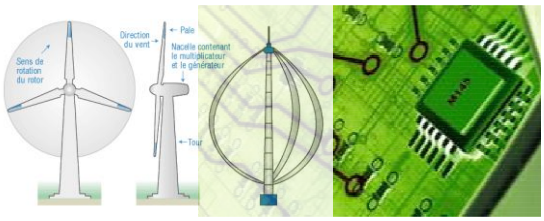
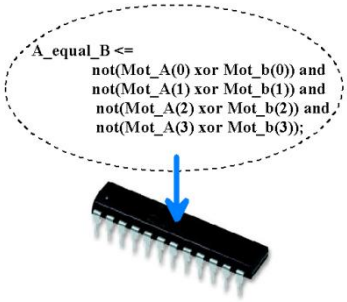


ELECTRONIQUE NUMERIQUE PAL EPLD PSOC

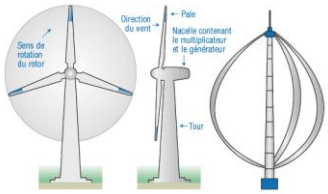


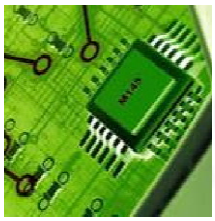
Initiation à la logique programmable

📖Indique un document ressource

⇦Retour au sommaire

⇦ Retour à la page courante

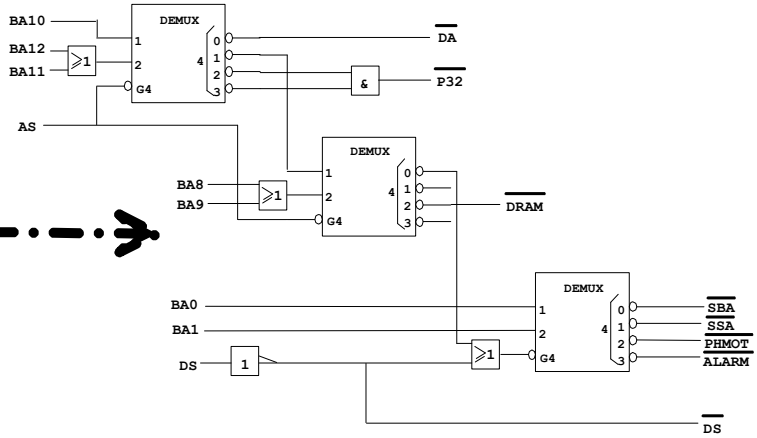




1 Logique programmable

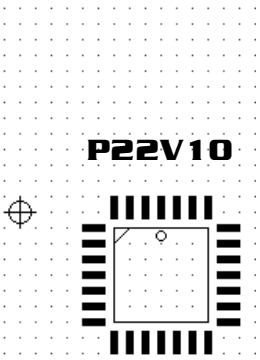
Synthèse d'un schéma logique complexe :

