*Experiment 3*

*Introduction to Verilog Programming using Quartus II software*

*Prepared by: Eng. Shatha Awawdeh, Eng.Eman Abu_Zaitoun*

# Introduction:

Verilog HDL is a hardware description language used to design electronic systems. Verilog HDL allows designers to design at various levels of abstraction. It is the most widely used HDL with a user community of more than 50,000 active designers.

This tutorial shows how the Quartus II software can be used to design and implement a circuit specified by using the Verilog hardware description language.

# Objectives:

- Creating a project using Quartus II software.
- Design entry using Verilog code.
- Assigning the circuit inputs and outputs to specific pins on the FPGA.
- Simulating the designed circuit.
- Programming and configuring the FPGA device.

## ➢ What is Verilog?

Verilog is one of the two major Hardware Description Languages (HDL) used by hardware designers in industry and academia. VHDL is the other one.

The Verilog language describes a digital system as a set of modules. Each of these modules has input(s) and output(s). Usually we place one module per file but that is not a requirement.

***Note***: Verilog is **case sensitive**

***The structure of a module is the following:***

```
module <module name> (<port list>);
<declares>
<module items>
endmodule
```

- The **<module name>** is an identifier that uniquely names the module.
- The **<port list>** is a list of input and output ports.
- The **<declares>** section specifies data objects as inputs, outputs, or wires.
- The **<module items>** may be assignments or instances of modules.

Modules can represent pieces of hardware ranging from simple gates to complete systems. Modules can either be specified **behaviorally or structurally** (or a combination of the two).

- A **behavioral specification** defines the behavior of a digital system (module) using traditional programming language constructs, e. g., **if**s, **while**s, **assignment statements**.

**Example:**

```
// Behavioral Model of a Nand gate
module NAND(in1,in2, out);
input in1, in2;
output out;
// continuous assign statement
assign out = ~(in1 & in2);
endmodule
```

**In the above example:**
**<module name>:** `NAND`
**<port list>:** `in1,in2,out`
**<Declares>:**
```
input in1, in2;
output out;
```
**<module items>:** `assign out = ~(in1 & in2);`

**Note:** The continuous assignment **assign** continuously watches for changes to variables in its right hand side and whenever that happened the right hand side is re-evaluated and the result immediately propagated to the left hand side (**out**).

- **Structural specification** expresses the behavior of a digital system (module) as a hierarchical interconnection of sub modules.

Here is a structural specification of a module **AND** obtained by connecting the output of one **NAND** to both inputs of another one.

```
module AND(in1, in2, out);
// Structural model of AND gate from two NANDS
input in1, in2;
output out;
wire w1;
// two instantiations of the module NAND
NAND NAND1 (in1, in2, w1);
NAND NAND2 (w1, w1, out);
endmodule
```

This module has two instances of the **NAND** module called **NAND1** and **NAND2** connected together by an internal wire **w1**.
The general form to invoke an instance of a module is:
<module name> <instance name> (<port list>);

## ➢ Quartus II Introduction Using Verilog Design:

The following example makes use of the Verilog design entry method, in which the user specifies the desired circuit in the Verilog hardware description language.

### 1-Getting Started:

Follow the steps in the previous experiment to create new project, and name it Xor1.

### 2-Using the Quartus II Text Editor:

1-Select **File > New** to get the window in Figure 1, then choose **Verilog HDL File** and click **OK**. This opens the Text Editor window in Figure 2.

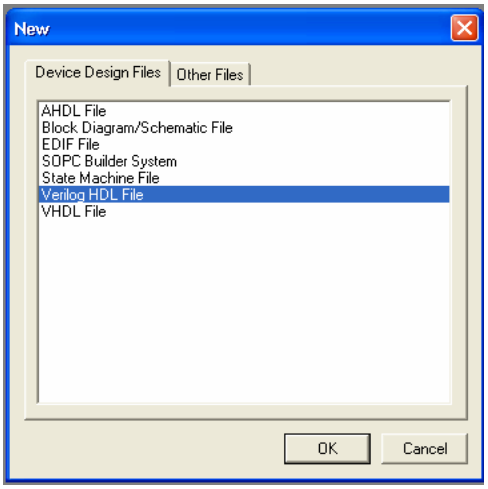


Figure (1)

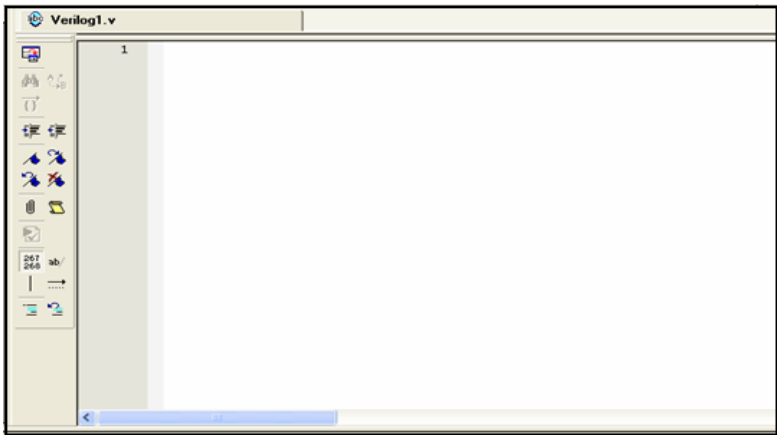


Figure (2)

2- Specify the name for the file that will be created. Select **File > Save As** to open the pop-up box depicted in Figure 3.

