Université de Boumerdès

Automatique-M2

Module: FPGA & VHDL

Chapitre 2: Les circuits numériques

Dr. Belkacem Samia

2018/2019

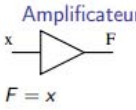
Plan

1. Introduction

Les circuits standards

Portes élémentaires

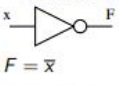
Amplificateur :



$F = x$

x	F
0	0
1	1

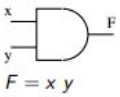
Inverseur :



$F = \overline{x}$

x	F
0	1
1	0

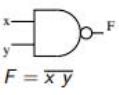
ET :



$F = x\,y$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

NON ET :



$F = \overline{x\,y}$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

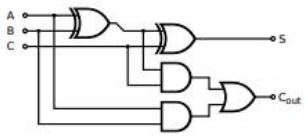
Les circuits standards

Conception de circuit combinatoire

1. Description du problème :
   Addition entre deux bits a et b et une retenue c
2. Table de vérité :

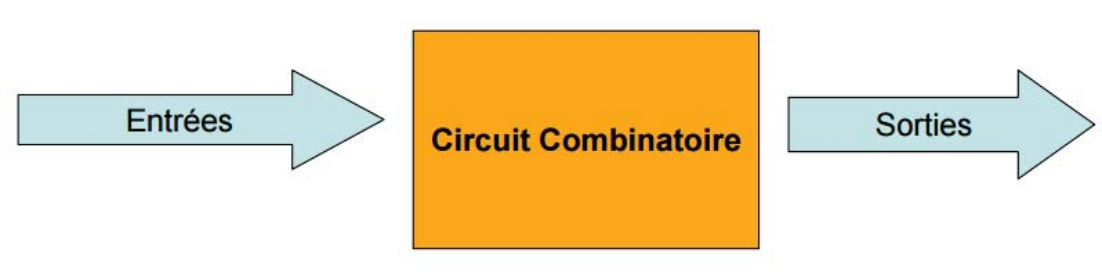
entrées			sorties	
a	b	c	S	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

3. Équations logiques :
   $S = \overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\overline{c} + abc$
   $c_{out} = \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$
4. Équations simplifiées :
   $S = (a \oplus b) \oplus c$
   $c_{out} = ab + (a \oplus b)c$
5. Portes logiques :



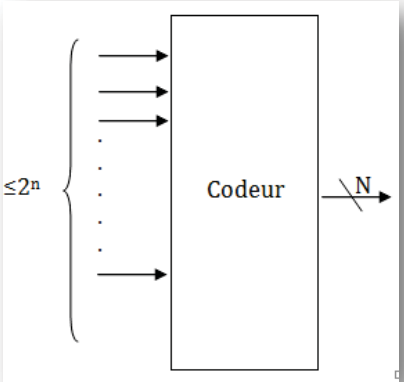
Systèmes combinatoires

- Systèmes combinatoires : Les sorties ne dépendent que des entrées.
- Les circuits logiques sont élaborés à partir de composants électroniques – transistors.



Les codeurs

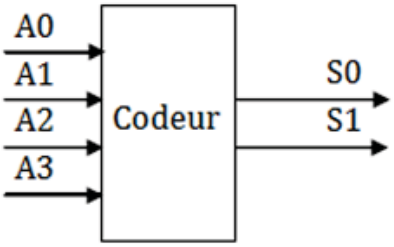
- Un codeur ou encodeur est un circuit logique
- possède 2^n voies d'entrées dont une seul est active.
- N voies de sorties.