Du schéma

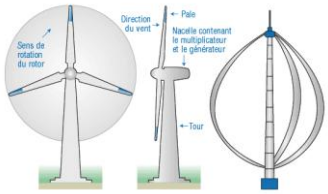


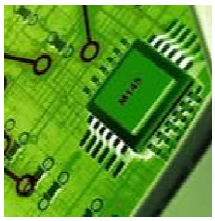
DECODAGE AVEC DES BOITIERS DE LOGIQUE COMBINATOIRE

Vers le PAL



DECODAGE AVEC UN UNIQUE BOITIER DE LOGIQUE PROGRAMMABLE

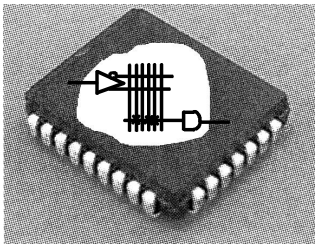




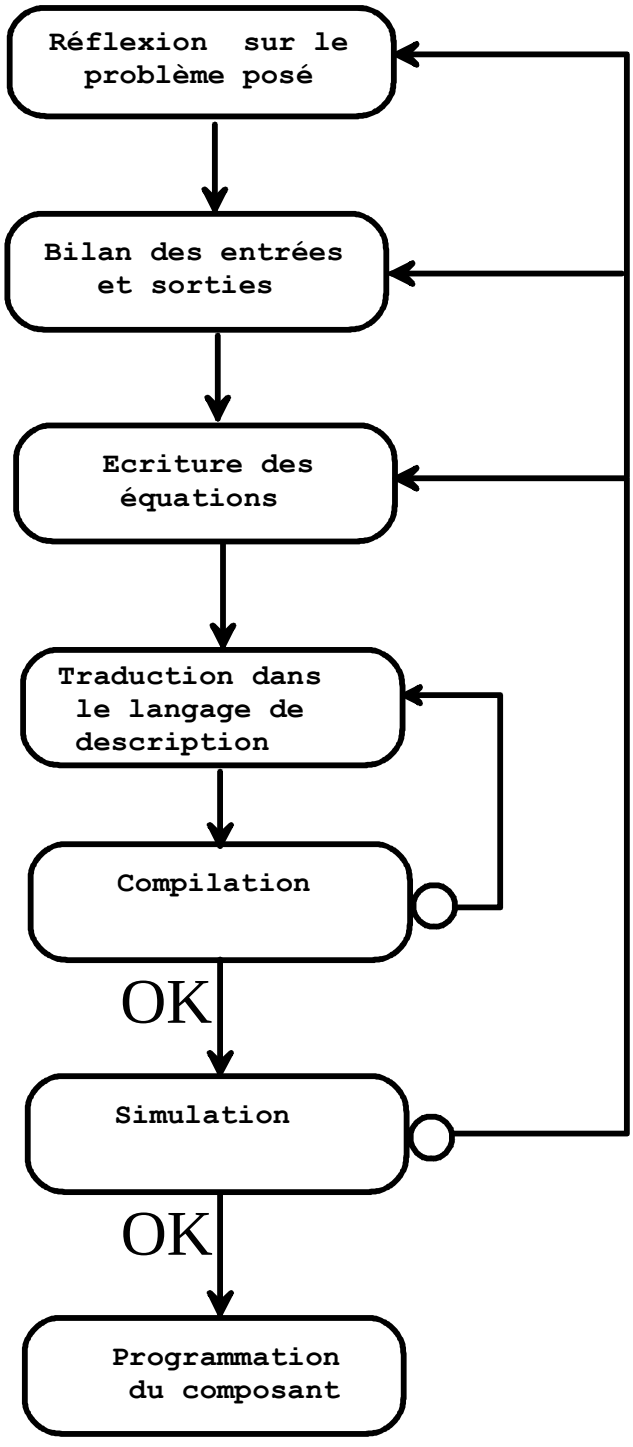
Principe de la synthèse

Projet

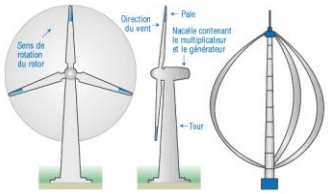
VHDL

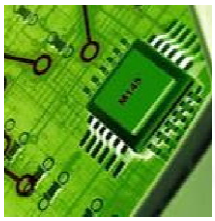


Composant

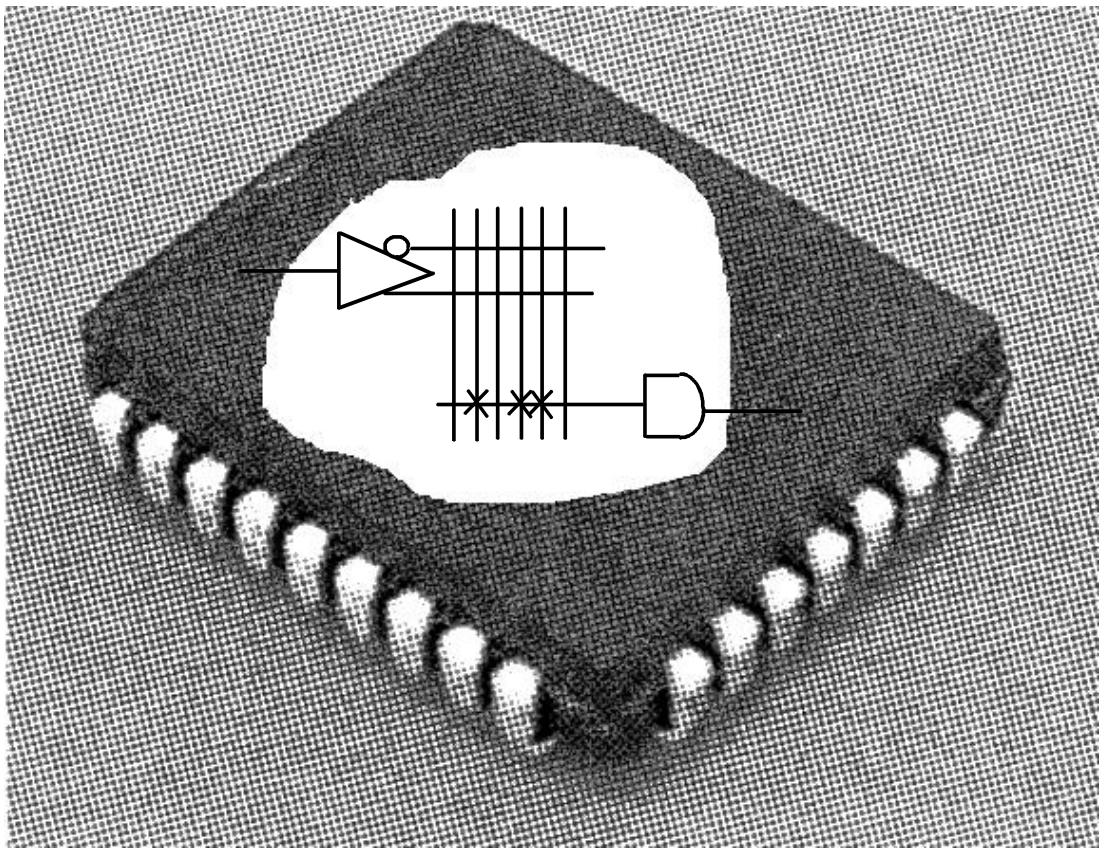


STI2D SIN



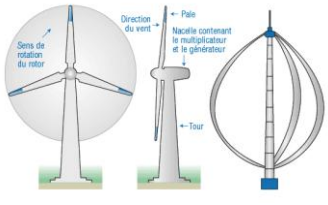


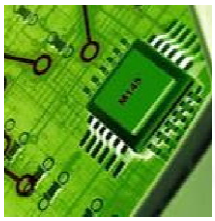
Un seul composant



=> contient de multiples équations logiques

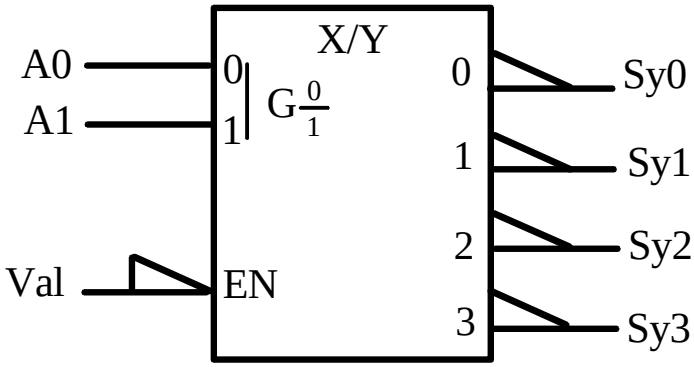
=> remplace plusieurs composants traditionnels



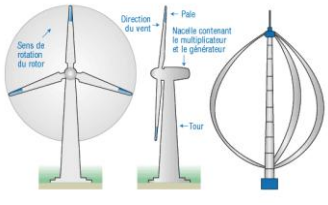


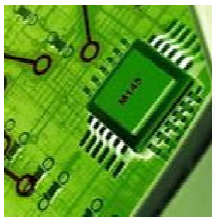
2 Le vhdl

Un langage de description mais on réalise la simulation et la synthèse :




```
3   library ieee;
4   use ieee.std_logic_1164.all;
5
6   entity decod1_4 is
7       port (
8           a0,a1 : in  std_logic;
9           val   : in  std_logic;
10          sy    : out std_logic_vector(3 downto 0)
11          );
12  end decod1_4;
13
14  library ieee;
15  use ieee.std_logic_1164.all;
16
17  architecture equation of decod1_4 is
18  begin
19
20      sy(0)<= not(not(val) and  not(a1) and not(a0));
21      sy(1)<= not(not(val) and  not(a1) and     a0 );
22      sy(2)<= not(not(val) and      a1  and not(a0) );
23      sy(3)<= not(not(val) and      a1  and     a0 );
24
25  end equation;
```

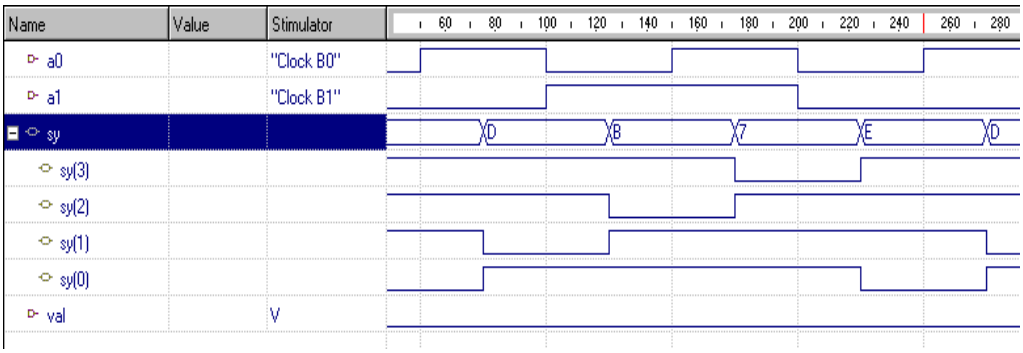




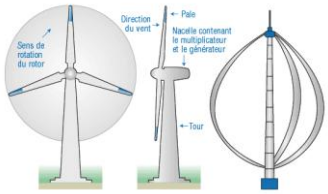
Fichier de programmation

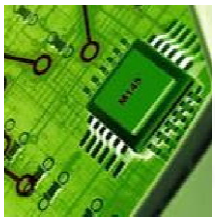
```
NOTE DEVICE C22V10*
NOTE PACKAGE palce22v10-5jc*
NOTE PROPERTY BUS_HOLD ENABLE*
NOTE PINS  val:2 a1:3 a0:4 sy(3):17 sy(1):18 sy(0):26 sy(2):27 *
NOTE PINS *
NOTE NODES *
L00000
0000000000000000000000000000000000000000000000
* Node val[1] => BANK : 1 *

L00044
1111111111111111111111111111111111111111111111
1011011110111111111111111111111111111111111111
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
* Node sy(2)[23] => OE : 1 ,LOGIC : 8 *

L00440
1111111111111111111111111111111111111111111111
1011011110111111111111111111111111111111111111
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
```

