

Tutorial 1

1 Introduction

1.1 Target

- Have knowledge of FPGA technology and How to design an **Digital circuit** by using Verilog.
- Be able to use Quartus to describe a circuit and test it.

1.2 Requirement

- Revise lesson 3.
- Read the tutorial and finish all requirement before going to class.
- Download and setup Quartus (see).
- Do all exercices in Part 3 and take more practice (froom books or Internet) after class.

1.3 Content

- Setup Quartus Lite 16.1
- Introduce to FPGA technology and FPGA design flow.
- Design simple **Digital circuits** by using block diagram.
- Analyze, synthesize and simulate your design by using **Quartus Verification**.

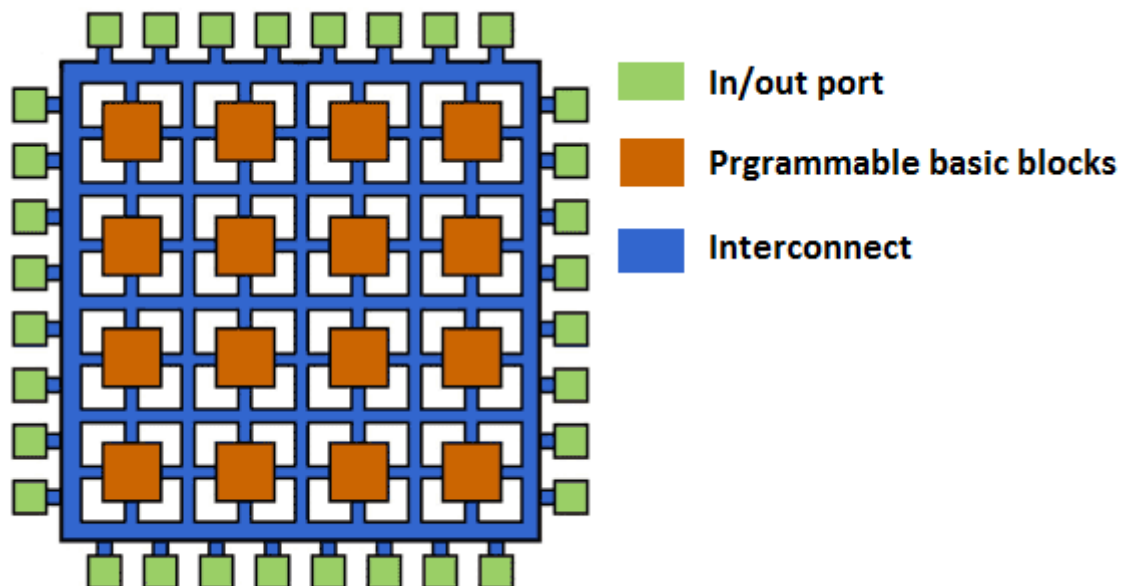
2 Quick Start Tutorial

2.1 What is FPGA?

2.1.1 Introduction to ASIC's world

An application-specific integrated circuit (ASIC), is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. ASIC is a chip that can be designed by an engineer with no particular knowledge of semiconductor physics or semiconductor processes. The ASIC vendor has created a library of cells and functions that the designer can use without needing to know precisely how these functions are implemented in silicon. The ASIC vendor also typically supports software tools that automate such processes as synthesis and circuit layout. The ASIC vendor may even supply application engineers to assist the ASIC design engineer with the task. The vendor then lays out the chip, creates the masks, and manufactures the ASICs.

The gate array is an ASIC with a particular architecture that consists of rows and columns of regular transistor structures. Each basic cell, or gate, consists of the same small number of transistors which are not connected. In fact, none of the transistors on the gate array are initially connected at all. The reason for this is that the connection is determined completely by the design that you implement.



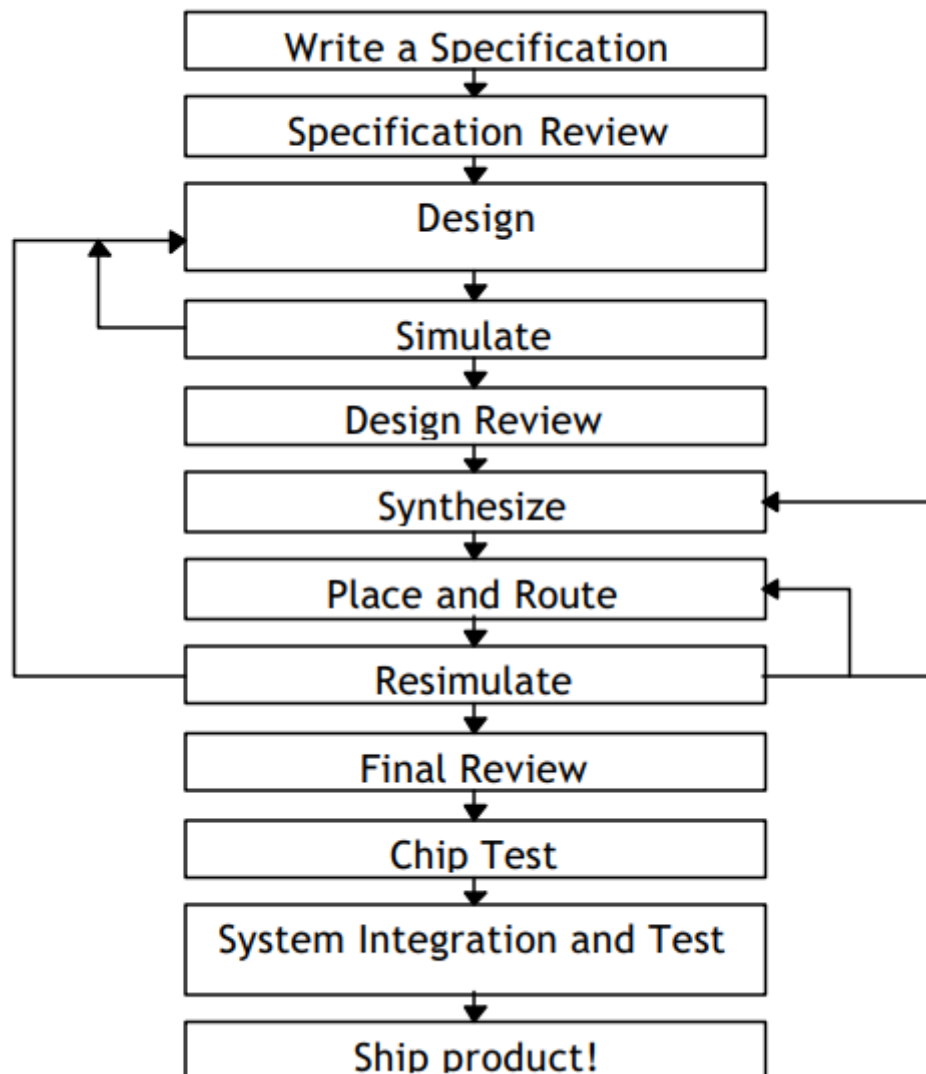
Field Programmable Gate Arrays (FPGAs) are called this because they are structured very much like a gate array ASIC. This makes FPGAs very nice for use in prototyping ASICs, or in places where an ASIC will eventually be used. For example, an FPGA may be used in a design that needs to get to market quickly regardless of cost. Later an ASIC can be used in place of the FPGA when the production volume increases, in order to reduce cost.

Mini quiz: Compare FPGAs and ASIC

FPGAs

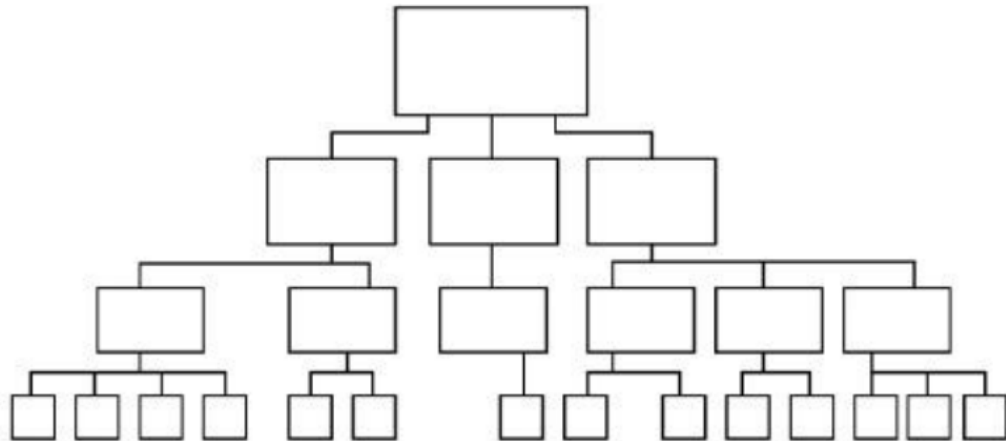
ASIC

2.1.2 FPGA's design flow



1. **Write a Specification:** A specification allows each engineer to understand the entire design and his or her piece of it. It allows the engineer to design the correct interface to the rest of the pieces of the chip. It also saves time and misunderstanding.

2. **Design - Top-Down method:** Top-down design is the design method whereby high level functions are defined first, and the lower level implementation details are filled in later.



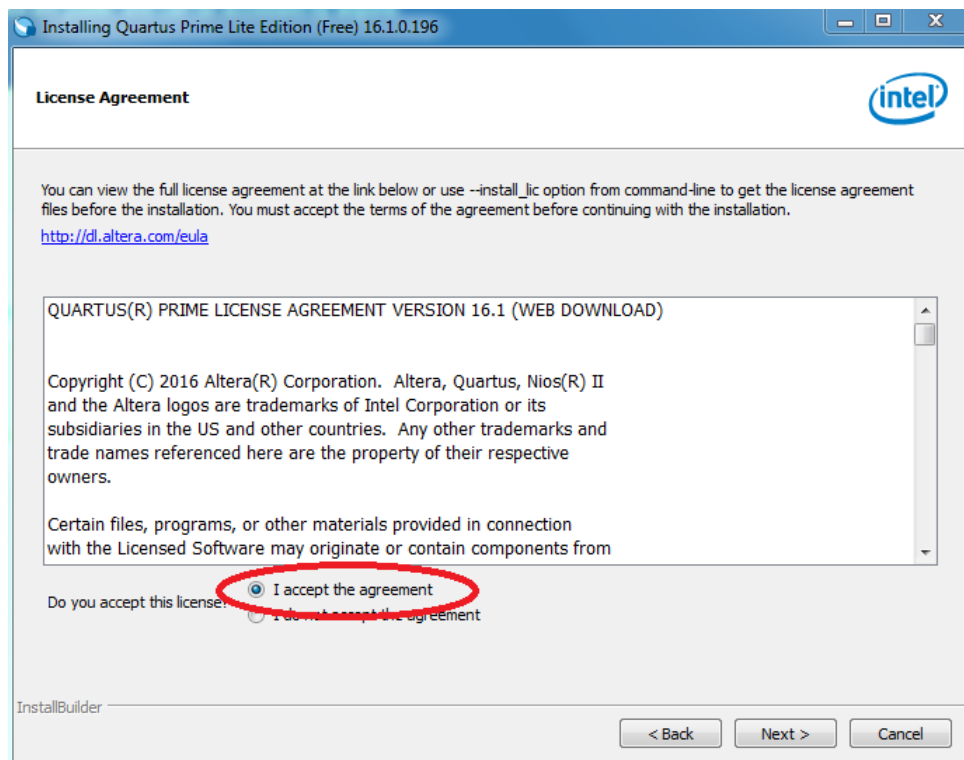
The top level block represents the entire chip. Each lower level block represents major functions of the chip. Intermediate level blocks may contain smaller functionality blocks combined with gate-level logic. The bottom level contains only gates and macrofunctions which are vendor-supplied high level functions. Fortunately, schematic capture software and hardware description languages used for chip design easily allows use of the top-down design methodology.

