EXPERIMENT 7

Decoders & Applications

OBJECTIVES:

- Examine the function of a decoder.
- Design and test a 2-to-4 decoder with active-low outputs using VHDL/HDL.
- Design and simulate a BCD decoder design with active-high outputs in HDL.

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.

DISCUSSION:

So far, we have used Xilinx's schematic design to create combinational circuits. But the Xilinx ISE also contains Hardware Description Language (HDLs) as an alternative method for creating combinational circuits. Xilinx foundation supports three types of HDL: ABEL, VHDL and verilog. We will use VHDL in this experiment. VHDL is an industry standard HDL, and can be used to model a digital system at many levels of abstraction ranging from the algorithm level to the gate level. VHDL code files have a sub-field with the suffix vhd. In VHDL design, the Netlist is extracted from the .vhd file whereas, in a schematic design, the Netlist is extracted from the .sch file.

In the Xilinx schematic design tool, symbols for such things as a 2-to-4 decoder or a BCD decoder can be obtained directly from a symbol library. In this experiment we will use VHDL to build a decoder from basic logic operations. One of the purposes of this experiment is to demonstrate how to use an HDL tool to design combinational circuits. We will also see how to design decoders.

We will start with a simple 2-to-4 decoder. We will show the steps for entering VHDL code for the decoder and then synthesizing the Netlist from the VHDL code. We also will show how to make pin selections using VHDL. The rest of the steps (such as implementation, simulation, and programming a target board) are exactly the same as in previous experiments. We will then design a BCD decoder using VHDL and run it in simulation.

The 2-to-4 decoder

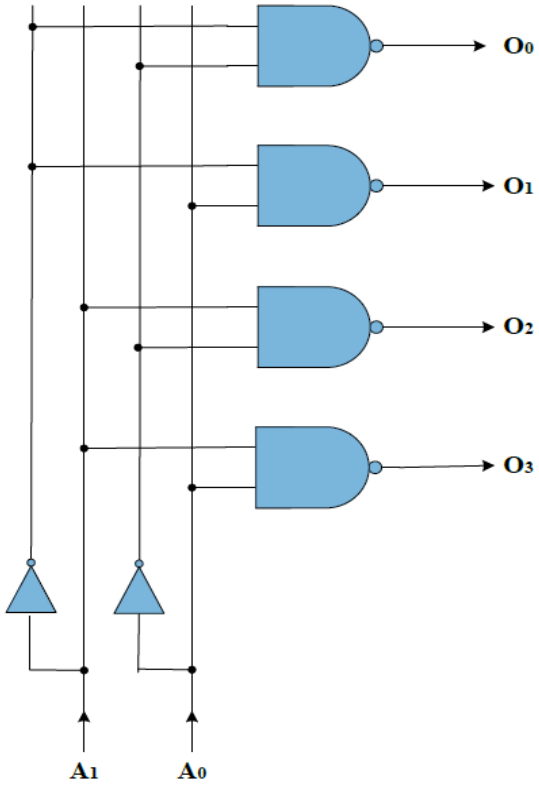
Decoders are used often in computer and communication circuits. A decoder activates one of its outputs depending on the binary number present on its input. If the number of data inputs is N, the maximum number of outputs for a decoder is 2N since there are 2N possible combinations of inputs. For a full decoder, there is an output for each combination of inputs. For example, a 2-to-4 decoder has 2 data inputs and hence 22 =4 outputs. A partial-decoder does not use all possible input combinations, and so has fewer outputs. The truth table shown below describes the functionality of the 2-to-4 decoder with active low outputs.

The Truth Table for a 2-to-4 Decoder with Active-Low Outputs

Input		Output			
A	**B**	**O_0**	**O_1**	**O_2**	**O_3**
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

If we examine the truth table closely, we find that there is only one 0 in each output column. So there is no need to use Boolean algebra or a K-map; we can get the equations by just seeing what combination of inputs produces the 0 in each column. The equations and schematics are:

2-to-4 Decoder with Active Low Output



$O'_0 = A'B'$

$O'_1 = A'B$

$O'_2 = AB'$

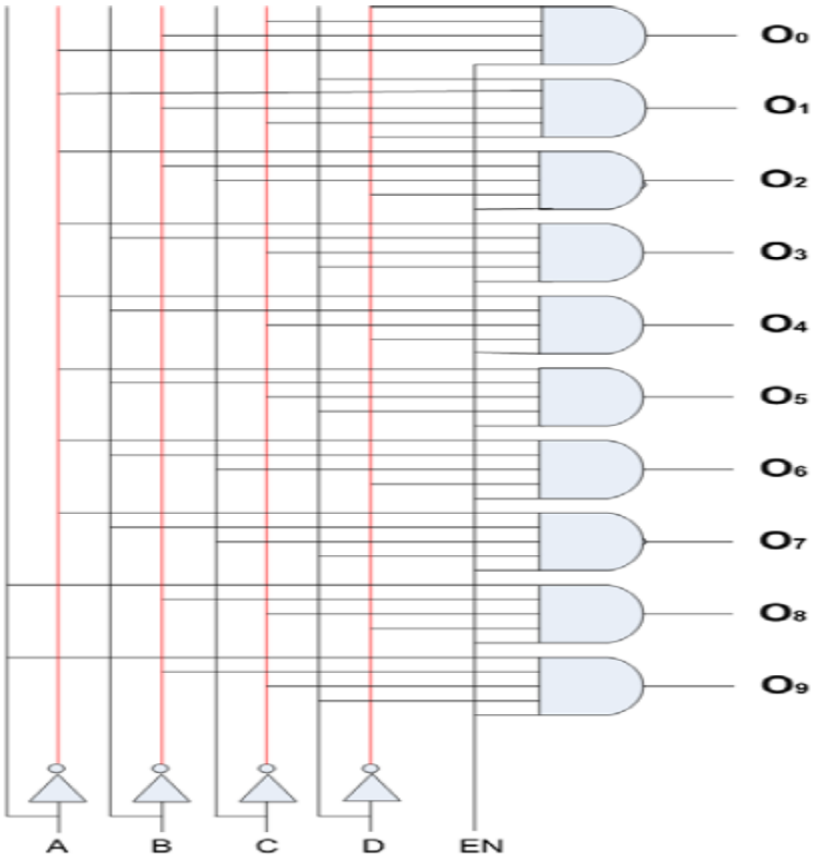
$O'_3 = AB$

The BCD decoder

Some decoders do not utilize all the 2N possible input codes but only certain ones. An example is the decoder used in converting a BCD number to a decimal one. The following truth table describes the function of the BCD decoder. The inputs, outputs, and enable (EN) are all active high.

Truth Table for Binary-coded Decimal Decoder with Active-High Outputs

Decoder Inputs					Decimal	Decoder Outputs									
EN	**A**	**B**	**C**	**D**		O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9
1	0	0	0	0	**0**	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	**1**	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	**2**	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	**3**	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	**4**	0	0	0	0	1	0	0	0	0	0
1	0	1	0	1	**5**	0	0	0	0	0	1	0	0	0	0
1	0	1	1	0	**6**	0	0	0	0	0	0	1	0	0	0
1	0	1	1	1	**7**	0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	**8**	0	0	0	0	0	0	0	0	1	0
1	1	0	0	1	**9**	0	0	0	0	0	0	0	0	0	1
1	1	0	1	0	inv	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	inv	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	inv	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	inv	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	inv	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	inv	0	0	0	0	0	0	0	0	0	0
0	X	X	X	X		0	0	0	0	0	0	0	0	0	0



As shown in the above truth table when the enable is active (EN=1) one of the ten outputs can be selected; but when EN=0, no output will be activated regardless of the input ('X' here means don't care). For the invalid BCD inputs 1010 through 1111 and designated inv), no output will be selected. The Boolean equations can be derived the same way as we did previously for the 2-to-4 decoder. There are ten Boolean equations for the ten outputs:

$$O_0 = EN \bullet (\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}), \quad O_1 = EN \bullet (\bar{A}\,\bar{B}\,\bar{C}\,D), \ \ldots, \quad O_9 = EN \bullet (A\,B\,C\,D)$$

The schematic diagram can be drawn easily once we have the Boolean equations. We can use four NOT gates and ten 5-input AND gates to implement the ten equations. The schematic diagram is left for you to draw in the questions section.

