Unité: Base de systèmes logiques (SysLog1 & 2)
Conception de systèmes numériques (CSN)

Les fonctions standards combinatoires

Version SysLog2

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

E. Messerli & R. Mosqueron

Février 2022

Polycopié : Electronique numérique

- Fonctions standards combinatoires

  chapitre 5-7 et suivants, pages 65 à 86

  - Décodeur chapitre 5-8, pages 66 à 70
  - Multiplexeur chapitre 5-9, pages 70 à 78
  - Comparateur chapitre 5-10, pages 79 à 81
  - Addition binaire chapitre 5-11, pages 82 à 85

    + Arithmétique et nombres signés :

    chapitre 3-1 à 3-5, pages 21 à 31

Qu'appelle-t-on fonctions standards

- Modules logiques renfermant une fonctionnalité simple
- Ces fonctionnalités correspondent à des éléments logiques fréquemment utilisés
- Ces modules se retrouvent fréquemment sous une forme simple ou combinée dans les systèmes logiques

Fonctions standards combinatoires

- Décodage X/Y
- Multiplexage
- Comparaison
- *Encodage de priorité*
- Opérations arithmétiques
  - Addition/ soustraction
- Méthodologie de conception et décomposition des systèmes combinatoires

Décodeur X/Y

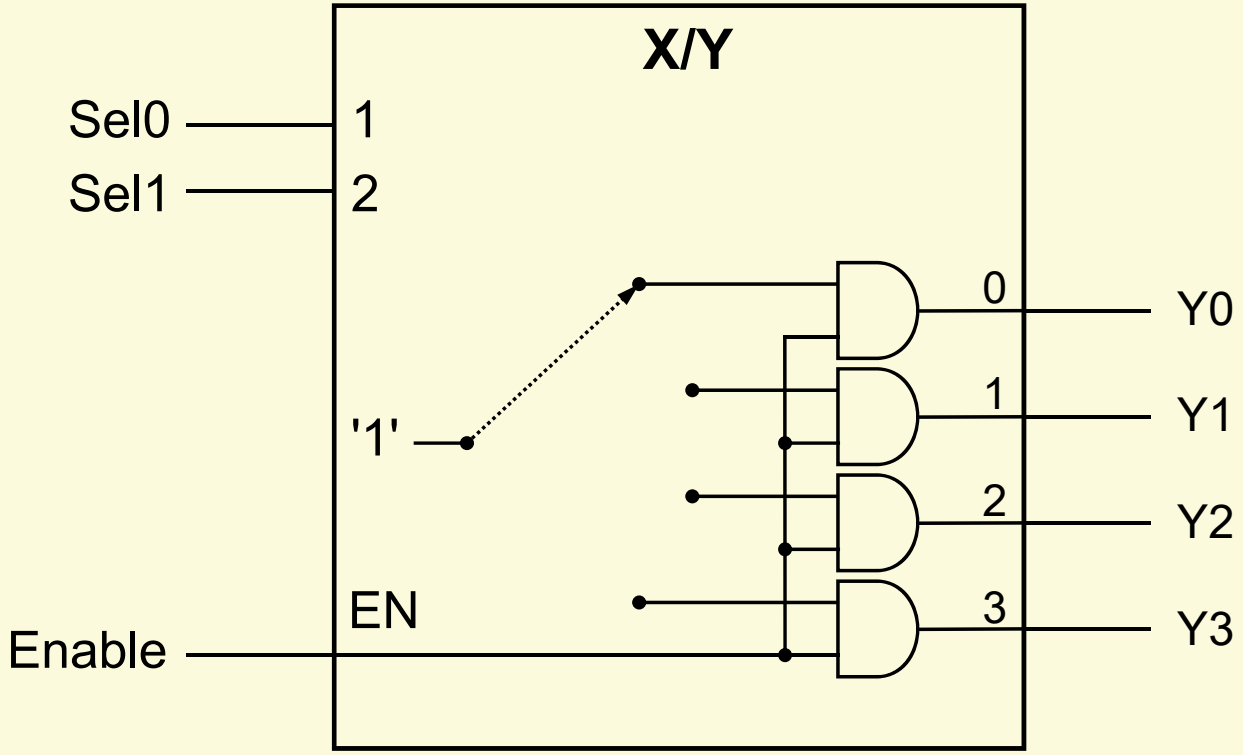
- But : activer la sortie dont on donne le numéro sous forme binaire (entier non signé)

  => décode la valeur binaire d'entrée de **n** bits

  => génère tous les mintermes de l'entrée **n** bits
- Une **seule** sortie active simultanément (minterme)
- Comporte souvent une entrée d ’activation (*Enable*) . Cette entrée est indispensable pour étendre le décodage.

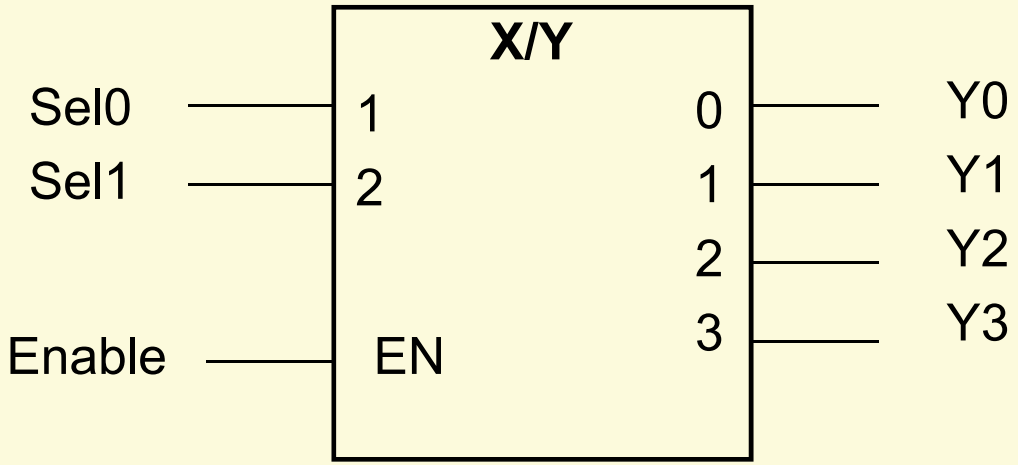
Décodeur 2 à 4 avec EN

- Fonctionnement de principe



Décodeur 2 à 4 avec EN, symbole

- Version avec *Enable*



Décodeur 2 à 4 avec EN (TDV, équ., schéma)

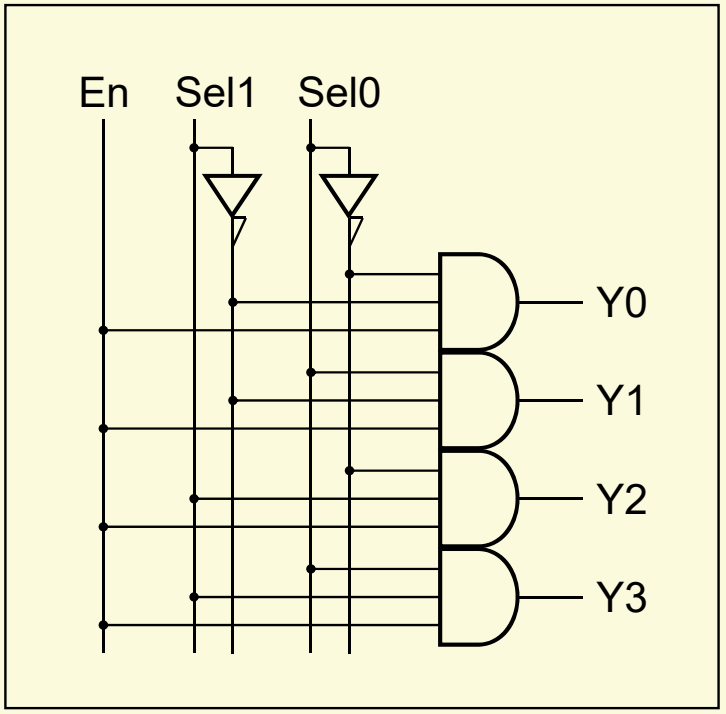
- Génère tous les mintermes de Sel1 et Sel0

Table de vérité

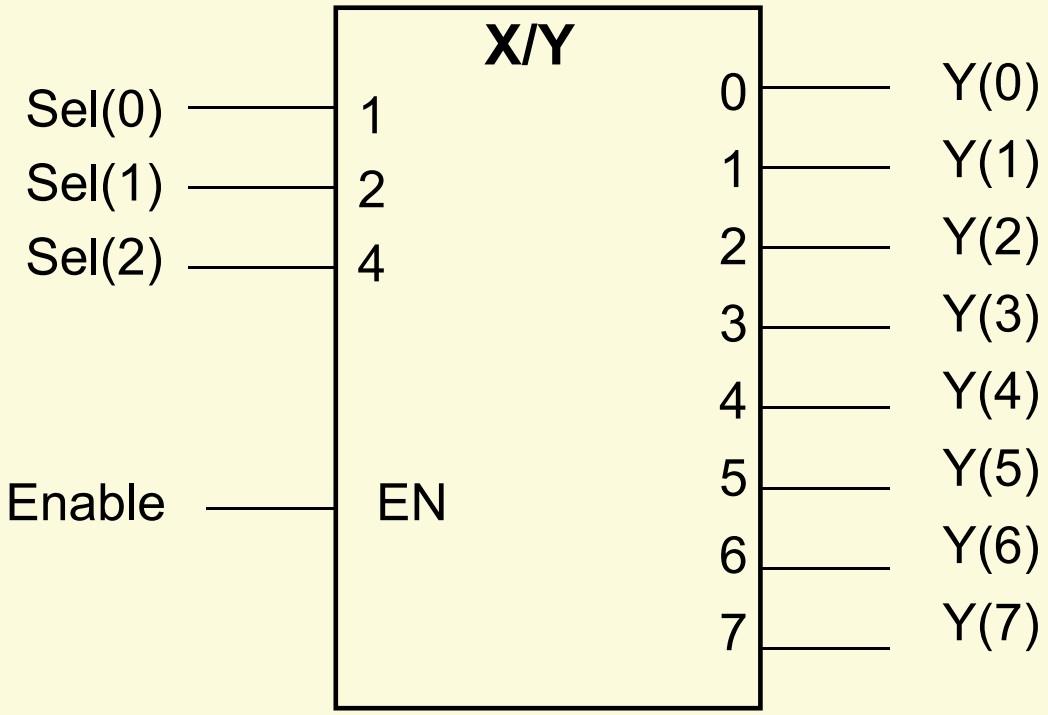
En	Sel1	Sel0	Y3	Y2	Y1	Y0
'0'	x	x	'0'	'0'	'0'	'0'
'1'	'0'	'0'	'0'	'0'	'0'	'1'
'1'	'0'	'1'	'0'	'0'	'1'	'0'
'1'	'1'	'0'	'0'	'1'	'0'	'0'
'1'	'1'	'1'	'1'	'0'	'0'	'0'

$Y0 = En \cdot (\overline{Sel1} \cdot \overline{Sel0})$ $Y1 = En \cdot (\overline{Sel1} \cdot Sel0)$

