
Department of Electrical and Computer Engineering

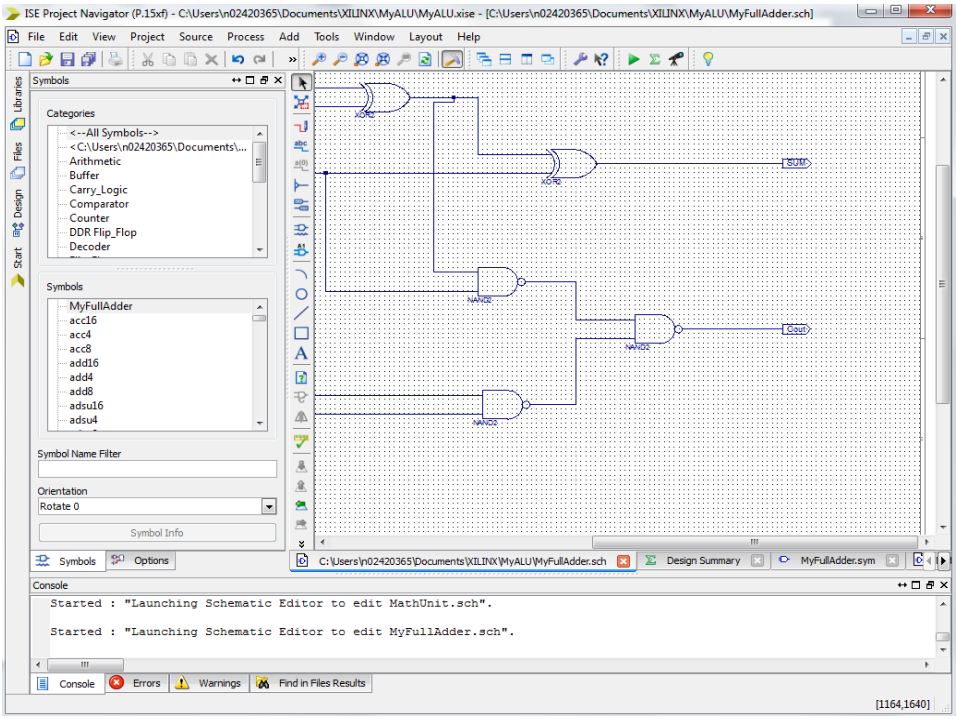
# Xilinx ISE

## <Release Version: 14.1i>

# Simulation Tutorial

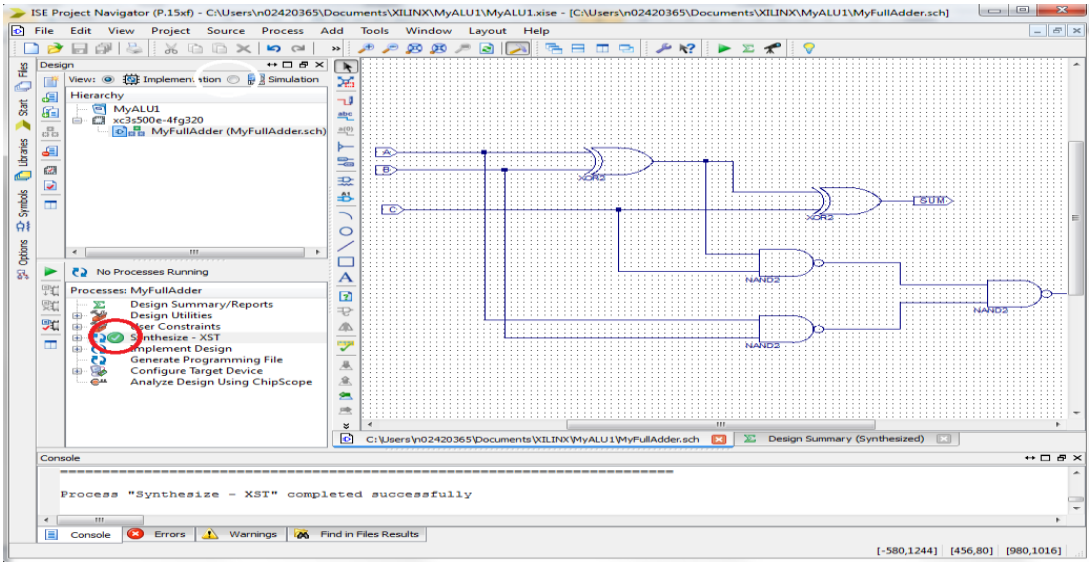

Baback Izadi

You will next test the full adder circuit that you built in the last tutorial via the ISim simulation tool so that you can be sure that it functions per specification. Now let's look at our full adder.