Simulation



STI2D SIN





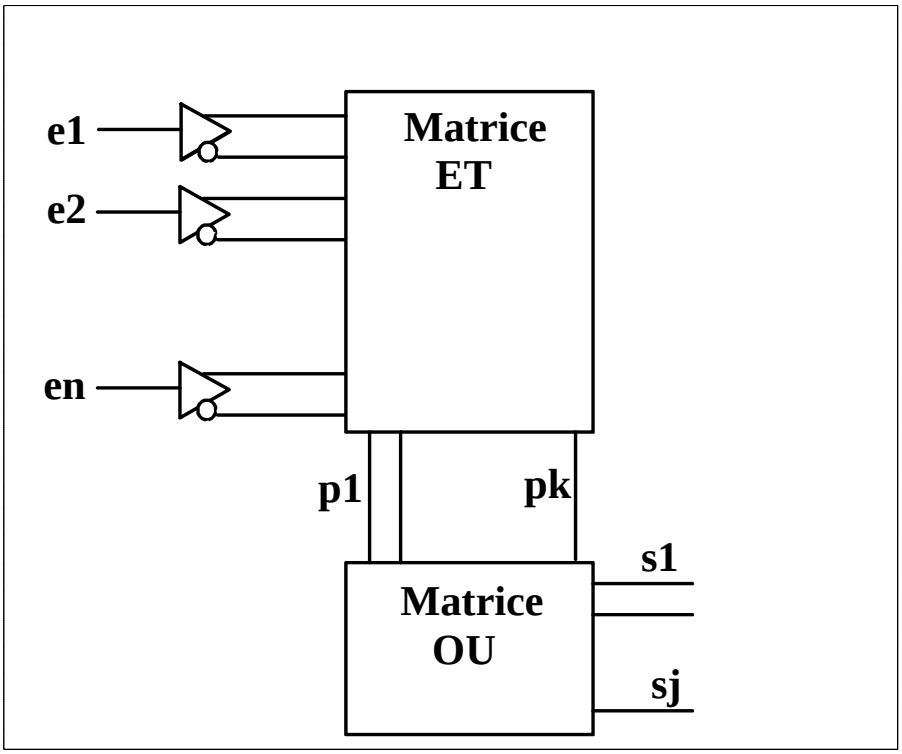

3 Structure des Pals combinatoires

Forme générale d'une équation logique :

$$F = b.\overline{c}.d + \overline{e}.f.d + \overline{a}.g$$

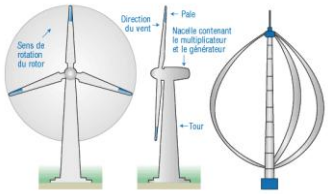
Somme de produits

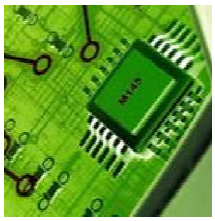
Structure générale d'un Pal :



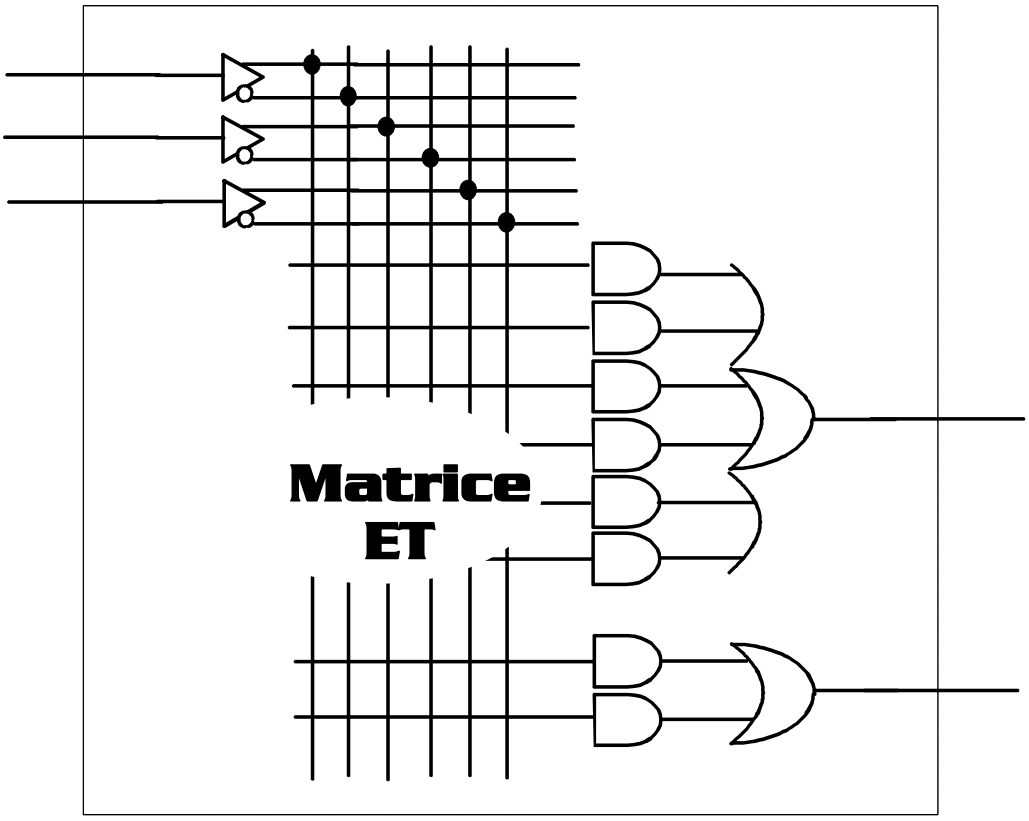
Somme de produits

STI2D SIN





Structure générale d'un Pal :



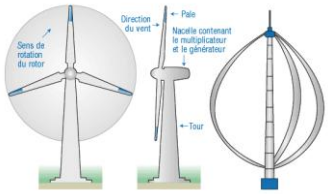
Bilan :

