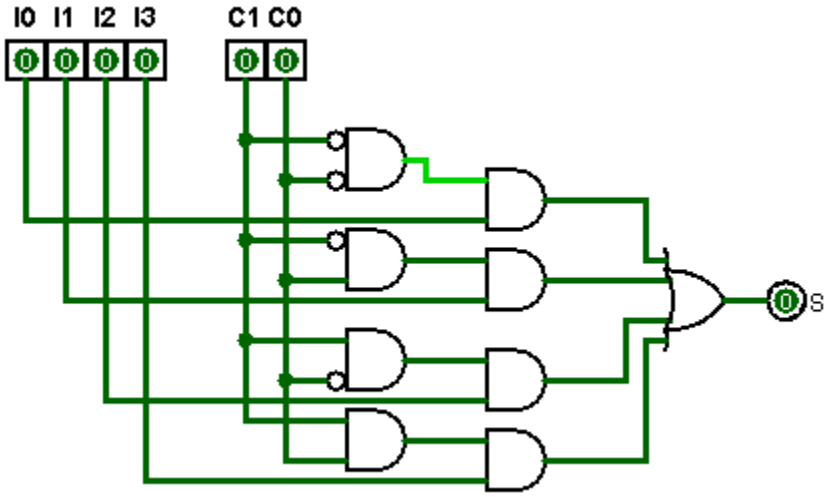
Décimo Roteiro de Laboratório

Multiplexadores e Demultiplexadores

Matricula	Nome

Neste roteiro de laboratório desejamos exercitar nossos conhecimentos acerca de multiplexadores e demultiplexadores. Estes são circuitos digitais fundamentais utilizados em diversos sistemas digitais complexos.

Passo 1: Com base no circuito do MUX4x1 abaixo, levante a expressão booleana correspondente.



S <= _____

Passo 2: Execute o software ModelSim. Crie um novo projeto. A seguir, adicione um novo arquivo ao projeto. Nomeie-o "***mux4x1.vhd***".

Passo 3: Escreva um programa em VHDL que implemente em sua arquitetura a expressão booleana levantada no passo 1.

Passo 4: Compile a descrição de hardware selecionando o item de menu "Compile>> Compile Selected" ou "Compile>> Compile All". Caso a especificação esteja completamente correta, a seguinte mensagem será apresentada:

```
# Compile of mux4x1_comb.vhd was successful.
```

Passo 5: A seguir, crie um novo arquivo clamado "***mux4x1_test.vhd***" que deve conter o código abaixo. Este arquivo refere-se a um test bench e serve para testar a descrição do hardware criada.

```
------------------------------------------------------------
--mux_test.vhd
--
--Desc: Arquivo test bench para a descrição mux.hdl
--
--DDA 10/08/2013
------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
--entidade vazia, pois vamos testar uma já existente
entity mux4x1_comb_test is
end mux4x1_comb_test;

architecture testcomb of mux4x1_comb_test is
  signal i3: std_logic;
  signal i2: std_logic;
  signal i1: std_logic;
  signal i0: std_logic;
  signal c1: std_logic;
  signal c0: std_logic;
  signal s:  std_logic;
component mux4x1_comb
port(   I3: in std_logic;
        I2: in std_logic;
        I1: in std_logic;
        I0: in std_logic;
        C1: in std_logic;
        C0: in std_logic;
        S: out std_logic
    );
end component;
begin
  --instancia a unidade em test (unit under test)
  uut_mux4x1_comb : mux4x1_comb port map(
        I3 => i3,
        I2 => i2,
        I1 => i1,
        I0 => i0,
        C1 => c1,
        C0 => c0,
        S  => s);
-- Corpo do teste dentro de um processo para ser executado
--sequencialmente
  main: process
  begin
    i3 <= '1';
    i2 <= '1';
    i1 <= '1';
    i0 <= '1';
    --simula C = "00" primeiro canal

    c1 <= '0';
    c0 <= '0';
```

```
    wait for 10 ns;
    assert(s='1') report "Erro no canal 0" severity error;

    --wait for 10 us;
    c1 <= '0';
    c0 <= '1';
    wait for 10 ns;
    assert(s='1') report "Erro no canal 1" severity error;

    --wait for 10 us;
    c1 <= '1';
    c0 <= '0';
    wait for 10 ns;
    assert(s='1') report "Erro no canal 2" severity error;

    --wait for 10 us;
    c1 <= '1';
    c0 <= '1';
    wait for 10 ns;
    assert(s='1') report "Erro no canal 3" severity error;

  end process main;
end testcomb;
```

Passo 6: Abra o software Quartus II, crie um novo projeto e mapeie as entradas I0-I3, C1 e C0 para chaves (SW) e a saída S para um led qualquer. Compile e programe a FPGA.

Boa Diversão Pessoal!