# TD 3

## Exercice 1

Soit le code VHDL suivant:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.std_arith.all;

entity PORTES is

port (A,B :in std_logic;
 Y1,Y2,Y3,Y4,Y5,Y6,Y7:out std_logic);
end PORTES;

architecture ARCH_PORTES of PORTES is

begin
  Y1 <= A and B;
  Y2 <= A or B;
  Y3 <= A xor B;
  Y4 <= not A;
  Y5 <= A nand B;
  Y6 <= A nor B;
  Y7 <= not(A xor B);

end ARCH_PORTES;
```

1. Représenter le schéma structurel de la fonction (on placera naturellement les entrées à gauche et les sorties à droite.

## Exercice 2

Soit le code VHDL suivant :

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL ;
ENTITY MACHIN IS
PORT (
Y0 : IN STD_LOGIC ;
 Y1 : IN STD_LOGIC ;
SEL : IN STD_LOGIC ;
Q : OUT STD_LOGIC ) ;
END MACHIN;
ARCHITECTURE ARCHI1 OF MACHIN IS
SIGNAL A : STD_LOGIC ;
SIGNAL B : STD_LOGIC ;
BEGIN
Q <= A OR B ;
A <= NOT SEL AND Y0 ;
B <= SEL AND Y1 ;
END ARCHI1
```

1. Dessiner la boîte noire correspondante au composant MACHIN décrit ci-dessous.
2. A l'intérieur de la boîte noire, dessiner, le schéma électronique en portes logiques.

## Exercice 3

1. Donner la table de vérité de O en fonction des entrées pour les architectures flot et comporte.
2. Donner le logigramme correspondant à l'architecture arch.

```
entity dec is
  port ( A, B, C, G1, G2A, G2B : in std_logic ;
       O : out unsigned (7 downto 0) );
end dec;
```

```
architecture flot of dec is
signal valid : std_logic ;
begin
valid <= G1 and not G2A and not
G2B ;
with valid & C & B & A select
  O <= "11111110" when "1000",
     "11111101" when "1001",
     "11111011" when "1010",
     "11110111" when "1011",
     "11101111" when "1100",
     "11011111" when "1101",
     "10111111" when "1110",
     "01111111" when "1111",
     "11111111" when others ;
end flot ;
```

```
architecture arch of dec  is
signal valid : std_logic ;
begin
  valid <= G1 and not G2A and not G2B ;
  O(0) <= not (valid and not A and not B and not C );
  O(1) <= not (valid and     A and not B and not C );
  O(2) <= not (valid and not A and     B and not C );
  O(3) <= not (valid and     A and     B and not C );
  O(4) <= not (valid and not A and not B and     C );
  O(5) <= not (valid and     A and not B and     C );
  O(6) <= not (valid and not A and     B and     C );
  O(7) <= not (valid and     A and     B and     C );
end arch ;
```

```
architecture comporte of dec  is
 signal valid : std_logic;
 signal select : unsigned (3 downto 0);
```

```
begin
 valid <= (G1 and not G2A and not G2B);
 select <= C & B & A;
 decode : process (select, valid)
 begin
  if (valid='0') then
    O <= (others => '1' ) ;
  else
    case select is
      when "000" => O <= "11111110" ;
      when "001" => O <= "11111101" ;
      when "010" => O <= "11111011" ;
      when "011" => O <= "11110111" ;
      when "100" => O <= "11101111" ;
      when "101" => O <= "11011111" ;
      when "110" => O <= "10111111" ;
      when "111" => O <= "01111111" ;
      when others => NULL;
    end case ;
  end if ;
 end process decode ;
end comporte ;
```

## Exercice 4

Soit le code VHDL suivant qui correspond à la figure 1.

```
...
ENTITY structurelle IS
PORT ( A, B  : IN     STD_LOGIC;
C, D  : OUT   STD_LOGIC);
END structurelle ;
ARCHITECTURE rtl OF structurelle IS
-- Déclaration des composantes
COMPONENT module1
PORT( a,b: IN     STD_LOGIC;
c:   OUT  STD_LOGIC;);
END COMPONENT;
COMPONENT module2
PORT( a,b: STD_LOGIC;
c:   OUT  STD_LOGIC;);
END COMPONENT;
SIGNAL t1, t2: IN  STD_LOGIC;
BEGIN
-- Instanciation des composantes
U1 : ……;
U2 : ……;
……;
END rtl;
```

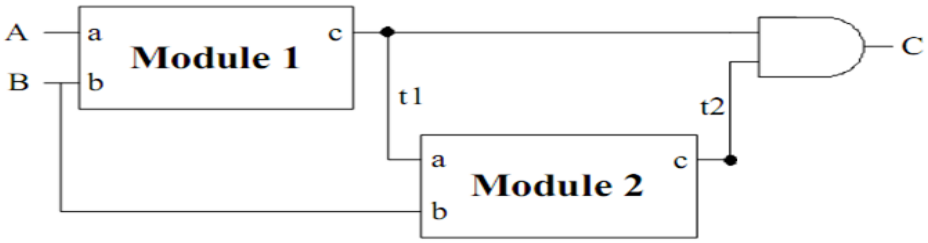


Figure 1

1. Compléter les lignes du code (Instanciation des composantes).

**Exercice 5**

Soit la description VHDL suivante

```
-- fonction.vhd
library ieee;
use ieee.std_logic_1164.all;
entity fct is
  port (a, b : in std_logic;
        s : out std_logic);
end fct;
architecture archi_fct of fct is
begin
  s <= '1' when a=b else '0';
end archi_fct;
```

Dans le même projet, on écrit la description VHDL suivante:

```
-- function.vhd
library ieee;
use ieee.std_logic_1164.all;
entity circuit is
  port (e1, e2, e3, e4 : in std_logic;
        s : out std_logic);
end circuit;
architecture archi_circuit of circuit is
signal s1, s2, s3 : std_logic;
component fct
  port (a, b : in std_logic;
        s : out std_logic);
end component;
begin
  cmp1 : fct port map (a=>e1, b=>e2, s=>s1);
  cmp2 : fct port map (a=>e3, b=>e4, s=>s2);
  cmp3 : fct port map (a=>e1, b=>e3, s=>s3);
  s <= s1 and s2 and s3;
end archi_circuit;
```

1. Faire une analyse structurelle (boîte noire avec entrées/sorties, chemins de données internes, représentation symbolique des blocs internes);

## Exercice 6

On considère le programme ci-dessous :

```
entity transitm is
        port ( hor, e : in bit ;
                s : out bit );
end transitm ;
architecture arch of transitm is
        signal qa, qb : bit ;
      begin
        s <= qa xor qb ;
        schem : process
        begin
          wait until hor = '1' ;
          qa <= e ;
          qb <= qa ;
        end process schem ;
end arch ;
```

1. Déduire de ce programme, par une construction méthodique, un schéma (bascules et portes logiques).
2. Compléter le chronogramme ci-dessous.