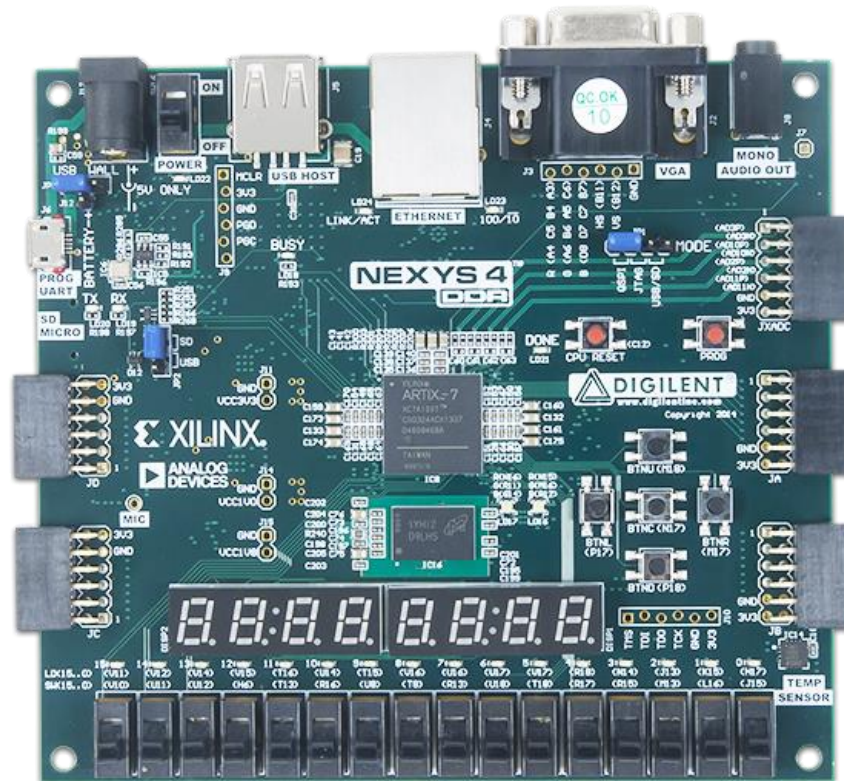


**CMPS 375**  
**Computer Architecture**  
**Demonstration: And2 (2-input And Gate)**  
**Introduction to the Nexys4 DDR Package (VIVADO 2016.4)**

## Objectives

The purpose of this demonstration is to learn the basics of the VIVADO software and the Xilinx Nexys4 DDR FPGA board as shown in Figure 1. After completing this tutorial, you will be able to:

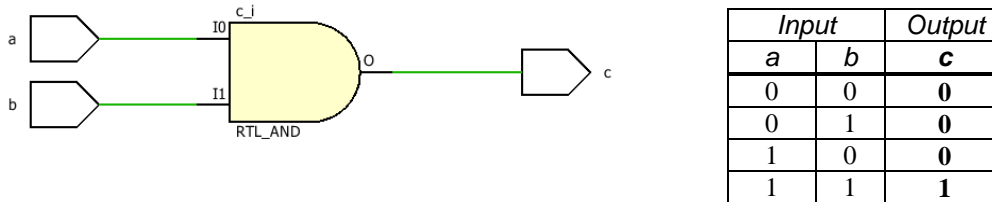
- Create a Vivado project sourcing VHDL models and targeting a specific FPGA device located on the Nexys4 DDR board
- Understand the design circuit writing in VHDL hardware description language and its schematic
- Use the provided user constraint file (XDC) to constrain pin locations
- Simulate the design using the XSIM simulator
- Synthesize and implement the design
- Generate the bitstream
- Program and download the design to the FPGA device on the Nexys4 DDR board
- Test and verify the functionality of the designed circuit on the Nexys4 DDR board



**Figure 1. Xilinx Nexys4 DDR Board: Artix-7 FPGA (XC7A100TCSG324-1)**

## Description

You learn how to create a 2-input And gate in VHDL (VHSIC Hardware Description Language) language like Figure 3.1 in textbook. Figure 2 shows the **truth table** and its **schematic** for And2 gate. You can perform a binary And operation using toggle switches as inputs and LEDs as outputs on the Nexys4 DDR board.