Les codeurs

- Exemple: Codeur 4 voies d'entrées et 2 bits de sortie

Schéma fonctionnel	La table de vérité



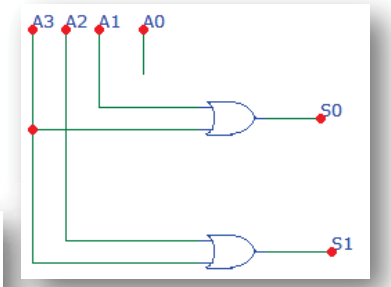
A3	A2	A1	A0	S1	S0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Les codeurs

- Exemple: Codeur 4 voies d'entrées et 2 bits de sortie

A_4A_3 \ A_1A_0	00	01	11	10
00	X	0	X	1
01	0	X	X	X
11	X	X	X	X
10	1	X	X	X

$S0 = A1 + A3$



A_4A_3 \ A_1A_0	00	01	11	10
00	X	0	X	0
01	1	X	X	X
11	X	X	X	X
10	1	X	X	X

$S1 = A2 + A3$

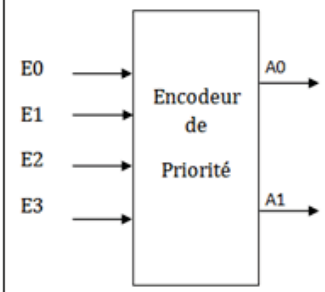
Les encodeurs de priorité

- Si plusieurs entrées sont mises à 1 simultanément, le codeur classique donne un résultat erroné.
- On utilisera un codeur prioritaire (appelé aussi encodeur de priorité).
- Ce type de codeur fixe un ordre de priorité entre les entrées.
- Pour un codage en binaire pur, le codeur prioritaire donne la priorité à l'entrée de poids le plus élevé.

Les encodeurs de priorité

- Encodeur de priorité (4 entrées vers 2 sorties)

Schéma fonctionnel

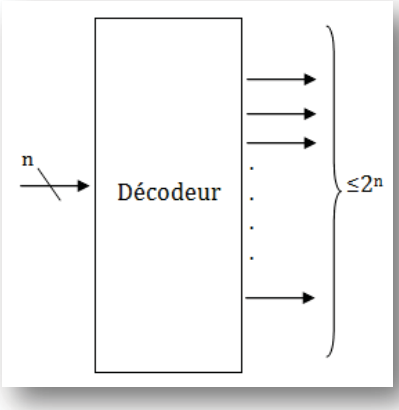


La table de vérité

A3	A2	A1	A0	S1	S0
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

Les décodeur

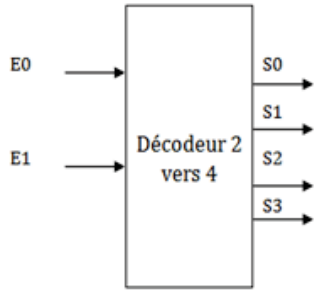
- Circuit qui "décode" le poids logique d'une entrée.
- n entrées de données
- N sorties avec une seule sortie est active à la fois



Les décodeur

- Décodeur 2 vers 4

Schéma fonctionnel	La table de vérité	Les équations



Entrées		Sorties			
E1	E0	S3	S2	S1	S0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$S0 = \overline{E_0}\,\overline{E_1}$

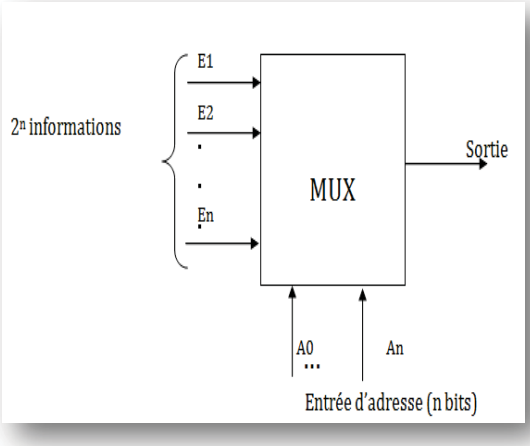
$S1 = E_0\,\overline{E_1}$

$S2 = E_1\,\overline{E_0}$

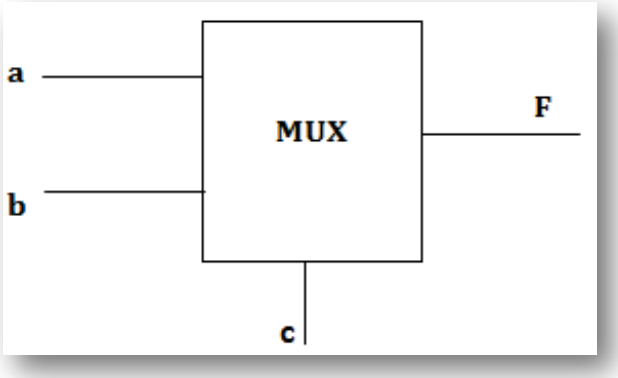
$S3 = E_0 E_1$

Les multiplexeurs

- Aiguiller un signal d'entrée parmi 2^n vers une sortie à l'aide de n bits d'adresse



