Unité:Base de systèmes logiques (SysLog1)

Les fonctions standards combinatoires

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

E. Messerli & C. Bornand
& collègues de l'institut REDS

Juillet 2022

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

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

Décodeur X/Y

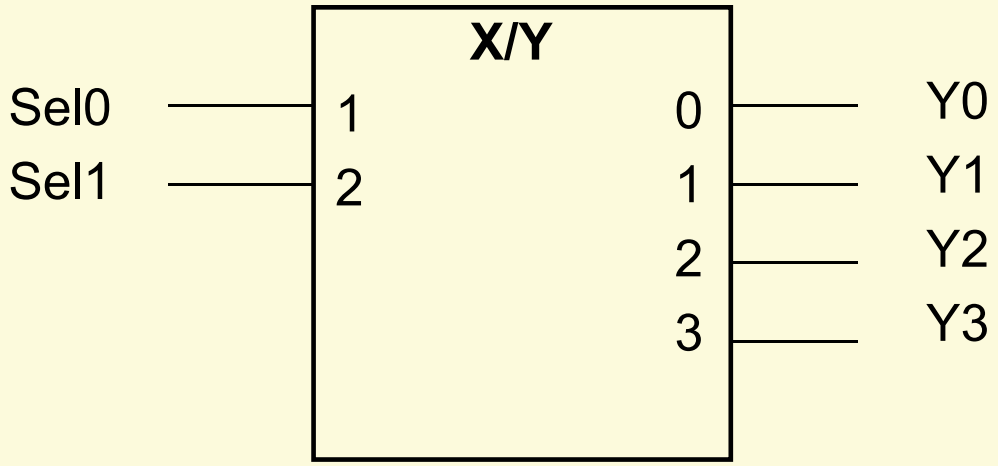
- But : activer la sortie dont on donne le numéro sous forme binaire (entier non signé)

  => 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.

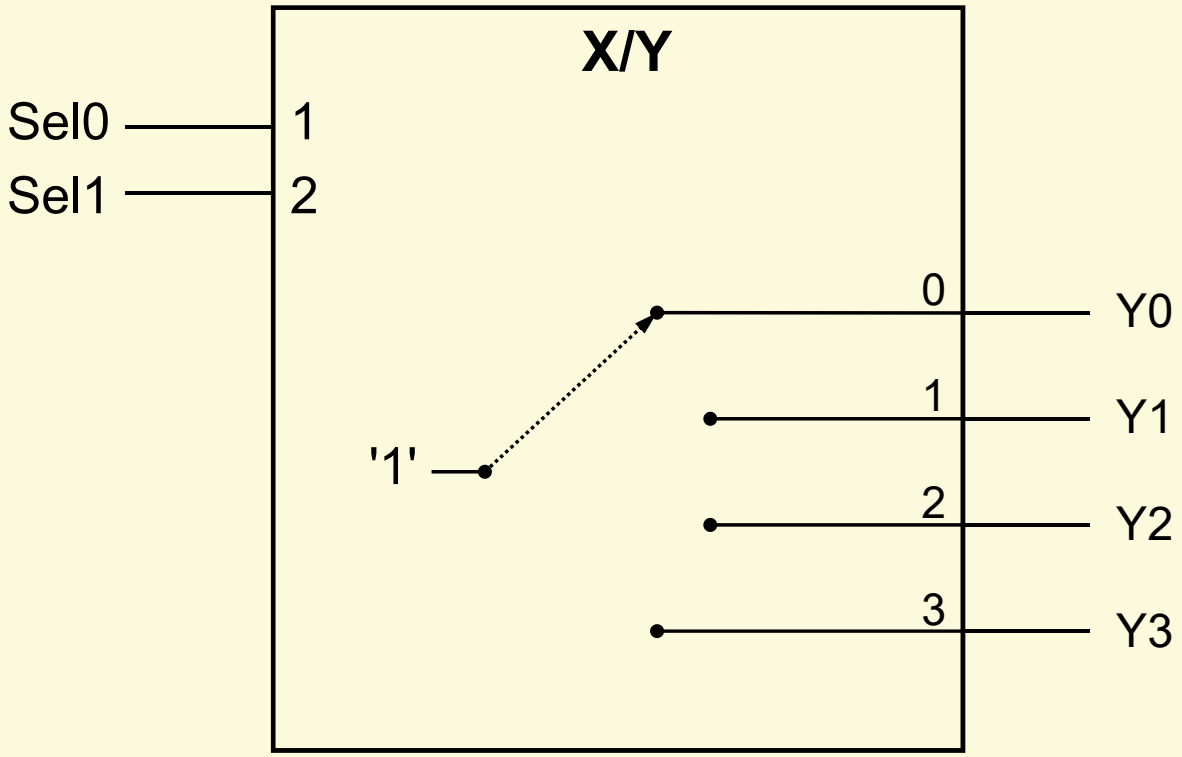
Décodeur 2 à 4, symbole



Symbole IEEE/CEI **RECOMMANDÉ**

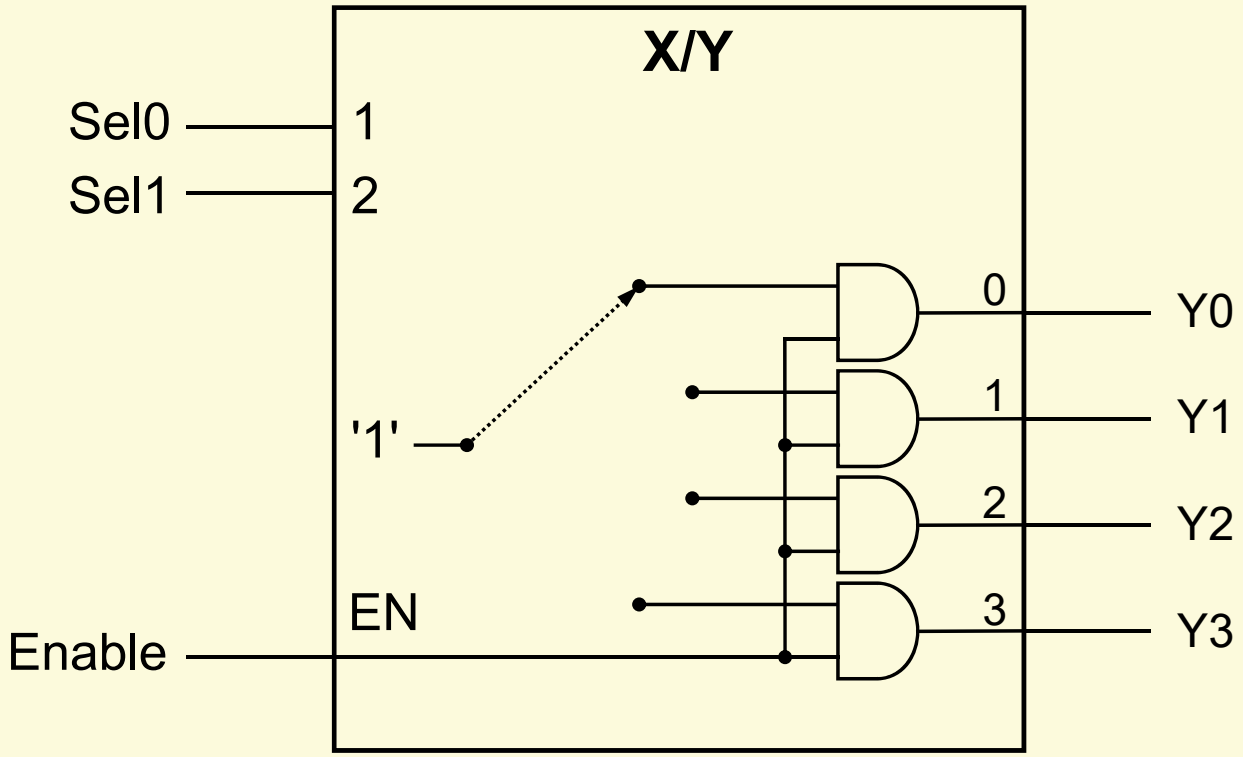
Décodeur 2 à 4

- Fonctionnement de principe



Décodeur 2 à 4 avec EN

- Fonctionnement de principe



Décodeur 2 à 4 (TDV, équations, schéma)

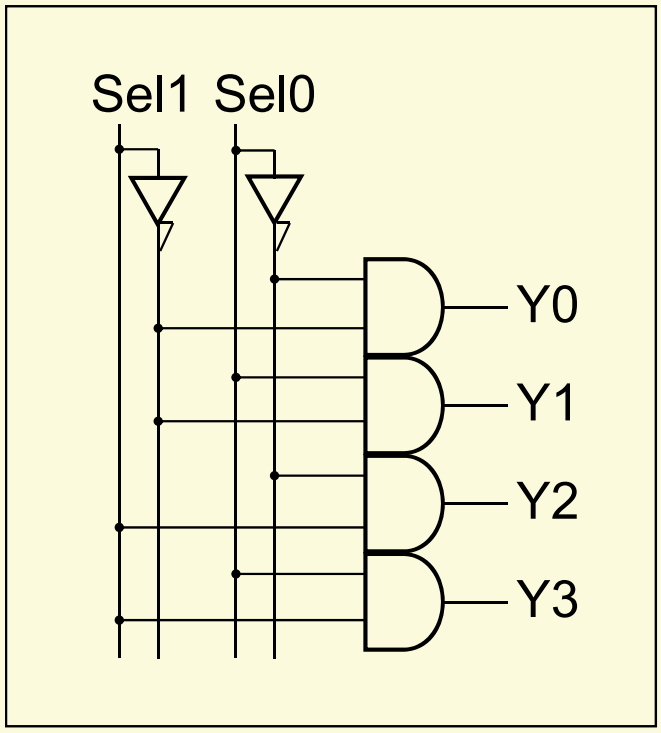
- Génère tous les mintermes de Sel1 et Sel0

Table de vérité

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

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

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



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

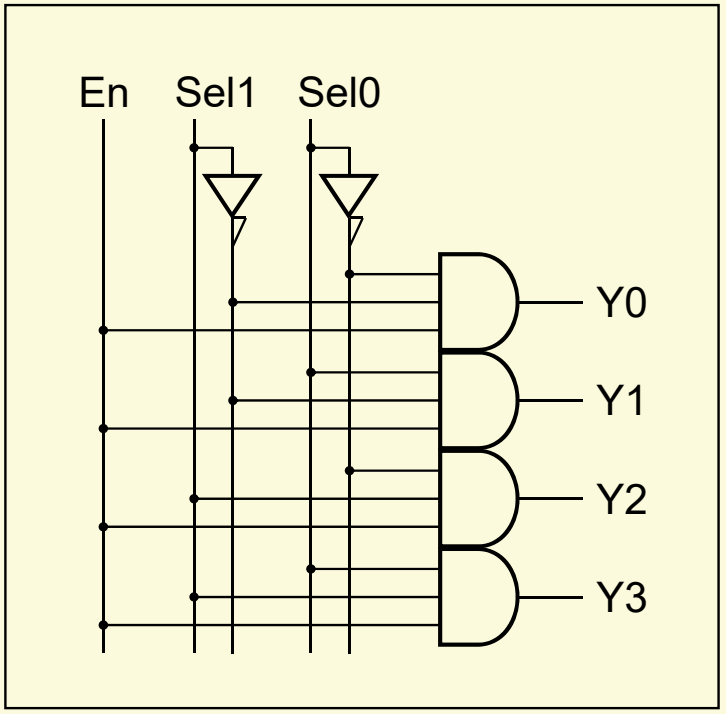
- Génère tous les mintermes de Sel1 et Sel0

Table de vérité

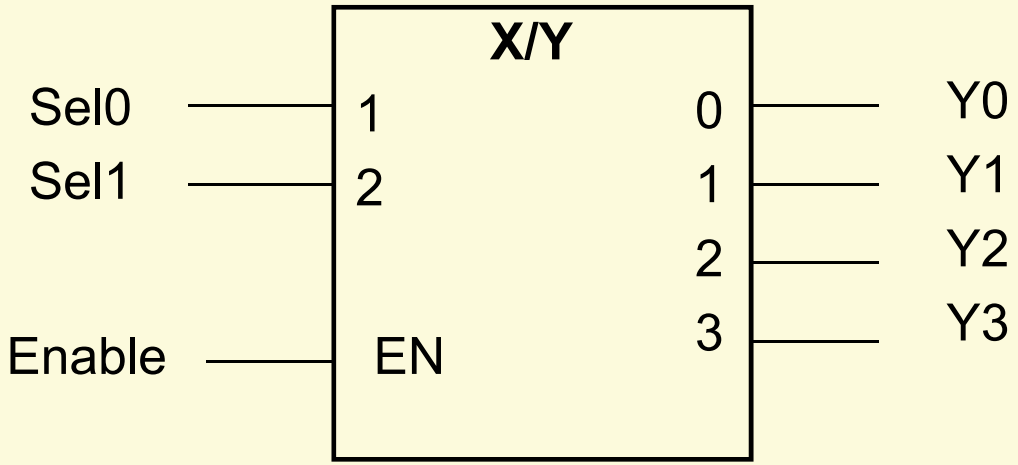
En	Sel1	Sel0	Y3	Y2	Y1	Y0
'0'	x	x	'0'	'0'	'0'	'0'
'1'	'0'	'0'	'0'	'0'	'0'	'1'
'1'	'0'	'1'	'0'	'0'	'1'	'0'
'1'	'1'	'0'	'0'	'1'	'0'	'0'
'1'	'1'	'1'	'1'	'0'	'0'	'0'