**Figure 2. AND2: A 2-input And Gate Like Figure 3.1 in Textbook**

## Files

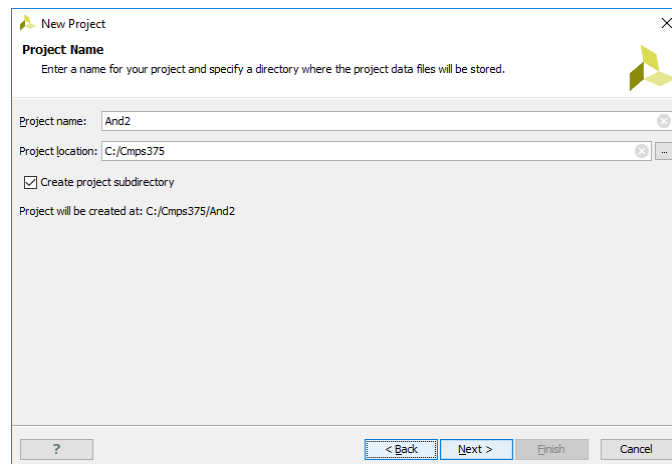
- And2Demo.zip consists of 4 files:
  1. Current document (CMPS375And2Demo.pdf)
  2. VHDL design file (And2.vhd)
  3. VHDL Test Bench simulation file (And2\_tb.vhd)
  4. I/O Constraints (Nexys4DDR\_Master.xdc)

Note: Perform binary And operation using toggle switches as the inputs and LEDs as the outputs on the Nexys4 DDR board.

Scalar Ports	Direction	Package Pin	Name
a	IN	L16	SW[1]
b	IN	J15	SW[0]
c	OUT	H17	LED[0]

## Demonstration: A 2-input And Gate in VHDL Language

1. Create a Vivado project: And2
  - a. Click on *File* menu and then *New Project* item to start the wizard.
  - b. In the *New Project* form as shown in Figure 3, enter **And2** in the *Project name* field and select **C:/Cmps375** in the *Project location*.
  - c. Make sure the *Create Project Subdirectory* box is checked. Click *Next*.



New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: And2

Project location: C:/Cmps375

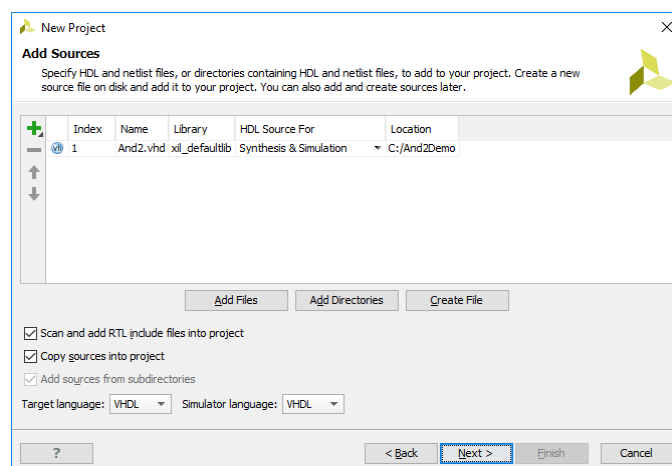
☒ Create project subdirectory

Project will be created at: C:/Cmps375/And2

< Back Next > Finish Cancel

Figure 3. Project Name and Location

- d. In the *Project Type* form, select the *RTL Project* option, and click *Next*.
    - e. In the *Add Sources* form as shown in Figure 4, click on the *Add File* button, then browse directory, select **And2.vhd**.
    - f. Verify the **Copy sources into project** box is checked. Select **VHDL** as the *Target Language* and as the *Simulator language*. Then, click *Next*.