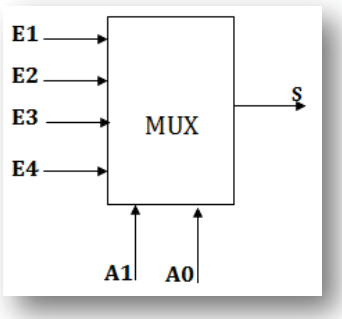
Multiplexeur à 2 entrées



c	F
0	a
1	b

$$F = a\bar{c} + bc$$

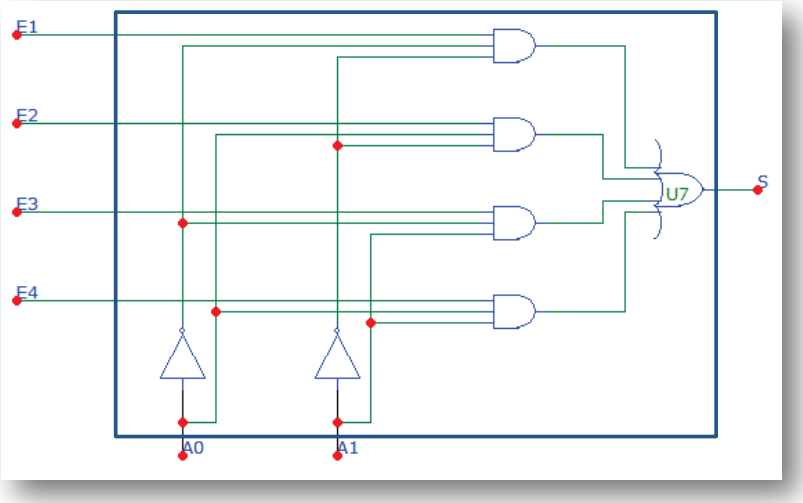
Multiplexeur à 4 entrées



A1	A0	S
0	0	E1
0	1	E2
1	0	E3
1	1	E4

$$S = E_1\,\overline{A_0}\,\overline{A_1} + E_2\,A_0\,\overline{A_1} + E_3\,\overline{A_0}A_1 + E_4\,A_0\,A_1$$

Multiplexeur à 4 entrées



Exercice: Concevoir un multiplexeur à 8 entrées (table de vérité, équations, logigramme).

Multiplexeur

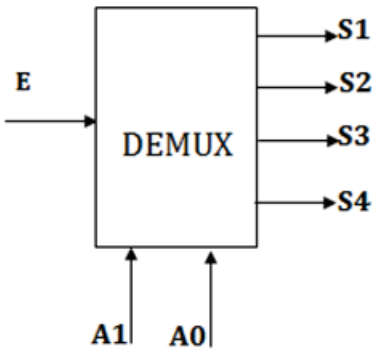
Exemple

Réaliser la table de vérité suivante avec un MUX 8 à 1.

C	B	A	S
0	0	0	1
0	0	1	0
0	1	0	X
0	1	1	1
1	0	0	X
1	0	1	1
1	1	0	1
1	1	1	X

Exemple d'un démultiplexeur à 4 voies

- Aiguiller un signal d'entrée vers une des 2^n sorties en fonction de l'état des bits d'adresse.



