**Memory Design**

**CENG 3151**

Submitted by

Michael Lanford

1816337

Computer Engineering

University of Houston-Clear Lake

Houston, Texas 77058

March 15, 2023

# Table of Contents

**1** [**Abstract 3**](#_Toc126060766)

[**2** **Introduction** 3](#_Toc126060767)

**3 Requirements……………………………………………………………………….……......3**

**4 Prelab………………………………………………………………………………..……….4**

**5 Implementation……………………………………………………….…………………......4**

5.1 Design Code / Design Diagrams**……………………………………………………………5**

5.2 Schematics**……………………………………………………………………………….....7**

5.3 Testbench**…………………………………………………………………………………....8**

5.4 Waveform / Results**………………………………………………………………………...13**

[**6** **Conclusion** 13](#_Toc126060768)

**Abstract**

# ROMs are decoders that receive input in binary form and translate it to its decimal form and outputs that. It permanently writes data on a chip and lasts even after a computer is turned off. RAM is temporary memory where the computer stores data it needs to be able to retrieve quickly. When the computer is turned off the data is erased. In this lab, we will be using Xilinx Vivado to build two circuits: one that simulates ROM and one that simulates RAM. Each of these will be 32 bits wide. The major results of this experiment will be a waveform that shows the correct output for each input, which will reflect our input values.

# Introduction

For this project, we will be using Xilinx Vivado to build and test a 32-bit ROM circuit and a 32-bit RAM circuit that each will accept some input and produce some output.

1. **Requirements**

Design ROM that has 16 locations each 32 bits wide with 3 inputs: the chipset(CS) that activates the chip, the Clock, and the Address which is a vector. This circuit will output a vector of initialized data. Design RAM that has 16 locations each 32 bits wide with 5 inputs: the chipset(CS) that activates the chip, the Clock, the Address, the R/W, and the Input Data. This circuit will output a vector that can send and receive data. The figures of the circuits can be seen below:

Diagram

Description automatically generated

**Figure 1:** Diagram for the ROM circuit to be designed.

Diagram

Description automatically generated

**Figure 2:** Diagram for the RAM circuit to be designed.

1. **Prelab**

For this prelab, we were required to write about the working of a ROM, type conversion in VHDL, and the working of RAM.

A ROM is a decoder that receives input in binary form and translates it to its decimal form and outputs that. It permanently writes data on a chip and lasts even after a computer is turned off. VHDL has several kinds of type conversion. To go between the std\_logic\_vector type and unsigned/signed you have to use type casting, but both signals have to have the same bit width. To go from signed/unsigned to integer, you need to use the to\_integer function call from numeric\_std. Finally, to go from std\_logic\_vector to integer, you need to use type casting. RAM is temporary memory where the computer stores data it needs to be able to retrieve quickly. When the computer is turned off the data is erased.

1. **Implementation**

The first step of implementation was to create a new project in Xilinx Vivado. After setting up the project with the correct options, we added a design source file for the ROM circuit and added the necessary inputs and outputs to it. We then coded the clock process and the TYPE and constant for the ROM. After that, we created the simulation file and added in the component instantiation, interface signal declarations, instance port map, the clock process, and the test cases to it then tested the waveform. For the RAM circuit, we created another new project in Xilinx Vivado. After setting up the project with the correct options, we added a design source file and added the necessary inputs and outputs to it. We then coded the clock process and the TYPE and signal for the RAM. After that, we created the simulation file and added in the component instantiation, interface signal declarations, instance port map, the clock process, and the test cases to it then tested the waveform.

**4.1 Design Code / Design Diagrams**

------Part 1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

--Input and Output declarations

entity Lab6Pt1Design is

Port ( Clk : in STD\_LOGIC;

CS : in STD\_LOGIC;

Address : in STD\_LOGIC\_VECTOR (3 downto 0);

Data\_Out : out STD\_LOGIC\_VECTOR (31 downto 0));

end Lab6Pt1Design;

architecture Behavioral of Lab6Pt1Design is

--Type declaration

TYPE rom\_16 is array(0 to 15) of std\_logic\_vector(31 downto 0);

constant ROM: rom\_16 := (

x"DEADEEEF",--The test bench should call for this output

x"1234ABCD",

x"CAFEBABE",

x"FFFFFFFF",--Should be seen as output after the first case

x"CAD8340E",

x"D2A35101",

x"A0FE1283",

x"E82A2B3C",--Should be seen as another output

x"A932E1BD",

x"ABCDEF01",

x"AAAAAAAA",--Another output

x"00000000",

x"11111111",

x"33333333",

x"CCCCCCCC",

x"AFAFAFAF" --Another output

);

begin

--Clock process

process(clk) is begin

if (rising\_edge(clk) and CS = '1') then

Data\_Out <= rom(to\_integer(unsigned(address)));--type casting

else

Data\_Out <= (others => 'Z');--This should be seen after each output is displayed once

end if;

end process;

end Behavioral;

------Part 2

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

--Input and Output declarations

entity Lab6Pt2Design is

Port ( Clk : in STD\_LOGIC;

CS : in STD\_LOGIC;

R\_W : in STD\_LOGIC;

Address : in STD\_LOGIC\_VECTOR (3 downto 0);

Data\_In : in STD\_LOGIC\_VECTOR (31 downto 0);