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

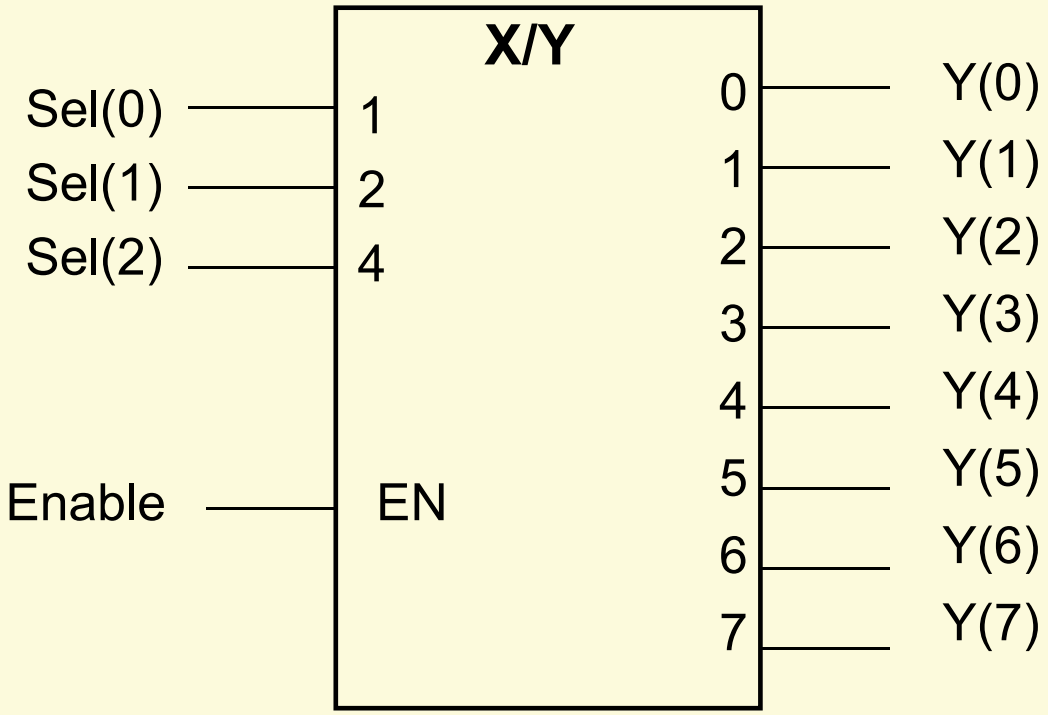
$Y2 = En \cdot (Sel1 \cdot \overline{Sel0})$ $Y3 = En \cdot (Sel1 \cdot Sel0)$



Décodeur 2 à 4 avec EN, symbole



Décodeur 3 à 8, symbole CEI

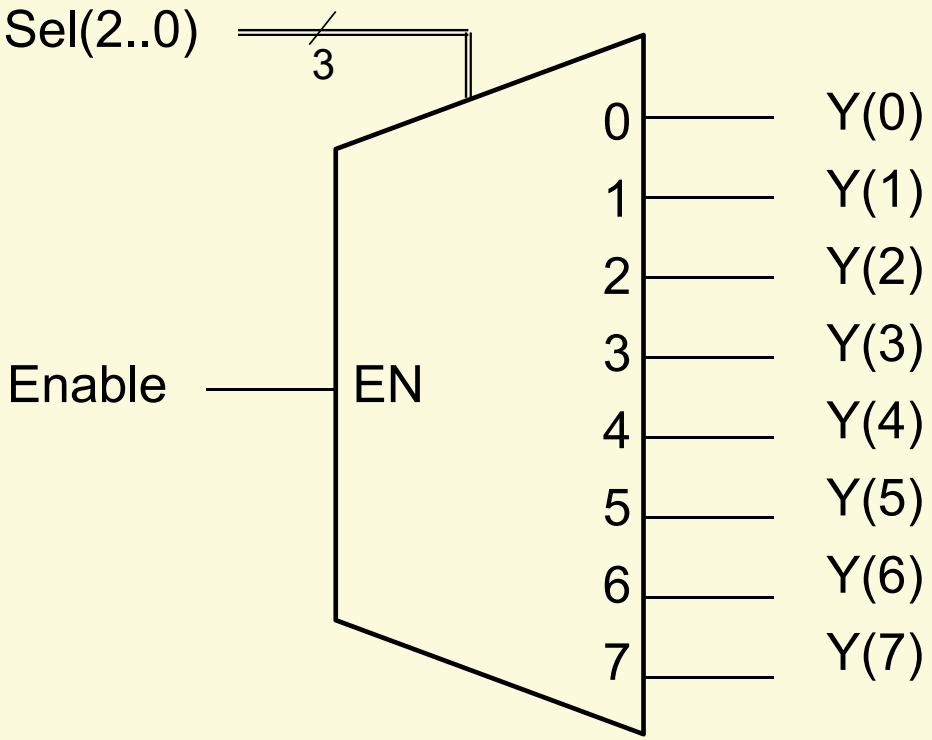


Symbole IEEE/CEI **RECOMMANDÉ**

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'

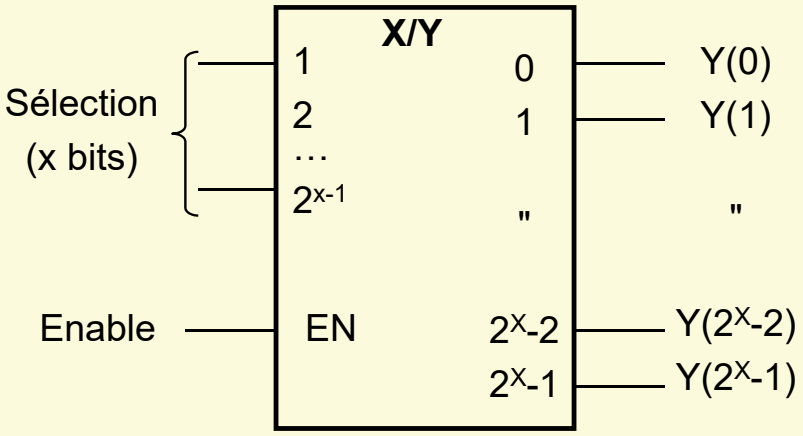
Décodeur 3 à 8, symbole US



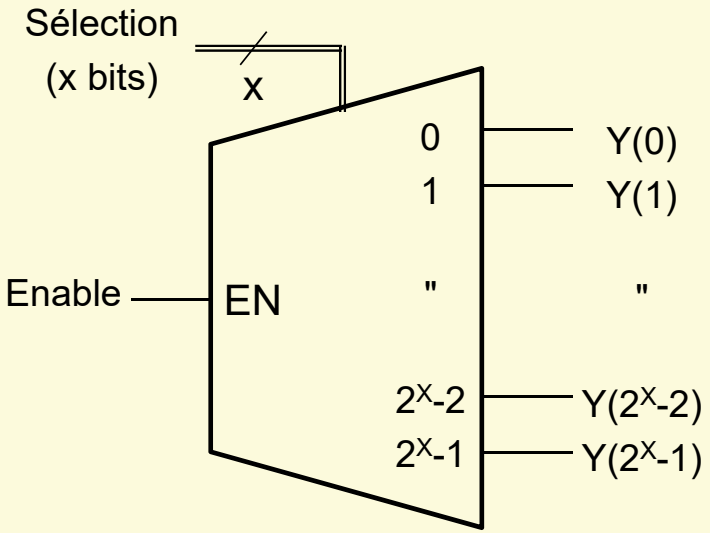
Symbole US **ACCEPTÉ**

Décodeur X/Y (symbole)

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



Symbole IEEE/CEI RECOMMANDÉ

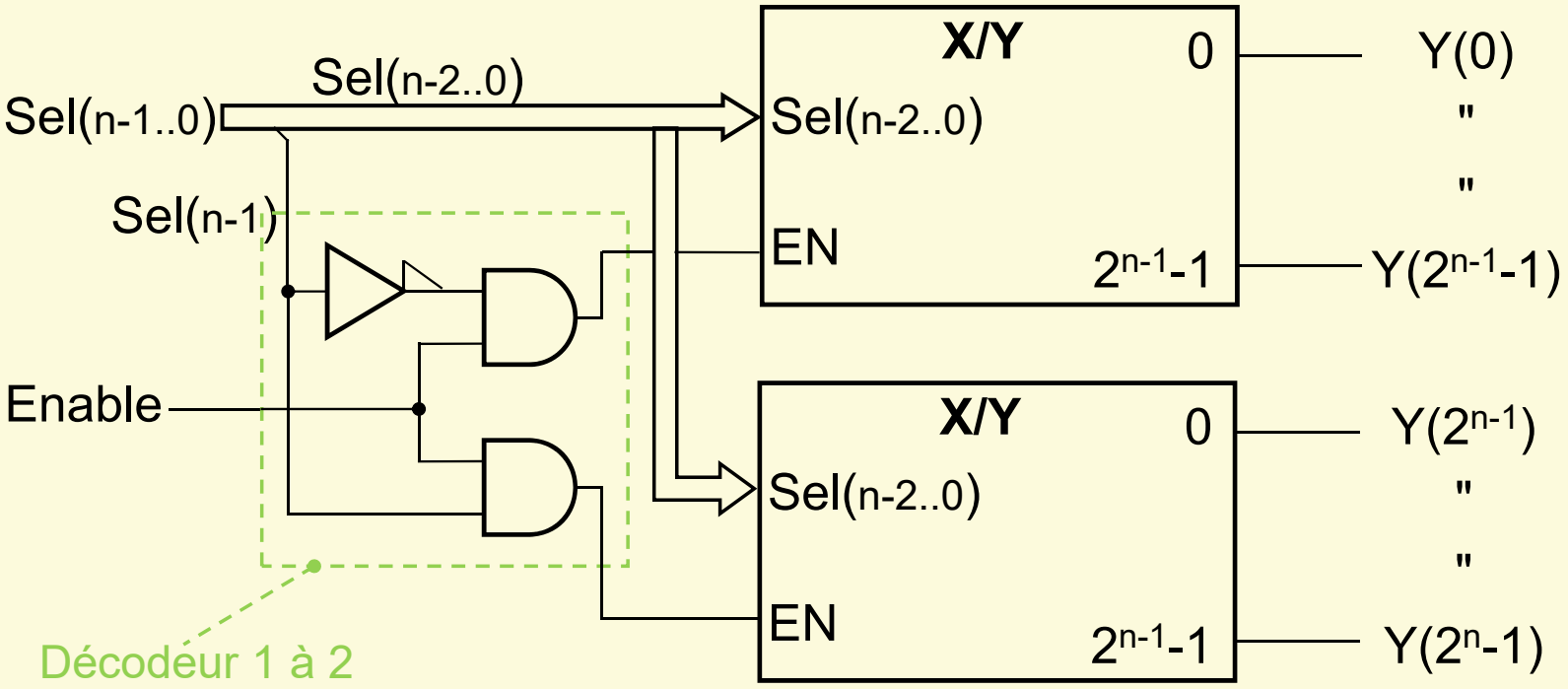


Symbole US ACCEPTÉ

utilisation de symbole explicite !

Décodeur, schéma de décomposition

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



Décodage: génération de mintermes

- Chaque sortie d'un décodeur n'est active que pour une et une seule valeur du code binaire d'entrée, donc pour une et une seule combinaison des bits d'entrée

  $\Rightarrow$ chaque sortie correspond à un minterme

- Avec un décodeur à N entrées et une porte OU à 2**N-1 entrées au maximum, on peut implémenter n'importe quelle fonction combinatoire à N entrées, sous la forme d'une somme de mintermes.

Décodeur en générateur de fonction ...

