Chapter-5

Programming the FPGA

In previous chapters, several digital circuits are implemented by using VHDL. In chapter 4, functional tests of the implemented designs are done. In this manner, next step is to upload our program to FPGA. FPGA coding and programming tools differs in between FPGA families of Xilinx, Altera, Actel, etc. In this lecture, programming via Xilinx's ISE Design Suite is going to be demonstrated for the explanation of the process.

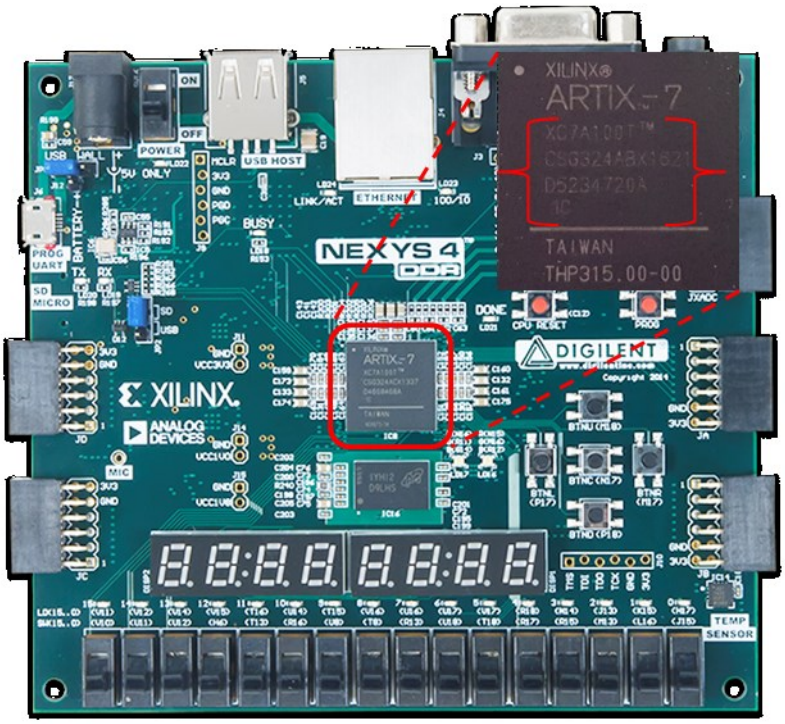


Figure 5-1

Figure 5-1 shows the Nexys 4 DDR programming kit and focuses on the FPGA chip. This IC belongs to the Xilinx's Artix 7 family with device code XC7A100T. This FPGA chip is going to be used during rest of the chapter.

Example: VHDL code of Boolean function $f(x, y, z) = x'y' + y'z$ is given below PS 5.1. Open a new project. Create its bit file. Upload it to Nexys 4 DDR development kit and see the results.

```
library ieee;
use ieee.std_logic_1164.all;

entity f_xyz is
  port( x, y, z: in std_logic;
        f: out std_logic );
end entity;

architecture logic_flow of f_xyz is

begin
        f<= (x nor y) or (not y and z);
end architecture;
```

PS 5.1

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. These properties can be achived simply by reading the top side of the FPGA chip. In our case, it can be read from the Figure 5-1. Figure 5-2 shows the overall assignment.

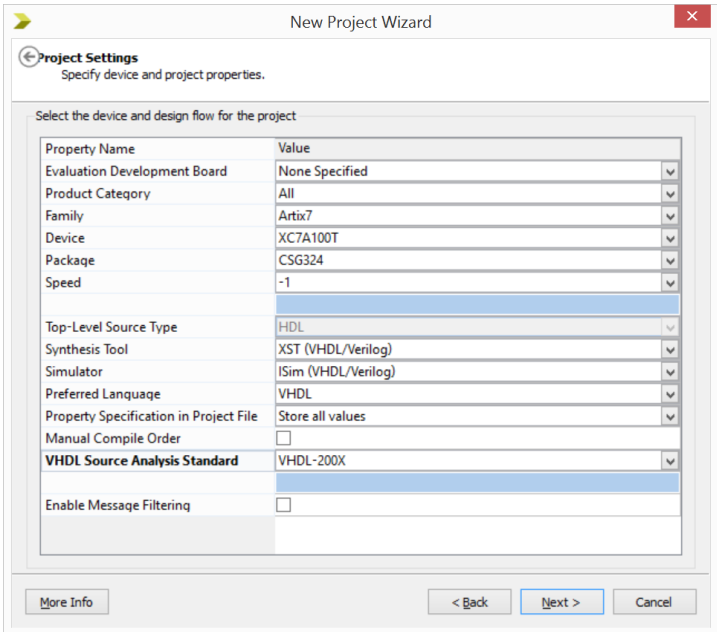


Figure 5-2

Step-2) After configuration, click next. An empty project with XC7A100T is created. VHDL file and its pin assignment should be added to the project. By right clicking the FPGA chip on the project screen open a VHDL Module and name it “f_xyz”.