3. **Simulate:** This step is important initially in order to get as many bugs out of the device as possible and to determine that the chip will work correctly in your system. During the first phases of simulation, you shouldn't be very concerned about timing because it will change as the design changes. In fact, the final timing will not be known precisely until the layout is complete.
4. **Synthesize:** Turn an abstract form of desired circuit behavior into a design implementation in terms of logic gates.
5. **Place and Route:** reconfigure the Interconnect system to connect Programmable basic blocks together.
6. **Resimulate:** This step is a process that looks at a synchronous design and determines the highest operating frequency of the design which does not violate any setup and hold times. You can also use the analysis softwares to specify a specific frequency, and the tool will list all paths that violate the timing requirements. These paths can then be adjusted to meet your requirements.

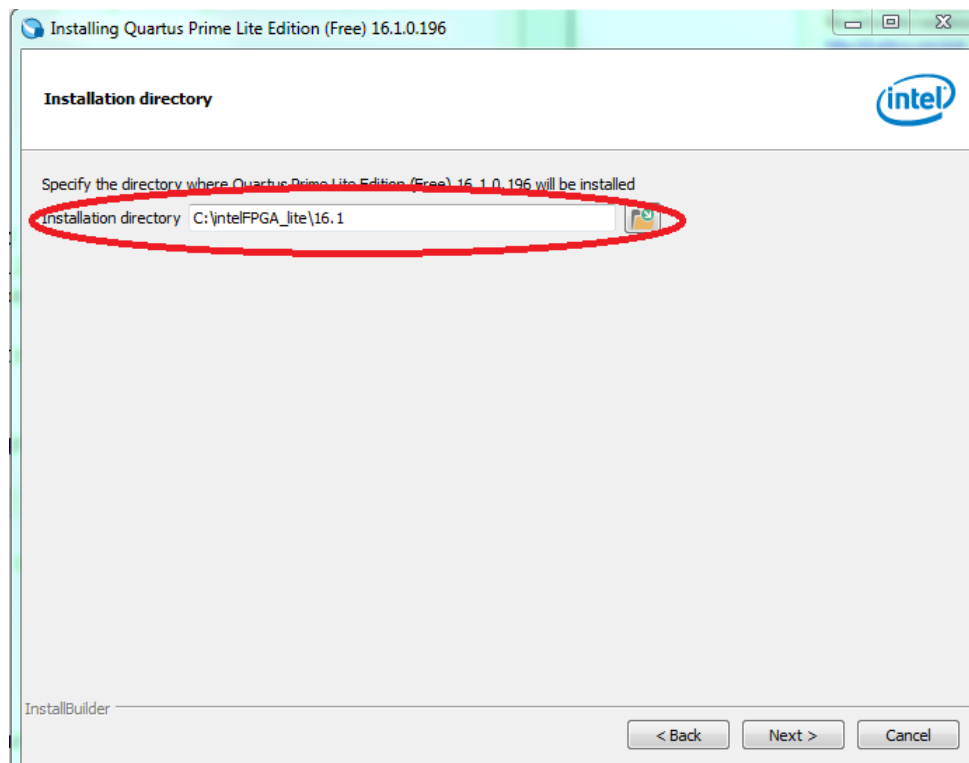
2.2 Installation guide (Quartus)

Quartus is programmable logic device design software produced by Altera. Quartus II enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Quartus includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation. Follow these instructions below to install Quartus Lite (free edition).

1. Get Quartus Lite 16.1 from download link <https://goo.gl/Ec3wv9>
2. Unzip downloaded file (*Quartus-lite-16.1.0.196-windows.tar*).
3. Run "setup.bat" to begin the installation and wait until the installation appearance appears.
4. After the installation appearance appears, follow these steps below:
 - (a) (Click) next (button)
 - (b) Select the radio button "I accept the agreement" and then next.



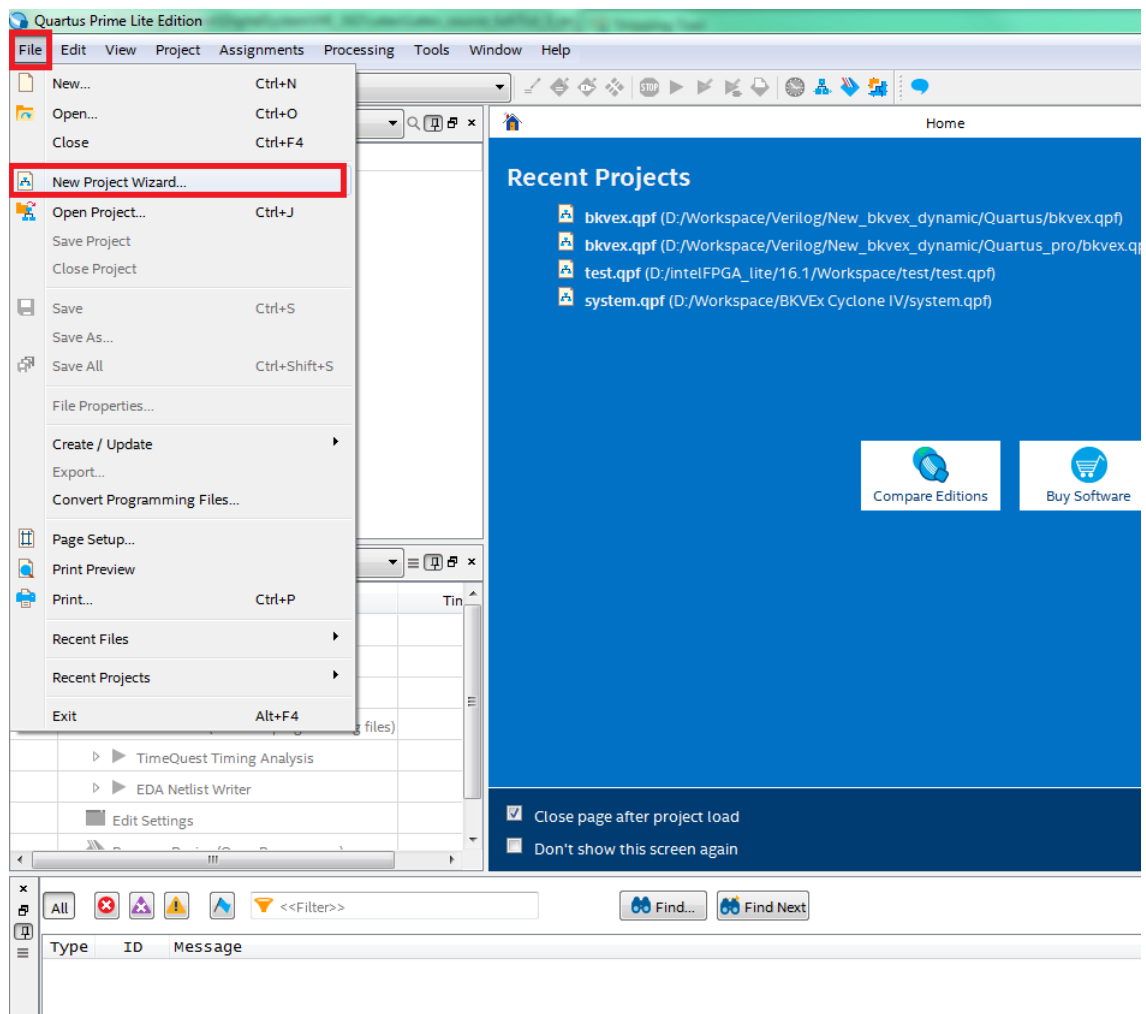
- (c) Select your Installation destination by completing the text box (optional) and then next.



(d) Next until the installation finishes.

2.3 Create a project in Quartus

1. Open Quartus.
2. On the very first appearance, select (Menu)File → New Project Wizard.



3. A new screen appears in order to create your project. Click "Next" first.
4. Select your project's directory by completing the first textbox and the project's name in the second one. And then, click "Next".

Directory, Name, Top-Level Entity project's directory

What is the working directory for this project?

What is the name of this project?

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.
project's name

< Back Next > Finish Cancel Help

5. Click "Next" for the two next appearances.
6. On fifth appearance, in the "Family" drop-down box, select Cyclone IV GX. On the "Name filter" textbox, put down "EP4CGX150DF31C7". Click Finish then.

Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.
 You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family
 Family: Cyclone IV GX
 Device: All

Target device
☒ Auto device selected by the Fitter
☐ Specific device selected in 'Available devices' list
☐ Other: n/a

Show in 'Available devices' list
 Package: Any
 Pin count: Any
 Core speed grade: Any
 Name filter: EP4CGX150DF31C7
☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	GXB Transmitter Channel PMA	GXB R
EP4CGX150DF31C7	1.2V	149760	508	464	8	8

< Back Next > Finish Cancel Help

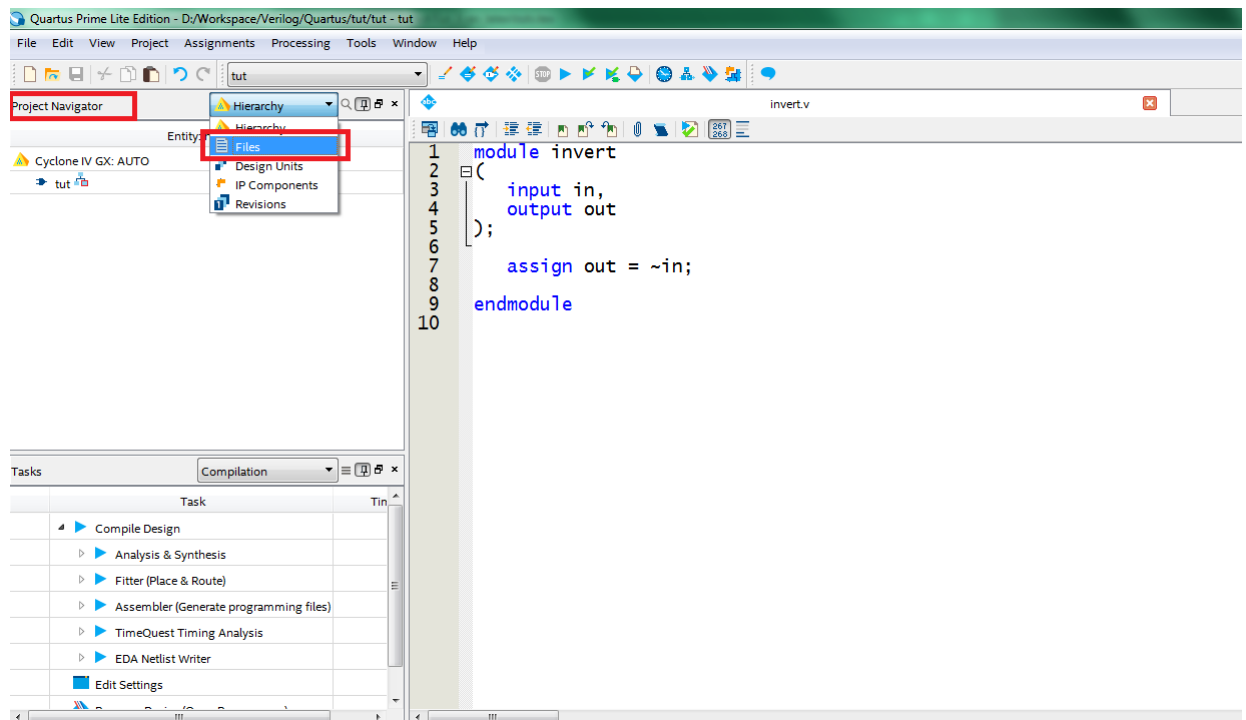
2.4 Create HDL design

Read section 3.

