How to AND, OR, XOR, NAND, etc. all bits of a signal in VHDL

Created by: David Rubio G.

Blog post: https://soceame.wordpress.com/2025/03/04/how-to-and-or-xor-nand-etc-all-bits-of-a-signal-in-vhdl/

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

GitHub: https://github.com/DRubioG

Last modification date: 04/03/25

A very common situation is when you want to perform a binary operation on a multi-bit signal in VHDL, such as an OR or an AND of all the bits of a signal (or part of it).

The general method is to resort to performing the operation bit by bit, which can leave parts of the code quite long. E.g.:

```
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);
```

Another option is to use a for loop with a variable (but this can make development very complex, because the variables must be initialized to avoid errors):

```
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;
...
```

Solution

Well, to solve this situation, there is a library called "std_logic_misc" with several functions that perform these calculations.

To use this library you have to load the library at the beginning:

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

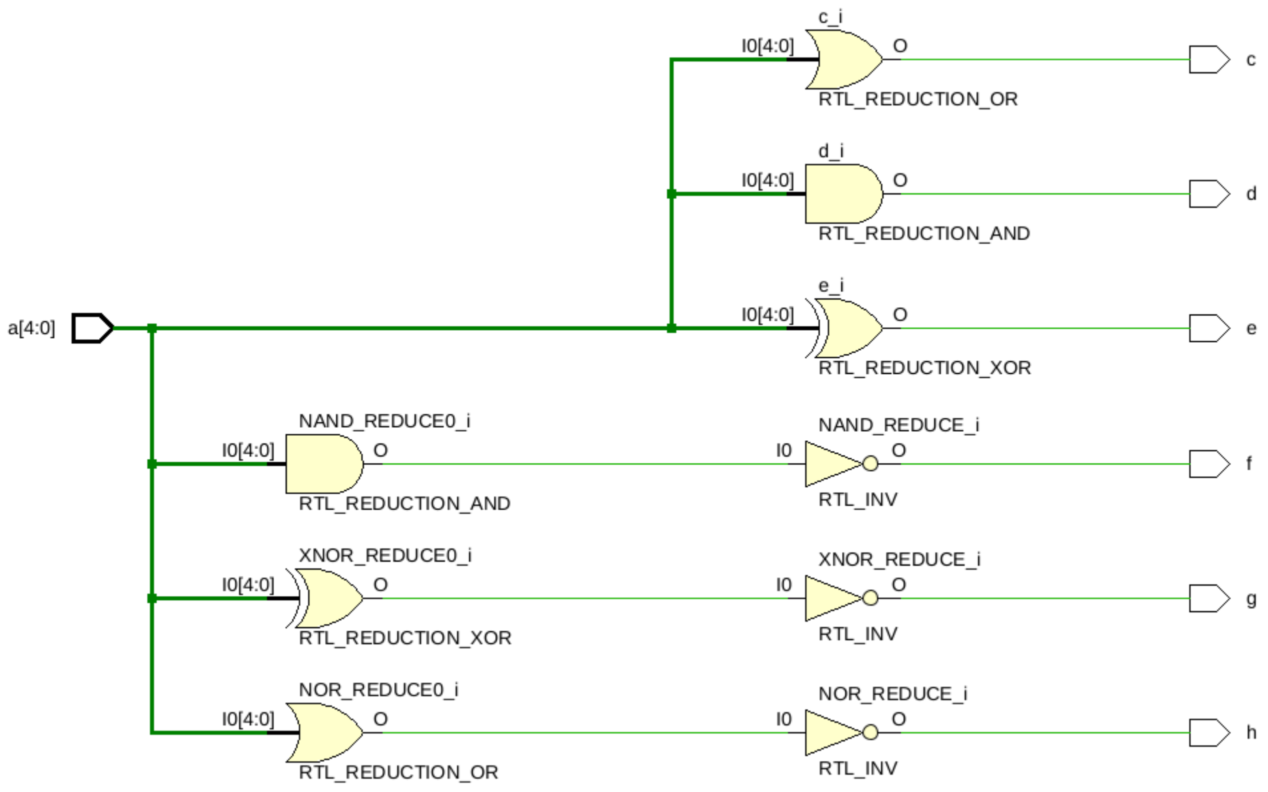
And with this library loaded, the following functions can be used:

- **or_reduce(<std_logic_vector>)**: This function ORs all the bits of the *std_logic_vector* signal and returns a 1-bit *std_logic* value.
- **and_reduce(<std_logic_vector>):** This function ANDs all the bits of the *std_logic_vector* signal and returns a 1-bit *std_logic* value.
- **nand_reduce(<std_logic_vector>)**: This function NANDs all the bits of the *std_logic_vector* signal and returns a 1-bit *std_logic* value.
- **nor_reduce(<std_logic_vector>)**: This function NORs all the bits of the *std_logic_vector* signal and returns a 1-bit *std_logic* value.
- **xor_reduce(<std_logic_vector>)**: This function XORs all the bits in the *std_logic_vector* signal and returns a 1-bit *std_logic* value.
- **xnor_reduce(<std_logic_vector>)**: This function XNORs all the bits in the *std_logic_vector* signal and returns a 1-bit *std_logic* value.