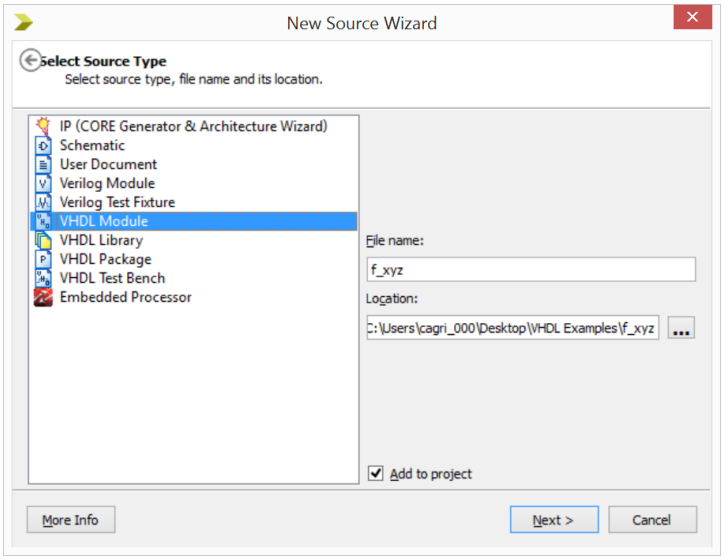


Figure 5-3

After typing project name click next. Enter **architecture** name and I/O port names as seen in the Figure 5-4. Click next.

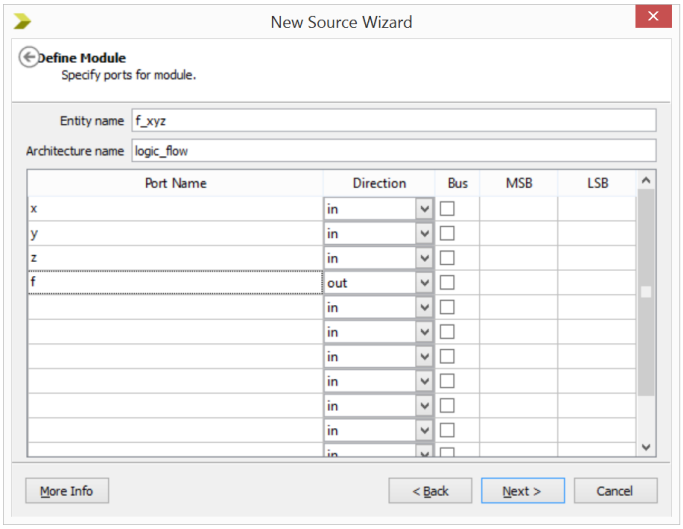


Figure 5-4

Step-3) Our function is implemented. Now we have to tie FPGA pins to our variables x, y, z, f. Nexys 4 DDR development kit has basic electronic components to realize input signal possibilities and observe results. Among these components, we are going to use switches as inputs and an LED as output. As a result, we are going to use three switches (SW0, SW1, SW2) and an LED (LED0). Pin information about Nexys 4 DDR FPGA board can be found from the internet easily. After choosing our peripherals from Nexys 4 DDR FPGA board reference manual, now we can define them in our ISE project.

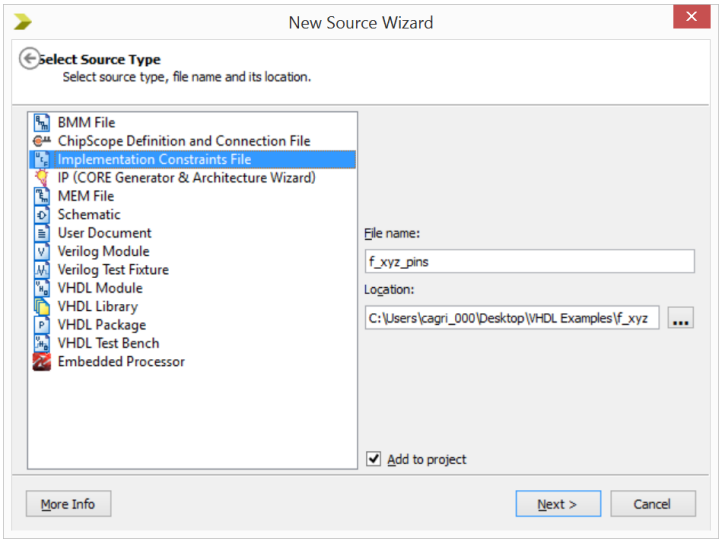


Figure 5-5

Step-4) By right clicking the FPGA chip on the project screen open a "Implementation Constraints File" and name it "f_xyz_pins". Click Next.