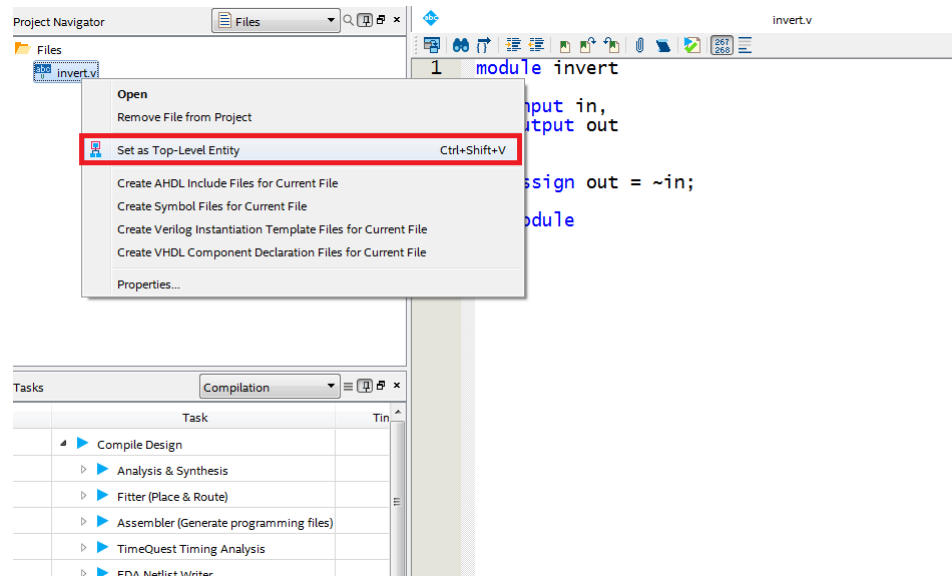
2.5 Synthesize your design

When you already finish your design, then begin to synthesize it.

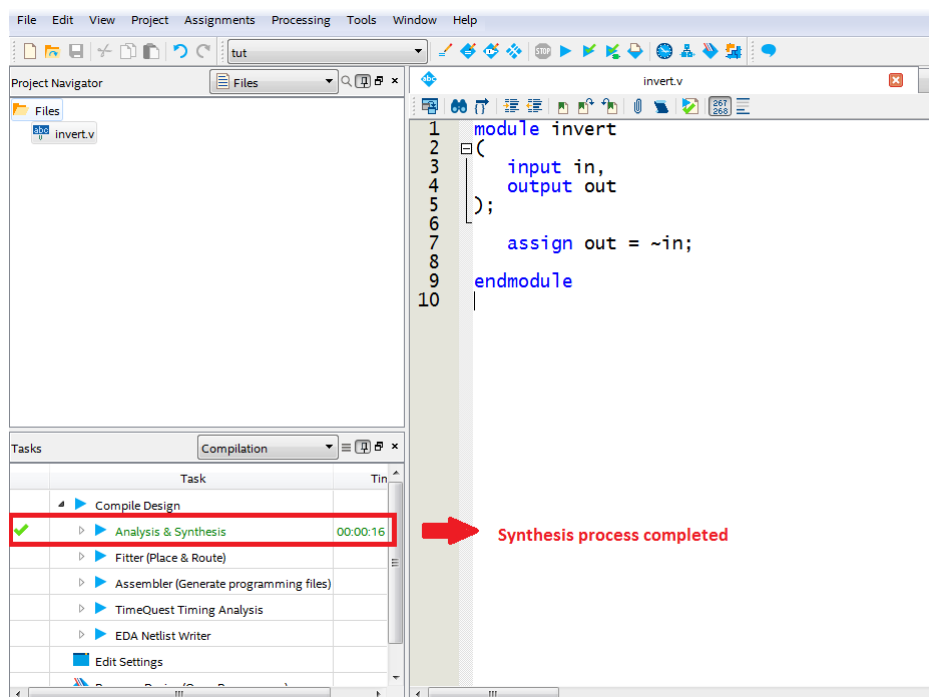
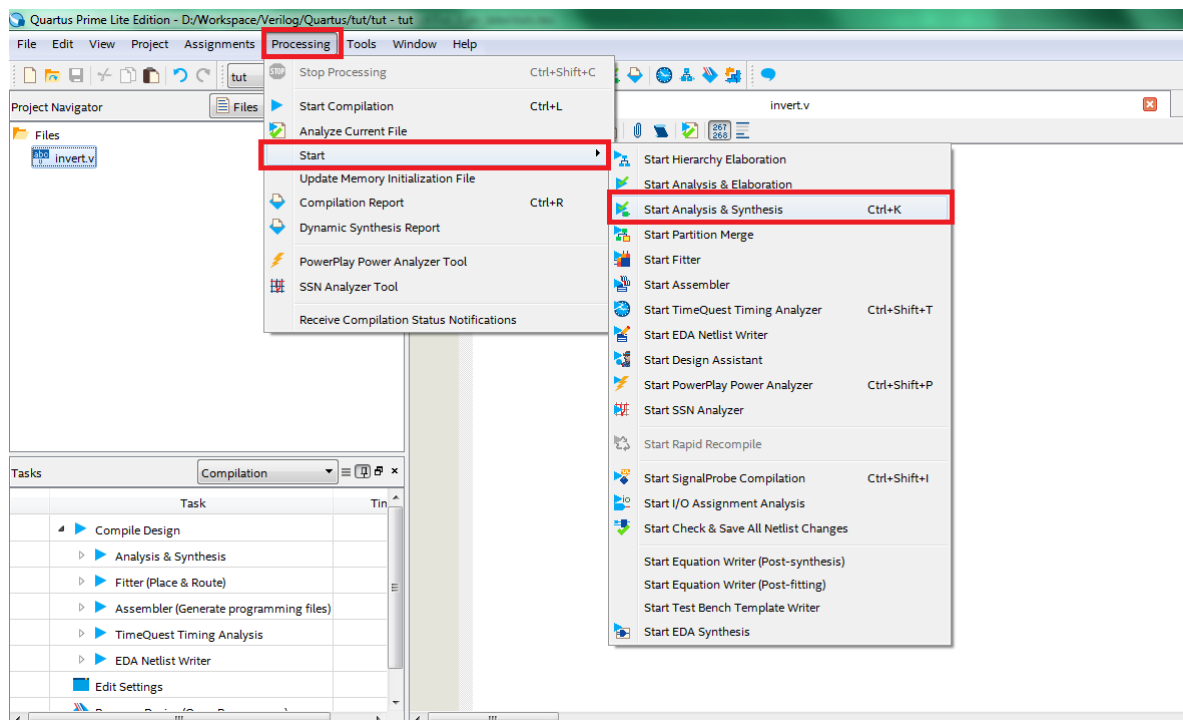
1. On the Project Navigator panel, select "File" in the drop-down box.



2. Right-click on top module file and select "Set as Top-Level Entity"

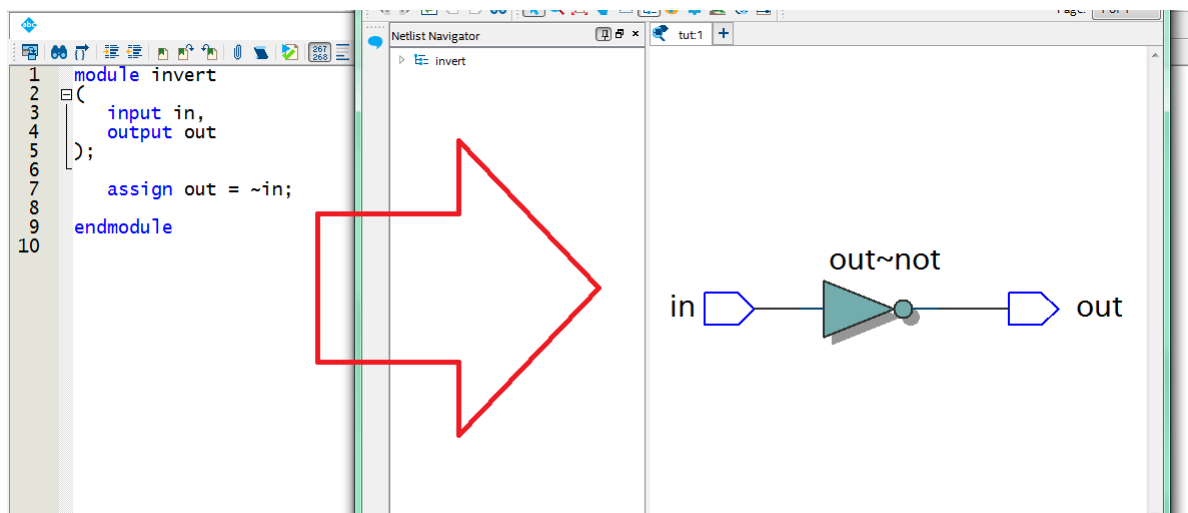
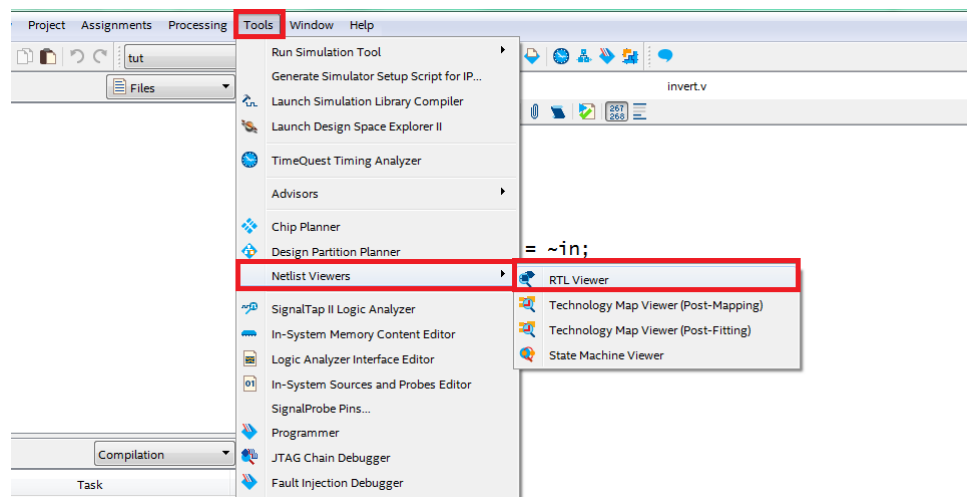


3. Select (Menu)Processing → Start → Start Analysis & Synthesis. And then, wait until the Synthesis process finishes.



2.6 Show netlists viewers

You can show you designed circuit after synthesize by select:
(Menu)Tools → Netlist Viewers → RTL Viewer



We finally finish the preparation, now let's do a few warm-up.

3 Warm-up (2 pts)

(Revise chapter 3 before read this section)

We've already mentioned 3 module styles in class, that is:

- Structural model.
- RTL model.
- Behavioral model.

