Homework Assignment #2

Name: Erivelton Gualter dos Santos

Win Numer: 143101929

Dr. Janos L. Grantner

WESTERN MICHIGAN UNIVERSITY
ECE 5510 APPLICATION SPECIFIC INTEGRATED CIRCUIT DESIGN
Homework Assignment #2

Task 1:

Construct a **behavioral** VHDL program for **a parity generator** of **7-bit words**. The parity bit (the 8^{th} bit) should be generated such that the **total number of 1s** of the 8-bit word will be an **even number**. Simulate your design and check for functional correctness.

Algorithm of parity generator of 7-bit words

The task of "parity generator of 7-bit words" consists in generate the 8^{th} bit such that the total number of 1s of the 8-bit word will be an even number. To execute this task, we have an input **A** that is a vector with 7 positions and another vector **p** with 8 positions for the output. Therefore, there is an algorithm that counts the number of 1s in the variable A. Then, there is a condition that adds '1' or '0' in the 8^{th} bit to form an even amount of number. I check the number if it is even or odd through modulo operator.

1. *Read the variable A*
2. *Create the variable "count" to count the number of '1s'*
3. *For I in 0 to 6 loop*
4. *If the A(I) = '1' -> count= count +1*
5. *End of loop*
6. *If the modulo of count is equal to 1, add '1' in the* 8^{th}
7. *If the modulo of count is equal to, add '0' in the* 8^{th}
8. *Finally, the vector P is a copy of the vector A, but there will be the last bit with the parity correctly*

Algorithm of parity generator of 7-bits words

For the simulation of this program, I created a file .do to test diverse of differents inputs.

Input	Output
0000000	00000000
0000001	00000011
0110011	01100110
0010100	00101000
0111000	01110001
1000100	10001000

0000001	00000011
1111000	11110000

Table of test

.VHD

parity_vhdl.vhd

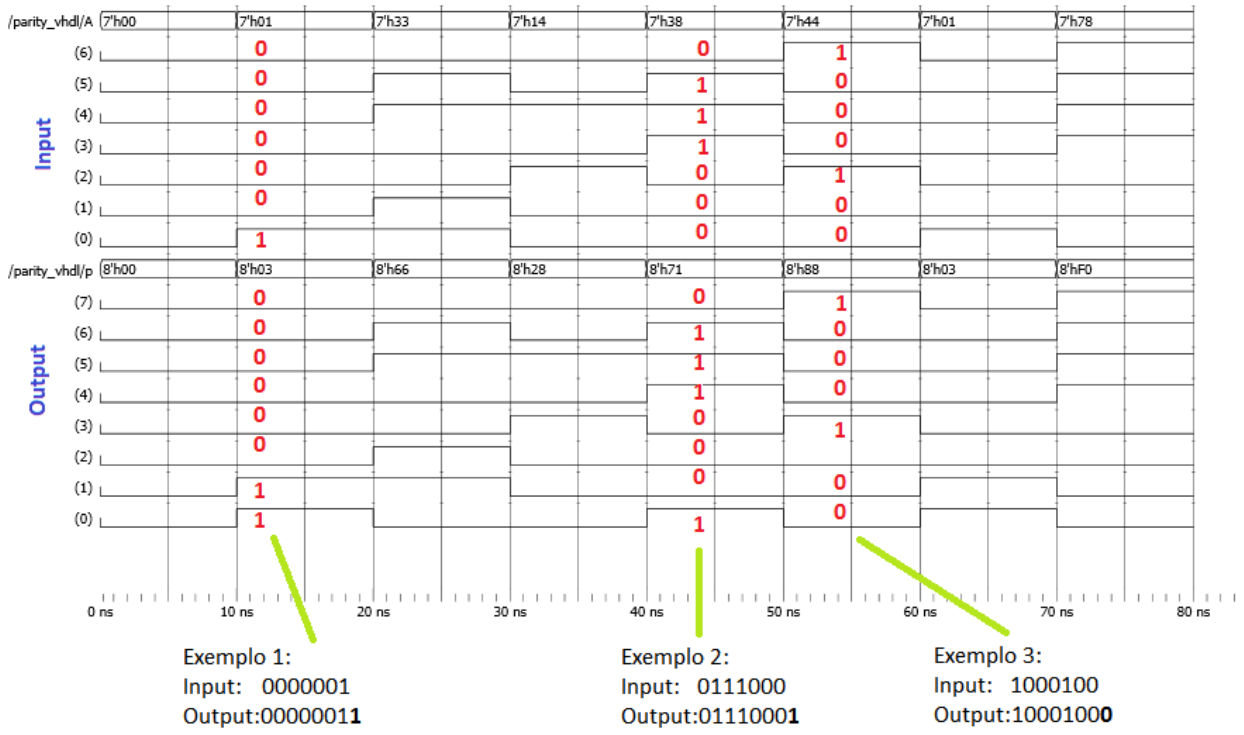
```
1   library IEEE;                       -- Declaration of the library
2   use IEEE.STD_LOGIC_1164.ALL;        -- Declaration of the packages
3   use IEEE.NUMERIC_STD.ALL;
4
5   entity parity_vhdl is                              -- partity_vhdl entity
6      port( A  : in  std_logic_vector(6 downto 0); -- Input:  7-but words
7            p  : out std_logic_vector(7 downto 0));                    -- Output:
    parity bit
8   end parity_vhdl;
9
10  architecture Behavioral of parity_vhdl is
11  begin
12     process(A)
13     variable count: integer;    -- Count how many 1s there are
14
15     begin
16     count:=0;
17     for i in 0 to 6 loop
18        p(i+1) <= A(i);
19        if(A(i) = '1') then
20           count:=count+1;
21        end if;
22     end loop;
23
24     if ( (count mod 2) = 1) then  -- It means that the vector is even
25        p(0) <= '1';
26     else                          -- It means that the vector is odd
27        p(0) <= '0';
28     end if;
29
30     end process;
31  end Behavioral;
32
```