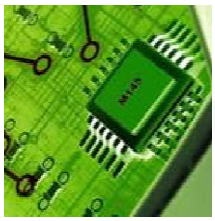
=> nombre d'entrées

=> nombre de sorties

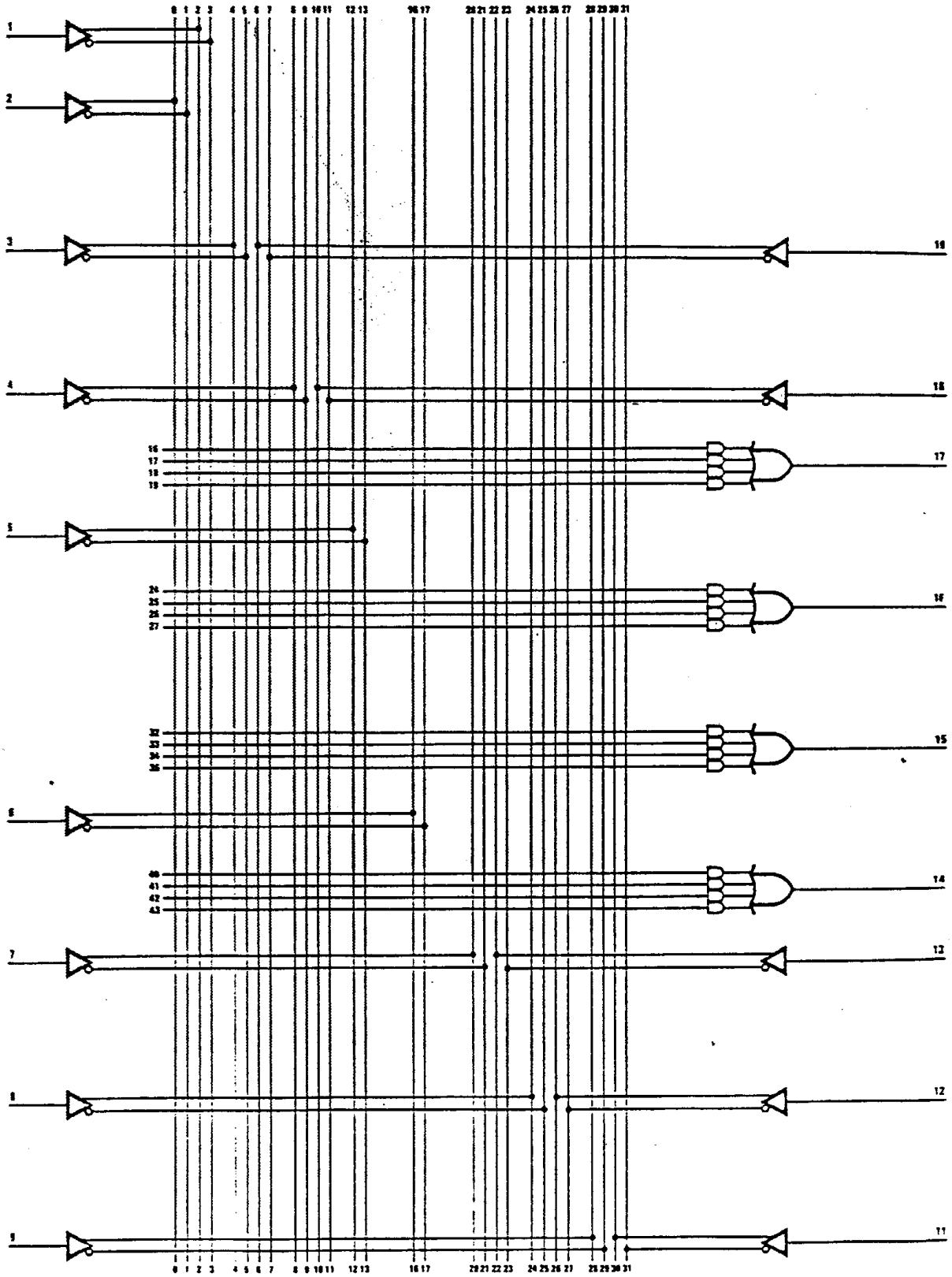
=> nombre de termes porduits/sortie

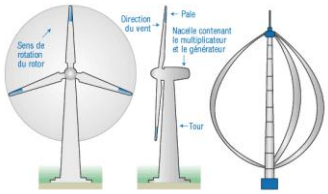
STI2D SIN

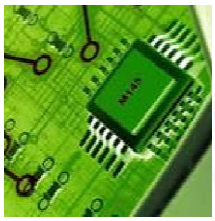




Logic Diagram **14H4**



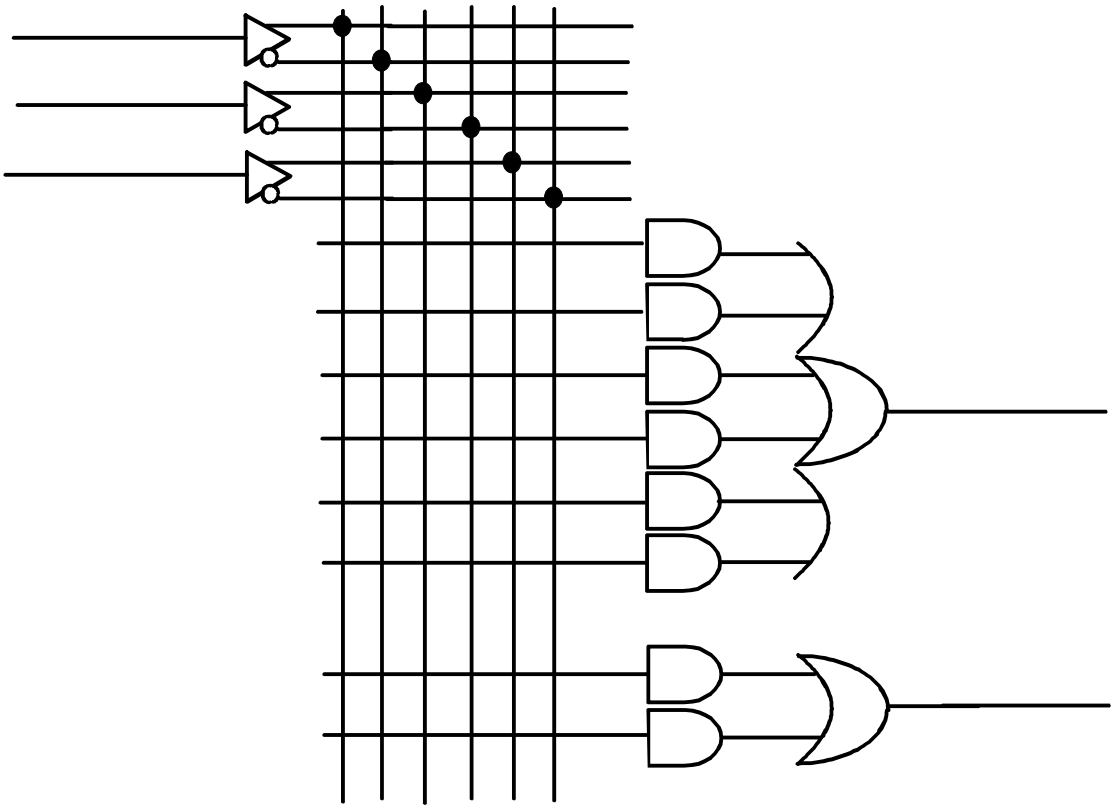




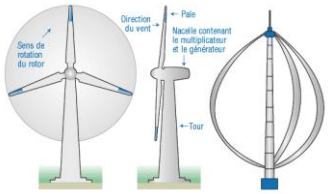
4 Exercices d'implantation

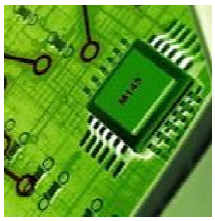
4.1 Exemple n° 1 :

$$F = \overline{b}.c.d \qquad G = b + c.\overline{d}$$



STI2D SIN

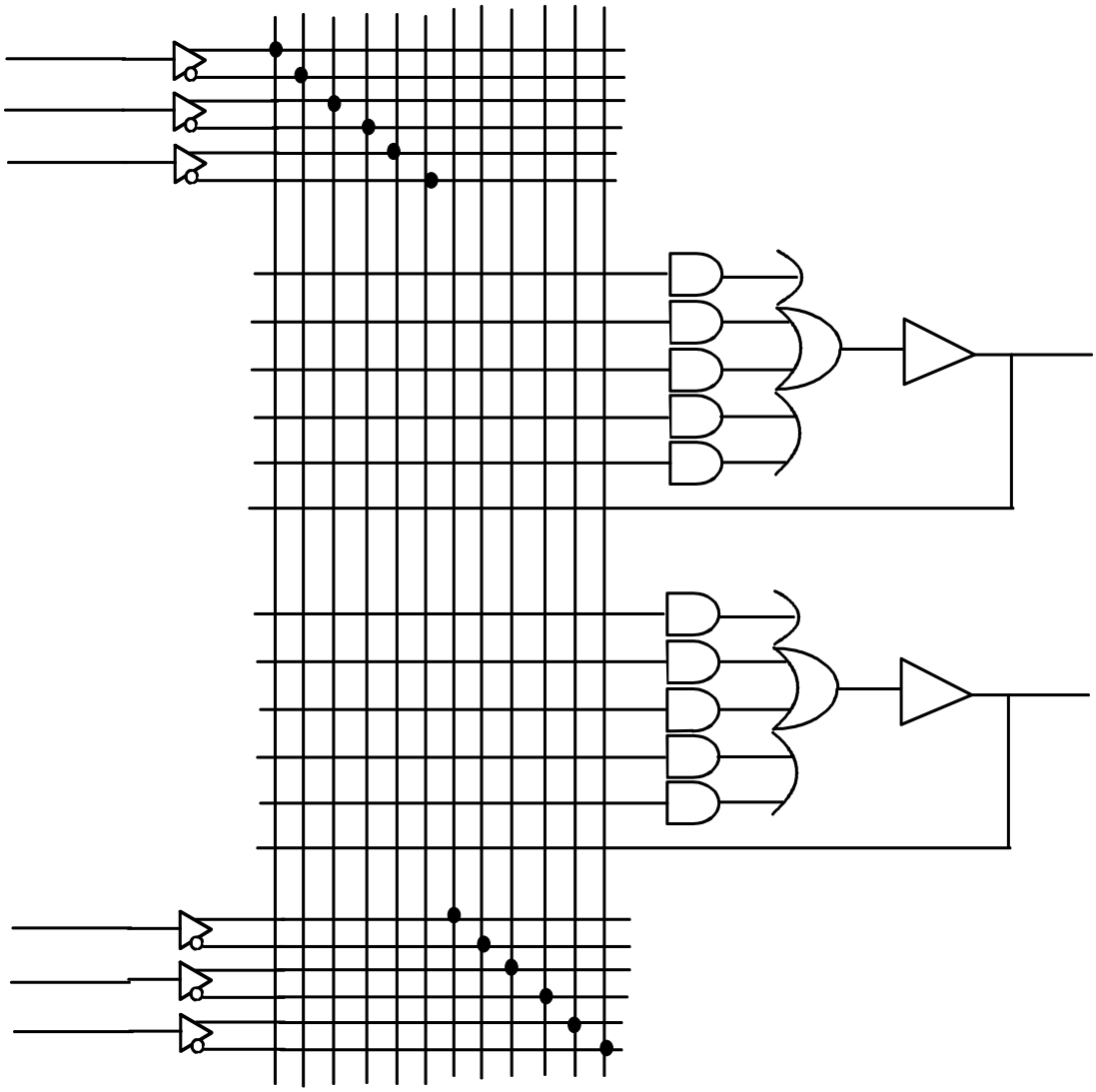




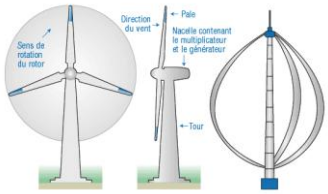
4.2 Exemple n° 2 :