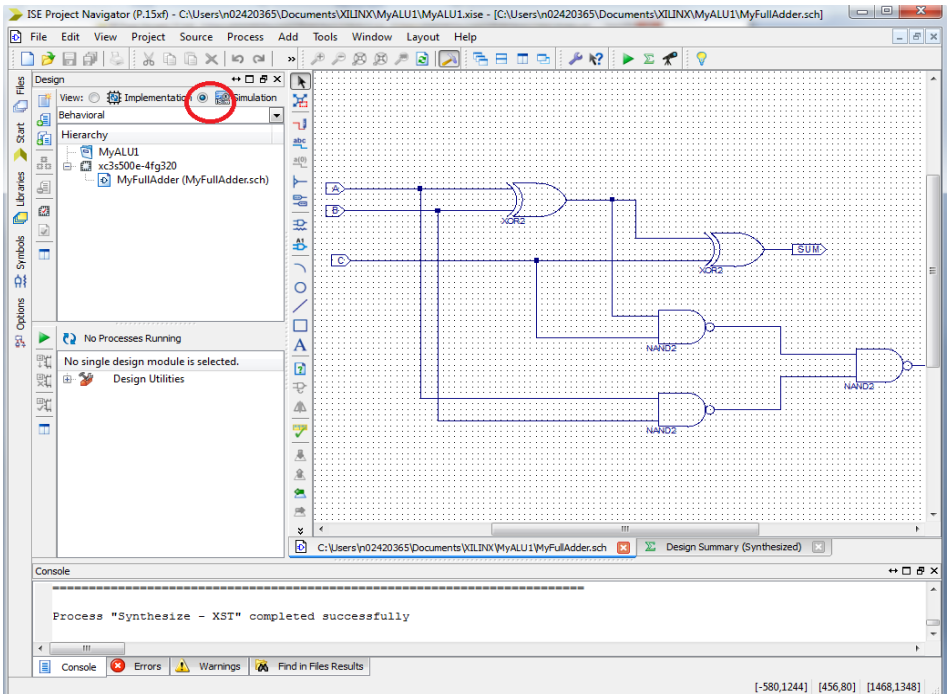



The full adder has three inputs (A, B, Cin) and two outputs (S, Cout). If you have not yet saved your schematic, do so and close the Schematic Editor.

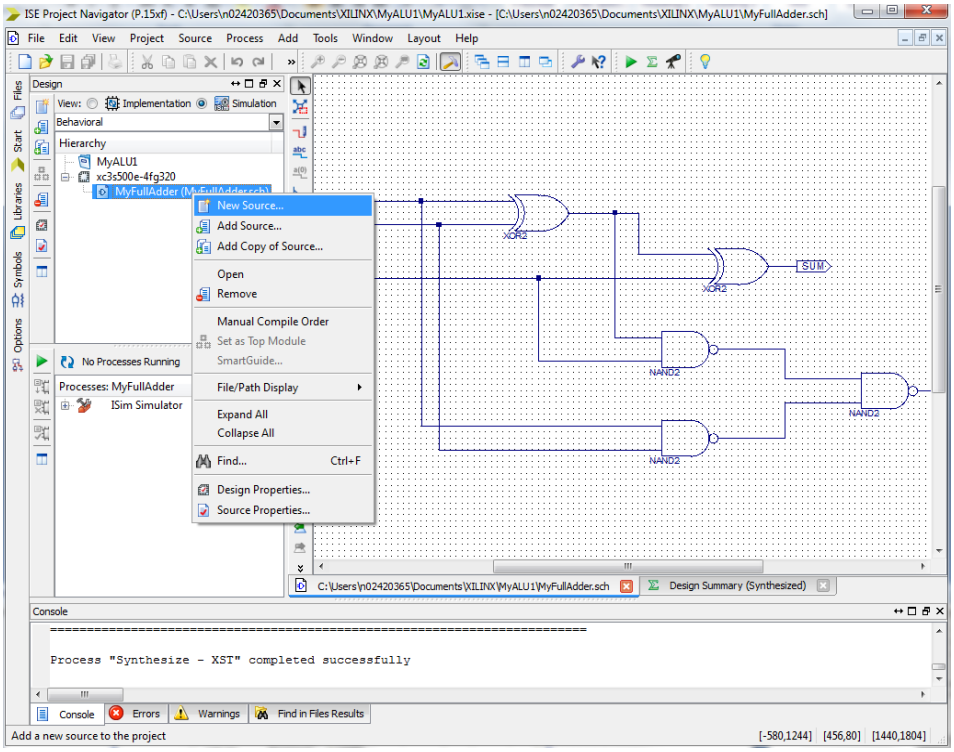
2. Now for the simulation, go to the **Design tab** and **Double Click** on **synthesize-XST.** You should get a green check mark, which means that no error was detected in your schematic.



