DIGITAL DESIGN

ASSIGNMENT REPORT

ASSIGNMENT ID : 1

Student Name: 林秋实

Student ID: 11610502

PART 1: DIGITAL DESIGN THEORY

Provide your answers here:

1.

Decimal	Octal	Hexadecimal	Duodecimal (Base 12)
10	12	A	A
11	13	B	B
12	14	C	10
13	15	D	11
14	16	E	12
15	17	F	13
16	20	10	14
17	21	11	15
18	22	12	16
19	23	13	17
20	24	14	18
21	25	15	19
22	26	16	1A
23	27	17	1B
24	30	18	20
25	31	19	21
26	32	1A	22
27	33	1B	23
28	34	1C	24
29	35	1D	25
30	36	1E	26
31	37	1F	27
32	40	20	28

2.

The largest binary number that can be expressed with 12 bits is 111111111111(binary) which equals $2^{12} - 1$ The equivalent decimal is 4095.

The equivalent hexadecimal is FFF.

3.

(a)

512 / 2 = 256, 512 % 2 = 0.

256 / 2 = 128, 256 % 2 = 0

128 / 2 = 56, 128 % 2 = 0

64 / 2 = 32, 64 % 2 = 0

32 / 2 = 16, 32 % 2 = 0

16 / 2 = 8, 16 % 2 = 0

8 / 2 = 4, 8 % 2 = 0

4 / 2 = 2, 4 % 2 = 0

2 / 2 = 1, 2 % 2 = 0

1 / 2 = 0, 1 % 2 = 1

Thus, the binary of 512(decimal) is 1000000000(binary).

(b)

512 / 16 = 32, 512 % 16 = 0

32 / 16 = 2, 32 % 16 = 0

2 / 16 = 0, 2 % 16 = 2

Thus, the hexadecimal of 512(decimal) is 200(hexadecimal).

200(hexadecimal) -> 2|0|0(hexadecimal) -> 10|0000|0000(binary) -> 1000000000(binary)

Thus, the binary of 512(decimal) is 1000000000(binary).

I think the method (b) is faster.

4.

Suppose P1, P2 are two propositions.

- ∧ “AND”

It is only true when both propositions are true.

P1	P2	P1 ∧ P2
T	T	T
F	F	F
T	F	F
F	T	F

- ∨ “OR”

It is true when any of the propositions are true.

P1	P2	P1 ∨ P2
T	T	T
F	F	F
T	F	T
F	T	T

- ┐ “OR”

Its result is the opposite of the proposition.

P1	┐P2
T	F
F	T

5.

a. 01101010010000010011010100

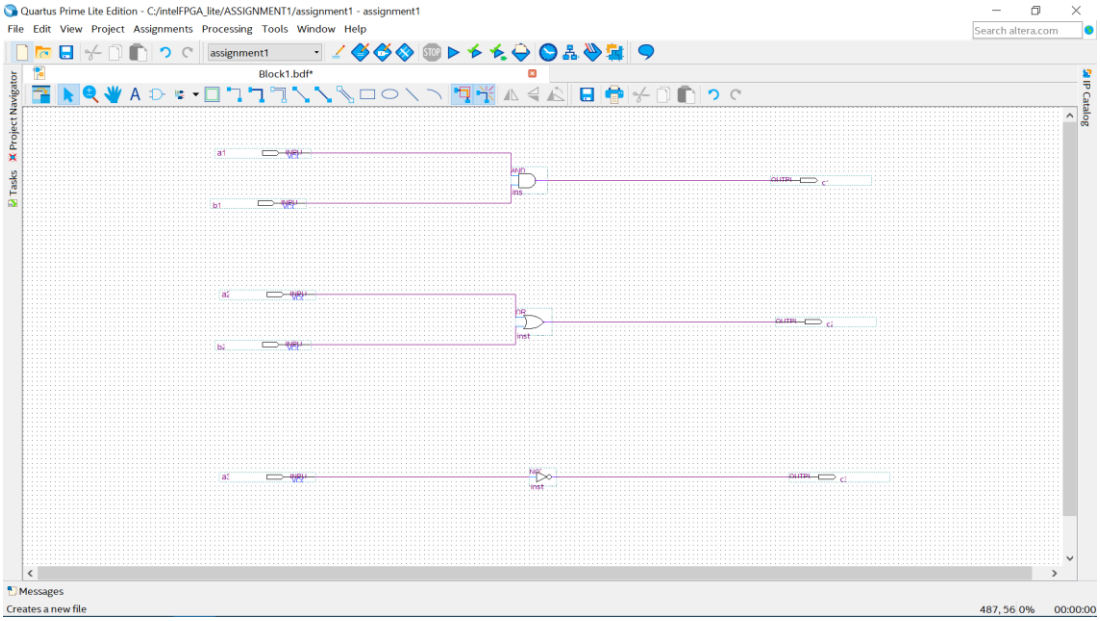
b. 01111000100111000111011111 0

PART 2: DIGITAL DESIGN LAB

DESIGN

Describe the design of your system by providing the following information:

- *Circuit design (provide screen shots)*



- *Verilog design (provide the verilog code)*

Verilog Code (from assignment1.v):

```
module assignment1c1, c2, c3, a1, a2, a3, b1, b2);
// Logic “AND”
input a1, b1;
output c1;
assign c1 = a1 && b1;

// Logic “OR”
input a2, b2;
output c2;
assign c2 = a2 || b2;

// Logic “NOT”
input a3;
output c3;
assign c3 = ~a3;

endmodule
```

- *The true table (write by yourself)*

Logic “AND”

a1	b1	c1
1	1	1
0	0	0
1	0	0
0	1	0

Logic “OR”

a2	b2	c2
1	1	1
0	0	0
1	0	1
0	1	1

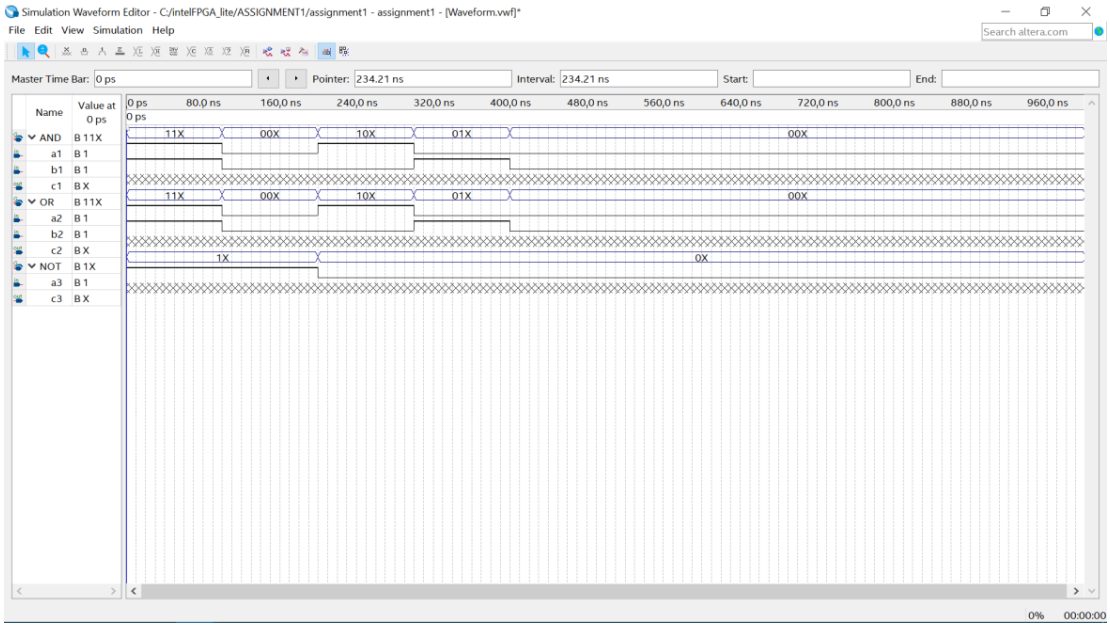
Logic “NOT”

a3	c3
1	0
0	1

SIMULATION

Describe how you build the testbench and do the simulation .

