EXPERIMENT 5

Use of XOR/XNOR Gates to Generate & Check Parity

Department of Electrical & Computer Engineering

I. OBJECTIVES:

- Demonstrate the use of XOR and XNOR gates to generate and check parity.

II. MATERIALS:

- Xilinx Vivado software, student or professional edition V2018.2 or higher.
- IBM or compatible computer with Pentium III or higher, 128 M-byte RAM or more, and 8 G-byte Or larger hard drive.
- BASYS 3 Board.

III. DISCUSSION:

A parity checker circuit is used to detect a 1-bit error, as could occur in data transmission. Such an error can be caused by an electrical noise "hit", or by a hardware failure such as a bit "stuck at 0" or "stuck at 1". Parity can be even or odd and requires that an extra bit (the parity bit) be generated and "tacked onto" the data. The value of the parity bit is derived from the data. The following examples illustrate even and odd parity.

1) Even parity

Determine the parity bit for the 4 bit data 1001 using even parity.

The number of 1 bit in 1001 is 2, an even number. To have even parity for this data, we must keep the total number of 1s, including the parity bit, even. Hence, the parity bit should be a 0. Note that, as a binary number, 1001 is odd (value is 9). But whether the value of 1001 is odd or even is not important. Only the number of 1s is important.

Data: 1001 Parity bit: 0 Data with parity: 10010

2) Odd Parity

Determine the parity bit for the 8 bit data 10110111.

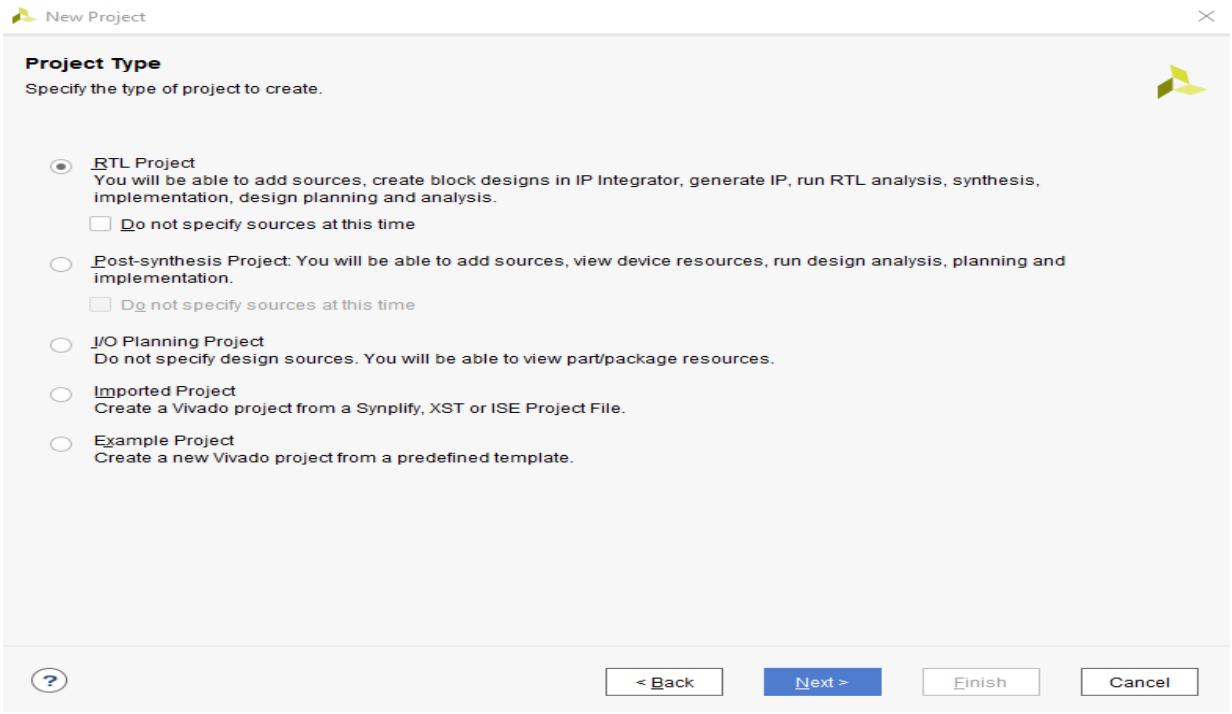
The number of 1 bit in 10110111 is 6, an even number. To have even parity for this data, we must make sure the total number of 1s, including the parity bit, is an odd number. Therefore, the parity bit must be 1. Again, the numerical value of the data is not important, all that counts is the number of 1s in the data.