In the box labeled **Save as type choose Verilog HDL File**. In the box labeled **File name type Xor1**. Put a checkmark in the box **Add file to current project**. Click **Save**, which puts the file into the directory Exp2 and leads to the Text Editor window shown in Figure (4).

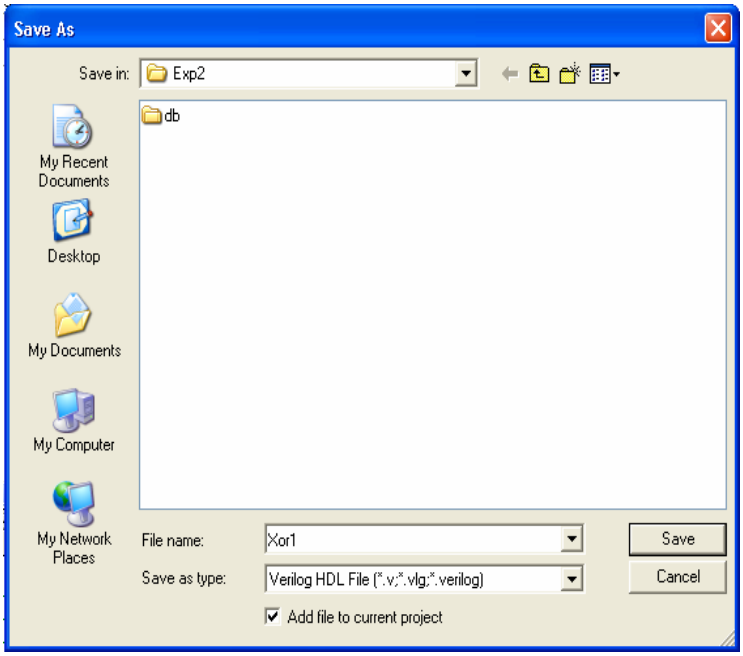


Figure (3)

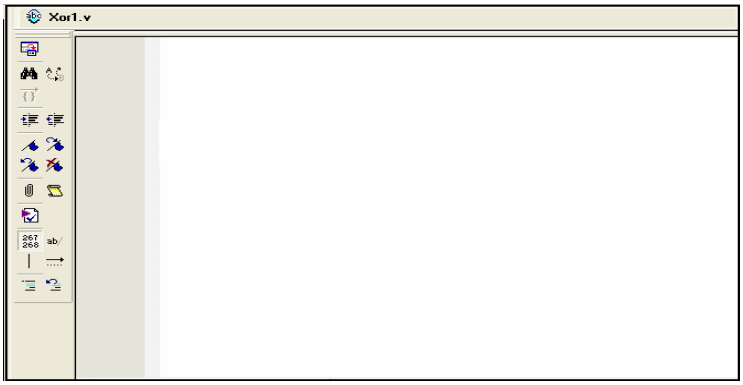


Figure (4)

3- Enter the Verilog code as shown in Figure 5. Then save the file by choosing **File > Save**, or by typing the shortcut Ctrl-s.

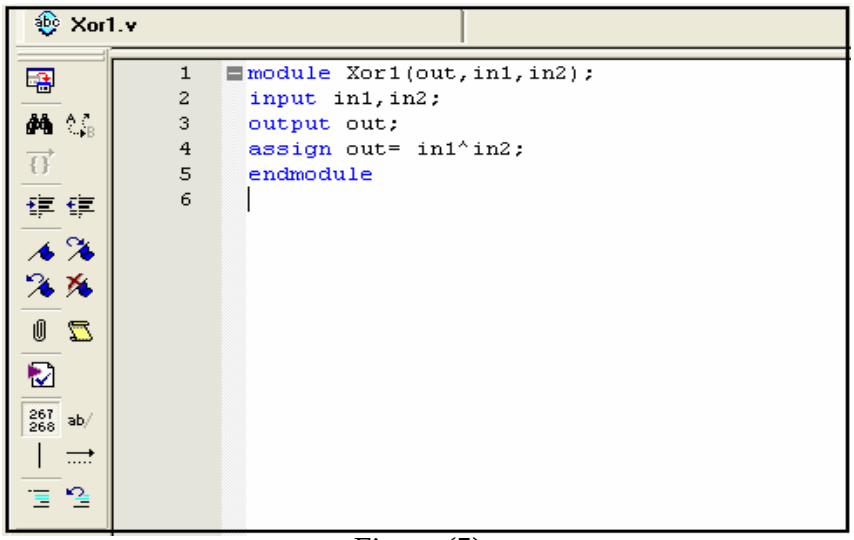


Figure (5)

**Using Verilog Templates**

The syntax of Verilog code is sometimes difficult for a designer to remember. To help with this issue, the Text Editor provides a collection of Verilog templates. The templates provide examples of various types of Verilog statements, such as a module declaration, an always block, and assignment statements. It is worthwhile to browse through the templates by selecting **Edit > Insert Template > Verilog HDL** to become familiar with this resource.