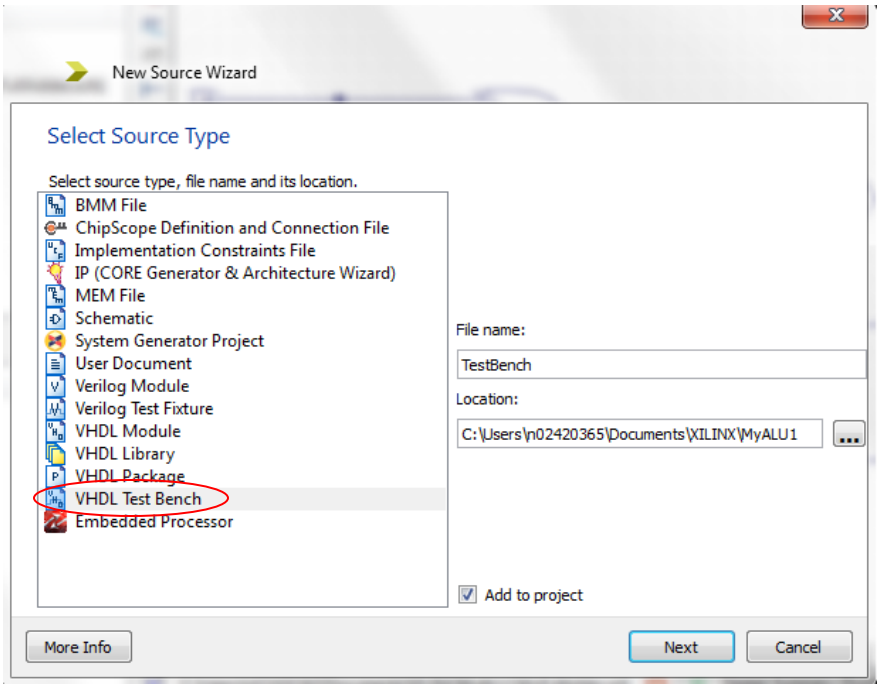
To start the simulation, change your view from Implementation to simulation, as shown below:




3. Next, go back to the Project Navigator. Highlight the source that you want to simulate. In this case, we want to simulate the circuit named MyFullAdder. Then, right click on the source MyFullAdder and select New Source.

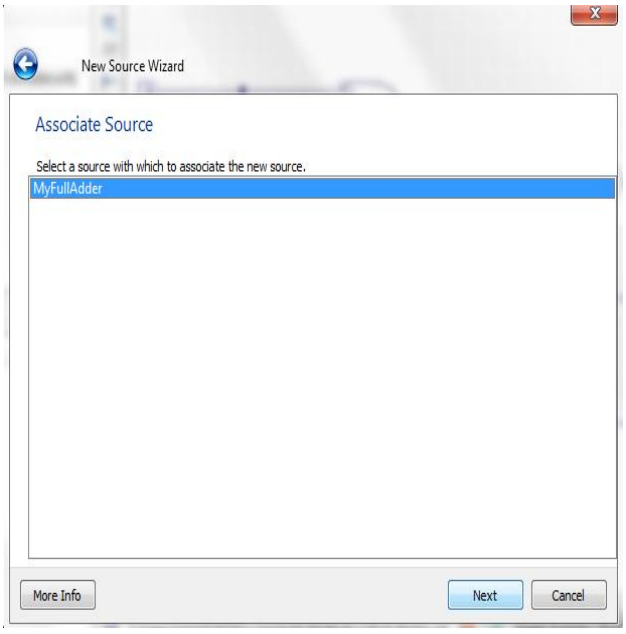


4. This will open a dialog box. Select "**VHDL Test Bench**" as the type of design entry and pick a name for your test-bench file.



