Faculté des Sciences et Technologies

Dpt Informatique

LIRIS

Logique combinatoire et représentation numérique des données

Hamid Ladjal

hamid.ladjal@univ-lyon1.fr
hamid.ladjal@liris.cnrs.fr

Master MEEF CAPES Maths Option Informatique

http://liris.cnrs.fr/nicolas.pronost/UCBL/CapesInfo/

Plan

1) Calcul propositionnel, l'algèbre de Boole et la logique combinatoire

2) Circuits combinatoires

3) Représentation et codage des données

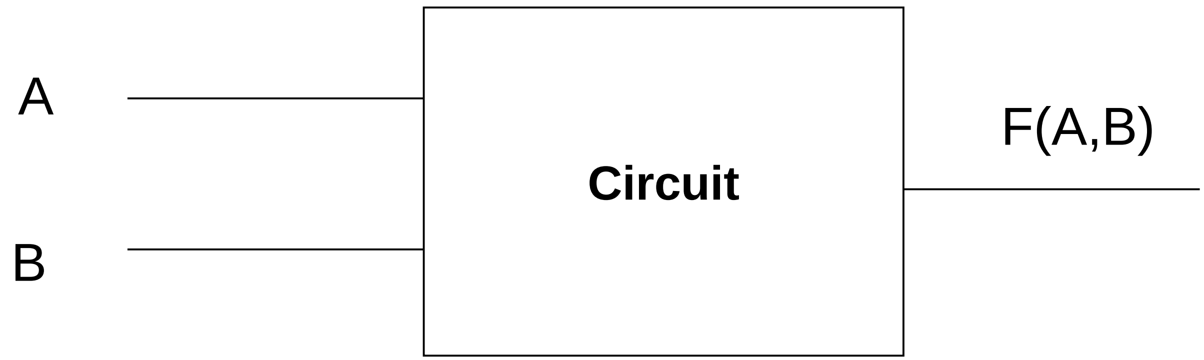
Logique combinatoire

- *Calcul propositionnel naïf et l'algèbre binaire*
- *Opérateurs de base*
- *Propriétés*
- *Circuits combinatoires*

Introduction

- Les machines numériques sont constituées d'un ensemble de circuits électroniques.

- Chaque circuit fournit une fonction logique bien déterminée; opérations logiques ou arithmétiques (addition, soustraction, comparaison ,…..).



Introduction

- Pour concevoir et réaliser ce circuit on doit avoir un modèle mathématique de la fonction réalisée par ce circuit .

- Ce modèle doit prendre en considération le système binaire.

- Le modèle mathématique utilisé est celui de Boole.

Algèbre de Boole

1854 : Georges Boole propose une algèbre

Propositions vraies ou fausses
et opérateurs possibles → Algèbre de Boole

↓

Étude des systèmes binaires :

Possédant deux états s’excluant mutuellement

C’est le cas des systèmes numériques
(des sous ensembles : les circuits logiques)

Algèbre binaire

On se limite : Base de l'algèbre de Boole
Propriétés indispensables aux systèmes logiques

Définitions :

- États logiques : 0 et 1, Vrai et Faux, H et L
  (purement symbolique)

- Variable logique : Symbole pouvant prendre
  comme valeur des états logiques (A,b,c, Out ...)

- Fonction logique : Expression de variables et d'opérateurs
  (f = not(a)^ (c OR r.t))

Calcul propositionnel

Algèbre de Boole sur [0,1] = algèbre binaire

Structure d'algèbre de Boole

- 2 lois de composition interne (LCI)
- 1 application unaire

2 LCI : ET, OU

- Somme (OU, Réunion, Disjonction)

  $s = a + b = a \vee b$

- Produit (ET, intersection, Conjonction)

  $s = a . b = ab = a \wedge b$

Application unaire :

- Not (complémentation, inversion, négation, non) $s = \overline{a} = \text{not}(a) = \neg a$

Fonctions logiques

Fonction logique à n variables f(a,b,c,d,...,n)

$$[0,1]^n \longrightarrow [0,1]$$

- Une fonction logique ne peut prendre que deux valeurs
- Les cas possibles forment un ensemble fini (2^n)

- Descriptions, preuves possibles par énumération
  comparer f(a,b,c,..n) et g(a,b,c,..,n)
  = comparer les tables représentant f et g

La table de fonction logique = table de vérité

Opérateurs logiques de base

OU (OR)

- Le OU est un opérateur binaire (deux variables) , à pour rôle de réaliser la somme logique entre deux variables logiques.
- Le OU fait la disjonction entre deux variables.
- Le OU est défini par F(A,B)= A + B (il ne faut pas confondre avec la somme arithmétique)

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

ET (AND)

- Le ET est un opérateur binaire (deux variables) , à pour rôle de réaliser le Produit logique entre deux variables booléennes.
- Le ET fait la conjonction entre deux variables.

- Le ET est défini par : F(A,B)= A . B

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

NON (négation)

- NON : est un opérateur unaire (une seule variable) qui à pour rôle d'inverser la valeur d'une variable .

$$F(A) = \text{Non } A = \overline{A}$$

(lire : A barre)

A	$\overline{A}$
0	**1**
1	**0**

Tables de vérité de ET, OU, NON

a \ b	0	1
0	0	1
1	1	1

$s = a + b$

S est vrai si a OU b est vrai.

a b	s
0 0	0
0 1	1
1 0	1
1 1	1

a \ b	0	1
0	0	0
1	0	1

$s = a \,.\, b$

S est vrai si a ET b sont vrais.

a b	s
0 0	0
0 1	0
1 0	0
1 1	1

a	
0	1
1	0

$s = \overline{a}$

S est vrai si a est faux

a	s
0	1
1	0

Deux autres opérateurs : NAND,NOR

a \ b	0	1
0	1	1
1	1	0

$s = a \uparrow b = \overline{a.b}$

S est vrai si a OU b est faux.

NAND (Not-AND)

a \ b	0	1
0	1	0
1	0	0

$s = a \downarrow b = \overline{a+b}$

S est vrai si ni a, ni b ne sont vrais.

NOR (Not-OR)

NAND et NOR ne sont pas associatifs

Encore un opérateur : XOR

a \ b	0	1
0	0	1
1	1	0

$$s = a \oplus b = a.\overline{b} + \overline{a}.b$$

S est vrai si a OU b est vrai mais pas les deux.

XOR (Ou-Exclusif) vaut 1 si a est différent de b
Opérateur de différence (disjonction)

Encore un opérateur : XOR

XOR est associatif $s = a \oplus b \oplus c \quad ..... \oplus n$

vaut 1 si le nombre de variables à 1 est impair.

$$s = \overline{a \oplus b} = \overline{a} \oplus b = a \oplus \overline{b} = a \text{ XNOR } b$$

XNOR = $\overline{\text{XOR}}$ vaut 1 si a = b

Inverseur programmable : (le programme vaut 0 ou 1)

$a \oplus 1 = \overline{a}$ $\qquad$ $a \oplus 0 = a$

Simplification des fonctions logiques

Simplification /optimisation ?

Méthodes «classiques» de simplifications :

- pas de solution unique
- indépendant de la technologie
- le temps n’est pas pris en compte

La simplification «mathématique» n’est pas toujours optimale en regard des critères d’optimisation technologiques.

Simplification des fonctions logiques

- L'objectif de la simplification des fonctions logiques est de :
  - réduire le nombre de termes dans une fonction
  - et de réduire le nombre de variables dans un terme

- Cela afin de réduire le nombre de portes logiques utilisées → réduire le coût du circuit

- Plusieurs méthodes existent pour la simplification :
  - La Méthode algébrique
  - Les Méthodes graphiques : (ex : tableaux de karnaugh)

Propriétés de ET,OU,NON

- Commutativité

  $a+b = b+a$

  $a.b = b.a$

- Associativité

  $a+(b+c) = (a+b)+c$

  $a.(b.c) = (a.b).c$

- Distributivité

  $a.(b+c) = a.b+a.c$

  $a+(b.c) = (a+b).(a+c)$

- Idempotence

  $a+a = a$

  $a.a = a$

- Absorption

  $a+a.b = a$

  $a.(a+b) = a$

- Involution

  $\overline{\overline{a}} = a$

Propriétés de ET,OU,NON

- Elément neutre

$$a+0 = a$$

$$a.1 = a$$

- Elément absorbant

$$a+1 = 1$$

$$a.0 = 0$$

- Inverse

$$a+\overline{a} = 1$$

$$a.\overline{a} = 0$$

- Théorème de "De Morgan"

$$\overline{a+b} = \overline{a}\,.\,\overline{b}$$

$$\overline{a.b} = \overline{a} + \overline{b}$$

$$\overline{\sum_i x_i} = \prod_i \overline{x_i}$$

$$\overline{\prod_i x_i} = \sum_i \overline{x_i}$$

- Théorème du Consensus

$$a.x+b.\overline{x}+a.b = a.x+b.\overline{x}$$

$$(a+x)(b+\overline{x})(a+b)=(a+x)(b+\overline{x})$$

Exercice 1:

Démontrer la proposition suivante :

$$ABC + AB\overline{C} + A\overline{B}CD = AB + ACD$$

$$A\,B\,C + \overline{A}BC + A\overline{B}C + AB\overline{C} = BC + AC + AB$$

Donner la forme simplifiée de la fonction suivante :

$$F(A,B,C,D) = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + ABCD$$

Correction

$$A\,B\,C + \overline{A}BC + A\overline{B}C + AB\overline{C} =$$

$$ABC + \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} =$$

$$BC + \qquad AC \qquad + \quad AB$$

$$ABC + AB\overline{C} + A\overline{B}CD = AB(C + \overline{C}) + A\overline{B}CD$$

$$= AB + A\overline{B}CD$$

$$= A(B + \overline{B}(CD))$$

$$= A(B + CD)$$

$$= AB + ACD$$

Simplification par la table de Karnaugh

Description de la table de karnaugh

•La méthode consiste a mettre en évidence par une méthode graphique (un tableaux) tous les termes qui sont adjacents (qui ne différent que par l'état d'une seule variable).

•Un tableau de Karnaugh = table de vérité de 2^n cases avec un changement unique entre 2 cases voisines d'où des codes cycliques (Gray ou binaire réfléchi).

•La méthode peut s'appliquer aux fonctions logiques de 2,3,4,5 et 6 variables.

•Un tableau de Karnaugh comportent 2^n cases (N est le nombre de variables).

Description de la table de karnaugh

Règles de regroupement :

- groupe de 2^n cases : 1,2,4 ou 8
- en ligne, colonne, rectangle, carré, mais pas diagonale
- tous les 1, mais pas les 0 au moins une fois dans les groupements

Règles de minimisation de la fonction :

- rechercher les groupements en commençant par les cases qui n'ont qu'une seule façon de se grouper
- rechercher les groupements les plus grands
- les groupements doivent contenir au moins un 1 non utilisé par les autres groupements
- L'expression logique finale est la réunion (la somme) des groupements après simplification et élimination des variables qui changent d'état.

Description de la table de karnaugh

B \ A	0	1
0		
1		

Tableau à 2 variables

C \ AB	00	01	11	10
0				
1				

Tableaux à 3 variables

Tableaux de Karnaugh

f (a,c,d, ..,n) fonction logique à N entrées

sera représentée par une table à 2^N lignes
un tableau à 2^N cases

a b c	f(a,b,c)
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	0
1 1 1	1

a \ bc	00	01	11	10
0	0	1	0	0
1	1	0	1	0

f(a,b,c)

Code Gray ou
binaire réfléchi
=
1 seul changement
entre 2 codes
successifs

Exemple 1 : 3 variables

C \ AB	00	01	11	10
0			1	
1	1	1	1	1

$$F(A,B,C) = C + AB$$

Tableaux de Karnaugh

Exemple 2 : 4 variables

CD \ AB	00	01	11	10
00				1
01	1	1	1	1
11				
10		1		

$$F(A,B,C,D) = \overline{C}.D + A.\overline{B}.\overline{C} + \overline{A}.B.C.\overline{D}$$

Tableaux de Karnaugh

Exemple 3 : 4 variables

CD \ AB	00	01	11	10
00	1			1
01		1	1	1
11				1
10	1			1

$$F(A,B,C,D) = A\overline{B} + \overline{B}\,\overline{D} + B\overline{C}D$$

Tableaux de Karnaugh

Exemple 4 : 5 variables

CD \ AB	00	01	11	10
00	1			
01	1		1	
11	1		1	
10	1			

U = 0

CD \ AB	00	01	11	10
00	1			
01	1			1
11	1			1
10	1	1		

U= 1

$$F(A,B,C,D,U) = \overline{A}\,\overline{B} + A.B.D.\overline{U} + \overline{A}.C.\overline{D}.U + A.\overline{B}.D.U$$

Exercice

Trouver la forme simplifiée des fonctions à partir des deux tableaux ?

