ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Enseignes et afficheurs à LED

Instructions concurrentes en VHDL

Dr. Mamadou Lamine NDIAYE

Instructions concurrentes en VHDL

Dr. Mamadou Lamine NDIAYE

Instructions concurrentes en VHDL

- Intructions concurrentes
  - Affectation simple
  - Affectation conditionnelle
  - Affectation sélective
- Component
- Generation conditionnelle

VHDL : Les instructions concurrentes

VHDL : Les instructions concurrentes

Logique combinatoire : Instructions concurrentes

- Décomposition en fonction simple (concurrente), ensemble de composants ou d'algorithmes travaillant en parallèle et agissant les uns sur les autres.
- Chaque instruction effectue donc un traitement en parallèle avec les autres instructions.
- L'ordre d'écriture des instructions est sans importance.

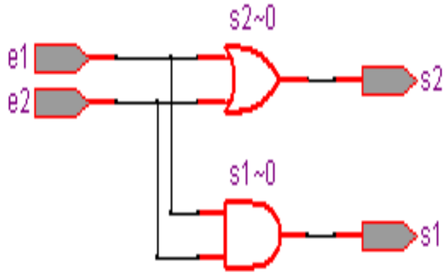
Affectation simple

signal <= signal1

- Connexion de deux signaux
- Utilise l'opérateur <=

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
Entity fonction_simple is
Port( e1, e2  :  in std_logic;
      s1, s2  :  out std_logic
);
End fonction_simple;
```

```
Architecture archi_simple of fonction_simple is
Begin
s1 <= e1 and  e2;     -- ordre d'écriture indiff
s2 <= e1 or  e2;      -- ordre d'écriture indiff

end archi_simple;
```



Affectation conditionnelle

```
signal  <=  signal1  when expresion1 else
            .........
            signal2  when expresion2 else
            signal3  ;
```

- Affectation conditionnelle WHEN / ELSE
  - C’est une instruction qui a une seule cible mais peut avoir plusieurs expressions.
  - Les conditions sont évaluées séquentiellement, si une condition est vraie alors l’expression correspondante est exécutée.

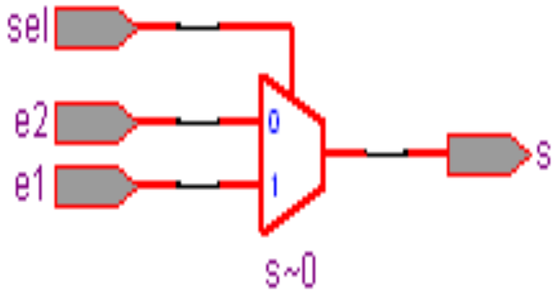
Affectation conditionnelle

```
signal  <=   signal1  when expresion1 else
             .........
             signal2  when expresion2 else
             signal3  ;
```

- Affectation conditionnelle WHEN / ELSE
  - C'est une instruction qui a une seule cible mais peut avoir plusieurs expressions.
  - Les conditions sont évaluées séquentiellement, si une condition est vraie alors l'expression correspondante est exécutée.

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;

Entity mux2 is
Port( e1, e2 , sel : in std_logic;
                s :  out std_logic

);

end mux2;
```

```
architecture archi_mux2 of mux2 is
begin
s <= e1 when  sel='0'   else
        e2;

end archi_mux2;
```



Affectation sélective

```
with  expression  select
        signal    <=   signal1  when   valeur1,
                       signal2  when   valeur2,
                       signal 3 when   others ;
```

