CPE 300L
DIGITAL SYSTEM ARCHITECTURE AND DESIGN LABORATORY

LABORATORY 1
INTRODUCTION TO VERILOG

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF NEVADA, LAS VEGAS

OBJECTIVE

Get started with Quartus, Modelsim and review the Verilog HDL.

INTRODUCTION

History of Verilog:

Verilog was originally developed at Gateway Design Automation in 1985. Phil Moorby developed it as a proprietary Hardware Description Language (HDL) for the Gateway Verilog-XL digital logic simulator. Cadence Design Systems became the owner of Verilog when they acquired Gateway Design Automation in 1989.

In 1990, as a response to the growing popularity of VHDL, Cadence put Verilog in the public domain. At the same time Open Verilog International (OVI) was formed to manage the Verilog HDL. The Verilog-XL user manual was the main reference for Verilog and served as its standard, the Verilog 1.0 Reference Manual. Some minor changes to Verilog were reflected in Verilog 2.0 a few years later.

In 1993 OVI submitted a request to the IEEE to make Verilog an IEEE Standard. In 1995 this process produced IEEE 1364-1995, the first official Verilog standard based on Verilog 2.0. This version of Verilog was popularly known as Verilog-95. In 1997 work began on an updated standard which addressed short-comings in Verilog-95 and incorporated significant enhancements. In 2001 this effort resulted in IEEE 1364-2001 with this version of Verilog known as Verilog-2001.

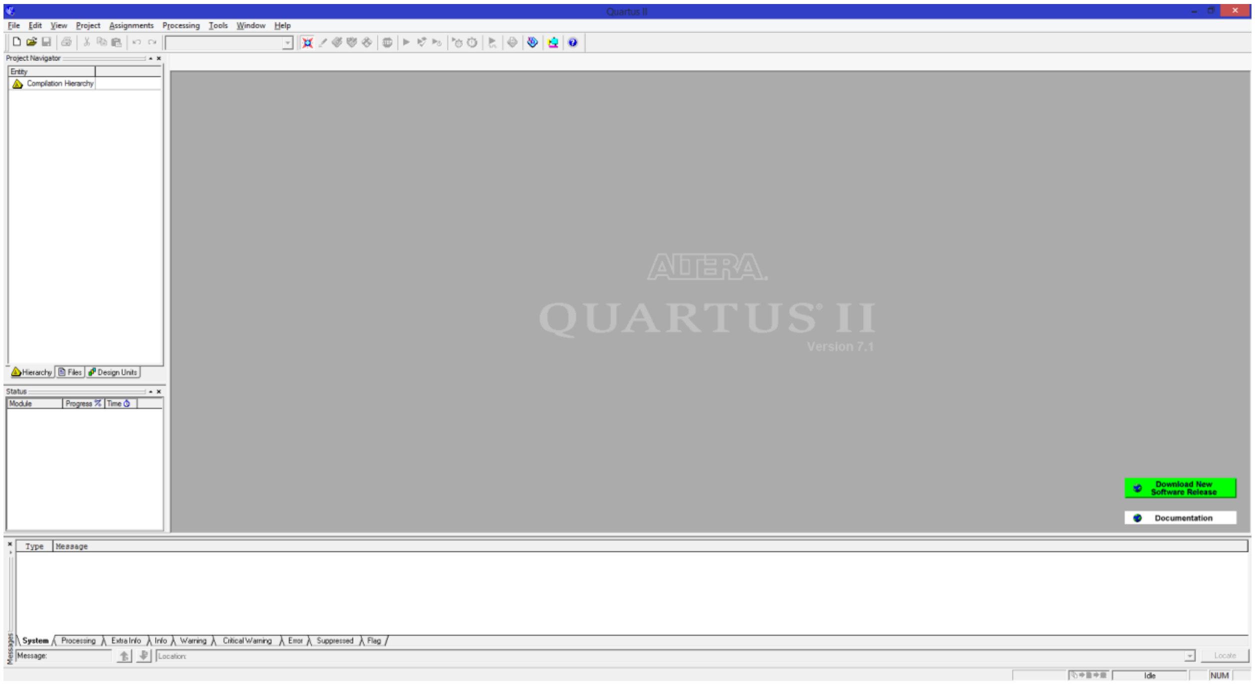
One of the traditional shortcomings of Verilog has been the lack of constructions that are useful for systems level. Verilog has always been strong for modeling at the gate and Register Transfer Level (RTL) but weaker than VHDL in modeling at a system level. To address these short-comings SystemVerilog was developed. SystemVerilog was originally developed on top of Verilog under the auspices of Accellera starting about 2002. It became IEEE Standard 1800-2005 in 2005.

