ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E
DIGITAL CIRCUITS LABORATORY HOMEWORK REPORT

HOMEWORK NO	:	3
HOMEWORK DATE	:	28.04.2023
LAB SESSION	:	FRIDAY - 14.00
GROUP NO	:	G3

GROUP MEMBERS:

150200054	:	ASLI YEL
150190028	:	SEVİM EFTAL AKŞEHİRLİ

SPRING 2023

Contents

1 INTRODUCTION

In this experiment, three-state buffers are used to create data buses and basic memory. Data buses provide an important function in carrying information between different parts of the system in contemporary digital circuits. Three-state buffers enable the connection of multiple parts to a single bus without conflicting with one another. In this experiment, we study the usefulness and advantages of creating data buses and basic memory using three-state buffers. We aim to have a detailed understanding of how three-state buffers can be used to build efficient and reliable digital systems by the end of the experiment.

2 MATERIALS AND METHODS

2.1 ADDITIONAL NECESSARY COMPONENTS

First of all, we designed an 8BIT three state buffer, a 3to8 Decoder, a 2to4 Decoder to be used in the other parts of the project. And we ensured that they work as expected by simulating them in order not to face unexpected problems in later parts.

2.1.1 Three-State Buffer

A three state buffer that forwards 8-bit input to the output when enable is "1" and gives an output state of high impedance (Hi-Z) when enable is "0" is implemented.

Enable	Data Input	Data Output
0	0	Hi-Z
0	1	Hi-Z
1	0	1
1	1	0

Table 1: Table For Three State Buffer

Enable	Data Input	Data Output
0	A	Hi-Z
1	A	A

Table 2: Table For Three State Buffer (A is 8-Bit)

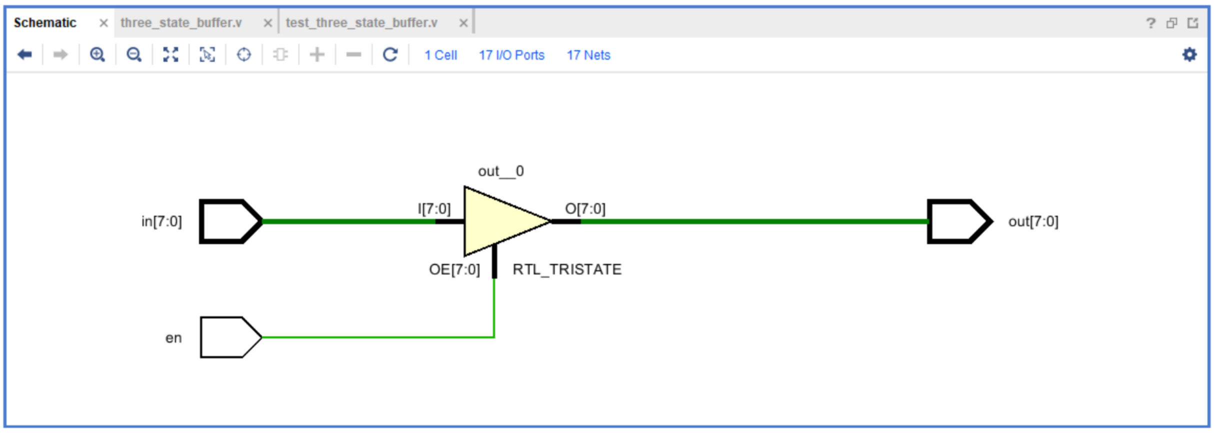


Figure 1: RTL Shematic of Three-State Buffer (8-Bit)

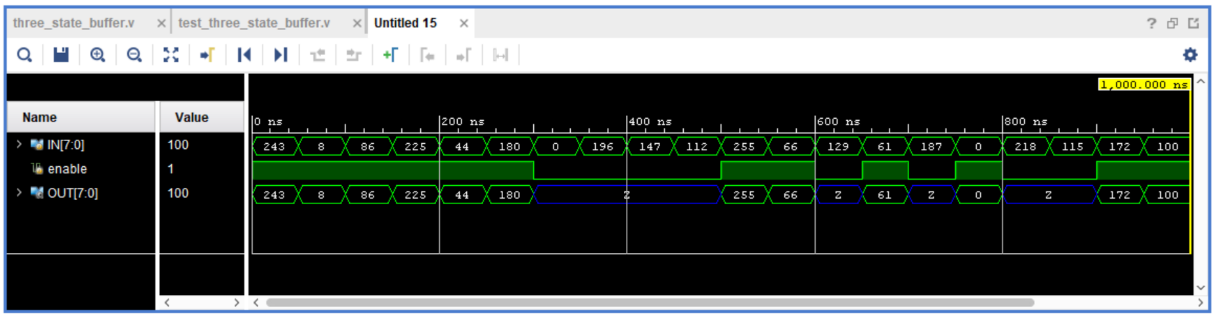


Figure 2: Simulation of Three-State Buffer (8-Bit)

2.1.2 2:4 DECODER

A 2to4 Decoder is implemented to be used for chip selection out of four 8Byte Chips according to the most significant 2 bits of the address input, in Part5 where we implement an 32Byte Memory module.