C \ AB	00	01	11	10
0		1	1	1
1	1		1	1

CD \ AB	00	01	11	10
00	1		1	1
01				
11				
10	1	1	1	1

Logique multi-niveaux

On peut généraliser l'algèbre binaire à plus de 2 niveaux

a \ b	0	1	Z	X
0	0	X	0	X
1	X	1	1	X
Z	0	1	Z	X
X	X	X	X	X

0 logique

1 logique

Z déconnecté

X inconnu

Logique multi-niveaux

•Pour les cas impossibles ou interdites

il faut mettre un X dans la T.V .

•Les cas impossibles sont représentées

aussi par des X dans la table de karnaugh

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

A	B	C	D		S
0	0	0	0		0
0	0	0	1		0
0	0	1	0		0
0	0	1	1		1
0	1	0	0		0
0	1	0	1		1
0	1	1	0		1
0	1	1	1		1
1	0	0	0		0
1	0	0	1		X
1	0	1	0		1
1	0	1	1		X
1	1	0	0		1
1	1	0	1		X
1	1	1	0		1
1	1	1	1		X

Tableaux de Karnaugh

- Il est possible d'utiliser les X dans des regroupements :
  - Soit les prendre comme étant des 1
  - Ou les prendre comme étant des 0
- Il ne faut pas former des regroupement qui contient uniquement des X

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

AB

Tableaux de Karnaugh

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD$$

Tableaux de Karnaugh

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD + BD$$

Tableaux de Karnaugh

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD + BD + AC$$

Tableaux de Karnaugh

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD + BD + AC + BC$$

Exercice 1

Trouver la fonction logique simplifiée à partir de la table suivante ?

CD \ AB	00	01	11	10
00		1	X	
01	1	X		1
11	1		X	1
10	X		1	X

Tableaux de Karnaugh

Exercice 2
Simplifier la fonction F:

$$F = \overline{a}bc + \overline{a}b\overline{c} + ab\overline{c} + cab + a\overline{b}\,\overline{c}$$

1) Par la méthode analytique
2) Par un tableau de **Karnaugh**

Tableaux de Karnaugh

Exercice 4 :

- A l’université, un distributeur automatique de boissons chaudes permet de distribuer du café ou du thé, avec ou sans lait, ou du lait seul.

- Trois boutons permettent de commander le distributeur : « café », « thé », « lait ». Pour obtenir l'une de ces boissons seule, il suffit d'appuyer sur le bouton correspondant. Pour obtenir une boisson avec lait, il faut appuyer en même temps sur le bouton correspondant à la boisson choisie et sur le bouton « lait ».

- De plus, le distributeur ne fonctionne que si un jeton a préalablement été introduit dans la fente de l'appareil. Une fausse manœuvre après introduction du jeton (par exemple, appui simultané sur « café » et « thé ») provoque la restitution du jeton. Le lait étant gratuit, le jeton est également restitué si du lait seul est choisi.

- Calculer et simplifier les fonctions de restitution du jeton, J, de distribution du café, C, du thé T, et du lait, L.

- On notera que la fonction de restitution du jeton peut indifféremment être active ou non lorsque aucun jeton n'est introduit dans l'appareil.

Tableaux de Karnaugh

Correction

Table de vérité de C, T, L et J :

c	t	l	j	C	T	L	J
0	0	0	0	0	0	0	-
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	-
0	0	1	1	0	0	1	1
0	1	0	0	0	0	0	-
0	1	0	1	0	1	0	0
0	1	1	0	0	0	0	-
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	-
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	-
1	0	1	1	1	0	1	0
1	1	0	0	0	0	0	-
1	1	0	1	0	0	0	1
1	1	1	0	0	0	0	-
1	1	1	1	0	0	0	1

Soient c, t, l, j les variables logiques correspondant aux propositions suivantes :

- c =1⇔ le bouton « café » est enfoncé,
- t = 1⇔ le bouton « thé » est enfoncé,
- l =1⇔ le bouton « lait » est enfoncé,
- j = 1⇔ un jeton a été introduit dans la fente de l'appareil.

$$L = \bar{c}\bar{t}lj + \bar{c}tlj + c\bar{t}lj = (\bar{c} + \bar{t})lj$$

$$C = c\bar{t}\bar{l}j + \bar{c}tlj = c\bar{t}j$$

$$T = \bar{c}t\bar{l}j + \bar{c}tlj = \bar{c}tj$$

Après simplification par diagramme de Karnaugh, en utilisant les états indifférents on obtient

$$J = ct + \bar{c}\bar{t}l$$

Réalisation en électronique

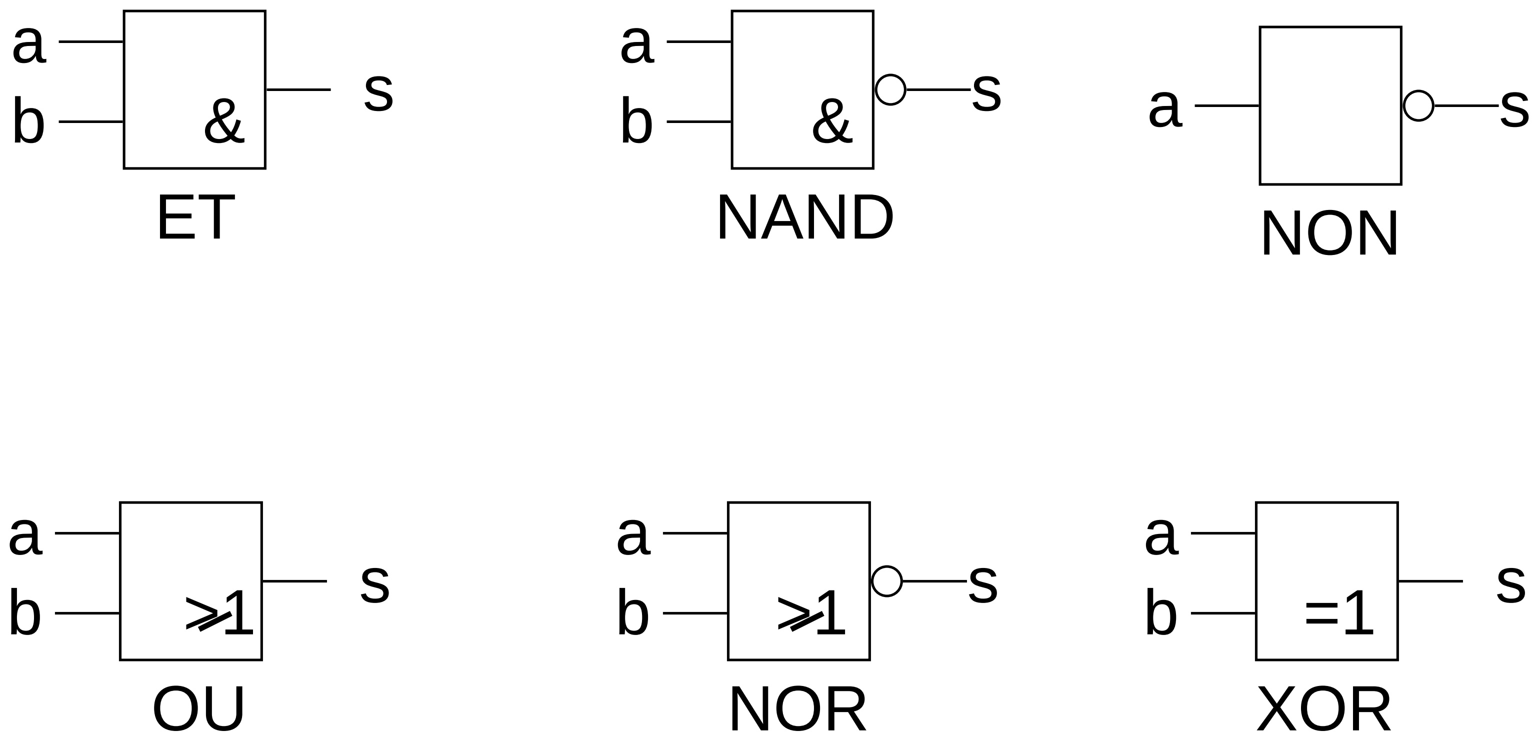
0/1 représentés par des tensions, courants, charges, fréquences, ...

Classiquement TENSIONS : Niveau haut = H (le plus positif)
Niveau bas = L (B) (le plus négatif)

Association d'une information binaire à un niveau :

Convention positive	H $\longrightarrow$ 1
(ou logique positive)	L $\longrightarrow$ 0
Convention négative	H $\longrightarrow$ 0
(ou logique négative)	L $\longrightarrow$ 1

Représentation graphique : Norme française



Représentation graphique :
Norme américaine

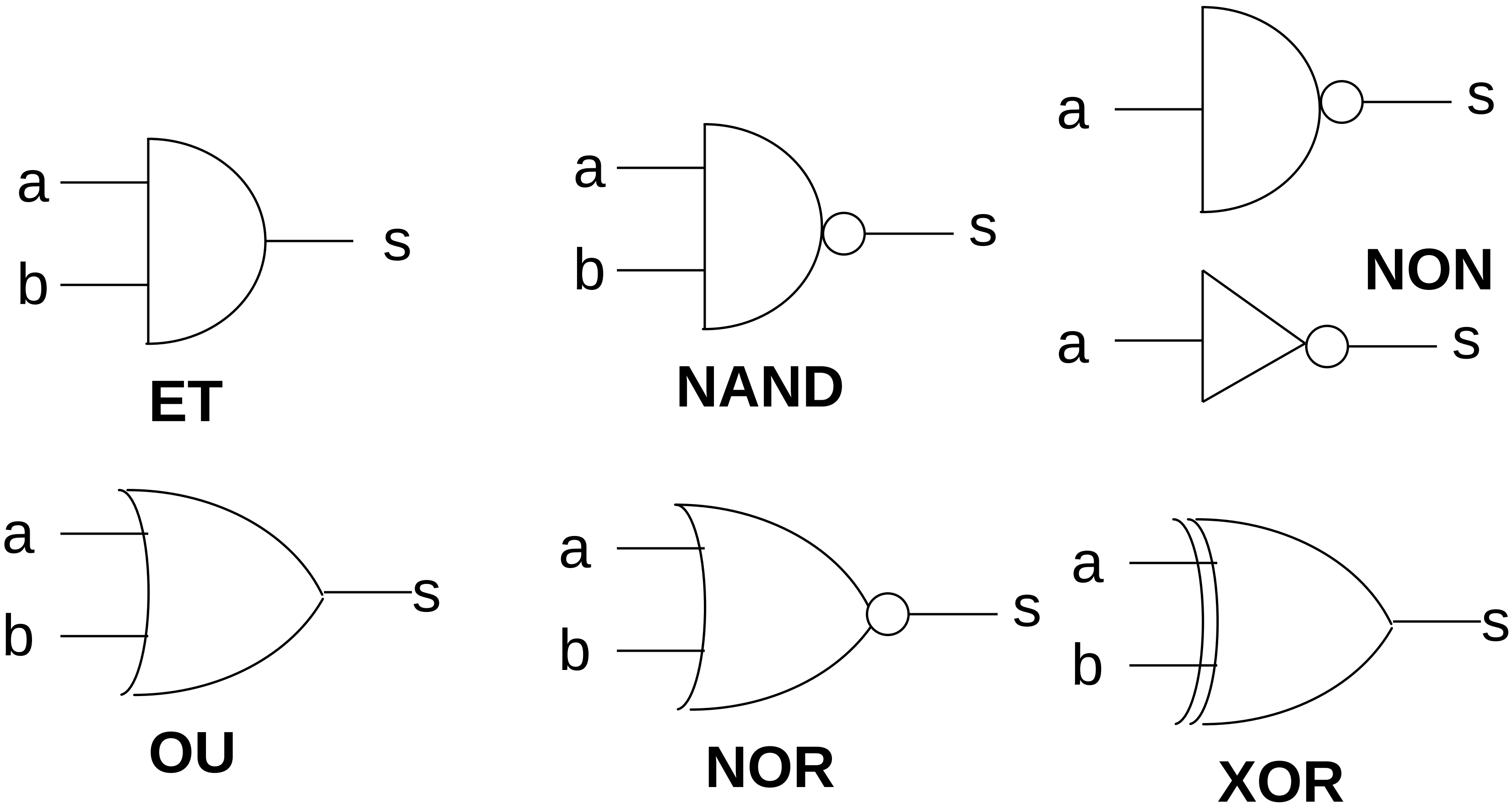
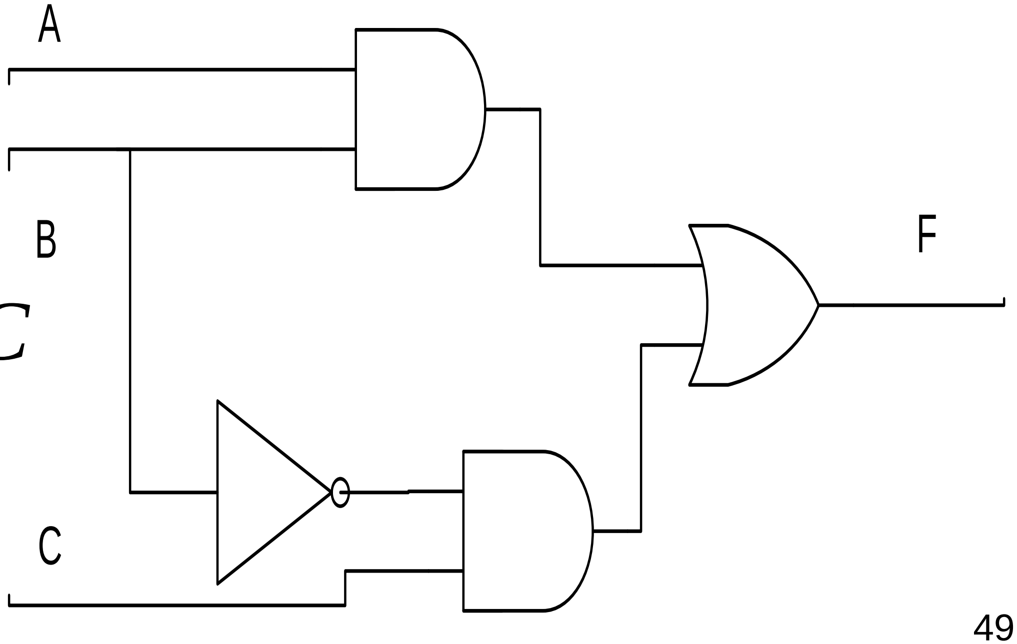


Schéma d'un circuit logique (Logigramme)

- C'est la traduction de la fonction logique en un schéma électronique.
- Le principe consiste à remplacer chaque opérateur logique par la porte logique qui lui correspond.