- *Using vmf (provide screen shots)*



- *Using Verilog (provide the verilog code)*

Verilog Code (from assignment1.vt):

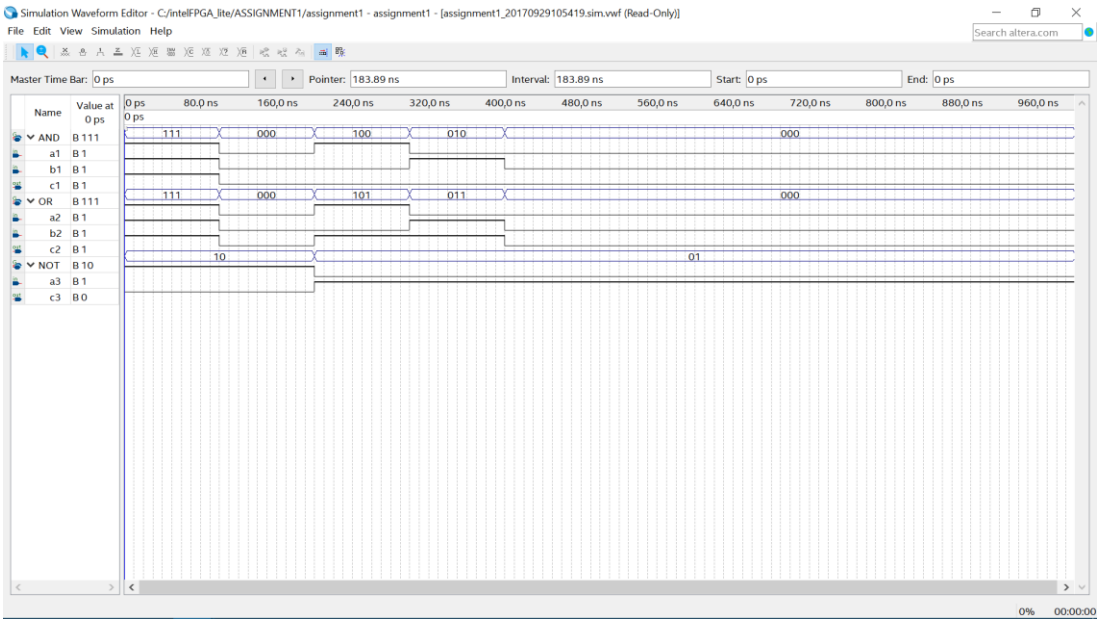
```
// Copyright (C) 2017  Intel Corporation. All rights reserved.
// Your use of Intel Corporation's design tools, logic functions
// and other software and tools, and its AMPP partner logic
// functions, and any output files from any of the foregoing
// (including device programming or simulation files), and any
// associated documentation or information are expressly subject
// to the terms and conditions of the Intel Program License
// Subscription Agreement, the Intel Quartus Prime License Agreement,
// the Intel MegaCore Function License Agreement, or other
// applicable license agreement, including, without limitation,
// that your use is for the sole purpose of programming logic
// devices manufactured by Intel and sold by Intel or its
// authorized distributors.  Please refer to the applicable
// agreement for further details.

// ********************************************************************
// This file contains a Verilog test bench template that is freely editable to
// suit user's needs .Comments are provided in each section to help the user
// fill out necessary details.
// ********************************************************************
```

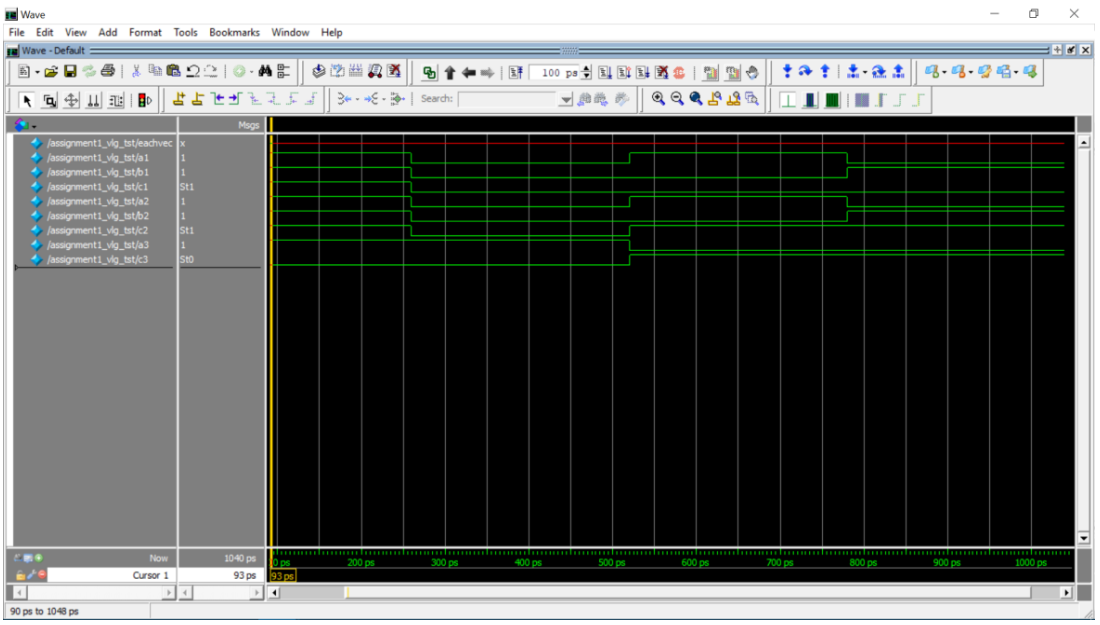
```
// Generated on "09/29/2017 11:38:10"

// Verilog Test Bench template for design : assignment1
//
// Simulation tool : ModelSim-Altera (Verilog)
//

`timescale 1 ps/ 1 ps
module assignment1_vlg_tst();
// constants
// general purpose registers
reg eachvec;
// test vector input registers
reg a1;
reg a2;
reg a3;
reg b1;
reg b2;
// wires
wire c1;
wire c2;
wire c3;

// assign statements (if any)
assignment1 i1 (
// port map - connection between master ports and signals/registers
  .a1(a1),
  .a2(a2),
  .a3(a3),
  .b1(b1),
  .b2(b2),
  .c1(c1),
  .c2(c2),
  .c3(c3)
);
initial
begin
// code that executes only once
// insert code here --> begin
```

```
// --> end
$display("Running testbench");
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin

   a1 = 1;
   b1 = 1;
   a2 = 1;
   b2 = 1;
   a3 = 1;
   #260;

   a1 = 0;
   b1 = 0;
   a2 = 0;
   b2 = 0;
   a3 = 1;
   #260;

   a1 = 1;
   b1 = 0;
   a2 = 1;
   b2 = 0;
   a3 = 0;
   #260;

   a1 = 0;
   b1 = 1;
   a2 = 0;
   b2 = 1;
   a3 = 0;
   #260;
```

```
@eachvec;
// --> end
end
endmodule
```

- *Wave form of simulation result (provide screen shots and compare with true table)*





After comparison, the results are the same as the truth tables.

