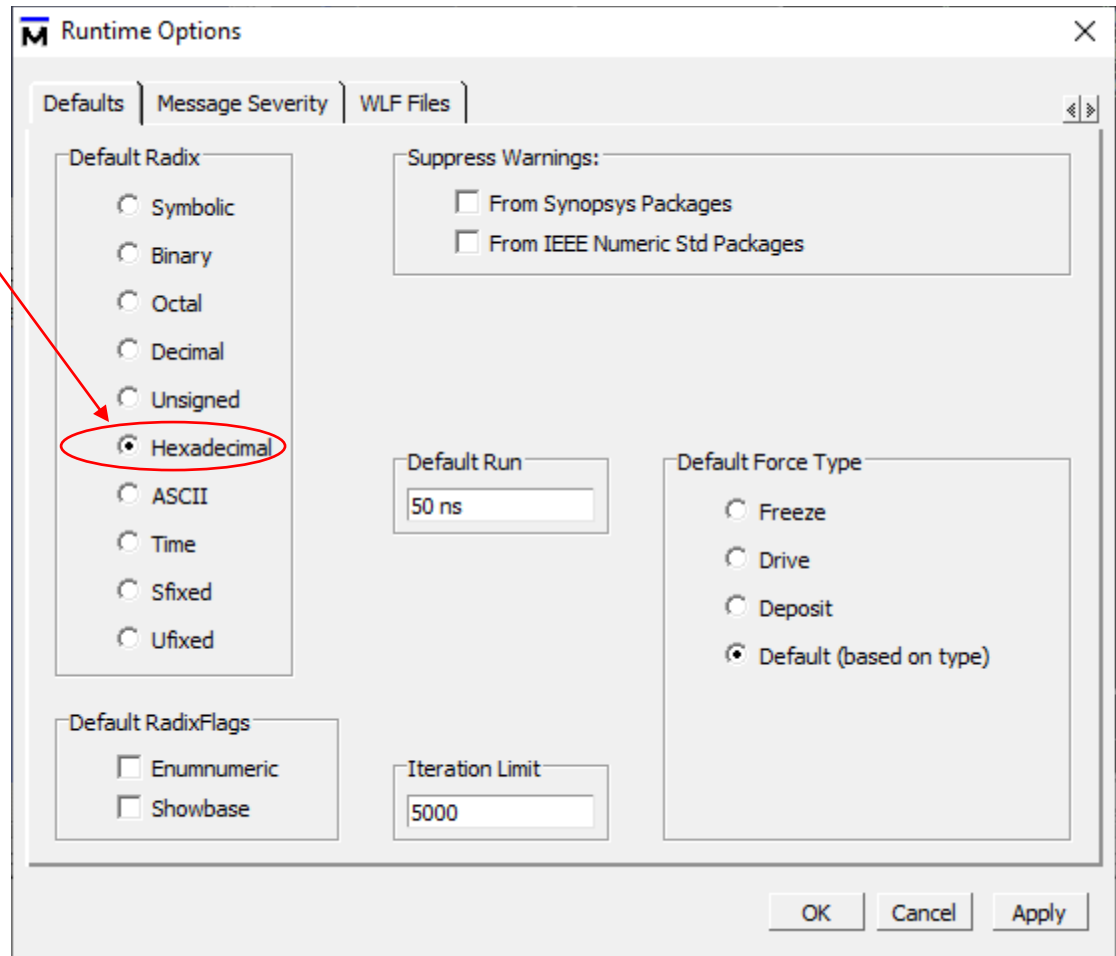
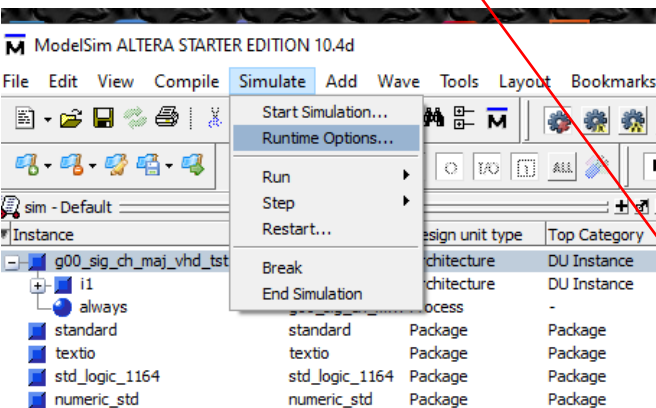
The Wave window will now look like this:



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



Since we are working with wide multi-bit vector 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.



