# COEN 313 Lab 3 (UR-X)

by

Mamadou Kaba (27070179)

Performed on

March 15th, 2023

"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"

## Objective

The main objective of the third lab is to familiarize with the design of combinational processes by using VHDL combinational processes. It is also to familiarize with some useful command line options to the vcom command.

## Introduction

The main purpose of this lab is to become experienced in using VHDL combination processes and design a combinational logic circuit. The circuit that will be designed in this lab essentially converts a 4-bit signed integer in sign-magnitude representation into two's complement notation. In sign-magnitude notation, the most significant bit represents the sign of the integer, with 0 denoting a positive integer and 1 denoting a negative integer. The remaining bits represent the magnitude of the integer. Positive numbers will be represented the same in both notations. On the other hand, negative numbers are represented differently.

The code will then be simulated on the terminal by creating a do file to generate a truth table of the result of the equation. Finally, using the software Vivado, the 2s complement will be programmed to the Nexys board to be demonstrated.

Here is a truth table of what to be expected for the 2s complement demonstration on the Nexys board using the LED outputs:

| LED out Nexys-A7100T | J15 | L16 | M13 | R15 | H17 | K15 | J13 | N14 |
|---|---|---|---|---|---|---|---|---|
| Inputs/outputs | Sign_mag[0] | Sign_mag [1] | Sign_mag[2] | Sign_mag[3] | 2s_comp[0] | 2s_comp[1] | 2s_comp[2] | 2s_comp[3] |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

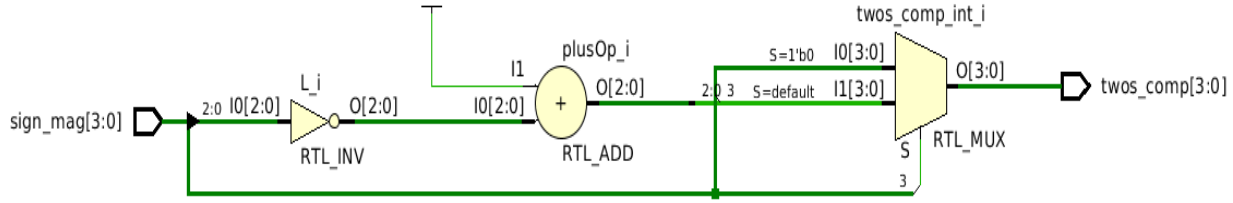
## Results

Converter VHDL

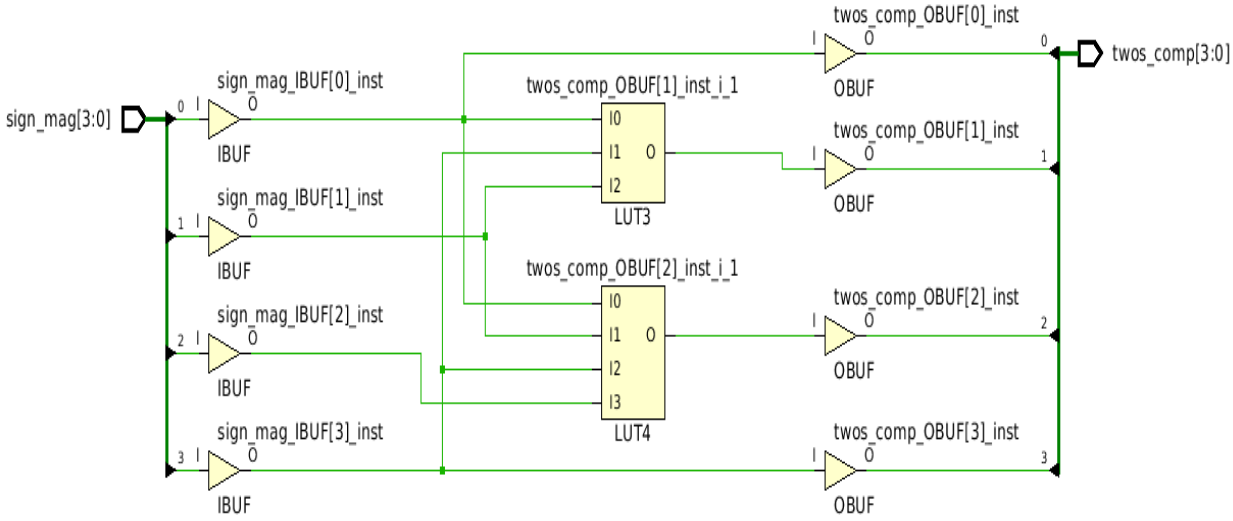
Below is the code that convert the values into 2s complement.