A1	A0	S1	S2	S3	S4
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

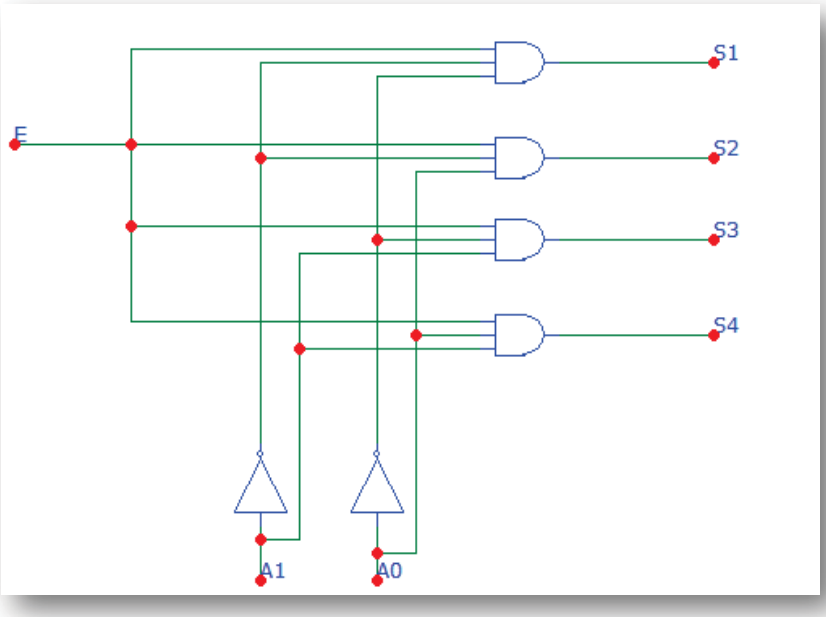
Exemple d'un démultiplexeur à 4 voies

$$s1 = E\,\overline{A_0}\,\overline{A_1}$$

$$s2 = E\,A_0\,\overline{A_1}$$

$$s3 = E\,\overline{A_0}A_1$$

$$s4 = E_4\,A_0\,A_1$$



La fonction comparaison

Un comparateur parallèle de mots permet la comparaison d'un mot binaire avec un autre mot de même longueur.

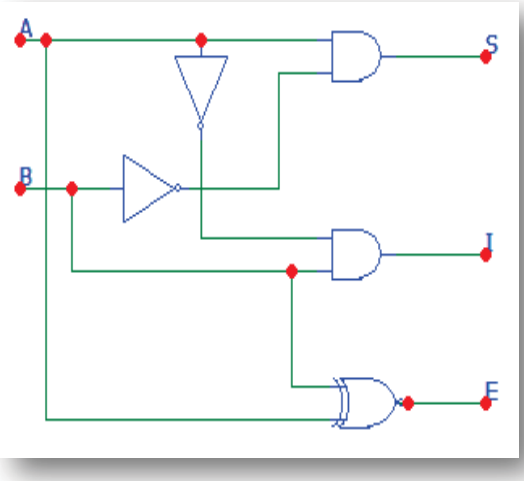
B	A	S	I	E
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

La fonction comparaison

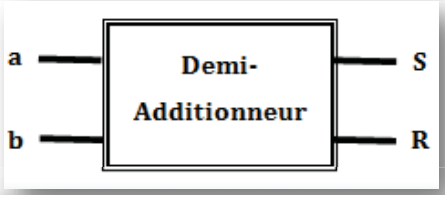
$$S = A\overline{B}$$

$$I = \overline{A}B$$

$$E = \overline{A \oplus B}$$



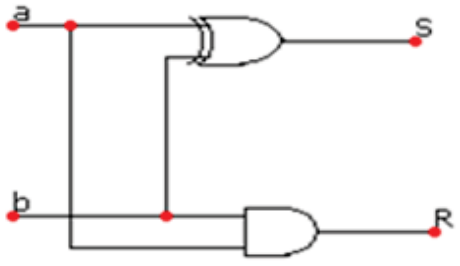
Demi additionneur



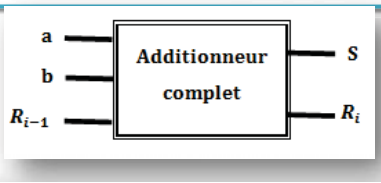
La table de vérité	Les équations	Le logigramme

b	a	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{cases} S = a \oplus b \\ R = a.b \end{cases}$$



