heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Quelques applications des systèmes combinatoires

Profs. Peña & Perez-Uribe & Mosqueron

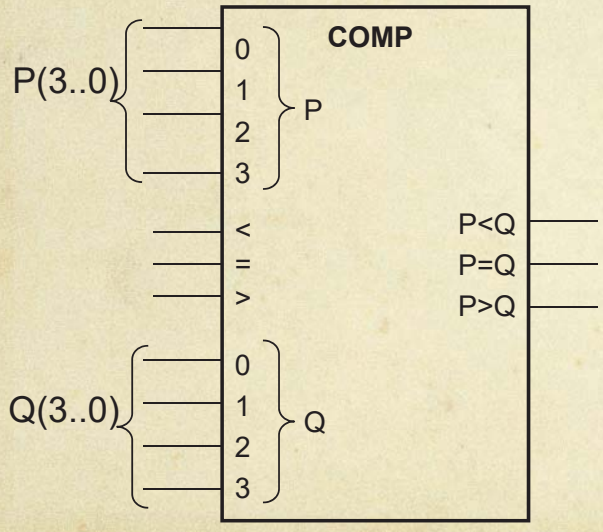
- Analyse:
  Déterminer le comportement d'un système à partir d'une description de sa structure

- Synthèse:
  Déterminer la structure qui produit un comportement donné. Plusieurs structures sont possibles pour un même comportement

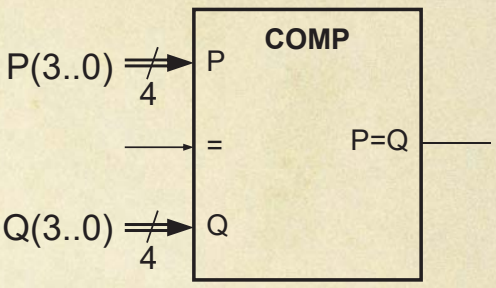
Comparateur

- 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)



Symbole IEEE/CEI **OBLIGATOIRE**



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)

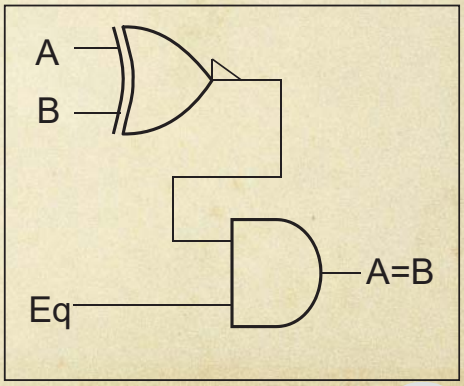
Table de vérité

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

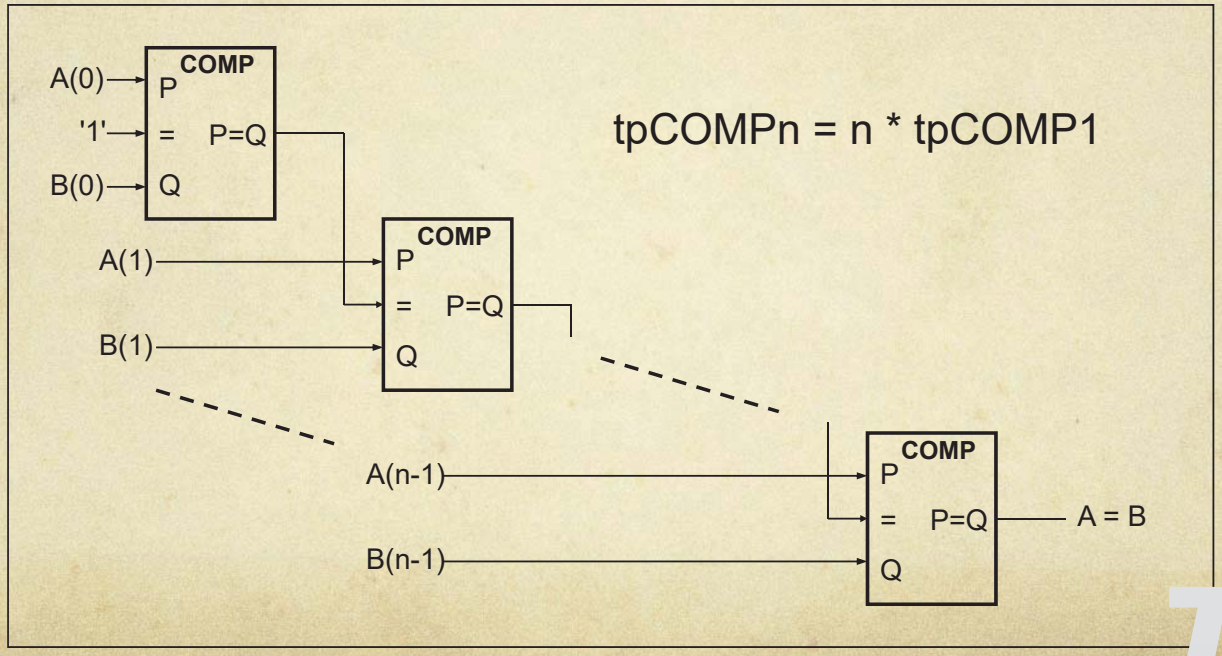
Eq \ B A	00	01	11	10
0	'0'	'0'	'0'	'0'
1	'1'	'0'	'1'	'0'

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

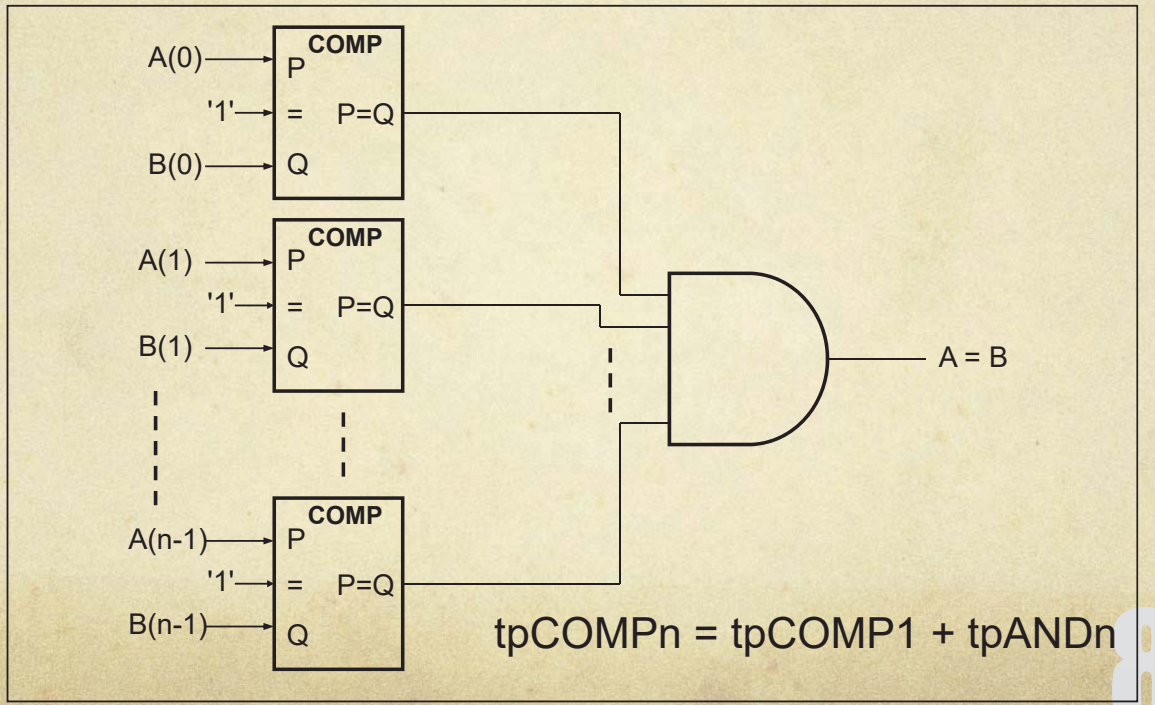
$= \overline{(A \oplus B)} \bullet Eq$



Décomposition en cascade

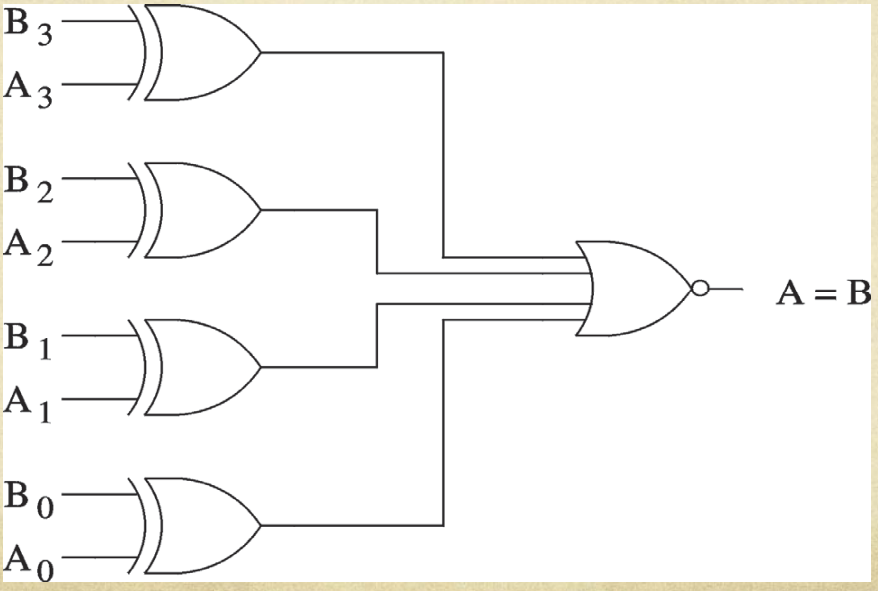


Décomposition en parallèle



Comparateurs

- Implémentation des opérateurs de comparaison (= , > , < , ≥ , ≤)



Comparateurs (2)

A_1	A_0	B_1	B_0	A>B	A==B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
...	...	...	...	...	...	...
1	1	1	1	0	1	0

Exercices I

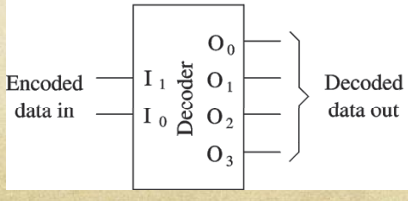
Soit un comparateur fournissant l'égalité
entre deux nombres de n bits.