We also have two ways to implement a digital circuit, this is:

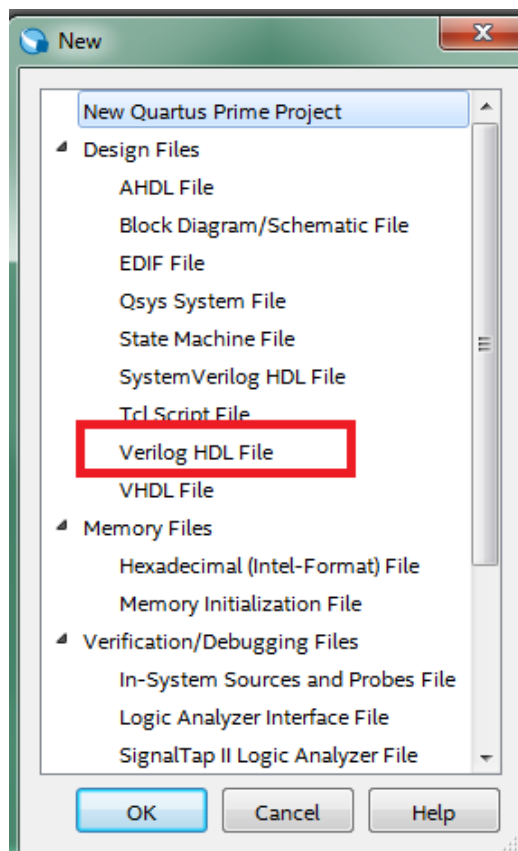
- Describe circuit by using HDL (hardware description language).
- Design circuit by using block diagram.

In this tutorial, we're going to create majority circuit by using Structural model.

3.1 Describe circuit by using HDL

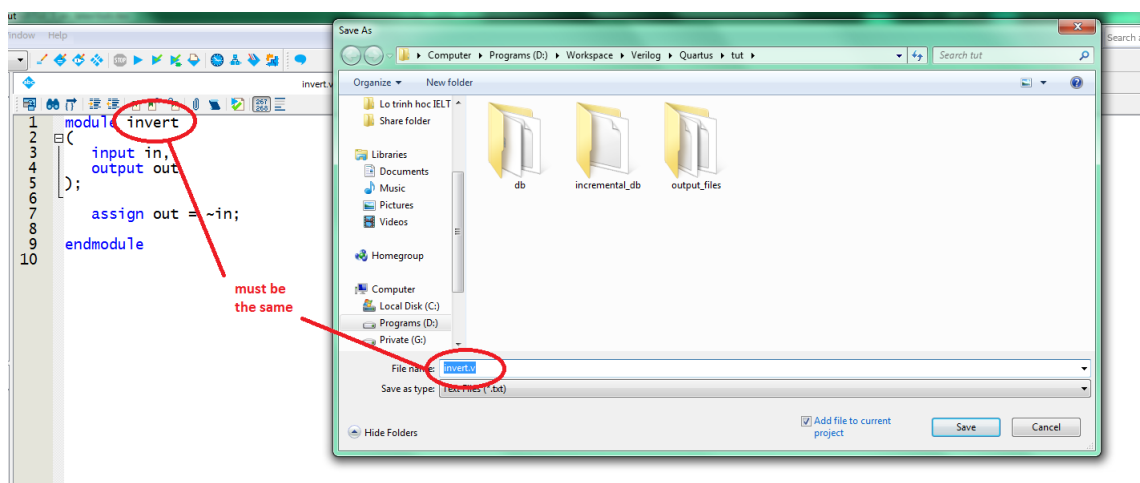
3.1.1 Create Verilog HDL file

Select (Menu)File → New → Verilog HDL File



Select (Menu)File → Save to save Verilog HDL File.

Note: The file's name must be the same as module's name.



3.1.2 Write a specification

Complete the truth table below for majority circuit:

V1	V2	V3	major

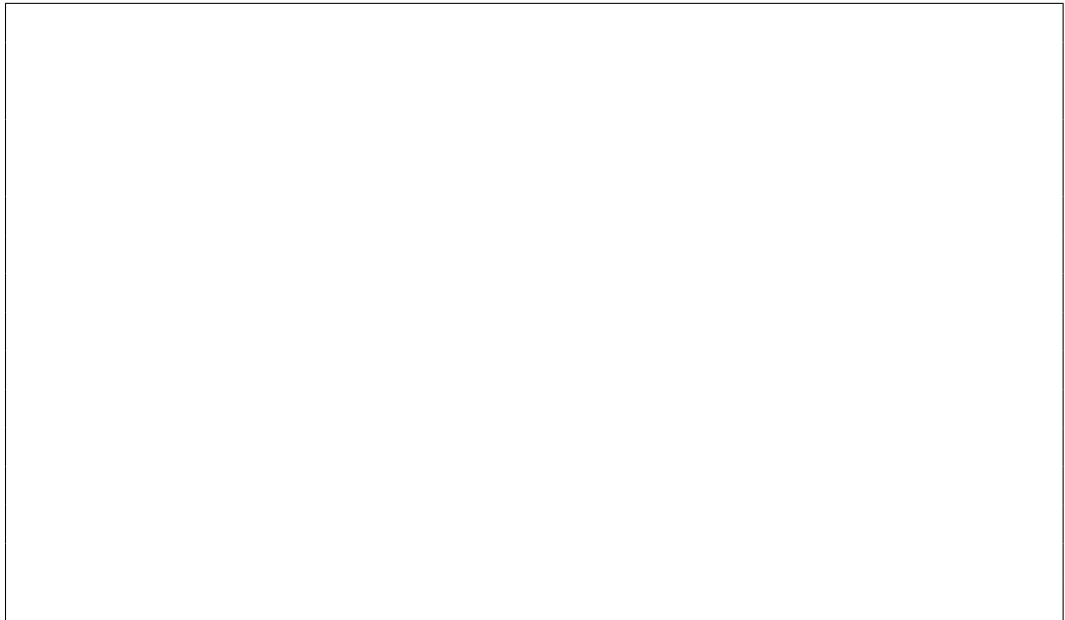
3.1.3 Design circuit

1. Using Structural

- (a) From the truth table above, write down the corresponding Boolean expression.

- (b) Minimize the expression by using Karnaugh map or the boolean theorems.

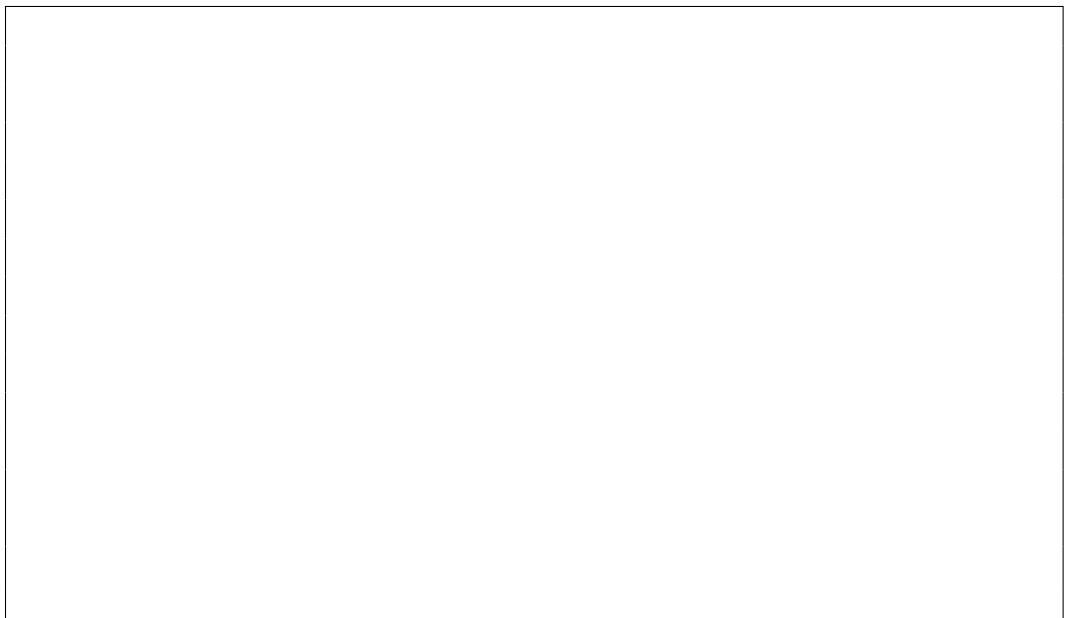
- (c) Implement your minized expression by using basic gates (and, or, etc)



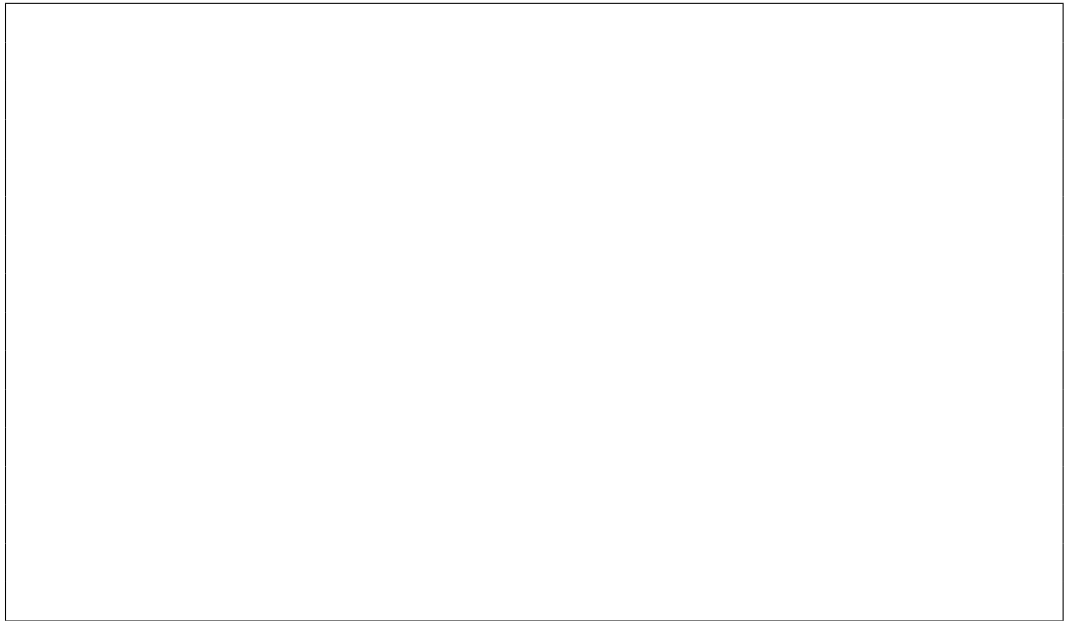
- (d) Create Verilog HDL File circuit under structural model and fill it up with the prototype below (use your minized expression):

```
module majority_structural (major, V1, V2, V3) ;  
// Enter your code here  
endmodule
```

Synthesize and then paste the netlists viewers into the box below:



- (e) Create another Verilog HDL File (named majority_structural_unminimized) describes majority circuit under structural model (use your bare expression in (a)). Synthesize and then paste the netlists viewers into the box below:



(f) Compare the two result from (e) and (d).

