

Lab 4 – Combinational Circuits

CS 2052 Computer Architecture

Dept. of Computer Science and Engineering, University of Moratuwa

Learning Outcomes

In this lab we will design a decoder and a multiplexer. Decoders and multiplexers are 2 of the key components of a microprocessor. After completing the lab, you will be able to:

- design and develop a 3-to-8 decoder using schematics
- design and develop a 8-to-1 multiplexer using schematics
- verify their functionality via simulation and on the development board

Introduction

A decoder converts binary data from n coded inputs to a maximum of 2^n unique outputs. The decoder that we are going to build also has an enable pin/input. Enable input must be on for the decoder to function, otherwise we assume its outputs as a “disabled” output. A multiplex receives binary data from 2^n lines and connect them to a single output line based on a given n -bit selection. We will also add an enable pin to the multiplexer.

Decoders and multiplexers are 2 key components of a microprocessor. We will later use these components to build our simple microprocessor.

Building the Circuits

We will first build a 2-to-4 decoder. Then by combining two 2-to-4 decoders we will build a 3-to-8 decoder. Then finally using 3-to-8 decoder we will build 8_to_1_multiplexer.

Step 1: Building a 2-to-4 decoder.

Build the decoder shown in Fig. 1. Name the project as **Lab 4** and the schematic file as **Decoder_2_to_4**.

Step 2: Using busses.

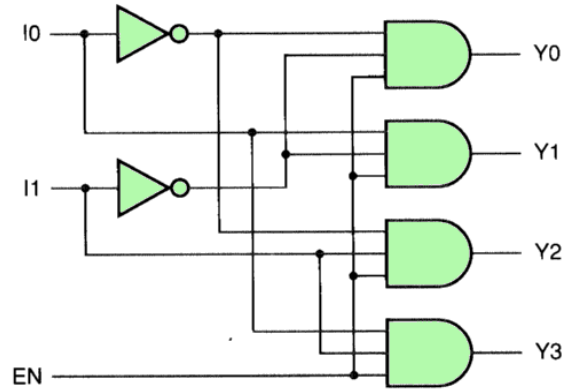
Instead of defining separate labels for each input and output as I0, I1, Y0, Y1, Y2, and Y3 we can define these as busses.

In digital design a *bus* is a collection of wires that carry data across multiple entities/modules. Then each wire in the bus can be separately accessed based on a zero-based index (like accessing an array variable).

When defining the module define the inputs and outputs as **I**, **EN**, and **Y** as seen in Fig. 2. Click **Ok**.

You would see the following lines added to your VHDL file:

```
Port ( I : in STD_LOGIC_VECTOR (1 downto 0);  
      EN : in STD_LOGIC;  
      Y : out STD_LOGIC_VECTOR (3 downto 0));
```



Source: <http://users.cis.fiu.edu/~prabakar/cda4101/Common/notes/lecture08.html>

Figure 1 – 2-to-4 decoder.

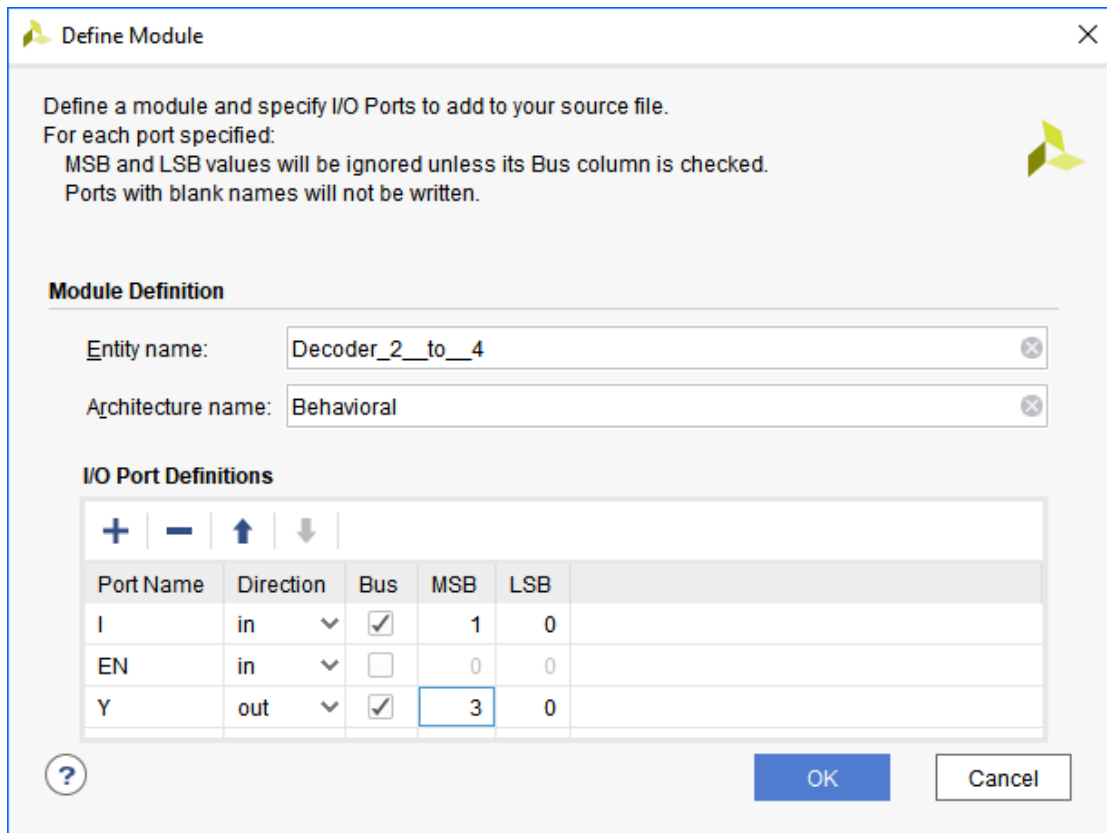


Figure 2 – Defining inputs and outputs as buses for Decoder_2_to_4.