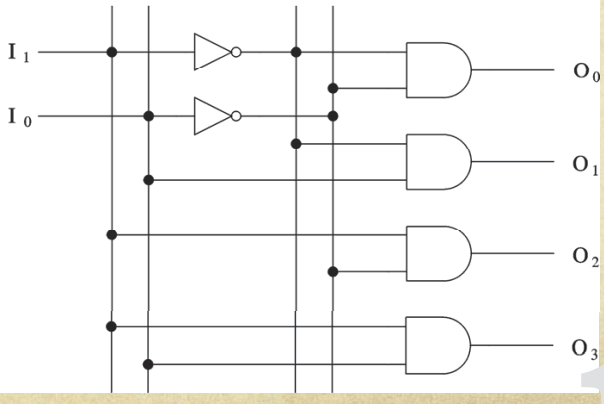
1. Donner l'équation logique de la sortie égalité (A_eq_B) pour des nombres de 5 bits.
   Indiquer le nombre total de termes de cette équation.
2. Indiquer le nombre total de termes de l'équation si les nombres ont 10 bits.
3. Quelle remarque pouvez-vous faire ?

Décodeurs

- Un décodeur détecte la présence d'une combinaison spécifique en entrée
- Pour un décodeur à n entrées, il y en a 2^n sorties
- Application typique: décodage d'adresses pour la mémoire

I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0





Décodeur X/Y

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

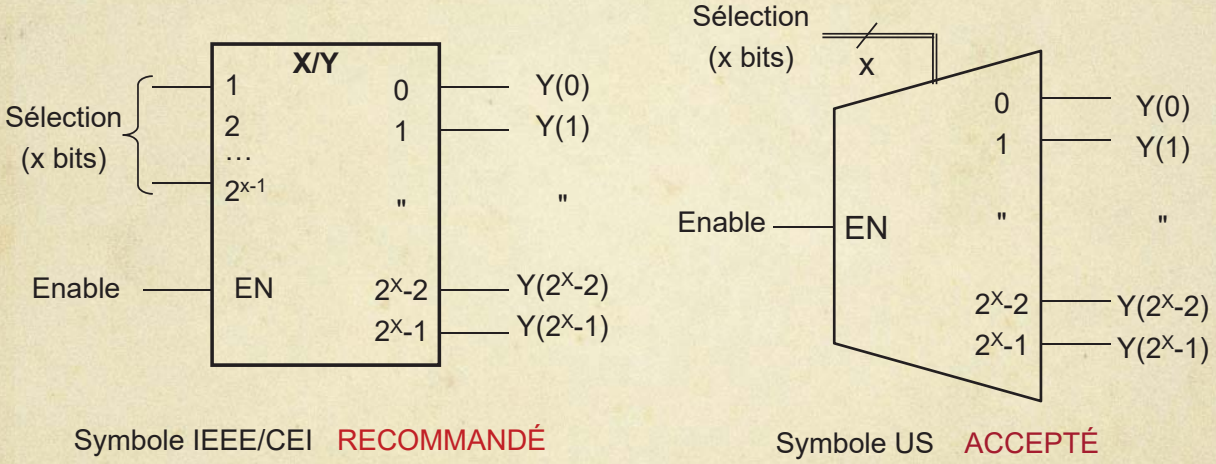
  => décode la valeur binaire d'entrée de **n** bits

  => 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 X/Y (symbole)

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

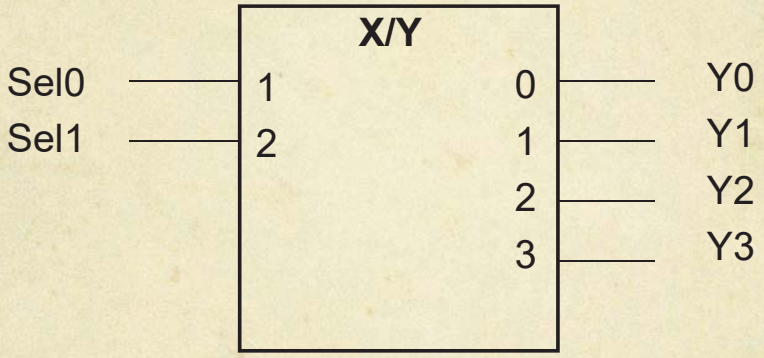


Symbole IEEE/CEI RECOMMANDÉ

Symbole US ACCEPTÉ

utilisation de symbole explicite !

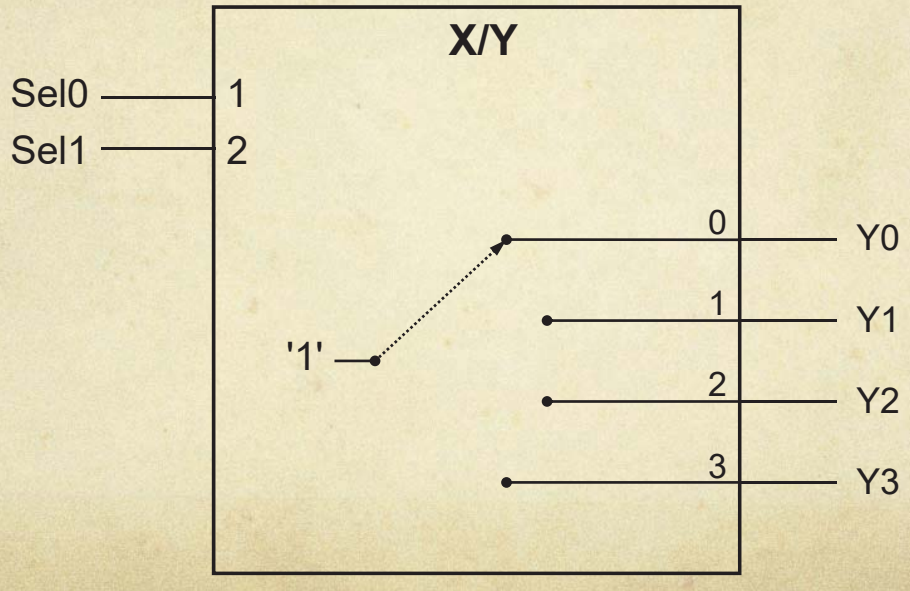
Décodeur 2 à 4, symbole



Symbole IEEE/CEI **RECOMMANDÉ**

Décodeur 2 à 4

- Fonctionnement de principe



Décodeur 2 à 4 (TDV, équations, schéma)

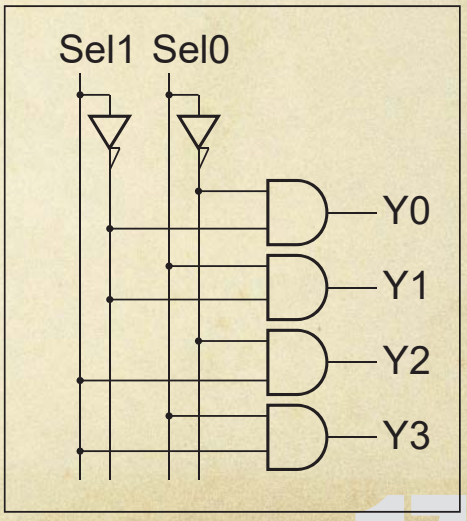
- Génère tous les mintermes de Sel1 et Sel0

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'

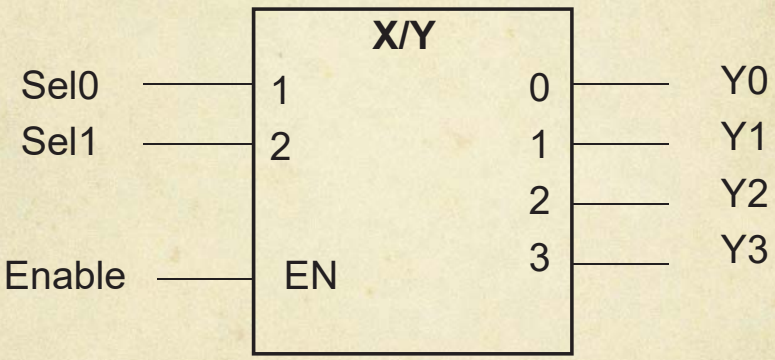
$Y0 = (\overline{Sel1} \cdot \overline{Sel0})$ $Y1 = (\overline{Sel1} \cdot Sel0)$