3.2 Describe circuit by using HDL and block diagram

1. In this circuit, we're using "AND" gates and "OR" gate, so we describe its first by HDL. Put down these code below into Verilog HDL File:

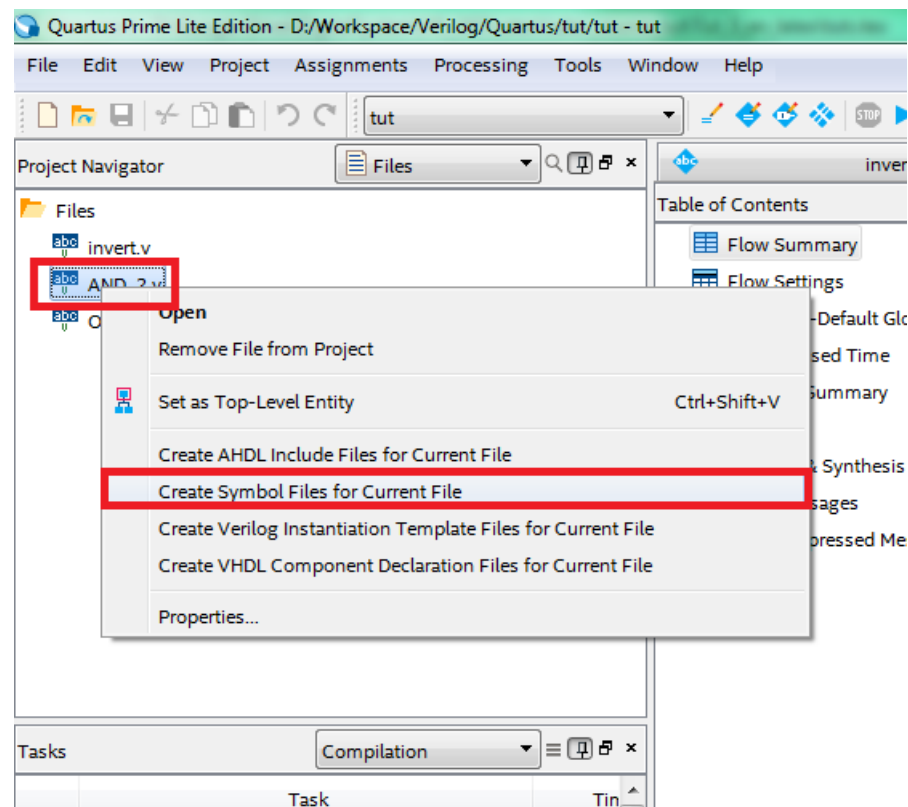
```
module AND_2(out, in_1, in_2);  
    output out;  
    input in_1, in_2;  
    assign out = in_1 & in_2;  
endmodule
```

```
module OR_2(out, in_1, in_2);  
    output reg out;  
    input in_1, in_2;  
    always @(in_1, in_2) begin  
        out = in_1 | in_2;  
    end  
endmodule
```

And then, synthesize both modules separately.

2. Now, we transform its to block symbol to using in block diagram.

(a) Right click on the file that you expect to create symbol and select *Create Symbol Files for Current File*



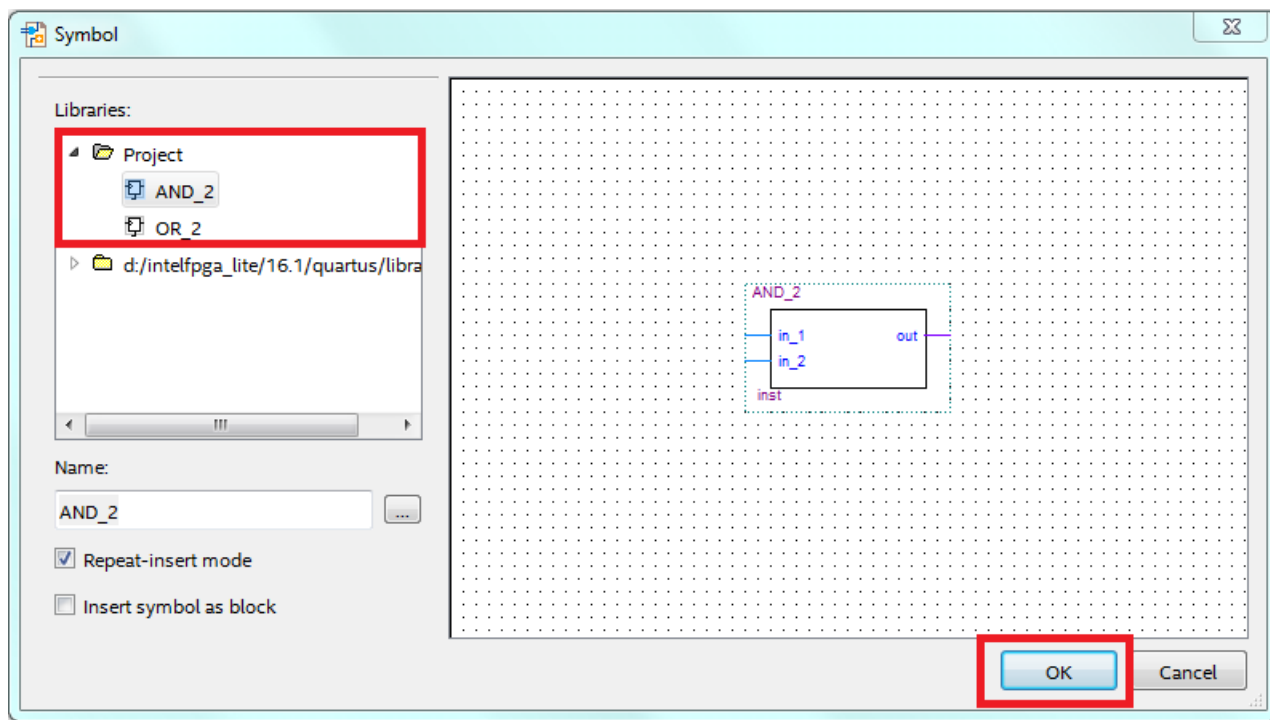
(b) Do the same job for the rest.

3. Create block diagram file

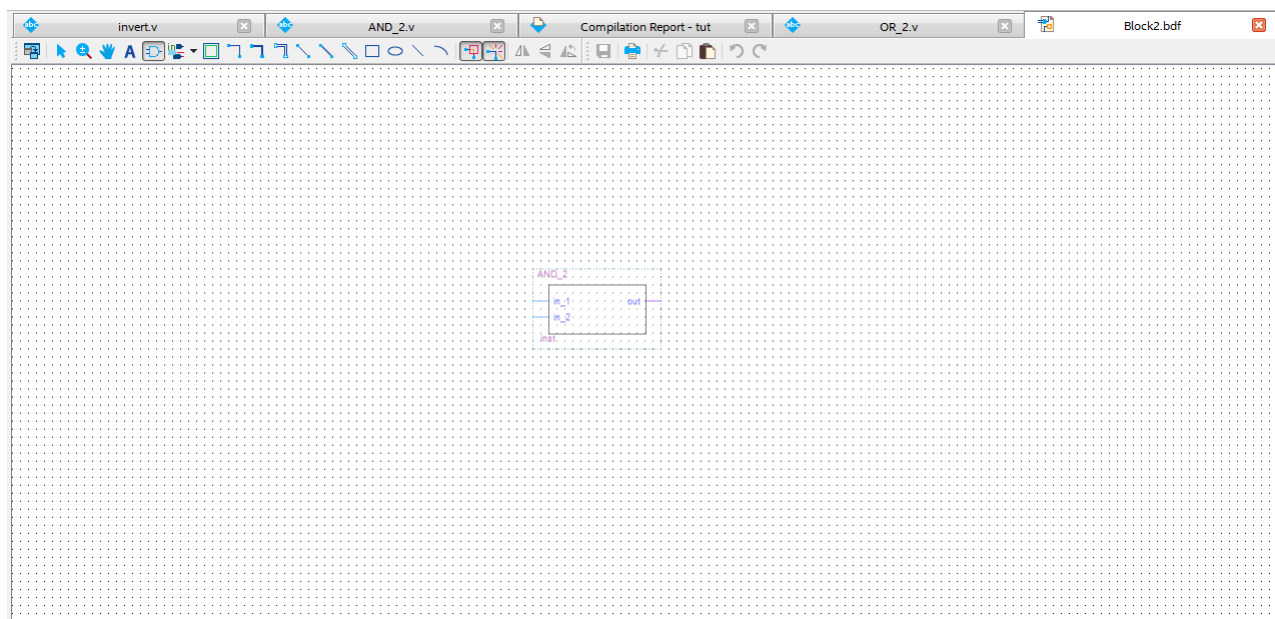
- (a) Select (Menu)File → New → Block Diagram/Schematic File.
- (b) A new appearance appears that help you to design circuit by using symbol block. There're some primary tools that are shown in the picture below.



- 1> Selection Tool: select the different components in your design.
 - 2> Hand Tool: Move between different zones in you design.
 - 3> Symbol Tool: Add these components that have been created.
 - 4> Pin Tool: Add input/output ports.
 - 5> Orthogonal Node Tool: Create connections between different components.
- (c) Add components: Select Symbol Tool → Libraries → Project. Select the appropriate component and click "OK".

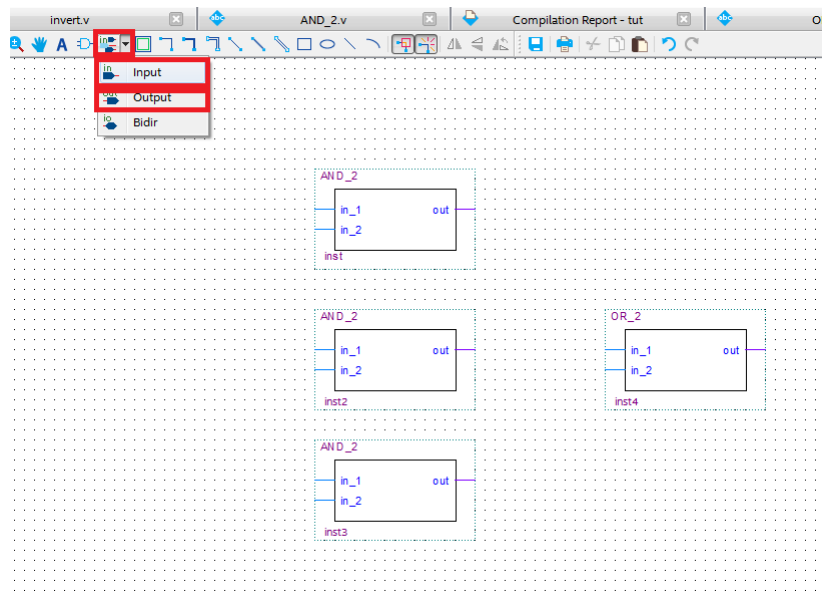


(d) Place your selected component by clicking on design's zone.

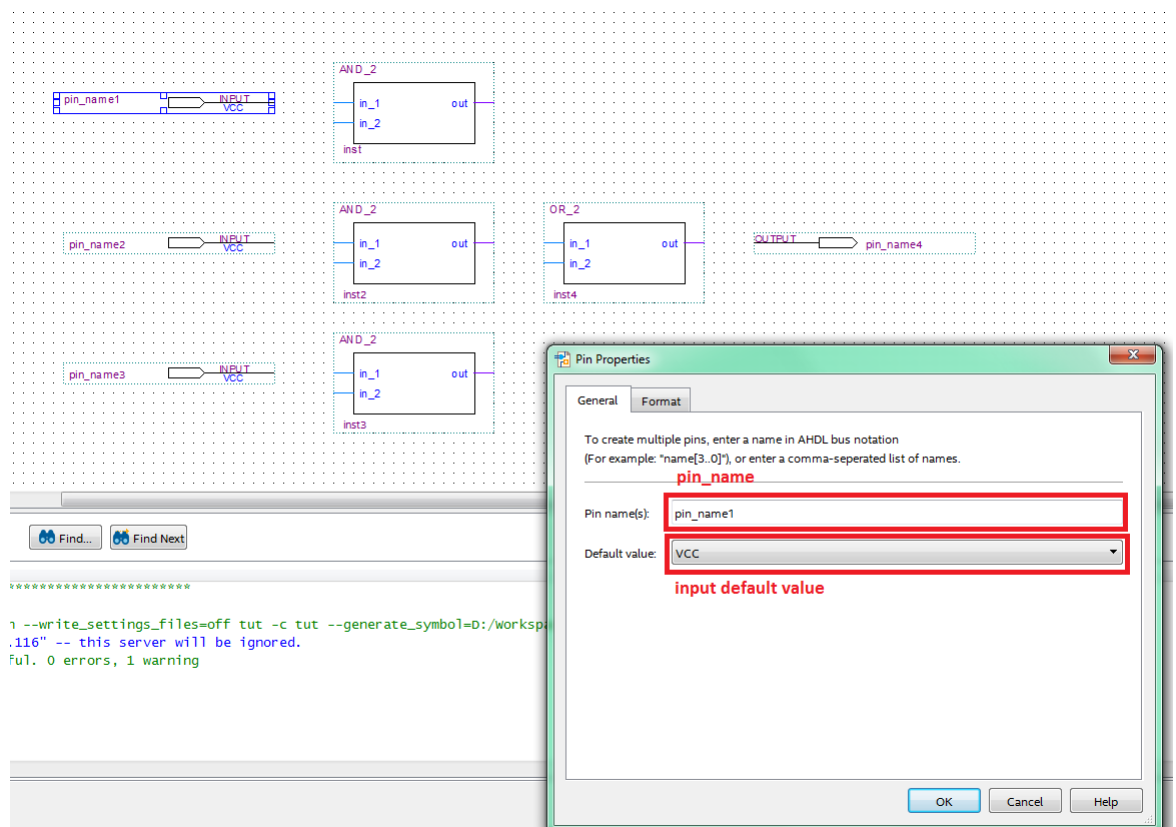


(e) And then, we do the same thing for the majority circuit.

