Simple RTL (VHDL) project

Introduction

In this tutorial we will create a simple VHDL project using the text editor of Xilinx Vivado 2016.1.

The design will have 4 1-bit inputs and 1 1-bit output. The 2 first inputs, which we will name A and B, will be connected to an AND gate and the two last inputs, C and D, will be connected to an OR gate. The two gates outputs will enter in another AND gate. The project is schematized in the Figure 1.

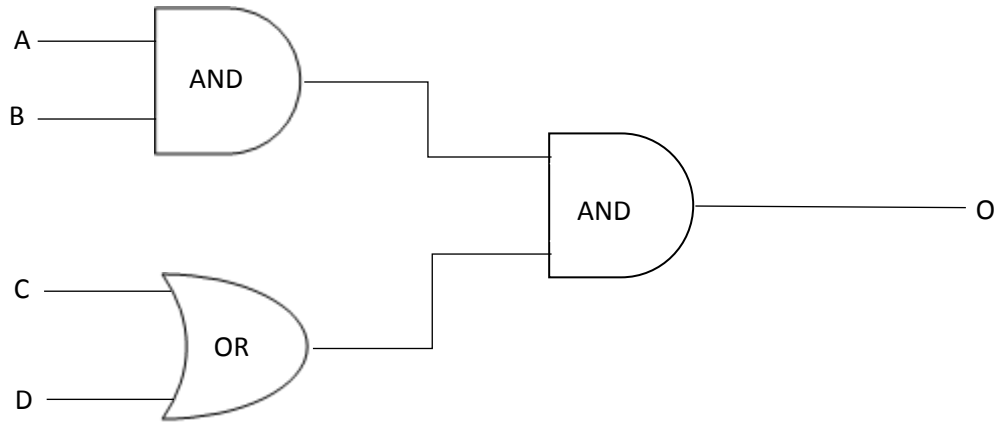

Figure 1 - Project Schematic

Create a VHDL project

Start Xilinx Vivado 2016.1 in a GUI mode using the command line as explained in the post 1 or using the desktop icon. On the welcome page click on "*Create New Project*". Enter a project name and location.

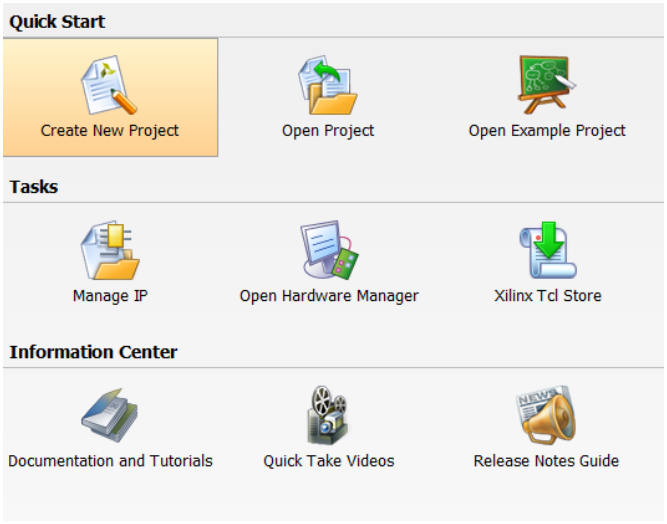

Figure 2 - Create a new project

In the "*Project Type*" window, select "*RTL project*" and check the "*Do not specify sources at this time*".

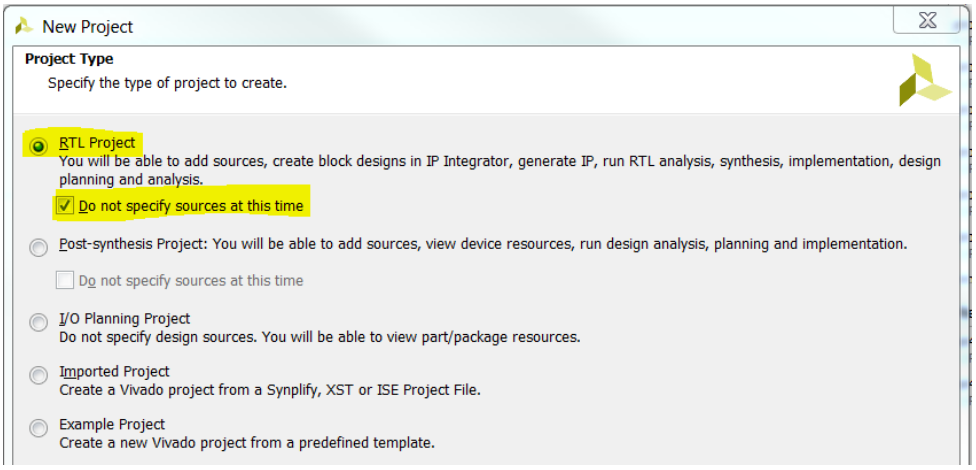

Figure 3 - RTL Project Selection

Select the Kintex-7 xc7k70tfbg676-1 as default part.