Additionneur complet

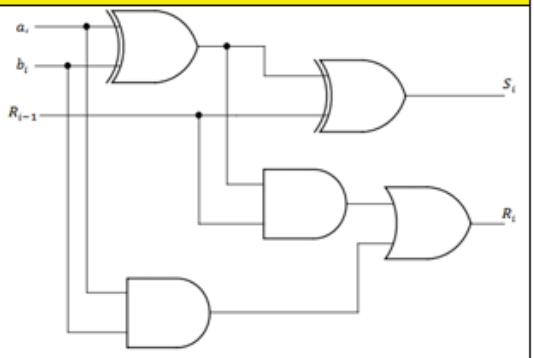


La table de vérité	Les équations	Le logigramme

R_{i-1}	b_i	a_i	S_i	R_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = R_{i-1} \oplus (a_i \oplus b_i)$$

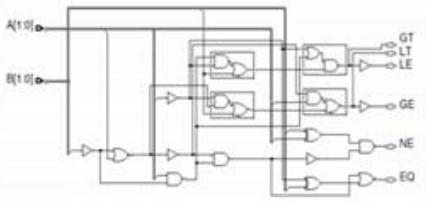
$$R_i = a_i b_i + R_{i-1}(a_i \oplus b_i)$$



Les circuits standards :Combinatoires

Comparateur

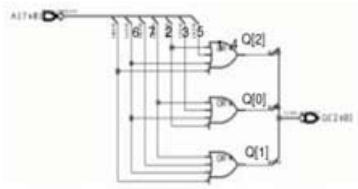
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity comparator is
    generic (NBits: POSITIVE := 2);
    port (A, B: in std_logic_vector(NBits-1 downto 0);
          EQ, GT, LT, NE, GE, LE: out std_logic);
end comparator;

architecture bhv of comparator is
begin
    process (A, B)
    begin
        if unsigned(A) > unsigned(B) then
            EQ <= '0'; LT <= '0'; LE <= '0'; GT <= '1'; NE <= '1'; GE <= '1';
        elsif unsigned(A) < unsigned(B) then
            EQ <= '0'; GT <= '0'; GE <= '0'; LT <= '1'; NE <= '1'; LE <= '1';
        else
            NE <= '0'; GT <= '0'; LT <= '0'; EQ <= '1'; GE <= '1'; LE <= '1';
        end if;
    end process;
end bhv;
```



Circuit équivalent pour Nbits = 2

Les circuits standards :Combinatoires

Encodeur

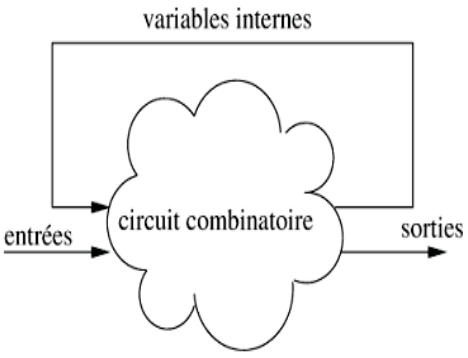


Entrées								Sorties		
A7	A6	A5	A4	A3	A2	A1	A0	Q2	Q1	Q0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

```
library IEEE;
use IEEE.std_logic_1164.all;

entity encoder83 is
    port (A: in  std_logic_vector(7 downto 0);
          Q: out std_logic_vector(2 downto 0));
end encoder83;

