Exercise No. 4	
Odd Parity Generator	
Course Code: CPE 402	**Program: BSCPE**
Course Title: Advanced Logic Circuits	**Date Performed:**
Section:	**Date Submitted:**
Leader: 1.	**Instructor:** Engr.Sampedro
Members: 2. Marjorie D	
3.	
4.	
5.	

1. Objective(s):

The activity aims to simulate an odd parity bit scheme using VHDL codes.

2. Intended Learning Outcomes (ILOs)

The students should be able to:

2.1 Create a design, test bench and an EPWave of odd parity generator.

2.2 Compare the VHDL process codes to previous experiment.

2.3 Debug VHDL codes and create a truth table of odd parity generator in 10 ns.

3. Discussion:

Parity Generator

A parity generator is a combinational logic circuit that generates the parity bit in the transmitter. The sum of the data bits and parity bits can be even or odd. In even parity, the added parity bit will make the total number of 1s an even amount whereas in odd parity the added parity bit will make the total number of 1s odd amount. The basic principle involved in the implementation of parity circuits is that sum of odd number of 1s is always 1 and sum of even number of 1s is always zero. Such error detecting and correction can be implemented by using Ex-OR gates (since Ex-OR gate produce zero output when there are even number of inputs).

Odd Parity Generator

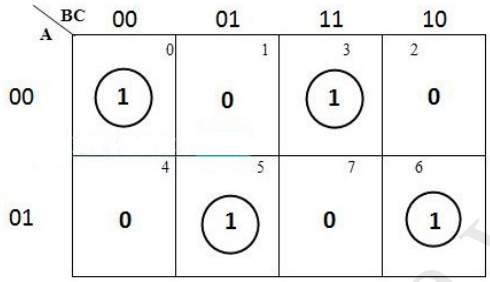
Let us consider that the 3-bit data is to be transmitted with an odd parity bit. The three inputs are A, B and C and P is the output parity bit. The total number of bits must be odd in order to generate the odd parity bit.

In the given truth table below, 1 is placed in the parity bit in order to make the total number of bits odd when the total number of 1s in the truth table is even.

Table 4-1 Odd Parity Generator Truth Table

3-bit message			Odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The truth table of the odd parity generator can be simplified by using K-map

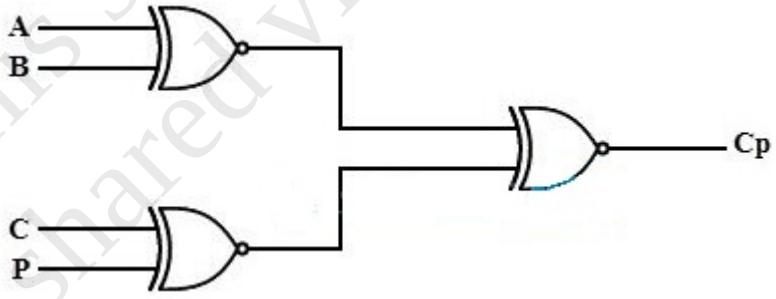


The output parity bit expression for this generator circuit is obtained as

$$P = A \oplus B \text{ Ex-NOR } C$$

The above Boolean expression can be implemented by using one Ex-OR gate and one Ex-NOR gate in order to design a 3-bit odd parity generator.

The logic circuit of this generator is shown in below figure, in which two inputs are applied at one Ex-OR gate, and this Ex-OR output and third input is applied to the Ex-NOR gate, to produce the odd parity bit. It is also possible to design this circuit by using two Ex-OR gates and one NOT gate.



4. Resources:

Computer System with internet connection

5. Procedure:

1. Create a VHDL file for Odd Parity Generator.
2. Create the VHDL design source code of Odd Parity Generator.
3. Create a test bench source code.
4. Generate the EPWave in 10 ns of the Odd Parity Generator.
5. And show the results on the space provided.

5.2.1 Odd Parity Generator VHDL design

```
Package anu is
constant m: integer :=8;
type input is array (0 to m-1) of bit;
end anu;

library ieee;
use ieee.std_logic_1164.all;
use Work.anu.all;

entity Parity_Generator1 is
port ( input_stream : in  input;
       clk : in std_logic ;
       parity : out bit );
end Parity_Generator1;

architecture odd of Parity_Generator1
is
begin
P1: process
Variable odd : bit ;
begin
wait until clk'event and clk = '1';
odd := '0';
for I in0 to m-1 loop
odd := odd xor input_stream (I);
end loop;
parity <= odd;
end process;
end odd;
```