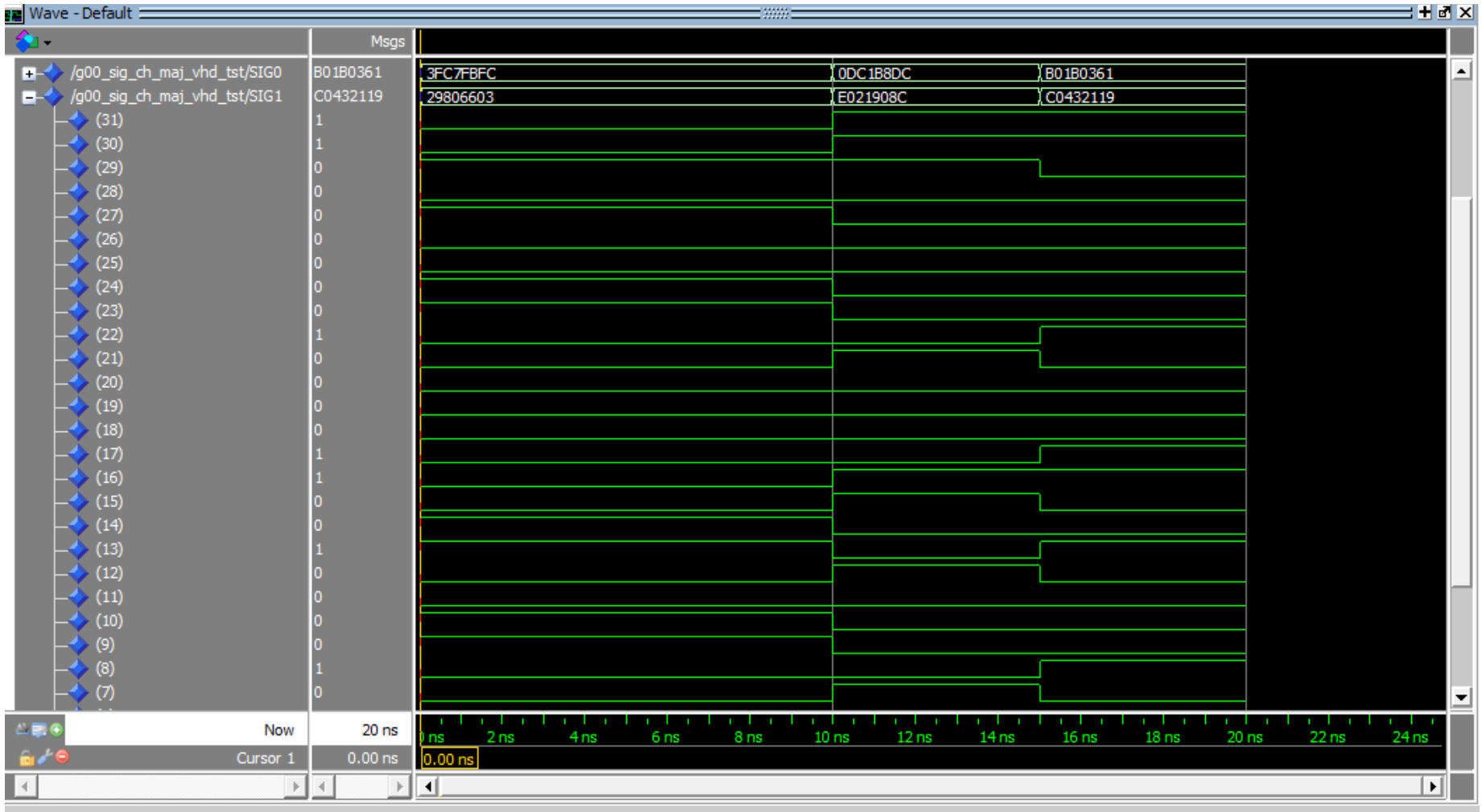
Here is the output you should get (you can right-click in the right-hand pane and select “Zoom Full” to see the entire time range).

The screenshot displays the ModelSim ALTERA STARTER EDITION 10.4d interface. The top menu bar includes File, Edit, View, Compile, Simulate, Add, Wave, Tools, Layout, Bookmarks, Window, and Help. The toolbar contains various icons for file operations, simulation control, and viewing options. The main workspace is divided into several panes:

- Project:** Shows the project hierarchy with files like g00\_sig\_ch\_maj\_vhd\_tst, i1, always, standard, textio, std\_logic\_1164, and numeric\_std.
- Objects:** Lists the objects in the design, including signals (A\_o, B\_o, C\_o, CH, E\_o, F\_o, G\_o, MAJ, SIG0, SIG1) and their values.
- Processes (Active):** Shows the active processes in the simulation.
- Wave:** Displays the simulation results for various signals. The signals listed are /g00\_sig\_ch\_maj\_vhd\_tst/A\_o, /g00\_sig\_ch\_maj\_vhd\_tst/B\_o, /g00\_sig\_ch\_maj\_vhd\_tst/C\_o, /g00\_sig\_ch\_maj\_vhd\_tst/CH, /g00\_sig\_ch\_maj\_vhd\_tst/E\_o, /g00\_sig\_ch\_maj\_vhd\_tst/F\_o, /g00\_sig\_ch\_maj\_vhd\_tst/G\_o, /g00\_sig\_ch\_maj\_vhd\_tst/MAJ, /g00\_sig\_ch\_maj\_vhd\_tst/SIG0, and /g00\_sig\_ch\_maj\_vhd\_tst/SIG1. The wave shows the values of these signals over time, with a zoomed-in view of the 0.00 ns to 24 ns range.