architecture csel of encoder83 is
begin
    with A select
        Q <= "000" when "00000001",
             "001" when "00000010",
             "010" when "00000100",
             "011" when "00001000",
             "100" when "00010000",
             "101" when "00100000",
             "110" when "01000000",
             "111" when "10000000",
             "---"  when others;
end csel;
```

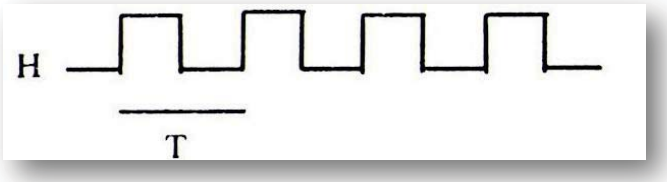

Introduction

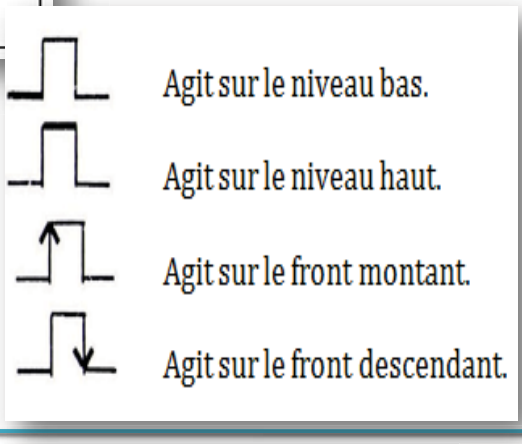
- La valeur de sortie d'un circuit séquentiel ne dépend pas que des variables logiques d'entrée, mais dépend aussi de la valeur de sortie antérieure.
- Composé d'un circuit combinatoire et l'élément mémoire appelés bascule.
- Dans le système séquentiel, le circuit de base est la bascule (flip-flop).



Classification des systèmes séquentiels

1. Circuit séquentiel synchrone
   - synchronisation par horloge
   - systèmes contrôlés

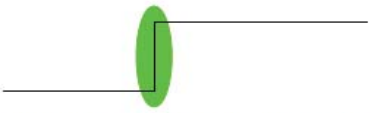




Les circuits standards :séquentielles

Génération d'horloge

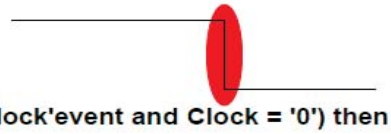
- Utilisation des attributs de signaux :
  - ✓ détection d'un front montant :



- if (Clock'event and Clock = '1') then

  - ✓ détection d'un front descendant :

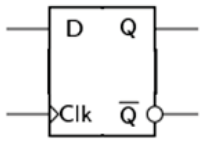
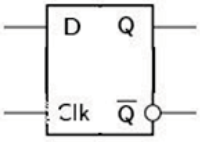
Logique synchrone



- if (Clock'event and Clock = '0') then

La bascule D

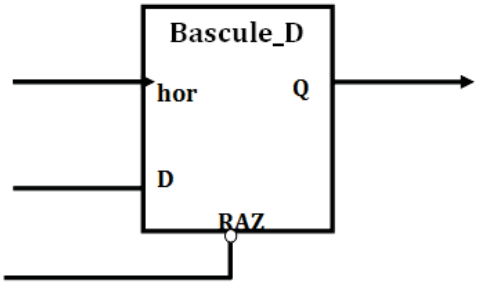
- La bascule D est une bascule suiveuse
- l'entrée D donnée (data) est recopiée sur la sortie

Bascule à commande par front.	Bascule de stockage type D (commande par niveau)
	

D	Horloge	Q+
0	⎍	0
1	⎍	1

Bascule D

Elle transfère l'état de l'entrée D sur la sortie Q après le front montant de l'horloge (clock).



Bascule

a) Bascule D avec RAZ synchrone

```
entity bascule_D is
   Port ( D : in  STD_LOGIC;
          RAZ : in  STD_LOGIC;
          CLOCK : in  STD_LOGIC;
          Q : out  STD_LOGIC);
end bascule_D;

architecture arch of bascule_D is

begin
process(clock)
begin
                    if clock'event and clock='1' then
                          if RAZ='0' then
                          Q<='0';
                          else
                          Q<=D;
                          end if;
                    end if;
end process;

end arch;
```

Remarque : RAZ synchrone, signifie que la RAZ ne sera pris en compte que lors d'un front d'horloge. Dans ce cas seule l'horloge est utile dans la liste de sensibilité.

Bascule

b) Bascule D avec RAZ asynchrone

