Chapter-6

Implementation of Logic Circuits by Using Schematic Designs on FPGA

FPGA vendors also offer schematic design features to program FPGA besides VHDL and Verilog programming language choices. Eventually, they all can be used in a hybrid structure to achieve the goal. In this chapter, schematic design examples are going to be represented on Xilinx's ISE Design Suite.

ISE Design Suite offers a graphical user interface to choose both combinational and sequential logic block. Designs start by simply dragging these symbols to project area. After dragging wire connections are done. When input and output (I/O) ports are added, project becomes ready for simulation and real time FPGA implementation.

There are many digital design blocks presented by the editor. List of these blocks are given below:

- Arithmetic
- Buffer
- Carry-Logic
- Clocking Resources
- Comparator
- Counter
- Decoder
- Flip-Flop
- Latch
- Logic
- Memory
- Mux
- Shift Register

Example: Implement Boolean function $f(x, y, z) = x'y' + y'z$ in schematic design editor in ISE Design Suite. Test the design in ISIM simulator.

Solution: Step-1) As the first step to achive our task, we need to open a new project. After setting file our project file, ISE Design Suite asks for the FPGA chip family, device, package and speed properties of the chosen FPGA. Set the FPGA chip of Nexys 4 DDR programming kit. As mentioned before, these properties can be achived simply by reading the top side of the FPGA chip.

Step-2) After configuration, click next. An empty project with XC7A100T is created. By right clicking the FPGA chip on the project screen choose "New Source" (Figure 6-1).

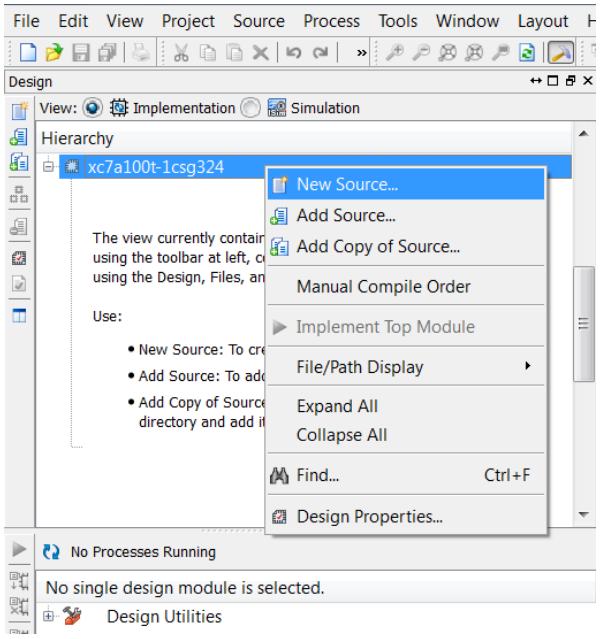


Figure 6-1

Open a "Schematic" and name it as "Schematic_of_f_xyz", Figure 6-2. Click **Finish** in the next window.

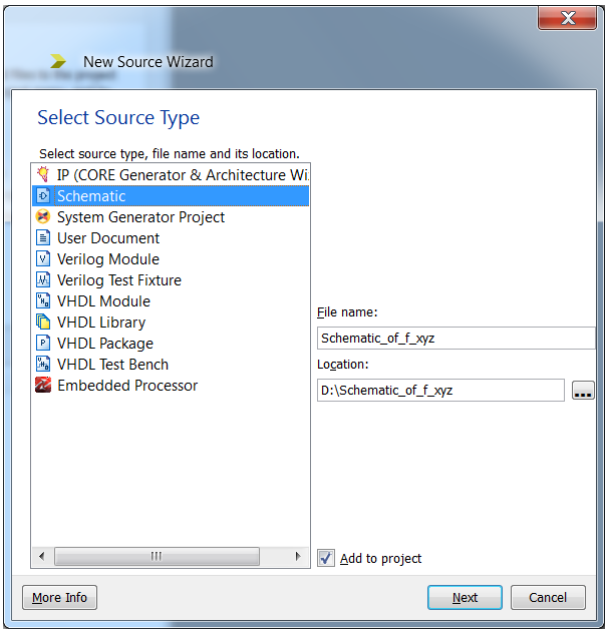


Figure 6-2

Figure 6-3 shows the Schematic window and list of symbols.