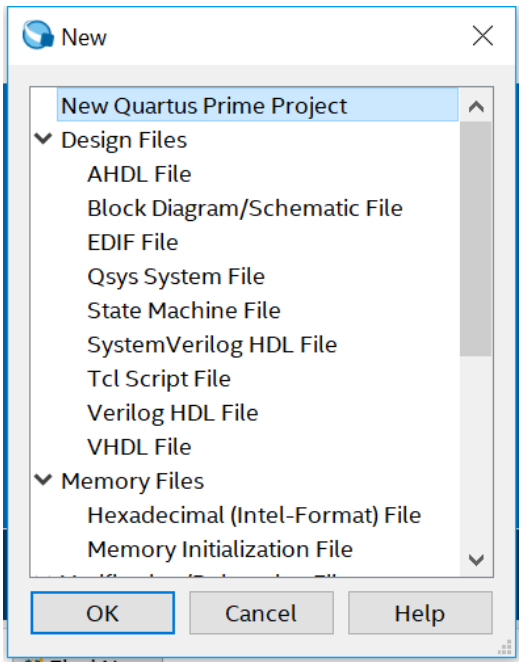
THE DESCRIPTION OF OPERATION

Describe the operation on QuartusII, including how to build a project, edit a design file (bdf and verilog), do the compiling, build a testbench, do the simulation. Provide screen shots of all the different steps and dialogues and any other relevant information you consider important.

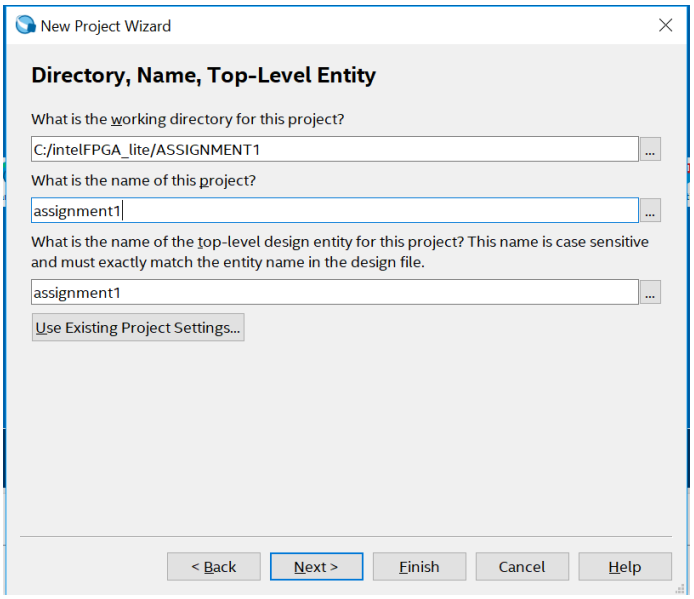
- *The operation steps*

Steps on how to build a new project:

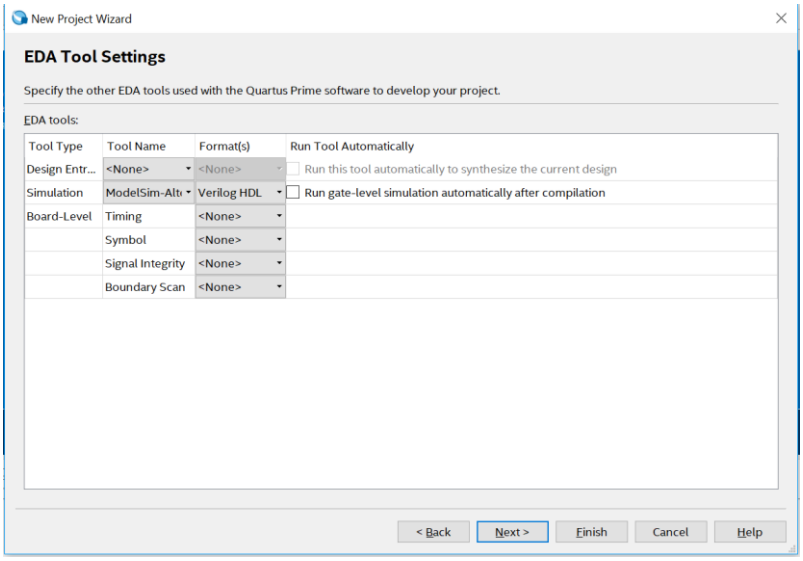
1) Make a New Quartus Prime Project.



2) Chose a name of this project.

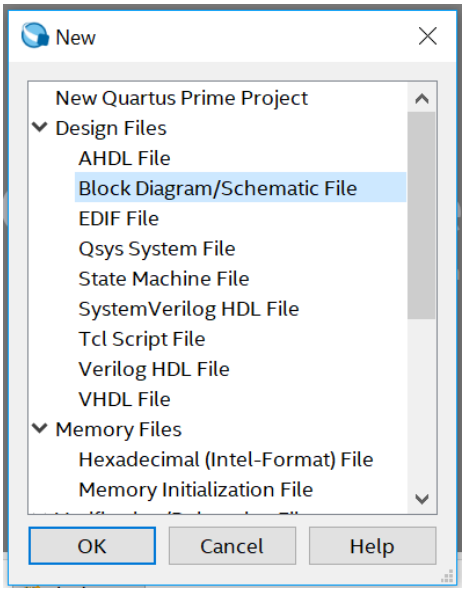


3) Set the simulation tool.

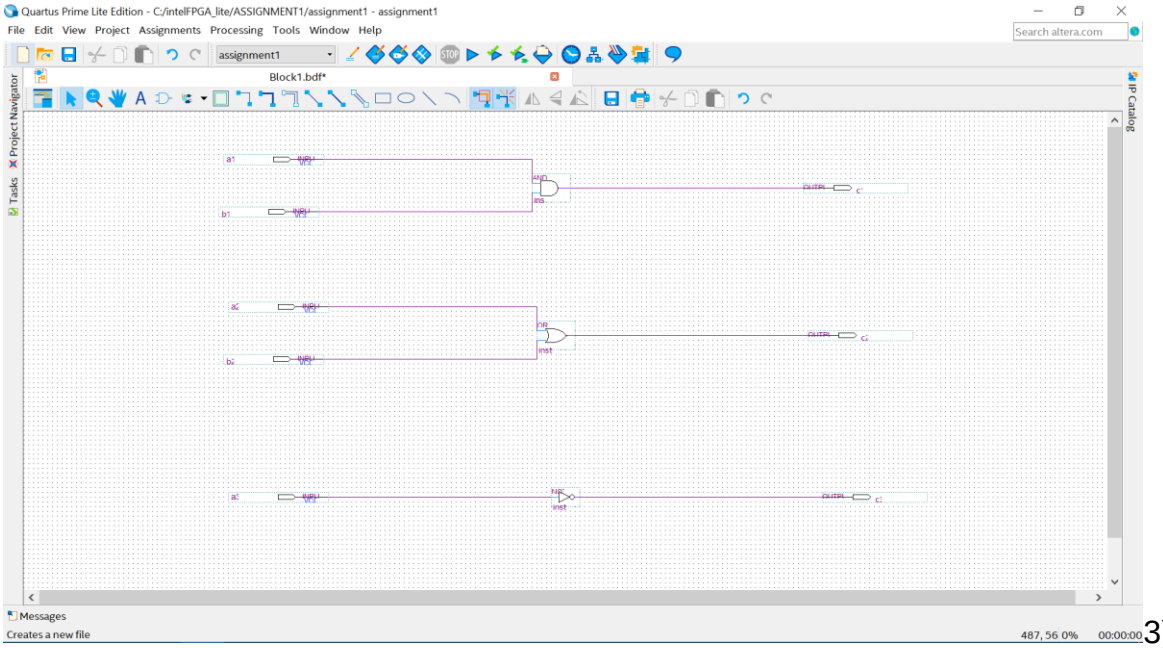


Steps on a BDF file:

1) Make a new Block Diagram/Schematic File.

