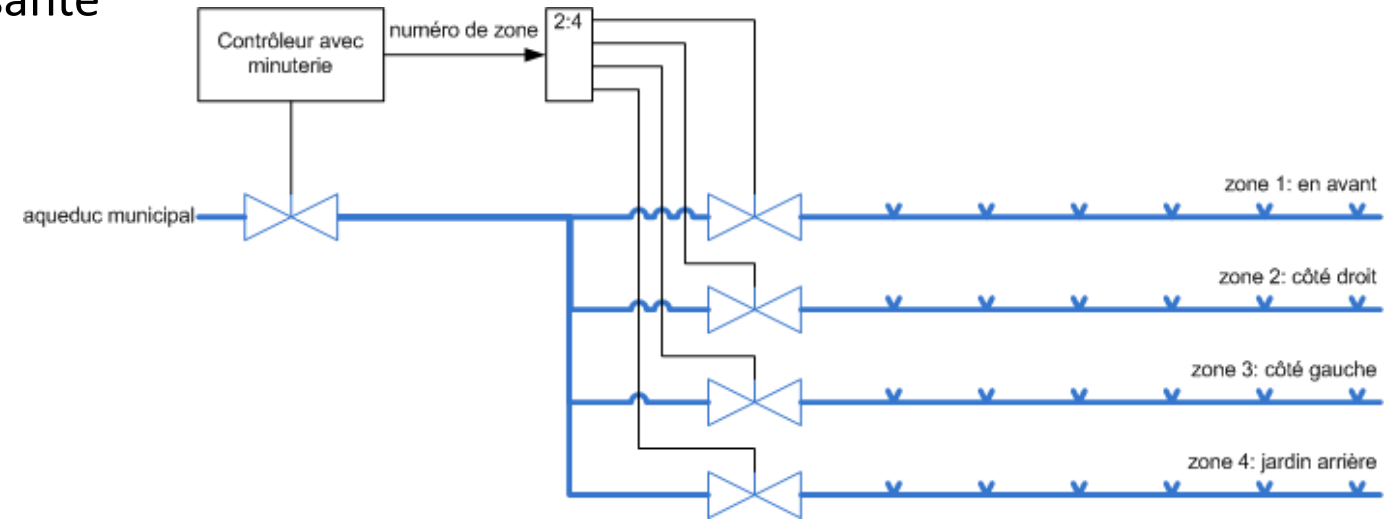
Décodeurs et encodeurs:
description, utilisation, modélisation VHDL et implémentation

Décodeurs et encodeurs : sujets de ce thème

- Décodeurs:
  - Fonctionnalité
  - Table de vérité
  - Modélisation en VHDL
  - Synthèse
- Encodeurs à priorité
  - Fonctionnalité
  - Table de vérité
  - Modélisation en VHDL
  - Synthèse

Module combinatoire utile: le décodeur – exemple d'utilisation

- Système de gicleurs automatiques domestique.
- La pression de l'aqueduc municipal est insuffisante pour activer tous les gicleurs en même temps.
- Les gicleurs sont divisés en quatre zones.
- Une seule zone doit arroser à la fois.
- Chaque zone est munie d'une valve.
- Une seule valve s'ouvre à la fois.
- Une valve principale doit aussi être ouverte.



Décodeur – table de vérité

- Un décodeur active un signal spécifique correspondant à un code numérique en particulier.
- Un décodeur a *n* signaux d'entrée et 2^n signaux de sortie.
  - Chacun des signaux de sortie correspond à un des mintermes et maxtermes composés des signaux d'entrée.
  - Une et une seule ligne de sortie est active à un moment donné.
  - Le numéro de la ligne active ligne correspond à la valeur binaire appliquée aux lignes d'entrée.
  - Selon les décodeurs, la ligne active pourra être à une valeur 0 ou une valeur 1, et toutes les autres lignes seront à l'autre valeur.

#	A_2	A_1	A_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	1	0	0
3	0	1	1	0	0	0	0	1	0	0	0
4	1	0	0	0	0	0	1	0	0	0	0
5	1	0	1	0	0	1	0	0	0	0	0
6	1	1	0	0	1	0	0	0	0	0	0
7	1	1	1	1	0	0	0	0	0	0	0

Table de vérité d'un décodeur 3:8
(A2, A1, A0) sont les entrées.
(F7, F6, …, F0) sont les sorties.

Décodeur 3:8 – modèle VHDL

- Une assignation choisie spécifie les huit cas possibles du signal d'entrée F.
- L'utilisation de la clause `others` permet de rendre le modèle plus robuste à la simulation. En effet, le type `std_logic` peut prendre des valeurs autres que '0' et '1'. Lors de la simulation, si le signal F prend une valeur comme « X1W », la sortie du décodeur sera un vecteur de 'X'.
- L'expression (`others => 'X'`) permet d'assigner la valeur 'X' à chacun des éléments du vecteur F.

