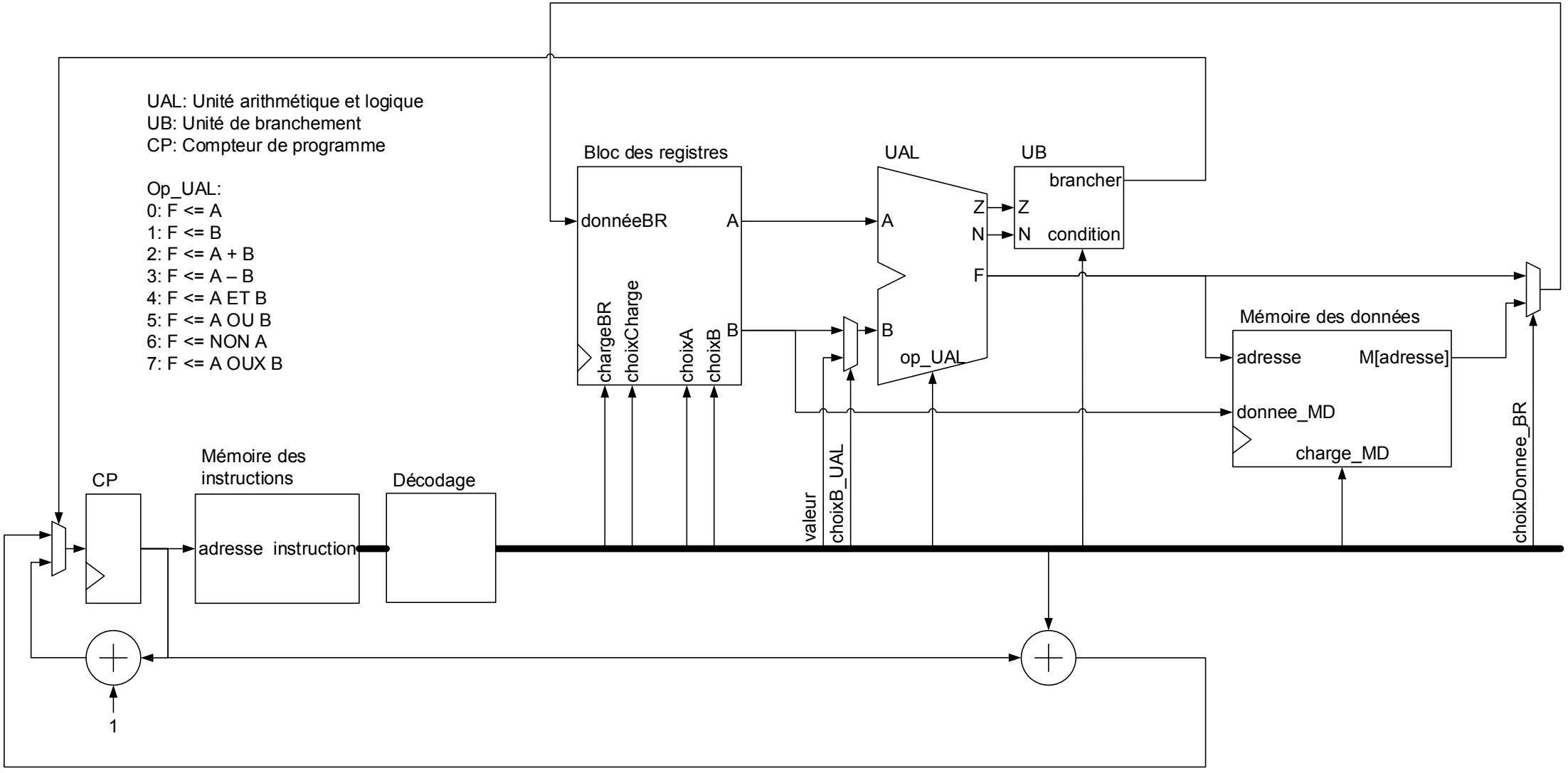
Modélisation VHDL du chemin des données d'un processeur à usage général