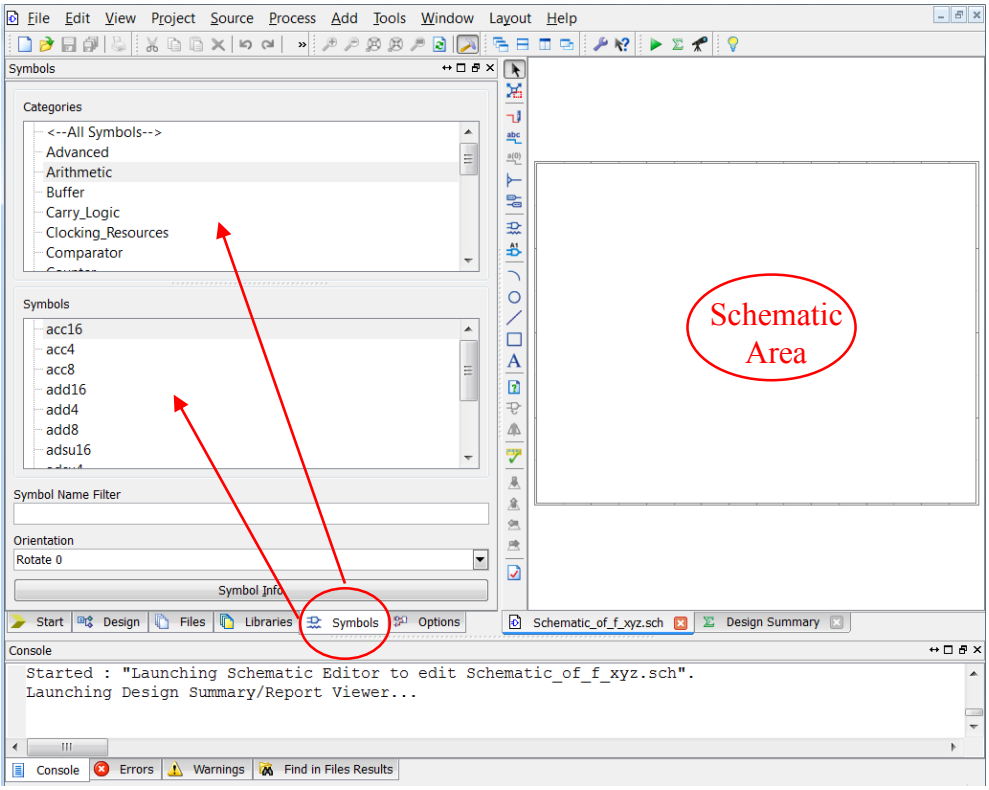


Figure 6-3

Step-3) Now we are ready to implement our function f. We need simple logic gates to achieve our task. Choose **Logic** (Figure 6-4) from the categories and add NOT, 2-input AND and 2-input OR gates.

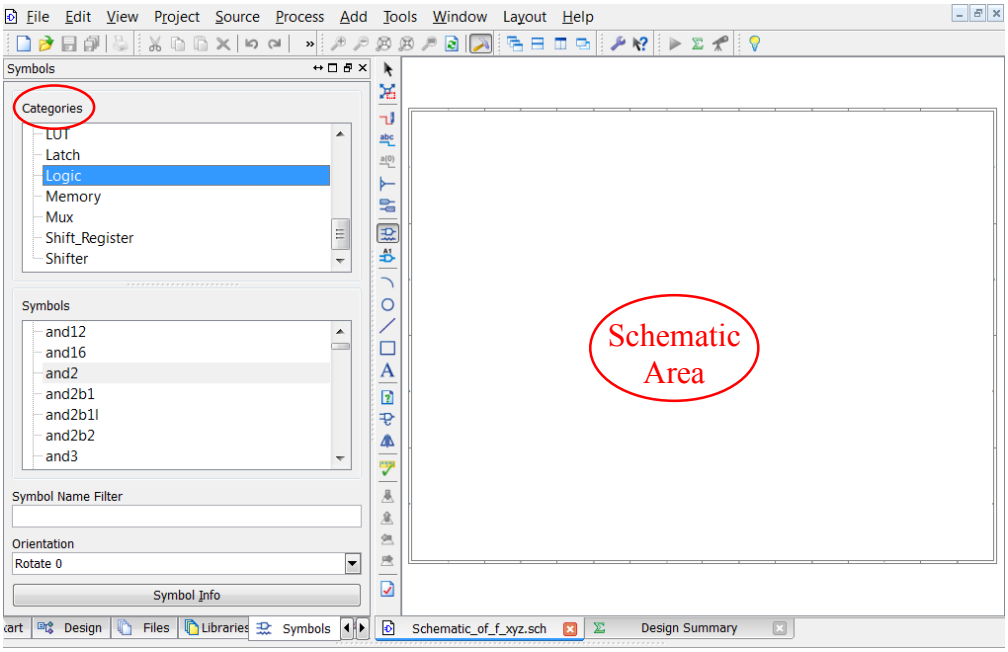


Figure 6-4

In Figure 6-5, logic gates that are needed to implement $f(x, y, z) = x'y' + y'z$ are added.