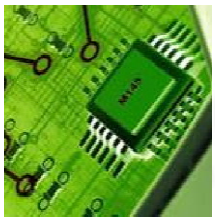
$$F = \overline{b}.c.d + a.\overline{d}$$

$$G = \overline{b}.\overline{c}+b.c$$



STI2D SIN

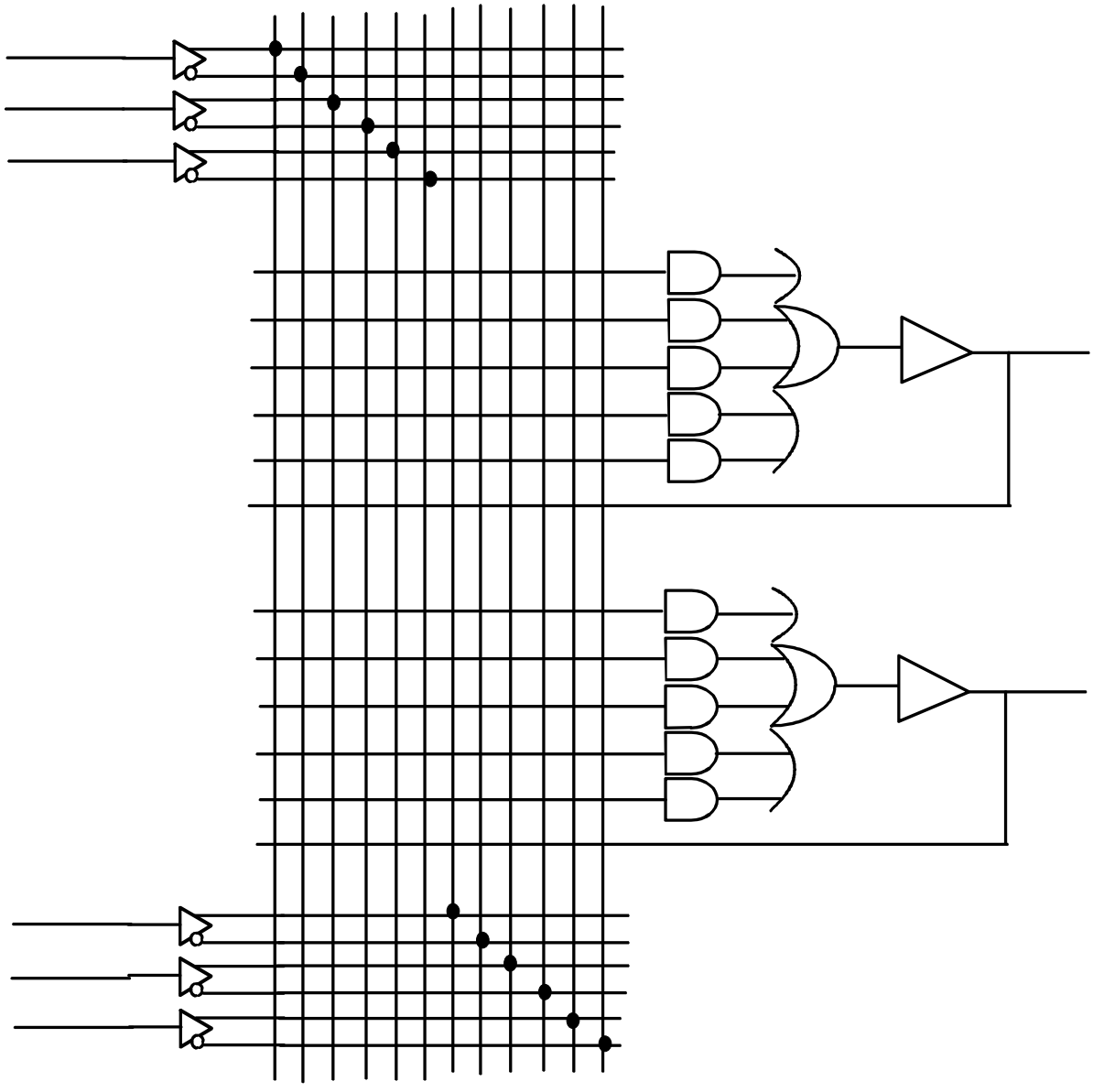




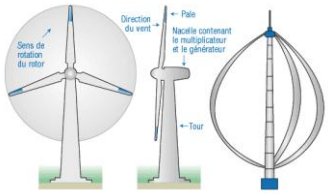
4.3 Exemple n° 3 :

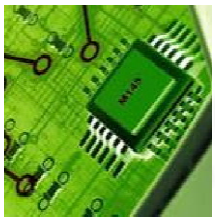
$$F = \overline{b}.c.d + e.\overline{d} + f$$

$$G = \overline{e}.\overline{c} + b.c$$



STI2D SIN






5 Mini projet : test de l'égalité de deux mots binaires

Test de l'égalité de deux mots de 4 bits

A = [a3 a2 a1 a0]

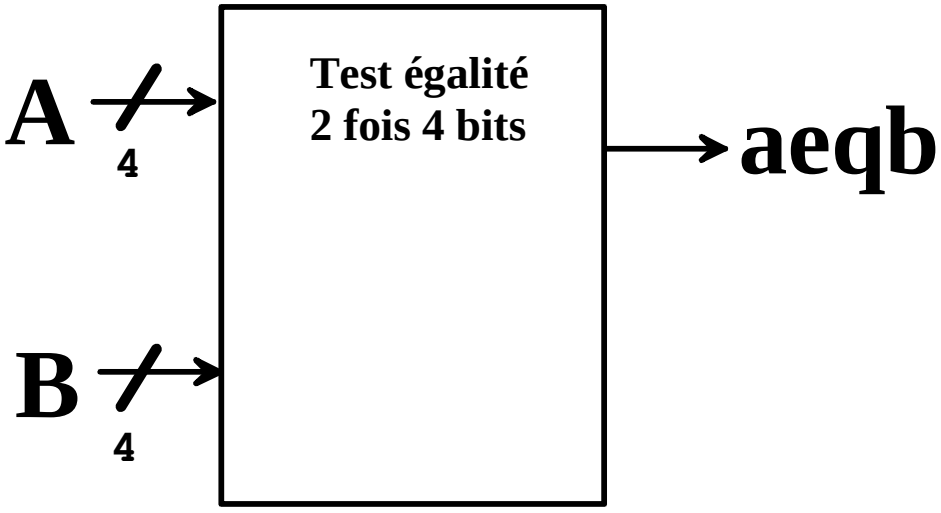
B = [b3 b2 b1 b0]

Egalité de deux bits :

$$aeqb = \overline{a}.\overline{b} + a.b$$

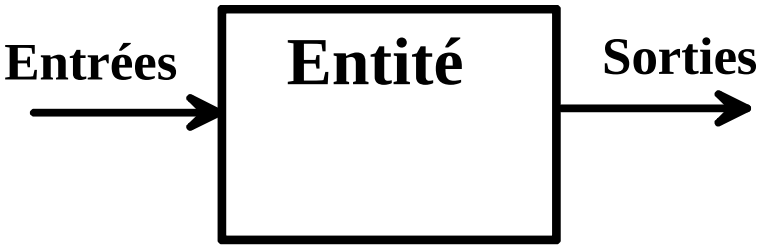
b	a	aeqb
0	0	1
0	1	0
1	0	0
1	1	1

Bilan des entrées sorties :