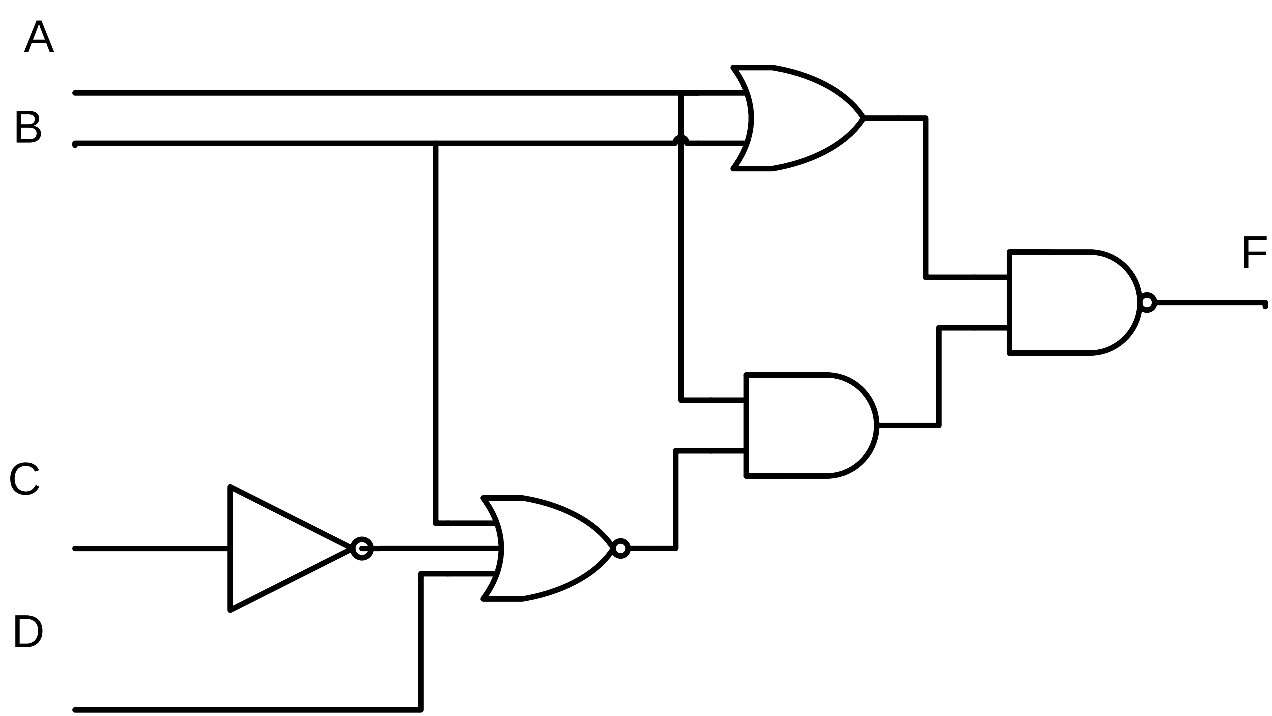
Exemple 1

$$F(A,B,C) = A.B + \overline{B}.C$$



Exemple 2:

$$F(A,B,C,D) = \overline{(A+B)\,.\,\overline{(B+\overline{C}+D)}\,.\,A}$$



Exercice 1

- Donner le logigramme des fonctions suivantes :

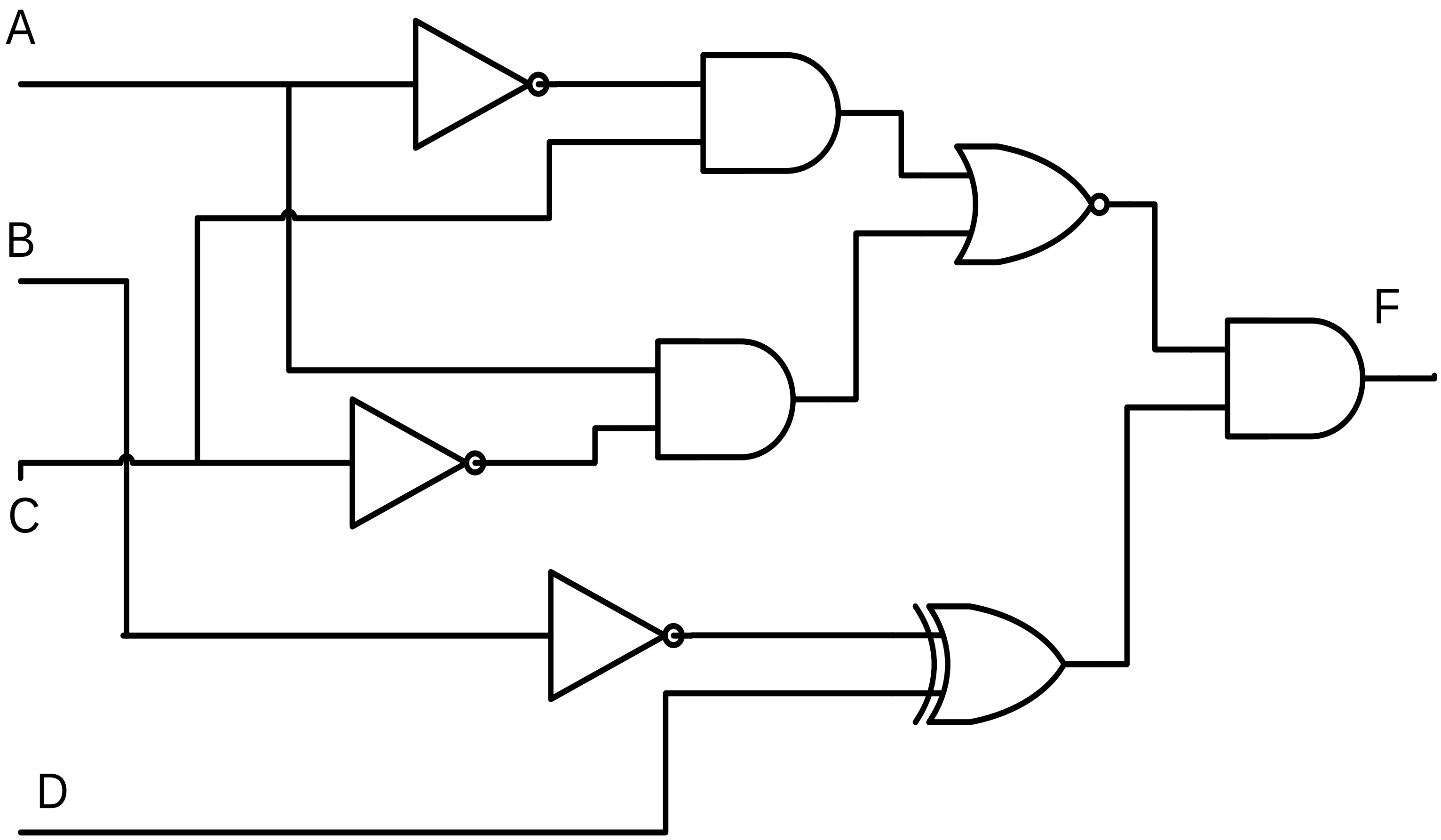
$$F(A,B) = \overline{A}.B + A.\overline{B}$$

$$F(A,B,C) = (A+B).(\overline{A}+C).(B+\overline{C})$$

$$F(A,B,C) = (\overline{A\,.\,B})\,.\,(C+B) + A.\overline{B}.C$$

Circuits logiques

Exercice 2 : Donner l'équation de F ?



Exercice 3 : Soit la fonction **F**

$$F(A,B,C,D) = (A+\overline{B}+C+D)(\overline{A}+\overline{B}+C+D)(\overline{A}+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+\overline{C}+\overline{D})$$
$$(A+\overline{B}+\overline{C}+D)(\overline{A}+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+D)$$

1) Simplifier la fonction F par la méthode des diagrammes de Karnaugh

2) Donner les schémas logiques ou logigrammes de la fonction simplifiée utilisant :
   - Logigramme 1 : avec uniquement des portes NON ET
   - Logigramme 2 : des portes ET, OU, et des inverseurs,

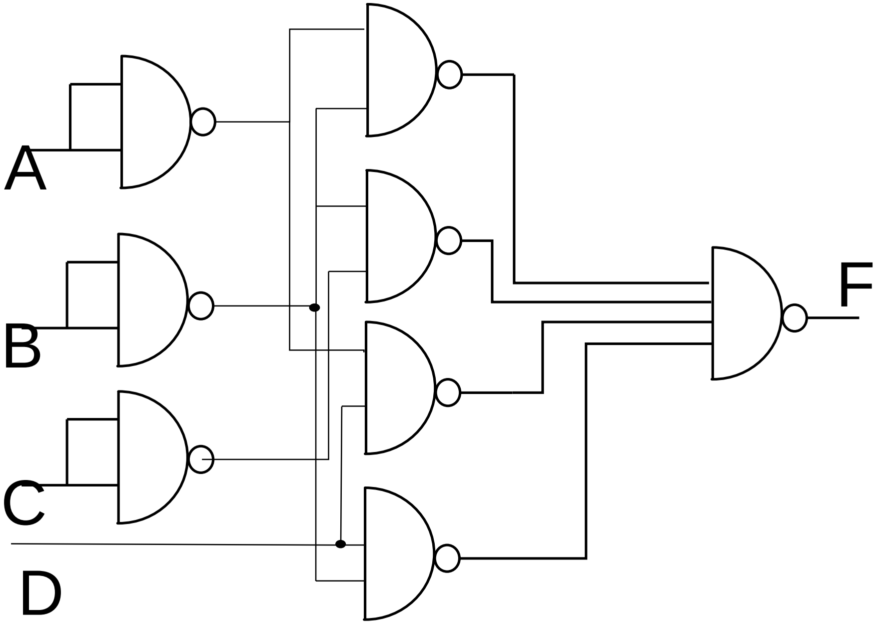
Circuits logiques

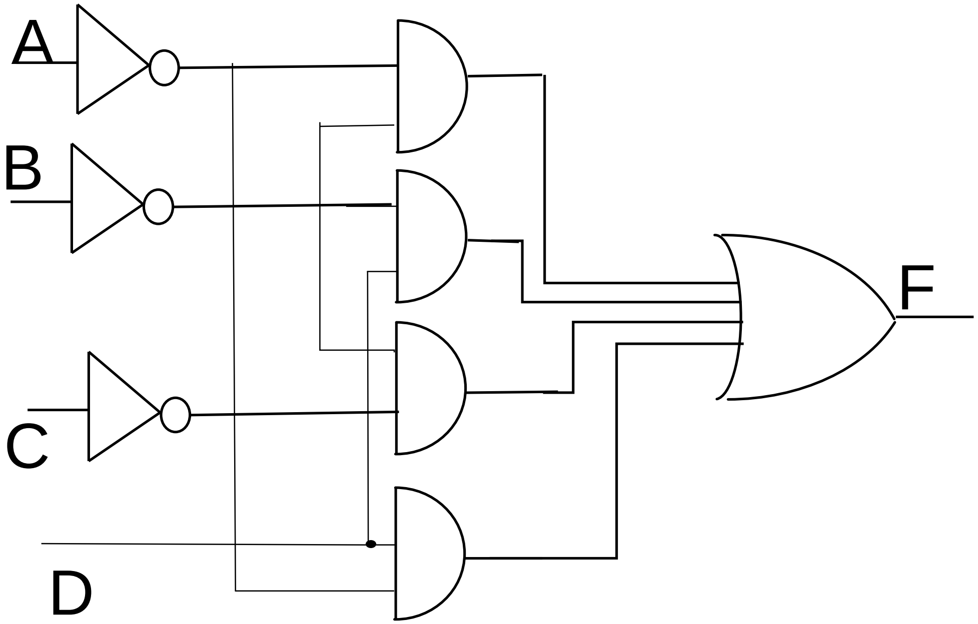
Exercice 3 : Soit la fonction **F correction**

$$F(A,B,C,D) = (A+\overline{B}+C+D)(\overline{A}+\overline{B}+C+D)\,(\overline{A}+\overline{B}+C+\overline{D})\,(\overline{A}+\overline{B}+\overline{C}+\overline{D})$$
$$(A+\overline{B}+\overline{C}+D)\,(\overline{A}+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\underline{D})$$

$$F(A,B,C,D) = \overline{B}\,\overline{C} + \overline{B}D + \overline{A}\,\overline{B} + \overline{A}B$$

$$F(A,B,C,D) = \overline{\overline{\overline{B}\,\overline{C} + \overline{B}D + \overline{A}\,\overline{B} + \overline{A}B}} = \overline{\overline{\overline{B}\,\overline{C}}.\overline{\overline{B}D}.\overline{\overline{A}\,\overline{B}}.\overline{\overline{A}B}}$$