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



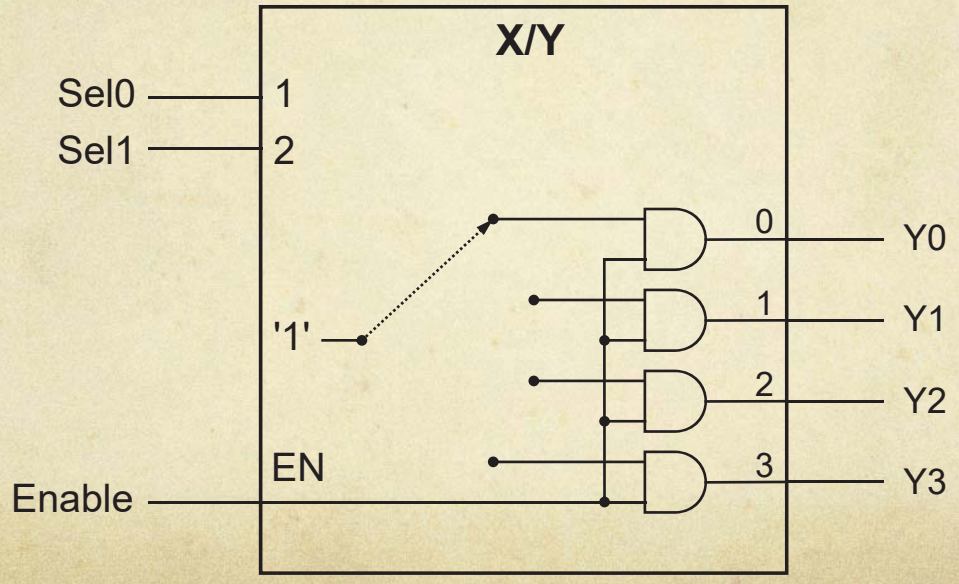
Décodeur 2 à 4 avec EN, symbole

- Version avec *Enable*



Décodeur 2 à 4 avec EN

- Fonctionnement de principe



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

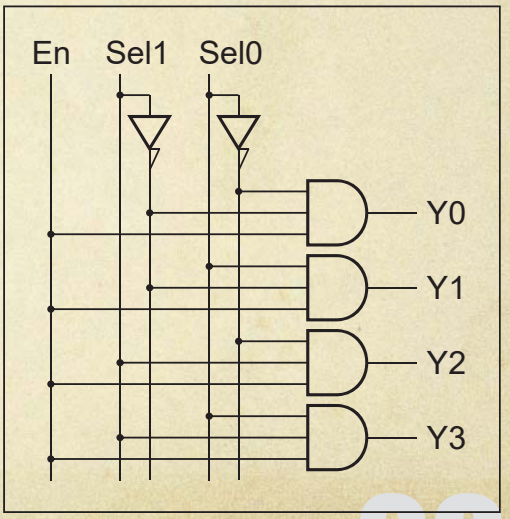
- Génère tous les mintermes de Sel1 et Sel0

Table de vérité

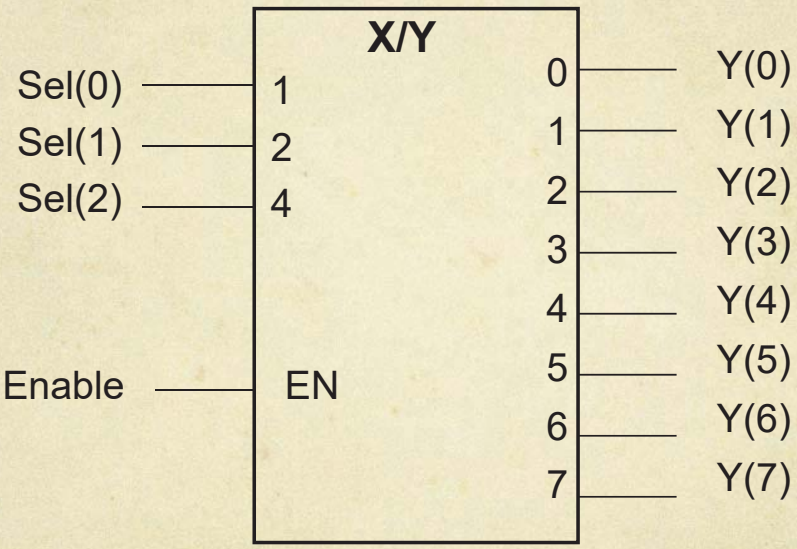
En	Sel1	Sel0	Y3	Y2	Y1	Y0
'0'	x	x	'0'	'0'	'0'	'0'
'1'	'0'	'0'	'0'	'0'	'0'	'1'
'1'	'0'	'1'	'0'	'0'	'1'	'0'
'1'	'1'	'0'	'0'	'1'	'0'	'0'
'1'	'1'	'1'	'1'	'0'	'0'	'0'

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

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

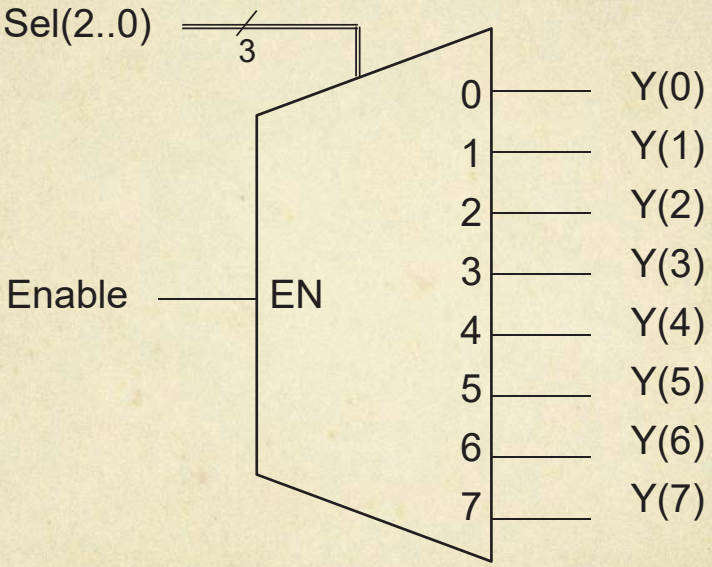


Décodeur 3 à 8, symbole CEI



Symbole IEEE/CEI **RECOMMANDÉ**

Décodeur 3 à 8, symbole US



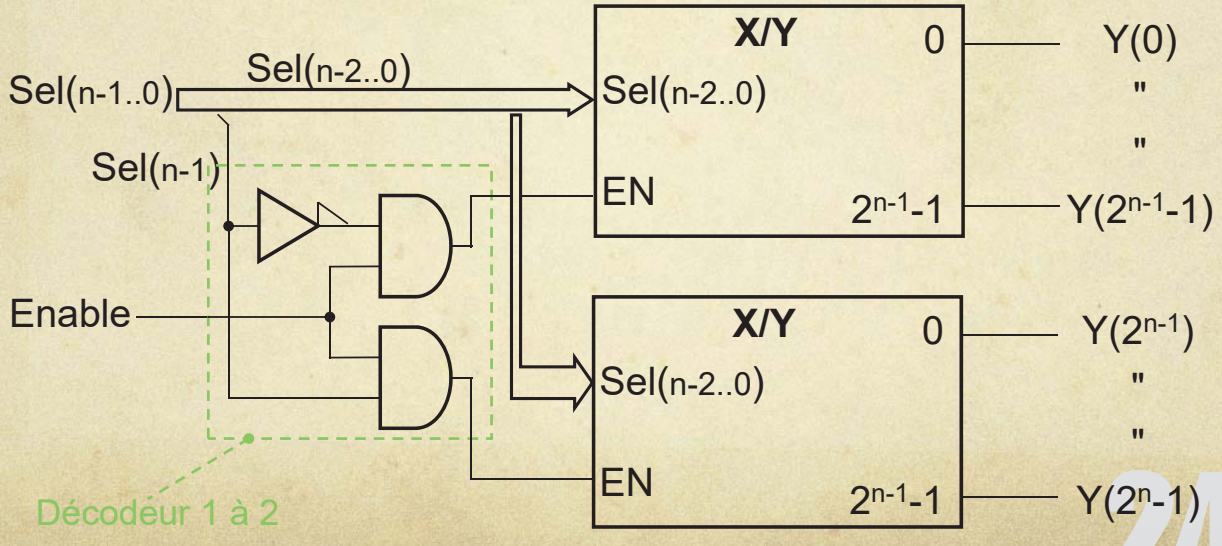
Symbole US **ACCEPTÉ**

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, 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

  ⇒ 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 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'