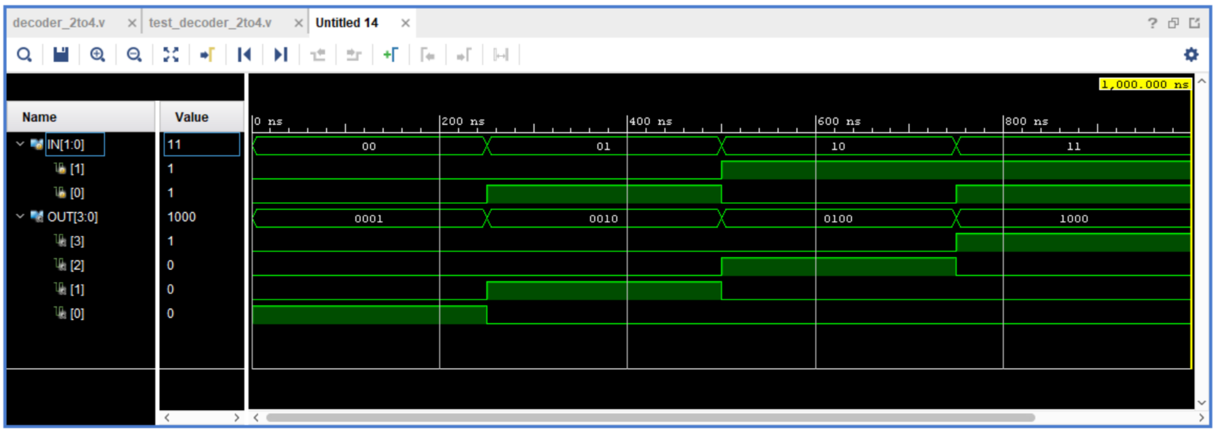


Figure 3: Simulation of 2:4 DECODER

2.1.3 3:8 DECODER

A 3to8 Decoder is implemented to be used for line selection of memory lines according to the address input, in Part4 where we implement an 8Byte Memory module.

