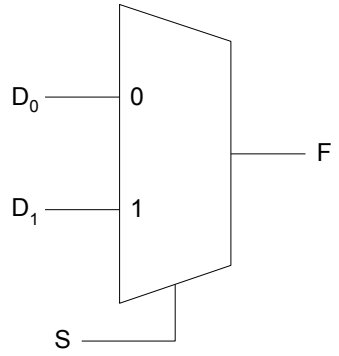
Modélisation VHDL d'un chemin des données

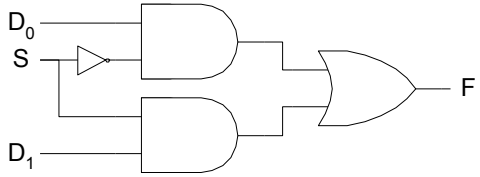
Modélisation VHDL d’un chemin des données: sujets de ce thème

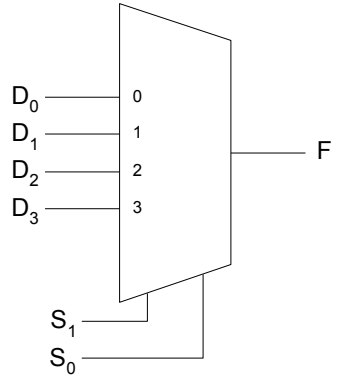
- Code VHDL des composantes d'un chemin des données:
  - multiplexeurs
  - registres à chargement parallèle
  - fonctions arithmétiques
  - fonctions logiques
  - comparateurs
  - registres à décalage
  - compteurs

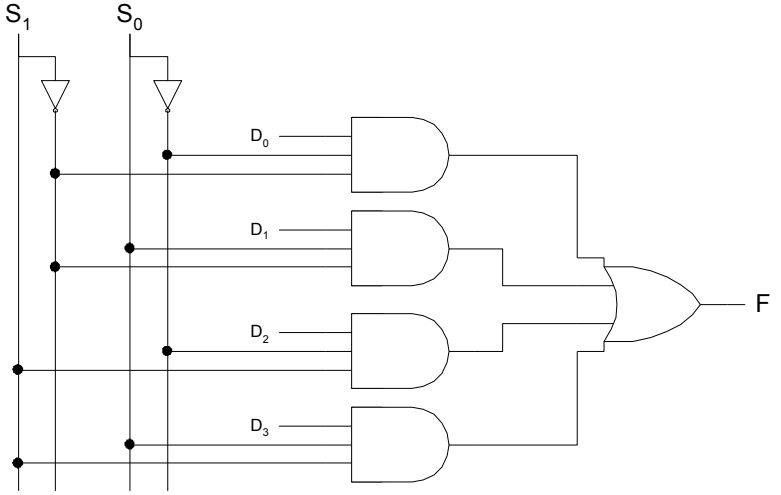
Composante de chemins des données #1: le multiplexeur

- Un multiplexeur permet de choisir un seul signal à partir d'un ensemble de signaux, selon la valeur d'un signal de contrôle.
- Un multiplexeur a :
  - un groupe de signaux d'entrée *D*;
  - un groupe de signaux de contrôle *S* (pour *sélection*); et,
  - un signal de sortie *F*.
- Le signal de sortie est égal au signal d'entrée choisi par les signaux de contrôle.









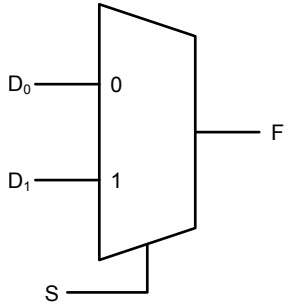
$$F = \sum_{k=0}^{2^n-1} m_k D_k$$

où m_k est un minterme formé par la *k*–ième combinaison de signaux de contrôle *S*.