5.1 Donner les équations :

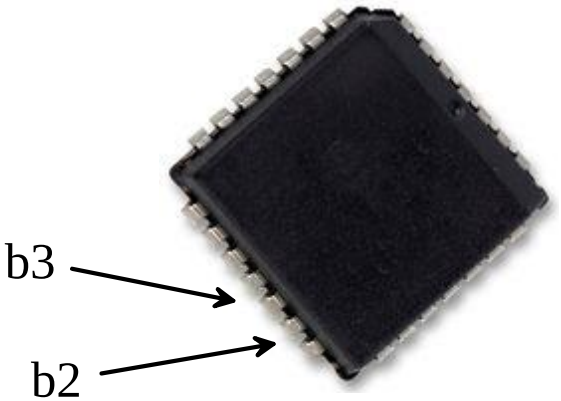
5.2 La liste des entrées sorties :



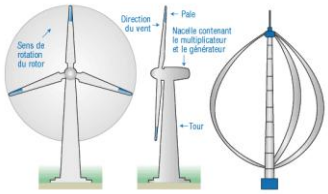
5.3 Le port :

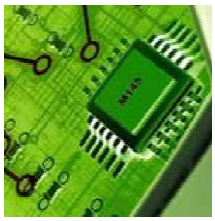
```
entity egalite_2x4 is

        port (
                Mot_A : std_logic_vector ( 3 downto 0 );
                Mot_B : std_logic_vector ( 3 downto 0 );
                A_equal_B  : out std_logic);
```

5.4 Le brochage :



STI2D SIN

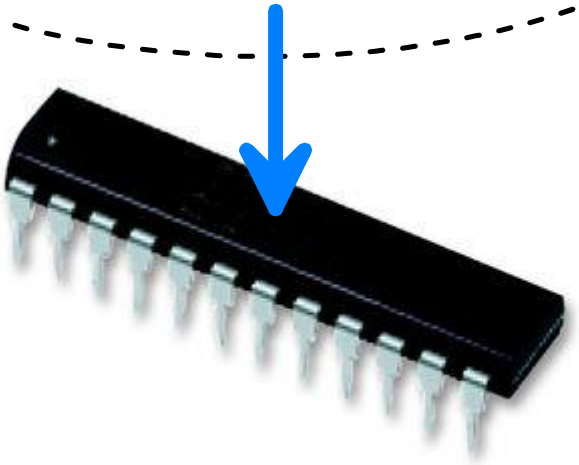




5.5 L'architecture :

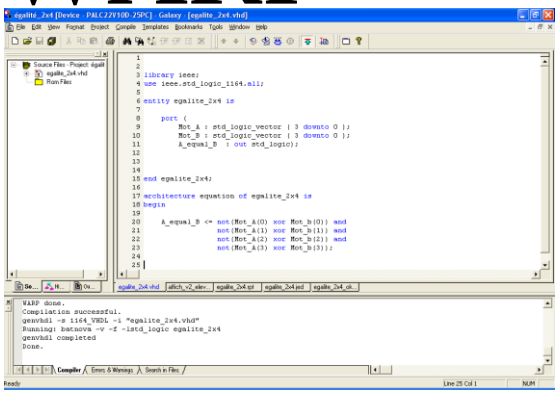
Architecture

```
A_equal_B <=
        not(Mot_A(0) xor Mot_b(0)) and
        not(Mot_A(1) xor Mot_b(1)) and
        not(Mot_A(2) xor Mot_b(2)) and
        not(Mot_A(3) xor Mot_b(3));
```



6 Fiche : Flux de simulation Warp => ISIS

WARP

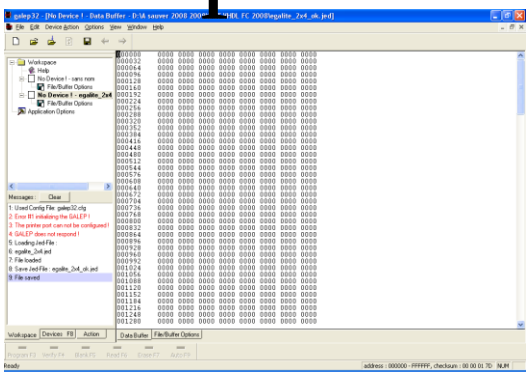


Fichier jedec

Fichier texte

<fichier>.jed

Galep32

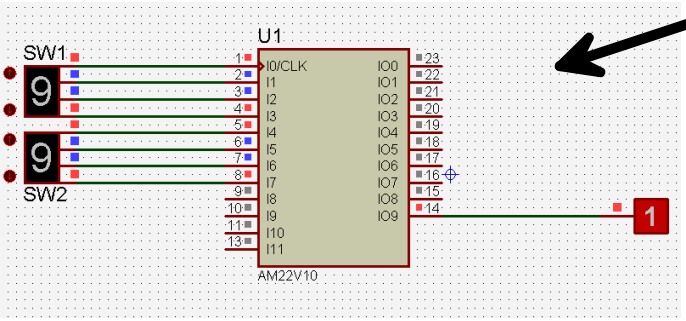


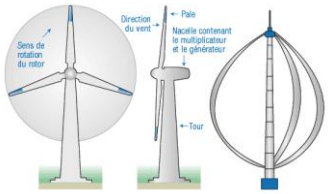
Fichier jedec 'basic'

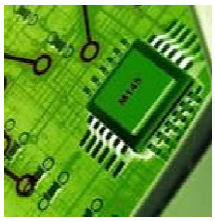
Fichier texte

<fichier>.jed

ISIS

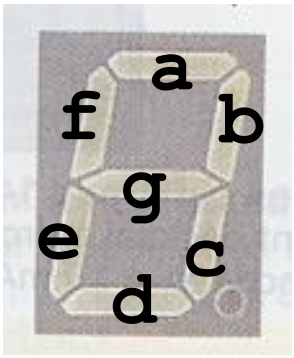






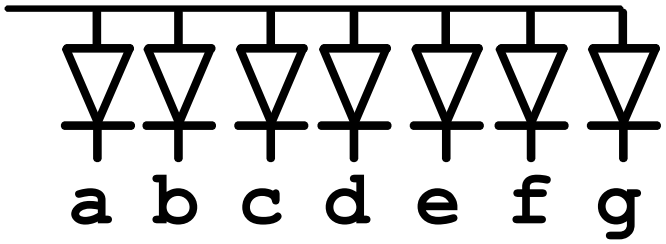
7 Mini projet de synthèse d'une commande d'afficheur

Commande afficheur 7 segments

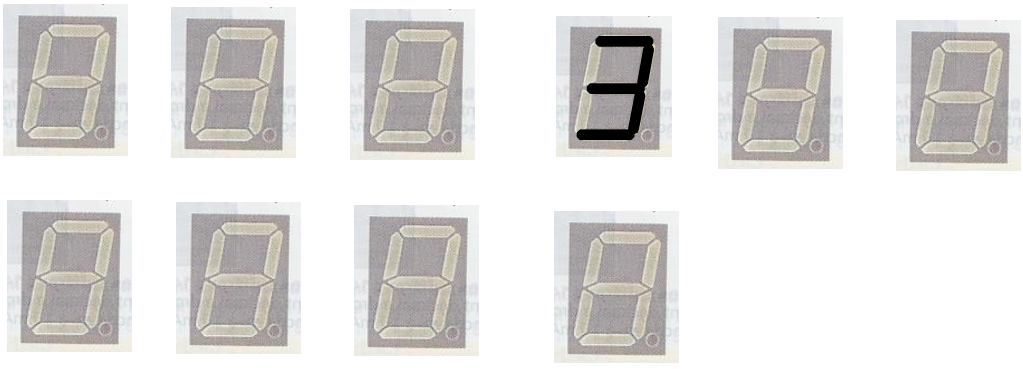


Point commun ici :