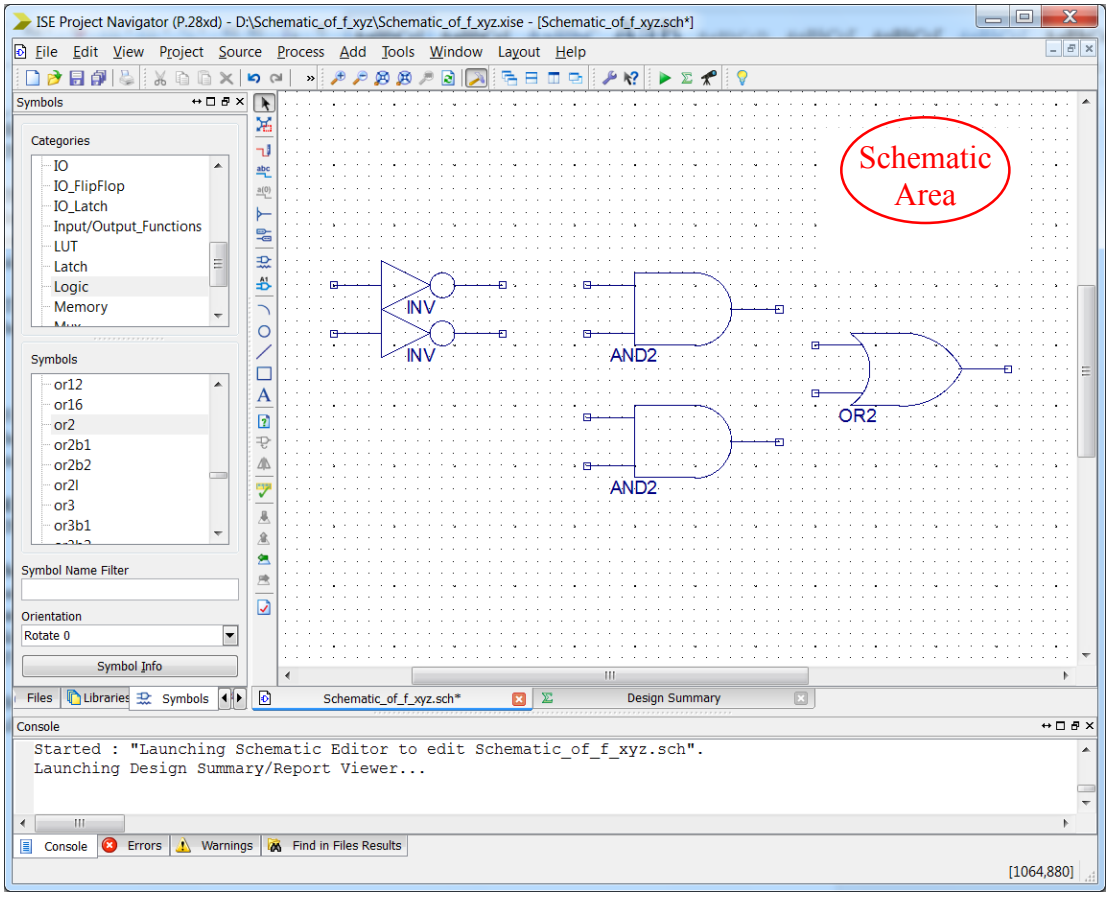


Figure 6-5

Step-4) In this step, we are going to add I/O pins. Three one-bit inputs and one-bit output are needed. Click the "Add I/O Marker" button as shown in Figure 6-6.

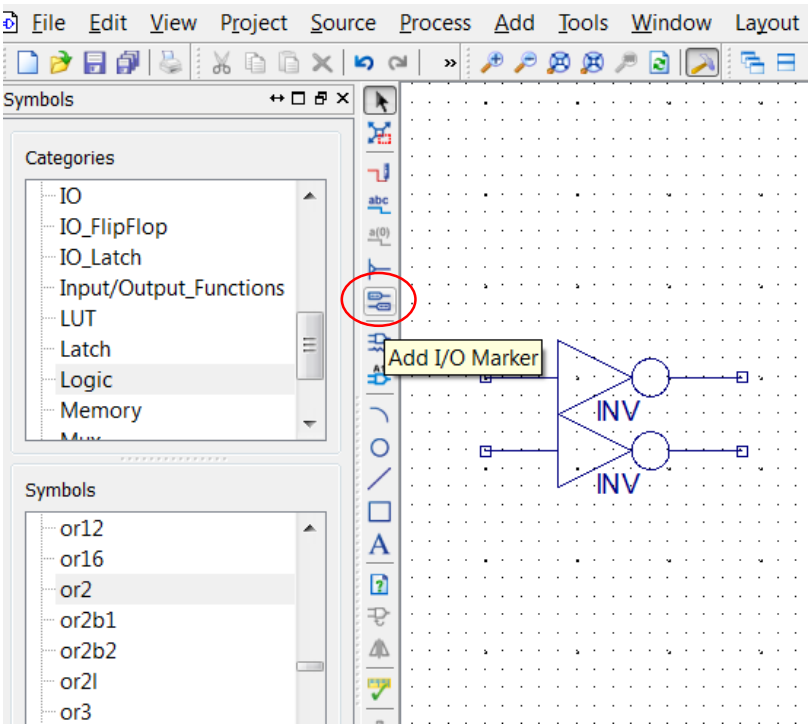


Figure 6-6

Add three input markers and one output marker to the project area. I/O marker options are shown in Figure 6-7 and Figure 6-8.

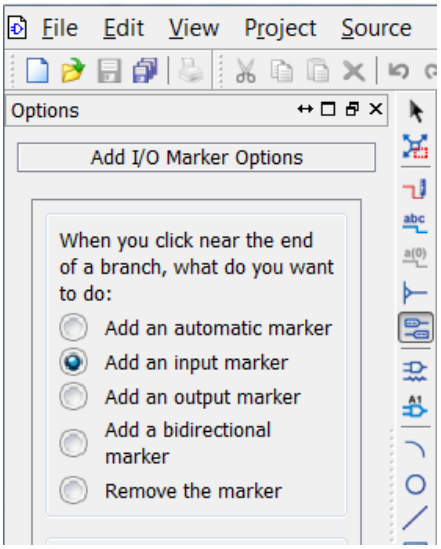


Figure 6-7

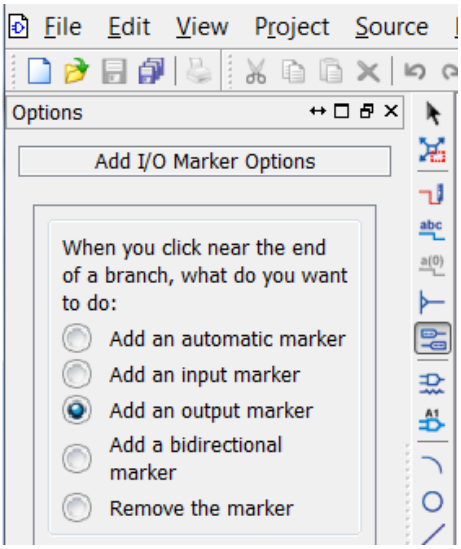


Figure 6-8

Pointy edge of the input marker should be put on the logic gate's input as in Figure 6-9. There is a square at the beginning of the input pin. If the pointy edge is put on this square then it

means connection is successful. Vertical edge of the output marker should be put to the output of the logic gate as it seen in Figure 6-10.