Step-5) An empty file with the extension of "*.ucf". Connect your one bit I/O variables to SW0, SW1, SW2 and LED0 as shown below in Figure 5-6. It is also important to state that "f_xyz_pins.ucf" file is in the sub directory of "f_xyz.vhd" file on the Hierarchy window of the design. It means constraints file is created correctly.

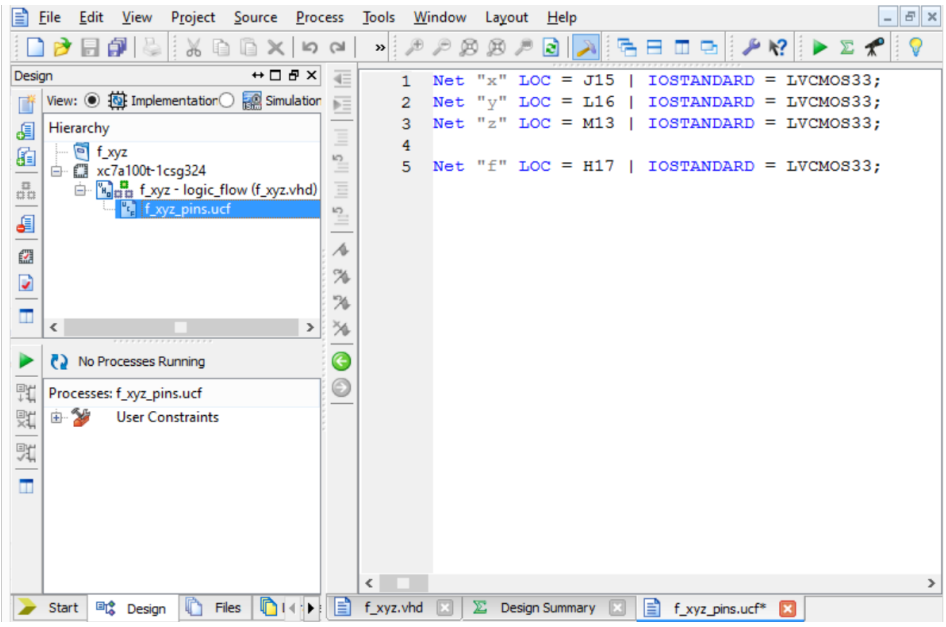


Figure 5-6

Step-6) We are ready to generate our "*.bit" file. Select your main file "f_xyz.vhd" and double click to *Generate Programming File* button as seen in Figure 5-7. By clicking this button

translating, mapping and routing stages start to work. If editor find no mistakes then green check mark emerges (Figure 5-8) and it means our "*.bit" file is generated inside the project file.