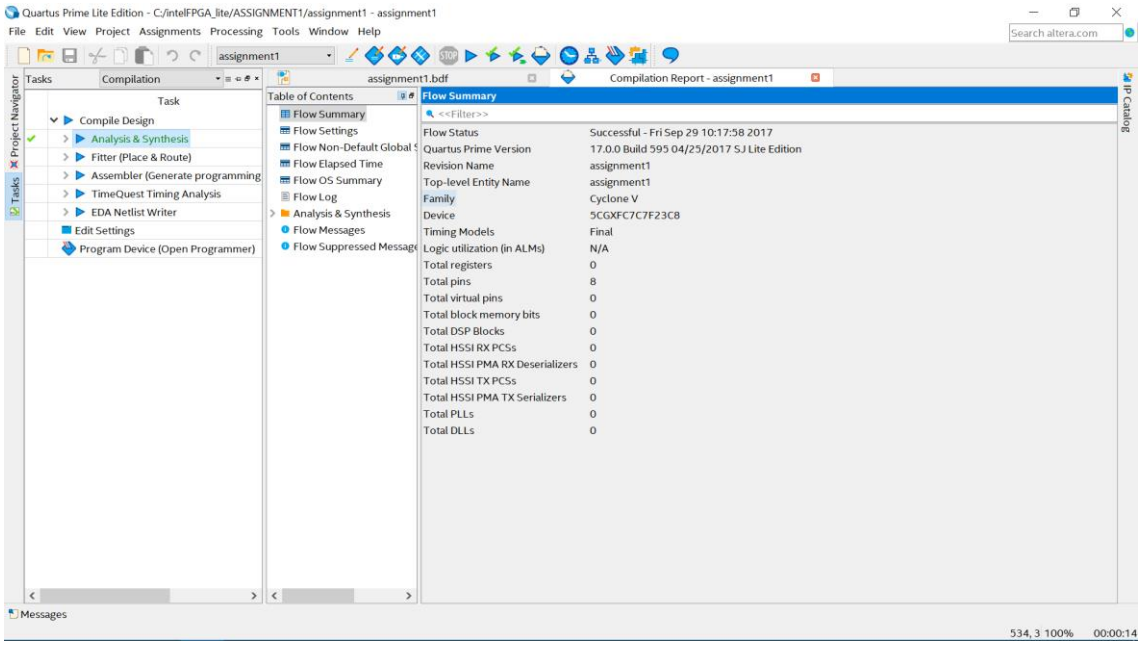


2) Design the circuit.

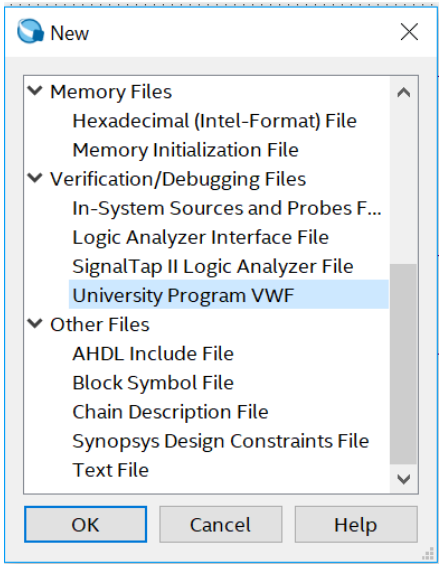


3)

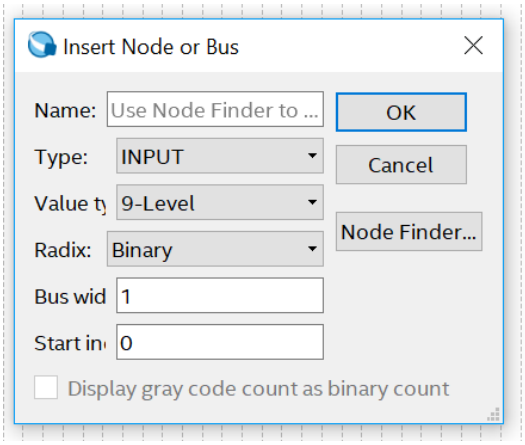
3) Compile the design.

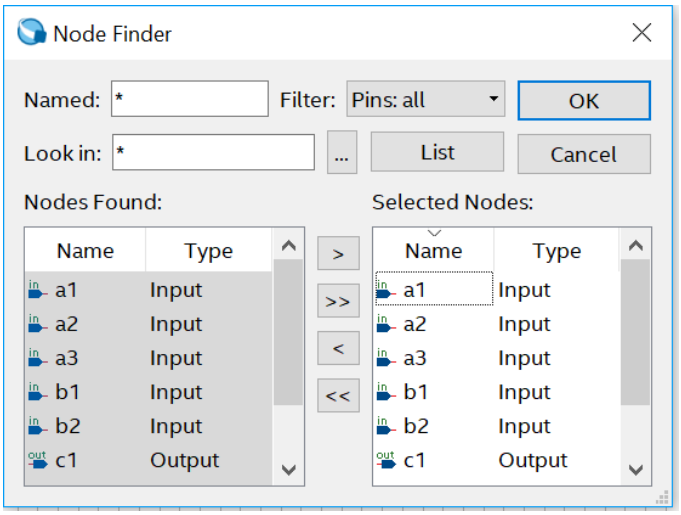


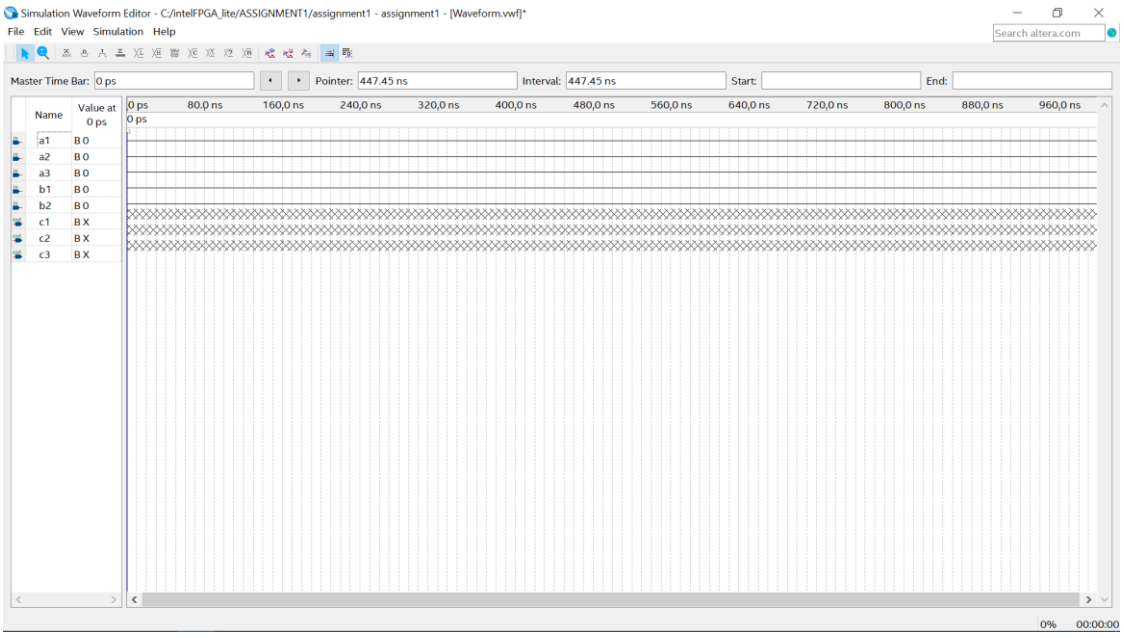
4) Make a new University Program VWF.



5) Insert all of the nodes.