5.3.1 Odd Parity Generator Test Bench

```
entity ODD_PARITY_TB is
end;
library ieee;
use ieee.std_logic_1164.all;
use WORK.anu.all;
architecture OP_TB_ARCH
ofODD_PARITY_TB is
component Parity_Generator1
port (input_stream : in input;
clk : in std_logic;
parity : out bit );
end component;
signal input_stream : input;
signal clk :std_logic;
signal parity :bit ;
begin
U1: Parity_Generator1
port map( input_stream,
          clk,
          parity => parity);
input1 : process (clk)
begin
if clk <= 'U' then clk <= '0' after 1
ns;
else clk <= not clk after 1 ns;
end if;
end process;
input2: process (input_stream)
begin
input_stream <= "10100110" after
1 ns,
"01111100" after 2 ns;
end process;
```

	end OP_TB_ARCH; configuration cfg_op of ODD_PARITY_TB is for OP_TB_ARCH end for; end cfg_op;

5.4.1 Show the EPWave of the Odd Parity Code Generator

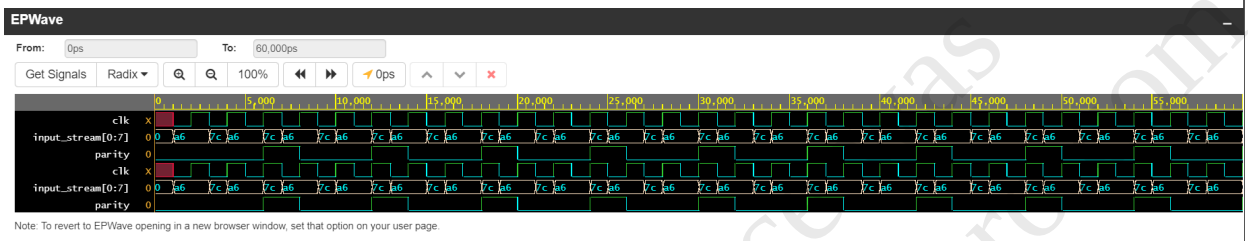


Figure 1: EPWave of the Odd Parity Code Generator at 0ns. Clk=x,input_stream[0:7]=0,parity=0,

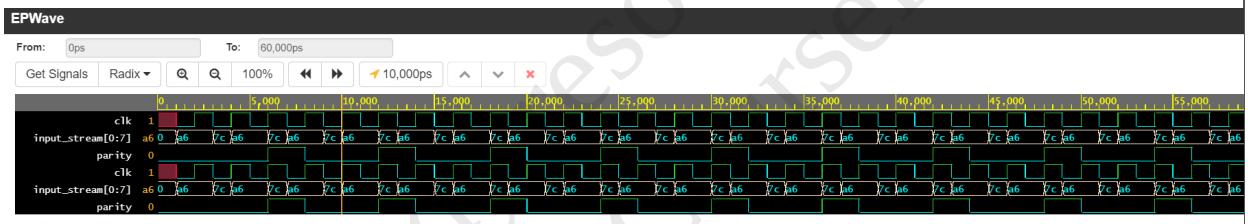


Figure 2: EPWave of the Odd Parity Code Generator at 10ns. Clk=1,input_stream[0:7]=a6,parity=0,

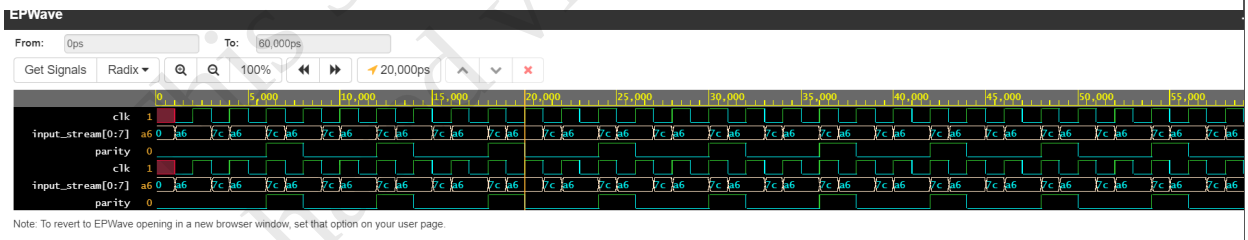


Figure 3: EPWave of the Odd Parity Code Generator at 20ns. Clk=1,input_stream[0:7]=a6,parity=0,