## 3-Compiling the Designed Circuit:

Refer to Experiment 2.

## Errors

Quartus II software displays messages produced during compilation in the Messages window. If the Verilog design file is correct, one of the messages will state that the compilation was successful and that there are no errors.

If the Compiler does not report zero errors, then there is at least one mistake in the Verilog code. In this case a message corresponding to each error found will be displayed in the Messages window. Double-clicking on an error message will highlight the offending statement in the Verilog code in the Text Editor window. Similarly, the Compiler may display some warning messages. Their details can be explored in the same way as in the case of error messages. The user can obtain more information about a specific error or warning message by selecting the message and pressing the F1 function key.

To see the effect of an error, open the file Xor1.v. Remove the semicolon in the assign statement, illustrating a typographical error that is easily made. Compile the erroneous design file by clicking on the icon. A pop-up box will ask if the changes made to the Xor1.v file should be saved; click Yes. After trying to compile the circuit, Quartus II software will display a pop-up box indicating that the compilation was not successful. Acknowledge it by clicking OK. The compilation report summary, now confirms the failed result. Expand the Analysis & Synthesis part of the report and then select Messages to have the messages displayed as shown in Figure 6.

.

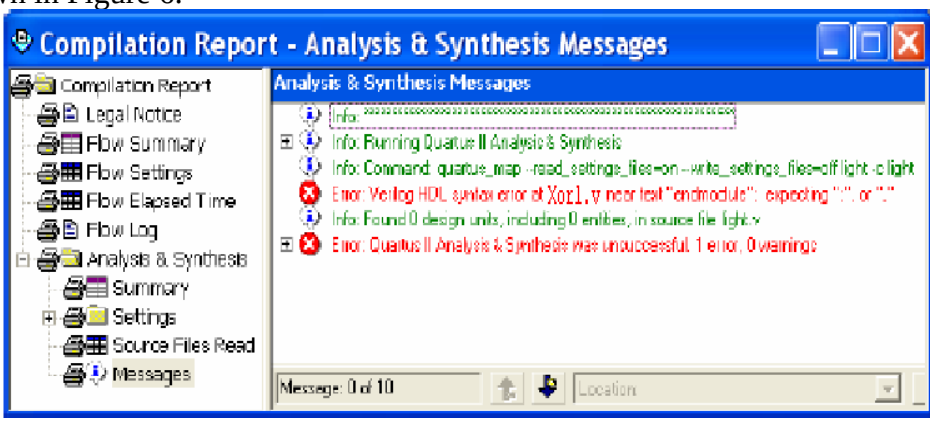


Figure 6

Double-click on the first error message. Quartus II software responds by opening the Xor1.v file and highlighting the statement which is affected by the error. Correct the error and recompile.

**4-Pin Assignment:**
Refer to Experiment 2.

**5- Simulating the Designed Circuit**
Refer to Experiment 2.

**6 -Programming and Configuring the FPGA Device**
Refer to Experiment 2.

**Note:**
*If you want to implement the Xor module using structural modeling do the following:*

*1. Write the primitive needed modules (AND, OR, INV) in a separate file name it (lib.v)*

```
module ANDGATE(in1, in2, out);
input in1, in2;
output out;
assign out= in1&in2;
endmodule

module ORGATE(in1, in2, out);
input in1, in2;
output out;
assign out= in1|in2;
endmodule

module INVGATE(in1, out);
input in1;
output out;
assign out= ~in1;
endmodule
```

*2. Write Xor module in a separate file, name it (Xor1.v)*

```
module Xor1(in1, in2, out);
input in1, in2;
output out;
wire w1,w2,w3,w4;

INVGATE inv1(in1,w1);              // w1 = ~in1
INVGATE inv2(in2,w2);              // w2 = ~in2

ANDGATE AND1(in1, w2, w3);        // w3 = in1&(~in2)
ANDGATE AND2(in2, w1, w4);         // w4 = in2&(~in1)

ORGATE or1(w3,w4,out);            // out = w3 | w4

endmodule
```

*Note that the Xor1 module should be the top level module, you can change top level module by* ***choosing project>set as top level.***

***If you need the file (lib.v) in another project you don't have to create it again, just make add file in project creation (step3) in Experiment 2.***