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity converter is
port( sign_mag : in std_logic_vector(3 downto 0) ;
twos_comp : out std_logic_vector(3 downto 0) );
end;

architecture using_variable of converter is
signal twos_comp_int : std_logic_vector(3 downto 0);
begin

process(sign_mag)
variable magnitude : std_logic_vector(2 downto 0);

begin
magnitude := sign_mag(2 downto 0);
magnitude := not magnitude ;
magnitude := magnitude + "001";
if sign_mag(3) = '0' then
        twos_comp_int <= sign_mag ;
else
        twos_comp_int <= sign_mag(3) & magnitude ;
end if;
end process;

twos_comp <= twos_comp_int ;

end using_variable;
```

Elaborated Schematic




Implemented Design




The 2s complement converter code implemented and simulated correctly. The waveform below shows that all possible input values were tested and it perfectly matches the truth table that was provided.

## Questions

1. For this VHDL code, the signal “stone” is defined on both sides of the signal assignment operator `keith <= stone ; -- tout le monde sais que Keith == stone`. When the output signal is used as input in the assignment statement, then a closed feedback loop is formed during synthesis which leads to creation of internal state.

2. When a signal is read from within a combinational process but does not appear in the process sensitivity list, it means that the outputs is not updated until another signal in the process sensitivity list updates later than the missing one.

3. 2s complement using variables

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity converter is
port( sign_mag : in std_logic_vector(3 downto 0) ;
twos_comp : out std_logic_vector(3 downto 0) );
end converter;

architecture sign_magnitude of converter is
begin
converting_process: process (sign_mag)
variable negated_magnitude,second_compliment: std_logic_vector(2 downto 0);
variable temporary_magnitude: std_logic_vector (3 downto 0);
begin

negated_magnitude := not sign_mag(2 downto 0); --first compliment
second_compliment := negated_magnitude + "001"; -- second compliment

if sign_mag(3) = '0' then
temporary_magnitude := sign_mag;
else
temporary_magnitude := '1' & second_compliment;
end if;
twos_comp <= temporary_magnitude;
end process;
end sign_magnitude;
```

2s complement simulation

```
# 6.6g

# vsim -do ../DO/2scomplement.do -c converter
# //  ModelSim SE-64 6.6g May 23 2012 Linux 3.10.0-1160.42.2.el7.x86_64
# //
# //  Copyright 1991-2012 Mentor Graphics Corporation
# //              All Rights Reserved.
# //
# //  THIS WORK CONTAINS TRADE SECRET AND
# //  PROPRIETARY INFORMATION WHICH IS THE PROPERTY
# //  OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS
# //  AND IS SUBJECT TO LICENSE TERMS.
# //
# Loading std.standard
# Loading ieee.std_logic_1164(body)
# Loading ieee.std_logic_arith(body)
# Loading ieee.std_logic_unsigned(body)
# Loading work.converter(using_variable)#1
# do ../DO/2scomplement.do
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand,
#    Time: 0 ns  Iteration: 0  Instance: /converter
# 0000 0000
# 0001 0001
# 0010 0010
# 0011 0011
# 0100 0100
# 0101 0101
# 0111 0111
# 1000 1000
# 1001 1111
# 1010 1110
# 1011 1101
# 1100 1100
# 1101 1011
# 1110 1010
# 1111 1001
#
```

## Conclusion

To conclude, this third lab offered a good learning of the design of combinational processes by using VHDL combinational processes. It also provided a meaningful exercise with some useful command line options to the vcom command.

# Appendix

converter.xdc

```
# Vivado does not support old UCF syntax
# must use XDC syntax


set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [ get_ports { sign_mag[0] }  ] ;
set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [ get_ports { sign_mag[1] }  ] ;
set_property -dict { PACKAGE_PIN M13 IOSTANDARD LVCMOS33 } [ get_ports { sign_mag[2] }  ] ;
set_property -dict { PACKAGE_PIN R15 IOSTANDARD LVCMOS33 } [ get_ports { sign_mag[3] }  ] ;
set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [ get_ports { twos_comp[0] }  ] ;
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [ get_ports { twos_comp[1] }  ] ;
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [ get_ports { twos_comp[2] }  ] ;
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [ get_ports { twos_comp[3] }  ] ;
```