Data: 10110111 Parity bit: 1 Data with parity: 101101111

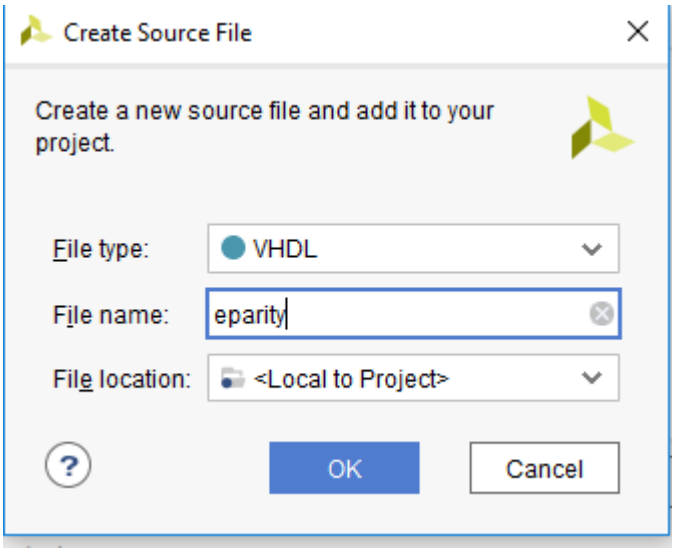
IV. PROCEDURE:

1. Open Xilinix Vivado and in the **Xilinx-Project Navigator** window, Quick start, **New Project**.

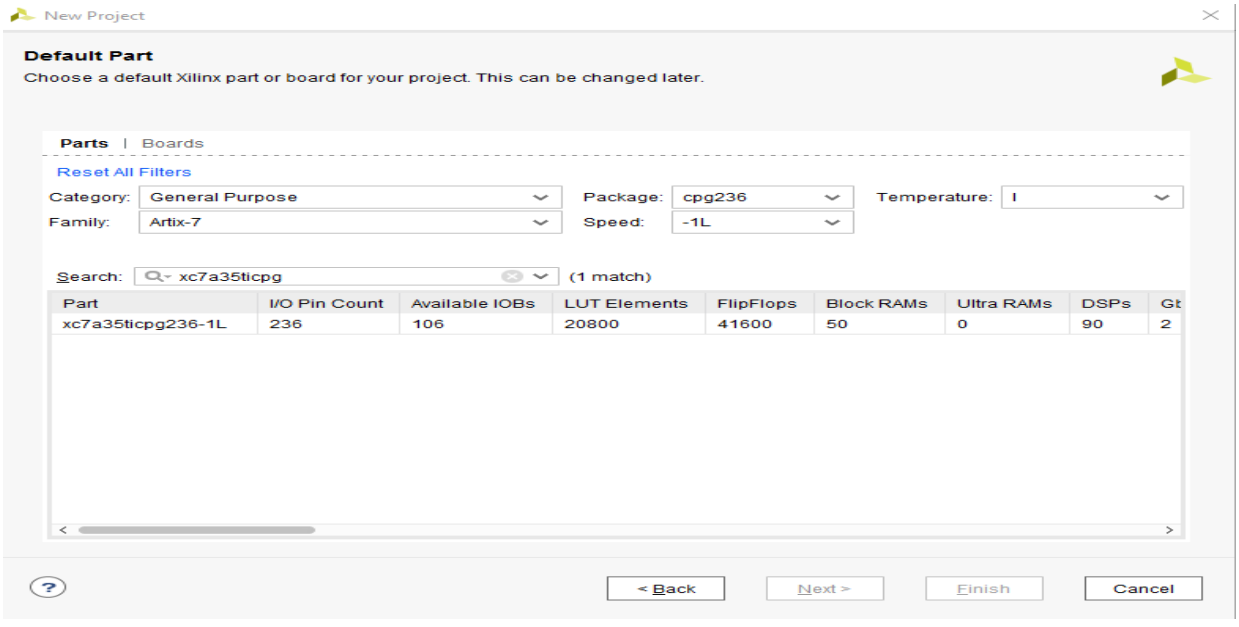
2. Choose "RTL Project" and check the "Do not specify sources at this time" as we will configure all the settings manually through the navigator from inside the project.