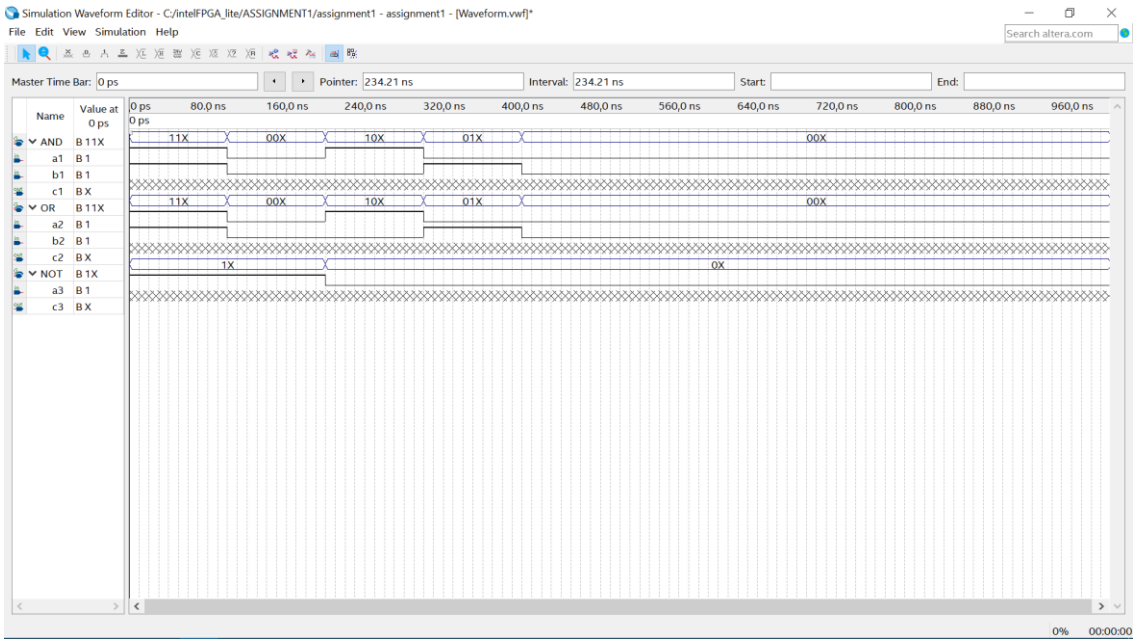




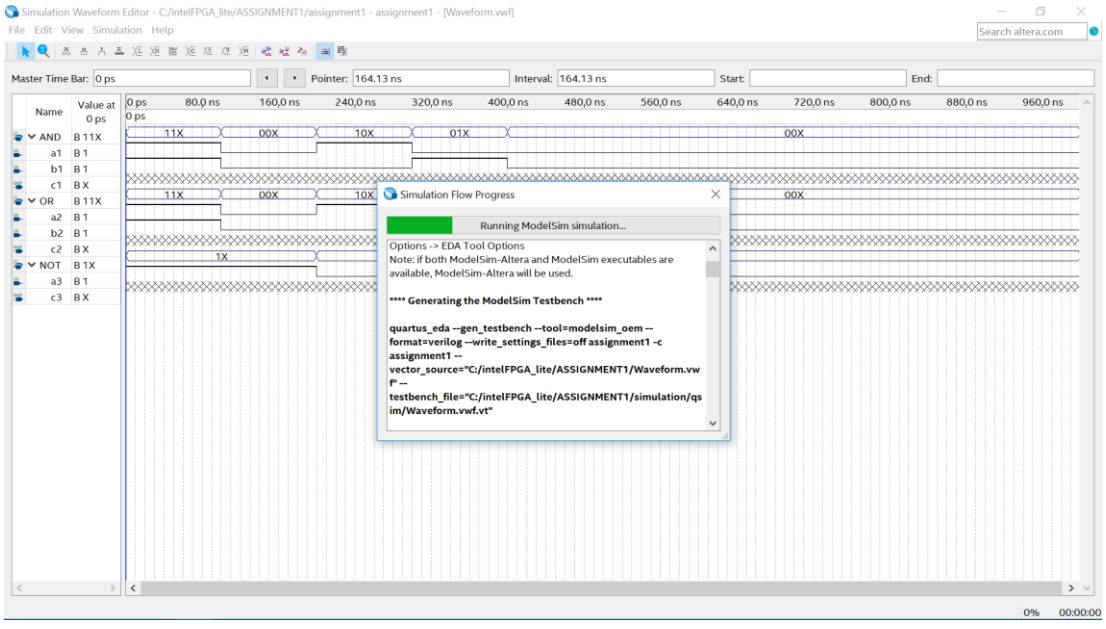


6) Change the order of the nodes and divide them into different groups.

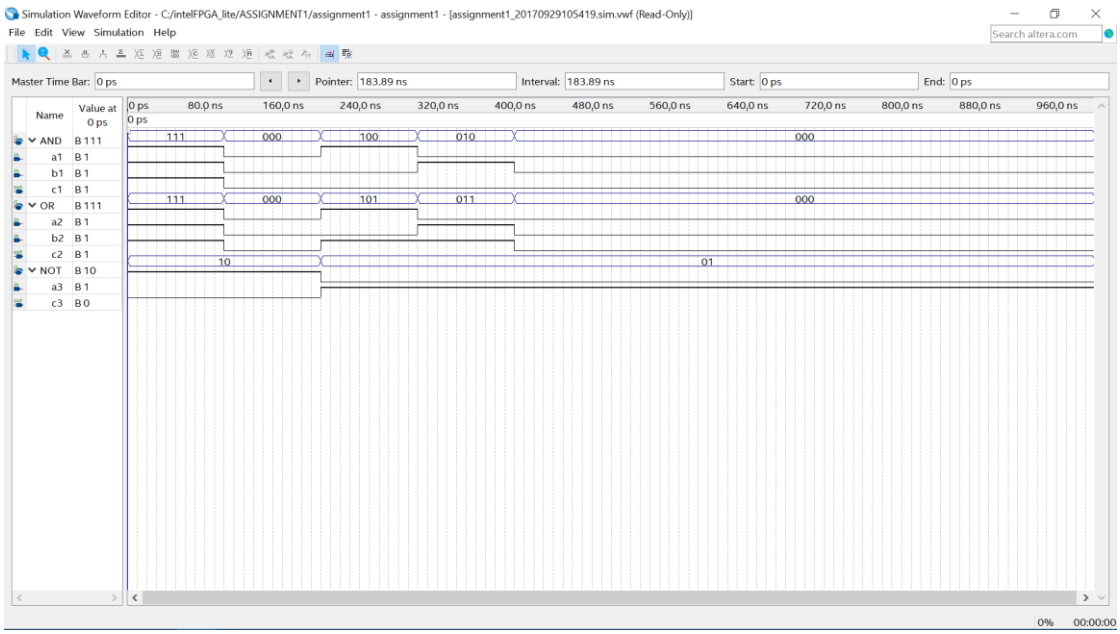
And then change the value of them at some specific point.



7) Run Functional Simulation.

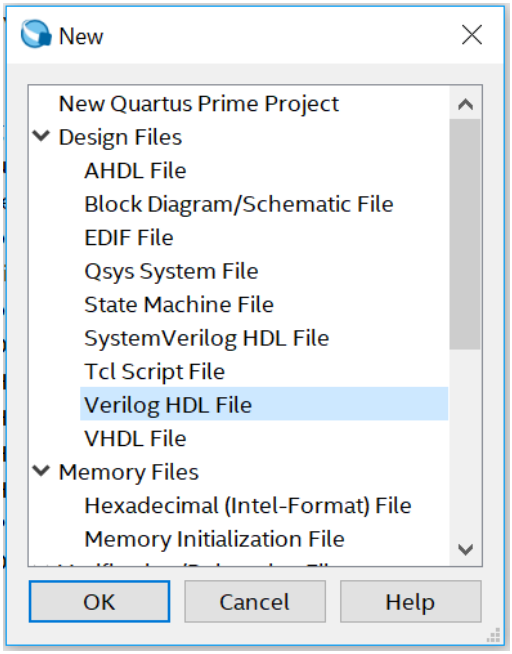


8) Get the wave form.

