README.md

Digital-electronics-1

Laboratory #1

Link to my Digital-electronics-1 repository

Exercise 1: Verification of De Morgan's laws:

$f(c,b,a) = \overline{b} \cdot a + \overline{c} \cdot \overline{b}$

c	b	a	$f(c,b,a)$
0	0	0	1
0	0	1	1
0	1	0	0

c	b	a	$f(c,b,a)$
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$$f(c,b,a)_{NAND} = \overline{\overline{(\overline{b} \cdot a)} \cdot (\overline{\overline{c} \cdot \overline{b}})}$$

c	b	a	$f(c,b,a)_{NAND}$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$$f(c,b,a)_{NOR} = \overline{\overline{(\overline{b} + a)} + (\overline{\overline{c} + \overline{b}})}$$

c	b	a	$f(c,b,a)_{NOR}$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1

c	b	a	$f(c,b,a)_{NOR}$
1	1	0	0
1	1	1	0

VHDL Code:

```
-- Example of basic OR, AND, XOR gates.
-- Nexys A7-50T, Vivado v2020.1, EDA Playground

-- Copyright (c) 2019-2020 Tomas Fryza
-- Dept. of Radio Electronics, Brno University of Technology, Czechia
-- This work is licensed under the terms of the MIT license.

library ieee;               -- Standard library
use ieee.std_logic_1164.all;-- Package for data types and logic operations

-- Entity declaration for basic gates

entity gates is
    port(
        a_i    : in  std_logic;         -- Data input
        b_i    : in  std_logic;         -- Data input
        c_i    : in  std_logic;         -- Data input
        f_base : out std_logic;         -- Output function
        f_NAND : out std_logic;         -- Output function
        f_NOR  : out std_logic          -- Output function
    );
end entity gates;


-- Architecture body for basic gates

architecture dataflow of gates is
begin
        f_base <= ((not b_i) and a_i) or ((not c_i) and (not b_i));
        f_NAND <= ((not b_i) nand a_i) nand ((not c_i) nand (not b_i));
        f_NOR  <= ((not b_i) nor a_i) nor (c_i nor (not b_i));

end architecture dataflow;
```

Waveform #1:

EDA Playground example

Exercise 2: Verification of Distributive laws:

$x \cdot y + x \cdot z = a \cdot (y + z)$

c	b	a	$x \cdot y + x \cdot z$	$a \cdot (y + z)$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

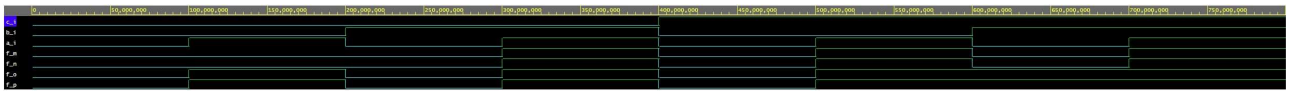
$(x + y) \cdot (x + z) = x + (y \cdot z)$

c	b	a	$(x + y) \cdot (x + z)$	$x + (y \cdot z)$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

VHDL Code:

```
-- Example of basic OR, AND, XOR gates.
-- Nexys A7-50T, Vivado v2020.1, EDA Playground

-- Copyright (c) 2019-2020 Tomas Fryza
-- Dept. of Radio Electronics, Brno University of Technology, Czechia
-- This work is licensed under the terms of the MIT license.


library ieee;               -- Standard library
use ieee.std_logic_1164.all;-- Package for data types and logic operations

-- Entity declaration for basic gates

entity gates is
    port(
        a_i    : in  std_logic;         -- Data input
        b_i    : in  std_logic;         -- Data input
        c_i    : in  std_logic;         -- Data input
        f_m    : out std_logic;         -- Output function
        f_n    : out std_logic;         -- Output function
        f_o    : out std_logic;         -- Output function
        f_p    : out std_logic          -- Output function
    );
end entity gates;


-- Architecture body for basic gates

architecture dataflow of gates is
begin
    f_m <= (a_i and b_i) or (a_i and c_i);
    f_n <= a_i and (b_i or c_i);
    f_o <= (a_i or b_i) and (a_i or c_i);
    f_p <= a_i or (b_i and c_i);

end architecture dataflow;
```

Waveform #2:



EDA Playground example