TDV da la fonction F :

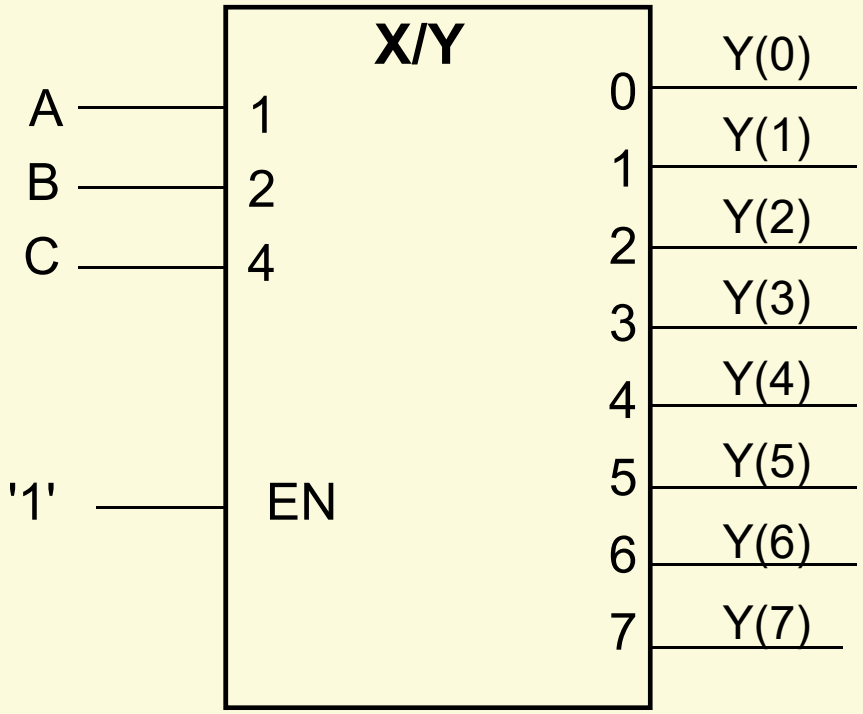
C	B	A	F
'0'	'0'	'0'	'0'
'0'	'0'	'1'	'1'
'0'	'1'	'0'	'0'
'0'	'1'	'1'	'0'
'1'	'0'	'0'	'1'
'1'	'0'	'1'	'0'
'1'	'1'	'0'	'1'
'1'	'1'	'1'	'0'

Equation de F en somme de mintermes :

$$F(C, B, A) = \Sigma\ 1, 4, 6$$

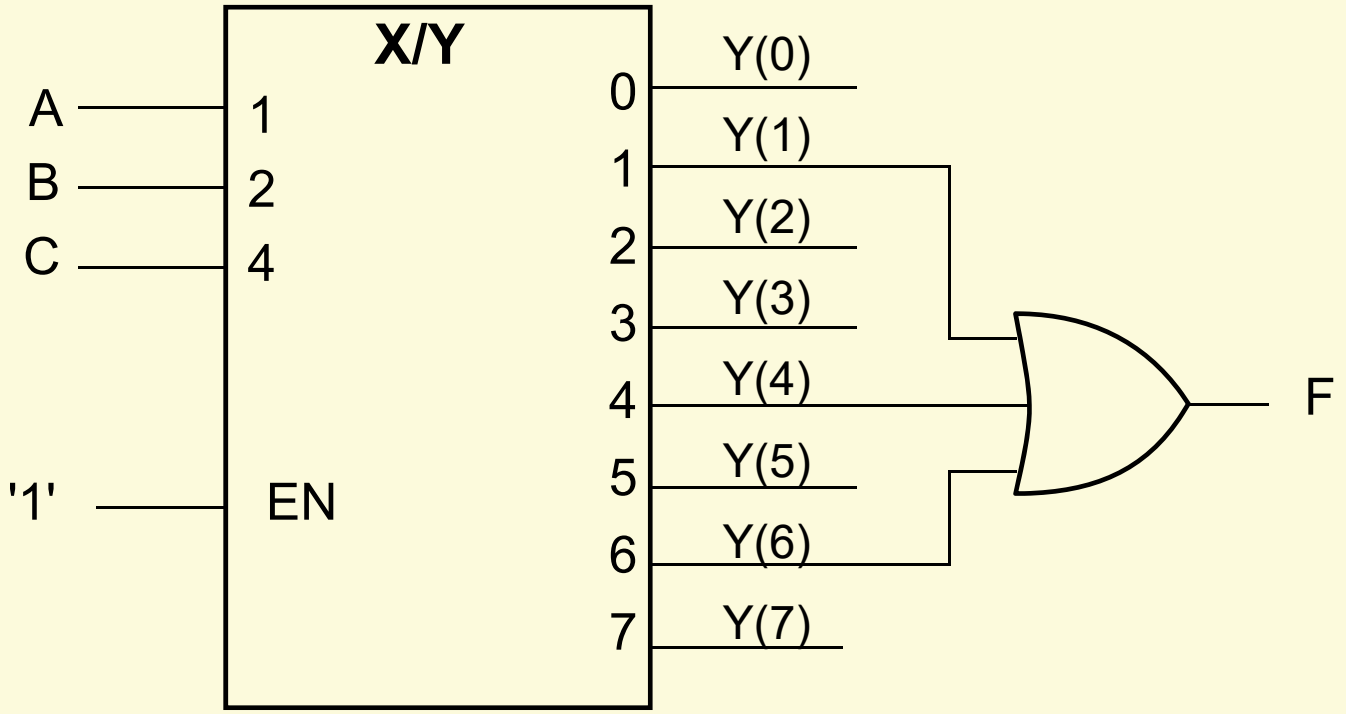
… décodeur en générateur de fonction …

Schéma F(C, B, A) = Σ 1, 4, 6



… décodeur en générateur de fonction

Schéma F(C, B, A) = Σ 1, 4, 6



Exercices décodeurs

- Série " Fonctions standard combinatoires"
  - Exercices nº 40, 41, 42, 43
  - Exercice nº 50 points a) et b)
  - Exercice nº 52 point f)

HEIG VD

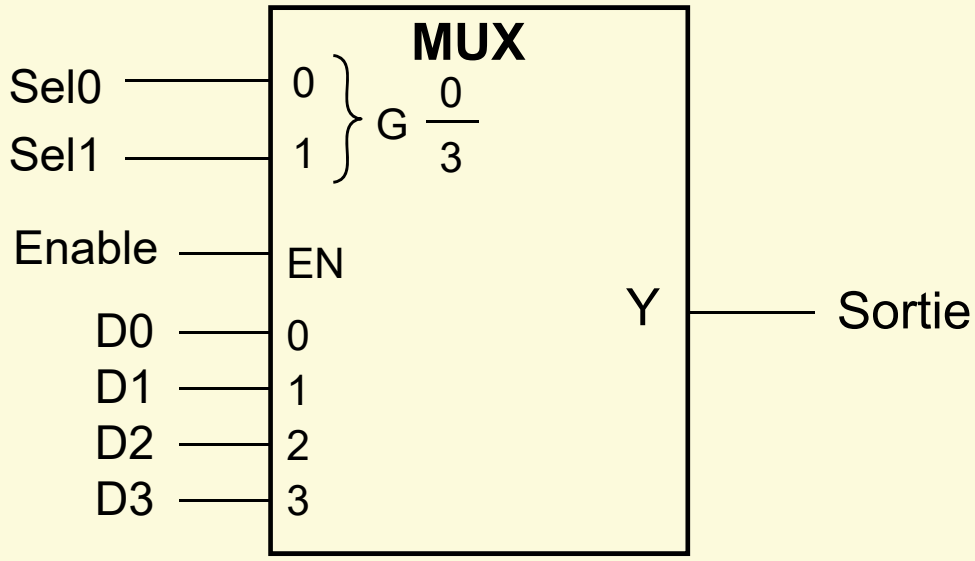
Dia volontairement laisser vide

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

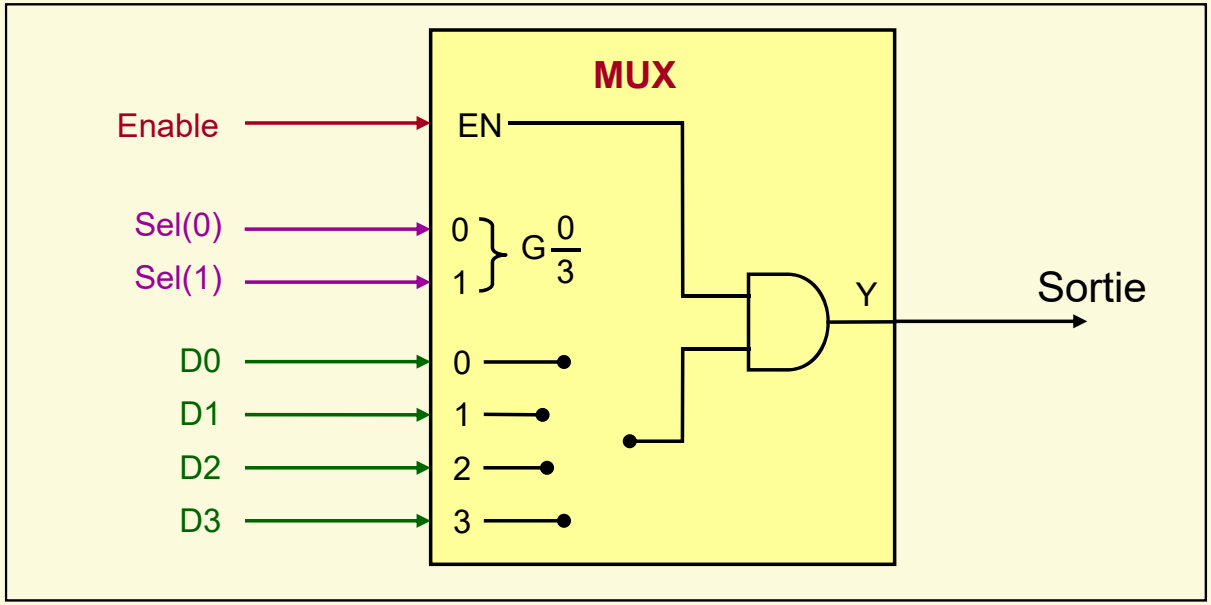
Multiplexeur 4 à 1, symbole

- Symbole IEEE/CEI



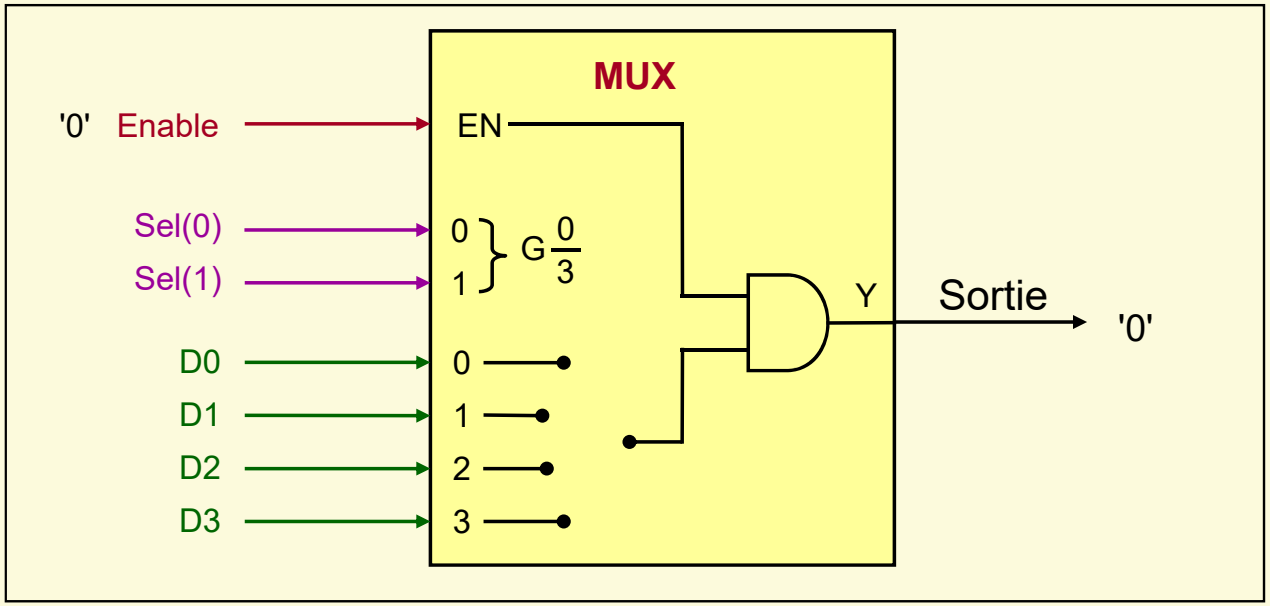
Exemple fonctionnel du multiplexeur

- 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)



