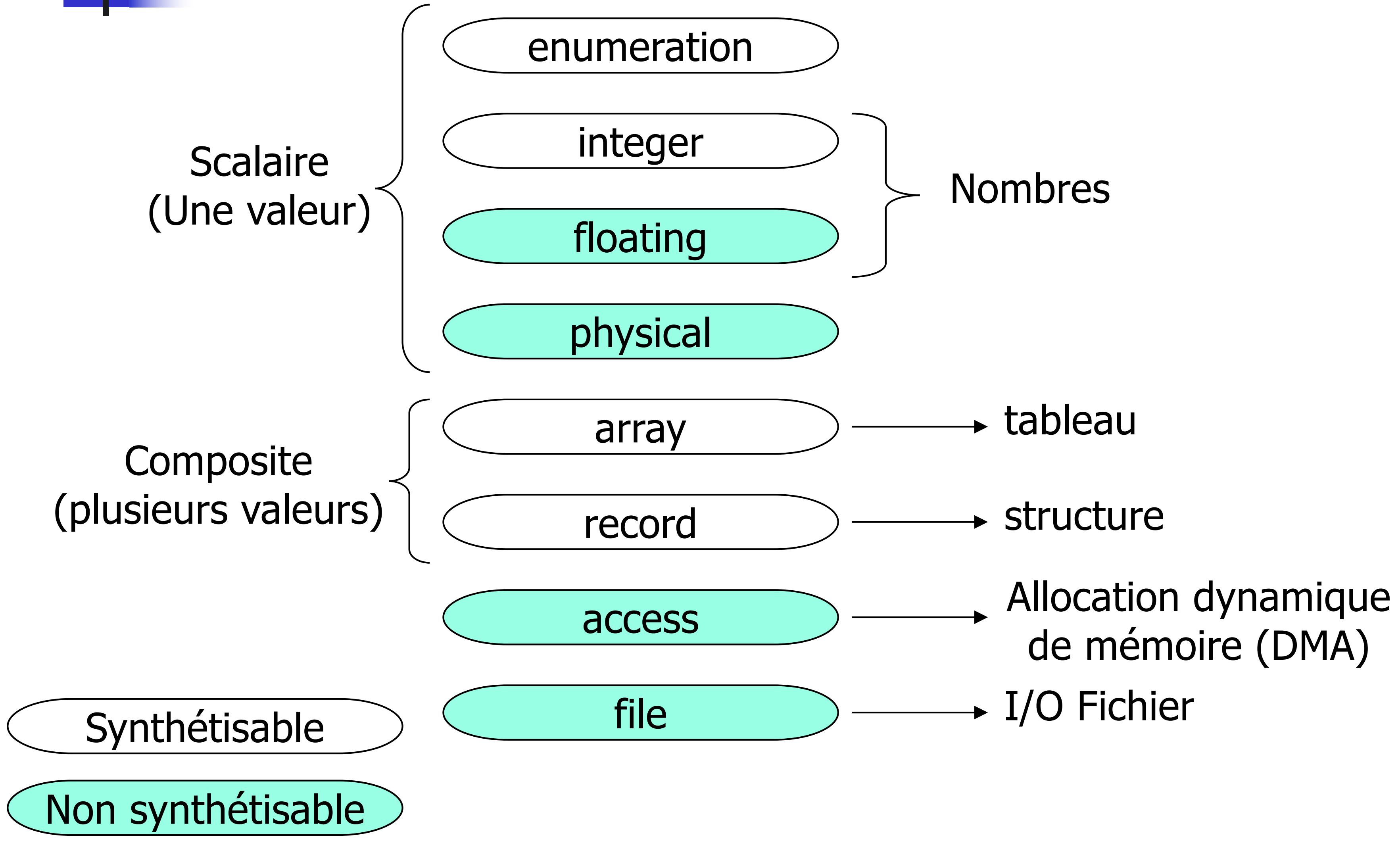
C8 – Sous-Types (subtype)

Yann DOUZE

VHDL

Les types de donnés en VHDL

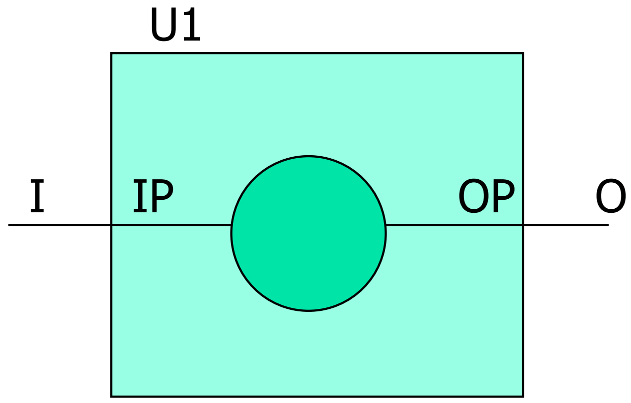


Types de données et Packages

```
package MY_TYPES is
   type CODE is (A, B, C, D, E);
end package MY_TYPES;
```

Dans un fichier séparé

```
use WORK.MY_TYPES.all;
entity FILTER is
   port(IP: in CODE;
        OP : out CODE);
end entity FILTER;
```



```
use WORK.MY_TYPES.all;
   ...
signal I, O : CODE;
   ...
U1: entity work.FILTER port map (IP => I, OP => O);
```

Sous-types du type entier

```
type INTEGER is range -2**31+1 to 2**31-1;
subtype NATURAL is INTEGER range 0 to 2**31-1;
subtype POSITIVE is INTEGER range 1 to 2**31-1;
```

Définit dans le package STD.STANDARD

```
subtype SHORT is INTEGER range -128 to +127;
subtype LONG is INTEGER range -2**15 to 2**15-1;
signal S: SHORT;
signal L: LONG;
```

Sous-types définit par l'utilisateur

```
variable I: INTEGER range 0 to 255;
```

Sous-type anonyme

```
I  := -1;
S  <= L;
```

Ces assignements sont ils erronés?