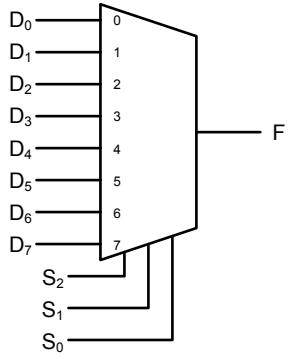
Composante de chemins des données #1: le multiplexeur

- Description comportementale très compacte.
- Normalement on ne décrirait pas un multiplexeur dans une entité séparée, on utiliserait plutôt ce patron de code dans un module, un énoncé `if-else`, ou un énoncé `case`.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux21 is
    port(D0, D1, S : in STD_LOGIC; F : out STD_LOGIC);
end mux21;

architecture flotDeDonnees of mux21 is
begin

   with S select
   F <= D0 when '0', D1 when others;

end flotDeDonnees;
```

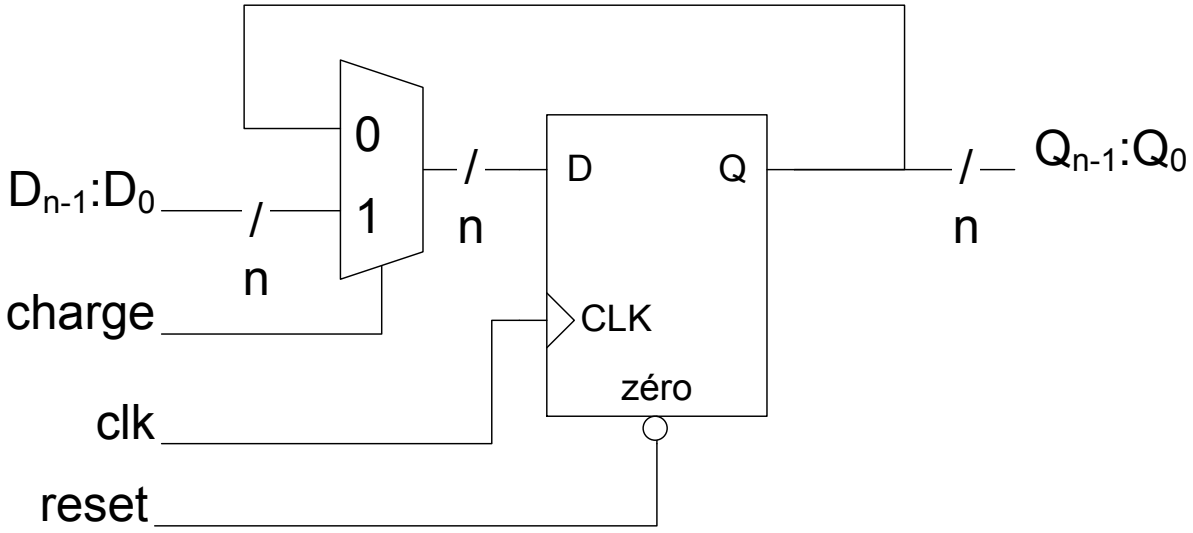


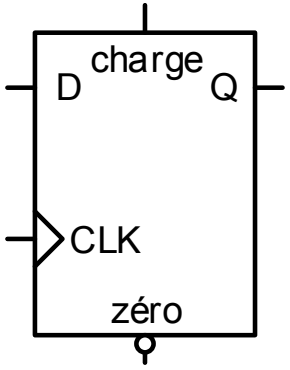

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mux is
   generic (
      n : positive := 3 -- nombre de signaux de contrôle
   );
   port (
      D : in std_logic_vector(2 ** n - 1 downto 0);
      S: in unsigned(n - 1 downto 0);
      F : out std_logic
   );
end mux;

architecture comportementale of mux is

begin

   process (D, S)
   begin
      F <= D(to_integer(S));
   end process;

end comportementale;
```