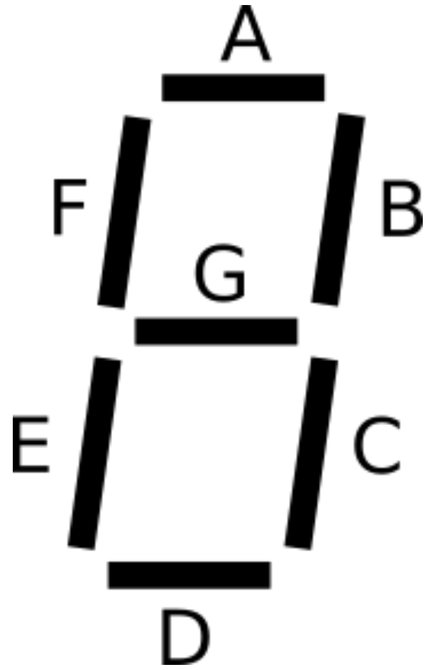
- Affectation sélective WITH ……SELECT
  - C'est une affectation d'une valeur suivant l'état de l'expression testée.
  - Exemple : Décodeur 7 segment

Affectation sélective

```
with  expression  select
        signal    <=   signal1  when   valeur1,
                       signal2  when   valeur2,
                       signal 3 when   others ;
```

- Affectation sélective WITH ……SELECT
  - C'est une affectation d'une valeur suivant l'état de l'expression testée.
  - Exemple : Décodeur 7 segment

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;

entity Bcd7Segment is
Port ( data : in STD_LOGIC_VECTOR (3 downto 0);
       7Segment : out STD_LOGIC_VECTOR (0 to 6));
end Bcd7Segment;
```

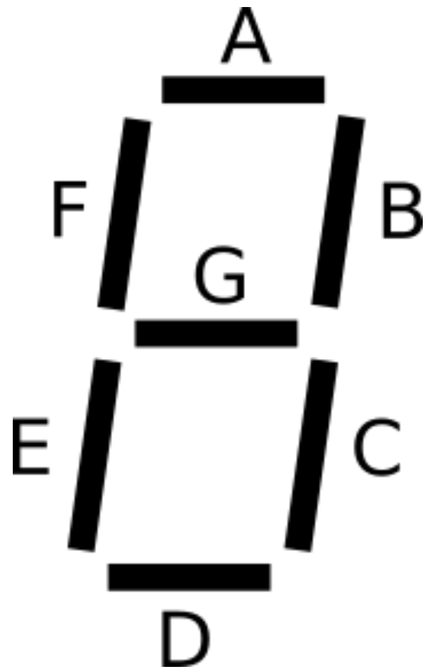


Affectation sélective

```
with  expression  select
        signal    <=   signal1  when   valeur1,
                       signal2  when   valeur2,
                       signal 3 when   others ;
```

```
architecture comportement of Bcd7Segment is
begin
with data select
7Segment <=
“0000001" when x"0" ,
”1001111" when  x"1" ,
"0010010" when x"2" ,
"0000110" when x"3" ,
"1001100" when x"4" ,
"0100100" when x"5" ,
"0100000" when x"6" ,
"0001111" when  x"7" ,
"0000000" when x"8" ,
"0000100" when x"9" ,
"1111111" when others;
end comportement;
```

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;

entity Bcd7Segment is
Port ( data : in STD_LOGIC_VECTOR (3 downto 0);
       7Segment : out STD_LOGIC_VECTOR (0 to 6));
end Bcd7Segment;
```



Component

Les Composants (component)

- Le mot COMPONENT permet de déclarer un prototype (modèle) de composant
- L'instanciation du composant se fait alors dans le corps de l'architecture
  - <NOM_INSTANCE>:<NOM_COMPOSANT> port map (LISTE DES CONNEXIONS);

Component

Les Composants (component)

- Le mot COMPONENT permet de déclarer un prototype (modèle) de composant
- L'instanciation du composant se fait alors dans le corps de l'architecture
  - <NOM_INSTANCE>:<NOM_COMPOSANT> port map (LISTE DES CONNEXIONS);

```
ARCHITECTURE Structure OF Decodeur_7seg IS
COMPONENT Bcd7Segment
PORT (data      : IN      STD_LOGIC_VECTOR(3 DOWNTO 0);
          HEX     : OUT    STD_LOGIC_VECTOR(0 TO 6));
END COMPONENT;
BEGIN
              LEDR <= SW;
    digit3: Bcd7Segment PORT MAP (SW(15 DOWNTO 12), HEX3);
    digit2: Bcd7Segment PORT MAP (SW(11 DOWNTO 8), HEX2);
    digit1: Bcd7Segment PORT MAP (SW(7 DOWNTO 4), HEX1);
    digit0: Bcd7Segment PORT MAP (SW(3 DOWNTO 0), HEX0);
END Structure;
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY Decodeur_7seg  IS
PORT (SW : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
LEDR : OUT     STD_LOGIC_VECTOR(15 DOWNTO 0);
          -- red LEDs
HEX3, HEX2, HEX1, HEX0 : OUT   STD_LOGIC_VECTOR(0 TO 6));
          -- 7-segs
END Decodeur_7seg ;
```