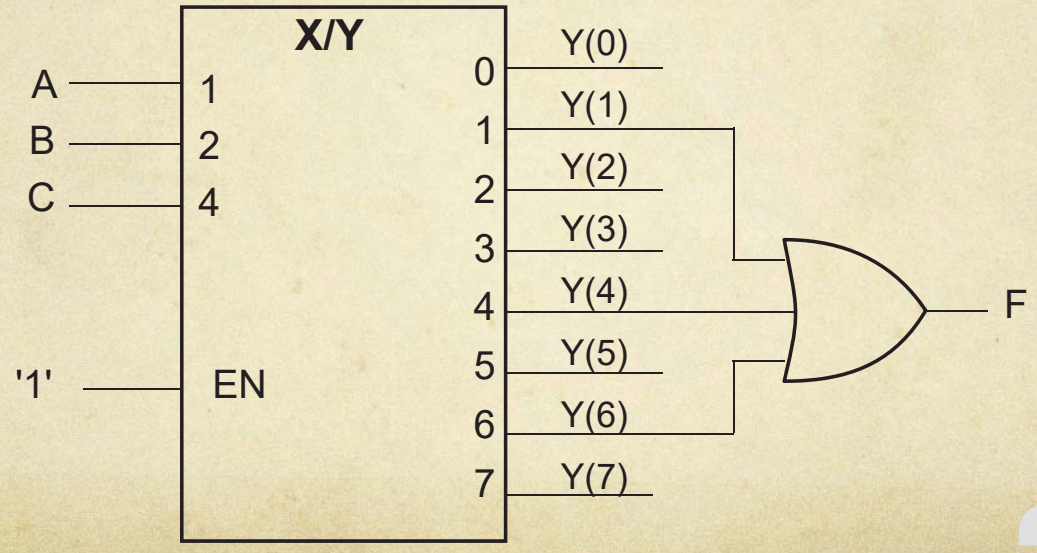
Equation F :

(somme de mintermes)

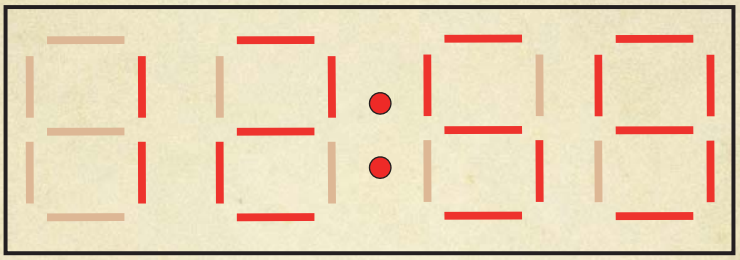
$F(C, B, A) = \Sigma\ 1, 4, 6$

... décodeur en générateur de fonction

Schéma F(C, B, A) = Σ 1, 4, 6

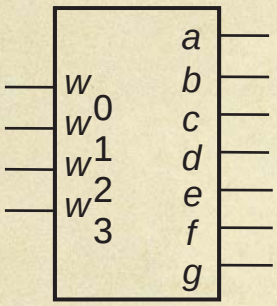


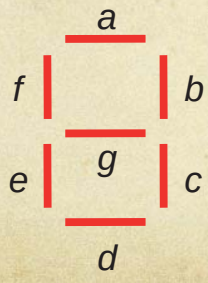
Application: afficheur 7 segments



Le code **BCD** ou Binary-coded decimal, qui peut se traduire en français par *décimal codé en binaire*, est très habituel lorsqu'une valeur numérique doit être affichée. Les valeurs 0 à 9 sont codées en binaire en utilisant 4 bits (un *nibble*). Dans certains systèmes les 4 bits de poids fort d'un octet sont à 0; dans d'autres, ils codent une deuxième valeur.

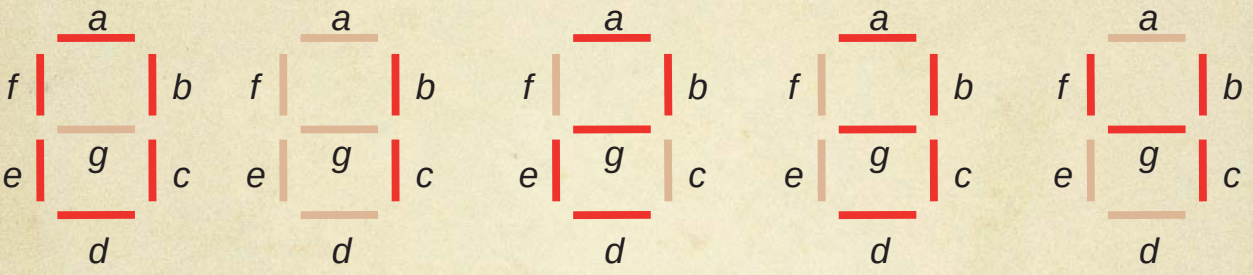
Décodeur BCD à 7-segments

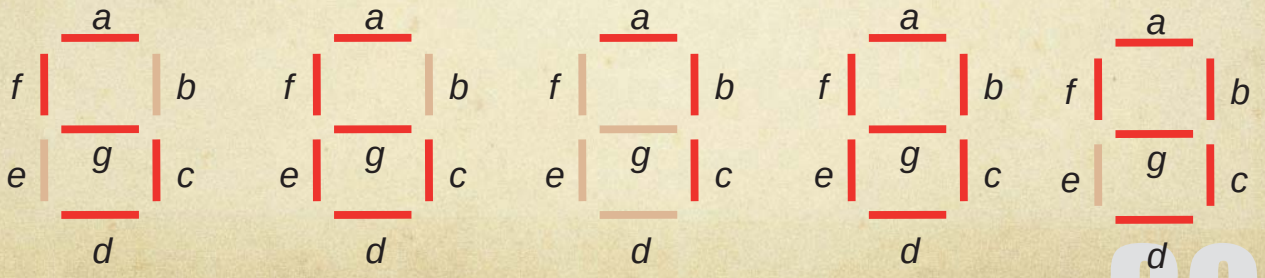




w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Décodeur BCD à 7-segments





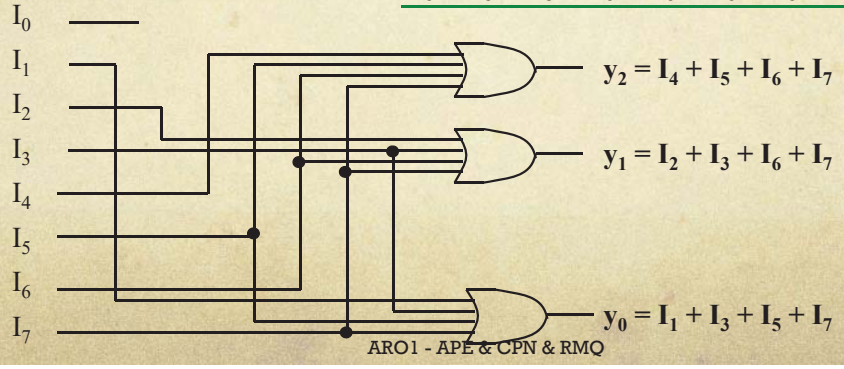
Décodeur binaire à 7-segments

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	Ø	Ø	Ø	Ø	Ø	Ø	Ø
...	...	...	...	...	...	...	...	...	...	...
1	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Encodeurs

- Convertisseur qui a moins de bits en sortie qu'en entrée
- Exemple: 2^n bits -> binaire
- Il ne peut pas gérer plus d'une entrée à 1

Inputs								Outputs		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	y_2	y_1	y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

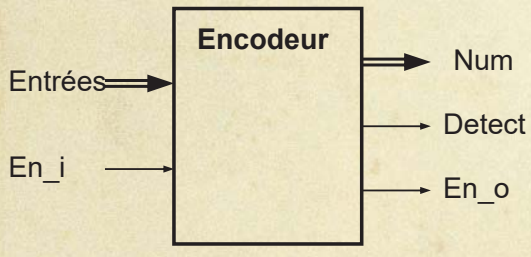


Encodeur de priorité

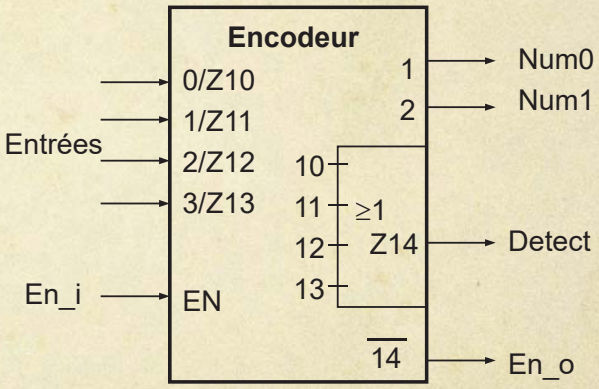
- But : déterminer le numéro de l 'entrée active ayant le degrés de priorité le plus élevé
- Ce module comporte souvent en plus une entrée d 'autorisation
- Ce module comporte une sortie indiquant qu'une des entrées au moins est active (Nécessaire pour identifier une action sur l 'entrée 0)