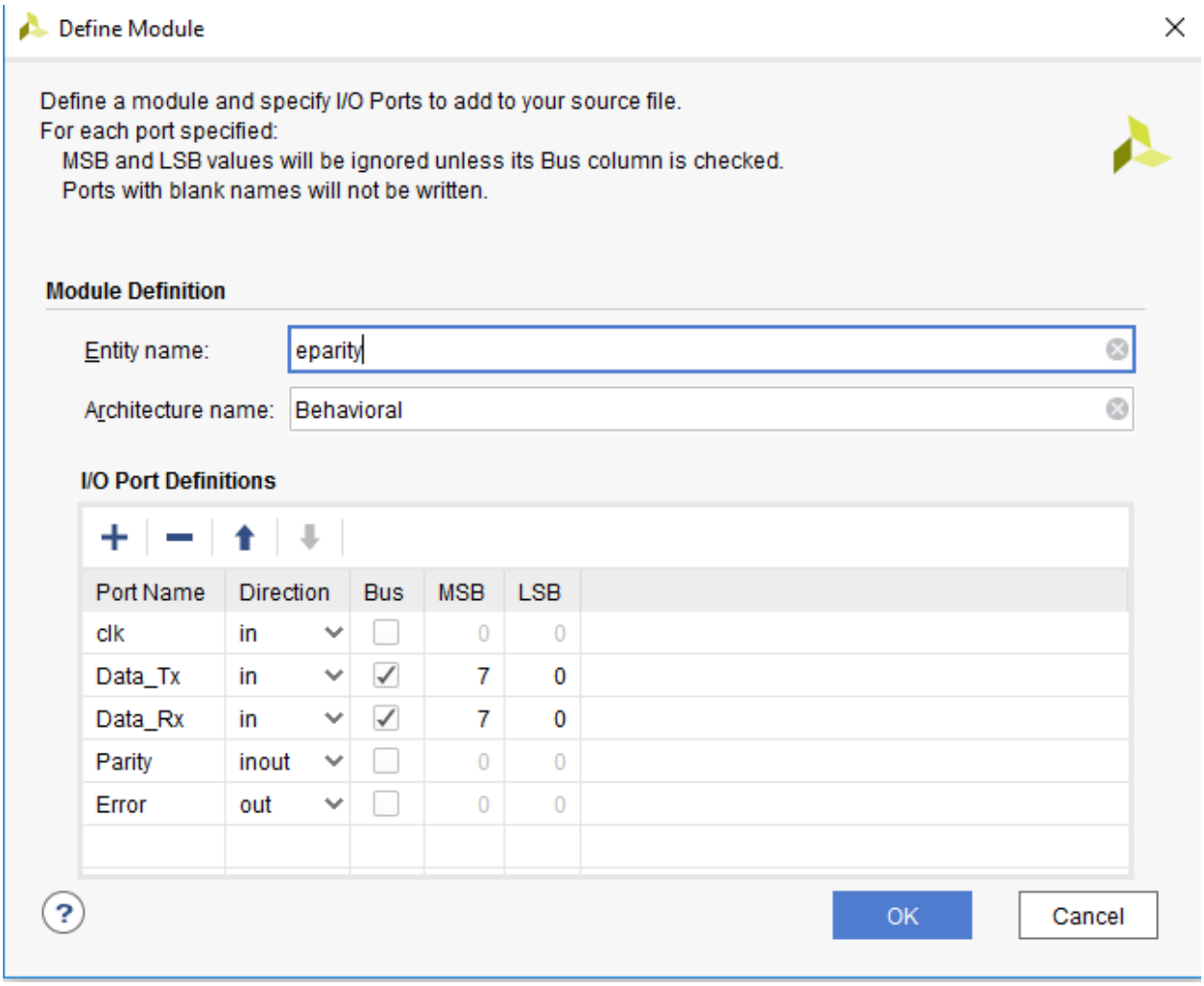
3. Select **New Source…** and the **New** window appears. In the New window, choose Schematic, type your file name (such as ***eparity***) in the File Name editor box, click on OK, and then click on the Next button.