The bottom status bar shows the current time (0 ps to 24732 ps), project name (g00\_SIG\_CH\_MAJ), and simulation parameters (Now: 20 ns, Delta: 0).

To see the individual bits in a vector, click on the “+” next to the signal name in the left-hand column. There are a lot of bits so usually you wouldn’t want to do this unless you are trying to locate a glitch.

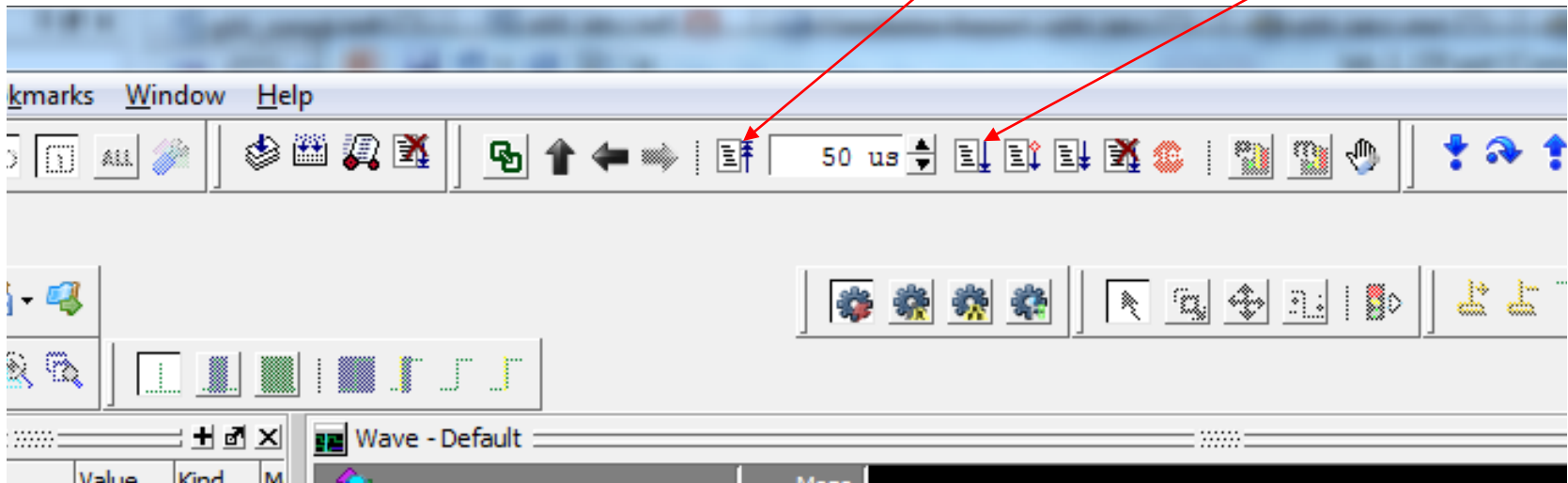


How many clock cycles are needed to arrive at the final result?

If you get an incorrect output waveform, you will have to go back and look at your design. If you make a correction to either of your schematic diagrams, you will have to re-run the conversion to VHDL (using the File/Create/Update/Create HDL Design File from Current File... command).

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

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



## Further Testing of the Project using ModelSim

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  $2^8=256$  possible input patterns, which is small enough that complete testing of the circuit is feasible.

Modify the testbench file to add a FOR loop that will generate the 256 test patterns, with a WAIT of 8 nsec per loop (so a total of 2 microseconds for the simulation).

Rerun the simulation with this amended testbench. Capture of screenshot of the resulting wave display for your report. Make sure that hexadecimal values are shown for the signal vector values.

Compute the expected values of the arccos approximation and the exact values using a calculator (or a computer program, such as in Matlab or python) for the 256 test values and compare to the simulated outputs.

*Although you do not have to do so for this lab, think about how you would check the output values in VHDL using ASSERT statements.*

## Using the TimeQuest Timing Analyzer .

To ensure a properly working circuit, the designer must take into consideration various timing constraints. In class we saw that for a register to correctly store an input value, the input must be held stable for a period (called the *setup time*) before the clock edge, and also for a period (called the *hold time*) after the clock edge.

Whether a circuit meets these timing constraints can only be known after the circuit is *synthesized*. After synthesis is done one can analyze the circuit to see if the timing constraints (setup and hold times) are satisfied.