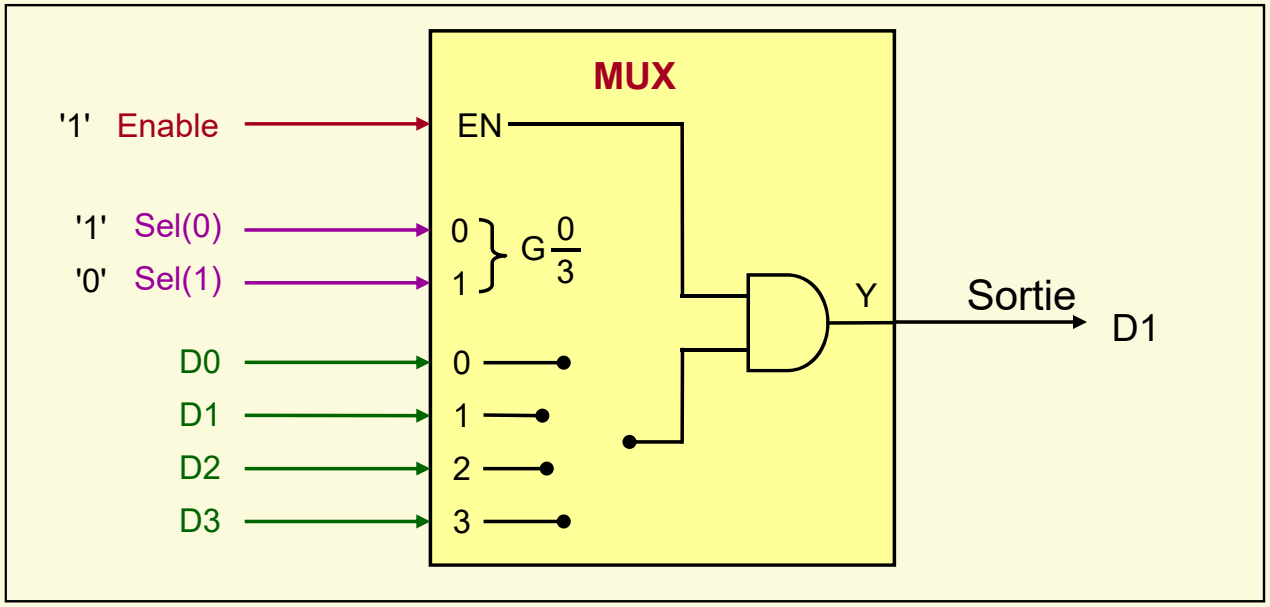
Fonctionnement multiplexeur 4 à 1 ...

- Signal *Enable* inactif :



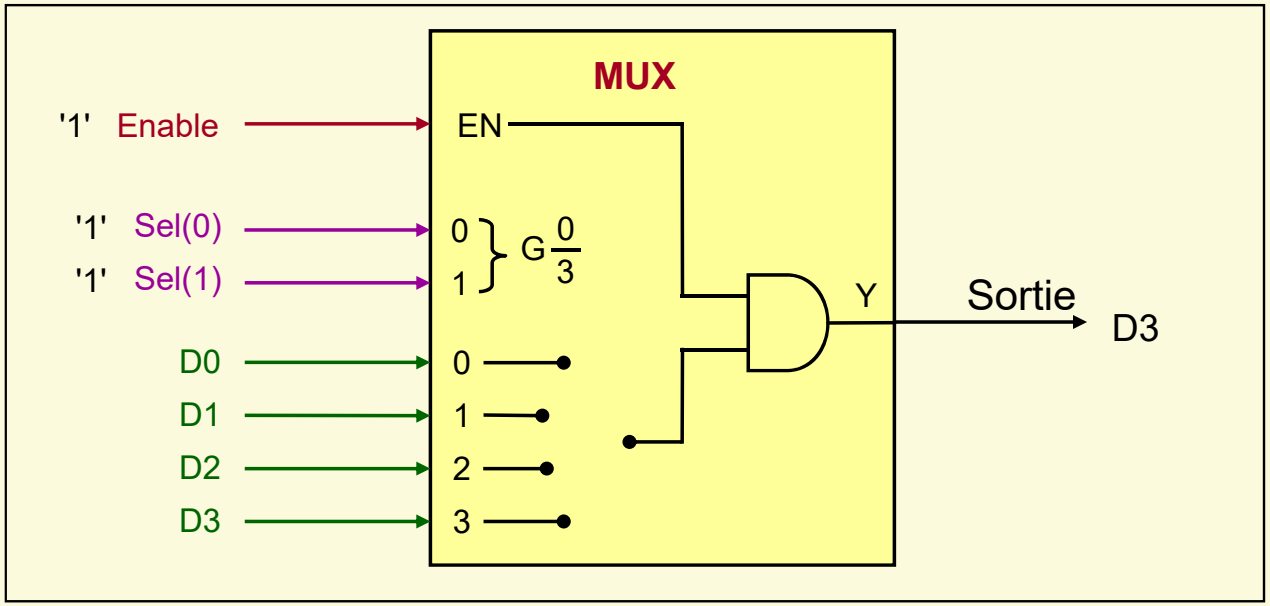
… fonctionnement multiplexeur 4 à 1 …

- Signal *Enable* actif et *Sel(1..0)* = "01" :



… fonctionnement multiplexeur 4 à 1

- Signal *Enable* actif, Sel(1..0) = "11" :



Multiplexeur 2 to 1 (schéma)

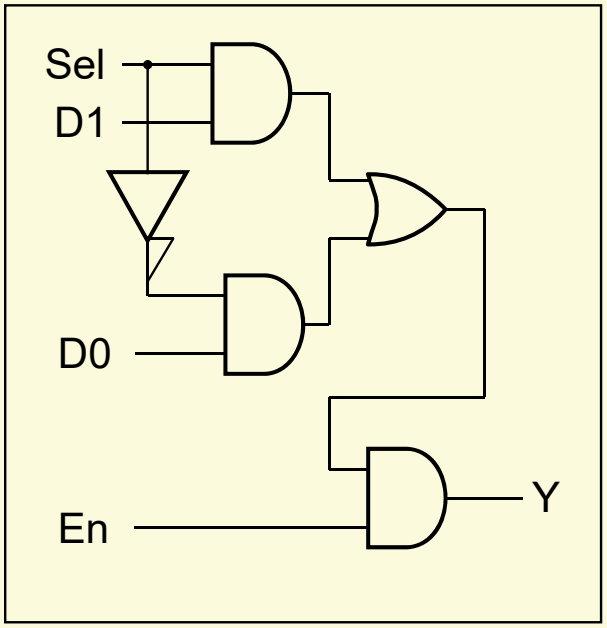
- Système combinatoire à 4 entrées (En, Sel, D1 et D0) et une sortie (Y)

Table de vérité

En	Sel	D1	D0	Y
'0'	x	x	x	'0'
'1'	'0'	'0'	'0'	'0'
'1'	'0'	'0'	'1'	'1'
'1'	'0'	'1'	'0'	'0'
'1'	'0'	'1'	'1'	'1'
'1'	'1'	'0'	'0'	'0'
'1'	'1'	'0'	'1'	'0'
'1'	'1'	'1'	'0'	'1'
'1'	'1'	'1'	'1'	'1'

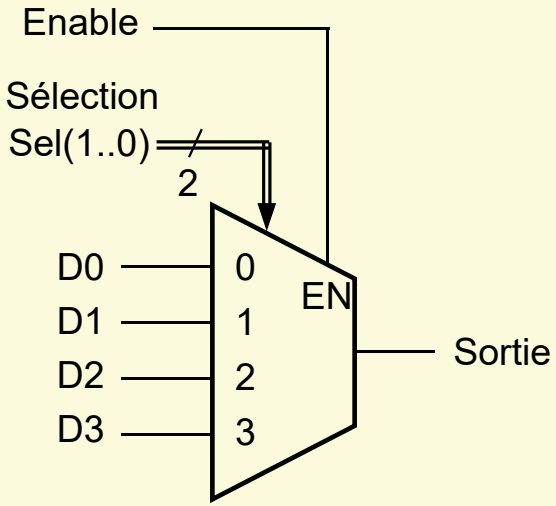
D1 D0 \ En Sel	00	01	11	10
00	'0'	'0'	'0'	'0'
01	'0'	'0'	'0'	'1'
11	'0'	'0'	'1'	'1'
10	'0'	'0'	'1'	'0'

$$Y = EN \bullet (\overline{Sel} \bullet D0 + Sel\ D1)$$

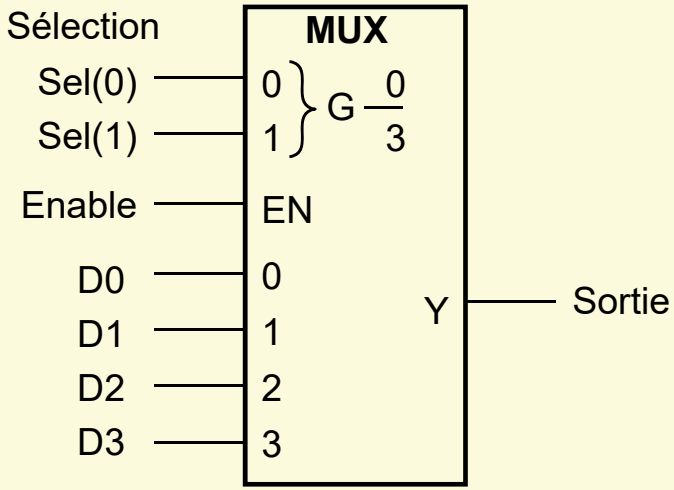


Multiplexeur 4 à 1, symbole

- Symboles multiplexeur 4 à 1:



Symbole US **ACCEPTÉ**



Symbole IEEE/CEI **RECOMMANDÉ**

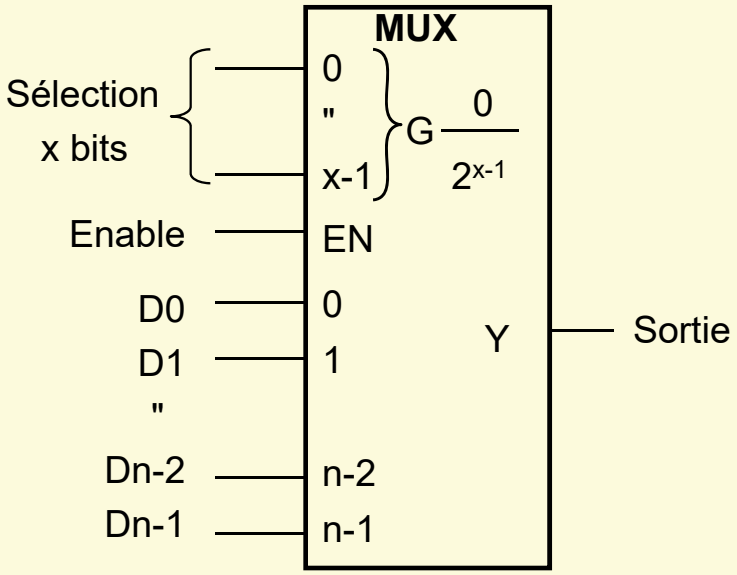
Multiplexeur 4 à 1, table de vérité

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

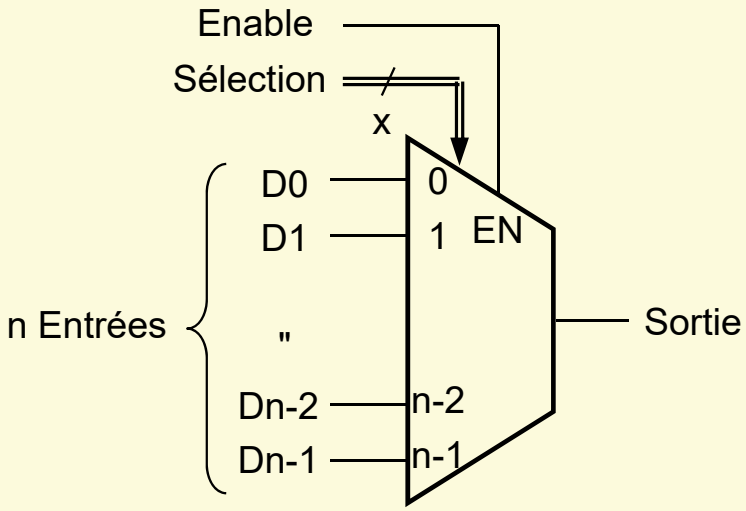
EN	Sel(1)	Sel(0)	Y
0	x	x	0
1	0	0	D0
1	0	1	D1
1	1	0	D2
1	1	1	D3

Multiplexeur n à 1, symboles

- Symbole multiplexeur n à 1 avec n = 2^x :