SystemVerilog was combined with the Verilog standard as IEEE 1800-2009. This standard is known as SystemVerilog. The most recent version of this standard is EEE 1800-2012, published in February, 2013.

THEORY OF OPERATION

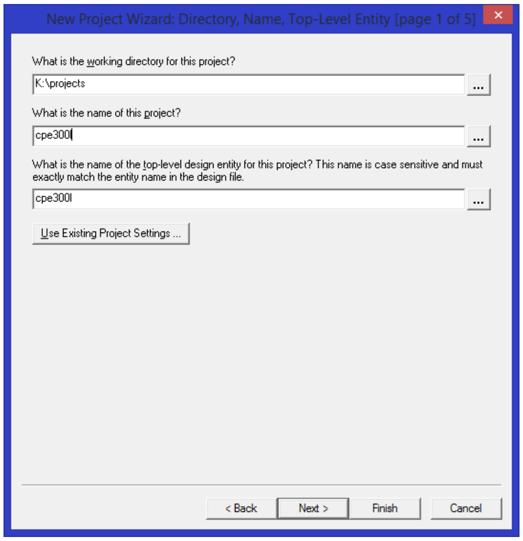
Simulating the Verilog code in Quartus

To implement the *and2* module in Quartus in order to simulate, use the following instruction.
1. Start Quartus II Software, version 7.1


Quartus II
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Compilation Hierarchy
Hierarchy Files Design Units
Status
Module Progress % Time
ALTERA
QUARTUS II
Version 7.1
Download New Software Release
Documentation
Type Message
System Processing Extra Info Info Warning Critical Warning Error Suppressed Flag
Messages
Message:
Location:
Locate
Idle
NUM

2. Create new project

File -> New Project Wizard. Fill the values in:


New Project Wizard: Directory, Name, Top-Level Entity [page 1 of 5]
Use Existing Project Settings ...
< Back
Next >
Finish
Cancel

What is the working directory for this project?
K:\projects

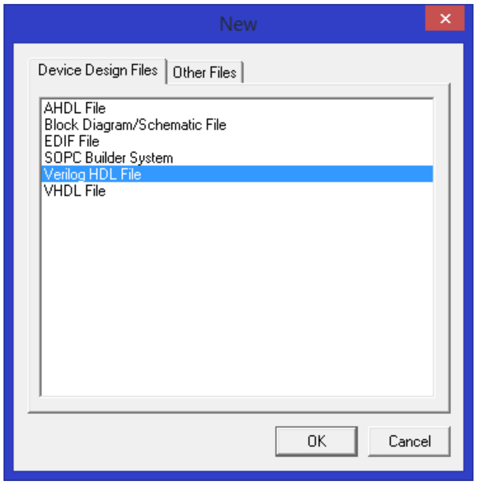
What is the name of this project?
cpe300l

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
cpe300l

Click *Finish*.