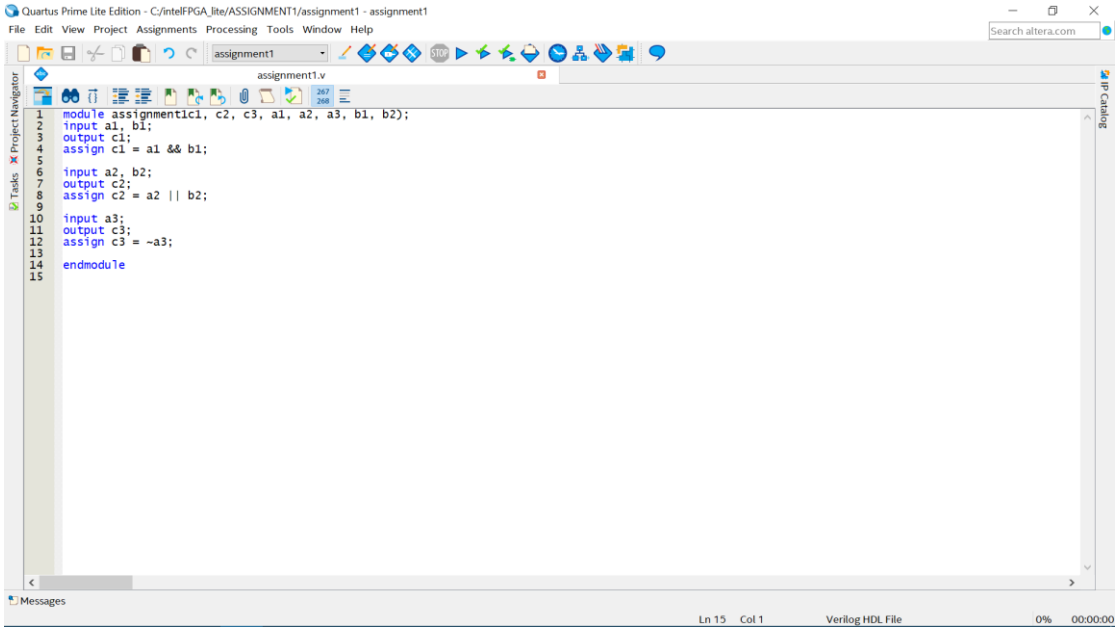


Steps on a VERILOG file:

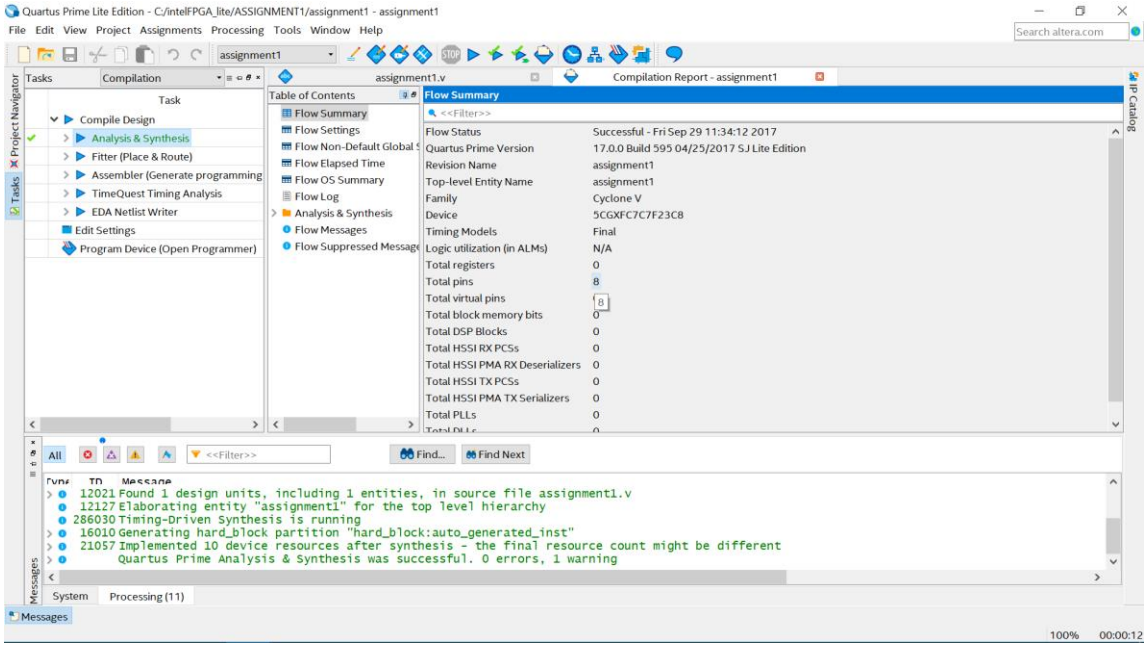
1) Make a new Verilog HDL File.



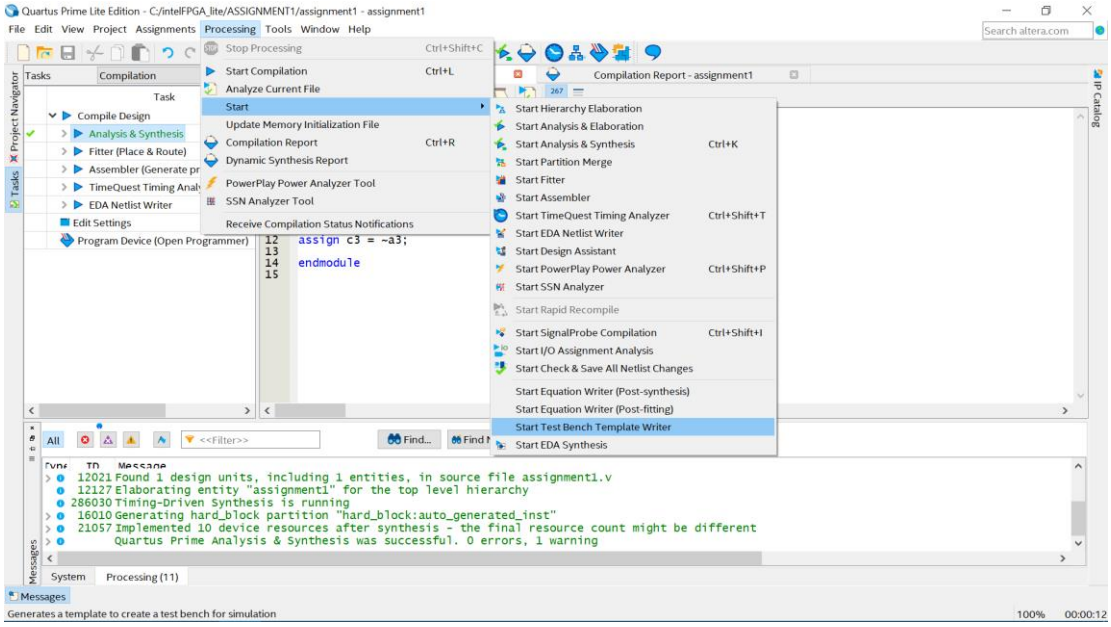
2) Design the code.