Symbole IEEE/CEI **RECOMMANDÉ**

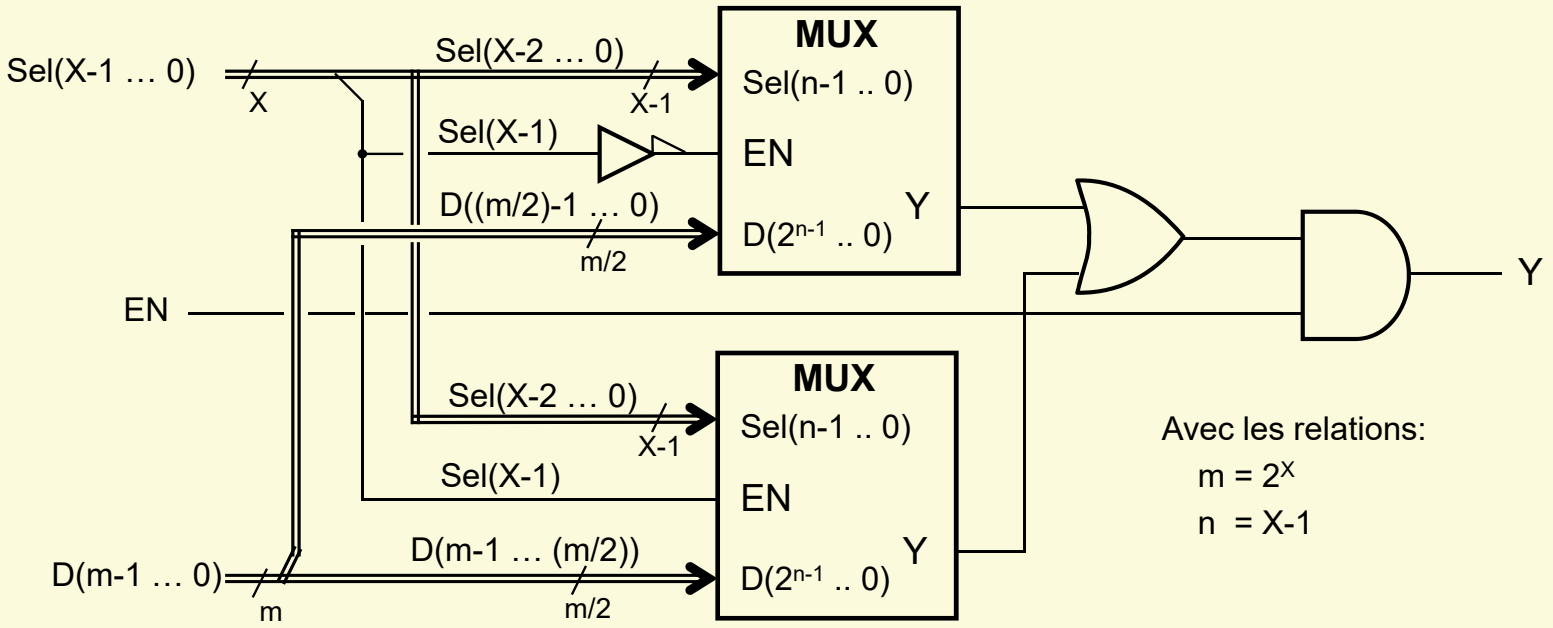


Symbole US **ACCEPTÉ**

utilisation de symbole explicite !

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



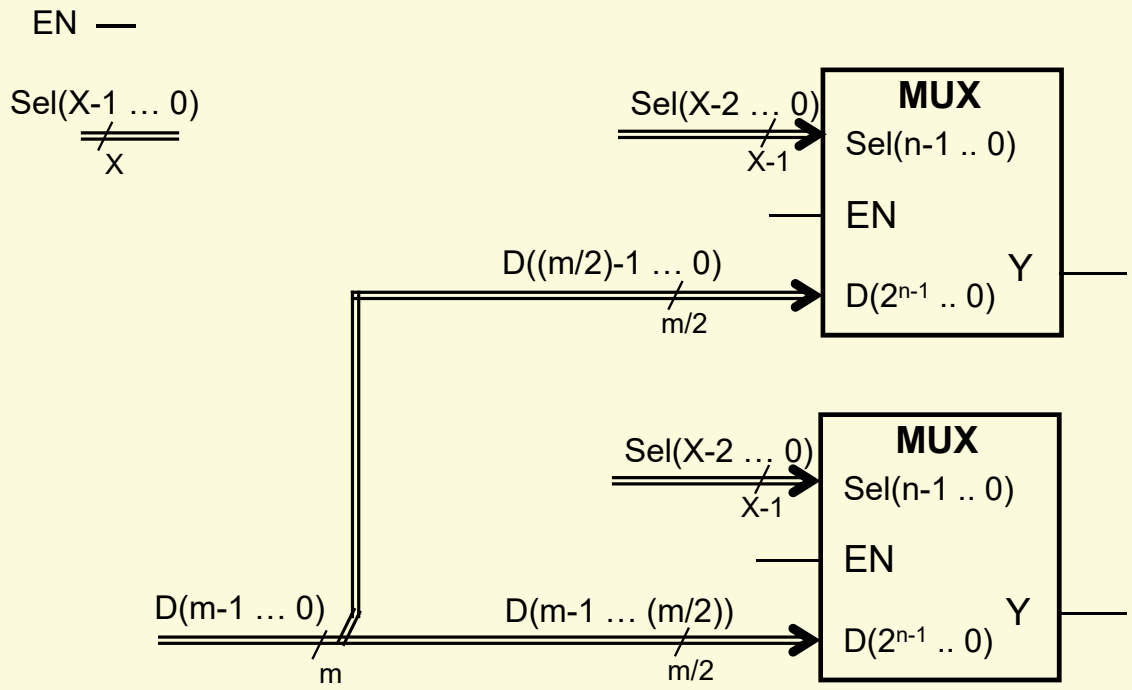
Exercice

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

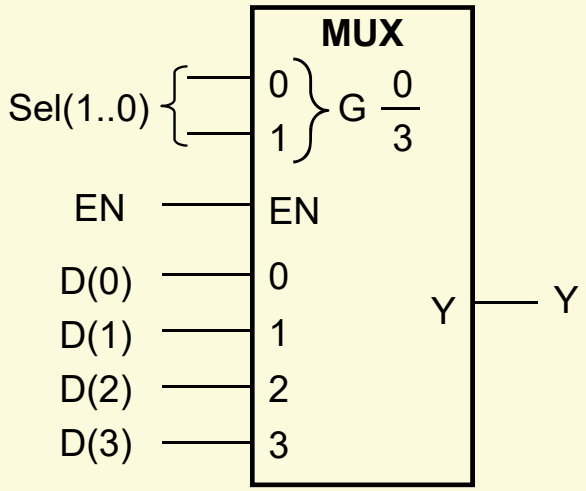
Avec :

$m = 2^X$

$n = X-1$



Le MUX en générateur de fonction ...



Equation logique :

Y = EN **and**

(**not** Sel(1) **and** **not** Sel(0) **and** D(0)

or **not** Sel(1) **and** Sel(0) **and** D(1)

or Sel(1) **and** **not** Sel(0) **and** D(2)

or Sel(1) **and** Sel(0) **and** D(3))

… MUX en générateur de fonction …

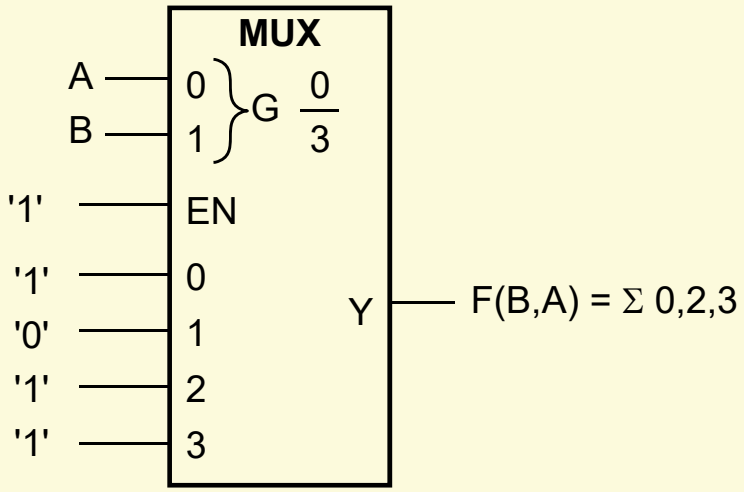
- Table de vérité Mux 4à1 compacte

F(B,A) = Σ 0,2,3

'1' EN	B Sel(1)	A Sel(0)	F Y	
0	X	X	0	**pas utilisé**
1	0	0	D(0)	**'1'**
1	0	1	D(1)	**'0'**
1	1	0	D(2)	**'1'**
1	1	1	D(3)	**'1'**

… MUX en générateur de fonction …

Voici le schéma logique :



… MUX en générateur de fonction

- On peut donc décomposer le MUX en
  - Un décodeur
  - Un aiguillage
- On peut donc générer n'importe quelle fonction de 2 variables (Sel(1), Sel(0)) en plaçant sur les entrées D(3)..D(0) les valeurs logiques souhaitées
- Cette technique est utilisée dans certains circuits programmables FPGA (exemple : FPGA de Xilinx)

Exe: multiplexeur en générateur de fonction ...

C Sel(2)	B Sel(1)	A Sel(0)	P Y
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

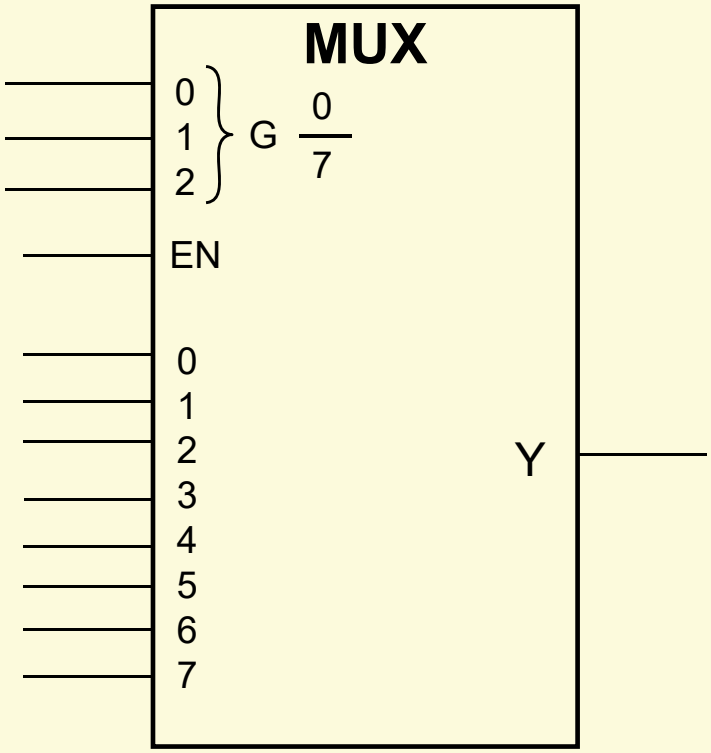
Exercice :

$P(C,B,A) = \Sigma\ 1, 2, 4, 7$

Donner le schéma logique de la fonction P en utilisant un multiplexeur 8 à 1?

… exe: multiplexeur en générateur de fonction

- Exercice P(C,B,A) = Σ 1, 2, 4, 7

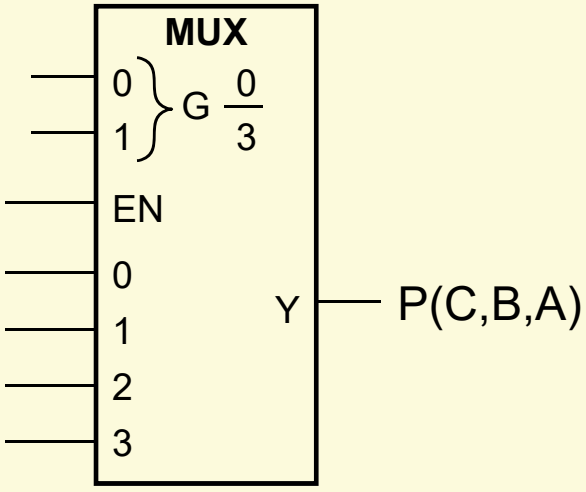


… exe: multiplexeur en générateur de fonction

- 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



Exercices

- Série "Fonctions standard combinatoires"
  - Exercices nº 44, 45, 46, 47, 48
  - Exercice nº 50 points c) et d)
  - Exercice nº 52 point e)