Program .vhd

Simulation

```
#######################################
#      Macro for parity generator     #
#######################################



restart

# Test the wave for the input 0000000
# The answer should be 00000000
force A 0

# Test the wave for the input 0000001
# The answer should be 00000011
run 10ns
force A 2'h1

# Test the wave for the input 0110011
# The answer should be 01100110
run 10ns
force A 2'h33
run 10ns

# Test the wave for the input 0010100
# The answer should be 00101000
force A 2'h14
run 10ns

# Test the wave for the input 0111000
# The answer should be 01110001
force A 2'h38
run 10ns

# Test the wave for the input 1000100
# The answer should be 10001000
force A 2'h44
run 10ns

# Test the wave for the input 0000001
# The answer should be 00000011
force A 2'h1
run 10ns

# Test the wave for the input 1111000
# The answer should be 11110000
force A 2'h78
run 10ns
```

Wave



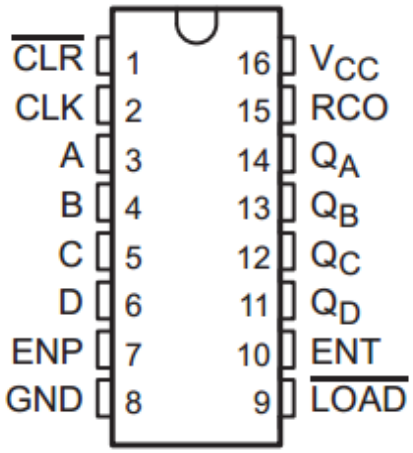
Some examples of the wave

Task 2:

Download the Data Sheets of the **SN74ALS163 Synchronous 4-Bit Binary Counter** from Texas Instruments' Web site. Use the available **CAD tools to design, simulate and compile a functionally** equivalent circuit on your **Xilinx Spartan-6 chip**. However, downloading the bit file to your Nexys 3 Board is **NOT** required.

Algorithm of SN74ALS163 Synchronous 4-Bit Binary Counter

The task elaborates the functionally equivalent circuit of SN74ALS163 Synchronous 4-Bit Binary Counter. To execute this task, we have some inputs and outputs. Look the figure below.



SN74ALS163

The counter can be present to any number between 0 and 15. Furthermore, it can start with any number between these numbers. It is possible because we can set up a low level at the load (LOAD). Consequently, the initial count will be the value in the input A, B, C, D. The CLR input when active in low can clear the count to 0000. Also, we have the inputs ENP and ENT that are conditional to count because both must be high to count and ENT is a conditional to enable RCO. RCO, always produces a high-level pulse while the count is 15.

I developed a finite state machine to reproduce the functionally equivalent circuit of SN74ALS163. To test the operation, I created a file .do to test the correct operation.