Data\_Out : out STD\_LOGIC\_VECTOR (31 downto 0));

end Lab6Pt2Design;

architecture Behavioral of Lab6Pt2Design is

--Type and signal declarations

type RAM\_16 is array (0 to 15) of std\_logic\_vector(31 downto 0);

signal RAM: RAM\_16;

begin

--Clock process

process(clk) is begin

if (rising\_edge(clk)) then

if (CS = '1') then--If chipset is 1

if (R\_W = '1') then--If R\_W is 1

Data\_Out <= ram(to\_integer(unsigned(address)));

else--If R\_W is 0

ram(to\_integer(unsigned(address))) <= Data\_In;

end if;

else--If chipset is 0

Data\_Out <= (others => 'Z');

end if;

end if;

end process;

end Behavioral;

**4.2 Schematics**

**Diagram

Description automatically generated**

**Figure 3:** ROM circuit.

Diagram, schematic

Description automatically generated

**Figure 4:** RAM circuit.

**4.3 Testbench**

--Part 1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity Lab6Pt1Sim is

-- Port ( );

end Lab6Pt1Sim;

architecture Behavioral of Lab6Pt1Sim is

component Lab6Pt1Design is--Instantiate the component

Port ( Clk : in STD\_LOGIC;

CS : in STD\_LOGIC;

Address : in STD\_LOGIC\_VECTOR (3 downto 0);

Data\_Out : out STD\_LOGIC\_VECTOR (31 downto 0));

end component;

signal Clk, CS: std\_logic;--Signal declarations

signal Address: std\_logic\_vector(3 downto 0);

signal Data\_Out: std\_logic\_vector(31 downto 0);

constant Clock\_period : time := 10 ns;--for clock signal

begin

uut: Lab6Pt1Design PORT MAP(Clk, CS, Address, Data\_Out);--Port maps

process--Clock process

begin

Clk <= '0';

wait for Clock\_period/2;

Clk <= '1';

wait for Clock\_period/2;

end process;

--Stimulus Address

process

begin

--Test case; CS is enabled

--Read operations

CS <= '1';

Address <= x"0";--output should be deadeef if CS = 1

wait for 20 ns;

Address <= x"3";--output should be ffffffff if CS = 1

wait for 20 ns;

Address <= x"7";--output should be E82A2B3C if CS = 1

wait for 20 ns;

Address <= x"a";--output should be aaaaaaaa if CS = 1

wait for 20 ns;

Address <= x"f";--output should be afafafaf if CS = 1

wait for 20 ns;

--Test case: CS is not enabled

CS <='0';--output should be zzzzzzzz if CS = 0

end process;

end Behavioral;

--Part 2

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity Lab6Pt2Sim is

-- Port ( );

end Lab6Pt2Sim;

architecture Behavioral of Lab6Pt2Sim is

component Lab6Pt2Design is--Instantiate the component

Port ( Clk : in STD\_LOGIC;

CS : in STD\_LOGIC;

R\_W : in STD\_LOGIC;

Address : in STD\_LOGIC\_VECTOR (3 downto 0);

Data\_In : in STD\_LOGIC\_VECTOR (31 downto 0);

Data\_Out : out STD\_LOGIC\_VECTOR (31 downto 0));

end component;

signal Clk, CS, R\_W: std\_logic;--Signal declarations

signal Address: std\_logic\_vector(3 downto 0);

signal Data\_In, Data\_Out: std\_logic\_vector(31 downto 0);

constant Clock\_period : time := 10 ns;--for clock signal

begin

uut: Lab6Pt2Design PORT MAP(Clk, CS, R\_W, Address, Data\_In, Data\_Out);--Port maps

process--Clock process

begin

Clk <= '0';

wait for Clock\_period/2;

Clk <= '1';

wait for Clock\_period/2;

end process;

process

begin

CS <= '1';--Test case; CS is enabled

R\_W <= '0';--write

Address <= x"0";

Data\_In <=x"00000000";

wait for 20 ns;

CS <= '1';--Read operation

R\_W <= '1';--read

Address <=x"0";

wait for 20 ns;

CS <= '1';--Write to address 15

R\_W <= '0';--write

Address <= x"f";

Data\_In <=x"AFAFAFAF";

wait for 20 ns;

CS <= '1';--Read from address 15

R\_W <= '1';--Read

Address <= x"f";

wait for 20 ns;

CS <='0';--Test case; CS is not enabled

wait for 20 ns;--output should be ZZZZZZZZ

CS <= '1';--One more read

R\_W <= '1';

Address <= x"0";

wait;

end process;

end Behavioral;

**4.4 Waveform / Results**

The waveforms below shows that the four programs we made above in the Testbench and Design Code / Design Diagrams sections was able to produce correct results for each of our inputs that we created.

A screenshot of a computer

Description automatically generated

**Figure 5:** ROM circuit Waveform.

A screenshot of a computer

Description automatically generated

**Figure 6:** RAM circuit Waveform.

# Conclusion

In this lab, we were able to successfully code a ROM and RAM circuit in Xilinx Vivado by using the little code snippets given to us in our prelab as a base for our code. These programs were made to be able to simulate how memory acts inside a computer, which can be seen in the waveforms due to the correct output being produced for what we inputted.