PROCEDURE:

Section 1 Design a 2-to-4 decoder:

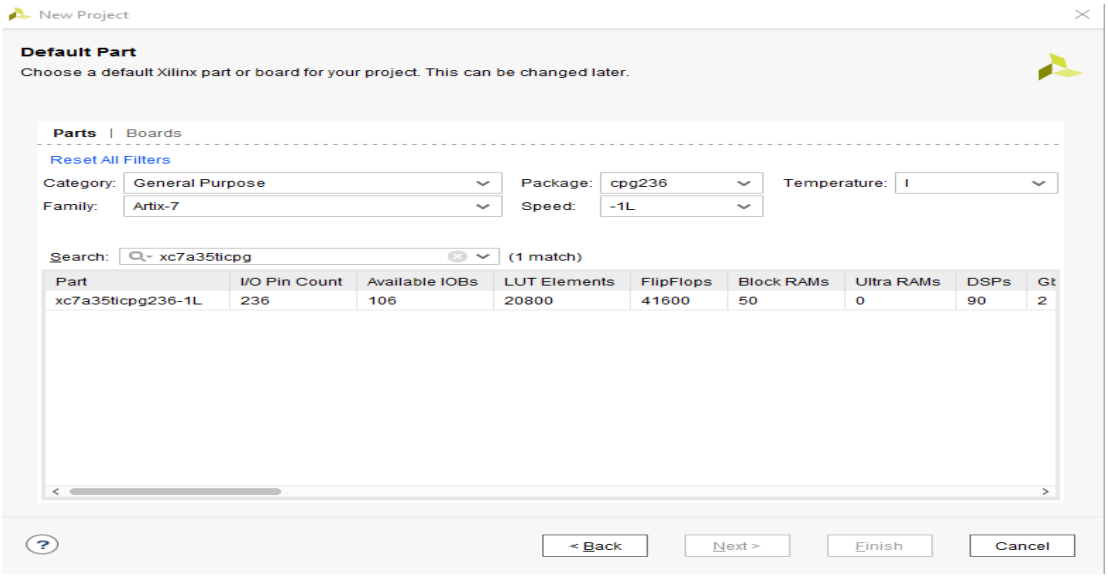
1. Open Xilinix Vivado.

2. In the **Xilinx-Project Navigator** window, Quick start, **New Project** and Name the project.

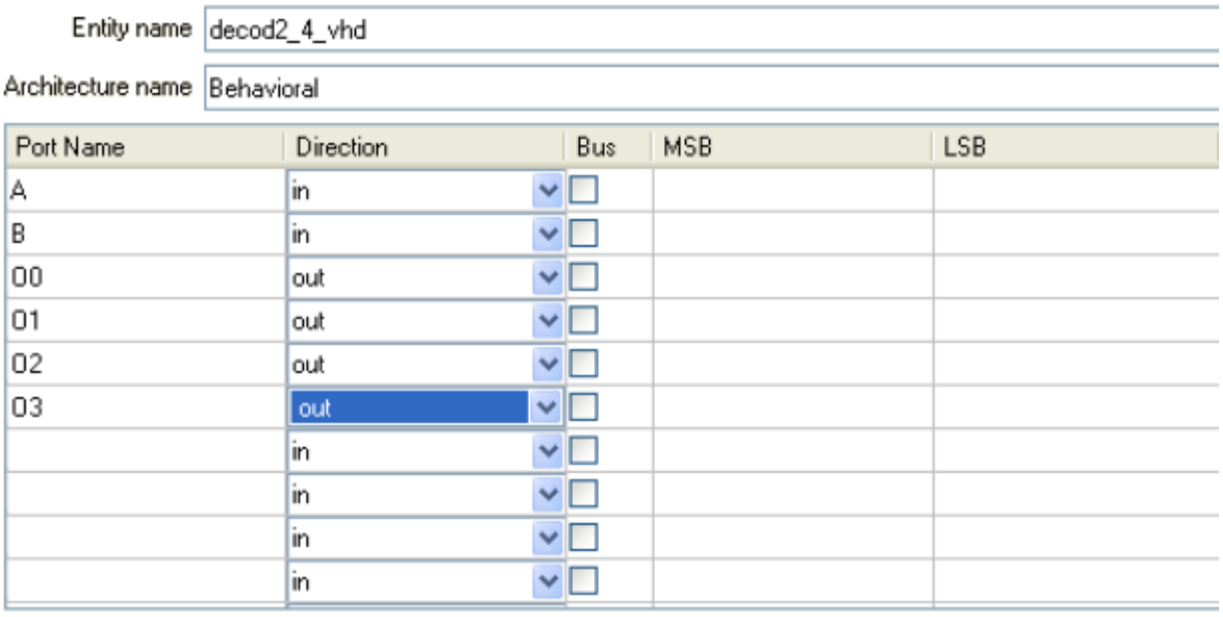
3. Select **New Source…** and the **New** window appears. In the New window, choose Schematic, type your file name (such as ***source_1***) in the File Name editor box, click on OK, and then click on the Next button.