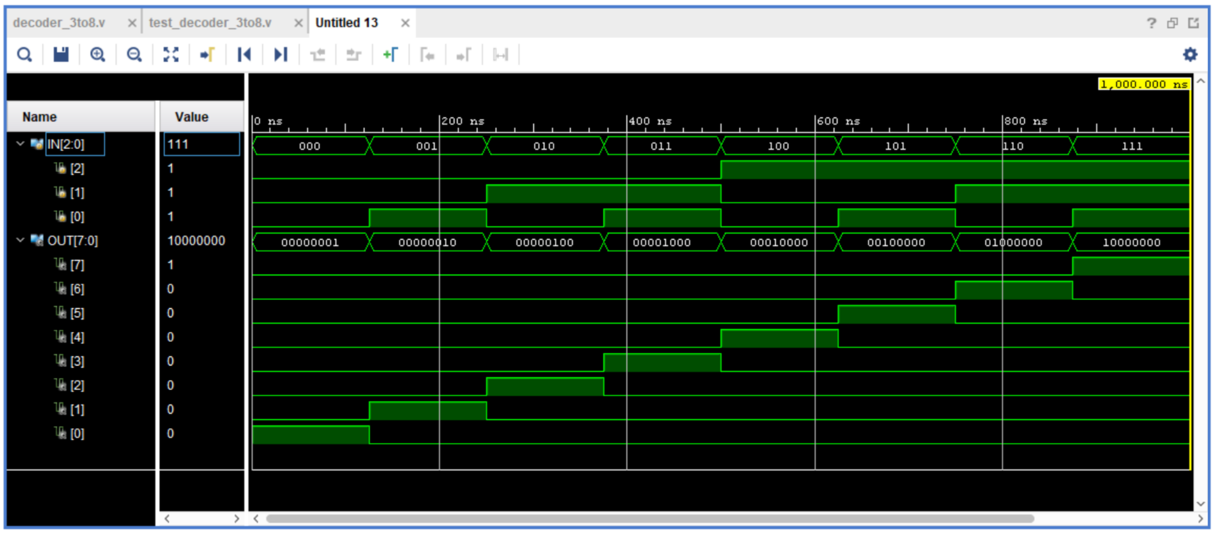


Figure 4: Simulation of 3:8 DECODER

2.2 EXPERIMENTS

2.2.1 Part 1 - 8-Bit Data Bus with 2 Drivers and 1 Reader

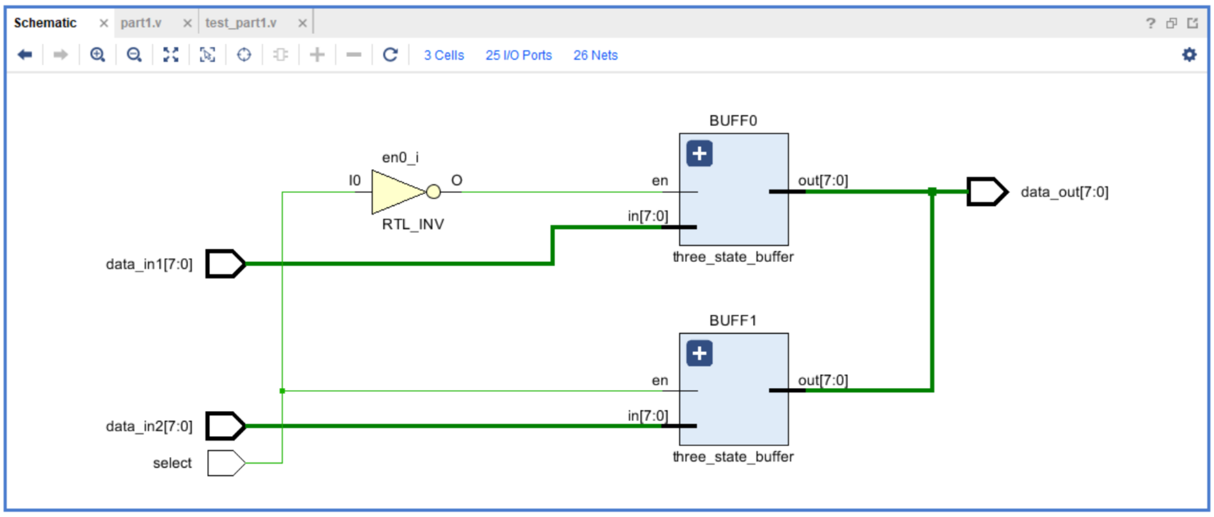


Figure 5: RTL Schematic of 8-Bit Data Bus with 2 Drivers and 1 Reader

2.2.2 Part 2 - 8-Bit Data Bus with 2 Drivers And 2 Readers

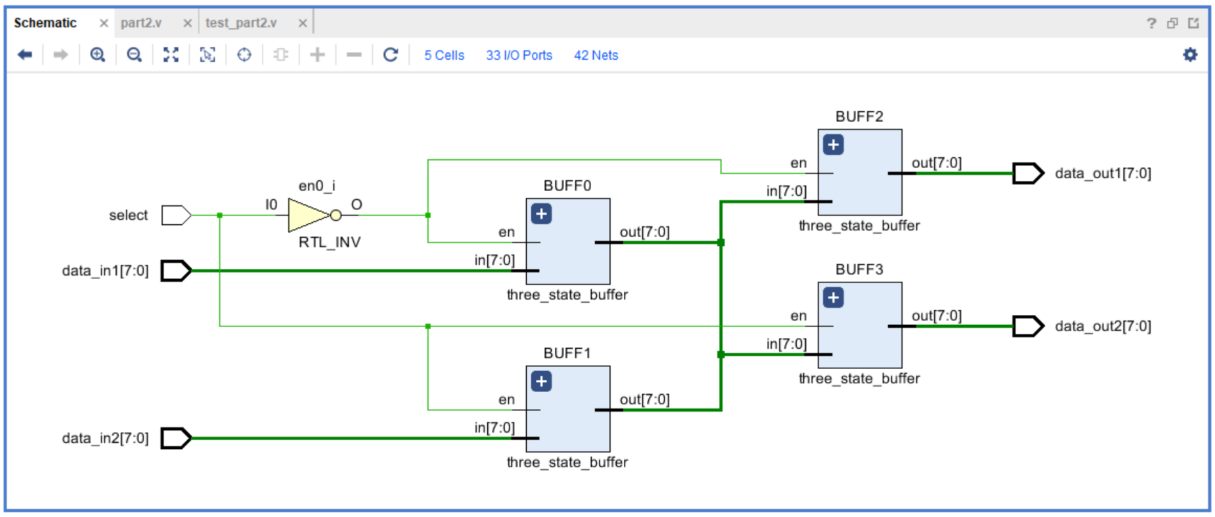