The `std_logic_vector` VHDL type is used to define arrays of `std_logic` variables. Here `3 downto 0` indicates that the bus has four wires labelled from bit 3 (MSB) to 0 (LSB).

Step 3: Simulating circuit.

Simulate the circuit using XSim and make sure your decoder functions correctly. Name the Test Bench File as **TB_Decoder_2_to_4**.

Step 4: Build a 3-to-8 decoder.

Create a new schematic file and save it as **Decoder_3_to_8**. Label inputs and outputs as **I**, **EN**, and **Y**. Make I as a 3-bit bus while Y as a 8-bit bus.

Using two 2-to-4 decoders (label as **Decoder_2_to_4_0** and **Decoder_2_to_4_1**) and other gates, build a 3-to-8 decoder. You need to use the `component` keyword to add decoders.

Your final VHDL code could look like the following:

```
entity Decoder_3_to_8 is
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
          EN : in STD_LOGIC;
          Y : out STD_LOGIC_VECTOR (7 downto 0));
end Decoder_3_to_8;

architecture Behavioral of Decoder_3_to_8 is
    component Decoder_2_to_4
    port (
        I: in std_logic_vector;
        EN: in std_logic;
        Y: out std_logic_vector);
    end component;

    SIGNAL Y0, Y1 : std_logic_vector (3 downto 0);

begin
    Decoder_2_to_4_0 : Decoder_2_to_4
        port map (
            I(1 downto 0) => I(1 downto 0),
            EN => EN,
            Y(3 downto 0) => Y0(3 downto 0));

    Decoder_2_to_4_1 : Decoder_2_to_4
        port map (
            I(1 downto 0) => I(1 downto 0),
            EN => EN,
            Y(3 downto 0) => Y1(3 downto 0));

    Y(0) <= Y0(0) AND (NOT I(2));
    ...
    Y(4) <= Y1(0) AND I(2);
    ...
end Behavioral;
```

If **Decoder_3_to_8** is not automatically set as the top-level design set it.

Simulate the new decoder using XSim and make sure it functions correctly. Name the Test Bench File as **TB_Decoder_3_to_8**. It is not essential to try all the possible combinations of inputs. Instead, you can try a subset of the possible inputs.

Here is a method to try several input combinations based on your index number (this also enables your instructor to make sure each student uses a unique simulation pattern). Consider the 6 digits of your index number. Then convert your index number to binary. Set **I(2)** to **I(0)** according to the 3 Least Significant Bits (LSBs) of your index number. Then try the next 3 LSBs, and so

on. For example, suppose your index number is 123456R. Then its binary representation is 011 110 001 001 000 000 (ignoring the check digit). Try setting **I(2) – I(0)** as 000, then 001, 110, and so on. You may ignore repetitive values.

Step 5: Designing an 8-to-1 multiplexer.

Take a piece of paper and draw a logic circuit of an 8-to-1 multiplexer built using 3-to-8 decoder. You may also need several AND and OR gates. Label your input and output pins as **S**, **D**, **EN**, and **Y** (**Y** is the output). Both **S** and **D** should be buses of appropriate size.

Name the schematic file as **Mux_8_to_1**. Build the circuit and simulate the multiplexer.

Similar to Step 4 try a set of input combinations based on your index number. Make sure your multiplexer is functioning correctly.

Step 6: Connecting inputs and outputs.

Input and output pins are connected only to the top-level design. Therefore, first make sure multiplexer is already set as the Top Design.

Connect switches **SW0** to **SW7** as the inputs (from **D0** to **D7**) to **Mux_8_to_1**. Connect push button **BTN0** to **EN**. Connect **SW13** to **SW15** to control pins **S0** to **S2**. Set output **Y** to LED **LD0**.

Step 7: Test on BASYS 3.

Generate the programming file (i.e., bitstream) and load it to the BASYS 3 board.

Change the switches and push buttons on the board and verify the functionality of your multiplexer (check the output on LED).

Demonstrate the circuit to the instructor and get the Lab Completion Log signed.

Step 10: Lab Report

You need to submit a report for this lab. Your report should include the following:

- Student name and index number. Do not attach a separate front page
- State the assigned lab task in a few sentences
- All VHDL files
- All timing diagrams. Show all possible inputs for 2-to-4 decoder. For 3-to-8 decoder and 8-to-1 multiplexer use your index number as the input
- Conclusions from the lab

Submit the lab report at the beginning of the next lab.

Bibliography

- Digilent Inc., "Xilinx ISE WebPACK Schematic Capture Tutorial," Feb. 27, 2010.

Prepared By

- Dilum Bandara, PhD – Feb 19, 2014.
- Updated on Oct 03, 2018.