$Y2 = En \cdot (Sel1 \cdot \overline{Sel0})$ $Y3 = En \cdot (Sel1 \cdot Sel0)$



Décodeur 3 à 8, symbole CEI



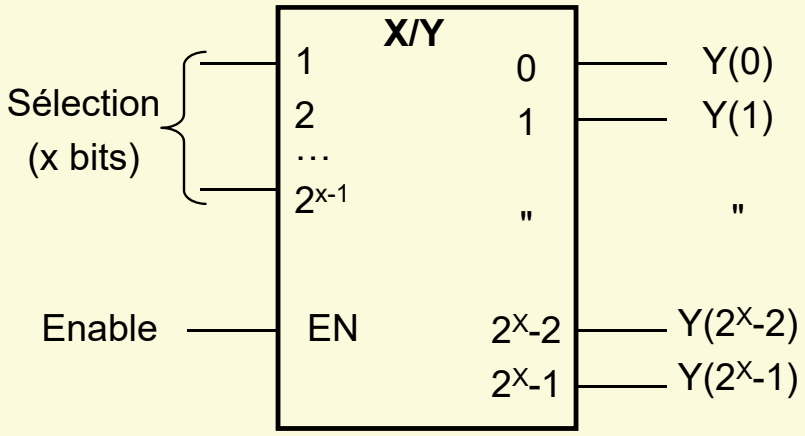
Symbole IEEE/CEI **RECOMMANDÉ**

Décodeur 3 à 8, table de vérité

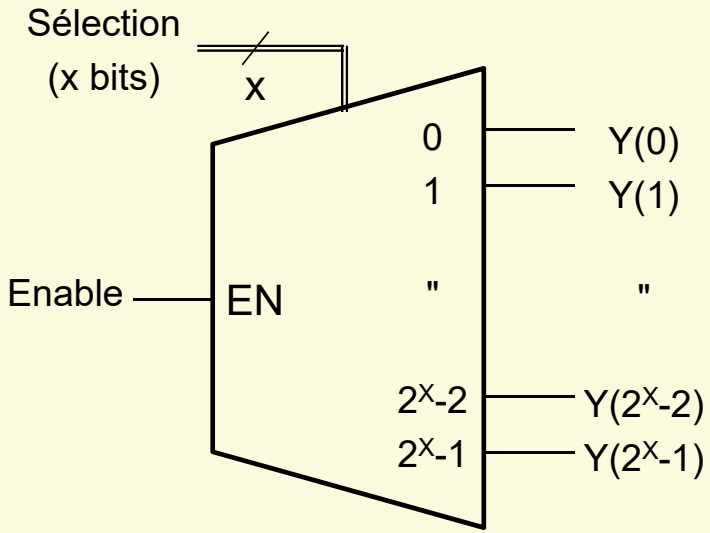
EN	Sel(2)	Sel(1)	Sel(0)	Y(7)	Y(6)	Y(5)	Y(4)	Y(3)	Y(2)	Y(1)	Y(0)
'0'	x	x	x	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'
'1'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'1'
'1'	'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'	'0'	'1'	'0'
'1'	'0'	'1'	'0'	'0'	'0'	'0'	'0'	'0'	'1'	'0'	'0'
'1'	'0'	'1'	'1'	'0'	'0'	'0'	'0'	'1'	'0'	'0'	'0'
'1'	'1'	'0'	'0'	'0'	'0'	'0'	'1'	'0'	'0'	'0'	'0'
'1'	'1'	'0'	'1'	'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'
'1'	'1'	'1'	'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'	'0'
'1'	'1'	'1'	'1'	'1'	'0'	'0'	'0'	'0'	'0'	'0'	'0'

Décodeur X/Y (symbole)

- Symbole décodeur x à n, avec n = 2^X



Symbole IEEE/CEI RECOMMANDÉ

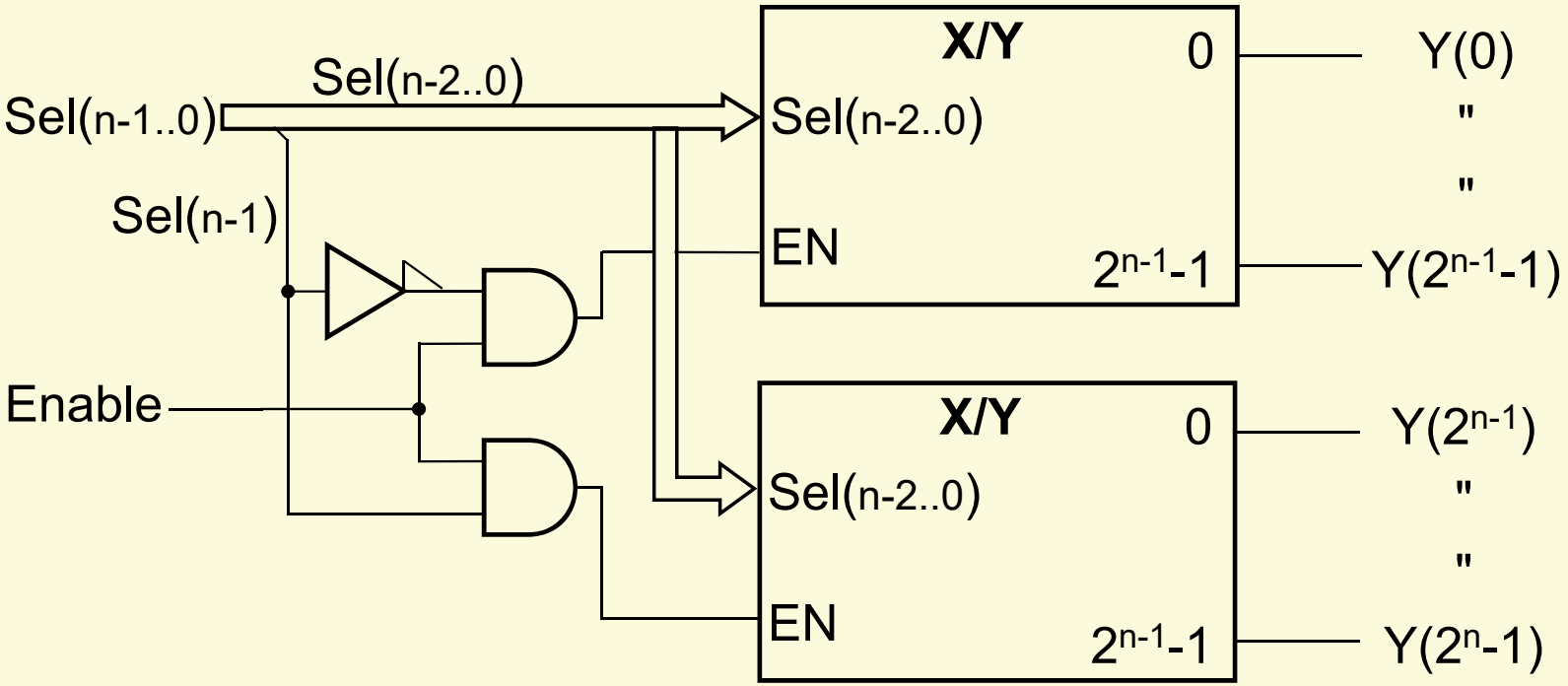


Symbole US ACCEPTÉ

utilisation de symbole explicite !

Décodeur, schéma de décomposition

- Un décodeur n bits peut se réaliser au moyen de deux décodeurs n-1 bits



Décodeur « entity »

- « entity » les entrées/sorties du module

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

entity dec_2a4 is
  port(sel_i : in  std_logic_vector(1 downto 0);
               -- Entrees de selection
       en_i  : in std_logic ;
               -- Entree de validation
       y_o   : out std_logic_vector(3 downto 0)
               -- sortie
           );
end dec_2a4 ;
```

Décodeur « architecture logique»

- Description de la fonctionnalité en exprimant les équations

```
architecture logique of dec_2a4 is
  signal y_s : std_logic_vector(3 downto 0) ;
begin
   -- equations logiques
   y_s(0) <= not sel_i(1) and not sel_i(0); --minterme 0
   y_s(1) <= not sel_i(1) and     sel_i(0); --minterme 1
   y_s(2) <=     sel_i(1) and not sel_i(0); --minterme 2
   y_s(3) <=     sel_i(1) and     sel_i(0); --minterme 3

   -- affectation de la valeur de sortie (gestion enable)
   y_o <= y_s   when (en_i = '1') else
          "0000";
end logique;
```

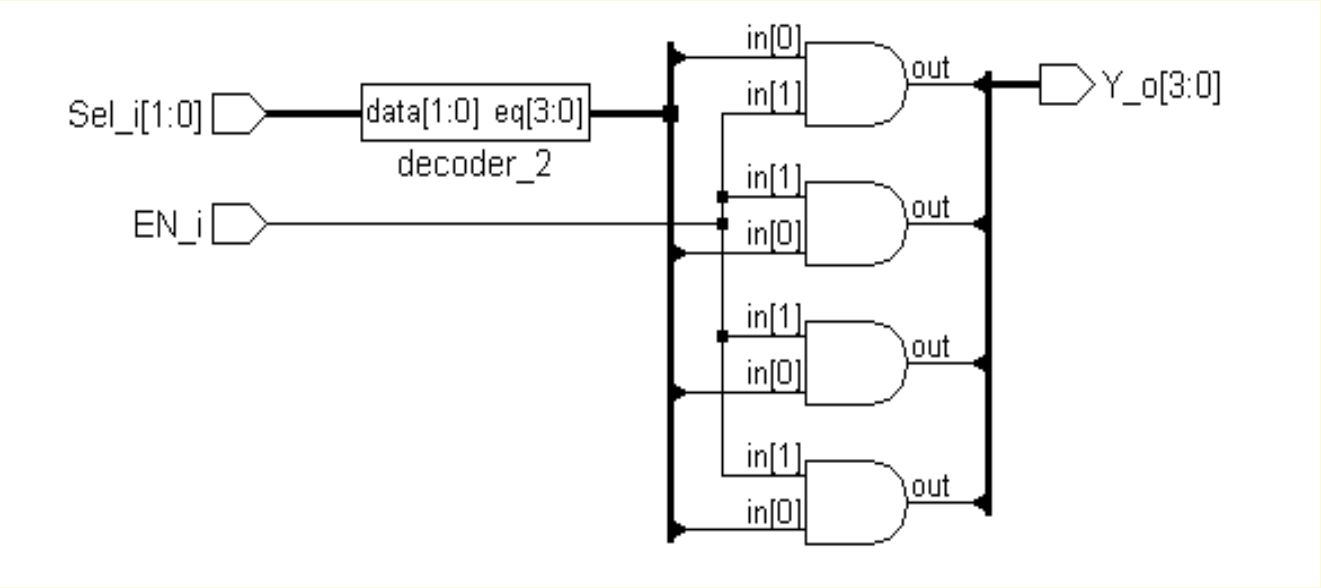
Décodeur « architecture flot_don»

- Description en explicitant le décodeur de base (sans enable) :

```
architecture flot_don of dec_2a4 is
  signal y_s : std_logic_vector(3 downto 0) ;
begin
   -- determination de la valeur de sortie
   with sel_i select
      y_s <= "0001" when "00",
             "0010" when "01",
             "0100" when "10",
             "1000" when "11",
             "XXXX" when others ;-- simulation

