**Design of a 32-bit shifter**

**CENG 3151**

Submitted by

Michael Lanford

1816337

Computer Engineering

University of Houston-Clear Lake

Houston, Texas 77058

March 31, 2023

# Table of Contents

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

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

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

**4 Prelab………………………………………………………………………………..……….3**

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

5.1 Design Code / Design Diagrams**……………………………………………………………4**

5.2 Schematics**……………………………………………………………………………….....5**

5.3 Testbench**…………………………………………………………………………………....6**

5.4 Waveform / Results**………………………………………………………………………...8**

[**6** **Conclusion** 8](#_Toc126060768)

**Abstract**

A logical shifter shifts bits left or right with vacant bits filled with zeros. An arithmetic shifter shifts bits left or right with the vacant bits filled with the sign bits. For example, a logical shift of the number 00001101 would be 00011010. These shifters are used to multiply and divide signed numbers, and this is the same reason why they are important in computer architecture, they reduce the amount of architecture needed to compute multiplication and division.

# Introduction

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

1. **Requirements**

Design a 32-bit shifter with 4 inputs: Shift\_Mode, Shift\_Direction, Nbr\_Of\_Bit, and Data\_In. This circuit will output a 32-bit vector of data. The figure of the circuit can be seen below:

Diagram

Description automatically generated with medium confidence

**Figure 1:** Diagram for the 32-bit shifter to be designed.

1. **Prelab**

For this prelab, we were required to write about the working of a logical shifter and arithmetic shifter.

A logical shifter shifts bits left or right with vacant bits filled with zeros. An arithmetic shifter shifts bits left or right with the vacant bits filled with the sign bits. For example, a logical shift of the number 00001101 would be 00011010. These shifters are used to multiply and divide signed numbers, and this is the same reason why they are important in computer architecture, they reduce the amount of architecture needed to compute multiplication and division.

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 Shifter circuit and added the necessary inputs and outputs to it. We then coded the arithmetic and logical shift operations for the shifter. After that, we created the simulation file and added in the component instantiation, interface signal declarations, instance port map, and the test cases to it then tested the waveform.

**4.1 Design Code / Design Diagrams**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity Lab8Design is -- Input and Output Declarations

Port ( Shift\_Mode : in STD\_LOGIC;

Shift\_Direction : in STD\_LOGIC;

Nbr\_of\_Bit : in STD\_LOGIC\_VECTOR (4 downto 0);

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

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

end Lab8Design;

architecture Behavioral of Lab8Design is

begin

process(Shift\_Mode, Shift\_Direction, Nbr\_of\_Bit, Data\_In) --Process for the arithmetic and logical shifts

begin

if Shift\_Mode = '0' then --Logical Shift

if Shift\_Direction = '0' then --Left

Data\_Out <= std\_logic\_vector(shift\_left(unsigned(Data\_In), to\_integer(unsigned(Nbr\_of\_Bit))));

else --Right

Data\_Out <= std\_logic\_vector(shift\_right(unsigned(Data\_In), to\_integer(unsigned(Nbr\_of\_Bit))));

end if;

else --Arithmetic Shift

if Shift\_Direction = '0' then --Left

Data\_Out <= std\_logic\_vector(shift\_left(signed(Data\_In), to\_integer(signed(Nbr\_of\_Bit))));

else --Right

Data\_Out <= std\_logic\_vector(shift\_right(signed(Data\_In), to\_integer(signed(Nbr\_of\_Bit))));

end if;

end if;

end process;

end Behavioral;

**4.2 Schematics**

**Diagram

Description automatically generated**

**Figure 2:** Shifter circuit.

**4.3 Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Lab8Sim is

-- Port ( );

end Lab8Sim;

architecture Behavioral of Lab8Sim is

component Lab8Design is -- Initialize the component

Port ( Shift\_Mode : in STD\_LOGIC;

Shift\_Direction : in STD\_LOGIC;

Nbr\_of\_Bit : in STD\_LOGIC\_VECTOR (4 downto 0);

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

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

end component;

signal Shift\_Mode : std\_logic; --Signal declarations

signal Shift\_Direction: std\_logic;

signal Nbr\_of\_Bit: std\_logic\_vector (4 downto 0);

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

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

begin

uut: Lab8Design port map (Shift\_Mode, Shift\_Direction, Nbr\_of\_Bit, Data\_In, Data\_Out); --Port map

process

begin

Shift\_Mode <= '0'; --Test Case 1: Logical left shift

Shift\_Direction <= '0';

Nbr\_of\_Bit <= "00100";

Data\_In <= x"79abcdef";

wait for 20 ns;

Nbr\_of\_Bit <= "01000";

Data\_In <= x"a79bcdfe";

wait for 20 ns;

Shift\_Direction <= '1'; --Test Case 2: Logical right shift

Nbr\_of\_Bit <= "00100";

Data\_In <= x"79abcdef";

wait for 20 ns;

Nbr\_of\_Bit <= "01000";

Data\_In <= x"a79bcdfe";

wait for 20 ns;

--Performing arithmetic shifts

Shift\_Mode <= '1'; --TestCase 3, arithmetic shift left

Shift\_Direction <= '0';

Nbr\_of\_Bit <= "00100";

Data\_In <= x"79abcdef";

wait for 20 ns;

Nbr\_of\_Bit <= "01000";

Data\_In <= x"a79bcdfe";

wait for 20 ns;

Shift\_Direction <= '1'; --TestCase 4: arithmetic shift right

Nbr\_of\_Bit <= "00100";

Data\_In <= x"79abcdef";

wait for 20 ns;

Nbr\_of\_Bit <= "01000";

Data\_In <= x"a79bcdfe";

wait;

end process;

end Behavioral;

**4.4 Waveform / Results**

The waveform below shows that the program 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. For example, at 100ns, the input a79bcdfe was shifted left to 9bcdfe00.

Graphical user interface

Description automatically generated

**Figure 3:** Shifter circuit Waveform.

# Conclusion

In this lab, we were able to successfully code a 32-bit shifter 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 the arithmetic and logical shifts work inside a computer, which can be seen in the waveform due to the correct output being produced for what we inputted.