Plan

1) Calcul propositionnel, l'algèbre de Boole et la logique combinatoire

2) **Circuits combinatoires**

3) Représentation et codage des données

Les circuits combinatoires

Moyens physiques de réalisation des fonctions logiques

Problème
(cahier des charges) → Fonctions logiques

↓

Fonctions logiques simplifiées

Attention : critères pas toujours compatibles

coût / vitesse / encombrement / fiabilité ?

↓

Réalisation Technologique

Les circuits combinatoires

Objectifs

- Apprendre la structure de quelques circuits combinatoires souvent utilisés (demi additionneur , additionneur complet,……..).

- Apprendre comment utiliser des circuits combinatoires pour concevoir d'autres circuits plus complexes.

Circuits combinatoires

- Un circuit combinatoire est un circuit numérique dont les sorties dépendent uniquement des entrées.
- $S_i = F(E_i)$
- $S_i = F(E_1, E_2, \ldots, E_n)$

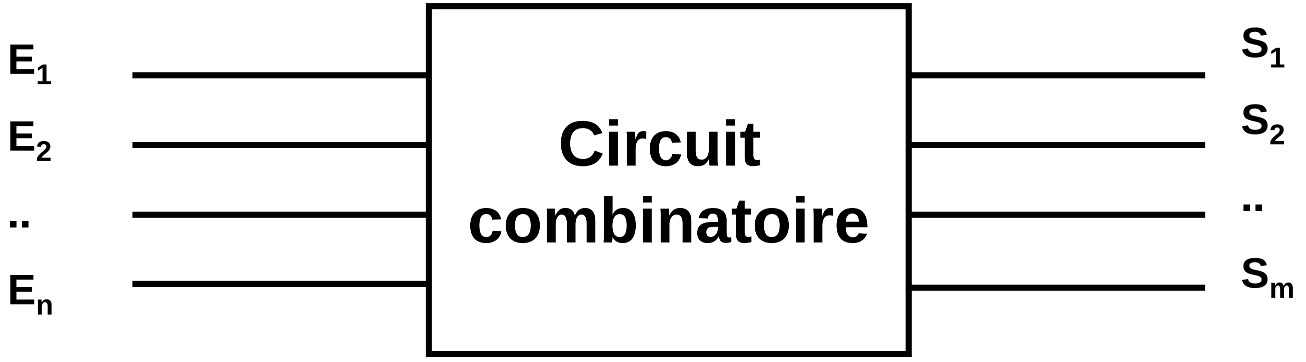


Schéma Bloc

- C'est possible d'utiliser des circuits combinatoires pour réaliser d'autres circuits plus complexes.

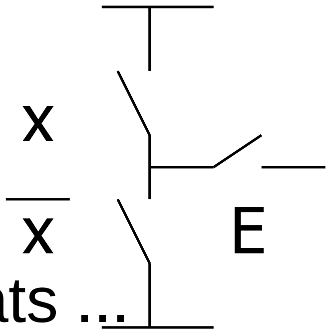
Composants combinatoires

- Multiplexeur / démultiplexeur
- Codeurs / Décodeurs
- Transcodeurs
- Comparateurs / Détection d'erreurs
- Circuits arithmétiques (add, ALU, mult)

Portes intégrées

Options technologiques : familles logiques

(TTL,CMOS, BiCMOS, ECL ..)



Entrées : classique, triggée

Sorties : collecteur (drain) ouvert, sortie 3 états ...

Remarque 1 :

10 entrées = 2^{10} fonctions possibles

$\Longrightarrow$ Choix des meilleures fonctions

Portes intégrées

Remarque 2:

Problème du nombre de boîtiers pour réaliser une fonction logique ⟶ INTEGRATION

- SSI (*small scale integration*) petite : inférieur à 12 portes
- MSI (*medium*) moyenne : 12 à 99
- LSI (*large*) grande : 100 à 9999
- VLSI (*very large*) très grande : 10 000 à 99 999
- ULSI (*ultra large*) ultra grande : 100 000 et plus

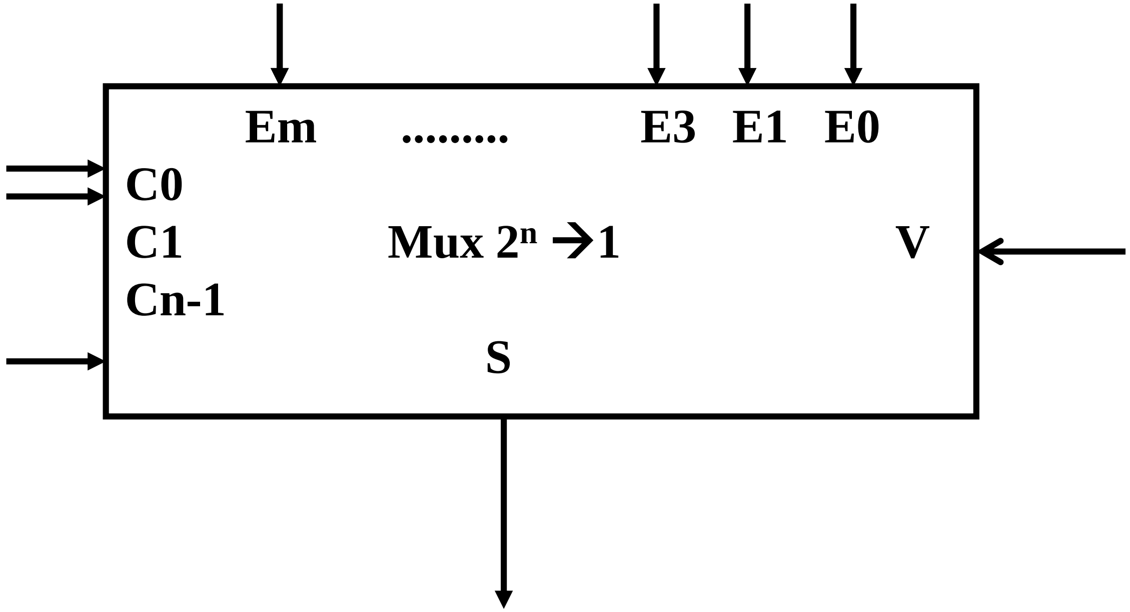
Remarque 3:

Une manière d'augmenter la puissance de traitement est de construire des CI dédiés à une application

(**ASIC** pour *Application Specific Integrated Circuit*)

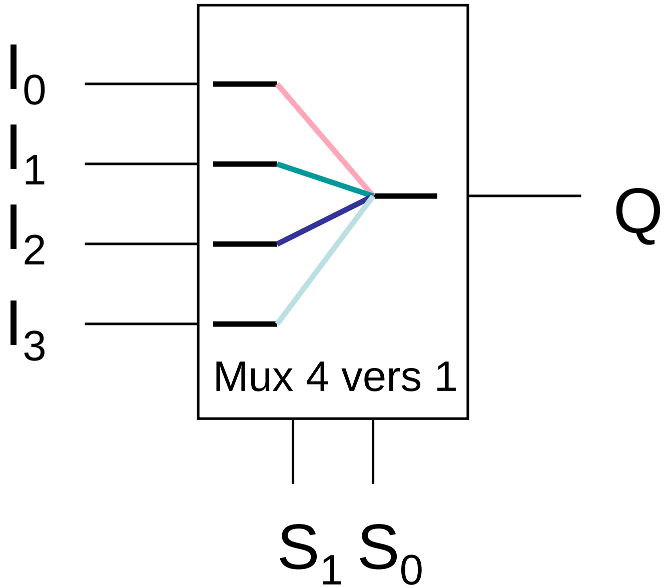
Multiplexeur

- Un multiplexeur est un circuit combinatoire qui permet de sélectionner une information (1 bit) parmi 2^n valeurs en entrée.
- Il possède :
  - 2^n entrées d'information
  - Une seule sortie
  - N entrées de sélection (commandes)



Multiplexeur 4 → 1

Sélection d'une voie parmi 2^N par N bits de commande



Si $(S_1S_0)_2 = (0)_{10}$ alors $Q = I_0$

$$Q = \overline{S_1}.\overline{S_0}.I_O$$

Si $(S_1S_0)_2 = (1)_{10}$ alors $Q = I_1$

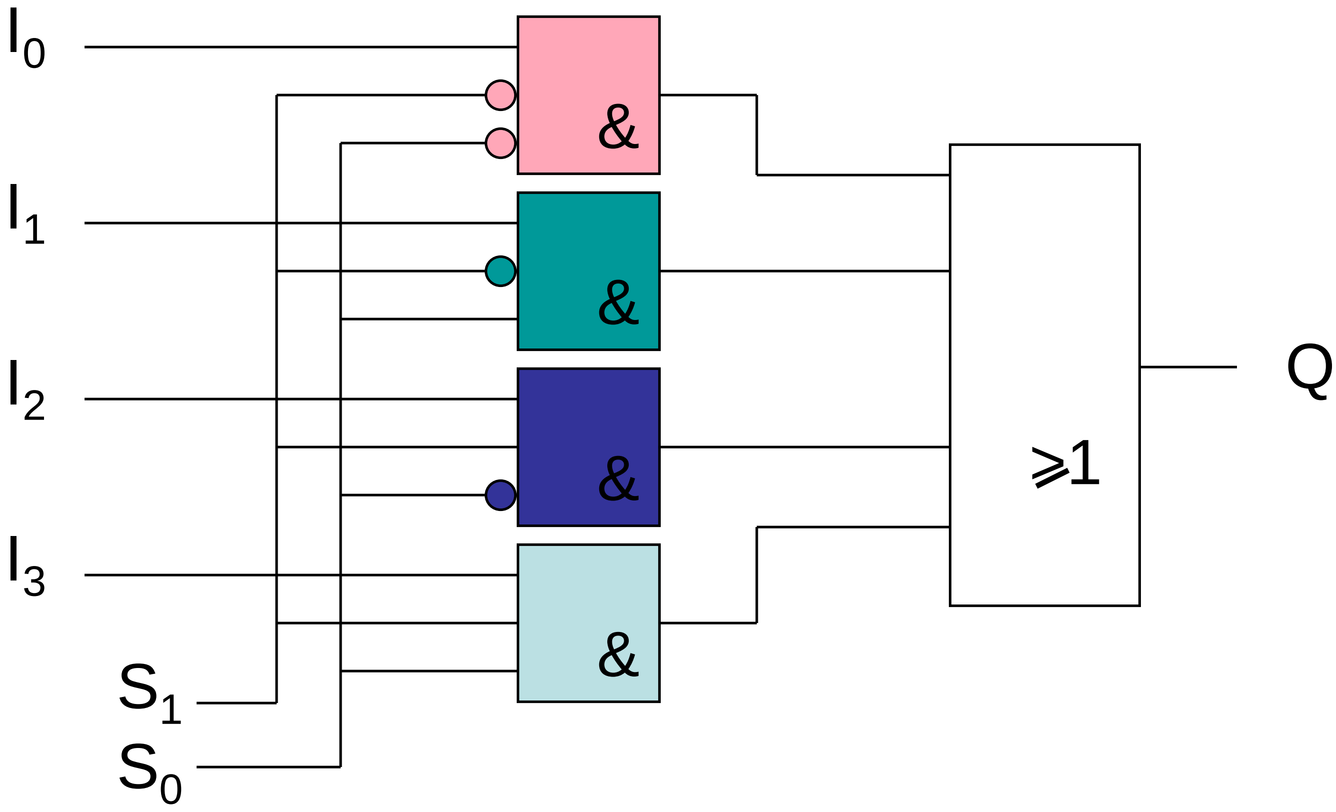
$$Q = \overline{S_1}.S_0.I_1$$

S1	S0		Q
0	0		I_0
0	1		I_1
1	0		I_2
1	1		I_3

$$Q = \overline{S_1}.\overline{S_0}.I_0 + \overline{S_1}.S_0.I_1 + S_1.\overline{S_0}.I_2 + S_1.S_0.I_3$$

