AnaSampaio13 / Digital-Electronics-1

MIT License

0 stars 0 forks

Star Unwatch

Code Issues 2 Pull requests Actions Projects Wiki Security Insights Sett

main

AnaSampaio13 ... 26 seconds ago

View code

README.md

Digital-Electronics-1

https://github.com/AnaSampaio13/Digital-Electronics-1

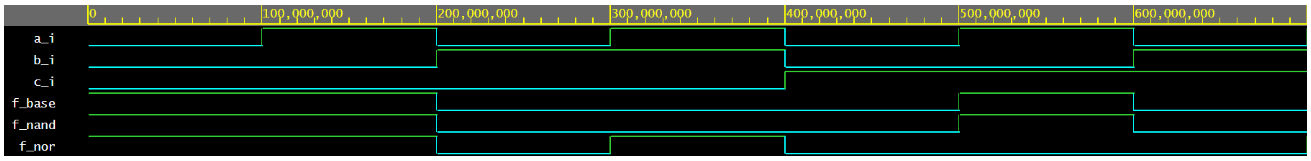
Exercise 1

```
------------------------------------------------------------------------
--
-- Example of basic OR, AND, XOR gates.
```

```vhdl
-- 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 ((not c_i) nor b_i);


end architecture dataflow;



------------------------------------------------------------------------
--
-- Testbench for basic gates circuit.
-- Nexys A7-50T, Vivado v2020.1, EDA Playground
--
-- Copyright (c) 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;
use ieee.std_logic_1164.all;

------------------------------------------------------------------------
-- Entity declaration for testbench
```

```vhdl
------------------------------------------------------------------------
entity tb_gates is
    -- Entity of testbench is always empty
end entity tb_gates;

------------------------------------------------------------------------
-- Architecture body for testbench
------------------------------------------------------------------------
architecture testbench of tb_gates is

    -- Local signals
    signal s_a    : std_logic;
    signal s_b    : std_logic;
    signal s_c    : std_logic;
    signal s_d    : std_logic;
    signal s_e    : std_logic;
    signal s_f    : std_logic;


begin
    -- Connecting testbench signals with gates entity (Unit Under Test)
    uut_gates : entity work.gates
        port map(
            a_i    => s_a,
            b_i    => s_b,
            c_i    => s_c,
            f_base => s_d,
            f_nand => s_e,
            f_nor  => s_f

        );

    --------------------------------------------------------------------
    -- Data generation process
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        s_b <= '0';             -- Set input values and wait for 100 ns
        s_a <= '0';
        s_c <= '0';
        wait for 100 ns;
        s_c <= '0';
        s_b <= '0';
        s_a <= '1';
        wait for 100 ns;
        s_c <= '0';
        s_b <= '1';
        s_a <= '0';
        wait for 100 ns;
        s_c <= '0';
        s_b <= '1';
        s_a <= '1';
        wait for 100 ns;
        s_c <= '1';
        s_b <= '0';
        s_a <= '0';
        wait for 100 ns;
        s_c <= '1';
        s_b <= '0';
        s_a <= '1';
        wait for 100 ns;
        s_c <= '1';
        s_b <= '1';
        s_a <= '0';
        wait for 100 ns;
        s_c <= '1';
        s_b <= '1';
        s_a <= '1';
        wait;                    -- Process is suspended forever
    end process p_stimulus;

end architecture testbench;
```

Waveform #Exercise 1:



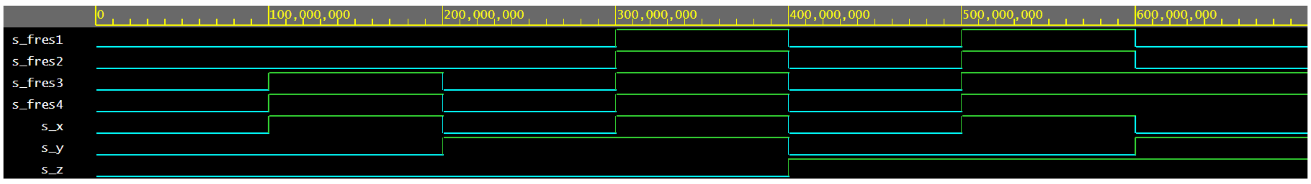
EDA Playground Example

Exercise 2

```
------------------------------------------------------------------------
--
-- 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(
            x_i    : in  std_logic;        -- Data input
            y_i    : in  std_logic;        -- Data input
            z_i    : in  std_logic;        -- Data input
            fres_1 : out std_logic;
            fres_2 : out std_logic;
            fres_3 : out std_logic;
            fres_4 : out std_logic

        );
end entity gates;

------------------------------------------------------------------------
-- Architecture body for basic gates
------------------------------------------------------------------------
architecture dataflow of gates is
begin
    fres_1  <= (x_i and y_i) or (x_i and z_i);
    fres_2  <= x_i and (y_i or z_i);
    fres_3  <= (x_i or y_i) and (x_i or z_i) ;
    fres_4  <= x_i or (y_i and z_i);


end architecture dataflow;



------------------------------------------------------------------------
--
-- Testbench for basic gates circuit.
-- Nexys A7-50T, Vivado v2020.1, EDA Playground
--
-- Copyright (c) 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;
use ieee.std_logic_1164.all;

------------------------------------------------------------------------
-- Entity declaration for testbench
------------------------------------------------------------------------
entity tb_gates is
    -- Entity of testbench is always empty
end entity tb_gates;

------------------------------------------------------------------------
-- Architecture body for testbench
------------------------------------------------------------------------
architecture testbench of tb_gates is

    -- Local signals
    signal s_x    : std_logic;
    signal s_y    : std_logic;
    signal s_z    : std_logic;
    signal s_fres1  : std_logic;
    signal s_fres2  : std_logic;
    signal s_fres3  : std_logic;
    signal s_fres4  : std_logic;

begin
    -- Connecting testbench signals with gates entity (Unit Under Test)
    uut_gates : entity work.gates
        port map(
            x_i    => s_x,
            y_i    => s_y,
            z_i    => s_z,
            fres_1  => s_fres1,
            fres_2  => s_fres2,
            fres_3  => s_fres3,
            fres_4  => s_fres4

        );

    --------------------------------------------------------------------
    -- Data generation process
    --------------------------------------------------------------------
    p_stimulus : process
    begin
        s_y <= '0';             -- Set input values and wait for 100 ns
        s_x <= '0';
        s_z <= '0';
        wait for 100 ns;
        s_z <= '0';
        s_y <= '0';
        s_x <= '1';
        wait for 100 ns;
        s_z <= '0';
        s_y <= '1';
        s_x <= '0';
        wait for 100 ns;
        s_z <= '0';
        s_y <= '1';
        s_x <= '1';
        wait for 100 ns;
        s_z <= '1';
        s_y <= '0';
        s_x <= '0';
        wait for 100 ns;
        s_z <= '1';
        s_y <= '0';
        s_x <= '1';
        wait for 100 ns;
        s_z <= '1';
        s_y <= '1';
        s_x <= '0';
        wait for 100 ns;
        s_z <= '1';
        s_y <= '1';
```

```
            s_x <= '1';
            wait;                       -- Process is suspended forever
        end process p_stimulus;

    end architecture testbench;
```

Waveform #Exercise 2:



EDA Playground Example

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package