

# VHDL Syntax : language constraints

---

## Identifiers :

- should be meaningful

- signal baud\_rate\_counter

not

- signal xr2\_x1\_10\_sig

- Should only use the characters

- A to Z

- a to z

- 0 to 9

- and the underscore \_

- should use upper and lower case consistently

- signal BaudRateCounter

# VHDL Syntax : language constraints

---

□ should be 15 characters or less

○ signal internal\_trap\_properly\_received

□ should not be confusing

○ signal BaudRateCounter

and

○ signal Baud\_Rate\_Counter

which are different identifiers

**BUT !!!!**

○ signal Baud\_Rate\_Counter

and

○ signal baud\_rate\_counter

**ARE THE SAME IDENTIFIER TO THE COMPILER**

**vHDL iS nOT cASe SenSITivE At aLL**

**unix based tools are !**

## VHDL Syntax : bit operators

---

VHDL has the following operators that act on type bit

not  
and or  
nand nor  
xor xnor

The precedence rules are different to the normal precedence of boolean algebra

- 'not' has the highest precedence
- 'and', 'or', 'nand', 'nor', 'xor', 'xnor' have equal precedence

SO

# VHDL Syntax : bit operators

---

how do you evaluate

```
z <= a AND b OR c ;
```

????

```
z <= (a AND b) OR c ;
```

```
z <= a AND (b OR c) ;
```

A good compiler/simulator will reject an ambiguous expression.  
A bad compiler/simulator will decide for itself.

# VHDL Syntax : Models

---

All Vhdl models have two parts : an entity

```
entity and2 is
    port (x, y : in  bit ;
          z   : out bit);
end entity and2 ;
```

This is a black box' description of our model, it describes the names, types and number of inputs and outputs. It tells us nothing about the internals of our model.

## VHDL Syntax : Models 2

---

The second part of the model is the architecture

```
architecture example1 of and2 is
    signal xy : bit_vector(0 to 1);
begin
    xy <= x&y;
    with xy select
        z <= '1' when "11" ,
           '0' when others ;
end architecture example1;
```

# VHDL Syntax : Models 3

---

The architecture describes the function or structure of the model. This can be :

- A data-flow model as in the example just seen
- An RTL model
- A behavioural model, as in
$$z \leq x + y;$$
- A structural model, or netlist
- A state machine model

or even

## VHDL Syntax : Models 4

---

□ a ROM

```
architecture rtl_1 of rom is
  subtype my_word is std_logic_vector
    (7 downto 0);
  type my_array is array(0 to 63) of my_word;
  constant s : my_array := ( "00001011",
                             "00000010",
                             "00010001",
                             "00000001",
```