Composante de chemins des données #2: le registre à chargement parallèle

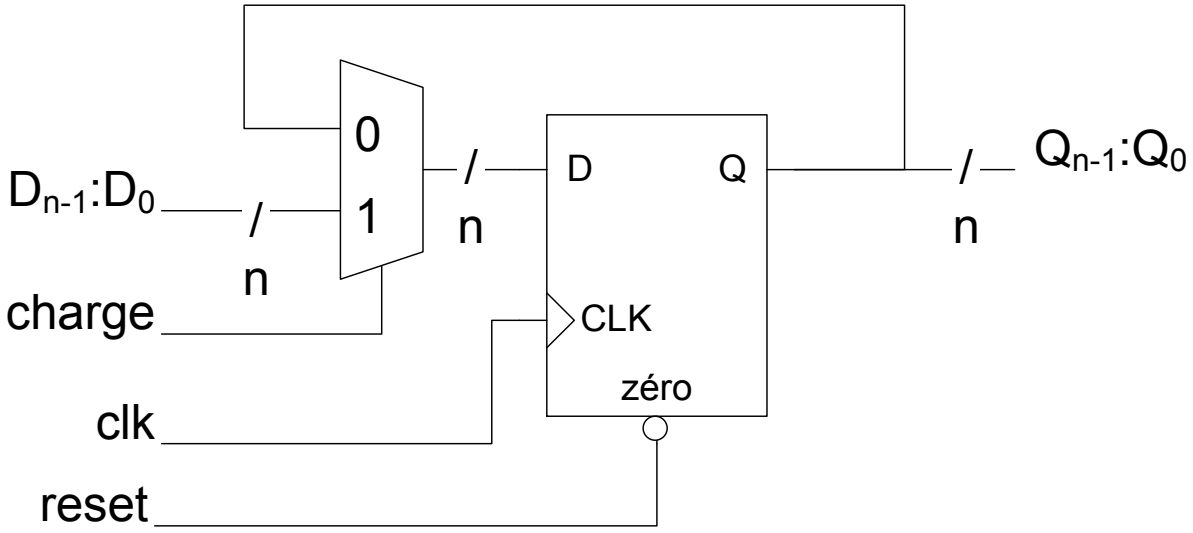
- Un registre est l'élément à mémoire de base pour des données. Il est utilisé pour entreposer une information encodée sur un groupe de bits.
- Un registre est composé d'un groupe de bascules contrôlées par une horloge commune et dont les entrées et sorties partagent un identificateur commun. Chaque bascule du registre est différenciée des autres par un indice unique.
- Un registre à chargement parallèle comporte un signal de chargement qui permet de choisir quand le registre doit être chargé.





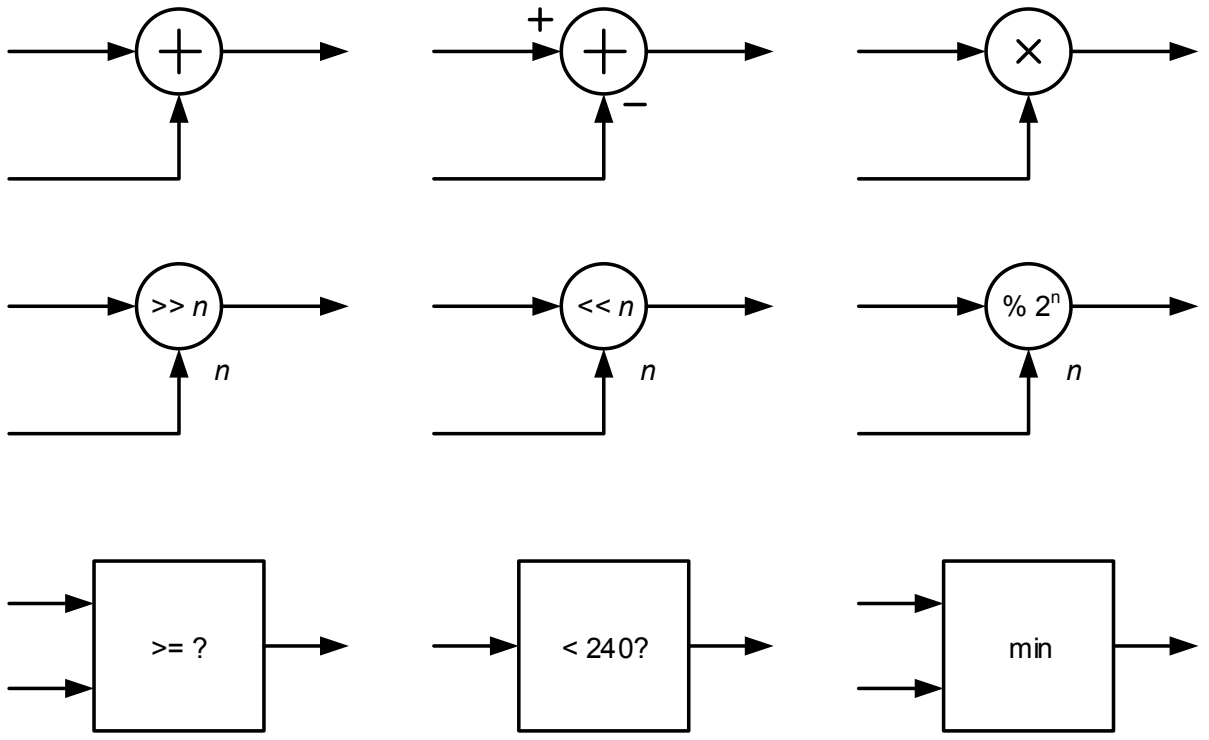
Composante de chemins des données #2: le registre à chargement parallèle