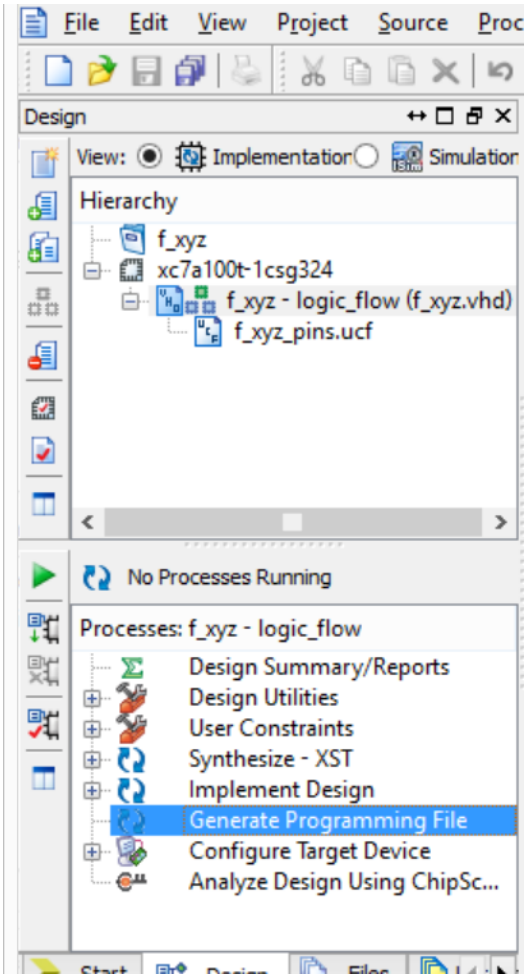


Figure 5-7

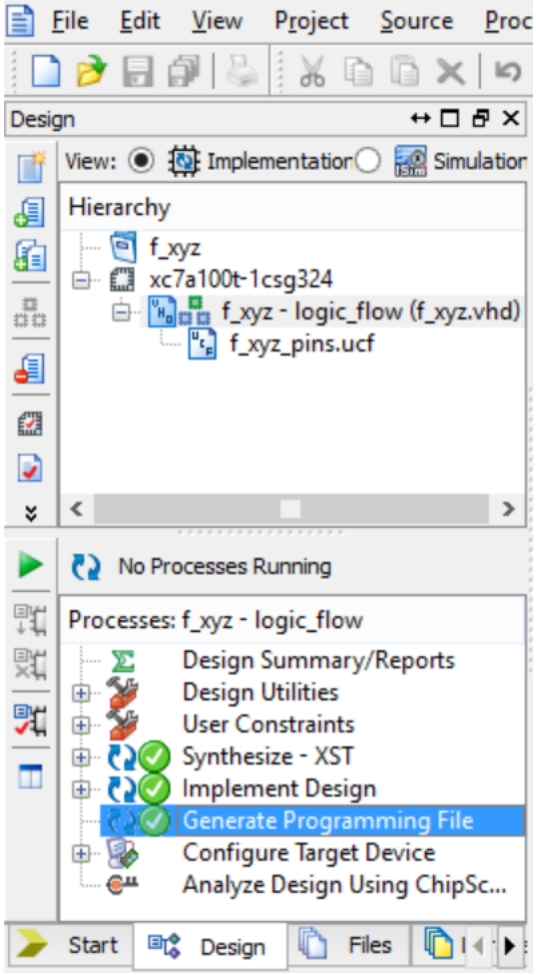


Figure 5-8

Step-7) In this step, turn is to upload our bit file into the FPGA. Connect your FPGA development kit to your computer. If it is the first time that your board is connected to your computer, be sure its USB driver is installed. Generally, operating systems install it automatically. Open *Manage Configuration Project (iMPACT)* from the *Configure Target Device* menu as given in Figure 5-9.

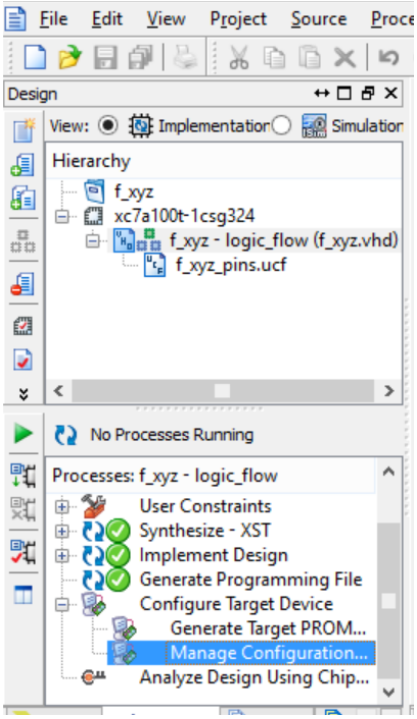


Figure 5-9

Step-8) An empty project screen will open as it seen in Figure 5-10. Double click *Boundary Scan* to add connected FPGA devices to configuration project.

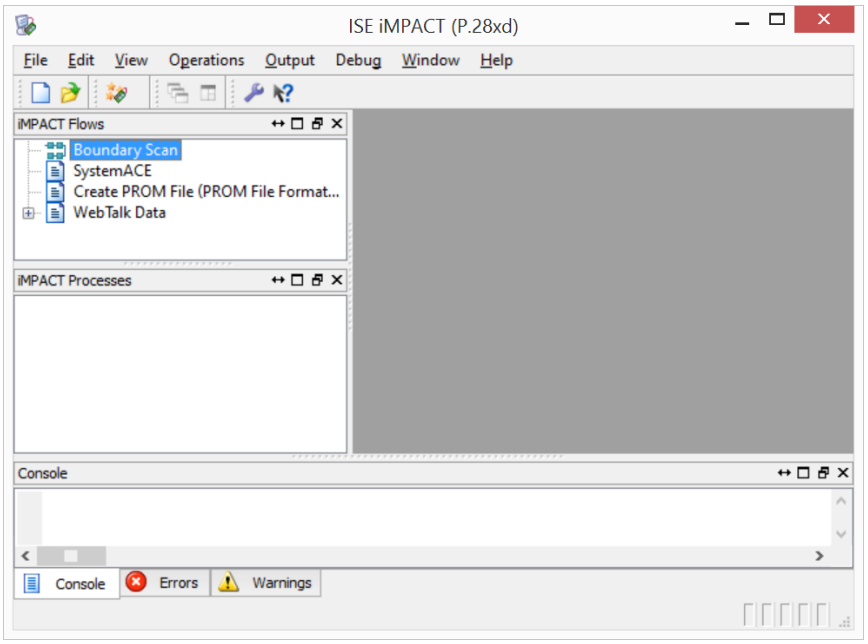


Figure 5-10

After *Boundary Scan* right side of the project window becomes ready to add FPGA chip.