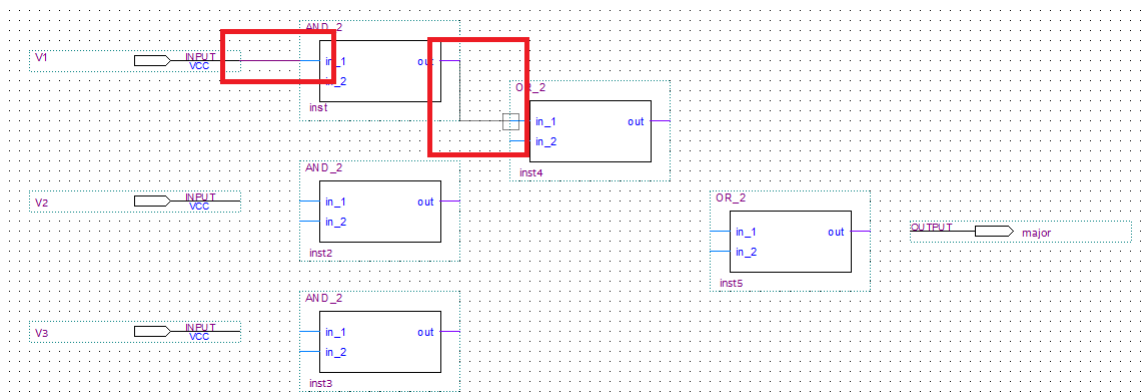
(f) Place input/output by selecting it in Pin Tool.



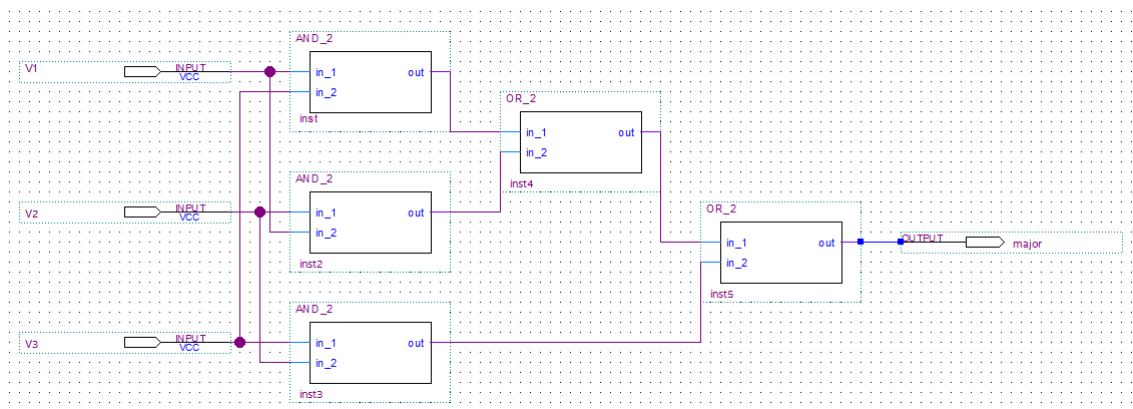
(g) Double click on pin to setup its name and its default value (if it's possible)



(h) Wire the components by selecting Orthogonal Node Tool. Click-and-hold the left mouse's button on the first point and leave it on the destination.



(i) And then, this is the result...

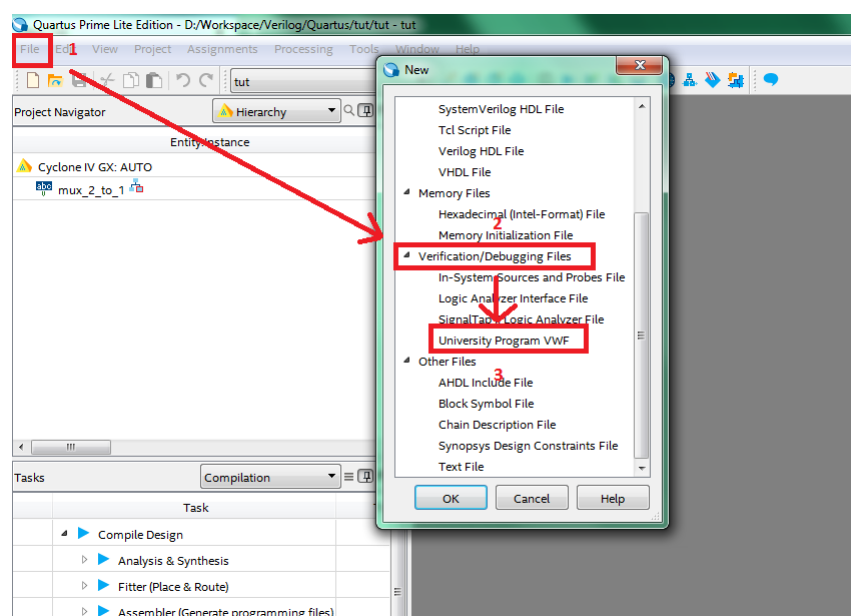


(j) Save your design and we have a *.bdf file. Set as top-module and synthesize it.

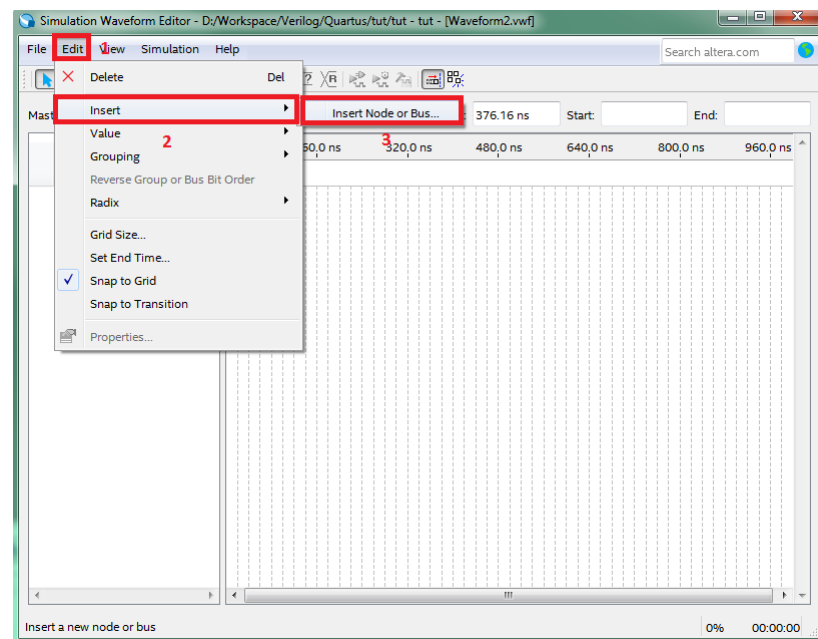
3.3 Simulate your design

After having a abstract circuit, you have to simulate it to determine that the chip will work correctly. Follow these steps below:

1. **Create VWF file:** select File → Verification/Debugging Files → University Program VWF



2. **Configure to integrate simulation:** When a new window appears, select Edit → Insert → Insert Node or Bus...

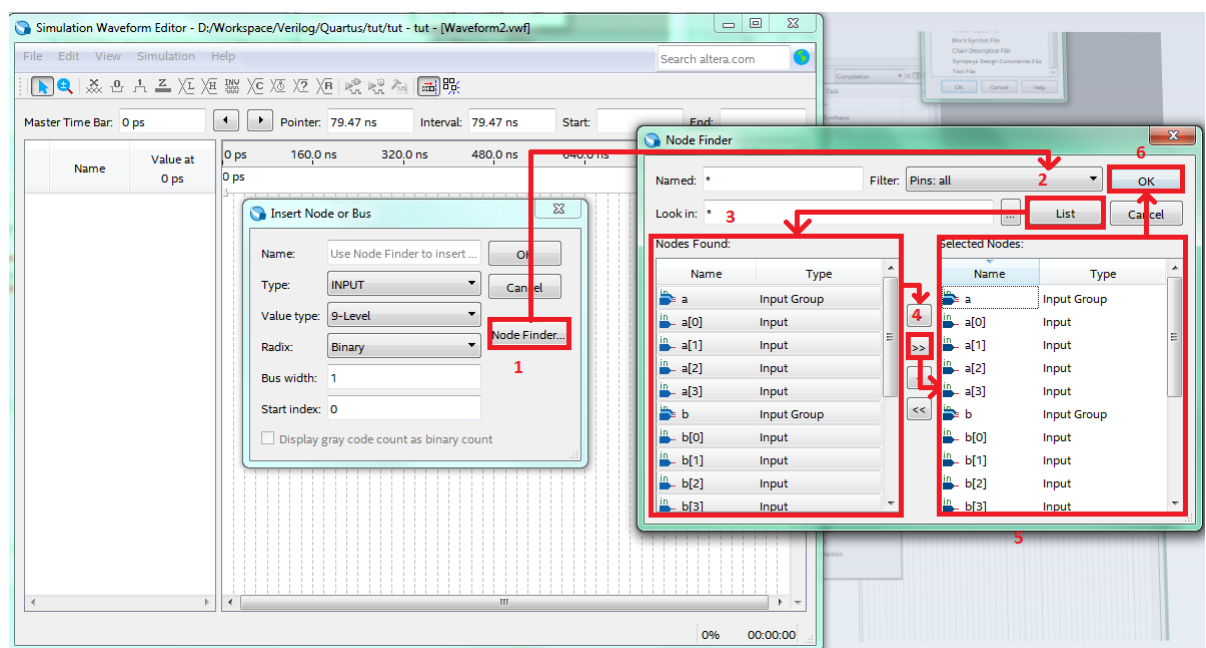


3. When "Insert Node or Bus" window appears, click on "Node Finder..." button.

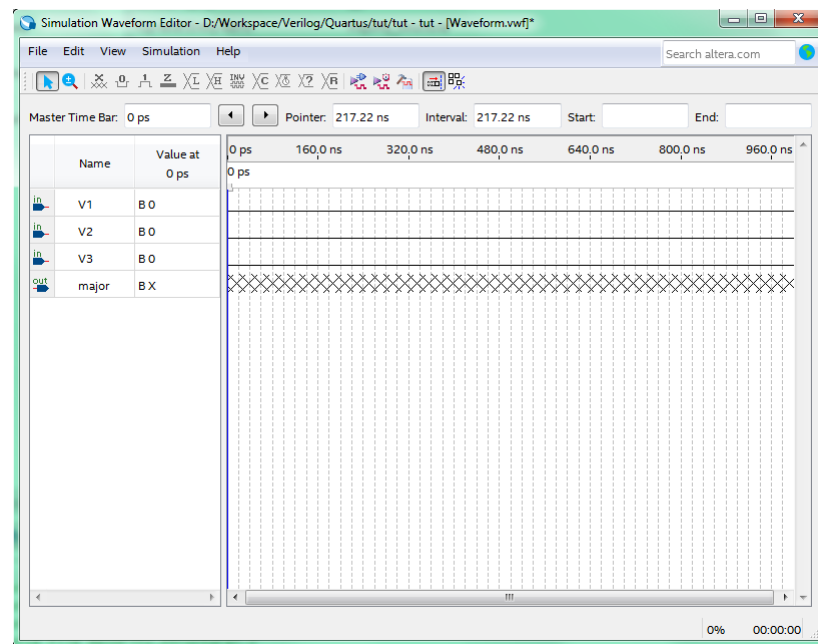
And then, click on "List" button on "Node Finder" window to add all available signals to "Nodes Found" scroll-down box. (Note: only the signals of last recently synthesized module could be found)

When all signals are added to "Nodes Found" scroll-down box, click on "»" button to select all available signals or select a signal and click on ">" button to select a single signal.