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

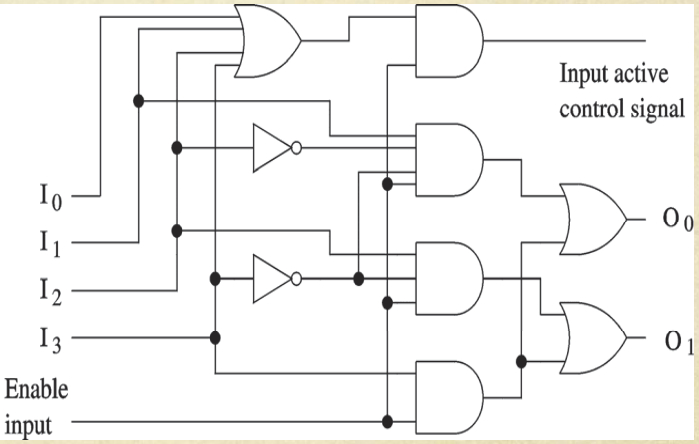
Num1 = In3 + In2

Num0 = In3 + $\overline{\text{In2}}$ • In1

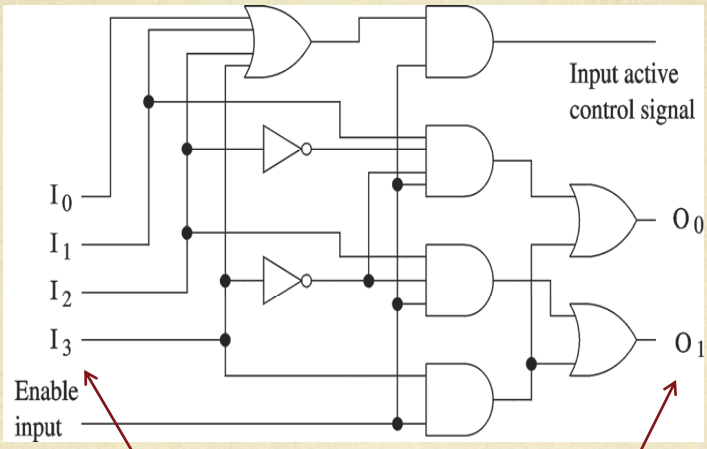
Encodeurs de priorité

- Encodeur qui peut gérer plus d'une entrée à 1

Enable input	I_3	I_2	I_1	I_0	O_1	O_0	Input active control signal
0	X	X	X	X	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	X	0	1	1
1	0	1	X	X	1	0	1
1	1	X	X	X	1	1	1



Interruptions



Qui a besoin de moi ?

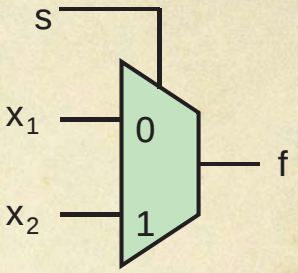
Qui est servi en premier ?

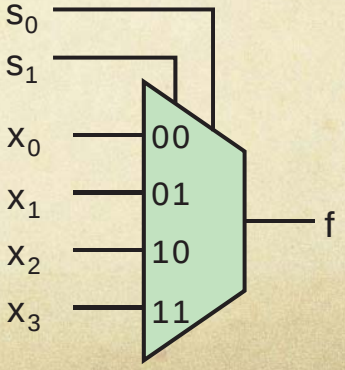
Multiplexeur

- 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

Multiplexeurs

- Multiplexeur
  - 2^n entrées
  - n entrées de sélection
  - une sortie
- Les entrées de sélection s déterminent quelle entrée est connectée à la sortie

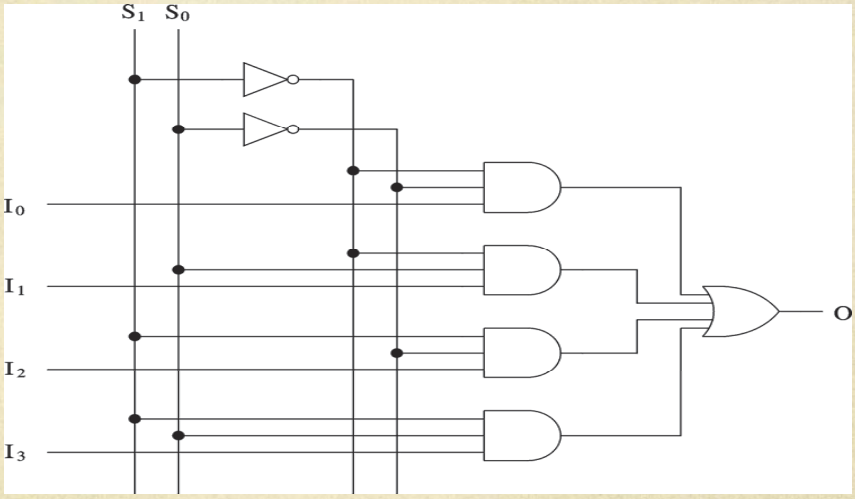




Multiplexeurs (suite)

Implementation d'un MUX à 4 entrées

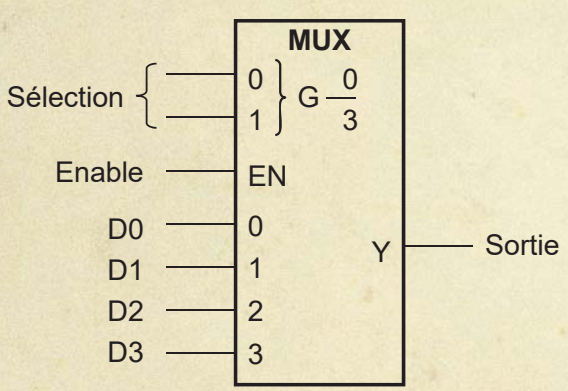
s_1	s_0	f
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



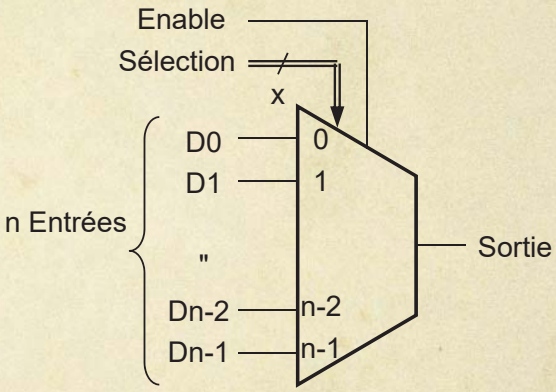
$f = s_1' s_0' I_0 + s_1' s_0 I_1 + s_1 s_0' I_2 + s_1 s_0 I_3$

Multiplexeur n à 1, symboles

- Symbole multiplexeur n à 1 :



Symbole IEEE/CEI **RECOMMANDÉ**

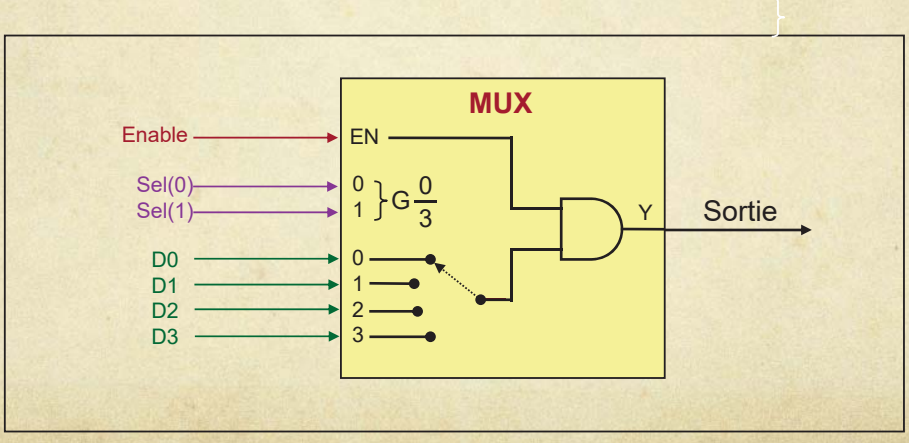


Symbole US **ACCEPTÉ**

utilisation de symbole explicite !

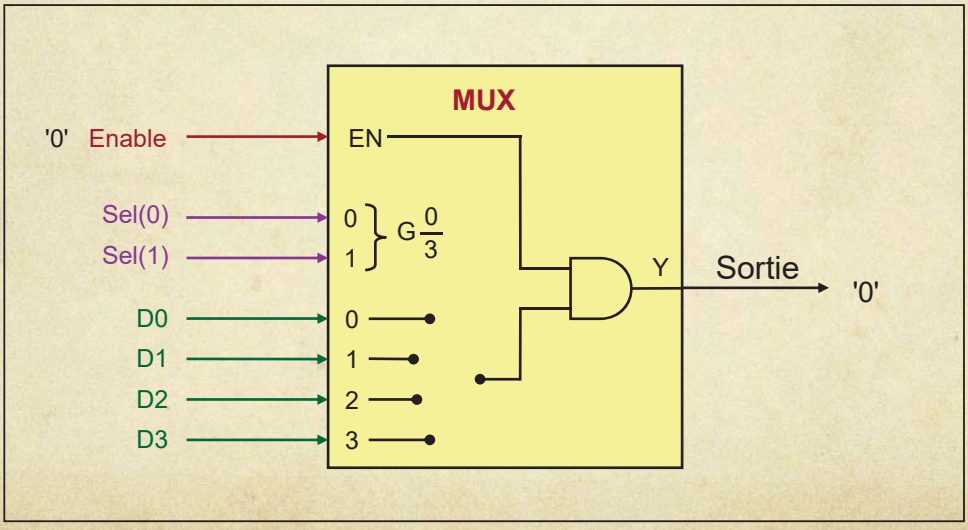
Exemple fonctionnel du multiplexeur

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



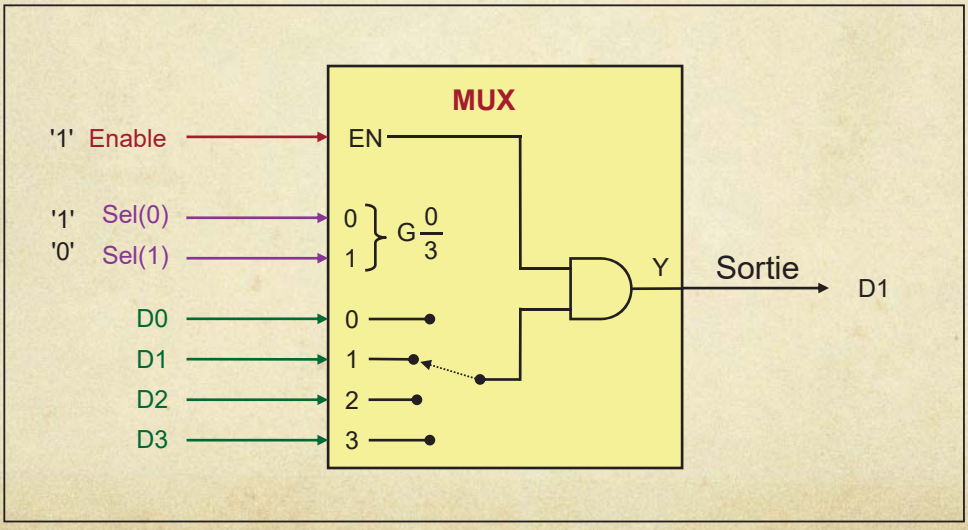
Fonctionnement multiplexeur 4 à 1

- Signal *Enable* inactif :



