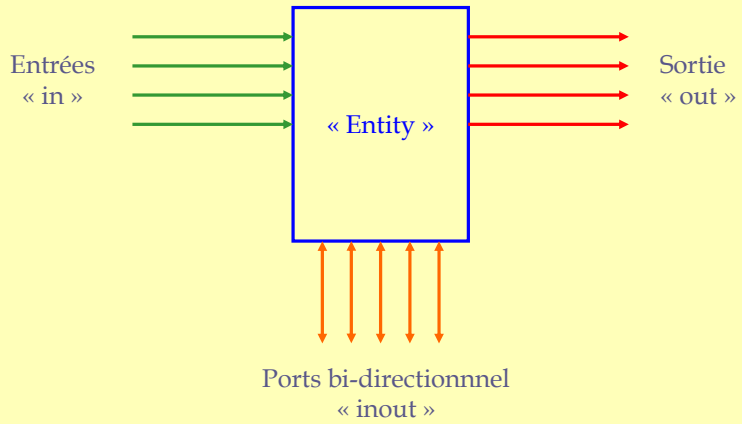
Plan

- Notions d'entité, architecture, de process et de signal
- Notions de hiérarchie et de composant
- Le caractère parallèle des transferts
- Les types de signaux et les opérateurs
- Diverses constructions :
  - IF THEN ELSIF ELSE – CASE WHEN
  - WITH SELECT WHEN – WHEN ELSE
  - Process synchrône
  - Machine à états

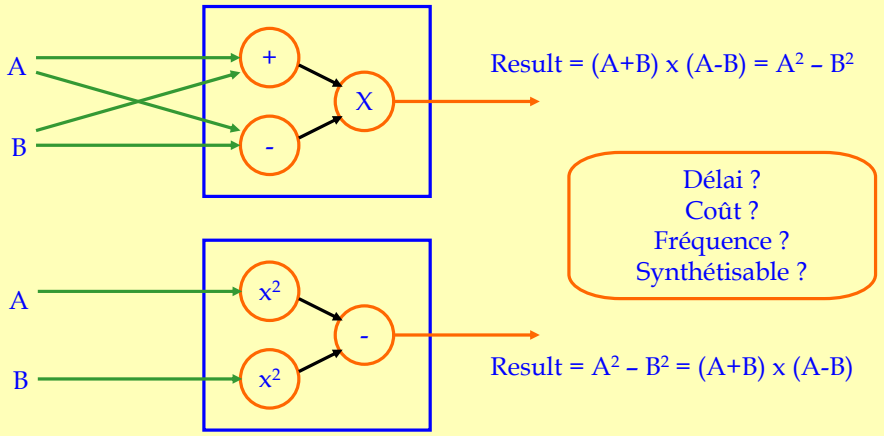
La notion d'entité



Exemple d'entité

```
ENTITY sqrt IS
    GENERIC (WIDTH : INTEGER:=24);
    PORT(
        clk         : IN    STD_LOGIC;
        reset       : IN    STD_LOGIC;
        START       : IN    STD_LOGIC;
        INPUT_X     : IN    STD_LOGIC_VECTOR (width-1 downto 0);
        RESULT      : OUT   STD_LOGIC_VECTOR (width-1 downto 0);
        READY : OUT STD_LOGIC);
END sqrt;
```

Notion d'architecture



Exemple d'architecture

```
ARCHITECTURE MAX2A OF TEST IS
   SIGNAL node1,node2: STD_LOGIC_VECTOR(7 downto 0);
BEGIN
   node1 <= a+b;
   node2<=a-b;
   result<=node1*node2;
END MAX2;

ARCHITECTURE MAX2B OF TEST IS
   SIGNAL node1,node2: STD_LOGIC_VECTOR(7 downto 0);
BEGIN
   node1 <= a*a;
   node2<=b*b;
   result<=node1-node2;
END MAX2;
```

Remarque :
Ces deux architectures offrent-elles la même précision des calculs ?