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

entity decodeur38 is
   port(
      A : in std_logic_vector(2 downto 0);
      F: out std_logic_vector(7 downto 0)
   );
end decodeur38;

architecture flotDeDonnees of decodeur38 is

begin
   with A select F <=
      "00000001" when "000",
      "00000010" when "001",
      "00000100" when "010",
      "00001000" when "011",
      "00010000" when "100",
      "00100000" when "101",
      "01000000" when "110",
      "10000000" when "111",
      (others => 'X') when others;

end flotDeDonnees;
```

Parenthèse: les agrégats en VHDL

```
-- Exemple 1
variable Data_1 : BIT_VECTOR (0 to 3) := ('0','1','0','1'); -- assignation positionelle

-- Exemple 2
variable Data_2 : BIT_VECTOR (0 to 3) := (1=>'1',0=>'0',3=>'1',2=>'0'); -- assignation nommée

-- Exemple 3
signal Data_Bus : Std_Logic_Vector (15 downto 0);
. . .
Data_Bus <= (15 downto 8 => '0', 7 downto 0 => '1'); -- assignation positionelle avec gammes d'indices

-- Exemple 4
type Status_Record is record
     Code : Integer;
     Name : String (1 to 4);
end record;
variable Status_Var : Status_Record := (Code => 57, Name => "MOVE");

-- Exemple 5
signal Data_Bus : Std_Logic_Vector (15 downto 0);
. . .
Data_Bus <= (14 downto 8 => '0', others => '1'); -- utilisation du choix others (en dernier)

-- Exemple 6
signal Data_Bus : Std_Logic_Vector (15 downto 0);
. . .
Data_Bus <= (others => 'Z'); -- utilisation du choix others par lui-même
```

Décodeur général – modèle VHDL

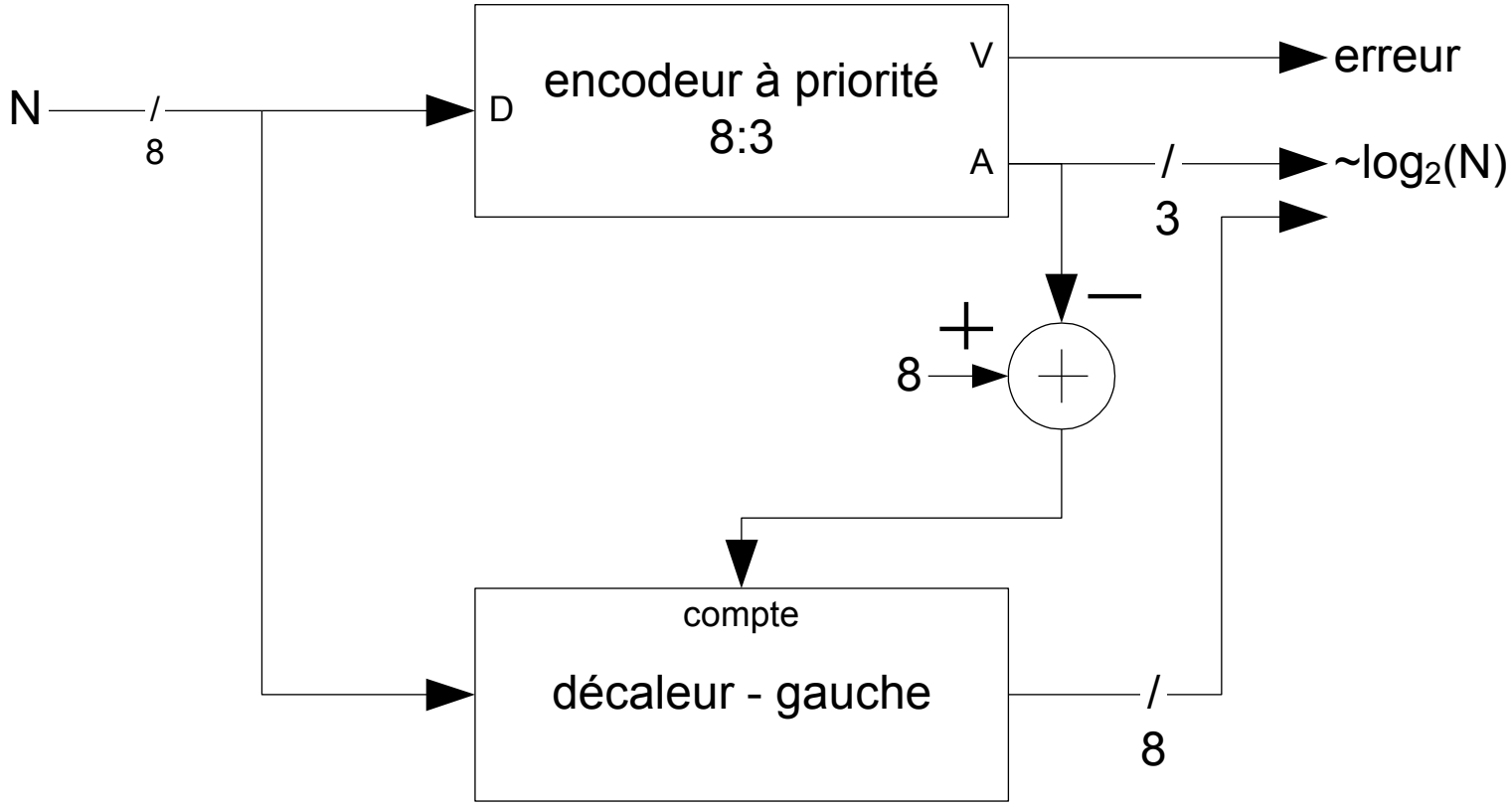
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity decodeur is
   generic (
      n : positive := 3 -- nombre de signaux d'entrée
   );
   port(
      A : in std_logic_vector(n - 1 downto 0);
      F: out std_logic_vector(2 ** n - 1 downto 0)
   );
end decodeur;

architecture comportementale of decodeur is
begin

   process(A)
   begin

      F <= (to_integer(unsigned(A)) => '1', others => '0');

   end process;

end comportementale;
```