Click on "OK" at the end.



4. Click "OK" again to terminate "Insert Node or Bus" window. Now we have a simulation workspace with all necessary signals.

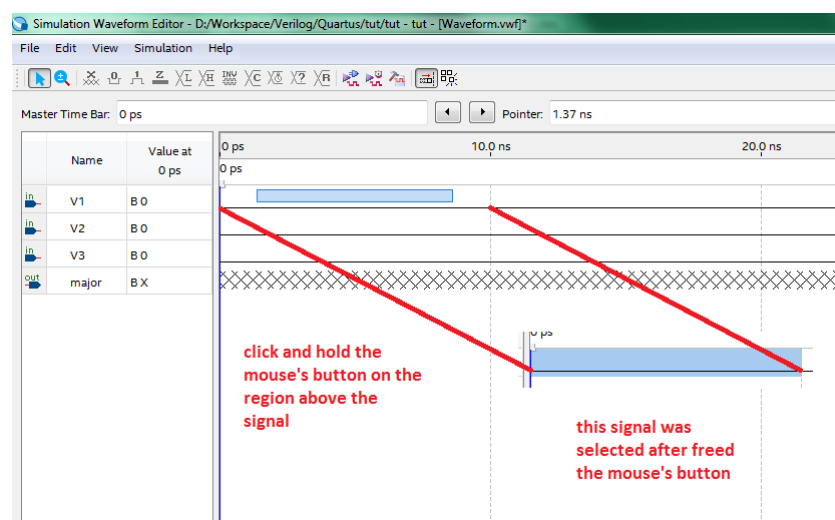


5. Take a look at all tool we have, there are some primary that we need to know first:

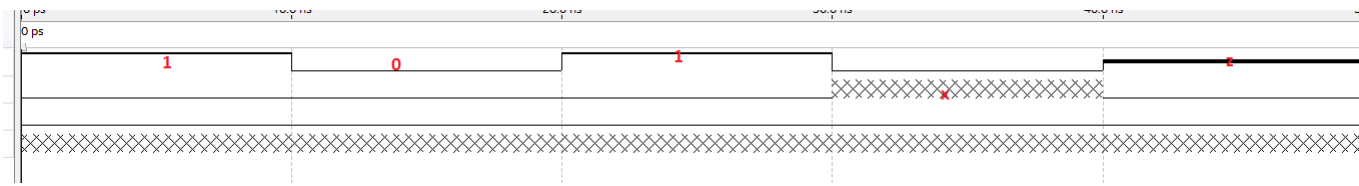
- <1> Selection tool: Select the signal you want to adjust.
- <2> Zoom tool: Zoom your workspace.
- <3,4,5,6>: set X, 0, 1, Z respectively for a signal.
- <7> Run functional simulation: Run simulation.



6. **Set value for a signal:** click-and-hold the left mouse's button on the region above the signal that you want to set, and then leave the button to select it.

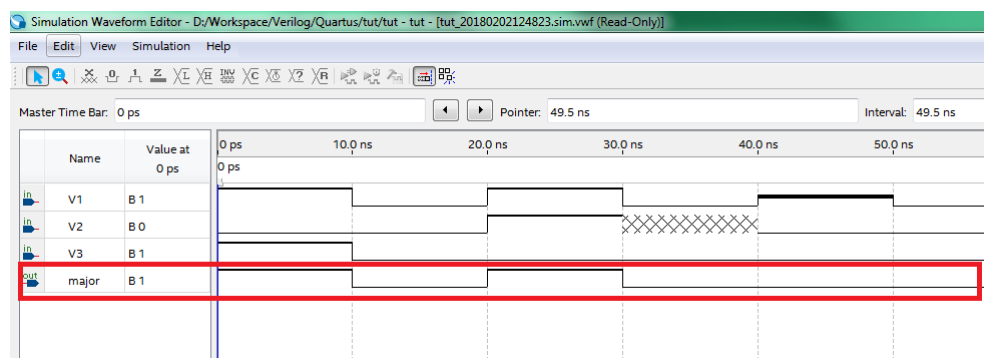


7. Set "X" , "1", "0", or "Z" for the selected signal by using <3, 4, 5, 6> buttons.
8. Do the same task for the rest.



Note: The output signals are always "x" when they're not processed.

9. Click on "Run functional simulation" button to perform the simulation. (make sure the waveform file was saved before executing)
10. And this is the result.



4 Exercises

4.1 Exercises 1: be familiar block diagram (2 pts)

From module "AND_2" that we've already had, implement module AND_16 (16 inputs - 1 output) by using block diagram and simulate it. Paste the block diagram design's image and waveform's image in the white boxes below.

attach the block
diagram's image
here

attach the sim-
ulation's image
here

4.2 Exercises 2: make your first simple design (3 pts)

In this exercises, we'll implement a multiplexer 2 to 1 by using structural model. Fill the code below, synthesize your code and paste the simulation's waveform into white box.

```

module mux_2_to_1(out, select, a, b);

    // output
    output [3:0]out;

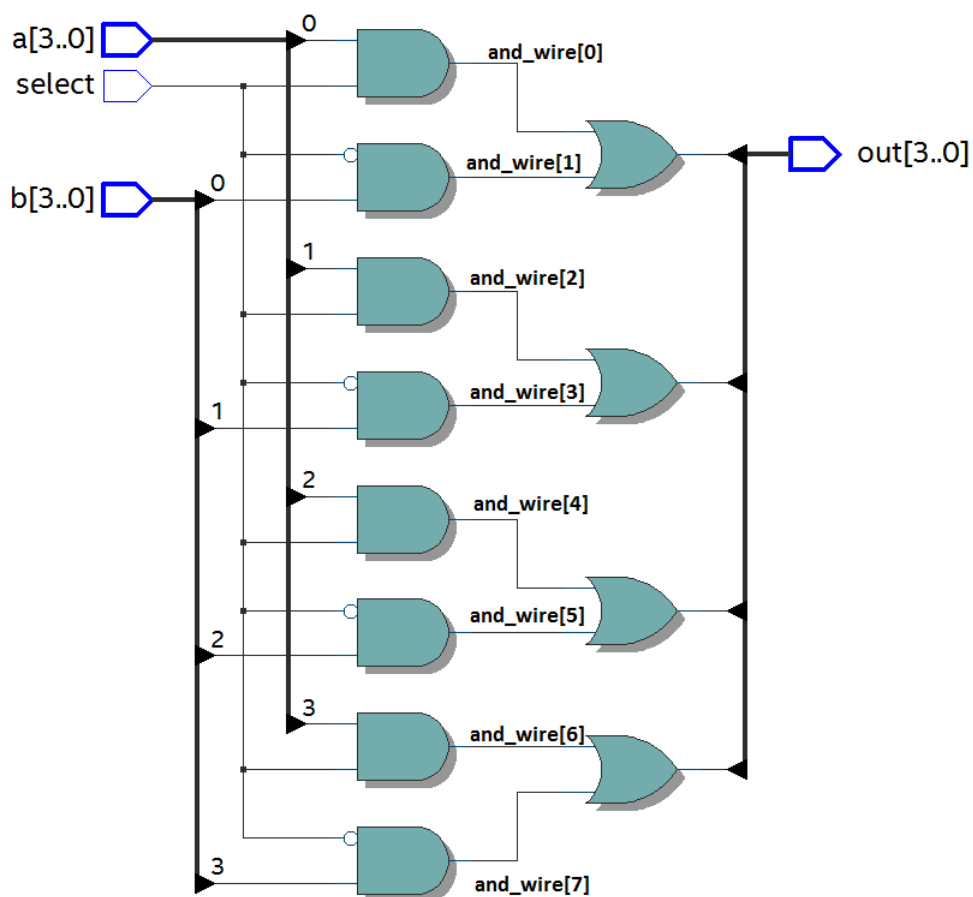
    // input
    input select;
    input [3:0]a;
    input [3:0]b;

    // wire
    wire [7:0]and_wire;
    wire [3:0]not_wire;

    and and_0(and_wire[0], select, a[0]);
    // continue your code here

endmodule

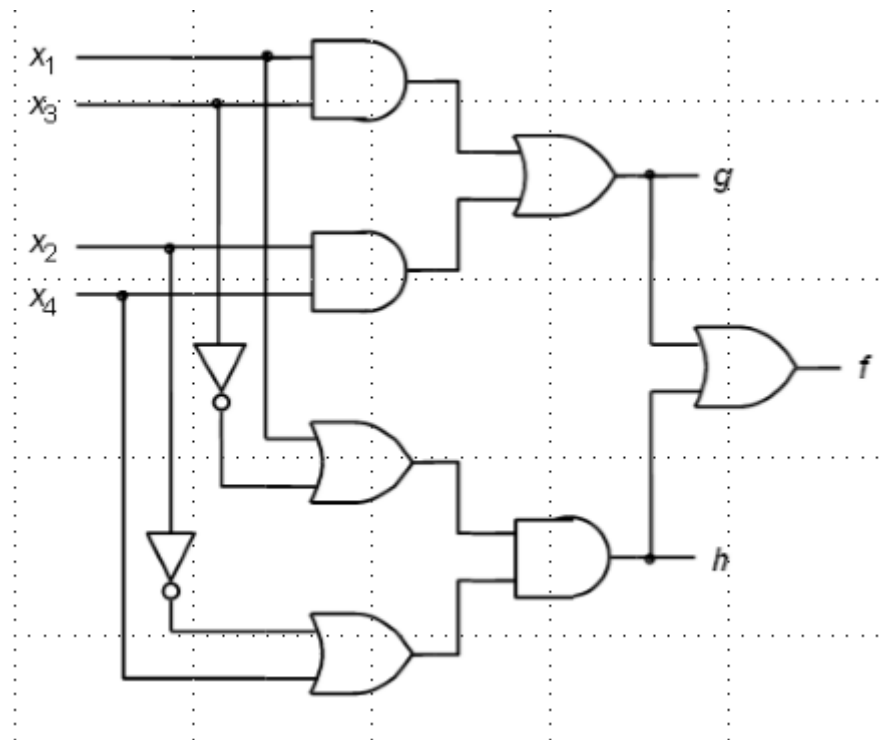
```



attach the wave-
form image here

4.3 Exercises 3: more practice (3 pts)

In this exercises, we'll implement a more complex circuit which is described by the picture below:



Using structural model to implement this circuit.