Modélisation VHDL du chemin des données d'un processeur à usage général
Sujets de ce thème

- Rappel: architecture du processeur PolyRISC et son chemin des données
- Modélisation du bloc des registres
- Modélisation de l'unité arithmétique et logique
- Modélisation de la mémoire des données
- Implémenter la mémoire des données dans un FPGA

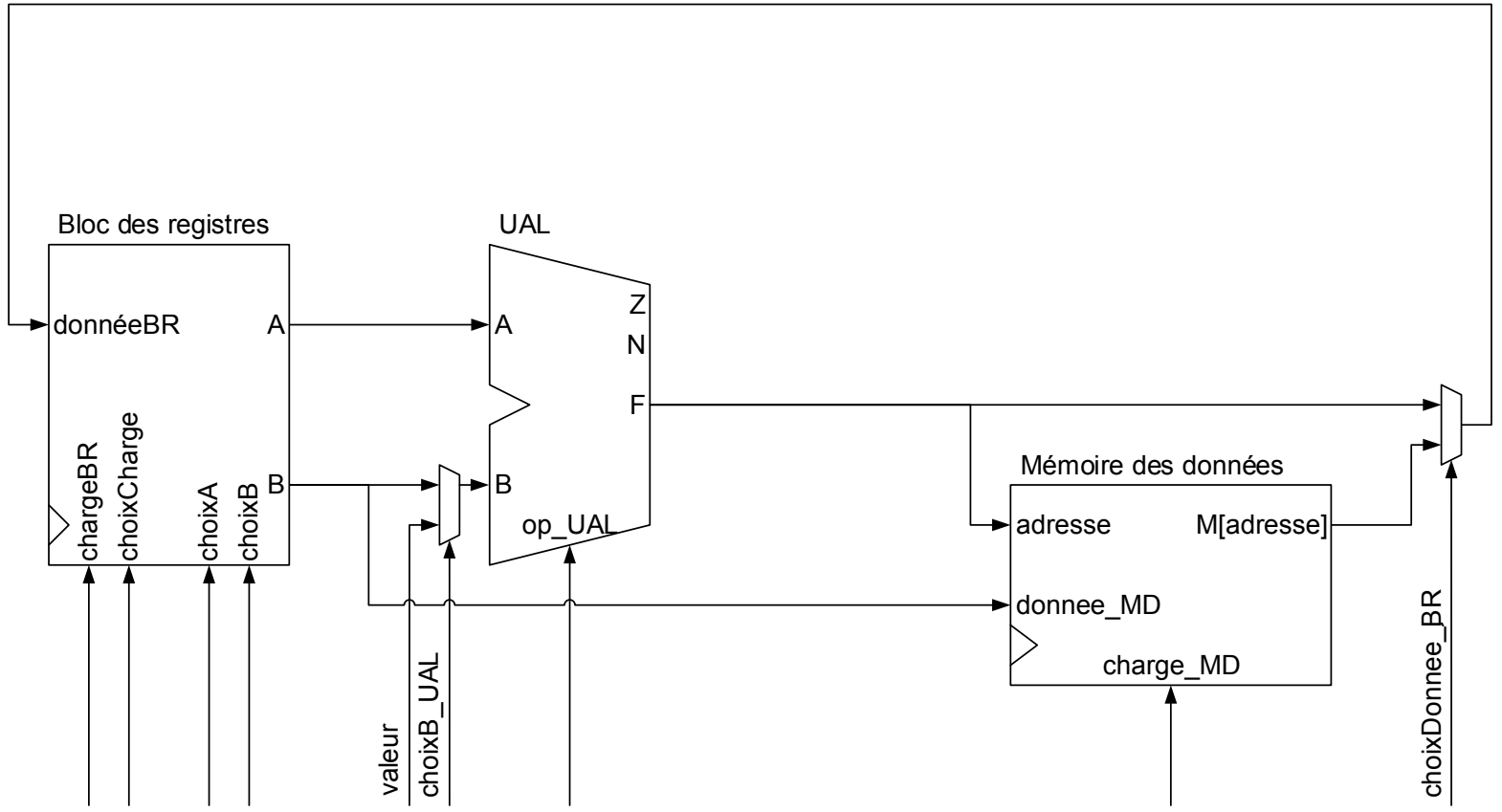
Le processeur à usage général PolyRISC



Chemin des données du processeur PolyRISC

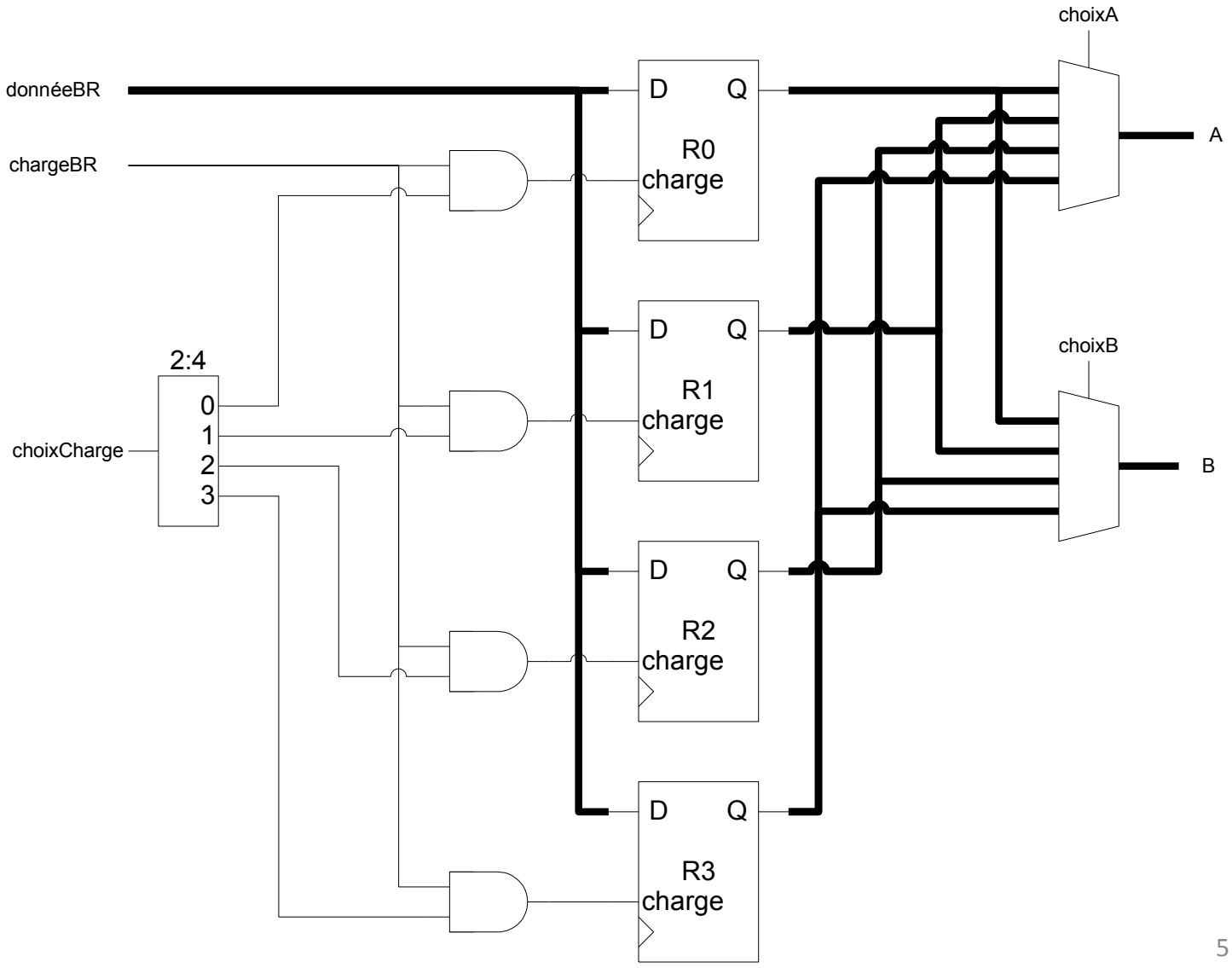
- Nous allons voir comment modéliser les trois composantes du chemin des données:

1. Le bloc des registres
2. L'unité arithmétique et logique
3. La mémoire des données



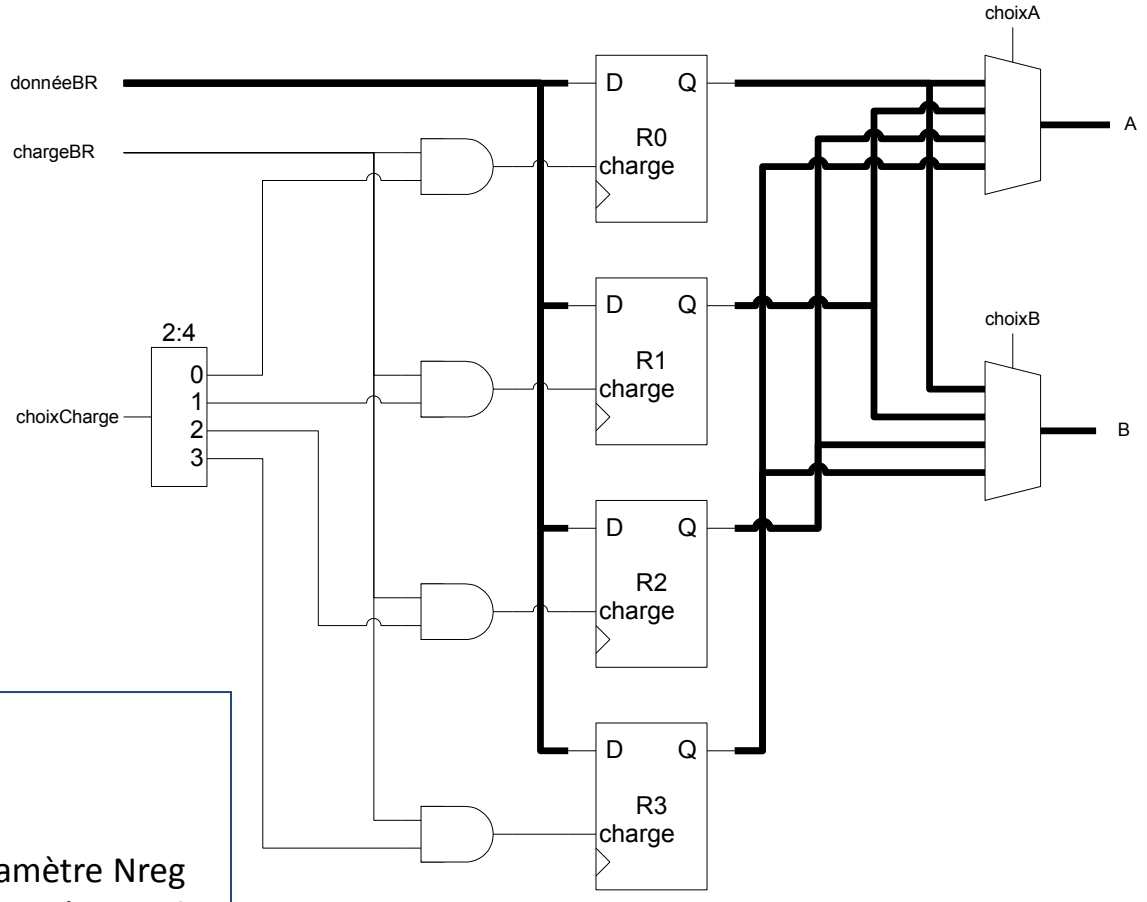
1. Bloc de registres

- Un bloc de registre contient:
  - Des registres
  - Un mécanisme pour les charger à partir d'un port d'entrée
  - Un mécanisme pour aiguiller leur valeur à des ports de sortie
- Un bloc de registre peut avoir plusieurs ports d'entrée et de sortie.
- Un registre peut être simultanément la cible d'une écriture et la source pour l'un des ports de sortie.