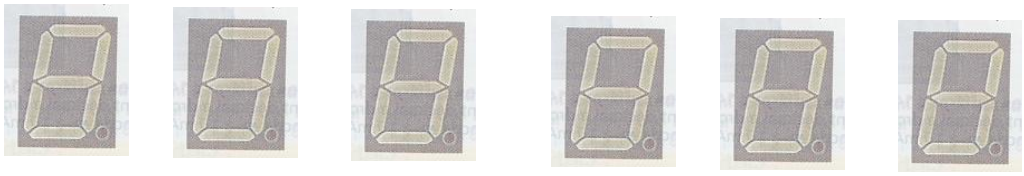
Anodes Communes



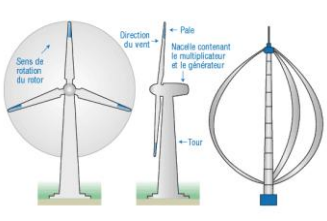
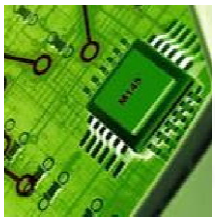
Les chiffres de 0 à 9

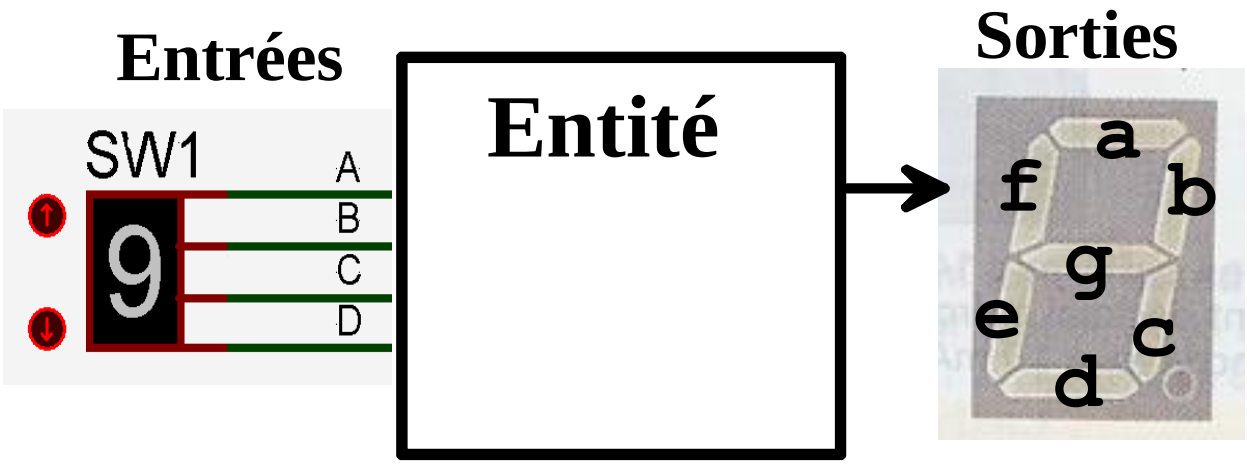


Les lettres de A à F



STI2D SIN

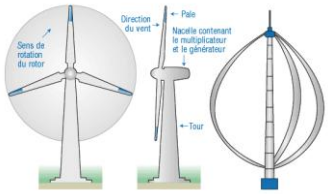



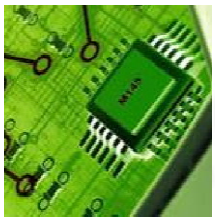



7.1 La liste des entrées sorties :

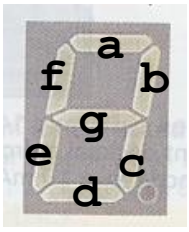
```
 8  entity af_7seg is
 9
10      port (
11          E_test_segment : in std_logic;
12          Ea_a,
13          Ea_b,
14          Ea_c,
15          Ea_d     : in std_logic;
16          Dig1_sa,
17          Dig1_sb,
18          Dig1_sc,
19          Dig1_sd,
20          Dig1_se,
21          Dig1_sf,
22          Dig1_sg : out std_logic);
23
24  end af_7seg;
```

STI2D SIN

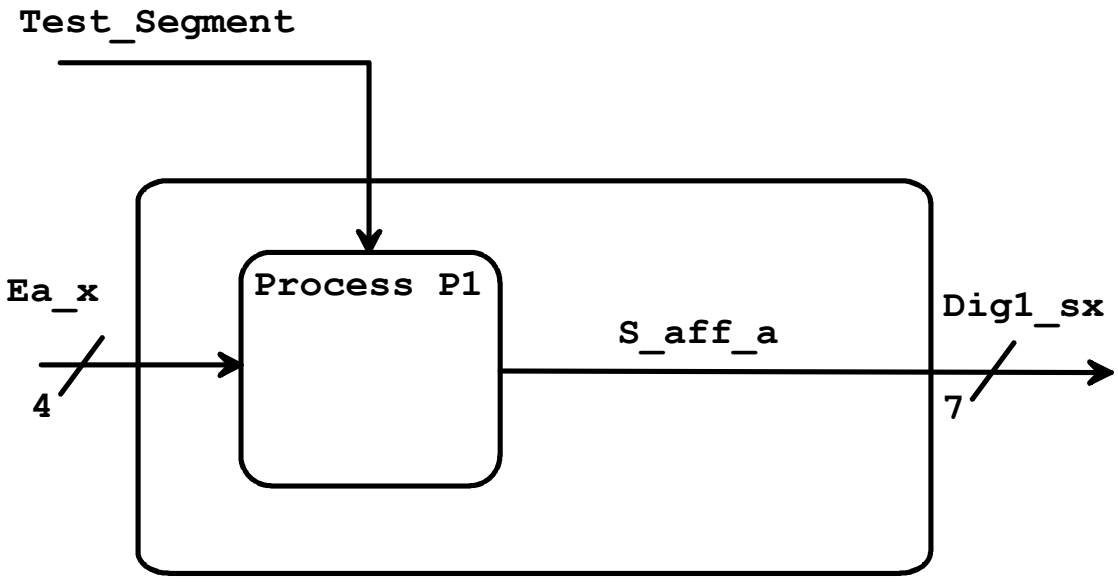




7.2 L'architecture :

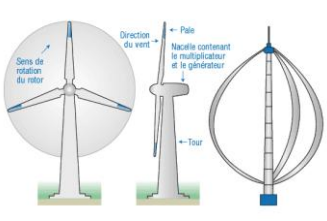
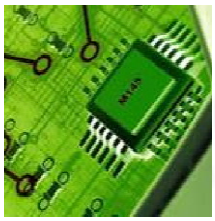