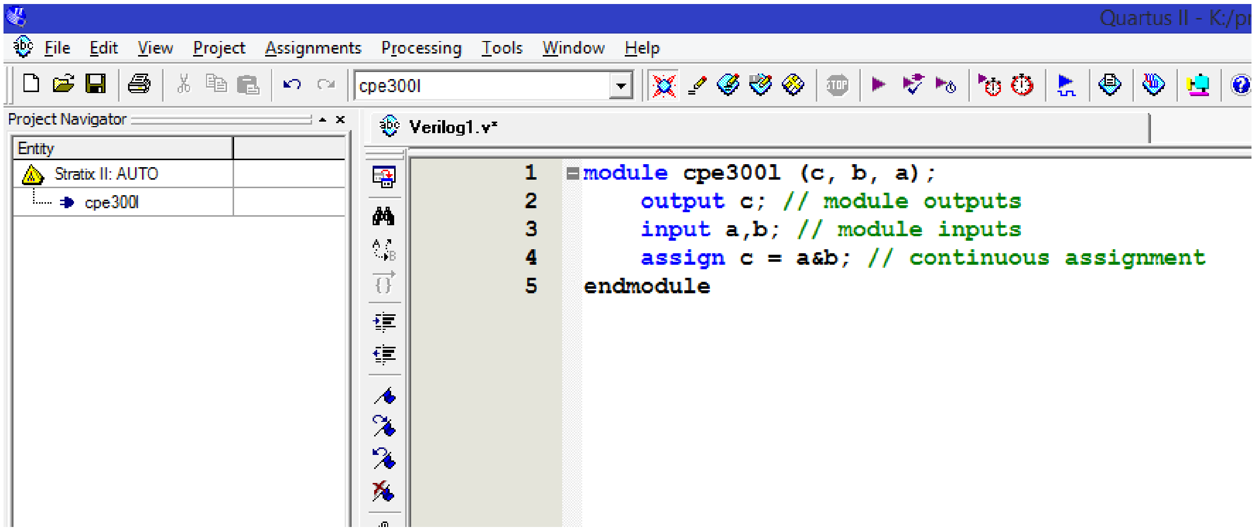
2. Add Verilog file to the project

File -> New -> "Verilog HDL File"



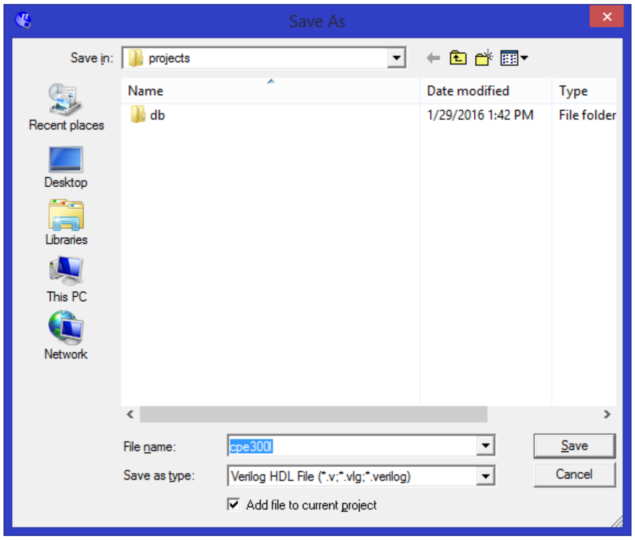
AHDL File
Block Diagram/Schematic File
EDIF File
SOPC Builder System
Verilog HDL File
VHDL File

3. Paste in the code, name module the same as the project. This must be done here as Quartus requires the top-level entity to have the same name as the project, and this only file is the top-level entity.



```
module cpe300l (c, b, a);
    output c; // module outputs
    input a,b; // module inputs
    assign c = a&b; // continuous assignment
endmodule
```

4. Save your Verilog file



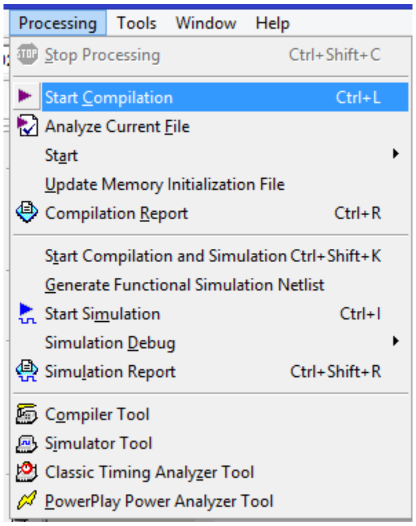
File name: cpe300l

Save as type: Verilog HDL File (*.v;*.vlg;*.verilog)