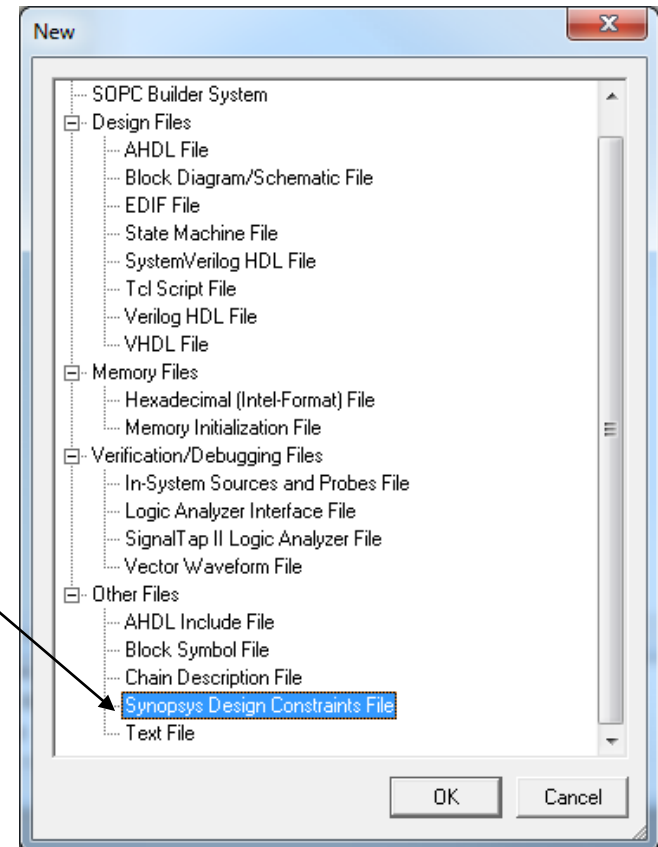
In Quartus II, timing analysis can be done using the *TimeQuest Timing Analyzer*.

From the “File/New” menu item, select “Synopsys Design Constraints File”.

This “.sdc” file is where we can specify various timing constraints for our design.

We will add a single constraint specifying the clock period.

The Quartus Fitter (the process that maps the design to the FPGA logic array) will attempt to do the mapping so as to meet the constraints.



The *gNN\_ARCCOS.sdc* file will be used to tell the timing analyzer that your clock period is *4ns* (corresponding to a clock frequency of 250MHz).

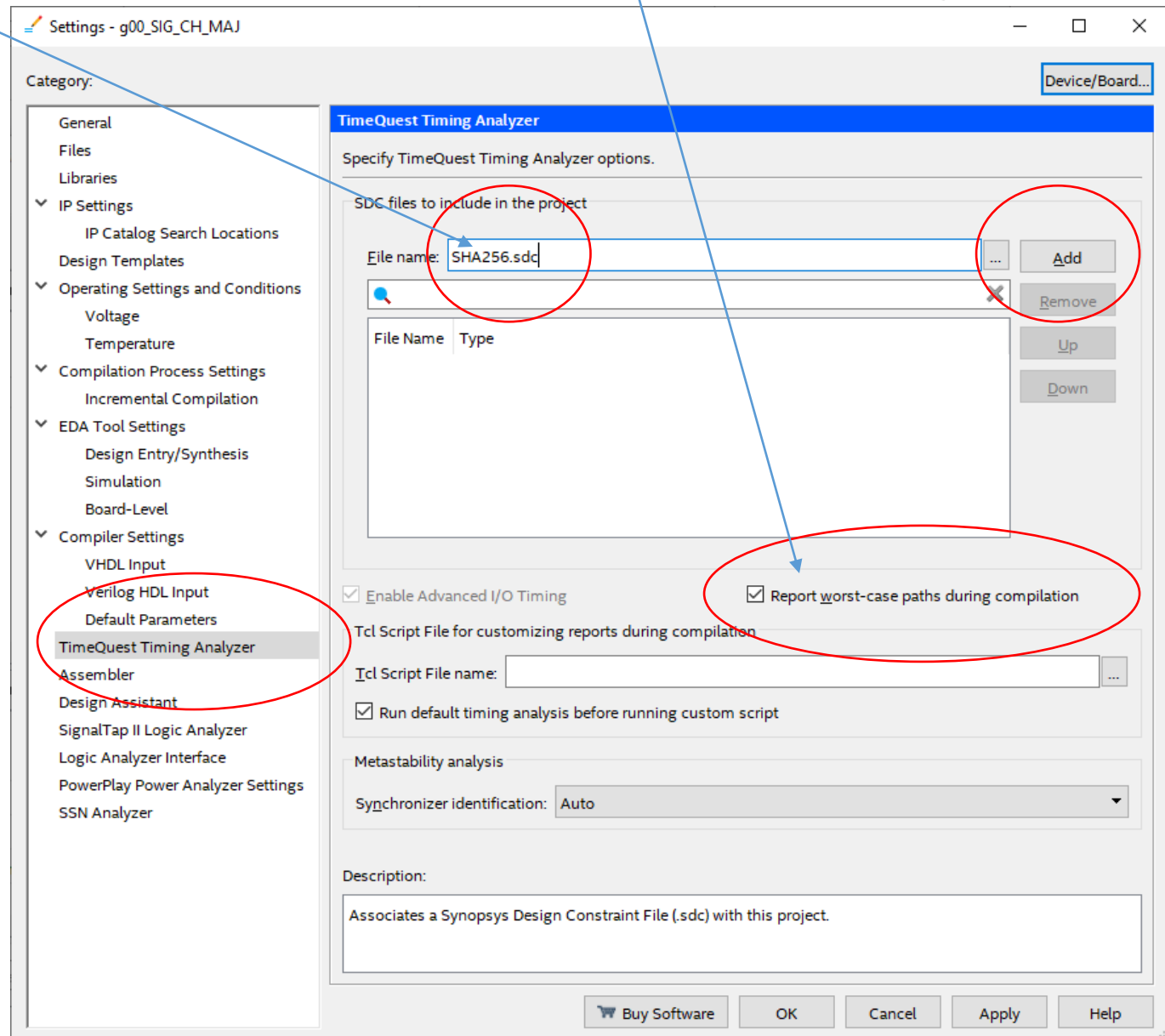
Enter the following single line into the file:

```
create_clock -period 4 [get_ports {clk}]
```