Create the VHDL file

In the Flow Navigator, click on "*Add Sources*" (alternatively you can click on "*File* > *Add Sources*").

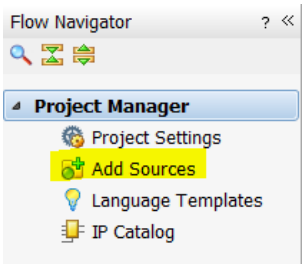

Figure 4 - Add Sources

In the first page of the "*Add Sources*" window, select "*Add or create design sources*".

Figure 5 - Add or create design sources

In the "*Add or Create Design Sources*" page, click on the + button on the left and click on "*Create File*" or click on the "*Create File*" button on the bottom.

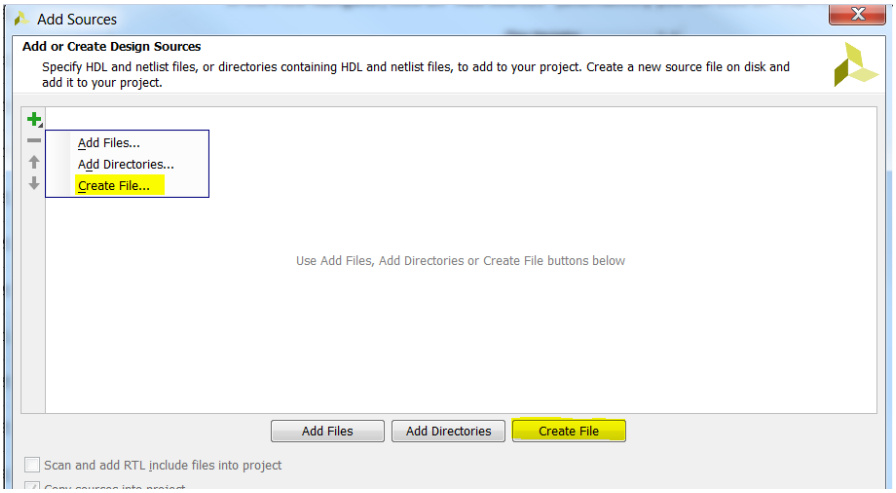

Figure 6 - Create Design Source File

In the "*Create Source File*" window, select "*VHDL*" for the "*File type*" field and add a file name. Click "*OK*" and click "*Finish*" on the "*Add Sources*" window.

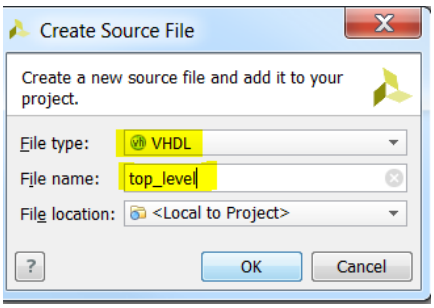

Figure 7 - Create a new VHDL file

A "*Define Module*" window will appear. It allows you to define the ports of your new module in order to have a pre-filed VHDL file. Fill it as below.

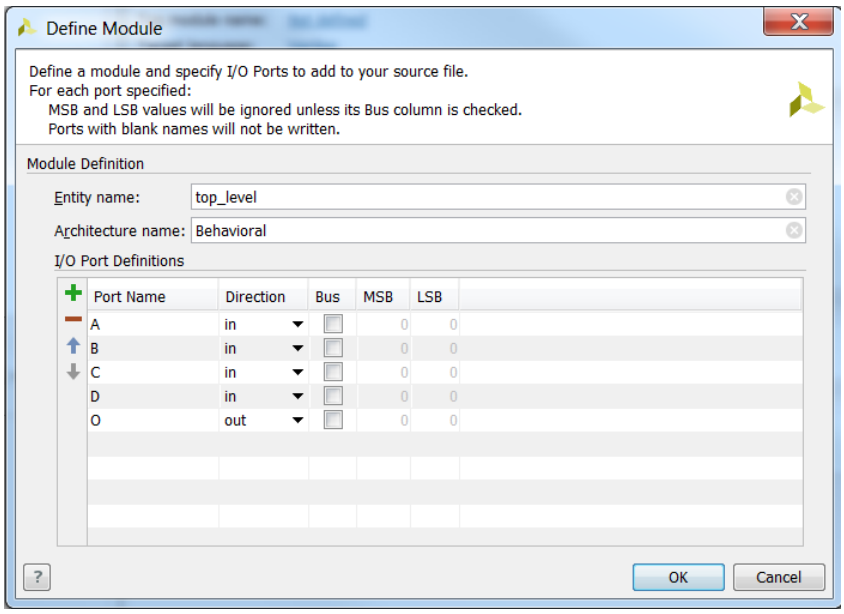

Figure 8 - Define Module window

Modify the VHDL file

In the Data Windows Area, in the "*Sources*" window, we can see the VHDL file in the "*Design Sources*". We can double click on this file to open the text editor.

