Experiment 6: Data Bus Implementation and Memory Design

Due date: 28.04.2021 23:00

Res. Asst. Ayşe Sayın
sayinays@itu.edu.tr

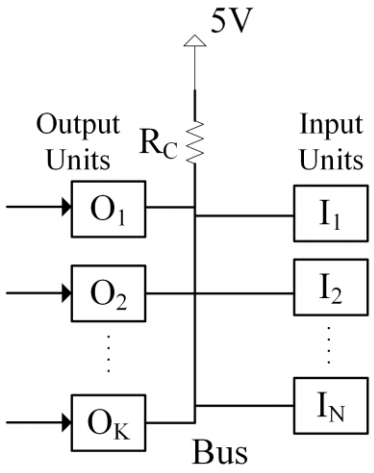
1 Introduction

In this experiment, data buses and basic memory are going to be implemented by using three-state buffers.

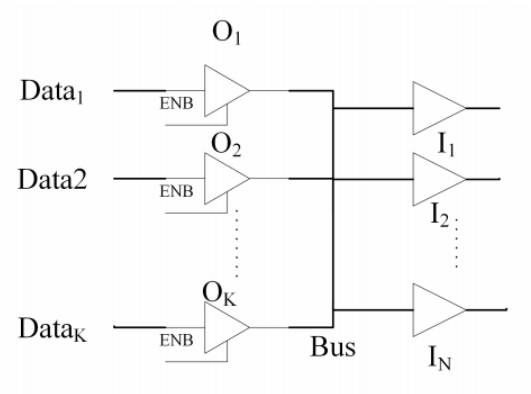
You can use any operator/code block in this experiment. Using multiplexer structure is forbidden. You must simulate the all parts and modules.

2 Basic Principles

Buses are frequently used in digital systems in order to reduce the cost of the physical connections. For example, a digital system which consists of N distinct units requires $N \cdot (N-1)/2$ physical connections to create a fully connected network. Since providing distinct physical lines (wires) for each connection is an expensive approach, sub-set of digital elements communicates through the shared bus in time shared manner. A bus can be implemented as a *wired OR* circuit by using ICs with open collector outputs or it can be implemented by using 3-state buffers. Abstract design of the *wired OR* bus is given in the figure below.



Buses can also be implemented by using 3-state buffers. There is no need for a resistor in this design. Output a non-active 3-state buffer will be in high-impedance. There has to be only one active 3-state buffer which runs the bus as an output. An example implementation of a bus by using 3-state buffers is given below.



3 Part 1

In this part of the experiment, you should implement an 8-bit bus by using 3-state buffers. The circuit diagram is given in Figure 1.

Hint: You must use the three-state buffer as a module. Thus, you must design a three-state buffer module before implementing the circuit.

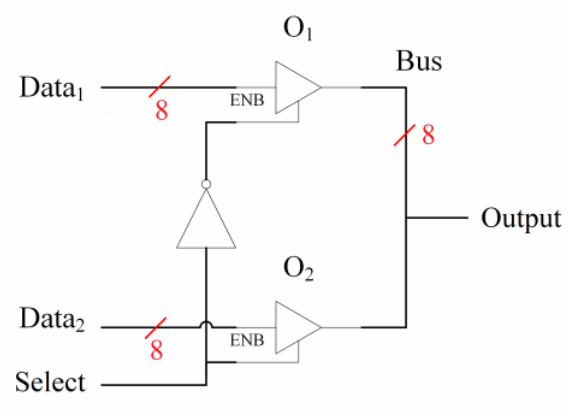


Figure 1: 8-bit data bus with 2 drivers with 3-state buffers

4 Part 2

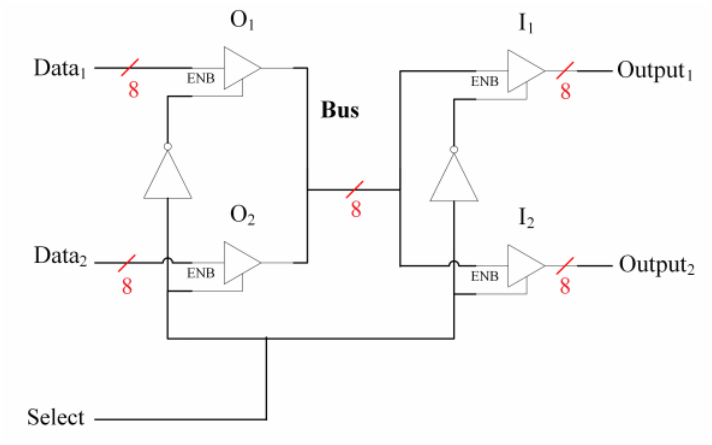


Figure 2: 8-bit data bus with 2 drivers and 2 readers

In this part of the experiment, you should implement the circuit given in Figure 2. This circuit contains a bus with two distinct outputs and inputs.