☑ Add file to current project

5. Compile the project

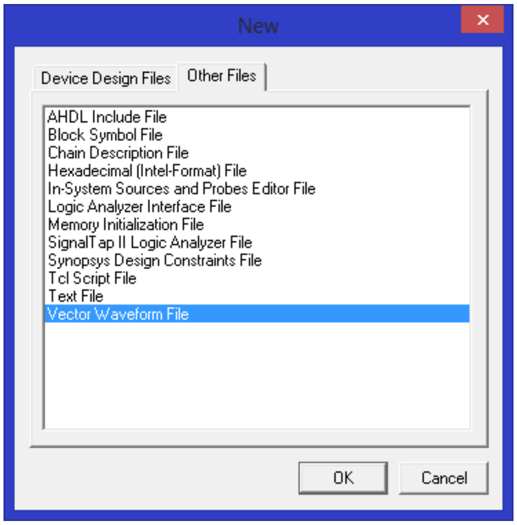
Processing -> Start Compilation


Processing
Tools
Window
Help
Stop Processing Ctrl+Shift+C
Start Compilation Ctrl+L
Analyze Current File
Start
Update Memory Initialization File
Compilation Report Ctrl+R
Start Compilation and Simulation Ctrl+Shift+K
Generate Functional Simulation Netlist
Start Simulation Ctrl+I
Simulation Debug
Simulation Report Ctrl+Shift+R
Compiler Tool
Simulator Tool
Classic Timing Analyzer Tool
PowerPlay Power Analyzer Tool

Wait for the message that the Full compilation was successful.

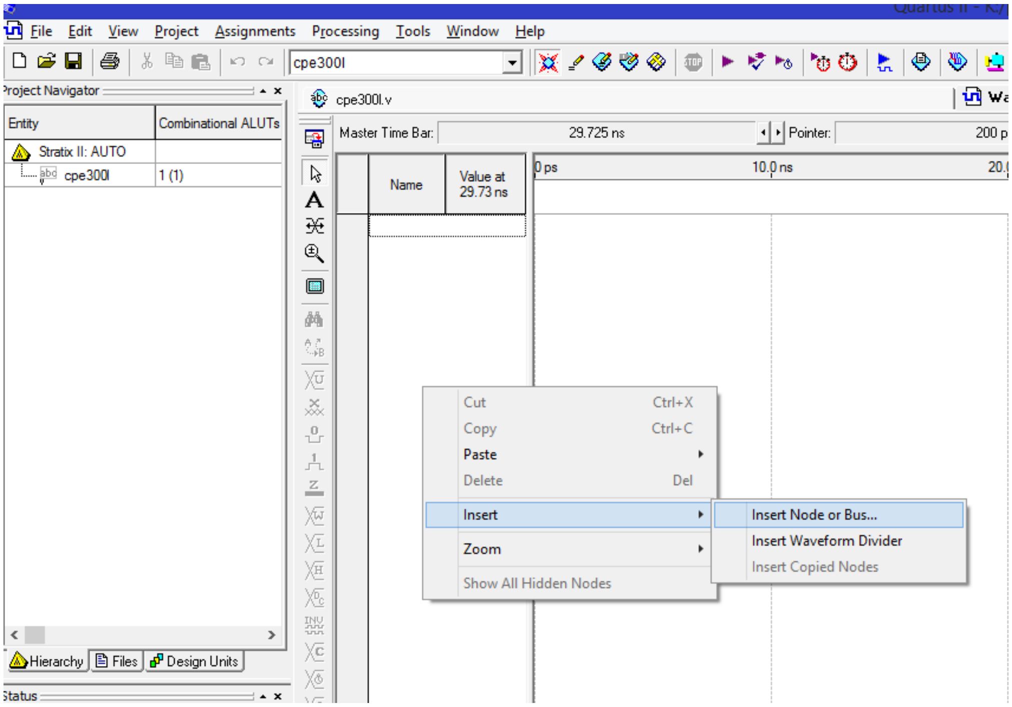
6. Add the waveform file.

