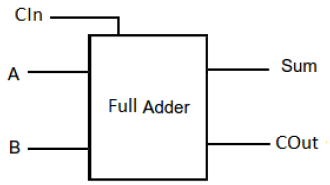
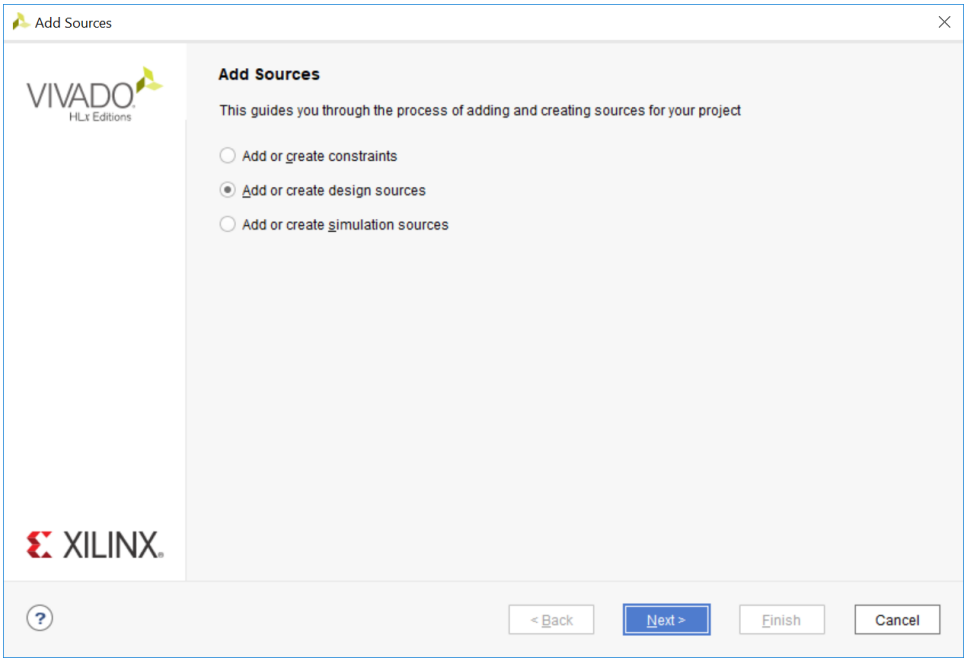
1-Lab preparation

Practice for designing a 1-bit Full Adder

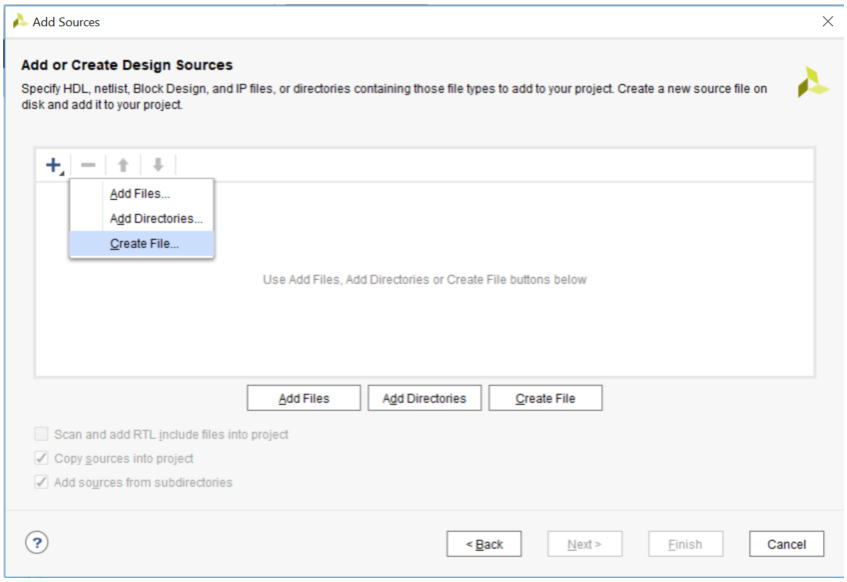
See the block diagram below.