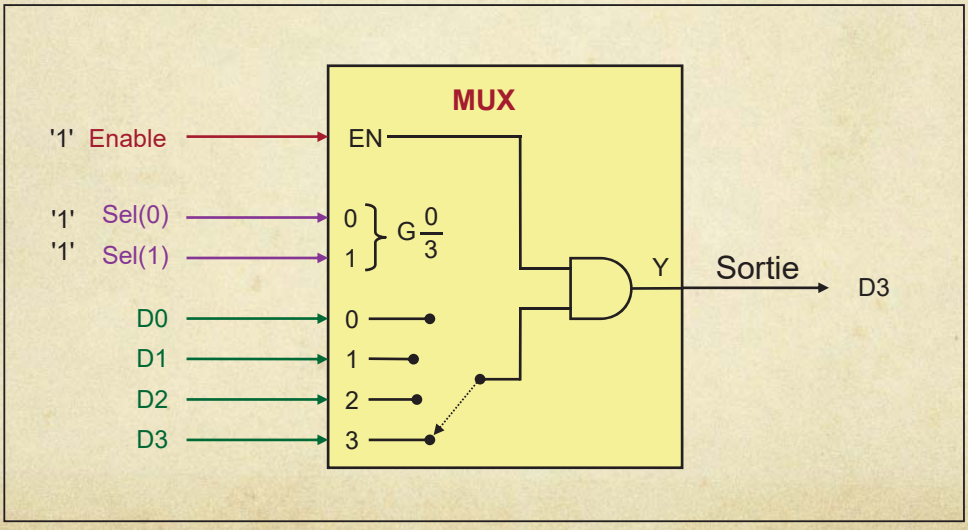
Fonctionnement multiplexeur 4 à 1

- 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)

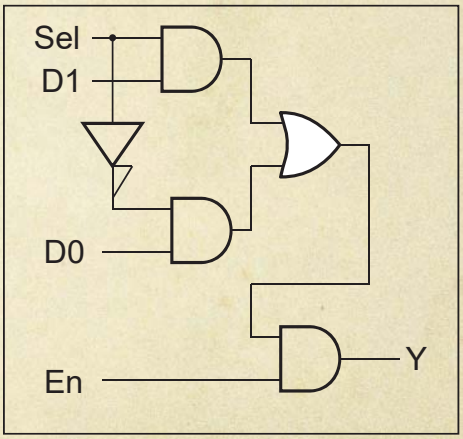
- Système combinatoire à 4 entrées (En, Sel, D1 et D0)
  et
  une sortie (Y)

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'

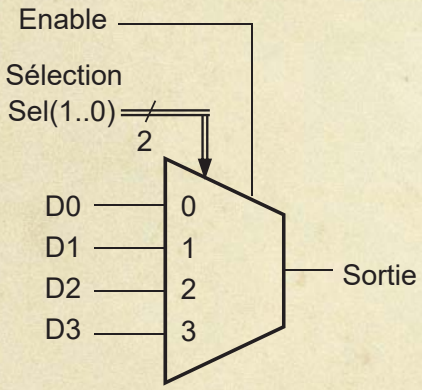
En Sel / D1 D0	00	01	11	10
00	'0'	'0'	'0'	'0'
01	'0'	'0'	'0'	'1'
11	'0'	'0'	'1'	'1'
10	'0'	'0'	'1'	'0'

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

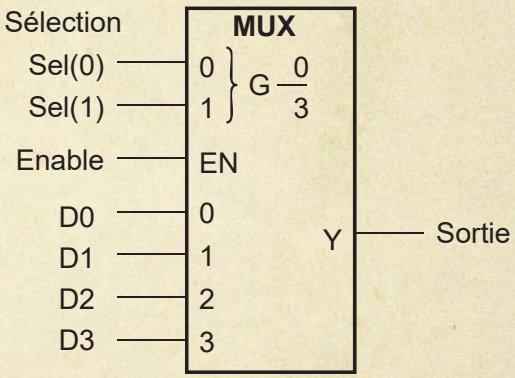


Multiplexeur 4 à 1, symbole

- Symboles multiplexeur 4 à 1:



Symbole US **ACCEPTÉ**



Symbole IEEE/CEI **RECOMMANDÉ**

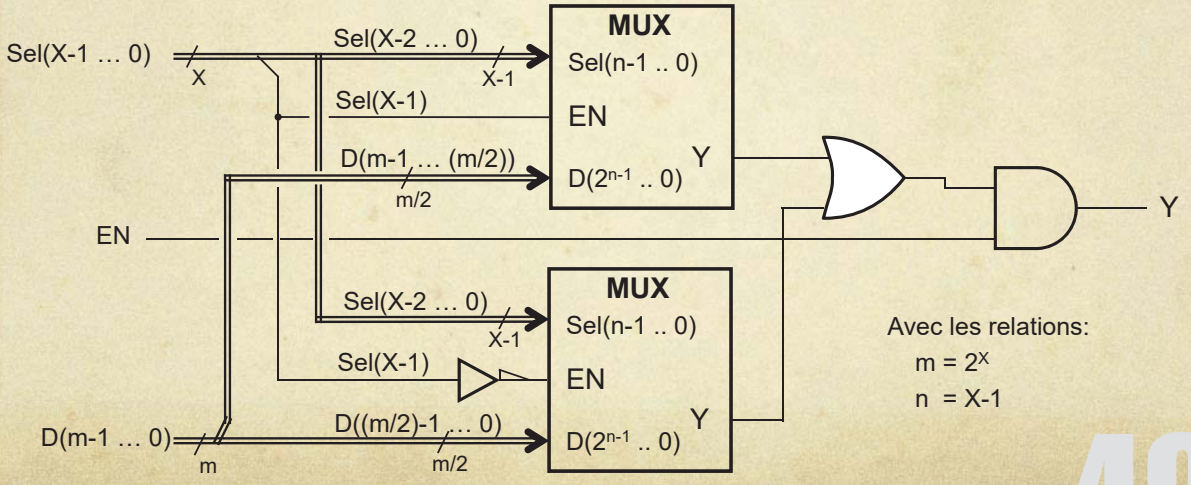
Multiplexeur 4 à 1, table de vérité

- Table de vérité compactée :

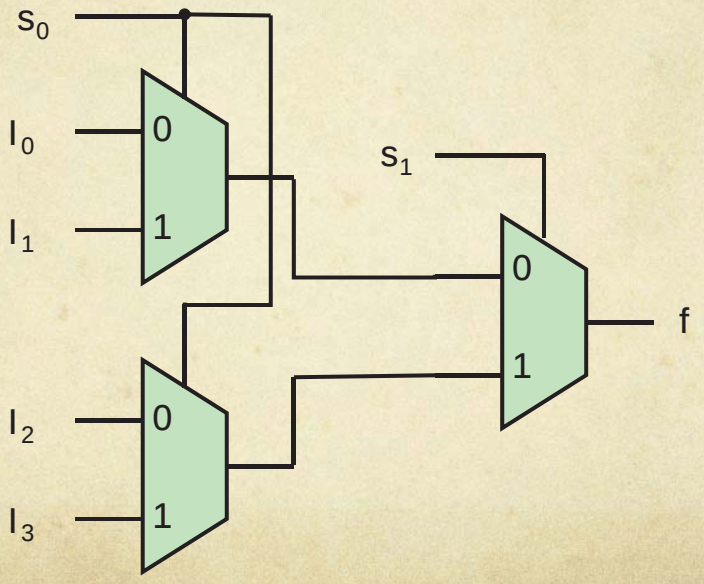
EN	Sel(1)	Sel(0)	Y
0	x	x	0
1	0	0	D0
1	0	1	D1
1	1	0	D2
1	1	1	D3

Multiplexeur 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



MUX 4-1 à l'aide des MUX 2-1



Le MUX en générateur de fonction

Equation logique :