   -- affectation de la valeur de sortie
   y_o <= y_s   when (en_i = '1') else
          "0000";
end flot_don;
```

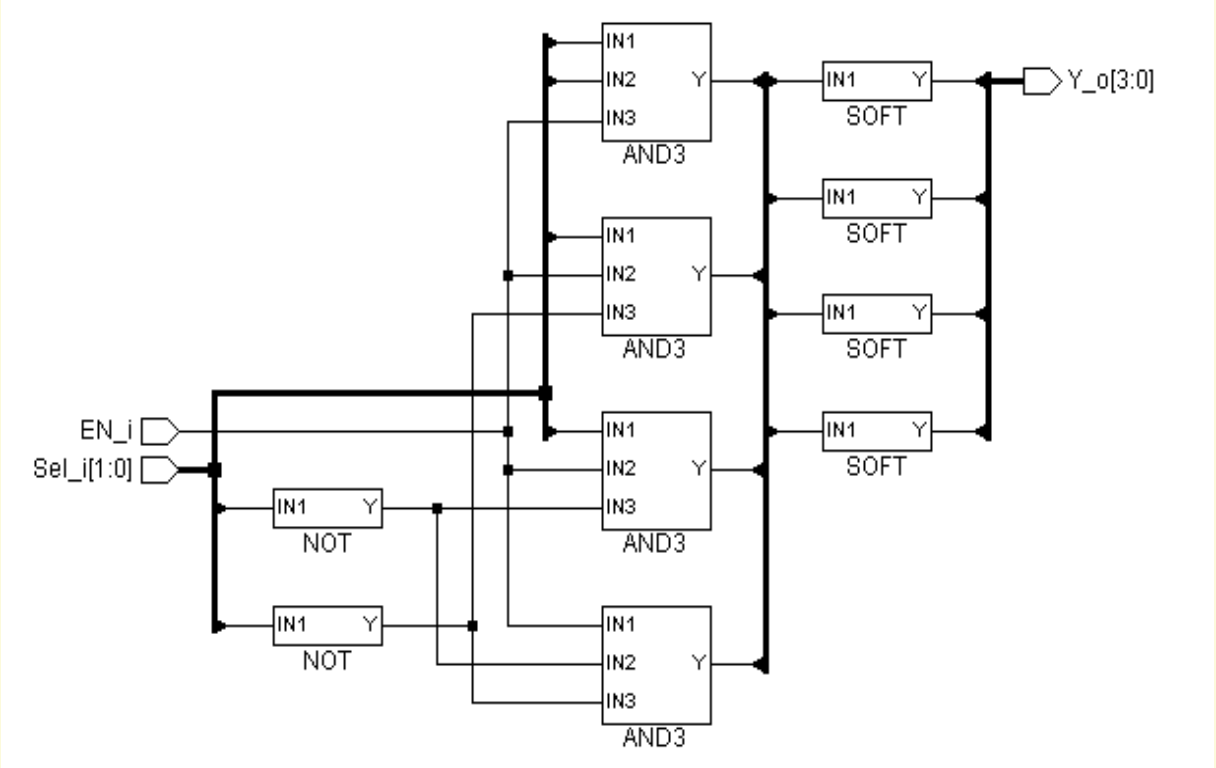
Décodeur, schéma RTL (arch. flot_don)

- Interprétation «fonctionnelle» du synthétiseur



Décodeur, vue "Technology" (arch. flot_don)

- Traduction en «logique» du synthétiseur

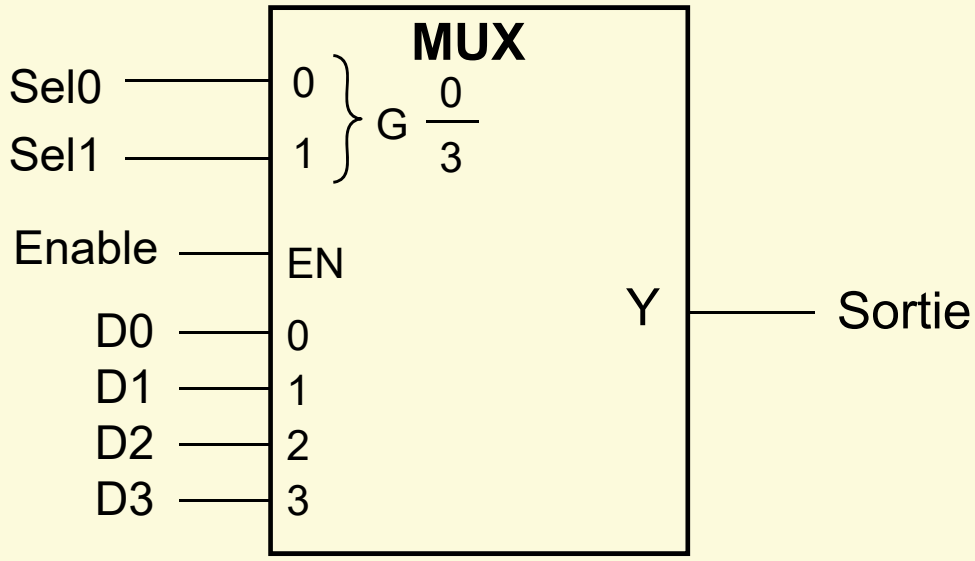


Multiplexeur

- But : Transmettre sur la sortie l'entrée sélectionnée par son numéro
- Une entrée supplémentaire « Enable » est souvent présente pour faciliter l'extension
- Les multiplexeurs sont utilisables à la place de portes logiques pour réaliser des fonctions combinatoires quelconques

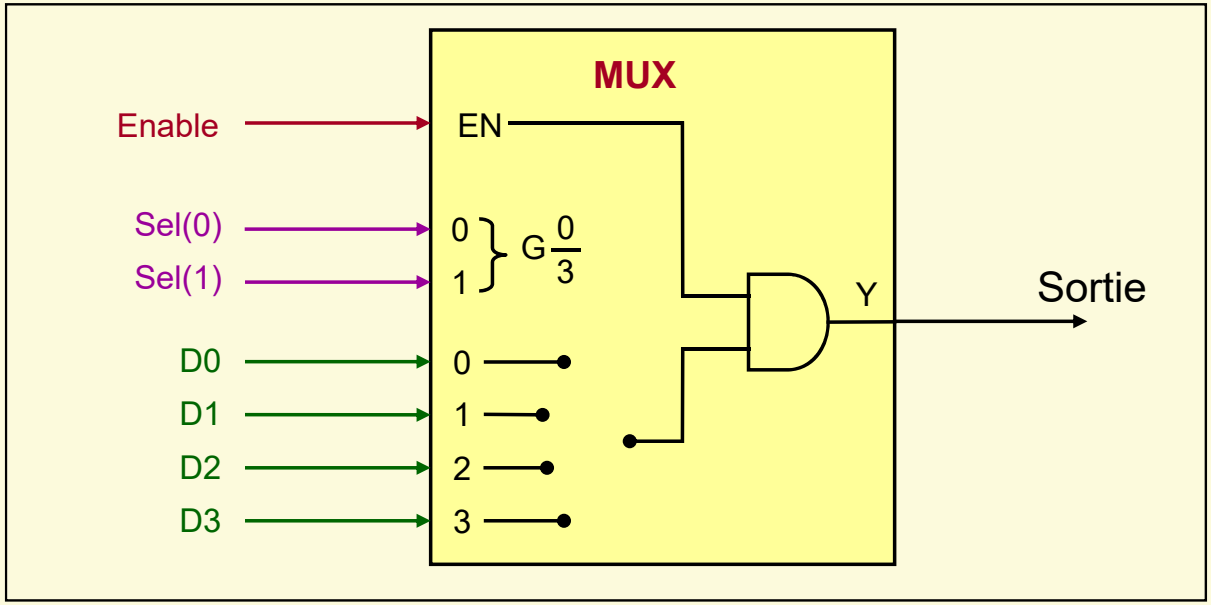
Multiplexeur 4 à 1, symbole

- Symbole IEEE/CEI



Exemple fonctionnel du multiplexeur

- Un multiplexeur 4 à 1 est un sélecteur à:
  - 1 ligne d'autorisation Enable, activation du circuit
  - 2 lignes de sélection Sel(1..0), choix de l'entrée
  - 4 lignes d'entrées D(3 .. 0)



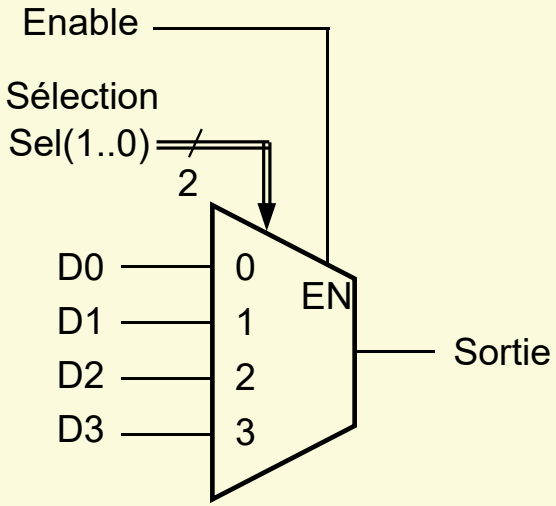
Multiplexeur 4 à 1, table de vérité

- Système combinatoire à 7 entrées
  => Table de vérité comprend 128 lignes !
- Table de vérité compactée :

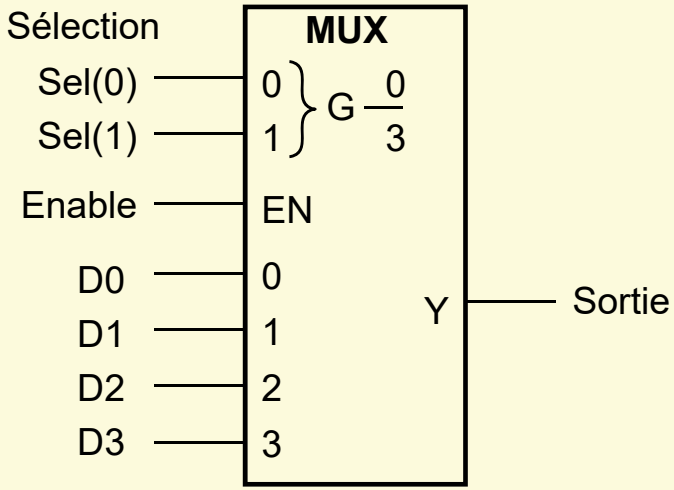
EN	Sel(1)	Sel(0)	Y
0	x	x	0
1	0	0	D0
1	0	1	D1
1	1	0	D2
1	1	1	D3

Multiplexeur 4 à 1, symbole

- Symboles multiplexeur 4 à 1:



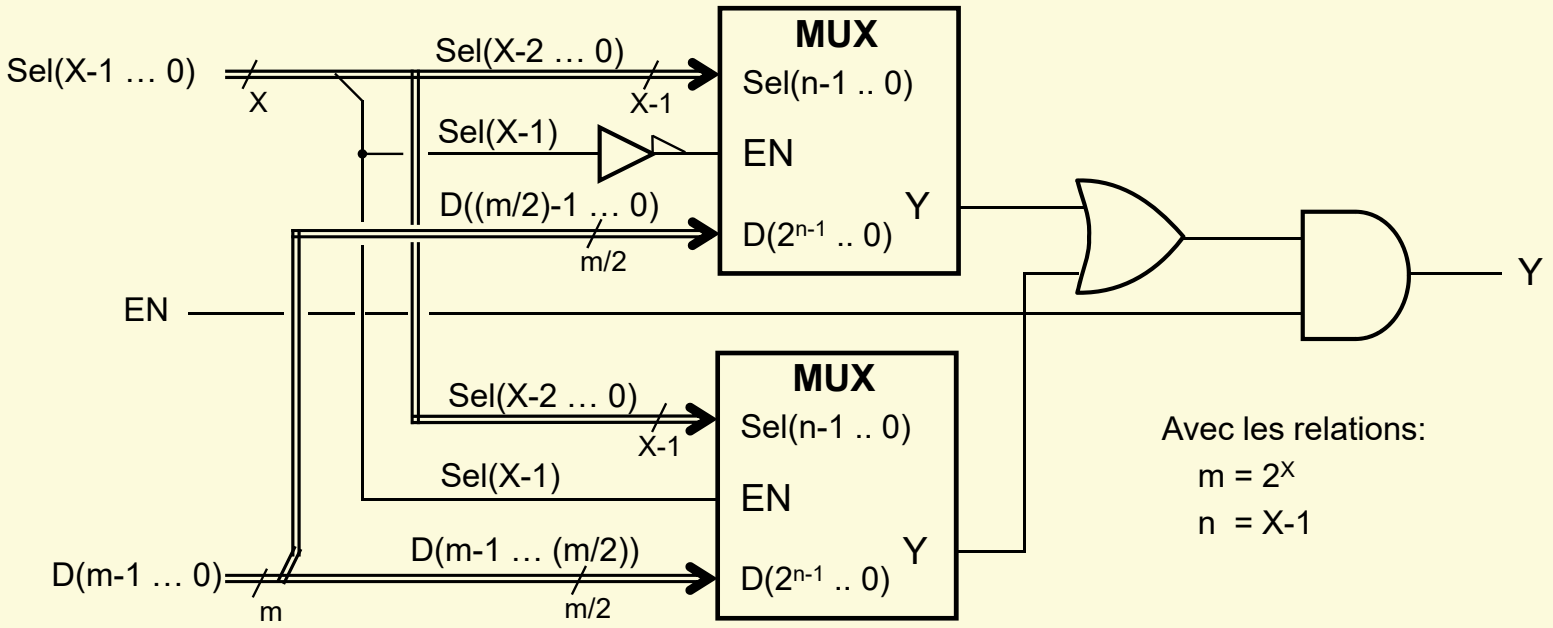
Symbole US **ACCEPTÉ**



Symbole IEEE/CEI **RECOMMANDÉ**

Multiplexeur m to 1 (décomposition)

- Un multiplexeur m to 1 peut toujours se réaliser au moyen de deux multiplexeurs m/2 to 1 et des portes



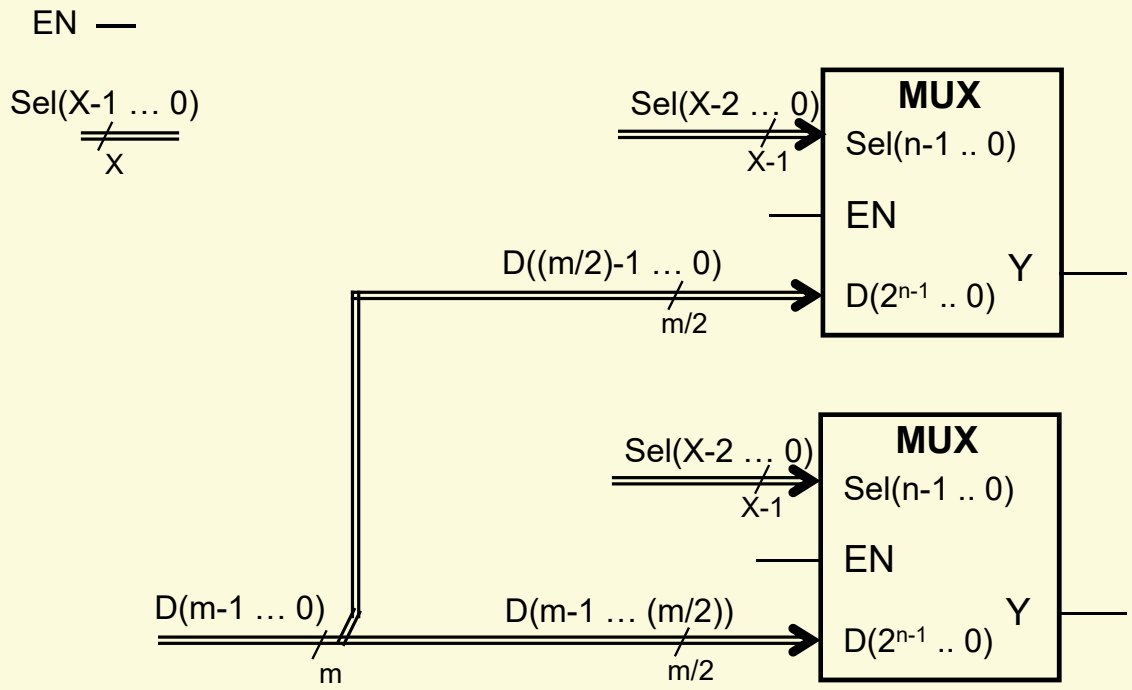
Exercice

- Réaliser un multiplexeur m to 1 avec deux multiplexeurs m/2 to 1 et un mux 2a1

Avec :

$m = 2^X$

$n = X-1$



Multiplexeur « entity »

- « entity » les entrées/sorties du module

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

entity mux_4x1 is
  port(sel_i : in  std_logic_vector(1 downto 0);
               -- Entrees de selection
       en_i  : in  std_logic ;
               -- Entree de validation
       d0_i, d1_i, d2_i, d3_i  : in  std_logic;
               -- Entree des donnees
       y_o   : out std_logic
               -- sortie
           );
end mux_4x1 ;
```

