TD/TP :

Le langage VHDL

Année universitaire 2023/2024

I. Opérateurs Logiques :

Rappel :

Le tableau ci-dessous indique la syntaxe VHDL permettant de décrire des équations logiques utilisant les opérateurs logiques usuels.

Opérateurs	Noms	Syntaxe	Fonction obtenue
not	Non	C <= not(A)	$\overline{A}$
and	Et	C <= A and B	A.B
or	Ou	C <= A or B	A+B
nand	Non Et	C <= A nand B	$\overline{A.B}$
nor	Non Ou	C <= A nor B	$\overline{A+B}$
xor	Ou Exclusif	C <= A xor B	$A \oplus B$
xnor	Non Ou Exclusif	C <= A xnor B	$\overline{A \oplus B}$

II. Composants combinatoires :

1. Equation logique combinatoire

Décrire un système logique en VHDL dont le fonctionnement est donné par l'équation suivante :

$$s = a + \overline{a + b}$$

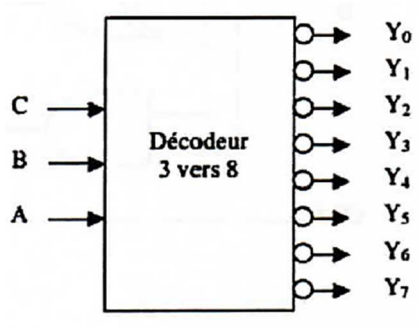
Corrigé :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ex2 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           s : out  STD_LOGIC);
end ex2;

architecture Behavioral of ex2 is

begin

S<= a or not (a or b);

end Behavioral;
```

2. Décodeur 3 vers 1

On souhaite faire la synthèse d'un décodeur 3 vers 8 avec les sorties actives au niveau bas.

1. Etablir la table de vérité du circuit.
2. Donner une implantation avec des portes NAND.
3. Comment faut-il modifier le schéma pour ajouter au montage une **entrée de validation V** telle que le circuit fonctionne normalement quand **V=1** et que toutes les sorties **Yi=1** quand **V=0** ?
4. Proposer un programme VHDL permettant de décrire ce décodeur 3 vers 8 avec une description sous forme de flot de données.

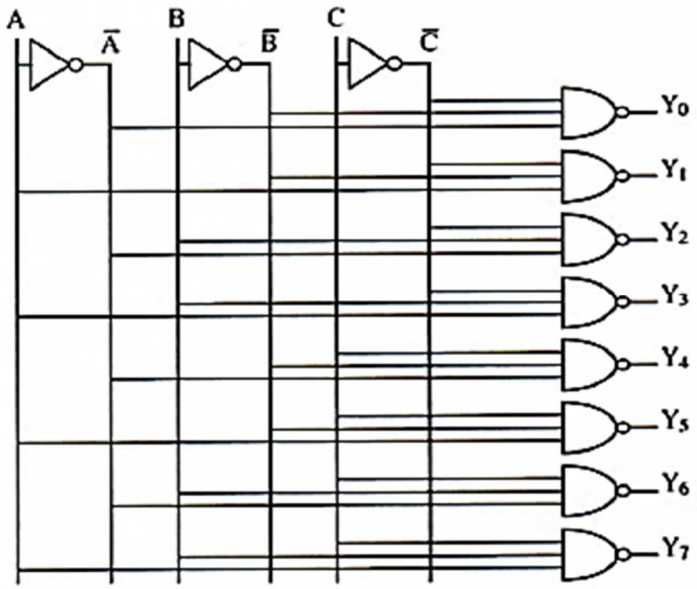


Corrigé :

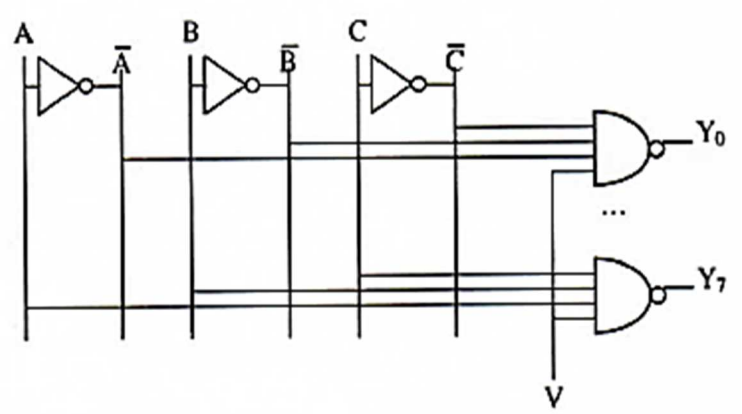
1. Etablir la table de vérité du circuit.

Décimal	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

2. Donner une implantation avec des portes NAND.



3. Comment faut-il modifier le schéma pour ajouter au montage une entrée de validation V telle que le circuit fonctionne normalement quand V=1 et que toutes les sorties Yi=1 quand V=0 ?

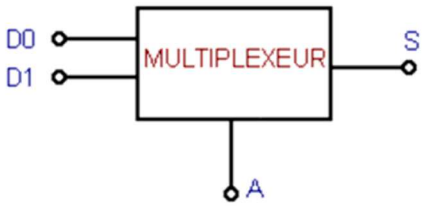


4. Proposer un programme VHDL permettant de décrire ce décodeur 3 vers 8 avec une description sous forme de flot de données.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ex3 is
    Port ( A, B, C : in  STD_LOGIC;
           Y : out  STD_LOGIC_VECTOR(7 downto 0);
end ex3;

architecture Behavioral of ex3 is

begin

Y(0)<= not( not (A) and not(B) and not(C));
Y(1)<= not(A and not(B) and not(C));
Y(3)<= not(not(A) and B and not(C));
....
Y(7)<= not(A and B and C);

end Behavioral;
```