New Project

Add Sources

Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

	Index	Name	Library	HDL Source For	Location
+	1	And2.vhd	xil_defaultlib	Synthesis & Simulation	C:/And2Demo

Add Files Add Directories Create File

☒ Scan and add RTL include files into project

☒ Copy sources into project

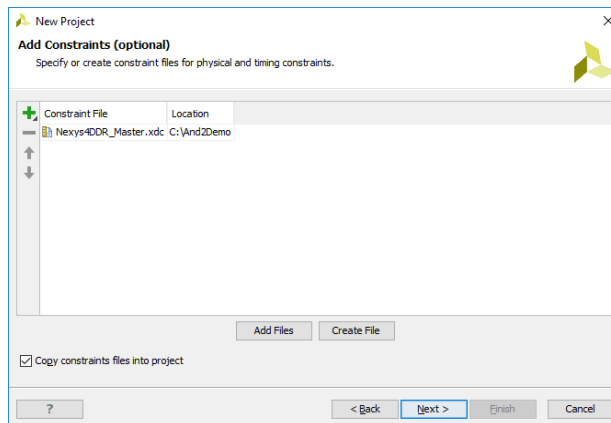
☒ Add sources from subdirectories

Target language: VHDL Simulator language: VHDL

< Back Next > Finish Cancel

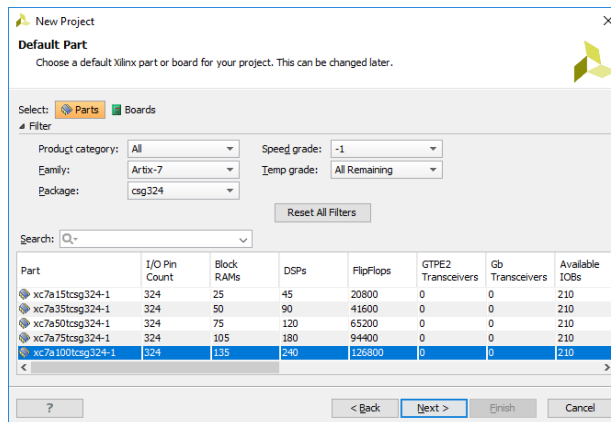
Figure 4. Add Source Form

- g. In the *Add Existing IP* form, click *Next*.
      - h. In the *Add Constraints* form as shown in Figure 5, click on the *Add File* button, then browse directory, select **Nexys4DDR\_Master.xdc**.
      - i. Verify the **Copy Constraints files into project** box is checked. Then, click *Next*.



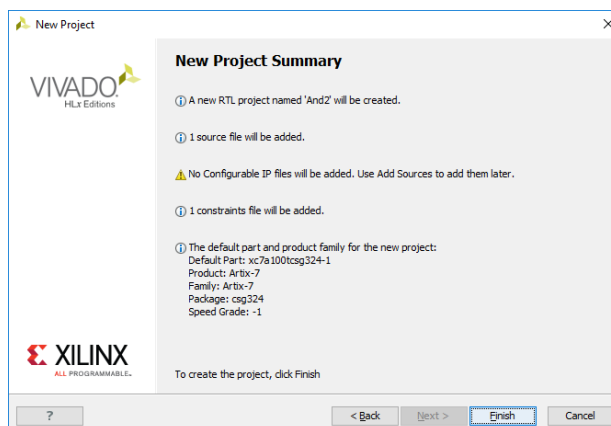
**Figure 5. Add Constraints Form**

- j. In the *Default Part* form as shown in Figure 6, select device Artix-7 FPGA **XC7A100TCSG324-1**. Then click *Next*.