Figure 6: RTL Schematic of 8-Bit Data Bus with 2 Drivers and 2 Readers

2.2.3 Part 3 - 8-Bit Memory Line Module

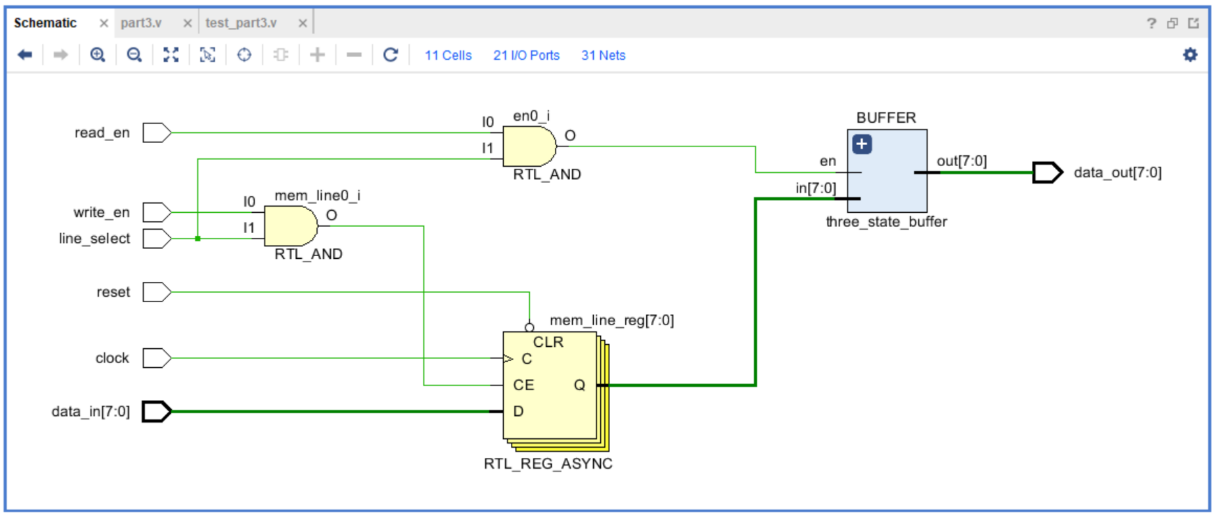


Figure 7: RTL Schematic of 8-Bit Memory Line Module

2.2.4 Part 4 - 8-Byte Memory Module

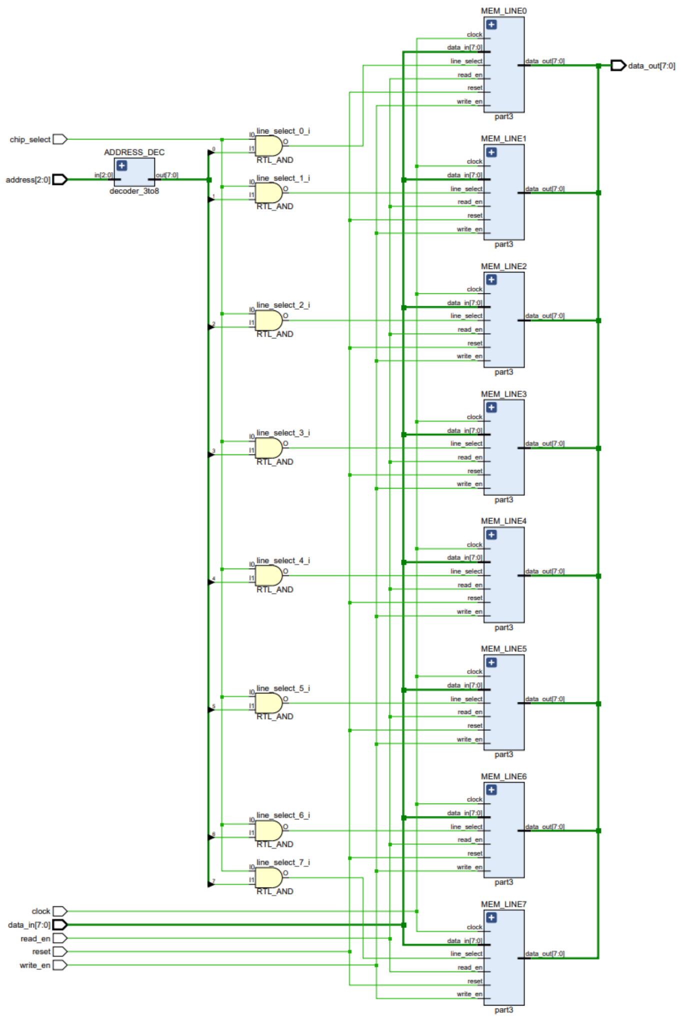


Figure 8: RTL Schematic of 8-Byte Memory Module

2.2.5 Part 5 - 32-Byte Memory Module

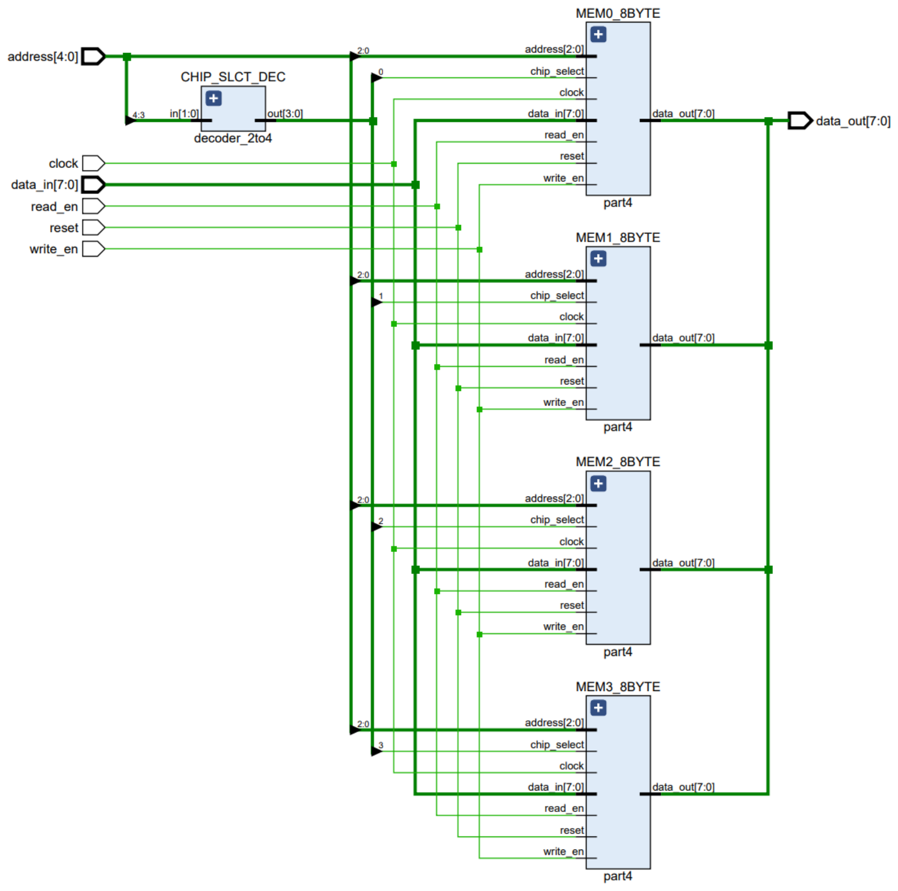