Génération conditionnelle

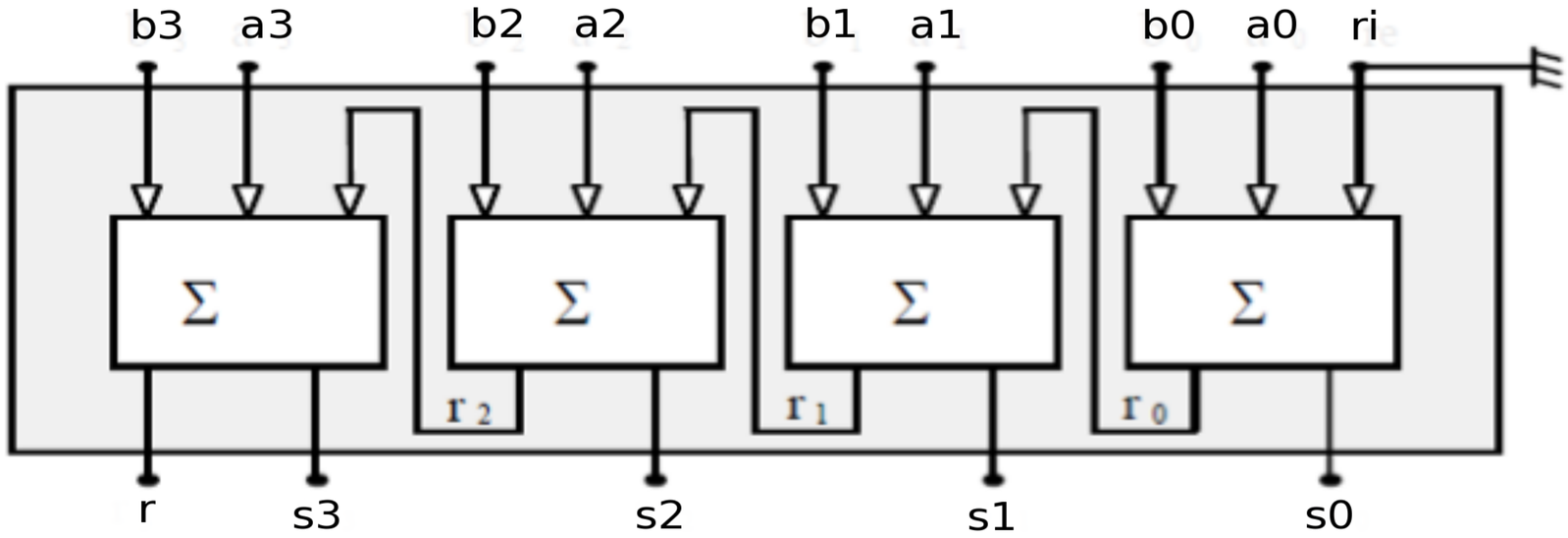
- Génération conditionnelle : GENERATE
  - Permet l'exécution conditionnelle d'instructions concurrentes
  - Permet l'exécution itérative d'instructions concurrentes
  - Les instructions ne seront prises en compte que si la condition est vraie.
  - s'utilise avec for (itérative) ou if (conditionnelle)

```
label : if  (condition)  generate
-- suite d'instructions concurrentes
end generate label;
label : for i in 0 to x generate
-- suite d'instruction concurrente
End generate label ;
```

