ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY HOMEWORK REPORT

HOMEWORK NO : 3

LAB SESSION : FRIDAY - 16.30

GROUP NO : 18

GROUP MEMBERS:

150200916 : Denıs Iurıe Davıdoglu

150220770 : Onur Baylam

SPRING 2023

Contents

1 INTRODUCTION

In this homework, three-state logic is explored for designing data buses and basic memory. Along low and high states, a net can also get high impedance state, which behaves as if not connected to the circuit at all. By connecting a zero or an one to high impedance wire, there is no risk of creating short circuit, and the wire gets the same voltage as the connected input. This allows turning outputs of a logic circuit on and off, and connecting many three-state units to a common bus.

2 MATERIALS, METHODS AND RESULTS

2.1 Part 1

Before starting with the first part, the most basic component of three-state logic, the buffer, is designed. Buffers can be inverting and non-inverting, and with the goal of transmitting data to a bus, non-inverting type is considered. There are two inputs and one output; if enable input is low, output should be high impedance state, otherwise output will be equal to input data. Using a continuous assignment with ternary operator in Verilog, buffer is implemented and simulated:

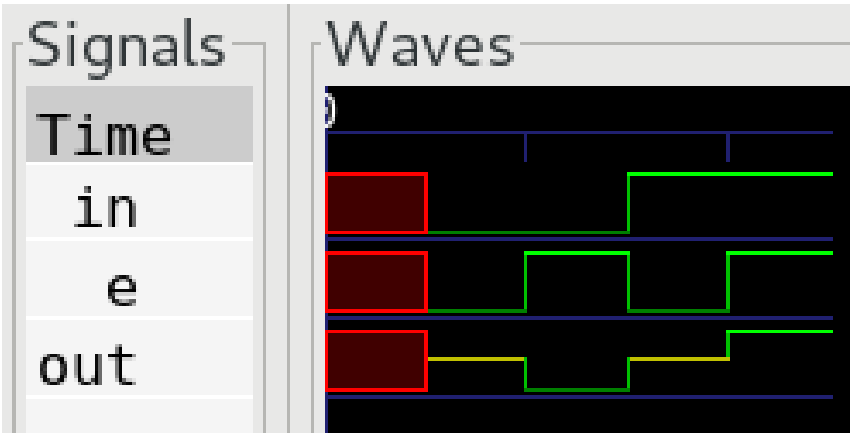

Figure 1: 3-state buffer simulation

Next, to test the capability of designed buffer to form a bus, the following circuit is tested:

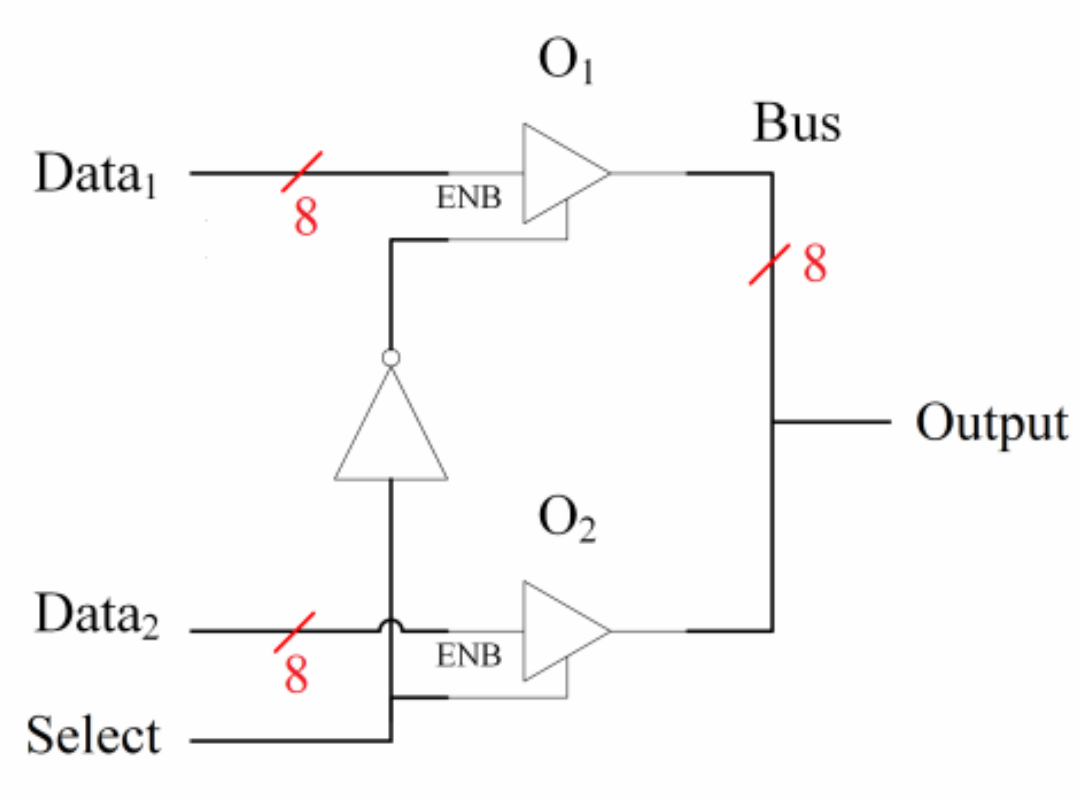

Figure 2: 8-bit data bus with 2 drivers with 3-state buffers

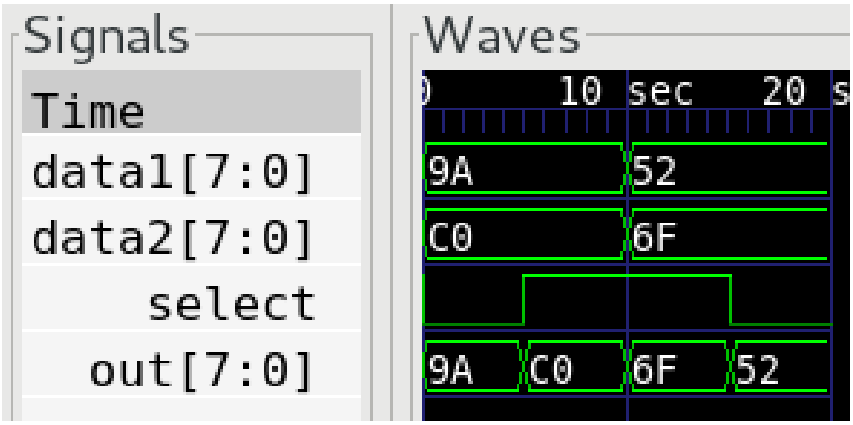

Figure 3: Part 1's simulation

As it is seen from simulation, we are capable of multiplexing data using select input, without actually using a multiplexer.

2.2 Part 2

Buses are supposed to not only be written to, but also read from. By extending the module from previous part, two controlled outputs can be added, as in the diagram:

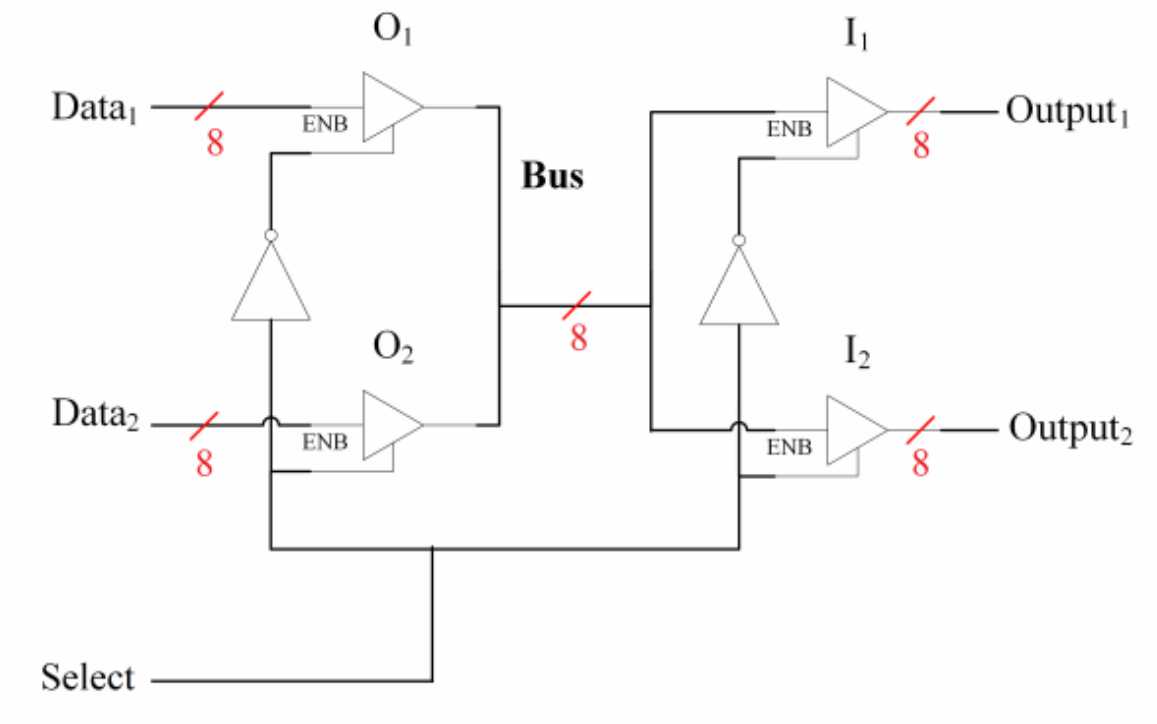

Figure 4: 8-bit data bus with 2 drivers and 2 readers

The behaviour of the outputs in this module slightly differs from the previous module. While the bus gets the same data, we do not pay attention to it anymore, and instead focus on the controlled outputs. When an output is not selected, it enters high impedance state:

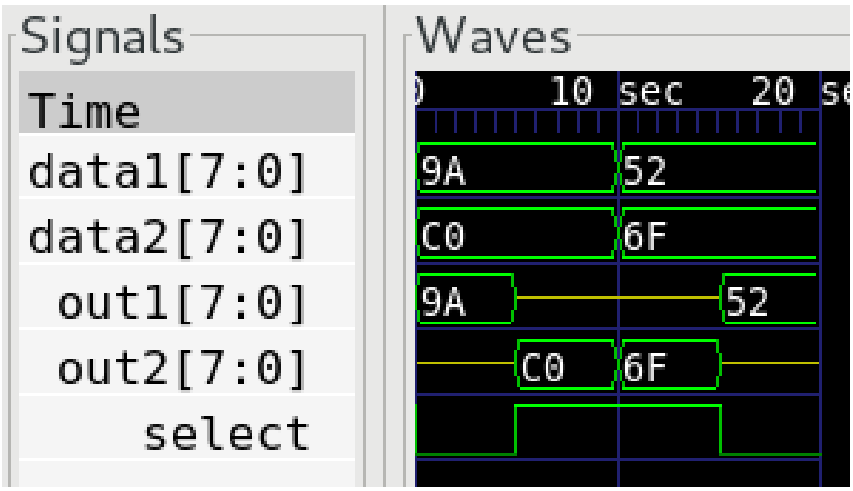

Figure 5: Part 2's simulation

2.3 Part 3

Here an 8-bit memory line module is explained. In order to use registers on a common bus, their outputs should be tri-stated using buffers. They should have write enable and read enable inputs, and normally these will not be active both at the same time, because the both are meant to be connected to bus. Module has a positive edge triggered clock for loading data and a negative edge triggered reset. Outputting internal data is done asynchronously, and the advantage is that in a real-world application, data should be already stabilized on the bus upon being read.