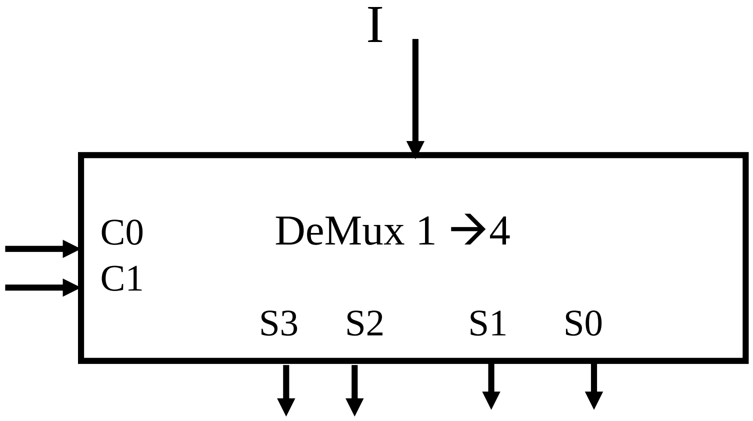
Multiplexeur (logigramme)

$$Q = \overline{S_1}.\overline{S_0}.I_0 + \overline{S_1}.S_0.I_1 + S_1.\overline{S_0}.I_2 + S_1.S_0.I_3$$

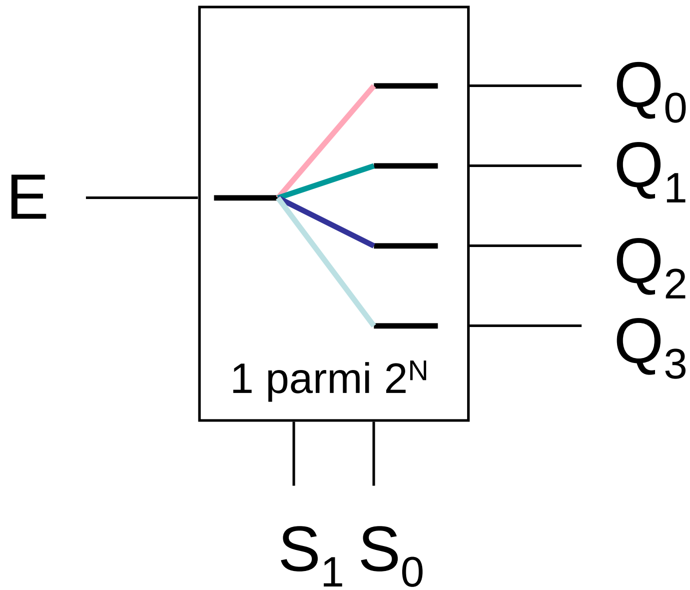


Démultiplexeur

- Il joue le rôle inverse d'un multiplexeurs, il permet de faire passer une information dans l'une des sorties selon les valeurs des entrées de commandes.
- Il possède :
  - une seule entrée
  - 2^n sorties
  - N entrées de sélection (commandes)



Démultiplexeur : 1 parmi 2^n



Q_0 = E si $(S_1S_0)_2=0$

0 sinon

Q_1 = E si $(S_1S_0)_2=1$

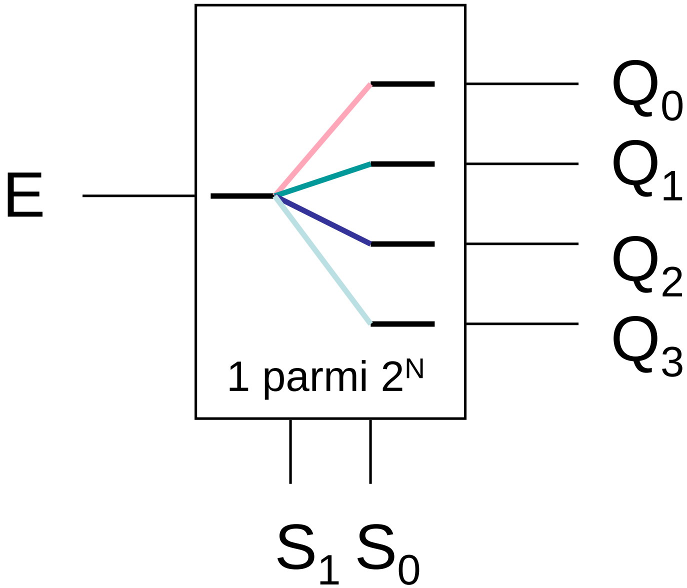
0 sinon

Remarque : E peut ne pas être «disponible»

Sortie sélectionnée = 1 les autres 0

ou Sortie sélectionnée = 0 les autres 1

Démultiplexeur : 1 →4



S₁	S₀		Q3	Q2	Q1	Q0
0	0		0	0	0	E
0	1		0	0	E	0
1	0		0	E	0	0
1	1		E	0	0	0

$$Q0 = \overline{S1}.\overline{S0}.(E)$$

$$Q1 = \overline{S1}.S0.(E)$$

$$Q2 = S1.\overline{S0}.(E)$$

$$Q3 = S1.S0.(E)$$

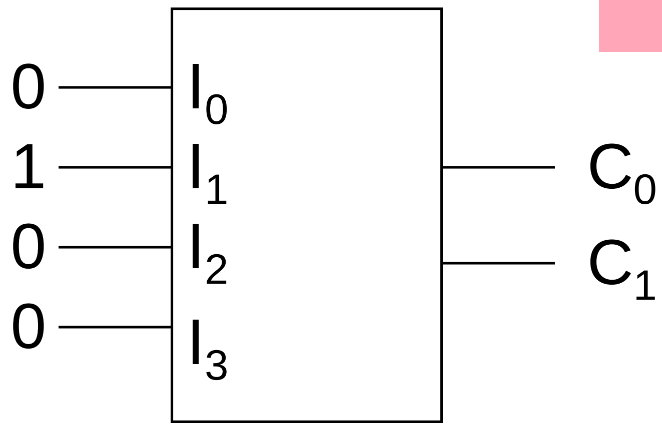
Codeur (ou Encodeur)

Faire correspondre un mot code à un symbole

2^N entrées

N sorties

1 entrée parmi 2^N



Le code de l'entrée

Mot

Code

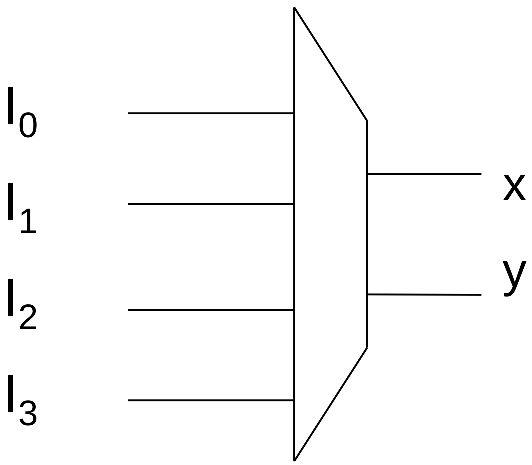
Traduit le rang de l'entrée active en un code binaire

Exemple : Clavier / Scan code

Caractère / Code ASCII

L'encodeur binaire (4→2)

I_0	I_1	I_2	I_3		x	y
0	0	0	0		0	0
1	x	x	x		0	0
0	1	x	x		0	1
0	0	1	x		1	0
0	0	0	1		1	1

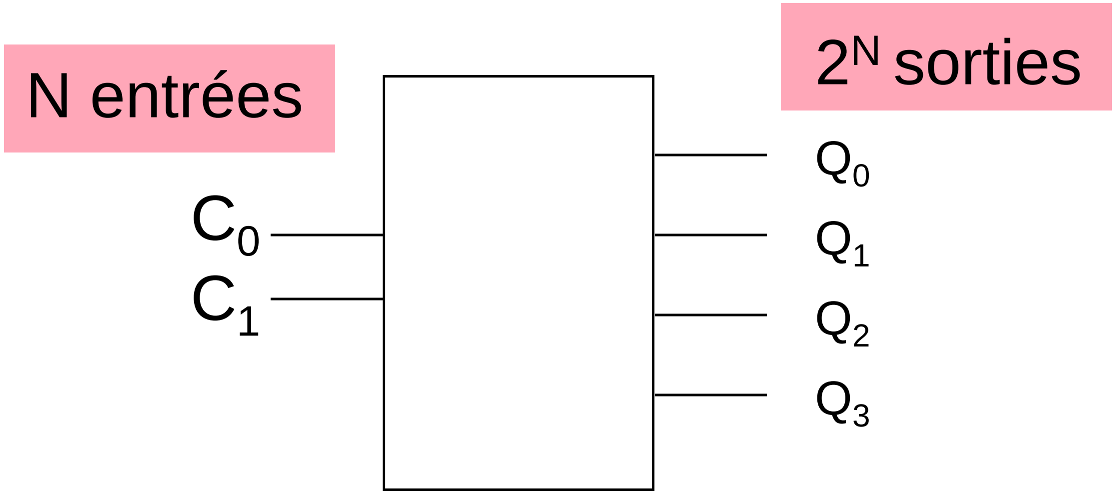


$$X = \overline{I0}.\overline{I1}.(I2 + I3)$$

$$Y = \overline{I0}.(I1 + .\overline{I2}.I3)$$

Le décodeur binaire

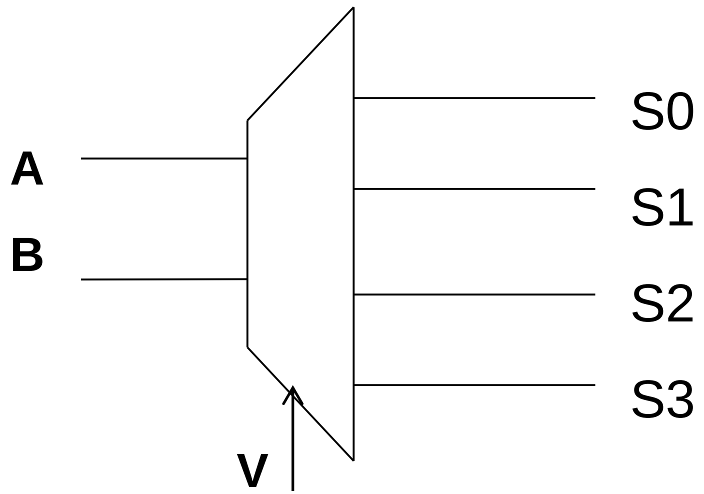
- C'est un circuit combinatoire qui est constitué de :
  - N : entrées de données
  - 2^n sorties
  - Pour chaque combinaison en entrée une seule sortie est active à la fois



Active la ligne de sortie correspondant au code binaire présent en entrée

Décodeur 2→4

V	A	B		S0	S1	S2	S3
0	X	X		0	0	0	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	0	1



$$S_0 = (\overline{A}.\overline{B}).V$$

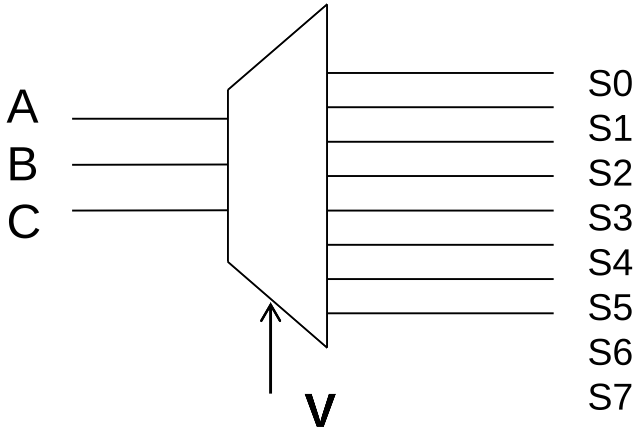
$$S_1 = (\overline{A}.B).V$$

$$S_2 = (A.\overline{B}).V$$

$$S_3 = (A.B).V$$

Décodeur 3→8

A	B	C		S0	S1	S2	S3	S4	S5	S6	S7
0	0	0		1	0	0	0	0	0	0	0
0	0	1		0	1	0	0	0	0	0	0
0	1	0		0	0	1	0	0	0	0	0
0	1	1		0	0	0	1	0	0	0	0
1	0	0		0	0	0	0	1	0	0	0
1	0	1		0	0	0	0	0	1	0	0
1	1	0		0	0	0	0	0	0	1	0
1	1	1		0	0	0	0	0	0	0	1



$$S_0 = \overline{A}.\overline{B}.\overline{C}$$

$$S_1 = \overline{A}.\overline{B}.C$$

$$S_2 = \overline{A}.B.\overline{C}$$

$$S_3 = \overline{A}.B.C$$

$$S_4 = A.\overline{B}.\overline{C}$$

$$S_5 = A.\overline{B}.C$$

$$S_6 = A.B.\overline{C}$$

$$S_7 = A.B.C$$

Remarque :