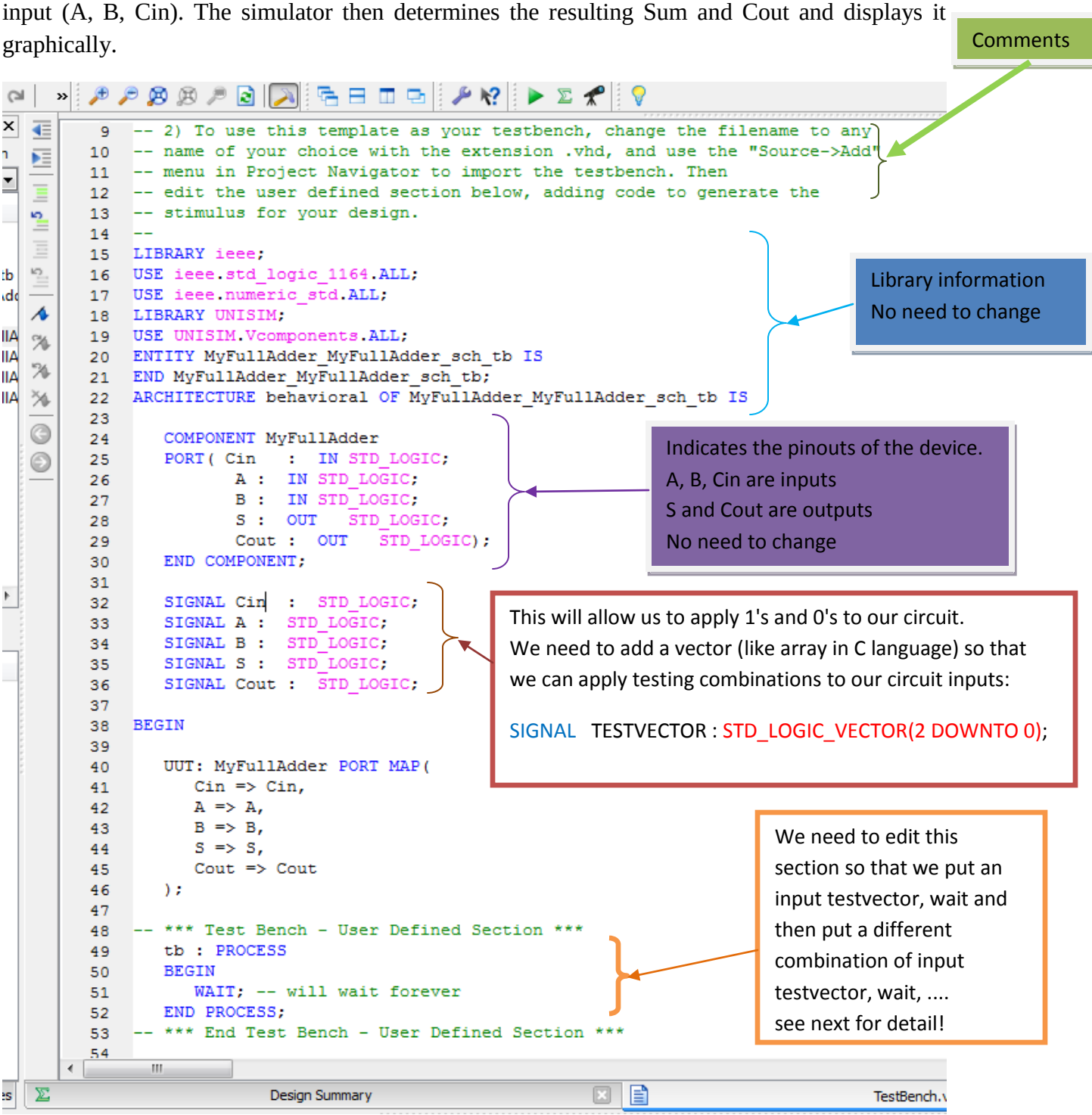

**Click Next**

Pick the design file that your test bench is suppose to simulate. In this case, there is only "MyFullAdder" is listed as your associated source.