Synthèse des sous-types entiers

```
subtype Byte is INTEGER range 0 to 255;
signal B: Byte;
```

8 bits, non signé

```
subtype Int8 is INTEGER range -128 to +127;
signal I: Int8;
```

8 bits, signé complément à 2

```
subtype Silly is INTEGER range 1000 to 1001;
signal S: Silly;
```

10 bits, non signé

```
signal J: INTEGER;
```

32 bits, complément à 2, attention!

Opérateurs Arithmétique

	+	OK pour la synthèse
	-	
	*	
	/	Dépend de l'outil
Exposant A^B= A**B	**	
Reste après A/B	rem	
A modulo B	mod	
Valeur absolue |A|	abs	
	=	OK pour la synthèse
Différent	/=	
	<	
Pareil que l'assignement d'un signal	<=	
	>	
	>=	

Représentation des valeurs entières

Nombres entiers

```
0      99     1e6    1E6    1000000      1_000_000
```

_ ignoré

Ecrire un STD_LOGIC_VECTOR en binaire, octal et hexadécimal

```
B"0000_1111"     O"017"      x"0F"
```

VHDL 1993

Sous-types d’un tableau (array)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package BusType is
   subtype DataBus is STD_LOGIC_VECTOR(7 downto 0);
end package Bustype;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
Use WORK.BusType.all;

Entity CNTL is
   port( CLK   : in     STD_LOGIC;
         D     : in     DataBus;
         Sin   : in     STD_LOGIC_VECTOR(3 downto 0);
         Q     : out    DataBus;
         Sout  : out    STD_LOGIC_VECTOR(1 downto 0));
End entity CNTL;
```

Les types tableaux (array)

Type tableau non contraint

```
type STD_LOGIC_VECTOR is array (NATURAL range <>) of STD_LOGIC;
```

Type de l’index

Type de l’élément

```
subtype Byte is STD_LOGIC_VECTOR(7 downto 0);
```

```
type RAM1Kx8 is array (0 to 1023) of Byte;
variable RAM: RAM1Kx8;
```

Type tableau contraint

Modélisation des mémoires

```
entity DualPortRam is
   port (  Clock, Wr, Rd   : in    Std_logic;
           AddrWr, AddrRd : in    Std_logic_vector(3 downto 0);
           DataWr          : in    Std_logic_vector(7 downto 0);
           DataRd          : out   Std_logic_vector(7 downto 0));
end entity;

architecture modele of DualPortRam is
    type RamType is  array (0 to 15) of Std_logic_vector(7 downto 0);
    signal RAM : RamType;
begin
   process (Clock)
   begin
      if RISING_EDGE(Clock) then
         if Wr = '1' then
            Ram(To_integer(Unsigned(AddrWr))) <= DataWr;
         end if;
         if Rd = '1' then
            DataRd <= Ram(To_integer(Unsigned(AddrRd)));
         end if;
      end if;
   end process;
end architecture;
```

Attribut ‘RANGE

```
Signal A: STD_LOGIC_VECTOR(...);
```

```
process(A)
   variable V: STD_LOGIC;
begin
   V := ‘0’;

   for I in 0 to 7 loop
      V := V xor A(I);
   end loop;

   for I in A’RANGE loop
      V := V xor A(I);
   end loop;
   ...
end process
```

Pas générique

Bien

Attributs de type et tableau

```
signal A: STD_LOGIC_VECTOR(7 downto 0);
subtype SHORT is INTEGER range 0 to 15;
type MODE is (W, X, Y, Z);
```

- Attributs de tableau (à utiliser dès que possible)

```
A'LOW   = 0
A'HIGH  = 7
A'LEFT  = 7
A'RIGHT = 0
```

```
A'RANGE = 7 downto 0
A'REVERSE_RANGE = 0 to 7
A'LENGTH = 8
```

- Attributs de type (à éviter en synthèse)

```
SHORT'LOW   = 0
SHORT'HIGH  = 15
SHORT'LEFT  = 0
SHORT'RIGHT = 15
```

```
MODE'LOW    = W
MODE'HIGH   = Z
MODE'LEFT   = W
MODE'RIGHT  = Z
```

Agrégats

```
Type BCD6 is array (5 downto 0) of STD_LOGIC_VECTOR(3 downto 0);

Variable V: BCD6 := (“1001”,“1000”,“0111”,“0110”,“0101”,“0100”);
```

```
V := (“1001”, “1000”, others => ”0000”);

V := (3 => “0110”, 1 => “1001”, others => ”0000”);

V := (others => ”0000”);
```

```
Variable A: STD_LOGIC_VECTOR(3 downto 0);
```

```
A := (others => ‘1’);
```

Types ambigus

```
port (A,B: in STD_LOGIC);
```

```
process(A, B)
begin
   case A & B is
   when “00” = > ...
```

Illégal!

```
library IEEE;
use IEEE.NUMERIC_STD.all
```

```
variable N: INTEGER;
```

Illégal!

```
N := TO_INTEGER(“1111”);
```

Expressions de qualification

```
process(A, B)
   subtype T is STD_LOGIC_VECTOR(0 to 1);
begin
   case T'(A & B) is
   when "00" = > ...
```

```
N := TO_INTEGER(UNSIGNED'("1111"));
```

N = 15

```
N := TO_INTEGER(SIGNED'("1111"));
```

N = -1