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 and select OK.



6. In the created file, type the gates equivalent VHDL code.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity decod2_4_vhd is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           O0 : out  STD_LOGIC;
           O1 : out  STD_LOGIC;
           O2 : out  STD_LOGIC;
           O3 : out  STD_LOGIC);
end decod2_4_vhd;

architecture Behavioral of decod2_4_vhd is

begin

    o0 <= not (not A and not B);
    o1 <= not (not A and B);
    o2 <= not(A and not B);
    o3 <= not (A and B);

end Behavioral;
```

7. Next, we need to add To add a constraint file with the".xdc" extension, 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 "lab_2.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.
8. Uncomment (by deleting the # sign) sw[0], sw[1], sw[3],….. led[0], led[1],… lines. Note that each of them has two successive lines (Uncomment both of them). Do the following replacements: sw[0] ➔ A0, sw[1] ➔ A1,……, led[0] ➔ X, led[1] ➔ Y,…. , then Save the file

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. A green tab will appear in the top of the Vivado window, from which choose “open target” to program the hardware.
13. 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.
14. Fill in the following truth tables for all the gates by observing the inputs/outputs on the programmed board.

Truth Table

A	B		O0	O1	O2	O3
Sw1	Sw2		LED1	LED2	LED3	LED4
0	0					
0	1					
1	0					
1	1					

Checked by_____ Date _____

Section 2 Design a BCD decoder:

1. Repeat section 1 from step 1 to 6.

2. Define the inputs EN, A, B, C and D (LSB), and outputs O0 through O9 in the define VHDL source window. Your VHDL code template should look similar to below figure.

```
entity decobcd_vhd is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : in  STD_LOGIC;
           D : in  STD_LOGIC;
           EN : in  STD_LOGIC;
           O0 : out  STD_LOGIC;
           O1 : out  STD_LOGIC;
           O2 : out  STD_LOGIC;
           O3 : out  STD_LOGIC;
           O4 : out  STD_LOGIC;
           O5 : out  STD_LOGIC;
           O6 : out  STD_LOGIC;
           O7 : out  STD_LOGIC;
           O8 : out  STD_LOGIC;
           O9 : out  STD_LOGIC);
end decobcd_vhd;

architecture Behavioral of decobcd_vhd is

begin
   O0 <= EN and (not A and not B and not C and not D);
   O1 <= EN and (not A and not B and not C and D);
   O2 <= EN and (not A and not B and C and not D);
   O3 <= EN and (not A and not B and C and D);
   O4 <= EN and (not A and B and not C and not D);
   O5 <= EN and (not A and B and not C and D);
   O6 <= EN and (not A and B and C and not D);
   O7 <= EN and (not A and B and C and D);
   O8 <= EN and (A and not B and not C and not D);
   O9 <= EN and (A and not B and not C and D);


end Behavioral;
```

3. Repeat section 1 from step 10 to 15.

4. Fill in the following truth tables for all the gates by observing the inputs/outputs on the programmed board.

	Input				Output									
EN	A	B	C	D	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9

Checked by_____ Date _____

Questions:

1.) For a 3-to-8 decoder with active high outputs and an active high enable line (EN):
   a. List the truth table:

   b. Write the Boolean equations:

   c. Sketch the input and output timing waveforms for all input combinations.

2.) Name two applications of decoders.

3.) What is the enable line function in a decoder?