The Timing Analyzer will read the .sdc file and use the constraint information when doing its analysis.



To tell the TimeQuest Timing Analyzer to use your constraints file, go to the “Settings” item under the “Assignments” menu item. Add the file “gNN\_ARCCOS.sdc”. Also, select “Report worst-case paths during compilation”.



Compile your *gNN\_ARCCOS* project.

After compiling your design, there will be a “*TimeQuest Timing Analyzer*” section in the Compilation Report. Click on “*Slow 1100mV 85C Model*”.

The screenshot displays the Quartus Prime IDE interface. On the left, the 'Table of Contents' pane shows a tree structure. The 'TimeQuest Timing Analyzer' folder is expanded, revealing several sub-items. Two red arrows originate from the text above: one points to the 'TimeQuest Timing Analyzer' folder, and the other points to the 'Slow 1100mV 85C Model' sub-folder. The main window on the right shows the 'Flow Summary' section, which contains a table of compilation statistics. A red arrow points from the text 'Click on “Slow 1100mV 85C Model”’ to the 'Flow Summary' section.

Flow Summary	
Flow Status	Successful - Sun Mar 14 13:33:28 2021
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	g00_SIG_CH_MAJ
Top-level Entity Name	g00_SHA256
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	202 / 32,070 ( < 1 % )
Total registers	329
Total pins	258 / 457 ( 56 % )
Total virtual pins	0
Total block memory bits	0 / 4,065,280 ( 0 % )
Total DSP Blocks	0 / 87 ( 0 % )
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 ( 0 % )
Total DLLs	0 / 4 ( 0 % )

Note that some of the headings will be in **RED** indicating a *failed* timing. The Fmax reported in the Slow 1100mV 85C Model is less than that specified in the constraints file.

File Edit View Project Assignments Processing Tools Window Help

g00\_SIG\_CH\_MAJ

Project Navigator Hierarchy

Entity/Instance

Cyclone V: 5CSEMA5F31C6

g00\_SHA256

Table of Contents

- Analysis & Synthesis
- Fitter
- Assembler
- TimeQuest Timing Analyzer
  - Summary
  - Parallel Compilation
  - SDC File List
  - Clocks
  - Slow 1100mV 85C Model
    - Fmax Summary
    - Timing Closure Recommender
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Width Sum
    - Metastability Summary
  - Slow 1100mV 0C Model
  - Fast 1100mV 85C Model

Slow 1100mV 85C Model Fmax Summary

Fmax	Restricted Fmax	Clock Name	Note
140.0 MHz	140.0 MHz	CLK	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera

Tasks Compilation

Task

- Compile Design
- Analysis & Synthesis
- Fitter (Place & Route)
- Assembler (Generate program)
- TimeQuest Timing Analysis

Messages

System (1) Processing (136)

100% 00:01:09

Click on the “Setup Summary” report. This will list the minimum setup time slack value over all paths in the circuit. The End Point TNS is the *total negative slack* summed over all paths and can give you an idea of how extensive the timing problems are.

If this value is negative, then your circuit has timing problems somewhere.

Table of Contents

Parallel Compilation

SDC File List

Clocks

Slow 1100mV 85C Model

Fmax Summary

Timing Closure Recommender

Setup Summary

Hold Summary

Recovery Summary

Removal Summary

Minimum Pulse Width Summary

Metastability Summary

Slow 1100mV 0C Model

Fmax Summary

Slow 1100mV 85C Model Setup Summary

<<Filter>>

	Clock	Slack	End Point TNS
1	CLK	-1.143	-43.539

Click on the “Timing Closure Recommendations” report. This will list the paths with the most negative failing setup time slack values.

Make a note of which registers form the path. This will tell you where the maximum delays are, usually due to long chains of logic such as adders/multipliers etc.

Try to determine where in your circuit the long delays are.

Table of Contents

Parallel Compilation

SDC File List

Clocks

Slow 1100mV 85C Model

- Fmax Summary
- Timing Closure Recommender
- Setup Summary
- Hold Summary
- Recovery Summary
- Removal Summary
- Minimum Pulse Width Summary
- Metastability Summary

Slow 1100mV OC Model

- Fmax Summary
- Setup Summary

Timing Closure Recommendations

Summary [\[hide details\]](#)

This design contains failing setup paths with a worst-case slack of -1.143 ns. Run [Report Timing Closure Recommendations](#) for more recommendations for any particular path, click the appropriate link in the table below.