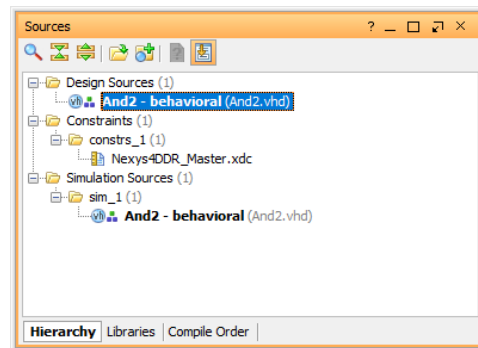
**Figure 6. Default Part Form**

- k. In the *New Project Summary* as shown in Figure 7, click *Finish* to create the project.



**Figure 7. New Project Summary Form**

2. Understand the design circuit writing in VHDL code and its schematic
  - a. In the *Sources* pane as shown in Figure 8, double-click the And2.vhd entry to open the file in the text mode. The file contents of **And2.vhd** are shown in Figure 9.



**Figure 8. Sources Pane**

```

-- Description:    2-input And Gate
-- Project:        And2
-- Program-ID:     And2.vhd
-- Author:         Kuo-pao Yang
-- Package:        Xilinx Nexys4 DDR Board
-- Device:         Artix-7 FPGA (XC7A100TCSG324-1)
-- Software:       Vivado Design Suite

```

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY And2 IS
    PORT ( a : IN STD_LOGIC;
          b : IN STD_LOGIC;
          c : OUT STD_LOGIC);
END And2;

ARCHITECTURE behavioral OF And2 IS
BEGIN

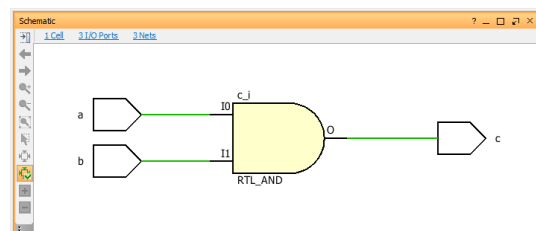
    c <= a AND b;

END behavioral;

```

**Figure 9. VHDL code for And2 Design Circuit: And2.vhd**

- b. In the *Flow Navigator* pane, click *Open Elaborated Design* entry under the *RTL Analysis* to perform RTL (Register-Transfer Level) analysis on the source file. Click the **Schematic** to see a logic view of the design as shown in Figure 10.



**Figure 10. Schematic of And2: A 2-input And Gate**

3. Use the provided user constraint file (XDC) to constrain pin locations
  - a. In the *Sources* pane, double-click Nexys4DDR\_Master.xdc entry under *Constraints* to open the file in the text mode. The file contents of **Nexys4DDR\_Master.xdc** shown in Figure 11 can be **edited** as the port names in the designed model, And2.vhd.