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity registre is
   generic (
      W : integer := 4
   );
   port(
      reset, CLK, charge : in STD_LOGIC;
      D : in STD_LOGIC_VECTOR(W - 1 downto 0);
      Q : out STD_LOGIC_VECTOR(W - 1 downto 0)
   );
end registre;

architecture arch of registre is
begin
   process (CLK, reset)
   begin
      if reset = '0' then
         Q <= (others => '0');
       elsif rising_edge(CLK) then
         if charge = '1' then
            Q <= D;
         end if;
      end if;
   end process;
end arch;
```



Composante de chemins des données #3: les unités fonctionnelles

- Les unités fonctionnelles peuvent inclure, entre autres:
  - les opérations arithmétiques, dont l'addition, la soustraction et la multiplication;
  - le décalage vers la droite par *n* bits (= ÷ 2^n) ou la gauche (= × 2^n), et le modulo 2^n;
  - les opérations logiques;
  - la comparaison de deux valeurs ou d'une valeur avec une constante; et,
  - le minimum et le maximum.



Types VHDL à utiliser pour la synthèse

- **`real`: ça dépend**
  - pour des valeurs constantes: ok
  - pour des registres et valeurs intermédiaires: non! trop précis et *pas synthétisable*.
- **`integer`, `natural`, `positive`: acceptables**
  - Bien supportés par les synthétiseurs pour les opérations arithmétiques.
  - Bonne abstraction par rapport à un vecteur de bits.
  - Important de spécifier la gamme de valeurs possibles de façon à contraindre les ressources matérielles utilisées pour les représenter.
    Par défaut: 32 bits pour chaque signal.
- **`signed`, `unsigned`: acceptables**
  - Définis dans le package normalisé `numeric_std`, comme des tableaux de `std_logic`.
  - Bien supportés par les outils de synthèse:
    un fil/un registre par bit
  - Le package `numeric_std` redéfinit les opérateurs de VHDL pour ces deux types.
- **`std_logic_vector`: pas pour l'arithmétique**
  - Défini dans le package `std_logic_1164` comme un tableau de `std_logic`.
  - Des packages populaires incluent des définitions pour les opérations arithmétiques, mais ils ne sont pas normalisés et *leur utilisation n'est pas recommandée pour représenter des nombres*.

Composante de chemins des données #3: unités fonctionnelles - ALU

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

entity unitearithmetique is
   generic (
      W : positive := 8 -- largeur des opérandes
   );
   port(
      A, B : in signed(W - 1 downto 0); -- les opérandes
      choix : in std_logic_vector(2 downto 0); -- l'opération à faire
      F : out signed(W - 1 downto 0) -- le résultat
   );
end unitearithmetique;

architecture arch of unitearithmetique is
begin
   process(A, B, choix)
   variable t : signed(2 * W - 1 downto 0);
   begin
      t := A * B;
      case to_integer(unsigned(choix)) is
         when 0 => F <= A + B;
         when 1 => F <= A - B;
         when 2 => F <= A + B + 1;
         when 3 => F <= A + 1;
         when 4 => F <= abs(A);
         when 5 => F <= -A;
         when 6 => F <= t(2 * W - 1 downto W);
         when 7 => F <= t(W - 1 downto 0);
         when others => F <= (others => 'X');
      end case;
   end process;
end arch;
```