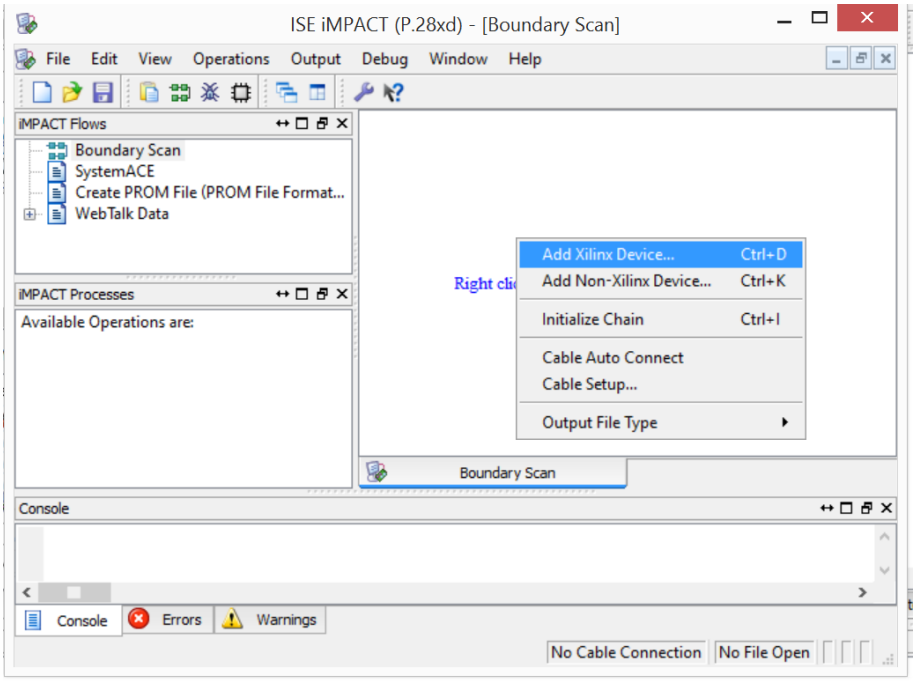


Figure 5-11

Step-9) Right click on the new-opened area and choose *Add Xilinx Device* (Figure 5-11). When clicked, window in the Figure 5-12 appears.

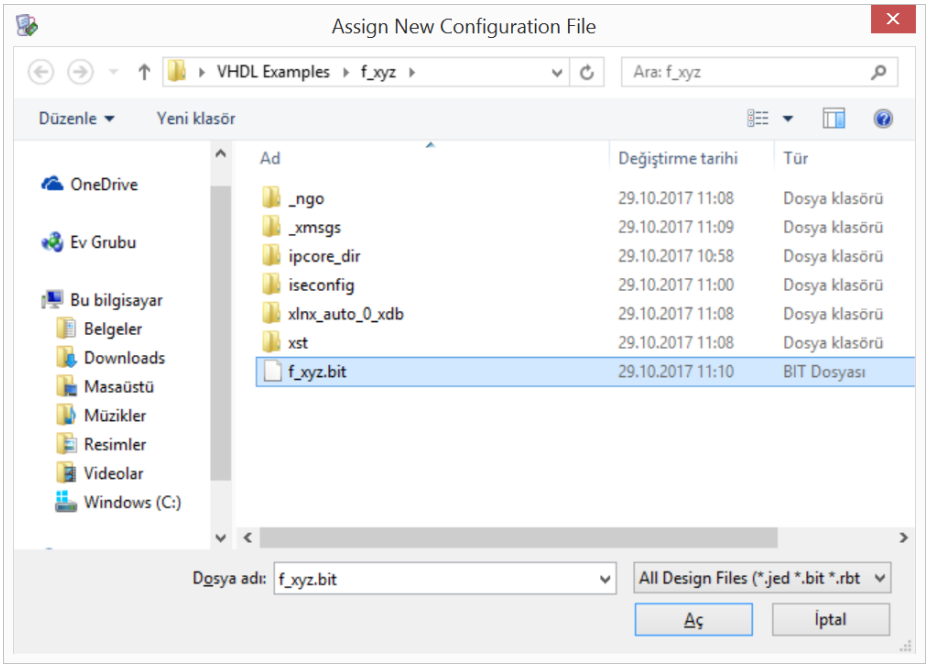


Figure 5-12

Choose the bit file that we created previously during the *Generate Programming File* process and click Open. When it's opened FPGA chip is found; check its name and you can see that it's XC7A100T as we set before.