Notion de process et de signal

Un « signal » représente un point « matériel » du circuit dans le sens ou il a une et une seule valeur à chaque instant donné (par opposition aux variables). Le temps est discrétisé.

L'état d'un signal est habituellement modifié par un et un seul process. Néanmoins, les signaux qui disposent d'une fonction de résolution peuvent être modifiés par plusieurs processes. La fonction de résolution calcule l'état du signal en fonction des valeurs imposées par les processes agissant sur le signal.

La valeur des signaux et des entrées/sorties est visible de tous les processes (portée globale)

Un process est « sensible à un certain nombre de signaux » Tant qu'aucun de ces signaux ne change de valeur, rien ne se passe à l'intérieur du process

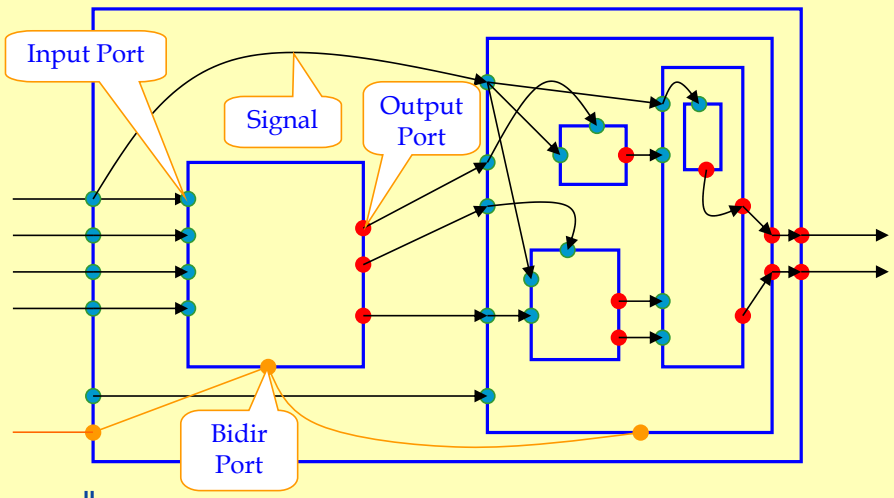
Exemple de process et de signal

```
SIGNAL x,y,A,B: STD_LOGIC_VECTOR(7 downto 0);

Process(x,y)
   begin
   if (x>y) then
     A<=x;B<=y;
   else A<=y;B<=x;
End process;

Process(x)
   begin
   y<=A-B;
End process;
```

Notions de hiérarchie et de composants



Déclaration d'un composant

```
architecture Arch_Simu of DDR_FIFO is
    component GFIFO is
    generic (
            width: integer:=32;
            depth: integer:=16);
    port (
            reset: in STD_LOGIC;
            clk: in STD_LOGIC;
            Data_In: in STD_LOGIC_VECTOR(width-1 downto 0);
            Data_Out: out STD_LOGIC_VECTOR(width-1 downto 0);
            Write: in STD_LOGIC;
            Read: in STD_LOGIC;
            Full: out STD_LOGIC;
            Empty: out STD_LOGIC);
    end component;

    signal … ;
    begin
```


Instantiation d'un composant

…

```
begin
TheFifo : GFIFO
    generic map (width,depth)
    port map (
            reset=>reset,
            clk=>clk,
            Data_In=>Data_In,
            Data_Out=>FIFO_Out,
            Write=>FIFO_WR,
            Read=>FIFO_RD,
            Full=>FIFO_Full,
            Empty=>FIFO_Empty);
```

Nom du composant

Nom de l'instance

Nom du port

Nom du signal que l'on connecte au port

Le caractère parallèle des transferts

```
Architecture …
    signal A,B : std_logic_vector(7 downto 0);
    begin
    …
    Process (clk)
        begin
        A<=B;   -- A (t+epsilon) prend la valeur B(t)
        B<=A;   -- B (t+epsilon) prend la valeur A(t)
    end process;
```

Résultat : à chaque changement d'état de l'horloge (flanc montant ou descendant), A et B échangent leurs valeurs.