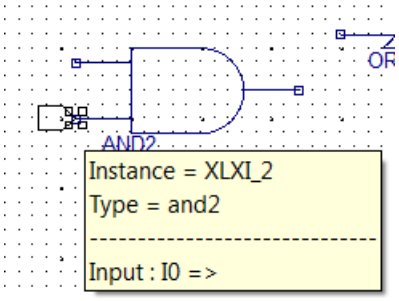


Figure 6-9

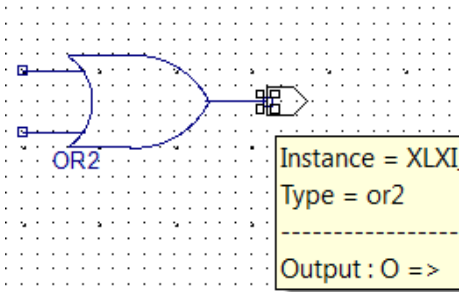


Figure 6-10

I/O markers (ports) are added. Their names automatically set as XLXN_1, XLXN_2, XLXN_3 and XLXN_4. Change their name as x, y , z , f , respectively.

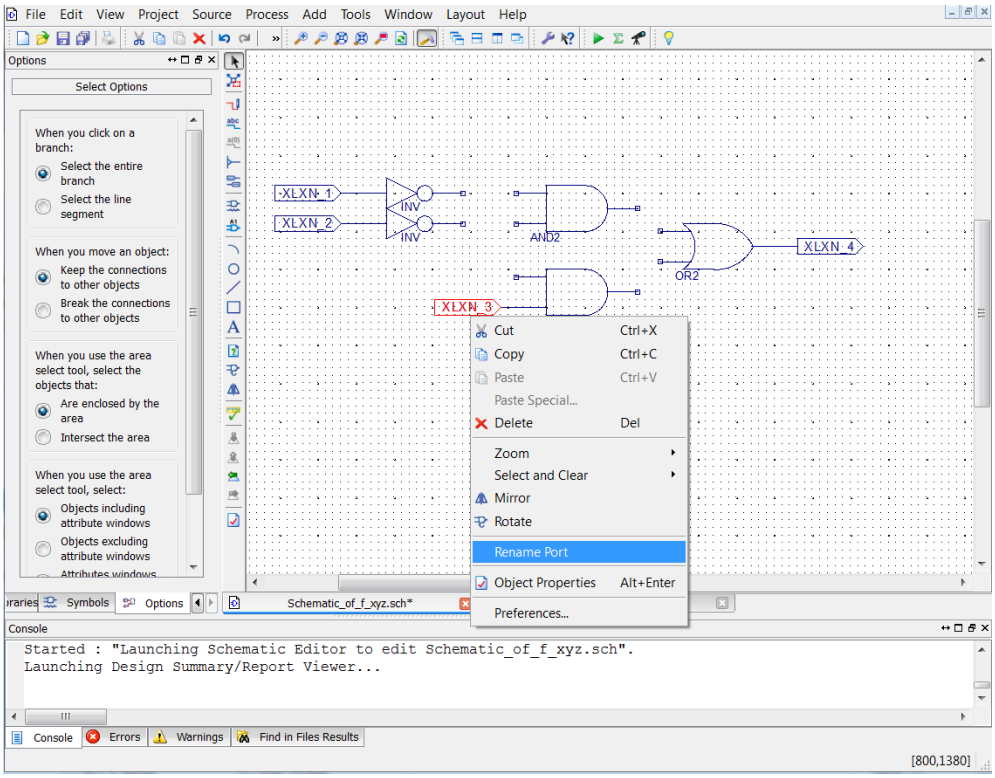


Figure 6-11

Step-5) Logic gates and I/O markers are added to the schematic project area. Now, turn is to connect them. Click the **Add Wire** (Figure 6-11) button to be able to draw connections between components.