Génération conditionnelle

- Génération conditionnelle : GENERATE
  - Permet l'exécution conditionnelle d'instructions concurrentes
  - Permet l'exécution itérative d'instructions concurrentes
  - Les instructions ne seront prises en compte que si la condition est vraie.
  - s'utilise avec for (itérative) ou if (conditionnelle)

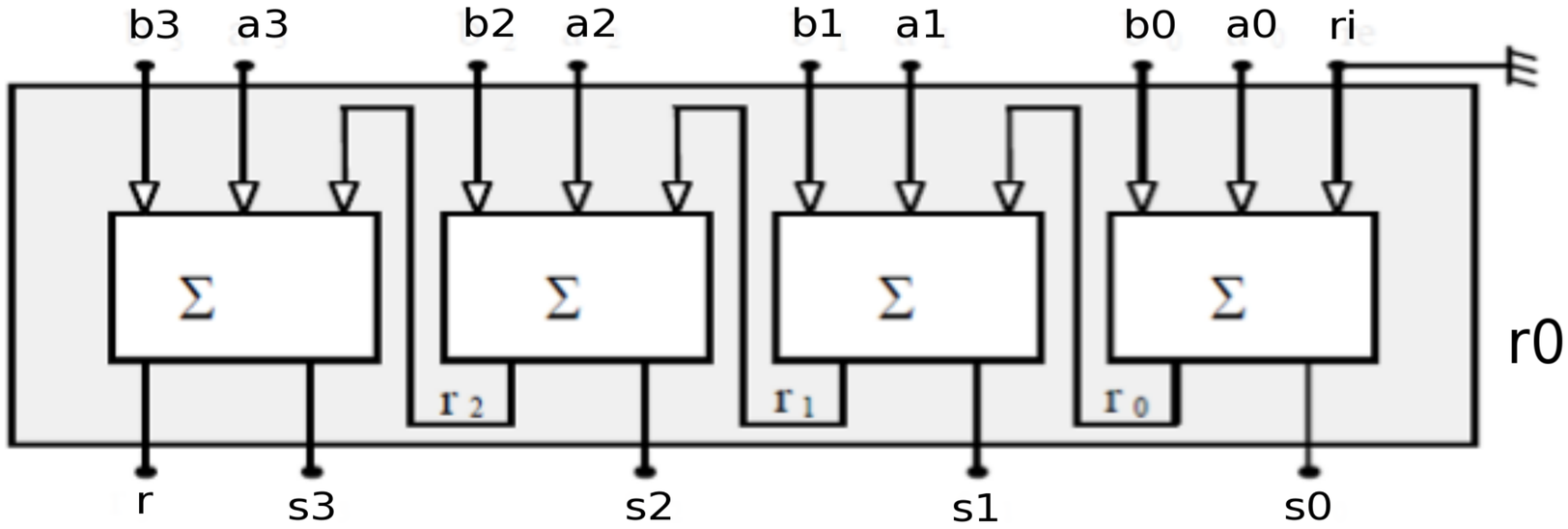
```
label : if  (condition)  generate
-- suite d'instructions concurrentes
end generate label;
label : for i in 0 to x generate
-- suite d'instruction concurrente
End generate label ;
```



Génération conditionnelle

- Génération conditionnelle : GENERATE
  - Permet l'exécution conditionnelle d'instructions concurrentes
  - Permet l'exécution itérative d'instructions concurrentes
  - Les instructions ne seront prises en compte que si la condition est vraie.
  - s'utilise avec for (itérative) ou if (conditionnelle)

```
label : if  (condition)  generate
-- suite d'instructions concurrentes
end generate label;
label : for i in 0 to x generate
-- suite d'instruction concurrente
End generate label ;
```



$$s0 = a0 \oplus b0 \oplus ri$$

$$r0 = (a0 \,.\, b0) + (ri.(a0 \oplus b0))$$

Génération conditionnelle

- Génération conditionnelle : GENERATE
  - s'utilise avec for (itérative) ou if (conditionnelle)

```
label : if  (condition)  generate
-- suite d'instructions concurrentes
end generate label;
label : for i in 0 to x generate
-- suite d'instruction concurrente
End generate label ;
```

```
Library ieee;
Use ieee.std_logic_1164.all;
ENTITY additionneur1bit IS
PORT ( ri, a, b : IN STD_LOGIC;
So, r : OUT STD_LOGIC);
END additionneur1bit;

ARCHITECTURE Archi OF
additionneur1bit IS
BEGIN
r <= (a AND b) OR (ri AND (a XOR b));
So <= a XOR b XOR ri;
END Archi;
```