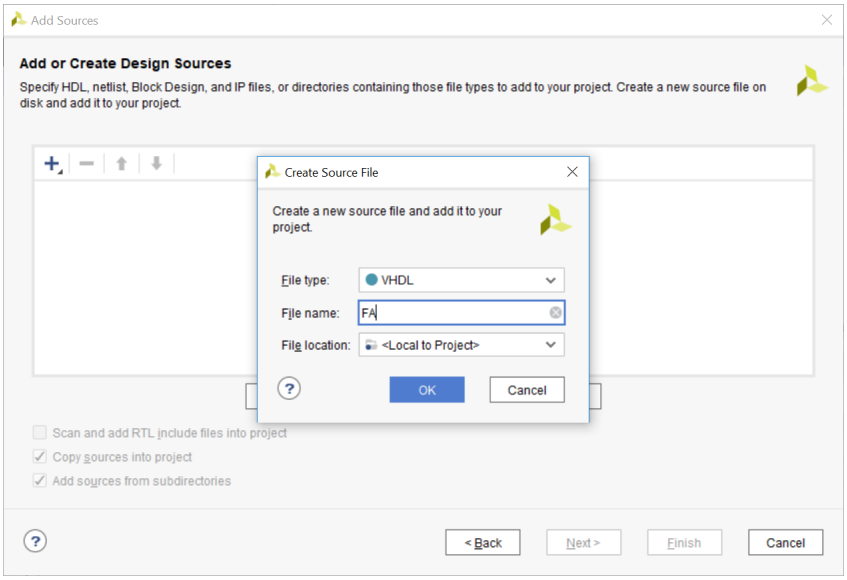
From the Flow Navigator window, choose Add Sources. Select "Add or create design sources", click Next.



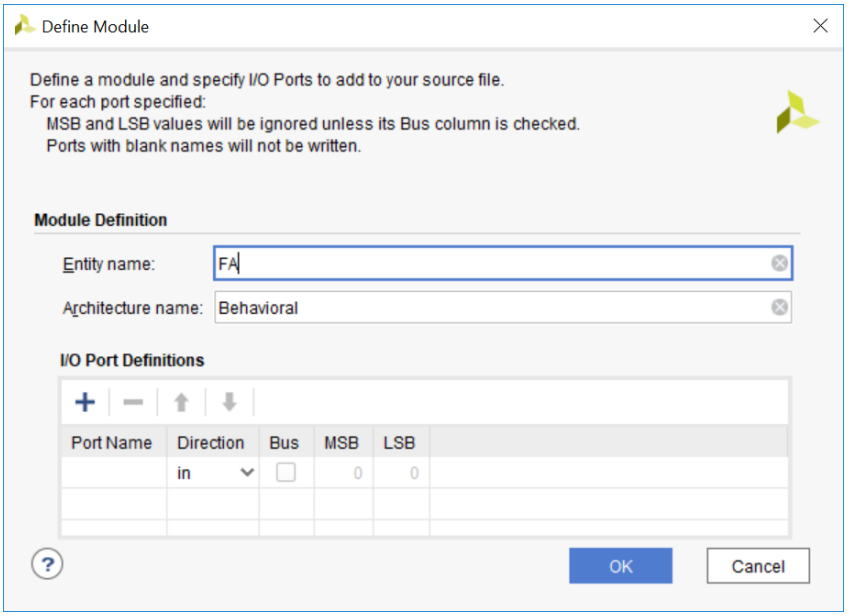
We want to create a new file so, click "Create File".



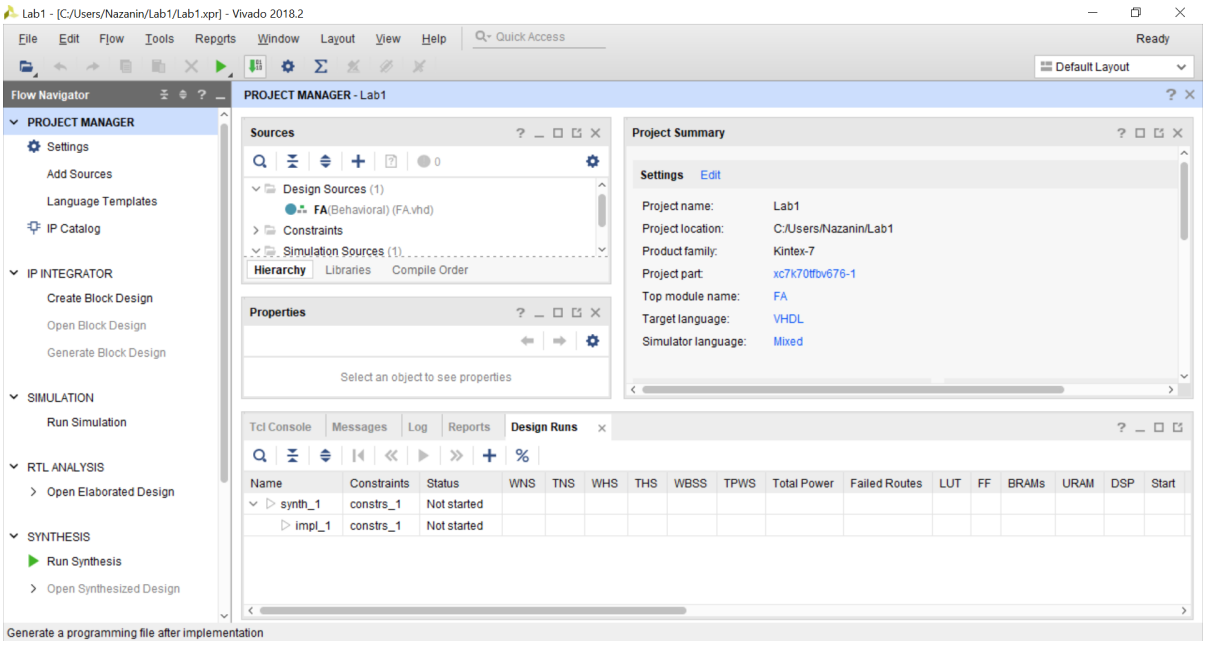
Define a name for your design and make sure the type is VHDL. Click Ok and Finish.