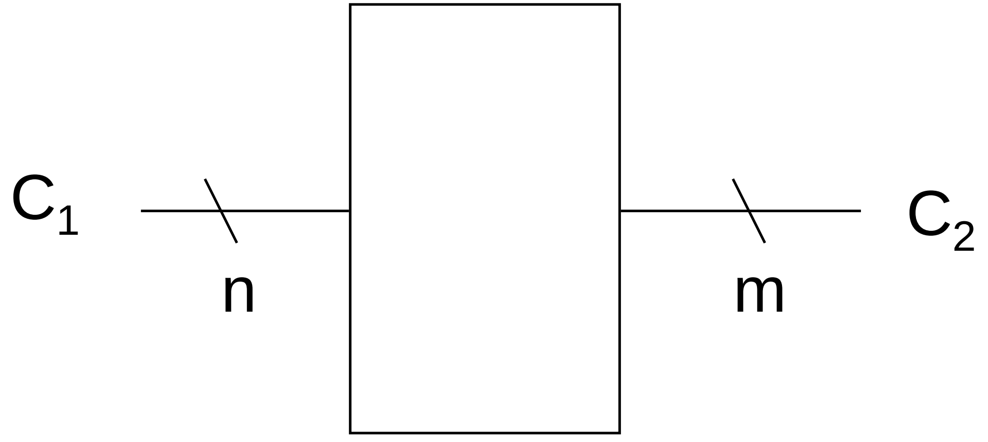
Multiplexeur ⇔ Démultiplexeur

Codeur ⇔ Décodeur

Transcodeur

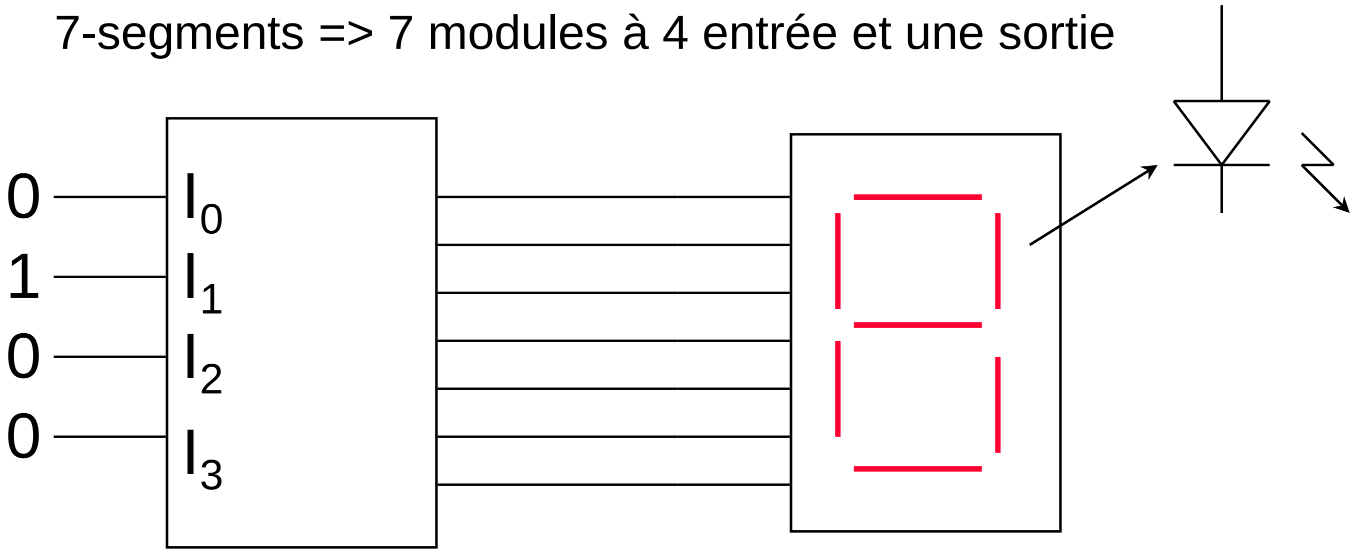
C'est un circuit combinatoire qui permet de transformer un code X (sur n bits) en entrée en un code Y (sur m bits) en sortie.

Passage d'un code C_1 à un code C_2



Transcodeur : exemple

7-segments => 7 modules à 4 entrée et une sortie



Code binaire 0 à 9

Configuration alimentation des diodes (ou LCD)

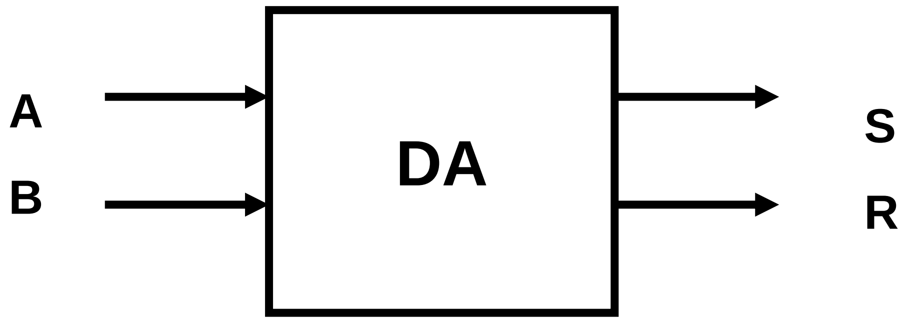
Exemples de code :

Binaire, binaire réfléchi, 7-segments, BCD, …

Demi Additionneur

- Le demi additionneur est un circuit combinatoire qui permet de réaliser la somme arithmétique de deux nombres A et B chacun sur un bit.

- A la sotie on va avoir la somme S et la retenu R (Carry).



Pour trouver la structure (le schéma) de ce circuit on doit en premier dresser sa table de vérité

Demi Additionneur

- En binaire l'addition sur un seul bit se fait de la manière suivante:

$$\begin{cases} 0+0 = 00 \\ 0+1 = 01 \\ 1+0 = 01 \\ 1+1 = 10 \end{cases}$$

•La table de vérité associée :

A	B		R	S
0	0		0	0
0	1		0	1
1	0		0	1
1	1		1	0

De la table de vérité on trouve :

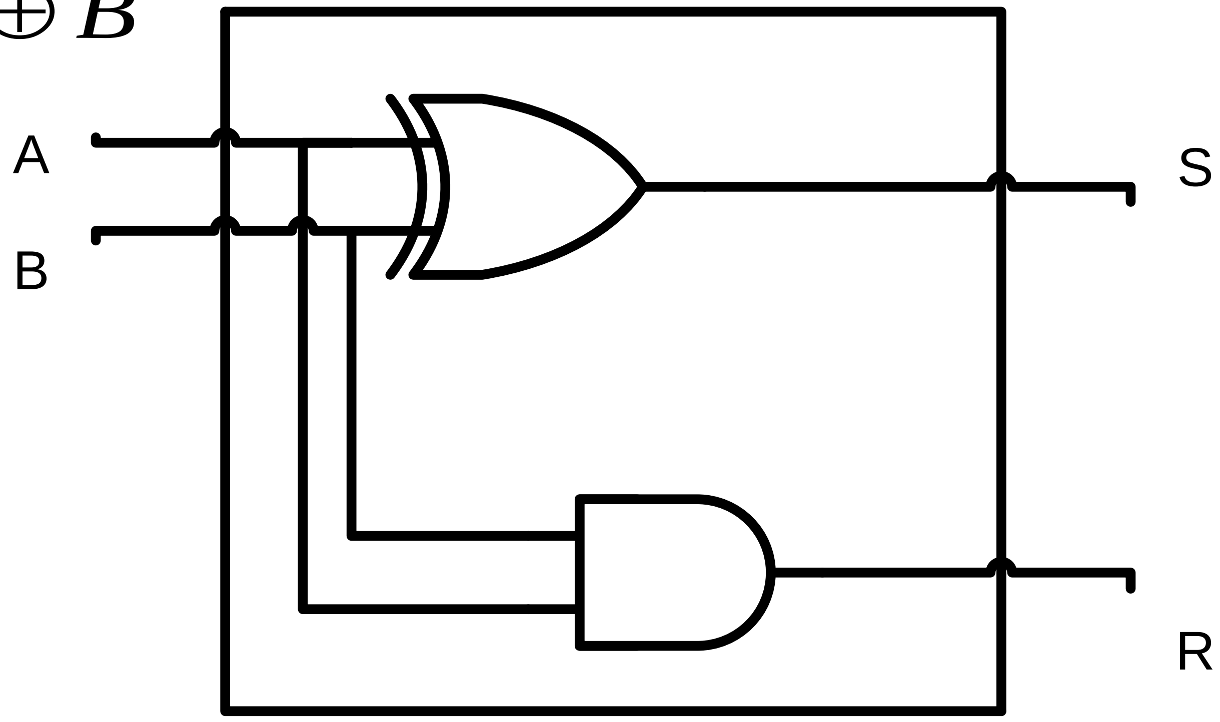
$$R = A.B$$

$$S = \overline{A}.B + A.\overline{B} = A \oplus B$$

Demi Additionneur

$$R = A.B$$

$$S = A \oplus B$$



Logigramme Demi-Additionneur

Additionneur complet

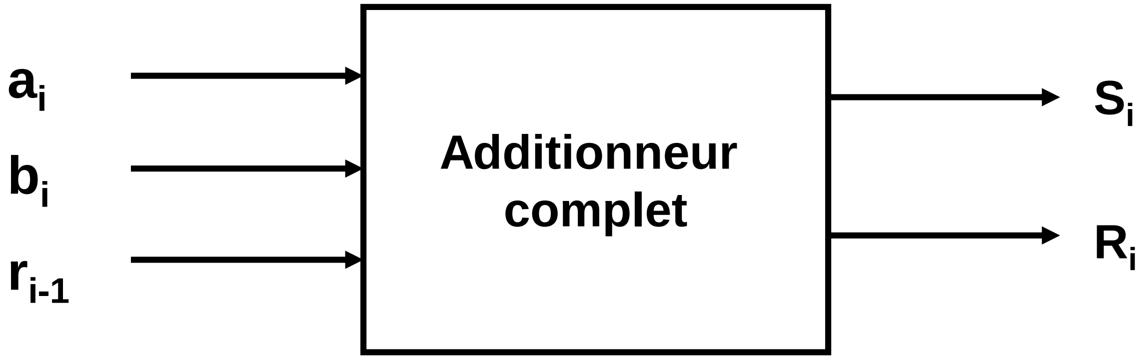
- En binaire lorsque on fait une addition il faut tenir en compte de la retenue entrante.

r_4	r_3	r_2	r_1	$r_0 = 0$
	a_4	a_3	a_2	a_1
$+$	b_4	b_3	b_2	b_1
r_4	s_4	s_3	s_2	s_1

$$
\begin{array}{rr}
 & r_{i-1} \\
 & a_i \\
+ & b_i \\
\hline
r_i & s_i
\end{array}
$$

Additionneur complet 1 bit

- L'additionneur complet un bit possède 3 entrées :
  - a_i : le premier nombre sur un bit.
  - b_i : le deuxième nombre sur un bit.
  - r_{i-1} : le retenue entrante sur un bit.
- Il possède deux sorties :
  - S_i : la somme
  - R_i la retenue sortante



Additionneur complet 1 bit

Table de vérité d'un additionneur complet sur 1 bit

a_i	b_i	r_{i-1}		r_i	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

$$S_i = \overline{A}_i.\overline{B}_i.R_{i-1} + \overline{A}_i.B_i.\overline{R}_{i-1} + A_i.\overline{B}_i.\overline{R}_{i-1} + A_i.B_i.R_{i-1}$$

$$R_i = \overline{A}_i B_i R_{i-1} + A_i \overline{B}_i R_{i-1} + A_i B_i \overline{R}_{i-1} + A_i B_i R_{i-1}$$