3. Multiplexeur 2 vers 1

On souhaite réaliser un multiplexeur à deux voies (2 vers 1) comme la montre la figure ci-contre. Suivant l'état de l'entrée de **sélection A**, la **sortie S** recopie soit l'entrée **D0**, soit l'entrée **D1** (Supposons que pour **A = 0**, **S = D0** et que pour **A = 1**, **S = D1**).



1. Déduire l'équation de la **sortie S** en fonction de **D0, D1** et **A**.
2. Donner la réalisation logique de cette fonction.
3. Donner une description flot de données en **VHDL** permettant de réaliser un multiplexeur 2 vers 1.

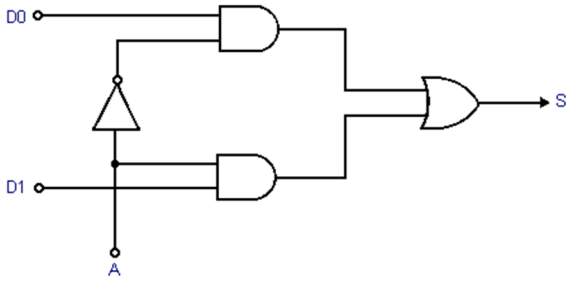
Corrigé :

1. Déduire l'équation de la sortie S en fonction de D0, D1 et A.

A		S
0		D0
1		D1

$$S = \overline{A}.D_0 + A.D_1$$

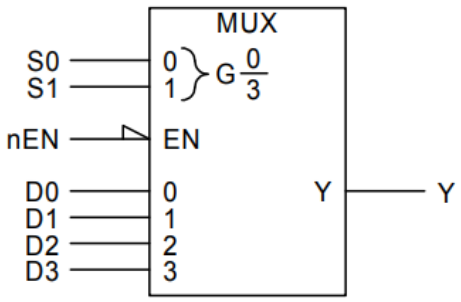
2. Donner la réalisation logique de cette fonction.