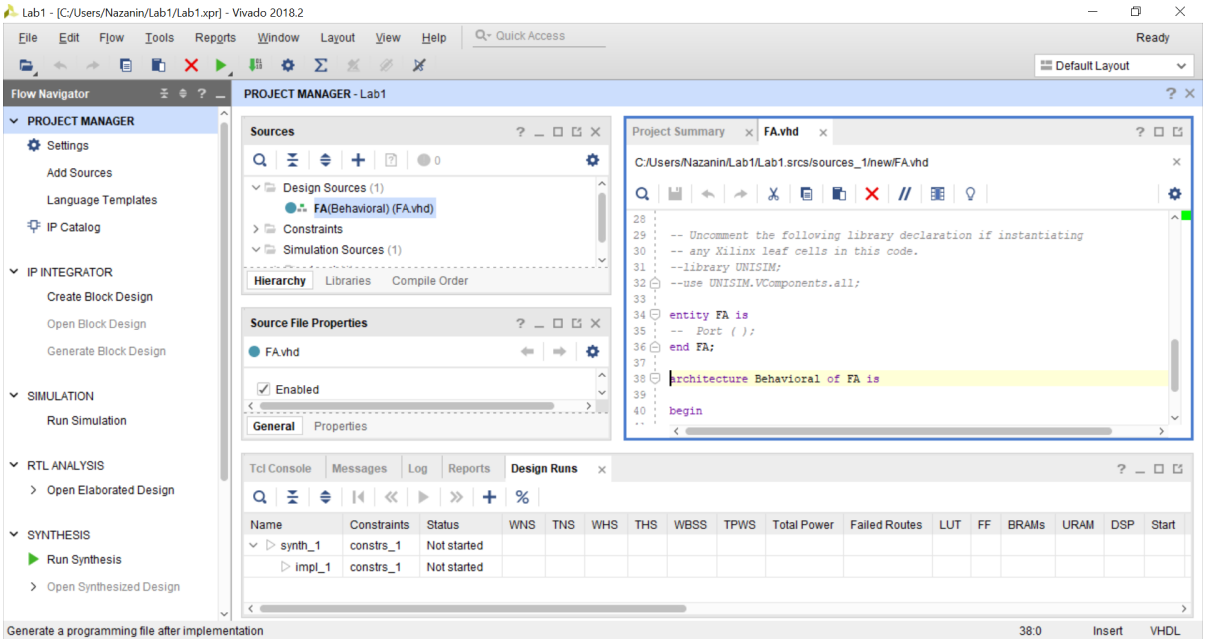
The next window asks you to define I/O ports for your module. We are going to define them later in the code so click Ok.



Now you see your new source file (FA) under Design Sources in the Sources window.



Double click on FA to open it.