Figure 9: RTL Schematic of 32-Byte Memory Module

2.2.6 Part 6 - 128-Byte Memory Module

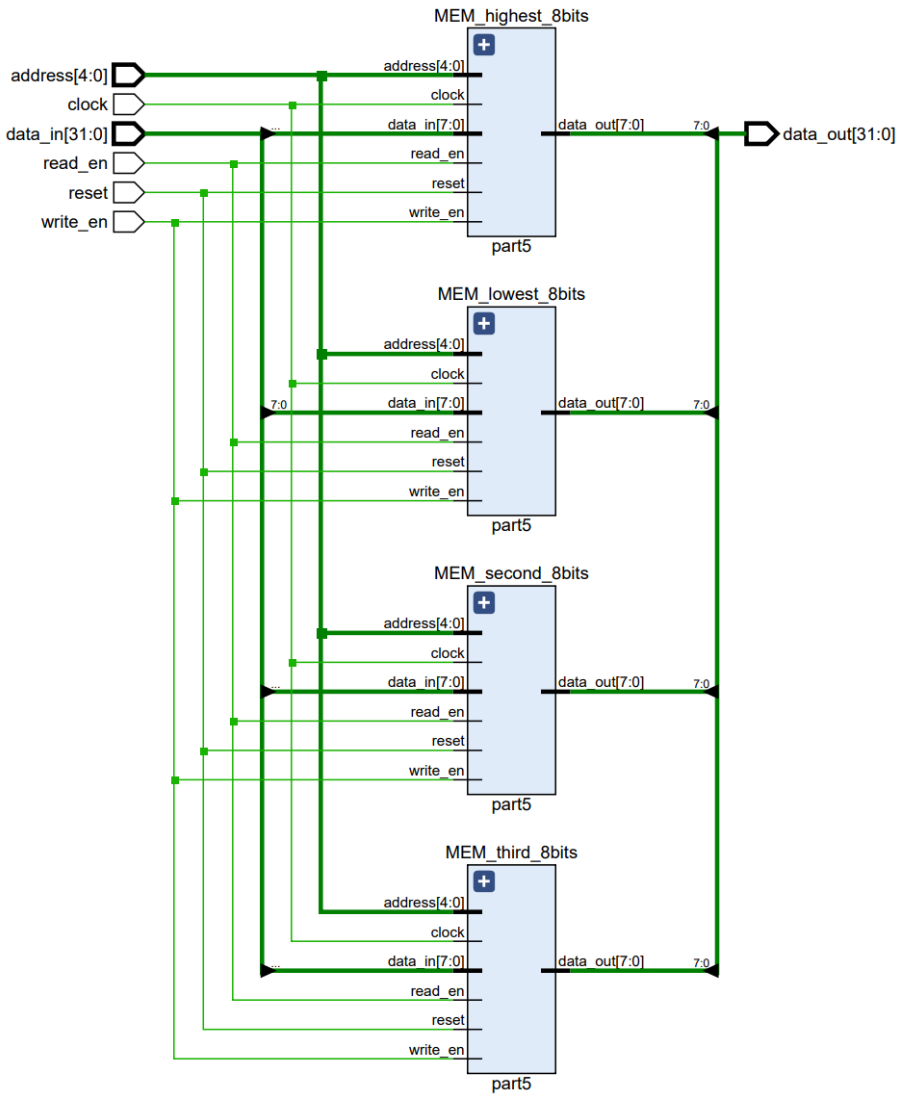


Figure 10: RTL Schematic of 128-Byte Memory Module

3 RESULTS

3.1 Part 1 - 8-bit Data Bus with 2 Drivers and 1 Reader

Selection Input	Data Input 1	Data Input 2	Data Output
0	selected	not selected	Data Input 1
1	not selected	selected	Data Input 2