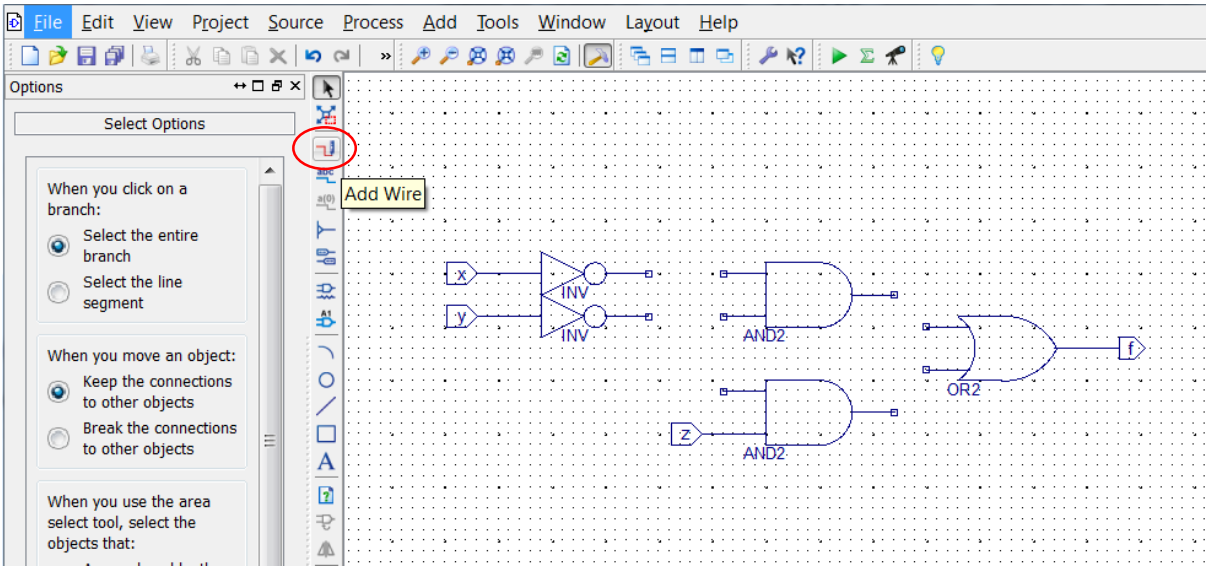


Figure 6-12

Wiring operation is completed. We concluded our design, Figure 6-13.

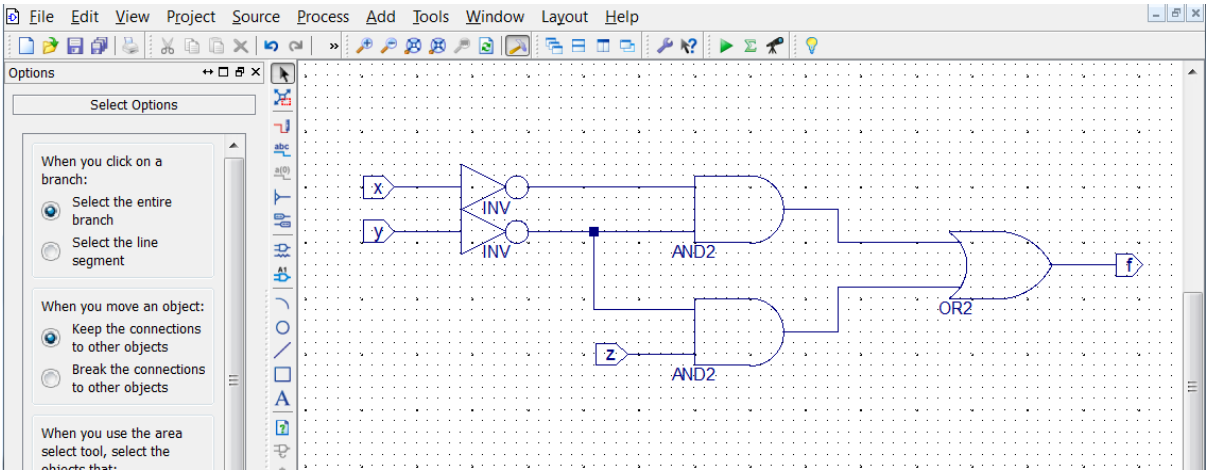


Figure 6-13

Step-6) At this step, we need to test our schematic design. Pass from **Symbol** window to **Design** window to see our project files. It can be easily seen that (Figure 6-14) file extension of our implementation is "*.sch" as it was named before "*.vhd" when we're dealing with VHDL programming.

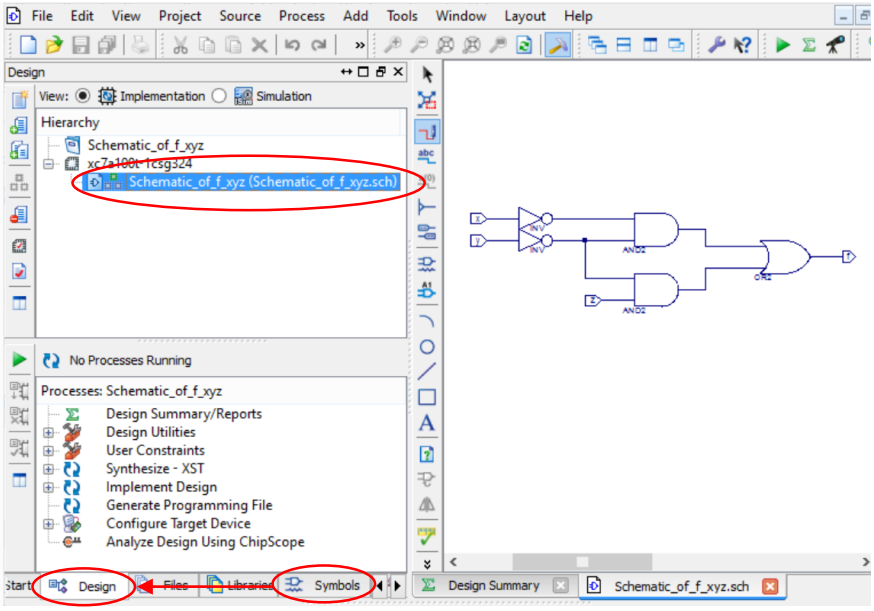


Figure 6-14

Right click the FPGA chip and add a **New Source** to the project. We are going to add our test bench code. Figure 6-15 shows the operation.