```

10
11 ##Switches
12
13 set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { b }]; #IO_L24N_T3_R50_15 Sch=sw[0]
14 set_property -dict { PACKAGE_PIN J16 IOSTANDARD LVCMOS33 } [get_ports { a }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
15 #set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { SW[2] }]; #IO_L26N_T0_D0R_VREF_14 Sch=sw[2]
16 #set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { SW[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
17 #set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
18 #set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L17N_T1_D10_14 Sch=sw[5]
19 #set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { SW[6] }]; #IO_L17N_T0_A13_D29_14 Sch=sw[6]
20 #set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { SW[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
21 #set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
22 #set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { SW[9] }]; #IO_L25_34 Sch=sw[9]
23 #set_property -dict { PACKAGE_PIN J16 IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
24 #set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports { SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
25 #set_property -dict { PACKAGE_PIN J16 IOSTANDARD LVCMOS33 } [get_ports { SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
26 #set_property -dict { PACKAGE_PIN J12 IOSTANDARD LVCMOS33 } [get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
27 #set_property -dict { PACKAGE_PIN J11 IOSTANDARD LVCMOS33 } [get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
28 #set_property -dict { PACKAGE_PIN J10 IOSTANDARD LVCMOS33 } [get_ports { SW[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]
29
30
31 ## LEDs
32
33 set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { c }]; #IO_L18P_T2_A24_15 Sch=led[0]
34 #set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { LED[1] }]; #IO_L24P_T3_R51_15 Sch=led[1]
35 #set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports { LED[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
36 #set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { LED[3] }]; #IO_L8P_T2_D11_14 Sch=led[3]
37 #set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
38 #set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { LED[5] }]; #IO_L18N_T2_A11_D07_14 Sch=led[5]
39 #set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { LED[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
40 #set_property -dict { PACKAGE_PIN J16 IOSTANDARD LVCMOS33 } [get_ports { LED[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
41 #set_property -dict { PACKAGE_PIN J16 IOSTANDARD LVCMOS33 } [get_ports { LED[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
42 #set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { LED[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
43 #set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
44 #set_property -dict { PACKAGE_PIN J16 IOSTANDARD LVCMOS33 } [get_ports { LED[11] }]; #IO_L15N_T2_DQS_DOUT_CS0_B_14 Sch=led[11]
45 #set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { LED[12] }]; #IO_L16P_T2_CS1_B_14 Sch=led[12]
46 #set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { LED[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
47 #set_property -dict { PACKAGE_PIN J12 IOSTANDARD LVCMOS33 } [get_ports { LED[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
48 #set_property -dict { PACKAGE_PIN J11 IOSTANDARD LVCMOS33 } [get_ports { LED[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]

```

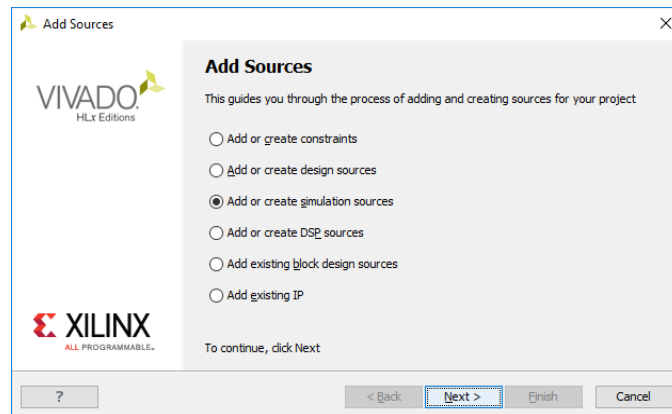
Figure 11. Editing I/O Constraints: **Nexys4DDR\_Master.xdc**

- b. Click on the *I/O Panning* under the *Layout* menu. Notice that once RTL analysis is performed, I/O Planning layout is available. In Figure 12, the design ports (a, b, and c) are listed in the *I/O Ports* tab of the *Console View* area.

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco
<b>Scalar ports (3)</b>							
a	IN		L16	✓	14	LVCMOS33*	3.300
b	IN		J15	✓	15	LVCMOS33*	3.300
c	OUT		H17	✓	15	LVCMOS33*	3.300

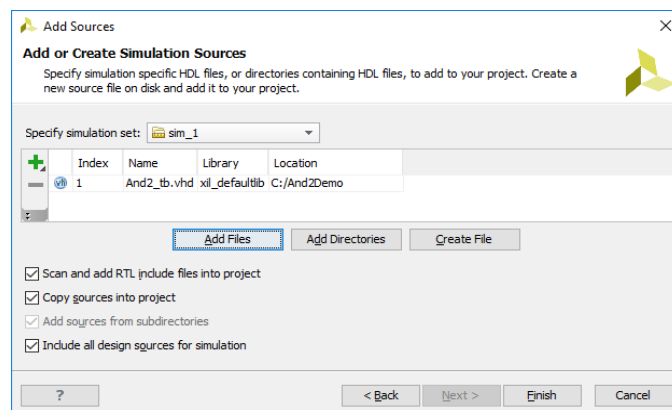
Figure 12. I/O Ports in Console View: a, b, and c

4. Simulate the design using the XSim Simulator
  - a. In the *Project Manager* tasks of the *Flow Navigator* pane, click the *Add Sources*, select the *Add or Create Simulation Sources* option as shown in Figure 13, and click *Next*.



**Figure 13. Selecting Simulation Sources option**

- b. In the *Add or Create Simulation Sources* form as shown in Figure 14, click on the *Add File* button, then browse directory, select **And2\_tb.vhd**.
    - c. Verify the *Copy sources into project* box is checked. Then, click *Finish*.



**Figure 14. Add Simulation Sources: And2\_tb.vhd**

- d. In the *Sources* pane, double-click the And2\_tb.vhd entry to open the file in the text mode. The file contents of **And2\_tb.vhd** are shown in Figure 15.