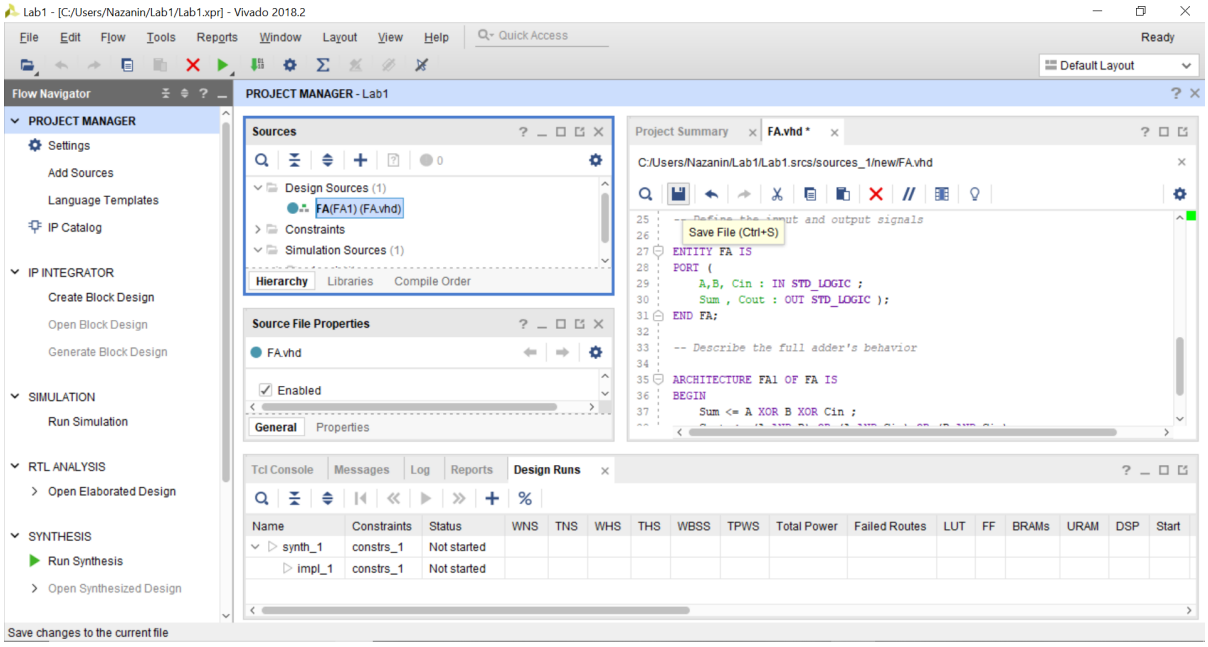


Below you see the VHDL code for 1-bit Full Adder. Copy and paste this code to the FA.vhd window.

Code 1: 1-bit Full Adder.

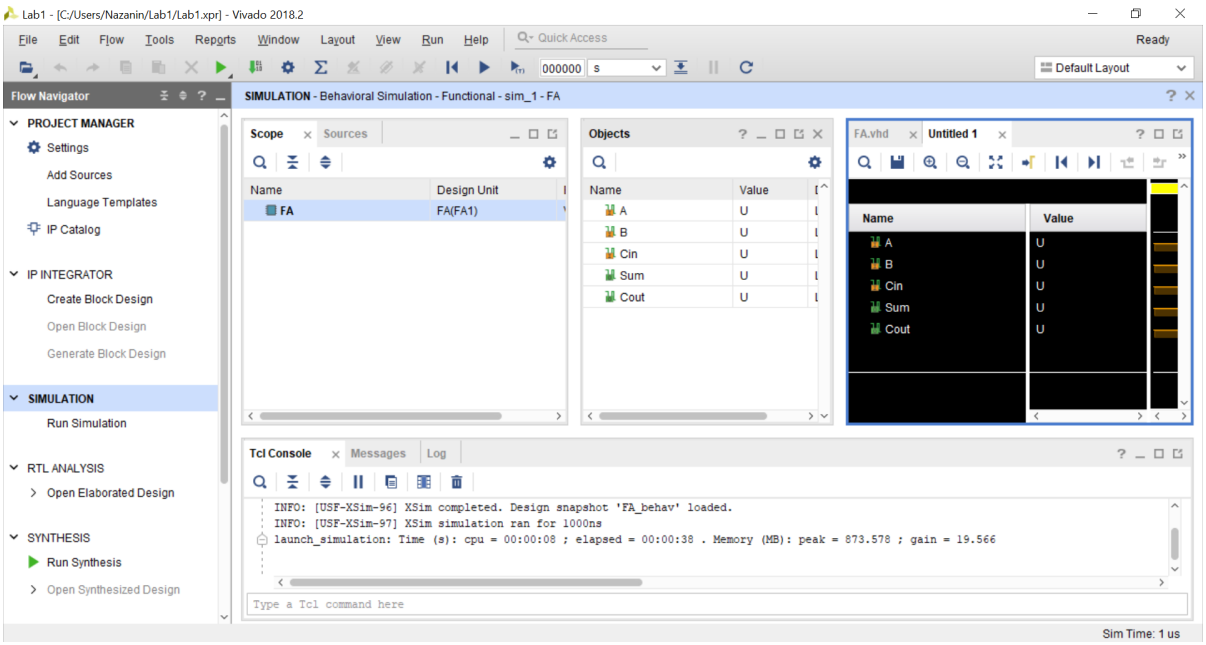
```
LIBRARY IEEE ;
USE IEEE.std_logic_1164.ALL ;

-- Define the input and output signals

ENTITY FA IS
PORT (
      A,B, Cin : IN STD_LOGIC ;
      Sum, Cout : OUT STD_LOGIC );
END FA;

-- Describe the full adder's behavior

ARCHITECTURE FA1 OF FA IS
BEGIN
      Sum <= A XOR B XOR Cin ;
      Cout <= (A AND B) OR (A AND Cin) OR (B AND Cin);
END FA1;
```