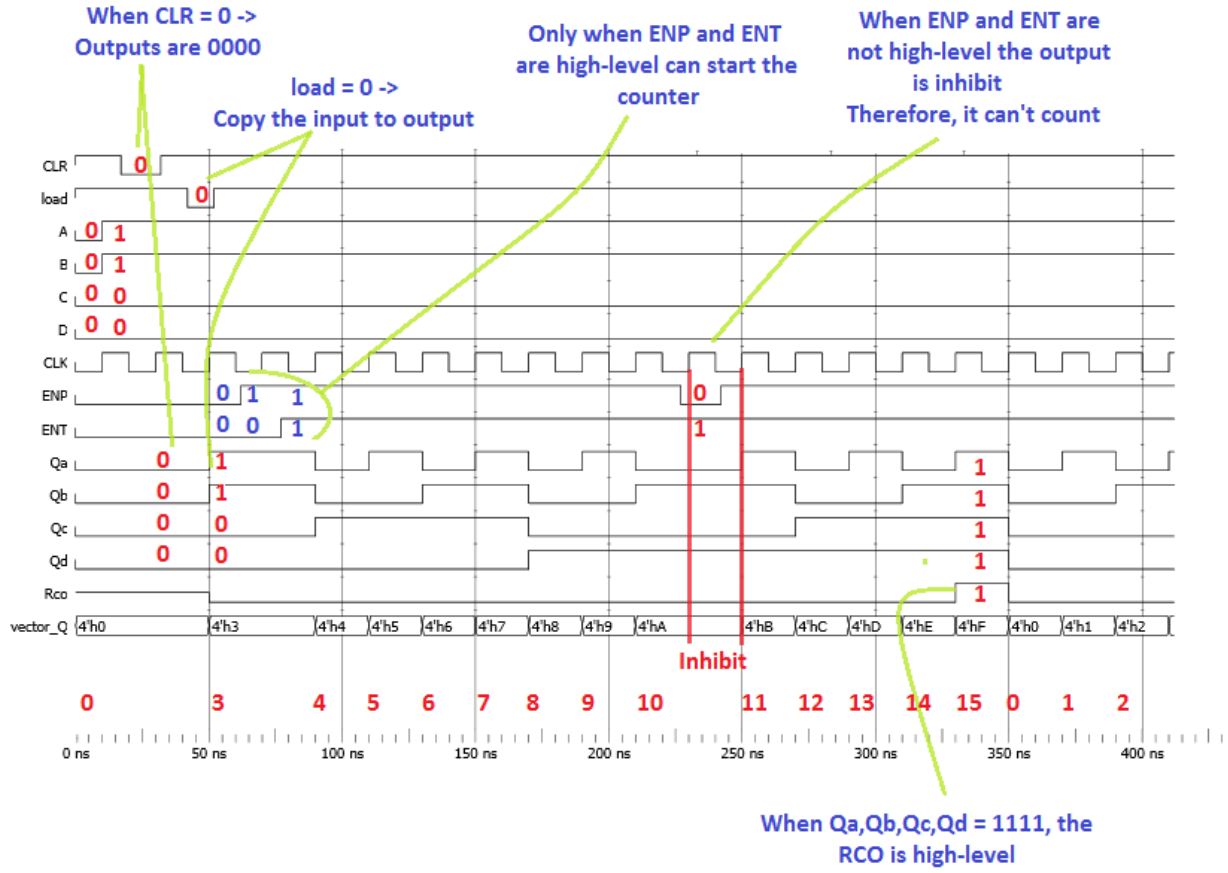
- I checked the CLR. When it is the low-level, the output must be 0000;
- I checked the LOAD. When the load is in the low-level, the input: A, B, C, and D is copied to the output: Qa, Qb, Qc, and Qd.
- I checked the RCO because it should be high-level when the output is 1111 = “15”.
- I checked the inputs: ENP and ENT because the counter just can work when the ENP and ENT if high-level. In case it doesn’t happen, the output cannot change.
- Finally, checked the all counter states.

VHDL

The file .vhd is attached.

Simulation - Wave

Look the wave below:



Simulation

```
####################################
#        Macro for the SN7ALS163   #
####################################
restart
# Generates de clock with T = 20ns
force CLK 0 0, 1 10ns -r 20ns

# Initial Information
force A 0
force B 0
force C 0
force D 0
force ENP 0
force ENT 0
force load 1
force CLR 1
# Initial condition for A, B, C, D
run 10ns
```

```
force A 1
force B 1
force C 0
force D 0
# Clear test
run 7ns
force CLR 0
run 15ns
force CLR 1

# Load of the initial condition
run 10ns
force load 0
run 10ns
force load 1
# Test of the inputs ENP and ENT
run 10ns
force ENP 1
run 15ns
force ENT 1

# Maintain the count
run 150ns
# Test the Inhibit
force ENP 0
run 15ns
force ENP 1
run 170ns
```

Observation: All files are attached