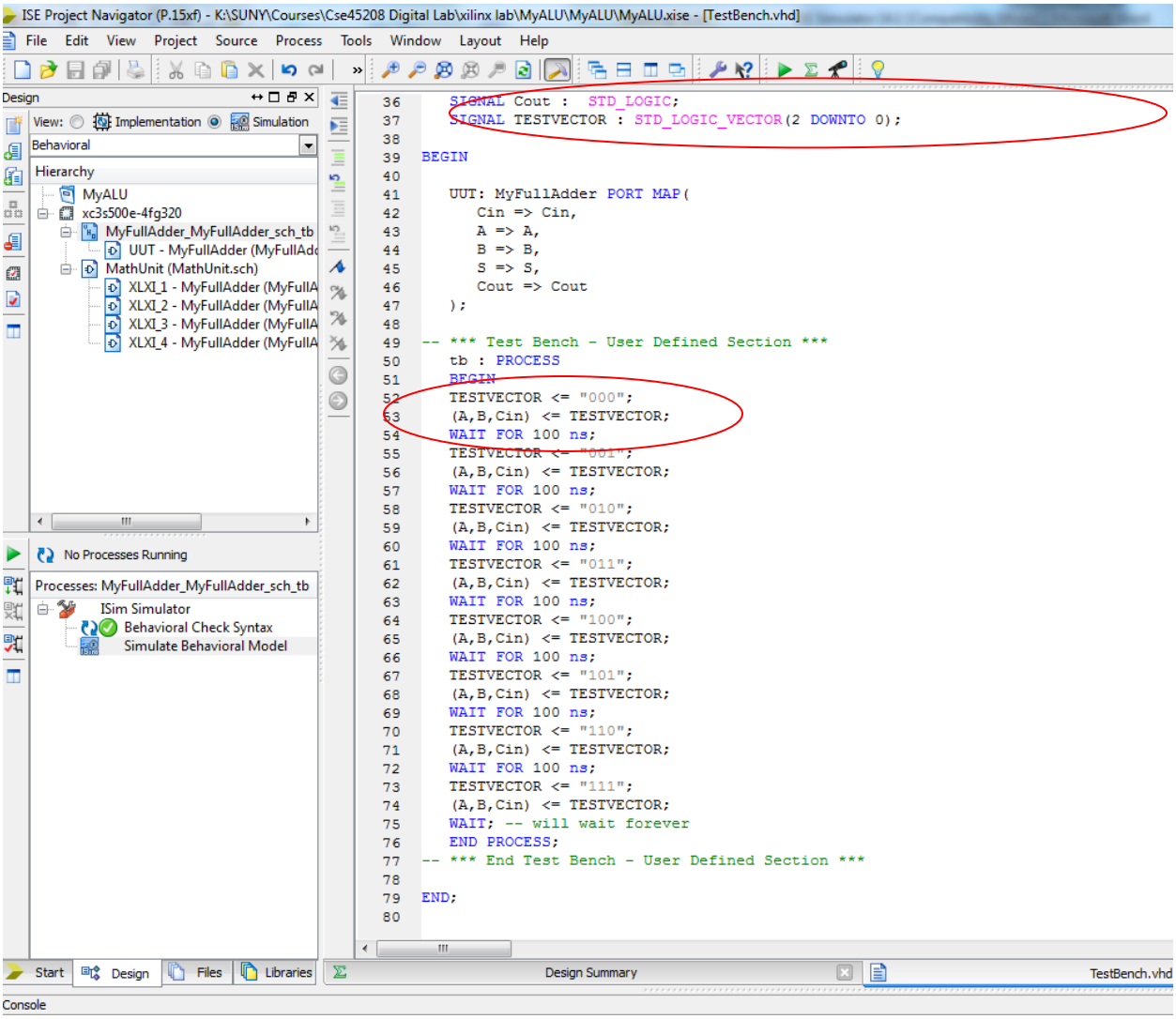

Click ->**Next!** and **finish!**

5. This opens the "**.vhd**" file editor window. This is a template for VHDL code for the full-adder circuit. VHDL code can be used to design circuits as well as simulate existing circuits. In this case, we want to use the VHDL code to simulate our full adder circuit. To do so, we need to make modifications to the code so that at each step, a different value is assigned to the three input (A, B, Cin). The simulator then determines the resulting Sum and Cout and displays it graphically.