If you have a syntax error in this part, the line which contains error will be underlined by red color. Make sure to correct all of them. Save your code.



2- Simulation

To simulate your design click on the "Run Simulation" in the Flow Navigator window. Choose "Run Behavioral Simulation". After a successful simulation, the simulation window opens.



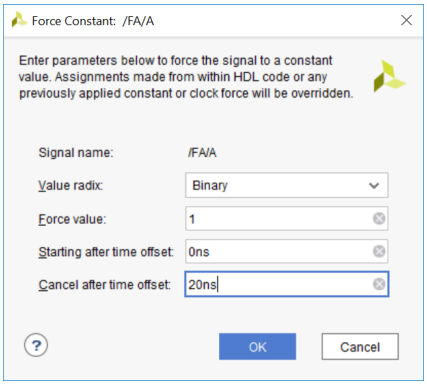
You can check your design function through this window. And if it is necessary, go back to your design code and change it.
To test we need to provide some values for inputs. Here we have 3 inputs: A, B, and Cin. In the table below you see two test scenarios.

Inputs			Outputs	
A	B	Cin	Sum	Cout
1	0	1	0	1
1	1	1	1	1

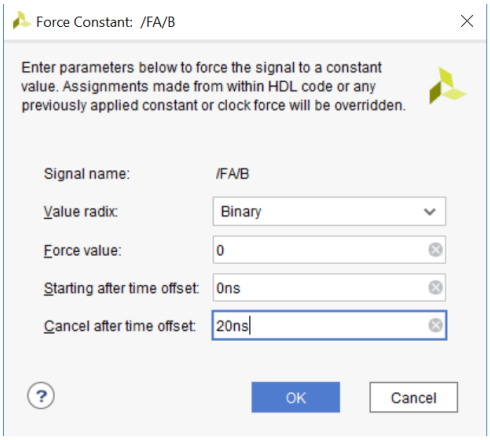
In the Objects window right click on `A` and select "Force Constant". The Force Constant window has several fields:

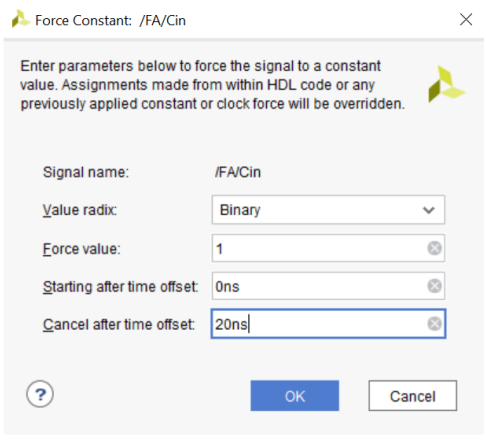
- Signal name: Displays the default signal name, that is, the full path name of the selected item.
- Value radix: Displays the current radix setting of the selected signal. You can choose one of the supported radix types: Binary, Hexadecimal, Unsigned Decimal, Signed Decimal, Octal, and ASCII. The GUI then disallows entry of the values based on the Radix setting. For example: if you choose Binary, no numerical values other than 0 and 1 are allowed.
- Force value: Specifies a force constant value using the defined radix value.
- Starting at time offset: Starts after the specified time. The default starting time is 0. Time can be a string, such as 10 or 10 ns. When you enter a number without a unit, the Vivado simulator uses the default.
- Cancel after time offset: Cancels after the specified time. Time can be a string such as 10 or 10 ns. If you enter a number without a unit, the default simulation time unit is used.