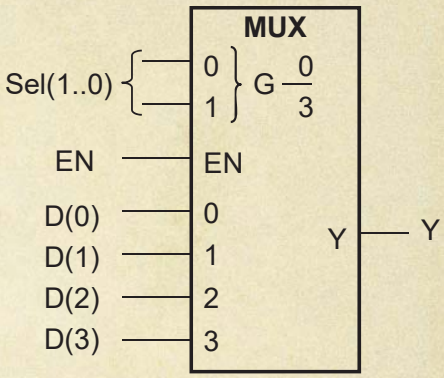
Y = EN **and**

(**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

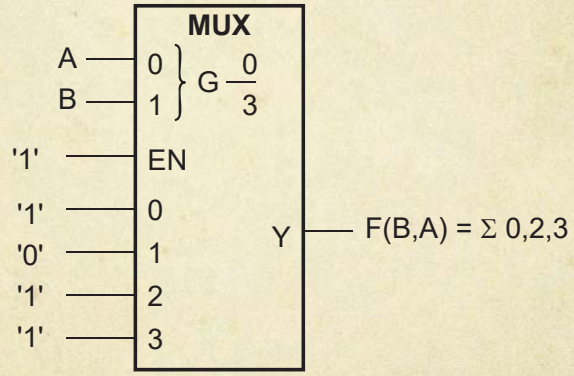
- Table de vérité Mux 4à1 compacte

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)

MUX 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

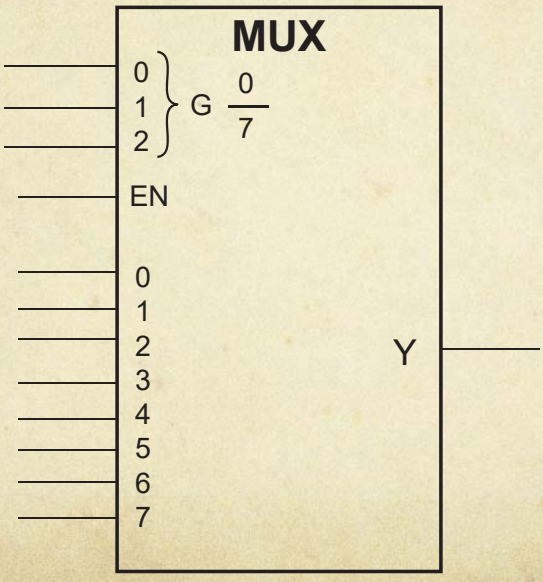
Exercice :

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

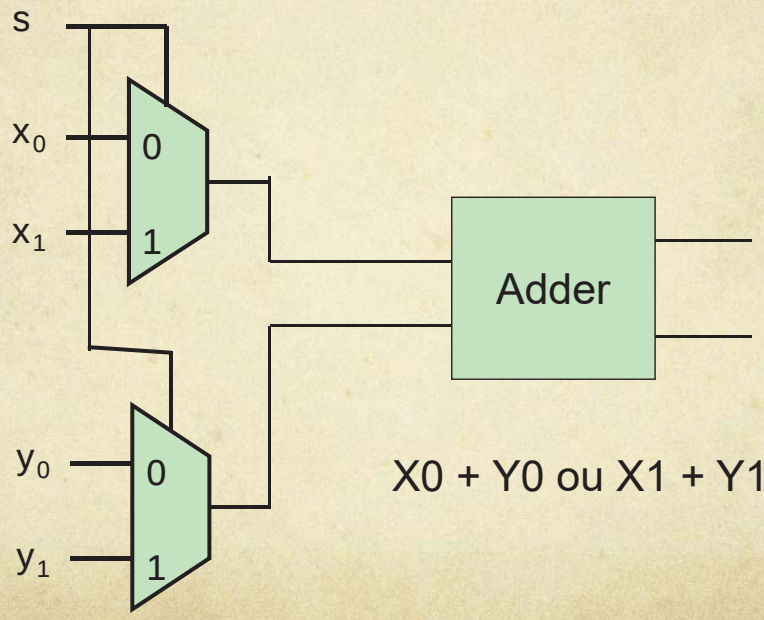
Donner le schéma logique de la fonction P en utilisant un multiplexeur 8 à 1?

MUX en générateur de fonction

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

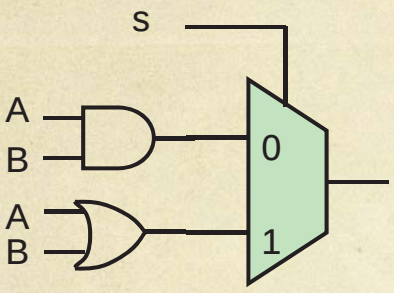


Application des MUXs



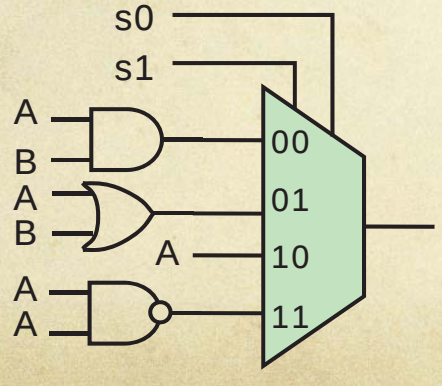
X0 + Y0 ou X1 + Y1

Application des MUXs



F = AB si s = 0

F = A+B si s = 1



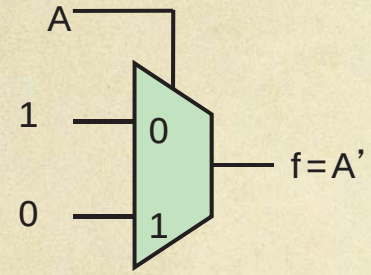
F = AB si s1s0 = 00

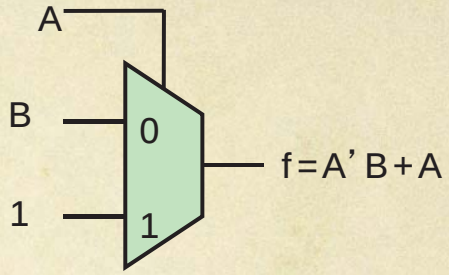
F = A+B si s1s0 = 01

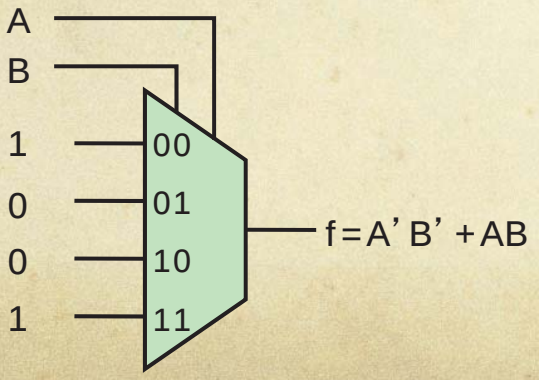
F = A si s1s0 = 10

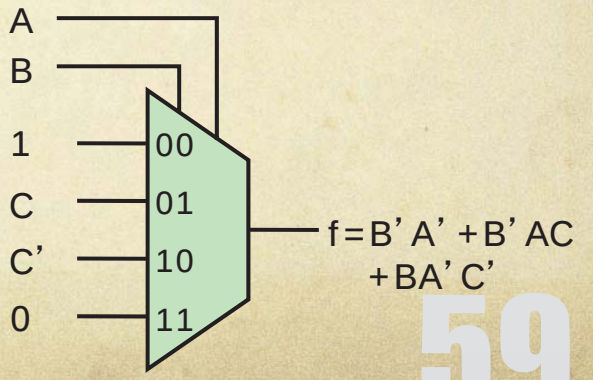
F = A' si s1s0 = 11

Des MUX pour réaliser des fonctions combinatoires





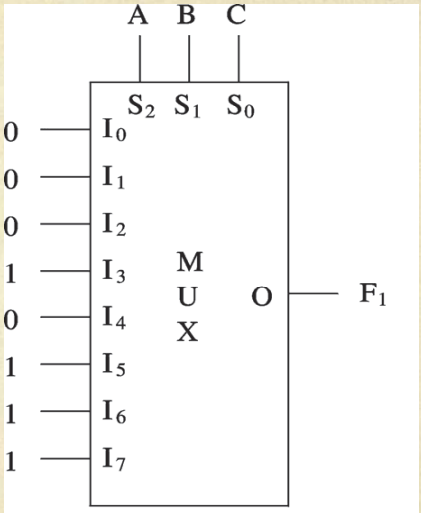




Fonction majorité à l'aide des MUX

Original truth table

A	B	C	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

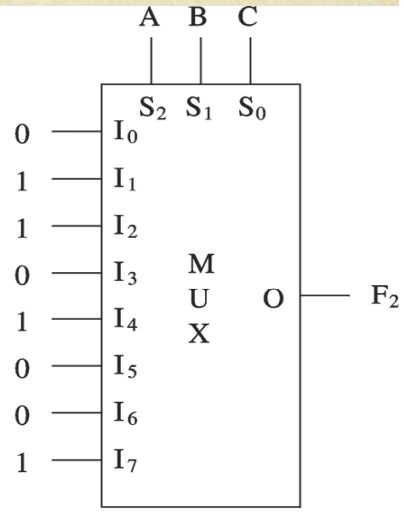


Majority function

Fonction parité paire à l'aide des MUX

Original truth table

A	B	C	F_1
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



Even-parity function

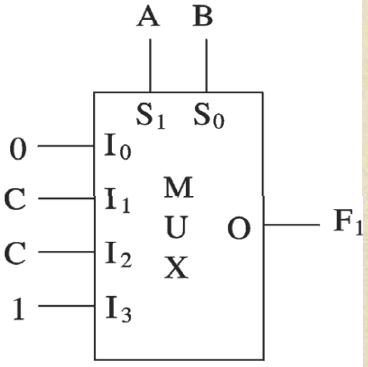
Fonction majorité: réalisation optimisée