4. In the **Xilinx - Project Navigator** window, select the following
   - Category: "General Purpose"
   - Family: "Artix-7"
   - Package: "cpg236"
   - Speed: "-1"
   - Choose "xc7a35tcpg236-1" that corresponds to the board we are using.
   - Then Choose Finish.



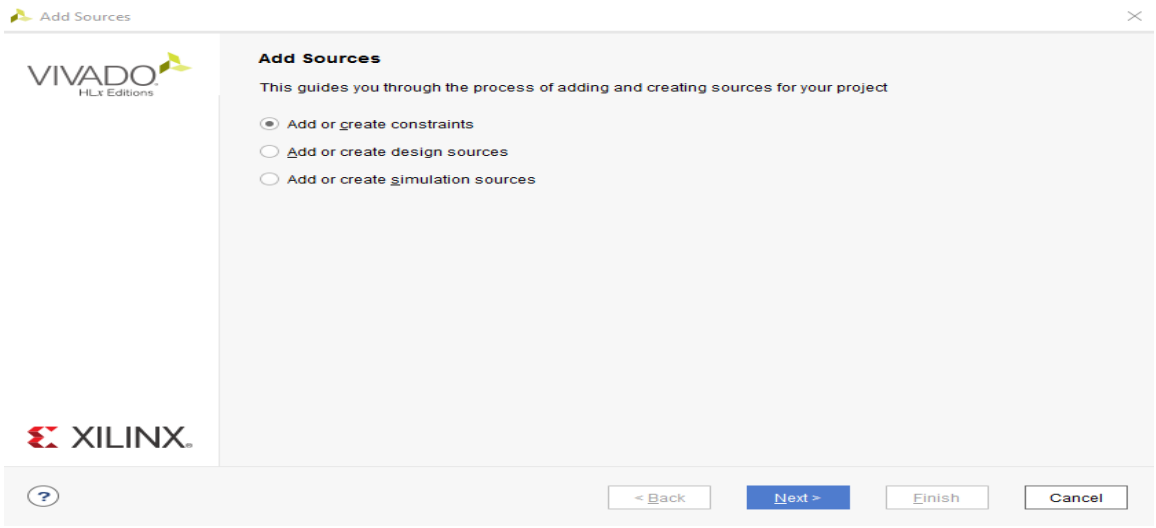
5. The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment. In this experiment, we have a bit victor as an input data. Then Under “Port Name”, add “clk” as input and then add “Data_Tx” and “Data_Rx” then check the “Bus” and make the “MSB” equal to 7 as we are designing an 8-bit parity check. Then, add “Parity”, as in/output as it will be output from the transmitter stage then input to the Rx stage. Alsi, add “Error” as output and select OK.