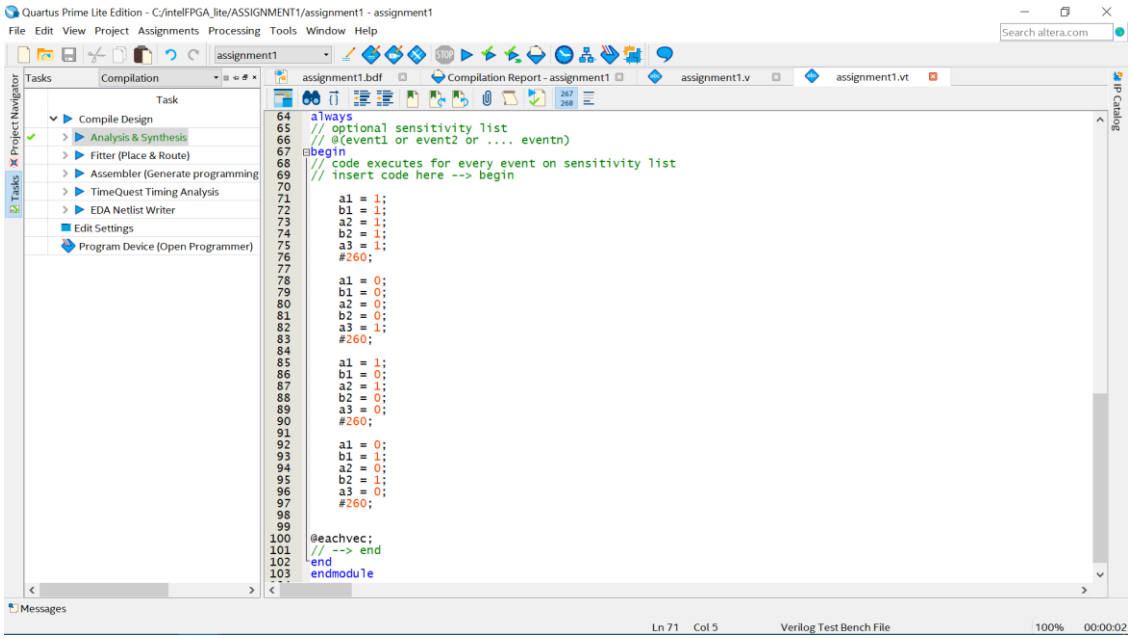
3) Compile the code.



4) Start Test Bench Template Writer

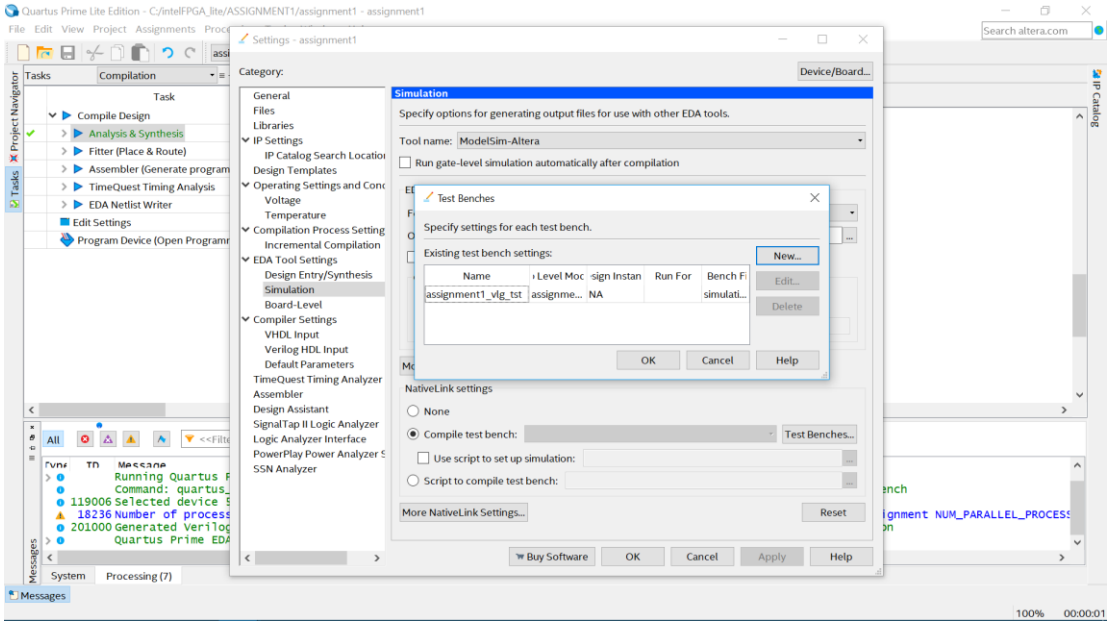


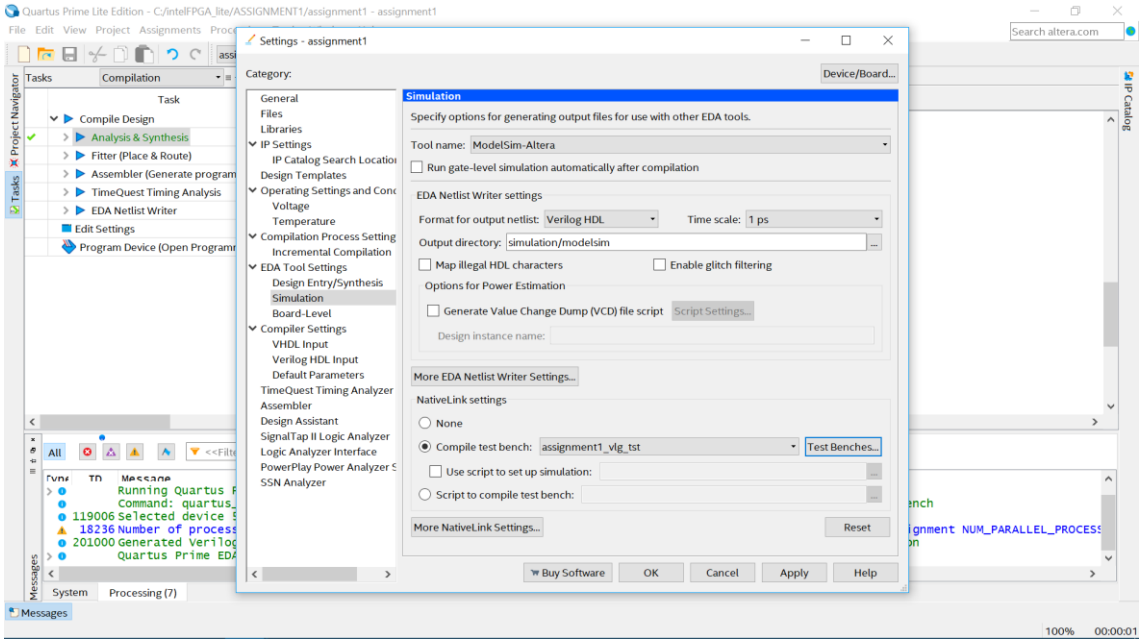
5) Open assignment.vt. Insert the code:



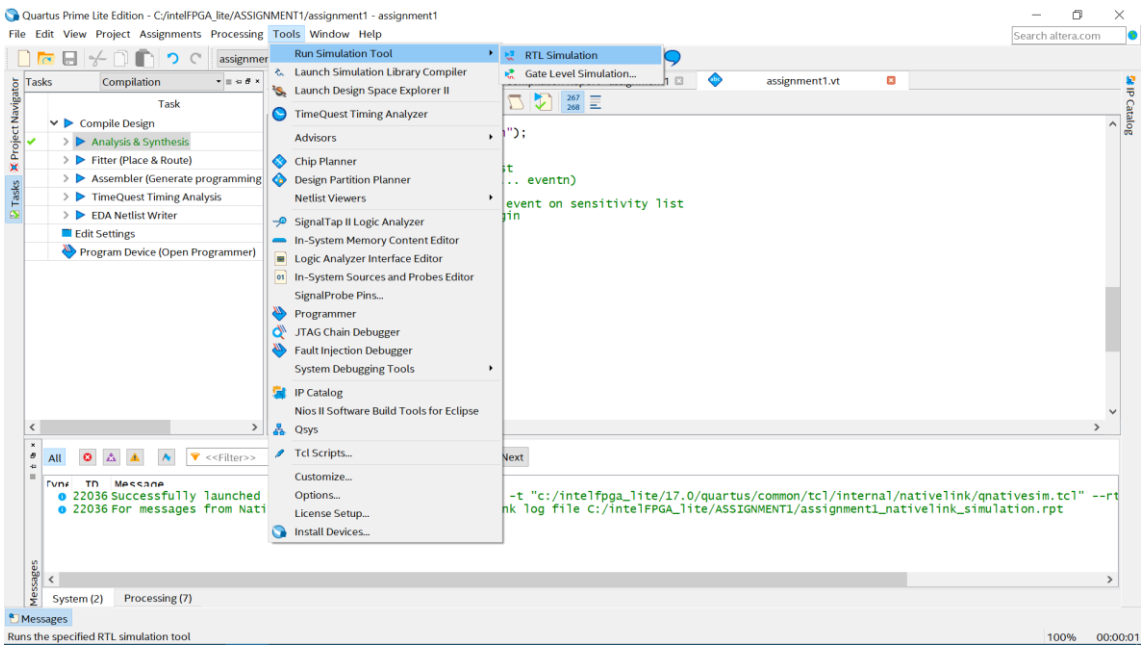
6) Compile the design again.

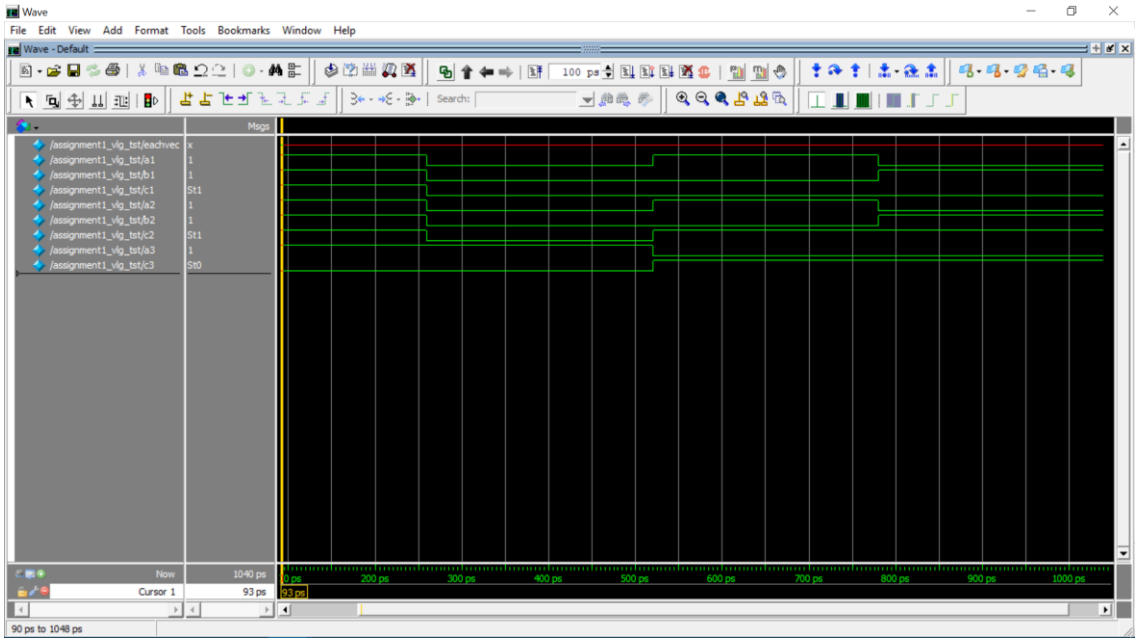
7) Change the simulation settings to compile the test bench.





8) Run RTL simulation and get the wave form.

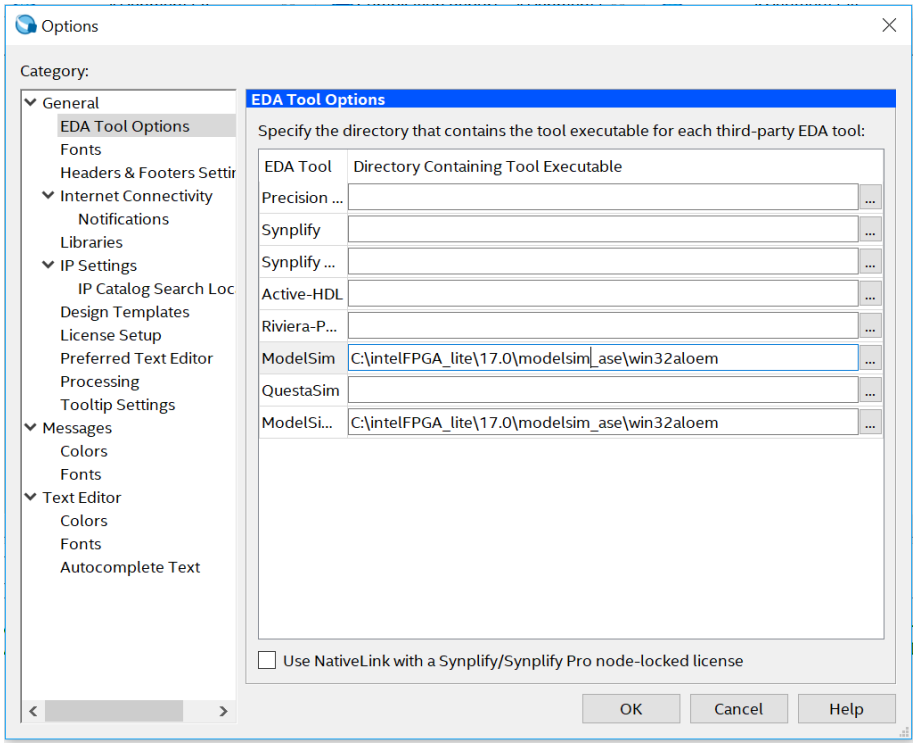

Quartus Prime Lite Edition - C:/intelFPGA_lite/ASSIGNMENT1/assignment1 - assignment1
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Run Simulation Tool
RTL Simulation
Gate Level Simulation...
Launch Simulation Library Compiler
Launch Design Space Explorer II
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
SignalProbe Pins...
Programmer
JTAG Chain Debugger
Fault Injection Debugger
System Debugging Tools
IP Catalog
Nios II Software Build Tools for Eclipse
Qsys
Tcl Scripts...
Customize...
Options...
License Setup...
Install Devices...
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
assignment1.vt
Project Navigator
IP Catalog
Messages
22036 Successfully launched
22036 For messages from Nati
System (2)
Processing (7)
Runs the specified RTL simulation tool
100%
00:00:01


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
/assignment1_vlg_tst/eachvec
/assignment1_vlg_tst/a1
/assignment1_vlg_tst/b1
/assignment1_vlg_tst/c1
/assignment1_vlg_tst/a2
/assignment1_vlg_tst/b2
/assignment1_vlg_tst/c2
/assignment1_vlg_tst/a3
/assignment1_vlg_tst/c3
Msgs
Now
1040 ps
Cursor 1
93 ps
0 ps
200 ps
300 ps
400 ps
500 ps
600 ps
700 ps
800 ps
900 ps
1000 ps
90 ps to 1048 ps

Problems and solutions

Problem 1: I once had a problem in the simulation part. Quartus showed that it couldn't find the modelsim tool.

Solution 1: I added the tool in the Option. Just like the picture below.



Problem 2: The BDF file was first made and was called assignment1.bwf.Then when I designed the verilog file, I named the file assignment1.v. The same name of files became a problem when I did the simulation to the verilog file.

Solution 2:

a) I can change the name of verilog file to solve the problem.

b) I think it's better if I just build a new project.