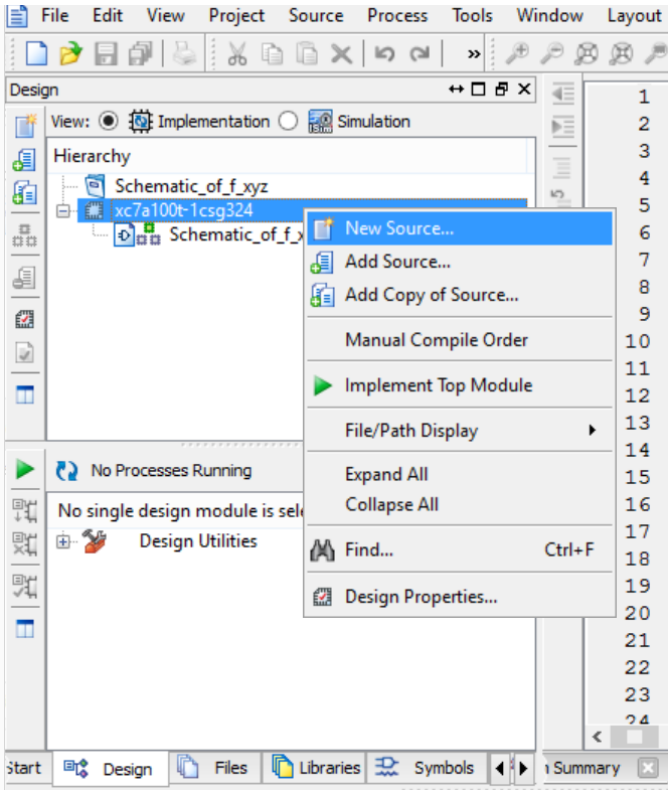


Figure 6-15

Open a VHDL Test Bench and name it as “f_xyz_TB”. Click **Next**.

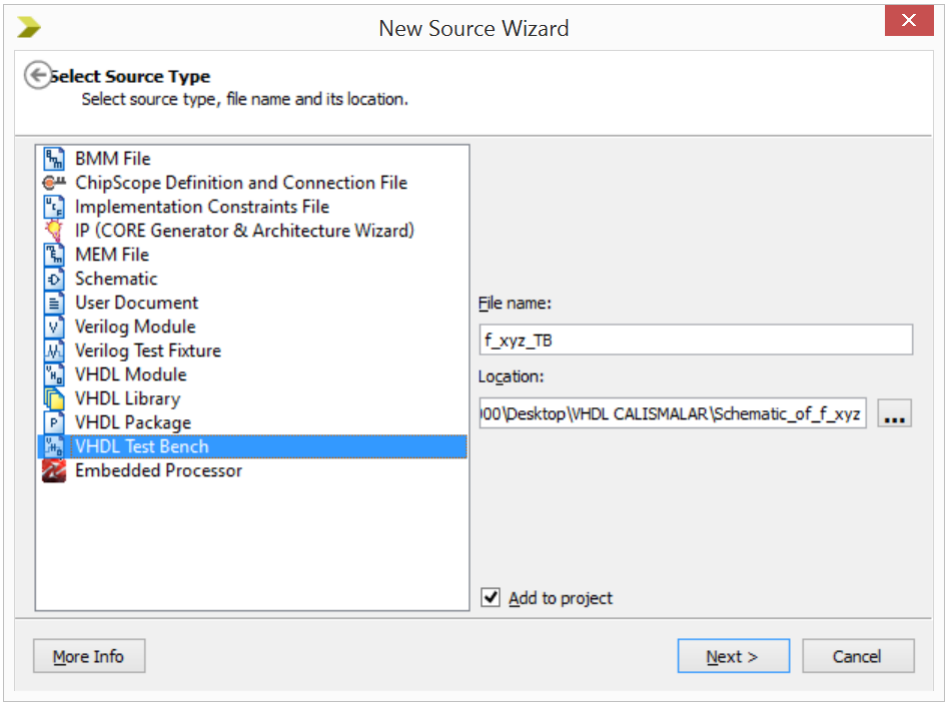


Figure 6-16

Select our schematic file as associate source for the simulation and click **Next**, as in Figure 6-17.

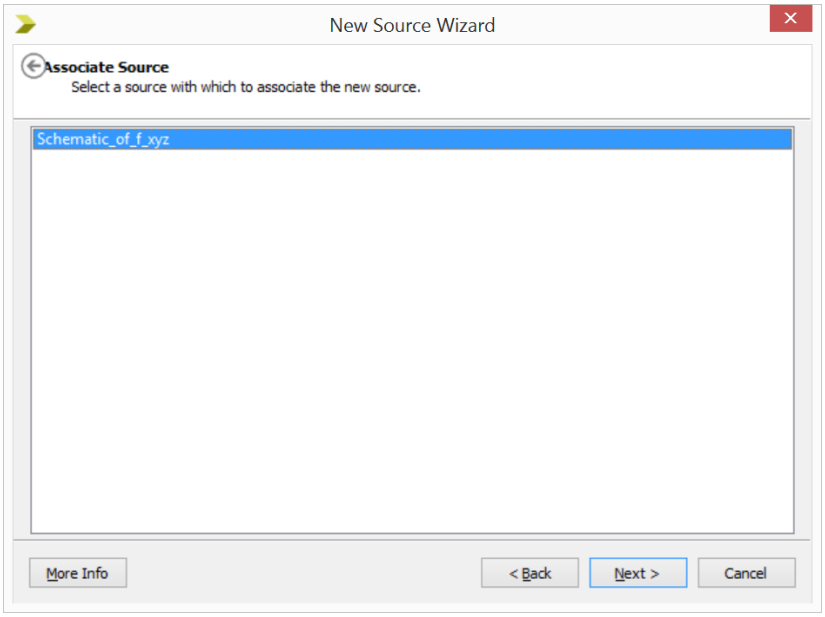


Figure 6-17

A core test bench appears like in Figure 6-18. Since we have schematic file for implementation, UNISIM library is added to the project.