File -> New -> Other Files -> Vector Waveform File

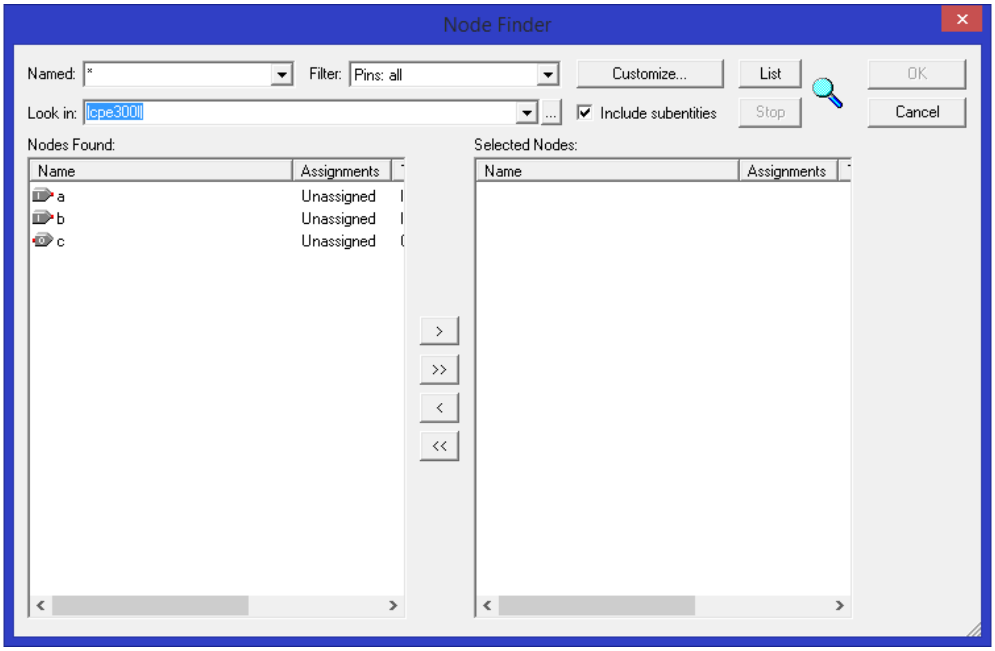

New
Device Design Files
Other Files
AHDL Include File
Block Symbol File
Chain Description File
Hexadecimal (Intel-Format) File
In-System Sources and Probes Editor File
Logic Analyzer Interface File
Memory Initialization File
SignalTap II Logic Analyzer File
Synopsys Design Constraints File
Tcl Script File
Text File
Vector Waveform File
OK
Cancel