6. In the “eparity.vhd” created file, type the gates equivalent VHDL code for the 8-bit even parity transmitter/ receiver between the “begin” and “end Behavioral” as follows and then save the file.

```
46      --end loop l_parity ;
47      --Parity <= temp ;
48  --end Behavioral;
49  use IEEE.numeric_std.all;
50  entity eparity is
51  port(
52    clk             : in std_logic;
53    Data_Tx            : in  std_logic_vector(7 downto 0);
54    Data_Rx         : in  std_logic_vector(7 downto 0);
55    Parity          : inout std_logic;
56    Error          : out std_logic);
57  end eparity;
58  architecture rtl of eparity is
59  signal temp_data            : std_logic_vector(7 downto 0);
60  signal temp_rx           : std_logic_vector(8 downto 0);
61  begin
62  p_parity_check : process (clk)
63  variable vparity           : std_logic;
64  variable rxparity           : std_logic;
65  begin
66       if rising_edge(clk) then
67       -- Tx
68      temp_data  <= Data_Tx;
69      vparity := '0';
70      rxparity := '0';
71      l_parity : for k in 0 to 7 loop
72        vparity := vparity xor temp_data(k);
73      end loop l_parity;

74    ------- Rx
75    parity <= vparity;
76      temp_rx <= Data_Rx & Parity;
77      Rx_parity : for k in 0 to 8 loop
78            rxparity := rxparity xor temp_rx(k);
79          end loop Rx_parity;
80      end if;
81      Error <= rxparity;
82  end process p_parity_check;
83  end rtl;
```

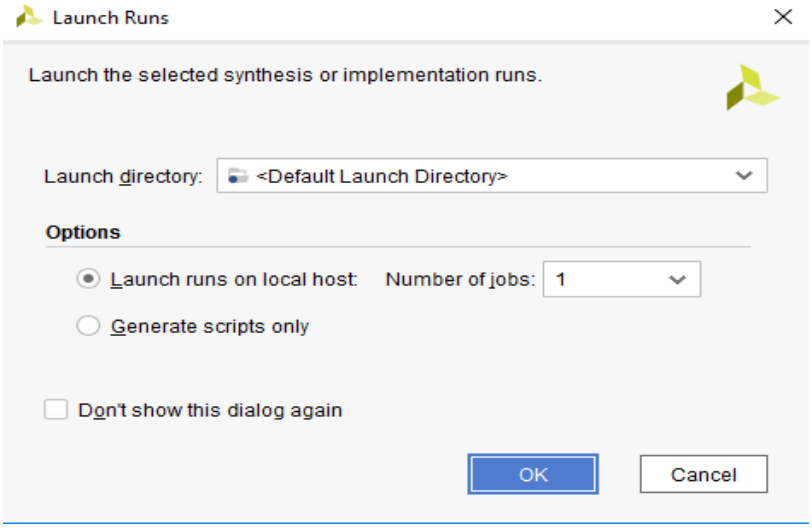
7. Next, we need to add To add a constraint file with the".xdc" extension, as following: Go to "Flow Navigator" and from "Project Manager" select "Add Sources" then "Add or create constraints". Next, choose "Create File" and enter the file name "lab_2" then "OK" followed by "Finish".