Additionneur complet 1 bit

Si on veut simplifier les équations on obtient :

$$S_i = \overline{A_i}.\overline{B_i}.R_{i-1} + \overline{A_i}.B_i.\overline{R_{i-1}} + A_i.\overline{B_i}.\overline{R_{i-1}} + A_i.B_i.R_{i-1}$$

$$S_i = \overline{A_i}.(\overline{B_i}.R_{i-1} + B_i.\overline{R_{i-1}}) + A_i.(\overline{B_i}.\overline{R_{i-1}} + B_i.R_{i-1})$$

$$S_i = \overline{A_i}(B_i \oplus R_{i-1}) + A_i.(\overline{B_i \oplus R_{i-1}})$$

$$S_i = A_i \oplus B_i \oplus R_{i-1}$$

$$R_i = \overline{A_i}B_iR_{i-1} + A_i\overline{B_i}R_{i-1} + A_iB_i\overline{R_{i-1}} + A_iB_iR_{i-1}$$

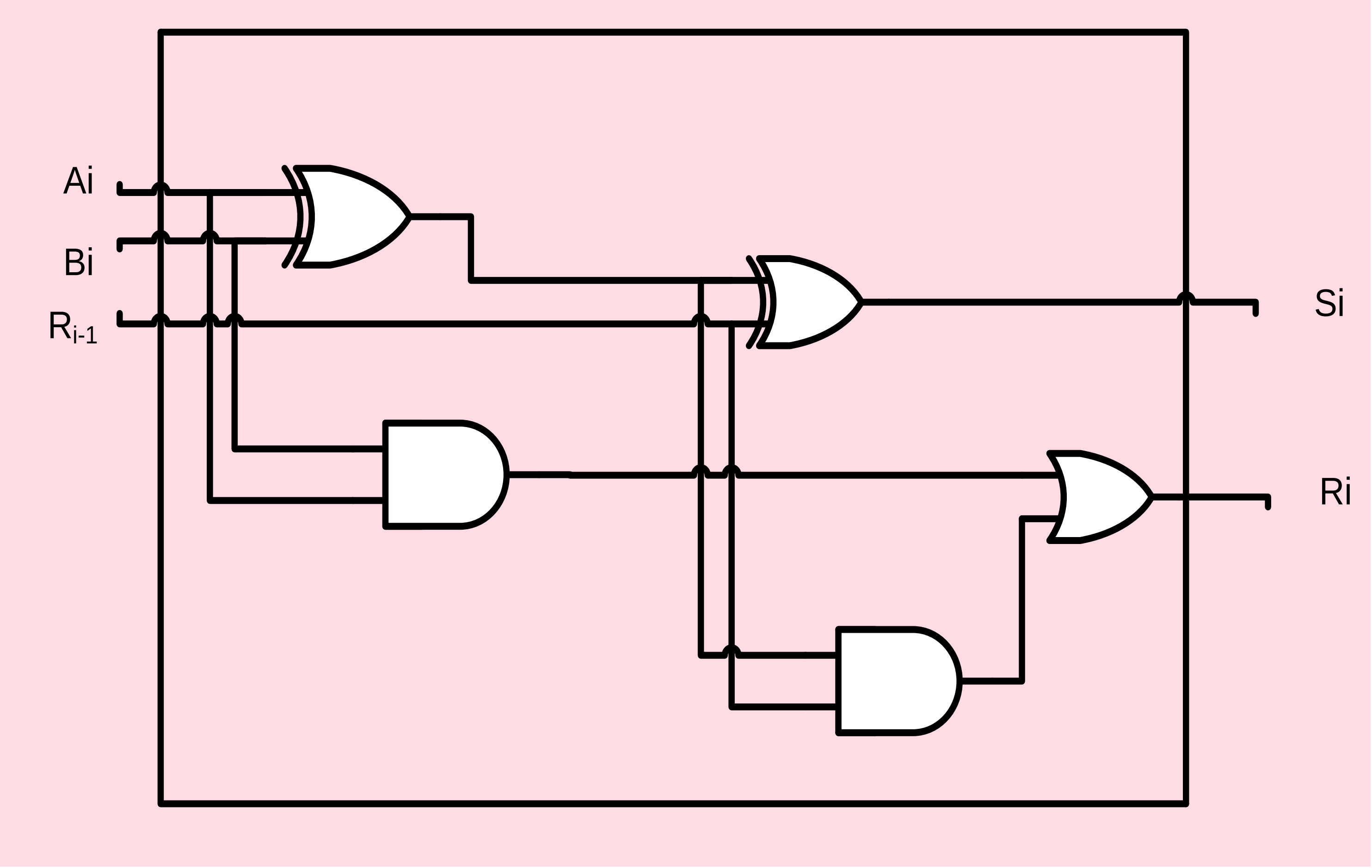
$$R_i = R_{i-1}.(\overline{A_i}.B_i + A_i.\overline{B_i}) + A_iB_i(\overline{R_{i-1}} +_i R_{i-1})$$

$$R_i = R_{i-1}.(A_i \oplus B_i) + A_iB_i$$

Schéma d'un additionneur complet

$$R_i = A_i.B_i + R_{i-1}.(B_i \oplus A_i)$$

$$S_i = A_i \oplus B_i \oplus R_{i-1}$$



Additionneur sur 4 bits

- Un additionneur sur 4 bits est un circuit qui permet de faire l'addition de deux nombres A et B de 4 bits chacun
  - $A(a_3a_2a_1a_0)$
  - $B(b_3b_2b_1b_0)$

  En plus il tient en compte de la retenu entrante

- En sortie on va avoir le résultat sur 4 bits ainsi que la retenu (5 bits en sortie)

- Donc au total le circuit possède 9 entrées et 5 sorties.

- Avec 9 entrées on a $2^9=512$ combinaisons !!!!!! Comment faire pour représenter la table de vérité ?????

- Il faut trouver une solution plus facile et plus efficace pour concevoir ce circuit ?

Additionneur sur 4 bits

•Lorsque on fait l'addition en binaire , on additionne bit par bit en commençant à partir du poids fiable et à chaque fois on propage la retenue sortante au bit du rang supérieur.

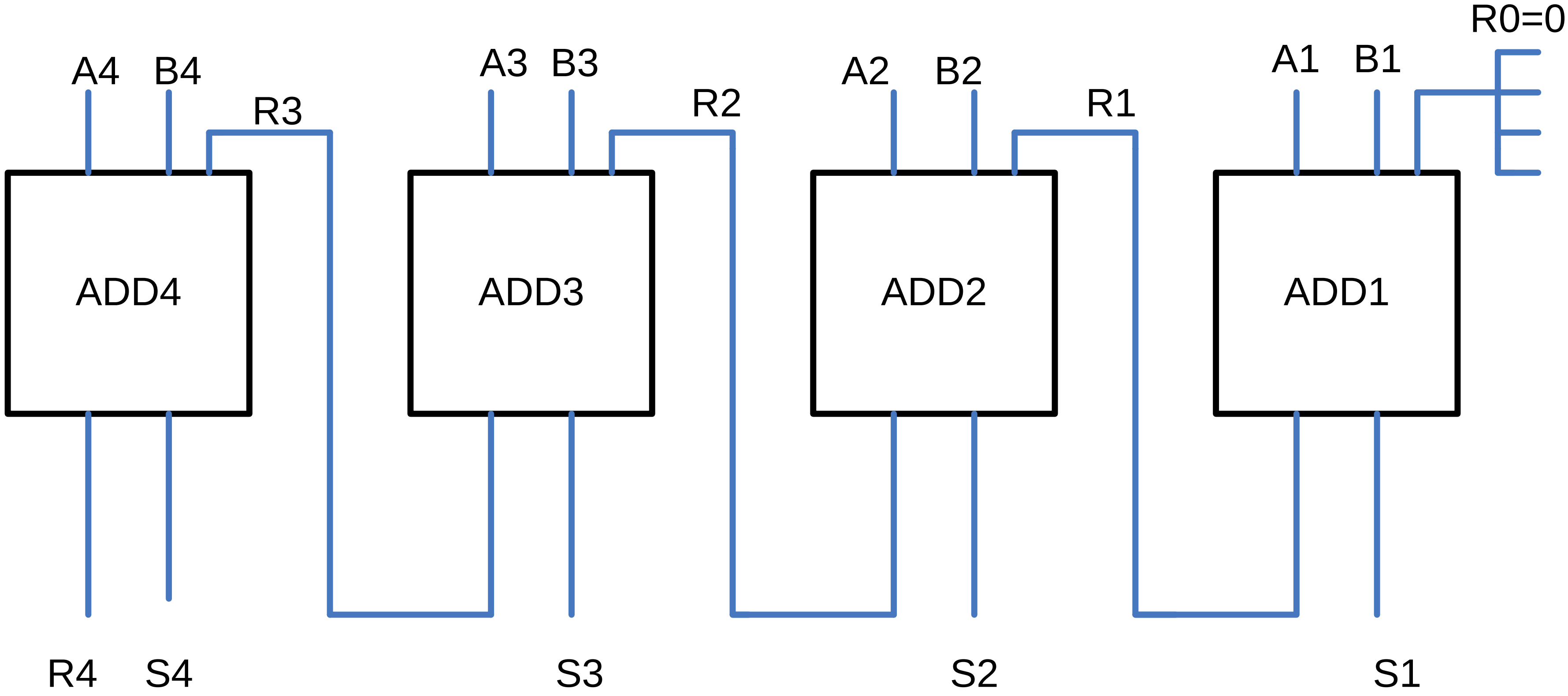
L'addition sur un bit peut se faire par un additionneur complet sur 1 bits.

$$\begin{array}{ccccc}
 & r_3 & r_2 & r_1 & r_0 = 0 \\
 & a_4 & a_3 & a_2 & a_1 \\
+ & b_4 & b_3 & b_2 & b_1 \\
\hline
 & r_4\ s_4 & r_3\ s_3 & r_2\ s_2 & r_1\ s_1
\end{array}$$

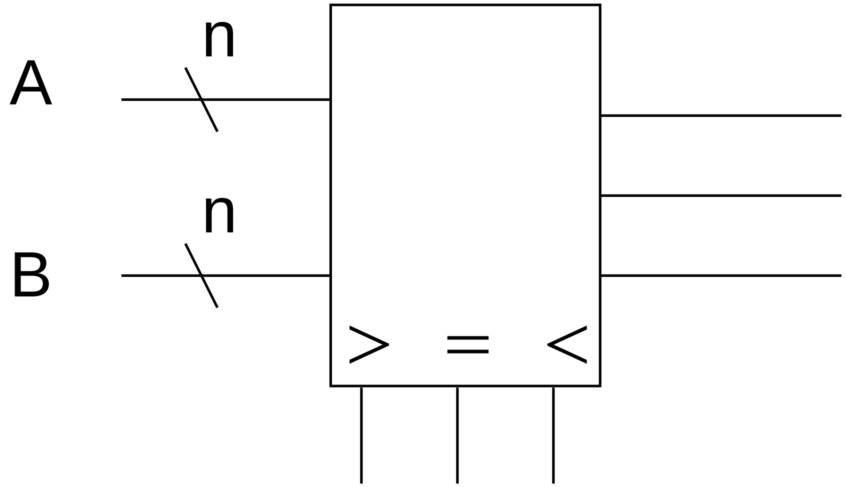
$$\boxed{r4 \quad s_4 \quad s_3 \quad s_2 \quad s_1}$$

Résultat final

Additionneur 4 bits (schéma)



Comparateur binaire



$S = 1 \text{ si } A > B$

$E = 1 \text{ si } A = B$

$I = 1 \text{ si } A < B$

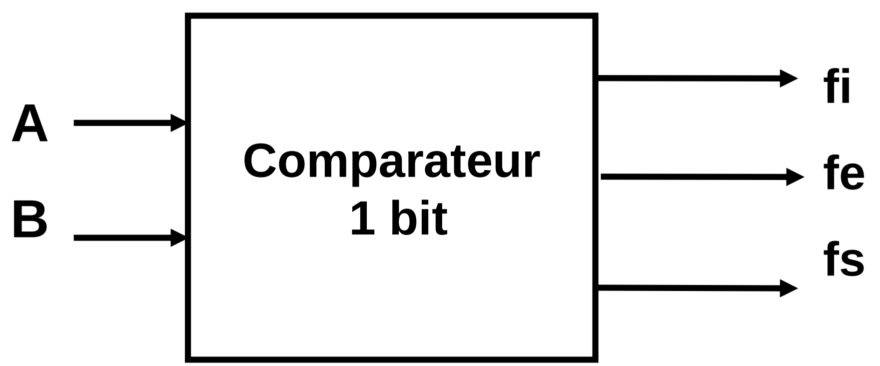
Entrées de cascadage

Pour une comparaison à n autres bits

Comparateur

- C'est un circuit combinatoire qui permet de comparer entre deux nombres binaire A et B.
- Il possède 2 entrées :
  - A : sur un bit
  - B : sur un bit

- Il possède 3 sorties
  - fe : égalité (A=B)
  - fi : inférieur (A < B)
  - fs : supérieur (A > B)