```
entity bascule_D_asyn is
   Port ( D : in  STD_LOGIC;
          RAZ : in  STD_LOGIC;
          CLOCK : in  STD_LOGIC;
          Q : out  STD_LOGIC);
end bascule_D_asyn;

architecture arch2 of bascule_D_asyn is

begin
process(clock,RAZ)
begin
                    if RAZ='0' then Q<='0';
                    elsif clock'event and clock='1' then
                    Q<=D;
                    end if;
end process;

end arch2;
```

Remarque : dans ce cas, la présence de la RAZ dans la liste de sensibilité est obligatoire.

Compteur

Exemple : concevoir un compteur modulo 8 avec une RAZ asynchrone active niveau haut, tel que la sortie doit être de type entier.

Solution1 :l'utilisation d'une variable

Solution1 :l'utilisation d'une variable

```
entity counter_var is
   Port ( clk : in  STD_LOGIC;
          raz : in  STD_LOGIC;
          count : out  integer range 0 to 7);
end counter_var;
architecture arch1 of counter_var is

begin
process (clk,raz)
variable temp:integer range 0 to 7;
begin
                    if raz='1' then
                          temp:=0;
                    elsif clk'event and clk='1' then
                          temp:=temp+1;
                    end if;
                          count<=temp;
end process;
end arch1;
```


Compteur

Solution2 :l'utilisation d'un signal

```
entity counter_sig is
   Port ( clk : in  STD_LOGIC;
          raz : in  STD_LOGIC;
          count : out  integer range 0 to 7);
end counter_sig;
architecture arch3 of counter_sig is
signal inter:integer range 0 to 7;
begin
process(clk,raz)
begin
                    if raz='1' then
                          inter<=0;
                    elsif clk'event and clk='1' then
                          inter<=inter+1;
                    end if;
                    count<=inter;
end process;
end arch3;
```

Compteur

La modélisation d'un compteur modulo $N \neq 2^n$

Décrire un compteur modulo 6 (peut compter de 0 à 5) avec les mêmes spécifications précédentes.

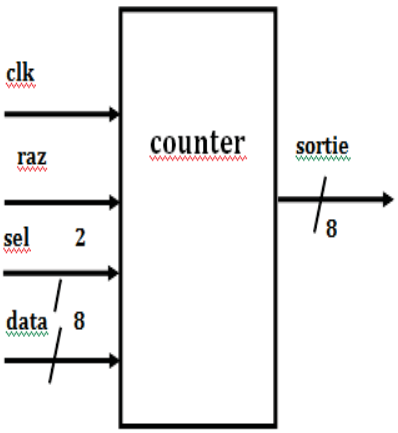
```
entity counter is
   Port ( clk : in  STD_LOGIC;
          raz : in  STD_LOGIC;
          count : out  integer range 0 to 7);
end counter;

architecture arch4 of counter is
signal inter:integer range 0 to 7;
begin
process(clk,raz)
begin
                    if raz='1' then
                            inter<=0;
                    elsif clk'event and clk='1' then
                            if inter=5 then
                            inter<=0;
                            else
                            inter<=inter+1;
                            end if;
                    end if;
                    count<=inter;
end process;
end arch4;
```


Compteur

Exercice

Ecrire le code VHDL (entity/architecture) d'un compteur 8 bits avec une raz asynchrone, peut effectuer les fonctions suivantes :

1. Initialisation à une valeur fournie en entrée (chargement parallèle).
2. Comptage.
3. Décomptage.
4. Maintie



Solution

```
entity exo is
   Port ( clk : in  STD_LOGIC;
          raz : in  STD_LOGIC;
          sel : in  STD_LOGIC_VECTOR (1 downto 0);
          data : in  STD_LOGIC_VECTOR (7 downto 0);
          sortie : out  STD_LOGIC_VECTOR (7 downto 0));
end exo;
architecture arch5 of exo is
signal cpt:STD_LOGIC_VECTOR (7 downto 0);
begin
process(clk,raz)
begin
                    if raz='0' then cpt<=(others=>'0');
                                elsif sel="00" then cpt<=data;
                                elsif sel="01" then cpt<=cpt+1;
                                elsif sel="10" then cpt<=cpt-1;
                                else
                                cpt<=cpt;
                    end if;
sortie<=cpt;
end process;
end arch5;
```