Module combinatoire utile: l'encodeur à priorité – exemple d'utilisation

- L'indice du '1' le plus significatif d'un nombre binaire non signé donne la valeur plancher du logarithme en base 2 de ce nombre: on peut le trouver avec un encodeur à priorité.
- Les autres bits du nombre peuvent être interprétés comme une valeur binaire fractionnaire.
- La somme de l'indice et de la valeur binaire fractionnaire donne une approximation du logarithme en base 2 du nombre.

N, base 10	N, base 2	indice du '1' le plus significatif	partie fractionnaire	~$\log_2$N = indice + partie fract. (base 2)	~$\log_2$N = indice + partie fract. (base 10)	$\log_2$N
4	00000100	2	.00	010.0000000	2.0	2.0
7	00000111	2	.11	010.1100000	2.75	2.8073…
15	00001111	3	.111	011.1110000	3.875	3.9068…
65	01000001	6	.000001	110.0000010	6.015625	6.0223…
237	11101101	7	.1101101	111.1101101	7.8515625	7.8887…

Module combinatoire utile: l'encodeur à priorité – exemple d'utilisation



Encodeur à priorité – table de vérité

- Un encodeur identifie un signal actif parmi un ensemble de signaux, et produit un code qui correspond à ce signal actif.
- Un encodeur fonctionne de façon contraire à un décodeur.
  - Il a *n* lignes de sortie et 2^n lignes d'entrée.
  - Le code à la sortie représente le numéro de la ligne qui est active.
  - Un signal de sortie spécial indique si au moins une des lignes en entrée est active.
- Un encodeur à priorité permet d'avoir plus d'une ligne d'entrée active à la fois. La priorité peut être accordée à la ligne ayant le plus grand ou le plus petit numéro, ou selon un autre ordre.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0	V
0	0	0	0	0	0	0	0	-	-	-	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	-	0	0	1	1
0	0	0	0	0	1	-	-	0	1	0	1
0	0	0	0	1	-	-	-	0	1	1	1
0	0	0	1	-	-	-	-	1	0	0	1
0	0	1	-	-	-	-	-	1	0	1	1
0	1	-	-	-	-	-	-	1	1	0	1
1	-	-	-	-	-	-	-	1	1	1	1

Table de vérité partielle d'un encodeur 8:3
(D7, D6, …, D0) sont les entrées.
(A2, A1, A0 et V) sont les sorties.

Encodeur à priorité – modèle VHDL

- La priorité est donnée aux lignes avec un numéro élevé.
- Au début du processus, on donne une valeur par défaut aux signaux de sortie V et A, au cas où aucune des entrées n'est active.
- La valeur par défaut donnée au signal A est un « peu-importe » (don't-care), représenté pour le type std_logic par un tiret '-'.

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

entity encodeurP is
   generic (
      n : positive := 3 -- largeur du code de sortie
   );
   port(
      D : in std_logic_vector(2 ** n - 1 downto 0); -- le bus d'entrée
      A : out std_logic_vector(n - 1 downto 0); -- le code de sortie
      V : out std_logic -- '1' si au moins un signal d'entrée est actif
   );
end encodeurP;

architecture comportementale of encodeurP is
begin

   process(D)
   begin
      -- des valeurs par défaut sont essentielles
      -- au cas où aucun signal d'entrée n'est actif
      V <= '0';
      A <= (others => '-');
      for k in 2 ** n - 1 downto 0 loop -- priorité aux valeurs élevées
         if D(k) = '1' then
            A <= std_logic_vector(to_unsigned(k, n));
            V <= '1';
            exit; -- termine la boucle
         end if;
      end loop;
   end process;

end comportementale;
```

Vous devriez maintenant être capable de …

- Utiliser des décodeurs et des encodeurs à priorité dans un chemin de données. (B3)
- Reconnaître et utiliser la modélisation de décodeurs et encodeurs en VHDL. (B2, B3)
- Effectuer le processus de synthèse d'un décodeur et d'un encodeur du modèle VHDL et son implémentation dans un FPGA. (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.