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

entity unitelogique is
   generic (
      W : positive := 8 -- largeur des opérandes
   );
   port(
      A, B : in std_logic_vector(W - 1 downto 0); -- les opérandes
      choix : in std_logic_vector(2 downto 0); -- l'opération à faire
      F : out std_logic_vector(W - 1 downto 0) -- le résultat
   );
end unitelogique;

architecture arch of unitelogique is
begin
   process(A, B, choix)
   begin
      case to_integer(unsigned(choix)) is
         when 0 => F <= A and B;
         when 1 => F <= A or B;
         when 2 => F <= A nand B;
         when 3 => F <= A nor B;
         when 4 => F <= A xor B;
         when 5 => F <= A xnor B;
         when 6 => F <= not(A);
         when 7 => F <= not(B);
         when others => F <= (others => 'X');
      end case;
   end process;
end arch;
```

Composante de chemins des données #3: unités fonctionnelles - comparateurs

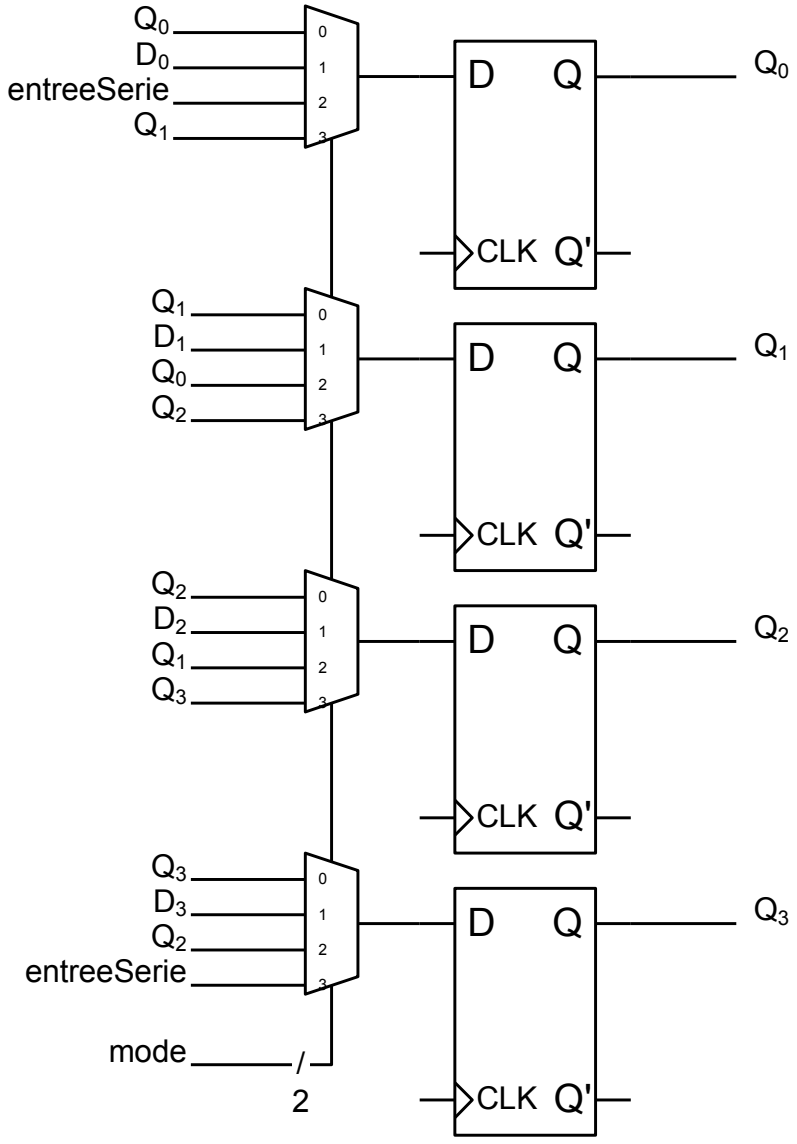
- Un comparateur permet de comparer les grandeurs relatives de deux valeurs et d'identifier leur égalité éventuelle.
- Ce type de circuit est essentiel dans un microprocesseur pour pouvoir effectuer des branchements conditionnels.
- Les opérateurs de VHDL pour la comparaison sont =, /=, <, <=, >, et >=. Dans chaque cas le résultat de la comparaison est de type `boolean`.
- Comme pour les opérations arithmétiques, le type des opérandes est critique et peut déterminer la valeur de la comparaison.

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

entity comparateur is
  generic (
    W : positive := 8 -- largeur des opérandes
  );
  port(
    A, B : in signed(W - 1 downto 0);
    eq, neq, gt, lt, ge, le : out std_logic
  );
end comparateur;

architecture arch of comparateur is
begin
eq <= '1' when A = B else '0';
neq <= '1' when A /= B else '0';
gt <= '1' when A > B else '0';
lt <= '1' when A < B else '0';
ge <= '1' when A >= B else '0';
le <= '1' when A <= B else '0';
end arch;
```