`A` is a 1-bit input so put Binary in value radix. In the first test, we want A to be 1. So, set the force value to 1 and set the starting and cancellation points to 0ns and 20ns respectively. Click ok.

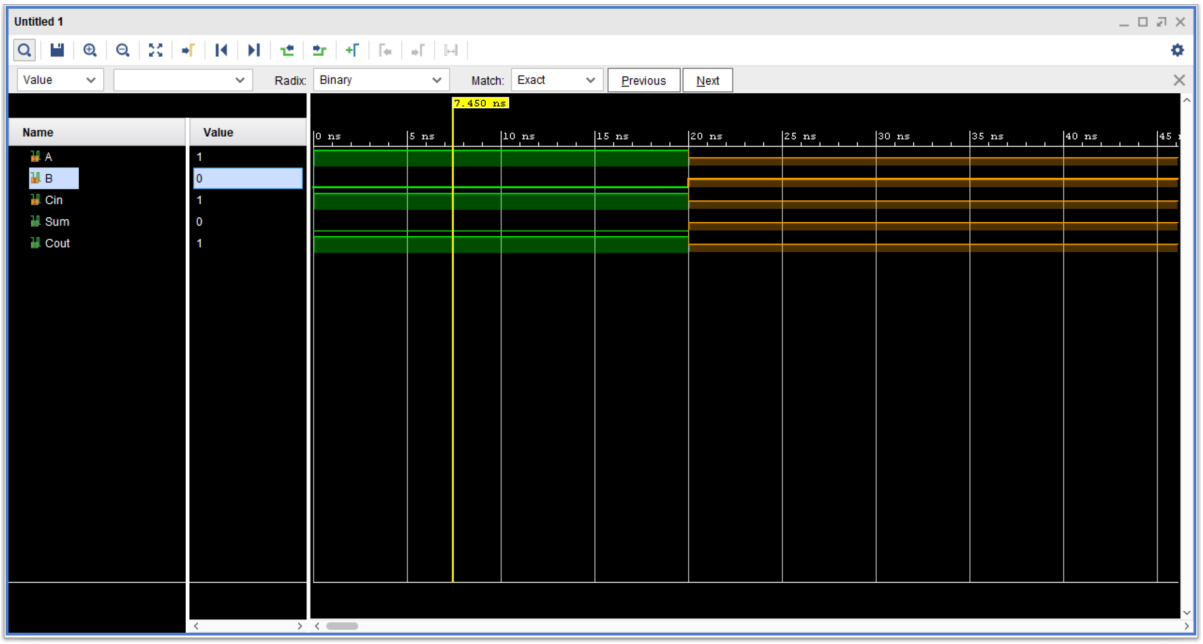


Do the same thing for `B` and `Cin`.