5 Part 3

In this part, you should implement an 8-bit memory line module. This module should take 8-bit data as input and give 8-bit data as output. Also, the module should take reset, line select, read enable, write enable, and clock inputs for required operations. Required operations are given below.

- At the rising edge of the clock signal, the module should store the data value which is given as input when the write enable and line select inputs are high.
- The module should clear the stored data at the falling edge of the reset signal.
- If read enable and line select inputs are high, the output of the module should be the stored data. Otherwise, the output should be high impedance.

6 Part 4

In this part, you should implement an 8 byte memory module using 8-bit memory line module. 8 byte memory module should take 8-bit data as input and give 8-bit data as output. Also, the module should take 3-bit address, chip select, reset, read enable, write enable, and clock inputs for required operations. Required operations are given below.

- **Nth** memory line should be selected when the chip select input high. Here, **N** is address number.
- At the rising edge of the clock signal, the selected memory line should store the data, which is given as input, if the write enable input is high.
- The module should clear all stored data in the memory lines at the falling edge of the reset signal.
- The output of the module should be the data of the selected memory line, if read enable input is high.

Hint: The output of the memory is a bus. Therefore, you do not need to use multiplexer in this experiment.

7 Part 5

In this part, you should implement a 32 byte memory module using 8 byte memory module. 32 byte memory module should take 8-bit data as input and give 8-bit data as output. Also, this module should take 5-bit address, reset, read enable, write enable, and clock inputs. 2 bits of the address input should be used for chip selection and rest of 3 bits should be used for selecting line. For instance, if the address input is 00111, zeroth chip (8-byte memory) and its 7th line (0-indexed) should be selected. Your module should be able to perform the operations below.

- At the rising edge of the clock signal, the selected memory line should store the data value, which is given as input, if the write enable is high.
- The module should clear the all stored data in the memory modules at the falling edge of the reset signal.

- If read enable is high, the output of the module should be the stored data of the selected memory line.

Example Memory Simulation

- Reset all lines
- Write 25 to Address 30
- Write 15 to Address 20
- Read Address 12
- Write 18 to Address 10
- Read Address 15
- Read Address 30
- Read Address 10

8 Part 6

In this part, you should implement a 128 byte memory module using 32 byte memory module. 128 byte memory module should take 32-bit data as input and give 32-bit data as output. Also, this module should take address, reset, read enable, write enable and clock inputs. Your module should be able to perform the operations below.

- At the rising edge of the clock signal, the selected memory line should store the data value, which is given as input, if the write enable is high.
- The module should clear the all stored data in the memory modules at the falling edge of the reset signal.
- If read enable is high, the output of the module should be the stored data of the selected memory line.

Hint: 32 byte memory modules have 8-bit inputs and 8-bit outputs. You can use concatenate operation to implement 128 byte memory.

9 Report

- You can use any software tool for your circuit designs and other figures. You may attach them to the report as figures by properly referencing them in the text. Also please do not put any hand written note in the report.
- Please use the table attributes of Latex. You can check out online Latex table generators (for example: https://www.tablesgenerator.com).

- Your report should contain information about the results of your simulations. If your implementations are not fully correct, discuss what the source of the errors might be in your report.
- You must cite your source used in the report. You can not directly copy and paste any sentences from any source even if you cite this document. You should read the document, write this information in your own words, then you must cite the resource.
- For further details about the report, please check Ninova e-learning system.

10 Submission Policy

- You should upload your experiment codes and report on Ninova, and please, do not send your experiment files via e-mail.
- Please submit 2 Verilog Design files for the experiment codes. One of them contains modules' codes, the other one consists of simulation codes.
- Please add comments to the code to clarify your intends.
- Your reports must be written with Latex format. Latex report template is available on Ninova. You can use any Latex editor whichever you want. If you upload your report without Latex file, you directly get 0 as your report grade. You should upload both .pdf and .tex files of your report.
- Please do not forget that late submissions are not accepted.
- Please make sure that you have written your full name and student ID number to every document you are submitting.
- Please only write the names of members who have worked on the experiment.
- Please do not hesitate to contact me (*sayinays@itu.edu.tr*) for any question.