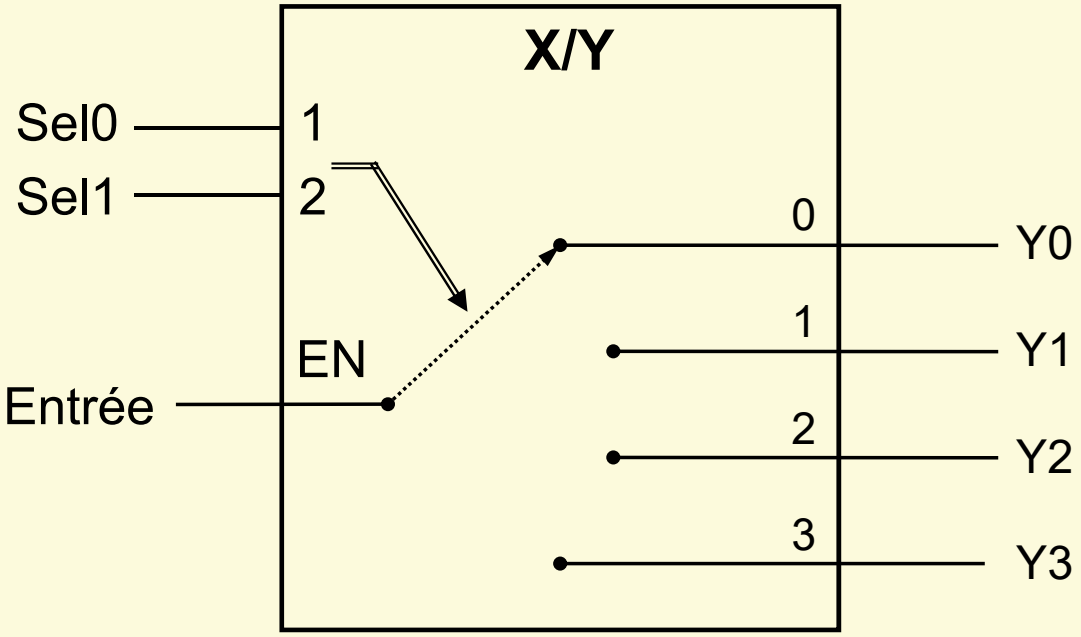
- Exercices Mux et Dec :
  - Exercices nº 49, 50, 51, 52, 53

Décodeur - Démultiplexeur

- Un décodeur peut aussi être utilisé comme **démultiplexeur**
- Dans le mode démultiplexeur
  - L'entrée « Enable » joue le rôle d'entrée de données
  - L'entrée de sélection indique la sortie sur laquelle l'entrée de données (EN) est aiguillée

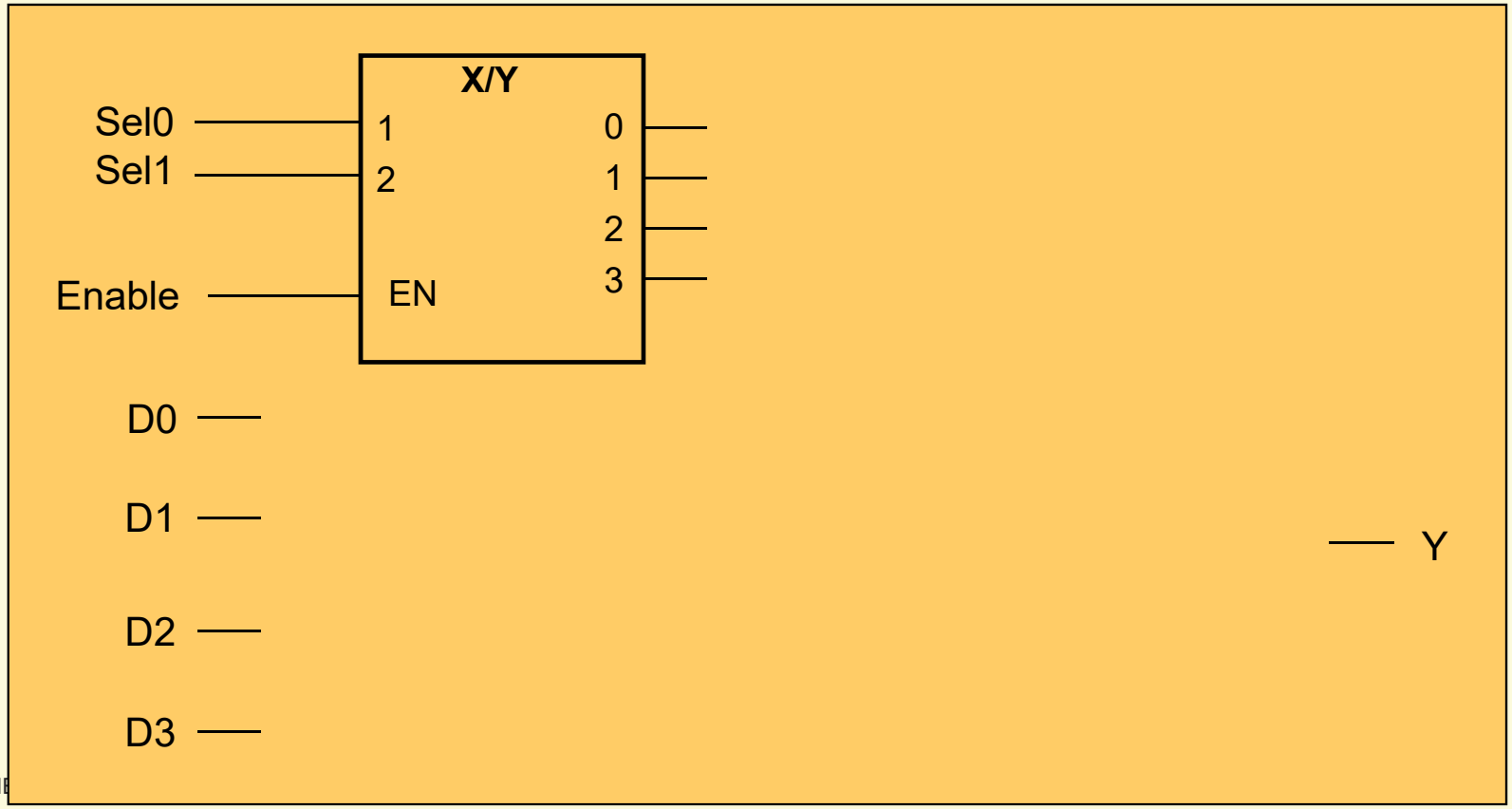
Démultiplexeur (schéma bloc)

- La sortie Y(i) copie l'entrée EN
- Les autres sorties sont inactives ('0')



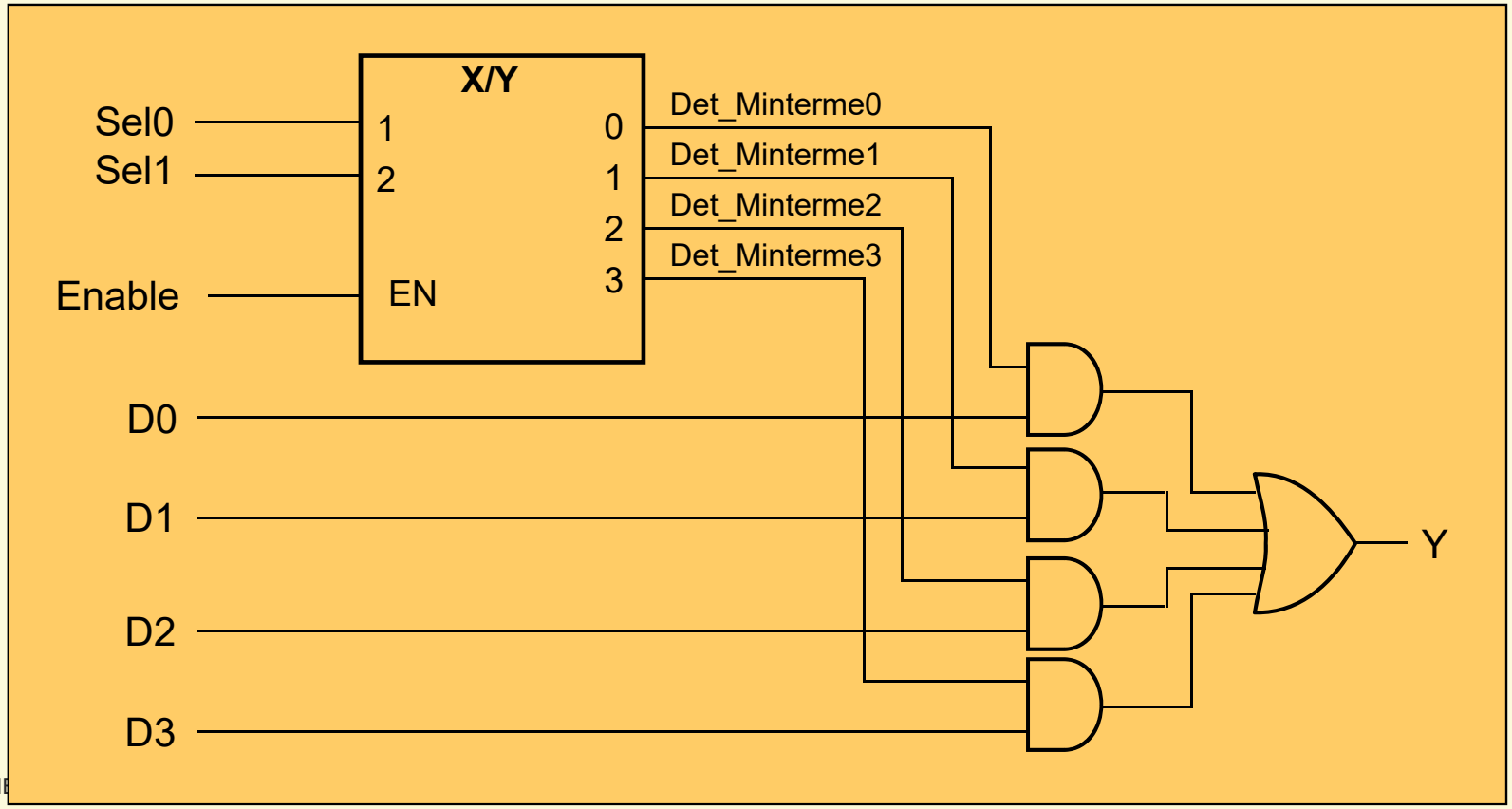
Multiplexeur et décodeur

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



Multiplexeur et décodeur

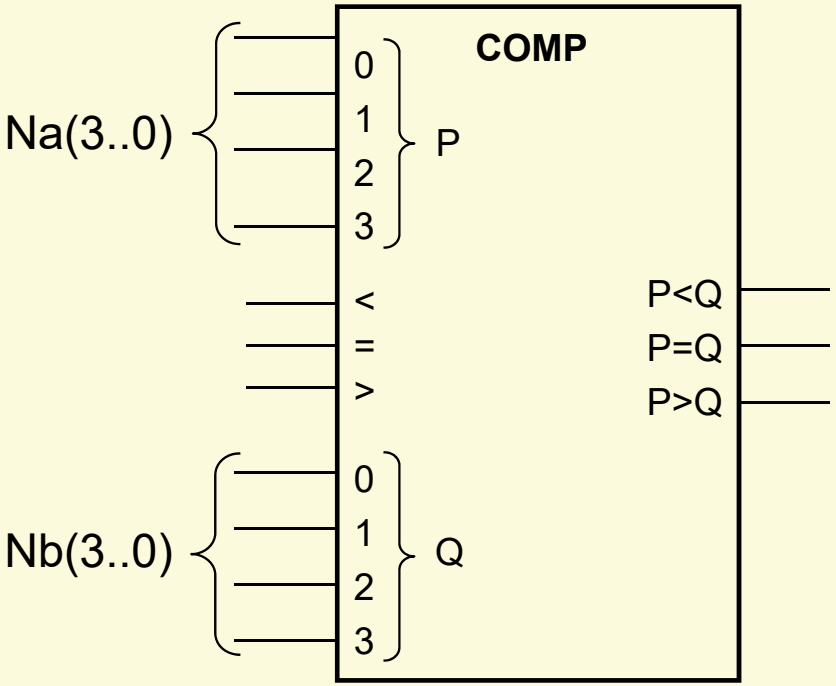
- Le multiplexeur peut être construit sur la base d'un décodeur

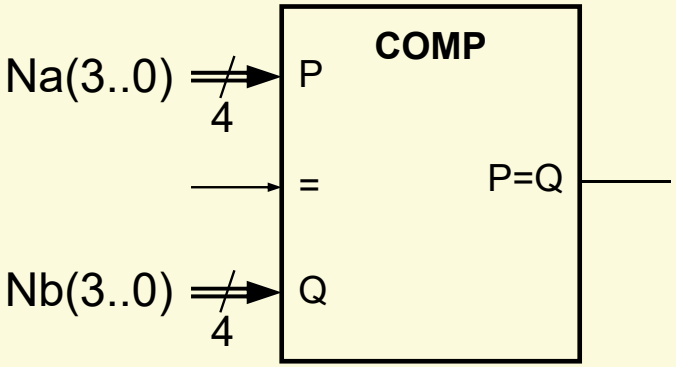


Comparateur

- 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**

Décomposition comparateur

- Tout comparateur à n bits peut se décomposer en n comparateurs 1 bit
- La décomposition peut être de type cascade ou parallèle
  - cascade : moins de matériel, plus lent
    => moins coûteux
  - parallèle : plus de matériel, plus rapide
    => plus performant

Comparateur 1 bit (schéma)

- Un comparateur 1 bit se réalise au moyen d'un ou-exclusif inversé (xnor)