Top Failing Paths [\[hide details\]](#)

	Slack	From	To	Recommendations
1	-1.143	g00_Hash_Core:1 E_o[21]~reg0	g00_Hash_Core:1 A_o[30]~reg0	<a href="#">Report recommendations for this path</a>
2	-1.135	g00_Hash_Core:1 E_o[21]~reg0	g00_Hash_Core:1 A_o[30]~reg0	<a href="#">Report recommendations for this path</a>
3	-1.117	g00_Hash_Core:1 E_o[21]~reg0	g00_Hash_Core:1 A_o[29]~reg0	<a href="#">Report recommendations for this path</a>
4	-1.110	g00_Hash_Core:1 E_o[21]~reg0	g00_Hash_Core:1 A_o[29]~reg0	<a href="#">Report recommendations for this path</a>
5	-1.104	g00_Hash_Core:1 E_o[21]~reg0	g00_Hash_Core:1 E_o[30]~reg0	<a href="#">Report recommendations for this path</a>

Click on the “Hold Summary” report for the Fast 100mV 0C. This will list the minimum hold time slack value over all paths in the circuit. The End Point TNS is the *total negative slack* summed over all paths and can give you an idea of how extensive the timing problems are.

If this value is negative, then your circuit has timing problems somewhere. Usually, the Hold time slacks are positive, but you should check them anyway.

Table of Contents

- Slow 1100mV 0C Model
  - Fmax Summary
  - Setup Summary
  - Hold Summary
  - Recovery Summary
  - Removal Summary
  - Minimum Pulse Width Sun
  - Metastability Summary
- Fast 1100mV 85C Model
  - Setup Summary
  - Hold Summary
  - Recovery Summary
  - Removal Summary
  - Minimum Pulse Width Sun

Fast 1100mV 85C Model Hold Summary

<<Filter>>

	Clock	Slack	End Point TNS
1	CLK	0.131	0.000

Mark down the following values, which you should include in your report:

Requested ***Fmax*** = \_\_\_\_ 250 MHz \_\_\_\_\_

Fast 1100mV 0C Model ***Hold*** Slack Value = \_\_\_\_\_

Slow 1100mV 85C Model ***Setup*** Slack Value = \_\_\_\_\_

Slow 1100mV 85C Model ***Fmax*** = \_\_\_\_\_

List the Worst-case Timing paths for the Setup times.

It may be that with a 250MHz clock frequency request, the timing analysis does not pass. In this case we have to lower the requested clock frequency.

To try and get a passing timing analysis tell the timing analyzer that your clock period is **8ns** (corresponding to a clock frequency of 125MHz). Enter the following single line into the file:

```
create_clock -period 8 [get_ports {clk}]
```

The Timing Analyzer will read the .sdc file and use the constraint information when doing its analysis.

Recompile the project and look at the new timing analysis report.

*Does your circuit now pass the timing analysis?*

*If not, keep increasing the clock period in 1 nsec increments until the timing analysis is passed.*



Mark down the following values, which you should include in your report, for the case where the timing analysis is passed with no problems:

Requested *Fmax* = \_\_\_\_ ?? MHz \_\_\_\_\_

Fast 1100mV 0C Model *Hold* Slack Value = \_\_\_\_\_

Slow 1100mV 85C Model *Setup* Slack Value = \_\_\_\_\_

Slow 1100mV 85C Model *Fmax* = \_\_\_\_\_

You should also take a look at the “Flow Summary” after the compilation. In particular, take note of the usage of ALMs (adaptive logic modules). So far, you should only be using a small fraction of the FPGA’s logic elements.

The screenshot shows the Quartus Prime IDE interface. The top menu bar includes File, Edit, View, Project, Assignments, Processing, Tools, Window, and Help. The toolbar contains various icons for file operations and compilation. The Project Navigator on the left shows the project hierarchy for 'g00\_SIG\_CH\_MAJ'. The main window displays the 'Flow Summary' window, which is divided into a 'Table of Contents' and a 'Flow Summary' pane. The 'Table of Contents' lists various flow steps, including 'Analysis & Synthesis', 'Fitter', 'Assembler', and 'TimeQuest Timing Analyzer'. The 'Flow Summary' pane shows the results of the compilation, including 'Flow Status' (Successful), 'Quartus Prime Version' (16.1.0), 'Revision Name' (g00\_SIG\_CH\_MAJ), 'Top-level Entity Name' (g00\_SHA256), 'Family' (Cyclone V), 'Device' (5CSEMA5F31C6), 'Timing Models' (Final), and 'Logic utilization (in ALMs)' (202 / 32,070 (< 1 %)). A red circle highlights the 'Logic utilization (in ALMs)' entry, and a red arrow points from the text in the first block to this value. The bottom pane shows the 'Messages' window with a list of messages, including 'Running Quartus Prime EDA Netlist Writer', 'Command: quartus\_eda --read\_settings\_files=off --write\_settings\_files=off g00\_SHA256 -c g00\_SIG\_CH\_MAJ', '18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM\_PARALLEL\_PROCESSORS in your QSF to an', '10905 Generated the EDA functional simulation netlist because it is the only supported netlist type for this device.', '204019 Generated file g00\_SIG\_CH\_MAJ.vho in folder "E:/Users/owner/Documents/work/teaching/ECSE325/labs/w2021/lab2/simulation/modelsim/" for EDA simulation tool', 'Quartus Prime EDA Netlist Writer was successful. 0 errors, 2 warnings', and '293000 Quartus Prime Full Compilation was successful. 0 errors, 54 warnings'.