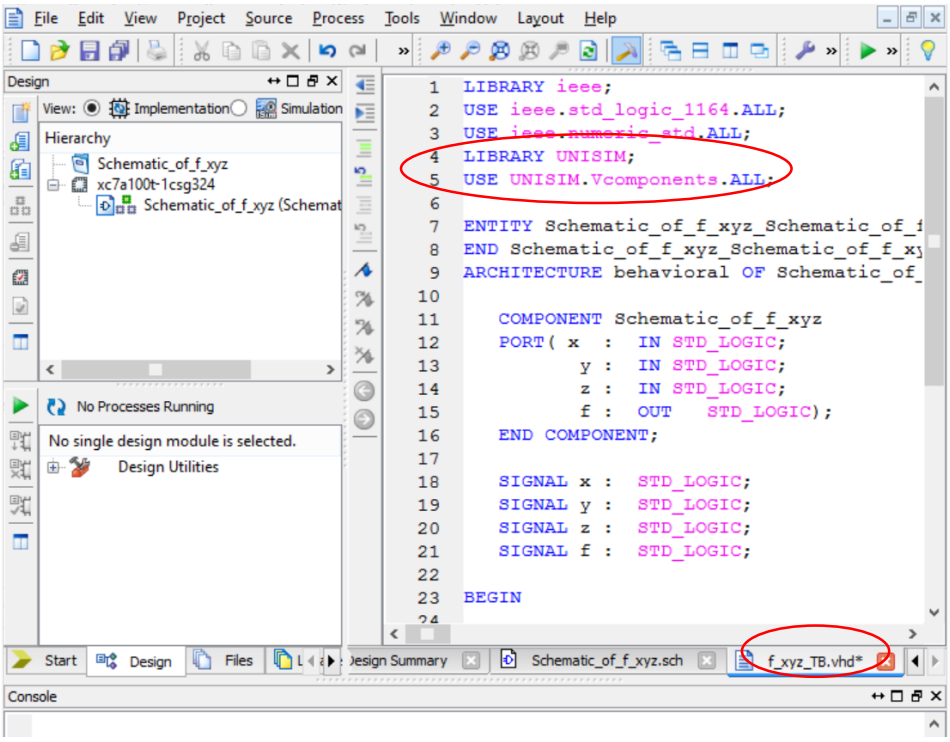


Figure 6-18

Add your test bench to below test code. This code covers all the states that we can achieve with three inputs. In this configuration, whole simulations takes 800 ns to end.

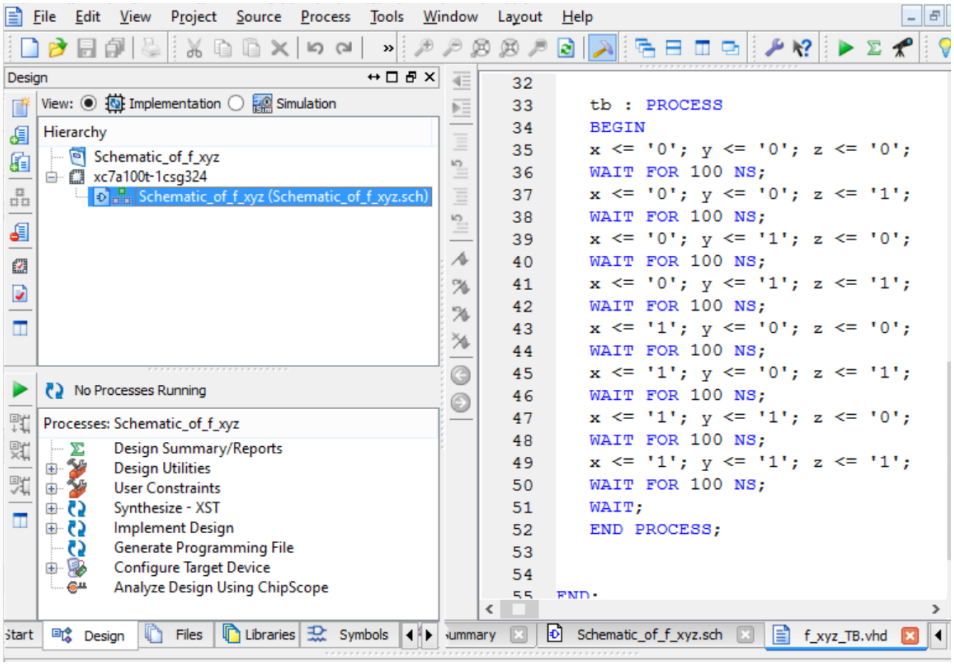


Figure 6-19

After this point, we have completed the test bench construction. Pass through the **Simulation** part of the design and choose our test bench. In order to simulate double click **Simulate Behavioral Model**.