[illegible]

Simulate your design and put down the waveform's image into the white box below.

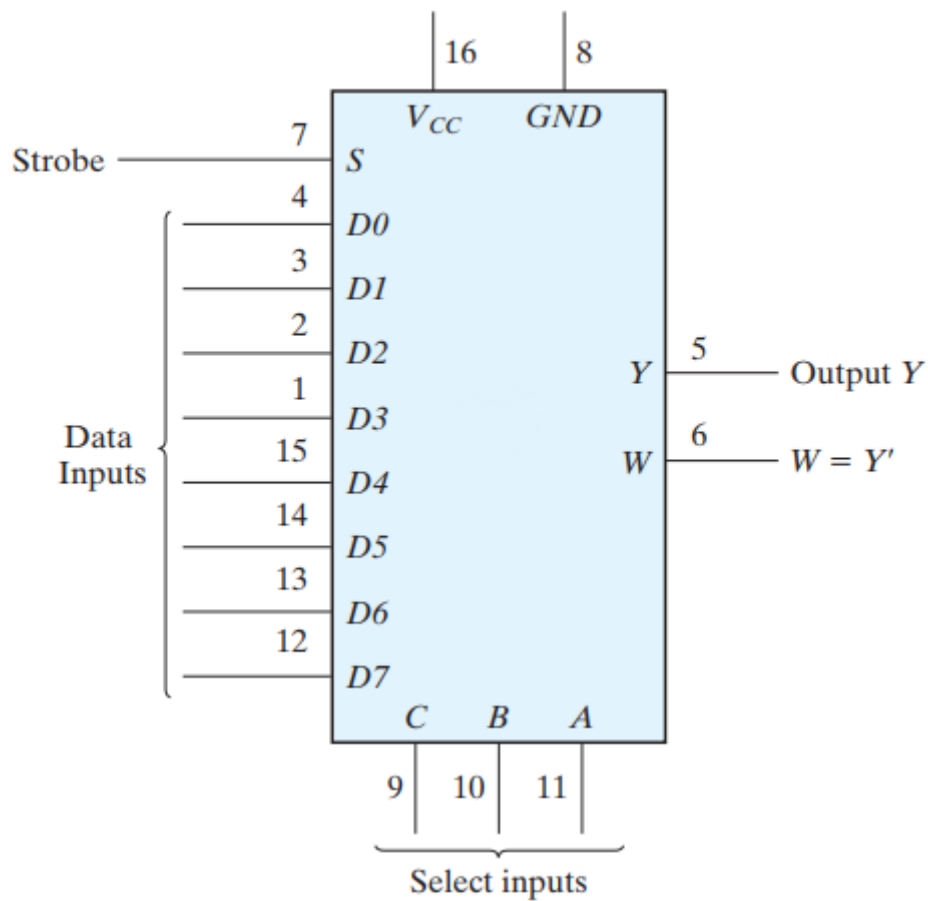


4.4 Exercises 4: be more fluent (bonus pts)

Give the IC and with its specification below, implement the description of it by following the instructions below and then write create a simulation to determine whether your circuit works properly.

1. Make and minimize the boolean expression corresponds to the Functions table.
2. Create "IC" model that describes the IC's functions by using Verilog primitives and structural model.
3. Synthesize and generate the netlist viewer.
4. Simulation your design and generate waveform.

Request: Write down your boolean expression and the minimization's steps, put down the netlist viewer and the waveform's image into the white box below.



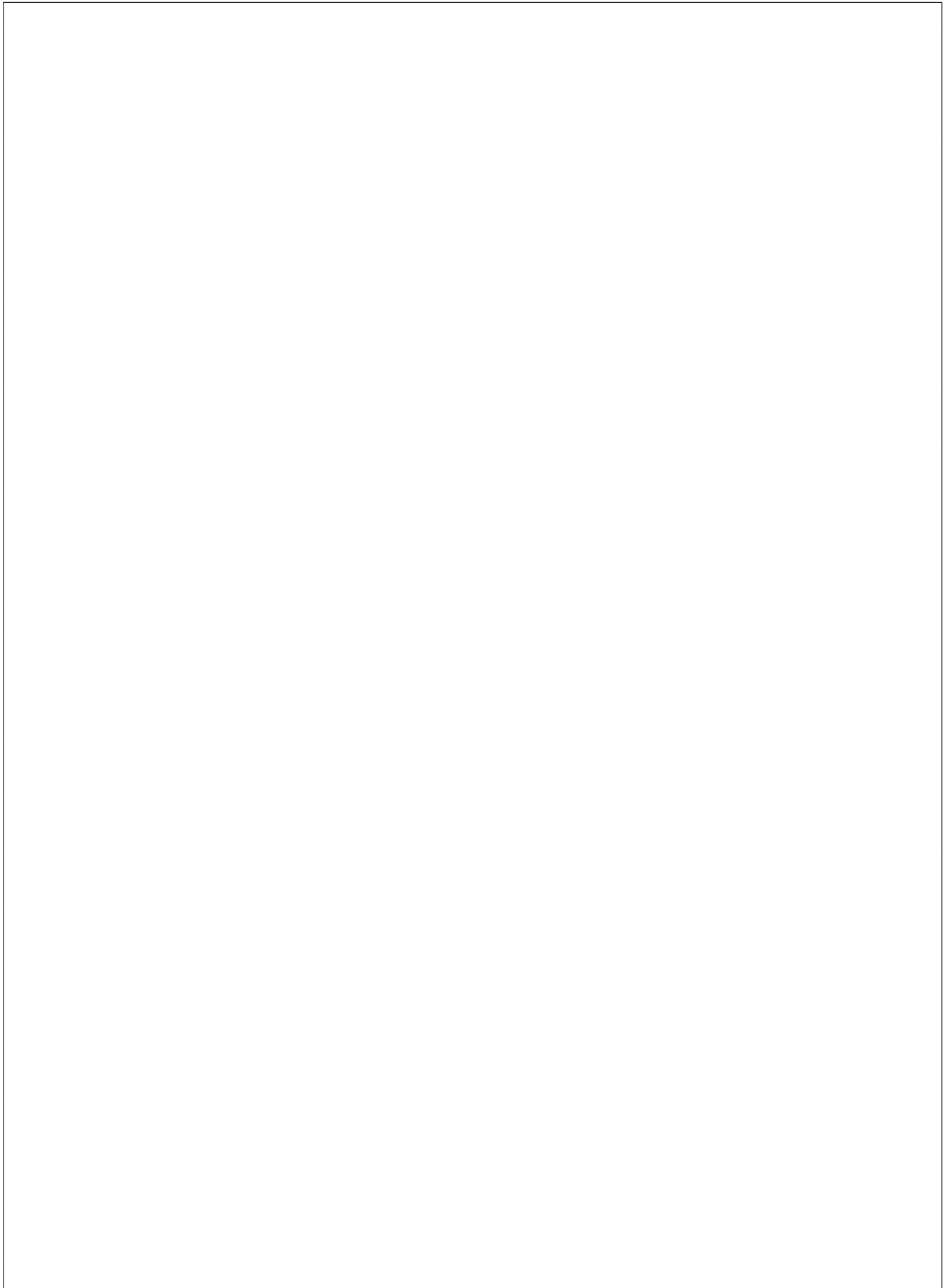
Function table

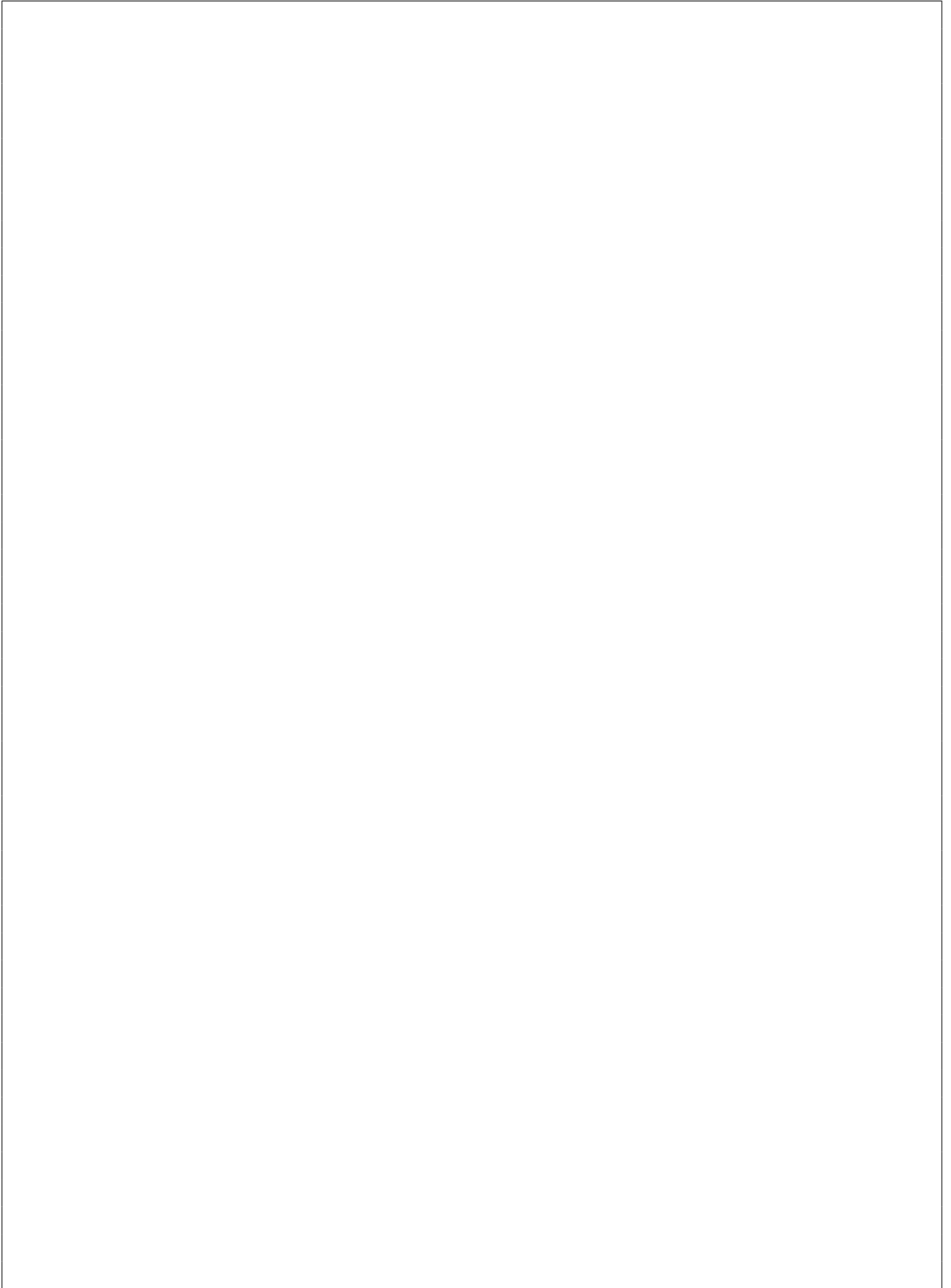
Strobe <i>S</i>	Select <i>C</i> <i>B</i> <i>A</i>			Output <i>Y</i>
1	X	X	X	0
0	0	0	0	<i>D0</i>
0	0	0	1	<i>D1</i>
0	0	1	0	<i>D2</i>
0	0	1	1	<i>D3</i>
0	1	0	0	<i>D4</i>
0	1	0	1	<i>D5</i>
0	1	1	0	<i>D6</i>
0	1	1	1	<i>D7</i>

Note:

$Y' = \text{not}(Y)$

X = don't care





END