1. Bloc de registres

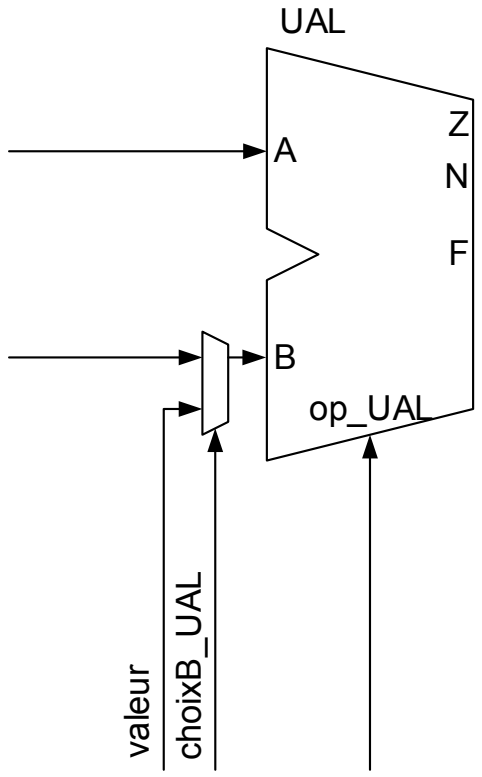
```
-- dans la partie déclarative de l'architecture

-- signaux du bloc des registres
type lesRegistres_type is array(0 to Nreg - 1) of signed(Wd - 1 downto 0);
signal lesRegistres : lesRegistres_type;

signal A, B : signed(Wd - 1 downto 0);
signal choixA, choixB, choixCharge : integer range 0 to Nreg - 1;
signal chargeBR : std_logic;

   -- dans le corps de l'architecture
   process (CLK, reset)
   begin
      if rising_edge(CLK) then
         if reset = '1' then
            lesRegistres <= (others => (others => '0'));
         else
            if chargeBR = '1' then
               lesRegistres(choixCharge) <= donneeBR;
            end if;
         end if;
      end if;
   end process;

   A <= lesRegistres(choixA);
   B <= lesRegistres(choixB);
```



Dans cette description du bloc des registres:
- le nombre de ports d'entrée est fixé à 1
- le nombre de ports de sortie est fixé à 2
- le nombre de registres est déterminé par le paramètre Nreg
- la largeur des registres est déterminée par le paramètre Wd

2. Unité arithmétique et logique

- Une unité arithmétique et logique effectue des calculs sur les données qui lui sont fournies par le bloc des registres.
- Les opérations normalement réalisées par une UAL incluent:
  - les opérations arithmétiques, dont l'addition, la soustraction et la multiplication; et
  - les opérations logiques.
- Le décalage est souvent réalisé par un bloc séparé, mais peut aussi être fait par l'UAL.
- L'UAL peut aussi donner une indication sur la nature du résultat calculé, p. ex. si il est zéro (Z) ou négatif (N).