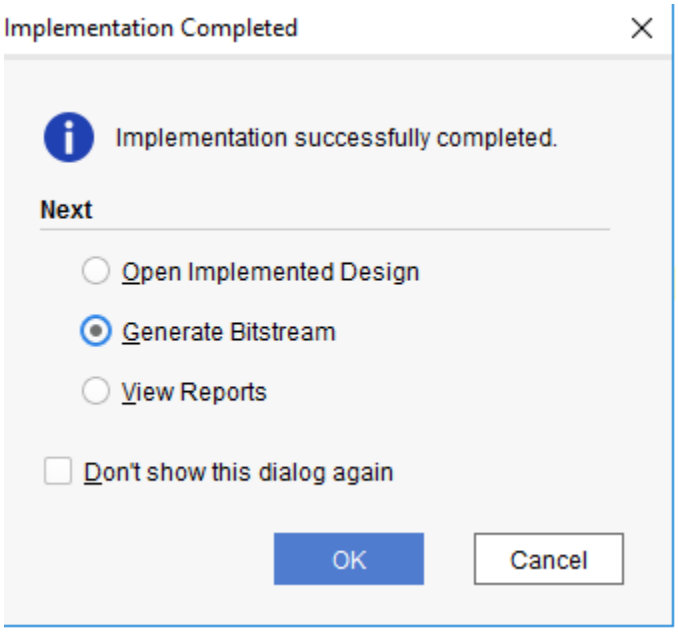
8. Then, we need to get a template xdc file that is going to be edited according to the different experiments. Google "basys 3 xdc file" and choose the "xilinix" link that appears (https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2015x/Basys3/Supporting%20Material/Basys3_Master.xdc). Copy the whole file and paste it into the "port_assign.xdc" that you have just created in the last step. Then uncomment and edit the input Switches and the output LEDs as in the next step.

```
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports {Data_Tx[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[0]}]
set_property PACKAGE_PIN V16 [get_ports {Data_Tx[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[1]}]
set_property PACKAGE_PIN W16 [get_ports {Data_Tx[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[2]}]
set_property PACKAGE_PIN W17 [get_ports {Data_Tx[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[3]}]
set_property PACKAGE_PIN W15 [get_ports {Data_Tx[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[4]}]
set_property PACKAGE_PIN V15 [get_ports {Data_Tx[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[5]}]
set_property PACKAGE_PIN W14 [get_ports {Data_Tx[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[6]}]
set_property PACKAGE_PIN W13 [get_ports {Data_Tx[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Tx[7]}]
set_property PACKAGE_PIN V2 [get_ports {Data_Rx[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[0]}]
set_property PACKAGE_PIN T3 [get_ports {Data_Rx[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[1]}]
set_property PACKAGE_PIN T2 [get_ports {Data_Rx[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[2]}]
set_property PACKAGE_PIN R3 [get_ports {Data_Rx[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[3]}]
set_property PACKAGE_PIN W2 [get_ports {Data_Rx[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[4]}]
set_property PACKAGE_PIN U1 [get_ports {Data_Rx[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[5]}]
set_property PACKAGE_PIN T1 [get_ports {Data_Rx[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[6]}]
set_property PACKAGE_PIN R2 [get_ports {Data_Rx[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Data_Rx[7]}]


## LEDs
set_property PACKAGE_PIN U16 [get_ports {Parity}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Parity}]
set_property PACKAGE_PIN E19 [get_ports {Error}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Error}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
```

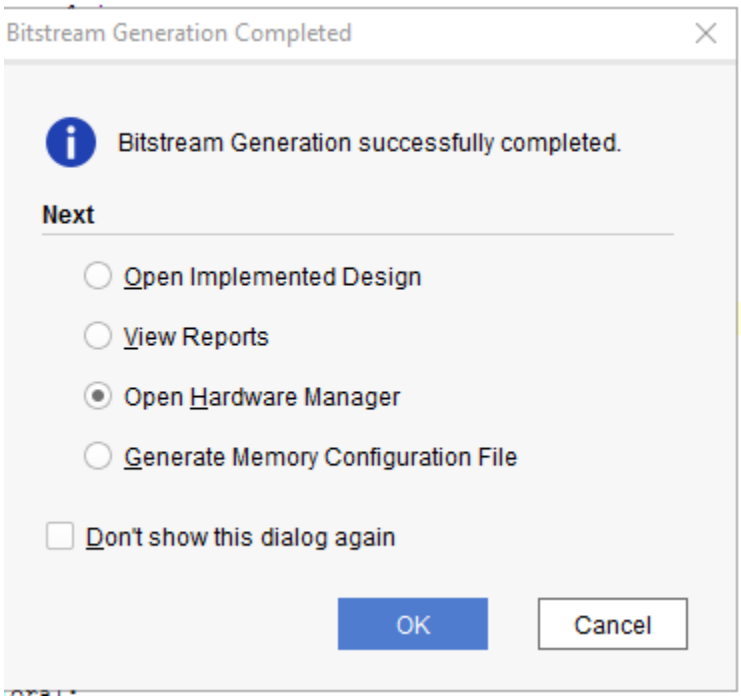
9. From the tool tab choose the play button ▶ and then “Run Implementation”. Select ”Number of jobs” =1 and then press OK.