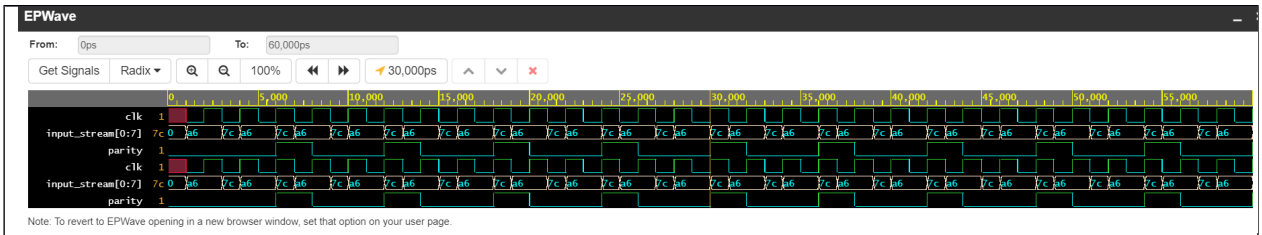


Figure 4: EPWave of the Odd Parity Code Generator at 30ns. Clk=1, input_stream[0:7]=7c,parity=1.

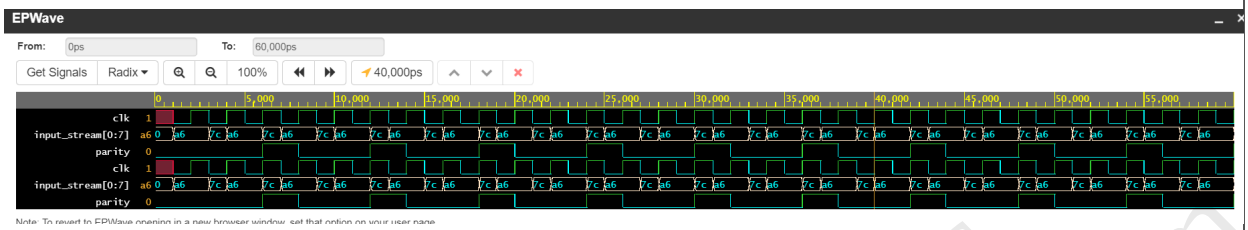


Figure 5: EPWave of the Odd Parity Code Generator at 40ns. Clk=1, input_stream[0:7]=a6,parity=0.

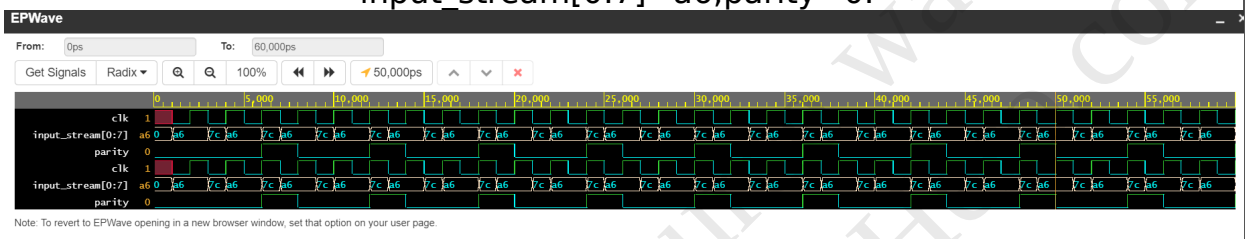


Figure 6: EPWave of the Odd Parity Code Generator at 50ns. Clk=1, input_stream[0:7]=a6,parity=0.

6. Questions:

a. How does the VHDL design and test bench differ from each other's?

On VHDL design uses type input is array (0 to m-1) of bit and an constant 8 integer while on the test Bench uses 8bit for if case.

b. Explain the behavior of the odd parity generator's design code.

The behavior of the odd parity generator's when the sum of the data bits and parity bits can be even or odd. In even parity, the added parity bit will make the total number of 1s an even amount whereas in odd parity the added parity bit will make the total number of 1s odd amount.

7. Data and Results:

(Show the equivalent truth table of each logic gates in 10nano seconds)

Note: Write or indicate the input and output variable used in the program and give analysis sentence for each table. You may change the number of column and rows of each table according to the number of input and output used in the codes.

Table 4-2 Odd Parity Generator Truth Table in 10 ns

Every 10 ns	Input(clk)	Input(Input_Stream)	Output(parity)
10ns	1	A6	0
20ns	1	A6	0
30ns	1	7c	1
40ns	1	A6	0
50ns	1	A6	0

Conclusion:

Marjorie D

- I therefore conclude that in this activity Parity Generator is a combinational logic circuit that generates the parity bit in the transmitter and the Odd Parity Generator is the combinational circuit whose output is always dependent upon the given input data. If there is an even number of 1’s then only parity bit is added to make the binary code into an odd number of 1’s.

8. Assessment (Rubric for Laboratory Performance):