¿Cómo hacer una AND, OR, XOR, NAND, etc de todos los bits de una señal en VHDL?

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/05/18/como-hacer-una-and-or-xor-nand-etc-de-todos-los-bits-de-una-senal-en-vhdl/

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

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

Una situación muy frecuente es cuando se quiere realizar una operación binaria a una señal de varios bits en VHDL, como una OR o una AND de todos los bits de una señal (o de parte de ella).

El método general es recurrir a realizar la operación bit a bit, lo que puede dejar partes del código bastante largas. Ej:

```
a <= "001010";

b <= a(5) or a(4) or a(3) or a(2) or a(1) or a(0);
c <= a(5) and a(4) and a(3) and a(2) and a(1) and a(0);
```

Otra opción es recurrir a un bucle for con una variable (pero esto puede volver muy complejo el desarrollo, debido a que las variables hay que inicializarlas para evitar errores):

```
variable b_aux, c_aux : std_logic_vector(5 downto 0);
begin
...
 for i in a'range loop
   b_aux := b_aux or a(i);
   c_aux := c_aux and a(i);
 end loop;
...
```

Solución

Bien, para solucionar esta situación, existe dentro de una librería llamada «***std_logic_misc***» con varias funciones que realizan estos cálculos.

Para usar esta librería se tiene que cargar la librería al principio:

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

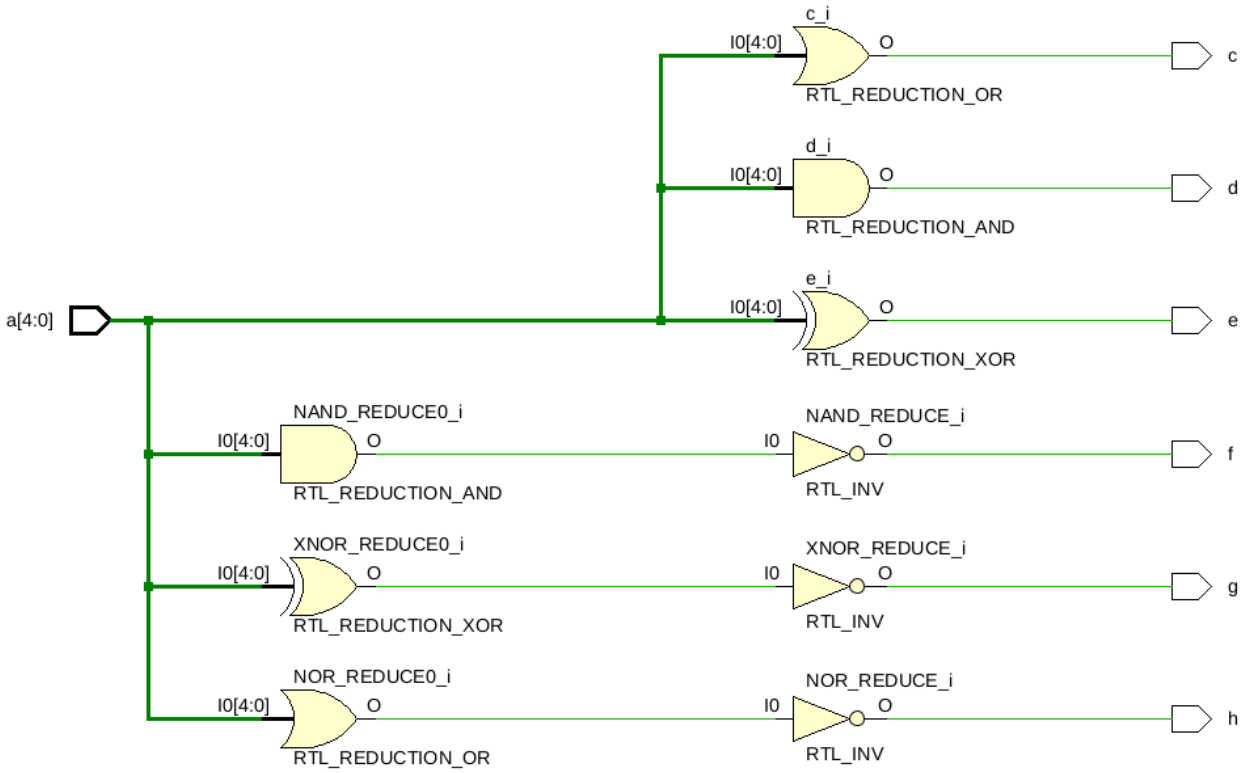
Y con esta librería cargada, se pueden utilizar las siguientes funciones:

- **or_reduce(<std_logic_vector>):** esta función hace una OR de todos los bits de la señal *std_logic_vector* y devuelve una valor *std_logic* de 1 bit.
- **and_reduce(<std_logic_vector>):** esta función hace una AND de todos los bits de la señal *std_logic_vector* y devuelve una valor *std_logic* de 1 bit.
- **nand_reduce(<std_logic_vector>):** esta función hace una NAND de todos los bits de la señal *std_logic_vector* y devuelve una valor *std_logic* de 1 bit.
- **nor_reduce(<std_logic_vector>):** esta función hace una NOR de todos los bits de la señal *std_logic_vector* y devuelve una valor *std_logic* de 1 bit.
- **xor_reduce(<std_logic_vector>):** esta función hace una XOR de todos los bits de la señal *std_logic_vector* y devuelve una valor *std_logic* de 1 bit.
- **xnor_reduce(<std_logic_vector>):** esta función hace una XNOR de todos los bits de la señal *std_logic_vector* y devuelve una valor *std_logic* de 1 bit.