Les types de signaux

- VHDL offre des types de signaux très variés et personnalisables.
- En synthèse, on se ramène toujours à des '1', '0' et parfois 'Z'
- La librairie internationalement utilisée est IEEE.std_logic1164
  - IEEE.std_logic_arith pour les opérations ALU
  - IEEE.std_logic_unsigned pour les opérations non signées
  - IEEE.std_logic_signed pour les opérations signées
- Deux types de signaux :
  - STD_ULOGIC
  - STD_LOGIC (idem mais avec fonction de résolution)
- Valeurs possibles des signaux STD_(U)LOGIC :
  - 'U' : Uninitialized; 'X' : Forcing Unknown :
  - '0' : Forcing Zero ; '1' : Forcing One; 'Z' : High Impédance
  - 'W' : Weak Unknown; 'L' : Weak Zero; 'H' : Weak '1'
  - '-' : Don't care

Fonction de résolution des signaux std_logic

	U	X	0	1	Z	W	L	H	-
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	X
1	U	X	X	1	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
-	U	X	X	X	X	X	X	X	X

Autres types de signaux supportés par VHDL

Type de signal	Valeurs possibles
Integer	Selon l'implémentation
Real	Selon l'implémentation
Boolean	TRUE,FALSE
Character	Selon le Package STANDARD
Bit	0 – 1
Bit_Vector	Tableau de « Bit »
Time	Selon l'implémentation
String	Tableau de « Character »
Natural	Selon l'implémentation (>=0)
Positive	Selon l'implémentation (>0)

Les opérateurs

Type d'opérateurs						
Logique	AND	OR	NAND	NOR	XOR	XNOR
Comparaison	=	/=	<	<=	>	>=
Décalage	SLL	SRL	SLA	SRA	ROL	ROR
Arithmétique	+	-	& *concat*			
Multiplication	*	/	MOD	REM		
Autres	**	ABS	NOT			

Diverses constructions (1/4)

```
IF __expression THEN
   __statement;
   __statement;
ELSIF __expression
   THEN
   __statement;
   __statement;
ELSE
   __statement;
   __statement;
END IF;
```

```
CASE __expression IS
   WHEN __constant_value
   =>
      __statement;
      __statement;
   WHEN __constant_value
   =>
      __statement;
      __statement;
   WHEN OTHERS =>
      __statement;
      __statement;
END CASE;
```


Diverses constructions (2/4)

```
WITH __signal SELECT
   __signal     <=     __statement WHEN __value,
                       __statement WHEN __value,
                       …
                       __statement WHEN OTHERS;
```

```
   __signal     <=     __value WHEN __condition ELSE
                       __value WHEN __condition ELSE
                       …
                       __value;
```


Diverses constructions (3/4)

```
Process (reset,clk)
   begin
   if (reset='1') then
         __signal<=__value;
         __signal<=__value;
   elsif (clk'event and clk='1') then
         __signal<=__value;
         __signal<=__value;
   end if;
End process;
```


Diverses constructions (4/4)

```
   TYPE STATE_TYPE IS (__state_name, __state_name, __state_name);
   SIGNAL state: STATE_TYPE;
BEGIN
   PROCESS (clk)
   BEGIN
      IF reset = '1' THEN
         state <= _state_name;
      ELSIF clk'EVENT AND clk = '1' THEN
         CASE state IS
            WHEN __state_name =>
               IF __condition THEN
            WHEN __state_name =>
               IF __condition THEN
         END CASE;
      END IF;
   END PROCESS;
   WITH state SELECT
      __output_name     <=     __output_value     WHEN     __state_name,
                               __output_value     WHEN     __state_name,
                               __output_value     WHEN     __state_name;
```

Moore ?
ou
Mealy ?