Original truth table

A	B	C	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

New truth table

A	B	F_1
0	0	0
0	1	C
1	0	C
1	1	1



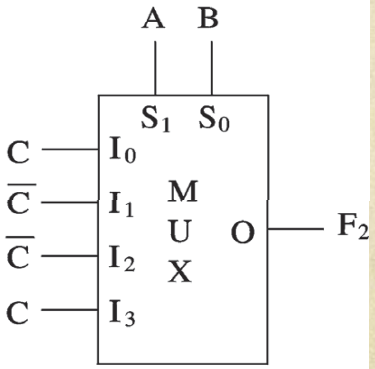
Fonction de parité paire: réalisation optimisée

Original truth table

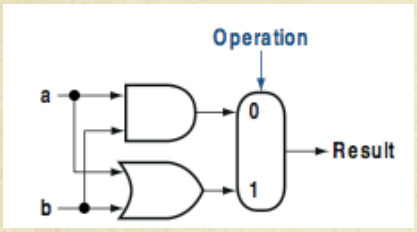
A	B	C	F_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

New truth table

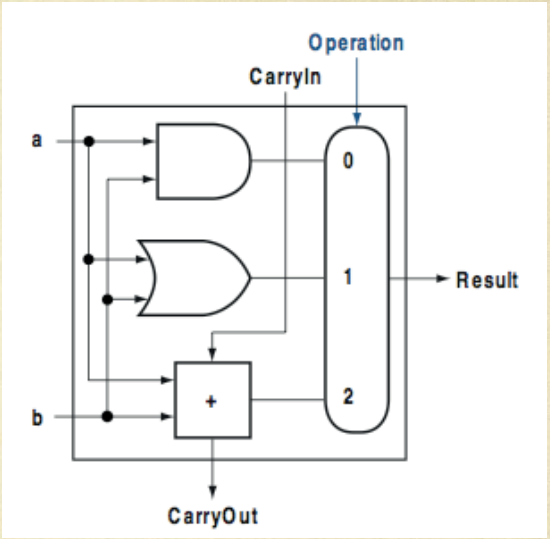
A	B	F_1
0	0	C
0	1	$\overline{C}$
1	0	$\overline{C}$
1	1	C



Unité Arithmétique et Logique

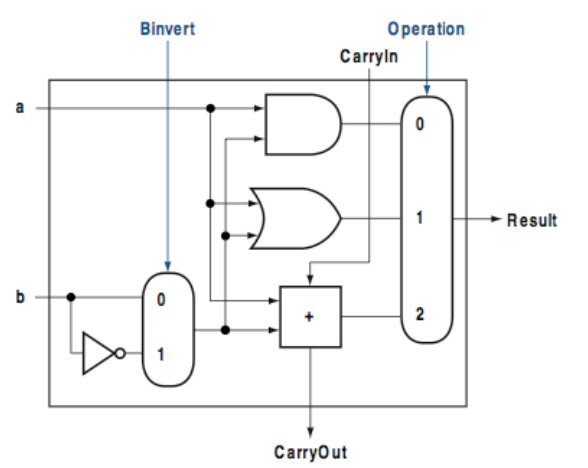


1-bit ALU: AND et OR

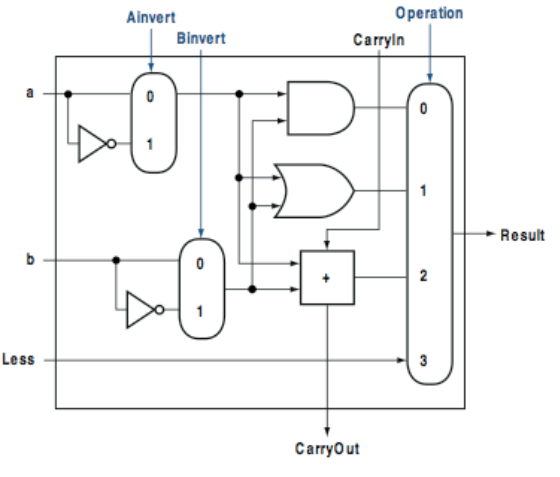


1-bit ALU: AND, OR et ADD

Unité Arithmétique et Logique (2)

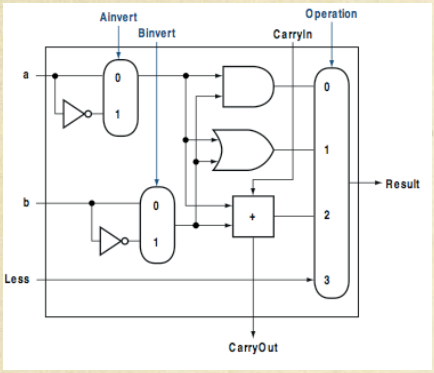


1-bit ALU: AND, OR, ADD et SUB



1-bit ALU: AND, OR, NOR, NAND, ADD, SUB

Unité Arithmétique et Logique (3)

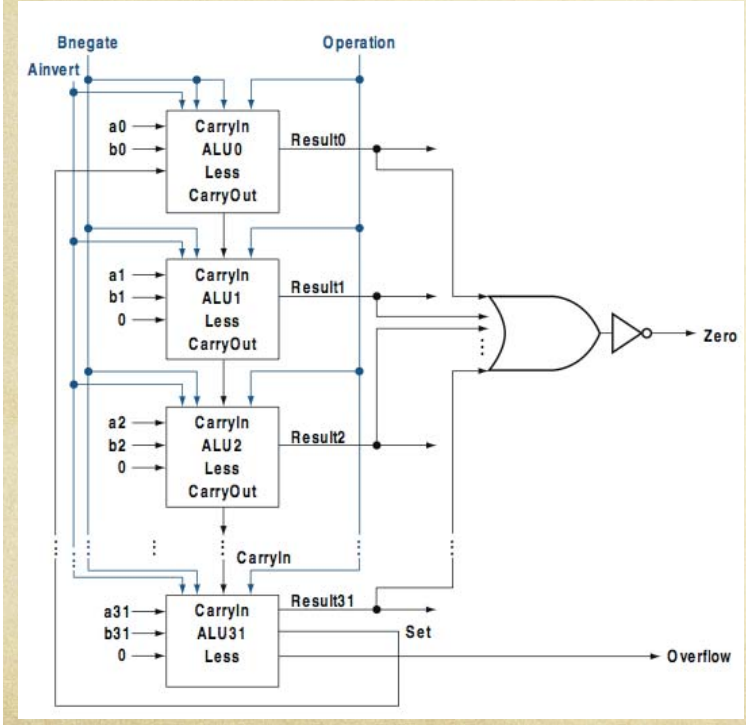


1-bit ALU

ALU op	Fonction
00000	AND
00001	OR
00010	ADD
01110	SUB
01111	SLT
11000	NOR
11001	NAND

Ainv Binv Cin Op(mux)

Vers une ALU "simple" 32-bit (3)



32-bit ALU:

AND,
OR,
NAND,
NOR,
ADD avec calcul d' overflow,
SUB,
détecteur de zéro et
SLT(set on less than)
si a < b alors sortie = 1
sinon sortie = 0

Jeu d' instructions d'un processeur

- Chaque processeur à un jeux d'instructions, qui comporte l'ensemble d'opérations que le processeur peut exécuter.
- Chaque instruction est identifié avec un code, appellé *opcode* (e.g., 00010, 00011) et avec un *mnémonique* (e.g., ADD, SUB).
- Une suite d'instructions d'un processeur donne lieu à un programme en langage machine

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

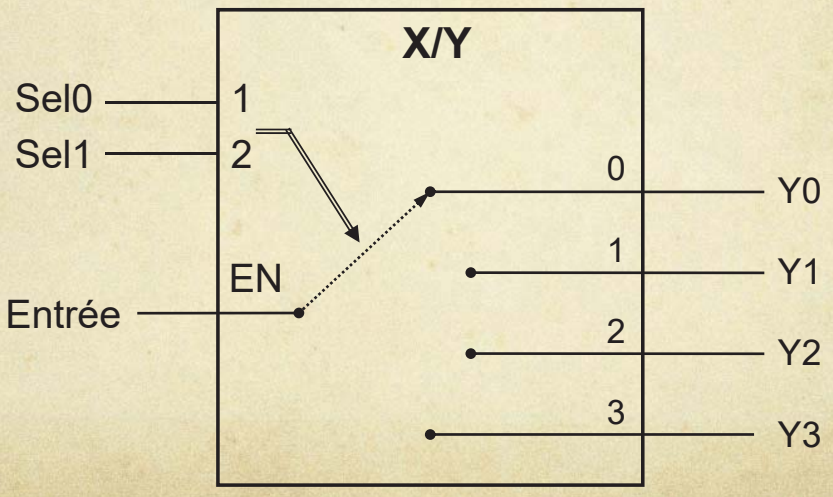
http://en.wikipedia.org/wiki/Carry_flag
http://en.wikipedia.org/wiki/Overflow_flag

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