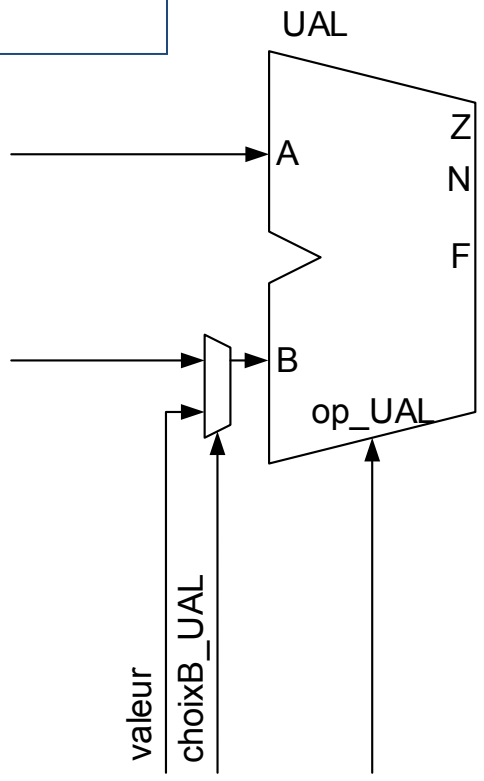


2. Unité arithmétique et logique

```
-- dans la partie déclarative de l'architecture
signal op_UAL : natural range 0 to 7;
signal valeur : signed(Wd - 1 downto 0);
signal choixB_UAL : natural range 0 to 1;
signal F : signed(Wd - 1 downto 0);
signal Z, N : std_logic;

constant passeA : natural := 0; constant passeB : natural := 1;
constant AplusB : natural := 2; constant AmoinsB : natural := 3;
constant AetB : natural := 4; constant AouB : natural := 5;
constant nonA : natural := 6; constant AouxB : natural := 7;


-- dans le corps de l'architecture
process(A, B, valeur, choixB_UAL, op_UAL)
variable B_UAL, F_UAL : signed(Wd - 1 downto 0);
begin

   -- opérations de l'UAL
   case op_UAL is
      when passeA => F_UAL := A;
      when passeB => F_UAL := B_UAL;
      when AplusB => F_UAL := A + B_UAL;
      when AmoinsB => F_UAL := A - B_UAL;
      when AetB => F_UAL := A and B_UAL;
      when AouB => F_UAL := A or B_UAL;
      when nonA => F_UAL := not(A);
      when AouxB => F_UAL := A xor B_UAL;
   end case;

   -- signal de sortie
   F <= F_UAL;
```

Dans cette description de l'UAL:
- L'encodage des opérations est fixe.
- La largeur de l'UAL est déterminée par le paramètre Wd.
- Les bits Z et N indiquent respectivement un résultat nul ou négatif et sont utilisés pour les branchements.

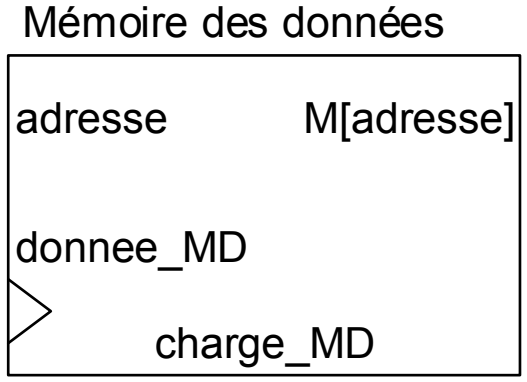
```
-- multiplexeur pour l'entrée B
if choixB_UAL = 0 then
   B_UAL := B;
else
   B_UAL := valeur;
end if;

-- drapeaux pour l'unité de branchement
if F_UAL = 0 then
   Z <= '1';
else
   Z <= '0';
end if;

N <= F_UAL(F_UAL'left);

end process;
```