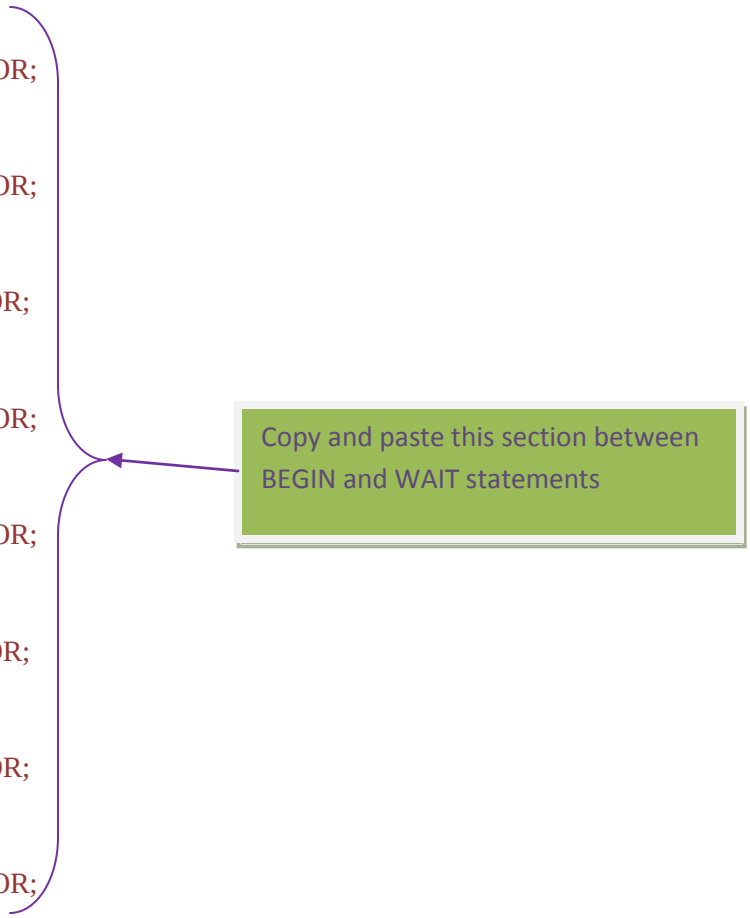

```
-- 2) To use this template as your testbench, change the filename to any
-- name of your choice with the extension .vhd, and use the "Source->Add"
-- menu in Project Navigator to import the testbench. Then
-- edit the user defined section below, adding code to generate the
-- stimulus for your design.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY MyFullAdder_MyFullAdder_sch_tb IS
END MyFullAdder_MyFullAdder_sch_tb;
ARCHITECTURE behavioral OF MyFullAdder_MyFullAdder_sch_tb IS

   COMPONENT MyFullAdder
   PORT( Cin   :  IN STD_LOGIC;
         A :  IN STD_LOGIC;
         B :  IN STD_LOGIC;
         S :  OUT   STD_LOGIC;
         Cout :  OUT   STD_LOGIC);
   END COMPONENT;

   SIGNAL Cin  :  STD_LOGIC;
   SIGNAL A :  STD_LOGIC;
   SIGNAL B :  STD_LOGIC;
   SIGNAL S :  STD_LOGIC;
   SIGNAL Cout :  STD_LOGIC;

BEGIN

   UUT: MyFullAdder PORT MAP(
      Cin => Cin,
      A => A,
      B => B,
      S => S,
      Cout => Cout
   );

-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

We start the modification, by adding: SIGNAL TESTVECTOR : STD_LOGIC_VECTOR (2 DOWNTO 0);



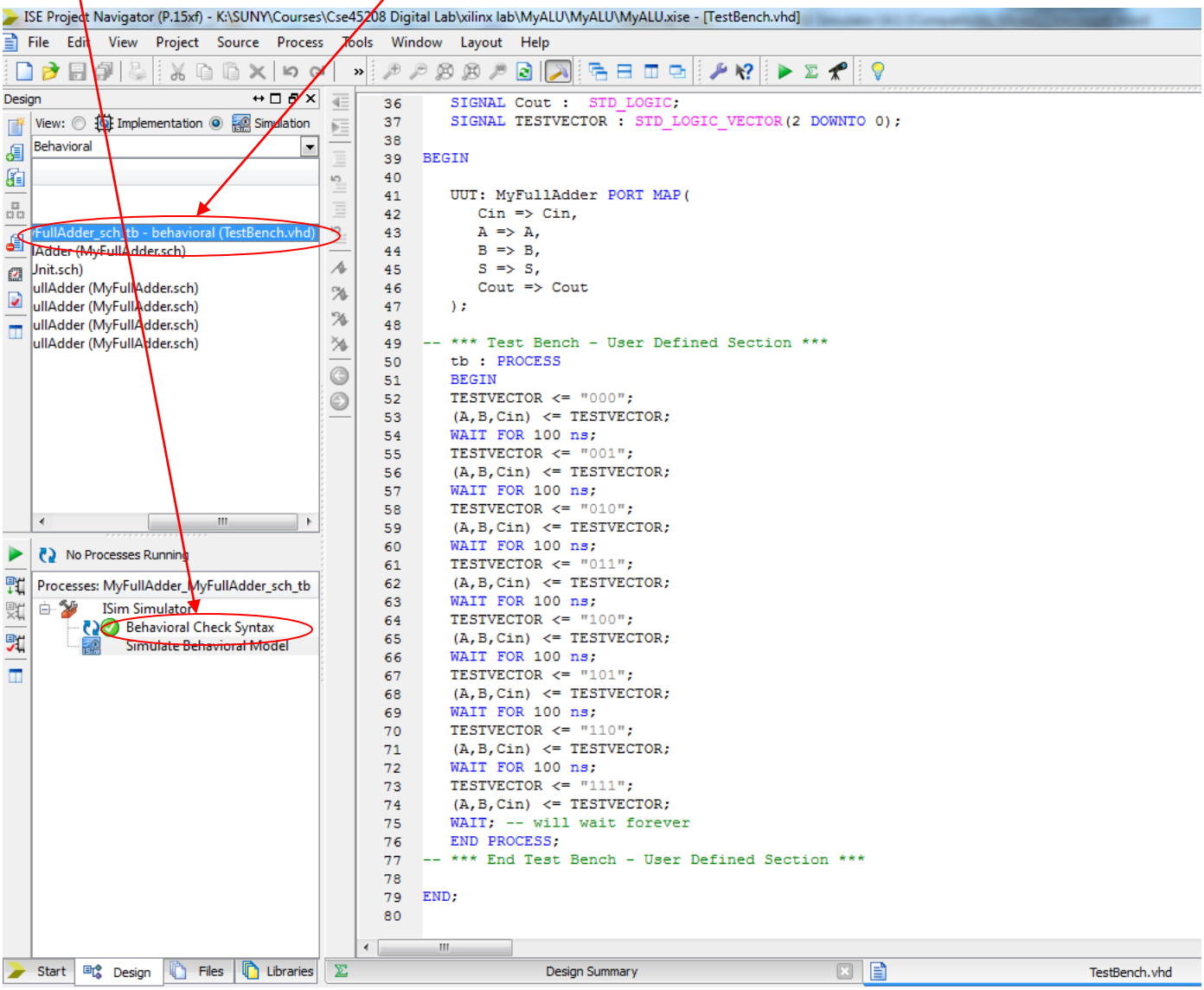

And under the **Test Bench -User Defined Section,** we need to assign different test cases. Since, we have three inputs A, B, and Cin, they could take on values from 000 to 111. For each case, we put the value in the TESTVECTOR, input the TESTVECTOR to A, B, and Cin using (A,B,Cin) <= TESTVECTOR command, and then wait for a few nanoseconds before repeating the process for a different input combination. So, the user defined section would look like the following:

```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
  BEGIN
TESTVECTOR <= "000";
 (A,B,Cin) <= TESTVECTOR;
WAIT FOR 100 ns;
TESTVECTOR <= "001";
 (A,B,Cin) <= TESTVECTOR;
WAIT FOR 100 ns;
TESTVECTOR <= "010";
(A,B,Cin) <= TESTVECTOR;
WAIT FOR 100 ns;
TESTVECTOR <= "011";
 (A,B,Cin) <= TESTVECTOR;
WAIT FOR 100 ns;
TESTVECTOR <= "100";
 (A,B,Cin) <= TESTVECTOR;
WAIT FOR 100 ns;
TESTVECTOR <= "101";
(A,B,Cin) <= TESTVECTOR;
WAIT FOR 100 ns;
TESTVECTOR <= "110";
(A,B,Cin) <= TESTVECTOR;
WAIT FOR 100 ns;
TESTVECTOR <= "111";
 (A,B,Cin) <= TESTVECTOR;
Wait; -- Will wait forever
 END PROCESS;
-- *** End Test Bench - User Defined Section ***
```




Now save your ".**vhd**" file

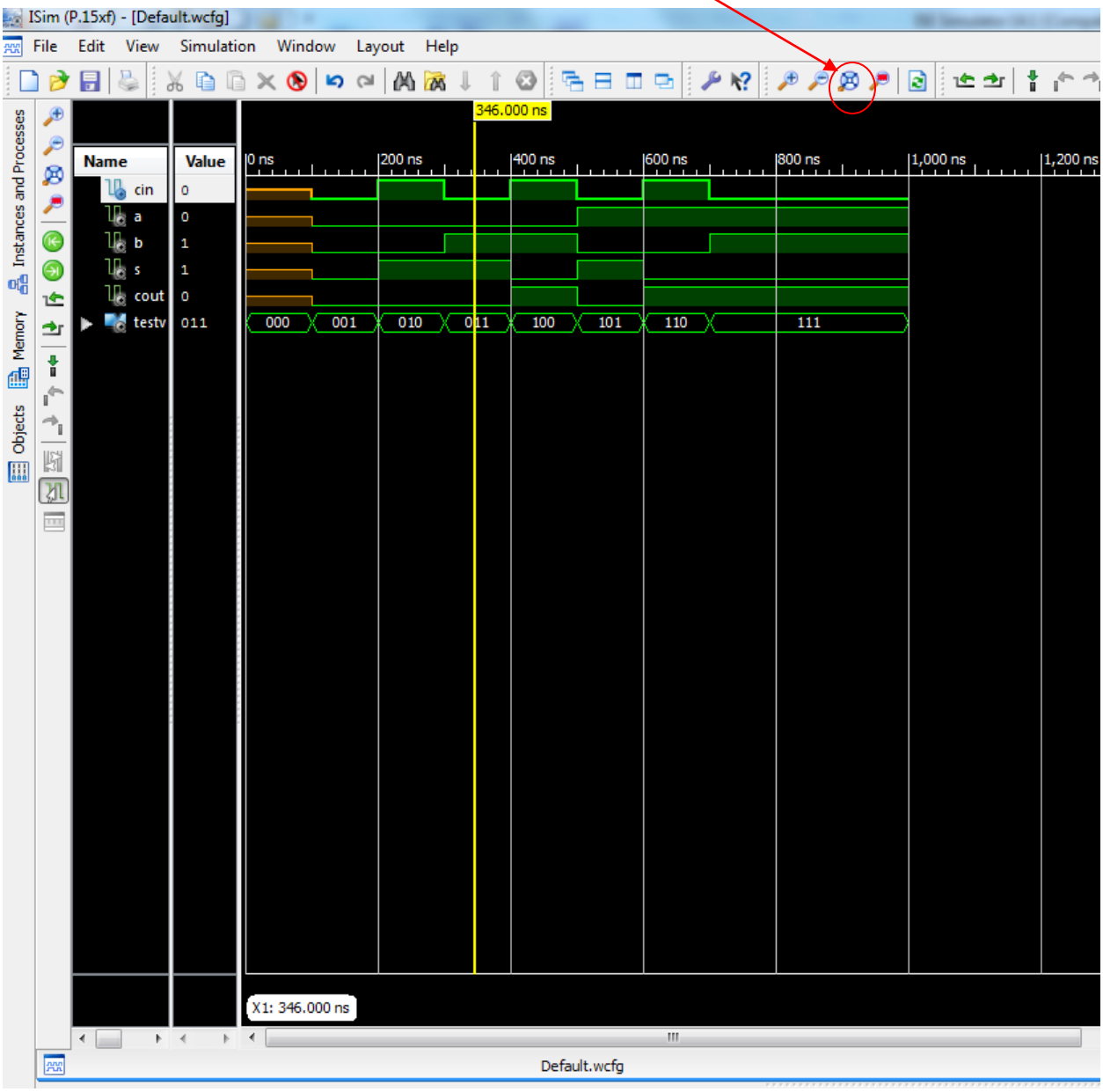
6. Next, highlight the "**.vhd**" file on the left hand side of the window under the design section, and expand the ISim Simulator.




8. Double click on the **Behavioral Check Syntax**

**7. O**nce you see the green check mark double click on the **Simulate Behavioral Model**.

8. This will open the waveform "**ISIM window".** ZOOM to the full view. Now you can see the waveforms of your full adder that you just simulated.




You can position the cursor on the various points and see the corresponding value displayed on the left side of the display. For example, for the current cursor position, Cin = 0, A = 0, B = 1, resulting in S = 1 and Cout = 0; which is the correct result.

Once you are convinced that your design is correct, you can move the next step in your design entry.

Note that to test large circuits, it is not possible to test all possible combinations. So, some reasonable combinations should be tried. For example for the final case, you may try several testvectors: One for four bit A, another for four bit B and yet another M, S1, and S0. So, your added signal lines would look something like:

```
SIGNAL   TESTVECTOR_A : STD_LOGIC_VECTOR (3 DOWNTO 0);

SIGNAL   TESTVECTOR_B : STD_LOGIC_VECTOR (3 DOWNTO 0);

SIGNAL   TEST_SELECT : STD_LOGIC_VECTOR (2 DOWNTO 0);
```

In the first and second one, I assume that the order will be $A_3$ to $A_0$, $B_3$ to $B_0$. That is how we define it 3 downto 0 -- a total of four bits. Moreover, TEST_SELECT has three bits that can take care of M, $S_1$, and $S_0$.

So, the following code will allow 0110 to be added to 0010

```
TESTVECTOR_A <= "0110";
TESTVECTOR_B <= "0010";
TEST_SELECT <= "000"
 (A3, A2, A1, A0) <= TESTVECTOR_A;
(B3, B2, B1, B0) <= TESTVECTOR_B;
(M, S1, S0) <=TEST_SELECT
WAIT FOR 100 ns;
```

Furthermore, to subtract 0010 from 0110, we only need to change S0 t0 1:

```
TEST_SELECT <= "001"
 (M, S1, S0) <=TEST_SELECT
WAIT FOR 100 ns;
```

Using this, you may simulate your other circuits.

**GOOD LUCK!!**