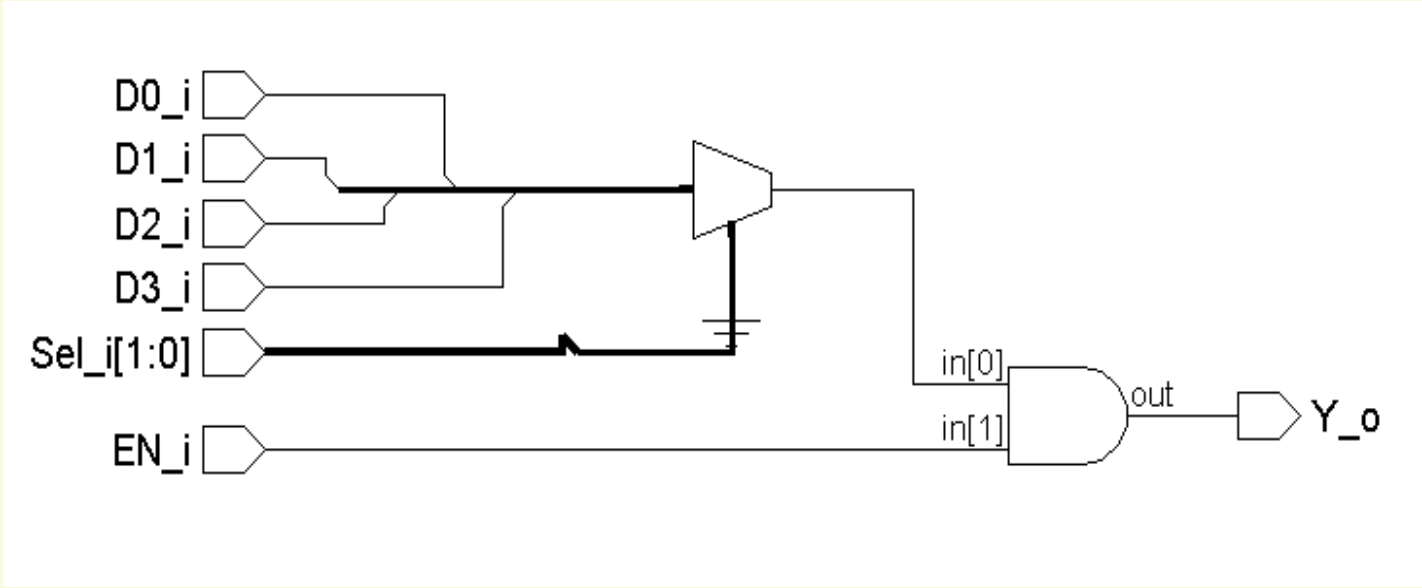
Multiplexeur « architecture »

- Description de la fonctionnalité du bloc

```
architecture flot_don of mux_4x1 is
   signal y_s : std_logic ;
begin
   -- determination de la valeur de sortie
   with sel_i select
      y_s <= d0_i when "00",
             d1_i when "01",
             d2_i when "10",
             d3_i when "11",
             'X'  when others; -- simulation
   -- affectation de la valeur de sortie
   y_o <= y_s when (en_i = '1') else
          '0' ;
end flot_don;
```