Composante de chemins des données #4: le registre à décalage

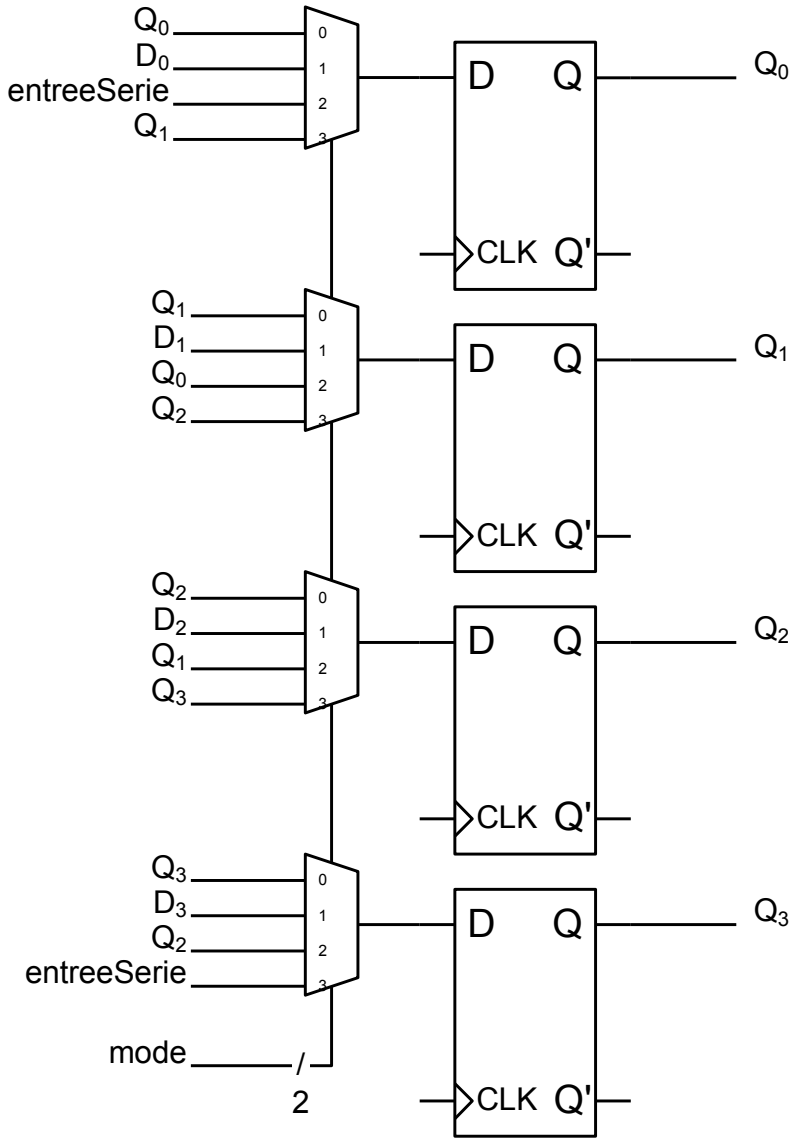
- Un registre à décalage peut décaler ses bits vers la gauche ou la droite.
- Un registre à décalage de 1 position peut être construit par une cascade de bascules D dont la sortie est reliée à l'entrée de la bascule suivante.
- Un registre à décalage peut être utilisé pour faire une conversion entre les formats série et parallèle et est donc une composante fondamentale de plusieurs circuits de communication.