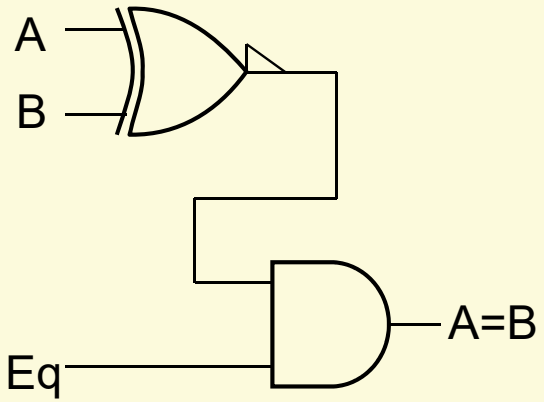
Table de vérité

Eq	B	A	A=B
'0'	'-'	'-'	'0'
'1'	'0'	'0'	'1'
'1'	'0'	'1'	'0'
'1'	'1'	'0'	'0'
'1'	'1'	'1'	'1'

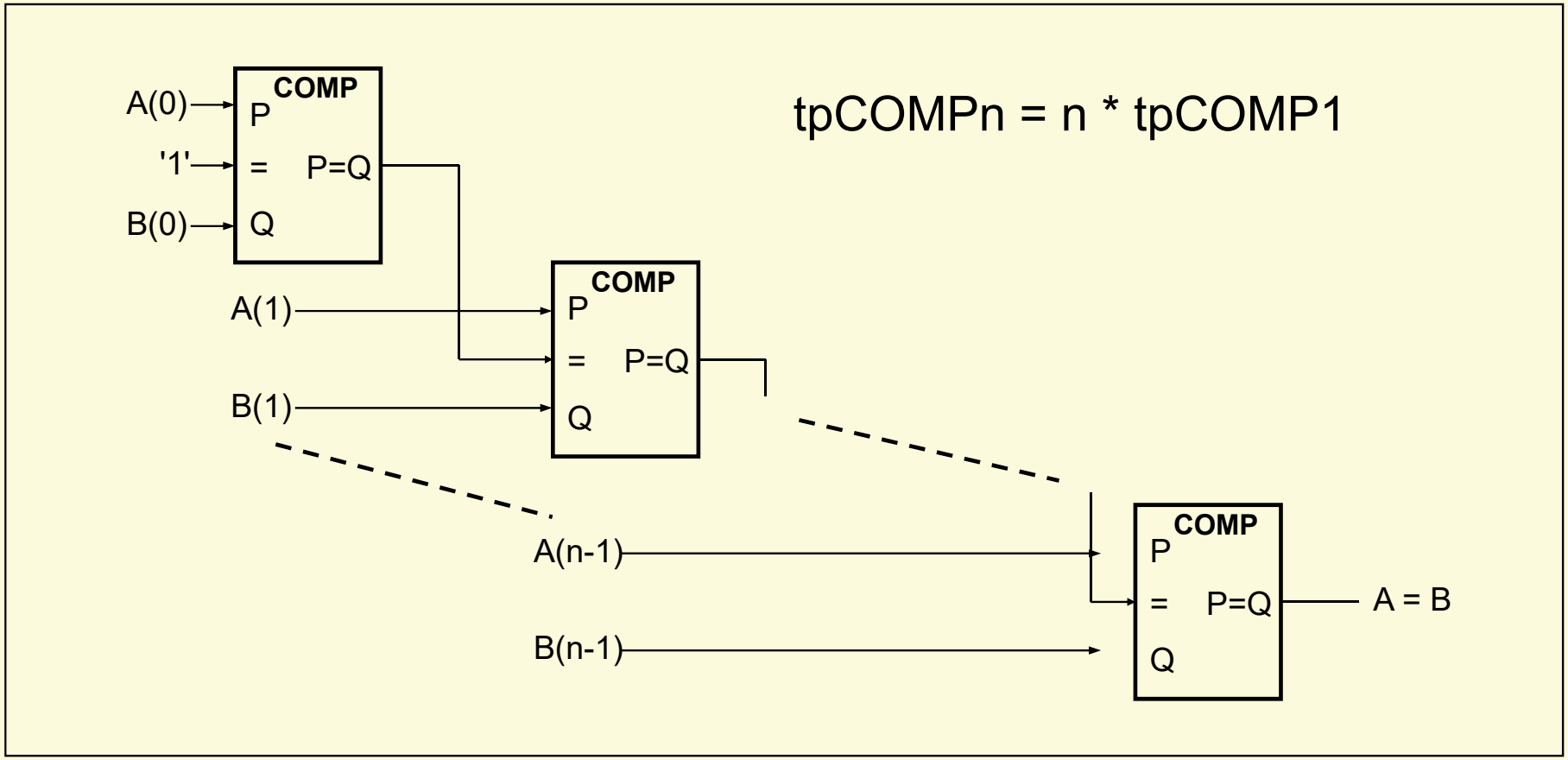
Eq \ B A	00	01	11	10
0	'0'	'0'	'0'	'0'
1	'1'	'0'	'1'	'0'

$A=B = (\overline{A} \cdot \overline{B} + A \cdot B) \cdot Eq$

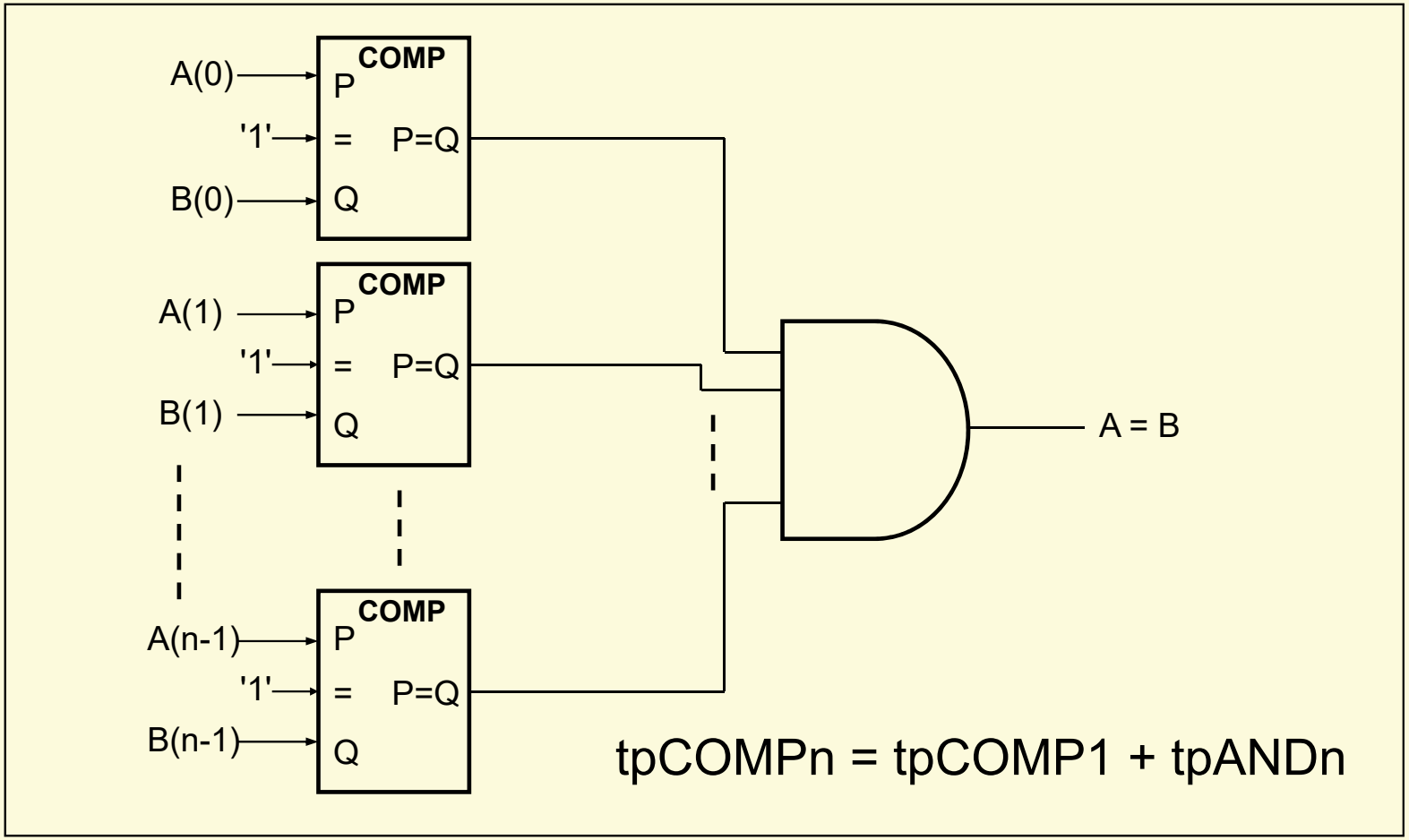
$= \overline{(A \oplus B)} \cdot Eq$



Décomposition en cascade



Décomposition en parallèle



Exercices I

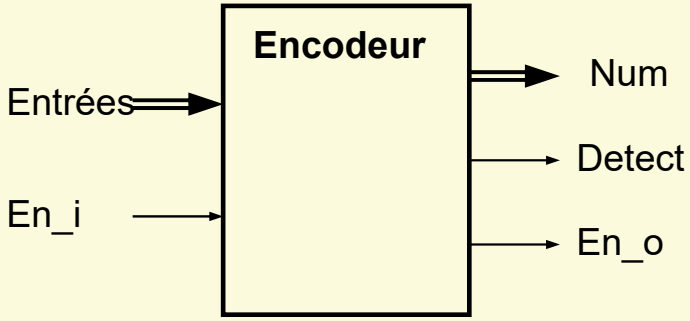
- Soit un comparateur fournissant l'égalité entre deux nombres de n bits.
  1. Donner l'équation logique de la sortie égalité (a_eq_b) pour des nombres de 5 bits.
     Indiquer le nombre total de termes de cette équation.
  2. Indiquer le nombre total de termes de l'équation si les nombres ont 10 bits.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

Encodeur de priorité

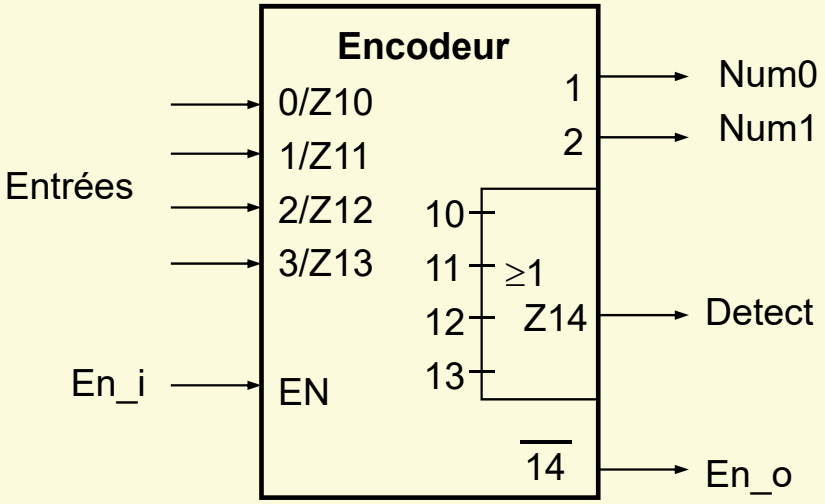
- But : déterminer le numéro de l ’entrée active ayant le degrés de priorité le plus élevé
- Ce module comporte souvent en plus une entrée d ’autorisation
- Ce module comporte une sortie indiquant qu’une des entrées au moins est active (Nécessaire pour identifier une action sur l ’entrée 0)

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 :

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

$$Num1 = In3 + In2$$

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

Additionneur

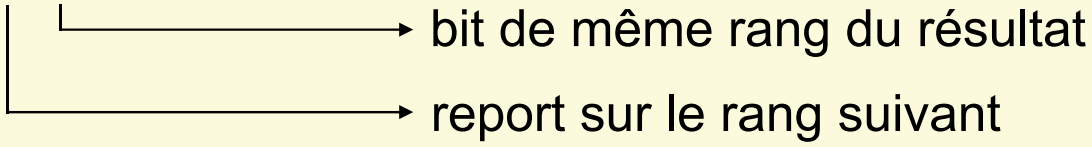
- But : réaliser l ’addition de deux nombres binaires
- Ce type de module se réalise fréquemment par une structure en chaîne; de ce fait, il comporte, en plus de l ’entrée des deux nombres, une entrée et une sortie pour les retenues

Addition…

- Pour effectuer une addition en binaire, on peut procéder comme en décimal : chiffre après chiffre
- Ainsi, l'addition de 2 nombres de n bits est une opération décomposable en n additions de 3 nombres de 1 bit
- Pourquoi 3 ?
- Commençons par le bit de poids faible (2^0)

...addition...