```
-----
-- Description:    2-input And Gate (Test Bench)
-- Project:       And2
-- Program-ID:    And2_tb.vhd
-- Author:       Kuo-pao Yang
-- Package:      Xilinx Nexys4 DDR Board
-- Device:       Artix-7 FPGA (XC7A100TCSG324-1)
-- Software:     Vivado Design Suite
-- Notes:      1. ENTITY: No code (empty)
--            2. ARCHITECTURE: Put simulation code under PROCESS
-----

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY and2_tb IS
END and2_tb;

ARCHITECTURE simulate OF and2_tb IS
    COMPONENT and2
        PORT ( a : IN STD_LOGIC;
              b : IN STD_LOGIC;
              c : OUT STD_LOGIC);
    END COMPONENT;
    SIGNAL a: STD_LOGIC;
    SIGNAL b: STD_LOGIC;
    SIGNAL c: STD_LOGIC;
BEGIN
    uut: and2 PORT MAP (a, b, c);

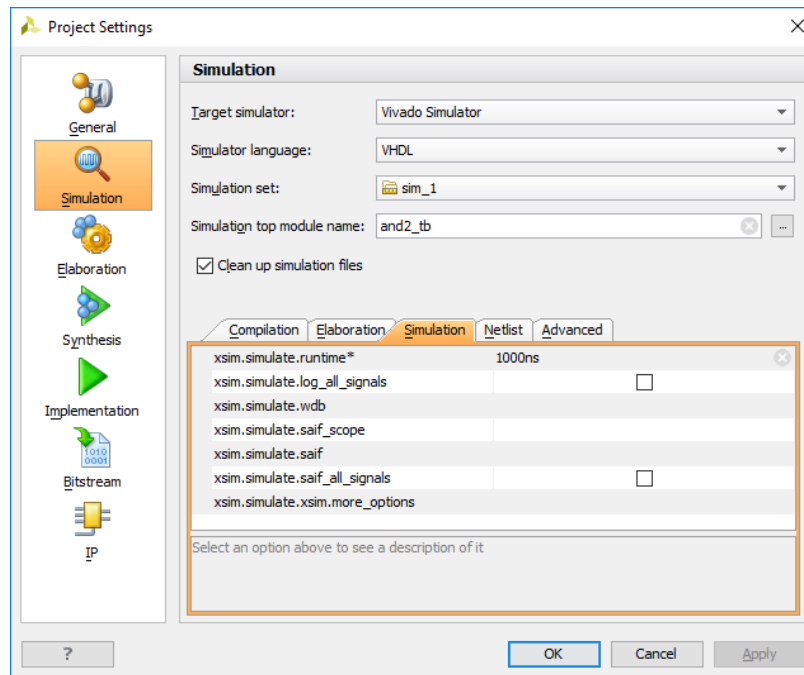
    stimulus: PROCESS
    BEGIN
        -- Put test bench stimulus code here
        a <= '0';
        b <= '0';
        WAIT FOR 100 ns;
        a <= '0';
        b <= '1';
        WAIT FOR 100 ns;
        a <= '1';
        b <= '0';
        WAIT FOR 100 ns;
        a <= '1';
        b <= '1';
        WAIT FOR 100 ns;
        a <= '0';
        b <= '0';