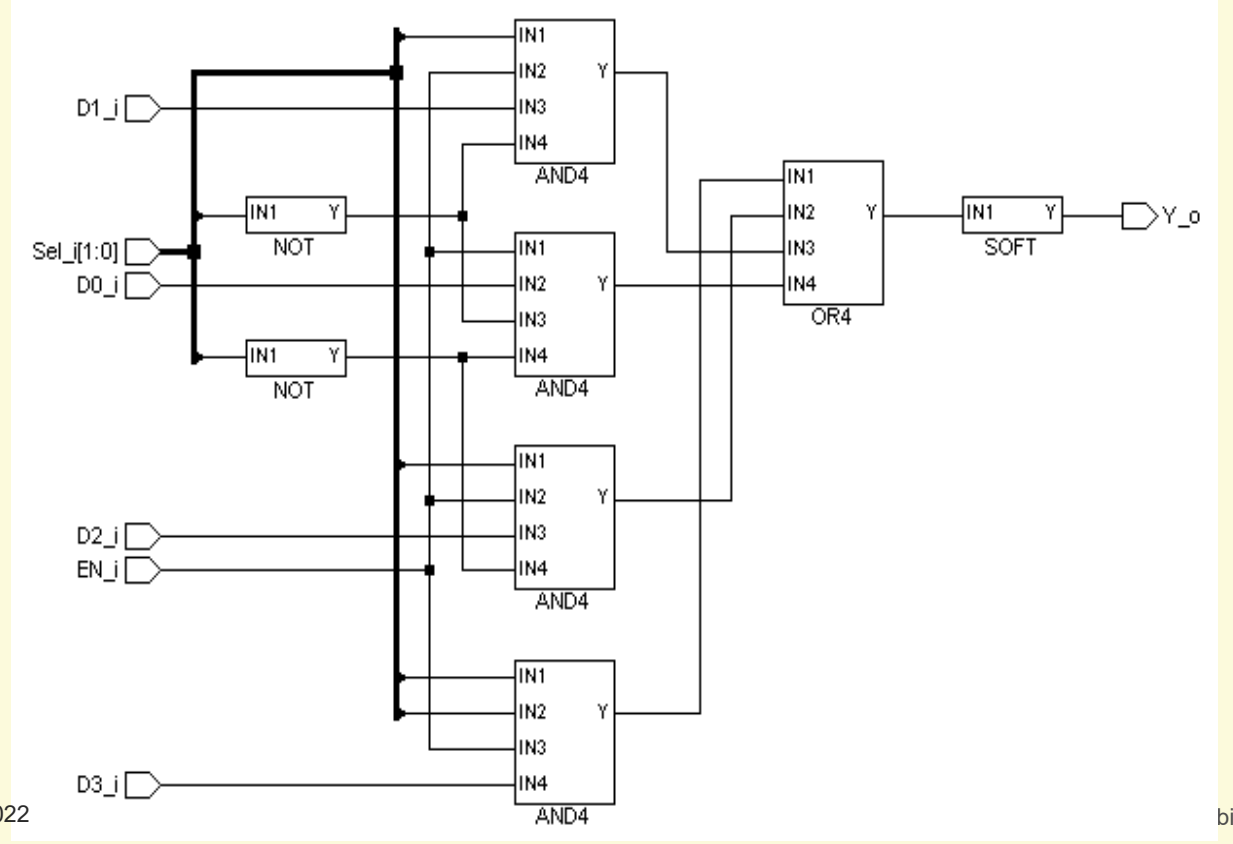
Multiplexeur, schéma RTL

- Interprétation «fonctionnelle» du synthétiseur



Multiplexeur, vue "Technology"

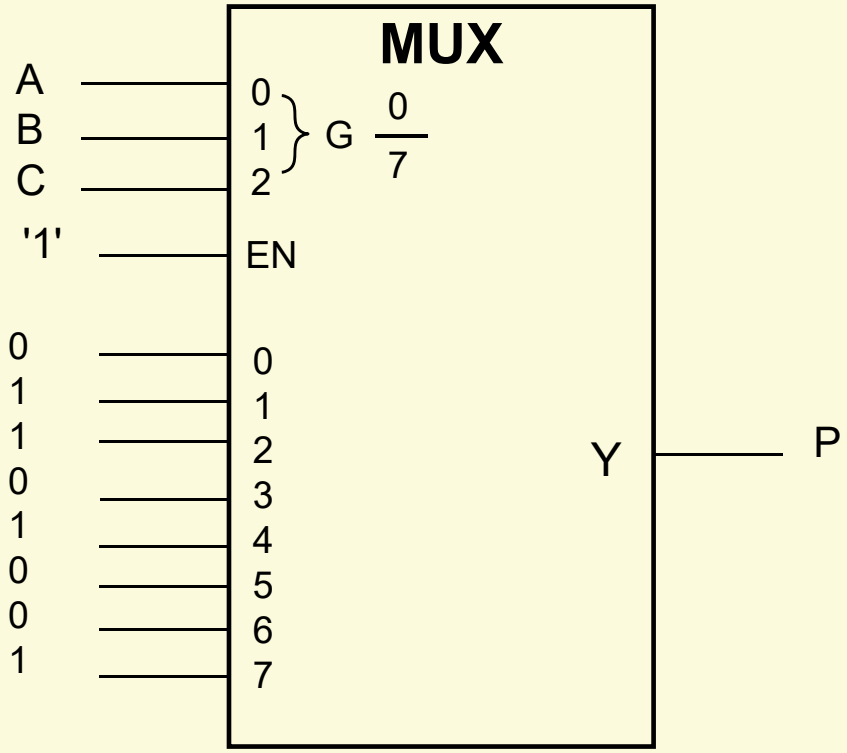
- Traduction en «logique» du synthétiseur



… exe: multiplexeur en générateur de fonction

- Exercice P(C,B,A) = Σ 1, 2, 4, 7

Correction

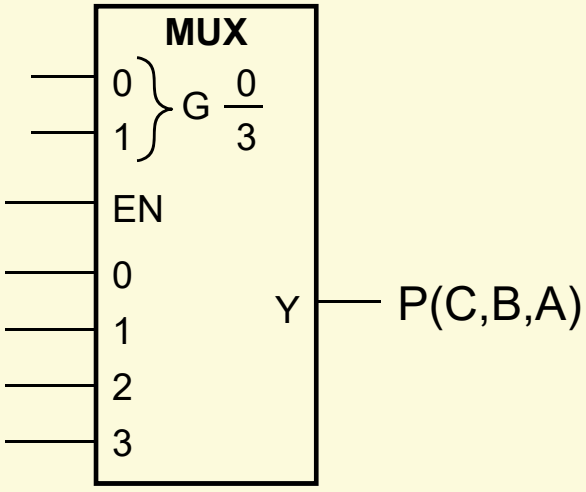


… exe: multiplexeur en générateur de fonction

- Exercice P(C,B,A) = Σ 1, 2, 4, 7
  - Chercher une solution en utilisant un mux 4 à 1

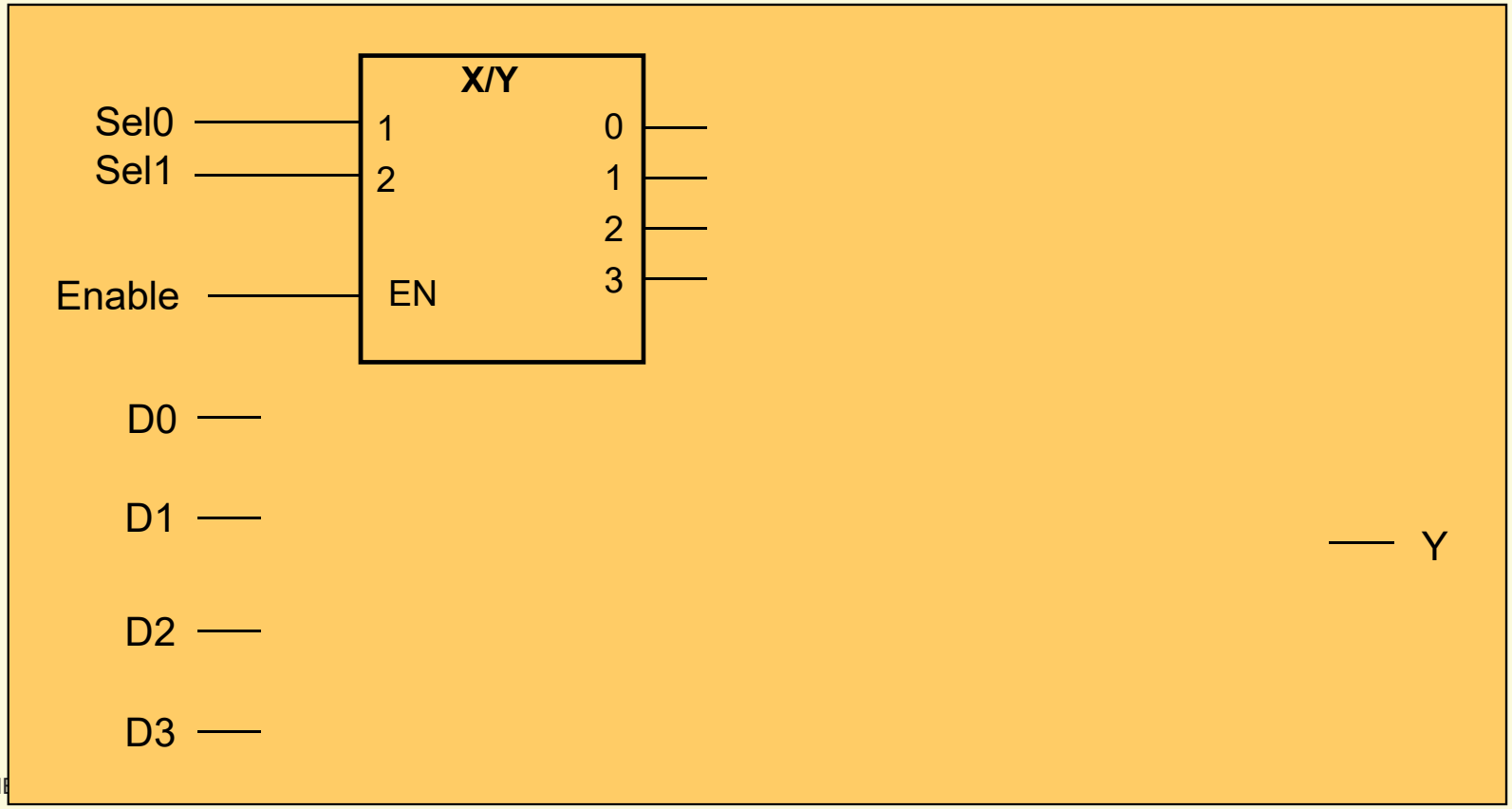
C	B	A	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A
B
C



Multiplexeur et décodeur

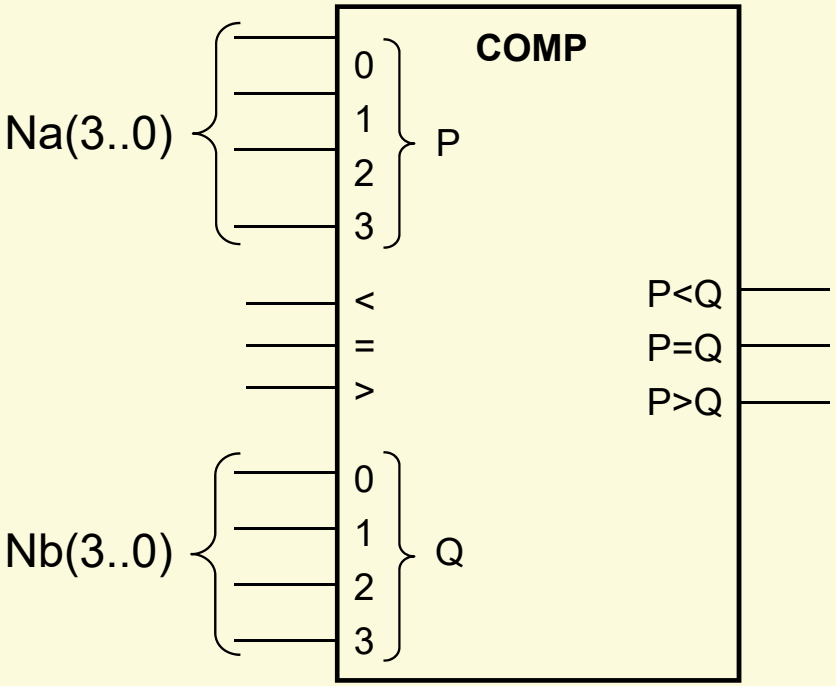
- Le multiplexeur peut être construit sur la base d'un décodeur