Mode	Effet
00	Mémoire
01	Chargement
10	Décale gauche
11	Décale droite



Composante de chemins des données #4: le registre à décalage

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity registreadecalage is
    generic (
        W : integer := 8 -- nombre de bits du registre
    );
    port(
        CLK, reset : in STD_LOGIC;
        mode : in STD_LOGIC_VECTOR(1 downto 0); -- mode
        entreeSerie : in STD_LOGIC; -- entree serielle
        D : in STD_LOGIC_VECTOR(W - 1 downto 0);
        Q : out STD_LOGIC_VECTOR(W - 1 downto 0)
    );
end registreadecalage;
architecture arch of registreadecalage is
begin
    process (CLK, reset)
    variable Qinterne : STD_LOGIC_VECTOR(W - 1 downto 0);
    begin
        if reset='0' then
            Qinterne := (others => '0');
            Q <= (others => '0');
         elsif CLK='1' and CLK'event then
             case mode is
               when "00" => -- garde
                   Qinterne := Qinterne;
               when "01" => -- charge
                   Qinterne := D;
               when "10" => -- decale gauche
                   Qinterne := Qinterne(W - 2 downto 0) & entreeSerie;
               when "11" => -- decale droite
                   Qinterne := entreeSerie & Qinterne(W - 1 downto 1);
               when others =>
                   Qinterne := Qinterne;
            end case;
            Q <= Qinterne;
        end if;
    end process;
end arch;
```



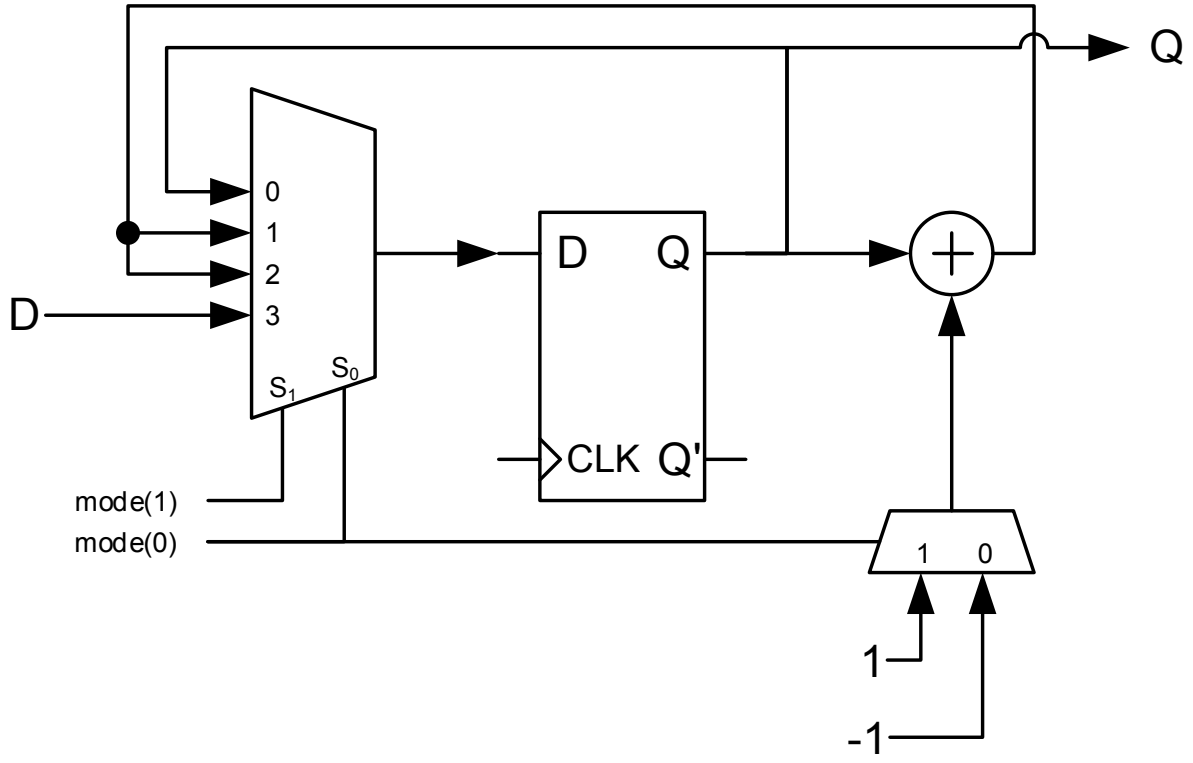
Composante de chemins des données #5: compteurs

- Un compteur compte le nombre d'occurrences d'un événement.
- Un compteur est habituellement composé d'un registre couplé à un circuit combinatoire qui calcule la prochaine valeur du compte en fonction de sa valeur présente.
- En plus de la séquence suivie, les compteurs peuvent être caractérisés par :
  - La valeur de réinitialisation (souvent 0).
  - La direction du compte (le haut, le bas ou les deux).
  - L'incrément du compte.
  - Le chargement parallèle d'une valeur de compte.
  - Une entrée ou une sortie sérielle.

- Quelques types de compteurs:
  - Compteur binaire. Progression monotone : 000, 001, 010, 011, …, 101, 110, 111, 000, 001, etc. Un compteur binaire à n bits a 2^n états différents.