- L'addition chiffre à chiffre est plus simple en binaire qu'en décimal, il y a trois résultats possible :
  - 0+0 = 0
  - 0+1 = 1+0 = 1
  - 1+1 = 1 0



- Pour le bit de rang 1 et suivant: **3 bits d'entrée!**
  - deux entrées pour les bits des nombres à additionner et un report en provenance du rang précédent

...addition : cas général ...

- 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

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

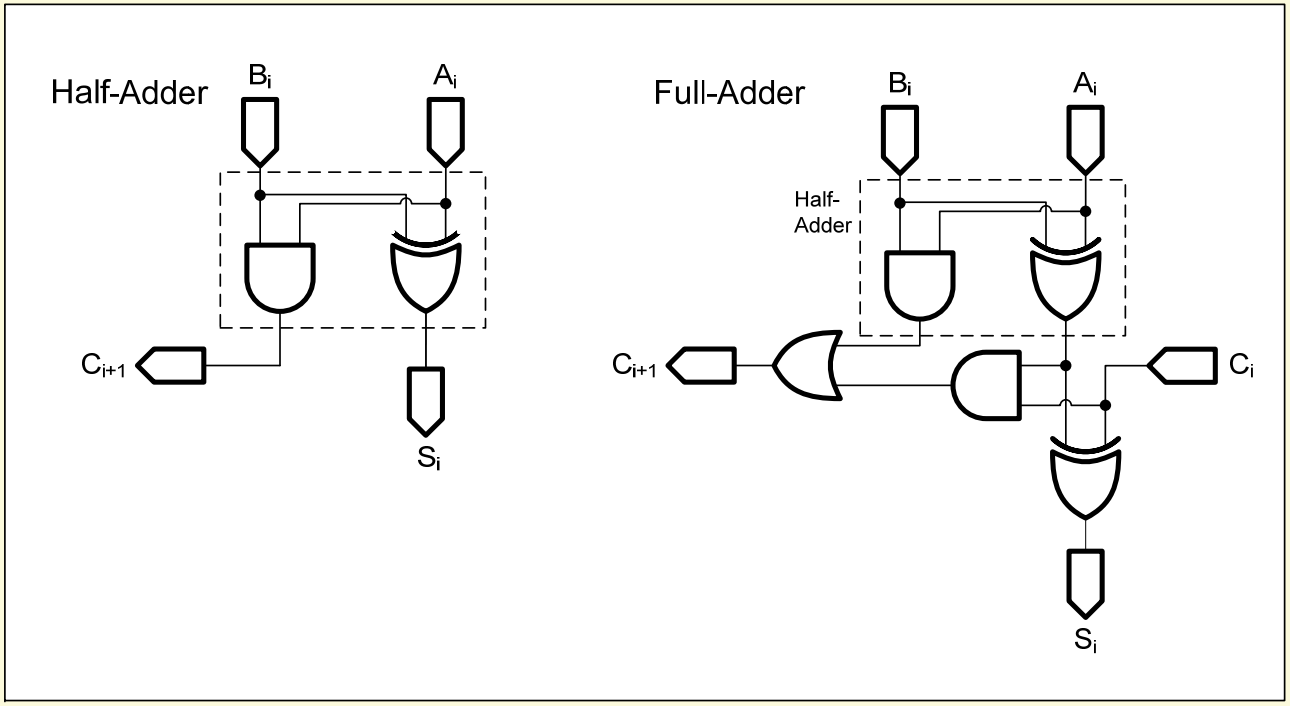
ou

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

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

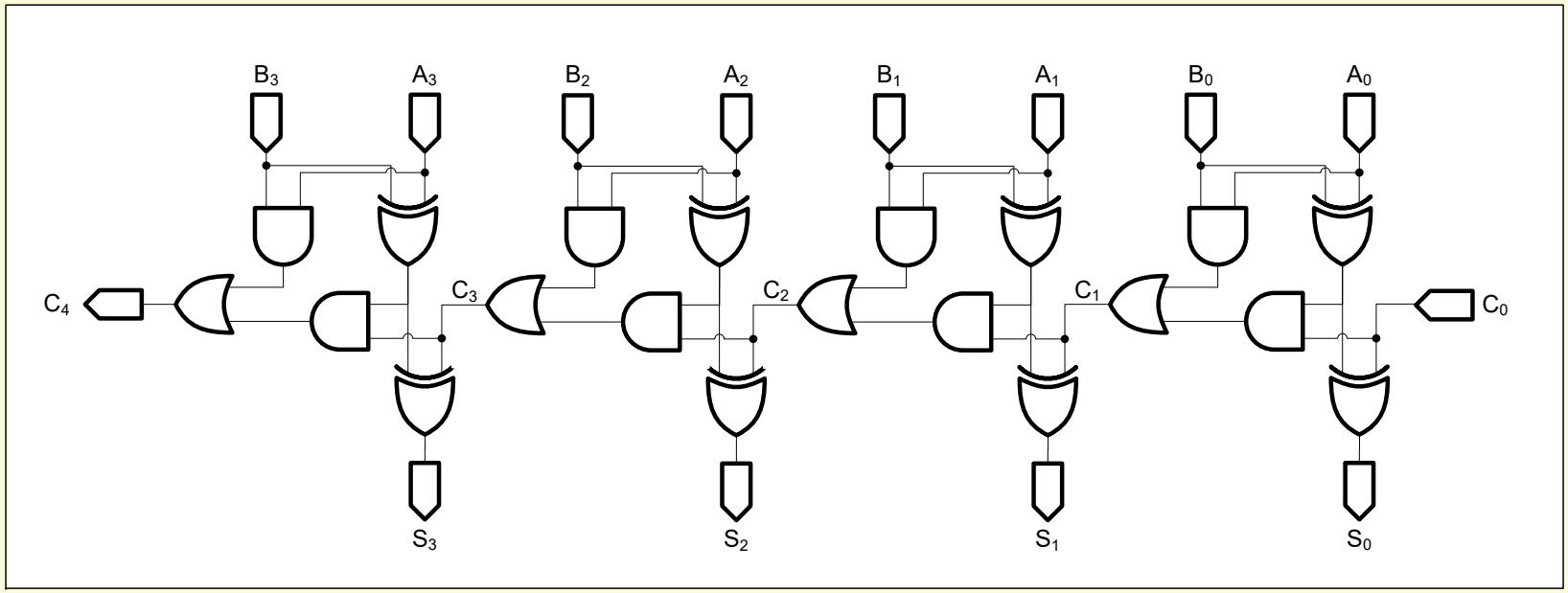
… additionneur 1 bit

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



Additionneur 4 bits …

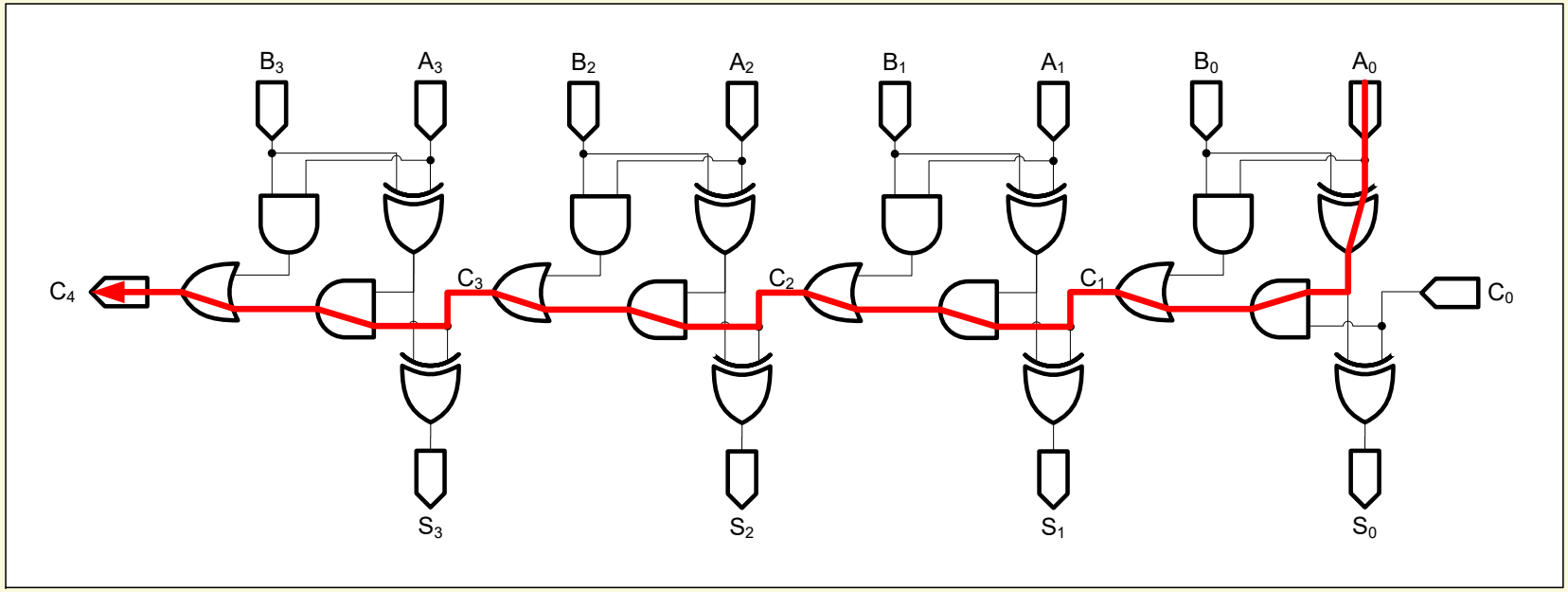
- Schéma additionneur 4 bits avec 4 Full-Adder :



Référence: Présentation additionneur de Yann Thoma

… additionneur 4 bits

- Chemin critique additionneur 4 bits

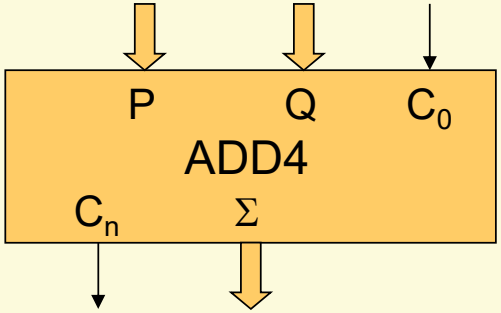


Référence: Présentation additionneur de Yann Thoma

Exercices II

1. Vous disposez de modules d'addition de deux nombres de 4 bits

   Donnez le schéma pour additionner deux nombres de 8 bits non signés



2. Proposez un schéma pour réaliser l'addition et la soustraction de deux nombres signés de 4 bits, en complément à 2, avec un seul additionneur 4 bits et de la logique.

2bis. Idem exe 2 pour deux nombres non signés de 4 bits

Exercices II

3. Déterminer l'équation du dépassement de capacité (nommé Overflow) pour les nombres signés en notation complément à 2
4. Proposer un schéma pour déterminer le bit supplémentaire pour tenir compte d'un dépassement lors d'une addition
   a. cas de nombres non-signés
   b. cas de nombres signés

Opérations et dépassement

Rappel représentation des nombres

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

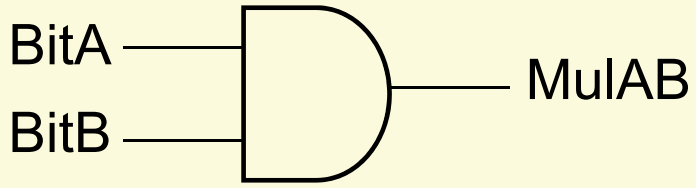
http://en.wikipedia.org/wiki/Carry_flag
http://en.wikipedia.org/wiki/Overflow_flag

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 la multiplication par 6 d'un nombre de 4 bits.
2. Réaliser un circuit qui réalise la conversion d'un nombre BCD de deux chiffres en binaire.
3. Réaliser un multiplicateur de deux nombres de 3 bits basé sur des addtionneurs
4. Réaliser un circuit qui réalise le calcul suivant: r = 5 * na + 7, 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

Décomposition en sous-systèmes

- Maîtrise de la complexité
- Simplifie la conception
  - spécification du fonctionnement d'un bloc
  - optimisation par bloc
  - identification fonctions communes
  - …
- Répartition du travail possible
- Test plus simple :
  - plus approfondi
  - meilleure fiabilité des sous-blocs

Exemple de décomposition

- Réaliser le système suivant :
  - Entrées : Na, Nb nombres non-signés de 8 bits
  - Sorties : Na_PP_Nb active si Na < Nb
    Na_PG_Nb active si Na > Nb
    Result nombre non-signé de 8 bits
  - Description du fonctionnement
    - Si Na > Nb alors
      Result = Na – Nb
      sinon si Na < Nb alors
      Result = Nb – Na
      sinon
      Result = 0

Exercices

- Série " Fonctions standard combinatoires"
  - Exercice nº 54, 55

Fin de la présentation