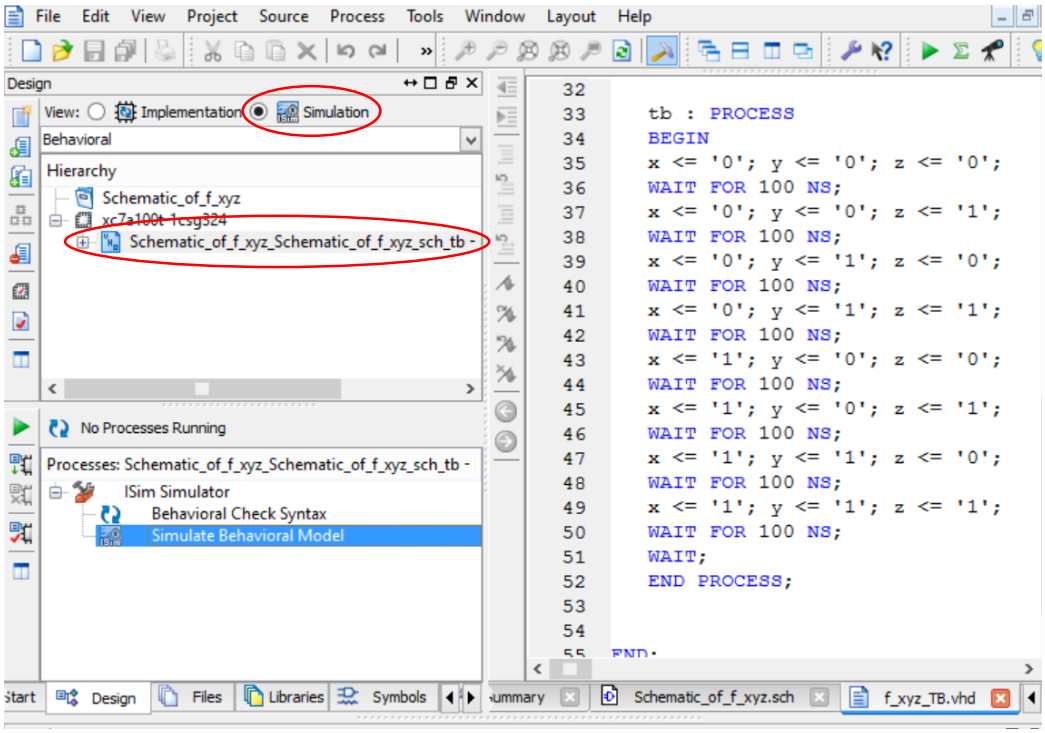


Figure 6-20

As a result of this test bench, we should observe the outcomes that are presented in Table 6-1.

Table 6-1

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Figure 6-21 shows the result of the ISIM Simulator test. If it's compared with Table 6-1, it can be say that results are fit.

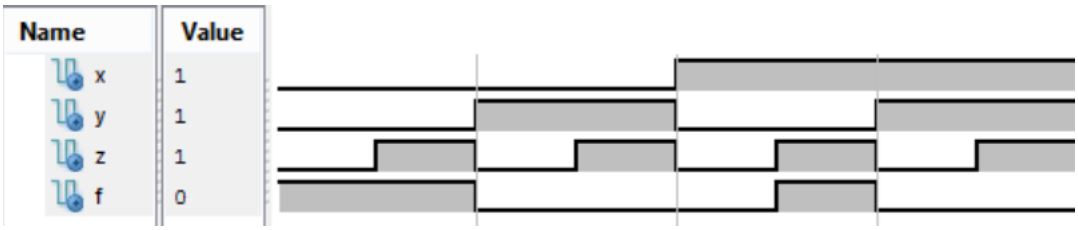
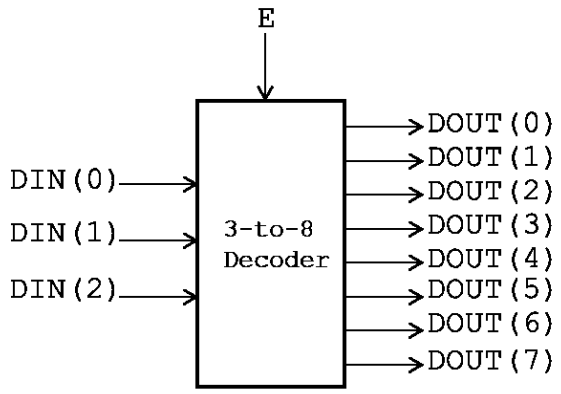


Figure 6-21

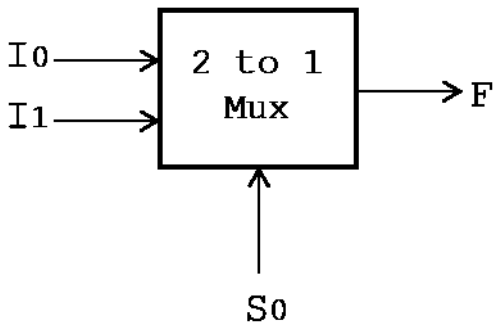
Example: Design a simple 3-to-8 Decoder whose structure and truth table are given below in Figure 5-20. Create "*.bit" file, upload it to Nexys 4 DDR development kit and observe the results.



E	D_IN(3 bits)	D_OUT(8 bits)
0	XXX	ZZZZZZZZ
1	000	10000000
1	001	01000000
1	010	00100000
1	011	00010000
1	100	00001000
1	101	00000100
1	110	00000010
1	111	00000001

Figure 6-22

Exercise: Implement the 2-to-1 multiplexer which is depicted in Figure 6-22. Inputs and output are should be two bit wide. Only the select bit of the multiplexer is one bit. Use the schematic model of 2 to 1 multiplexer. Create "*.bit" file, upload it to Nexys 4 DDR development kit and observe the results.



S0	F
0	I0
1	I1

Figure 6-23