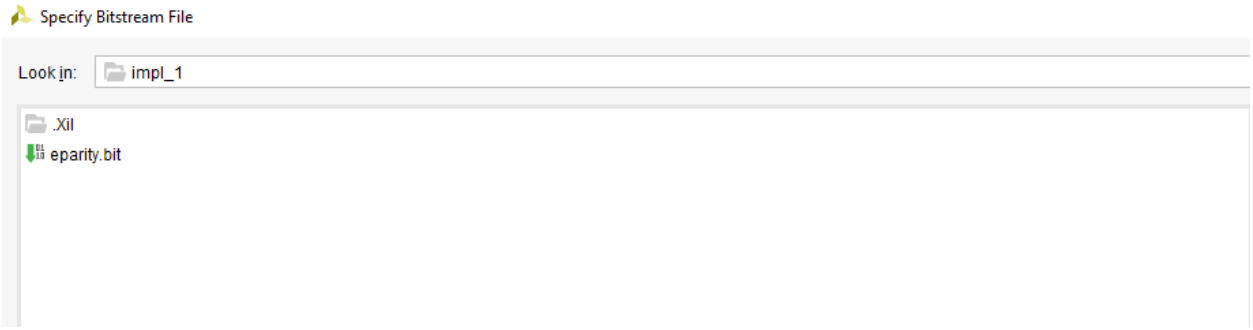
10. The implementation errors window will appear if any or the successfully completed window. From this window select “Generate Bitstream” and then OK. This will make the software generate “.bin” file to be used in programing the hardware BAYAS 3.



11. The next window will appear in which choose “Open Hardware Manger”, connect the Hardware Kit to the USB port and then press OK.



12. From the window appears, select the ".bin" file from the Project you create by browsing for the generated ".bit file" under the ".runs" folder and program the board then press OK.



13. Check the Parity and the Error pattern on your board in the following cases,

Data_Tx	Parity	Data_Rx		Error	Comments
“0001 0101”		“0001 0101”	No change		
“0001 0101”		“0001 0100”	1-bit change		
“0001 0101”		“0001 0110”	2-bit change		
“0001 0101”		“1101 0100”	3-bit change		
“0001 0101”		“1110 0100”	4-bit change		

14. List your comments from last step

Checked by_____ Date _____

V. Questions:

1. Draw the logic circuit for each of the following. For each gate, determine if it generates either EVEN or ODD parity bit and finds the output for the given input data:

(Remember: A XOR generates EVEN parity bit. A XNOR generates Odd Parity bit, whatever how many inputs they have.)

Data Inputs	**Which kind parity bit can it generate?**	
	EVEN Parity Bit	ODD Parity Bit
4-input XOR, input data=1001		
5-input XOR, input data=1001 0		
6-input XOR, input data=101001		
7-input XNOR, input data = 1011011		

2. We discussed how to construct an 8-bit even parity generator/checker using XOR gates, can you use XNOR gates to perform the same function? Draw the schematic diagram.

3. If you are asked to build an 8 bit odd parity generator/ checker with the output ERROR=1 when there is a 1 bit error, which type of gates, XOR or XNOR is simpler to use? Explain.

4. Why can’t a parity generator/checker detect even-bit errors? Given an example in your explanation.