  - Compteur modulo-n. Ce compteur est réinitialisé à zéro dès qu'une valeur spécifiée est atteinte. Cas particulier: compteur BCD: 0000, 0001, … 1000, 1001, 0000, …;
  - Compteur à anneau. 0001, 0010, 0100, 1000, 0001, 0010, etc. Peut entrer dans une séquence d'états interdits si une erreur se produit.
  - Compteur Johnson. 0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000, 0000, 0001, …
  - Compteur à séquence arbitraire. L'utilisateur détermine la séquence, comme par exemple 0, 3, 1, 4, 2, 6, 0, 3, 1, etc.

Composante de chemins des données #5: compteurs

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.all;

entity compteurSynchrone is
   generic (
      W : integer := 4 -- nombre de bits du compteur
   );
   port(
      reset : in STD_LOGIC;
      CLK : in STD_LOGIC;
      mode : in unsigned(1 downto 0);
      D : in unsigned(W - 1 downto 0);
      Q : out unsigned(W - 1 downto 0)
   );
end compteurSynchrone;

architecture comportementale of compteurSynchrone is
begin
   process (CLK, reset)
   variable Qinterne : unsigned(W - 1 downto 0);
   begin
      if reset='0' then
         Qinterne := (others => '0');
      elsif CLK='1' and CLK'event then
         case mode is
            when "01" => Qinterne := Qinterne + 1;
            when "10" => Qinterne := Qinterne - 1;
            when "11" => Qinterne := D;
            when others => Qinterne := Qinterne;
         end case;
      end if;
      Q <= Qinterne;
   end process;
end comportementale;
```



Vous devriez maintenant être capable de …

- Donner le code VHDL correspondant à des micro-opérations. (B3)
- Modéliser en VHDL les composantes suivantes d'un chemin des données: multiplexeurs, registres à chargement parallèle, registres à décalage, fonctions arithmétiques, fonctions logiques et comparateurs. (B3)
- Donner le code VHDL d'un compteur rencontrant des spécifications particulières en termes de valeur de chargement, valeur initiale, direction de compte, intervalle de compte ou compte arbitraire. (B3)

Code	Niveau (http://fr.wikipedia.org/wiki/Taxonomie_de_Bloom)
B1	Connaissance - mémoriser de l'information.
B2	Compréhension – interpréter l'information.
B3	Application – confronter les connaissances à des cas pratiques simples.
B4	Analyse – décomposer un problème, cas pratiques plus complexes.
B5	Synthèse – expression personnelle, cas pratiques plus complexes.