Génération conditionnelle

- Génération conditionnelle : GENERATE
  - s'utilise avec for (itérative) ou if (conditionnelle)

```
label : if  (condition)  generate
-- suite d'instructions concurrentes
end generate label;
label : for i in 0 to x generate
-- suite d'instruction concurrente
End generate label ;
```

```
--librairie pour inclure type std_logic
Library ieee;
Use ieee.std_logic_1164.all;
entity Additionneur4bit is
Port(     ri : IN std_logic ;
          A, B : IN std_logic_vector(3 downto 0) ;
          s : OUT std_logic_vector(3 downto 0) ;
          r : OUT std_logic
          );
end Additionneur4bit;

architecture Structure of Additionneur4bit is
-- declaration  du composant additionneur1bit
component additionneur1bit is
port( ri : IN std_logic ;
      a,b :IN std_logic ;
      So,r : OUT std_logic ) ;
end component additionneur1bit ;
```

```
Library ieee;
Use ieee.std_logic_1164.all;
ENTITY additionneur1bit IS
PORT ( ri, a, b : IN STD_LOGIC;
So, r : OUT STD_LOGIC);
END additionneur1bit;

ARCHITECTURE Archi OF
additionneur1bit IS
BEGIN
r <= (a AND b) OR (ri AND (a XOR b));
So <= a XOR b XOR ri;
END Archi;
```

Génération conditionnelle

- Génération conditionnelle : GENERATE
  - s'utilise avec for (itérative) ou if (conditionnelle)

```
label : if  (condition)  generate
-- suite d'instructions concurrentes
end generate label;
label : for i in 0 to x generate
-- suite d'instruction concurrente
End generate label ;
```

```
--librairie pour inclure type std_logic
Library ieee;
Use ieee.std_logic_1164.all;
entity Additionneur4bit is
Port(     ri : IN std_logic ;
          A, B : IN std_logic_vector(3 downto 0) ;
          s : OUT std_logic_vector(3 downto 0) ;
          r : OUT std_logic
          );
end Additionneur4bit;

architecture Structure of Additionneur4bit is
-- declaration  du composant additionneur1bit
component additionneur1bit is
port( ri : IN std_logic ;
      a,b :IN std_logic ;
      So,r : OUT std_logic ) ;
end component additionneur1bit ;
```

```
-- declaration des signaux internes pour le report
des retenues ci
signal C : std_logic_vector( 3 downto 0 );
Begin
C(0) <= ri ;
-- Création des 4 aditionneurs
A : for  i  IN 0 to 3 GENERATE
LSB : if i = 0 GENERATE
b0 :  additionneur1bit port map (C(i) , A(i) , B(i) ,
s(i) ,  C(i) );
End GENERATE LSB
MSB : if i > 0 GENERATE
b3 :  additionneur1bit port map (C(i-1) , A(i) ,
B(i) , s(i) ,  C(i) );
End GENERATE MSB;
End GENERATE A;
r <= C(3) ;
end Structure;
```

```
Library ieee;
Use ieee.std_logic_1164.all;
ENTITY additionneur1bit IS
PORT ( ri, a, b : IN STD_LOGIC;
So, r : OUT STD_LOGIC);
END additionneur1bit;

ARCHITECTURE Archi OF
additionneur1bit IS
BEGIN
r <= (a AND b) OR (ri AND (a XOR b));
So <= a XOR b XOR ri;
END Archi;
```

Instructions concurrentes en VHDL

- Intructions concurrentes
  - Affectation simple
  - Affectation conditionnelle
  - Affectation sélective
- Component
- Generation conditionnelle