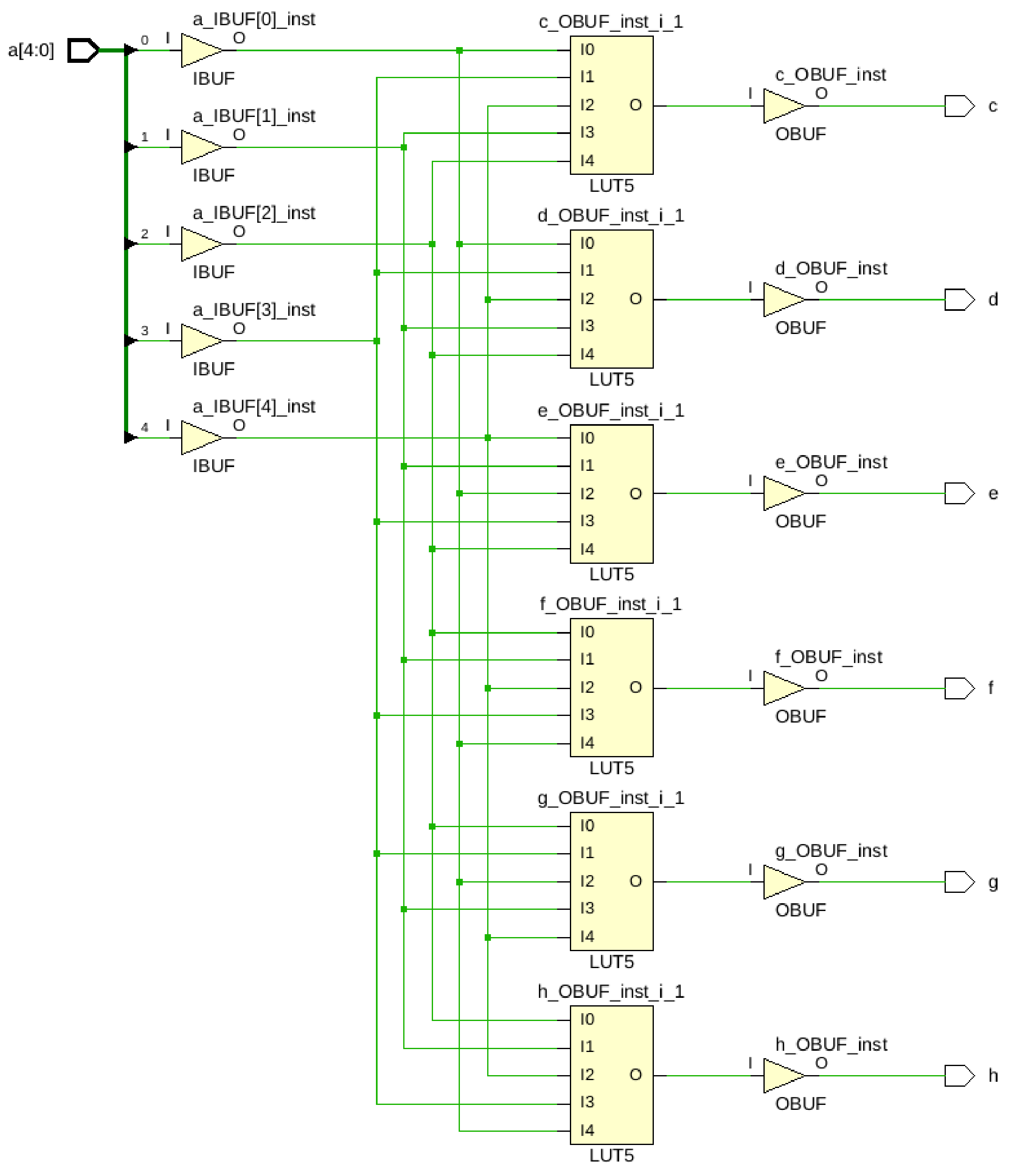
An example of using these functions

```
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;
```

The functions generate an RTL model like the following



And a synthesis model like the following https://soceame.wordpress.com/2025/03/04/how-to-and-or-xor-nand-etc-all-bits-of-a-signal-in-vhdl/



NOTE

The 2008 VHDL standard supports a new, simpler way of doing the above, without resorting to the '*std_logic_misc*' library, and that is to use the word logic before the signal/port. Example:

```
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;
```