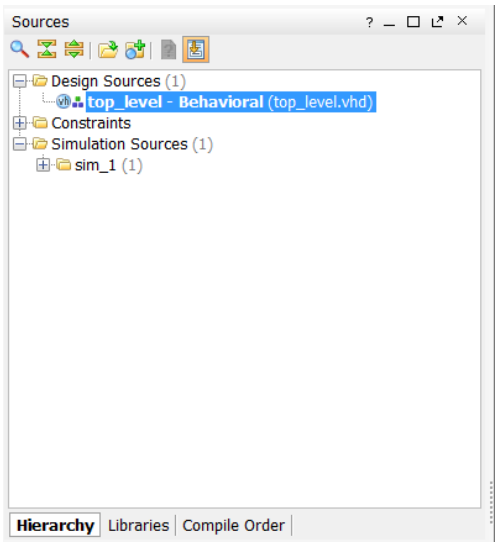

Figure 9 - Sources window

We can see that Vivado has already created the structure of the VHDL file.

```
34 entity top_level is
35     Port ( A : in STD_LOGIC;
36            B : in STD_LOGIC;
37            C : in STD_LOGIC;
38            D : in STD_LOGIC;
39            O : out STD_LOGIC);
40 end top_level;
41
42 architecture Behavioral of top_level is
43
44 begin
45
46
47 end Behavioral;
```

Figure 10 - VHDL structure created by Vivado

In the text editor we can code our block.

```
42 architecture Behavioral of top_level is
43
44 signal AND_1_out_sig  :  std_logic;
45 signal OR_out_sig    :  std_logic;
46
47 begin
48
49 AND_1_out_sig   <= A AND B;
50 OR_out_sig      <= C OR D;
51
52 O <= AND_1_out_sig AND OR_out_sig;
53
54
55 end Behavioral;
```

Figure 11 - VHDL code of the block

When you save your file, if there are syntax errors, the file will be displays under "*Syntax Error Files*" in the "*Sources*" window and the errors will be displayed in the messages console.

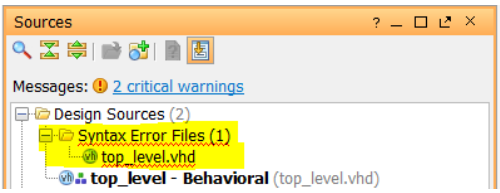

Figure 12 - Syntax Error Files

View the Schematic of the RTL (VHDL) block

In the "*Flow Navigator*", in "*RTL Analysis*", if we expend "*Elaborated Design*" we can click on "*Schematic*".

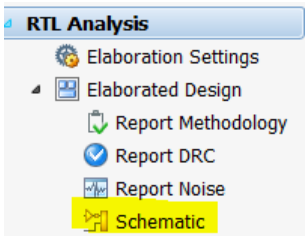

Figure 13 - Open the Elaborated Design Schematic

This will open the schematic of our RTL bloc. From this schematic we can check that we have correctly coded our bloc.

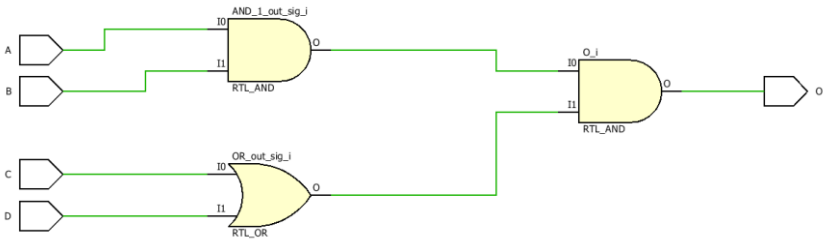

Figure 14 - Elaborated Design Schematic

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity top_level is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : in STD_LOGIC;
           D : in STD_LOGIC;
           O : out STD_LOGIC);
end top_level;

architecture Behavioral of top_level is

signal AND_1_out_sig    : std_logic;
signal OR_out_sig       : std_logic;

begin

AND_1_out_sig   <= A AND B;
OR_out_sig      <= C OR D;

O   <= AND_1_out_sig AND OR_out_sig;

end Behavioral;
```