File Edit View Project Assignments Processing Tools Window Help

g00\_SIG\_CH\_MAJ

Project Navigator Hierarchy

Entity:Instance

Cyclone V: 5CSEMA5F31C6

g00\_SHA256

Table of Contents

Flow Summary

Flow Status Successful - Sun Mar 14 14:09:21 2021

Quartus Prime Version 16.1.0 Build 196 10/24/2016 SJ Lite Edition

Revision Name g00\_SIG\_CH\_MAJ

Top-level Entity Name g00\_SHA256

Family Cyclone V

Device 5CSEMA5F31C6

Timing Models Final

Logic utilization (in ALMs) 202 / 32,070 (< 1 %)

Total registers 316

Total pins 258 / 457 (56 %)

Total virtual pins 0

Total block memory bits 0 / 4,065,280 (0 %)

Total DSP Blocks 0 / 87 (0 %)

Total HSSI RX PCSs 0

Total HSSI PMA RX Deserializers 0

Total HSSI TX PCSs 0

Total HSSI PMA TX Serializers 0

Total PLLs 0 / 6 (0 %)

Total DLLs 0 / 4 (0 %)

Tasks

Compilation

Task

Compile Design

Analysis & Synthesis

Fitter (Place & Route)

Assembler (Generate program)

TimeQuest Timing Analysis

Messages

Type ID Message

Running Quartus Prime EDA Netlist Writer

Command: quartus\_eda --read\_settings\_files=off --write\_settings\_files=off g00\_SHA256 -c g00\_SIG\_CH\_MAJ

18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM\_PARALLEL\_PROCESSORS in your QSF to an

10905 Generated the EDA functional simulation netlist because it is the only supported netlist type for this device.

204019 Generated file g00\_SIG\_CH\_MAJ.vho in folder "E:/Users/owner/Documents/work/teaching/ECSE325/labs/w2021/lab2/simulation/modelsim/" for EDA simulation tool

Quartus Prime EDA Netlist Writer was successful. 0 errors, 2 warnings

293000 Quartus Prime Full Compilation was successful. 0 errors, 54 warnings

System (1) Processing (150)

100% 00:01:08

Also look at the Chip Planner, found in the Fitter report. It gives an overview of how the design was mapped to the FPGA logic array.

The screenshot displays the Quartus Prime Lite Edition interface. The top menu bar includes File, Edit, View, Project, Assignments, Processing, Tools, Window, and Help. The toolbar contains various icons for file operations and compilation. The Project Navigator on the left shows the project hierarchy with 'g00\_SHA256' selected. The Table of Contents in the center lists various reports, with 'Chip Planner' highlighted under the 'Fitter' section. The Flow Summary on the right provides a detailed overview of the compilation process, including flow status, version, revision name, top-level entity name, family, device, timing models, and resource utilization. The Messages window at the bottom shows the compilation results, including the command used to generate the netlist and a warning about the number of processors.

Quartus Prime Lite Edition - E:/Users/owner/Documents/work/teaching/ECSE325/labs/W2021/lab2/g00\_SHA256 - g00\_SIG\_CH\_MAJ

File Edit View Project Assignments Processing Tools Window Help

g00\_SIG\_CH\_MAJ

Project Navigator Hierarchy

Entity: Instance

Cyclone V: 5CSEMA5F31C6

g00\_SHA256

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Assembler
- TimeQuest Timing Analyzer
- EDA Netlist Writer
- Flow Messages
- Flow Suppressed Messages

Flow Summary

Flow Status: Successful - Sun Mar 14 14:09:21 2021

Quartus Prime Version: 16.1.0 Build 196 10/24/2016 SJ Lite Edition

Revision Name: g00\_SIG\_CH\_MAJ

Top-level Entity Name: g00\_SHA256

Family: Cyclone V

Device: 5CSEMA5F31C6

Timing Models: Final

Logic utilization (in ALMs): 202 / 32,070 (< 1 %)

Total registers: 316

Total pins: 258 / 457 (56 %)

Total virtual pins: 0

Total block memory bits: 0 / 4,065,280 (0 %)

Total DSP Blocks: 0 / 87 (0 %)

Total HSSI RX PCSs: 0

Total HSSI PMA RX Deserializers: 0

Total HSSI TX PCSs: 0

Total HSSI PMA TX Serializers: 0

Total PLLs: 0 / 6 (0 %)

Total DLLs: 0 / 4 (0 %)