7. Add signals

In the Waveform window, right click in the area on the left



Choose *Insert -> Insert Node or Bus -> Node Finder*



Named: *

Filter: Pins: all

Look in: |cpe300l|

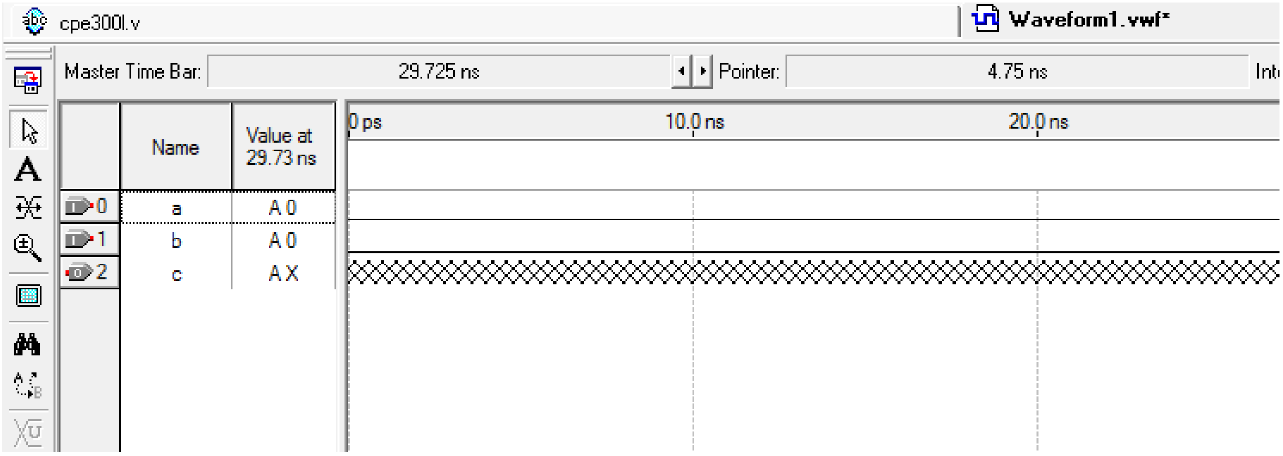
☑ Include subentities

Name	Assignments
a	Unassigned
b	Unassigned
c	Unassigned

Set *Filter: Pins: all*, click *List*. Add all the signals (user >> function) and click OK, OK.

8. Setting input values

After adding the signals, they will appear in the waveform file.

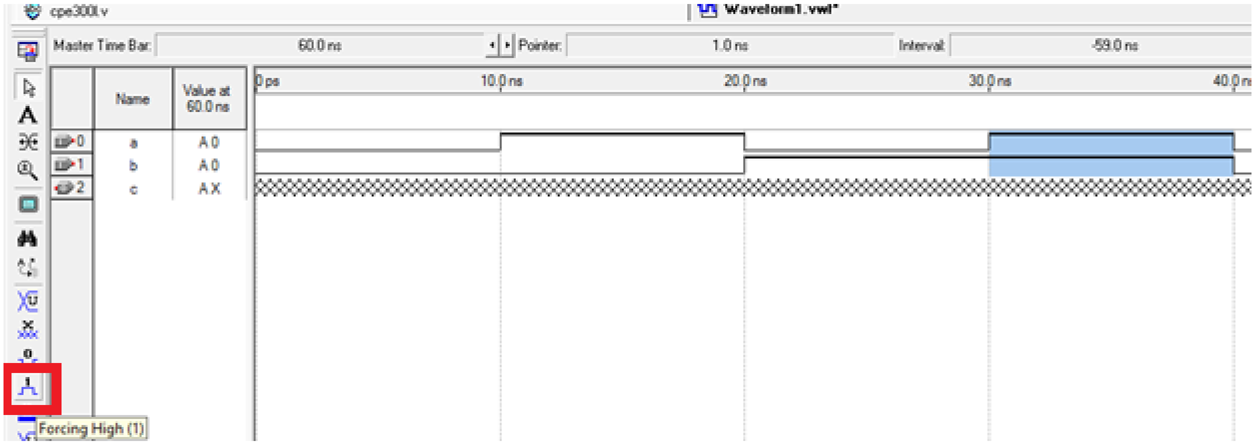


Master Time Bar: 29.725 ns

Pointer: 4.75 ns

	Name	Value at 29.73 ns
0	a	A 0
1	b	A 0
2	c	A X

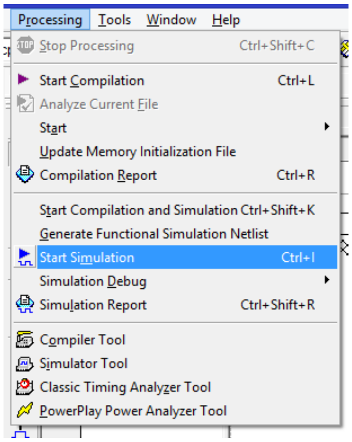
Mark signals in the respective time periods, and use the controls in the left bar to set values.