        WAIT;
    END PROCESS;
END simulate;
```

**Figure 15. VHDL code for And2 Test Bench Simulation File: [And2\\_tb.vhd](#)**

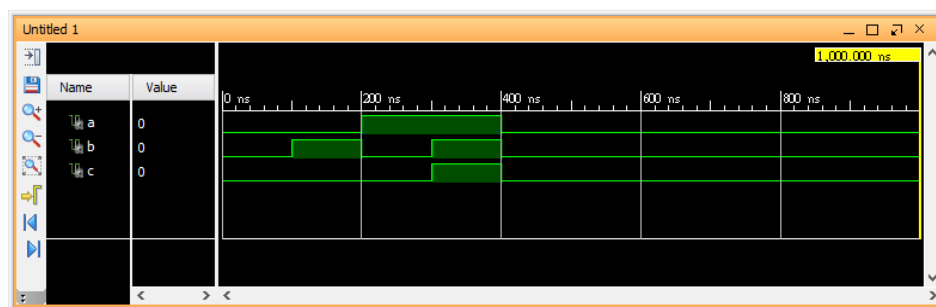


- e. Simulate the design for 1000 ns using the XSIM simulator. Click on *Simulation settings* of *Flow Navigator* pane, select the *Simulation* tab, and change the simulation time to **1000 ns** as shown in Figure 16.



**Figure 16. Simulate the Design for 1000 ns**

- f. Click on the *Run Simulation* under the *Flow Navigator* pane. Select the **Run Behavioral Simulation** option. The simulator results are similar to Figure 17.



**Figure 17. Simulation Results: And2\_tb.vhd**

5. Synthesize and implement the design

- a. Click on the *Run Synthesis* under the *Synthesis* tasks of the *Flow Navigator* pane.

Note that: Vivado synthesis is the process of transforming an RTL-specified design into a gate-level representation. This synthesis is **timing-driven** and optimized for memory usage and performance.

- b. Click on the *Run Implementation* under the *Implementation* tasks of the *Flow Navigator* pane.

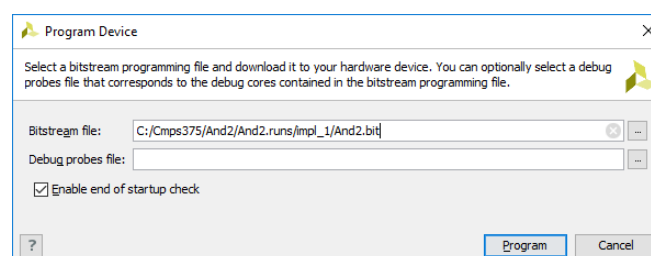
Note that: Vivado implementation includes all steps necessary to place and route the netlist onto **device** resources, within the logical, physical, and timing constraints of the design.

6. Generate the bitstream

- a. Click on the *Generate Bitstream* entry under the *Program and Debug* tasks of the *Flow Navigator* pane.

7. Program and download the design to the FPGA device

- a. Make sure the Micro-USB cable is connected between the board and the PC. Power *ON* the switch on the board. Note that it does **not** need to connect the power jack and the board.
- b. Click on the *Open Target* of the *Hardware Manager* entry under the *Program and Debug* tasks of the *Flow Navigator* pane, and then select the *Auto Connect*.
- c. Click on the *Program Device* of the *Hardware Manager* entry under the *Program and Debug* tasks of the *Flow Navigator* pane, and then select **xc7a100t\_1**.
- d. Make sure the *Bitstream file* of *Program Device* form is **And2.bit** as shown in Figure 18. Then, click **Program** to program the FPGA. Note that: the *DONE* light on the board will light when the device is programmed.



**Figure 18. Bitstream for Nexys4 DDR: And2.bit**

8. Test and verify functionality of the designed circuit

- a. Verify the functionality by flipping switches and observing the output on the LEDs.

<i>Scalar Ports</i>	<i>Direction</i>	<i>Package Pin</i>	<i>Name</i>
a	IN	L16	SW[1]
b	IN	J15	SW[0]
c	OUT	H17	LED[0]