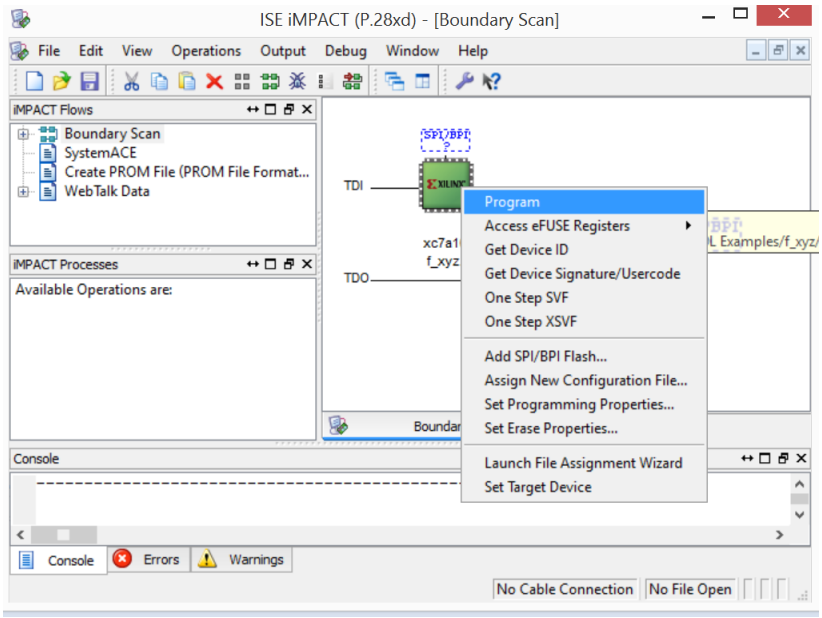


Figure 5-13

Step-10) Right click to the green FPGA chip and **Program** it (Figure 5-13). When it's done the indicator "Program Succeeded" should be seen, otherwise check your USB driver status.

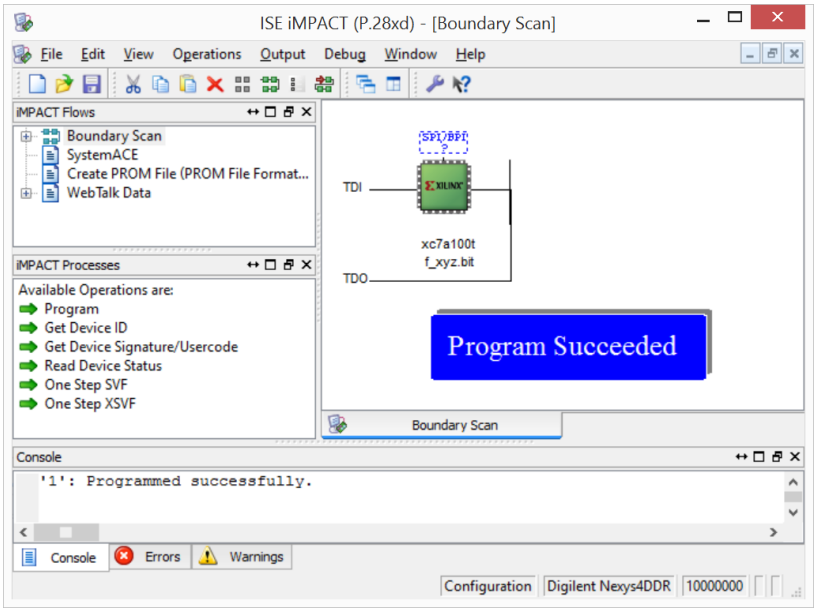


Figure 5-14

Step-11) This is the last step of the process. We successfully upload our code to FPGA. Now we should check the results whether they are true or not. In this case, we need to look at truth table of the given function at first. Truth table of the $f(x, y, z)$ given below in Table 5-1.

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

Table 5-1

Figure 5-15 shows the result of one possible combination of inputs, $x =' 1', y =' 0', z =' 1'$. As it's expected the result equals f equals logic '1' such that, LED is on. It's also important to state that sliding switches in + y direction means input is logic '1', sliding switches in –y direction makes any input logic '0'.

