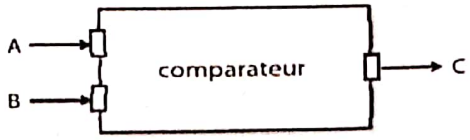
TD 2 : SLP

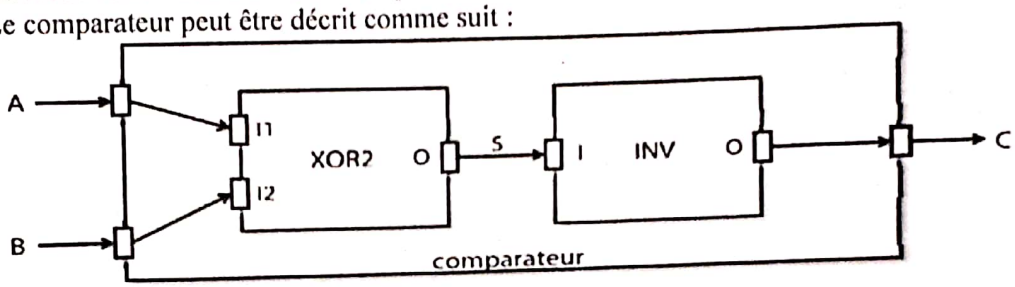
Exercice 1 :

Dans cet exercice, on va comparer l'égalité entre deux bits à l'aide de bloc **comp2** suivant :



Bloc **comp2.**

a) Ecrire le code VHDL du bloc **comp2** à l'aide d'une architecture flot.
b) Ecrire le code VHDL du bloc **comp2** à l'aide d'une architecture comportementale.
c) Le comparateur peut être décrit comme suit :

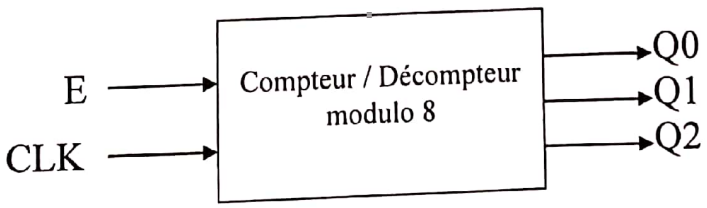


Ecrire le code VHDL du composant **XOR2**.

d) Ecrire le code VHDL du composant **INV**.
e) Réécrire le code VHDL du **comp2** à l'aide d'une architecture structurelle.

Exercice 2 :

On veut réaliser le circuit séquentiel synchrone illustré par la figure suivante :



avec :

- CLK est l'entrée d'horloge du circuit.
- Si E=0, le circuit est un compteur.
- Si E=1, le circuit est un décompteur.

1. Ecrire le code VHDL du circuit à l'aide d'une architecture comportementale.
2. Ecrire le code VHDL du composant **basc** (bascule D).

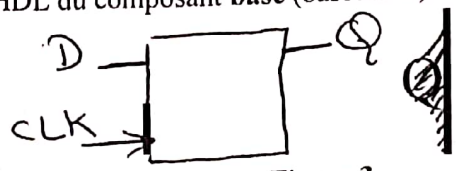


Figure 3.

3. Réécrire le code VHDL du circuit à l'aide d'une architecture structurelle.

Exercice 3

On veut réaliser le circuit illustré par la figure suivante.

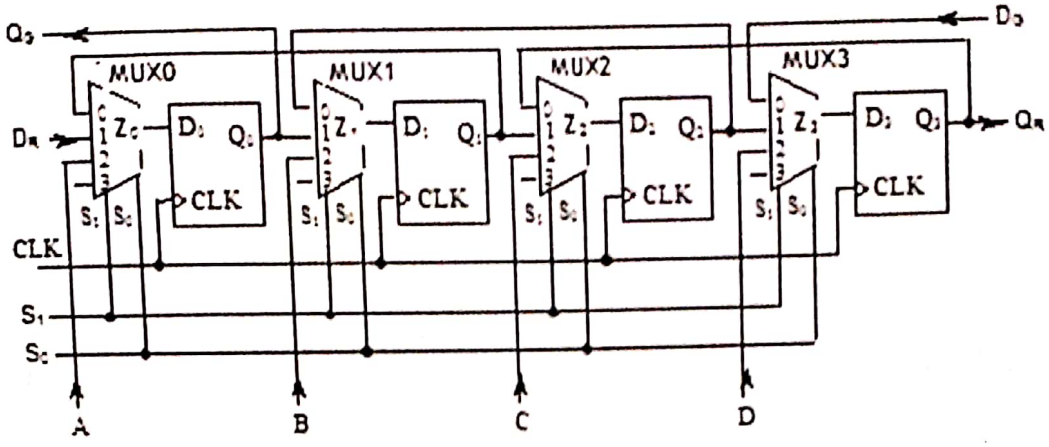


Figure 3.