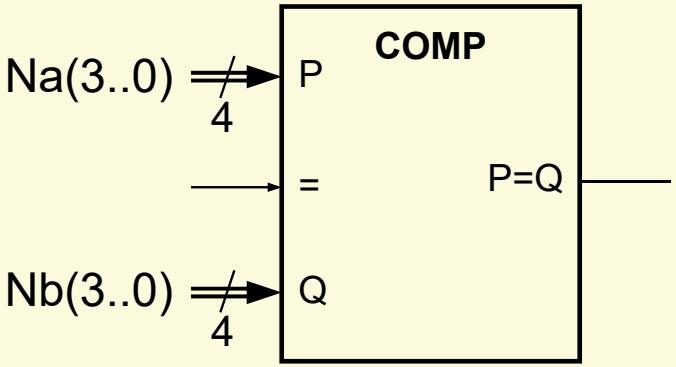


Comparateur

- But : indiquer si deux nombres binaires sont égaux
- Les sorties '<' ou '>' sont souvent ajoutées pour les applications numériques
- Modulaire si dispose d'entrées '<', '=' et '>'

Comparateur (symbole CEI)





Recommandé :
version compacte avec des bus

Symbole IEEE/CEI **OBLIGATOIRE**

Comparateur 4 bits « entity »

- « entity » les entrées/sorties du module

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

entity comp_4 is
  port(nb_a_i   : in  std_logic_vector(3 downto 0);
                -- Entree du nombre A
       nb_b_i   : in  std_logic_vector(3 downto 0);
                -- Entree du nombre B
       eq_i     : in  std_logic;
                -- Entree chainage (module precedent)
       a_eq_b_o : out std_logic
                -- sortie
           );
end comp_4 ;
```

Comparateur 4 bits « architecture »

- Description de la fonctionnalité du bloc

```
architecture flot_don of comp_4 is
  signal a_eq_b _s : std_logic;
begin
   -- determination de l'egalite entre A et B
   a_eq_b _s <= '1'  when (nb_a_i = nb_b_i) else
                '0';
   -- affectation de la sortie avec l'entree chainage
   a_eq_b _o <= a_eq_b_s  when (eq_i = '1') else
               '0';
end flot_don;
```

Attention: comparaison de position avec *std_logic_vector*
Recommandé : utiliser le paquetage *numeric_std*

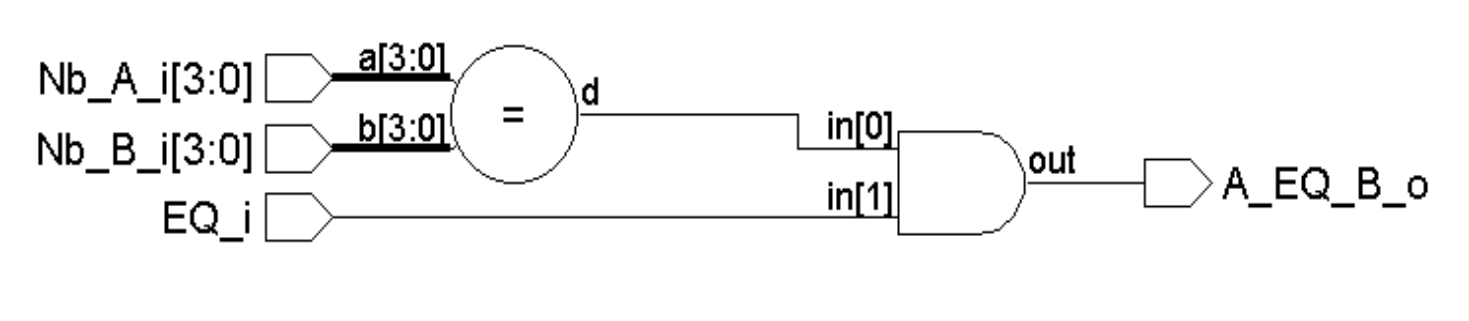
Comparateur 4 bits « architecture »

- Utiliser le paquetage *numeric_std* pour la comparaison

```
library ieee;
use ieee.std_logic_1164.all;
--Déclarer le paquetage
use ieee.numeric_std.all;

--Modifier la description de la comparaison comme suit :
-- détermination de l'égalite entre A et B
 a_eq_b_int <=
     '1'  when (unsigned(nb_a_i) = unsigned(nb_b_i)) else
     '0';
```

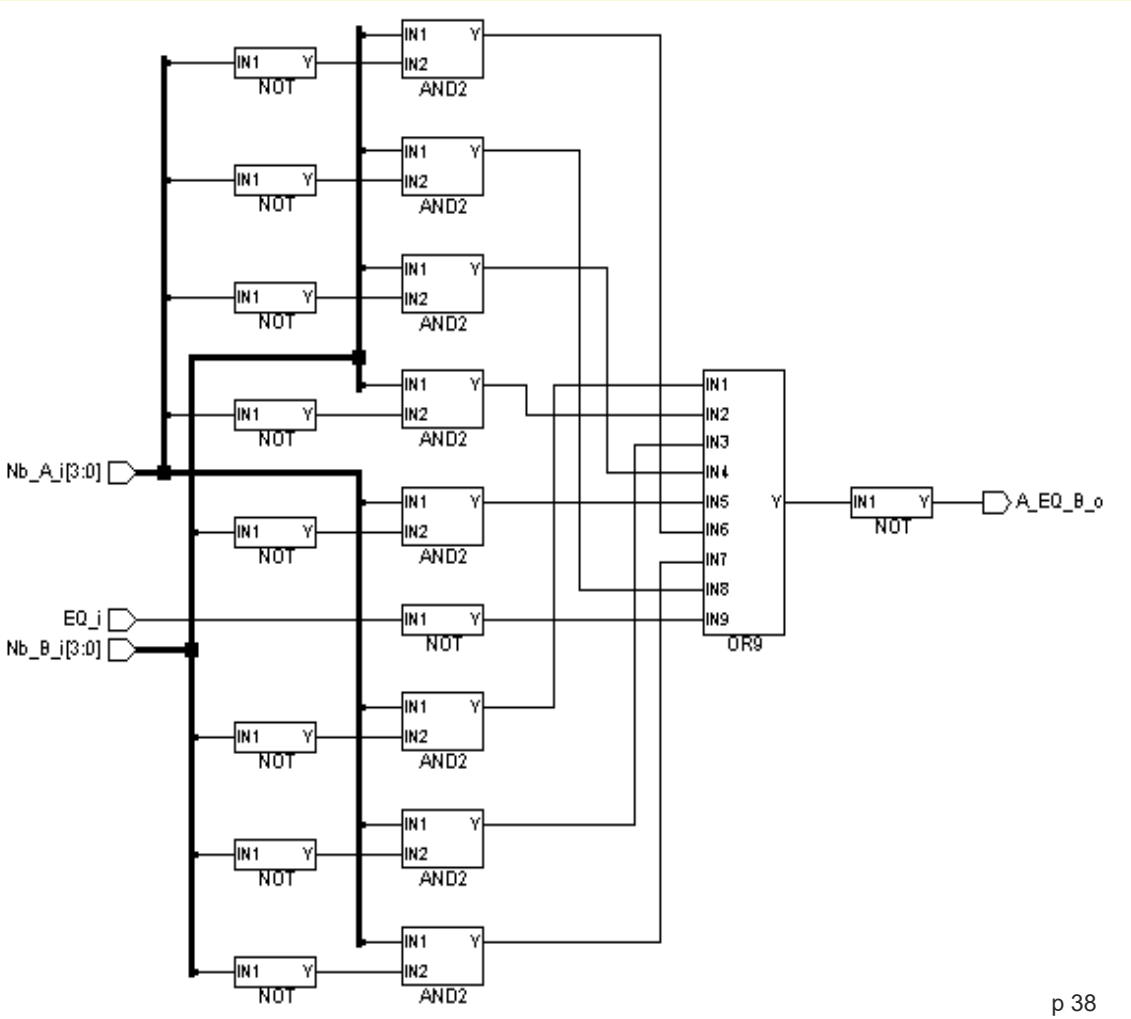
Comparateur 4 bits, vue RTL

- Interprétation «fonctionnelle» du synthétiseur



Comparateur 4 bits, vue Technology

- Traduction en «logique» du synthétiseur



Exercices I

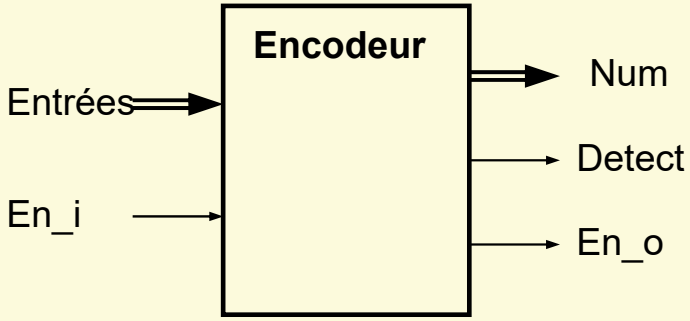
- Soit un comparateur fournissant l'égalité entre deux nombres de n bits.
  1. Donner l'équation logique de la sortie égalité (a_eq_b) pour des nombres de 5 bits.
     Indiquer le nombre total de termes de cette équation.
  2. Indiquer le nombre total de termes de l'équation si les nombres ont 10 bits.
  3. Quelle remarque pouvez-vous faire ?

Encodeur de priorité

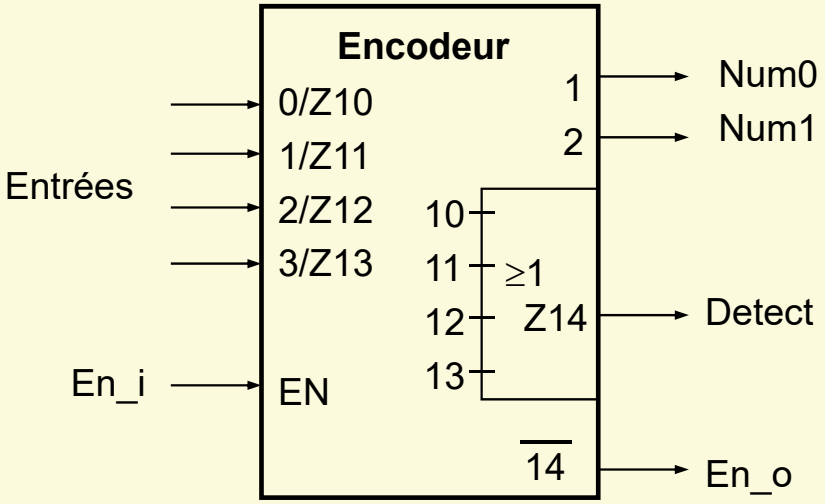
- But : déterminer le numéro de l ’entrée active ayant le degrés de priorité le plus élevé
- Ce module comporte souvent en plus une entrée d ’autorisation
- Ce module comporte une sortie indiquant qu’une des entrées au moins est active (Nécessaire pour identifier une action sur l ’entrée 0)

Encodeur (Schéma bloc)

- Exemple avec un encodeur à 4 entrées



Symbole non explicite **INTERDIT**



Symbole IEEE/CEI **OBLIGATOIRE**

Encodeur sans chaînage

Table de vérité (TDV) :

In3	In2	In1	In0	Detect	Num1	Num0
'0'	'0'	'0'	'0'	'0'	'-'	'-'
'0'	'0'	'0'	'1'	'1'	'0'	'0'
'0'	'0'	'1'	x	'1'	'0'	'1'
'0'	'1'	x	x	'1'	'1'	'0'
'1'	x	x	x	'1'	'1'	'1'

Equations logiques :

$$Detect = In3 + In2 + In1 + In0$$

$$Num1 = In3 + In2$$

$$Num0 = In3 + \overline{In2} \bullet In1$$

Description de l'encodeur

- Description textuelle du fonctionnement de l'encodeur de priorité :

Si l'entrée In3, la plus prioritaire, est active alors
Detect est activé & Num = 3
sinon ….

Si l'entrée In2, priorité suivante, est active alors
Detect est activé & Num = 2
sinon …

Description en VHDL de l'encodeur

- Sur la base de la description de la page précédente, décrire le comportement de l'encodeur de priorité avec l'instruction suivante :