Clock	Reset	LineSel	Write	Read	Data	Output
⌐¯	ϕ	1	1	ϕ	Input	?
ϕ	¯⌐	ϕ	ϕ	ϕ	0	?
ϕ	ϕ	1	ϕ	1	?	Data
ϕ	ϕ	0	ϕ	1	?	Z
ϕ	ϕ	1	ϕ	0	?	Z
ϕ	ϕ	0	ϕ	0	?	Z

Figure 6: 8-bit memory line's truth table

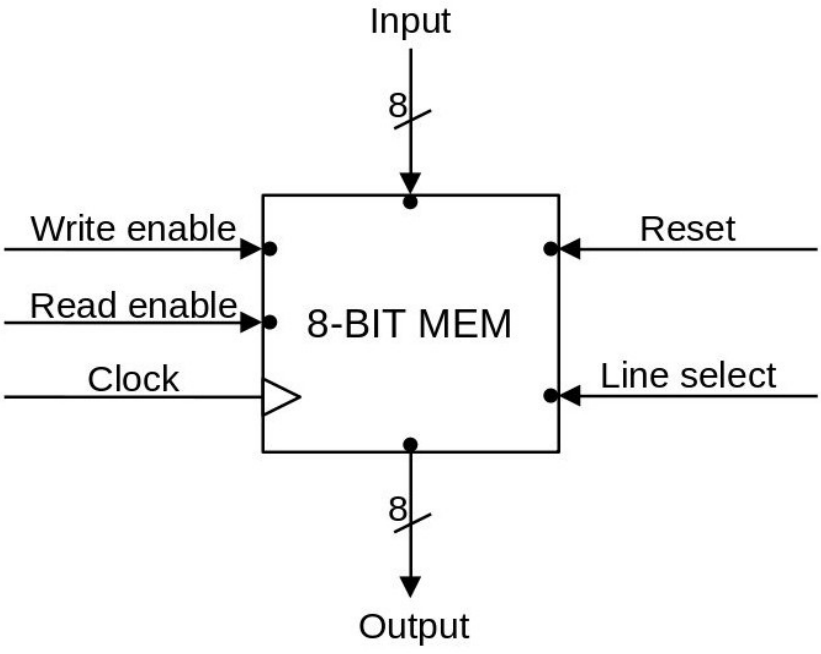

Figure 7: 8-bit memory line module

Memory line module's simulation results corresponds to the function table:

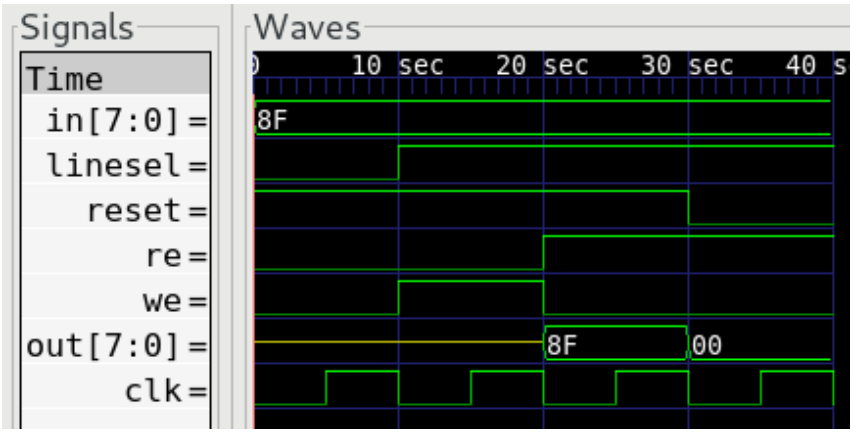

Figure 8: Part 3's simulation

2.4 Part 4

Many 8-bit memories can be arranged to form an 8-byte memory. Input, clock, reset, read enable and write enable are common for all of 8-bit memories, and to perform individual read and write operations, line selects are utilized. 8-bit memory has extra 3-bit adress input and chip select, which ramify through a decoder to enable small memories one at a time:

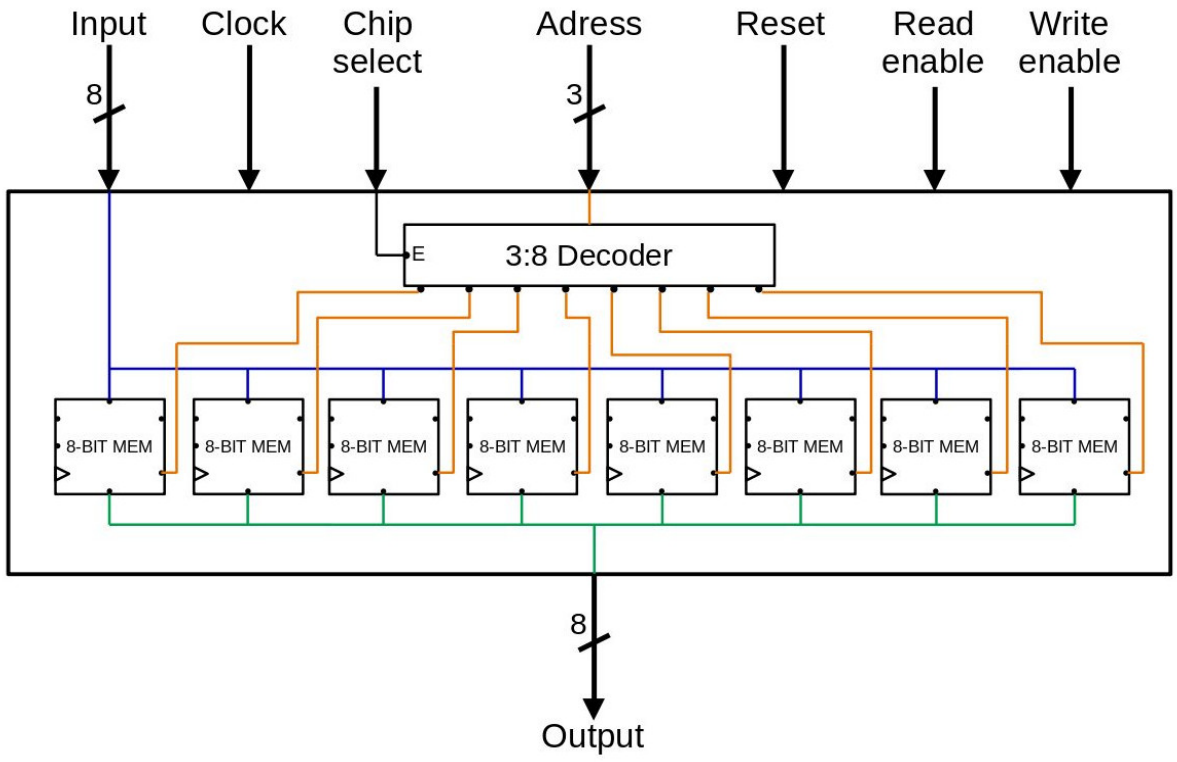

Figure 9: 8-byte memory using 8-bit memory line modules

Testbench for this memory checks the correctness of load, output and clear operations:

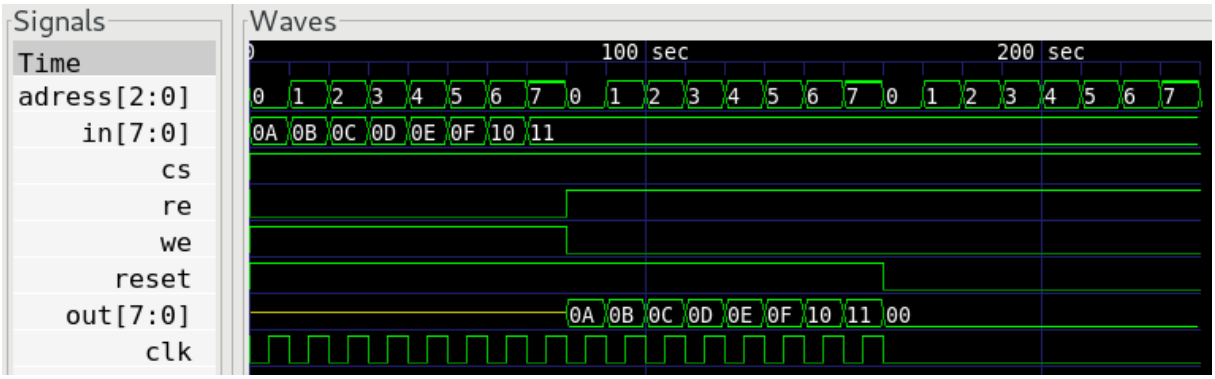

Figure 10: Part 4's simulation

2.5 Part 5

2.6 Part 6

3 CONCLUSION

Comment on any difficulties you have faced, what you have learned etc.