1. Quel est le rôle de ce circuit.
2. Ecrire le code VHDL du circuit à l'aide d'une architecture comportementale.
3. Ecrire le code VHDL du composant **MUX** (multiplexeur 4 vers 1).

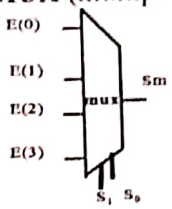


Figure 4.

4. Ecrire le code VHDL du composant **base** (bascule D).
5. Réécrire le code VHDL du circuit à l'aide d'une architecture structurelle.

TD n° ② : SLB

Exercice ① :

a)

entrée		sortie
a	b	c
0	0	1
0	1	0
1	0	0
1	1	1

$c = \bar{a}\bar{b} + ab$

$= \overline{a \oplus b} = a \odot b$

```
library ieee ;
use ieee. std_logic_1164.all;
Entity comp2 is
port ( a,b : in std_logic ;
       c : out std_logic);
end comp2 ;
-- architecture flot des données
   architecture arch_comp of comp2 is
   Begin
    c <= a nXOR B ;
   end arch_comp ;
```

b)

```
library ieee ;
use ieee. std_logic_1164. all;
Entity comp2 is
Port ( a,b : in std_logic ; c : out std_logic);
End comp2 ;
-- architecture comportementale
   architecture Archcomp2 of comp2 is
   begin
   -- concurrente
   c <= '1' when a=b else
        '0';
```

End Archcomp2;

-- equalité

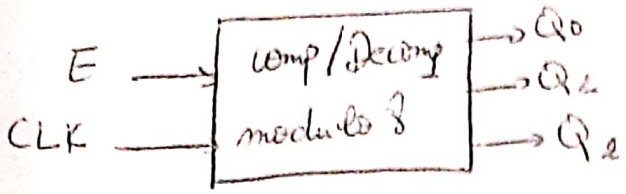
Process (a, b)
begin.
if a = b then c <= '1'.
else c <= '0'.
endif ;
end process ;

c) -- composant XOR 2.
d)
Library ieee ;
use ieee . std_logic_1164 . all ;
Entity XOR 2 is
port (I_1, I_2 : In std_logic ;
S : out std_logic) ;
end XOR 2 ;
architecture archXOR2 of XOR 2 is
Begin
O <= I_1 XOR I_2 ;
end archXOR2 ;

-- composant INV
Library ieee ;
use ieee. std_logic_1164 . all ;
entity inv is
port (I : In std_logic ;
O : out std_logic) ;
end inv ;
architecture archinv of inv is

```
Begin
    O <= not I;
end arch inv ;
library ieee ;
Use ieee . std_logic_1164 . all;
Entity comp2 is
Port ( a,b : in std_logic ;
            C : out std_logic);
End comp2 ;
-- architecture
Architecture Archcomp2 of Comp2 is
  Component XOR 2
  port (I1, I2 ; In std_logic ;
        O: out std_logic);
  End component;
  Component inv
  port ( I : In std_logic ;
         O : out std_logic);
  End Component;
For C1 : XOR 2 use entity work . XOR 2 (arch XOR 2);
For C2 : Inv use entity work . Inv (arch Inv);
  Signal S : std_logic ;
  Begin
  C1 : XOR 2 port map (A,B,S);
  C2 : inv port map (O=>C, I=>S);
  end archcomp 2 ;
```

Exercice ② :



1)

```
library ieee;
use ieee.std_logic_1164.all;
Entity CompDecomp is
port (E, CLK : In std_logic;
      Q0, Q1, Q2 : out std_logic);
End CompDecomp;
-- architecture comportementale.
Architecture archCompDecomp of CompDecomp is
Signal C : std_logic_vector (2 downto 0);
          C <= "000";
Begin
process (E, CLK)
Begin
   if E = '1' then
      if rising-edge (CLK) then
            C <= C + 1;
      end if.
   else
      if rising-edge (CLK) then
            C <= C - 1;
      end if.
   end if.
End process;
Q0 <= C(0);
   <= C(1);
```

... <= C(2);

... arch compDecomp;

Exercice (3) :

1) Registre à décalage à gauche (02)

2)
```
library ieee;
use ieee, std_logic_1164.all;
entity exemple is.
port (S0, S1, CLK, DR, DG, A, B, C, D : in std_logic;
                    QR, QG : out std_logic);
end exemple;
architecture archexple of exemple is
signal Z0, Q0, Z1, Q1, Z2, Q2, Z3, Q3 : std_logic;
                    Q : std_logic vecter (3 downto 0);
Begin. Q <= "0000"
Process (S0, S1, CLK)
Begin
Z(0) <= (not S1) and (not S1) and (D(1)) or (S0 and (not S1)
                    and (PR or (not S0) and (...S1) and A.
Z(1) <=
Z(2) <=
Z(3) <=
for i in 0 to 3 loop
if rising_edge (CLK) then
D(i) <= Z(i);
```

end if
end loop
Qe <= Q(3),
Qc <= D(0);
end process;
end arch_cycle