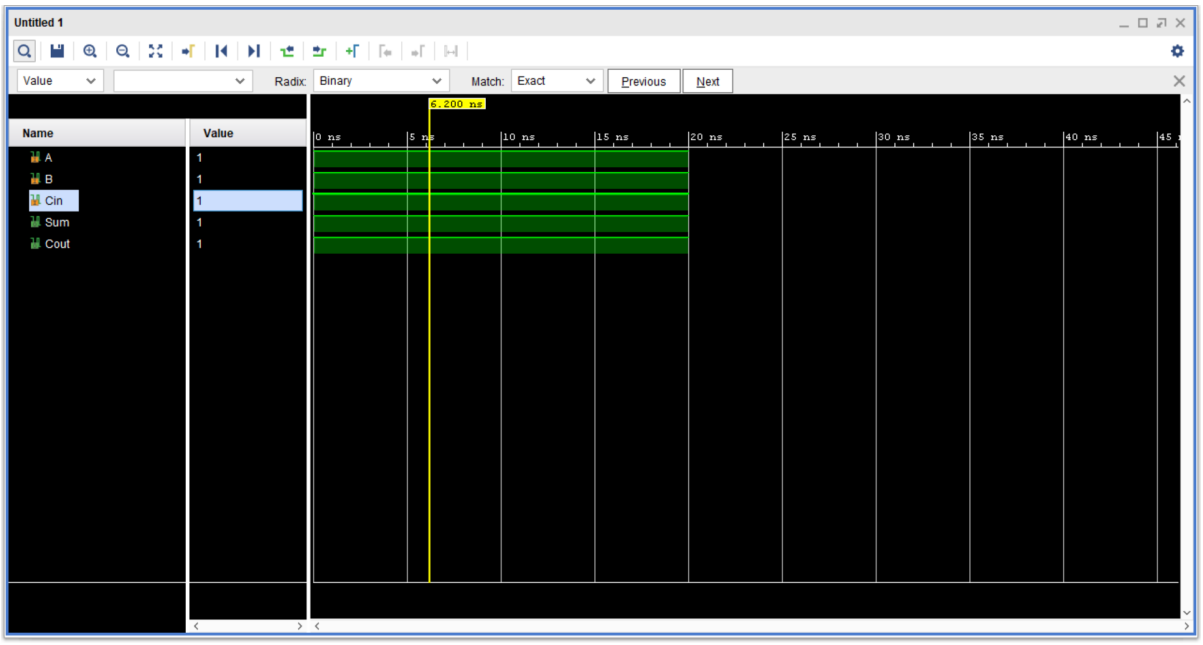




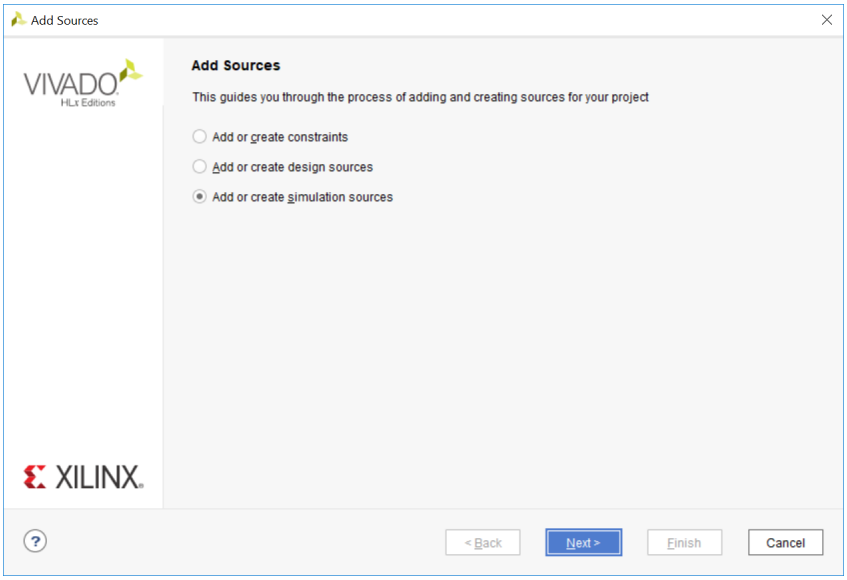
Now form the top menu select Run All. You can see the result here. "Sum" is 0 and "Cout" is 1 as expected.



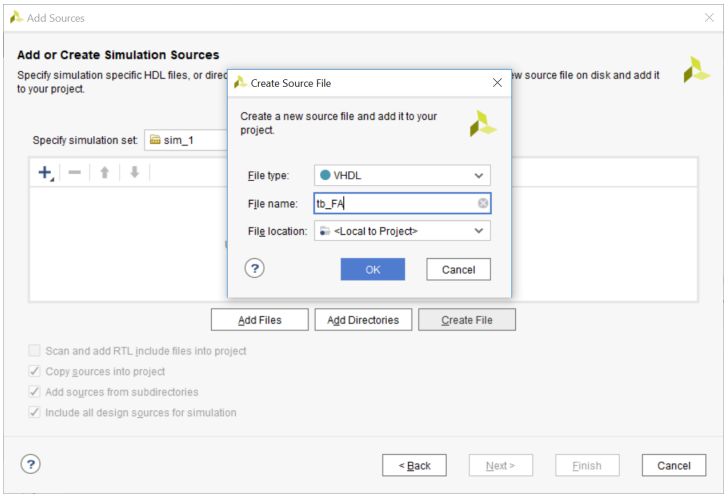
Now try the second test by setting A, B, and Cin to 1, 1, and 1. See the result is 1 for both Sum and Cout.



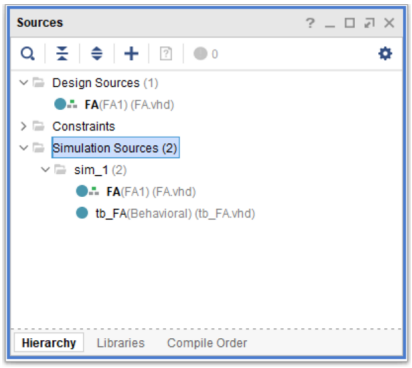
Another method of testing your design is by writing a testbench code. To write a testbench first choose "Add Sources" from the Flow Navigator window. This time select "Add or create simulation source".