3. Mémoire des données

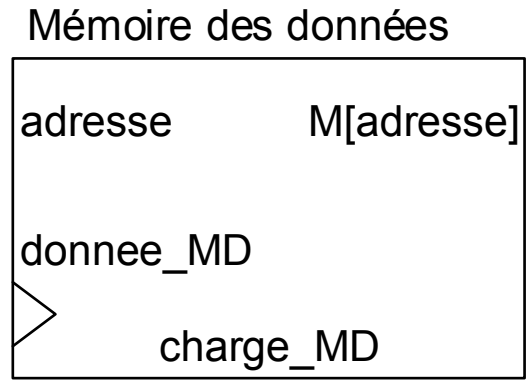
- Un processeur avec un bloc des registres suffisamment grand n'aurait pas besoin d'une mémoire des données associée. C'est souvent le cas pour les processeurs spécialisés. En pratique cependant, un processeur à usage général a toujours besoin d'une telle mémoire.
- Il est utile que les cellules de la mémoire aient la même taille que celles du bloc des registres, mais ce n'est pas strictement nécessaire.
- Pour la plupart des processeurs, la mémoire des données est placée à l'extérieur de celui-ci.

- La mémoire des données a en général:
  - un port de sortie (pour lire une donnée);
  - un port d'entrée (pour écrire une donnée);
  - un port d'adresse; et,
  - un port de contrôle de l'opération (lecture ou écriture).
- Une mémoire peut avoir plusieurs ports d'entrée et de sortie.



3. Mémoire des données

```
-- dans la partie déclarative de l'architecture
type memoireDonnees_type is array(0 to 2 ** Md - 1) of signed(Wd - 1 downto 0);
signal memoireDonnees : memoireDonnees_type;
signal charge_MD : std_logic;

   -- dans le corps de l'architecture
   process (CLK)
   begin
      if rising_edge(CLK) then
         if (charge_MD = '1') then
            memoireDonnees(to_integer(adresse)) <= donnee_MD;
         end if;
      end if;
   end process;

   sortie_MD <= memoireDonnees(to_integer(adresse));
```



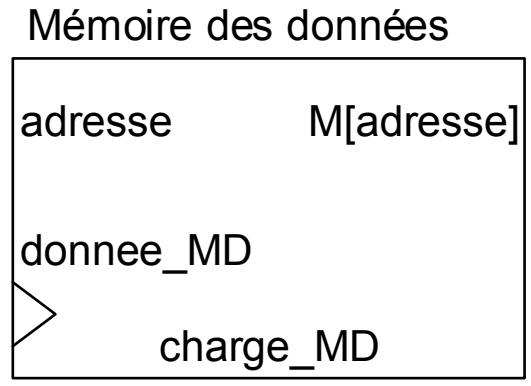
Dans cette description de la mémoire:
- le nombre de mots est déterminée par le paramètre Md
- la taille des mots est déterminée par le paramètre Wd

Habituellement, la lecture d'une donnée nécessite un cycle d'horloge pour charger l'adresse, ce n'est pas modélisé ici.

Important:
Il n'y a pas de signal de réinitialisation dans le modèle. Un tel signal empêcherait le synthétiseur d'utiliser la mémoire par blocs (BRAM).

3. Mémoire des données

- La description d'une mémoire des données en VHDL peut prendre plusieurs formes, selon une multitude de paramètres. Ceux-ci incluent, entre autres :
  - le nombre de ports d'entrée et de sortie;
  - le fait que les sorties soient synchrones ou asynchrones;
  - le nombre de cycles nécessaires à la mémoire pour déplacer des données;
  - la présence de signaux d'activation; et,
  - la spécification de valeurs initiales.



Il faut consulter la documentation du synthétiseur pour adopter le bon patron de code VHDL correspondant aux paramètres de la mémoire RAM désirée.

Comment s'implémente la mémoire dans un FPGA?

- Colonnes de blocs de mémoire intégrées à travers les CLBs (Xilinx: *Block RAM*)
  - Grande densité
- LUTs des CLBs (Xilinx: *Distributed RAM*)
  - Faible latence d'accès
- Le choix peut être contrôlé par les paramètres du synthétiseur.

Vous devriez maintenant être capable de …

- Donner le code VHDL pour le bloc des registres, l'UAL et la mémoire des données d'un processeur à usage général. (B3)
- Expliquer comment s'implémente la mémoire des données dans un FPGA. (B2)

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.