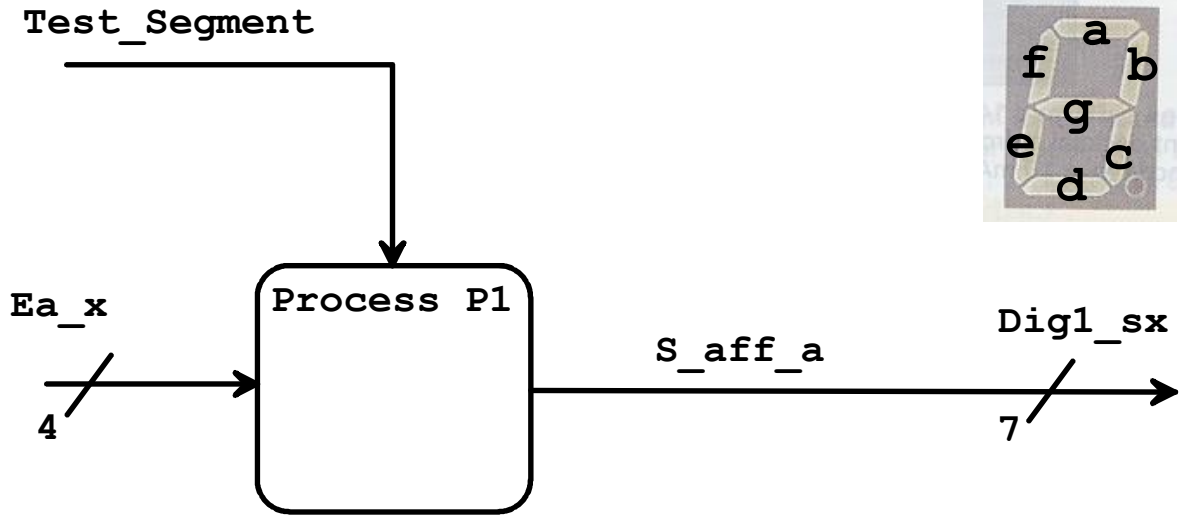
Architecture



Architecture Organisation générale

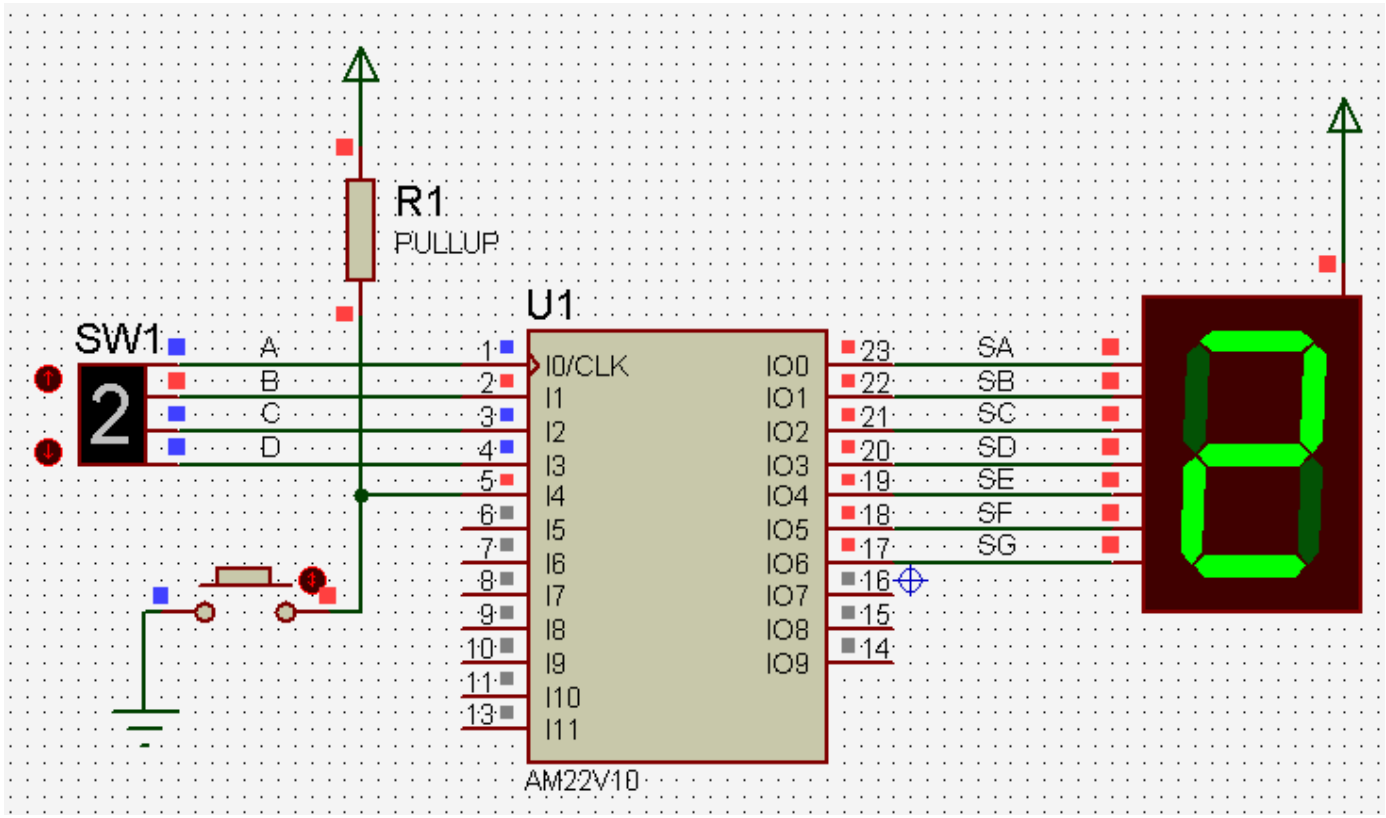
STI2D SIN

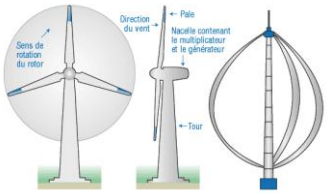


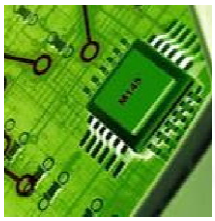

```
45      p1 : process (E_test_segment,E_a)
46          begin                                           -- gfedcba
47              if (E_test_segment = '0') then S_aff_a <= "0000000";--$00
48                  elsif (E_A = "0000") then S_aff_a <= "1000000"; --$
49                  elsif (E_A = "0001") then S_aff_a <= "1111001"; --$
50                  elsif (E_A = "0010") then S_aff_a <= "0100100"; --$
51                  elsif (E_A = "0011") then S_aff_a <= "1100000"; --$
52                  elsif (E_A = "0100") then S_aff_a <= "0110001"; --$
53                  elsif (E_A = "0101") then S_aff_a <= "0010001"; --$
54                  elsif (E_A = "0110") then S_aff_a <= "1000001"; --$
55                  elsif (E_A = "0111") then S_aff_a <= "0111100"; --$
56                  elsif (E_A = "1000") then S_aff_a <= "1000000"; --$
57                  elsif (E_A = "1001") then S_aff_a <= "1110011"; --$
58
59                  else S_aff_a <= "1111111";                  --$
60              end if;
61      end process;
```

STI2D SIN


R1
PULLUP
U1
SW1
A
B
C
D
2
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
SA
SB
SC
SD
SE
SF
SG
AM22V10

STI2D SIN

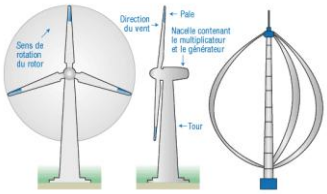


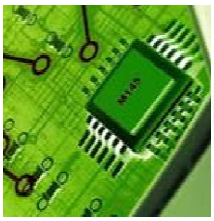



7.3 Le texte complet de l'exemple 2 :

```
library ieee;
use ieee.std_logic_1164.all;



entity af_7seg is

        port (
                E_test_segment : in std_logic;
                Ea_a,
                Ea_b,
                Ea_c,
                Ea_d    : in std_logic;
                Dig1_sa,
                Dig1_sb,
                Dig1_sc,
                Dig1_sd,
                Dig1_se,
                Dig1_sf,
                Dig1_sg : out std_logic);

end af_7seg;

architecture composant of af_7seg is

        signal S_aff_a : std_logic_vector(6 downto 0);
        signal zero    : std_logic :='0';
        signal E_A     : std_logic_vector (3 downto 0);
        signal E_B     : std_logic_vector (3 downto 0);

begin

        E_A <= Ea_d & Ea_c & Ea_b & Ea_a;

                Dig1_sa <= S_aff_a (0);
                Dig1_sb <= S_aff_a (1);
                Dig1_sc <= S_aff_a (2);
                Dig1_sd <= S_aff_a (3);
                Dig1_se <= S_aff_a (4);
                Dig1_sf <= S_aff_a (5);
                Dig1_sg <= S_aff_a (6);
```

STI2D SIN





```
p1 : process (E_test_segment,E_a)
        begin                                   -- gfedcba
                if (E_test_segment = '0') then S_aff_a <= "0000000";--$00
                        elsif (E_A = "0000") then S_aff_a <= "1000000"; --$
                        elsif (E_A = "0001") then S_aff_a <= "1111001"; --$
                        elsif (E_A = "0010") then S_aff_a <= "0100100"; --$
                        elsif (E_A = "0011") then S_aff_a <= "1100000"; --$
                        elsif (E_A = "0100") then S_aff_a <= "0110001"; --$
                        elsif (E_A = "0101") then S_aff_a <= "0010001"; --$
                        elsif (E_A = "0110") then S_aff_a <= "1000001"; --$
                        elsif (E_A = "0111") then S_aff_a <= "0111100"; --$
                        elsif (E_A = "1000") then S_aff_a <= "1000000"; --$
                        elsif (E_A = "1001") then S_aff_a <= "1110011"; --$

                else S_aff_a <= "1111111";              --$
                 end if;
end process;


end composant;
```