In the Add source window select "Create File" and define a name for your testbench. Here I use "tb_FA". Click Ok and Finish.



Now in the source window under Simulation sources, you can see your testbench file.

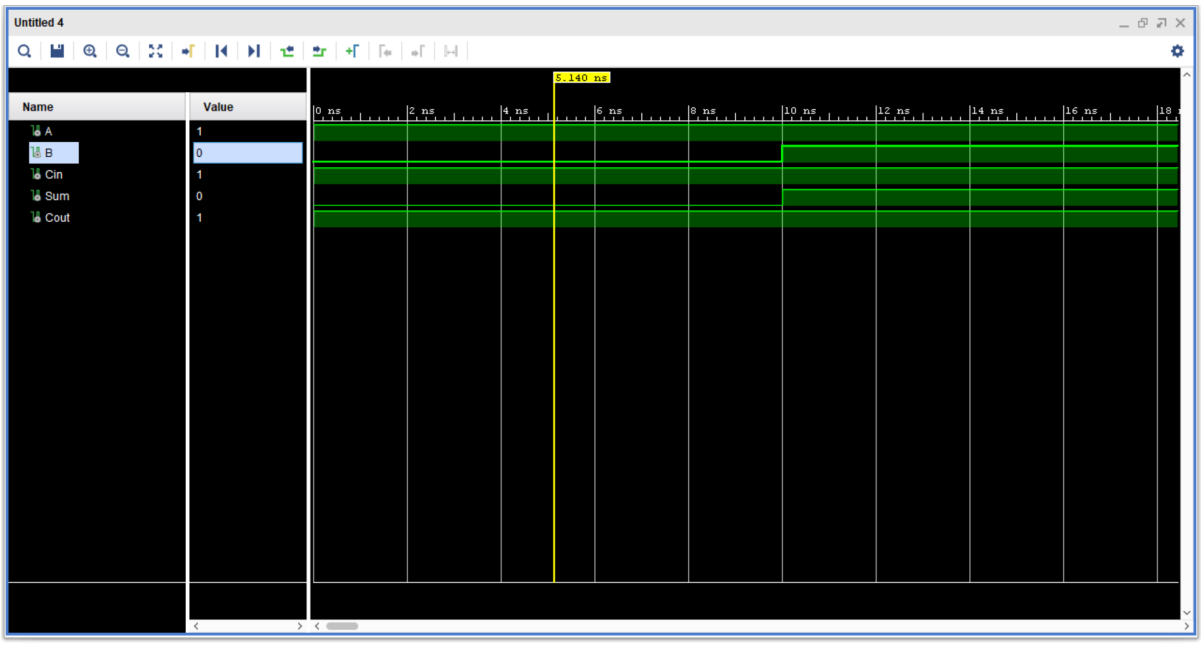


Double click on tb_FA and use the code below.

Code 2: Testbench for 1-bit Full Adder.

```
LIBRARY IEEE ;
USE IEEE . std_logic_1164 . ALL ;

ENTITY tb_FA IS
END tb_FA;

ARCHITECTURE Behavioral OF tb_FA IS
    COMPONENT FA IS -- component declaration
    PORT(
          A,B, Cin : IN STD_LOGIC ;
          Sum , Cout : OUT STD_LOGIC
    );
    END COMPONENT;

    SIGNAL A,B, Cin: STD_LOGIC := '0'; -- signal declarations
    SIGNAL Sum, Cout: STD_LOGIC;

BEGIN
   uut : FA -- component instantiation
   PORT MAP(
      A => A, -- signal mappings
      B => B,
      Cin => Cin,
      Sum => Sum,
      Cout => Cout);

      A <= '1';
      B <= '0', '1' AFTER 10ns;
      Cin <= '1';

END Behavioral;
```

Save your code and use "Run simulation" from Flow Navigator window. In the first 10ns of simulation we test the first scenario and from 10ns to 20ns we test the second one.



3- Lab1 assignment: design of 2-bit adder

Modify Entity and Architecture portion of 1-bit Adder described above. Write a VHDL code for a 2-bit full adder with these inputs: A2A1, B2B1, and Cin. Then the outputs are Sum2Sum1, and Cout. Simulate your code with these inputs:

Inputs			Outputs	
A2 A1	B2 B1	Cin	Sum2 Sum1	Cout
10	01	1	00	1
01	00	1	10	0
11	10	0	01	1

Submit two items:

1- All VHDL files

2-Your report in doc or pdf format with the copy and paste all vhdl codes and screenshot of all tests. In the first line of your report write if your code is working.