Un ejemplo de uso de estas funciones

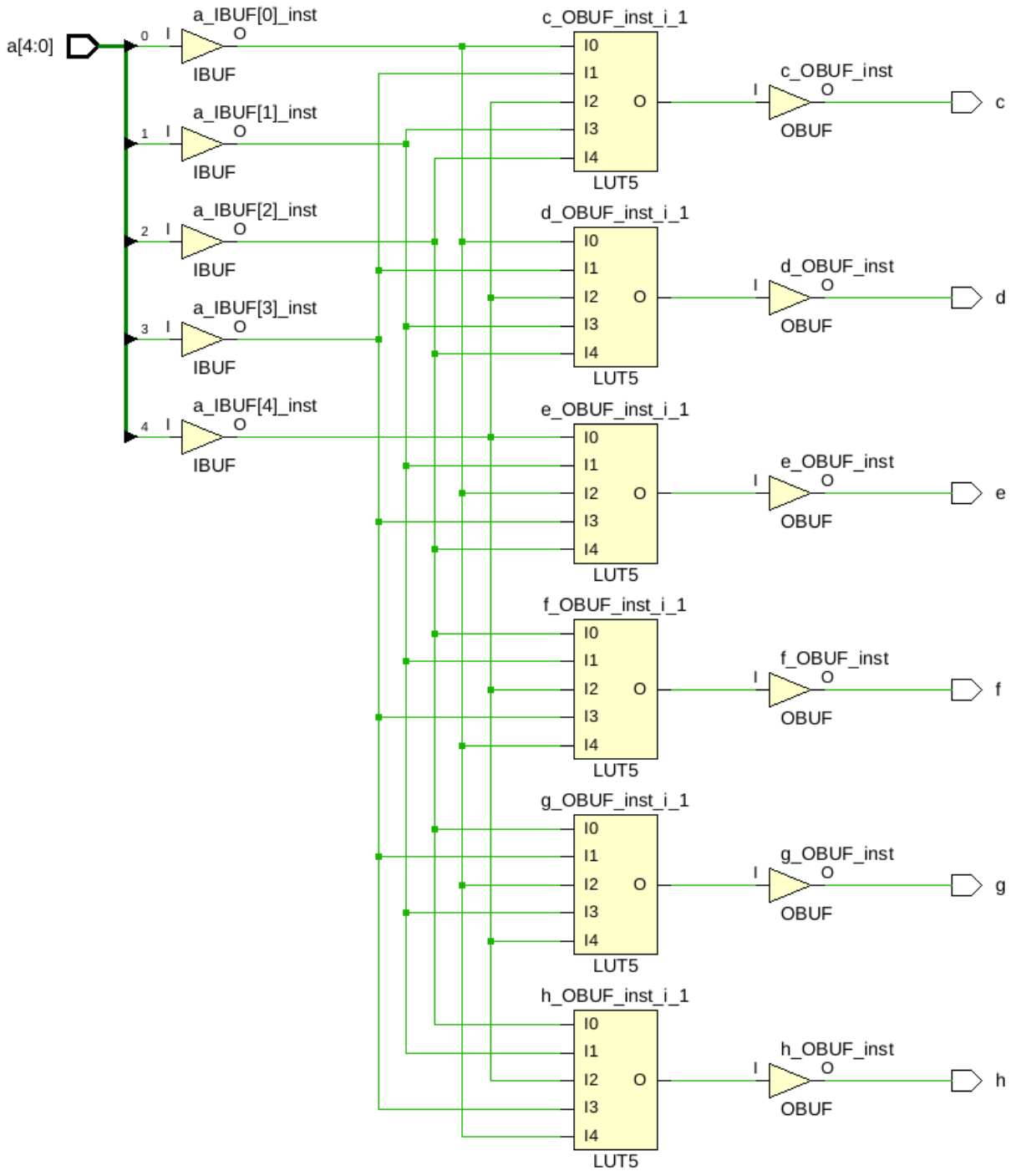
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_misc.all;

entity test is
    port(
        a : in std_logic_vector(4 downto 0);
        c, d, e, f, g, h : out std_logic
    );
end entity;

architecture arch_test of test is

begin

    c <= or_reduce(a);
    d <= and_reduce(a);
    e <= xor_reduce(a);
    f <= nand_reduce(a);
    g <= xnor_reduce(a);
    h <= nor_reduce(a);

end architecture;
```

Las funciones generan un modelo RTL como el siguiente



Y un modelo de síntesis como el siguiente

https://soceame.wordpress.com/2024/05/18/como-hacer-una-and-or-xor-nand-etc-de-todos-los-bits-de-una-senal-en-vhdl/



NOTA

En el estándar del 2008 de VHDL se admite una nueva forma de hacer lo anterior, más simple y sin recurrir a la librería '*std_logic_misc*', y es utilizar la palabra lógica delante de la señal/puerto. Ejemplo:

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

entity borrar is
    port(
        a : in std_logic_vector(4 downto 0);
        c, d, e, f, g, h : out std_logic
    );
end entity;

architecture arch_borrar of borrar is

begin

    c <= or a;
    d <= and a;
    e <= xor a;
    f <= nand a;
    g <= xnor a;
    h <= nor a;

end architecture;
```