Cómo implementar un ‘voter’ en VHDL

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/12/10/como-implementar-un-voter-en-vhdl/

Blog: https://soceame.wordpress.com/

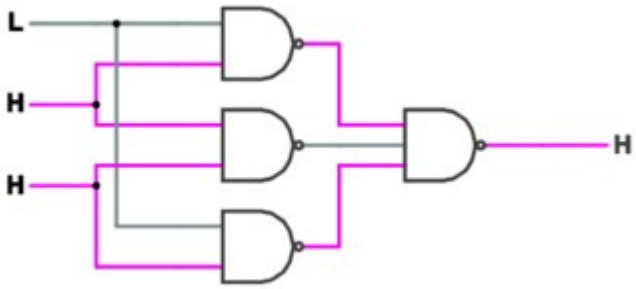
GitHub: https://github.com/DRubioG

Fecha última modificación: 23/02/2025

Si has trabajado con sistema críticos, sobretodo espaciales y aeronáuticos, sabes que existe un concepto llamada TMR (**Triple modular redundancy**). Esto es debido a que se tiene que garantizar que ninguna posible alteración externa altere el trabajo del sistema, por ello se impone la necesidad de tener un sistema que sea seguro ante posibles interferencia. Esto se consigue mediante redundancia de sistemas, pero claro, si redundo los sistemas necesito algo que me permita seleccionar cuál es la salida correcta, por ello se hace necesario un sistema que se encargue de hacer una «votación» entre los sistemas redundantes, para ello se crea un «voter» (dejo más info de la Wikipedia), es «voter» lo único que hace es elegir la salida que dos sistemas han elegido de forma igual.

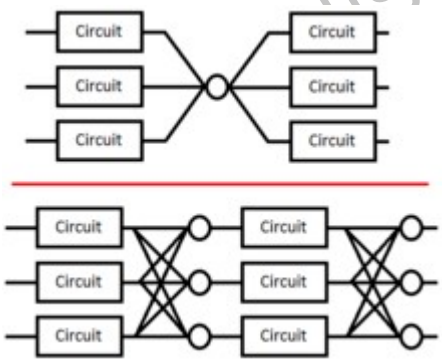
https://en.wikipedia.org/wiki/Triple_modular_redundancy

Un sistema «*voter*» tiene una forma como la siguiente.



Esto es un *voter* de 1 bit, si lo escalas puedes conseguir el *voter* de un bus.

Además, el *voter* también es susceptible de generar errores, bien, pues para evitar los posibles errores del *voter*, el *voter* también tiene que ser redundante.

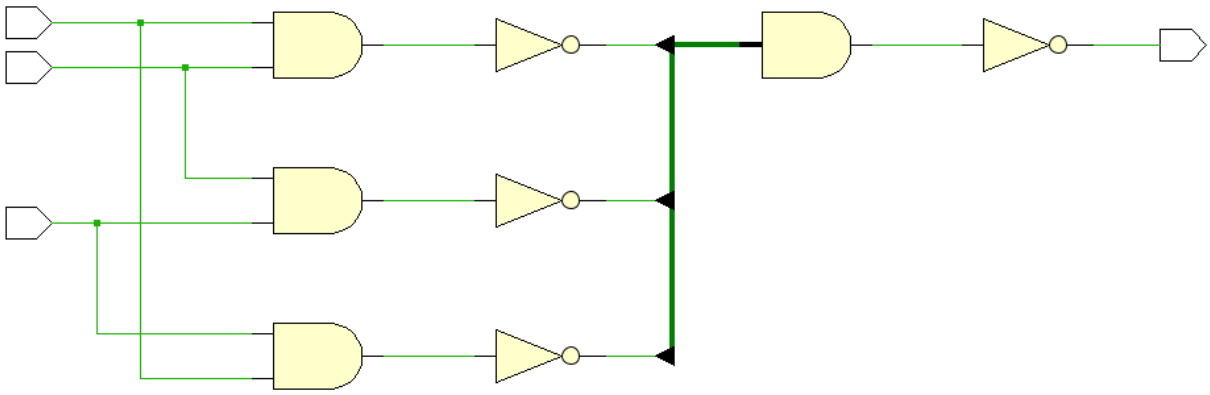


Implementación

Si lo miras, son 4 NANDs en un esquema, por lo que te puedes sentir tentado a utilizar un esquema de VHDL normal con algún apaño.

Pero claro, **VHDL no permite hacer una NAND de tres señales de un golpe**, por lo que se te puede ocurrir la idea de dividir la salida en dos NANDs, una para A y B, y otra para esta salida anterior y C. Bien, pues **NO LO HAGAS**, porque no vas a conseguir el resultado deseado.

La única forma de que un sintetizador de VHDL te genere el modelo de un *voter* válido es utilizando la función ***nand_reduce*** de la librería *std_logic_misc*.



Para que se pueda ver mejor te dejo un ejemplo de implementación en VHDL utilizando un sistema combinacional.

Voter de 1 bit

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_misc.all;

entity voter_bit is
    port(
        a_bit : in std_logic;
        b_bit : in std_logic;
        c_bit : in std_logic;
        Y_bit : out std_logic
    );
end entity;

architecture arch_voter_bit of voter_bit is

signal out1, out2, out3 : std_logic;
signal out_array : std_logic_vector(2 downto 0);

begin

    out1 <= a_bit nand b_bit;
    out2 <= b_bit nand c_bit;
    out3 <= c_bit nand a_bit;

    out_array <= out1 & out2 & out3;

    Y_bit <= nand_reduce( out_array );
```

```
end architecture;
```

Voter de un bus

Para ampliar el *voter*, lo único que hay que hacer es instanciar tantas veces como se quiera el *voter* de 1 bit. Para ello se utiliza la estructura *generate* de VHDL.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity voter is
    generic (
        N : integer := 3
    );
    Port (
        a : in std_logic_vector(N-1 downto 0);
        b : in std_logic_vector(N-1 downto 0);
        c : in std_logic_vector(N-1 downto 0);
        Y : out std_logic_vector(N-1 downto 0);
        error : out std_logic
    );
end voter;

architecture arch_voter of voter is
component voter_bit is
    port(
        a_bit : in std_logic;
        b_bit : in std_logic;
        c_bit : in std_logic;
        Y_bit : out std_logic
    );
end component;

signal Y_aux : std_logic_vector(N-1 downto 0);

begin

impl_voter : for i in 0 to N-1 generate
impl_voter_bit : voter_bit
        port map(
            a_bit => a(i),
            b_bit => b(i),
            c_bit => c(i),
            Y_bit => Y_aux(i)
        );
    end generate;

    Y <= Y_aux;

end arch_voter;
```


Ejemplos anteriores

Para conseguir los ejemplos anteriores: https://github.com/DRubioG/Voter_VHDL