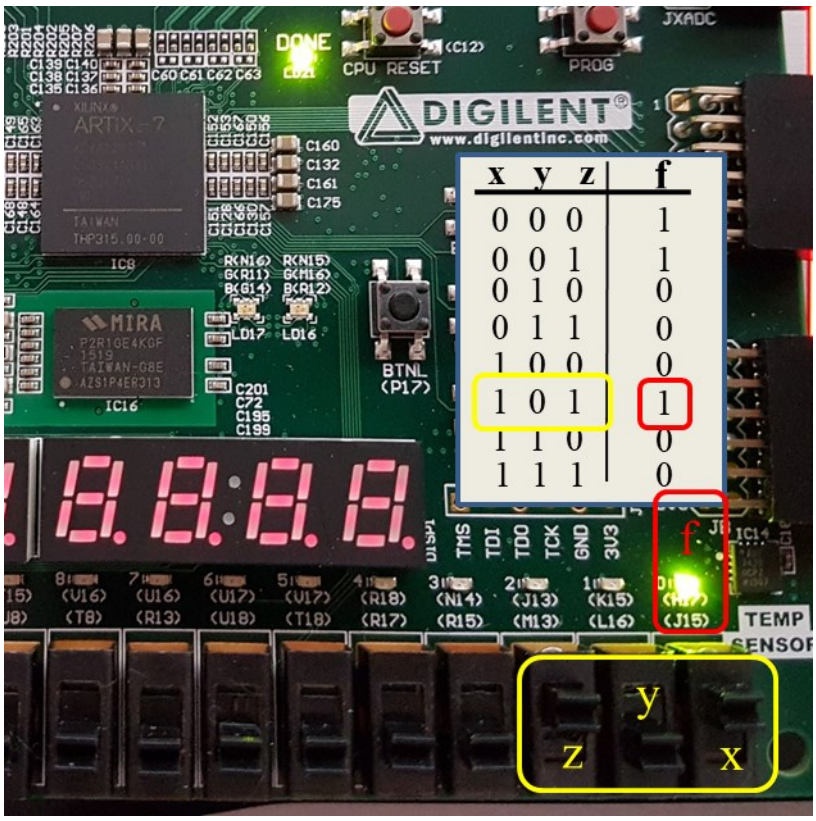
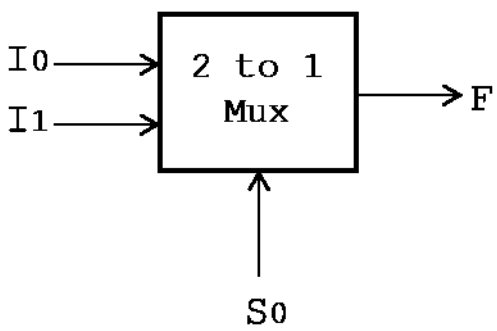


Figure 5-15

Example: Implement the 2-to-1 multiplexer which is depicted in Figure 5-16. Inputs and output are should be two bit wide. Only the select bit of the multiplexer is one bit. Create "*.bit" file, upload it to Nexys 4 DDR development kit and observe the results.



So	F
0	I0
1	I1

Figure 5-16

Solution: Below program segment PS 5.2 is the VHDL solution of this example.

```
library ieee;
use ieee.std_logic_1164.all;

entity multiplexer_2x1 is
  port(I0, I1: in std_logic_vector (1 downto 0);
        S0: in std_logic;
        F: out std_logic_vector(1 downto 0));
end entity;

architecture logic_flow of multiplexer_2x1 is

begin
        f<=x when s0='0' else
            y;
end architecture;
```

PS 5.2

Follow the steps (1-3) that are shown in the previous example. At this time, constraints file a bit different since, some of our inputs and our single output are two bits wide. Constraints should be written as shown in Figure 5-17.