Table 3: Table For 8-bit Data Bus with 2 Drivers and 1 Readers

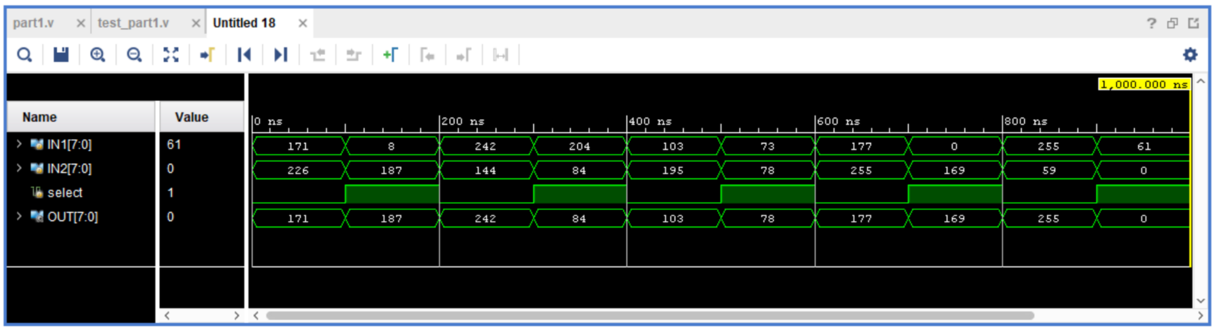


Figure 11: Simulation of 8-bit Data Bus with 2 Drivers and 1 Reader

3.2 Part 2 - 8-bit Data Bus with 2 Drivers and 2 Readers

Select	Data Input 1	Data Input 2	Bus	Data Out 1	Data Out 2
0	selected	not selected	Data Input 1	Data Input 1	Hi-Z
1	not selected	selected	Data Input 2	Hi-Z	Data Input 2