3. Donner une description flot de données en VHDL permettant de réaliser un multiplexeur 2 vers 1.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MUX2_1 is
    Port ( A : in  STD_LOGIC;
           D : out  STD_LOGIC_VECTOR(1 downto 0);
           S : out  STD_LOGIC);
end MUX2_1;

architecture Behavioral of MUX2_1 is

begin

S<= (D(0)and not (A)) or ( D(1) and A );

end Behavioral;
```

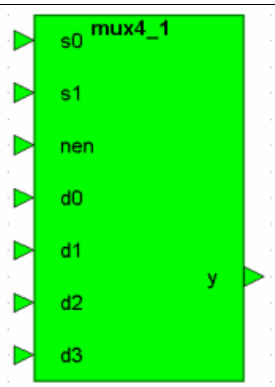
4. Multiplexeur 4 vers 1 avec entrée de validation :



Symbole CEI

operation	description	EN	S1	S0
DISABLE	Y = 0	0	-	-
SELECT 0	Y = D0	1	0	0
SELECT 1	Y = D1	1	0	1
SELECT 2	Y = D2	1	1	0
SELECT 3	Y = D3	1	1	1

Table des opérations



Symbole VHDL

- **Spécification d'entité :**

```
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24  use IEEE.numeric_std.all;
25
26  entity mux4_1 is
27      Port ( d0 : in STD_LOGIC;
28             d1 : in STD_LOGIC;
29             d2 : in STD_LOGIC;
30             d3 : in STD_LOGIC;
31             nen : in STD_LOGIC;
32             s0 : in STD_LOGIC;
33             s1 : in STD_LOGIC;
34             y : out STD_LOGIC);
35  end mux4_1;
```

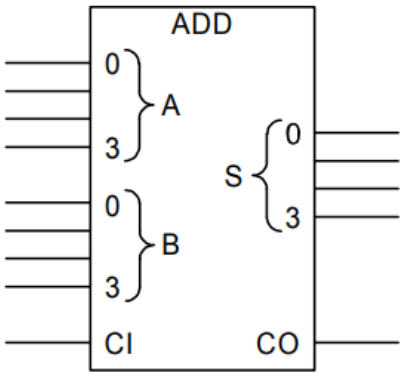
- **Architecture concurrente avec assignement conditionnel :**

```
architecture Behavioral of mux4_1 is
    signal en : std_logic;
    signal y_s : std_logic;
begin
    en <= not nen;
y_s <= d0 when s1='0' and s0='0' else
    d1 when s1='0' and s0='1' else
    d2 when s1='1' and s0='0' else
    d3 when s1='1' and s0='1' else
    'X';
y <= y_s when en ='1' else '0';

end Behavioral;
```

- **Architecture concurrente avec assignement sélectionné :**

```
architecture Behavioral of mux4_1 is

    signal en : std_logic;
    signal y_s : std_logic;
    signal sel : std_logic_vector(1 downto 0);
begin
    sel <= s1 & s0;
    en <= not nen;
    with sel select
        y_s <= d0 when "00",
               d1 when "01",
               d2 when "10",
               d3 when "11",
               'X' when others;
        y <= y_s when en='1' else '0';

end Behavioral;
```

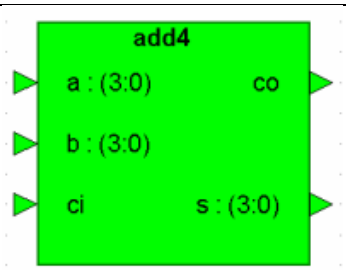
5. Additionneur de deux nombre de 4 bits avec retenue



Symbole CEI

operation	description
ADD	(CO,S) = A + B + CI

Table des opérations



Symbole VHDL

- **Spécification d'entité :**

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.NUMERIC_STD.ALL;
5
6  entity add is
7      Port ( a : in STD_LOGIC_VECTOR (3 downto 0);
8             b : in STD_LOGIC_VECTOR (3 downto 0);
9             ci : in STD_LOGIC;
10            co : out STD_LOGIC;
11            s : out STD_LOGIC_VECTOR (3 downto 0));
12 end add;
```

- **Architecture concurrente flot de données**

```
14  architecture Behavioral of add is
15      signal a_s, b_s, s_s : unsigned (4 downto 0);
16      signal ci_s : unsigned(0 downto 0);
17  begin
18      a_s <= unsigned('0' & a);
19      b_s <= unsigned('0' & b);
20      ci_s(0) <= ci;
21      s_s <= a_s + b_s + ci_s;
22
23      s <= std_logic_vector (s_s(3 downto 0));
24      co <= std_logic (s_s(4));
25
26  end Behavioral;
```