```
S <= A when Condition1 else
     B when Condition2 else
     . . .
     G;
```

Encodeur « entity »

- Les entrées/sorties du module

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

entity encode4 is
  port(in_i     : in std_logic_vector(3 downto 0) ;
               -- Entrees (determiner priorite)
       en_i     : in std_logic ;
               -- Entree de validation
       num_o    : out std_logic_vector(1 downto 0) ;
               -- numero d l'entree la plus prioritaire
       detect_o : out std_logic ;
               -- Une entree detectee avec En_i actif
       en_o     : out std_logic     );
               -- Sortie de validation pour chaine
end encode4;
```

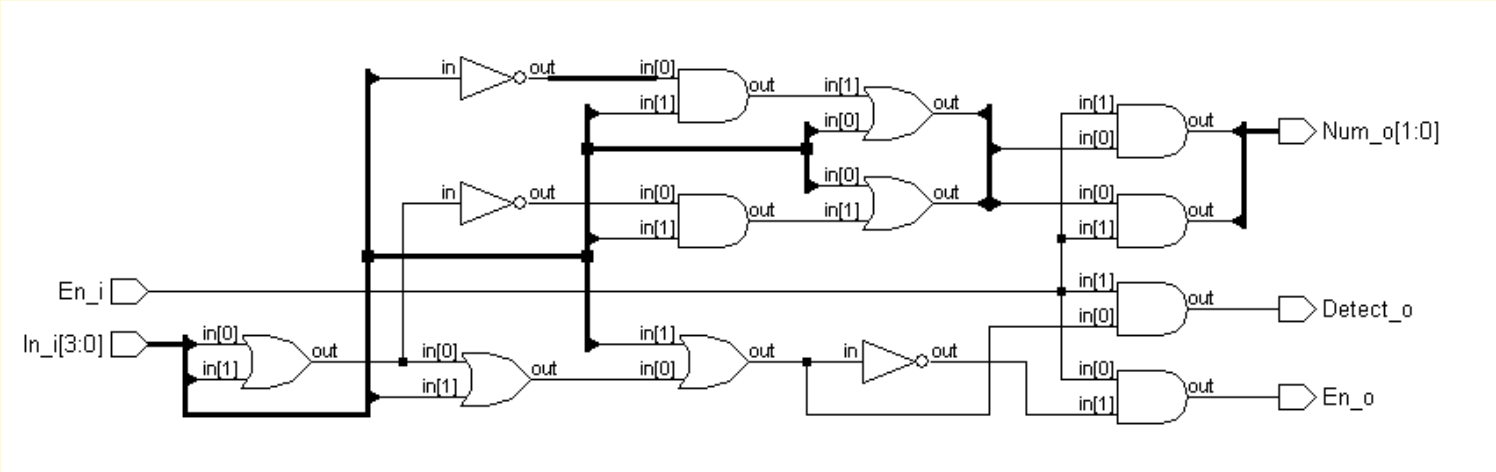
Encodeur « architecture »

- Description de la fonctionnalité du bloc

```
architecture flot_don of encode4 is
   signal num_s : std_logic_vector(1 downto 0);
   signal detect_s : std_logic ;
begin

   -- determination de la valeur de sortie
   num_s <=



   detect_s <=
   -- affectation valeurs de sortie
   num_o    <=
   detect_o <=
   en_o     <=
end flot_don;
```

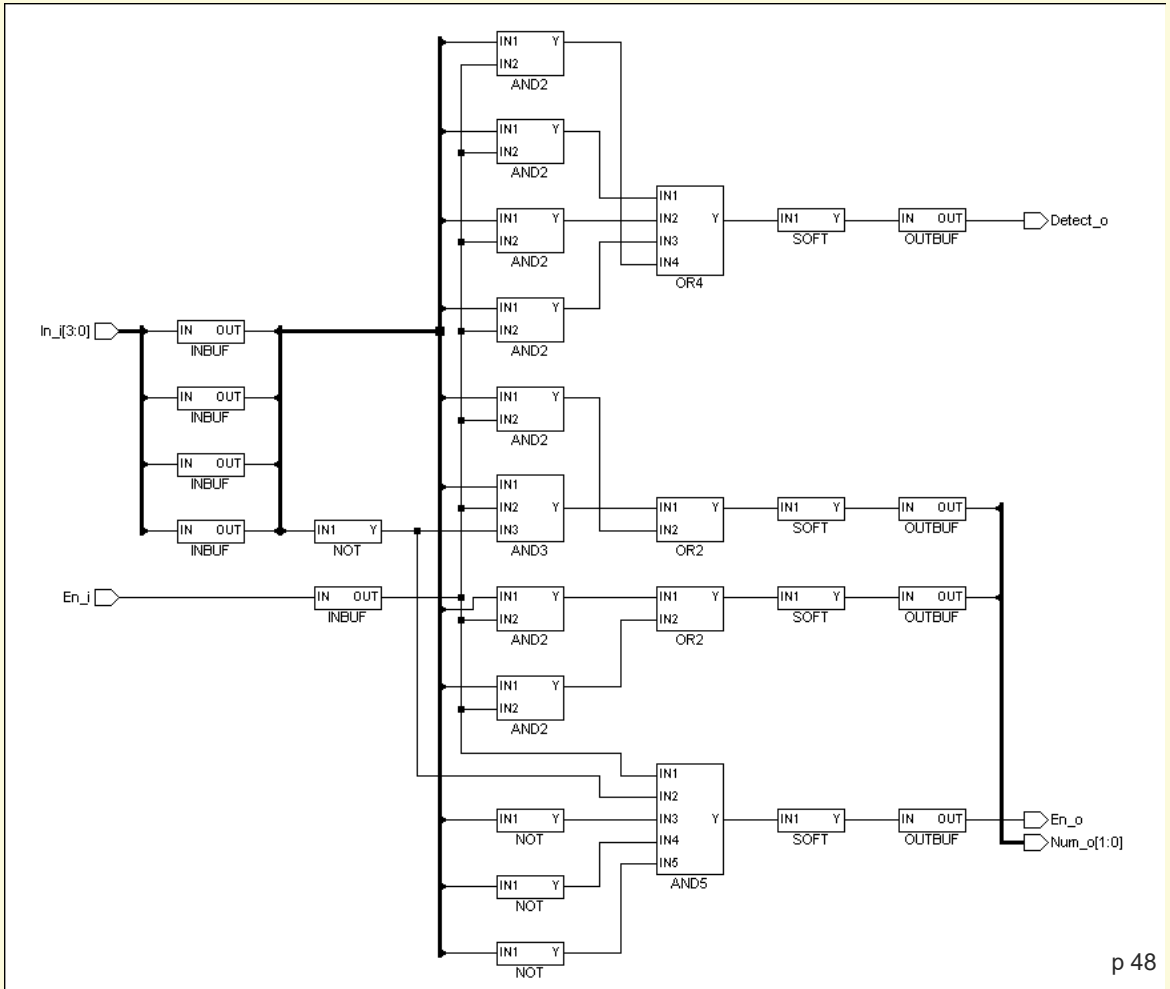
Encodeur, schéma RTL

- Interprétation «fonctionnelle» du synthétiseur



Encodeur, vue "Technology"

- Traduction en «logique» du synthétiseur



Additionneur

- But : réaliser l ’addition de deux nombres binaires
- Ce type de module se réalise fréquemment par une structure en chaîne; de ce fait, il comporte, en plus de l ’entrée des deux nombres, une entrée et une sortie pour les retenues

Additionneur 1 bit …

- Table de vérité additionneur 1 bit avec report:

C(i)	B(i)	A(i)	C(i+1)	S(i)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Simplification:

Tables de Karnaugh => équations logiques

$C(i+1) = C(i) \cdot B(i) + C(i) \cdot A(i) + B(i) \cdot A(i)$

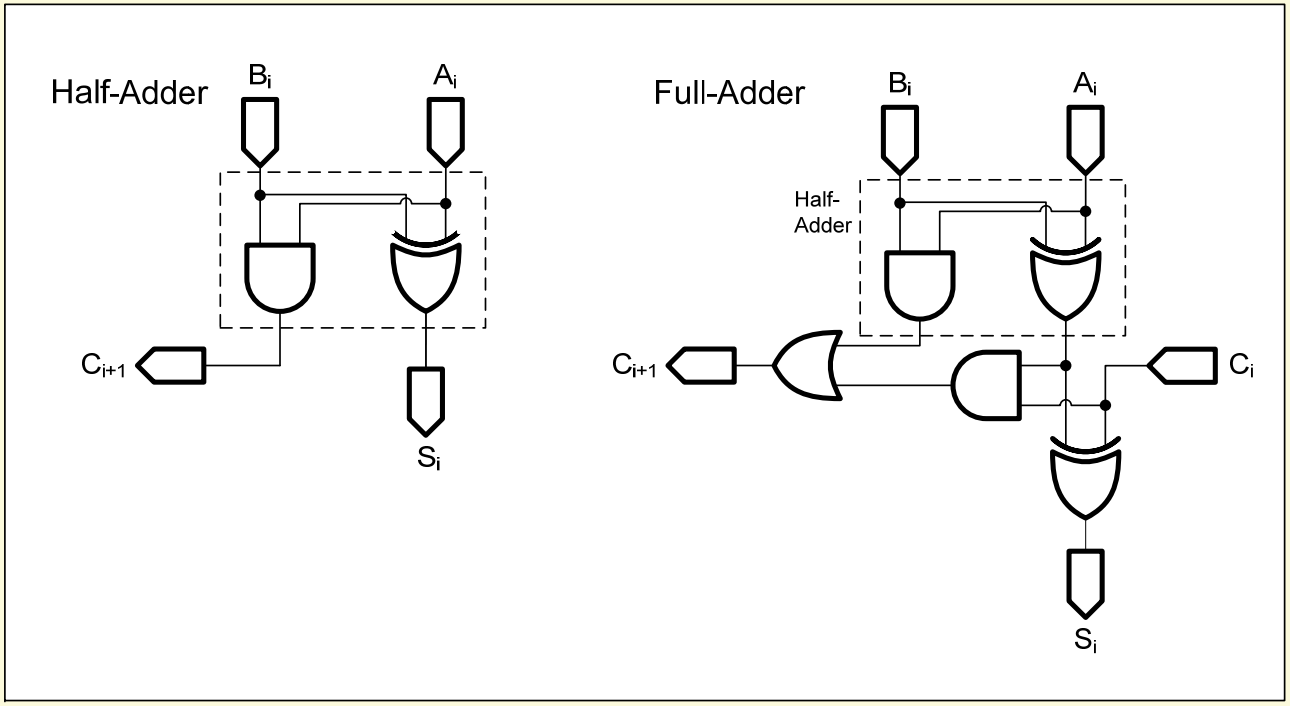
ou

$C(i+1) = C(i) \cdot (B(i) \oplus A(i)) + B(i) \cdot A(i)$

$S(i) = C(i) \oplus B(i) \oplus A(i)$

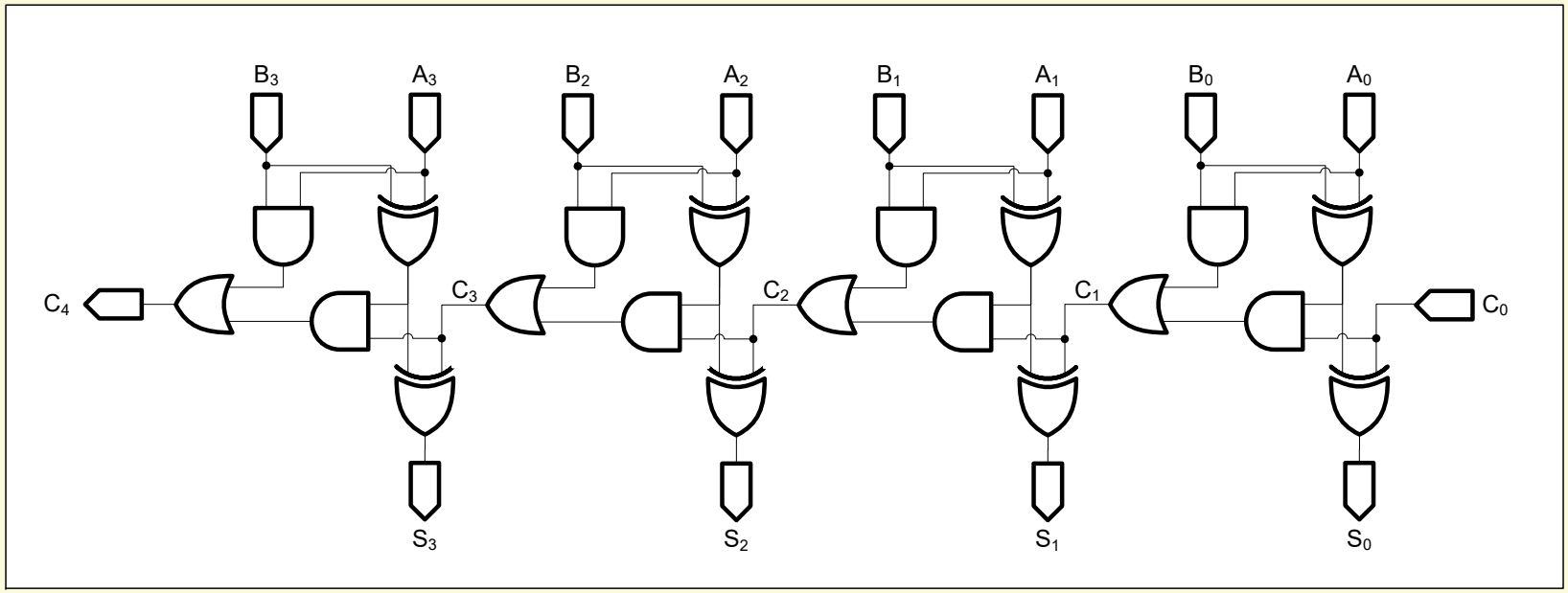
… additionneur 1 bit

- Schéma d'un additionneur complet 1 bit (Full-Adder) et du demi-additionneur (Half-Adder)



Additionneur 4 bits …

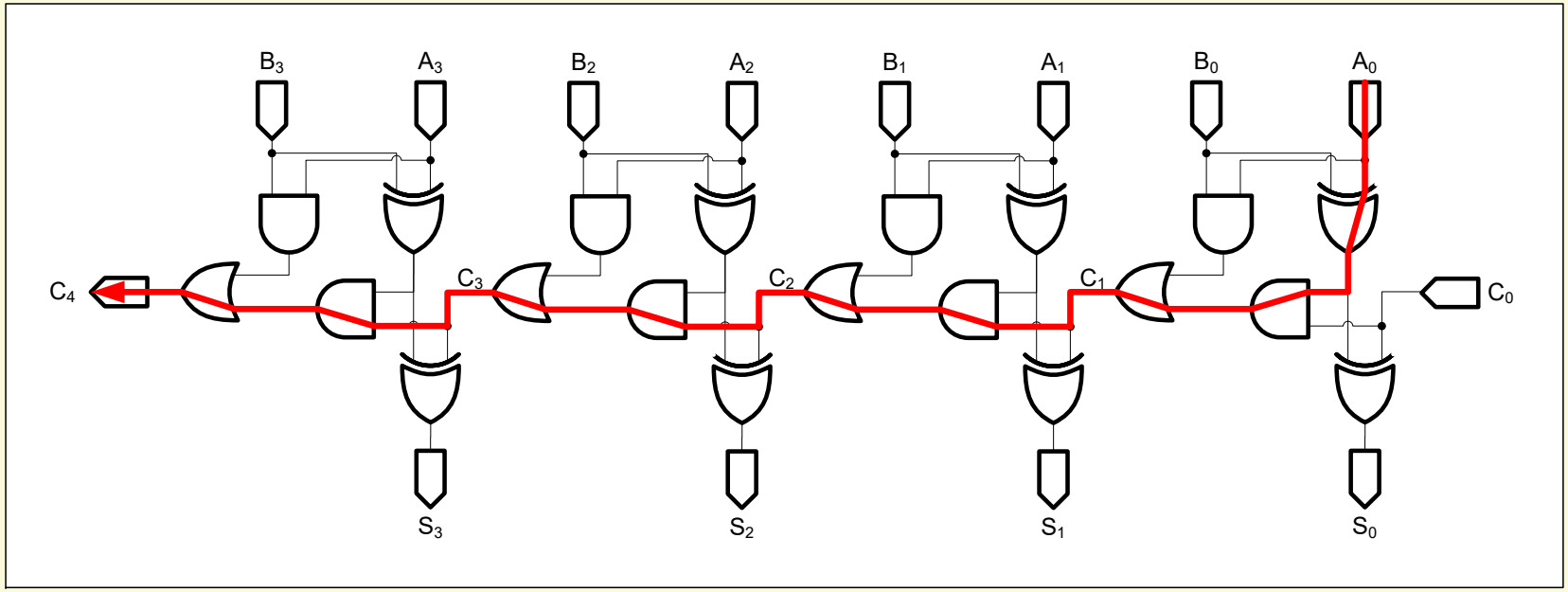
- Schéma additionneur 4 bits avec 4 Full-Adder :



Référence: Présentation additionneur de Yann Thoma

… additionneur 4 bits

- Chemin critique additionneur 4 bits



Référence: Présentation additionneur de Yann Thoma

Additionneur & VHDL

- Description VHDL :
  - ne pas utiliser les équations logiques !
  - utiliser l'opérateur ' + '

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;  --operations arithmétiques

entity add4 is
   port(nbr_a_i : in  std_logic_vector(3 downto 0);
        nbr_b_i : in  std_logic_vector(3 downto 0);
        cin_i   : in  std_logic;
        somme_o : out std_logic_vector(3 downto 0);
        cout_o  : out std_Logic;
        ovr_o       : out std_logic  );
end Add4 ;
```

Additionneur & VHDL

```
architecture flot_don of add4 is
  …
begin

  --Syntaxe directe : FONCTIONNE PAS
  --   Addition pas définie pour type logique !!

  somme <= nbr_a_i + nbr_b_i + cin_i;

  cout_o <= ??

end flot_don;
```

Voir présentation et laboratoire sur le paquetage : *Numeric_std*

Opérations et dépassement

Détection des cas de dépassement:

- Représentation des nombres non signés:
  - Addition => dépassement indiqué par "Carry"
  - Soustraction => dépassement indiqué par "Borrow"

    Borrow = Emprunt = not Carry
- Représentation des nombres signés en C2:
  - Addition => dépassement indiqué par "Overflow"
  - Soustraction => dépassement indiqué par "Overflow"

    Overflow = Cn xor Cn-1

Dépassement de capacité: résumé

- Carry
  - dépassement de capacité pour les additions de nombres non signés
  - généralement abrégé: C
- Borrow
  - dépassement de capacité pour les soustractions de nombres non signés
  - Borrow = not Carry
- Overflow
  - Dépassement de capacité pour les additions et soustractions de nombres signés en notation C2
  - généralement abrégé: V

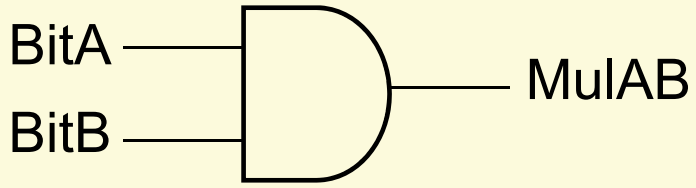
http://en.wikipedia.org/wiki/Carry_flag
http://en.wikipedia.org/wiki/Overflow_flag

Multiplication en binaire …

- Multiplication binaire :
  - MulAB = BitA x BitB correspond à la fonction logique ET!

BitA	BitB	MulAB
0	0	0
0	1	0
1	0	0
1	1	1

Il s'agit de la table de vérité de la porte ET



… multiplication en binaire …

- Multiplication par une puissance de 2 :
  - il s'agit d'un simple décalage du nombre
- Multiplication par une constante

  décomposer la multiplication en addition de puissance de 2
  - NbrA x 10 = NbrA * "1010"
  - NbrA x 10 = NbrA * "1000" + NbrA * "10"

Exercices III

1. Réaliser un circuit qui réalise le calcul suivant: r = 5 * na + 6, nbr signé en C2
   - na nombre signé de 4 bits
   - déterminer le nombre de bits du résultat r

Méthodologie de conception des systèmes combinatoires

- Analyse du système
  - Identification des entrées & sorties (symbole, noms signaux)
- Décomposition en sous-systèmes
  - Schéma bloc, table de fonctionnement (optimisation), variantes
- Spécification claire de chaque bloc
  - symbole CEI, TDV, description VHDL, texte, …
- Simplification des équations (Karnaugh)
- Description du comportement (schéma, VHDL)
- Simulation du comportement
- Intégration
- Montage et test final

Description VHDL et simulation ....

- Pour un sous-bloc:
  - Décrire en VHDL synthétisable
  - Simulation VHDL du sous-bloc
  - Synthétiser chaque sous-bloc
    - vérifier que la description est synthétisable
    - vérifier la quantité de logique obtenue
- Description structurelle du système complet
- Simulation du système complet (niveau VHDL)

… description VHDL et simulation

- Synthèse du système complet
  - Vérification de la quantité de logique obtenue
- Intégration dans un circuit
  - Placement routage du système complet
  - Vérification statique des temps de propagation
- Simulation après placement-routage
  - Temps de propagation visible
- Programmation et test du circuit

Exercices

- Série " Fonctions standard combinatoires"
  - Exercice nº 60 à 64