**Homework Assignment #2**

**Name:** Erivelton Gualter dos Santos

**Win Numer:** 143101929

Dr. Janos L. Grantner

Task 1:

Construct a **behavioral** VHDL program for **a parity generator** of **7-bit words**. The parity bit (the 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 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 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*
7. *If the modulo of count is equal to, add ‘0’ in the*
8. *Finally, the vector P is a copy of the vector A, but there will be the last bit with the parity correctly*

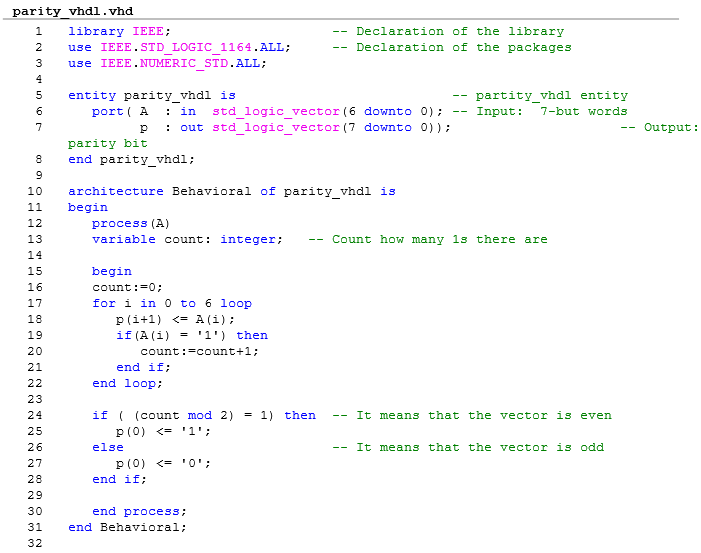
**Algorithm of parity generator of 7-bits words**

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



**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

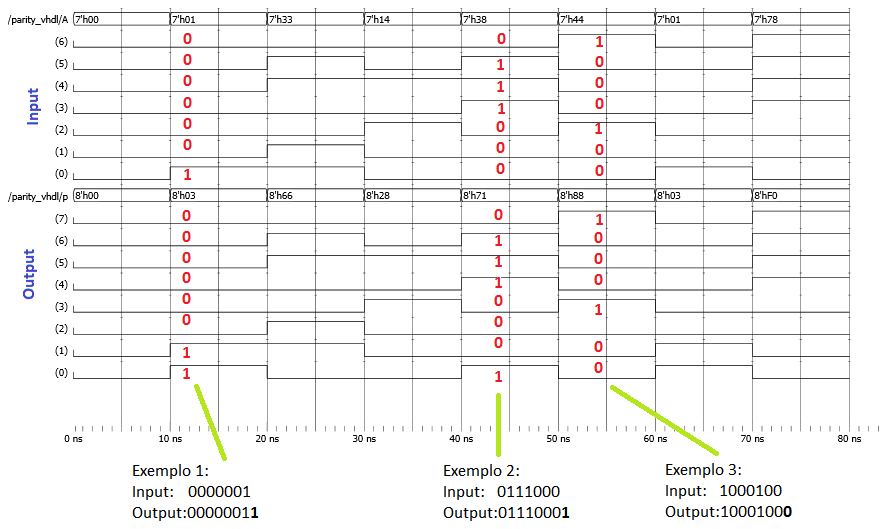
# Test the wave for the input 1111000

# The answer should be 11110000

force A 2'h78

run 10ns

# Wave



**Some examples of the wave**

# File .ucf

# PlanAhead Generated physical constraints

NET "A[6]" LOC = T10;

# PlanAhead Generated IO constraints

NET "A[6]" IOSTANDARD = LVCMOS33;

NET "A[5]" IOSTANDARD = LVCMOS33;

NET "A[4]" IOSTANDARD = LVCMOS33;

NET "p[7]" IOSTANDARD = LVCMOS33;

NET "p[6]" IOSTANDARD = LVCMOS33;

NET "p[5]" IOSTANDARD = LVCMOS33;

NET "p[4]" IOSTANDARD = LVCMOS33;

NET "p[3]" IOSTANDARD = LVCMOS33;

NET "p[2]" IOSTANDARD = LVCMOS33;

NET "p[1]" IOSTANDARD = LVCMOS33;

NET "p[0]" IOSTANDARD = LVCMOS33;

NET "A[0]" IOSTANDARD = LVCMOS33;

NET "A[1]" IOSTANDARD = LVCMOS33;

NET "A[2]" IOSTANDARD = LVCMOS33;

NET "A[3]" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "A[5]" LOC = T9;

NET "A[4]" LOC = V9;

NET "A[3]" LOC = M8;

NET "A[2]" LOC = N8;

NET "A[1]" LOC = U8;

NET "A[0]" LOC = V8;

NET "p[7]" LOC = T11;

NET "p[6]" LOC = R11;

NET "p[5]" LOC = N11;

NET "p[4]" LOC = M11;

NET "p[3]" LOC = V15;

NET "p[2]" LOC = U15;

NET "p[1]" LOC = V16;

NET "p[0]" LOC = U16;