Comparateur sur un bit

A	B		fs	fe	fi
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

$$fs = A.\overline{B}$$

$$fi = \overline{A}B$$

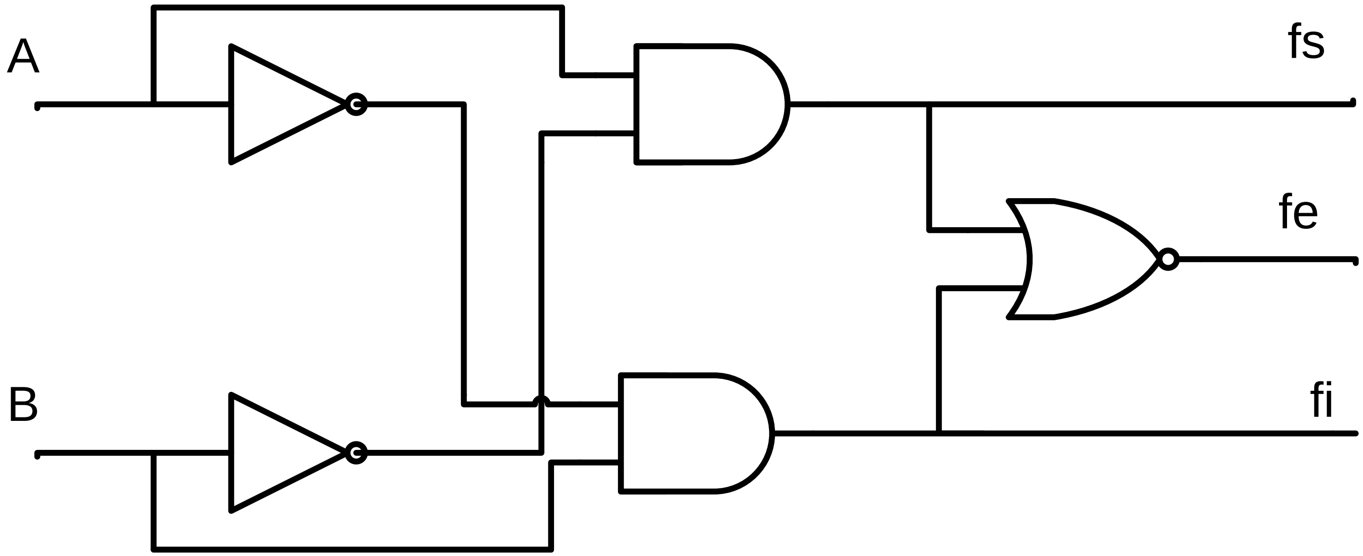
$$fe = \overline{A}\,\overline{B} + AB = \overline{A \oplus B} = \overline{fs + fi}$$

Schéma d'un comparateur dur un bit

$$fs = A.\overline{B}$$

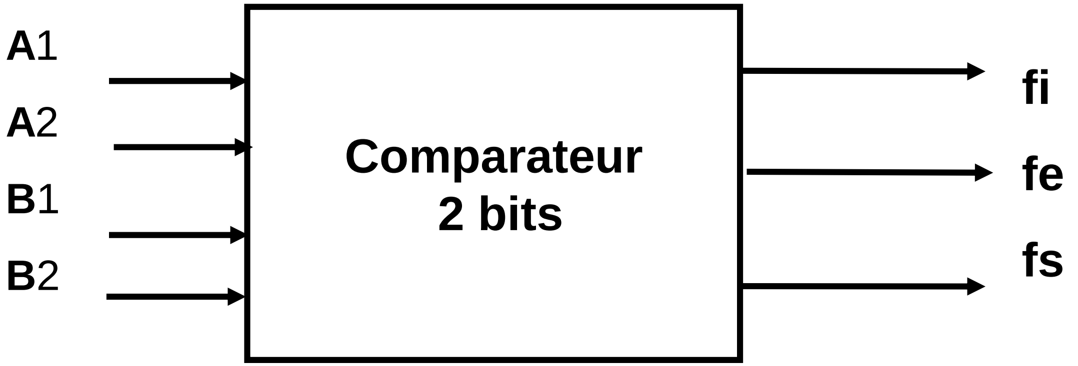
$$fi = \overline{A}B$$

$$fe = \overline{fs + fi}$$



Comparateur 2 bits

- Il permet de faire la comparaison entre deux nombres A (a_2a_1) et B(b_2b_1) chacun sur deux bits.



Comparateur 2 bits

A=B si **A2=B2** et **A1=B1**

$$fe = (\overline{A2 \oplus B2}).(\overline{A1 \oplus B1})$$

A>B si

A2 > B2 ou (A2=B2 et A1>B1)

$$fs = A2.\overline{B2} + (\overline{A2 \oplus B2}).(A1.\overline{B1})$$

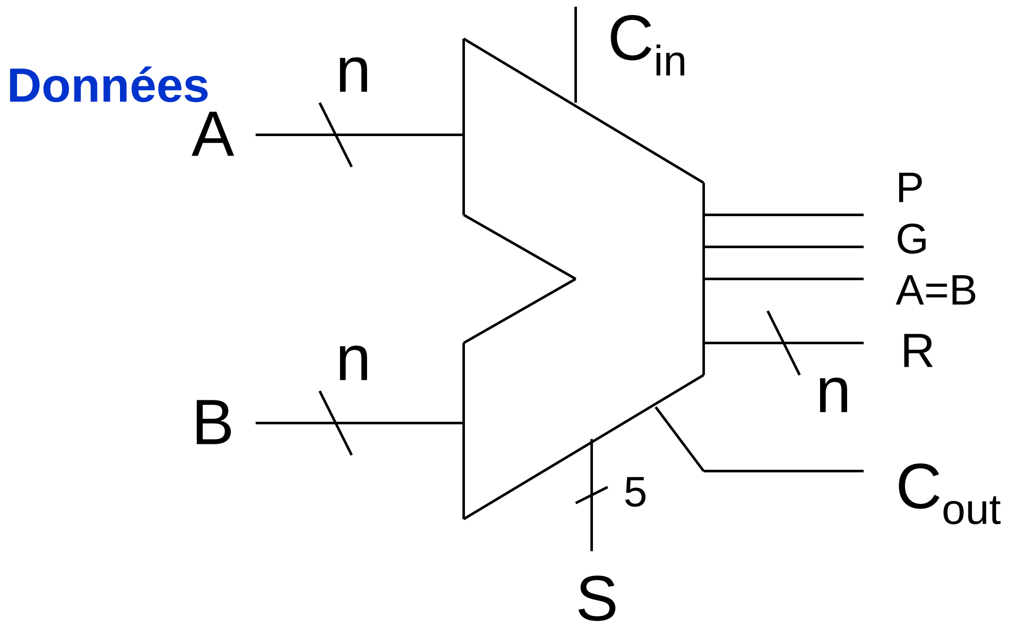
A<B si

A2 < B2 ou (A2=B2 et A1<B1)

$$fi = \overline{A2}.B2 + (\overline{A2 \oplus B2}).(\overline{A1}.B1)$$

A2	A1	B2	B1		fs	fe	fi
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	0	1	1		0	0	1
1	1	0	0		1	0	0
1	1	0	1		1	0	0
1	1	1	0		1	0	0
1	1	1	1		0	1	0

ALU (ou UAL)
Unité Arithmétique et Logique



Choix de la fonction (32 cas)

Instruction

Exemple :

Résultat

$R = A + \overline{B}$

$R = A + B$

$R = A + B + 1$

...

R = A ou B

R = A nand B

...

Plan

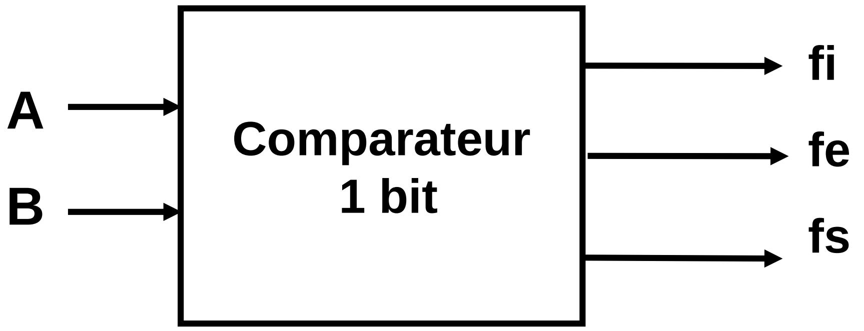
1) Calcul propositionnel, l'algèbre de Boole et la logique combinatoire

2) Circuits combinatoires

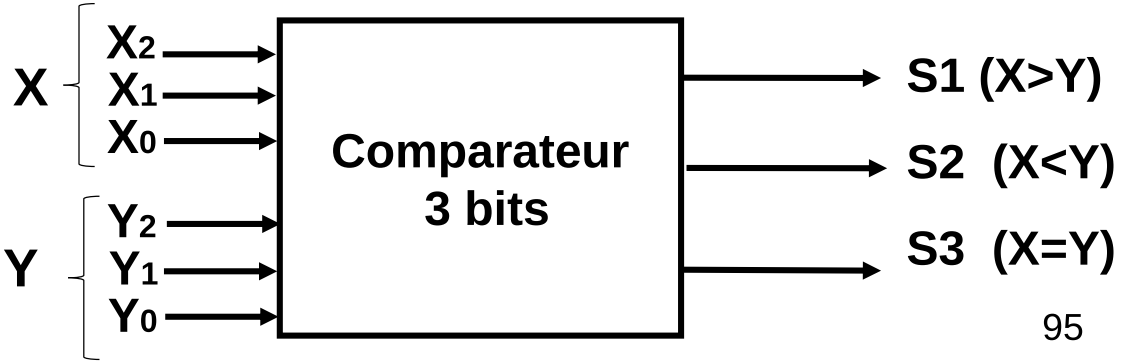
3) Représentation et codage des données

Comparateur

- C’est un circuit combinatoire qui permet de comparer entre deux nombres binaire A et B.
- Il possède 2 entrées :
  - A : sur un bit
  - B : sur un bit

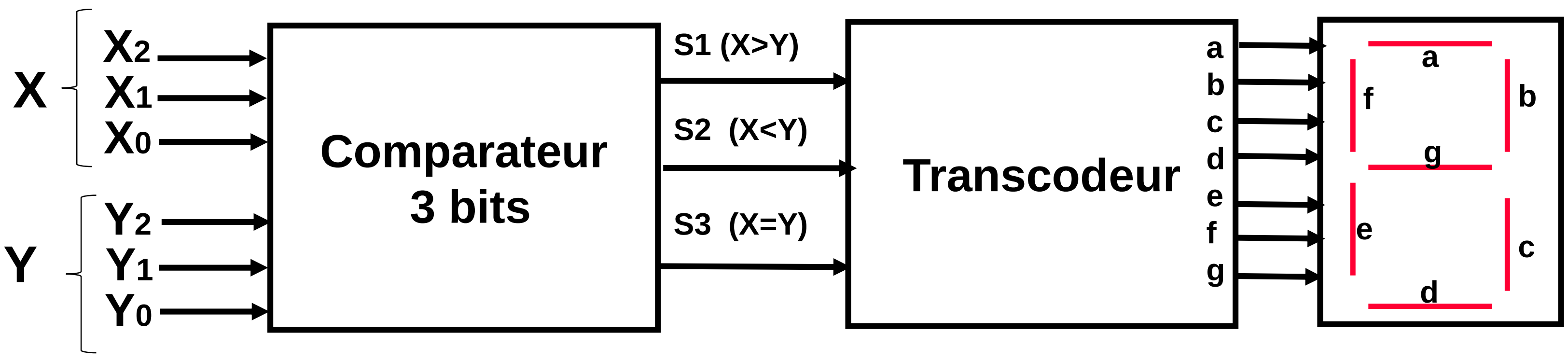


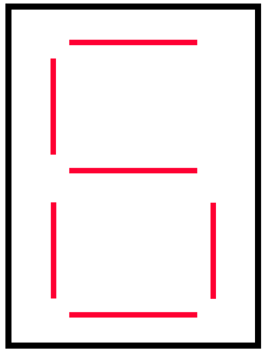
- Il possède 3 sorties
  - fe : égalité (A=B)
  - fi : inférieur (A < B)
  - fs : supérieur (A > B)



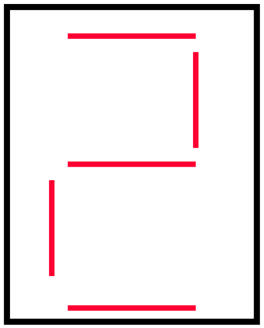
Comparateur

Exercice

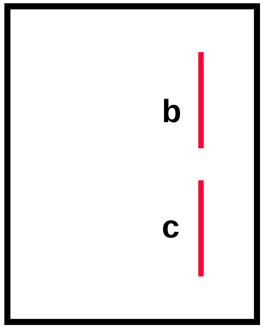




(Si X>Y)



(Si X=Y)



(Si X<Y)

Merci pour votre attention