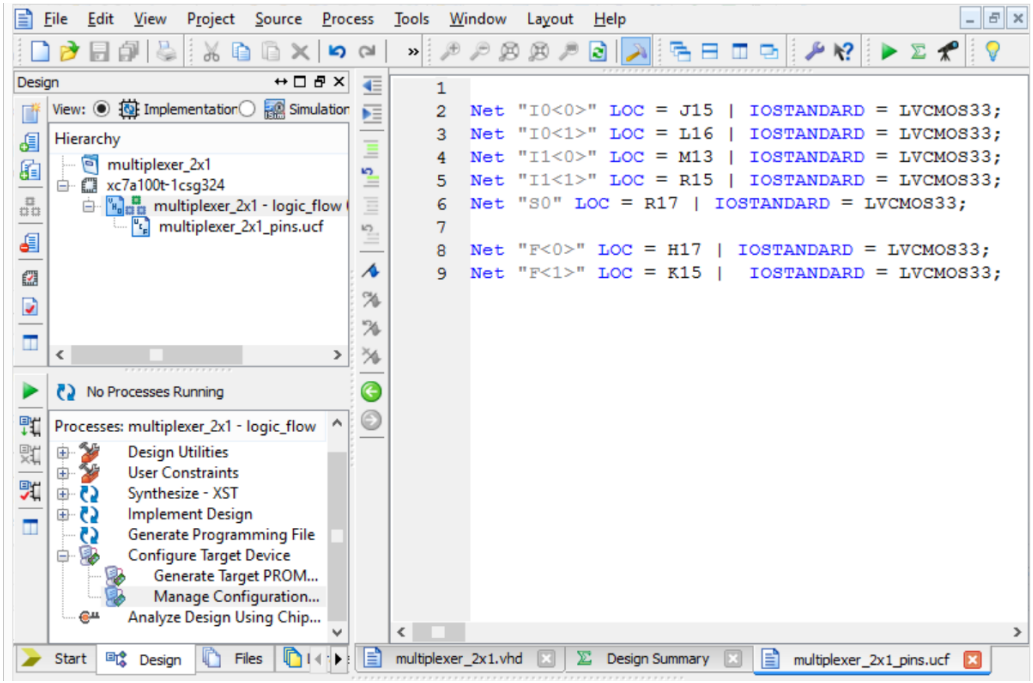


Figure 5-17

Constraints file is set, so step 4 and 5 are completed. When rest of the steps (6-11) are done, bit file of the multiplexer is created and uploaded into the development kit. In Figure 5-18, select bit of the multiplexer is set to '0' so, "01" is observed at the output which is equal to I_0.

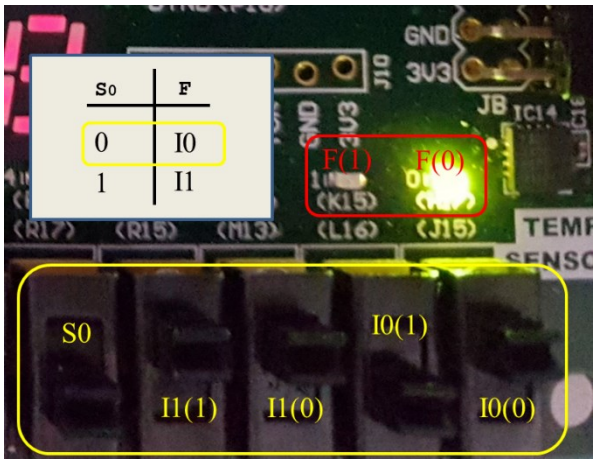


Figure 5-18

Figure 5-19 shows the results if S_0 becomes logic '1'. In this time, since I_1 is set as "11" output F equals "11". As a result, we covered all the states of the truth table.

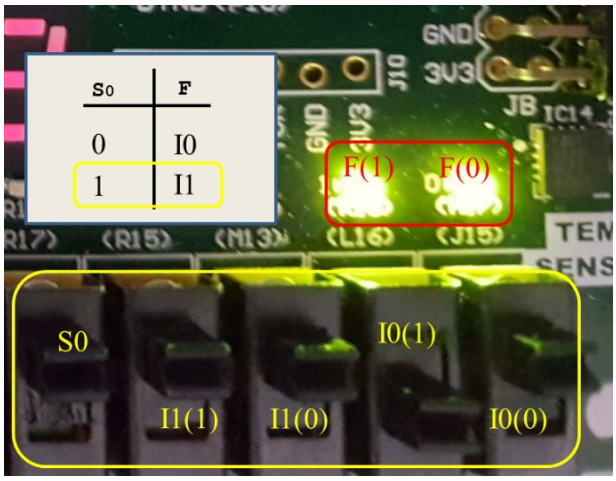
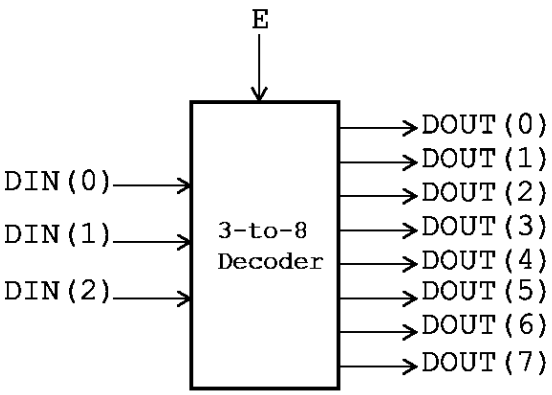


Figure 5-19

Exercise: 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 5-20