**Mux Design and Extension Unit Design**

**CENG 3151**

Submitted by

Michael Lanford

1816337

Computer Engineering

University of Houston-Clear Lake

Houston, Texas 77058

March 20, 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**……………………………………………………………4**

5.2 Schematics**……………………………………………………………………………….....6**

5.3 Testbench**…………………………………………………………………………………....7**

5.4 Waveform / Results**………………………………………………………………………...10**

[**6** **Conclusion** 11](#_Toc126060768)

**Abstract**

A mux is a device that selects between analog or digital input signals and forwards the selected input to a single output. It takes in many inputs and selects the correct one using a rotary, which will then be sent to the output. Sign extension units replicate the most significant bit loaded into the remaining bits of a binary number. Sign extension is used to increase the number of bits representing a value while preserving the original sign and value.

# Introduction

For this project, we will be using Xilinx Vivado to build and test a Mux circuit and a sign extender circuit that each will accept some input and produce some output.

1. **Requirements**

Design a 32-bit 2-to-1 mux with 3 inputs: source selection, source A, and source B. This circuit will output a 32-bit vector of data. Design a 16-bit extension unit that has 2 inputs: Sign\_Ctrl and Data\_In. This circuit will output a 32-bit sign extended or zero extended number. The figures of the circuits can be seen below:

Diagram

Description automatically generated

**Figure 1:** Diagram for the 32-bit 2-to-1 Mux to be designed.

Text, whiteboard

Description automatically generated

**Figure 2:** Diagram for the 16-bit to 32-bit Extender to be designed.

1. **Prelab**

For this prelab, we were required to write about the working of a Mux and the sign extension unit in computer architecture.

A mux is a device that selects between analog or digital input signals and forwards the selected input to a single output. It takes in many inputs and selects the correct one using a rotary, which will then be sent to the output. Sign extension units replicate the most significant bit loaded into the remaining bits of a binary number. Sign extension is used to increase the number of bits representing a value while preserving the original sign and value.

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 Mux circuit and added the necessary inputs and outputs to it. We then coded the source selection conditionals for the Mux. 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. For the Extender 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 conditional process that took in the Sign\_Ctrl and Data\_In inputs. 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**

------Part 1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

--Input and Output Declarations

entity Lab7Part1Design is

Port ( S : in STD\_LOGIC;--2 to 1 Mux so we have either 0 or 1 on Select line

A : in STD\_LOGIC\_VECTOR (31 downto 0);--32 bit input

B : in STD\_LOGIC\_VECTOR (31 downto 0);--32 bit input

Y : out STD\_LOGIC\_VECTOR (31 downto 0));--32 bit output

end Lab7Part1Design;

architecture Behavioral of Lab7Part1Design is

begin

with S select --Conditional statement for the Source Selection

Y <= A when '0' ,--When S = 1, we output the data from A

B when others; --When S = 0, we output the data from B

end Behavioral;

------Part 2

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

--Input and Output declarations

entity Lab7Part2Design is

Port ( Sign\_Ctrl : in STD\_LOGIC;

Data\_In : in STD\_LOGIC\_VECTOR (15 downto 0);

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

end Lab7Part2Design;

architecture Behavioral of Lab7Part2Design is

begin

process(Sign\_Ctrl, Data\_In) --Process for the sign control signal

begin

if(Sign\_Ctrl = '0') --Sign\_Ctrl 0 means 0 extension, whereas 1 means Signed extension

then Data\_Out <= std\_logic\_vector(resize (unsigned(Data\_In), Data\_Out'length));--Using the resize function, we can zero extend

else

Data\_Out <= std\_logic\_vector(resize (signed(Data\_In), Data\_Out'length));--Using the signed resize, we can sign extend our values

end if;

end process;

end Behavioral;

**4.2 Schematics**

**A picture containing chart

Description automatically generated**

**Figure 3:** Mux circuit.

A picture containing timeline

Description automatically generated

**Figure 4:** Extender circuit.

**4.3 Testbench**

--Part 1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Lab7Part1Sim is

-- Port ( );

end Lab7Part1Sim;

architecture Behavioral of Lab7Part1Sim is

component Lab7Part1Design is --Instantiate the component

Port ( S : in STD\_LOGIC;

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

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

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

end component;

signal S: std\_logic := '0'; --Signal declarations

signal A, B: std\_logic\_vector (31 downto 0) := (others => '0');

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

begin

uut: Lab7Part1Design Port Map(S => S, A => A, B => B, Y => Y); --Port maps

stim\_proc: process --Test case Process

begin

S <= '0'; --Test Case: Select Channel 1

A <= x"AAAA\_AAAA";

B <= x"BBBB\_BBBB";

wait for 20 ns;

S <= '0'; --Test Case 2: Value Change on Channel 1

A <= x"1234\_5678";

B <= x"FFFF\_FFFF";

wait for 20 ns;

S <= '1'; --Test Case 3: Select channel 2

A <= x"CAFE\_BABE";

B <= x"DEAD\_BEEF";

wait for 20 ns;

S <= '1'; --Test Case 4: value change on channel 2

A <= x"9876\_5432";

B <= x"BABE\_CAFE";

wait for 20ns;

S <= '0'; --Test Case 5: Select channgel 1 with value change on channel 1

A <= x"1234\_ABCD";

B <= x"BABE\_CAFE";

wait;

end process;

end Behavioral;

--Part 2

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Lab7Pt2TestBench is

-- Port ( );

end Lab7Pt2TestBench;

architecture Behavioral of Lab7Pt2TestBench is

component Lab7Part2Design is --Instantiate the component

Port ( Sign\_Ctrl : in STD\_LOGIC;

Data\_In : in STD\_LOGIC\_VECTOR (15 downto 0);

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

end component;

signal Sign\_Ctrl: std\_logic := '0'; --Signal declarations

signal Data\_In: std\_logic\_vector (15 downto 0) := (others => '0');

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

begin

uut: Lab7Part2Design port map (Sign\_Ctrl => Sign\_Ctrl, Data\_In => Data\_In, Data\_Out => Data\_Out); -- Port maps

stim\_proc: process --Test case process

begin

Sign\_Ctrl <= '0'; --Test Case 1: Zero Extension

Data\_In <= x"0ABD";--All cases here should have leading 0's

wait for 20 ns;

Data\_In <= x"7ABD";

wait for 20 ns;

Data\_In <= x"FFFF";

wait for 20 ns;

Data\_In <= x"ABCD";

wait for 20 ns;

Sign\_Ctrl <= '1'; --Test Case 2: Sign Extension

Data\_In <= x"0ABD";--Expect leading 0's

wait for 20 ns;

Data\_In <= x"7ABD";--Expect leading 0's

wait for 20 ns;

Data\_In <= x"FFFF";--Expect leading 1's

wait for 20 ns;

Data\_In <= x"ABCD";--Expect leading 1's

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. For example, at 60ns on Figure 5 the source selection bit is 1 so the value at source B should be outputted accordingly and it is. Also, at 40ns on Figure 6 the Sign\_Ctrl bit is 0 so the Data\_In value should’ve been zero extended and it was.

Graphical user interface, timeline

Description automatically generated

**Figure 5:** Mux circuit Waveform.

Graphical user interface, application

Description automatically generated

**Figure 6:** Extender circuit Waveform.

# Conclusion

In this lab, we were able to successfully code a Mux and bit extender 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 a multiplexer and sign/zero extender works inside a computer, which can be seen in the waveforms due to the correct output being produced for what we inputted.