Tasks

Compilation

Task

- Fitter (Place & Route)
- Edit Settings
- View Report
- Chip Planner
- Technology Map Viewer (Pc)

















Messages

Type ID Message

- Quartus Prime EDA Netlist Writer was successful. 0 errors, 2 warnings
- 293000 Quartus Prime Full Compilation was successful. 0 errors, 54 warnings
- Running Quartus Prime Netlist Viewers Preprocess
- Command: quartus\_npp g00\_SHA256 -c g00\_SIG\_CH\_MAJ --netlist\_type=atom\_fit
- 18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM\_PARALLEL\_PROCESSORS in your QSF to an
- Quartus Prime Netlist viewers Preprocess was successful. 0 errors, 1 warning

System (1) Processing (155)

100% 00:00:01

Report

Report not available

Tasks

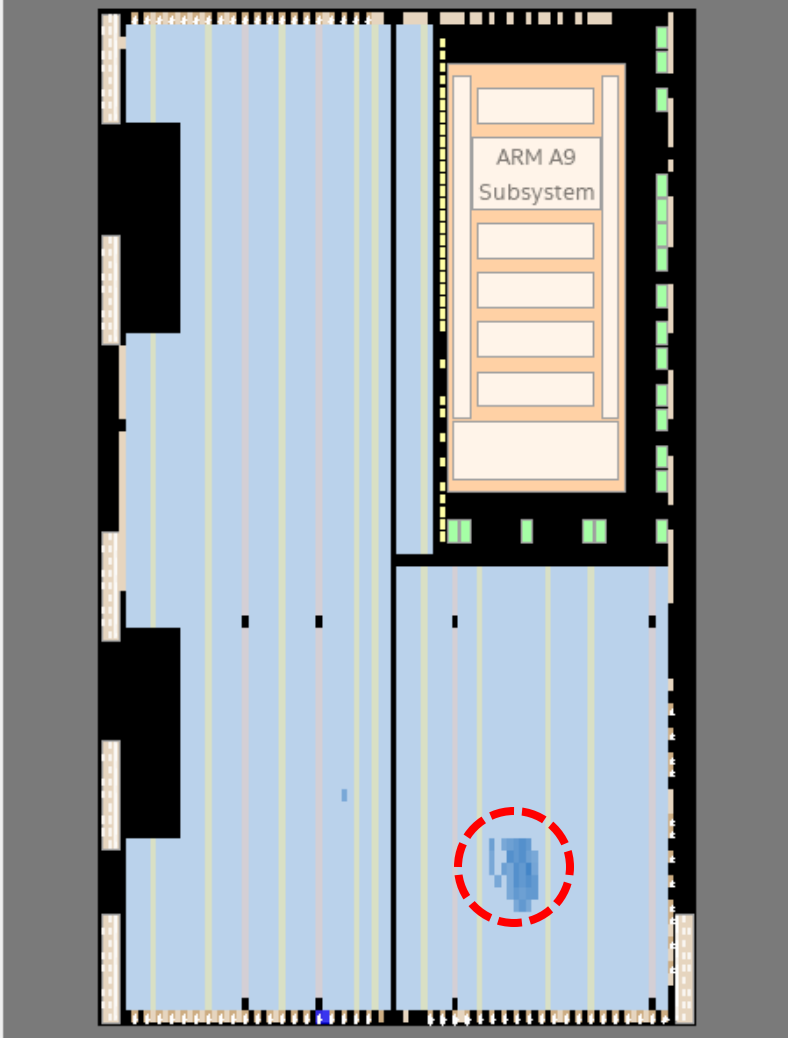
- Generate Clock Data
- Toggle Background C
- Report Resources...
- Report Compilation M

Console

tcl>

Console History

Coordinate: Editing Mode: ECO - 5CSEMA5F31C6



Layers Settings

Basic

☒ Background


- ☐ None
- ☒ Block Utilization

Layers Settings

Color Legend

Node Properties

Selected elements: CLK



Properties/Modes	Values
Full Name	g00_SHA256 CLK
Coordinate	(32 , 0)

Pad

Input Buffer

Properties

Fan-in

Fan-out

# Writeup the Lab Report .

Write up a short report describing the *gNN\_ARCCOS* circuit that you designed 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, and a title, giving the name (e.g. *gNN\_ARCCOS*) of the circuit.
- A description of the circuit's function, listing the inputs and outputs.
- The VHDL description of the circuit.
- The final version of the VHDL testbench file.
- A screenshot of the simulation results for the first simulation run. This should show clearly the hexadecimal values of the signals over the time interval 0 to 500 nsec. Compare the obtained results with pre-computed values (using a calculator).
- The contents of the timing constraints file (.sdc file).
- Timing analysis reports produced during the second part of the lab, including the recorded values from pages 39 and 41.

**The report is due on March 24, at 11:59 PM.**

# Submit the Lab Report to myCourses .

The lab report, and all associated design files (*the design files are the vhdl, testbench and timing constraints (.vhd, .vht, .sdc) 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 gNN\_LAB\_1.zip (where NN is your group number).**