Set all the combinations of the outputs. Save the waveform file with the default filename.

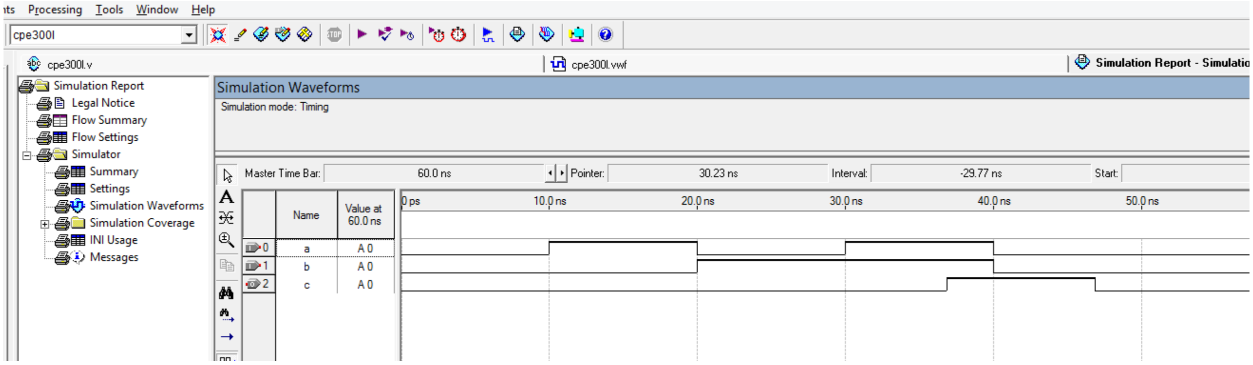
9. Simulate

Processing -> Start Simulation


Processing Tools Window Help
Stop Processing Ctrl+Shift+C
Start Compilation Ctrl+L
Analyze Current File
Start
Update Memory Initialization File
Compilation Report Ctrl+R
Start Compilation and Simulation Ctrl+Shift+K
Generate Functional Simulation Netlist
Start Simulation Ctrl+I
Simulation Debug
Simulation Report Ctrl+Shift+R
Compiler Tool
Simulator Tool
Classic Timing Analyzer Tool
PowerPlay Power Analyzer Tool

10. Results

You can see, that now the output signal *y* is determined.



Lab Deliveries

Prelab

1. Read safety slides: http://eelabs.faculty.unlv.edu/docs/rules/safety.pdf
2. Watch safety movie: https://youtu.be/28ixSnZ0ycE
3. Follow the EDAPlayground tutorial video in the link: https://www.youtube.com/watch?v=NXlqdrYga9M

Prelab deliveries:

1. No deliveries

Lab Experiments

1. Follow the steps in the screenshot above and get the waveform for the **AND** gate in Quartus 7.1 software
2. Add an inverter to the output of circuit from experiment 1. Write the code and get the waveform.

Postlab Report:

Include the following elements in the report document:

<table>
<tr><th>Section</th><th colspan="2">Element</th></tr>
<tr><td>1</td><td colspan="2">Theory of operation
<i>What is a Hardware Description Language (HDL)? List any two HDL.</i></td></tr>
<tr><td rowspan="4">2</td><td colspan="2">Results of the experiments</td></tr>
<tr><th>Experiment</th><th>Experiment Results</th></tr>
<tr><td>1</td><td>Waveform from Quartus.</td></tr>
<tr><td>2</td><td>Verilog code and waveform</td></tr>
<tr><td rowspan="4">3</td><td colspan="2">Answer the questions</td></tr>
<tr><th>Question no.</th><th>Question</th></tr>
<tr><td>1</td><td>What is Register Transfer Level (RTL)?</td></tr>
<tr><td>2</td><td>What is the difference between HDL and general purpose languages (such as C/C++ for example)?</td></tr>
<tr><td>4</td><td colspan="2">Conclusions
<i>Write down your conclusions, things learned, problems encountered during the lab and how they were solved, etc.</i></td></tr>
</table>