Table 4: Table For 8-Bit Data Bus with 2 Drivers and 2 Readers

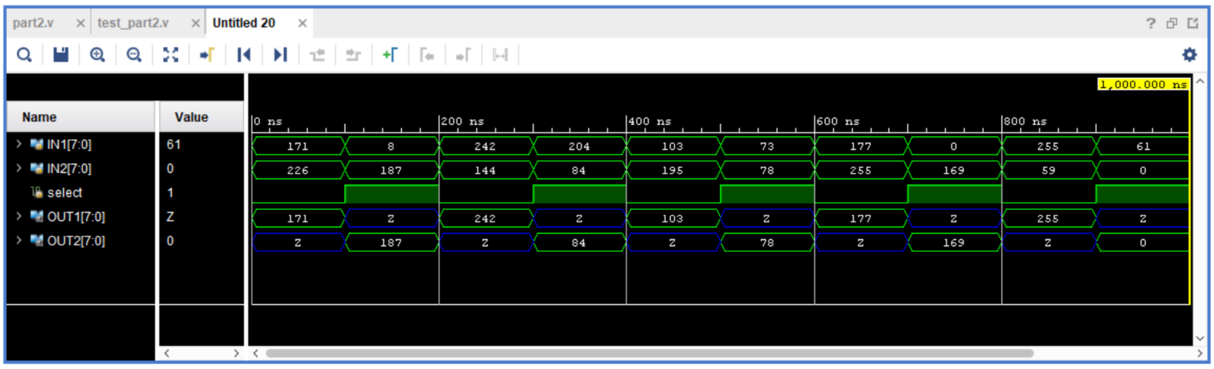


Figure 12: Simulation of 8-Bit Data Bus with 2 Drivers and 2 Readers

3.3 Part 3 - 8-Bit Memory Line Module

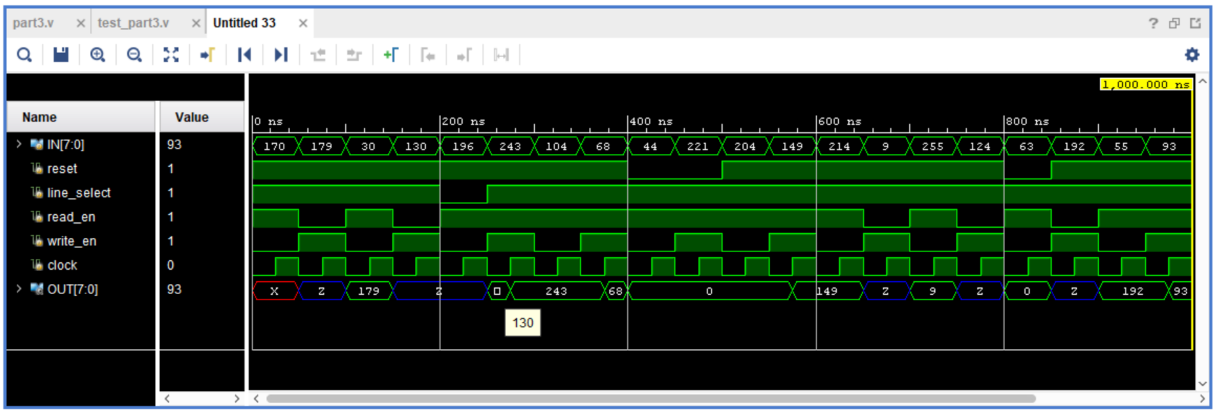


Figure 13: Simulation of 8-Bit Memory Line Module

3.4 Part 4 - 8-Byte Memory Module Using 8-bit Memory Line Module

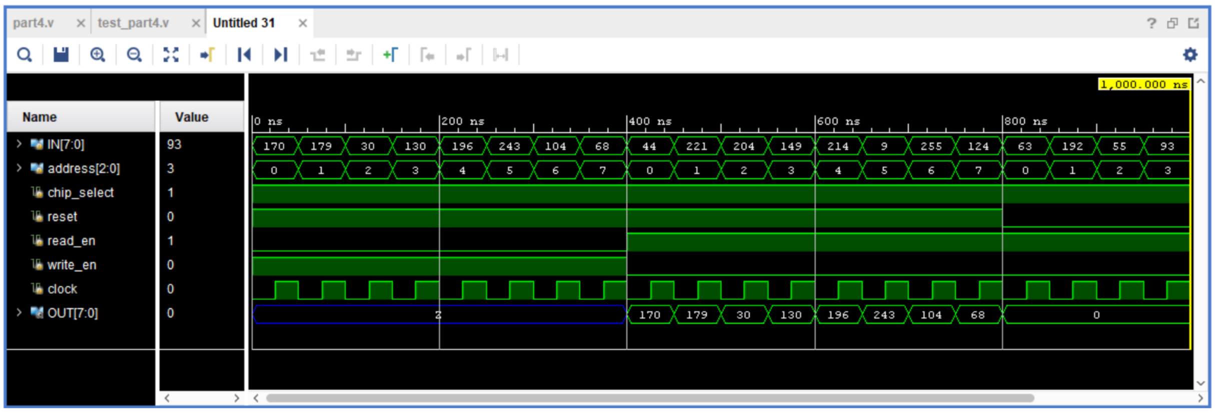


Figure 14: Simulation of 8-Byte Memory Module

3.5 Part 5 - 32-Byte Memory Module

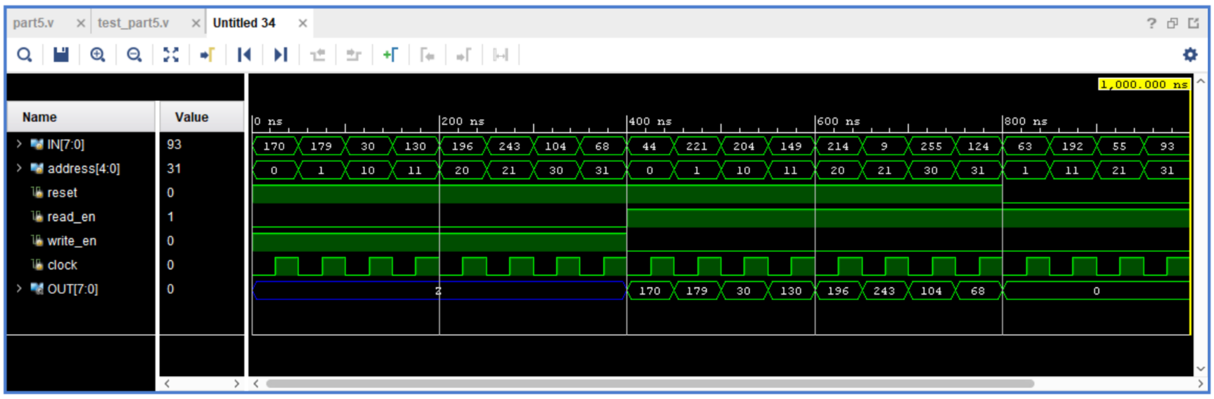
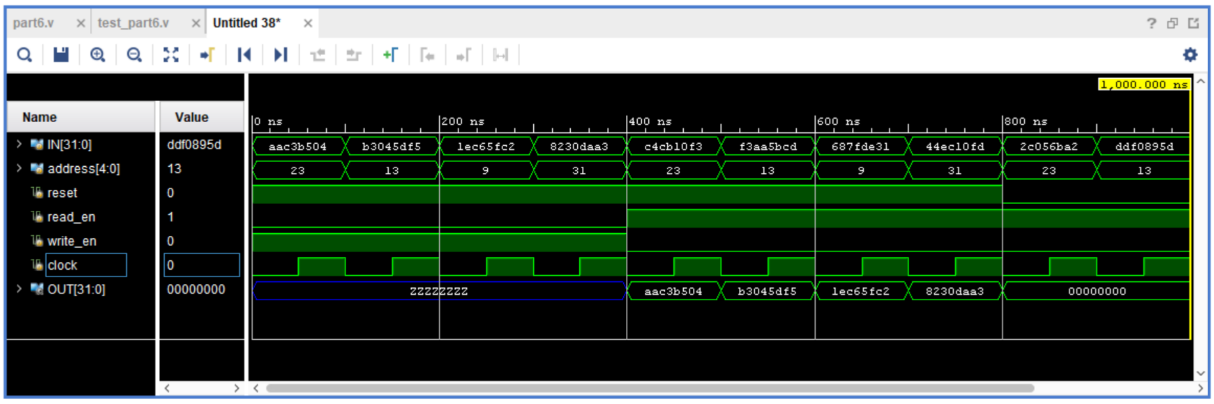


Figure 15: Simulation of 32-Byte Memory Module

3.6 Part 6 - 128-Byte Memory Module


part6.v
test_part6.v
Untitled 38*
1,000.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
aac3b504 b3045df5 1ec65fc2 8230daa3 c4cb10f3 f3aa5bcd 687fde31 44ec10fd 2c056ba2 ddf0895d
23 13 9 31 23 13 9 31 23 13
ZZZZZZZZ aac3b504 b3045df5 1ec65fc2 8230daa3 00000000

Name	Value
IN[31:0]	ddf0895d
address[4:0]	13
reset	0
read_en	1
write_en	0
clock	0
OUT[31:0]	00000000

Figure 16: Simulation of 128-Byte Memory Module

4 DISCUSSION

We started the experiment by creating three state buffer for 8-bit data, a 2to4 decoder and a 3to8 decoder.

In Part 1 we implemented a reader that selects and reads one of the two data inputs according to the selection input. Then we implemented two readers version of this in Part 2.

Then we implemented an 8-bit memory line module that stores the given input when line select and read enable inputs are high, clears the stored data at the falling edge of reset input, and forwards the stored data to the output when line select and write enable inputs are high, in Part 3.

We created an 8-byte memory module in Part 4 using the eight of 8-bit memory line module we created in Part 3. Address is decoded by using a 3to8 decoder and output of it identifies which line is selected and will be used for reading or writing data.

Using the four of the 8-byte memory module (memory chips) implemented in the previous part, we implemented a 32-byte memory unit in Part 5. The most significant two bits of the address input is decoded with a 2to4 decoder and the output of it selects the memory chip that will be used to store or forward data. Which line of the selected chip is used is again selected by a 3to8 decoder that takes the last 3 bits of the address input.

We created a 128 byte memory unit in the experiment's final part using four of the 32-byte memory unit from the part before. This element is different from others in that it takes 32-bit input and expects 32-bit output. To provide input data to a 32-byte memory module with 8-bit input, we separated the data into 4 parts. The received result was then added to the main output.

5 CONCLUSION

Using mainly three-state buffers, we created several parts of units that include memory units in this experiment. We started with creating the 3-state buffers that were given to us, after which we implemented an 8-bit memory unit. With the help of it, we completed the experiment and created memory units of 8 bytes, 32 bytes, and 128 bytes, respectively.

REFERENCES